实时音视频

含 UI 集成方案

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】

©2013-2022 腾讯云版权所有

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

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

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

【联系我们】

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

文档目录

含 UI 集成方案
- TUIKit 概述
- 音视频通话
  - 集成 TUICalling (Android)
  - 集成 TUICalling (iOS)
  - 集成 TUICalling (Web)
  - 集成 TRTCCalling（Flutter）
  - 集成 TUICalling (小程序)
  - 集成 TUICalling (uni-app / 客户端)
  - 集成 TUICalling (uni-app /小程序)
- 多人视频会议
  - 集成 TUIRoom (Web)
  - 集成 TUIRoom (Android)
  - 集成 TUIRoom (iOS)
  - 集成 TUIRoom (小程序)
  - 集成 TUIRoom (Windows&Mac)
  - 集成 TUIRoom (Flutter)
  - 集成 TUIRoom (Electron)
  - TUIRoom API 查询
    - TUIRoom(Android)
    - TUIRoom API（iOS）
    - TRTCMeeting API（Flutter）
    - TUIRoom(Windows&Mac)
- 语音互动直播
  - 集成 TUIVoiceRoom (Android)
  - 集成 TUIVoiceRoom (iOS)
  - 集成 TUIVoiceRoom (小程序)
  - TUIVoiceRoom API 查询
    - TRTCVoiceRoom (iOS)
    - TRTCVoiceRoom (Android)
- 视频互动直播
  - 集成 TUILiveRoom (Android)
  - 集成 TUILiveRoom (iOS)
  - 集成 TUILiveRoom (Flutter)
  - 集成 TUIPusher&TUIPlayer (Web)

版权所有：腾讯云计算（北京）有限责任公司

TUILiveRoom API 查询
TRTCLiveRoom API (iOS)
TRTCLiveRoom API (Android)
TRTCLiveRoom API (Flutter)
在线 K 歌
集成 TUIKaraoke (iOS)
集成 TUIKaraoke (Android)
TUIKaraoke API 查询
TRTCKaraoke(iOS)
TRTCKaraoke(Android)
实时合唱
集成 TUIChorus（Android）
集成 TUIChorus（iOS）
TUIChorus API 查询
TRTCChorus(Android)
TRTCChorus(iOS)

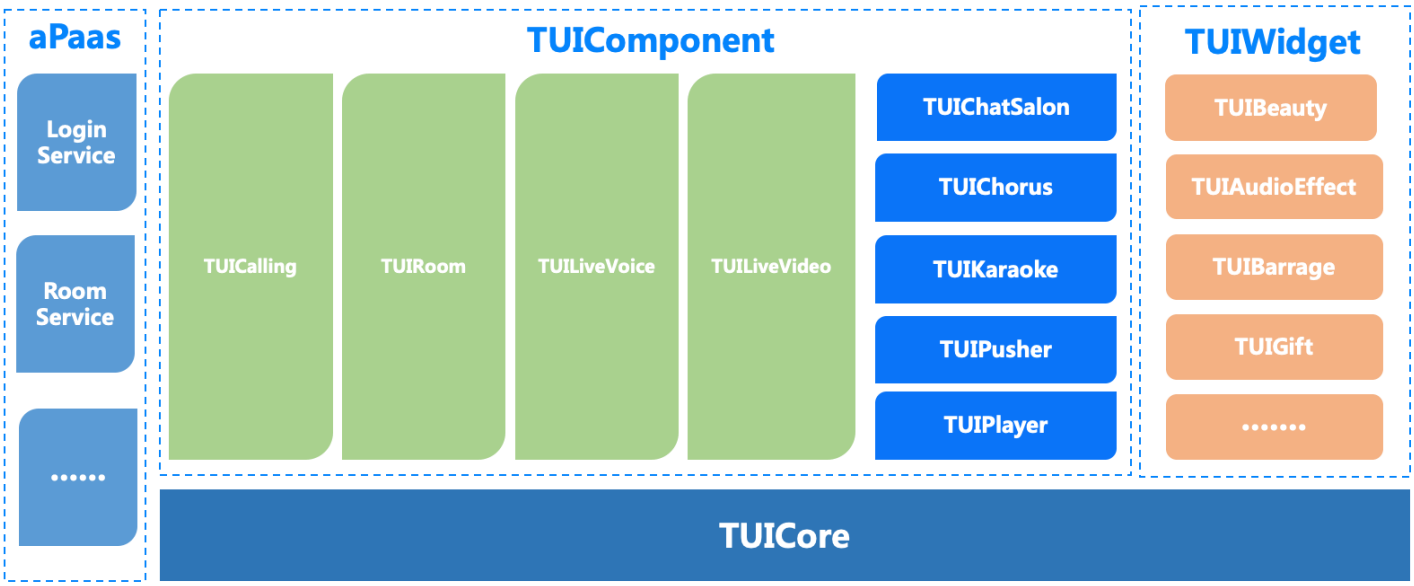
含 UI 集成方案
TUIKit 概述

最近更新时间：2022-06-02 11:02:47

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

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TUIKit 全家桶



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

```
├── TUIComponent
│   ├── TUICalling // 通话组件（类微信通话），针对视频通话、客服、金融审核等音视频场景；
│   ├── TUIRoom // 多人视频房间组件，针对教育、会议、面试等音视频场景，具备禁言禁画等功能；
│   ├── TUILiveVoice // 语聊直播组件，针对交友语聊、排队语聊，音乐房等音频场景；
│   ├── TUILiveVideo // 视频互动直播组件，具备连麦、PK、音效等功能；
│   ├── TUIChatSolon // 语音沙龙组件，针对商务圆桌，兴趣论坛等音视频场景；
│   ├── TUIPusher // 含完整UI的推流组件，支持多种协议推流，PK等功能，支持挂载音效、弹幕等挂件；
```

版权所有：腾讯云计算（北京）有限责任公司

```
  ├── TUIPusher // 含完整UI的播放组件，支持多种协议播放，连麦主播等功能，支持挂载礼物、弹幕等挂件；
  ├── TUIPlayer // 创新组件，针对在线KTV等新玩法的音视频的组件，可以搭配腾讯云正版曲库解锁更多玩法；
  ├── TUIChorus // 创新组件，针对合唱的音视频新玩法的音视频组件，可以搭配腾讯云正版曲库解锁更多玩法；
  ├── TUIKaraoke // 创新组件，针对在线KTV等新玩法的音视频的组件，可以搭配腾讯云正版曲库解锁更多玩法；
├── TUWidget
  ├── TUIBeauty // 美颜挂件
  ├── TUIAudioEffect // 音效挂件
  ├── TUIBarrage // 弹幕挂件
  ├── TUIGift // 礼物挂件
├── aPass （可选）
  ├── Room Service // 提供房间列表、房间热度、主播列表等功能；
```

交流&反馈

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

- TUIKit 常见问题
- **TUIKit 技术交流 QQ 群：592465424**



- 更多联系

音视频通话
集成 TUICalling (Android)

最近更新时间：2022-06-23 17:13:41

组件介绍

TUICalling 是一个开源的音视频 UI 组件，通过在项目中集成 TUICalling 组件，您只需要编写几行代码就可以为您的 App 添加“一对一音视频通话”场景，并且支持离线唤起能力。TUICalling 同时支持 iOS、Web、小程序、Flutter、UniApp 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



> 说明：

> TUICalling 组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

组件集成

步骤一：下载并导入 TUICalling 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android 目录下的 tuicalling 和 debug 目录到您的工程的 app 同一级目录，并完成如下导入动作：

- 在 setting.gradle 中完成导入，参考如下：

```
include ':tuicalling'
include ':debug'
```

- 在 app 的 build.gradle 文件中添加对 tuicalling 的依赖：

```
api project(':tuicalling')
```

- 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

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

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" /> // 使用场景：悬浮窗、应用在后台时拉起通话界面时需要此权限；
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

```
<uses-permission android:name="android.permission.BLUETOOTH" /> // 使用场景：使用蓝牙耳机时需要此权限；
<uses-permission android:name="android.permission.READ_PHONE_STATE" /> // 使用场景：判断是否是系统来电打断时需要此权限；
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus" />
```

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

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

步骤三：创建并初始化组件

```
// 1.添加事件监听及登录
TUILogin.addLoginListener(new TUILoginListener() {
@Override
public void onConnecting() { // 正在连接中
super.onConnecting();
}
@Override
public void onConnectSuccess() { // 连接成功通知
super.onConnectSuccess();
}
@Override
public void onConnectFailed(int errorCode, String errorMsg) { // 连接失败通知
super.onConnectFailed(errorCode, errorMsg);
}
@Override
public void onKickedOffline() { // 登录被踢下线通知（示例：账号在其他设备登录）
super.onKickedOffline();
}
@Override
public void onUserSigExpired() { // userSig过期通知
super.onUserSigExpired();
}
});
```

```
TUILogin.login(mContext, "Your SDKAppID", "Your userId", "Your userSig", new TUICallback() {
@Override
public void onSuccess() {
}
@Override
public void onError(int errorCode, String errorMsg) {
Log.d(TAG, "errorCode: " + errorCode + " errorMsg:" + errorMsg);
}
});
// 2.初始化TUICalling实例
TUICalling callingImpl = TUICallingImpl.sharedInstance(context);
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息请参见 如何计算及使用 UserSig。

步骤四：实现音视频通话

实现1对1视频通话/音频通话 TUICalling#call：

```
// 发起1对1视频通话，假设userId为：1111；
callingImpl.call(["1111"], TUICalling.Type.VIDEO);
```

> **说明：**
>
> - 当接收方完成步骤三后，即登录成功后，再收到通话请求后，TUICalling 组件会自动启动相应的接听界面。
> - 如果您想发起语音通话，更改类型为 TUICalling.Type.AUDIO 即可。

步骤五：离线推送（可选）

完成以上四个步骤，就可以实现音视频通话的拨打和接通，但如果您的业务场景需要在 App 的进程被杀死后 或者 APP 退到后台后 ，还可以正常接收到音视频通话请求，就需要为 TUICalling 组件增加离线推送功能，详情见 **Android 离线推送接入指引**。

步骤六：状态监听（可选）

如果您的业务需要 监听通话的状态，例如通话开始、结束等，可以监听以下事件：

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

@Override
public void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, View tuiCalli
ngView) {

}

@Override
public void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTime
) {

}
```

```
@Override
public void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, Strin
g message) {
Log.d(TAG, "onCallEvent: event = " + event + " ,message = " + message);
}
});
```

步骤七：悬浮窗功能（可选）

如果您的业务需要开启悬浮窗功能，您可以在 TUICalling 组件初始化时调用 callingImpl.enableFloatWindow(true) 开启该功能。

目前组件支持以下两种情况悬浮窗：

- 系统悬浮窗(单击 home 键退到后台)：需开启悬浮窗权限。
- 应用内悬浮窗(最小化退到上一层界面)：需开启悬浮窗权限，并设置要返回的上一层界面。
  其中开启悬浮窗权限方法：打开**手机设置**，找到**应用管理**，找到您的应用，单击**权限**，单击**悬浮窗并允许**（手机厂商、平台不同，该位置可能有差异）。

设置要返回的上一层界面，在 AndroidManifest.xml 为您需要跳转的界面配置跳转动作 com.tencent.trtc.tuicalling，例如：

```
<activity
android:name="{packageName}.MainActivity"
android:launchMode="singleTop">
<intent-filter>
<action android:name="com.tencent.trtc.tuicalling" />
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
```

常见问题

TUICalling 组件支持自定义头像和昵称吗?

支持，调用 setUserNickname/setUserAvatar 即可。

TUICalling 组件支持自定义铃声吗?

支持，调用 setCallingBell 即可。

> **说明：**
> 更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUICalling (iOS)

最近更新时间：2022-06-23 17:13:45

组件介绍

TUICalling 是一个开源的音视频 UI 组件，通过在项目中集成 TUICalling 组件，您只需要编写几行代码就可以为您的 App 添加“一对一音视频通话”场景，并且支持离线唤起能力。TUICalling 同时支持 Android、Web、小程序、Flutter、UniApp 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



> 说明：
> TUICalling 组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

组件集成

步骤一：导入 TUICalling 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 在您的工程 Podfile 文件同一级目录下创建 TUICalling 文件夹。
2. 单击进入 **Github/TUICalling** ，选择克隆/下载代码，然后将 **TUICalling/iOS/** 目录下的 Source、Resources 文件夹 和 TUICalling.podspec 文件拷贝到您在 步骤1 创建的 TUICalling 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖，之后执行 pod install 命令，完成导入。

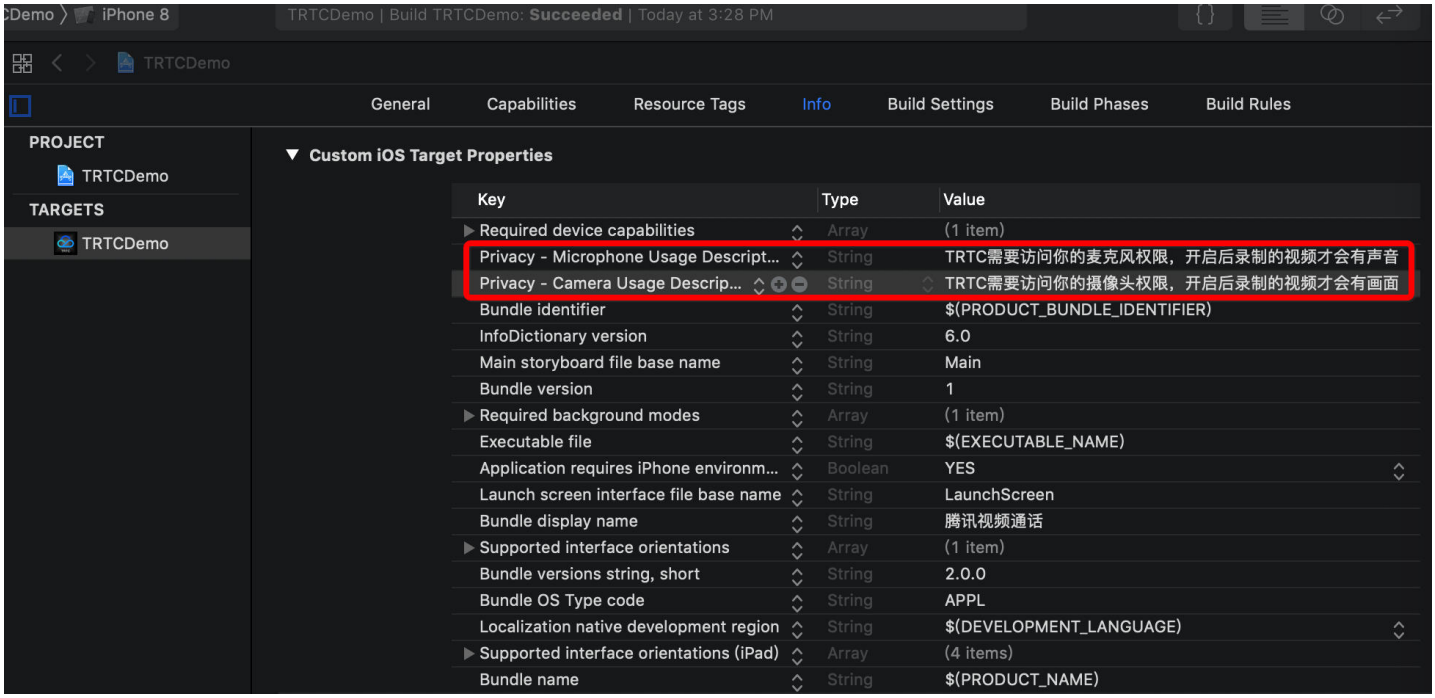
```
# :path => "指向TUICalling.podspec的相对路径"
pod 'TUICalling', :path => "TUICalling/TUICalling.podspec", :subspecs => ["TRTC"]
```

> ⚠ 注意：
>
> Source 、 Resources 文件夹 和 TUICalling.podspec 文件必需在同一目录下。

步骤二：配置权限

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

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



步骤三：创建并初始化组件

Objective-C

```
// 1.组件登录
[TUILogin initWithSdkAppID:@"您的SDKAppID"];
[TUILogin login:@"您的UserID" userSig:@"您的UserSig" succ:^{
NSLog(@"login success");
} fail:^(int code, NSString *msg) {
NSLog(@"login failed, code: %d, error: %@", code, msg);
}];

// 2.初始化TUICalling实例
[TUICalling shareInstance];
```

Swift

```
// 1.组件登录
TUILogin.initWithSdkAppID("您的SDKAppID")
TUILogin.login("您的UserID", userSig: "您的UserSig") {
print("login success")
} fail: { code, message in
print("login failed, code: \(code), error: \(message ?? "nil")")
}
```

```
// 2.初始化TUICalling实例
TUICalling.shareInstance()
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppID 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **UserID**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **UserSig**：根据 SDKAppId、UserID，SecretKey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现音视频通话

实现1对1视频通话 TUICalling#call：

Objective-C

```
// 发起1对1视频通话，假设userId为：1111
[[TUICalling shareInstance] call:@[@"1111"] type:TUICallingTypeVideo];
```

Swift

```
// 发起1对1视频通话，假设userId为：1111
TUICalling.shareInstance().call(userIDs: ["1111"], type: .video)
```

> **说明：**
> 当接收方完成步骤三后，即登录成功后，再收到通话请求后，TUICalling组件会自动启动相应的接听界面。

步骤五：离线推送（可选）

完成以上四个步骤，就可以实现视频通话的拨打和接通，但如果您的业务场景需要在 App 的进程被杀死后 或者 App 退到后台后 ，还可以正常接收到音视频通话请求，就需要为 TUICalling 组件增加离线推送功能，详情见 **离线推送（iOS）**。

步骤六：状态监听（可选）

如果您的业务需要 **监听通话的状态**，例如通话开始、结束等，可以监听以下事件：

Objective-C

```
[[TUICalling shareInstance] setCallingListener:self];

- (BOOL)shouldShowOnCallView {
return YES;
}

- (void)callStart:(nonnull NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUI
CallingRole)role viewController:(UIViewController * _Nullable)viewController {

}

- (void)callEnd:(nonnull NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUIC
allingRole)role totalTime:(float)totalTime {

}

- (void)onCallEvent:(TUICallingEvent)event type:(TUICallingType)type role:(TUICallingRole)rol
e message:(nonnull NSString *)message {

}
```

Swift

```
TUICalling.shareInstance().setCallingListener(listener: self)

public func shouldShowOnCallView() -> Bool {
return true
}

public func callStart(userIDs: [String], type: TUICallingType, role: TUICallingRole, viewControll
er: UIViewController?) {

}

public func callEnd(userIDs: [String], type: TUICallingType, role: TUICallingRole, totalTime: Fl
oat) {

}

public func onCallEvent(event: TUICallingEvent, type: TUICallingType, role: TUICallingRole, m
essage: String) {

}
```

步骤七：悬浮窗功能（可选）

如果您的业务需要开启 悬浮窗功能，您可以在 TUICalling 组件初始化时调用 TUICalling.shareInstance().enableFloatWindow(enable: true) 开启该功能。

> **说明：**
> 目前组件仅支持应用内悬浮窗（最小化退到上一层界面）。

常见问题

TUICalling 组件支持自定义铃声吗?

支持，调用 TUICalling#setCallingBell 即可。

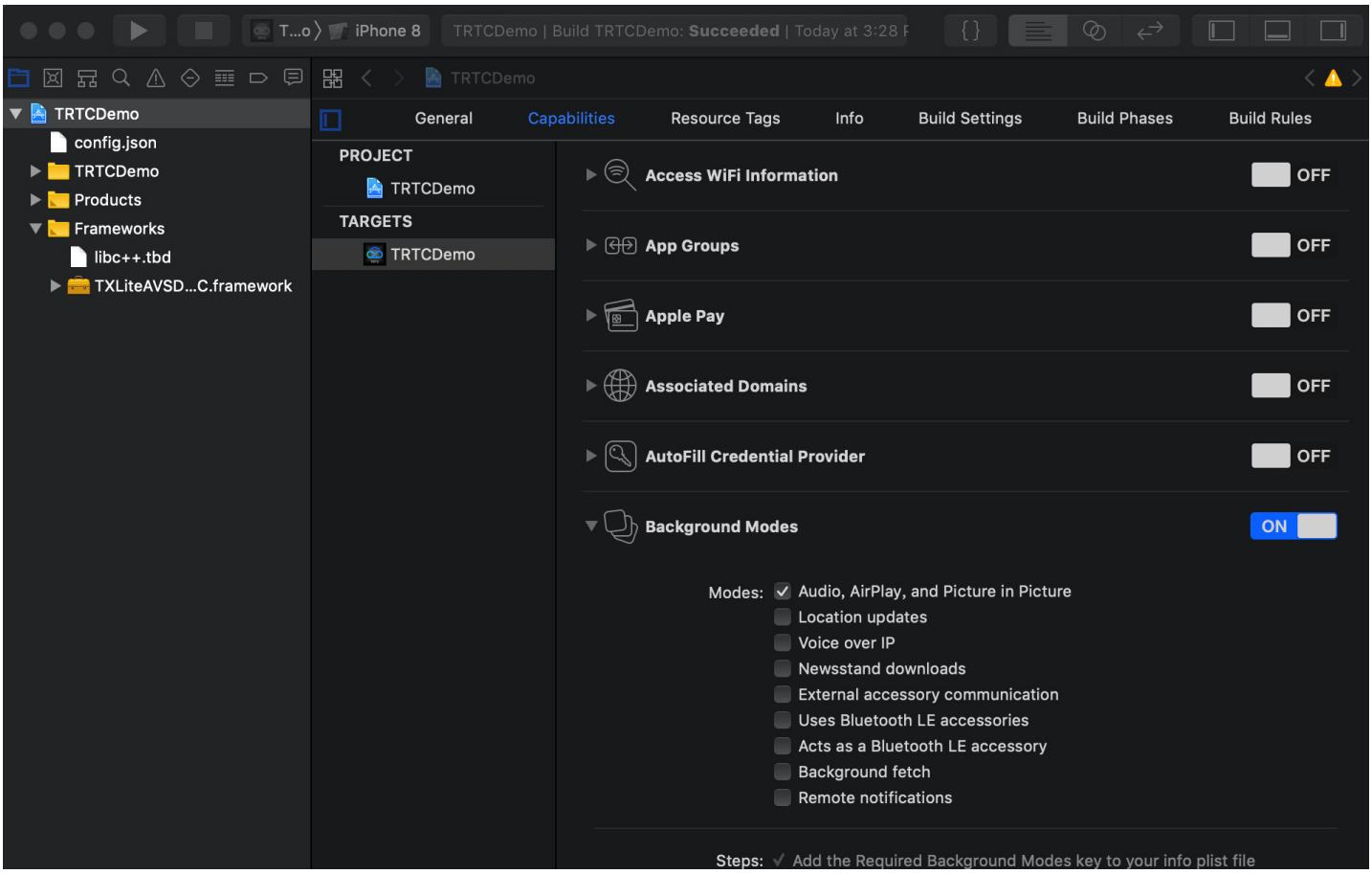
CocoaPods如何安装?

在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：

```
sudo gem install cocoapods
```

TUICalling 是否支持后台运行?

支持，如需要进入后台仍然运行相关功能，可选中当前工程项目，在 Capabilities 下的设置 Background Modes 打开为 ON，并勾选 Audio，AirPlay and Picture in Picture ，如下图所示：



说明：

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

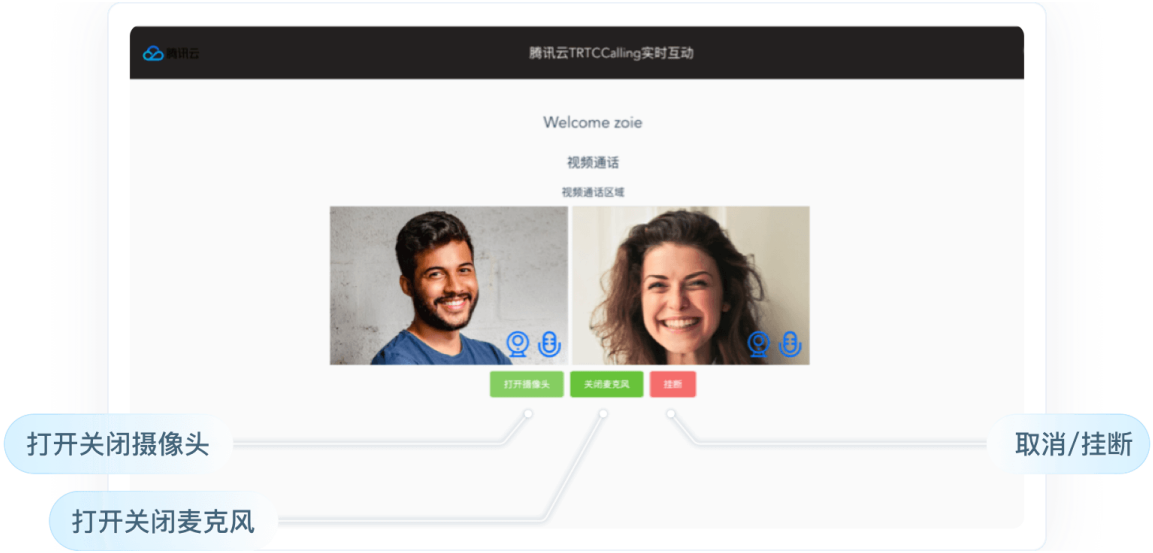
集成 TUICalling (Web)

最近更新时间：2022-06-23 17:13:49

组件介绍

TUICalling 组件是一个开源的音视频组件，帮助您快速在您的桌面浏览器中集成**视频通话**功能，非常适合用于在线问诊，在线客服，远程理赔等场景中。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



> 说明：
> TUICalling 组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

其它平台

除了 Web 版的 TUICalling，我们同时也推出了 Android、iOS、小程序、Flutter、Uniapp 等平台的源代码，其中 Android、iOS 版本的 TUICalling 支持“来电提醒”功能。

> 说明：

- **除了 Web 版的 TUICalling，我们同时也推出了 Android、iOS、小程序、Flutter、Uniapp 等平台的源代码，其中 Android、iOS 版本的 TUICalling 支持“来电提醒”功能。**
- 如果您有任何咨询或建议，可以加入 QQ 群 **592465424** 一起交流。

组件集成

步骤一：获取 SdkAppId 和签名密钥

- 如果您还没有腾讯云账号，请注册一个腾讯云账号，并完成 实名认证。然后跳转到 TRTC 管理控制台中的应用管理界面。
- 如果您的应用列表为空，可以点击**创建应用**按钮创建一个新的应用，之后单击**应用信息**，打开该应用管理界面。在该界面中寻找**快速上手**页签，就能够看到如下界面：



- **SDKAppID**：TRTC 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话彼此不能互通；
- **Secretkey**：TRTC 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 TRTC 服务的鉴权用票据 UserSig，我们会在接下来的步骤五中用到这个 Key。

步骤二：下载并集成 TRTCCalling 组件

单击进入 Github ，选择克隆/下载代码，可参考 Web 目录下的实现。

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>
```

步骤三：创建 TRTCCalling 对象

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

```
// 可参考 Web/src/trtc-calling/index.js
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);
```

步骤四：进行登录与事件监听

```
// 可参考 Web/src/components/login/index.vue
trtcCalling.login({
```

```
userID,
userSig,
});

// 可参考 Web/src/App.vue

export default {
name: "App",
components: {
},
async created() {
this.initListener();
},
data() {
return {};
},
destroyed() {
this.removeListener();
},
methods: {
initListener: function() {
// 远端用户呼叫
trtcCalling.on(trtcCalling.EVENT.INVITED, this.handleNewInvitationReceived);
// 远端用户接听
trtcCalling.on(trtcCalling.EVENT.USER_ACCEPT, this.handleUserAccept);
// 远端用户拒绝
trtcCalling.on(trtcCalling.EVENT.REJECT, this.handleInviteeReject);
// ...
},
removeListener: function() {
trtcCalling.off(trtcCalling.EVENT.INVITED, this.handleNewInvitationReceived);
trtcCalling.off(trtcCalling.EVENT.USER_ACCEPT, this.handleUserAccept);
trtcCalling.off(trtcCalling.EVENT.REJECT, this.handleInviteeReject);
},
handleNewInvitationReceived: async function(payload) {
},
handleUserAccept: function() {
},
```

```
handleInviteeReject: function() {
}
}
}
```

步骤五：实现 1v1 通话

- **主叫方：呼叫某个用户**

```
// 可参考 Web/src/components/video-call/index.vue 或 Web/src/components/audio-call/index.v
ue
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**

```
// 可参考 Web/src/App.vue handleAcceptCall 方法
// 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。

对于环境有哪些要求?

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+

操作系统	浏览器类型	浏览器最低版本要求
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+
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	-

更多常见问题

请参见 TRTCCalling Web 相关问题。

集成 TRTCCalling（Flutter）

最近更新时间：2022-06-02 11:01:32

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。

> 说明：
> TUICalling 组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

复用 Demo 的 UI 界面

步骤1：创建新的应用

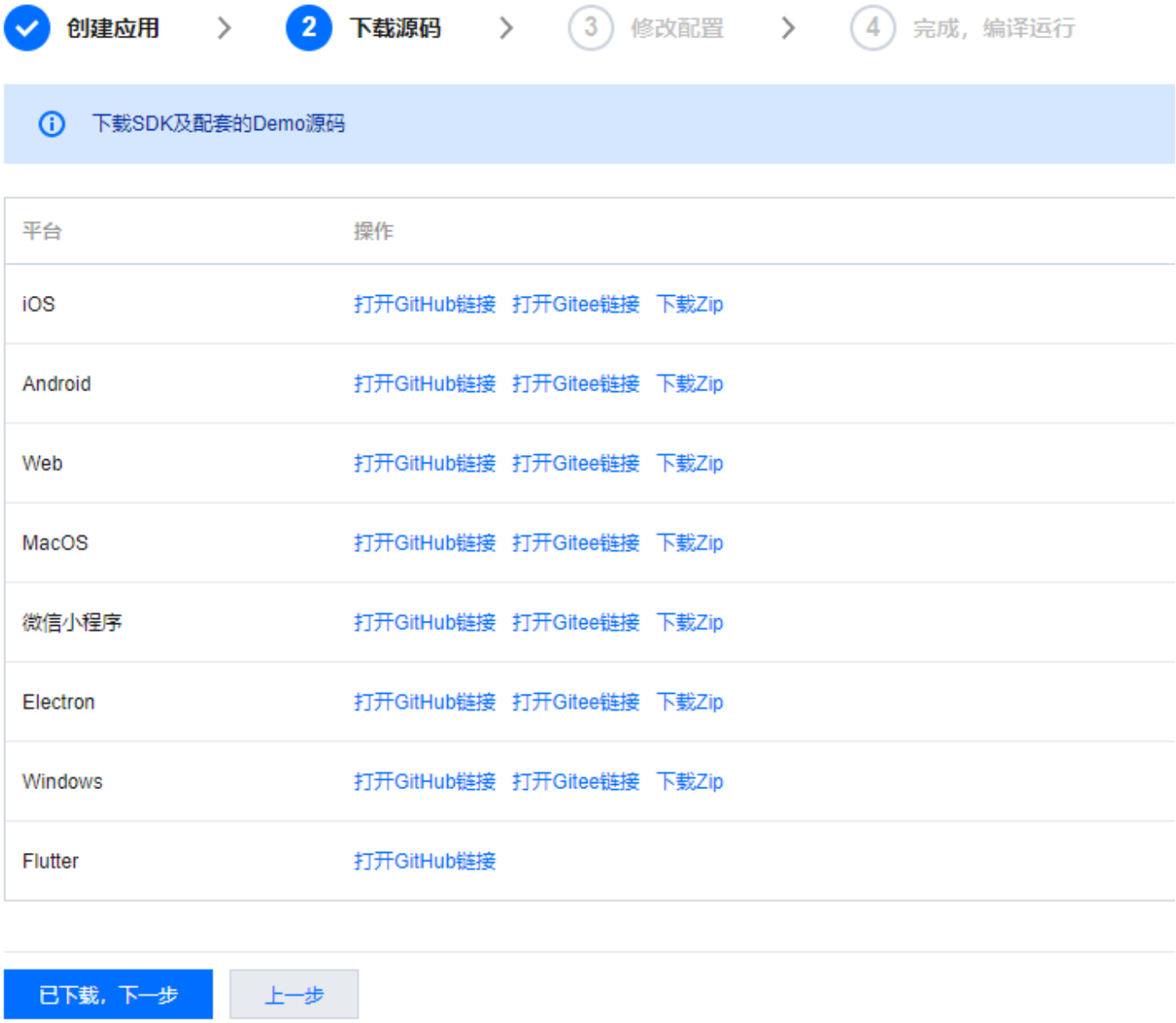
1. 登录实时音视频控制台，选择**开发辅助** > **快速跑通Demo**。
2. 输入应用名称，例如 TestVideoCall ，单击**创建**。

> 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。

```
class GenerateTestUserSig {
  /*
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  static int sdkAppId = 0;
  /*
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  static int expireTime = 604800;

  /*
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc )，如果还没有应用就创建一个，
   * step2. 单击"应用配置"进入基础配置页面，并进一步找到"帐号体系集成"部分。
   * step3. 点击"查看密钥"按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  static String secretKey = '';
```

将SDKAppID粘贴到此处

将 密钥 粘贴到此处

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

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

步骤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**
   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。
   iOS 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。

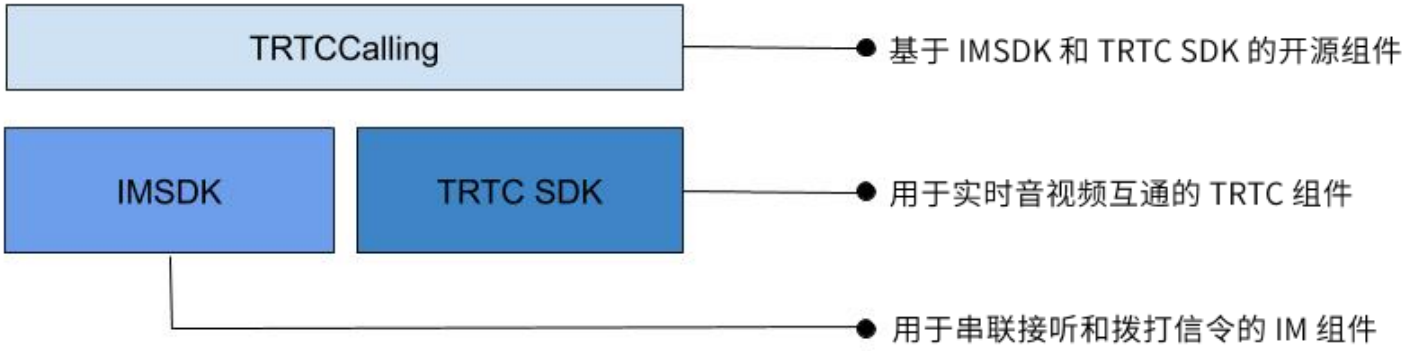
步骤5：修改 Demo 源代码

源码 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

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>
```

Android 端

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

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

android > app > src > main > AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mlp">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        tools:replace="android:label"
        android:name="io.flutter.app.FlutterApplication"
        android:label="mlp"
        android:icon="@mipmap/ic_launcher">
```

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

版权所有：腾讯云计算（北京）有限责任公司

接口函数	接口功能
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

集成 TUICalling (小程序)

最近更新时间：2022-06-23 17:13:53

组件介绍

TUICalling 组件是一个开源的音视频组件，帮助您快速在您的小程序中集成**视频通话**功能，非常适合用于在线问诊，在线客服，远程理赔等场景中。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

主动呼叫

由于用户不需要安装 App 就可以使用该功能，因此 TUICalling 非常适合用于快速发起视频通话的场景。

被动呼叫

由于小程序无法获得类似微信通话那样强大的**来电提醒**能力，因此只有当小程序在前台时才支持“被呼叫”的能力，因此它并不适合需要“来电提醒”的场景中。如果用在“在线问诊”或者“在线客服”等场景中，我们推荐您使用小程序主动呼叫 + 桌面端被动应答的组合方案。

其它平台

除了小程序版的 TUICalling，我们同时也推出了 Android、iOS、Web、Flutter、Uniapp 等平台的源代码，其中 Android、iOS 版本的 TUICalling 支持“来电提醒”功能。



> 说明：
> 如果您有任何咨询或建议，可以加入 QQ 群 **592465424** 一起交流~

组件集成

步骤一：开通小程序权限

由于 TUICalling 所使用的小程序标签有更苛刻的权限要求，因此集成 TUICalling 的第一步就是要开通小程序的类目和标签使用权限，**否则无法使用**，这包括如下步骤：

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



- 小程序推拉流标签使用权限暂时只开放给有限 类目。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：



步骤二：在小程序控制台配置域名

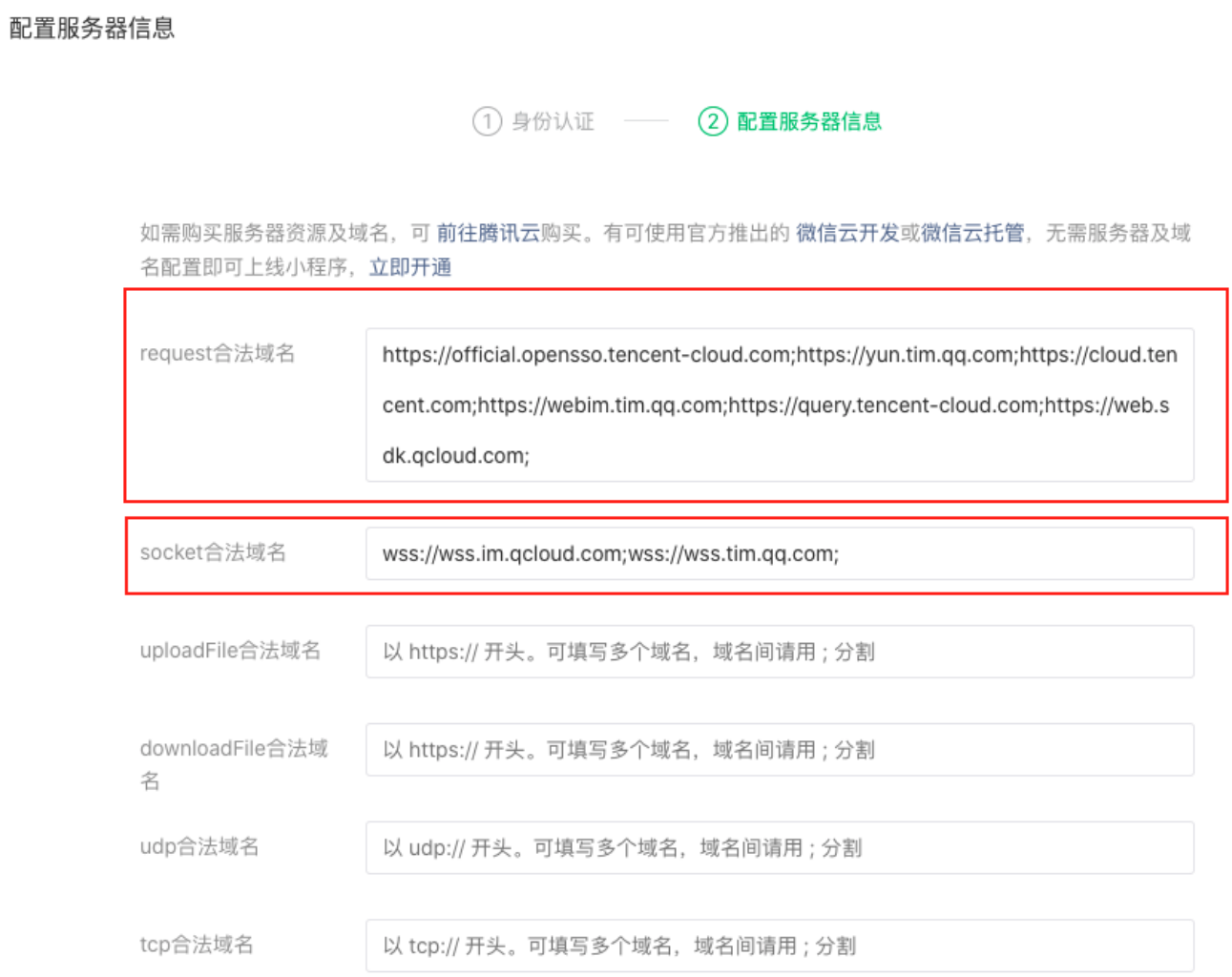
在 **微信公众平台** > **开发** > **开发管理** > **开发设置** > **服务器域名**中设置 **request 合法域名** 和 **socket 合法域名**，如下图所示：

- request 合法域名：

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

- socket 合法域名：

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



步骤三：获取 SdkAppId 和签名密钥

- 如果您还没有腾讯云账号，请注册一个腾讯云账号，并完成 实名认证。然后跳转到 TRTC 管理控制台中的 应用管理 界面。
- 如果您的应用列表为空，可以单击**创建应用**创建一个新的应用，之后单击**应用信息**，打开该应用管理界面。在该界面中寻找**快速上手**页签，就能够看到如下界面：



- **SDKAppID**：TRTC 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话彼此不能互通。
- **Secretkey**：TRTC 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 TRTC 服务的鉴权用票据 UserSig，我们会在接下来的步骤五中用到这个 Key。

步骤四：下载并导入 TUICalling 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 MiniProgram 下的 TUICalling 目录到您的工程中。

步骤五：创建并初始化 TUI 组件库

1. 添加组件到对应页面的 页面配置，例如 pages/index/index.json：

```
// 可参考 MiniProgram/pages/videoCall/videoCall.json 或 MiniProgram/pages/audioCall/audioCall.json
{
"usingComponents": {
"TUICalling": "../../TUICalling/TUICalling"
}
}
```

2. 添加 JS 逻辑交互，初始化并使用组件，例如 pages/index/index.js：

```
// 可参考 MiniProgram/pages/videoCall/videoCall.js 或 MiniProgram/pages/audioCall/audioCall.js
Page({
/**
```

```
* 页面的初始数据
*/
data: {
config: {
sdkAppID: 0,
userID: 0,
userSig: '',
type: 1,
tim: null, // 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性
}
},

/**
* 生命周期函数--监听页面加载
*/
onLoad: function (options) {
// 将初始化后到TUICalling实例注册到this.TUICalling中，this.TUICalling 可使用TUICalling所以方法功能。
this.TUICalling = this.selectComponent('#TUICalling-component');
//初始化TUICalling
this.TUICalling.init()
},
/**
* 生命周期函数--监听页面卸载
*/
onUnload() {
this.TUICalling.destroyed()
},

call: function() {
// 注：初始化完成后调用call方法
// type 1：语音通话，2：视频通话。
this.TUICalling.call({ userID: this.data.userId, type:2})
},
})
```

3. 在 WXML 模板 中添加一个 TUICalling 组件，例如示例代码中的 pages/index/index.wxml：

```
// 可参考 MiniProgram/pages/videoCall/videoCall.wxml 或 MiniProgram/pages/audioCall/audioCall.wxml
<TUICalling
id="TUICalling-component"
config="{{config}}"
></TUICalling>
```

步骤六：用 JS 代码动态设置 Config 参数

在您的项目中添加 javascript 的代码，用于设置 wxml 文件中的 {{config}} 变量。这部分工作可参考 MiniProgram/pages/videoCall/videoCall.js 或 MiniProgram/pages/audioCall/audioCall.js 中的示例代码，如下所示：

```
// 引入 userSig 生成函数
import { genTestUserSig } from "../../../public/debug/GenerateTestUserSig";
const userID = 'testMe';
Page({
/**
* 页面的初始数据
*/
data: {
config: {
sdkAppID: 0, // 替换为步骤三中获取的 SDKAppID
userID: userID, // 填写当前用的 userID
userSig: genTestUserSig(userID), // 通过 genTestUserSig(userID) 生成
tim: null, // 如果您的项目中未集成。
}
}
}
```

这里详细介绍一下 config 中的几个参数：

- **sdkAppID**：在步骤三中您已经获取到，这里不再赘述。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **userSig**：使用步骤三中获取的 SecretKey 对 sdkAppID、userId 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务，更多信息请参见 如

何计算及使用 UserSig。

> ⚠ 注意：
> 示例代码中使用 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在小程序的代码中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由小程序在每次使用 TUICalling 组件时向您的服务器请求实时计算出的 UserSig。

步骤七：发起视频通话请求

- **实现1对1视频通话**
  在 JS 逻辑交互例如 pages/index/index.js 中填写如下代码，就可以实现一对一视频通话。

```
// 发起1对1视频通话，假设 userId为: 1111, type 1：语音通话，2：视频通话。
this.TUICalling.call({ userID: '1111', type: 2 });
```

- **实现多人视频通话**
  在 JS 逻辑交互例如 pages/index/index.js 中填写如下代码，就可以实现同时发起对多个人的视频通话邀请。

```
// 发起多人视频通话，假设 userId 分别为：1111、2222、3333, type 1：语音通话，2：视频通话。
this.TUICalling.groupCall({ userIDList: ['1111', '2222', '3333'], type: 2 });
```

- **实现事件通知监听**
  在 TUICalling 组件内部 MiniProgram/TUICalling/TUICalling.js 内部已经默认实现远端用户接听、远端用户挂断的处理逻辑。业务侧也可以进行主动监听，并做相应的业务处理。在 JS 逻辑交互，例如 pages/index/index.js 中填写如下代码。

```
// 可参考 MiniProgram/pages/videoCall/videoCall.js 或 MiniProgram/pages/audioCall/audioCall.j
s 里 onLoad 生命周期里的逻辑
// ...
onLoad: function() {
// ...
this.TUICalling = this.selectComponent('#TUICalling-component')
this.TUICalling.init()
// 远端用户接听
wx.$TRTCCalling.on(wx.$TRTCCalling.EVENT.USER_ACCEPT, this.handleUserAccept, this);
// 远端用户拒绝
wx.$TRTCCalling.on(wx.$TRTCCalling.EVENT.REJECT, this.handleUserReject, this);
```

```
// 远端用户无应答
wx.$TRTCCalling.on(wx.$TRTCCalling.EVENT.NO_RESP, this.handleNoResponse, this);
// 更多事件可参考组件内部事件处理逻辑 MiniProgram/TUICalling/TUICalling.js _addTSignalingEven
t 方法
},

handleUserAccept: function(event) {
console.log(`远端${event.data.userID}已接通电话`);
},
handleUserReject: function(event) {
console.log(`远端${event.data.invitee}拒绝电话`);
},
handleNoResponse: function(event) {
console.log(`远端${event.data.timeoutUserList[0]}无应答`);
},
// ...
onUnload: function() {
wx.$TRTCCalling.off(wx.$TRTCCalling.EVENT.USER_ACCEPT, this.handleUserAccept);
wx.$TRTCCalling.off(wx.$TRTCCalling.EVENT.REJECT, this.handleUserReject);
wx.$TRTCCalling.off(wx.$TRTCCalling.EVENT.NO_RESP, this.handleNoResponse);
this.TUICalling.destroyed()
},
// ...
```

步骤八：实现一个建议的在线医疗方案

如果您的产品只有手机端或者只有小程序端，则可以不用关心这一步。
如果您的业务场景是“在线医疗”或者“在线客服”类的场景，则还需要实现被叫方的代码逻辑，由于被叫方通常是“医生”或者“客服”，使用的通常是 PC 台式机，因此我们推荐您用 Web 端的 TUICalling 方案实现坐席

（我们将被咨询的客服或者医生称之为“坐席”）方逻辑，具体方案如下图所示：



1. 当用户在小程序中单击”呼叫医生“或者”在线咨询”按钮时，会向您的服务器发送一个分配“坐席”的请求。
2. 您的后台分配系统查询当前有无空闲的“坐席”人员，如果当前所有“坐席”都出于“忙线状态“，则返回用户”请等待或稍后再试”的提示；如果当前有空闲的“坐席”，则将坐席的 UserId 返回给发起咨询的用户，一同返回给用户的还有 SDKAppID，以及用户自己的 UserId 所签发出的 UserSig。
3. 用户端的小程序此时可以按照步骤六，给自己的 TUICalling 组件设置 Config 变量，并按照步骤七发起对“坐席”的呼叫。
4. “坐席端” 接到通话请求，选择接通，并完成接下来的在线问诊或者在线客服的流程。

常见问题

是否支持 uniapp、taro 等开发环境?

taro 暂不支持，uniapp 请参见 快速集成方案（uni-app）。

如何全局监听，在所有页面都可以唤起通话界面?

目前不支持全局监听，页面唤起需要在唤起的页面添加组件。

相关文档

- TUICalling demo 源码
- TUICalling API
- Web 端 TUICalling

- 小程序端相关问题

集成 TUICalling (uni-app / 客户端)

最近更新时间：2022-06-23 17:13:57

组件介绍

TUICalling 插件是**腾讯云官方推出**的基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的插件，支持双人和多人音视频通话。插件提供了全套定制 UI，开发者仅需两个 API 可集成实现通话功能。TUICalling 同时支持 iOS、Web、小程序、Flutter、UniApp 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



> 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。

组件集成

步骤一：导入插件

1. **购买 uni-app 原生插件：**

   登录 uni 原生插件市场，在插件详情页中购买（免费插件也可以在插件市场0元购）。购买后才能够云端打包使用插件。**购买插件时请选择正确的 appid，以及绑定正确包名。**



2. 使用自定义基座打包 uni 原生插件（**请使用真机运行自定义基座**）。

   使用 uni 原生插件必须先提交云端打包才能生效，购买插件后在应用的 manifest.json 页面的 **App原生插件配置** 项下单击**选择云端插件**，选择**腾讯云原生音视频插件**。



直接云端打包后无法打 log，无法排查问题，需要自定义基座调试原生插件。

> ⚠ 注意：
>
> - 自定义基座不是正式版，真正发布时，需要再打正式包。使用自定义基座是无法正常升级替换 APK 的。
> - 请尽量不要使用本地插件，插件包超过自定义基座的限制，可能导致调试收费

步骤二：在 vue 页面中引入原生插件

使用 uni.requireNativePlugin 的 API 在 vue 页面中引入原生插件，参数为插件的 ID。

```
const TUICalling = uni.requireNativePlugin("TUICallingUniPlugin-TUICallingModule");
```

步骤三：获取 SdkAppId 和签名密钥

- SDKAppID：TRTC 应用 ID，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：在 vue 页面中实现功能

1. 登录并填写您的应用 sdkAppID 等信息。

```
TUICalling.login({
sdkAppID: 0,
userID: 'your userID',
userSig: 'your userSig'
})
```

2. 发起音视频通话。
   - 发起双人通话：

```
TUICalling.call({ userID: 'user1', type: 1 })
```

- 发起群通话：

```
TUICalling.groupCall({ userIDList: ['user1', 'user2'], type: 1 })
```

3. 登出。

```
TUICalling.logout()
```

步骤五：本地调试和发布

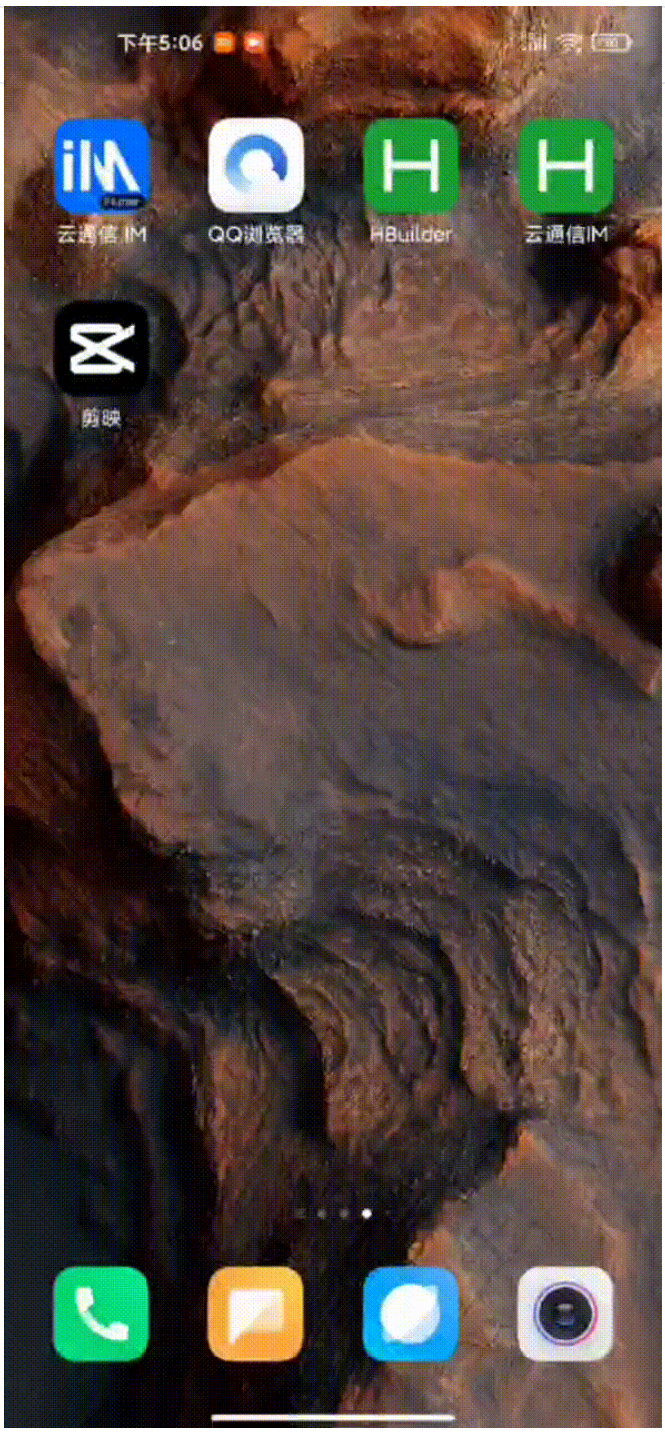
使用自定义基座开发调试 uni-app 原生插件 后，不可直接将自定义基座 APK 作为正式版发布。
应该重新提交云端打包（不能勾选“**自定义基座**”）生成正式版本。

实现案例

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

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

Android 演示	iOS 演示



iOS 演示



技术咨询

了解更多详情您可 QQ 咨询：**309869925** (技术交流群)

相关文档

- uni-app 原生音视频插件示例

- uni-app TUIKit 源码
- 一分钟跑通 Demo (uni-app)
- 快速集成 uni-app TUIKit

集成 TUICalling (uni-app /小程序)

最近更新时间：2022-06-23 17:14:01

组件介绍

TUICalling 小程序组件是基于腾讯云实时音视频（TRTC）和腾讯云信令 SDK（TSignalling）组合而成，支持1V1，多人场景下的视频通话。TUICalling 同时支持 iOS、Web、小程序、Flutter、UniApp 等平台，基本功能如下图所示：



> **说明：**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

> **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。

环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.10.0。
- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在真机上进行运行体验。
- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，需要申请常规小程序账号进行开发。

组件集成

步骤一：开通小程序权限

1. 您已 注册腾讯云 账号，并完成 实名认证。
2. **开通小程序类目与推拉流标签权限（如不开通则无法正常使用）。**
   出于政策和合规的考虑，微信暂未放开所有小程序对实时音视频功能（即 <live-pusher> 和 <live-player> 标签）的支持：
   - 小程序推拉流标签不支持个人小程序，只支持企业类小程序。
   - 小程序推拉流标签使用权限暂时只开放给有限 类目。
   - 符合类目要求的小程序，需要在 **微信公众平台 > 开发 > 开发管理 > 接口设置** 中自助开通该组件权限，如下图所示：



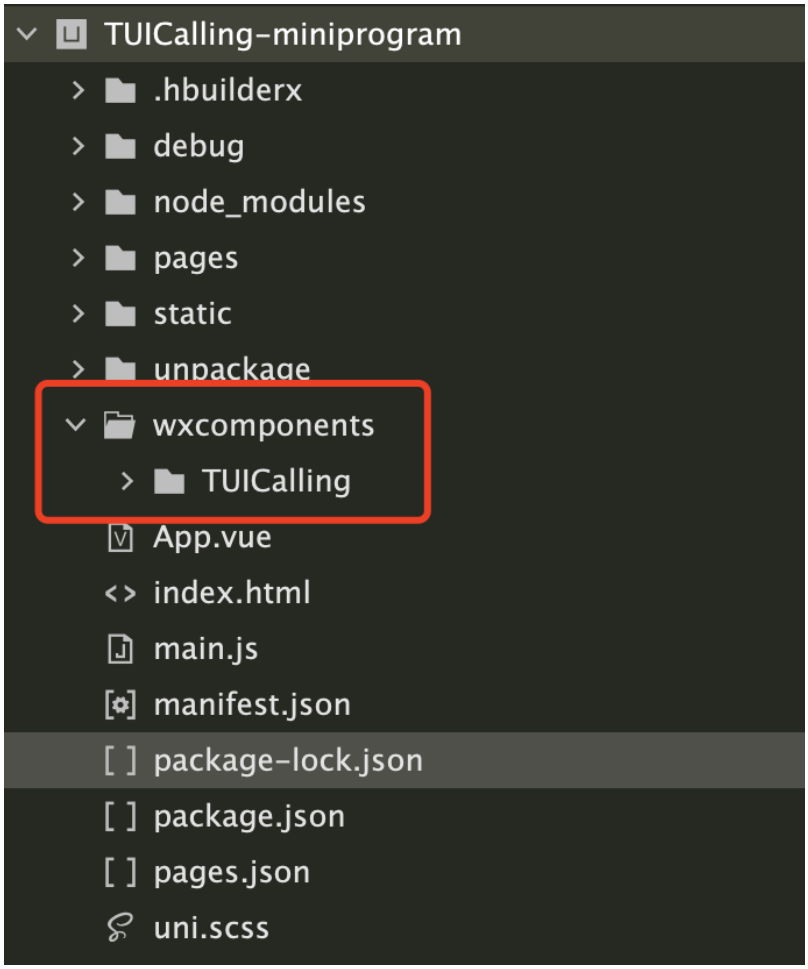
步骤二：下载并集成 TUICalling 组件

1. 根据您的实际业务需求，下载 TUICalling 组件。

```
# 命令行执行
git clone https://gitee.com/cloudtencent/TIMSDK

# 进入 uni-app TUICalling 项目
cd TIMSDK/uni-app/TUICalling/TUICalling-miniprogram
```

2. 将项目中的 wxcomponents 中的 TUICalling 组件复制到自己项目的 wxcomponents 中。如果没有 wxcomponents文件，请将 wxcomponents 复制到自己的项目中。



步骤三：获取 SDKAppID 和签名密钥

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：创建并配置组件

1. **pages 配置组件：**
   在 pages.json 文件中，呼叫音视频页面配置组件。

```
{
"usingComponents": {
"tuicalling": "/wxcomponents/TUICalling/TUICalling"
}
}
```

> **注意：**
> 组件名称均是小写字母。

2. **在页面的 xml 中引入组件：**

```
<tuicalling
ref="TUICalling"
id="TUICalling-component"
:config="config">
</tuicalling>
```

> ⚠ 注意：
>
> 组件的名称要与 page.json 中的保持一致，组件名称均是小写字母。

3. **填写 TIM 实例（如果您没有创建 TIM 实例，可忽略）：**

在 app.vue 中，如果您已创建了 TIM，请将 TIM 实例挂载在 wx 上，且不可以修改 wx.$TIM。

> ⚠ 注意：
>
> - TIM 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性。
> - 填写后，不可以修改 wx.$TIM（修改变量可能导致 TUICalling 组件无法正常使用）。

```
onLaunch() {
    // 重点注意：为了 uni-app 更好地接入使用 tim，快速定位和解决问题，请勿修改 uni.$TUIKit 命名。
    // 如果您已经接入 tim ，请将 uni.tim 修改为 uni.$TUIKit。
    uni.$TUIKit = TIM.create({
      SDKAppID: this.globalData.SDKAppID
    });
    // 如果您已创建了 tim，请将 tim 实例挂载在 wx 上，且不可以修改 wx.$TIM（修改变量可能导致 TUICalling 组件无法正常使用）。
    // 如果您没有创建，可以不传
    wx.$TIM = uni.$TUIKit
},
```

不可修改 wx.$TIM

4. **填写 config 配置信息：**

```
config = {
sdkAppID: 0, // 开通实时音视频服务创建应用后分配的 SDKAppID
userID: 'user0', // 用户 ID，可以由您的帐号系统指定
userSig: 'xxxxxxxxxxxx', // 身份签名，相当于登录密码的作用
type: 2, // 通话模式
}
```

步骤五：初始化 TUICalling 组件

```
// 将初始化后到 TUICalling 实例注册到 this.TUICalling。
this.TUICalling = this.$refs.TUICalling;
// 初始化 TUICalling
this.$nextTick(() => {
this.TUICalling.init()
})
```

步骤六： 进行通话

- **双人通话**

```
this.TUICalling.call({ userID: 'user1', type:2 })
```

- **多人通话**

```
this.TUICalling.groupCall({ userIDList: ['user1','user2'], type: 2 })
```

步骤七： 回收 TUICalling

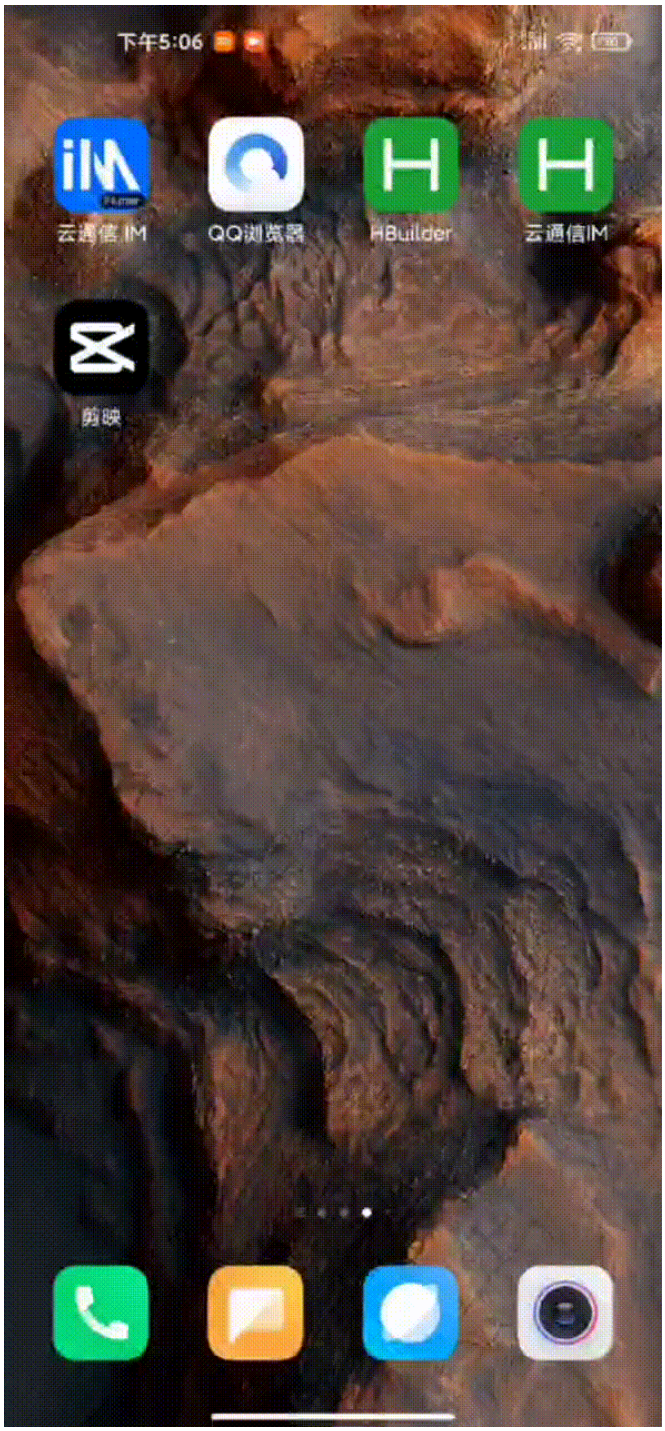
```
// 回收 TUICalling
this.TUICalling.destroyed()
```

实现案例

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

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

Android 演示	iOS 演示

Android 演示	iOS 演示
	

技术咨询

了解更多详情您可 QQ 咨询：

- **309869925** (uni-app 技术交流群)
- **646165204**（TUICalling 技术交流群)

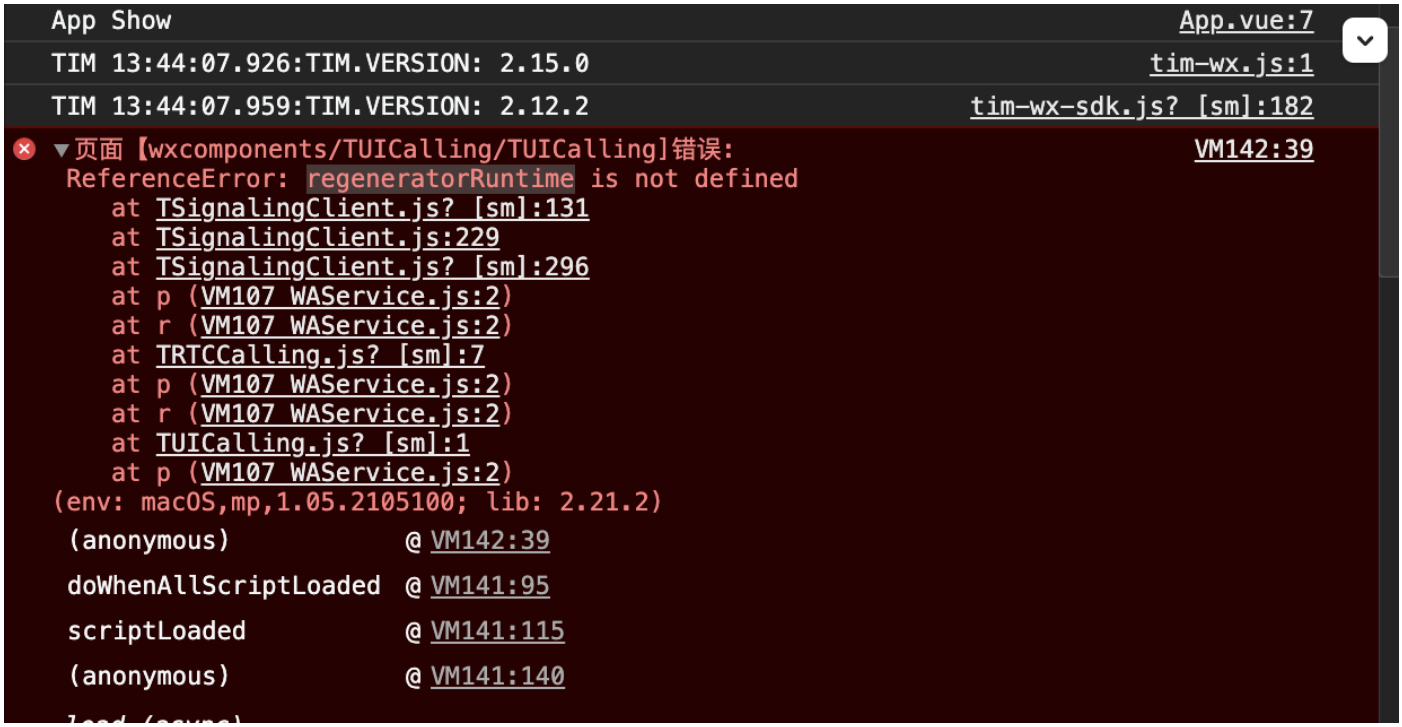
参考文档

- 小程序音视频 TUICalling 组件源码
- 集成 TUICalling (小程序)
- 小程序端相关问题

常见问题

调试中出现以下错误怎么办?

- **错误1:**



解决方法:请勾选开发者工具中的增强编译,再调试。

- **错误2:**



解决方法:请在 manifest.json 的文件中填写小程序 APPID。

多人视频会议
集成 TUIRoom (Web)

最近更新时间：2022-06-02 15:22:05

组件介绍

TUIRoom 是一个包含 UI 的开源音视频组件，通过集成 TUIRoom，您可以在业务中快速上线音视频房间，屏幕分享，聊天等功能。Web 端 TUIRoom 基础功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。





您可以单击 TUIRoom 在线链接 体验 TUIRoom 更多功能。
您可以单击 Github 下载 TUIRoom 代码，并参考 TUIRoom Web 示例工程快速跑通 文档跑通 TUIRoom Web 示例工程。
如需在现有业务中集成 Web 端 TUIRoom 组件，请参考本文档。

组件集成

TUIRoom 组件使用 Vue3 + TS + Pinia + Element Plus + SCSS 开发，要求接入项目使用 Vue3 + TS 技术栈。

步骤一： 开通腾讯云实时音视频及即时通信服务

TUIRoom 基于腾讯云实时音视频和即时通信服务进行开发。

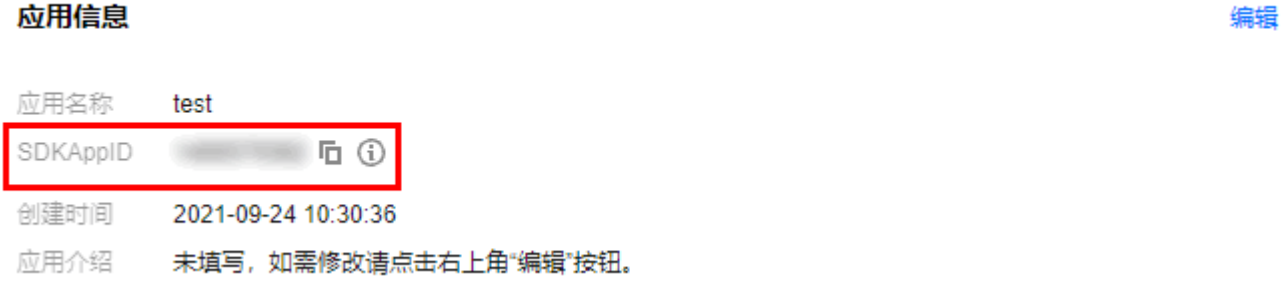
1. 创建实时音视频 TRTC 应用

- 如果您还没有腾讯云账号，请 注册腾讯云账号 ，并完成 实名认证。
- 在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。



2. 获取 TRTC 应用及密钥信息

i. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。SDKAppID 为 TRTC 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话不能互通；



ii. 在 **应用管理 > 快速上手** 中获取应用的 secretKey 信息。SecretKey 为 TRTC 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 TRTC 服务的鉴权用票据 UserSig。



3. 签发 UserSig

UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。TUIRoom 初始化需要您提供 UserSig 参数。

- 调试跑通阶段签发 userSig 的方式请参见 调试跑通阶段如何计算 UserSig。
- 生产环境签发 userSig 的方式参见 正式运行阶段如何计算 UserSig。

步骤二：下载并拷贝 TUIRoom 组件

1. 单击 Github，克隆或下载 TUIRoom 仓库代码。
2. 打开业务侧已有 Vue3 + TS 项目，如果无 Vue3 + TS 项目可通过以下脚本生成 Vue 3.0 + VITE + TS 的模板工程。

```
npm create vite@latest TUIRoom-demo -- --template vue
```

> ⚠ 注意：
> 执行生成模板工程脚本的过程中，第一步直接回车，第二步选择 Vue，第三步选择 vue-ts。

成功生成模板工程后，执行以下脚本：

```
cd TUIRoom-demo
npm install
npm run dev
```

3. 复制 TUIRoom/Web/src/TUIRoom 文件夹到已有项目 src/ 目录下。

步骤三：引用 TUIRoom 组件

1. 在页面中引用 TUIRoom 组件。例如：在 App.vue 组件中引入 TUIRoom 组件。
   - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
   - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
<room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';
```

```
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
const TUIRoomRef = ref();

onMounted(async () => {
// 初始化 TUIRoom 组件
// 主持人在创建房间前需要先初始化 TUIRoom 组件
// 普通成员在进入房间前需要先初始化 TUIRoom 组件
await TUIRoomRef.value.init({
// 获取 sdkAppId 请您参考 步骤一
sdkAppId: 0,
// 用户在您业务中的唯一标示 Id
userId: '',
// 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig,
注意 userSig 与 userId 为一一对应关系
userSig: '',
// 用户在您业务中使用的昵称
userName: '',
// 用户在您业务中使用的头像链接
userAvatar: '',
// 用户用于屏幕分享的唯一 Id，要求 shareUserId = `share_${userId}`, 无屏幕分享功能需求可不
传入
shareUserId: '',
// 请您参考本文 步骤一 > 第三步 并使用 sdkAppId 及 shareUserId 签发 shareUserSig
shareUserSig: '',
})
// 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
await handleCreateRoom();
})

// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
// roomId 为用户进入的房间号, 要求 roomId 类型为 number
// roomMode 包含'FreeSpeech'(自由发言模式) 和'ApplySpeech'(举手发言模式) 两种模式，默认
为'FreeSpeech'，注意目前仅支持自由发言模式
// roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒
体设备Id)
const roomId = 123456;
const roomMode = 'FreeSpeech';
```

```
const roomParam = {
isOpenCamera: true,
isOpenMicrophone: true,
}
await TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
// roomId 为用户进入的房间号, 要求 roomId 类型为 number
// roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
const roomId = 123456;
const roomParam = {
isOpenCamera: true,
isOpenMicrophone: true,
}
await TUIRoomRef.value.enterRoom(roomId, roomParam);
}
</script>

<style>
html, body {
width: 100%;
height: 100%;
margin: 0;
}

#app {
width: 100%;
height: 100%;
}
</style>
```

注意：

在页面中复制以上代码之后，需要修改 TUIRoom 接口的参数为实际数据。

步骤四：配置开发环境

TUIRoom 组件引入之后，为了确保项目可以正常运行，需要进行以下配置：

1. **安装依赖**
   - 安装开发环境依赖：

```
npm install sass typescript unplugin-auto-import unplugin-vue-components -S -D
```

   - 安装生产环境依赖：

```
npm install element-plus events mitt pinia rtc-beauty-plugin tim-js-sdk trtc-js-sdk tsignaling -S
```

2. **注册 Pinia**

   TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 src/main.ts 文件。

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

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

3. **配置 svg-icon 组件加载 SVG 图标**

   TUIRoom 将所有图标资源（SVG 文件）放置在 src/TUIRoom/assets/icons/svg 目录下，需要您在 vite.config.ts 文件中配置 SVG 图标加载方式。

> ⚠ 注意：
>
> 以下配置项为增量配置，不要删除已经存在的 Vite 配置项。

```
// vite.config.ts
import { createSvg } from './src/TUIRoom/assets/icons/index';

export default defineConfig({
// ...
```

```
plugins: [
// ...
createSvg('./src/TUIRoom/assets/icons/svg/'),
],
// ...
});
```

4. **配置 element-plus 按需引入**
   - TUIRoom 使用 element-plus UI 组件，为避免引入所有 element-plus组件，需要您在 vite.config.ts 中配置 element-plus 组件按需加载。

> ⚠ **注意：**
> 以下配置项为增量配置，不要删除已经存在的 Vite 配置项。

```
// vite.config.ts
import AutoImport from 'unplugin-auto-import/vite';
import Components from 'unplugin-vue-components/vite';
import { ElementPlusResolver } from 'unplugin-vue-components/resolvers';

export default defineConfig({
// ...
plugins: [
AutoImport({
resolvers: [ElementPlusResolver()],
}),
Components({
resolvers: [ElementPlusResolver({
importStyle: 'sass',
})],
}),
],
css: {
preprocessorOptions: {
scss: {
// ...
additionalData: `
```

```
@use "./src/TUIRoom/assets/style/element.scss" as *;
`,
},
},
},
});
```

- 同时为了保证 element-plus 带 UI 组件能够正常显示样式，需要您在入口文件 src/main.ts 中加载 element-plus 组件样式。

```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css';
import 'element-plus/theme-chalk/el-message-box.css';
```

步骤五：开发环境运行

在控制台执行开发环境运行脚本，使用浏览器打开包含 TUIRoom 的页面，即可在页面中使用 TUIRoom 组件。

如果您是使用 步骤二 中的脚本生成 Vue + TS 项目，您需要：

1. 执行开发环境命令。

```
npm run dev
```

2. 在浏览器中打开页面 http://localhost:3000/。

> **注意：**
> 因 TUIRoom 按需引入 element-plus 组件，会导致开发环境路由页面第一次加载时反应较慢，等待 element-plus 按需加载完成即可正常使用。element-plus 按需加载不会影响打包之后的页面加载。

3. 体验 TUIRoom 组件功能。

附录：TUIRoom API

TUIRoom 接口

init

初始化 TUIRoom 数据，任何使用 TUIRoom 的用户都需要调用 init 方法。

```
TUIRoomRef.value.init(roomData);
```

参数如下表所示：

参数	类型	含义
roomData	object	
roomData.sdkAppId	number	客户的 SDKAppId
roomData.userId	string	用户的唯一 ID
roomData.userSig	string	用户的 UserSig
roomData.userName	string	用户的昵称
roomData.userAvatar	string	用户的头像
roomData.shareUserId	string	非必填，用户进行屏幕分享的 UserId，要求 shareUserId = share_${userId}，无屏幕分享功能可不传入
roomData.shareUserSig	string	非必填，用户进行屏幕分享的 UserSig

createRoom

主持人创建房间。

```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	number	房间 ID
roomMode	string	房间模式，'FreeSpeech'（自由发言模式）和 'ApplySpeech'（举手发言模式），默认为 'FreeSpeech'，注意目前仅支持自由发言模式
roomParam	Object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭

参数	类型	含义
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	String	非必填，默认扬声器设备 ID

enterRoom

普通成员加入房间。

```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	number	房间 ID
roomParam	Object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	String	非必填，默认扬声器设备 ID

TUIRoom 事件

onRoomCreate

创建房间回调。

```
<template>
<room ref="TUIRoomRef" @on-room-create="handleRoomCreate"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
```

```
import Room from './TUIRoom/index.vue';

function handleRoomCreate(info) {
if （info.code === 0) {
console.log('创建房间成功')
}
}
</script>
```

onRoomEnter

进入房间回调。

```
<template>
<room ref="TUIRoomRef" @on-room-enter="handleRoomEnter"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomEnter(info) {
if （info.code === 0) {
console.log('进入房间成功')
}
}
</script>
```

onRoomDestory

主持人销毁房间通知。

```
<template>
<room ref="TUIRoomRef" @on-room-destory="handleRoomDestory"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
```

版权所有：腾讯云计算（北京）有限责任公司

```
import Room from './TUIRoom/index.vue';

function handleRoomDestory(info) {
if （info.code === 0) {
console.log('主持人销毁成功')
}
}
</script>
```

onRoomExit

普通成员退出房间通知。

```
<template>
<room ref="TUIRoomRef" @on-room-exit="handleRoomExit"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomExit(info) {
if （info.code === 0) {
console.log('普通成员退出房间成功')
}
}
</script>
```

集成 TUIRoom (Android)

最近更新时间：2022-06-23 17:13:00

组件介绍

TUIRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIRoom 组件，您只需要编写几行代码就可以为您的 App 添加屏幕分享、美颜、低延时视频通话等。TUIRoom 同时支持 iOS、Windows，Mac 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIRoom 组件

单击进入 Github，选择克隆/下载代码，然后拷贝 Android 下的 tuiroom、debug、tuibeauty 目录到您的工程中，并完成如下导入动作：

1. 在 setting.gradle 中完成导入，参考如下：

```
include ':tuiroom'
include ':debug'
include ':tuibeauty'
```

2. 在 app 的 build.gradle 文件中添加对 tuiroom、debug、tuibeauty 的依赖：

```
api project(':tuiroom')
api project(':debug')
api project(':tuibeauty')
```

3. 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

1. 在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请麦克风权限等）：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.CAMERA" />
<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.** { *; }
```

版权所有：腾讯云计算（北京）有限责任公司

步骤三：创建并初始化 TUI 组件库

```
// 1.组件登录
TUILogin.addLoginListener(new TUILoginListener() {
@Override
public void onKickedOffline() { // 登录被踢下线通知（示例：账号再其他设备登录）
}

@Override
public void onUserSigExpired() { // userSig过期通知
}
});

TUILogin.login(context, "您的SDKAppId", "您的userId", "您的userSig", null);


// 2.初始化TUIRoom实例
TUIRoom tuiRoom = TUIRoom.sharedInstance(this);
```

参数说明

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的userSig，也可以参照我们的 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现多人音视频互动

1. **实现房主创建多人音视频互动房间。**

```
tuiRoom.createRoom(12345, TUIRoomCoreDef.SpeechMode.FREE_SPEECH, true, true);
```

2. **实现其他成员加入音视频房间。**

```
tuiRoom.enterRoom(12345, true, true);
```

步骤五：房间管理（可选）

1. **房主解散房间 TUIRoomCore#destroyRoom。**

```
// 1.房主调用解散房间
mTUIRoomCore.destroyRoom(new TUIRoomCoreCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
```

```
}
});

成员端会收到 onDestroyRoom 回调消息，通知房间解散
@Override
public void onDestroyRoom() {
//房主解散，退出房间
}
```

2. 成员离开房间 TUIRoomCore#leaveRoom。

```
// 1.非房主身份调用离开房间
mTUIRoomCore.leaveRoom(new TUIRoomCoreCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {

}
});

//成员端会收到 onRemoteUserLeave 回调消息，通知有人离开
@Override
public void onRemoteUserLeave(String userId) {
Log.d(TAG, "onRemoteUserLeave userId: " + userId);
}
```

步骤六：屏幕分享（可选）

实现屏幕分享 TUIRoomCore#startScreenCapture。

```
// 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" />
</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("packag
e:" + getPackageName()));
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);
}
```

步骤七：美颜特效（可选）

TUIRoom 美颜使用了 腾讯特效SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License申请指引。

```
TUIBeautyView.getBeautyService().setLicense(context, “XMagicLicenseURL”, “XMagicLicenseKey”);
```

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIRoom (iOS)

最近更新时间：2022-06-23 17:13:03

组件介绍

TUIRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIRoom 组件，您只需要编写几行代码就可以为您的 App 添加屏幕分享、美颜、低延时视频通话等。TUIRoom 同时支持 Android、Windows，Mac 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：导入 TUIRoom 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 在您的工程 Podfile 文件同一级目录下创建 TUIRoom 文件夹。
2. 单击进入 Github/TUIRoom ，选择克隆/下载代码，然后将 TUIRoom/iOS/ 目录下的 Source、Resources 、TUIBeauty、TXAppBasic文件夹 和 TUIRoom.podspec 文件拷贝到您在 步骤1 创建的 TUIRoom 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖，之后执行 pod install 命令，完成导入。

```
# :path => "指向TUIRoom.podspec的相对路径"
pod 'TUIRoom', :path => "./TUIRoom/TUIRoom.podspec", :subspecs => ["TRTC"]
# :path => "指向TXAppBasic.podspec的相对路径"
pod 'TXAppBasic', :path => "./TUIRoom/TXAppBasic/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIBeauty', :path => "./TUIRoom/TUIBeauty/"
```

⚠ 注意：

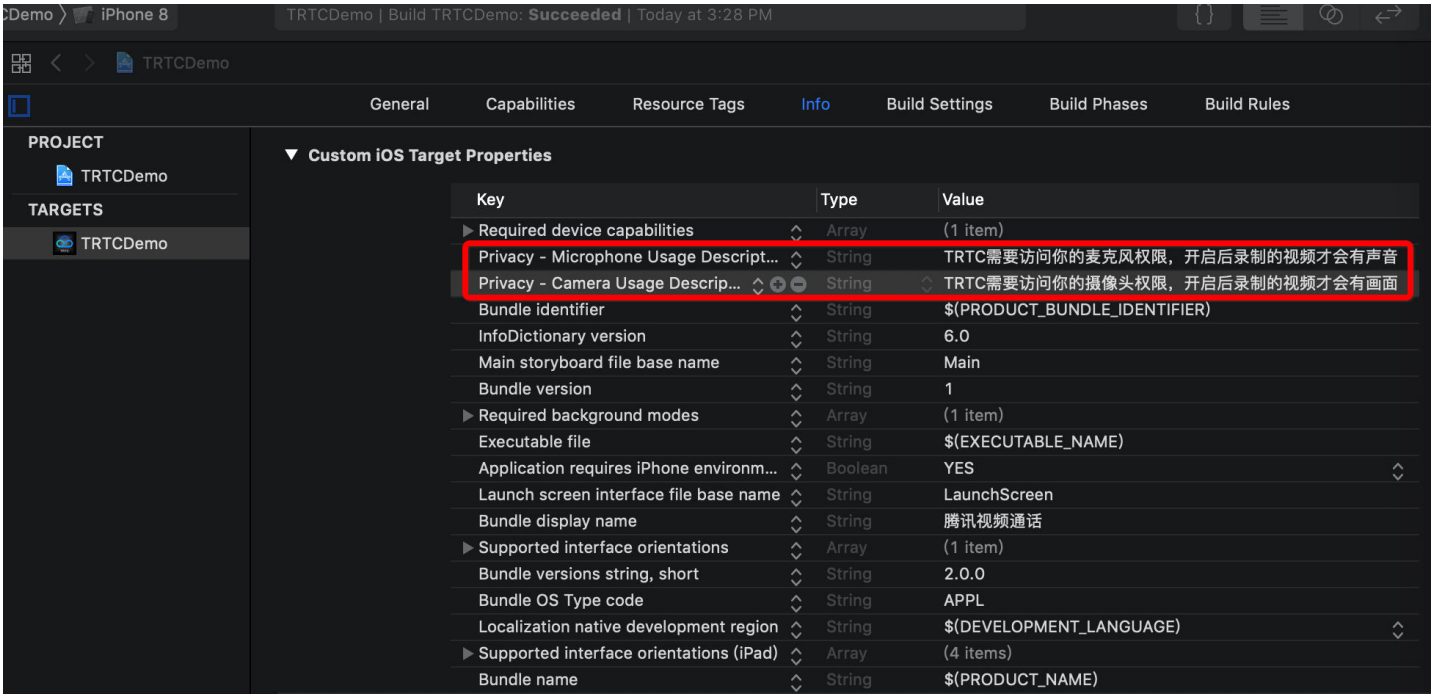
Source 、 Resources 文件夹 和 TUIRoom.podspec 文件必需在同一目录下。

- TXAppBasic.podspec 在 TXAppBasic 文件夹下。
- TUIBeauty.podspec 在 TCBeautyKit 文件夹下。

步骤二：配置权限

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

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



步骤三：创建并初始化 TUI 组件库

Objective-C

```
@import TUIRoom;
@import TUICore;

// 1.组件登录
[TUILogin login:@"您的SDKAppID" userID:@"您的UserID" userSig:@"您的UserSig" succ:^{

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

}];
// 2.初始化TUIRoom实例
TUIRoom *tuiRoom = [TUIRoom sharedInstance];
```

Swift

```
import TUIRoom
import TUICore

// 1.组件登录
TUILogin.login("您的SDKAppID", userID: "您的UserID", userSig: "您的UserSig") {

} fail: { code, msg in
```

```
}

// 2.初始化TUIRoom实例
let tuiRoom = TUIRoom.sharedInstance
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppID 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **UserID**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **UserSig**：根据 SDKAppId、UserID，SecretKey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUIRoom示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现多人音视频互动

1. **实现房主创建多人音视频互动房间。Objective-C**

```
@import TUIRoom;
```

```
[tuiRoom createRoomWithRoomId:12345 speechMode:TUIRoomFreeSpeech isOpenCamera:YES isOpenMicrophone:YES];
```

Swift

```
import TUIRoom

tuiRoom.createRoom(roomId: 12345, speechMode: .freeSpeech, isOpenCamera: true, isOpenMicrophone: true)
```

2. **实现其他成员加入音视频房间。Objective-C**

```
@import TUIRoom;

[tuiRoom enterRoomWithRoomId:12345 isOpenCamera:YES isOpenMicrophone:YES]
```

Swift

```
import TUIRoom

tuiRoom.enterRoom(roomId: 12345, isOpenCamera: true, isOpenMicrophone: true)
```

步骤五：房间管理（可选）

1. **房主解散房间 TUIRoomCore#destroyRoom。Objective-C**

```
@import TUIRoom;

[[TUIRoomCore shareInstance] destroyRoom:^(NSInteger code, NSString * _Nonnull message) {

}];
```

Swift

```
import TUIRoom

TUIRoomCore.shareInstance().destroyRoom { [weak self] _, _ in
```

```
guard let self = self else { return }
self.navigationController?.popViewController(animated: true)
}
```

2. 成员离开房间 TUIRoomCore#leaveRoom。Objective-C

```
@import TUIRoom;

[[TUIRoomCore shareInstance] leaveRoom:^(NSInteger code, NSString * _Nonnull messag
e) {

}];
```

Swift

```
import TUIRoom

TUIRoomCore.shareInstance().leaveRoom { [weak self] _, _ in
guard let self = self else { return }
self.navigationController?.popViewController(animated: true)
}
```

步骤六：屏幕分享（可选）

实现屏幕分享 TUIRoomCore#startScreenCapture。屏幕分享工程配置请参见 实时屏幕分享(iOS)。

Objective-C

```
@import TUIRoom;
@import TXLiteAVSDK_Professional;

TRTCVideoEncParam *params = [[TRTCVideoEncParam alloc] init];
params.videoResolution = TRTCVideoResolution_1280_720;
params.resMode = TRTCVideoResolutionModePortrait;
params.videoFps = 10;
params.enableAdjustRes = NO;
params.videoBitrate = 1500;

[[TUIRoomCore shareInstance] startScreenCapture:param];
```

Swift

```
import TUIRoom

// 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)
```

常见问题

CocoaPods 如何安装?

在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：

```
sudo gem install cocoapods
```

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIRoom (小程序)

最近更新时间：2022-06-23 17:13:07

组件介绍

TUIRoom 组件是一个开源的场景化解决方案，帮助您快速在您的小程序中集成**多人音视频房间**功能，非常适合用于**多人会议、视频面试 、在线课堂、远程签约**等场景中。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

创建房间

由于用户不需要安装 App 就可以使用该功能，因此 TUIRoom 非常适合用于快速发起多人音视频通话的场景。

房间管理

房间内部继承**低延迟、高质量**音视频通话功能，包含美颜面板，创建者可对房间内其余用户进行禁言、禁画以及踢出操作。

其它平台

除了小程序版的 TUIRoom，我们同时也推出了 Android、iOS、Flutter、windows/mac 等平台的源代码，用户可直接使用并对源码进行修改。

效果展示

您可以单击 Demo 体验 扫描二维码单击**多人音视频房间**进行体验，包括禁言禁画、推流美颜、低延时视频通话等 TRTC 在多人音视频场景下的相关能力。



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 Demo 进行修改适配。

环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.10.0。
- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在真机上进行运行体验。
- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，需要申请常规小程序账号进行开发。
- 不支持 uniapp 开发环境，请使用原生小程序开发环境。

前提条件

1. 您已 注册腾讯云 账号，并完成 实名认证。
2. **开通小程序类目与推拉流标签权限（如不开通则无法正常使用）。**
   出于政策和合规的考虑，微信暂未放开所有小程序对实时音视频功能（即 <live-pusher> 和 <live-player> 标签）的支持：
3. 小程序推拉流标签不支持个人小程序，只支持企业类小程序。

版权所有：腾讯云计算（北京）有限责任公司

4. 小程序推拉流标签使用权限暂时只开放给有限 类目。
5. 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置 中自助开通该组件权限，如下图所示：



跑通 Demo

步骤一：开通小程序权限

由于 TUIRoom 所使用的小程序标签有更苛刻的权限要求，因此集成 TUIRoom 的第一步就是要开通小程序的类目和标签使用权限，**否则无法使用**，这包括如下步骤：

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



- 小程序推拉流标签使用权限暂时只开放给有限 类目。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：



步骤二：在小程序控制台配置域名

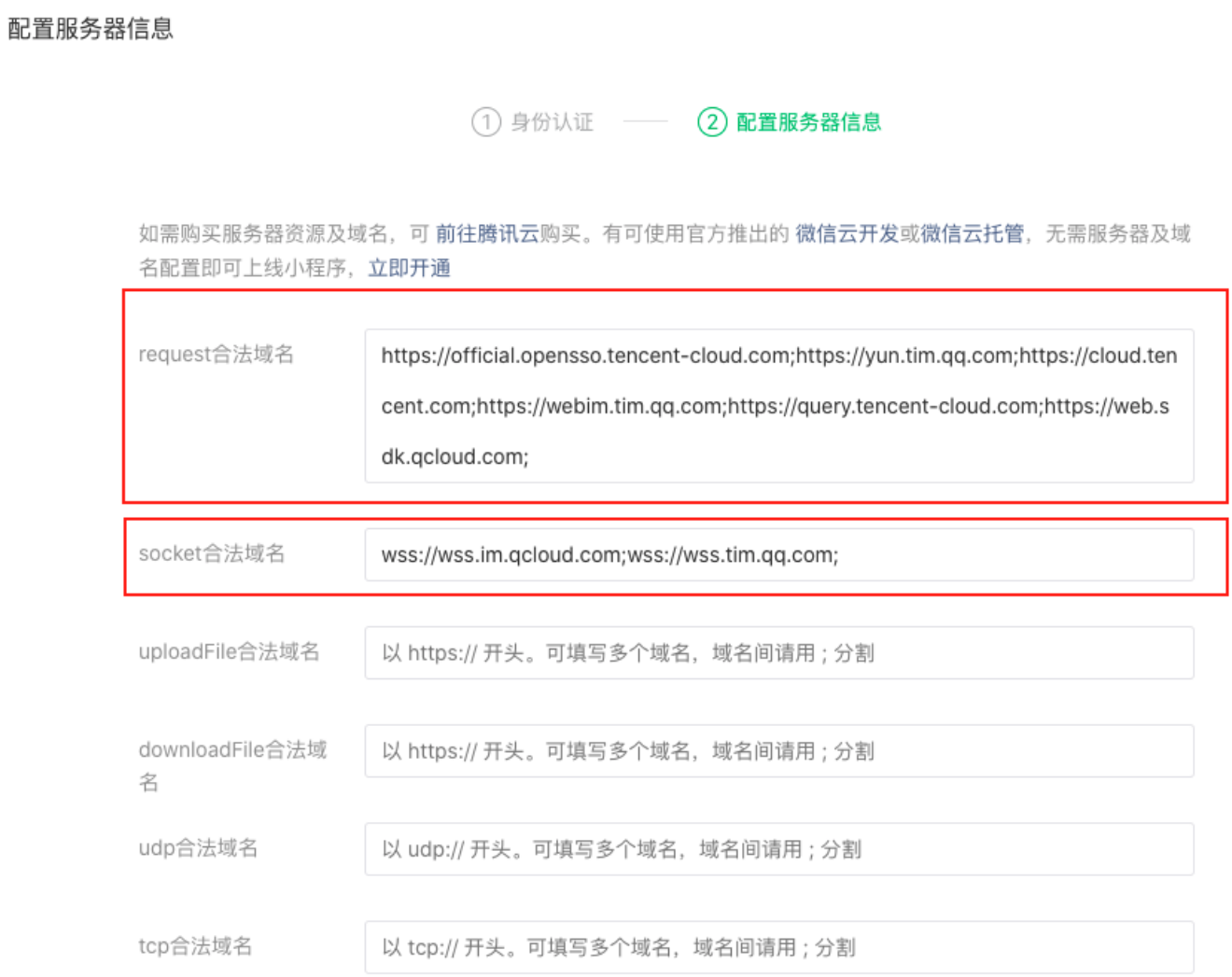
在 **微信公众平台** > **开发** > **开发管理** > **开发设置** > **服务器域名**中设置 **request合法域名**和 **socket合法域名**，如下图所示：

- request 合法域名：

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

- socket 合法域名：

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



步骤三：创建新的应用

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

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

步骤四：下载 App 源码

单击进入 TUIRoom，Clone 或者下载源码。

步骤五：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 ./debug/GenerateTestUserSig.js 文件。
3. 设置 GenerateTestUserSig.js 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。

```
function genTestUserSig(userID) {
  /**
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  var SDKAPPID    = 0;
  /**
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  var EXPIRETIME = 604800;

  /**
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一个，
   * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
   * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  var SECRETKEY = "";
```

将 SDKAppID 粘贴到此处

将 密钥 粘贴到此处

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

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

步骤六：编译运行

1. 打开微信开发者工具，选择**小程序**，单击新建图标，选择**导入项目**。

2. 填写您微信小程序的 AppID，单击**确定**。



> **注意：**
> 此处应输入您微信小程序的 AppID，而非 SDKAppID。

3. 在 project.config.json 文件中，修改 setting 下的 useCompilerPlugins 字段为 ["typescript", "sass"]。
4. 单击**预览**，生成二维码，通过手机微信扫码二维码即可进入小程序。

技术咨询

了解更多详情您可 QQ 咨询：941036374 *技术交流群*

参考文档

- 实时音视频 trtc-wx API 手册
- 即时通信 IM sdk API 手册
- 小程序 TUIVoiceRoom 源码
- 小程序 TUIRoom 源码
- 小程序如何支持 typescript + sass/less

集成 TUIRoom (Windows&Mac)

最近更新时间：2022-06-23 17:13:11

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。
您可以下载并安装 Windows 或者 Mac 平台的 App 进行体验。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

效果展示



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

为了让您快速接入多人音视频房间功能，这边有两种推荐的接入方式，您可以选择适合您的方式进行二次开发。

- 外部进程启动
- 实现自定义 UI 界面

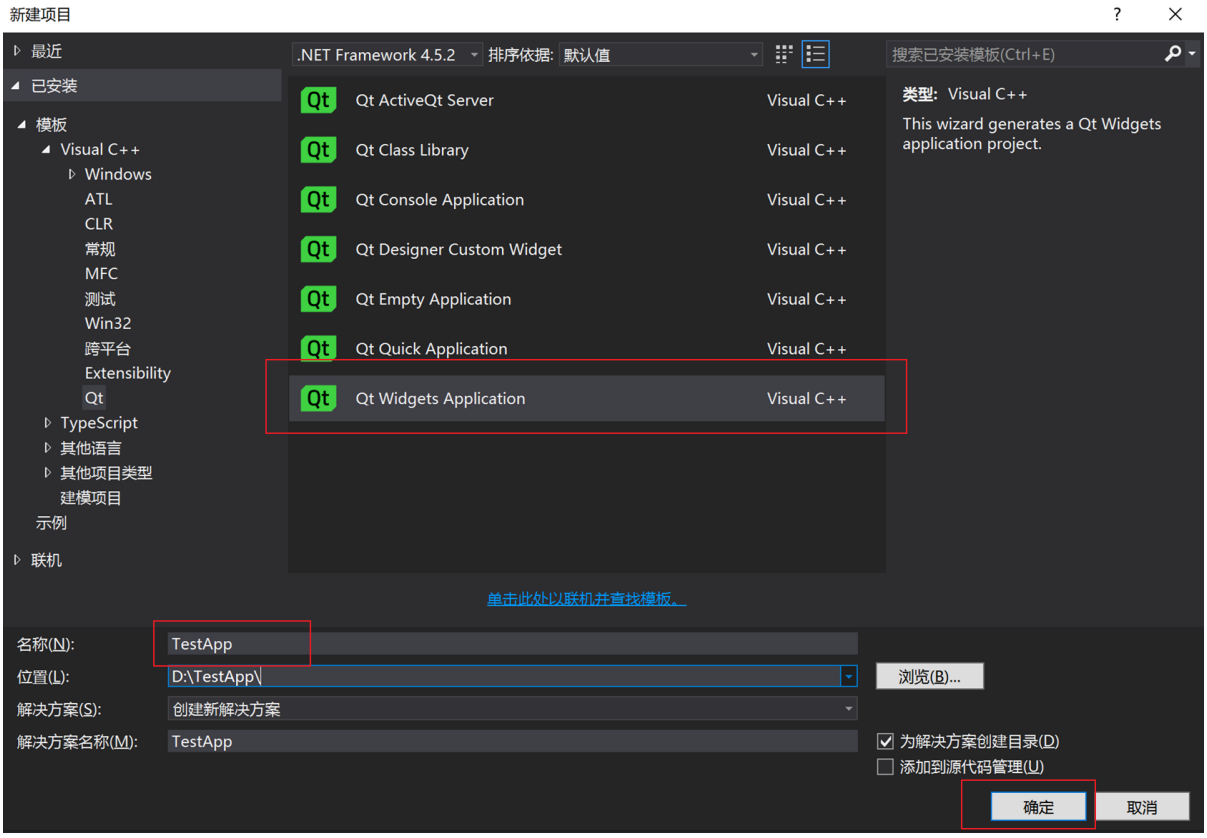
环境准备

- **Windows环境：**
  - Visual Studio 2015及以上集成开发环境。
  - QT5.9.1及以上版本的 QT 开发库。
  - VS 下的 QT 开发插件 Qt Visual Studio Tools 2.2.0及以上。
  - 最低支持系统：Windows 8。
  - 请确保您的集成开发环境能够正常开发。
- **Mac环境:**
  - QT5.9.1及以上版本的 QT 开发库。
  - QtCreator 集成开发环境，在安装 QT 时选择同时安装 QtCreator 即可，版本跟随 QT 官方安装包。
  - 请确保您的 QtCreator 集成开发环境能够正常开发。

外部进程启动

1. **编译 RoomApp 程序**
   - 使用外部进程启动 RoomApp 的方式，依赖原 RoomApp 的运行程序，需要提前进行编译。
   - 可以单击进入 RoomApp，Clone 源码，配置工程并编译生成 RoomApp。
2. **新建 TestApp 工程Windows 平台**

i. 打开 VS，选择 Qt Widgets Application 工程类型，创建 TestApp 工程。



ii. 编写启动进程的程序，并在合适的位置调用 LoadRoomApp 函数。

```
#include <QProcess>
#include <QApplication>
void LoadRoomApp() {
QString executable_file_path = QApplication::applicationDirPath();
QString app_path = executable_file_path + "/RoomApp.exe";
QProcess::startDetached(app_path);
}
```

```
#include "TestApp.h"

#include <QProcess>                    启动RoomApp
#include <QApplication>
void LoadRoomApp() {
    QString executable_file_path = QApplication::applicationDirPath();
    QString app_path = executable_file_path + "/RoomApp.exe";
    QProcess::startDetached(app_path);
}

TestApp::TestApp(QWidget *parent)
    : QMainWindow(parent) {
    ui.setupUi(this);
    LoadRoomApp();          TestApp启动的时候启动RoomApp
}
```

iii. 编译项目，并将 RoomApp 编译的成果物复制到当前可执行程序目录，以 release x86 程序为例：
复制 TUIRoom\Windows-Mac\RoomApp\bin\Win32\Release 目录下所有文件到当前程序目录下。

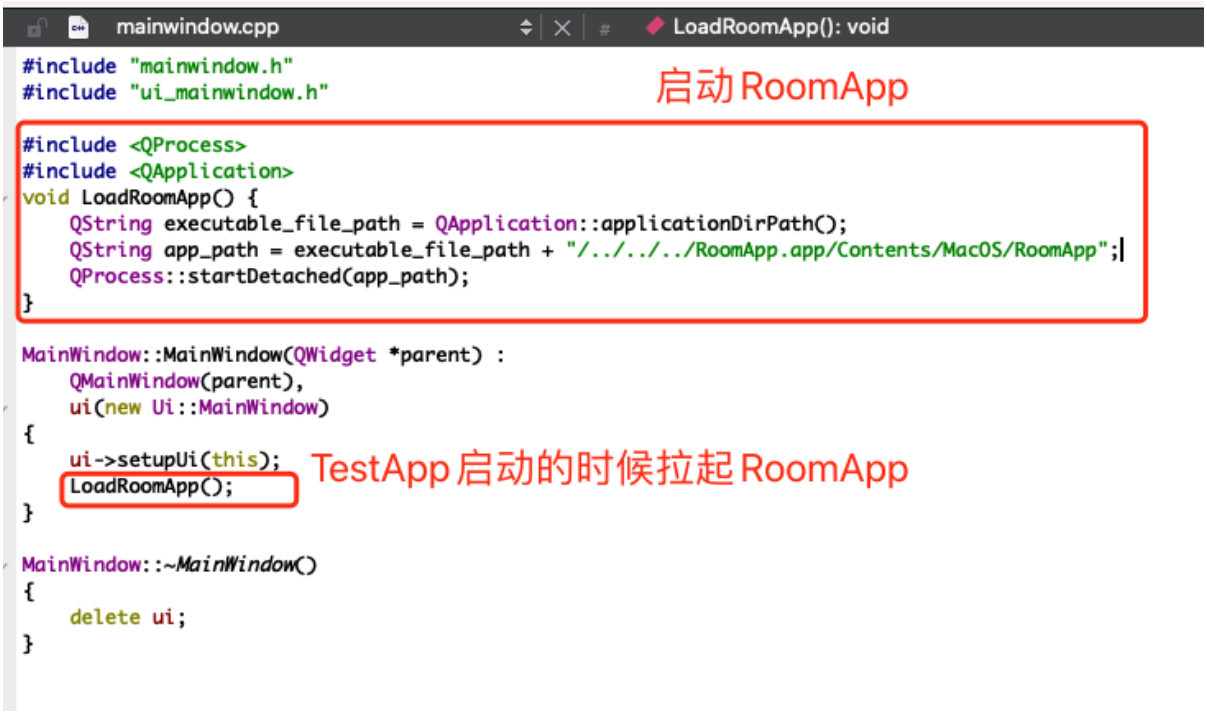
iv. 执行程序，启动 TestApp 的同时启动 RoomApp。

Mac 平台

i. 打开 QtCreator，选择 Qt Widgets Application 工程，新建项目，创建 TestApp 工程。

ii. 编写启动进程的程序，并在合适的位置调用 LoadRoomApp 函数。

```
#include <QProcess>
#include <QApplication>
void LoadRoomApp() {
QString executable_file_path = QApplication::applicationDirPath();
QString app_path = executable_file_path + "/../../../RoomApp.app/Contents/MacOS/RoomA
pp";
QProcess::startDetached(app_path);
}
```



iii. 编译项目，并将 RoomApp 编译的产物 RoomApp.app 拷贝到与当前项目的产物同级目录下，以上图创建的项目路径为例：
拷贝 RoomApp.app 到 /Users/mac/Desktop/TestApp/build-TestApp-Desktop_Qt_5_9_1_clang_64bit-Release 目录下。

iv. 执行程序，启动 TestApp 的同时启动 RoomApp。

实现自定义 UI 界面

- 您可以直接基于我们提供的 App 进行修改适配，也可以使用 App 源码中的 Module 模块实现自定义 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
- App 目录包含了 UI 相关的设计与逻辑，您可以根据需求，修改 RoomApp 源码进行二次开发，主要功能点如下：

功能点	文件目录
首页登录	Windows-Mac\RoomApp\App\LoginViewController.cpp
设备检测	Windows-Mac\RoomApp\App\PresetDeviceController.cpp
主页面	Windows-Mac\RoomApp\App\MainWindow.cpp
麦上列表	Windows-Mac\RoomApp\App\StageListController.cpp
成员列表	Windows-Mac\RoomApp\App\MemberListViewController.cpp
设置页面	Windows-Mac\RoomApp\App\SettingViewController.cpp

版权所有：腾讯云计算（北京）有限责任公司

功能点	文件目录
聊天室	Windows-Mac\RoomApp\App\ChatRoomViewController.cpp
屏幕分享	Windows-Mac\RoomApp\App\ScreenShareWindow.cpp
底部工具栏	Windows-Mac\RoomApp\App\BottomBarController.cpp

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIRoom (Flutter)

最近更新时间：2022-06-02 15:21:55

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。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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。



下载SDK及配套的Demo源码

平台	操作
iOS	打开GitHub链接 打开Gitee链接 下载Zip
Android	打开GitHub链接 打开Gitee链接 下载Zip
Web	打开GitHub链接 打开Gitee链接 下载Zip
MacOS	打开GitHub链接 打开Gitee链接 下载Zip
微信小程序	打开GitHub链接 打开Gitee链接 下载Zip
Electron	打开GitHub链接 打开Gitee链接 下载Zip
Windows	打开GitHub链接 打开Gitee链接 下载Zip
Flutter	打开GitHub链接

已下载，下一步 上一步

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。

```
class GenerateTestUserSig {
  /*
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  static int sdkAppId = 0;    将SDKAppID粘贴到此处

  /*
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  static int expireTime = 604800;

  /*
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc )，如果还没有应用就创建一个，
   * step2. 单击"应用配置"进入基础配置页面，并进一步找到"帐号体系集成"部分。
   * step3. 点击"查看密钥"按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  static String secretKey = '';    将 密钥 粘贴到此处
```

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

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

步骤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**
   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。
   Android 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。

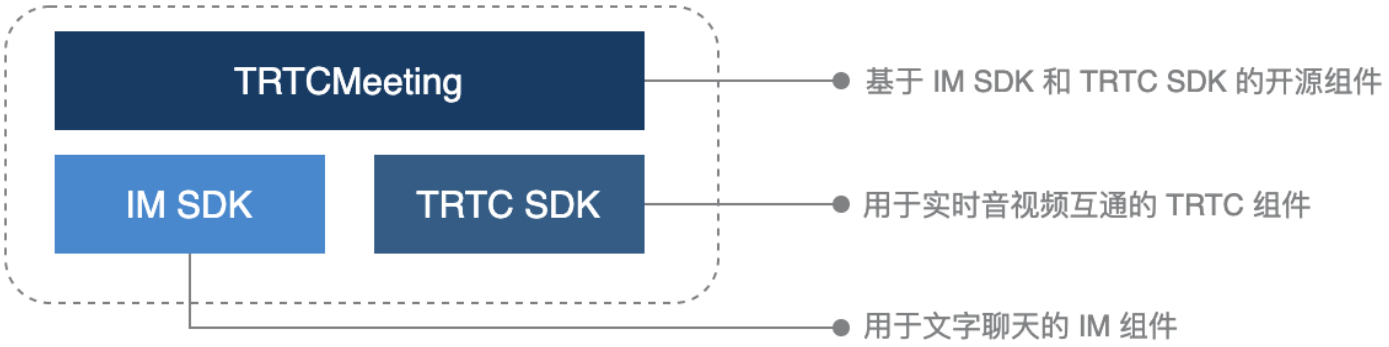
步骤5：修改 Demo 源代码

源码中的 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

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>
```

Android 端

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

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

android > app > src > main > AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mlp">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        tools:replace="android:label"
        android:name="io.flutter.app.FlutterApplication"
        android:label="mlp"
        android:icon="@mipmap/ic_launcher">
```

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：

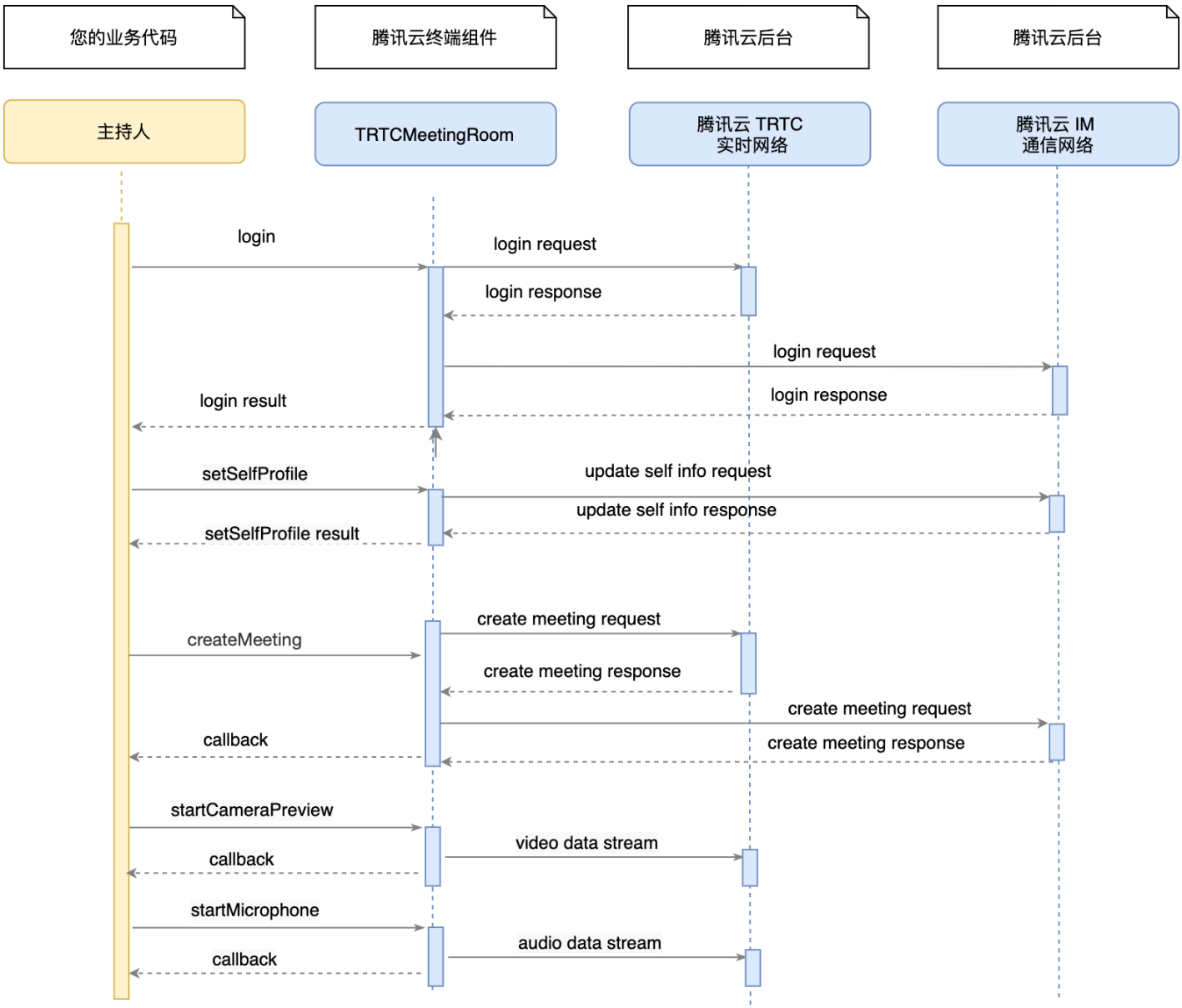
参数名	作用
sdkAppId	您可以在 实时音视频控制台 中查看 SDKAppID。
userId	当前用户的 ID，字符串类型，只允许包含英文字母（a-z、A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
userSig	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算及使用 UserSig。

```
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。



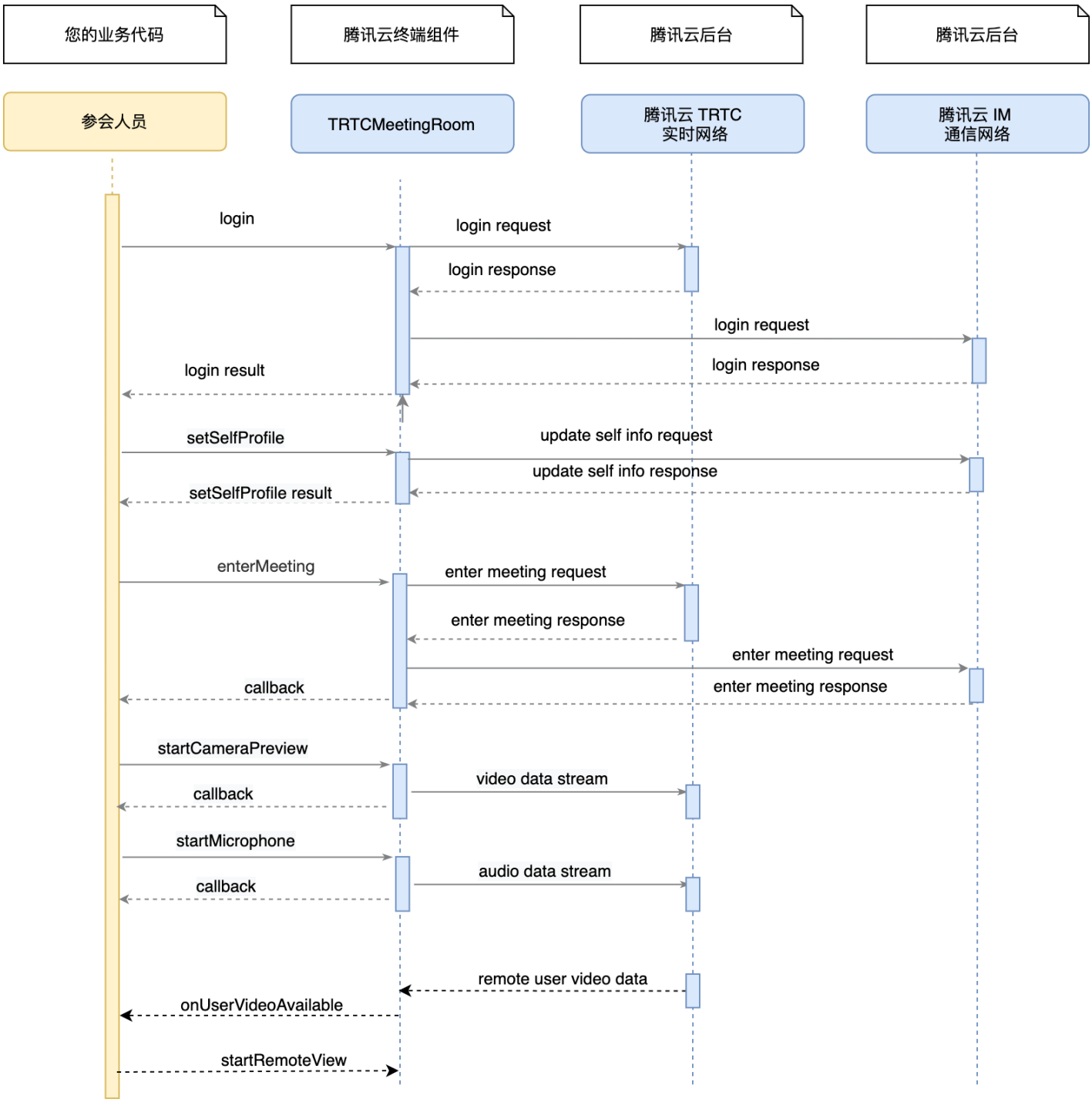

```
// 1. 主持人设置昵称和头像
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 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:
```

版权所有：腾讯云计算（北京）有限责任公司

```
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;
}
}
```

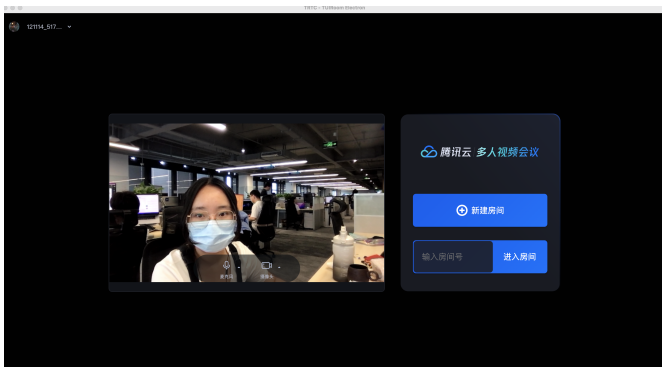
集成 TUIRoom (Electron)

最近更新时间：2022-06-17 18:28:11

组件介绍

TUIRoom 是一个包含 UI 的开源音视频组件，通过集成 TUIRoom，您可以在业务中快速上线音视频房间，屏幕分享，聊天等功能。Electron 端 TUIRoom 基础功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。





您可以单击 Github 下载 TUIRoom 代码，并参考 TUIRoom Electron 示例工程快速跑通 文档跑通 TUIRoom Electron 示例工程。
如需在现有业务中集成 Electron 端 TUIRoom 组件，请参考本文档。

组件集成

TUIRoom 组件使用 Vue3 + TS + Pinia + Element Plus + SCSS 开发，要求接入项目使用 Electron + Vue3 + TS 技术栈。

步骤一： 开通腾讯云实时音视频及即时通信服务

TUIRoom 基于腾讯云实时音视频和即时通信服务进行开发。

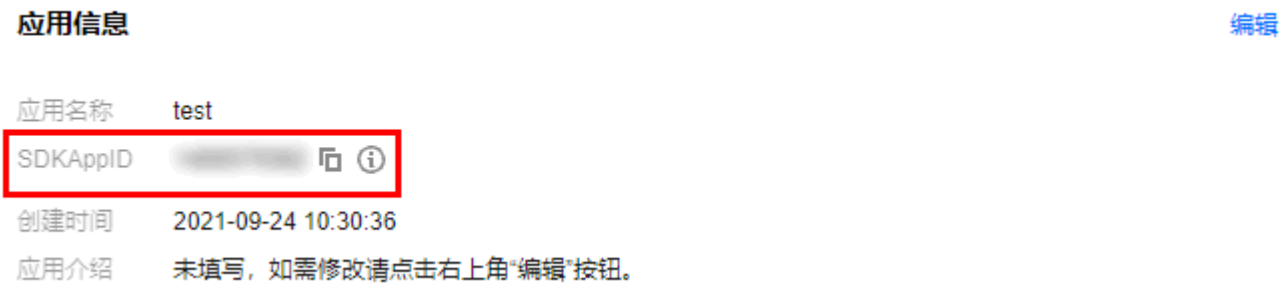
1. **创建实时音视频 TRTC 应用**
   - 如果您还没有腾讯云账号，请 注册腾讯云账号 ，并完成 实名认证。

- 在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。



2. 获取 TRTC 应用及密钥信息

i. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。SDKAppID 为 TRTC 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话不能互通；



ii. 在 **应用管理 > 快速上手** 中获取应用的 secretKey 信息。SecretKey 为 TRTC 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 TRTC 服务的鉴权用票据 UserSig。



3. 签发 UserSig

UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。TUIRoom 初始化需要您提供 UserSig 参数。

- 调试跑通阶段签发 userSig 的方式请参见 调试跑通阶段如何计算 UserSig。

- 生产环境签发 userSig 的方式参见 正式运行阶段如何计算 UserSig。

步骤二：下载并拷贝 TUIRoom 组件

1. 打开业务侧已有 Electron + Vue3 + TS 项目，如果无 Electron + Vue3 + TS 项目,可通过以下脚本生成 Electron + Vue3 + TS 的模板工程。

```
npm create electron-vite
```

> ⚠ 注意：
> 执行生成模板工程脚本的过程中，选择 Vue。

成功生成模板工程后，执行以下脚本：

```
cd electron-vite-vue
npm install
npm run dev
```

2. 单击 Github , 克隆或下载 TUIRoom 仓库代码，复制 TUIRoom/Electron/packages/renderer/src/TUIRoom 文件夹到已有项目 packages/renderer/src/ 目录下。

步骤三：引用 TUIRoom 组件

1. 在页面中引用 TUIRoom 组件。例如：在 App.vue 组件中引入 TUIRoom 组件。
   - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
   - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
<room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
```

```
const TUIRoomRef = ref();

onMounted(async () => {
// 初始化 TUIRoom 组件
// 主持人在创建房间前需要先初始化 TUIRoom 组件
// 普通成员在进入房间前需要先初始化 TUIRoom 组件
await TUIRoomRef.value.init({
// 获取 sdkAppId 请您参考 步骤一
sdkAppId: 0,
// 用户在您业务中的唯一标示 Id
userId: '',
// 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig,注意 userSig 与 userId 为一一对应关系
userSig: '',
// 用户在您业务中使用的昵称
userName: '',
// 用户在您业务中使用的头像链接
userAvatar: '',
// 用户用于屏幕分享的唯一 Id，要求 shareUserId = `share_${userId}`, 无屏幕分享功能需求可不传入
shareUserId: '',
// 请您参考本文 步骤一 > 第三步 并使用 sdkAppId 及 shareUserId 签发 shareUserSig
shareUserSig: '',
})
// 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
await handleCreateRoom();
})

// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
// roomId 为用户进入的房间号, 要求 roomId 类型为 number
// roomMode 包含'FreeSpeech'(自由发言模式) 和'ApplySpeech'(举手发言模式) 两种模式，默认为'FreeSpeech'，注意目前仅支持自由发言模式
// roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
const roomId = 123456;
const roomMode = 'FreeSpeech';
const roomParam = {
```

```
isOpenCamera: true,
isOpenMicrophone: true,
}
await TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
// roomId 为用户进入的房间号, 要求 roomId 类型为 number
// roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
const roomId = 123456;
const roomParam = {
isOpenCamera: true,
isOpenMicrophone: true,
}
await TUIRoomRef.value.enterRoom(roomId, roomParam);
}
</script>

<style>
html, body {
width: 100%;
height: 100%;
margin: 0;
}

#app {
width: 100%;
height: 100%;
}
</style>
```

注意：

在页面中复制以上代码之后，需要修改 TUIRoom 接口的参数为实际数据。

步骤四：配置开发环境

TUIRoom 组件引入之后，为了确保项目可以正常运行，需要进行以下配置：

1. **安装依赖**
   - 安装开发环境依赖：

```
npm install sass typescript unplugin-auto-import unplugin-vue-components -S -D
```

   - 安装生产环境依赖：

```
npm install element-plus events mitt pinia trtc-electron-sdk tim-js-sdk tsignaling -S
```

2. **注册 Pinia**

   TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 packages/renderer/src/main.ts 文件。

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

const app = createApp(App);
// 注册pina
createApp(App)
.use(createPinia())
.mount('#app')
.$nextTick(window.removeLoading)
```

3. **配置 svg-icon 组件加载 SVG 图标**

   TUIRoom 将所有图标资源（SVG 文件）放置在 packages/renderer/src/TUIRoom/assets/icons/svg 目录下，需要您在 packages/renderer/vite.config.ts 文件中配置 SVG 图标加载方式。

> ⚠ **注意：**
>
> 以下配置项为增量配置，不要删除已经存在的 Vite 配置项。

```
// vite.config.ts
import { createSvg } from './src/TUIRoom/assets/icons/index';
const path = require('path');
```

```
export default defineConfig({
// ...
plugins: [
// ...
createSvg(path.join(path.resolve(__dirname, 'src/TUIRoom/assets/icons/svg/'), '/')),
],
// ...
});
```

4. **配置 element-plus 按需引入**
   - **TUIRoom 使用 element-plus UI 组件，为避免引入所有 element-plus组件，需要您在** packages/renderer/vite.config.ts **中配置 element-plus 组件按需加载。**

> ⚠ **注意：**
> **以下配置项为增量配置，不要删除已经存在的 Vite 配置项。**

```
// vite.config.ts
import AutoImport from 'unplugin-auto-import/vite';
import Components from 'unplugin-vue-components/vite';
import { ElementPlusResolver } from 'unplugin-vue-components/resolvers';

export default defineConfig({
// ...
plugins: [
AutoImport({
resolvers: [ElementPlusResolver()],
}),
Components({
resolvers: [ElementPlusResolver({
importStyle: 'sass',
})],
}),
],
css: {
preprocessorOptions: {
scss: {
```

```
additionalData: `
@use '${path.resolve(__dirname, 'src/TUIRoom/assets/style/element.scss')}' as *;
`,
},
},
},
});
```

- 同时为了保证 element-plus 带 UI 组件能够正常显示样式，需要您在入口文件 packages/renderer/src/main.ts 中加载 element-plus 组件样式。

```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css'
import 'element-plus/theme-chalk/el-message-box.css'
```

5. **引入 trtc-electron-sdk**

为了在 UI 层以 import 方式引入 trtc-electron-sdk，统一代码风格，否则必须要以 require 的方式引入，需要您在 packages/renderer/vite.config.ts 中配置。

> ⚠ **注意：**
>
> 以下配置项将 resolve 中的内容替换掉：

```
// vite.config.ts

export default defineConfig({
// ...
plugins: [
resolve(
{
"trtc-electron-sdk": `
const TRTCCloud = require("trtc-electron-sdk");
const TRTCParams = TRTCCloud.TRTCParams;
const TRTCAppScene = TRTCCloud.TRTCAppScene;
const TRTCVideoStreamType = TRTCCloud.TRTCVideoStreamType;
const TRTCScreenCaptureSourceType = TRTCCloud.TRTCScreenCaptureSourceType;
const TRTCVideoEncParam = TRTCCloud.TRTCVideoEncParam;
const Rect = TRTCCloud.Rect;
const TRTCAudioQuality = TRTCCloud.TRTCAudioQuality;
```

```
const TRTCScreenCaptureSourceInfo = TRTCCloud.TRTCScreenCaptureSourceInfo;
const TRTCDeviceInfo = TRTCCloud.TRTCDeviceInfo;
const TRTCVideoQosPreference = TRTCCloud.TRTCVideoQosPreference;
const TRTCQualityInfo = TRTCCloud.TRTCQualityInfo;
const TRTCStatistics = TRTCCloud.TRTCStatistics;
const TRTCVolumeInfo = TRTCCloud.TRTCVolumeInfo;
const TRTCDeviceType = TRTCCloud.TRTCDeviceType;
const TRTCDeviceState = TRTCCloud.TRTCDeviceState;
const TRTCBeautyStyle = TRTCCloud.TRTCBeautyStyle;
const TRTCVideoResolution = TRTCCloud.TRTCVideoResolution;
const TRTCVideoResolutionMode = TRTCCloud.TRTCVideoResolutionMode;
const TRTCVideoMirrorType = TRTCCloud.TRTCVideoMirrorType;
const TRTCVideoRotation = TRTCCloud.TRTCVideoRotation;
const TRTCVideoFillMode = TRTCCloud.TRTCVideoFillMode;
export {
TRTCParams,
TRTCAppScene,
TRTCVideoStreamType,
TRTCScreenCaptureSourceType,
TRTCVideoEncParam,
Rect,
TRTCAudioQuality,
TRTCScreenCaptureSourceInfo,
TRTCDeviceInfo,
TRTCVideoQosPreference,
TRTCQualityInfo,
TRTCStatistics,
TRTCVolumeInfo,
TRTCDeviceType,
TRTCDeviceState,
TRTCBeautyStyle,
TRTCVideoResolution,
TRTCVideoResolutionMode,
TRTCVideoMirrorType,
TRTCVideoRotation,
TRTCVideoFillMode,
};
export default TRTCCloud.default;
`,
}
```

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

6. **env.d.ts文件配置**

env.d.ts **文件配置需要您在** packages/renderer/src/env.d.ts **中配置。**

> ⚠ **注意：**
> **以下配置项为增量配置，不要删除已经存在的** env.d.ts **文件配置。**

```
// env.d.ts

declare module 'tsignaling/tsignaling-js' {
import TSignaling from 'tsignaling/tsignaling-js';
export default TSignaling;
}

declare module 'tim-js-sdk' {
import TIM from 'tim-js-sdk';
export default TIM;
}
```

7. **如果项目中存在 import 动态加载，需要修改构建配置，打包生成 es 模块**

打包生成 es 模块需要您在 packages/renderer/vite.config.ts **中配置。**

> ⚠ **注意：**
> **项目中若不存在 import 动态加载，请不要进行此配置。以下配置项为增量配置，不要删除已经存在的 Vite 配置项。**

```
// vite.config.ts

export default defineConfig({
```

```
// ...
build: {
rollupOptions: {
output: {
format: 'es'
}
}
},
});
```

步骤五：开发环境运行

在控制台执行开发环境运行脚本，使用浏览器打开包含 TUIRoom 的页面，即可在页面中使用 TUIRoom 组件。如果您是使用 步骤二 中的脚本生成 Electron + Vue3 + TS 项目，您需要：

1. 执行开发环境命令。

```
npm run dev
```

> ⚠ 注意：
> 因 TUIRoom 按需引入 element-plus 组件，会导致开发环境路由页面第一次加载时反应较慢，等待 element-plus 按需加载完成即可正常使用。element-plus 按需加载不会影响打包之后的页面加载。

2. 体验 TUIRoom 组件功能。

步骤六：构建安装包、运行

在命令行终端中，执行以下命令构建安装包，构建好的安装包位于 release 目录下，可以安装运行。

```
npm run build
```

> ⚠ 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。

附录：TUIRoom API

TUIRoom 接口

init

初始化 TUIRoom 数据，任何使用 TUIRoom 的用户都需要调用 init 方法。

```
TUIRoomRef.value.init(roomData);
```

参数如下表所示：

参数	类型	含义
roomData	object	
roomData.sdkAppId	number	客户的 SDKAppId
roomData.userId	string	用户的唯一 ID
roomData.userSig	string	用户的 UserSig
roomData.userName	string	用户的昵称
roomData.userAvatar	string	用户的头像
roomData.shareUserId	string	非必填，用户进行屏幕分享的 UserId，要求 shareUserId = share_${userId}，无屏幕分享功能可不传入
roomData.shareUserSig	string	非必填，用户进行屏幕分享的 UserSig

createRoom

主持人创建房间。

```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	number	房间 ID
roomMode	string	房间模式，'FreeSpeech'（自由发言模式）和 'ApplySpeech'（举手发言模式），默认为 'FreeSpeech'，注意目前仅支持自由发言模式

参数	类型	含义
roomParam	Object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	String	非必填，默认扬声器设备 ID

enterRoom

普通成员加入房间。

```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	number	房间 ID
roomParam	Object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	String	非必填，默认扬声器设备 ID

TUIRoom 事件

onRoomCreate

创建房间回调。

```
<template>
<room ref="TUIRoomRef" @on-room-create="handleRoomCreate"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomCreate(info) {
if (info.code === 0) {
console.log('创建房间成功')
}
}
</script>
```

onRoomEnter

进入房间回调。

```
<template>
<room ref="TUIRoomRef" @on-room-enter="handleRoomEnter"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomEnter(info) {
if (info.code === 0) {
console.log('进入房间成功')
}
}
</script>
```

onRoomDestory

主持人销毁房间通知。

```
<template>
<room ref="TUIRoomRef" @on-room-destory="handleRoomDestory"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomDestory(info) {
if (info.code === 0) {
console.log('主持人销毁成功')
}
}
</script>
```

onRoomExit

普通成员退出房间通知。

```
<template>
<room ref="TUIRoomRef" @on-room-exit="handleRoomExit"></room>
</template>

<script setup lang="ts">
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';

function handleRoomExit(info) {
if (info.code === 0) {
console.log('普通成员退出房间成功')
}
}
</script>
```

TUIRoom API 查询
TUIRoom(Android)

最近更新时间：2022-06-02 15:22:10

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：
2. 主持人创建房间，进入房间人员输入房间号后进入房间。
3. 进入房间人员之间进行屏幕分享。
4. 支持发送各种文本消息和自定义消息。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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）。

API	描述
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。
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。

版权所有：腾讯云计算（北京）有限责任公司

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。
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。

版权所有：腾讯云计算（北京）有限责任公司

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager。

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。

版权所有：腾讯云计算（北京）有限责任公司

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<TUIRoomCoreDef.UserInfo> 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 callback);
```

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。

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();
```

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, TUIRoomCoreCallback.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。

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

版权所有：腾讯云计算（北京）有限责任公司

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。

参数	类型	含义
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

版权所有：腾讯云计算（北京）有限责任公司

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。

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 callback);
```

参数如下表所示：

参数	类型	含义

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.TRTCScr
eenShareParams 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。

参数	类型	含义
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

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
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
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。
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);
```

版权所有：腾讯云计算（北京）有限责任公司

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);
```

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
muted	boolean	是否禁用。

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<TRTCCloudDef.TRTCQuality > 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。

TUIRoom API（iOS）

最近更新时间：2022-06-02 15:22:14

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。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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）。

API	描述
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。
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

版权所有：腾讯云计算（北京）有限责任公司

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。
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

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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。

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。

版权所有：腾讯云计算（北京）有限责任公司

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。

```
+ (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<TUIRoomCoreDelegate>)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<TUIRoomUserInfo *> *)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
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。
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

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
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

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
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
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。

参数	类型	含义
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

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

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。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
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<TRTCQualityInfo *> *)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。

TRTCMeeting API（Flutter）

最近更新时间：2022-06-02 15:22:22

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。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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）。
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。

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

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。
onUserVideoAvailable	成员开启/关闭摄像头的回调。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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<TRTCMeeting> 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<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，获取方式请参见 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	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<ActionCallback> 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<ActionCallback> 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<ActionCallback> 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<ActionCallback> leaveMeeting();
```

getUserInfoList

获取房间内所有的人员列表，enterMeeting()成功后调用才有效。

```
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 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<UserListCallback> 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。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
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。

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

版权所有：腾讯云计算（北京）有限责任公司

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
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
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<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。

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。

版权所有：腾讯云计算（北京）有限责任公司

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。
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。

TUIRoom(Windows&Mac)

最近更新时间：2022-06-02 15:22:28

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。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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）。

API	描述
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）。
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。

版权所有：腾讯云计算（北京）有限责任公司

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。
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。

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

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<TUIUserInfo> 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&
view) = 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
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;
```

版权所有：腾讯云计算（北京）有限责任公司

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

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
user_id	string	用户 ID。
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;
```

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
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;
```

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
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。
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
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;
```

版权所有：腾讯云计算（北京）有限责任公司

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
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

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

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
volume	uint32_t	音量大小。
muted	bool	是否被禁用。

语音互动直播
集成 TUIVoiceRoom (Android)

最近更新时间：2022-06-23 17:12:46

组件介绍

TUIVoiceRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIVoiceRoom 组件，您只需要编写几行代码就可以为您的 App 添加“多人语音聊天”等场景，TUIVoiceRoom 同时支持 iOS、小程序 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android/debug 和 Android/tuivoiceroom 目录到您的工程中，并完成如下导入动作：

- 在 setting.gradle 中完成导入，参考如下：

```
include ':debug'
include ':tuivoiceroom'
```

- 在 app 的 build.gradle 文件中添加对 tuivoiceroom 的依赖：

```
api project(':tuivoiceroom')
```

- 在根目录的build.gradle文件中添加TRTC SDK和IM SDK的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请麦克风权限等）：

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

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

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

步骤三：初始化并登录

```
// 1.初始化，
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
```

```
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
);
// 2.登录，
mTRTCVoiceRoom.login(SDKAppID, userId, userSig, new TRTCVoiceRoomCallback.ActionCallbac
k() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的userSig，也可以参照我们的 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现语音聊天房间

1. 实现房主创建语音聊天房间 TRTCVoiceRoom#createRoom

```
// 1.房主调用创建房间
int roomId = 12345; //房间id
final TRTCVoiceRoomDef.RoomParam roomParam = new TRTCVoiceRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 7; // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.coverUrl = "房间封面图的 URL 地址";
mTRTCVoiceRoom.createRoom(roomId, roomParam, new TRTCVoiceRoomCallback.ActionCallb
ack() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//创建成功
}
}
});
```

2. 实现听众加入语音聊天房间 TRTCVoiceRoom#enterRoom

```
// 1.听众调用加入房间
mTRTCVoiceRoom.enterRoom(roomId, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
}
}
});
```

3. 实现听众主动上麦 TRTCVoiceRoom#enterSeat

```
// 1: 听众调用上麦
int seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
```

```
public void onCallback(int code, String msg) {
if (code == 0) {
//操作成功
}
}
});

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

4. **实现房主抱人上麦 TRTCVoiceRoom#pickSeat**

```
// 1: 房主调用抱人麦位
int seatIndex = 2; //麦位的index
String userId = "123"; //需要上麦的用户id
mTRTCVoiceRoom.pickSeat(1, userId, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//操作成功
}
}
});

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

5. **实现听众申请上麦 TRTCVoiceRoom#sendInvitation**

```
// 听众端视角
// 1.听众调用申请上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("takeSeat", userId, seatIndex, null);
```

```
// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
if(id.equals(inviteId)) {
mTRTCVoiceRoom.enterSeat(index, null);
}
}

// 房主端视角
// 1.房主收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String cont
ent) {
if (cmd.equals("takeSeat")) {
// 2.房主同意听众请求
mTRTCVoiceRoom.acceptInvitation(id, null);
}
}
```

6. 实现房主邀请上麦 TRTCVoiceRoom#sendInvitation

```
// 房主端视角
// 1.房主调用请求抱人上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("pickSeat", userId, seatIndex, null);

// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
if(id.equals(inviteId)) {
mTRTCVoiceRoom.pickSeat(index, null);
}
}

// 听众端视角
// 1.听众收到请求
```

```
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String cont
ent) {
if (cmd.equals("pickSeat")) {
// 2.听众同意房主请求
mTRTCVoiceRoom.acceptInvitation(id, null);
}
}
```

7. 实现文字聊天 TRTCVoiceRoom#sendRoomTextMsg

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

8. 实现弹幕消息 TRTCVoiceRoom#sendRoomCustomMsg

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInf
o 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！ ");
```

```
}
}
});
```

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIVoiceRoom (iOS)

最近更新时间：2022-06-23 17:12:49

组件介绍

TUIVoiceRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIVoiceRoom 组件，您只需要编写几行代码就可以为您的 App 添加“多人语音聊天”等场景。TUIVoiceRoom 同时支持 Android、小程序 等平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

在您的 xcode 工程 Podfile 文件同一级目录下创建 TUIVoiceRoom 文件夹，将 Github仓库 iOS 目录 下的 TXAppBasic、Resources、Source、TUIVoiceRoom.podspec 等文件拷贝至您在自己工程创建的 TUIVoiceRoom 目录下。并完成如下导入动作：

- 打开工程的 Podfile 文件，引入 TUIVocieRoom.podspec，参考如下：

```
# path 为TXAppBasic.podspec相对于Podfile文件的相对路径
pod 'TXAppBasic', :path => "TUIVoiceRoom/TXAppBasic/"
# path 为TUIVoiceRoom.podspec相对于Podfile文件的相对路径
pod 'TUIVoiceRoom', :path => "TUIVoiceRoom/", :subspecs => ["TRTC"]
```

- 终端进入 Podfile 所在的目录下，执行 pod install，参考如下：

```
pod install
```

步骤二：配置权限及混淆规则

在 info.plist 文件中需要添加 Privacy > Microphone Usage Description 申请麦克风权限。

```
<key>NSMicrophoneUsageDescription</key>
<string>VoiceRoomApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```

步骤三：初始化并登录

```
// 初始化
let mTRTCVoiceRoom = TRTCVoiceRoom.shared()
// 登录
mTRTCVoiceRoom.login(sdkAppID: SDKAppID, userId: userId, userSig: userSig) { code, message
in
if code == 0 {
//登录成功
}
}
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的userSig，也可以参照我们的 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现语音聊天房间

1. **实现房主创建语音聊天房间 TRTCVoiceRoom#createRoom**

```
// 初始化语聊房参数
let roomParam = VoiceRoomParam()
roomParam.roomName = "房间名称"
roomParam.needRequest = false // 听众上麦是否需要房主同意
roomParam.coverUrl = "房间封面图的 URL 地址"
roomParam.seatCount = 7 // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.seatInfoList = []
// 初始化麦位信息
for _ in 0..< param.seatCount {
let seatInfo = VoiceRoomSeatInfo()
param.seatInfoList.append(seatInfo)
}
// 房主端创建房间
mTRTCVoiceRoom.createRoom(roomID: yourRoomID, roomParam: roomParam) { (code, mess
```

```
age) in
if code == 0 {
// 创建成功
}
}
```

2. 实现听众加入语音聊天房间 TRTCVoiceRoom#enterRoom

```
// 1.听众调用加入房间
mTRTCVoiceRoom.enterRoom(roomID: roomID) { (code, message) in
// 进入房间结果回调
if code == 0 {
// 进房成功
}
}
```

3. 实现听众主动上麦 TRTCVoiceRoom#enterSeat

```
// 1: 听众调用上麦
let seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex: 2) { (code, message) in
if code == 0 {
// 上麦成功
}
}

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
// 刷新的麦位列表
}
```

4. 实现房主抱人上麦 TRTCVoiceRoom#pickSeat

```
// 1: 房主调用抱人麦位
let seatIndex = 2; //麦位的index
let userId = "123"; //需要上麦的用户id
```

```
mTRTCVoiceRoom.pickSeat(seatIndex: 1, userId: "123") { (code, message) in
if code == 0 {
}
}

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
// 刷新的麦位列表
}
```

5. 实现听众申请上麦 TRTCVoiceRoom#sendInvitation

```
// 听众端视角
// 1.听众调用申请上麦
let seatIndex = "1"; //麦位的index
let userId = "123"; //用户id
let inviteId = mTRTCVoiceRoom.sendInvitation(cmd: "takeSeat", userId: ownerUserId, conten
t: "1") { (code, message) in
// 发送结果回调
}

// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
self.mTRTCVoiceRoom.enterSeat(seatIndex: ) { (code, message) in
// 上麦结果回调
}
}
}

// 房主端视角
// 1.房主收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
if cmd == "takeSeat" {
// 2.房主同意听众请求
self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

6. 实现房主邀请上麦 TRTCVoiceRoom#sendInvitation

```
// 房主端视角
// 1.房主调用 sendInvitation，请求抱听众“123”上2号麦
let inviteId = self.mTRTCVoiceRoom.sendInvitation(cmd: "pickSeat", userId: ownerUserId, con
tent: "2") { (code, message) in
// 发送结果回调
}

// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
self.mTRTCVoiceRoom.pickSeat(seatIndex: ) { (code, message) in
// 上麦结果回调
}
}
}

// 听众端视角
// 1.听众收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
if cmd == "pickSeat" {
// 2.听众同意房主请求
self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

7. 实现文字聊天 TRTCVoiceRoom#sendRoomTextMsg

```
// 发送端：发送文本消息
self.mTRTCVoiceRoom.sendRoomTextMsg(message: message) { (code, message) in

}

// 接收端：监听文本消息
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
```

```
//收到的message信息处理方法
}
```

8. **实现弹幕消息 TRTCVoiceRoom#sendRoomCustomMsg**

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: “CMD_DANMU”, message: "hello world", c
allback: nil)
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)

// 接收端：监听自定义消息
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserInfo) {
if cmd == "CMD_DANMU" {
// 收到弹幕消息
}
if cmd == "CMD_LIKE" {
// 收到点赞消息
}
}
```

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIVoiceRoom (小程序)

最近更新时间：2022-06-02 15:22:55

组件介绍

TUIVoiceRoom 组件是一个开源的场景化解决方案，帮助您快速在您的小程序中集成**语音聊天室**功能，内部集成语音聊天与文字聊天功能，非常适合用于多人语音会议，娱乐聊天室，语音交友平台等场景中。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

创建房间

由于用户不需要安装 App 就可以使用该功能，因此 TUIVoiceRoom 非常适合用于快速发起多人语音聊天室的场景。

房间管理

房间内部继承**低延迟、高质量**语音通话功能，创建者可对房间麦位进行锁定解锁并对用户进行禁言、禁画以及踢出操作。

其它平台

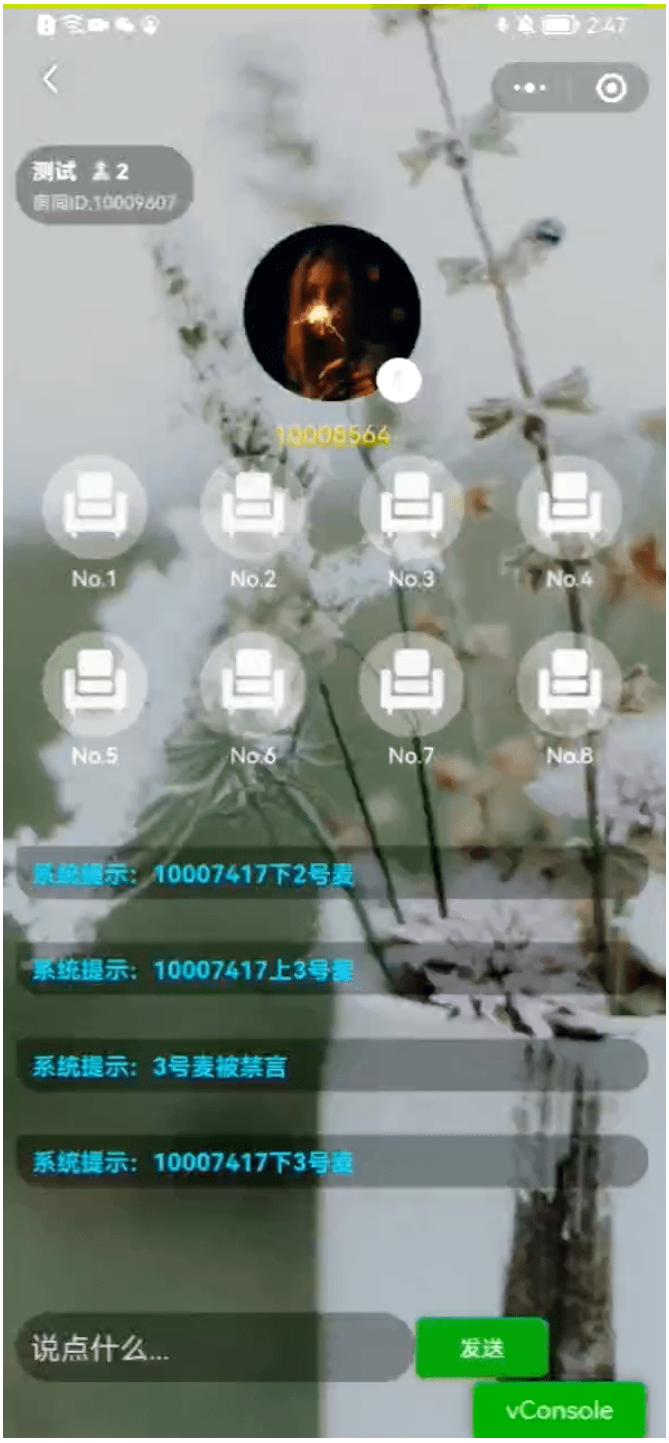
除了小程序版的 TUIVoiceRoom，我们同时也推出了 Android、iOS 等平台的源代码，用户可直接使用并对源码进行修改。

效果展示

您可以单击 Demo 体验 扫描二维码单击**语音聊天室**进行体验。语音聊天室包括麦位管理、低延时语音互动、文字聊天等 TRTC 在语音聊天场景下的相关能力。

房主麦位操作	听众麦位操作





如需快速接入语音聊天室功能，您可以直接基于我们提供的 Demo 进行修改适配。

环境要求

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

- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在真机上进行运行体验。
- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，需要申请常规小程序账号进行开发。
- 不支持 uniapp 开发环境，请使用原生小程序开发环境。

前提条件

1. 您已 注册腾讯云 账号，并完成 实名认证。
2. **开通小程序类目与推拉流标签权限（如不开通则无法正常使用）。**
   出于政策和合规的考虑，微信暂未放开所有小程序对实时音视频功能（即 <live-pusher> 和 <live-player> 标签）的支持：
   - 小程序推拉流标签不支持个人小程序，只支持企业类小程序。
   - 小程序推拉流标签使用权限暂时只开放给有限 类目。
   - 符合类目要求的小程序，需要在 **微信公众平台 > 开发 > 开发管理 > 接口设置** 中自助开通该组件权限，如下图所示：



跑通 Demo

步骤一：开通小程序权限

由于 TUIVoiceRoom 所使用的小程序标签有更苛刻的权限要求，因此集成 TUIVoiceRoom 的第一步就是要开通小程序的类目和标签使用权限，**否则无法使用**，这包括如下步骤：

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



- 小程序推拉流标签使用权限暂时只开放给有限 类目。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：



步骤二：在小程序控制台配置域名

在 **微信公众平台** > **开发** > **开发管理** > **开发设置** > **服务器域名**中设置 **request合法域名**和 **socket合法域名**，如下图所示：

- request 合法域名：

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

- socket合法域名：

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



步骤三：创建新的应用

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

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

步骤四：下载 App 源码

单击进入 TUIVoiceRoom，Clone 或者下载源码。

步骤五：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。

2. 找到并打开 ./debug/GenerateTestUserSig.js 文件。
3. 设置 GenerateTestUserSig.js 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。

```
function genTestUserSig(userID) {
  /**
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  var SDKAPPID    = 0;    将 SDKAppID 粘贴到此处

  /**
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  var EXPIRETIME = 604800;

  /**
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一个，
   * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
   * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  var SECRETKEY = "";    将 密钥 粘贴到此处
```

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

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

步骤六：编译运行

1. 打开微信开发者工具，选择**小程序**，单击新建图标，选择**导入项目**。

2. 填写您微信小程序的 AppID，单击**导入**。



> **注意：**
> 此处应输入您微信小程序的 AppID，而非 SDKAppID。

3. 单击**预览**，生成二维码，通过手机微信扫码二维码即可进入小程序。

技术咨询

了解更多详情您可 QQ 咨询：941036374 *技术交流群*

参考文档

- 实时音视频 trtc-wx API 手册
- 即时通信 IM sdk API 手册
- 小程序 TUIVoiceRoom 源码
- 小程序 TUIRoom 源码
- 小程序如何支持 typescript + sass/less

TUIVoiceRoom API 查询
TRTCVoiceRoom (iOS)

最近更新时间：2022-06-02 15:25:40

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。

API	描述
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 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。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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。
acceptInvitation	接受邀请。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
/**
* 获取 TRTCVoiceRoom 单例对象
*
* - returns: TRTCVoiceRoom 实例
* - note: 可以调用 {@link TRTCVoiceRoom#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCVoiceRoom 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCVoiceRoom 实例不能再使用，需要重新调用 {@link TRTCVoiceRoom#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

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

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCVoiceRoom 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，可使用 {@link TRTCVoiceRoom#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCVoiceRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

> 说明：
> setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送动主线程 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:(ActionCallback _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	NSString	昵称。
avatarURL	NSString	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(VoiceRoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	VoiceRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. **房主调用** createRoom **创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。**
2. **房主创建房间成功后，调用** enterSeat **进入座位。**
3. **房主收到组件的** onSeatListChange **麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。**
4. **房主还会收到麦位表有成员进入的** onAnchorEnterSeat **的事件通知，此时会自动打开麦克风采集。**

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	NSInteger	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用 enterRoom 并传入房间号即可进入该房间。
3. 进房后会收到组件的 onRoomInfoChange 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 onSeatListChange 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 onAnchorEnterSeat 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

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。

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。

moveSeat

移动麦位 (麦上主播端可调用)。

> 说明：
> 移动麦位成功后，房间内所有成员会收到 onSeatListChange 、 onAnchorLeaveSeat 和 onAnchorEnterSeat 的事件通知。(主播调用后，只是修改麦位座位号信息，并不会切换该用户的主播身份。)

```
- (NSInteger)moveSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(moveSeat(seatIndex:callback:))
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要移动到的麦位序号。
callback	ActionCallback	操作回调。

返回值：

返回值	类型	含义
code	NSInteger	移动麦位操作结果（0为成功，其它为失败，10001为接口调用限频）。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(pickSeat(seatIndex:userId:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要抱上麦的麦位序号。
userId	NSString	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 sendInvitation 向听众申请，收到 onInvitationAccept 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> **说明：**
> 房主踢人下麦，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(kickSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> **说明：**
> 静音/解除静音某个麦位，房间内所有成员会收到 onSeatListChange 和 onSeatMute 的事件通知。

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(muteSeat(seatIndex:isMute:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。
isMute	BOOL	YES：静音对应麦位；NO：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 onSeatListChange 和 onSeatClose 的事件通知。

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(closeSeat(seatIndex:isClose:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。
isClose	BOOL	YES：封禁对应麦位； NO：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	静音/取消静音，详情请参见 TRTC SDK。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
useSpeaker	BOOL	YES：扬声器；NO：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)voluem NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (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。

参数	类型	含义
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。

参数	类型	含义
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。

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

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:(NSInte
ger)totalVolume
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

参数	类型	含义
index	NSInteger	操作的麦位。
isClose	BOOL	YES：封禁麦位； NO： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)command
message:(NSString *)message
userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(command:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
command	NSString	命令字，由开发者自定义，主要用于区分不同消息类型。
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
inviter:(NSString *)inviter
cmd:(NSString *)cmd
```

```
content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(id:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。
cmd	NSString	业务指定的命令字，由开发者自定义。
content	NSString	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。

TRTCVoiceRoom (Android)

最近更新时间：2022-06-02 15:25:08

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

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。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
public static synchronized TRTCVoiceRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

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

setDelegate

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

```
public abstract void setDelegate(TRTCVoiceRoomDelegate delegate);
```

> 说明：
> setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
String userId, String userSig,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

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

logout

登出。

```
public abstract void logout(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCVoiceRoomCallback.
ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam, TRTC
VoiceRoomCallback.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

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用enterRoom并传入房间号即可进入该房间。
3. 进房后会收到组件的onRoomInfoChange房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的onSeatListChange麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的onAnchorEnterSeat的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

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。
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

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。

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。

```
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

版权所有：腾讯云计算（北京）有限责任公司

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。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
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 callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callbac
k);
```

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。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
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
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用 muteLocalAudio，房间内的其他用户都会收到此通知。

```
void onUserMicrophoneMute(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	true：静音麦位； false：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVolume)
;
```

参数如下表所示：

参数	类型	含义
userVolumes	ListList<trtcclouddef.trtcvolumeinfo>	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

版权所有：腾讯云计算（北京）有限责任公司

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(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 userInf
o);
```

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。

版权所有：腾讯云计算（北京）有限责任公司

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。
</trtcclouddef.trtcvolumeinfo>		

视频互动直播
集成 TUILiveRoom (Android)

最近更新时间：2022-06-23 17:12:28

组件介绍

TUILiveRoom 是一个开源的视频直播 UI 组件，通过在项目中集成 TUILiveRoom 组件，您只需要编写几行代码就可以为您的 App 添加“视频互动直播”场景。TUILiveRoom 包含 Android、iOS、小程序等平台的源代码，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUILiveRoom 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android/debug ， Android/tuiaudioeffect ， Android/tuibarrage ， Android/tuibeauty ， Android/tuigift 和 Android/tuiliveroom 目录到您的工程中，并完成如下导入动作：

- 在 setting.gradle 中完成导入，参考如下：

```
include ':debug'
include ':tuibeauty'
include ':tuibarrage'
include ':tuiaudioeffect'
include ':tuigift'
include ':tuiliveroom'
```

- 在 app 的 build.gradle 文件中添加对 tuiliveroom 的依赖：

```
api project(":tuiliveroom")
```

- 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

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

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

版权所有：腾讯云计算（北京）有限责任公司

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

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

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

步骤三：初始化并登录组件

```
// 1.添加事件监听及登录
TUILogin.addLoginListener(new TUILoginListener() {
@Override
public void onConnecting() { // 正在连接中
super.onConnecting();
}
@Override
public void onConnectSuccess() { // 连接成功通知
super.onConnectSuccess();
}
@Override
public void onConnectFailed(int errorCode, String errorMsg) { // 连接失败通知
super.onConnectFailed(errorCode, errorMsg);
}
@Override
public void onKickedOffline() { // 登录被踢下线通知（示例：账号在其他设备登录）
super.onKickedOffline();
}
@Override
public void onUserSigExpired() { // userSig过期通知
super.onUserSigExpired();
}
});
TUILogin.login(mContext, "Your SDKAppID", "Your userId", "Your userSig", new TUICallback() {
@Override
public void onSuccess() {
}
```

```
@Override
public void onError(int errorCode, String errorMsg) {
Log.d(TAG, "errorCode: " + errorCode + " errorMsg:" + errorMsg);
}
});

// 2.初始化TUILiveRoom组件
TUILiveRoom mLiveRoom = TUILiveRoom.sharedInstance(mContext);
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的userSig，也可以参照我们的 示例工程 自行计算，更多信息请参见 如何计算及使用 UserSig。

步骤四：实现视频互动直播间

1. 主播端开播

```
mLiveRoom.createRoom(int roomId, String roomName, String coverUrl);
```

2. 观众端观看

```
mLiveRoom.enterRoom(roomId);
```

3. 观众与主播连麦 TRTCLiveRoom#requestJoinAnchor

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

// 2.主播端收到连麦请求
mTRTCLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
@Override
public void onRequestJoinAnchor(final TRTCLiveRoomDef.TRTCLiveUserInfo userInfo,
String reason, final int timeout) {
// 同意对方的连麦请求
mTRTCLiveRoom.responseJoinAnchor(userInfo.userId, true, "同意连麦");
}

@Override
public void onAnchorEnter(final String userId) {
```

```
// 主播收到连麦观众的上麦通知
TXCloudVideoView view = new TXCloudVideoView(context);
parentView.add(view);
// 主播播放观众画面
mTRTCLiveRoom.startPlay(userId, view, null);
}
});
```

4. 主播与主播 PK TRTCLiveRoom#requestRoomPK

```
// 主播 A 创建12345的房间
mLiveRoom.createRoom(12345, "roomA", "Your coverUrl");
// 主播 B 创建54321的房间
mLiveRoom.createRoom(54321, "roomB", "Your coverUrl");

// 主播 A:
TRTCLiveRoom mTRTCLiveRoom=TRTCLiveRoom.sharedInstance(mContext);

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

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

```
});

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

步骤五：美颜特效（可选）

TUILiveRoom 美颜使用了 腾讯特效 SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License 申请指引。

```
TUIBeautyView.getBeautyService().setLicense(context, “XMagicLicenseURL”, “XMagicLicenseKey”);
```

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUILiveRoom (iOS)

最近更新时间：2022-06-23 17:12:32

组件介绍

TUILiveRoom 是一个开源的视频直播 UI 组件，通过在项目中集成 TUILiveRoom 组件，您只需要编写几行代码就可以为您的 App 添加“视频互动直播”场景。TUILiveRoom包含 Android、iOS、小程序等平台的源代码，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：导入 TUILiveRoom 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 在您的工程 Podfile 文件同一级目录下创建 TUILiveRoom 文件夹。
2. 单击进入 Github/TUILiveRoom ，选择克隆/下载代码，然后将 TUILiveRoom/iOS/ 目录下的 Source、Resources 、TUIBeauty、TUIAudioEffect、TUIBarrage、TUIGift、TXAppBasic文件夹 和 TUILiveRoom.podspec 文件拷贝到您在 步骤1 创建的 TUILiveRoom 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖，之后执行 pod install 命令，完成导入。

```
# :path => "指向TUILiveRoom.podspec的相对路径"
pod 'TUILiveRoom', :path => "./TUILiveRoom/TUILiveRoom.podspec", :subspecs => ["TRTC"]
# :path => "指向TXAppBasic.podspec的相对路径"
pod 'TXAppBasic', :path => "./TUILiveRoom/TXAppBasic/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIBeauty', :path => "./TUILiveRoom/TUIBeauty/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIAudioEffect', :path => "./TUILiveRoom/TUIAudioEffect/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIBarrage', :path => "./TUILiveRoom/TUIBarrage/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIGift', :path => "./TUILiveRoom/TUIGift/"
```

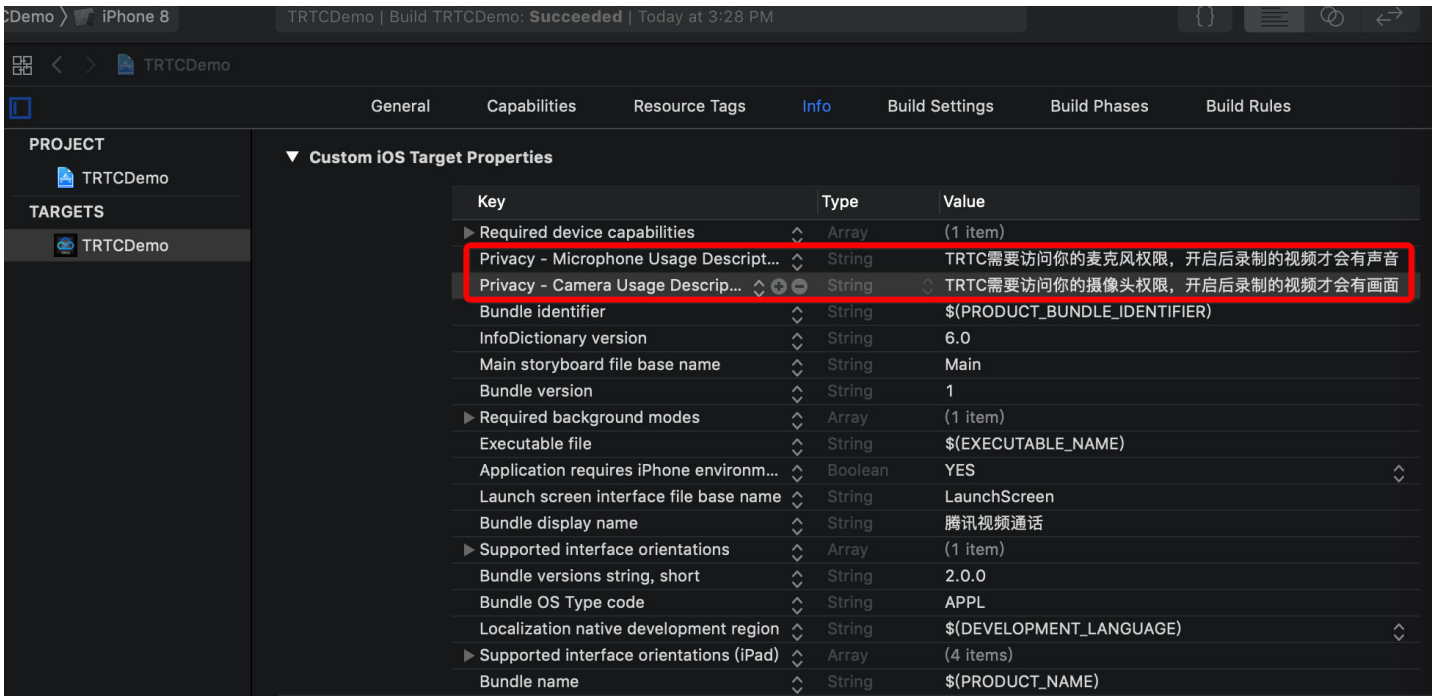
⚠ 注意：

- Source、Resources 文件夹 和 TUILiveRoom.podspec 文件必需在同一目录下。
- TXAppBasic.podspec 在 TXAppBasic 文件夹下。

步骤二：配置权限

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

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



步骤三：初始化并登录组件

Objective-C

```
@import TUILiveRoom;
@import TUICore;

// 1.组件登录
[TUILogin login:@"您的SDKAppID" userID:@"您的UserID" userSig:@"您的UserSig" succ:^{

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

}];
// 2.初始化TUILiveRoom实例
TUILiveRoom *mLiveRoom = [TUILiveRoom sharedInstance];
```

Swift

```
import TUILiveRoom
import TUICore

// 1.组件登录
TUILogin.login("您的SDKAppID", userID: "您的UserID", userSig: "您的UserSig") {

} fail: { code, msg in
```

```
}
// 2.初始化TUILiveRoom实例
let mLiveRoom = TUILiveRoom.sharedInstance
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppID 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **UserID**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **UserSig**：根据 SDKAppId、UserID，SecretKey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUILiveRoom 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现视频互动直播间

1. 主播端开播

Objective-C

```
[mLiveRoom createRoomWithRoomId:123 roomName:@"test room" coverUrl:@""];
```

Swift

```
mLiveRoom.createRoom(roomId: 123, roomName: "test room", coverUrl:"")
```

2. **观众端观看**

Objective-C

```
[mLiveRoom enterRoomWithRoomId:123];
```

Swift

```
mLiveRoom.createRoom(roomId: 123)
```

3. **观众与主播连麦 TRTCLiveRoom#requestJoinAnchor**

Objective-C

```
// 1.观众端发起连麦请求
[TRTCLiveRoom shareInstance].delegate = self;
// @param mSelfUserId String 当前用户id
NSString *mSelfUserId = @"1314";
[[TRTCLiveRoom shareInstance] requestJoinAnchor:[NSString stringWithFormat:@"%@ 请求和你连麦", mSelfUserId] timeout:30 responseCallback:^(BOOL agreed, NSString * _Nullable reason) {
if (agreed) {
// 主播接受了观众的请求
UIView *playView = [UIView new];
[self.view addSubView:playView];
// 观众启动预览，开启推流
[[TRTCLiveRoom shareInstance] startCameraPreviewWithFrontCamera:YES view:playView callback:nil];
[[TRTCLiveRoom shareInstance] startPublishWithStreamID:[NSString stringWithFormat:@"%@_stream", mSelfUserId] callback:nil];
}
}];

// 2.主播端收到连麦请求
#pragma mark - TRTCLiveRoomDelegate
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onRequestJoinAnchor:(TRTCLiveUserI
```

```
nfo *)user reason:(NSString *)reason {
// 同意对方的连麦请求
[[TRTCLiveRoom shareInstance] responseJoinAnchor:user.userId agree:YES reason:@"同意连麦"];
}

- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onAnchorEnter:(NSString *)userID {
// 主播收到连麦观众的上麦通知
UIView *playView = [UIView new];
[self.view addSubview:playView];
// 主播播放观众画面
[[TRTCLiveRoom shareInstance] startPlayWithUserID:userID view:playView callback:nil];
}
```

Swift

```
// 1.观众端发起连麦请求
TRTCLiveRoom.shareInstance().delegate = self
let mSelfUserId = "1314"
TRTCLiveRoom.shareInstance().requestJoinAnchor(reason: mSelfUserId + "请求和您连麦", timeout: 30) { [weak self] (agree, msg) in
guard let self = self else { return }
if agree {
// 主播接受了观众的请求
let playView = UIView()
self.view.addSubView(playView)
// 观众启动预览，开启推流
TRTCLiveRoom.shareInstance().startCameraPreview(frontCamera: true, view: playView)
TRTCLiveRoom.shareInstance().startPublish(streamID: mSelfUserId + "_stream")
}
}

// 2.主播端收到连麦请求
extension ViewController: TRTCLiveRoomDelegate {

func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestJoinAnchor user: TRTCLiveUserInfo, reason: String?) {
// 同意对方的连麦请求
TRTCLiveRoom.shareInstance().responseRoomPK(userID: user.userId, agree: true, reason:
```

```
"同意连麦")
}

func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
// 主播收到连麦观众的上麦通知
let playView = UIView()
view.addSubview(playView)
// 主播播放观众画面
TRTCLiveRoom.shareInstance().startPlay(userID: userID, view: playView);
}
}
```

4. **主播与主播 PK TRTCLiveRoom#requestRoomPK**

Objective-C

```
// 主播 A 创建12345的房间
[[TUILiveRoom sharedInstance] createRoomWithRoomId:12345 roomName:@"roomA" cov
erUrl:@"roomA coverUrl"];
// 主播 B 创建54321的房间
[[TUILiveRoom sharedInstance] createRoomWithRoomId:54321 roomName:@"roomB" cov
erUrl:@"roomB coverUrl"];

// 主播 A
// 主播 A 向 主播 B 发起PK请求
[[TRTCLiveRoom shareInstance] requestRoomPKWithRoomID:543321 userID:@"roomB use
rId" timeout:30 responseCallback:^(BOOL agreed, NSString * _Nullable reason) {
if (agreed) {
// 用户B接受
} else {
// 用户B拒绝
}
}];

// 主播 B:
// 2.主播 B 收到主播 A 的消息
#pragma mark - TRTCLiveRoomDelegate
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onRequestRoomPK:(TRTCLiveUserInfo
*)user {
// 3.主播 B 回复主播 A 接受请求
```

```
[[TRTCLiveRoom shareInstance] responseRoomPKWithUserID:user.userId agree:YES reaso
n:@""];
}

- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onAnchorEnter:(NSString *)userID {
// 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
[[TRTCLiveRoom shareInstance] startPlayWithUserID:userID view:playAView callback:nil];
}
```

Swift

```
// 主播 A 创建12345的房间
TUILiveRoom.sharedInstance.createRoom(roomId: 12345, roomName: "roomA")
// 主播 B 创建54321的房间
TUILiveRoom.sharedInstance.createRoom(roomId: 54321, roomName: "roomB")

// 主播 A
// 主播 A 向 主播 B 发起PK请求
TRTCLiveRoom.shareInstance().requestRoomPK(roomID: 543321, userID: "roomB userId", t
imeout: 30) { [weak self] (agreed, msg) in
guard let self = self else { return }
if agreed {
// 用户B接受
} else {
// 用户B拒绝
}
}

// 主播 B:
// 2.主播 B 收到主播 A 的消息
extension ViewController: TRTCLiveRoomDelegate {
func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestRoomPK user: TRTCLiveUserI
nfo) {
// 3.主播 B 回复主播 A 接受请求
TRTCLiveRoom.shareInstance().responseRoomPK(userID: user.userId, agree: true, reason:
"")
}

func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
```

```
// 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
TRTCLiveRoom.shareInstance().startPlay(userID: userID, view: playAView);
}
}
```

步骤五：美颜特效（可选）

TUILiveRoom 美颜使用了 腾讯特效 SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License 申请指引。

Objective-C

```
@import TUIBeauty;

- (void)setXMagicLicense {
[[TUIBeautyView getBeautyService] setLicenseUrl:@"XMagicLicenseURL" key:@"XMagicLicen
seKey" completion:^(NSInteger authResult, NSString * _Nonnull errorMsg) {

}];
}
```

Swift

```
import TUIBeauty

func setXMagicLicence() {
// [Option] Tencent Effect: XMagic Beauty License
TUIBeautyView.getBeautyService().setLicenseUrl(XMagicLicenseURL, key: XMagicLicenseKey
) { code, msg in
debugPrint("auth result code:\(code) msg:\(msg)")
}
}
```

常见问题

更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUILiveRoom (Flutter)

最近更新时间：2022-06-02 15:26:16

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。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

复用 Demo 的 UI 界面

步骤1：创建新的应用

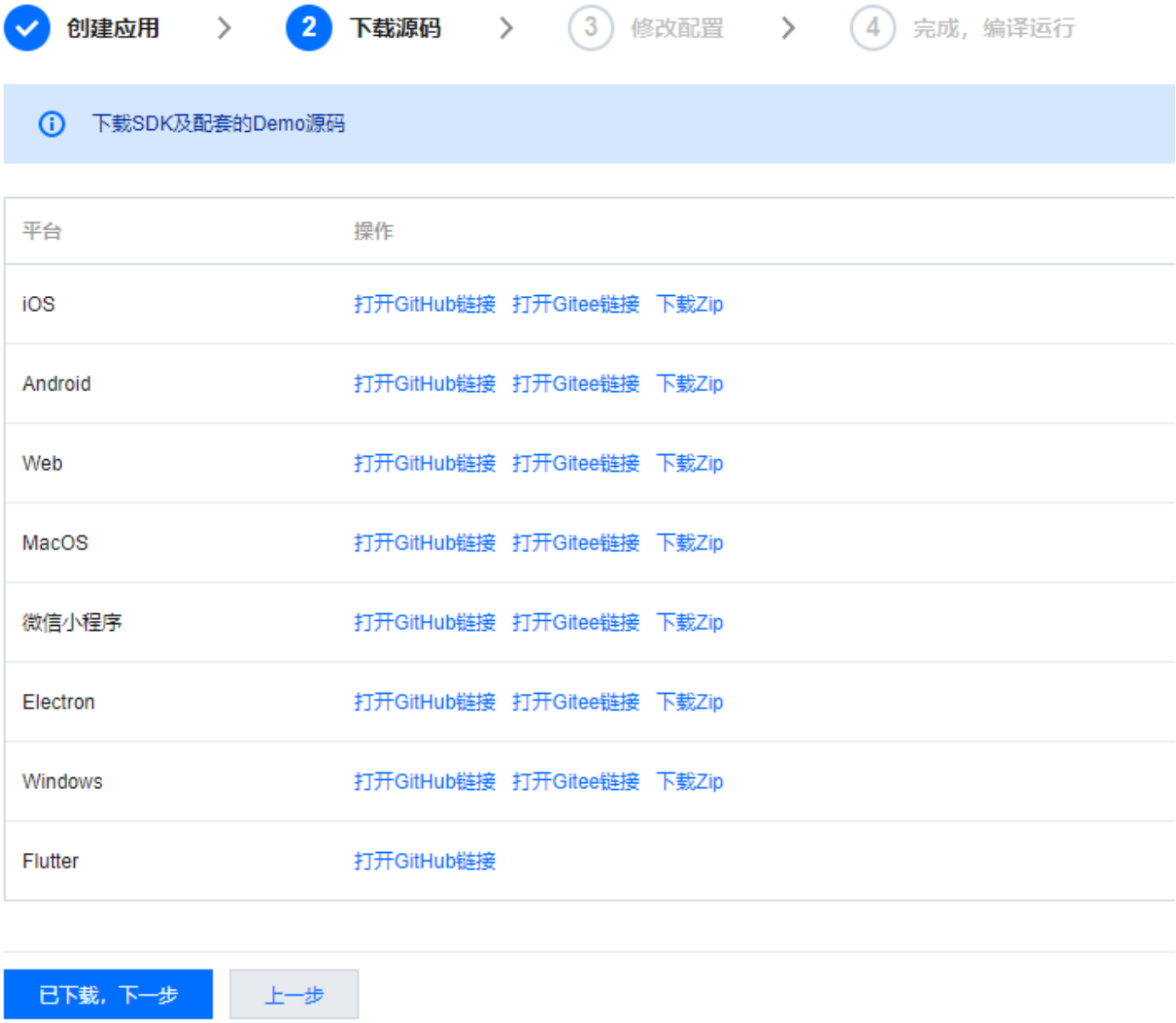
1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 输入应用名称，例如 TestLive ，单击【创建】。

> 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。

```
class GenerateTestUserSig {
  /*
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  static int sdkAppId = 0;    将SDKAppID粘贴到此处

  /*
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  static int expireTime = 604800;

  /*
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/trtc )，如果还没有应用就创建一个，
   * step2. 单击"应用配置"进入基础配置页面，并进一步找到"帐号体系集成"部分。
   * step3. 点击"查看密钥"按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  static String secretKey = '';    将 密钥 粘贴到此处
```

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

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

步骤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**
   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。
   iOS 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 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

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>
```

Android 端

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

3. 将 tools:replace="android:label" 加入到 application 中。

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

android > app > src > main > AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mlp">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        tools:replace="android:label"
        android:name="io.flutter.app.FlutterApplication"
        android:label="mlp"
        android:icon="@mipmap/ic_launcher">
```

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。

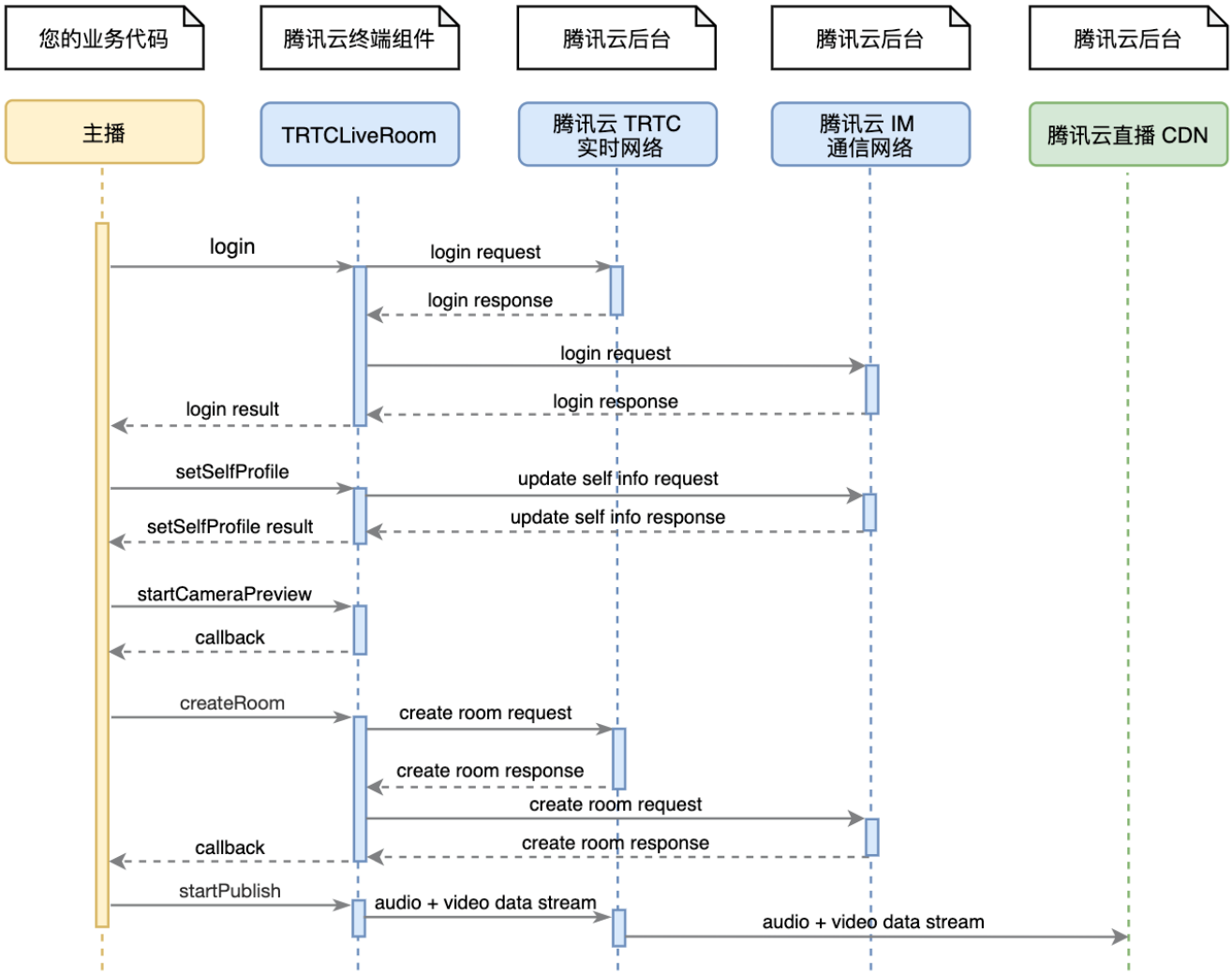
CDNPlayDomain	在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >【域名管理】页面中进行设置。

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

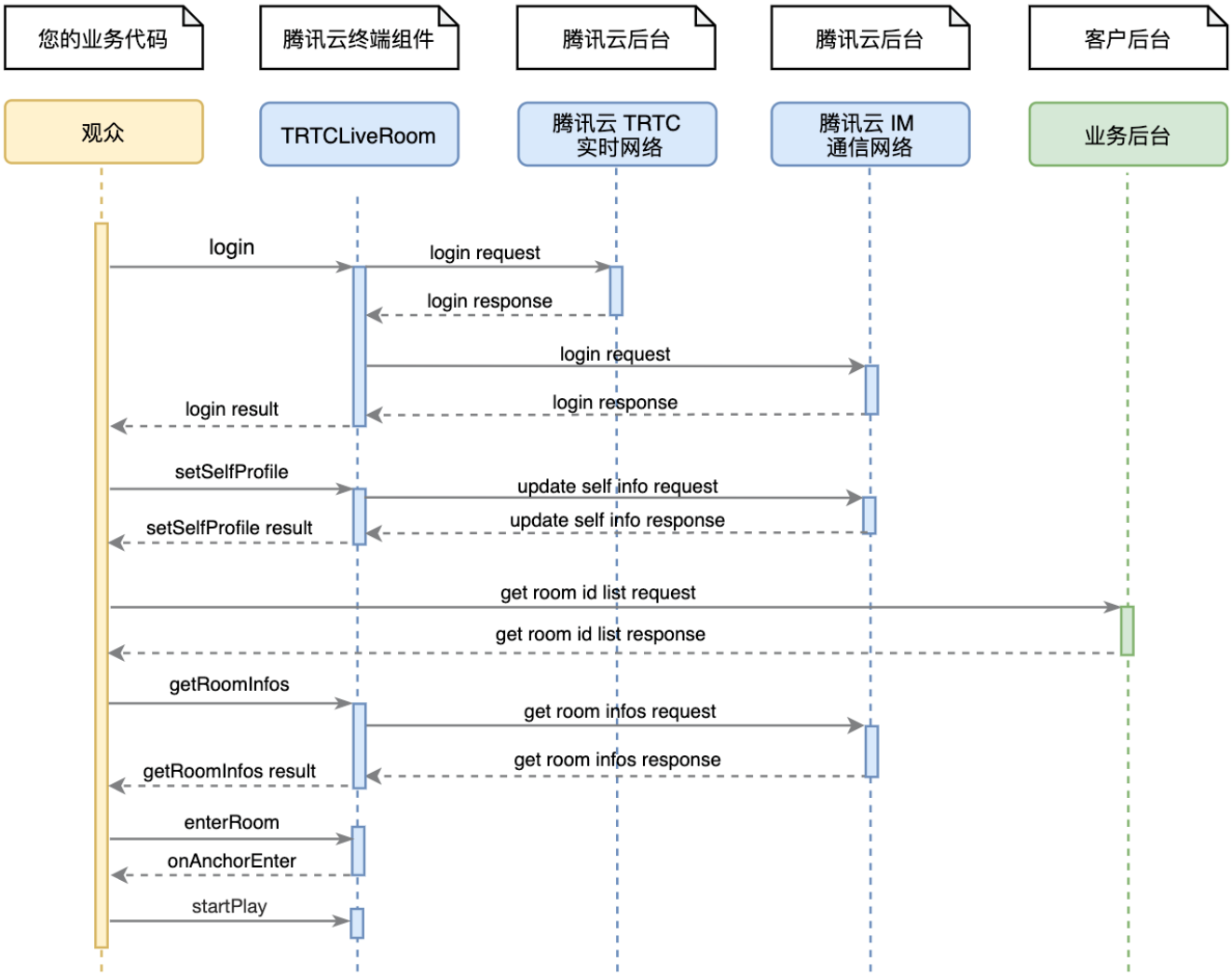
1. 观众端执行 步骤4 登录后，可以调用 setSelfProfile 设置自己的昵称和头像。
2. 观众端向业务后台获取最新的直播房间列表。

> **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。

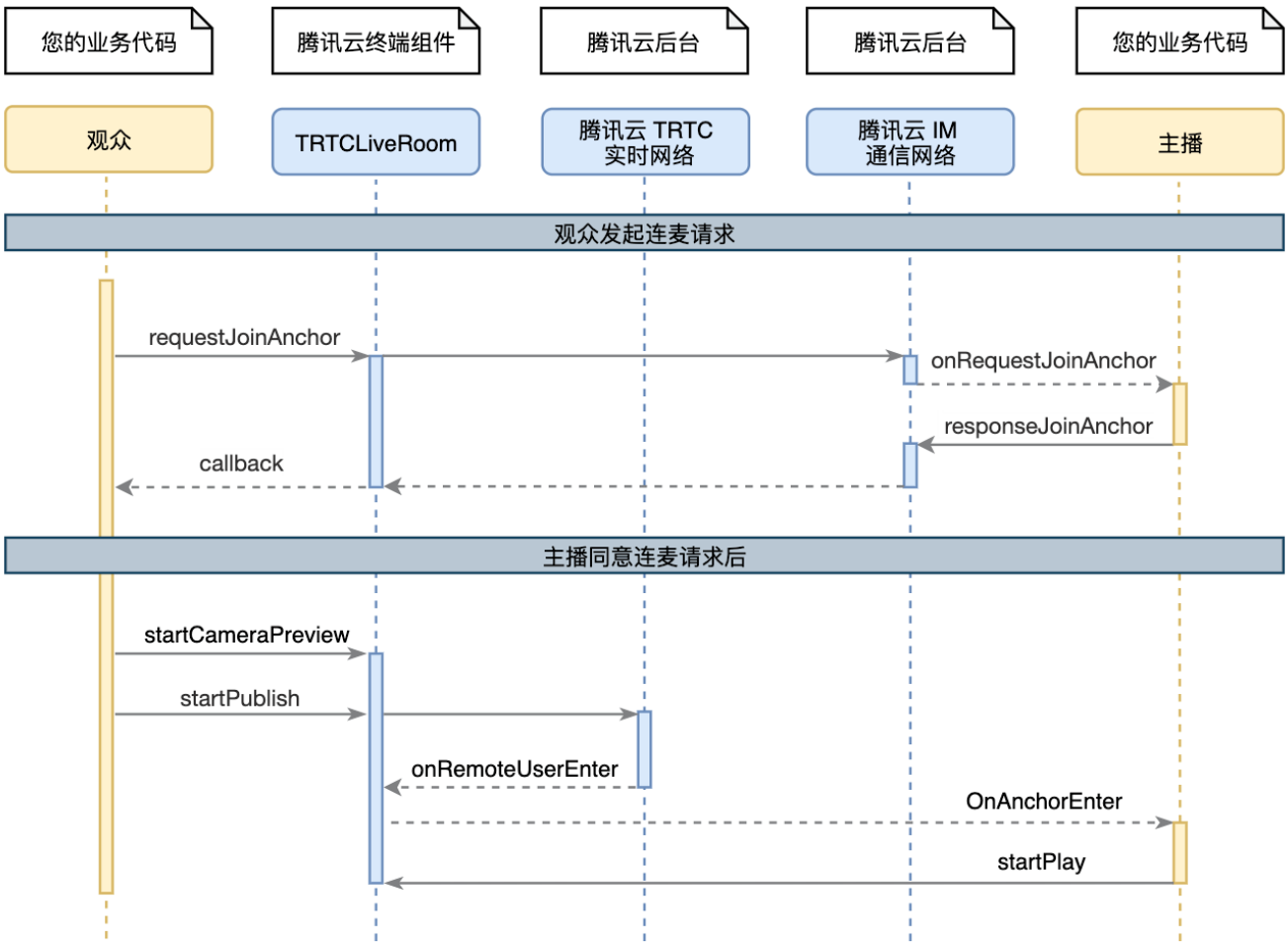


步骤7：观众与主播连麦上下麦

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。

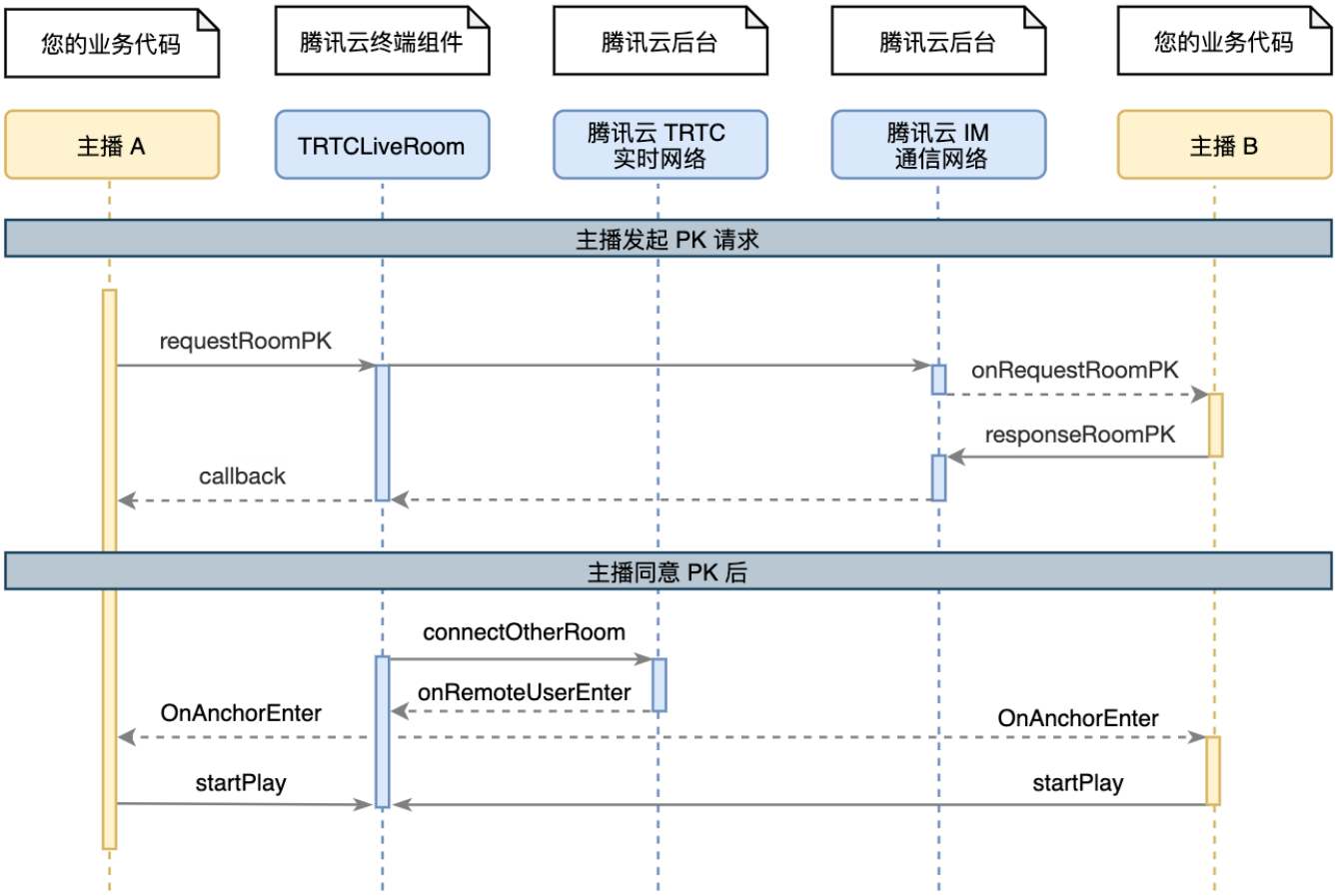
版权所有：腾讯云计算（北京）有限责任公司

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。



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。

集成 TUIPusher&TUIPlayer (Web)

最近更新时间：2022-06-02 15:26:25

组件介绍

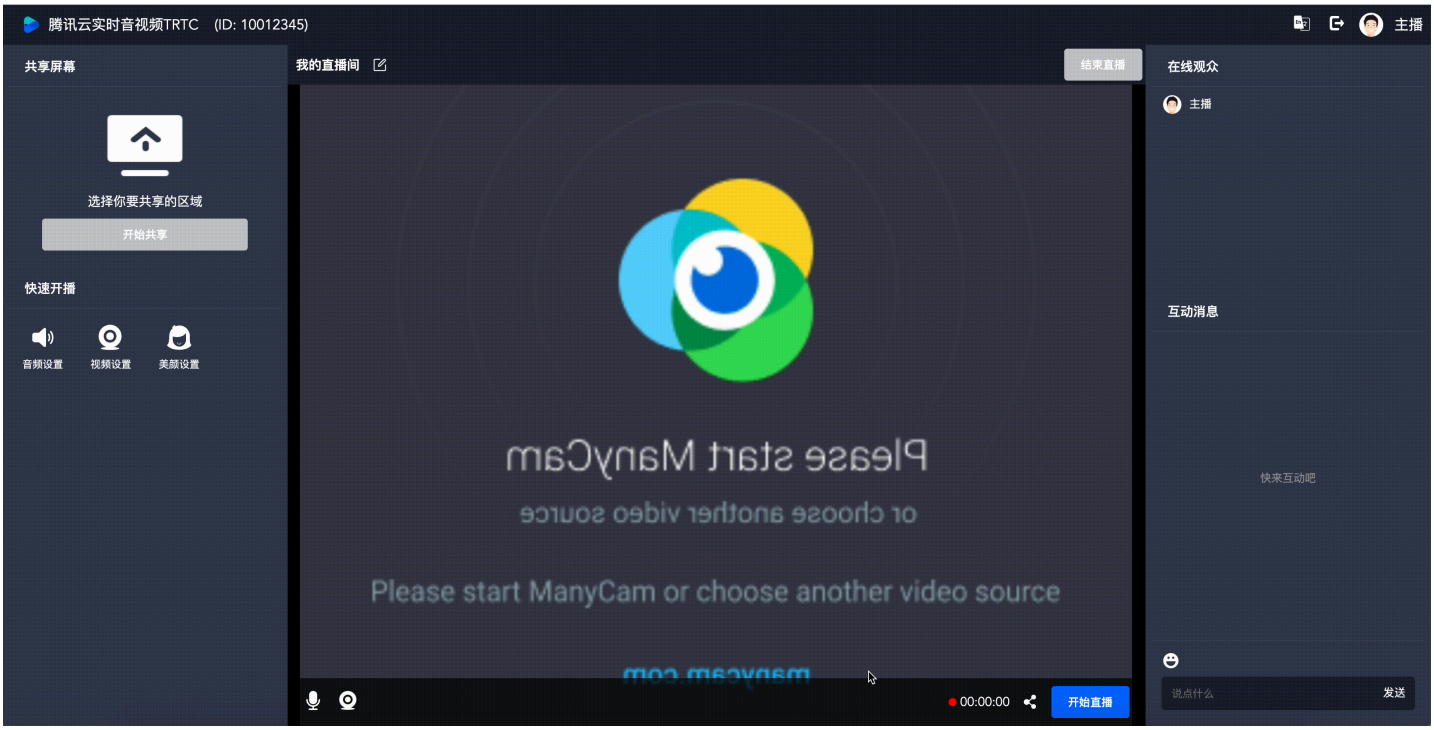
TUIPusher & TUIPlayer 是 Web 端开源的含 UI 直播互动组件。TUIPusher & TUIPlayer 集成 实时音视频 TRTC 、 即时通信 IM 等基础 SDK，为企业直播、电商带货、行业培训、远程教学等多种直播场景提供快速上线 Web 端直播推拉流工具的解决方案。

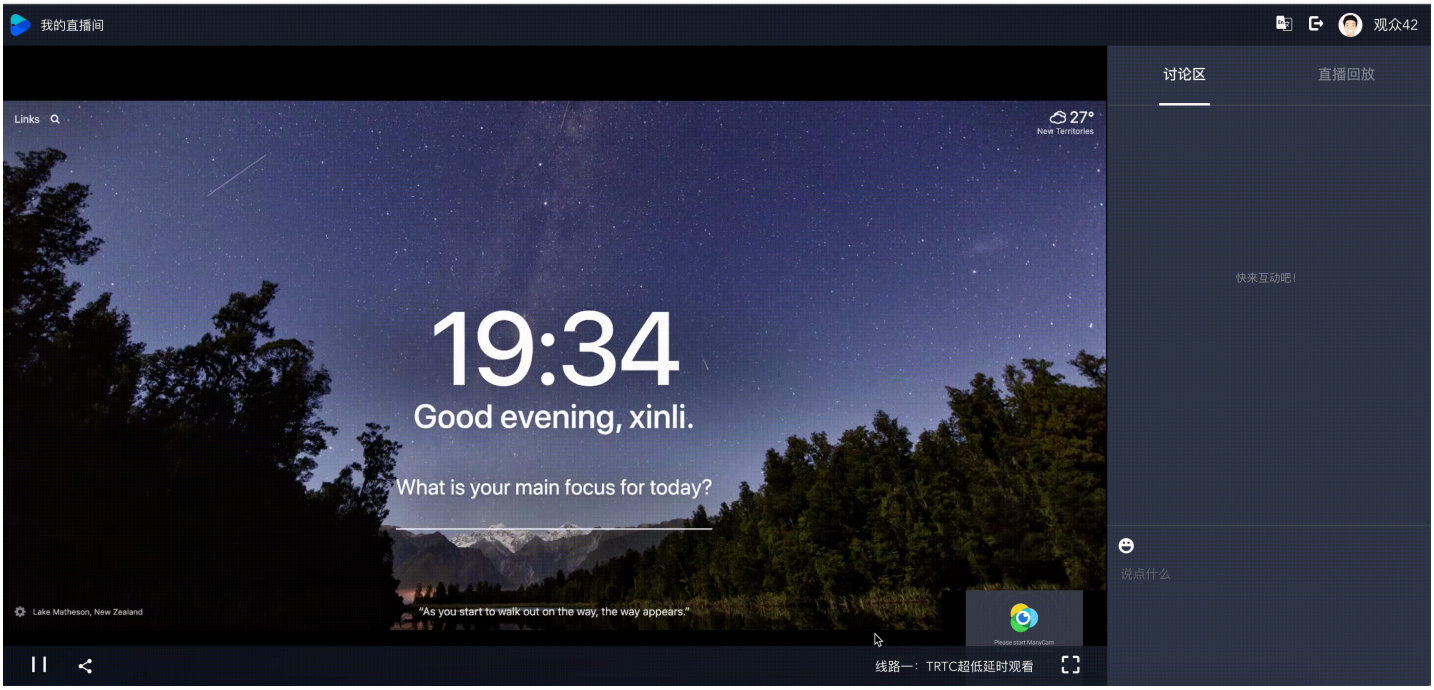
> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TUIPusher & TUIPlayer 的优势：

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

版权所有：腾讯云计算（北京）有限责任公司





快速体验

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

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

TUIPusher 推流组件功能介绍

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

TUIPlayer 拉流组件功能介绍

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

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

组件集成

步骤一：开通腾讯云服务

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

方式1：基于实时音视频

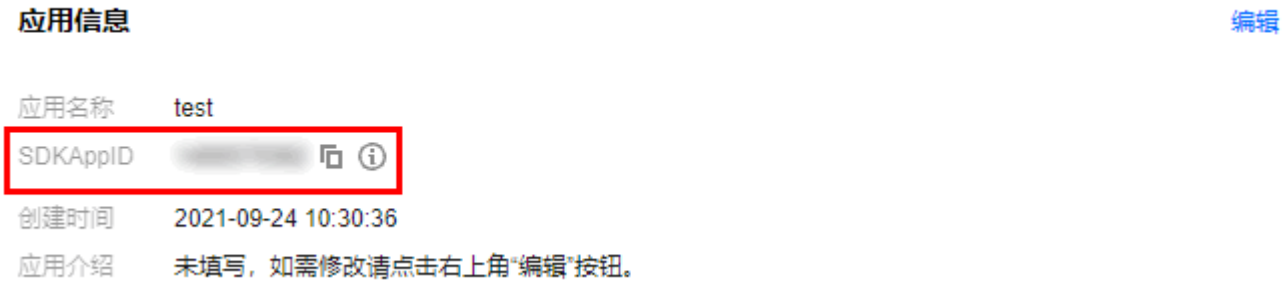
步骤1：创建实时音视频 TRTC 应用

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



步骤2: 获取 TRTC 密钥信息

1. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。



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



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

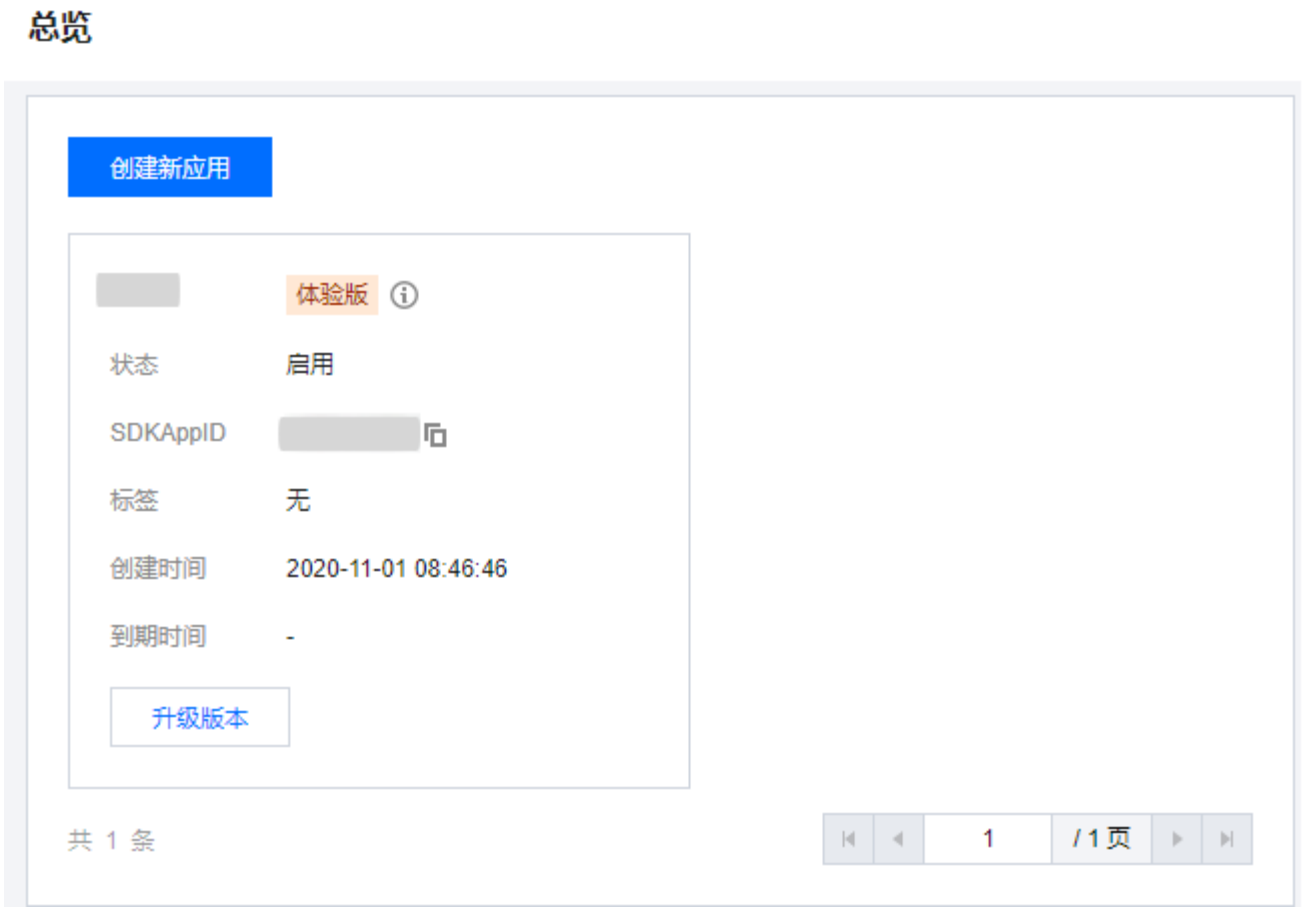
方式2：基于即时通信 IM

步骤1：创建即时通信 IM 应用

1. 登录 即时通信 IM 控制台，单击 **创建新应用** 将弹出对话框。



2. 输入您的应用名称，单击 **确认** 即可完成创建。



3. 您可在 即时通信 IM 控制台 总览页面查看新建应用的状态、业务版本、SDKAppID、创建时间以及到期时间。请记录 SDKAppID 信息。

步骤2：获取 IM 密钥并开通实时音视频服务

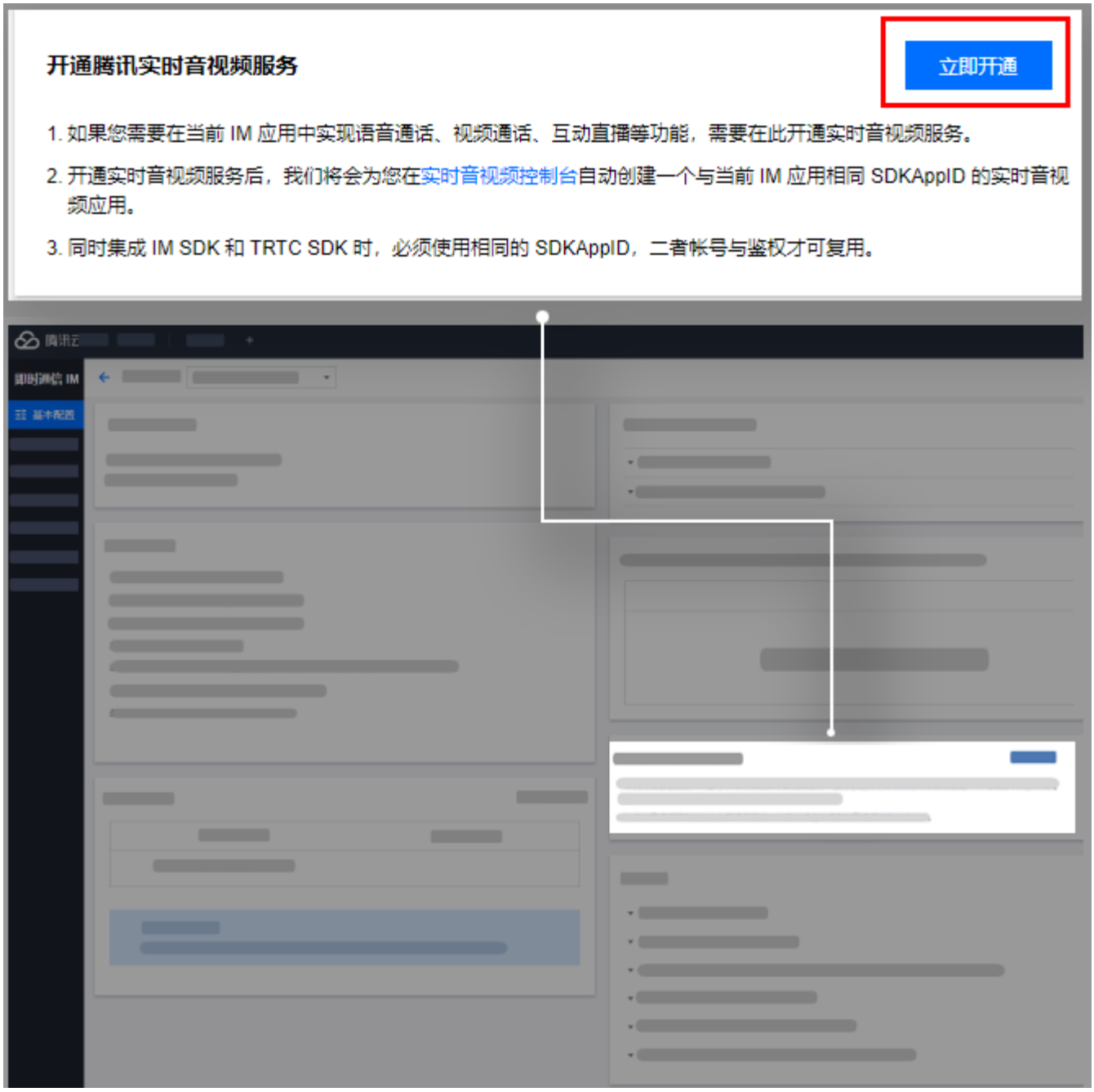
1. 在 即时通信 IM 控制台 总览页单击您创建完成的即时通信 IM 应用，随即跳转至该应用的基础配置页。在 **基本信息** 区域，单击 **显示密钥**，复制并保存密钥信息。



注意：

请妥善保管密钥信息，谨防泄露。

2. 在该应用的基础配置页，开通腾讯云实时音视频服务。



步骤二：项目准备

1. 在 GitHub 下载 TUIPusher & TUIPlayer 代码。
2. 为 TUIPusher & TUIPlayer 安装依赖。

```
cd Web/TUIPusher
npm install

cd Web/TUIPlayer
npm install
```

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

```
/**
 * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
 *
 * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId，
 * 它是腾讯云用于区分客户的唯一标识。
 */
export const sdkAppId = 0;
/**
 * 签名过期时间，建议不要设置的过短
 * <p>
 * 时间单位：秒
 * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
 */
export const expireTime = 604800;

/**
 * 计算签名用的加密密钥，获取步骤如下：
 *
 * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一
 * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
 * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
 *
 * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄
 * 文档：https://cloud.tencent.com/document/product/647/17275#Server
 */
export const secretKey = '';
```

4. 本地开发环境运行 TUIPusher & TUIPlayer。

```
cd Web/TUIPusher
npm run serve

cd Web/TUIPlayer
npm run serve
```

5. 可打开 http://localhost:8080 和 http://localhost:8081 体验 TUIPusher 和 TUIPlayer 功能。
6. 可更改 TUIPusher/src/config/basic-info-config.js 及 TUIPlayer/src/config/basic-info-config.js 配置文件中的房间，主播及观众等信息，**注意保持 TUIPusher 和 TUIPlayer 的房间信息，主播信息一致**。

> ⚠ 注意：
>
> - 完成以上配置，您可以使用 TUIPusher & TUIPlayer 进行超低延时直播，如您需要支持快直播和标准直播，请继续阅读 步骤三：旁路直播。

- 本地计算 UserSig 的方式仅用于本地开发调试，请勿直接发布到线上，一旦您的 SECRETKEY 泄露，攻击者就可以盗用您的腾讯云流量。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤三：旁路直播

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

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



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

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

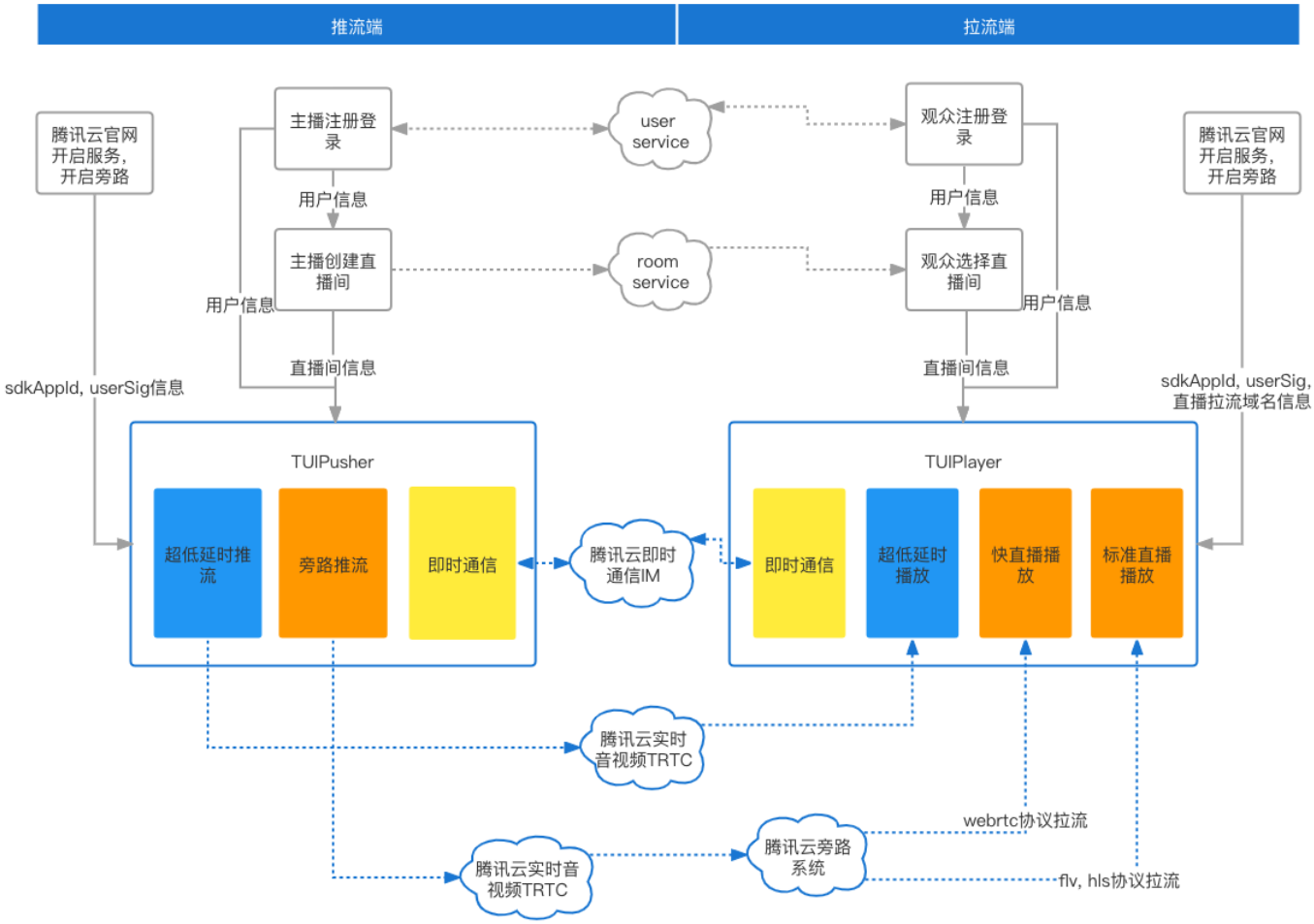
步骤四：生产环境应用

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

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

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

- 参考 TUIPusher/src/pusher.vue 及 TUIPlayer/src/player.vue 文件，将用户信息、直播间信息、SDKAppId 及 UserSig 等账号信息提交到 vuex 的 store 进行全局存储，您就可以跑通推拉流两个客户端的所有功能。详细业务流程参见下图：



相关问题

Web 端如何实现美颜功能?

请参见 开启美颜。

Web 端如何实现屏幕共享?

请参见 屏幕分享。

Web 端如何实现云端录制?

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

Web 端如何实现推流到 CDN？

Web 端推流到 CDN 请参见 实现推流到 CDN。

Web 端如何实现快直播拉流?

实现快直播拉流的方式是通过 Web SDK 推流到 CDN 之后使用 WebRTC 协议拉流，具体请参见参见 快直播拉流 > Web（H5）播放器。

Web 端如何实现标准直播拉流?

实现标准直播直播拉流，请参见 标准直播拉流 > Web（H5）播放器。

注意事项

平台支持

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

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 QQ 浏览器（极速内核）	10.4+	支持	支持	不支持
Windows	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Windows	桌面版 Edge 浏览器	80+	支持	支持	支持
Windows	桌面版微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版企业微信内嵌网页	-	支持	不支持	不支持
iOS	微信内嵌浏览器	-	支持	不支持	不支持
iOS	企业微信内嵌浏览器	-	支持	不支持	不支持
iOS	移动版 Safari 浏览器	-	支持	不支持	不支持
iOS	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	微信内嵌浏览器	-	支持	不支持	不支持
Android	企业微信浏览器	-	支持	不支持	不支持
Android	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	移动版 QQ 浏览器	-	支持	不支持	不支持
Android	移动版 Firefox 浏览器	-	支持	不支持	不支持

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
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。

版权所有：腾讯云计算（北京）有限责任公司

如果有任何需要或者反馈，可扫描下方二维码，或者单击 反馈链接 同步给我们。

此外，我们欢迎加入 TUI 组件使用交流 QQ 群（群号：592465424）进行技术交流和问题反馈。



TUILiveRoom API 查询
TRTCLiveRoom API (iOS)

最近更新时间：2022-06-02 15:27:22

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
delegate	设置事件回调。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
getAudienceList	获取房间内所有的观众信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播踢除连麦观众。

主播跨房间 PK

API	描述

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。

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

delegate

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

```
@property(nonatomic, weak)id<TRTCLiveRoomDelegate> delegate;
```

> 说明：
> delegate 是 TRTCLiveRoom 的代理回调。

login

登录。

```
/// 登录到组件系统
/// - Parameters:
/// - sdkAppID: 您可以在实时音视频控制台 > 【[应用管理](https://console.cloud.tencent.com/trtc/app)】> 应用信息中查看 SDKAppID。
/// - userID: 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（\_）。
/// - userSig: 腾讯云设计的一种安全保护签名，获取方式请参见 [如何计算及使用 UserSig](https://cloud.tencent.com/document/product/647/17275)。
```

```
/// - config: 全局配置信息，请在登录时初始化，登录之后不可变更。 isAttachedTUIKit 项目中是否引入并使用TUIKit
/// - callback: 登录回调，成功时 code 为0。
/// - Note:
/// - userSig 建议设定 7 天，能够有效规避 usersign 过期导致的 IM 收发消息失败、TRTC 连麦失败等情况
- (void)loginWithSdkAppID:(int)sdkAppID
userID:(NSString *)userID
userSig:(NSString *)userSig
config:(TRTCLiveRoomConfig *)config
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(login(sdkAppID:userID:userSig:config:callback:));
```

参数如下表所示：

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

logout

登出。

```
// 退出登录
/// - Parameter callback: 登出回调，成功时 code 为0
- (void)logout:(Callback _Nullable)callback
NS_SWIFT_NAME(logout(_:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
/// 设置用户信息，您设置的用户信息会被存储于腾讯云 IM 云服务中。
/// - Parameters:
/// - name: 用户昵称
/// - avatarURL: 用户头像地址
/// - callback: 个人信息设置回调，成功时 code 为0
- (void)setSelfProfileWithName:(NSString *)name
avatarURL:(NSString * _Nullable)avatarURL
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(setSelfProfile(name:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
name	String	昵称。
avatarURL	String	头像地址。
callback	(_ code: Int, _ message: String?) -> Void	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
/// 创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
/// 主播开播的正常调用流程是：
/// 1.【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
/// 2.【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 callback 通知给主播。
/// 3.【主播】调用 startPublish() 开始推流。
/// - Parameters:
/// - roomID: 房间标识，需要由您分配并进行统一管理。多个 roomid 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
/// - roomParam: TRTCCreateRoomParam | 房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
/// - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播开始直播的时候调用，可重复创建自己已创建过的房间。
- (void)createRoomWithRoomID:(UInt32)roomID
roomParam:(TRTCCreateRoomParam *)roomParam
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
callback	(_ code: Int, _ message: String?) -> Void	创建房间的结果回调，成功时 code 为0。

主播开播的正常调用流程如下：

1. 【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
2. 【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 callback 通知给主播。
3. 【主播】调用 starPublish() 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
/// 销毁房间（主播调用）
/// 主播在创建房间后，可以调用这个函数来销毁房间。
/// - Parameter callback: 销毁房间的结果回调，成功时 code 为0。
/// - Note:
/// - 主播在创建房间后，可以调用该函数来销毁房间。
- (void)destroyRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（观众调用）。

```
/// 进入房间（观众调用）
/// 观众观看直播的正常调用流程是：
/// 1.【观众】向您的服务端获取最新的直播间列表，其中有多个直播间的 roomid 和房间信息。
/// 2.【观众】观众选择一个直播间以后，调用 enterRoom() 进入该房间。
/// 3.【观众】如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom() 成功后调用 startPlay(userID) 即可播放主播的画面。
/// 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
/// - Parameters:
/// - roomID: 房间标识。
/// - useCDNFirst: 是否优先使用CDN播放
/// - cdnDomain: CDN域名
/// - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
/// - 观众进入直播房间的时候调用
/// - 主播不可调用这个接口进入自己已创建的房间，而要用createRoom
```

```
- (void)enterRoomWithRoomID:(UInt32)roomID
useCDNFirst:(BOOL)useCDNFirst
cdnDomain:(NSString * _Nullable)cdnDomain
callback:(Callback)callback
NS_SWIFT_NAME(enterRoom(roomID:useCDNFirst:cdnDomain:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识。
callback	(_ code: Int, _ message: String?) -> Void	进入房间的结果回调，成功时 code 为 0。

观众观看直播的正常调用流程如下：

1. **【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。**
2. **【观众】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

参数	类型	含义
streamID	String	用于绑定直播 CDN 的 streamID，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamID。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用。
- 观众结束连麦时调用。

```
/// 停止直播（推流），适用于如下两种场景：
/// 1. 主播结束直播时调用
/// 2. 观众结束连麦时调用
/// - Parameter callback: 操作回调。
- (void)stopPublish:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPublish(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
/// 播放远端视频画面，可以在普通观看和连麦场景中调用
///【普通观看场景】
/// 1. 如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom() 成功后调用 startPlay(userID) 即可播放主播的画面。
/// 2. 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
```

```
/// 【直播连麦场景】
/// 发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调，此时使用回调中的 userID 调用 startPlay(userID) 即可播放连麦画面。
/// - Parameters:
/// - userID: 需要观看的用户 ID。
/// - view: 承载视频画面的 view 控件。
/// - callback: 操作回调。
- (void)startPlayWithUserID:(NSString *)userID
view:(UIView *)view
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startPlay(userID:view:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	需要观看的用户 ID。
view	UIView	承载视频画面的 view 控件。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

普通观看场景

- 若直播间列表已包含主播端的 userID 信息，观众端可以直接在 enterRoom() 成功后调用 startPlay(userID) 播放主播的画面。
- 若在进房前暂未获取主播的 userID，观众端在进房后会收到 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 的事件回调，该回调中携带主播的 userID 信息，再调用startPlay(userID)即可播放主播的画面。

直播连麦场景

发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调，此时使用回调中的 userID 调用 startPlay(userID) 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 onAnchorExit() 回调时，调用该接口。

```
/// 停止渲染远端视频画面
/// - Parameters:
/// - userID: 对方的用户信息。
```

```
/// - callback: 操作回调。
/// - Note:
/// - 在 onAnchorExit 回调时，调用这个接口
- (void)stopPlayWithUserID:(NSString *)userID
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPlay(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	对方的用户信息。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
/// 观众端请求连麦
/// - Parameters:
/// - reason: 连麦请求原因。
/// - responseCallback: 请求连麦的回调。
/// - Note: 观众发起请求后，主播端会收到`onRequestJoinAnchor`回调
- (void)requestJoinAnchor:(NSString *)reason
timeout:(double)timeout
responseCallback:(ResponseCallback _Nullable)responseCallback
NS_SWIFT_NAME(requestJoinAnchor(reason:timeout:responseCallback:));
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	long	主播响应回调。
responseCallback	(_ agreed: Bool, _ reason: String?) -> Void	主播响应回调。

主播和观众的连麦流程如下：

1. 【观众】调用 requestJoinAnchor() 向主播发起连麦请求。
2. 【主播】会收到 TRTCLiveRoomDelegate 的 onRequestJoinAnchor() 回调通知。
3. 【主播】调用 responseJoinAnchor() 决定是否接受来自观众的连麦请求。
4. 【观众】会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 【观众】如果请求被同意，则调用 startCameraPreview() 开启本地摄像头。
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。

参数	类型	含义
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

参数	类型	含义
userID	String	远端的用户 ID。
isMuted	Bool	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
/// 静音所有远端音频
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteAllRemoteAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteAllRemoteAudio(_:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

setAudioQuality

设置音频质量

```
/// 设置音频质量，支持的值为1 2 3，代表低中高
/// - Parameter quality 音频质量
- (void)setAudioQuality:(NSInteger)quality
NS_SWIFT_NAME(setAudioiQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	1：语音；2：标准； 3：音乐。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
/// 获取音效管理对象
- (TXAudioEffectManager *)getAudioEffectManager;
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
/* 获取美颜管理对象 TXBeautyManager
*
* 通过美颜管理，您可以使用以下功能：
* - 设置"美颜风格"、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
* - 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”
* - 设置人脸挂件（素材）等动态效果
* - 添加美妆
* - 进行手势识别
*/
- (TXBeautyManager *)getBeautyManager;
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
/// 发送文本消息，房间内所有成员都可见
/// - Parameters:
```

```
/// - message: 文本消息。
/// - callback: 发送回调。
- (void)sendRoomTextMsg:(NSString *)message callback:(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
/// 发送自定义消息
/// - Parameters:
/// - command: 命令字，由开发者自定义，主要用于区分不同消息类型
/// - message: 本文消息。
/// - callback: 发送回调。
- (void)sendRoomCustomMsgWithCommand:(NSString *)command message:(NSString *)messa
ge callback:(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomCustomMsg(command:message:callback:));
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示 debug 信息。

```
/// 是否在界面中展示debug信息
/// - Parameter isShow: 开启/关闭 Debug 信息显示。
- (void)showVideoDebugLog:(BOOL)isShow
NS_SWIFT_NAME(showVideoDebugLog(_:));
```

参数如下表所示：

参数	类型	含义
isShow	Bool	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

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

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onError:(NSInteger)code
message:(NSString *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onError:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码。

参数	类型	含义
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。

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
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。

参数	类型	含义
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。

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。

```
- (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。

TRTCLiveRoom API (Android)

最近更新时间：2022-06-02 15:27:02

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
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

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。

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。

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。
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.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

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

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
public abstract void startCameraPreview(boolean isFront, TXCloudVideoView view, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。
view	TXCloudVideoView	承载视频画面的控件。
callback	ActionCallback	操作回调。

stopCameraPreview

停止本地视频采集及预览。

```
public abstract void stopCameraPreview();
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用
- 观众开始连麦时调用

```
public abstract void startPublish(String streamId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
streamId	String	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
callback	ActionCallback	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- **主播结束直播时调用**
- **观众结束连麦时调用**

```
public abstract void stopPublish(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
public abstract void startPlay(String userId, TXCloudVideoView view, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要观看的用户id。
view	TXCloudVideoView	承载视频画面的 view 控件。
callback	ActionCallback	操作回调。

普通观看场景

- 若直播间列表已包含主播端的 userId 信息，观众端可以直接在 enterRoom() 成功后调用 startPlay(userId) 播放主播的画面。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 的事件回调，该回调中携带主播的 userId 信息，再调用startPlay(userId)即可播放主播的画面。

直播连麦场景

发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 回调，此时使用回调中的 userId 调用 startPlay(userId) 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 onAnchorExit() 回调时，调用该接口。

```
public abstract void stopPlay(String userId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	对方的用户信息。
callback	ActionCallback	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
public abstract void requestJoinAnchor(String reason, int timeout, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	int	超时时间。

参数	类型	含义
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 callback);
```

参数如下表所示：

参数	类型	含义
userId	String	连麦观众 ID。
callback	ActionCallback	操作回调。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
public abstract void requestRoomPK(int roomId, String userId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。
callback	ActionCallback	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 【主播 A】调用 requestRoomPK() 向主播 B 发起连麦请求。
2. 【主播 B】会收到 TRTCLiveRoomDelegate 的 onRequestRoomPK() 回调通知。
3. 【主播 B】调用 responseRoomPK() 决定是否接受主播 A 的 PK 请求。
4. 【主播 B】如果接受主播 A 的要求，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 来显示主播 A 的视频画面。
5. 【主播 A】会收到 responseCallback 回调通知，该通知会携带来自主播 B 的处理结果。
6. 【主播 A】如果请求被同意，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 requestRoomPK 传入的 responseCallback 回调。

```
public abstract void responseRoomPK(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 TRTCLiveRoomDelegate 的 onQuitRoomPk() 回调通知。

```
public abstract void quitRoomPK(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
public abstract void switchCamera();
```

setMirror

设置是否镜像展示。

```
public abstract void setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	boolean	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
public abstract TXBeautyManager getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示debug信息。

```
public abstract void showVideoDebugLog(boolean isShow);
```

参数如下表所示：

参数	类型	含义
isShow	boolean	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

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

```
void onError(int code, String message);
```

参数如下表所示：

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

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
void onRoomInfoChange(TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 TRTCLiveRoom 的 startPlay() 显示该主播的视频画面。

```
void onAnchorEnter(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 TRTCLiveRoom 的 stopPlay() 关闭该主播的视频画面。

```
void onAnchorExit(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
void onAudienceExit(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

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

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
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);
```

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
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。

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：2022-06-02 15:27:44

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
registerListener	设置事件回调。
unRegisterListener	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
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

版权所有：腾讯云计算（北京）有限责任公司

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。

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
onRequestRoomPK	收到请求跨房 PK 通知。
onRoomPKAccepted	主播接受跨房 PK 请求。
onRoomPKRejected	主播拒绝跨房 PK 请求。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom 单例对象。

```
static Future<TRTCLiveRoom> sharedInstance()
```

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
static void destroySharedInstance()
```

registerListener

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

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

> 说明：
> registerListener 是 TRTCLiveRoom 的代理回调。

unRegisterListener

移除组件事件监听接口。

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

login

登录。

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

参数如下表所示：

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

logout

登出。

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

setSelfProfile

修改个人信息。

```
Future<ActionCallback> setSelfProfile(String userName, String avatarURL);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
Future<ActionCallback> createRoom(int roomId, TRTCCreateRoomParam roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	RoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。

主播开播的正常调用流程如下：

1. 【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
2. 【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 ActionCallback 通知给主播。
3. 【主播】调用 starPublish() 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

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

enterRoom

进入房间（观众调用）。

```
Future<ActionCallback> enterRoom(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。

观众观看直播的正常调用流程如下：

1. 【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 【观众】观众选择一个直播间，并调用 enterRoom() 进入该房间。
3. 【观众】调用 startPlay(userId) 并传入主播的 userId 开始播放。
   - 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 startPlay(userId) 即可开始播放。
   - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 的事件回调，该回调中携带主播的 userId 信息，再调用 startPlay(userId) 即可播放。

exitRoom

离开房间。

```
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();
```

版权所有：腾讯云计算（北京）有限责任公司

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
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

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

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

参数如下表所示：

参数	类型	含义
errCode	int	错误码。
errMsg	String	错误信息。

onWarning

警告回调。

参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。

onKickedOffline

其他用户登录了同一账号，被踢下线。

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。
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。

参数	类型	含义
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。

参数	类型	含义
userAvatar	String	用户头像地址。

onRoomPKAccepted

主播接受跨房 PK 请求。

参数如下表所示：

参数	类型	含义
userId	String	接收跨房 PK 的用户 ID。

onRoomPKRejected

主播接受跨房 PK 请求。

参数如下表所示：

参数	类型	含义
userId	String	拒绝跨房 PK 的用户 ID。

onQuitRoomPK

收到断开跨房 PK 通知。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

参数如下表所示：

参数	类型	含义
message	String	文本消息。

onRecvRoomCustomMsg

收到自定义消息。

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

在线 K 歌
集成 TUIKaraoke (iOS)

最近更新时间：2022-06-23 17:12:21

组件介绍

TUIKaraoke 是一个开源的音视频 UI 组件，通过在项目中集成 TUIKaraoke 组件，您只需要编写几行代码就可以为您的应用添加在线 K 歌场景，体验 K 歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。TUIKaraoke 同时支持 Android 平台的源代码，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIKaraoke 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 iOS目录下的 Source 、 Resources 、 TXAppBasic 文件夹和 TUIKaraoke.podspec 文件到您的工程中，并完成如下导入动作：

- 在您的 Podfile 文件内添加导入命令，参考如下：

```
pod 'TUIKaraoke', :path => "./", :subspecs => ["TRTC"]
pod 'TXLiteAVSDK_TRTC'
pod 'TXAppBasic', :path => "TXAppBasic/"
```

- 打开终端，进入 Podfile 文件所在目录下执行安装命令，参考如下：

```
pod install
```

步骤二：配置权限

在您的工程的 info.plist文件中配置 App 的权限，SDK 需要以下权限（iOS 系统需要动态申请麦克风）：

```
<key>NSMicrophoneUsageDescription</key>
<string>Karaoke 需要访问您的麦克风权限</string>
```

步骤三：初始化并登录

相关接口说明请参见 TUIKaraoke。

```
// 1.初始化
let karaokeRoom = TRTCKaraokeRoom.shared()
karaokeRoom.setDelegate(delegate: self)
// 2.登录
karaokeRoom.login(SDKAppID: Int32(SDKAppID), UserId: UserId, UserSig: ProfileManager.share
d.curUserSig()) { code, message in
if code == 0 {
//登录成功
}
}
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现在线KTV场景

1. 主播创建房间 TUIKaraoke.createRoom

```
int roomId = "房间ID";
let param = RoomParam.init()
param.roomName = "房间名称";
param.needRequest = false; // 上麦是否需要房主确认
param.seatCount = 8; // 房间座位数，一共8个座位
param.coverUrl = "房间封面图URL";

karaokeRoom.createRoom(roomID: Int32(roomInfo.roomID), roomParam: param) { [weak self
] (code, message) in
guard let `self` = self else { return }
if code == 0 {
```

```
//创建成功
}
}
```

2. 听众进入房间 TUIKaraoke.enterRoom

```
karaokeRoom.enterRoom(roomID: roomInfo.roomID) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
//进房成功
}
}
```

3. 听众主动上麦 TUIKaraoke.enterSeat

```
// 1.听众调用上麦
int seatIndex = 1;
karaokeRoom.enterSeat(seatIndex: seatIndex) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
//上麦成功
}
}
// 2.收到 onSeatListChange 回调，刷新麦位列表
func onSeatListChange(seatInfoList: [SeatInfo]) {
}
```

> 说明：
> 其他关于麦位管理的相关操作，您可参考 TUIKaraoke接口文档 按需实现，或者可以参考 TUIKaraoke 示例工程。

4. 实现音乐播放并体验 KTV 场景

您可以根据自己的业务获取音乐 ID 和 URL 链接，播放歌曲。详情请参见 TUIKaraoke 音乐播放接口。

```
//播放音乐
karaokeRoom.startPlayMusic(musicID: musicID, originalUrl: muscicLocalPath, accompanyUrl:
```

```
accompanyLocalPath);
//停止音乐
karaokeRoom.stopPlayMusic();
```

完成以上步骤，就可以实现 KTV 基本功能。如果您的业务还需要聊天、发送礼物等功能，可以接入以下能力。

步骤六：文字聊天功能（可选）

如果您需要实现各主播或听众之间文字聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口说明请参见 TRTCKaraokeRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
karaokeRoom.sendRoomTextMsg(message: message) { [weak self] (code, message) in
if code == 0 {
//发送成功
}
}
// 接收端：监听文本消息
karaokeRoom.setDelegate(delegate: self)
func onRecvRoomTextMsg(message: String, userInfo: UserInfo) {
debugPrint("收到来自：" + userInfo.userName + "的消息：" + message)
}
```

步骤七：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送礼物或接收礼物并展示。

```
// 发送端：通过自定义 "IMCMD_GIFT" 来区分礼物消息
karaokeRoom.sendRoomCustomMsg(cmd: kSendGiftCmd, message: message) { code, msg in
if (code == 0) {
//发送成功
}
}

// 接收端：监听礼物消息
karaokeRoom.setDelegate(delegate: self)
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: UserInfo) {
if cmd == kSendGiftCmd {
debugPrint("收到来自：" + userInfo.userName + "的礼物：" + message)
```

```
}
}
```

常见问题

TUIKaraoke 组件支持变声、变调、混响等音效功能吗?

支持，具体请参见 TUIKaraoke 示例工程。

> 说明：
> 更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：592465424，进行技术交流和反馈。

集成 TUIKaraoke (Android)

最近更新时间：2022-06-23 17:12:24

组件介绍

TUIKaraoke 是一个开源的音视频 UI 组件，通过在项目中集成 TUIKaraoke 组件，您只需要编写几行代码就可以为您的应用添加在线 K 歌场景，体验 K 歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。TUIKaraoke 同时支持 iOS 平台的源代码，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIKaraoke 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android目录下的 Source 和 Debug 目录到您的工程中，并完成如下导入动作：

- 在 setting.gradle 中完成导入，参考如下：

```
include ':Source'
include ':Debug'
```

- 在 app 的 build.gradle 文件中添加对 TUIKaraoke 的依赖：

```
api project(':Source')
```

- 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

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

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" /> // 使用场景：悬浮窗功能需要此权限；
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" /> // 使用场景：使用蓝牙耳机时需要此权限；
```

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

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

步骤三：初始化并登录

相关接口详情请参见 TUIKaraoke。

```
// 1.初始化
TRTCKaraokeRoom mTRTCKaraokeRoom = TRTCKaraokeRoom.sharedInstance(this);
mTRTCKaraokeRoom.setDelegate(this);
// 2.登录
mTRTCKaraokeRoom.login(SDKAppID, UserID, UserSig, new TRTCKaraokeRoomCallback.ActionC
allback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。

- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现在线KTV场景

1. **主播创建房间 TUIKaraoke.createRoom**

```
int roomId = "房间ID";
TRTCKaraokeRoomDef.RoomParam roomParam = new TRTCKaraokeRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 8; // 房间座位数，一共8个座位
roomParam.coverUrl = "房间封面图URL";
mTRTCKaraokeRoom.createRoom(roomId, roomParam, new TRTCKaraokeRoomCallback.Actio
nCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//创建成功
}
}
});
```

2. **听众进入房间 TUIKaraoke.enterRoom**

```
mTRTCKaraokeRoom.enterRoom(roomId, new TRTCKaraokeRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
}
}
});
```

3. **听众主动上麦 TUIKaraoke.enterSeat**

```
// 1.听众调用上麦
int seatIndex = 1;
mTRTCKaraokeRoom.enterSeat(seatIndex, new TRTCKaraokeRoomCallback.ActionCallback()
{
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//上麦成功
}
}
});
// 2.收到 onSeatListChange 回调，刷新麦位列表
@Override
public void onSeatListChange(final List<TRTCKaraokeRoomDef.SeatInfo> seatInfoList) {
}
```

> **说明：**
> 其他关于麦位管理的相关操作，您可参考 TUIKaraoke接口文档 按需实现，或者可以参考我们的 TUIKaraoke示例工程。

4. **实现音乐播放并体验 KTV 场景**

您可以根据自己的业务获取音乐 ID 和 URL链接播放歌曲，接口详情请参见 TUIKaraoke 音乐播放接口。

```
//播放音乐
mTRTCKaraokeRoom.startPlayMusic(musicID,url);
//停止音乐
mTRTCKaraokeRoom.stopPlayMusic();
```

完成以上步骤，您可以实现KTV基本功能。如果您的业务还需要文字聊天、发送礼物等功能，可以接入以下能力。

步骤五：文字聊天功能（可选）

如果您需要实现各主播或听众之间文字聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口详情请参见 TRTCKaraokeRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
mTRTCKaraokeRoom.sendRoomTextMsg("Hello Word!", new TRTCKaraokeRoomCallback.ActionC
```

```
allback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//发送成功
}
}
});
// 接收端：监听文本消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
@Override
public void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

步骤六：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送或接收礼物并展示。

```
// 发送端：通过自定义 "CMD_GIFT" 来区分礼物消息
mTRTCKaraokeRoom.sendRoomCustomMsg("CMD_GIFT",date, new TRTCKaraokeRoomCallback.
ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//发送成功
}
}
});

// 接收端：监听礼物消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInf
o userInfo) {
if ("CMD_GIFT".equals(cmd)) {
// 收到礼物消息
```

```
Log.d(TAG, "收到来自" + userInfo.userName + "的礼物:" + message);
}
}
});
```

常见问题

TUIKaraoke 组件支持变声、变调、混响等音效功能吗?

支持，具体请参见 TUIKaraoke 示例工程。

> **说明：**
> 更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：**592465424**，进行技术交流和反馈。

TUIKaraoke API 查询
TRTCKaraoke(iOS)

最近更新时间：2022-06-02 15:28:13

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 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。

API	描述
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 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）。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述

版权所有：腾讯云计算（北京）有限责任公司

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。

版权所有：腾讯云计算（北京）有限责任公司

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。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
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（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(RoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 createRoom 创建新的 Karaoke 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。

2. 房主创建房间成功后，调用 enterSeat 进入座位。
3. 房主收到组件的 onSeatListChange 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 onAnchorEnterSeat 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Karaoke 房间列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个 Karaoke 房间，调用 enterRoom 并传入房间号即可进入该房间。
3. 进房后会收到组件的 onRoomInfoChange 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 onSeatListChange 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 onAnchorEnterSeat 的事件通知。

版权所有：腾讯云计算（北京）有限责任公司

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 createRoom() 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(KaraokeInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(KaraokeUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

音乐播放接口

startPlayMusic

播放音乐（上麦后调用）。

> 说明：
>
> - 播放音乐后，自身会收到 onMusicPrepareToPlay 的事件通知。
> - 音乐播放中，房间内所有成员会不断收到 onMusicProgressUpdate 的事件通知。
> - 音乐播放完成，自身会收到 onMusicCompletePlaying 的事件通知。

```
- (void)startPlayMusic:(int32_t)musicID originalUrl:(NSString *)originalUrl accompanyUrl:(NSStri
ng *)backingUrl NS_SWIFT_NAME(startPlayMusic(musicID:originalUrl:accompanyUrl:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	音乐的 ID。
originalUrl	String	原唱音乐的绝对路径。
accompanyUrl	String	伴奏音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

> 说明：
>
> 停止播放后，会收到 onMusicCompletePlaying 的事件通知。

```
- (void)stopPlayMusic NS_SWIFT_NAME(stopPlayMusic());
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

> 说明：
>
> - onMusicProgressUpdate 的事件通知会暂停
> - 不会收到 onMusicCompletePlaying 的事件通知。

```
- (void)pausePlayMusic NS_SWIFT_NAME(pausePlayMusic());
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> 说明：
>
> 不会收到 onMusicPrepareToPlay 的事件通知。

```
- (void)resumePlayMusic NS_SWIFT_NAME(resumePlayMusic());
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
>
> 上麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

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:conten
t: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_S
WIFT_NAME(acceptInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(rejectInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

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。

参数	类型	含义
mute	boolean	音量大小，取值：0 – 100。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSInteger)totalVolume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<KaraokeSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义

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

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

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。

TRTCKaraoke(Android)

最近更新时间：2022-06-02 15:28:21

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 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。

API	描述
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

版权所有：腾讯云计算（北京）有限责任公司

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	Karaoke 房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

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，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

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。

参数	类型	含义
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 callbac
k);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void leaveSeat(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> 说明：
> 房主抱人上麦，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
public abstract void pickSeat(int seatIndex, String userId, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 sendInvitation 向听众申请，收到 onInvitationAccept 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> 说明：
> 房主踢人下麦，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void kickSeat(int seatIndex, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> 说明：
> 静音/解除静音某个麦位，房间内所有成员会收到 onSeatListChange 和 onSeatMute 的事件通知。

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> 说明：
> 房主封禁/解禁对应麦位，房间内所有成员会收到 onSeatListChange 和 onSeatClose 的事件通知。

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK。

版权所有：腾讯云计算（北京）有限责任公司

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
userId	String	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。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCKaraokeRoomDelegate 事件回调

通用事件回调

onError

错误回调。

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

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。

参数	类型	含义
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)
;
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	下麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

音乐播放状态回调

onMusicPrepareToPlay

准备播放音乐的回调

```
void onMusicPrepareToPlay(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。

onMusicProgressUpdate

歌曲播放进度的回调

```
void onMusicProgressUpdate(int musicID, long progress, long total);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。
progress	long	当前播放时间，单位： ms。
total	long	总时间，单位： ms。

onMusicCompletePlaying

播放完成音乐的回调

```
void onMusicCompletePlaying(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。

实时合唱
集成 TUIChorus（Android）

最近更新时间：2022-06-23 17:12:17

组件介绍

TUIChorus 是一个开源的音视频 UI 组件，通过在项目中集成 TUIChorus 组件，您只需要编写几行代码就可以为您的应用添加在线合唱场景，体验低延时合唱、麦位管理、收发礼物、文字聊天等 TRTC 在合唱场景下的相关能力。TUIChorus 同时支持 iOS 平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIChorus 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android目录下的 Source 和 Debug 目录到您的工程中，并完成如下导入动作：

- 在 setting.gradle 中完成导入，参考如下：

```
include ':Source'
include ':Debug'
```

- 在 app 的 build.gradle 文件中添加对 TUIChorus 的依赖：

```
api project(':Source')
```

- 在根目录的build.gradle文件中添加TRTC SDK和IM SDK的依赖：

```
ext {
liteavSdk = "com.tencent.liteav:LiteAVSDK_Professional:latest.release"
imSdk = "com.tencent.imsdk:imsdk-plus:$IMSDK_VERSION"
}
```

步骤二：配置权限及混淆规则

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

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" /> // 使用场景：悬浮窗功能需要此权限；
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" /> // 使用场景：使用蓝牙耳机时需要此权限；
```

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

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

步骤三：初始化并登录

相关接口说明请参见 TUIChorus。

```
// 1.初始化
TRTCChorusRoom mTRTCChorusRoom = TRTCChorusRoom.sharedInstance(this);
mTRTCChorusRoom.setDelegate(delegate);
// 2.登录
mTRTCChorusRoom.login(SDKAppID, UserID, UserSig, new TRTCChorusRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。

- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：开通推拉流服务

体验 TUIChorus 合唱场景您需要开通腾讯云推拉流服务，用于主播端推流，听众端拉流。您可参考 推拉流 URL，生成自己的推拉流地址并保存，可保存为 pushUrl 和 playerUrl 。

步骤五：实现实时合唱场景

1. **主播创建房间 TUIChorus.createRoom**

```
int roomId = "房间ID";
TRTCChorusRoomDef.RoomParam roomParam = new TRTCChorusRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 2; // 房间座位数，这里一共2个座位，主播及副唱，其他人进房为听众
roomParam.mPushUrl = pushUrl; // 步骤三开通的推拉流地址
roomParam.mPlayUrl = playUrl;
roomParam.coverUrl = "房间封面图URL";
mTRTCChorusRoom.createRoom(roomId, roomParam, new TRTCChorusRoomCallback.ActionC
allback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//创建成功
}
}
});
```

2. **听众进入房间 TUIChorus.enterRoom**

```
mTRTCChorusRoom.enterRoom(roomId, new TRTCChorusRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//进房成功
```

```
}
}
});
```

3. 听众主动上麦成为副唱 TUIChorus.enterSeat

```
// 1.听众调用上麦
int seatIndex = 1;
mTRTCChorusRoom.enterSeat(seatIndex, new TRTCChorusRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//上麦成功
}
}
});
// 2.收到 onSeatListChange 回调，刷新麦位列表
@Override
public void onSeatListChange(final List<TRTCChorusRoomDef.SeatInfo> seatInfoList) {
}
```

> **说明：**
> 其他关于麦位管理的相关操作，您可参考 TUIChorus接口文档 按需实现，或者可以参考我们的 TUIChorus示例工程。

4. 实现音乐播放并合唱

您可以根据自己的业务获取音乐 ID 和 URL 链接播放歌曲并开始合唱，相关说明请参见 TUIChorus 音乐播放接口。

```
//播放音乐
mTRTCChorusRoom.startPlayMusic(musicID,url);
//停止音乐
mTRTCChorusRoom.stopPlayMusic();
```

完成以上步骤，就可以实现合唱基本功能。如果您的业务还需要聊天、发送礼物等功能，可以接入以下能力。

步骤六：文字聊天功能（可选）

如果您需要实现主播、副唱以及听众之间文字聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口详情请参见 TRTCChorusRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
mTRTCChorusRoom.sendRoomTextMsg("Hello Word!", new TRTCChorusRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//发送成功
}
}
});
// 接收端：监听文本消息
mTRTCChorusRoom.setDelegate(new TRTCChorusRoomDelegate() {
@Override
public void onRecvRoomTextMsg(String message, TRTCChorusRoomDef.UserInfo userInfo) {
Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
}
});
```

步骤七：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送或接收礼物并展示。

```
// 发送端：通过自定义 "IMCMD_GIFT" 来区分礼物消息
mTRTCChorusRoom.sendRoomCustomMsg("IMCMD_GIFT",date, new TRTCChorusRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//发送成功
}
}
});

// 接收端：监听礼物消息
```

```
mTRTCChorusRoom.setDelegate(new TRTCChorusRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCChorusRoomDef.UserInfo
userInfo) {
if ("IMCMD_GIFT".equals(cmd)) {
// 收到礼物消息
Log.d(TAG, "收到来自" + userInfo.userName + "的礼物:" + message);
}
}
});
```

常见问题

TUIChorus 组件支持变声、变调、混响等音效功能吗?

支持，具体请参见 TUIChorus 示例工程。

> **说明：**
> 更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：**592465424**，进行技术交流和反馈。

集成 TUIChorus（iOS）

最近更新时间：2022-06-23 17:12:11

组件介绍

TUIChorus 是一个开源的音视频 UI 组件，通过在项目中集成 TUIChorus 组件，您只需要编写几行代码就可以为您的应用添加在线合唱场景，体验低延时合唱、麦位管理、收发礼物、文字聊天等 TRTC 在合唱场景下的相关能力。TUIChorus 同时支持 Android 平台，基本功能如下图所示：

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIChorus 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 iOS 目录下的 Resources 、 Source 、 TXAppBasic 文件夹及 TUIChorus.podspec 文件到您的工程中，并完成如下导入动作：

- 在您的 Podfile 文件内添加导入命令，参考如下：

```
pod 'TUIChorus', :path => "./", :subspecs => ["Professional"]
pod 'TXAppBasic', :path => "TXAppBasic/"
```

- 打开终端，进入Podfile文件所在目录下执行安装命令，参考如下：

```
pod install
```

步骤二：配置权限

在您的工程的 info.plist文件中配置 App 的权限，SDK 需要以下权限（iOS 系统需要动态申请麦克风）：

```
<key>NSMicrophoneUsageDescription</key>
<string>Chorus 需要访问您的麦克风权限</string>
```

步骤三：初始化并登录

相关接口说明请参见 TUIChorus。

```
// 1.初始化
let chorusRoom = TRTCChorusRoom.shared()
chorusRoom.setDelegate(delegate: self)

// 2.登录
chorusRoom.login(SDKAppID: Int32(SDKAppID), UserID: UserID, UserSig: ProfileManager.shared.
curUserSig()) { code, message in
if code == 0 {
//登录成功
}
}
```

参数说明：

- **SDKAppID：TRTC 应用ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：开通推拉流服务

体验 TUIChorus 合唱场景您需要开通腾讯云推拉流服务，用于主播端推流，听众端拉流。您可参考 推拉流 URL，生成自己的推拉流地址并保存，可保存为 pushUrl 和 playerUrl 。

步骤五：实现实时合唱场景

1. **主播创建房间 TUIChorus.createRoom**

```
let roomId = "房间Id"
let param = ChorusParam.init()
param.roomName = "房间名称"
param.needRequest = false // 上麦是否需要房主确认
param.coverUrl = "房间封面图URL"
param.seatCount = 2 // 房间座位数，这里一共2个座位，主播及副唱，其他人进房为听众
param.rtmpPushURL = pushUrl // 步骤三开通的推拉流地址
param.rtmpPlayURL = playUrl
```

```
chorusRoom.createRoom(roomID: roomId, roomParam: param) { [weak self] (code, message)
in
guard let `self` = self else { return }
if code == 0 {
//创建成功
}
}
```

2. **听众进入房间 TUIChorus.enterRoom**

```
chorusRoom.enterRoom(roomID: roomInfo.roomID) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
//进房成功
}
}
```

3. **听众主动上麦成为副唱 TUIChorus.enterSeat**

```
// 1.听众调用上麦
int seatIndex = 1
chorusRoom.enterSeat(seatIndex: seatIndex) { [weak self] (code, message) in
guard let `self` = self else { return }
if code == 0 {
//上麦成功
}
}

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
func onSeatListChange(seatInfoList seatInfolist: [ChorusSeatInfo]) {
}
```

> **说明：**
> 其他关于麦位管理的相关操作，您可参考 TUIChorus接口文档 按需实现，或者可以参考我们的 TUIChorus示例工程。

4. **实现音乐播放并合唱**

您可以根据自己的业务获取音乐 ID 和 URL 链接播放歌曲并合唱，相关接口说明请参见 TUIChorus 音乐播放接口。

```
//播放音乐
chorusRoom.startPlayMusic(musicID: musicModel.musicID, url: musicModel.contentUrl)
//停止音乐
chorusRoom.stopPlayMusic()
```

完成以上步骤，就可以实现合唱基本功能。如果您的业务还需要聊天、发送礼物等功能，可以接入以下能力。

步骤六：文字聊天功能（可选）

如果您需要实现主播、副唱以及听众之间聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口说明请参见 TRTCChorusRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
chorusRoom.sendRoomTextMsg(message: "Hello World") { [weak self] (code, message) in
guard let `self` = self else { return }
if (code == 0) {
//发送成功
}
}

// 接收端：监听文本消息
chorusRoom.setDelegate(delegate: self)
func onRecvRoomTextMsg(message: String, userInfo: ChorusUserInfo) {
debugPrint("收到来自：" + userInfo.userName + "的消息：" + message)
}
```

步骤七：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送礼物或接收礼物并展示。

```
// 发送端：通过自定义 "IMCMD_GIFT" 来区分礼物消息
chorusRoom.sendRoomCustomMsg(cmd: kSendGiftCmd, message: message) { code, msg in
if (code == 0) {
//发送成功
}
```

```
}

// 接收端：监听礼物消息
chorusRoom.setDelegate(delegate: self)
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: ChorusUserInfo) {
if cmd == kSendGiftCmd {
debugPrint("收到来自：" + userInfo.userName + "的礼物：" + message)
}
}
```

常见问题

TUIChorus 组件支持变声、变调、混响等音效功能吗?

支持，具体请参见 TUIChorus 示例工程。

> **说明：**
> 更多帮助信息，详情请参见 TUI 场景化解决方案常见问题。欢迎加入 QQ 群：**592465424**，进行技术交流和反馈。

TUIChorus API 查询
TRTCChorus(Android)

最近更新时间：2022-06-02 15:29:33

TRTCChorusRoom 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 Chorus 房间开播，听众进入 Chorus 房间收听/合唱。
- 房主可以管理点歌、邀请观众上麦，踢人下麦，开始合唱。
- 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，上麦变为副唱，上麦后可以和主播进行合唱，也可以等合唱结束后下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCChorusRoom 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 Chorus（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。

TRTCChorusRoom 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。

API	描述
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

版权所有：腾讯云计算（北京）有限责任公司

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。

TRTCChorusRoomDelegate 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	Chorus 房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onUserMicrophoneMute	用户麦克风是否静音。

| onSeatClose | 房主封麦。 |

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onReceiveAnchorSendChorusMsg	房主发送的合唱消息回调。

SDK 基础函数

sharedInstance

获取 TRTCChorusRoom 单例对象。

```
public static synchronized TRTCChorusRoom 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 TRTCChorusRoom 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCChorusRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

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

setDelegate

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

```
public abstract void setDelegate(TRTCChorusRoomDelegate delegate);
```

> 说明：
> setDelegate 是 TRTCChorusRoom 的代理回调。

版权所有：腾讯云计算（北京）有限责任公司

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	TRTCChorusRoom 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,
TRTCChorusRoomCallback.ActionCallback callback);
```

参数如下表所示：

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

logout

登出。

```
public abstract void logout(TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomDef.RoomParam roomParam, TRTCChorusRoomCallback.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 Chorus 房间列表，腾讯云暂不提供 Chorus 房间列表的管理服务，请自行管理您的 Chorus 房间列表。

参数	类型	含义
roomParam	TRTCChorusRoomParam	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 Chorus 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(TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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。

版权所有：腾讯云计算（北京）有限责任公司

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 Chorus 房间列表，可能包含多个 Chorus 房间的 roomId 和房间信息。
2. 听众选择一个 Chorus 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(TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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 url);
```

参数如下表所示：

参数	类型	含义
musicID	int	音乐的 ID。
url	String	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, TRTCChorusRoomCallback.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(TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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, TRTCChorusRoomCallback.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。

TRTCChorusRoomDelegate 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。

参数	类型	含义
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(TRTCChorusRoomDef.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)
;
```

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 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, TRTCChorusRoomDef.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, TRTCChorusRoomDef.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。

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(TRTCChorusRoomDef.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(TRTCChorusRoomDef.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, TRTCChorusRoomDef.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, TRTCChorusRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	UserInfo	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。

onReceiveAnchorSendChorusMsg

接收到房主发送的合唱消息回调。

```
void onReceiveAnchorSendChorusMsg(String musicID, long startDelay);
```

参数如下表所示：

参数	类型	含义
musicID	String	合唱的歌曲 musicID。
startDelay	long	合唱的歌曲延迟多少秒播放。

TRTCChorus(iOS)

最近更新时间：2022-06-02 15:29:41

TRTCChorusRoom 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 Chorus 房间开播，听众进入 Chorus 房间收听/合唱。
- 房主可以管理点歌、邀请观众上麦，踢人下麦，开始合唱。
- 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，上麦变为副唱，上麦后可以和主播进行合唱，也可以等合唱结束后下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> 说明：
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCChorusRoom 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 Chorus（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。

TRTCChorusRoom 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。

API	描述
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 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）。
closeSeat	封禁/解禁某个麦位（房主调用）。

版权所有：腾讯云计算（北京）有限责任公司

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

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。

版权所有：腾讯云计算（北京）有限责任公司

TRTCChorusRoomDelegate 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）。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。
onReceiveAnchorSendChorusMsg	房主发送的合唱消息回调。

SDK 基础函数

sharedInstance

获取 TRTCChorus 单例对象。

```
/**
* 获取 TRTCChorus 单例对象
*
* - returns: TRTCChorus 实例
* - note: 可以调用 {@link TRTCChorus#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCChorus 单例对象。

> 说明：
> 销毁实例后，外部缓存的 TRTCChorus 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCChorus 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCChorus 实例不能再使用，需要重新调用 {@link TRTCChorus#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

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

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCChorusDelegate 获得 TRTCChorus 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCChorus 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 TRTCChorus#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCChorusDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

> 说明：
> setDelegate 是 TRTCChorus 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，TRTCChorus 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	TRTCChorus 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
userId:(NSString *)userId
userSig:(NSString *)userSig
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback:));
```

参数如下表所示：

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

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(RoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 createRoom 创建新的 Chorus 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 enterSeat 进入座位。
3. 房主收到组件的 onSeatListChange 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 onAnchorEnterSeat 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Chorus 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 Chorus 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:(ChorusInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(ChorusUserListCallback _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 url:(NSString *)url NS_SWIFT_NAME(startPlayMusic(musicID:url:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	音乐的 ID。
url	String	音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

> 说明：
> 停止播放后，会收到 onMusicCompletePlaying 的事件通知。

```
- (void)stopPlayMusic NS_SWIFT_NAME(stopPlayMusic());
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

> 说明：
> - onMusicProgressUpdate 的事件通知会暂停。
> - 不会收到 onMusicCompletePlaying 的事件通知。

```
- (void)pausePlayMusic NS_SWIFT_NAME(pausePlayMusic());
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> 说明：
> 不会收到 onMusicPrepareToPlay 的事件通知。

```
- (void)resumePlayMusic NS_SWIFT_NAME(resumePlayMusic());
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> 说明：
> 上麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> 说明：
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

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。

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。

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)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:));
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 – 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
mute	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

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:conten
t:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(acceptInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(rejectInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCChorusDelegate 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
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:(ChorusInfo *)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。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSInteger)totalVolume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<ChorusSeatInfo *> *)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:(ChorusUserInfo *)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:(ChorusUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(ChorusUserInfo *)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:(ChorusUserInfo *)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:(ChorusUserInfo *)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:(ChorusUserInfo *)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。
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:(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。

onReceiveAnchorSendChorusMsg

接收到房主发送的合唱消息回调。

```
- (void)onReceiveAnchorSendChorusMsg:(NSString *)musicId startDelay:(NSInteger)startDelay
NS_SWIFT_NAME(onReceiveAnchorSendChorusMsg(musicId:startDelay:));
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
musicId	NSString	合唱的歌曲 musicID。
startDelay	NSInteger	合唱的歌曲延迟多少秒播放。