实时音视频

搭建 Demo

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2019 腾讯云版权所有

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

【商标声明】

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

【服务声明】

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

文档目录

- 搭建 Demo
  - Web 互动直播
    - Web 直播互动组件
  - 视频通话
    - 视频通话(iOS)
    - 视频通话(Android)
    - 视频通话(Web)
    - 视频通话(Flutter)
    - TUICalling API（iOS）
    - TUICalling API（Android）
    - TRTCCalling API（Web）
    - TRTCCalling API（Flutter）
  - 语音通话
    - 语音通话(iOS)
    - 语音通话(Android)
    - 语音通话(Web)
    - 语音通话(Flutter)
    - TUICalling API（iOS）
    - TUICalling API（Android）
    - TRTCCalling API（Web）
    - TRTCCalling API（Flutter）
  - 视频互动直播
    - 视频互动直播(iOS)
    - 视频互动直播(Android)
    - 视频互动直播(Flutter)
    - TRTCLiveRoom API (iOS)
    - TRTCLiveRoom API (Android)
    - TRTCLiveRoom API (Flutter)
  - 语音聊天室
    - 语音聊天室(iOS)
    - 语音聊天室(Android)
    - TRTCVoiceRoom (iOS)
    - TRTCVoiceRoom (Android)
  - 多人音视频房间
    - 多人音视频房间(iOS)

版权所有：腾讯云计算（北京）有限责任公司

- 多人音视频房间(Android)
- 多人音视频房间(Flutter)
- 多人音视频房间(Windows&Mac)
- TUIRoom (iOS)
- TUIRoom(Android)
- TRTCMeeting（Flutter）
- TUIRoom(Windows&Mac)

语音沙龙

- 语音沙龙(Android)
- 语音沙龙(Flutter)
- TRTCChatSalon（iOS）
- TRTCChatSalon（Android）
- TRTCChatSalon（Flutter）

实时互动课堂

- 实时互动课堂(Electron)
- trtc-electron-education API

在线 K 歌_KTV

- 在线 K 歌_KTV（iOS）
- 在线 K 歌_KTV（Android）
- TRTCKaraoke(iOS)
- TRTCKaraoke(Android)

版权所有：腾讯云计算（北京）有限责任公司

搭建 Demo
Web 互动直播
Web 直播互动组件

最近更新时间：2022-04-02 17:02:04

本文介绍带 UI 的 Web 端直播互动组件 TUIPusher & TUIPlayer。TUIPusher & TUIPlayer 集成 实时音视频 TRTC 、 即时通信 IM 等基础 SDK，为企业直播、电商带货、行业培训、远程教学等多种直播场景提供快速上线 Web 端直播推拉流工具的解决方案。

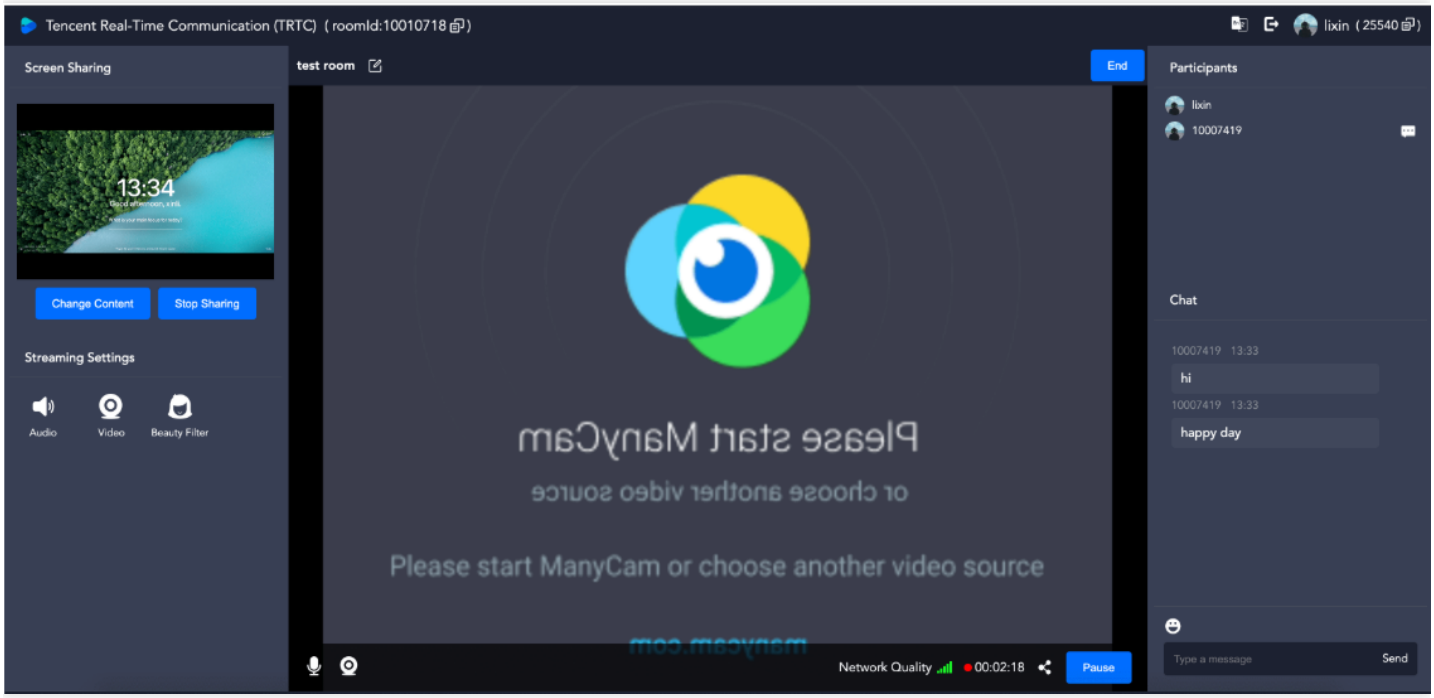
TUIPusher & TUIPlayer 的优势：

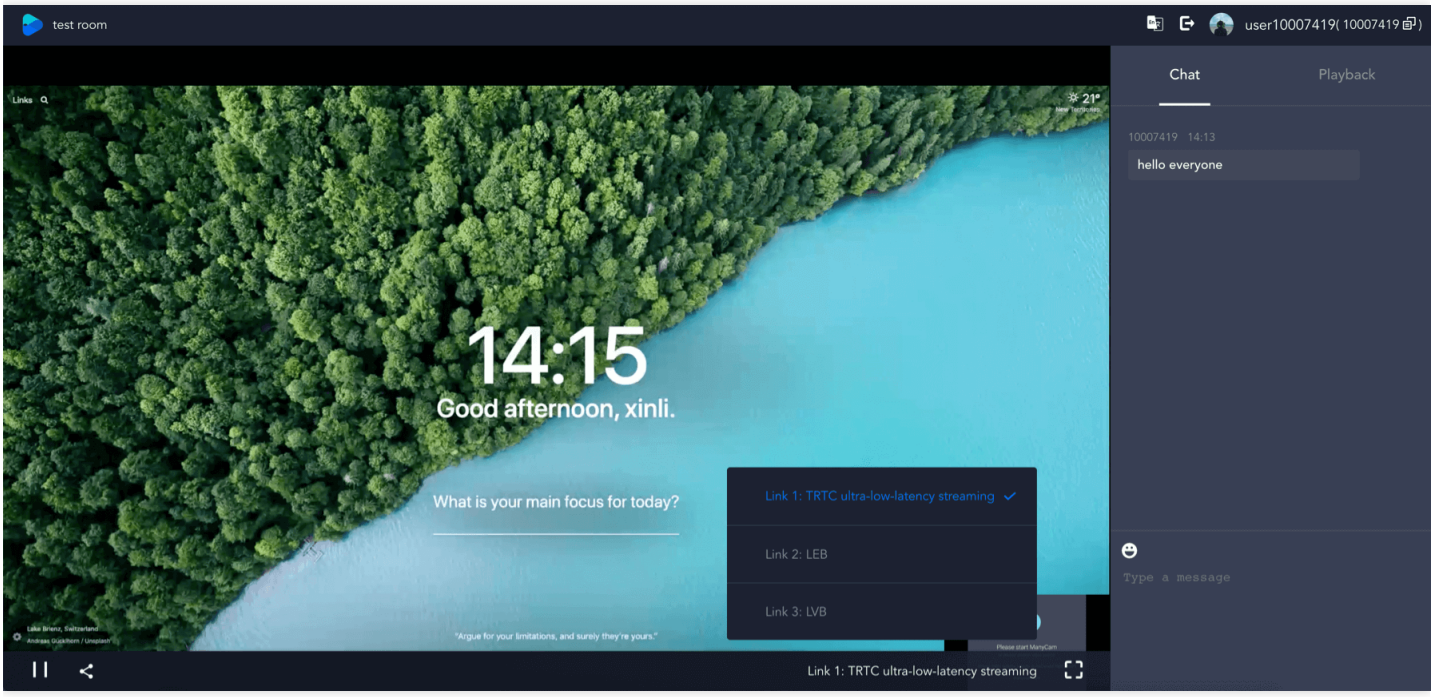
- 贴合直播场景需求，提供了含 UI 的直播场景通用解决方案，覆盖了直播场景常见功能（如设备选择、美颜、直播推流、观众拉流、聊天等），助力业务快速上线；
- 直接接入腾讯云实时音视频 TRTC、腾讯云即时通信 IM 以及腾讯云超级播放器 TCPlayer 等基础 SDK，方便客户灵活扩展业务功能；
- Web 端易于用户使用，易于功能迭代的天然优势；

快速体验

TUIPusher 及 TUIPlayer 功能演示请看下图。同时，为了方便您快速体验 TUIPusher & TUIPlayer 的功能，我们结合用户管理系统和房间管理系统提供了 TUIPusher 体验链接 及 TUIPlayer 体验链接。

> 注意：
> 同时体验 TUIPusher 和 TUIPlayer 需要使用两个不同的账号登录。





代码下载

Web 端直播互动组件（TUIPusher & TUIPlayer）下载方式如下：

- TUIPusher
- TUIPlayer

功能介绍

TUIPusher 推流组件

- 支持采集摄像头和麦克风的流并推流
  - 可根据需求设置视频参数（帧率，分辨率，码率）
  - 支持开启美颜并设置视频美颜参数
- 支持采集屏幕分享流并推流
- 支持推流到腾讯云实时音视频后台，推流到腾讯云 CDN
- 支持在线聊天室，和在线观众进行聊天互动
- 支持获取观众列表，对在线观众进行禁言操作

TUIPlayer 拉流组件

- 支持同时播放音视频流和屏幕分享流
- 支持在线聊天室，和主播及其他观众进行聊天互动
- 支持超低延时直播（300ms 延时），快直播（1000ms 以内延时）以及标准直播（支持超高并发观看）三种拉流线路
- 兼容桌面浏览器及移动端浏览器，支持移动端浏览器横屏观看

> 注意：
> 部分浏览器不支持 WebRTC，只能使用标准直播线路观看，如需体验其他线路，请尝试更换浏览器。

接入方式

注意事项

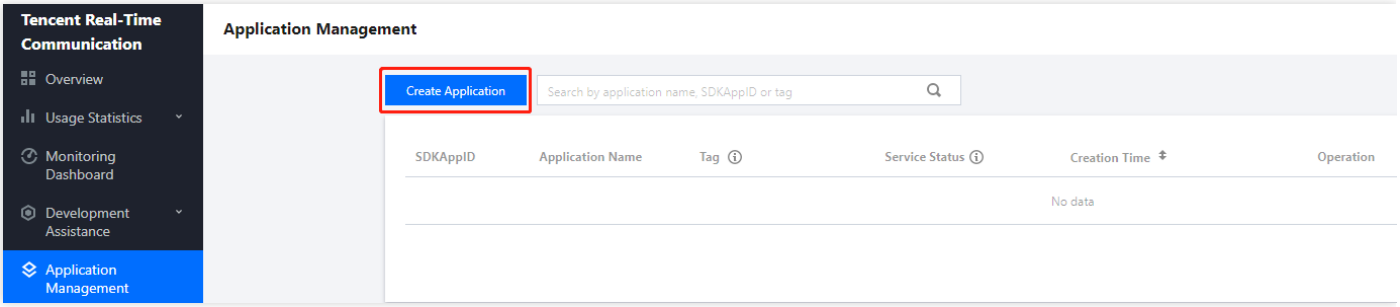
- TUIPusher & TUIPlayer 基于腾讯云实时音视频和即时通信服务进行开发。实时音视频 TRTC 应用与 即时通信 IM 应用的 SDKAppID 一致，才能复用账号与鉴权。
- 即时通信 IM 应用针对文本消息，提供基础版本的 安全打击能力，如果希望使用自定义不雅词功能，可以单击 **升级** 。
- 本地计算 UserSig 的方式仅用于本地开发调试，请勿直接发布到线上，一旦 SECRETKEY 泄露，攻击者就可以盗用您的腾讯云流量。正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤一：开通腾讯云服务

- 方式1：基于实时音视频
- 方式2：基于即时通信 IM

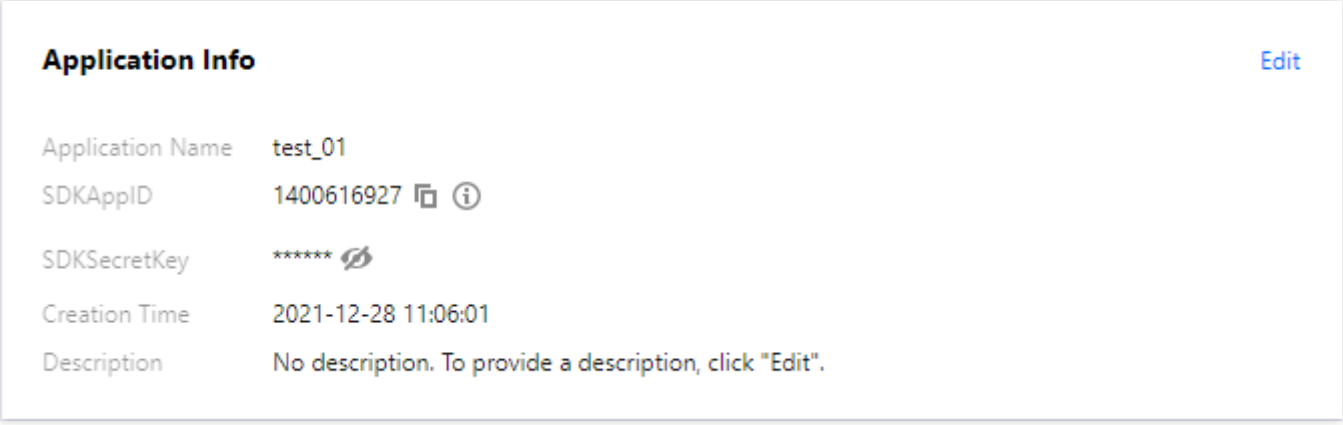
步骤1：创建实时音视频 TRTC 应用

1. 注册腾讯云账号 并开通 实时音视频 和 即时通信 服务。
2. 在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。

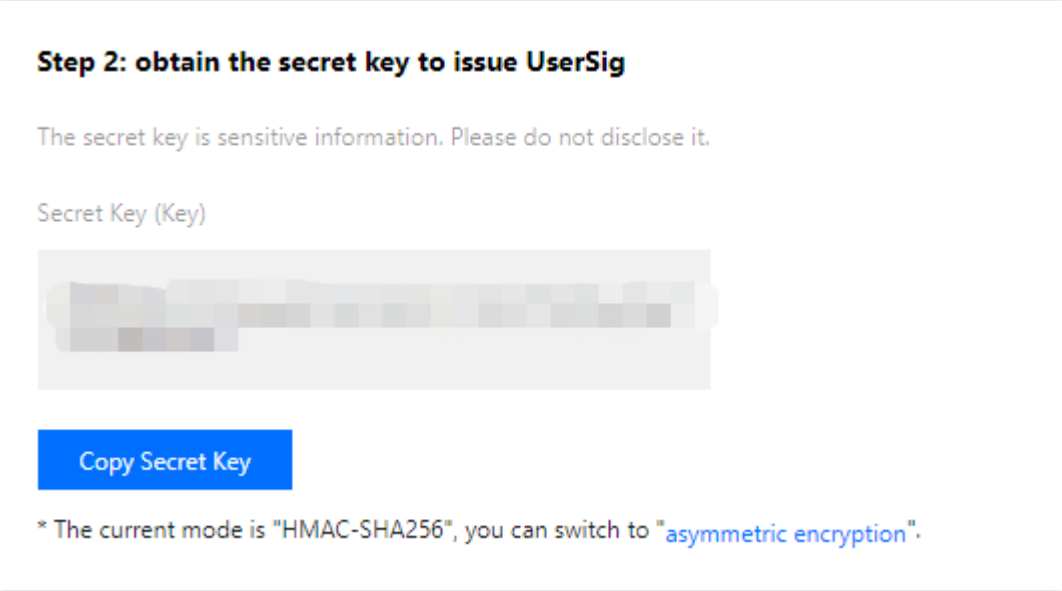


步骤2: 获取 TRTC 密钥信息

1. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。



2. 在 **应用管理 > 快速上手** 中获取应用的 secretKey 信息。



说明

- 首次创建实时音视频应用的腾讯云账号，可获赠一个10000分钟的音视频资源免费试用包。
- 创建实时音视频应用之后会自动创建一个 SDKAppID 相同的即时通信 IM 应用，可在 即时通信控制台 配置该应用的套餐信息。

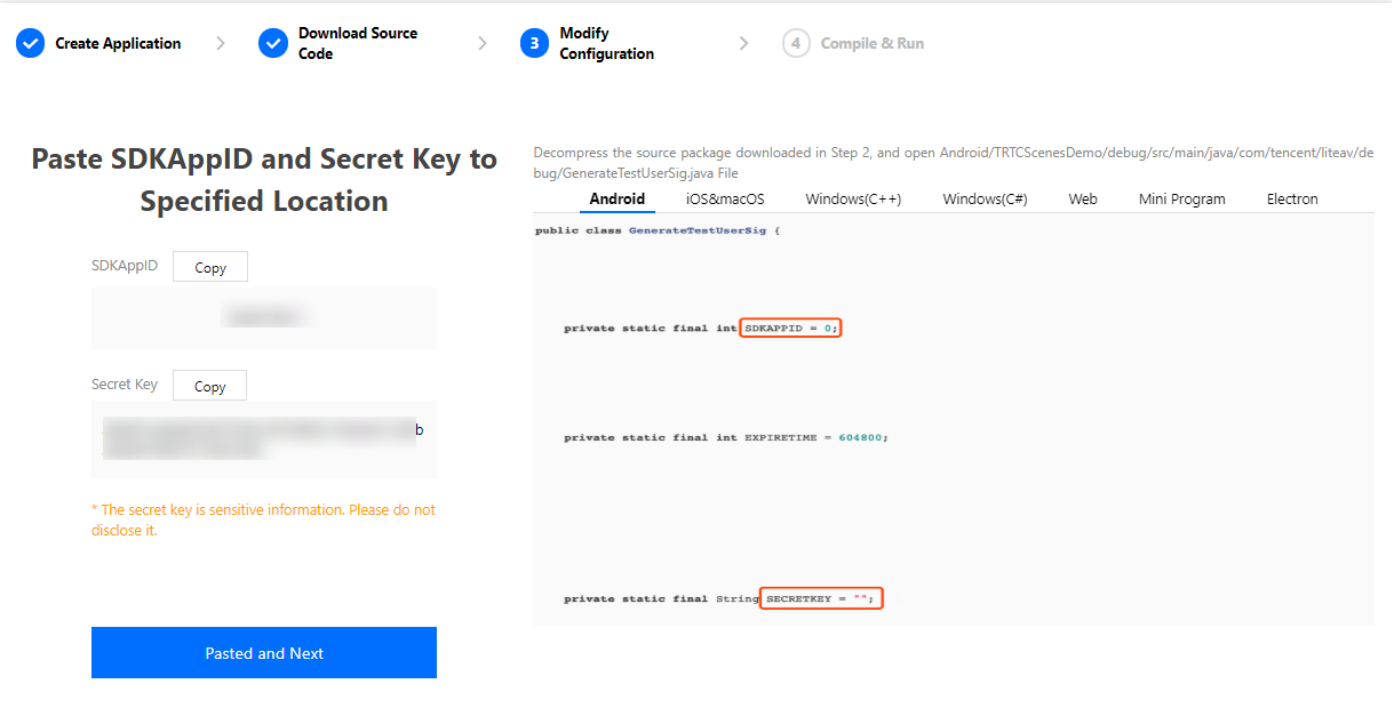
步骤二：项目准备

1. 在 GitHub 下载 TUIPusher & TUIPlayer 代码。

2. 为 TUIPusher & TUIPlayer 安装依赖。

```
cd Web/TUIPusher
npm install
cd Web/TUIPlayer
npm install
```

3. 将 sdkAppId 和 secretKey 填入 `TUIPusher/src/config/basic-info-config.js` 及 `TUIPlayer/src/config/basic-info-config.js` 配置文件中。



4. 本地开发环境运行 TUIPusher & TUIPlayer。

```
cd Web/TUIPusher
npm run serve
cd Web/TUIPlayer
npm run serve
```

5. 可打开 `http://localhost:8080` 和 `http://localhost:8081` 体验 TUIPusher 和 TUIPlayer 功能。

6. 可更改 `TUIPusher/src/config/basic-info-config.js` 及 `TUIPlayer/src/config/basic-info-config.js` 配置文件中的房间，主播及观众等信息，**注意保持 TUIPusher 和 TUIPlayer 的房间信息，主播信息一致**。

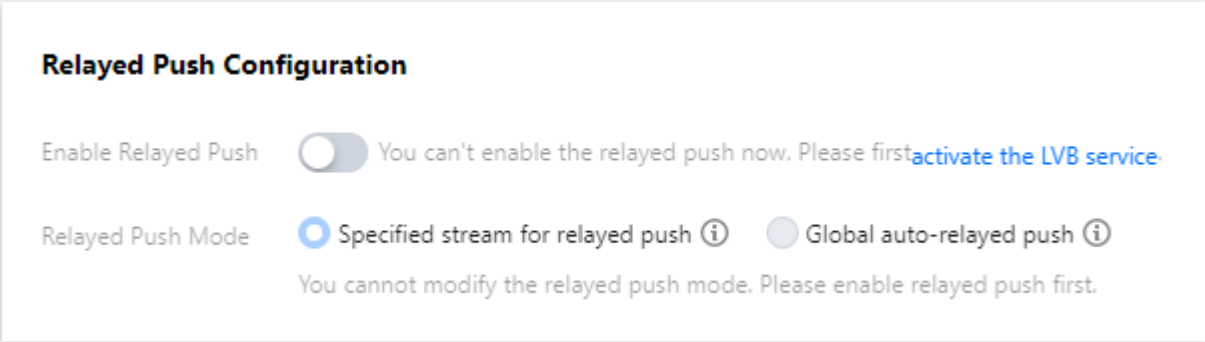
注意：

- 完成以上配置，您可以使用 TUIPusher & TUIPlayer 进行超低延时直播，如您需要支持快直播和标准直播，请继续阅读 步骤三：旁路直播。
- 本地计算 UserSig 的方式仅用于本地开发调试，请勿直接发布到线上，一旦您的 `SECRETKEY` 泄露，攻击者就可以盗用您的腾讯云流量。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤三：旁路直播

TUIPusher & TUIPlayer 实现的快直播和标准直播依托于腾讯云 云直播服务，因此支持快直播和标准直播线路需要您开启旁路推流功能。

1. 在 **实时音视频控制台** 中为您正在使用的应用开启旁路推流配置，可按需开启指定流旁路或全局自动旁路。



2. 请在 **域名管理** 页面添加自有播放域名，具体请参见 添加自有域名。
3. 在 `TUIPlayer/src/config/basic-info-config.js` 配置文件中配置播放域名。

完成以上配置，您可以体验 TUIPusher & TUIPlayer 支持超低延时直播，快直播以及标准直播的所有功能。

步骤四：生产环境应用

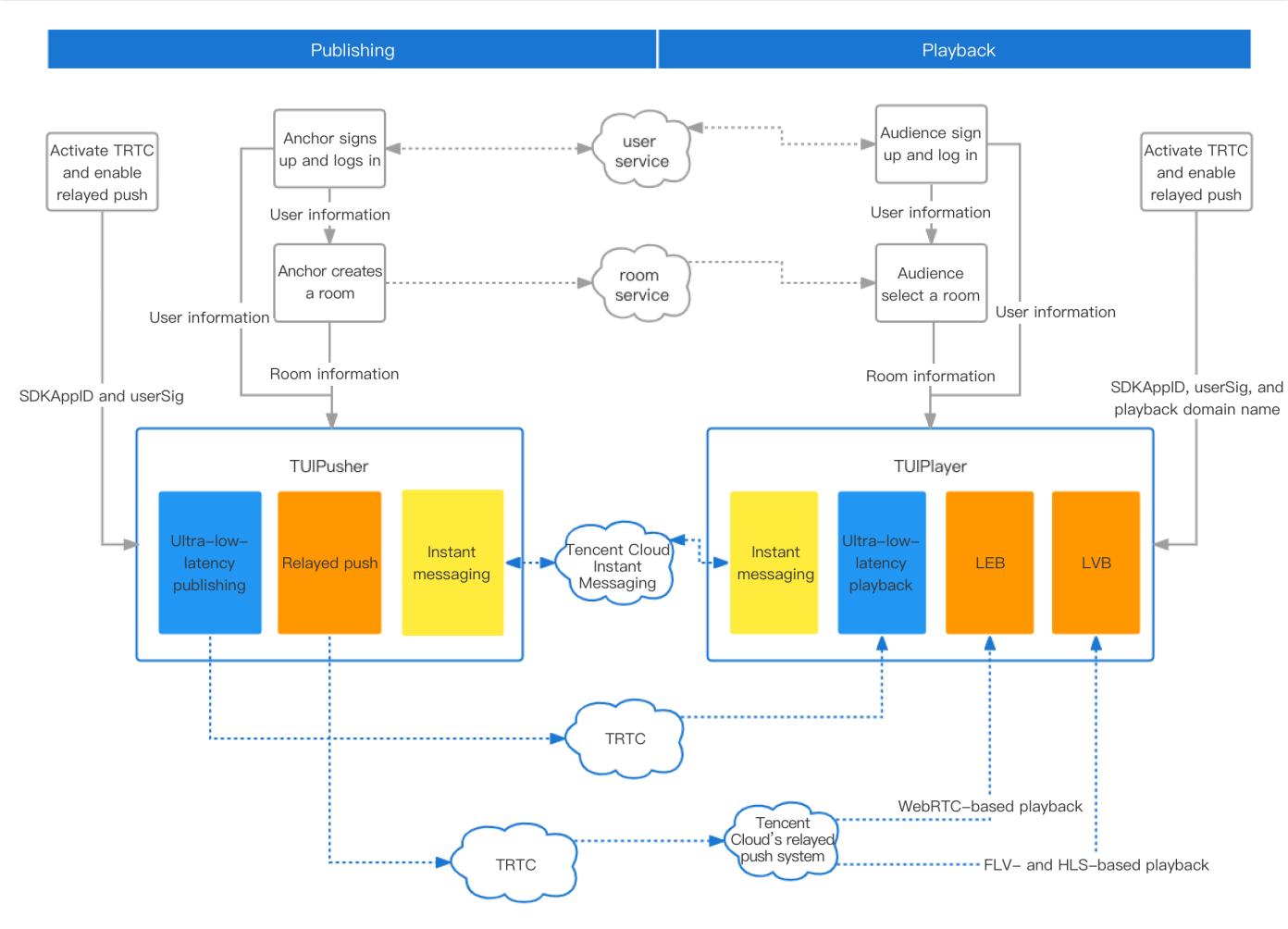
当您将 TUIPusher & TUIPlayer 用于生产应用时，在接入 TUIPusher & TUIPlayer 之外，您需要：

- 创建用户管理系统，用于管理产品用户信息，包括但不限于用户 ID，用户名，用户头像等。
- 创建房间管理系统，用于管理产品直播间信息，包括但不限于直播间 ID、直播间名称，直播间主播信息等。
- 服务端生成 UserSig。

> 注意：
>
> - 本文生成 UserSig 的方式，是在客户端根据您填入的 sdkAppId 及 secretKey 生成 userSig，该方式的 secretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TUIPusher & TUIPlayer 进行功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的应用向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

- 参考 `TUIPusher/src/pusher.vue` 及 `TUIPlayer/src/player.vue` 文件，将用户信息、直播间信息、SDKAppId 及 UserSig 等账号信息提交到 vuex 的 store 进行全局存储，您就可以跑通推拉流两个客户端的所

有功能。详细业务流程参见下图：



相关问题

Web 端如何实现美颜功能？

请参见 开启美颜。

Web 端如何实现屏幕共享？

请参见 屏幕分享。

Web 端如何实现云端录制?

1. 开启**云端录制**功能，具体操作请参见 实现云端录制与回放。
2. 开启**云端录制> 指定用户录制**之后，Web 端可通过在调用 TRTC.createClient 接口时传入 userDefineRecordId 参数开启录制。

Web 端如何实现推流到 CDN ？

Web 端推流到 CDN 请参见 实现推流到 CDN。

Web 端如何实现快直播拉流？

实现快直播拉流的方式是通过 Web SDK 推流到 CDN 之后使用 WebRTC 协议拉流。

注意事项

平台支持

操作系统	浏览器类型	浏览器最低版本要求	TUIPlayer	TUIPusher	TUIPusher 屏幕分享
Mac OS	桌面版 Safari 浏览器	11+	支持	支持	支持（需要 Safari13+ 版本）
Mac OS	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Mac OS	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Mac OS	桌面版 Edge 浏览器	80+	支持	支持	支持
Mac OS	桌面版微信内嵌网页	-	支持	不支持	不支持
Mac OS	桌面版企业微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Windows	桌面版 QQ 浏览器（极速内核）	10.4+	支持	支持	不支持
Windows	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）

版权所有：腾讯云计算（北京）有限责任公司

操作系统	浏览器类型	浏览器最低版本要求	TUIPlayer	TUIPusher	TUIPusher 屏幕分享
Windows	桌面版 Edge 浏览器	80+	支持	支持	支持
Windows	桌面版微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版企业微信内嵌网页	-	支持	不支持	不支持
iOS	微信内嵌浏览器	-	支持	不支持	不支持
iOS	企业微信内嵌浏览器	-	支持	不支持	不支持
iOS	移动版 Safari 浏览器	-	支持	不支持	不支持
iOS	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	微信内嵌浏览器	-	支持	不支持	不支持
Android	企业微信浏览器	-	支持	不支持	不支持
Android	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	移动版 QQ 浏览器	-	支持	不支持	不支持
Android	移动版 Firefox 浏览器	-	支持	不支持	不支持
Android	移动端 UC 浏览器	-	支持（仅支持标准直播观看）	不支持	不支持

域名要求

出于对用户安全、隐私等问题的考虑，浏览器限制网页在 HTTPS 协议下才能正常使用 TUIPusher & TUIPlayer 的全部功能。为确保生产环境用户顺畅接入和体验 TUIPusher & TUIPlayer 的全部功能，请使用 HTTPS 协议访问音视频应用页面。

注意：

本地开发可以通过 `http://localhost` 协议进行访问。

URL 域名及协议支持情况请参考如下表格：

应用场景	协议	TUIPlayer	TUIPusher	TUIPusher 屏幕分享	备注
生产环境	HTTPS 协议	支持	支持	支持	推荐
生产环境	HTTP 协议	支持	不支持	不支持	-
本地开发环境	`http://localhost`	支持	支持	支持	推荐
本地开发环境	`http://127.0.0.1`	支持	支持	支持	-
本地开发环境	`http://[本机IP]`	支持	不支持	不支持	-

防火墙限制

TUIPusher & TUIPlayer 依赖以下端口进行数据传输，请将其加入防火墙白名单。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名：qcloud.rtc.qq.com

结语

在后续的迭代中, TRTC Web 端推拉流组件会逐渐与 iOS、Andriod 等各端连通，并在 Web 端实现观众连麦、高级美颜、自定义布局、转推多平台、上传图片文字音乐等能力，欢迎大家多多使用、提出您的宝贵意见。

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

视频通话
视频通话(iOS)

最近更新时间：2022-03-10 11:09:33

效果展示

您可以 下载 安装我们的 App 体验实时音视频通话的效果。

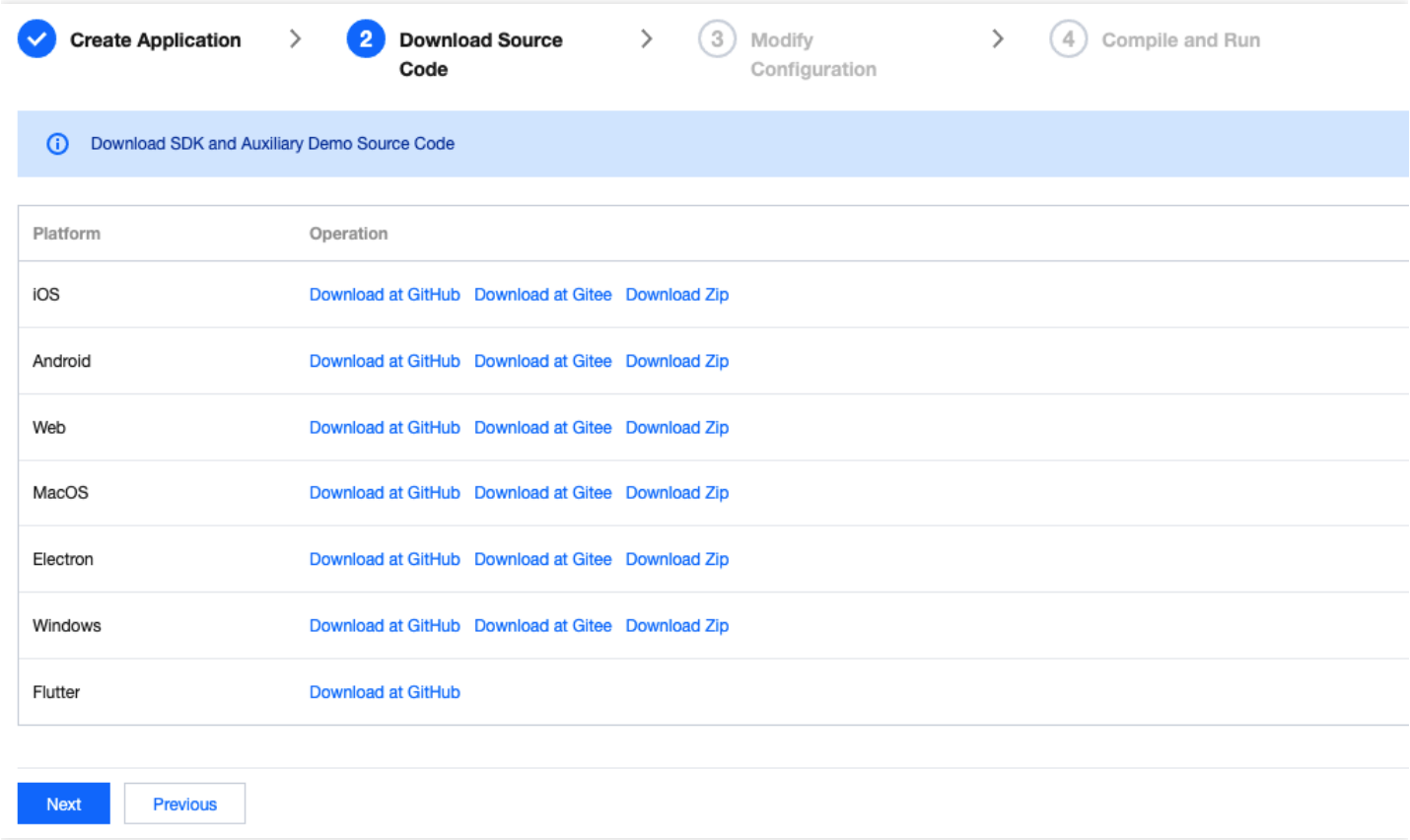
主动呼叫	被叫接听

注意：
为方便您快速实现音视频通话功能，我们对 TUICalling 组件进行了改造，通话 UI 在 TUICalling 组件内部实现，您可以无需关注 UI。

运行并体验 App

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助>快速跑通Demo**。
2. 输入应用名称，例如 `TestVideoCall` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

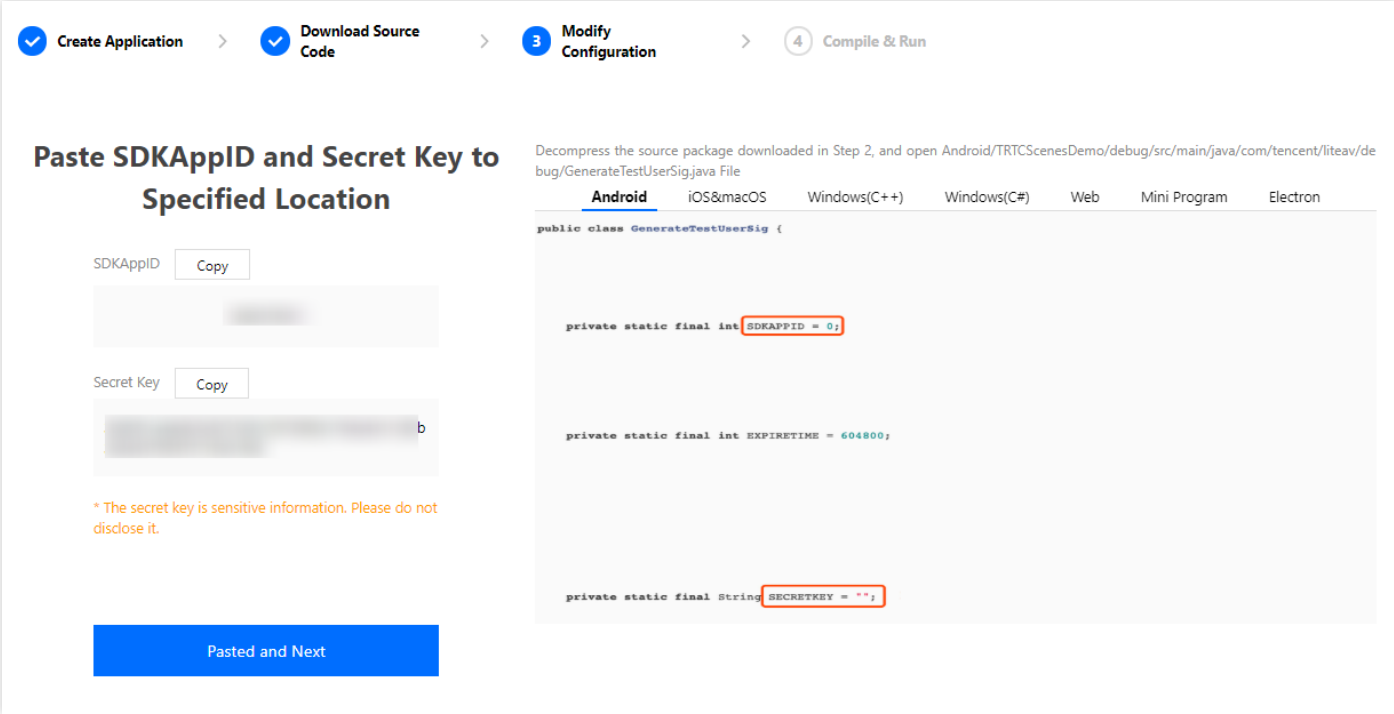
单击进入 TUICalling，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `iOS/Example/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

Xcode（11.0及以上的版本）打开源码工程 `TUICalling/Example/TUICallingApp.xcworkspace`，单击 **运行** 即可开始调试本 App。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

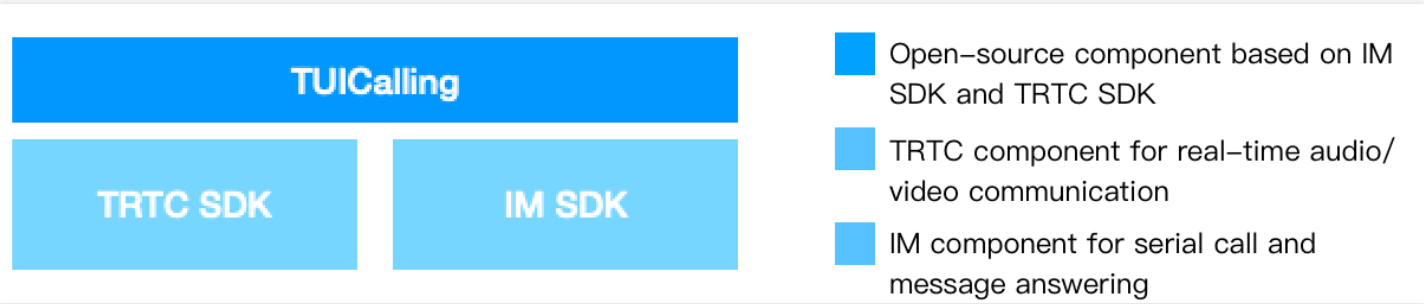
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入要拨打的 userId，单击搜索。
3. 单击 **呼叫**，选择拨打 **视频通话** （**请确保被叫方保持在应用内，否则可能会拨打失败**）。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 进入主页，等待接听来电。

具体接入流程

源码 文件夹 `Source` 中包含三个子文件夹 ui、model 和 Service，其中 Service 文件夹中包含了我们对外暴露的开源组件 TUICallingManager，您可以在 `TUICallingManager.h` 文件中看到该组件提供的接口函数。



您直接使用开源组件 TUICalling 的TUICallingManager即可轻松实现音视频通话功能，而无需再自己实现复杂的通话UI界面和逻辑。

步骤1：集成 SDK

通话组件 TRTCCalling 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 info.plist 文件中需要添加 `Privacy - Camera Usage Description`， `Privacy - Microphone Usage Description` 申请摄像头和麦克风权限。

步骤3：导入 TUICalling 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 将工程目录下的 `Source`、 `Resources`、 `TXAppBasic` 文件夹， `TUICalling.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
pod 'TXAppBasic', :path =&gt; "../TXAppBasic/"
pod 'TXLiteAVSDK_TRTC'
pod 'TUICalling', :path =&gt; "../", :subspecs =&gt; ["TRTC"]
```

步骤4：初始化并登录组件

1. 调用 `TUICallingManager.sharedInstance()` 进行组件初始化。
2. 调用 `TUILogin.initWithSdkAppID(SDKAPPID)` 进行登录初始化。
3. 调用 `TUILogin.login(userId, userSig)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
sdkAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。

userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。

```
// 组件初始化
TUICallingManager.sharedInstance();
// 登录
TUILogin.initWithSdkAppID(SDKAPPID)
TUILogin.login(userId, userSig) {
print("login success")
} fail: { code, errorDes in
print("login failed, code:¥(code), error: ¥(errorDes ?? "nil")")
}
```

步骤5：实现音视频通话

1. 发起方：调用 TUICallingManager 的 `call();` 方法发起通话的请求，并传入用户 ID数组（userIDs）和通话类型（type），通话类型参数传入 `.audio` （音频通话）或者 `.video` （视频通话）。如果用户 ID数组（userIDs）只有1个 userID 时视为单人通话，如果用户 ID 数组（userIDs）有多个 userID 时（>=2）视为多人通话。
2. 接收方：当接收方处于已登录状态时，会自动启动相应的界面。如果希望接收方在不处于登录状态时也能收到通话请求，请参见 离线接听。

```
// 1. 注册监听器
TUICallingManager.shareInstance().setCallingListener(listener: TUICallingListerner())

// 2. 设置是否自定义页面（默认关）
TUICallingManager.shareInstance().enableCustomViewRoute(enable: true)

// 3. 实现监听器回调方法
public func shouldShowOnCallView() -&gt; Bool {
return true;
}

public func callStart(userIDs: [String], type: TUICallingType, role: TUICallingRole, viewControll
er: UIViewController?) { if let vc = viewController {
callingVC = vc;
vc.modalPresentationStyle = .fullScreen

if var topController = UIApplication.shared.keyWindow?.rootViewController {
while let presentedViewController = topController.presentedViewController {
topController = presentedViewController
}

if let navigationVC = topController as? UINavigationController {
```

```
if navigationVC.viewControllers.contains(self) {
present(vc, animated: false, completion: nil)
} else {
navigationVC.popToRootViewController(animated: false)
navigationVC.pushViewController(self, animated: false)
navigationVC.present(vc, animated: false, completion: nil)
}
} else {
topController.present(vc, animated: false, completion: nil)
}
}
}
}

public func callEnd(userIDs: [String], type: TUICallingType, role: TUICallingRole, totalTime: Flo
at) {
callingVC.dismiss(animated: true, completion: nil)
}

public func onCallEvent(event: TUICallingEvent, type: TUICallingType, role: TUICallingRole, messa
ge: String) {

}
// 4.拨打电话
TUICallingManager.shareInstance().call(userIDs, .video)
```

步骤6：实现离线接听

> 说明：
> 如果您的业务定位是在线客服等不需要离线接听功能的场景，那么完成上述 步骤1 - 步骤5 的对接即可。但如果您的业务定位是社交场景，建议实现离线接听。

IM SDK 支持离线推送，您需要进行相应的设置才能达到可用标准。

1. 申请 Apple 推送证书，具体操作请参见 Apple 推送证书申请。
2. 在后台及客户端配置离线推送，具体操作请参见 离线推送（iOS）。
3. 目前在 TRTCCallingImpl 的 sendModel 信令发送函数中已经集成了离线发送的函数，当配置好 App 的离线推送后，消息就可实现离线推送。

组件 API 列表

TUICalling 组件的 API 接口列表如下：

接口函数	接口功能
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器
setCallingBell	设置铃声(建议在30s以内)
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图， 开启后，会在呼叫/被叫开始回调中，接收到CallingView的实例，由开发者自行决定展示方式。注意：必须全屏或者与屏幕等比例展示，否则会有展示异常

视频通话(Android)

最近更新时间：2022-03-09 17:41:34

效果展示

您可以 下载 安装我们的 App 体验实时音视频通话的效果。

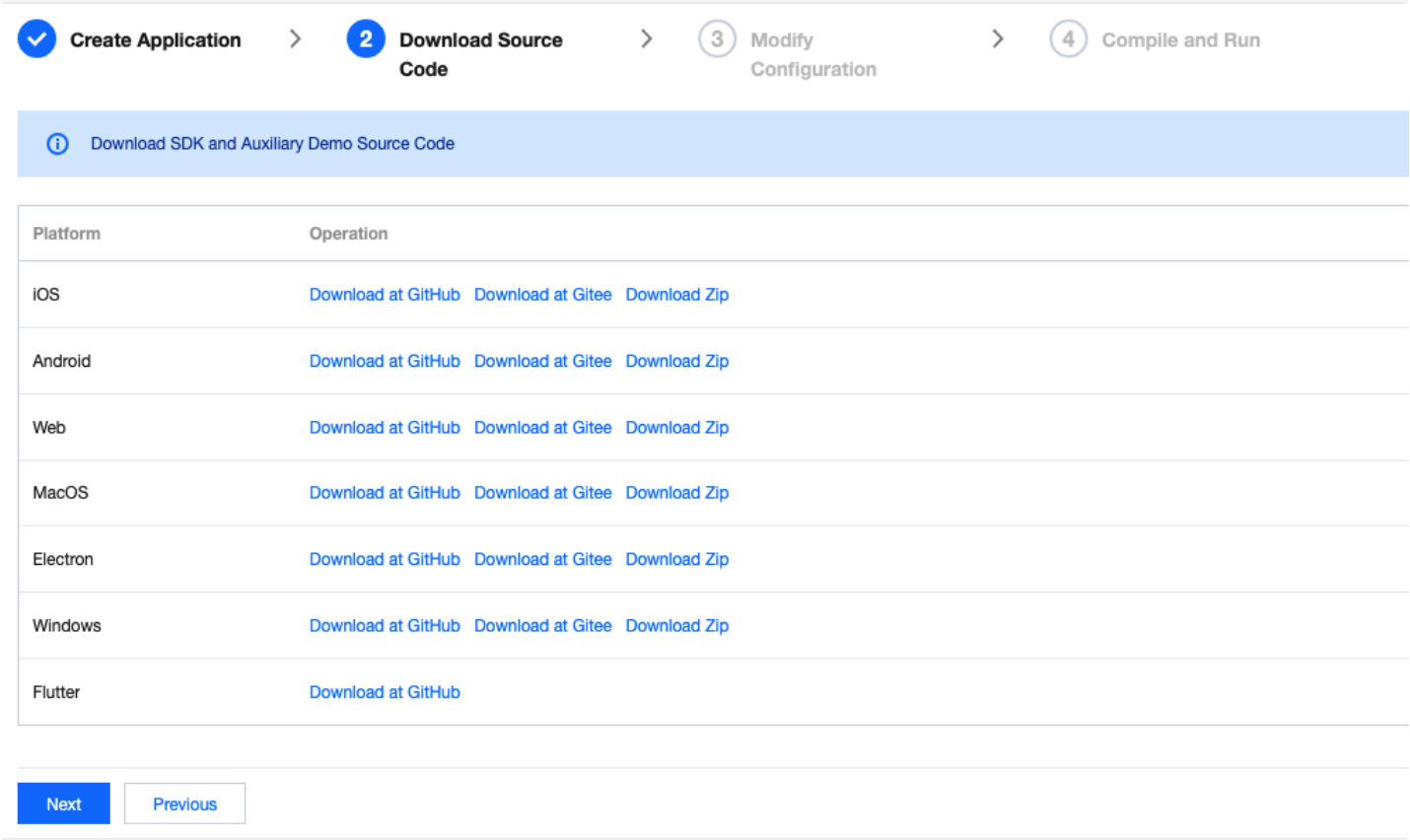
主动呼叫	被叫接听

注意：
为方便您快速实现音视频通话功能，我们对 TUICalling 组件进行了改造，通话UI在TUICalling 组件内部实现，您可以无需关注UI。

运行并体验 App

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助>快速跑通Demo**。
2. 输入应用名称，例如 `TestVideoCall` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

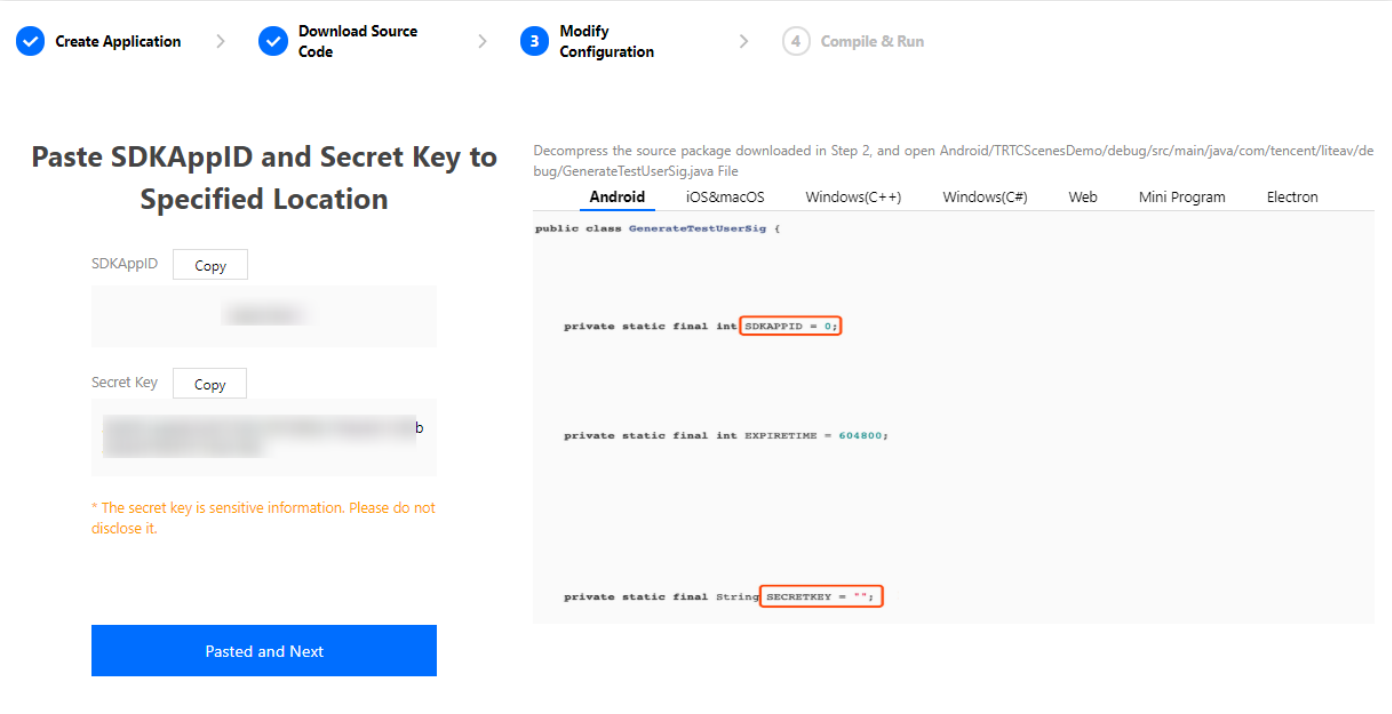
单击进入 TUICalling，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5 以上的版本）打开源码工程 `TUICalling`，单击 **运行** 即可开始调试本 App。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

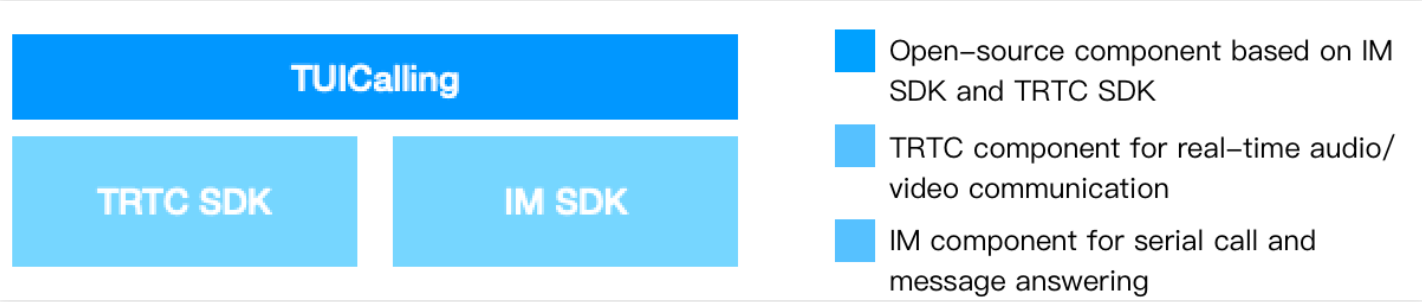
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入要拨打的 userId，单击搜索。
3. 单击 **呼叫**，选择拨打 **视频通话** （**请确保被叫方保持在应用内，否则可能会拨打失败**）。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 进入主页，等待接听来电。

具体接入流程

源码 文件夹 `Source` 中包含两个子文件夹 ui 和 model，其中 model 文件夹中包含了我们对外暴露的开源组件 TUICallingManager，您可以在 `TUICalling.java` 文件中看到该组件提供的接口函数。



您直接使用开源组件 TUICalling 的 TUICallingManager 即可轻松实现音视频通话功能，而无需再自己实现复杂的通话 UI 界面和逻辑。

步骤1：集成 SDK

音视频通话组件 TUICalling 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 mavenCentral 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
compile "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
compile 'com.tencent.imsdk:imsdk:latest.release'

// 由于我们使用到了 gson 解析，所以还需要依赖 google 的 Gson
compile 'com.google.code.gson:gson:latest.release'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击 **Sync Now** 同步 SDK。

> 说明：
> 若您的网络连接 mavenCentral 没有问题，SDK 会自动下载集成到工程里。

方法二：通过本地 AAR 依赖

如果您的开发环境访问 mavenCentral 仓库较慢，可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：

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

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

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

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

步骤3：导入 TUICalling 组件

拷贝 Source 目录到您的工程中，并在 `setting.gradle` 中完成导入，参考如下：

```
include ':Source'
```

步骤4：初始化并登录组件

1. 调用 `TUICallingManager.sharedInstance()` 进行组件初始化。
2. 调用 `TUILogin.init(context, SDKAppID, config, listener)` 进行登录初始化。
3. 调用 `TUILogin.login(userId, userSig, callback)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
SDKAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。
config	SDK配置。用于设置日志级别和日志回调（也可传null），详情可参考下方示例代码。
listener	IM监听器。用于接收一些必要的系统回调通知，比如：被踢下线、userSig过期等，详情可参考下方示例代码。
callback	登录结果监听器。通知登录是否成功，详情可参考下方示例代码。

```
// 组件初始化
TUICallingManager manager = TUICallingManager.sharedInstance();
```

```
// 登录
V2TIMSDKConfig config = new V2TIMSDKConfig();
config.setLogLevel(V2TIMSDKConfig.V2TIM_LOG_DEBUG);
config.setLogListener(new V2TIMLogListener() {
@Override
public void onLog(int logLevel, String logContent) {

}
});
TUILogin.init(this, ${您的SDKAPPID}, config, new V2TIMSDKListener() {
@Override
public void onKickedOffline() { // 登录被踢下线通知
mIsKickedOffline = true;
checkUserStatus();
}
@Override
public void onUserSigExpired() { // suerSig过期通知
mIsUserSigExpired = true;
checkUserStatus();
}
});
TUILogin.login("${您的userId}", "${您的userSig}", new V2TIMCallback() {
@Override
public void onError(int code, String msg) {
Log.d(TAG, "code: " + code + " msg:" + msg);
}
@Override
public void onSuccess() {
Log.d(TAG, "onSuccess");
}
});
```

步骤5：实现音视频通话

1. 发起方：调用 TUICallingManager 的 `call();` 方法发起通话的请求，并传入用户 ID数组（userids）和通话类型（type），通话类型参数传入 `TUICalling.Type.AUDIO` （音频通话）或者 `TUICalling.Type.VIDEO` （视频通话）。如果用户 ID数组（userids）只有1个userId时视为单人通话，如果用户 ID数组（userids）有多个userId时（>=2）视为多人通话。
2. 接收方：当接收方处于已登录状态时，会自动启动相应的界面。如果希望接收方在不处于登录状态时也能收到通话请求，请参见 离线接听。

```
// 1. 初始化组件
TUICallingManager manager = TUICallingManager.sharedInstance();
// 2. 注册监听器
```

```
manager.setCallingListener(new TUICalling.TUICallingListener() {
@Override
public boolean shouldShowOnCallView() {
return true;
}

@Override
public void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, final View
tuiCallingView) {
if (!shouldShowOnCallView() || null == tuiCallingView) {
return;
}
runOnUiThread(new Runnable() {
@Override
public void run() {
FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(FrameLayout.LayoutParams.MATCH_PAR
ENT, FrameLayout.LayoutParams.MATCH_PARENT);
mCallingView = tuiCallingView;
addContentView(tuiCallingView, params);
}
});
}

@Override
public void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTim
e) {
removeView();
}

@Override
public void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, Strin
g message) {
if (TUICalling.Event.CALL_FAILED == event) {
removeView();
}
}
});
// 3.拨打电话
manager.call(userIDs, TUICalling.Type.VIDEO);
```

步骤6：实现离线接听

> 说明：
> 如果您的业务定位是在线客服等不需要离线接听功能的场景，那么完成上述 步骤1 - 步骤5 的对接即可。但如果您的业务定位是社交场景，建议实现离线接听。

IM SDK 支持离线推送，但是 Android 端各个手机厂商均有各自的离线推送服务，因此接入复杂度要高于 iOS 平台，您需要进行相应的设置才能达到可用标准。

1. 申请对应厂商的推送渠道需要的证书等，并将其配置到即时通信 IM 控制台中，按照推送要求增加证书和 ID 等，详细的操作步骤请参见 即时通信 IM > 离线推送（Android） 。
2. 目前在 TRTCCallingImpl 的 sendModel 信令发送函数中已经集成了离线发送的函数，当配置好 App 的离线推送后，消息就可实现离线推送。

组件 API 列表

TUICalling 组件的 API 接口列表如下：

接口函数	接口功能
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器
setCallingBell	设置铃声(建议在30s以内)
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图， 开启后，会在呼叫/被叫开始回调中，接收到 CallingView 的实例，由开发者自行决定展示方式。注意：必须全屏或者与屏幕等比例展示，否则会有展示异常

视频通话(Web)

最近更新时间：2022-04-02 16:27:03

本文介绍如何实现一套可以在浏览器上运行的视频通话解决方案，文章分成两个部分：

- 第一部分：介绍如何开通服务并跑通我们提供的演示 Demo。
- 第二部分：介绍如何使用 TRTCCalling 组件快速搭建自己的视频通话功能。

环境要求

请使用最新版本的 Chrome 浏览器。目前桌面端 Chrome 浏览器支持 TRTC Web SDK 的相关特性比较完整，因此建议使用 Chrome 浏览器进行体验。

TRTCCalling 依赖以下端口进行数据传输，请将其加入防火墙白名单，配置完成后，您可以通过访问并体验 官网 Demo 检查配置是否生效。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名：qcloud.rtc.qq.com
  具体请参见 应对防火墙限制相关。

平台支持

目前该方案支持如下平台：

操作系统	浏览器类型	浏览器最低版本要求
Mac OS	桌面版 Safari 浏览器	11+
Mac OS	桌面版 Chrome 浏览器	56+
Mac OS	桌面版 Firefox 浏览器	56+
Mac OS	桌面版 Edge 浏览器	80+
Windows	桌面版 Chrome 浏览器	56+
Windows	桌面版 QQ 浏览器（极速内核）	10.4+
Windows	桌面版 Firefox 浏览器	56+

操作系统	浏览器类型	浏览器最低版本要求
Windows	桌面版 Edge 浏览器	80+

说明：
详细兼容性查询，具体请参见 浏览器支持情况。同时，您可通过 TRTC 检测页面 在线检测。

URL 域名协议限制

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

跑通测试 Demo

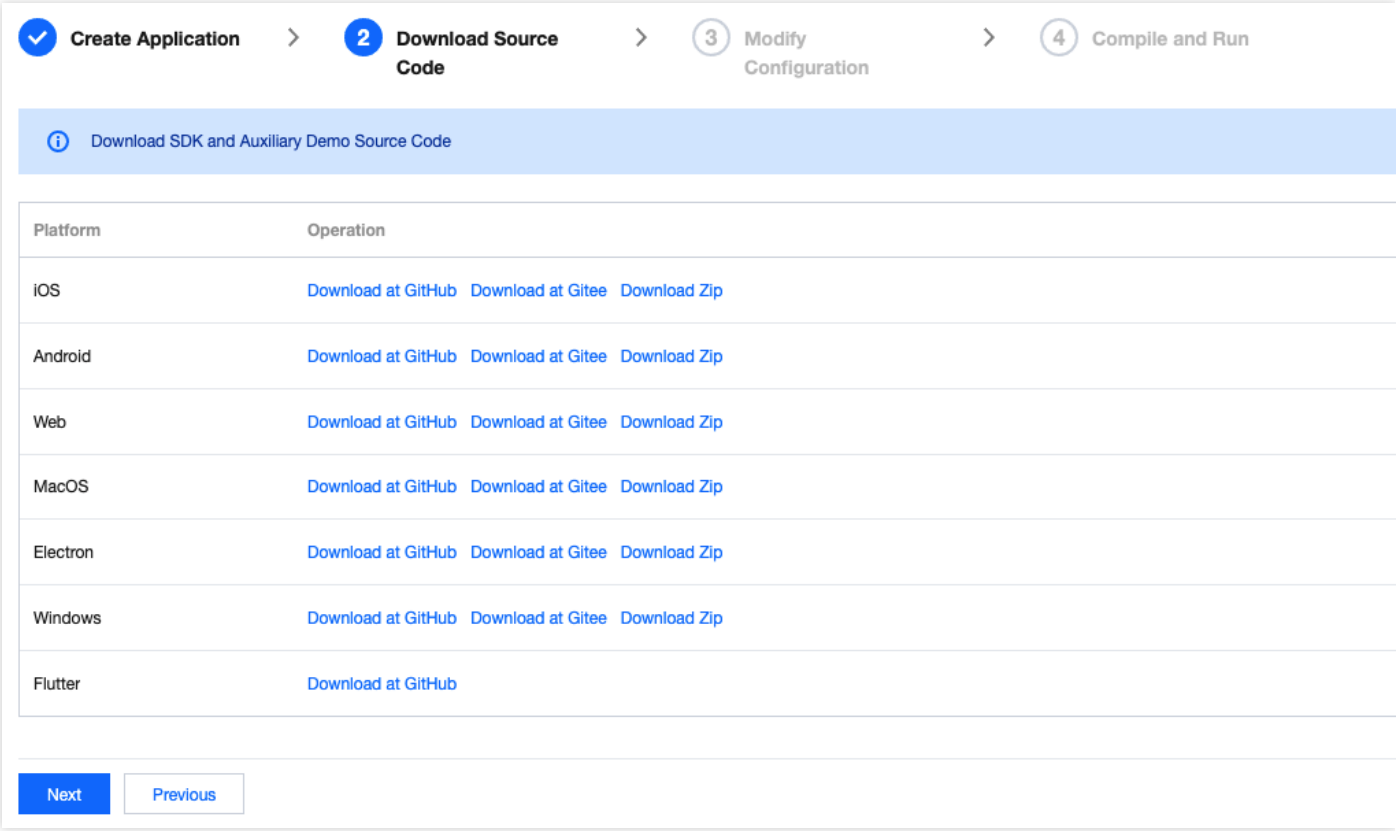
步骤1：创建新的应用

1. 注册腾讯云 账号，并完成实名认证。
2. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
3. 输入应用名称，例如 TestTRTC ，单击 **创建**。

步骤2：下载 Demo

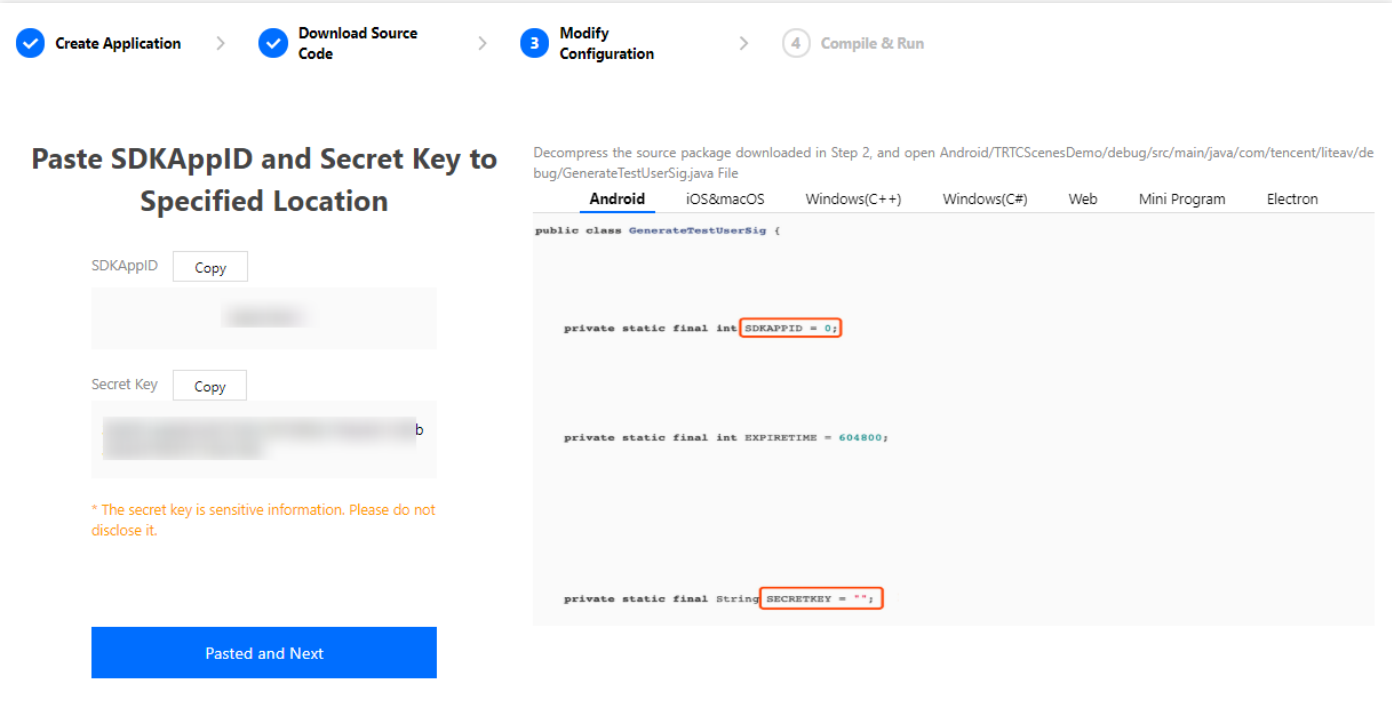
1. 根据实际业务需求下载 Demo 源码或单击进入 TUICalling下载 Demo 源码。

2. 下载完成后，单击 **已下载，下一步**。



步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `Web/public/debug/GenerateTestUserSig.js` 文件。
3. 设置 `GenerateTestUserSig.js` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。
5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

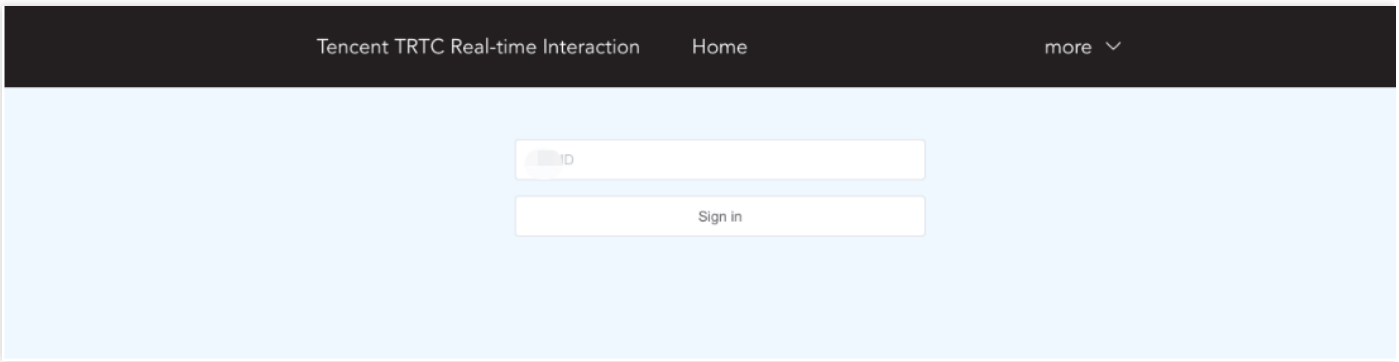
- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 Demo

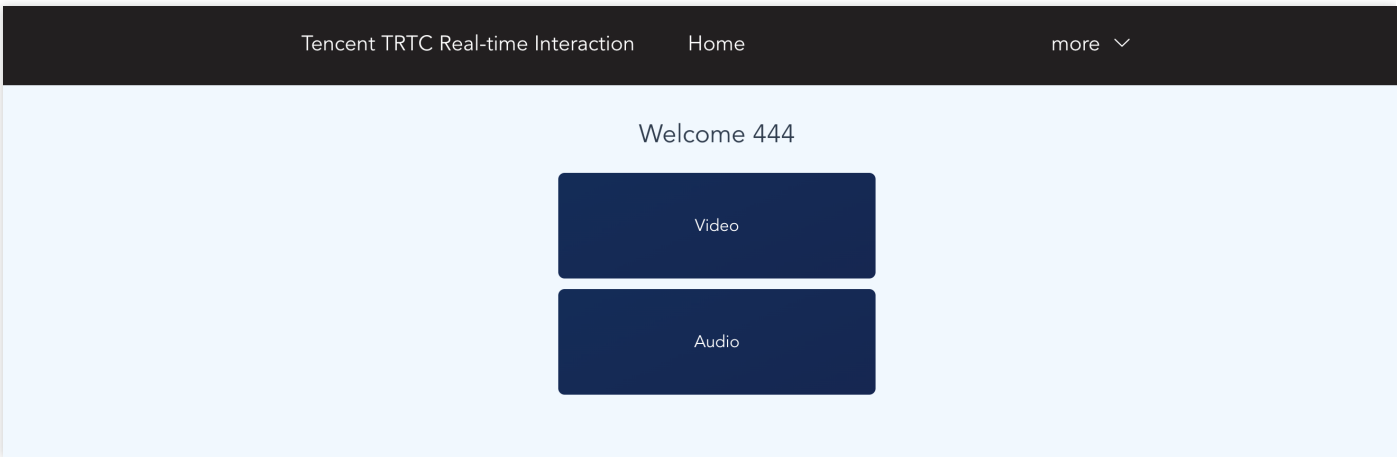
1. 在 Demo 目录下依次输入如下命令：

```
npm install
npm run serve
```

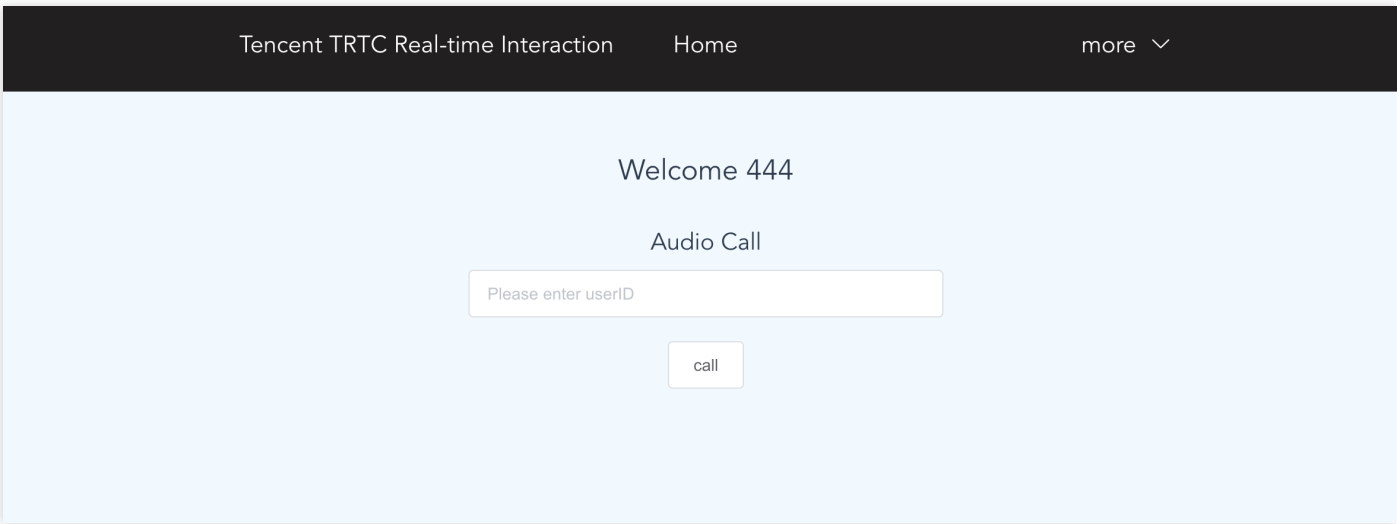
2. 启动 Chrome 浏览器中打开链接 `http://localhost:8080/` ，如果一切正常，Demo 运行界面如图所示：



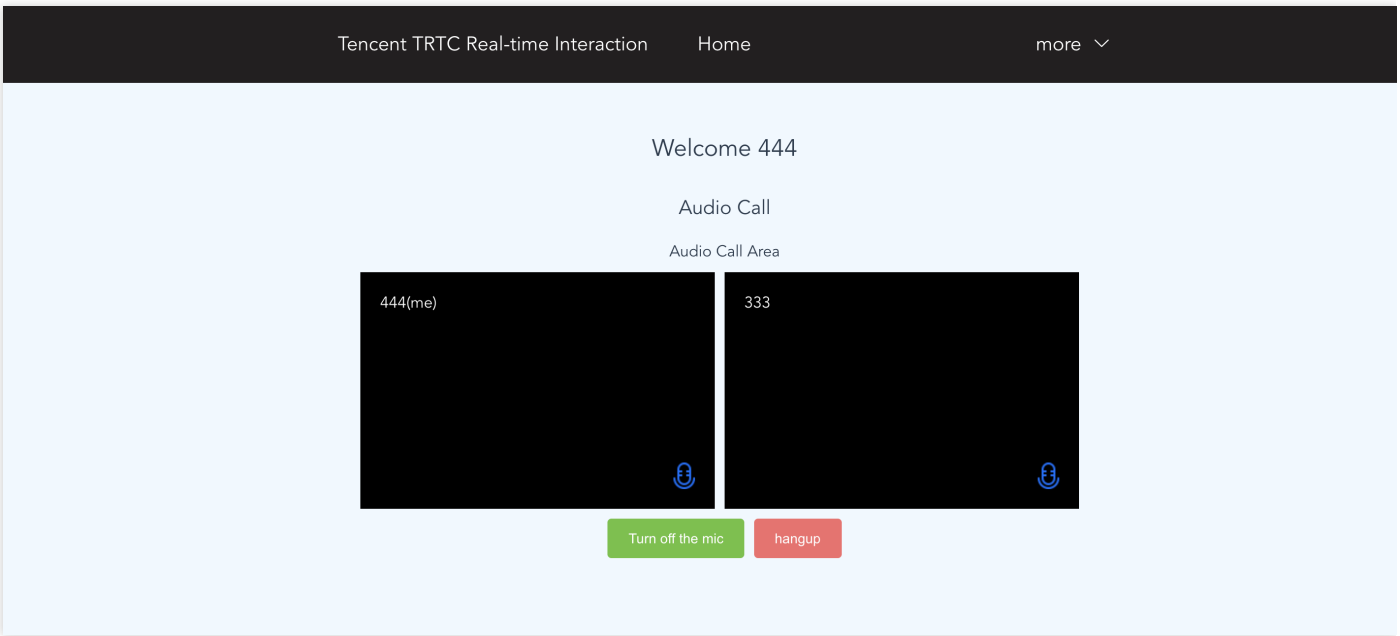
3. 输入用户 userid，单击 **登录**，并选择 **视频通话**：



4. 输入呼叫用户 userid，单击 **呼叫**：



5. 即可进行视频通话：



搭建自己的视频通话

步骤1：集成 TRTCCalling 组件

说明：

- 从v0.6.0起，需要手动安装依赖 trtc-js-sdk 和 tim-js-sdk 以及 tsignaling。
- 为了减小 trtc-calling-js.js 的体积，避免和接入侧已使用的 trtc-js-sdk 和 tim-js-sdk 以及 tsignaling 发生版本冲突，trtc-calling-js.js 将 trtc-js-sdk，tim-js-sdk，tsignaling 打包为外部依赖，在使用前您需要手动安装。

```
// npm 方式安装
npm install trtc-js-sdk --save

npm install tim-js-sdk --save

npm install tsignaling --save

npm install trtc-calling-js --save
```

```
// 如果您需要通过 script 方式使用 trtc-calling-js，需要按顺序引入以下资源

<script src="./trtc.js"></script>
<script src="./tim-js.js"></script>
<script src="./tsignaling.js"></script>
<script src="./trtc-calling-js.js"></script>
```

步骤2：创建 TRTCCalling 对象

创建 TRTCCalling 对象，并将 SDKAppID 参数设置为您自己的 SDKAppID。

```
import TRTCCalling from 'trtc-calling-js';

let options = {
SDKAppID: 0, // 接入时需要将0替换为您的 SDKAppID
// 从v0.10.2起，新增 tim 参数
// tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性
tim: tim
};
const trtcCalling = new TRTCCalling(options);
```

步骤3：完成登录

调用 login 函数完成登录操作，参数中的 userID 为用户名，userSig 为用户签名，userSig 的计算方式请参见 如何计算及使用 UserSig。

```
trtcCalling.login({
userID,
userSig,
});
```

步骤4：实现 1v1 通话

- **主叫方：呼叫某个用户**

```
trtcCalling.call({
userID, //用户 ID
type: 2, //通话类型，0-未知， 1-语音通话，2-视频通话
timeout //邀请超时时间，单位 s(秒)
});
```

- **被叫方：接听新的呼叫**

```
// 接听
trtcCalling.accept();
//拒绝
trtcCalling.reject()
```

- **打开本地摄像头**

```
trtcCalling.openCamera()
```

- **展示远端的视频画面**

```
trtcCalling.startRemoteView({
userID, //远端用户 ID
videoViewDomID //该用户数据将渲染到该 DOM ID 节点里
})
```

- **展示本地的预览画面**

```
trtcCalling.startLocalView({
userID, //本地用户 ID
videoViewDomID //该用户数据将渲染到该 DOM ID 节点里
})
```

- **挂断**

```
trtcCalling.hangup()
```

常见问题

为什么拨打不通，或者被踢下线？

组件暂不支持多实例登入，不支持**离线推送信令**功能，请您确认登入账号的唯一性。

> 说明：
>
> - **多实例**：一个 UserID 重复登入，或在不同端登入，将会引起信令的混乱。

- **离线推送**：实例在线才能接收消息，实例离线时接收到的信令不会在上线后重新推送。
  更多常见问题，请参见 TRTCCalling Web 相关问题。

技术咨询

了解更多详情您可以联系我们或发送邮件至colleenyu@tencent.com。

参考文档

- TRTCCalling web 官网体验
- TRTCCalling npm
- TRTCCalling web demo 源码
- TRTCCalling web API
- TRTCCalling web 相关问题

视频通话(Flutter)

最近更新时间：2022-01-10 16:34:59

如需快速实现视频通话功能，您可以直接基于我们提供的 Demo 进行修改适配，也可以使用我们提供的 TRTCCalling 组件并实现自定义 UI 界面。

复用 Demo 的 UI 界面

步骤1：创建新的应用

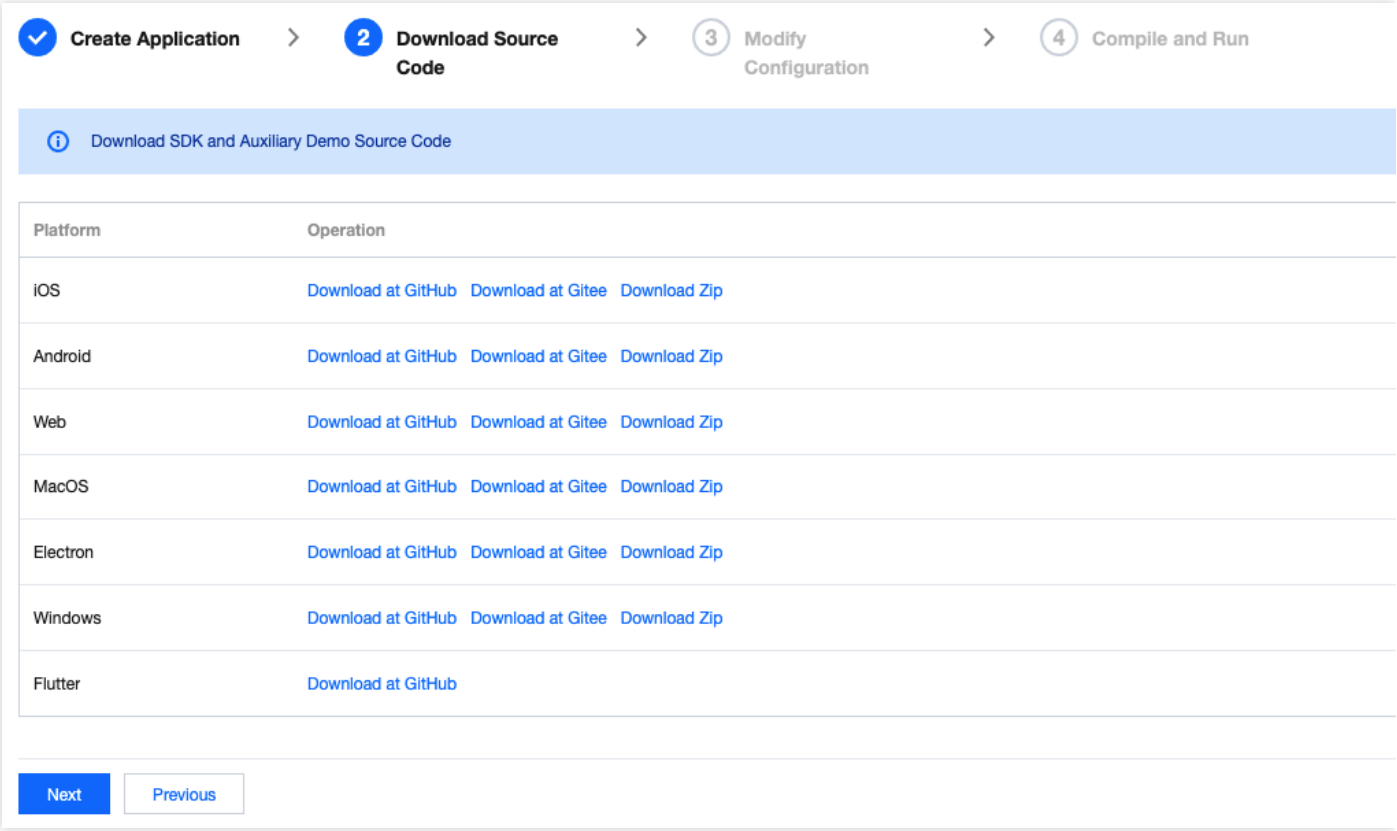
1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestVideoCall` ，单击【创建】。

> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击【已下载，下一步】。

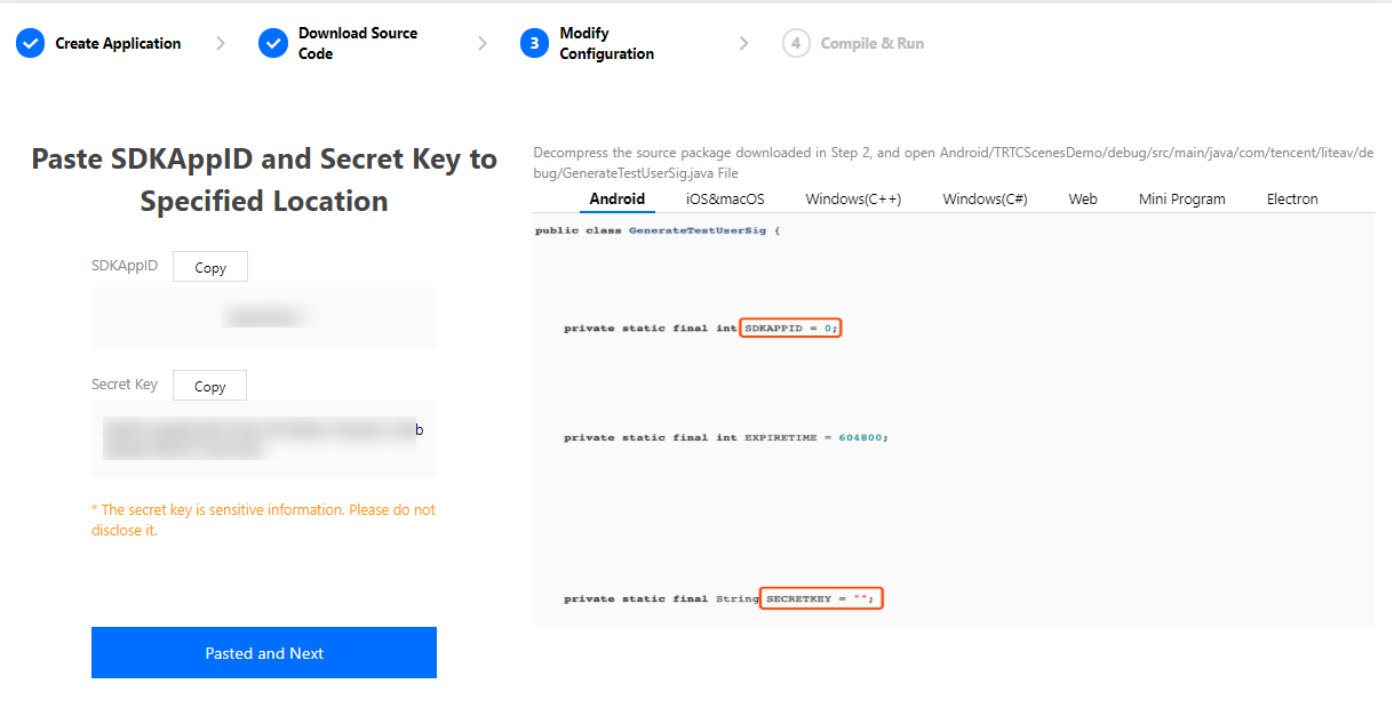


步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `/example/lib/debug/GenerateTestUserSig.dart` 文件。

3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：
   - SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为 PLACEHOLDER，请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 Demo

> 注意：
> Android 端需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。

2. 编译运行调试：
   - Android 端
   - iOS 端
   i. 执行 `flutter run` 。
   ii. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击【运行】即可。

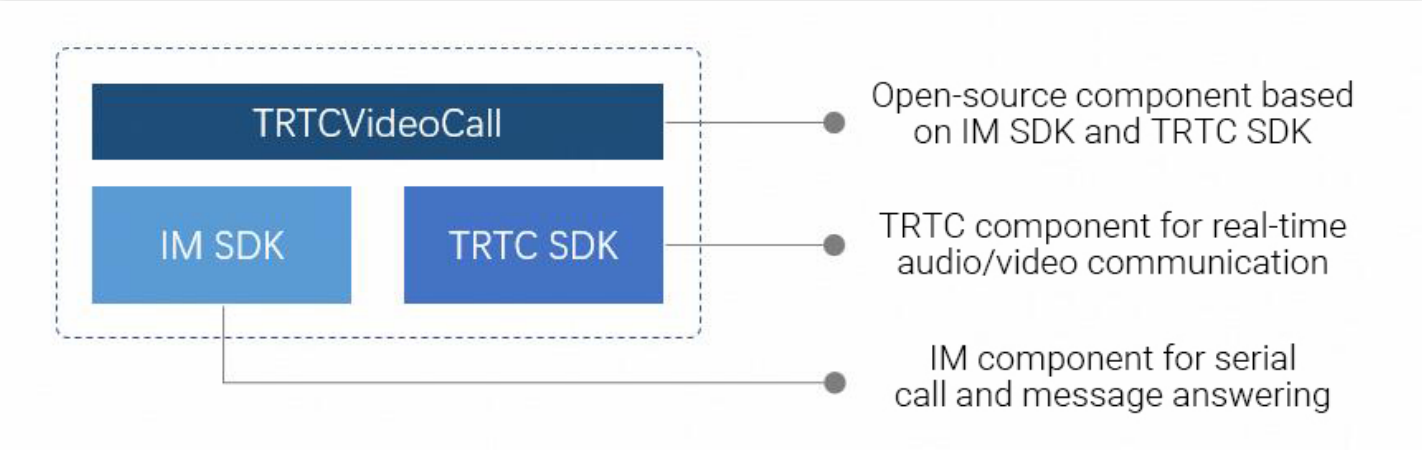
步骤5：修改 Demo 源代码

源码 文件夹 `TRTCCallingDemo` 中包含两个子文件夹 ui 和 model，其中 ui 文件夹中均为界面代码：

文件或文件夹	功能描述
TRTCCallingVideo.dart	展示音视频通话的主界面，通话的接听和拒绝就是在这个界面中完成的
TRTCCallingContact.dart	用于展示选择联系人的界面，可以通过此界面搜索已注册用户，发起通话

实现自定义 UI 界面

源码 文件夹 `TRTCCallingDemo` 中包含两个子文件夹 ui 和 model，其中 model 文件夹中包含了我们实现的可重用开源组件 TRTCCalling，您可以在 `TRTCCalling.dart` 文件中看到该组件提供的接口函数。



您可以使用开源组件 TRTCCalling 实现自己的 UI 界面，即只复用 model 部分，自行实现 UI 部分。

步骤1：集成 SDK

音视频通话组件 TRTCCalling 依赖 TRTC SDK 和 IM SDK，您可以通过配置 `pubspec.yaml` 自动下载更新。

在项目的 `pubspec.yaml` 中写如下依赖：

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

```
tencent_im_sdk_plugin: 最新版本号
```

步骤2：配置权限及混淆规则

- iOS 端
- Android 端

需要在 `Info.plist` 中加入对相机和麦克风的权限申请：

```
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

步骤3：导入 TRTCCalling 组件

拷贝以下目录中的所有文件到您的项目中：

```
/lib/TRTCCallingDemo/model
```

步骤4：初始化并登录组件

1. 调用 `TRTCCalling.sharedInstance()` 获取组件实例。
2. 调用 `login(SDKAppID, userId, userSig)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
SDKAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。

```
// 初始化
sCall = await TRTCCalling.sharedInstance();
sCall.login(1400000123, "userA", "xxxx");
```

步骤5：实现 1v1 视频通话

1. 发起方：调用 `TRTCCalling` 的 `call()` 方法发起通话的请求，并传入用户 ID（userid）和通话类型（type），通话类型参数传入 `TRTCCalling.typeVideoCall` 。
2. 接收方：当接收方处于已登录状态时，会收到名为 `onInvited()` 的事件通知，回调中 `callType` 的参数是发起方填写的通话类型，您可以通过此参数启动相应的界面。

3. 接收方：如果希望接听电话，接收方可以调用 `accept()` 函数，并同时调用 `openCamera()` 函数打开自己本地的摄像头。接收方也可以调用 `reject()` 拒绝此次通话。
4. 当双方的音视频通道建立完成后，通话的双方都会接收到名为 `onUserVideoAvailable()` 的事件通知，表示对方的视频画面已经拿到。此时双方用户均可以调用 `startRemoteView()` 展示远端的视频画面。远端的声音默认是自动播放的。

组件 API 列表

TRTCCalling 组件的 API 接口列表如下：

接口函数	接口功能
registerListener	增加 TRTCCalling 监听器，用户可以通过该监听器获取状态通知
unRegisterListener	移除监听器
destroy	销毁实例
login	登录 IM，所有功能需要先进行登录后才能使用
logout	登出 IM，登出后无法再进行拨打操作
call	C2C 邀请通话，被邀请方会收到 onInvited 的事件通知
accept	作为被邀请方接听来电
reject	作为被邀请方拒绝来电
hangup	结束通话
startRemoteView	将远端用户的摄像头数据渲染到指定的 TXCloudVideoView 中
stopRemoteView	停止渲染某个远端用户的摄像头数据
openCamera	开启摄像头，并渲染在指定的 TXCloudVideoView 中
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
setMicMute	是否静音 mic
setHandsFree	是否开启免提

版权所有：腾讯云计算（北京）有限责任公司

TUICalling API（iOS）

最近更新时间：2021-11-12 16:31:40

TUICalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频通话。TUICalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时视频通话（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TUICalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例。
call	C2C 邀请通话。
receiveAPNSCalled	作为被邀请方接听来电。
setCallingListener	设置监听器。
setCallingBell	设置铃声（建议在30s以内）
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图

TUICallingListener API 概览

事件回调

API	描述
shouldShowOnCallView	被叫时请求拉起接听页面
onCallStart	呼叫开始回调。主叫、被叫均会触发
onCallEnd	通话回调。主叫、被叫均会触发
onCallEvent	通话事件回调

Type API 概览

通话类型

enum	描述
TUICallingTypeAudio	音频通话
TUICallingTypeVideo	视频通话

Role API 概览

用户角色类型

enum	描述
TUICallingRoleCall	通话发起方（主叫）
TTUICallingRoleCalled	通话接听方（被叫）

Event API 概览

事件类型

enum	描述
TUICallingEventCallStart	通话开始
TUICallingEventCallSucceed	通话接通成功
TUICallingEventCallEnd	通话结束
TUICallingEventCallFailed	通话失败

SDK 基础函数

sharedInstance

sharedInstance 是 TUICallingManager 的组件单例。

```
+ (instancetype)shareInstance;
```

call

C2C 邀请通话。

```
- (void)call:(NSArray<NSString *> *)userIDs type:(TUICallingType)type;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频

receiveAPNSCalled

作为被邀请方接听来电。

```
- (void)receiveAPNSCalled:(NSArray<NSString *> *)userIDs type:(TUICallingType)type;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频

setCallingListener

设置监听器。

```
- (void)setCallingListener:(id<TUICallingListerner>)listener;
```

参数如下表所示：

参数	类型	含义
listener	TUICallingListener	TUIcalling 组件监听器

setCallingBell

设置铃声（建议在30s以内）。

```
- (void)setCallingBell:(NSString *)filePath;
```

参数如下表所示：

参数	类型	含义
filePath	NSString	铃音资源路径

enableMuteMode

开启静音模式。

```
- (void)enableMuteMode:(BOOL)enable;
```

参数如下表所示：

参数	类型	含义
enable	BOOL	是否开启静音模式

enableCustomViewRoute

开启自定义视图。
开启后，会在呼叫/被叫开始回调中，接收到 CallingViewController 的实例，由开发者自行决定展示方式。

> 注意：
> 必须全屏展示，否则会有展示异常。

```
- (void)enableCustomViewRoute:(BOOL)enable;
```

参数如下表所示：

参数	类型	含义
enable	BOOL	是否开启自定义视图

TUICallingListener 回调函数

shouldShowOnCallView

是否同意被叫时请求拉起接听页面。

```
- (BOOL)shouldShowOnCallView;
```

参数如下表所示：

参数	类型	含义
返回值	BOOL	是否同意

onCallStart

呼叫开始回调。主叫、被叫均会触发。

```
- (void)callStart:(NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)r
ole viewController:(UIViewController * _Nullable)viewController;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫
viewController	UIViewController	通话视图 ViewController

onCallEnd

通话结束回调。主叫、被叫均会触发。enableCustomViewRoute 设置为 NO 时，此回调方法不会触发。

```
- (void)callEnd:(NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)rol
e totalTime:(CGFloat)totalTime;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫

参数	类型	含义
totalTime	CGFloat	通话时长，单位：秒

onCallEvent

通话事件回调。enableCustomViewRoute 设置为 NO 时，此回调方法不会触发。

```
- (void)onCallEvent:(TUICallingEvent)event type:(TUICallingType)type role:(TUICallingRole)role me
ssage:(NSString *)message;
```

参数如下表所示：

参数	类型	含义
event	TUICallingEvent	通话事件类型。
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫
message	NSString	事件的描述信息

版权所有：腾讯云计算（北京）有限责任公司

TUICalling API（Android）

最近更新时间：2021-11-12 16:32:09

TUICalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频通话。TUICalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时视频通话（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TUICalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器。
setCallingBell	设置铃声（建议在30s以内）
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图

TUICallingListener API 概览

事件回调

API	描述
shouldShowOnCallView	被叫时请求拉起接听页面
onCallStart	呼叫开始回调。主叫、被叫均会触发
onCallEnd	通话回调。主叫、被叫均会触发

API	描述
onCallEvent	通话事件回调

Type API 概览

通话类型

enum	描述
AUDIO	音频通话
VIDEO	视频通话

Role API 概览

用户角色类型

enum	描述
CALL	通话发起方（主叫）
CALLED	通话接听方（被叫）

Event API 概览

事件类型

enum	描述
CALL_START	通话开始
CALL_SUCCEED	通话接通成功
CALL_END	通话结束
CALL_FAILED	通话失败

SDK 基础函数

sharedInstance

sharedInstance 是 TUICalling 的组件单例。

```
public static TUICallingManager sharedInstance();
```

call

C2C 邀请通话。

```
void call(String[] userIDs, Type type);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表
type	TUICalling.Type	通话类型：音频/视频

receiveAPNSCalled

作为被邀请方接听来电。

```
void receiveAPNSCalled(String[] userIDs, Type type);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表
type	TUICalling.Type	通话类型：音频/视频

setCallingListener

设置监听器。

```
void setCallingListener(TUICallingListener listener);
```

参数如下表所示：

参数	类型	含义
listener	TUICallingListener	TUIcalling 组件监听器

版权所有：腾讯云计算（北京）有限责任公司

setCallingBell

设置铃声(建议在30s以内)。

```
void setCallingBell(String filePath);
```

参数如下表所示：

参数	类型	含义
filePath	String	铃音资源路径

enableMuteMode

开启静音模式。

```
void enableMuteMode(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	是否开启静音模式

enableCustomViewRoute

开启自定义视图。
开启后，会在呼叫/被叫开始回调中，接收到 CallingView 的实例，由开发者自行决定展示方式。

> 注意：
> 必须全屏或者与屏幕等比例展示，否则会有展示异常。

```
void enableCustomViewRoute(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	是否开启自定义视图

TUICallingListener 回调函数

shouldShowOnCallView

是否同意被叫时请求拉起接听页面。

```
boolean shouldShowOnCallView();
```

参数如下表所示：

参数	类型	含义
返回值	boolean	是否同意

onCallStart

呼叫开始回调。主叫、被叫均会触发。

```
void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, View tuiCallingVie
w);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表。
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
tuiCallingView	View	通话视图 View。enableCustomViewRoute 设置为 false 时，view 为 null

onCallEnd

通话结束回调。主叫、被叫均会触发。

```
void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTime);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表

参数	类型	含义
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
totalTime	long	通话时长，单位：秒

onCallEvent

通话事件回调。

```
void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, String message);
```

参数如下表所示：

参数	类型	含义
event	TUICalling.Event	通话事件类型
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
message	String	事件的描述信息

版权所有：腾讯云计算（北京）有限责任公司

TRTCCalling API（Web）

最近更新时间：2022-04-02 16:27:03

TRTCCalling 简介

TRTCCalling 组件是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频/语音通话。具体的实现过程请参见 实时视频通话（Web）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

下载 Demo

单击进入 TUICalling，根据实际业务需求下载 Demo 源码。

环境要求

请使用最新版本的 Chrome 浏览器。目前桌面端 Chrome 浏览器支持 TRTC Web SDK 的相关特性比较完整，因此建议使用 Chrome 浏览器进行体验。具体请参见 环境要求。

URL 域名协议限制

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

版权所有：腾讯云计算（北京）有限责任公司

TRTCCalling API

事件订阅/取消订阅相关接口函数

本组件基于事件分发进行管理，应用层可以根据组件下发的事件进行交互的改变。

API	描述
on(eventName, callback, context)	订阅事件
off(eventName, callback, context)	取消事件订阅

SDK 基础函数

API	描述
login({userID, userSig})	登录 IM 接口，所有功能需要先进行登录后才能使用
logout()	登出接口，登出后无法再进行拨打操作

通话操作相关接口函数

API	描述
call({userID, type, offlinePushInfo}))	单人通话邀请
groupCall({userIDList, type, groupID, offlinePushInfo})	群聊通话邀请
accept()	接受通话邀请
reject()	拒绝通话邀请
hangup()	挂断当前通话

视频控制相关接口函数

API	描述
startRemoteView({userID, videoViewDomID})	启动远端画面渲染
stopRemoteView({userID})	停止远端画面渲染
startLocalView({userID, videoViewDomID})	启动本地画面渲染
stopLocalView({userID})	停止本地画面渲染

版权所有：腾讯云计算（北京）有限责任公司

API	描述
openCamera()	启动摄像头
closeCamera()	关闭摄像头
setMicMute(isMute)	设备麦克风是否静音
setVideoQuality(profile)	设置视频质量
switchToAudioCall()	视频通话切换语音通话
switchToVideoCall()	语音通话切换视频通话
getCameras()	获取摄像头设备列表
getMicrophones()	获取麦克风设备列表
switchDevice({deviceType, deviceId})	切换摄像头或麦克风设备

TRTCCalling 详解

创建 TRTCCalling 组件实例

首先，您需要在 实时音视频控制台 中创建一个应用，并取得 SDKAppID。

之后，就可以通过 `new TRTCCalling()` 方法获取 TRTCCalling 组件的一个实例。

```
npm i trtc-js-sdk --save
npm i tim-js-sdk --save
npm i tsignaling --save
npm i trtc-calling-js --save
// 如果是通过node下载的依赖，则使用 import 引入
import TRTCCalling from 'trtc-calling-js';
// 如果您通过 script 方式使用 trtc-calling-js，需要按顺序
// 手动引入 trtc.js
<script src="./trtc.js"></script>
// 接着手动引入 tim-js.js
<script src="./tim-js.js"></script>
// 然后再手动引入 tsignaling.js
<script src="./tsignaling.js"></script>
// 最后再手动引入 trtc-calling-js.js
<script src="./trtc-calling-js.js"></script>
let options = {
SDKAppID: 0, // 接入时需要将0替换为您的即时通信IM应用的 SDKAppID
// 从v0.10.2起，新增 tim 参数
```

```
// tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性
tim: tim
};
let trtcCalling = new TRTCCalling(options);
```

事件订阅/取消订阅相关接口函数

on(eventName, callback, context)

用于监听组件派发的事件，详细事件请参见 事件表。

```
let handleInvite = function ({inviteID, sponsor, inviteData}) {
console.log(`inviteID: ${inviteID}, sponsor: ${sponsor}, inviteData: ${JSON.stringify(inviteDat
a)}`);
};
trtcCalling.on('onInvited', handleInvite, this);
```

off(eventName, callback, context)

用于取消事件监听。

```
let handleInvite = function ({inviteID, sponsor, inviteData}) {
console.log(`inviteID: ${inviteID}, sponsor: ${sponsor}, inviteData: ${JSON.stringify(inviteDat
a)}`);
};
trtcCalling.off('onInvited', handleInvite, this);
```

SDK 基础函数

login({userID, userSig})

登录接口。

```
trtcCalling.login({userID, userSig})
```

参数如下表所示：

参数	类型	含义
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。

logout()

登出接口。

```
trtcCalling.logout()
```

通话操作相关接口函数

call({userID, type, offlinePushInfo})

1对1通话邀请，其中 type 为通话类型，1-语音通话，2-视频通话。

> 说明：
>
> - v1.0.0 及其之后版本，取消 timeout 参数。
> - v1.0.0 及其之后版本，新增 offlinePushInfo 参数（**离线推送仅适用于 Android 或 iOS 终端，Web 和微信小程序不支持**）。

```
// v1.0.0 之前
trtcCalling.call({userID, type, timeout});

// v1.0.0 版本及其之后
const offlinePushInfo = {
title: '',
description: '您有一个通话请求',
}
trtcCalling.call({userID, type, offlinePushInfo})
```

参数如下表所示：

参数	类型	含义
userID	String	被邀请方 userID
type	Number	1：语音通话，2：视频通话
timeout	Number	0为不超时，单位 s（秒）。**仅限于v1.0.0之前的版本**
offlinePushInfo	Object	自定义离线消息推送（选填)。**仅限于v1.0.0及其之后的版本**

offlinePushInfo 参数 (仅限于v1.0.0及其之后的版本)

参数	类型	含义

参数	类型	含义
title	String	离线推送标题（选填）
description	String	离线推送内容（选填）
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
extension	String	离线推送透传内容（选填）。**仅限于TRTCCalling 版本>=1.0.2, tsignaling 版本 >= 0.9.0**

groupCall({userIDList, type, groupID, offlinePushInfo})

groupID 参数是 IM SDK 中的群组 ID，如果填写该参数，那么通话请求消息是通过群消息系统广播出去的，这种消息广播方式比较简单可靠。如果不填写，那么 TRTCCalling 组件会采用单发消息逐一通知。

> 说明：
> v1.0.0 及其之后版本，新增 offlinePushInfo 参数（**离线推送仅适用于 Android 或 iOS 终端，Web 和微信小程序不支持**）。

```
// v1.0.0之前
trtcCalling.groupCall({userIDList, type, groupID});

// v1.0.0及其之后
const offlinePushInfo = {
title: '',
description: '您有一个通话请求',
}
trtcCalling.groupCall({userIDList, type, groupID, offlinePushInfo})
```

参数如下表所示：

参数	类型	含义
userIDList	Array	邀请列表
type	Number	1：语音通话，2：视频通话
groupID	String	IM 群组 ID（选填）
offlinePushInfo	Object	自定义离线消息推送（选填）。**仅限于v1.0.0及其之后的版本**

offlinePushInfo 参数 (仅限于v1.0.0及其之后的版本)

参数	类型	含义
title	String	离线推送标题（选填）
description	String	离线推送内容（选填）
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
extension	String	离线推送透传内容（选填）。**仅限于TRTCCalling 版本>=1.0.2, tsignaling 版本 >= 0.9.0**

accept()

当收到邀请后，调用该接口将接受当前的邀请。

说明：

- 当上一个 invitation 未处理完成时，组件会默认占线，之后的邀请都会回复忙线。
- v1.0.0 及其之后版本，取消 params 参数。

```
import TRTCCalling from 'trtc-calling-js';
trtcCalling.on(TRTCCalling.EVENT.INVITED, ({inviteID, sponsor, inviteData}) => {
// ...
// v1.0.0之前
const { roomID, callType } = inviteData;
trtcCalling.accept({inviteID, roomID, callType})
// v1.0.0及其之后
trtcCalling.accept();
})
```

参数如下表所示：

参数	类型	含义
inviteID	String	邀请 ID，标识一次邀请（监听事件 INVITED 回调数据 inviteID）。**仅限于v1.0.0之前的版本**
roomID	Number	通话房间号 ID（监听事件 INVITED 回调数据 inviteData.roomID）。**仅限于v1.0.0之前的版本**
callType	Number	1：语音通话，2：视频通话（监听事件 INVITED 回调数据 inviteData.callType）。**仅限于v1.0.0之前的版本**

reject()

当收到邀请后，调用该接口将拒绝当前的邀请。

> 说明：
> v1.0.0 及其之后版本，取消 params 参数。

```
import TRTCCalling from 'trtc-calling-js';
trtcCalling.on(TRTCCalling.EVENT.INVITED, ({inviteID, sponsor, inviteData}) => {
// ...
// v1.0.0之前
const { callType } = inviteData;
trtcCalling.reject({inviteID, isBusy, callType})
// v1.0.0及其以后
trtcCalling.reject();
})
```

参数如下表所示：

参数	类型	含义
inviteID	String	邀请 ID, 标识一次邀请（监听事件 INVITED 回调数据 inviteID）。**仅限于v1.0.0之前的版本**
isBusy	Boolean	是否是忙线中。**仅限于v1.0.0之前的版本**
callType	Number	1：语音通话，2：视频通话（监听事件 INVITED 回调数据 inviteData.callType）。**仅限于v1.0.0之前的版本**

hangup()

1. 当您处于通话中，可以调用该函数结束通话。
2. 当未拨通时，可用来取消通话。

```
trtcCalling.hangup()
```

视频控制相关接口函数

startRemoteView({userID, videoViewDomID})

将远端用户的摄像头数据渲染到指定的 DOM ID 节点里。

```
trtcCalling.startRemoteView({userID, videoViewDomID})
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该用户数据将通过渲染到该 DOM ID 节点的 video 标签进行播放

stopRemoteView({userID})

将远端用户的摄像头数据渲染的 DOM 节点删除。

> 说明：
> v1.0.0 及其之后版本，移除 videoViewDomID 参数。

```
// v1.0.0之前
trtcCalling.stopRemoteView({userID, videoViewDomID});
// v1.0.0及其之后
trtcCalling.stopRemoteView({userID});
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该 DOM ID 节点的 video 标签进行移除，停止播放视频。**仅限于v1.0.0之前的版本**

startLocalView({userID, videoViewDomID})

将本地用户的摄像头数据渲染到指定的 DOM ID 节点里。

```
trtcCalling.startLocalView({userID, videoViewDomID})
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

参数	类型	含义
videoViewDomID	String	本地用户数据将通过渲染到该 DOM ID 节点的 video 标签进行播放

stopLocalView({userID})

将本地用户的摄像头数据渲染的 DOM 节点删除。

> 说明：
> v1.0.0 及其之后版本，移除 videoViewDomID 参数。

```
// v1.0.0之前
trtcCalling.stopLocalView({userID, videoViewDomID});
// v1.0.0及其之后
trtcCalling.stopLocalView({userID});
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该 DOM ID 节点的 video 标签进行移除，停止播放视频。**仅限于v1.0.0之前的版本**

openCamera()

开启本地摄像头。

```
trtcCalling.openCamera()
```

closeCamera()

关闭摄像头。

```
trtcCalling.closeCamera()
```

setMicMute(isMute)

开启/关闭麦克风。

```
trtcCalling.setMicMute(true) // 关闭麦克风
```

参数如下表所示：

参数	类型	含义
isMute	Boolean	• true：麦克风关闭 • false：麦克风打开

setVideoQuality(profile)

设置视频质量。

> 说明：
>
> - v0.8.0 及其之后版本，新增该方法。
> - 此方法需在 call、groupCall、accept 之前设置，之后设置不生效。

```
trtcCalling.setVideoQuality('720p') // 设置视频质量为720p
```

参数如下表所示：

参数	类型	含义
profile	String	• 480p：640 × 480 • 720p：1280 × 720 • 1080p：1920 × 1080

switchToAudioCall()

视频通话切换语音通话。

> 说明：
>
> - v0.10.0 及其之后版本，新增该方法。
> - 仅支持1v1通话过程中使用。
> - 失败监听 ERROR 事件，code：60001。

```
trtcCalling.switchToAudioCall() // 视频通话切换语音通话
```

switchToVideoCall()

语音通话切换视频通话。

> 说明：
>
> - v0.10.0 及其之后版本，新增该方法。
> - 仅支持1v1通话过程中使用。
> - 失败监听 ERROR 事件，code：60002。

```
trtcCalling.switchToVideoCall() // 语音通话切换视频通话
```

getCameras()

您可以调用此接口获取摄像头设备列表。

> 说明：
> v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.getCameras() // 获取摄像头列表
```

getMicrophones()

您可以调用此接口获取麦克风设备列表。

> 说明：
> v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.getMicrophones() // 获取麦克风列表
```

switchDevice({deviceType, deviceId})

您可以调用此接口切换摄像头或麦克风设备。

说明：
v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.switchDevice({deviceType: 'video', deviceId: deviceId}) // 切换设备
```

参数如下表所示：

参数	类型	含义
deviceType	String	video：摄像头, audio：麦克风
deviceId	String	• 摄像头设备标识通过 getCameras() 获取 • 麦克风设备标识通过 getMicrophones() 获取

TRTCCalling 事件表

您可以参考如下代码监听 TRTCCalling 组件事件：

```
import TRTCCalling from 'trtc-calling-js';
// etc
function handleInviteeReject({userID}) {

}
trtcCalling.on(TRTCCalling.EVENT.REJECT, handleInviteeReject)
```

邀请方事件

CODE	事件接收方	说明
REJECT	邀请方	被邀用户拒绝通话
NO_RESP	邀请方	被邀用户超时无应答
LINE_BUSY	邀请方	被邀用户正在通话中，忙线
INVITED	被邀方	收到了邀请通知
CALLING_CANCEL	被邀方	本次通话被取消了
CALLING_TIMEOUT	被邀方	本次通话超时未应答

CODE	事件接收方	说明
USER_ENTER	邀请方和被邀方	用户进房
USER_LEAVE	邀请方和被邀方	有用户离开通话
CALL_END	邀请方和被邀方	本次通话结束
KICKED_OUT	邀请方和被邀方	重复登录，被踢出房间
USER_VIDEO_AVAILABLE	邀请方和被邀方	远端用户开启/关闭了摄像头
USER_AUDIO_AVAILABLE	邀请方和被邀方	远端用户开启/关闭了麦克风

通用事件回调

SDK_READY

SDK 进入 ready 状态时收到该回调

> 说明：
> v1.0.0 及其之后版本，新增此事件。

```
let onSDKReady = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.SDK_READY, onSDKReady);
```

USER_ENTER

用户进房。
触发条件：当有用户进入通话。

```
let handleUserEnter = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.USER_ENTER, handleUserEnter);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userID	String	用户 ID

USER_LEAVE

用户退出房间。

触发条件：当有用户退出通话。

```
let handleUserLeave = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.USER_LEAVE, handleUserLeave);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

GROUP_CALL_INVITEE_LIST_UPDATE

群聊更新邀请列表收到该回调。

> 说明：
> v1.0.0 及其之后版本，新增此事件。

```
let handleGroupInviteeListUpdate = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeListUpdate);
```

CALL_END

本次通话结束。

触发条件：结束本次通话。

```
let handleCallingEnd = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALL_END, handleCallingEnd);
```

KICKED_OUT

重复登录，被踢出房间。

触发条件：在其他页面重复登录。

```
let handleKickedOut = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.KICKED_OUT, handleKickedOut);
```

USER_VIDEO_AVAILABLE

远端用户打开关闭摄像头。

触发条件：远端用户打开/关闭摄像头。

```
let handleUserVideoChange = function({userID, isVideoAvailable}) {
console.log(userID, isVideoAvailable)
};
trtcCalling.on(TRTCCalling.EVENT.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
isVideoAvailable	Boolean	• true：远端用户打开摄像头 • false：远端用户关闭摄像头

USER_AUDIO_AVAILABLE

远端用户开启/关闭了麦克风。

触发条件：远端用户开启/关闭了麦克风。

```
let handleUserAudioChange = function({userID, isAudioAvailable}) {
console.log(userID, isAudioAvailable)
};
trtcCalling.on(TRTCCalling.EVENT.USER_AUDIO_AVAILABLE, handleUserAudioChange);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

参数	类型	含义
isAudioAvailable	Boolean	• true：远端用户打开麦克风 • false：远端用户关闭麦克风

邀请方事件回调

REJECT

用户拒绝通话。

触发条件：被邀请方拒绝通话，发起方收到 REJECT 事件回调。

```
let handleInviteeReject = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.REJECT, handleInviteeReject);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

NO_RESP

邀请用户无应答。

触发条件：当 call/groupCall 设置 timeout，被邀请方未在 timeout 内未在接听，发起方收到 NO_RESP 事件回调。

```
let handleNoResponse = function({userID, userIDList}) {
console.log(userID, userIDList)
};
trtcCalling.on(TRTCCalling.EVENT.NO_RESP, handleNoResponse);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
userIDList	Array	超时用户列表

LINE_BUSY

被邀请方正在通话中，忙线。

触发条件：被邀请方已再另一通话中，发起方收到 LINE_BUSY 事件回调。

```
let handleLineBusy = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.LINE_BUSY, handleLineBusy);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

被邀请方事件回调

INVITED

收到邀请通知。

触发条件：当有邀请通话时，被邀请方收到 INVITED 事件回调。

```
let handleNewInvitationReceived = function({
sponsor, userIDList, isFromGroup, inviteData, inviteID
}) {
console.log(sponsor, userIDList, isFromGroup, inviteData, inviteID)
};
trtcCalling.on(TRTCCalling.EVENT.INVITED, handleNewInvitationReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
userIDList	Array	同时还被邀请的人
isFromGroup	Boolean	是否 IM 群组邀请
inviteData	Object	针对新用户邀请： {version, callType, roomID}
inviteID	String	邀请 ID，标识一次邀请

CALLING_CANCEL

本次通话被取消了。

触发条件：发起方在呼叫过程中取消通话，被邀请方收到 CALLING_CANCEL 事件回调。

```
let handleCallingCancel = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALLING_CANCEL, handleCallingCancel);
```

CALLING_TIMEOUT

本次通话超时未应答。

触发条件：当 call/groupCall 设置 timeout，被邀请方未在 timeout 内未在接听，被邀请方收到 CALLING_TIMEOUT 事件回调。

```
let handleCallingTimeout = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALLING_TIMEOUT, handleCallingTimeout);
```

TRTCCalling 错误码表

您可以通过监听 EVENT 里的 ERROR 字段，对组件抛出的错误进行处理，示例代码如下：

```
import TRTCCalling from 'trtc-calling-js';
let onError = function(error) {
console.log(error)
};
trtcCalling.on(TRTCCalling.EVENT.ERROR, onError);
```

Error code 码

code	错误类型	含义
60001	方法调用失败	switchToAudioCall 调用失败
60002	方法调用失败	switchToVideoCall 调用失败
60003	权限获取失败	没有可用的麦克风设备
60004	权限获取失败	没有可用的摄像头设备
60005	权限获取失败	用户禁止使用设备
60006	环境检测失败	当前环境不支持 WebRTC（>=v1.0.4版本）

升级指引

- **升级 TRTCCalling 版本 >= 1.0.2**
  - 注意：需升级 TSignaling 版本 >= 0.9.0
  - 原因：更新日志
- **升级 1.0.2 > TRTCCalling 版本 >=1.0.0**
  - 注意：需升级 TSignaling 版本 >= 0.8.0
  - 原因：更新日志

常见问题

为什么拨打不通，或者被踢下线？

组件暂不支持多实例登入，不支持**离线推送信令**功能，请您确认登入账号的唯一性。

> 说明：
>
> - **多实例**：一个 UserID 重复登入，或在不同端登入，将会引起信令的混乱。
> - **离线推送**：实例在线才能接收消息，实例离线时接收到的信令不会在上线后重新推送。
>
> 更多常见问题，请参见 TRTCCalling Web 相关问题。

技术咨询

了解更多详情您可以联系我们或发送邮件至colleenyu@tencent.com。

参考文档

- TRTCCalling web 官网体验
- TRTCCalling npm
- TRTCCalling web demo 源码
- TRTCCalling web API
- TRTCCalling web 相关问题

TRTCCalling API（Flutter）

最近更新时间：2022-01-10 11:12:11

TRTCCalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1音视频通话。TRTCCalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时视频通话（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TRTCCalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例。
destroySharedInstance	销毁组件单例。
registerListener	增加事件回调。
unRegisterListener	移除回调接口。
destroy	销毁函数，如果无需再运行该实例，请调用该接口。
login	登录组件接口，所有功能需要先进行登录后才能使用。
logout	登出组件接口，登出后无法再进行拨打操作。

通话操作相关接口函数

API	描述
call	单人通话邀请。
accept	接受当前通话。
reject	拒绝当前通话。
hangup	结束当前通话。

推拉流相关接口函数

API	描述
startRemoteView	远端用户的摄像头数据渲染到指定的 TXCloudVideoView 中。
stopRemoteView	停止渲染远端数据。

音视频控制相关接口函数

API	描述
openCamera	开启摄像头，并渲染在指定的 TXCloudVideoView 中。
switchCamera	切换前后摄像头。
closeCamara	关闭摄像头。
setMicMute	静音本地音频采集。
setHandsFree	设置免提。

TRTCCallingDelegate API 概览

通用事件回调

API	描述
onError	错误回调。

邀请方回调

API	描述
onReject	拒绝通话回调。
onNoResp	对方无回应回调。
onLineBusy	通话忙线回调。

被邀请方回调

API	描述
onInvited	被邀请通话回调。

API	描述
onCallingCancel	当前通话被取消回调。
onCallingTimeOut	当前通话超时回调。

通用回调

API	描述
onUserEnter	用户进入通话回调。
onUserLeave	用户离开通话回调。
onUserAudioAvailable	用户是否开启音频上行回调。
onUserVideoAvailable	用户是否开启视频上行回调。
onUserVoiceVolume	用户通话音量回调。
onCallEnd	通话结束回调。

SDK 基础函数

sharedInstance

sharedInstance 是 TRTCCalling 的组件单例。

```
static Future<TRTCCalling> sharedInstance();
```

destroySharedInstance

销毁组件单例。

```
static void destroySharedInstance();
```

destroy

销毁函数，如果无需再运行该实例，请调用该接口。

```
void destroy();
```

registerListener

TRTCCalling 事件回调，您可以通过 TRTCCallingDelegate 获得 TRTCCalling 的各种状态通知。

```
void registerListener(VoiceListenerFunc func);
```

unRegisterListener

移除回调接口。

```
void unRegisterListener(VoiceListenerFunc func);
```

login

登录组件。

```
Future<ActionCallback> login(int sdkAppId, String userId, String userSig);
```

参数如下表所示：

参数	类型	含义
sdkAppID	int	您可以在实时音视频控制台 >【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

logout

登出组件。

```
Future<ActionCallback> logout();
```

通话操作相关接口函数

call

单人通话邀请，当前处于通话中也可继续调用邀请他人。

```
Future<ActionCallback> call(String userId, int type);
```

参数如下表所示：

参数	类型	含义
userId	String	呼叫用户 ID。
type	int	1 表示语音通话，2 表示视频通话。

accept

接受当前通话。当您作为被邀请方收到 `onInvited()` 的回调时，可以调用该函数接听来电。

```
Future<ActionCallback> accept();
```

reject

拒绝当前通话。当您作为被邀请方收到 `onInvited()` 的回调时，可以调用该函数拒绝来电。

```
Future<ActionCallback> reject();
```

hangup

挂断当前通话。当您处于通话中，可以调用该函数结束通话。

```
void hangup();
```

推拉流相关接口函数

startRemoteView

远端用户的摄像头数据渲染到指定的 TRTCCloudVideoView 中。

```
void startRemoteView(String userId, int streamType, int viewId);
```

参数如下表所示：

参数	类型	含义
userId	String	远端用户 ID。
streamType	int	指定要停止观看的 userId 的视频流类型。

参数	类型	含义
viewId	int	承载视频画面的控件 TRTCCloudVideoView 回调的 viewId。

stopRemoteView

停止渲染远端数据。

```
void stopRemoteView(String userId, int streamType);
```

参数如下表所示：

参数	类型	含义
userId	String	远端用户 ID。
streamType	int	指定要停止观看的 userId 的视频流类型。

音视频控制相关接口函数

openCamera

开启摄像头，并渲染在指定的 TRTCCloudVideoView 中。

```
void openCamera(bool isFrontCamera, int viewId);
```

参数如下表所示：

参数	类型	含义
isFrontCamera	bool	true 表示开启前置摄像头，false 表示开启后置摄像头。
viewId	int	承载视频画面的控件 TRTCCloudVideoView 回调的 viewId。

switchCamera

切换前后摄像头。

```
void switchCamera(bool isFrontCamera);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
isFrontCamera	bool	true 表示切换到前置摄像头，false 表示切换到后置摄像头。

closeCamara

关闭摄像头。

```
void closeCamera();
```

setMicMute

静音本地音频采集。

```
void setMicMute(bool isMute);
```

参数如下表所示：

参数	类型	含义
isMute	bool	true 表示麦克风关闭，false 表示麦克风打开。

setHandsFree

静音远端音频。

```
void setHandsFree(bool isHandsFree);
```

参数如下表所示：

参数	类型	含义
isHandsFree	bool	true 表示开启免提，false 表示关闭免提。

TRTCCallingDelegate 事件回调

通用事件回调

onError

错误回调。

说明：
SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

参数如下表所示：

参数	类型	含义
code	int	错误码。
msg	String	错误信息。

邀请方回调

onReject

拒绝通话回调。

参数如下表所示：

参数	类型	含义
userId	String	拒绝用户的 ID。

onNoResp

对方无回应回调。

参数如下表所示：

参数	类型	含义
userId	String	无回应用户的 ID。

onLineBusy

通话忙线回调。

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID。

被邀请方回调

onInvited

被邀请通话回调。

参数如下表所示：

参数	类型	含义
sponsor	String	发起方的 ID。
userIds	List<String>	除自己外被邀请 ID 列表。
isFromGroup	bool	是否多人通话邀请。
type	int	1 表示语音通话，2 表示视频通话。

onCallingCancel

当前通话被取消回调。接收方未处理请求，邀请方取消后会收到此回调。

onCallingTimeOut

当前通话超时回调。

通用回调

onUserEnter

用户进入通话回调。

参数如下表所示：

参数	类型	含义
userId	String	进入通话用户 ID。

onUserLeave

用户离开通话回调。

参数如下表所示：

参数	类型	含义

参数	类型	含义
userId	String	离开通话用户 ID。

onUserAudioAvailable

用户是否开启音频上行回调。

参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID。
available	boolean	用户音频是否可用。

onUserVideoAvailable

用户是否开启视频上行回调。收到通知后，用户可调用 startRemoteView 渲染远端视频。

参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID。
available	boolean	用户视频是否可用。

onUserVoiceVolume

用户通话音量回调。

参数如下表所示：

参数	类型	含义
userVolumes	List	所有正在说话的房间成员的音量，取值范围：0 - 100。
totalVolume	int	所有远端成员的总音量，取值范围：0 - 100。

onCallEnd

通话结束回调。

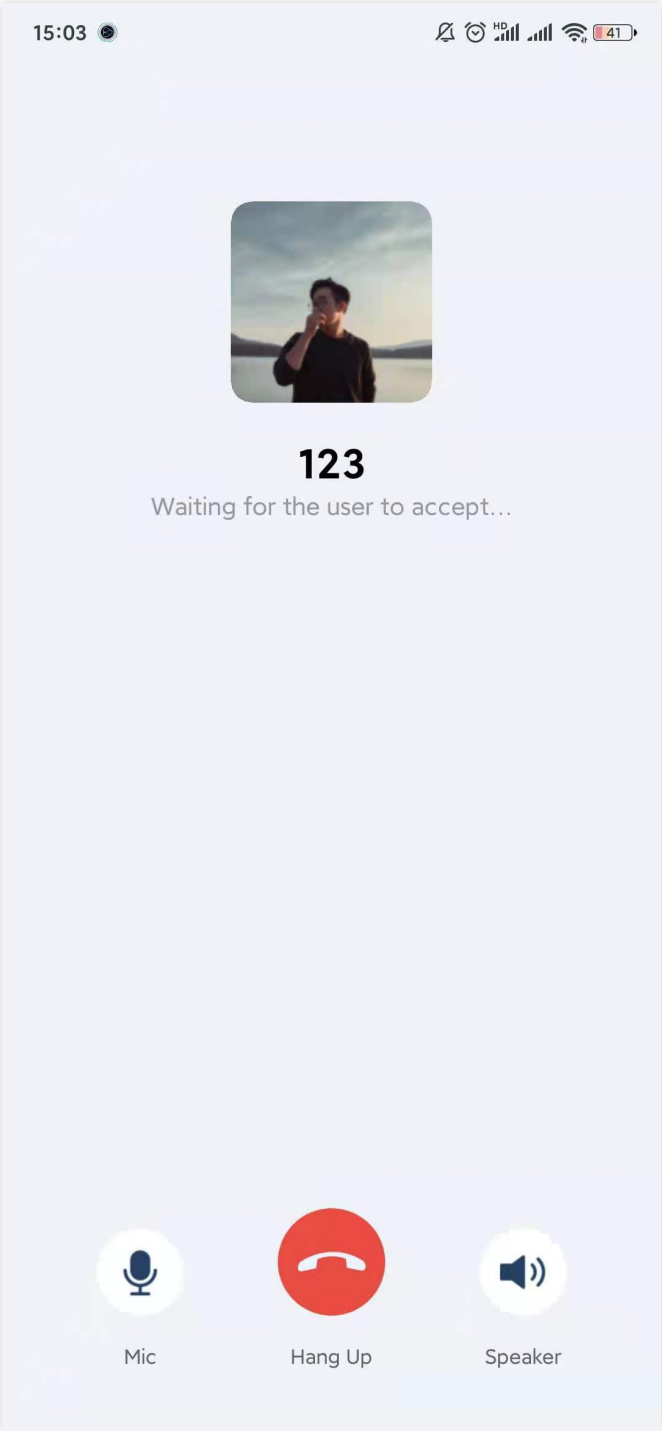
语音通话
语音通话(iOS)

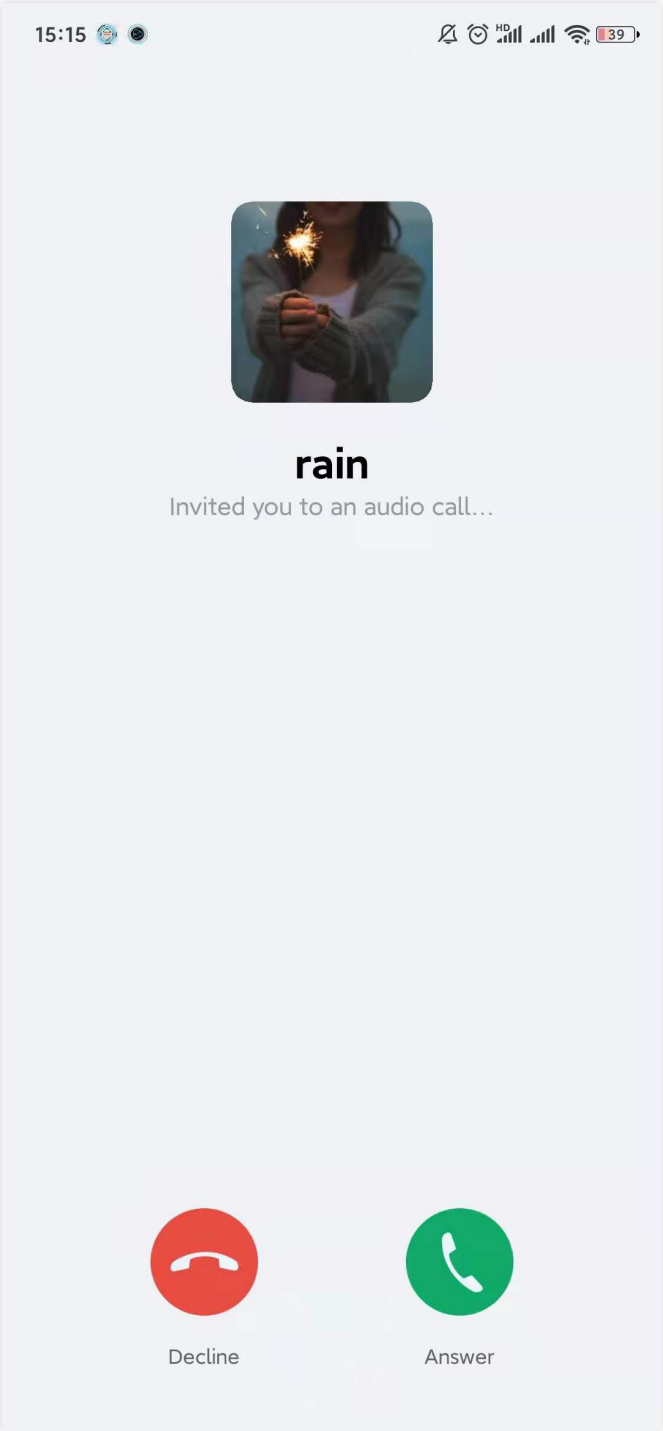
最近更新时间：2022-03-10 11:13:14

效果展示

您可以 下载 安装我们的 App 体验实时音视频通话的效果。

主动呼叫	被叫接听



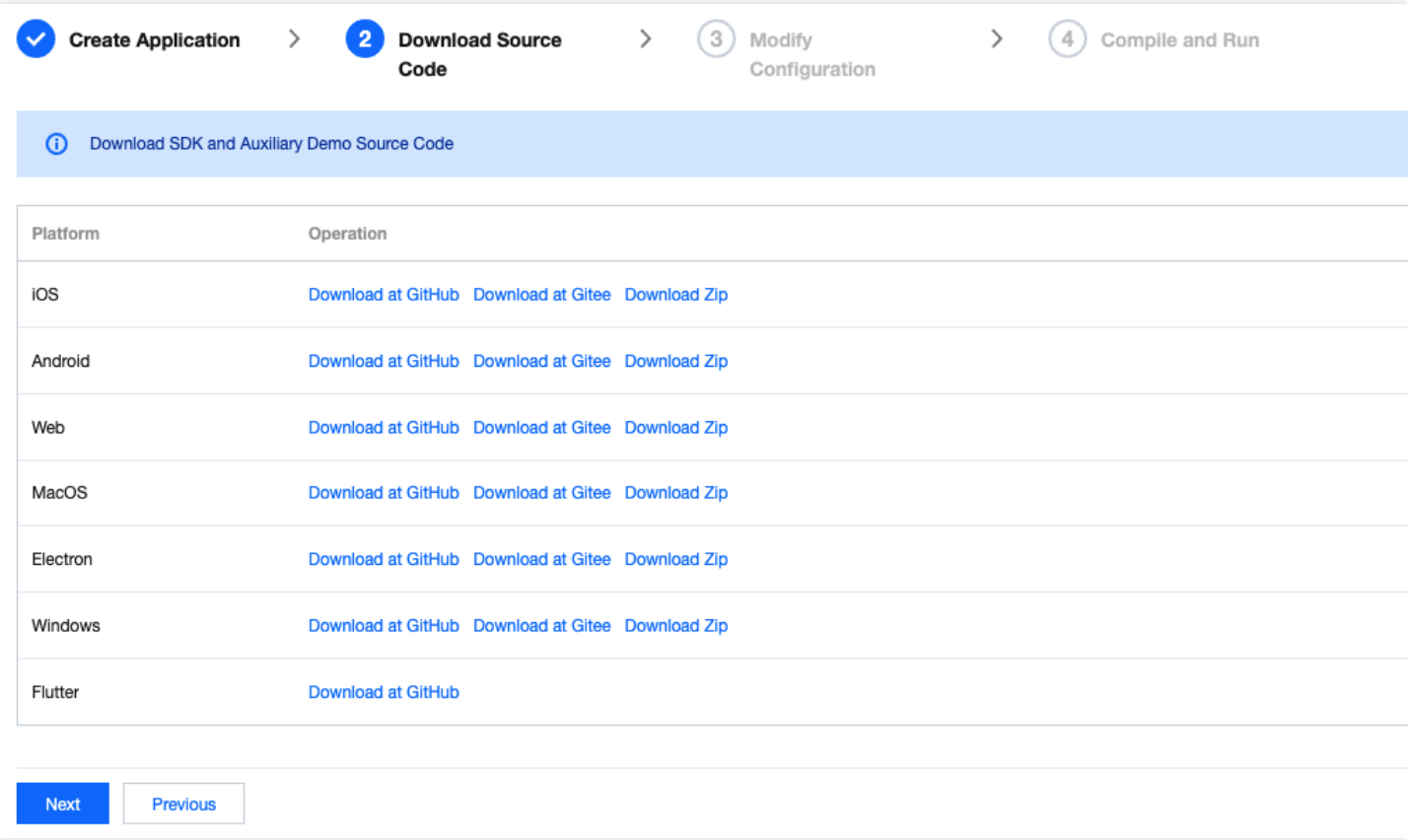


注意：
为方便您快速实现音视频通话功能，我们对 TUICalling 组件进行了改造，通话 UI 在 TUICalling 组件内部实现，您可以无需关注 UI。

运行并体验 App

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助>快速跑通Demo**。
2. 输入应用名称，例如 `TestVideoCall` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

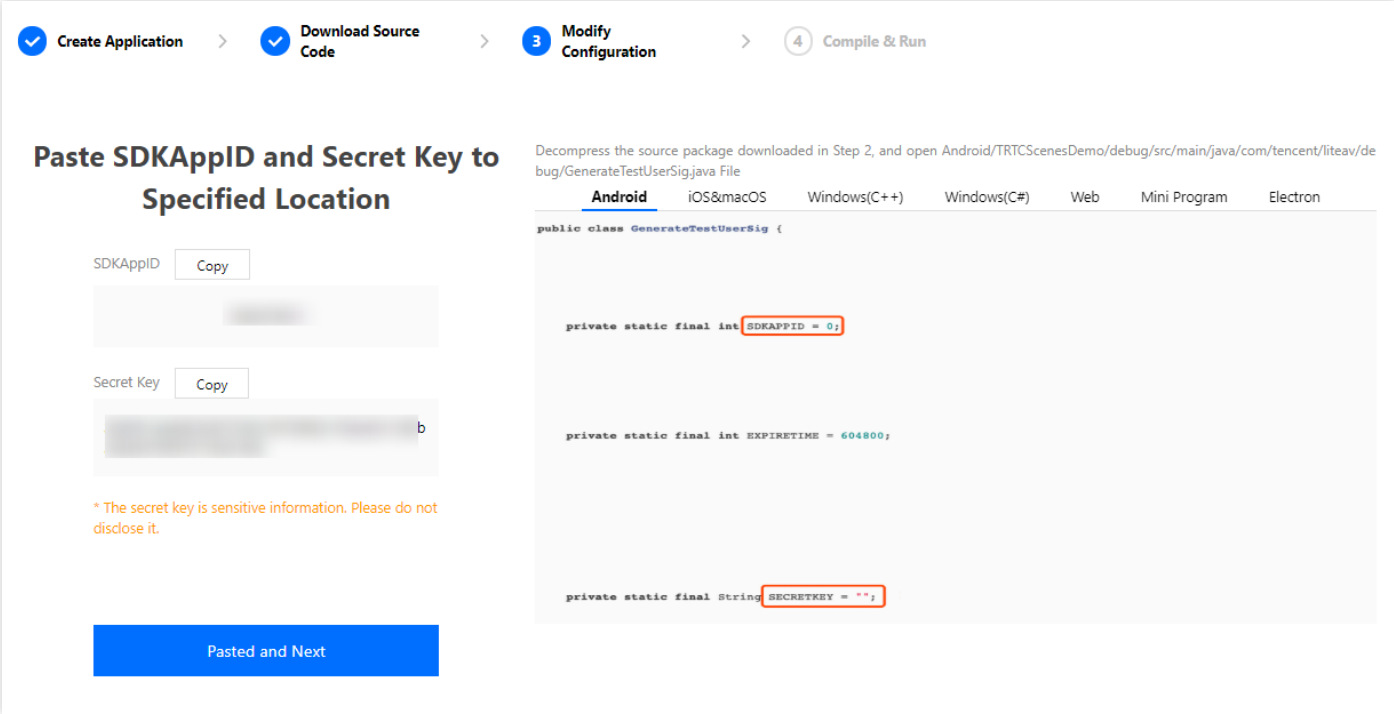
单击进入 TUICalling，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `iOS/Example/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

Xcode（11.0及以上的版本）打开源码工程 `TUICalling/Example/TUICallingApp.xcworkspace`，单击 **运行** 即可开始调试本 App。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

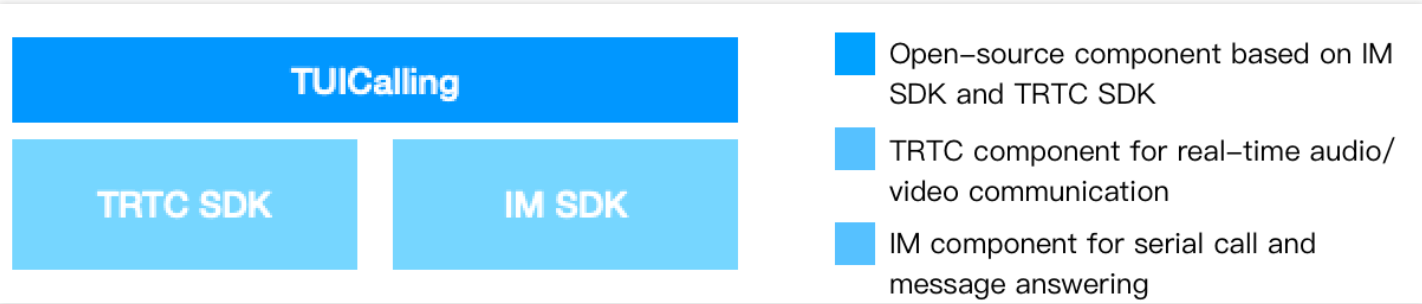
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入要拨打的 userId，单击搜索。
3. 单击 **呼叫**，选择拨打 **视频通话** （**请确保被叫方保持在应用内，否则可能会拨打失败**）。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 进入主页，等待接听来电。

具体接入流程

源码 文件夹 `Source` 中包含三个子文件夹 ui、model 和 Service，其中 Service 文件夹中包含了我们对外暴露的开源组件 TUICallingManager，您可以在 `TUICallingManager.h` 文件中看到该组件提供的接口函数。



您直接使用开源组件 TUICalling 的TUICallingManager即可轻松实现音视频通话功能，而无需再自己实现复杂的通话UI界面和逻辑。

步骤1：集成 SDK

通话组件 TRTCCalling 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 info.plist 文件中需要添加 `Privacy - Camera Usage Description`， `Privacy - Microphone Usage Description` 申请摄像头和麦克风权限。

步骤3：导入 TUICalling 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 将工程目录下的 `Source`、 `Resources`、 `TXAppBasic` 文件夹， `TUICalling.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
pod 'TXAppBasic', :path =&gt; "../TXAppBasic/"
pod 'TXLiteAVSDK_TRTC'
pod 'TUICalling', :path =&gt; "../", :subspecs =&gt; ["TRTC"]
```

步骤4：初始化并登录组件

1. 调用 `TUICallingManager.sharedInstance()` 进行组件初始化。
2. 调用 `TUILogin.initWithSdkAppID(SDKAPPID)` 进行登录初始化。
3. 调用 `TUILogin.login(userId, userSig)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
sdkAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。

userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。

```
// 组件初始化
TUICallingManager.sharedInstance();
// 登录
TUILogin.initWithSdkAppID(SDKAPPID)
TUILogin.login(userId, userSig) {
print("login success")
} fail: { code, errorDes in
print("login failed, code:¥(code), error: ¥(errorDes ?? "nil")")
}
```

步骤5：实现音视频通话

1. 发起方：调用 TUICallingManager 的 `call();` 方法发起通话的请求，并传入用户 ID数组（userIDs）和通话类型（type），通话类型参数传入 `.audio` （音频通话）或者 `.video` （视频通话）。如果用户 ID数组（userIDs）只有1个 userID 时视为单人通话，如果用户 ID 数组（userIDs）有多个 userID 时（>=2）视为多人通话。
2. 接收方：当接收方处于已登录状态时，会自动启动相应的界面。如果希望接收方在不处于登录状态时也能收到通话请求，请参见 离线接听。

```
// 1. 注册监听器
TUICallingManager.shareInstance().setCallingListener(listener: TUICallingListerner())

// 2. 设置是否自定义页面（默认关）
TUICallingManager.shareInstance().enableCustomViewRoute(enable: true)

// 3. 实现监听器回调方法
public func shouldShowOnCallView() -&gt; Bool {
return true;
}

public func callStart(userIDs: [String], type: TUICallingType, role: TUICallingRole, viewControll
er: UIViewController?) { if let vc = viewController {
callingVC = vc;
vc.modalPresentationStyle = .fullScreen

if var topController = UIApplication.shared.keyWindow?.rootViewController {
while let presentedViewController = topController.presentedViewController {
topController = presentedViewController
}

if let navigationVC = topController as? UINavigationController {
```

```
if navigationVC.viewControllers.contains(self) {
present(vc, animated: false, completion: nil)
} else {
navigationVC.popToRootViewController(animated: false)
navigationVC.pushViewController(self, animated: false)
navigationVC.present(vc, animated: false, completion: nil)
}
} else {
topController.present(vc, animated: false, completion: nil)
}
}
}
}

public func callEnd(userIDs: [String], type: TUICallingType, role: TUICallingRole, totalTime: Flo
at) {
callingVC.dismiss(animated: true, completion: nil)
}

public func onCallEvent(event: TUICallingEvent, type: TUICallingType, role: TUICallingRole, messa
ge: String) {

}
// 4.拨打电话
TUICallingManager.shareInstance().call(userIDs, .video)
```

步骤6：实现离线接听

> 说明：
> 如果您的业务定位是在线客服等不需要离线接听功能的场景，那么完成上述 步骤1 - 步骤5 的对接即可。但如果您的业务定位是社交场景，建议实现离线接听。

IM SDK 支持离线推送，您需要进行相应的设置才能达到可用标准。

1. 申请 Apple 推送证书，具体操作请参见 Apple 推送证书申请。
2. 在后台及客户端配置离线推送，具体操作请参见 离线推送（iOS）。
3. 目前在 TRTCCallingImpl 的 sendModel 信令发送函数中已经集成了离线发送的函数，当配置好 App 的离线推送后，消息就可实现离线推送。

组件 API 列表

TUICalling 组件的 API 接口列表如下：

接口函数	接口功能
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器
setCallingBell	设置铃声(建议在30s以内)
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图， 开启后，会在呼叫/被叫开始回调中，接收到CallingView的实例，由开发者自行决定展示方式。注意：必须全屏或者与屏幕等比例展示，否则会有展示异常

版权所有：腾讯云计算（北京）有限责任公司

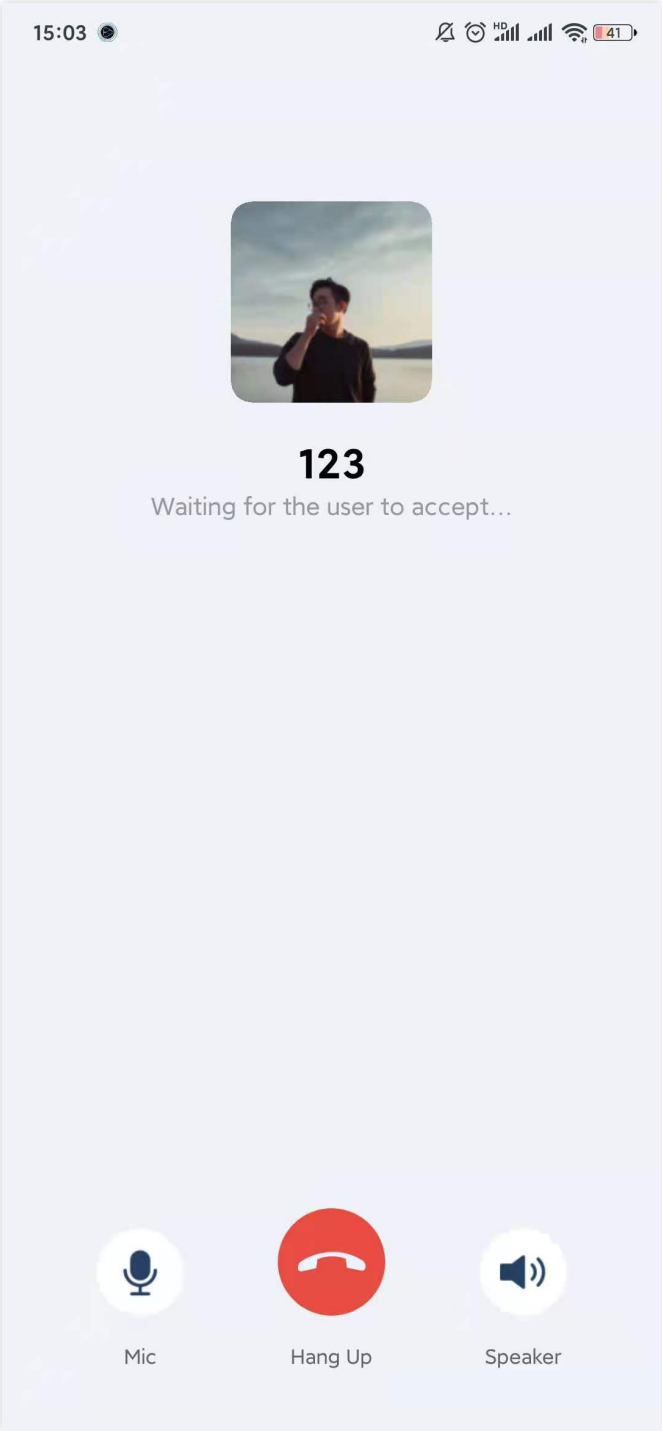
语音通话(Android)

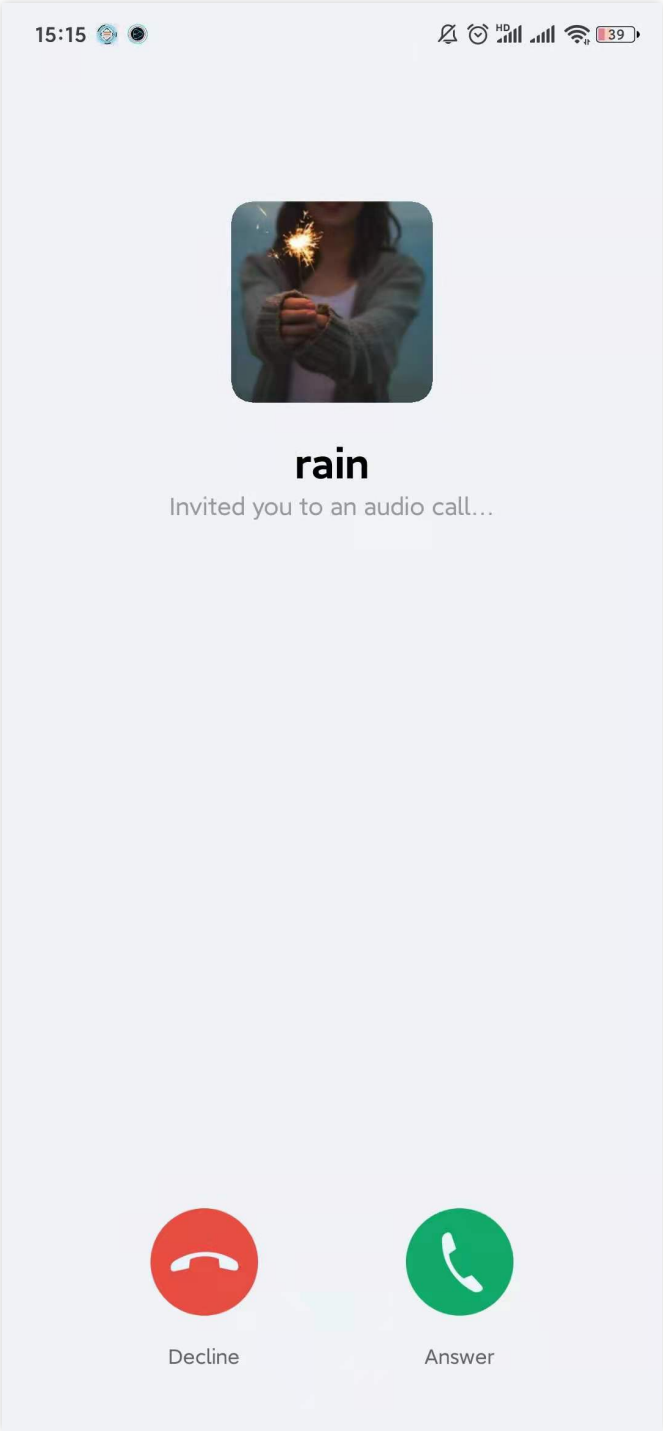
最近更新时间：2022-03-10 11:17:09

效果展示

您可以 下载 安装我们的 App 体验实时音视频通话的效果。

主动呼叫	被叫接听



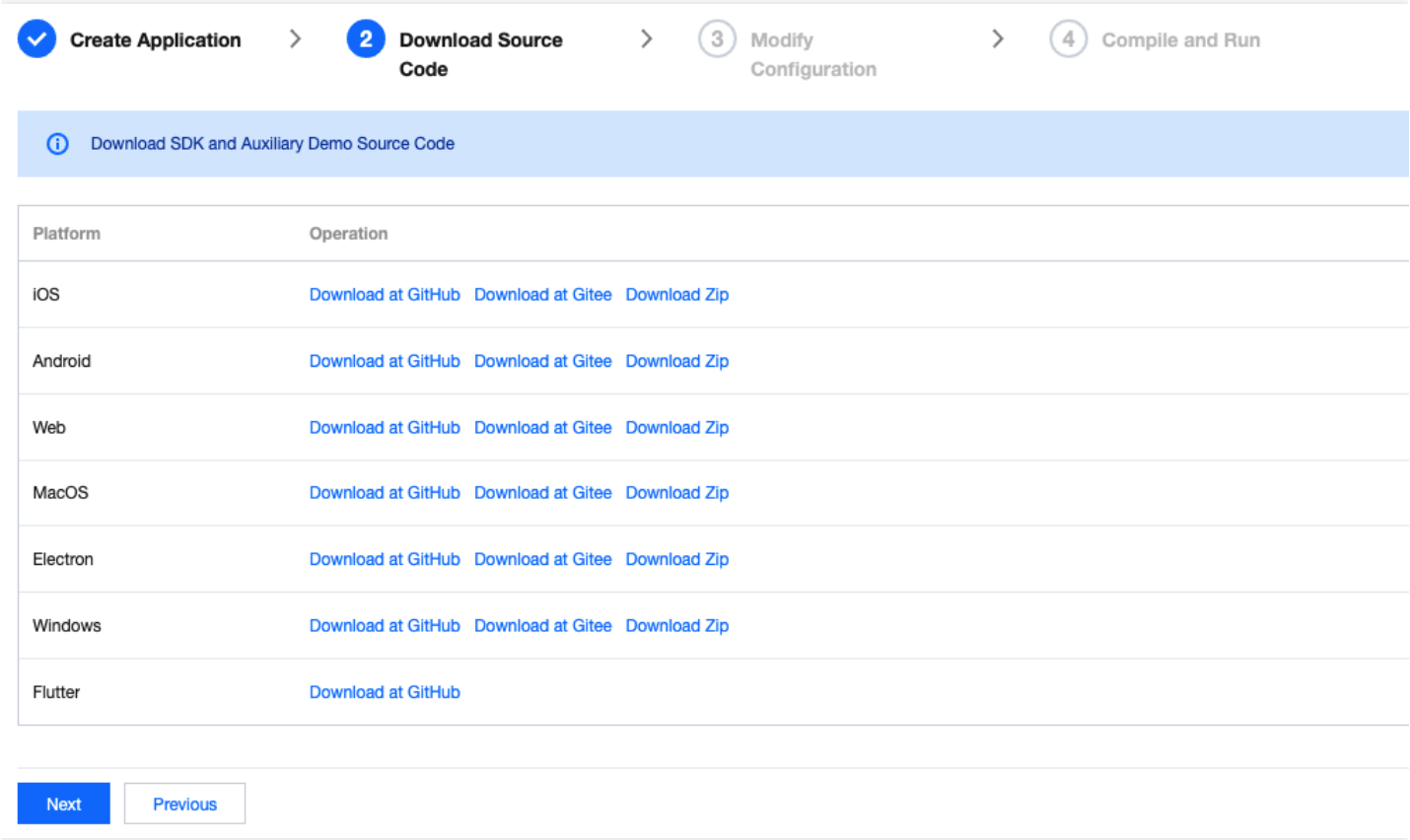


注意：
为方便您快速实现音视频通话功能，我们对 TUICalling 组件进行了改造，通话UI在TUICalling 组件内部实现，您可以无需关注UI。

运行并体验 App

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助>快速跑通Demo**。
2. 输入应用名称，例如 `TestVideoCall` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

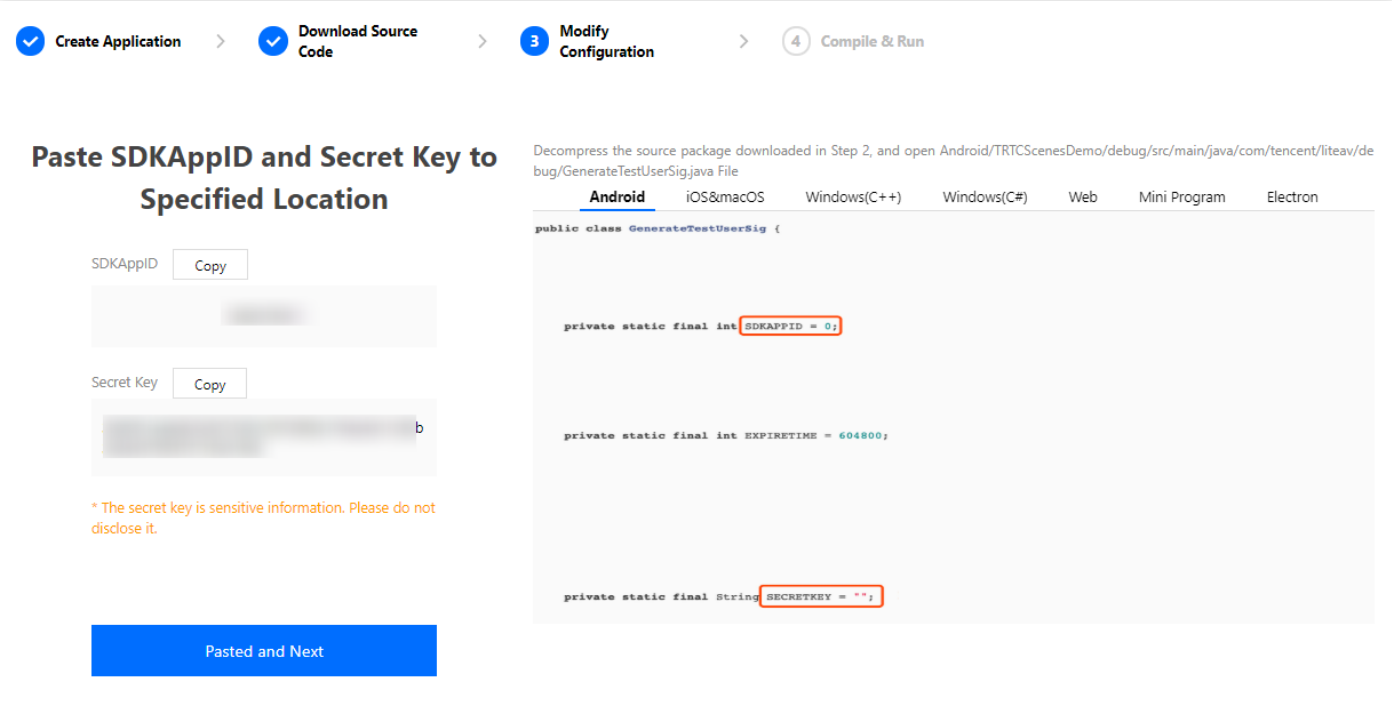
单击进入 TUICalling，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5 以上的版本）打开源码工程 `TUICalling`，单击 **运行** 即可开始调试本 App。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

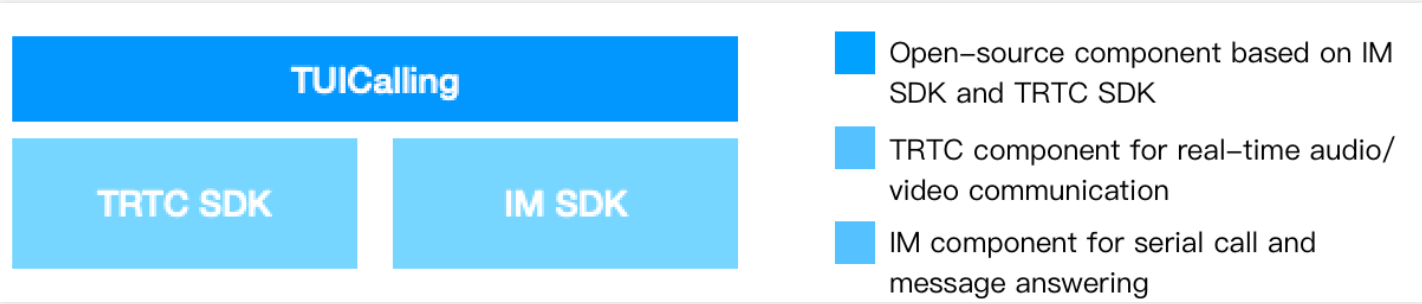
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入要拨打的 userId，单击搜索。
3. 单击 **呼叫**，选择拨打 **视频通话** （**请确保被叫方保持在应用内，否则可能会拨打失败**）。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 进入主页，等待接听来电。

具体接入流程

源码 文件夹 `Source` 中包含两个子文件夹 ui 和 model，其中 model 文件夹中包含了我们对外暴露的开源组件 TUICallingManager，您可以在 `TUICalling.java` 文件中看到该组件提供的接口函数。



您直接使用开源组件 TUICalling 的 TUICallingManager 即可轻松实现音视频通话功能，而无需再自己实现复杂的通话 UI 界面和逻辑。

步骤1：集成 SDK

音视频通话组件 TUICalling 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 mavenCentral 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
compile "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
compile 'com.tencent.imsdk:imsdk:latest.release'

// 由于我们使用到了 gson 解析，所以还需要依赖 google 的 Gson
compile 'com.google.code.gson:gson:latest.release'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击 **Sync Now** 同步 SDK。

> 说明：
> 若您的网络连接 mavenCentral 没有问题，SDK 会自动下载集成到工程里。

方法二：通过本地 AAR 依赖

如果您的开发环境访问 mavenCentral 仓库较慢，可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：

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

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

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

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

步骤3：导入 TUICalling 组件

拷贝 Source 目录到您的工程中，并在 `setting.gradle` 中完成导入，参考如下：

```
include ':Source'
```

步骤4：初始化并登录组件

1. 调用 `TUICallingManager.sharedInstance()` 进行组件初始化。
2. 调用 `TUILogin.init(context, SDKAppID, config, listener)` 进行登录初始化。
3. 调用 `TUILogin.login(userId, userSig, callback)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
SDKAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。
config	SDK配置。用于设置日志级别和日志回调（也可传null），详情可参考下方示例代码。
listener	IM监听器。用于接收一些必要的系统回调通知，比如：被踢下线、userSig过期等，详情可参考下方示例代码。
callback	登录结果监听器。通知登录是否成功，详情可参考下方示例代码。

```
// 组件初始化
TUICallingManager manager = TUICallingManager.sharedInstance();
```

```
// 登录
V2TIMSDKConfig config = new V2TIMSDKConfig();
config.setLogLevel(V2TIMSDKConfig.V2TIM_LOG_DEBUG);
config.setLogListener(new V2TIMLogListener() {
@Override
public void onLog(int logLevel, String logContent) {

}
});
TUILogin.init(this, ${您的SDKAPPID}, config, new V2TIMSDKListener() {
@Override
public void onKickedOffline() { // 登录被踢下线通知
mIsKickedOffline = true;
checkUserStatus();
}
@Override
public void onUserSigExpired() { // suerSig过期通知
mIsUserSigExpired = true;
checkUserStatus();
}
});
TUILogin.login("${您的userId}", "${您的userSig}", new V2TIMCallback() {
@Override
public void onError(int code, String msg) {
Log.d(TAG, "code: " + code + " msg:" + msg);
}
@Override
public void onSuccess() {
Log.d(TAG, "onSuccess");
}
});
```

步骤5：实现音视频通话

1. 发起方：调用 TUICallingManager 的 `call();` 方法发起通话的请求，并传入用户 ID数组（userids）和通话类型（type），通话类型参数传入 `TUICalling.Type.AUDIO` （音频通话）或者 `TUICalling.Type.VIDEO` （视频通话）。如果用户 ID数组（userids）只有1个userId时视为单人通话，如果用户 ID数组（userids）有多个userId时（>=2）视为多人通话。
2. 接收方：当接收方处于已登录状态时，会自动启动相应的界面。如果希望接收方在不处于登录状态时也能收到通话请求，请参见 离线接听。

```
// 1. 初始化组件
TUICallingManager manager = TUICallingManager.sharedInstance();
// 2. 注册监听器
```

```
manager.setCallingListener(new TUICalling.TUICallingListener() {
@Override
public boolean shouldShowOnCallView() {
return true;
}

@Override
public void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, final View
tuiCallingView) {
if (!shouldShowOnCallView() || null == tuiCallingView) {
return;
}
runOnUiThread(new Runnable() {
@Override
public void run() {
FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(FrameLayout.LayoutParams.MATCH_PAR
ENT, FrameLayout.LayoutParams.MATCH_PARENT);
mCallingView = tuiCallingView;
addContentView(tuiCallingView, params);
}
});
}

@Override
public void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTim
e) {
removeView();
}

@Override
public void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, Strin
g message) {
if (TUICalling.Event.CALL_FAILED == event) {
removeView();
}
}
});
// 3.拨打电话
manager.call(userIDs, TUICalling.Type.VIDEO);
```

步骤6：实现离线接听

> 说明：
> 如果您的业务定位是在线客服等不需要离线接听功能的场景，那么完成上述 步骤1 - 步骤5 的对接即可。但如果您的业务定位是社交场景，建议实现离线接听。

IM SDK 支持离线推送，但是 Android 端各个手机厂商均有各自的离线推送服务，因此接入复杂度要高于 iOS 平台，您需要进行相应的设置才能达到可用标准。

1. 申请对应厂商的推送渠道需要的证书等，并将其配置到即时通信 IM 控制台中，按照推送要求增加证书和 ID 等，详细的操作步骤请参见 即时通信 IM > 离线推送（Android） 。
2. 目前在 TRTCCallingImpl 的 sendModel 信令发送函数中已经集成了离线发送的函数，当配置好 App 的离线推送后，消息就可实现离线推送。

组件 API 列表

TUICalling 组件的 API 接口列表如下：

接口函数	接口功能
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器
setCallingBell	设置铃声(建议在30s以内)
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图， 开启后，会在呼叫/被叫开始回调中，接收到 CallingView 的实例，由开发者自行决定展示方式。注意：必须全屏或者与屏幕等比例展示，否则会有展示异常

语音通话(Web)

最近更新时间：2022-04-02 16:27:03

本文介绍如何实现一套可以在浏览器上运行的语音通话解决方案，文章分成两个部分：

- 第一部分：介绍如何开通服务并跑通我们提供的演示 Demo。
- 第二部分：介绍如何使用 TRTCCalling 组件快速搭建自己的语音通话功能。

环境要求

请使用最新版本的 Chrome 浏览器。目前桌面端 Chrome 浏览器支持 TRTC Web SDK 的相关特性比较完整，因此建议使用 Chrome 浏览器进行体验。

TRTCCalling 依赖以下端口进行数据传输，请将其加入防火墙白名单，配置完成后，您可以通过访问并体验 官网 Demo 检查配置是否生效。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名：qcloud.rtc.qq.com
  具体请参见 应对防火墙限制相关。

平台支持

目前该方案支持如下平台：

操作系统	浏览器类型	浏览器最低版本要求
Mac OS	桌面版 Safari 浏览器	11+
Mac OS	桌面版 Chrome 浏览器	56+
Mac OS	桌面版 Firefox 浏览器	56+
Mac OS	桌面版 Edge 浏览器	80+
Windows	桌面版 Chrome 浏览器	56+
Windows	桌面版 QQ 浏览器（极速内核）	10.4+
Windows	桌面版 Firefox 浏览器	56+

操作系统	浏览器类型	浏览器最低版本要求
Windows	桌面版 Edge 浏览器	80+

说明：
详细兼容性查询，具体请参见 浏览器支持情况。同时，您可通过 TRTC 检测页面 在线检测。

URL 域名协议限制

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

跑通测试 Demo

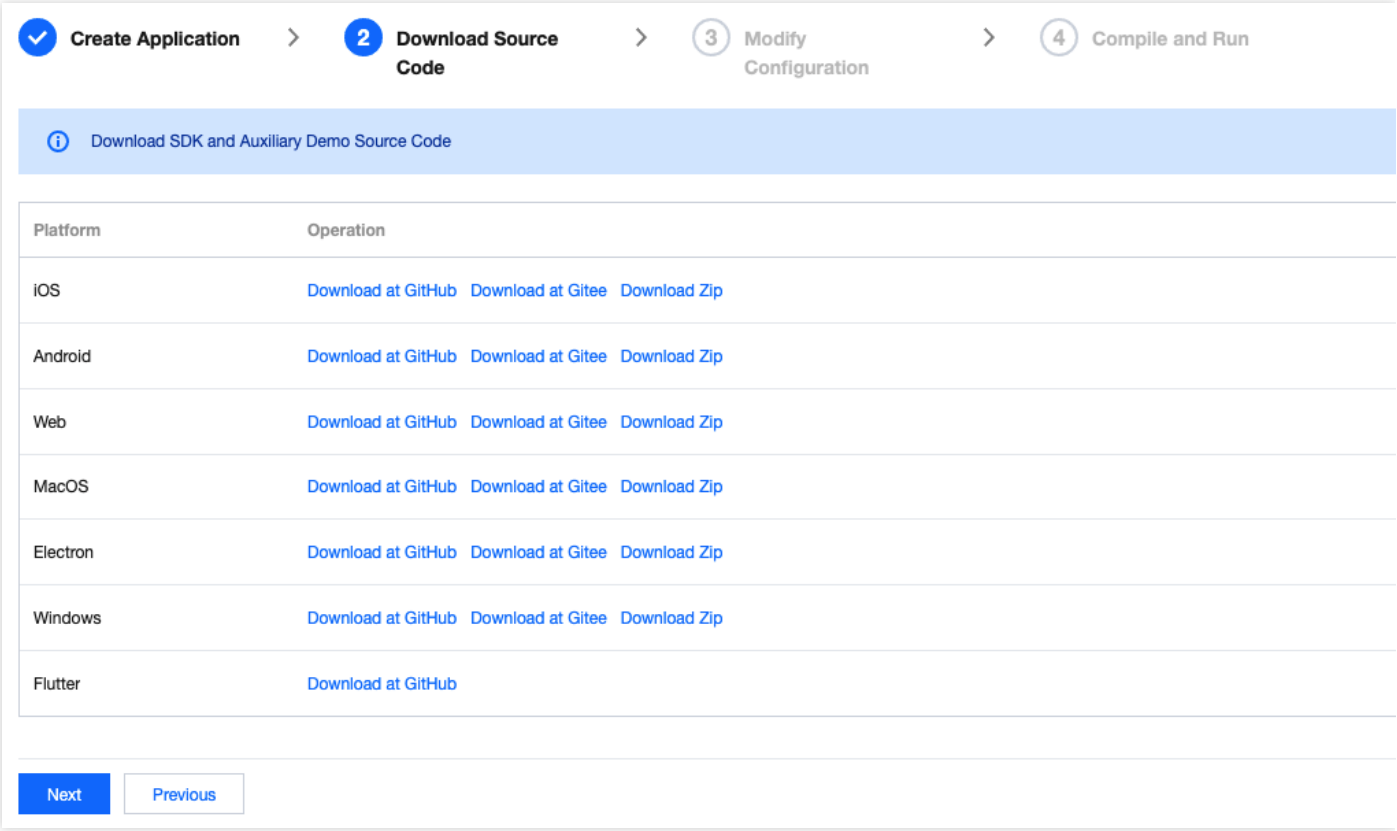
步骤1：创建新的应用

1. 注册腾讯云 账号，并完成 实名认证。
2. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
3. 输入应用名称，例如 `TestTRTC` ，单击 **创建**。

步骤2：下载 Demo

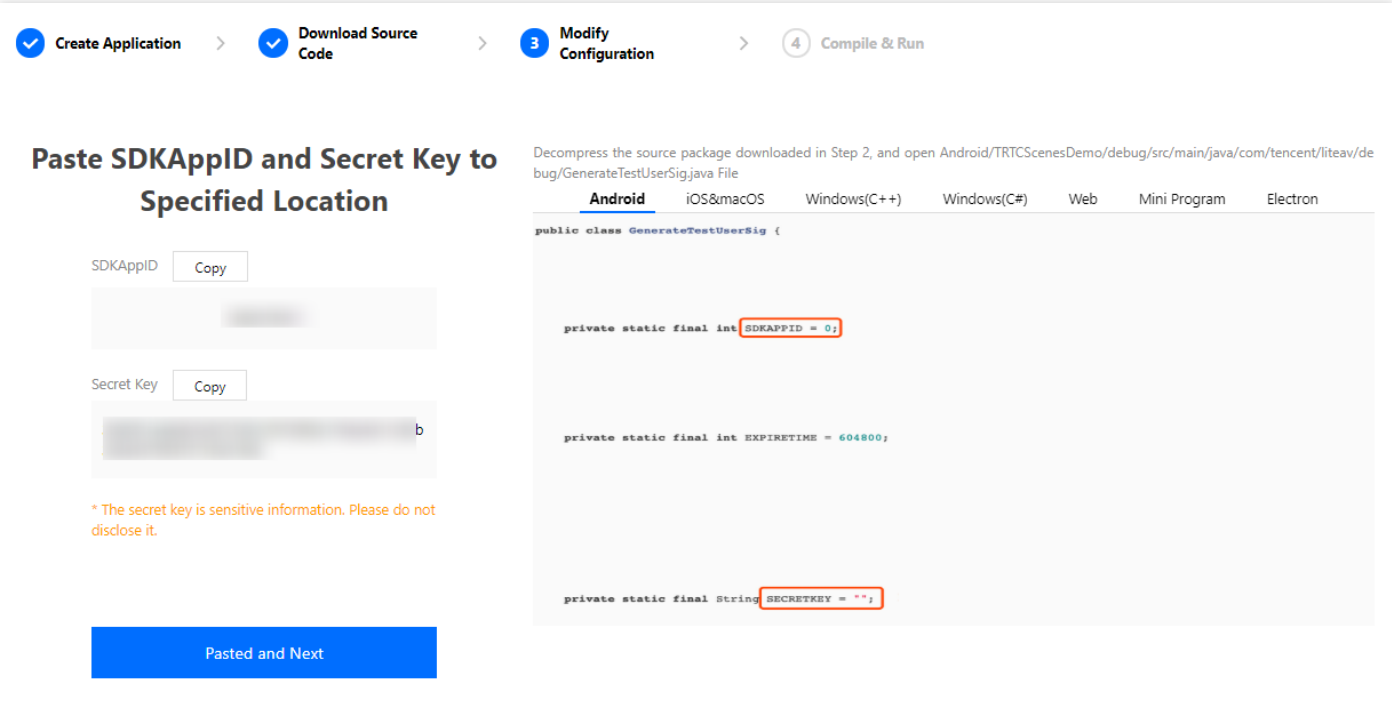
1. 根据实际业务需求下载 Demo 源码或单击进入 TUICalling下载 Demo 源码。

2. 下载完成后，单击 **已下载，下一步**。



步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `Web/public/debug/GenerateTestUserSig.js` 文件。
3. 设置 `GenerateTestUserSig.js` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。
5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

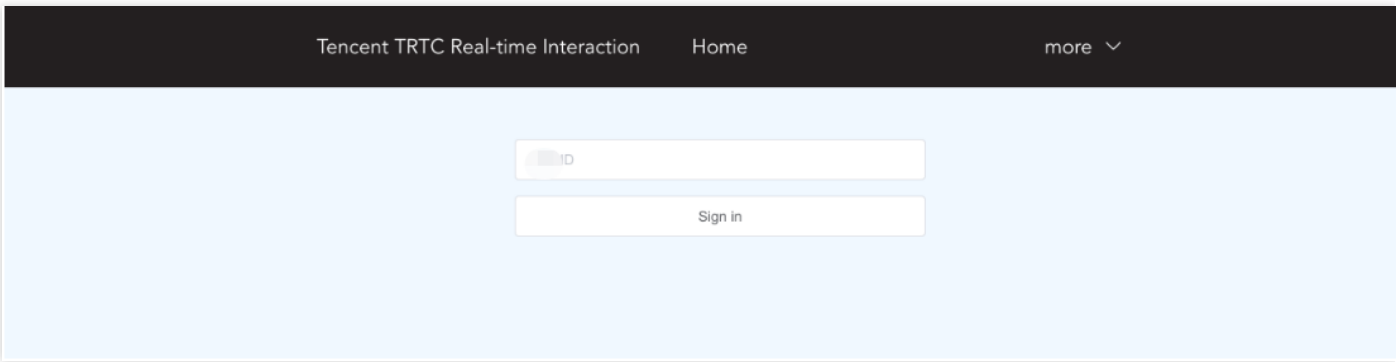
- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 Demo

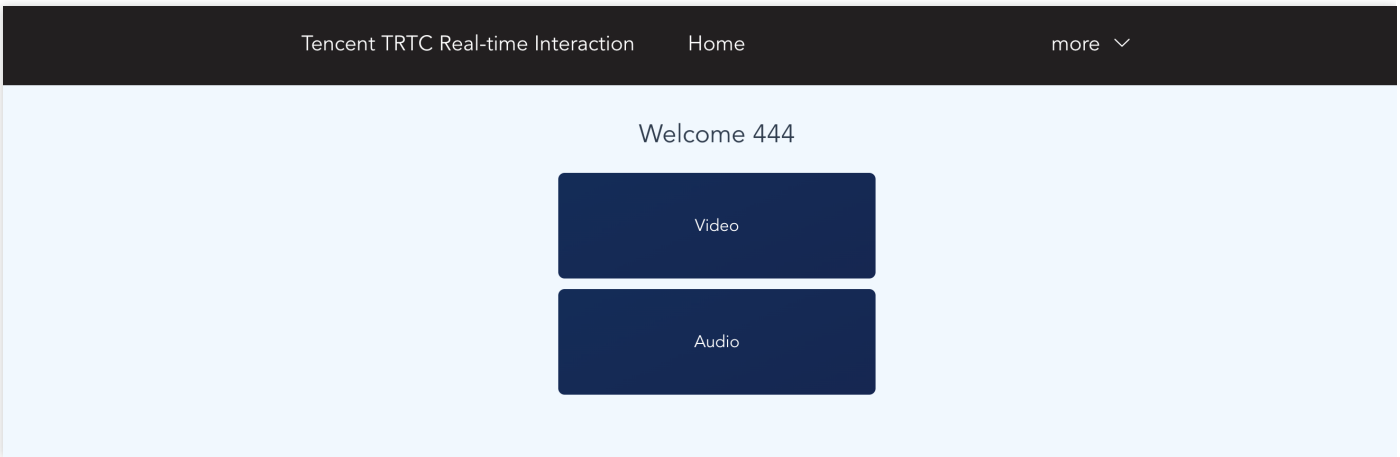
1. 在 Demo 目录下依次输入如下命令：

```
npm install
npm run serve
```

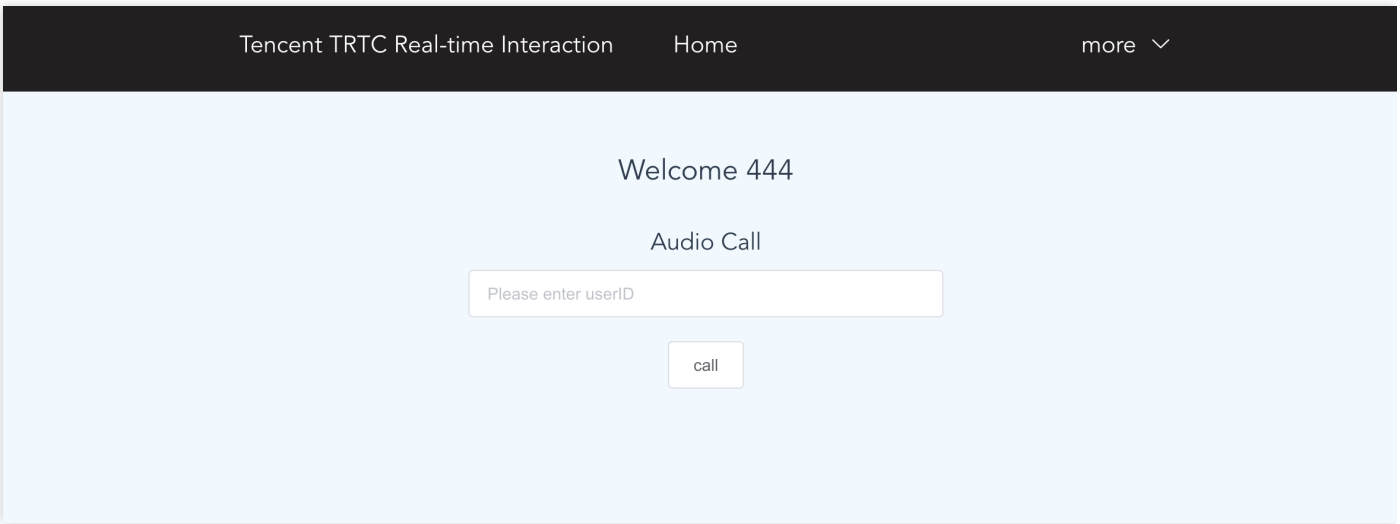
2. 启动 Chrome 浏览器中打开链接 `http://localhost:8080/` ，如果一切正常，Demo 运行界面如图所示：



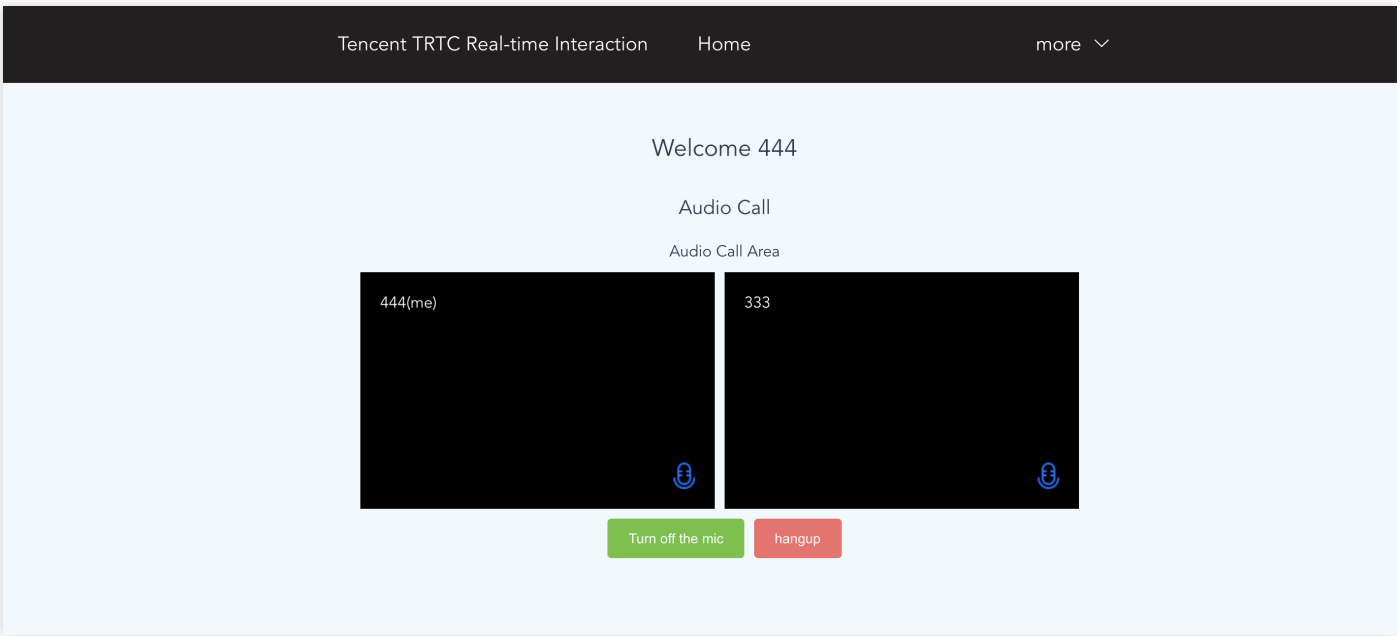
3. 输入用户 userid，单击 **登录**，并选择 **语音通话**：



4. 输入呼叫用户 userid，单击 **呼叫**：



5. 即可进行语音通话：



搭建自己的语音通话

步骤1：集成 TRTCCalling 组件

说明：

- 从v0.6.0起，需要手动安装依赖 trtc-js-sdk 和 tim-js-sdk 以及 tsignaling。
- 为了减小 trtc-calling-js.js 的体积，避免和接入侧已使用的 trtc-js-sdk 和 tim-js-sdk 以及 tsignaling 发生版本冲突，trtc-calling-js.js 将 trtc-js-sdk，tim-js-sdk，tsignaling 打包为外部依赖，在使用前您需要手动安装。

```
// npm 方式安装
npm install trtc-js-sdk --save

npm install tim-js-sdk --save

npm install tsignaling --save

npm install trtc-calling-js --save
```

```
// 如果您需要通过 script 方式使用 trtc-calling-js，需要按顺序引入以下资源

<script src="./trtc.js"></script>
<script src="./tim-js.js"></script>
<script src="./tsignaling.js"></script>
<script src="./trtc-calling-js.js"></script>
```

步骤2：创建 TRTCCalling 对象

创建 TRTCCalling 对象，并将 SDKAppID 参数设置为您自己的 SDKAppID。

```
import TRTCCalling from 'trtc-calling-js';

let options = {
SDKAppID: 0, // 接入时需要将0替换为您的 SDKAppID
// 从v0.10.2起，新增 tim 参数
// tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性
tim: tim
};
const trtcCalling = new TRTCCalling(options);
```

步骤3：完成登录

```
trtcCalling.login({
userID,
userSig,
});
```

步骤4：实现 1v1 通话

- **主叫方：呼叫某个用户**

```
trtcCalling.call({
userID, //用户 ID
type: 1, //通话类型，0-未知，  1-语音通话，2-视频通话
timeout //邀请超时时间，单位 s(秒)
});
```

- **被叫方：接听新的呼叫**

```
// 接听
trtcCalling.accept();
```

```
//拒绝
trtcCalling.reject()
```

- **挂断**

```
trtcCalling.hangup()
```

常见问题

为什么拨打不通，或者被踢下线？

组件暂不支持多实例登入，不支持**离线推送信令**功能，请您确认登入账号的唯一性。

> 说明：
>
> - **多实例**：一个 UserID 重复登入，或在不同端登入，将会引起信令的混乱。
> - **离线推送**：实例在线才能接收消息，实例离线时接收到的信令不会在上线后重新推送。
>
> 更多常见问题，请参见 TRTCCalling Web 相关问题。

技术咨询

了解更多详情您可以联系我们或发送邮件至colleenyu@tencent.com。

参考文档

- TRTCCalling web 官网体验
- TRTCCalling npm
- TRTCCalling web demo 源码
- TRTCCalling web API
- TRTCCalling web 相关问题

语音通话(Flutter)

最近更新时间：2022-02-21 17:24:02

如需快速实现语音通话功能，您可以直接基于我们提供的 Demo 进行修改适配，也可以使用我们提供的 TRTCCalling 组件并实现自定义 UI 界面。

复用 Demo 的 UI 界面

步骤1：创建新的应用

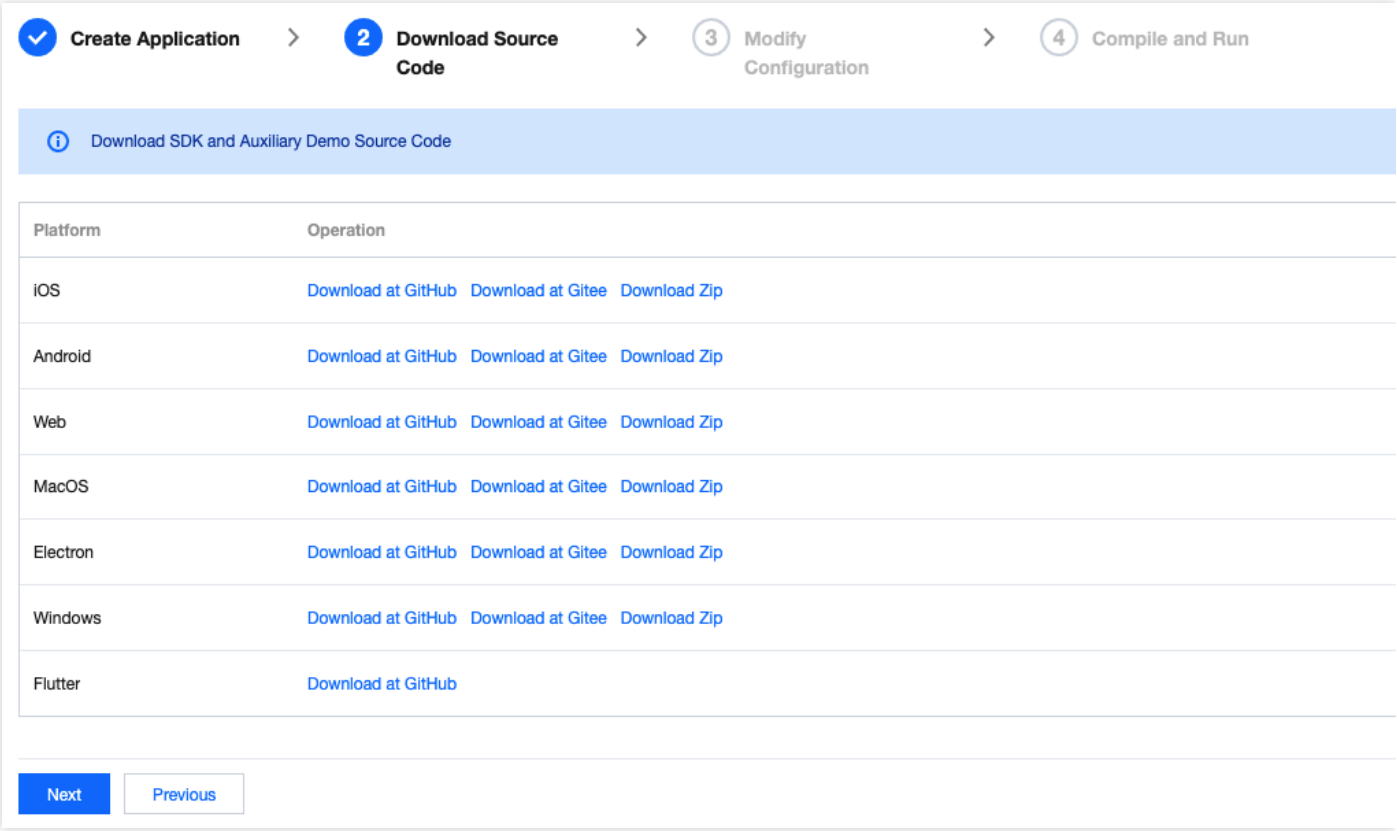
1. 登录实时音视频控制台，选择**开发辅助**> 快速跑通Demo。
2. 输入应用名称，例如 `TestAudioCall` ，单击**创建**。

> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 SDK 和 Demo 源码

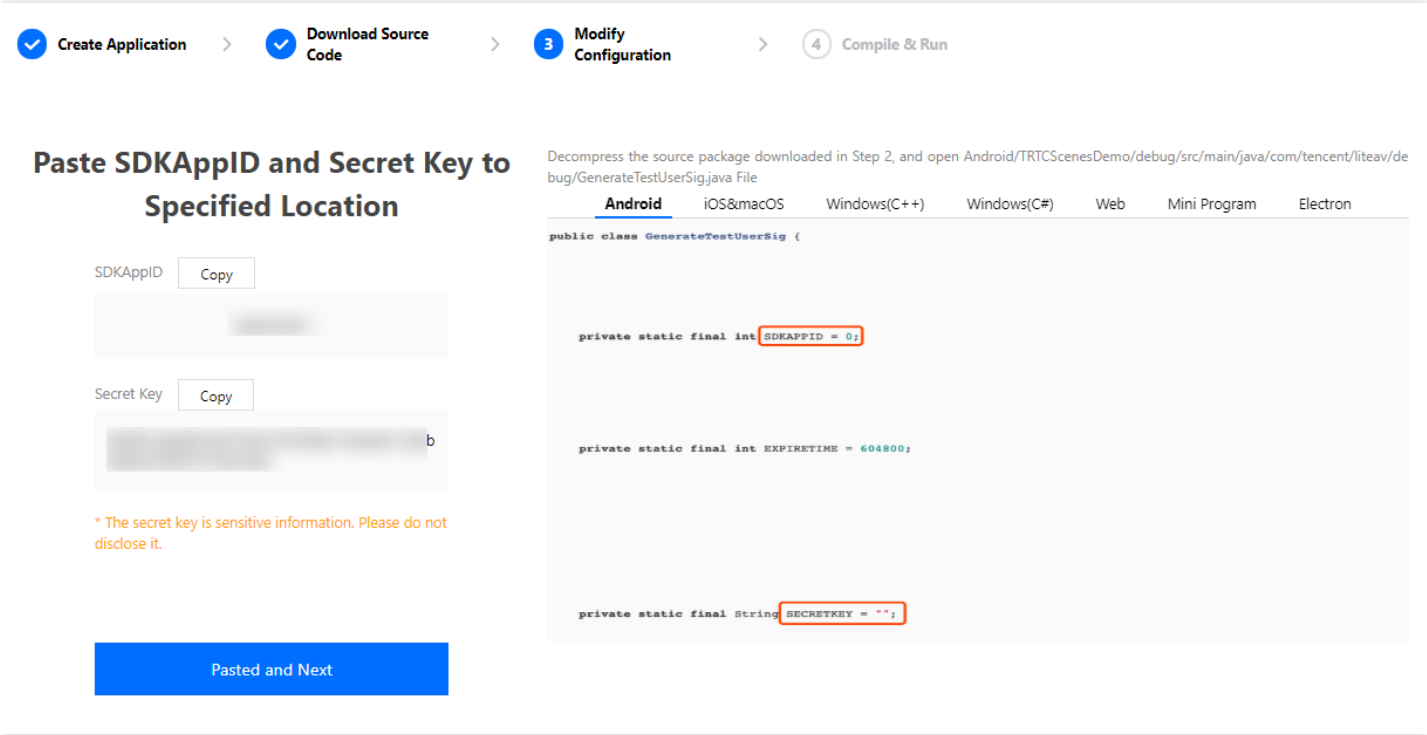
1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击**已下载，下一步**。



步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `/example/lib/debug/GenerateTestUserSig.dart` 文件。
3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：
   - SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为 PLACEHOLDER，请设置为实际的密钥信息。



4. 粘贴完成后，单击**已复制粘贴，下一步**即创建成功。
5. 编译完成后，单击**回到控制台概览**即可。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 Demo

> 注意：
> Android 端需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。
2. 编译运行调试：

- Android 端
- iOS 端

i. 执行 `flutter run` 。

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

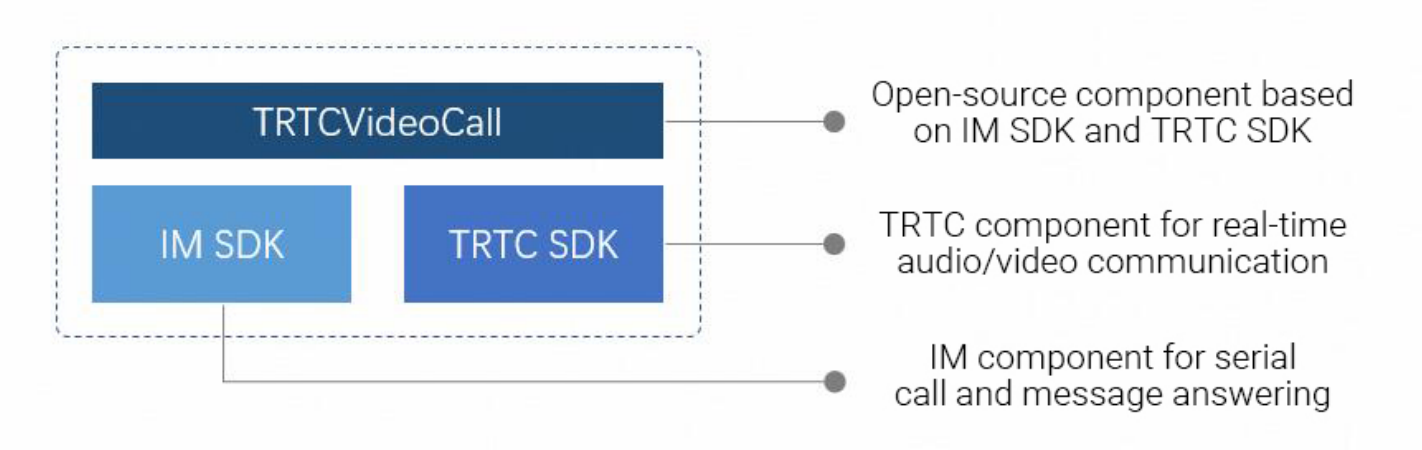
步骤5：修改 Demo 源代码

源码 文件夹 `TRTCCallingDemo` 中包含两个子文件夹 ui 和 model，其中 ui 文件夹中均为界面代码：

文件或文件夹	功能描述
TRTCCallingVideo.dart	展示音视频通话的主界面，通话的接听和拒绝就是在这个界面中完成的
TRTCCallingContact.dart	用于展示选择联系人的界面，可以通过此界面搜索已注册用户，发起通话

实现自定义 UI 界面

源码 文件夹 `TRTCCallingDemo` 中包含两个子文件夹 ui 和 model，其中 model 文件夹中包含了我们实现的可重用开源组件 TRTCCalling，您可以在 `TRTCCalling.dart` 文件中看到该组件提供的接口函数。



您可以使用开源组件 TRTCCalling 实现自己的 UI 界面，即只复用 model 部分，自行实现 UI 部分。

步骤1：集成 SDK

音视频通话组件 TRTCCalling 依赖 TRTC SDK 和 IM SDK，您可以通过配置 `pubspec.yaml` 自动下载更新。

在项目的 `pubspec.yaml` 中写如下依赖：

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

步骤2：配置权限及混淆规则

- iOS 端
- Android 端

需要在 `Info.plist` 中加入对相机和麦克风的权限申请：

```
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

步骤3：导入 TRTCCalling 组件

拷贝 model 目录中的所有文件到您的项目中：

```
/lib/TRTCCallingDemo/model
```

步骤4：初始化并登录组件

1. 调用 `TRTCCalling.sharedInstance()` 获取组件实例。
2. 调用 `login(SDKAppID, userId, userSig)` 完成组件的登录，其中几个关键参数的填写请参考下表：

参数名	作用
SDKAppID	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，计算方式请参考 如何计算及使用 UserSig。

```
// 初始化
sCall = await TRTCCalling.sharedInstance();
sCall.login(1400000123, "userA", "xxxx");
```

步骤5：实现 1v1 语音通话

1. 发起方：调用 `TRTCCalling` 的 `call()` 方法发起通话的请求，并传入用户 ID（userid）和通话类型（type），通话类型参数传入 `TRTCCalling.typeAudioCall` 。
2. 接收方：当接收方处于已登录状态时，会收到名为 `onInvited()` 的事件通知，回调中 `callType` 的参数是发起方填写的通话类型，您可以通过此参数启动相应的界面。

3. 接收方：如果希望接听电话，接收方可以调用 `accept()` 函数，并同时调用 `openCamera()` 函数打开自己本地的摄像头。接收方也可以调用 `reject()` 拒绝此次通话。
4. 当双方的音视频通道建立完成后，通话的双方都会接收到名为 `onUserVideoAvailable()` 的事件通知，表示对方的视频画面已经拿到。此时双方用户均可以调用 `startRemoteView()` 展示远端的视频画面。远端的声音默认是自动播放的。

组件 API 列表

TRTCCalling 组件的 API 接口列表如下：

接口函数	接口功能
registerListener	增加 TRTCCalling 监听器，用户可以通过该监听器获取状态通知
unRegisterListener	移除监听器
destroy	销毁实例
login	登录 IM，所有功能需要先进行登录后才能使用
logout	登出 IM，登出后无法再进行拨打操作
call	C2C 邀请通话，被邀请方会收到 onInvited 的事件通知
accept	作为被邀请方接听来电
reject	作为被邀请方拒绝来电
hangup	结束通话
setMicMute	是否静音 mic
setHandsFree	是否开启免提

TUICalling API（iOS）

最近更新时间：2021-11-12 16:32:40

TUICalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频通话。TUICalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时语音通话（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TUICalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例。
call	C2C 邀请通话。
receiveAPNSCalled	作为被邀请方接听来电。
setCallingListener	设置监听器。
setCallingBell	设置铃声（建议在30s以内）
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图

TUICallingListener API 概览

事件回调

API	描述
shouldShowOnCallView	被叫时请求拉起接听页面
onCallStart	呼叫开始回调。主叫、被叫均会触发
onCallEnd	通话回调。主叫、被叫均会触发
onCallEvent	通话事件回调

Type API 概览

通话类型

enum	描述
TUICallingTypeAudio	音频通话
TUICallingTypeVideo	视频通话

Role API 概览

用户角色类型

enum	描述
TUICallingRoleCall	通话发起方（主叫）
TTUICallingRoleCalled	通话接听方（被叫）

Event API 概览

事件类型

enum	描述
TUICallingEventCallStart	通话开始
TUICallingEventCallSucceed	通话接通成功
TUICallingEventCallEnd	通话结束
TUICallingEventCallFailed	通话失败

SDK 基础函数

sharedInstance

sharedInstance 是 TUICallingManager 的组件单例。

```
+ (instancetype)shareInstance;
```

call

C2C 邀请通话。

```
- (void)call:(NSArray<NSString *> *)userIDs type:(TUICallingType)type;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频

receiveAPNSCalled

作为被邀请方接听来电。

```
- (void)receiveAPNSCalled:(NSArray<NSString *> *)userIDs type:(TUICallingType)type;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频

setCallingListener

设置监听器。

```
- (void)setCallingListener:(id<TUICallingListerner>)listener;
```

参数如下表所示：

参数	类型	含义
listener	TUICallingListener	TUIcalling 组件监听器

setCallingBell

设置铃声（建议在30s以内）。

```
- (void)setCallingBell:(NSString *)filePath;
```

参数如下表所示：

参数	类型	含义
filePath	NSString	铃音资源路径

enableMuteMode

开启静音模式。

```
- (void)enableMuteMode:(BOOL)enable;
```

参数如下表所示：

参数	类型	含义
enable	BOOL	是否开启静音模式

enableCustomViewRoute

开启自定义视图。
开启后，会在呼叫/被叫开始回调中，接收到 CallingViewController 的实例，由开发者自行决定展示方式。

> 注意：
> 必须全屏展示，否则会有展示异常。

```
- (void)enableCustomViewRoute:(BOOL)enable;
```

参数如下表所示：

参数	类型	含义
enable	BOOL	是否开启自定义视图

TUICallingListener 回调函数

shouldShowOnCallView

是否同意被叫时请求拉起接听页面。

```
- (BOOL)shouldShowOnCallView;
```

参数如下表所示：

参数	类型	含义
返回值	BOOL	是否同意

onCallStart

呼叫开始回调。主叫、被叫均会触发。

```
- (void)callStart:(NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)r
ole viewController:(UIViewController * _Nullable)viewController;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫
viewController	UIViewController	通话视图 ViewController

onCallEnd

通话结束回调。主叫、被叫均会触发。enableCustomViewRoute 设置为 NO 时，此回调方法不会触发。

```
- (void)callEnd:(NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)rol
e totalTime:(CGFloat)totalTime;
```

参数如下表所示：

参数	类型	含义
userIDs	NSArray	通话用户 ID 列表
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫

参数	类型	含义
totalTime	CGFloat	通话时长，单位：秒

onCallEvent

通话事件回调。enableCustomViewRoute 设置为 NO 时，此回调方法不会触发。

```
- (void)onCallEvent:(TUICallingEvent)event type:(TUICallingType)type role:(TUICallingRole)role me
ssage:(NSString *)message;
```

参数如下表所示：

参数	类型	含义
event	TUICallingEvent	通话事件类型。
type	TUICallingType	通话类型：音频/视频
role	TUICallingRole	用户角色类型：主叫/被叫
message	NSString	事件的描述信息

版权所有：腾讯云计算（北京）有限责任公司

TUICalling API（Android）

最近更新时间：2021-11-12 16:33:03

TUICalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频通话。TUICalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时语音通话（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TUICalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例
call	C2C 邀请通话
receiveAPNSCalled	作为被邀请方接听来电
setCallingListener	设置监听器。
setCallingBell	设置铃声（建议在30s以内）
enableMuteMode	开启静音模式
enableCustomViewRoute	开启自定义视图

TUICallingListener API 概览

事件回调

API	描述
shouldShowOnCallView	被叫时请求拉起接听页面
onCallStart	呼叫开始回调。主叫、被叫均会触发
onCallEnd	通话回调。主叫、被叫均会触发

API	描述
onCallEvent	通话事件回调

Type API 概览

通话类型

enum	描述
AUDIO	音频通话
VIDEO	视频通话

Role API 概览

用户角色类型

enum	描述
CALL	通话发起方（主叫）
CALLED	通话接听方（被叫）

Event API 概览

事件类型

enum	描述
CALL_START	通话开始
CALL_SUCCEED	通话接通成功
CALL_END	通话结束
CALL_FAILED	通话失败

SDK 基础函数

sharedInstance

sharedInstance 是 TUICalling 的组件单例。

```
public static TUICallingManager sharedInstance();
```

call

C2C 邀请通话。

```
void call(String[] userIDs, Type type);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表
type	TUICalling.Type	通话类型：音频/视频

receiveAPNSCalled

作为被邀请方接听来电。

```
void receiveAPNSCalled(String[] userIDs, Type type);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表
type	TUICalling.Type	通话类型：音频/视频

setCallingListener

设置监听器。

```
void setCallingListener(TUICallingListener listener);
```

参数如下表所示：

参数	类型	含义
listener	TUICallingListener	TUIcalling 组件监听器

版权所有：腾讯云计算（北京）有限责任公司

setCallingBell

设置铃声(建议在30s以内)。

```
void setCallingBell(String filePath);
```

参数如下表所示：

参数	类型	含义
filePath	String	铃音资源路径

enableMuteMode

开启静音模式。

```
void enableMuteMode(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	是否开启静音模式

enableCustomViewRoute

开启自定义视图。
开启后，会在呼叫/被叫开始回调中，接收到 CallingView 的实例，由开发者自行决定展示方式。

> 注意：
> 必须全屏或者与屏幕等比例展示，否则会有展示异常。

```
void enableCustomViewRoute(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	是否开启自定义视图

TUICallingListener 回调函数

shouldShowOnCallView

是否同意被叫时请求拉起接听页面。

```
boolean shouldShowOnCallView();
```

参数如下表所示：

参数	类型	含义
返回值	boolean	是否同意

onCallStart

呼叫开始回调。主叫、被叫均会触发。

```
void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, View tuiCallingVie
w);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表。
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
tuiCallingView	View	通话视图 View。enableCustomViewRoute 设置为 false 时，view 为 null

onCallEnd

通话结束回调。主叫、被叫均会触发。

```
void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTime);
```

参数如下表所示：

参数	类型	含义
userIDs	String[]	通话用户 ID 列表

参数	类型	含义
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
totalTime	long	通话时长，单位：秒

onCallEvent

通话事件回调。

```
void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, String messa
ge);
```

参数如下表所示：

参数	类型	含义
event	TUICalling.Event	通话事件类型
type	TUICalling.Type	通话类型：音频/视频
role	TUICalling.Role	用户角色类型：主叫/被叫
message	String	事件的描述信息

版权所有：腾讯云计算（北京）有限责任公司

TRTCCalling API（Web）

最近更新时间：2022-04-02 16:27:03

TRTCCalling 简介

TRTCCalling 组件是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1和多人视频/语音通话。具体的实现过程请参见 实时语音通话（Web）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

下载 Demo

单击进入 TUICalling，根据实际业务需求下载 Demo 源码。

环境要求

请使用最新版本的 Chrome 浏览器。目前桌面端 Chrome 浏览器支持 TRTC Web SDK 的相关特性比较完整，因此建议使用 Chrome 浏览器进行体验。具体请参见 环境要求。

URL 域名协议限制

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

版权所有：腾讯云计算（北京）有限责任公司

TRTCCalling API

事件订阅/取消订阅相关接口函数

本组件基于事件分发进行管理，应用层可以根据组件下发的事件进行交互的改变。

API	描述
on(eventName, callback, context)	订阅事件
off(eventName, callback, context)	取消事件订阅

SDK 基础函数

API	描述
login({userID, userSig})	登录 IM 接口，所有功能需要先进行登录后才能使用
logout()	登出接口，登出后无法再进行拨打操作

通话操作相关接口函数

API	描述
call({userID, type, offlinePushInfo}))	单人通话邀请
groupCall({userIDList, type, groupID, offlinePushInfo})	群聊通话邀请
accept()	接受通话邀请
reject()	拒绝通话邀请
hangup()	挂断当前通话

视频控制相关接口函数

API	描述
startRemoteView({userID, videoViewDomID})	启动远端画面渲染
stopRemoteView({userID})	停止远端画面渲染
startLocalView({userID, videoViewDomID})	启动本地画面渲染
stopLocalView({userID})	停止本地画面渲染

版权所有：腾讯云计算（北京）有限责任公司

API	描述
openCamera()	启动摄像头
closeCamera()	关闭摄像头
setMicMute(isMute)	设备麦克风是否静音
setVideoQuality(profile)	设置视频质量
switchToAudioCall()	视频通话切换语音通话
switchToVideoCall()	语音通话切换视频通话
getCameras()	获取摄像头设备列表
getMicrophones()	获取麦克风设备列表
switchDevice({deviceType, deviceId})	切换摄像头或麦克风设备

TRTCCalling 详解

创建 TRTCCalling 组件实例

首先，您需要在 实时音视频控制台 中创建一个应用，并取得 SDKAppID。

之后，就可以通过 `new TRTCCalling()` 方法获取 TRTCCalling 组件的一个实例。

```
npm i trtc-js-sdk --save
npm i tim-js-sdk --save
npm i tsignaling --save
npm i trtc-calling-js --save
// 如果是通过node下载的依赖，则使用 import 引入
import TRTCCalling from 'trtc-calling-js';
// 如果您通过 script 方式使用 trtc-calling-js，需要按顺序
// 手动引入 trtc.js
<script src="./trtc.js"></script>
// 接着手动引入 tim-js.js
<script src="./tim-js.js"></script>
// 然后再手动引入 tsignaling.js
<script src="./tsignaling.js"></script>
// 最后再手动引入 trtc-calling-js.js
<script src="./trtc-calling-js.js"></script>
let options = {
SDKAppID: 0, // 接入时需要将0替换为您的即时通信IM应用的 SDKAppID
// 从v0.10.2起，新增 tim 参数
```

```
// tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性
tim: tim
};
let trtcCalling = new TRTCCalling(options);
```

事件订阅/取消订阅相关接口函数

on(eventName, callback, context)

用于监听组件派发的事件，详细事件请参见 事件表。

```
let handleInvite = function ({inviteID, sponsor, inviteData}) {
console.log(`inviteID: ${inviteID}, sponsor: ${sponsor}, inviteData: ${JSON.stringify(inviteDat
a)}`);
};
trtcCalling.on('onInvited', handleInvite, this);
```

off(eventName, callback, context)

用于取消事件监听。

```
let handleInvite = function ({inviteID, sponsor, inviteData}) {
console.log(`inviteID: ${inviteID}, sponsor: ${sponsor}, inviteData: ${JSON.stringify(inviteDat
a)}`);
};
trtcCalling.off('onInvited', handleInvite, this);
```

SDK 基础函数

login({userID, userSig})

登录接口。

```
trtcCalling.login({userID, userSig})
```

参数如下表所示：

参数	类型	含义
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。

logout()

登出接口。

```
trtcCalling.logout()
```

通话操作相关接口函数

call({userID, type, offlinePushInfo})

1对1通话邀请，其中 type 为通话类型，1-语音通话，2-视频通话。

> 说明：
>
> - v1.0.0 及其之后版本，取消 timeout 参数
> - v1.0.0 及其之后版本，新增 offlinePushInfo 参数（**离线推送仅适用于 Android 或 iOS 终端，Web 和微信小程序不支持**）。

```
// v1.0.0 之前
trtcCalling.call({userID, type, timeout});

// v1.0.0 版本及其之后
const offlinePushInfo = {
title: '',
description: '您有一个通话请求',
}
trtcCalling.call({userID, type, offlinePushInfo})
```

参数如下表所示：

参数	类型	含义
userID	String	被邀请方 userID
type	Number	1：语音通话，2：视频通话
timeout	Number	0为不超时，单位 s（秒）。**仅限于v1.0.0之前的版本**
offlinePushInfo	Object	自定义离线消息推送（选填)。**仅限于v1.0.0及其之后的版本**

offlinePushInfo 参数 (仅限于v1.0.0及其之后的版本)

参数	类型	含义

参数	类型	含义
title	String	离线推送标题（选填）
description	String	离线推送内容（选填）
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
extension	String	离线推送透传内容（选填），**仅限于TRTCCalling 版本>=1.0.2, tsignaling 版本 >= 0.9.0**

groupCall({userIDList, type, groupID, offlinePushInfo})

groupID 参数是 IM SDK 中的群组 ID，如果填写该参数，那么通话请求消息是通过群消息系统广播出去的，这种消息广播方式比较简单可靠。如果不填写，那么 TRTCCalling 组件会采用单发消息逐一通知。

> 说明：
> v1.0.0 及其之后版本，新增 offlinePushInfo 参数（**离线推送仅适用于 Android 或 iOS 终端，Web 和微信小程序不支持**）。

```
// v1.0.0之前
trtcCalling.groupCall({userIDList, type, groupID});

// v1.0.0及其之后
const offlinePushInfo = {
title: '',
description: '您有一个通话请求',
}
trtcCalling.groupCall({userIDList, type, groupID, offlinePushInfo})
```

参数如下表所示：

参数	类型	含义
userIDList	Array	邀请列表
type	Number	1：语音通话，2：视频通话
groupID	String	IM 群组 ID（选填）
offlinePushInfo	Object	自定义离线消息推送（选填）。**仅限于v1.0.0及其之后的版本**

offlinePushInfo 参数 (仅限于v1.0.0及其之后的版本)

参数	类型	含义
title	String	离线推送标题（选填）
description	String	离线推送内容（选填）
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
extension	String	离线推送透传内容（选填），**仅限于TRTCCalling 版本>=1.0.2, tsignaling 版本 >= 0.9.0**

accept()

当收到邀请后，调用该接口将接受当前的邀请。

> 说明：
>
> - 当上一个 invitation 未处理完成时，组件会默认占线，之后的邀请都会回复忙线。
> - v1.0.0 及其之后版本，取消 params 参数。

```
import TRTCCalling from 'trtc-calling-js';
trtcCalling.on(TRTCCalling.EVENT.INVITED, ({inviteID, sponsor, inviteData}) => {
// ...
// v1.0.0之前
const { roomID, callType } = inviteData;
trtcCalling.accept({inviteID, roomID, callType})
// v1.0.0及其之后
trtcCalling.accept();
})
```

参数如下表所示：

参数	类型	含义
inviteID	String	邀请 ID，标识一次邀请（监听事件 INVITED 回调数据 inviteID）。**仅限于v1.0.0之前的版本**
roomID	Number	通话房间号 ID（监听事件 INVITED 回调数据 inviteData.roomID）。**仅限于v1.0.0之前的版本**
callType	Number	1：语音通话，2：视频通话（监听事件 INVITED 回调数据 inviteData.callType）。**仅限于v1.0.0之前的版本**

reject()

当收到邀请后，调用该接口将拒绝当前的邀请。

> 说明：
> v1.0.0 及其之后版本，取消 params 参数。

```
import TRTCCalling from 'trtc-calling-js';
trtcCalling.on(TRTCCalling.EVENT.INVITED, ({inviteID, sponsor, inviteData}) => {
// ...
// v1.0.0之前
const { callType } = inviteData;
trtcCalling.reject({inviteID, isBusy, callType})
// v1.0.0及其以后
trtcCalling.reject();
})
```

参数如下表所示：

参数	类型	含义
inviteID	String	邀请 ID, 标识一次邀请（监听事件 INVITED 回调数据 inviteID）。**仅限于v1.0.0之前的版本**
isBusy	Boolean	是否是忙线中。**仅限于v1.0.0之前的版本**
callType	Number	1：语音通话，2：视频通话（监听事件 INVITED 回调数据 inviteData.callType）。**仅限于v1.0.0之前的版本**

hangup()

1. 当您处于通话中，可以调用该函数结束通话。
2. 当未拨通时，可用来取消通话。

```
trtcCalling.hangup()
```

视频控制相关接口函数

startRemoteView({userID, videoViewDomID})

将远端用户的摄像头数据渲染到指定的 DOM ID 节点里。

```
trtcCalling.startRemoteView({userID, videoViewDomID})
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该用户数据将通过渲染到该 DOM ID 节点的 video 标签进行播放

stopRemoteView({userID})

将远端用户的摄像头数据渲染的 DOM 节点删除。

> 说明：
> v1.0.0 及其之后版本，移除 videoViewDomID 参数。

```
// v1.0.0之前
trtcCalling.stopRemoteView({userID, videoViewDomID});
// v1.0.0及其之后
trtcCalling.stopRemoteView({userID});
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该 DOM ID 节点的 video 标签进行移除，停止播放视频。**仅限于v1.0.0之前的版本**

startLocalView({userID, videoViewDomID})

将本地用户的摄像头数据渲染到指定的 DOM ID 节点里。

```
trtcCalling.startLocalView({userID, videoViewDomID})
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

参数	类型	含义
videoViewDomID	String	本地用户数据将通过渲染到该 DOM ID 节点的 video 标签进行播放

stopLocalView({userID})

将本地用户的摄像头数据渲染的 DOM 节点删除。

> 说明：
> v1.0.0 及其之后版本，移除 videoViewDomID 参数。

```
// v1.0.0之前
trtcCalling.stopLocalView({userID, videoViewDomID});
// v1.0.0及其之后
trtcCalling.stopLocalView({userID});
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
videoViewDomID	String	该 DOM ID 节点的 video 标签进行移除，停止播放视频。**仅限于v1.0.0之前的版本**

openCamera()

开启本地摄像头。

```
trtcCalling.openCamera()
```

closeCamera()

关闭摄像头。

```
trtcCalling.closeCamera()
```

setMicMute(isMute)

开启/关闭麦克风。

```
trtcCalling.setMicMute(true) // 关闭麦克风
```

参数如下表所示：

参数	类型	含义
isMute	Boolean	• true: 麦克风关闭 • false: 麦克风打开

setVideoQuality(profile)

设置视频质量。

> 说明：
>
> - v0.8.0 及其之后版本，新增该方法。
> - 此方法需在 call、groupCall、accept 之前设置，之后设置不生效。

```
trtcCalling.setVideoQuality('720p') // 设置视频质量为720p
```

参数如下表所示：

参数	类型	含义
profile	String	• 480p : 640 × 480 • 720p : 1280 × 720 • 1080p : 1920 × 1080

switchToAudioCall()

视频通话切换语音通话。

> 说明：
>
> - v0.10.0 及其之后版本，新增该方法。
> - 仅支持1v1通话过程中使用。
> - 失败监听 ERROR 事件，code：60001。

```
trtcCalling.switchToAudioCall() // 视频通话切换语音通话
```

switchToVideoCall()

语音通话切换视频通话。

> 说明：
>
> - v0.10.0 及其之后版本，新增该方法。
> - 仅支持1v1通话过程中使用。
> - 失败监听 ERROR 事件，code：60002。

```
trtcCalling.switchToVideoCall() // 语音通话切换视频通话
```

getCameras()

您可以调用此接口获取摄像头设备列表。

> 说明：
> v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.getCameras() // 获取摄像头列表
```

getMicrophones()

您可以调用此接口获取麦克风设备列表。

> 说明：
> v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.getMicrophones() // 获取麦克风列表
```

switchDevice({deviceType, deviceId})

您可以调用此接口切换摄像头或麦克风设备。

说明：
v1.0.0 及其之后版本，新增该方法。

```
trtcCalling.switchDevice({deviceType: 'audio', deviceId: deviceId}) // 切换设备
```

参数如下表所示：

参数	类型	含义
deviceType	String	video：摄像头，audio：麦克风
deviceId	String	• 摄像头设备标识通过 getCameras() 获取 • 麦克风设备标识通过 getMicrophones() 获取

TRTCCalling 事件表

您可以参考如下代码监听 TRTCCalling 组件事件：

```
import TRTCCalling from 'trtc-calling-js';
// etc
function handleInviteeReject({userID}) {

}
trtcCalling.on(TRTCCalling.EVENT.REJECT, handleInviteeReject)
```

邀请方事件

CODE	事件接收方	说明
REJECT	邀请方	被邀用户拒绝通话
NO_RESP	邀请方	被邀用户超时无应答
LINE_BUSY	邀请方	被邀用户正在通话中，忙线
INVITED	被邀方	收到了邀请通知
CALLING_CANCEL	被邀方	本次通话被取消了
CALLING_TIMEOUT	被邀方	本次通话超时未应答

CODE	事件接收方	说明
USER_ENTER	邀请方和被邀方	用户进房
USER_LEAVE	邀请方和被邀方	有用户离开通话
CALL_END	邀请方和被邀方	本次通话结束
KICKED_OUT	邀请方和被邀方	重复登录，被踢出房间
USER_VIDEO_AVAILABLE	邀请方和被邀方	远端用户开启/关闭了摄像头
USER_AUDIO_AVAILABLE	邀请方和被邀方	远端用户开启/关闭了麦克风

通用事件回调

SDK_READY

SDK 进入 ready 状态时收到该回调

> 说明：
> v1.0.0 及其之后版本，新增此事件。

```
let onSDKReady = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.SDK_READY, onSDKReady);
```

USER_ENTER

用户进房。
触发条件：当有用户进入通话。

```
let handleUserEnter = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.USER_ENTER, handleUserEnter);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userID	String	用户 ID

USER_LEAVE

用户退出房间。

触发条件：当有用户退出通话。

```
let handleUserLeave = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.USER_LEAVE, handleUserLeave);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

GROUP_CALL_INVITEE_LIST_UPDATE

群聊更新邀请列表收到该回调

> 说明：
> v1.0.0 及其之后版本，新增此事件。

```
let handleGroupInviteeListUpdate = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeListUpdate);
```

CALL_END

本次通话结束。

触发条件：结束本次通话。

```
let handleCallingEnd = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALL_END, handleCallingEnd);
```

KICKED_OUT

重复登录，被踢出房间。

触发条件：在其他页面重复登录。

```
let handleKickedOut = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.KICKED_OUT, handleKickedOut);
```

USER_VIDEO_AVAILABLE

远端用户打开关闭摄像头。

触发条件：远端用户打开/关闭摄像头。

```
let handleUserVideoChange = function({userID, isVideoAvailable}) {
console.log(userID, isVideoAvailable)
};
trtcCalling.on(TRTCCalling.EVENT.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
isVideoAvailable	Boolean	• true：远端用户打开摄像头 • false：远端用户关闭摄像头

USER_AUDIO_AVAILABLE

远端用户开启/关闭了麦克风。

触发条件：远端用户开启/关闭了麦克风。

```
let handleUserAudioChange = function({userID, isAudioAvailable}) {
console.log(userID, isAudioAvailable)
};
trtcCalling.on(TRTCCalling.EVENT.USER_AUDIO_AVAILABLE, handleUserAudioChange);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

参数	类型	含义
isAudioAvailable	Boolean	• true：远端用户打开麦克风 • false：远端用户关闭麦克风

邀请方事件回调

REJECT

用户拒绝通话。

触发条件：被邀请方拒绝通话，发起方收到 REJECT 事件回调。

```
let handleInviteeReject = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.REJECT, handleInviteeReject);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

NO_RESP

邀请用户无应答。

触发条件：当 call/groupCall 设置 timeout，被邀请方未在 timeout 内未在接听，发起方收到 NO_RESP 事件回调。

```
let handleNoResponse = function({userID, userIDList}) {
console.log(userID, userIDList)
};
trtcCalling.on(TRTCCalling.EVENT.NO_RESP, handleNoResponse);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID
userIDList	Array	超时用户列表

LINE_BUSY

被邀请方正在通话中，忙线。

触发条件：被邀请方已再另一通话中，发起方收到 LINE_BUSY 事件回调。

```
let handleLineBusy = function({userID}) {
console.log(userID)
};
trtcCalling.on(TRTCCalling.EVENT.LINE_BUSY, handleLineBusy);
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID

被邀请方事件回调

INVITED

收到邀请通知。

触发条件：当有邀请通话时，被邀请方收到 INVITED 事件回调。

```
let handleNewInvitationReceived = function({
sponsor, userIDList, isFromGroup, inviteData, inviteID
}) {
console.log(sponsor, userIDList, isFromGroup, inviteData, inviteID)
};
trtcCalling.on(TRTCCalling.EVENT.INVITED, handleNewInvitationReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
userIDList	Array	同时还被邀请的人
isFromGroup	Boolean	是否 IM 群组邀请
inviteData	Object	针对新用户邀请： {version, callType, roomID}
inviteID	String	邀请 ID，标识一次邀请

CALLING_CANCEL

本次通话被取消了。

触发条件：发起方在呼叫过程中取消通话，被邀请方收到 CALLING_CANCEL 事件回调。

```
let handleCallingCancel = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALLING_CANCEL, handleCallingCancel);
```

CALLING_TIMEOUT

本次通话超时未应答。

触发条件：当 call/groupCall 设置 timeout，被邀请方未在 timeout 内未在接听，被邀请方收到 CALLING_TIMEOUT 事件回调。

```
let handleCallingTimeout = function(event) {
console.log(event)
};
trtcCalling.on(TRTCCalling.EVENT.CALLING_TIMEOUT, handleCallingTimeout);
```

TRTCCalling 错误码表

您可以通过监听 EVENT 里的 ERROR 字段，对组件抛出的错误进行处理，示例代码如下：

```
import TRTCCalling from 'trtc-calling-js';
let onError = function(error) {
console.log(error)
};
trtcCalling.on(TRTCCalling.EVENT.ERROR, onError);
```

Error code 码

code	错误类型	含义
60001	方法调用失败	switchToAudioCall 调用失败
60002	方法调用失败	switchToVideoCall 调用失败
60003	权限获取失败	没有可用的麦克风设备
60004	权限获取失败	没有可用的摄像头设备
60005	权限获取失败	用户禁止使用设备
60006	环境检测失败	当前环境不支持 WebRTC（ >=v1.0.4版本 ）

升级指引

- **升级 TRTCCalling 版本 >= 1.0.2**
  - 注意：需升级 TSignaling 版本 >= 0.9.0
  - 原因：更新日志
- **升级 1.0.2 > TRTCCalling 版本 >=1.0.0**
  - 注意：需升级 TSignaling 版本 >= 0.8.0
  - 原因：更新日志

常见问题

为什么拨打不通，或者被踢下线？

组件暂不支持多实例登入，不支持**离线推送信令**功能，请您确认登入账号的唯一性。

> 说明：
>
> - **多实例**：一个 UserID 重复登入，或在不同端登入，将会引起信令的混乱。
> - **离线推送**：实例在线才能接收消息，实例离线时接收到的信令不会在上线后重新推送。
>
> 更多常见问题，请参见 TRTCCalling Web 相关问题。

技术咨询

了解更多详情您可以联系我们或发送邮件至colleenyu@tencent.com。

参考文档

- TRTCCalling web 官网体验
- TRTCCalling npm
- TRTCCalling web demo 源码
- TRTCCalling web API
- TRTCCalling web 相关问题

TRTCCalling API（Flutter）

最近更新时间：2022-01-10 11:28:35

TRTCCalling 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持1v1音视频通话。TRTCCalling 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 实时语音通话（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

TRTCCalling API 概览

SDK 基础函数

API	描述
sharedInstance	组件单例。
destroySharedInstance	销毁组件单例。
registerListener	增加事件回调。
unRegisterListener	移除回调接口。
destroy	销毁函数，如果无需再运行该实例，请调用该接口。
login	登录组件接口，所有功能需要先进行登录后才能使用。
logout	登出组件接口，登出后无法再进行拨打操作。

通话操作相关接口函数

API	描述
call	单人通话邀请。
accept	接受当前通话。
reject	拒绝当前通话。
hangup	结束当前通话。

音频控制相关接口函数

API	描述
setMicMute	静音本地音频采集。
setHandsFree	设置免提。

TRTCCallingDelegate API 概览

通用事件回调

API	描述
onError	错误回调。

邀请方回调

API	描述
onReject	拒绝通话回调。
onNoResp	对方无回应回调。
onLineBusy	通话忙线回调。

被邀请方回调

API	描述
onInvited	被邀请通话回调。
onCallingCancel	当前通话被取消回调。
onCallingTimeOut	当前通话超时回调。

通用回调

API	描述
onUserEnter	用户进入通话回调。
onUserLeave	用户离开通话回调。
onUserAudioAvailable	用户是否开启音频上行回调。

API	描述
onUserVoiceVolume	用户通话音量回调。
onCallEnd	通话结束回调。

SDK 基础函数

sharedInstance

sharedInstance 是 TRTCCalling 的组件单例。

```
static Future<TRTCCalling> sharedInstance();
```

destroySharedInstance

销毁组件单例。

```
static void destroySharedInstance();
```

destroy

销毁函数，如果无需再运行该实例，请调用该接口。

```
void destroy();
```

registerListener

TRTCCalling 事件回调，您可以通过 TRTCCallingDelegate 获得 TRTCCalling 的各种状态通知。

```
void registerListener(VoiceListenerFunc func);
```

unRegisterListener

移除回调接口。

```
void unRegisterListener(VoiceListenerFunc func);
```

login

登录组件。

```
Future<ActionCallback> login(int sdkAppId, String userId, String userSig);
```

参数如下表所示：

参数	类型	含义
sdkAppID	int	您可以在实时音视频控制台 >【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

logout

登出组件。

```
Future<ActionCallback> logout();
```

通话操作相关接口函数

call

单人通话邀请，当前处于通话中也可继续调用邀请他人。

```
Future<ActionCallback> call(String userId, int type);
```

参数如下表所示：

参数	类型	含义
userId	String	呼叫用户 ID。
type	int	1 表示语音通话，2 表示视频通话。

accept

接受当前通话。当您作为被邀请方收到 `onInvited()` 的回调时，可以调用该函数接听来电。

```
Future<ActionCallback> accept();
```

reject

拒绝当前通话。当您作为被邀请方收到 `onInvited()` 的回调时，可以调用该函数拒绝来电。

```
Future<ActionCallback> reject();
```

hangup

挂断当前通话。当您处于通话中，可以调用该函数结束通话。

```
void hangup();
```

音频控制相关接口函数

setMicMute

静音本地音频采集。

```
void setMicMute(bool isMute);
```

参数如下表所示：

参数	类型	含义
isMute	bool	true 表示麦克风关闭，false 表示麦克风打开。

setHandsFree

静音远端音频。

```
void setHandsFree(bool isHandsFree);
```

参数如下表所示：

参数	类型	含义
isHandsFree	bool	true 表示开启免提，false 表示关闭免提。

TRTCCallingDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

参数如下表所示：

参数	类型	含义
code	int	错误码。
msg	String	错误信息。

邀请方回调

onReject

拒绝通话回调。

参数如下表所示：

参数	类型	含义
userId	String	拒绝用户的 ID。

onNoResp

对方无回应回调。

参数如下表所示：

参数	类型	含义
userId	String	无回应用户的 ID。

onLineBusy

通话忙线回调。

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID。

被邀请方回调

onInvited

被邀请通话回调。

参数如下表所示：

参数	类型	含义
sponsor	String	发起方的 ID。
userIds	List<String>	除自己外被邀请 ID 列表。
isFromGroup	bool	是否多人通话邀请。
type	int	1 表示语音通话，2 表示视频通话。

onCallingCancel

当前通话被取消回调。接收方未处理请求，邀请方取消后会收到此回调。

onCallingTimeOut

当前通话超时回调。

通用回调

onUserEnter

用户进入通话回调。

参数如下表所示：

参数	类型	含义
userId	String	进入通话用户 ID。

onUserLeave

用户离开通话回调。

参数如下表所示：

参数	类型	含义
userId	String	离开通话用户 ID。

onUserAudioAvailable

用户是否开启音频上行回调。

参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID。
available	boolean	用户音频是否可用。

onUserVoiceVolume

用户通话音量回调。

参数如下表所示：

参数	类型	含义
userVolumes	List	所有正在说话的房间成员的音量，取值范围：0 - 100。
totalVolume	int	所有远端成员的总音量，取值范围：0 - 100。

onCallEnd

通话结束回调。

视频互动直播
视频互动直播(iOS)

最近更新时间：2022-03-10 10:33:37

效果展示

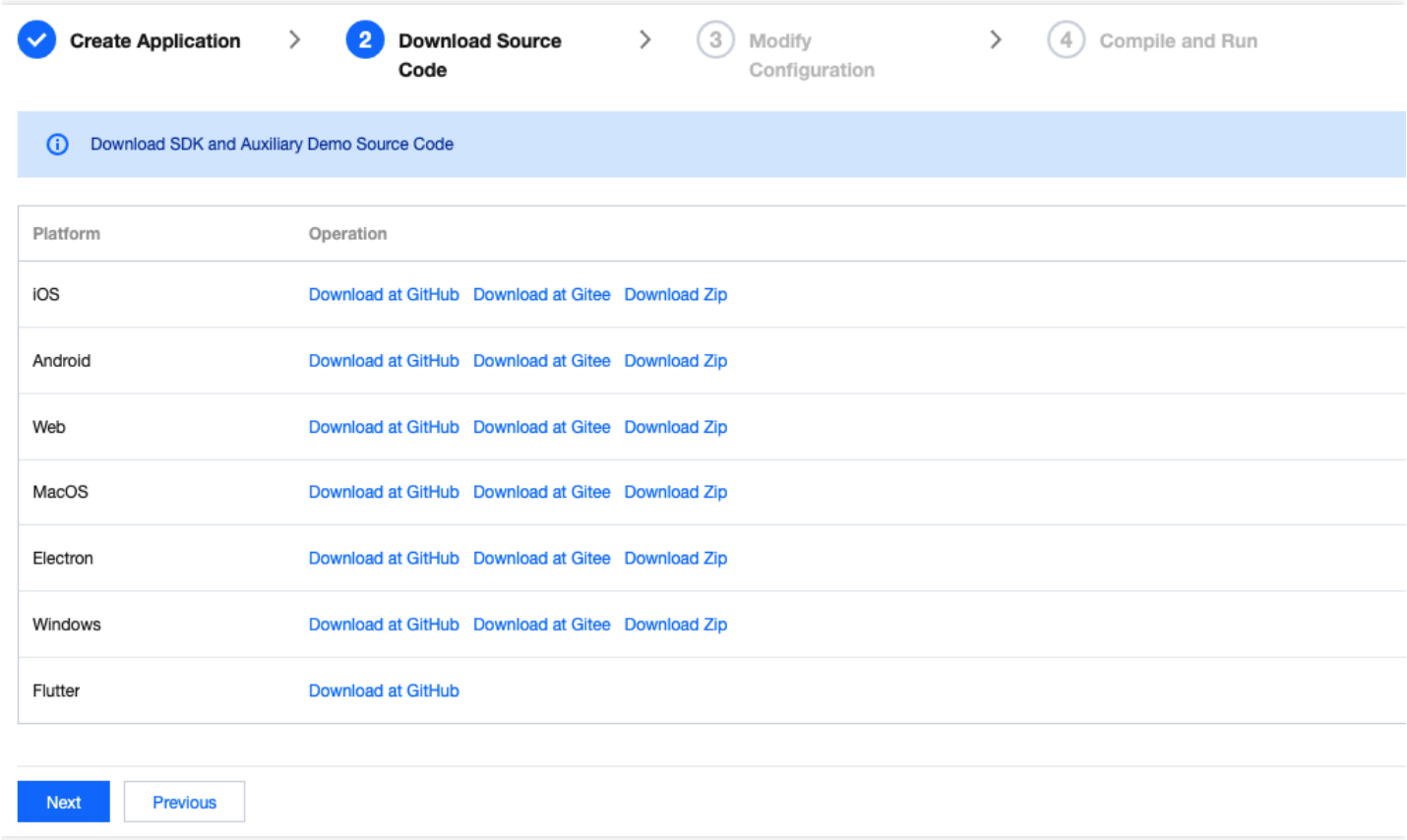
您可以 下载 安装我们的 App 体验互动直播的能力效果，包括互动连麦、主播 PK、低延时观看、弹幕聊天等 TRTC 在互动直播场景下的相关能力。

如需快速接入视频互动直播功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TUILiveRoom 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestLiveRoom` ，单击【创建】。
3. 单击【已下载，下一步】，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

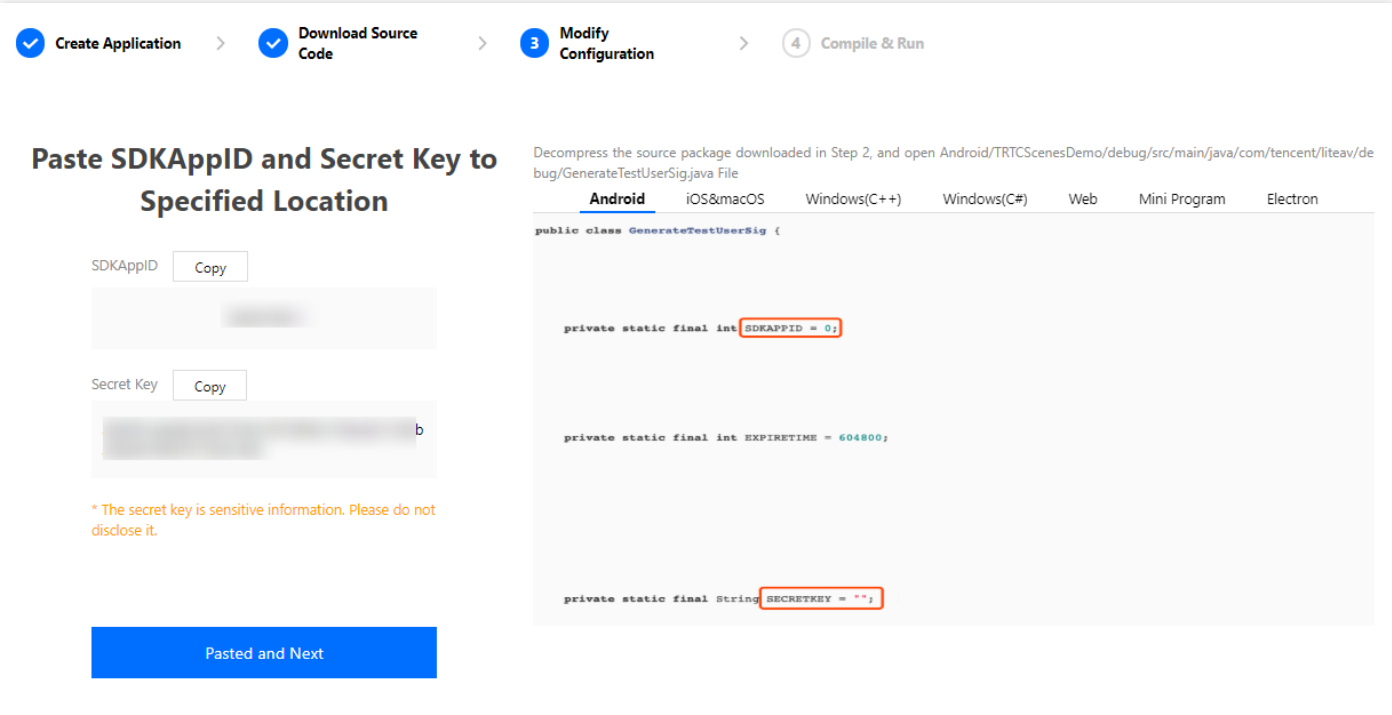
单击进入 TUILiveRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `TUILiveRoom/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Xcode（11.0及以上的版本）打开源码工程 `TUILiveRoom/TUILiveRoomApp.xcworkspace`，单击【运行】即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 `Source` 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个 swift 文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
Anchor	主播端相关 UI 的实现代码。
Audience	观众端相关 UI 的实现代码。
ChatRoom	文字聊天室以及弹幕消息的 UI 实现代码。
Common	可复用的一些 UI 组件的实现代码。
StatusView	状态浮层，会覆盖在视频画面上面，用于显示日志信息和视频加载动画。
LiveRoomMainViewController.swift	视频互动直播主页面 UI。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 单击【创建房间】。
3. 输入房间名称，单击【开始直播】。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入用户 A 创建的房间号，单击【进入房间】。

> 注意：
> 房间号在用户 A 房间创建完成后的系统弹框中获取。

房间状态监听&PK 列表接入

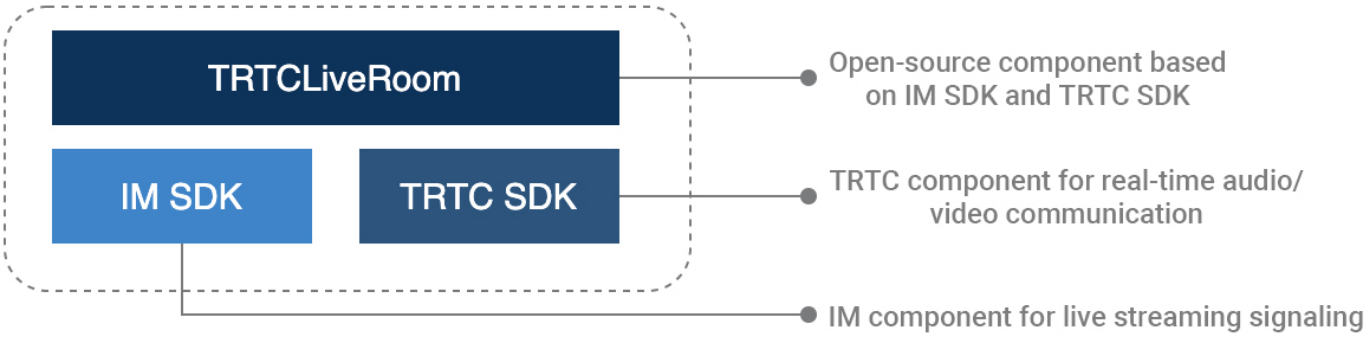
房间状态可使用 `TRTCLiveRoom` 进行监听，如下：

```
/////////////////////////////////////////////////////////////
//
// 房间管理相关
//
/////////////////////////////////////////////////////////////
/// 创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
/// 主播开播的正常调用流程是：
/// 1.【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
/// 2.【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 callback 通知给主播。
/// 3.【主播】调用 startPublish() 开始推流。
/// - Parameters:
/// - roomID: 房间标识，需要由您分配并进行统一管理。多个 roomid 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
/// - roomParam: TRTCCreateRoomParam | 房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
/// - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播开始直播的时候调用，可重复创建自己已创建过的房间。
- (void)createRoomWithRoomID:(UInt32)roomID
roomParam:(TRTCCreateRoomParam *)roomParam
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
/// 销毁房间（主播调用）
/// 主播在创建房间后，可以调用这个函数来销毁房间。
/// - Parameter callback: 销毁房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播在创建房间后，可以调用该函数来销毁房间。
- (void)destroyRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(destroyRoom(callback:));
/// 获取房间列表的详细信息
/// 其中的信息是主播在创建 createRoom() 时通过 roomInfo 设置进来的，如果房间列表和房间信息都由您自行管理，可忽略该函数。
/// - Parameter roomIDs: 房间号列表
/// - Parameter callback: 房间详细信息回调
- (void)getRoomInfosWithRoomIDs:(NSArray<nsnumber *="">> *)roomIDs
callback:(RoomInfoCallback _Nullable)callback
NS_SWIFT_NAME(getRoomInfos(roomIDs:callback:));
```

采用 Callback 方式进行回调是为了方便用户接入，例如获取房间列表可能需要进行异步操作，采用 Callback 方式将更加灵活。

实现自定义 UI 界面

源码 中的 `Source` 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCLiveRoom，您可以在 `TRTCLiveRoom.h` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

视频通话组件 TRTCLiveRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 info.plist 文件中需要添加 `Privacy > Camera Usage Description`， `Privacy > Microphone Usage Description` 申请摄像头和麦克风权限。

步骤3：导入 TUILiveRoom 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 将工程目录下的 `Source`、`Resources`、`TCBeautyKit`、`TXAppBasic` 文件夹、`TUILiveRoom.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
pod 'TXAppBasic', :path =&gt; "TXAppBasic/"
pod 'TCBeautyKit', :path =&gt; "TCBeautyKit/"
pod 'TXLiteAVSDK_TRTC'
pod 'TUILiveRoom', :path =&gt; "./", :subspecs =&gt; ["TRTC"]
```

步骤4：创建并登录组件

1. 调用 TRTCLiveRoom 的 `init` 接口可以创建一个 TRTCLiveRoom 组件的实例对象。
2. 创建一个 `TRTCLiveRoomConfig` 对象，该对象可以设置 useCDNFirst 和 CDNPlayDomain 属性：

- useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。
- CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。

3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
config	全局配置信息，请在登录时初始化，登录之后不可变更。 ◦ useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 ◦ CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。
callback	登录回调，成功时 code 为0。

```
class LiveRoomController: UIViewController {
let mLiveRoom = TRTCLiveRoom()
}
//useCDNFirst：true 表示普通观众通过 CDN 观看，false 表示普通观众通过低延时观看
```

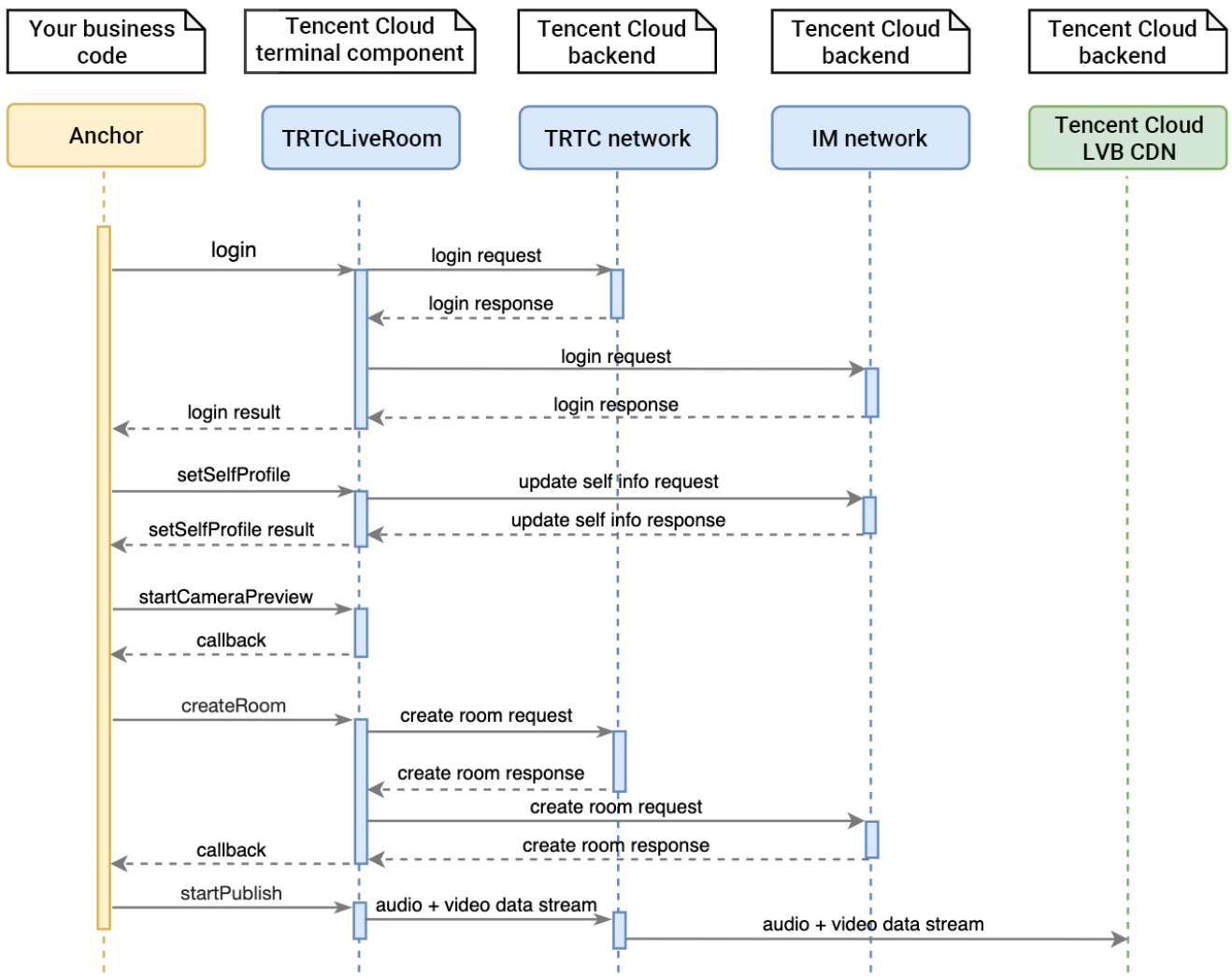
```
//CDNPlayDomain：表示 CDN 观看时配置的播放域名
let config = TRTCLiveRoomConfig(useCDNFirst: useCDNFirst, cdnPlayDomain: yourCDNPlayDomain)
mLiveRoom.login(SDKAPPID, userID, userSig, config) { (code, error) in
if code == 0 {
//登录成功
}
}
```

步骤5：主播端开播

1. 主播执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 主播在开播前可先调用 `startCameraPreview` 开启摄像头预览，界面上可以配置美颜调节按钮调用，通过 `getBeautyManager` 进行美颜设置。

> 说明：
> 非企业版 SDK 不支持变脸和贴图挂件等高级美颜功能。

3. 主播调整美颜效果后，可以调用 `createRoom` 创建新的直播间。
4. 主播调用 `startPublish` 开始推流。如需支持 CDN 观看，请在 login 时传入的 `TRTCLiveRoomConfig` 参数中指定 `useCDNFirst` 和 `CDNPlayDomain` 并在 `startPublish` 时指定直播拉流用的 streamID。




```
// 1.主播设置昵称和头像
mLiveRoom.setSelfProfile(name: "A", avatarURL: "faceUrl", callback: nil)
// 2.主播开播前预览并设置美颜参数
let view = UIView()
parentView.add(view)
mLiveRoom.startCameraPreview(frontCamera: true, view: view, callback: nil)
mLiveRoom.getBeautyManager().setBeautyStyle(.nature)
mLiveRoom.getBeautyManager().setBeautyLevel(6)
// 3.主播创建房间
let param = TRTCCreateRoomParam(roomName: "测试房间", coverUrl: "")
mLiveRoom.createRoom(roomID: 123456789, roomParam: param) { [weak self] (code, error) in
if code == 0 {
// 4.主播开启推流并将流发布到 CDN
self?.mLiveRoom.startPublish(streamID: mSelfUserId + "_stream", callback: nil)
}
}
```

步骤6：观众端观看

1. 观众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 观众端向业务后台获取最新的直播房间列表。

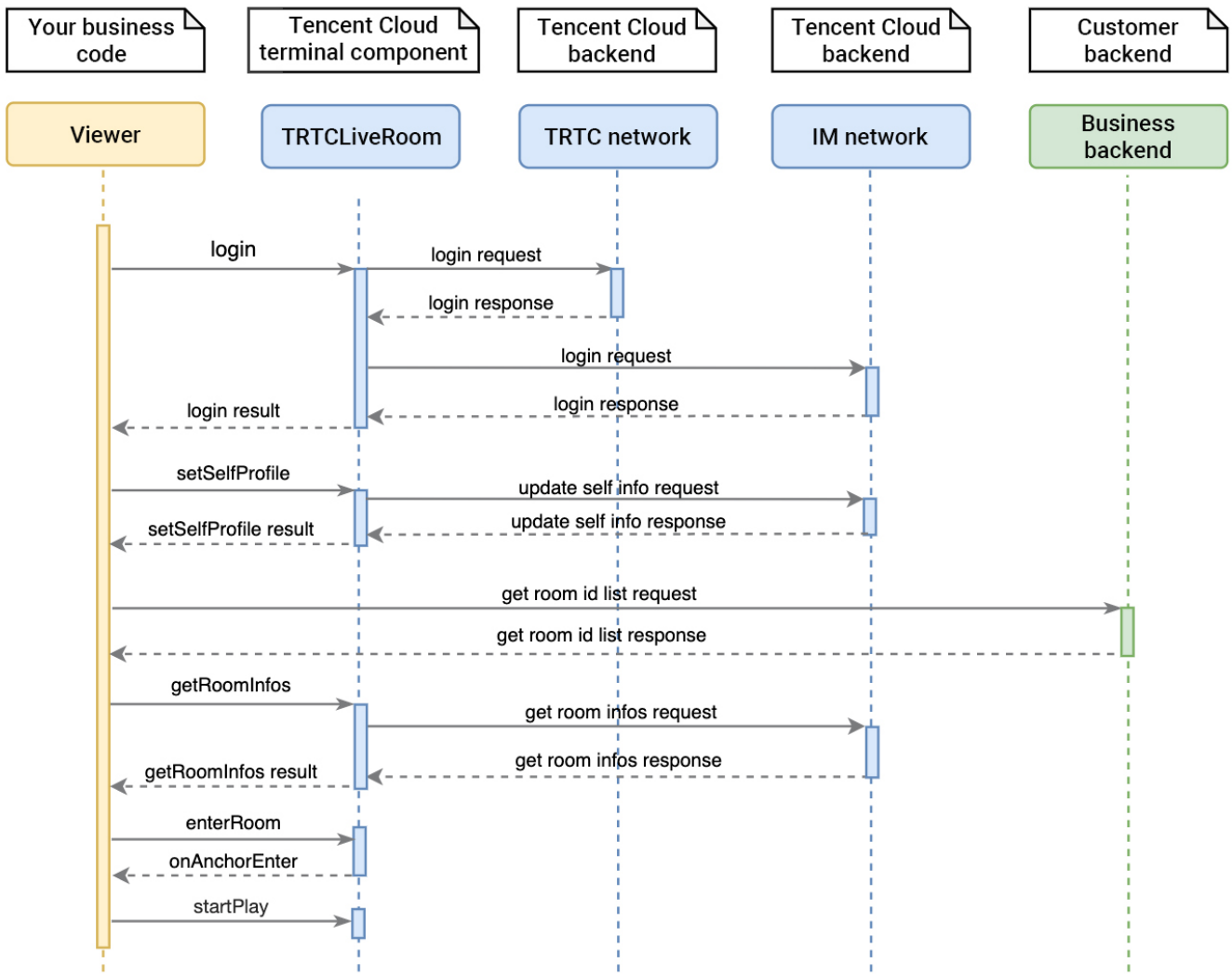
> 说明：
> App 中的直播间列表仅做演示使用，直播间列表的业务逻辑千差万别，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。

3. 观众端调用 `getRoomInfos` 获取房间的详细信息，该信息是在主播端调用 `createRoom` 创建直播间时设置的简单描述信息。

> 注意：
> 如果您的直播间列表包含了足够全面的信息，可跳过调用 `getRoomInfos` 相关步骤。

4. 观众选择一个直播间，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 调用 `startPlay` 并传入主播的 userId 开始播放。

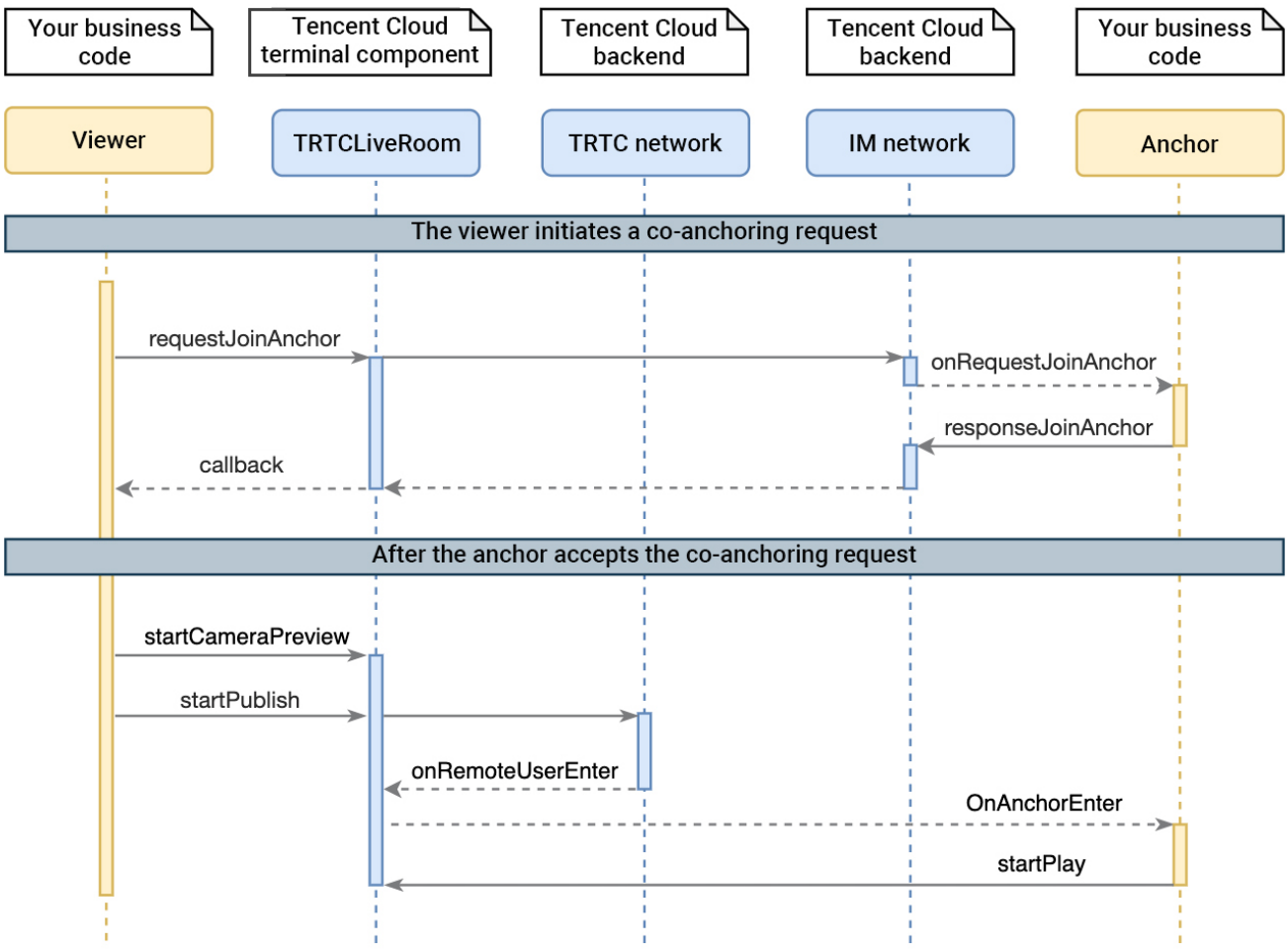
- 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay` 并传入主播的 userId 即可开始播放。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到主播 `onAnchorEnter` 的事件回调，该回调中携带主播的 userId 信息，调用 `startPlay` 即可播放。




```
// 1.假定您从业务后台获取房间列表为 roomList
var roomList: [UInt32] = GetRoomList()
// 2.通过调用 getRoomInfos 获取房间的详细信息
mliveRoom.getRoomInfos(roomIDs: roomList, callback: { (code, msg, list) in
if code == 0 {
// 获取到房间详细信息后，您可以在主播列表页面展示主播昵称、头像等相关信息
}
})
// 3.选择房间 roomid 进入
mliveRoom.enterRoom(roomID: roomID, callback: callback)
// 4.观众收到主播进房通知，开始播放
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
// 5.观众播放主播画面
mliveRoom.startPlay(userID: userID, view: renderView, callback: nil)
}
```

步骤7：观众与主播连麦

1. 观众端调用 `requestJoinAnchor` 向主播端发起连麦请求。
2. 主播端会收到 `TRTCLiveRoomDelegate#onRequestJoinAnchor` （即有观众请求与您连麦）的事件通知。
3. 主播端可以通过调用 `responseJoinAnchor` 决定是否接受来自观众端的连麦请求。
4. 观众端会收到 `TRTCLiveRoomDelegate#responseCallback` 事件通知，该通知会携带来自主播端的处理结果。
5. 如果主播同意连麦请求，观众端可调用 `startCameraPreview` 开启本地摄像头，随后调用 `startPublish` 启动观众端的推流。
6. 主播端会在观众端启动通知后收到 `TRTCLiveRoomDelegate#onAnchorEnter` （即另一路音视频流已到来）通知，该通知会携带观众端的 userId。
7. 主播端调用 `startPlay` 即可看到连麦观众的画面。

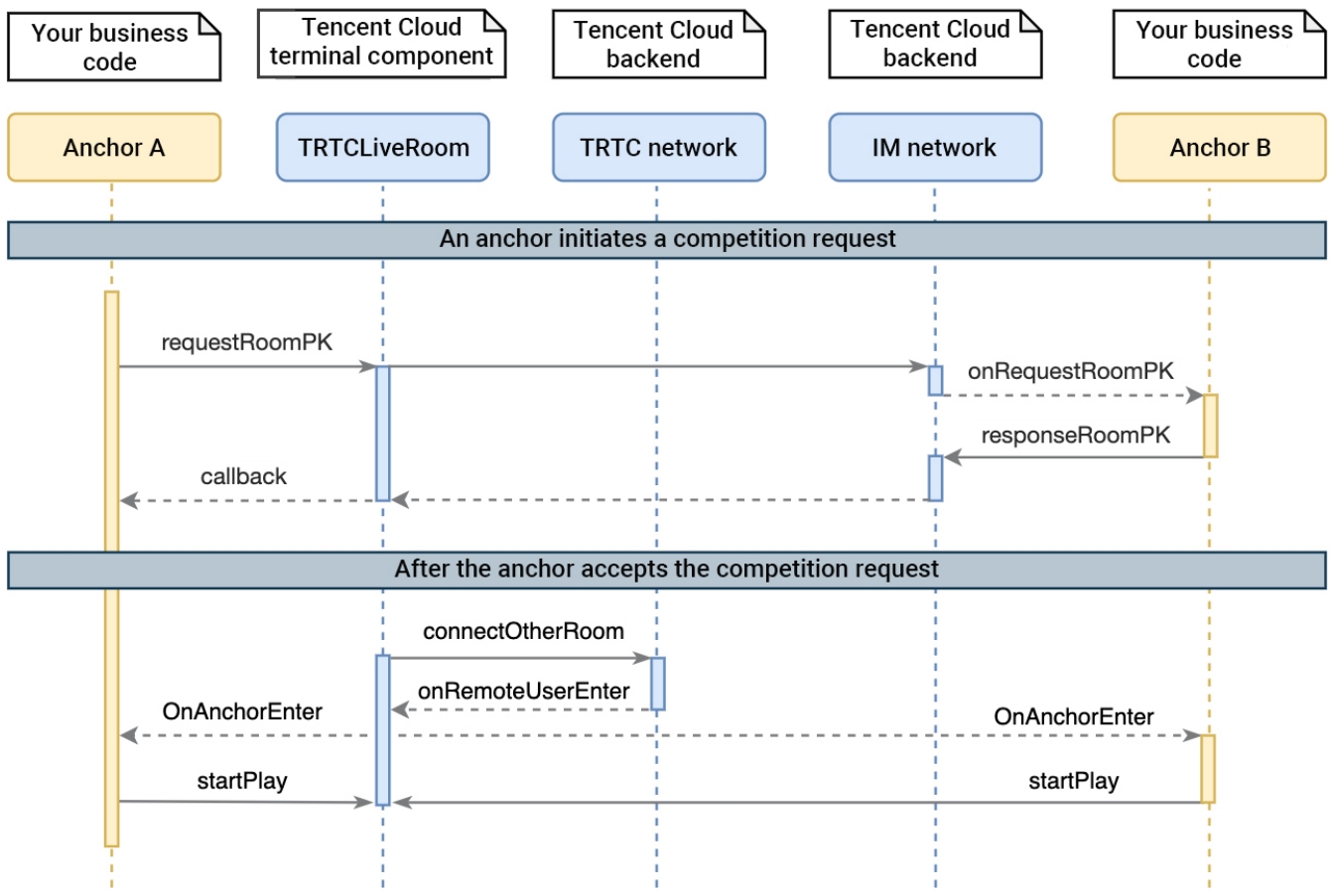



```
// 观众端：
// 1. 观众端发起连麦请求
mliveRoom.requestJoinAnchor(reason: mSelfUserId + "请求和您连麦", responseCallback: { [weak self]
(agreed, reason) in
```

```
// 4.主播接受了观众的请求
if agreed {
// 5.观众启动预览，开启推流
self?.mliveRoom.startCameraPreview(frontCamera: true, view: localView, callback: nil)
self?.mliveRoom.startPublish(streamID: streamID, callback: nil)
}
}, callback: callback)
// 主播端：
// 2.主播端收到连麦请求
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestJoinAnchor user: TRTCLiveUserInfo
, reason: String?, timeout: Double) {
// 3.同意对方的连麦请求
mliveRoom.responseJoinAnchor(userID: userID, agree: true, reason: "同意连麦")
}
// 6.主播收到连麦观众的上麦通知
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
// 7.主播播放观众画面
mliveRoom.startPlay(userID: userID, view: view, callback: nil)
}
```

步骤8：主播与主播 PK

1. 主播 A 调用 `requestRoomPK` 向主播 B 发起 PK 请求。
2. 主播 B 会收到 `TRTCLiveRoomDelegate onRequestRoomPK` 回调通知。
3. 主播 B 调用 `responseRoomPK` 决定是否接受主播 A 的 PK 请求。
4. 主播 B 接受主播 A 的请求，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 A。
5. 主播 A 收到 `responseCallback` 回调通知，PK 请求是否被同意。
6. 主播 A 请求被同意，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 B。




```
// 主播 A:
// 主播 A 创建12345的房间
mLiveRoom.createRoom(roomID: 12345, roomParam: param, callback: nil)

// 1.主播 A 向主播 B 发起 PK 请求
mLiveRoom.requestRoomPK(roomID: 54321, userID: "B", responseCallback: { (agree, reason) in
// 5.收到是否同意的回调
if agree {
}
}, callback: callback)

// 主播 A 收到主播 B 进入回调
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
// 6.收到 B 进房的通知，播放 B 的画面
mLiveRoom.startPlay(userID: userID, view: view, callback: callback)
}

// 主播 B:
// 主播 B 创建54321的房间
mLiveRoom.createRoom(roomID: 54321, roomParam: param, callback: nil)
```

```
// 2.主播 B 收到主播 A 的消息
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestRoomPK user: TRTCLiveUserInfo, ti
meout: Double) {
// 3.主播 B 回复主播 A 接受请求
mLiveRoom.responseRoomPK(userID: userID, agree: true, reason: reason)
}

public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
// 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
mLiveRoom.startPlay(userID: userID, view: view, callback: callback)
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomTextMsg` 回调。
  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
mLiveRoom.sendRoomTextMsg(message: "Hello Word!", callback: callback)
// 接收端：监听文本消息
mLiveRoom.delegate = self
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRecvRoomTextMsg message: String, from
User user: TRTCLiveUserInfo) {
debugPrint("收到来自¥(user.userName)的文本消息:¥(message)")
}
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomCustomMsg` 回调。
  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mLiveRoom.sendRoomCustomMsg(command: "CMD_DANMU", message: "Hello world", callback: nil)
mLiveRoom.sendRoomCustomMsg(command: "CMD_LIKE", message: "", callback: nil)
// 接收端：监听自定义消息
mLiveRoom.delegate = self
public func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRecvRoomCustomMsg command: String, me
ssage: String, fromUser user: TRTCLiveUserInfo) {
if "CMD_DANMU" == command {
// 收到弹幕消息
debugPrint("收到来自¥(user.userName)的弹幕消息:¥(message)")
```

```
} else if "CMD_LIKE" == command {
// 收到点赞消息
debugPrint("¥(user.userName)给您点了个赞！")
}
}
```

视频互动直播(Android)

最近更新时间：2022-03-11 16:22:14

效果展示

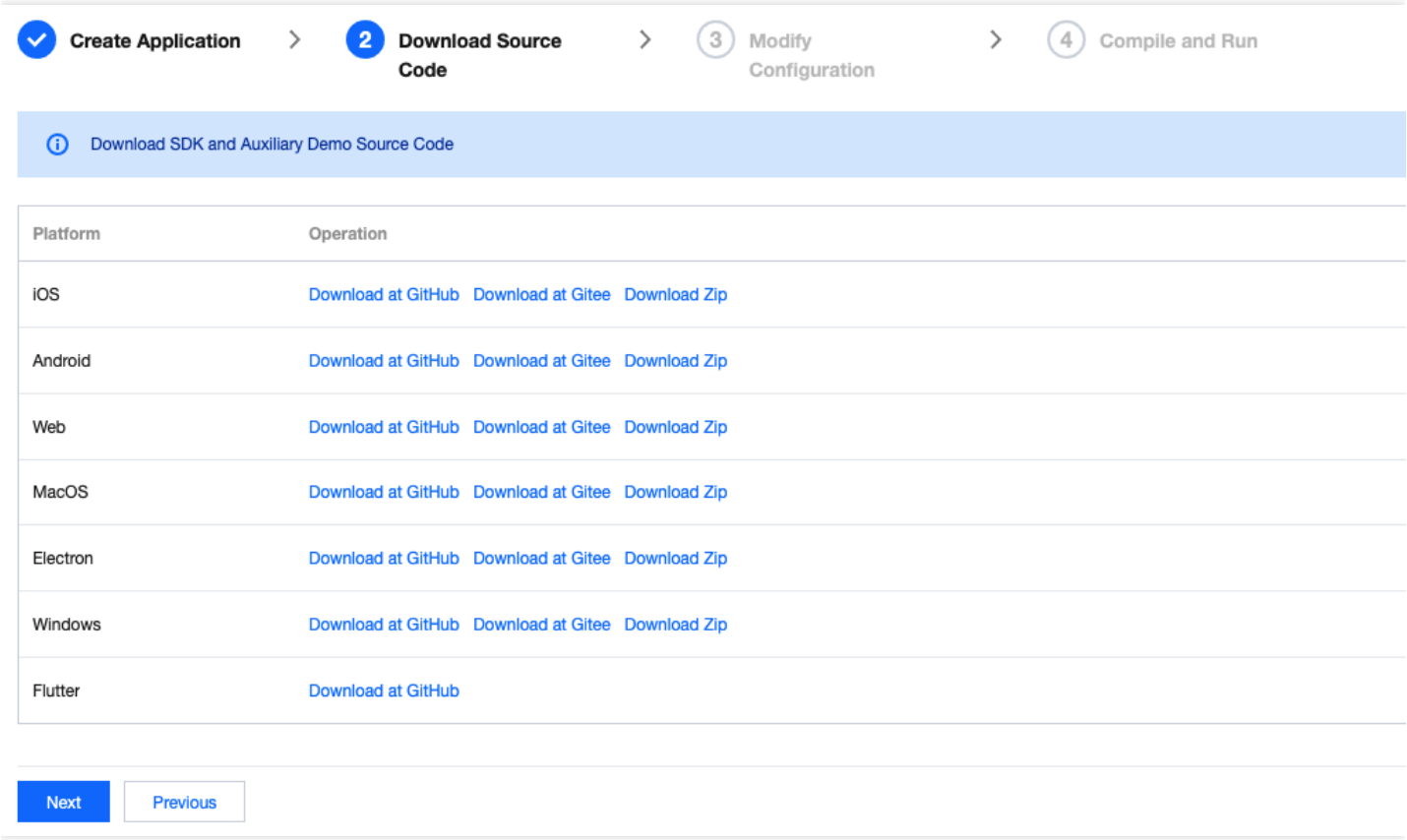
您可以 下载 安装我们的 App 体验互动直播的能力效果，包括互动连麦、主播 PK、低延时观看、弹幕聊天等 TRTC 在互动直播场景下的相关能力。

如需快速接入视频互动直播功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TRTCLiveRoom 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestLiveRoom`，单击【创建】。
3. 单击【已下载，下一步】，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

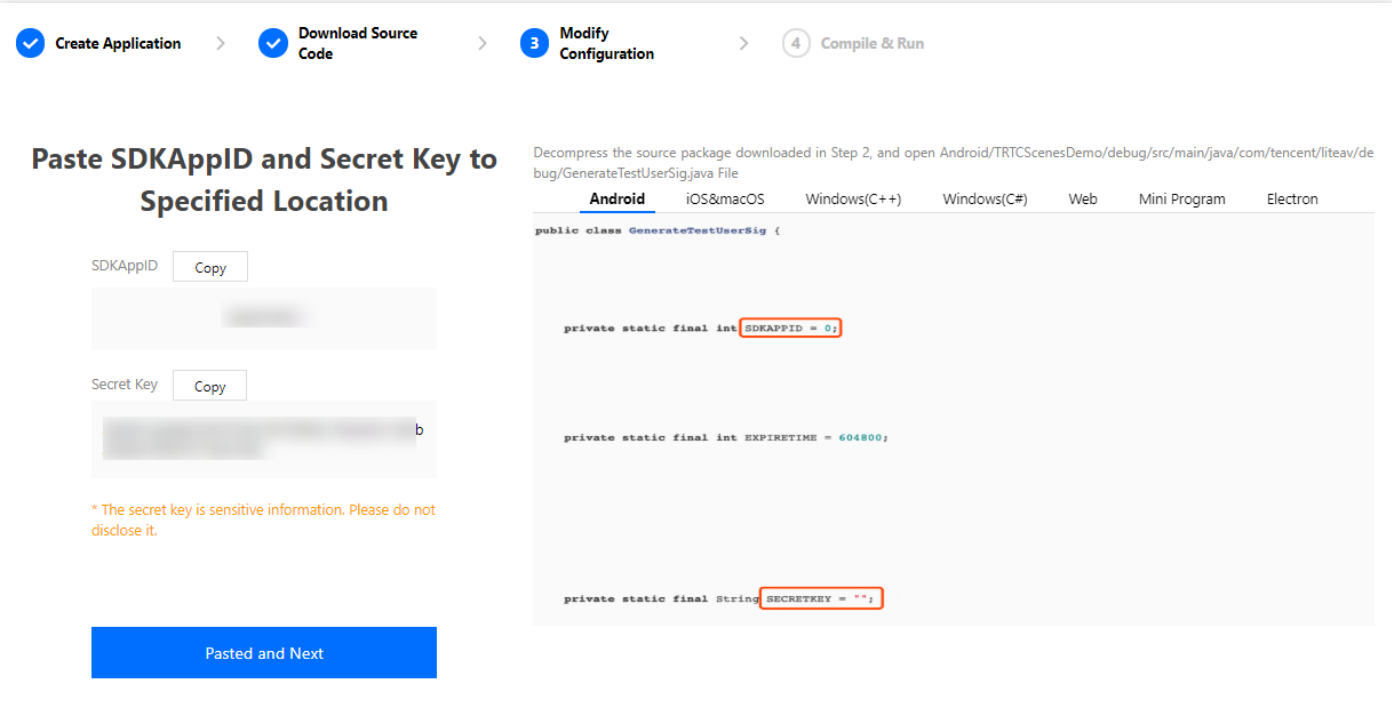
单击进入 TUILiveRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

> 注意：
> 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
> 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5以上的版本）打开源码工程 `TUILiveRoom`，单击【运行】即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 Source 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述

文件或文件夹	功能描述
anchor	主播端相关 UI 的实现代码。
audience	观众端相关 UI 的实现代码。
common	通用的 UI 组件实现代码。
widget	通用控件。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 进入后，单击【创建房间】。
3. 输入房间名称，单击【开始直播】。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）。
2. 输入用户 A 创建的房间号，单击进入房间。

> 注意：
> 房间号在用户 A 的房间顶部查看。

房间状态监听&PK 列表接入

房间状态可使用 `LiveRoomManager` 进行监听，如下：

```
LiveRoomManager.getInstance().addCallback(new LiveRoomManager.RoomCallback() {
/**
* 房间创建
* @param roomId
```

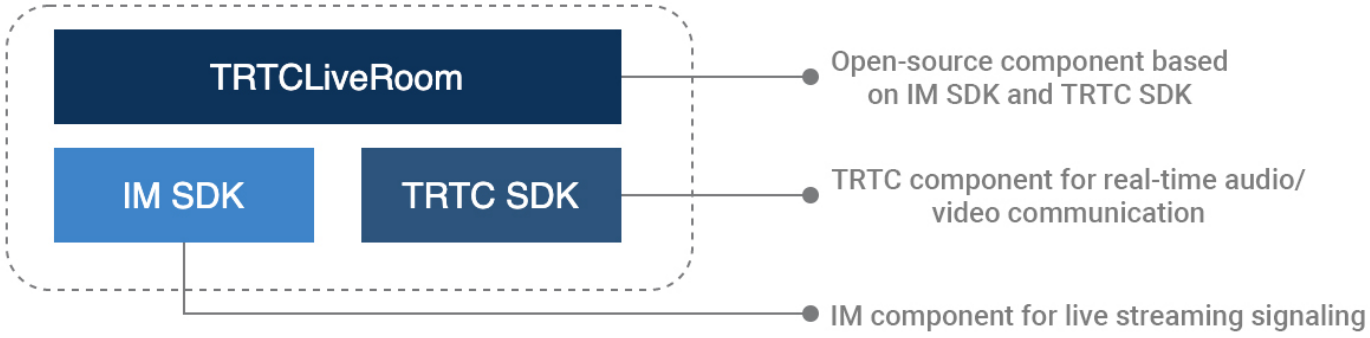
```
* @param callback 用于通知内部处理的结果
*/
@Override
public void onRoomCreate(int roomId, final LiveRoomManager.ActionCallback callback) {
// doSomething
}

/**
* 房间销毁
* @param roomId
* @param callback 用于通知内部处理的结果
*/
@Override
public void onRoomDestroy(int roomId, final LiveRoomManager.ActionCallback callback) {
// doSomething
}

/**
* 获取房间列表
* @param callback
*/
@Override
public void onGetRoomIdList(final LiveRoomManager.GetCallback callback) {
// 获取房间列表，需自行维护，用于用户之间的 PK
if(callback != null) {
if(success) {
// 回调成功，传入房间列表
callback.onSuccess(roomList);
} else {
// 房间列表获取失败
callback.onError(code, message);
}
}
}
});
```

采用 Callback 方式进行回调是为了方便用户接入，例如获取房间列表可能需要进行异步操作，采用 Callback 方式将更加灵活。

实现自定义 UI 界面

源码 中的 Source 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCLiveRoom，您可以在 `TRTCLiveRoom.java` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义

UI 界面。



步骤1：集成 SDK

视频互动直播组件 TRTCLiveRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 Maven 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
compile "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
compile 'com.tencent.imsdk:imsdk:latest.release'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击【Sync Now】，自动下载 SDK 并集成到工程里。

方法二：通过本地 AAR 依赖

若您的开发环境访问 maven 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：

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

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

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

步骤3：导入 TRTCLiveRoom 组件

拷贝以下目录中的所有文件到您的项目中：

```
src/main/java/com/tencent/liteav/liveroom/model
```

步骤4：创建并登录组件

1. 调用 sharedInstance 接口可以创建一个 TRTCLiveRoom 组件的实例对象。
2. 创建一个 TRTCLiveRoomConfig 对象，该对象可以设置 useCDNFirst 和 CDNPlayDomain 属性：

- useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。
- CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。

3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
config	全局配置信息，请在登录时初始化，登录之后不可变更。 ◦ useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 ◦ CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。
callback	登录回调，成功时 code 为0。

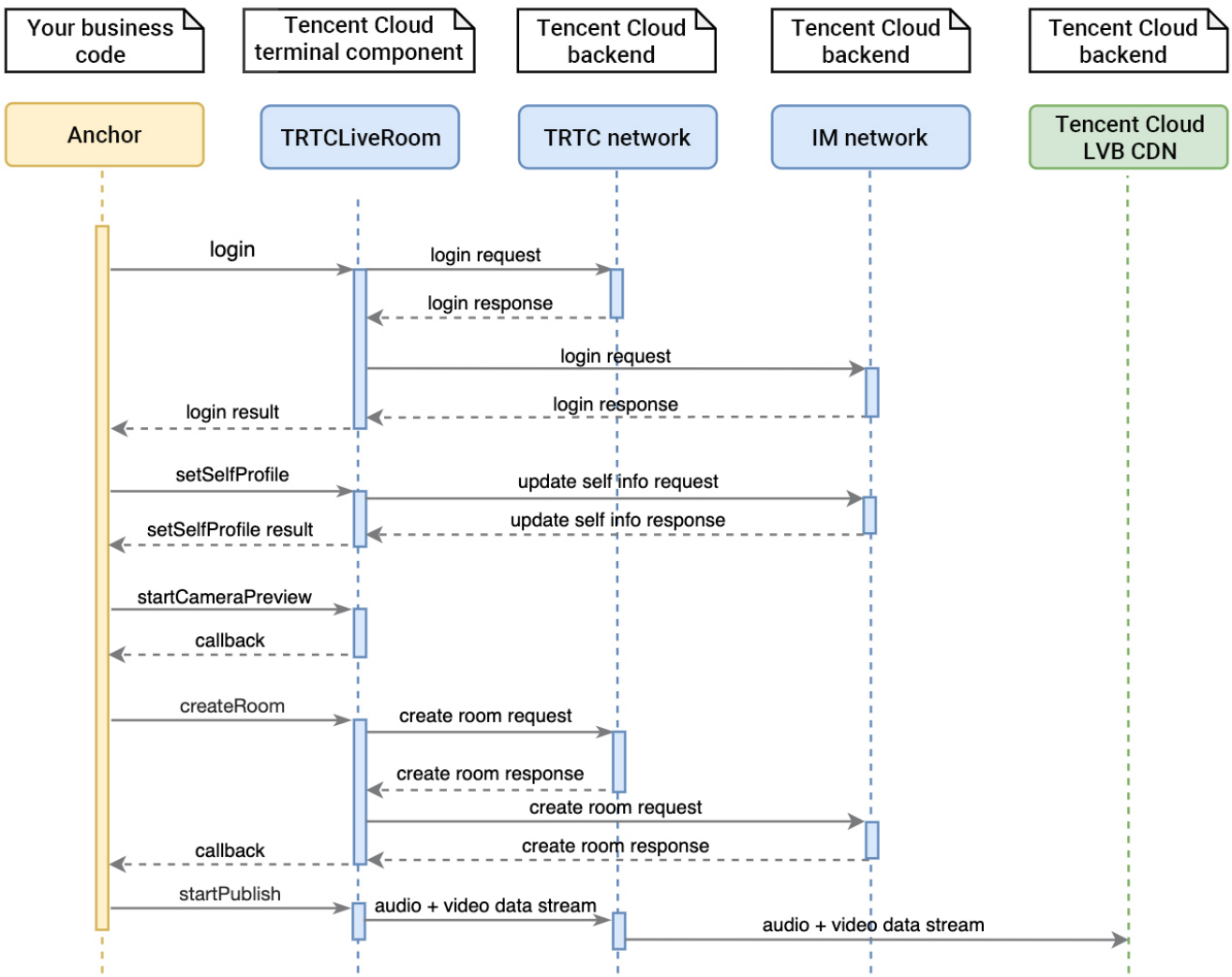
```
TRTCLiveRoom mLiveRoom = TRTCLiveRoom.sharedInstance(this);
//useCDNFirst：true 表示普通观众通过 CDN 观看，false 表示普通观众通过低延时观看
//yourCDNPlayDomain：表示 CDN 观看时配置的播放域名
TRTCLiveRoomDef.TRTCLiveRoomConfig config =
new TRTCLiveRoomDef.TRTCLiveRoomConfig(useCDNFirst, yourCDNPlayDomain);
mLiveRoom.login(SDKAPPID, userId, userSig, config,
new TRTCLiveRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

步骤5：主播端开播

1. 主播执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 主播在开播前可先调用 `startCameraPreview` 开启摄像头预览，界面上可以配置美颜调节按钮调用，通过 `getBeautyManager` 进行美颜设置。

> 说明：
> 非企业版 SDK 不支持变脸和贴图挂件等高级美颜功能。

3. 主播调整美颜效果后，可以调用 `createRoom` 创建新的直播间。
4. 主播调用 `startPublish` 开始推流。如需支持 CDN 观看，请在 login 时传入的 `TRTCLiveRoomConfig` 参数中指定 `useCDNFirst` 和 `CDNPlayDomain` 并在 `startPublish` 时指定直播拉流用的 streamID。




```
// 1. 主播设置昵称和头像
mLiveRoom.setSelfProfile("A", "your_face_url", null);

// 2. 主播开播前预览并设置美颜参数
TXCloudVideoView view = new TXCloudVideoView(context);
parentView.add(view);
mLiveRoom.startCameraPreview(true, view, null);
mLiveRoom.getBeautyManager().setBeautyStyle(1);
mLiveRoom.getBeautyManager().setBeautyLevel(6);

// 3. 主播创建房间
```

```
TRTCLiveRoomDef.TRTCCreateRoomParam param = new TRTCLiveRoomDef.TRTCCreateRoomParam();
param.roomName = "测试房间";
mLiveRoom.createRoom(123456789, param, new TRTCLiveRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
// 4.主播开启推流并将流发布到 CDN
mLiveRoom.startPublish(mSelfUserId + "_stream", null);
}
}
});
```

步骤6：观众端观看

1. 观众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 观众端向业务后台获取最新的直播房间列表。

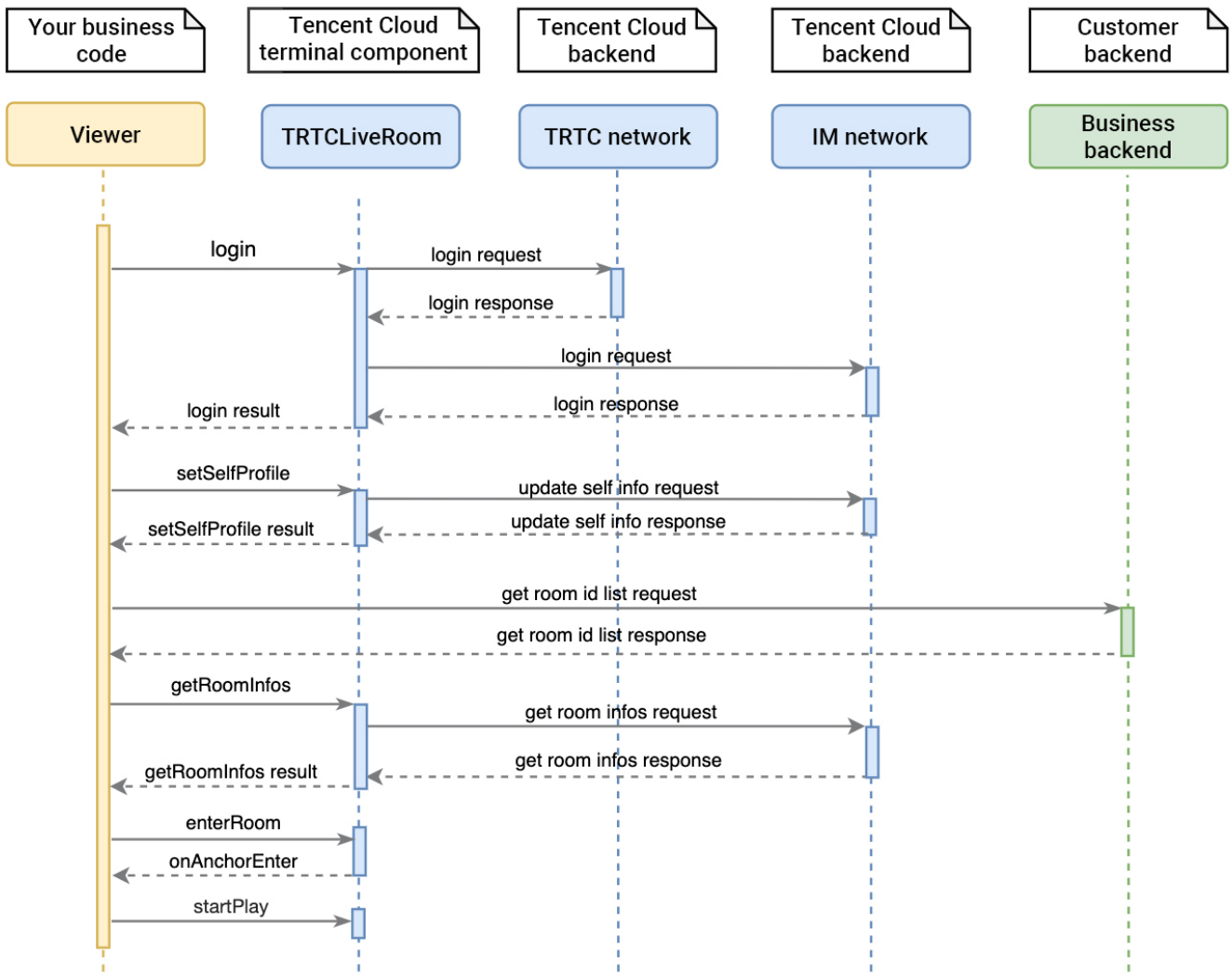
> 说明：
> App 中的直播间列表仅做演示使用，直播间列表的业务逻辑千差万别，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。

3. 观众端调用 `getRoomInfos` 获取房间的详细信息，该信息是在主播端调用 `createRoom` 创建直播间时设置的简单描述信息。

> 注意：
> 如果您的直播间列表包含了足够全面的信息，可跳过调用 `getRoomInfos` 相关步骤。

4. 观众选择一个直播间，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 调用 `startPlay` 并传入主播的 userId 开始播放。

- 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay` 并传入主播的 userId 即可开始播放。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到主播 `onAnchorEnter` 的事件回调，该回调中携带主播的 userId 信息，调用 `startPlay` 即可播放。

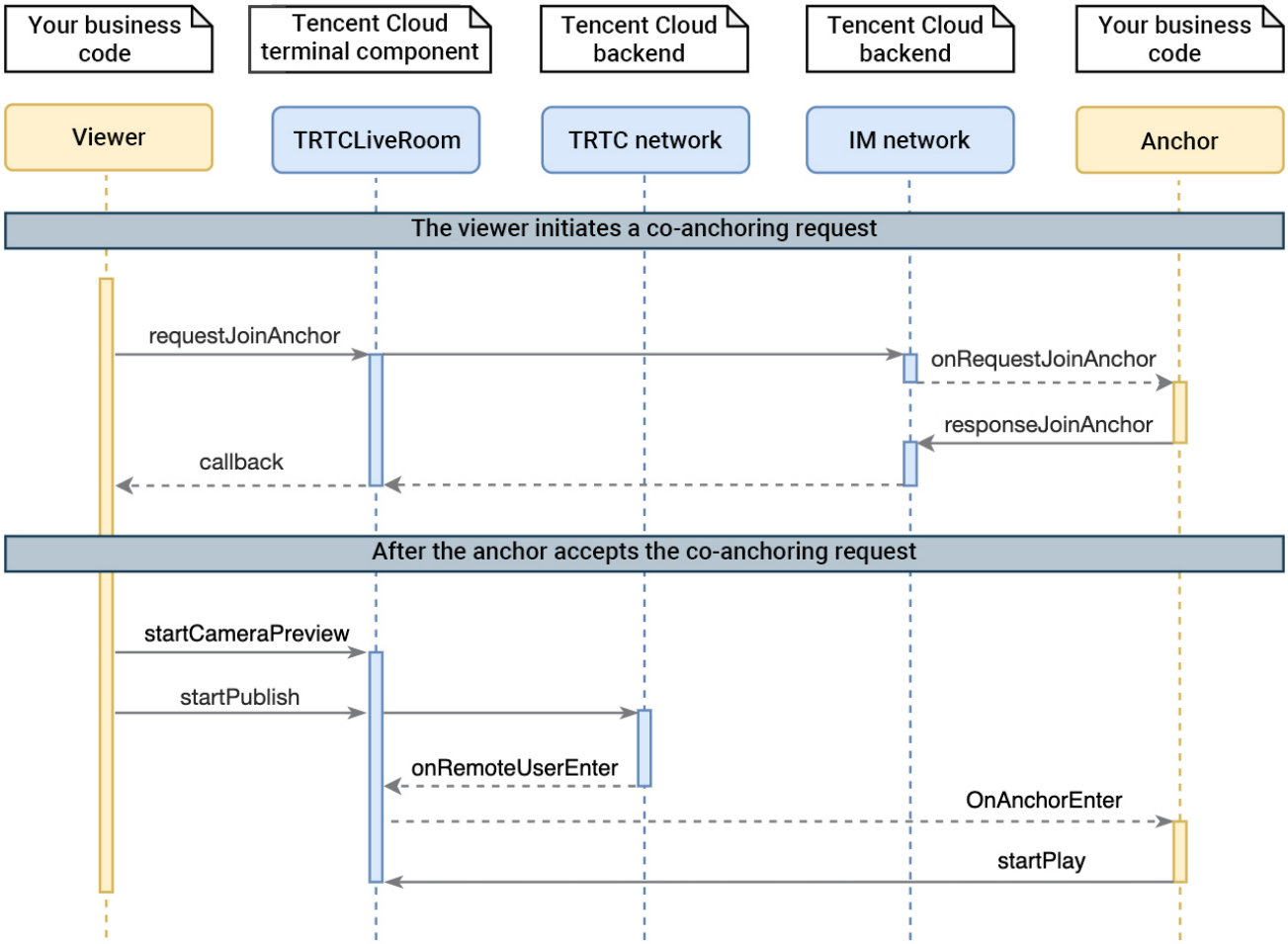



```
// 1.假定您从业务后台获取房间列表为 roomList
List<integer> roomList = GetRoomList();

// 2.通过调用 getRoomInfos 获取房间的详细信息
mLiveRoom.getRoomInfos(roomList, new TRTCLiveRoomCallback.RoomInfoCallback() {
@Override
public void onCallback(int code, String msg, List<trtcliveroomdef.trtcliveroominfo> list) {
if (code == 0) {
// 获取到房间详细信息后，您可以在主播列表页面展示主播昵称、头像等相关信息
}
}
})

// 3.选择房间 roomid 进入
mLiveRoom.enterRoom(roomid, null);
```

```
// 4.观众收到主播进房通知，开始播放
mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onAnchorEnter(final String userId) {
// 5.观众播放主播画面
mLiveRoom.startPlay(userId, mTXCloudVideoView, null);
}
});
```

步骤7：观众与主播连麦

1. 观众端调用 requestJoinAnchor 向主播端发起连麦请求。
2. 主播端会收到 TRTCLiveRoomDelegate#onRequestJoinAnchor（即有观众请求与您连麦）的事件通知。
3. 主播端可以通过调用 responseJoinAnchor 决定是否接受来自观众端的连麦请求。
4. 观众端会收到 `TRTCLiveRoomDelegate#responseCallback` 事件通知，该通知会携带来自主播端的处理结果。
5. 如果主播同意连麦请求，观众端可调用 `startCameraPreview` 开启本地摄像头，随后调用 startPublish 启动观众端的推流。
6. 主播端会在观众端启动通知后收到 `TRTCLiveRoomDelegate#onAnchorEnter` （即另一路音视频流已到来）通知，该通知会携带观众端的 userId。
7. 主播端调用 startPlay 即可看到连麦观众的画面。

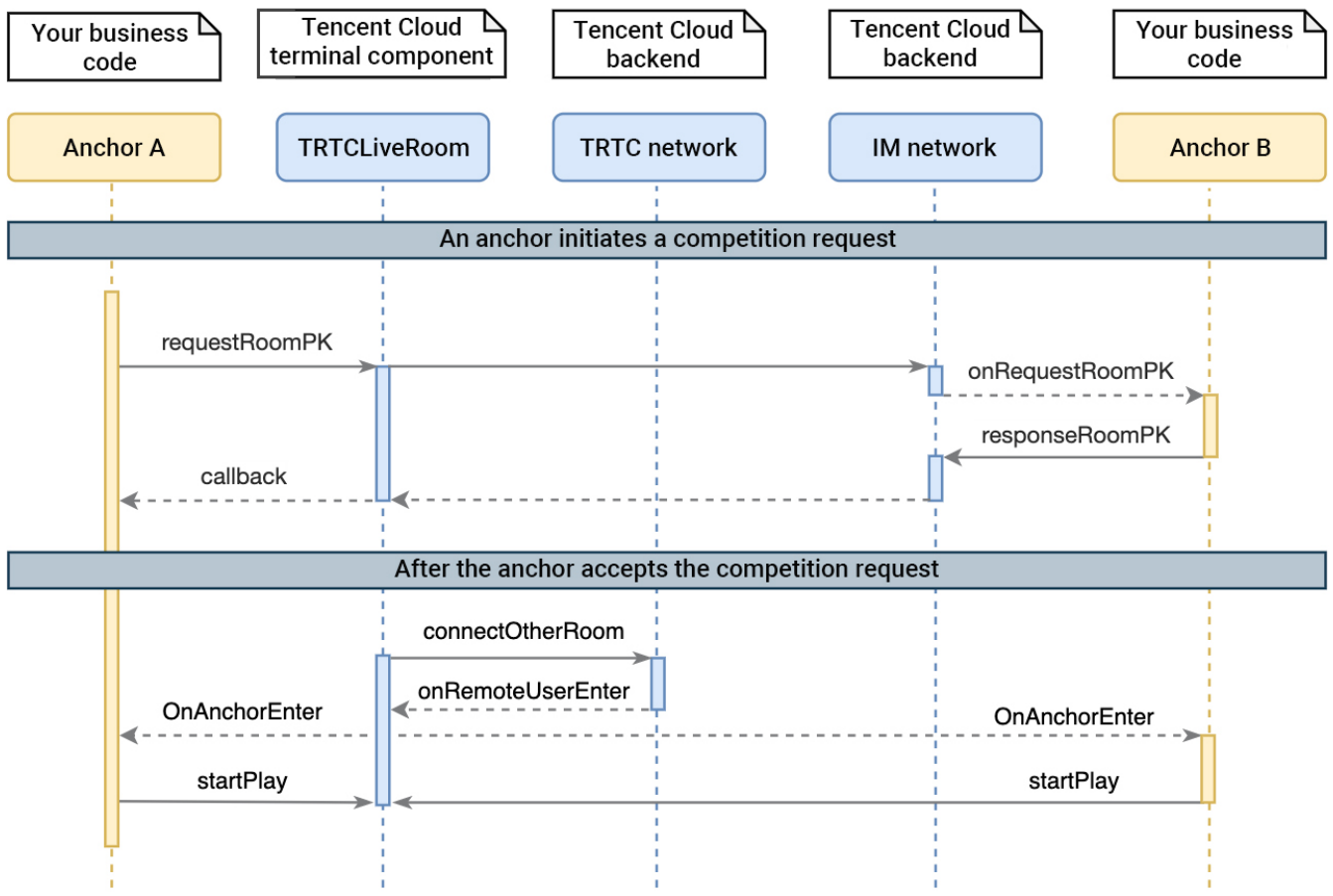



```
// 1.观众端发起连麦请求
mLiveRoom.requestJoinAnchor(mSelfUserId + "请求和您连麦",
new TRTCLiveRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
// 4.主播接受了观众的请求
TXCloudVideoView view = new TXCloudVideoView(context);
parentView.add(view);
// 5.观众启动预览，开启推流
mLiveRoom.startCameraPreview(true, view, null);
mLiveRoom.startPublish(mSelfUserId + "_stream", null);
}
}
});

// 2.主播端收到连麦请求
mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
```

```
@Override
public void onRequestJoinAnchor(final TRTCLiveRoomDef.TRTCLiveUserInfo userInfo,
String reason, final int timeout) {
// 3.同意对方的连麦请求
mLiveRoom.responseJoinAnchor(userInfo.userId, true, "同意连麦");
}

@Override
public void onAnchorEnter(final String userId) {
// 6.主播收到连麦观众的上麦通知
TXCloudVideoView view = new TXCloudVideoView(context);
parentView.add(view);
// 7.主播播放观众画面
mLiveRoom.startPlay(userId, view, null);
}
});
```

步骤8：主播与主播 PK

1. 主播 A 调用 requestRoomPK 向主播 B 发起 PK 请求。
2. 主播 B 会收到 `TRTCLiveRoomDelegate onRequestRoomPK` 回调通知。
3. 主播 B 调用 `responseRoomPK` 决定是否接受主播 A 的 PK 请求。
4. 主播 B 接受主播 A 的要求，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 A。
5. 主播 A 收到 `responseCallback` 回调通知，PK 请求是否被同意。
6. 主播 A 请求被同意，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 B。




```
// 主播 A:
// 主播 A 创建12345的房间
mLiveRoom.createRoom(12345, param, null);

mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onAnchorEnter(final String userId) {
// 6.收到 B 进房的通知
mLiveRoom.startPlay(userId, mTXCloudVideoView, null);
}
});

// 1.主播 A 向主播 B 发起 PK 请求
mLiveRoom.requestRoomPK(54321, "B",
new TRTCLiveRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
// 5.收到是否同意的回调
if (code == 0) {
// 用户接受
```

```
} else {
// 用户拒绝
}
}
});

// 主播 B：
// 主播 B 创建54321的房间
mLiveRoom.createRoom(54321, param, null);

// 2.主播 B 收到主播 A 的消息
mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onRequestRoomPK(
final TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, final int timeout) {
// 3.主播 B 回复主播 A 接受请求
mLiveRoom.responseRoomPK(userInfo.userId, true, "");
}
@Override
public void onAnchorEnter(final String userId) {
// 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
mLiveRoom.startPlay(userId, mTXCloudVideoView, null);
}
});
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
mLiveRoom.sendRoomTextMsg("Hello Word!", null);
// 接收端：监听文本消息
mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onRecvRoomTextMsg(String roomId,
String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomCustomMsg` 回调。

  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// eg:"CMD_DANMU"表示弹幕消息, "CMD_LIKE"表示点赞消息
mLiveRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mLiveRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String roomId, String cmd,
String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo) {
if ("CMD_DANMU".equals(cmd)) {
// 收到弹幕消息
Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
} else if ("CMD_LIKE".equals(cmd)) {
// 收到点赞消息
Log.d(TAG, userInfo.userName + "给您点了个赞！");
}
}
});
```

视频互动直播(Flutter)

最近更新时间：2022-03-31 11:37:48

如需快速接入视频互动直播功能，您可以直接基于我们提供的 Demo 进行修改适配，也可以使用我们提供的 TRTCLiveRoom 组件并实现自定义 UI 界面。

复用 Demo 的 UI 界面

步骤1：创建新的应用

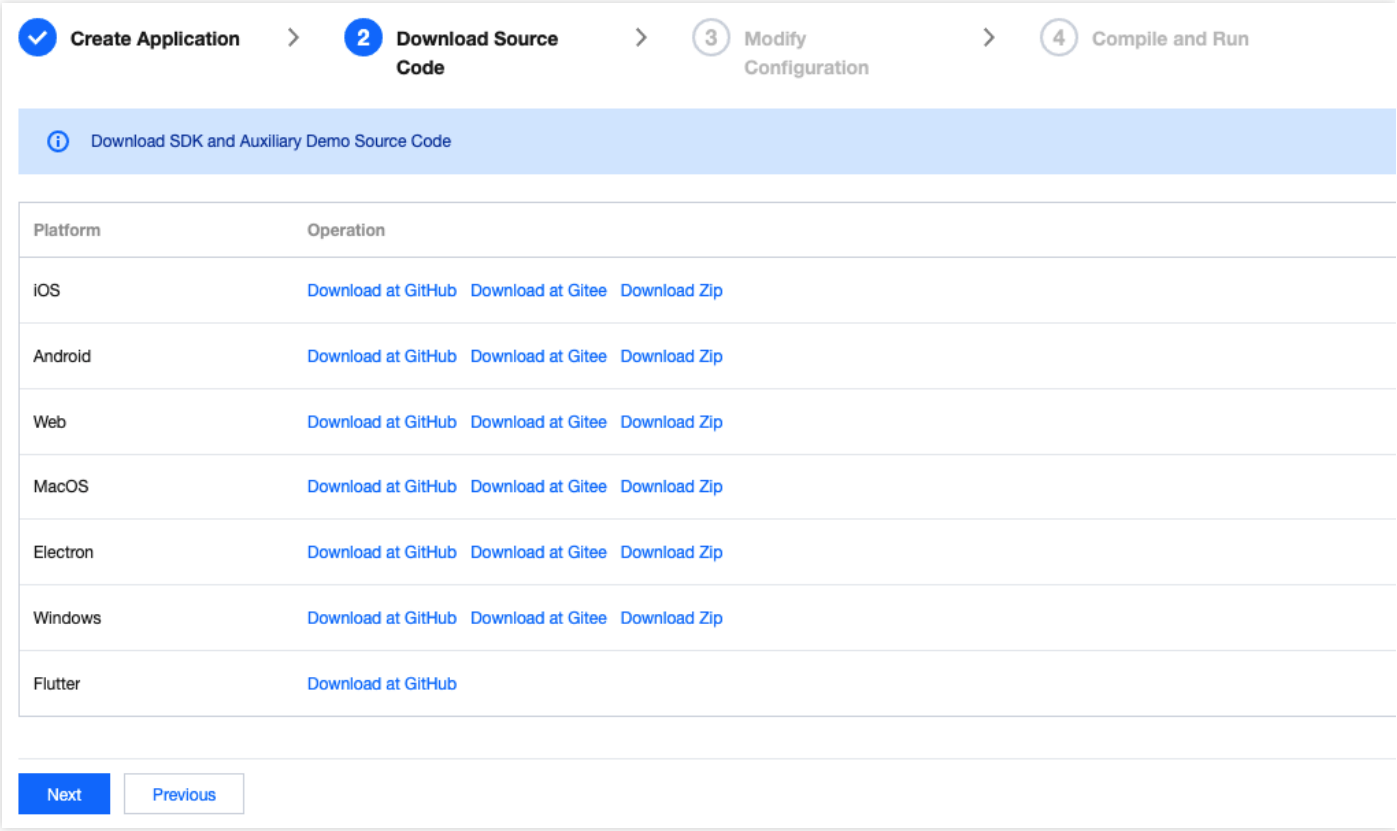
1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestLive` ，单击【创建】。

> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击【已下载，下一步】。

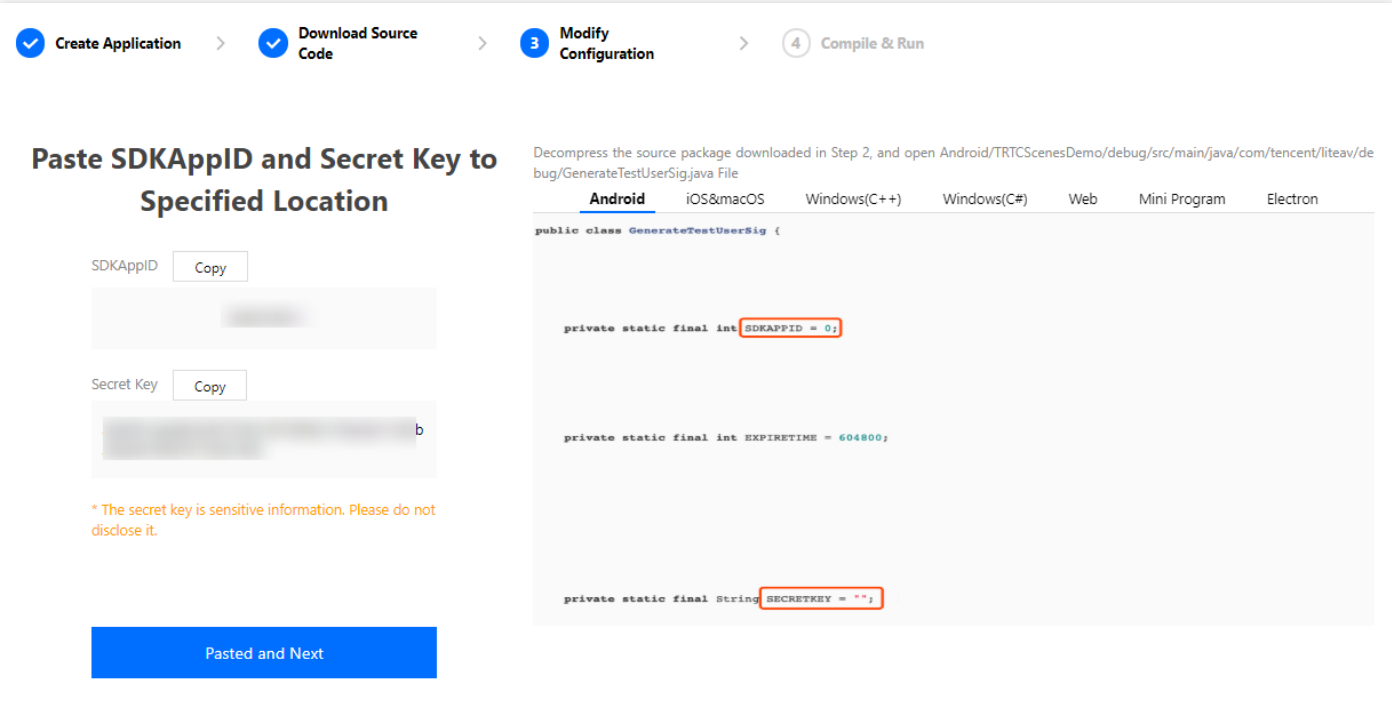


步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `/example/lib/debug/GenerateTestUserSig.dart` 文件。

3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：
   - SDKAPPID：默认为PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为PLACEHOLDER，请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：编译运行

> 注意：
> Android 端需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。

2. 编译运行调试：
   - Android 端
   - iOS 端
   i. 执行 `flutter run` 。
   ii. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击【运行】即可。

步骤5：修改 Demo 源代码

源码中的 TRTCLiveRoom 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
base	UI 使用的基础类。
list	列表页和创建房间页。
room	主房间页面，包括主播和观众两种界面。

实现自定义 UI 界面

源码 中的 TRTCLiveRoom 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCLiveRoom，您可以在 `TRTCLiveRoom.dart` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。

步骤1：集成 SDK

互动直播组件 TRTCLiveRoom 依赖 TRTC SDK 和 IM SDK，您可以通过配置 `pubspec.yaml` 自动下载更新。

在项目的 `pubspec.yaml` 中写如下依赖：

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

步骤2：配置权限及混淆规则

- iOS 端
- Android 端

需要在 `Info.plist` 中加入对相机和麦克风的权限申请：

```
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

步骤3：导入 TRTCLiveRoom 组件

拷贝以下目录中的所有文件到您的项目中：

```
lib/TRTCLiveRoom/model/
```

步骤4：创建并登录组件

1. 调用 `sharedInstance` 接口可以创建一个 TRTCLiveRoom 组件的实例对象。
2. 调用 `registerListener` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
useCDNFirst	用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。
CDNPlayDomain	在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。

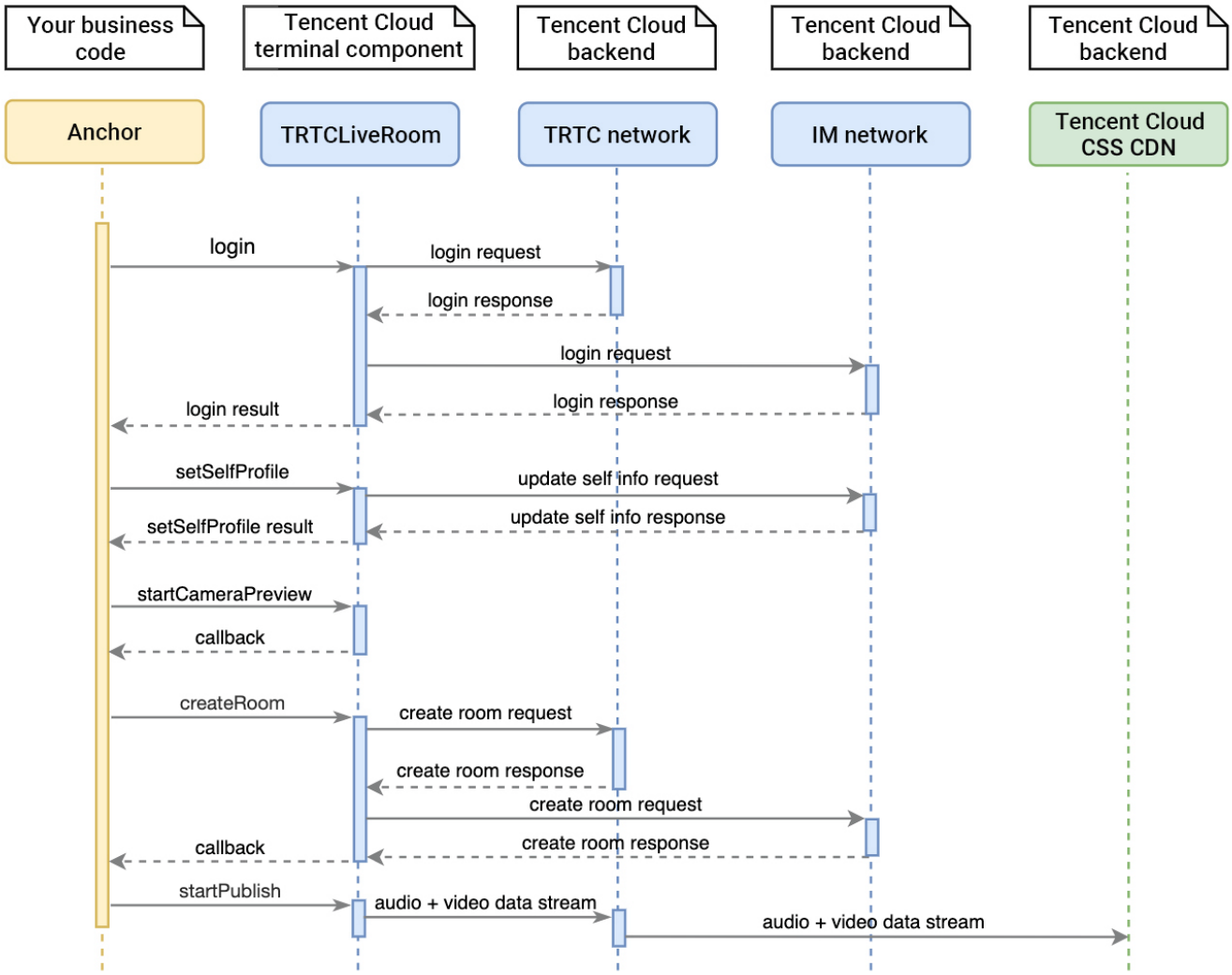
步骤5：主播端开播

1. 主播执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 主播在开播前可先调用 `startCameraPreview` 开启摄像头预览，界面上可以配置美颜调节按钮调用，通过 `getBeautyManager` 进行美颜设置。

> 说明：
> 非企业版 SDK 不支持变脸和贴图挂件功能。

3. 主播调整美颜效果后，可以调用 `createRoom` 创建新的直播间。

4. 主播调用 `startPublish` 开始推流。如需支持 CDN 观看，请在 login 时传入的 `TRTCLiveRoomConfig` 参数中指定 `useCDNFirst` 和 `CDNPlayDomain` 并在 `startPublish` 时指定直播拉流用的 streamID。



步骤6：观众端观看

1. 观众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 观众端向业务后台获取最新的直播房间列表。

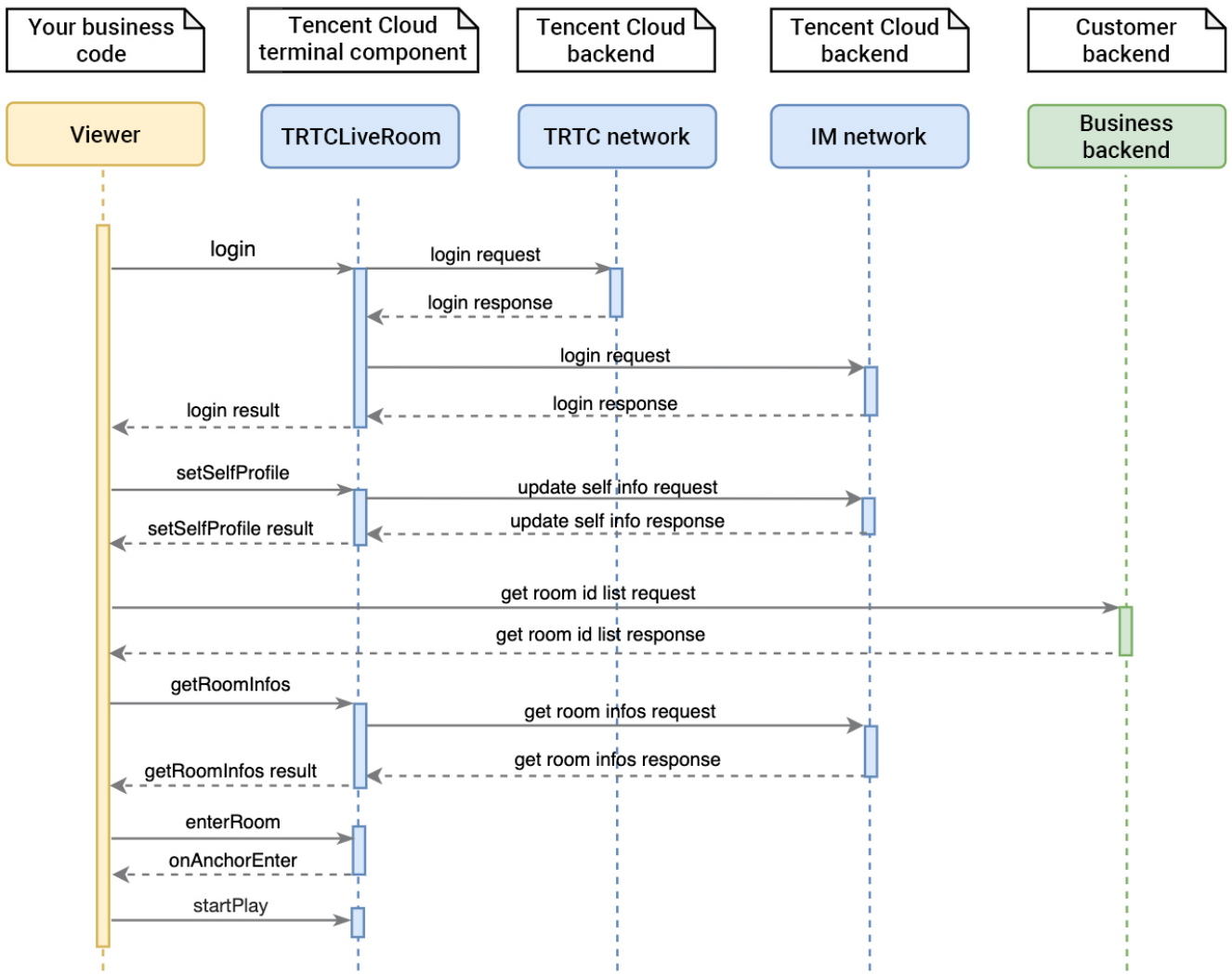
> 说明：
> Demo 中的直播间列表仅做演示使用，直播间列表的业务逻辑千差万别，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。

3. 观众端调用 `getRoomInfos` 获取房间的详细信息，该信息是在主播端调用 `createRoom` 创建直播间时设置的简单描述信息。

> 注意：
> 如果您的直播间列表包含了足够全面的信息，可跳过调用 `getRoomInfos` 相关步骤。

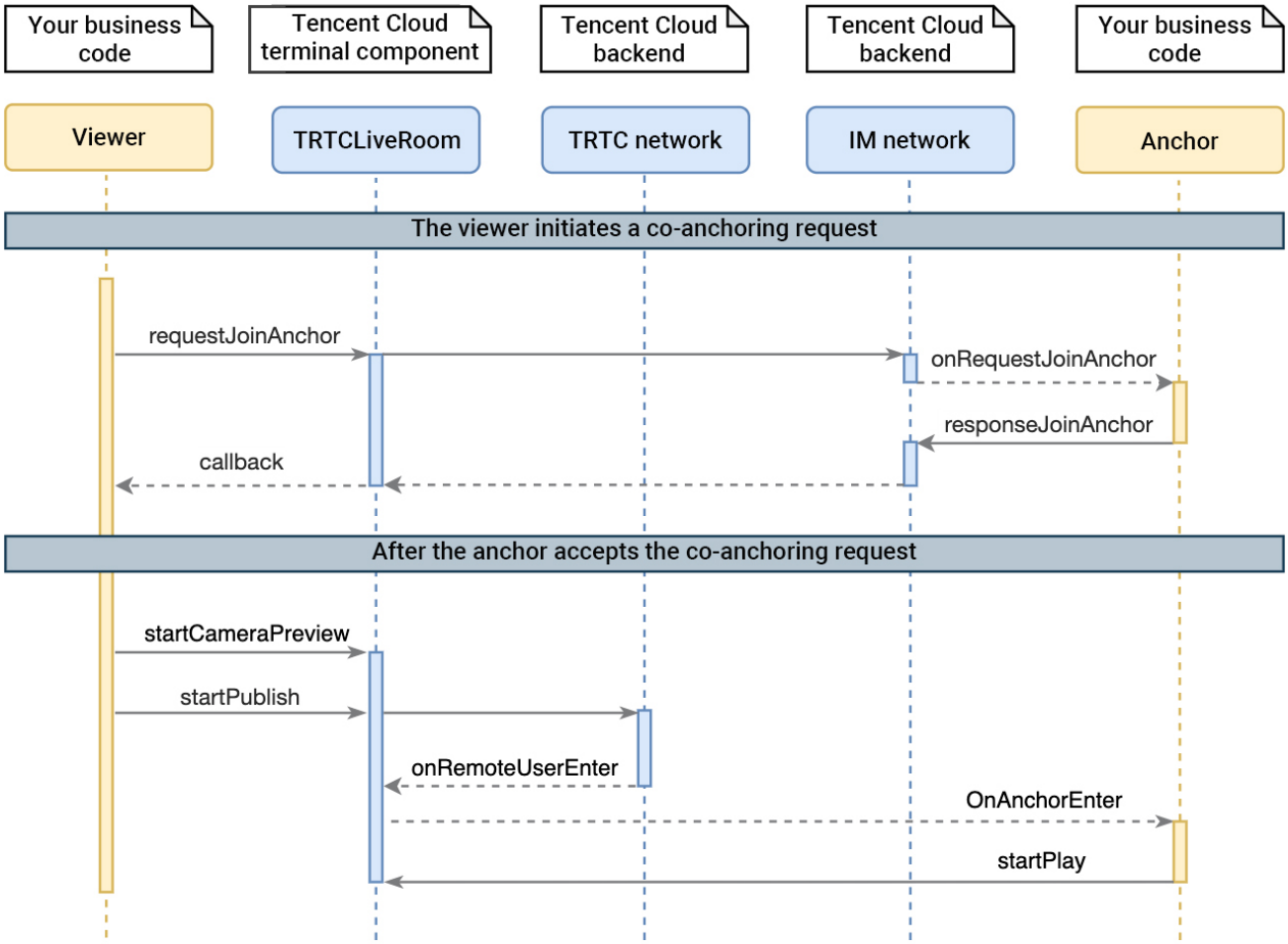
4. 观众选择一个直播间，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 调用 `startPlay` 并传入主播的 userId 开始播放。

- 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay` 并传入主播的 userId 即可开始播放。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到主播 `onAnchorEnter` 的事件回调，该回调中携带主播的 userId 信息，调用 `startPlay` 即可播放。



步骤7：观众与主播连麦上下麦

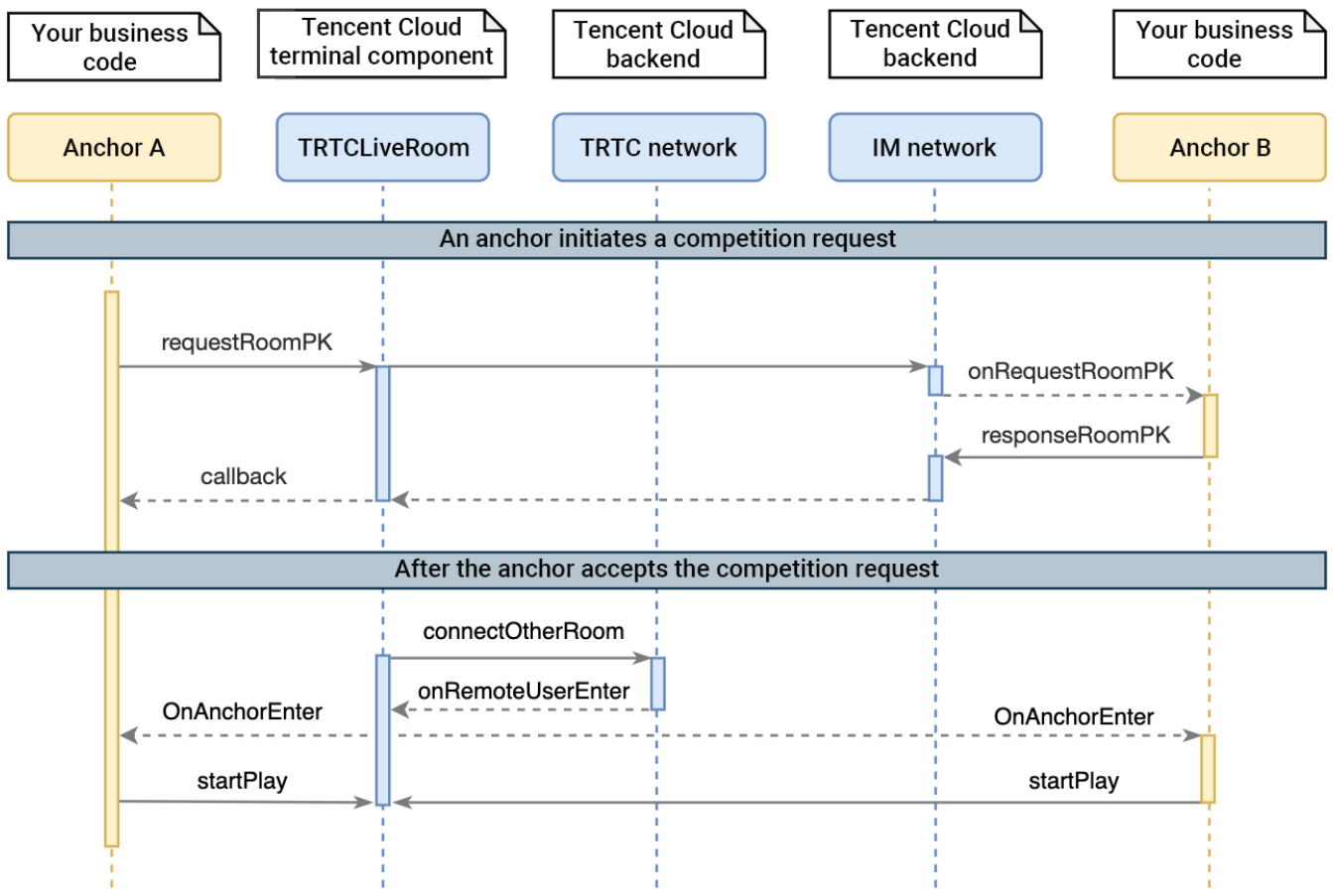
1. 观众端调用 `requestJoinAnchor` 向主播端发起连麦请求。
2. 主播端会收到 `TRTCLiveRoomDelegate#onRequestJoinAnchor` （即有观众请求与您连麦）的事件通知。
3. 主播端可以通过调用 `responseJoinAnchor` 决定是否接受来自观众端的连麦请求。
4. 观众端会收到 `onAnchorAccepted` 事件通知，该通知会携带来自主播端的处理结果。
5. 如果主播同意连麦请求，观众端可调用 `startCameraPreview` 开启本地摄像头，随后调用 `startPublish` 启动观众端的推流。
6. 主播端会在观众端启动通知后收到 `TRTCLiveRoomDelegate#onAnchorEnter` （即另一路音视频流已到来）通知，该通知会携带观众端的 userId。
7. 主播端调用 `startPlay` 即可看到连麦观众的画面。



步骤8：主播与主播 PK

1. 主播 A 调用 `requestRoomPK` 向主播 B 发起 PK 请求。
2. 主播 B 会收到 `TRTCLiveRoomDelegate onRequestRoomPK` 回调通知。
3. 主播 B 调用 `responseRoomPK` 决定是否接受主播 A 的 PK 请求。
4. 主播 B 接受主播 A 的要求，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 A。
5. 主播 A 收到 `onRoomPKAccepted` 回调通知，PK 请求是否被同意。
6. 主播 A 请求被同意，等待 `TRTCLiveRoomDelegate onAnchorEnter` 通知，调用 `startPlay` 显示主播 B。

版权所有：腾讯云计算（北京）有限责任公司



步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
trtcLiveCloud.sendRoomTextMsg("Hello Word!");


// 接收端：监听文本消息
onListenerFunc(type, params) async {
switch (type) {
case TRTCLiveRoomDelegate.onRecvRoomTextMsg:
//收到群文本消息，可以用作文本聊天室
break;
}
}
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和观众均可以收到 `onRecvRoomCustomMsg` 回调。
  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

TRTCLiveRoom API (iOS)

最近更新时间：2022-01-10 11:42:39

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
delegate	设置事件回调。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。

API	描述
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
getAudienceList	获取房间内所有的观众信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播踢除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

调试相关接口函数

API	描述
showVideoDebugLog	是否在界面中展示debug信息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	直播房间信息变更回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述
onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onKickoutJoinAnchor	连麦观众收到被踢出连麦的通知。

主播 PK 事件回调

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

delegate

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
@property(nonatomic, weak)id<TRTCLiveRoomDelegate> delegate;
```

> 说明：
> delegate 是 TRTCLiveRoom 的代理回调。

login

登录。

```
/// 登录到组件系统
/// - Parameters:
/// - sdkAppID: 您可以在实时音视频控制台 > 【[应用管理](https://console.cloud.tencent.com/trtc/app)】> 应用信息中查看 SDKAppID。
/// - userID: 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（¥_）。
/// - userSig: 腾讯云设计的一种安全保护签名，获取方式请参考 [如何计算及使用 UserSig](https://intl.cloud.tencent.com/document/product/647/35166)。
/// - config: 全局配置信息，请在登录时初始化，登录之后不可变更。 isAttachedTUIKit 项目中是否引入并使用TUIKit
/// - callback: 登录回调，成功时 code 为0。
/// - Note:
/// - userSig 建议设定 7 天，能够有效规避 usersign 过期导致的 IM 收发消息失败、TRTC 连麦失败等情况
- (void)loginWithSdkAppID:(int)sdkAppID
userID:(NSString *)userID
userSig:(NSString *)userSig
config:(TRTCLiveRoomConfig *)config
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(login(sdkAppID:userID:userSig:config:callback:));
```

参数如下表所示：

参数	类型	含义
sdkAppID	Int	您可以在实时音视频控制台 >【应用管理】> 应用信息中查看 SDKAppID。
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 • useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。
callback	(_ code: Int, _ message: String?) -> Void	登录回调，成功时 code 为0。

logout

登出。

```
// 退出登录
/// - Parameter callback: 登出回调，成功时 code 为0
- (void)logout:(Callback _Nullable)callback
NS_SWIFT_NAME(logout(_:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
/// 设置用户信息，您设置的用户信息会被存储于腾讯云 IM 云服务中。
/// - Parameters:
/// - name: 用户昵称
/// - avatarURL: 用户头像地址
/// - callback: 个人信息设置回调，成功时 code 为0
- (void)setSelfProfileWithName:(NSString *)name
avatarURL:(NSString * _Nullable)avatarURL
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(setSelfProfile(name:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
name	String	昵称。
avatarURL	String	头像地址。
callback	(_ code: Int, _ message: String?) -> Void	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
/// 创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
/// 主播开播的正常调用流程是：
/// 1.【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
/// 2.【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 callback 通知给主播。
/// 3.【主播】调用 startPublish() 开始推流。
/// - Parameters:
/// - roomID: 房间标识，需要由您分配并进行统一管理。多个 roomid 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
/// - roomParam: TRTCCreateRoomParam | 房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
/// - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播开始直播的时候调用，可重复创建自己已创建过的房间。
- (void)createRoomWithRoomID:(UInt32)roomID
roomParam:(TRTCCreateRoomParam *)roomParam
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
callback	(_ code: Int, _ message: String?) -> Void	创建房间的结果回调，成功时 code 为0。

主播开播的正常调用流程如下：

1. 【主播】调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. 【主播】调用 `createRoom()` 创建直播间，房间创建成功与否会通过 callback 通知给主播。
3. 【主播】调用 `starPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
/// 销毁房间（主播调用）
/// 主播在创建房间后，可以调用这个函数来销毁房间。
/// - Parameter callback: 销毁房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播在创建房间后，可以调用该函数来销毁房间。
- (void)destroyRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（观众调用）。

```
/// 进入房间（观众调用）
/// 观众观看直播的正常调用流程是：
/// 1.【观众】向您的服务端获取最新的直播间列表，其中有多个直播间的 roomid 和房间信息。
```

```
/// 2. 【观众】观众选择一个直播间以后，调用 enterRoom() 进入该房间。
/// 3. 【观众】如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom
() 成功后调用 startPlay(userID) 即可播放主播的画面。
/// 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自 TRTCLiveR
oomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
/// - Parameters:
/// - roomID: 房间标识。
/// - useCDNFirst: 是否优先使用CDN播放
/// - cdnDomain: CDN域名
/// - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
/// - 观众进入直播房间的时候调用
/// - 主播不可调用这个接口进入自己已创建的房间，而要用createRoom
- (void)enterRoomWithRoomID:(UInt32)roomID
useCDNFirst:(BOOL)useCDNFirst
cdnDomain:(NSString * _Nullable)cdnDomain
callback:(Callback)callback
NS_SWIFT_NAME(enterRoom(roomID:useCDNFirst:cdnDomain:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识。
callback	(_ code: Int, _ message: String?) -> Void	进入房间的结果回调，成功时 code 为0。

观众观看直播的正常调用流程如下：

1. 【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 【观众】观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. 【观众】调用 `startPlay(userID)` 并传入主播的 userID 开始播放。
   - 若直播间列表已包含主播端的 userID 信息，观众端可直接调用 `startPlay(userID)` 即可开始播放。
   - 若在进房前暂未获取主播的 userID，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 的事件回调，该回调中携带主播的 userID 信息，再调用 `startPlay(userID)` 即可播放。

exitRoom

离开房间。

```
/// 退出房间（观众调用）
/// - Parameter callback: 退出房间的结果回调，成功时 code 为0。
/// - Note:
```

```
/// - 观众离开直播房间的时候调用
/// - 主播不可调用这个接口离开房间
- (void)exitRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	退出房间的结果回调，成功时 code 为0。

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
/// 获取房间列表的详细信息
/// 其中的信息是主播在创建 createRoom() 时通过 roomInfo 设置进来的，如果房间列表和房间信息都由您自行管理，可忽略该函数。
/// - Parameter roomIDs: 房间号列表
/// - Parameter callback: 房间详细信息回调
- (void)getRoomInfosWithRoomIDs:(NSArray<NSNumber *> *)roomIDs
callback:(RoomInfoCallback _Nullable)callback
NS_SWIFT_NAME(getRoomInfos(roomIDs:callback:));
```

参数如下表所示：

参数	类型	含义
roomIDs	[UInt32]	房间号列表。
callback	(_ code: Int, _ message: String?, _ roomList: [TRTCLiveRoomInfo]) -> Void	房间详细信息回调。

getAnchorList

获取房间内所有的主播列表， `enterRoom()` 成功后调用才有效。

```
/// 获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
/// - Parameter callback: 用户详细信息回调
```

```
- (void)getAnchorList:(UserListCallback _Nullable)callback
NS_SWIFT_NAME(getAnchorList(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String, _ userList: [TRTCLiveUserInfo]) -> Void	用户详细信息回调。

getAudienceList

获取房间内所有的观众信息， `enterRoom()` 成功后调用才有效。

```
/// 获取房间内所有的观众信息，enterRoom() 成功后调用才有效。
/// - Parameter callback: 用户详细信息回调
- (void)getAudienceList:(UserListCallback _Nullable)callback
NS_SWIFT_NAME(getAudienceList(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String, _ userList: [TRTCLiveUserInfo]) -> Void	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
/// 开启本地视频的预览画面
/// - Parameters:
/// - frontCamera: true：前置摄像头；false：后置摄像头。
/// - view: 承载视频画面的控件。
/// - callback: 操作回调。
- (void)startCameraPreviewWithFrontCamera:(BOOL)frontCamera
view:(UIView *)view
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startCameraPreview(frontCamera:view:callback:));
```

参数如下表所示：

参数	类型	含义
frontCamera	Bool	true：前置摄像头；false：后置摄像头。
view	UIView	承载视频画面的控件。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

stopCameraPreview

停止本地视频采集及预览。

```
/// 停止本地视频采集及预览
- (void)stopCameraPreview;
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用
- 观众开始连麦时调用

```
/// 开始直播（推流），适用于如下两种场景：
/// 1. 主播开播的时候调用
/// 2. 观众开始连麦时调用
/// - Parameters:
/// - streamID: 用于绑定直播 CDN 的 streamId，如果您希望您的观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
/// - callback: 操作回调
- (void)startPublishWithStreamID:(NSString *)streamID
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startPublish(streamID:callback:));
```

参数如下表所示：

参数	类型	含义
streamID	String	用于绑定直播 CDN 的 streamID，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamID。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用。
- 观众结束连麦时调用。

```
/// 停止直播（推流），适用于如下两种场景：
/// 1. 主播结束直播时调用
/// 2. 观众结束连麦时调用
/// - Parameter callback: 操作回调。
- (void)stopPublish:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPublish(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
/// 播放远端视频画面，可以在普通观看和连麦场景中调用
/// 【普通观看场景】
/// 1. 如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom() 成功后调用 startPlay(userID) 即可播放主播的画面。
/// 2. 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
/// 【直播连麦场景】
/// 发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调，此时使用回调中的 userID 调用 startPlay(userID) 即可播放连麦画面。
/// - Parameters:
/// - userID: 需要观看的用户 ID。
/// - view: 承载视频画面的 view 控件。
/// - callback: 操作回调。
- (void)startPlayWithUserID:(NSString *)userID
view:(UIView *)view
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startPlay(userID:view:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userID	String	需要观看的用户 ID。
view	UIView	承载视频画面的 view 控件。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

普通观看场景

- 若直播间列表已包含主播端的 userID 信息，观众端可以直接在 `enterRoom()` 成功后调用 `startPlay(userID)` 播放主播的画面。
- 若在进房前暂未获取主播的 userID，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 的事件回调，该回调中携带主播的 userID 信息，再调用 `startPlay(userID)` 即可播放主播的画面。

直播连麦场景

发起连麦后，主播会收到来自 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 回调，此时使用回调中的 userID 调用 `startPlay(userID)` 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 `onAnchorExit()` 回调时，调用该接口。

```
/// 停止渲染远端视频画面
/// - Parameters:
/// - userID: 对方的用户信息。
/// - callback: 操作回调。
/// - Note:
/// - 在 onAnchorExit 回调时，调用这个接口
- (void)stopPlayWithUserID:(NSString *)userID
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPlay(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	对方的用户信息。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
/// 观众端请求连麦
/// - Parameters:
/// - reason: 连麦请求原因。
/// - responseCallback: 请求连麦的回调。
/// - Note: 观众发起请求后，主播端会收到`onRequestJoinAnchor`回调
- (void)requestJoinAnchor:(NSString *)reason
timeout:(double)timeout
responseCallback:(ResponseCallback _Nullable)responseCallback
NS_SWIFT_NAME(requestJoinAnchor(reason:timeout:responseCallback:));
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	long	主播响应回调。
responseCallback	(_ agreed: Bool, _ reason: String?) -> Void	主播响应回调。

主播和观众的连麦流程如下：

1. 【观众】调用 `requestJoinAnchor()` 向主播发起连麦请求。
2. 【主播】会收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调通知。
3. 【主播】调用 `responseJoinAnchor()` 决定是否接受来自观众的连麦请求。
4. 【观众】会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 【观众】如果请求被同意，则调用 `startCameraPreview()` 开启本地摄像头。
6. 【观众】调用 `startPublish()` 正式进入推流状态。
7. 【主播】一旦观众进入连麦状态，主播会收到 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知。
8. 【主播】主播调用 `startPlay()` 即可看到连麦观众的视频画面。
9. 【观众】如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 `onAnchorEnter()` 通知，调用 `startPlay()` 播放其他连麦者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调后，需要调用该接口来处理观众的连麦请求。

```
/// 主播回复观众连麦请求
/// - Parameters:
```

```
/// - user: 观众 ID。
/// - agree: true：同意；false：拒绝。
/// - reason: 同意/拒绝连麦的原因描述。
/// - Note: 主播回复后，观众端会收到`requestJoinAnchor`传入的`responseCallback`回调
- (void)responseJoinAnchor:(NSString *)userID
agree:(BOOL)agree
reason:(NSString *)reason
NS_SWIFT_NAME(responseJoinAnchor(userID:agree:reason:));
```

参数如下表所示：

参数	类型	含义
userID	String	观众 ID。
agree	Bool	true：同意；false：拒绝。
reason	String?	同意/拒绝连麦的原因描述。

kickoutJoinAnchor

主播踢除连麦观众。主播调用此接口踢除连麦观众后，被踢连麦观众会收到 `TRTCLiveRoomDelegate` 的 `onKickoutJoinAnchor()` 回调通知。

```
/// 主播踢除连麦观众
/// - Parameters:
/// - userID: 连麦观众 ID。
/// - callback: 操作回调。
/// - Note: 主播调用此接口踢除连麦观众后，被踢连麦观众会收到 trtcLiveRoomOnKickoutJoinAnchor() 回调通知
- (void)kickoutJoinAnchor:(NSString *)userID
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(kickoutJoinAnchor(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	连麦观众 ID。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
/// 主播请求跨房 PK
/// - Parameters:
/// - roomID: 被邀约房间 ID。
/// - userID: 被邀约主播 ID。
/// - responseCallback: 请求跨房 PK 的结果回调。
/// - Note: 发起请求后，对方主播会收到 `onRequestRoomPK` 回调
- (void)requestRoomPKWithRoomID:(UInt32)roomID
userID:(NSString *)userID
responseCallback:(ResponseCallback _Nullable)responseCallback
NS_SWIFT_NAME(requestRoomPK(roomID:userID:responseCallback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	被邀约房间 ID。
userID	String	被邀约主播 ID。
responseCallback	(_ agreed: Bool, _ reason: String?) -> Void	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 【主播 A】调用 `requestRoomPK()` 向主播 B 发起连麦请求。
2. 【主播 B】会收到 `TRTCLiveRoomDelegate` 的 `onRequestRoomPK()` 回调通知。
3. 【主播 B】调用 `responseRoomPK()` 决定是否接受主播 A 的 PK 请求。
4. 【主播 B】如果接受主播 A 的请求，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来显示主播 A 的视频画面。
5. 【主播 A】会收到 `responseCallback` 回调通知，该通知会携带来自主播 B 的处理结果。
6. 【主播 A】如果请求被同意，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调。

```
/// 响应跨房 PK 请求
/// 主播响应其他房间主播的 PK 请求。
/// - Parameters:
/// - user: 发起 PK 请求的主播 ID
/// - agree: true：同意；false：拒绝
/// - reason: 同意/拒绝 PK 的原因描述
/// - Note: 主播回复后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调
```

```
- (void)responseRoomPKWithUserID:(NSString *)userID
agree:(BOOL)agree
reason:(NSString *)reason
NS_SWIFT_NAME(responseRoomPK(userID:agree:reason:));
```

参数如下表所示：

参数	类型	含义
userID	String	发起 PK 请求的主播 ID。
agree	Bool	true：同意；false：拒绝。
reason	String?	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `trtcLiveRoomOnQuitRoomPK()` 回调通知。

```
/// 主播退出跨房 PK
/// - Parameter callback: 退出跨房 PK 的结果回调
/// - Note: 当两个主播中的任何一个退出跨房 PK 状态后，另一个主播会收到 `trtcLiveRoomOnQuitRoomPK`
回调通知。
- (void)quitRoomPK:(Callback _Nullable)callback
NS_SWIFT_NAME(quitRoomPK(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
/// 切换前后摄像头
- (void)switchCamera;
```

setMirror

设置是否镜像展示。

```
/// 设置是否镜像展示
/// - Parameter isMirror: 开启/关闭镜像。
- (void)setMirror:(BOOL)isMirror
NS_SWIFT_NAME(setMirror(isMirror:));
```

参数如下表所示：

参数	类型	含义
isMirror	Bool	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
/// 静音本地音频。
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteLocalAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteLocalAudio(isMuted:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
/// 静音远端音频
/// - Parameters:
/// - userID: 远端的用户ID。
/// - isMuted: true：开启静音；false：关闭静音。
- (void)muteRemoteAudioWithUserID:(NSString *)userID isMuted:(BOOL)isMuted
NS_SWIFT_NAME(muteRemoteAudio(userID:isMuted:));
```

参数如下表所示：

参数	类型	含义
userID	String	远端的用户 ID。

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
/// 静音所有远端音频
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteAllRemoteAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteAllRemoteAudio(_:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

setAudioQuality

设置音频质量

```
/// 设置音频质量，支持的值为1 2 3，代表低中高
/// - Parameter quality 音频质量
- (void)setAudioQuality:(NSInteger)quality
NS_SWIFT_NAME(setAudioiQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	1：语音；2：标准； 3：音乐。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
/// 获取音效管理对象
- (TXAudioEffectManager *)getAudioEffectManager;
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
/* 获取美颜管理对象 TXBeautyManager
*
* 通过美颜管理，您可以使用以下功能：
* - 设置"美颜风格"、"美白"、"红润"、"大眼"、"瘦脸"、"V脸"、"下巴"、"短脸"、"小鼻"、"亮眼"、"白牙"、"祛眼袋"、"祛皱纹"、"祛法令纹"等美容效果。
* - 调整"发际线"、"眼间距"、"眼角"、"嘴形"、"鼻翼"、"鼻子位置"、"嘴唇厚度"、"脸型"
* - 设置人脸挂件（素材）等动态效果
* - 添加美妆
* - 进行手势识别
*/
- (TXBeautyManager *)getBeautyManager;
```

通过美颜管理，您可以使用以下功能：

- 设置"美颜风格"、"美白"、"红润"、"大眼"、"瘦脸"、"V脸"、"下巴"、"短脸"、"小鼻"、"亮眼"、"白牙"、"祛眼袋"、"祛皱纹"、"祛法令纹"等美容效果。
- 调整"发际线"、"眼间距"、"眼角"、"嘴形"、"鼻翼"、"鼻子位置"、"嘴唇厚度"、"脸型"。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
/// 发送文本消息，房间内所有成员都可见
/// - Parameters:
/// - message: 文本消息。
/// - callback: 发送回调。
- (void)sendRoomTextMsg:(NSString *)message callback:(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
/// 发送自定义消息
/// - Parameters:
/// - command: 命令字，由开发者自定义，主要用于区分不同消息类型
/// - message: 本文消息。
/// - callback: 发送回调。
- (void)sendRoomCustomMsgWithCommand:(NSString *)command message:(NSString *)message callback:(Ca
llback _Nullable)callback
NS_SWIFT_NAME(sendRoomCustomMsg(command:message:callback:));
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示 debug 信息。

```
/// 是否在界面中展示debug信息
/// - Parameter isShow: 开启/关闭 Debug 信息显示。
- (void)showVideoDebugLog:(BOOL)isShow
NS_SWIFT_NAME(showVideoDebugLog(_:));
```

参数如下表所示：

参数	类型	含义
isShow	Bool	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onError:(NSInteger)code
message:(NSString *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onError:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码。
message	String?	错误信息。

onWarning

警告回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onWarning:(NSInteger)code
message:(NSString *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onWarning:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码 TRTCWarningCode。
message	String?	警告信息。

onDebugLog

Log 回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onDebugLog:(NSString *)log
NS_SWIFT_NAME(trtcLiveRoom(_:onDebugLog:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
log	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRoomDestroy:(NSString *)roomID
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomDestroy:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
roomID	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRoomInfoChange:(TRTCLiveRoomInfo *)info
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomInfoChange:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
info	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAnchorEnter:(NSString *)userID
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	新进房用户 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAnchorExit:(NSString *)userID
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAudienceEnter:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAudienceExit:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRequestJoinAnchor:(TRTCLiveUserInfo *)user
reason:(NSString * _Nullable)reason
timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestJoinAnchor:reason:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	请求连麦观众信息。
reason	String?	连麦原因描述。
timeout	Double	处理请求的超时时间。

onKickoutJoinAnchor

连麦观众收到被踢出连麦的通知。连麦观众收到被主播踢除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

```
- (void)trtcLiveRoomOnKickoutJoinAnchor:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnKickoutJoinAnchor(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRequestRoomPK:(TRTCLiveUserInfo *)user
timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestRoomPK:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	发起跨房连麦的主播信息。
timeout	Double	处理请求的超时时间。

onQuitRoomPK

收到断开跨房 PK 通知。

```
- (void)trtcLiveRoomOnQuitRoomPK:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnQuitRoomPK(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRecvRoomTextMsg:(NSString *)message
fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomTextMsg:fromUser:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRecvRoomCustomMsgWithCommand:(NSString *)command
message:(NSString *)message
fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomCustomMsg:message:fromUser:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

版权所有：腾讯云计算（北京）有限责任公司

TRTCLiveRoom API (Android)

最近更新时间：2022-04-21 16:51:12

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。

API	描述
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
getAudienceList	获取房间内所有的观众信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播踢除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。

API	描述
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

调试相关接口函数

API	描述

API	描述
showVideoDebugLog	是否在界面中展示 debug 信息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	直播房间信息变更回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述
onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onKickoutJoinAnchor	连麦观众收到被踢出连麦的通知。

版权所有：腾讯云计算（北京）有限责任公司

主播 PK 事件回调

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom单例对象。

```
public static synchronized TRTCLiveRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCLiveRoomDelegate delegate);
```

说明：
setDelegate 是 TRTCLiveRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCLiveRoom 中的各种状态通知回调会通过该 handler 通知给您，请勿与 setDelegate 混用。

login

登录。

```
public abstract void login(int sdkAppId,
String userId, String userSig,
TRTCLiveRoomDef.TRTCLiveRoomConfig config,
TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

参数	类型	含义
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 • useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCLiveRoomCallback.Actio
nCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
public abstract void createRoom(int roomId, TRTCLiveRoomDef.TRTCCreateRoomParam roomParam, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

主播开播的正常调用流程如下：

1. 【主播】调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. 【主播】调用 `createRoom()` 创建直播间，房间创建成功与否会通过 ActionCallback 通知给主播。
3. 【主播】调用 `starPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
public abstract void destroyRoom(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（观众调用）。

```
public abstract void enterRoom(int roomId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

观众观看直播的正常调用流程如下：

1. 【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 【观众】观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. 【观众】调用 `startPlay(userId)` 并传入主播的 userId 开始播放。

- 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay(userId)` 即可开始播放。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放。

exitRoom

离开房间。

```
public abstract void exitRoom(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfos(List<Integer> roomIdList, TRTCLiveRoomCallback.RoomInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getAnchorList

获取房间内所有的主播列表， `enterRoom()` 成功后调用才有效。

```
public abstract void getAnchorList(TRTCLiveRoomCallback.UserListCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

getAudienceList

获取房间内所有的观众信息， `enterRoom()` 成功后调用才有效。

```
public abstract void getAudienceList(TRTCLiveRoomCallback.UserListCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
public abstract void startCameraPreview(boolean isFront, TXCloudVideoView view, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。
view	TXCloudVideoView	承载视频画面的控件。
callback	ActionCallback	操作回调。

stopCameraPreview

停止本地视频采集及预览。

```
public abstract void stopCameraPreview();
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用
- 观众开始连麦时调用

```
public abstract void startPublish(String streamId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
streamId	String	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
callback	ActionCallback	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用
- 观众结束连麦时调用

```
public abstract void stopPublish(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
public abstract void startPlay(String userId, TXCloudVideoView view, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要观看的用户id。
view	TXCloudVideoView	承载视频画面的 view 控件。
callback	ActionCallback	操作回调。

普通观看场景

- 若直播间列表已包含主播端的 userId 信息，观众端可以直接在 `enterRoom()` 成功后调用 `startPlay(userId)` 播放主播的画面。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放主播的画面。

直播连麦场景

发起连麦后，主播会收到来自 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 回调，此时使用回调中的 userId 调用 startPlay(userId) 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 `onAnchorExit()` 回调时，调用该接口。

```
public abstract void stopPlay(String userId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	对方的用户信息。

参数	类型	含义
callback	ActionCallback	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
public abstract void requestJoinAnchor(String reason, int timeout, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	int	超时时间。
callback	ActionCallback	主播响应回调。

主播和观众的连麦流程如下：

1. 【观众】调用 `requestJoinAnchor()` 向主播发起连麦请求。
2. 【主播】会收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调通知。
3. 【主播】调用 `responseJoinAnchor()` 决定是否接受来自观众的连麦请求。
4. 【观众】会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 【观众】如果请求被同意，则调用 `startCameraPreview()` 开启本地摄像头。
6. 【观众】然后调用 `startPublish()` 正式进入推流状态。
7. 【主播】一旦观众进入连麦状态，主播会收到 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知。
8. 【主播】主播调用 `startPlay()` 即可看到连麦观众的视频画面。
9. 【观众】如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 `onAnchorEnter()` 通知，调用 `startPlay()` 播放其他连麦者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调后需要调用此接口来处理观众的连麦请求。

```
public abstract void responseJoinAnchor(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	观众 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝连麦的原因描述。

kickoutJoinAnchor

主播踢除连麦观众。主播调用此接口踢除连麦观众后，被踢连麦观众会收到 `TRTCLiveRoomDelegate` 的 `onKickoutJoinAnchor()` 回调通知。

```
public abstract void kickoutJoinAnchor(String userId, TRTCLiveRoomCallback.ActionCallback callbac
k);
```

参数如下表所示：

参数	类型	含义
userId	String	连麦观众 ID。
callback	ActionCallback	操作回调。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
public abstract void requestRoomPK(int roomId, String userId, TRTCLiveRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。
callback	ActionCallback	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 【主播 A】调用 `requestRoomPK()` 向主播 B 发起连麦请求。
2. 【主播 B】会收到 `TRTCLiveRoomDelegate` 的 `onRequestRoomPK()` 回调通知。
3. 【主播 B】调用 `responseRoomPK()` 决定是否接受主播 A 的 PK 请求。
4. 【主播 B】如果接受主播 A 的要求，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来显示主播 A 的视频画面。
5. 【主播 A】会收到 `responseCallback` 回调通知，该通知会携带来自主播 B 的处理结果。
6. 【主播 A】如果请求被同意，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调。

```
public abstract void responseRoomPK(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `onQuitRoomPk()` 回调通知。

```
public abstract void quitRoomPK(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
public abstract void switchCamera();
```

setMirror

设置是否镜像展示。

```
public abstract void setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	boolean	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

版权所有：腾讯云计算（北京）有限责任公司

muteAllRemoteAudio

静音所有远端音频。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
public abstract TXBeautyManager getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCLiveRoomCallback.ActionCallback callbac
k);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCLiveRoomCallback.ActionCal
lback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示debug信息。

```
public abstract void showVideoDebugLog(boolean isShow);
```

参数如下表所示：

参数	类型	含义
isShow	boolean	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
void onRoomInfoChange(TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

```
void onAnchorEnter(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

```
void onAnchorExit(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
void onAudienceExit(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

```
void onRequestJoinAnchor(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, String reason, int timeOut);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	请求连麦观众信息。
reason	String	连麦原因描述。
timeout	int	处理请求的超时时间，如果上层超过该时间没有处理，则会自动将该次请求废弃。

onKickoutJoinAnchor

连麦观众收到被踢出连麦的通知。连麦观众收到被主播踢除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

```
void onKickoutJoinAnchor();
```

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK ，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。

```
void onRequestRoomPK(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, int timeout);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	发起跨房连麦的主播信息。
timeout	int	处理请求的超时时间。

onQuitRoomPK

收到断开跨房 PK 通知。

```
void onQuitRoomPK();
```

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	TRTCLiveUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	TRTCLiveUserInfo	发送者用户信息。

TRTCAudioEffectManager

playBGM

播放背景音乐。

```
void playBGM(String url, int loopTimes, int bgmVol, int micVol, TRTCCloud.BGMNotify notify);
```

参数如下表所示：

参数	类型	含义
url	String	背景音乐文件路径。
loopTimes	int	循环次数
bgmVol	int	BGM 音量
micVol	int	采集音量
notify	TRTCCloud.BGMNotify	播放通知

stopBGM

停止播放背景音乐。

```
void stopBGM();
```

pauseBGM

暂停播放背景音乐。

```
void pauseBGM();
```

resumeBGM

继续播放背景音乐。

```
void resumeBGM();
```

setBGMVolume

设置背景音乐的音量大小，播放背景音乐混音时使用，用来控制背景音的音量大小。

```
void setBGMVolume(int volume);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
volume	int	音量大小，100表示正常音量，取值范围为0 - 100，默认值为100。

setBGMPosition

设置背景音乐播放进度。

```
int setBGMPosition(int position);
```

参数如下表所示：

参数	类型	含义
position	int	背景音乐播放进度，单位为毫秒（ms）。

返回

0：成功。

setMicVolume

设置麦克风的音量大小，播放背景音乐混音时使用，用来控制麦克风音量大小。

```
void setMicVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	Int	音量大小，取值0 - 100，默认值为100。

setReverbType

设置混响效果。

```
void setReverbType(int reverbType);
```

参数如下表所示：

参数	类型	含义
reverbType	int	混响类型，详情请参见 `TRTCCloudDef` 中的 TRTC_REVERB_TYPE 定义。

setVoiceChangerType

设置变声类型。

```
void setVoiceChangerType(int type);
```

参数如下表所示：

参数	类型	含义
type	int	混响类型，详情请参见 `TRTCCloudDef` 中的 TRTC_VOICE_CHANGER_TYPE 定义。

playAudioEffect

播放音效，每个音效都需要您指定具体的 ID，您可以通过该 ID 对音效的开始、停止、音量等进行设置。支持 aac、mp3 以及 m4a 格式。

```
void playAudioEffect(int effectId, String path, int count, boolean publish, int volume);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。
path	String	音效路径。
count	int	循环次数。
publish	boolean	是否推送 / true 推送给观众, false 本地预览。
volume	int	音量大小，取值范围为0 - 100，默认值为100。

pauseAudioEffect

暂停音效播放。

```
void pauseAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

resumeAudioEffect

恢复音效播放。

```
void resumeAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

stopAudioEffect

停止音效播放。

```
void stopAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

stopAllAudioEffects

停止全部音效播放。

```
void stopAllAudioEffects();
```

setAudioEffectVolume

设置音效音量。

```
void setAudioEffectVolume(int effectId, int volume);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。
volume	int	音量大小，取值范围为0 - 100，默认值为100。

版权所有：腾讯云计算（北京）有限责任公司

setAllAudioEffectsVolume

设置所有音效的音量。

```
void setAllAudioEffectsVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	音量大小，取值范围为0 - 100，默认值为100。

TRTCLiveRoom API (Flutter)

最近更新时间：2022-01-10 11:45:59

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
registerListener	设置事件回调。
unRegisterListener	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。

API	描述
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
getRoomMemberList	获取房间内所有的成员信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播踢除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。

API	描述
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onKickedOffline	其他用户登录了同一账号，被踢下线。

房间事件回调

API	描述
onEnterRoom	本地进房回调。
onUserVideoAvailable	远端用户是否存在可播放的主路画面（一般用于摄像头）。
onRoomDestroy	房间被销毁的回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述
onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onAnchorAccepted	主播同意观众的连麦请求。
onAnchorRejected	主播拒绝观众的连麦请求。
onKickoutJoinAnchor	连麦观众收到被踢出连麦的通知。

主播 PK 事件回调

API	描述

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onRoomPKAccepted	主播接受跨房 PK 请求。
onRoomPKRejected	主播拒绝跨房 PK 请求。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom 单例对象。

```
static Future<TRTCLiveRoom> sharedInstance()
```

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
static void destroySharedInstance()
```

registerListener

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
void registerListener(VoiceListenerFunc func);
```

说明：
registerListener 是 TRTCLiveRoom 的代理回调。

unRegisterListener

移除组件事件监听接口。

```
void unRegisterListener(VoiceListenerFunc func);
```

login

登录。

```
Future<ActionCallback> login(
int sdkAppId, String userId, String userSig, TRTCLiveRoomConfig config);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 >【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 • useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。

logout

登出。

```
Future<ActionCallback> logout();
```

setSelfProfile

修改个人信息。

```
Future<ActionCallback> setSelfProfile(String userName, String avatarURL);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
Future<ActionCallback> createRoom(int roomId, TRTCCreateRoomParam roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	RoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。

主播开播的正常调用流程如下：

1. 【主播】调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. 【主播】调用 `createRoom()` 创建直播间，房间创建成功与否会通过 ActionCallback 通知给主播。
3. 【主播】调用 `starPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
Future<ActionCallback> destroyRoom();
```

enterRoom

进入房间（观众调用）。

```
Future<ActionCallback> enterRoom(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。

观众观看直播的正常调用流程如下：

1. 【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 【观众】观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. 【观众】调用 `startPlay(userId)` 并传入主播的 userId 开始播放。
   - 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay(userId)` 即可开始播放。
   - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放。

exitRoom

离开房间。

```
Future<ActionCallback> exitRoom();
```

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
Future<RoomInfoCallback> getRoomInfos(List<String> roomIdList);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<String>	房间号列表。

getAnchorList

获取房间内所有的主播列表， `enterRoom()` 成功后调用才有效。

```
Future<UserListCallback> getAnchorList();
```

getRoomMemberList

获取房间内所有的观众信息， `enterRoom()` 成功后调用才有效。

```
Future<UserListCallback> getRoomMemberList(int nextSeq)
```

参数如下表所示：

参数	类型	含义
nextSeq	int	分页拉取标志，第一次拉取填0，回调成功如果 nextSeq 不为零，需要分页，传入再次拉取，直至为0。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
Future<void> startCameraPreview(bool isFrontCamera, int viewId);
```

参数如下表所示：

参数	类型	含义
isFrontCamera	bool	true：前置摄像头；false：后置摄像头。
viewId	int	视频 view 的回调 ID。

stopCameraPreview

停止本地视频采集及预览。

```
Future<void> stopCameraPreview();
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用。
- 观众开始连麦时调用。

```
Future<void> startPublish(String streamId);
```

参数如下表所示：

参数	类型	含义
streamId	String?	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用。
- 观众结束连麦时调用。

```
Future<void> stopPublish();
```

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
Future<void> startPlay(String userId, int viewId);
```

参数如下表所示：

参数	类型	含义
userId	String	需要观看的用户id。
viewId	int	视频view的回调id。

- **普通观看场景：**
  - 若直播间列表已包含主播端的 userId 信息，观众端可以直接在 `enterRoom()` 成功后调用 `startPlay(userId)` 播放主播的画面。
  - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放主播的画面。

- **直播连麦场景：**
  发起连麦后，主播会收到来自 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 回调，此时使用回调中的 userId 调用 `startPlay(userId)` 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 `onAnchorExit()` 回调时，调用该接口。

```
Future<void> stopPlay(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	对方的用户信息。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
Future<ActionCallback> requestJoinAnchor();
```

主播和观众的连麦流程如下：

1. 【观众】调用 `requestJoinAnchor()` 向主播发起连麦请求。
2. 【主播】会收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调通知。
3. 【主播】调用 `responseJoinAnchor()` 决定是否接受来自观众的连麦请求。
4. 【观众】会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 【观众】如果请求被同意，则调用 `startCameraPreview()` 开启本地摄像头。
6. 【观众】然后调用 `startPublish()` 正式进入推流状态。
7. 【主播】一旦观众进入连麦状态，主播会收到 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知。
8. 【主播】主播调用 `startPlay()` 即可看到连麦观众的视频画面。

9. 【观众】如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 `onAnchorEnter()` 通知，调用 `startPlay()` 播放其他连麦者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调后需要调用此接口来处理观众的连麦请求。

```
Future<ActionCallback> responseJoinAnchor(String userId, boolean agreee);
```

参数如下表所示：

参数	类型	含义
userId	String	观众 ID。
agree	bool	true：同意；false：拒绝。

kickoutJoinAnchor

主播踢除连麦观众。主播调用此接口踢除连麦观众后，被踢连麦观众会收到 `TRTCLiveRoomDelegate` 的 `onKickoutJoinAnchor()` 回调通知。

```
Future<ActionCallback> kickoutJoinAnchor(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	连麦观众 ID。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
Future<ActionCallback> requestRoomPK(int roomId, String userId);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 【主播 A】调用 `requestRoomPK()` 向主播 B 发起连麦请求。
2. 【主播 B】会收到 `TRTCLiveRoomDelegate` 的 `onRequestRoomPK()` 回调通知。
3. 【主播 B】调用 `responseRoomPK()` 决定是否接受主播 A 的 PK 请求。
4. 【主播 B】如果接受主播 A 的要求，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来显示主播 A 的视频画面。
5. 【主播 A】会收到 `onRoomPKAccepted` 或 `onRoomPKRejected` 回调通知。
6. 【主播 A】如果请求被同意，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调。

```
Future<ActionCallback> responseRoomPK(String userId, boolean agree);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	bool	true：同意；false：拒绝。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `onQuitRoomPk()` 回调通知。

```
Future<ActionCallback> quitRoomPK();
```

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
Future<void> switchCamera(boolean isFrontCamera);
```

setMirror

设置是否镜像展示。

```
Future<void> setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	bool	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
Future<void> muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
Future<void> muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
Future<void> muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
Future<ActionCallback> sendRoomTextMsg(String message);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。

sendRoomCustomMsg

发送自定义文本消息。

```
Future<ActionCallback> sendRoomCustomMsg(String cmd, String message);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

参数如下表所示：

参数	类型	含义
errCode	int	错误码。

参数	类型	含义
errMsg	String	错误信息。

onWarning

警告回调。

参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。

onKickedOffline

其他用户登录了同一账号，被踢下线。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

onEnterRoom

本地进房

参数如下表所示：

参数	类型	含义
result	int	result > 0 时为进房耗时（ms），result < 0 时为进房错误码。

onUserVideoAvailable

远端用户是否存在可播放的主路画面（一般用于摄像头）。

参数如下表所示：

参数	类型	含义
userId	String	用户标识。

参数	类型	含义
available	boolean	画面是否开启。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

参数如下表所示：

参数	类型	含义
userId	String	退出主播 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAudienceEnter

收到观众进房通知。

```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveRoomDef.TRTCLiveUserInfo	进房观众用户 ID、昵称、头像等信息。

onAudienceExit

收到观众退房通知。

参数如下表所示：

参数	类型	含义
userId	String	退房观众信息。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

参数如下表所示：

参数	类型	含义
userId	String	请求连麦用户 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAnchorAccepted

主播同意观众的连麦请求。

参数如下表所示：

参数	类型	含义
userId	String	主播的用户 ID。

onAnchorRejected

主播拒绝观众的连麦请求。

参数如下表所示：

参数	类型	含义
userId	String	主播的用户 ID。

onKickoutJoinAnchor

连麦观众收到被踢出连麦的通知。连麦观众收到被主播踢除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK ，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。

参数如下表所示：

参数	类型	含义
userId	String	请求跨房用户 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onRoomPKAccepted

主播接受跨房 PK 请求。

参数如下表所示：

参数	类型	含义
userId	String	接收跨房 PK 的用户 ID。

onRoomPKRejected

主播接受跨房 PK 请求。

参数如下表所示：

参数	类型	含义
userId	String	拒绝跨房 PK 的用户 ID。

版权所有：腾讯云计算（北京）有限责任公司

onQuitRoomPK

收到断开跨房 PK 通知。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

参数如下表所示：

参数	类型	含义
message	String	文本消息。

onRecvRoomCustomMsg

收到自定义消息。

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

语音聊天室
语音聊天室(iOS)

最近更新时间：2022-03-09 17:59:05

效果展示

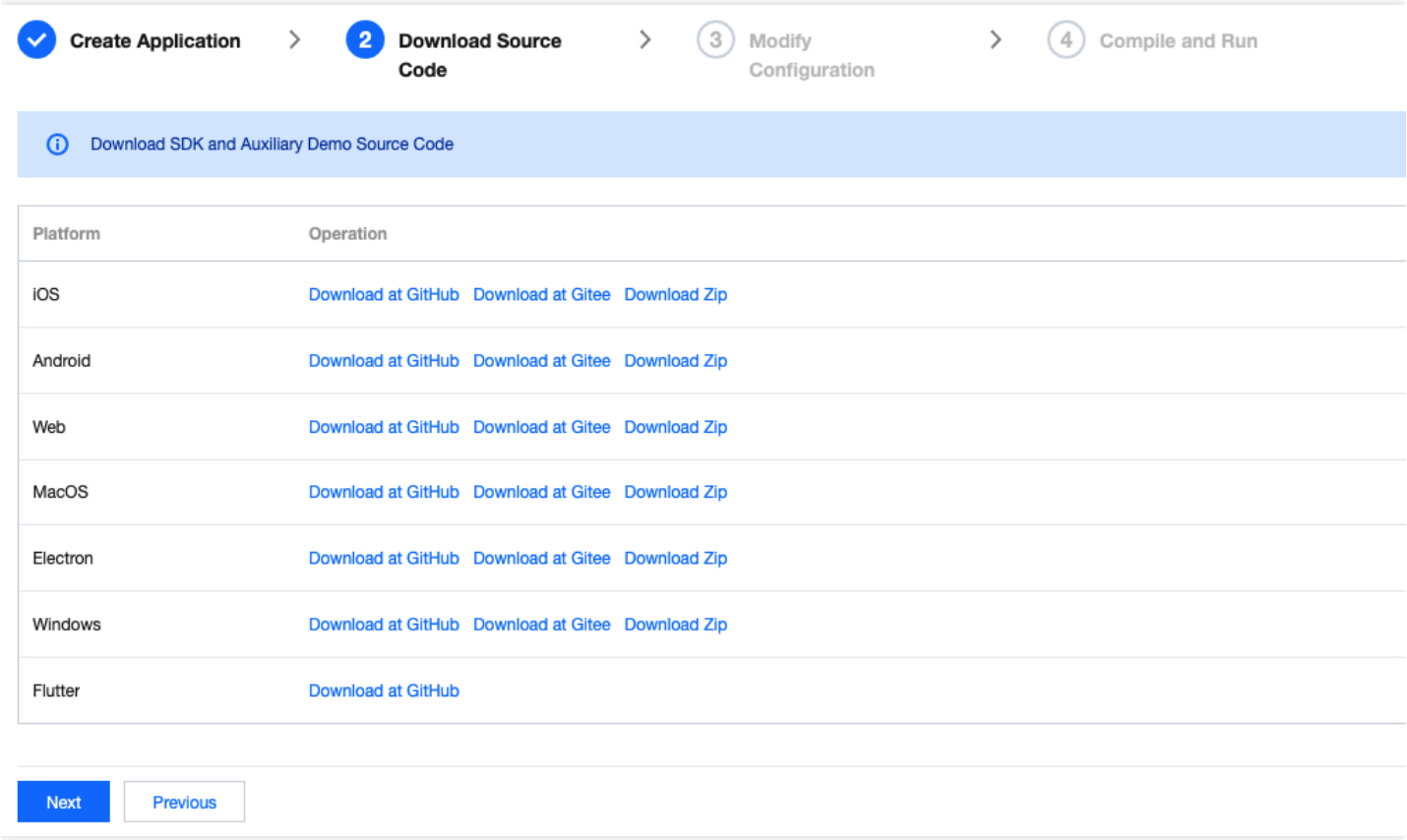
您可以 下载 安装我们的 App 体验语音聊天室的能力，包括麦位管理、低延时语音互动、文字聊天等 TRTC 在语音聊天场景下的相关能力。

如需快速接入语音聊天室功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TUIVoiceRoom 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
2. 输入应用名称，例如 `TestVoiceRoom` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



说明：
本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PAAS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

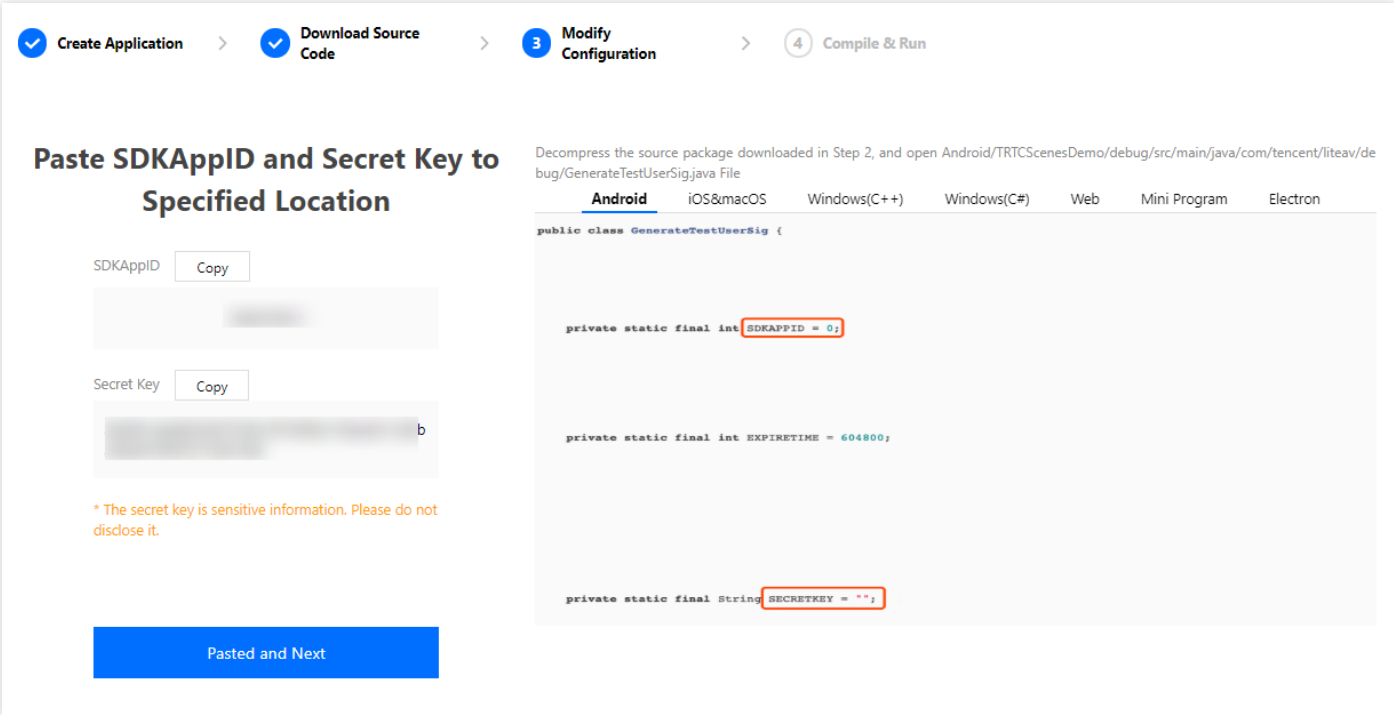
单击进入 TUIVoiceRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `TUIVoiceRoom/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Xcode（11.0及以上的版本）打开源码工程 `TUIVoiceRoom/TUIVoiceRoomApp.xcworkspace`，单击 **运行** 即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 `Source` 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码以及涉及界面相关的逻辑，如下表格列出了各个 swift 文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
TRTCVoiceRoomEnteryControl.swift	该文件包含所有 ViewController 的初始化获取方法，您可以通过该实例，快速获取 ViewController 对象。
TRTCCreateVoiceRoomViewController	创建语音聊天室页面逻辑。
TRTCVoiceRoomViewController	主房间页面，包括房主和听众两种界面。

每个 `TRTC'XXXX'ViewController` 文件夹下均包含 `ViewController` 、 `RootView` 和 `ViewModel` ，各个文件的作用如下表所示：

文件	功能描述
ViewController.swift	页面控制器，负责页面路由工作，以及 RootView 和 ViweModel 的绑定工作。
RootView.swift	视图，所有的视图布局。
ViewModel.swift	视图控制器，负责响应视图交互，返回视图响应状态。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 单击 **创建房间**。
3. 输入房间主题，单击 **开始交谈**。

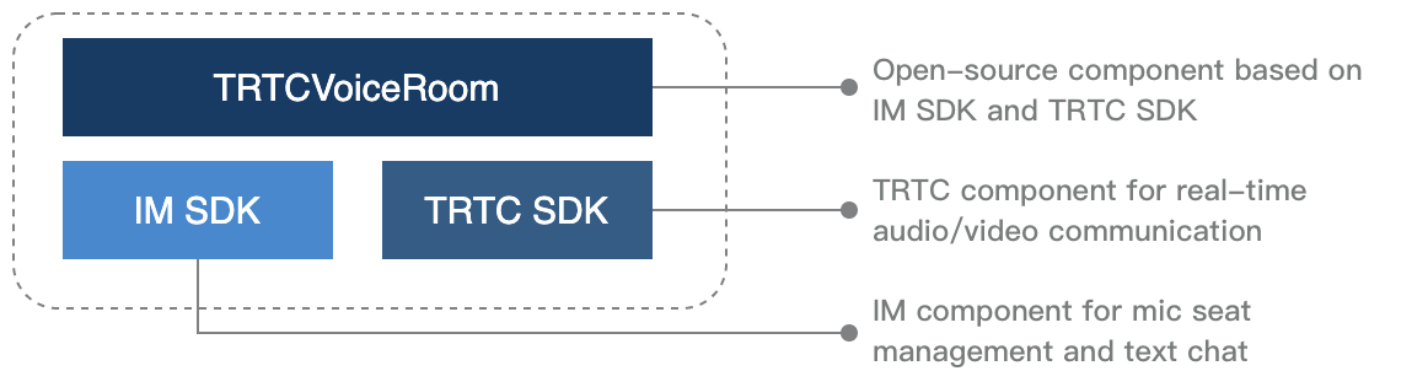
用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入用户 A 创建的房间号，单击 **进入房间**。

> 注意：
> 房间号在用户 A 的房间顶部查看。

实现自定义 UI 界面

源码 中的 `Source` 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCVoiceRoom，您可以在 `TRTCVoiceRoom.h` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

语音聊天组件 TRTCVoiceRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 info.plist 文件中需要添加 `Privacy > Camera Usage Description`， `Privacy > Microphone Usage Description` 申请麦克风权限。

步骤3：导入 TUIVoiceRoom 组件

通过 cocoapods 导入组件，具体操作如下：

1. 将工程目录下的 `Source` 、 `Resources` 、 `TXAppBasic` 文件夹、 `TUIVoiceRoom.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
pod 'TXAppBasic', :path =&gt; "TXAppBasic/"
pod 'TXLiteAVSDK_TRTC'
pod 'TUIVoiceRoom', :path =&gt; "./", :subspecs =&gt; ["TRTC"]
```

步骤4：创建并登录组件

1. 调用 TRTCVoiceRoom 的 `sharedInstance` 类方法可以创建一个遵守 TRTCVoiceRoom 协议的实例对象。也可以使用调用 `shared` 类方法，获取 TRTCVoiceRoomImp实例对象直接使用，二者在 TRTCVoiceRoom 的接口使用上没有任何区别。
2. 调用 `setDelegate` 函数注册组件的事件回调通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

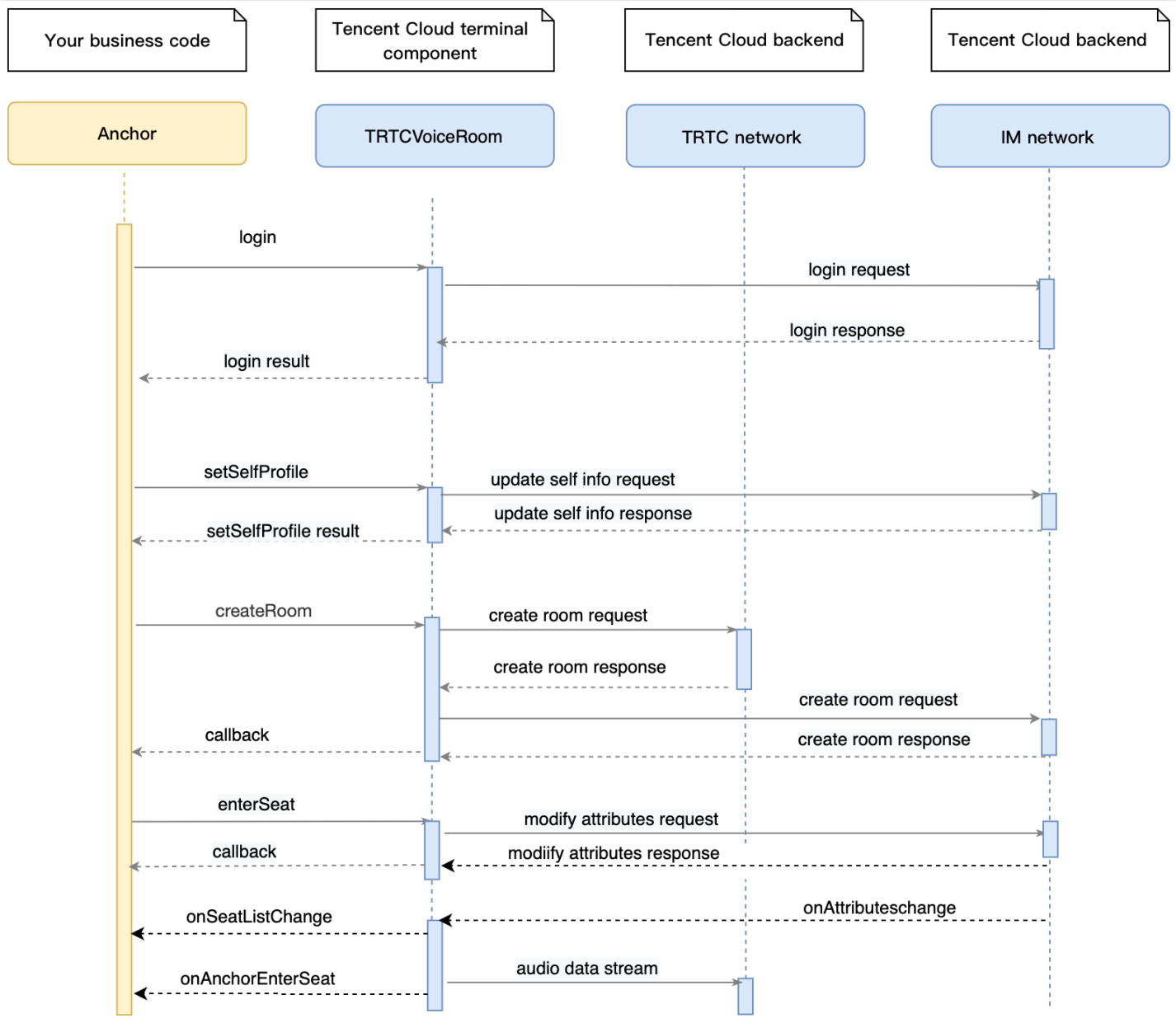
参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	登录回调，成功时 code 为0。

```
// Swift 示例
// 您代码里负责业务逻辑的类
class YourController {
// 计算属性获取单例对象
var voiceRoom: TRTCVoiceRoomImp {
return TRTCVoiceRoomImp.shared()
}

// 其他代码逻辑
......
}
// 设置 voiceroom 代理
self.vocieRoom. setDelegate(delegate: voiceRoomDelegate)

// 调用方式如下,闭包内建议使用 weak self 防止循环引用（下面示例代码省略 weak self 示例）
self.vocieRoom.login(sdkAppId: sdkAppID, userId: userId, userSig: userSig) { [weak self] (cod
```

```
e, message) in
guard let `self` = self else { return }
// 您的回调业务逻辑
}
```

步骤5：房主端开播

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
3. 房主创建房间成功后，调用 `enterSeat` 进入座位。
4. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。



示例代码：

```
// 1.房主设置昵称和头像
self.voiceRoom.setSelfProfile(userName: userName, avatarUrl: avatarURL) { (code, message) in
// 结果回调
}

// 2.房主端创建房间
let param = VoiceRoomParam.init()
param.roomName = "房间名称"
param.needRequest = true // 听众上麦是否需要房主同意
param.coverUrl = "封面URL"
param.seatCount = 7 // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
```

```
param.seatInfoList = []

for _ in 0..&lt; param.seatCount {
let seatInfo = VoiceRoomSeatInfo.init()
param.seatInfoList.append(seatInfo)
}

self.voiceRoom.createRoom(roomID: yourRoomID, roomParam: param) { (code, message) in
guard code == 0 else { reutrn }
// 创建房间成功后开始占座
self.voiceRoom.enterSeat(seatIndex: 0) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
// 房主占座成功
} else {
// 房主占座失败
}
}
}

// 3.占座成功后，收到 onSeatListChange 事件通知
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
// 刷新您的麦位列表
}

// 4. 收到 onAnchorEnterSeat 事件通知
func onAnchorEnterSeat(index: Int, user: VoiceRoomUserInfo) {
// 处理房主上麦事件
}
```

步骤6：听众端观看

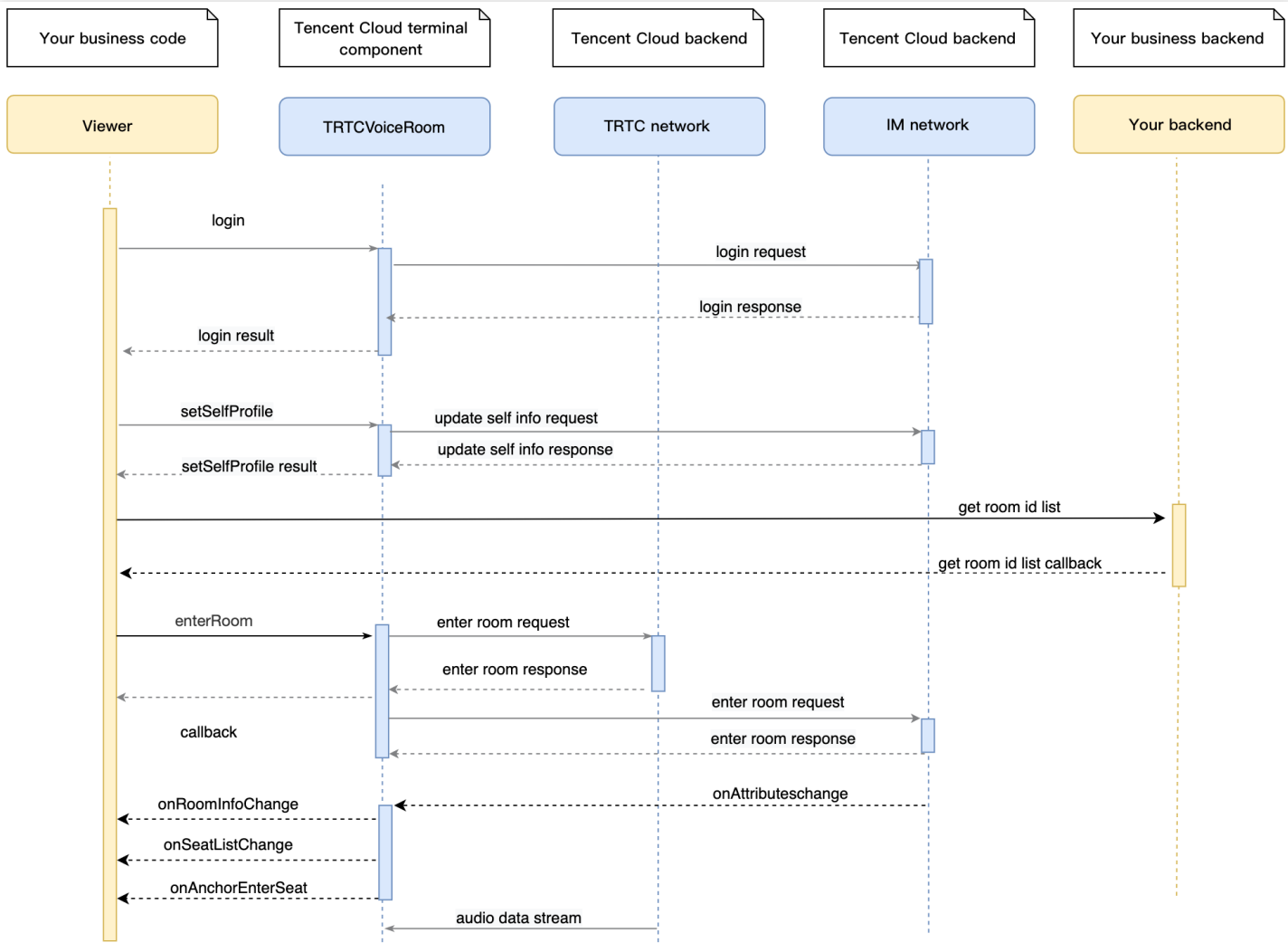
1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的语音聊天室房间列表。

> 说明：
> App 中的语音聊天室列表仅做演示使用，语音聊天室列表的业务逻辑千差万别，腾讯云暂不提供语音聊天室列表的管理服务，请自行管理您的语音聊天室列表。

3. 听众端调用 `getRoomInfoList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建语音聊天室时设置的简单描述信息。

> 注意：
> 如果您的语音聊天室列表包含了足够全面的信息，可跳过调用 `getRoomInfoList` 相关步骤。

4. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
6. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
7. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。



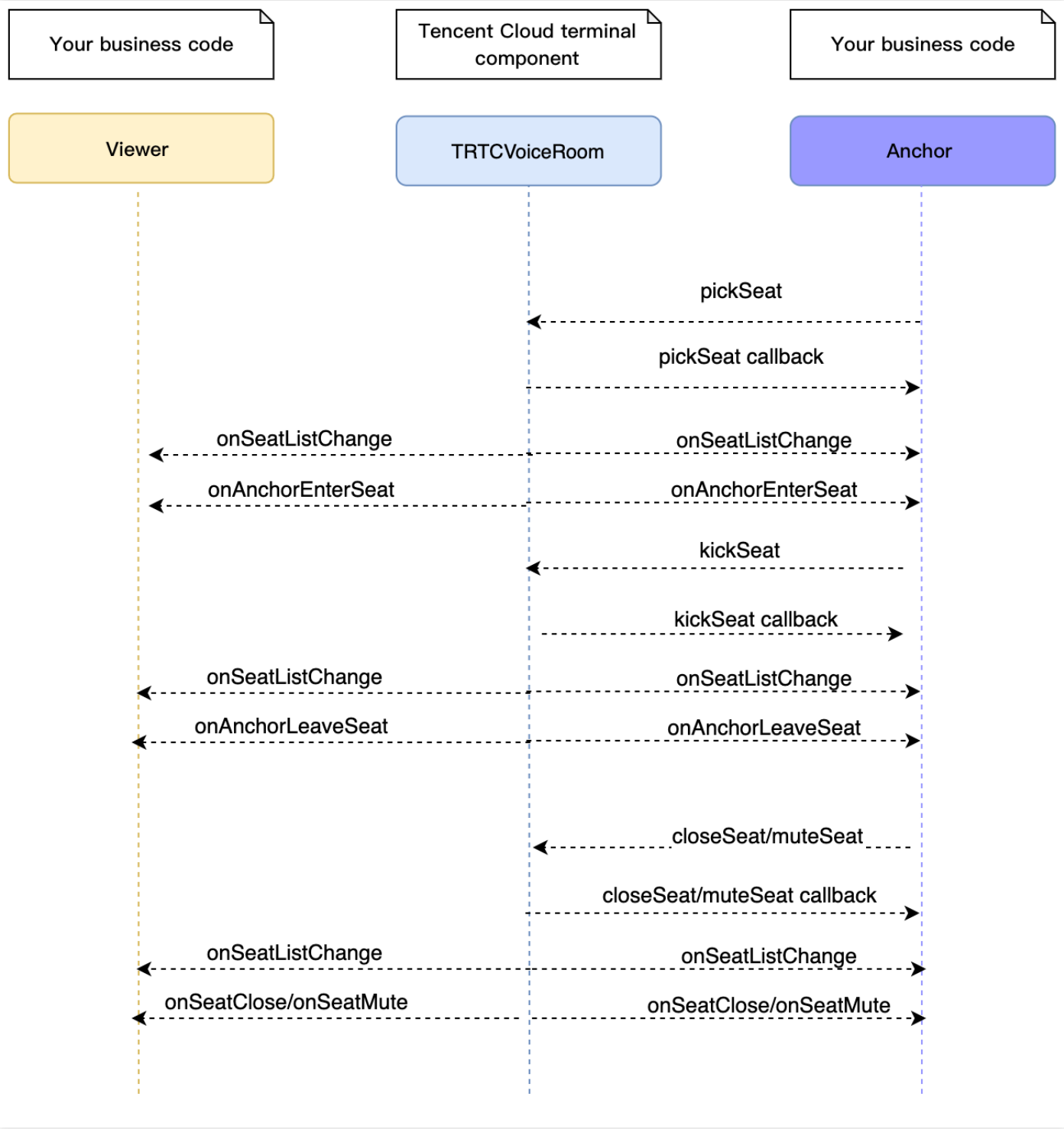

```
// 1.听众设置昵称和头像
self.voiceRoom.setSelfProfile(userName: userName, avatarUrl: avatarURL) { (code, message) in
// 结果回调
}
```

```
// 2.假定您从业务后台获取房间列表为 roomList
let roomList: [Int] = getRoomIDList() // 您获取房间ID列表的函数

// 3.通过调用 getRoomInfoList 获取房间的详细信息
self.voiceRoom.getRoomInfoList(roomIdList: roomIdsInt) { (code, message, roomInfos: [VoiceRoomInf
o]) in
// 获取结果，此时可以刷新UI
}

// 4.选择语音聊天室后，传入 roomId 进入房间
self.voiceRoom.enterRoom(roomID: roomInfo.roomID) { (code, message) in
// 进入房间结果回调
if code == 0 {
// 进房成功
}
}

// 5.进房成功后，收到 onRoomInfoChange 事件通知
func onRoomInfoChange(roomInfo: VoiceRoomInfo) {
// 可以更新房间名称等信息
}

// 6.进房成功后，收到 onSeatListChange 事件通知
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
// 刷新麦位列表
}

// 7. 收到 onAnchorEnterSeat 事件通知
func onAnchorEnterSeat(index: Int, user: VoiceRoomUserInfo) {
// 处理上麦事件
}
```

步骤7：麦位管理

- 房主端
- 听众端

1. `pickSeat` 传入对应的麦位和听众 userId，可以抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。
2. `kickSeat` 传入对应麦位后，可以踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。
3. `muteSeat` 传入对应麦位后，可以静音/解除静音，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。
4. `closeSeat` 传入对应麦位后，可以封禁/解禁某个麦位，封禁后听众端将不能再上麦，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。



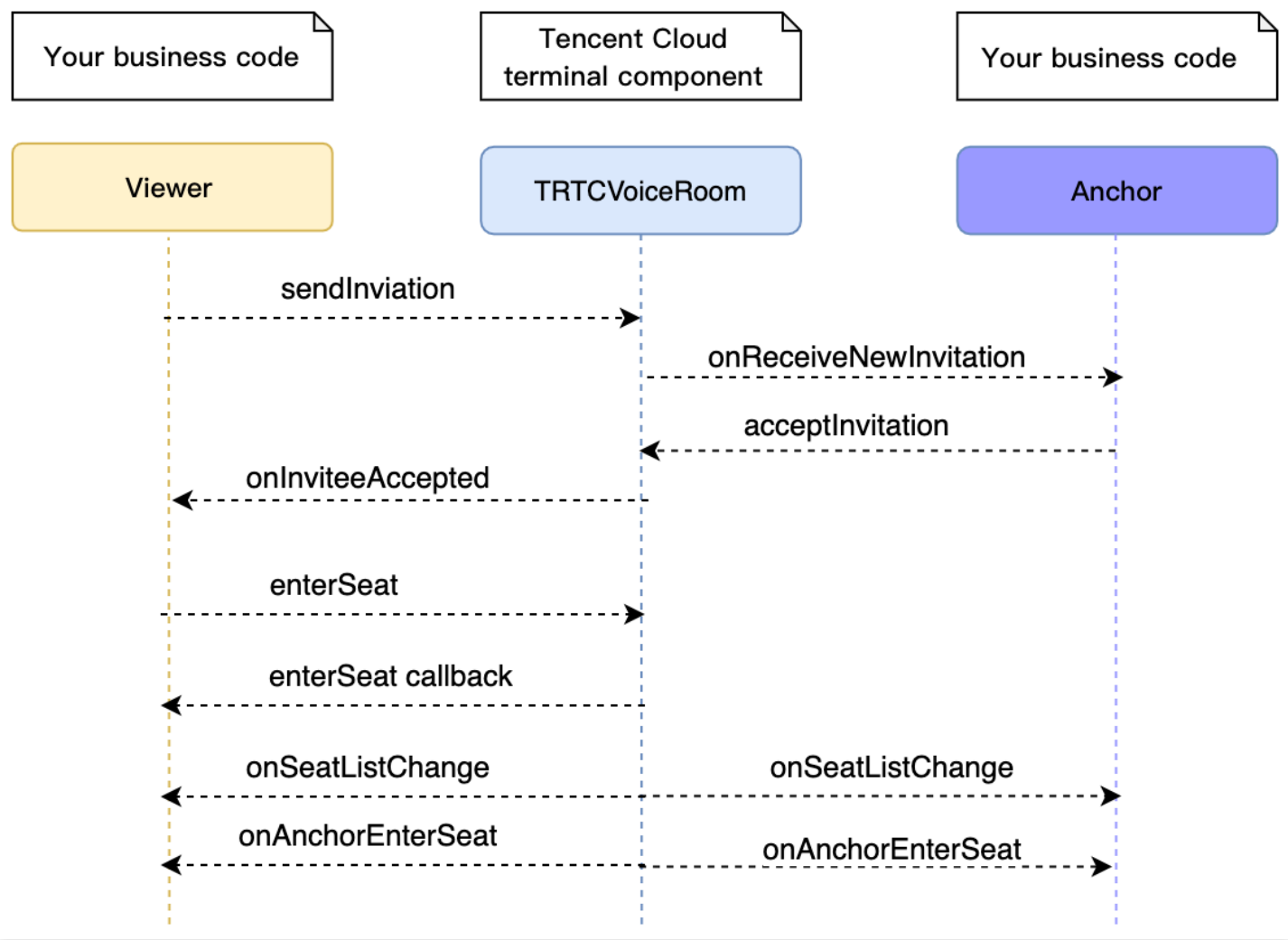
步骤8：邀请信令的使用

在 麦位管理 中，听众上下麦、房主抱人上麦都不需要经过对方的同意就可以直接操作。

如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

- 听众主动申请上麦
- 房主邀请听众上麦

1. 听众端调用 `sendInvitation` 传入房主的 userId 和业务的自定义命令字等，此时函数会返回一个 inviteId，记录该 inviteId。
2. 房主端收到 `onReceiveNewInvitation` 的事件通知，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `acceptInvitation` 并传入 inviteId。
4. 听众端收到 `onInviteeAccepted` 的事件通知，调用 `enterSeat` 进行上麦。




```
// 听众端视角
// 1.调用 sendInvitation，请求上1号麦位
let inviteId = self.voiceRoom.sendInvitation(cmd: "ENTER_SEAT", userId: ownerUserId, content: "1"
) { (code, message) in
// 发送结果回调
}
// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
self.voiceRoom.enterSeat(seatIndex: ) { (code, message) in
```

```
// 上麦结果回调
}
}
}

// 房主端视角
// 1.房主收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
if cmd == "ENTER_SEAT" {
// 2.房主同意听众请求
self.voiceRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
self.voiceRoom.sendRoomTextMsg(message: message) { (code, message) in


}
// 接收端：监听文本消息
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
//收到的message信息处理方法
}
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomCustomMsg` 回调。

  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
self.vocieRoom.sendRoomCustomMsg(cmd: "CMD_DANMU", message: "hello world", callback: nil)
self.voiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)
// 接收端：监听自定义消息
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserInfo) {
if cmd == "CMD_DANMU" {
// 收到弹幕消息
}
if cmd == "CMD_LIKE" {
// 收到点赞消息
```

```
}
}
```

语音聊天室(Android)

最近更新时间：2022-03-09 18:10:19

效果展示

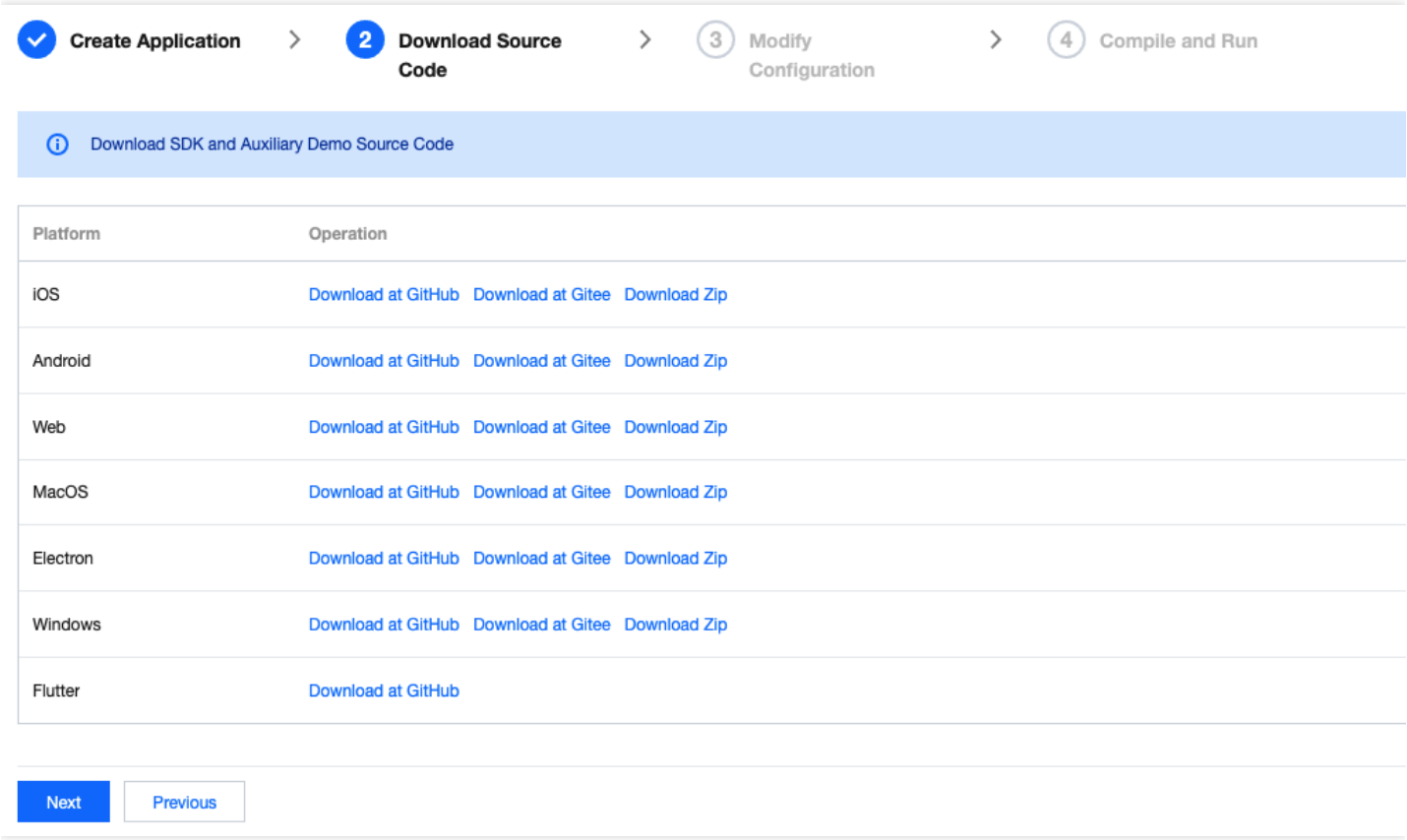
您可以 下载 安装我们的 App 体验语音聊天室的能力，包括麦位管理、低延时语音互动、文字聊天等 TRTC 在语音聊天场景下的相关能力。

如需快速接入语音聊天室功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TRTCVoiceRoom 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助**>**快速跑通Demo**。
2. 输入应用名称，例如 `TestVoiceRoom` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

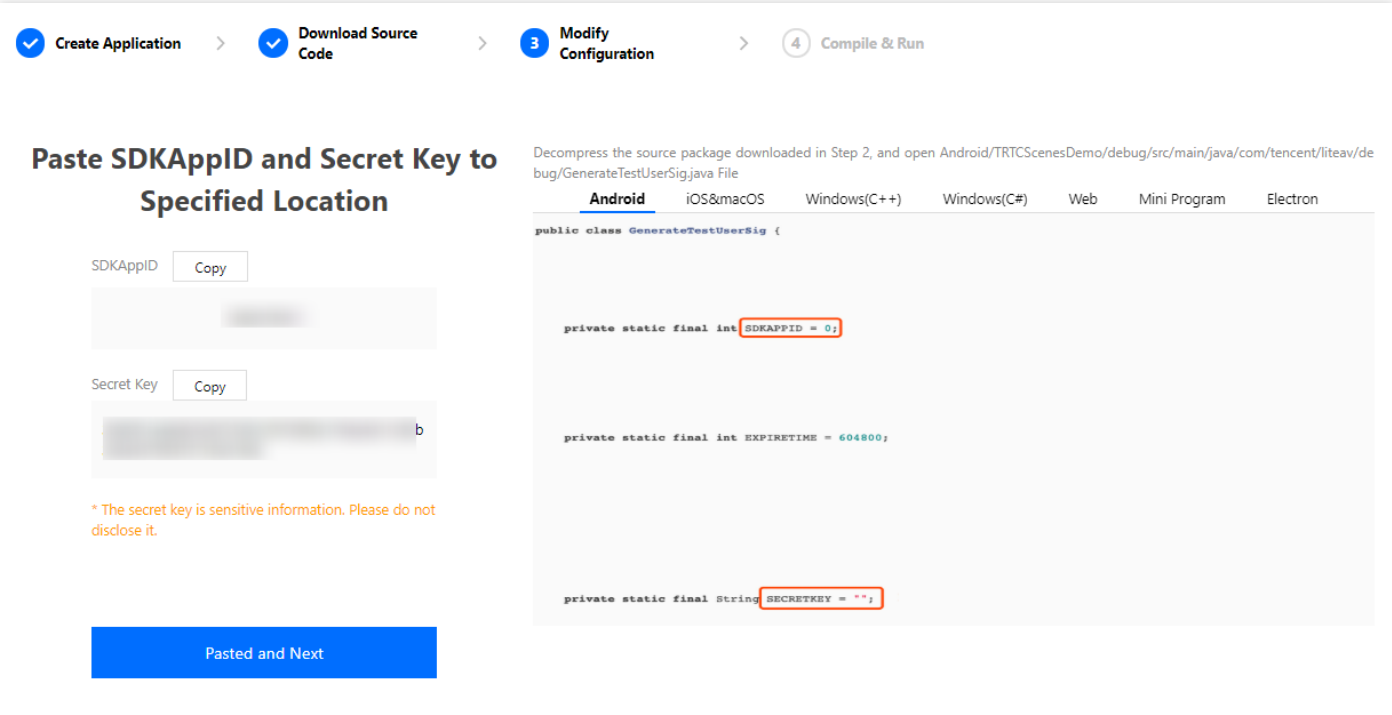
单击进入 TUIVoiceRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5以上的版本）打开源码工程 `TUIVoiceRoom`，单击 **运行** 即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 Source 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
base	UI 使用的基础类。
room	主房间页面，包括房主和听众两种界面。
widget	通用控件。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 单击 **创建房间**。
3. 输入房间主题，单击 **开始交谈**。

用户 B

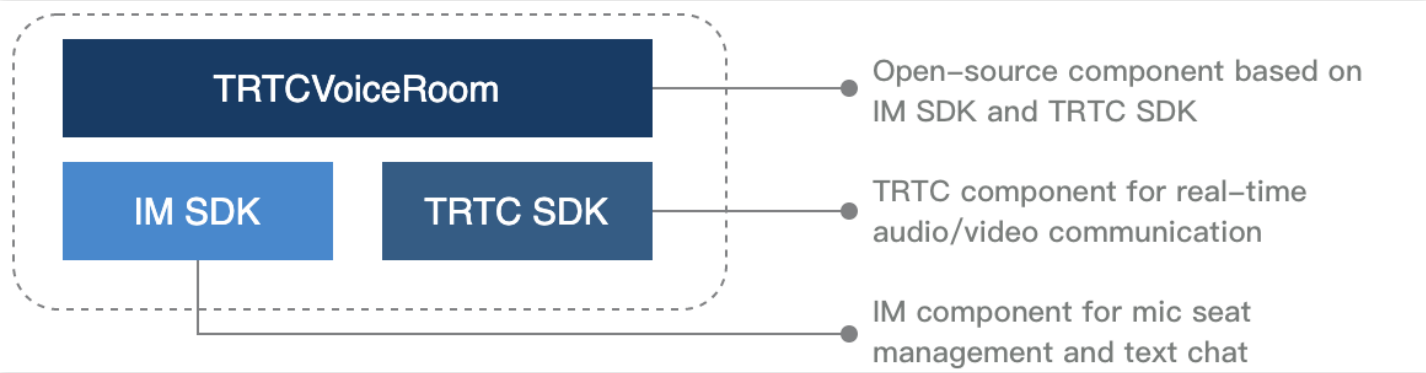
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入用户 A 创建的房间号，单击 **进入房间**。

> 注意：
> 房间号在用户 A 的房间顶部查看。

实现自定义 UI 界面

源码 中的 Source 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCVoiceRoom，您可以在 `TRTCVoiceRoom.java` 文件中看到该组件提供的接口函数，并使用对应接口实现自定

义 UI 界面。



步骤1：集成 SDK

语音聊天组件 TRTCVoiceRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 Maven 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
complie "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
complie 'com.tencent.imsdk:imsdk:latest.release'
compile 'com.google.code.gson:gson:2.3.1'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击 **Sync Now**，自动下载 SDK 并集成到工程里。

方法二：通过本地 AAR 依赖

若您的开发环境访问 Maven 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档

IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请读取存储权限）：

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

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

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

步骤3：导入 TRTCVoiceRoom 组件

拷贝以下目录中的所有文件到您的项目中：

```
Source/src/main/java/com/tencent/liteav/trtcvoiceroom/model
```

步骤4：创建并登录组件

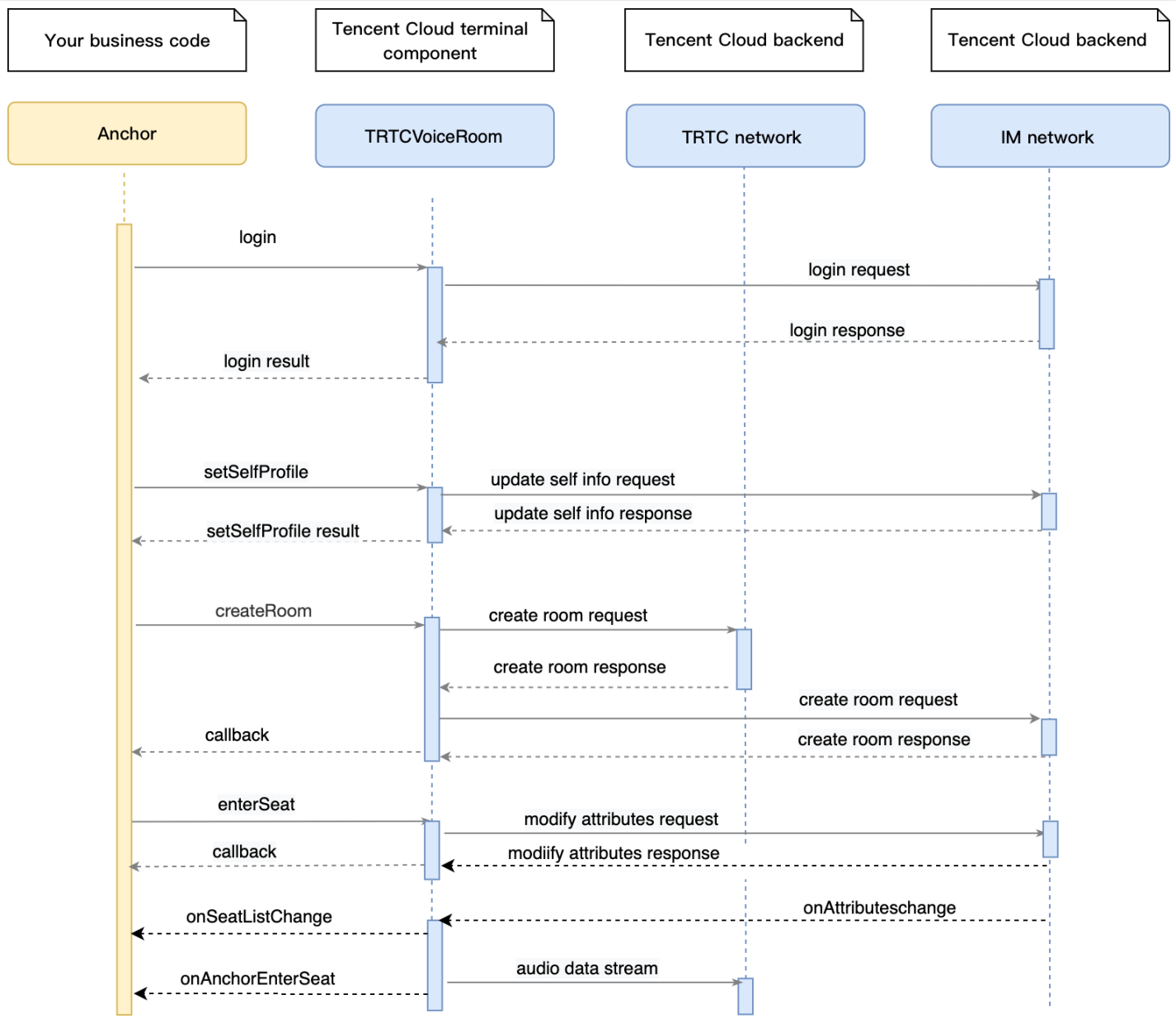
1. 调用 `sharedInstance` 接口可以创建一个 TRTCVoiceRoom 组件的实例对象。
2. 调用 `setDelegate` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	登录回调，成功时 code 为0。

```
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
mTRTCVoiceRoom.setDelegate(this);
mTRTCVoiceRoom.login(SDKAPPID, userId, userSig, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {![](https://main.qcloudimg.com/raw/3151fc0df16c063db7a75f6c1facf562.png)
//登录成功
}
}
});
```

步骤5：房主端开播

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
3. 房主创建房间成功后，调用 `enterSeat` 进入座位。
4. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。




```
// 1.房主设置昵称和头像
mTRTCVoiceRoom.setSelfProfile("my_name", "my_face_url", null);

// 2.房主调用 createRoom 创建房间
final TRTCVoiceRoomDef.RoomParam roomParam = new TRTCVoiceRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 7; // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.coverUrl = "房间封面图的 URL 地址";
mTRTCVoiceRoom.createRoom(mRoomId, roomParam, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
```

```
//3. 占0号座位
mTRTCVoiceRoom.enterSeat(0, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
}
}
});
}
}
});

// 4.占座成功后，收到 onSeatListChange 事件通知
@Override
public void onSeatListChange(final List<trtcvoiceroomdef.seatinfo> seatInfoList) {
// 展示您的麦位列表
}

// 5. 收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCVoiceRoomDef.UserInfo userInfo) {
}
```

步骤6：听众端观看

1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的语音聊天室房间列表。

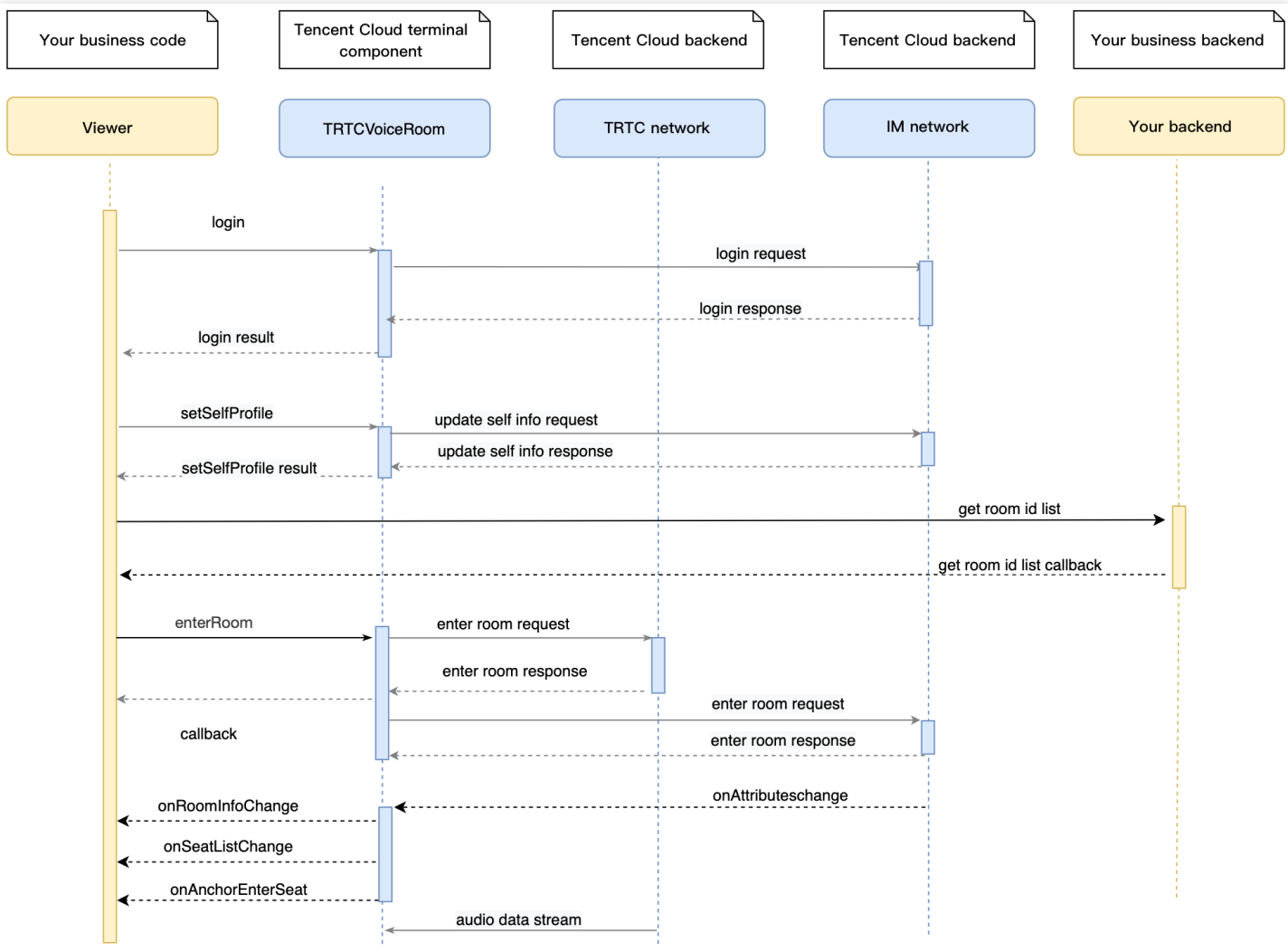
> 说明：
> App 中的语音聊天室列表仅做演示使用，语音聊天室列表的业务逻辑千差万别，腾讯云暂不提供语音聊天室列表的管理服务，请自行管理您的语音聊天室列表。

3. 听众端调用 `getRoomInfoList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建语音聊天室时设置的简单描述信息。

> 注意：
> 如果您的语音聊天室列表包含了足够全面的信息，可跳过调用 `getRoomInfoList` 相关步骤。

4. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。

5. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
6. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
7. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。




```
// 1.听众设置昵称和头像
mTRTCVoiceRoom.setSelfProfile("my_name", "my_face_url", null);

// 2.假定您从业务后台获取房间列表为 roomList
List<integer> roomList = GetRoomList();

// 3.通过调用 getRoomInfoList 获取房间的详细信息
mTRTCVoiceRoom.getRoomInfoList(roomList, new TRTCVoiceRoomCallback.RoomInfoCallback() {
@Override
public void onCallback(int code, String msg, List<trtcvoiceroomdef.roominfo> list) {
if (code == 0) {
```

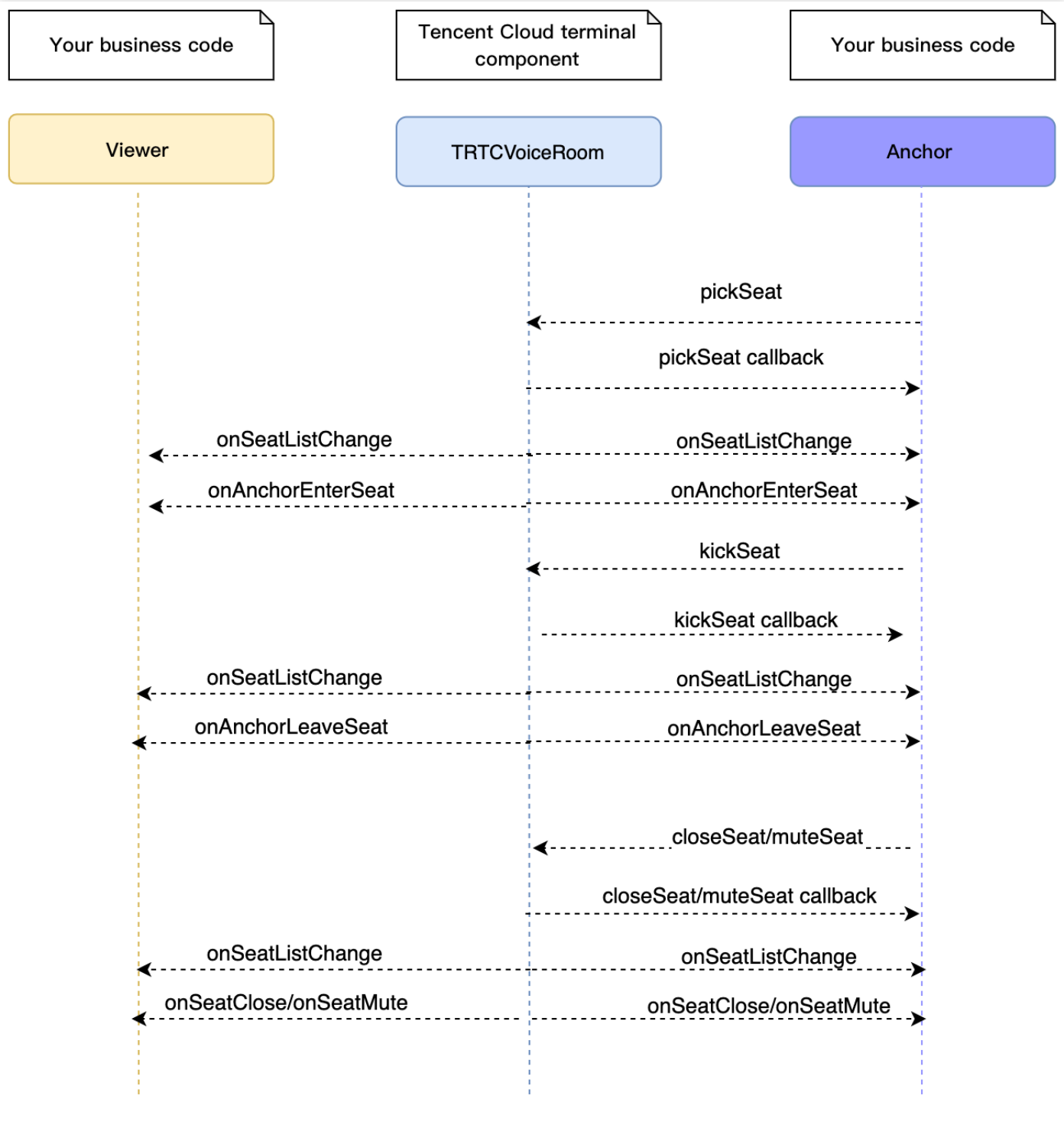
```
// 此时可以刷新您自己的 UI 房间列表
}
}
});

// 4.选择语音聊天室后，传入 roomId 进入房间
mTRTCVoiceRoom.enterRoom(roomId, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
}
}
});

// 5.进房成功后，收到 onRoomInfoChange 事件通知
@Override
public void onRoomInfoChange(TRTCVoiceRoomDef.RoomInfo roomInfo) {
mNeedRequest = roomInfo.needRequest;
mRoomName = roomInfo.roomName;
// UI 可以展示标题等
}

// 6.进房成功后，收到 onSeatListChange 事件通知
@Override
public void onSeatListChange(final List<trtcvoiceroomdef.seatinfo> seatInfoList) {
// 展示您的麦位列表
}

// 7. 收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCVoiceRoomDef.UserInfo userInfo) {
}
```

步骤7：麦位管理

- 房主端
- 听众端

1. `pickSeat` 传入对应的麦位和听众 userId，可以抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。
2. `kickSeat` 传入对应麦位后，可以踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。
3. `muteSeat` 传入对应麦位后，可以静音/解除静音，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

4. `closeSeat` 传入对应麦位后，可以封禁/解禁某个麦位，封禁后听众端将不能再上麦，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。



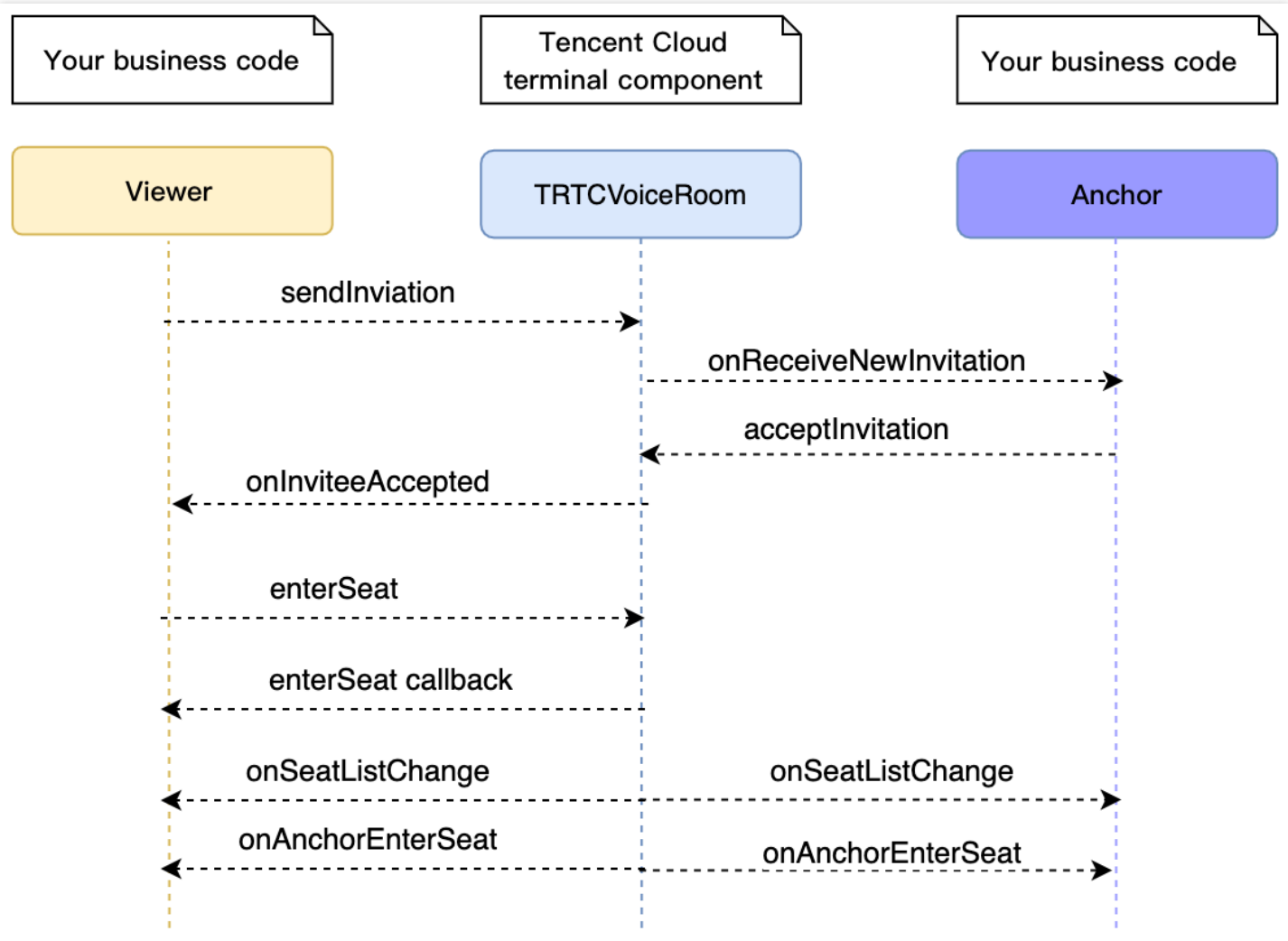
步骤8：邀请信令的使用

在 麦位管理 中，听众上下麦、房主抱人上麦都不需要经过对方的同意就可以直接操作。

如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

- 听众主动申请上麦
- 房主邀请听众上麦

1. 听众端调用 `sendInvitation` 传入房主的 userId 和业务的自定义命令字等，此时函数会返回一个 inviteId，记录该 inviteId。
2. 房主端收到 `onReceiveNewInvitation` 的事件通知，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `acceptInvitation` 并传入 inviteId。
4. 听众端收到 `onInviteeAccepted` 的事件通知，调用 `enterSeat` 进行上麦。




```
// 听众端视角
// 1.调用 sendInvitation，请求上1号麦位
String inviteId = mTRTCVoiceRoom.sendInvitation("ENTER_SEAT", ownerUserId, "1", null);

// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
```

```
if(id.equals(inviteId)) {
mTRTCVoiceRoom.enterSeat(1, null);
}
}

// 房主端视角
// 1.房主收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String cont
ent) {
if (cmd.equals("ENTER_SEAT")) {
// 2.房主同意听众请求
mTRTCVoiceRoom.acceptInvitation(id, null);
}
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
mTRTCVoiceRoom.sendRoomTextMsg("Hello Word!", null);
// 接收端：监听文本消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
@Override
public void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomCustomMsg` 回调。

  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo userInf
o) {
if ("CMD_DANMU".equals(cmd)) {
```

```
// 收到弹幕消息
Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
} else if ("CMD_LIKE".equals(cmd)) {
// 收到点赞消息
Log.d(TAG, userInfo.userName + "给您点了个赞！");
}
}
});
```

TRTCVoiceRoom (iOS)

最近更新时间：2022-01-10 11:51:58

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 nil，则获取房间内所有人的信息。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位 (麦上主播端可调用) 。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
/**
* 获取 TRTCVoiceRoom 单例对象
*
* - returns: TRTCVoiceRoom 实例
* - note: 可以调用 {@link TRTCVoiceRoom#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCVoiceRoom 单例对象
```

```
*
* - note: 销毁实例后，外部缓存的 TRTCVoiceRoom 实例不能再使用，需要重新调用 {@link TRTCVoiceRoom#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCVoiceRoom 事件回调，您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知。

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCVoiceRoom 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，可使用 {@link TRTCVoiceRoom#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCVoiceRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

说明：

setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送动主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
* - parameter queue 队列，TRTCVoiceRoom 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
userId:(NSString *)userId
userSig:(NSString *)userSig
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback
:));
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在**实时音视频控制台 >**应用管理> 应用信息中查看 SDKAppID。
userId	NSString	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	NSString	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallb
ack _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userName	NSString	昵称。
avatarURL	NSString	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(VoiceRoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	VoiceRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	NSInteger	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(VoiceRoomInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义
roomIdList	NSArray<NSNumber>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(VoiceRoomUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	NSArray<NSString>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME
(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

moveSeat

移动麦位 (麦上主播端可调用)。

> 说明：
> 移动麦位成功后，房间内所有成员会收到 `onSeatListChange`、 `onAnchorLeaveSeat` 和 `onAnchorEnterSeat` 的事件通知。(主播调用后，只是修改麦位座位号信息，并不会切换该用户的主播身份。)

```
- (NSInteger)moveSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(moveSeat(seatIndex:callback:))
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要移动到的麦位序号。
callback	ActionCallback	操作回调。

返回值：

返回值	类型	含义
code	NSInteger	移动麦位操作结果（0为成功，其它为失败，10001为接口调用限频）。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCallback _Nullabl
e)callback NS_SWIFT_NAME(pickSeat(seatIndex:userId:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要抱上麦的麦位序号。
userId	NSString	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(kickSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> 说明：
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(muteSeat(seatIndex:isMute:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。
isMute	BOOL	YES：静音对应麦位；NO：解除静音对应麦位。

参数	类型	含义
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(closeSeat(seatIndex:isClose:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。
isClose	BOOL	YES：封禁对应麦位； NO：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
useSpeaker	BOOL	YES：扬声器；NO：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)voluem NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	指定的用户 ID。
mute	BOOL	YES：开启静音；NO：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	YES：开启静音；NO：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enable:));
```

参数如下表所示：

参数	类型	含义
enable	BOOL	YES：开启耳返；NO：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_
NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	NSString	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	NSString	命令字，由开发者自定义，主要用于区分不同消息类型。
message	NSString	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
userId:(NSString *)userId
content:(NSString *)content
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendInvitation(cmd:userId:content:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	NSString	业务自定义指令。
userId	NSString	邀请的用户 ID。
content	NSString	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	NSString	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(acceptInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(rejectInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

版权所有：腾讯云计算（北京）有限责任公司

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
- (void)onError:(int)code
message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	NSString	错误信息。

onWarning

警告回调。

```
- (void)onWarning:(int)code
message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	NSString	警告信息。

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	NSString	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)roomId
NS_SWIFT_NAME(onRoomDestroy(roomId:));
```

参数如下表所示：

参数	类型	含义
roomId	NSString	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(VoiceRoomInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
roomInfo	VoiceRoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用muteLocalAudio，房间内的其他用户都会收到此通知。

```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
mute	BOOL	YES：静音麦位； NO：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSInteger)totalV
olume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
userVolumes	NSArray	用户列表。
totalVolume	NSInteger	音量大小，取值：0 - 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	NSArray<VoiceRoomSeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	成员上麦的麦位。
user	VoiceRoomUserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	下麦的麦位。
user	VoiceRoomUserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	操作的麦位。
isMute	BOOL	YES：静音麦位； NO：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
 isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	操作的麦位。
isClose	BOOL	YES：封禁麦位； NO： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	退房听众信息。

版权所有：腾讯云计算（北京）有限责任公司

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)command
message:(NSString *)message
userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(command:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
command	NSString	命令字，由开发者自定义，主要用于区分不同消息类型。
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
inviter:(NSString *)inviter
cmd:(NSString *)cmd
content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(id:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。
cmd	NSString	业务指定的命令字，由开发者自定义。
content	NSString	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(id:invitee:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。

TRTCVoiceRoom (Android)

最近更新时间：2022-01-10 11:54:14

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位（麦上主播端可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
public static synchronized TRTCVoiceRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCVoiceRoom 事件回调，您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCVoiceRoomDelegate delegate);
```

> 说明：
> setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
String userId, String userSig,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在**实时音视频控制台 >**应用管理> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

参数	类型	含义
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCVoiceRoomCallback.Acti
onCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam, TRTCVoiceRoomCa
llback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
public abstract void destroyRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。

参数	类型	含义
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCVoiceRoomCallback.RoomInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。

参数	类型	含义
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCVoiceRoomCallback.UserListCallback userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
public abstract void enterSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

moveSeat

移动麦位 (麦上主播端可调用)。

> 说明：
> 移动麦位成功后，房间内所有成员会收到 `onSeatListChange`、 `onAnchorLeaveSeat` 和 `onAnchorEnterSeat` 的事件通知。(主播调用后，只是修改麦位座位号信息，并不会切换该用户的主播身份。)

```
public abstract int moveSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要移动到的麦位序号。
callback	ActionCallback	操作回调。

返回值：

返回值	类型	含义
code	int	移动麦位操作结果（0为成功，其它为失败，10001 为接口调用限频）。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void leaveSeat(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
public abstract void pickSeat(int seatIndex, String userId, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void kickSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> 说明：
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCVoiceRoomCallback.ActionCallback callbac
k);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCVoiceRoomCallback.ActionCa
llback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCVoiceRoomCal
lback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCVoiceRoomDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用 muteLocalAudio，房间内的其他用户都会收到此通知。

```
void onUserMicrophoneMute(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	true：静音麦位； false：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVolume);
```

参数如下表所示：

参数	类型	含义
userVolumes	ListList<trtcclouddef.trtcvolumeinfo>	用户列表。
totalVolume	int	音量大小，取值：0 - 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

版权所有：腾讯云计算（北京）有限责任公司

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

参数	类型	含义
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。

参数	类型	含义
inviter	String	邀请人的用户 ID。
</trtcclouddef.trtcvolumeinfo>		

多人音视频房间
多人音视频房间(iOS)

最近更新时间：2022-03-02 11:30:43

效果展示

您可以 下载 安装我们的 App 体验多人视频会话的效果，包括屏幕分享、美颜、低延时视频通话等 TRTC 在多人音视频场景下的相关能力。

方案优势

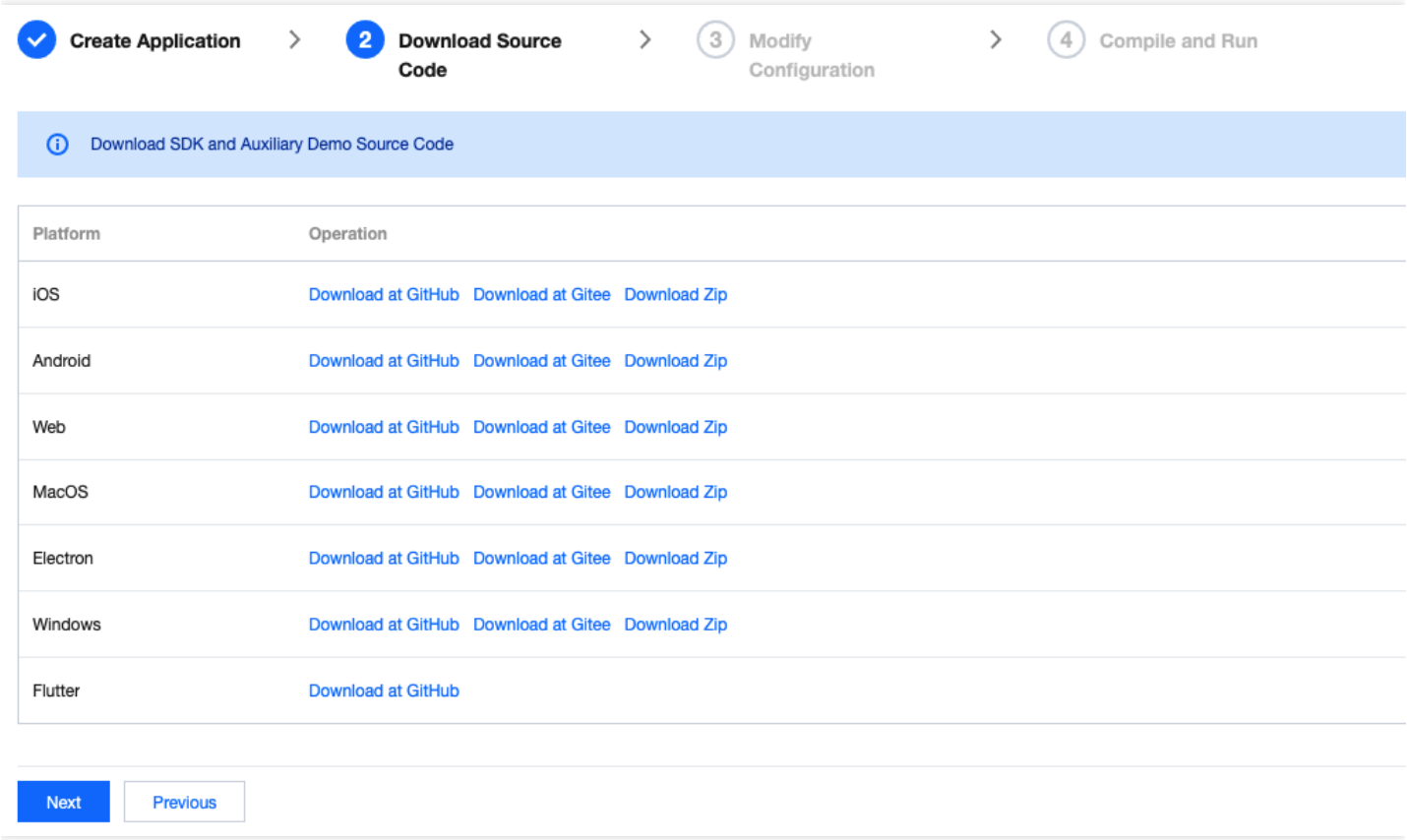
- 集成了超低延时音视频通话、屏幕共享、美颜等能力，覆盖多人音视频房间常见功能。
- 根据需求二次开发，可以快速实现自定义 UI 界面和布局，助力业务快速上线。
- 封装了 TRTC 和 IM 基础 SDK，实现基础的逻辑控制，并提供接口方便调用。

接入指引

如需快速接入多人音视频房间功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用的 App 内的 Module 模块实现自定义 UI 界面。

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > 快速跑通Demo。
2. 输入应用名称，例如 `TestTUIRoom` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

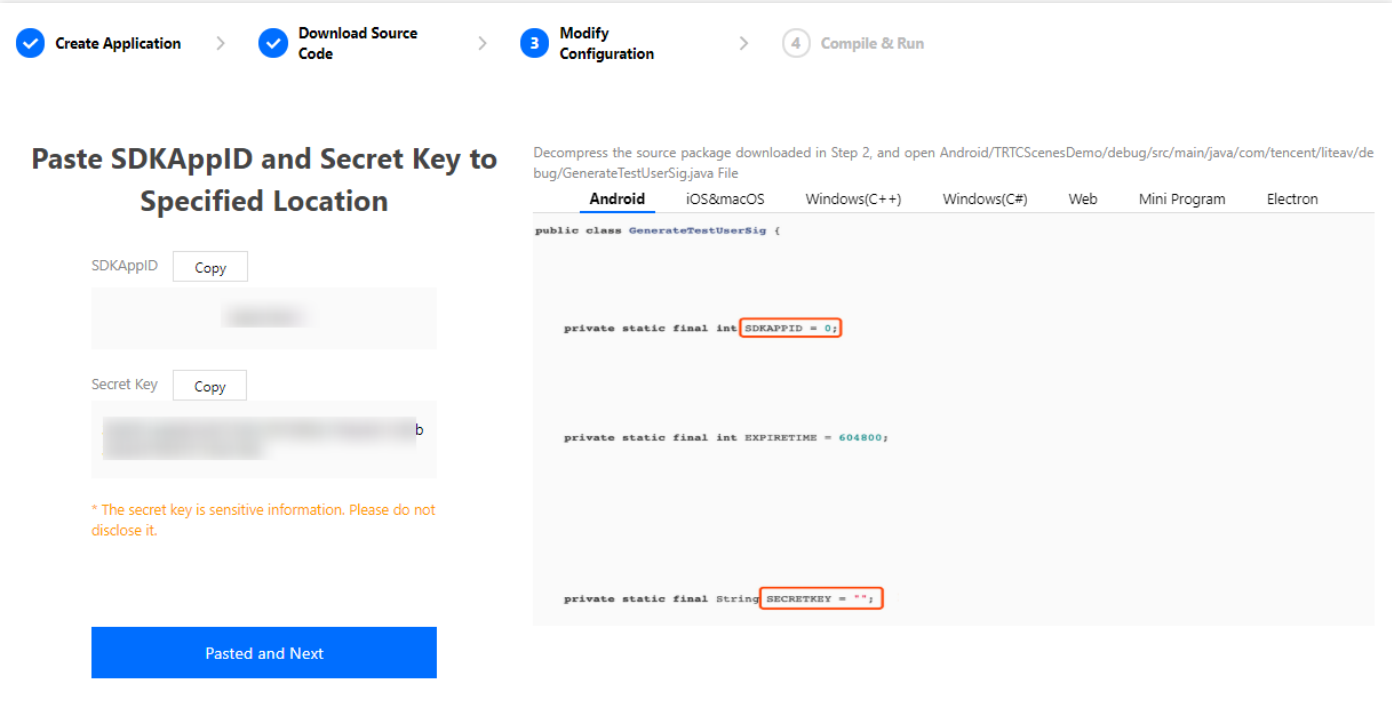
单击进入 TUIRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Example/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

说明：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行工程

使用 Xcode（11.0及以上的版本）打开源码工程 `Example/DemoApp.xcworkspace`，单击 **运行** 即可开始调试本 App。

步骤5：修改工程源代码

源码中的 `Source` 中包 UI 文件夹，此文件夹均为 UI 相关代码，如下表格列出了各个文件夹所对应的 UI 界面，以便于您进行二次调整：

文件夹	功能描述
TUIRoomEnter	TUIRoom入口相关实现代码，TUIRoomEntranceViewController类为 Pods 对外暴露的公共类。
TUIRoomMain	TUIRoom主界面相关 UI 实现代码，此类为 Pods 私有类。
TUIRoomMemberList	成员列表界面相关 UI 实现代码，此类为 Pods 私有类。
TUIRoomSet	设置界面相关 UI 实现代码，此类为 Pods 私有类。

体验应用

> 注意：
> 体验应用至少需要两台设备。

入口界面

请选择 **创建房间** 或 **加入房间**.

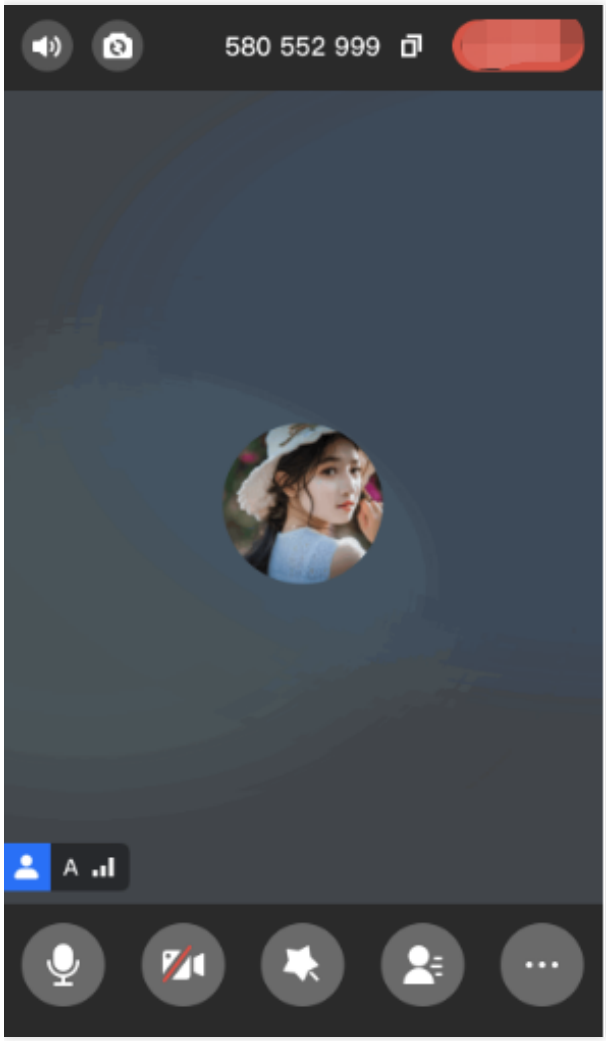
创建房间页面

A 用户创建，房间号会默认生成，单击 **创建房间** 可以进入主页面。

加入房间页面

B 用户输入 A 用户的房间号，单击 **加入房间** 可以进入主页面。

主页面（A 用户）



麦上列表

麦上列表可以展示当前房间内的成员，对方打开摄像头和麦克风后可以看到对方的画面，听到对方的声音。

顶部控制栏

实现了耳麦切换，前后摄像头切换，房间信息，退出的功能。

底部工具栏

实现了控制自己的麦克风/摄像头，美颜，成员列表，设置等功能。

美颜

可在直播中针对画面进行美颜和特效展示.

设置窗口

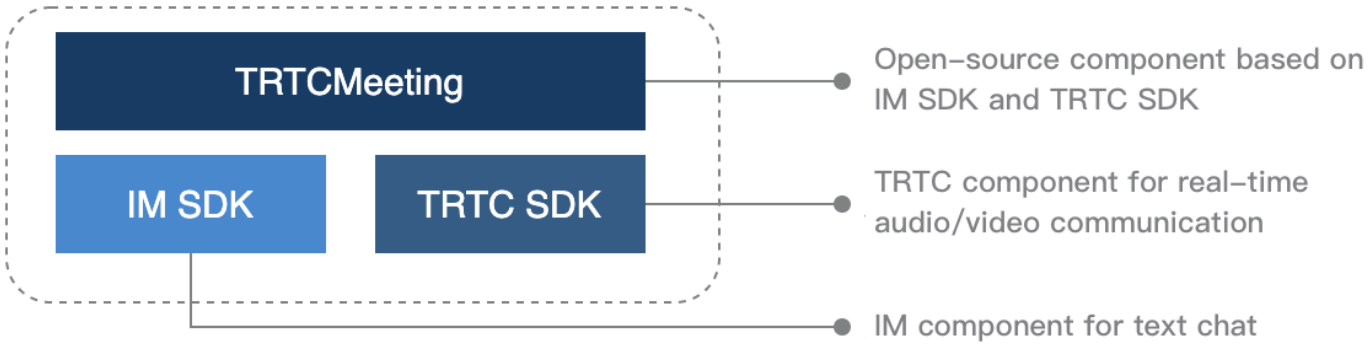
可对音视频相关参数进行设置，支持开启**屏幕共享**。

退出房间

- **主持人**：解散房间，即所有人退出房间。
- **非主持人**：自己离开房间。

实现自定义 UI 界面

源码 中的 `Source` 文件夹包含 UI 和 Presenter，Presenter 文件夹中包含可重用的开源组件 TUIRoomCore，您可以在 `TUIRoomCore.h` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

多人音视频视频组件 TUIRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

> 说明：
> 两个 SDK 产品的最新版本号，可以在 TRTC 和 IM 的 Github 首页获取。

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档

SDK	下载页面	集成指引
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 `info.plist` 文件中需要添加 `Privacy > Camera Usage Description` 和 `Privacy > Microphone Usage Description` 申请摄像头和麦克风权限。

步骤3：导入 TUIRoom 组件

您可通过 **cocoapods 导入组件**，具体步骤如下：

1. 将工程目录下的 `Source`、`Resources`、`TCBeautyKit`、`TXAppBasic` 文件夹、`TUIRoom.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
# 本地依赖库
def local
pod 'TXAppBasic', :path => "../../TXAppBasic/"
pod 'TCBeautyKit', :path => "../../TCBeautyKit/"
pod 'TXLiteAVSDK_TRTC',:path => "../../../SDK/"
end
def pod_local(type)
loadLocalPod('TUIRoom', type)
end
def loadLocalPod(name, type)
pod "#{name}/#{type}", :path => "../"
end
target 'DemoApp' do
local
pod_local('TRTC')
end
```

步骤4：创建并登录组件

1. 调用 TUICore 中的 TUILogin 进行登录，请参考如下示例：

```
TUILogin.initWithSdkAppID(Int32(SDKAPPID))
let userID = ProfileManager.shared().curUserID()
let userSig = GenerateTestUserSig.genTestUserSig(userID)
TUILogin.login(userID, userSig: userSig, succ: {
debugPrint("login success")
}, fail: { (code, errorDes) in
```

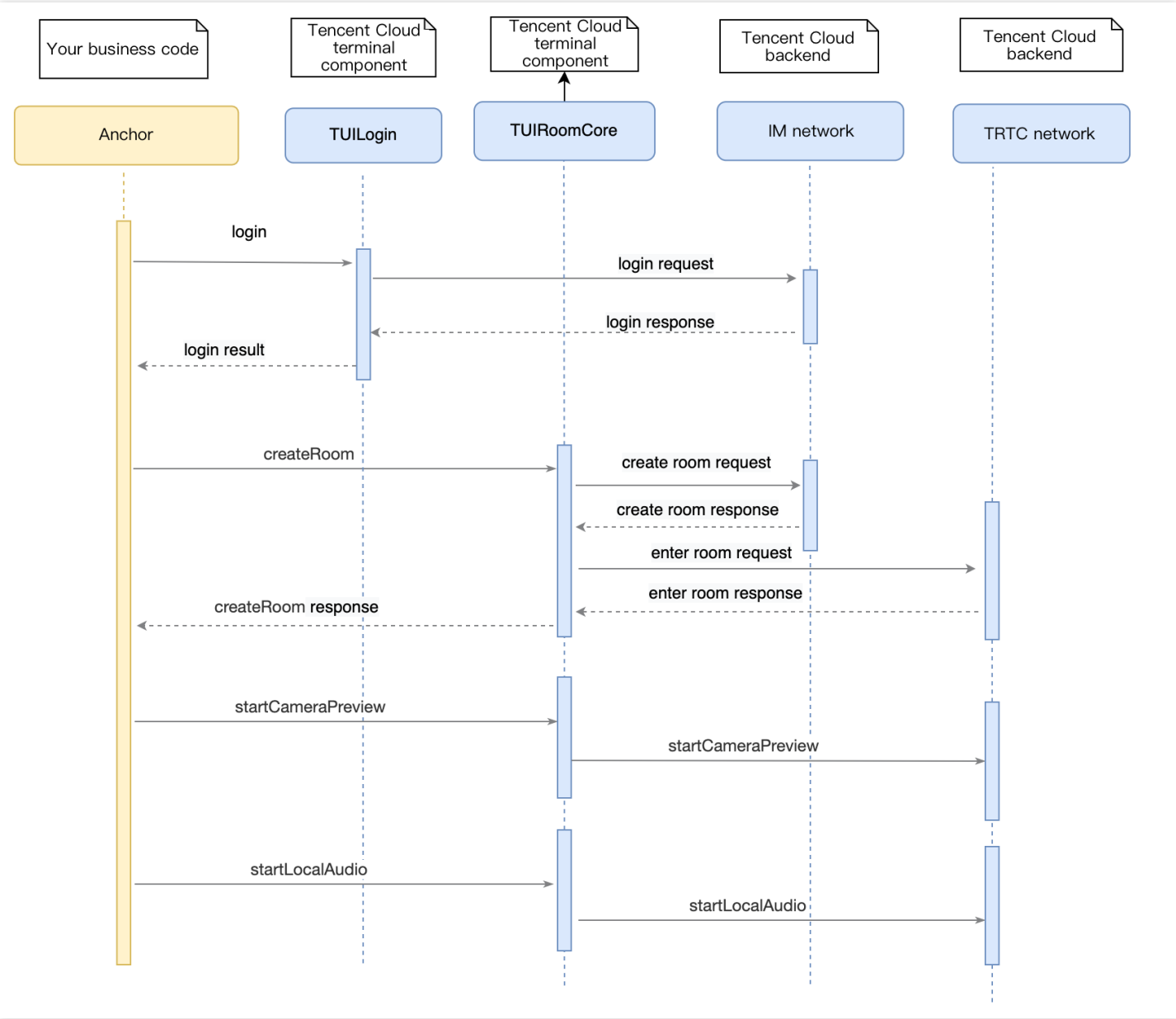
```
debugPrint("login failed, code:¥(code), error: ¥(errorDes ?? "nil")")
})
```

2. 登录成功后，可以调用 TUIRoomCore 创建房间。

```
let roomId = 123
TUIRoomCore.shareInstance().createRoom("¥(roomId)",speechMode: .freeSpeech,callback: { [weak s
elf] code, message in
if code == 0 {
debugPrint("create room success")
} else {
}
})
```

3. 创建成功，进入主页面。

```
let vc = TUIRoomMainViewController(roomId: roomId, isVideoOn: openCameraSwitch.isOn, isAudioO
n: openMicSwitch.isOn)
TUIRoomCore.shareInstance().setDelegate(vc)
navigationController?.pushViewController(vc, animated: true)
```



步骤5：成员进入房间

1. 成员调用 TUICore 中的 TUILogin 进行登录，请参考如下示例：

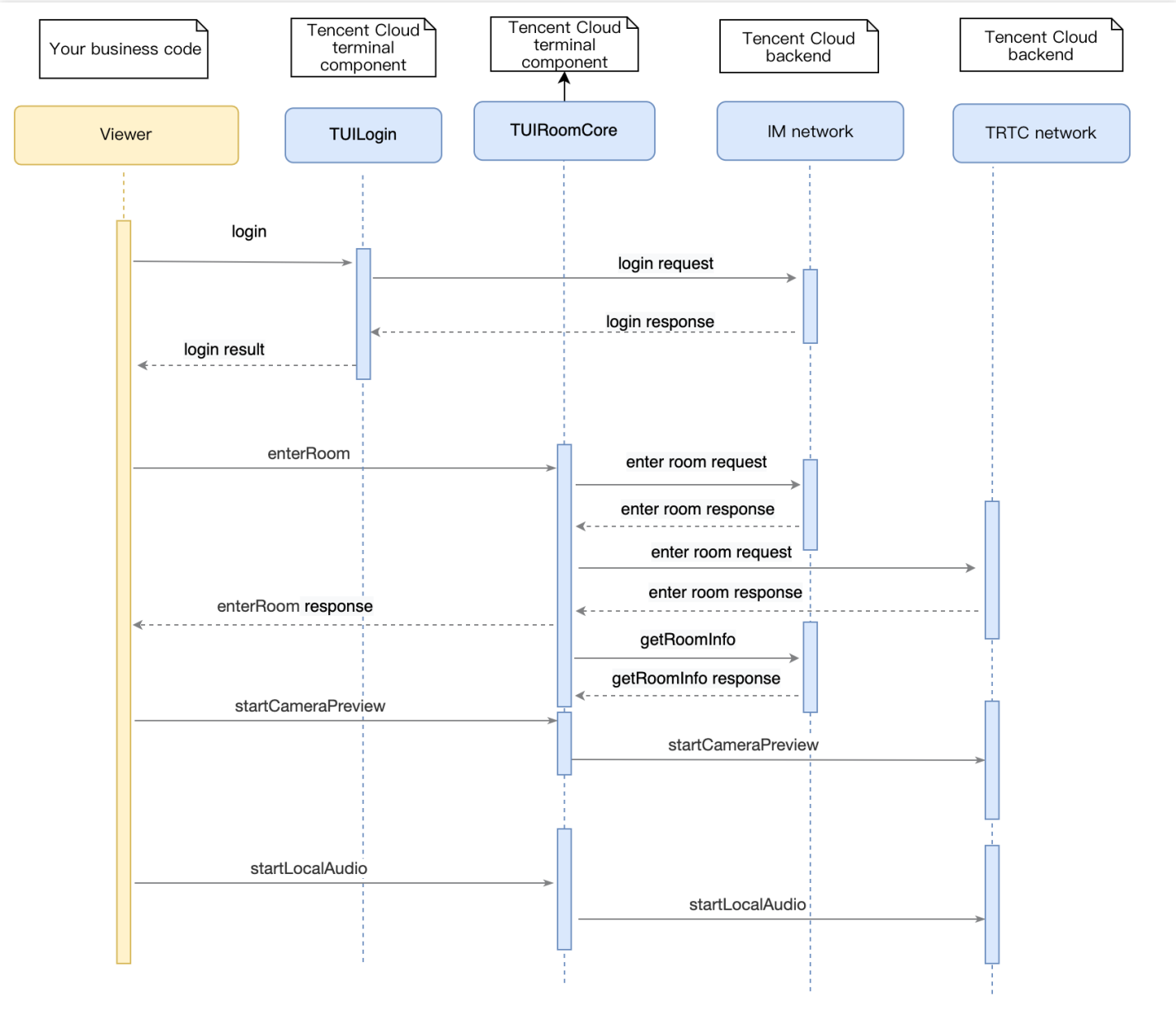
```
TUILogin.initWithSdkAppID(Int32(SDKAPPID))
let userID = ProfileManager.shared().curUserID()
let userSig = GenerateTestUserSig.genTestUserSig(userID)
TUILogin.login(userID, userSig: userSig, succ: {
debugPrint("login success")
}, fail: { (code, errorDes) in
debugPrint("login failed, code:¥(code), error: ¥(errorDes ?? "nil")")
})
```

2. 登录成功后，可以调用 TUIRoomCrre 进入房间。

```
let roomId = 123
TUIRoomCore.shareInstance().enterRoom("¥(roomId)", callback: { [weak self] code, message in
if code == 0 {
debugPrint("enter room success")
} else {
}
})
```

3. 进入成功后，再进入主页面。

```
let vc = TUIRoomMainViewController(roomId: roomId, isVideoOn: openCameraSwitch.isOn, isAudioO
n: openMicSwitch.isOn)
TUIRoomCore.shareInstance().setDelegate(vc)
navigationController?.pushViewController(vc, animated: true)
```



步骤6：屏幕分享

1. 调用 TUIRoomCore 的 `startScreenCapture` 实现分享。
2. 房间中其他成员会收到 `onRemoteUserScreenVideoAvailable` 的事件通知。

```
// 按钮点击实现屏幕分享
if #available(iOS 12.0, *) {
// 屏幕分享
let params = TRTCVideoEncParam()
params.videoResolution = TRTCVideoResolution._1280_720
params.resMode = TRTCVideoResolutionMode.portrait
params.videoFps = 10
params.enableAdjustRes = false
```

```
params.videoBitrate = 1500
TUIRoomCore.shareInstance().startScreenCapture(params)
TUIRoomBroadcastExtensionLauncher.launch()
} else {
view.window?.makeToast(.versionLowToastText)
}
```

步骤7：离开房间

- **主持人**调用 destoryRoom 接口解散房间，解散 IM 群聊，退出 TRTC 房间，成员端会收到 onDestroyRoom 回调消息，通知群解散，退出 TRTC 房间。
- **成员**调用 leaveRoom 接口解散房间，退出 IM 群聊，退出 TRTC 房间，其他成员端会收到 onRemoteUserLeave 回调，通知有成员离开房间。

```
if isHomeowner {
TUIRoomCore.shareInstance().destroyRoom { [weak self] _, _ in
guard let self = self else { return }
self.navigationController?.popViewController(animated: true)
}
} else {
TUIRoomCore.shareInstance().leaveRoom { [weak self] _, _ in
guard let self = self else { return }
self.navigationController?.popViewController(animated: true)
}
}
```

多人音视频房间(Android)

最近更新时间：2022-03-14 16:07:51

效果展示

您可以 下载 安装我们的 App 体验多人音视频通话的效果，包括屏幕分享、美颜、低延时视频通话等 TRTC 在多人音视频场景下的相关能力。

方案优势

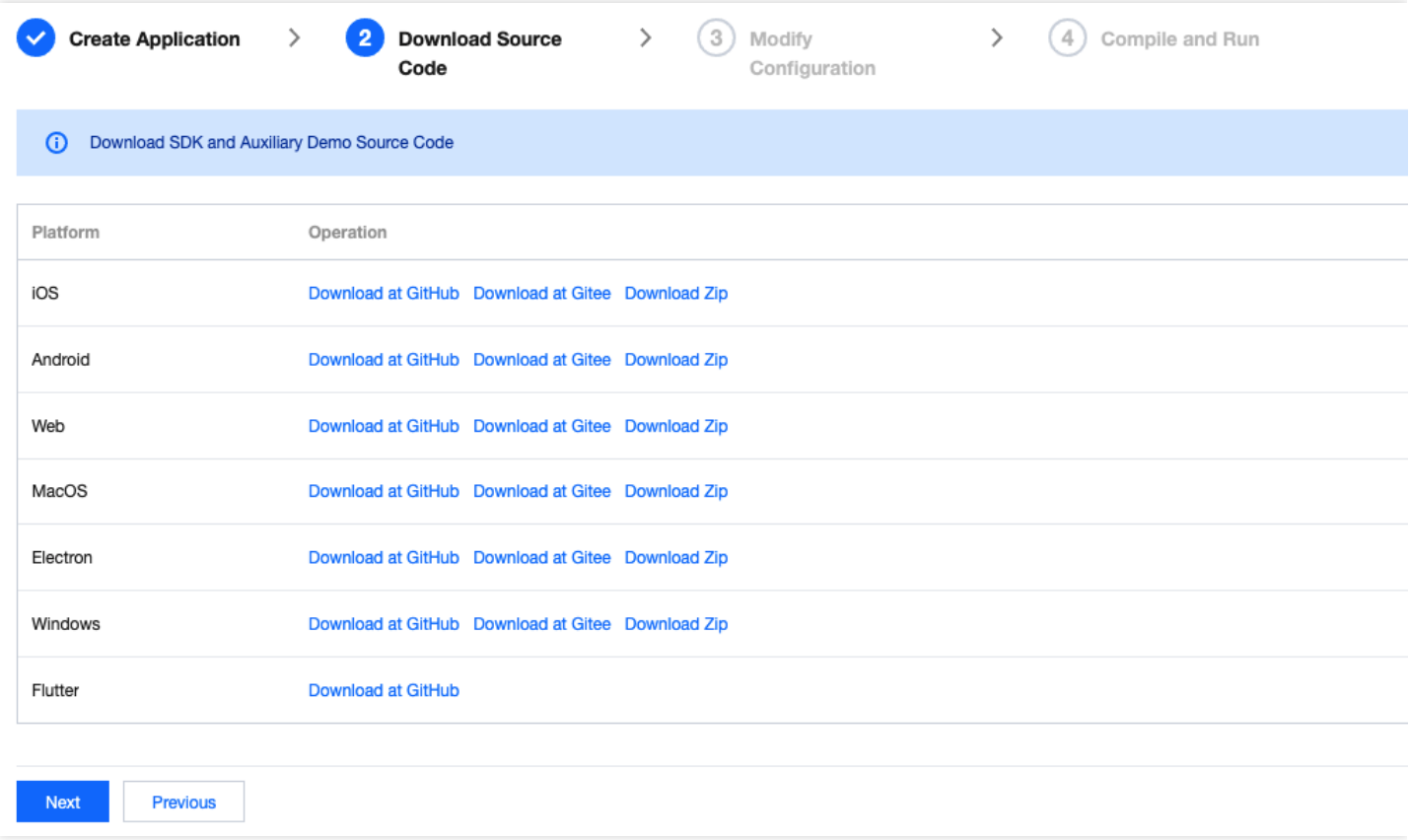
- 集成了超低延时音视频通话、屏幕共享、美颜等能力，覆盖多人音视频房间常见功能。
- 根据需求二次开发，可以快速实现自定义 UI 界面和布局，助力业务快速上线。
- 封装了 TRTC 和 IM 基础 SDK，实现基础的逻辑控制，并提供接口方便调用。

接入指引

如需快速接入多人音视频房间功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用的 App 内的 Module 模块实现自定义 UI 界面。

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
2. 输入应用名称，例如 `TestTUIRoom` ，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

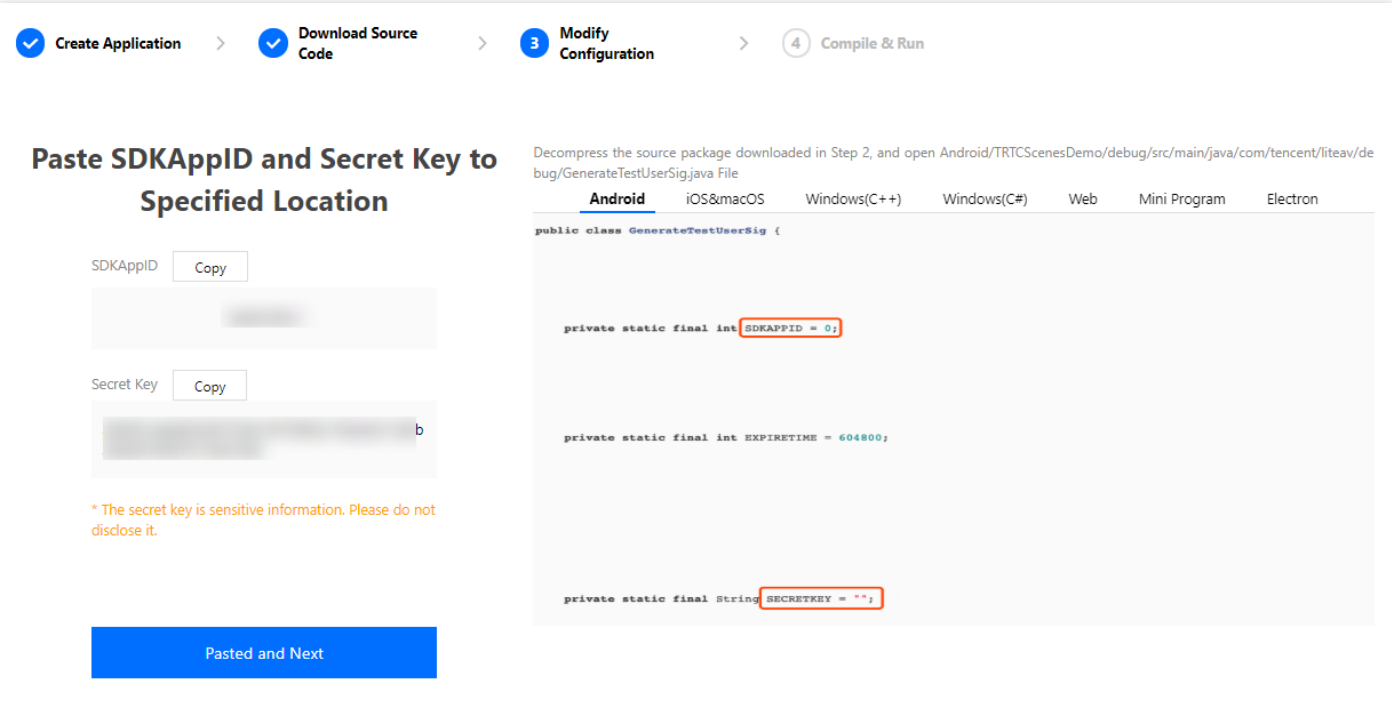
单击进入 TUIRoom，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

说明：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5以上的版本）打开源码工程 `TUIRoom`，单击 **运行** 即可开始调试本 App。

步骤5：修改工程源代码

源码文件夹 `Source` 中包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
remote	远端用户的列表界面
widget	通用的 UI 组件
CreateRoomActivity.java	创建音视频房间的界面
RoomMainActivity.java	音视频房间的主界面
RoomVideoView.java	封装了 TRTC 的 TXCloudVideoView，用于展示自己和远端用户的视频数据
MemberEntity.java	UI 层的用户数据
MemberListAdapter.java	音视频房间主界面的 Adapter

体验应用

> 注意：
> 体验应用至少需要两台设备。

入口界面

请选择“创建房间”或“加入房间”。

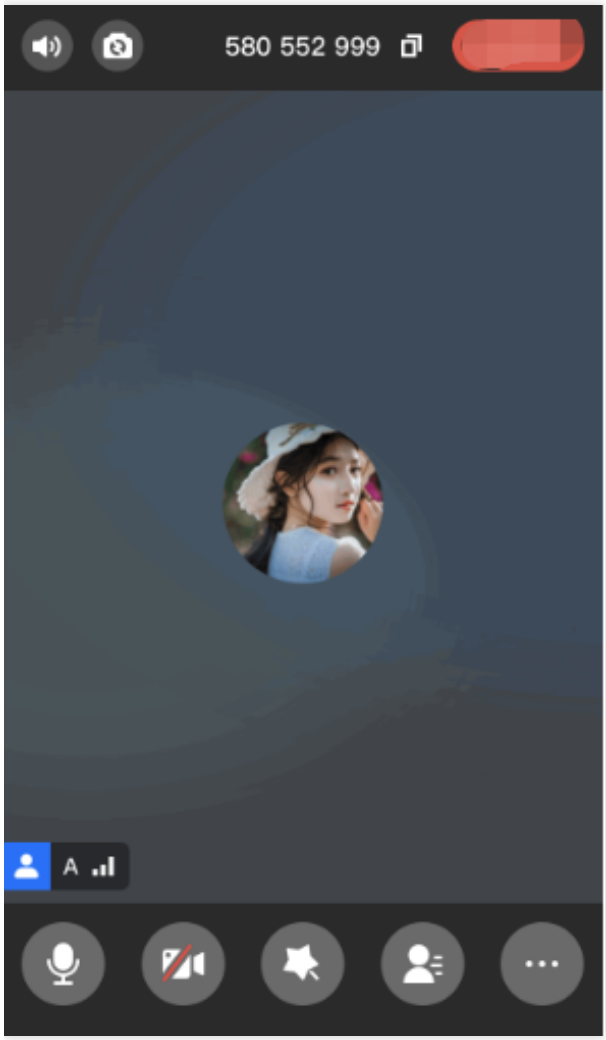
创建房间页面

A 用户创建，房间号会默认生成，单击 **创建房间** 可以进入主页面。

加入房间页面

B 用户输入 A 用户的房间号，单击 **加入房间** 可以进入主页面。

主页面（A 用户）



麦上列表

麦上列表可以展示当前房间内的成员，对方打开摄像头和麦克风后可以看到对方的画面，听到对方的声音。

顶部控制栏

实现了耳麦切换，前后摄像头切换，房间信息，退出的功能。

底部工具栏

实现了控制自己的麦克风/摄像头，美颜，成员列表，设置等功能。

美颜

可在直播中针对画面进行美颜和特效展示。

设置窗口

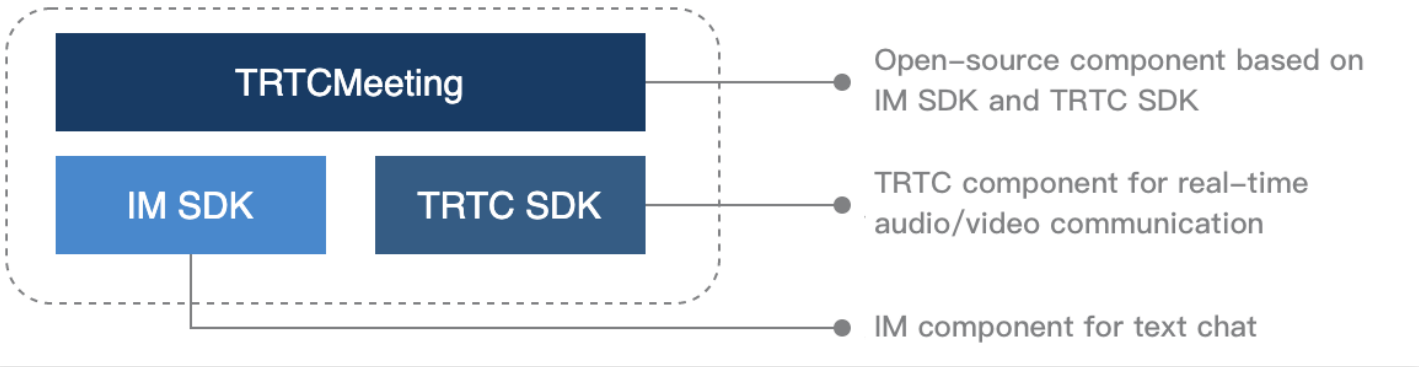
可对音视频相关参数进行设置，支持开启 **屏幕共享**。

退出房间

- **主持人**：解散房间，即所有人退出房间。
- **非主持人**：自己离开房间。

实现自定义 UI 界面

源码 中的 `Source` 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TUIRoomCore，您可以在 `TUIRoomCore.java` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

多人音视频房间组件 TUIRoomCore 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 Maven 仓库依赖**
  i. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
compile "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
compile 'com.tencent.imsdk:imsdk:latest.release'
}
```

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

defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击 **Sync Now** ，自动下载 SDK 并集成到工程里。

- **方法二：通过本地 AAR 依赖**
  若您的开发环境访问 Maven 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

1. 在 `AndroidManifest.xml` 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：

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

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

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

步骤3：导入 TUIRoomCore 组件

拷贝以下目录中的所有文件到您的项目中：

```
src/main/java/com/tencent/liteav/tuiroom/model
```

步骤4：创建并登录组件

调用 TUICore 中的 TUILogin 进行登录，请参考如下示例：

```
TUILogin.init(this, "sdkAppid", null, new V2TIMSDKListener() {
@Override
public void onKickedOffline() {
}
@Override
public void onUserSigExpired() {
}
});
TUILogin.login("userId", "userSig", new V2TIMCallback() {
@Override
public void onError(int code, String msg) {
}
@Override
public void onSuccess() {
}
});
```

步骤5：设置昵称

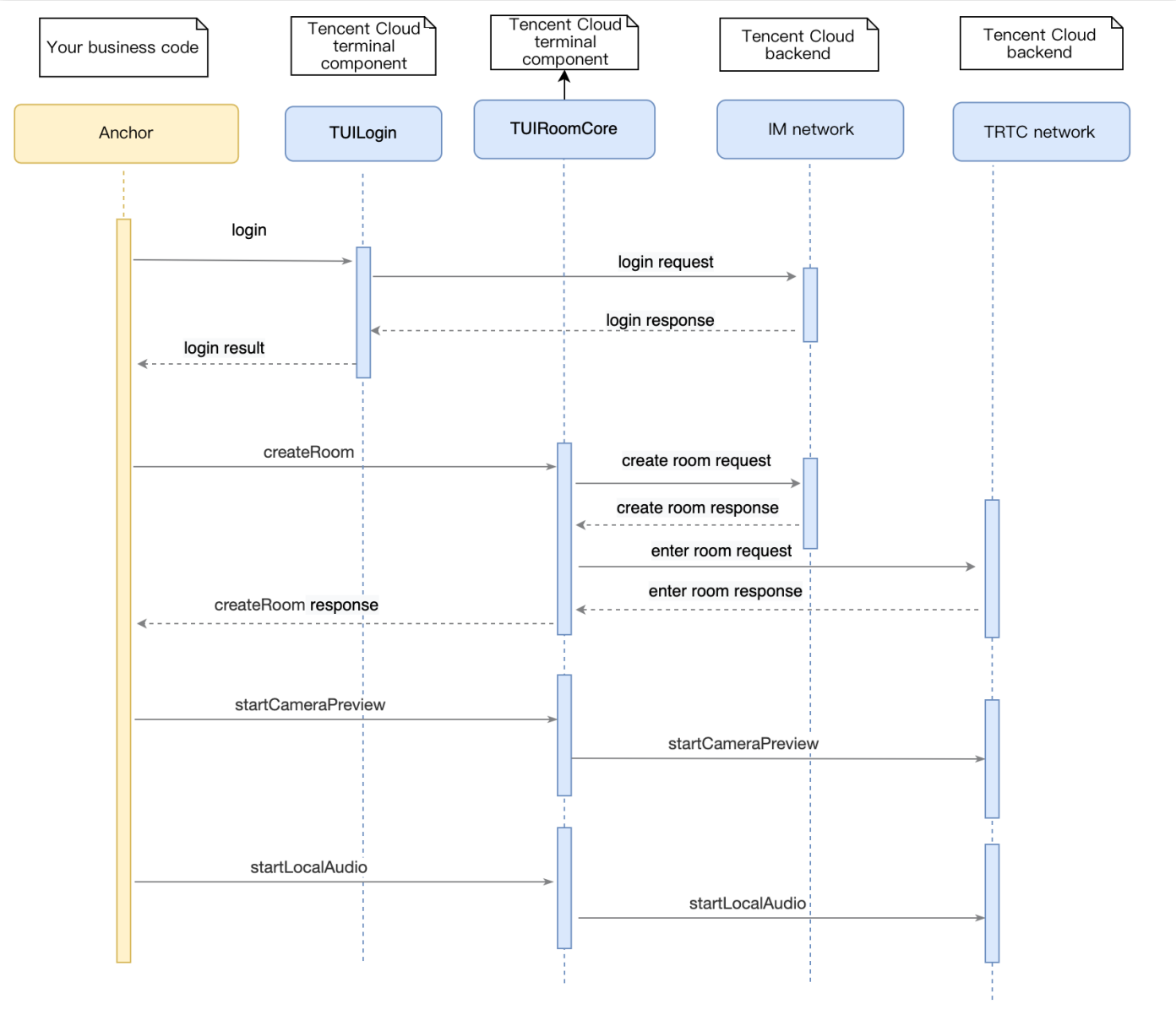
登录成功后，调用 setSelfProfile 设置用户信息。

步骤6：创建房间

1. 成员调用 createRoom 接口创建新的房间，房间创建成功后以主持人身份进入房间，包括聊天室与 TRTC 房间。

```
TUIRoomCore tuiRoomCore = TUIRoomCore.getInstance(context);
tuiCore.setListener(listener);
tuiCore.createRoom("roomId", TUIRoomCoreDef.SpeechMode.FREE_SPEECH,
new TUIRoomCoreCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
// 创建房间成功
} else {
}
}
});
```

2. 调用 startCameraPreview 接口开始本地视频的采集与显示。



步骤7：解散房间

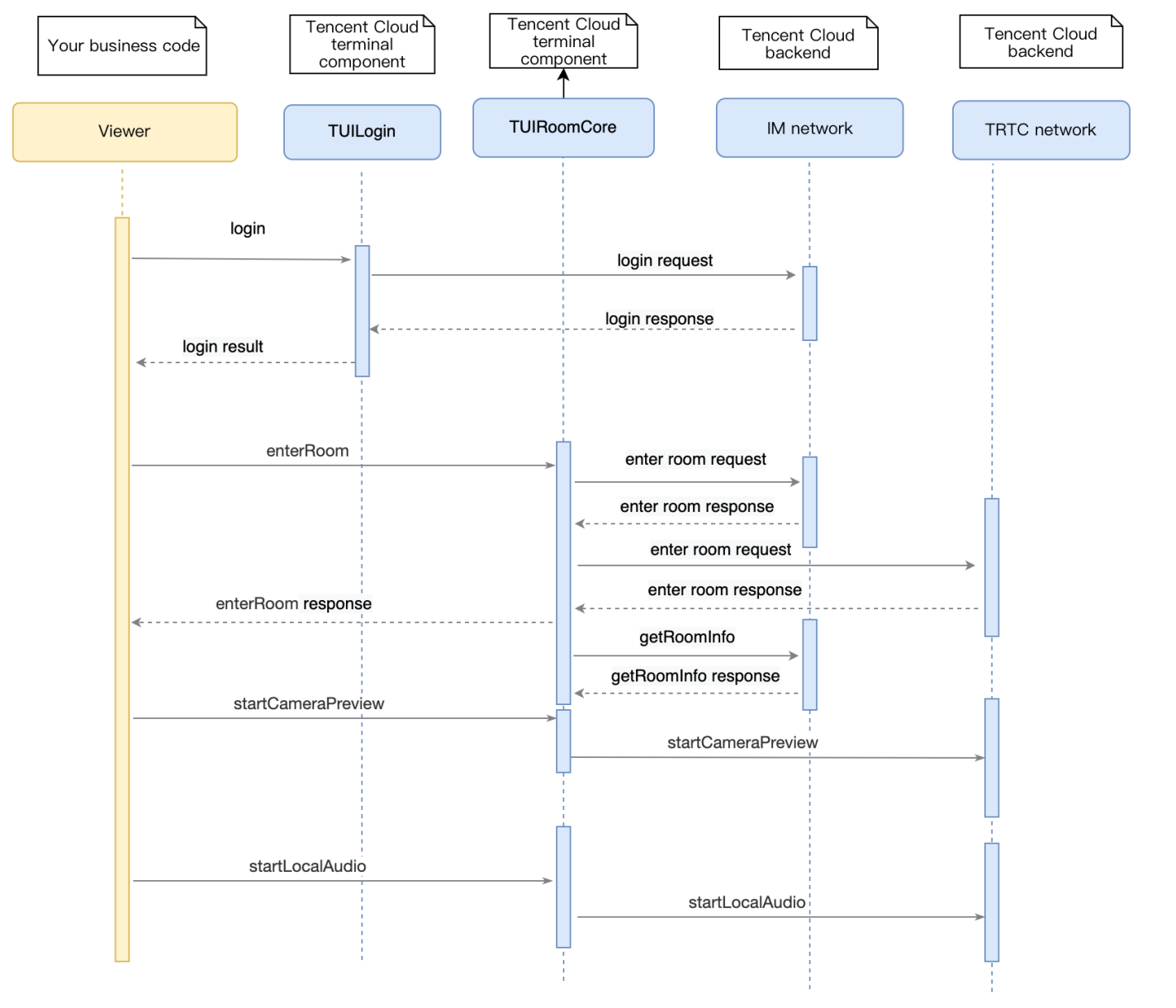
- 主持人调用 destoryRoom 接口解散房间，解散 IM 群聊，退出 TRTC 房间。
- 成员端会收到 onDestroyRoom 回调消息，通知群解散，退出 TRTC 房间。

步骤8：加入房间

1. 加入房间流程与创建流程基本一致，成员需要调用 enterRoom 接口进入房间。
2. 其他成员会收到 onRemoteUserEnter 回调，通知有成员进入房间。

```
TUIRoomCore tuiRoomCore = TUIRoomCore.getInstance(context);
tuiCore.setListener(listener);
tuiCore.enterRoom("roomId", new TUIRoomCoreCallback.ActionCallback() {
```

```
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
// 进入房间成功
} else {
}
}
});
```



步骤9：离开房间

1. 成员调用 leaveRoom 接口退出 IM 房间和 TRTC 房间。
2. 其他成团端收到 onRemoteUserLeave 回调，通知有成员离开房间。

步骤10：屏幕分享

1. 屏幕分享功能需向系统需要申请悬浮窗权限，需要您在 UI 中实现具体的逻辑。
2. 调用 TUIRoomCore的 `startScreenCapture` 实现分享，传入编码参数和录屏过程中的悬浮窗即可实现屏幕分享功能，具体信息请参见 TRTC SDK。
3. 房间中其他成员会收到 onRemoteUserScreenVideoAvailable 的事件通知。

> 注意：
> 屏幕分享和摄像头采集是两个互斥的操作，如果需要打开屏幕分享功能，请先调用 stopCameraPreview 关闭摄像头采集。

```
// 1.在 AndroidManifest.xml 文件中添加 SDK 录屏功能的 activity 和权限
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW">
<application>
<activity android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantActivity"
android:theme="@android:style/Theme.Translucent">
</activity></application>
```

```
// 2.在您的界面中申请悬浮窗的权限
if (Build.VERSION.SDK_INT >= 23) {
if (!Settings.canDrawOverlays(getApplicationContext())) {
Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION, Uri.parse("package:" + getP
ackageName()));
startActivityForResult(intent, 100);
} else {
startScreenCapture();
}
} else {
startScreenCapture();
}

// 3.系统回调结果
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (requestCode == 100) {
if (Build.VERSION.SDK_INT >= 23) {
if (Settings.canDrawOverlays(this)) {
// 用户成功授权
startScreenCapture();
} else {
}
}
}
```

```
}

// 4.打开屏幕分享
private void startScreenCapture() {
TRTCCloudDef.TRTCVideoEncParam encParams = new TRTCCloudDef.TRTCVideoEncParam();
encParams.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720;
encParams.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
encParams.videoFps = 10;
encParams.enableAdjustRes = false;
encParams.videoBitrate = 1200;

TRTCCloudDef.TRTCScreenShareParams params = new TRTCCloudDef.TRTCScreenShareParams();
mTUIRoom.stopCameraPreview();
mTUIRoom.startScreenCapture(encParams, params);
}
```

多人音视频房间(Flutter)

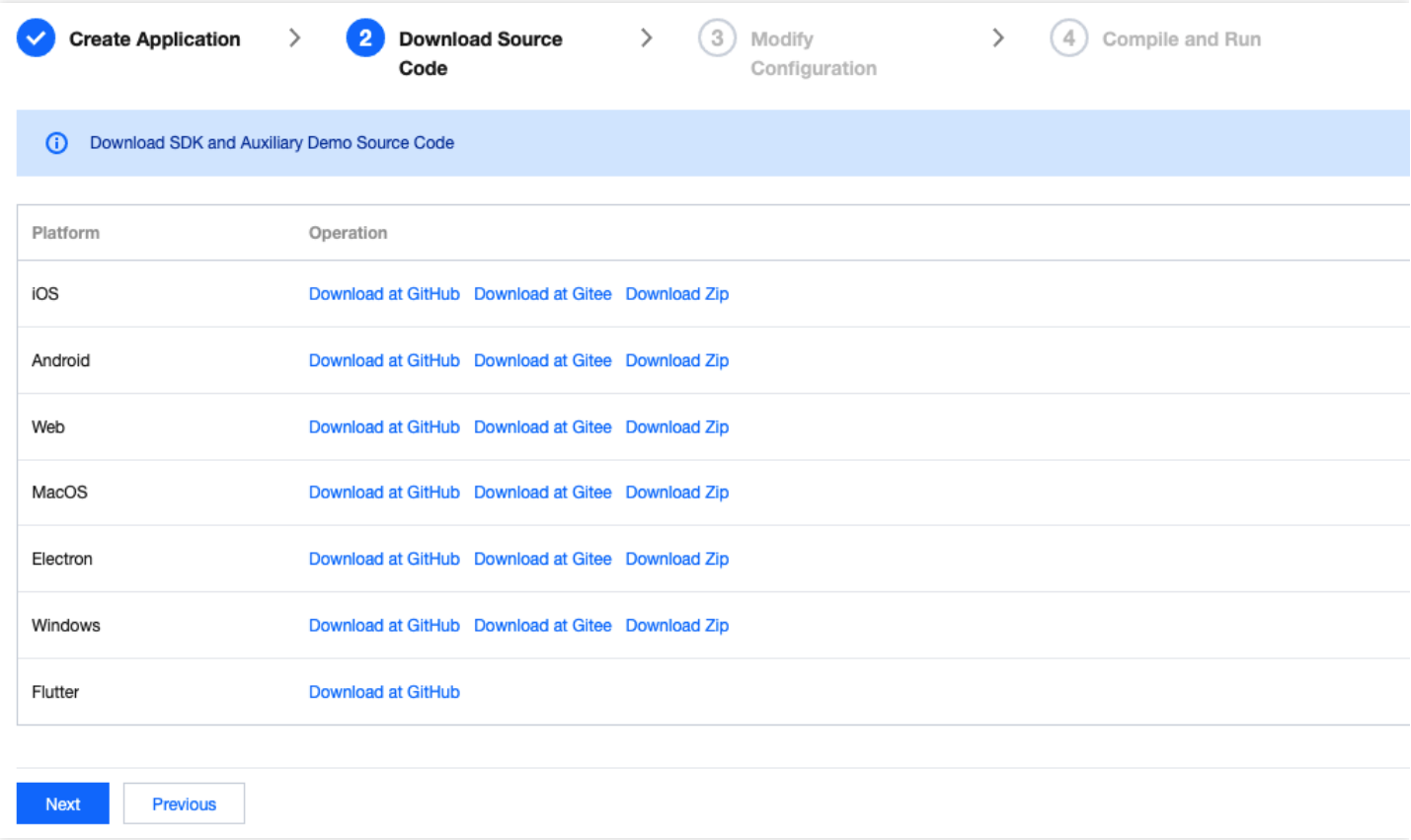
最近更新时间：2022-03-01 12:18:47

您可以 下载 安装我们的 App 体验多人音视频房间的效果，包括屏幕分享、美颜、低延时会议等 TRTC 在多人音视频房间场景下的相关能力。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
2. 输入应用名称，例如 `TestMeetingRoom` ，单击 **创建**。
3. 单击 **已下载，下一步** ，跳过此步骤。



注意：
本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

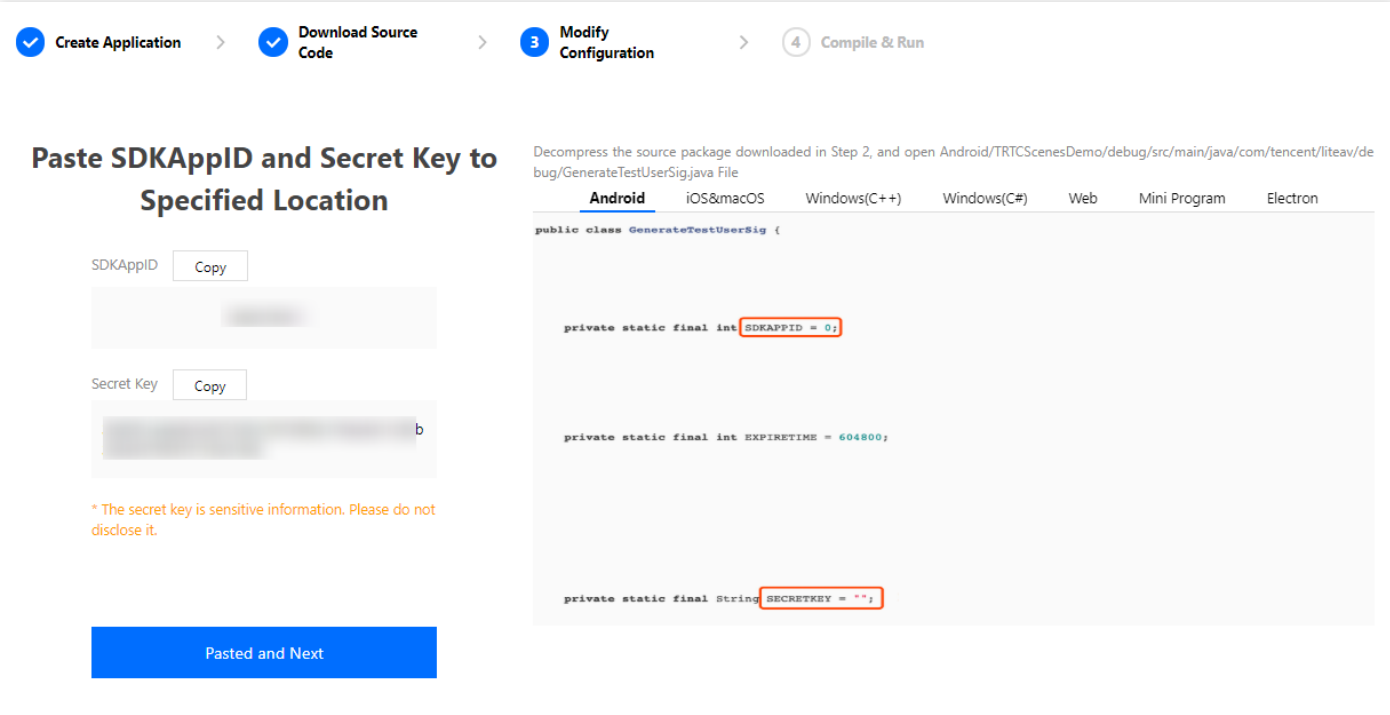
单击进入 TRTCFlutterScenesDemo，Clone 或者下载源码。

步骤3：配置 App 文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `/lib/debug/GenerateTestUserSig.dart` 文件。

3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：
   - SDKAPPID：默认为PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为PLACEHOLDER，请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：编译运行

注意：
Android 端需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。
2. 编译运行调试：
   - iOS 端
   - Android 端
   i. 使用 XCode（11.0及以上的版本）打开源码目录下的 `/ios工程` 。
   ii. 编译并运行 Demo 工程即可。

步骤5：修改 Demo 源代码

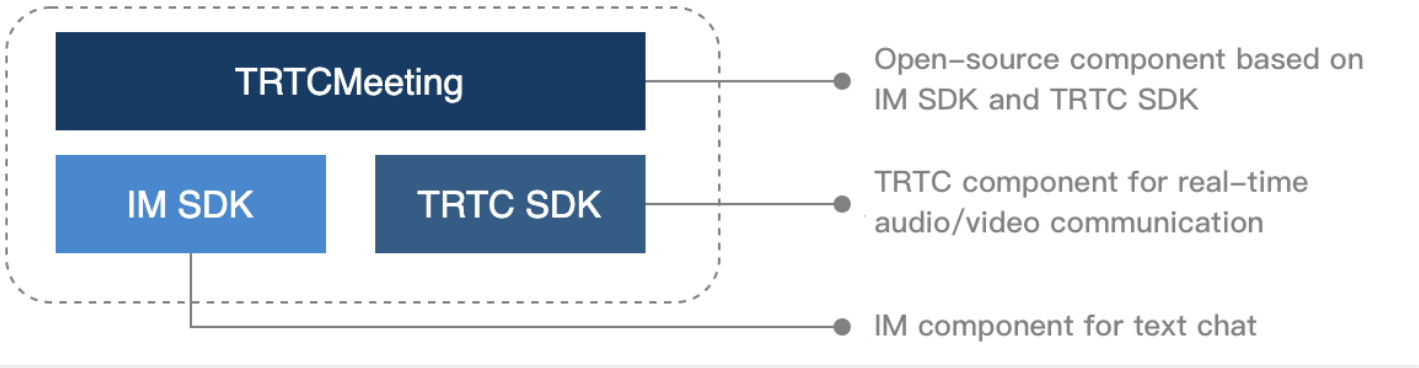
源码中的 TRTCMeetingDemo 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
TRTCMeetingIndex.dart	创建或进入会议界面
TRTCMeetingRoom.dart	视频会议的主界面
TRTCMeetingMemberList.dart	参会人员列表界面
TRTCMeetingSetting.dart	视频会议相关参数设置界面

实现自定义 UI 界面

源码 中的 TRTCMeetingDemo 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCMeeting，您可以在 `TRTCMeeting.dart` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI

界面。



步骤1：集成 SDK

互动直播组件 TRTCMeeting 依赖 TRTC SDK 和 IM SDK，您可以通过配置 `pubspec.yaml` 自动下载更新。

在项目的 `pubspec.yaml` 中写如下依赖：

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

步骤2：配置权限及混淆规则

- iOS 端
- Android 端

需要在 `Info.plist` 中加入对相机和麦克风的权限申请：

```
<key>NSMicrophoneUsageDescriptionkey>
<string>授权麦克风权限才能正常语音通话string>
```

步骤3：导入 TRTCMeeting 组件

拷贝以下目录中的所有文件到您的项目中：

```
lib/TRTCMeetingDemo/model/
```

步骤4：创建并登录组件

1. 调用 `sharedInstance` 接口可以创建一个 TRTCMeeting 组件的实例对象。
2. 调用 `registerListener` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

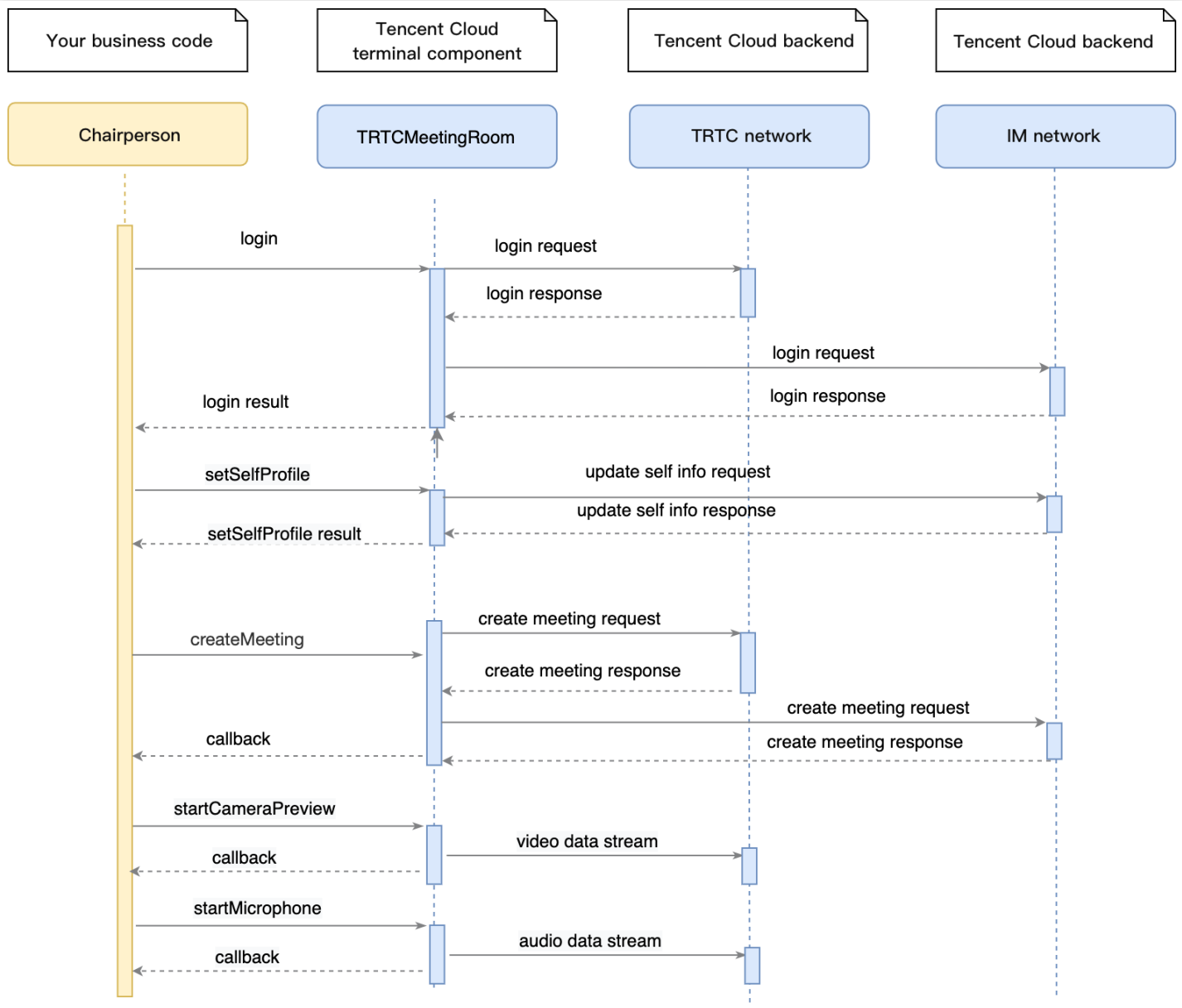
参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

```
TRTCMeeting trtcMeeting = TRTCMeeting.sharedInstance();
trtcMeeting.registerListener(onListener);
ActionCallback res = await trtcMeeting.login(
GenerateTestUserSig.sdkAppId,
userId,
GenerateTestUserSig.genTestSig(userId),
);
if (res.code == 0) {
// 登录成功
}
```

步骤5：创建多人会议

1. 主持人执行 步骤4 登录后，可以调用 setSelfProfile 设置自己的昵称和头像。
2. 主持人调用 createMeeting 创建新的会议房间。
3. 主持人可以调用 startCameraPreview 进行视频画面的采集，也可以调用 startMicrophone 进行声音的采集。
4. 如果主持人有美颜的需求，界面上可以配置美颜调节按钮调用，通过 getBeautyManager 进行美颜设置。

> 说明：
> 非企业版 SDK 不支持变脸和贴图挂件功能。

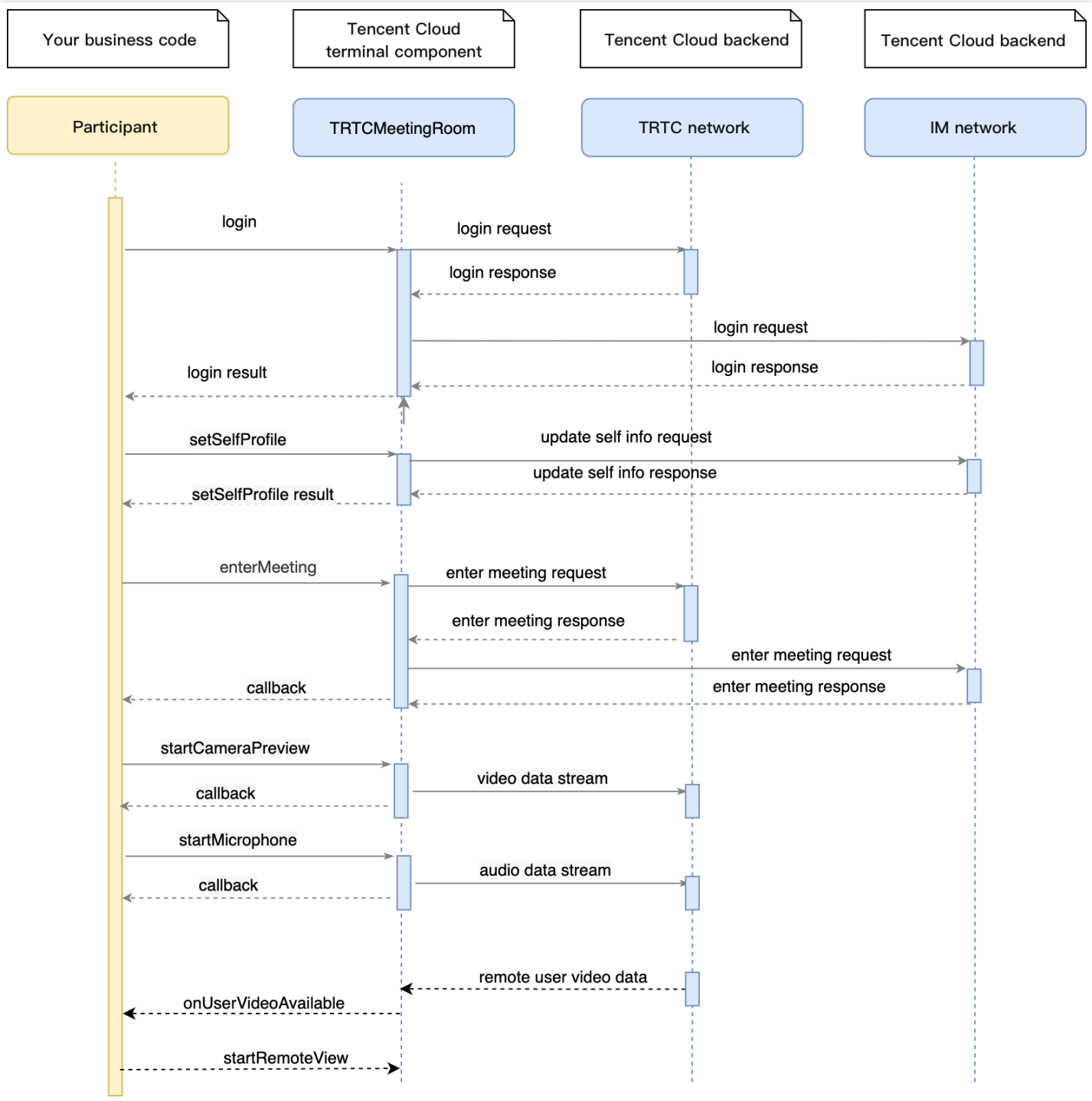



```
// 1. 主持人设置昵称和头像
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 2. 主持人创建会议
ActionCallback res = await trtcMeeting.createMeeting(roomId);
if (res.code == 0) {
// 创建会议成功
// 3. 打开摄像头和音频采集
trtcMeeting.startCameraPreview(true, TRTCCloudVideoViewId);
trtcMeeting.startMicrophone();
// 4. 设置美颜
trtcMeeting.getBeautyManager().setBeautyStyle(TRTCCloudDef.TRTC_BEAUTY_STYLE_PITU);
```

```
trtcMeeting.getBeautyManager().setBeautyLevel(6);
}
```

步骤6：参会成员进入多人会议

1. 参会成员执行 步骤4 登录后，可以调用 setSelfProfile 设置自己的昵称和头像。
2. 参会成员调用 enterMeeting 并传入会议房间号即可进入会议房间。
3. 参会成员可以调用 startCameraPreview 进行视频画面的采集，也可以调用 startMicrophone 进行声音的采集。
4. 如果有其他的参会成员打开了摄像头，会收到 onUserVideoAvailable 的事件，此时可以调用 startRemoteView 并传入 userId 开始播放。




```
// 1. 参会成员设置昵称和头像
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 2. 参会成员调用 enterMeeting 进入会议房间
ActionCallback res = await trtcMeeting.enterMeeting(roomId);
if (res.code == 0) {
// 进入会议成功
// 3. 打开摄像头和音频采集
```

```
trtcMeeting.startCameraPreview(true, TRTCCloudVideoViewId);
trtcMeeting.startMicrophone();
// 4. 设置美颜
trtcMeeting.getBeautyManager().setBeautyStyle(TRTCCloudDef.TRTC_BEAUTY_STYLE_PITU);
trtcMeeting.getBeautyManager().setBeautyLevel(6);
}

// 5. 参会成员收到其他成员摄像头打开的通知，开始播放
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
switch (type) {
case TRTCMeetingDelegate.onUserVideoAvailable:
if (param['available']) {
trtcMeeting.startCameraPreview(
param['userId'],
TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
TRTCCloudVideoViewId
);
} else {
trtcMeeting.stopRemoteView(
param['userId'],
TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG
);
}
break;
}
}
```

步骤7：屏幕分享

1. 屏幕分享功能需向系统申请悬浮窗权限，需要您在 UI 中实现具体的逻辑。
2. 调用 startScreenCapture，传入编码参数和录屏过程中的悬浮窗即可实现屏幕分享功能，具体信息请参见 TRTC SDK。
3. 会议中其他成员会收到 onUserVideoAvailable 的事件通知。

> 注意：
> 屏幕分享和摄像头采集是两个互斥的操作，如果需要打开屏幕分享功能，请先调用 stopCameraPreview 关闭摄像头采集。详情请参见 TRTC SDK。

```
await trtcMeeting.stopCameraPreview();
trtcMeeting.startScreenCapture(
videoFps: 10,
```

```
videoBitrate: 1600,
videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720,
videoResolutionMode: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT,
appGroup: iosAppGroup,
);
```

步骤8：实现文字聊天和禁言消息

- 通过 sendRoomTextMsg 可以发送普通的文本消息，所有在该会议内的参会人员均可以收到 onRecvRoomTextMsg 回调。即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
trtcMeeting.sendRoomTextMsg('Hello Word!');
// 接收端：监听文本消息
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
switch (type) {
case TRTCMeetingDelegate.onRecvRoomTextMsg:
print('收到来自' + param['userName'] + '的消息：' + param['message']);
break;
}
}
```

- 通过 sendRoomCustomMsg 可以发送自定义（信令）的消息，所有在该会议内的参会人员均可以收到 onRecvRoomCustomMsg 回调。自定义消息常用于传输自定义信令，例如用于禁言之类的会场控制等。

```
// 发送端：您可以通过自定义 cmd 来区分禁言通知
// eg: "CMD_MUTE_AUDIO"表示禁言通知
trtcMeeting.sendRoomCustomMsg('CMD_MUTE_AUDIO', '1');
// 接收端：监听自定义消息
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
switch (type) {
case TRTCMeetingDelegate.onRecvRoomCustomMsg:
if (param['command'] == 'CMD_MUTE_AUDIO') {
// 收到禁言通知
print('收到来自' + param['userName'] + '的禁言通知：' + param['message']);
trtcMeeting.muteLocalAudio(message == '1');
}
break;
}
}
```

多人音视频房间(Windows&Mac)

最近更新时间：2022-02-16 09:45:31

本文介绍 PC 端 TUIRoom 组件，是一款布局灵活、适用性强的音视频沟通协作工具，可用于协同办公、远程招聘、远程问诊、保险理赔、在线客服、视频面试、数字政务、金融数字化、在线会议、在线教育等场景。与各行业场景深度融合，助力企业降本增效，加快数字化转型，提升竞争力。
您可以 下载 安装我们的 App 体验。

方案优势

- 集成了超低延时音视频通话、聊天室、屏幕共享、美颜、设备检测、数据统计等能力，覆盖多人音视频房间常见功能。
- 根据需求二次开发，可以快速实现自定义 UI 界面和布局，助力业务快速上线。
- 封装了 TRTC 和 IM 基础 SDK，实现基础的逻辑控制，并提供接口方便调用。

接入指引

如需快速接入多人音视频房间功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用的 App 内的 Module 模块实现自定义 UI 界面。

步骤1：下载 App 源码

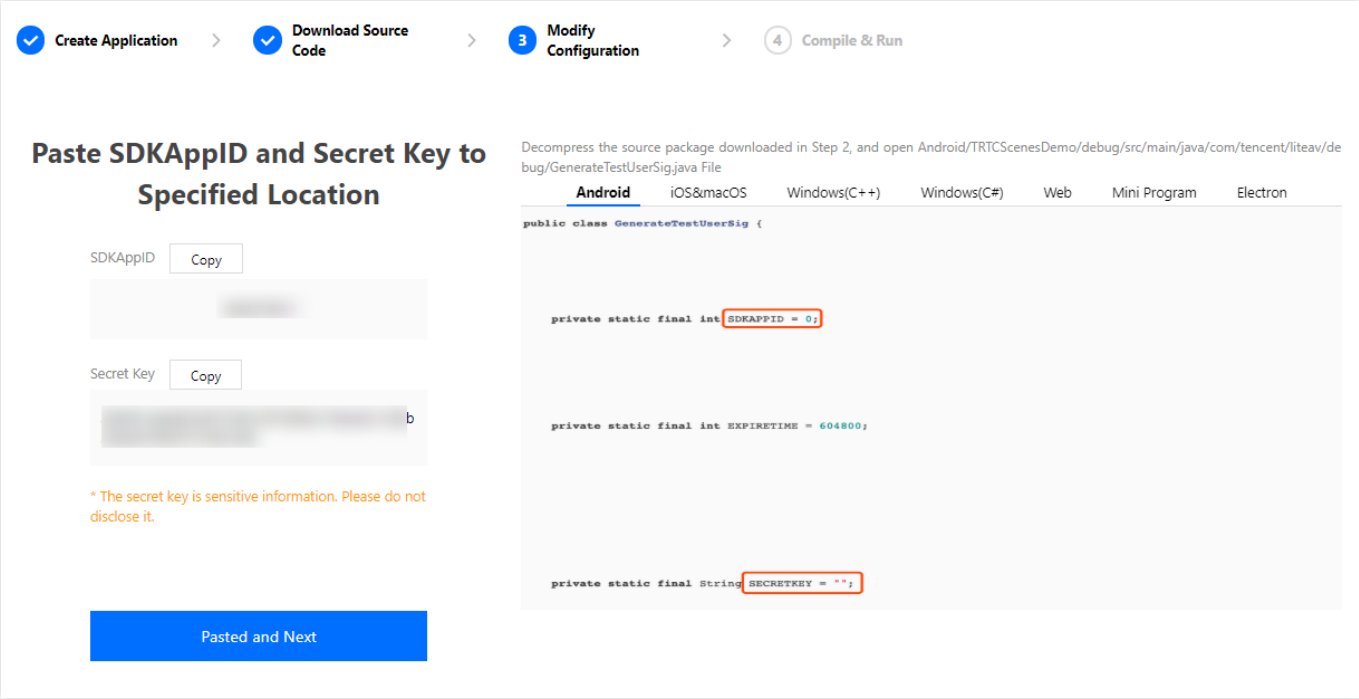
单击进入 RoomApp，Clone 或者下载源码。

步骤2：配置 App 文件

- Windows 端配置
- Mac 端配置

1. 找到并打开 `Windows¥RoomApp¥utils¥usersig¥win¥GenerateTestUserSig.h` 文件。
2. 设置 `GenerateTestUserSig.h` 中的参数：
   - **SDKAPPID**：默认为0，请设置为实际的 SDKAPPID。

- **SECRETKEY**：默认为空，请设置为实际的 SECRETKEY。



注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤3：配置并运行 App

- Windows 端配置
- Mac 端配置

1. 编译环境使用 Virtual Studio（最低为2015版本）。
2. App 的 UI 使用的是当前主流的 QT 框架，版本为5.9版本，请下载并安装 QT，配置 QT 工具到 VS 中。或者根据您的版本，修改 App 工程中 QT 配置，即可完成配置。
3. 打开 RoomApp 下的 RoomApp.vcxproj 源码工程，编译并运行程序。

步骤4：修改 App 源代码

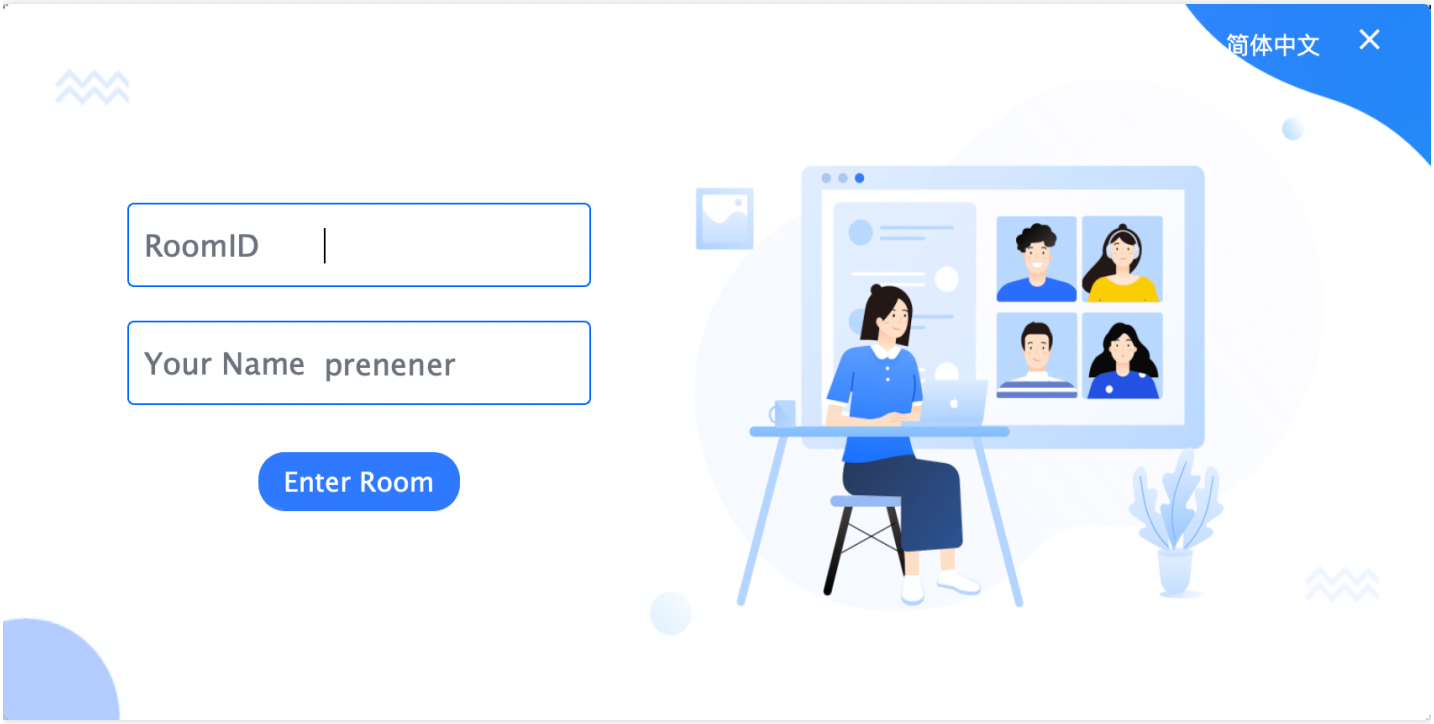
- Module 层为对 TRTC 和 IM 的封装，实现了基础的逻辑控制，并提供了功能接口，方便 UI 层调用。
- App 模块是 UI 模块，包含了界面的功能实现，您可以根据需求进行二次开发，或者替换整体 UI。

体验应用

> 注意：
> 体验应用至少需要两台设备。

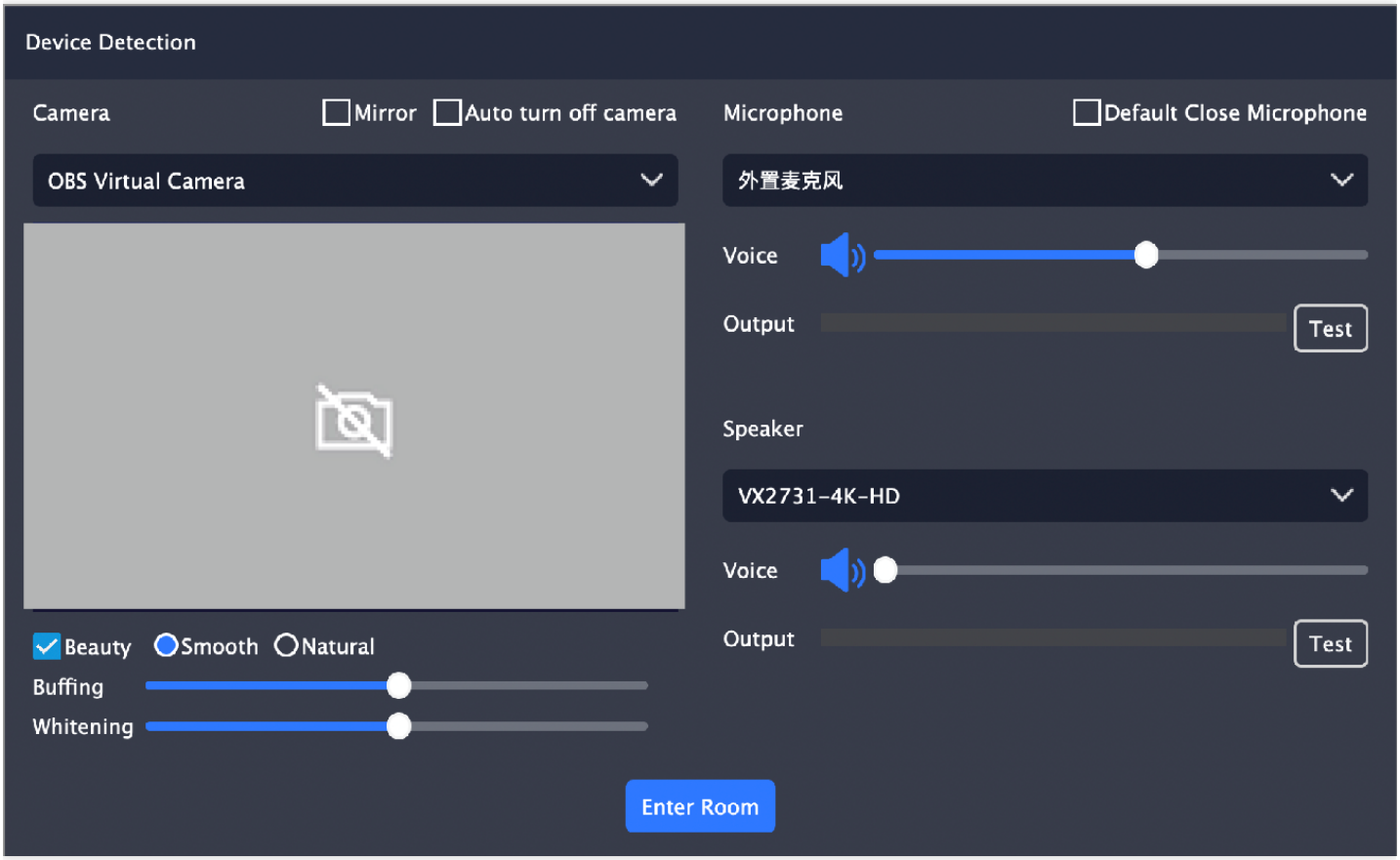
登录界面

输入房间号和用户名（请确保用户名唯一性，不能与其他用户重复）并登录，如图示。



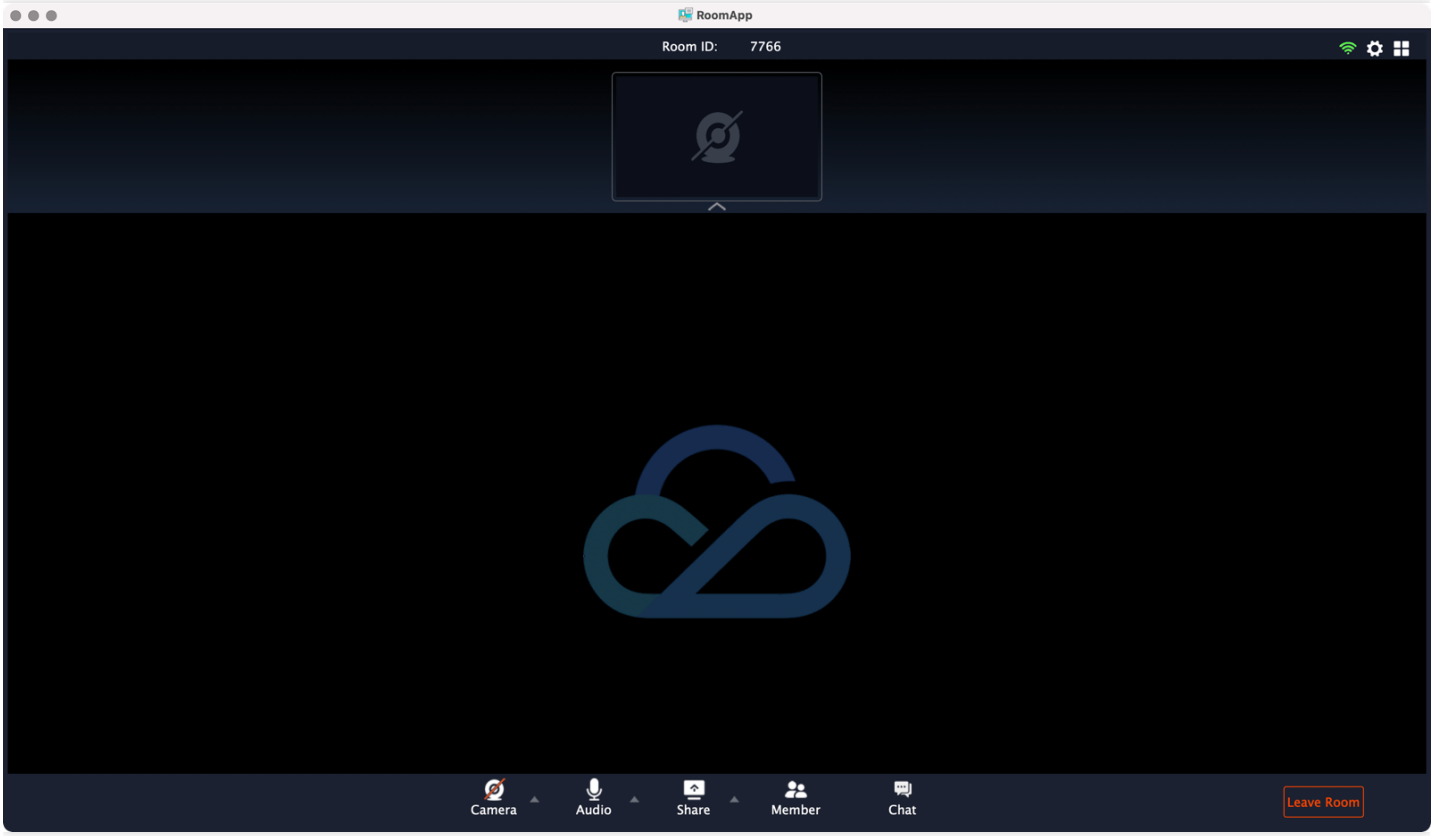
设备检测页面

此处您可以进行设备检测，或根据需求，设置进入房间使用的设备，设置美颜效果等。



主界面

第一个进入房间的为主持人，主界面包括麦上列表，底部工具栏，顶部工具栏等。



麦上列表

麦上列表可以展示当前房间内的成员，对方打开摄像头和麦克风后可以看到对方的画面，听到对方的声音。

底部工具栏

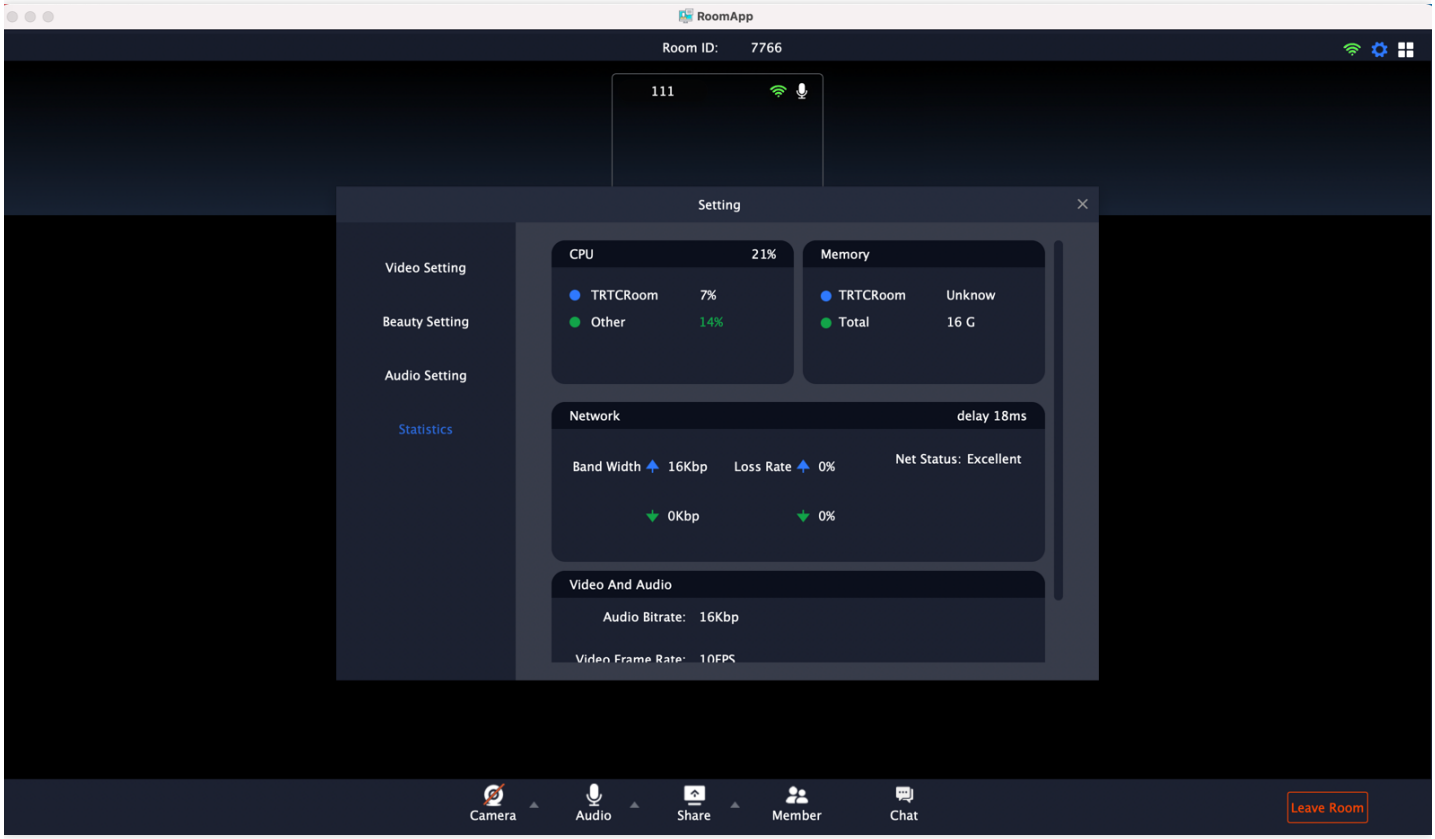
实现了控制自己的麦克风/摄像头，打开屏幕共享，成员列表，聊天室等功能。

顶部控制栏

实现了网络信息，设置页面，麦上列表布局的功能。

设置窗口

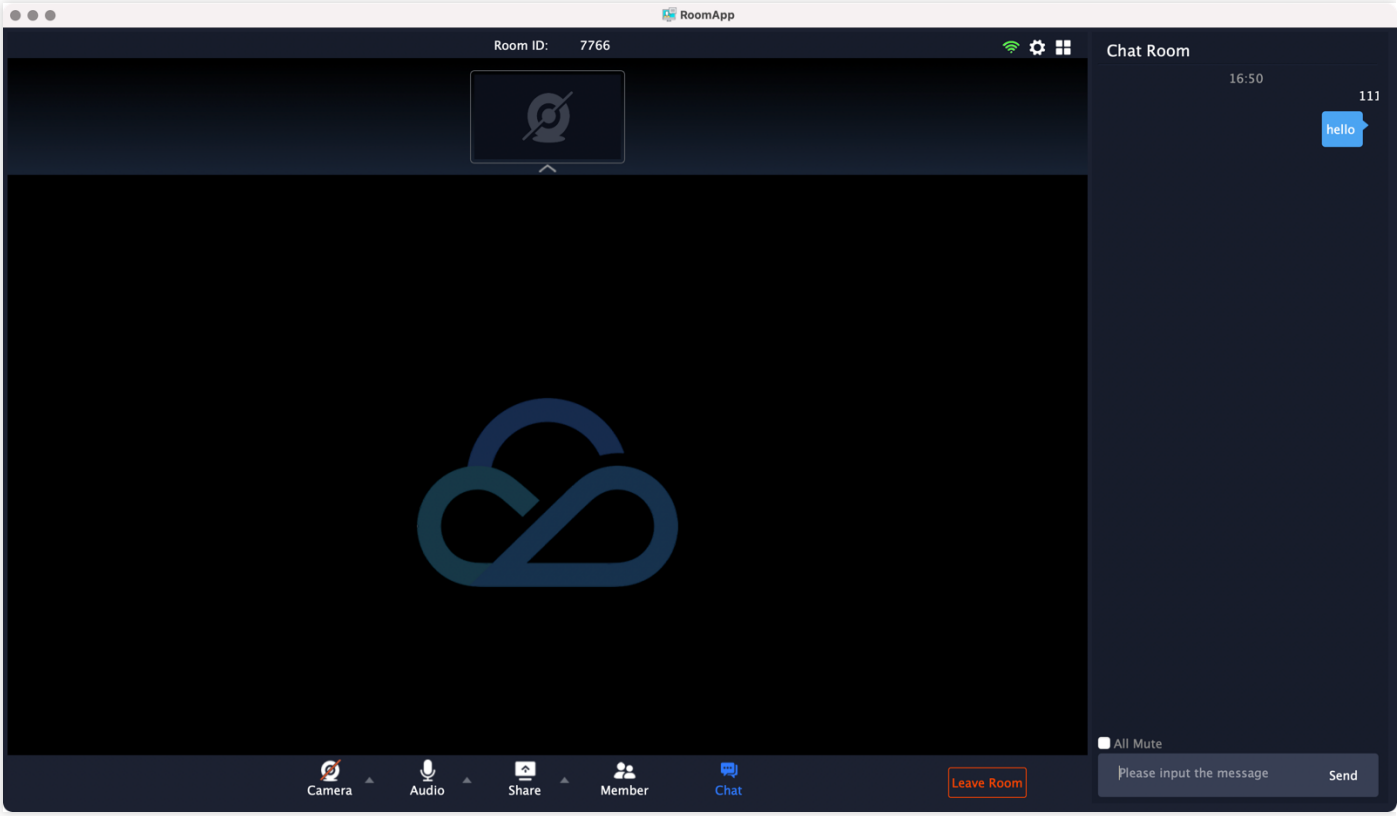
设置窗口可以对选择设备，设置美颜等参数。



聊天室

- 聊天室可以进行群成员聊天。

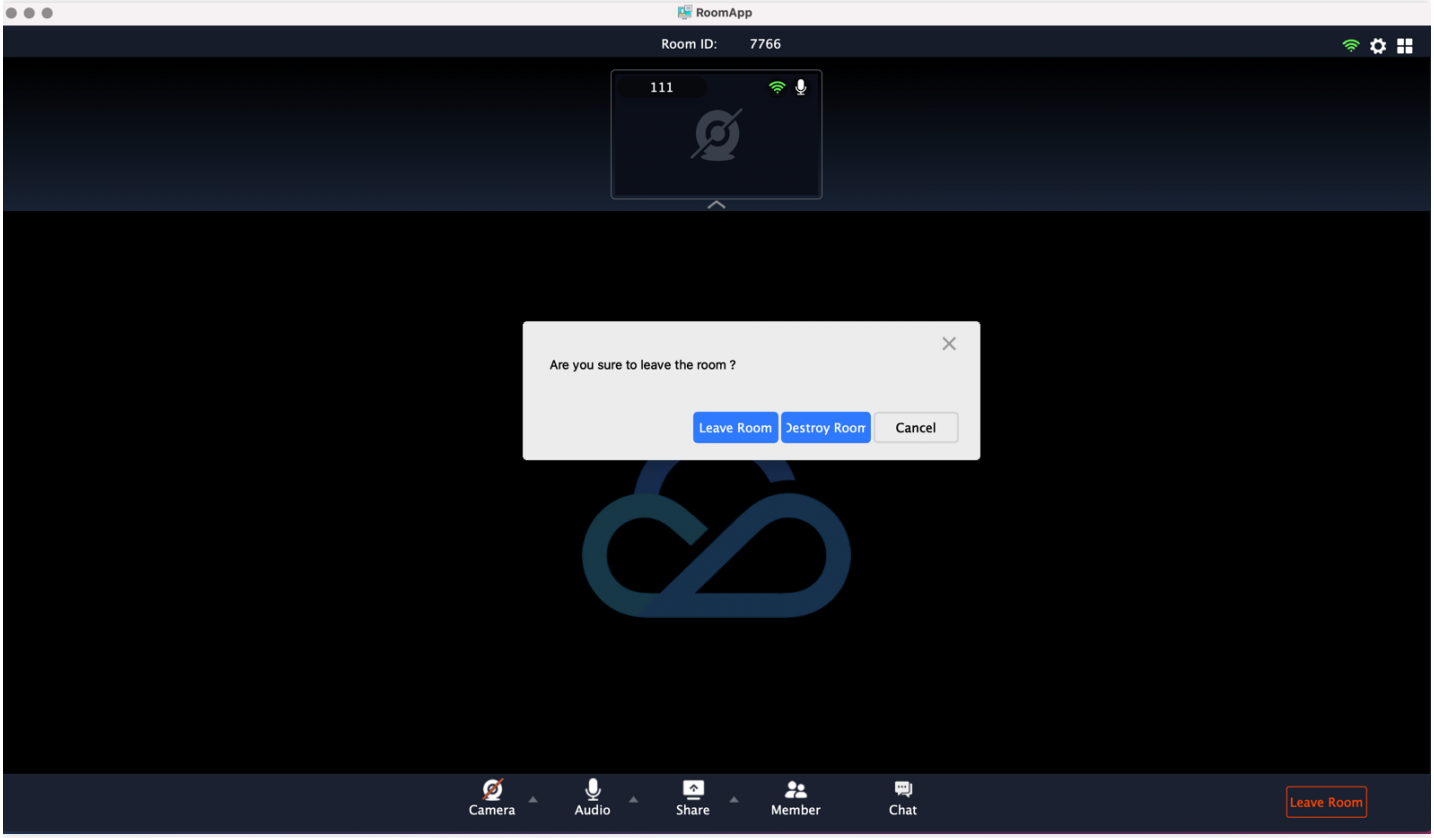
- 主持人可以对全员进行禁言/解除禁言。



退出房间

主持人退出房间时可以有两种选择。

- **解散房间**：即所有人退出房间，所有人都需要退出。
- **自己离开房间**：将房间主持人权限转交给其他用户作为主持人。



实现自定义 UI 界面

源码中的 Module 模块包含了对 TRTC SDK 以及 IM SDK 的封装，您可以在 `TUIRoomCore.h` 、 `TUIRoomCoreCallback.h` 、 `TUIRoomDef.h` 等文件中查看该模块提供的接口函数以及其他定义，并使用对应接口实现自定义 UI 界面。

步骤1：集成SDK

从官网下载 TRTC SDK 和 IM SDK，替换外层的 SDK 目录中 IM SDK 和 LiteAVSDK 目录下的文件。

> 注意：
> IM SDK 使用 C++ 版本。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：工程配置SDK目录以及库文件

- **Windows 端**：在 VS 中配置头文件目录，静态库目录，以及依赖的静态库文件。
- **Mac 端**：在 pro 和 pri 文件中增加依赖库和文件目录。

步骤3：创建并登录

1. 使用 `IUIRoomCore::Instance` 接口创建并使用 IUIRoomCore 单例。
2. 调用 AddCallback 接口设置回调类，接受 SDK 的回调通知。
3. 调用 Login 接口进行登录。

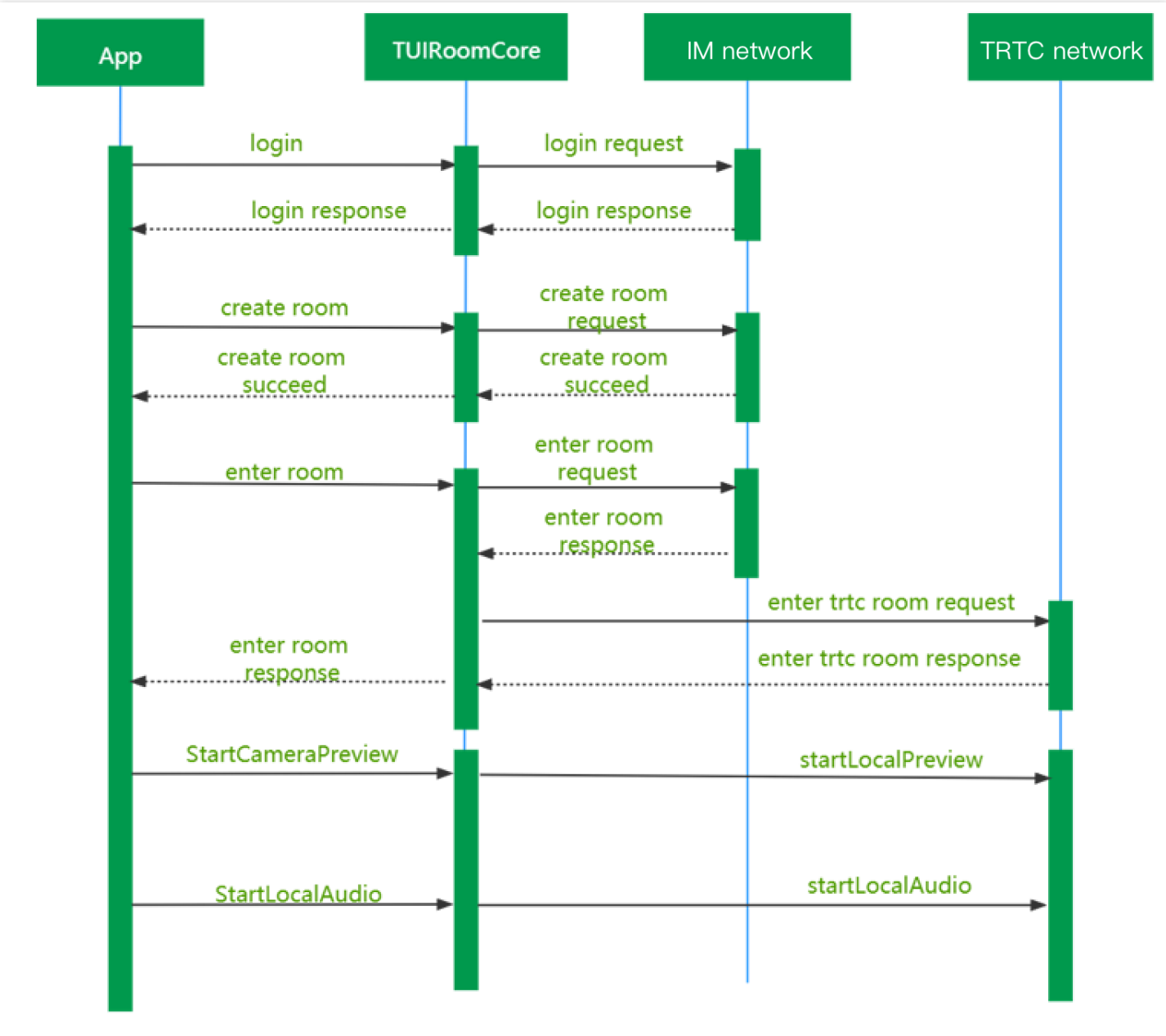
参数名	说明
sdk_appid	您可以在 实时音视频控制台 中查看 SDKAppID。
user_id	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
user_sig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

步骤4：设置昵称

登录成功后，调用 SetSelfProfile 设置用户信息。

步骤5：创建房间

1. 成员调用 CreateRoom 接口创建新的房间，房间创建成功后以主持人身份进入房间，包括聊天室与 TRTC 房间。
2. 成员调用 StartCameraPreview 接口开始本地视频的采集与显示。



步骤6：解散房间

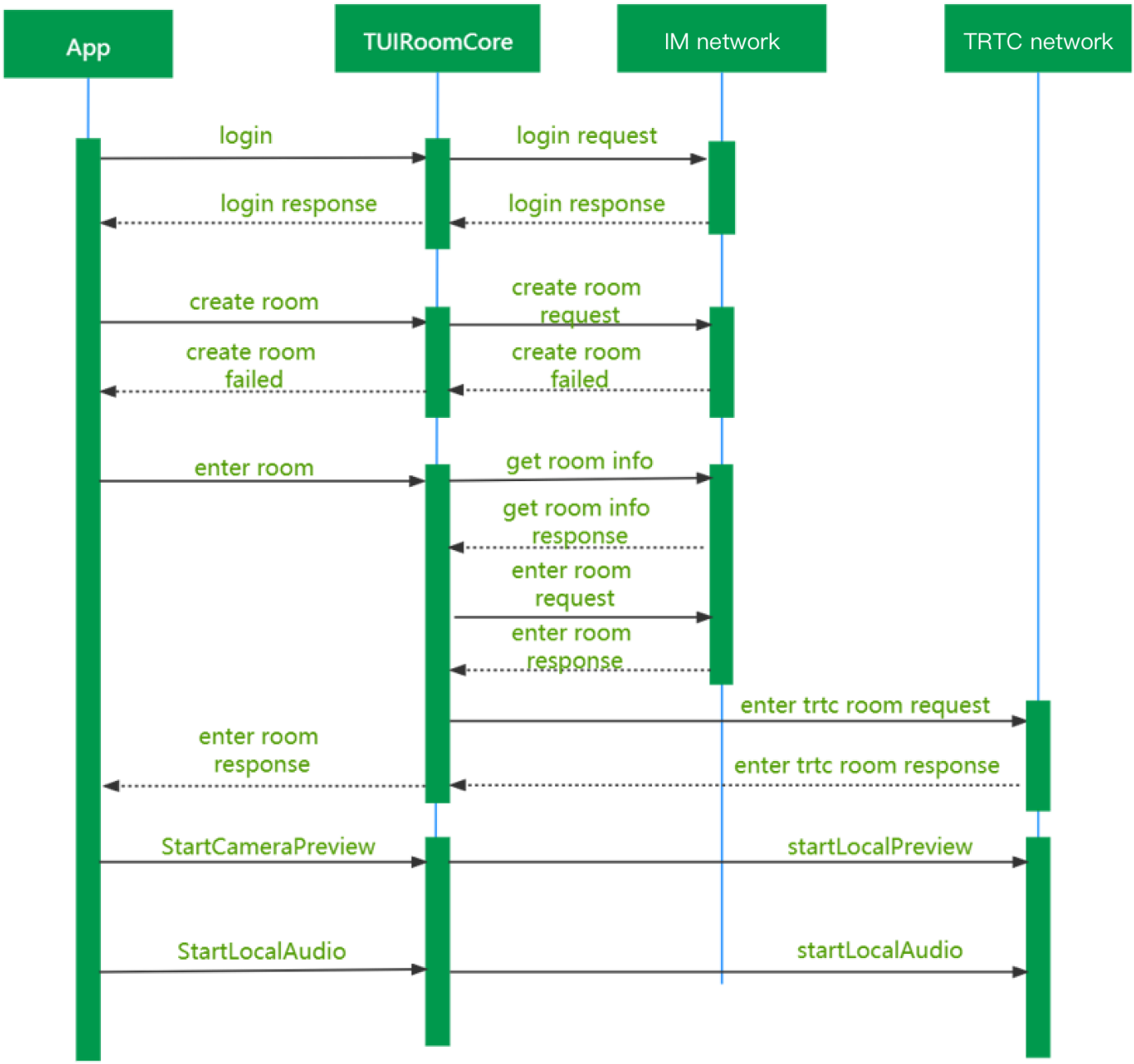
1. 主持人调用 DestoryRoom 接口解散房间，解散 IM 群聊，退出 TRTC 房间
2. 成员端会收到 OnExitRoom 回调消息，通知群解散，退出 TRTC 房间

步骤7：转让房间

1. 主持人调用 TransferRoomMaster 接口将房间转让给其他人，用于离开房间的操作。
2. 成员端会收到 OnRoomMasterChanged 回调消息，通知房间主持人更改。
3. 更改后的主持人获取到主持人权限，可以对群成员进行控制

步骤8：加入房间

1. 加入房间流程与创建流程基本一致，成员在进入房间时都需要先去创建房间，当创建失败后以成员身份进入房间。
2. 创建房间失败后会先获取房间信息，然后再进入房间。
3. 其他成员会收到 OnRemoteUserEnter 回调，通知有成员进入房间。



步骤9：离开房间

1. 成员调用 LeaveRoom 接口退出 IM 房间和 TRTC 房间。
2. 其他成团端收到 OnRemoteUserLeave 回调，通用有成员离开房间。

步骤10：屏幕分享

1. 调用 StartScreenCapture 接口进行屏幕分享。
2. 其他成员收到 OnRemoteUserScreenAvailable 回调，通知有成员正在分享屏幕。

> 说明：
> 屏幕分享模块需要做窗口选择的逻辑，具体实现请参考 App 中的 `ScreenShareWindow.h` 的实现。

步骤11：实现文字聊天和禁言消息

1. 调用 SendChatMessage 接口发送文字，房间内的成员可以收到 OnRecevieChatMessage 回调消息，获取聊天消息并显示。
2. 主持人可以调用 MuteChatRoom 接口进行聊天禁言或者解除禁言，成员端收到 OnChatRoomMuted 回调，UI 层做对应的处理。

TUIRoom (iOS)

最近更新时间：2022-03-01 12:18:48

TUIRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主持人创建房间，进入房间人员输入房间号后进入房间。
- 进入房间人员之间进行屏幕分享。
- 支持发送各种文本消息和自定义消息。

TUIRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 多人音视频房间(iOS)。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频房间组件。
- IM SDK：使用 IM SDK 实现聊天室的功能（**IM SDK 使用 iOS 版本**）。

TUIRoom API 概览

TUIRoomCore 基础函数

API	描述
shareInstance	获取单例对象。
destroyInstance	销毁单例对象。
setDelegate	设置事件回调。

房间相关接口函数

API	描述
createRoom	创建房间（主持人调用）。
destroyRoom	销毁房间（主持人调用）。
enterRoom	进入房间（进入房间成员调用）。
leaveRoom	离开房间（其他房间成员调用）。
getRoomInfo	获取房间信息。
getRoomUsers	获取房间内所有成员信息。
getUserInfo	获取某个用户的信息。

API	描述
transferRoomMaster	转移主持人权限（主持人调用）。

本地音视频操作接口

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startLocalAudio	开启麦克风采集。
stopLocalAudio	停止麦克风采集。
setVideoMirror	设置本地画面镜像预览模式。
setSpeaker	设置开启扬声器。

远端用户相关接口

API	描述
startRemoteView	订阅并播放指定成员的远端视频画面。
stopRemoteView	取消订阅并停止播放远端视频画面。

发送聊天消息接口

API	描述
sendChatMessage	发送聊天消息。
sendCustomMessage	发送自定义消息。

场控相关接口

API	描述
muteUserMicrophone	禁用/恢复某用户的麦克风。
muteAllUsersMicrophone	禁用/恢复所有用户的麦克风，并且状态会同步到房间信息中。
muteUserCamera	禁用/恢复某用户的摄像头。

API	描述
muteAllUsersCamera	禁用/恢复所有用户的摄像头，并且状态会同步到房间信息中。
muteChatRoom	开启/停止聊天室禁言（主持人调用）。
kickOffUser	移除房间内的某人（主持人调用）。
startCallingRoll	主持人开始点名。
stopCallingRoll	主持人结束点名。
replyCallingRoll	成员回复主持人点名。
sendSpeechInvitation	主持人邀请成员发言。
cancelSpeechInvitation	主持人取消邀请成员发言。
replySpeechInvitation	成员同意/拒绝主持人的申请发言。
sendSpeechApplication	成员申请发言。
replySpeechApplication	主持人同意/拒绝成员的申请发言。
forbidSpeechApplication	主持人禁止申请发言。
sendOffSpeaker	主持人令成员停止发言。
sendOffAllSpeakers	主持人令全体停止发言。
exitSpeechState	成员停止发言，转变为观众。

屏幕分享接口

API	描述
startScreenCapture	启动屏幕分享。
stopScreenCapture	停止屏幕采集。

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

相关设置接口

API	描述
setVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
getSDKVersion	获取 SDK 版本。

TUIRoomCoreDelegate API 概览

错误事件回调

API	描述
onError	错误回调。

基础事件回调

API	描述
onDestroyRoom	房间解散回调。
onUserVoiceVolume	音量大小回调回调。
onRoomMasterChanged	主持人更改回调。

远端用户事件回调

API	描述
onRemoteUserEnter	远端用户进入房间回调。
onRemoteUserLeave	远端用户离开房间回调。
onRemoteUserCameraAvailable	远端用户是否开启摄像头视频回调。
onRemoteUserScreenVideoAvailable	远端用户是否开启屏幕分享回调。
onRemoteUserAudioAvailable	远端用户是否开启音频上行回调。
onRemoteUserEnterSpeechState	远端用户开始发言回调。

API	描述
onRemoteUserExitSpeechState	远端用户结束发言回调。

消息事件回调

API	描述
onReceiveChatMessage	收到文本消息回调。

场控事件回调

API	描述
onReceiveSpeechInvitation	用户收到主持人发言邀请回调。
onReceiveInvitationCancelled	用户收到主持人取消发言邀请回调。
onReceiveSpeechApplication	主持人收到用户发言申请的回调。
onSpeechApplicationCancelled	用户取消申请发言回调。
onSpeechApplicationForbidden	主持人禁止申请发言回调。
onOrderedToExitSpeechState	成员被请求停止发言的回调。
onCallingRollStarted	主持人开始点名，成员收到的回调。
onCallingRollStopped	主持人结束点名，成员收到的回调。
onMemberReplyCallingRoll	成员回复点名，主持人收到的回调。
onChatRoomMuted	主持人更改聊天室是否禁言回调。
onMicrophoneMuted	主持人设置禁用麦克风回调。
onCameraMuted	主持人设置禁用摄像头回调。
onReceiveKickedOff	成员收到主持人踢人的回调。

统计和质量回调

API	描述
onStatistics	技术指标统计回调。

API	描述
onNetworkQuality	网络质量回调。

屏幕分享相关回调

API	描述
onScreenCaptureStarted	开始屏幕分享回调。
onScreenCaptureStopped	停止屏幕分享回调。

TUIRoomCore 基础函数

getInstance

获取 TUIRoomCore 单例对象。

```
+ (instancetype)shareInstance;
```

destroyInstance

```
+ (void)destroyInstance;
```

setDelegate

TUIRoomCore 事件回调，您可以通过 TUIRoomCoreDelegate 获得 TUIRoomCore 的各种状态通知。

```
- (void)setDelegate:(id)delegate;
```

参数如下表所示：

参数	类型	含义
delegate	TUIRoomCoreDelegate	接收事件回调类。

createRoom

创建房间（主持人调用）。

```
- (void)createRoom:(NSString *)roomId
speechMode:(TUIRoomSpeechMode)speechMode
```

```
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
roomId	NSString	房间标识，需要由您分配并进行统一管理。
speechMode	TUIRoomSpeechMode	发言模式。
callback	TUIRoomActionCallback	创建房间的结果回调。

主持人正常调用流程如下：

1. **主持人**调用 `createRoom()` 创建房间，房间创建成功与否会通过 TUIRoomActionCallback 通知给主持人。
2. **主持人**调用 `startCameraPreview()` 打开摄像头采集和预览。
3. **主持人**调用 `startLocalAudio()` 打开本地麦克风。

destroyRoom

销毁房间房间（主持人调用）。主持人在创建房间后，可以调用该函数来销毁房间。

```
- (void)destroyRoom:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	销毁房间的结果回调。

enterRoom

进入房间（加入房间成员调用）。

```
- (void)enterRoom:(NSString *)roomId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
roomId	NSString	房间标识。
callback	TUIRoomActionCallback	结果回调。

加入房间成员进入房间的正常调用流程如下：

1. **进入房间成员**调用 `enterRoom` 并传入 roomId 即可进入房间房间。
2. **进入房间成员**调用 `startCameraPreview()` 打开摄像头预览，调用 `startLocalAudio()` 打开麦克风采集。
3. **进入房间成员**收到 `onRemoteUserCameraAvailable` 的事件，调用 `startRemoteView()` 开始播放视频。

leaveRoom

离开房间（进入房间成员调用）。

```
- (void)leaveRoom:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

getRoomInfo

获取房间信息。

```
- (nullable TUIRoomInfo *)getRoomInfo;
```

getRoomUsers

获取房间所有成员信息。

```
- (nullable NSArray *)getRoomUsers;
```

getUserInfo

获取房间成员信息。

```
- (void)getUserInfo:(NSString *)userId
callback:(TUIRoomUserInfoCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户标识。
callback	TUIRoomUserInfoCallback	房间人员详细信息回调。

setSelfProfile

设置用户信息。

```
- (void)setSelfProfile:(NSString *)userName
avatarURL:(NSString *)avatarURL
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userName	NSString	用户姓名。
avatarURL	NSString	用户头像 URL。
callback	TUIRoomActionCallback	是否设置成功的结果回调。

transferRoomMaster

将群转交给其他用户。

```
- (void)transferRoomMaster:(NSString *)userId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户标识。
callback	TUIRoomActionCallback	结果回调。

本地推流接口

startCameraPreview

开始本地摄像头预览。

```
- (void)startCameraPreview:(BOOL)isFront
view:(UIView *)view;
```

参数如下表所示：

参数	类型	含义
isFront	BOOL	YES：前置摄像头，NO：后置摄像头。
view	UIView	承载视频画面的控件。

stopCameraPreview

停止本地摄像头预览。

```
- (void)stopCameraPreview;
```

startLocalAudio

开启麦克风采集。

```
- (void)startLocalAudio:(TRTCAudioQuality)quality;
```

参数如下表所示：

参数	类型	含义
quality	TRTCAudioQuality	采集的声音音质。

stopLocalAudio

停止麦克风采集

```
- (void)stopLocalAudio;
```

setVideoMirror

设置本地画面镜像预览模式。

```
- (void)setVideoMirror:(TRTCVideoMirrorType)type;
```

参数如下表所示：

参数	类型	含义
type	TRTCVideoMirrorType	镜像类型。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)isUseSpeaker;
```

参数如下表所示：

参数	类型	含义
isUseSpeaker	BOOL	YES：扬声器，NO：听筒。

远端用户相关接口

startRemoteView

订阅远端用户的视频流。

```
- (void)startRemoteView:(NSString *)userId
view:(UIView *)view
streamType:(TUIRoomStreamType)streamType
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	需要播放的用户 ID。
view	UIView	承载视频画面的 view 控件。
streamType	TUIRoomStreamType	流类型。
callback	TUIRoomActionCallback	结果回调。

stopRemoteView

取消订阅并停止播放远端视频画面。

```
- (void)stopRemoteView:(NSString *)userId
streamType:(TUIRoomStreamType)streamType
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userId	NSString	需要停止播放的用户 ID。
streamType	TUIRoomStreamType	流类型。
callback	TUIRoomActionCallback	结果回调。

switchCamera

切换前后摄像头。

```
- (void)switchCamera:(BOOL)isFront;
```

参数如下表所示：

参数	类型	含义
isFront	BOOL	YES：前置摄像头；NO：后置摄像头。

发送消息接口

sendChatMessage

在房间中广播文本消息，一般用于文本聊天。

```
- (void)sendChatMessage:(NSString *)message
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
message	NSString	消息内容。
callback	TUIRoomActionCallback	发送结果回调。

场控相关接口

muteUserMicrophone

禁用/恢复某用户的麦克风。

```
- (void)muteUserMicrophone:(NSString *)userId
mute:(BOOL)mute
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
mute	BOOL	是否禁用。
callback	TUIRoomActionCallback	结果回调。

muteAllUsersMicrophone

禁用/恢复所有用户的麦克风。

```
- (void)muteAllUsersMicrophone:(BOOL)mute
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
mute	BOOL	是否禁用。
callback	TUIRoomActionCallback	结果回调。

muteUserCamera

禁用/恢复某用户的摄像头。

```
- (void)muteUserCamera:(NSString *)userId
mute:(BOOL)mute
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
mute	BOOL	是否禁用。

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

muteAllUsersCamera

禁用/恢复所有用户的摄像头。

```
- (void)muteAllUsersCamera:(BOOL)mute
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
mute	BOOL	是否禁用。
callback	TUIRoomActionCallback	结果回调。

muteChatRoom

禁言/恢复文字聊天。

```
- (void)muteChatRoom:(BOOL)mute
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
mute	BOOL	是否禁用。
callback	TUIRoomActionCallback	结果回调。

kickOffUser

主持人踢人。

```
- (void)kickOffUser:(NSString *)userId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userId	NSString	用户 ID。
callback	TUIRoomActionCallback	结果回调。

startCallingRoll

主持人开始点名。

```
- (void)startCallingRoll:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

stopCallingRoll

主持人结束点名。

```
- (void)stopCallingRoll:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

replyCallingRoll

成员回复主持人点名。

```
- (void)replyCallingRoll:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

sendSpeechInvitation

主持人邀请成员发言。

```
- (void)sendSpeechInvitation:(NSString *)userId
callback:(TUIRoomInviteeCallback)callback
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
callback	TUIRoomInviteeCallback	结果回调。

cancelSpeechInvitation

主持人取消邀请成员发言。

```
- (void)cancelSpeechInvitation:(NSString *)userId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
callback	TUIRoomActionCallback	结果回调。

replySpeechInvitation

成员同意/拒绝主持人的发言邀请。

```
- (void)replySpeechInvitation:(BOOL)agree
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
agree	BOOL	是否同意。
callback	TUIRoomActionCallback	结果回调。

sendSpeechApplication

成员申请发言。

```
- (void)sendSpeechApplication:(TUIRoomInviteeCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomInviteeCallback	结果回调。

cancelSpeechApplication

成员取消申请发言。

```
- (void)cancelSpeechApplication:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

replySpeechApplication

主持人同意/拒绝成员的申请发言。

```
- (void)replySpeechApplication:(BOOL)agree
userId:(NSString *)userId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
agree	BOOL	是否同意
userId	NSString	用户 ID。
callback	TUIRoomActionCallback	结果回调。

forbidSpeechApplication

主持人禁止申请发言。

```
- (void)forbidSpeechApplication:(BOOL)forbid
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义
forbid	BOOL	是否禁止。
callback	TUIRoomActionCallback	结果回调。

sendOffSpeaker

主持人令成员停止发言。

```
- (void)sendOffSpeaker:(NSString *)userId
callback:(TUIRoomInviteeCallback)callback;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
callback	TUIRoomInviteeCallback	结果回调。

sendOffAllSpeakers

主持人令所有成员停止发言。

```
- (void)sendOffAllSpeakers:(TUIRoomInviteeCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomInviteeCallback	结果回调。

exitSpeechState

成员停止发言，转变为观众。

```
- (void)exitSpeechState:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
callback	TUIRoomActionCallback	结果回调。

屏幕分享接口

startScreenCapture

启动屏幕分享。

```
- (void)startScreenCapture:(TRTCVideoEncParam *)encParam API_AVAILABLE(ios(11.0));
```

参数如下表所示：

参数	类型	含义
encParams	TRTCVideoEncParam	设置屏幕分享时的编码参数。

> 说明：
> 详情请参见 TRTC SDK

stopScreenCapture

停止屏幕采集。

```
- (void)stopScreenCapture API_AVAILABLE(ios(11.0));
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
- (TXBeautyManager *)getBeautyManager;
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。

- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

相关设置接口

setVideoQosPreference

设置网络流控相关参数。

```
- (void)setVideoQosPreference:(TRTCNetworkQosParam *)preference;
```

参数如下表所示：

参数	类型	含义
preference	TRTCNetworkQosParam	网络流控策略。

setAudioQuality

设置音质

```
- (void)setAudioQuality:(TRTCAudioQuality)quality;
```

参数如下表所示：

参数	类型	含义
quality	TRTCAudioQuality	音频质量，详情请参见 TRTC SDK

setVideoResolution

设置分辨率。

```
- (void)setVideoResolution:(TRTCVideoResolution)resolution;
```

参数如下表所示：

参数	类型	含义
resolution	TRTCVideoResolution	视频分辨率，详细请参见 TRTC SDK。

setVideoFps

设置帧率。

```
- (void)setVideoFps:(int)fps;
```

参数如下表所示：

参数	类型	含义
fps	int	视频采集帧率。

> 说明：
> **推荐取值**：15fps或20fps，5fps以下，卡顿感明显。10fps以下，会有轻微卡顿感。20fps以上，则过于浪费（电影的帧率为24fps）。

setVideoBitrate

设置码率。

```
- (void)setVideoBitrate:(int)bitrate;
```

参数如下表所示：

参数	类型	含义
bitrate	int	码率，SDK 会按照目标码率进行编码，只有在网络不佳的情况下才会主动降低视频码率。详情请参见 TRTC SDK。

> 说明：
> **推荐取值**：请参考 TRTCVideoResolution 在各档位注释的最佳码率，也可以在此基础上适当调高。 例如 TRTC_VIDEO_RESOLUTION_1280_720 对应1200kbps的目标码率，您也可以设置为1500kbps以便获得更好的清晰度观感。

enableAudioEvaluation

启用音量大小提示。

```
- (void)enableAudioEvaluation:(BOOL)enable;
```

参数如下表所示：

参数	类型	含义
enable	BOOL	YES：打开，NO：关闭。

说明：
开启后会在 onUserVolumeUpdate 中获取到 SDK 对音量大小值的评估。

setAudioPlayVolume

设置播放音量。

```
- (void)setAudioPlayVolume:(NSInteger)volume;
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0-100， 默认100。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)volume;
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0-100， 默认100。

startFileDumping

开始录音。

```
- (void)startFileDumping:(TRTCAudioRecordingParams *)params;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
params	TRTCAudioRecordingParams	录音参数，详情请参见 TRTC SDK

说明：
该方法调用后， SDK 会将通话过程中的所有音频（包括本地音频，远端音频，BGM 等）录制到一个文件里。无论是否进房，调用该接口都生效。如果调用 leaveRoom 时还在录音，录音会自动停止。

stopFileDumping

停止录音。

```
- (void)stopFileDumping;
```

获取 SDK 版本接口

getSdkVersion

获取 SDK 版本信息。

```
- (NSInteger)getSdkVersion;
```

错误事件回调

onError

```
- (void)onError:(NSInteger)code message:(NSString *)message;
```

参数如下表所示：

参数	类型	含义
code	NSInteger	错误码。
message	NSString	错误信息。

基础事件回调

onDestroyRoom

房间解散回调。

```
- (void)onDestroyRoom;
```

onUserVoiceVolume

用户音量大小回调。

```
- (void)onUserVoiceVolume:(NSString *)userId volume:(NSInteger)volume;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
volume	NSInteger	用户的音量大小，取值范围 0 - 100。

onRoomMasterChanged

主持人更改回调。

```
- (void)onRoomMasterChanged:(NSString *)previousUserId
currentUserId:(NSString *)currentUserId;
```

参数如下表所示：

参数	类型	含义
previousUserId	NSString	更改前的主持人用户 ID。
currentUserId	NSString	更改后的主持人用户 ID。

远端用户回调事件

onRemoteUserEnter

远端用户进入房间回调。

```
- (void)onRemoteUserEnter:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserLeave

远端用户离开房间回调。

```
- (void)onRemoteUserLeave:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserCameraAvailable

远端用户是否开启摄像头视频。

```
- (void)onRemoteUserCameraAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	YES：有视频流数据；NO：无视频流数据。

onRemoteUserScreenVideoAvailable

成员**开启/关闭**视频分享的通知。

```
- (void)onRemoteUserScreenVideoAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	是否有屏幕分享流数据。

onRemoteUserAudioAvailable

远端用户是否开启音频上行回调。

```
- (void)onRemoteUserAudioAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	是否有音频数据。

onRemoteUserEnterSpeechState

远端用户开始发言。

```
- (void)onRemoteUserEnterSpeechState:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserExitSpeechState

远端用户结束发言。

```
- (void)onRemoteUserExitSpeechState:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

聊天室消息事件回调

onReceiveChatMessage

收到文本消息。

```
- (void)onReceiveChatMessage:(NSString *)userId message:(NSString *)message;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
message	NSString	文本消息。

场控消息回调

onReceiveSpeechInvitation

用户收到主持人发言邀请回调。

```
- (void)onReceiveSpeechInvitation:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

onReceiveInvitationCancelled

用户收到主持人取消发言邀请回调。

```
- (void)onReceiveInvitationCancelled:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

OnReceiveSpeechApplication

主持人收到用户发言申请的回调。

```
void onReceiveSpeechApplication(String userId);
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onSpeechApplicationCancelled

用户取消申请发言回调。

```
- (void)onSpeechApplicationCancelled:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onSpeechApplicationForbidden

主持人禁止申请发言回调。

```
- (void)onSpeechApplicationForbidden:(BOOL)isForbidden userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
isForbidden	BOOL	是否禁止。
userId	NSString	用户 ID。

onOrderedToExitSpeechState

成员被请求停止发言的回调。

```
- (void)onOrderedToExitSpeechState:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户ID。

onCallingRollStarted

主持人开始点名，成员收到的回调。

```
- (void)onCallingRollStarted:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

onCallingRollStopped

主持人结束点名，成员收到的回调。

```
- (void)onCallingRollStopped:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

onMemberReplyCallingRoll

成员回复点名，主持人收到的回调。

```
- (void)onMemberReplyCallingRoll:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onChatRoomMuted

主持人更改聊天室是否禁言回调。

```
- (void)onChatRoomMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onMicrophoneMuted

主持人设置禁用麦克风回调。

```
- (void)onMicrophoneMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onCameraMuted

主持人设置禁用摄像头回调。

```
- (void)onCameraMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onReceiveKickedOff

主持人踢人的回调。

```
- (void)onReceiveKickedOff:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人/管理员 用户 ID。

统计和质量回调

onStatistics

技术指标统计回调。

```
- (void)onStatistics:(TRTCStatistics *)statistics;
```

参数如下表所示：

参数	类型	含义
statis	TRTCStatistics	统计数据。

onNetworkQuality

网络状况回调。

```
- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray *)remoteQuality;
```

参数如下表所示：

参数	类型	含义
localQuality	TRTCQualityInfo	上行网络质量。
remoteQuality	NSArray<TRTCQualityInfo *>	下行网络质量。

> 说明：
> 详情请参见 TRTC SDK。

屏幕分享事件回调

onScreenCaptureStarted

开始屏幕分享回调。

```
- (void)onScreenCaptureStarted;
```

onScreenCaptureStopped

停止屏幕分享回调。

```
- (void)onScreenCaptureStopped:(NSInteger)reason;
```

参数如下表所示：

参数	类型	含义
reason	NSInteger	停止原因，0：用户主动停止；1：被其他应用抢占导致停止。

TUIRoom(Android)

最近更新时间：2022-03-02 14:49:09

1. TUIRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主持人创建房间，进入房间人员输入房间号后进入房间。
- 进入房间人员之间进行屏幕分享。
- 支持发送各种文本消息和自定义消息。

2. TUIRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 多人音视频房间(Android)。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频房间组件。
- IM SDK：使用 IM SDK 实现聊天室的功能（**IM SDK 使用 Android 版本**）。

TUIRoom API 概览

TUIRoomCore 基础函数

API	描述
getInstance	获取单例对象。
destroyInstance	销毁单例对象。
setListener	设置事件回调。

房间相关接口函数

API	描述
createRoom	创建房间（主持人调用）。
destroyRoom	销毁房间（主持人调用）。
enterRoom	进入房间（进入房间成员调用）。
leaveRoom	离开房间（其他房间成员调用）。
getRoomInfo	获取房间信息。

API	描述
getRoomUsers	获取房间内所有成员信息。
getUserInfo	获取某个用户的信息。
transferRoomMaster	转移主持人权限（主持人调用）。

本地音视频操作接口

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startLocalAudio	开启麦克风采集。
stopLocalAudio	停止麦克风采集。
setVideoMirror	设置本地画面镜像预览模式。
setSpeaker	设置开启扬声器。

远端用户相关接口

API	描述
startRemoteView	订阅并播放指定成员的远端视频画面。
stopRemoteView	取消订阅并停止播放远端视频画面。

发送聊天消息接口

API	描述
sendChatMessage	发送聊天消息。
sendCustomMessage	发送自定义消息。

场控相关接口

API	描述
muteUserMicrophone	禁用/恢复某用户的麦克风。

API	描述
muteAllUsersMicrophone	禁用/恢复所有用户的麦克风，并且状态会同步到房间信息中。
muteUserCamera	禁用/恢复某用户的摄像头。
muteAllUsersCamera	禁用/恢复所有用户的摄像头，并且状态会同步到房间信息中。
muteChatRoom	开启/停止聊天室禁言（主持人调用）。
kickOffUser	移除房间内的某人（主持人调用）。
startCallingRoll	主持人开始点名。
stopCallingRoll	主持人结束点名。
replyCallingRoll	成员回复主持人点名。
sendSpeechInvitation	主持人邀请成员发言。
cancelSpeechInvitation	主持人取消邀请成员发言。
replySpeechInvitation	成员同意/拒绝主持人的申请发言。
sendSpeechApplication	成员申请发言。
replySpeechApplication	主持人同意/拒绝成员的申请发言。
forbidSpeechApplication	主持人禁止申请发言。
sendOffSpeaker	主持人令成员停止发言。
sendOffAllSpeakers	主持人令全体停止发言。
exitSpeechState	成员停止发言，转变为观众。

屏幕分享接口

API	描述
startScreenCapture	启动屏幕分享。
stopScreenCapture	停止屏幕采集。

美颜滤镜相关接口函数

API	描述

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

相关设置接口

API	描述
setVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
getSDKVersion	获取 SDK 版本。

TUIRoomCoreListener API 概览

错误事件回调

API	描述
onError	错误回调。

基础事件回调

API	描述
onDestroyRoom	房间解散回调。
onUserVoiceVolume	音量大小回调回调。
onRoomMasterChanged	主持人更改回调。

远端用户事件回调

API	描述
onRemoteUserEnter	远端用户进入房间回调。
onRemoteUserLeave	远端用户离开房间回调。

API	描述
onRemoteUserCameraAvailable	远端用户是否开启摄像头视频回调。
onRemoteUserScreenVideoAvailable	远端用户是否开启屏幕分享回调。
onRemoteUserAudioAvailable	远端用户是否开启音频上行回调。
onRemoteUserEnterSpeechState	远端用户开始发言回调。
onRemoteUserExitSpeechState	远端用户结束发言回调。

消息事件回调

API	描述
onReceiveChatMessage	收到文本消息回调。
onReceiveRoomCustomMsg	收到自定义消息回调。

场控事件回调

API	描述
onReceiveSpeechInvitation	用户收到主持人发言邀请回调。
onReceiveInvitationCancelled	用户收到主持人取消发言邀请回调。
onReceiveSpeechApplication	主持人收到用户发言申请的回调。
onSpeechApplicationCancelled	用户取消申请发言回调。
onSpeechApplicationForbidden	主持人禁止申请发言回调。
onOrderedToExitSpeechState	成员被请求停止发言的回调。
onCallingRollStarted	主持人开始点名，成员收到的回调。
onCallingRollStopped	主持人结束点名，成员收到的回调。
onMemberReplyCallingRoll	成员回复点名，主持人收到的回调。
onChatRoomMuted	主持人更改聊天室是否禁言回调。
onMicrophoneMuted	主持人设置禁用麦克风回调。
onCameraMuted	主持人设置禁用摄像头回调。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onReceiveKickedOff	成员收到主持人踢人的回调。

统计和质量回调

API	描述
onStatistics	技术指标统计回调。
onNetworkQuality	网络质量回调。

屏幕分享相关回调

API	描述
onScreenCaptureStarted	开始屏幕分享回调。
onScreenCaptureStopped	停止屏幕分享回调。

TUIRoomCore 基础函数

getInstance

获取 TUIRoomCore 单例对象。

```
public static TUIRoomCore getInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用。

destroyInstance

```
void destroyInstance();
```

setListener

TUIRoomCore 事件回调，您可以通过 TUIRoomCoreListener 获得 TUIRoomCore 的各种状态通知。

```
void setListener(TUIRoomCoreListener listener);
```

参数如下表所示：

参数	类型	含义
listener	TUIRoomCoreListener	接收事件回调类。

createRoom

创建房间（主持人调用）。

```
void createRoom(String roomId, TUIRoomCoreDef.SpeechMode speechMode, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间标识，需要由您分配并进行统一管理。
speechMode	TUIRoomCoreDef.SpeechMode	发言模式。
callback	TUIRoomCoreCallback.ActionCallback	创建房间的结果回调。

主持人正常调用流程如下：

1. **主持人**调用 `createRoom()` 创建房间，房间创建成功与否会通过 `TUIRoomCoreCallback.ActionCallback` 通知给主持人。
2. **主持人**调用 `startCameraPreview()` 打开摄像头采集和预览。
3. **主持人**调用 `startLocalAudio()` 打开本地麦克风。

destroyRoom

销毁房间房间（主持人调用），主持人在创建房间后，可以调用该函数来销毁房间。

```
void destroyRoom(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UIRoomCoreCallback.ActionCallback	销毁房间的结果回调。

enterRoom

进入房间（加入房间成员调用）。

```
void enterRoom(String roomId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间标识。
callback	UIRoomCoreCallback.ActionCallback	结果回调。

加入房间成员进入房间的正常调用流程如下：

1. **进入房间成员**调用 `enterRoom` 并传入 roomId 即可进入房间房间。
2. **进入房间成员**调用 `startCameraPreview()` 打开摄像头预览，调用 `startLocalAudio()` 打开麦克风采集。
3. **进入房间成员**收到 `onRemoteUserCameraAvailable` 的事件，调用 `startRemoteView()` 开始播放视频。

leaveRoom

离开房间（进入房间成员调用）。

```
void leaveRoom(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UIRoomCoreCallback.ActionCallback	结果回调。

getRoomInfo

获取房间信息。

```
TUIRoomCoreDef.RoomInfo getRoomInfo();
```

getRoomUsers

获取房间所有成员信息。

```
List getRoomUsers();
```

getUserInfo

获取房间成员信息。

```
void getUserInfo(String userId, TUIRoomCoreCallback.UserInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户标识。
callback	UIRoomCoreCallback.UserInfoCallback	房间人员详细信息回调。

setSelfProfile

设置用户信息。

```
void setSelfProfile(String userName, String avatarURL, TUIRoomCoreCallback.ActionCallback callbac
k);
```

参数如下表所示：

参数	类型	含义
userName	String	用户姓名。
avatarURL	String	用户头像 URL。
callback	TUIRoomCoreCallback.ActionCallback	是否设置成功的结果回调。

transferRoomMaster

将群转交给其他用户。

```
void transferRoomMaster(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户标识。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

版权所有：腾讯云计算（北京）有限责任公司

本地推流接口

startCameraPreview

开始本地摄像头预览。

```
void startCameraPreview(boolean isFront, TXCloudVideoView view);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。
view	TXCloudVideoView	承载视频画面的控件。

stopCameraPreview

停止本地摄像头预览。

```
void stopCameraPreview();
```

startLocalAudio

开启麦克风采集。

```
void startLocalAudio(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	采集的声音音质： • TRTC_AUDIO_QUALITY_MUSIC • TRTC_AUDIO_QUALITY_DEFAULT • TRTC_AUDIO_QUALITY_SPEECH

stopLocalAudio

停止麦克风采集

```
void stopLocalAudio();
```

版权所有：腾讯云计算（北京）有限责任公司

setVideoMirror

设置本地画面镜像预览模式。

```
void setVideoMirror(int type);
```

参数如下表所示：

参数	类型	含义
type	int	镜像类型。

setSpeaker

设置开启扬声器。

```
void setSpeaker(boolean isUseSpeaker);
```

参数如下表所示：

参数	类型	含义
isUseSpeaker	boolean	true：扬声器，false：听筒。

远端用户相关接口

startRemoteView

订阅远端用户的视频流。

```
void startRemoteView(String userId, TXCloudVideoView view, TUIRoomCoreDef.SteamType streamType, T
UIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要播放的用户 ID。
view	TXCloudVideoView	承载视频画面的 view 控件。
streamType	TUIRoomCoreDef.SteamType	流类型。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

版权所有：腾讯云计算（北京）有限责任公司

stopRemoteView

取消订阅并停止播放远端视频画面。

```
void stopRemoteView(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要停止播放的用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

switchCamera

切换前后摄像头。

```
void switchCamera(boolean isFront);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。

发送消息接口

sendChatMessage

在房间中广播文本消息，一般用于文本聊天。

```
void sendChatMessage(String message, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	消息内容。
callback	TUIRoomCoreCallback.ActionCallback	发送结果回调。

sendCustomMessage

发送自定义消息。

```
void sendCustomMessage(String data, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
data	String	消息内容。
callback	TUIRoomCoreCallback.ActionCallback	发送结果回调。

场控相关接口

muteUserMicrophone

禁用/恢复某用户的麦克风。

```
void muteUserMicrophone(String userId, boolean mute, TUIRoomCoreCallback.ActionCallback callback)
;
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteAllUsersMicrophone

禁用/恢复所有用户的麦克风。

```
void muteAllUsersMicrophone(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteUserCamera

禁用/恢复某用户的摄像头。

```
void muteUserCamera(String userId, boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteAllUsersCamera

禁用/恢复所有用户的摄像头。

```
void muteAllUsersCamera(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteChatRoom

禁言/恢复文字聊天。

```
void muteChatRoom(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

kickOffUser

主持人踢人。

```
void kickOffUser(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

startCallingRoll

主持人开始点名。

```
void startCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

stopCallingRoll

主持人结束点名。

```
void stopCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

replyCallingRoll

成员回复主持人点名。

```
void replyCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

sendSpeechInvitation

主持人邀请成员发言。

```
void sendSpeechInvitation(String userId, TUIRoomCoreCallback.InvitationCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	TUIRoomCoreCallback.InvitationCallback	结果回调。

cancelSpeechInvitation

主持人取消邀请成员发言。

```
void cancelSpeechInvitation(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

replySpeechInvitation

成员同意/拒绝主持人的发言邀请。

```
void replySpeechInvitation(boolean agree, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
agree	boolean	是否同意。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

版权所有：腾讯云计算（北京）有限责任公司

sendSpeechApplication

成员申请发言。

```
void sendSpeechApplication(TUIRoomCoreCallback.InvitationCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.InvitationCallback	结果回调。

cancelSpeechApplication

成员取消申请发言。

```
void cancelSpeechApplication(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

replySpeechApplication

主持人同意/拒绝成员的申请发言。

```
void replySpeechApplication(boolean agree, String userId, TUIRoomCoreCallback.ActionCallback call
back);
```

参数如下表所示：

参数	类型	含义
agree	boolean	是否同意。
userId	String	用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

forbidSpeechApplication

主持人禁止申请发言。

```
void forbidSpeechApplication(boolean forbid, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
forbid	boolean	是否禁止。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

sendOffSpeaker

主持人令成员停止发言。

```
void sendOffSpeaker(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

sendOffAllSpeakers

主持人令所有成员停止发言。

```
void sendOffAllSpeakers(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

exitSpeechState

成员停止发言，转变为观众。

```
void exitSpeechState(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

屏幕分享接口

startScreenCapture

启动屏幕分享。

```
void startScreenCapture(TRTCCloudDef.TRTCVideoEncParam encParams, TRTCCloudDef.TRTCScreenSharePar
ams screenShareParams);
```

参数如下表所示：

参数	类型	含义
encParams	TRTCCloudDef.TRTCVideoEncParam	设置屏幕分享时的编码参数，推荐采用上述推荐配置，如果您指定 encParams 为 null，则使用您调用 startScreenCapture 之前的编码参数设置。
screenShareParams	TRTCCloudDef.TRTCScreenShareParams	设置屏幕分享的特殊配置，其中推荐设置 floatingView，一方面可以避免 App 被系统强杀；另一方面也能助于保护用户隐私。

> 说明：
> 详情请参见 TRTC SDK。

stopScreenCapture

停止屏幕采集。

```
void stopScreenCapture();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
TXBeautyManager getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

相关设置接口

setVideoQosPreference

设置网络流控相关参数。

```
void setVideoQosPreference(TRTCCloudDef.TRTCNetworkQosParam preference);
```

参数如下表所示：

参数	类型	含义
preference	TRTCCloudDef.TRTCNetworkQosParam	网络流控策略。

setAudioQuality

设置音质。

```
void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

setVideoResolution

设置分辨率。

```
void setVideoResolution(int resolution);
```

参数如下表所示：

参数	类型	含义
resolution	int	视频分辨率，详细请参见 TRTC SDK。

setVideoFps

设置帧率。

```
void setVideoFps(int fps);
```

参数如下表所示：

参数	类型	含义
fps	int	视频采集帧率。

> 说明：
> **推荐取值**：15fps或20fps，5fps以下，卡顿感明显。10fps以下，会有轻微卡顿感。20fps以上，则过于浪费（电影的帧率为24fps）。

setVideoBitrate

设置码率。

```
void setVideoBitrate(int bitrate);
```

参数如下表所示：

参数	类型	含义
bitrate	int	码率，SDK 会按照目标码率进行编码，只有在网络不佳的情况下才会主动降低视频码率。详情请参见 TRTC SDK。

> 说明：

推荐取值：请参考 TRTCVideoResolution 在各档位注释的最佳码率，也可以在此基础上适当调高。 例如 TRTC_VIDEO_RESOLUTION_1280_720 对应1200kbps的目标码率，您也可以设置为1500kbps以便获得更好的清晰度观感。

enableAudioEvaluation

启用音量大小提示。

```
void enableAudioEvaluation(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：打开，false：关闭。

说明：
开启后会在 onUserVolumeUpdate 中获取到 SDK 对音量大小值的评估。

setAudioPlayVolume

设置播放音量。

```
void setAudioPlayVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0-100， 默认100。

setAudioCaptureVolume

设置麦克风采集音量

```
void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
volume	int	采集音量，0-100， 默认100。

startFileDumping

开始录音。

```
void startFileDumping(TRTCCloudDef.TRTCAudioRecordingParams trtcAudioRecordingParams);
```

参数如下表所示：

参数	类型	含义
trtcAudioRecordingParams	TRTCCloudDef.TRTCAudioRecordingParams	录音参数，详情请参见 TRTC SDK。

> 说明：
> 该方法调用后， SDK 会将通话过程中的所有音频（包括本地音频，远端音频，BGM 等）录制到一个文件里。无论是否进房，调用该接口都生效。如果调用 leaveRoom 时还在录音，录音会自动停止。

stopFileDumping

停止录音。

```
void stopFileDumping();
```

获取 SDK 版本接口

getSdkVersion

获取 SDK 版本信息。

```
int getSdkVersion();
```

错误事件回调

onError

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

基础事件回调

onDestroyRoom

房间解散回调。

```
void onDestroyRoom();
```

onUserVoiceVolume

用户音量大小回调。

```
void onUserVoiceVolume(String userId, int volume);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
volume	int	用户的音量大小，取值范围 0 - 100。

onRoomMasterChanged

主持人更改回调。

```
void onRoomMasterChanged(String previousUserId, String currentUserId);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
previousUserId	String	更改前的主持人用户 ID。
currentUserId	String	更改后的主持人用户 ID。

远端用户回调事件

onRemoteUserEnter

远端用户进入房间回调。

```
void onRemoteUserEnter(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onRemoteUserLeave

远端用户离开房间回调。

```
void onRemoteUserLeave(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onRemoteUserCameraAvailable

远端用户是否开启摄像头视频。

```
void onRemoteUserCameraAvailable(String userId, boolean available);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

参数	类型	含义
available	boolean	true：有视频流数据；false：无视频流数据。

onRemoteUserScreenVideoAvailable

成员**开启/关闭**视频分享的通知。

```
void onRemoteUserScreenVideoAvailable(String userId, boolean available);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	是否有屏幕分享流数据。

onRemoteUserAudioAvailable

远端用户是否开启音频上行回调。

```
void onRemoteUserAudioAvailable(String userId, boolean available);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	是否有音频数据。

onRemoteUserEnterSpeechState

远端用户开始发言。

```
void onRemoteUserEnterSpeechState(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onRemoteUserExitSpeechState

远端用户结束发言。

```
void onRemoteUserExitSpeechState(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

聊天室消息事件回调

onReceiveChatMessage

收到文本消息。

```
void onReceiveChatMessage(String userId, String message);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
message	String	文本消息。

onReceiveRoomCustomMsg

收到自定义消息。

```
void onReceiveRoomCustomMsg(String userId, String data);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
message	String	自定义消息。

场控消息回调

onReceiveSpeechInvitation

用户收到主持人发言邀请回调。

```
void onReceiveSpeechInvitation(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	主持人用户 ID。

onReceiveInvitationCancelled

用户收到主持人取消发言邀请回调。

```
void onReceiveInvitationCancelled(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	主持人用户 ID。

onReceiveSpeechApplication

主持人收到用户发言申请的回调。

```
void onReceiveSpeechApplication(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onSpeechApplicationCancelled

用户取消申请发言回调。

```
void onSpeechApplicationCancelled(String userId);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userId	String	用户 ID。

onSpeechApplicationForbidden

主持人禁止申请发言回调。

```
void onSpeechApplicationForbidden(boolean isForbidden);
```

参数如下表所示：

参数	类型	含义
isForbidden	boolean	是否禁止。

onOrderedToExitSpeechState

成员被请求停止发言的回调。

```
void onOrderedToExitSpeechState(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	主持人用户 ID。

onCallingRollStarted

主持人开始点名，成员收到的回调。

```
void onCallingRollStarted(String userId);
```

onCallingRollStopped

主持人结束点名，成员收到的回调。

```
void onCallingRollStopped(String userId);
```

onMemberReplyCallingRoll

成员回复点名，主持人收到的回调。

```
void onMemberReplyCallingRoll(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onChatRoomMuted

主持人更改聊天室是否禁言回调。

```
void onChatRoomMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

onMicrophoneMuted

主持人设置禁用麦克风回调。

```
void onMicrophoneMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

onCameraMuted

主持人设置禁用摄像头回调。

```
void onCameraMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

版权所有：腾讯云计算（北京）有限责任公司

onReceiveKickedOff

主持人踢人的回调。

```
void onReceiveKickedOff(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	主持人/管理员 用户 ID。

统计和质量回调

onStatistics

技术指标统计回调。

```
void onStatistics(TRTCStatistics statistics);
```

参数如下表所示：

参数	类型	含义
statis	TRTCStatistics	统计数据。

onNetworkQuality

网络状况回调。

```
void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality, List remoteQuality);
```

参数如下表所示：

参数	类型	含义
localQuality	TRTCCloudDef.TRTCQuality	上行网络质量。
remoteQuality	List<TRTCCloudDef.TRTCQuality>	下行网络质量。

说明：
详情请参见 TRTC SDK

屏幕分享事件回调

onScreenCaptureStarted

开始屏幕分享回调。

```
void onScreenCaptureStarted();
```

onScreenCaptureStopped

停止屏幕分享回调。

```
void onScreenCaptureStopped(int reason);
```

参数如下表所示：

参数	类型	含义
reason	int	停止原因，0：用户主动停止；1：被其他应用抢占导致停止。

版权所有：腾讯云计算（北京）有限责任公司

TRTCMeeting（Flutter）

最近更新时间：2022-03-01 12:18:48

TRTCMeeting 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主持人创建会议房间，参会人员输入房间号后进入会议。
- 参会人员之间进行屏幕分享。
- 支持发送各种文本消息和自定义消息。

TRTCMeeting 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 多人音视频房间（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时视频会议组件。
- IM SDK：使用 IM SDK 的 MeetingRoom 实现会议中聊天室的功能。

TRTCMeeting API 概览

SDK 基础接口

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
registerListener	设置事件监听。
unRegisterListener	销毁事件监听。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

会议房间相关接口

API	描述
createMeeting	创建会议房间（主持人调用）。
destroyMeeting	销毁会议房间（主持人调用）。

API	描述
enterMeeting	进入会议房间（参会成员调用）。
leaveMeeting	离开会议房间（参会成员调用）。
getUserInfoList	获取房间内所有的人员列表，enterMeeting()成功后调用才有效。
getUserInfo	获取房间内指定人员的详细信息，enterMeeting()成功后调用才有效。

远端用户相关接口

API	描述
startRemoteView	播放指定成员的远端视频画面。
stopRemoteView	停止播放指定成员的远端视频画面。
setRemoteViewParam	设置指定成员的远端图像渲染参数。
muteRemoteAudio	静音/取消静音指定成员的远端音频。
muteAllRemoteAudio	静音/取消静音所有成员的远端音频。
muteRemoteVideoStream	暂停/恢复指定成员的远端视频。
muteAllRemoteVideoStream	暂停/恢复所有成员的远端视频流。

本地视频操作接口

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
switchCamera	切换前后摄像头。
setVideoEncoderParam	设置视频编码器相关参数。
setLocalViewMirror	设置本地画面镜像预览模式。

本地音频操作接口

API	描述

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器或听筒。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
startAudioRecording	开始录音。
stopAudioRecording	停止录音。
enableAudioVolumeEvaluation	启用音量大小提示。

录屏接口

API	描述
startScreenCapture	启动屏幕分享。
stopScreenCapture	停止屏幕采集。
pauseScreenCapture	暂停屏幕采集。
resumeScreenCapture	恢复屏幕采集。

获取相关管理对象接口

API	描述
getDeviceManager	获取设备管理对象 TXDeviceManager。
getBeautyManager	获取美颜管理对象 TXBeautyManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在会议中广播文本消息，一般用于聊天。

API	描述
sendRoomCustomMsg	发送自定义文本消息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onKickedOffline	其他用户登录了同一账号，被踢下线。

会议房间事件回调

API	描述
onRoomDestroy	会议房间被销毁的回调。
onNetworkQuality	网络状态回调。
onUserVolumeUpdate	用户通话音量回调。

成员进出事件回调

API	描述
onEnterRoom	本地进会回调。
onLeaveRoom	本地退会回调。
onUserEnterRoom	新成员进会回调。
onUserLeaveRoom	成员退会回调。

成员音视频事件回调

API	描述
onUserAudioAvailable	成员开启/关闭麦克风的回调。

API	描述
onUserVideoAvailable	成员开启/关闭摄像头的回调。
onUserSubStreamAvailable	成员开启/关闭辅路画面的回调。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息的回调。
onRecvRoomCustomMsg	收到自定义消息的回调。

录屏事件回调

API	描述
onScreenCaptureStarted	录屏开始回调。
onScreenCapturePaused	录屏暂停回调。
onScreenCaptureResumed	录屏恢复回调。
onScreenCaptureStoped	录屏停止回调。

SDK 基础接口

sharedInstance

获取 TRTCMeeting 单例对象。

```
static Future sharedInstance();
```

destroySharedInstance

销毁 TRTCMeeting 单例对象。

```
static void destroySharedInstance();
```

说明：

销毁实例后，外部缓存的 TRTCMeeting 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

registerListener

设置事件监听，您可以通过 TRTCMeetingDelegate 获得 TRTCMeeting 的各种状态通知。

```
void registerListener(MeetingListenerFunc func);
```

说明：
func 是 TRTCMeeting 的代理回调。

unRegisterListener

销毁事件监听。

```
void unRegisterListener(MeetingListenerFunc func);
```

login

登录。

```
Future login(int sdkAppId, String userId, String userSig);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 >【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。

logout

登出。

```
Future logout();
```

setSelfProfile

修改个人信息。

```
Future setSelfProfile(String userName, String avatarURL);
```

参数如下表所示：

参数	类型	含义
userName	String	用户昵称。
avatarURL	String	用户头像地址。

会议房间相关接口

createMeeting

创建会议（主持人调用）。

```
Future createMeeting(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	会议房间标识，需要由您分配并进行统一管理。

主持人正常调用流程如下：

1. 【主持人】调用 `createMeeting()` 并传入 `roomId` 创建会议，会议房间创建成功与否会通过 `ActionCallback` 通知给主持人。
2. 【主持人】调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
3. 【主持人】调用 `startMicrophone()` 打开麦克风采集。

destroyMeeting

销毁会议房间（主持人调用）。主持人在创建会议后，可以调用该函数来销毁会议。

```
Future destroyMeeting(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	会议房间标识，需要由您分配并进行统一管理。

enterMeeting

进入会议房间（参会成员调用）。

```
Future enterMeeting(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	会议房间标识。

参会成员进入会议的正常调用流程如下：

1. 【参会成员】调用 `enterMeeting()` 并传入 `roomId` 即可进入会议房间。
2. 【参会成员】调用 `startCameraPreview()` 打开摄像头预览，调用 `startMicrophone()` 打开麦克风采集。
3. 【参会成员】收到 `onUserVideoAvailable` 的事件，调用 `startRemoteView()` 并传入成员的 `userId` 开始播放。

leaveMeeting

离开会议房间（参会成员调用）。

```
Future leaveMeeting();
```

getUserInfoList

获取房间内所有的人员列表，enterMeeting()成功后调用才有效。

```
Future getUserInfoList(List<String> userIdList);
```

参数如下表所示：

参数	类型	含义
userIdList	List	需要获取的 userId 列表。

getUserInfo

获取房间内指定人员的详细信息，enterMeeting()成功后调用才有效。

```
Future getUserInfo(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。

远端用户相关接口

startRemoteView

播放指定成员的远端视频画面。

```
Future<void> startRemoteView(String userId, int streamType, int viewId);
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。
streamType	int	要观看的视频流类型。详情请参见 TRTC SDK。
viewId	int	TRTCCloudVideoView 生成的 viewId。

stopRemoteView

停止播放指定成员的远端视频画面。

```
Future<void> stopRemoteView(String userId, int streamType);
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。
streamType	int	要观看的视频流类型。详情请参见 TRTC SDK。

setRemoteViewParam

设置指定成员的远端图像渲染参数。

```
Future<void> setRemoteViewParam(String userId, int streamType,
{int fillMode, int rotation, int mirrorType});
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。
streamType	int	要观看的视频流类型。详情请参见 TRTC SDK。
fillMode	int	图像渲染模式，填充（默认值）或适应模式。详情请参见 TRTC SDK。
rotation	int	图像顺时针旋转角度。详情请参见 TRTC SDK。
mirrorType	int	镜像模式。详情请参见 TRTC SDK。

muteRemoteAudio

静音/取消静音指定成员的远端音频。

```
Future<void> muteRemoteAudio(String userId, bool mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。
mute	boolean	true：静音；false：关闭静音。

muteAllRemoteAudio

静音/取消静音所有成员的远端音频。

```
Future<void> muteAllRemoteAudio(bool mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：静音；false：关闭静音。

版权所有：腾讯云计算（北京）有限责任公司

muteRemoteVideoStream

暂停/恢复指定成员的远端视频。

```
Future<void> muteRemoteVideoStream(String userId, bool mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定成员的 ID。
mute	boolean	true：暂停；false：恢复。

muteAllRemoteVideoStream

暂停/恢复所有成员的远端视频流。

```
Future<void> muteAllRemoteVideoStream(bool mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：暂停；false：恢复。

本地视频操作接口

startCameraPreview

开启本地视频的预览画面。

```
Future<void> startCameraPreview(bool isFront, int viewId);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。
viewId	int	TRTCCloudVideoView 生成的 viewId。

stopCameraPreview

停止本地视频采集及预览。

```
Future<void> stopCameraPreview();
```

switchCamera

切换前后摄像头。

```
Future<void> switchCamera(bool isFront);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。

setVideoEncoderParam

设置视频编码器相关参数。

```
Future<void> setVideoEncoderParam({
int videoFps,
int videoBitrate,
int videoResolution,
int videoResolutionMode,
});
```

参数如下表所示：

参数	类型	含义
videoFps	int	视频采集帧率。
videoBitrate	int	码率，SDK 会按照目标码率进行编码，只有在网络不佳的情况下才会主动降低视频码率。
videoResolution	int	视频分辨率。
videoResolutionMode	int	分辨率模式。

说明：
详情请参见 TRTC SDK。

setLocalViewMirror

设置本地画面镜像预览模式。

```
Future<void> setLocalViewMirror(bool isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	boolean	是否开启镜像预览模式，true：开启；false：不开启。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
Future<void> startMicrophone({int quality});
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量。详情请参见 TRTC SDK。

stopMicrophone

停止麦克风采集。

```
Future<void> stopMicrophone();
```

muteLocalAudio

开启/关闭本地静音。

```
Future<void> muteLocalAudio(bool mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：静音；false：取消静音。

setSpeaker

设置开启扬声器或听筒。

```
Future<void> setSpeaker(bool useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
Future<void> setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，取值0 - 100，默认值为100。

setAudioPlayoutVolume

设置播放音量。

```
Future<void> setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，取值0 - 100，默认值100。

startAudioRecording

开始录音。

```
Future<int?> startAudioRecording(String filePath);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
filePath	String	录音文件的保存路径，该路径需要用户自行指定，请确保路径存在且可写。该路径需精确到文件名及格式后缀，格式后缀决定录音文件的格式，目前支持的格式有 PCM、WAV 和 AAC。

stopAudioRecording

停止录音。

```
Future<void> stopAudioRecording();
```

enableAudioVolumeEvaluation

启用音量大小提示。

```
Future<void> enableAudioVolumeEvaluation(int intervalMs);
```

参数如下表所示：

参数	类型	含义
intervalMs	int	决定了 onUserVoiceVolume 回调的触发间隔，单位为ms，最小间隔为100ms，如果小于等于0则会关闭回调，建议设置为300ms。

录屏接口

startScreenCapture

启动屏幕分享。

```
Future<void> startScreenCapture({
int videoFps,
int videoBitrate,
int videoResolution,
int videoResolutionMode,
String appGroup,
});
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
videoFps	int	视频采集帧率。
videoBitrate	int	码率，SDK 会按照目标码率进行编码，只有在网络不佳的情况下才会主动降低视频码率。
videoResolution	int	视频分辨率。
videoResolutionMode	int	分辨率模式。
appGroup	String	该参数仅仅在iOS端有效，Android端不需要关注这个参数。该参数是主 App 与 Broadcast 共享的 Application Group Identifier。

说明：
详情请参见 TRTC SDK。

stopScreenCapture

停止屏幕采集。

```
Future<void> stopScreenCapture();
```

pauseScreenCapture

暂停屏幕采集。

```
Future<void> pauseScreenCapture();
```

resumeScreenCapture

恢复屏幕采集。

```
Future<void> resumeScreenCapture();
```

获取相关管理对象接口

getDeviceManager

获取设备管理对象 TXDeviceManager。

```
getDeviceManager();
```

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口

sendRoomTextMsg

在会议中广播文本消息，一般用于聊天。

```
Future sendRoomTextMsg(String message);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。

sendRoomCustomMsg

发送自定义文本消息。

```
Future sendRoomCustomMsg(String cmd, String message);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。

参数	类型	含义
message	String	文本消息。

TRTCMeetingDelegate事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

参数如下表所示：

参数	类型	含义
errCode	int	错误码。
errMsg	String	错误信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onWarning

警告回调。

参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onKickedOffline

其他用户登录了同一账号，被踢下线。

会议房间事件回调

onRoomDestroy

会议房间被销毁的回调。主持人退房时，房间内的所有用户都会收到此通知。

参数如下表所示：

参数	类型	含义
roomId	String	会议房间 ID。

onNetworkQuality

网络状态回调。

参数如下表所示：

参数	类型	含义
localQuality	TRTCCloudDef.TRTCQuality	上行网络质量。
remoteQuality	List<TRTCCloudDef.TRTCQuality>	下行网络质量。

onUserVolumeUpdate

用户通话音量回调。

参数如下表所示：

参数	类型	含义
userVolumes	List	所有正在说话的房间成员的音量，取值范围0 - 100。
totalVolume	int	所有远端成员的总音量，取值范围0 - 100。

成员进出事件回调

onEnterRoom

本地进会回调。

参数如下表所示：

参数	类型	含义
result	int	大于0时为进会耗时（ms），小于0时为进会错误码。

onLeaveRoom

本地退会回调。

参数如下表所示：

参数	类型	含义
reason	int	离开会议原因，0：主动调用 leaveMeeting 退会；1：被服务器踢出当前会议；2：当前会议整个被解散。

onUserEnterRoom

新成员进会回调。

参数如下表所示：

参数	类型	含义
userId	String	新进会成员的用户 ID。

onUserLeaveRoom

成员退会回调。

参数如下表所示：

参数	类型	含义
userId	String	退会成员的用户 ID。

成员音视频事件回调

onUserAudioAvailable

成员开启/关闭麦克风的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开麦克风；false：用户关闭麦克风。

onUserVideoAvailable

成员开启/关闭摄像头的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开摄像头；false：用户关闭摄像头。

onUserSubStreamAvailable

成员开启/关闭辅路画面的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开辅路画面；false：用户关闭辅路画面。

消息事件回调

onRecvRoomTextMsg

收到文本消息的回调。

参数如下表所示：

参数	类型	含义
message	String	文本消息。
sendId	String	发送者用户 ID。
userAvatar	String	发送者用户头像。

参数	类型	含义
userName	String	发送者用户昵称。

onRecvRoomCustomMsg

收到自定义消息的回调。

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
sendId	String	发送者用户 ID。
userAvatar	String	发送者用户头像。
userName	String	发送者用户昵称。

录屏事件回调

onScreenCaptureStarted

录屏开始回调。

onScreenCapturePaused

录屏暂停回调。

onScreenCaptureResumed

录屏恢复回调。

onScreenCaptureStoped

录屏停止回调。

参数如下表所示：

参数	类型	含义
reason	int	停止原因，0：用户主动停止；1：屏幕窗口关闭导致停止。

版权所有：腾讯云计算（北京）有限责任公司

TUIRoom(Windows&Mac)

最近更新时间：2022-03-02 14:38:10

TUIRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主持人创建房间，参会人员输入房间号后进入房间。
- 参会人员之间进行屏幕分享。
- 支持发送各种文本消息和自定义消息。

TUIRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 多人音视频房间(Windows&Mac)。

- TRTC SDK：使用 TRTC SDK 作为低延时视频会议组件。
- IM SDK：使用 IM SDK 实现聊天室的功能（**IM SDK 使用 C++ 版本**）。

TUIRoom API 概览

TUIRoomCore 基础函数

API	描述
GetInstance	获取单例对象。
DestroyInstance	销毁单例对象。
SetCallback	设置事件回调。

房间相关接口函数

API	描述
login	登录。
logout	登出。
CreateRoom	创建房间（主持人调用）。
DestroyRoom	销毁房间（主持人调用）。
EnterRoom	进入房间（参会成员调用）。
LeaveRoom	离开房间（参会成员或主持人调用）。

API	描述
GetRoomInfo	获取房间信息。
GetRoomUsers	获取房间内所有成员信息。
GetUserInfo	获取某个用户的信息。
TransferRoomMaster	转移主持人权限（主持人调用）。

本地音视频操作接口

API	描述
StartCameraPreview	开启本地视频的预览画面。
StopCameraPreview	停止本地视频采集及预览。
UpdateCameraPreview	改变本地视频渲染窗口。
StartLocalAudio	开启麦克风采集。
StopLocalAudio	停止麦克风采集。
StartSystemAudioLoopback	开启/关闭系统声音的采集。
StopSystemAudioLoopback	开启/关闭系统声音的采集。
SetVideoMirror	设置本地画面镜像预览模式。

远端用户相关接口

API	描述
StartRemoteView	订阅并播放指定成员的远端视频画面。
StopRemoteView	取消订阅并停止播放远端视频画面。
UpdateRemoteView	改变远端用户的视频渲染窗口。

发送聊天消息接口

API	描述
SendChatMessage	发送聊天消息。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
SendCustomMessage	发送自定义消息。

场控相关接口

API	描述
MuteUserMicrophone	禁用/恢复某用户的麦克风。
MuteAllUsersMicrophone	禁用/恢复所有用户的麦克风，并且状态会同步到房间信息中。
MuteUserCamera	禁用/恢复某用户的摄像头。
MuteAllUsersCamera	禁用/恢复所有用户的摄像头，并且状态会同步到房间信息中。
MuteChatRoom	开启/停止聊天室禁言（主持人调用）。
KickOffUser	移除房间内的某人（主持人调用）。
StartCallingRoll	主持人开始点名。
StopCallingRoll	主持人结束点名。
ReplyCallingRoll	成员回复主持人点名。
SendSpeechInvitation	主持人邀请成员发言。
CancelSpeechInvitation	主持人取消邀请成员发言。
ReplySpeechInvitation	成员同意/拒绝主持人的申请发言。
SendSpeechApplication	成员申请发言。
CancelSpeechApplication	成员取消申请发言。
ReplySpeechApplication	主持人同意/拒绝成员的申请发言。
ForbidSpeechApplication	主持人禁止申请发言。
SendOffSpeaker	主持人令成员停止发言。
SendOffAllSpeakers	主持人令全体停止发言。
ExitSpeechState	成员停止发言，转变为观众。

基础组件接口函数

API	描述
GetDeviceManager	获取本地设置管理对象 ITXDeviceManager。
GetScreenShareManager	获取屏幕分享管理对象 IScreenShareManager。

云录制接口函数

API	描述
StartCloudRecord	开始云录制 。
StopCloudRecord	停止云录制 。

美颜相关接口函数

API	描述
SetBeautyStyle	美颜设置 。

相关设置接口

API	描述
SetVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
GetSDKVersion	获取 SDK 版本。

TUIRoomCoreCallback API 概览

错误事件回调

API	描述
OnError	错误回调。

基础事件回调

API	描述
OnLogin	登录回调。
OnLogout	登出回调。
OnCreateRoom	创建房间回调。
OnDestroyRoom	房间解散回调。
OnEnterRoom	进入房间回调。
OnExitRoom	退出房间回调。
OnFirstVideoFrame	首帧画面回调。
OnUserVoiceVolume	音量大小回调回调。
OnRoomMasterChanged	主持人更改回调。

远端用户事件回调

API	描述
OnRemoteUserEnter	远端用户进入房间回调。
OnRemoteUserLeave	远端用户离开房间回调。
OnRemoteUserCameraAvailable	远端用户是否开启摄像头视频回调。
OnRemoteUserScreenAvailable	远端用户是否开启屏幕分享回调。
OnRemoteUserAudioAvailable	远端用户是否开启麦克风回调。
OnRemoteUserEnterSpeechState	远端用户开始发言回调。
OnRemoteUserExitSpeechState	远端用户结束发言回调。

消息事件回调

API	描述
OnReceiveChatMessage	收到文本消息回调。
OnReceiveCustomMessage	收到文本消息回调。

场控事件回调

API	描述
OnReceiveSpeechInvitation	用户收到主持人发言邀请回调。
OnReceiveInvitationCancelled	用户收到主持人取消发言邀请回调。
OnReceiveReplyToSpeechInvitation	主持人收到用户同意邀请发言的回调。
OnReceiveSpeechApplication	主持人收到用户发言申请的回调。
OnSpeechApplicationCancelled	用户取消申请发言回调。
OnReceiveReplyToSpeechApplication	主持人同意发言申请回调。
OnSpeechApplicationForbidden	主持人禁止申请发言回调。
OnOrderedToExitSpeechState	成员被请求停止发言的回调。
OnCallingRollStarted	主持人开始点名，成员收到的回调。
OnCallingRollStopped	主持人结束点名，成员收到的回调。
OnMemberReplyCallingRoll	成员回复点名，主持人收到的回调。
OnChatRoomMuted	主持人更改聊天室是否禁言回调。
OnMicrophoneMuted	主持人设置禁用麦克风回调。
OnCameraMuted	主持人设置禁用摄像头回调。

统计和质量回调

API	描述
OnStatistics	技术指标统计回调。
OnNetworkQuality	网络质量回调。

屏幕分享相关回调

API	描述
OnScreenCaptureStarted	开始屏幕分享回调。
OnScreenCaptureStopped	停止屏幕分享回调。

视频录制回调

API	描述
OnRecordError	录制错误回调。
OnRecordComplete	录制完成回调。
OnRecordProgress	录制进度回调。

本地设备测试回调

API	描述
OnTestSpeakerVolume	扬声器大小回调。
OnTestMicrophoneVolume	麦克风大小回调。
OnAudioDeviceCaptureVolumeChanged	调节系统采集音量回调。
OnAudioDevicePlayoutVolumeChanged	调节系统播放音量回调。

TUIRoomCore 基础函数

GetInstance

获取 TUIRoomCore 单例对象。

```
static TUIRoomCore* GetInstance();
```

DestroyInstance

```
static void DestroyInstance();
```

SetCallback

TUIRoomCore 事件回调，您可以通过 TUIRoomCoreCallback 获得 TUIRoomCore 的各种状态通知。

```
virtual void SetCallback(const TUIRoomCoreCallback* callback) = 0;
```

Login

登录。

```
virtual int Login(int sdk_appid, const std::string& user_id, const std::string& user_sig) = 0;
```

参数如下表所示：

参数	类型	含义
sdk_appid	int	您可以在实时音视频控制台 > 应用管理> 应用信息中查看 SDKAppID。
user_id	string	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
user_sig	string	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。

Logout

登出。

```
virtual int Logout() = 0;
```

CreateRoom

创建房间（主持人调用）。

```
virtual int CreateRoom(const std::string& room_id, TUISpeechMode speech_mode) = 0;
```

参数如下表所示：

参数	类型	含义
room_id	string	房间标识，需要由您分配并进行统一管理。
speech_mode	TUISpeechMode	发言模式。

主持人正常调用流程如下：

1. **主持人**调用 `CreateRoom()` 创建房间，房间创建成功与否会通过 OnCreateRoom 通知给主持人。
2. **主持人**调用 `EnterRoom()` 进入房间。
3. **主持人**调用 `StartCameraPreview()` 打开摄像头采集和预览。
4. **主持人**调用 `StartLocalAudio()` 打开本地麦克风。

DestroyRoom

销毁房间房间（主持人调用）。主持人在创建房间后，可以调用该函数来销毁房间。

```
virtual int DestroyRoom() = 0;
```

EnterRoom

进入房间（参会成员调用）。

```
virtual int EnterRoom(const std::string& room_id) = 0;
```

参数如下表所示：

参数	类型	含义
room_id	string	房间标识。

参会成员进入房间的正常调用流程如下：

1. **参会成员**调用 `EnterRoom` 并传入 room_id 即可进入房间房间。
2. **参会成员**调用 `startCameraPreview()` 打开摄像头预览，调用 `StartLocalAudio()` 打开麦克风采集。
3. **参会成员**收到 `OnRemoteUserCameraAvailable` 的事件，调用 `StartRemoteView()` 开始播放视频。

LeaveRoom

离开房间（参会成员调用）。

```
virtual int LeaveRoom() = 0;
```

GetRoomInfo

获取房间信息。

```
virtual TUIRoomInfo GetRoomInfo() = 0;
```

GetRoomUsers

获取房间所有成员信息。

```
virtual std::vector GetRoomUsers() = 0;
```

GetUserInfo

获取房间成员信息。

```
virtual const TUIUserInfo* GetUserInfo(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户标识。

SetSelfProfile

设置用户属性。

```
virtual int SetSelfProfile(const std::string& user_name, const std::string& avatar_url) = 0;
```

参数如下表所示：

参数	类型	含义
user_name	string	用户姓名。
avatar_url	string	用户头像 URL。

TransferRoomMaster

将群转交给其他用户。

```
virtual int TransferRoomMaster(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户标识。

本地推流接口

StartCameraPreview

开始本地摄像头预览。

```
virtual int StartCameraPreview(const liteav::TXView& view) = 0;
```

参数如下表所示：

参数	类型	含义
view	liteav::TXView	窗口句柄。

StopCameraPreview

停止本地摄像头预览。

```
virtual int StopCameraPreview() = 0;
```

UpdateCameraPreview

更新本地视频预览画面的窗口。

```
virtual int UpdateCameraPreview(const liteav::TXView& view) = 0;
```

参数如下表所示：

参数	类型	含义
view	liteav::TXView	窗口句柄。

StartLocalAudio

打开本地音频设备。

```
virtual int StartLocalAudio(const liteav::TRTCAudioQuality& quality) = 0;
```

参数如下表所示：

参数	类型	含义
view	liteav::TXView	窗口句柄。

StopLocalAudio

关闭本地音频设备。

```
virtual int StopLocalAudio() = 0;
```

StartSystemAudioLoopback

开启系统声音的采集。

```
virtual int StartSystemAudioLoopback() = 0;
```

StopSystemAudioLoopback

关闭系统声音的采集。

```
virtual int StopSystemAudioLoopback() = 0;
```

SetVideoMirror

镜像设置。

```
virtual int SetVideoMirror(bool mirror) = 0;
```

参数如下表所示：

参数	类型	含义
mirror	bool	是否镜像。

远端用户相关接口

StartRemoteView

订阅远端用户的视频流。

```
virtual int StartRemoteView(const std::string& user_id, const liteav::TXView& view,
TUIStreamType type = TUIStreamType::kStreamTypeCamera) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	需要播放的用户 ID。
liteav::TXView	TXView	承载视频画面的 view 控件。
type	TUIStreamType	流类型。

StopRemoteView

取消订阅并停止播放远端视频画面。

```
virtual int StopRemoteView(const std::string& user_id,
TUIStreamType type = TUIStreamType::kStreamTypeCamera) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	需要停止播放的用户 ID。
type	TUIStreamType	流类型。

UpdateRemoteView

更新远端视频渲染窗口。

```
virtual int UpdateRemoteView(const std::string& user_id, TUIStreamType type, liteav::TXView& vie
w) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
type	TUIStreamType	流类型。
view	liteav::TXView	渲染窗口句柄。

发送消息接口

SendChatMessage

发送文本消息。

```
virtual int SendChatMessage(const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
message	string	消息内容。

SendCustomMessage

发送自定义消息。

```
virtual int SendCustomMessage(const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
message	string	消息内容。

场控相关接口

MuteUserMicrophone

禁用/恢复某用户的麦克风。

```
virtual int MuteUserMicrophone(const std::string& user_id, bool mute, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
mute	bool	是否禁用。
callback	Callback	接口回调。

MuteAllUsersMicrophone

禁用/恢复所有用户的麦克风。

```
virtual int MuteAllUsersMicrophone(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

MuteUserCamera

禁用/恢复某用户的摄像头。

```
virtual int MuteUserCamera(const std::string& user_id, bool mute, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
mute	bool	是否禁用。
callback	Callback	接口回调。

MuteAllUsersCamera

禁用/恢复所有用户的摄像头。

```
virtual int MuteAllUsersCamera(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

MuteChatRoom

开启/停止聊天室禁言。

```
virtual int MuteChatRoom(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

KickOffUser

主持人踢人。

```
virtual int KickOffUser(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
callback	Callback	接口回调。

StartCallingRoll

主持人开始点名。

```
virtual int StartCallingRoll() = 0;
```

StopCallingRoll

主持人结束点名。

```
virtual int StopCallingRoll() = 0;
```

ReplyCallingRoll

成员回复主持人点名。

```
virtual int ReplyCallingRoll(Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
callback	Callback	接口回调。

SendSpeechInvitation

主持人邀请成员发言。

```
virtual int SendSpeechInvitation(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
callback	Callback	接口回调。

CancelSpeechInvitation

主持人取消邀请成员发言。

```
virtual int CancelSpeechInvitation(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
callback	Callback	接口回调。

ReplySpeechInvitation

成员同意/拒绝主持人的发言邀请。

```
virtual int ReplySpeechInvitation(bool agree, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
agree	bool	是否同意。
callback	Callback	接口回调。

SendSpeechApplication

成员申请发言。

```
virtual int SendSpeechApplication(Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
callback	Callback	接口回调。

CancelSpeechApplication

成员取消申请发言。

```
virtual int CancelSpeechApplication(Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
callback	Callback	接口回调。

ReplySpeechApplication

主持人同意/拒绝成员的申请发言。

```
virtual int ReplySpeechApplication(const std::string& user_id, bool agree, Callback callback) = 0
;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
callback	Callback	接口回调。

ForbidSpeechApplication

主持人禁止申请发言。

```
virtual int ForbidSpeechApplication(bool forbid) = 0;
```

参数如下表所示：

参数	类型	含义
forbid	bool	是否禁止。

SendOffSpeaker

主持人令成员停止发言。

```
virtual int SendOffSpeaker(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
callback	Callback	接口回调。

SendOffAllSpeakers

主持人令所有成员停止发言。

```
virtual int SendOffAllSpeakers(Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
callback	Callback	接口回调。

ExitSpeechState

成员停止发言，转变为观众。

```
virtual int ExitSpeechState() = 0;
```

基础组件接口

GetDeviceManager

获取设备管理的对象指针。

```
virtual liteav::ITXDeviceManager* GetDeviceManager() = 0;
```

GetScreenShareManager

获取屏幕分享管理的对象指针。

```
virtual IScreenShareManager* GetScreenShareManager() = 0;
```

云录制接口

StartCloudRecord

开始云录制。

```
virtual int StartCloudRecord() = 0;
```

StopCloudRecord

停止云录制。

```
virtual int StopCloudRecord() = 0;
```

美颜相关接口函数

SetBeautyStyle

设置美颜、美白、红润效果级别。

```
virtual int SetBeautyStyle(liteav::TRTCBeautyStyle style, uint32_t beauty_level,
uint32_t whiteness_level, uint32_t ruddiness_level) = 0;
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”，光滑或者自然，光滑风格磨皮更加明显。
- 设置“美颜级别”，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显。
- 设置“美白级别”，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显。

参数如下表所示：

参数	类型	含义
style	liteav::TRTCBeautyStyle	美颜风格。
beauty_level	uint32_t	美颜级别。
whiteness_level	uint32_t	美白级别。
ruddiness_level	uint32_t	红润级别。

相关设置接口

SetVideoQosPreference

设置网络流控相关参数。

```
virtual int SetVideoQosPreference(TUIVideoQosPreference preference) = 0;
```

参数如下表所示：

参数	类型	含义
preference	TUIVideoQosPreference	网络流控策略。

获取 SDK 版本接口

GetSDKVersion

获取 SDK 版本信息。

```
virtual const char* GetSDKVersion() = 0;
```

错误事件回调

OnError

```
void OnError(int code, const std::string& message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	错误信息。

基础事件回调

OnLogin

```
virtual void OnLogin(int code, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	登录信息或登录失败的错误信息。

OnLogout

```
virtual void OnLogout(int code, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	错误信息。

OnCreateRoom

创建房间回调。

```
virtual void OnCreateRoom(int code, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	错误信息。

OnDestroyRoom

房间解散回调。

```
virtual void OnDestroyRoom(int code, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	错误信息。

OnEnterRoom

进入房间回调。

```
virtual void OnEnterRoom(int code, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	string	错误信息。

OnExitRoom

退出房间回调。

```
virtual void OnExitRoom(TUIExitRoomType type, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
type	TUIExitRoomType	退出房间的类型。
message	string	错误信息。

OnFirstVideoFrame

开始渲染自己本地或远端用户的首帧画面。

```
virtual void OnFirstVideoFrame(const std::string& user_id, const TUIStreamType stream_type) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
stream_type	TUIStreamType	流类型。

OnUserVoiceVolume

用户音量大小回调。

```
virtual void OnUserVoiceVolume(const std::string& user_id, int volume)
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
volume	int	用户的音量大小，取值范围 0 - 100。

OnRoomMasterChanged

主持人更改回调。

```
virtual void OnRoomMasterChanged(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

远端用户回调事件

OnRemoteUserEnter

远端用户进入房间回调。

```
virtual void OnRemoteUserEnter(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnRemoteUserLeave

远端用户离开房间回调。

```
virtual void OnRemoteUserLeave(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnRemoteUserCameraAvailable

远端用户是否开启摄像头视频。

```
virtual void OnRemoteUserCameraAvailable(const std::string& user_id, bool available) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
available	bool	true：有视频流数据；false：无视频流数据。

OnRemoteUserScreenAvailable

远端用户是否开启屏幕分享。

```
virtual void OnRemoteUserScreenAvailable(const std::string& user_id, bool available) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
available	bool	true：有视频流数据；false：无视频流数据。

OnRemoteUserAudioAvailable

远端用户是否开启麦克风。

```
virtual void OnRemoteUserAudioAvailable(const std::string& user_id, bool available) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
available	bool	true：有音频流数据；false：无音频流数据。

OnRemoteUserEnterSpeechState

远端用户开始发言。

```
virtual void OnRemoteUserEnterSpeechState(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnRemoteUserExitSpeechState

远端用户结束发言。

```
virtual void OnRemoteUserExitSpeechState(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

聊天室消息事件回调

OnReceiveChatMessage

收到文本消息。

```
virtual void OnReceiveChatMessage(const std::string& user_id, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
message	string	文本消息。

OnReceiveCustomMessage

收到自定义消息。

```
virtual void OnReceiveCustomMessage(const std::string& user_id, const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
message	string	自定义消息。

场控消息回调

OnReceiveSpeechInvitation

用户收到主持人发言邀请回调。

```
virtual void OnReceiveSpeechInvitation() = 0;
```

OnReceiveInvitationCancelled

用户收到主持人取消发言邀请回调。

```
virtual void OnReceiveInvitationCancelled() = 0;
```

OnReceiveReplyToSpeechInvitation

主持人收到用户同意邀请发言的回调。

```
virtual void OnReceiveReplyToSpeechInvitation(const std::string& user_id, bool agree) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
agree	bool	是否同意。

OnReceiveSpeechApplication

主持人收到用户发言申请的回调。

```
virtual void OnReceiveSpeechApplication(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnSpeechApplicationCancelled

用户取消申请发言回调。

```
virtual void OnSpeechApplicationCancelled(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnReceiveReplyToSpeechApplication

主持人同意发言申请回调。

```
virtual void OnReceiveReplyToSpeechApplication(bool agree) = 0;
```

参数如下表所示：

参数	类型	含义
agree	bool	是否同意。

OnSpeechApplicationForbidden

主持人禁止申请发言回调。

```
virtual void OnSpeechApplicationForbidden(bool forbidden) = 0;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
forbidden	bool	是否禁止。

OnOrderedToExitSpeechState

成员被请求停止发言的回调。

```
virtual void OnOrderedToExitSpeechState() = 0;
```

OnCallingRollStarted

主持人开始点名，成员收到的回调。

```
virtual void OnCallingRollStarted() = 0;
```

OnCallingRollStopped

主持人结束点名，成员收到的回调。

```
virtual void OnCallingRollStopped() = 0;
```

OnMemberReplyCallingRoll

成员回复点名，主持人收到的回调。

```
virtual void OnMemberReplyCallingRoll(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnChatRoomMuted

主持人更改聊天室是否禁言回调。

```
virtual void OnChatRoomMuted(bool muted) = 0;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
muted	bool	是否禁用。

OnMicrophoneMuted

主持人设置禁用麦克风回调。

```
virtual void OnMicrophoneMuted(bool muted) = 0;
```

参数如下表所示：

参数	类型	含义
muted	bool	是否禁用。

OnCameraMuted

主持人设置禁用摄像头回调。

```
virtual void OnCameraMuted(bool muted) = 0;
```

参数如下表所示：

参数	类型	含义
muted	bool	是否禁用。

统计和质量回调

OnStatistics

技术指标统计回调。

```
virtual void OnStatistics(const liteav::TRTCStatistics& statis) {}
```

参数如下表所示：

参数	类型	含义
statis	liteav::TRTCStatistics	统计数据。

OnNetworkQuality

网络状况回调。

```
virtual void OnNetworkQuality(const liteav::TRTCQualityInfo& local_quality, liteav::TRTCQualityInfo* remote_quality,
uint32_t remote_quality_count) {}
```

参数如下表所示：

参数	类型	含义
local_quality	liteav::TRTCQualityInfo	本地用户质量信息。
remote_quality	liteav::TRTCQualityInfo*	远端用户质量信息指针。
remote_quality_count	uint32_t	远端用户数量。

录屏事件回调

OnScreenCaptureStarted

开始屏幕分享回调。

```
virtual void OnScreenCaptureStarted() {}
```

OnScreenCaptureStopped

停止屏幕分享回调。

```
void OnScreenCaptureStopped(int reason) {}
```

参数如下表所示：

参数	类型	含义
reason	int	停止原因，0：用户主动停止；1：被其他应用抢占导致停止。

视频录制回调

OnRecordError

录制错误回调。

```
virtual void OnRecordError(TXLiteAVLocalRecordError error, const std::string& messgae) {}
```

参数如下表所示：

参数	类型	含义
error	TXLiteAVLocalRecordError	错误信息。
messgae	string	错误描述。

OnRecordComplete

录制完成回调。

```
virtual void OnRecordComplete(const std::string& path) {}
```

参数如下表所示：

参数	类型	含义
path	string	错误描述。

OnRecordProgress

录制进度回调。

```
virtual void OnRecordProgress(int duration, int file_size) {}
```

参数如下表所示：

参数	类型	含义
duration	int	文件的时长。
file_size	int	文件的大小。

本地设备测试回调

OnTestSpeakerVolume

扬声器音量大小回调。

```
virtual void OnTestSpeakerVolume(uint32_t volume) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。

OnTestMicrophoneVolume

麦克风音量大小回调。

```
virtual void OnTestMicrophoneVolume(uint32_t volume) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。

OnAudioDeviceCaptureVolumeChanged

调节系统采集音量回调。

```
virtual void OnAudioDeviceCaptureVolumeChanged(uint32_t volume, bool muted) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。
muted	bool	是否被禁用。

OnAudioDevicePlayoutVolumeChanged

调节系统播放音量回调。

```
virtual void OnAudioDevicePlayoutVolumeChanged(uint32_t volume, bool muted) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。
muted	bool	是否被禁用。

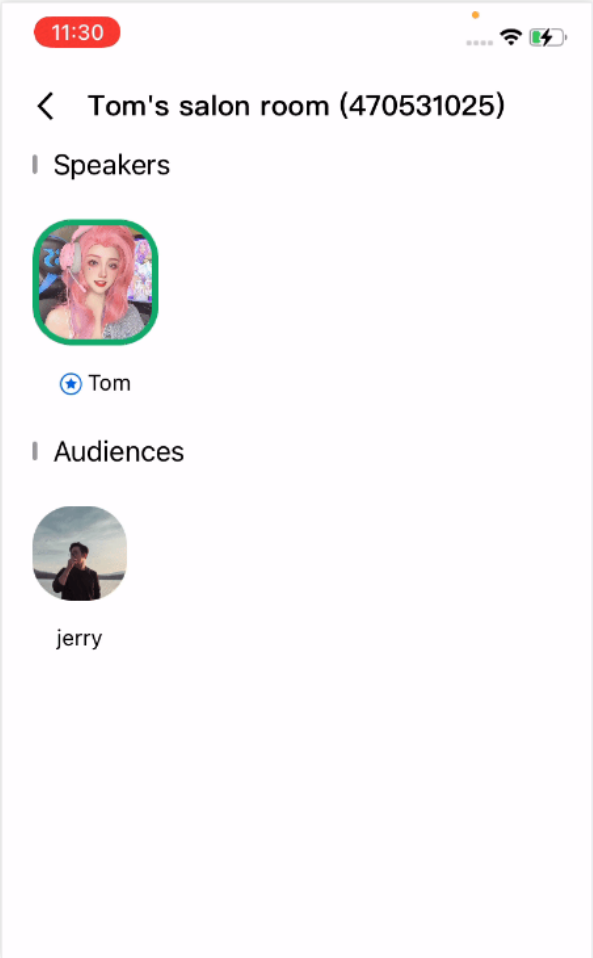
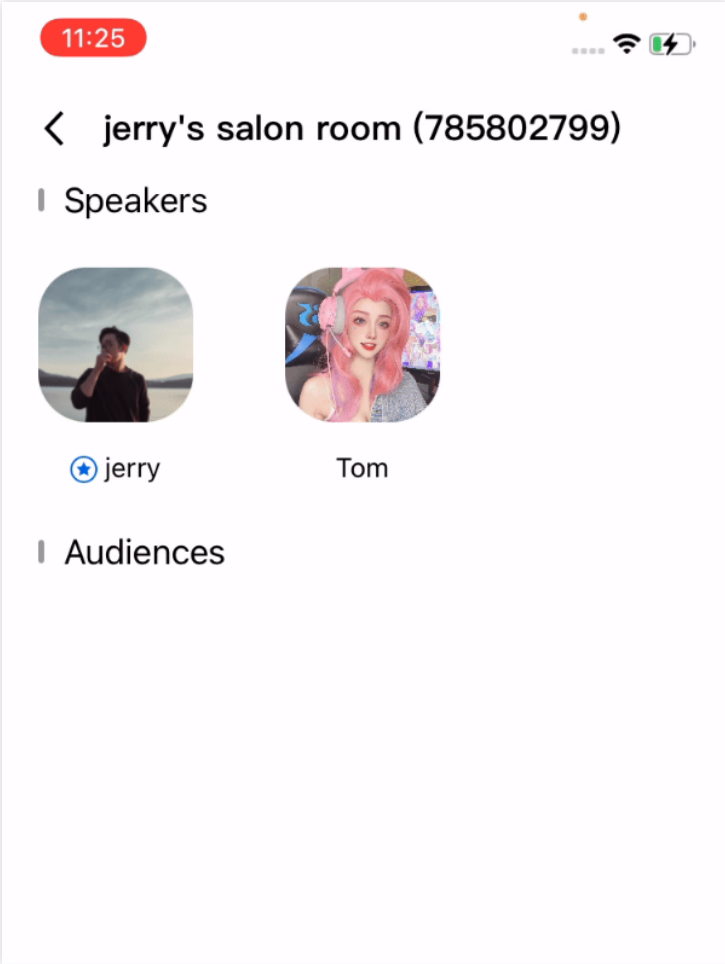
版权所有：腾讯云计算（北京）有限责任公司

语音沙龙
语音沙龙(Android)

最近更新时间：2022-03-09 17:55:16

效果展示

您可以 下载 安装我们的 App 体验语音沙龙的能力，包括语音聊天、上下麦、低延时语音互动等 TRTC 在语音聊天场景下的相关能力。

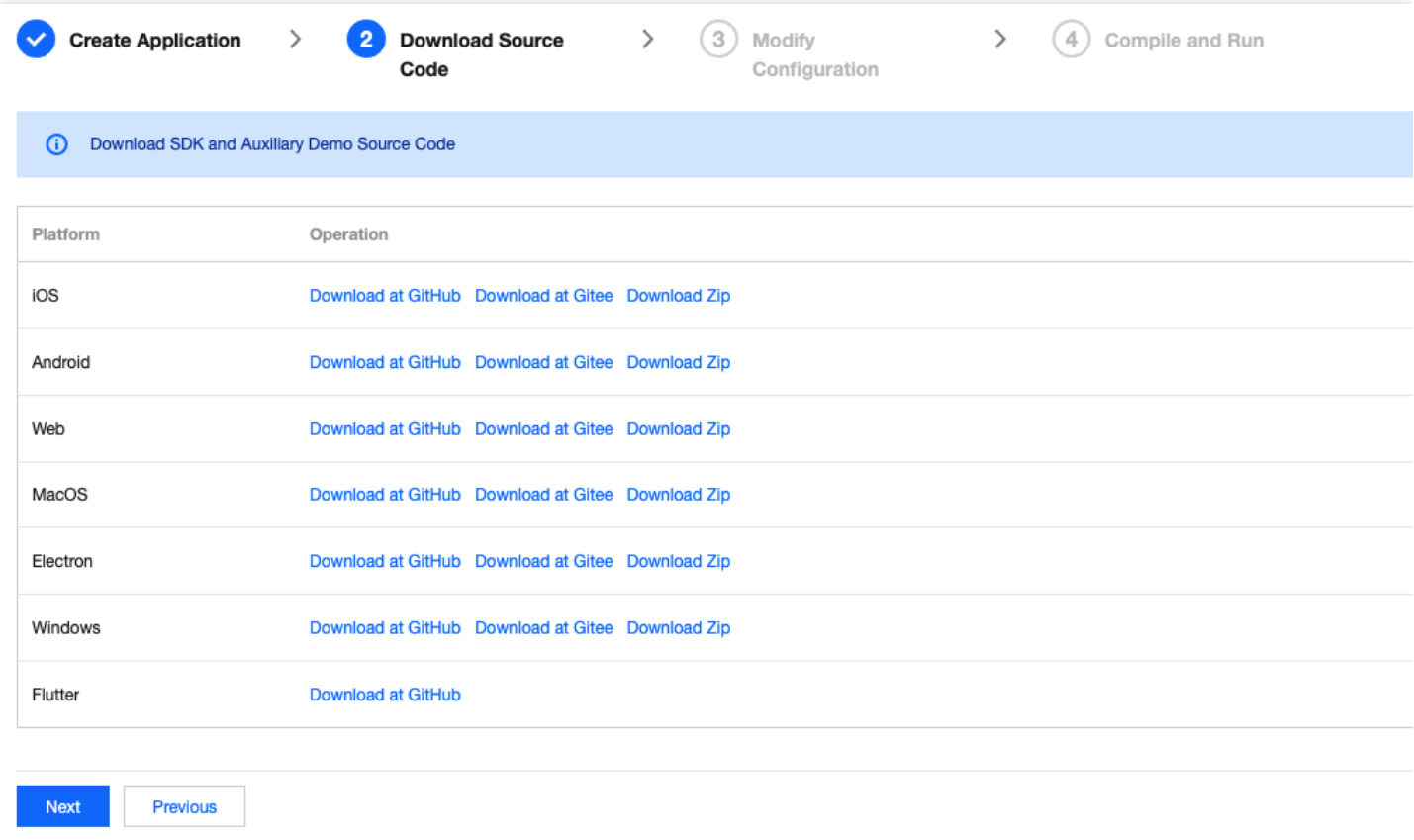
房主麦位操作	听众麦位操作
	

如需快速接入语音沙龙功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TRTCChatSalon 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
2. 输入应用名称，例如 `TestChatSalon`，单击 **创建**。
3. 单击 **已下载，下一步**，跳过此步骤。



> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

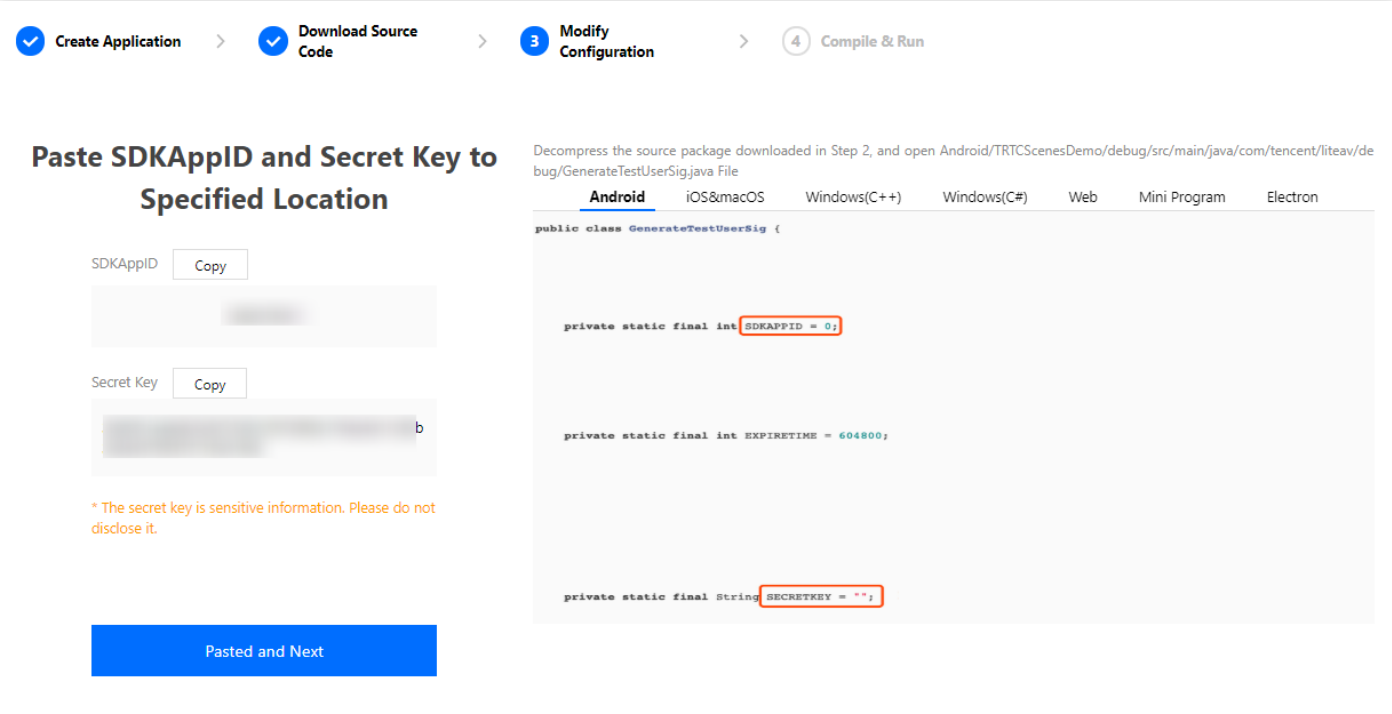
单击进入 TUIChatSalon，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。

3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5以上的版本）打开源码工程 `TUIChatSalon`，单击 **运行** 即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 Source 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

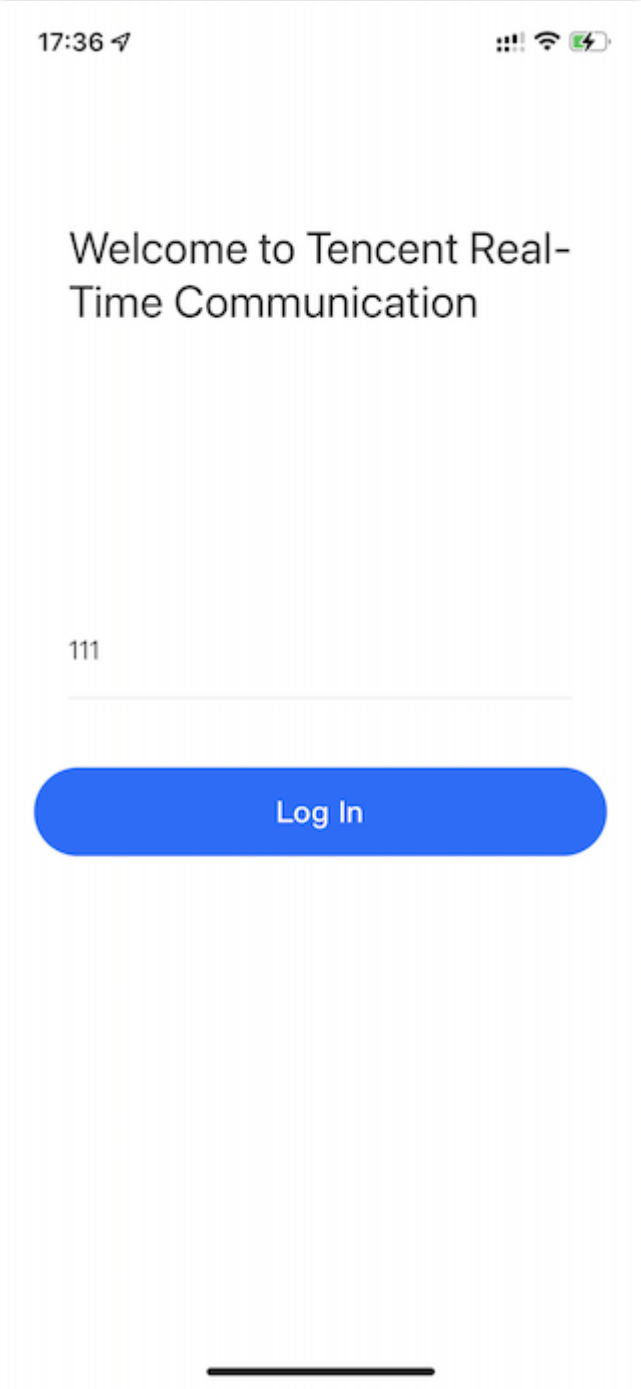
文件或文件夹	功能描述
base	UI 使用的基础类。
list	创建房间页。
room	主房间页面，包括房主和听众两种界面。
widget	通用控件。

体验应用

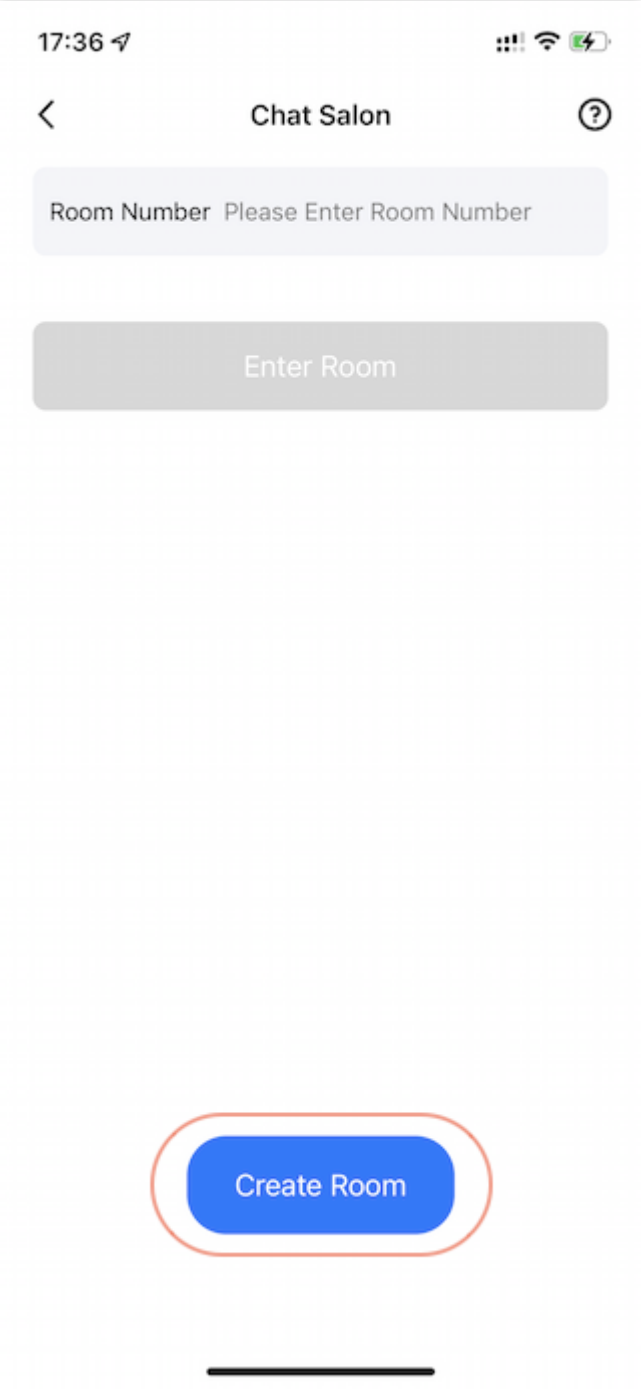
> 注意：
> 体验应用至少需要两台设备。

用户 A

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。



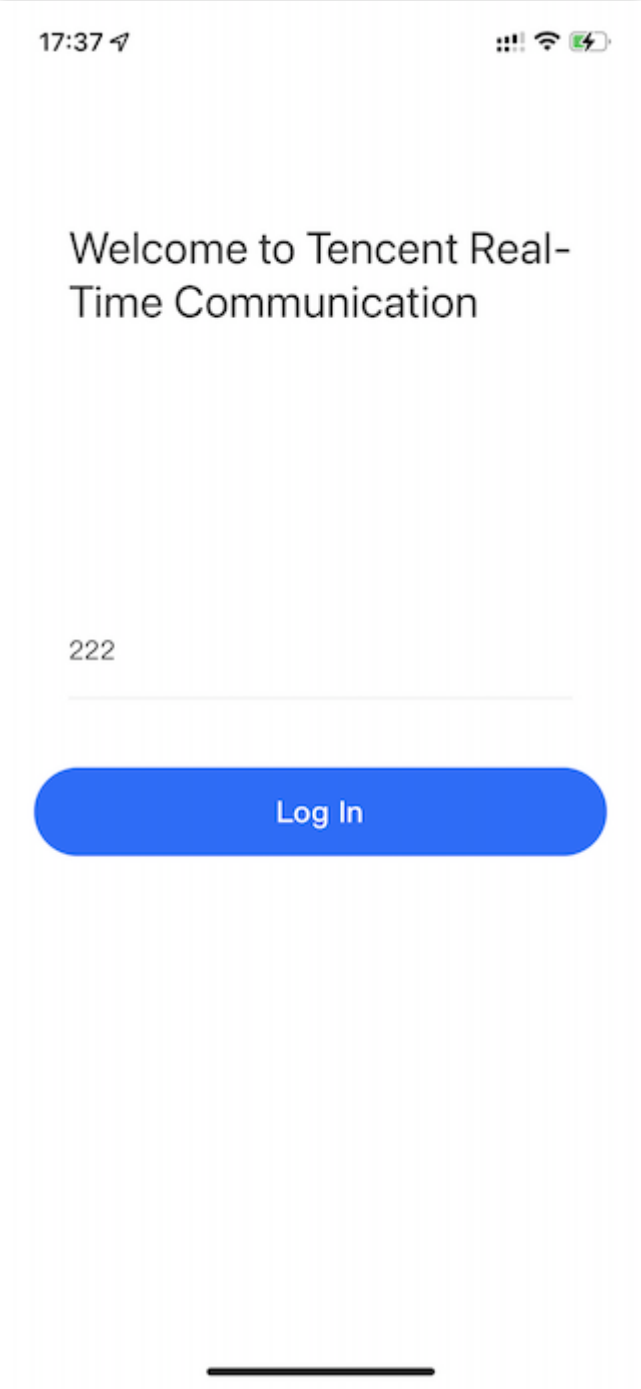
2. 进入后，单击 **创建房间**。



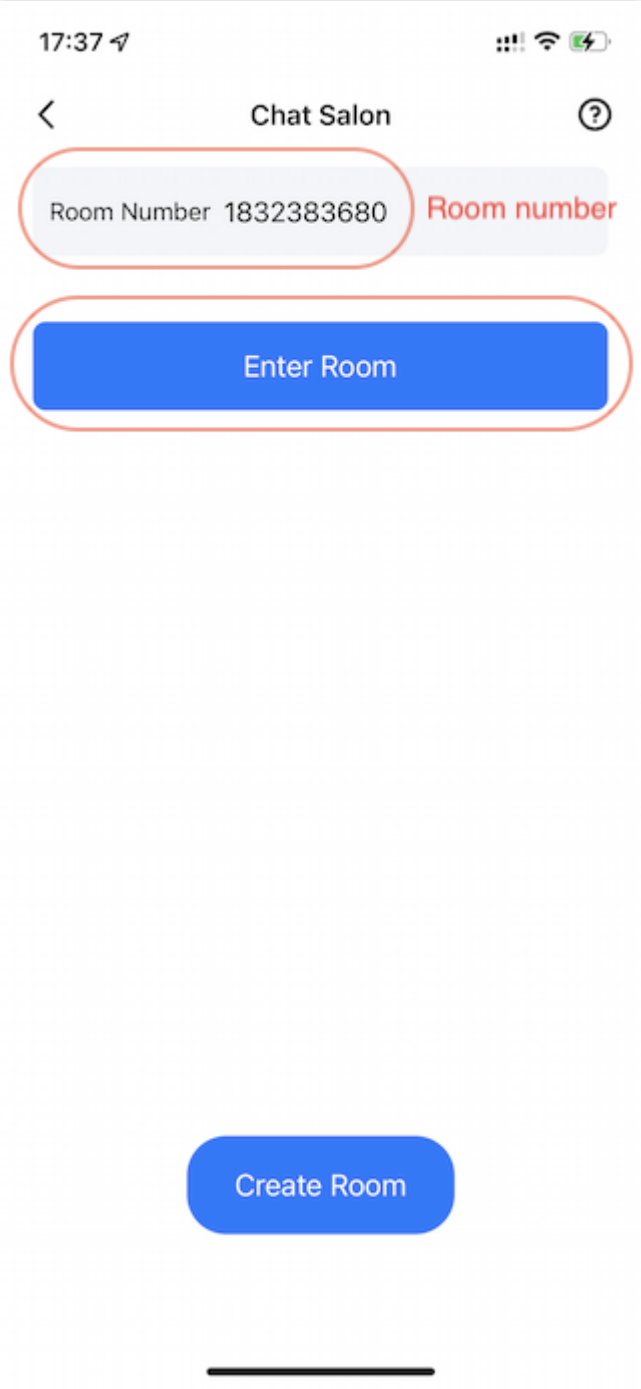
3. 输入房间主题，单击 **开始交谈**。

用户 B

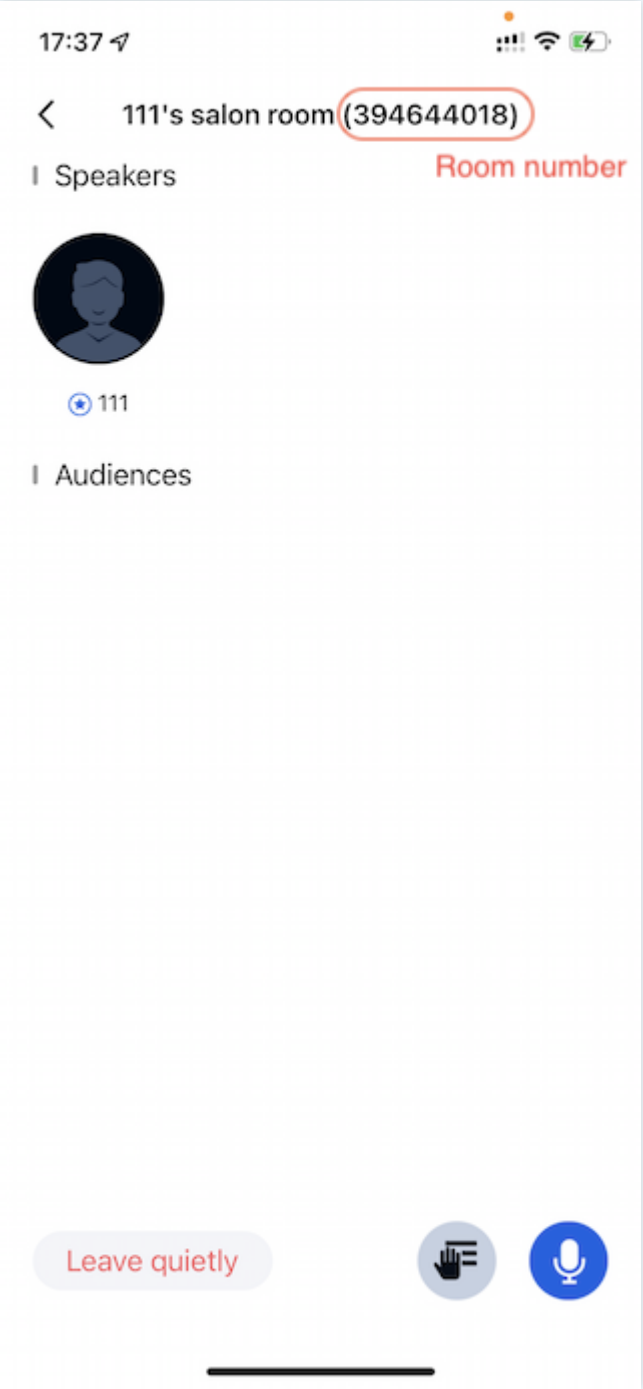
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。



2. 输入用户 A 创建的房间号，单击 **进入房间**。



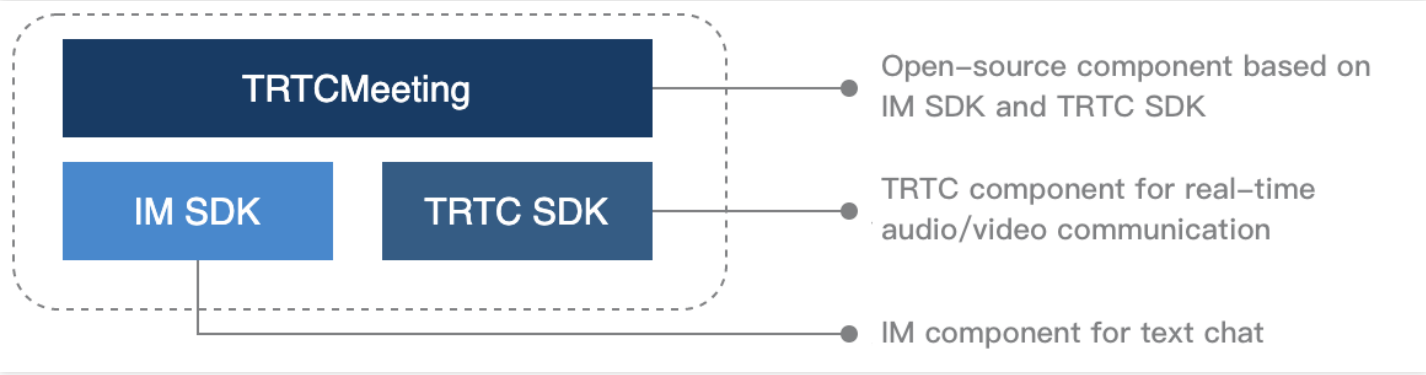
注意：
房间号在用户 A 的房间顶部查看。



实现自定义 UI 界面

源码 中的 Source 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCChatSalon，您可以在 `TRTCChatSalon.java` 文件中看到该组件提供的接口函数，并使用对应接口实现自定

义 UI 界面。



步骤1：集成 SDK

语音沙龙组件 Source 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 Maven 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
complie "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
complie 'com.tencent.imsdk:imsdk:latest.release'
compile 'com.google.code.gson:gson:2.3.1'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击 **Sync Now**，自动下载 SDK 并集成到工程里。

方法二：通过本地 AAR 依赖

若您的开发环境访问 Maven 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档

IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态读取存储权限）：

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

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

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

步骤3：导入 TRTCChatSalon 组件

拷贝以下目录中的所有文件到您的项目中：

```
Source/src/main/java/com/tencent/liteav/trtcchatsalon/model
```

步骤4：创建并登录组件

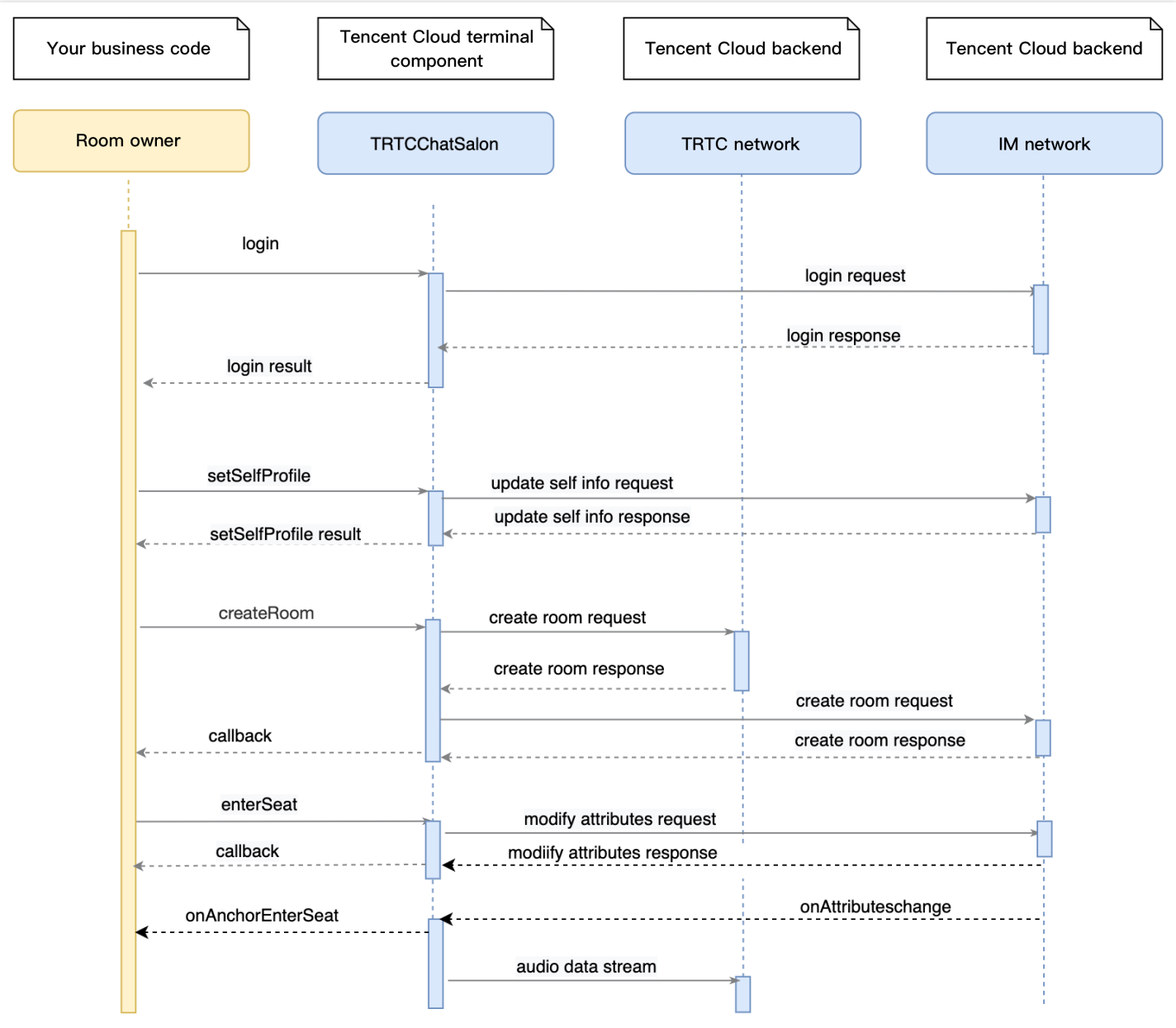
1. 调用 `sharedInstance` 接口可以创建一个 TRTCChatSalon 组件的实例对象。
2. 调用 `setDelegate` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	登录回调，成功时 code 为0。

```
TRTCChatSalon mTRTCChatSalon = TRTCChatSalon.sharedInstance(this);
mTRTCChatSalon.setDelegate(this);
mTRTCChatSalon.login(SDKAPPID, userId, userSig, new TRTCChatSalonCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

步骤5：房主端开播

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的语音沙龙，此时传入房间 ID、上麦是否需要房主确认、房间类型等房间属性信息。
3. 房主会收到有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。




```
// 1.房主设置昵称和头像
mTRTCChatSalon.setSelfProfile("my_name", "my_face_url", null);

// 2.房主调用 createRoom 创建房间
final TRTCChatSalonDef.RoomParam roomParam = new TRTCChatSalonDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = true; // 上麦是否需要房主确认
roomParam.coverUrl = "房间封面图的 URL 地址";
mTRTCChatSalon.createRoom(mRoomId, roomParam, new TRTCChatSalonCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
// 3.占座位
```

```
mTRTCChatSalon.enterSeat(new TRTCChatSalonCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
}
}
});
}
}
});

// 4.占座成功后， 收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCChatSalonDef.UserInfo userInfo) {
}
```

步骤6：听众端观看

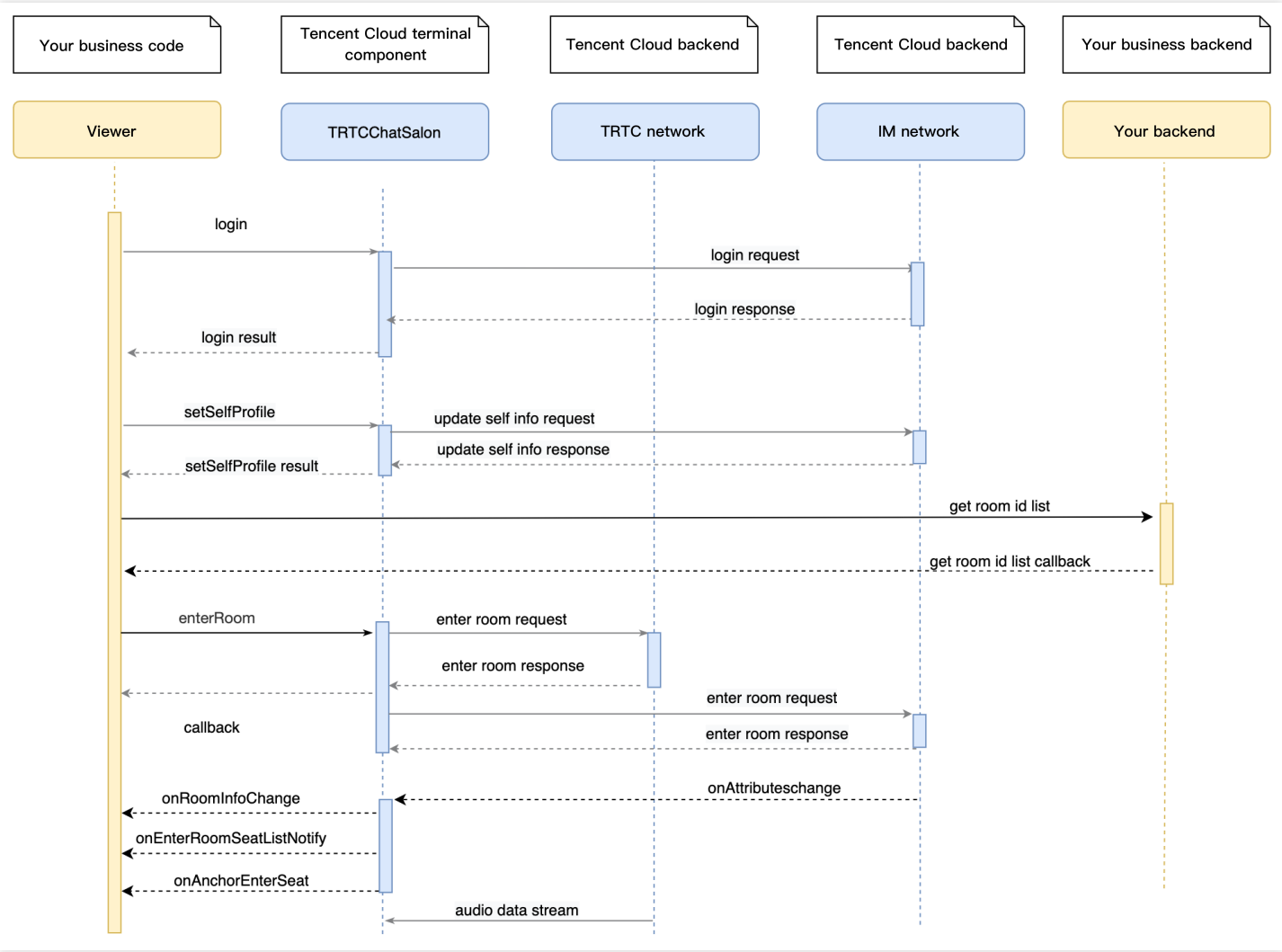
1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的语音沙龙房间列表。

> 说明：
> App 中的语音沙龙列表仅做演示使用，语音沙龙列表的业务逻辑千差万别，腾讯云暂不提供语音沙龙列表的管理服务，请自行管理您的语音沙龙列表。

3. 听众端调用 `getRoomInfoList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建语音沙龙时设置的简单描述信息。

> 注意：
> 如果您的语音沙龙列表包含了足够全面的信息，可跳过调用 `getRoomInfoList` 相关步骤。

4. 听众选择一个语音沙龙，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
6. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。



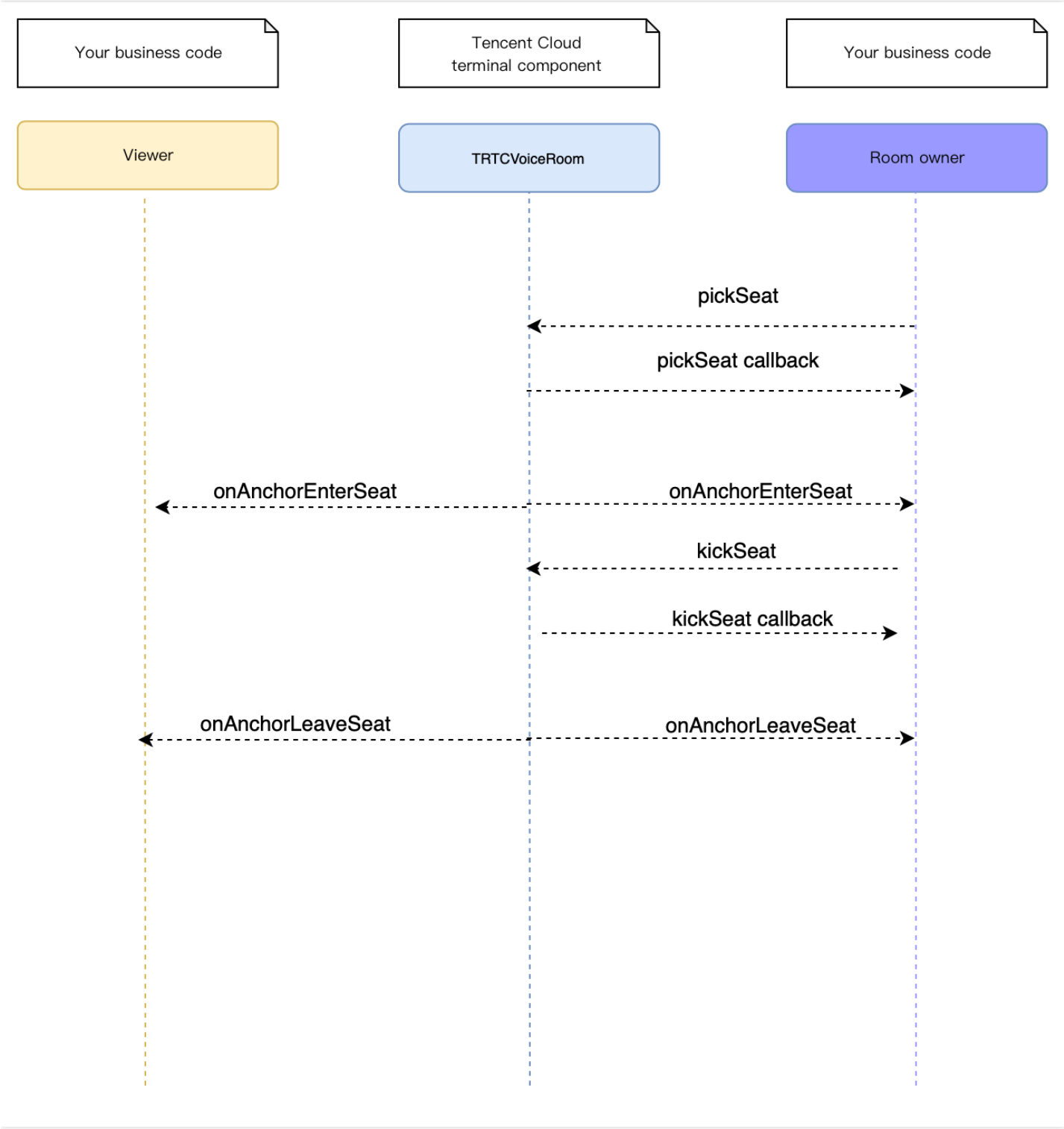

```
// 1.听众设置昵称和头像
mTRTCChatSalon.setSelfProfile("my_name", "my_face_url", null);

// 2.假定您从业务后台获取房间列表为 roomList
List<integer> roomList = GetRoomList();

// 3.通过调用 getRoomInfoList 获取房间的详细信息
mTRTCChatSalon.getRoomInfoList(roomList, new TRTCChatSalonCallback.RoomInfoCallback() {

@Override
public void onCallback(int code, String msg, List<trtcchatsalondef.roominfo> list) {
if (code == 0) {
// 此时可以刷新您自己的 UI 房间列表
}
}

});
```

```
// 4.传入 roomId 进入房间
mTRTCChatSalon.enterRoom(roomId, new TRTCChatSalonCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
}
}
});

// 5.进房成功后，收到 onRoomInfoChange 事件通知
@Override
public void onRoomInfoChange(TRTCChatSalonDef.RoomInfo roomInfo) {
mNeedRequest = roomInfo.needRequest;
mRoomName = roomInfo.roomName;
// UI 可以展示标题等
}

// 6.收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCChatSalonDef.UserInfo userInfo) {
}
```

步骤7：上下麦

- 房主端
- 听众端

1. `pickSeat` 传入听众 userId，可以抱人上麦，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。
2. `kickSeat` 传入对应用户的 userId 后，可以踢人下麦，房间内所有成员会收到 `onAnchorLeaveSeat` 的事件通知。



麦位操作后的事件通知顺序如下：callback > onAnchorEnterSeat 等独立事件。

```
// 1.房主抱人上麦位
mTRTCChatSalon.pickSeat("123", new TRTCChatSalonCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
// 2.收到 callback 回调
```

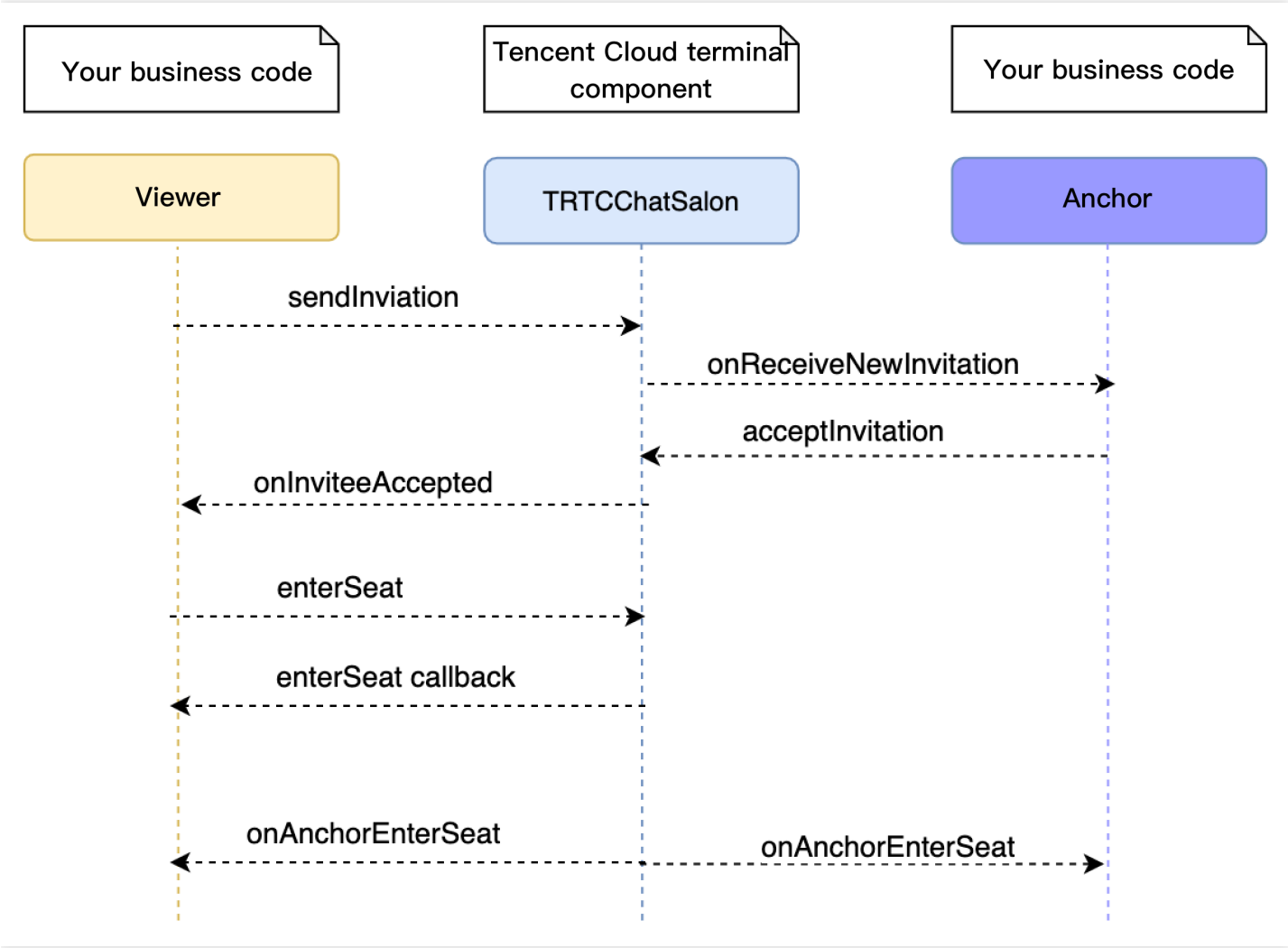
```
if (code == 0) {
}
}
});

// 3.听众成为主播，进入麦位通知，可以在这里判断听众是不是真的上麦成功
public void onAnchorEnterSeat(TRTCChatSalonDef.UserInfo user) {
}
```

步骤8：邀请信令的使用

如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

- 听众主动申请上麦
- 房主邀请听众上麦

1. 听众端调用 `sendInvitation` 传入房主的 userId 和业务的自定义命令字等，此时函数会返回一个 inviteId，记录该 inviteId。
2. 房主端收到 `onReceiveNewInvitation` 的事件通知，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `acceptInvitation` 并传入 inviteId。
4. 听众端收到 `onInviteeAccepted` 的事件通知，调用 `enterSeat` 进行上麦。




```
// 听众端视角
// 1.调用 sendInvitation, 请求上麦
String inviteId = mTRTCChatSalon.sendInvitation("ENTER_SEAT", ownerUserId, "123", null);

// 2.收到邀请的同意请求, 正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
if(id.equals(inviteId)) {
mTRTCChatSalon.enterSeat(null);
}
}

// 房主端视角
// 1.房主收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String cont
ent) {
if (cmd.equals("ENTER_SEAT")) {
```

```
// 2.房主同意听众请求
mTRTCChatSalon.acceptInvitation(id, null);
}
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
mTRTCChatSalon.sendRoomTextMsg("Hello Word!", null);
// 接收端：监听文本消息
mTRTCChatSalon.setDelegate(new TRTCChatSalonDelegate() {
@Override
public void onRecvRoomTextMsg(String message, TRTCChatSalonDef.UserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomCustomMsg` 回调。

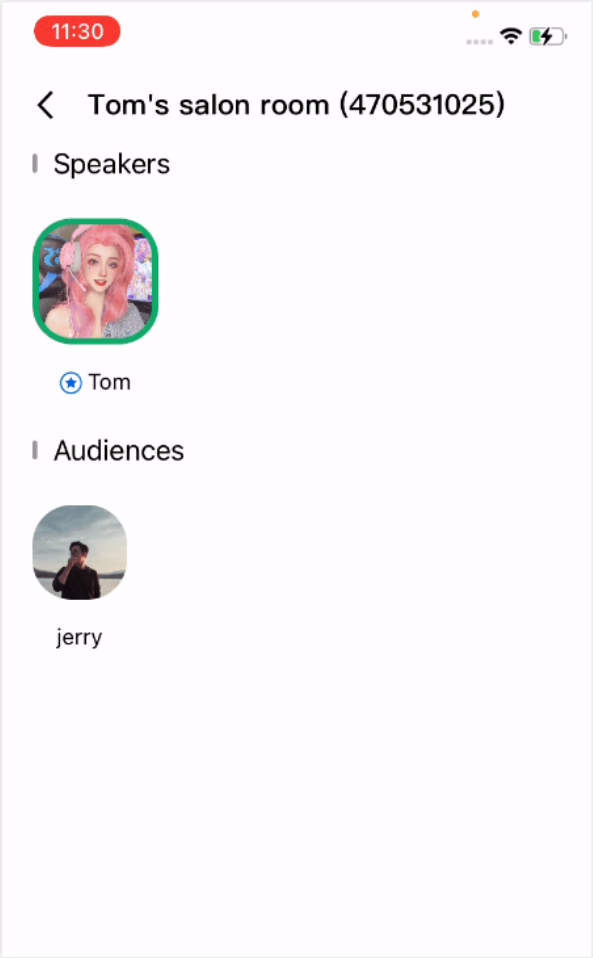
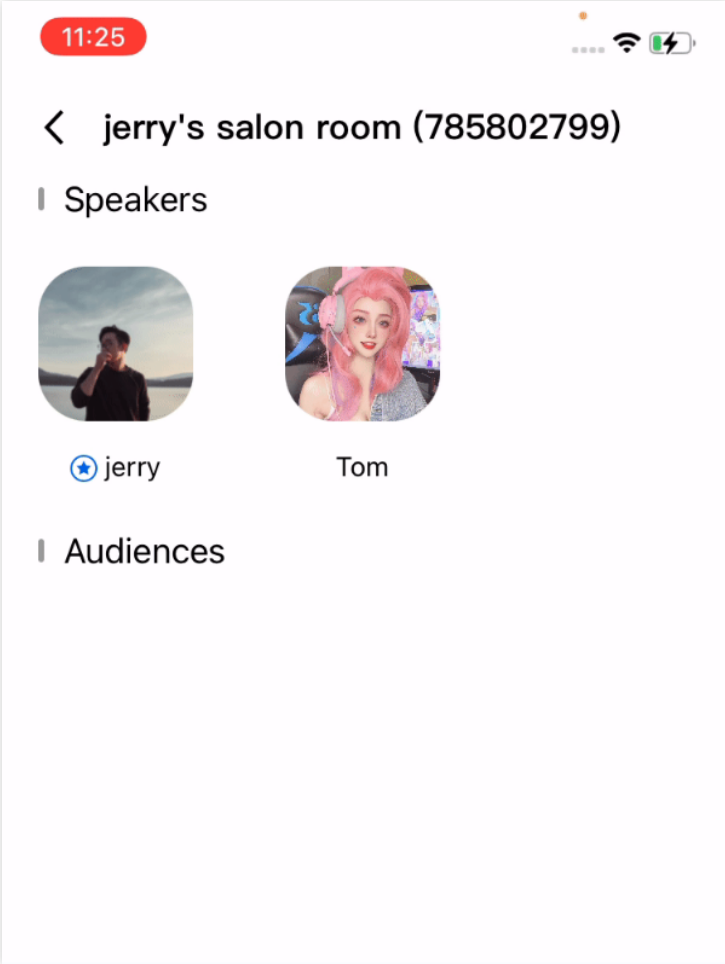
  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCChatSalon.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCChatSalon.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCChatSalon.setDelegate(new TRTCChatSalonDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCChatSalonDef.UserInfo userInf
o) {
if ("CMD_DANMU".equals(cmd)) {
// 收到弹幕消息
Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
} else if ("CMD_LIKE".equals(cmd)) {
// 收到点赞消息
Log.d(TAG, userInfo.userName + "给您点了个赞！");
}
}
});
```

语音沙龙(Flutter)

最近更新时间：2022-02-10 12:13:39

效果展示

房主麦位操作	听众麦位操作
	

如需快速接入语音沙龙功能，您可以直接基于我们提供的 Demo 进行修改适配，也可以使用我们提供的 TRTCChatSalon 组件并实现自定义 UI 界面。

复用 Demo 的 UI 界面

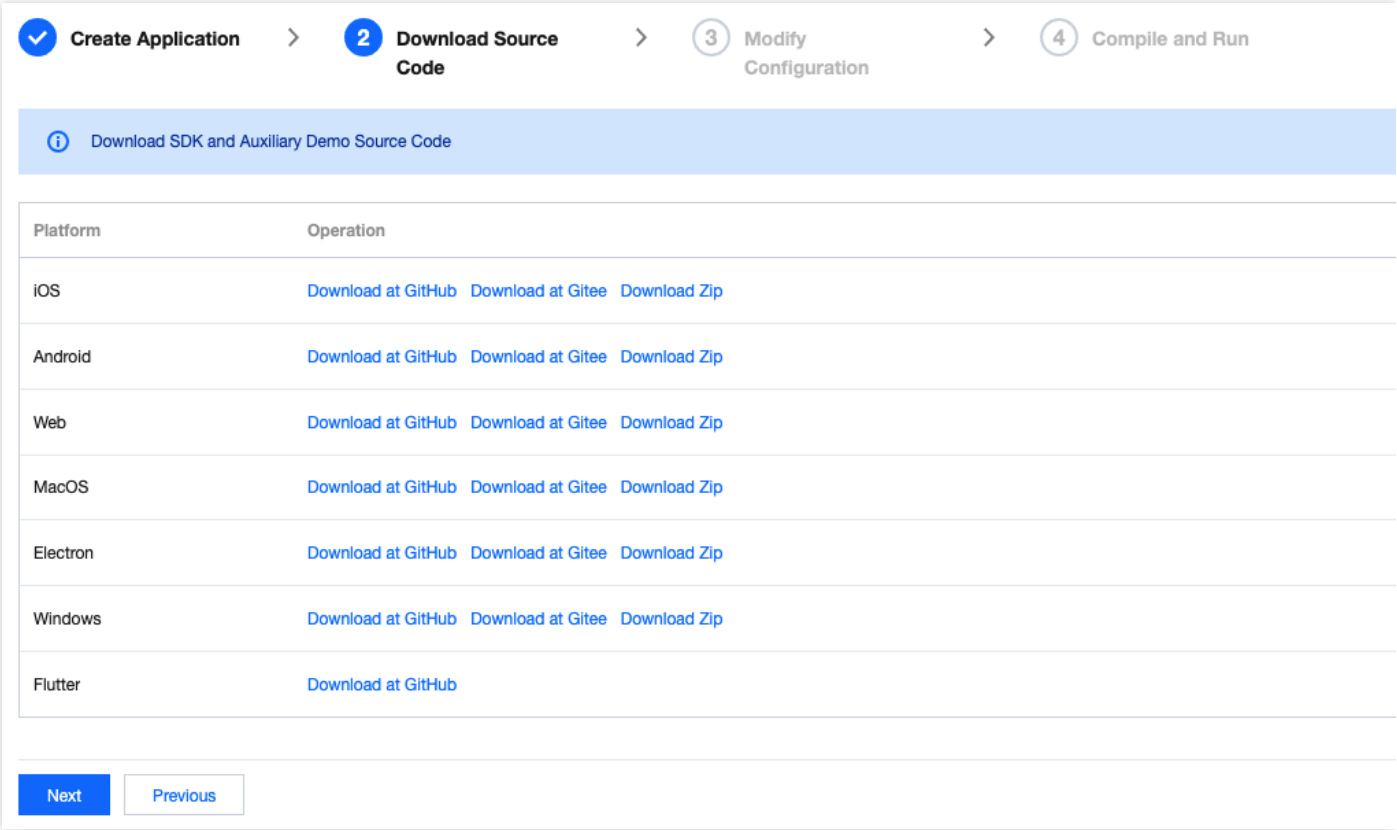
步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestChatSalon` ，单击【创建】。

> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。
2. 下载完成后，单击【已下载，下一步】。



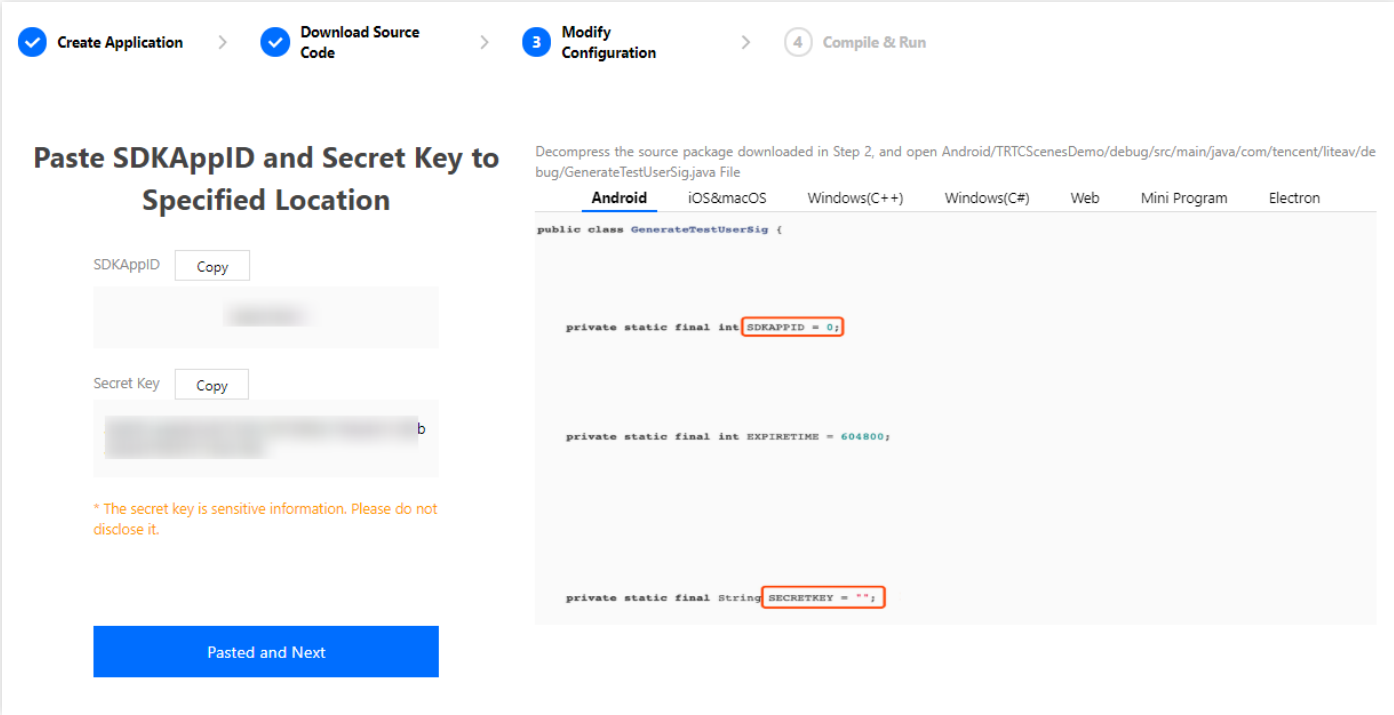
步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `/example/lib/debug/GenerateTestUserSig.dart` 文件。

3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：

- SDKAPPID：默认为 PLACEHOLDER， 请设置为实际的 SDKAppID。
- SECRETKEY：默认为 PLACEHOLDER， 请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：编译运行

注意：
安卓需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。
2. 编译运行调试：
   - Android 端
   - iOS 端
   i. 执行 `flutter run` 。
   ii. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击【运行】即可。

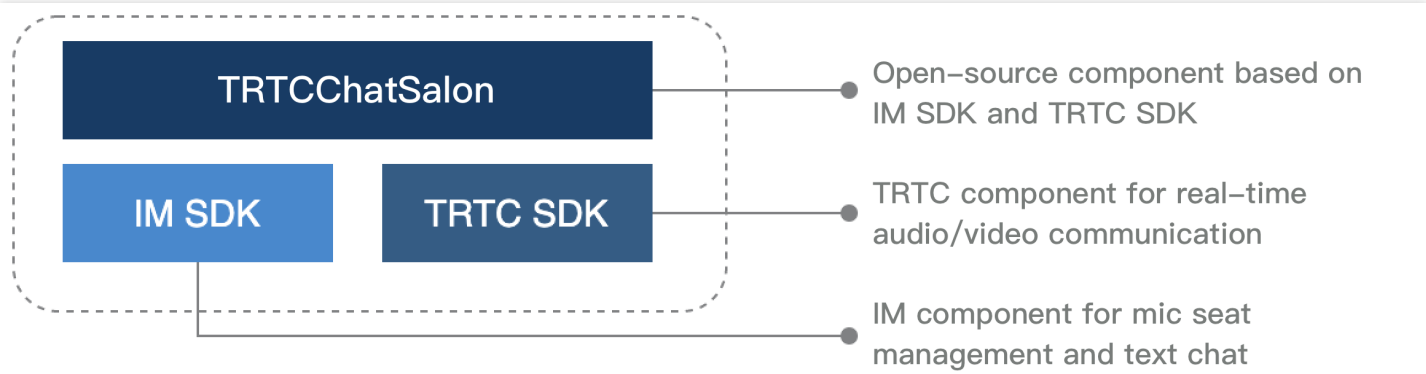
步骤5：修改 Demo 源代码

源码 中的 trtcchatsalondemo 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
base	UI 使用的基础类。
list	列表页和创建房间页。
room	主房间页面，包括房主和听众两种界面。
widget	通用控件。

实现自定义 UI 界面

源码 中的 trtcchatsalondemo 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCChatSalon，您可以在 `TRTCChatSalon.dart` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

语音沙龙组件 trtcchatsalondemo 依赖 TRTC SDK 和 IM SDK，您可以通过配置 `pubspec.yaml` 自动下载更新。

在项目的 `pubspec.yaml` 中写如下依赖：

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

步骤2：配置权限及混淆规则

- iOS 端
- Android 端

需要在 `Info.plist` 中加入对麦克风的权限申请：

```
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

步骤3：导入 TRTCChatSalon 组件

拷贝以下目录中的所有文件到您的项目中：

```
lib/TRTCChatSalonDemo/model/
```

步骤4：创建并登录组件

1. 调用 `sharedInstance` 接口可以创建一个 TRTCChatSalon 组件的实例对象。
2. 调用 `registerListener` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

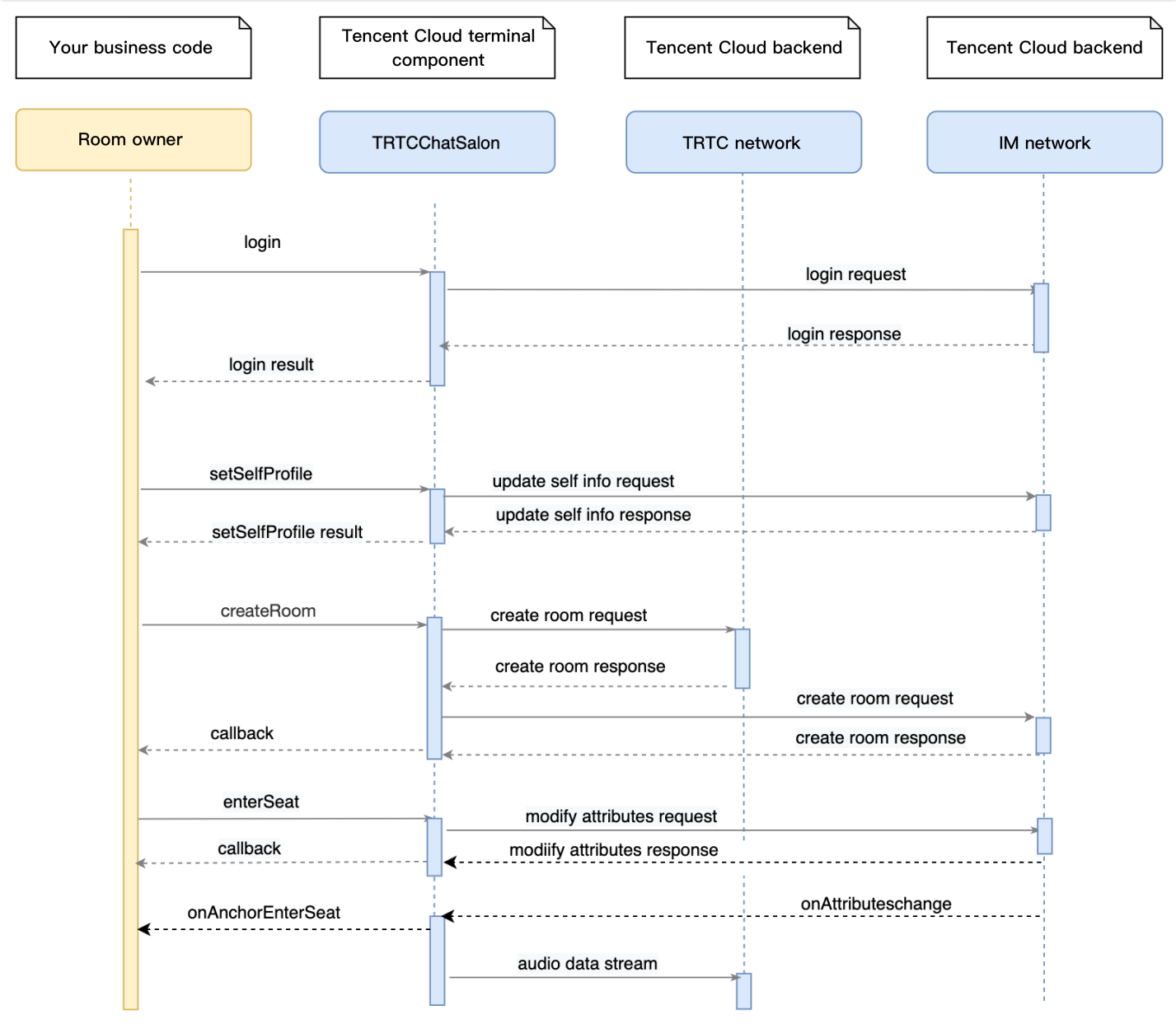
参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

```
TRTCChatSalon trtcVoiceRoom = await TRTCChatSalon.sharedInstance();
trtcVoiceRoom.registerListener(onVoiceListener);
ActionCallback resValue = await trtcVoiceRoom.login(
GenerateTestUserSig.sdkAppId,
userId,
GenerateTestUserSig.genTestSig(userId),
```

```
);
if (resValue.code == 0) {
//登录成功
}
```

步骤5：房主端开播

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的语音沙龙，此时传入房间 ID、房间名等房间属性信息。
3. 房主会收到有成员进入的 `TRTCChatSalonDelegate.onAudienceEnter` 的事件通知，此时会自动打开麦克风采集。



```
// 1.房主设置昵称和头像
trtcVoiceRoom.setSelfProfile("my_name", "my_face_url", null);

// 2.房主调用 createRoom 创建房间
ActionCallback resp = await trtcVoiceRoom.createRoom(
roomId,
RoomParam(
coverUrl: '房间封面图的 URL 地址',
roomName: '房间名称',
),
);
if (resp.code == 0) {
//3.占座成功
}

// 4.占座成功后， 收到 TRTCChatSalonDelegate.onAudienceEnter 事件通知
onVoiceListener(type, param) async {
switch (type) {
case TRTCChatSalonDelegate.onAudienceEnter:
//听众进入房间
break;
}
}
```

步骤6：听众端观看

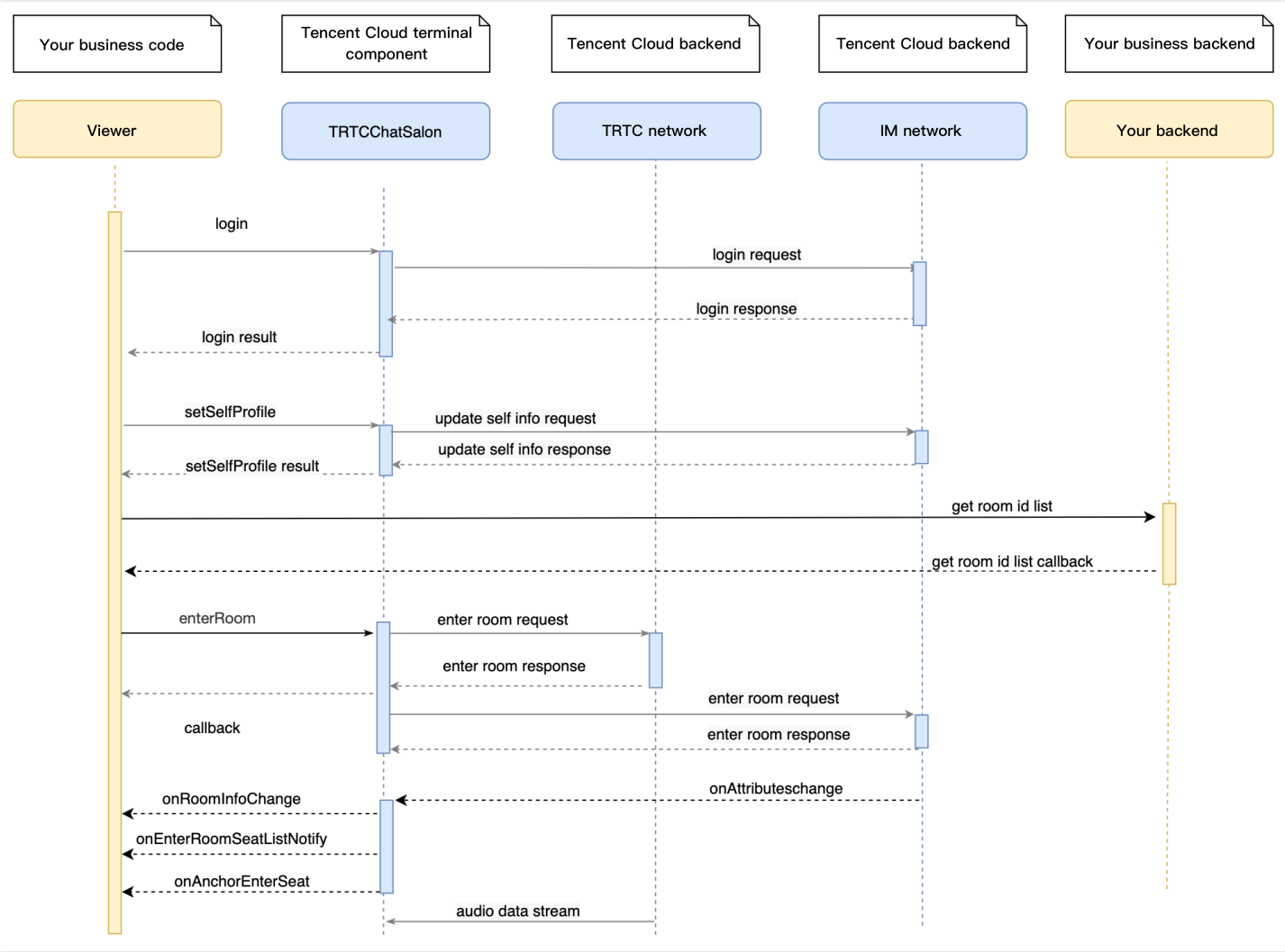
1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的语音沙龙房间列表。

> 说明：
> Demo 中的语音沙龙列表仅做演示使用，语音沙龙列表的业务逻辑千差万别，腾讯云暂不提供语音沙龙列表的管理服务，请自行管理您的语音沙龙列表。

3. 听众端调用 `getRoomList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建语音沙龙时设置的简单描述信息。

> 注意：
> 如果您的语音沙龙列表包含了足够全面的信息，可跳过调用 `getRoomList` 相关步骤。 并传入房间号即可进入该房间。

4. 进房后会收到组件的 `TRTCChatSalonDelegate.onAudienceEnter` 和 `TRTCChatSalonDelegate.onAudienceExit` 听众进退房通知，监听到事件回调后可以将变化然后刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `TRTCChatSalonDelegate.onAnchorEnterMic` 和 `TRTCChatSalonDelegate.onAnchorLeaveMic` 的事件通知。



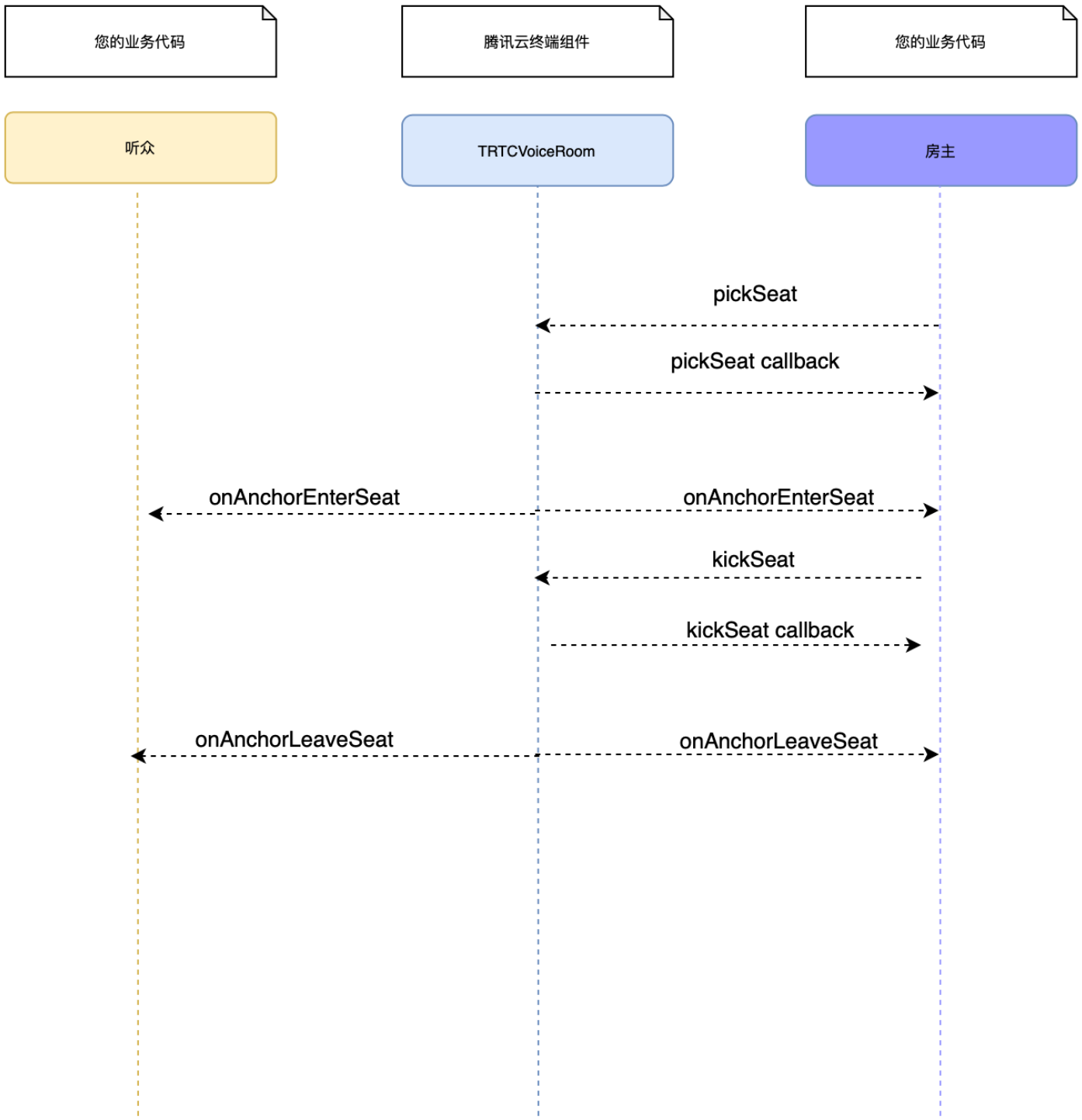

```
// 1.听众设置昵称和头像
trtcVoiceRoom.setSelfProfile("my_name", "my_face_url");

// 2.假定您从业务后台获取房间列表为 roomList
List<integer> roomList = GetRoomList();

// 3.通过调用 getRoomInfoList 获取房间的详细信息
RoomInfoCallback resp = await trtcVoiceRoom.getRoomInfoList(roomList);
if (resp.code == 0) {
//此时可以刷新您自己的 UI 房间列表
}
```

```
// 4.传入 roomId 进入房间
ActionCallback enterRoomResp =
await trtcVoiceRoom.enterRoom(_currentRoomId);
if (enterRoomResp.code == 0) {
//进房成功
}
// 5.进房成功后，收到处理事件通知
onVoiceListener(type, param) async {
switch (type) {
case TRTCChatSalonDelegate.onAudienceEnter:
//听众进入房间
break;
case TRTCChatSalonDelegate.onAudienceExit:
//听众离开房间
break;
case TRTCChatSalonDelegate.onAnchorLeaveMic:
//房主离开房间
break;
case TRTCChatSalonDelegate.onAnchorEnterMic:
//房主进入房间
break;
}
}
```

步骤7：上下麦

- 房主端
- 听众端

1. `leaveMic` 主动下麦,房间内所有成员会收到 `onAnchorLeaveMic` 的事件通知。
2. `kickMic` 传入对应用户的userId后，可以踢人下麦。房主踢人下麦，房间内所有成员会收到 `onAnchorLeaveMic` 的事件通知。



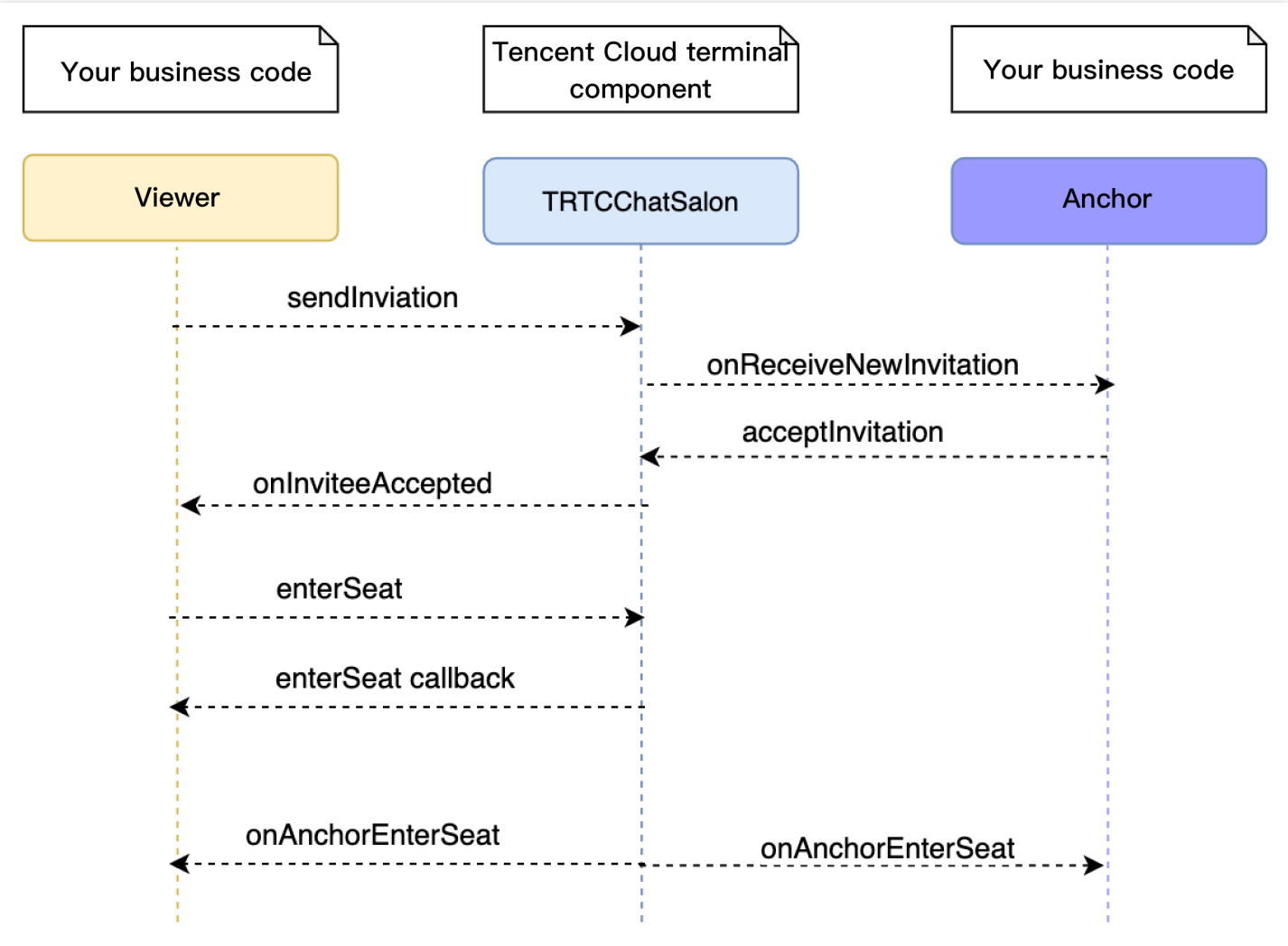

```
// 1.主动下麦
trtcVoiceRoom.leaveMic();

//2.踢人下麦
trtcVoiceRoom.kickMic(userId);
```

步骤8：举手要求上麦信令的使用

如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

听众主动申请上麦

1. 听众端调用 `raiseHand` 申请举手。
2. 房主端收到 `onRaiseHand` 的事件，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `agreeToSpeak` 并传入 userId。
4. 听众端收到 `onAgreeToSpeak` 的事件通知，调用 `enterMic` 进行上麦。




```
// 听众端视角
// 1.调用 sendInvitation，请求上麦
trtcVoiceRoom.raiseHand();

// 2.收到邀请的同意请求，正式上麦
onVoiceListener(type, param) async {
```

```
switch (type) {
case TRTCChatSalonDelegate.onRaiseHand:
trtcVoiceRoom.enterMic();
break;
}
}

// 房主端视角
// 1.房主收到请求
onVoiceListener(type, param) async {
switch (type) {
case TRTCChatSalonDelegate.onAgreeToSpeak:
trtcVoiceRoom.agreeToSpeak();
break;
}
}
```

步骤9：实现文字聊天和弹幕消息

通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
trtcVoiceRoom.sendRoomTextMsg("Hello Word!");
// 接收端：监听文本消息
onVoiceListener(type, param) async {
switch (type) {
case TRTCChatSalonDelegate.onRecvRoomTextMsg:
//收到群文本消息，可以用作文本聊天室
break;
}
}
```

TRTCChatSalon（iOS）

最近更新时间：2022-04-21 16:18:01

TRTCChatSalon 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音沙龙开播，听众进入语音沙龙收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

TRTCChatSalon 是一个开源的 Class，依赖腾讯云的两个闭源 SDK。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCChatSalon API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateQueue	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。

API	描述
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 nil，则获取房间内所有人的信息。

上下麦接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。

远端用户音频操作接口

API	描述

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCChatSalonDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语音沙龙房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCChatSalon单例对象。

```
/**
* 获取 TRTCChatSalon 单例对象
*
* - returns: TRTCChatSalon 实例
* - note: 可以调用 {@link TRTCChatSalon#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCChatSalon单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCChatSalon 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCChatSalon 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCChatSalon 实例不能再使用，需要重新调用 {@link TRTCChatSalon#
sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCChatSalon 事件回调，您可以通过 TRTCChatSalonDelegate 获得 TRTCChatSalon 的各种状态通知。

```
/**
* 设置组件回调接口
*
```

```
* 您可以通过 TRTCChatSalonDelegate 获得 TRTCChatSalon 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCChatSalon 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，可使用 {@link TRTCChatSalon#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCChatSalonDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

说明：

setDelegate 是 TRTCChatSalon 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送在主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
* - parameter queue 队列，TRTCChatSalon 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCChatSalon 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
userID:(NSString *)userID
userSig:(NSString *)userSig
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userID:userSig:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】 > 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallb
ack _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(ChatSalonParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语音沙龙房间列表，腾讯云暂不提供语音沙龙房间列表的管理服务，请自行管理您的语音沙龙房间列表。
roomParam	ChatSalonParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音沙龙，此时传入房间 ID、上麦是否需要房主确认等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音沙龙列表，可能包含多个语音沙龙房间的 roomId 和房间信息。
2. 听众选择一个语音沙龙，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(ChatSalonInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	ChatSalonInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(ChatSalonUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
callback	ChatSalonUserListCallback	用户详细信息回调。

上下麦接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。

```
- (void)enterSeat:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(enterSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onAnchorLeaveSeat` 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。

```
- (void)pickSeat:(NSString *)userID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(pickSeat(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)kickSeat:(NSString *)userID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(kickSeat(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	需要踢下麦的用户id。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
userSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)voluem NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userID mute:(BOOL)mute NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userID	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
isMute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
userID:(NSString *)userID
content:(NSString *)content
```

```
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendInvitation(cmd:userId:content:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userID	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(acceptInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(rejectInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCChatSalonDelegate事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
- (void)onError:(int)code
message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
- (void)onWarning:(int)code
message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)message
NS_SWIFT_NAME(onRoomDestroy(message:));
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(ChatSalonInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
roomInfo	ChatSalonInfo	房间信息。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSInteger)totalV
olume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
userVolumes	NSArray<TRTCVolumeInfo *>	各个用户音量信息。
totalVolume	int	整体音量信息。

麦位回调

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(ChatSalonUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(user:));
```

参数如下表所示：

参数	类型	含义
user	ChatSalonUserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(ChatSalonUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(user:));
```

参数如下表所示：

参数	类型	含义
user	ChatSalonUserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSString *)userID
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(userID:isMute:));
```

参数如下表所示：

参数	类型	含义
userID	String	麦位的用户id。
isMute	boolean	true：静音麦位； false：解除静音。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(ChatSalonUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	ChatSalonUserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(ChatSalonUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	ChatSalonUserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
userInfo:(ChatSalonUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	ChatSalonUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)cmd
message:(NSString *)message
```

```
userInfo:(ChatSalonUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(cmd:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	ChatSalonUserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
inviter:(NSString *)inviter
cmd:(NSString *)cmd
content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(identifier:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。
invitee	String	邀请人的用户 ID。

onInvitationTimeout

邀请超时。

```
- (void)onInvitationTimeout:(NSString *)identifier;
```

参数如下表所示：

参数	类型	含义
identifier	String	邀请 ID。

TRTCChatSalon（Android）

最近更新时间：2022-01-10 12:04:12

TRTCChatSalon 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音沙龙开播，听众进入语音沙龙收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

TRTCChatSalon 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音沙龙（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCChatSalon API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。

API	描述
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

上下麦接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。

远端用户音频操作接口

API	描述

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCChatSalonDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语音沙龙房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。

API	描述
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCChatSalon 单例对象。

```
public static synchronized TRTCChatSalon sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCChatSalon 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCChatSalon 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCChatSalon 事件回调，您可以通过 TRTCChatSalonDelegate 获得 TRTCChatSalon 的各种状态通知。

```
public abstract void setDelegate(TRTCChatSalonDelegate delegate);
```

> 说明：
> setDelegate 是 TRTCChatSalon 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCChatSalon 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
String userId, String userSig,
TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCChatSalonDef.RoomParam roomParam, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语音沙龙房间列表，腾讯云暂不提供语音沙龙房间列表的管理服务，请自行管理您的语音沙龙房间列表。
roomParam	RoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音沙龙，此时传入房间 ID、上麦是否需要房主确认等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
public abstract void destroyRoom(TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音沙龙列表，可能包含多个语音沙龙房间的 roomId 和房间信息。
2. 听众选择一个语音沙龙，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCChatSalonCallback.RoomInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCChatSalonCallback.UserListCallback userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

上下麦接口

enterSeat

主动上麦（听众端和房主均可调用）。

说明：
上麦成功后，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。

```
public abstract void enterSeat(TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

说明：
下麦成功后，房间内所有成员会收到 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void leaveSeat(TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

说明：
房主抱人上麦，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。

```
public abstract void pickSeat(String userId, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void kickSeat(String userId, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要踢下麦的用户id。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCChatSalonCallback.ActionCallback callbac
k);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCChatSalonCallback.ActionCa
llback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCChatSalonCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCChatSalonDelegate 事件回调

通用事件回调

onError

错误回调。

说明：
SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCChatSalonDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(String userId, int volume);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
volume	int	音量大小，取值：0 - 100。

麦位回调

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(TRTCChatSalonDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(TRTCChatSalonDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(String userId, boolean isMute);
```

参数如下表所示：

参数	类型	含义
userId	String	麦位的用户 id。
isMute	boolean	true：静音麦位；false：解除静音。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCChatSalonDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCChatSalonDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCChatSalonDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCChatSalonDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

onInvitationTimeout

邀请超时。

```
void onInvitationTimeout(String id);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。

版权所有：腾讯云计算（北京）有限责任公司

TRTCChatSalon（Flutter）

最近更新时间：2022-01-10 12:05:28

TRTCChatSalon 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音沙龙开播，听众进入语音沙龙收听/互动。
- 房主可以同意听众上麦、将麦上主播踢下麦。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息。

TRTCChatSalon 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音沙龙（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCChatSalon API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
registerListener	设置事件监听。
unRegisterListener	销毁事件监听。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。

API	描述
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getRoomMemberList	获取房间内所有用户信息。
getArchorInfoList	获取房间主播列表。
getUserInfoList	获取指定 userId 的用户信息。

上下麦接口

API	描述
enterMic	听众上麦。
leaveMic	主播下麦。
muteMic	静音/解除静音某个麦位（房主调用）。
kickMic	踢人下麦（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。

申请上麦信令相关接口

API	描述
raiseHand	听众申请上麦。
agreeToSpeak	房主同意上麦。
refuseToSpeak	房主拒绝上麦。

TRTCChatSalonDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onKickedOffline	被踢下线。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onAnchorListChange	主播列表发生变化的回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onAnchorEnterMic	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveMic	有成员下麦（主动下麦/房主踢人下麦）。
onMicMute	主播禁麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。

申请上麦信令事件回调

API	描述
onRaiseHand	有听众举手，申请上麦。
onAgreeToSpeak	听众申请举手后，收到房主同意举手的回调。
onRefuseToSpeak	听众申请举手后，房主拒绝举手的回调。

版权所有：腾讯云计算（北京）有限责任公司

SDK 基础函数

sharedInstance

获取 TRTCChatSalon 单例对象。

```
static Future<TRTCChatSalon> sharedInstance()
```

destroySharedInstance

销毁 TRTCChatSalon 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCChatSalon 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
static void destroySharedInstance()
```

registerListener

设置事件监听

```
void registerListener(VoiceListenerFunc func)
```

> 说明：
> setDelegate 是 TRTCChatSalon 的代理回调。

unRegisterListener

移除组件事件监听接口。

```
void unRegisterListener(VoiceListenerFunc func)
```

参数如下表所示：

参数	类型	含义
func	VoiceListenerFunc	TRTCChatSalon 中的各种状态通知，会派发到您指定的 func函数

login

登录。

```
Future<ActionCallback> login(int sdkAppId, String userId, String userSig)
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

logout

登出。

```
Future<ActionCallback> logout()
```

setSelfProfile

修改个人信息。

```
Future<ActionCallback> setSelfProfile(String userName, String avatarURL)
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。

房间相关接口函数

createRoom

创建房间（新建房间时调用）。

```
Future<ActionCallback> createRoom(int roomId, RoomParam roomParam)
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语音沙龙房间列表，腾讯云暂不提供语音沙龙房间列表的管理服务，请自行管理您的语音沙龙房间列表。
roomParam	RoomParam	房间信息，用于房间描述的信息。例如房间名称、封面信息等。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音沙龙，此时传入房间 ID等房间属性信息。
2. 房主还会收到麦位表有成员进入的 `onAnchorEnterMic` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
Future<ActionCallback> destroyRoom()
```

enterRoom

进入房间（听众调用）。

```
Future<ActionCallback> enterRoom(int roomId)
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音沙龙列表，可能包含多个语音沙龙房间的 roomId 和房间信息。
2. 听众选择一个语音沙龙，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会可查询 `getArchorInfoList` 获取主播列表，并根据 `getRoomMemberList` 获取房间所有用户列表，减去主播列表可以得到听众列表。
4. 进房后还会收到麦位表有主播进入的 `onAnchorEnterMic` 的事件通知。

exitRoom

退出房间。

```
Future<ActionCallback> exitRoom()
```

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
Future<RoomInfoCallback> getRoomInfoList(List<String> roomIdList)
```

参数如下表所示：

参数	类型	含义
roomIdList	List<String>	房间号列表。

getRoomMemberList

获取房间内所有成员列表。

> 说明：
> IM 直播聊天群默认只能拉取最近31个成员列表。

```
Future<MemberListCallback> getRoomMemberList(double nextSeq)
```

参数如下表所示：

参数	类型	含义
nextSeq	double	分页拉取标志，第一次拉取填0，回调成功如果 nextSeq 不为零，需要分页，传入再次拉取，直至为0。

getArchorInfoList

获取房间内主播列表。

```
Future<UserListCallback> getArchorInfoList()
```

getUserInfoList

获取指定 userId 的用户信息。

```
Future<UserListCallback> getUserInfoList(List<String> userIdList)
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。

上下麦接口

enterMic

上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onAnchorEnterSeat` 的事件通知。

```
Future<ActionCallback> enterMic();
```

调用该接口会立即修改麦位表。听众需先调用 `raiseHand` 向房主申请，收到 `onAgreeToSpeak` 后再调用该函数。

leaveMic

主播下麦。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onAnchorLeaveMic` 的事件通知。

```
Future<ActionCallback> leaveMic()
```

muteMic

静音/解除静音某个麦位（房主调用）。

> 说明：
> 改变麦位的状态后，房间内所有成员会收到 `onAnchorListChange` 和 `onMicMute` 的事件通。

```
Future<ActionCallback> muteMic(bool mute)
```

kickMic

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onAnchorLeaveMic` 的事件通知。

```
Future<ActionCallback> kickMic(String userId)
```

参数如下表所示：

参数	类型	含义
userId	String	需要踢下麦的用户 ID。

调用该接口会立即修改麦位表。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
void startMicrophone(int quality)
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

stopMicrophone

停止麦克风采集。

```
void stopMicrophone()
```

muteLocalAudio

静音/取消静音本地的音频。

```
void muteLocalAudio(bool mute)
```

参数如下表所示：

参数	类型	含义
mute	bool	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
void setSpeaker(bool useSpeaker)
```

参数如下表所示：

参数	类型	含义
useSpeaker	bool	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
void setAudioCaptureVolume(int volume)
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
void setAudioPlayoutVolume(int volume)
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
void muteRemoteAudio(String userId, bool mute)
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	bool	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
void muteAllRemoteAudio(bool mute)
```

参数如下表所示：

参数	类型	含义
mute	bool	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
TXAudioEffectManager getAudioEffectManager()
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
Future<ActionCallback> sendRoomTextMsg(String message)
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。

申请上麦信令相关接口

raiseHand

听众申请上麦。

```
void raiseHand()
```

agreeToSpeak

房主同意上麦。

```
Future<ActionCallback> agreeToSpeak(String userId)
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

refuseToSpeak

房主拒绝用户上麦。

```
Future<ActionCallback> refuseToSpeak(String userId)
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

TRTCChatSalonDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

参数如下表所示：

参数	类型	含义
errCode	int	错误码。
errMsg	String	错误信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onWarning

警告回调。

参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onKickedOffline

其他用户登录了同一账号，被踢下线。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

onAnchorListChange

主播列表发生变化的通知。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	bool	静音状态。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
volume	int	音量大小，取值：0 - 100。

麦位回调

onAnchorEnterMic

有成员上麦。

参数如下表所示：

参数	类型	含义
userId	String	进房的用户 ID。
userName	String	用户昵称。

参数	类型	含义
userAvatar	String	头像地址。
mute	bool	麦位状态，默认开麦。

onAnchorLeaveMic

有成员下麦。

参数如下表所示：

参数	类型	含义
userId	String	退房的用户 ID。

onMicMute

房主是否禁麦。

参数如下表所示：

参数	类型	含义
userId	String	下麦的用户 ID。
mute	bool	麦位状态。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

参数如下表所示：

参数	类型	含义
userId	String	上麦的用户 ID。
userName	String	用户昵称。
userAvatar	String	头像地址。

onAudienceExit

收到听众退房通知。

参数如下表所示：

参数	类型	含义
userId	String	下麦的用户 ID。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

参数如下表所示：

参数	类型	含义
message	String	文本消息。
sendId	String	发送者用户 ID。
userAvatar	String	发送者用户头像。
userName	String	发送者用户昵称。

申请举手信令事件回调

onRaiseHand

收到新的上麦请求。

参数如下表所示：

参数	类型	含义
userId	String	申请举手的用户 ID。

onAgreeToSpeak

房主同意上麦的回调。

参数如下表所示：

参数	类型	含义
userId	String	房主的用户 ID。

onRefuseToSpeak

房主拒绝上麦的回调。

参数如下表所示：

参数	类型	含义
userId	String	房主的用户 ID。

实时互动课堂
实时互动课堂(Electron)

最近更新时间：2022-04-02 16:28:10

Demo 体验

Windows 版 MacOS 版

效果展示

您可以下载、安装我们已经构建好的 App 安装包， 体验实时互动课堂的能力效果。不仅提供了基础的音视频通话、屏幕分享、白板、文字聊天等基础功能，还实现了全员禁麦、学生举手申请发言、老师邀请学生发言、点名、签到等高级功能。

跑通实时互动课堂源代码

步骤1：创建应用并获取 SDKAppID 和密钥

如果您之前已经创建过腾讯云实时音视频的应用，可以跳过该步骤，直接使用之前创建应用的 SDKAppID 和密钥。

1. 登录实时音视频控制台，选择**开发辅助** > **快速跑通 Demo**，在创建应用页签，输入您的应用名称，例如 `TestTRTC` ，单击创建按钮。

2. 跳过**下载源码**页签，直接单击**下一步**按钮，进入**修改配置**页签，记录下页面上显示的 SDKAppID 和密钥，后续步骤将会用到。

步骤2：配置即时通信 IM

> 说明：
> 实时互动课堂同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

1. 进入**相关云服务**菜单，单击下图中**即时通信 IM 应用**跳转到 IM 应用管理页面。

2. 找到刚创建的应用，单击进入该应用管理页面。

3. 打开菜单**功能配置 > 登录与消息**，如下图所示，单击**登录设置**区域的**编辑**链接，将 **Web 端可同时在线个数**设置为大于等于 2 的值（目前本应用最多需要同时登录 2 个 Web IM 实例，可以设置更多一些，以备后续使用）。

步骤3：运行环境准备

本代码工程的运行依赖于 node.js 和 yarn。

1. **安装 node.js：**
   建议 node.js 使用 14.16.0 以上版本，安装完成后，在命令行终端执行以下命令检查 node.js 版本。

```
node --version
```

2. **安装 yarn：**

- 如果 node.js 版本小于 16.10，在命令行终端执行以下命令安装 yarn。

```
npm i -g corepack
```

- 如果 node.js 版本大于等于 16.10，在命令行终端执行以下命令安装 yarn。

```
corepack enable
```

> 注意：
> Window 10、11 下如果遇到权限不足的错误提示，请尝试以管理员身份，在 cmd 中执行。

步骤4：克隆代码工程

您可以直接下载代码，解压后进入代码目录 `trtc-education-electron`，或者使用 git 工具克隆代码工程。使用 git 工具克隆代码工程，请在命令行终端执行以下命令：

```
git clone https://github.com/TencentCloud/trtc-education-electron.git
cd trtc-education-electron
```

步骤5：配置 SDKAppID 和密钥

1. 找到并打开 `src/main/config/generateUserSig.js` 文件。

2. 设置 `generateUserSig.js` 文件中的相关参数，用于生成身份认证用的用户签名 UserSig：：
   - SDKAPPID：默认为 0，请设置为 步骤1 创建应用的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为 步骤1 创建应用的密钥。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤6：开发模式运行

在命令行终端中，进入代码目录 `trtc-education-electron`，执行以下命令。

```
yarn
yarn start
```

> 注意：
>
> - 第一次执行 yarn 命令安装依赖时，Window10、Window11 下如果遇到权限不足的错误提示，请尝试以管理员身份，在 cmd 中执行一次。之后就可以以普通用户身份在 cmd 中或者集成开发工具自带终端中执行，例如：Visual Studio Code、WebStorm 等。
> - 安装依赖过程中，如遇到 Electron 下载慢甚至卡住不动等问题，您可以 联系我们 解决。

步骤7：构建安装包、运行

在命令行终端中，进入代码目录 `trtc-education-electron`，执行以下命令构建安装包，构建好的安装包位于 `trtc-education-electron/build/release` 目录下，可以安装运行。

```
yarn package
```

> 注意：
> 只能使用 Mac 电脑构建 Mac 安装包，使用 Windows 电脑构建 Windows 安装包。

技术咨询

了解更多详情您可以联系我们或发送邮件至colleenyu@tencent.com。

参考文档

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

trtc-electron-education API

最近更新时间：2022-01-10 15:15:39

组件 trtc-electron-education 是针对实时互动课堂场景中使用实时音视频 TRTC 和即时通信 IM 能力的二次封装，在封装基本的音视频聊天、屏幕分享能力的同时，针对在线教育场景封装了老师开始问答、学生举手、老师邀请学生上台回答、结束回答等相关能力。
trtc-electron-education 是一个开源的 npm 组件，依赖腾讯云的两个闭源 SDK，具体实现过程请参见 实时互动课堂(Electron)。

- TRTC SDK：使用 TRTC SDK 作为低延时音视频通话组件。
- IM SDK：使用 IM SDK 发送和处理信令消息。

组件接入

```
// yarn 方式引入
yarn add trtc-electron-education
// npm 方式引入
npm i trtc-electron-education --save
```

组件参数

必填的关键参数介绍如下表所示。

参数	类型	说明
sdkAppId	number	必填参数，您可以在 实时音视频控制台 中查看 SDKAppID。
userID	string	必填参数，用户 ID，可以由您的帐号体系指定。
userSig	string	必填参数，身份签名（相当于登录密码），由 userID 计算得出，具体计算方法请参见 如何计算及使用 UserSig。

初始化示例

```
import TrtcElectronEducation from 'trtc-electron-education';
const rtcClient = new TrtcElectronEducation({
sdkAppId: 1400***803,
```

```
userID: '123'
userSig: 'eJwtzM9****-reWMQw_'
});
```

组件概览

基础函数

on(EventCode, handler, context)

用于监听组件派发的事件。

参数：

参数名	类型	说明
EventCode	String	事件码
handler	Function	监听函数
context	Object	当前执行上下文

示例：

```
const EVENT = rtcClient.EVENT
rtcClient.on(EVENT.MESSAGE_RECEIVED, onMessageReceived);
```

off(EventCode, handler)

取消事件监听。

参数：

参数名	类型	说明
EventCode	String	事件码
handler	Function	需要取消的具名监听函数

示例：

```
const EVENT = rtcClient.EVENT
rtcClient.off(EVENT.MESSAGE_RECEIVED, onMessageReceived);
```

版权所有：腾讯云计算（北京）有限责任公司

createRoom(params: CreateRoomParams)

老师创建教室。

参数：

参数名	类型	说明
classId	number	教室 ID
nickName	string	昵称
avatar	string	头像地址，可以不填

示例：

```
interface CreateRoomParams {
classId: number; // 教室 ID
nickName: string; // 昵称
avatar?:string; // 头像地址
}
rtcClient.createRoom(params).then(() => {
})
```

destroyRoom(classId: number)

老师退出教室，销毁房间。

参数：

参数名	类型	说明
classId	number	教室 ID

示例：

```
rtcClient.destroyRoom(classId)
```

enterRoom(params: EnterRoomParams)

老师开始上课，学生进入教室，准备听课。

参数：

参数名	类型	说明
classId	number	教室 ID

参数名	类型	说明
nickName	string	昵称
role	string	角色，teacher 表示老师，student 表示学生
avatar	string	头像地址，可以不填

示例：

```
interface EnterRoomParams {
role: string; // 角色
classId: number; // 教室 ID
nickName?: string; // 昵称
avatar?:string; // 头像地址
}
rtcClient.enterRoom(params).then(() => {
})
```

exitRoom(role:string, classId: number)

老师下课，学生退出课堂。
参数：

参数名	类型	说明
classId	number	教室 ID
role	string	角色，teacher 表示老师，student 表示学生

示例：

```
rtcClient.exitRoom(role, classId);
```

举手操作环节函数

startQuestionTime(classId: number) 老师开始问答时间，老师端广播通知，学生收到开始问答事件，开启举手功能。 参数：

参数名	类型	说明
classId	number	教室 ID

示例：

```
rtcClient.startQuestionTime(classId)
```

raiseHand()

学生举手，学生发送举手通知，老师端收到学生举手通知。

参数：无

示例：

```
rtcClient.raiseHand()
```

inviteToPlatform(userID: string)

老师邀请学生上台回答，老师端选择举手列表里的学生 userID ，发送邀请通知，学生端收到邀请上台回答事件，受邀请的学生开麦。如果没有学生举手，老师直接点名回答，学生端收到邀请上台回答事件，被点名的学生开麦。

参数：

参数名	类型	说明
userID	string	用户 ID

示例：

```
rtcClient.inviteToPlatform(userID).then(() => {
})
```

finishAnswering(userID: string)

结束回答，老师端结束学生端回答，学生收到结束回答的通知，指定的学生停止连麦。

参数：

参数名	类型	说明
userID	string	用户 ID

示例：

```
rtcClient.finishAnswering(userID).then(() => {
})
```

stopQuestionTime(classId: number)

停止问答时间，老师端停止问答时间，学生端收到停止回答时间的通知，已连麦的学生要停止连麦，关闭举手功能。

参数：

参数名	类型	说明
classId	number	教室 ID

示例：

```
rtcClient.stopQuestionTime(classId)
```

推拉流操作相关函数

getScreenShareList()

获取屏幕分享窗口列表。

参数：无

示例：

```
rtcClient.getScreenShareList();
```

startScreenCapture(source: SourceParam)

选择分享屏幕，开始推流。

参数：

参数名	类型	说明
type	number	采集源类型
sourceId	string	采集源 ID，对于窗口，该字段指示窗口句柄；对于屏幕，该字段指示屏幕 ID
sourceName	string	采集源名称，UTF8 编码

示例：

```
interface SourceParam {
type: number; // 采集源类型
sourceId: string; // 采集源 ID
sourceName: string; // 采集源名称，UTF8 编码
}
rtcClient.startScreenCapture({
```

```
type,
sourceId,
sourceName
})
```

startRemoteView(params: RemoteParams)

开始显示远端视频画面或屏幕分享画面。

参数：

参数名	类型	说明
userID	string	用户 ID
streamType	number	画面类型，1表示大画面，2表示小画面，3表示屏幕分享
view	HTMLElement	承载显示画面的 DOM

示例：

```
interface RemoteParams {
userID: string; // 用户 ID
streamType: number; // 画面类型,1-大画面, 2-小画面, 3-屏幕分享
view: HTMLElement; //承载显示画面的 DOM
}
const view = document.getElementById('localVideo');
rtcClient.startRemoteView({
userID: userID,
streamType: 1,//1-大画面, 2-小画面, 3-屏幕分享
view: view
});
```

stopRemoteView(params: StopRemoteParams)

停止显示远端视频画面或屏幕分享画面，同时不再拉取该远端用户的数据流。

参数：

参数名	类型	说明
userID	string	用户 ID
streamType	number	画面类型，1表示大画面，2表示小画面，3表示屏幕分享

示例：

```
interface StopRemoteParams {
userID: string; // 用户 ID
streamType: number; // 画面类型,1-大画面, 2-小画面, 3-屏幕分享
}
rtcClient.stopRemoteView({
userID: userID,
streamType: 1 //1-大画面, 2-小画面, 3-屏幕分享
});
```

消息收发相关函数

sendTextMessage(params: MessageParams)

发送聊天室消息。

参数：

参数名	类型	说明
classId	number	教室 ID
message	string	消息文本

示例：

```
interface MessageParams {
classId: number; // 教室 ID
message: string; // 消息文本
}
rtcClient.sendTextMessage(params).then(() => {
})
```

sendCustomMessage(userID: string, data: string)

发送自定义 C2C 消息。

参数：

参数名	类型	说明
userID	string	用户 ID
data	string	自定义消息

示例：

```
rtcClient.sendCustomMessage(userID, JSON.stringify(params)
```

sendGroupCustomMessage(classId: number, data: string)

发送自定义群组消息。

参数：

参数名	类型	说明
classId	number	教室 ID
data	string	自定义消息

示例：

```
rtcClient.sendGroupCustomMessage(classId, JSON.stringify(params))
```

设备操作相关函数

openCamera(view: HTMLElement)

打开摄像头。

参数：

参数名	类型	说明
view	HTMLElement	承载显示画面的 DOM

示例：

```
const domEle = document.getElementById('localVideo');
rtcClient.openCamera(domEle);
```

closeCamera()

关闭摄像头。

参数：无

示例：

```
rtcClient.closeCamera();
```

getCameraList()

获取摄像头列表。

参数：无

示例：

```
rtcClient.getCameraList()
```

setCurrentCamera(deviceId:string)

设置摄像头，参数是从 getCameraDevicesList 中得到的设备 ID。

参数：

参数名	类型	说明
deviceId	string	设备 ID

示例：

```
rtcClient.setCurrentCamera(deviceId)
```

openMicrophone()

打开麦克风。

参数：无

示例：

```
rtcClient.openMicrophone();
```

closeMicrophone()

关闭麦克风。

参数：无

示例：

```
rtcClient.closeMicrophone();
```

getMicrophoneList()

获取麦克风设备列表。

参数：无

示例：

```
rtcClient.getMicrophoneList()
```

setCurrentMicDevice(micId:string)

设置麦克风，参数是从 getMicDevicesList 中得到的设备 ID。

参数：

参数名	类型	说明
micId	string	设备 ID

示例：

```
rtcClient.setCurrentMicDevice(micId)
```

setBeautyStyle(params: BeautyParams)

设置美颜、美白、红润效果级别。

参数：

参数名	类型	说明
beautyStyle	number	1表示光滑，适用于美女秀场，效果比较明显 2表示自然，磨皮算法更多地保留了面部细节，主观感受上会更加自然
beauty	number	美颜级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显
white	number	美白级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显
ruddiness	number	红润级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显，该参数 Windows 平台暂未生效

示例：

```
interface BeautyParams {
beautyStyle: number;//光滑、自然
beauty: number; //美颜级别
white: number; //美白级别
ruddiness: number; //红润级
}
rtcClient.setBeautyStyle({
beautyStyle: 1,
beauty: 5,
white: 5,
```

```
ruddiness: 5
})
```

组件事件

示例

```
const EVENT = rtcClient.EVENT
rtcClient.on(EVENT.STUDENT_RAISE_HAND, () => {
//学生举手
})
```

详细事件

CODE	说明
ENTER_ROOM_SUCCESS	成功进入房间
LEAVE_ROOM_SUCCESS	成功离开房间
TEACHER_ENTER	老师进入房间
TEACHER_LEAVE	老师离开房间
STUDENT_ENTER	学生加入房间
STUDENT_LEAVE	学生离开房间
SCREEN_SHARE_ADD	老师开启屏幕分享
SCREEN_SHARE_REMOVE	老师关闭屏幕分享
REMOTE_VIDEO_ADD	远端视频流添加事件，当远端用户发布视频流后会收到该通知
REMOTE_VIDEO_REMOVE	远端视频流移出事件，当远端用户取消发布视频流后会收到该通知
REMOTE_AUDIO_ADD	远端音频流添加事件
REMOTE_AUDIO_REMOVE	远端音频流移除事件
ROOM_DESTROYED	房间被销毁
QUESTION_TIME_STARTED	开始问答时间
QUESTION_TIME_STOPPED	结束问答时间

版权所有：腾讯云计算（北京）有限责任公司

CODE	说明
STUDENT_RAISE_HAND	学生举手
BE_INVITED_TO_PLATFORM	被邀请回答问题、点名
ANSWERING_FINISHED	结束回答，禁音
MESSAGE_RECEIVED	收到消息
KICKED_OUT	同一账户在其他地方登录，被踢下线
ERROR	异常
WARNING	警告
	

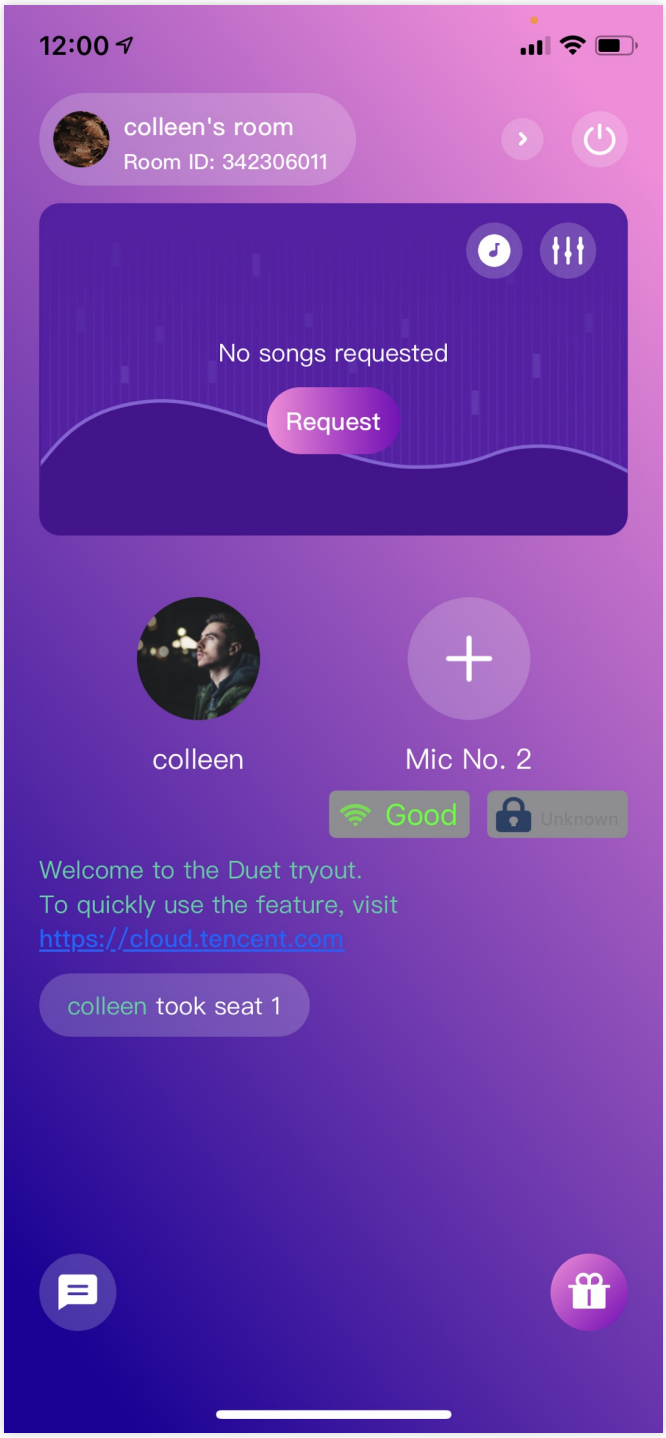
在线 K 歌_KTV
在线 K 歌_KTV（iOS）

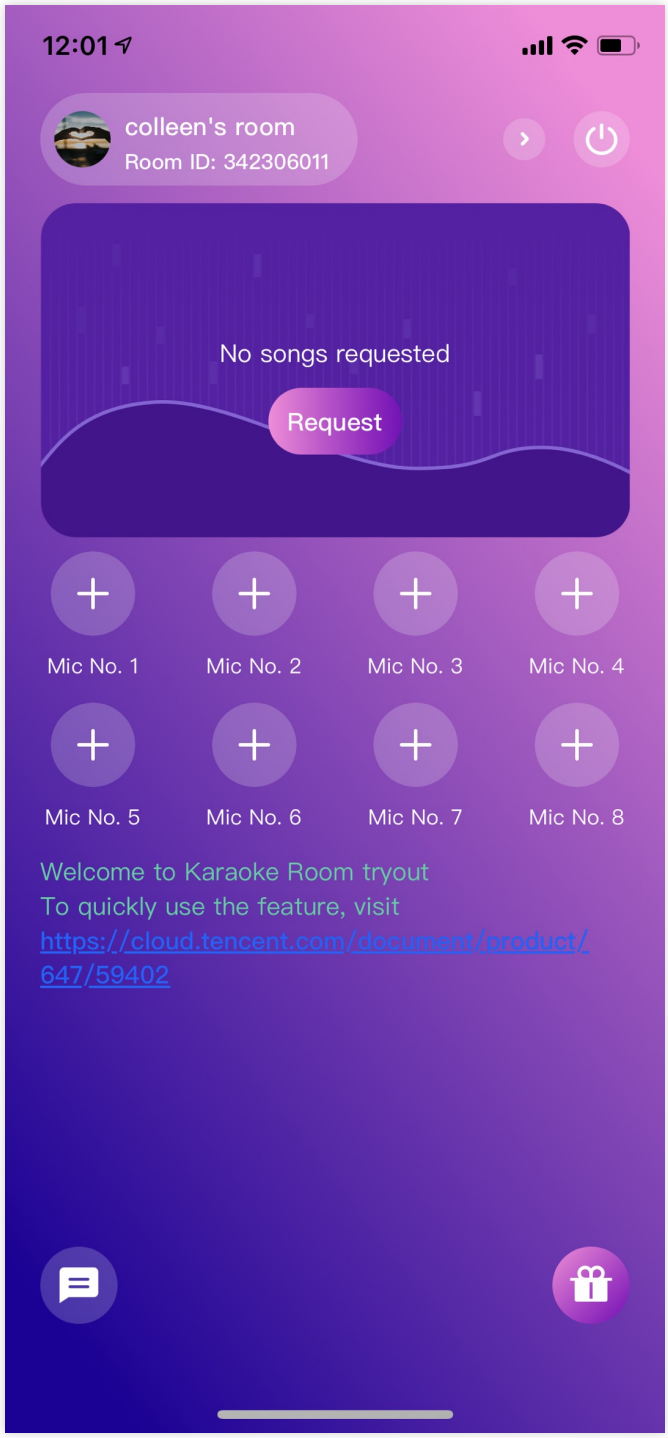
最近更新时间：2022-03-09 17:11:05

效果展示

您可以 下载 安装我们的 App 体验 KTV 的能力，包括低延时K歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。

房主麦位操作	听众麦位操作



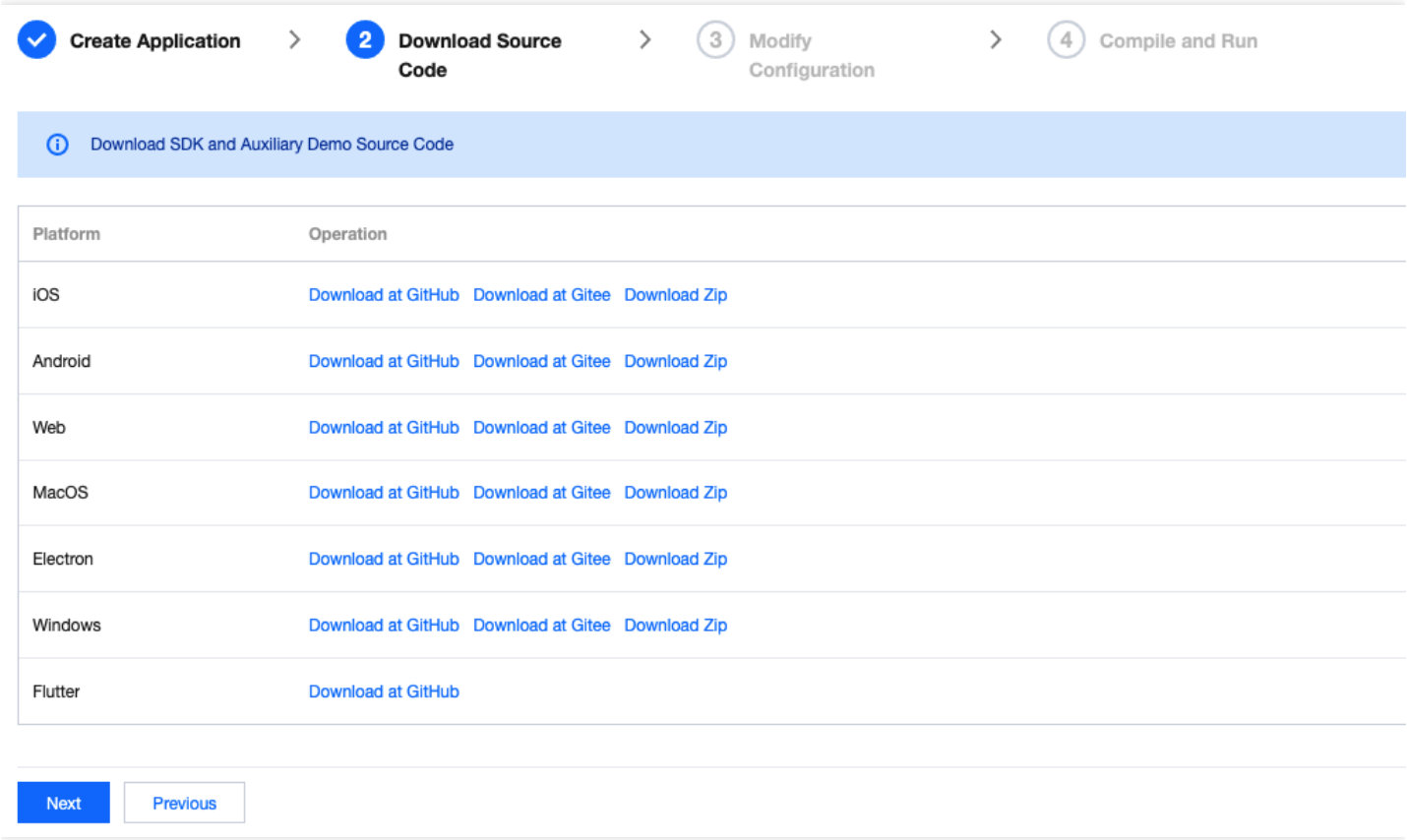


如需快速接入 KTV 功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TUIKaraoke 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestKaraoke` ，单击【创建】。
3. 单击【已下载，下一步】，跳过此步骤。



> 说明：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PAAS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

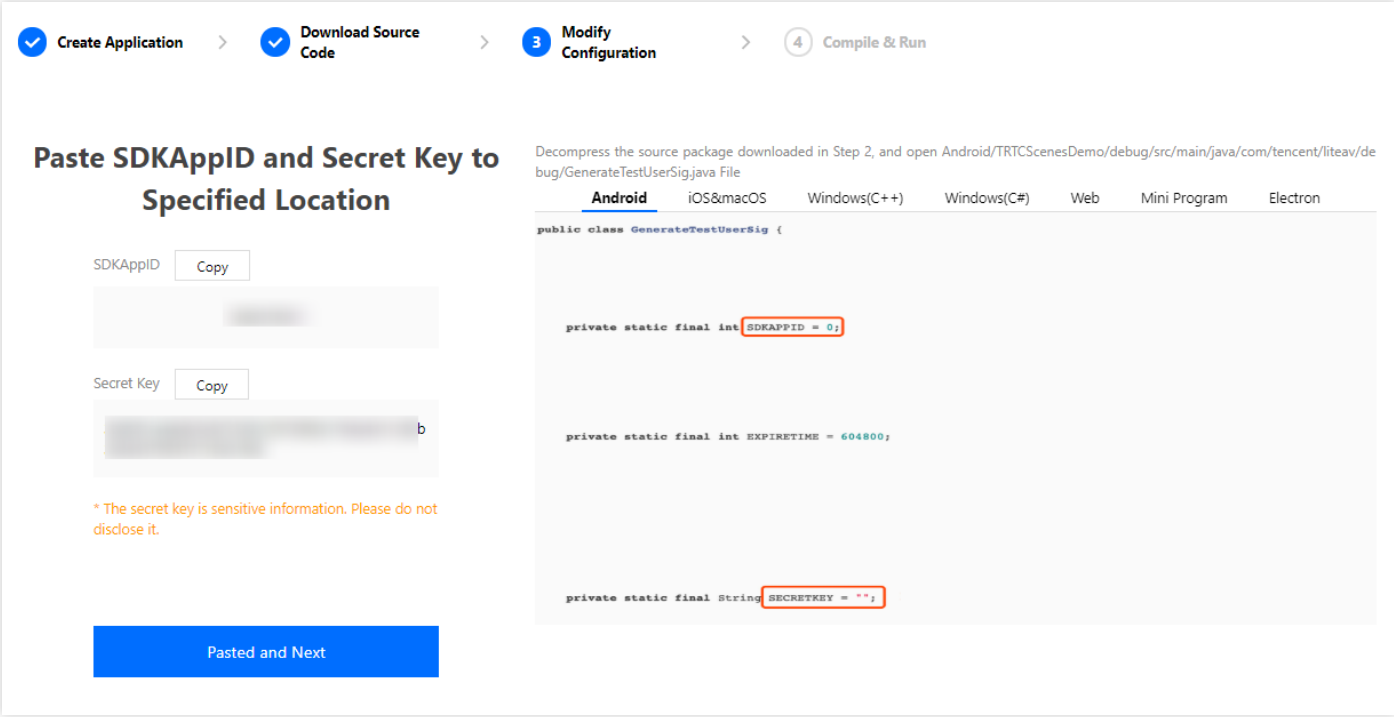
单击进入 TUIKaraoke，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 `TUIKaraoke/Debug/GenerateTestUserSig.swift` 文件。

3. 设置 `GenerateTestUserSig.swift` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。

5. 编译完成后，单击【回到控制台概览】即可。

> 注意：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Xcode（11.0及以上的版本）打开源码工程 `TUIKaraoke/TUIKaraokeApp.xcworkspace`，单击【运行】即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 `Source` 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码以及涉及界面相关的逻辑，如下表格列出了各个 swift 文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述
TRTCKaraokeEnteryControl.swift	该文件包含所有 ViewController 的初始化获取方法，您可以通过该实例，快速获取 ViewController 对象。
TRTCCreateKaraokeViewController	创建 KTV 页面逻辑。
TRTCKaraokeViewController	主房间页面，包括房主和听众两种界面。

每个 `TRTC'XXXX'ViewController` 文件夹下均包含 `ViewController`、 `RootView` 和 `ViewModel`，各个文件的作用如下表所示：

文件	功能描述
ViewController.swift	页面控制器，负责页面路由工作，以及 RootView 和 ViewModel 的绑定工作。
RootView.swift	视图，所有的视图布局。
ViewModel.swift	视图控制器，负责响应视图交互，返回视图响应状态。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

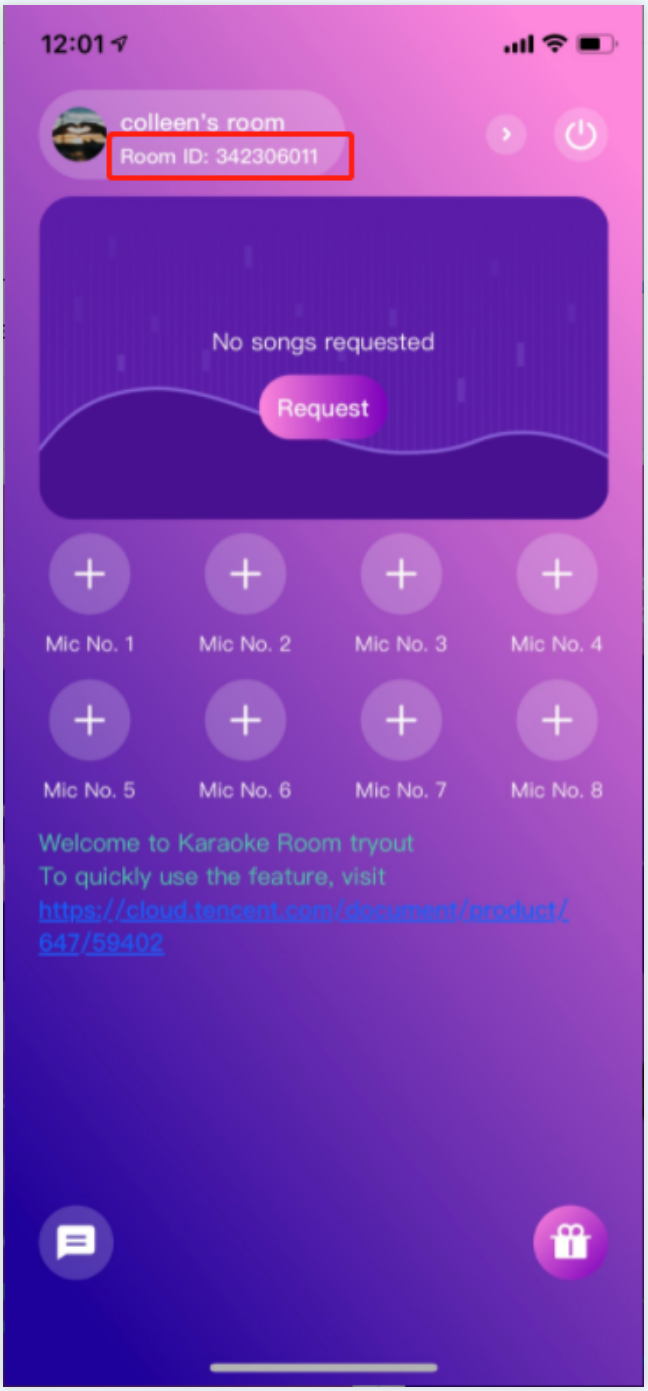
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 单击【创建房间】。
3. 输入房间主题，单击【一起嗨歌】。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入用户 A 创建的房间号，单击【进入房间】。

注意：

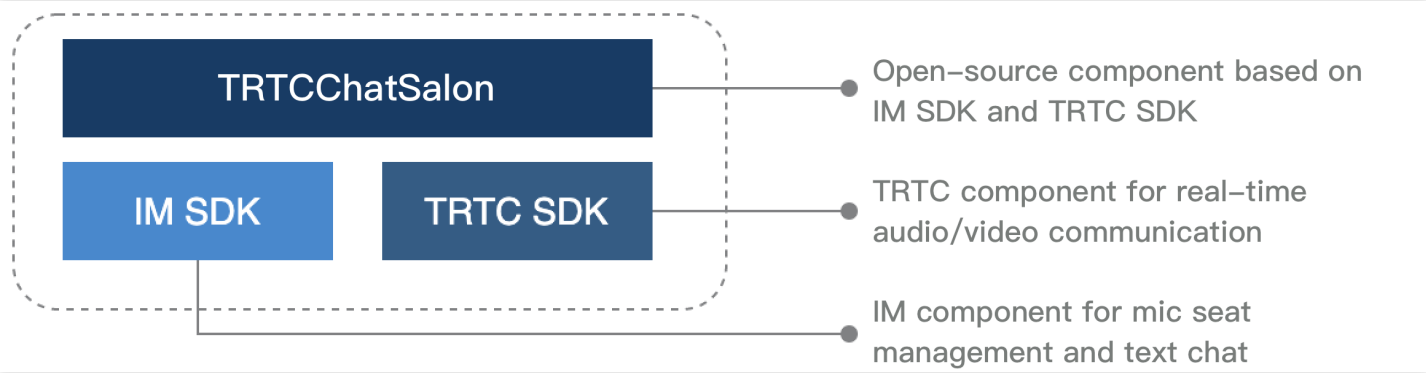
房间号在用户 A 的房间顶部查看，如下图示：



实现自定义 UI 界面

源码 中的 `Source` 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCKaraokeRoom，您可以在 `TRTCKaraokeRoom.h` 文件中看到该组件提供的接口函数，并使用对应接口实现自

定义 UI 界面。



步骤1：集成 SDK

KTV 组件 TRTCKaraokeRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

- **方法一：通过 cocoapods 仓库依赖**

```
pod 'TXIMSDK_iOS'
pod 'TXLiteAVSDK_TRTC'
```

- **方法二：通过本地依赖**
  如果您的开发环境访问 cocoapods 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档
IM SDK	DOWNLOAD	集成文档

步骤2：配置权限

在 info.plist 文件中需要添加 `Privacy > Microphone Usage Description` 申请麦克风权限。

步骤3：导入 TUIKaraoke 组件

通过 cocoapods 导入组件，具体操作如下：

1. 将工程目录下的 `Source` 、 `Resources` 、 `TXAppBasic` 文件夹、 `TUIKaraoke.podspec` 文件拷贝到您的工程目录下。
2. 在您的 `Podfile` 文件中添加以下依赖。之后执行 `pod install` 命令，完成导入。

```
pod 'TXAppBasic', :path =&gt; "TXAppBasic/"
pod 'TXLiteAVSDK_TRTC'
pod 'TUIKaraoke', :path =&gt; "./", :subspecs =&gt; ["TRTC"]
```

步骤4：创建并登录组件

1. 调用 TRTCKaraokeRoom 的 `sharedInstance` 类方法可以创建一个遵守 TRTCKaraokeRoom 协议的实例对象。也可以使用调用 `shared` 类方法，获取 TRTCKaraokeRoom 实例对象直接使用，二者在 TRTCKaraokeRoom 的接口使用上没有任何区别。
2. 调用 `setDelegate` 函数注册组件的事件回调通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

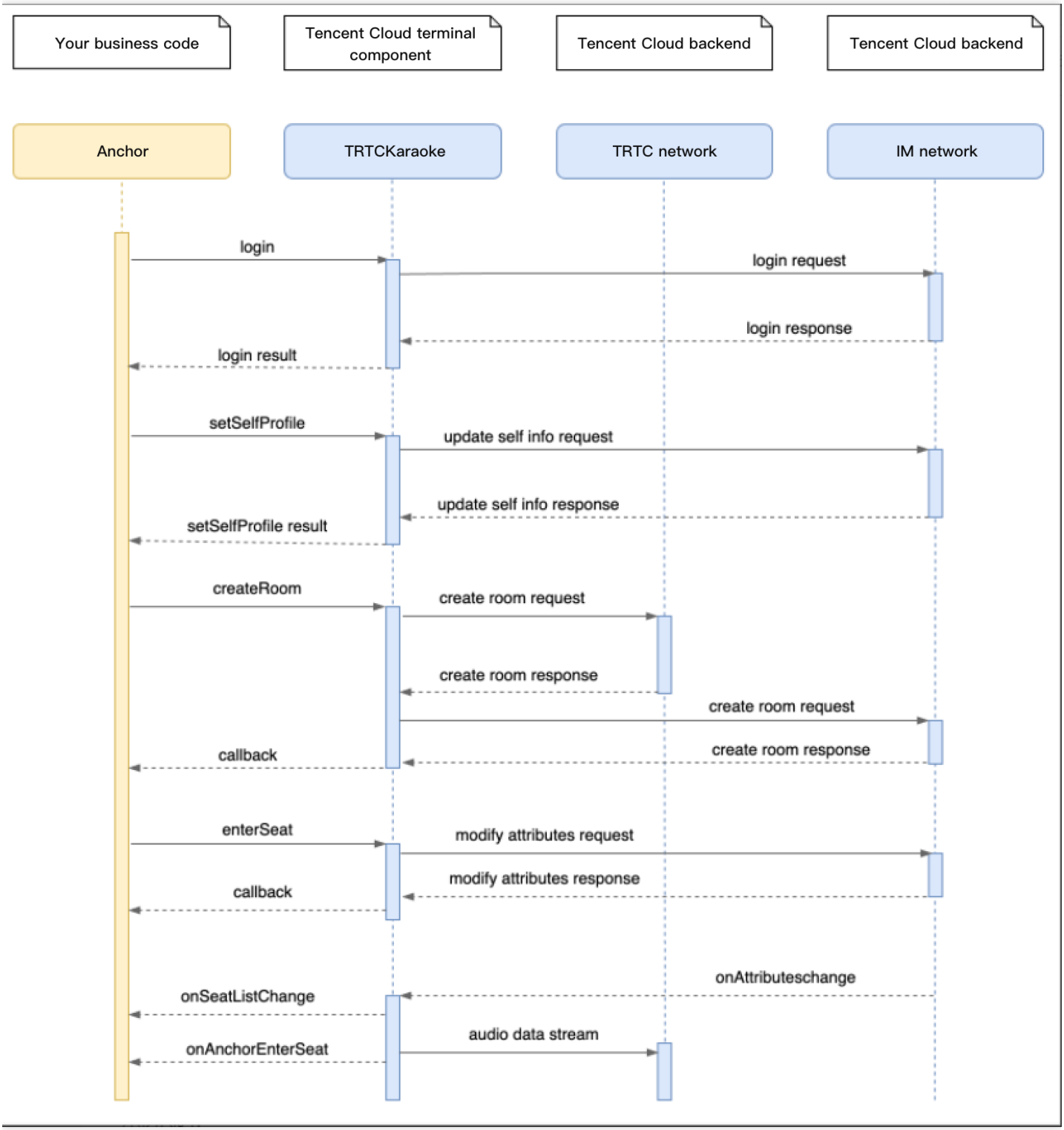
参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	登录回调，成功时 code 为0。

```
// Swift 示例
// 您代码里负责业务逻辑的类
class YourController {
// 计算属性获取单例对象
var karaokeRoom: TRTCKaraokeRoom {
return TRTCKaraokeRoom.shared()
}

// 其他代码逻辑
......
}
// 设置 KTV 代理
self.karaokeRoom.setDelegate(delegate: karaokeRoomDelegate)

// 调用方式如下,闭包内建议使用 weak self 防止循环引用（下面示例代码省略 weak self 示例）
self.karaokeRoom.login(sdkAppId: sdkAppID, userId: userId, userSig: userSig) { [weak self] (co
de, message) in
guard let `self` = self else { return }
// 您的回调业务逻辑
}
```

步骤5：房主创建房间

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的 KTV 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
3. 房主创建房间成功后，可以调用 `enterSeat` 进入座位。
4. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

版权所有：腾讯云计算（北京）有限责任公司



示例代码：

```
// 1.房主设置昵称和头像
self.karaokeRoom.setSelfProfile(userName: userName, avatarUrl: avatarURL) { (code, message) in
// 结果回调
}
```

```
// 2.房主端创建房间
let param = RoomParam.init()
param.roomName = "房间名称"
param.needRequest = true // 听众上麦是否需要房主同意
param.coverUrl = "封面URL"
param.seatCount = 8 // 房间座位数，这里一共8个座位
param.seatInfoList = []

for _ in 0..&lt; param.seatCount {
let seatInfo = SeatInfo.init()
param.seatInfoList.append(seatInfo)
}

self.karaokeRoom.createRoom(roomID: yourRoomID, roomParam: param) { (code, message) in
guard code == 0 else { reutrn }
// 创建成功

// 3.上麦
self.karaokeRoom.enterSeat(seatIndex: 0) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
// 房主占座成功
} else {
// 房主占座失败
}
}
}

// 4.上麦成功后，收到 onSeatListChange 事件通知
func onSeatListChange(seatInfoList: [SeatInfo]) {
// 刷新您的麦位列表
}

// 5. 收到 onAnchorEnterSeat 事件通知
func onAnchorEnterSeat(index: Int, user: UserInfo) {
// 处理房主上麦事件
}
```

步骤6：听众端观看

1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的 KTV 房间列表。

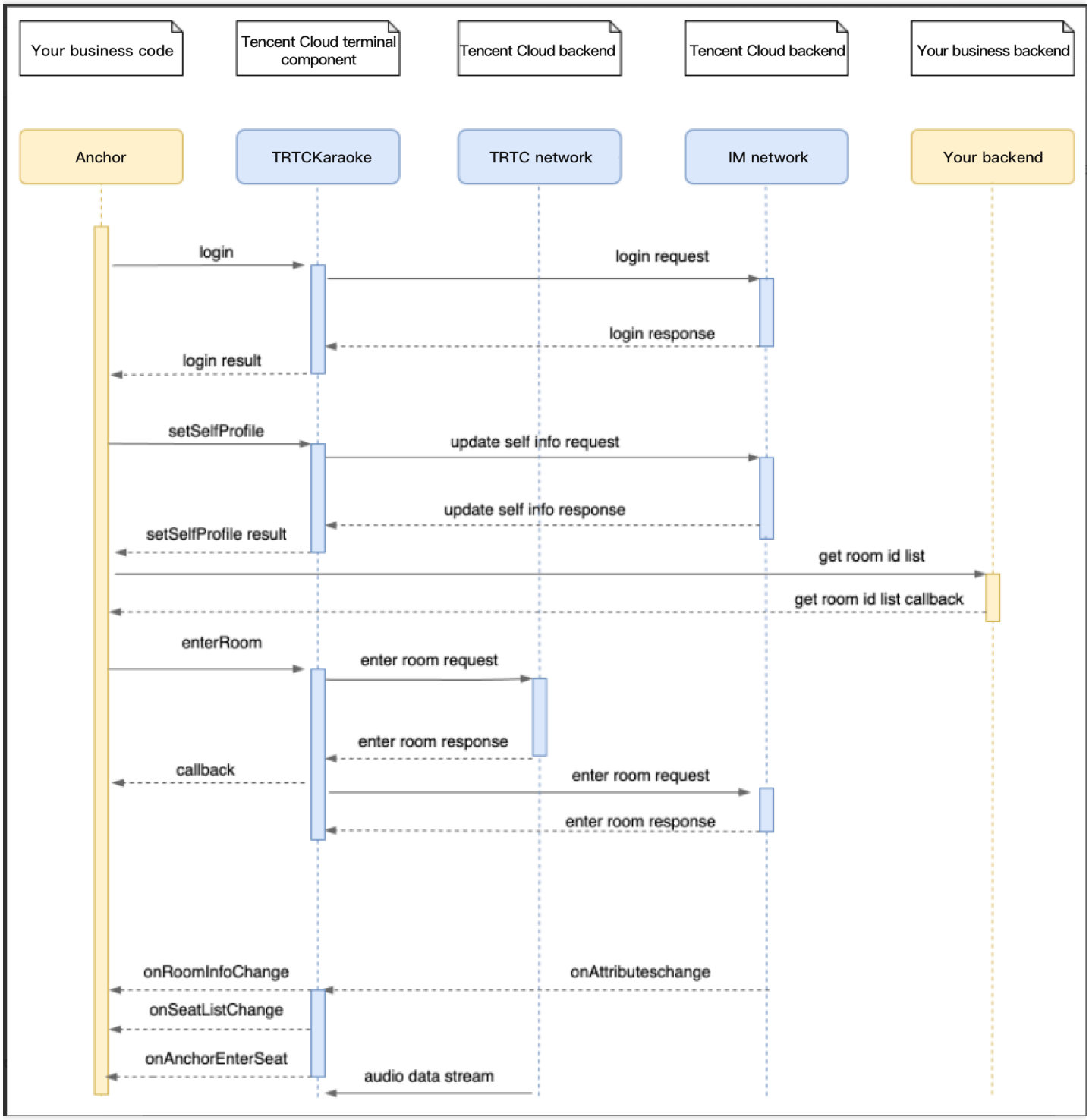
> 说明：

> App 中的 KTV 房间列表仅做演示使用，KTV 房间列表的业务逻辑千差万别，腾讯云暂不提供 KTV 房间列表的管理服务，请自行管理您的 KTV 房间列表。

3. 听众端调用 `getRoomInfoList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建 KTV 房间时设置的简单描述信息。

> 注意：
> 如果您的 KTV 房间列表包含了足够全面的信息，可跳过调用 `getRoomInfoList` 相关步骤。

4. 听众选择一个 KTV 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
6. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
7. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。



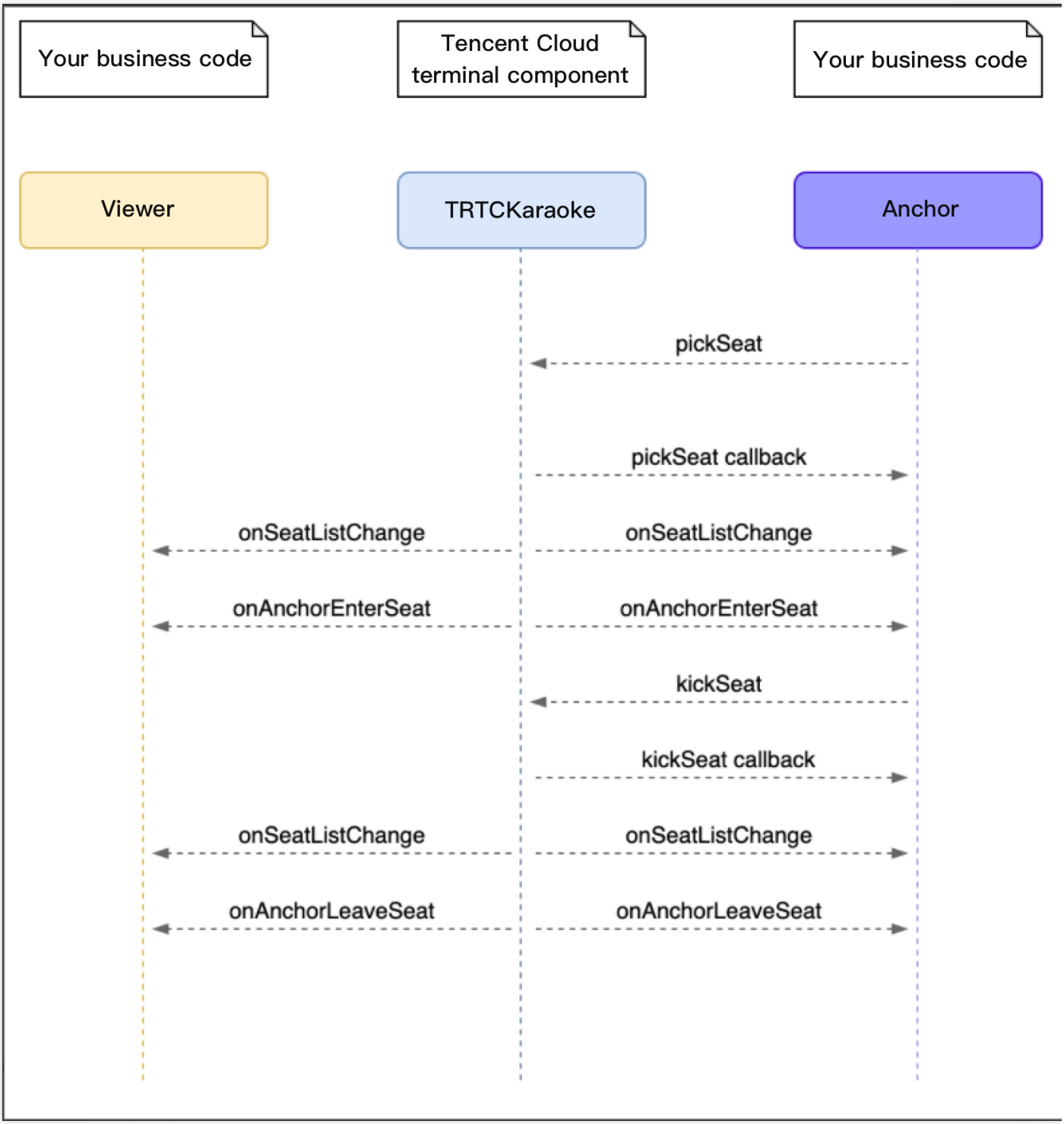

```
// 1.听众设置昵称和头像
self.karaokeRoom.setSelfProfile(userName: userName, avatarUrl: avatarURL) { (code, message) in
// 结果回调
}

// 2.假定您从业务后台获取房间列表为 roomList
let roomList: [Int] = getRoomIDList() // 您获取房间ID列表的函数
```

```
// 3.通过调用 getRoomInfoList 获取房间的详细信息
self.karaokeRoom.getRoomInfoList(roomIdList: roomIdsInt) { (code, message, roomInfos: [RoomInfo])
in
// 获取结果，此时可以刷新UI
}

// 4.选择 KTV 后，传入 roomId 进入房间
self.karaokeRoom.enterRoom(roomID: roomInfo.roomID) { (code, message) in
// 进入房间结果回调
if code == 0 {
// 进房成功
}
}

// 5.进房成功后，收到 onRoomInfoChange 事件通知
func onRoomInfoChange(roomInfo: RoomInfo) {
// 可以更新房间名称等信息
}

// 6.进房成功后，收到 onSeatListChange 事件通知
func onSeatListChange(seatInfoList: [SeatInfo]) {
// 刷新麦位列表
}

// 7. 收到 onAnchorEnterSeat 事件通知
func onAnchorEnterSeat(index: Int, user: UserInfo) {
// 处理上麦事件
}
```

步骤7：麦位管理

- 房主端
- 听众端

1. `pickSeat` 传入对应的麦位和听众 userId，可以抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。
2. `kickSeat` 传入对应麦位后，可以踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。
3. `muteSeat` 传入对应麦位后，可以静音/解除静音，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。
4. `closeSeat` 传入对应麦位后，可以封禁/解禁某个麦位，封禁后听众端将不能再上麦，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。



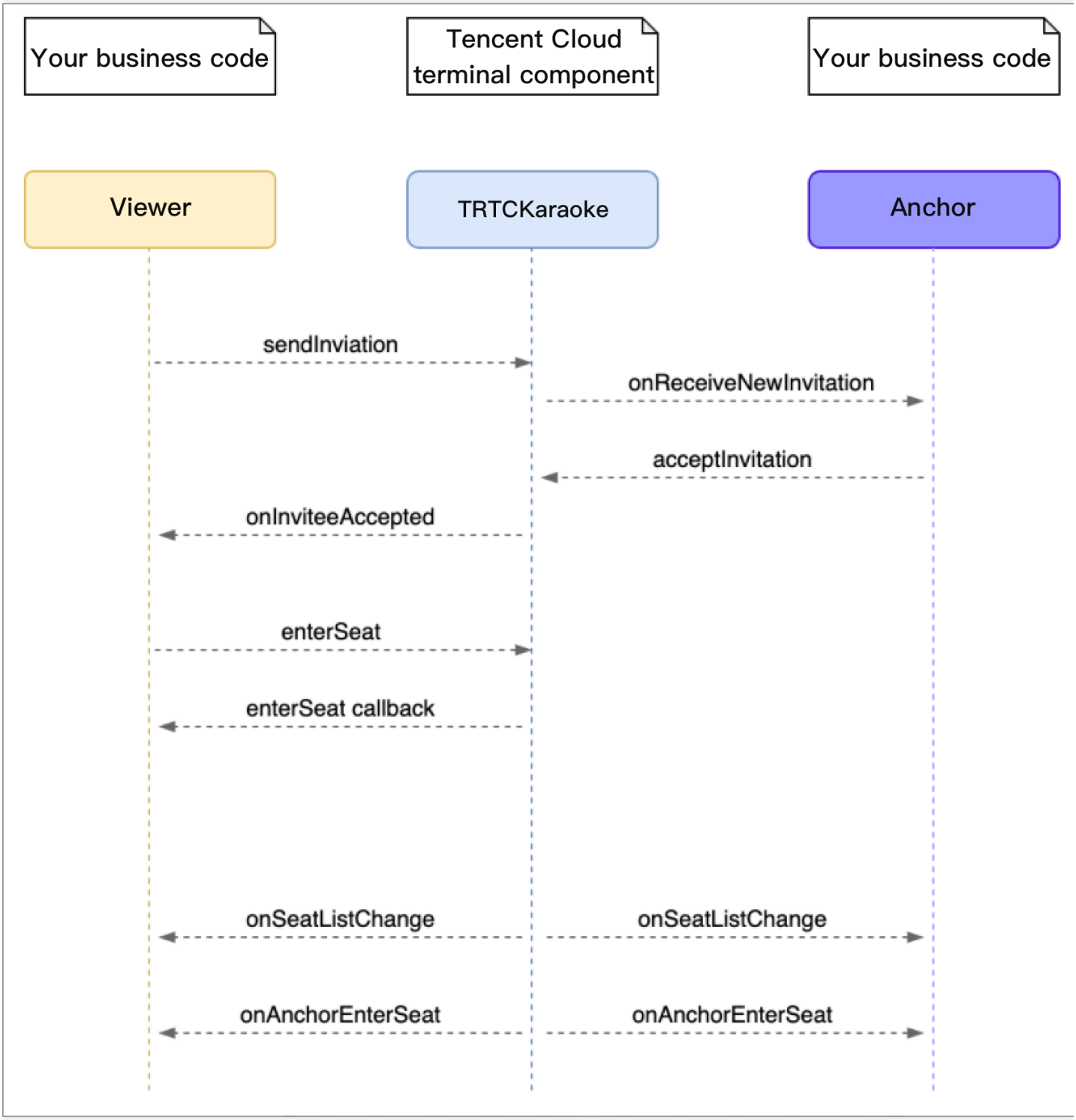
步骤8：邀请信令的使用

在 麦位管理 中，听众上下麦、房主抱人上麦都不需要经过对方的同意就可以直接操作。

如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

- 听众主动申请上麦
- 房主邀请听众上麦

1. 听众端调用 `sendInvitation` 传入房主的 userId 和业务的自定义命令字等，此时函数会返回一个 inviteId，记录该 inviteId。
2. 房主端收到 `onReceiveNewInvitation` 的事件通知，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `acceptInvitation` 并传入 inviteId。
4. 听众端收到 `onInviteeAccepted` 的事件通知，调用 `enterSeat` 进行上麦。



```
// 听众端视角
// 1.调用 sendInvitation，请求上1号麦位
let inviteId = self.karaokeRoom.sendInvitation(cmd: "ENTER_SEAT", userId: ownerUserId, content:
"1") { (code, message) in
// 发送结果回调
}
// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
self.karaokeRoom.enterSeat(seatIndex: ) { (code, message) in
// 上麦结果回调
}
}
}

// 房主端视角
// 1.房主收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
if cmd == "ENTER_SEAT" {
// 2.房主同意听众请求
self.karaokeRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

  ```
  // 发送端：发送文本消息
  self.karaokeRoom.sendRoomTextMsg(message: message) { (code, message) in


  }
  // 接收端：监听文本消息
  func onRecvRoomTextMsg(message: String, userInfo: UserInfo) {
  //收到的message信息处理方法
  }
  ```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomCustomMsg` 回调。

  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息, "CMD_LIKE"表示点赞消息
self.karaokeRoom.sendRoomCustomMsg(cmd: "CMD_DANMU", message: "hello world", callback: nil)
self.karaokeRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)
// 接收端：监听自定义消息
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: UserInfo) {
if cmd == "CMD_DANMU" {
// 收到弹幕消息
}
if cmd == "CMD_LIKE" {
// 收到点赞消息
}
}
```

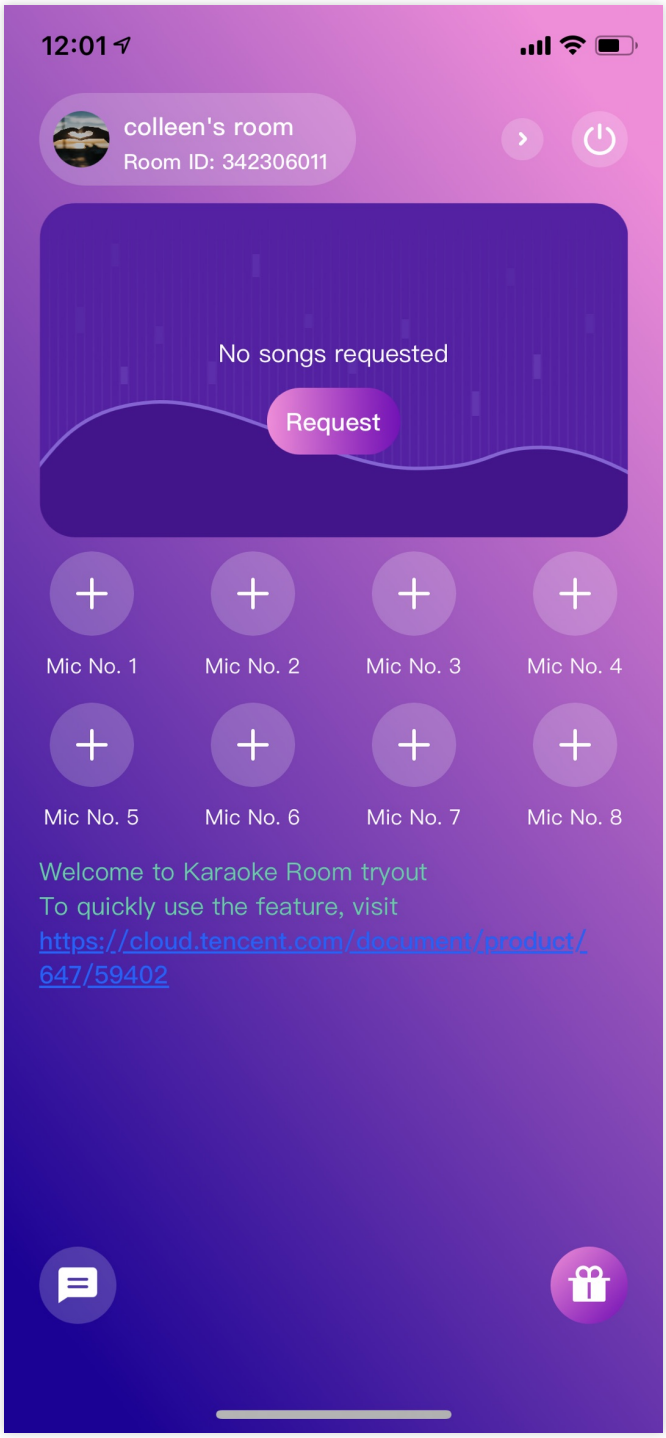
在线 K 歌_KTV（Android）

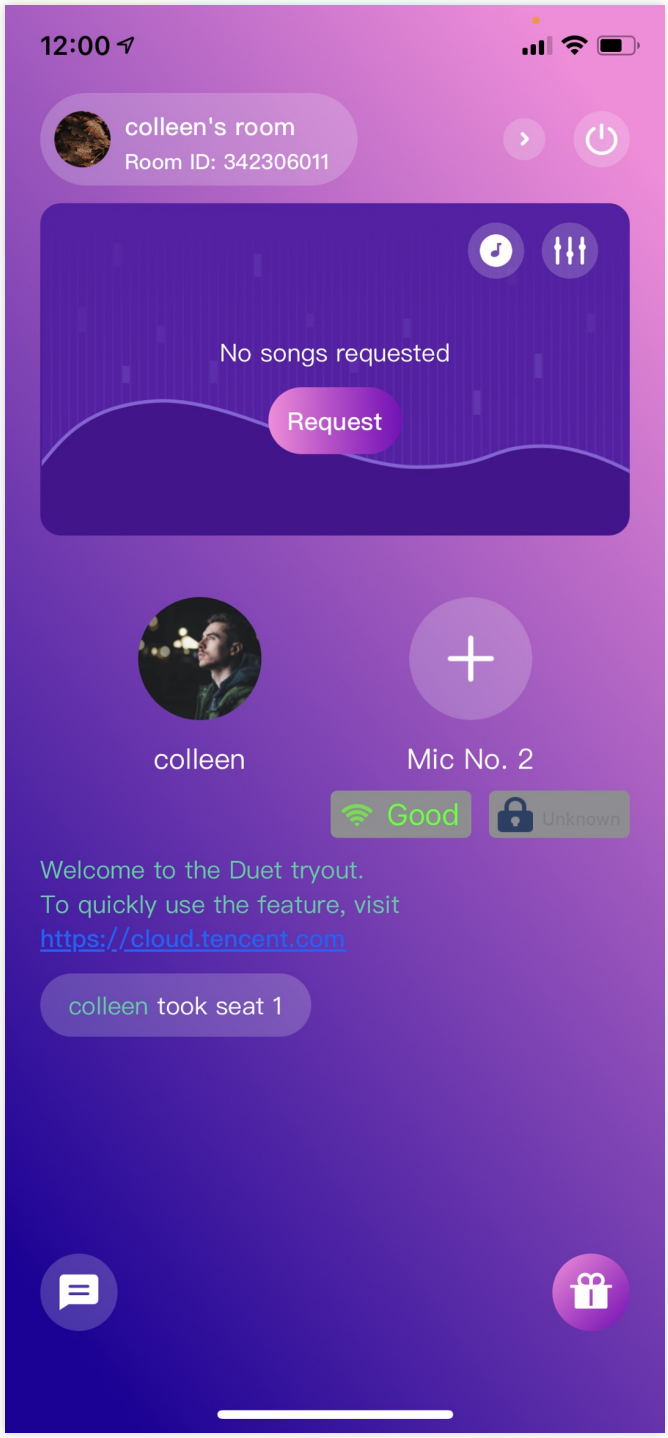
最近更新时间：2022-03-09 17:18:32

效果展示

您可以 下载 安装我们的 App 体验 KTV 的能力，包括低延时K歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。

房主麦位操作	听众麦位操作



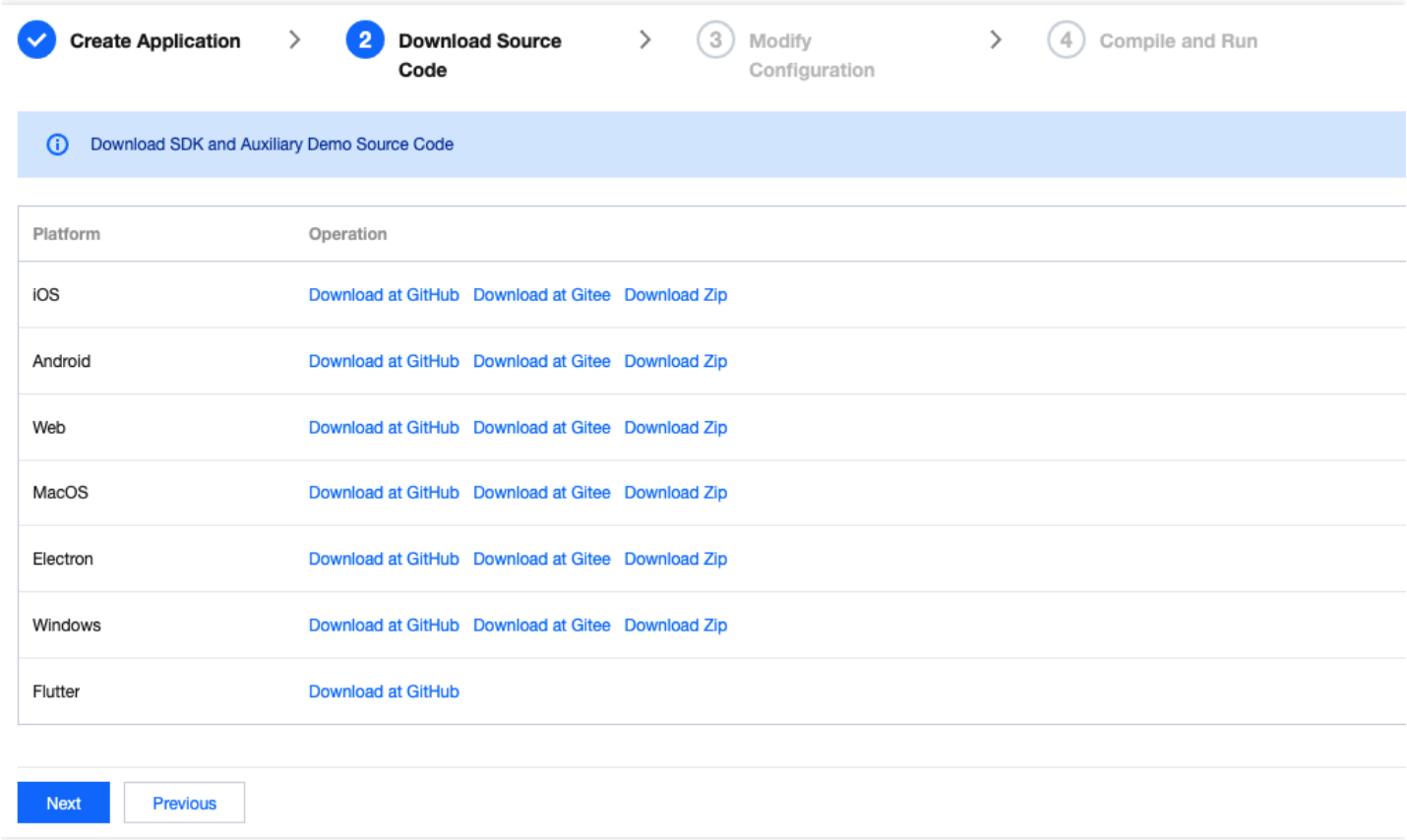


如需快速接入 KTV 功能，您可以直接基于我们提供的 App 进行修改适配，也可以使用我们提供的 TUIKaraoke 组件并实现自定义 UI 界面。

复用 App 的 UI 界面

步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 `TestKaraoke` ，单击【创建】。
3. 单击【已下载，下一步】，跳过此步骤。



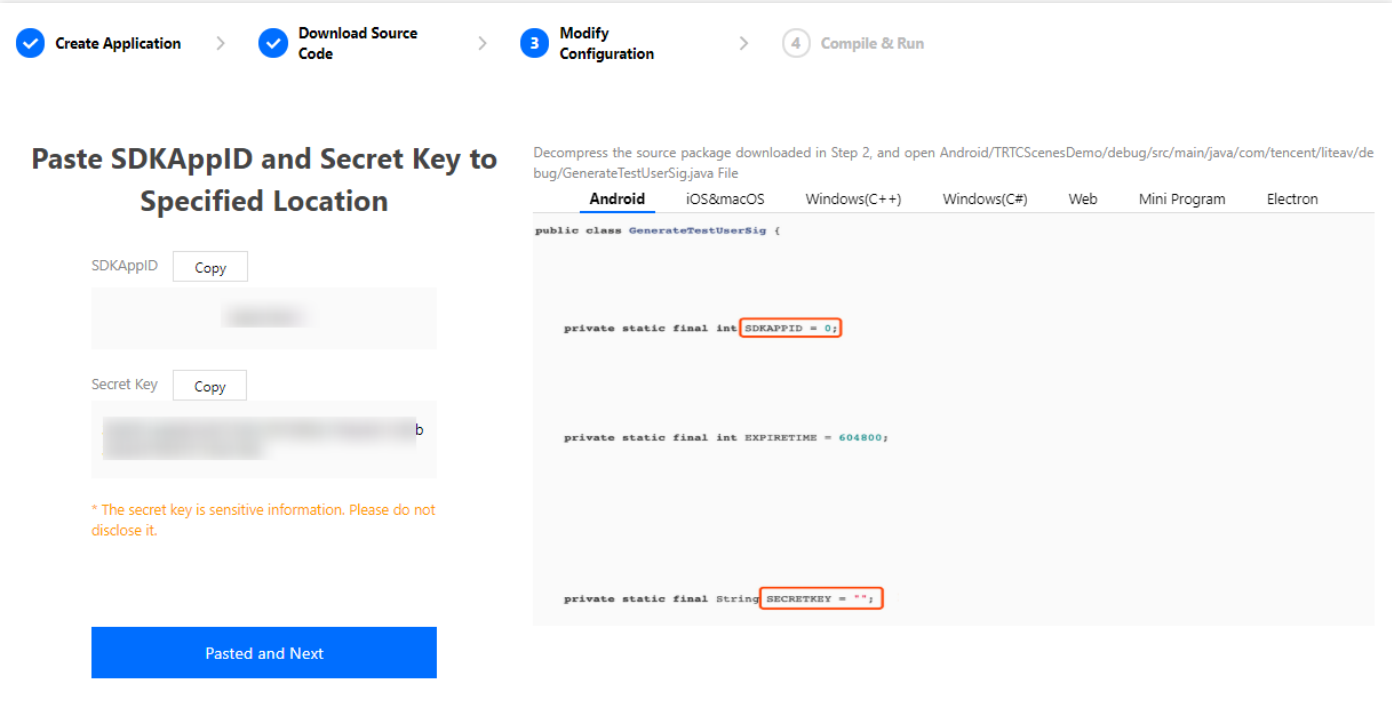
> 注意：
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。 即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明。

步骤2：下载 App 源码

单击进入 TUIKaraoke，Clone 或者下载源码。

步骤3：配置 App 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `Android/Debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` 文件。
3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为占位符（PLACEHOLDER），请设置为实际的 SDKAppID。
   - SECRETKEY：默认为占位符（PLACEHOLDER），请设置为实际的密钥信息。



4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 App 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤4：运行 App

使用 Android Studio（3.5以上的版本）打开源码工程 `TUIKaraoke` ，单击【运行】即可开始调试本 App。

步骤5：修改 App 源代码

源码中的 Source 文件夹包含两个子文件夹 ui 和 model，ui 文件夹中均为界面代码，如下表格列出了各个文件或文件夹及其所对应的 UI 界面，以便于您进行二次调整：

文件或文件夹	功能描述

文件或文件夹	功能描述
base	UI 使用的基础类。
room	主房间页面，包括房主和听众两种界面。
widget	通用控件。

体验应用

> 注意：
> 体验应用至少需要两台设备。

用户 A

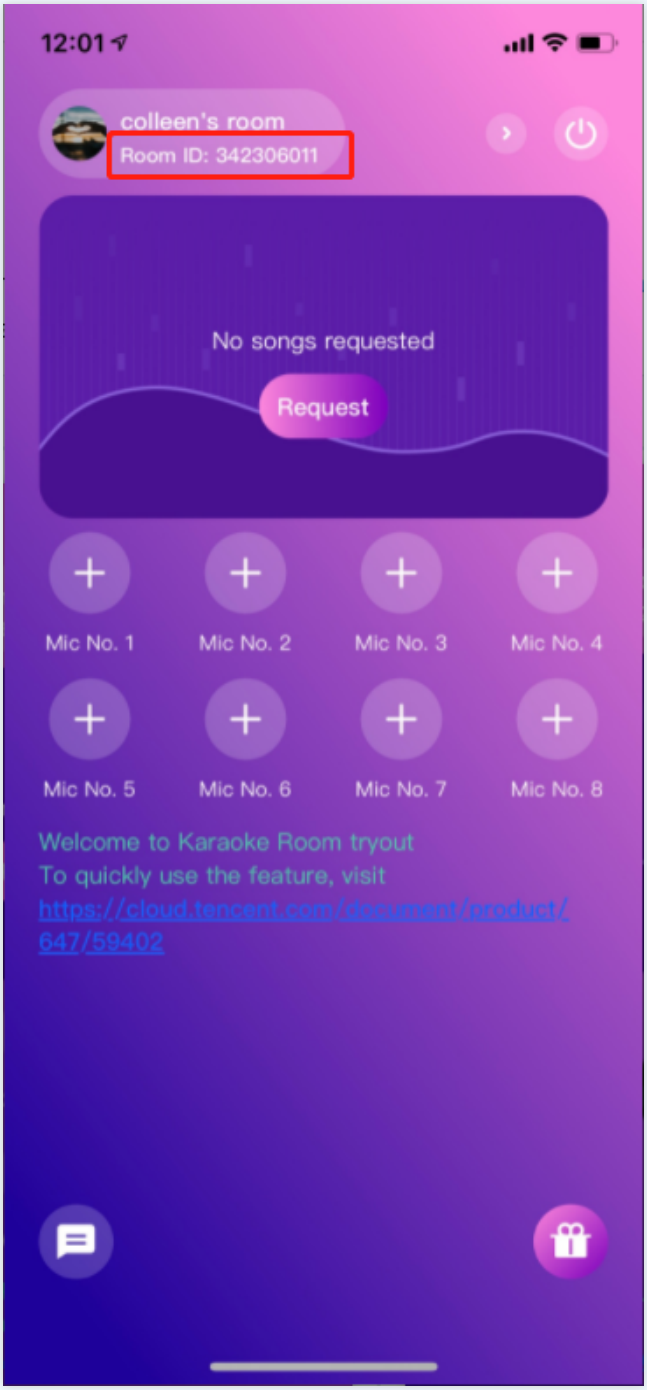
1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 单击【创建房间】。
3. 输入房间主题，单击【一起嗨歌】。

用户 B

1. 输入用户名（**请确保用户名唯一性，不能与其他用户重复**）并登录。
2. 输入用户 A 创建的房间号，单击【进入房间】。

> 注意：

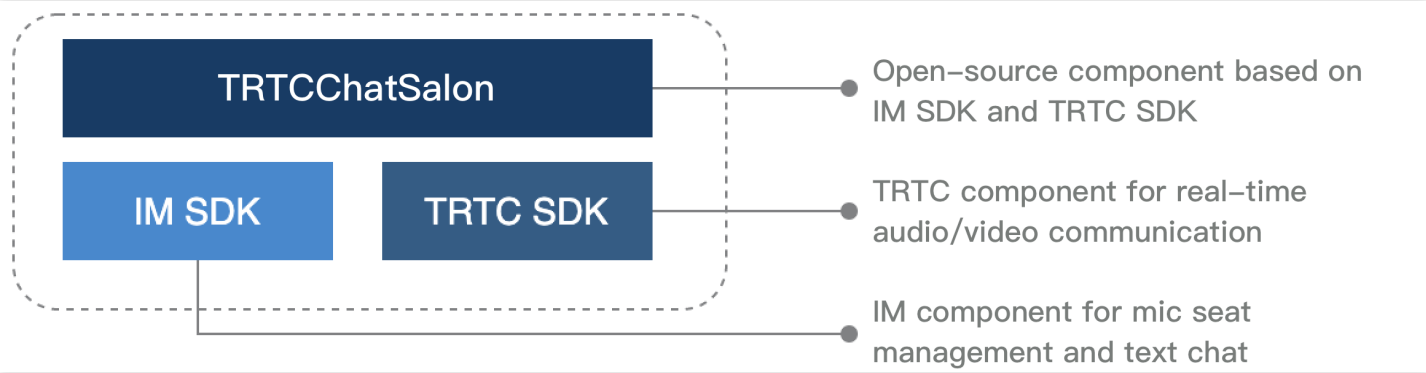
房间号在用户 A 的房间顶部查看，如下图示：



实现自定义 UI 界面

源码 中的 Source 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCKaraokeRoom，您可以在 `TRTCKaraokeRoom.java` 文件中看到该组件提供的接口函数，并使用对应接口实现

自定义 UI 界面。



步骤1：集成 SDK

KTV 组件 TRTCKaraokeRoom 依赖 TRTC SDK 和 IM SDK，您可以按照如下步骤将两个 SDK 集成到项目中。

方法一：通过 Maven 仓库依赖

1. 在 dependencies 中添加 TRTCSDK 和 IMSDK 的依赖。

```
dependencies {
complie "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
complie 'com.tencent.imsdk:imsdk:latest.release'
compile 'com.google.code.gson:gson:2.3.1'
}
```

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

```
defaultConfig {
ndk {
abiFilters "armeabi-v7a"
}
}
```

3. 单击【Sync Now】，自动下载 SDK 并集成到工程里。

方法二：通过本地 AAR 依赖

若您的开发环境访问 Maven 仓库较慢，您可以直接下载 ZIP 包，并按照集成文档手动集成到您的工程中。

SDK	下载页面	集成指引
TRTC SDK	DOWNLOAD	集成文档

IM SDK	DOWNLOAD	集成文档

步骤2：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请读取存储权限）：

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

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

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

步骤3：导入 TRTCKaraoke 组件

拷贝以下目录中的所有文件到您的项目中：

```
Source/src/main/java/com/tencent/liteav/tuikaraoke/model
```

步骤4：创建并登录组件

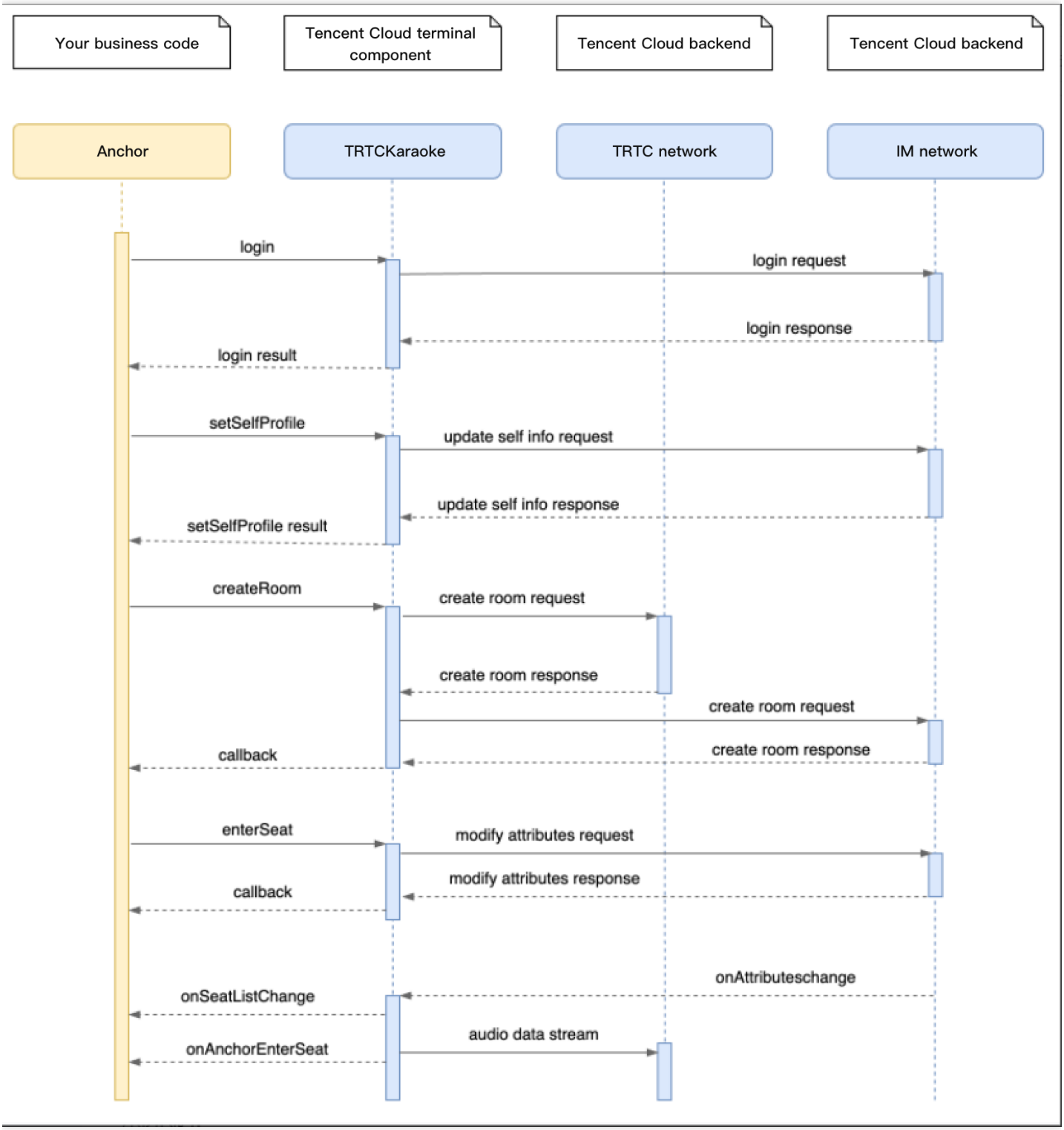
1. 调用 `sharedInstance` 接口可以创建一个 TRTCKaraoke 组件的实例对象。
2. 调用 `setDelegate` 函数注册组件的事件通知。
3. 调用 `login` 函数完成组件的登录，请参考下表填写关键参数：

参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	登录回调，成功时 code 为0。

```
TRTCKaraoke mTRTCKaraokeRoom = TRTCKaraokeRoom.sharedInstance(this);
mTRTCKaraokeRoom.setDelegate(this);
mTRTCKaraokeRoom.login(SDKAPPID, userId, userSig, new TRTCKaraokeRoomCallback.ActionCallback()
{
@Override
public void onCallback(int code, String msg) {
if (code == 0) {![](https://main.qcloudimg.com/raw/7e61af0f3de8302260a1e10d7da6231e.png)
//登录成功
}
}
});
```

步骤5：房主端开播

1. 房主执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 房主调用 `createRoom` 创建新的 KTV 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
3. 房主创建房间成功后，可以调用 `enterSeat` 进入座位。
4. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。




```
// 1.房主设置昵称和头像
mTRTCKaraokeRoom.setSelfProfile("my_name", "my_face_url", null);

// 2.房主调用 createRoom 创建房间
final TRTCKaraokeRoomDef.RoomParam roomParam = new TRTCKaraokeRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
```

```
roomParam.seatCount = 7; // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.coverUrl = "房间封面图的 URL 地址";
mTRTCKaraokeRoom.createRoom(mRoomId, roomParam, new TRTCKaraokeRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//3. 占0号座位
mTRTCKaraokeRoom.enterSeat(0, new TRTCKaraokeRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
}
}
});
}
}
});

// 4.占座成功后，收到 onSeatListChange 事件通知
@Override
public void onSeatListChange(final List<trtckaraokeroomdef.seatinfo> seatInfoList) {
// 展示您的麦位列表
}

// 5. 收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCKaraokeRoomDef.UserInfo userInfo) {
}
```

步骤6：听众端观看

1. 听众端执行 步骤4 登录后，可以调用 `setSelfProfile` 设置自己的昵称和头像。
2. 听众端向业务后台获取最新的 KTV 房间列表。

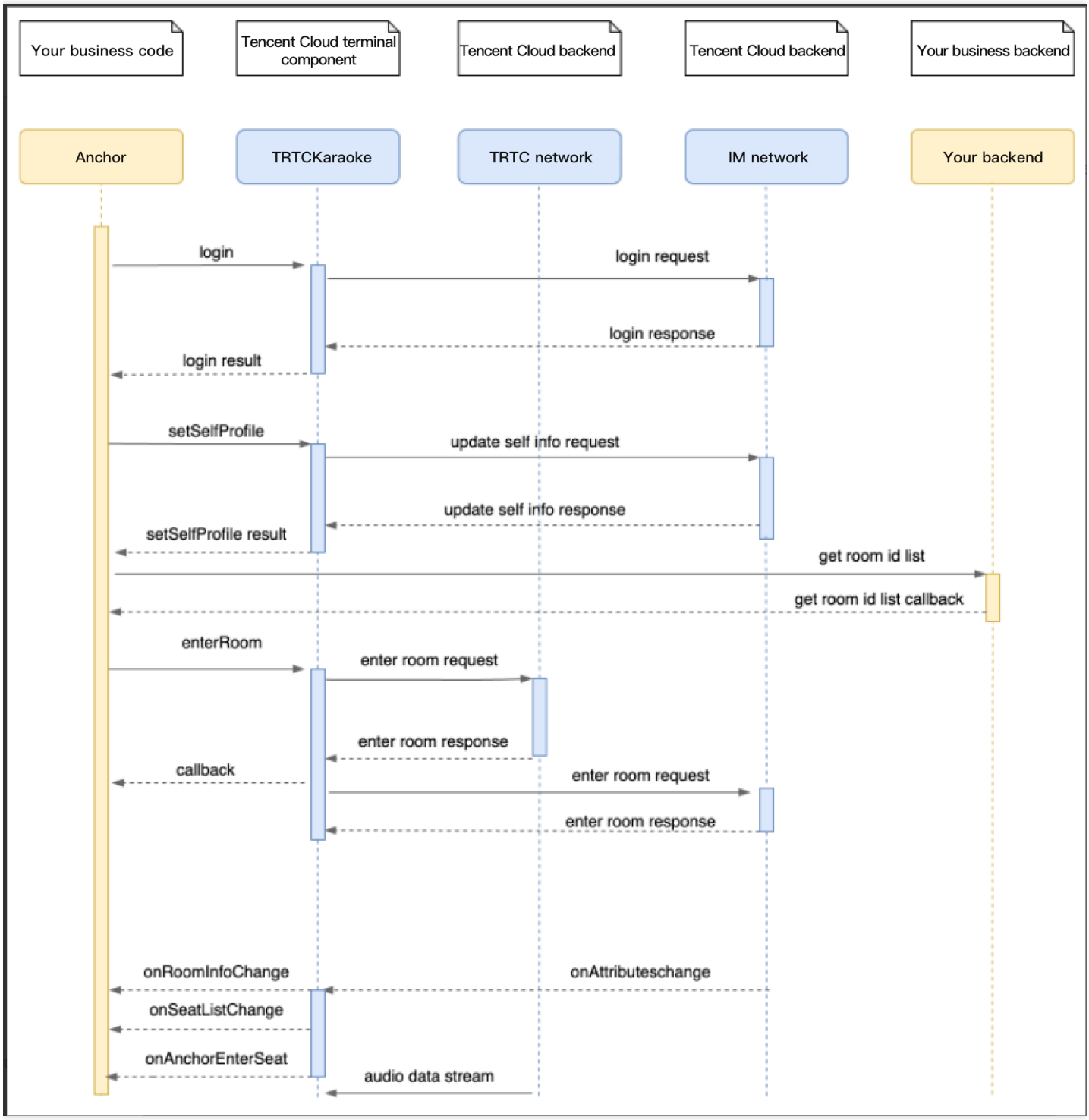
> 说明：
> App 中的 KTV 房间列表仅做演示使用， KTV 房间列表的业务逻辑千差万别，腾讯云暂不提供 KTV 房间列表的管理服务，请自行管理您的 KTV 房间列表。

3. 听众端调用 `getRoomInfoList` 获取房间的详细信息，该信息是在房主端调用 `createRoom` 创建 KTV 房间时设置的简单描述信息。

> 注意：

> 如果您的 KTV 房间列表包含了足够全面的信息，可跳过调用 `getRoomInfoList` 相关步骤。

4. 听众选择一个 KTV 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
5. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
6. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
7. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。




```
// 1.听众设置昵称和头像
mTRTCKaraokeRoom.setSelfProfile("my_name", "my_face_url", null);

// 2.假定您从业务后台获取房间列表为 roomList
List<integer> roomList = GetRoomList();

// 3.通过调用 getRoomInfoList 获取房间的详细信息
mTRTCKaraokeRoom.getRoomInfoList(roomList, new TRTCKaraokeRoomCallback.RoomInfoCallback() {
```

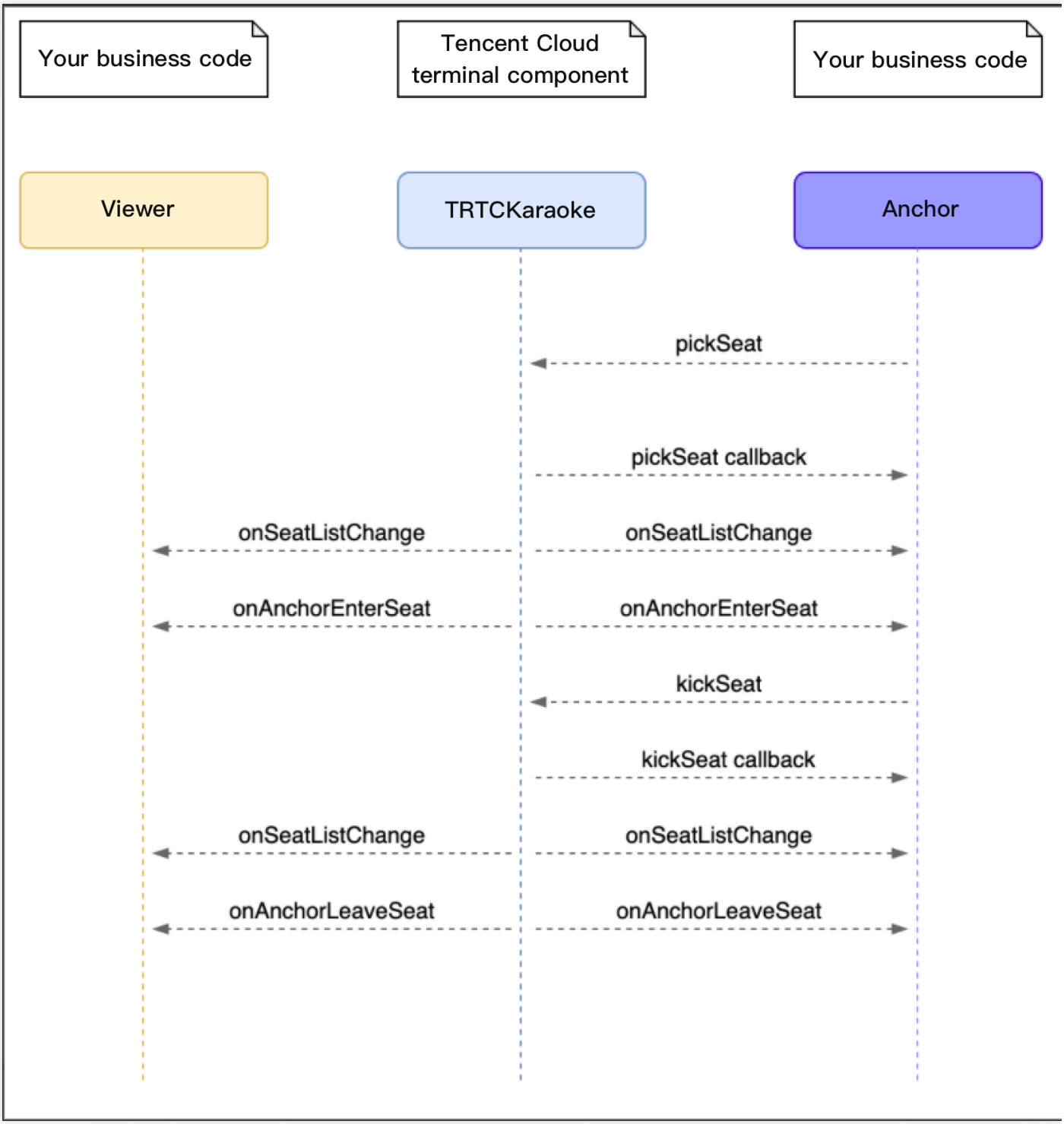
```
@Override
public void onCallback(int code, String msg, List<trtckaraokeroomdef.roominfo> list) {
if (code == 0) {
// 此时可以刷新您自己的 UI 房间列表
}
}
});

// 4.选择 KTV 后，传入 roomId 进入房间
mTRTCKaraokeRoom.enterRoom(roomId, new TRTCKaraokeRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
}
}
});

// 5.进房成功后，收到 onRoomInfoChange 事件通知
@Override
public void onRoomInfoChange(TRTCKaraokeRoomDef.RoomInfo roomInfo) {
mNeedRequest = roomInfo.needRequest;
mRoomName = roomInfo.roomName;
// UI 可以展示标题等
}

// 6.进房成功后，收到 onSeatListChange 事件通知
@Override
public void onSeatListChange(final List<trtckaraokeroomdef.seatinfo> seatInfoList) {
// 展示您的麦位列表
}

// 7. 收到 onAnchorEnterSeat 事件通知
@Override
public void onAnchorEnterSeat(TRTCKaraokeRoomDef.UserInfo userInfo) {
}
```

步骤7：麦位管理

- 房主端
- 听众端

1. `pickSeat` 传入对应的麦位和听众 userId，可以抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。
2. `kickSeat` 传入对应麦位后，可以踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

3. `muteSeat` 传入对应麦位后，可以静音/解除静音，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。
4. `closeSeat` 传入对应麦位后，可以封禁/解禁某个麦位，封禁后听众端将不能再上麦，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

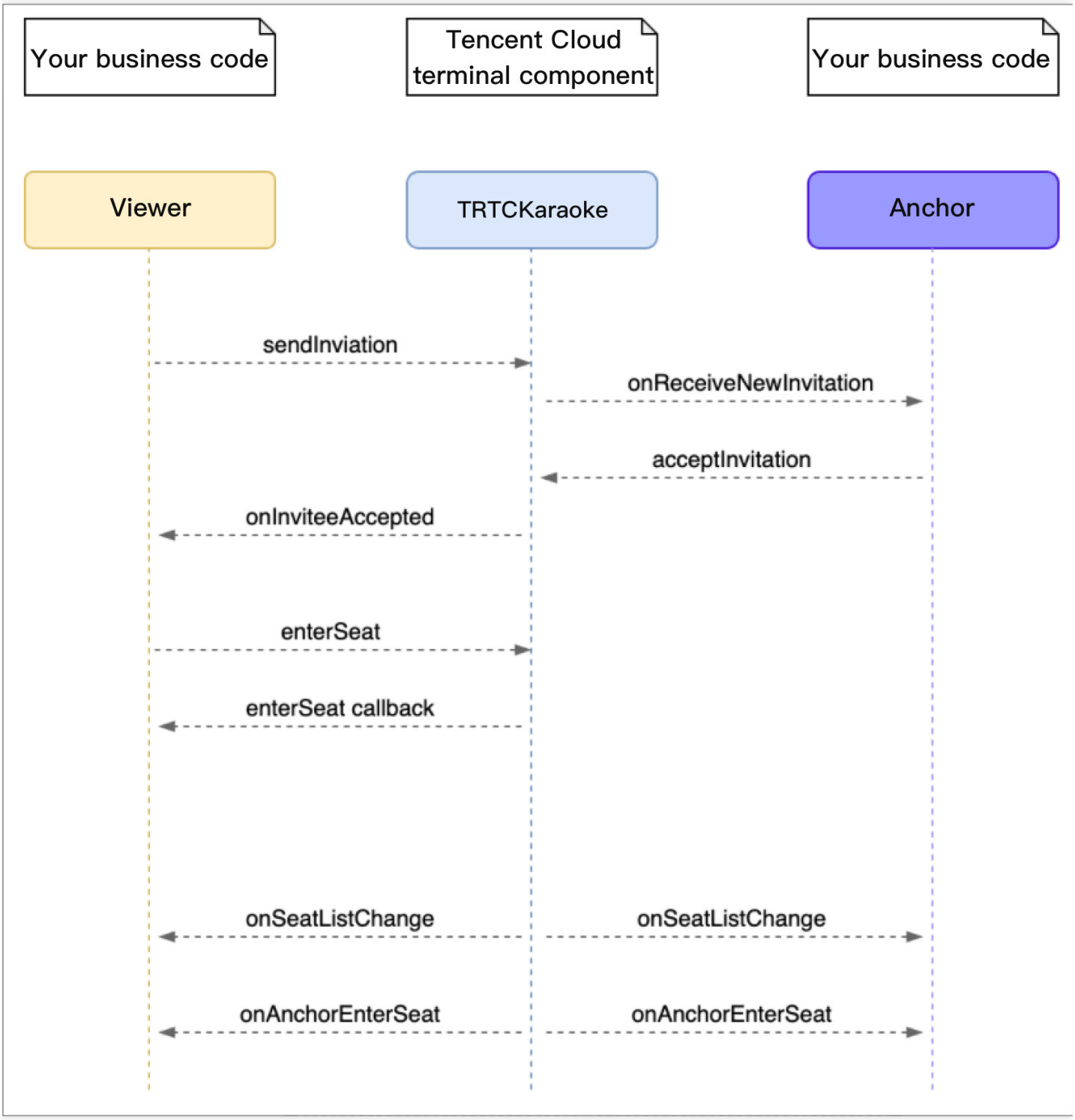


步骤8：邀请信令的使用

在 麦位管理 中，听众上下麦、房主抱人上麦都不需要经过对方的同意就可以直接操作。
如果您的 App 需要对方同意才能进行下一步操作的业务流程，那么邀请信令可以提供相应支持。

- 听众主动申请上麦
- 房主邀请听众上麦

1. 听众端调用 `sendInvitation` 传入房主的 userId 和业务的自定义命令字等，此时函数会返回一个 inviteId，记录该 inviteId。
2. 房主端收到 `onReceiveNewInvitation` 的事件通知，此时 UI 可以弹窗并询问房主是否同意。
3. 房主选择同意后，调用 `acceptInvitation` 并传入 inviteId。
4. 听众端收到 `onInviteeAccepted` 的事件通知，调用 `enterSeat` 进行上麦。




```
// 听众端视角
// 1.调用 sendInvitation, 请求上1号麦位
String inviteId = mTRTCKaraokeRoom.sendInvitation("ENTER_SEAT", ownerUserId, "1", null);

// 2.收到邀请的同意请求, 正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
```

```
if(id.equals(inviteId)) {
mTRTCKaraokeRoom.enterSeat(1, null);
}
}

// 房主端视角
// 1.房主收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String cont
ent) {
if (cmd.equals("ENTER_SEAT")) {
// 2.房主同意听众请求
mTRTCKaraokeRoom.acceptInvitation(id, null);
}
}
```

步骤9：实现文字聊天和弹幕消息

- 通过 `sendRoomTextMsg` 可以发送普通的文本消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomTextMsg` 回调。

  即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。

```
// 发送端：发送文本消息
mTRTCKaraokeRoom.sendRoomTextMsg("Hello Word!", null);
// 接收端：监听文本消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
@Override
public void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

- 通过 `sendRoomCustomMsg` 可以发送自定义（信令）的消息，所有在该房间内的主播和听众均可以收到 `onRecvRoomCustomMsg` 回调。

  自定义消息常用于传输自定义信令，例如用于点赞消息的发送和广播。

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCKaraokeRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCKaraokeRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInfo userIn
fo) {
if ("CMD_DANMU".equals(cmd)) {
```

```
// 收到弹幕消息
Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
} else if ("CMD_LIKE".equals(cmd)) {
// 收到点赞消息
Log.d(TAG, userInfo.userName + "给您点了个赞！");
}
}
});
```

TRTCKaraoke(iOS)

最近更新时间：2022-01-10 16:21:17

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 Karaoke（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCKaraokeRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateQueue	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 nil，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述

版权所有：腾讯云计算（北京）有限责任公司

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。

API	描述
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

SDK 基础函数

sharedInstance

获取 TRTCKaraokeRoom 单例对象。

```
/**
* 获取 TRTCKaraokeRoom 单例对象
*
* - returns: TRTCKaraokeRoom 实例
* - note: 可以调用 {@link TRTCKaraokeRoom#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCKaraokeRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCKaraokeRoom 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例不能再使用，需要重新调用 {@link TRTCKaraokeRoom#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCKaraokeRoom 事件回调，您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知。

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCKaraokeRoom 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，可使用 {@link TRTCKaraokeRoom#setDelegateQueue(queue)}
```

```
*/
- (void)setDelegate:(id<TRTCKaraokeRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

说明：
setDelegate 是 TRTCKaraokeRoom 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送动主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
* - parameter queue 队列，TRTCKaraokeRoom 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCKaraokeRoom 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
userId:(NSString *)userId
userSig:(NSString *)userSig
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback
:));
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】 > 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

参数	类型	含义
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(RoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的 Karaoke 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Karaoke 房间列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个 Karaoke 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(KaraokeInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(KaraokeUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

音乐播放接口

startPlayMusic

播放音乐（上麦后调用）。

> 说明：
>
> - 播放音乐后，自身会收到 `onMusicPrepareToPlay` 的事件通知。
> - 音乐播放中，房间内所有成员会不断收到 `onMusicProgressUpdate` 的事件通知。
> - 音乐播放完成，自身会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)startPlayMusic:(int32_t)musicID originalUrl:(NSString *)originalUrl accompanyUrl:(NSString *)backingUrl NS_SWIFT_NAME(startPlayMusic(musicID:originalUrl:accompanyUrl:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
musicID	int32_t	音乐的 ID。
originalUrl	String	原唱音乐的绝对路径。
accompanyUrl	String	伴奏音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

> 说明：
> 停止播放后，会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)stopPlayMusic NS_SWIFT_NAME(stopPlayMusic());
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

> 说明：
> - `onMusicProgressUpdate` 的事件通知会暂停
> - 不会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)pausePlayMusic NS_SWIFT_NAME(pausePlayMusic());
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> 说明：
> 不会收到 `onMusicPrepareToPlay` 的事件通知。

```
- (void)resumePlayMusic NS_SWIFT_NAME(resumePlayMusic());
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(pickSeat(seatIndex:userId:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(kickSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> 说明：
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(muteSeat(seatIndex:isMute:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(closeSeat(seatIndex:isClose:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
isMute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enable:));
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
userId:(NSString *)userId
content:(NSString *)content
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendInvitation(cmd:userId:content:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(acceptInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(rejectInvitation(identifier:callback:));
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCKaraokeRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
- (void)onError:(int)code
message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
- (void)onWarning:(int)code
message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)message
NS_SWIFT_NAME(onRoomDestroy(message:));
```

参数如下表所示：

参数	类型	含义
message	String	回调信息。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(KaraokeInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用muteLocalAudio，房间内的其他用户都会收到此通知。

```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	音量大小，取值：0 - 100。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSInteger)totalV
olume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 - 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<KaraokeSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
userInfo:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)cmd
message:(NSString *)message
userInfo:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(cmd:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
inviter:(NSString *)inviter
cmd:(NSString *)cmd
content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(identifier:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	UserInfo	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	邀请人的用户 ID。

音乐播放状态回调

onMusicPrepareToPlay

准备播放音乐的回调

```
- (void)onMusicPrepareToPlay:(int32_t)musicID
NS_SWIFT_NAME(onMusicPrepareToPlay(musicID:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。

onMusicProgressUpdate

歌曲播放进度的回调

```
- (void)onMusicProgressUpdate:(int32_t)musicID
progress:(NSInteger)progress total:(NSInteger)total
NS_SWIFT_NAME(onMusicProgressUpdate(musicID:progress:total:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。
progress	NSInteger	当前播放时间，单位： ms。
total	NSInteger	总时间，单位： ms。

onMusicCompletePlaying

播放完成音乐的回调

```
- (void)onMusicCompletePlaying:(int32_t)musicID
NS_SWIFT_NAME(onMusicCompletePlaying(musicID:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。

版权所有：腾讯云计算（北京）有限责任公司

TRTCKaraoke(Android)

最近更新时间：2022-01-10 16:23:47

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 Karaoke（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCKaraokeRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。

API	描述
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	Karaoke 房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

版权所有：腾讯云计算（北京）有限责任公司

SDK 基础函数

sharedInstance

获取 TRTCKaraokeRoom 单例对象。

```
public static synchronized TRTCKaraokeRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCKaraokeRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCKaraokeRoom 事件回调，您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCKaraokeRoomDelegate delegate);
```

> 说明：
> setDelegate 是 TRTCKaraokeRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCKaraokeRoom 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
String userId, String userSig,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 > 【应用管理】> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCKaraokeRoomDef.RoomParam roomParam, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个 Karaoke 房间列表，腾讯云暂不提供 Karaoke 房间列表的管理服务，请自行管理您的 Karaoke 房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的 Karaoke 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。

4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
public abstract void destroyRoom(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Karaoke 房间列表，可能包含多个 Karaoke 房间的 roomId 和房间信息。
2. 听众选择一个 Karaoke 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCKaraokeRoomCallback.RoomInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCKaraokeRoomCallback.UserListCallback userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

音乐播放接口

startPlayMusic

播放音乐（上麦后调用）。

说明：

- 播放音乐后，自身会收到 `onMusicPrepareToPlay` 的事件通知。
- 音乐播放中，房间内所有成员会不断收到 `onMusicProgressUpdate` 的事件通知。
- 音乐播放完成，自身会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void startPlayMusic(int musicID, String originalUrl, String accompanyUrl);
```

参数如下表所示：

参数	类型	含义
musicID	int	音乐的 ID。
originalUrl	String	原唱音乐的绝对路径。
accompanyUrl	String	伴奏音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

说明：
停止播放后，会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void stopPlayMusic();
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

说明：

- `onMusicProgressUpdate` 的事件通知会暂停
- 不会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void pausePlayMusic();
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> 说明：
> 不会收到 `onMusicPrepareToPlay` 的事件通知。

```
public abstract void resumePlayMusic();
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
public abstract void enterSeat(int seatIndex, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 `sendInvitation` 向房主申请，收到 `onInvitationAccept` 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void leaveSeat(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
public abstract void pickSeat(int seatIndex, String userId, TRTCKaraokeRoomCallback.ActionCallbac
k callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
public abstract void kickSeat(int seatIndex, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> 说明：
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCKaraokeRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> 说明：
> SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCKaraokeRoomDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用 muteLocalAudio，房间内的其他用户都会收到此通知。

```
void onUserMicrophoneMute(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	音量大小，取值：0 - 100。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVolume);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 - 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	下麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

音乐播放状态回调

onMusicPrepareToPlay

准备播放音乐的回调

```
void onMusicPrepareToPlay(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。

onMusicProgressUpdate

歌曲播放进度的回调

```
void onMusicProgressUpdate(int musicID, long progress, long total);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。
progress	long	当前播放时间，单位： ms。
total	long	总时间，单位： ms。

onMusicCompletePlaying

播放完成音乐的回调

```
void onMusicCompletePlaying(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。