实时音视频

多人会议（含 UI）

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 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

多人会议（含 UI）
组件介绍（TUIRoomKit）
开通服务（TUIRoomKit）
跑通 Demo（TUIRoomKit）
Web&H5 (Vue2/Vue3)
Android
iOS
uni-app (小程序)
uni-app (客户端)
Flutter
Electron (Windows&Mac)
快速接入（TUIRoomKit）
Web&H5 (Vue2/Vue3)
Android
iOS
uni-app (小程序)
uni-app (客户端)
WebView
Electron (Windows&Mac)
Flutter
屏幕共享（TUIRoomKit）
Android&iOS&Flutter
Web&Electron
uni-app (客户端)
预定会议（TUIRoomKit）
Android&iOS&Flutter
Web&Electron
Web（H5）&小程序
会中呼叫（TUIRoomKit）
Android&iOS
Web&Electron
小程序&H5
会议控制 (TUIRoomKit)
Android&iOS&Flutter
Web&Electron
小程序&H5
界面定制（TUIRoomKit）
Android
iOS
Web
Electron (Windows&Mac)
uni-app (小程序)
Flutter
自定义流布局（TUIRoomKit）
Web

版权所有：腾讯云计算（北京）有限责任公司

iOS
白板功能（TUIRoomKit）
Electron
云端录制 (TUIRoomKit)
机器人推流（TUIRoomKit）
AI 降噪（TUIRoomKit）
更多特性（TUIRoomKit）
AI 实时字幕
Android
Web
弹幕聊天（Android&iOS&Flutter）
房间密码（全平台）
基础美颜（Web）
设置昵称、头像（全平台）
会中聊天
Web&Electron
Android&iOS
Flutter
uniapp（客户端）
离线唤醒
会前提醒（Android/iOS）
会中呼叫提醒（Android/iOS）
悬浮窗（Android&iOS)
文字水印
Android&iOS
Web&Electron
虚拟背景
Web
监听会议状态
客户端 API（TUIRoomKit）
Web
Android
iOS&Mac
uni-app（小程序）
uni-app（客户端）
Flutter
Electron（Windows&Mac）
服务端 API（TUIRoomKit）
账号系统
账号管理
登录鉴权
生成 UserSig
资料管理
REST API
REST API 简介
REST API 列表
房间管理

创建房间
解散房间
更新房间信息
获取房间信息
用户管理
获取房间成员列表
更新房间成员资料
转让房主
设置房间成员标签
房间成员封禁
房间成员解封
获取被封禁房间成员列表
踢出房间成员
连麦管理
获取麦位列表
用户上麦
用户下麦
锁定麦位
机器人管理
添加机器人
删除机器人
第三方回调
第三方回调简介
回调命令列表
回调配置
查询回调配置
创建回调配置
更新回调配置
删除回调配置
房间相关
创建房间之后回调
解散房间后的回调
更新房间信息之后的回调
用户相关
用户进房之后的回调
用户退房之后的回调
连麦相关
麦位列表变更之后回调
发布日志（TUIRoomKit）
常见问题（TUIRoomKit）
iOS
Android
Web
Electron
小程序
Flutter
升级方案（TUIRoomKit）

2.0版本升级指引
错误码（TUIRoomKit）

版权所有：腾讯云计算（北京）有限责任公司

多人会议（含 UI）
组件介绍（TUIRoomKit）

最近更新时间：2025-02-25 15:06:02

组件介绍

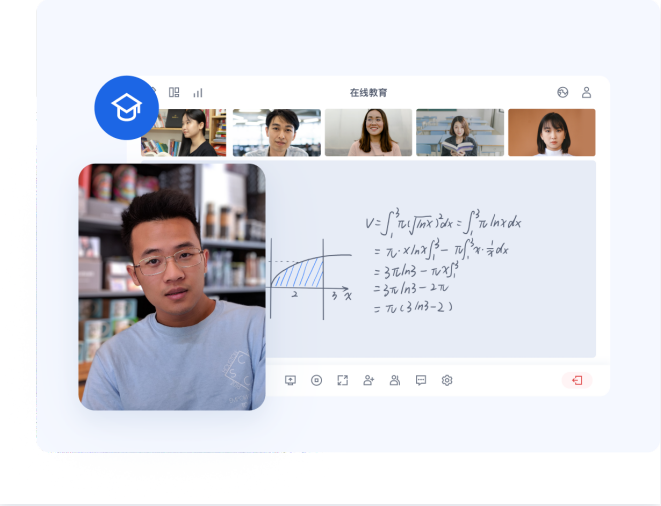
TUIRoomKit 是一款定位**企业会议、网络研讨会、在线教育** 等多人音视频会话场景的含 UI 低代码组件，支持小程序、 iOS、Android、Web、uni-app、Flutter 等平台全球互通。提供房间管理、成员管理、屏幕分享等会控功能，支持标清、高清、超高清等多种画质。通过集成该组件，仅需3步1天内就可以为您的 App 添加多人音视频会话功能，快速上线业务。

应用场景

应用场景	场景展示	功能说明

音视频会议		音视频会议应用快速搭建，提供标准会议UI，可满足音视频会议场景所需功能。 • 预定会议 • 屏幕分享 • 会中聊天 • 虚拟背景 • 上台发言 • 分享白板
远程医疗		远程医疗应用快速搭建，连接患者和医生，轻松实现远程问诊等场景。 • 低延迟实时互动 • 云端录制与回放 • 屏幕分享 • 虚拟背景 • 支持微信小程序、Android、Web 等多平台互通
企业协同		企业协同应用快速搭建，帮助企业实现在线培训和内容沉淀。 • 成员管理 • 会中聊天 • 云端录制与回放 • 上台发言 • 预约会议

版权所有：腾讯云计算（北京）有限责任公司

教育培训		教育培训应用快速搭建，支持1对1互动教学、职业/技能小班培训等多种教学场景。 • 房间管理 • 成员管理 • 屏幕分享 • 云端录制与回放 • 会中聊天 • 互动工具 • 分享白板

功能概览

功能说明：
TUIRoomKit 组件提供预定会议、音视频控制、屏幕共享、会议录制、成员管理、会议悬浮窗、会中聊天、会议水印等丰富的功能交互，可助力您快速上线多种业务场景。更多功能与计费说明参见 开通服务 。

基础功能

功能列表	功能描述
多人音视频（大规模参会人数、支持多人同时开启视频）	支持大规模人员参会，并支持多人同时开启视频。
标准会议 UI	含有标准会议所需要的UI组件，可满足音视频会议场景所需功能。
自定义 UI	可根据业务场景自定义UI。
预定会议 （web/Android/iOS/Flutter）	支持预定会议、并将预定会议显示在会议列表，提供丰富日程功能，
屏幕分享 （web/Android/iOS/Flutter）	支持屏幕共享，让会中成员同时观看共享屏幕中所展示的内容。
会议控制 （web/Android/iOS/Flutter）	分为会前控制和会中控制： • **会前**创建并加入会议时，可以通过预先设定会议的相关参数调整麦克风、摄像头、扬声器的打开情况，并决定**会议的发言模式**（自由发言模式/上台发言模式）。 • **会中**可以对所有成员进行管控，全体静音/禁视频、移出房间、转交主持人、邀请上台/下台等、仅房主/管理员可屏幕共享（全体禁止屏幕共享）。
会中聊天 （web/Android/iOS/Flutter）	参会者可以实时在聊天区发送消息，主持人或管理员也可以设置禁止参会者在会中发送消息。
云端录制	支持用户在视频会议、在线教育、直播互动等场景中进行录制，满足对重要内容进行存档、审核等多样化需求。
开启/关闭悬浮窗	允许用户通过一个可自由拖动的悬浮窗口继续参会，可以同时处理其他事宜。

高级功能

功能列表	功能描述
分享白板	支持用户开启白板进行绘制并分享至房间内，同时支持用户在屏幕分享时进行批注。
文字水印	支持用户在进行多人会议时设置文字水印。
虚拟背景（Web端）	让用户在进行多人会议时可以设置模糊背景，隐藏实际会议环境，保护隐私并增加会议趣味性。
机器人推流	允许用户将本地或在线媒体文件推流至房间内。在某些场景中，会议中可能需要推送预先录制的教学视频或在线视频流到在线课堂中，以便用户观看和学习。
同平台多端登录会议	支持同一平台，如多个 iOS 设备登录，来电后任一设备接通，自动停止其他设备的接入请求。
无序/有序麦位	支持会议管理员设置会议的发言排序模式。
请求发言	在会议类型为上台发言的会议中，成员可以向管理员发出发言请求。
自定义麦位数量	可以设置会中麦位数量。
麦位邀请/锁定/踢下	邀请会中成员发言，或者锁定/移出发言位。
AI 降噪	源自腾讯天籁实验室的 AI 算法，能够智能检测和去除混合在传播信号中的噪声干扰，显著提高语音的质量。
会中呼叫	用户在会议进行过程中，可以随时呼叫其他用户加入当前会议，无需提前预定或安排
弹幕聊天	支持发送弹幕消息、弹幕消息支持表情输入等，增加消息趣味性，让互动更愉悦。
房间密码	支持创建密码房间、预定密码房间。
设置昵称、头像	支持在会议过程中修改用户昵称。
会前提醒	当您成功预定会议后，TUIRoomKit 组件将在会议开始前10分钟向参会成员推送会前提醒消息。您也可以自定义会前提醒时间。
悬浮窗	允许用户创建一个可自由拖动的悬浮窗口，用于展示和管理 TUIRoomKit 的视频会议界面。

支持平台

平台	Android	iOS	Web	uni-app（小程序）	uni-app（客户端）	Flutter	Electron
是否支持	✓	✓	✓	✓	✓	✓	✓
支持语言/框架	• Java	• Swift	• Vue3 • Vue2	• Vue3	• Vue3	• Dart	• Vue3 • Vue2

在线体验

平台	Web	Android	iOS	其他平台
在线体验				\

示例代码	Github: web	Github: Andorid	Github: iOS	Github

热门场景体验

如果您的业务场景是 在线教育 场景，我们为您提供了更贴近于教育场景的方案，您可以体验到不同的 课堂类型，排课系统，课堂内容保护，课前提醒，AI 课堂记录，课中呼叫 等个性化功能。

TRTC 技术交流群

加入 腾讯实时音视频 TRTC 技术交流群，您将获得：

- 可靠的技术支持
- 详细的产品信息
- 紧密的行业交流

开通服务（TUIRoomKit）

最近更新时间：2025-07-03 12:02:32

本文介绍如何 免费开通体验版 、开通正式版 和 续费正式版 服务，您可以根据 多人音视频能力版本计费说明 选择您需要版本，并参考下述指引免费开通和购买开通正式版多人音视频能力。

开通体验版

TUIRoomKit 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。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）。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 实时音视频 TRTC 控制台，单击进入左侧应用管理页面，点击**创建应用**。在弹出的对话框中输入您的应用名称，选择**多人会议（TUIRoomKit）**场景，并单击**创建应用**。

注意：

若您创建时勾选免费领取7天体验版，即可快速创建应用时并领取体验版。

2. 单击刚创建的应用，进入应用管理页面，并在右侧找到**含 UI 低代码集成接入 > 多人会议（TUIRoomKit）**，单击**领取体验**即可开通 TUIRoomKit 的**7天免费试用服务**。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。这里的 `SDKAppID` 、`SDKSecretKey` 会在后续步骤中使用到。

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；
- 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；
- 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. 若您当前 SDKAppID 领取的体验版已到期，您可单击**版本详情**，在版本详情弹窗中单击**体验续期**直接使用第二次体验机会，此时体验版有效期将延长7天。

注意：

通过**体验续期**延长体验版有效期，将消耗1次免费体验机会。

开通正式版

多人音视频能力为 TRTC 组件能力之一，仅当 IM 应用为专业版或旗舰版时才可购买正式版音视频通话能力。您可参见 功能与计费说明 确认所需要使用的版本，前往 购买正式版 的音视频通话能力。

1. 登录 实时音视频购买页 。
2. 根据实际业务需求选择配置。接入方案选择**含 UI 低代码快速开发**，以及选择需要购买音视频通话能力的 SDKAppID，购买后无法修改。

3. 套餐类型选择**多人音视频**，并在套餐版本中选择所需版本即可，多人音视频套餐包能力详情可参考 功能与计费说明 。

4. 购买完成后，您可回到 实时音视频 TRTC 控制台 ，在**含 UI 低代码集成接入**功能区查看当前版本信息。开通完成后即可参见 集成指引 进行集成。

注意

- 购买 TUIRoomKit 相当于订阅对应版本的 TRTC 包月套餐，若您已经订阅过付费版的 TRTC 包月套餐，则无需重复加购音视频通话能力，仅需购买 IM 付费版后即可使用 TUIRoomKit。
- 若您在满足前述条件下多人音视频能力仍显示未开通，此时控制台上当前版本将显示**开通基础互动版**、**开通进阶互动版**或**开通超大房间互动版**按钮，点击后即可开通（此步骤免费）。

功能与计费说明

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 TUIRoomKit 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 音视频时长补充包，赠送时长包按月赠送，仅在当月生效。 **您需要购买指定套餐后方可使用 TUIRoomKit，**其中免费体验版的开通方式可参见 开通 TUIRoomKit 体验版 。

下表为 TUIRoomKit 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 TUIRoomKit。

- TRTC 尊享版/尊享版 Plus 配合 IM 专业版/旗舰版可使用 TUIRoomKit 的基础互动版。
- TRTC 旗舰版/旗舰版 Plus 配合 IM 专业版可使用 TUIRoomKit 的进阶互动版。
- TRTC 旗舰版/旗舰版 Plus 配合 IM 旗舰版可使用 TUIRoomKit 的超大房间互动版。

套餐包分为**含录制/转码/ AI 套餐**和**不含录制/转码/ AI 套餐**两种类型，请按需选购。

含录制/转码/ AI 套餐（New）

对比项		基础互动版	进阶互动版	超大房间互动版
价格		4374元/月 立即选购	9499元/月 立即选购	10999元/月 立即选购
包含套餐版本		TRTC 尊享版 Plus + IM 专业版	TRTC 旗舰版 Plus + IM 专业版	TRTC 旗舰版 Plus + IM 旗舰版
赠送资源 (超出后付费)	音视频时长	380,000分钟/月	1,400,000分钟/月	1,400,000分钟/月
	日活跃用户数（DAU）	10,000 DAU	10,000 DAU	10,000 DAU
	峰值群组数	100,000个/月	100,000个/月	100,000个/月

	录制转码时长（new）	260,000分钟/月	260,000分钟/月	600,000分钟/月
	AI 智能识别时长（new）	10,000分钟/月	10,000分钟/月	20,000分钟/月
基本功能	参会人数规模	50	200	300（如需扩容请 工单联系）
	同时开启视频人数	50	50	50
	视频清晰度设置	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	–	–	✓
	同平台多端登录会议	–	–	✓
麦位管理	无序/有序麦位	–	✓	✓

版权所有：腾讯云计算（北京）有限责任公司

	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 IM 功能		详见 IM 专业版功能	详见 IM 专业版功能	详见 IM 旗舰版功能
实时音视频 TRTC 功能		详见 TRTC 尊享版 Plus 功能	详见 TRTC 旗舰版 Plus 功能	详见 TRTC 旗舰版 Plus 功能
超出赠送资源部分后付费		• 音视频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 • 录制转码时长后付费价格详见 云端录制与投递计费说明 和 云端混流转码计费说明 • AI 智能识别时长后付费价格详见 AI 智能识别计费说明		
支持平台		iOS、Android、Flutter、Web、Windows、Electron、uni-app（小程序）		

不含录制/转码/ AI 套餐

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		0元/7天	3374元/月 立即选购	7749元/月 立即选购	9249元/月 立即选购
包含套餐版本		TRTC 体验版 + IM 任意版本	TRTC 尊享版 + IM 专业版	TRTC 旗舰版 + IM 专业版	TRTC 旗舰版 + IM 旗舰版
赠送资源 (超出后付费)	音视频时长	-	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	✓	–	–	✓
	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	✓	✓	✓	✓

版权所有：腾讯云计算（北京）有限责任公司

	AI 降噪	✓	✓	✓	✓
	音频默认混流	✓	✓	✓	✓
即时通信 IM 功能		可配合 IM 不同版本 试用	详见 IM 专业版/旗舰版功能	详见 IM 专业版功能	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 尊享版功能	详见 TRTC 旗舰版功能	详见 TRTC 旗舰版功能
超出赠送资源部分后付费		音视频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、Flutter、Web、Windows、Electron、uni-app（小程序）			

说明：

- **包含套餐版本：**TUIRoomKit 由 TRTC 和 IM 联合提供，使用 TUIRoomKit 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 TUIRoomKit 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。

续费正式版

您可对正式版应用进行续费，具体操作如下：

1. 登录登录 实时音视频 TRTC 控制台，单击左侧**应用管理**，选择您需要续期多人音视频（TUIRoomKit）能力的应用，进入应用管理页面。
2. 在页面的右下角找到**含 UI 低代码集成接入 > 多人音视频（TUIRoomKit）**，单击**续期**前往 实时音视频购买页 进行续费。套餐包价格说明请参见 多人音视频能力版本计费说明 。

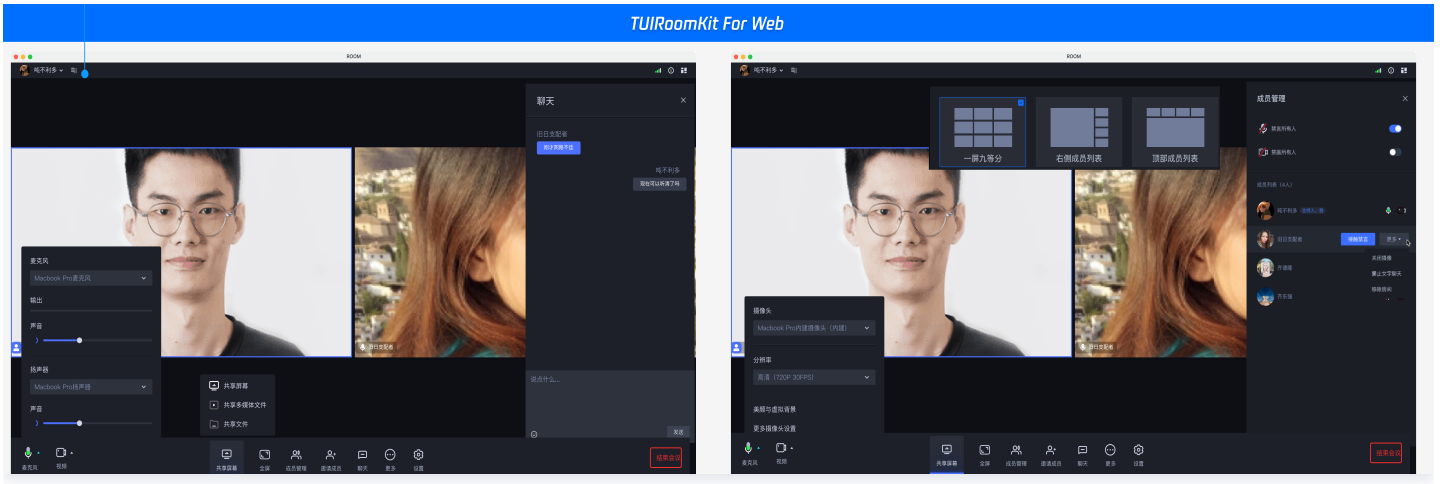
3. 根据已开通的版本进行续费。

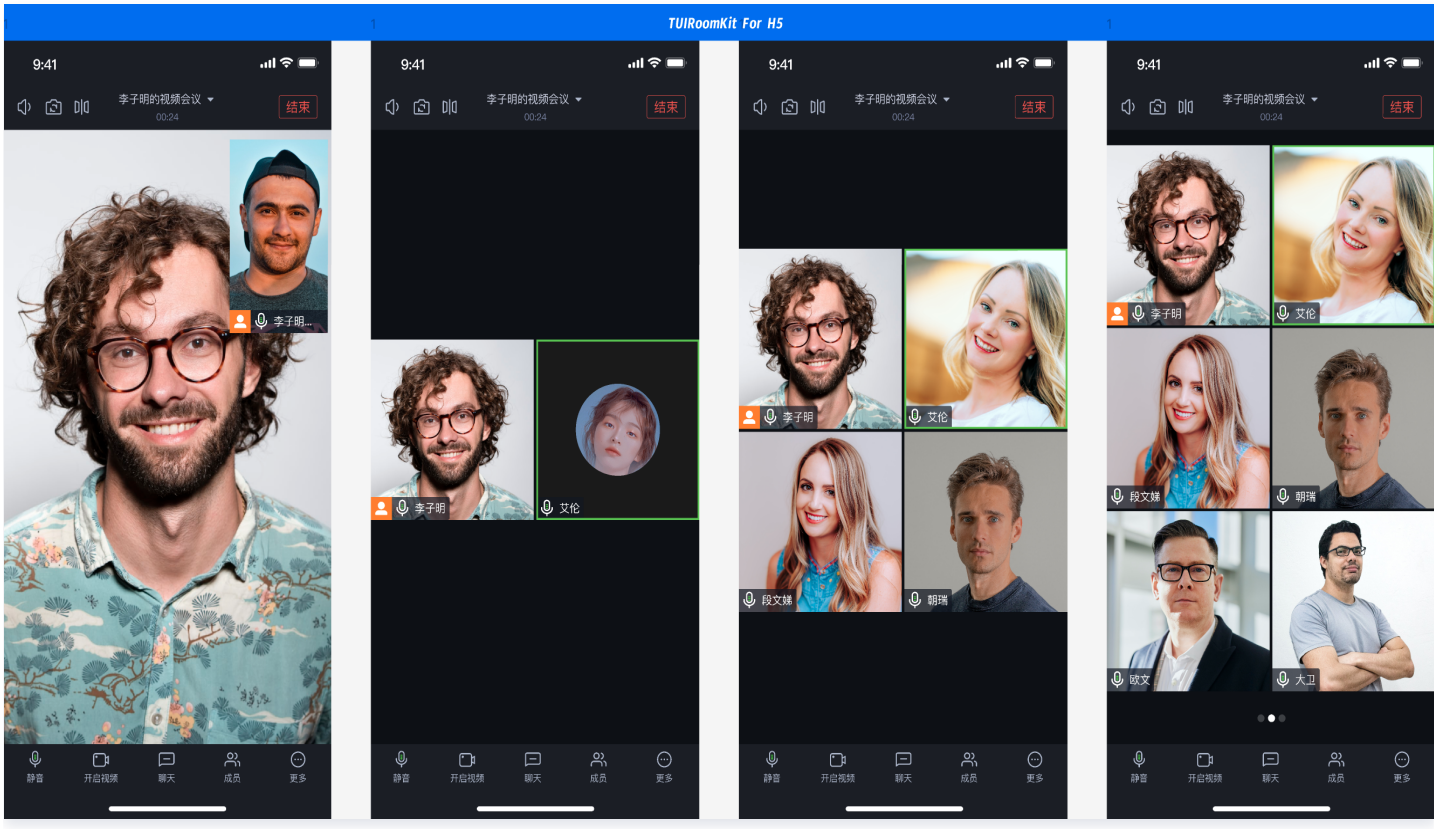
选择套餐
套餐类型
音视频通话（TUICallKit）
适用于社交聊天、医疗问诊、视频客服、视频面试等轻量音视频通话场景。查看 UI 示例
多人音视频（TUIRoomKit）
适用于企业会议、网络研讨会、在线教育、语音聊天室等多人音视频互动场景。查看 UI 示例
在线直播/语音聊天室（TUILiveKit）
适用于电商直播、语音聊天室、秀场直播、直播课等直播场景。查看 UI 示例
套餐版本
含录制/转码/AI权益套餐
不含录制/转码/AI权益套餐
根据已开通版本进行续费
对比项
基础互动版 续期
支持小程序，适用小型会议
进阶互动版
新增麦位管理，高性价比
超大房间互动版 升级
丰富功能，海量时长
套餐价格
套餐组合方式
TRTC 尊享版 Plus 已开通
IM 旗舰版 已开通
TRTC 旗舰版 Plus 未开通
IM 专业版 未开通
TRTC 旗舰版 Plus 未开通
IM 旗舰版 已开通
套餐组合单价
套餐赠送资源
音视频时长
38 万分钟/月
140 万分钟/月
140 万分钟/月
录制转码时长（new）
/
26 万分钟/月
60 万分钟/月
AI 智能识别时长（new）
/
1 万分钟/月
2 万分钟/月
会议规模
50 人
200 人
300 人（如需提升人数上限，请提交工单 联系我们）
同时开启视频人数
50 人
50 人
50 人
视频清晰度设置
最高 1080P
最高 4K
最高 4K
配置费用
共计2款商品
立即购买

跑通 Demo（TUIRoomKit）
Web&H5 (Vue2/Vue3)

最近更新时间：2025-02-11 11:43:02

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在 10 分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。





环境准备

- Node.js 版本：Node.js ⩾ 18（推荐使用官方 LTS 版本，npm 版本请与 node 版本匹配）。

- 现代浏览器，支持 WebRTC APIs。

下载 Demo

> **说明：**
> 如果您的业务偏向于**教育场景**，推荐您接入在线教育解决方案的 **Classroom** 应用。

1. 打开终端复制输入示例命令克隆仓库。

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

2. 安装依赖

Vue3

```
cd ./TUIRoomKit/Web/example/vite-vue3-ts
```

Vue2

```
cd ./TUIRoomKit/Web/example/webpack-vue2.7-ts
```

```
npm install
```

配置 Demo

1. 激活 TUIRoomKit 服务，获取 **SDKAppID** 和 **SDKSecretKey**。

2. 打开 TUIRoomKit/Web/example/vite-vue3-ts/src/config/basic-info-config.js 文件，并输入激活服务时获得的 SDKAppID 和 SDKSecretKey：

```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

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

/**
 * 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.
 *
 */

export const SDKAPPID = 0;

/**
 * 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
 *
 */
export const SDKSECRETKEY = '';
```

> **注意：**
> webpack-vue2.7-ts 项目请打开 **TUIRoomKit/Web/example/webpack-vue2.7-ts/src/config/basic-info-config.js** 文件，并输入激活服务时获得的 **SDKAppID** 和 **SDKSecretKey** 即可。

跑通 Demo

1. 在终端中输入命令，运行 Demo。

Vue3

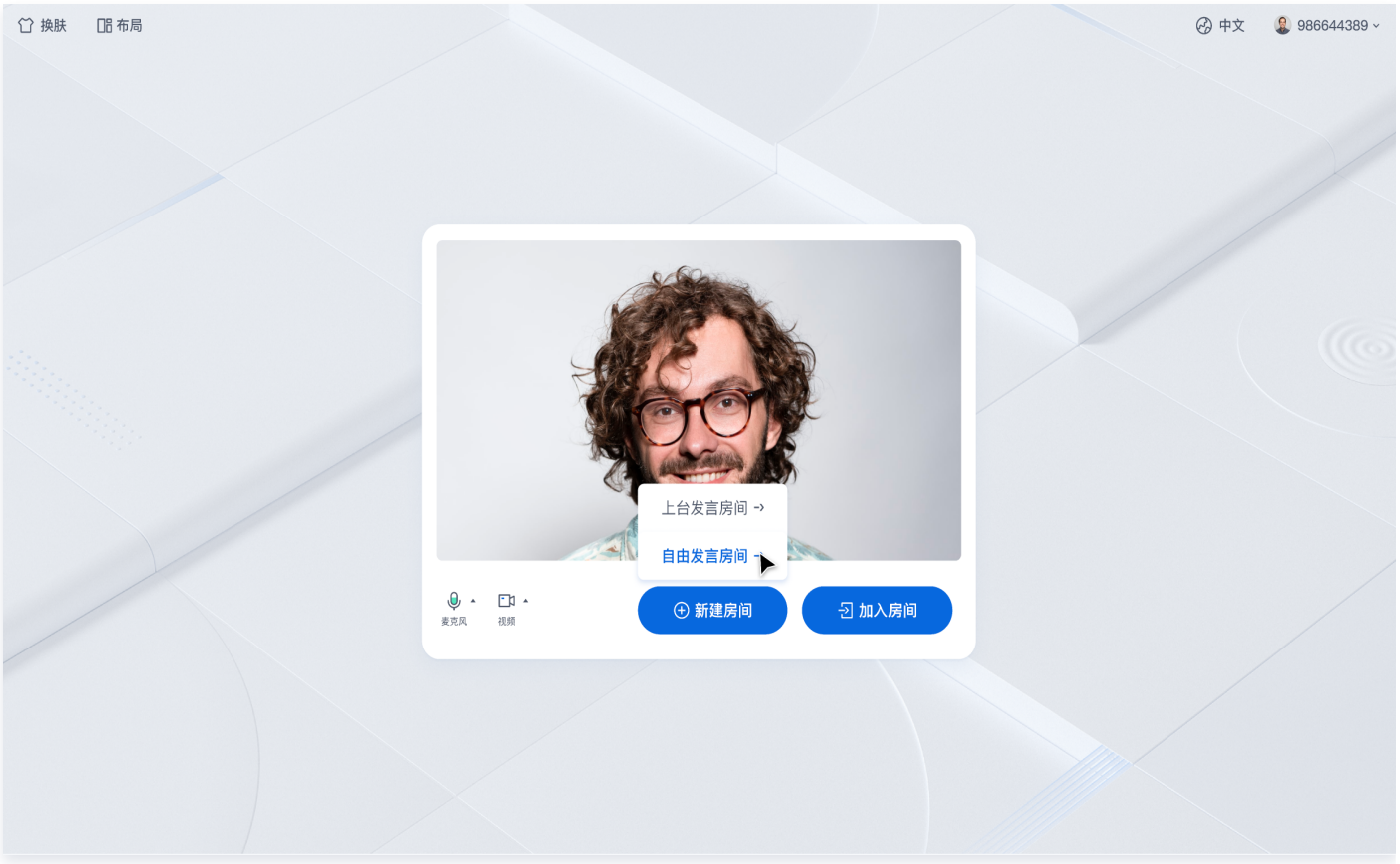
```
# cd TUIRoomKit/Web/example/vite-vue3-ts
npm run dev
```

Vue2

```
# cd TUIRoomKit/Web/example/webpack-vue2.7-ts
npm run serve
```

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

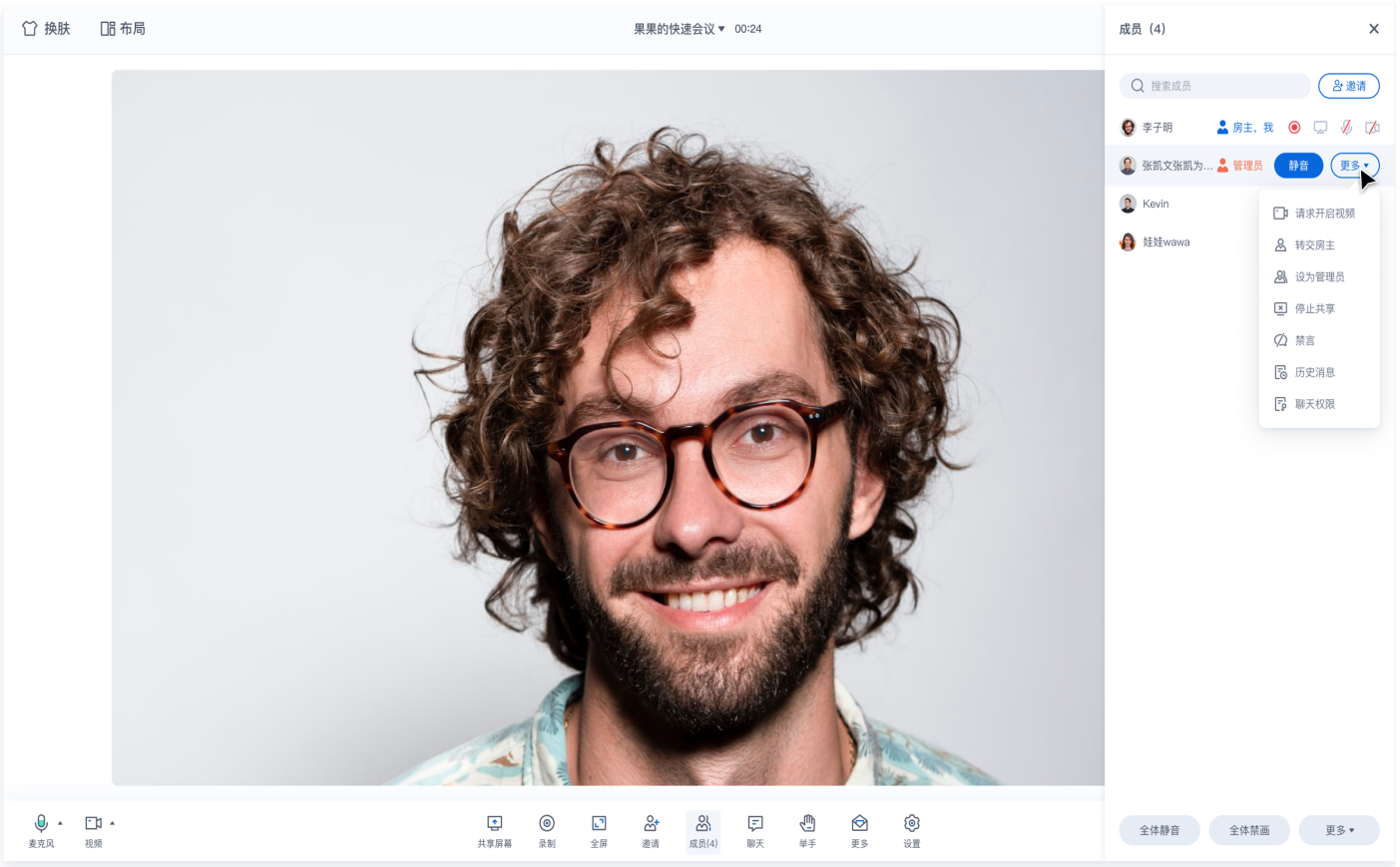
2. 打开浏览器页面输入相对应的 URL。



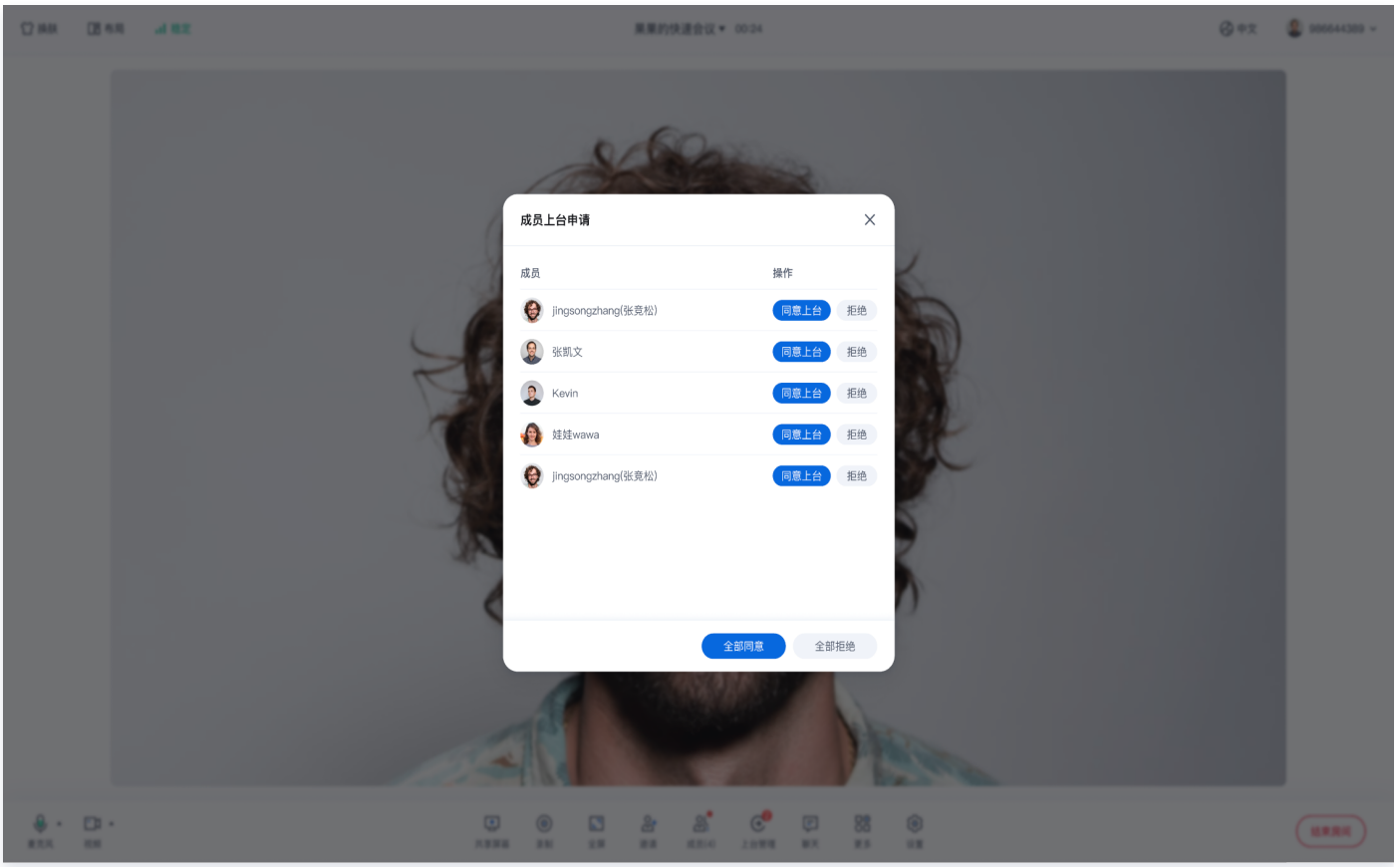
创建您的第一个会议

单击新建房间创建您的第一个会议房间，房间类型分别为举手发言房间和自由发言房间。

1. 自由发言房间



2. 举手发言房间



加入会议

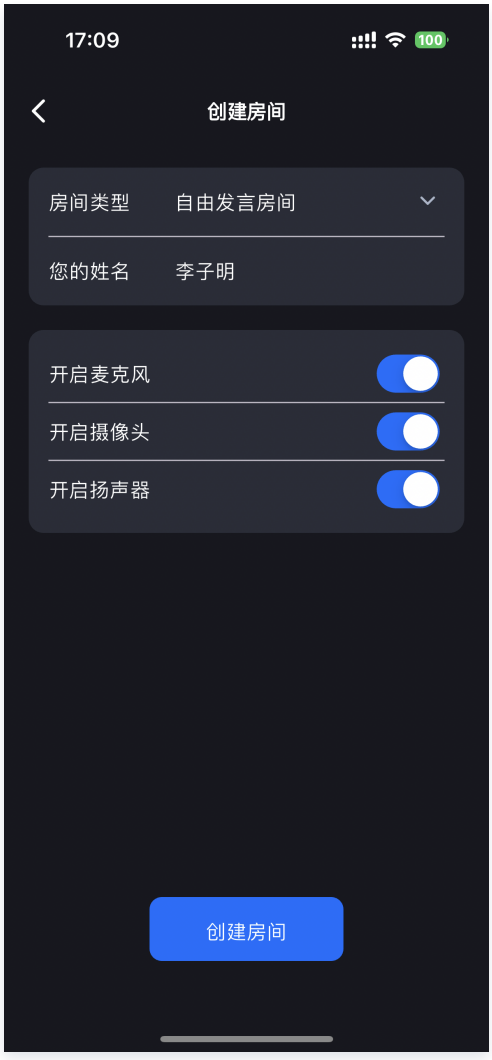
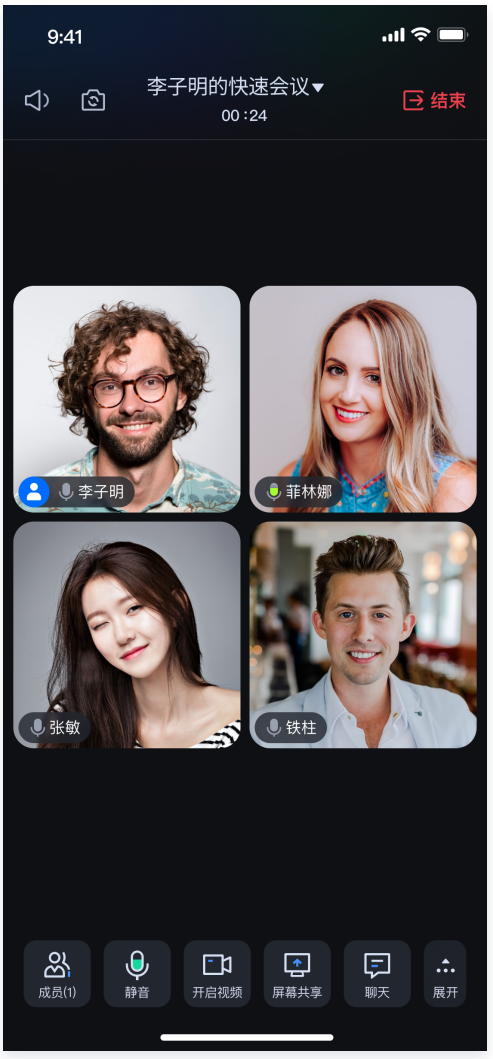
参与者可以通过填写对应的 `RoomId`，来加入由会议主持人创建的会议。

Hortensia的快速会议
00:32
房主
Hortensia
房间类型
自由发言房间
房间号
580216
复制
房间链接
https://web.sdk.qcloud.com...
复制
麦克风
摄像头
新建房间
加入房间
张竞松正在申请上台
同意
拒绝

Android

最近更新时间：2024-08-09 17:40:21

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在 10 分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。

创建会议界面	会议主界面
	

环境准备

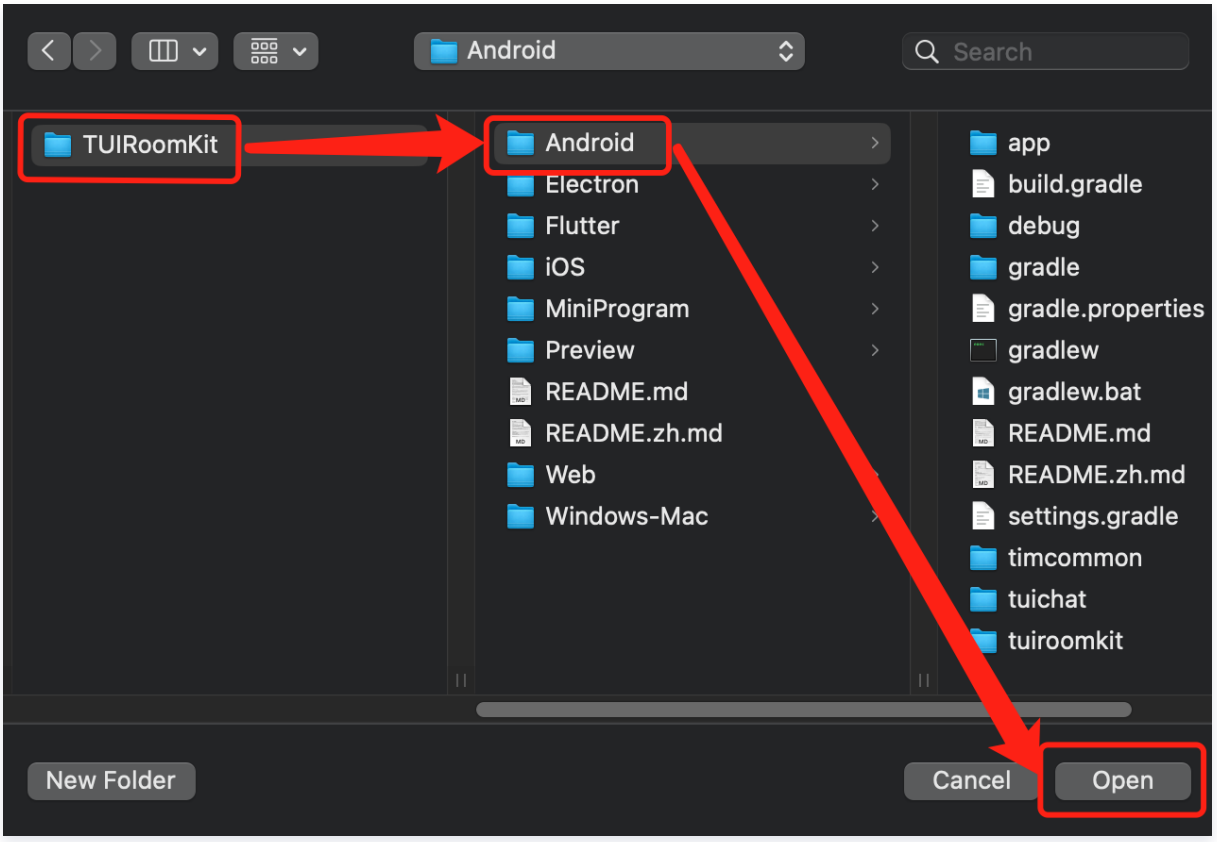
- 最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本。
- Android Studio 3.5及以上版本。

下载 Demo

1. 从 github 下载 TUIRoomKit Demo 源码，或者直接在命令行运行以下命令：

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

2. 通过 Android Studio 打开 **TUIRoomKit Android** 项目：



配置 Demo

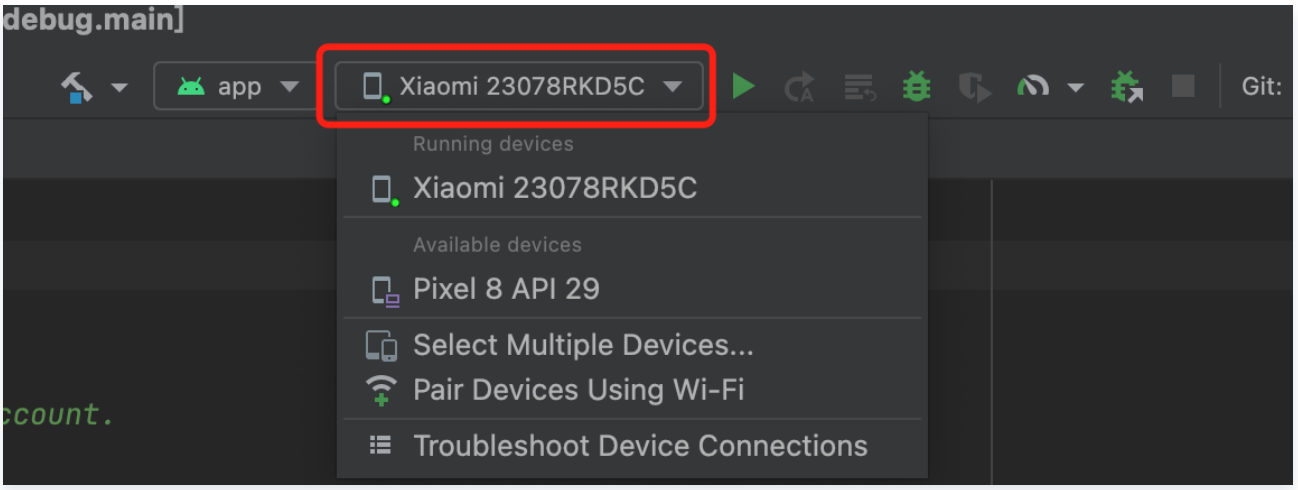
1. 开通音视频服务，获取 SDKAppID 和 SDKSecretKey。

2. 打开工程，在工程内找到 `Android/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。
将上一步中获取到的对应的 SDKAppID 和 SDKSecretKey 填入其中：

```java
public class GenerateTestUserSig {

    /**
     * Tencent Cloud SDKAppID. Set it to the SDKAppID of your account.
     * <p>
     * You can view your `SDKAppId` after creating an application in the [TRTC console](https://console.trtc.io/).
     * SDKAppID uniquely identifies a Tencent Cloud account.
     */
    2 usages
    public static final int SDKAppID = PLACEHOLDER;


    /**
     * Signature validity period, which should not be set too short
     * <p>
     * Unit: Second
     * Default value: 604800 (seven days)
     */
    1 usage
    private static final int EXPIRE_TIME = 604800;

    /**
     * Follow the steps below to obtain the key required for UserSig calculation.
     * <p>
     * Step 1. Log in to the [TRTC console](https://console.trtc.io/). If you don't have an application yet, create one.
     * Step 2. Click your application and find "Basic Information".
     * Step 3. Click "Display Key" to view the key used for UserSig calculation.
     * Copy and paste the key to the variable below.
     * <p>
     * Note: This method is for testing only. Before commercial launch,
     * please migrate the UserSig calculation code and key to your backend server
     * to prevent key disclosure and traffic stealing.
     * Documentation: https://trtc.io/document/35166
     */
    1 usage
    private static final String SDKSecretKey = "PLACEHOLDER";
```

跑通 Demo

1. 在 Android Studio 右上角如下图所示处选择您要将 Demo 运行的设备：



2. 选择完成后单击运行，将 TUIRoomKit Android Demo 运行到目标设备上。

APP主界面	创建会议界面

创建您的第一个会议

单击新建房间创建您的第一个会议房间，房间类型分别为自由发言房间和举手发言房间。

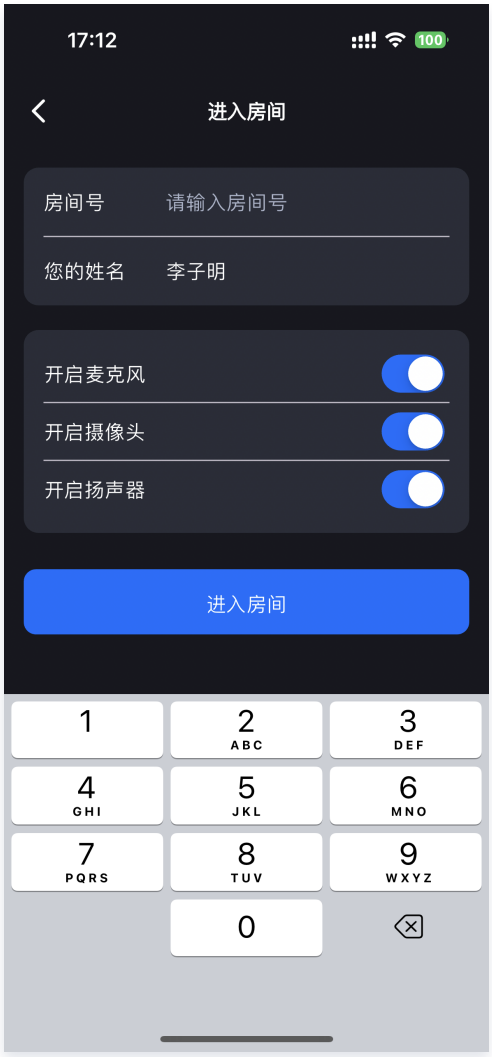
- **自由发言房间：普通用户可以自由进行发言，且自由开关麦克风和摄像头。**
- **上台发言房间：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。**

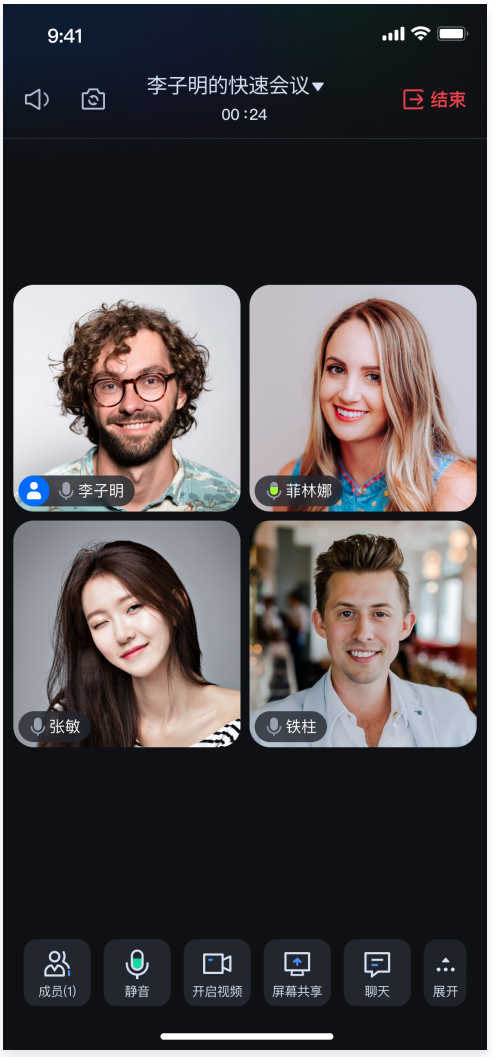
自由发言房间成员管理界面	上台发言房间成员列表

加入会议

单击加入房间后，参与者可以通过填写对应的 `RoomId` ，来加入由会议主持人创建的会议。

加入会议界面	会中界面

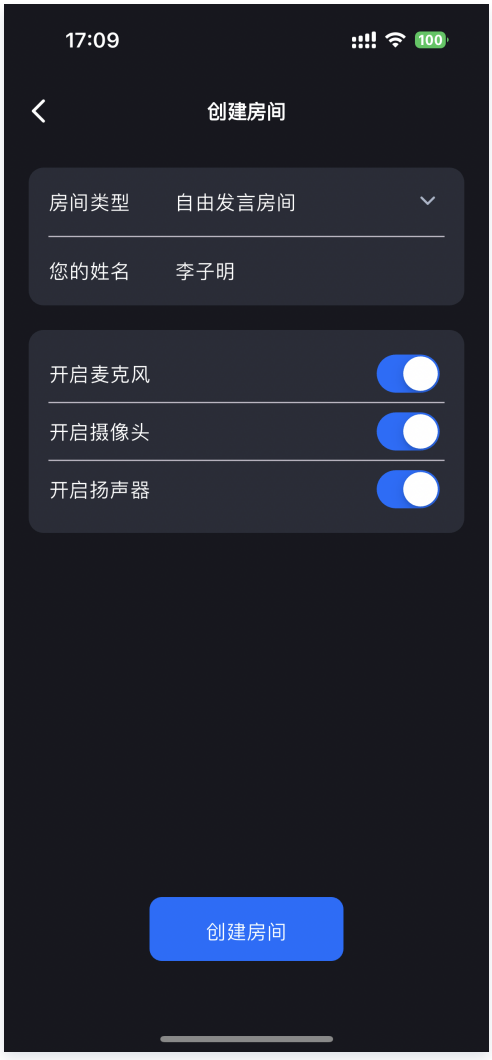
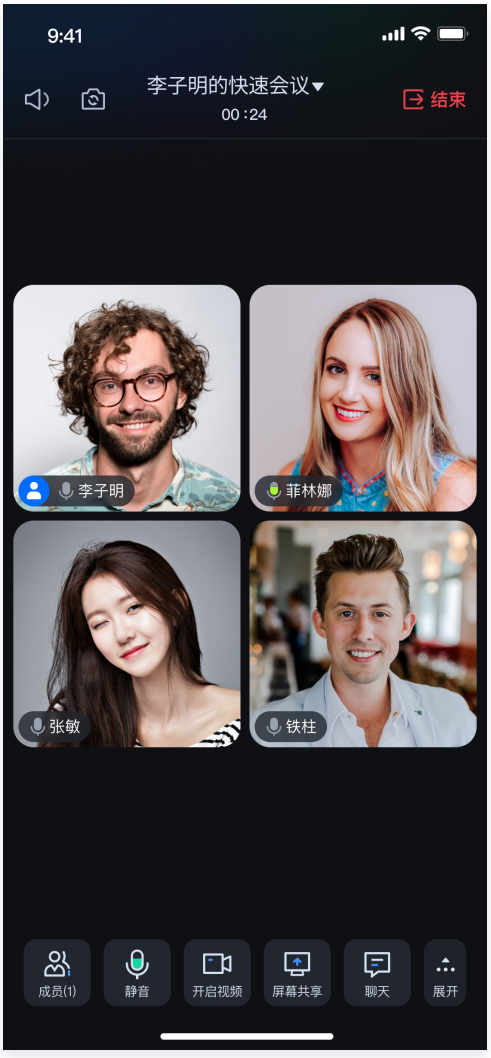

17:12
进入房间
房间号
请输入房间号
您的姓名
李子明
开启麦克风
开启摄像头
开启扬声器
进入房间
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0


9:41
李子明的快速会议
00:24
结束
李子明
菲林娜
张敏
铁柱
成员(1)
静音
开启视频
屏幕共享
聊天
展开

iOS

最近更新时间：2024-09-20 21:28:31

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在10分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。

创建会议界面	会议主界面
	

环境准备

iOS 13.0 及更高。

下载 Demo

1. 下载源码。
   - 方式一： 从 github 下载 TUIRoomKit Demo 源码。
   - 方式二： 从 gitee 下载 TUIRoomKit Demo 源码。
   - 方式三： 直接在命令行运行以下命令：

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

2. 在命令行中进入 iOS 项目目录：

```
cd TUIRoomKit/iOS/Example
```

3. 加载依赖库：

```
pod install
```

> **注意：**
> - 如果您尚未安装 CocoaPods ，可以参见 此处 了解如何安装。
> - 如果无法安装 TUIRoomKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表，然后再次安装：
>
> ```
> pod repo update
> ```

配置 Demo

1. 开通 TUIRoomKit 服务，获取 SDKAppID 和 SDKSecretKey。

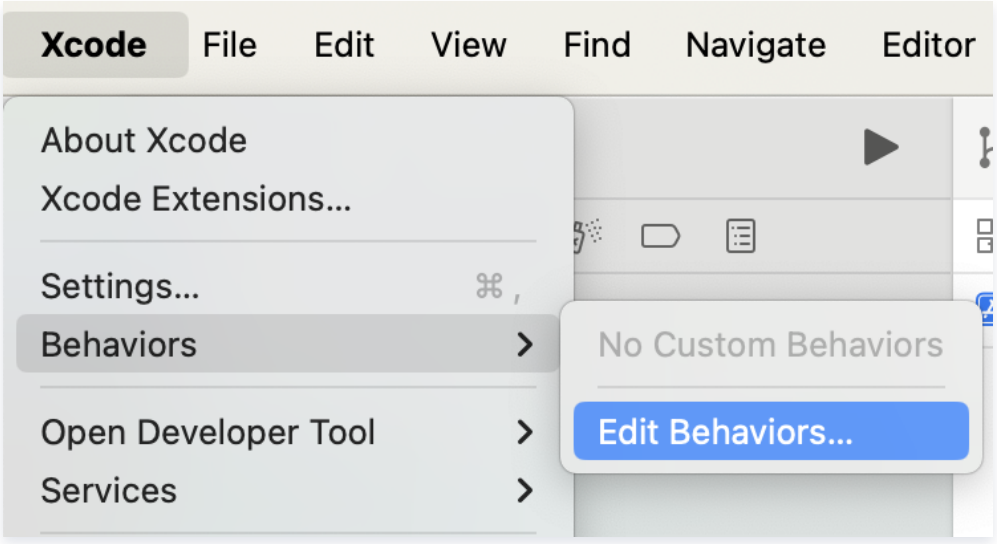
2. 点击 `iOS/Example/DemoApp.xcworkspace` 打开工程，在工程内找到 `iOS/Example/Debug/GenerateTestUserSig.swift` 文件，将上一步中获取到的对应的 SDKAppID 和 SDKSecretKey 填入其中：

```
/**
 * Tencent Cloud SDKAppID. Set it to the SDKAppID of your account.
 * You can view your `SDKAppID` after creating an application in the [TRTC console](https://console.trtc.io/).
 * SDKAppID uniquely identifies a Tencent Cloud account.
 */
let SDKAppID: Int = 0

/**
 * Signature validity period, which should not be set too short
 * Time unit: Second
 * Default value: 604800 (7 days)
 */
let EXPIRETIME: Int = 604_800

/**
 * Follow the steps below to obtain the key required for UserSig calculation.
 * Step 1. Log in to the [TRTC console](https://console.trtc.io/). If you don't have an application yet, create one.
 * Step 2. Click your application and find "Basic Information".
 * Step 3. Click "Display Key" to view the key used for UserSig calculation. Copy and paste the key to the variable below.
 * Note: This method is for testing only. Before commercial launch,
 * please migrate the UserSig calculation code and key to your backend server to prevent key disclosure and traffic stealing.
 * Documentation: https://trtc.io/document/35166
 */
let SDKSecretKey = ""

```

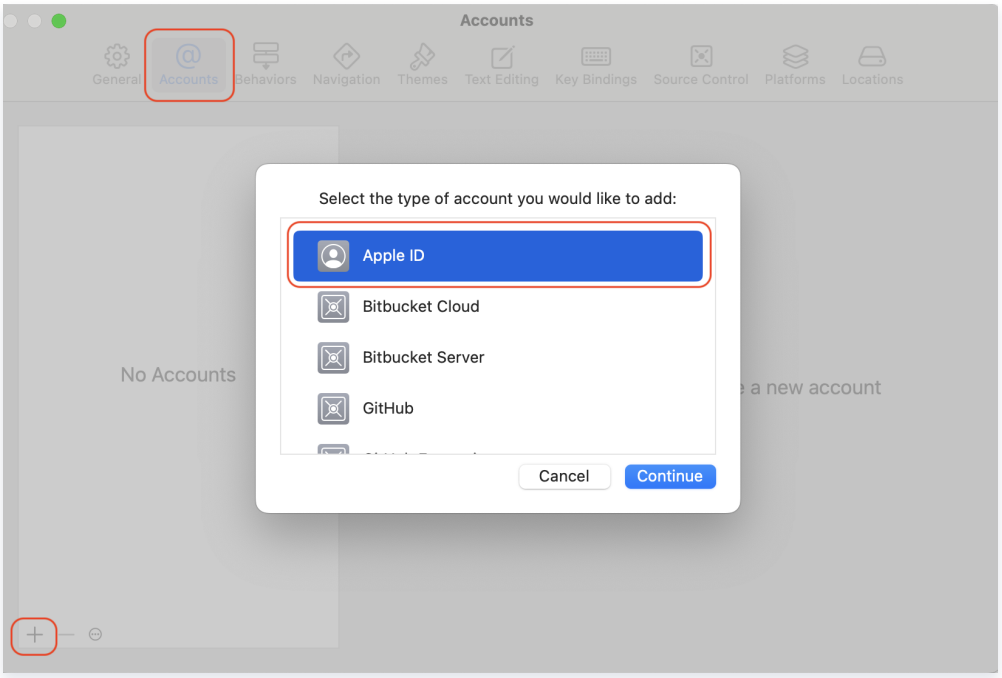
跑通 Demo

1. 生成您自己的证书（可选，如果已经有证书请忽略该步骤）

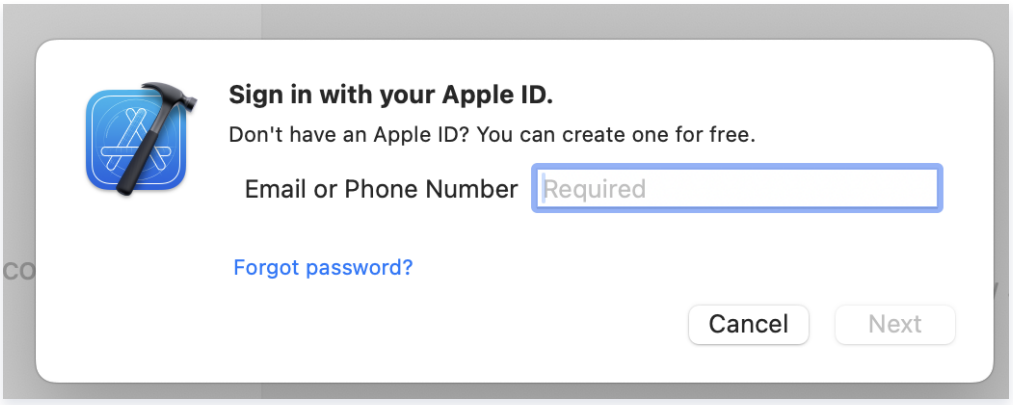
1.1 点击 Xcode 并找到 Edit Behaviors。



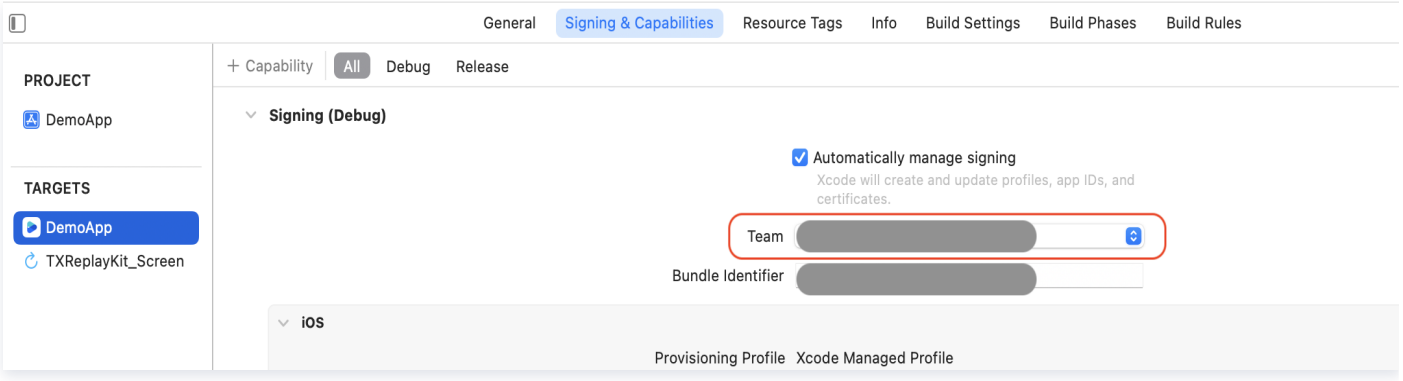
1.2 单击 Account 选项卡，然后单击左下角的 + 号，选择添加 Apple ID 并单击 Continue。



1.3 输入您的 Apple ID 和密码进行登录。

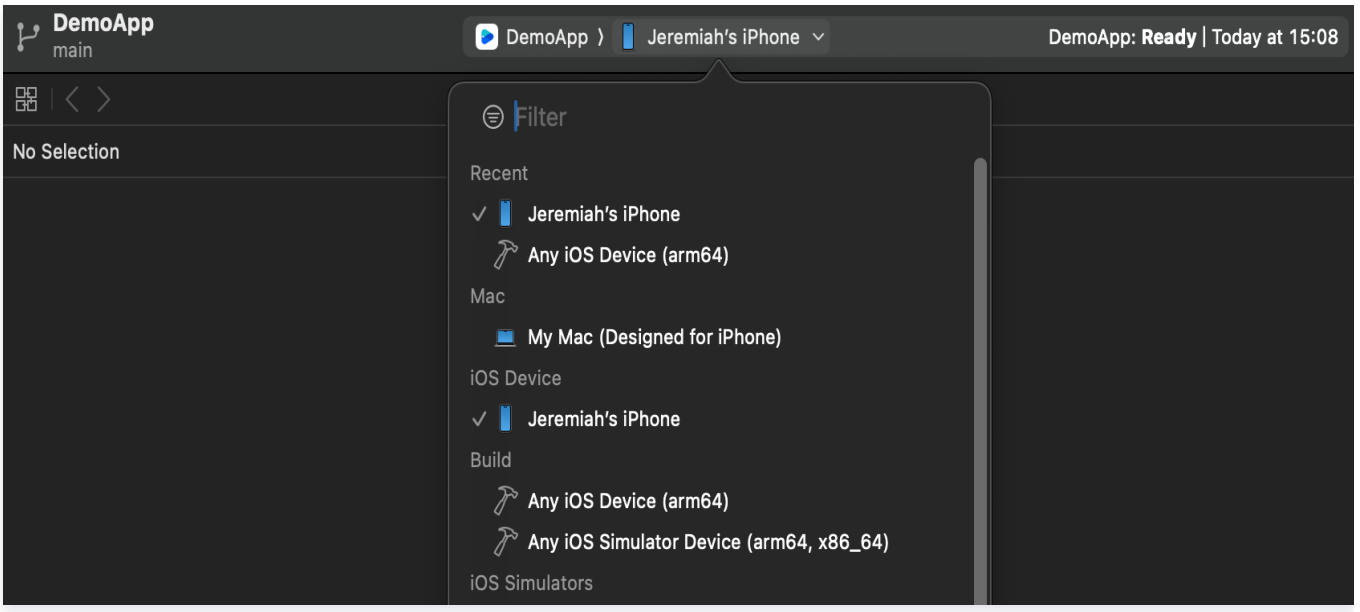


2. 点击进入项目 **TARGETS** 下的 **Signing & Capabilities** 选项卡，在 **Team** 中填入自己的开发者证书。



3. 运行项目

3.1 请先打开 iOS 设备的开发者模式，通过点击**设置 > 隐私与安全性 > 开发者模式**进行开启。将设备连接上电脑，并在 XCode 上选择运行 Demo 的设备。



3.2 单击运行，即可将我们的 TUIRoomKit iOS Demo 运行到目标设备上。

APP主界面	创建会议界面

说明：
此项目也支持在模拟器上运行，只要在选择设备处直接选择模拟器的型号即可。

创建您的第一个会议

单击**新建房间**创建您的第一个会议房间，房间类型分别为**自由发言房间**和**举手发言房间**。

- **自由发言房间**：普通用户可以自由进行发言，且自由开关麦克风和摄像头。
- **上台发言房间**：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。

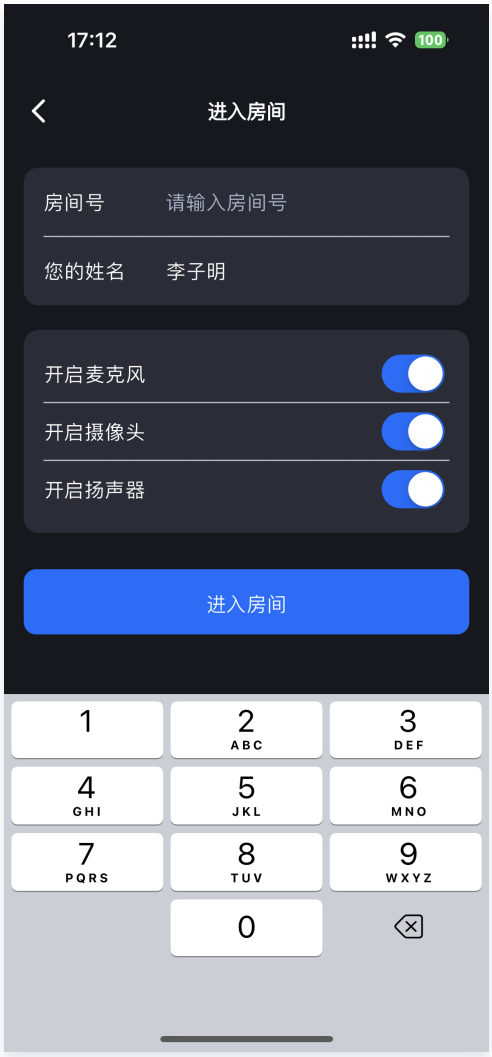
自由发言房间成员管理界面	上台发言房间成员列表

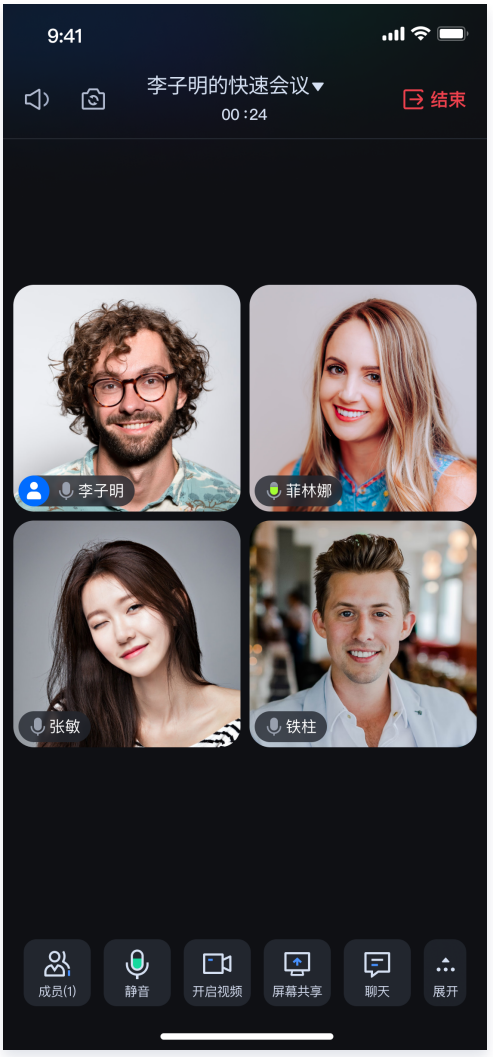
版权所有：腾讯云计算（北京）有限责任公司

加入会议

单击加入房间后，参与者可以通过填写对应的 `RoomId` ，来加入由会议主持人创建的会议。

加入会议界面	会中界面





注意：

如果想用不同手机体验音视频互通场景，请确保在 `iOS/Example/Debug/GenerateTestUserSig.swift` 文件中填写的 SDKAppID 是一致的。

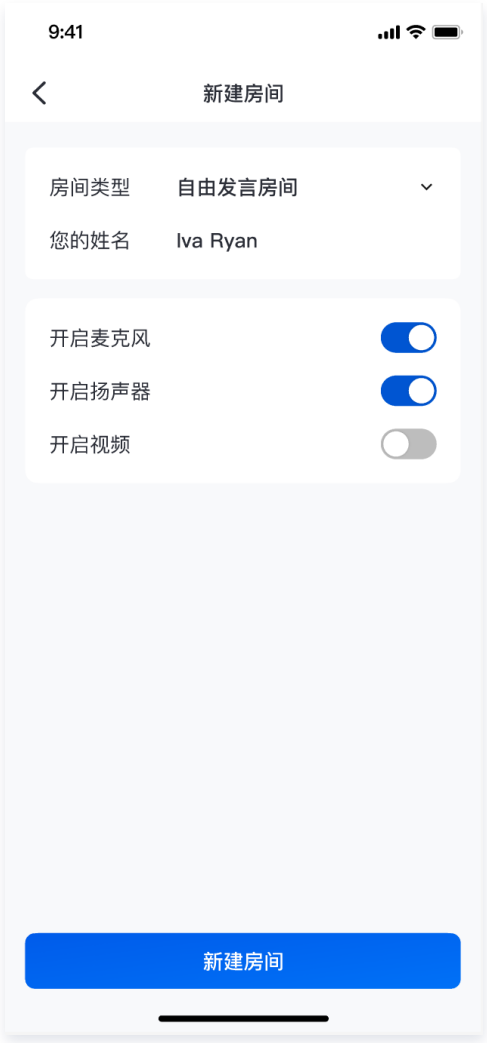
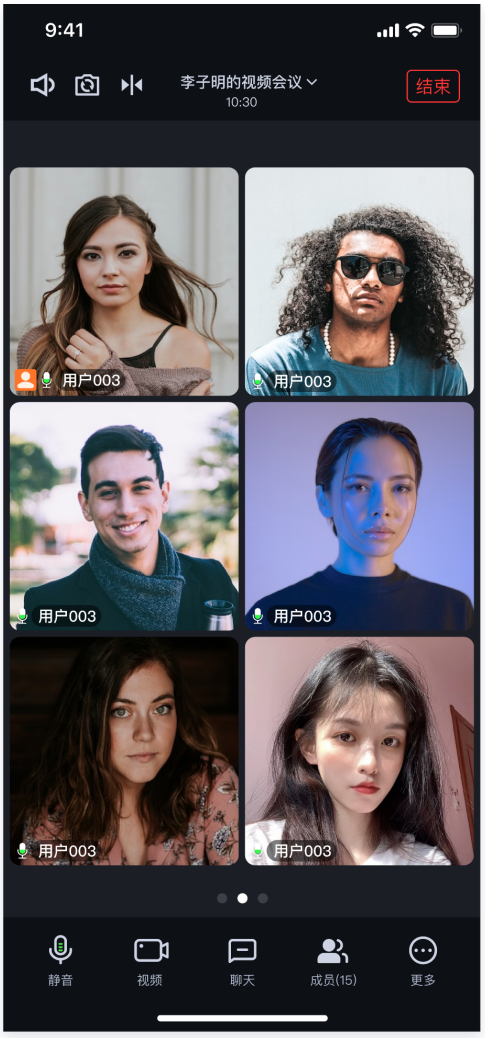
交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

uni-app (小程序)

最近更新时间：2024-12-27 18:23:33

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在10分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。

创建会议界面	会议主界面
	

环境准备

- 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
- 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（参见：小程序权限申请）
- 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。
- 配置项目合法域名（参见：小程序控制台配置合法域名）

快速跑通

1. 下载或者克隆源码并安装依赖。

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

2. 安装依赖

```
cd ./TUIRoomKit/MiniProgram
```

```
npm install
```

3. 开发环境运行示例工程

```
npm run dev:mp-weixin
```

4. 进入小程序目录安装项目依赖

> **注意：**
> 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 脚本中，因此可以在编译完成后直接执行 bash wxmini_dev.sh 安装依赖。

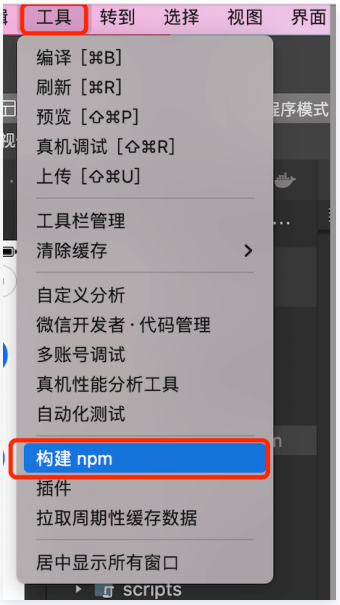
Windows：

```
# npm run dev:mp-weixin 编译完成后执行该指令
./wxmini_dev.bat
```

Mac：

```
# npm run dev:mp-weixin 编译完成后执行该指令
bash wxmini_dev.sh
```

5. 打开微信开发者工具，导入项目 **./MiniProgram/dist/dev/mp-weixin**。
6. 构建 npm，微信开发者工具中单击**工具 > 构建 npm**。具体如下图：



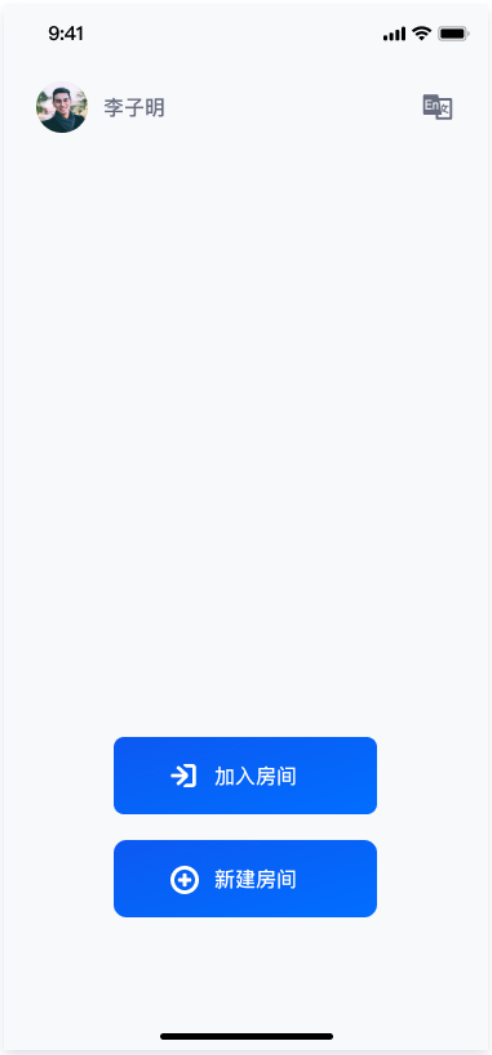
7. **找到并打开 MiniProgram/src/roomkit/config/basic-info-config.js 文件，配置** basic-info-config.js 文件中的相关参数（获取 SDKAppID 与 SDKSECRETKEY）：

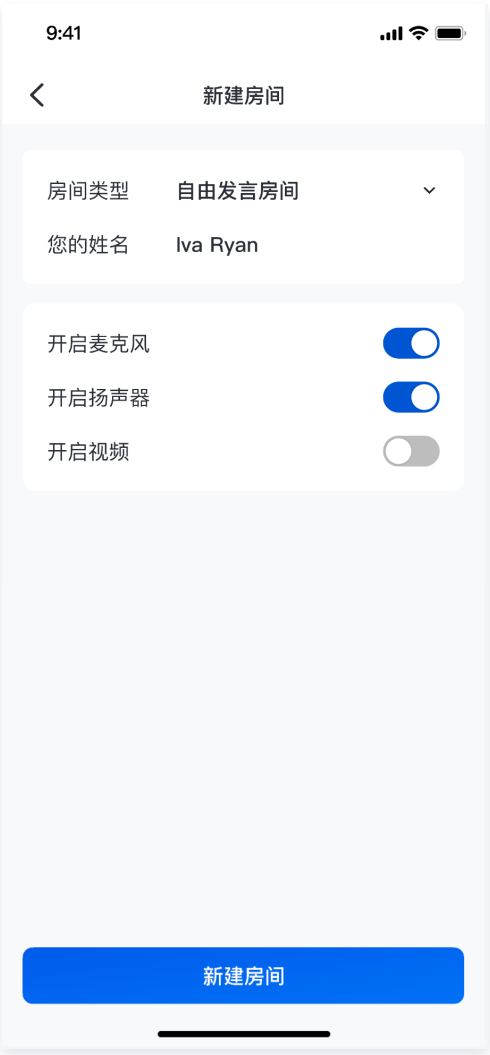
```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

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

/**
 * 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.
 *
 */

export const SDKAPPID = 0;

/**
 * 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
 *
 */
export const SDKSECRETKEY = '';
```

> **说明：**
> 激活 TUIRoomKit 服务，获取 **SDKAppID** 和 **SDKSecretKey**。

8. 单击 预览 手机扫码进入小程序。

APP主界面	创建会议界面





创建您的第一个会议

单击**新建房间**创建您的第一个会议房间，房间类型分别为**自由发言房间**和**举手发言房间**。

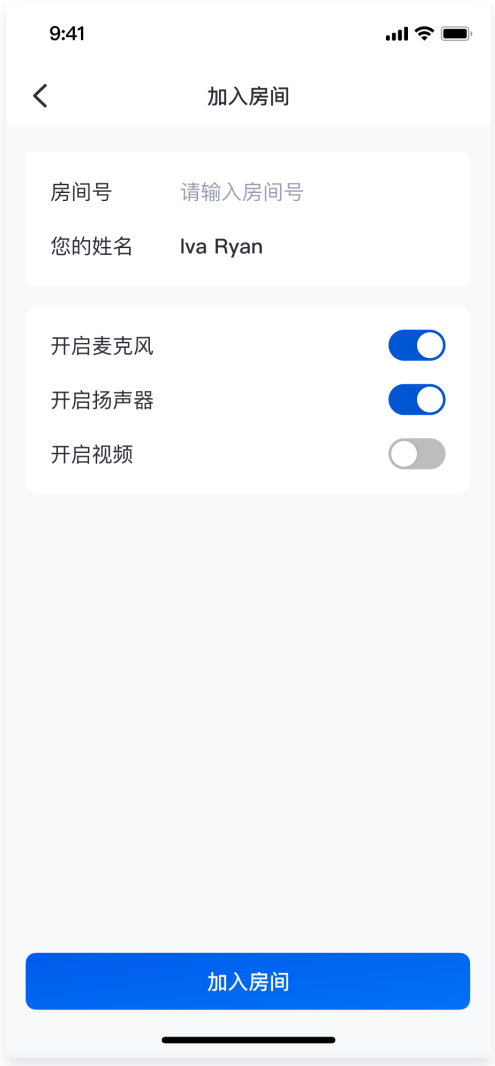
- **自由发言房间**：普通用户可以自由进行发言，且自由开关麦克风和摄像头。
- **上台发言房间**：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。

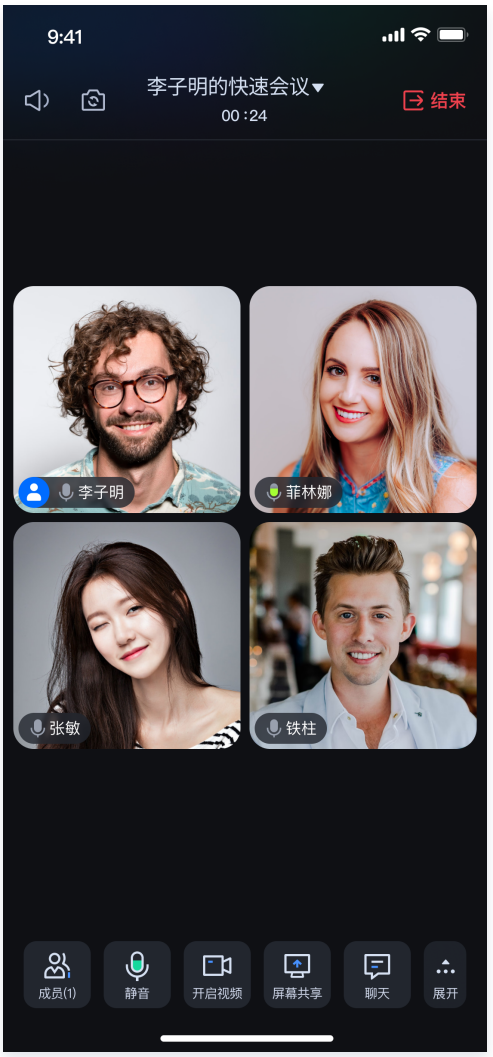
自由发言房间成员管理界面	上台发言房间成员列表

加入会议

单击加入房间后，参与者可以通过填写对应的 `RoomId` ，来加入由会议主持人创建的会议。

加入会议界面	会中界面





小程序权限申请

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：**

小程序控制台配置合法域名

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：**

> **说明：**
> 自 Chat v3.4.6起，SDK 支持独立域名，您需要配置独立域名可更好地保障服务稳定性。
> 例如您的 SDKAppID 是 `1400xxxxxx`，则独立域名为：`wss://1400xxxxxxw4c.my-imcloud.com`。

```
wss://${SDKAppID}w4c.my-imcloud.com
wss://wss.im.qcloud.com
wss://wss.tim.qq.com
wss://wssv6.im.qcloud.com
```

配置服务器信息

① 身份认证 —— ② 配置服务器信息

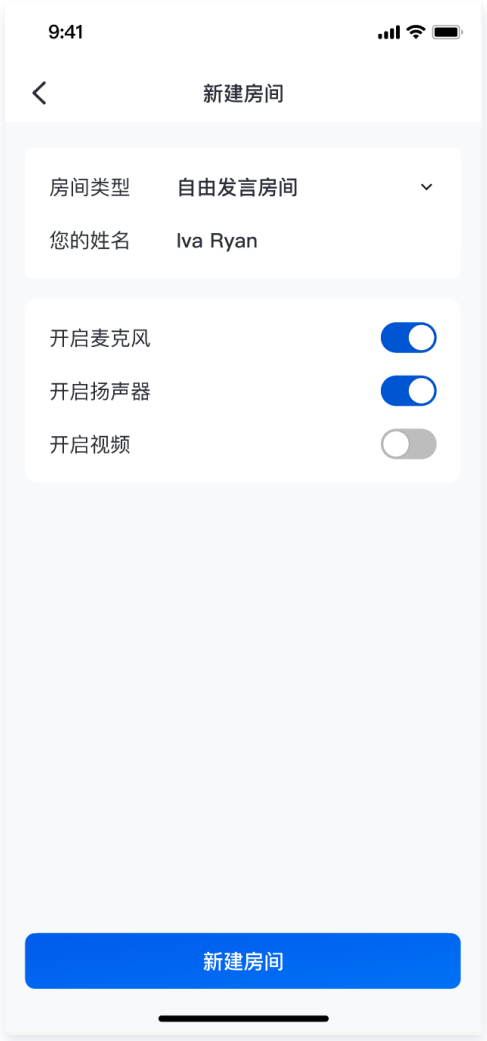
如需购买服务器资源及域名，可 前往腾讯云购买。有可使用官方推出的 微信云开发或微信云托管，无需服务器及域名配置即可上线小程序，立即开通

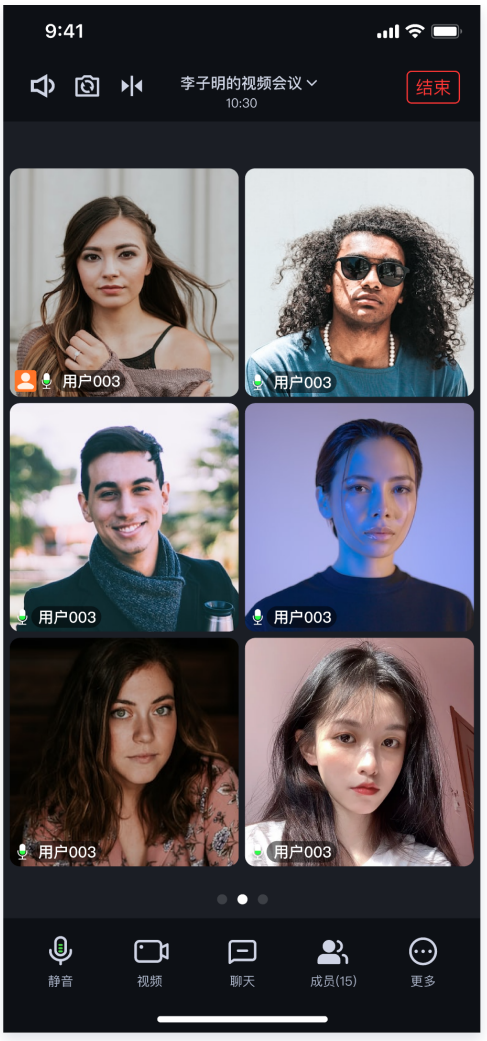
request合法域名	https://official.opensso.tencent-cloud.com;https://yun.tim.qq.com;https://cloud.tencent.com;https://webim.tim.qq.com;https://query.tencent-cloud.com;https://web.sdk.qcloud.com;
socket合法域名	wss://wss.im.qcloud.com;wss://wss.tim.qq.com;
uploadFile合法域名	以 https:// 开头。可填写多个域名，域名间请用 ; 分割
downloadFile合法域名	以 https:// 开头。可填写多个域名，域名间请用 ; 分割
udp合法域名	以 udp:// 开头。可填写多个域名，域名间请用 ; 分割
tcp合法域名	以 tcp:// 开头。可填写多个域名，域名间请用 ; 分割

uni-app (客户端)

最近更新时间：2024-12-13 16:58:32

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在10分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。





环境准备

- 建议使用最新的 HBuilderX 编辑器 。
- iOS 9.0 或以上版本且支持音视频的 iOS 设备，暂不支持模拟器。
- Android 版本不低于 4.1 且支持音视频的 Android 设备，暂不支持模拟器。如果为真机，请开启允许调试选项。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。
- iOS/Android 设备已经连接到 Internet。

快速跑通

1. 下载或者克隆源码并安装依赖。

```
git clone https://github.com/tencentyun/TUIRoomKit.git
```

2. 安装依赖

```
cd ./TUIRoomKit/uni-app
```

```
npm install
```

3. 下载并导入 TUIRoomEngine 插件。
   - 登录 uni 原生插件市场，并访问 TencentCloud-TUIRoomEngine 插件，在插件详情页中购买（TencentCloud-TUIRoomEngine 为免费插件）。购买后才能够云端打包使用插件。**购买插件时请选择正确的 appid，以及绑定正确包名。**

   - 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。

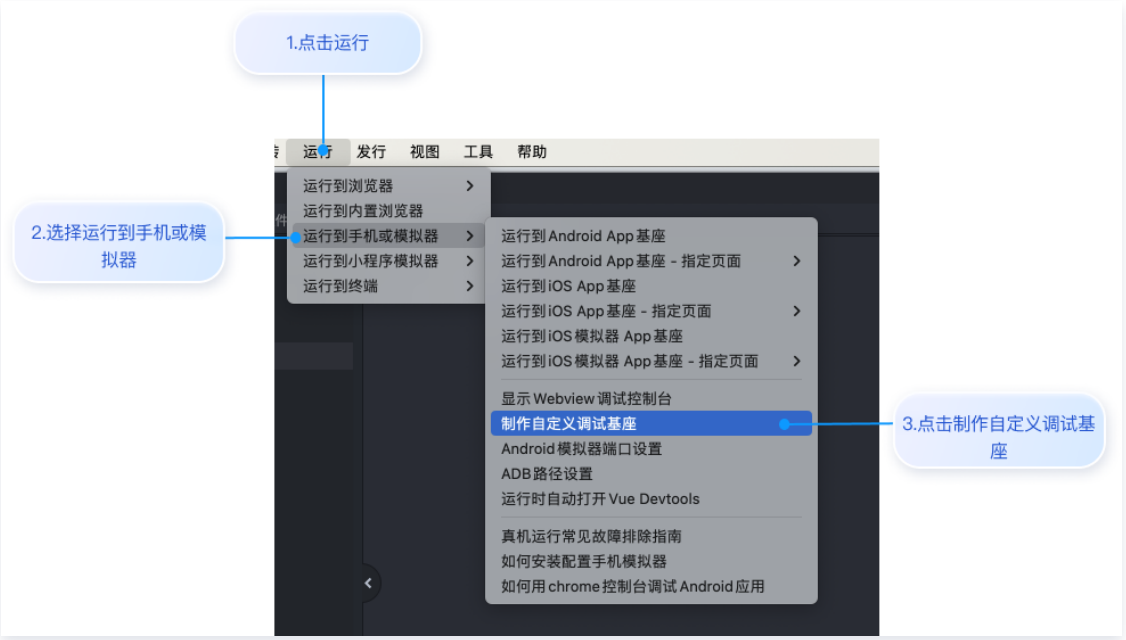
4. 找到并打开 TUIRoomKit/uni-app/src/roomkit/config/basic-info-config.js 文件。（获取 SDKAppID 与 SDKSECRETKEY）：

```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

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

/**
 * 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.
 *
 */

export const SDKAPPID = 0;

/**
 * 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
 *
 */
export const SDKSECRETKEY = '';
```

说明：

激活 TUIRoomKit 服务，获取 **SDKAppID** 和 **SDKSecretKey**。

5. 制作自定义调试基座，请选择传统打包方式进行打包。



6. 自定义调试基座成功后，使用自定义基座运行项目。

创建您的第一个会议

主持人端：

2.选择房间类型
4.点击查看房间信息
15:51
home
myName
腾讯云
多人视频会议
进入房间
新建房间
1.点击新建房间
新建房间
房间类型
自由发言房间
您的名字
user_2
开启麦克风
开启视频
3.点击新建房间
16:04
user_2的快速会议
00:04
麦克风
摄像头
成员 (1)
更多
16:02
user_2的快速会议
00:17
取消
5.记录房间号

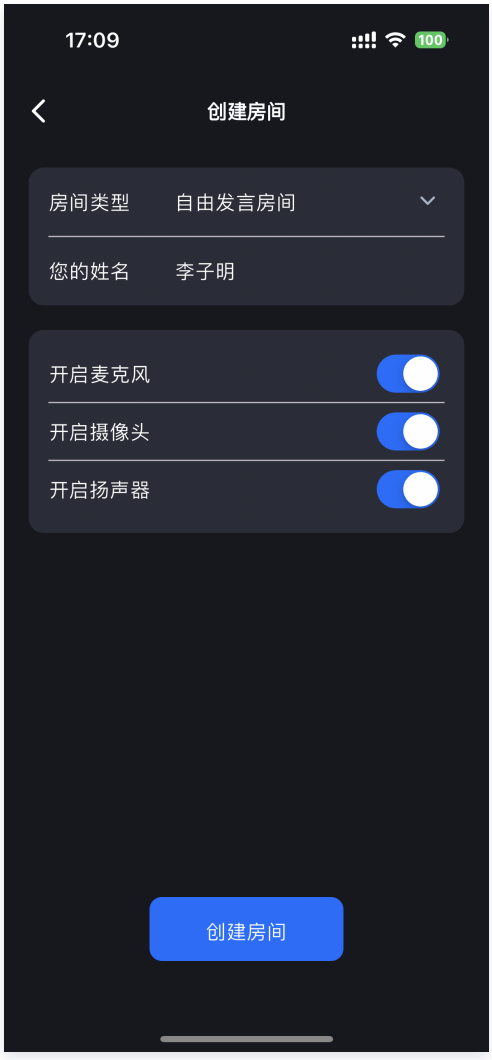
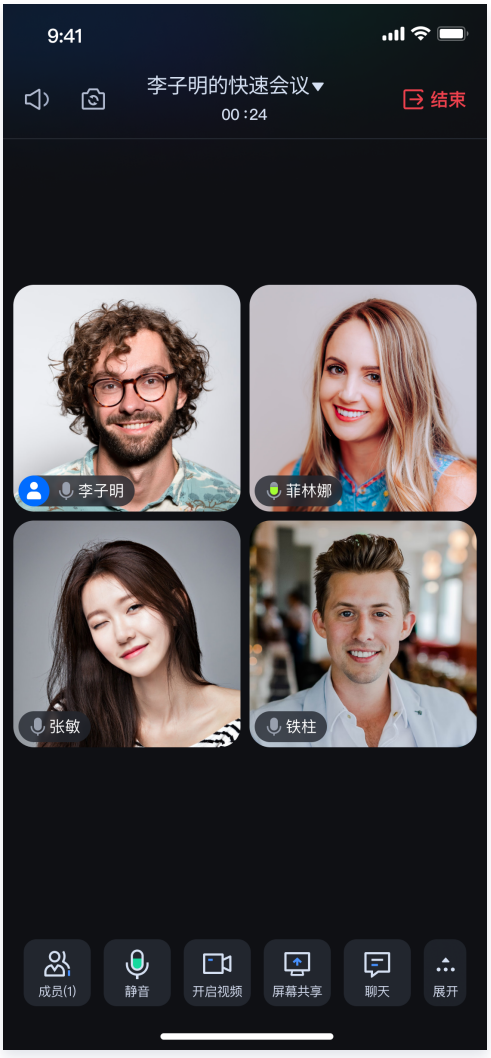
普通成员端：

2.输入记录的房间号
4.成功加入会议
15:51
home
myName
腾讯云
多人视频会议
进入房间
新建房间
1.点击进入房间
16:57
home
进入房间
房间号
输入房间号
您的名字
user_10
开启麦克风
开启视频
进入房间
3.点击进入房间
16:57
user_2的快速会议
00:14
结束
user_10
user_2
麦克风
摄像头
成员 (2)
更多

Flutter

最近更新时间：2024-09-14 10:47:01

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在 10 分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。

创建会议界面	会议主界面
	

环境准备

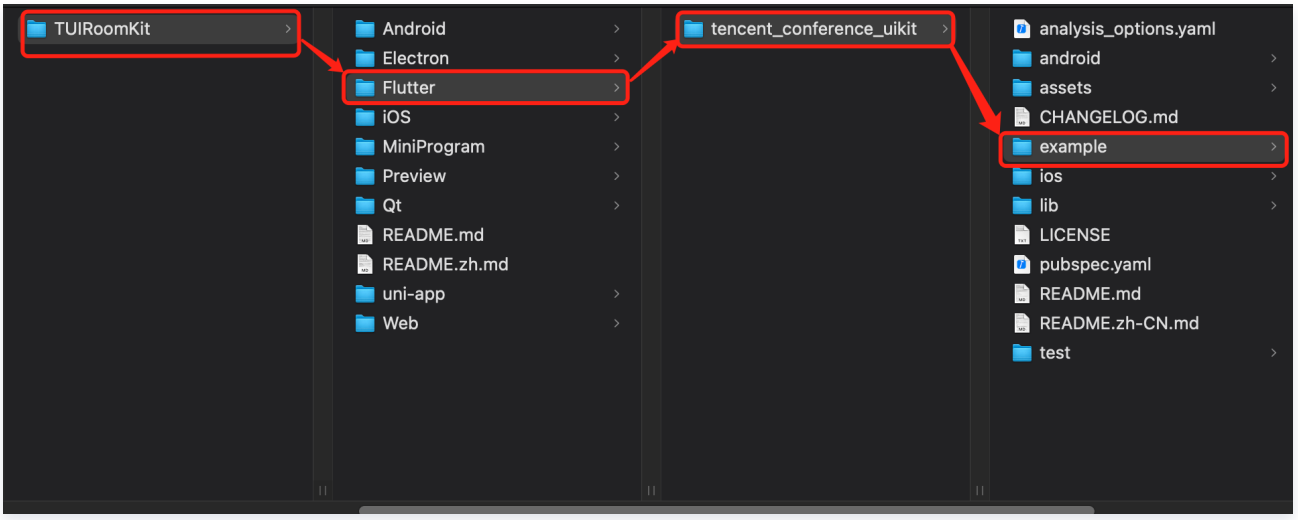
平台	版本
Flutter	3.22.0 及以上版本。
Android	最低兼容 Android 4.1（SDK API Level 16），建议使用 Android 5.0（SDK API Level 21）及以上版本。 Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本）。Android 4.1 及以上的手机设备。
iOS	iOS 12.0 及更高。

下载 Demo

1. 从 github 下载 TUIRoomKit Demo 源码，或者直接在命令行运行以下命令：

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

2. 通过 Android Studio 或 VSCode 打开 TUIRoomKit Flutter 的 example 工程。以下流程将以 VSCode 为例：



配置 Demo

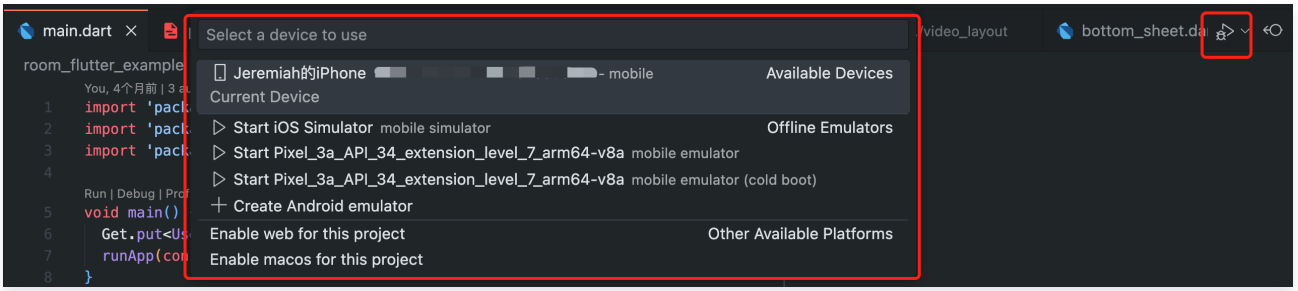
1. 开通 TUIRoomKit 服务，获取 SDKAppID 和 SDKSecretKey。

2. 打开工程，在工程内找到 `example/lib/debug/generate_test_user_sig.dart` 文件。将上一步中获取到的对应的 SDKAppID 和 SDKSecretKey 填入其中：

```
class GenerateTestUserSig {
  /// Tencent Cloud `SDKAppID`. Set it to the `SDKAppID` of your account.
  ///
  /// You can view your `SDKAppID` after creating an application in the [TRTC console](https://console.cloud.tencent.com/trtc).
  /// `SDKAppID` uniquely identifies a Tencent Cloud account.
  static int sdkAppId = 0;

  /// Signature validity period, which should not be set too short
  /// <p>
  /// Unit: second
  /// Default value: 604800 (7 days)
  static int expireTime = 604800;

  /// Follow the steps below to obtain the key required for UserSig calculation.
  ///
  /// Step 1. Log in to the [TRTC console](https://console.cloud.tencent.com/trtc), and create an application if you don't have one.
  /// Step 2. Find your application, click "Application Info", and click the "Quick Start" tab.
  /// Step 3. Copy and paste the key to the code, as shown below.
  ///
  /// Note: this method is for testing only. Before commercial launch, please migrate the UserSig calculation code and key to your backend serve
  /// Reference: https://cloud.tencent.com/document/product/647/17275#Server
  static String secretKey = '';      You, 1秒钟前 • Uncommitted changes
  static genTestSig(String userId) {
```

跑通 Demo

1. 使用 VSCode 打开 `example/lib/main.dart`，点击右下角的设备连接按钮。在上方弹出的弹框中选择需要运行的设备。选择后点击右上角的运行按钮，即可将TUIRoomKit Flutter Demo运行在设备上。



2. 您也可以在 `example` 目录下，执行以下命令以在您的设备上运行。

```
flutter run
```

APP主界面	创建会议界面

创建您的第一个会议

单击新建房间创建您的第一个会议房间，房间类型分别为自由发言房间和举手发言房间。

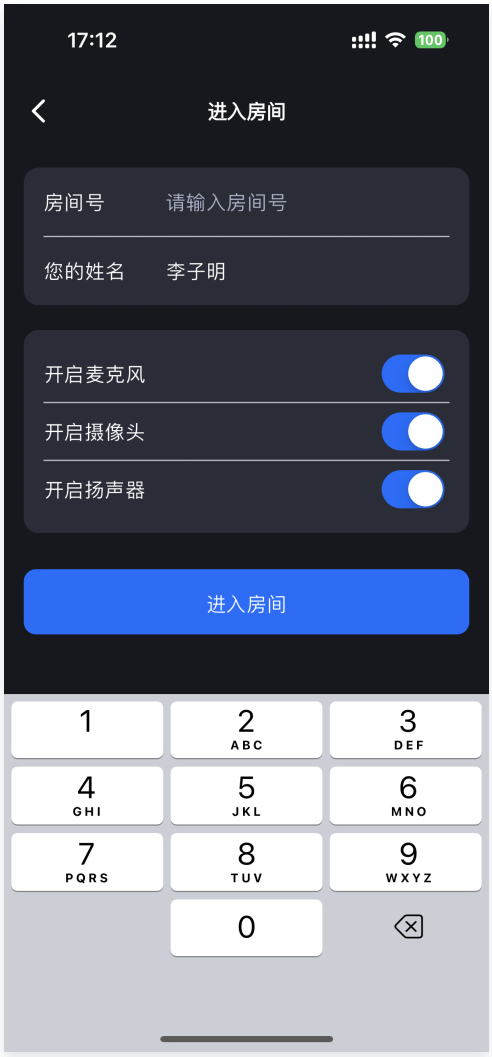
- **自由发言房间：普通用户可以自由进行发言，且自由开关麦克风和摄像头。**
- **上台发言房间：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。**

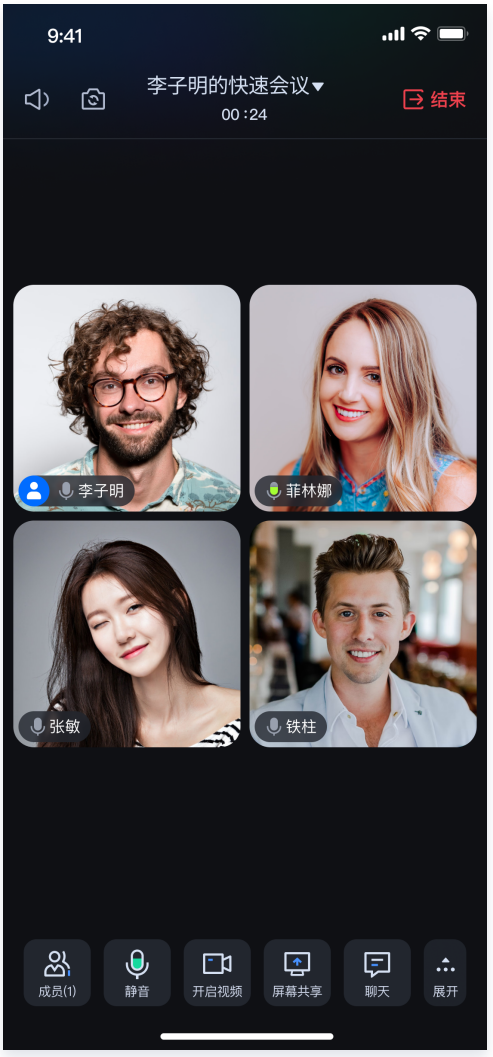
自由发言房间成员管理界面	上台发言房间成员列表

加入会议

单击加入房间后，参与者可以通过填写对应的 `RoomId` ，来加入由会议主持人创建的会议。

加入会议界面	会中界面

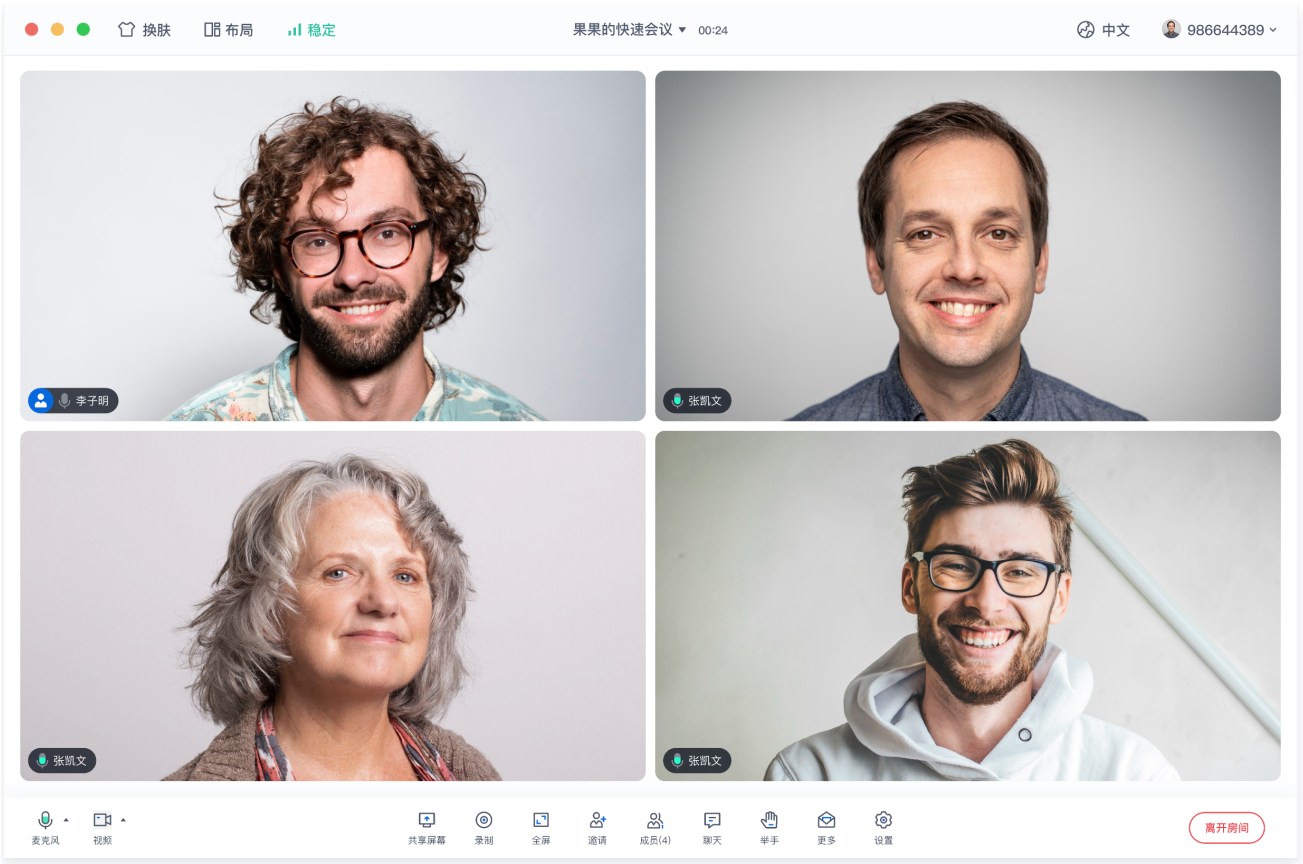

17:12
进入房间
房间号
请输入房间号
您的姓名
李子明
开启麦克风
开启摄像头
开启扬声器
进入房间
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0


9:41
李子明的快速会议
00:24
结束
李子明
菲林娜
张敏
铁柱
成员(1)
静音
开启视频
屏幕共享
聊天
展开

Electron (Windows&Mac)

最近更新时间：2025-05-27 16:03:51

本文档主要介绍如何快速跑通 TUIRoomKit 示例工程，体验高质量多人视频会议。跟随本文档，您可以在10分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的多人视频会议功能。



环境准备

- Node.js 版本：Node.js ⩾ 16.19.1（推荐使用官方 LTS 版本，npm 版本请与 node 版本匹配）。
- 现代浏览器，支持 WebRTC APIs。

下载 Demo

1. 打开终端复制输入示例命令克隆仓库。

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

2. 安装依赖

Vue3

```
cd ./TUIRoomKit/Electron/example/vue3
```

Vue2

```
cd ./TUIRoomKit/Electron/example/vue2
```

```
npm install --legacy-peer-deps
```

配置 Demo

1. 激活 TUIRoomKit 服务，获取 SDKAppID 和 SDKSecretKey。

2. 打开 TUIRoomKit/Electron/example/vue3/packages/renderer/src/config/basic-info-config.js 文件，并输入激活服务时获得的 SDKAppID 和 SDKSecretKey：

```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

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

/**
 * 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.
 *
 */

export const SDKAPPID = 0;

/**
 * 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
 *
 */
export const SDKSECRETKEY = '';
```

> **注意：**
> Vue2 项目请打开 **TUIRoomKit/Electron/example/vue2/src/config/basic-info-config.js** 文件，并输入激活服务时获得的 **SDKAppID** 和 **SDKSecretKey** 即可。

跑通 Demo

在终端中输入命令，运行 Demo。

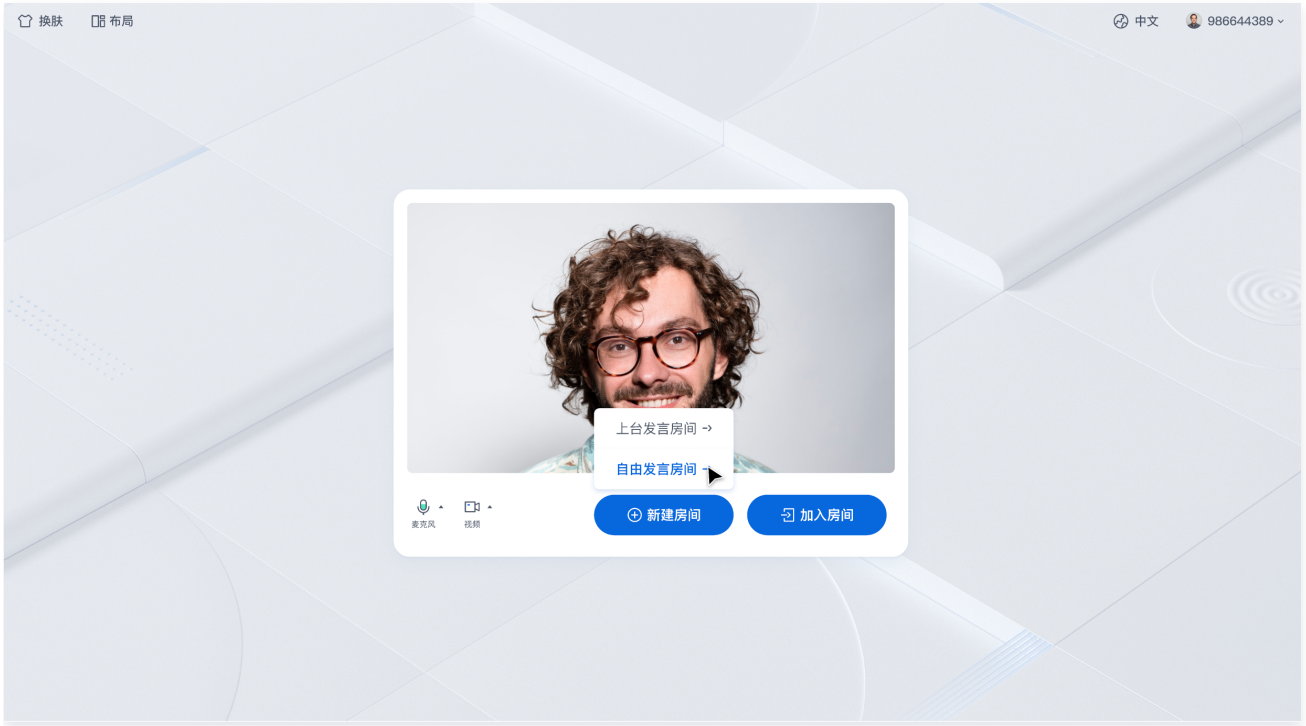
Vue3

```
# cd TUIRoomKit/Electron/example/vue3
npm run dev
```

Vue2

```
# cd TUIRoomKit/Electron/example/vue2
npm run start
```

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

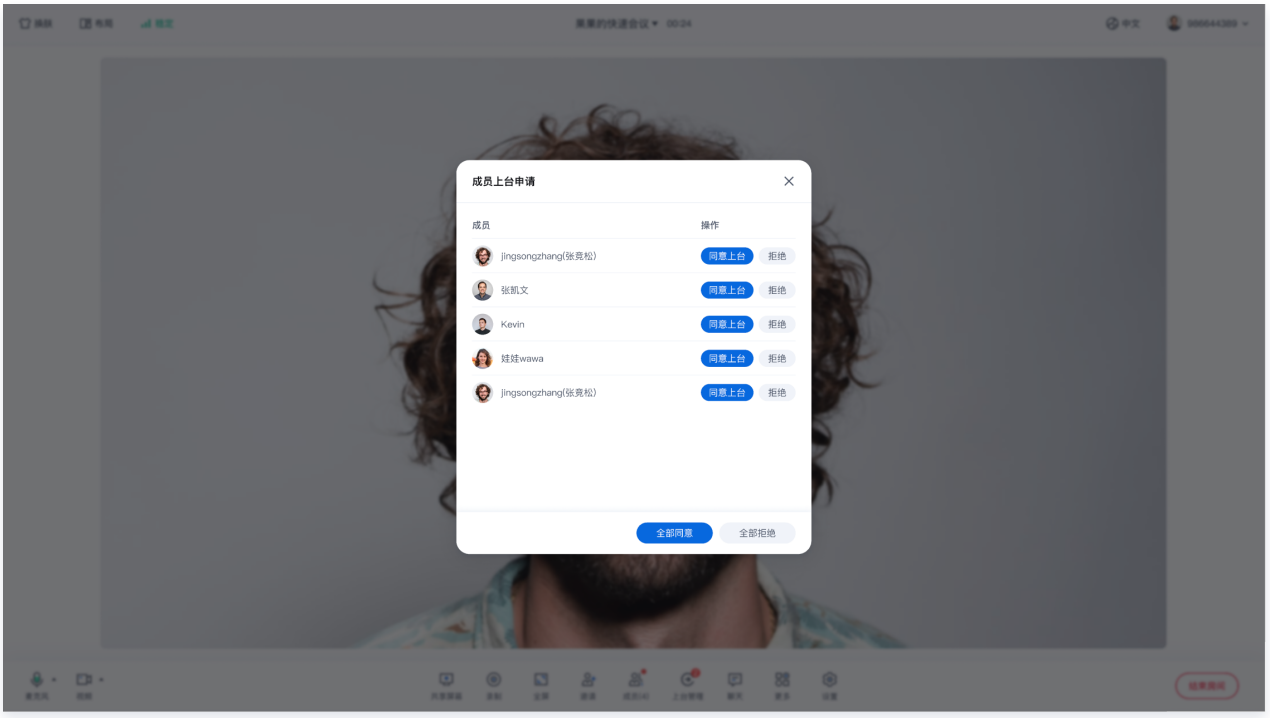


创建您的第一个会议

单击 新建房间 创建您的第一个会议房间，房间类型分别为**举手发言房间**和**自由发言房间**。

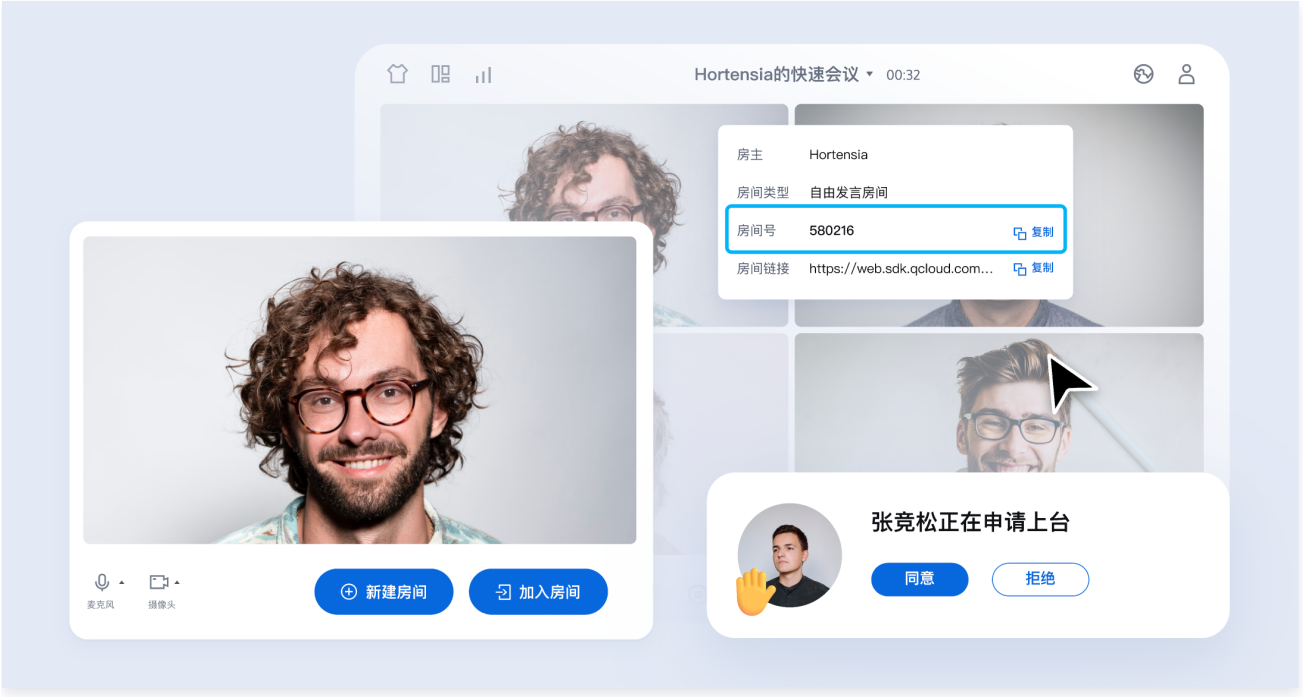
1. 自由发言房间

2. 举手发言房间



加入会议

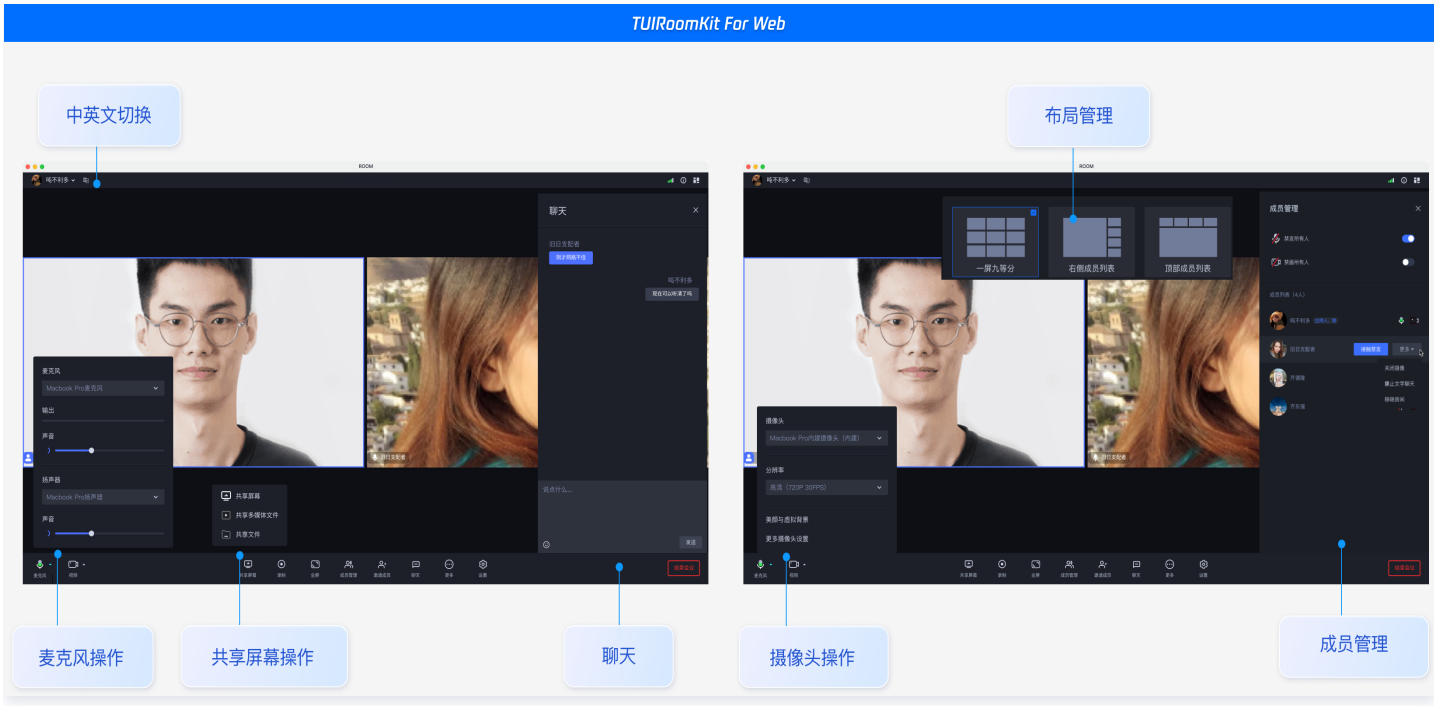
参与者可以通过填写对应的 `RoomId`，来加入由会议主持人创建的会议。



快速接入（TUIRoomKit）Web&H5 (Vue2/Vue3)

最近更新时间：2025-02-11 11:43:02

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。



TUIRoomKit Demo 体验

- 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 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Web 示例工程。

环境准备

- Node.js 版本：Node.js ≥ 18（推荐使用官方 LTS 版本，npm 版本请与 node 版本匹配）。
- Modern browser, supporting WebRTC APIs.

集成 TUIRoomKit 组件

> **说明：**
> 如果您当前尚无 Vue 项目，建议您前往 Github 下载 TUIRoomKit example 项目，参考项目 README.md 文档运行示例工程，并根据自身需求增加业务逻辑。

如果需要在已有项目中集成，请按照以下步骤进行接入。

步骤一：安装依赖

Vue3

```
npm install @tencentcloud/roomkit-web-vue3 pinia --save
```

Vue2

```
# 注意此处要求 vue版本 >= 2.7.16，如安装失败请检查您的 vue 版本是否支持
npm install @tencentcloud/roomkit-web-vue2.7 pinia
```

说明：
TUIRoomKit npm 包提供了会前预览组件、会中组件以及发起会议、加入会议和界面微调的方法。如需了解更多，请参考 RoomKit API 。若这些 API 无法满足您的业务需求，您可以参见 UIKit 源码导出 方案接入 TUIRoomKit 源码。

步骤二：项目工程配置

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。

Vue3

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册 pinia
app.use(createPinia());
app.mount('#app')
```

Vue2

```
// 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');
```

步骤三：引用 TUIRoomKit 组件

说明：
引入 ConferenceMainView 组件，组件默认处于 常驻模式（组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态）。

Vue3

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue3';
```

```
</script>
```

Vue2

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script>
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
};
</script>
```

步骤四：登录 TUIRoomKit 组件

开启会议前需要调用 login 接口进行登录。获取 sdkAppId、userId、userSig 可参见 开通服务 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，https://cloud.tencent.com/document/product/647/104842
    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: '',
});
```

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 开通服务 中的最后一步中获取。
- **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**：通过对在 开通服务 中获取的 `SDKSecretKey` 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：发起新的会议

会议主持人可以通过调用 start 接口来发起一场新的会议，其他参会者可以参见 步骤六 的描述，调用 join 接口加入该会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
const startConference = async () => {
    await conference.login({
        // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
        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: '',
    });
    await conference.start('123456', {
      roomName: 'TestRoom',
      isSeatEnabled: false,
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
}
startConference()
```

步骤六：进入已有会议

参与者可以通过调用 join 接口，填写对应的 roomId 参数，来加入由会议主持人在 步骤五 中发起的会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
const joinConference = async () => {
    await conference.login({
        // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
        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: '',
    });
    await conference.join('123456', {
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
}
```

```
joinConference()
```

开发环境运行

1. 执行开发环境命令。（此处以 vue3 + vite 默认项目为例，不同项目 dev 指令可能不同，请根据您自己的项目进行调整）

```
npm run dev
```

2. 根据控制台提示，在浏览器中打开页面，如：`http://localhost:3000/`。
3. 体验 TUIRoomKit 组件功能。

生产环境部署

1. 打包 dist 文件。

```
npm run build
```

2. 部署 dist 文件到服务器上。

> **注意：**
> 生产环境要求使用 HTTPS 域名，参见下图：

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	

其他文档

- TUIRoomKit
- TUIRoom Demo 快速跑通
- 设置昵称、头像
- Web&Electron
- 界面定制 (TUIRoomKit)
- 常见问题
- 教育场景解决方案

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。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2024-11-12 16:06:52

本文将指导您如何在较短时间内接入 `TUIRoomKit` 组件。遵循本指南，您将在10分钟内完成以下关键步骤，并最终实现具备完整 UI 界面的音视频会议功能。

环境准备

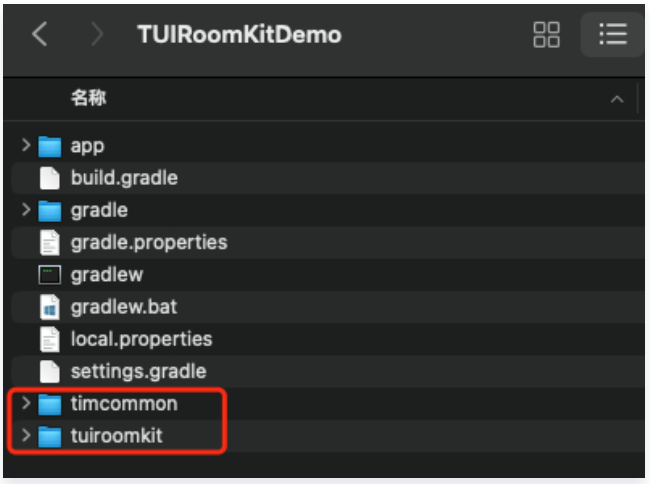
- 最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本。
- Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本）。
- Android 4.4 及以上的手机设备。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 开通服务 。

步骤二：下载 TUIRoomKit 组件

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



步骤三：工程配置

1. 工程根目录下找到 `setting.gradle`（或`settings.gradle.kts`）文件，并在其中增加如下代码，它的作用是将 `tuiroomkit` 组件导入到您当前的项目中。

settings.gradle

```
include ':timcommon'
include ':tuiroomkit'
```

setting.gradle.kts

```
include (":timcommon")
include (":tuiroomkit")
```

2. 在 app 目录下找到 `build.gradle`（或`build.gradle.kts`）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。

build.gradle

```
api project(':tuiroomkit')
```

build.gradle.kts

```
api(project(":tuiroomkit"))
```

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.** { *; }
```

4. 在 app目录下找到 AndroidManifest.xml 文件，在 `application` 节点中添加 `tools:replace="android:allowBackup"`，覆盖组件内的设置，使用自己的设置。

```
// app/src/main/AndroidManifest.xml
<application
  android:name=".DemoApplication"
  android:allowBackup="false"
  android:icon="@drawable/app_ic_launcher"
  android:label="@string/app_name"
  android:largeHeap="true"
  android:theme="@style/AppTheme"
  tools:replace="android:allowBackup">
```

步骤四：登录

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。

Java

```
import com.tencent.qcloud.tuicore.TUILogin;
import com.tencent.qcloud.tuicore.interfaces.TUICallback;
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig;

String userId = "denny"; // 请替换为您的 UserID
int sdkAppId = 1400000001; // 请替换为步骤一取到的 sdkAppId
String sdkSecretKey = "xxxx"; // 请替换为您的 sdkSecretKey
String userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId, sdkSecretKey);

TUILogin.login(context,
    sdkAppId,
    userId,
    userSig,
    new TUICallback() {
        @Override
        public void onSuccess() {
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
        }
});
```

Kotlin

```
import com.tencent.qcloud.tuicore.TUILogin
import com.tencent.qcloud.tuicore.interfaces.TUICallback
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig

val userId = "denny" // 请替换为您的 UserID
val sdkAppId = 1400000001 // 请替换为步骤一取到的 sdkAppId
val sdkSecretKey = "xxxx" // 请替换为您的 sdkSecretKey
val userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId, sdkSecretKey)
```

```
TUILogin.login(this,
    sdkAppId,
    userId,
    userSig,
    object : TUICallback() {
      override fun onSuccess() {
      }

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

TUILogin.login 函数参数说明：

- **SDKAppID**：在 开通服务 中的最后一步中获取。
- **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 开通服务 的第3步中获取的 `SDKSecretKey` 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 。

> **说明：**
> - **开发环境**：如果您正在本地开发调试阶段，可以采用本地 GenerateTestUserSig.genTestUserSig() 函数生成 userSig。该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。
> - **生产环境**：如果您的项目要发布上线，请采用 服务端生成 UserSig 的方式。

步骤五：发起您的第一次会议

在 TUILogin.login 成功后，参考如下代码，即可发起会议。

Java

```
// 请替换 "123456" 为您自定义的房间号
ConferenceDefine.StartConferenceParams params = new
ConferenceDefine.StartConferenceParams("123456");
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent);
```

Kotlin

```
// 请替换 "123456" 为您自定义的房间号
val params = ConferenceDefine.StartConferenceParams("123456")
val intent = Intent(this, ConferenceMainActivity::class.java)
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent)
```

步骤六：加入会议

在 TUILogin.login 成功后，参考如下代码，即可加入会议。

Java

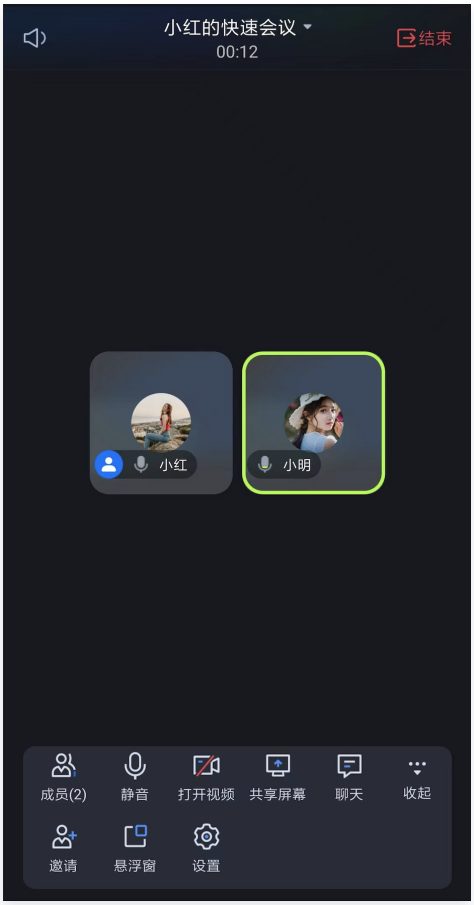
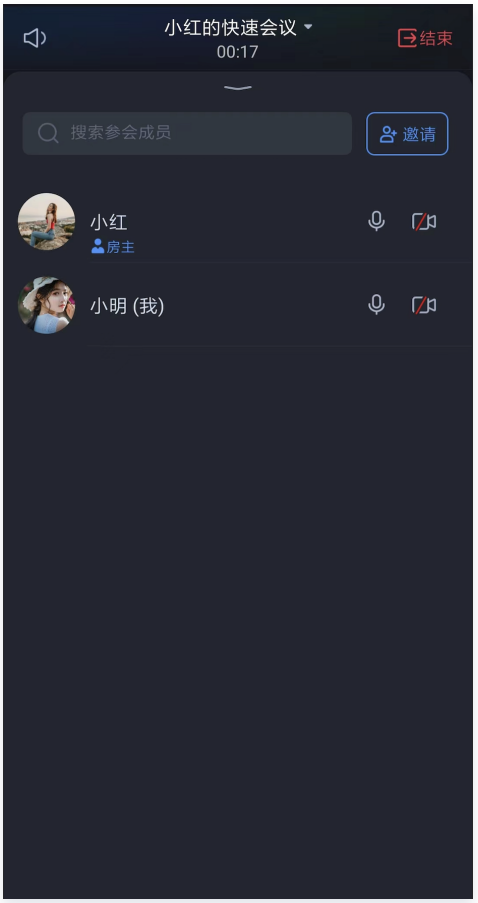
```
// 请替换 "123456" 为您自定义的房间号
ConferenceDefine.JoinConferenceParams params = new
ConferenceDefine.JoinConferenceParams("123456");
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent);
```

Kotlin

```
// 请替换 "123456" 为您自定义的房间号
val params = ConferenceDefine.JoinConferenceParams("123456")
val intent = Intent(this, ConferenceMainActivity::class.java)
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent)
```

界面展示

当您成功完成步骤1 – 步骤6，UI 界面效果如下：

会议主界面	用户列表
	

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题 。

更多特性

- 预定会议
- 会中呼叫
- 会中聊天
- 弹幕聊天
- 房间密码
- 设置昵称、头像
- 会前提醒
- 悬浮窗
- 文字水印

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao ，进行技术交流和问题反馈。

iOS

最近更新时间：2024-09-26 11:02:51

本文将介绍如何快速完成 `TUIRoomKit` 组件的接入，跟随本文档，您将在一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。
- Xcode 12.0 及更高。
- Swift 4.2 及更高。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 开通服务 。

步骤二：导入TUIRoomKit组件

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。

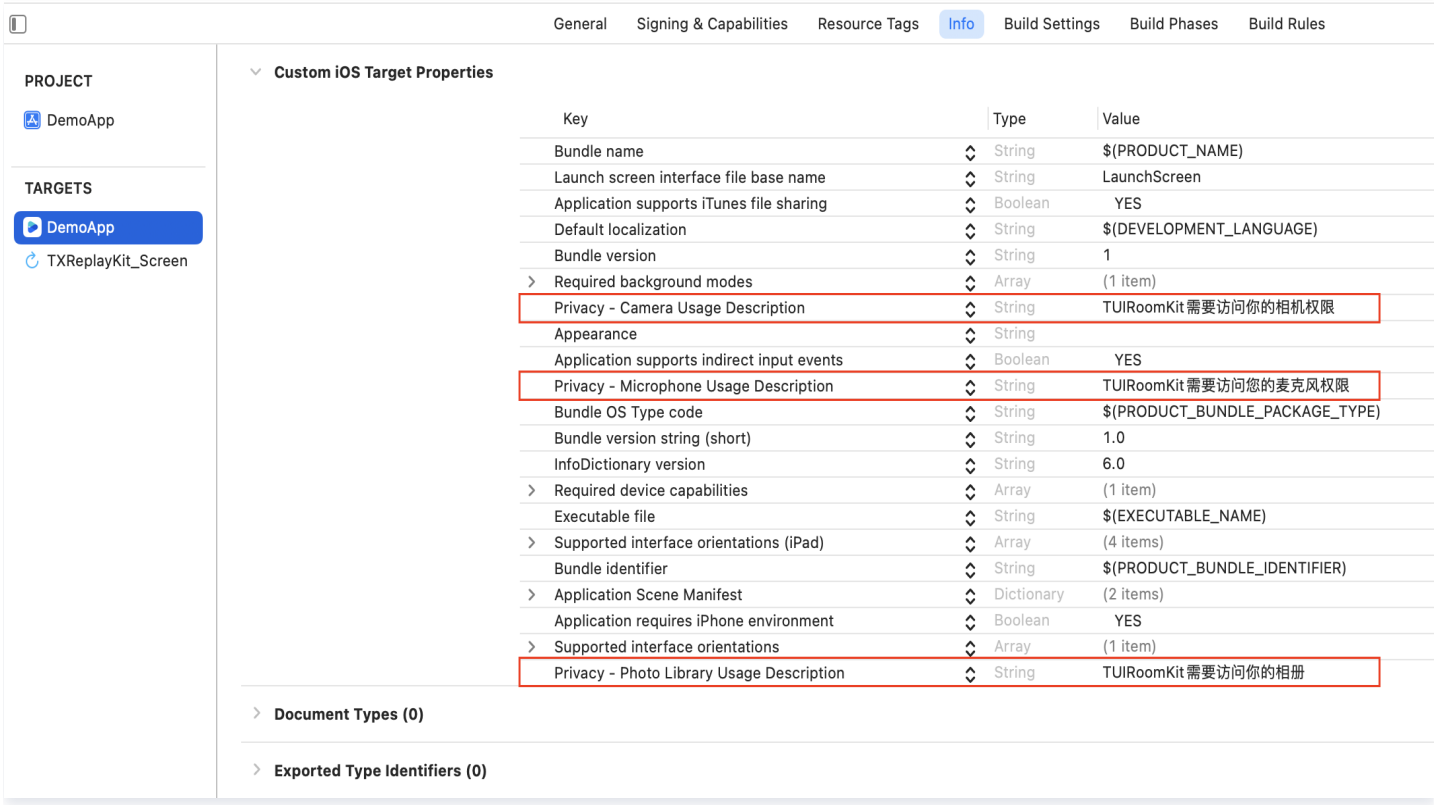
```
pod install
```

> **说明：**
> 如果无法安装 TUIRoomKit 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：工程配置

使用音视频功能，需要授权麦克风、摄像头和相册的使用权限。在 App 的 Info.plist 中添加以下几项，分别对应麦克风、摄像头和相册在系统弹出授权对话框时的提示信息。

```
<key>NSCameraUsageDescription</key>
<string>TUIRoomKit需要访问您的相机权限</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUIRoomKit需要访问您的麦克风权限</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>TUIRoomKit需要访问您的相册</string>
```



步骤四：登录

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：

Swift

```
import TUICore

TUILogin.login(1400000001,                          // 请替换为步骤一取到的 SDKAppID
            userID: "998",                          // 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")")
}
```

OC

```
#import "TUICore/TUILogin.h"

[TUILogin login:1400000001                          // 请替换为步骤一取到的 SDKAppID
         userID:@"998"                              // 请替换为您的 UserID
        userSig:@"xxxxxxxxxx" succ:^{               // 您可以在控制台中计算一个 UserSig 并填在这个位置
    NSLog(@"login,success");
```

```
} fail:^(int code, NSString * _Nullable msg) {
    NSLog(@"login,failed,code:%d,msg:%@",code,msg);
}];
```

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 开通服务 中的第3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 开通服务 的第3步中获取的 SDKSecretKey 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 。

步骤五：主持人发起快速会议

会议主界面为 ConferenceMainViewController，您只需要在登录完成后，按照如下示例创建并跳转 ConferenceMainViewController 即可发起快速会议。

Swift

```
import TUIRoomKit

// CreateConferenceViewController 为您自己的 ViewController
class CreateConferenceViewController: UIViewController {
    func quickStartConferenceAction() {
        let params = StartConferenceParams(roomId: "123456") // 请替换 "123456" 为您自定义的会议号
        let conferenceViewController = ConferenceMainViewController()
        conferenceViewController.setStartConferenceParams(params: params)

        navigationController?.pushViewController(conferenceViewController, animated: true)
    }
}
```

OC

```
#import <TUIRoomKit/TUIRoomKit-Swift.h>

// CreateConferenceViewController 为您自己的 ViewController
@interface CreateConferenceViewController ()
@end

@implementation CreateConferenceViewController

- (void)quickStartConferenceAction {
    // 请替换 "123456" 为您自定义的房间号
    StartConferenceParams *params = [[StartConferenceParams alloc] initWithRoomId:@"12345"
                                                                  isOpenMicrophone:YES
                                                                      isOpenCamera:NO
                                                                     isOpenSpeaker:YES
                                                          isMicrophoneDisableForAllUser:NO
```

```
                                                    isCameraDisableForAllUser:NO
                                                                isSeatEnabled:NO
name:@"YourConferenceName"];
    ConferenceMainViewController *conferenceViewController = [[ConferenceMainViewController
alloc] init];
    [conferenceViewController setStartConferenceParamsWithParams:params];

    [self.navigationController pushViewController:conferenceViewController animated:YES];
}
@end
```

步骤六：普通成员加入会议

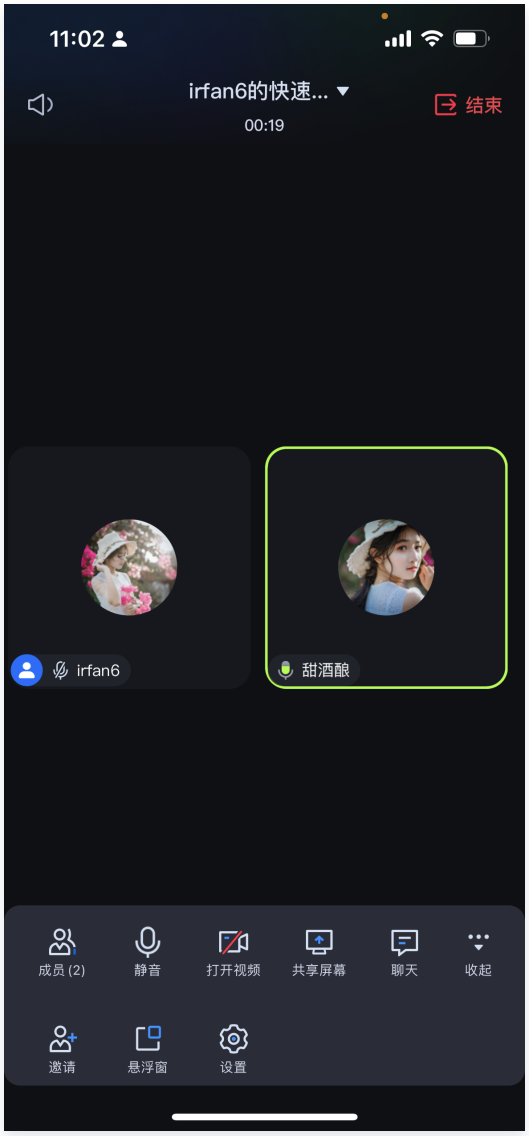
按照如下示例创建并跳转 ConferenceMainViewController，即可加入一场会议。

Swift

```
import TUIRoomKit

// EnterConferenceViewController 为您自己的 ViewController
class EnterConferenceViewController: UIViewController {
    private func joinConferenceAction() {
        let params = JoinConferenceParams(roomId: "123456") // 请替换 "123456" 为您要加入的会议号
        let conferenceViewController = ConferenceMainViewController()
        conferenceViewController.setJoinConferenceParams(params: params)

        navigationController?.pushViewController(conferenceViewController, animated: true)
    }
 }
```

OC

```
#import <TUIRoomKit/TUIRoomKit-Swift.h>

// EnterConferenceViewController 为您自己的 ViewController
@interface EnterConferenceViewController ()
@end

@implementation EnterConferenceViewController

- (void)joinConferenceAction {
    // 请替换 "123456" 为您自定义的房间号
    JoinConferenceParams *params = [[JoinConferenceParams alloc]initWithRoomId:@"12345"
isOpenMicrophone:YES isOpenCamera:NO isOpenSpeaker:YES];
    ConferenceMainViewController *conferenceViewController = [[ConferenceMainViewController
alloc] init];
    [conferenceViewController setJoinConferenceParamsWithParams:params];

    [self.navigationController pushViewController:conferenceViewController animated:YES];
}
@end
```



成员会议主界面

成员用户列表

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题 。

更多特性

- 设置昵称、头像
- 预定会议
- 会中聊天

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

uni-app (小程序)

最近更新时间：2024-11-26 15:14:12

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：**

> **说明：**
> 自 Chat v3.4.6起，SDK 支持独立域名，您需要配置独立域名可更好地保障服务稳定性。
> 例如您的 SDKAppID 是 `1400xxxxxx`，则独立域名为：`wss://1400xxxxxxw4c.my-imcloud.com`。

```
wss://${SDKAppID}w4c.my-imcloud.com
wss://wss.im.qcloud.com
wss://wss.tim.qq.com
wss://wssv6.im.qcloud.com
```

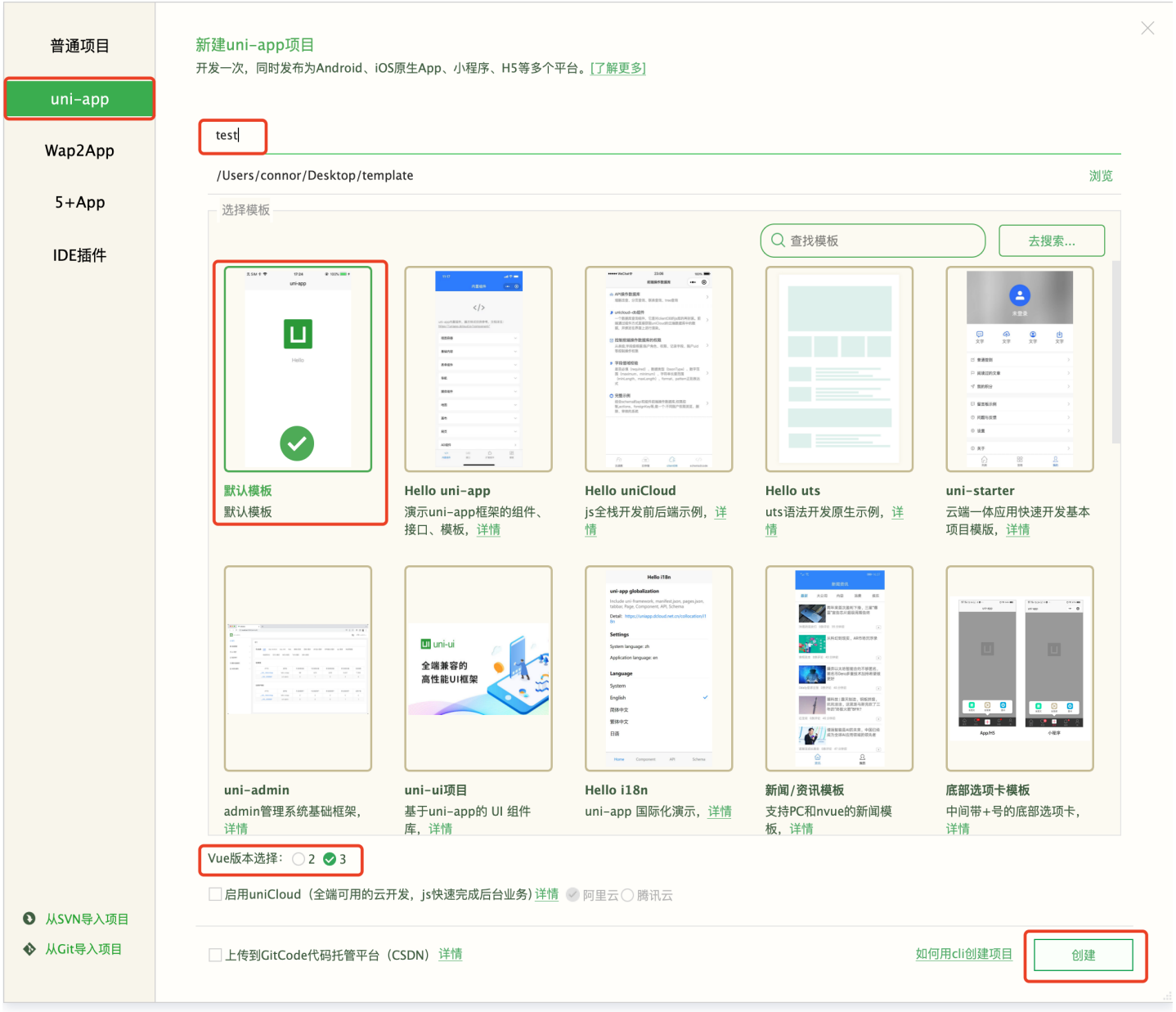
3. 开通服务

在使用 TUIRoomKit 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 开通服务 。

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 pinia
```

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 `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";

// #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());
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)
```

```
  app.use(createPinia());
  return {
    app
  }
}
// #endif
```

CLI

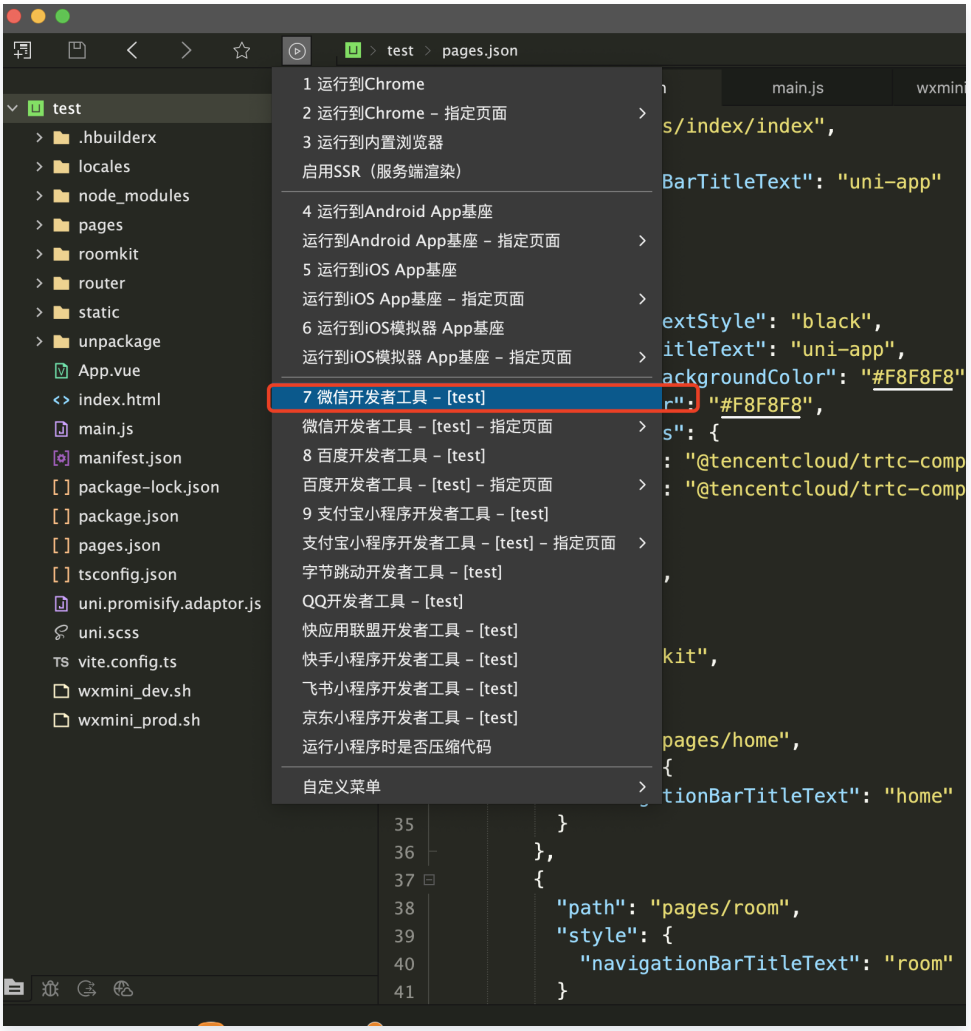
```
import { createPinia } from 'pinia';
// locales 为 TUIRoom/locales，因为主包无法使用分包资源，此处拷贝了一份 locales 在 src 路径下

app.use(createPinia());
```

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;

/**
 * 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 SDKSECRETKEY = '';
```

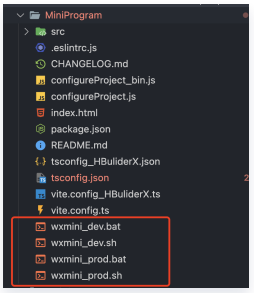
3.4 运行到微信开发者工具

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。



2. 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
```

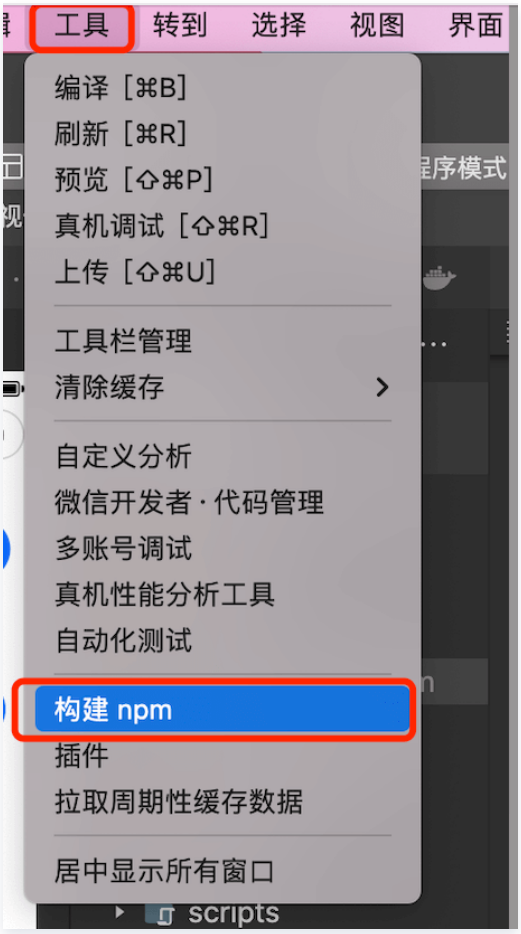
Mac

```
# 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
```

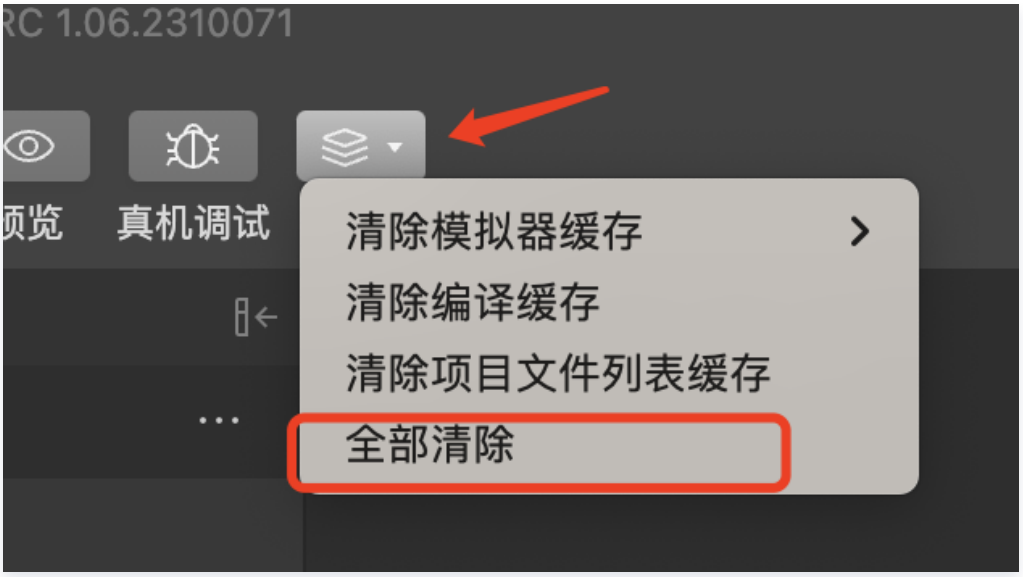
```
bash wxmini_dev.sh
```

> **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。

3. 构建 npm，微信开发者工具单击工具 > 构建 npm。具体如下图：



4. 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。

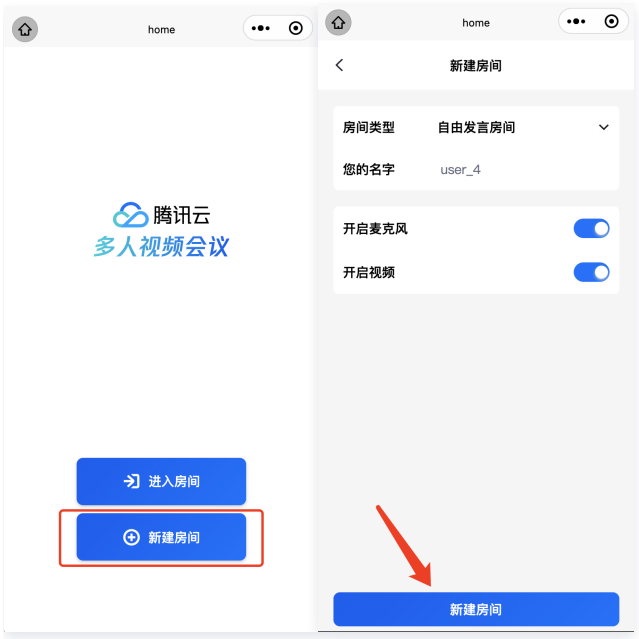


5. 编译小程序。

6. 点击预览，扫描二维码，体验程序。

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"]
}
```

常见问题

常见问题

其他文档

- TUIRoomKit
- TUIRoom demo 快速跑通
- 设置昵称、头像

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 (客户端)

最近更新时间：2025-04-22 10:07:02

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 (客户端) 示例工程。

TUIRoomKit Demo 体验

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

- HbuilderX 版本要求：HbuilderX 版本 ≥ 3.94。
- 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：开通服务

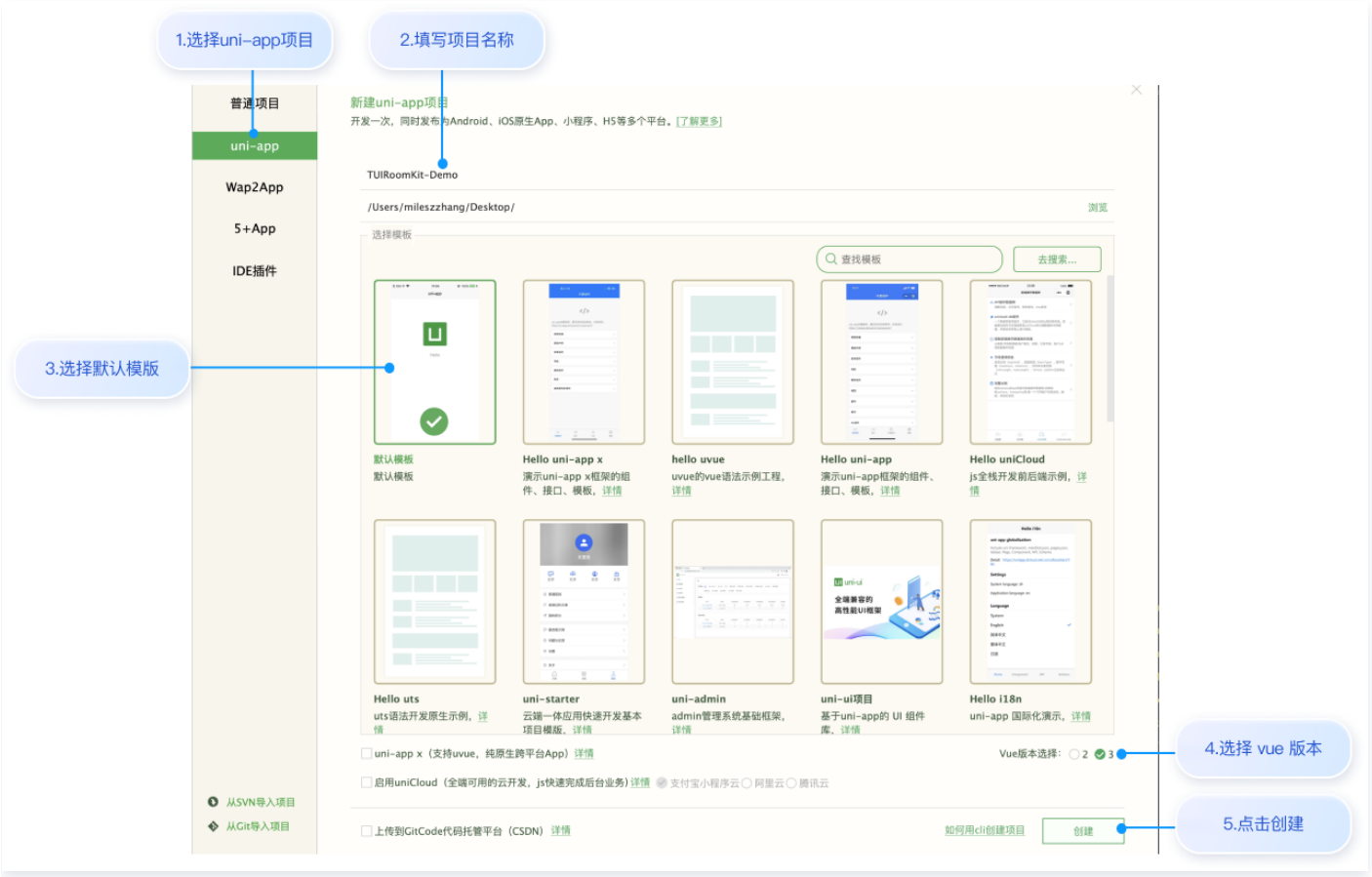
在使用腾讯云提供的音视频服务前，您需要前往控制台，为应用开通音视频服务。具体步骤请参见 开通服务 。

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 (TUIRoomKit-Demo)。

> ⚠ 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。



步骤三：下载并导入 TUIRoomEngine 插件

1. 访问 TencentCloud-TUIRoomEngine 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 `TUIRoomKit-Demo` 项目中导入插件。

步骤四：下载并导入 TUIRoomKit 组件

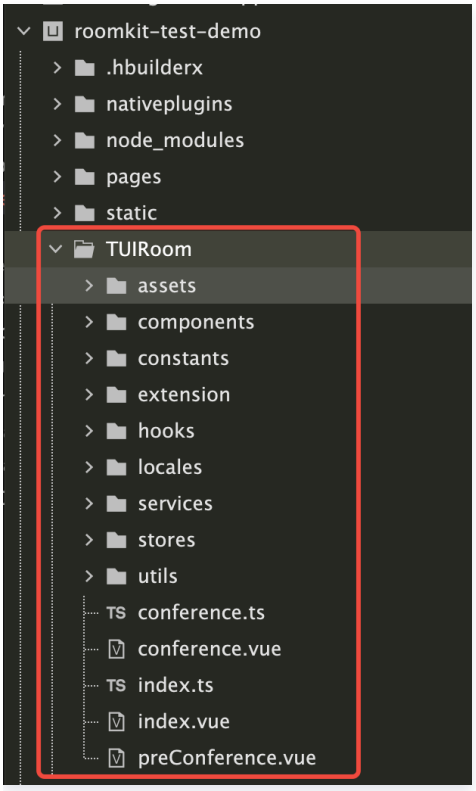
1. **下载 github 源码，建议下载到和步骤一创建的项目同级目录**

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

2. **手动拷贝 TUIRoom 组件至自己的项目中**

   拷贝 `TUIRoomKit/uni-app/src/roomkit/` **下的** `TUIRoom` **文件夹至创建的项目中**

   拷贝完目录结构如下：



3. **安装依赖**

   HBuilderX 创建的工程不会默认创建 package.json 文件，因此您需要先创建 package.json 文件。新建终端，如下：

```
# 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/tuiroom-engine-uniapp-app @tencentcloud/chat @tencentcloud/universal-
api mitt pinia --save
```

4. **项目工程配置**

   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 `TUIRoomKit-Demo/main.js` **文件。**

```
// #ifndef VUE3
import Vue from 'vue'
import './uni.promisify.adaptor'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
app.$mount()
// #endif
```

```
// #ifdef VUE3
import { createSSRApp } from 'vue';
import App from '@/App.vue';
import * as Pinia from 'pinia';
export function createApp() {
  const app = createSSRApp(App);
  app.use(Pinia.createPinia());
  return {
    app,
    Pinia,
  };
}
// #endif
```

配置 pages.json： 为了更好的使用 TUIRoom 组件的能力，建议您在 `TUIRoomKit-Demo/pages.json` 中新增如下配置。

```
"navigationStyle": "custom",
"app-plus": {
  "titleNView": false
}
```

```
{
  "pages": [ //pages数组中第一项表示应用启动页，参考：https://uniapp.dcloud.io/collocation/pages
    {
      "path": "pages/index/index",
      "style": {
        "navigationBarTitleText": "uni-app",
        "navigationStyle": "custom",
        "app-plus": {
          "titleView": false
        }
      }
    }
  ],
  "globalStyle": {
    "navigationBarTextStyle": "black",
    "navigationBarTitleText": "uni-app",
    "navigationBarBackgroundColor": "#F8F8F8",
    "backgroundColor": "#F8F8F8"
  },
  "uniIdRouter": {}
}
```

5. **模块配置**

   安卓需要使用到麦克风权限，所以需要在 App 模块配置中添加录音模块。

6. **引用 TUIRoom 组件，填写 sdkAppId 等信息。**

> ⚠ **注意：**
> 由于原生插件只可以在 nvue 页面中使用，所以您需要将引入 TUIRoom 组件的页面更改为 nvue 页面，详情见 nvue 使用场景 。

在 `pages/index/index.nvue` 页面中引入 `TUIRoom` 组件

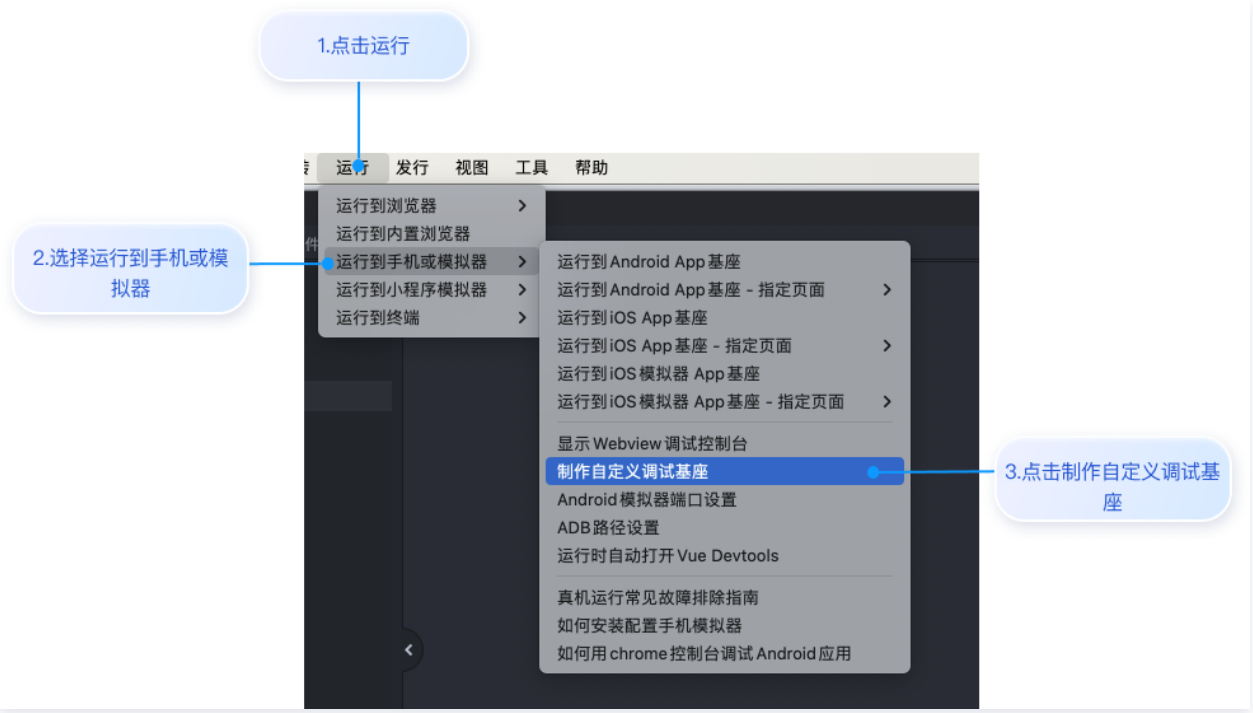
> ⚠ **注意：**
> 请务必将 `pages/index/index.vue` 修改为 `pages/index/index.nvue` 。

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>


<script setup lang="ts">
import ConferenceMainView from '../TUIRoom/conference.vue';
import { conference } from '../TUIRoom/index.ts';

onMounted(async () => {
    await conference.login({
        // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/139678220708134912#d9d2b8d1-f19f-441b-a436-457748c94997
        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: '',
    });
```

```
    await conference.start('123456', {
      roomName: 'TestRoom',
      isSeatEnabled: false,
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
     await conference.join('123456', {
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
}
</script>
```

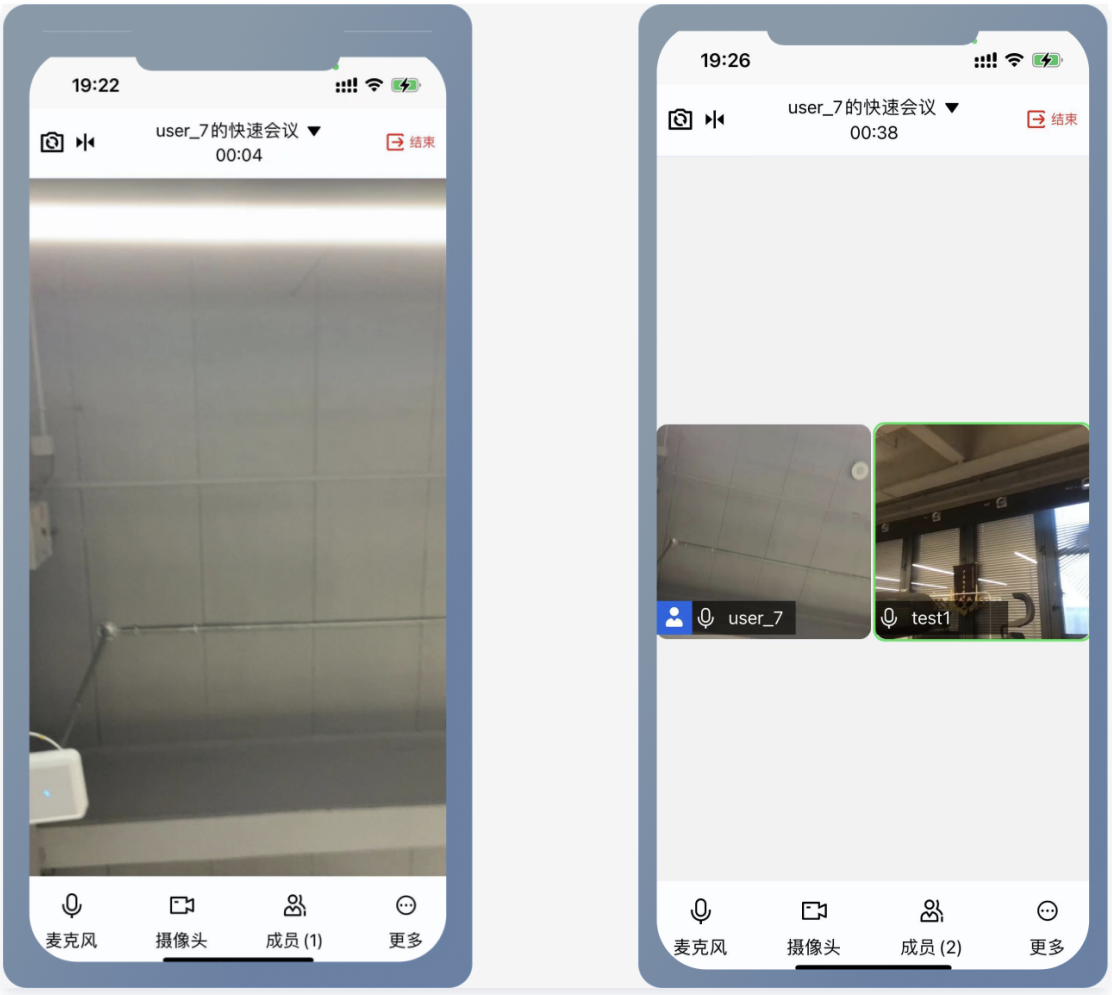
步骤五：开启您的第一场会议

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。

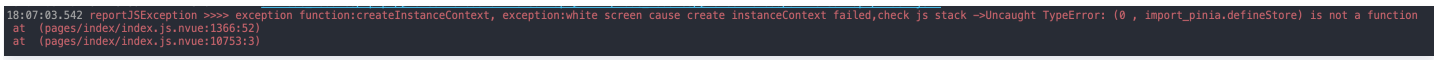
3. 主持人端创建会议以及普通成员加入会议界面。



常见问题

1. 若安卓平台，跑通出现 “ reportJSException >>>> exception function:createInstanceContext, exception:white screen cause create instanceContext

failed,check js stack ->Uncaught TypeError: (0，import pinia.defineStore) is not a function ” 错误，如何解决?



可能是由于 Pinia 版本与 HBuilder 版本不兼容导致的，详情请参见 状态管理 Pinia 。

2. 若 iOS 平台打包自定义基座时，出现体积超过40M的提示。如何解决?

这是和其他原生插件一块使用时造成的体积超出，详情见： 本地 uni-app 原生插件提交云端打包 。

更多特性

- 设置昵称、头像

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。

WebView

最近更新时间：2025-02-08 17:48:33

本文将介绍如何通过 WebView 的方式，在 uni-app、Android、iOS 应用中接入 TUIRoomKit。

uni-app 使用 WebView

1. 在 vue 页面中引入 webview 组件。

```
<template>
    <div>
        <web-view src="https://www.example.com"></web-view>
    </div>
</template>
```

2. 新增如下请求权限的代码：

> **说明：**
> 在使用 uni-app 打包 App 时，打包安卓应用，需要增加请求权限的代码；打包成 iOS 应用，无需进行额外配置。

```
import { onMounted, onBeforeMount } from "vue"
const isIos = plus.os.name === 'iOS';
onMounted(() => {
  if (isIos) return;
  watchPermission();
  requestPermisson();
});

const watchPermission = () => {
  const requestPermission = uni.createRequestPermissionListener();
  // 可以根据弹出的弹框操作，来提示对应的语句
  requestPermission.onConfirm((e) => {
    console.log('the permission request is confirm ', e);
  });
};
const requestPermisson = () => {
  // 请求对应的权限 允许程序修改全局音频设置、允许程序录制音频、允许访问摄像头进行拍照
  plus.android.requestPermissions(['android.permission.MODIFY_AUDIO_SETTINGS',
'android.permission.RECORD_AUDIO', 'android.permission.CAMERA']);
};
```

Android 原生 App 使用 WebView

以 java 语言为例

1. 在 `res/layout/activity_main.xml` 文件中，添加一个 WebView 控件。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/web_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. 在 `AndroidManifest.xml` 文件中添加媒体权限。

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

3. 在 `MainActivity.java` 文件中，新增获取权限方法。

```
package com.example.webview_test; // 替换为自己项目
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.annotation.TargetApi;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.webkit.PermissionRequest;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {
    private static final int MY_PERMISSIONS_RECORD_AUDIO = 1;
    private static final int MY_PERMISSIONS_CAMERA = 1;
    private static final int MY_PERMISSIONS_MODIFY_AUDIO_SETTINGS = 1;
    private WebView webView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        webView=(WebView) findViewById(R.id.web_view);
        webView.setWebViewClient(new WebViewClient());
        webView.loadUrl("https://www.example.com");
        WebSettings webSettings=webView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        webSettings.setDomStorageEnabled(true);
```

```
        webSettings.setMediaPlaybackRequiresUserGesture(false);
        webView.setWebChromeClient(new WebChromeClient(){
            @TargetApi(Build.VERSION_CODES.LOLLIPOP)
            @Override
            public void onPermissionRequest(final PermissionRequest request) {
                request.grant(request.getResources());
            }
        });

        if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.RECORD_AUDIO}, MY_PERMISSIONS_RECORD_AUDIO);
        }

        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.CAMERA}, MY_PERMISSIONS_CAMERA);
        }

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.MODIFY_AUDIO_SETTINGS) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.MODIFY_AUDIO_SETTINGS}, MY_PERMISSIONS_MODIFY_AUDIO_SETTINGS);
        }
    }
}
```

鸿蒙原生 App 使用 WebView

1. 在 `ability_main.xml` ， 新增 WebView 控件。

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:orientation="vertical">

    <WebView
        ohos:id="$+id:web_view"
        ohos:height="match_parent"
        ohos:width="match_parent"
        ohos:background_element="#FFFFFF"/>
</DirectionalLayout>
```

2. 在 `MainAbility.java` 中增加获取权限代码：

```
// MainAbility.java
import ohos.aafwk.ability.Ability;
import ohos.aafwk.content.Intent;
import ohos.agp.components.webengine.WebView;
```

```
import ohos.agp.components.webengine.WebConfig;
import ohos.agp.components.Component;
import ohos.bundle.IBundleManager;

public class MainAbility extends Ability {
    private WebView webView;
    private static final int PERMISSION_REQUEST_CODE = 1001;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        // 请求必要的权限
        requestPermissions();
        // 初始化WebView
        initWebView();
    }

    private void requestPermissions() {
        String[] permissions = {
            "ohos.permission.MICROPHONE",
            "ohos.permission.CAMERA"
        };
        requestPermissionsFromUser(permissions, PERMISSION_REQUEST_CODE);
    }

    private void initWebView() {
        // 获取 WebView 实例
        webView = (WebView) findComponentById(ResourceTable.Id_web_view);
        // 配置 WebView
        WebConfig webConfig = webView.getWebConfig();
        // 启用 JavaScript
        webConfig.setJavaScriptPermit(true);
        });
        // 加载 URL
        webView.loadUrl("https://www.example.com");
    }
}
```

3. 在 `config.json` 中添加必要的权限声明：

```
{
    "app": {
        "bundleName": "com.example.webview",
        "vendor": "example",
        "version": {
            "code": 1000000,
            "name": "1.0.0"
        }
    },
    "deviceConfig": {},
    "module": {
        "requestPermissions": [
            {
```

```
                "name": "ohos.permission.INTERNET"
            },
            {
                "name": "ohos.permission.CAMERA"
            },
            {
                "name": "ohos.permission.MICROPHONE"
            }
        ]
    }
}
```

iOS 原生 App 使用 WebView

1. 打开 `ViewController.swift` 文件。
2. 导入 WebKit 框架：

```
import UIKit
import WebKit
```

3. 在 ViewController 类中添加 WebView 并加载网页：

```
class ViewController: UIViewController, WKNavigationDelegate {
    var webView: WKWebView!
    override func viewDidLoad() {
        super.viewDidLoad()
        // 创建 WebView
        webView = WKWebView(frame: self.view.frame)
        webView.navigationDelegate = self
        self.view.addSubview(webView)
        // 加载网页
        if let url = URL(string: "https://www.example.com") {
            let request = URLRequest(url: url)
            webView.load(request)
        }
    }
}
```

4. 确保您的应用有访问相机和麦克风的权限。您可以在 `Info.plist` 文件中添加以下键值对：

```
<key>NSCameraUsageDescription</key>
<string>We need access to your camera to use certain features.</string>
<key>NSMicrophoneUsageDescription</key>
<string>We need access to your microphone to use certain features.</string>
```

5. 如果您需要在运行时请求权限，可以在 ViewController 中添加权限请求的代码：

```
import UIKit
import WebKit

class ViewController: UIViewController {
```

```
    private var webView: WKWebView!

    override func viewDidLoad() {
        super.viewDidLoad()
        setupWebView()
        requestPermissions()
        loadWebContent()
    }

    private func setupWebView() {
        webView = WKWebView(frame: view.frame)
        webView.navigationDelegate = self
        view.addSubview(webView)
    }

    private func loadWebContent() {
        guard let url = URL(string: "https://www.example.com") else {
            return
        }
        let request = URLRequest(url: url)
        webView.load(request)
    }

    private func requestPermissions() {
        requestCameraPermission()
        requestMicrophonePermission()
    }

    private func requestCameraPermission() {
        let status = AVCaptureDevice.authorizationStatus(for: .video)
        switch status {
        case .authorized:
            print("Camera access granted")
        case .notDetermined:
            AVCaptureDevice.requestAccess(for: .video) { [weak self] granted in
                DispatchQueue.main.async {
                    if granted {
                        print("Camera access granted")
                    } else {
                        self?.showPermissionDeniedAlert(for: "Camera")
                    }
                }
            }
        case .denied, .restricted:
            showPermissionDeniedAlert(for: "Camera")
        @unknown default:
            print("Unknown camera access status")
        }
    }

    private func requestMicrophonePermission() {
        let status = AVCaptureDevice.authorizationStatus(for: .audio)
        switch status {
        case .authorized:
```

```
            print("Microphone access granted")
        case .notDetermined:
            AVCaptureDevice.requestAccess(for: .audio) { [weak self] granted in
                DispatchQueue.main.async {
                    if granted {
                        print("Microphone access granted")
                    } else {
                        self?.showPermissionDeniedAlert(for: "Microphone")
                    }
                }
            }
        case .denied, .restricted:
            showPermissionDeniedAlert(for: "Microphone")
        @unknown default:
            print("Unknown microphone access status")
        }
    }

    private func showPermissionDeniedAlert(for device: String) {
        let alert = UIAlertController(
            title: "\(device) Access Required",
            message: "Please enable \(device.lowercased()) access in Settings to use this
feature.",
            preferredStyle: .alert
        )

        alert.addAction(UIAlertAction(title: "Settings", style: .default) { _ in
            if let settingsURL = URL(string: UIApplication.openSettingsURLString) {
                UIApplication.shared.open(settingsURL)
            }
        })

        alert.addAction(UIAlertAction(title: "Cancel", style: .cancel))

        present(alert, animated: true)
    }
}

extension ViewController: WKNavigationDelegate {
    func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
        print("Web page loaded successfully")
    }

    func webView(_ webView: WKWebView, didFail navigation: WKNavigation!, withError error:
Error) {
        print("Failed to load web page: \(error.localizedDescription)")
    }
}
```

常见问题

WebView 是否支持屏幕共享?

目前 WebView 不支持屏幕共享。

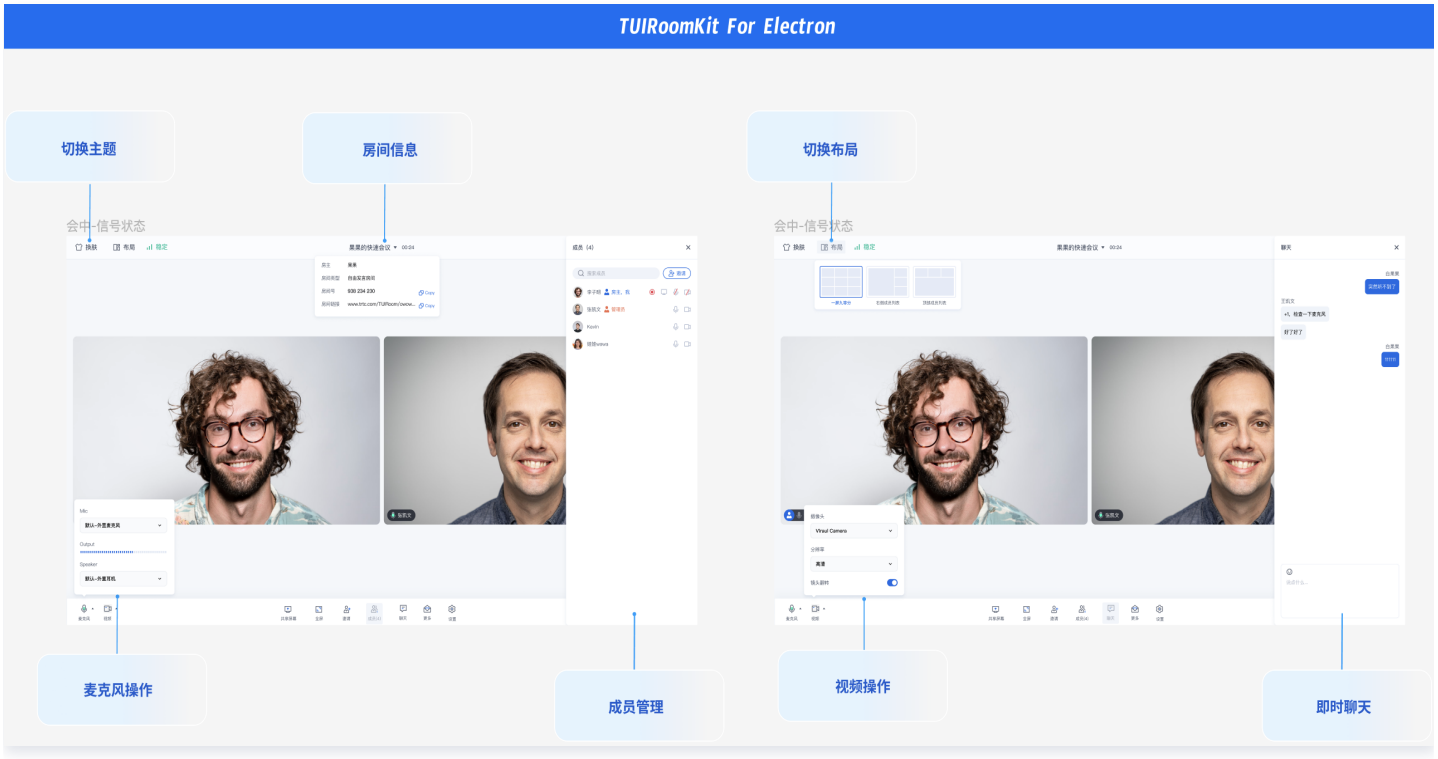
其他文档

- TUIRoomKit
- TUIRoom Demo 快速跑通
- 设置昵称、头像
- Web&Electron
- 界面定制 (TUIRoomKit)
- 常见问题

Electron (Windows&Mac)

最近更新时间：2024-11-06 09:31:52

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 (Electron) 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。



TUIRoomKit Demo 体验

您可以单击下载 Mac OS版 或 Windows版 体验 TUIRoomKit（Electron）更多功能。

环境准备

- Node.js 版本：Node.js ⩾ 16.19.1（推荐使用官方 LTS 版本，npm 版本请与 node 版本匹配）。
- Electron 版本: Electron < 24.8.8
- **npm 包集成**
  - Vue3 开发环境，集成 @tencentcloud/room-electron-vue3 NPM 包。
  - Vue2.7 开发环境：集成 @tencentcloud/roomkit-electron-vue2.7 NPM 包。
- **源码集成**
  - Vue3 + TypeScript 开发环境：从 @tencentcloud/room-electron-vue3 NPM 包拷贝源码。
  - Vue2.7 + TypeScript 开发环境：从 @tencentcloud/roomkit-electron-vue2.7 NPM 包拷贝源码。
- **模版工程**
  - Electron + vite + vue3 electron-vite-vue

集成 TUIRoomKit 组件

如果您没有 Vue 项目，可以前往 Github 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Electron 示例工程。另外您如果需要在已有项目中进行集成，请按照以下步骤进行接入。

> ⚠ 注意：
> 若您通过下载模版工程 electron-vite-vue 的方式来进行集成，您需要切换模版工程至 v1.0.0，可参考如下指令：

```
git clone https://github.com/electron-vite/electron-vite-vue.git
cd electron-vite-vue
git checkout v1.0.0
```

步骤一：安装依赖

Vue3

```
npm install @tencentcloud/roomkit-electron-vue3 pinia@2.0.24 --save
```

Vue2

```
# 注意此处要求 vue版本 >= 2.7.16，如安装失败请检查您的 vue 版本是否支持
npm install @tencentcloud/roomkit-electron-vue2.7 pinia
```

步骤二：项目工程配置

1. 调整内容安全策略

在 Mac 系统中，默认的内容安全策略设置项会在加载 TUIRoomKit 组件时报错，为避免 TUIRoom 页面无法加载或者接口调用报错，可以调整内容安全策略。调整文件为 `packages/renderer/index.html` 文件。

```
<head>
  <meta charset="UTF-8" />
  <link rel="icon" href="/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- Mac下修改此设置 -->
  <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline'
'unsafe-eval'; worker-src 'self' blob:;"/>
  <title>Vite App</title>
</head>
```

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。

Vue3

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
```

```
// 注册 pinia
app.use(createPinia());
app.mount('#app')
```

Vue2

```
// 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 vite.config.ts

为了统一代码风格并在 UI 层通过 import 方式引入 trtc-electron-sdk（否则必须通过 require 方式引入），您需要在 `packages/renderer/vite.config.ts` 中进行配置。请参照以下配置项替换 resolve 中的内容，具体可参考文件 packages/renderer/vite.config.ts 。

> **注意：**
> 若您的工程采用的是 vite 打包方式，请按如下步骤进行配置。原因是 vite 默认只支持 ES6 模块，而 trtc-electron-sdk 需要对接 Node.js API，其是一个Common JS 模块，此步骤的目的是为了进行模块类型的兼容。若您的工程采用的是 webpack 打包方式，则可跳过**配置 vite.config.ts** 这一步。

```
// 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;
        `,
    }),
  ]
});
```

步骤三：引用 TUIRoomKit 组件

> **说明：**
> 引入 ConferenceMainView 组件，组件默认处于 常驻模式（组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态）。

Vue3

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script setup>
```

```
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue3';
</script>
```

Vue2

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script>
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
};
</script>
```

步骤四：登录 TUIRoomKit 组件

开启会议前需要调用 login 接口进行登录。获取 sdkAppId、userId、userSig 可参见 开通服务 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.login({
    sdkAppId: 0,  // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
    userId: '',   // 用户在您业务中的唯一标示 Id
    userSig: '',  // 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
});
```

参数	类型	说明
userID	String	客户根据自己的业务自定义用户 ID，只允许包含大小写英文字母(a-z A-Z)、数字(0-9)及下划线和连词符。
sdkAppID	int	在 实时音视频 TRTC 控制台 创建的音视频应用的唯一标识 SDKAppID。
secretKey	String	在 实时音视频 TRTC 控制台 创建的音视频应用的 SDKSecretKey。
userSig	String	一种安全保护签名，用于对用户进行登录鉴权认证，确认用户是否真实，阻止恶意攻击者盗用您的云服务使用权。

说明：

- **开发环境**：如果您正在本地开发调试阶段，可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig。该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。
- **生产环境**：如果您的项目要发布上线，请采用 服务端生成 UserSig 的方式。

步骤五：发起新的会议

会议主持人可以通过调用 start 接口来发起一场新的会议，其他参会议者可以参见 步骤六 的描述，调用 join 接口加入该会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const startConference = async () => {
    await conference.login({
      sdkAppId: 0,  // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
      userId: '',   // 用户在您业务中的唯一标示 Id
      userSig: '',  // 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
    });
    await conference.start('123456', {
      roomName: 'TestRoom',
      isSeatEnabled: false,
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
}
startConference()
```

步骤六：进入已有会议

参与者可以通过调用 join 接口，填写对应的 roomId 参数，来加入由会议主持人在 步骤五 中发起的会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const joinConference = async () => {
    await conference.login({
      sdkAppId: 0,  // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
      userId: '',   // 用户在您业务中的唯一标示 Id
      userSig: '',  // 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
    });
    await conference.join('123456', {
      isOpenCamera: false,
      isOpenMicrophone: false,
    });
}
joinConference()
```

开发环境运行

1. 执行开发环境命令。（此处以 vue3 + vite 默认项目为例，不同项目 dev 指令可能不同，请根据您自己的项目进行调整）

```
npm run dev
```

> ⚠ **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。

2. 体验 TUIRoomKit 组件功能。

生产环境部署

项目进行打包

```
npm run build
```

> **说明：**
> 以模板工程接入为例，执行上述打包命令之后，会在您的项目根目录下生成 release 文件夹，其中包含您所需要的对应安装包。

常见问题

若您在步骤一安装依赖时出现 vue 版本不匹配等问题，如何解决？

需要在package.json 文件中将vue版本固定为 3.3.13

如果您在运行阶段产生了app crash，如何解决？

原因可能是获取不到摄像头和麦克风的权限而引起的。
解决方案：

1. 请检查您的摄像头或麦克风设备使用是否正常，有无被其他应用占用。
2. 在 /packages/main/index.ts 中，注释掉 `checkAndApplyDevicePrivilege` 方法的调用，即可跳过设备权限的检查。

若您在运行阶段或打包阶段遇到如下图报错，如何解决？

```
error TS6504: File 'D:/electron-vite-vue/packages/renderer/src/App.vue.__VLS_template.jsx' is a JavaScript file. Did you mean to enable the 'allowJs' option?
  The file is in the program because:
    Root file specified for compilation

error TS6504: File 'D:/electron-vite-vue/packages/renderer/src/App.vue.js' is a JavaScript file. Did you mean to enable the 'allowJs' option?
  The file is in the program because:
    Root file specified for compilation
```

需要在 tsconfig.json 文件中启用 allowJs 选项：

```
// tsconfig.json
{
  "compilerOptions": {
    "allowJs": true
    // ...其他选项
  }
  // ...其他配置
}
```

若您在运行或打包阶段遇到如下图报错，如何解决？

```
Uncaught ReferenceError: exports is not defined                    index.a12c1be5.js:1
    at index.a12c1be5.js:1:706
```

需要您在 index.html 文件中进行如下增加如下内容：

```
// index.html
<head>
  ....
  <script>const exports = module.exports;</script>
<head>
```

其他文档

- TUIRoomKit
- TUIRoom Demo 快速跑通
- 设置昵称、头像
- 预定会议
- 界面定制 (TUIRoomKit)
- 常见问题

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。

Flutter

最近更新时间：2024-11-12 16:06:52

本文将指导您如何在快速完成 `TUIRoomKit` 组件的接入。遵循本文档的步骤，您将在一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	Flutter 3.7.0 及以上版本，Dart 2.19.0 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 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 开通服务 。

步骤二：集成 TUIRoomKit 组件

在您的工程 `pubspec.yaml` 文件中，添加 tencent_conference_uikit 插件依赖。

```
dependencies:
    tencent_conference_uikit: 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 tencent_conference_uikit 组件使用了 `GetX` 状态管理库的相关功能，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。

```
//此步骤需导入get包后再操作。由于tencent_conference_uikit中已有对get的依赖，您无需再您的pubspec.yaml
中进行额外的配置。
import 'package:get/get.dart';

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(  // 使用GetMaterialApp来代替MaterialApp
      // 您原先的MaterialApp内容
    );
  }
}
```

- 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3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 SDKSecretKey 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（ `SDKSecretKey` ），`UserSig` 是用 `SDKSecretKey` 把 `SDKAppID` 、`UserID` 以及过期时间等信息加密得来的，而不是直接把 `SDKSecretKey` 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 `SDKSecretKey` 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SDKSecretKey，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。

登录弹幕聊天（可选）

Flutter **TUIRoomKit**（**tencent_conference_uikit**）自**2.4.1**版本开始引入**弹幕聊天功能**。如您需要使用弹幕聊天功能，需要完成以下初始化及登录。（如您还需要使用 会中聊天 页面，可**忽略**此步骤，完成会中聊天的 初始化及登录 即可）：

```
import 'package:tencent_cloud_chat_sdk/enum/V2TimSDKListener.dart';
import 'package:tencent_cloud_chat_sdk/enum/log_level_enum.dart';
import 'package:tencent_cloud_chat_sdk/models/v2_tim_callback.dart';
import 'package:tencent_cloud_chat_sdk/tencent_im_sdk_plugin.dart';

// 初始化
var initResult = await TencentImSDKPlugin.v2TIMManager.initSDK(
    sdkAppID: SDKAPPID,                    // 请替换为您的SDKAPPID
    loglevel: LogLevelEnum.V2TIM_LOG_INFO,  // 日志登记等级
    listener: V2TimSDKListener(),           // 事件监听器。使用弹幕聊天时，这里传空对象即可。
);

if (initResult.code == 0) { // 初始化成功
     // 登录
    V2TimCallback imLoginResult = await TencentImSDKPlugin.v2TIMManager.login(
      userID: 'userId',    // 请替换为您的UserID
      userSig: 'userSig', // 请替换为您的userSig
    );
}
```

> **说明：**
>
> 弹幕聊天功能在 TUIRoomKit 中默认开启。如您无需弹幕聊天功能，您无需进行上述初始化及登录操作，并可以通过**底部栏 > 设置 > 开启弹幕聊天**选项来关闭弹幕聊天。

步骤五：使用组件

设置用户信息（可选）

通过调用 `TUIRoomEngine` 的 `setSelfInfo` 即可设置当前用户的用户名、头像。

```
import 'package:rtc_room_engine/rtc_room_engine.dart';

TUIRoomEngine.setSelfInfo(userName, avatarURL);
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像 URL

开始快速会议

通过调用 `ConferenceSession` 的 `quickStart` 方法，便可以开始快速会议。

```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

ConferenceSession.newInstance('roomId')    //您的room id
  ..onActionSuccess = _quickStartSuccess
  ..onActionError = _quickStartError
  ..quickStart();

void _quickStartSuccess() {
  //您可以在开始快速会议的成功回调中，自行导航至会议页面。
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ConferenceMainPage(),
    ),
  );
}

void _quickStartError(ConferenceError error, String message) {
  debugPrint("code: $error message: $message");
}
```

加入会议

通过调用 `ConferenceSession` 的 `join` 方法，便可以加入指定会议。

```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

ConferenceSession.newInstance('roomId')    //您的room id
  ..onActionSuccess = _joinSuccess
  ..onActionError = _joinError
  ..join();

void _joinSuccess() {
  //您可以在加入会议的成功回调中，自行导航至会议页面。
  Navigator.push(
```

```
      context,
      MaterialPageRoute(
        builder: (context) => ConferenceMainPage(),
      ),
    );
  }

  void _joinError(ConferenceError error, String message) {
    debugPrint("code: $error message: $message");
  }
```

说明：
上述的示例代码中，只展示了最简单的创建会议/加入会议的方式。如果您需要在进会前完成更多设置，请参见 会前控制 。

运行

运行代码后，您将看到如下的会议界面。

更多功能

- 设置昵称、头像
- 预定会议

- 会中聊天
- RoomKit API
- 弹幕聊天

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.

屏幕共享（TUIRoomKit）Android&iOS&Flutter

最近更新时间：2025-07-03 15:42:56

本文档将详细介绍 `TUIRoomKit` 屏幕共享功能，帮助您更好地掌握 `TUIRoomKit` 会议过程中的屏幕共享相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。 **若 TUIRoomKit 的 UI 交互不满足您的产品需求，您有自己的交互和业务逻辑需要自定义实现屏幕共享 相关的交互功能，可以接入 TUIRoomEngineSDK，并参见 关键代码 相关调用来实现您的需求。**

功能介绍

TUIRoomKit 支持屏幕共享，用户可以在进入房间之后分享自己的屏幕，房间内的其他成员可以观看正在分享中的屏幕。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

使用说明

`Android&iOS&Flutter` 端用户进入房间后，通过点击底部工具栏**共享按钮**，可以实现屏幕共享操作。

屏幕分享

功能接入

在接入屏幕共享功能时，您可能需要进行一些配置。不同平台的配置方法请参见：

Android

由于 Android 系统隐私策略的更改，在 Android 10 及以上版本 App 若使用录屏等功能需要在前台 Service 中进行，否则录屏/屏幕共享时系统将报错或 App 被系统终止。**在我们的组件中，会自动为您开启该 Android Foreground Service，您只需要点击屏幕共享按钮**，同意相关隐私政策后即可进行共享。
Android Foreground Service 相关代码可参见我们的 代码示例 。

iOS

iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

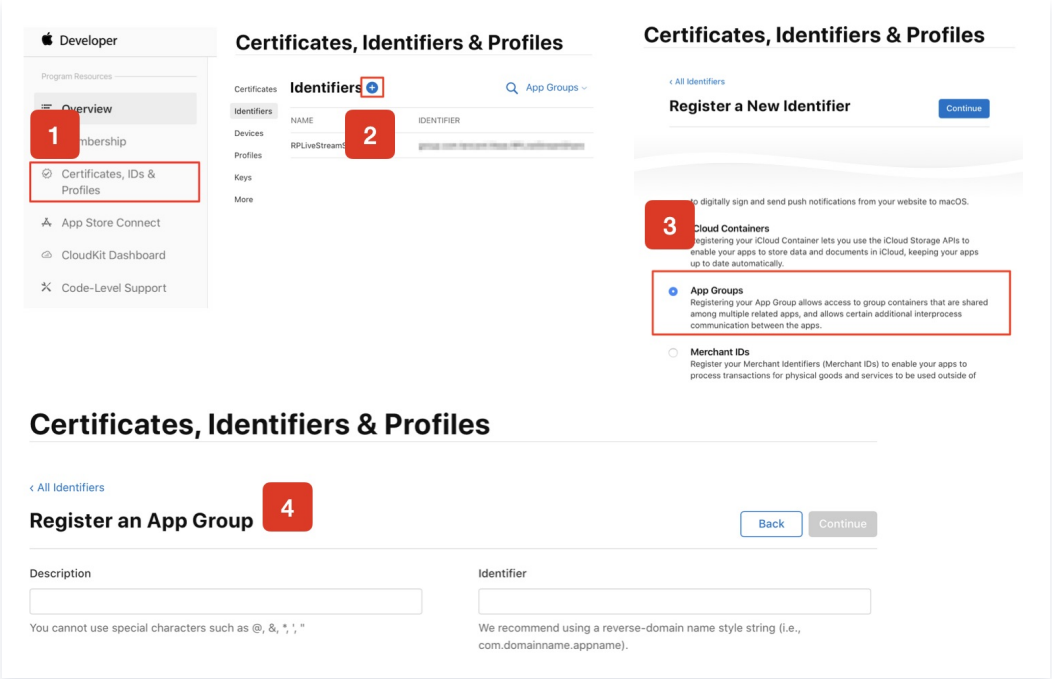
1. 创建 App Group，并在 XCode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以同主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 TXLiteAVSDK_ReplayKitExt.framework 。
3. 点击屏幕分享按钮，开始分享您的屏幕。

> ⚠ **注意：**
> 如果跳过步骤1，也就是不配置 App Group（接口传 null），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

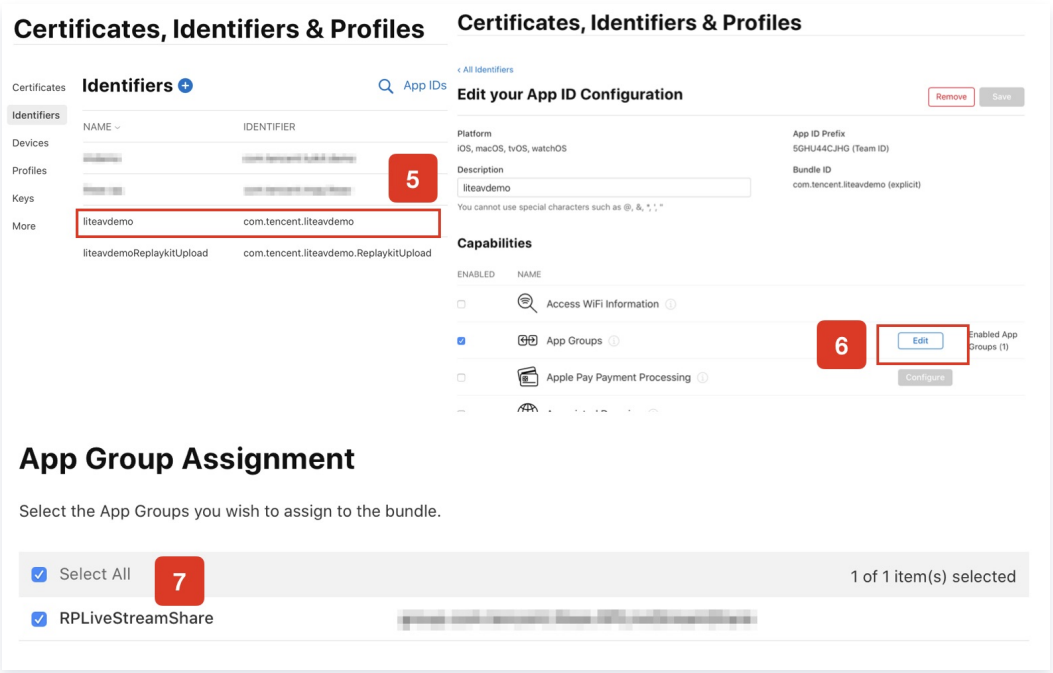
步骤1：创建 App Group（可选）

使用您的账号 登录，进行以下操作，注意完成后需要重新下载对应的 Provisioning Profile。

1. 单击 Certificates, IDs & Profiles。
2. 在右侧的界面中单击加号。
3. 选择 App Groups，单击 Continue。
4. 在弹出的表单中填写 Description 和 Identifier，其中 Identifier 需要传入接口中的对应的 AppGroup 参数。完成后单击 Continue。



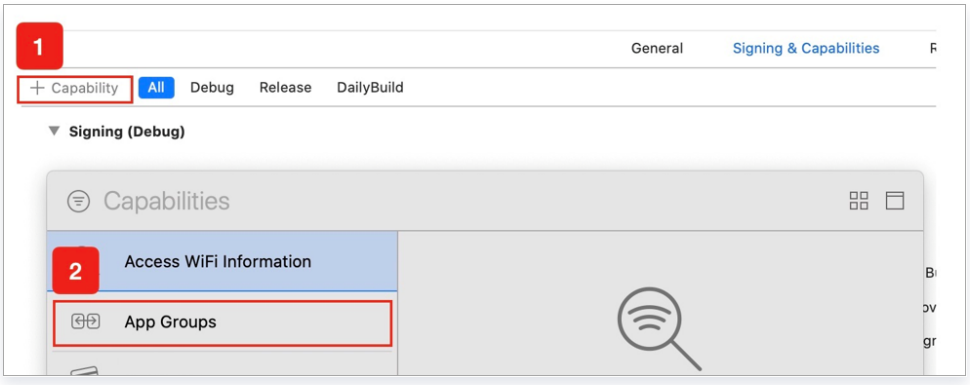
5. 回到 Identifier 页面，左上边的菜单中选择 App IDs，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。
6. 选中 App Groups 并单击 Edit。
7. 在弹出的表单中选择您之前创建的 App Group，单击 Continue 返回编辑页，单击 Save 保存。



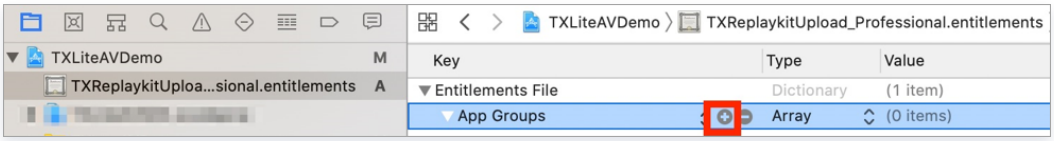
8. 重新下载 Provisioning Profile 并配置到 XCode 中。

步骤2：创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 File > New > Target...，选择 Broadcast Upload Extension。
2. 在弹出的对话框中填写相关信息，不用勾选 Include UI Extension，单击 Finish 完成创建。
3. 将下载到的 SDK 压缩包中的 TXLiteAVSDK_ReplayKitExt.framework 拖动到工程中，勾选刚创建的 Target。
4. 如果选择执行 步骤1，需要在新增加的 Target 中，依次单击 + Capability，双击 App Groups，如下图：



操作完成后，会在文件列表中生成一个名为 Target.entitlements 的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



5. 选中主 App 的 Target，并按照上述步骤对主 App 的 Target 做同样的处理。

6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。如果执行了 步骤1，请将代码中的 APPGROUP 替换为之前创建的 App Group Identifier，然后找到 TUIRoomKit中的 EngineManager.swift 文件，将其中的appGroupString 也设置为 App Group Identifier。如果 步骤1 被跳过，则不需要修改 APPGROUP。

Swift

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler, TXReplayKitExtDelegate {

    let recordScreenKey = Notification.Name.init("TRTCRecordScreenKey")

    override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
        // User has requested to start the broadcast. Setup info from the UI extension
can be supplied but optional.
        TXReplayKitExt.sharedInstance().setup(withAppGroup: APPGROUP, delegate: self)
    }

    override func broadcastPaused() {
        // User has requested to pause the broadcast. Samples will stop being delivered.
    }

    override func broadcastResumed() {
        // User has requested to resume the broadcast. Samples delivery will resume.
    }

    override func broadcastFinished() {
        // User has requested to finish the broadcast.
        TXReplayKitExt.sharedInstance().broadcastFinished()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason: TXReplayKitExtReason) {
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误（SDK 版本号不相符合）"
            break
        default:
            break
        }

        let error = NSError(domain: NSStringFromClass(self.classForCoder), code: 0,
userInfo: [NSLocalizedFailureReasonErrorKey:tip])
```

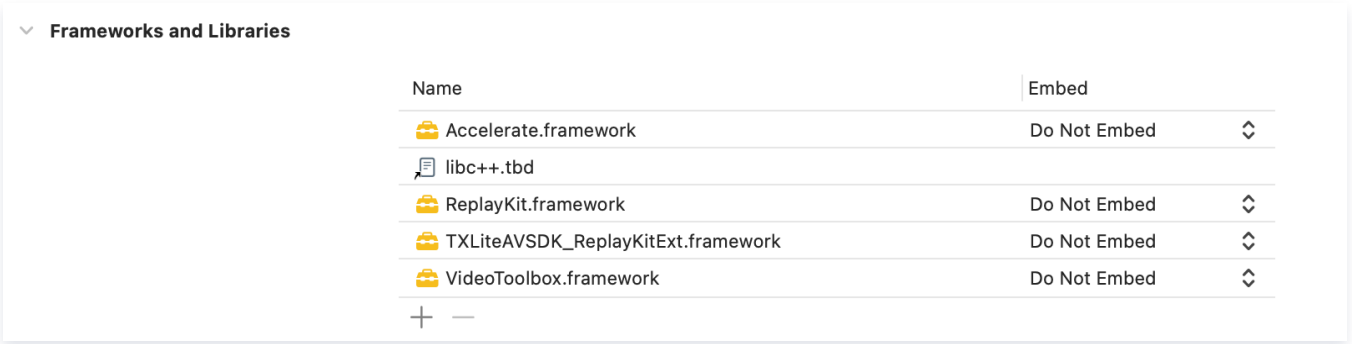
```
        finishBroadcastWithError(error)
    }

    override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with
sampleBufferType: RPSampleBufferType) {
        switch sampleBufferType {
        case RPSampleBufferType.video:
            // Handle video sample buffer
            TXReplayKitExt.sharedInstance() .sendVideoSampleBuffer(sampleBuffer)
            break
        case RPSampleBufferType.audioApp:
            // Handle audio sample buffer for app audio
            break
        case RPSampleBufferType.audioMic:
            // Handle audio sample buffer for mic audio
            break
        @unknown default:
            // Handle other sample buffer types
            fatalError("Unknown type of sample buffer")
        }
    }
}
```

OC

```
#import "SampleHandler.h"
@import TXLiteAVSDK_ReplayKitExt;
#define APPGROUP @"group.com.tencent.comm.trtc.demo"

@protocol TXReplayKitExtDelegate;
@interface SampleHandler()
@end

@implementation SampleHandler

- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *> *)setupInfo {
    [[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP delegate:self];
}

- (void)broadcastPaused {
    // User has requested to pause the broadcast. Samples will stop being delivered.
}

- (void)broadcastResumed {
    // User has requested to resume the broadcast. Samples delivery will resume.
}

- (void)broadcastFinished {
    [[TXReplayKitExt sharedInstance] broadcastFinished];
    // User has requested to finish the broadcast.
}

#pragma mark - TXReplayKitExtDelegate
```

```
- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:(TXReplayKitExtReason)reason
{
    NSString *tip = @"";
    switch (reason) {
        case TXReplayKitExtReasonRequestedByMain:
            tip = @"屏幕共享已结束";
            break;
        case TXReplayKitExtReasonDisconnected:
            tip = @"应用断开";
            break;
        case TXReplayKitExtReasonVersionMismatch:
            tip = @"集成错误（SDK 版本号不相符合）";
            break;
    }

    NSError *error = [NSError errorWithDomain:NSStringFromClass(self.class)
                                         code:0
                                     userInfo:@{
        NSLocalizedFailureReasonErrorKey:tip
    }];
    [self finishBroadcastWithError:error];
}

- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:
(RPSampleBufferType)sampleBufferType {
    switch (sampleBufferType) {
        case RPSampleBufferTypeVideo:
            [[TXReplayKitExt sharedInstance] sendSampleBuffer:sampleBuffer
withType:sampleBufferType];
            break;
        case RPSampleBufferTypeAudioApp:
            // Handle audio sample buffer for app audio
            break;
        case RPSampleBufferTypeAudioMic:
            // Handle audio sample buffer for mic audio
            break;
        default:
            // Handle other sample buffer types
            break;
    }
}

@end
```

7. 点击新创建的 Target，选择 General 选项并且在 Frameworks and Libraries 中添加下面的库：



Flutter

Android：由于 Android 系统隐私策略的更改，在 Android 10 及以上版本 App 若使用录屏等功能需要在前台 Service 中进行，否则录屏/屏幕共享时系统将报错或 App 被系统终止。**在我们的组件中，会自动为您开启该 Android Foreground Service，您只需要点击屏幕共享按钮**，同意相关隐私政策后即可进行共享。
Android Foreground Service 相关代码可参见我们的 代码示例 。

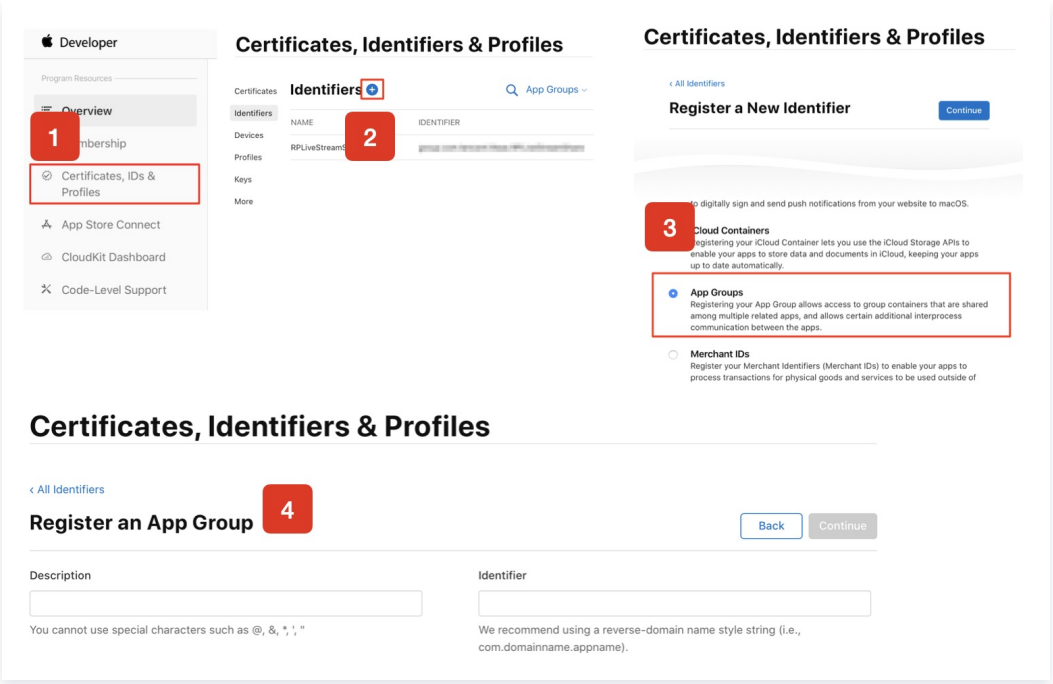
iOS：iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 App Group，并在 XCode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以同主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 `TXLiteAVSDK_ReplayKitExt.framework` 。
3. 在组件中使用您的 App Group 和 BroadCast Upload Extension name 替换我们应用的 App Group 和 BroadCast Upload Extension name ，代码位置请点击 文件链接 。
4. 点击屏幕分享按钮，开始分享您的屏幕。

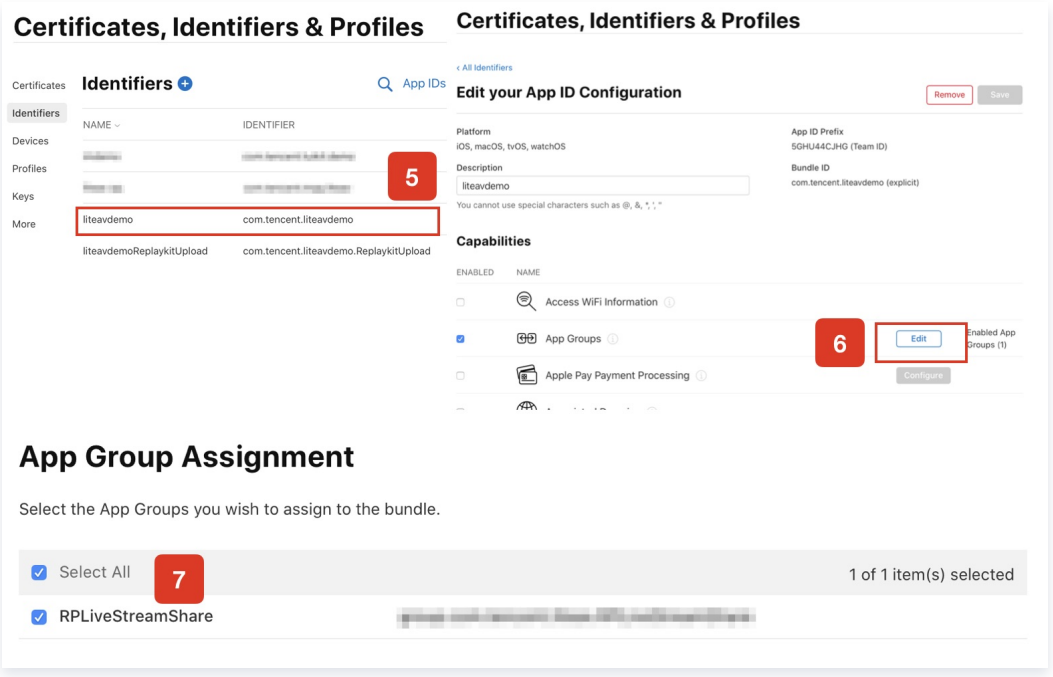
> **注意：**
> 如果跳过步骤1，也就是不配置 App Group（接口传 null），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

步骤1：创建 App Group
使用您的账号 登录，进行以下操作，注意完成后需要重新下载对应的 Provisioning Profile。

1. 单击 Certificates, IDs & Profiles。
2. 在右侧的界面中单击加号。
3. 选择 App Groups，单击 Continue。
4. 在弹出的表单中填写 Description 和 Identifier，其中 Identifier 需要传入接口中的对应的 AppGroup 参数。完成后单击 Continue。



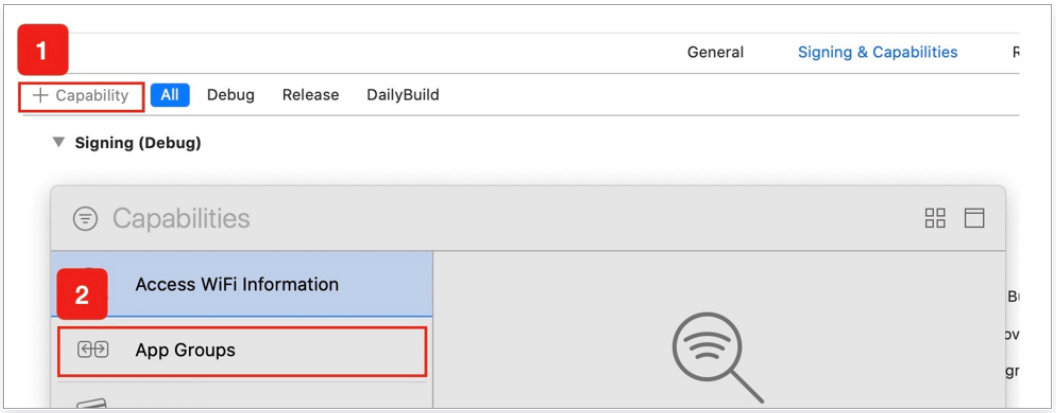
5. 回到 Identifier 页面，左上边的菜单中选择 App IDs，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。
6. 选中 App Groups 并单击 Edit。
7. 在弹出的表单中选择您之前创建的 App Group，单击 Continue 返回编辑页，单击 Save 保存。



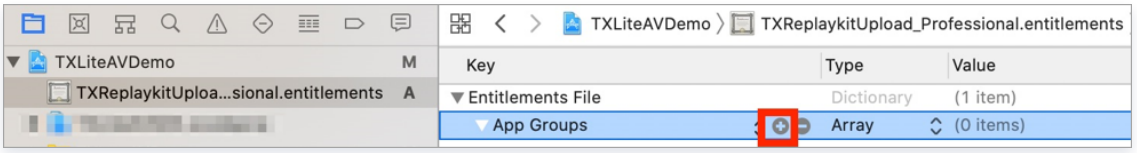
8. 重新下载 Provisioning Profile 并配置到 XCode 中。

步骤2：创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 File > New > Target...，选择 Broadcast Upload Extension。
2. 在弹出的对话框中填写相关信息，不用勾选 Include UI Extension，单击 Finish 完成创建。
3. 将下载到的 SDK 压缩包中的 TXLiteAVSDK_ReplayKitExt.framework 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击 + Capability，双击 App Groups，如下图：



操作完成后，会在文件列表中生成一个名为Target.entitlements。
的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



5. 选中主 App 的 Target ，并按照上述步骤对主 App 的 Target 做同样的处理。
6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。需将代码中的 APPGROUP 改为上文中的创建的 App Group Identifier。

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler, TXReplayKitExtDelegate {

    let recordScreenKey = Notification.Name.init("TRTCRecordScreenKey")

    override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
        // User has requested to start the broadcast. Setup info from the UI extension
can be supplied but optional.
        TXReplayKitExt.sharedInstance().setup(withAppGroup: APPGROUP, delegate: self)
    }

    override func broadcastPaused() {
        // User has requested to pause the broadcast. Samples will stop being
delivered.
    }

    override func broadcastResumed() {
        // User has requested to resume the broadcast. Samples delivery will resume.
    }

    override func broadcastFinished() {
        // User has requested to finish the broadcast.
        TXReplayKitExt.sharedInstance() .finishBroadcast()
    }
```

```
    func broadcastFinished(_ broadcast: TXReplayKitExt, reason: TXReplayKitExtReason)
{
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误（SDK 版本号不相符合）"
            break
        default:
            break
        }

        let error = NSError(domain: NSStringFromClass(self.classForCoder), code: 0,
userInfo: [NSLocalizedFailureReasonErrorKey:tip])
        finishBroadcastWithError(error)
    }

    override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with
sampleBufferType: RPSampleBufferType) {
        switch sampleBufferType {
        case RPSampleBufferType.video:
            // Handle video sample buffer
            TXReplayKitExt.sharedInstance() .sendVideoSampleBuffer(sampleBuffer)
            break
        case RPSampleBufferType.audioApp:
            // Handle audio sample buffer for app audio
            break
        case RPSampleBufferType.audioMic:
            // Handle audio sample buffer for mic audio
            break
        @unknown default:
            // Handle other sample buffer types
            fatalError("Unknown type of sample buffer")
        }
    }
}.
```

步骤3：在组件中使用您的 App Group 和 BroadCast Upload Extension name 替换我们应用的 App Group 和 BroadCast Upload Extension name，代码位置请点击 文件链接 。

```
  void _startScreenSharing() {
    if (_isOffSeatInSeatMode()) {
      return;
    }

    if (_store.isSharing.value &&
        _store.screenShareUser.userId.value !=
            _store.currentUser.userId.value) {
      makeToast(
        msg: 'otherUserScreenSharing'.roomTr,
      );
      return;
    }

    String appGroup = '';
    if (Platform.isIOS) {
      appGroup = 'com.tencent.TUIRoomTXReplayKit-Screen';
      ReplayKitLauncher.launchReplayKitBroadcast('TXReplayKit_Screen');
    }
    _engineManager.startScreenSharing(appGroup: appGroup);
  }
```

步骤4：点击屏幕分享按钮，开始分享您的屏幕。

关键代码

若您想自定义实现屏幕共享功能，请参见 TUIRoomEngine SDK：Android 、iOS&Mac、Flutter。

> **说明：**
> 如果您在使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

Web&Electron

最近更新时间：2024-12-13 16:58:32

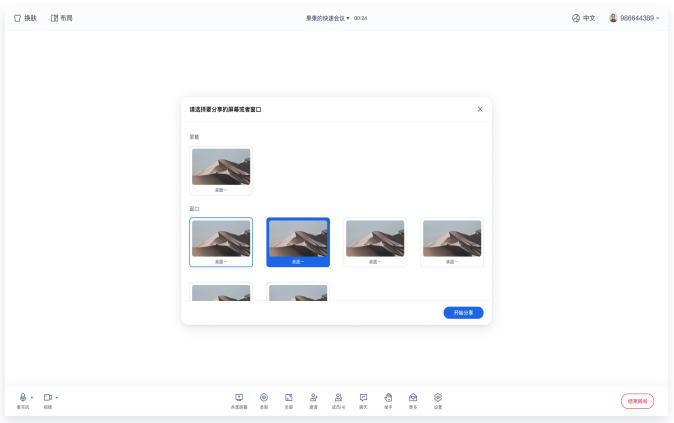
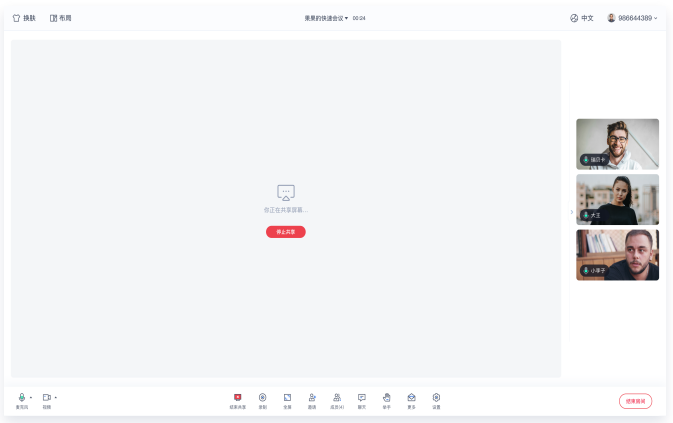
本文档将详细介绍 `TUIRoomKit` 屏幕共享功能，帮助您更好地掌握 `TUIRoomKit` 会议过程中的屏幕共享相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。

功能介绍

TUIRoomKit 支持屏幕共享，用户可以在进入房间之后分享自己的屏幕，房间内的其他成员可以观看正在分享中的屏幕。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

使用说明

`Web&Electron` 端用户进入房间后，通过点击底部工具栏共享按钮，可以实现屏幕共享操作。

共享屏幕选择窗口或屏幕	共享屏幕中
	

uni-app (客户端)

最近更新时间：2025-05-23 14:40:52

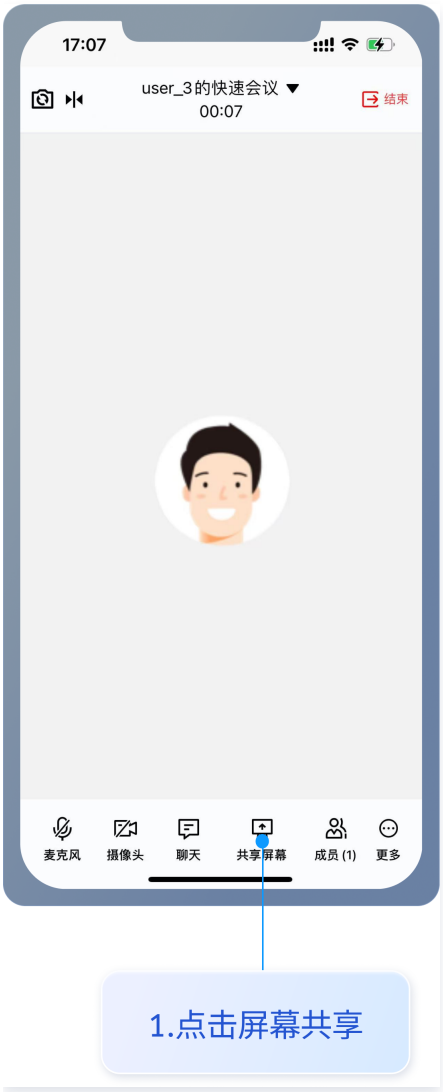
本文档将详细介绍 `TUIRoomKit` 屏幕共享功能，帮助您更好地掌握 `TUIRoomKit` 会议过程中的屏幕共享相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。

功能介绍

TUIRoomKit 支持屏幕共享，用户可以在进入房间之后分享自己的屏幕，房间内的其他成员可以观看正在分享中的屏幕。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

使用说明

`uniapp 客户端` 用户进入房间后，通过点击底部工具栏共享按钮，可以实现屏幕共享操作。



功能接入

在接入屏幕共享功能时，您可能需要进行一些配置。不同平台的配置方法请参见：

Android

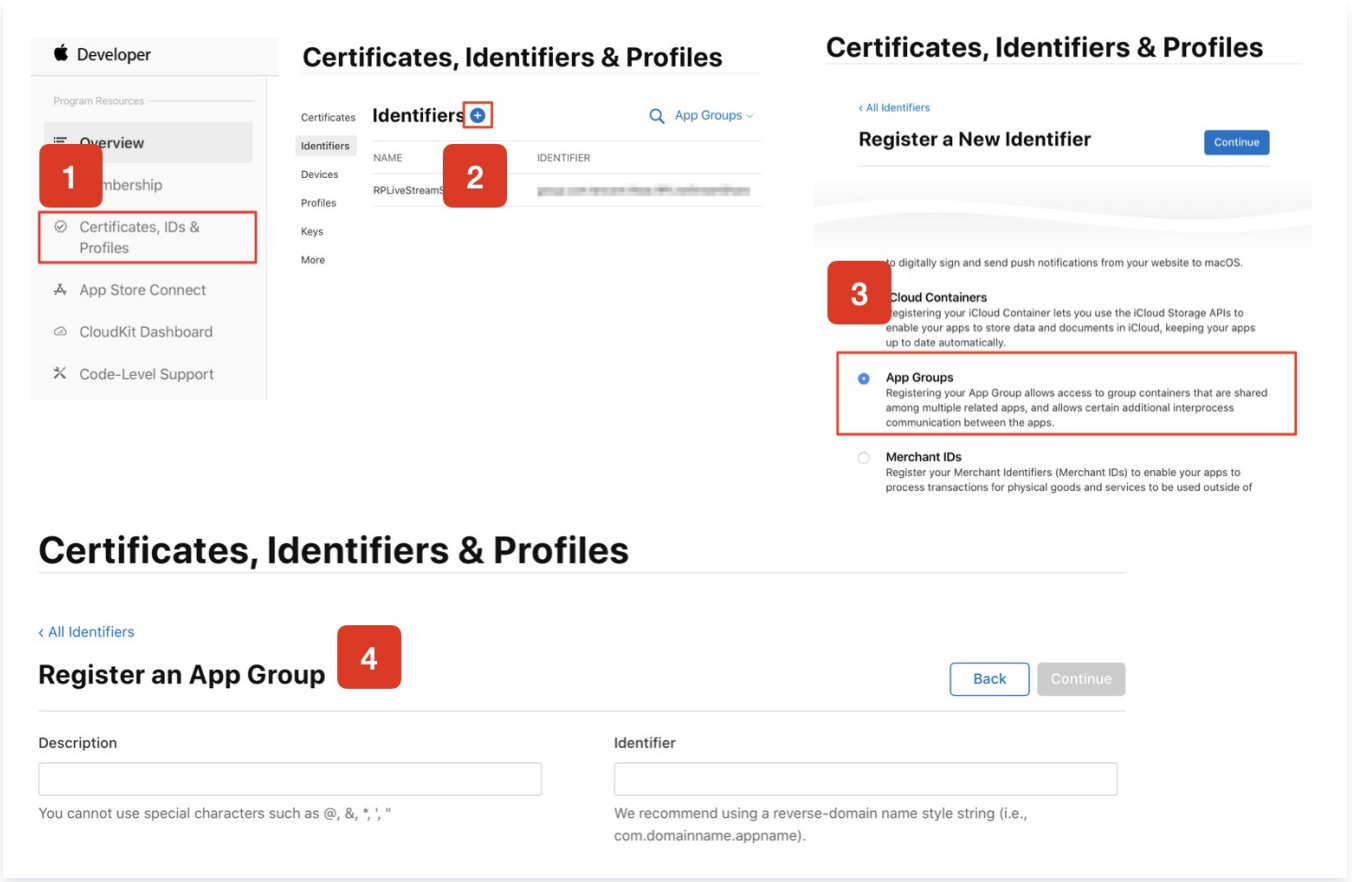
在我们的原生插件中，您只需要点击屏幕共享按钮，同意相关隐私政策后即可进行共享。

iOS

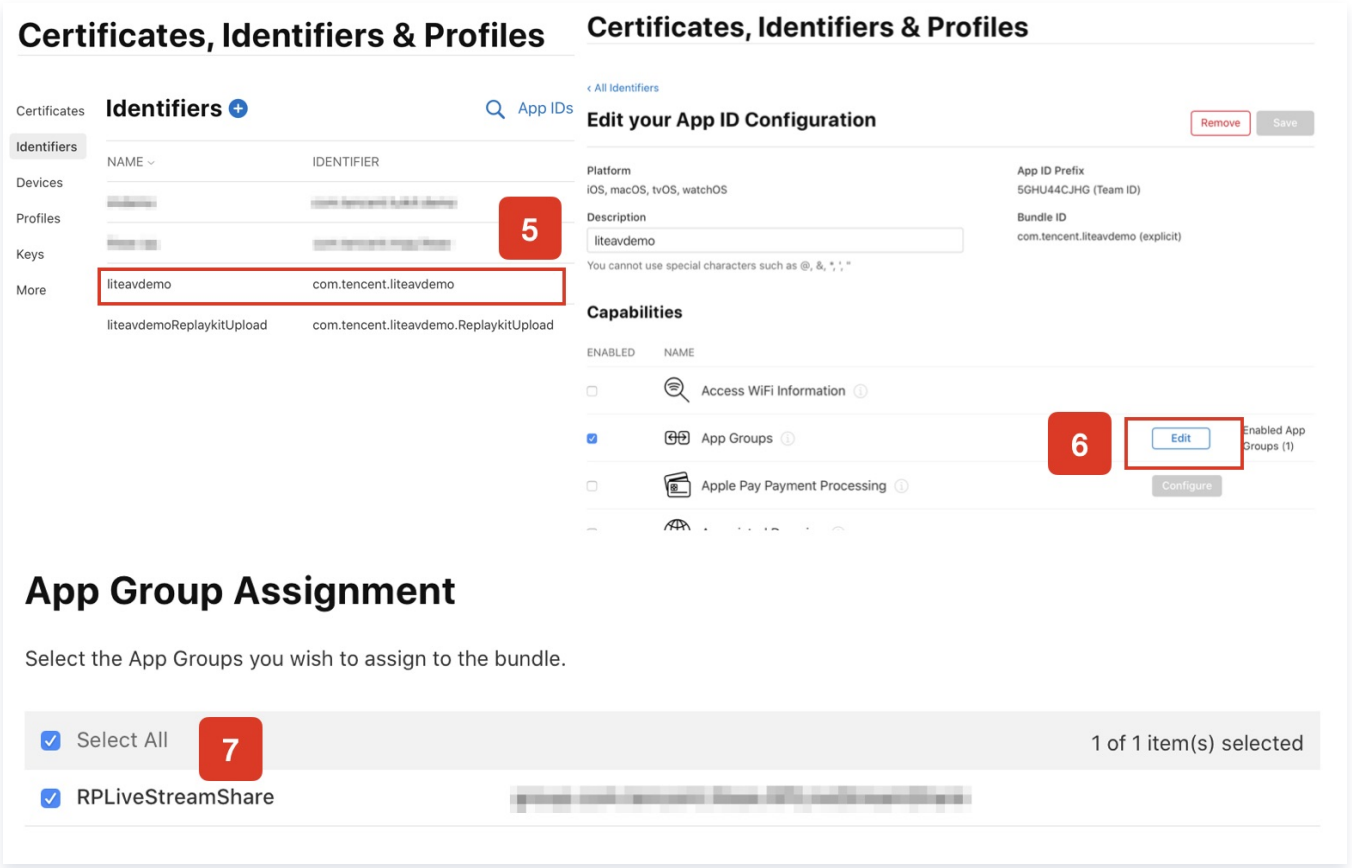
步骤1：创建 App Group

使用您的账号 登录，进行以下操作，注意完成后需要重新下载对应的 Provisioning Profile。

1. 单击 Certificates, IDs & Profiles。
2. 在右侧的界面中单击加号。
3. 选择 App Groups，单击 Continue。
4. 在弹出的表单中填写 Description 和 Identifier，其中 Identifier 需要传入接口中的对应的 AppGroup 参数。完成后单击 Continue。



5. 回到 Identifier 页面，左上边的菜单中选择 App IDs，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。
6. 选中 App Groups 并单击 Edit。
7. 在弹出的表单中选择您之前创建的 App Group，单击 Continue 返回编辑页，单击 Save 保存。

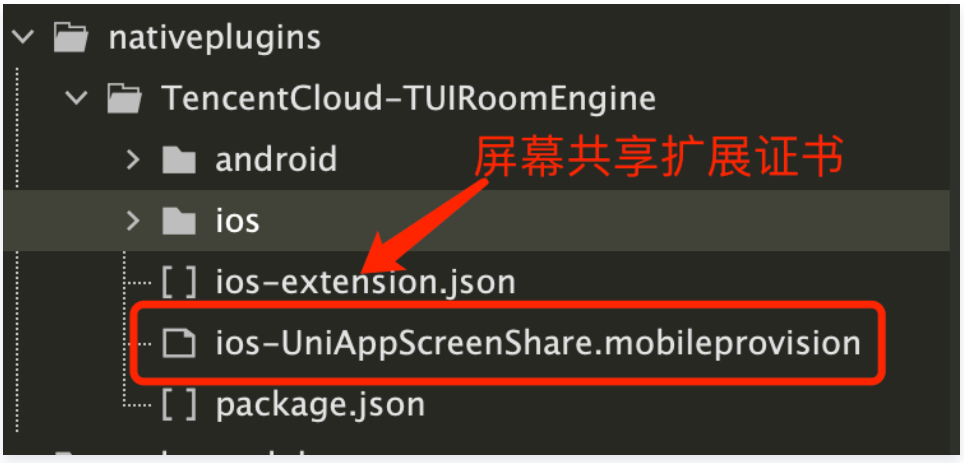


8. 重新下载 Provisioning Profile 并配置到 Hbuilder 中。

步骤2：配置屏幕共享扩展。（目录层级和示例保持一致）

配置扩展能力基础指引请参见 uniapp 官网文档 。

1. 将屏幕共享证书放在原生插件目录下，具体路径：**nativeplugins** > TencentCloud-TUIRoomEngine 。



2. 新增扩展配置文件

3. 配置文件内容

步骤3：重新制作自定义基座

注意：

这里的证书文件，还是主 App 的 profile 文件，不是 Extension 的 profile 文件。

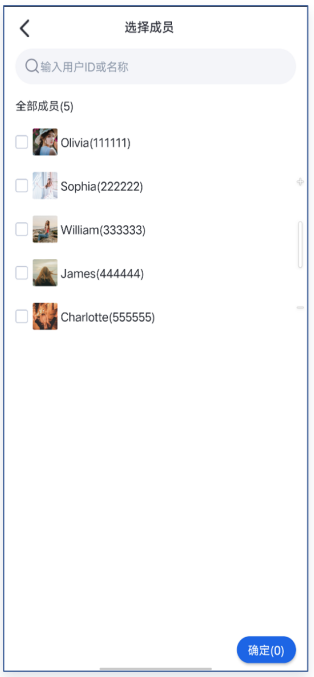
预定会议（TUIRoomKit）Android&iOS&Flutter

最近更新时间：2025-05-07 09:43:32

若 TUIRoomKit 的 UI 交互不满足您的产品需求，您有自己的交互和业务逻辑需要自定义实现预定会议相关的交互功能，可以接入 TUIRoomEngineSDK，并参考 关键代码 相关调用来实现您的需求。

功能介绍

TUIRoomKit 支持预定房间，用户可以预定一个房间并安排日程中的会议。当会议时间到来时，用户只需一键即可开启会议沟通。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

预定会议	会议列表	设置会议	邀请成员
			

功能接入

在使用 TUIRoomKit 提供的预定房间特性之前，您需要完成 TUIRoomKit 组件接入的相关配置及登录，详情可参见 快速接入 。

> **注意：**
> **预定房间特性需使用 TUIRoomKit v2.5.0 及以上版本。**

如何预定房间

使用预定房间功能时，您需要拉起 TUIRoomKit 中提供的预定房间页面，不同平台请参考：

Android

根据您的业务，只需在对应的 Activity 中调用如下代码即可预定一个房间。

java

```
Intent intent = new Intent(this, ScheduleConferenceActivity.class);
startActivity(intent);
```

kotlin

```
val intent = Intent(this, ScheduleConferenceActivity::class.java)
startActivity(intent)
```

iOS

根据您的业务，只需将下面的 **YourViewController** 替换为您自己 ViewController 即可进入预定房间页面。

Swift

```
class YourViewController: UIViewContrller {
    func jumpToScheduleViewController {
        let scheduleViewController = ScheduleConferenceViewController()
        navigationController?.pushViewController(scheduleViewController, animated: true)
  }
}
```

OC

```
#import "TUIRoomKit/TUIRoomKit-Swift.h"
@implementation YourViewController
-(void) jumpToScheduleViewController {
    ScheduleConferenceViewController *vc = [[ScheduleConferenceViewController alloc]
init];
    [[self navigationController] pushViewController:vc animated:YES];
}
@end
```

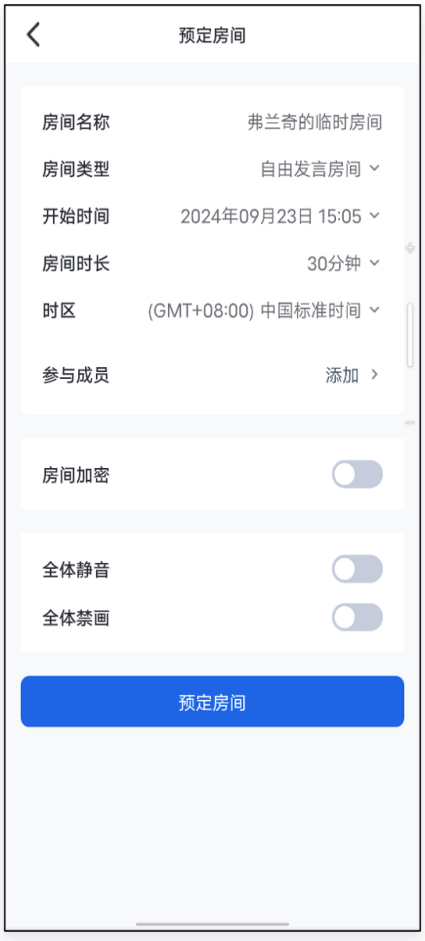
Flutter

根据您的业务，在您需要时自行导航至 `ScheduleRoomPage` 预定房间界面。

```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart

Navigator.push(
  context,
```

```
    MaterialPageRoute(
      builder: (context) => ScheduleRoomPage(),
    ),
  );
```

- 预定会议可设置的属性：其中可设置内容包含：房间名称、房间类型、开始时间、房间时长、时区、邀请成员（需导入成员列表）、房间加密、成员管理（全体静音、全体禁画）等。
- 如何邀请成员：点击 "添加" 即可邀请成员。**默认情况下用户列表需要您传入，为了方便您快速体验，我们提供了用户列表 Demo。**具体请参见 如何邀请成员 。

进入预定会议界面	设置会议信息	预定成功
		

查看已预定的房间

TUIRoomKit 中提供了房间列表的UI组件。通过将房间列表组件自行布局到自己的页面上，您便可以完成对当前用户所有已预定房间的查看与修改：

Android

1. 根据您的业务，在您的布局中添加会议列表的布局：

```
<!-- 以您的父布局是ConstraintLayout为例，添加如下代码 -->
<FrameLayout
    android:id="@+id/tuiroomkit_fl_conference_list_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

2. 在对应的 Activity 中添加如下代码，拉起会议列表：

java

```
ConferenceListFragment fragment = new ConferenceListFragment();
FragmentTransaction transaction = this.getSupportFragmentManager().beginTransaction();
transaction.add(R.id.tuiroomkit_fl_conference_list_container, fragment);
transaction.commitAllowingStateLoss();
```

kotlin

```
val fragment = ConferenceListFragment()
val transaction = supportFragmentManager.beginTransaction()
transaction.add(R.id.tuiroomkit_fl_conference_list_container, fragment)
transaction.commitAllowingStateLoss()
```

iOS

根据您的业务，只需将下面的 **YourViewController** 替换为您自己 ViewController 即可展示已经预定的会议列表。

Swift

```
class YourViewController: UIView {
        // ConferenceListView初始化时需要两个参数：
        // @param viewController，会议列表页被添加到的viewcontroller
        func showConferenceList {
            let listView = ConferenceListView(viewController: self)
            view.addSubview(listView)
        }
}
```

OC

```
@implementation YourViewController
// ConferenceListView初始化时需要两个参数：
// @param viewController，会议列表页被添加到的viewcontroller
-(void) showConferenceList {
```

```
    ConferenceListView *listView = [[ConferenceListView
alloc]initWithViewController:self];
    [[self view] addSubview:listView];
}
@end
```

Flutter

```
import 'package:flutter/material.dart';
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

// 在您自己的页面中，添加ConferenceListWidget
class YourPage extends StatelessWidget {
  const YourPage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Your Page")),
      body: Column(
        children: [
          ...YourWidgets(),              // 您其他的widget，这里仅作示例。
          const ConferenceListWidget(),  // 会议列表组件
        ],
      ),
    );
  }
}
```

会议列表提供以下功能：

- **查看会议列表：列表中包括您创建的会议以及您被邀请参加的会议。**
- **查看会议详情：您可以点击某个会议查看该会议的详情。**
- **修改会议信息：点击会议列表某个会议，若该会议未进行，且您是发起人，您可以修改该会议信息。**

点击会议列表	查看该会议详情	修改会议信息

如何邀请成员

您可在预定房间界面或修改房间界面，通过点击 `参与成员` 按钮弹出成员选择界面。默认情况下 TUIRoomKit 使用 IM 的关系链，我们也支持您设置通讯录列表，不同平台请参考：

Android

设置通讯录数据

您可以通过 setParticipants 方法设置您的通讯录用户列表。

java

```
List<ConferenceDefine.User> participants = new ArrayList<>();
ConferenceDefine.User user = new ConferenceDefine.User();
user.id = "Jack";
participants.add(user);

ConferenceSession.sharedInstance().setParticipants(participants);
```

kotlin

```
// 获取已选择的用户列表
val participants = bundle.getSerializable(CONFERENCE_PARTICIPANTS) as
ConferenceParticipants
val selectedList: ArrayList<User> = participants.selectedList
```

说明：
若 TUIRoomKit 的通讯录 UI 不满足您的业务，我们支持您将自己的通讯录 UI 与 TUIRoomKit 关联，请参考：如何设置自定义通讯录。

iOS

设置通讯录数据

您可以通过 **setParticipants** 接口设置您的通讯录用户列表。

Swift

```
let user = User(userId: "Jack", userName: "", avatarUrl: "")
ConferenceSession.sharedInstance.setParticipants([user])
```

OC

```
User *user = [[User alloc] initWithUserId:@"111" userName:@"Jack" avatarUrl:@""];
[ConferenceSession.sharedInstance setParticipants:@[user]];
```

说明：
若 TUIRoomKit 的通讯录 UI 不满足您的业务，我们支持您将自己的通讯录 UI 与 TUIRoomKit 关联，请参考：如何设置自定义通讯录。

Flutter

当您需要使用邀请成员进入房间的功能时，我们提供了以下两种方案供您添加需要邀请的成员：

方案一：使用 json 配置用户信息

您可以参考我们的 example 工程，在 example/assets 目录下，`members.json` 中存储了邀请所需要的用户信息。使用步骤如下：

1. 在您项目的 `assets` 目录下，新增 `members.json` 文件，列出所有您所需要的用户信息，文件格式与上文提到的 `members.json` 相同。
2. 在您项目的 `pubspec.yaml` 中，完成以下配置：

```
assets:
```

```
  - assets/members.json
```

完成以上配置后，您便可以在邀请成员界面中选择您在 `members.json` 中所列出的成员。

方案二：使用 IM 好友信息

当您没有配置上文所描述的 `members.json` 时，邀请好友界面会默认拉取您的IM好友信息。当您需要邀请其他成员进入房间前，您可以根据您的业务侧需求，自行添加想要邀请的成员为好友。

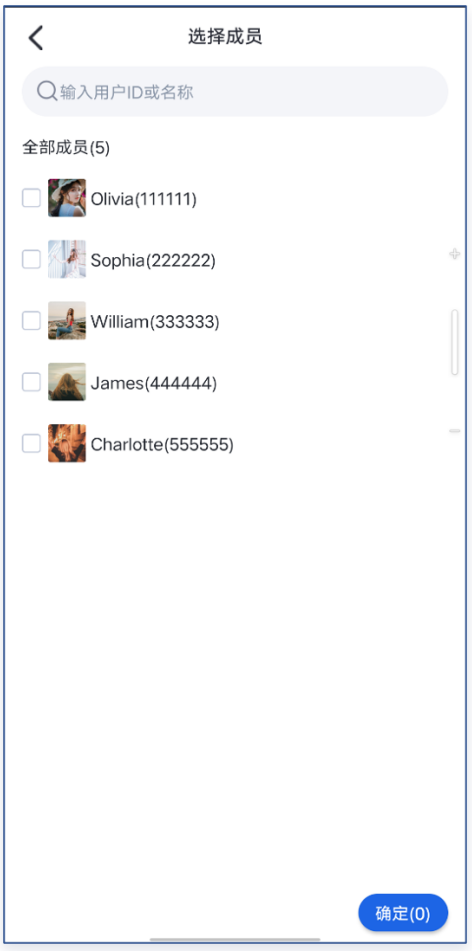
- 前提条件：登录IM SDK，登录操作与 登录弹幕聊天 操作相同。如您已完成其登录或已使用 会中聊天，可忽略此步骤。
- 添加好友的代码如下：

```
import 'package:tencent_cloud_chat_sdk/manager/v2_tim_manager.dart';

// Flutter TUIRoomKit中，已有对tencent_cloud_chat_sdk的依赖，您无需另行配置
// 将代码中的userID替换为您需要添加的用户的UserID即可完成好友的添加
V2TIMManager().getFriendshipManager().addFriend(
       userID: 'userID', addType: FriendTypeEnum.V2TIM_FRIEND_TYPE_BOTH);
```

完成好友的添加后，您便可以选择在**每次**预定时完成对所添加用户的邀请。
如您需要更多相关的好友操作，请参考：好友管理。如您需要使用Rest API批量添加好友，请参考：添加好友 RestAPI。

完成以上内容后，通过点击 `参与成员` 按钮弹出成员选择界面，您可以在这里邀请或删除参与的成员。邀请并完成房间的预定或修改后，对方的会议列表中便会**出现您所预定的会议**。

预定会议界面 点击 参与成员	用户列表中添加或删除成员



注意事项

- 预定房间成功后，即可获取房间号和预定信息。
- 若同时预定不同日期/时间的房间，分次选择时间提交即可。
- 预定房间开始时间不能早于当前时间，但没有提前天数的限制。
- 预定的房间到达结束时间后，若房间没有被解散且房间内没有用户，该房间将从预定结束时间起保留6小时，在此期间您仍可以随时进入该房间。

功能定制

如果当前的 UI 不满足您的需求，您可以通过修改源码来实现您满意的 UI 效果，不同平台请参考：

Android

您可以通过修改 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/schedule 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对预定会议的文件做了介绍。

```
SchdeuleConference
    ├── ConferenceDetails          // 会议详情界面
    ├── ConferenceList             // 会议列表界面
    ├── ModifyConference           // 修改会议界面
    ├── SelectScheduleParticipant  // 调用从外部传如的通讯录列表
    ├── TimeZone                   // 设置时区界面
```

```
    └── View                        // 预定会议界面
```

如何设置自定义通讯录

1. 传入您的通讯录 UI

您可在 预定会议 和拉起 会议列表 前，通过以下方法传入您的通讯录：

java

```
// 将 SelectParticipantActivity.class 替换为您通讯录的 activity
ConferenceSession.sharedInstance().setContactsViewProvider(SelectParticipantActivity.clas
s);
```

kotlin

```
// 将 SelectParticipantActivity::class.java 替换为自定义通讯录的 activity
ConferenceSession.sharedInstance().setContactsViewProvider(SelectParticipantActivity::cla
ss.java)
```

2. 您的通讯录向 TUIRoomKit 返回选择完毕的用户名单：

在通讯录完成用户选择后，您需要将已选用户列表返回给 TUIRoomKit。您可以通过以下方法将数据返回给 TUIRoomKit。

java

```
Intent intent = new Intent();
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants);
setResult(3, intent);
finish();
```

kotlin

```
val intent = Intent()
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants)
setResult(3, intent)
finish()
```

3. TUIRoomKit 向您的通讯录传入已选择的用户列表：

您可通过以下方法获得该会议已选择的用户名单。

java

```
// 获取已选择的用户列表
ConferenceParticipants participants = (ConferenceParticipants)
bundle.getSerializable(CONFERENCE_PARTICIPANTS);
ArrayList<User> selectedList = participants.selectedList;
```

kotlin

```kotlin
// 获取已选择的用户列表
val participants = bundle.getSerializable(CONFERENCE_PARTICIPANTS) as
ConferenceParticipants
val selectedList: ArrayList<User> = participants.selectedList
```

iOS

您可以通过修改 iOS/TUIRoomKit/Source 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对预定会议的文件做了介绍。

```
Source
├── ConferenceListView.swift                    // 预定会议列表页面
├── ScheduleConferenceViewController.swift      // 预定会议主页面
└── View
    └── ConferenceOptions
        ├── ConferenceInvitation                // 会中呼叫
        ├── MemberSelect                        // 预定会议邀请成员
        ├── ModifySchedule                      // 修改预定会议信息
        └── ScheduleConference
            └── View
                ├── ScheduleConference          // 预定会议
                ├── ScheduleDetails             // 预定会议详情
                └── Widget                      // 预定会议弹出页面
```

如何设置自定义通讯录

1. 准备好您的好友选择页 viewController，实现 `ContactViewProtocol` 协议。

```swift
// 示例代码
class SelectMemberViewController: UIViewController, ContactViewProtocol {
    weak var delegate: ContactViewSelectDelegate?
    var participants: ConferenceParticipants

    func didSelectFinished() {
    // 在完成选择的方法中通过delegate把选择的成员回调给RoomKit
        delegate?.onMemberSelected(self, invitees: selectedMembers)
    }
}
```

> **说明：**
> 会中呼叫功能同样需要使用您自定义的通讯录组件。如您还需使用会中呼叫功能，这里建议您将一个 `ConferenceParticipants` 对象置于您通讯录页面的构造函数参数中，数据来源会在第二步的代码中提到。
> `ConferenceParticipants` 类中有两个成员：
> - selectedList：已选择的成员；

- unSelectableList：不可取消选择的成员，您可以在UI上将对应的成员设置其为不可取消勾选。预定会议功能没有使用此字段。

2. TUIRoomKit 关联自定义通讯录：您可在预定会议和拉起会议列表前，通过以下方法设置您在上一步中自定义的通讯录：

```
// 示例代码
ConferenceSession.sharedInstance.setContactsViewProvider { participants in
    return SelectMemberViewController(participants: participants)
}
```

Flutter

您可以通过修改 tencent_conference_uikit/lib/pages 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对预定会议的文件做了介绍。

```
pages
  ├── conferenceList                          // 预定会议列表文件夹
  |       ├── view.dart                       // 预定会议列表页面
  |       └── widget.dart
  |             ├── conference_date_item.dart  // 会议列表中单个时间widget
  |             ├── conference_item.dart       // 会议列表中单个会议widget
  |             └── no_schedule_widget.dart    // 会议列表中暂无会议时显示widget
  ├── schedule_conference                     // 预定会议页面文件夹
  |       ├── view.dart                       // 预定会议页面
  |       └── widgets
  |             ├── attendee_selector         // 成员选择页面文件夹
  |             |       ├── view.dart         // 成员选择页面
  |             |       └── widgets
  |             |             └── selected_attendees.dart  // 弹出的已选择成员页面
  |             ├── duration_selector.dart    // 预定会议时长选择widget
  |             ├── room_control.dart         // 预定会议全体静音/禁画widget
  |             ├── room_info.dart            // 预定会议总体房间信息widget
  |             ├── room_type.dart            // 房间类型选择widget
  |             ├── start_time_selector.dart  // 预定会议开始时间选择widget
  |             └── time_zone_selector.dart   // 预定会议时区选择页
  └── schedule_details                        // 会议详情文件夹
  |       ├── view.dart                       // 会议详情页面
  |       └── widgets
  |           ├── details_button_item.dart    // 会议详情页面内单个按钮widget
  └──         └── room_info.dart              // 会议信息widget
```

说明：
如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

关键代码

若您想自定义实现预定会议功能，请参考 TUIConferenceListManager： Android、iOS&Mac、Flutter。

更多特性

- 会前提醒
- 房间密码

版权所有：腾讯云计算（北京）有限责任公司

Web&Electron

最近更新时间：2025-01-03 14:56:32

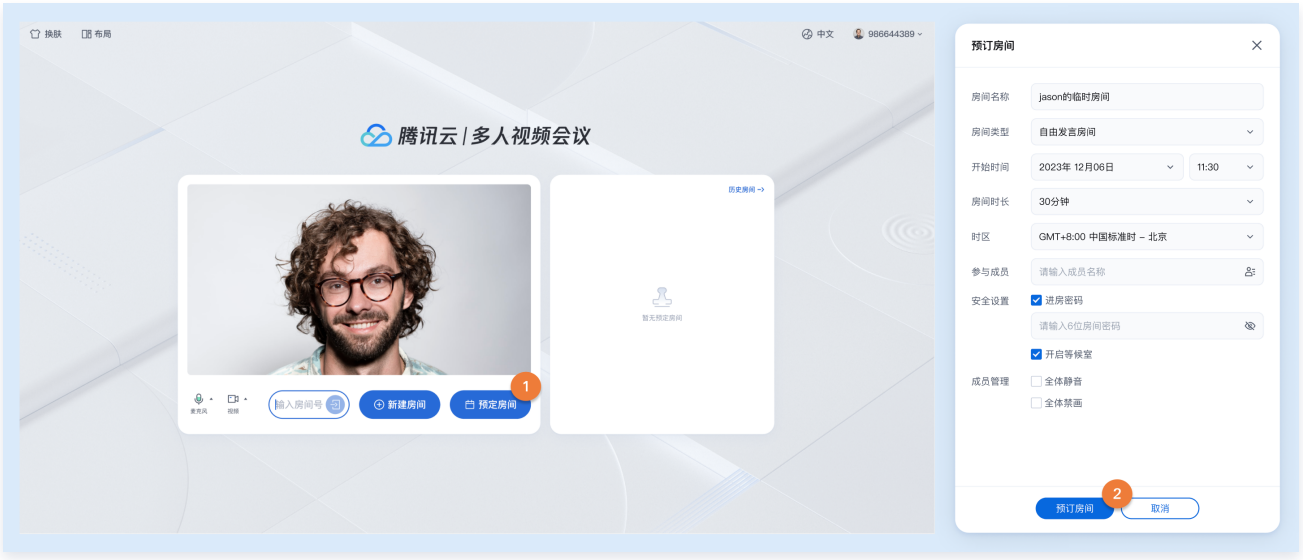
功能介绍

TUIRoomKit 推出了预定房间新特性，用户可预定一个房间，安排一场日程中的会议。

> **说明：**
> TUIRoomKit 自 v2.5.0 版本支持预定房间、查看房间列表、修改房间信息等能力，集成最新版本 TUIRoomKit 可体验预定房间流程。

如何预定房间

1. 在 TUIRoomKit 会前预览页面单击**预定房间** > 填写房间预定信息及设置相关权限 > 完成设置后单击**预定房间**即可。



2. 预定房间成功后将会同步预定结果呈现在会前预览页面右侧**预定房间列表**中，并支持**查看详情**、**修改房间**、**取消房间**、**复制邀请信息**等操作。另外，您还可以单击**加入**进入已预定的房间。

准备条件

在使用腾讯云提供的预定房间特性前，您需要前往控制台，为应用开通多人会议服务。具体步骤请参见 开通服务 。接着，您需要引入 TUIRoomKit 组件，具体可参见 快速跑通 。

预定房间示例跑通

说明：

您需要引入 PreConferenceView（会前预览组件）和 ConferenceMainView（TUIRoomkit UI 组件主体），示例中采用 v-if 和 v-else 指令来控制两个组件的展示与隐藏，您也可以采用路由跳转的方式进行组件之间的切换。

- 在 PreConferenceView 组件中，通过设置 enable-scheduled-conference 属性的值，控制预定房间特性的展示或隐藏。另外，组件监听了 on-enter-room 事件，当用户点击进入房间时，触发 handleEnterRoom 方法即可调用 join 接口。
- 在 ConferenceMainView 组件中，监听了 on-destroy-room 事件，当房间被销毁时，触发 onDestroyRoom 方法。

Web

```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // 设置是否开启预定房间特性展示
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
</template>
```

```
<script setup lang="ts">
import { ref } from 'vue';
 // 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { PreConferenceView, conference, ConferenceMainView } from '@tencentcloud/roomkit-web-
vue3';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，https://cloud.tencent.com/document/product/647/104842
    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: '',
  });
}
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

Electron

```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // 设置是否开启预定房间特性展示
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
</template>

<script setup lang="ts">
import { ref } from 'vue';
 // 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
```

```
import { PreConferenceView, conference, ConferenceMainView } from '@tencentcloud/roomkit-electron-vue3';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，https://cloud.tencent.com/document/product/647/104842
    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: '',
  });
}
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

预定房间控制

预定房间

会前预览页单击**预定房间**后，弹出预定房间设置框，用户可根据自己的需求设置房间信息，其中可设置内容包含：**房间名称、房间类型、开始时间、房间时长、时区、参与成员、安全设置（进房密码）、成员管理（全体静音、全体禁画）**等。

自定义通讯录

方式一：使用 IM 好友关系

TUIRoomKit 的邀请成员列表的成员信息默认来自于 IM **好友关系** 。您可以通过**添加 IM 好友关系**的方式来完成用户数据的替换。此时您需要通过 IM REST API 采用添加好友关系的方式得到 IM 好友关系链的数据。若您进行参会成员通讯录的数据导入或添加用户，可参见 添加好友 。若您需要移除参会成员，即删除 IM 关系链中的好友关系即可，具体可参见 删除好友 。

方式二：调用 setParticipants 接口设置自己的通信录列表

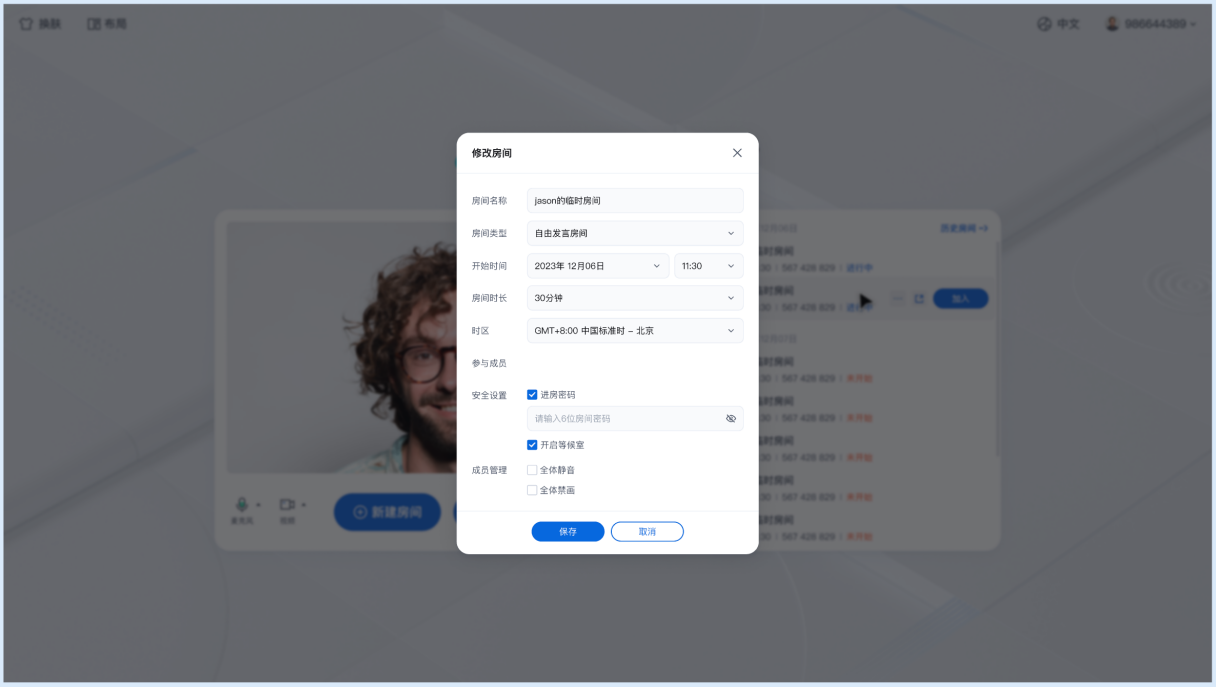
```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setParticipants([{ userId: 'xxx', userName: 'xxx',  avatarUrl: 'xxx' }]);
```

查看详情

用户可单击**查看详情**，查看对应预定房间的详细信息。

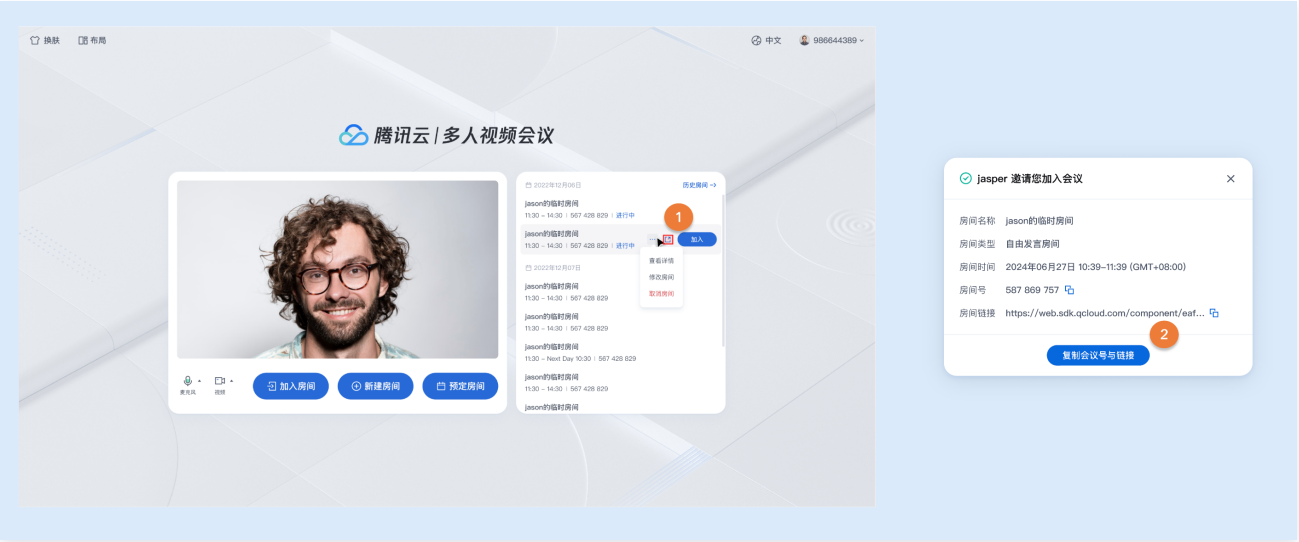
修改房间信息

房间拥有者可以对已预定的房间信息进行修改，修改完毕单击**保存**即可将当前预定房间的信息调整为修改后的房间信息。



邀请进入房间

用户可通过单击邀请 弹出会议信息邀请框，通过单击**复制会议号与链接**，将预定房间信息复制到粘贴板分享给其他用户。



注意事项

- 预定房间开始时间不能早于当前时间，但没有提前天数的限制。
- 若同时预定不同日期/时间的房间，分次选择时间提交即可。
- 房间预定成功后，房间号将从预定开始时间起，若房间没有用户，那么房间保留6小时，在此期间您都可以随时回到该房间。
- 预定房间成功后，即可获取房间号和预定信息。

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

版权所有：腾讯云计算（北京）有限责任公司

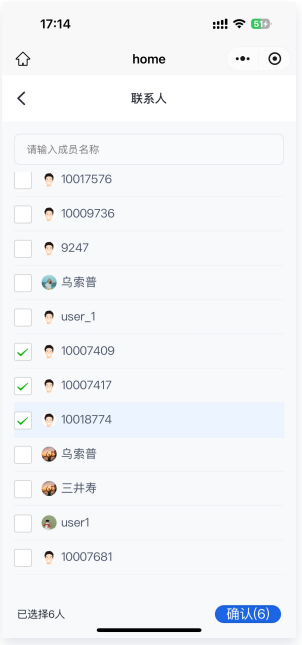
Web（H5）&小程序

最近更新时间：2025-01-03 14:56:32

功能介绍

TUIRoomKit 推出了预定房间新特性，用户可预定一个房间，安排一场日程中的会议。

> **说明：**
> **TUIRoomKit 自 v2.6.0 版本 Web（H5）&小程序支持预定房间、查看房间列表、修改房间信息等能力，集成最新版本 TUIRoomKit 可体验预定房间流程。**

会议列表	预定会议参数	邀请会议成员	会议详情
			

准备条件

在使用腾讯云提供的预定房间特性前，您需要前往控制台，为应用开通多人会议服务。具体步骤请参见 开通服务 。接着，您需要引入 TUIRoomKit 组件，具体可参见 快速跑通 。

预定房间示例跑通

> **说明：**
> **您需要引入 PreConferenceView（会前预览组件）和 ConferenceMainView（TUIRoomkit UI 组件主体），示例中采用 v-show 和 v-else 指令来控制两个组件的展示与隐藏，您也可以采用路由跳转的方式进行组件之间的切换。（注意: 小程序进房需要确保 ConferenceMainView 存在，因此此处不能使用 v-if）**
>
> - **在 PreConferenceView 组件中，通过设置 enable-scheduled-conference 属性的值，控制预定房间特性的展示或隐藏。另外，组件监听了 on-enter-room 事件，当用户点击进入房间时，触发 handleEnterRoom 方法即可调用 join 接口。**
> - **在 ConferenceMainView 组件中，监听了 on-destroy-room 事件，当房间被销毁时，触发 onDestroyRoom 方法。**

Web-H5

```
<template>
  <PreConferenceView
    v-show="isShowPreConferenceView"
    :enable-scheduled-conference="true"
    @on-enter-room="handleEnterRoom"
  />
  <ConferenceMainView
    v-show="!isShowPreConferenceView"
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  />
</template>

<script setup lang="ts">
import { ref } from 'vue';
import { PreConferenceView, conference, ConferenceMainView } from '@tencentcloud/roomkit-web-
vue3';
const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
      // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
      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: '',
    });
};
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

Uniapp-WX

```
<template>
  <PreConferenceView
```

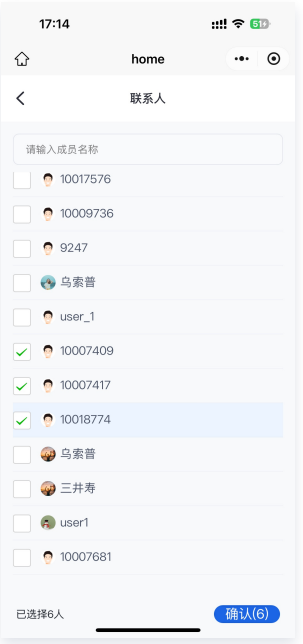
```
    v-show="isShowPreConferenceView"
    :enable-scheduled-conference="true"
    @on-enter-room="handleEnterRoom"
  />
  <ConferenceMainView
    v-show="!isShowPreConferenceView"
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  />
</template>

<script setup lang="ts">
import { ref } from 'vue';
import PreConferenceView from '../TUIRoom/preConference.vue';
import ConferenceMainView from '../TUIRoom/conference.vue';
import { conference } from '../TUIRoom/index';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，https://cloud.tencent.com/document/product/647/104842
    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: '',
  });
};
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

预定房间功能介绍

预定房间

- 预定会议可设置的属性：其中可设置内容包含：房间名称、房间类型、开始时间、房间时长、时区、邀请成员（需导入成员列表）、房间加密、成员管理（全体静音、全体禁画）等。

- 如何邀请成员：点击 "添加" 即可邀请成员。

会前预览页	填写预定会议参数	邀请会议成员	预定成功
			

自定义通讯录

方式一：使用 IM 好友关系

TUIRoomKit 的邀请成员列表的成员信息默认来自于 IM 好友关系 。您可以通过添加 IM 好友关系的方式来完成用户数据的替换。此时您需要通过 IM REST API 采用添加好友关系的方式得到 IM 好友关系链的数据。若您进行参会成员通讯录的数据导入或添加用户，可参见 添加好友 。若您需要移除参会成员，即删除 IM 关系链中的好友关系即可，具体可参见 删除好友 。

方式二：调用 setParticipants 接口设置自己的通信录列表

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setParticipants([{ userId: 'xxx', userName: 'xxx',  avatarUrl: 'xxx' }]);
```

查看会议详情与修改会议信息

会议列表提供以下功能：

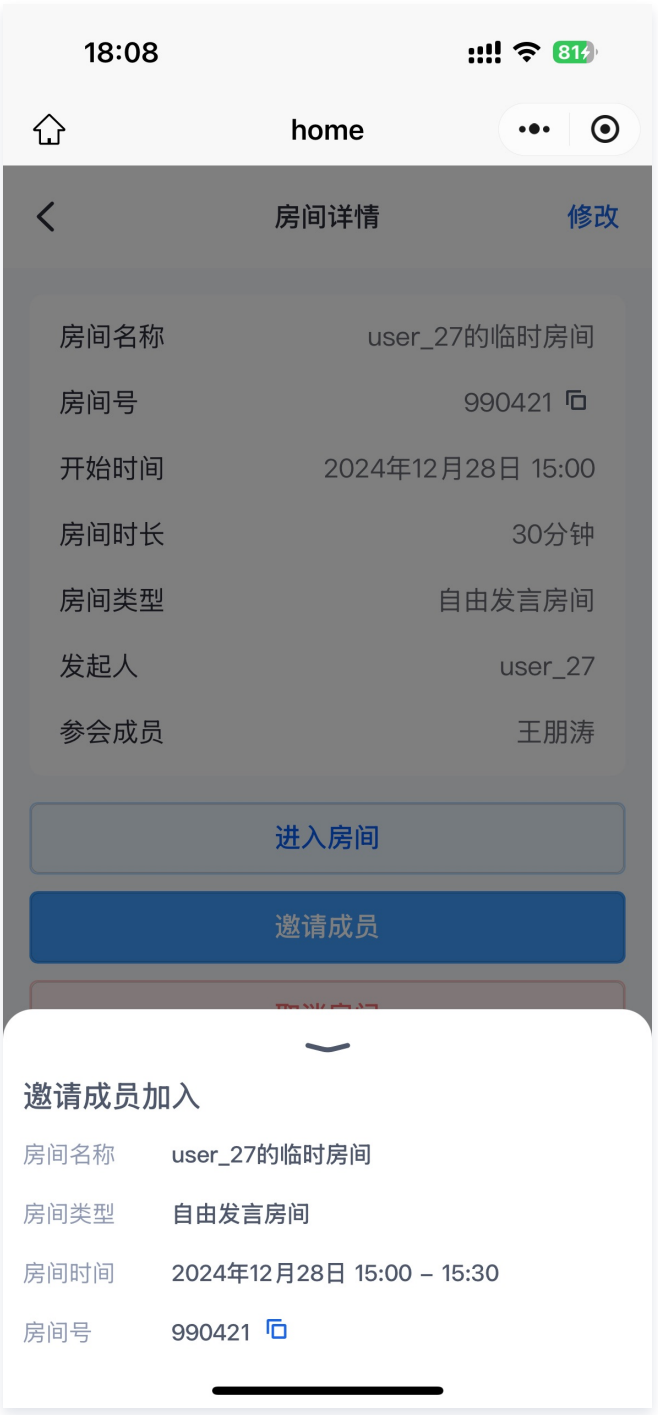
- 查看会议列表：列表中包括您创建的会议以及您被邀请参加的会议。
- 查看会议详情：您可以点击某个会议查看该会议的详情。
- 修改会议信息：点击会议列表某个会议，若该会议未进行，且您是发起人，您可以修改该会议信息。

会议列表	会议详情	修改会议信息

邀请进入房间

用户可通过单击**邀请成员**弹出会议信息邀请框，通过单击**复制会议号**，将预定房间信息复制到粘贴板分享给其他用户。

会议详情	邀请成员



注意事项

- 预定房间开始时间不能早于当前时间，但没有提前预定的天数限制。
- 若同时预定不同日期/时间的房间，请分次选择时间后提交。
- 房间预定成功后，房间号将从预定开始时间起，若房间没有用户，那么房间保留6小时，在此期间您都可以随时回到该房间。
- 预定房间成功后，即可获取房间号和预定信息。

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

版权所有：腾讯云计算（北京）有限责任公司

会中呼叫（TUIRoomKit）Android&iOS

最近更新时间：2025-01-22 09:27:42

功能介绍

TUIRoomKit 支持会中呼叫功能。用户在会议进行过程中，可以随时呼叫其他用户加入当前会议，无需提前预定或安排。通过会中呼叫功能，用户可以灵活地邀请或提醒相关人员参与会议，提升会议的互动性和效率。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

呼叫端	被呼叫端

使用说明

呼叫用户

当您在会议中时，您可以通过如下两种方式对未进房的用户进行呼叫：

方式一：呼叫成员列表中的未进入用户

在房间的成员列表中，您会看到一个名为**未进入**的标题栏。点击**未进入**，会显示所有当前未进入会议的成员，您可以对这些尚未进入会议的成员进行呼叫。
未进入的列表中包含两类用户：

- 当前会议在预定时邀请且未进入的成员
- 已对其进行呼叫但仍未入会的成员

方式二：呼叫通讯录中的用户

默认情况下通讯录中的好友列表使用 IM 的**关系链**，您也可以通过以下接口设置您自己的通讯录列表，不同平台请参考：

Android

设置通讯录数据

您可以通过 setParticipants 方法设置您的通讯录用户列表。

java

```
List<ConferenceDefine.User> participants = new ArrayList<>();
ConferenceDefine.User user = new ConferenceDefine.User();
user.id = "Jack";
participants.add(user);

ConferenceSession.sharedInstance().setParticipants(participants);
```

kotlin

```
// 获取已选择的用户列表
val participants = bundle.getSerializable(CONFERENCE_PARTICIPANTS) as
ConferenceParticipants
val selectedList: ArrayList<User> = participants.selectedList
```

iOS

设置通讯录数据

您可以通过 setParticipants 接口设置您的通讯录用户列表。

Swift

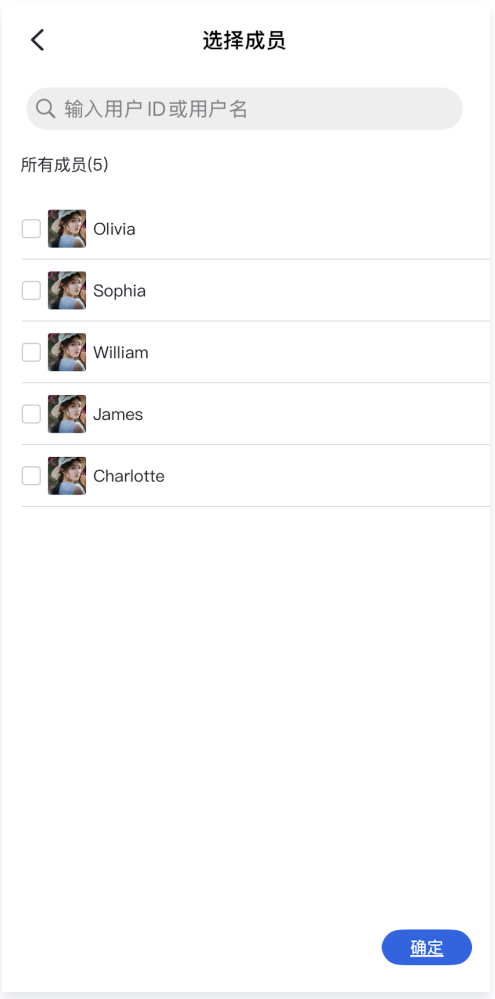
```
let user = User(userId: "111", userName: "Jack", avatarUrl: "")
ConferenceSession.sharedInstance.setParticipants([user])
```

OC

```
User *user = [[User alloc] initWithUserId:@"111" userName:@"Jack" avatarUrl:@""];
[ConferenceSession.sharedInstance setParticipants:@[user]];
```

通过点击底部栏中的邀请 > 添加成员，您可以唤起通讯录界面，并对其中您所选定的成员进行呼叫。

会中呼叫入口	通讯录页面



说明：
若 TUIRoomKit 的通讯录 UI 不满足您的业务，我们支持您将自己的通讯录 UI 与 TUIRoomKit 关联，请参考：如何设置自定义通讯录 。

收到呼叫

当您在应用内收到呼叫时，会弹出如下图所示的页面。您可以拖动滑块选择**立即加入**，或点击**暂不进入**以拒绝此呼叫。

说明：
当用户已在会议中或正在被呼叫时，该用户会自动拒绝所有呼叫。

功能定制

铃声定制

iOS

自定义被呼叫铃声

如果您想要自定义被呼叫的铃声，可以先将音乐资源下载到本地，然后调用以下接口传入资源的路径。

Swift

```
ConferenceSession.sharedInstance.setCallingBell(filePath: "yourCallingBellPath")
```

OC

```
[ConferenceSession.sharedInstance setCallingBellWithFilePath:@"yourCallingBellPath"];
```

或者直接替换 TUIRoomKit/iOS/TUIRoomKit/Resources/AudioFile/phone_ringing.mp3 中的资源文件。

设置被呼叫时静音模式

如果您的被叫端在收到会中呼叫后不想要响铃，可以调用以下接口设置静音模式。

Swift

```
ConferenceSession.sharedInstance.enableMuteMode(enable: true)
```

OC

```
[ConferenceSession.sharedInstance enableMuteModeWithEnable:YES];
```

设置被呼叫时震动模式

如果您的被叫端在收到会中呼叫后不想要震动，可以调用以下接口关闭震动模式。

Swift

```
ConferenceSession.sharedInstance.enableVibrationMode(enable: false)
```

OC

```
[ConferenceSession.sharedInstance enableVibrationModeWithEnable:NO];
```

Android

Android 暂时不支持会中呼叫铃声。

页面定制

自定义通讯录

如果当前的通讯录页面不满足您的需求，我们支持您将自己的通讯录 UI 与 TUIRoomKit 关联。

Android

1. **TUIRoomKit 关联自定义通讯录：**您需要在呼叫通讯录中的用户之前，通过以下方法设置自定义通讯录：

java

```
// 将 SelectParticipantActivity.class 替换为自定义通讯录的activity
```

```
ConferenceSession.sharedInstance().setContactsViewProvider(SelectParticipantActivity.clas
s);
```

kotlin

```
// 将 SelectParticipantActivity::class.java 替换为自定义通讯录的 activity
ConferenceSession.sharedInstance().setContactsViewProvider(SelectParticipantActivity::cla
ss.java)
```

2. 您的通讯录向 TUIRoomKit 返回选择完毕的用户名单：

在通讯录完成用户选择后，您需要将已选用户列表返回给 TUIRoomKit。您可以通过以下方法将数据返回给 TUIRoomKit。

java

```
Intent intent = new Intent();
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants);
setResult(3, intent);
finish();
```

kotlin

```
val intent = Intent()
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants)
setResult(3, intent)
finish()
```

3. TUIRoomKit 向您的通讯录传入已选择的用户列表：

您可通过以下方法获得已选择的呼叫用户名单。

java

```
// 获取已选择的用户列表
ConferenceParticipants participants = (ConferenceParticipants)
bundle.getSerializable(CONFERENCE_PARTICIPANTS);
ArrayList<User> selectedList = participants.selectedList;
```

kotlin

```
// 获取已选择的用户列表
val participants = bundle.getSerializable(CONFERENCE_PARTICIPANTS) as
ConferenceParticipants
val selectedList: ArrayList<User> = participants.selectedList
```

iOS

1. 准备好您的好友选择页 viewController，实现 `ContactViewProtocol` 协议。

```
// 示例代码
class SelectMemberViewController: UIViewController, ContactViewProtocol {
    weak var delegate: ContactViewSelectDelegate?
    var selectedList: [User]

    func didSelectFinished() {
      // 在完成选择的方法中通过delegate把选择的成员回调给RoomKit
        delegate?.onMemberSelected(self, invitees: selectedMembers)
    }
}
```

> **说明：**
> 这里建议您将一个 `ConferenceParticipants` 对象置于您通讯录页面的构造函数参数中，数据来源在第二步的代码中提到。
> `ConferenceParticipants` 类中有两个成员：
> - selectedList：已选择的成员；
> - unSelectableList：不可选择的成员，您可以在UI上将对应的成员设置其为不可选择。在会中呼叫时，不可选择的成员为已在会中的成员。

2. 在呼叫通讯录中的用户之前，您需要通过以下方法将您自定义的通讯录传入TUIRoomKit：

```
ConferenceSession.sharedInstance.setContactsViewProvider { participants in
            return SelectMemberViewController(participants: participants)
}
```

自定义被呼叫页面视图

如您需要自定义被呼叫页面的视图，请参考以下路径更改源代码：

Android

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/component/

component
    └──InvitationReceivedView.java
```

iOS

```
// 文件位置：iOS/TUIRoomKit/Source/View/ConferenceOptions/ConferenceInvitation

ConferenceInvitation
    └── ConferenceInvitationViewController.swift  // 被呼叫页面视图
```

自定义成员列表中呼叫视图

如您需要自定义成员列表中呼叫成员的视图，请参考以下路径更改源代码：

Android

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/UserCo
ntrolPanel/

UserControlPanel
    └── CallUserView.java // 成员列表呼叫按钮
```

iOS

```
// 文件位置：iOS/TUIRoomKit/Source/Page/Widget/UserControlPanel

UserControlPanel            // 成员列表相关的视图目录
    └── UserListCell.swift  // 成员列表中单个成员视图，包含用户呼叫状态视图
```

关键代码

呼叫用户

Android

```
// 文件位置：
TUIRoomKit/blob/main/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/mo
del/controller/InvitationController.java

public void inviteUsers(List<UserState.UserInfo> userInfoList,
TUIConferenceInvitationManager.InviteUsersCallback callback) {
    Log.d(TAG, "inviteUsers");
    if (userInfoList.isEmpty()) {
        return;
    }
```

```
RoomToast.toastShortMessageCenter(TUILogin.getAppContext().getString(R.string.tuiroomkit_
invitation_has_been_sent));
    mConferenceInvitationManager.inviteUsers(mRoomState.roomId.get(),
getUserIdListFromUserList(userInfoList), INVITE_TIME_OUT_SECONDS, "", new
TUIConferenceInvitationManager.InviteUsersCallback() {
        @Override
        public void onSuccess(Map<String, TUIConferenceInvitationManager.InvitationCode>
invitationResultMap) {
            Log.d(TAG, "inviteUsers success");
            if (callback != null) {
                callback.onSuccess(invitationResultMap);
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String message) {
            Log.d(TAG, "inviteUsers error=" + error + " message=" + message);
            if (callback != null) {
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置：TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService.swift

func inviteUsers(roomId: String, userIdList: [String]) -> AnyPublisher<InviteUsersResult,
RoomError> {
    return Future<InviteUsersResult, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.inviteUsers(roomId, userIdList: userIdList, timeout:
timeout, extensionInfo: "") {dic in
            promise(.success((dic)))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message: message)))
        }
    }
    .eraseToAnyPublisher()
}
```

接受呼叫

Android

```
// 文件位置:
TUIRoomKit/blob/main/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/mo
del/controller/InvitationController.java

public void accept(String roomId, TUIRoomDefine.ActionCallback callback) {
    Log.d(TAG, "accept");
    mConferenceInvitationManager.accept(roomId, new TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.d(TAG, "accept success");
            if (callback != null) {
                callback.onSuccess();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String message) {
            Log.d(TAG, "accept error=" + error + " message=" + message);
            if (callback != null) {
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置: TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService.swift

func accept(roomId: String) -> AnyPublisher<String, RoomError> {
    return Future<String, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.accept(roomId) {
            promise(.success(roomId))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message: message)))
        }
    }
    .eraseToAnyPublisher()
}
```

拒绝呼叫

Android

```
// 文件位置:
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/model/contro
ller/InvitationController.java

public void reject(String roomId, TUIConferenceInvitationManager.RejectedReason reason,
TUIRoomDefine.ActionCallback callback) {
    Log.d(TAG, "reject roomId= " + roomId + " reason=" + reason);
    mConferenceInvitationManager.reject(roomId, reason, new
TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.d(TAG, "reject success");
            if (callback != null) {
                callback.onSuccess();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String message) {
            Log.d(TAG, "reject error=" + error + " message=" + message);
            if (callback != null) {
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置: TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService.swift

func reject(roomId: String, reason: TUIInvitationRejectedReason) -> AnyPublisher<String,
RoomError> {
    return Future<String, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.reject(roomId, reason: reason) {
            promise(.success(roomId))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message: message)))
        }
    }
    .eraseToAnyPublisher()
}
```

获取房间内呼叫列表

Android

```
// 文件位置：
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/model/contro
ller/InvitationController.java

private void getInvitationList() {
    Log.d(TAG, "getInvitationList");
    mConferenceInvitationManager.getInvitationList(mRoomState.roomId.get(),
getAttendeeListCursor, SINGLE_FETCH_COUNT, new
TUIConferenceInvitationManager.GetInvitationListCallback() {
        @Override
        public void onSuccess(TUIConferenceInvitationManager.InvitationListResult
invitationListResult) {
            Log.d(TAG, "getInvitationList");
            for (TUIConferenceInvitationManager.Invitation invitation :
invitationListResult.invitationList) {
                InvitationState.Invitation invitationState = new
InvitationState.Invitation();
                invitationState.invitee = new UserState.UserInfo(invitation.invitee);
                invitationState.inviter = new UserState.UserInfo(invitation.inviter);
                invitationState.invitationStatus = invitation.status;
                mInvitationState.invitationList.add(invitationState);
            }
            getInvitationListCursor = invitationListResult.cursor;
            if (!"".equals(getInvitationListCursor)) {
                getInvitationList();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String message) {
            Log.d(TAG, "getInvitationList onError error=" + error + " message=" +
message);
        }
    });
}
```

iOS

```
// 文件位置：TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService.swift

func getInvitationList(roomId: String, cursor: String, count: Int = 20) ->
AnyPublisher<InvitationfetchResult, RoomError> {
    return Future<InvitationfetchResult, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.getInvitationList(roomId, cursor: cursor, count: count)
{invitations, cursor in
```

```
            promise(.success((invitations, cursor)))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message: message)))
        }
    }
    .eraseToAnyPublisher()
}
```

用户收到呼叫监听

Android

```
// 文件位置:
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/model/Confer
enceServiceInitializer.java

private void initConferenceInvitationObserver() {
    TUIConferenceInvitationManager invitationManager = (TUIConferenceInvitationManager)
TUIRoomEngine.sharedInstance().getExtension(TUICommonDefine.ExtensionType.CONFERENCE_INVI
TATION_MANAGER);
    invitationManager.addObserver(new TUIConferenceInvitationManager.Observer() {
        @Override
        public void onReceiveInvitation(TUIRoomDefine.RoomInfo roomInfo,
TUIConferenceInvitationManager.Invitation invitation, String extensionInfo) {
            if
(ConferenceController.sharedInstance().getViewState().isInvitationPending.get()) {

ConferenceController.sharedInstance().getInvitationController().reject(roomInfo.roomId,
REJECT_TO_ENTER, null);
                return;
            }
            if (ConferenceController.sharedInstance().getRoomController().isInRoom()) {

ConferenceController.sharedInstance().getInvitationController().reject(roomInfo.roomId,
IN_OTHER_CONFERENCE, null);
                return;
            }

            Bundle bundle = new Bundle();
            bundle.putString("roomId", roomInfo.roomId);
            bundle.putString("conferenceName", roomInfo.name);
            bundle.putString("ownerName", roomInfo.ownerName);
            bundle.putString("inviterName", invitation.inviter.userName);
            bundle.putString("inviterAvatarUrl", roomInfo.ownerAvatarUrl);
            bundle.putInt("memberCount", roomInfo.memberCount);
            TUICore.startActivity("InvitationReceivedActivity", bundle);
        }
    });
```

```
}
```

iOS

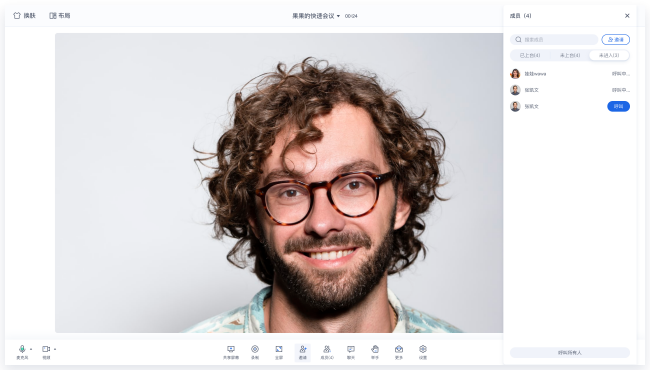
```
// 文件位置：TUIRoomKit/iOS/TUIRoomKit/Source/Service/InvitationObserverService.swift

func onReceiveInvitation(roomInfo: TUIRoomInfo, invitation: TUIInvitation, extensionInfo:
String) {
    let store = Container.shared.conferenceStore()
    store.dispatch(action: ConferenceInvitationActions.onReceiveInvitation(payload:
(roomInfo, invitation)))
}
```

Web&Electron

最近更新时间：2024-11-12 16:06:52

功能介绍

TUIRoomKit 支持会中呼叫功能。用户在会议进行过程中，可以随时呼叫其他用户加入当前会议，无需提前预定或安排。通过会中呼叫功能，用户可以灵活地邀请或提醒相关人员参与会议，提升会议的互动性和效率。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

呼叫端	被呼叫端
	

呼叫用户

当您在会议中时，您可以通过如下两种方式对未进房的用户进行呼叫：

方式一：呼叫成员列表中的未进入用户

在房间的成员列表中，您会看到一个名为**未进入**的标题栏。点击**未进入**，会显示所有当前未进入会议的成员，您可以对这些尚未进入会议的成员进行呼叫。

未进入的列表中包含两类用户：

- 当前会议在预定时邀请且未进入的成员。
- 已对其进行呼叫但仍未入会的成员。

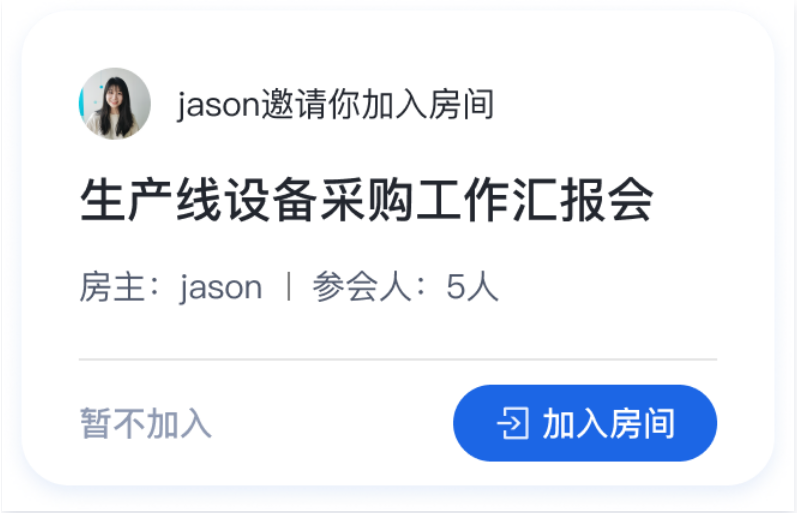
方式二：呼叫通讯录中的用户

通过点击底部栏中的**邀请 > 添加成员**，您可以唤起您自己的通讯录界面，并对其中您所选定的成员进行呼叫。

> **注意：**
> TUIRoomKit 的邀请成员列表的成员信息来自于 IM **好友关系**，需要使用 IM **完成好友关系的添加**。您可以通过**添加 IM 好友关系**的方式来完成用户数据的替换。此时您需要通过 IM REST API 采用添加好友关系的方式得到 IM 好友关系链的数据。若您进行参会成员通讯录的数据导入或添加用户，可参见 添加好友 。若您需要移除参会成员，即删除 IM 关系链中的好友关系即可，具体可参见 删除好友 。

收到呼叫

当您在应用内收到呼叫时，会弹出如下图所示的通知，您可以选择**加入房间**，或**暂不进入**。



说明：
当用户**已在会议中**或**正在被呼叫**时，该用户会自动拒绝所有呼叫。

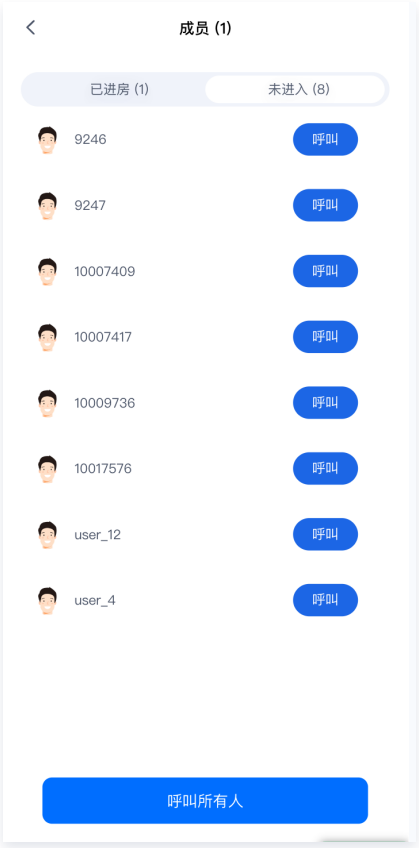
版权所有：腾讯云计算（北京）有限责任公司

小程序&H5

最近更新时间：2024-11-12 16:47:12

功能介绍

TUIRoomKit 支持会中呼叫功能。用户在会议进行过程中，可以随时呼叫其他用户加入当前会议，无需提前预定或安排。通过会中呼叫功能，用户可以灵活地邀请或提醒相关人员参与会议，提升会议的互动性和效率。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。

呼叫端	被呼叫端
	

呼叫用户

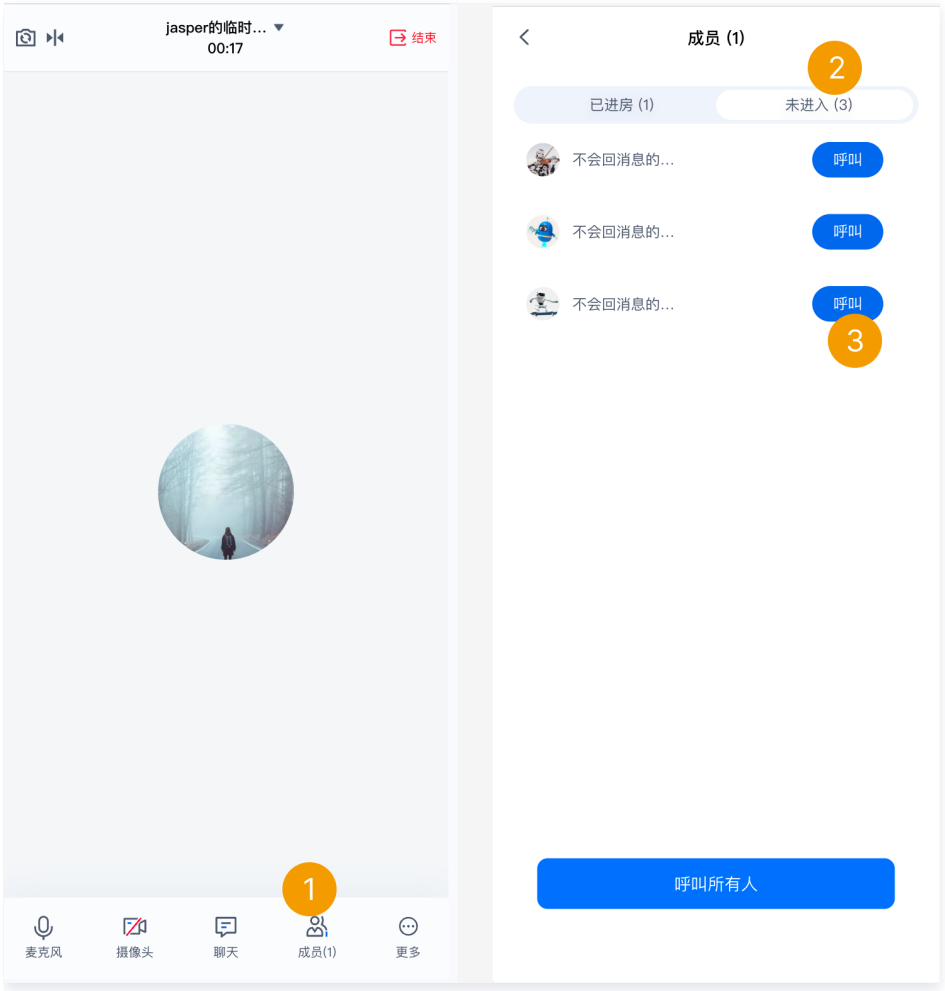
当您在会议中时，您可以通过如下两种方式对未进房的用户进行呼叫：

方式一：呼叫成员列表中的未进入用户

在房间的成员列表中，您会看到一个名为**未进入**的标题栏。点击**未进入**，会显示所有当前未进入会议的成员，您可以对这些尚未进入会议的成员进行呼叫。

未进入的列表中包含两类用户：

- 当前会议在预定时邀请且未进入的成员。
- 已对其进行呼叫但仍未入会的成员。



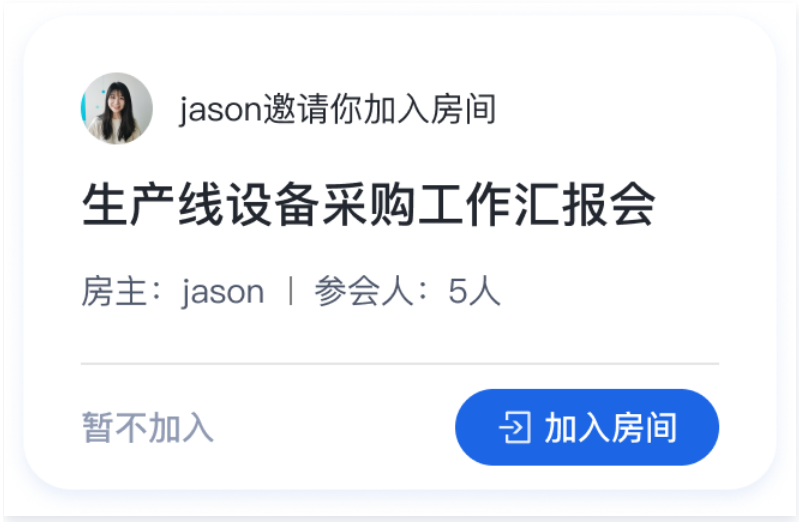
方式二：呼叫通讯录中的用户

通过点击底部栏中的**更多** > **邀请** > **添加成员**，您可以唤起您自己的通讯录界面，并对其中您所选定的成员进行呼叫。

> **注意：**
> TUIRoomKit 的邀请成员列表的成员信息来自于 IM **好友关系**，需要使用 IM **完成好友关系的添加**。您可以通过**添加 IM 好友关系**的方式来完成用户数据的替换。此时您需要通过 IM REST API 采用添加好友关系的方式得到 IM 好友关系链的数据。若您进行参会成员通讯录的数据导入或添加用户，可参见 添加好友 。若您需要移除参会成员，即删除 IM 关系链中的好友关系即可，具体可参见 删除好友 。

收到呼叫

当您在应用内收到呼叫时，会弹出如下图所示的通知，您可以选择**加入房间**，或**暂不进入**。



说明：
当用户**已在会议中**或**正在被呼叫**时，该用户会自动拒绝所有呼叫。

会议控制 (TUIRoomKit) Android&iOS&Flutter

最近更新时间：2024-11-12 16:06:52

本文档将详细介绍 `TUIRoomKit` 会议控制方面的内容，帮助您更好地掌握 `TUIRoomKit` 在进会前及会议过程中的相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。**若 TUIRoomKit 的 UI 交互不满足您的产品需求，您有自己的交互和业务逻辑需要自定义实现会控相关的交互功能，可以接入 TUIRoomEngineSDK，并参考 关键代码 相关调用来实现您的需求。**

功能介绍

`Android&iOS&Flutter` 端用户创建并进入房间后，房主或管理员角色通过点击底部工具栏**成员**按钮，在底部弹出的成员列表中既可以选中任一普通成员进行**请求开始视频/语音、设为管理员/房主、禁言、踢出房间**等会控操作，也可以对房间内所有成员进行**全体静音、仅房主/管理员可屏幕共享**等会控操作。

进入房间	成员管理	全体静音/禁画

使用说明

会前控制	
创建房间	您可以配置房间：开启麦克风/摄像头/扬声器、全体静音、全体禁画、房间密码、房间名称、创建的房间类型（自由发言/上台发言）

进入房间	您可以配置房间：开启麦克风/摄像头/扬声器。
会中控制	
您是房主或管理员	您可以控制房间：全体静音/禁画、邀请成员上台或踢下台（**上台发言房间**）、设置管理员和转交房主、将成员踢出房间、开启或关闭成员的摄像头/麦克风。

会前控制

在创建并加入会议时，您需要预先设定会议的相关参数，通过 `TUIRoomKit` 会前控制的相关功能，确保会议顺利进行。

创建房间	加入房间

- 您可以通过以下方式创建房间，并设置进房参数，完成会前控制：

Android

```
ConferenceDefine.StartConferenceParams params = new
ConferenceDefine.StartConferenceParams("该房间的roomId");
params.isSeatEnabled = false;     // true:上台发言房间,false:自由发言房间。
params.password = "该房间的密码";  // 房间密码。
params.name = "该房间的名字";      // 房间名称。
params.isOpenCamera = true;       // 开启摄像头。
params.isOpenMicrophone = true;   // 开启麦克风。
params.isOpenSpeaker = true;      // 开启扬声器。
```

```
params.isCameraDisableForAllUser = false;      // 开启全体禁画。
params.isMicrophoneDisableForAllUser = false; // 开启全体静音。
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent);
```

iOS

```
import TUIRoomKit

func quickStartConference() {
    let vc = ConferenceMainViewController()
    let params = StartConferenceParams(roomId: "111111") // 请替换 "111111" 为您自定义的会议号
    params.isSeatEnabled = false          // true:上台发言房间,false:自由发言房间。
    params.password = "12345"             // 请替换 "12345" 为您自定义的会议密码。
    params.name = "YourConferenceName"    // 请替换 "YourConferenceName" 为您自定义的会议名称。
    params.isOpenCamera = true            // 开启摄像头。
    params.isOpenMicrophone = true        // 开启麦克风。
    params.isOpenSpeaker = true           // 开启扬声器。
    params.isCameraDisableForAllUser = true       // 开启全体禁画。
    params.isMicrophoneDisableForAllUser = true   // 开启全体静音。
    vc.setStartConferenceParams(params: params)
    navigationController?.pushViewController(vc, animated: true)
}
```

Flutter

```
var conferenceSession = ConferenceSession.newInstance("您的conferenceId")
      ..isMuteMicrophone = false
      ..isOpenCamera = false
      ..isSoundOnSpeaker = true
      ..name = "您的会议名称"
      ..enableMicrophoneForAllUser = true
      ..enableCameraForAllUser = true
      ..enableMessageForAllUser = true
      ..enableSeatControl = false
      ..onActionSuccess = () {     //操作成功的回调,您可以在这里导航至会议界面
        Navigator.push(
          context,
          MaterialPageRoute(
          builder: (context) => ConferenceMainPage(),
          ),
        );
      }
      ..onActionError = (ConferenceError error, String message) {}   //操作失败的回调
      ..quickStart();
```

- 您可以通过以下方式加入房间，并设置进房参数，完成会前控制：

Android

```
ConferenceDefine.JoinConferenceParams params = new ConferenceDefine.JoinConferenceParams("加入房间的roomId");
params.isOpenCamera = true;      // 开启摄像头
params.isOpenMicrophone = true; // 开启麦克风
params.isOpenSpeaker = true;     // 开启扬声器
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent);
```

iOS

```
import TUIRoomKit

func joinConference() {
    let vc =  ConferenceMainViewController()
    let params = JoinConferenceParams(roomId: "111111")  // 请替换 "111111" 为您要加入的会议号
    params.isOpenCamera = true         // 开启摄像头
    params.isOpenMicrophone = true     // 开启麦克风
    params.isOpenSpeaker = true        // 开启扬声器
    vc.setJoinConferenceParams(params: params)
    navigationController?.pushViewController(vc, animated: true)
}
```

Flutter

```
var conferenceSession = ConferenceSession.newInstance("您的conferenceId")
      ..isMuteMicrophone = false
      ..isOpenCamera = false
      ..isSoundOnSpeaker = true
      ..onActionSuccess = () {    //操作成功的回调，您可以在这里导航至会议界面
        Navigator.push(
          context,
          MaterialPageRoute(
          builder: (context) => ConferenceMainPage(),
          ),
        );
      }
      ..onActionError = (ConferenceError error, String message) {}  //操作失败的回调
      ..join()
```

以下是对上述代码中参数的详细介绍。

字段	类型	含义
isMuteMicrophone	bool	是否静音麦克风（默认为false）
isOpenCamera	bool	是否开启摄像头（默认为false）
isSoundOnSpeaker	bool	是否使用扬声器（默认为true）
password	String	会议密码（默认为空，不开启）

name	String	会议名称（默认为您的conferenceId）
enableMicrophoneForAllUser	bool	是否开启全员的麦克风权限（默认为true）
enableCameraForAllUser	bool	是否开启全员的摄像头权限（默认为true）
enableMessageForAllUser	bool	是否开启全员的发言权限（默认为true）
enableSeatControl	bool	是否开启上台发言模式（默认为false）

注意：

上述是针对上述代码中创建房间并加入房间参数的介绍。如可根据传入 isSeatEnable 参数值的不同来创建自由发言房间和上台发言房间，其中两种类型的房间会中控制可操作功能特性也会不同：

- **自由发言房间**：普通用户可以自由进行发言，且自由开关麦克风和摄像头。
- **上台发言房间**：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。

会中控制

管理自由发言房间

房间类型为自由发言房间时，房主或管理员可以在**成员** > 成员列表中管理会中所有成员。

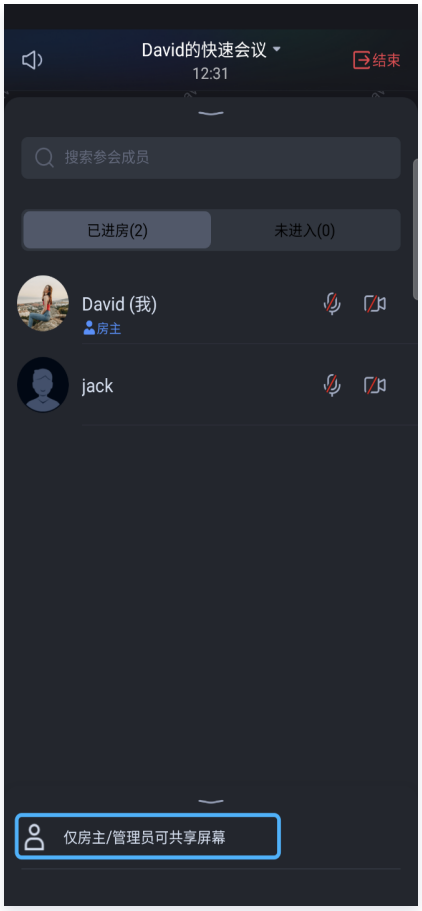
- 主持人或管理员可以选中任一成员进行单独管控：对其**解除静音/静音、开启视频/关闭视频、设置为管理员、转交主持人、禁言/解除禁言、修改名称、踢出房间**。

成员管理	转交房主	踢出房间

- **主持人或管理员可以对房间内所有成员进行全体管控：全体静音/解除全体静音 、 全体禁画/解除全体禁画、仅房主/管理员可共享屏幕（全体成员可共享屏幕）**

全体静音	解除全体静音

开启仅房主/管理员可共享屏幕	如果有普通成员正在共享，给出提示	普通成员无法开启屏幕共享

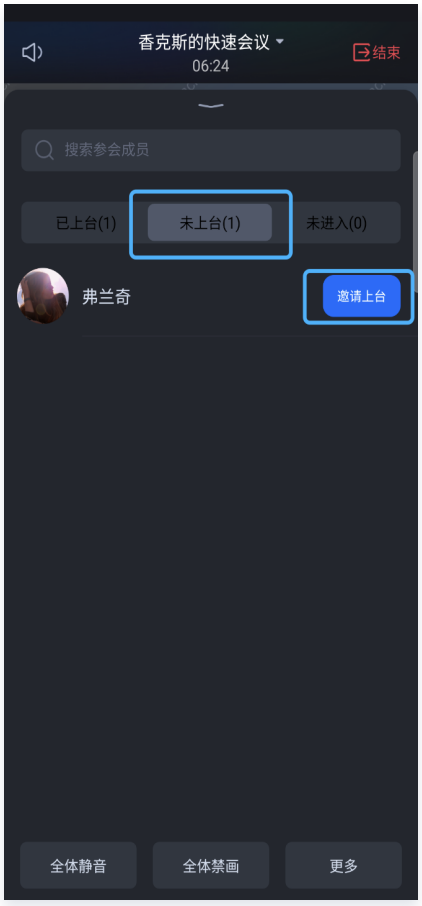


管理上台发言房间

房间类型为上台发言房间时，房主或管理员除了在**成员** > 成员列表中管理会中成员之外，还可以在**上台管理**中针对选中成员的上台状态进行管理。

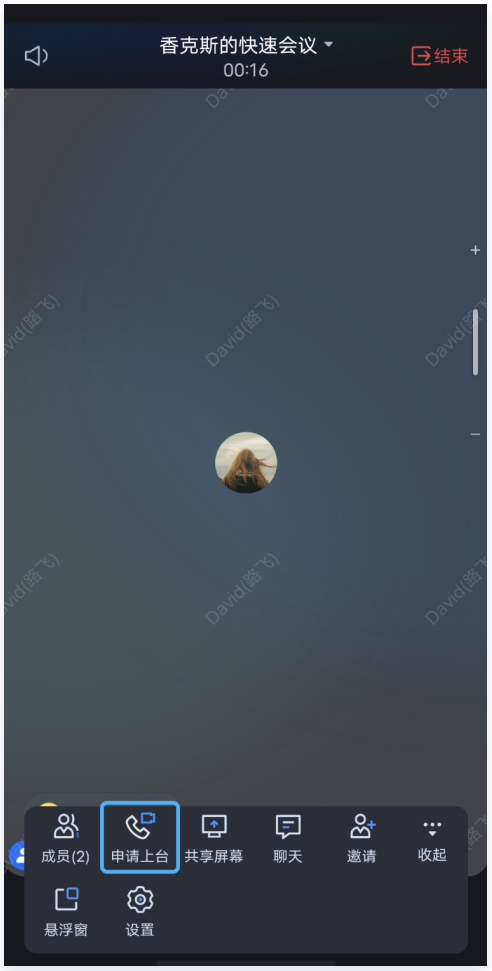
- **主持人或管理员可以选中任一普通成员进行单独管控：除了自由发言房间包含的解除静音/静音、开启视频/关闭视频、禁言/解除禁言、设置为管理员、转交主持人、踢出房间、仅房主/管理员可共享屏幕（全体成员可共享屏幕）之外，还包含上台发言房间独有的邀请上台和请下台等操作。**

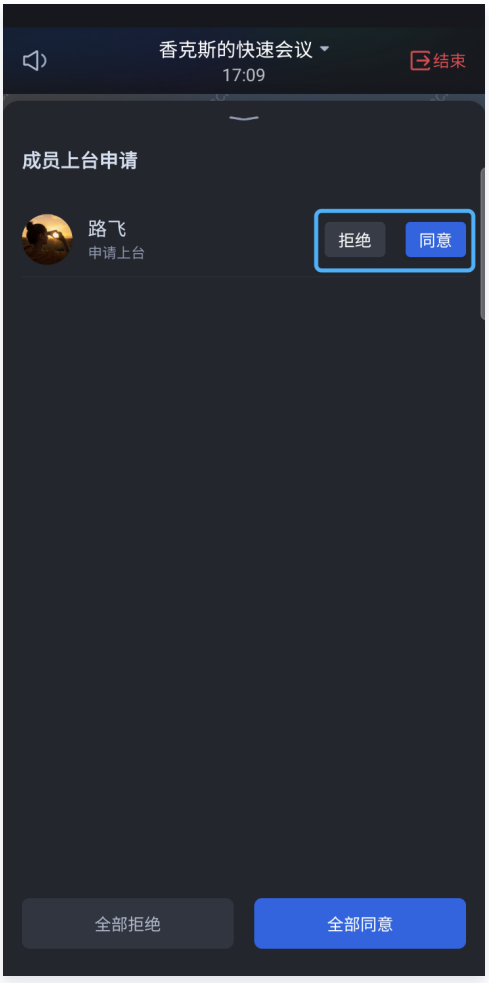
邀请上台（房主或管理员）	被邀请房	踢某人下台（房主或管理员）



- **主持人或管理员可以对房间内已申请上台的成员进行状态管理：可在上台管理中对选中成员进行同意或拒绝，或对所有进行上台申请成员进行全部同意处理。**

普通成员申请上台	房主或管理员审批上台





关键代码

若您想自定义实现会控功能，请参考 TUIRoomEngine SDK： Android 、iOS&Mac、Flutter 。

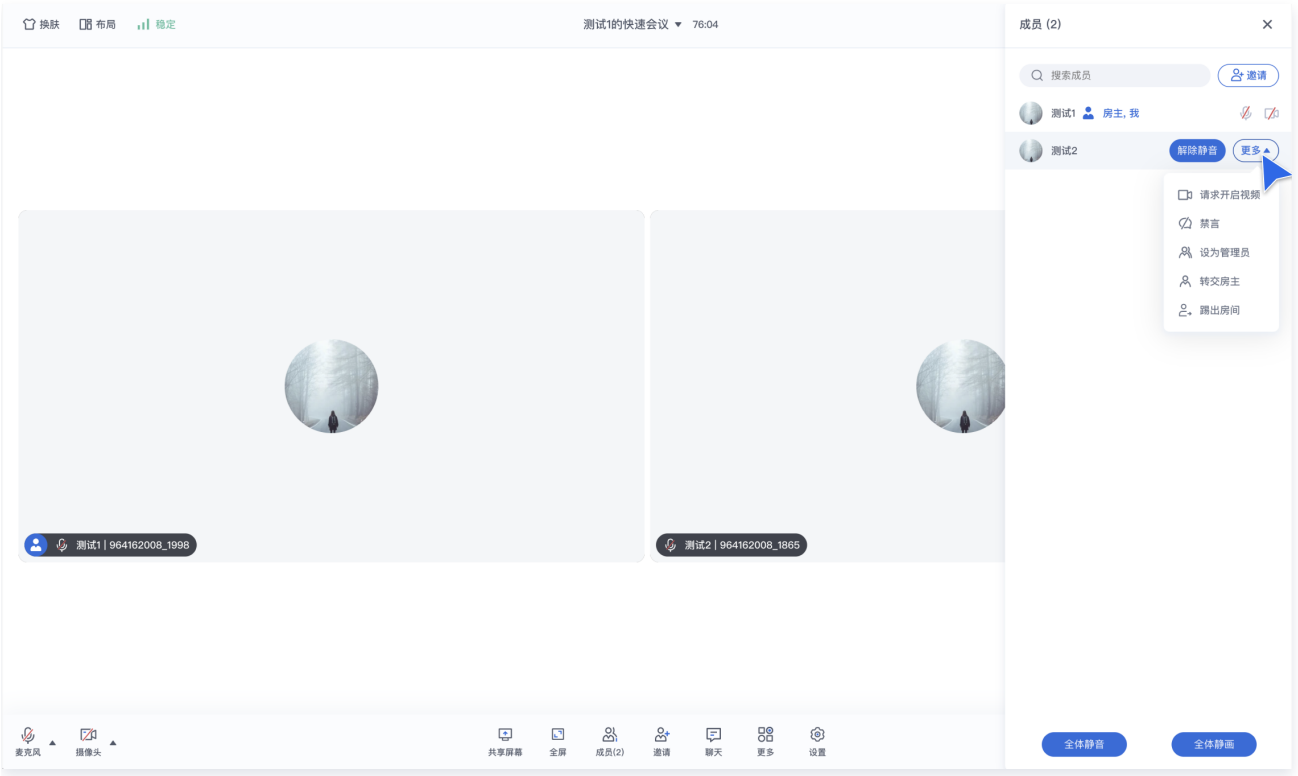
> **说明：**
> 若您还有其他问题，可通过 TUIRoomKit 技术交流平台 zhiliao，向我们反馈。

Web&Electron

最近更新时间：2024-10-14 15:23:31

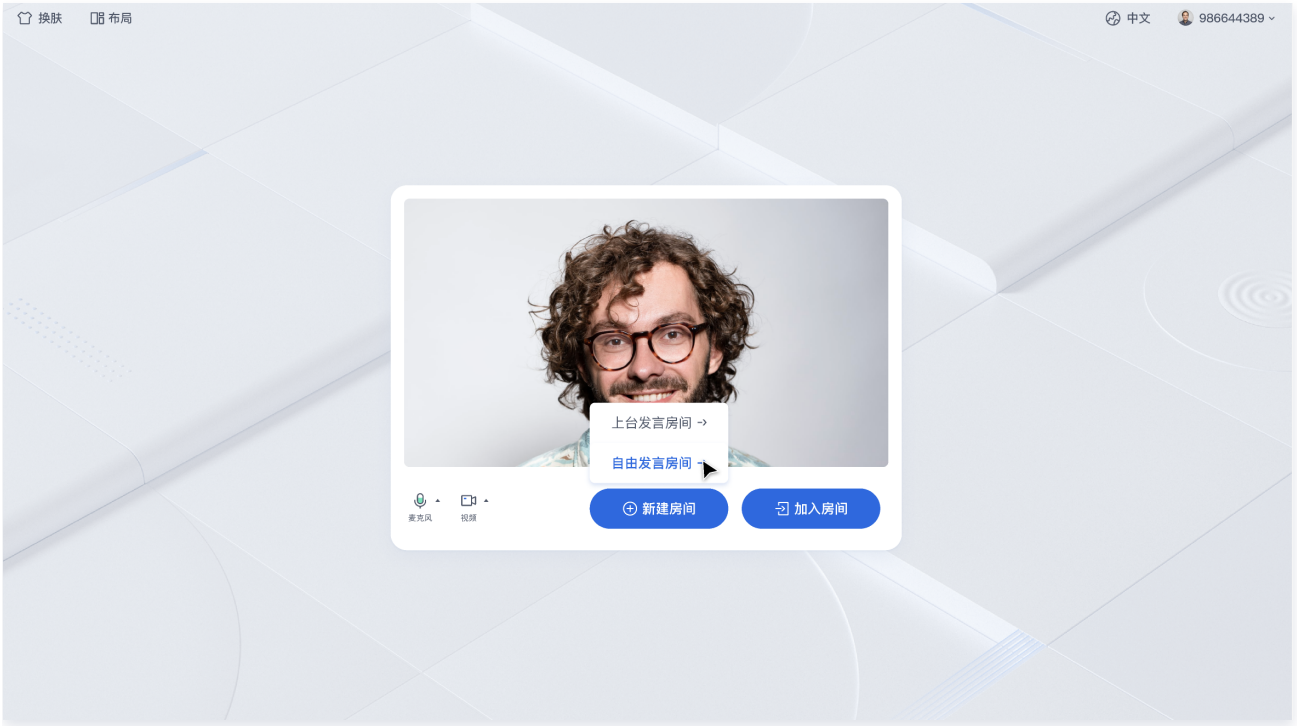
本文档将详细介绍 `TUIRoomKit` 会议控制方面的内容，帮助您更好地掌握 `TUIRoomKit` 在进会前及会议过程中的相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。

`Web` 端和 `Electron` 端用户创建并进入房间后，房主或管理员角色通过点击底部工具栏**成员**按钮，在右侧弹出的成员列表中既可以选中任一普通成员进行**禁言、设为管理员**等会控操作，也可以对房间内所有成员进行**全体静音**等会控操作。



会前控制

在创建并加入房间时，您需要创建并进入房间时预先设定会议的相关参数，通过 `TUIRoomKit` 会前控制的相关功能，确保会议顺利进行。




```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

字段	类型	含义
isSeatEnable	bool	是否开启上台发言房间（默认为 false）
isOpenCamera	bool	是否开启摄像头（默认为 true）
isOpenMicrophone	bool	是否开启麦克风（默认为 true）

⚠ 注意：

上述是针对上述代码中创建房间并加入房间参数的介绍。如可根据传入 isSeatEnable 参数值的不同来创建自由发言房间和上台发言房间，其中两种类型的房间会中控制可操作功能特性也会不同：

- **自由发言房间**：普通用户可以自由进行发言，且自由开关麦克风和摄像头。
- **上台发言房间**：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。

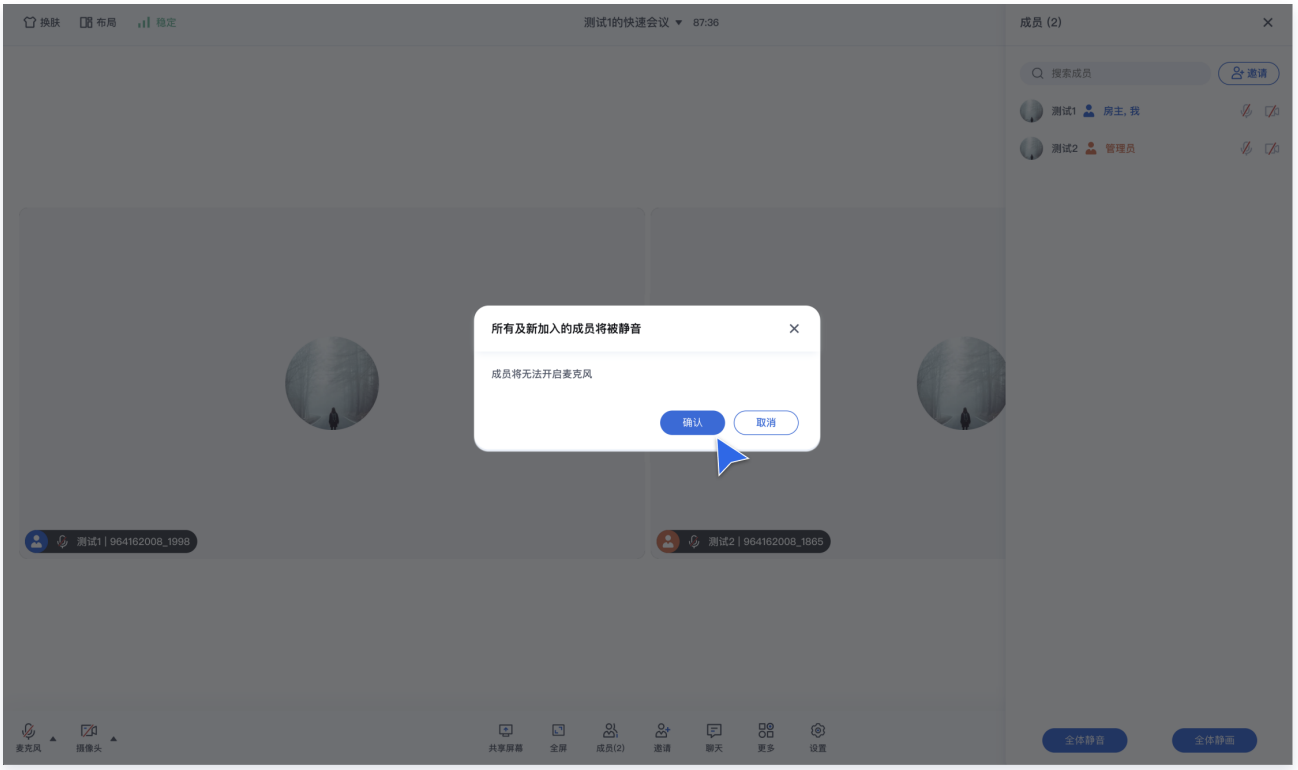
会中控制

管理自由发言房间

房间类型为自由发言房间时，房主或管理员可以在**成员** > 在成员列表中管理会中所有成员。

- 主持人或管理员可以选中任一成员进行单独管控：对其**解除静音/静音**，还可以点击**更多**进行请求**开启视频/关闭视频、禁言/解除禁言、设置为管理员、转交主持人、踢出房间**等操作。

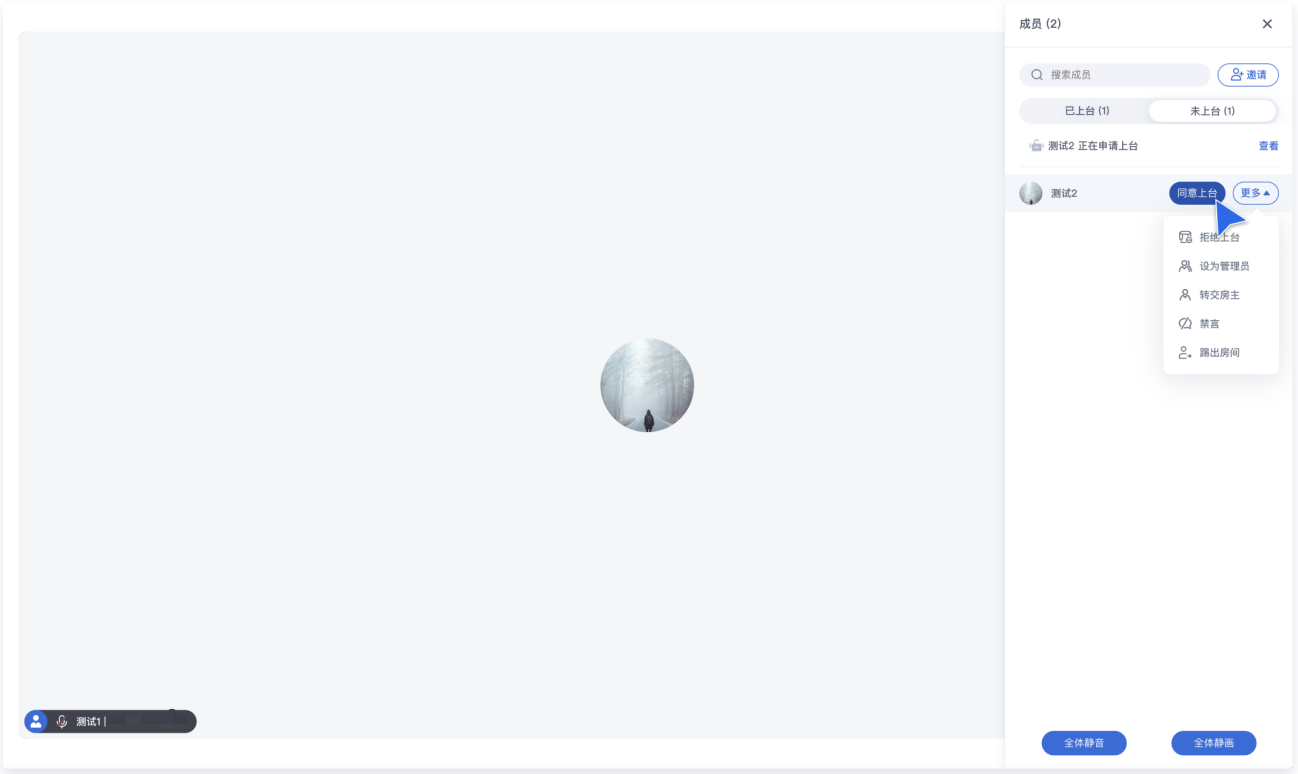
- 主持人或管理员可以对房间内所有成员进行全体管控：**全体静音/解除全体静音**或**全体禁画/解除全体禁画**。



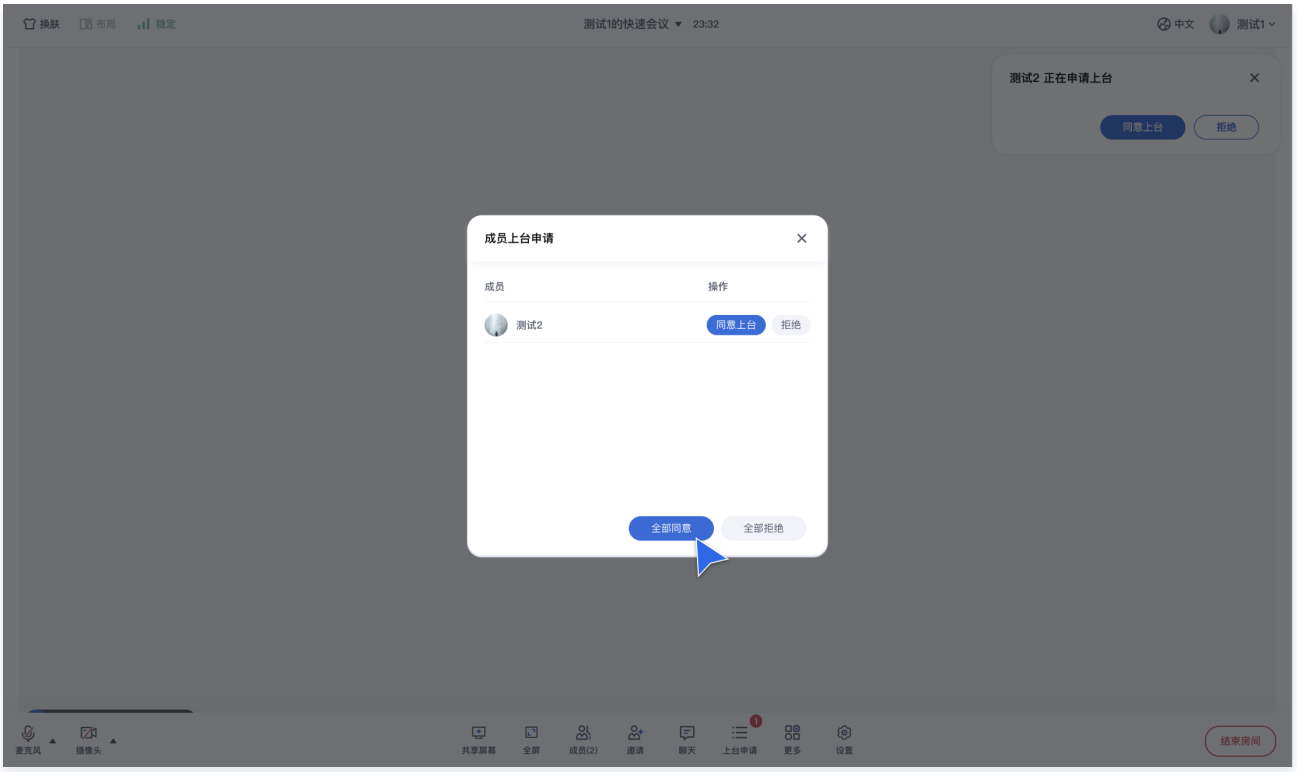
管理上台发言房间

房间类型为上台发言房间时，房主或管理员除了在**成员** > 在成员列表中管理会中成员之外，还可以在**上台申请**中针对选中成员的上台状态进行管理。

- 主持人或管理员可以选中任一普通成员进行单独管控：除了自由发言房间包含的**解除静音/静音、开启视频/关闭视频、禁言/解除禁言、设置为管理员、转交主持人、踢出房间**之外，上台发言房间独有的**邀请上台、同意上台/拒绝上台**和**下台**等操作。



- 主持人或管理员可以对房间内已申请申请上台成员进行状态管理：可在**上台申请**中对选中成员进行**同意**或**拒绝**，或对所有进行上台申请成员进行**全部同意**或**全部拒绝**处理。

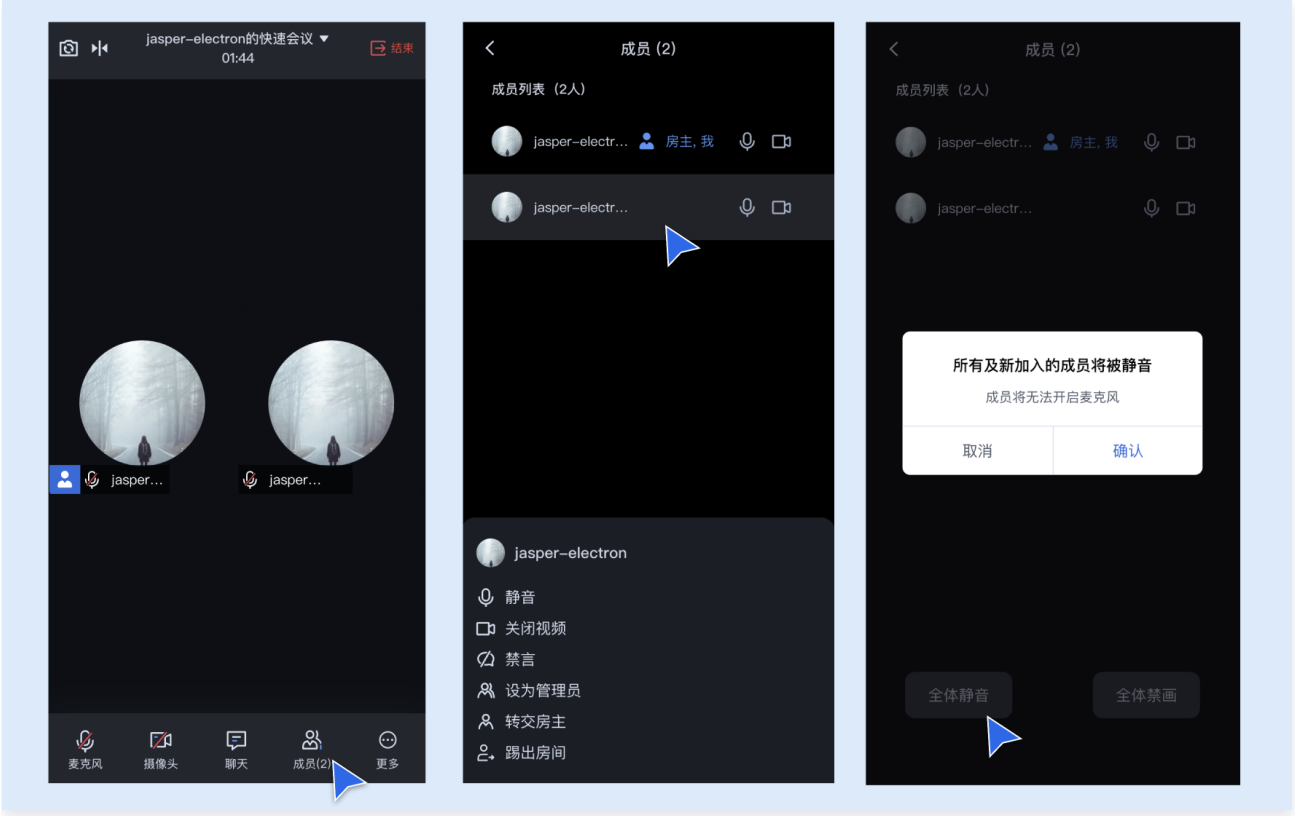


小程序&H5

最近更新时间：2024-05-08 15:16:02

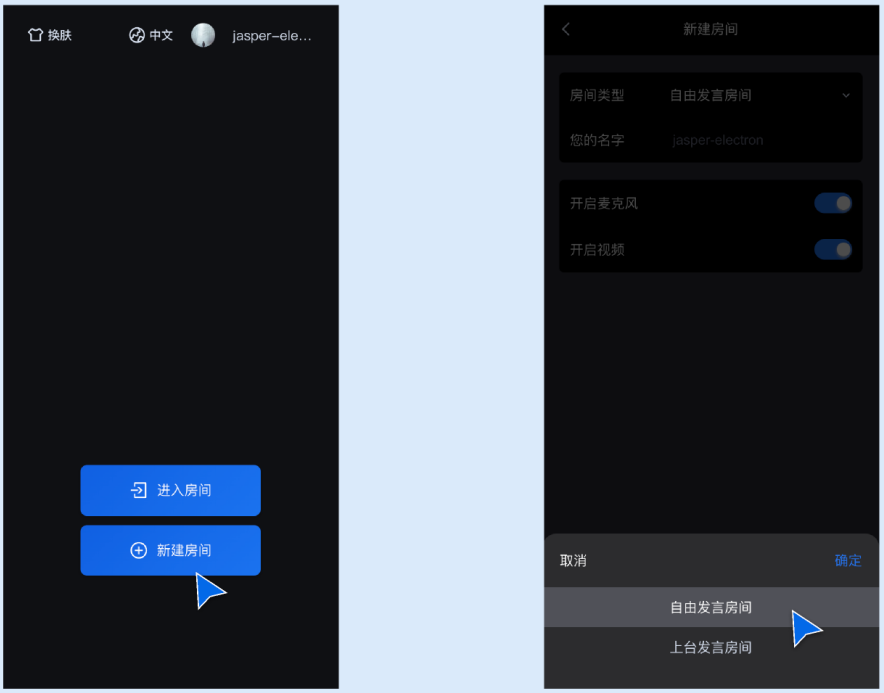
本文档将详细介绍 `TUIRoomKit` 会议控制方面的内容，帮助您更好地掌握 `TUIRoomKit` 在进会前及会议过程中的相关功能操作。通过本文档，您能够充分利用 `TUIRoomKit` 的功能，实现高质量的音视频会议。

`小程序` 端和 `H5` 端用户创建并进入房间后，房主或管理员角色通过点击底部工具栏**成员**按钮，在右侧弹出的成员列表中既可以选中任一普通成员进行**禁言、设为管理员**等会控操作，也可以对房间内所有成员进行**全体静音**等会控操作。



会前控制

在实现上述功能的前提下，您需要创建并进入房间时预先设定会议的相关参数，通过 `TUIRoomKit` 会前控制的相关功能，确保会议顺利进行。




```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

字段	类型	含义
isSeatEnable	bool	是否开启上台发言房间（默认为 false）
isOpenCamera	bool	是否开启摄像头（默认为 true）
isOpenMicrophone	bool	是否开启麦克风（默认为 true）

注意：

上述是针对上述代码中创建房间并加入房间参数的介绍。如可根据传入 isSeatEnable 参数值的不同来创建自由发言房间和上台发言房间，其中两种两种类型的房间会中控制可操作功能特性也会不同：

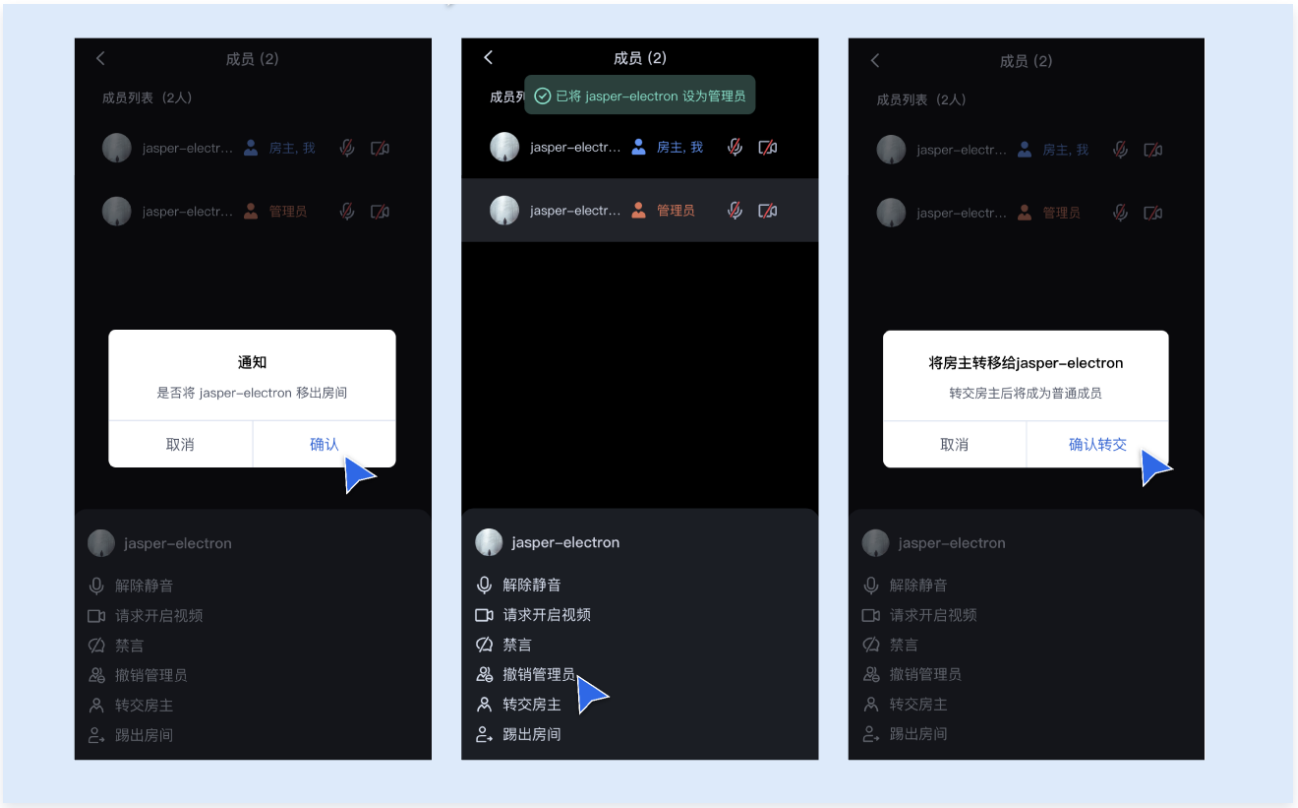
- **自由发言房间**：普通用户可以自由进行发言，且自由开关麦克风和摄像头。
- **上台发言房间**：仅台上用户可自由开启或关闭麦克风和摄像头，普通观众可通过举手申请成为台上用户。

会中控制

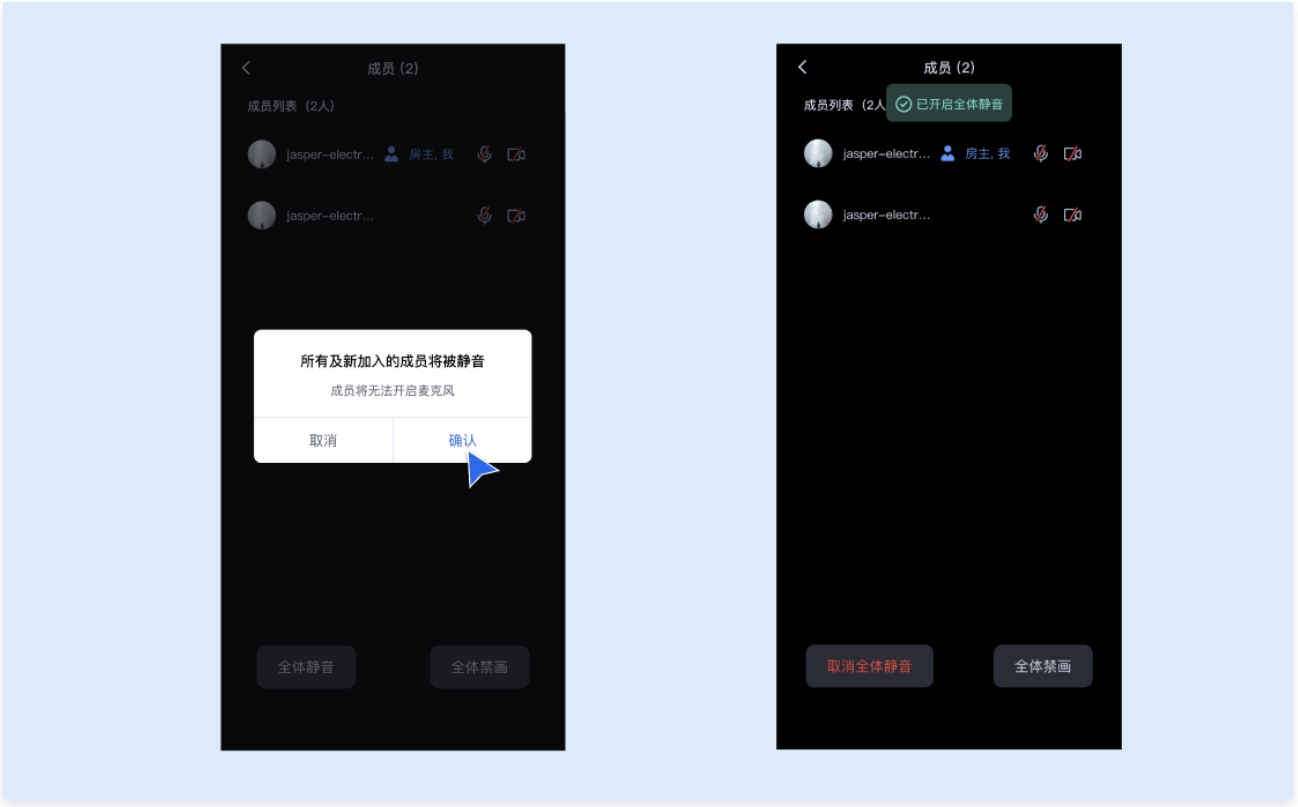
管理自由发言房间

房间类型为自由发言房间时，房主或管理员可以在**成员** > 在成员列表中管理会中所有成员。

- 主持人或管理员可以选中任一成员进行单独管控：对其**解除静音/静音**，还可以点击**更多**进行请求**开启视频/关闭视频、禁言/解除禁言、设置为管理员、转交主持人、踢出房间**。



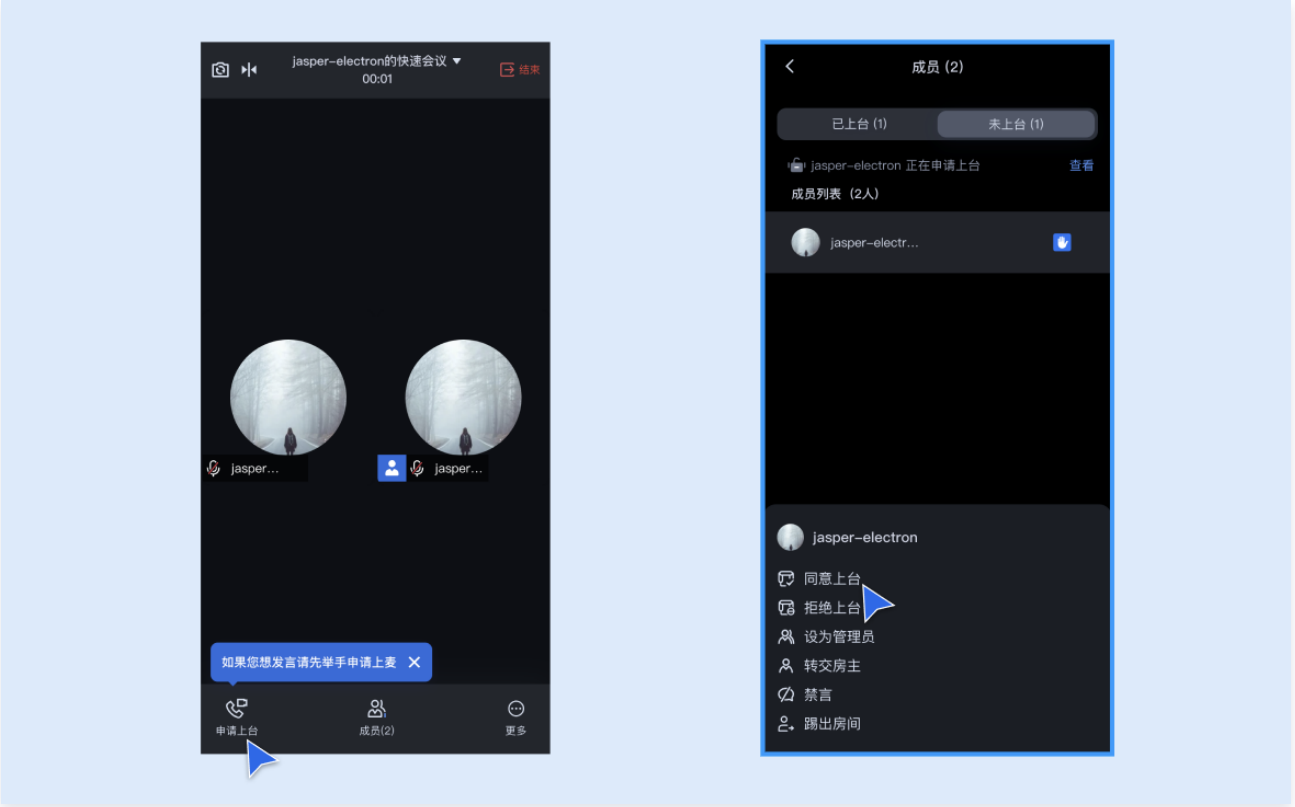
- 主持人或管理员可以对房间内所有成员进行全体管控：**全体静音/解除全体静音**或**全体禁画/解除全体禁画**。



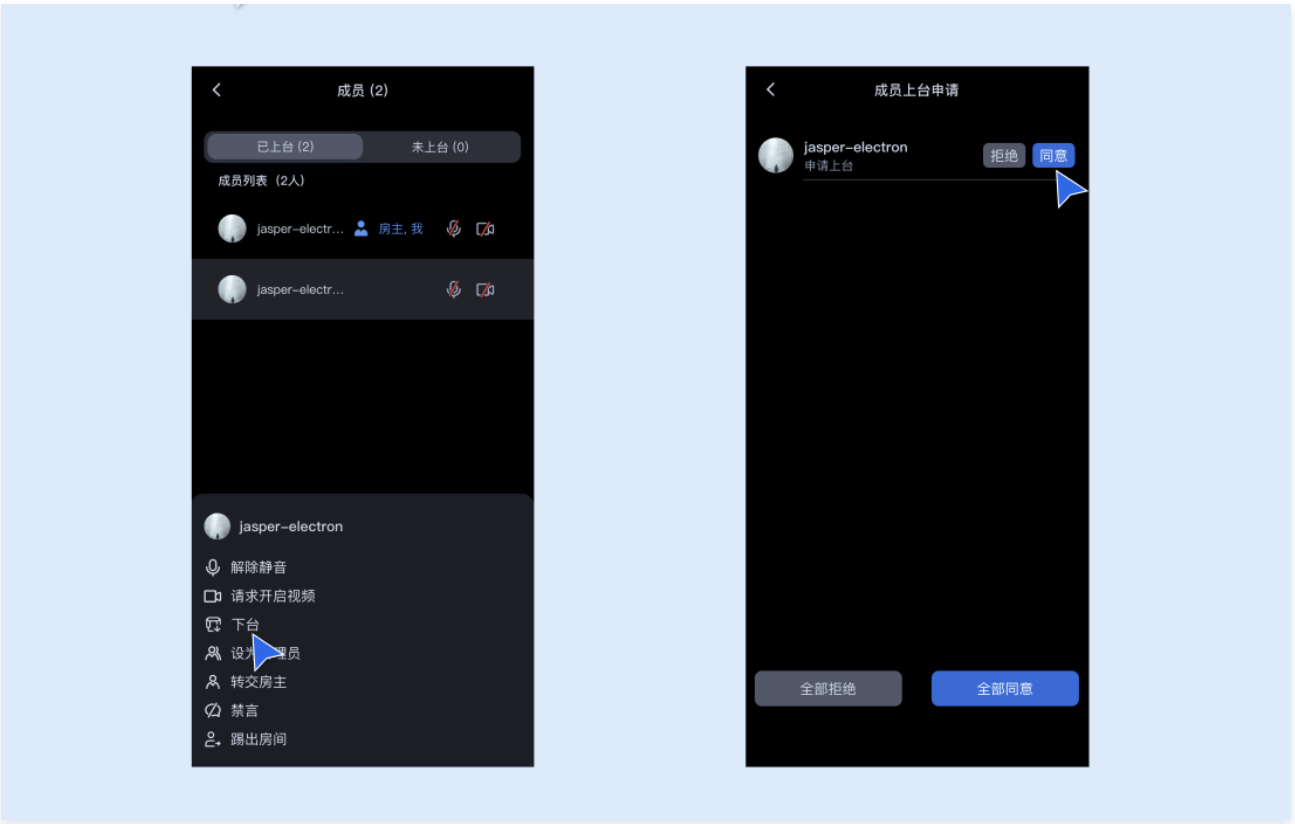
管理上台发言房间

房间类型为上台发言房间时，房主或管理员除了在**成员** > 在成员列表中管理会中成员之外，还可以在**上台申请**中针对选中成员的上台状态进行管理。

- 主持人或管理员可以选中任一普通成员进行单独管控：除了自由发言房间包含的**解除静音/静音、开启视频/关闭视频、禁言/解除禁言、设置为管理员、转交主持人、踢出房间**之外，上台发言房间独有的**邀请上台、同意上台/拒绝上台**和**下台**等操作。



- 主持人或管理员可以对房间内已申请申请上台成员进行状态管理：可在**上台申请**中对选中成员进行**同意**或**拒绝**，或对所有进行上台申请成员进行**全部同意**或**全部拒绝**处理。


成员 (2)
已上台 (2)
未上台 (0)
成员列表（2人）
jasper-electr...
房主, 我
jasper-electr...
jasper-electron
解除静音
请求开启视频
下台
转交房主
禁言
踢出房间
成员上台申请
jasper-electron
申请上台
拒绝
同意
全部拒绝
全部同意

界面定制（TUIRoomKit）
Android

最近更新时间：2025-06-17 12:12:12

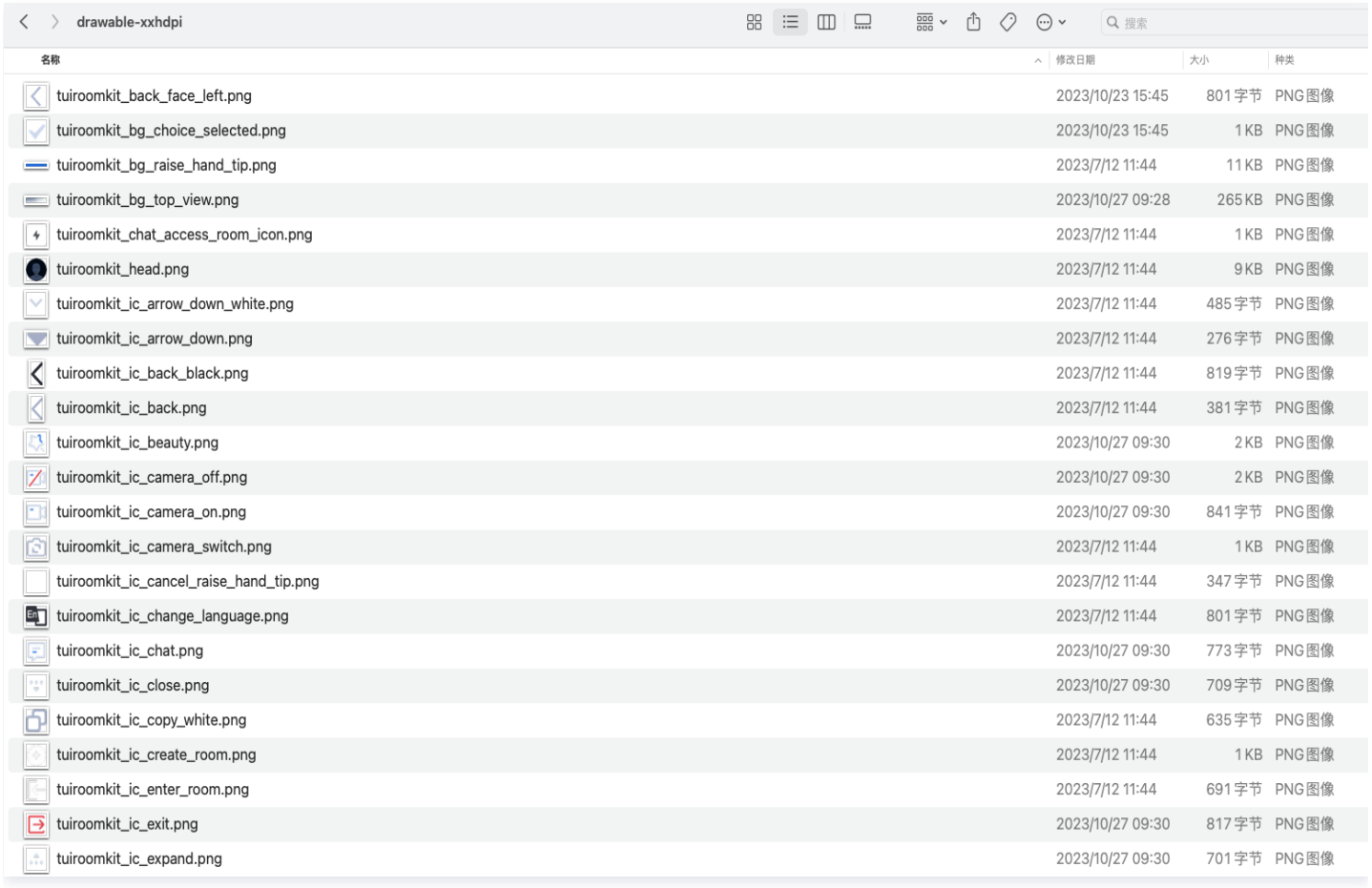
本文将介绍如何定制 TUIRoomKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

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/main/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 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
├── basic
│   ├── BaseBottomDialog.java                  // 通用底部弹框
│   ├── BaseDialogFragment.java                // 通用对话框
│   ├── BaseSettingItem.java                   // 通用设置项
│   ├── ConfirmDialog.java                     // 通用确认对话框
│   ├── NotificationView.java                  // 通用通知栏
│   ├── PrepareView.java                       // 准备界面
│   ├── RoomToast.java                         // 通用普通 Toast
│   ├── RoundRelativeLayout.java               // 通用圆角 RelativeLayout
│   ├── SwitchSettingItem.java                 // 通用 switch 开关
│   └── TipToast.java                          // 通用提示 Toast
├── create                                     // 创建会议界面
│   ├── CreateConferenceActivity.java
│   ├── CreateConferenceView.java
│   └── RoomTypeSelectView.java
├── join                                       // 加入会议界面
│   ├── EnterConferenceActivity.java
│   └── EnterConferenceView.java
├── main                                       // 会议主界面
│   ├── ConferenceMainFragment.java            // 会议主界面 Fragment
│   ├── ConferenceMainView.java                // 会议主界面 View
│   ├── RoomMainActivity.java                  // 会议主界面 Activity
│   ├── RoomWindowManager.java                 // 会议主界面和悬浮窗切换的管理者
│   ├── aisssistant                            // AI 助手
│   │   └── AIAssistantDialog.java
│   ├── bottomnavigationbar                    // 底部栏
│   │   ├── BottomLayout.java
│   │   ├── BottomView.java
│   │   └── SeatRequestCountView.java
│   ├── chat                                   // 聊天弹窗
│   │   └── ChatActivity.java
│   ├── conferenceinvitation                   // 会中邀请
│   │   ├── InvitationReceivedActivity.java
│   │   ├── InvitationReceivedView.java
│   │   └── SlideToAcceptView.java
│   ├── exitroom                               // 退出会议
│   │   └── ExitRoomDialog.java
│   ├── floatwindow                            // 悬浮窗
│   │   ├── screensharingindicate
│   │   │   └── ScreenSharingIndicateFloatView.java
│   │   └── videoplaying                       // 视频播放悬浮窗
│   │       ├── RoomFloatViewService.java
│   │       └── RoomVideoFloatView.java
│   ├── invite                                 // 底部邀请弹窗
│   │   └── InviteDialog.java
│   ├── localaudioindicator                    // 本地音频状态显示
│   │   └── LocalAudioToggleView.java
│   ├── mediasettings                          // 媒体设置，包括音频和视频的设置
│   │   ├── MediaSettingPanel.java
│   │   ├── QualityInfoPanel.java
│   │   ├── VideoFrameRateChoicePanel.java
│   │   └── VideoResolutionChoicePanel.java
```

```
|   ├── raisehandcontrolpanel                // 举手用户管理面板
|   |   ├── RaiseHandApplicationListPanel.java
|   |   └── RaiseHandNotificationView.java
|   ├── roominfo                             // 会议信息
|   |   └── RoomInfoDialog.java
|   ├── screensharecontrol                   // 屏幕共享控制（成员列表更多更多功能）
|   |   └── MoreFunctionDialog.java
|   ├── share                                // 分享会议
|   |   └── ShareRoomDialog.java
|   ├── speechtotext                         // 语音转文字
|   |   ├── SpeechToTextActivity.java
|   |   ├── SpeechToTextRecyclerView.java
|   |   └── SpeechToTextSubtitleView.java
|   ├── topnavigationbar                     // 顶部栏
|   |   ├── AudioRouteSwitchView.java
|   |   ├── CameraSwitchView.java
|   |   ├── ConferenceDurationView.java
|   |   ├── ConferenceNameView.java
|   |   └── TopView.java
|   ├── transferownercontrolpanel            // 转让房主选择面板
|   |   └── TransferMasterPanel.java
|   ├── usercontrolpanel                     // 用户管理面板
|   |   ├── UserListPanel.java
|   |   ├── UserListTypeSelectView.java
|   |   ├── UserRecyclerView.java
|   |   ├── userlistitem                     // 用户管理面板列表项
|   |   |   ├── CallUserView.java
|   |   |   ├── CameraIconView.java
|   |   |   ├── InviteSeatButton.java
|   |   |   ├── ListUserInfoView.java
|   |   |   ├── MicIconView.java
|   |   |   └── ScreenIconView.java
|   |   └── usermanager                      // 用户管理面板点击后的用户管理弹窗
|   |       ├── ModifyNameKeyboard.java
|   |       └── UserManagementPanel.java
|   ├── videoseat                            // 视频流画面布局
|   |   └── TUIVideoSeatView.java
|   └── watermark                            // 水印
|       └── TextWaterMarkView.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_camer
a));
    //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 RTC Room Engine SDK实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

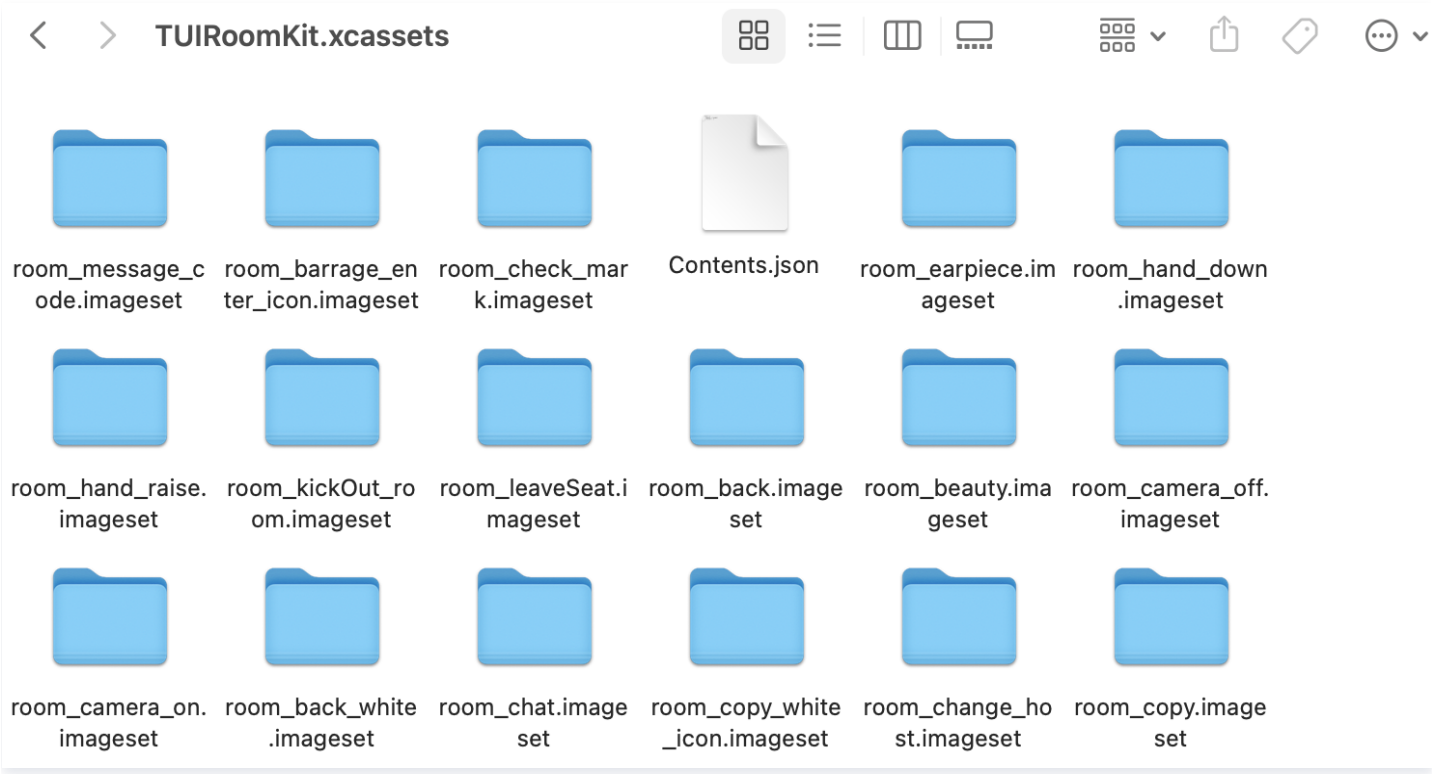
iOS

最近更新时间：2024-09-20 21:28:31

本文将介绍如何定制 TUIRoomKit 的用户界面。通过阅读本文，您将了解在 TUIRoomKit 中进行用户界面定制的各种方案，以满足您特定的应用需求。通过这些方案，您可以灵活地调整和优化 UI 元素，使其更符合您的要求。

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。

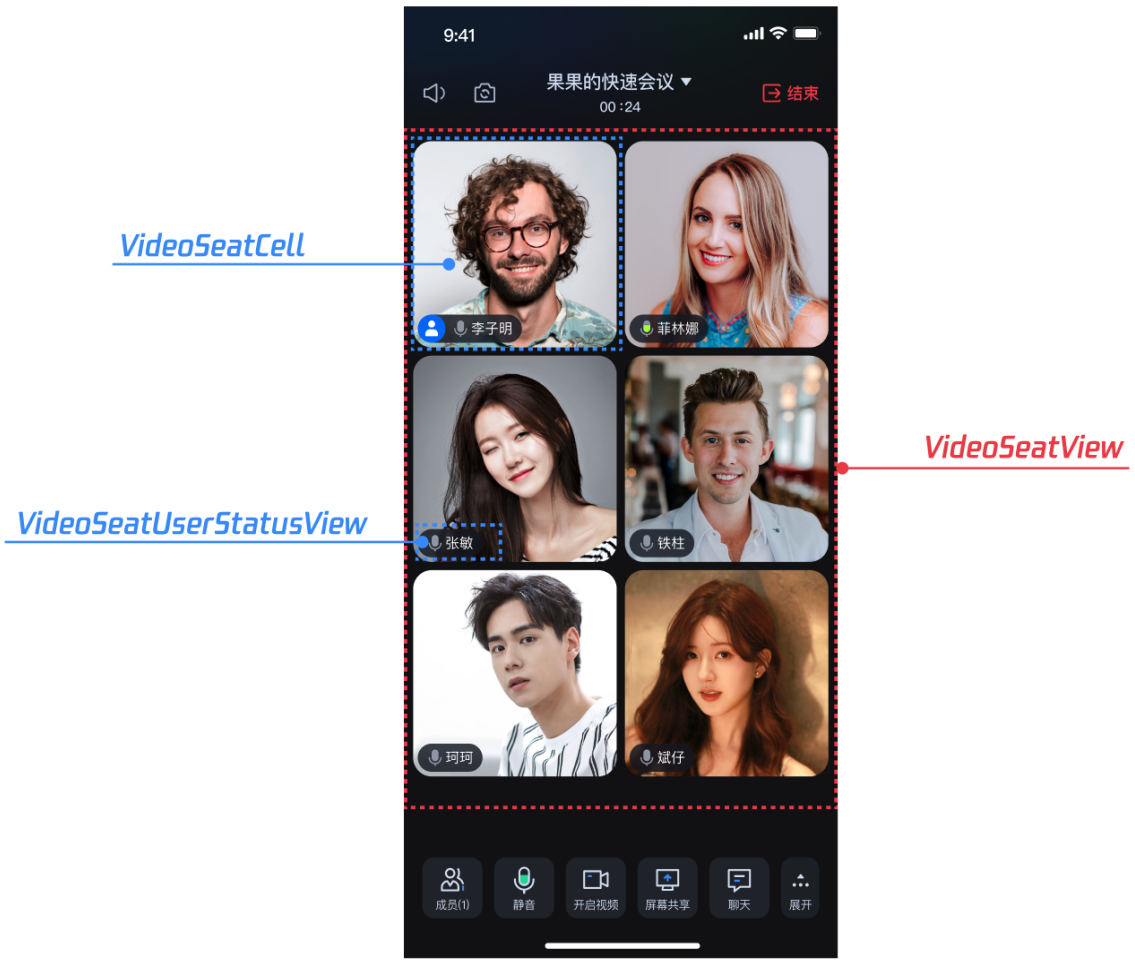


替换文案

您可以通过修改 `TUIRoomKit/Resources/Localized` 文件中的 `TUIRoomKitLocalized.xcstrings` 文件来修改视频会议界面的字符串内容。

调整视频窗口布局

会议主界面中，与视频画面相关的类如图所示：

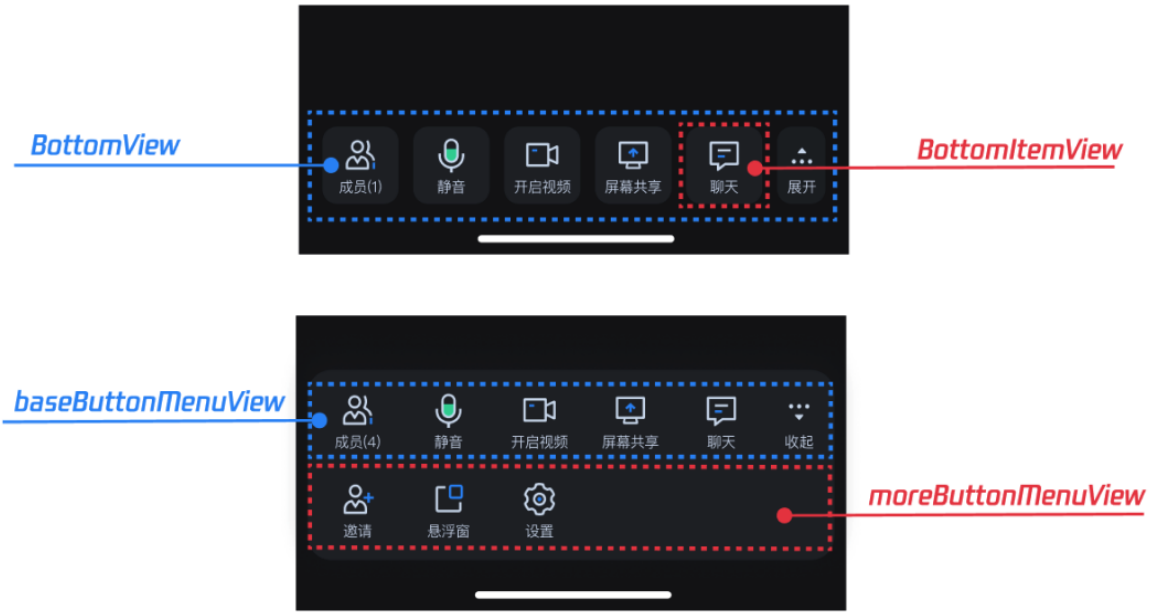


视频画面相关类的文件目录结构如下，您可以通过调整如下 `TUIRoomKit/Source/View` 文件中的内容实现对视频画面的调整。

```
View
└── Page
    ├── ConferenceMainView.swift                    // 会议主界面
    └── Widget
        └── VideoSeat
            ├── ScreenCaptureMaskView.swift         // 本地屏幕共享时显示面板
            ├── VideoSeatCell.swift                 // 视频画面cell
            ├── VideoSeatLayout.swift               // 视频画面布局
            ├── VideoSeatUserStatusView.swift       // 视频画面下方用户信息
            └── VideoSeatView.swift                 // 视频画面整体面板
```

调整底部工具栏

在会议界面的底部工具栏中，有各种与会议进行相关的按钮。与底部栏及其中UI相关的类或对象如图所示：



底部工具栏相关类的文件目录结构如下，您可以通过调整如下 `TUIRoomKit/Source/View` 文件中的内容实现对底部栏的调整。

```
View
└── Page
    └── Widget
        └── BottomNavigationBar
            ├── BottomItemView.swift                    // 底部栏通用按钮
            └── BottomView.swift                        // 底部工具栏
```

底部工具栏按钮的修改

为了方便您自定义底部的功能按钮，我们的 BottomView 通过读取 list 来自动构造，以开关视频按钮为例，代码如下。您可以自行修改按钮的内容以实现您期望的需求。

```
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"
        //设置按钮背景颜色
        muteVideoItem.backgroundColor = UIColor(0xA3AEC7)
        //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)
        }
}
```

调整顶部工具栏

会议主界面中，与顶部工具栏相关的类或对象如图所示：

顶部工具栏相关类的文件目录结构如下，您可以通过调整如下 `TUIRoomKit/Source/View` 文件中的内容实现对顶部栏的调整。

```
View
└── Page
    └── Widget
        └── TopNavigationBar
            ├── TopItemView.swift                    // 顶部栏通用按钮
            └── TopView.swift                        // 顶部工具栏
```

调整其余 UI

当您需要调整其他 UI 时，可以参考如下 `TUIRoomKit/Source/View` 文件下其他 UI 的目录结构。在下方的目录结构中，每个文件所对应的 UI 已被标注，您可以根据您的需求，对您所希望更改的部分 UI 进行更改。

```
View
├── Component
└── Page
    ├── ConferenceMainView.swift                     // 会议主界面
    └── Widget
        ├── Dialog
        |   ├── ExitRoomView.swift                   // 退出房间弹窗
        |   ├── MemberInviteView.swift               // 邀请成员弹窗
        |   ├── RaiseHandNoticeView.swift            // 举手发言提示框
        |   └── RoomInfoView.swift                   // 房间信息弹窗
        ├── FloatWindow
        |   ├── RoomUserStatusView.swift             // 悬浮窗用户信息
        |   └── RoomVideoFloatView.swift             // 悬浮窗
        ├── LocalAudioIndicator
        |   └── LocalAudioView.swift                 // 底部麦克风按钮
```

```
        ├── MediaSettings
        │   ├── MediaSettingView.swift                    // 设置界面
        │   ├── QualityInfoPanel.swift                    // 质量检测面板
        │   └── VideoChoicePanel.swift                    // 视频设置面板
        ├── PopUpControlPanel
        │   └── PopUpView.swift                           // 通用底部弹框
        ├── RaiseHandControlPanel
        │   ├── RaiseHandApplicationCell.swift            // 上台申请列表成员cell
        │   ├── RaiseHandApplicationListView.swift        // 上台申请列表
        │   └── RaiseHandApplicationNotificationView.swift  // 上台申请提示框
        ├── TransferOwnerControlPanel
        │   └── TransferMasterView.swift                  // 房主退房时转移房主面板
        ├── UserControlPanel
        │   ├── UserListCell.swift                        // 用户列表成员cell
        │   ├── UserListManagerView.swift                 // 管理用户面板
        │   └── UserListView.swift                        // 用户列表面板
        └── WaterMark
            ├── FeatureSwitch.swift                       // 水印开关
            ├── WaterMarkLayer.swift                      // 水印view
            └── WaterMarkLineStyle.swift                  // 水印文字样式
```

自定义 UI 方案

TUIRoomKit 的整体功能是基于 RTC Room Engine SDK 实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

Web

最近更新时间：2024-12-12 18:13:12

本文将详细介绍如何定制 TUIRoomkit 的用户界面。TUIRoomkit 提供了两种定制方式，一种是通过简单的自定义 UI API 来进行定制，另一种是替换现有的 UI 组件。下面我们将分别进行详细介绍。

方案一：界面微调

TUIRoomkit 提供了一系列的 API，可以方便地进行 UI 的定制。下表列出了一些主要的 API 和它们的作用：

API	描述
setLanguage	设置界面语言。
setTheme	设置界面主题。
enableWatermark	开启应用程序中的文字水印功能。具体详见：文字水印 。
enableVirtualBackground	开启应用程序的虚拟背景功能。调用此函数后，将在 UI 上显示虚拟背景功能按钮。具体内容详见：虚拟背景 。
disableTextMessaging	禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。
disableScreenSharing	禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。
hideFeatureButton	隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。

按钮隐藏

如下图所示按钮均可通过调用 hideFeatureButton 接口进行隐藏。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue3';
conference.hideFeatureButton(FeatureButton.SwitchTheme);
```

隐藏前	隐藏后

方案二：UIKit 源码导出修改

您可以直接修改我们提供的 UI 源代码，以便根据需求调整 TUIRoomKit 的用户界面。

步骤一：导出 UIKit 源码

Vue3

1. 执行源码导出脚本，默认拷贝路径为 `./src/components/TUIRoom`

```
node ./node_modules/@tencentcloud/roomkit-web-vue3/scripts/eject.js
```

2. 根据脚本提示确认是否要将 TUIRoomKit 源码拷贝到 `./src/components/TUIRoom` 目录。如您需要自定义拷贝目录请输入 'y'，否则输入'n'。

```
● → tuiroomkit-vue3-webpack-ts git:(main) ✗ node ./node_modules/@tencentcloud/roomkit-web-vue3/scripts/eject.js
This script copies the TUIRoomKit source code to the ./src/components/TUIRoom path, do you want to change the copy path? (y|n)y
Please enter your copy path: ./src/TUIRoomKit
  【SUCCESS】The TUIRoomKit source code has been successfully copied to the ./src/TUIRoomKit path!
○ → tuiroomkit-vue3-webpack-ts git:(main) ✗
```

3. 源码导出后，在您指定的项目路径中会新增 TUIRoomKit 源码。此时，您需要手动将 ConferenceMainView 组件，conference 对象的引用从 npm 包地址更改为 TUIRoom 源码的相对路径地址。

```
- import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue3';
// 替换引用路径为 TUIRoomKit 源码的真实路径
+ import { ConferenceMainView, conference } from './src/components/TUIRoom/index.ts';
```

Vue2

1. 执行源码导出脚本，默认拷贝路径为 `./src/components/TUIRoom`

```
node ./node_modules/@tencentcloud/roomkit-web-vue2.7/scripts/eject.js
```

2. 根据脚本提示确认是否要将 TUIRoomKit 源码拷贝到 `./src/components/TUIRoom` 目录。如您需要自定义拷贝目录请输入 'y'，否则输入 'n'。

```
● → tuiroomkit-vue2.7-js git:(main) ✗ node ./node_modules/@tencentcloud/roomkit-web-vue2.7/scripts/eject.js
This script copies the TUIRoomKit source code to the ./src/components/TUIRoom path, do you want to change the copy path? (y|n)y
Please enter your copy path: ./src/TUIRoomKit
  【SUCCESS】The TUIRoomKit source code has been successfully copied to the ./src/TUIRoomKit path!
○ → tuiroomkit-vue2.7-js git:(main) ✗
```

3. 源码导出后，在您指定的项目路径中会新增 TUIRoomKit 源码。此时，您需要手动将 ConferenceMainView 组件，conference 对象的引用从 npm 包地址更改为 TUIRoom 源码的相对路径地址。

```
- import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue2.7';
// 替换引用路径为 TUIRoomKit 源码的真实路径
+ import { ConferenceMainView, conference } from './src/components/TUIRoom/index.ts';
```

步骤二：配置 UIKIT 源码开发环境

配置 Vue3 + Vite + TS 开发环境

1. **安装开发环境依赖**

```
npm install typescript -S -D
```

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 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导致运行报错，您可以在 `.eslintignore` 中添加忽略 TUIRoom 文件夹。

```
// 请添加 TUIRoom 源码真实路径
src/components/TUIRoom
```

4. 此时您可以运行项目查看源码导入效果（如果您在项目 build 的过程中，出现打包报错，具体可参见文档，Web 常见问题）

```
npm run dev
```

配置 Vue3 + Webpack + TS 开发环境

1. **安装开发环境依赖**

```
npm install typescript -S -D
```

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 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。

```
// 请添加 TUIRoom 源码真实路径
src/components/TUIRoom
```

4. 此时您可以运行项目查看源码导入效果（如果您在项目 build 的过程中，出现打包报错，具体可参考文档，Web 常见问题）

```
npm run serve
```

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. **升级 vue2 版本到 v2.7+，如果您项目的 vue 版本已经是 v2.7+, 忽略此步骤**

```
npm install vue@2 -S
```

2. **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
```

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 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导致运行报错，您可以在 `.eslintignore` 中添加忽略 TUIRoom 文件夹。

```
// 请添加 TUIRoom 源码真实路径
src/components/TUIRoom
```

5. 此时您可以运行项目查看源码导入效果。（如果您在项目 build 的过程中，出现打包报错，具体可参考文档，Web 常见问题 ）

```
npm run serve
```

步骤三：根据需求修改源码

1. 替换图标

您可以直接修改 `/TUIRoom/components/common/icons` 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。

2. 调整 UI 布局

您可以通过修改 `TUIRoom/components/` 路径下的不同组件来调整多人视频会议界面的 UI 布局

```
- components/
    - Chat                聊天
    - common              公共组件
    - 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
```

方案三：自实现 UI 方法

TUIRoomKit 的整体功能是基于 RTC Room Engine SDK 实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

Electron (Windows&Mac)

最近更新时间：2024-11-28 15:30:42

本文将详细介绍如何定制 TUIRoomkit 的用户界面。TUIRoomkit 提供了两种定制方式，一种是通过简单的自定义 UI API 来进行定制，另一种是替换现有的 UI 组件。下面我们将分别进行详细介绍。

方案一：界面微调

TUIRoomkit 提供了一系列的 API，可以方便地进行 UI 的定制。下表列出了一些主要的 API 和它们的作用：

API	描述
setLanguage	设置界面语言。
setTheme	设置界面主题。
enableWatermark	开启应用程序中的文字水印功能。具体详见：文字水印 。
disableTextMessaging	禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。
disableScreenSharing	禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。
hideFeatureButton	隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。

按钮隐藏

如下图所示按钮均可通过调用 hideFeatureButton 接口进行隐藏。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference, FeatureButton } from '@tencentcloud/roomkit-electron-vue3';
conference.hideFeatureButton(FeatureButton.SwitchTheme);
```

隐藏前	隐藏后

方案二：UIKit 源码导出修改

您可以直接修改我们提供的 UI 源代码，以便根据需求调整 TUIRoomKit 的用户界面。执行以下 node 脚本，将 TUIRoomKit 的界面源代码自动复制到您的项目中（默认路径 `./src/components/TUIRoom` ）。

> **说明：**
> 在导出源码后，您需要手动将 TUIRoom 组件的引用从 npm 包地址更改为 TUIRoom 源码的相对路径地址。

Vue3

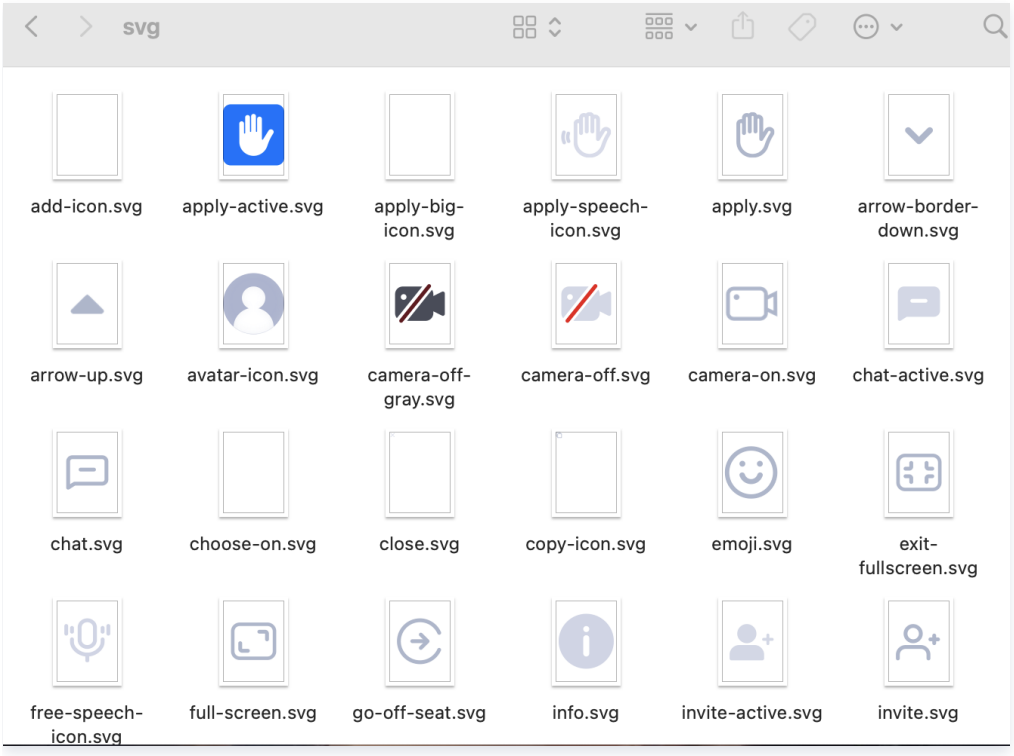
```
node ./node_modules/@tencentcloud/roomkit-electron-vue3/scripts/eject.js
```

Vue2

```
node ./node_modules/@tencentcloud/roomkit-electron-vue2.7/scripts/eject.js
```

1. 替换图标

您可以直接修改 `/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。



2. 调整 UI 布局

您可以通过修改 `/TUIRoom/components/` 文件下的不同组件来调整多人视频会议界面的UI布局

```
- components/
    - 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
```

方案三：自实现 UI 方法

TUIRoomKit 的整体功能是基于 RTC Room Engine SDK 实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

uni-app (小程序)

最近更新时间：2024-09-20 21:28:31

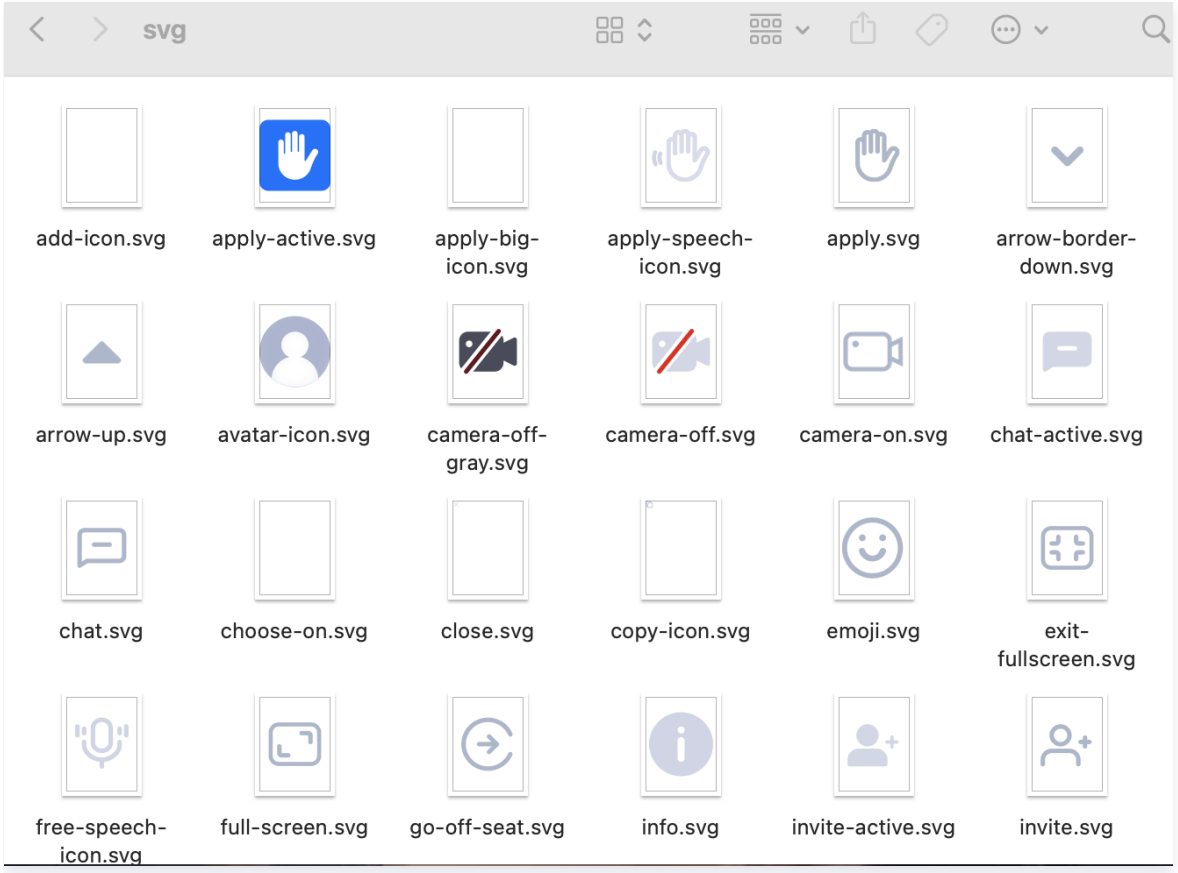
本文将介绍如何定制 TUIRoomkit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自定义 UI 方案**。

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 TUIRoom 的用户界面进行调整，TUIRoom 的界面源代码位于 Github 中的 MiniProgram/ 文件夹下面：

替换图标

您可以直接修改 **MiniProgram/src/roomkit/TUIRoom/assests/icons** 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 **MiniProgram/src/roomkit/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

TUIRoomKit 的整体功能是基于 RTC Room Engine SDK 实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

Flutter

最近更新时间：2024-09-20 21:28:31

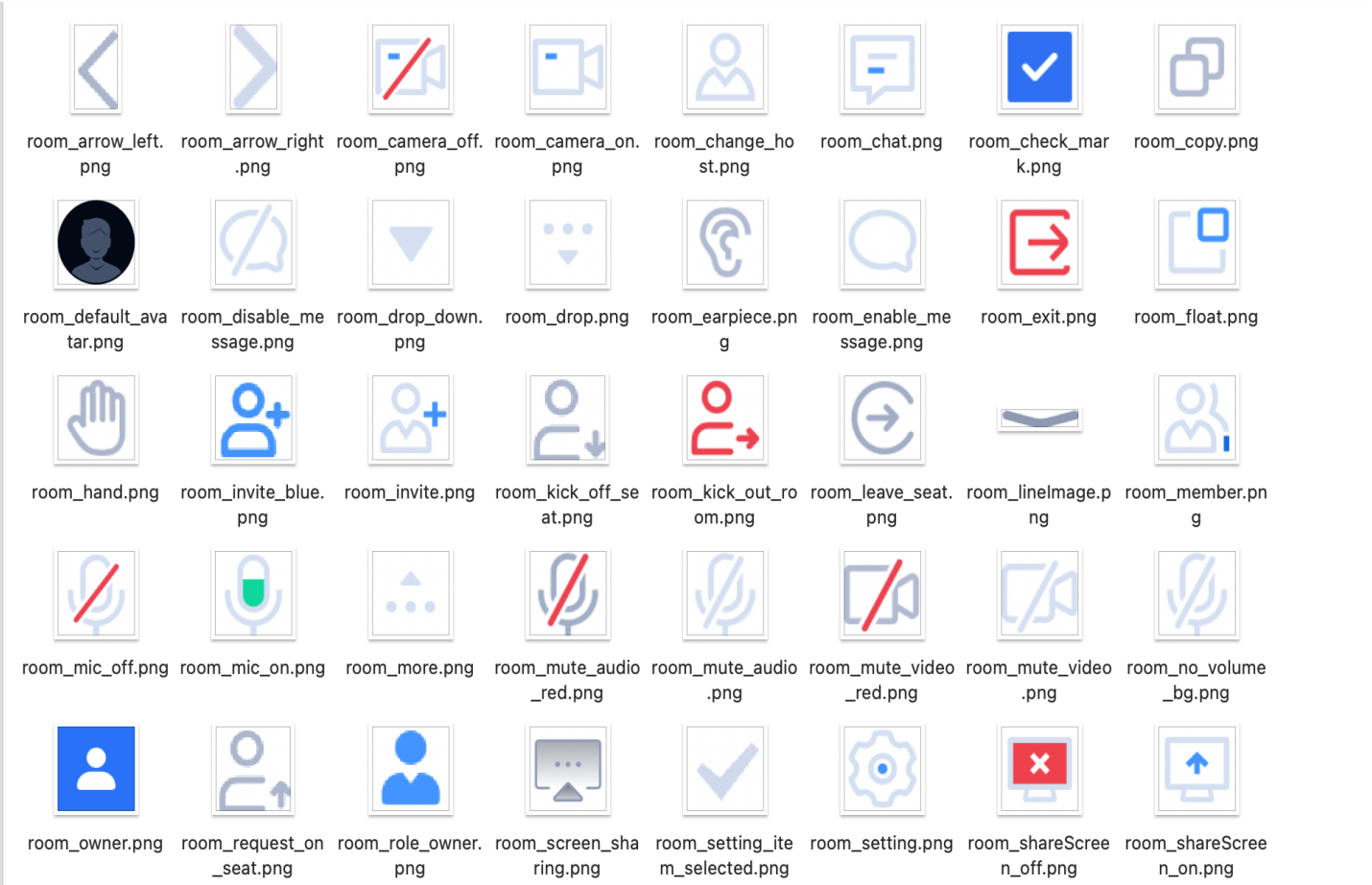
本文将介绍如何定制 `TUIRoomKit` 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

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 (4 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
|       ├── 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 (3 KB)                       // 通用用户信息item组件
|       └── volume_bar.dart (2 KB)                      // 通用动态麦克风音量条
├── pages
|   └── conference_main
|       ├── view.dart (3 KB)
|       └── widgets
|           ├── bottom_view
|           |   ├── view.dart (1 KB)                    // 底部功能按钮文件
|           |   └── widgets
|           |       ├── base_button.dart (10 KB)        // 未展开的基础底部功能按钮
|           |       ├── bottom_button_item.dart (2 KB)  // 封装的通用底部单个item按钮
|           |       ├── mic_button.dart (1 KB)          // 底部栏隐藏后弹出的麦克风按钮
|           |       └── more_button.dart (3 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
|           ├── invite_sheet
|           |   └── invite_sheet.dart (1 KB)            // 邀请弹窗
|           ├── local_screen_sharing
|           |   └── 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 (3 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 (3 KB)                    // 顶部功能按钮
|           |   └── widgets
|           |       ├── meeting_title.dart (2 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 (7 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_layout
|           |   |   ├── view.dart (3 KB)                    // 单页视频画面中视频item的布局方式
|           |   |   └── widgets
|           |   |   |   ├── video_item                      // 单个视频画面item文件夹
|           |   |   |   |   ├── view.dart (4 KB)            // 单个视频画面item
|           |   |   |   |   └── widgets
|           |   |   |   |       ├── video_user_info.dart (2 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
|           |   |   |   └── 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
```

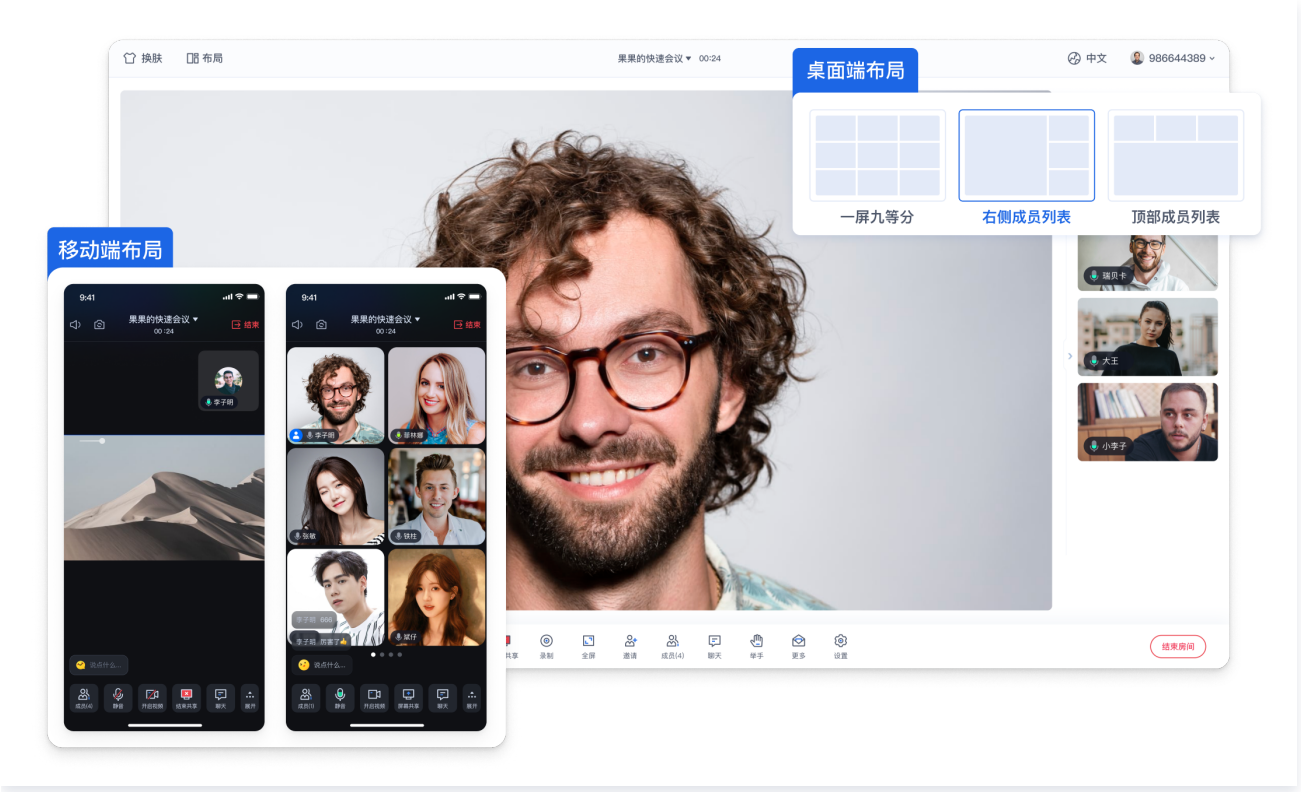
方案三：自定义全部 UI 方案

TUIRoomKit 的整体功能是基于 RTC Room Engine SDK 实现的，您可以完全基于 RTC Room Engine SDK 实现一套自己的 UI 界面。详情可见
RTC Room Engine SDK 。

自定义流布局（TUIRoomKit）
Web

最近更新时间：2024-12-27 09:50:32

TUIRoomKit PC 端提供九宫格布局、侧边栏布局和顶部栏布局三种流布局方式供用户切换；而 TUIRoomKit H5 及小程序端则支持用户切换演讲者视图和六宫格布局。
本文将介绍 TUIRoomKit 流布局的微调方法及如何快速自定义流布局。



Roomkit 流布局微调指引

以下代码展示了 TUIRoomKit 已经支持的流布局的具体实现，如果您需要对已经支持布局进行微调，可以在 导出 TUIRoomKit 源码 后对 `TUIRoom/components/RoomContent/StreamContainer` 文件中的布局样式及参数进行调整。

> **说明：**
> SingleStreamView 组件及 MultiStreamView 组件的详细使用说明请查看 TUIRoomKit 流布局基础组件介绍 。

PC 端宫格布局

```
<template>
  <div class="stream-container">
    <multi-stream-view
      :maxColumn="maxColumn"
      :maxRow="maxRow">
    </multi-stream-view>
  </div>
</template>
```

```
<script setup>
  const maxColumn = ref(3);
  const maxRow = ref(3);
</script>

<style lang="scss">
  div.stream-container {
    width: 100%;
    heght: 100%;
  }
</style>
```

PC 端侧边栏布局

```
<template>
  <div class="stream-container">
    <single-stream-view
      class="single-stream-view"
      :streamInfo="enlargeStreamInfo"
    </single-stream-view>
    <multi-stream-view
      class="multi-stream-view"
      :maxColumn="maxColumn"
      :maxRow="maxRow"
      :excludeStreamInfoList="excludeStreamInfoList"
      @stream-view-dblclick="handleStreamViewDblclick">
    </multi-stream-view>
  </div>
</template>

<script setup>
  import { TUIVideoStreamType } from '@tencentcloud/tuiroom-engine-js';
  const enlargeStreamInfo = ref({
    userId: 'user_master',
    streamType: TUIVideoStreamType.kCameraStream,
  })
  const maxColumn = ref(1);
  const maxRow = ref(Infinity);
  const excludeStreamInfoList = computed(() => {
    return [enalrgeStreamInfo];
  })

  function handleStreamViewDblclick(streamInfo) {
    enlargeStreamInfo.value = streamInfo;
  }
</script>

<style lang="scss">
  div.stream-container {
    width: 100%;
    heght: 100%;
    display: flex;
    .single-stream-view {
```

```
      flex: 1;
    }
    .multi-stream-view {
      width: 240px;
      height: 100%;
    }
  }
</style>
```

PC 顶部栏布局

```
<template>
  <div class="stream-container">
    <single-stream-view
      class="single-stream-view"
      :streamInfo="enlargeStreamInfo"
    </single-stream-view>
    <multi-stream-view
      class="multi-stream-view"
      :maxColumn="maxColumn"
      :maxRow="maxRow"
      :excludeStreamInfoList="excludeStreamInfoList"
      @stream-view-dblclick="handleStreamViewDblclick">
    </multi-stream-view>
  </div>
</template>

<script setup>
  import { TUIVideoStreamType } from '@tencentcloud/tuiroom-engine-js';
  const enlargeStreamInfo = ref({
    userId: 'user_master',
    streamType: TUIVideoStreamType.kCameraStream,
  })
  const maxColumn = ref(Infinity);
  const maxRow = ref(1);
  const excludeStreamInfoList = computed(() => {
    return [enalrgeStreamInfo];
  })

  function handleStreamViewDblclick(streamInfo) {
    enlargeStreamInfo.value = streamInfo;
  }
</script>

<style lang="scss">
  div.stream-container {
    width: 100%;
    heght: 100%;
    display: flex;
    flex-direction: column-reverse;
    .single-stream-view {
      flex: 1;
    }
```

```
    .multi-stream-view {
      width: 100%;
      height: 175px;
    }
  }
</style>
```

移动端布局

```
<template>
  <div ref="streamContainerRef" class="stream-container">
    <TuiSwiper
      :active-index="swiperActiveIndex"
      :show-page-dots="showSwiperPageDots"
      @change="handleSwiperChange"
      class="page-stream-container"
    >
      <TuiSwiperItem v-if="showLargeAndFloatPage">
        <single-stream-view
          class="enlarged-stream"
          :streamInfo="enlargeStream"
          :support-touch-scale="isEnlargeScreenStream"
        />
        <single-stream-view
          v-if="floatStream"
          class="float-stream"
          :style="floatStreamStyle"
          :streamInfo="floatStream"
        />
      </TuiSwiperItem>
      <TuiSwiperItem
        v-for="(item, index) in equalPointsLayoutStreamList"
        :key="index"
      >
        <multi-stream-view
          class="page-stream-container"
          :maxColumn="maxColumn"
          :maxRow="maxRow"
          :stream-info-list="item"
          @stream-view-dblclick="handleStreamViewDblclick"
        />
      </TuiSwiperItem>
    </TuiSwiper>
  </div>
</template>

<script setup>
  import { TUIVideoStreamType } from '@tencentcloud/tuiroom-engine-js';
  const maxColumn = ref(2);
  const maxRow = ref(3);

  const enlargeStream = ref({
    userId: 'user_master',
```

```
    streamType: TUIVideoStreamType.kCameraStream,
  })
  const floatStream = ref({
    userId: 'user_me',
    streamType: TUIVideoStreamType.kCameraStream,
  });
  const equalPointsLayoutStreamList: ComputedRef<StreamInfo[][]> = computed(
    () => {
      return [
        ...new Array(
          Math.ceil(cameraStreamList.value.length / maxCountEveryPage.value)
        ).keys(),
      ].map((pageIndex: number) => {
        return cameraStreamList.value.slice(
          pageIndex * maxCountEveryPage.value,
          (pageIndex + 1) * maxCountEveryPage.value
        );
      });
    }
  );

  function handleStreamViewDblclick(streamInfo) {
    enlargeStreamInfo.value = streamInfo;
  }
</script>

<style lang="scss">
  div.stream-container {
    width: 100%;
    heght: 100%;
  }
  .enlarge-stream {
    width: 100%;
    heght: 100%;
  }
  .float-stream {
    position: absolute;
    top: 0;
    right: 0;
    width: 100px;
    height: 100px;
  }
</style>
```

Roomkit 流布局基础组件

RoomKit 内部提供 singleStreamView 及 MultiStreamView 流渲染组件。使用以下组件，开发者可以快速自定义流布局交互模式。

SingleStreamView 组件

SingleStreamView 用于播放指定用户的流。当当前用户正在推流时，会自动播放；若无流，将显示用户的头像信息。

组件属性

参数	说明	类型	默认值

streamInfo	当前区域播放的流信息	{
		userId: string;
		streamType: TUIVideoStreamType;
		}
			-

fillMode	流播放时的填充模式	'fill'/'contain' 'fill'： 视频流画面填充整个区域，如果视频流的宽高比与容器不匹配，视频流会被拉伸 'contain'：视频流画面保持原始宽高比播放，如果视频流的宽高比与容器不匹配，视频流被填充黑边	PC 端默认为 'contain'，移动端默认为 'fill'， 屏幕分享统一为 'contain' 模式

组件事件

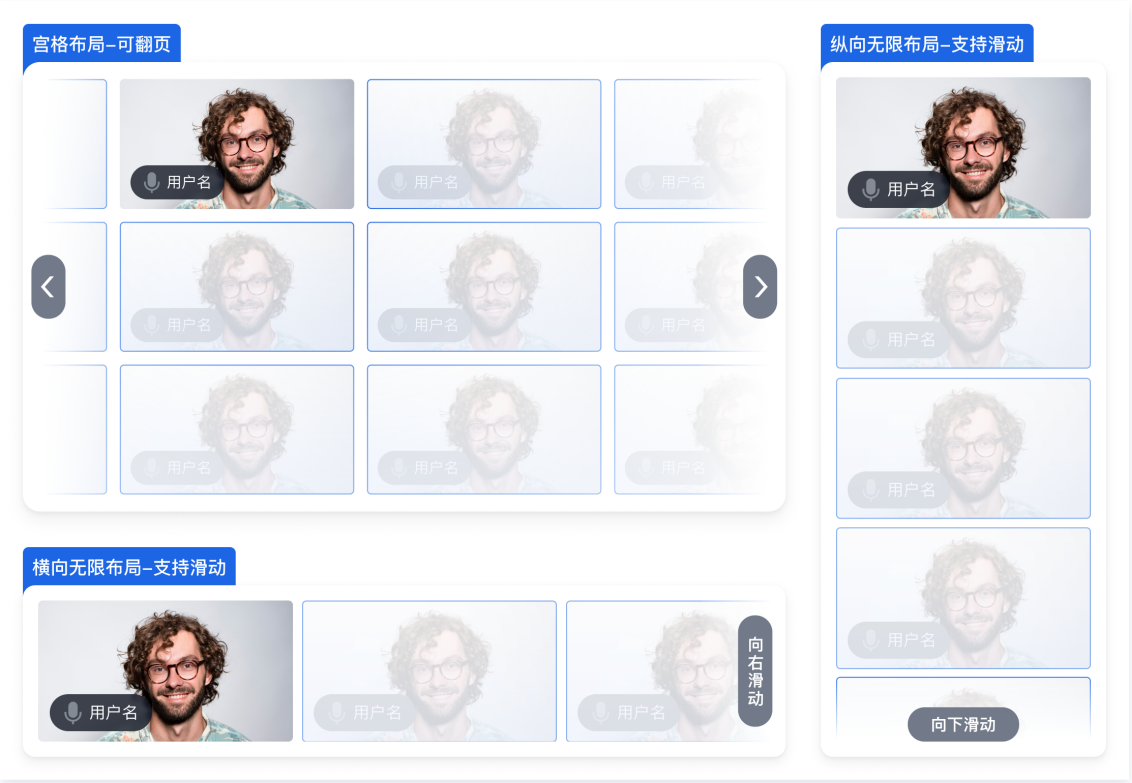
事件名称	说明	回调参数

stream-view-dblclick	当前流播放区域被用户双击	{
		userId: string;
		streamType: TUIVideoStreamType;
		}

MultiStreamView 组件

MultiStreamView 组件默认采用 SingleStreamView 来展示房间内所有用户的流。该组件根据传入的 maxColumn 和 maxRow 参数自动调整流布局。开发者还可以通过 excludeStreamInfoList 参数来指定不在 MultiStreamView 组件中播放的流。

版权所有：腾讯云计算（北京）有限责任公司



组件属性

参数	说明	类型	默认值
maxColume	横向展示的流的最大个数（1，Infinity]	number	-
maxRow	纵向展示的流的最大个数（1，Infinity]	number	-
fillMode	列表中所有流的渲染模式	'fill'/'contain' 'fill'：视频流画面填充整个区域，如果视频流的宽高比与容器不匹配，视频流会被拉伸 'contain'：视频流画面保持原始宽高比播放，如果视频流的宽高比与容器不匹配，视频流被填充黑边	'contain'
excludeStreamInfoList	房间中不需要在当前 MultiStreamView 组件中展示的流	{ userId: string; streamType: TUIVideoStreamType; }[]	[]

组件事件

事件名称	说明	回调参数

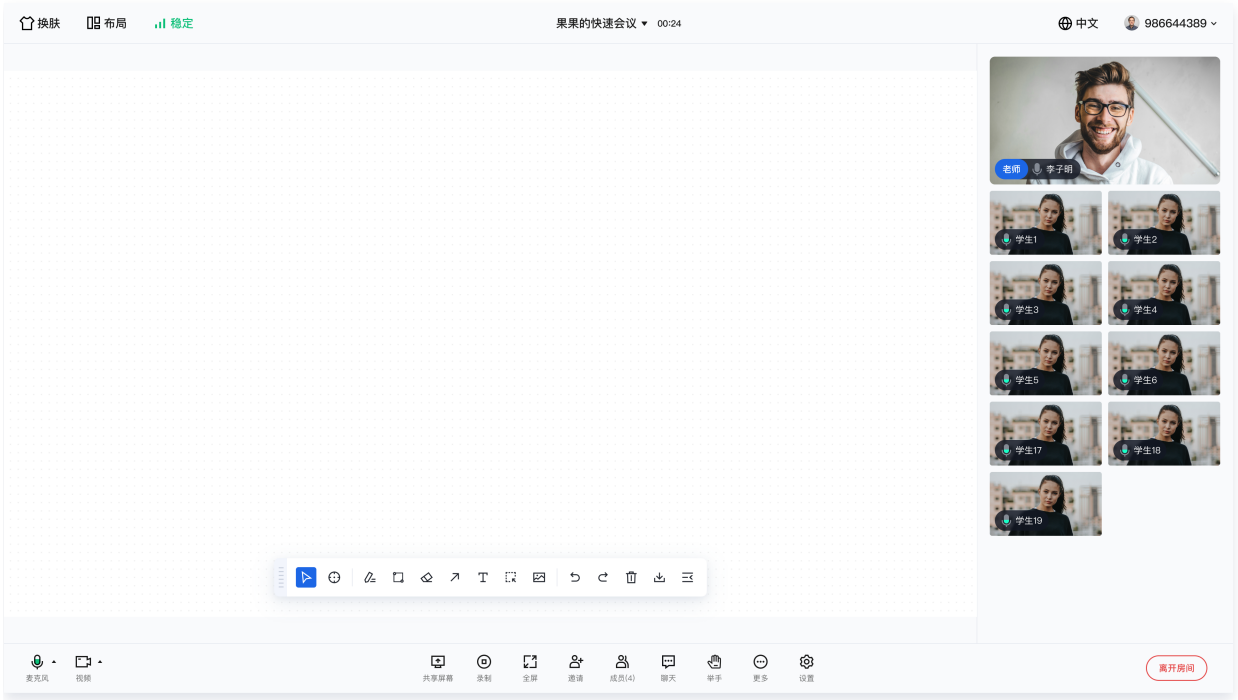
stream-view-dblclick	被用户双击的单个流的信息	{
		userId: string;
		streamType: TUIVideoStreamType;
		}

自定义 Roomkit 流布局

如果 TUIRoomKit 提供的流布局交互不符合您的业务需求，您可以使用 SingleStreamView 组件及 MultiStreamView 组件快速搭建符合您需求的流布局。使用 SingleStreamView 及 MultiStreamView 基础组件，您只需要指定特定区域需要播放的流信息，处理简单的流信息替换，处理 css 布局即可实现复杂的流布局交互。

> **说明：**
> SingleStreamView 组件及 MultiStreamView 组件的详细使用说明请查看 TUIRoomKit 流布局基础组件介绍 。

以经典教育场景布局为例，介绍 SingleStreamView 组件及 MultiStreamView 组件的使用示例。



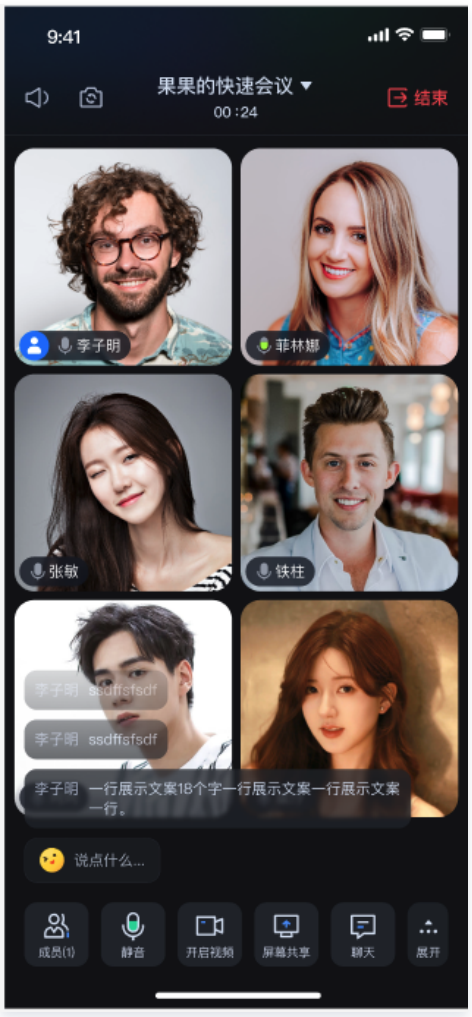

```
<template>
  <div class="stream-container">
    <white-board v-if="isWhiteBoardInEnlarge"></white-board>
    <single-stream-view
      v-else
      class="enlarge-stream-view"
      :streamInfo="masterStream"
    </single-stream-view>
    <div class="side-stream-container">
        <white-board v-if="!isWhiteBoardInEnlarge"></white-board>
        <single-stream-view
          v-else
          class="top-right-stream-view"
          :streamInfo="masterStream"
          @stream-view-dblclick="handleStreamViewDblclick">
        </single-stream-view>
        <multi-stream-view
          class="multi-stream-view"
          :maxColumn="maxColumn"
          :maxRow="maxRow"
          :excludeStreamInfoList="[masterStream]">
        </multi-stream-view>
    </div>
```

```
  </div>
</template>

<script setup>
  import { TUIVideoStreamType } from '@tencentcloud/tuiroom-engine-js';
  const masterStream = ref({
    userId: 'user_master',
    streamType: TUIVideoStreamType.kCameraStream,
  })
  const isWhiteBoardInEnlarge = ref(true);

  const maxColumn = ref(2);
  const maxRow = ref(Infinity);

  function handleStreamViewDblclick(streamInfo) {
    const tempStreamInfo = enlargeStreamInfo;
    enlargeStream.value = topRightStream.value;
    topRightStream.value = tempStreamInfo;
  }
</script>

<style lang="scss">
  div.stream-container {
    width: 100%;
    heght: 100%;
    display: flex;
    .enlarge-stream-view {
      flex: 1;
    }
    .side-stream-container {
      width: 240px;
      height: 100%;
      display: flex;
      .top-right-stream-view {
        width: 240px;
        height: 135px;
      }
      .multi-stream-view {
        flex: 1;
      }
    }
  }
</style>
```

iOS

最近更新时间：2025-01-10 11:18:32

TUIRoomKit 在 iOS 端提供了宫格布局、侧边栏布局、顶部栏布局等多种布局供用户选择。

宫格模式	边栏模式
	

Roomkit 流布局介绍

流布局设计包括两个基本组件：SingleStreamView 和 MultiStreamView。您可以使用这两个组件来组合成您所需的布局。
目前，我们已经支持会议的六宫格布局。如果您需要对已经支持的布局进行微调，可以在 `Source/View/Page/Widget/VideoSeatV2/ConferenceStreamContainer.swift` 文件中调整布局样式及参数。

Roomkit 流布局基础组件

RoomKit 内部提供了 SingleStreamView 和 MultiStreamView 流渲染组件。通过这些组件，开发者可以快速自定义流布局和交互模式。

SingleStreamView 组件

SingleStreamView 播放指定用户的流，在当前用户推流时自动播放流，无流时显示头像信息。

组件属性

参数	说明	类型	默认值
videoItem	当前区域播放的流信息	UserInfo	-

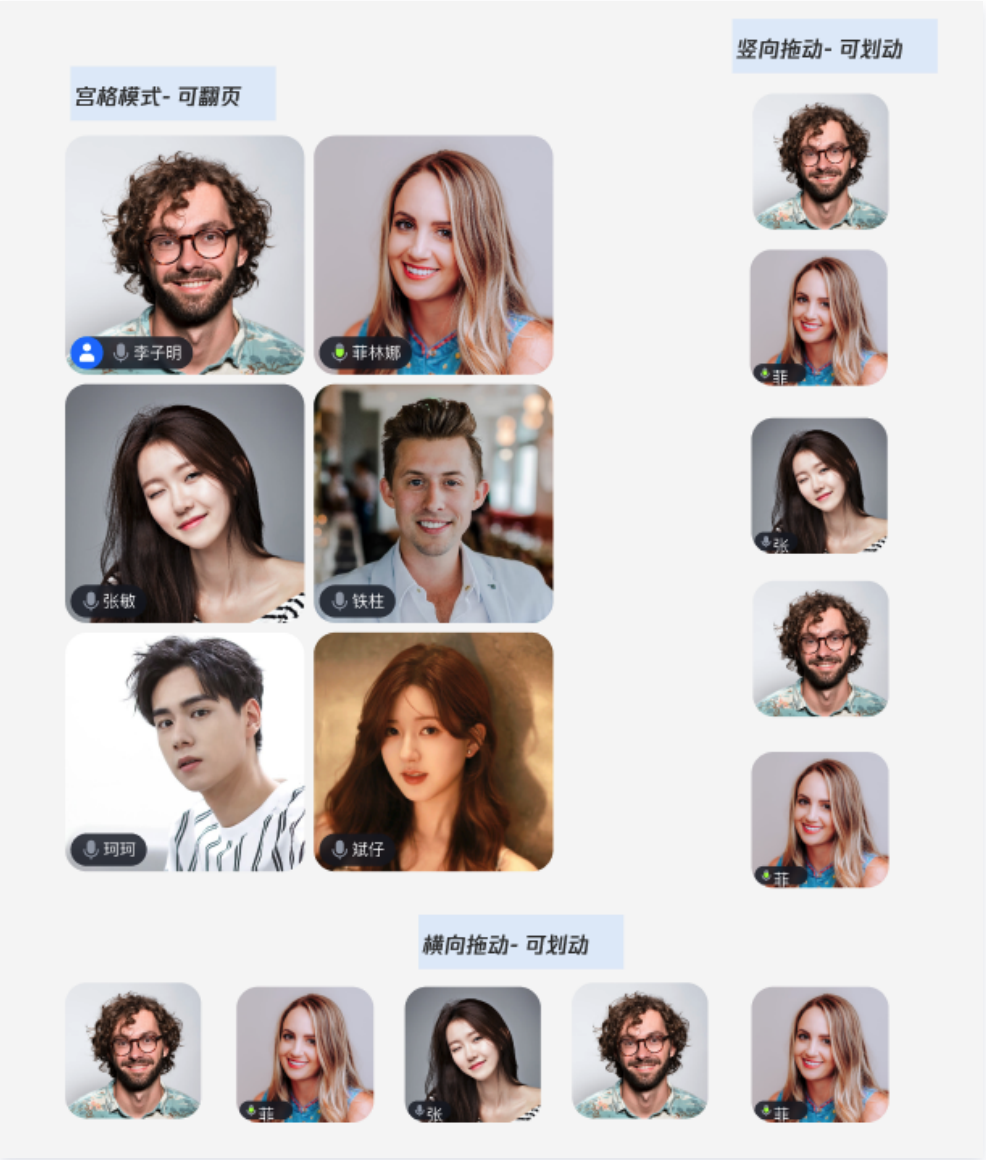
组件事件

事件名称	说明	回调参数
singleViewDidClick	当前流播放区域被用户点击	view: SingleStreamView videoItem: UserInfo

```
let playItem = ... //获取播放的流信息
let singleView = SingleStreamView()

singleView.updateVideoItem(item: playItem) //设置播放的流信息
```

MultiStreamView 组件

MultiStreamView 组件默认显示房间内所有用户的流。该组件根据传入的 maxColumn 和 maxRow 参数自动调整流布局。开发者还可以通过 excludeVideoItems 参数来排除在 MultiStreamView 组件中播放的特定流。



组件属性

参数	说明	类型	默认值
maxColume	横向展示的流的最大个数	number	-
maxRow	纵向展示的流的最大个数	number	-
excludeVideoItems	房间中不需要在当前 MultiStreamView 组件中展示的流	[UserInfo]	[]

```
// 3行2列宫格模式
let multiStreamView = MultiStreamView(maxRows: 3, maxColumns: 2)
// 1行上边栏模式
let multiStreamView = MultiStreamView(maxRows: 1, maxColumns: 0)
// 1列侧边栏模式
let multiStreamView = MultiStreamView(maxRows: 0, maxColumns: 1)


let excludeItem = ... //获取排除的流信息
```

```
multiStreamView.setExcludeVideoItems(items: [excludeItem])
```

自定义 Roomkit 流布局

如果 TUIRoomKit 提供的流布局交互不符合您的业务需求，您可以使用 SingleStreamView 组件和 MultiStreamView 组件快速搭建符合您需求的流布局。通过使用 SingleStreamView 和 MultiStreamView 基础组件，您只需根据您的业务需求设置流信息和布局组件即可。
以传统教育场景布局为例，下文展示的为 SingleStreamView 组件和 MultiStreamView 组件的使用示例。

```
class streamContainer: UIView {
    private let studentsView: MultiStreamView = {
        let studentsView = MultiStreamView(maxRows: 0, maxColumns: 1)
        return studentsView
    }()

    private let teacherView: UIView = {
        let teacherView = SingleStreamView()
         return teacherView
    }()

    override init(frame: CGRect) {
        super.init(frame: frame)
        setupUI()
    }

    required init?(coder: NSCoder) {
        super.init(coder: coder)
        setupUI()
    }

    private func setupUI() {
        self.addSubview(teacherView)
        self.addSubview(studentsView)

        teacherView.snp.makeConstraints { make in
            make.leading.top.bottom.equalToSuperview() //上左下对齐
            make.width.height.equalTo(600) // 设置宽
        }
```

```
            studentsView.snp.makeConstraints { make in
                make.top.bottom.trailing.equalToSuperview() // 上下右对齐
                make.width.equalTo(120)            // 设置固定宽度
            }
        }
    }
```

白板功能（TUIRoomKit）
Electron

最近更新时间：2024-10-09 10:18:01

`TUIRoomKit` 推出了白板功能，以丰富您的会议内容，提高您的会议体验。本文将详细介绍这一功能，帮助您更好地掌握 `TUIRoomKit` 中白板的相关功能的接入和使用。

功能介绍

白板功能主要分为两种：一种是单独开启白板，并将白板分享，使会议中的其他成员可以观看到白板内容；另一种则是开启屏幕分享后，在屏幕上进行批注，适用于需要在屏幕，文件，图片上进行批注的场景。
白板的基本功能包括自由画线、绘图、添加文字、线段、箭头、橡皮擦、撤销和重绘等。下面将对这些功能进行逐一介绍。

> **注意：**
> 目前仅 Electron 端支持白板功能。

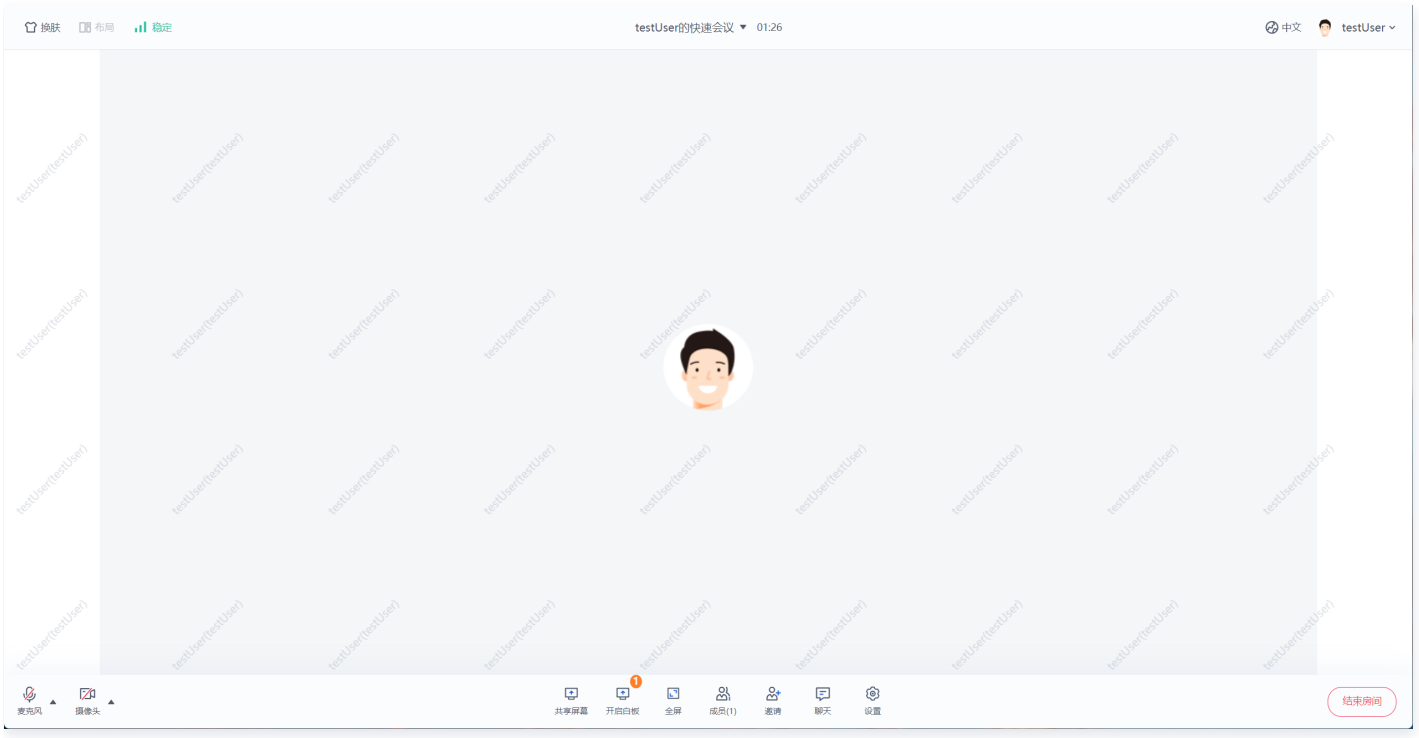
使用说明

白板功能

开启白板

> **注意：**
> 开启白板的功能与共享屏幕功能不能同时使用，且远端有成员在使用白板或共享屏幕时，也无法再开启白板。

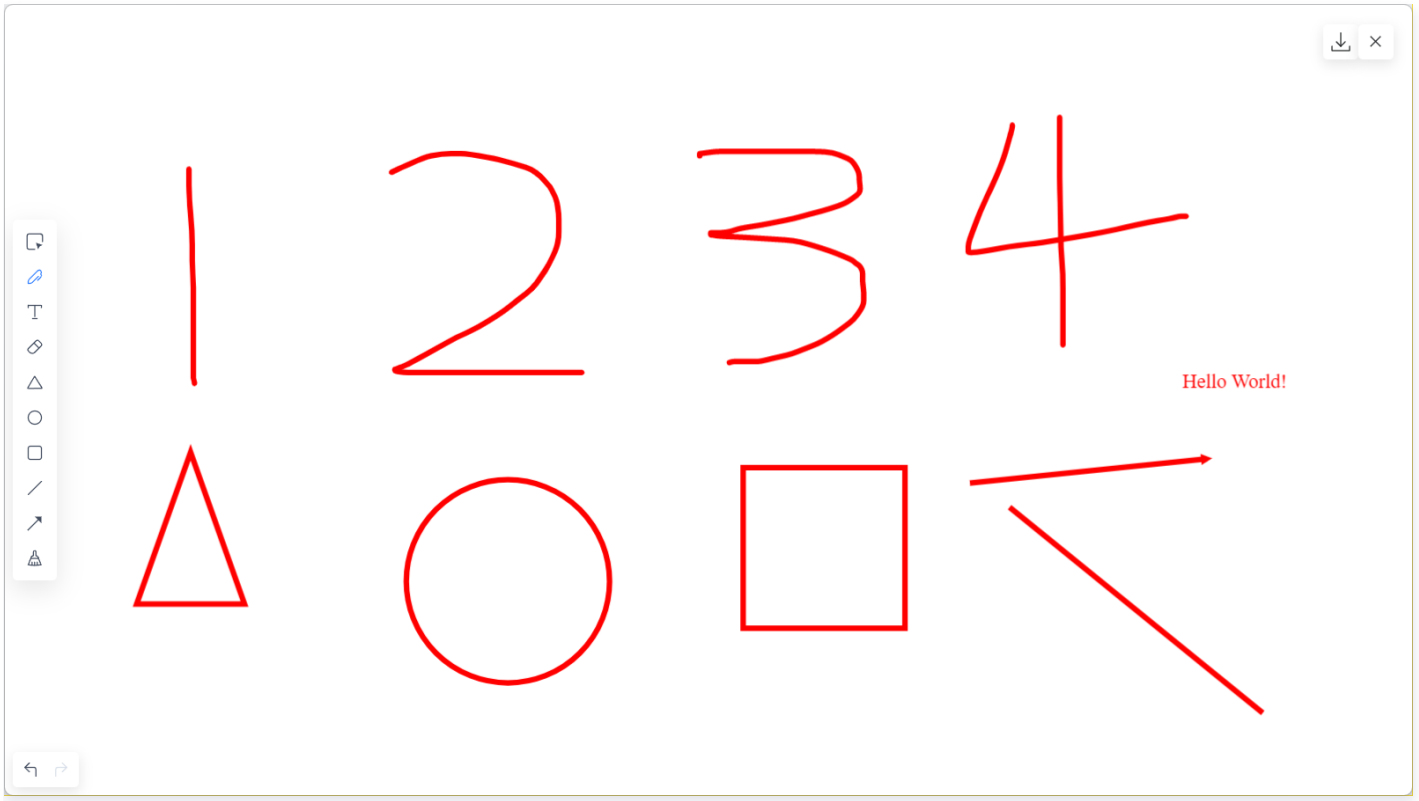
在会议界面底部菜单栏中， 您可以点击底部工具栏的**开启白板**按钮。点击它，会弹出一个白板的窗口，并将窗口分享出去，会议中的其他成员将会看到白板内容。



白板内容绘制

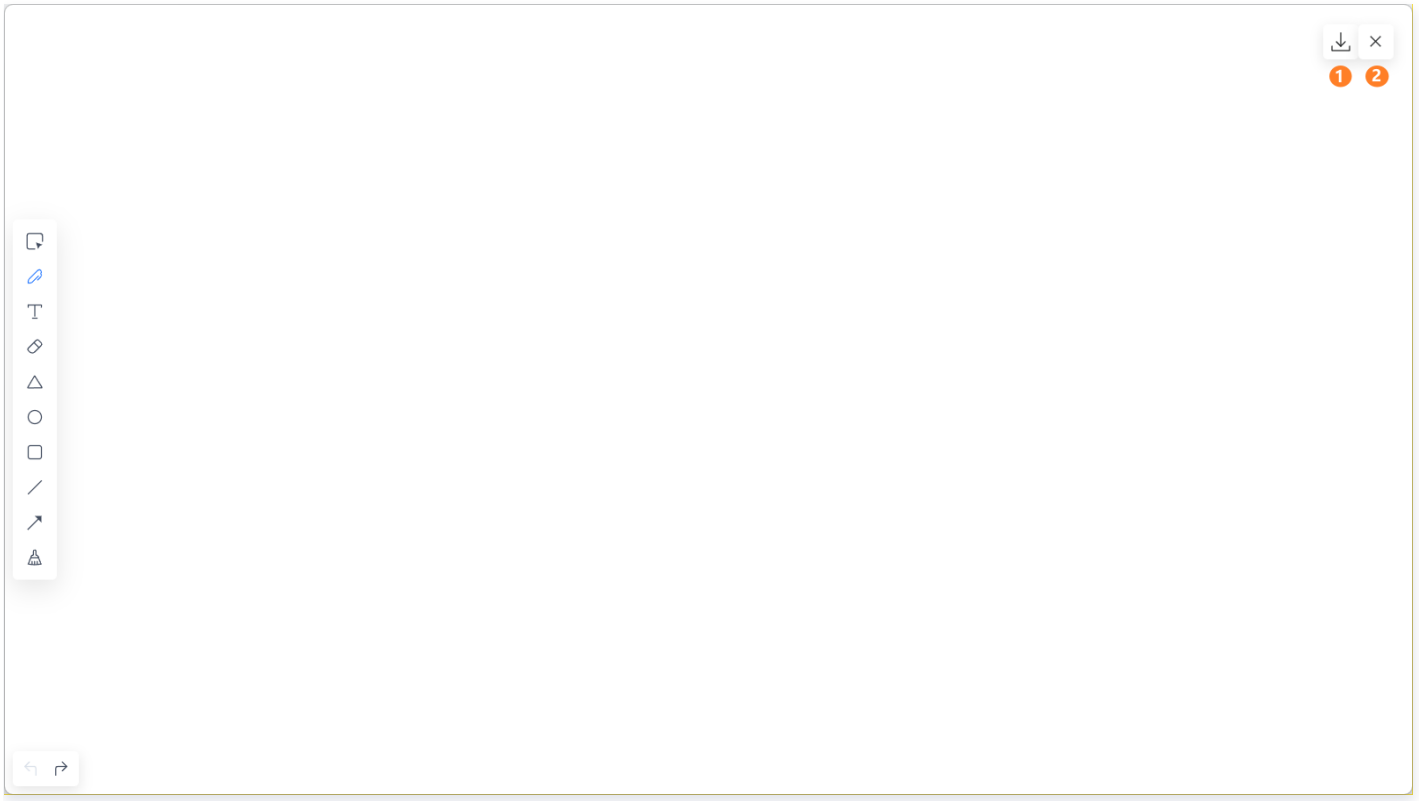
- 开启白板后，您可以点击左侧工具栏来更换工具，您可以使用画笔来进行自由绘制，可以使用图形工具来绘制图形，箭头工具可以用来绘制箭头，

  橡皮擦工具可以擦除绘制的图形，清空工具可以清空当前画板。
- 左下角工具栏为撤销/恢复操作，您可以根据需要进行撤销和恢复操作。



保存白板

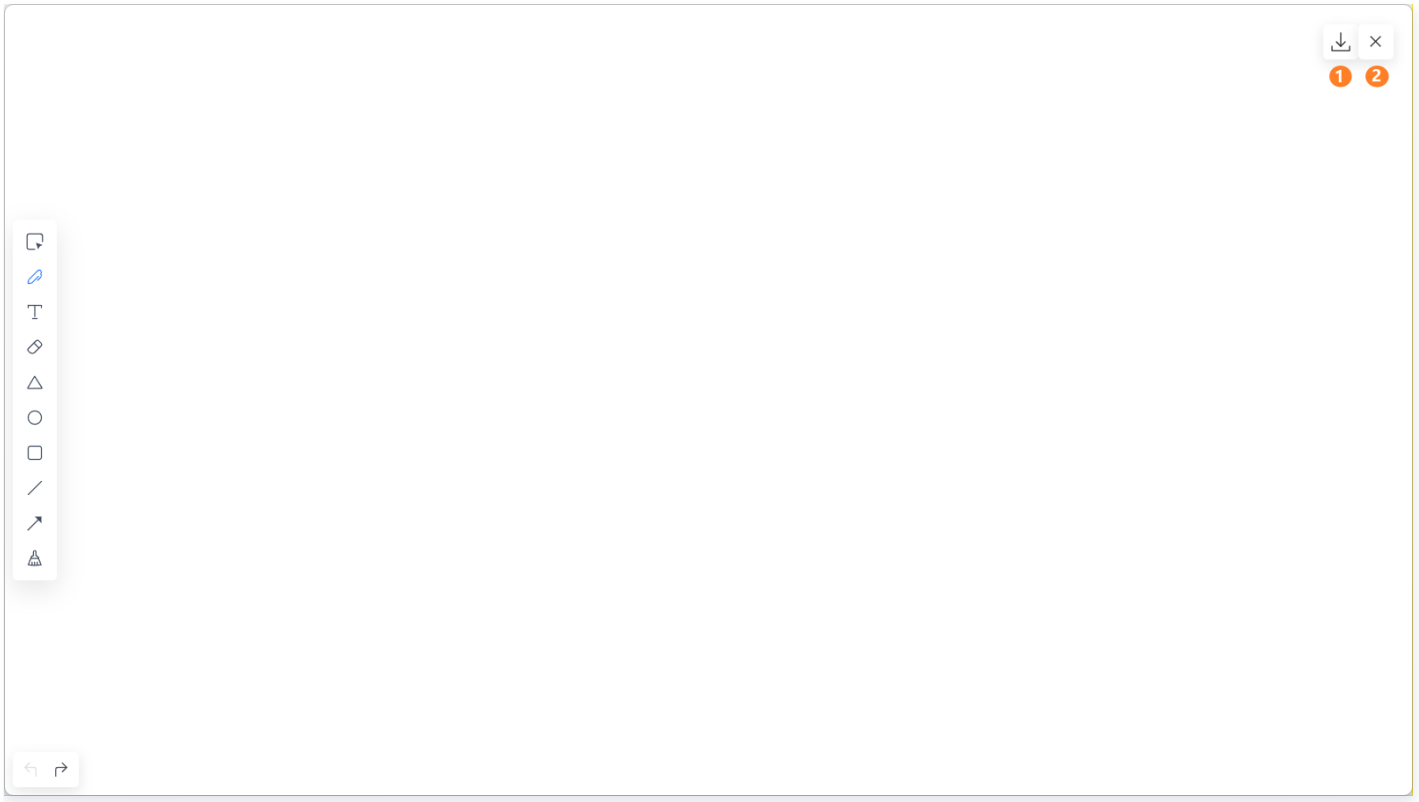
您可以通过单击 ，来将白板内容保存为图片。



关闭白板

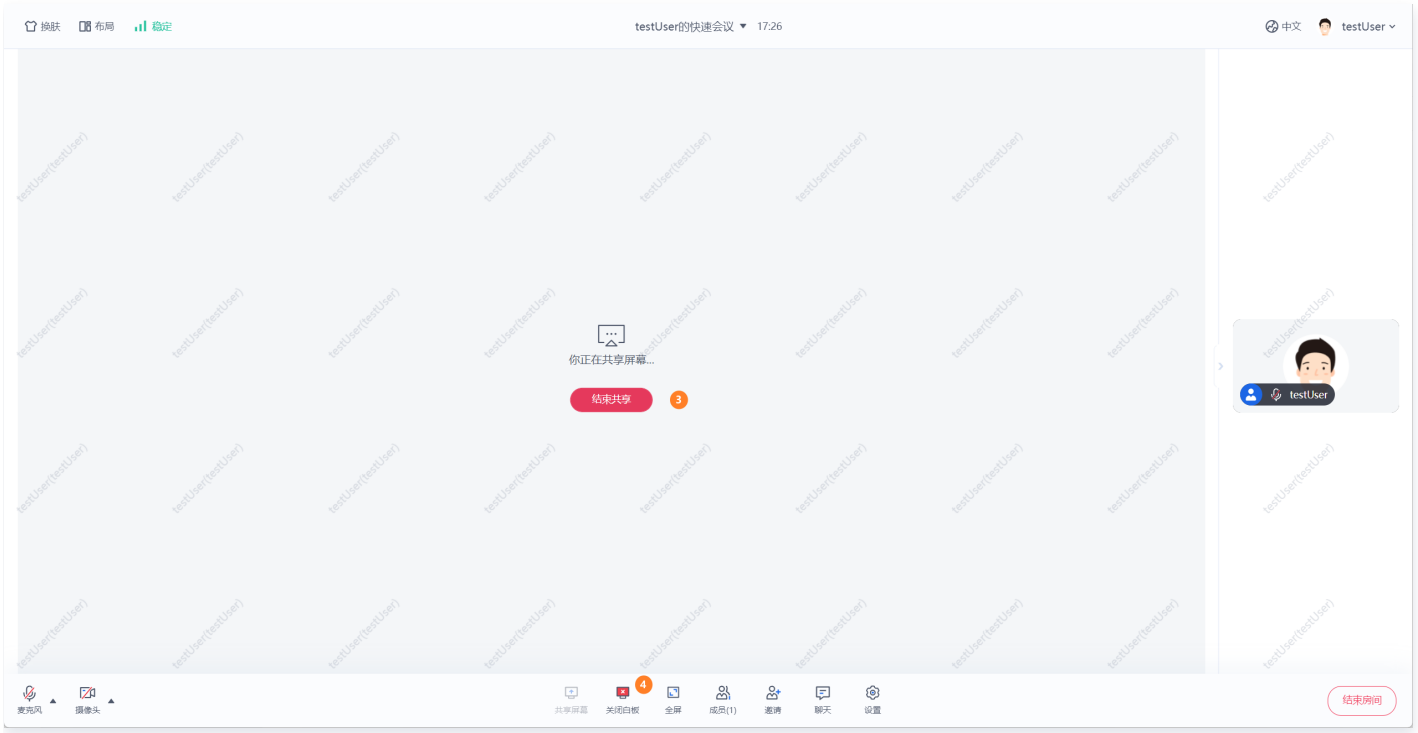
方式一：

您可以通过单击 × 结束共享白板。



方式二：

单击 结束共享 或点击页面上底部工具栏的**关闭白板**按钮。

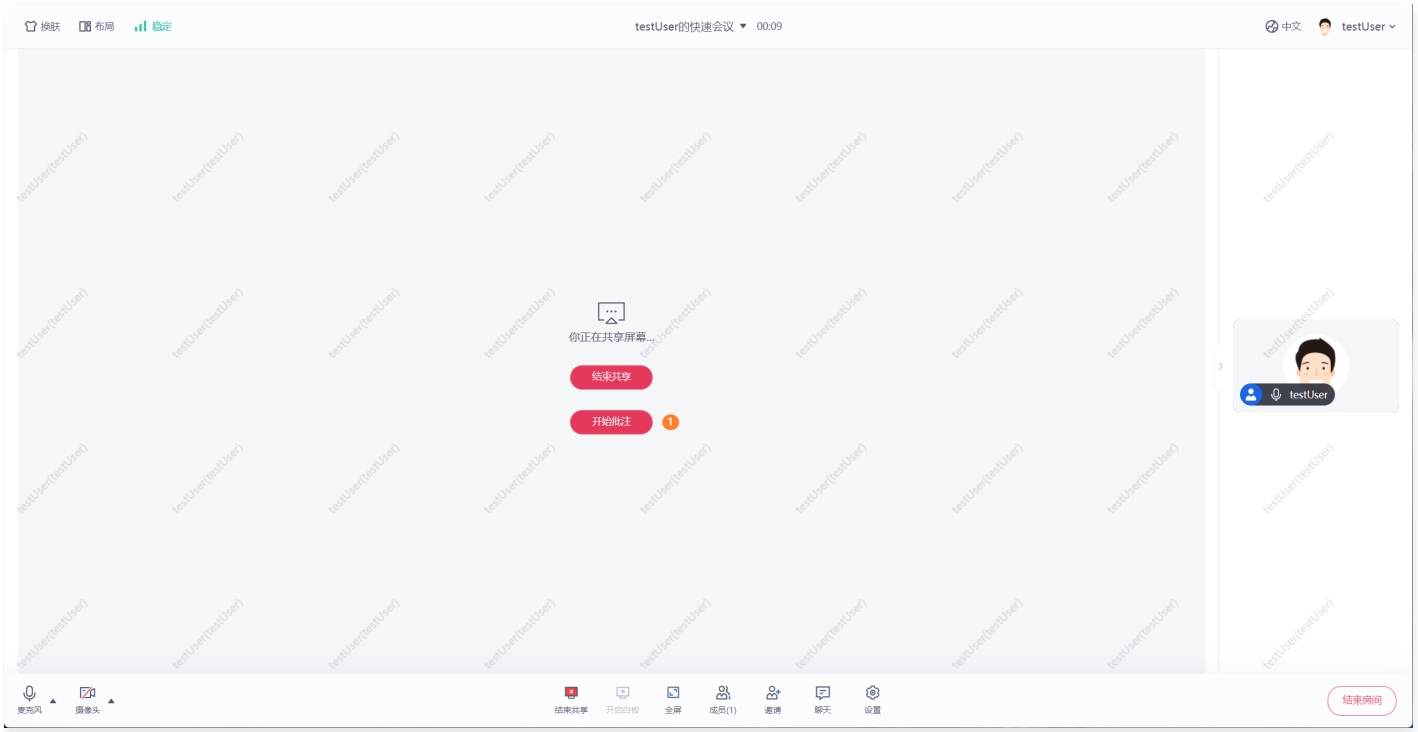


批注功能

> **注意：**
> **目前仅支持在共享屏幕时进行批注，共享窗口暂不支持。**

开始批注

开始屏幕分享后，您可以点击**开始批注**按钮，进行批注。批注时，点击其他窗口将无效。您可以通过点击底部工具栏来切换窗口或文档。



结束批注

您可以点击批注界面右上角的**结束批注**按钮或房间页面上的**结束批注**按钮结束批注。

快速接入

以下接入方式以 RoomKit 快速接入文档 中使用的 electron-vite-vue 工程为例。

步骤一：引入 ConferenceMainView 组件

在 packages/renderer/src 目录下创建 views 文件夹，并在 views 目录下建一个 room.vue 文件，从 roomKit 库中引入 ConferenceMainView 组件。

```
<template>
  <ConferenceMainView />
</template>

<script setup lang="ts">
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue3';
</script>
```

步骤二：引入 WhiteboardView 组件

在 views 目录下建一个 whiteboard.vue 文件，从 roomKit 库中引入 WhiteboardView 组件。

```
<template>
  <WhiteboardView />
</template>

<script setup lang="ts">
import { WhiteboardView } from '@tencentcloud/roomkit-electron-vue3';
</script>
```

步骤三：增加路由配置

1. 添加 router.ts 文件

在项目路径下运行 npm install vue-router 命令安装依赖
在 packages/renderer/src 目录下创建 router.ts 文件，增加路由配置。

```
import { createRouter, createWebHashHistory } from 'vue-router';

import Room from './views/room.vue';
import Whiteboard from './views/whiteboard.vue';

const routes = [
  {
    path: '/',
    redirect: '/room',
  },
  {
    path: '/room',
    component: Room,
  },
```

```
  {
    path: '/whiteboard',
    component: Whiteboard,
  },
];

const router = createRouter({
  history: createWebHashHistory(),
  routes,
});

export default router;
```

2. 修改 main.ts 文件

修改 **packages/renderer/src/main.ts** 文件， 引入 **router** 文件。

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'; // 引入 router 依赖
import { createPinia } from 'pinia';

const app = createApp(App);
app.use(createPinia());
app.use(router);  // 引入路由配置
app.mount('#app').$nextTick(window.removeLoading);
```

3. 修改 App.vue 文件

修改 **packages/renderer/src/App.vue** 文件，在UI上对应位置设置路由。

```
<template>
  <router-view></router-view>
</template>
```

步骤四：增加白板窗口和批注窗口

1. 拷贝 whiteboard.ts 和 annotation.ts 文件

在 **packages/main** 目录下创建 **whiteboard.ts** 和 **annotation.ts** 文件，您可以直接拷贝 github RoomKit 仓库中的 whiteboard.ts 和 annotation.ts 文件到工程中。

2. 修改 index.ts 文件

修改 **packages/main/index.ts** 文件引入白板和批注功能，首先引入白板和批注初始化的方法 **initWhiteboardWindow** 和 **initAnnotationWindow**，然后修改 **createWindow** 方法，在主窗口 **win** 创建完成后，调用 **initWhiteboardWindow** 和 **initAnnotationWindow** 方法初始化白板和批注窗口，即可在工程中集成白板和批注功能。

```
import { initWhiteboardWindow } from './whiteboard';
import { initAnnotationWindow } from './annotation';

async function createWindow() {
  win = new BrowserWindow({
```

```
    title: 'Main window',
    webPreferences: {
      preload: join(__dirname, '../preload/index.cjs'),
      nodeIntegration: true,
      contextIsolation: false,
    },
  });

  ......

  // 在 createWindow 方法最后新增初始化白板和批注窗口
  initWhiteboardWindow(win);
  initAnnotationWindow(win);
}
```

云端录制 (TUIRoomKit)

最近更新时间：2024-03-22 15:46:32

本文将介绍如何快速使用 TUIRoomKit 云端录制功能，帮助开发者在视频会议、在线教育、直播互动等场景中，实现对重要内容进行存档、审核等多样化需求。我们提供了两个方案供开发者选择：自动录制方案 和 REST API 录制方案 。

> **说明：**
> TUIRoomKit 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。

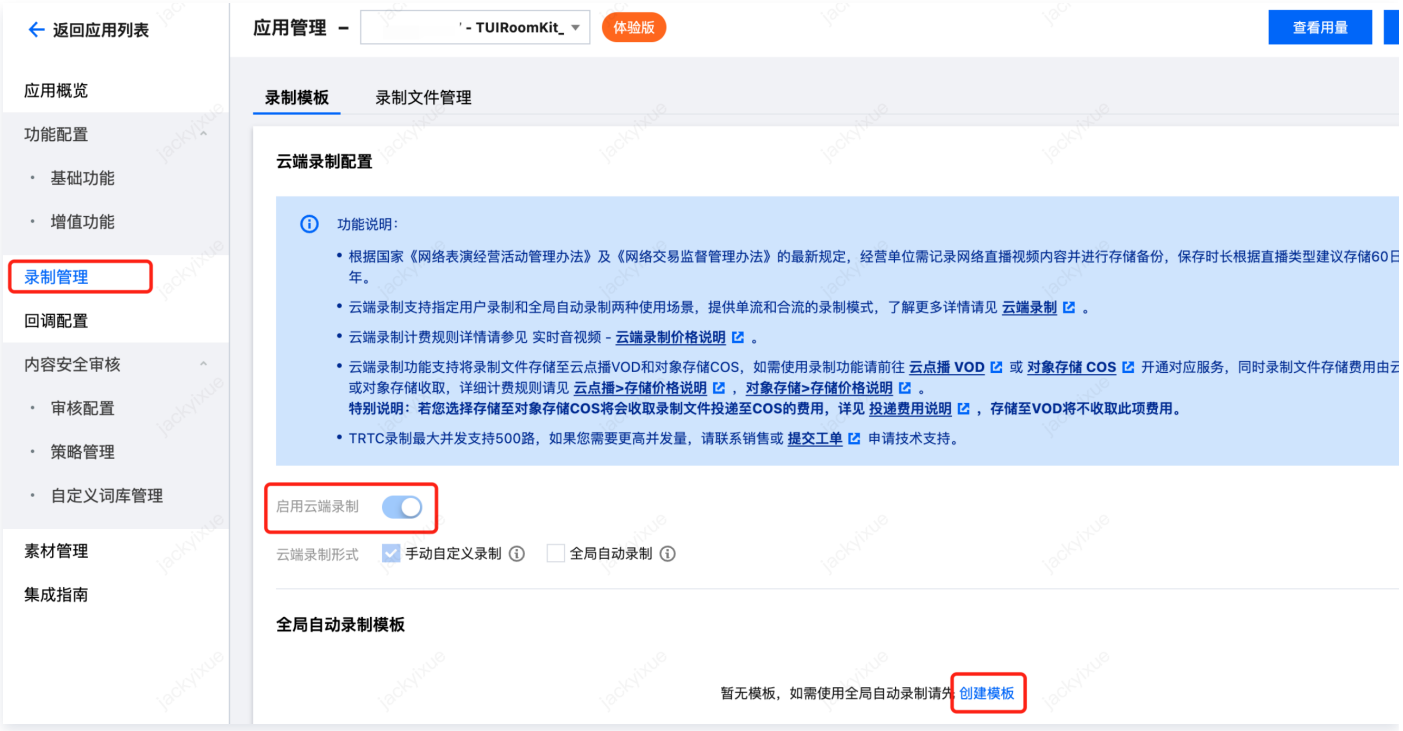
方案一：自动录制方案（推荐）

自动录制方案的优点是录制不需客户手动启动和停止，由 TRTC 后台**自动管理录制任务**：若检测到房间中有上行音视频后将触发**启动录制任务**，若房间内主播都退房且超过设置的等待续录时间后将触发**停止录制任务**。您可以通过如下几步快速完成配置：

1. 在 实时音视频控制台 > 应用管理 找到对应 SDKAppId 的应用，选择**操作 > 配置**，详见下图：

2. 点击左侧导航栏的**录制管理**，点击**开通录制服务**。

3. **开通录制服务后**，在配置页中打开**启用云端录制**功能，然后在**全局自动录制模板**中点击**创建模板**。



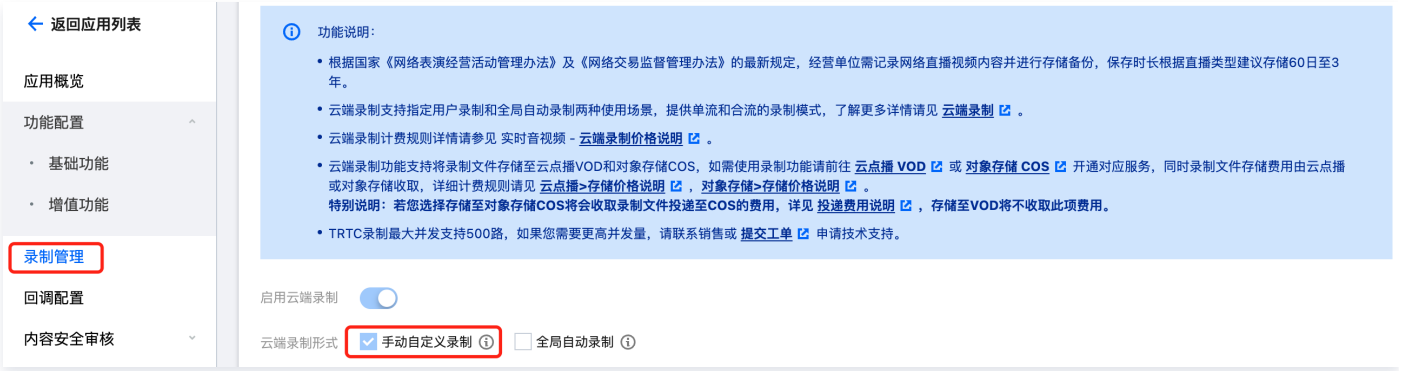
4. 配置**全局自动录制模板**可参考以下参数进行配置，您也可以根据自己的业务需求配置自定义录制模板，具体配置项说明参见：全局自动录制配置项说明 。

> **说明：**
> - 全局自动录制仅支持 单流录制（即每个主播单个录制一个文件），开启后只对新创建的房间有效，对开启自动录制功能之前已经创建的房间不生效。如需录制多路流混合后的画面，请使用REST API 录制方案 。
> - 全局自动录制支持最多录制一个房间内的25个主播，如果超过25个主播将会按照进房时间由先到后排序，录制前25位主播（如需单流录制超过25位主播，请参见 REST API 录制方案 ）。

方案二：REST API 录制方案

如果您在业务上需要指定录制订阅房间内的主播、自定义合流布局、录制中途更新布局等自定义业务需求。您可以使用API 录制方案，相比自动录制方案会更加灵活、功能完备。

1. **在 实时音视频控制台 > 应用管理 找到对应 SDKAppId 的应用，选择操作 > 配置。**
2. **点击左侧导航栏的录制管理，点击开通录制服务，此时默认勾选手动自定义录制，即 REST API 模式。**



3. **点击左侧导航栏的回调配置，可通过配置并监听服务端的 房间与媒体回调 事件，在您的后台服务调用 REST API（ CreateCloudRecording ）来控制启动云端录制的时机。详细操作说明请参考： API 手动录制文档 。**

> **说明：**

手动录制下，您可以前往控制台配置回调地址，以接受录制回调事件，请见 录制回调说明 。

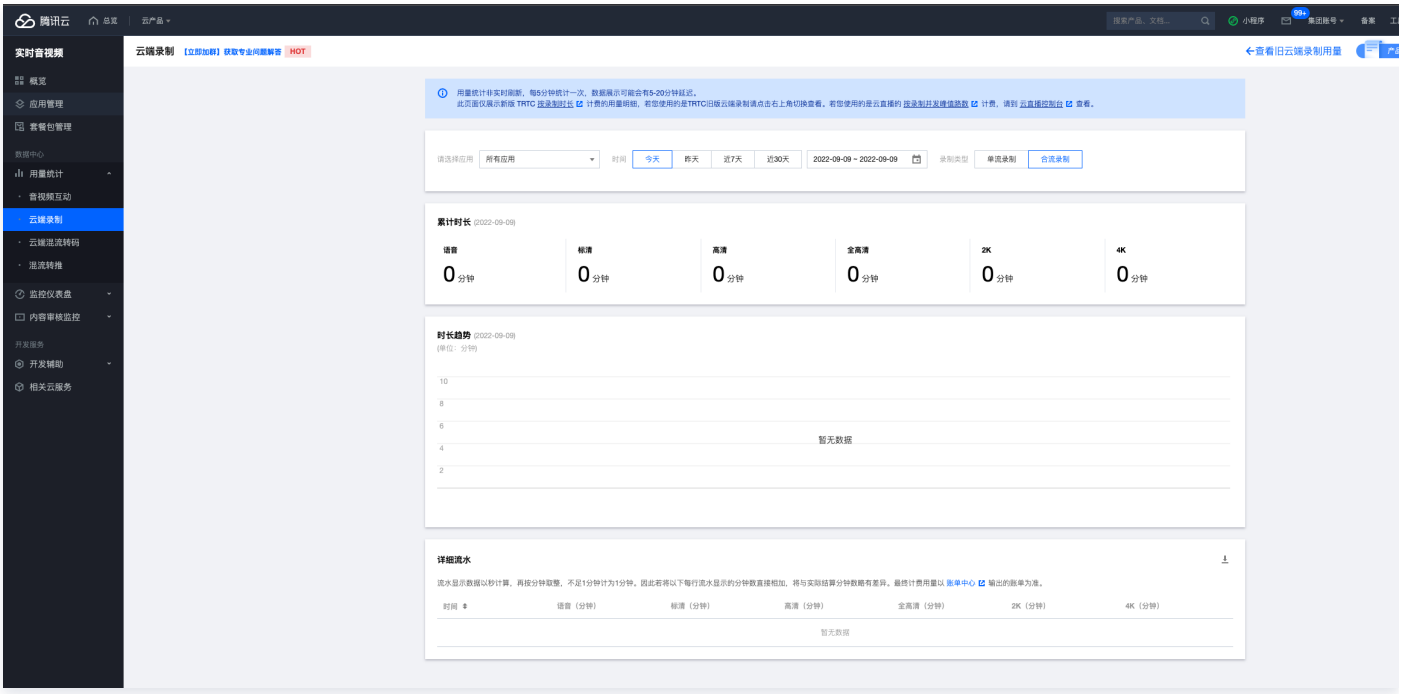
常见问题

1. 两种录制方案可以同时使用吗?

自动录制方案 和 REST API 录制方案 这两种方案并不冲突，可以同时使用两种录制方案，但会产生两份录制文件和 费用 。

2. 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：



3. 如何管理录制文件?

目前提供的 录制文件管理 主要有以下操作：
查找录制文件 、接收录制文件 、删除录制文件 、回放录制文件 。

4. 如何自定义录制文件名?

目前暂不支持重命名录制文件名，仅支持使用 API 录制 自定义设置文件名称前缀：使用 API 录制 存储至云点播 VOD 时，可通过 TencentVod 中的 UserDefineRecordId 参数自定义文件名称前缀，前缀与自动生成的录制文件名之间用 `_UserDefine_u_` 分开。

更多文档

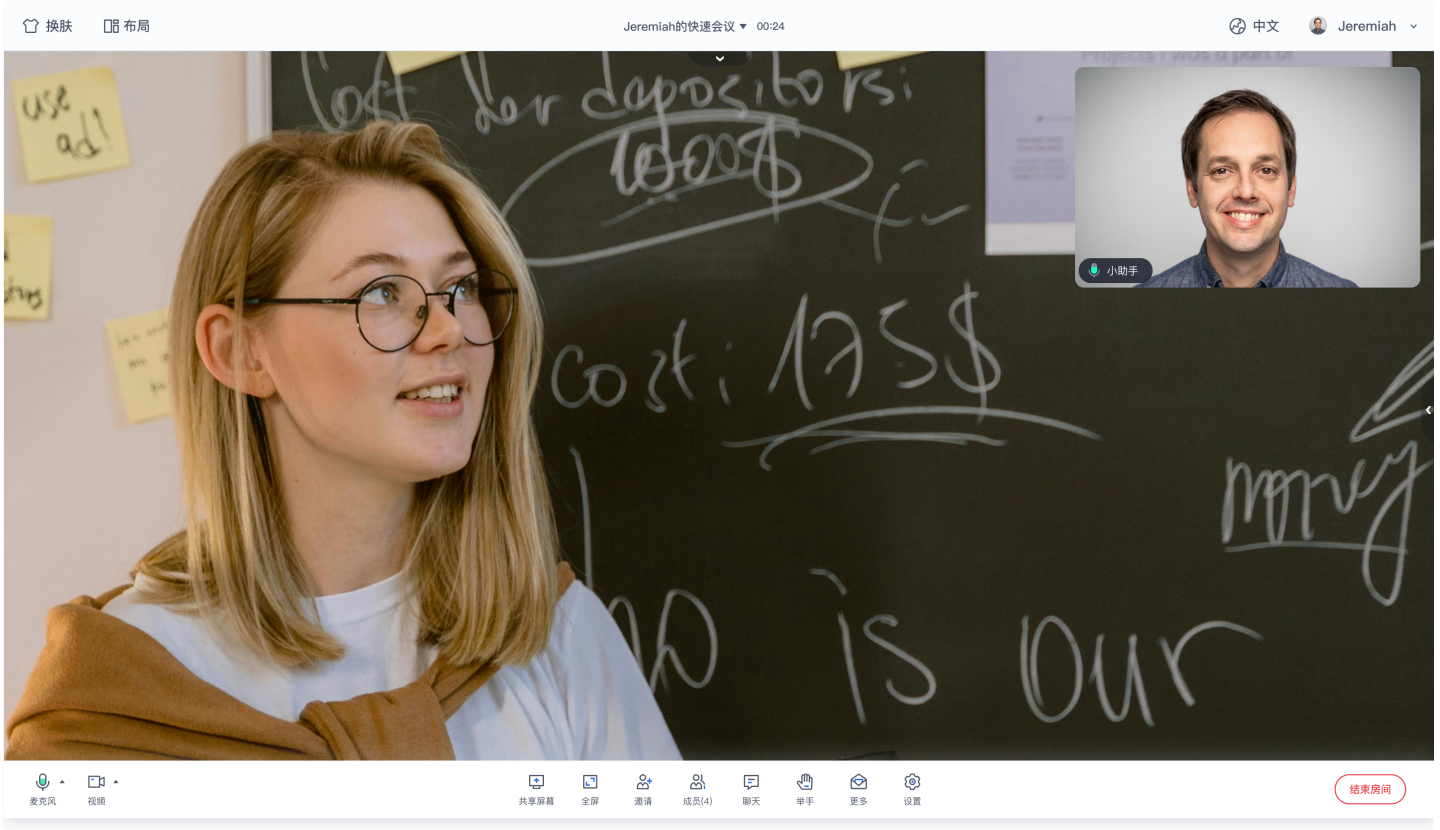
- 实现云端录制与回放
- 云端录制服务端事件回调
- 云端录制计费说明

机器人推流（TUIRoomKit）

最近更新时间：2024-08-16 17:53:01

功能介绍

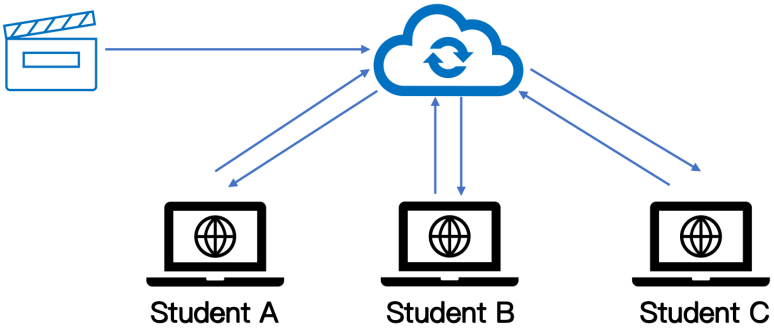
TUIRoomKit 支持使用机器人推流，允许用户将本地或在线媒体文件推流至房间内。在某些场景中，会议中可能需要推送预先录制的教学视频或在线视频流到在线课堂中，以便用户观看和学习。TUIRoomKit 可以满足相关需求，实现自动化和高效的视频播放。
使用本功能，需开通 TUIRoomKit 包月套餐 。接入的机器人推流会作为房间中的虚拟用户产生正常的通话费用，一条流是一个虚拟用户，详情参见 音视频时长计费说明；同时，接入机器人推流过程中会进行转码操作，从而产生转码费用，详情参见 云端混流转码计费说明 。



应用场景

- 一起看、一起听、一起玩、一起学等原来需要线下面对面才能实现的各种体验正被不断搬到线上。相隔千里还能和好友们一起看电影、一起听音乐，然后一起交流吐槽，这样神奇的实时互动体验正受到当下年轻人的喜爱，并成为如今音视频产品的重点玩法和主流方向。
- TUIRoomKit 输入媒体流功能可以将外部媒体流共享至房间内，用户可通过这一功能向房间内其他用户分享音乐、电影、讲座、课程等媒体内容并同时与他们进行实时互动。平台可借助 TUIRoomKit 的这一新功能快速实现**一起看**场景。除了一起看电影、听音乐等场景外，TUIRoomKit 输入媒体流能力还能够在互动课堂、体育赛制、网络会议等场景中，为平台带来更多创新可能。

版权所有：腾讯云计算（北京）有限责任公司



使用方法

方法一：Rest API 输入在线媒体流

功能描述

输入在线媒体流功能说明如下：

类型	描述
发起任务方式	用户可以通过 REST API 发起输入在线媒体流任务，观众可观看这路流，支持录制、转推等功能。
多种源流协议和格式	协议：HTTP、HTTPS、RTMP、HLS 格式：FLV、MP3、MP4、MPEG-TS、MOV、MKV、M4A 视频编码：H.264、VP8 音频编码：AAC、OPUS
服务端回调	输入在线媒体流任务创建和结束时可回调给业务侧服务器，用于业务侧做逻辑，详细输入在线媒体流事件，前往查看。

相关 Rest API

- 开启输入在线媒体流：StartStreamIngest
- 停止输入在线媒体流：StopStreamIngest
- 查询输入在线媒体流：DescribeStreamIngest

方法二：使用 RTMP 推流

TUIRoomKit 支持 RTMP 标准协议推流。您可根据实际情况选择安装 OBS 、FFmpeg 或其他 RTMP 库进行推流。

使用 FFmpeg 推流

FFmpeg 不同场景下指令配置参数不同，因此需要您具有一定的 FFmpeg 使用经验，以下列出 FFmpeg 常用命令行选项，更多 FFmpeg 选项请参见 FFmpeg 官网 。

FFmpeg 命令行

```
ffmpeg [global_options] {[input_file_options] -i input_url} ... {[output_file_options] output_url}
```

常见的 FFmpeg 选项

选项	说明
-re	以 native 帧率读取输入，通常只用于读取本地文件

其中 output_file_options 可配置选项包括：

选项	说明
-c:v	视频编码，建议用 libx264
-b:v	视频码率，例如 1500k 表示 1500kbps
-r	视频帧率
-profile:v	视频 profile，指定 baseline 将不编码 B 帧，TUIRoomKit 后端不支持 B 帧
-g	GOP 帧数间隔
-c:a	音频编码，建议用 libfdk_aac
-ac	声道数，填2或1
-b:a	音频码率
-f	指定格式，固定填 `flv` ，发送到 TUIRoomKit 使用 FLV 容器封装

使用

下面的例子是 使用 FFmpeg 命令读取文件推到 TUIRoomKit，注意 URL 两边加引号。

```
ffmpeg -loglevel debug -re -i sample.flv -c:v libx264 -preset ultrafast -profile:v baseline -g 30 -sc_threshold 0 -b:v 1500k -c:a libfdk_aac -ac 2 -b:a 128k -f flv 'rtmp://rtmp.rtc.qq.com/push/yourRoomId?userid=yourUserId&sdkappid=xxxxxxxxx&usersig=xxxxxxxxxx'
```

对上述命令中相关参数的解释如下：

参数	含义
-i sample.flv	需要推流至 TUIRoomKit 的媒体文件。您可以将 `sample.flv` 更换为您需要推流的本地或在线媒体文件。
yourRoomId	您需要推流的房间 ID。您需要将 yourRoomId 更换为您实际的 RoomId。
userId	您需要推流的 UserID。您需要将“=”后面的值更改为实际的 userId。

sdkappid	您的 sdkappid，您之前已在 开通服务 中获得。您需要将“=”后面的值更改为实际的 sdkAppID。
usersig	您的 usersig。您已在登录 TUIRoomKit 组件时获得。您需要将“=”后面的值更改为实际的 userSig。

当您需要在代码中实现上述功能时，可以使用您所在平台上的相关 FFmpeg 库或通过命令行调用 FFmpeg 来实现上述命令。

使用 OBS 推流

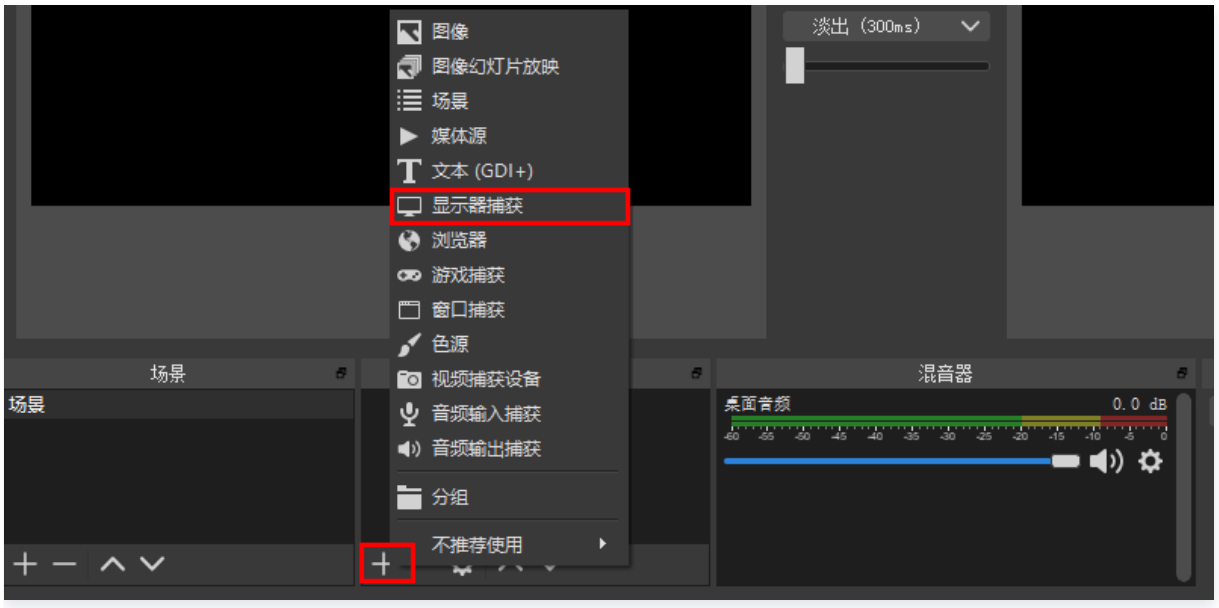
准备工作

安装并打开 OBS 工具进行下述操作。

步骤1：选择输入源

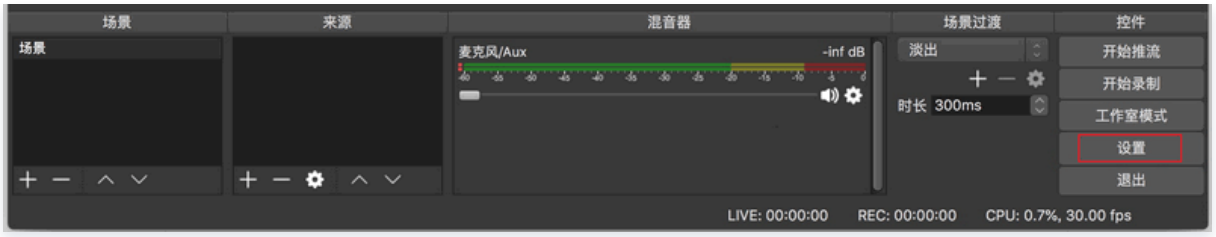
查看底部工具栏的**来源**标签，单击 + ，根据您的业务需要选择输入源。常用来源输入有：

输入源	说明
图像	适用于单张图像直播
图像幻灯片放映	可循环或者顺序多张播放图片
场景	实现各种强大的直播效果。此时，另一个场景是作为来源被添加进当前场景的，可以实现整个场景的插入
媒体源	可上传本地视频，并本地点播视频文件进行直播化处理
文本	实时添加文字在直播窗口中
窗口捕获	可根据您选择的窗口进行实时捕获，直播仅显示您当前窗口内容，其他窗口不会进行直播捕获
视频捕获设备	实时动态捕捉摄像设备，可将摄像后的画面进行直播
音频输入捕获	用于音频直播活动（音频输入设备）
音频输出捕获	用于音频直播活动（音频输出设备）



步骤2：设置推流参数

1. 通过底部工具栏的**控件 > 设置**按钮进入设置界面。

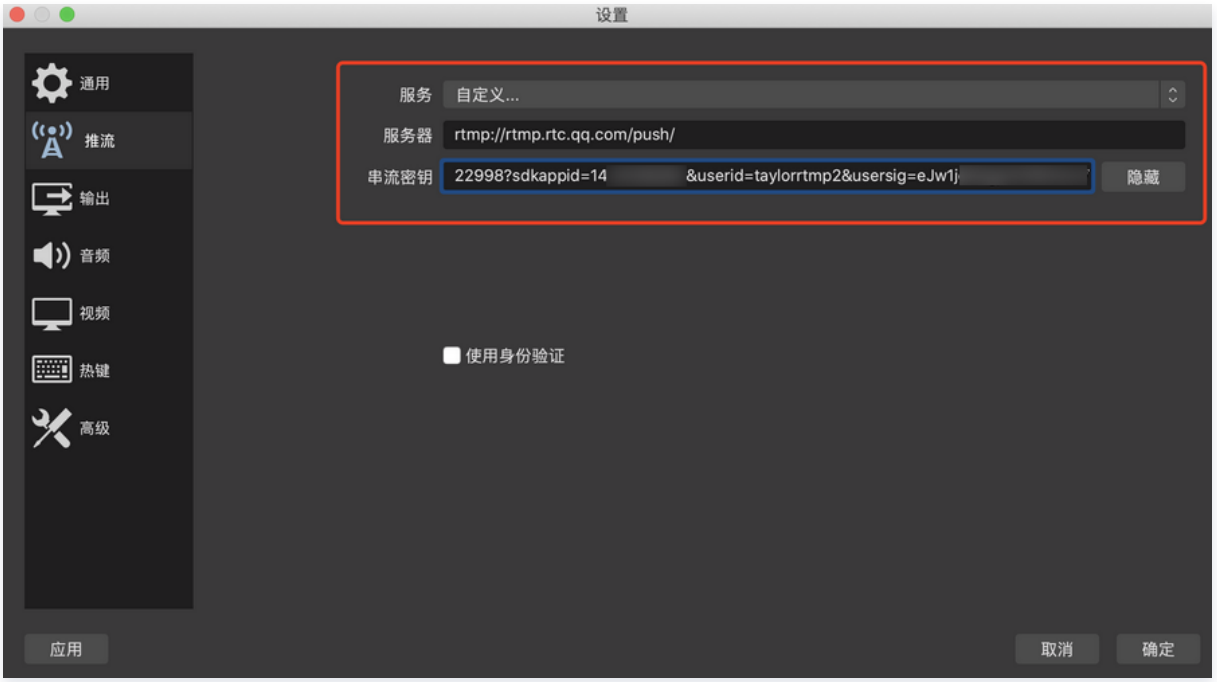


2. 单击**推流**进入推流设置页签，选择服务类型为**自定义**。
3. 服务器填写：`rtmp://rtmp.rtc.qq.com/push/` 。
4. 填写串流密钥格式如下：

```
房间号?sdkappid=应用&userid=用户名&usersig=签名
```

其中房间号、应用、用户名、签名需要换成业务的，参考流地址生成章节。例如：

```
yourRoomId?
sdkappid=140*****66&userid=******rtmp2&usersig=eJw1jdE**************ZLgi5UAgOzoMhrayt*cjb
miCJ699T09juc833IMT94Ld7I0iHZqVDzvVAqkZsG-
IKlzLiXOnEhswHu1iUyTc9pv*****D8MQwoA496Ke6U1ip4EAH4UMc5H9pSmv6MeTBWLamhwFnWRBZ8qKGRj8Yp-
wVbv*mGMVZqS7w-mMDQL
```



步骤3：设置输出

RTMP 后台不支持传输 B 帧，用户可以通过如下设置调整推流端软件的视频编码参数来去除 B 帧。

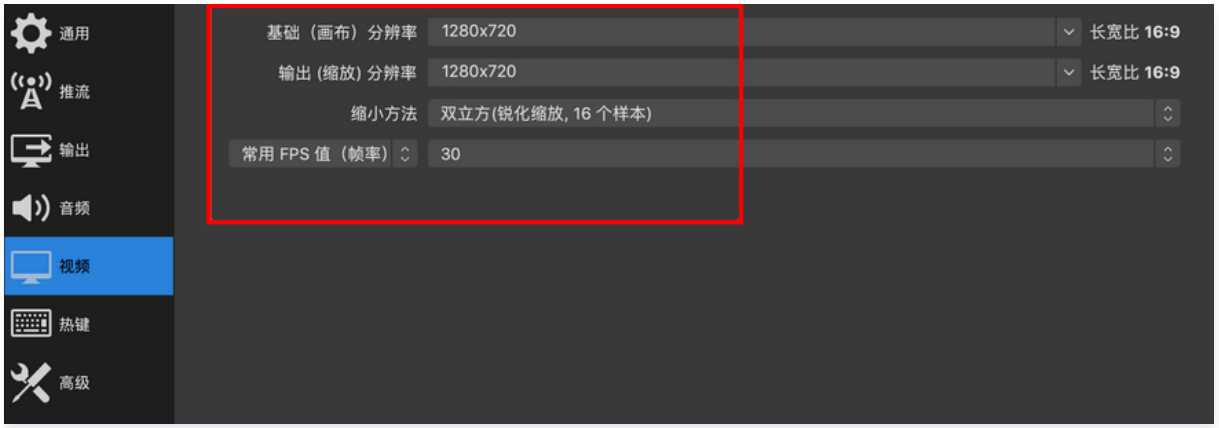
1. 在**设置**中单击**输出**页签进行配置。
2. 在**输出模式**中选择**高级**，**关键帧间隔建议填写1或2**，CPU 使用预设为 ultrafast，配置选择 baseline，微调选择 zerolatency，x264 选项填写 threads=1，单击**确定**保存设置。

> **警告：**

推流需要去除 B 帧，否则推流后连接会被断开，下面的配置选择 baseline 可去除 B 帧。

步骤4：设置视频选项

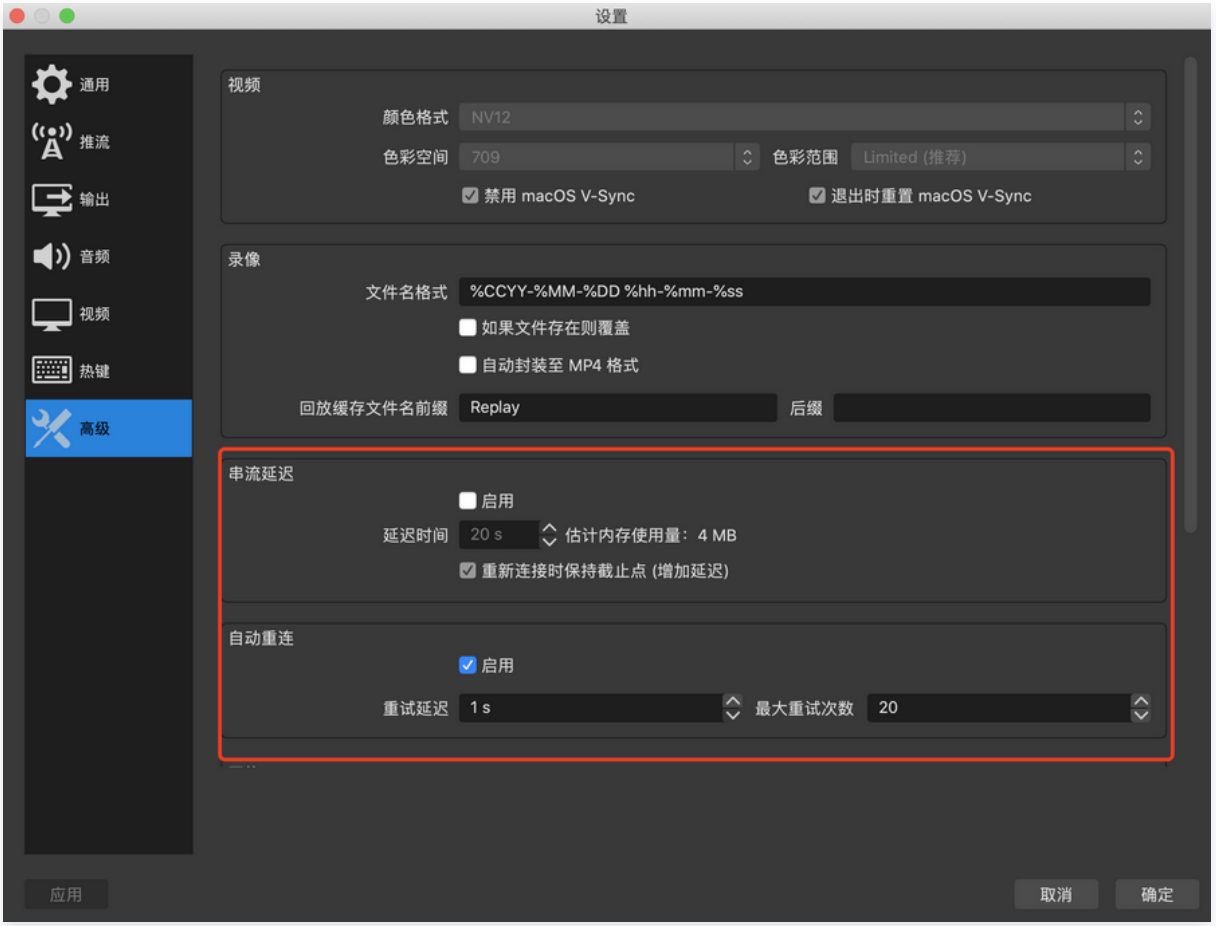
在**设置**中单击**视频**页签，设置分辨率和帧率。分辨率决定了观众看到的画面清晰程度，分辨率越高画面越清晰。FPS 是视频帧率，它控制观看视频的流畅，普通视频帧率有24帧 – 30帧，低于16帧画面看起来有卡顿感，而游戏对帧率要求比较高，一般小于30帧游戏会显得不连贯。



步骤5：设置高级选项

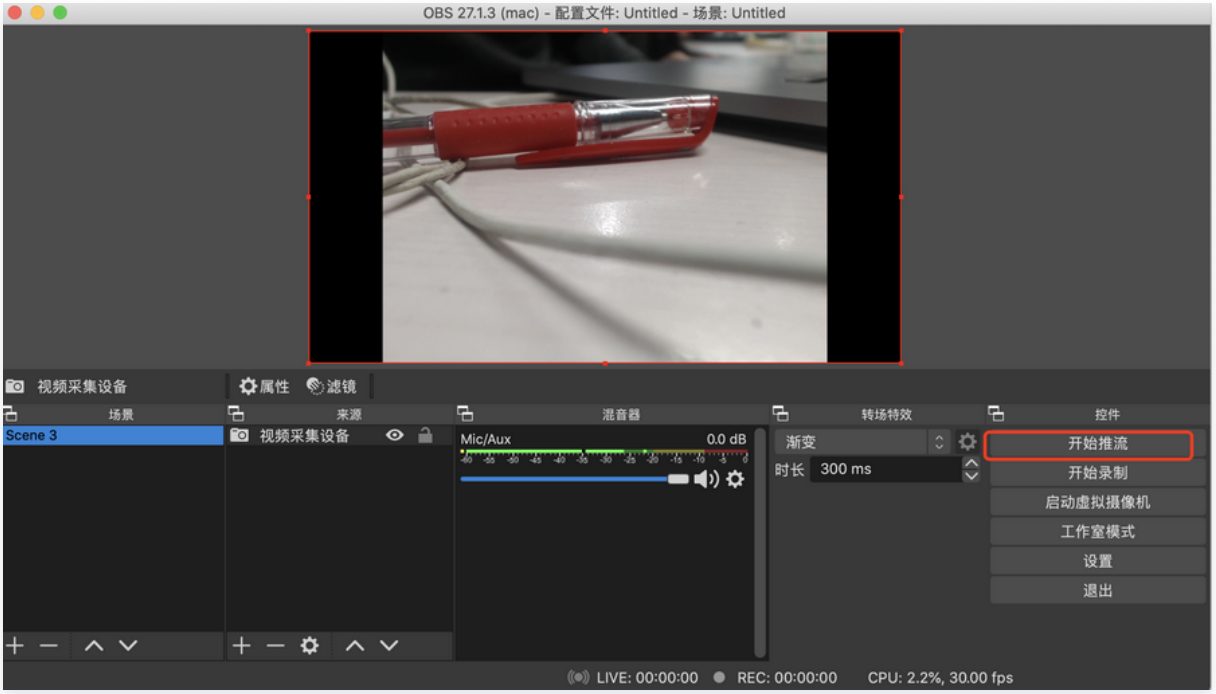
- 建议不启用**串流延迟**以减少端到端延迟。

- 启动**自动重连**，建议设置**重试延迟**时长尽量短，网络抖动时如果连接断开可尽快重连上。

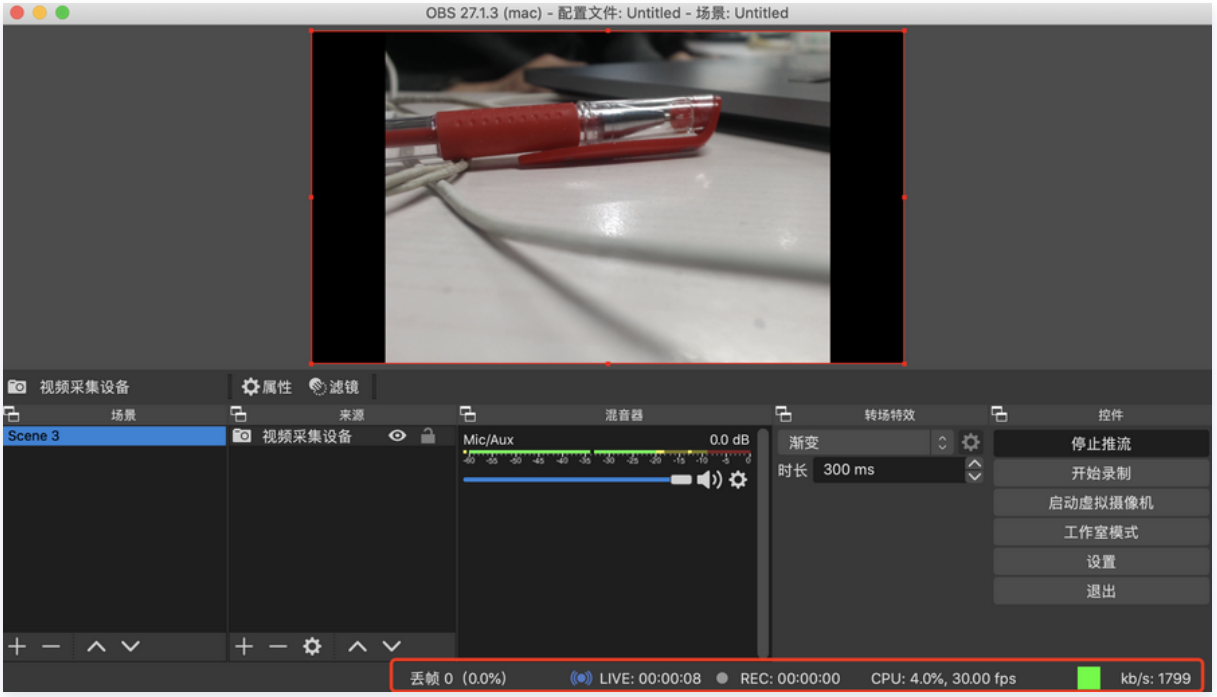


步骤6：单击推流

1. 查看 OBS 底部工具栏的**控件**，单击**开始推流**。



2. 推流成功后，正常情况在界面底部会展示推流状态，控制台仪表盘 上有该用户进房记录。



AI 降噪（TUIRoomKit）

最近更新时间：2024-04-03 13:35:01

AI 降噪源自腾讯天籁实验室的 AI 算法，能够智能检测和去除混合在传播信号中的噪声干扰，从而显著提高语音的质量，提升声音的清晰度，改善用户听感。AI 降噪特性在 TUIRoomKit 中的应用，可以让用户在办公室、网吧、商场、户外等各种环境下都能够获得清晰、稳定的声音体验。

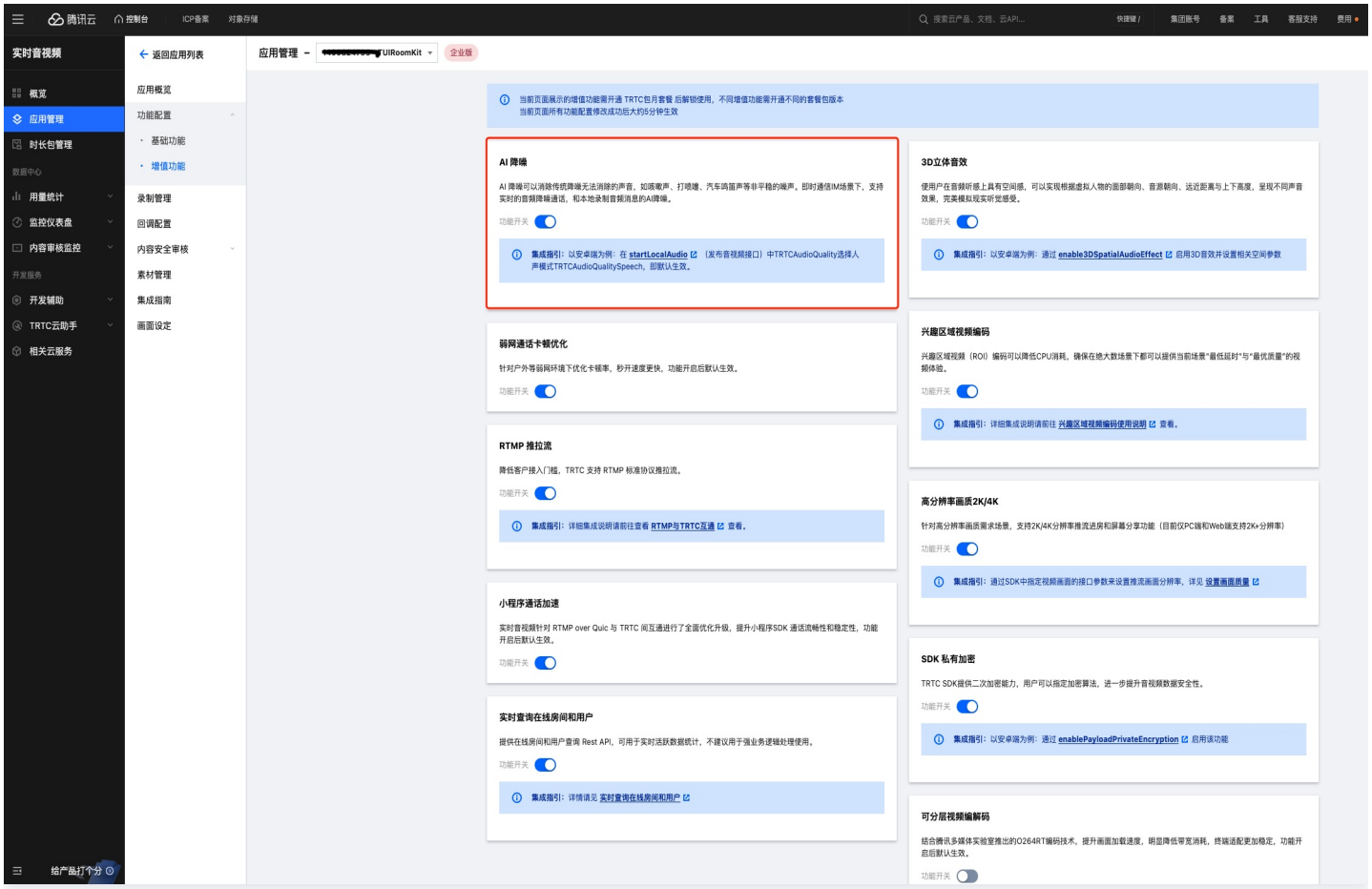
在线体验

您也可以进入我们的 实时音视频体验馆，在线体验 AI 降噪能力带来的优秀声音效果。

开通 AI 降噪

TUIRoomKit 提供了非常便捷的开通流程，您只需要两个步骤即可体验：

1. 进入 实时音视频控制台 ，选择您需要开通的应用。
2. 进入增值功能，打开 AI 降噪开关即可。



然后您就可以在您的应用上愉快的体验 TUIRoomKit 更好的降噪效果了！

更多特性（TUIRoomKit）
AI 实时字幕
Android

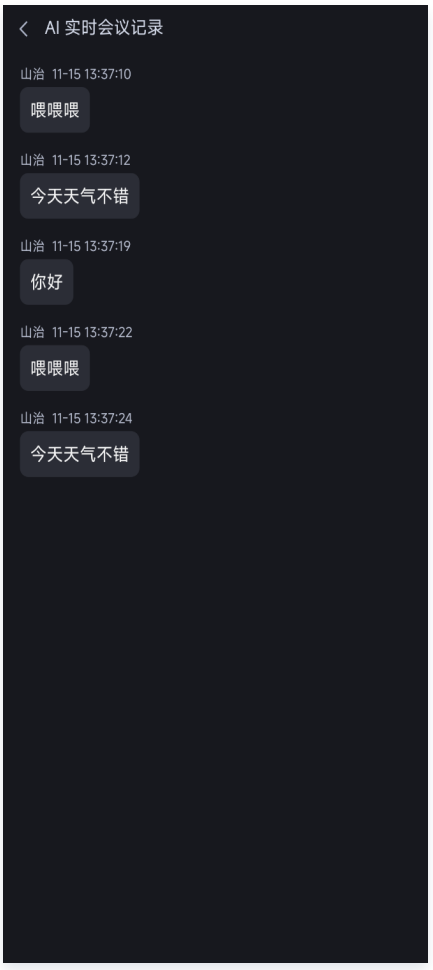
最近更新时间：2024-11-21 15:35:12

功能介绍

接入 TUIRoomKit 后，您可以通过点击底部栏的 "AI 助手" 开启 AI 实时字幕功能，该功能可实现：

- **AI 实时字幕**：将会议过程中的讨论内容以字幕形式进行显示。
- **AI 实时会议记录**：将会议过程中的讨论内容以文字形式记录下来。

> **注意：**
> 使用该功能需开通 TUIRoomKit 包月套餐 。除了正常通话费用（ 音视频时长计费说明 ）；该功能进行语音转文本时，会产生额外的 AI 智能识别费用，详情请参考 AI 智能识别计费说明 。

在底部栏点击 AI 助手	开启 AI 实时字幕	查看 AI 实时会议记录
		

功能接入

接入前提

开启底部栏中 “AI 助手” 按钮。

```
ConferenceSessionImpl.sharedInstance().isShowAISpeechToTextButton = true;
```

开启 AI 实时字幕

开启 AI 实时字幕功能，实际上是启动了一次 AI 转录任务，该过程中，后台会通过机器人拉流进行实时进行语音识别并下发字幕和转录消息。关于更多开启 AI 转录任务的参数配置、查询转录任务状态参见 AI 服务相关接口 。

- 您可在创建房间后，通过以下示例代码，开启 AI 实时字幕功能。

> **说明：**
> 语音转写功能需要密钥开启，由于代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。建议采用更安全的方式来使用密钥，详情请参见 云 API 密钥安全方案推荐 ，密钥可前往 API 密钥管理 进行获取。

Java

```
private String mTaskId;                          // 开启 AI 转录后的任务ID
private String aiSecretId  = "PLACEHOLDER";      // AI 语音转写的 ID
private String aiSecretKey = "PLACEHOLDER";      // AI 语音转写的 Key
private String roomId = "PLACEHOLDER";           // 房间 ID
private int    sdkAppId = 0L;                    // 您的 SDKAppId
private String sdkSecretKey = "PLACEHOLDER";     // 您的 SDKAppId 密钥
private String robotUserId = "PLACEHOLDER";      // 房间内 AI 机器人的 userId，和房间内的其他人
userId 不能一样

private void startAITranscription() {
    Runnable startAITranscriptionRun = () -> {
        try {
            // 代码泄露可能会导致 aiSecretId 和 aiSecretKey 泄露，并威胁账号下所有资源的安全性。
            // 以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：
https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
            Credential credential = new Credential(aiSecretId, aiSecretKey);
            TrtcClient client = new TrtcClient(credential, null);

            StartAITranscriptionRequest request = new StartAITranscriptionRequest();
            request.setRoomId(roomId);
            request.setRoomIdType(1L);
            request.setSdkAppId((long)sdkAppId);

            TranscriptionParams transcriptionParams = new TranscriptionParams();
            transcriptionParams.setUserId(robotUserId);
            transcriptionParams.setUserSig(GenerateTestUserSig.genTestUserSig(sdkAppId,
robotUserId, sdkSecretKey));
            request.setTranscriptionParams(transcriptionParams);
            StartAITranscriptionResponse response = client.StartAITranscription(request);
            mTaskId = response.getTaskId();
        } catch (TencentCloudSDKException e) {
            e.printStackTrace();
        }
```

```
    };
    new Thread(startAITranscriptionRun).start();
}
```

Kotlin

```
private var mTaskId: String? = null       // 开启 AI 转录后的任务ID
private val aiSecretId = "PLACEHOLDER"    // AI 语音转写的 ID
private val aiSecretKey = "PLACEHOLDER"   // AI 语音转写的 Key
private val roomId = "PLACEHOLDER"        // 房间 ID
private val sdkAppId = 0L                 // 您的 SDKAppId
private val sdkSecretKey = "PLACEHOLDER" // 您的 SDKAppId 密钥
private val robotUserId = "PLACEHOLDER"   // 房间内 AI 机器人的 userId，和房间内的其他人 userId 不能
一样

private fun startAITranscription() {
    Thread {
        try {
            // 代码泄露可能会导致 aiSecretId 和 aiSecretKey 泄露，并威胁账号下所有资源的安全性。
            // 以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
            // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
            val credential = Credential(aiSecretId, aiSecretKey)
            val client = TrtcClient(credential, null)

            val request = StartAITranscriptionRequest()
            request.roomId = roomId
            request.roomIdType = 1L
            request.sdkAppId = sdkAppId

            val transcriptionParams = TranscriptionParams()
            transcriptionParams.userId = robotUserId
            transcriptionParams.userSig = GenerateTestUserSig.genTestUserSig(sdkAppId,
robotUserId, sdkSecretKey)
            request.transcriptionParams = transcriptionParams

            val response = client.StartAITranscription(request)
            mTaskId = response?.taskId
        } catch (e: TencentCloudSDKException) {
            e.printStackTrace()
        }
    }.start()
}
```

关闭 AI 实时字幕功能

您可在离开房间/销毁房间时，调用以下代码关闭 AI 实时字幕功能。

Java

```
private void stopAITranscription() {
    Runnable stopAITranscriptionRun = () -> {
```

```
        try {
            Credential credential = new Credential(secretId, secretKey);
            TrtcClient client = new TrtcClient(credential, null);
            StopAITranscriptionRequest stopAITranscriptionRequest = new
StopAITranscriptionRequest();
            stopAITranscriptionRequest.setTaskId(mTaskId);
            client.StopAITranscription(stopAITranscriptionRequest);
        } catch (TencentCloudSDKException e) {
            e.printStackTrace();
        }
    };
    new Thread(stopAITranscriptionRun).start();
}
```

Kotlin

```
private fun stopAITranscription() {
    Thread {
        try {
            val credential = Credential(secretId, secretKey)
            val client = TrtcClient(credential, null)
            val stopAITranscriptionRequest = StopAITranscriptionRequest()
            stopAITranscriptionRequest.taskId = mTaskId
            client.StopAITranscription(stopAITranscriptionRequest)
        } catch (e: TencentCloudSDKException) {
            e.printStackTrace()
        }
    }.start()
}
```

说明：
如果您在使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

Web

最近更新时间：2024-11-21 15:35:12

功能介绍

接入 TUIRoomKit 后，您可以通过点击底部栏的 "AI 助手" 开启 AI 实时字幕功能，该功能可实现：

- **AI 实时字幕**：将会议过程中的讨论内容以字幕形式进行显示。
- **AI 实时会议记录**：将会议过程中的讨论内容以文字形式记录下来。

> **注意：**
> 使用该功能需开通 TUIRoomKit 包月套餐 。除了正常通话费用（ 音视频时长计费说明 ）；该功能进行语音转文本时，会产生额外的 AI 智能识别费用，详情请参考 AI 智能识别计费说明 。

功能接入

步骤1: 开启本地后台服务

> **注意：**
> 开启 AI 转录需要用到用户的云 id 与 key，敏感度较高，因此不在客户端集成此接口，您需要自己在您的业务后台增加 StartAITranscription 相关代码，此处以 Nodejs 服务为例，如果您的后台服务为其它语言，可 点击调试，生成对应语言事例代码。

开启 Nodejs 后台服务，客户端监听用户进房，通过 http 请求打开 AI 转录任务，示例代码： 点击下载 。

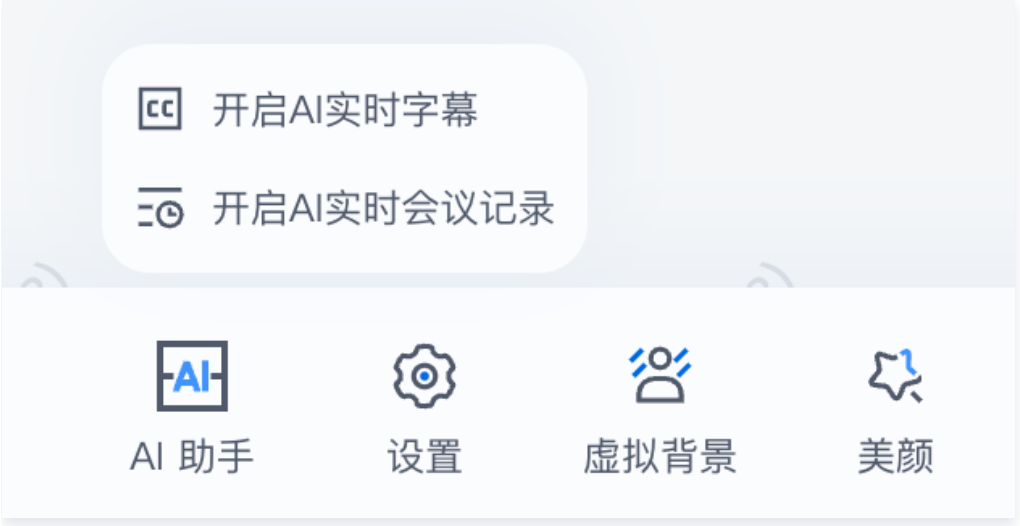
```
require('dotenv').config();
const express = require('express');
const tencentcloud = require('tencentcloud-sdk-nodejs-trtc');
const cors = require('cors')

const TrtcClient = tencentcloud.trtc.v20190722.Client;

// 从环境变量中获取腾讯云账户 SecretId 和 SecretKey
// 入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
// 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。
// 以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见:
https://cloud.tencent.com/document/product/1278/85305
// 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
const secretId = process.env.TENCENT_SECRET_ID || '';
const secretKey = process.env.TENCENT_SECRET_KEY || '';
const region = process.env.TENCENT_REGION || '';

const clientConfig = {
  credential: {
    secretId: secretId,
    secretKey: secretKey,
  },
  region: region,
  profile: {
    httpProfile: {
      endpoint: 'trtc.tencentcloudapi.com',
    },
  },
};

const client = new TrtcClient(clientConfig);

const app = express();
app.use(express.json());
app.use(cors());
app.post('/start', async (req, res) => {
  const { SdkAppId, RoomId, RoomIdType = 1, UserId, UserSig } = req.body;
  const params = {
    SdkAppId: SdkAppId,
    RoomId: RoomId,
    RoomIdType: RoomIdType,
    TranscriptionParams: {
      UserId: UserId,
      UserSig: UserSig,
    },
  };

  try {
    const data = await client.StartAITranscription(params);
    console.log('success',data)
    res.status(200).json(data);
  } catch (err) {
    console.error('error', err);
```

```
    res.status(500).json({ error: err.message });
  }
});

app.post('/stop', async (req, res) => {
  const { TaskId } = req.body;

  try {
    const data = await client.StopAITranscription({ TaskId: TaskId });
    res.status(200).json(data);
  } catch (err) {
    console.error('error', err);
    res.status(500).json({ error: err.message });
  }
});

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

步骤2: Roomkit 开启 AI 助手功能

> **注意:**
> Roomkit 仅进行 AI字幕/会议总结 的数据处理，实际 ASR 的开启时机是客户端用户进房，此处您可根据自己的业务需求调整时机。

```
<template>
  <conference-main-view display-mode="permanent"></conference-main-view>
</template>
<script setup lang="ts">
import { roomService } from '@tencentcloud/roomkit-web-vue3';
// 在 conference-main-view 组件 onmounted 之前调用
roomService.setComponentConfig({ AIControl: { visible: true } });
</script>
```



步骤3：Roomkit 监听用户进房，调用 node 服务，开启 AI 转录。

```
<template>
  <conference-main-view display-mode="permanent"></conference-main-view>
</template>
<script setup lang="ts">
import { conference } from '@tencentcloud/roomkit-web-vue3';
import { startAITranscription } from '../http';
const handleAITask = (data: {roomId: string}) => {
  const { roomId } = data;
  startAITranscription({
    RoomId: roomId,
    UserId: 'robot', // 此处需要一个机器人用户，以 robot 为例，此处不能与房间内用户的 userId 一样，建议使用 robot
    UserSig: 'xxx', // robot 对应的 userSig
    SdkAppId: sdkAppId,
    RoomIdType: 1, // 房间类型为字符串房间
  });
};
conference.on(RoomEvent.ROOM_JOIN, handleAITask);
conference.on(RoomEvent.ROOM_START, handleAITask);
onUnmounted(() => {
  conference.off(RoomEvent.ROOM_JOIN, handleAITask);
  conference.off(RoomEvent.ROOM_START, handleAITask);
});
</script>
```

```
// http.ts
import axios from 'axios';
const http = axios.create({
  baseURL: 'http://localhost:3000', // 您的 Node.js 服务地址
  timeout: 10000, // 请求超时时间
});

interface TranscriptionParams {
    SdkAppId: number;
    RoomId: string;
    RoomIdType?: number;
    UserId: string;
    UserSig: string;
  }

  interface StopParams {
    TaskId: string;
  }

// 启动 AI 转写任务
export function startAITranscription(params: TranscriptionParams) {
  return http.post('/start', params);
}

// 停止 AI 转写任务
```

```
export function stopAITranscription(params: StopParams) {
  return http.post('/stop', params);
}
```

其他文档

- 开始AI转录任务-实时音视频-文档中心-腾讯云
- 停止AI转录任务-API 中心-腾讯云
- 查询AI转录任务状态-API 中心-腾讯云
- AI 智能识别计费说明

> **说明：**
> 如果您在使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

弹幕聊天（Android&iOS&Flutter）

最近更新时间：2024-11-12 16:06:52

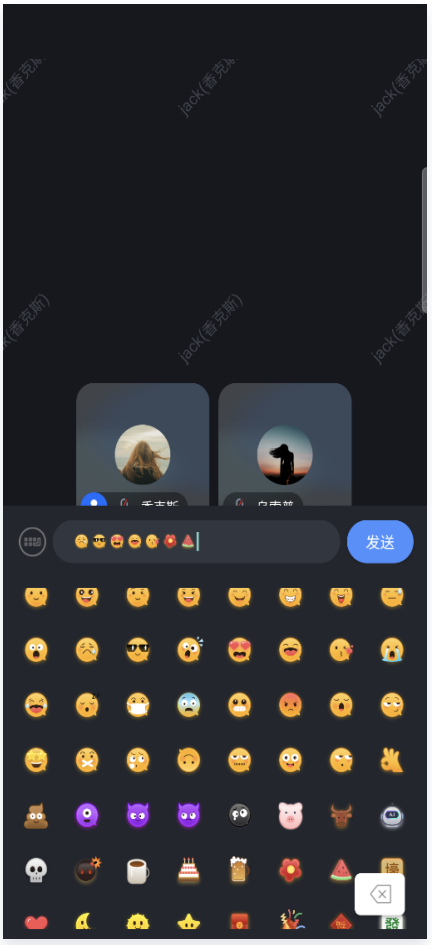
功能介绍

互动弹幕功能，支持如下功能：发送弹幕消息、插入自定义消息、自定义消息样式等。弹幕消息支持表情输入，增加消息趣味性，让互动更愉悦。

使用说明

您成功 接入 TUIRoomKit 后并成功登录后，可使用该功能。TUIRoomKit 弹幕支持以下功能：

- 发送/接收文字或表情消息。
- 隐藏弹幕按钮。
- 弹幕超过5秒自动消失。

弹幕聊天	发送表情	隐藏弹幕（底部栏：设置）
		

功能定制

如果当前的 UI 不满足您的需求，您可以修改通过修改源码来实现您满意的 UI 效果，不同平台请参考：

Android

您可以通过修改目录 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/FloatChat 的源码来实现您满意的 UI 效果，为了您更方便的定制 UI，这里对弹幕文件做了介绍。

```
// 位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/FloatC
hat
FloatChat
  └── View
  |   └── MessageBarrageView.java        // 每条弹幕消息的界面
  |   └── FloatChatSendView.java         // 弹幕发送的输入框界面
  └── TUIFloatChatButton.java            // 弹幕发送按钮
  └── TUIFloatChatDisplayView.java       // 弹幕消息展示的区域
```

iOS

您可以通过修改 Source/Common/Components/FloatChat 目录下的源码来实现您满意的 UI 效果，为了您更方便的定制 UI，这里对弹幕文件做了介绍。

```
FloatChat
  └── FloatChatButton.swift           // 弹幕发送按钮
  └── FloatChatDisplayView.swift       // 弹幕消息展示的区域
  └── FloatChatInputController.swift       // 弹幕发送的输入框界面
```

Flutter

Flutter 的弹幕聊天组件位于 tencent_float_chat_widget 插件中，您可以通过修改目录 `tencent_float_chat_widget/lib/float_chat` 下的源码来实现您满意的 UI 效果，为了您更方便的定制 UI，这里对弹幕文件做了介绍。

```
// 文件位置：tencent_float_chat_widget/lib/float_chat
float_chat
  ├── widgets
  |   ├── emoji_panel.dart           // 表情包面板widget
  |   ├── float_chat_button.dart     // 弹幕聊天按钮
  |   ├── input_widget.dart          // 发送弹幕的输入框widget
  |   ├── message_item.dart          // 单个弹幕消息widget
  |   └── message_widget.dart        // 所有弹幕消息widget
  └── view.dart                      // 房间内显示的整体弹幕widget(包含弹幕聊天按钮及所有弹幕消息
widget)
```

关键代码

Android

核心 API

弹幕聊天组件主要提供两个 API：
`TUIFloatChatButton`：点击后拉起弹幕发送界面。
`TUIFloatChatDisplayView`：弹幕显示区域。
在需要发送弹幕的场景，创建 TUIFloatChatButton，点击后可以拉起输入界面：

```
TUIFloatChatButton button = new TUIFloatChatButton(mContext, roomId);
mButtonContainer.addView(button);
```

在需要展示弹幕的场景，使用 TUIFloatChatDisplayView 来展示弹幕消息：

```
TUIFloatChatDisplayView view = new TUIFloatChatDisplayView(mContext, roomId);
mLayoutDisplayViewContainer.addView(view);
```

iOS

弹幕聊天组件主要分成两部分：
FloatChatButton：点击后拉起弹幕发送界面。
FloatChatDisplayView：弹幕显示区域。
您可自行创建这两个组件，并放置在视图的任意位置。

```
let floatchatButton = FloatChatButton()
floatchatButton.updateRoomId(roomId:"your room Id") // 设置roomId
self.view.addSubView(floatchatButton)
```

```
let displayView = FloatChatDisplayView()
self.view.addSubView(displayView)
```

Flutter

弹幕聊天组件（tencent_float_chat_widget）中，对外暴露了以下几个类：

- **FloatChatWidget**：包含弹幕聊天按钮和弹幕消息显示区域。您可以在需要的地方通过以下方式进行布局：

```
import 'package:tencent_float_chat_widget/tencent_float_chat_widget.dart';

FloatChatWidget(roomId: yourRoomId)  // 在您需要的地方布局，传入发送和接收消息的房间的roomID。
```

- **InputWidget**：发送弹幕聊天消息的输入框。您需要将其布局在您所布局页面的 `Stack` 的最上层，以防止被其他组件遮挡。布局完成后默认不显示，当您点击**FloatChatWidget**中的弹幕聊天按钮后，输入框及键盘会自动弹出。

```
import 'package:tencent_float_chat_widget/tencent_float_chat_widget.dart';
```

```
Stack(
    children: [
      ...yourWidget,          // 您的其他widget,这里仅作示例
      const InputWidget(),    // 发送弹幕聊天消息的输入框
    ],
)
```

- **FloatChatManager**：类中提供了一个deleteStatus()方法，当您退出当前房间时，需要调用此方法来清空弹幕聊天状态：

```
import 'package:tencent_float_chat_widget/tencent_float_chat_widget.dart';

FloatChatManager().deleteStatus();
```

房间密码（全平台）

最近更新时间：2025-04-29 08:56:52

功能介绍

TUIRoomKit 支持房间加密，您可以通过接入 TUIRoomKit 来预定或创建一个密码房间，**若 TUIRoomKit 的 UI 交互不满足您的产品需求，您可以接入 TUIRoomEngineSDK 自定义实现房间密码相关的交互功能，具体请参见 关键代码 。**

使用说明

创建密码房间

成功接入 TUIRoomKit 并登录成功后可创建密码房间。不同平台创建密码房间请参见：

Android

请确保您已成功 接入 TUIRoomKit 并 登录 成功，调用以下示例代码即可创建密码房间：

```
ConferenceDefine.StartConferenceParams params = new
ConferenceDefine.StartConferenceParams("222222"); // 请替换 "222222" 为您自定义的房间号
params.passWord = "123456";// 请替换 "123456" 为您设置的密码（小于等于6位的纯数字）。
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent);
```

iOS

请确保您已成功 接入 TUIRoomKit 并 登录 成功，调用以下示例代码即可创建密码房间：

Swift

```
import TUIRoomKit

func quickStartConference() {
    let vc = ConferenceMainViewController()
    let params = StartConferenceParams(roomId: "111111")  // 请替换 "111111" 为您自定义的房
间号
    params.password = "123456"  // 请替换 "123456" 为您自定义的房间密码（小于等于6位的纯数字）。
    vc.setStartConferenceParams(params: params)
    navigationController?.pushViewController(vc, animated: true)
 }
```

OC

```
#import "TUIRoomKit/TUIRoomKit-Swift.h"

- (void)quickStartConference {
    ConferenceMainViewController * vc = [[ConferenceMainViewController alloc]init];
```

```
    StartConferenceParams * params = [[StartConferenceParams alloc]
                                      initWithRoomId: @"111111" // 请替换 "111111" 为您自定
义的房间号
                                   isOpenMicrophone:YES
                                       isOpenCamera:NO
                                     isOpenSpeaker:YES
                          isMicrophoneDisableForAllUser:NO
                              isCameraDisableForAllUser:NO
                                     isSeatEnabled:NO
                                              name:@"YourRoomName"
                                          password:@"123456"]; // 请替换 "123456" 为您自
定义的房间密码
    [vc setStartConferenceParamsWithParams:params];
    [self.navigationController pushViewController:vc animated:YES];
}
```

Web

请确保您已成功 接入 TUIRoomKit 并 登录 成功，调用以下示例代码即可创建密码房间：

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', { // 请替换 '123456' 为您的房间号
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
  password: '123456', // 请替换 '123456' 为您设置的密码
});
```

⚠ 注意：

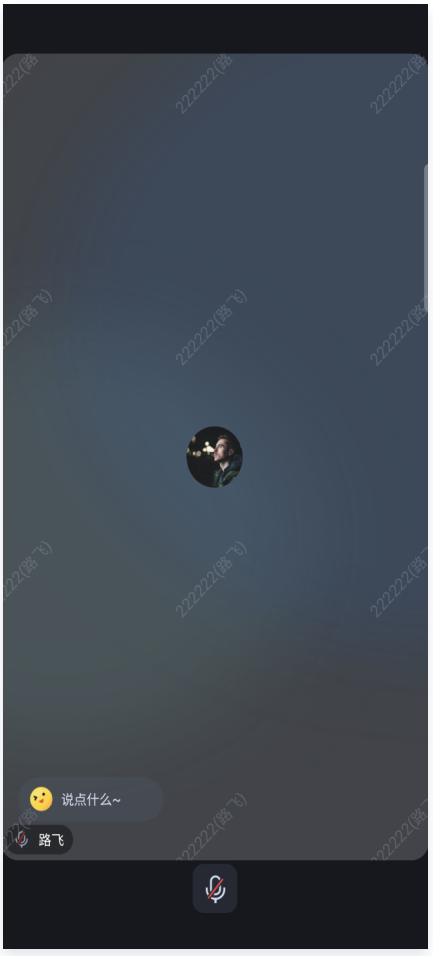
密码格式：数字、字母、特殊字符：`~`!@#$%^&*()-_=+{}[\]\\|;:'",<.>/?`

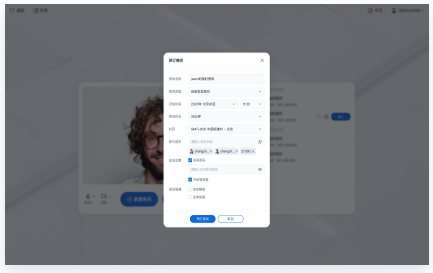
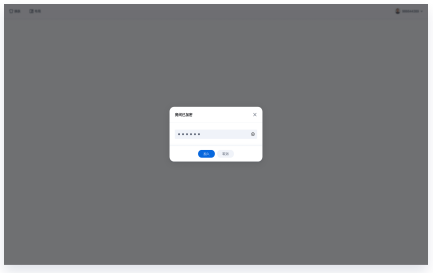
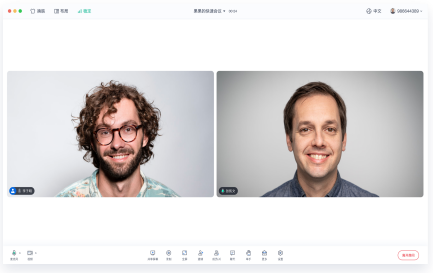
预定密码房间

预定密码房间，请参见 预定会议（Android&iOS&Flutter）或 预定会议（Web&Electron）指引打或开预定会议界面，并设置密码。交互方案如下：

- **受邀成员进入房间**：可通过会议列表或房间号，无需密码直接进入房间。
- **未受邀成员进入房间**：只能通过房间号进入房间，且需要输入正确密码。

Android & iOS		
预定密码房间	未受邀成员加入房间输入密码	受邀成员可直接进入房间



Web		
预定密码房间	未受邀成员加入房间输入密码	受邀成员可直接进入房间
		

说明：
若预定会议 UI 不满足您的需求，您需要按自己的UI交互设计来实现该功能，相关 API 调用请参见 关键代码 。

功能定制

如果当前的 UI 不满足您的需求，您可以通过修改源码来实现您满意的 UI 效果，不同平台请参考：

Android

您可以通过修改 Android/TUIRoomKit/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/ScheduleConference/view 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对房间密码的文件做了介绍。

```
// 位置：
Android/TUIRoomKit/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/ScheduleConference/view
view
  ├── EnterConferencePasswordView.java   // 输入密码弹窗的界面
  └── SetConferenceEncryptView.java      // 预定会议设置密码的界面
```

iOS

您可以通过修改 iOS/TUIRoomKit/Source/View 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对房间密码的文件做了介绍。

```
view
  ├── ConferencePasswordView.swift            // 输入密码弹窗的界面
  └── ScheduleConferenceDataHelper.swift      // 预定会议中设置密码弹出界面样式
```

Web

您可以通过修改如下位置目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对房间密码的文件做了介绍。

```
// 位置：TUIRoomKit/Web/roomkit/vue3/src/TUIRoom/components/
PreRoom
  ├── PasswordDialog.vue   // 输入密码弹窗的界面

ScheduleConference/ScheduleConferencePanel
  ├── ScheduleConferencePanelPC.vue      // 预定会议设置密码的界面 Web
  └── ScheduleConferencePanelH5.vue      // 预定会议设置密码的界面 H5
```

Electron

您可以通过修改如下位置目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对房间密码的文件做了介绍。

```
// 位置：TUIRoomKit/Electron/roomkit/vue3/src/TUIRoom/components/
PreRoom
  ├── PasswordDialog.vue   // 输入密码弹窗的界面

ScheduleConference/ScheduleConferencePanel
  ├── ScheduleConferencePanelPC.vue      // 预定会议设置密码的界面 Web
```

```
└── ScheduleConferencePanelH5.vue      // 预定会议设置密码的界面 H5
```

说明：
若您还有其他关于密码房间的功能定制需求，可通过 TUIRoomKit 技术交流平台 zhiliao，向我们反馈。

关键代码

- 创建密码房间，不同平台请参考：

Android

```
public abstract void createRoom(TUIRoomDefine.RoomInfo roomInfo,
TUIRoomDefine.ActionCallback callback);
```

您可通过设置 roomInfo 参数的 password 字段设置房间密码。接口详情请参见 createRoom 。
示例代码如下：

```
TUIRoomDefine.RoomInfo roomInfo = new TUIRoomDefine.RoomInfo();
roomInfo.roomId = "222222";  // 请替换 "222222" 为您的房间号。
roomInfo.password = "123456" // 请替换 "123456" 为您设置的密码。
TUIRoomEngine.sharedInstance().createRoom(roomInfo, new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
        // 创建房间成功回调
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // 创建房间失败回调
    }
});
```

iOS

```
- (void)createRoom:(TUIRoomInfo *)roomInfo onSuccess:(TUISuccessBlock)onSuccess onError:
(TUIErrorBlock)onError NS_SWIFT_NAME(createRoom(_:onSuccess:onError:));
```

您可通过设置 roomInfo 参数的 password 字段设置房间密码。接口详情请参见 createRoom 。
示例代码如下：

Swift

```
import RTCRoomEngine
```

```
func createRoom() {
    let roomInfo = TUIRoomInfo()
    roomInfo.roomId = "111111"      // 请替换 "111111" 为您自定义的房间号
    roomInfo.password = "123456"    // 请替换 "123456" 为您自定义的房间密码
    TUIRoomEngine.sharedInstance().createRoom(roomInfo) {
        print("CreateRoom success")
    } onError: { code, message in
        print("CreateRoom error, code:\(code), message:\(message)")
    }
}
```

OC

```
#import "RTCRoomEngine/TUIRoomDefine.h"
#import "RTCRoomEngine/TUIRoomEngine.h"

- (void)createRoom {
    TUIRoomInfo * roomInfo = [[TUIRoomInfo alloc] init];
    roomInfo.roomId = @"111111";      // 请替换 "111111" 为您自定义的房间号
    roomInfo.password = @"123456";    // 请替换 "123456" 为您自定义的房间密码
    [[TUIRoomEngine sharedInstance] createRoom:roomInfo onSuccess:^{
        NSLog(@"CreateRoom success");
    } onError:^(TUIError code, NSString * _Nonnull message) {
        NSLog(@"CreateRoom error, code:%ld, message:%@", (long)code, message);
    }];
}
```

Web/Electron

您可通过设置 password 字段设置房间密码，接口详情请参见 start 。
示例代码如下：

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {  // 请替换 '123456' 为您设置的密码
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
  password: '123456',  // 请替换 '123456' 为您设置的密码
});
```

- **预定密码房间，不同平台请参考：**

Android

```
public abstract void scheduleConference(ConferenceInfo conferenceInfo,
TUIRoomDefine.ActionCallback callback);
```

您可通过设置 conferenceInfo 参数的 password 字段设置房间密码。接口详情请参见 scheduleConference 。示例代码如下：

```
TUIConferenceListManager manager =
TUIRoomEngine.sharedInstance().getExtension(CONFERENCE_LIST_MANAGER);
TUIConferenceListManager.ConferenceInfo conferenceInfo = new
TUIConferenceListManager.ConferenceInfo();
conferenceInfo.basicRoomInfo.roomId = "222222";    // 请替换 "222222" 为您预定房间的房间号。
conferenceInfo.basicRoomInfo.password = "123456"; // 请替换 "123456" 为您设置的密码。

manager.scheduleConference(conferenceInfo, new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
        // 预定房间成功回调
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // 预定房间失败回调
    }
});
```

iOS

```
- (void)scheduleConference:(TUIConferenceInfo *)conferenceInfo onSuccess:
(TUISuccessBlock)onSuccess onError:(TUIErrorBlock)onError
NS_SWIFT_NAME(scheduleConference(_:onSuccess:onError:));
```

您可通过设置 conferenceInfo 参数的 password 字段设置房间密码。接口详情请参见 scheduleConference 。示例代码如下：

Swift

```
import RTCRoomEngine

func scheduleConference() {
    let manager = TUIRoomEngine.sharedInstance().getExtension(extensionType:
.conferenceListManager) as? TUIConferenceListManager
    let conferenceInfo = TUIConferenceInfo()
    conferenceInfo.basicRoomInfo.roomId = "111111"     // 请替换 "111111" 为您自定义的房间号
    conferenceInfo.basicRoomInfo.password = "123456"   // 请替换 "123456" 为您自定义的房间密码
    manager?.scheduleConference(conferenceInfo, onSuccess: {
        print("ScheduleConference success")
    }, onError: { code, message in
        print("ScheduleConference failed, code:\(code), message:\(message)")
```

```
    })
}
```

OC

```
#import "RTCRoomEngine/TUIRoomEngine.h"
#import "RTCRoomEngine/TUIConferenceListManager.h"

- (void)scheduleConference {
    TUIConferenceListManager * manager = [[TUIRoomEngine sharedInstance] getExtension:
TUIExtensionTypeConferenceListManager];
    TUIConferenceInfo * conferenceInfo = [[TUIConferenceInfo alloc] init];
    conferenceInfo.basicRoomInfo.roomId = @"111111";     // 请替换 "111111" 为您自定义的房间号
    conferenceInfo.basicRoomInfo.password = @"123456";  // 请替换 "123456" 为您自定义的房间密
码
    [manager scheduleConference:conferenceInfo onSuccess:^{
        NSLog(@"ScheduleConference success");
    } onError:^(TUIError code, NSString * _Nonnull message) {
        NSLog(@"ScheduleConference failed, code:%ld, message:%@", (long)code, message);
    }];
}
```

- 进入密码房间

Android

```
public abstract void enterRoom(String roomId, TUIRoomDefine.RoomType roomType,
TUIRoomDefine.EnterRoomOptions options, TUIRoomDefine.GetRoomInfoCallback callback);
```

您可通过设置 options 参数的 password 字段设置进房密码。接口详情请参见 enterRoom 。示例代码如下：

```
String roomId = "222222";     // 请替换 "222222" 为您加入的房间号
TUIRoomDefine.EnterRoomOptions options = new TUIRoomDefine.EnterRoomOptions();
options.password = "123456"; // 请替换 "123456" 为您加入房间的密码。
TUIRoomEngine.sharedInstance().enterRoom(roomId, TUIRoomDefine.RoomType.CONFERENCE,
options, new TUIRoomDefine.GetRoomInfoCallback() {
    @Override
    public void onSuccess(TUIRoomDefine.RoomInfo engineRoomInfo) {
        // 进入房间成功回调
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // 进入房间失败回调
        if (error == TUICommonDefine.Error.WRONG_PASSWORD) {
            // 密码错误，这里执行进房密码错误的业务。
        }
```

```
    }
});
```

iOS

```
- (void)enterRoom:(NSString *)roomId roomType:(TUIRoomType)roomType options:
(TUIEnterRoomOptions *)options onSuccess:(TUIRoomInfoBlock)onSuccess onError:
(TUIErrorBlock)onError NS_SWIFT_NAME(enterRoom(_:roomType:options:onSuccess:onError:));
```

您可通过设置 options 参数的 password 字段设置进房密码。接口详情请参见 enterRoom 。示例代码如下：

Swift

```
import RTCRoomEngine

func enterRoom() {
    let roomId = "111111"         // 请替换 "111111" 为您自定义的房间号
    let options = TUIEnterRoomOptions()
    options.password = "123456"   // 请替换 "123456" 为您自定义的房间密码
    TUIRoomEngine.sharedInstance().enterRoom(roomId, roomType: .conference, options:
options) { roomInfo in
        print("EnterRoom success")
    } onError: { code, message in
        print("EnterRoom failed, code:\(code), message:\(message)")
    }
}
```

OC

```
#import "RTCRoomEngine/TUIRoomEngine.h"

- (void)enterRoom {
    NSString * roomId = @"111111";    // 请替换 "111111" 为您自定义的房间号
    TUIEnterRoomOptions * options = [[TUIEnterRoomOptions alloc] init];
    options.password = @"123456";     // 请替换 "123456" 为您自定义的房间密码
    [[TUIRoomEngine sharedInstance] enterRoom:roomId roomType:TUIRoomTypeConference
options:options onSuccess:^(TUIRoomInfo * _Nullable roomInfo) {
        NSLog(@"EnterRoom success");
    } onError:^(TUIError code, NSString * _Nonnull message) {
        NSLog(@"EnterRoom failed, code:%ld, message:%@", (long)code, message);
    }];
}
```

Web/Electron

您可通过设置 password 字段进入房间，接口详情请参见 join 。

示例代码如下：

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.join('123456', { // 请替换 '123456' 为您加入的房间号
  isOpenCamera: false,
  isOpenMicrophone: false,
  password: '123456', // 请替换 '123456' 为您进入房间的密码
});
```

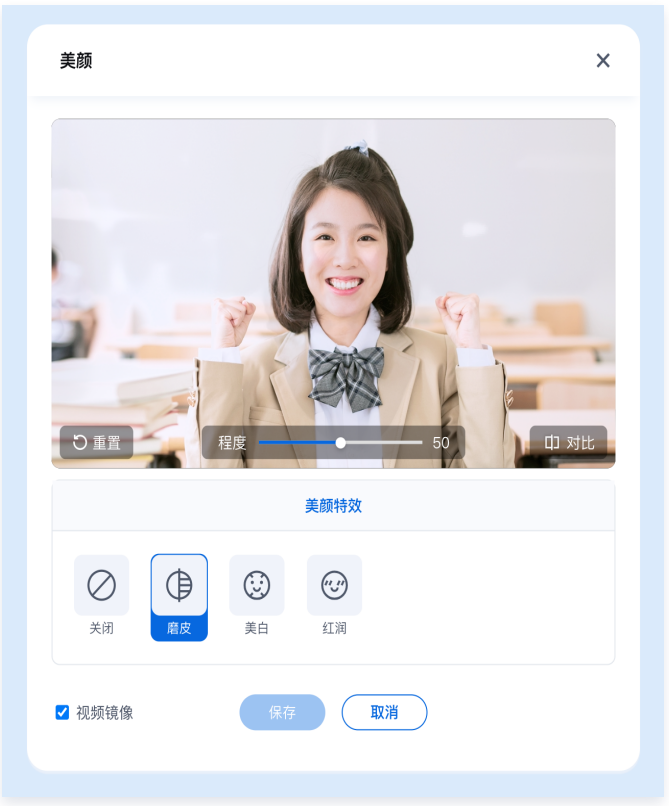
基础美颜（Web）

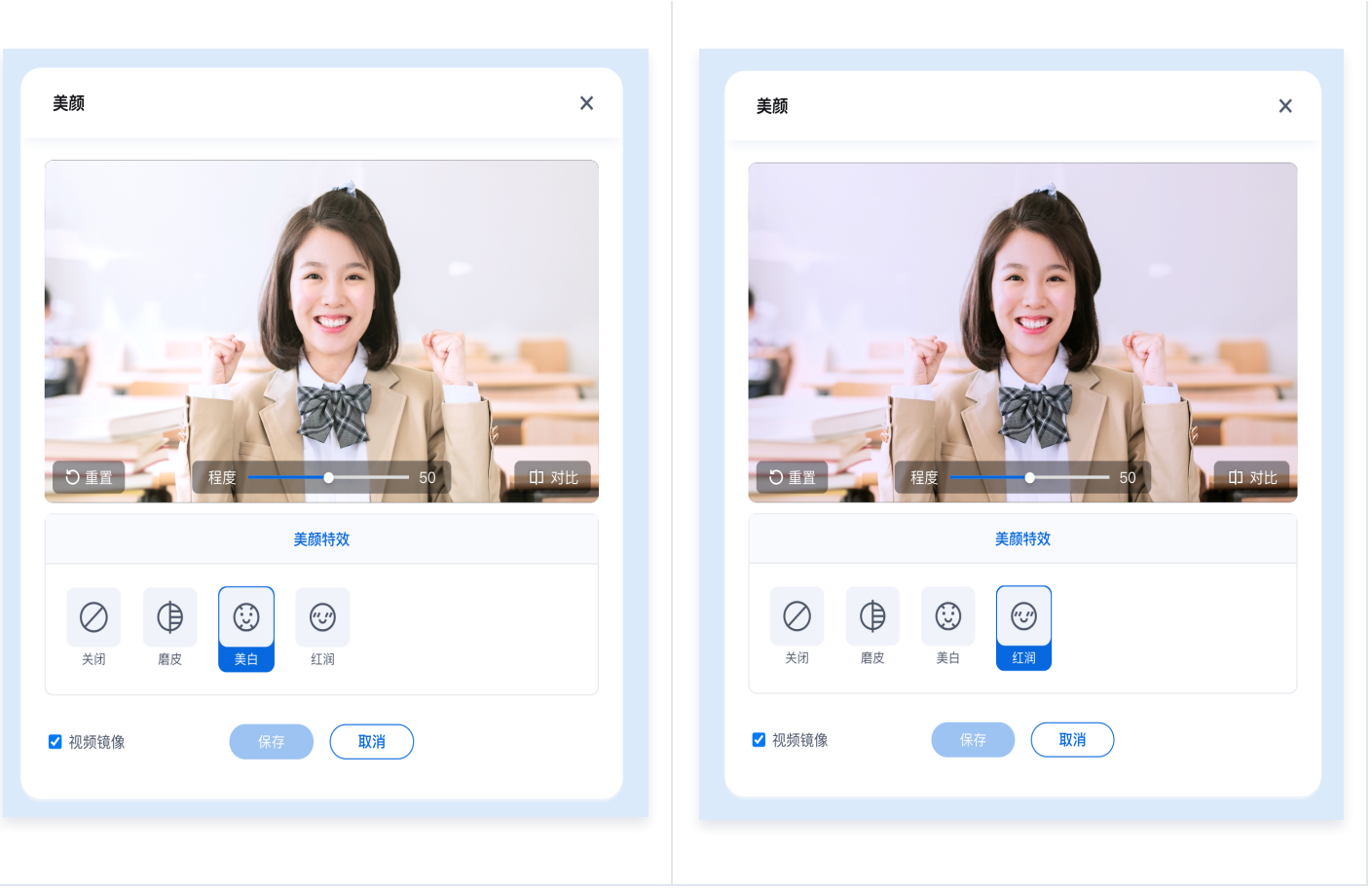
最近更新时间：2024-09-27 09:56:01

TUIRoomKit 推出了基础美颜功能，使得用户在进行多人会议时可以优化自己的磨皮、美白和红润效果，提升视频会议的视觉体验，增加会议趣味性。本文将详细介绍如何在 TUIRoomKit 组件中使用这一特性。

集成效果

在 TUIRoomKit 组件中集成基础美颜功能后，显示效果如下：

关闭	磨皮
	
美白	**红润**



准备条件

在使用腾讯云提供的基础美颜功能前，您需要前往控制台，为应用开通多人会议服务。具体步骤请参见 开通服务 。

开启基础美颜

> **注意：**
> - H5 暂不支持，仅适用于 Web PC 端。
> - 需使用 TUIRoomKit v2.6.2 及以上版本。

TUIRoomKit 的 UI 方案默认支持设置基础美颜。若您不需要基础美颜功能的展示，可以通过如下代码进行基础美颜 UI 的隐藏：

Vue3

```
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue3';
conference.hideFeatureButton(FeatureButton.BasicBeauty);
```

Vue2

```
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue2.7';
conference.hideFeatureButton(FeatureButton.BasicBeauty);
```

常见问题

开启基础美颜无反应或有延迟?

网络较差时，基础美颜模型文件可能未下载完，从而导致开启基础美颜失败。

关闭摄像头，能否能开启基础美颜?

不能。

版权所有：腾讯云计算（北京）有限责任公司

设置昵称、头像（全平台）

最近更新时间：2024-08-02 17:12:01

本文介绍如何在 TUIRoomKit 中设置用户的头像和昵称。

设置头像、昵称

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：

Web&H5

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl: 'http://xxx' });
```

Android

```
TUIRoomEngine.setSelfInfo("userName", "avatarUrl", null);
```

iOS

```
import TUIRoomEngine

TUIRoomEngine.setSelfInfo(userName: "xxx", avatarUrl: "xxx") {
    print("setSelfInfo success")
} onError: { code, message in
    print("setSelfInfo failed, code:\(code),message:\(message)")
}
```

uni-app

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl: 'http://xxx' });
```

Flutter

```
import 'package:rtc_room_engine/rtc_room_engine.dart';
```

```
TUIRoomEngine.setSelfInfo("userName", "avatarURL");
```

Electron

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl: 'http://xxx' });
```

> **注意：**
> 由于用户隐私限制，昵称和头像更新可能会有延迟。如您需要更高的实时性，可以使用 会中修改昵称 功能。

会中修改昵称

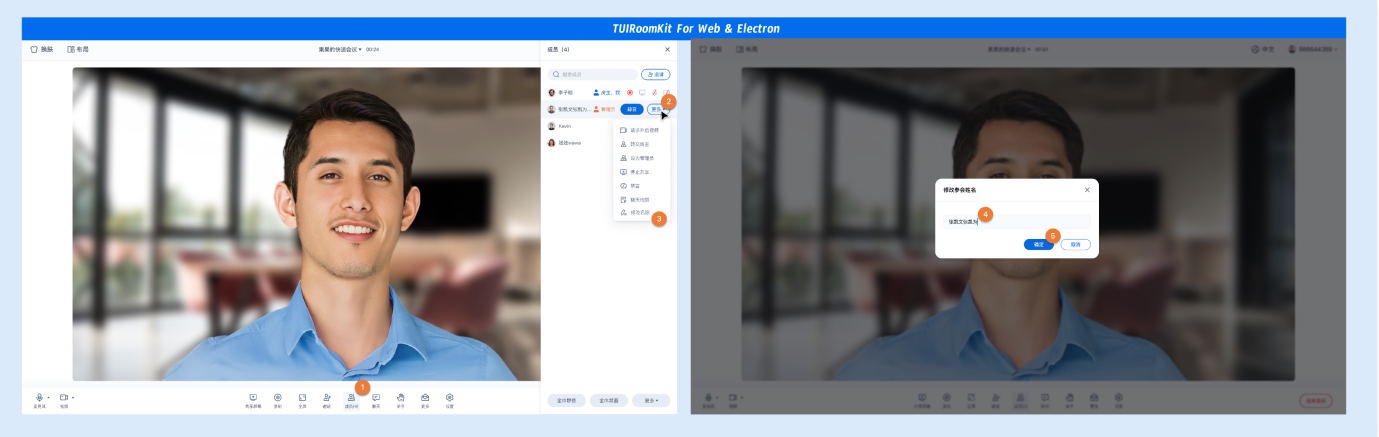
在会议中，参会人员可以实时修改自己的昵称，以方便在不同场景中展示不同的身份。修改后的昵称会即时生效，但**仅限于当前会议**。

> **说明：**
> 会中改名特性需使用 **TUIRoomKit v2.5.0** 及以上版本，该特性目前**仅支持** Web、Electron、H5 和小程序端。

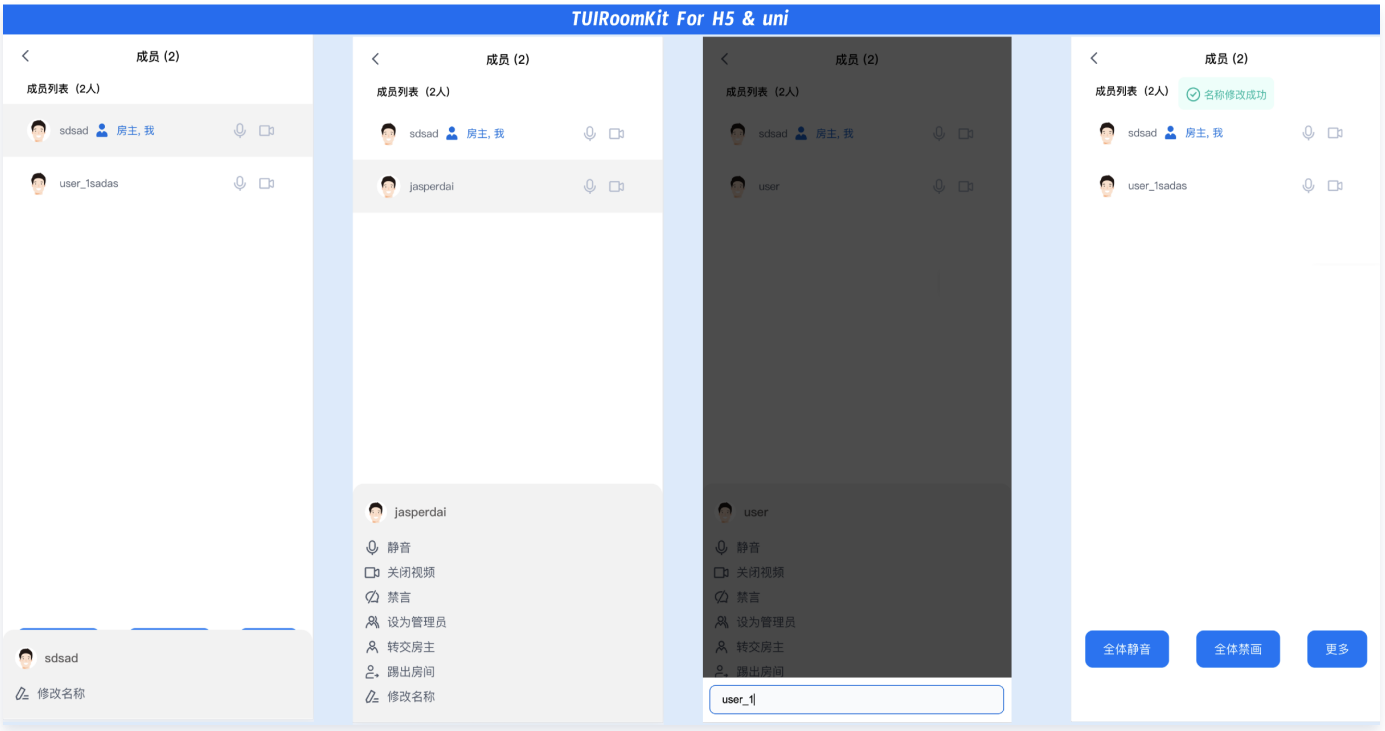
操作流程

1. 在TUIRoomKit内，会中点击底部工具栏**成员管理** > 选中自己或需要改名用户 > **更多** > **修改名称**；
2. 弹窗中输入需要修改后的名称后，点击确定即可即时生效。

Web&Electron



H5&小程序



操作权限

- 普通用户仅能修改自己的昵称。
- 房主或管理员可以修改自己或其他用户的昵称。

示例代码

如您需要在您的项目中，自行修改以支持会中修改昵称的功能，可使用如下 TUIRoomEngine 接口：

Web&H5

```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```

Electron

```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
```

版权所有：腾讯云计算（北京）有限责任公司

```
});
```

小程序

```
const roomEngine = new TUIRoomEngine();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```

会中聊天
Web&Electron

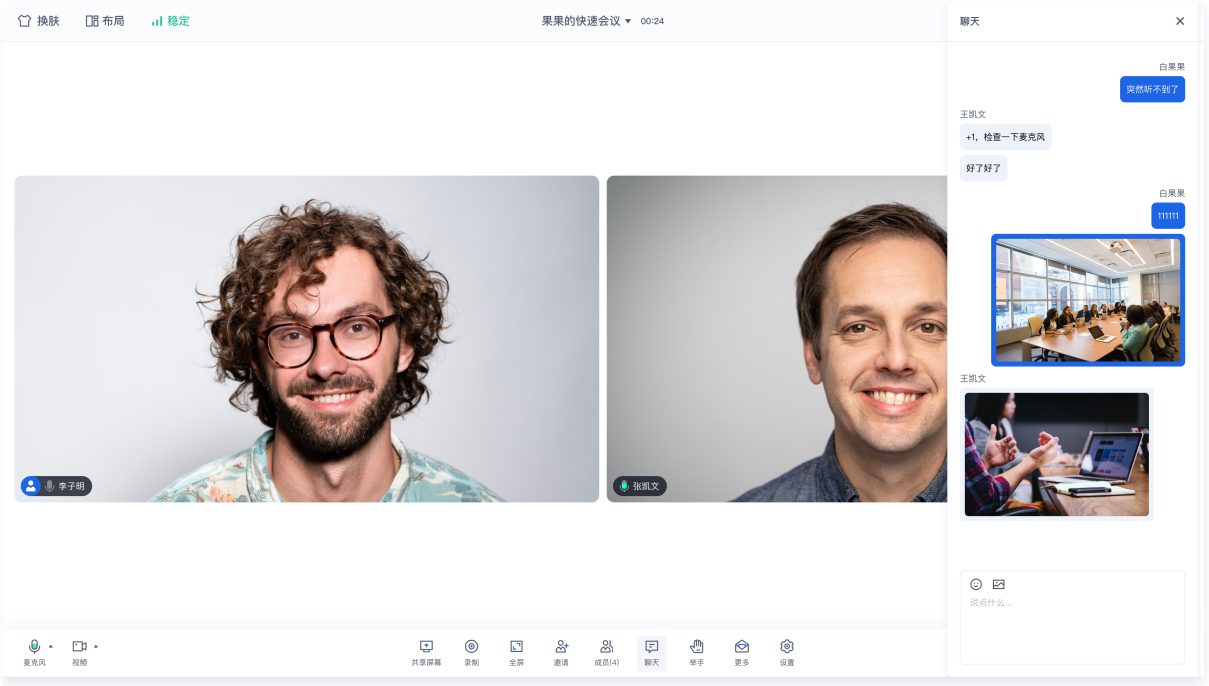
最近更新时间：2024-11-05 18:48:12

在视频会议中，参会者可以实时在聊天区发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

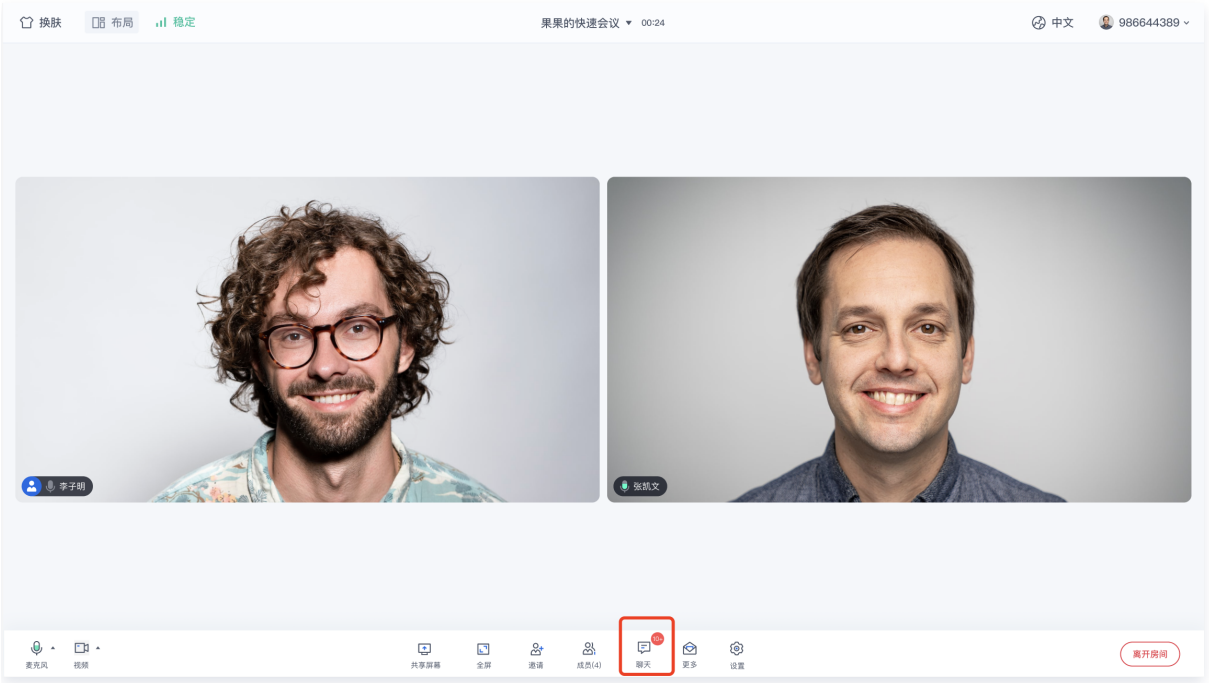
功能介绍

聊天互动

在会议界面底部菜单栏中，您可以找到一个名为**聊天**的按钮。点击后屏幕右侧会出现聊天界面。在聊天框中，参会人员可以通过发送文字、图片、文件信息与其他参会者进行沟通。这样的设计既方便了参会者之间的实时交流，又不会影响到正在发言的人。

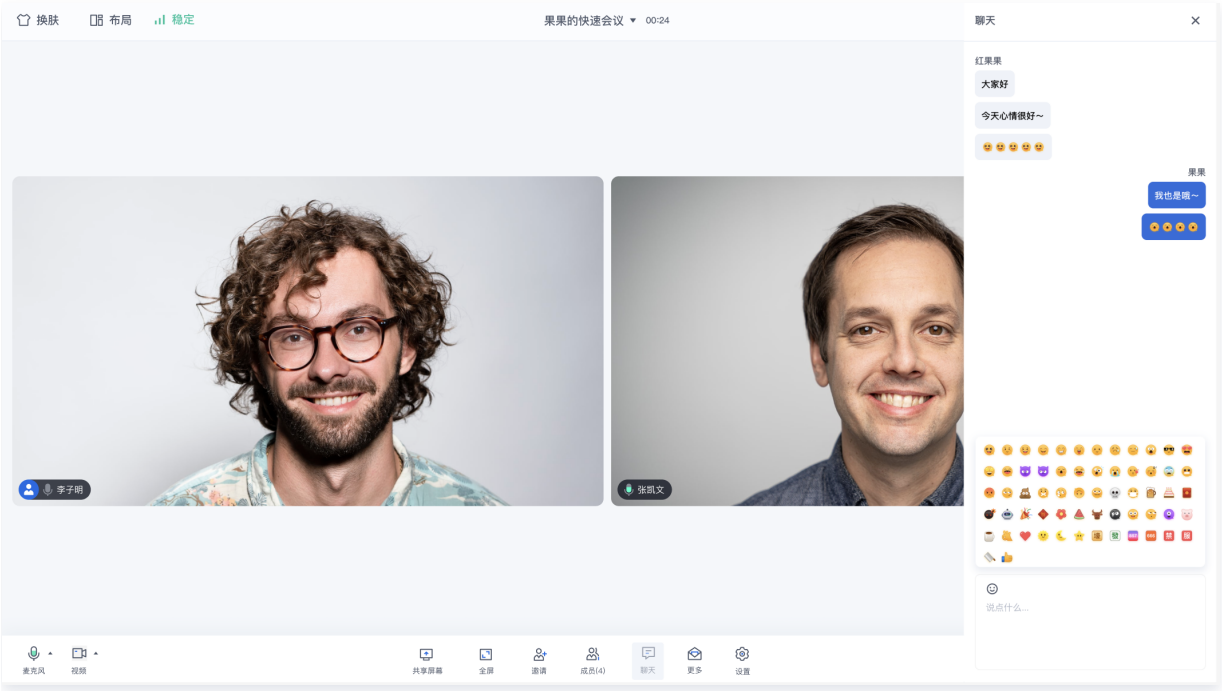


当聊天区域处于未激活状态，且接收到新的聊天消息时，底部工具栏**聊天**右上角会出现一个红色气泡提示，提醒您有未读信息。这样，您可以随时关注会议中的实时动态，确保不错过任何重要信息。



表情互动

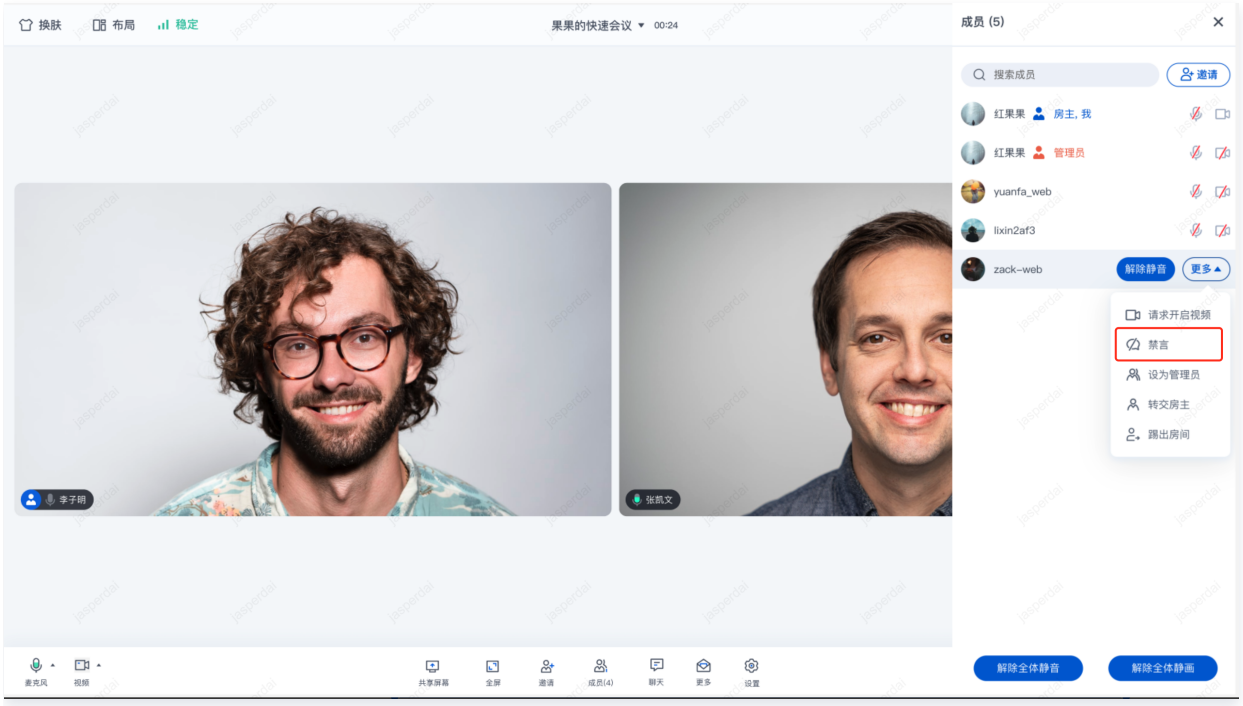
点击聊天界面编辑消息栏中的**表情图标**，出现表情列表，点击对应表情即可进行发送，为演讲人的精彩发言点赞。



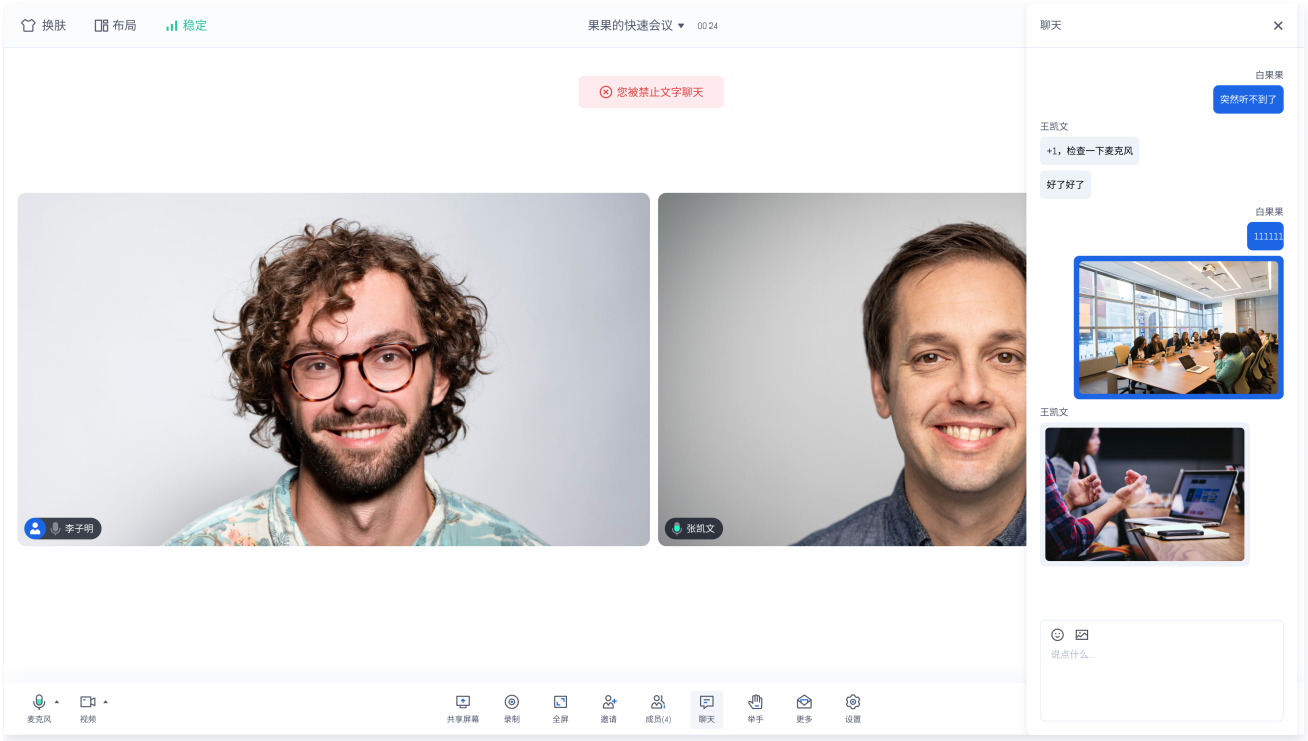
> **说明：**
> 为尊重表情设计版权，TUIRoomKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。
> 默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

管控会中聊天权限

主持人/管理员可在成员管理中设置某一成员的聊天权限。



普通成员受到禁止发言，无法进行消息的编辑和发送。



功能接入

目前 Web 和 Electron 的 TUIRoomKit 内部已经集成会中的聊天能力，您可以单击 Github 下载 TUIRoomKit 代码，并参见代码仓库 README.md 文档跑通 TUIRoomKit Web 示例工程。

> **注意：**
> TUIRoomKit Web&Electron 自 2.6.3 版本起支持图片、视频、文件消息，2.6.3 版本之前仅支持文字消息。
> TUIRoomKit 目前聊天组件接入了 TUIChat 组件，如果需要更多能力请查看 相关文档。

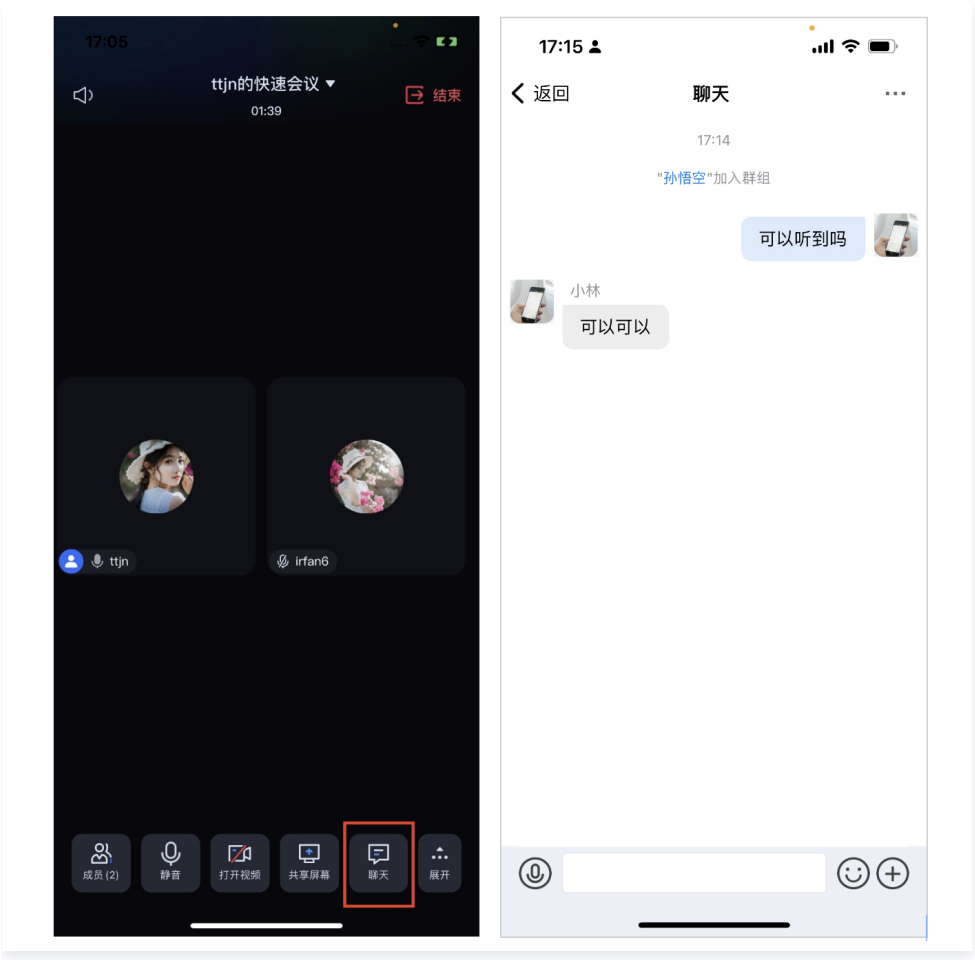
Android&iOS

最近更新时间：2024-09-27 16:18:21

在视频会议中，参会者可以实时在聊天区发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

功能介绍

在会议界面底部菜单栏中，您可以找到一个名为**聊天**的选项。点击它，会跳转到聊天页面。在聊天页面，参会人员可以通过发送自定义文字信息与其他参会者进行沟通。这样的设计既方便了参会者之间的实时交流，又不会影响到正在发言的人。



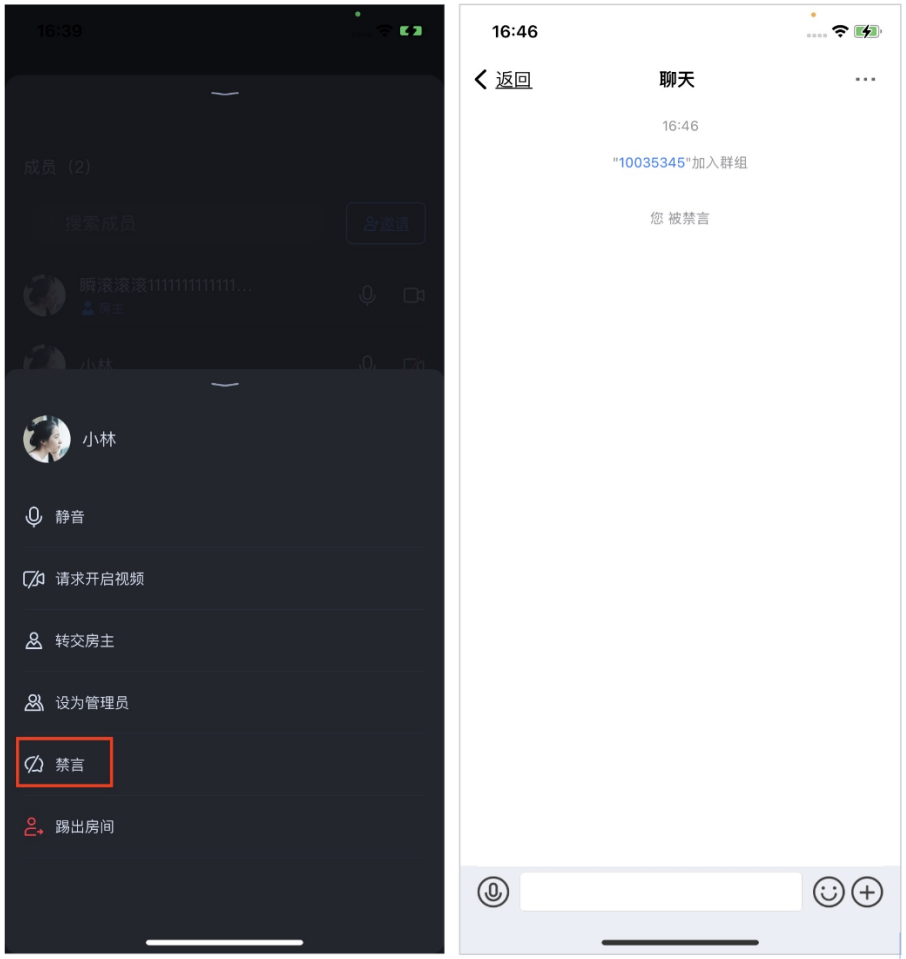
表情互动

点击聊天界面编辑消息栏中的 **表情图标**，出现表情列表，点击对应表情即可进行发送，为演讲人的精彩发言点赞。

> **说明：**
> 为尊重表情设计版权，TUIRoomKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。
> 默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

管控会中聊天权限

主持人/管理员可在成员管理中设置某一成员的聊天权限。如果被禁言，普通成员无法进行文字或表情的编辑和发送。



功能接入

Android

module 源码集成

1. 在 Github 中克隆/下载代码，然后拷贝Android目录下的tuichat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tuichat作为本地模块导入到您当前的项目中。

```
include ':tuichat'
```

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tuichat组件的依赖。

```
api project(':tuichat')
```

4. 添加 maven 仓库 和 Kotlin 支持，在 root 工程的 build.gradle 文件（与 settings.gradle 同级）中添加：

```
buildscript {
  repositories {
      mavenCentral()
      maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
```

```
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.0.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"
    }
}
```

如果您使用 Gradle 8.x，则需要添加以下代码。

```
buildscript {
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:8.0.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"
    }
}
```

说明：
Kotlin 、Gradle 和 AGP 的版本对应关系可以 在此查看 。

iOS

接入聊天挂件

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
```

Flutter

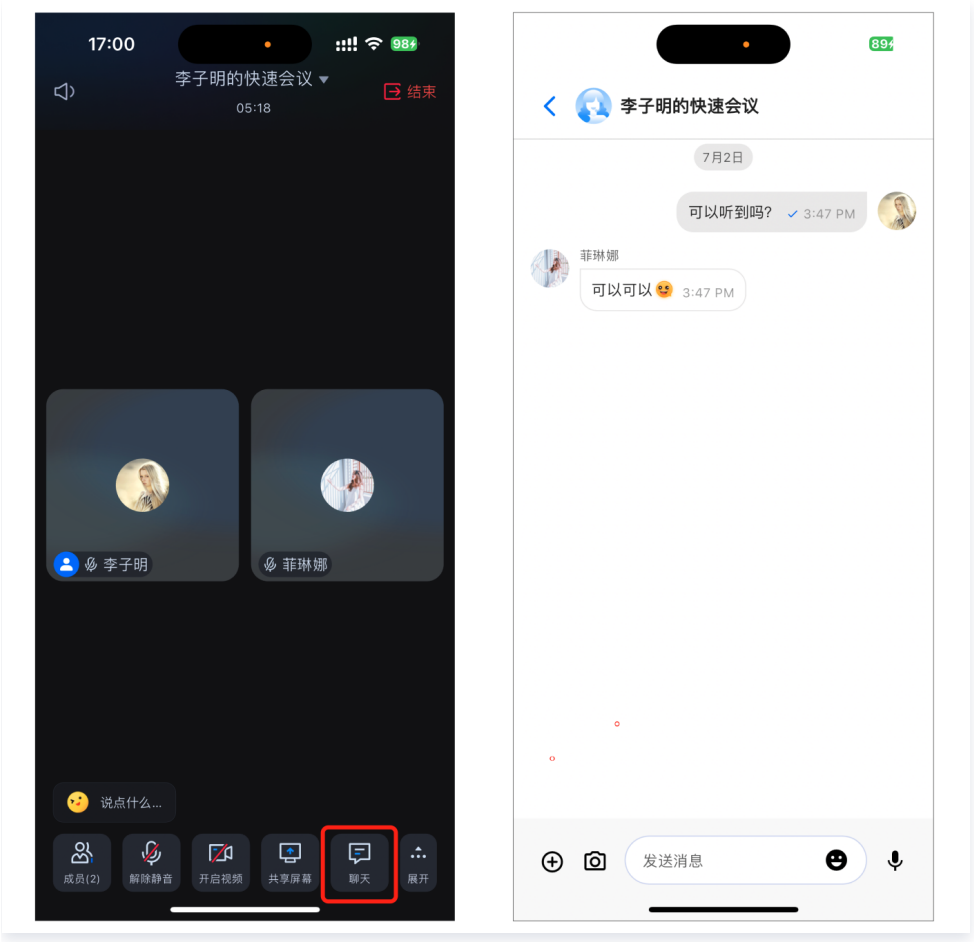
最近更新时间：2024-07-29 11:14:21

在视频会议中，参会者可以实时在聊天页面发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

功能介绍

文字、多媒体信息互动

单击会议界面底部**聊天**选项即可到达聊天页面。参会者可自由发送**文字**、**图片**、**视频**、**语音**，实时沟通无阻，且不影响会议发言进程。



表情互动

单击聊天界面编辑消息栏中的**表情图标**，出现表情列表，单击对应表情即可出现在消息栏中进行发送。

管控会中聊天权限

主持人/管理员可在**成员管理**中设置某一成员的聊天权限。如果被禁言，普通成员将无法进行消息的发送。

功能接入

集成聊天组件

在您的工程 `pubspec.yaml` 文件中，添加 tencent_cloud_chat_message 插件依赖。

```
dependencies:
    tencent_cloud_chat_message: 最新版本
```

国际化语言配置

此步骤为必需配置。首先，将本地化工具导入到应用程序的入口文件中。

```
import 'package:tencent_cloud_chat_intl/localizations/tencent_cloud_chat_localizations.dart';
```

接下来，将本地化配置添加到 `MaterialApp` 或 `GetMaterialApp` 等第三方包提供的其他条目中，这里以 `GetMaterialApp` 为例：

```
GetMaterialApp(
  localizationsDelegates: const [
    /// 您原先的配置
    GlobalMaterialLocalizations.delegate,

    /// 添加此行
    ...TencentCloudChatLocalizations.localizationsDelegates,
  ],
```

```
  supportedLocales: [
    /// 您原先的配置
    ...S.delegate.supportedLocales,

    /// 添加此行
    ...TencentCloudChatLocalizations.supportedLocales,
  ],
  /// 其他配置
)
```

初始化和登录

在您的项目中添加如下代码，它的作用是通过调用聊天中的相关接口完成组件的初始化及登录。这个步骤非常关键，因为只有在初始化后才能正常使用聊天，故请您耐心检查相关参数是否配置正确。其中的参数 **sdkAppId**、**userID** 和 **userSig** 您已在 登录TUIRoomKit 时使用过。

```
import
'package:tencent_cloud_chat/components/component_config/tencent_cloud_chat_message_common_def
ines.dart';
import
'package:tencent_cloud_chat/components/component_config/tencent_cloud_chat_message_config.dar
t';
import 'package:tencent_cloud_chat/models/tencent_cloud_chat_models.dart';
import 'package:tencent_cloud_chat/tencent_cloud_chat.dart';
import 'package:tencent_cloud_chat_message/tencent_cloud_chat_message.dart';

await TencentCloudChat.controller.initUIKit(
    options: TencentCloudChatInitOptions(
      sdkAppID: 'SDKAPPID',  // 您的SDKAPPID
      userID: 'userID',      // 您的userID
      userSig: 'userSig',    // 您的UserSig
    ),
    components: TencentCloudChatInitComponentsRelated(
      usedComponentsRegister: [TencentCloudChatMessageManager.register],  // 注册聊天组件
      componentConfigs: TencentCloudChatComponentConfigs(
        messageConfig: TencentCloudChatMessageConfig(
          // 以下代码中的配置为建议配置。
          showMessageSenderName: ({groupID, topicID, userID}) => true,
          showSelfAvatar: ({groupID, topicID, userID}) => true,
          defaultMessageMenuConfig: ({groupID, topicID, userID}) =>
              TencentCloudChatMessageDefaultMessageMenuConfig(
            enableMessageForward: false,
            enableMessageSelect: false,
          ),
        ),
      ),
    ),
    plugins: [],
);
```

使用表情（可选）

如您需要使用发送和接收表情，您需要进行如下配置：

添加依赖

在您工程的 `pubspec.yaml` 文件中，添加 tencent_cloud_chat_sticker 插件依赖。

完成配置

在上一步 初始化和登录 的 `plugins` 中，添加如下代码：

```
plugins: [
  TencentCloudChatPluginItem(
    name: "sticker",
    initData: TencentCloudChatStickerInitData(
      useDefaultSticker: true,             // 默认表情，仅此表情包可与TUIRoomKit其他端互通。
      useDefaultCustomFace_4350: false,    // 如您无需使用其他端TUIRoomKit，可启用下列表情包。更多详情请查看下方说明。
      useDefaultCustomFace_4351: false,
      useDefaultCustomFace_4352: false,
      userID: 'userId',                    // 您的userId
    ).toJson(),
    pluginInstance: TencentCloudChatStickerPlugin(
      context: context,
    ),
  ),
],
```

> **说明：**
> 为尊重表情设计版权，TUIRoomKit example 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

使用聊天

当您 创建会议或加入会议成功 时，您需要向会议界面 `ConferenceMainPage` 中传入 `chatWidget` 。传入后将显示底部工具栏中的聊天按钮，点击聊天按钮后自动导航到聊天页面。

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ConferenceMainPage( // 会议主界面
      chatWidget: TencentCloudChatMessage(
          options: TencentCloudChatMessageOptions(groupID: 'yourConferenceId'),  //您的ConferenceId
      ),
    ),
  ),
);
```

完成以上配置后，您便可以在会议中点击聊天按钮进行聊天交流。

uniapp（客户端）

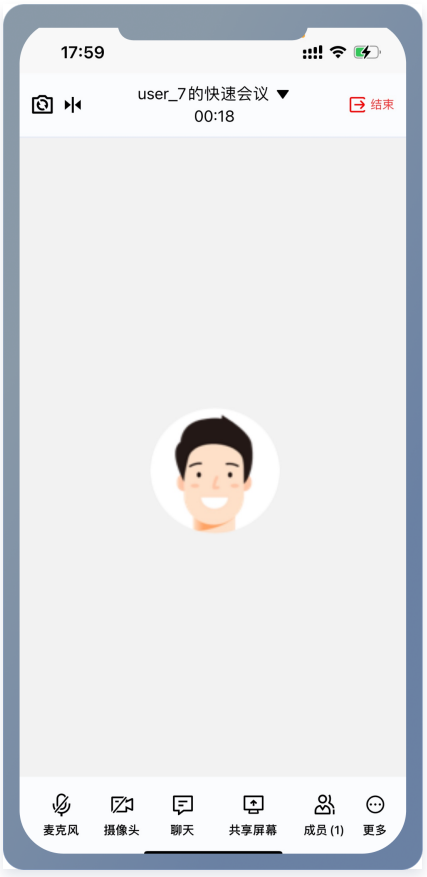
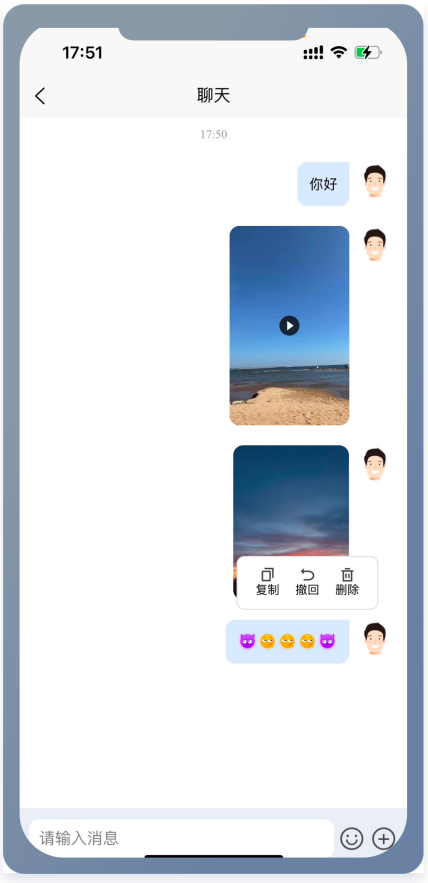
最近更新时间：2025-01-16 14:47:02

在视频会议中，参会者可以实时在聊天区发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

功能介绍

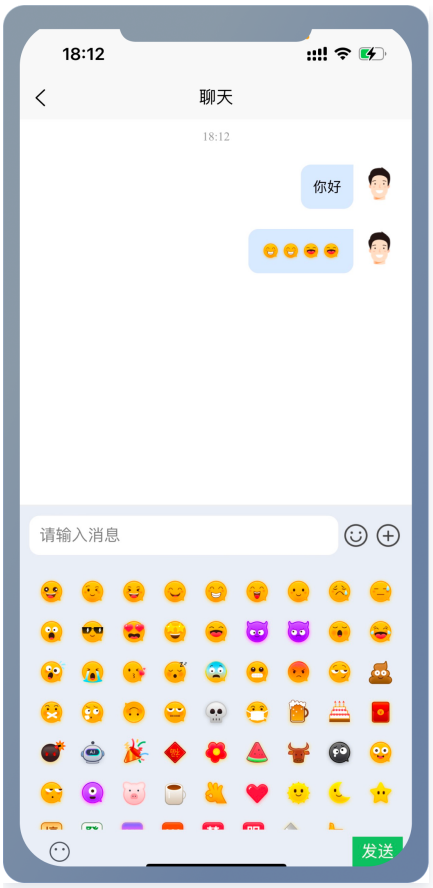
聊天互动

在会议界面底部菜单栏中，您可以找到一个名为**聊天**的按钮。点击后进入聊天界面。在聊天框中，参会人员可以通过参会人员可以通过发送文字、图片信息与与其他参会者进行沟通。这样的设计既方便了参会者之间的实时交流，又不会影响到正在发言的人。

聊天入口	聊天界面
	

表情互动

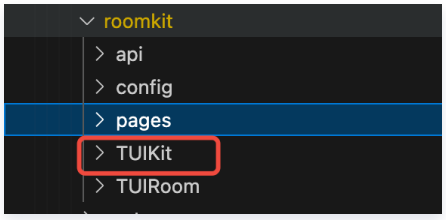
单击聊天界面编辑消息栏中的**表情图标**，出现表情列表，单击对应表情即可出现在消息栏中进行发送。



说明：
为尊重表情设计版权，TUIRoomKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。
默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

功能接入

目前 uniapp 内部已经集成会中的聊天能力，您可以单击 Github 下载 TUIRoomKit 代码，并参见代码仓库 README.md 文档跑通 TUIRoomKit uniapp示例工程。
若您是参见 快速接入 文档，接入自己已有项目里面的话，请参见下面指引进行使用。
1. 前往 Github 将聊天模块代码拷贝到自己项目里面，建议与 TUIRoom 放在平级。



2. 配置聊天页面路径

配置 pages.json

```
{
  "pages": [
    //pages数组中第一项表示应用启动页，参考：https://uniapp.dcloud.io/collocation/pag
    {
      "path": "pages/index/index",
      "style": {
        "navigationBarTitleText": ""
      }
    },
    {
      "path": "roomkit/TUIKit/components/TUIChat/index",
      "style": {
        "navigationBarTitleText": "聊天"
      }
    },
    {
      "path": "roomkit/TUIKit/components/TUIChat/video-play",
      "style": {
        "navigationBarTitleText": "腾讯云 IM"
      }
    }
  ],
```

3. 初始化聊天

在 App.vue 中，新增如下代码。

```
  <script lang="ts">
  import { TUILogin } from '@tencentcloud/tui-core';
  import { roomChatInit } from '../TUIKit';// 修改为自己的 TUIKit 的相对路径

    TUILogin.login({
        SDKAppID: '', // Your SDKAppID
        userID: '', // Your userID
        userSig: '', // Your userSig
        useUploadPlugin: true, // If you need to send rich media messages, please set to
true.
      });
  roomChatInit();
}
</script>
```

4. 在 RoomKit 中配置跳转聊天路径

在 TUIRoom 组件中，`TUIRoom/components/RoomFooter/ChatControl.vue` 文件中，修改为自己在 步骤2 中配置的路径。

```
  uni.navigateTo({ url: `/src/roomkit/TUIKit/components/TUIChat/index?
conversationID=${conversationID}` });
```

常见问题

1. 若需要使用视频播放能力，需要在 App 模块配置中勾选视频播放模块。

manifest.json | uniapp-app-vue3
package.json | uniapp-app.....RoomEngine
ios-extension.json
manifest.json | roomkit-test-dem
基础配置
App图标配置
App启动界面配置
App模块配置
App权限配置
App原生插件配置
App常用其它设置
Web配置
微信小程序配置
百度小程序配置
使用指南
Share(分享)
分享配置指南 分享功能依赖三方SDK，上架到国内应用市场需要在隐私协议中添加相应的条款 [详情]
Statistic(统计，只能选一个)
推荐使用uni-app自带的uni统计，无需勾选此模块，详情
统计配置指南 统计功能依赖三方SDK，上架到国内应用市场需要在隐私协议中添加相应的条款 [详情]
SQLite(数据库)
VideoPlayer(视频播放)
Android X5 Webview(腾讯TBS)
详情 X5功能依赖腾讯TBS SDK，上架到国内应用市场需要在隐私协议中添加相应的条款 [详情]
iOS UIWebview
详情
uni-ad
使用指南

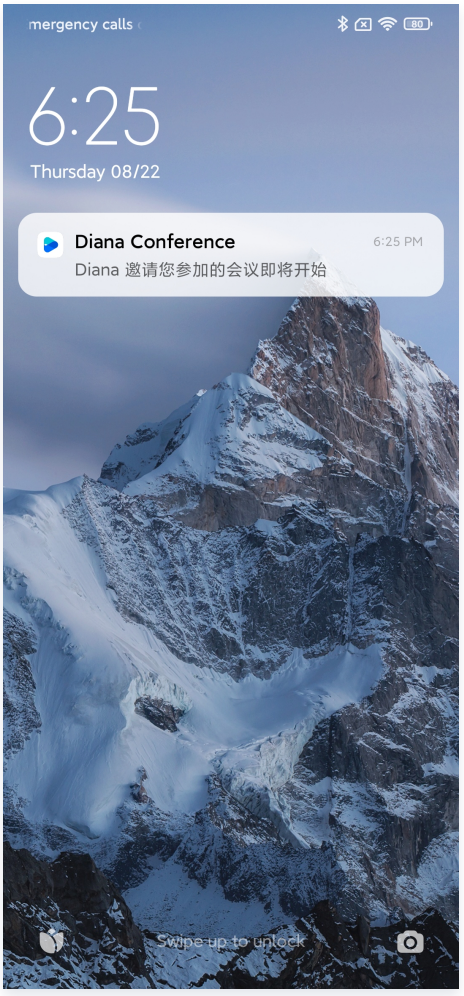
离线唤醒
会前提醒（Android/iOS）

最近更新时间：2025-03-07 12:06:33

当您成功预定会议后，TUIRoomKit 组件将在会议开始前10分钟向参会成员推送会前提醒消息。本文档将指引您接入 TIMPush 推送插件，以开启会前提醒功能。

功能介绍

- 在进行离线推送功能的集成前，**请确保已按照官方文档完成 厂商配置，以确保 TUIRoomKit 的离线推送功能正常运行。**
- TUIRoomKit 接入 TIMPush 推送插件的效果如下（以小米手机 Redmi Note 8 Pro 的显示效果为例）：

应用在后台时或离线时	锁屏时
	

功能接入

Android

1. 请参见推送插件 TIMPush 快速接入文档，完成除步骤6以外的所有步骤（TUIRoomKit 组件内部已经进行会前提醒的离线消息推送，所以步骤6不需要单独配置）。

2. (可选) 若您想实现点击通知立即进房，可参考以下代码，注册回调时机建议放在应用 Application 的 onCreate() 函数中：

```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY,
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION, new ITUINotification() {
    @Override
    public void onNotifyEvent(String key, String subKey, Map<String, Object> param) {
        if (TUIConstants.TIMPush.EVENT_NOTIFY.equals(key) &&
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION.equals(subKey)) {
            if (param != null) {
                String extString = (String)
param.get(TUIConstants.TIMPush.NOTIFICATION_EXT_KEY);
                try {
                    // 可根据您的业务，点击通知跳转到相应的界面，以下是跳转到房间的示例。
                    JSONObject roomObject = new JSONObject(extString);
                    String roomId = roomObject.getString("RoomId");
                    if (!TextUtils.isEmpty(roomId)) {
                        loginAndEnterRoom(roomId);
                    }
                } catch (Exception e) {
                }
            }
        }
    }
});

private void loginAndEnterRoom(String roomId) {
   int sdkAppId = "您的appId";
   String userId = "您的UserId"
   String userSig = GenerateTestUserSig.genTestUserSig(userId);

   // 登录
   TUILogin.login(this.getApplicationContext(), sdkAppId, userId, userSig, new
TUICallback() {
       @Override
       public void onSuccess() {
           // 登录成功后进入房间
           ConferenceDefine.JoinConferenceParams params = new
ConferenceDefine.JoinConferenceParams(roomId);
           Intent intent = new Intent(getApplicationContext(),
ConferenceMainActivity.class);
           intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
           intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
           getApplicationContext().startActivity(intent);
       }

       @Override
       public void onError(int errorCode, String errorMessage) {
       }
   });
  }
```

⚠ 注意：

> 若您想实现点击推送进入房间，**必须进房前完成登录**。

iOS

1. 集成 TIMPush 组件

```
pod 'TIMPush', '8.1.6108'
```

2. 配置推送参数

完成 厂商配置 后，可以在 即时通信 IM 控制台 得到证书 ID。您需要在 AppDelegate 中，实现 `offlinePushCertificateID` 协议方法并返回证书 ID 。

Swift

```
import TIMPush

extension AppDelegate: TIMPushDelegate {
    func offlinePushCertificateID() -> Int32 {
        return kAPNSBusiId
    }
}
```

OC

```
#import "TIMPush/TIMPushManager.h"

@interface AppDelegate () <TIMPushDelegate>

- (int)offlinePushCertificateID {
    return kAPNSBusiId;
}
```

3. 点击离线推送后自定义跳转（可选）

在默认情况下，点击通知会跳转到 app。您可以参考以下代码实现点击通知立即进入会议，也可以查看 github 中的 SceneDelegate 和 AppDelegate 文件。
如果是冷启动，需要在 SceneDelegate 中解析通知消息，得到要进入会议的 roomId。

Swift

```
import UIKit

class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?

    func scene(_ scene: UIScene,
               willConnectTo session: UISceneSession,
               options connectionOptions: UIScene.ConnectionOptions) {
```

```
        guard let windowScene = (scene as? UIWindowScene) else { return }
        window = UIWindow(windowScene: windowScene)
        let loginVC = TRTCLoginViewController() //您自己的登录页面
        loginVC.roomId = processOfflinePush(connectionOptions: connectionOptions)
        let nav = UINavigationController(rootViewController: loginVC)
        window?.rootViewController = nav
        window?.makeKeyAndVisible()
    }

    private func processOfflinePush(connectionOptions: UIScene.ConnectionOptions) ->
String? {
        guard let pushNotification =
connectionOptions.notificationResponse?.notification.request.content.userInfo else {
return nil }
        guard let extString = pushNotification["ext"] as? String else { return nil }
        guard let dict = extString.convertToDic() else { return nil }
        return dict["RoomId"] as? String
    }
```

OC

```
#import "SceneDelegate.h"
#import <UserNotifications/UserNotifications.h>
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface SceneDelegate ()
@property (nonatomic, strong) NSString *roomId;
@end

@implementation SceneDelegate

- (void)scene:(UIScene *)scene willConnectToSession:(UISceneSession *)session options:
(UISceneConnectionOptions *)connectionOptions {
    [self processOfflinePush:connectionOptions];
}

- (void)processOfflinePush: (UISceneConnectionOptions *)connectionOptions {
    NSDictionary *pushNotification =
connectionOptions.notificationResponse.notification.request.content.userInfo;
    NSString *notice = pushNotification[@"ext"];
    NSDictionary *dic = [self dictionaryFromString:notice];
    NSString *roomId = dic[@"RoomId"];
    //将roomId传给您自己的登录页面YourLoginViewController
}

@end
```

在您的登录页面完成 TUICore 的登录，并且判断是否需要跳转到会议主界面。

Swift

```
import TUICore
import TUIRoomKit

//YourLoginViewController 是您自己的登录页面
class YourLoginViewController: UIViewController {
    var roomId: String?

    override func viewDidLoad() {
        super.viewDidLoad()
        TUILogin.login(Int32(SDKAppID), userID: "yourUserName", userSig: "yourUserSig") {
[weak self] in
            guard let self = self else { return }
            self.navigationController?.pushViewController(YourSelfViewController(),
animated: false)
            //YourSelfViewController是正常登录后应该显示的您自己的界面
            guard let roomId = self.roomId else { return }
            //通过roomId判断是否通过离线通知进入前台，由此判断是否再跳转到会议主界面
            self.showConferenceMainViewController(roomId: roomId)
            self.roomId = nil
        } fail: { (code, errorDes) in
            print("code:\(code), errorDes:\(String(describing: errorDes))")
        }
    }

    func showConferenceMainViewController(roomId: String) {
        let conferenceViewController = ConferenceMainViewController()
        let params = JoinConferenceParams(roomId: roomId)
        conferenceViewController.setJoinConferenceParams(params: params)
        navigationController?.pushViewController(conferenceViewController, animated:
true)
    }
}
```

OC

```
#import "YourLoginViewController.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TUILogin.h"

@interface YourLoginViewController ()
@property (nonatomic, strong) NSString *roomId;
@end

@implementation YourLoginViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [TUILogin login:yourSDKAPPID userID:@"youruserID" userSig:@"yourUserSig" succ:^{
        //先显示您自己的界面
        //如果roomId有值，说明是从离线推送进来的，可以调用showConferenceMainViewController再跳转
到会议主界面
```

```
    } fail:^(int code, NSString * _Nullable msg) {
    }];
    // Do any additional setup after loading the view.
}

- (void)showConferenceMainViewController: (NSString *)roomId {
    ConferenceMainViewController * vc = [[ConferenceMainViewController alloc]init];
    JoinConferenceParams * params = [[JoinConferenceParams alloc]initWithRoomId:
                                     roomId isOpenMicrophone:true isOpenCamera:false
isOpenSpeaker:true];
    [vc setJoinConferenceParamsWithParams:params];
    [self.navigationController pushViewController:vc animated:true];
}
```

如果是从后台进入前台，需要在 AppDelegate文件中实现 `onRemoteNotificationReceived` 方法。

Swift

```
import TUIRoomKit
import TIMPush

@main
class AppDelegate: UIResponder, UIApplicationDelegate, TIMPushDelegate {
    var roomId: String?
    func onRemoteNotificationReceived(_ notice: String?) -> Bool {
        guard let notice = notice else { return false }
        guard let dict = convertToDic(string: notice) else { return false }
        guard let roomId = dict["RoomId"] as? String else { return false }
        if V2TIMManager.sharedInstance().getLoginStatus() == .STATUS_LOGINED {
            //如果从后台唤醒并且已经完成登录，可以直接进入会议。
            showConferenceMainViewController(roomId: roomId)
        }
        return true
    }
}
```

OC

```
#import "AppDelegate.h"
#import "TIMPush/TIMPushManager.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface AppDelegate ()<TIMPushDelegate>
@property (nonatomic, strong) NSString *roomId;
@end
@implementation AppDelegate

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    NSDictionary * dic = [self dictionaryFromString:notice];
    NSString * roomId = dic[@"RoomId"];
    if (!roomId) {
```

```
        return false;
    }
    if ([V2TIMManager sharedInstance].getLoginStatus == V2TIM_STATUS_LOGINED ) {
        //如果从后台唤醒并且已经完成登录，可以直接进入会议。
       [self showConferenceMainViewController:roomId];
    }
    return true;
}

@end
```

常见问题

1. 若在集成过程中遇到问题，请务必先查阅 插件推送-常见问题 进行自助排查。
2. 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
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
荣耀	否	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

如有问题，欢迎您加入我们的 TUIRoomKit 技术交流平台 zhiliao 进行反馈。

功能定制

如果您想自定义会前提醒时间，您可以通过修改源码来实现，不同平台请参考：

Android

您可通过修改 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/ScheduleConference/view/ScheduleConferenceView.java 中 `REMINDER_TIME` 成员变量的值，设置会前提醒的时间（秒）。

iOS

您可通过修改 iOS/TUIRoomKit/Source/ScheduleConferenceViewController.swift 中 **reminderSecondsBeforeStart** 成员变量的值，设置会前提醒的时间（秒）。

> ⚠ **注意：**
> 若会前提醒的时间大于会议预定的时间，则不会推送会前提醒的通知。

> **说明：**
> 若您还有其他问题，可通过 TUIRoomKit 技术交流平台 zhiliao，向我们反馈。

关键代码

会前提醒时间的设置，不同平台可参考：

Android

```
public abstract void scheduleConference(ConferenceInfo conferenceInfo,
TUIRoomDefine.ActionCallback callback);
```

您可以通过修改 conferenceInfo 的 reminderSecondsBeforeStart 字段设置会前提醒时间（秒），详情请参考 scheduleConference 。示例代码如下：

```
TUIConferenceListManager manager = (TUIConferenceListManager)
TUIRoomEngine.sharedInstance().getExtension(CONFERENCE_LIST_MANAGER);
TUIConferenceListManager.ConferenceInfo conferenceInfo = new
TUIConferenceListManager.ConferenceInfo();
conferenceInfo.basicRoomInfo.roomId = "您的roomId";
conferenceInfo.reminderSecondsBeforeStart = 600; // 请替换600为您的会前提醒时间
manager.scheduleConference(conferenceInfo, new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
      // 预定会议成功回调
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
      // 预定会议失败回调
    }
});
```

iOS

```
- (void)scheduleConference:(TUIConferenceInfo *)conferenceInfo onSuccess:
(TUISuccessBlock)onSuccess onError:(TUIErrorBlock)onError
NS_SWIFT_NAME(scheduleConference(_:onSuccess:onError:));
```

您可以通过修改 conferenceInfo 的 reminderSecondsBeforeStart 字段设置会前提醒时间（秒），详情请参考 scheduleConference 。示例代码如下：

Swift

```
import RTCRoomEngine
```

```
func scheduleConference() {
    let manager = TUIRoomEngine.sharedInstance().getExtension(extensionType:
.conferenceListManager) as? TUIConferenceListManager
    let conferenceInfo = TUIConferenceInfo()
    conferenceInfo.basicRoomInfo.roomId = "111111"    // 请替换 "111111" 为您自定义的房间号
    conferenceInfo.reminderSecondsBeforeStart = 600  // 请替换 600 为您自定义的会前提醒时间
    manager?.scheduleConference(conferenceInfo, onSuccess: {
        print("ScheduleConference success")
    }, onError: { code, message in
        print("ScheduleConference error, code:\(code), message:\(message)")
    })
}
```

OC

```
#import "RTCRoomEngine/TUIRoomEngine.h"
#import "RTCRoomEngine/TUIConferenceListManager.h"

- (void)scheduleConference {
    TUIConferenceListManager * manager = [[TUIRoomEngine sharedInstance] getExtension:
TUIExtensionTypeConferenceListManager];
    TUIConferenceInfo * conferenceInfo = [[TUIConferenceInfo alloc] init];
    conferenceInfo.basicRoomInfo.roomId = @"111111";  // 请替换 @"111111" 为您自定义的房间号
    conferenceInfo.reminderSecondsBeforeStart = 600;  //请替换 600 为您自定义的会前提醒时间
    [manager scheduleConference:conferenceInfo onSuccess:^{
        NSLog(@"ScheduleConference success");
    } onError:^(TUIError code, NSString * _Nonnull message) {
        NSLog(@"ScheduleConference failed, code:%ld, message:%@", (long)code, message);
    }];
}
```

会中呼叫提醒（Android/iOS）

最近更新时间：2025-03-07 12:06:33

当您收到一条会中消息后，TUIRoomKit 组件会向您推送会中呼叫提醒消息。本文档将指引您接入 TIMPush 推送插件，以开启会中呼叫提醒功能。

功能介绍

- 在进行离线推送功能的集成前，**请确保已按照官方文档完成 厂商配置，以确保 TUIRoomKit 的离线推送功能正常运行。**
- TUIRoomKit 接入 TIMPush 推送插件后收到会中呼叫提醒消息的效果如下：

应用在后台时或离线时	锁屏时

功能接入

Android

1. 请参见推送插件 TIMPush 快速接入文档，完成除步骤6以外的所有步骤（TUIRoomKit 组件内部已经进行会中呼叫的离线消息推送，所以步骤6不需要单独配置）。

2. (可选) 若您想实现点击通知立即拉起被邀请页面，可参考以下代码，注册回调时机建议放在应用 Application 的 onCreate() 函数中：

```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY,
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION, new ITUINotification() {
    @Override
    public void onNotifyEvent(String key, String subKey, Map<String, Object> param) {
        if (TUIConstants.TIMPush.EVENT_NOTIFY.equals(key) &&
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION.equals(subKey) && param != null) {
            String extString = (String)
param.get(TUIConstants.TIMPush.NOTIFICATION_EXT_KEY);
            try {
                JSONObject roomObject = new JSONObject(extString);
                String notificationType = roomObject.getString("NotificationType");
                if ("conference_invitation".equals(notificationType)) {
                    login(new TUICallback() {
                        @Override
                        public void onSuccess() {

                        }

                        @Override
                        public void onError(int errorCode, String errorMessage) {

                        }
                    });
                }
            } catch (Exception e) {
            }
        }
    }
});

private void login(TUICallback callback) {
    int sdkAppId = 您的sdkappid;
    String userId = "您的userId";
    String userSig = "您的userSig";
    TUILogin.login(this.getApplicationContext(), sdkAppId, userId, userSig, new
TUICallback() {
        @Override
        public void onSuccess() {
            if (callback != null) {
                callback.onSuccess();
            }
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
            if (callback != null) {
                callback.onError(errorCode, errorMessage);
            }
        }
    });
}
```

iOS

1. 集成 TIMPush 组件

```
pod 'TIMPush', '8.1.6108'
```

2. 配置推送参数

完成 厂商配置 后，可以在 即时通信 IM 控制台 得到证书 ID。您需要在 AppDelegate 中，实现 `offlinePushCertificateID` 协议方法并返回证书 ID 。

Swift

```
import TIMPush

extension AppDelegate: TIMPushDelegate {
    func offlinePushCertificateID() -> Int32 {
        return kAPNSBusiId
    }
}
```

OC

```
#import "TIMPush/TIMPushManager.h"

@interface AppDelegate () <TIMPushDelegate>

- (int)offlinePushCertificateID {
    return kAPNSBusiId;
}
```

3. 点击离线推送后拉起被呼叫界面

在默认情况下，点击通知会跳转到 app。您可以参考以下代码实现点击通知立即拉起被呼叫界面，也可以查看 github 中的 SceneDelegate 和 AppDelegate 文件。
如果是冷启动，您需要在 SceneDelegate 中解析通知消息，得到推送消息的 extString。

Swift

```
import UIKit

class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?

    func scene(_ scene: UIScene,
               willConnectTo session: UISceneSession,
               options connectionOptions: UIScene.ConnectionOptions) {
```

```
        guard let windowScene = (scene as? UIWindowScene) else { return }
        window = UIWindow(windowScene: windowScene)
        let loginVC = YourLoginViewController() // 您自己的登录页面
        loginVC.extString = processOfflinePush(connectionOptions: connectionOptions)
        let nav = UINavigationController(rootViewController: loginVC)
        window?.rootViewController = nav
        window?.makeKeyAndVisible()
    }

    private func processOfflinePush(connectionOptions: UIScene.ConnectionOptions) ->
String? {
        guard let pushNotification =
connectionOptions.notificationResponse?.notification.request.content.userInfo else {
return nil }
        guard let extString = pushNotification["ext"] as? String else { return nil }
        return extString
    }
```

OC

```
#import "SceneDelegate.h"
#import <UserNotifications/UserNotifications.h>
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface SceneDelegate ()
@end

@implementation SceneDelegate

- (void)scene:(UIScene *)scene willConnectToSession:(UISceneSession *)session options:
(UISceneConnectionOptions *)connectionOptions {
    [self processOfflinePush:connectionOptions];
}

- (void)processOfflinePush: (UISceneConnectionOptions *)connectionOptions {
    NSDictionary *pushNotification =
connectionOptions.notificationResponse.notification.request.content.userInfo;
    NSString *extString = pushNotification[@"ext"];
    //将extString传给您自己的登录页面YourLoginViewController
}

@end
```

在您的登录页面完成 TUICore 的登录，并且判断是否需要跳转到被呼叫界面。

Swift

```
import TUICore
import TUIRoomKit

//YourLoginViewController 是您自己的登录页面
```

```
class YourLoginViewController: UIViewController {
    var extString: String?

    override func viewDidLoad() {
        super.viewDidLoad()
        TUILogin.login(Int32(SDKAppID), userID: "yourUserName", userSig: "yourUserSig") {
[weak self] in
            guard let self = self else { return }
            self.navigationController?.pushViewController(YourSelfViewController(),
animated: false)
            // YourSelfViewController是正常登录后应该显示的您自己的界面
            guard let extString = self.extString else { return }
            guard let notificationType = dict["NotificationType"] as? String else {
return }
            // 通过notificationType判断是否为会中呼叫提醒，由此判断是否再拉起被呼叫页面
            if notificationType == "conference_invitation" {
                InvitationObserverService.shared.show(extString: extString)  // 调用此方法
以拉起被呼叫界面
            }
            self.extString = nil
        } fail: { (code, errorDes) in
            print("code:\(code), errorDes:\(String(describing: errorDes))")
        }
    }
}
```

OC

```
#import "YourLoginViewController.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TUILogin.h"

@interface YourLoginViewController ()
@property (nonatomic, strong) NSString *extString;
@end

@implementation YourLoginViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [TUILogin login:yourSDKAPPID userID:@"youruserID" userSig:@"yourUserSig" succ:^{
        //先显示您自己的界面
        //如果extString有值，说明是从离线推送进来的，可以调用showInitationView拉起被呼叫界面
    } fail:^(int code, NSString * _Nullable msg) {
    }];
    // Do any additional setup after loading the view.
}

- (void)showInitationView: (NSString *)extString {
    [[InvitationObserverService shared] show:extString];
}
```

如果是从后台进入前台，需要在 AppDelegate文件中实现 onRemoteNotificationReceived 方法。

Swift

```
import TUIRoomKit
import TIMPush


@main
class AppDelegate: UIResponder, UIApplicationDelegate, TIMPushDelegate {
    var roomId: String?
    func onRemoteNotificationReceived(_ notice: String?) -> Bool {
        guard let notice = notice else { return false }
        guard let dict = convertToDic(string: notice) else { return false }
        guard let roomId = dict["RoomId"] as? String else { return false }
        if V2TIMManager.sharedInstance().getLoginStatus() == .STATUS_LOGINED {
            InvitationObserverService.shared.show(extString: extString)  // 调用此方法以拉起被呼叫界面
        }
        return true
    }
}
```

OC

```
#import "AppDelegate.h"
#import "TIMPush/TIMPushManager.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface AppDelegate ()<TIMPushDelegate>
@property (nonatomic, strong) NSString *roomId;
@end
@implementation AppDelegate

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    NSDictionary * dic = [self dictionaryFromString:notice];
    NSString * roomId = dic[@"RoomId"];
    if (!roomId) {
        return false;
    }
    if ([V2TIMManager sharedInstance].getLoginStatus == V2TIM_STATUS_LOGINED ) {
       [[InvitationObserverService shared] show:extString];
    }
    return true;
}

@end
```

常见问题

1. 若在集成过程中遇到问题，请务必先查阅 插件推送-常见问题 进行自助排查。
2. 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
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
荣耀	否	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

功能定制

如您需要自定义被呼叫页面的视图，请参考以下路径进行更改：

Android

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/component/

component
    └──InvitationReceivedView.java
```

iOS

```
// 文件位置：iOS/TUIRoomKit/Source/View/ConferenceOptions/ConferenceInvitation

ConferenceInvitation
    └── ConferenceInvitationViewController.swift  // 被呼叫页面视图
```

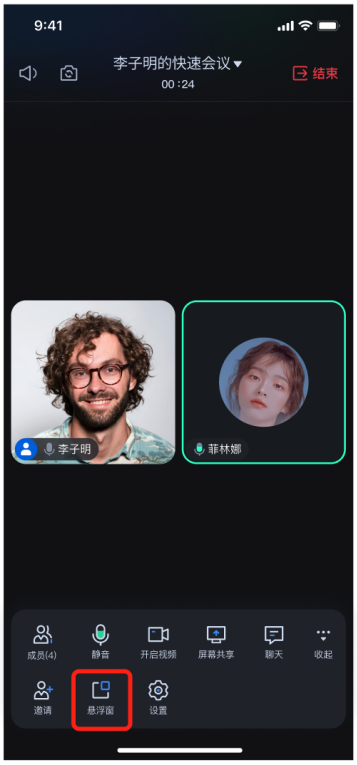
悬浮窗（Android&iOS）

最近更新时间：2024-09-27 16:18:21

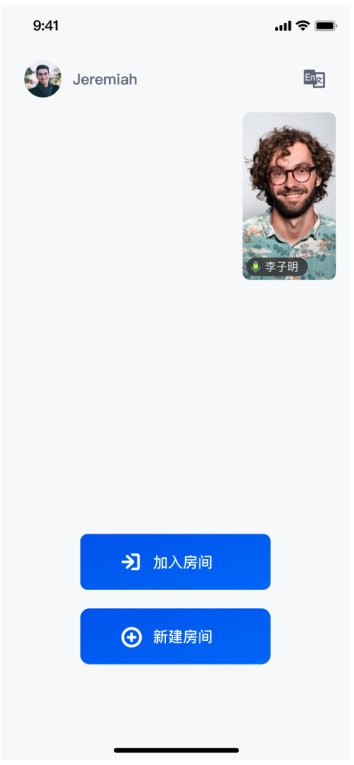
功能介绍

TUIRoomKit 支持悬浮窗功能，允许用户创建一个可自由拖动的悬浮窗口，用于展示和管理 TUIRoomKit 的视频会议界面，使用户在参与视频会议时能够更轻松地同时处理其他任务。

悬浮窗按钮



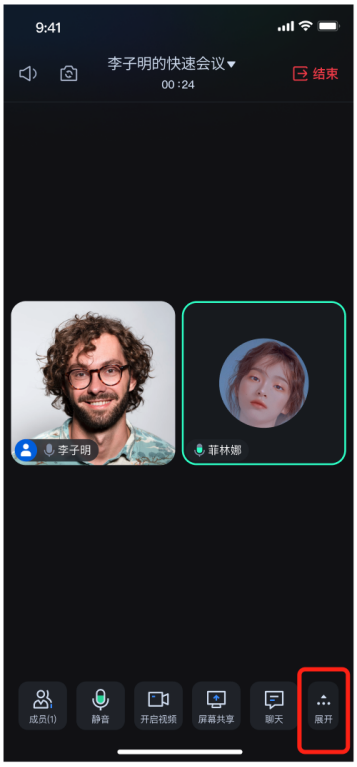
悬浮窗样式



使用说明

1. 会中点击底部工具栏**展开** > **悬浮窗**即可开启悬浮窗。
2. 首次开启**悬浮窗**功能时，Android 机型会跳转相关系统设置页面，需勾选开启应用的相关权限，如**悬浮窗**、**后台弹出界面**、**允许显示在其他应用的上层**等。不同机型的相关系统设置项名称可能略有不同，因此您需要根据具体机型进行相应设置。

底部栏 -> 展开按钮



Android系统设置示例

3. 在开启系统相关权限后，Android 端支持应用内和应用外的悬浮窗，iOS 端仅支持应用内的悬浮窗。
4. 在悬浮窗状态下，点击悬浮窗即可返回会议。

关键代码

您可通过以下方法开启/关闭悬浮窗功能，不同平台请参考：

Android

您可以删除 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/viewmodel/BottomViewModel.java 中 createItemList 函数的代码 `addFloatItemIfNeeded(itemDataList)` 实现关闭悬浮窗。

```
private List<BottomItemData> createItemList() {
    List<BottomItemData> itemDataList = new ArrayList<>();
    addUserListItemIfNeeded(itemDataList);
    addMicItemIfNeeded(itemDataList);
    addCameraItemIfNeeded(itemDataList);
    addRaiseHandItemIfNeeded(itemDataList);
    addApplyListItemIfNeeded(itemDataList);
    addScreenItemIfNeeded(itemDataList);
    addChatItemIfNeeded(itemDataList);
    addInviteItemIfNeeded(itemDataList);
    // 删除这行代码关闭悬浮窗功能
    // addFloatItemIfNeeded(itemDataList);
    addSettingsItemIfNeeded(itemDataList);
```

```
    return itemDataList;
}
```

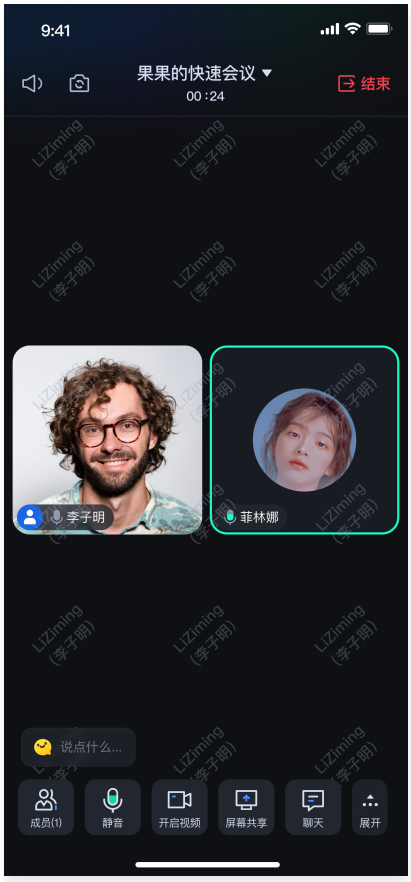
说明：
若您还有其他问题，可通过 TUIRoomKit 技术交流平台 zhiliao，向我们反馈。

文字水印
Android&iOS

最近更新时间：2024-09-26 11:02:52

功能介绍

TUIRoomKit 支持文字水印功能，用户可以在会议中添加自定义的文字水印，以保护内容版权或传达特定信息。通过文字水印功能，用户可以在会议画面上显示个人信息、公司名称或会议主题等信息，增强内容的安全性和专业性。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。



功能接入

开启水印

TUIRoomKit中，水印功能默认关闭。如果您想开启文字水印功能，可以通过以下代码进行开启。

Android

```
ConferenceSession.sharedInstance().enableWaterMark();
```

iOS

```
ConferenceSession.sharedInstance.enableWaterMark()
```

说明：
开启后，默认水印的文本内容为 `您的userId(您的userName)` 。

设置水印文本

在TUIRoomKit中，您可以自定义水印显示的文本内容以满足您特定的业务需求。您可以通过如下代码设置您的水印文本内容。

Android

```
ConferenceSession.sharedInstance().setWaterMarkText("yourWaterMarkText");  // 将字符串替换为您需要设置的水印内容
```

iOS

```
ConferenceSession.sharedInstance.setWaterMarkText(waterMarkText: "yourWaterMarkText")  // 将字符串替换为您需要设置的水印内容
```

功能定制

如果当前的 UI 不满足您的需求，您可以通过修改源代码来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

Android

您可以通过修改 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/WaterMark 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/WaterMark

WaterMark                          // 文字水印相关的视图目录
    ├── TextWaterMarkView.java     // 文字水印视图
    └── WaterMarkLineStyle.java    // 文字水印格式
```

iOS

您可以通过修改 iOS/TUIRoomKit/Source/View/Page/Widget/WaterMark 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

```
// 文件位置：iOS/TUIRoomKit/Source/View/Page/Widget/WaterMark

WaterMark                              // 文字水印相关的视图目录
    ├── WaterMarkLayer.swift           // 文字水印视图
    └── WaterMarkLineStyle.swift       // 文字水印格式
```

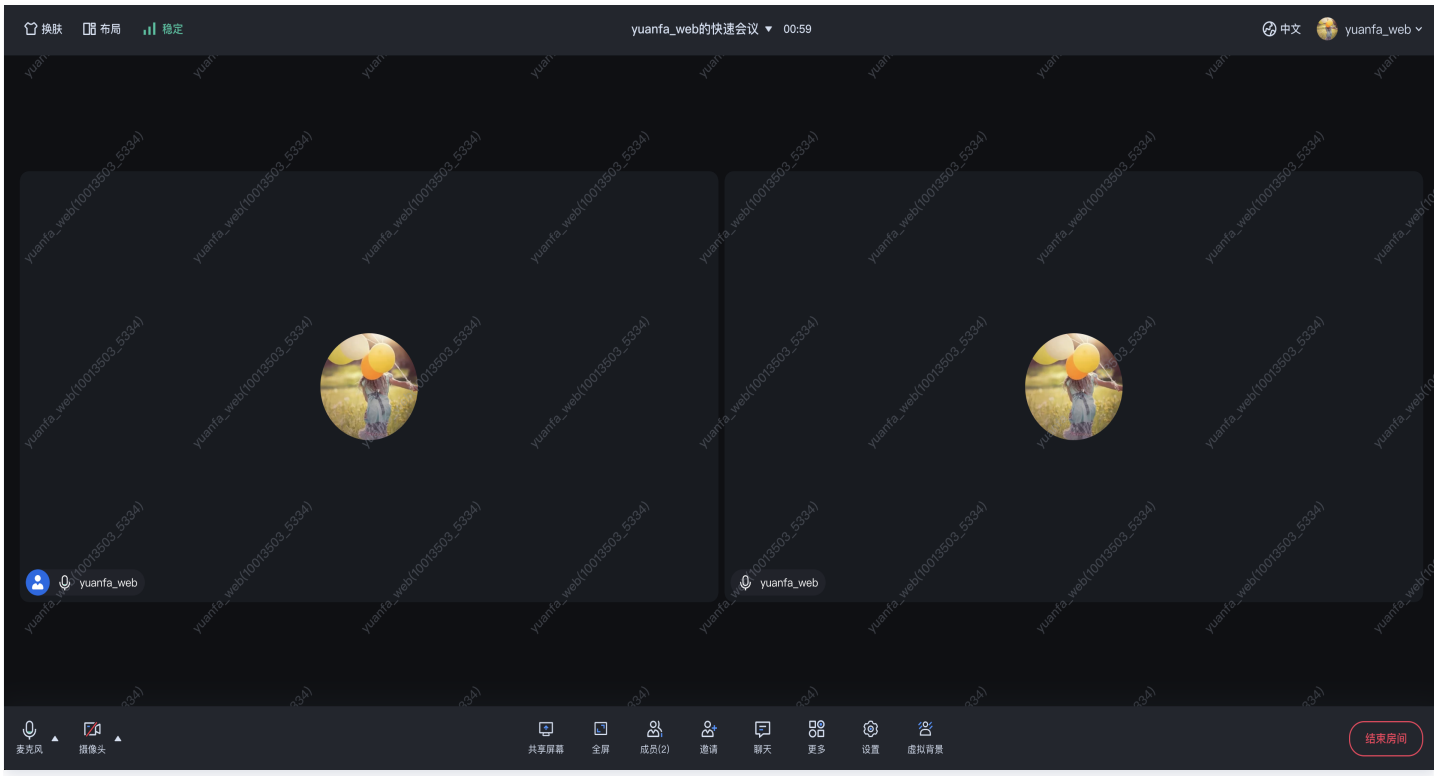
Web&Electron

最近更新时间：2024-05-24 15:04:01

TUIRoomKit 推出了文字水印功能，让用户在进行多人会议时可以设置文字水印。 本文将详细介绍如何在 TUIRoomKit 组件中使用这一特性。

集成效果

在 TUIRoomKit 组件中集成文字水印功能后，显示效果如下：



准备条件

在使用腾讯云提供的文字水印功能前，您需要前往控制台，为应用开通多人会议服务。具体步骤请参见 开通服务 。

开启文字水印

> **注意：**
> 1. 小程序平台暂不支持，electron 与 web 端均已支持。
> 2. 需使用 TUIRoomKit v2.3.3 及以上版本。

TUIRoomKit 提供了设置文字水印的功能。您可以通过调用 enableWatermark 接口来启用文字水印功能。

Web

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
```

```
conference.enableWatermark();
```

Electron

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.enableWatermark();
```

常见问题

开启文字水印后页面样式显示异常?

请检查 id 为 'roomContainer' 的父级元素，确认其 `style` 是否已设置为：`width: 100%; height: 100%; overflow: hidden;` 。

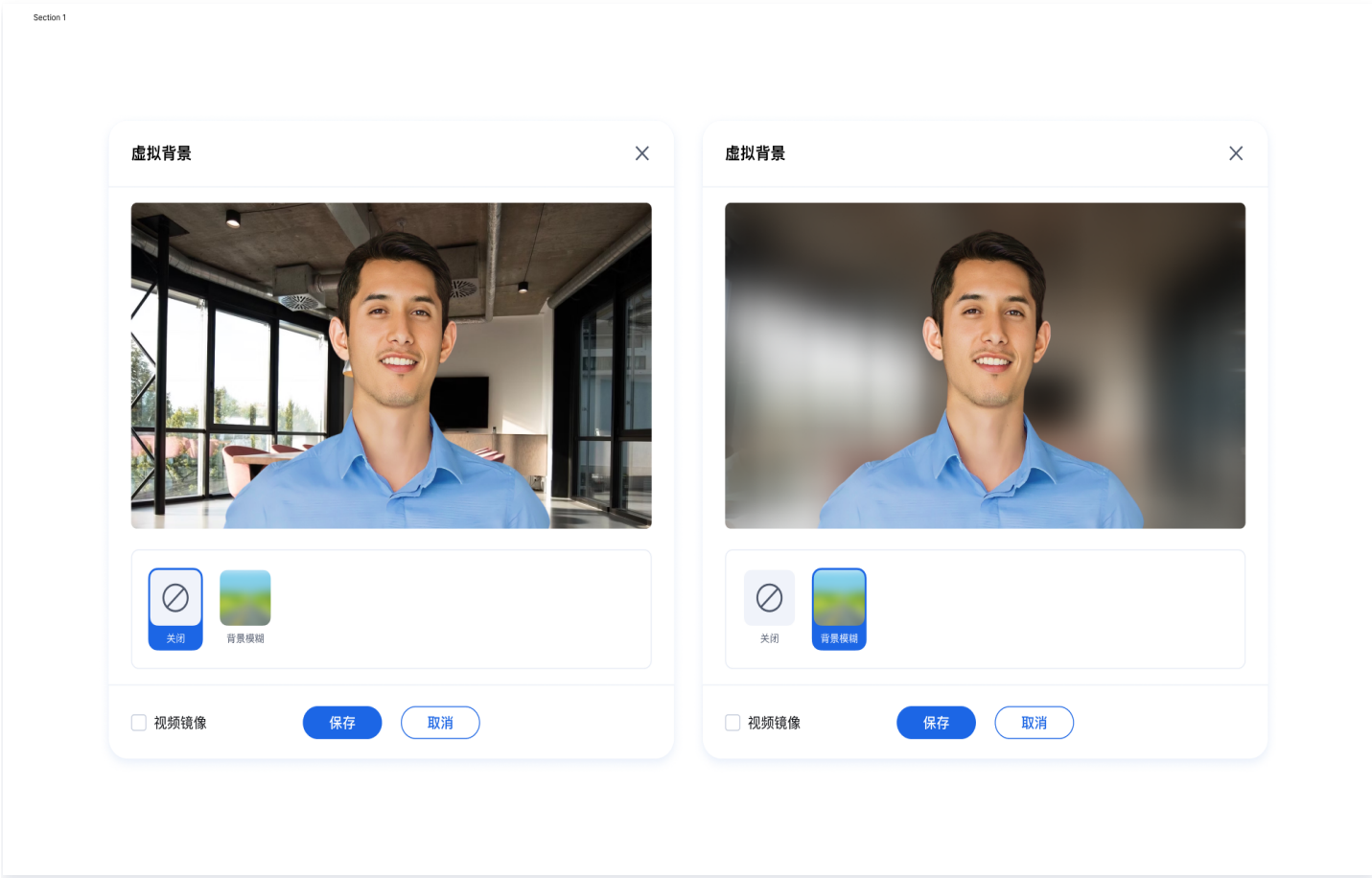
虚拟背景
Web

最近更新时间：2024-05-24 15:04:01

TUIRoomKit 推出了虚拟背景功能，让用户在进行多人会议时可以设置模糊背景，隐藏实际会议环境，保护隐私并增加会议趣味性。 本文将详细介绍如何在 TUIRoomKit 组件中使用这一特性。

集成效果

在 TUIRoomKit 组件中集成虚拟背景功能后，显示效果如下：



准备条件

在使用腾讯云提供的虚拟背景功能前，您需要前往控制台，为应用开通多人会议服务，并购买 **进阶互动版/超大房间互动版** 套餐。具体步骤请参见 开通服务 。

开启模糊背景

注意：

- **H5 暂不支持，仅适用于 Web PC 端。**
- **需使用 TUIRoomKit v2.3.3 及以上版本。**

TUIRoomKit 的 UI 方案支持设置虚拟背景。您可以通过调用 enableVirtualBackground 接口，在 UI 上显示虚拟背景功能按钮。点击该按钮即可启用虚拟背景功能。

Vue3

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableVirtualBackground();
```

Vue2

```
import { conference } from '@tencentcloud/roomkit-web-vue2.7';
conference.enableVirtualBackground();
```

常见问题

开启虚拟背景无反应或有延迟?

- 确保已购买多人会议**进阶互动版/超大房间互动版**套餐，详见 开通服务 。
- 网络较差时，虚拟背景模型文件可能未下载完，从而导致开启虚拟背景失败。

关闭摄像头，是否能开启虚拟背景?

不能。

监听会议状态

最近更新时间：2024-11-06 09:31:52

本文介绍 TUIRoomKit 组件会议状态回调的使用。

会议状态监听

如果您的业务需要**监听会议的状态**，例如会议开始、结束等会议过程中的事件，可以参见如下代码：

Android（Java）

```
ConferenceDefine.ConferenceObserver observer = new ConferenceDefine.ConferenceObserver() {
    @Override
    public void onConferenceStarted(TUIRoomDefine.RoomInfo roomInfo, TUICommonDefine.Error
error, String message) {
    }

    @Override
    public void onConferenceJoined(TUIRoomDefine.RoomInfo roomInfo, TUICommonDefine.Error
error, String message) {
    }

    @Override
    public void onConferenceExisted(String roomId) {
    }

    @Override
    public void onConferenceFinished(String roomId) {
    }
};
ConferenceSession.sharedInstance().addObserver(observer);
```

Android（Kotlin）

```
val observer: ConferenceObserver = object : ConferenceObserver() {
    override fun onConferenceStarted(roomInfo: TUIRoomDefine.RoomInfo?, error:
TUICommonDefine.Error?, message: String?) {
    }

    override fun onConferenceJoined(roomInfo: TUIRoomDefine.RoomInfo?, error:
TUICommonDefine.Error?, message: String?) {
    }

    override fun onConferenceExisted(roomId: String?) {
    }

    override fun onConferenceFinished(roomId: String?) {
    }
}
ConferenceSession.sharedInstance().addObserver(observer)
```

iOS (Swift)

```swift
import TUIRoomKit
import RTCRoomEngine

class EnterRoomViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        ConferenceSession.sharedInstance.addObserver(observer: self)
    }
}


extension EnterRoomViewController: ConferenceObserver {
    // 发起会议回调
    func onConferenceStarted(roomInfo: TUIRoomInfo, error: TUIError, message: String) {
        // Your code here
     }

    // 加入会议回调
    func onConferenceJoined(roomInfo: TUIRoomInfo, error: TUIError, message: String) {
        // Your code here
    }

    // 会议被解散回调
    func onConferenceFinished(conferenceId: String) {
        // Your code here
    }

    // 退出会议回调
    func onConferenceExited(conferenceId: String) {
       // Your code here
    }
}
```

iOS (OC)

```objc
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "RTCRoomEngine/TUIRoomEngine.h"

@interface EnterRoomViewController () <ConferenceObserver>

@end

@implementation EnterRoomViewController
- (void)viewDidLoad {
    [super viewDidLoad];
    [[ConferenceSession sharedInstance] addObserver:self];
 }

#pragma mark - ConferenceObserver
```

```
// 发起会议回调
- (void)onConferenceStartedWithRoomInfo:(TUIRoomInfo *)roomInfo error:(TUIError *)error
message:(NSString *)message {
    // Your code here
}

// 加入会议回调
- (void)onConferenceJoinedWithRoomInfo:(TUIRoomInfo *)roomInfo error:(TUIError *)error
message:(NSString *)message {
    // Your code here
}

// 会议被解散回调
- (void)onConferenceFinishedWithConferenceId:(NSString *)conferenceId {
    // Your code here
}

// 退出会议回调
- (void)onConferenceExitedWithConferenceId:(NSString *)conferenceId {
    // Your code here
}

@end
```

客户端 API（TUIRoomKit）
Web

最近更新时间：2025-01-03 14:56:32

API 简介

TUIRoomKit API 是多人会议组件的**含 UI 接口**，使用 TUIRoomKit API，您可以通过简单接口快速实现一个类会议场景，更详细的接入步骤，详见：快速接入 TUIRoomKit 。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

API 概览

- `<ConferenceMainView />`: TUIRoomkit UI 组件主体。
- Conference：依赖 `ConferenceMainView` 提供的 API。

API	描述
getRoomEngine	获取 roomEngine 实例。如果 roomEngine 不存在，则返回 null。
on	监听指定类型的事件。当事件发生时，将调用回调函数。
off	取消监听指定类型的事件。
login	登录会议系统。
logout	登出会议系统。
start	开始一个新的会议。
join	加入一个已经存在的会议。
leave	离开当前会议。
dismiss	解散当前会议。
setSelfInfo	设置自己的用户信息。
setLanguage	设置界面语言。
setTheme	设置界面主题。
enableWatermark	开启应用程序中的文字水印功能。具体详见：文字水印 。
enableVirtualBackground	开启应用程序的虚拟背景功能。调用此函数后，将在 UI 上显示虚拟背景功能按钮。具体内容详见：虚拟背景 。
disableTextMessaging	禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。
disableScreenSharing	禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。
hideFeatureButton	隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。

setParticipants	设置预定会议和会中邀请的邀请人列表。

ConferenceMainView 属性介绍

属性概览

属性	描述	类型	是否必填	默认值
displayMode	组件显示模式控制 • permanent：常驻模式。组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态。 • wake-up：唤醒模式。组件需通过调用 conference start/join 接口并正式加入会议后才被激活，之前不会显示。	'permanent' | 'wake-up'	否	permanent

示例代码

Vue3

```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue3';
const init = async () => {
    await conference.login({
      // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
      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: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
}
init();
</script>
```

Vue2

```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
```

```
</template>
<script>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
  async created() {
    await conference.login({
      // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
      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: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
  },
};
</script>
```

Conference API 详情

Conference 提供了一系列用于管理和控制在线会议功能的方法。通过实现这个接口，开发者可以方便地在他们的应用中集成在线会议的功能。

getRoomEngine

获取 roomEngine 实例。如果 roomEngine 不存在，则返回 null。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference, TUIRoomEngine } from '@tencentcloud/roomkit-web-vue3';
TUIRoomEngine.once('ready', () => {
    const roomEngine = conference.getRoomEngine();
});
```

Returns：*TUIRoomEngine | null*

on

监听指定类型的事件。当事件发生时，将调用回调函数。
参数：

参数	类型	默认值	含义
eventType	RoomEvent	-	要监听的事件类型

callback	() => void	-	事件发生时调用的回调函数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference, RoomEvent } from '@tencentcloud/roomkit-web-vue3';
conference.on(RoomEvent.RoomStart, () => {
  console.log('[conference] The meeting has already started.')
});
conference.on(RoomEvent.ROOM_DISMISS, () => {
  console.log('[conference] The meeting has been dismissed')
});
```

Returns：*void*

off

取消监听指定类型的事件。
参数：

参数	类型	默认值	含义
eventType	RoomEvent	-	要取消监听的事件类型
callback	() => void	-	之前添加的回调函数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.off('event', callback);
```

Returns：*void*

login

登录会议系统。
参数：

参数	类型	默认值	含义
params	{sdkAppId: number; userId: string; userSig: string; tim?: ChatSDK}	-	登录参数对象
sdkAppId	number	-	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	-	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	-	userSig 签名 计算 userSig 的方式请参见 UserSig 相关 。
tim	ChatSDK (可选)	-	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见 TIM.create 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.login({
  sdkAppId: 123456,
  userId: 'testUser',
  userSig: 'testSig'
});
```

Returns：*Promise<void>*

logout

登出会议系统。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.logout();
```

Returns：*Promise<void>*

start

开始一个新的会议。
参数：

参数	类型	默认值	含义
roomId	string	-	会议房间 ID
params	StartParams	-	开始会议的参数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns：*Promise<void>*

join

加入一个已经存在的会议。
参数：

参数	类型	默认值	含义
roomId	string	-	会议房间 ID
params	JoinParams	-	加入会议的参数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.join('123456', {
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns：*Promise<void>*

leave

离开当前会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.leave();
```

Returns：*Promise<void>*

dismiss

解散当前会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.dismiss();
```

Returns：*Promise<void>*

setSelfInfo

设置自己的用户信息。
参数：

参数	类型	默认值	含义
options	{userName: string; avatarUrl: string}	-	用户信息对象
userName	string（可选）	-	用户昵称
avatarUrl	string（可选）	-	用户头像

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setSelfInfo({
  userName: 'test-name',
  avatarUrl: 'https://avatar.png'
});
```

Returns：*Promise<void>*

setLanguage

设置界面语言。
参数：

参数	类型	默认值	含义
language	'zh-CN' | 'en-US'	-	语言类型

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setLanguage('en-US');
```

Returns：*void*

setTheme

设置界面主题。
参数：

参数	类型	默认值	含义
theme	'LIGHT' | 'DARK'	-	主题类型

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setTheme('DARK');
```

Returns：*void*

enableWatermark

开启应用程序中的文字水印功能。具体详见： 文字水印 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableWatermark();
```

Returns：*void*

enableVirtualBackground

开启应用程序的虚拟背景功能。调用此函数后，将在 UI 上显示虚拟背景功能按钮。具体内容详见： 虚拟背景 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableVirtualBackground();
```

Returns：*void*

disableTextMessaging

禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableTextMessaging();
```

Returns：*void*

disableScreenSharing

禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableScreenSharing();
```

Returns：*void*

hideFeatureButton

隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。
参数：

参数	类型	默认值	含义
name	FeatureButton	-	需要隐藏的按钮名称

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue3';
conference.hideFeatureButton(FeatureButton.SwitchTheme);
```

Returns：*void*

setParticipants

设置预定会议和会中邀请的邀请人列表。
参数：

参数	类型	默认值	含义
participants	Array<{ userName: string; userId: string; avatarUrl: string; }>	-	需要设置的邀请人列表

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setParticipants([{ userId: 'xxx', userName: 'xxx',  avatarUrl: 'xxx' }]);
```

Returns：*void*

类型定义

RoomEvent（枚举值）

参数	类型	描述
ROOM_START	string	创建会议
ROOM_JOIN	string	加入会议
ROOM_LEAVE	string	离开会议
ROOM_DISMISS	string	会议解散
KICKED_OFFLINE	string	用户被踢下线
KICKED_OUT	string	用户被踢出会议
USER_LOGOUT	string	用户登出
ROOM_ERROR	string	会议错误
ROOM_NEED_PASSWORD	string	入会密码
CONFERENCE_INVITATION_ACCEPTED	string	接受会中呼叫

FeatureButton（枚举值）

参数	类型	描述
SwitchTheme	string	切换主题功能按钮
SwitchLayout	string	切换布局功能按钮
SwitchLanguage	string	切换语言功能按钮
FullScreen	string	全屏功能按钮
Invitation	string	邀请功能按钮
BasicBeauty	string	基础美颜功能按钮

StartParams

参数	类型	描述	默认值
roomName	string（可选）	房间名称	-
isSeatEnabled	boolean（可选）	是否启用席位	false
isOpenCamera	boolean（可选）	是否开启摄像头	false
isOpenMicrophone	boolean（可选）	是否开启麦克风	false
defaultCameraId	string（可选）	默认摄像头ID	-
defaultMicrophoneId	string（可选）	默认麦克风ID	-
defaultSpeakerId	string（可选）	默认扬声器ID	-
password	string（可选）	入会密码	-

版权所有：腾讯云计算（北京）有限责任公司

JoinParams

参数	类型	描述	默认值
isOpenCamera	boolean（可选）	是否开启摄像头	false
isOpenMicrophone	boolean（可选）	是否开启麦克风	false
defaultCameraId	string（可选）	默认摄像头ID	-
defaultMicrophoneId	string（可选）	默认麦克风ID	-
defaultSpeakerId	string（可选）	默认扬声器ID	-
password	string（可选）	入会密码	-

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2025-06-19 17:48:02

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会议 app。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

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。

版权所有：腾讯云计算（北京）有限责任公司

iOS&Mac

最近更新时间：2025-07-03 15:42:56

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。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

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

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

版权所有：腾讯云计算（北京）有限责任公司

uni-app（小程序）

最近更新时间：2024-09-24 17:43:31

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) 。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

API 概览

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

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
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>

<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>
```

uni-app（客户端）

最近更新时间：2024-09-24 17:43:31

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会议场景。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

API 概览

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

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
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>

<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>
```

Flutter

最近更新时间：2024-09-24 17:43:31

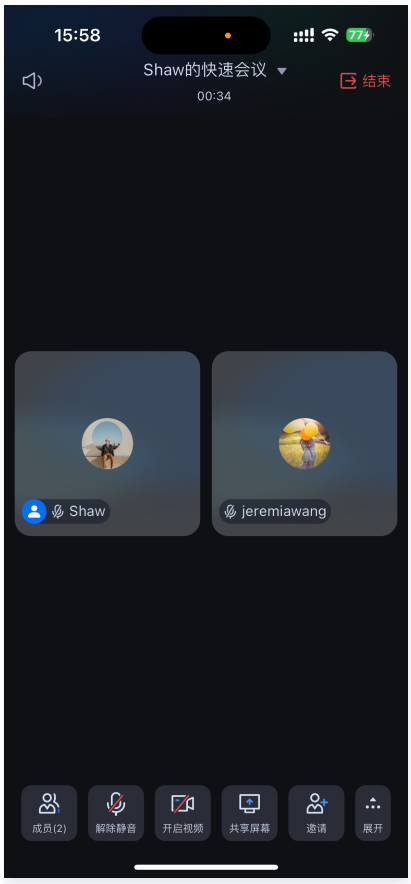
简介

- TUIRoomKit API 是多人会议组件的**含 UI 接口**。使用 TUIRoomKit 的相关API，您可以通过简单接口快速实现一个类会议场景。详细的接入步骤请参见：快速接入（TUIRoomKit）。本文档将详细介绍 Flutter TUIRoomKit 中您可能使用到的类以及相关接口，通过查阅本文档，您可以对 Flutter TUIRoomKit 的使用有一个更为详细的了解。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

ConferenceMainPage

会议主界面



字段	类型	描述
conferenceId	String	需要创建/加入的会议 Id
isCreateConference	bool	是否为创建会议（true 为创建会议，false 为加入会议）
conferenceParams	ConferenceParams	创建/加入会议的相关参数
conferenceObserver	ConferenceObserver	会议状态变化回调监听

> **说明：**
> 当您使用 `ConferenceSession` 创建或加入会议时，这里的参数均可不传。

ConferenceSession

当您期望在创建/加入会议成功后再拉起会议界面，您可以使用 `ConferenceSession` 类进行相关操作。

字段	类型	描述
isMuteMicrophone	bool	是否静音麦克风（默认为 false）
isOpenCamera	bool	是否开启摄像头（默认为 false）
isSoundOnSpeaker	bool	是否使用扬声器（默认为 true）
name	String	会议名称（默认为您的会议 Id，加入会议时传入无效）
enableMicrophoneForAllUser	bool	是否开启全员的麦克风权限（默认为 true，加入会议时传入无效）
enableCameraForAllUser	bool	是否开启全员的摄像头权限（默认为 true，加入会议时传入无效）
enableMessageForAllUser	bool	是否开启全员的发言权限（默认为 true，加入会议时传入无效）
enableSeatControl	bool	是否开启上台发言模式（默认为 false，加入会议时传入无效）
onActionSuccess	VoidCallback	创建/加入会议成功的回调，您可以在这个回调中导航至会议页面
onActionError	Function（ConferenceError，String）	创建/加入会议失败的回调

newInstance

创建一个新的ConferenceSession对象。

```
factory ConferenceSession.newInstance(String id)
```

参数	类型	描述
id	String	需要创建/加入的会议id

quickStart

快速创建会议接口。

```
Future<void> quickStart()
```

join

加入会议接口。

```
Future<void> join()
```

> ⚠ **注意：**
> 在调用快速创建会议或加入会议接口之前，您需要完成所有您需要设置的ConferenceSession的参数，详情可以参见 会前控制 。
> 直接导航至 ConferenceMainPage 并传入相关参数以 创建/加入会议 时，无需使用ConferenceSession。

ConferenceParams

字段	类型	描述
isMuteMicrophone	bool	是否静音麦克风（默认为 false）
isOpenCamera	bool	是否开启摄像头（默认为 false）
isSoundOnSpeaker	bool	是否使用扬声器（默认为 true）
name	String	会议名称（默认为您的会议 Id，加入会议时传入无效）
enableMicrophoneForAllUser	bool	是否开启全员的麦克风权限（默认为 true，加入会议时传入无效）
enableCameraForAllUser	bool	是否开启全员的摄像头权限（默认为 true，加入会议时传入无效）
enableMessageForAllUser	bool	是否开启全员的发言权限（默认为 true，加入会议时传入无效）
enableSeatControl	bool	是否开启上台发言模式（默认为 false，加入会议时传入无效）

ConferenceObserver

onConferenceStarted

会议开始事件。

```
Function(String conferenceId, ConferenceError error) onConferenceStarted
```

参数	类型	描述
conferenceId	String	会议id
error	ConferenceError	错误码

onConferenceJoined

加入会议事件。

```
Function(String conferenceId, ConferenceError error) onConferenceJoined
```

参数	类型	描述
conferenceId	String	会议id

error	ConferenceError	错误码

onConferenceFinished

会议结束事件，当主动结束会议或会议被解散时会触发该回调。

```
Function(String conferenceId) onConferenceFinished
```

参数	类型	描述
conferenceId	String	会议id

onConferenceExited

退出会议事件，当主动退出会议或被踢出会议时会触发该回调。

```
Function(String conferenceId) onConferenceFinished
```

参数	类型	描述
conferenceId	String	会议id

ConferenceError

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
errConferenceIdNotExist	-2100	进房时房间不存在，或许已被解散
errConferenceIdInvalid	-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
errConferenceIdOccupied	-2106	房间 ID 已被使用，请选择别的房间 ID
errConferenceNameInvalid	-2107	房间名称非法，名称最长30字节，如果包含中文，字符编码必须是 UTF-8。

Electron（Windows&Mac）

最近更新时间：2025-01-03 14:56:32

API 简介

TUIRoomKit API 是多人会议组件的**含 UI 接口**，使用 TUIRoomKit API，您可以通过简单接口快速实现一个类会议场景，更详细的接入步骤，详见：快速接入 TUIRoomKit 。

> **说明：**
> 如果您有自己的 UI 设计，想结合 RoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 RTC Room Engine SDK API 文档，了解更多功能。

API 概览

- `<ConferenceMainView />`: TUIRoomkit UI 组件主体。
- Conference：依赖 `ConferenceMainView` 提供的 API。

API	描述
getRoomEngine	获取 roomEngine 实例。如果 roomEngine 不存在，则返回 null。
on	监听指定类型的事件。当事件发生时，将调用回调函数。
off	取消监听指定类型的事件。
login	登录会议系统。
logout	登出会议系统。
start	开始一个新的会议。
join	加入一个已经存在的会议。
leave	离开当前会议。
dismiss	解散当前会议。
setSelfInfo	设置自己的用户信息。
setLanguage	设置界面语言。
setTheme	设置界面主题。
enableWatermark	开启应用程序中的文字水印功能。具体详见：文字水印 。
disableTextMessaging	禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。
disableScreenSharing	禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。
hideFeatureButton	隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。
setParticipants	设置预定会议和会中邀请的邀请人列表。

ConferenceMainView 属性介绍

属性概览

属性	描述	类型	是否必填	默认值
displayMode	组件显示模式控制： • permanent：常驻模式。组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态。 • wake-up：唤醒模式。组件需通过调用 conference start/join 接口并正式加入会议后才被激活，之前不会显示。	'permanent' | 'wake-up'	否	permanent

示例代码

Vue3

```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-electron-vue3';
const init = async () => {
    await conference.login({
      // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
      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: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
}
init();
</script>
```

Vue2

```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-electron-vue2.7';
export default {
```

```
  components: {
    ConferenceMainView,
  },
  async created() {
    await conference.login({
      // 获取 sdkAppId 可参考文档开通服务部分，
https://write.woa.com/document/13967822070813491 2#d9d2b8d1-f19f-441b-a436-457748c94997
      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: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
  },
};
</script>
```

Conference API 详情

Conference 提供了一系列用于管理和控制在线会议功能的方法。通过实现这个接口，开发者可以方便地在他们的应用中集成在线会议的功能。

getRoomEngine

获取 roomEngine 实例。如果 roomEngine 不存在，则返回 null。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference, TUIRoomEngine } from '@tencentcloud/roomkit-electron-vue3';
TUIRoomEngine.once('ready', () => {
    const roomEngine = conference.getRoomEngine();
});
```

Returns：*TUIRoomEngine | null*

on

监听指定类型的事件。当事件发生时，将调用回调函数。
参数：

参数	类型	默认值	含义
eventType	RoomEvent	-	要监听的事件类型
callback	() => void	-	事件发生时调用的回调函数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference, RoomEvent } from '@tencentcloud/roomkit-electron-vue3';
```

```
conference.on(RoomEvent.RoomStart, () => {
  console.log('[conference] The meeting has already started.')
});
conference.on(RoomEvent.ROOM_DISMISS, () => {
  console.log('[conference] The meeting has been dismissed')
});
```

Returns：*void*

off

取消监听指定类型的事件。
参数：

参数	类型	默认值	含义
eventType	RoomEvent	-	要取消监听的事件类型
callback	() => void	-	之前添加的回调函数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.off('event', callback);
```

Returns：*void*

login

登录会议系统。
参数：

参数	类型	默认值	含义
params	{sdkAppId: number; userId: string; userSig: string; tim?: ChatSDK}	-	登录参数对象
sdkAppId	number	-	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	-	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	-	userSig 签名 计算 userSig 的方式请参考 UserSig 相关 。
tim	ChatSDK (可选)	-	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 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.login({
  sdkAppId: 123456,
```

```
  userId: 'testUser',
  userSig: 'testSig'
});
```

Returns：*Promise<void>*

logout

登出会议系统。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.logout();
```

Returns：*Promise<void>*

start

开始一个新的会议。
参数：

参数	类型	默认值	含义
roomId	string	-	会议房间 ID
params	StartParams	-	开始会议的参数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns：*Promise<void>*

join

加入一个已经存在的会议。
参数：

参数	类型	默认值	含义
roomId	string	-	会议房间 ID
params	JoinParams	-	加入会议的参数

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.join('123456', {
  isOpenCamera: false,
  isOpenMicrophone: false,
```

```
});
```

Returns：*Promise<void>*

leave

离开当前会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.leave();
```

Returns：*Promise<void>*

dismiss

解散当前会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.dismiss();
```

Returns：*Promise<void>*

setSelfInfo

设置自己的用户信息。
参数：

参数	类型	默认值	含义
options	{userName: string; avatarUrl: string}	-	用户信息对象
userName	string（可选）	-	用户昵称
avatarUrl	string（可选）	-	用户头像

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.setSelfInfo({
  userName: 'test-name',
  avatarUrl: 'https://avatar.png'
});
```

Returns：*Promise<void>*

setLanguage

设置界面语言。
参数：

参数	类型	默认值	含义

language	'zh-CN' | 'en-US'	-	语言类型

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.setLanguage('en-US');
```

Returns：*void*

setTheme

设置界面主题。
参数：

参数	类型	默认值	含义
theme	'LIGHT' | 'DARK'	-	主题类型

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.setTheme('DARK');
```

Returns：*void*

enableWatermark

开启应用程序中的文本消息功能。具体详见： 文字水印 。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.enableWatermark();
```

Returns：*void*

disableTextMessaging

禁用应用程序中的文本消息功能。调用此函数后，用户将无法发送或接收文本消息。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.disableTextMessaging();
```

Returns：*void*

disableScreenSharing

禁用应用程序中的屏幕共享功能。调用此函数后，用户将无法与他人共享屏幕。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.disableScreenSharing();
```

Returns：*void*

hideFeatureButton

隐藏应用程序中的特定功能按钮。调用此函数并传入适当的 FeatureButton 枚举值后，相应的按钮将从用户界面中隐藏。
参数：

参数	类型	默认值	含义
name	FeatureButton	-	需要隐藏的按钮名称

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference, FeatureButton } from '@tencentcloud/roomkit-electron-vue3';
conference.hideFeatureButton(FeatureButton.SwitchTheme);
```

Returns：*void*

setParticipants

设置预定会议和会中邀请的邀请人列表。
参数：

参数	类型	默认值	含义
participants	Array<{ userName: string; userId: string; avatarUrl: string; }>	-	需要设置的邀请人列表

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference, FeatureButton } from '@tencentcloud/roomkit-electron-vue3';
conference.setParticipants([{ userId: 'xxx', userName: 'xxx',  avatarUrl: 'xxx' }]);
```

Returns：*void*

类型定义

RoomEvent（枚举值）

参数	类型	描述
ROOM_START	string	创建会议
ROOM_JOIN	string	加入会议
ROOM_LEAVE	string	离开会议
ROOM_DISMISS	string	会议解散
KICKED_OFFLINE	string	用户被踢下线
KICKED_OUT	string	用户被踢出会议
USER_LOGOUT	string	用户登出
ROOM_ERROR	string	会议错误

ROOM_NEED_PASSWORD	string	入会密码

FeatureButton（枚举值）

参数	类型	描述
SwitchTheme	string	切换主题功能按钮
SwitchLayout	string	切换布局功能按钮
SwitchLanguage	string	切换语言功能按钮
FullScreen	string	全屏功能按钮
Invitation	string	邀请功能按钮
BasicBeauty	string	基础美颜功能按钮

StartParams

参数	类型	描述	默认值
roomName	string（可选）	房间名称	-
isSeatEnabled	boolean（可选）	是否启用席位	false
isOpenCamera	boolean（可选）	是否开启摄像头	false
isOpenMicrophone	boolean（可选）	是否开启麦克风	false
defaultCameraId	string（可选）	默认摄像头 ID	-
defaultMicrophoneId	string（可选）	默认麦克风 ID	-
defaultSpeakerId	string（可选）	默认扬声器 ID	-
password	string（可选）	入会密码	-

JoinParams

参数	类型	描述	默认值
isOpenCamera	boolean（可选）	是否开启摄像头	false
isOpenMicrophone	boolean（可选）	是否开启麦克风	false
defaultCameraId	string（可选）	默认摄像头 ID	-
defaultMicrophoneId	string（可选）	默认麦克风 ID	-
defaultSpeakerId	string（可选）	默认扬声器 ID	-
password	string（可选）	入会密码	-

版权所有：腾讯云计算（北京）有限责任公司

服务端 API（TUIRoomKit）
账号系统
账号管理

最近更新时间：2025-06-26 17:11:51

账号即用户名(UserId)，分为普通账号和管理员账号，登录 即时通信 IM 控制台，在左侧导航栏选择**消息服务 Chat > 账号管理**，您可以根据实际业务需求管理账号。

> **说明：**
> Live 和即时通信 IM 共用同一套账号系统。

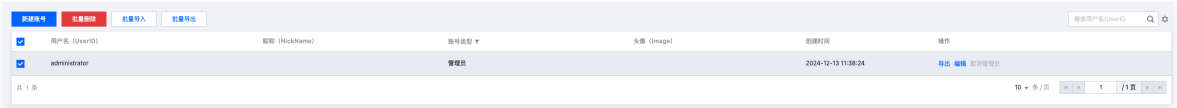
新建账号

1. 在账号管理页面，单击**新建账号**。
2. 在弹出的新建账号对话框中，配置以下参数：

   - 账号类型：区分普通账号和管理员账号。“App管理员”是对 App 具有最高管理权限的角色，可调用 REST API 接口，进行用户管理，房间管理，房间成员管理等操作。每个应用最多支持配置 10 个管理员。
   - 用户名：需输入用户名（UserID），必填。
   - 用户昵称：可输入用户昵称，非必填。
   - 头像：可输入用户头像链接 URL，非必填。
3. 单击**确定**保存配置。
4. 账号新建完成后，在账号列表中可以查看用户名、昵称、账号类型、头像和创建时间。

删除账号

1. 在账号管理页面，在列表中选中需要删除的账号，单击**批量删除**。



2. 在弹出的确认对话框中，单击**确定**确认删除。删除后数据无法恢复，请谨慎操作。

编辑账号

1. 在账号管理页面，在列表中希望编辑的账号操作列，单击**编辑**。
2. 在弹出的编辑账号对话框中，配置以下参数：

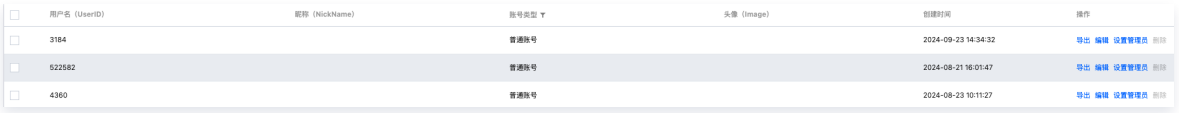
- 用户名：不可编辑。
- 昵称：可编辑用户昵称，非必填。
- 账号类型：不可编辑。
- 性别：可编辑性别，非必填。

- 生日：可编辑生日，非必填。
- 所在地：可编辑所在地，非必填。
- 个性签名：可编辑个性签名，非必填。
- 语言：可编辑语言，非必填。
- 头像：可编辑头像，非必填。
- 等级：可编辑等级，非必填。
- 角色：可编辑角色，非必填。

2.1 单击**确定**保存配置。

导出账号

1. 单独导出：在账号管理页面，在列表中希望导出的账号操作列，单击**导出**。

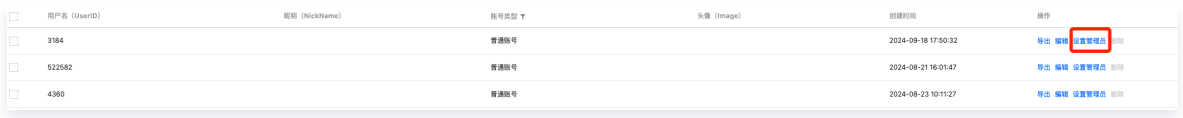


2. 批量导出：在账号管理页面，在列表中选中需要导出的账号，单击**批量导出**。

3. 在弹出的导出对话框中提示导出成功后，单击**下载**，即完成导出。

设置/取消管理员账号

1. 设置/取消管理员：在账号管理页面，在列表中找到希望设置或取消管理员的普通账号，单击**设置/取消管理员**。



> **注意：**
> administrator 账号不支持取消其管理员身份。

2. 在弹出的确认对话框中，单击**确定**确认即可。

设置管理员
确认设置 8521 的管理员权限?
确定
取消

登录鉴权

最近更新时间：2025-06-26 17:11:51

登录鉴权简介

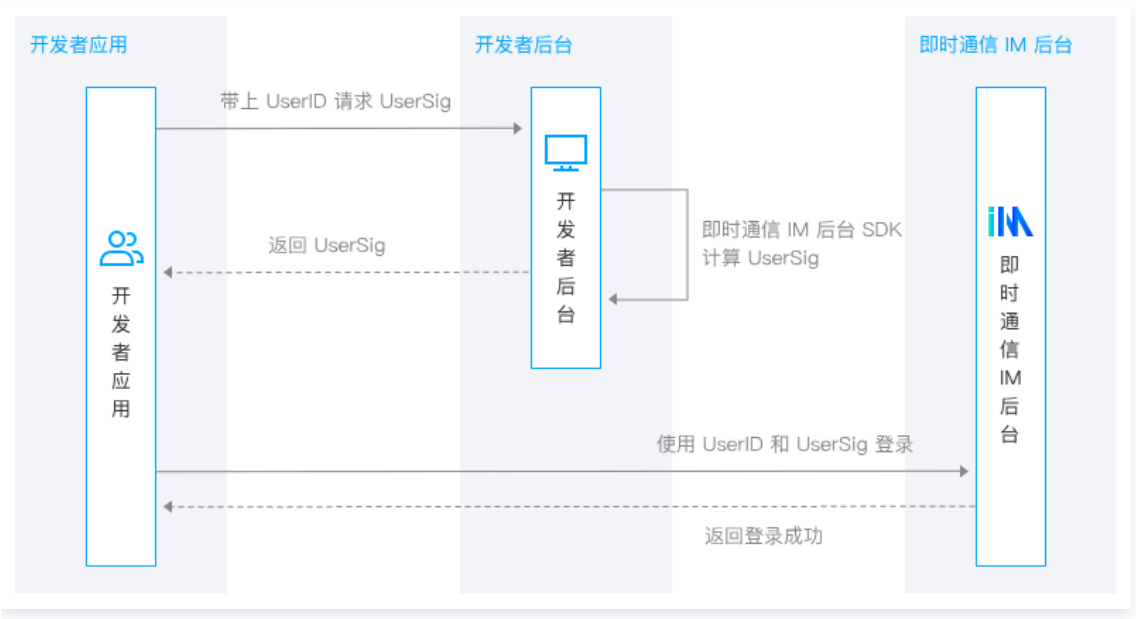
TUICallKit、TUIRoomKit、TUILiveKit 与即时通信 IM 共用同一套登录鉴权体系，即时通信 IM 的前身是 QQ 的即时通信系统，我们抽离 QQ 的通用模块，并将其整合成适合终端接入的 IM SDK 及后台服务。您可以把 IM SDK 理解为一个没有用户交互界面的 QQ，把 IM SDK 集成到您的 App 里，就相当于把一个 QQ 的内核集成在您的 App 内部。
QQ 可以用来收发消息，但前提是您必须先登录才能使用。登录 QQ 用的是 QQ 号和密码，登录 IM SDK 需要使用您指定的用户名（`UserID`）和密码（`UserSig`）。

- **UserID**：旧称为 Identifier，用户登录即时通信 IM 时使用的用户名，即您 App 里的用户 ID。
- 例如，App 里有一个用户，该用户的 ID 是27149 ，那么您可以用27149作为登录即时通信 IM 的 UserID。
- **UserSig**：用户登录即时通信 IM 时使用的密码，其本质是 App Server 用密钥对 UserID 等信息加密后的数据。具体生成方法请参见 生成 UserSig 。

> **说明：**
> TUICallKit、TUIRoomKit、TUILiveKit 的系统和即时通信 IM 系统共用一套登录鉴权体系。

App 登录流程

推荐的 App 登录即时通信 IM 流程如下：



> **注意：**
> - 即时通信 IM 后台完全信赖 UserSig，为避免数据和业务受影响，请务必确保私钥的安全。
> - 即时通信 IM 后台 SDK 默认接口生成的 UserSig 有效期为180天，可以使用含有效期参数的接口自行设定有效期，开发者需要在 UserSig 过期前到开发者后台获取新的 UserSig。
> - 用于生成 UserSig 的即时通信 IM 后台 SDK 详细请参阅 生成 UserSig 。

App 管理员

即时通信 IM 的部分服务需要管理员权限，例如调用 REST API 接口、解散群以及全员推送等。App 管理员是对 App 具有最高管理权限的角色，与普通账号相比，其区别如下：

- 读取权限更高。例如，获取 App 内部的所有群组、获取任意群组的任意资料。
- 操作权限更高。例如，给任意用户发消息、在任意群组中增删成员。

设置 App 管理员仅能通过登录控制台操作，具体操作请参见 配置账号管理员 。

生成 UserSig

最近更新时间：2025-06-29 18:01:02

UserSig 是用户登录的密码，也是使用RestAPI的签名，其本质是对 UserID 等信息加密后得到的密文，本文将指导您如何生成 UserSig。以下视频将帮助您快速了解如何生成 UserSig：

> **说明：**
> TUICallKit、TUIRoomKit、TUILiveKit 的系统和即时通信 IM 系统共用同一套 Usersig 的生成方式。

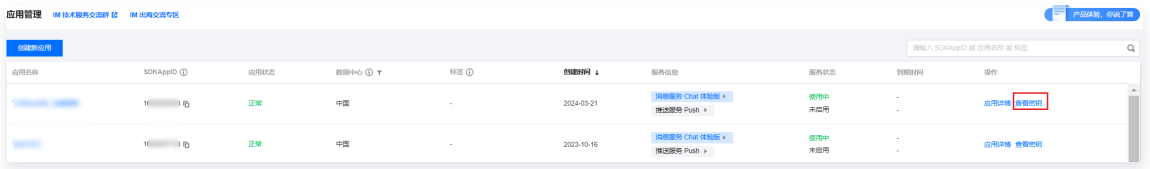
观看视频

获取密钥

1. 登录即时通信 IM 控制台 。

> **说明：**
> 如果您还没有应用，请先通过 开通服务 创建一个应用。

2. 单击**应用管理**，进入应用管理列表。
3. 在目标应用的操作列，单击**查看密钥**。



4. 在对话框中单击**显示密钥**。

5. 单击复制图标即可复制并储存密钥信息。

> **注意：**

> 请妥善保管密钥信息，谨防泄露。

客户端计算 UserSig

IM SDK 示例代码中提供的 `GenerateTestUserSig` 的开源模块可以帮忙您快速生成 UserSig。您只需设置 SDKAPPID（应用 SDKAppID）、EXPIRETIME（UserSig 过期时间）和 SECRETKEY（密钥信息）三个成员变量的取值，然后调用 genTestUserSig() 函数即可快速获取 UserSig。
为了简化实现过程，我们提供了下列语言或平台的 UserSig 计算源码，您可以直接下载并集成到您的客户端。

编程语言	所属平台	GenerateTestUserSig 源代码
Java	Android	GenerateTestUserSig.java
Objective-C	iOS	GenerateTestUserSig.h
Objective-C	Mac	GenerateTestUserSig.h
C++	Windows	GenerateTestUserSig.h
Javascript	Web	GenerateTestUserSig.js
Javascript	小程序	GenerateTestUserSig.js
Dart	Flutter	GenerateTestUserSig.dart

> ⚠ **注意：**
> 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 。

服务端计算 UserSig

采用服务端计算 UserSig，可以最大限度地保障计算 UserSig 所用的密钥信息不被泄露。您只需将计算代码部署在您的服务器上，并提供面向 App 的服务端接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。
为了简化实现过程，我们提供了下列语言或平台的 UserSig 计算源码，您可以直接下载并集成到您的服务端。

语言版本	关键函数	下载链接
Java	HMAC-SHA256	genSig
GO	HMAC-SHA256	GenSig
PHP	HMAC-SHA256	genSig
Nodejs	HMAC-SHA256	genSig
Python	HMAC-SHA256	gen_sig
C#	HMAC-SHA256	GenSig
C++	HMAC-SHA256	gen_sig

UserSig 计算函数中主要包括 SDKAppID、UserID 以及 UserSig 有效期等关键参数，关键参数的详细介绍如下表所示：

版权所有：腾讯云计算（北京）有限责任公司

> **说明：**
> 下表中的字段名以 Java 语言的源码为例，其他语言略有差异，请以实际字段名为准。

字段名示例	参数说明
sdkappid	应用 SDKAppID，可在即时通信 IM 控制台 的应用卡片中获取。
userId	用户 ID，旧称为 Identifier。
expire	UserSig 的有效期，单位为秒。建议 UserSig 有效期最短不小于24小时，最长不超过50年。为了您的账号安全，建议将 UserSig 有效期设置为两个月。
userbuf	即时通信 IM 中均默认使用不带 UserBuf 的接口，即该参数默认填写为 `null` 。
key	密钥信息，可在即时通信 IM 控制台 的应用详情页面中获取。

老版本算法

为了简化签名计算难度，方便客户更快速地使用腾讯云服务，即时通信 IM 服务自2019.07.19开始启用新的签名算法，从之前的 ECDSA-SHA256 升级为 HMAC-SHA256。2019.07.19以后创建的 SDKAppID 均会采用新的 HMAC-SHA256 算法。
如果您的 SDKAppID 是2019.07.19之前创建的，建议升级为 HMAC-SHA256 算法，升级过程不会影响现网业务。您也可以继续使用老版本的签名算法，ECDSA-SHA256 算法的源码下载链接如下：

语言版本	签名算法	下载链接
Java	ECDSA-SHA256	Github
GO	ECDSA-SHA256	Github
PHP	ECDSA-SHA256	Github
Nodejs	ECDSA-SHA256	Github
Python	ECDSA-SHA256	Github
C#	ECDSA-SHA256	Github
C++	ECDSA-SHA256	Github

版权所有：腾讯云计算（北京）有限责任公司

资料管理

最近更新时间：2025-06-26 17:11:51

TUICallKit、TUIRoomKit、TUILiveKit 的系统和即时通信 IM 系统共用同一套账号体系，资料属于账号体系里用户绑定的一些信息。

资料系统简介

即时通信 IM 开放了用户资料托管能力，提供资料相关的一整套完整解决方案。如果您想让每个用户都有自己的资料，并且轻松实现资料的设置和拉取功能，那么您可以选择即时通信 IM 的资料托管服务：

- 即时通信 IM 提供资料的存储能力，确保您的数据具备异地容灾、多地部署和自动扩容/缩容的能力，帮助您从服务器宕机、多拷贝主从复制和扩容缩容等复杂处理流程中得到完全地解放。
- 即时通信 IM 提供业界通用的业务处理流程，帮助您在用户资料的业务逻辑上彻底地解放。
- 即时通信 IM 提供专业的运营流程和运营团队，全年99.99%的稳定服务质量，帮助您为用户提供具有稳定口碑的服务。
- 即时通信 IM 提供简单易用的服务接口和快捷接入的帮助指引，全程为您提供星级服务。

使用即时通信 IM 的资料托管服务，可以得到以下能力：

- 标配资料字段的存储、读写能力。
- 自定义资料字段的存储、读写能力。

资料字段

资料是用来描述用户属性的一组数据，即时通信 IM 资料系统支持标配资料字段和自定义资料字段。资料字段有如下特性：

- 资料字段通过 Key-Value 来表示。
- Key 为 String 类型，其命名仅支持英文大小写字母、数字、下划线。
- Value 有以下几种类型：
  - a. uint32_t 类型的整数（自定义资料字段不支持）。
  - b. uint64_t 类型的整数（自定义资料字段不支持）。
  - c. string 类型的字符串（string 的长度不得超过500字节）。
  - d. bytes 类型的一段 buffer（buffer 的长度不得超过500字节）。

标配资料字段

目前即时通信 IM 支持的标配资料字段如下：

字段名称	类型	描述	更新时有 Push	备注
Tag_Profile_IM_Nick	String	昵称	有	长度不得超过500个字节
Tag_Profile_IM_Image	String	头像 URL	有	长度不得超过500个字节
Tag_Profile_IM_Level	uint32	等级	有	通常一个 UINT-8 数据即可保存一个等级信息您可以考虑拆分保存，从而实现多种角色的等级信息
Tag_Profile_IM_Role	uint32	角色	有	通常一个 UINT-8 数据即可保存一个角色信息，您可以考虑拆分保存，从而保存多种角色信息

版权所有：腾讯云计算（北京）有限责任公司

自定义资料字段

自定义资料字段是各 App 根据各自业务需要而设置的用户数据。通过自定义资料，各 App 可以将一些额外数据附加到用户资料上，并通过现有接口进行读写操作。

自定义资料字段的申请

App 管理员可以通过即时通信 IM 控制台 > **应用配置** > **功能配置** 申请自定义资料字段，申请提交后，自定义资料字段将在10分钟内生效。
申请自定义资料字段时，需要为每一个自定义资料字段提交如下资料：

- 自定义资料字段的名称（Key）：详情可参见 自定义资料字段的命名规范 。
- 自定义资料字段的类型（Value）：详情可参见 资料字段 。

自定义资料字段的命名规范

自定义资料字段的命名规则如下：

- 自定义资料字段的名称包括前缀和关键字两部分。
- 自定义资料字段的前缀是：Tag_Profile_Custom。
- 关键字：必须是英文字母，且长度不得超过8字节，建议用一个英文单词或该英文单词的缩写。
- 示例：某 App 要申请的自定义字段的关键字是 Test，则自定义资料字段的名称是：Tag_Profile_Custom_Test。

版权所有：腾讯云计算（北京）有限责任公司

REST API
REST API 简介

最近更新时间：2025-07-02 20:53:41

REST API 是多人音视频房间 SDK 的后台 HTTP 管理接口，其主要目的在于为开发者提供一套简单的管理入口。
为了安全性，REST API 仅提供 HTTPS 接口。

前提条件

要调用 REST API，您必须购买或领取 RoomSdk 所需的套餐包。

> **警告：**
> 新版本的 REST API 在后台架构上做了较大的优化升级，提供了丰富的能力，但遗憾的是，导致目前无法和1.x版本的 SDK 互通。
> 所以您在接入最新的 REST API 时，请务必使用最新2.0版本及以上的客户端 SDK。

调用方法

请求 URL

REST API 的 URL 格式如下：

```
https://console.tim.qq.com/$ver/$servicename/$command?
sdkappid=$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
console.tim.qq.com	请求域名	固定为 `console.tim.qq.com`
ver	协议版本号	固定为 `v4`
servicename	内部服务名，不同的 servicename 对应不同的服务类型	示例：`v4/room_engine_http_srv/create_room`，其中 `room_engine_http_srv` 为 `servicename` 更多详情请参见 REST API 接口列表
command	命令字，与 servicename 组合用来标识具体的业务功能	示例：`v4/room_engine_http_srv/create_room`，其中 `create_room` 为 `command` 更多详情请参见 REST API 接口列表
sdkappid	App 在即时通信 IM 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	参见 App 管理员
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

- App 服务端在调用 REST API 时，identifier 必须为 App 管理员账号。
- 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，但请特别注意 UserSig 的有效期。

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。
需要特别注意的是，POST 包体不能为空，即使某条协议包体中不需要携带任何信息，也需要携带一个空的 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，真正的 API 调用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：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "RequestId": "Id-70e312f1de024af5a36714b7b71da224-0-Seq-63504"
    // REST API 其他应答内容
}
```

应答包体中必然包含 ActionStatus、ErrorInfo、ErrorCode、RequestId 这三个属性，其含义如下：

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败，如果为 FAIL，ErrorInfo 带上失败原因
ErrorInfo	String	失败原因
ErrorCode	Integer	错误码，0为成功，其他为失败，可查询 错误码表 得到具体的原因
RequestId	String	错误码，0为成功，其他为失败，可查询 错误码表 得到具体的原因

调用示例

以下为通过 REST API 来 获取 App 中所有群组 示例。
HTTPS 请求：

```
POST /v4/group_open_http_svc/get_appid_group_list?
usersig=xxx&identifier=admin&sdkappid=88888888&random=99999999&contenttype=json HTTP/1.1
Host: console.tim.qq.com
Content-Length: 22
{
```

```
    "Limit": 2
}
```

HTTPS应答：

```
HTTP/1.1 200 OK
Server: nginx/1.7.10
Date: Fri, 09 Oct 2015 02:59:55 GMT
Content-Length: 156
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Allow-Methods: POST

{
    "ActionStatus": "OK",
    "ErrorCode": 0,
    "GroupIdList": [
        {
            "GroupId": "@TGS#1YTTZEAEG"
        },
        {
            "GroupId": "@TGS#1KVTZEAEZ"
        }
    ],
    "TotalCount": 58530
}
```

REST API 公共错误码

错误码	含义说明
60002	HTTP 解析错误，请检查 HTTP 请求 URL 格式
60003	HTTP 请求 JSON 解析错误，请检查 JSON 格式
60004	请求 URL 或 JSON 包体中账号或签名错误
60005	请求 URL 或 JSON 包体中账号或签名错误
60006	SDKAppID 失效，请核对 SDKAppID 有效性
60007	REST 接口调用频率超过限制，请降低请求频率
60008	服务请求超时或 HTTP 请求格式错误，请检查并重试
60009	请求资源错误，请检查请求 URL
60010	请求需要 App 管理员权限
60011	SDKAppID 请求频率超限，请降低请求频率
60012	REST 接口需要带 SDKAppID，请检查请求 URL 中的 SDKAppID
60013	HTTP 响应包 JSON 解析错误

版权所有：腾讯云计算（北京）有限责任公司

60014	置换账号超时
60015	请求包体账号类型错误，请确认账号为字符串格式
60016	SDKAppID 被禁用。
60017	请求被禁用。
60018	请求过于频繁，请稍后重试。
60019	请求过于频繁，请稍后重试。
60020	您的专业版套餐包已到期并停用，请登录 即时通信 IM 购买页面 重新购买套餐包。购买后，将在5分钟后生效。
60021	RestAPI 调用来源 IP 非法。

FAQ

REST API 请求有概率超时，收不到任何响应

1. Room 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 console.tim.qq.com 443` 确认能否连接服务端口。
3. 使用 `curl -I https://console.tim.qq.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。

> **说明**
> 由于 HTTPS 短连接建连耗时比较大，每次请求都有 TCP + tls 握手开销，所以建议 REST API 长连接接入。
> 使用标准 HTTP 库的场景：HTTP1.0 需要指定请求头部 Connection: keep-alive，HTTP1.1 默认支持长连接；基于 TCP 封装 HTTPS 请求的场景，可以复用 TCP 连接来收发请求。

REST API 列表

最近更新时间：2024-04-03 13:35:02

房间管理

功能说明	接口
创建房间	room_engine_http_srv/create_room
解散房间	room_engine_http_srv/destroy_room
更新房间信息	room_engine_http_srv/update_room_info
获取房间信息	room_engine_http_srv/get_room_info

用户管理

功能说明	接口
获取房间成员列表	room_engine_http_srv/get_room_member_list
更新房间成员资料	room_engine_http_srv/update_room_member_info
转让房主	room_engine_http_srv/change_room_owner
设置房间成员标签	room_engine_http_srv/change_room_owner
房间成员封禁	room_engine_http_srv/ban_room_member
房间成员解封	room_engine_http_srv/unban_room_member
获取被封禁房间成员列表	room_engine_http_srv/get_banned_member_list
踢出房间内成员	room_engine_http_srv/kick_user_out

连麦管理

功能说明	接口
获取麦位列表	room_engine_http_mic/get_seat_list
用户上麦	room_engine_http_mic/pick_user_on_seat
用户下麦	room_engine_http_mic/kick_user_off_seat
锁定麦位	room_engine_http_mic/lock_seat

版权所有：腾讯云计算（北京）有限责任公司

房间管理
创建房间

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口创建房间、创建预定会议。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/create_room?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/create_room	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomInfo":{
      "RoomId": "room-test",
      "RoomName": "room-name-test",
      "RoomType": "Conference",
      "Owner_Account": "user2",
      "MaxMemberCount": 300,
      "ScheduleStartTime" : 1693271355,
      "ScheduleEndTime" : 1693272355,
      "IsVideoDisabled": true,
      "IsAudioDisabled": true,
```

```
        "IsMessageDisabled": true,
        "IsScreenSharingDisabled": true,
        "IsCloudRecordingDisabled": true,
        "CustomInfo": "custom123",
        "IsSeatEnabled": true,
        "MaxSeatCount": 16,
        "TakeSeatMode": "ApplyToTake"
    },
    "ScheduleInviteeList_Account":[
        "user1", "user2", "user3"
    ]
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID，最长48个字节
RoomName	String	选填	房间名称，默认为房间 ID，最长100个字节
RoomType	String	必填	房间类型：Conference（会议房间）
Owner_Account	String	选填	房主 ID（需是 已导入 的账号），默认为接口调用者的用户 ID
MaxMemberCount	Integer	选填	最大房间成员数量，缺省时的默认值：付费套餐包上限，例如体验版是20，如果升级套餐包，需按照修改房间资料修改这个字段
ScheduleStartTime	Integer	选填	预定会议开始时间，默认为当前时间
ScheduleEndTime	Integer	选填	预定会议结束时间，默认为开始时间后1小时，会议最小时间不得低于5分钟，最长时间不得超过24h
IsVideoDisabled	Bool	选填	全体禁画，默认 false
IsAudioDisabled	Bool	选填	全体禁音，默认 false
IsMessageDisabled	Bool	选填	禁止所有成员发文本消息，默认 false
IsScreenSharingDisabled	Bool	选填	禁止屏幕共享，默认 false
IsCloudRecordingDisabled	Bool	选填	禁止云录制，默认 false
CustomInfo	String	选填	自定义信息，最长500个字节
IsSeatEnabled	Bool	选填	是否支持麦位，默认不支持
MaxSeatCount	Integer	选填	最大麦位数，默认为套餐包上限（默认最大20）

TakeSeatMode	String	选填	麦位模式：FreeToTake（自由上麦），ApplyToTake（申请上麦）
ScheduleInviteeList_Account	Array	选填	预约成员列表，最多300个

注意：
- 用户需要在会议结束以后，手动调用 解散房间 接口，结束本场会议。
- 如果没有手动调用解散房间接口，后台会在会议结束时间6小时以后，尝试回收房间，回收的条件是房间内成员数为0。

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100003	房间 ID 已存在，请选择其他房间 ID
100007	无付费信息，需在控制台购买套餐包
100010	房间 ID 已被使用，并且操作者为房主，可以直接使用
100011	房间 ID 已被 IM 占用，可以换一个房间 ID 使用，或者先通过 IM 接口解散该群

100012	创建房间超过频率超限，同一房间 ID 1秒内只能创建一次

可能触发的回调

- 创建房间之后回调

解散房间

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口解散。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/destroy_room?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/destroy_room	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId": "room-test"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100006	操作权限不足

可能触发的回调

- 解散房间之后回调

更新房间信息

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口更新房间信息。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/update_room_info?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/update_room_info	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomInfo":{
      "RoomId": "room-test",
      "RoomName": "room-name-test",
      "MaxMemberCount": 300,
      "IsVideoDisabled": true,
      "IsAudioDisabled": true,
      "IsMessageDisabled": true,
      "IsScreenSharingDisabled": true,
      "IsCloudRecordingDisabled": true,
      "CustomInfo": "custom123",
      "TakeSeatMode": "ApplyToTake"
  }
```

```
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
RoomName	String	选填	房间名称，默认为房间ID，最长100个字节
MaxMemberCount	Integer	选填	最大房间成员数量
IsVideoDisabled	Bool	选填	全体禁画
IsAudioDisabled	Bool	选填	全体禁音
IsMessageDisabled	Bool	选填	禁止所有成员发文本消息
IsScreenSharingDisabled	Bool	选填	禁止屏幕共享
IsCloudRecordingDisabled	Bool	选填	禁止云录制
CustomInfo	String	选填	自定义信息，最长500个字节
TakeSeatMode	String	选填	麦位模式：FreeToTake（自由上麦），ApplyToTake（申请上麦）

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

可能触发的回调

- 更新房间信息之后回调

获取房间信息

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口获取房间信息。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/get_room_info?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/get_room_info	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId": "room-test"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-81fb8ae1529f409a9ed83ef3c3071657-O-Seq-56057",
  "Response": {
      "RoomInfo": {
          "RoomId": "room-test",
          "RoomName": "room-name-test",
          "RoomType": "Conference",
          "Owner_Account": "user2",
          "MaxMemberCount": 300,
          "MaxSeatCount": 16,
          "IsVideoDisabled": true,
          "IsAudioDisabled": true,
          "IsMessageDisabled": true,
          "IsScreenSharingDisabled": true,
          "IsCloudRecordingDisabled": true,
          "CustomInfo": "custom123",
          "ScheduleStartTime": 1703491546,
          "ScheduleEndTime": 1703495146,
          "RoomStatus": "Running",
          "IsSeatEnabled": true,
          "TakeSeatMode": "ApplyToTake",
          "CreateTime": 1703491546,
          "MemberCount": 1
      }
  }
}
```


应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId
RoomId	String	房间 ID
RoomName	String	房间名称
RoomType	String	房间类型：Conference（会议房间）
Owner_Account	String	房主 ID
MaxMemberCount	Integer	最大房间成员数量
ScheduleStartTime	Integ	预定会议开始时间

	er	
ScheduleEndTime	Integer	预定会议结束时间
IsVideoDisabled	Bool	全体禁画
IsAudioDisabled	Bool	全体禁音
IsMessageDisabled	Bool	禁止所有成员发文本消息
IsScreenSharingDisabled	Bool	禁止屏幕共享
IsCloudRecordingDisabled	Bool	禁止云录制
CustomInfo	String	自定义信息
RoomStatus	String	房间状态：None（关闭），NotStart（待开始），Running（进行中）
IsSeatEnabled	Bool	是否支持麦位
MaxSeatCount	Integer	最大麦位数
TakeSeatMode	String	麦位模式：None（关闭），FreeToTake（自由上麦），ApplyToTake（申请上麦）
CreateTime	Integer	预定会议开始时间
MemberCount	Integer	房间成员数量

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

用户管理
获取房间成员列表

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口获取房间成员列表。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/get_room_member_list?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/get_room_member_list	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId": "room-test",
  "Count": 10,
  "Next": ""
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

Count	Integer	选填	获取成员数量，默认50
Next	String	选填	上一次拉取到的成员位置，首次调用填写""，续拉使用返回值中的 Next 值

应答包体示例

基础形式


```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-52f049ba12d741b7a84959700acdb1d5-0-Seq-56287",
  "Response": {
      "Next": "",
      "MemberList": [
          {
              "Member_Account": "user2",
              "NameCard": "123",
              "Role": "RoomOwner",
              "Marks": [
                  1000,
                  1002,
                  1003
              ],
              "CustomInfo": ""
          }
      ]
  }
}
```


应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId
MemberList	Array	房间成员列表
Member_Account	String	房间成员 ID
NameCard	String	房间成员昵称
Role	String	房间成员权限：Owner（房主），Admin（管理员），Member（普通成员）
Marks	Array	房间成员标签
CustomInfo	String	房间成员自定义字段

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

更新房间成员资料

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口更新房间成员资料。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/update_room_member_info?
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
v4/room_engine_http_srv/update_room_member_info	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "MemberInfo":{
      "Member_Account": "user4",
      "NameCard": "jack",
      "Role": "Admin",
      "CustomInfo": "{\"key1\":\"value1\"}"
}
}
```

请求包字段说明

字段	类型	属性	说明

RoomId	String	必填	房间 ID
Member_Account	String	必填	房间成员 ID
NameCard	String	选填	房间成员昵称，最长32个字节
Role	String	选填	房间成员权限：Admin（管理员），Member（普通成员）
CustomInfo	String	选填	房间成员自定义字段，最长60个字节，仅支持 JSON 类型字符串

应答包体示例

- **基础形式**

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-8d5f107ce00f420597a2db9fbe7f71ca-0-Seq-102831"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员
100006	操作权限不足

转让房主

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口转让房主。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/change_room_owner?
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
v4/room_engine_http_srv/change_room_owner	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "From_Account":"user2",
  "NewOwner_Account":"user4"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
From_Account	String	必填	原房主

NewOwner_Account	String	必填	新房主，必须是房间成员

应答包体示例

- 基础形式

```
{
    "ErrorCode": 0,
    "ErrorInfo": "",
    "ActionStatus": "OK",
    "RequestId": "Id-8d5f107ce00f420597a2db9fbe7f71ca-0-Seq-102831"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员

设置房间成员标签

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口设置成员标签。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/mark_room_member?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/mark_room_member	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "CommandType":1,
  "MemberList":[
      {
          "Member_Account":"user2",
          "Marks":[1000,1002,1003]
      }
  ]
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
CommandType	String	必填	1：设置标记，2：删除标记
MemberList	Array	必填	需要设置的账号列表，每次可以设置最多不超过10个账号
Marks	Array	必填	标签为大于等于1000的数字，每个房间最多只能设置10个不同的标记，单次设置标签不超过3个

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-ee0a1a5f2c70432f8273a2b279a5fa8f-O-Seq-57703"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

100009	标签个数超上限

版权所有：腾讯云计算（北京）有限责任公司

房间成员封禁

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口封禁房间成员，封禁后成员将会被移出房间，并且在封禁时间内无法再次进入。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/ban_room_member?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/ban_room_member	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "MemberList_Account":["user1"],
  "Reason":"you are banned because of irregularities",
  "Duration":3600
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

MemberList_Account	Array	必填	封禁成员的 ID，单次请求最大20个限制
Reason	String	必填	封禁信息，最大长度1000字节
Duration	Integer	必填	封禁时长，单位：秒

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-O-Seq-47158"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

房间成员解封

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口解封房间成员，解封后，之前封禁的成员可重新进房间。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/unban_room_member?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/unban_room_member	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "MemberList_Account":["user1"]
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
MemberList_Account	Array	必填	解封成员的 ID，单次请求最大20个限制

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-0-Seq-47158"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	非房间成员
100006	操作权限不足

获取被封禁房间成员列表

最近更新时间：2025-07-02 20:53:41

功能说明

App 管理员可以通过该接口获取被封禁的成员列表。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/get_banned_member_list?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/get_banned_member_list	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "Limit":100,
  "Offset":0
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
Limit	Integer	选填	单次获取封禁成员的数量，最大为100

Offset	Integer	选填	偏移量

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-99d6df2213b54216a02bc18024a5c218-0-Seq-57681",
  "Response": {
      "BannedAccountList": [
          {
              "Member_Account": "user2",
              "BannedUntil": 1703559417
          }
      ],
      "NextOffset": 0
  }
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId
BannedAccountList	Array	封禁成员信息，BannedUntil 为封禁成员的封禁截止时间，Member_Account 为封禁成员账号 ID
NextOffset	Integer	下次请求的 Offset 值，如果此值为0，表示该封禁成员列表已经拉取完毕

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员

100006	操作权限不足

版权所有：腾讯云计算（北京）有限责任公司

踢出房间成员

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口将成员踢出房间。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/kick_user_out?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/kick_user_out	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "MemberList_Account":["user1"],
  "Reason":"you are kicked because of irregularities"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

MemberList_Account	Array	必填	踢出成员的 ID，单次请求最大20个限制
Reason	String	必填	踢出信息，最大长度1000字节

应答包体示例

- 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-O-Seq-47158"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员
100006	操作权限不足

连麦管理
获取麦位列表

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口获取麦位列表。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/get_seat_list?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_mic/get_seat_list	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717",
  "Response": {
      "SeatList": [
          {
              "Index": 0,
              "Member_Account": "user2",
              "IsTakenDisabled": false,
              "IsVideoDisabled": false,
              "IsAudioDisabled": false
          },
          {
              "Index": 1,
              "IsTakenDisabled": false,
              "IsVideoDisabled": false,
              "IsAudioDisabled": false
          }
      ]
  }
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId
SeatList	Array	麦位列表
Index	Integer	麦位号
Member_Account	String	麦位上的用户 ID，空表示麦上没有用户
IsTakenDisabled	Bool	麦位锁定
IsVideoDisabled	Bool	禁止麦位视频流
IsAudioDisabled	Bool	禁止麦位音频流

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散

版权所有：腾讯云计算（北京）有限责任公司

用户上麦

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口让指定成员上麦。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/pick_user_on_seat?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_mic/pick_user_on_seat	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "Member_Account":"user2",
  "Index":1
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
Index	Integer	必填	麦位号

Member_Account	String	必填	上麦用户 ID，需要为房间成员

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100005	上麦用户非房间成员
100200	麦位已锁定，可以尝试换一个麦位
100202	已经处于排麦状态
100203	已经处于麦上状态
100205	麦位已满
100210	已有用户在麦位上
100211	该房间不支持连麦

可能触发的回调

- 麦位列表变更之后回调

用户下麦

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口让指定成员下麦。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/kick_user_off_seat?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_mic/kick_user_off_seat	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "Member_Account":"user2",
  "Data":"time over"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID
Member_Accou	String	必填	麦上用户 ID

nt			
Data	String	必填	自定义信息，该信息踢人下麦下行通知时下发

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散

可能触发的回调

- 麦位列表变更之后回调

锁定麦位

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口锁定麦位，麦位锁定后，用户无法再上此麦位。锁定麦位时，如果麦上有人，会先将用户下麦，再锁定。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/lock_seat?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_mic/lock_seat	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "RoomId":"room-test",
  "Index":2,
  "ForbidTaken":true,
  "ForbidVideo":true,
  "ForbidAudio":true
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID

Index	Integer	必填	麦位号
ForbidTaken	Bool	选填	锁定麦位
ForbidVideo	Bool	选填	禁止视频流
ForbidAudio	Bool	选填	禁止音频流

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	请参数非法，请根据错误描述检查请求是否正确
100004	房间不存在，或者曾经存在过，但是目前已经被解散
100211	该房间不支持连麦

可能触发的回调

- 麦位列表变更之后回调

机器人管理
添加机器人

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口添加房间内机器人。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/add_robot?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/add_robot	请求接口。
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID。
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员 。
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

```
{
  "RoomId": "room-test",
  "Member_Account": "user1"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID。

Member_Account	String	选填	机器人ID。

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-52f049ba12d741b7a84959700acdb1d5-O-Seq-56287"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败。
ErrorCode	Integer	错误码，0表示成功，非0表示失败。
ErrorInfo	String	错误信息。
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId。

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试。
100002	请参数非法，请根据错误描述检查请求是否正确。
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员。
100006	操作权限不足。
100700	机器人数已达到上限。

版权所有：腾讯云计算（北京）有限责任公司

删除机器人

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口删除房间内机器人。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_srv/delete_robot?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_engine_http_srv/delete_robot	请求接口。
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID。
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员 。
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

```
{
  "RoomId": "room-test",
  "Member_Account": "user1"
}
```

请求包字段说明

字段	类型	属性	说明
RoomId	String	必填	房间 ID。
Member_Account	String	选填	机器人ID。

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-52f049ba12d741b7a84959700acdb1d5-0-Seq-56287"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败。
ErrorCode	Integer	错误码，0表示成功，非0表示失败。
ErrorInfo	String	错误信息。
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId。

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试。
100002	请参数非法，请根据错误描述检查请求是否正确。
100004	房间不存在，或者曾经存在过，但是目前已经被解散。
100005	非房间成员。
100006	操作权限不足。

第三方回调
第三方回调简介

最近更新时间：2024-03-19 09:31:22

为了方便您控制 App 的功能形态，RoomKit 提供了回调能力。

功能说明

用户可通过 REST API 方式配置指定 Url 的回调，当执行的 CallbackCommand 在配置列表中，将触发回调。

注意事项

- 一个 sdkAppId 仅允许配置一个回调。
- 确保回调的 URL 是可以正常访问。

版权所有：腾讯云计算（北京）有限责任公司

回调命令列表

最近更新时间：2024-08-23 17:18:01

房间相关

回调类型	回调命令字
创建房间之后回调	Room.CallbackAfterCreateRoom
解散房间之后回调	Room.CallbackAfterDestroyRoom
更新房间信息之后回调	Room.CallbackAfterUpdateRoomInfo

用户相关

回调类型	回调命令字
用户进房之后回调	Room.CallbackAfterMemberEnter
用户退房之后回调	Room.CallbackAfterMemberLeave

连麦相关

回调类型	回调命令字
麦位列表变更之后回调	Mic.CallbackAfterSeatInfoChanged

版权所有：腾讯云计算（北京）有限责任公司

回调配置
查询回调配置

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口创建回调。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_config/get_callback?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_config/get_callback	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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
传空即可。

```
{
}
```

请求包字段说明

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
```

```
    "ErrorInfo": "",
    "ActionStatus": "OK",
    "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-0-Seq-61309",
    "Response": {
        "Url": "http://www.example.com/callback",
        "CallbackCommandList": [
            "Room.CallbackAfterCreateRoom",
            "Room.CallbackAfterDestroyRoom",
            "Room.CallbackUpdateRoomInfo",
            "Mic.CallbackAfterMemberEnter",
            "Room.CallbackAfterMemberLeave",
            "Mic.CallbackAfterSeatInfoChanged"
        ]
    }
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId
Url	String	回调的地址，需要以 http/https 开头，建议使用更为安全的 https
CallbackCommandList	Array	需要触发回调的命令字，参见 回调命令字列表

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	参数非法，请根据错误描述检查请求是否正确
100301	回调配置不存在，可以时间创建回调配置接口进行创建

创建回调配置

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口创建回调配置。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_config/set_callback?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_config/set_callback	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "Url":"http://www.example.com/callback",
  "CallbackCommandList":[
      "Room.CallbackAfterMemberLeave",
      "Mic.CallbackAfterSeatInfoChanged"
  ]
}
```

请求包字段说明

字段	类型	属性	说明
Url	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

CallbackCommandList	Array	必填	需要触发回调的命令字，参见 回调命令字列表

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	参数非法，请根据错误描述检查请求是否正确
100300	回调配置已存在，可以使用更新回调配置接口进行更新

更新回调配置

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口创建回调。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_config/update_callback?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_config/update_callback	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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

```
{
  "Url":"http://www.example.com/callback",
  "CallbackCommandList":[
      "Room.CallbackAfterMemberLeave",
      "Mic.CallbackAfterSeatInfoChanged"
  ]
}
```

请求包字段说明

字段	类型	属性	说明
Url	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

CallbackCommandList	Array	必填	需要触发回调的命令字，参见 回调命令字列表

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	参数非法，请根据错误描述检查请求是否正确
100301	回调配置不存在，可以时间创建回调配置接口进行创建

删除回调配置

最近更新时间：2025-07-02 20:53:42

功能说明

App 管理员可以通过该接口删除回调配置。

接口调用说明

请求 URL 示例

```
https://console.tim.qq.com/v4/room_config/delete_callback?
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，更多参数详情请参见 REST API 简介 。

参数	说明
v4/room_config/delete_callback	请求接口
sdkappid	创建应用时即时通信 IM 控制台分配的 SDKAppID
identifier	必须为 App 管理员账号，更多详情请参见 App 管理员
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
传空即可。

```
{
}
```

请求包字段说明

无。

应答包体示例

基础形式

```
{
```

```
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-O-Seq-61309"
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	错误码，0表示成功，非0表示失败
ErrorInfo	String	错误信息
RequestId	String	唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId

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。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。
公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
100001	服务器内部错误，请重试
100002	参数非法，请根据错误描述检查请求是否正确

房间相关
创建房间之后回调

最近更新时间：2025-04-07 09:48:12

功能说明

App 后台可以通过该回调实时查看用户创建房间的信息，包括：App 后台有房间创建成功通知时，可以据此进行数据同步等操作。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户通过客户端创建房间成功
- App 管理员通过 REST API 创建房间成功

回调发生时机

房间创建成功之后。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Room.CallbackAfterCreateRoom
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

请求包示例

```
{
    "CallbackCommand":"Room.CallbackAfterCreateRoom",
    "Operator_Account":"admin",
    "RoomInfo":{
        "RoomId":"tandy-test-rest",
        "RoomName":"tandy-test-rest",
        "RoomType":"Meeting",
        "Owner_Account":"user3",
        "MaxMemberCount":300,
        "MaxSeatCount":16,
        "IsVideoDisabled":true,
        "IsAudioDisabled":true,
        "IsMessageDisabled":true,
        "IsScreenSharingDisabled":true,
        "IsCloudRecordingDisabled":true,
        "CustomInfo":"custom123",
        "ScheduleStartTime":1703589922,
        "ScheduleEndTime":1703593522,
        "RoomStatus":"Running",
        "IsSeatEnabled":true,
        "TakeSeatMode":"ApplyToTake"
    },
    "ScheduleInviteeList_Account":["user2", "user3", "user3"],
    "EventTime":1703589922780
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起创建群组请求的操作者 UserID
RoomId	String	房间 ID
RoomName	String	房间名称
RoomType	String	房间类型：Meeting（会议房间）
Owner_Account	String	房主 ID
MaxMemberCount	Integer	最大房间成员数量
ScheduleStartTime	Integer	预定会议开始时间
ScheduleStartTime	Integer	预定会议结束时间
IsVideoDisabled	Bool	全体禁画
IsAudioDisabled	Bool	全体禁音

版权所有：腾讯云计算（北京）有限责任公司

IsMessageDisabled	Bool	禁止所有成员发文本消息
IsScreenSharingDisabled	Bool	禁止屏幕共享
IsCloudRecordingDisabled	Bool	禁止云录制
CustomInfo	String	自定义字段
RoomStatus	String	房间状态：None（关闭），NotStarted（待开始），Running（进行中）
IsSeatEnabled	Bool	是否支持麦位
MaxSeatCount	Integer	最大麦位数
TakeSeatMode	String	麦位模式：None（关闭），FreeToTake（自由上麦），ApplyToTake（申请上麦）
EventTime	Integer	事件触发的毫秒级别时间戳

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

参考

- 第三方回调简介
- REST API：创建房间

解散房间后的回调

最近更新时间：2024-09-26 11:02:51

功能说明

App 后台可以通过该回调实时查看房间的解散动态，包括：对房间的解散实时记录（例如记录日志，或者同步到其他系统）。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户通过客户端解散房间成功。
- App 管理员通过 REST API 解散房间成功。

回调发生时机

房间销毁成功之后。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Room.CallbackAfterDestroyRoom
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

请求包示例

```
{
    "CallbackCommand":"Room.CallbackAfterDestroyRoom",
    "Operator_Account":"admin",
    "RoomId":"tandy-test-rest",
    "EventTime":1703589922780
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起销毁房间请求的操作者 UserID
RoomId	String	房间 ID
EventTime	Integer	事件触发的毫秒级别时间戳

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

参考

- 第三方回调简介
- REST API：解散房间

更新房间信息之后的回调

最近更新时间：2024-09-26 11:02:51

功能说明

App 后台可以通过该回调实时查看房间信息的变更，包括对变动房间信息的实时记录（例如记录日志，或者同步到其他系统）。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户通过客户端变动房间信息。
- App 管理员通过 REST API 变动房间信息。

回调发生时机

房间信息变动成功之后。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Room.CallbackAfterUpdateRoomInfo
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

请求包示例

```
{
    "CallbackCommand":"Room.CallbackAfterUpdateRoomInfo",
    "Operator_Account":"bob",
    "RoomInfo" : {
        "RoomId":"rid-123",
        "RoomName" : "rname-123",
        "Owner_Account" : "jack",
        "TakeSeatMode" : "FreeToTake",
        "MaxMemberCount" : 300, // 最大房间人数
        "IsVideoDisabled" : false, //是否开启全体禁画 默认false
        "IsAudioDisabled" : false, //是否开启全体静音 默认false
        "IsMessageDisabled" : false, //是否禁止发文本消息 默认false
        "IsScreenSharingDisabled" : false, //是否禁止屏幕共享 默认false
        "IsCloudRecordingDisabled" : "false", //是否禁止云录制 默认false
        "CustomInfo" : "123", // 房间自定义信息
    }
    "EventTime":1670574414123//毫秒级别，事件触发时间戳
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起创建群组请求的操作者 UserID
RoomId	String	房间 ID
RoomName	String	房间名称
Owner_Account	String	房主 ID
MaxMemberCount	Integer	最大房间成员数量
IsVideoDisabled	Bool	全体禁画
IsAudioDisabled	Bool	全体禁音
IsMessageDisabled	Bool	禁止所有成员发文本消息
IsScreenSharingDisabled	Bool	禁止屏幕共享
IsCloudRecordingDisabled	Bool	禁止云录制
CustomInfo	String	自定义字段
EventTime	Integer	事件触发的毫秒级别时间戳

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
```

```
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

参考

- 第三方回调简介
- REST API：更新房间资料

用户相关
用户进房之后的回调

最近更新时间：2024-09-26 11:02:51

功能说明

App 后台可以通过该回调实时查看房间成员加入的消息，包括：通知 App 后台有成员入群，App 可以据此进行必要的数据同步。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户通过客户端主动加入房间。

回调发生时机

用户加入房间成功之后

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Room.CallbackAfterMemberEnter
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

请求包示例

```
{
    "CallbackCommand":"Room.CallbackAfterMemberEnter",
    "Operator_Account":"user1",
    "RoomId":"rid-123",
    "MemberCount":20,
    "Type":"Enter", // 加入房间方式：Enter（自己加入）
    "MemberList_Account":["user1"],
    "EventTime":1670574414123//毫秒级别，事件触发时间戳
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起请求的操作者 UserID
RoomId	String	房间 ID
MemberCount	Integer	房间人数
Type	String	加入房间方式： Enter（自己加入）
MemberList_Account	Array	加入房间成员列表
EventTime	Integer	事件触发的毫秒级别时间戳

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

参考

- 第三方回调简介

用户退房之后的回调

最近更新时间：2024-09-26 11:02:51

功能说明

App 后台可以通过该回调实时查看房间成员退出的消息，包括：通知 App 后台有成员退出房间，App 可以据此进行必要的数据同步。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户通过客户端主动退出房间。
- App 用户被踢出房间。
- App 用户心跳过期导致退出房间。

回调发生时机

用户退出房间成功之后。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Room.CallbackAfterMemberLeave
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

请求包示例

```
{
  "CallbackCommand":"Room.CallbackAfterMemberLeave",
  "Operator_Account":"user1",
  "RoomId":"rid-123",
  "MemberCount":20,

  "Type":"Leave",
  "MemberList_Account":["user1"],
  "Reason":"xxxx",
  "EventTime":1670574414123//毫秒级别，事件触发时间戳
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起请求的操作者 UserID
RoomId	String	房间 ID
MemberCount	Integer	房间人数
Type	String	退房方式：Leave(自己离开)；Kicked(被踢)；Expired(心跳过期)
MemberList_Account	Array	退出房间成员列表
Reason	String	退房原因
EventTime	Integer	事件触发的毫秒级别时间戳

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

版权所有：腾讯云计算（北京）有限责任公司

参考

- 第三方回调简介

连麦相关
麦位列表变更之后回调

最近更新时间：2024-09-26 11:02:51

功能说明

App 后台可以通过该回调实时查看麦位列表变更消息。

注意事项

- 要启用回调，必须通过 REST API 配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 创建回调配置 文档。
- 回调的方向是 Room 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 用户上下麦。
- 操作麦位锁定。

回调发生时机

麦位信息变更之后。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPl
atform=$OptPlatform
```

请求参数说明

参数	说明
https	请求协议为 HTTPS，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
CallbackCommand	固定为 Mic.CallbackAfterSeatInfoChanged
contenttype	固定值为 JSON
ClientIP	客户端 IP，格式如：127.0.0.1
OptPlatform	客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义

版权所有：腾讯云计算（北京）有限责任公司

请求包示例

```
{
    "CallbackCommand":"Mic.CallbackAfterSeatInfoChanged",
    "Operator_Account":"user1",
    "RoomId":"rid-123",
    "SeatList":[
        {
            // 麦位编号
            "Index": 1,
            // 如果当前麦位有人占用，则返回对应用户的 ID
            "Member_Account": 144115233775727695,
             // false:允许上麦 true:禁止上麦
             "IsTakenDisabled": false,
             // false:允许推视频流 true:禁止推视频流
             "IsVideoDisabled": false,
             // false:允许推音频流 true:禁止推音频流
             "IsAudioDisabled": false
         }
    ]
    "EventTime":1670574414123//毫秒级别，事件触发时间戳
}
```

请求包字段说明

字段	类型	说明
CallbackCommand	String	回调命令
Operator_Account	String	发起请求的操作者 UserID
RoomId	String	房间 ID
SeatList	Array	麦位列表
Index	Integer	麦位号
Member_Account	String	麦位上的用户 ID，空表示麦上没有用户
IsTakenDisabled	Bool	麦位锁定
IsVideoDisabled	Bool	禁止麦位视频流
IsAudioDisabled	Bool	禁止麦位音频流

应答包示例

App 后台同步数据后，发送回调应答包。

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 忽略回调结果
}
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

应答包字段说明

字段	类型	属性	说明
ActionStatus	String	必填	请求处理的结果，OK 表示处理成功，FAIL 表示失败
ErrorCode	Integer	必填	错误码，此处填0表示忽略应答结果
ErrorInfo	String	必填	错误信息

参考

- 第三方回调简介

发布日志（TUIRoomKit）

最近更新时间：2025-05-19 15:51:42

2025年04月

发布动态	更新日志	发布时间
3.0.0 iOS、Android、Flutter、Web、Electron 版本正式发布	**RoomEngine** • 全平台错误码调整。 **RoomKit** Electron&Web&小程序： • Web、Electron 更新 Button 组件。 • Web 优化媒体设备（摄像头、麦克风、扬声器）检测及配置逻辑。 Android&iOS&Flutter： • Android 会议回调名称 onConferenceExisted 改为 onConferenceExited。 • Android 优化文件目录结构。 • Android 修复使用 manifestPlaceholders 设置 application label 时 crash 的问题。 • Flutter 修复偶现房间内已上麦成员无法开关音视频的问题。 • Flutter 修复重复进房后房间状态异常的问题。 • iOS 视频流、成员列表添加排序逻辑。	2025-04-07

2025年02月

发布动态	更新日志	发布时间
2.9.1 iOS、Android、Flutter、Web、Uniapp (小程序)、Uniapp (客户端)、Electron 版本正式发布	**RoomKit** Electron&Web&小程序： • Web、Electron 支持自定义颜色主题换肤； • Web、Electron、小程序删除主动接口 setMaxSeatCount、startPushLocalAudio、stopPushLocalAudio； • Web、Electron、小程序删除回调接口：onRoomMaxSeatCountChanged、onRoomDismissed、onUserRoleChanged、onKickedOffSeat、onRequestCancelled、onRequestProcessed。 • Web、Electron、小程序预定会议获取会议列表产生超频的问题； • 修复 Web、Electron Vue2 下全屏展示异常的问题；	2025-02-27
2.8.3 Uniapp (客户端)	**RoomKit** 修复房间人数超过 50 人情况下，拉取成员信息异常问题。	2025-02-08

2024年12月

发布动态	更新日志	发布时间
SDK 2.8.2 iOS、Android、Flutter、Web、Uniapp (小程序)、	**RoomKit** Electron&Web&小程序： • Web、Electron、小程序 conference 实例新增 setParticipants 接口，用于设置预定会议或者会中呼叫的邀请人列表 • 修复 vue2 + webpack 源码接入 input、select 组件加载错误	2024-12-27

Uniapp (客户端)、 Electron 版本正式发布		
SDK 2.8.1 iOS、Android、 Flutter、Web、 Uniapp (小程序)、 Uniapp (客户端)、 Electron 版本正式发布	**RoomKit** Electron&Web&小程序： • Web、Electron、小程序 支持房间内只有主持人一个人时离开房间 • Web、Electron、小程序 支持自定义用户排序和流排序 • Web、Electron 修复被禁止文字聊天时，发送消息失败提示语和当前语言类型不一致问题 • Web、Electron 修复举手上台 Notification 在成员进退房时重复展示的问题 • Web、Electron 修复在没有开启全体禁止屏幕共享下，主持人转移房主会关闭掉屏幕共享问题 • Web、Electron 修复流播放偶现黑屏问题 • Web、Electron 修复 vue2 下预定会议模块联系人列表不更新问题 • Web、Electron 移除房间内右上角用户信息模块 • Web、Electron、小程序 修复 roomStore 数据错乱，房间内偶现出现多名成员 • Web、Electron、小程序 修复入会密码弹窗中加入按钮频繁点击出现进入房间失败的问题 Android&iOS： • Android、iOS 弹幕中增加发送文件、图片、视频的提示； • Android&iOS 房间内只有房主一人时，允许房主离开房间，不解散房间； • Android 8.0 及以上支持画中画； • Android 优化流布局排序，将有音视频流的用户提前； • Android 解决小概率不显示视频流画面的问题； • Android 解决小概率内存泄露问题； • iOS 新增支持网格布局的 MultiStreamView，用于展示多个画面，满足会议中多画面并行显示的需求。支持动态添加、移除或调整网格中的子视图，灵活适配不同的显示场景。 • iOS 新增支持单画面显示的 SingleView，适用于强调某一特定画面的场景；用户可通过切换视图模式，在单画面与多画面之间快速切换。 • iOS 支持灵活布局，可自由组合 MultiStreamView 和 SingleView，满足多样化的布局需求；提供多种预设布局方式，并支持实时调整画面排列顺序。	2024-12-18

2024年11月

发布动态	更新日志	发布时间
SDK 2.7.2 iOS、Android、 Flutter、Web、 Uniapp (小程序)、 Uniapp (客户端)、 Electron 版本正式发布	**RoomKit** Electron&Web： • Electron & Web AI 字幕后台升级优化，客户端同步修改数据结构 iOS&Android： • 修复偶现预定会议详情页面，成员列表不准确问题 • 修复退出会议列表页面后，重新进入该列表页面，再次点击进入会议出错	2024-11-15

SDK 2.7.0 iOS、Android、Flutter、Web、Uniapp (小程序)、Uniapp (客户端)、Electron 版本正式发布	**RoomEngine SDK** 全平台： • 新增 EnableSystemAudioSharing 接口，支持系统混音能力 • 优化弱网环境下麦位数据更新，增加缓存和 sequence 验证 • 会中邀请功能 GetInvitationList 接口调整为进房前也可拉取邀请信息，仅被邀请用户可拉取 **RoomKit** Electron&Web： • Electron 白板工具栏优化改造 • 小程序端、H5 端支持会中呼叫 • 小程序端、H5 端支持预定会议设置入会密码 • 小程序端支持扬声器和麦克风切换 • 小程序端、H5 端支持屏幕分享双指放大	2024-11-06

2024年09月

发布动态	更新日志	发布时间
SDK 2.6.3 iOS、Android、Web、Uniapp (小程序)、Uniapp (客户端)、Electron 版本正式发布	**RoomKit** Electron&Web： • Web 端支持基础美颜能力。 • Web、Electron 端支持发送图片、视频、文件、大表情消息能力。 • Web、Electron 端支持入会密码能力。 • Web 端支持 AI 实时字幕、会议记录能力 。（更多内容可加入 腾讯云实时音视频 TRTC 技术交流群 进行咨询 ） iOS： • 修复发送弹幕消息失败，提示参数无效的问题。 Android： • 支持悬浮窗音频焦点能力。	2024-09-27
SDK 2.6.0 iOS、Android、Flutter、Web、Uniapp (小程序)、Uniapp (客户端)、Electron 版本正式发布	**RoomEngine SDK** 全平台： • 新增会议场景 会中呼叫 `TUIConferenceManager` 插件，支持会中邀请功能。 • 新增 `setRoomMetadataByAdmin` 接口，支持Room自定义房间信息设置。 • 新增 `fetchRoomInfo` 接口，支持房间外获取房间信息 。 • 新增 `onUserInfoChanged` 回调通知 `nameCard/userRole` 变更。 • 修复房主变更本地缓存异常问题。 Electron&Web： • Electron 端支持白板能力。 • H5、小程序端支持预定会议能力。 • Web、Electron 端支持会中邀请能力。 • Web、Electron 端支持入会密码能力。 • Web、Electron 端流布局支持视频流在多节点播放，优化视频流懒加载逻辑。 • H5 端流布局支持视频流懒加载及大小流逻辑，优化流布局页面切换体验。 • 支持房间结束后 5s 自动退出房间。 接口变更： • 删除 connectOtherRoom/disconnect 以及回调。	2024-09-14

	• 删除 sendTextMessage/sendCustomMessage 以及回调。	

2024年07月

发布动态	更新日志	发布时间
SDK 2.5.0 iOS、Android、 Flutter、Web、 Uniapp (小程序)、 Uniapp (客户端)、 Electron 版本正式发布	**RoomEngine SDK** 全平台： • 新增 `TUIConferenceListManager` ,支持会议列表、预定会议功能。 • `TUIRoomInfo` 新增字段 `password` 字段，支持入会密码功能。 • `TUIUserInfo` 、 `TUISeatInfo` 、 `TUIRequest` 中新增 `nameCard` 字段标识房间内昵称。 • 新增更新房间内昵称接口 `changeUserNameCard` 。 • 新增 `onUserInfoChanged` 回调通知 `nameCard/userRole` 变更。 • 删除 `speechMode` 相关类型定义、接口、回调`。 Android&iOS&Flutter： • `TUILiveListManager` 中 `setLiveInfo` 接口与 `onLiveInfoChanged` 回调 中参数 `modifyFlag` 改为 `modifyFlagList` 。 • `TUIListListManager` 中 `setObserver` 接口变更为 `addObserver` 。 Web： • 修复 `web` 及 `electron` 无设备权限或设备采集失败错误信息回调异常。 • `web` 及 `electron` 支持多实例。	2024-07-29

2024年06月

发布动态	更新日志	发布时间
SDK 2.4.0 iOS、Android、 Flutter、Web、 Uniapp (小程序)、 Uniapp (客户端)、 Electron 版本正式发布	**RoomEngine SDK** 全平台： • 新增移麦接口 moveToSeat。 • 新增直播列表插件 TUILiveListManager。 • 修复 updateVideoQualityEx 接口编码参数失效问题。 Android&iOS&Flutter： • 开启IM SDK 忽略重复登录特性。 • 废弃发送消息接口sendTextMessage，建议使用 IMSDK sendGroupTextMessage 接口代替。 • 废弃发送消息接口sendCustomMessage，建议使用 IMSDK sendGroupCustomMessage 接口代替。 • 废弃接收文本消息回调 onReceiveTextMessage，建议使用IMSDK onRecvGroupTextMessage 回调代替。 • 废弃接收自定义消息回调 onReceiveCustomMessage，建议使用 IMSDK onRecvGroupCustomMessage 回调代替。 **RoomKit** iOS： • 调整最低支持版本到 iOS12，使用此功能，请在 Podfile 中添加 `pod 'TUIRoomKit/TRTC_iOS12'` ，然后运行 `pod install` 。 • 增加申请上台的通知，区分已上台和未上台用户。	2024-06-06

- 上台发言模式下，优化麦位已占满情况下邀请上台或申请上台时的交互表现。
- 优化视频显示，防止 iOS17 版本出现黑屏现象。
- 优化弹幕功能交互。
- 优化顶部栏标题。

Android：
- 增加申请上台的通知，区分已上台和未上台用户。
- 上台发言模式下，优化麦位已占满情况下邀请上台或申请上台时的交互表现。
- 优化弹幕功能交互。
- 优化顶部栏标题。

Flutter：
- 适配横屏模式。
- 将 **ConferenceMainPage** 中的参数 roomId 修改为 conferenceId，将 isCreateRoom 修改为 isCreateConference。
- 修改默认语言为系统语言。
- 优化 **ConferenceObserver** 中各回调参数。
- 优化大小窗布局的拖动。
- 优化顶部栏标题。

Electron：
- 上台发言模式下，优化麦位已占满情况下邀请上台或申请上台时的交互表现。
- 修复用户角色从管理员变成房主，成员的操作列表会被遮挡的问题。
- 修复 startCameraDeviceTest 接口进行摄像头预览时，切换摄像头设备未释放旧设备占用的问题。
- 修复 createRoom 时，设置房间名称不生效的问题。
- 修复 RoomKit 展示区域 resize 时，水印未更新导致样式异常问题。

web：
- 虚拟背景功能交互优化: 新增 loading 效果，新增保存按钮。
- 上台发言模式下，优化麦位已占满情况下邀请上台或申请上台时的交互表现。
- 修复用户角色从管理员变成房主，成员的操作列表会被遮挡的问题。
- 修复 createRoom 时，设置房间名称不生效的问题。
- 修复 RoomKit 展示区域 resize 时，水印未更新导致样式异常问题。

Uniapp (小程序)：
- 修复发送表情时，输入框没有背景色问题。
- 修复 createRoom 时，设置房间名称不生效的问题。

Uniapp (客户端)：
- Uniapp (客户端) 版本含 UI 方案 RoomKit 发布，支持 android 和 ios 平台。

2024年05月

发布动态	更新日志	发布时间
SDK 2.3.3 iOS、Android、Windows、Web、Uniapp(小程序)、Electron 版本正式发布	**RoomEngine SDK** 全平台： • 修复摄像头、麦克风被占用错误码回调异常。 Web： • 回调参数优化，OnKickedOffSeat 参数 userId 调整为 seatIndex、userInfo。 • 回调参数优化，OnRequestCancelled 参数 requestId、userId 调整为 request、userInfo。 • 回调参数优化，OnRequestProcessed 参数 requestId、userId 调整为 request、userInfo。	2024-05-17

	• 回调参数优化, OnUserRoleChanged 参数 userId、userRole 调整为 userInfo。 • 回调参数优化，OnRoomDismissed 新增 TUIRoomDismissedReason。 • 新增 onError 事件错误码 -1102 代表摄像头设备被占用。 • 修复被邀请打开摄像头后，切换摄像头设备未生效。 **RoomKit** Android&iOS： • 支持会议中进行网络检测。 • 支持会议中进行弹幕聊天。 Web： • 支持设置背景模糊。 • 支持设置文字水印。 • 修复被邀请打开摄像头后, 切换摄像头设备未生效 Electron： • 支持设置文字水印。 • 修复被邀请打开摄像头后, 切换摄像头设备未生效。	

2024年04月

发布动态	更新日志	发布时间
SDK 2.3.1 iOS、Android、Windows、Web、Uniapp(小程序)、Electron 版本正式发布	**RoomEngine SDK** 全平台： • 支持直播场景，最大房间人数10w+。 • 新增房间内人数变化回调回调 onRoomUserCountChanged。 • 新增带 RoomType 的 enterRoom 接口。 • TUISeatInfo 结构体新增头像、昵称字段。 • TUIRequest 结构体新增头像、昵称字段。 • TUIUserInfo 结构体新增被禁止发送消息字段 isMessageDisabled。 Android&iOS： • 修复 startPlayRemoteVideo 内存泄漏。 • 新增参数为 seatIndex、userInfo 的 OnKickedOffSeat 回调，原回调废弃。 • 新增参数为 request、userInfo 的 OnRequestCancelled 回调，原回调废弃。 • 新增参数为request、userInfo 的 OnRequestProcessed 回调，原回调废弃。 • 新增参数为 userInfo 的 OnUserRoleChanged 回调，原回调废弃。 • 新增参数为 roomId、TUIRoomDismissedReason 的 OnRoomDismissed 回调，原回调废弃。 **RoomKit** Android&iOS： • 支持会议中设置水印。 • 优化登出逻辑，会议过程中监听 logout 的动作，退出当前会议并给出提示。 Android： • 解决上台过程中变更为房主，无法自动上台的问题。 • 解决在显示用户列表时，被踢出房间造成的内存泄露问题。 • 移除 support-v4 的第 3 方库。 iOS： • 解决双人会议过程中，切换大小窗，小窗口黑屏问题。	2024-04-29

	• 解决双人会议小窗口没有水印的问题。 Electron： • 修复 Electron 下拔出外接摄像头摄像头列表默认设备未更新的问题。 • 修复成员操作面板距顶部距离不够，展示不完全问题。 • 修复 Notification 组件收到其他成员处理事件时显隐展示的问题。 • 修复获取主持人和管理员身份后没有更新申请上麦列表的问题。 • 修复聊天消息未读数不准确的问题。 • 修复转交房主或管理员状态下多次点击麦克风和摄像头按钮 disable 状态错误的问题。 Web： • 修复 H5 移动端入会时间展示不全的问题。 • 修复 H5 移动端上台提示信息出现换行的问题。 • 修复 H5 移动端成员管理列表无法滚动问题。 • 修复成员操作面板距顶部距离不够，展示不完全问题。 • 修复 H5 移动端 onePlus 自带浏览器加载 demo 页面白屏的问题。 • 修复 Notification 组件收到其他成员处理事件时显隐展示的问题。 • 修复获取主持人和管理员身份后没有更新申请上麦列表的问题。 • 修复聊天消息未读数不准确的问题。 • 修复 MessageBox 组件进房后点击按钮无法关闭的问题。 • 修复转交房主或管理员状态下多次点击麦克风和摄像头按钮 disable 状态错误的问题。 小程序： • 修复小程序入会时间展示不全的问题。 • 修复小程序上台提示信息出现换行的问题。 • 修复小程序成员管理列表无法滚动问题。 • 修复小程序创建房间后，点击左上角 home 按钮回到主页，再次新建的房间异常。 • 修复小程序转交房主之后 icon 无法切换的问题。 • 修复获取主持人和管理员身份后没有更新申请上麦列表的问题。 • 修复聊天消息未读数不准确的问题。 • 修复 Notification 组件收到其他成员处理事件时显隐展示的问题。 • 修复转交房主或管理员状态下多次点击麦克风和摄像头按钮 disable 状态错误的问题。	
多人音视频房间（TUIRoomEngine）SDK 2.2.1 iOS、Android、Windows、Web、Uniapp(小程序)、Electron 版本正式发布	**RoomEngine SDK** 全平台： • 修复单点登录设备被踢收不到回调的问题。 • 优化单点登录设备被踢 Room 房间状态清理。 • 修复 Logout 接口调用 Room 房间异常问题。	2024.04.01

2024年03月

发布动态	更新日志	发布时间
多人音视频房间（TUIRoomEngine）SDK 2.2.0 iOS、Android、Windows、Web、	**RoomEngine SDK** 全平台： • 新增屏幕分享会控业务。 • 新增 SetCustomInfoForUser 自定义用户属性接口，可通过 getUserInfo/getUserList 获取。	2024.03.26

Uniapp(小程序)、Electron 版本正式发布	Electron： • 修复远端邀请打开摄像头时,开启本地预览无画面问题。 **RoomKit** Android： • 废弃 TUIRoomKit 类，修改接入方式，通过 Fragment 对象唤起 UI，收敛 API 到Fragment 内。 • 解决 Toast 内存泄露问题。 • 修复成为管理者后，不会自动拉取上台申请列表的问题。 • 修复被禁言用户成为房主，不会自动解除禁言的问题。 iOS： • 废弃 TUIRoomKit 类，修改接入方式，通过 ViewController 对象唤起 UI，收敛 API 到 ViewController 内。 • 增加用户被踢下线时的监听和处理。 • 修改举手发言房间，房主相互转移时，偶现同时出现两个相同 id 的情况。 • 修改在用户列表页面，被邀请开启摄像头时又被踢出房间时，有页面不销毁的问题。 Flutter： • `rtc_conference_tuikit.dart` 更名为 `rtc_conference_tui_kit.dart`。 • 废弃 `TUIRoomKit` 类，修改接入方式，请改用 `ConferenceSession`。在开始快速会议或加入会议后，您可以自行导航至会议页面。 • 增加用户的麦克风按钮和视频画面下方的麦克风图标上动态显示音量的功能。在观看屏幕共享时，右上角显示正在说话的用户。 • 修复成为管理者后，不会自动拉取上台申请列表的问题。 Electron&小程序&Web： • 更新聊天组件表情包资源。 • 优化上台发言房间 UI。 • 修复上台发言模式二次上台无法打开麦克风的问题。 • 修复上台发言模式下，普通成员成为新房主，不能处理之前旧房主未处理的上台申请的问题。 • 新增网络状态组件。 • 修复进房前音量条不显示的问题。 小程序&H5： • 修复切换前后置摄像头不生效的问题。	
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”。	2024.03.01

	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年02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01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、Uniapp(小程序)、Electron 版本正式发布	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。 Web&Electron&小程序： • 渲染 view 偶现黑屏问题修复。	2024.01.29
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	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。	2024.01.19

序)、Electron 版本正式发布	○ 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。	
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。 ○ 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。	2024.01.05
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：	2024.01.19

	• setRemoteView 偶现失败问题修复。	
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。	2023.12.19

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(TUIRoomEngine) SDK 1.5.1 版本发布	更新日志： • 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。	2023.09.27
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。	2023.09.07

	• 新增回调 onDeviceChanged（桌面端）。 • 废弃若干接口，详情见各平台头文件。	

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。 • 修复音视频状态回调时序异常问题。	2023.08.03

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。	2023.05.25

	• 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

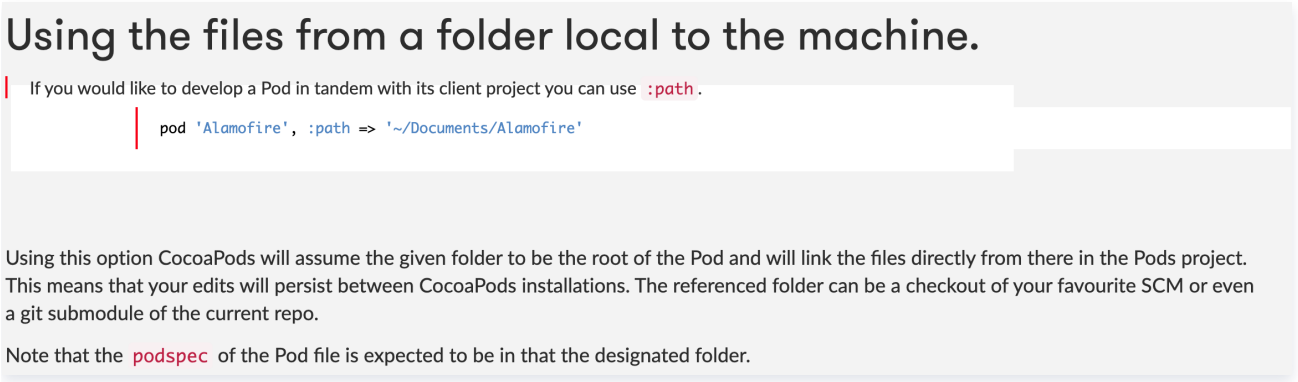
说明：
多人音视频房间 SDK（TUIRoomKit）于2023年4月8日起，正式上线。限时免费版同期结束限时免费体验，如果您在限免版发布期间接入了本产品，我们已经为您延长了一个月的限时体验时间到2023年5月8日，方便您全平台接入升级正式版，在此期间，您依旧可以使用体验本产品。
详细信息参考：产品公告 。

常见问题（TUIRoomKit）
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：



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

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：2024-12-12 18:13:12

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,
    };
  }
});
```

源码导出引用项目执行 build 指令报错

如果您在执行 `npm run build` 时，出现如下图所示错误：

```
src/TUIRoom/components/Chat/ChatKit/components/TUIChat/message-input/message-input-editor.vue:433:23 - error TS2345: Argument of type 'JSONContent | undefined' is not
ble to parameter of type 'JSONContent'.
  Type 'undefined' is not assignable to type 'JSONContent'.

433   handleEditorContent(editorJSON, content);
                          ~~~~~~~~~~

src/TUIRoom/components/Chat/ChatKit/components/TUIContact/contact-info/index.vue:417:19 - error TS2339: Property 'infoKeyList' does not exist on type 'IContactInfoTyp
  Property 'infoKeyList' does not exist on type 'IBlackListUserItem'.

417   if (contactInfo.infoKeyList) {
                      ~~~~~~~~~~

src/TUIRoom/components/Chat/ChatKit/components/TUIContact/contact-info/index.vue:418:45 - error TS2339: Property 'infoKeyList' does not exist on type 'IContactInfoTyp
  Property 'infoKeyList' does not exist on type 'IBlackListUserItem'.

418     contactInfoMoreList.value = contactInfo.infoKeyList.map((key: string) => {
                                                ~~~~~~~~~~

src/TUIRoom/components/Chat/ChatKit/components/TUIContact/contact-info/index.vue:422:19 - error TS2339: Property 'btnKeyList' does not exist on type 'IContactInfoType
  Property 'btnKeyList' does not exist on type 'IBlackListUserItem'.
```

此时需要修改配置文件 `tsconfig.json` 的 exclude 从而忽略 TUIRoom 文件夹，您可使用如下推荐配置：（vite + vue3）

```
{
  "compilerOptions": {
    "target": "esnext",
    "useDefineForClassFields": true,
    "module": "esnext",
    "moduleResolution": "node",
    "strict": true,
    "jsx": "preserve",
    "jsxImportSource": "vue",
    "sourceMap": true,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "lib": ["esnext", "dom", "dom.iterable"],
    "allowJs": true
  },
  "files": [],
  "include": [],
  "exclude": ["src/TUIRoom"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

如果上述修改仍然不生效，您可以直接修改打包指令 `vue-tsc --noEmit --skipLibCheck && vite build` 为 `vite build`。（package.json）

如果您项目使用的是 webpack + vue2，并且报错如下图所示：

```
  51:35  error  There should be no line break before or after '='  operator-linebreak
  56:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
  57:73  error  Missing trailing comma                             comma-dangle
  75:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
  76:78  error  Missing trailing comma                             comma-dangle
  93:12  error  Missing trailing comma                             comma-dangle
 102:33  error  Missing trailing comma                             comma-dangle
 109:35  error  There should be no line break before or after '='  operator-linebreak
 114:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
 115:66  error  Missing trailing comma                             comma-dangle
 142:32  error  Missing trailing comma                             comma-dangle
 145:5   error  Expected no linebreak before this statement        nonblock-statement-body-position
 146:74  error  Missing trailing comma                             comma-dangle
 150:35  error  There should be no line break before or after '='  operator-linebreak
 155:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
 156:65  error  Missing trailing comma                             comma-dangle
 169:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
 170:70  error  Missing trailing comma                             comma-dangle
 186:12  error  Missing trailing comma                             comma-dangle
 200:35  error  There should be no line break before or after '='  operator-linebreak
 205:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
 220:32  error  Missing trailing comma                             comma-dangle
 224:7   error  Expected no linebreak before this statement        nonblock-statement-body-position
 225:66  error  Missing trailing comma                             comma-dangle
 243:12  error  Missing trailing comma                             comma-dangle
```

除了上述配置，您还需要额外修改 `.eslintignore` 文件，添加 TUIRoom 文件夹进行忽略。（如没有需手动在项目根目录创建）

```
TUIRoom
```

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。

Electron

最近更新时间：2024-05-30 12:48:51

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 工程模板支持的 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 。

trtc-electron-sdk 是否兼容官方 Electron v12.0.1 版本?

兼容的，trtc-electron-sdk 没有特别依赖 electron 自身的 SDK，所以没有相关的版本依赖。

小程序

最近更新时间：2024-04-03 13:35:02

接入过程中 pinia 报错?

报错原因：RoomKit 依赖了 pinia，而 pinia 与 vue 版本存在兼容性问题。为解决此问题，请安装指定版本的 pinia。

```
npm i pinia@2.0.36 --legacy-peer-deps
```

```
html
"hasInjectionContext" is not exported by "node_modules/vue-demi/lib/index.mjs", imported by "node_modules/pinia/dist/pinia.mjs".
at ../node_modules/pinia/dist/pinia.mjs:6:9
  4:  * @license MIT
  5:  */
```

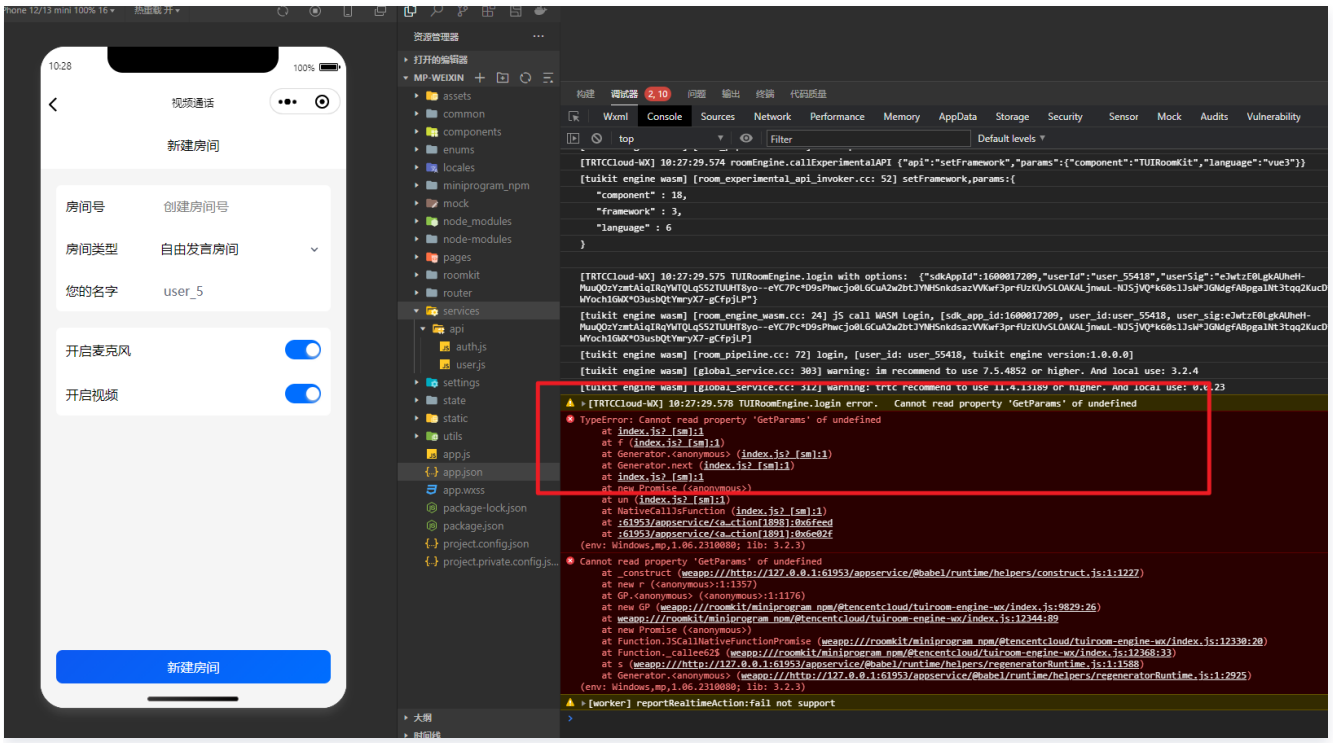
```
uni-app 有新版本发布，请执行 `npx @dcloudio/uvm@latest` 更新，更新日志详见：https://download1.dcloud.net.cn/hbuil
html
[plugin:uni:mp-using-component] Cannot find module 'pinia/dist/pinia.mjs'
Require stack:
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/resolve.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/hbx/env.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/logs/format.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/logs/index.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/mp/style.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/mp/index.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/uni-cli-shared/dist/index.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/vite-plugin-uni/dist/cli/index.js
- /Users/connor/Desktop/template/room-test-uniapp/node_modules/@dcloudio/vite-plugin-uni/bin/uni.js
at roomkit/pages/home.vue:1:0
```

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。


视频通话
新建房间
房间号
创建房间号
房间类型
自由发言房间
您的名字
user_5
开启麦克风
开启视频
新建房间
资源管理器
MP-WEIXIN
Wxml
Console
Sources
Network
Performance
Memory
AppData
Storage
Security
Sensor
Mock
Audits
Vulnerability
[TRTCCloud-WX] 10:27:29.578 TUIRoomEngine.login error. Cannot read property 'GetParams' of undefined
TypeError: Cannot read property 'GetParams' of undefined
Cannot read property 'GetParams' of undefined
[worker] reportRealtimeAction:fail not support

Flutter

最近更新时间：2024-11-12 16:47:12

1. iOS上运行时报错：Error (Xcode): Library not found for -ld64?

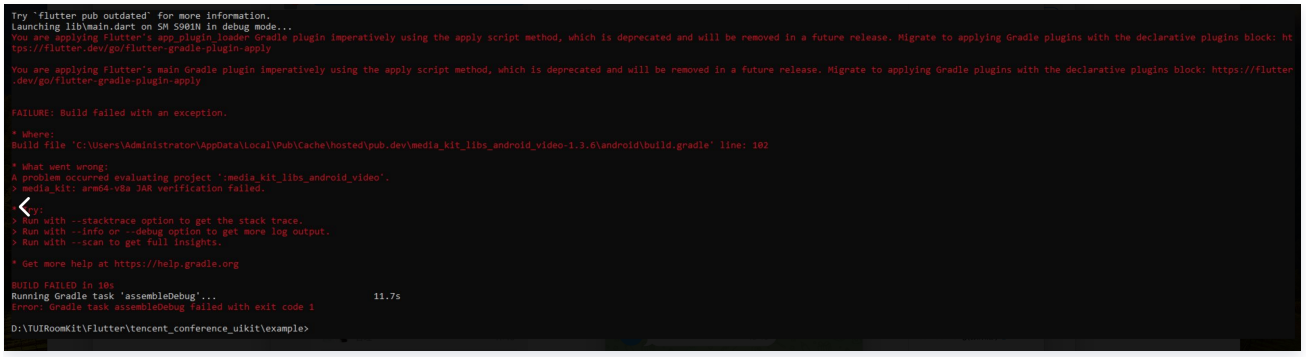
`Xcode` 15 换了编译的 `linker` ， `RoomEngine Flutter` 在1.6.0做了适配，老版本的 xcode 无法编译，使用最新的 `Xcode` 即可.

2. 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。

3. Android 端运行报错 media_kit: arm64-v8a JAR verification failed，如何处理?

详细报错见下图：



网络问题导致部分依赖未完全加载，执行以下命令重新运行即可，可能需要多次重试。

```
flutter clean
flutter pub cache clean
flutter pub get
```

4. Flutter 3.24 版本 Android compileSDK 34 运行报错 A failure occurred while executing com.android.build.gradle.tasks.VerifyLibraryResourcesTask$Action，如何处理?

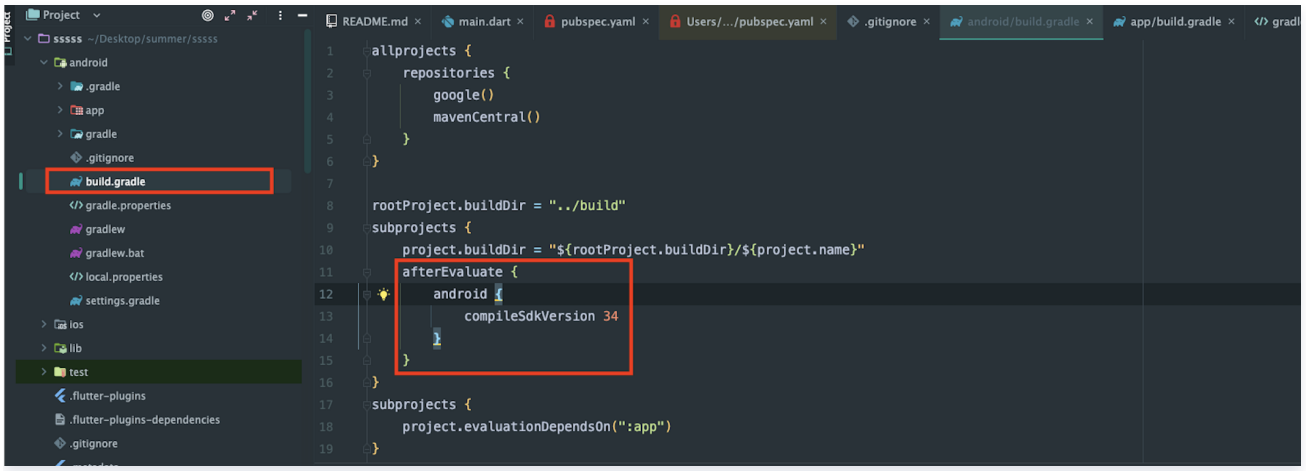
详细报错见下图：

Android 34版本特性，需要在您 App 的 `android/build.gradle` 中添加如下内容。

```
afterEvaluate {
        android {
```

```
            compileSdkVersion 34
        }
}
```

添加位置如图所示：



升级方案（TUIRoomKit）2.0版本升级指引

最近更新时间：2024-05-09 21:48:12

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：

```
implementation "com.tencent.liteav:LiteAVSDK_TRTC:11.8.0.14176"
implementation "io.trtc.uikit:rtc_room_engine:2.3.1.27"
implementation "com.tencent.imsdk:imsdk-plus:7.9.5680"
```

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', '~> 2.3.1'
```

- 使用RoomEngine：

```
pod 'RTCRoomEngine', '~> 2.3.1'
```

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（TUIRoomKit）

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
42002	预定会议无效

版权所有：腾讯云计算（北京）有限责任公司

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