实时音视频

旧版文档

产品文档

腾讯云

文档目录

旧版文档
集成 TUIRoom (Web)

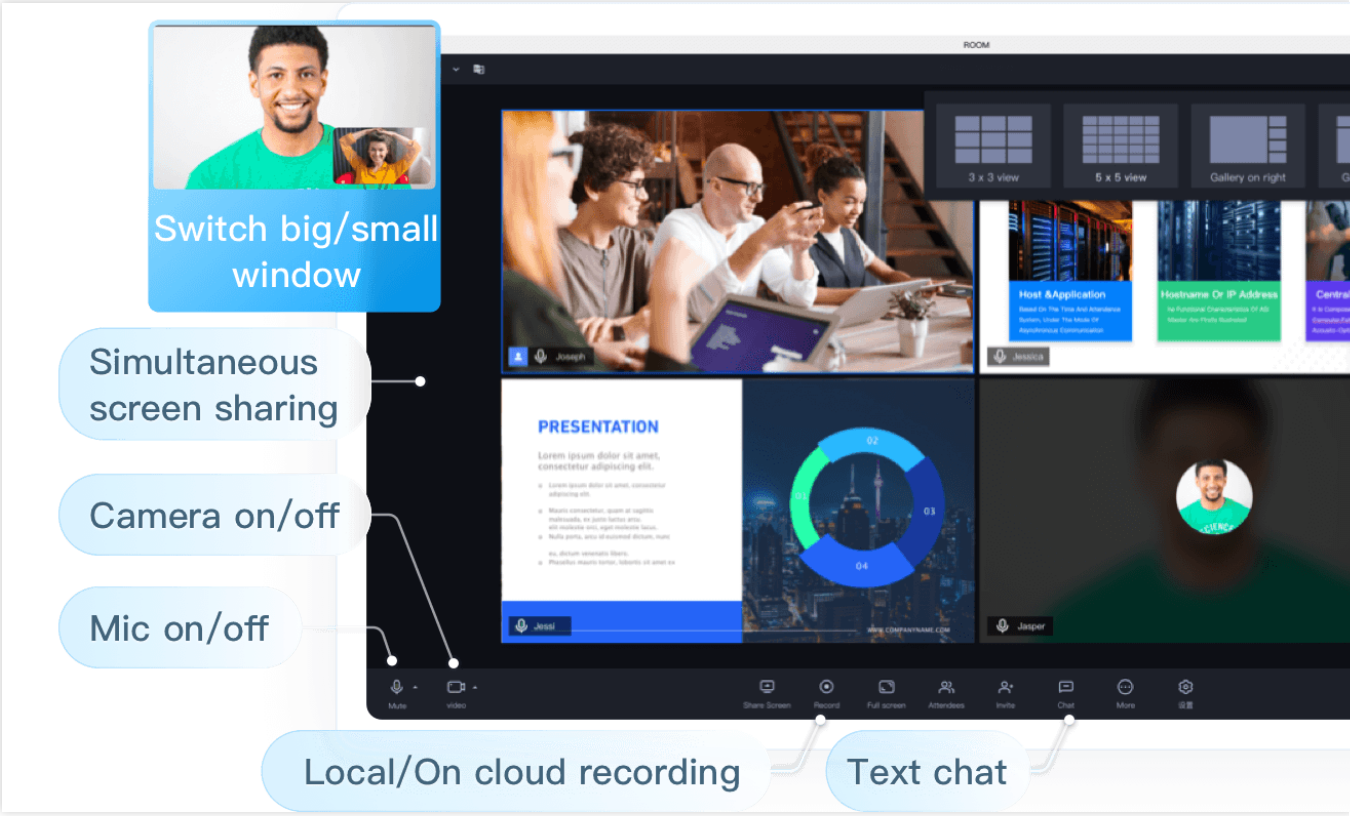
最近更新时间：2024-06-07 11:20:49

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。

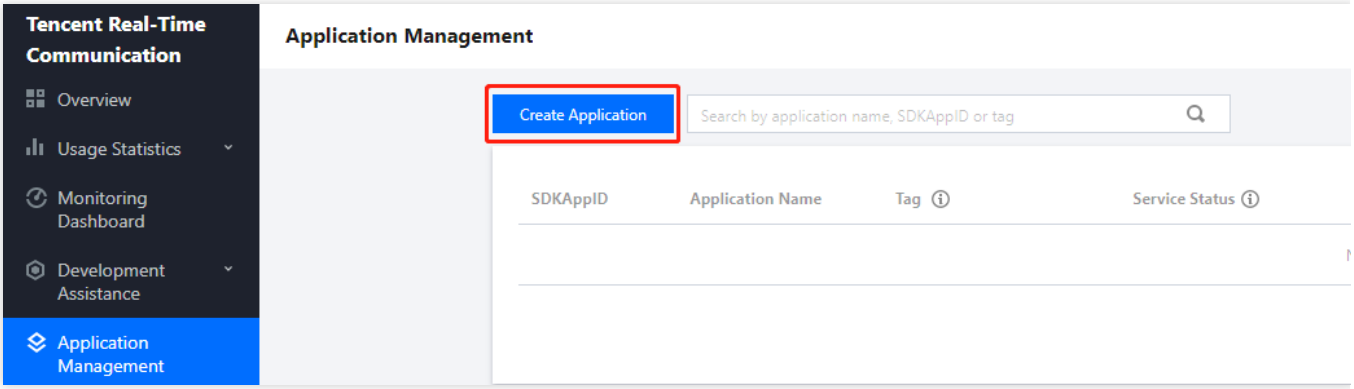
步骤一： 开通腾讯云实时音视频及即时通信服务

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**

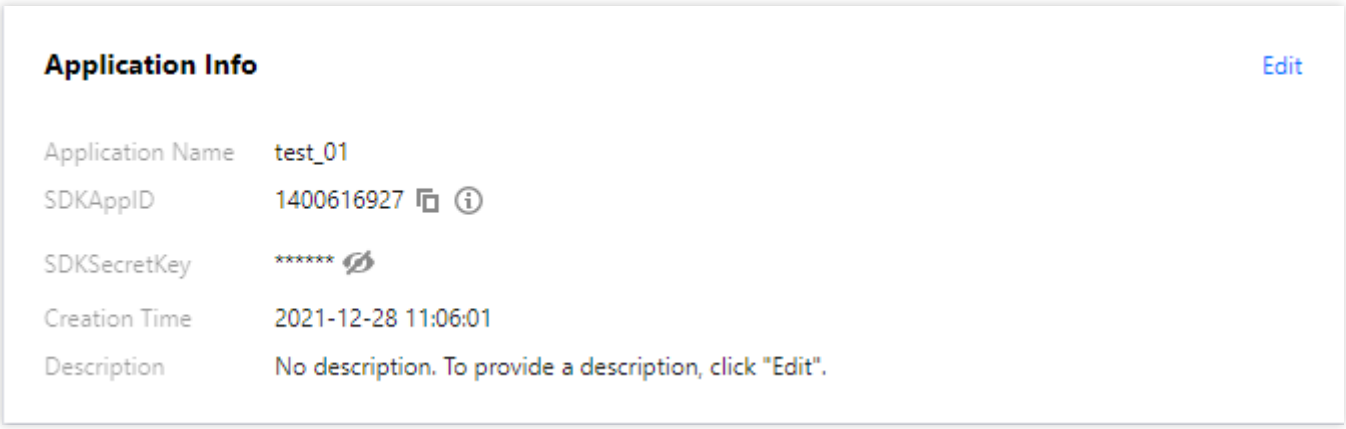
如果您还没有腾讯云账号，请 注册腾讯云账号 。

在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。

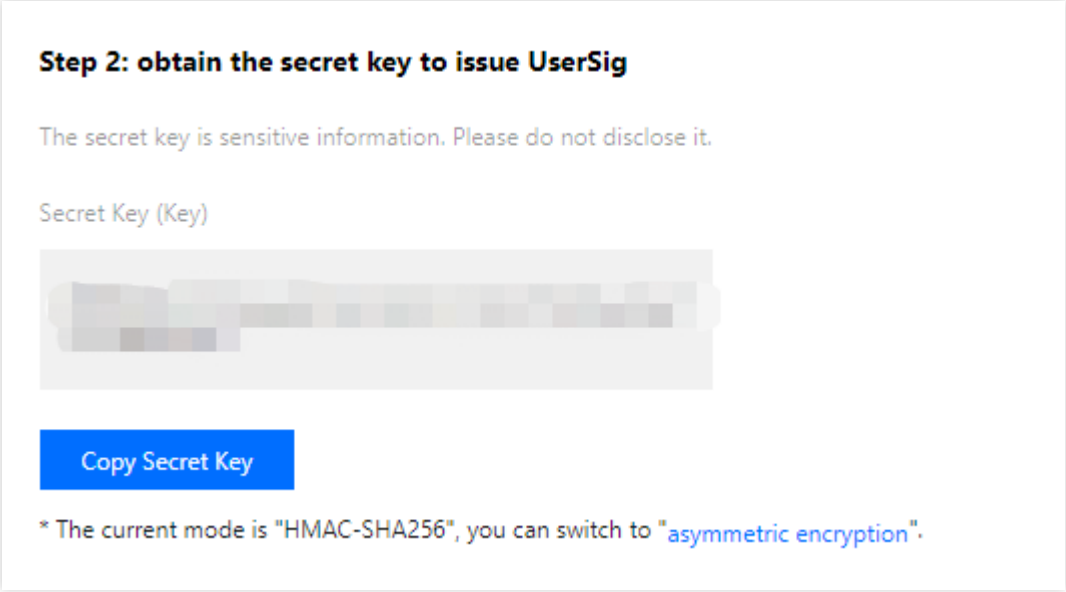


2. **获取 TRTC 应用及密钥信息。**

2.1 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。



2.2 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。

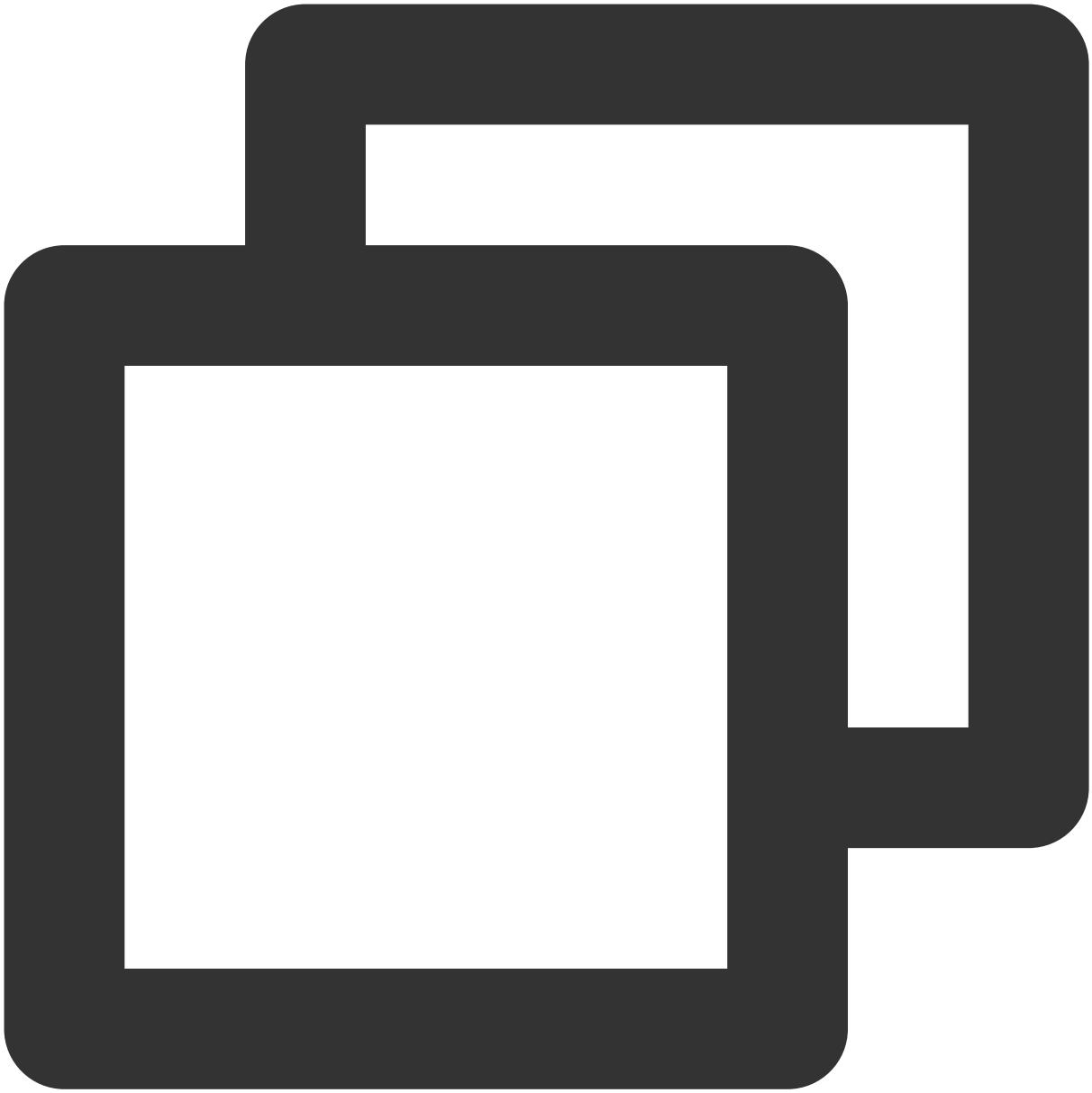
步骤二：下载并拷贝 TUIRoom 组件

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，支持使用 Vite 及 Webpack 打包方式。如果无 Vue3 + TS 项目，可选择以下任意一种方式生成模版工程。

生成 Vue3 + Vite + TS 模版工程

生成 Vue3 + Webpack + TS 模版工程

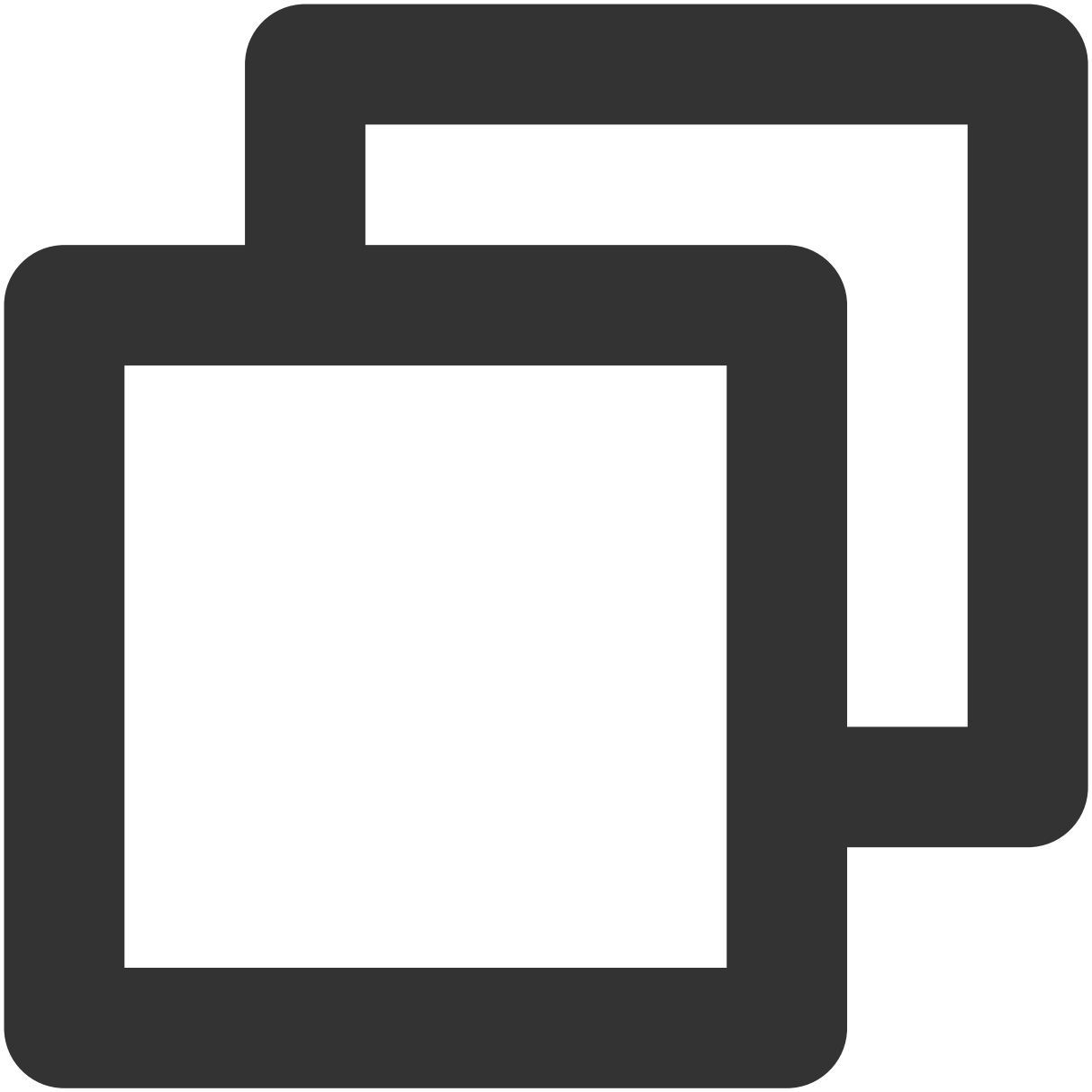


```
npm create vite@latest TUIRoom-demo -- --template vue
```

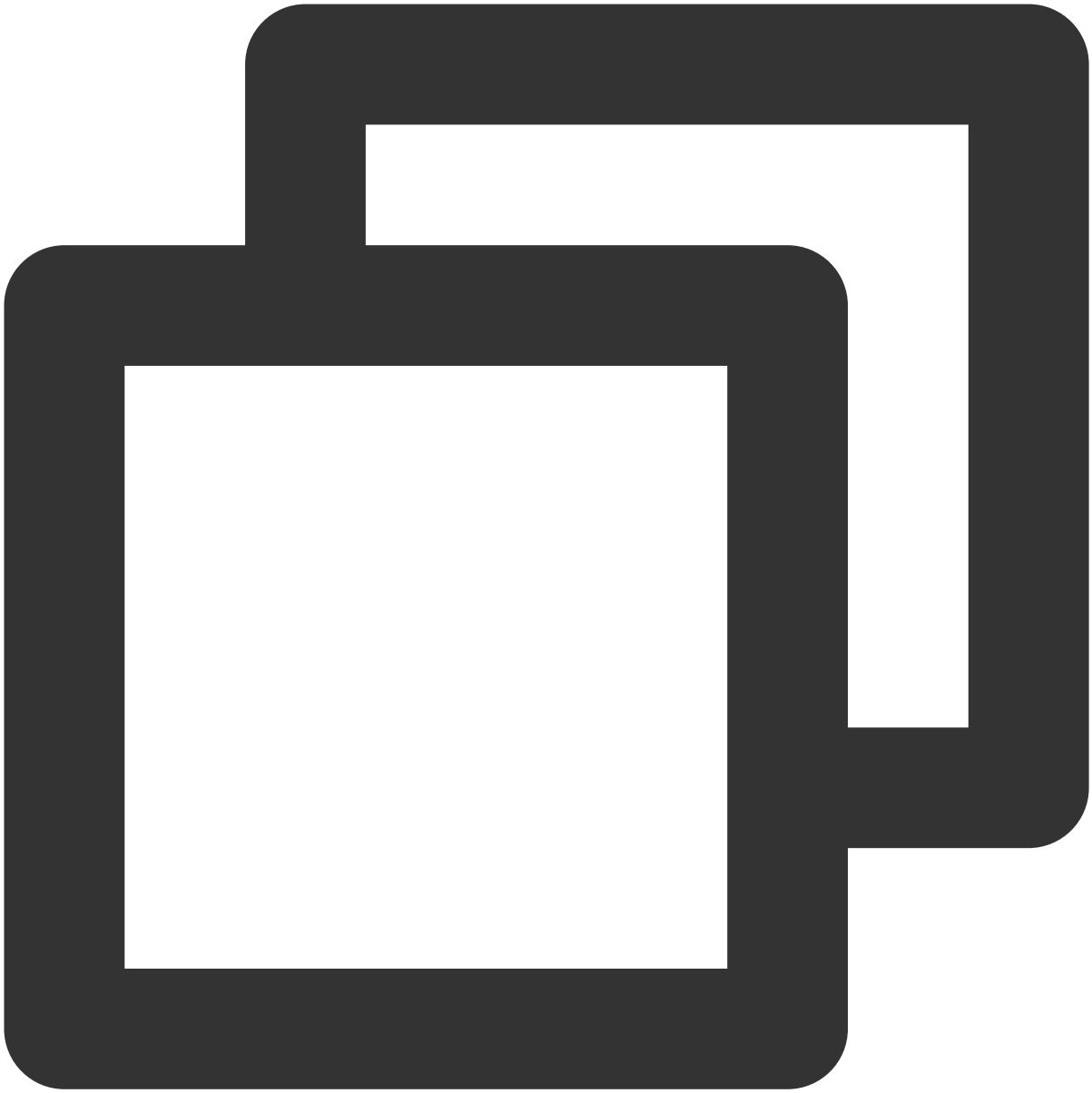
注意：

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 Vue3 + Vite + TS 模板工程后，执行以下脚本：



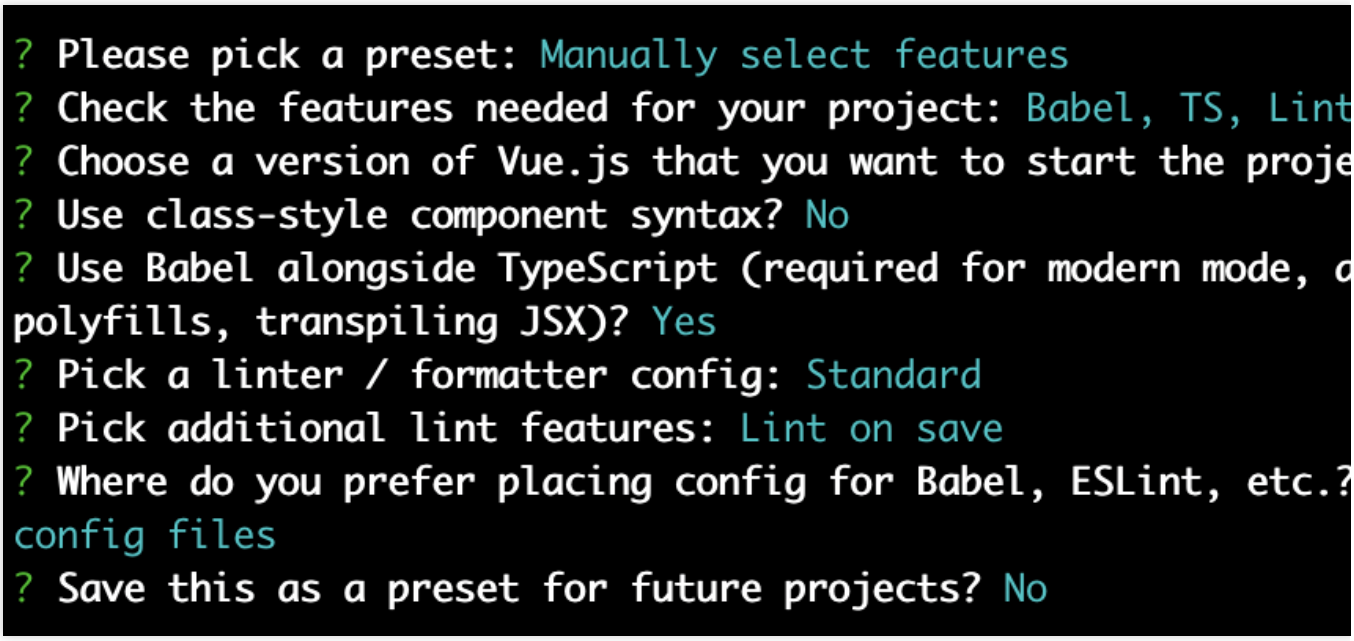
```
cd TUIRoom-demo
npm install
npm run dev
```



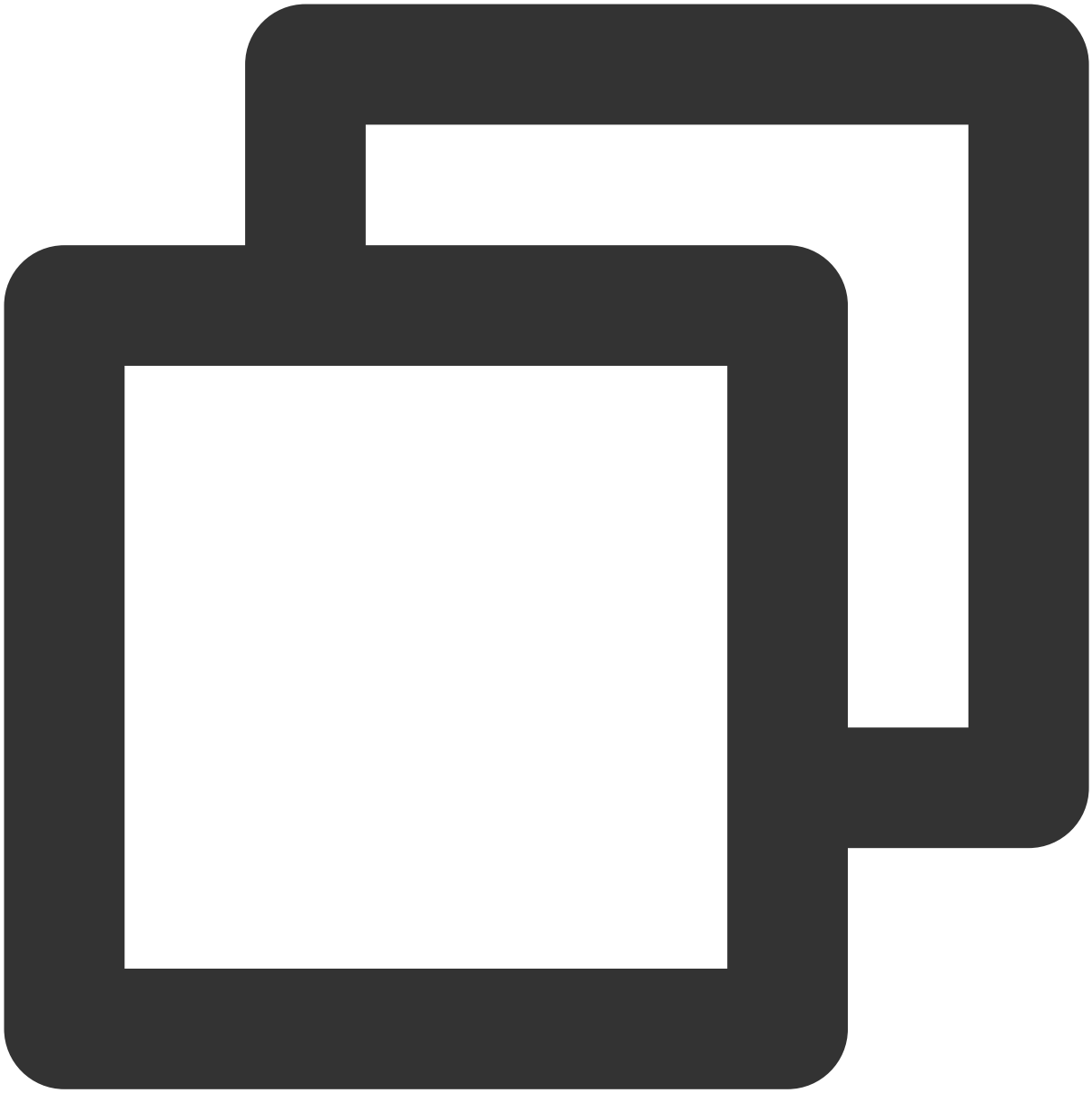
```
// 安装 vue-cli，注意 Vue CLI 4.x 要求 Node.js 为 v10 以上版本
npm install -g @vue/cli
// 创建 Vue3 + Webpack + TS 模版工程
vue create TUIRoom-demo
```

注意：

执行生成模板工程脚本的过程中，生成模版的方式选择 Manually select features，其余配置选项参考图片。



成功生成 Vue3 + Webpack + TS 模板工程后，执行以下脚本：



```
cd TUIRoom-demo
npm run serve
```

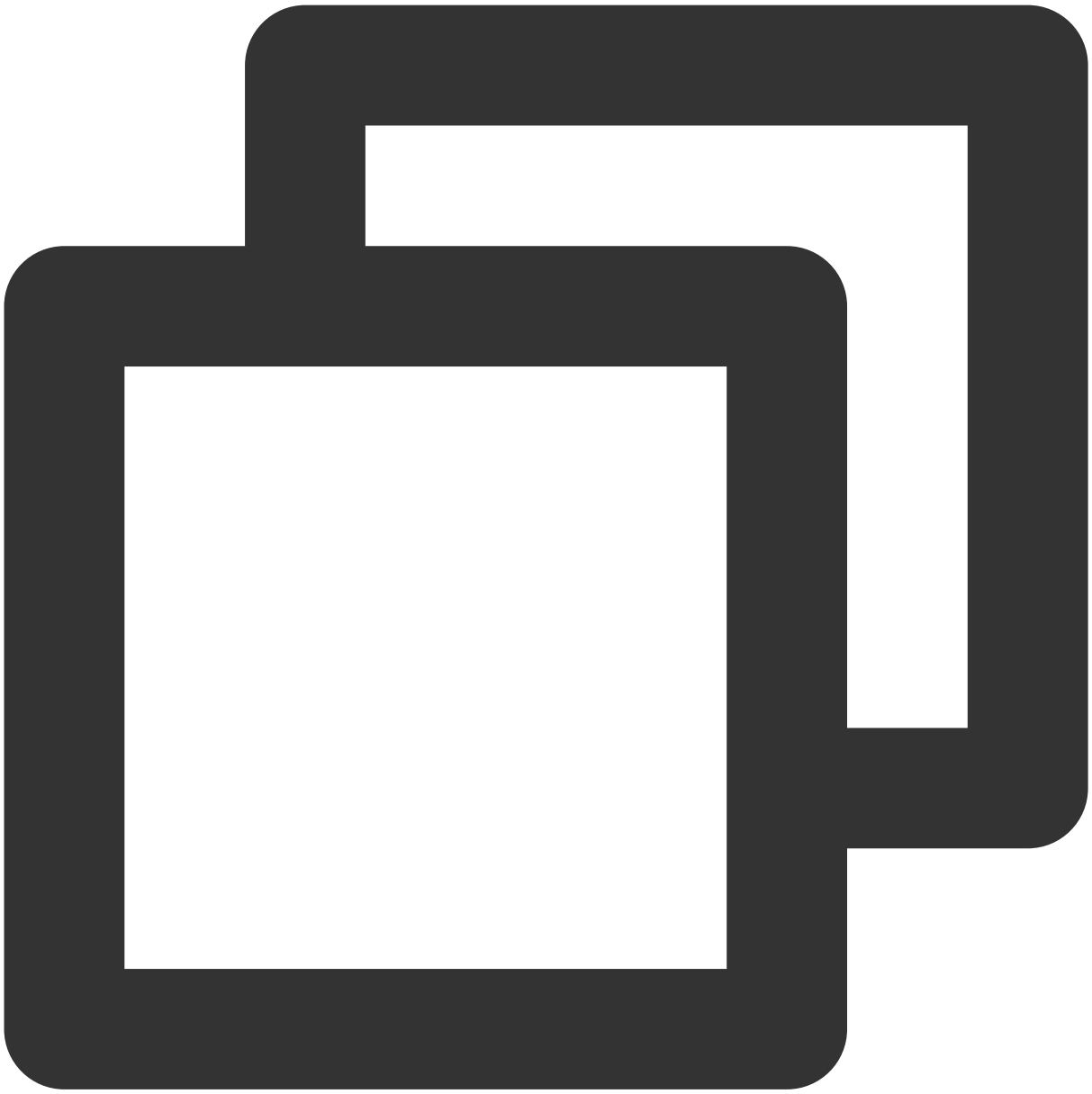
3. 复制 `TUIRoom/Web/src/TUIRoom` 文件夹到已有项目 `src/` 目录下。

步骤三：引用 TUIRoom 组件

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.tencentcloud.com/trtc/usersigtool 页面快速生成
     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
     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
 // 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
 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
 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

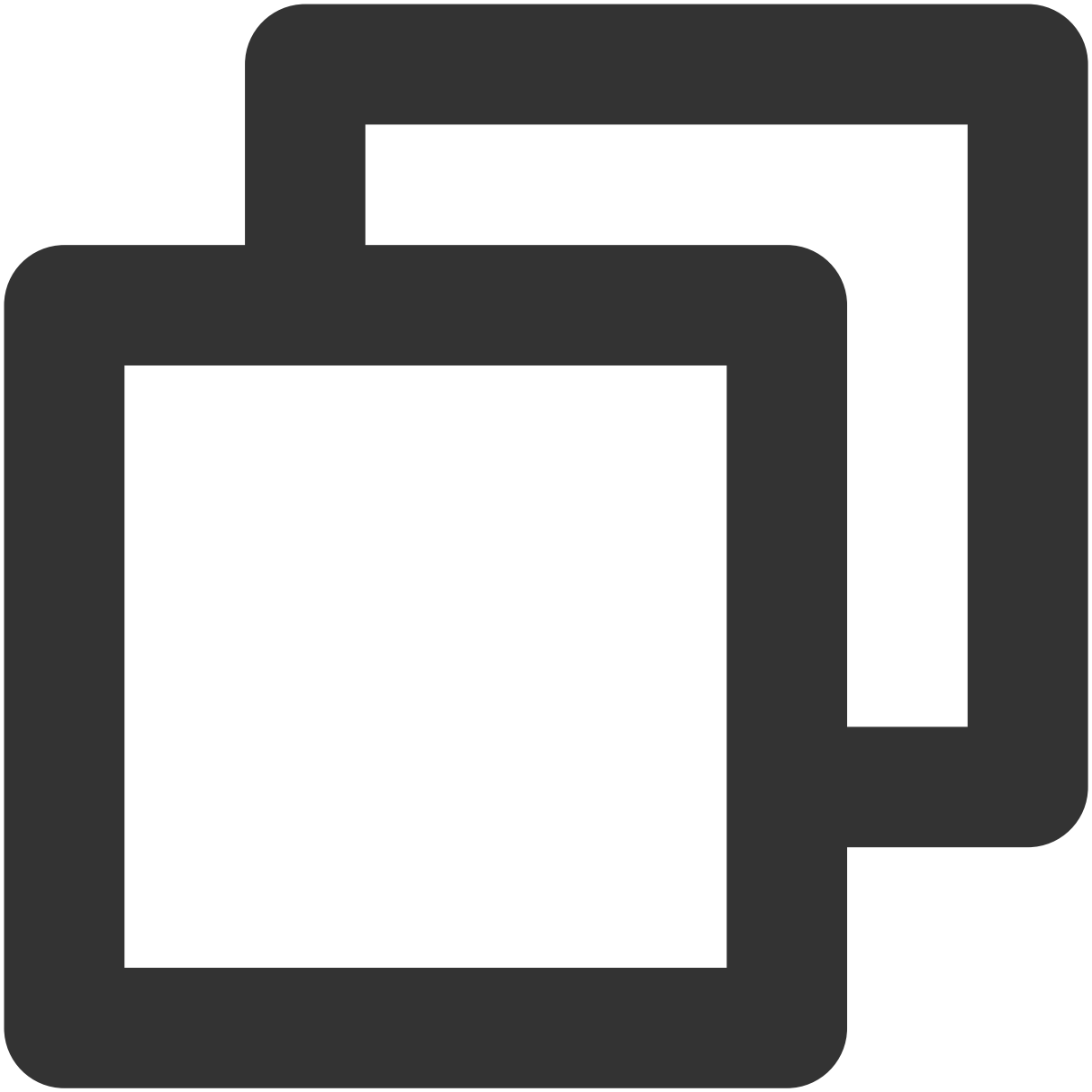
TUIRoom 组件引入之后，为了确保项目可以正常运行，需要进行以下配置：

配置 Vue3 + Vite + TS 项目开发环境

配置 Vue3 + Webpack + TS 项目开发环境

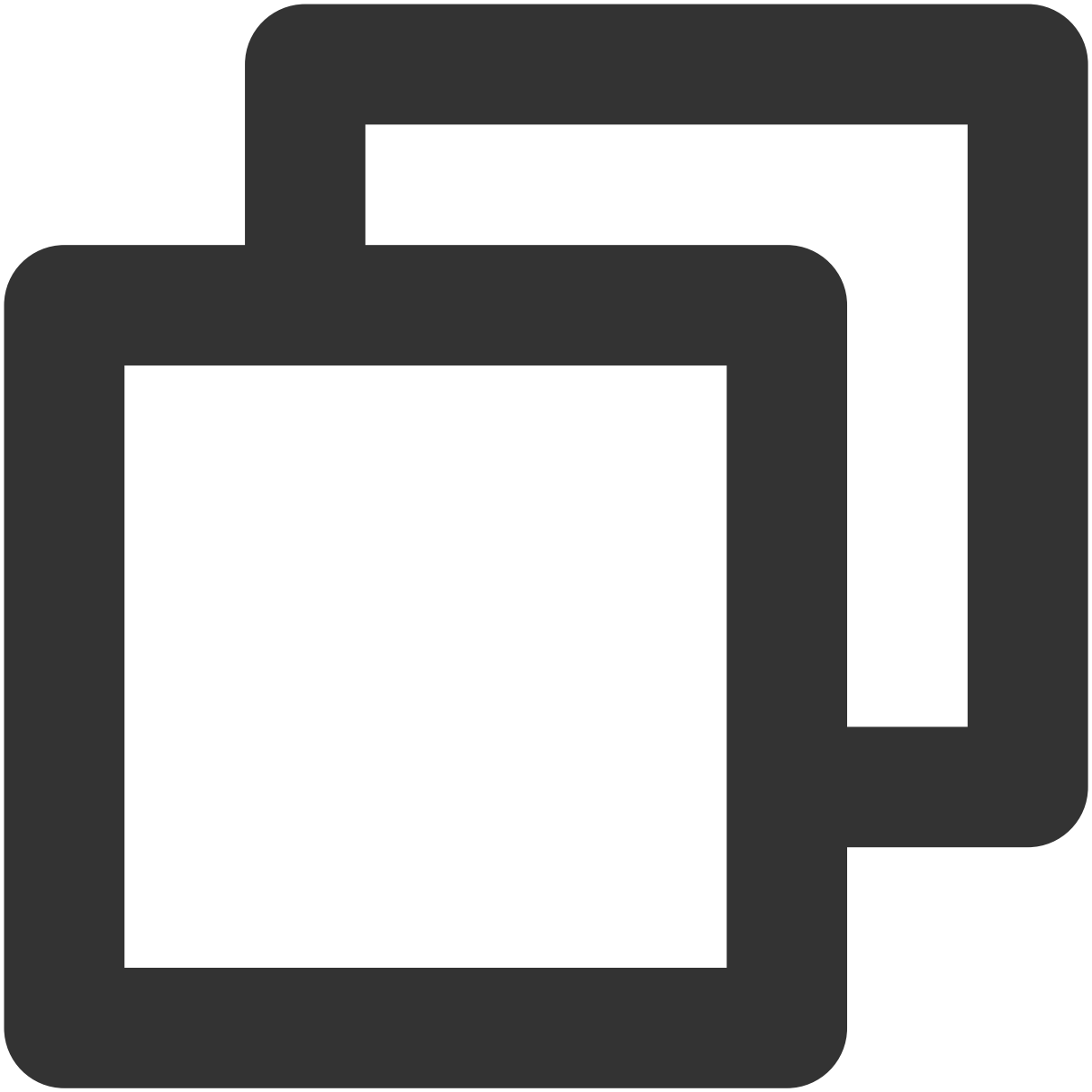
1. 安装依赖

安装开发环境依赖：



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

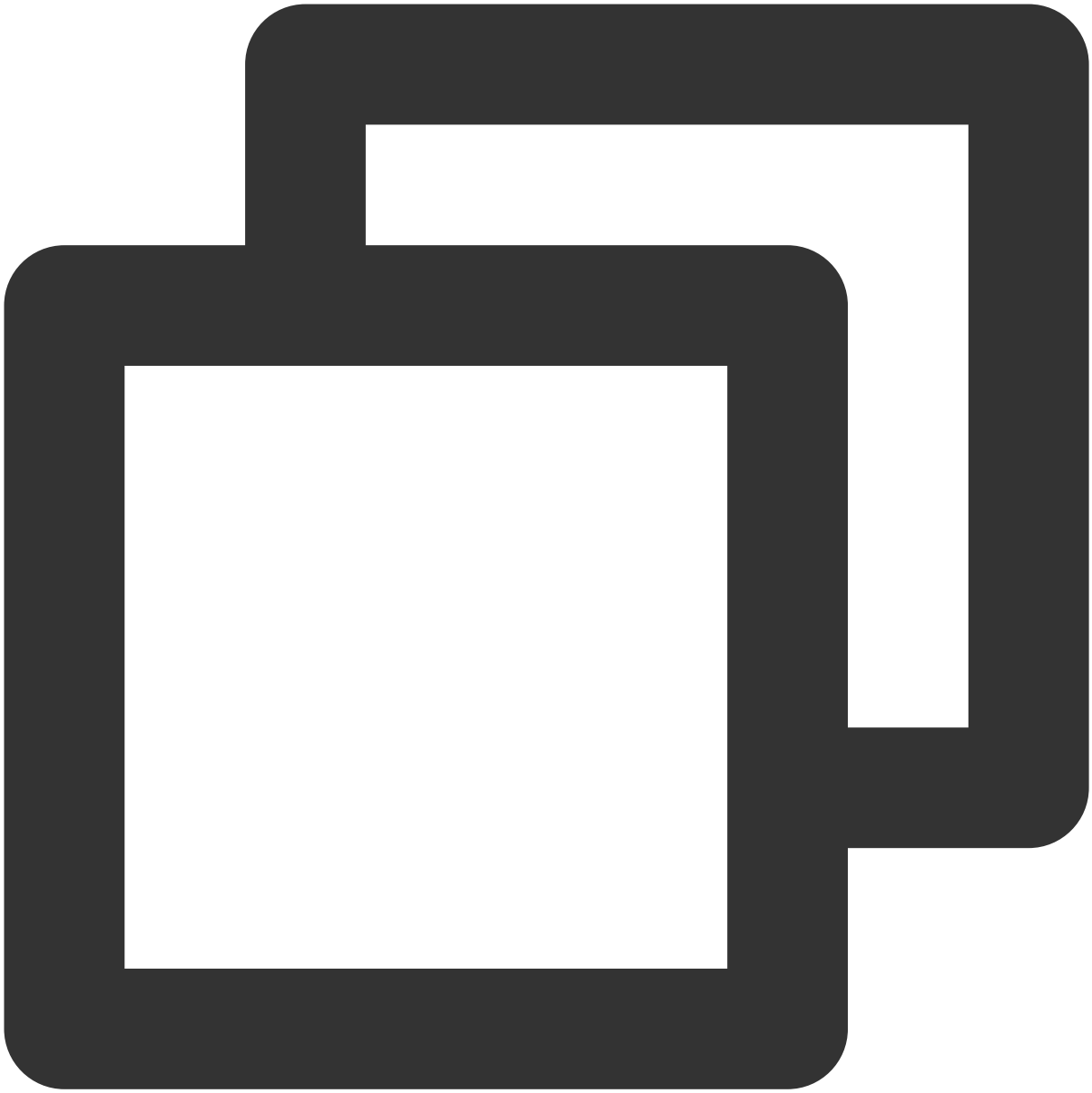
安装生产环境依赖：



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

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。



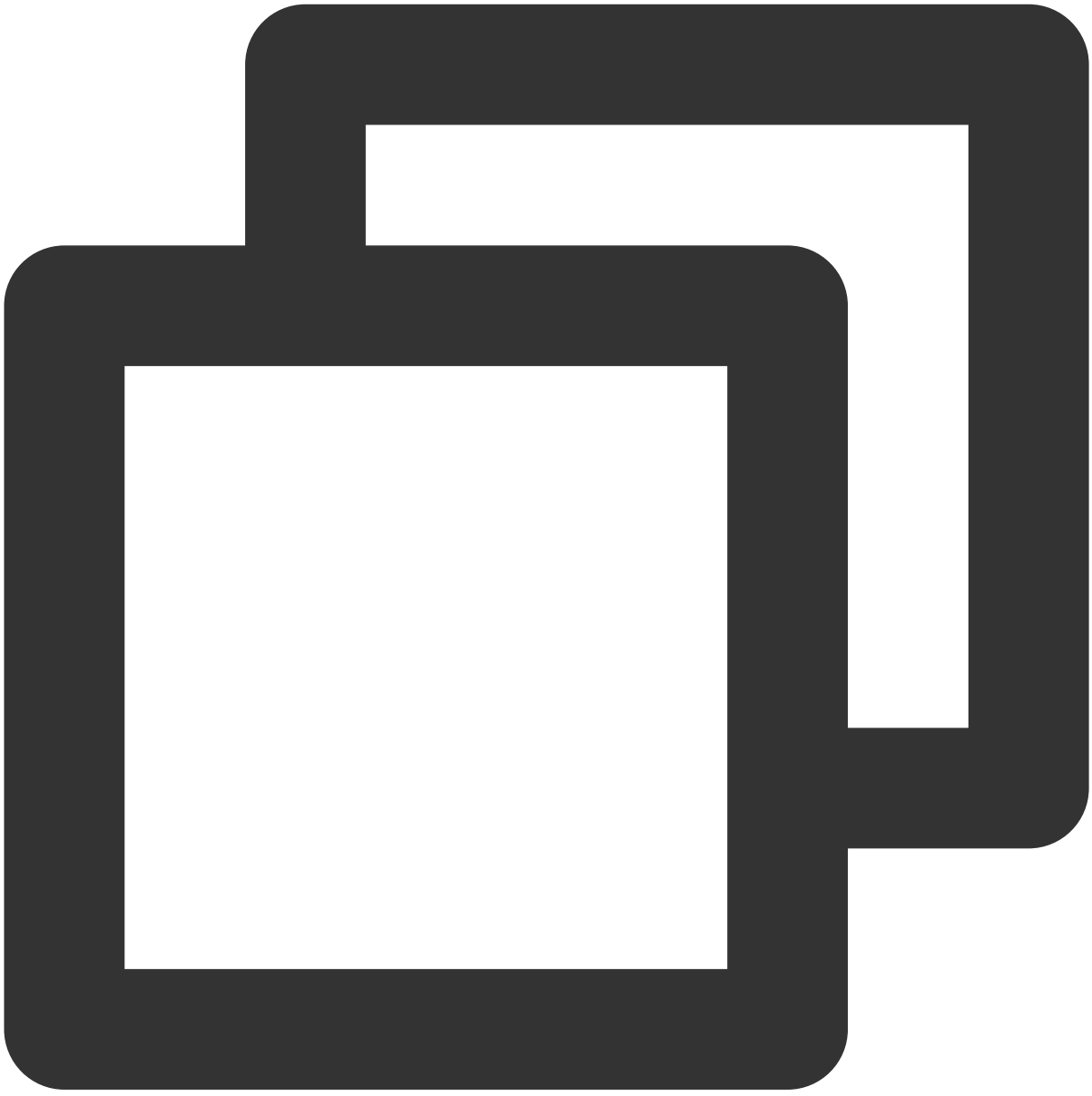
```
// src/main.ts 文件
import { createPinia } from 'pinia';

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

3. **配置 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：

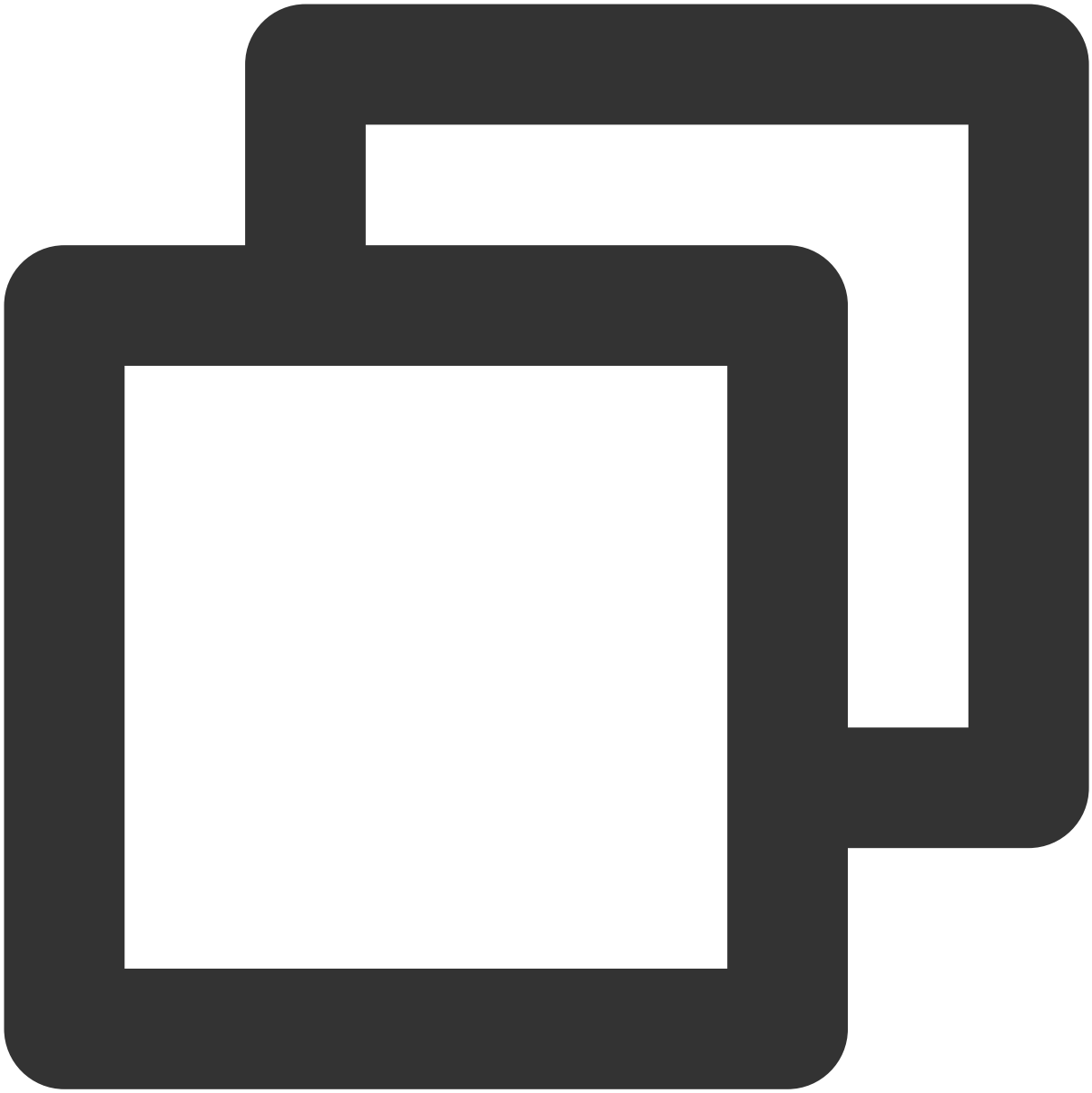
以下配置项为增量配置，不要删除已经存在的 Vite 配置项。



```
// 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 *;
                `,
            },
        },
    },
});
```

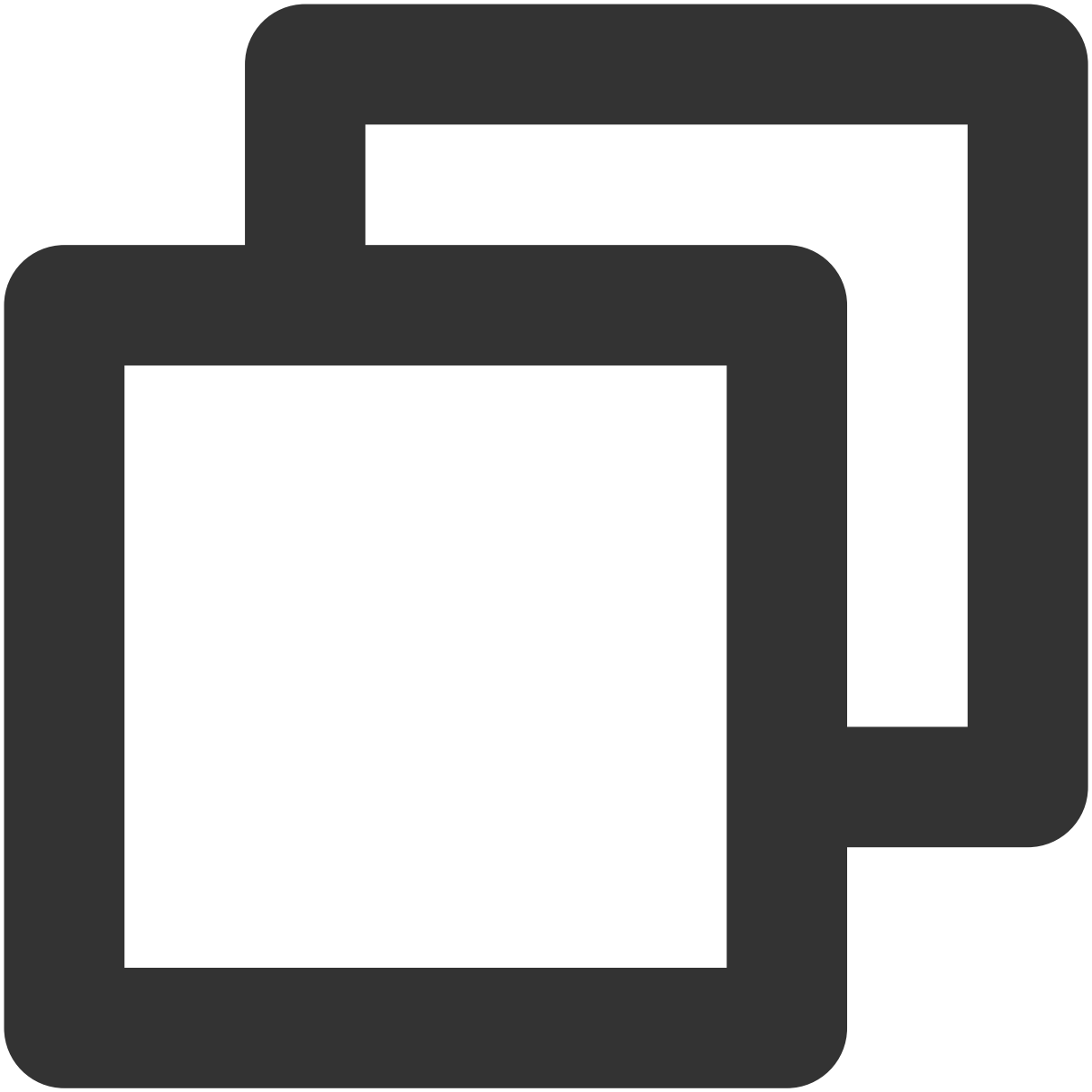
同时为了保证 element-plus 带 UI 组件能够正常显示样式，需要您在入口文件 `src/main.ts` 中加载 element-plus 组件样式。



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

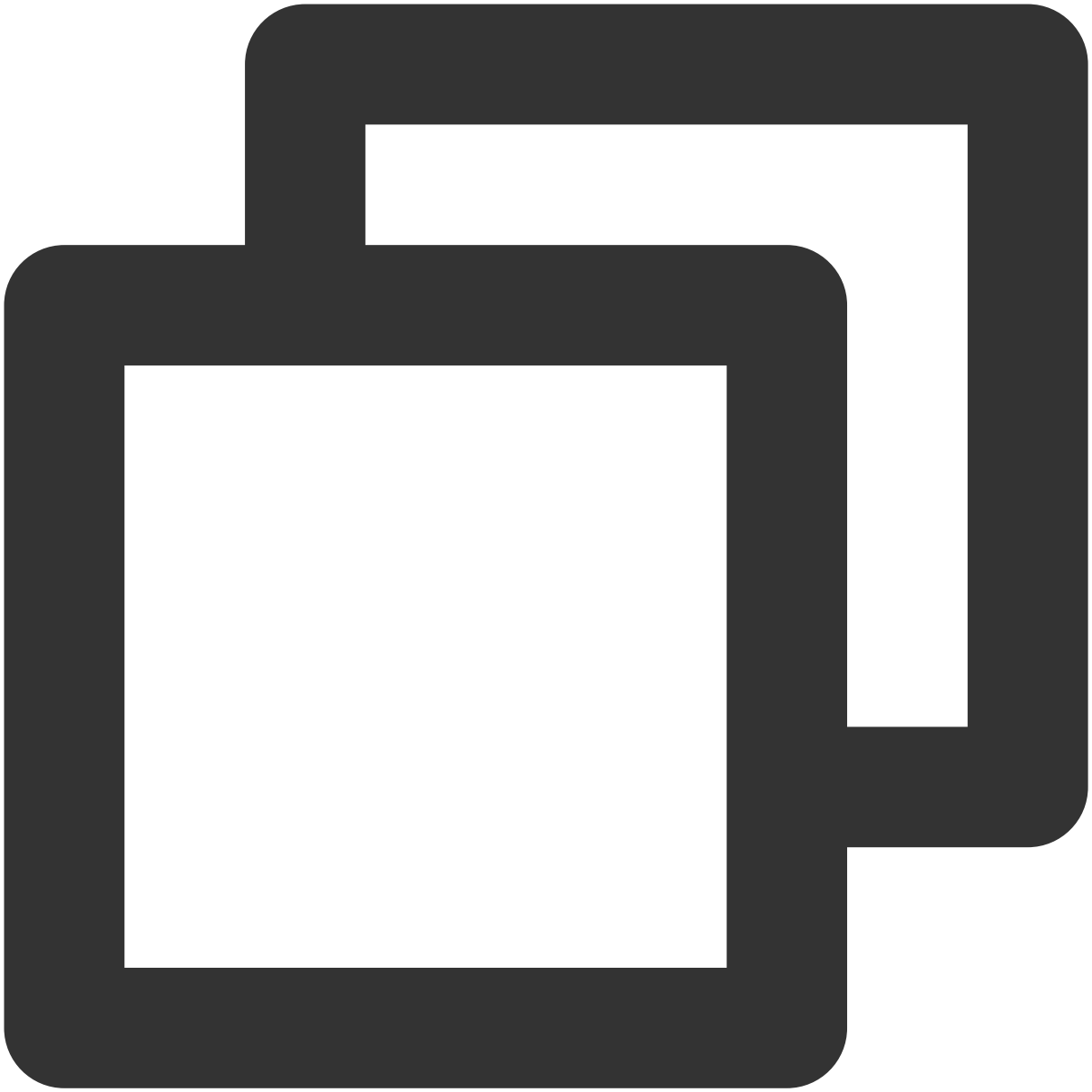
1. **安装依赖**

安装开发环境依赖：



```
npm install sass sass-loader typescript unplugin-auto-import unplugin-vue-component
```

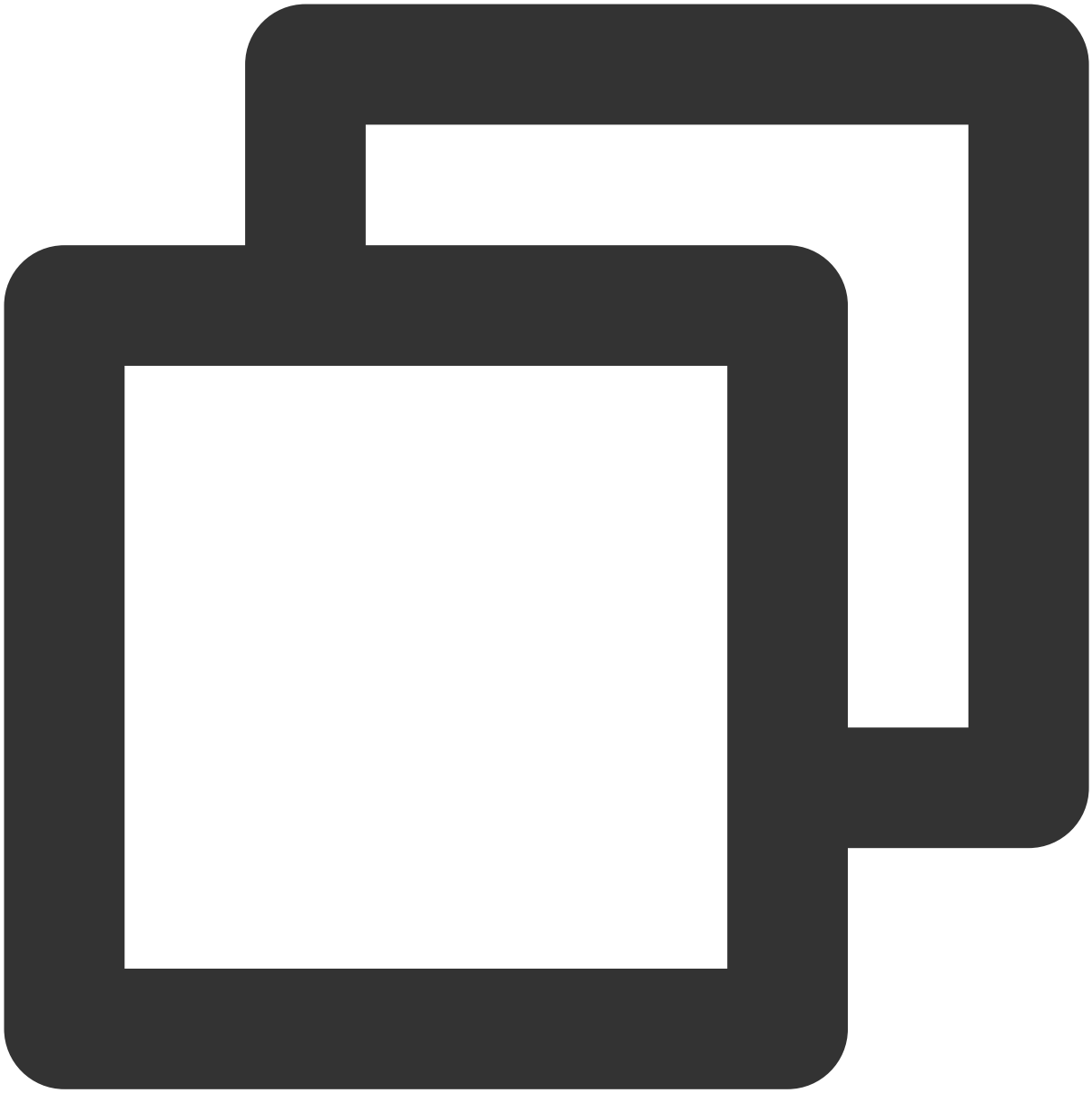
安装生产环境依赖：



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

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。



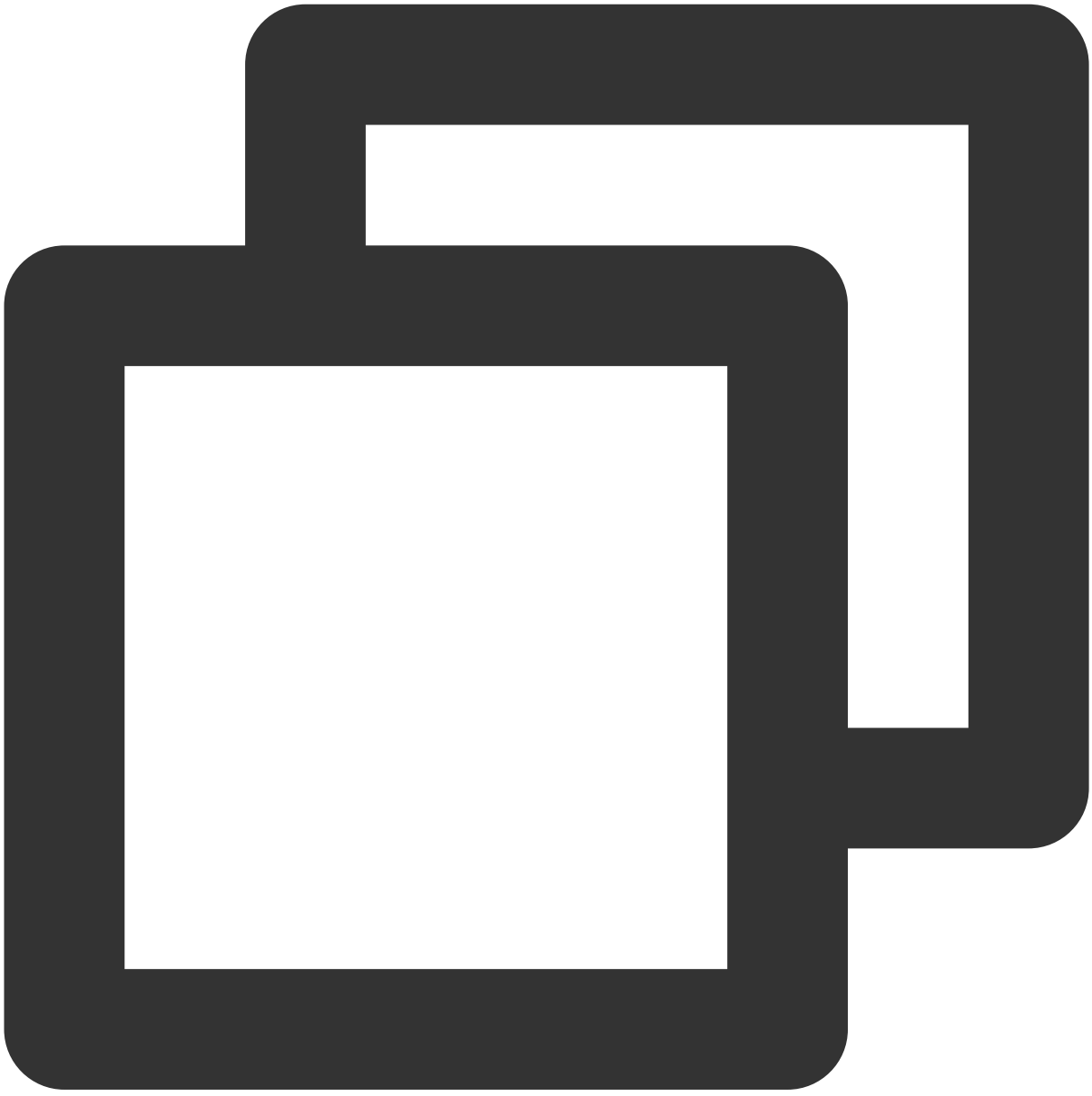
```
// src/main.ts 文件
import { createPinia } from 'pinia';

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

3. **配置 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 `vue.config.js` 中配置 element-plus 组件按需加载。

注意：

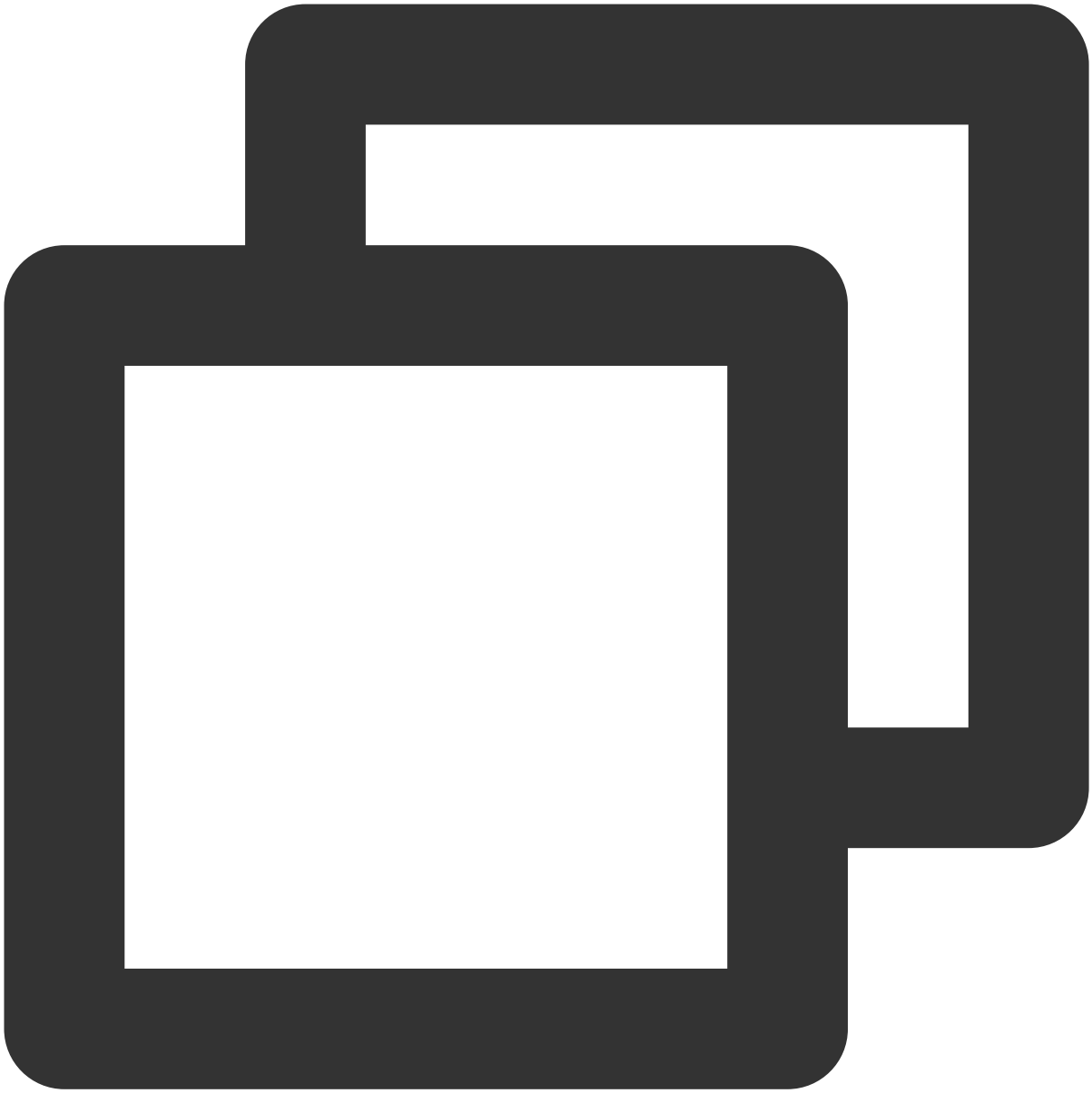
以下配置项为增量配置，不要删除已经存在的 vue.config.js 配置项。



```
// vue.config.js
const { defineConfig } = require('@vue/cli-service')
const AutoImport = require('unplugin-auto-import/webpack')
const Components = require('unplugin-vue-components/webpack')
const { ElementPlusResolver } = require('unplugin-vue-components/resolvers')
const ElementPlus = require('unplugin-element-plus/webpack')
```

```
module.exports = defineConfig({
  transpileDependencies: true,
  css: {
    loaderOptions: {
      scss: {
        additionalData: '@use "./src/TUIRoom/assets/style/element.scss" as *;'
      }
    }
  },
  configureWebpack: {
    plugins: [
      AutoImport({
        resolvers: [ElementPlusResolver({ importStyle: 'sass' })]
      }),
      Components({
        resolvers: [ElementPlusResolver({ importStyle: 'sass' })]
      }),
      // 在按需导入时自定义主题颜色
      ElementPlus({
        useSource: true
      })
    ]
  }
})
```

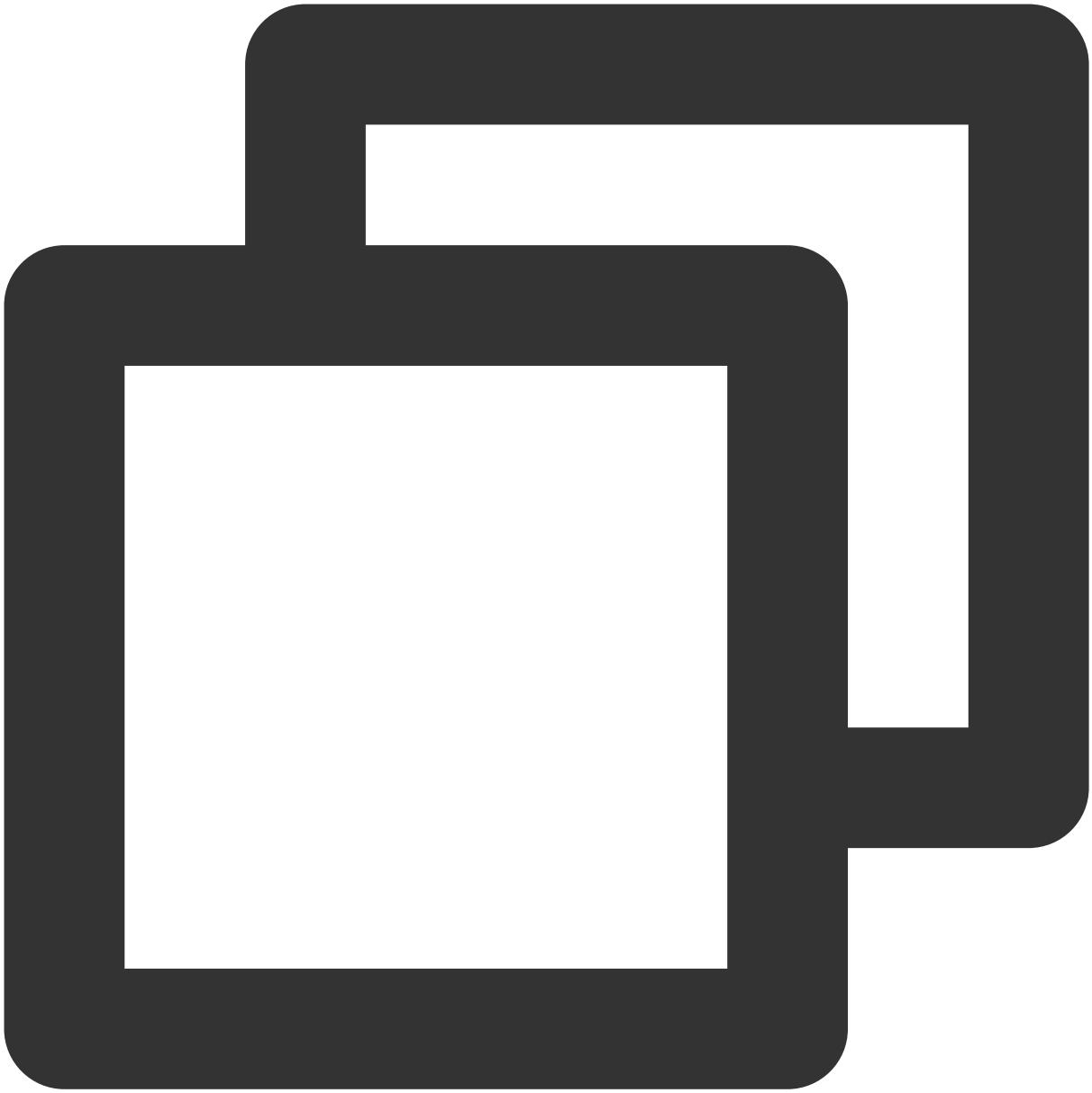
同时为了保证 element-plus 带 UI 组件能够正常显示样式，需要您在入口文件 `src/main.ts` 中加载 element-plus 组件样式。



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

4. 配置 ts 声明文件

在 `src/shims-vue.d.ts` 文件中添加以下配置：



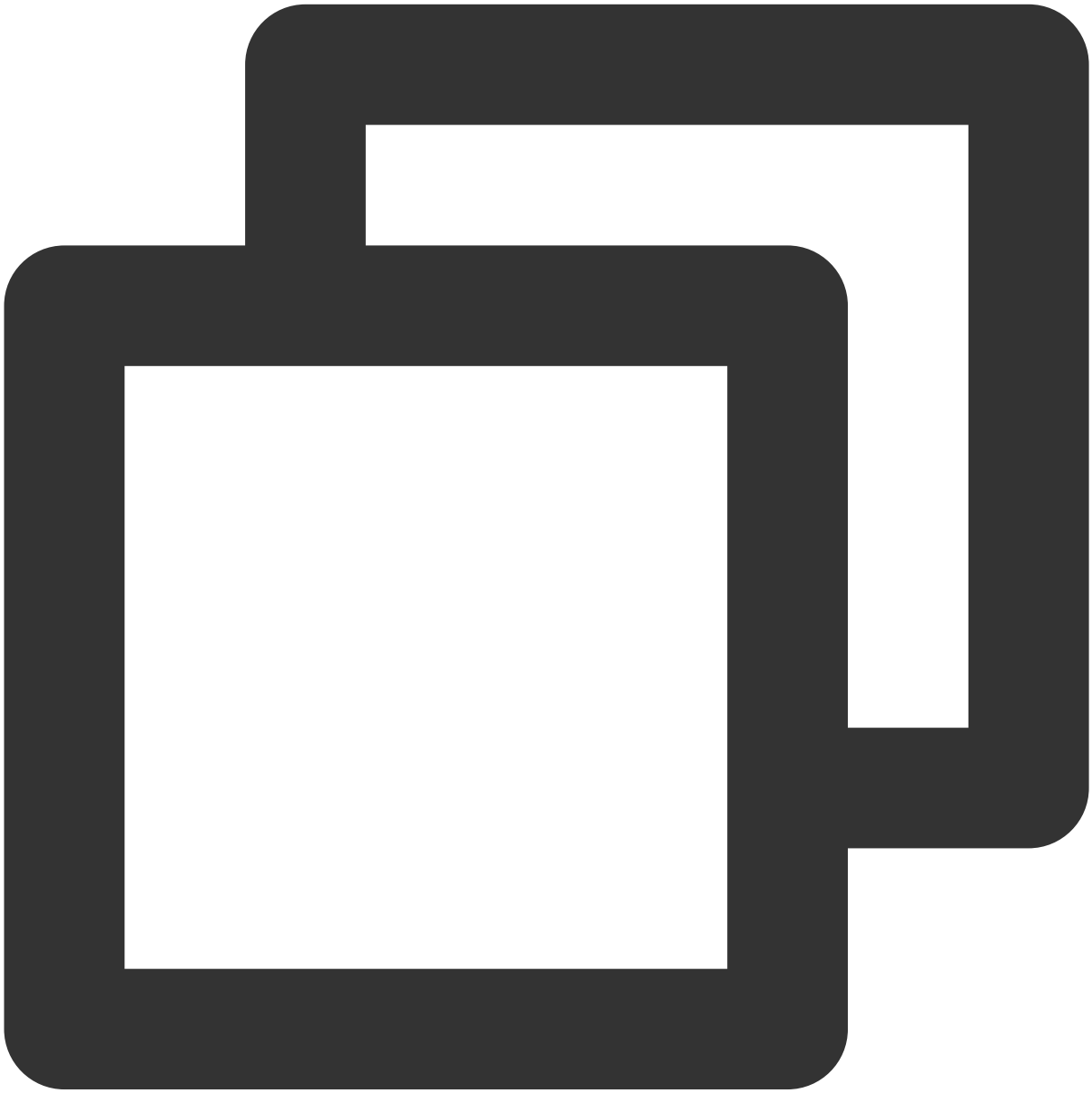
```
declare module 'tsignaling/tsignaling-js' {
  import TSignaling from 'tsignaling/tsignaling-js';
  export default TSignaling;
}

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

declare const Aegis: any;
```

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 + Vite + TS 项目，您需要：

1.1 执行开发环境命令。



```
npm run dev
```

1.2 在浏览器中打开页面 `http://localhost:3000/` 。

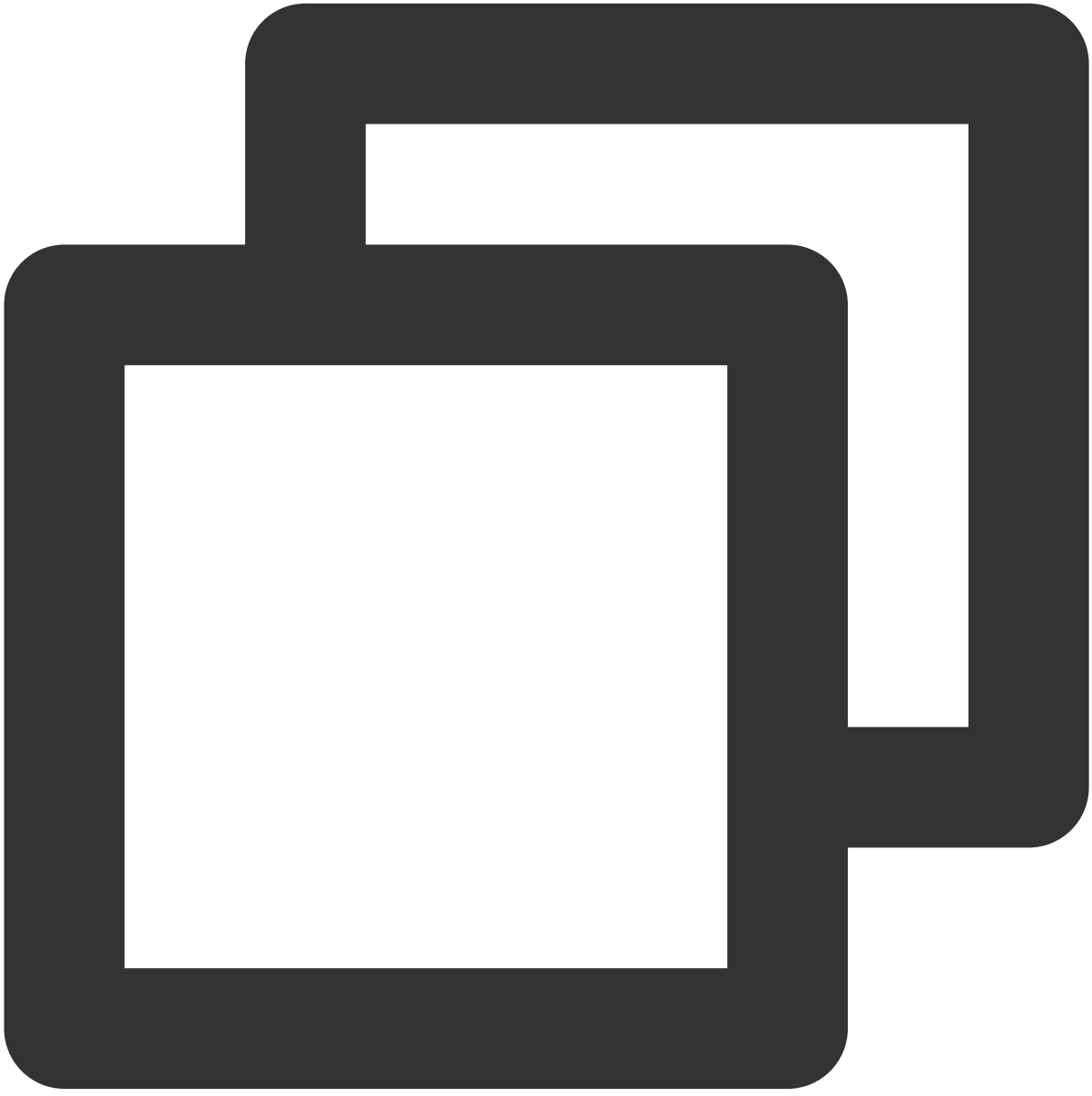
注意：

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。

1.3 体验 TUIRoom 组件功能。

如果您是使用 步骤二 中的脚本生成 Vue + Webpack + TS 项目，您需要：

1.1 执行开发环境命令



```
npm run serve
```

1.2 在浏览器中打开页面 `http://localhost:8080/`

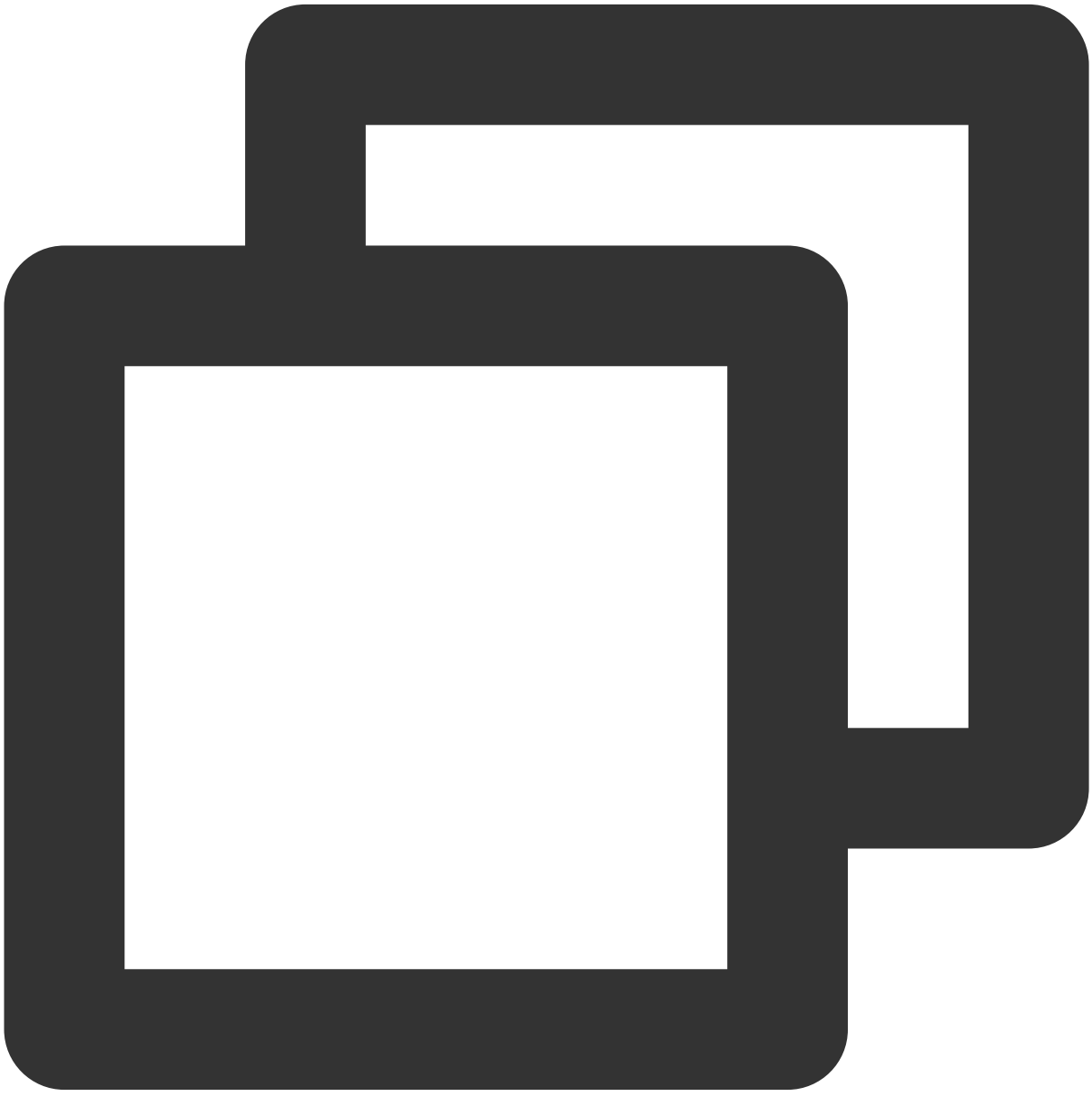
注意：

运行过程中若 src/TUIRoom 目录中有 eslint 报错，可在 .eslintignore 文件中添加 /src/TUIRoom 路径屏蔽 eslint 检查。

1.3 体验 TUIRoom 组件功能。

步骤六：生产环境部署

1. 打包 dist 文件



```
npm run build
```

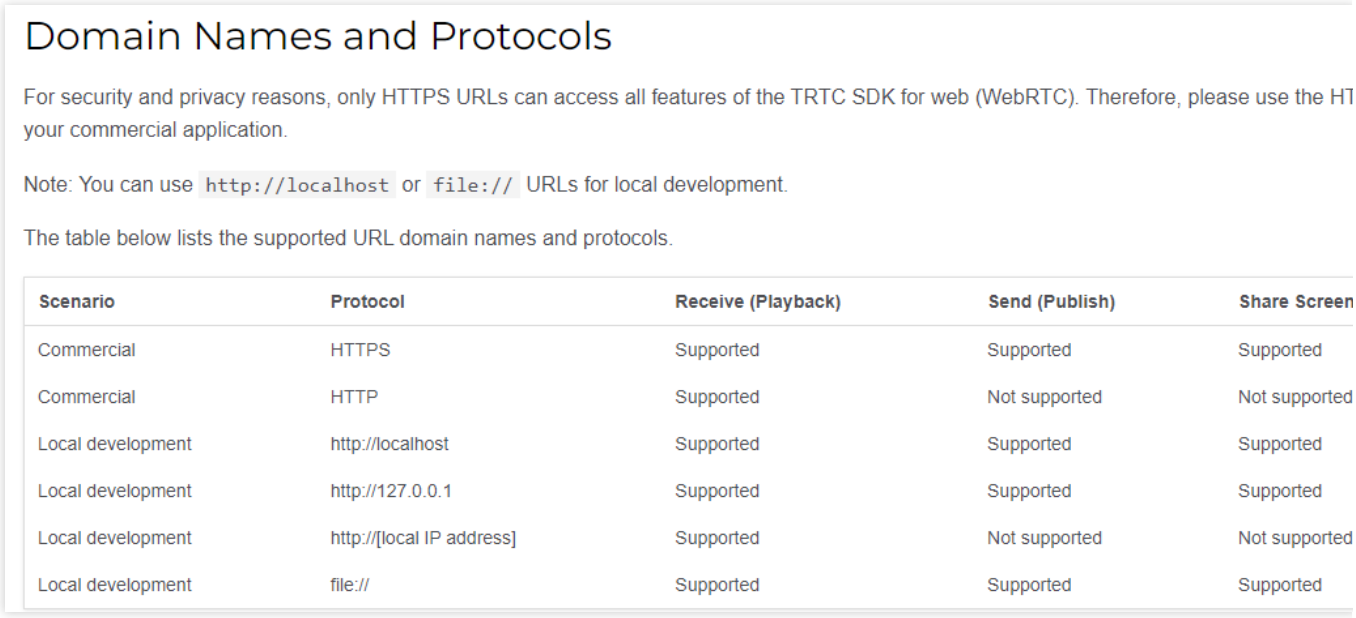
说明：

实际打包命令请查看 package.json 文件。

2. 部署 dist 文件到服务器上

注意：

生产环境要求使用 HTTPS 域名：



Domain Names and Protocols

For security and privacy reasons, only HTTPS URLs can access all features of the TRTC SDK for web (WebRTC). Therefore, please use the HT your commercial application.

Note: You can use `http://localhost` or `file://` URLs for local development.

The table below lists the supported URL domain names and protocols.

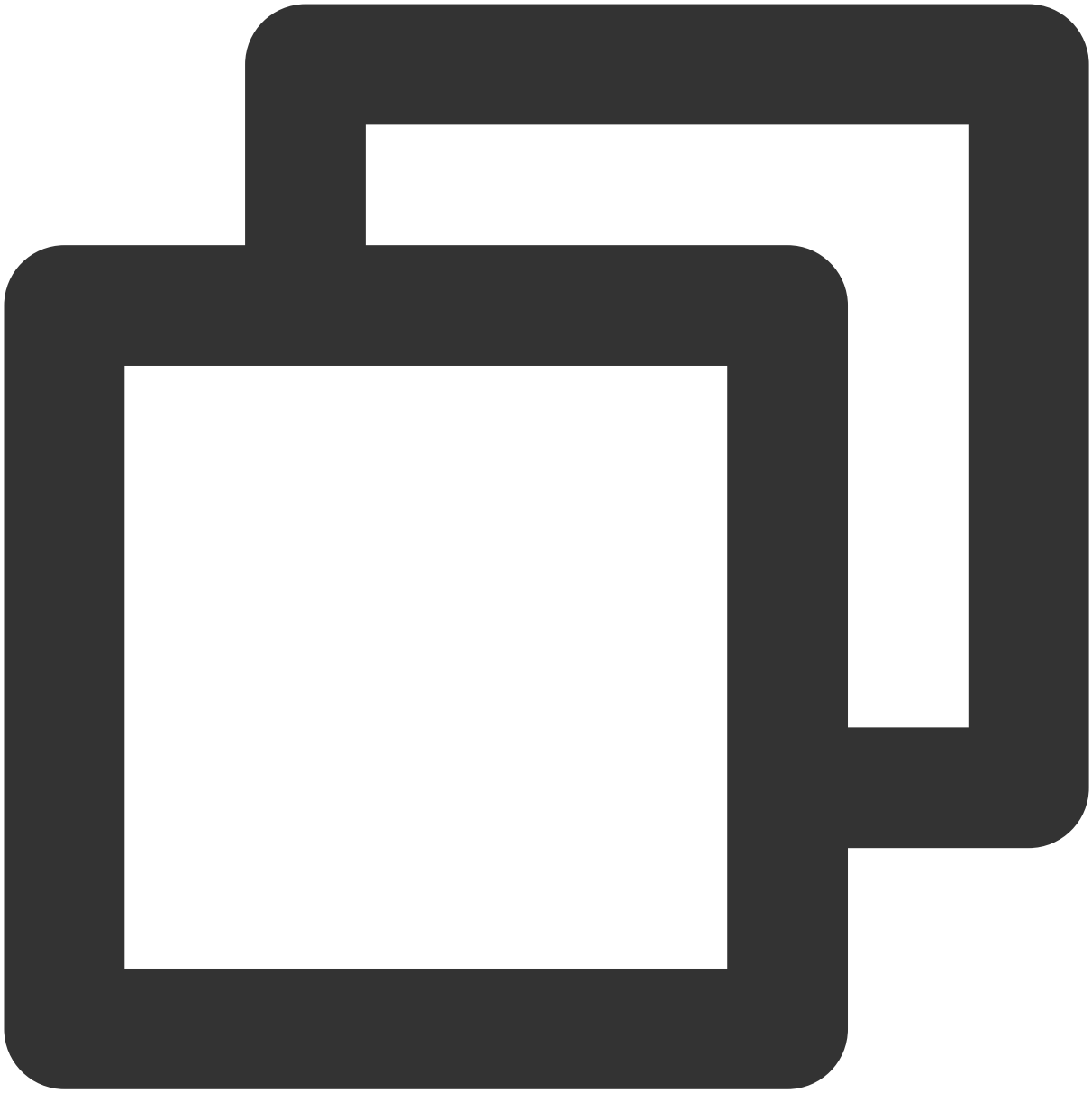
Scenario	Protocol	Receive (Playback)	Send (Publish)	Share Screen
Commercial	HTTPS	Supported	Supported	Supported
Commercial	HTTP	Supported	Not supported	Not supported
Local development	http://localhost	Supported	Supported	Supported
Local development	http://127.0.0.1	Supported	Supported	Supported
Local development	http://[local IP address]	Supported	Not supported	Not supported
Local development	file://	Supported	Supported	Supported

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);
```

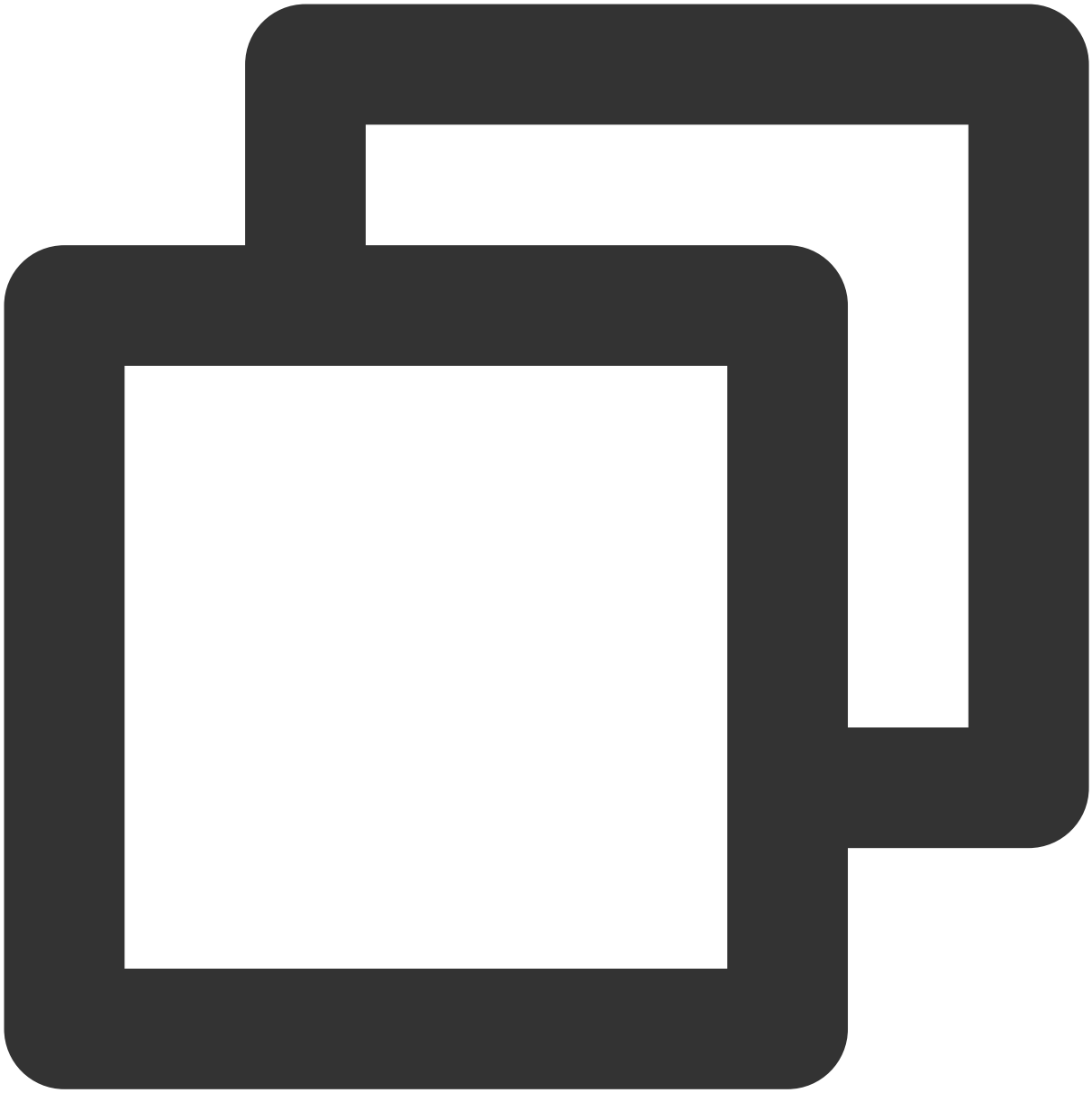
参数如下表所示：

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);
```

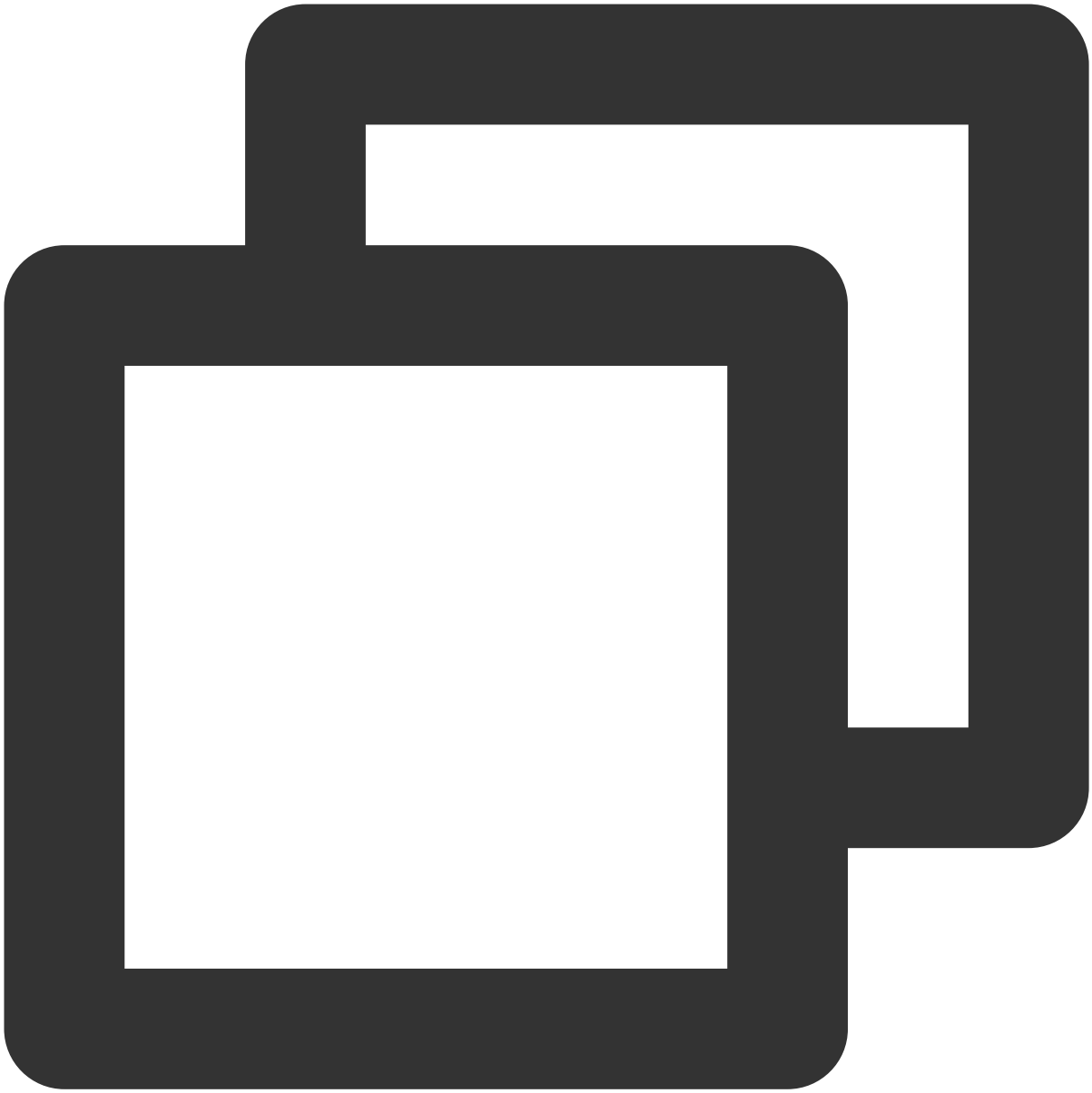
参数如下表所示：

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
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：

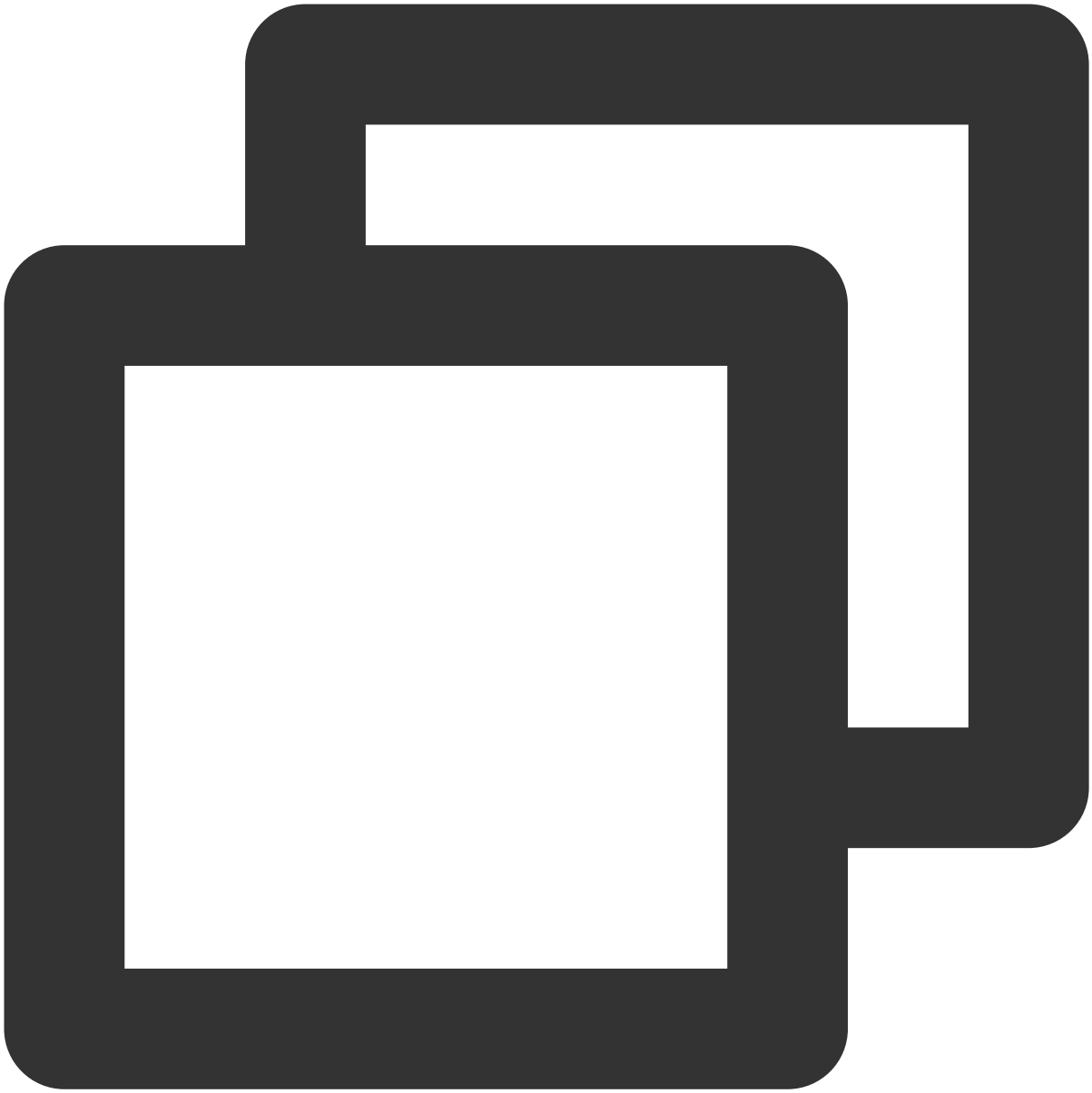
参数	类型	含义
roomId	number	房间 ID
roomParam	Object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭

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。

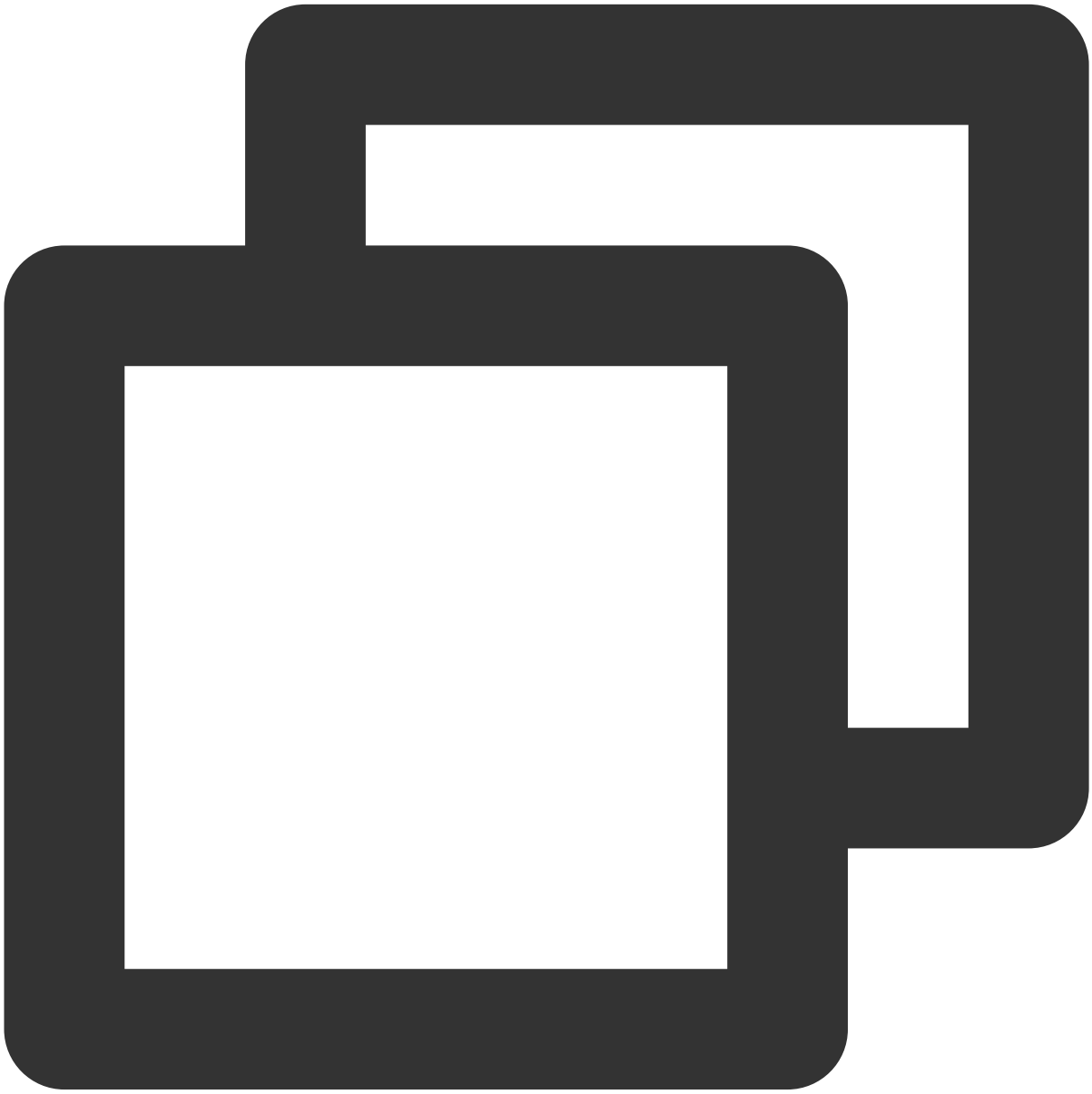


```
<template>
  <room ref="TUIRoomRef" @on-room-create="handleRoomCreate"></room>
</template>

<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。

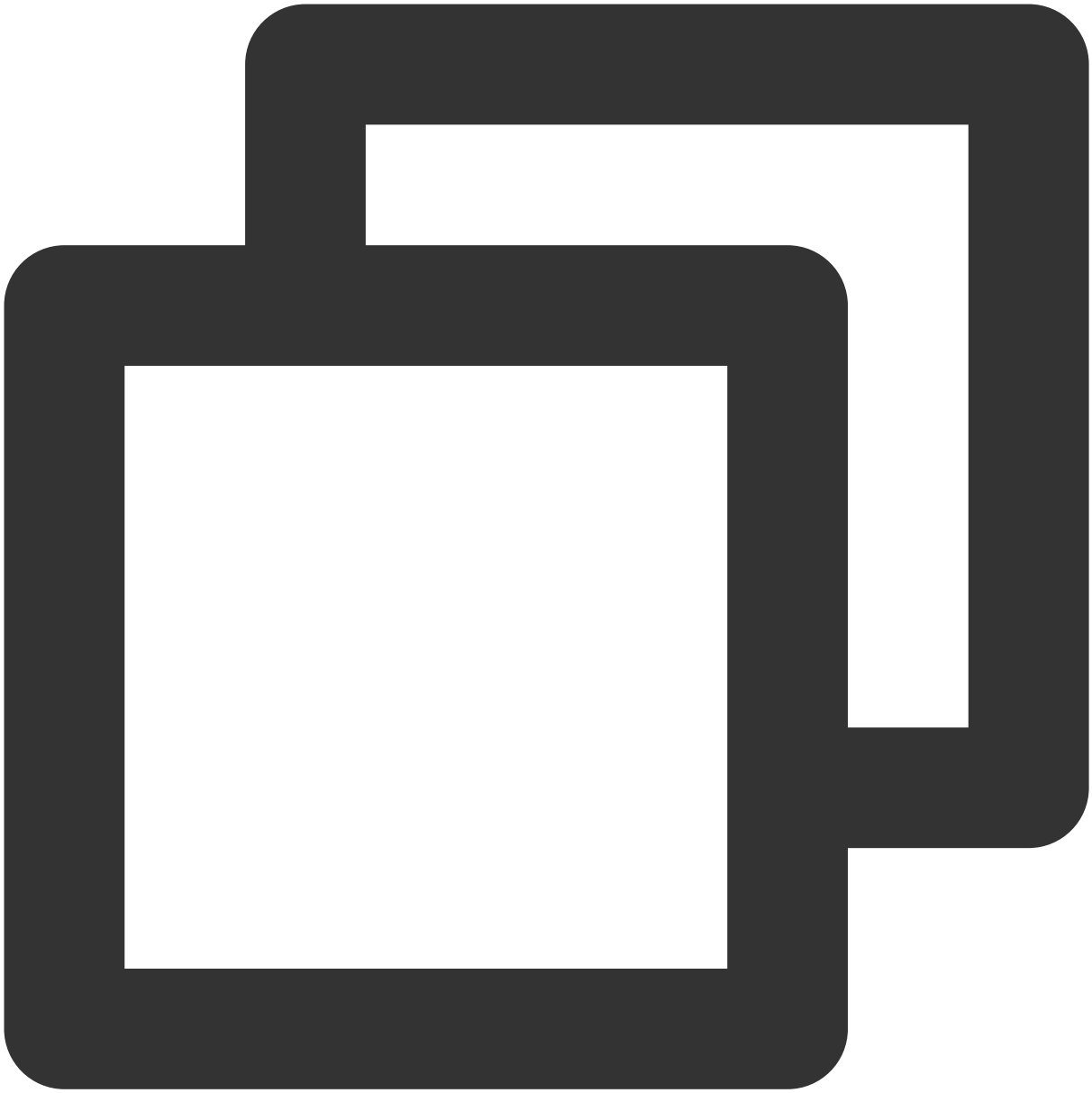


```
<template>
  <room ref="TUIRoomRef" @on-room-enter="handleRoomEnter"></room>
</template>
```

```
<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。

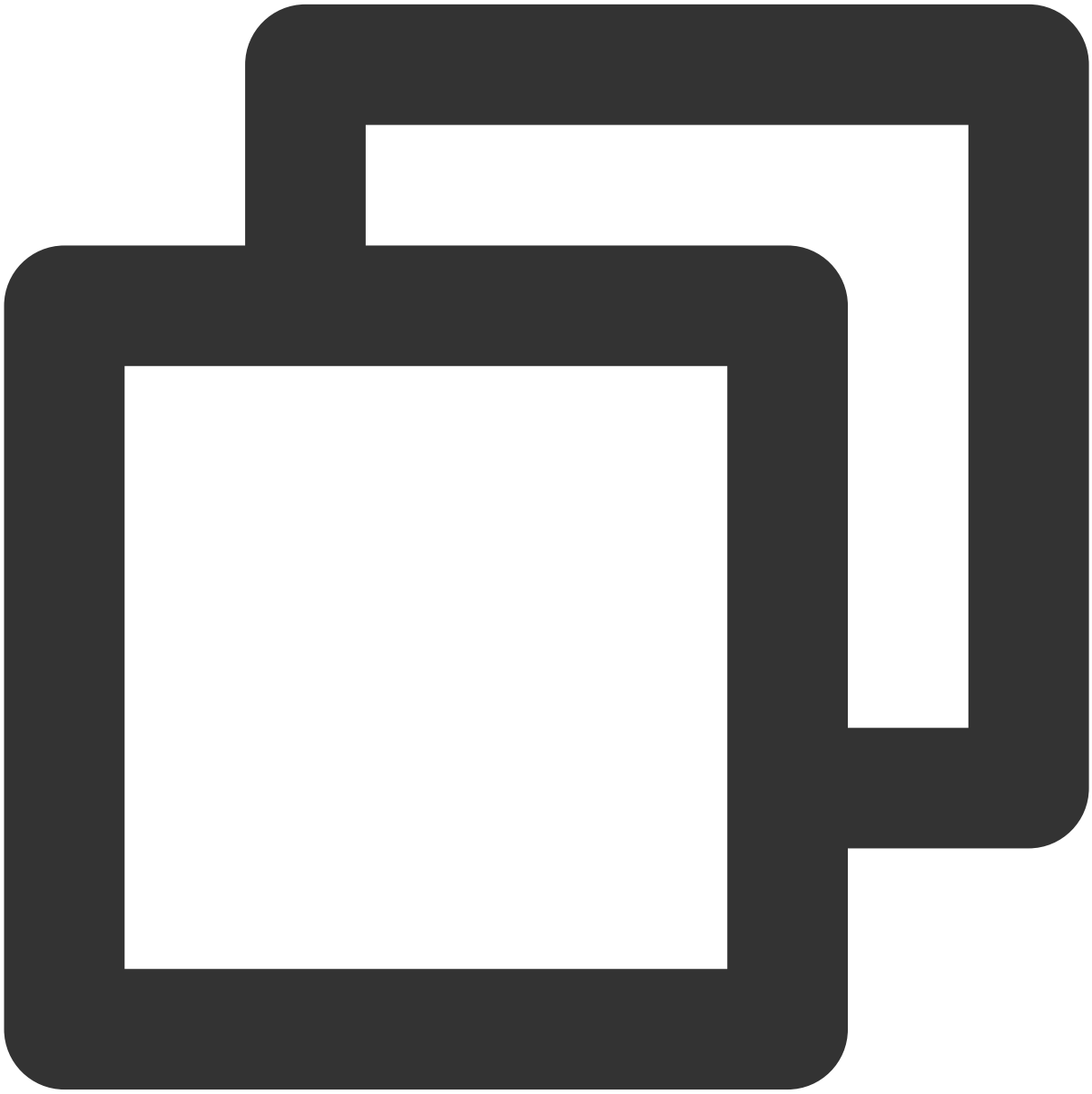


```
<template>
  <room ref="TUIRoomRef" @on-room-destory="handleRoomDestory"></room>
</template>

<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。

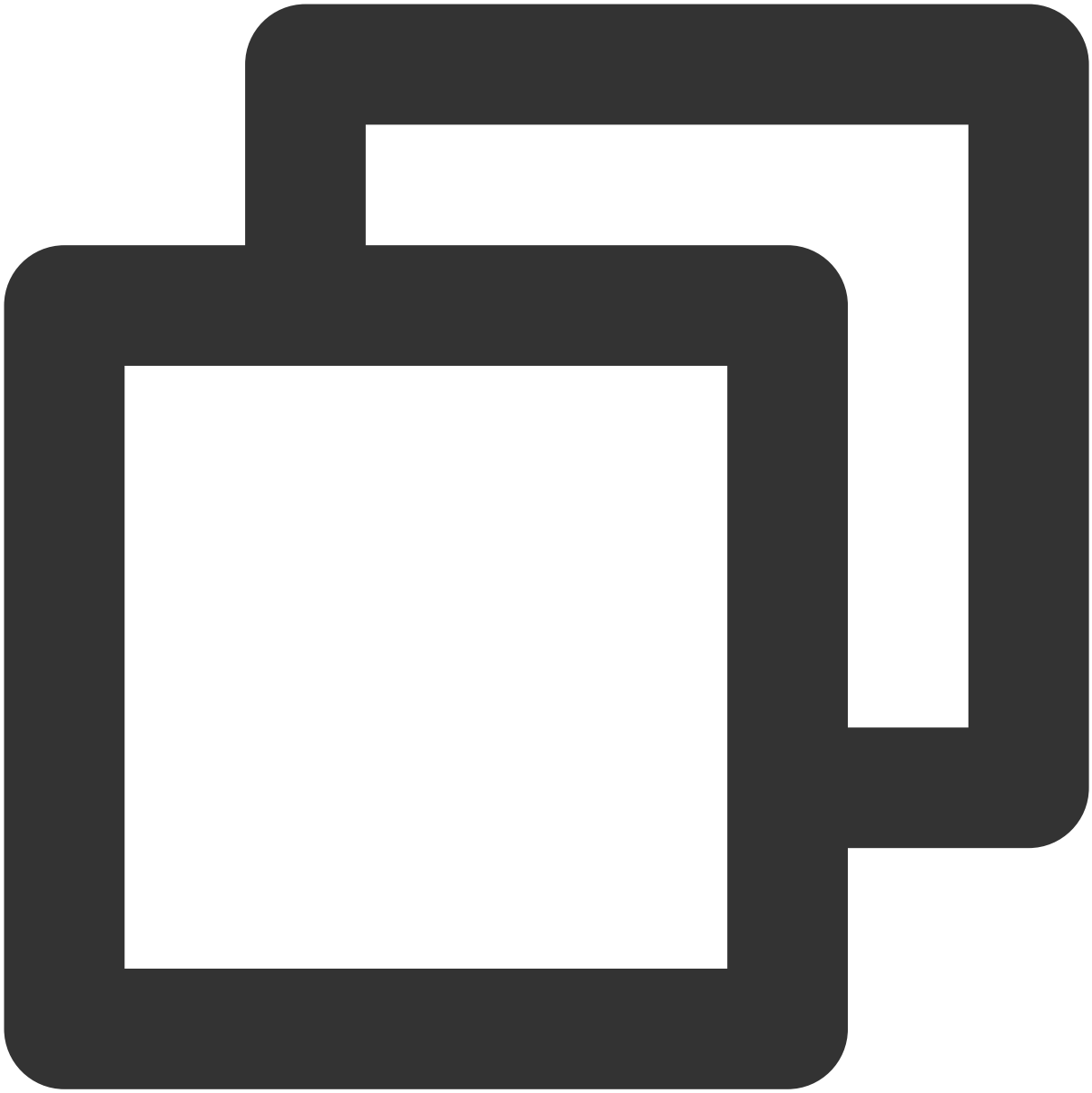


```
<template>
  <room ref="TUIRoomRef" @on-room-exit="handleRoomExit"></room>
</template>
```

```
<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>
```

onKickOff

普通成员被主持人踢出房间通知。



```
<template>
  <room ref="TUIRoomRef" @on-kick-off="handleKickOff"></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 handleKickOff(info) {
    if (info.code === 0) {
      console.log('普通成员被主持人踢出房间')
```

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

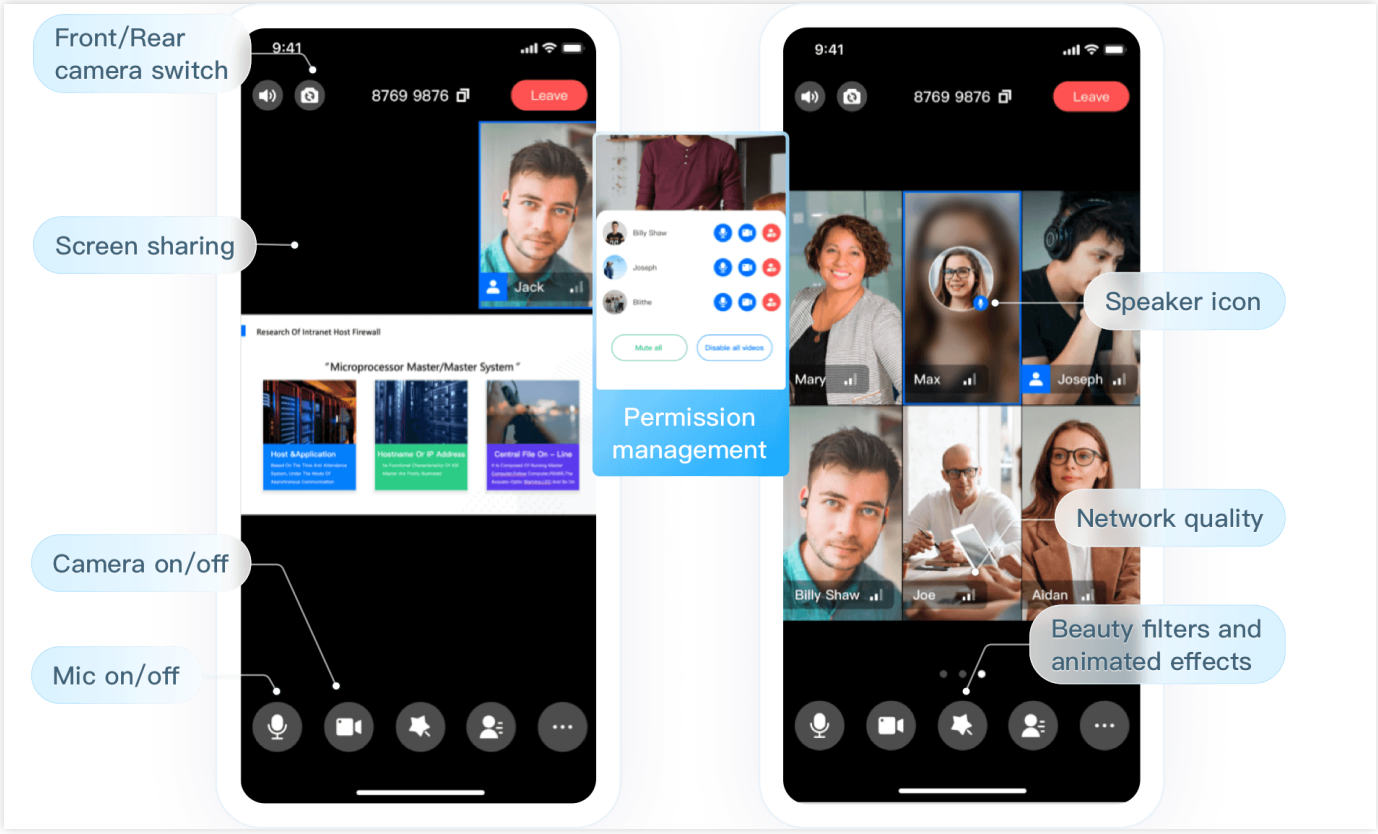
集成 TUIRoom (Android)

最近更新时间：2022-09-06 14:39:31

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：

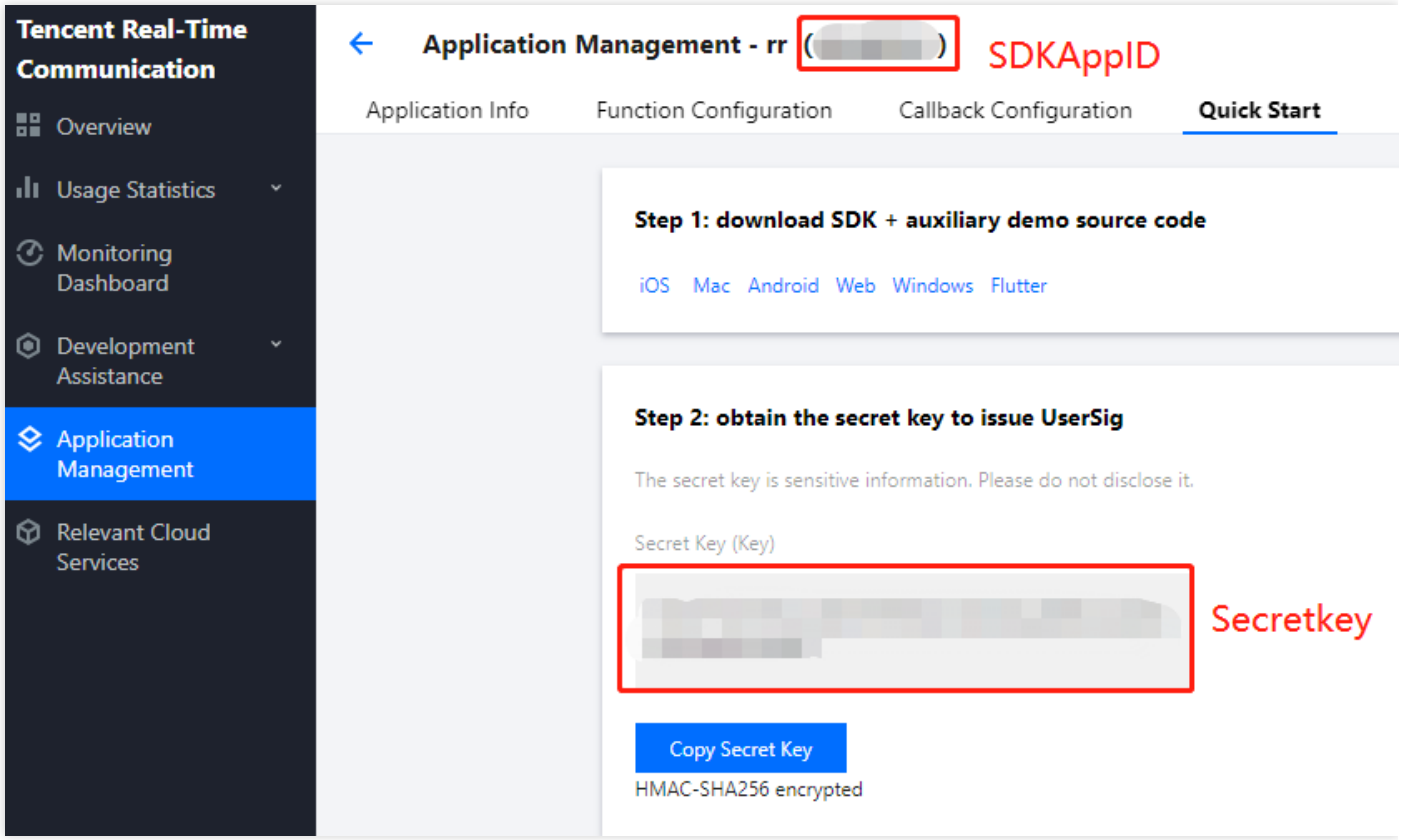
```
-keep class com.tencent.** { *; }
```

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，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$TXScreenCaptureAssistantActi
vity"
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
```

```
("package:" + 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.TRTCScreenShareParam
s();
mTUIRoom.stopCameraPreview();
mTUIRoom.startScreenCapture(encParams, params);
}
```

常见问题

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

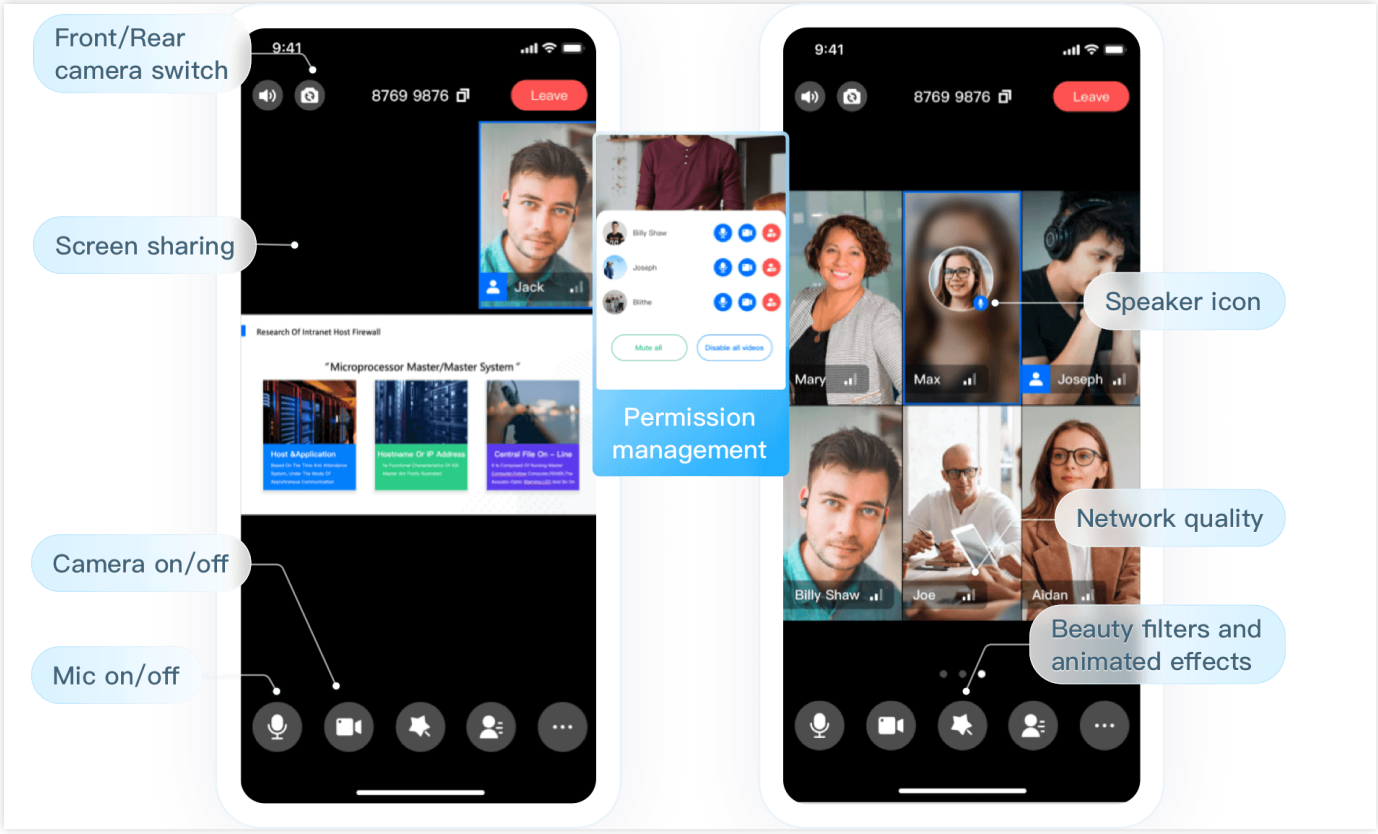
集成 TUIRoom (iOS)

最近更新时间：2022-07-22 15:13:43

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：

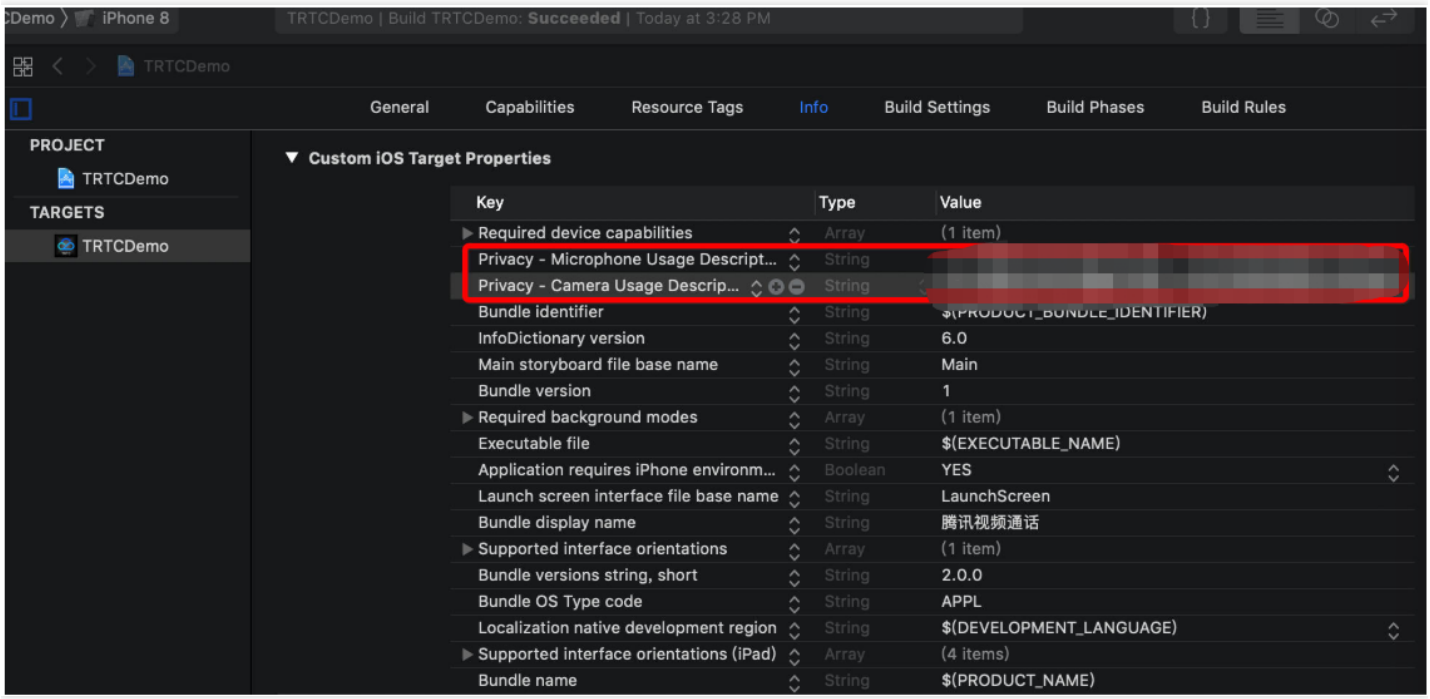
`Source`、`Resources` 文件夹 和 `TUIRoom.podspec` 文件必需在同一目录下。

- 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>
```

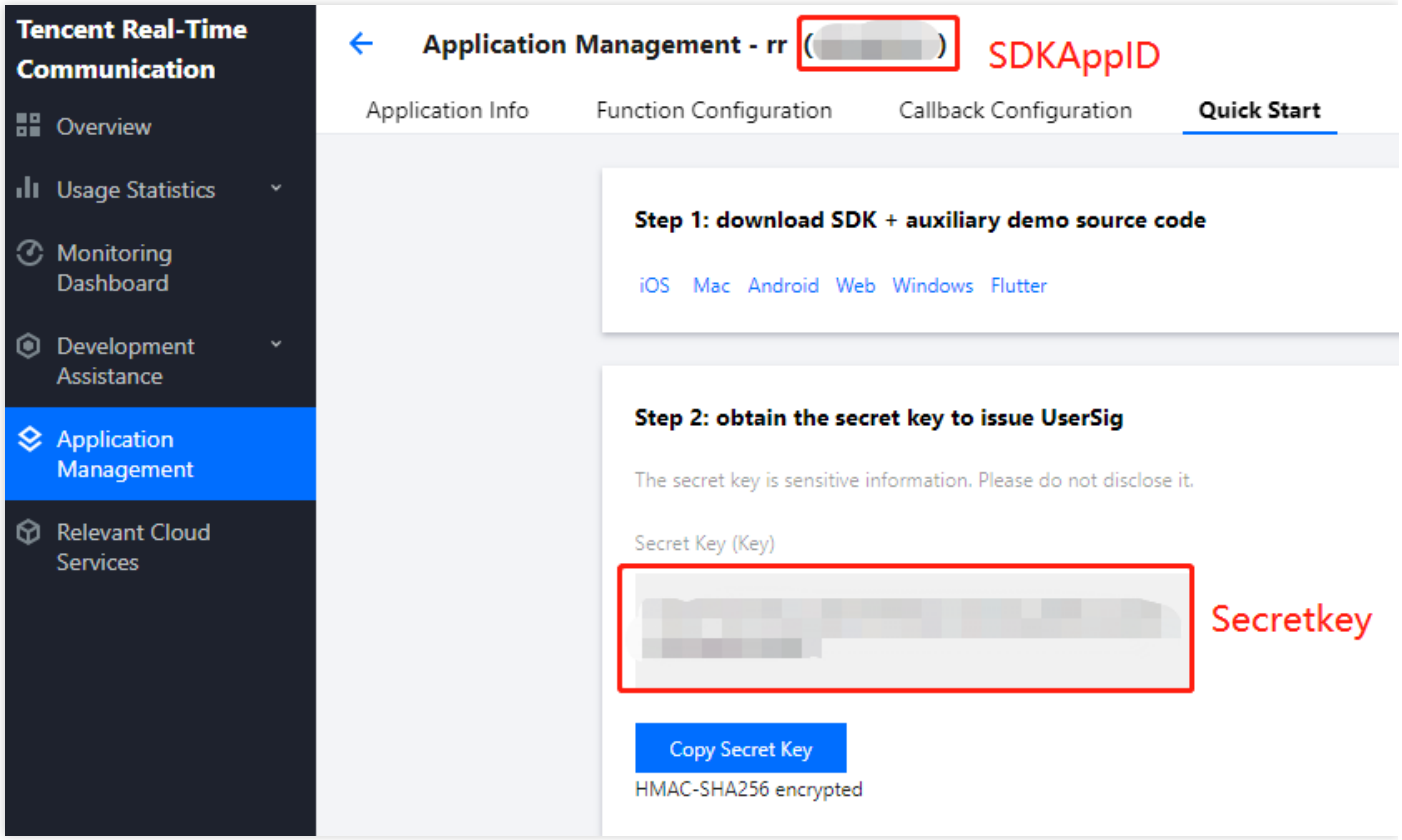


步骤三：创建并初始化 TUI 组件库

```
@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];
<dx-code-holder data-codeindex="2"></dx-code-holder>
```

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 ObjectiveC
   - Swift Swift

```
@import TUIRoom;

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

步骤五：房间管理（可选）

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

```
@import TUIRoom;

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

}];
<dx-code-holder data-codeindex="4"></dx-code-holder>
```

2. **成员离开房间 TUIRoomCore#leaveRoom**。

```
@import TUIRoom;

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

}];
<dx-code-holder data-codeindex="5"></dx-code-holder>
```

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)。

```
@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];
<dx-code-holder data-codeindex="6"></dx-code-holder>
```

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）：

```
:::
::: Swift 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
```

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

集成 TUIRoom (Flutter)

最近更新时间：2023-02-14 15:34:52

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。

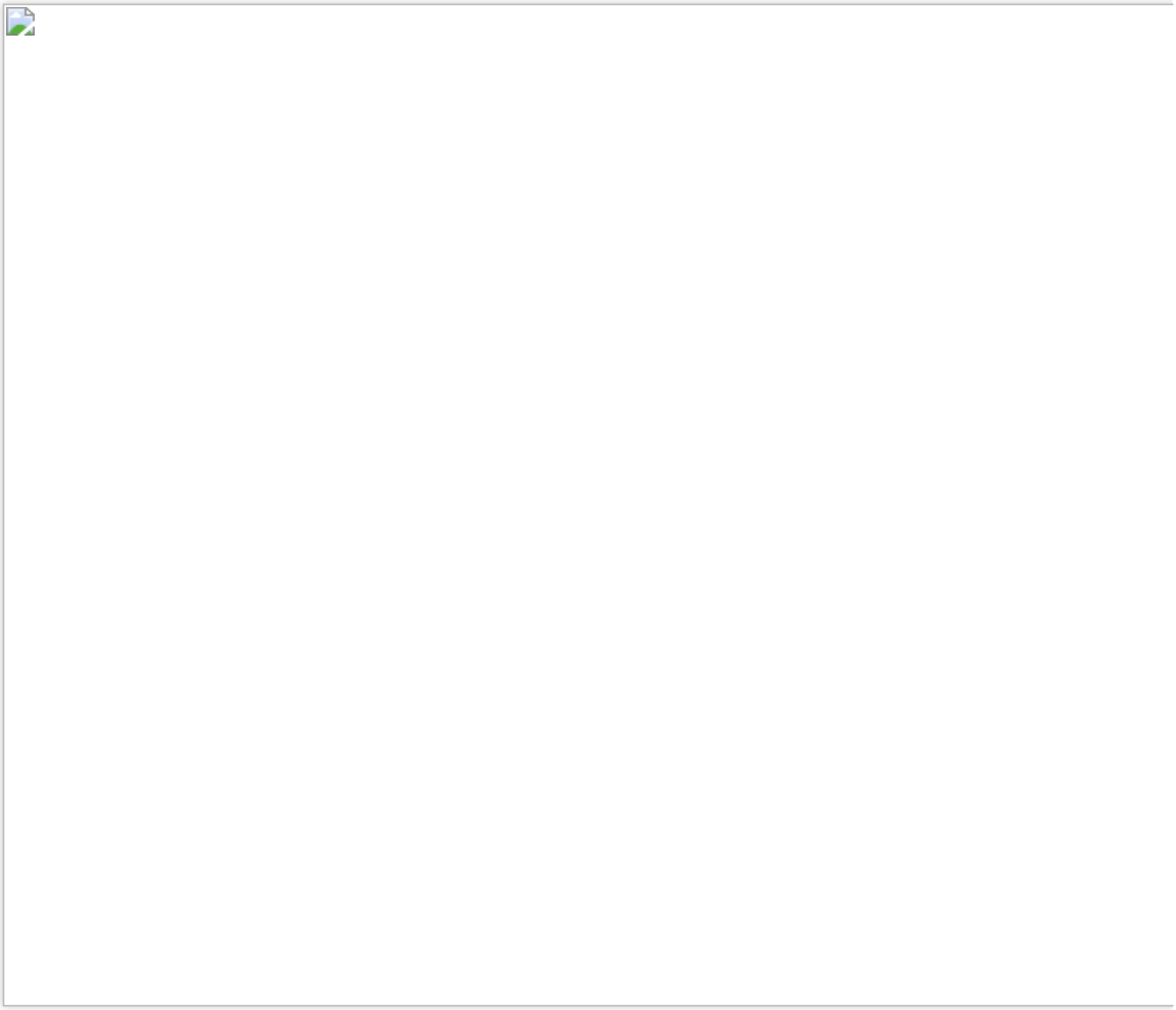
说明：

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。



注意：

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

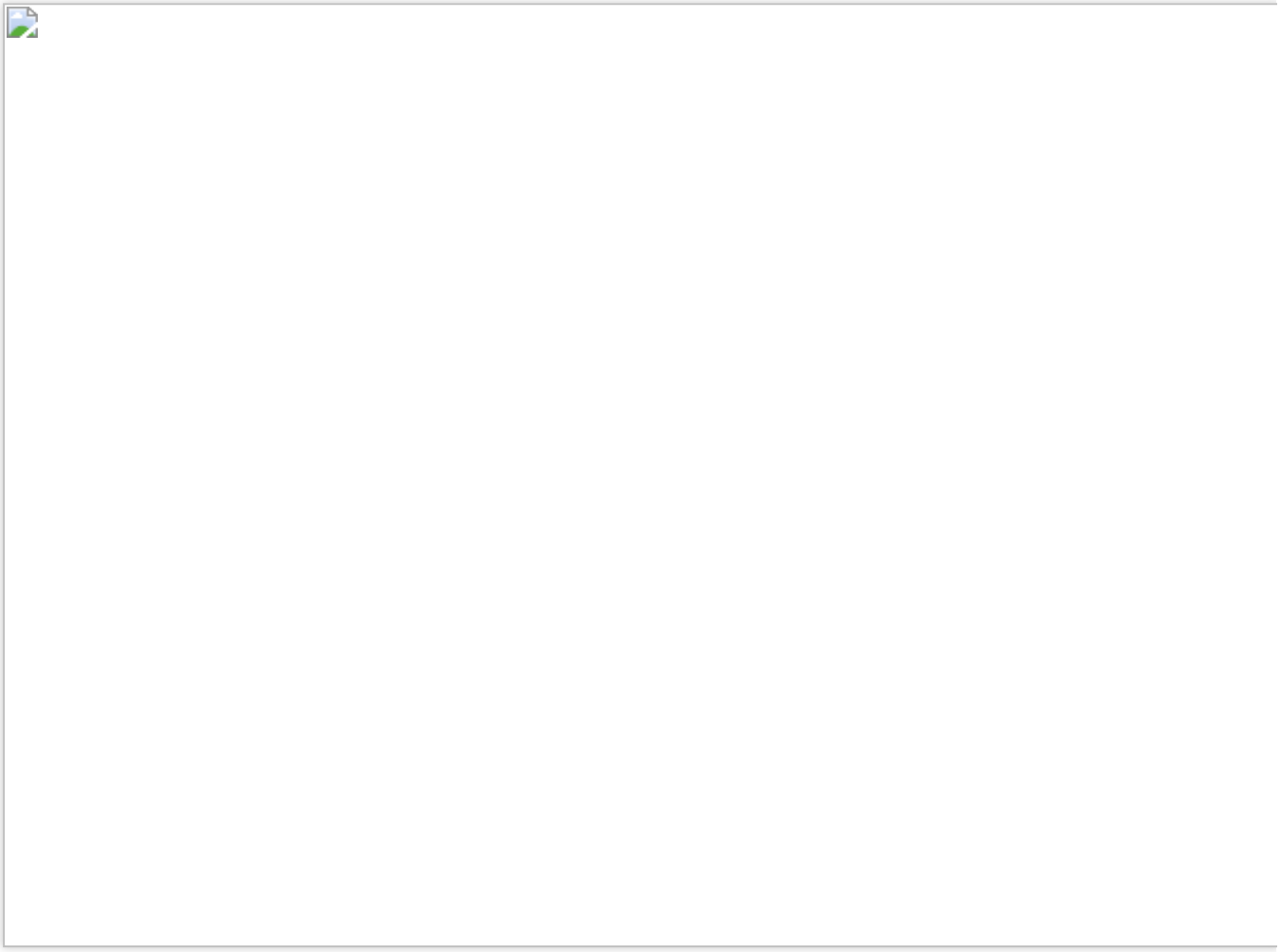
步骤2：下载 App 源码

单击进入 TRTCFlutterScenesDemo，Clone 或者下载源码。

步骤3：配置 App 文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `/lib/debug/GenerateTestUserSig.dart` 文件。
3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：

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



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

注意：

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

步骤4：编译运行

注意：

Android 端需要在真机下运行，不支持模拟器调试。

1. 执行 `flutter pub get` 。

2. 编译运行调试：

iOS 端

Android 端

1. 使用 XCode（11.0及以上的版本）打开源码目录下的 `/ios工程` 。
2. 编译并运行 Demo 工程即可。
1. 执行 `flutter run` 。
2. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击 **运行** 即可。

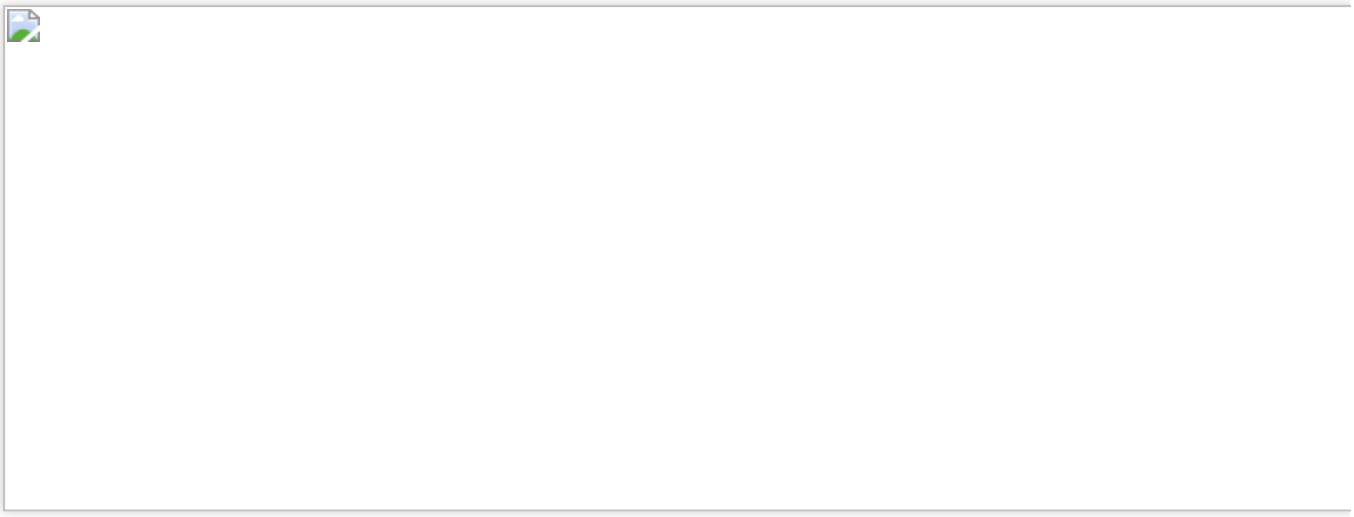
步骤5：修改 Demo 源代码

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

文件或文件夹	功能描述
TRTCMeetingIndex.dart	创建或进入会议界面
TRTCMeetingRoom.dart	视频会议的主界面
TRTCMeetingMemberList.dart	参会人员列表界面
TRTCMeetingSetting.dart	视频会议相关参数设置界面

实现自定义 UI 界面

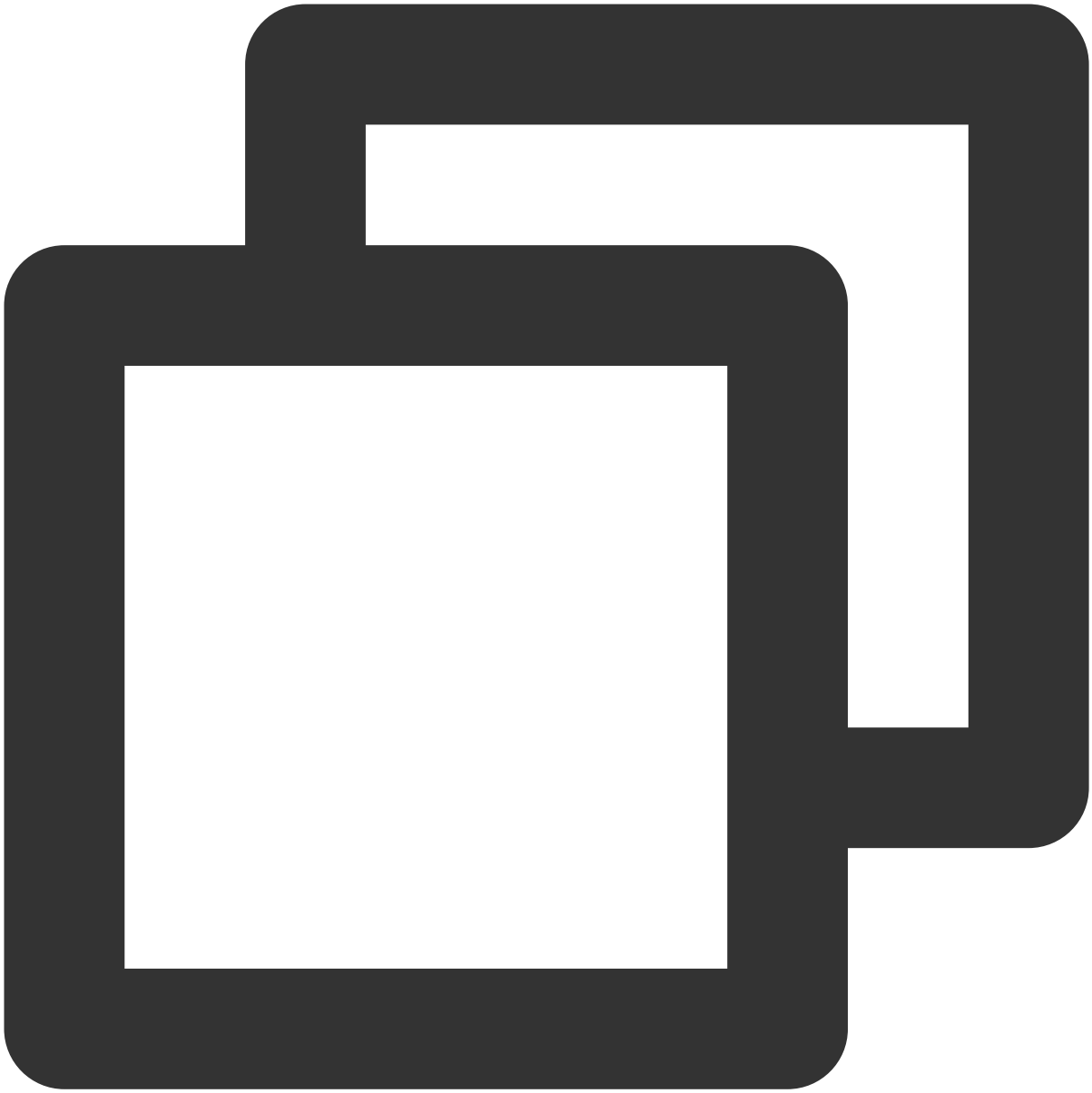
源码 中的 TRTCMeetingDemo 文件夹包含两个子文件夹 ui 和 model，model 文件夹中包含可重用的开源组件 TRTCMeeting，您可以在 `TRTCMeeting.dart` 文件中看到该组件提供的接口函数，并使用对应接口实现自定义 UI 界面。



步骤1：集成 SDK

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

在项目的 `pubspec.yaml` 中写如下依赖：



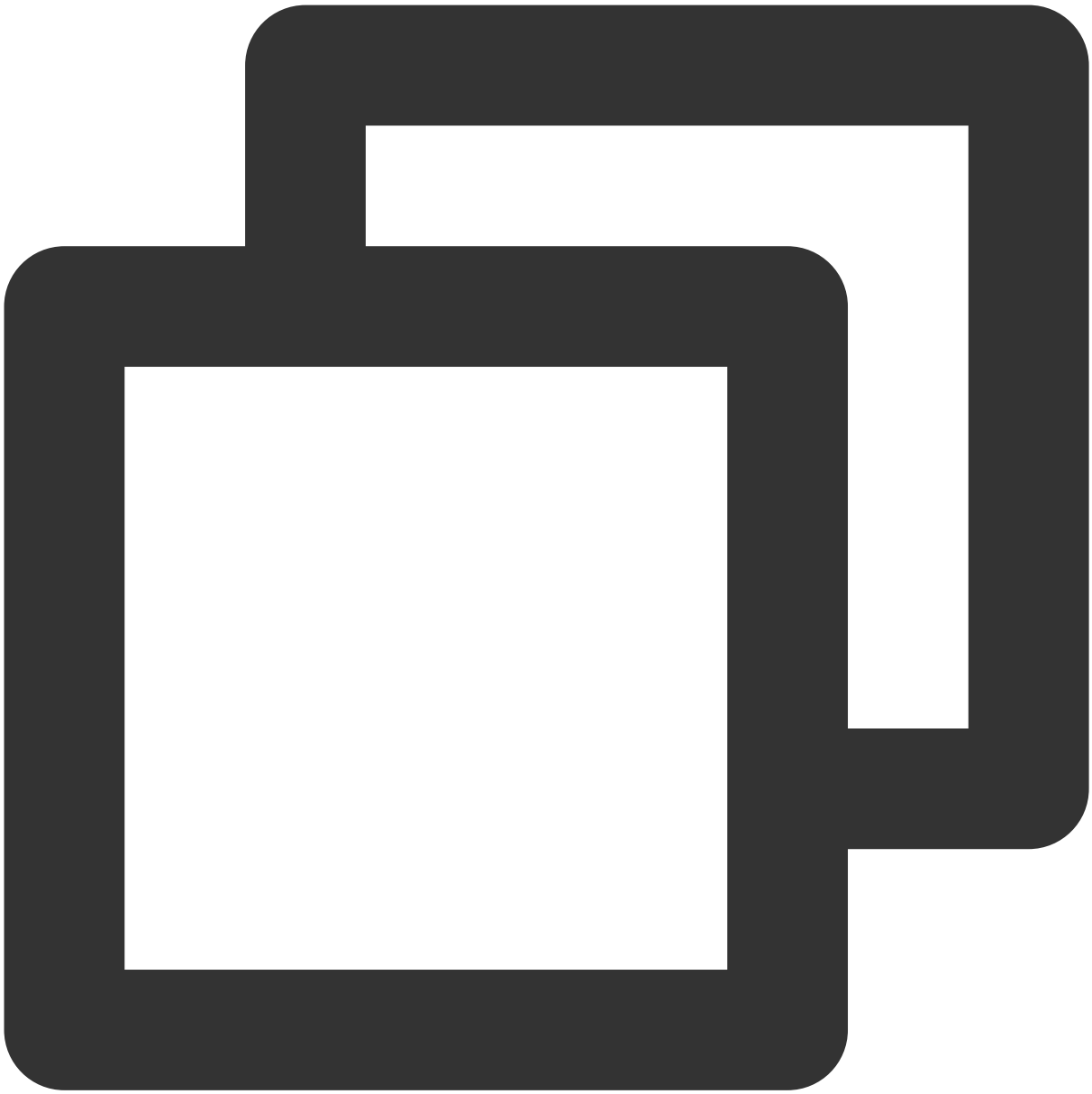
```
dependencies:
  tencent_trtc_cloud: 最新版本号
  tencent_im_sdk_plugin: 最新版本号
```

步骤2：配置权限及混淆规则

iOS 端

Android 端

需要在 `Info.plist` 中加入对相机和麦克风的权限申请：



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

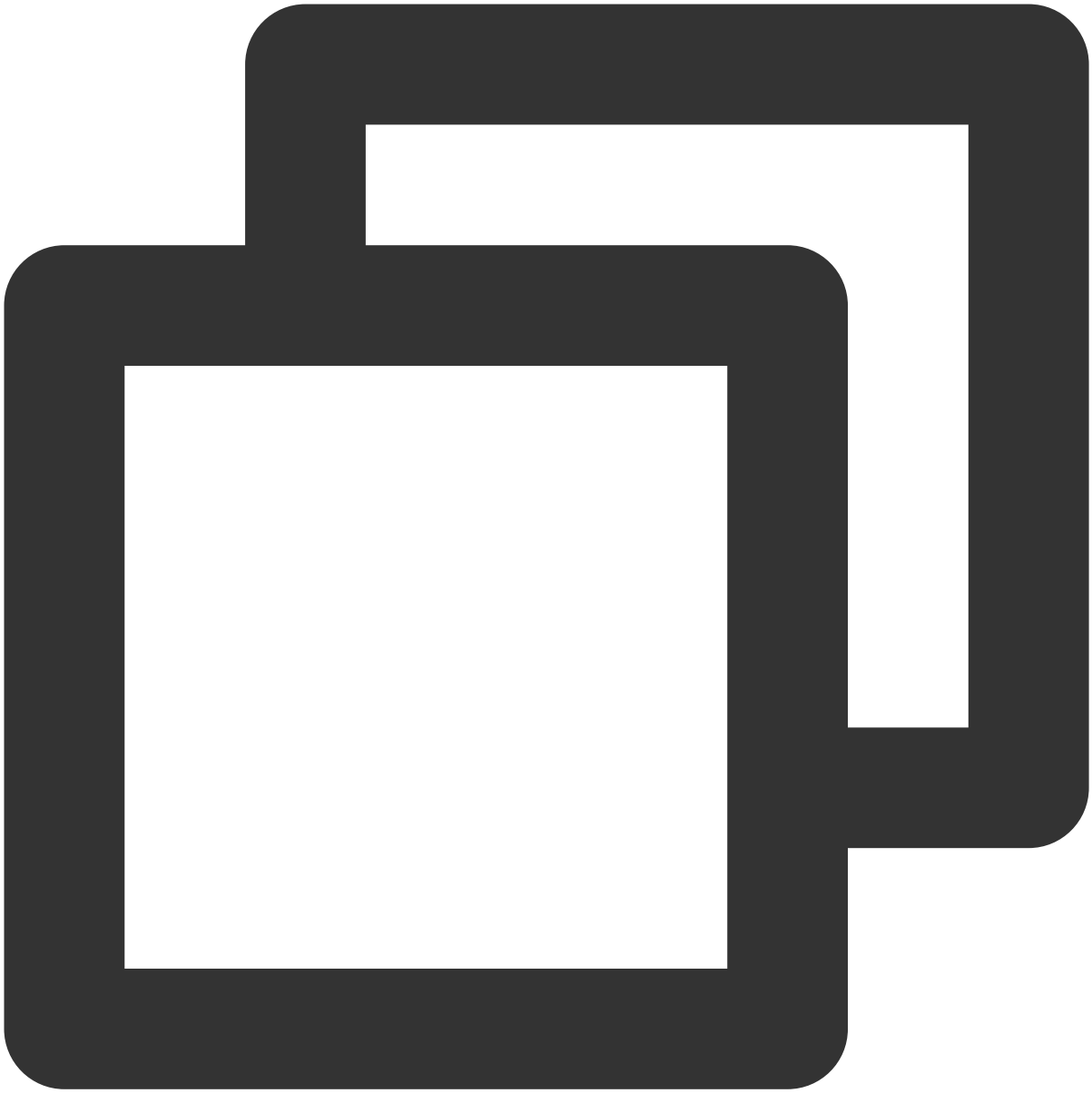
1. 打开 `/android/app/src/main/AndroidManifest.xml` 文件。
2. 将 `xmlns:tools="http://schemas.android.com/tools"` 加入到 manifest 中。
3. 将 `tools:replace="android:label"` 加入到 application 中。

说明：

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

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：



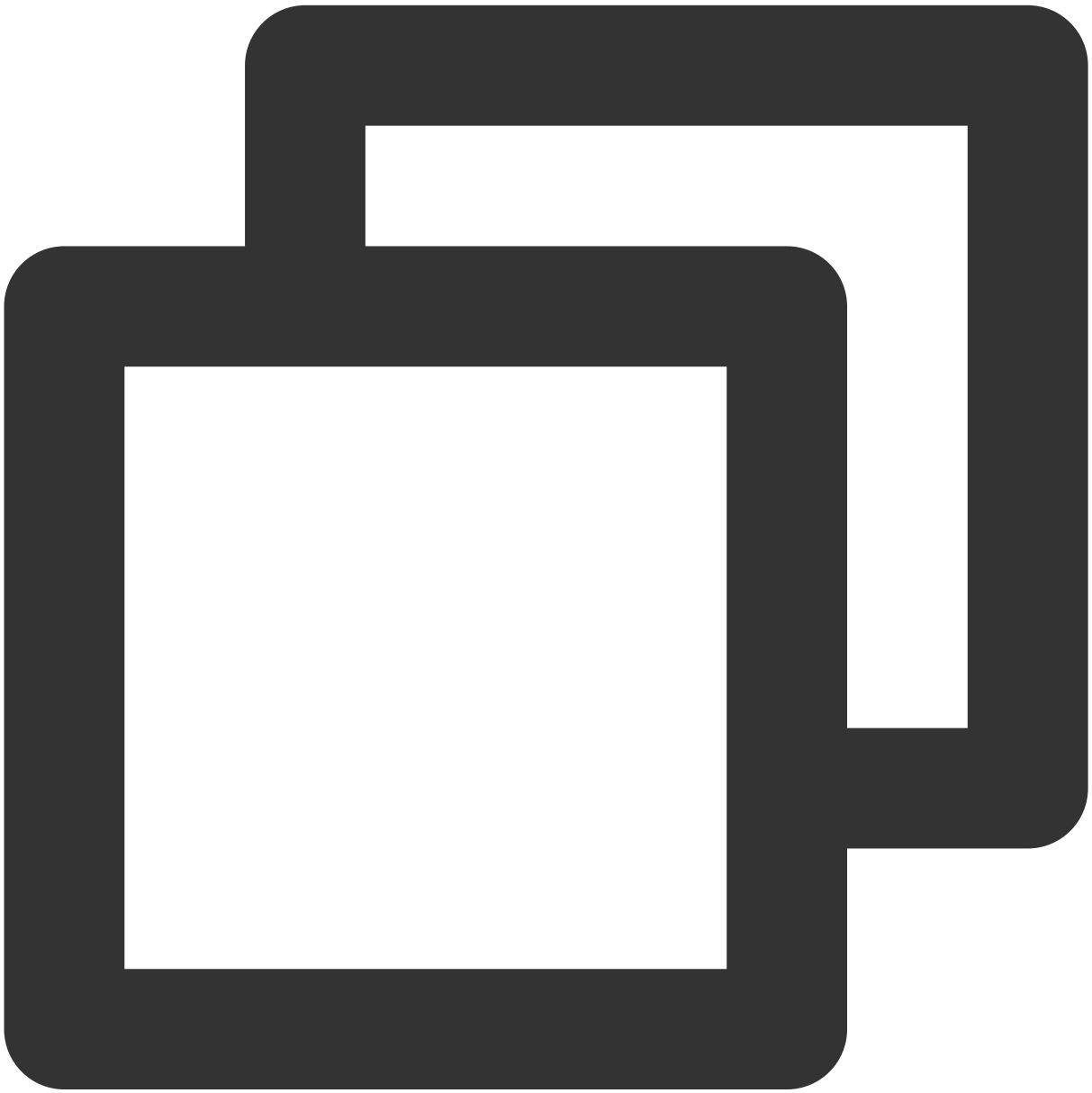
```
lib/TRTCMeetingDemo/model/
```

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(
```

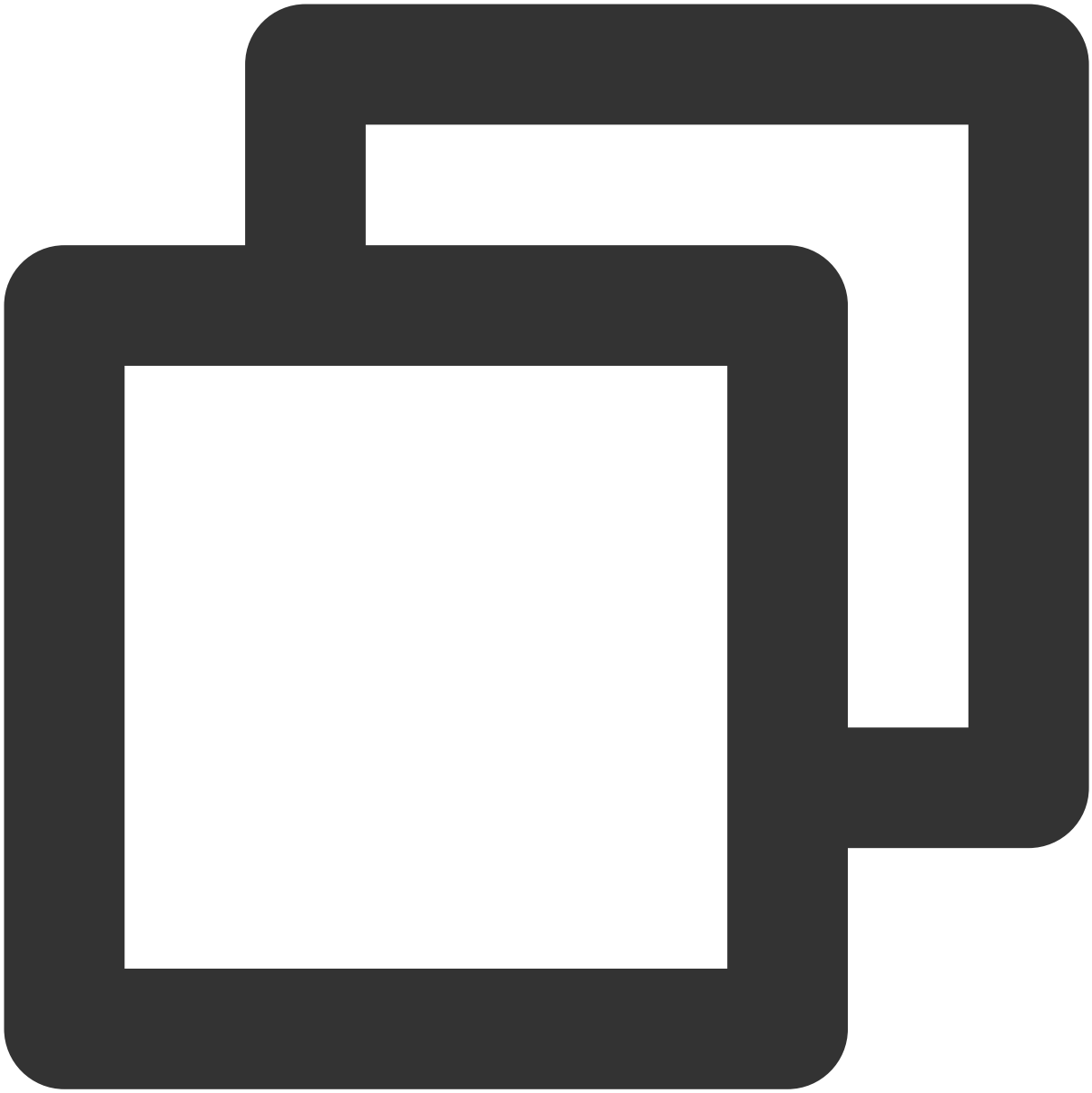
```
    GenerateTestUserSig.sdkAppId,
    userId,
    GenerateTestUserSig.genTestSig(userId),
);
if (res.code == 0) {
    // 登录成功
}
```

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。

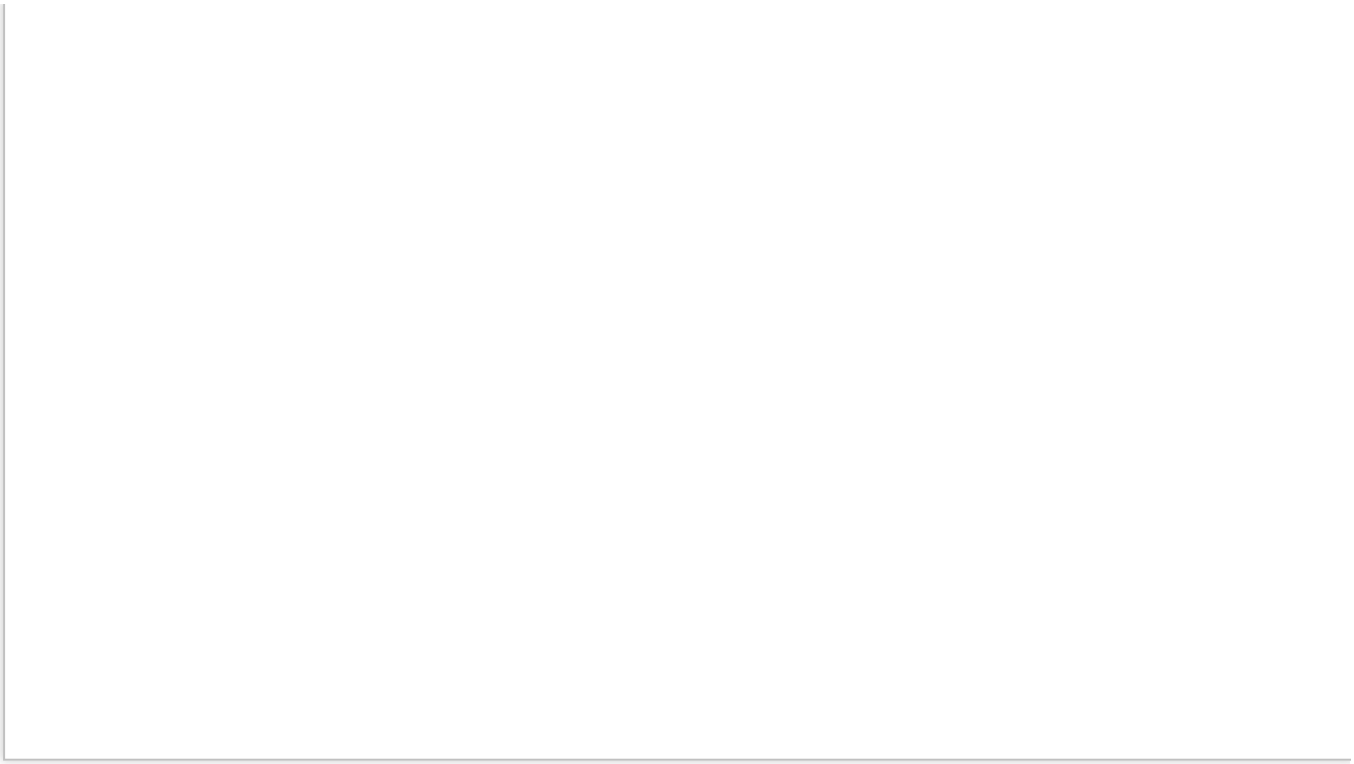


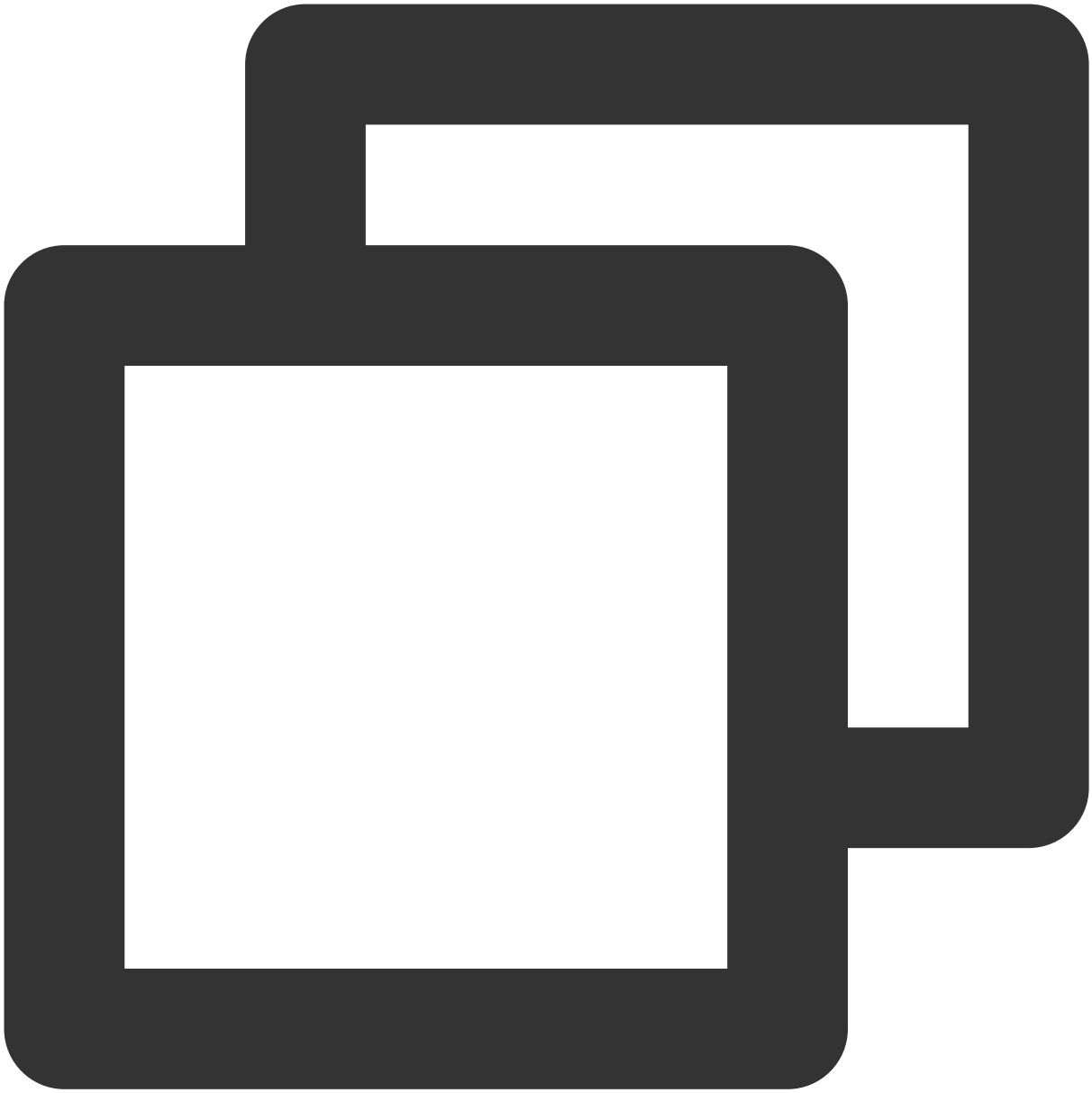
```
// 1. 主持人设置昵称和头像
trtcMeeting.setSelfProfile('my_name', 'my_avatar');

// 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_PI
    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
```

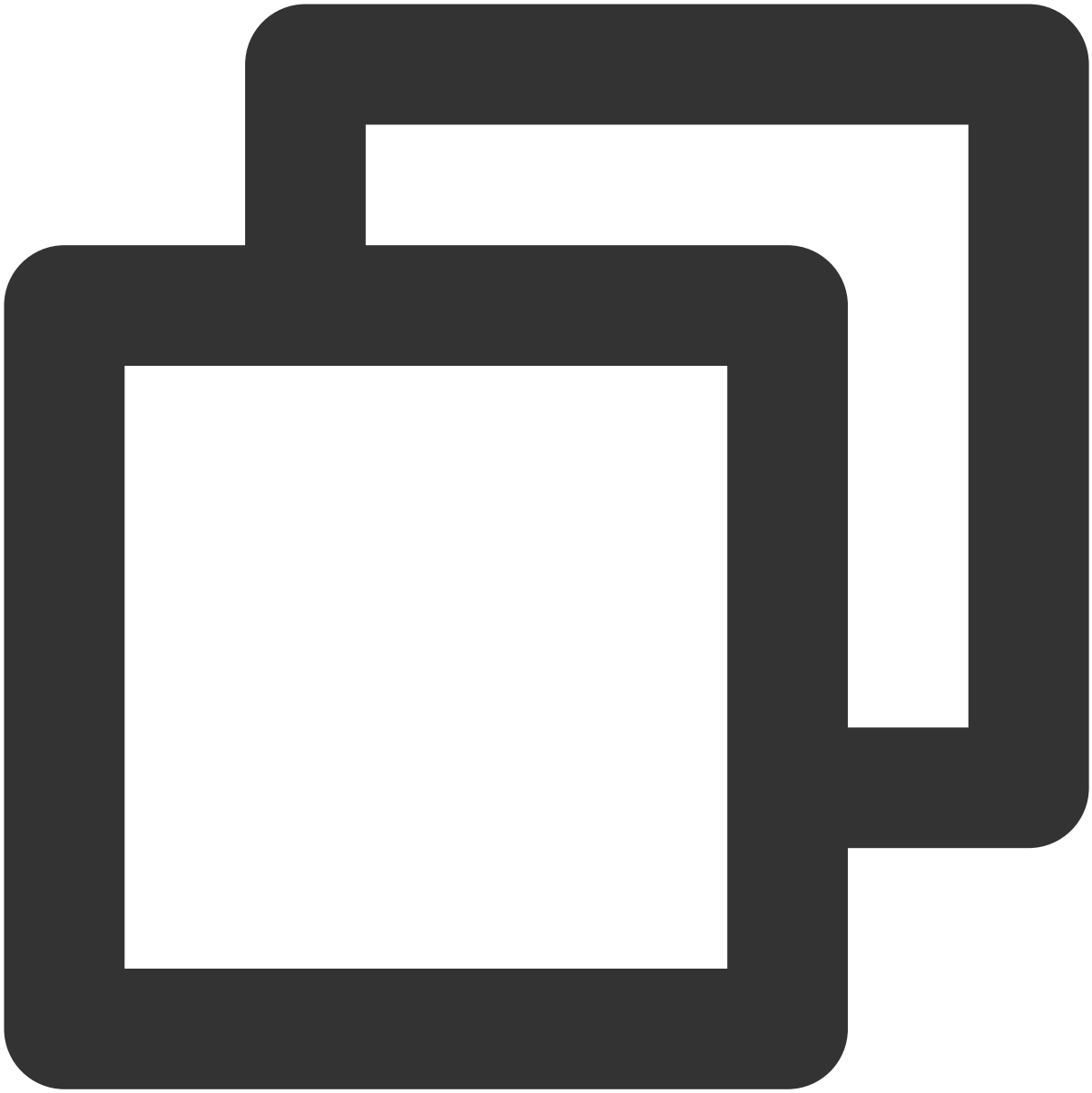
```
    trtcMeeting.getBeautyManager().setBeautyStyle(TRTCCloudDef.TRTC_BEAUTY_STYLE_PI
    trtcMeeting.getBeautyManager().setBeautyLevel(6);
}

// 5. 参会成员收到其他成员摄像头打开的通知，开始播放
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onUserVideoAvailable:
            if (param['available']) {
                trtcMeeting.startCameraPreview(
                    param['userId'],
                    TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
                    TRTCCloudVideoViewId
                );
            } else {
                trtcMeeting.stopRemoteView(
                    param['userId'],
                    TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG
                );
            }
            break;
    }
}
```

步骤7：屏幕分享

1. 屏幕分享功能需向系统申请悬浮窗权限，需要您在 UI 中实现具体的逻辑。
2. 调用 startScreenCapture，传入编码参数和录屏过程中的悬浮窗即可实现屏幕分享功能，具体信息请参见 TRTC SDK。
3. 会议中其他成员会收到 onUserVideoAvailable 的事件通知。

注意：

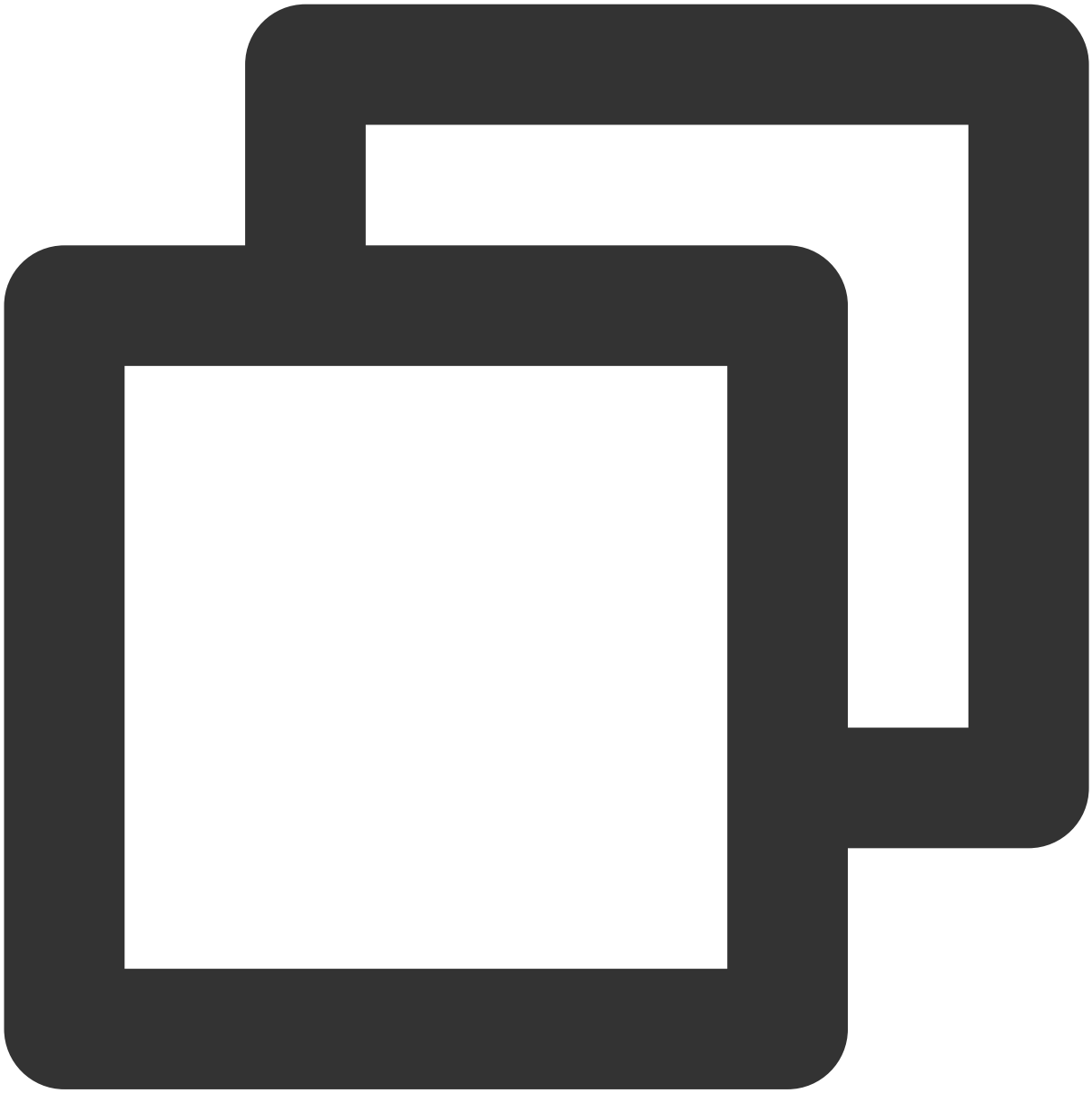
屏幕分享和摄像头采集是两个互斥的操作，如果需要打开屏幕分享功能，请先调用 stopCameraPreview 关闭摄像头采集。详情请参见 TRTC SDK。



```
await trtcMeeting.stopCameraPreview();
trtcMeeting.startScreenCapture(
    videoFps: 10,
    videoBitrate: 1600,
    videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720,
    videoResolutionMode: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT,
    appGroup: iosAppGroup,
);
```

步骤8：实现文字聊天和禁言消息

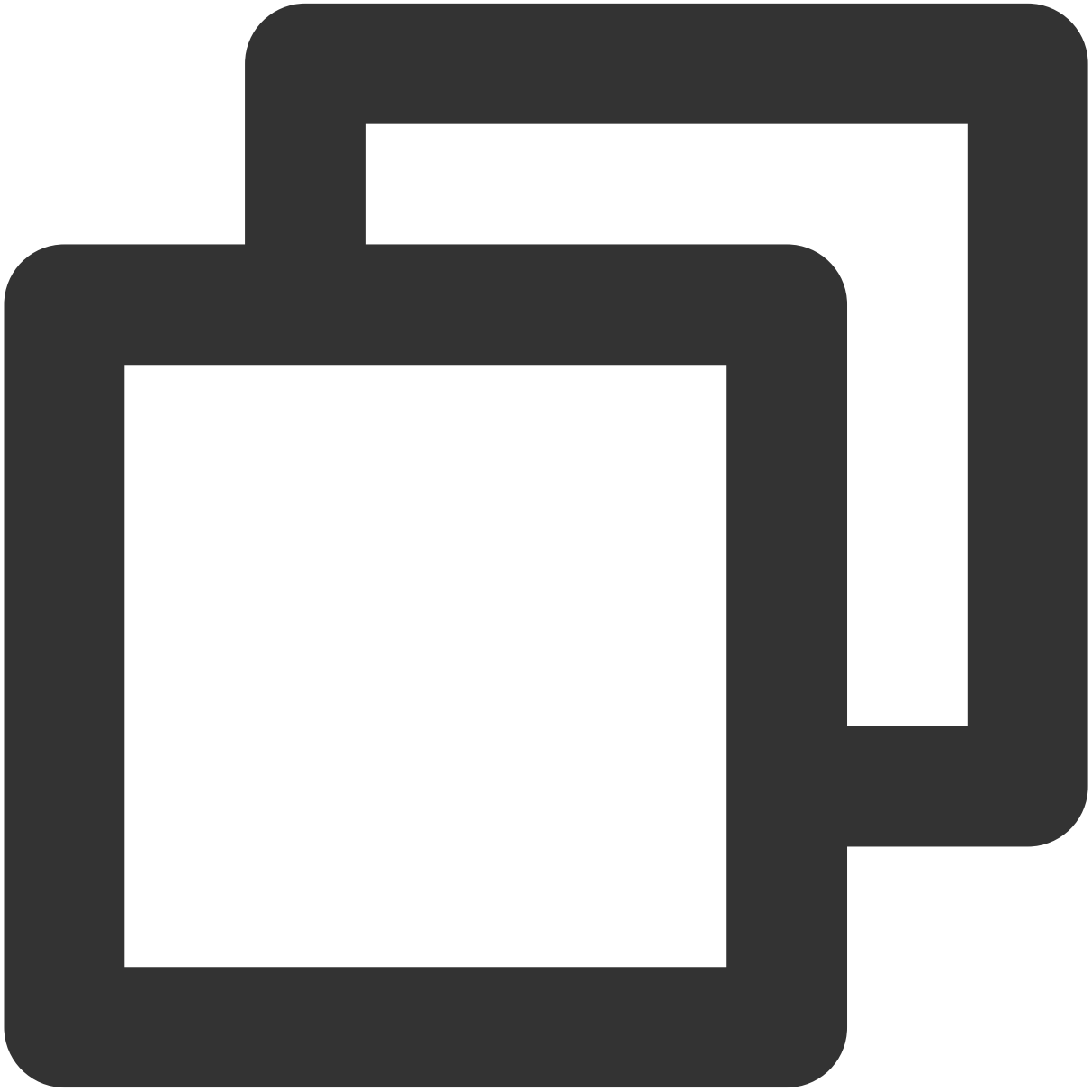
通过 sendRoomTextMsg 可以发送普通的文本消息，所有在该会议内的参会人员均可以收到 onRecvRoomTextMsg 回调。即时通信 IM 后台有默认的敏感词过滤规则，被判定为敏感词的文本消息不会被云端转发。



```
// 发送端：发送文本消息
trtcMeeting.sendRoomTextMsg('Hello Word!');
// 接收端：监听文本消息
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onRecvRoomTextMsg:
```

```
            print('收到来自' + param['userName'] + '的消息：' + param['message']);
            break;
    }
}
```

通过 sendRoomCustomMsg 可以发送自定义（信令）的消息，所有在该会议内的参会人员均可以收到 onRecvRoomCustomMsg 回调。自定义消息常用于传输自定义信令，例如用于禁言之类的会场控制等。



```
// 发送端：您可以通过自定义 cmd 来区分禁言通知
// eg: "CMD_MUTE_AUDIO"表示禁言通知
```

```
trtcMeeting.sendRoomCustomMsg('CMD_MUTE_AUDIO', '1');
// 接收端：监听自定义消息
trtcMeeting.registerListener(onListener);
onListener(TRTCMeetingDelegate type, param) {
    switch (type) {
        case TRTCMeetingDelegate.onRecvRoomCustomMsg:
            if (param['command'] == 'CMD_MUTE_AUDIO') {
                // 收到禁言通知
                print('收到来自' + param['userName'] + '的禁言通知：' + param['message
                trtcMeeting.muteLocalAudio(message == '1');
            }
            break;
    }
}
```

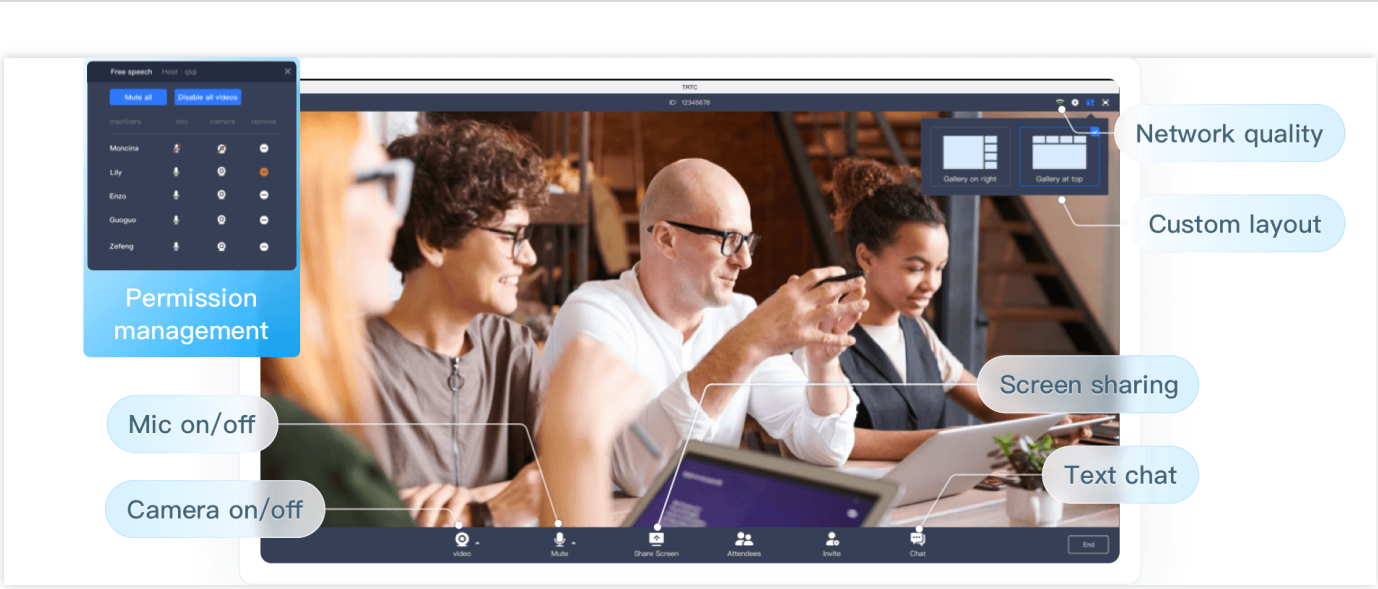
集成 TUIRoom (Windows&Mac)

最近更新时间：2022-07-22 15:13:43

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

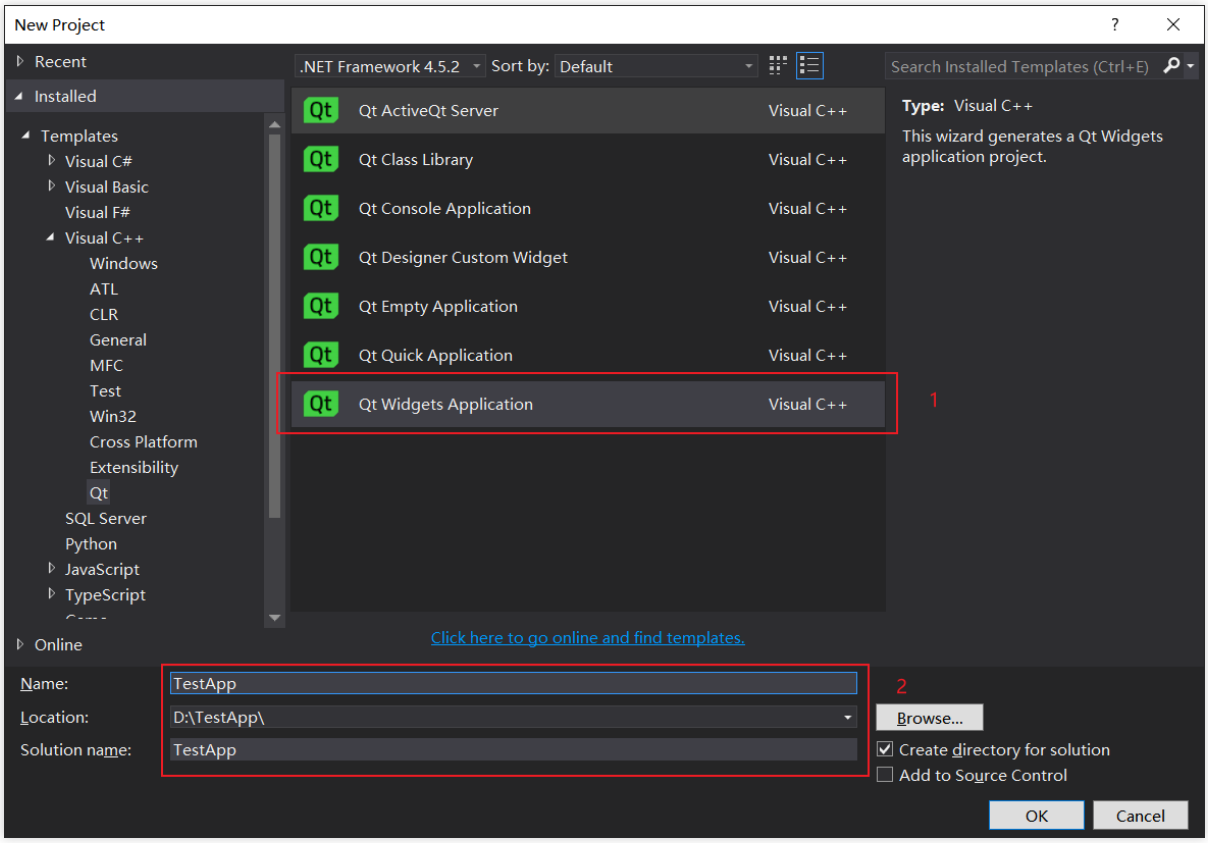
环境准备

- **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
   - Mac 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"                                    Start the RoomApp

#include <QProcess>
#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();          Start the RoomApp when you start the
}
```

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。

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
聊天室	Windows-Mac\RoomApp\App\ChatRoomViewController.cpp

功能点	文件目录
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

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

集成 TUIRoom (Electron)

最近更新时间：2022-10-31 14:41:31

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。

您可以单击在线体验链接： Mac OS版 及 Windows版 下载体验 TUIRoom Electron 更多功能。
您也可以单击 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。

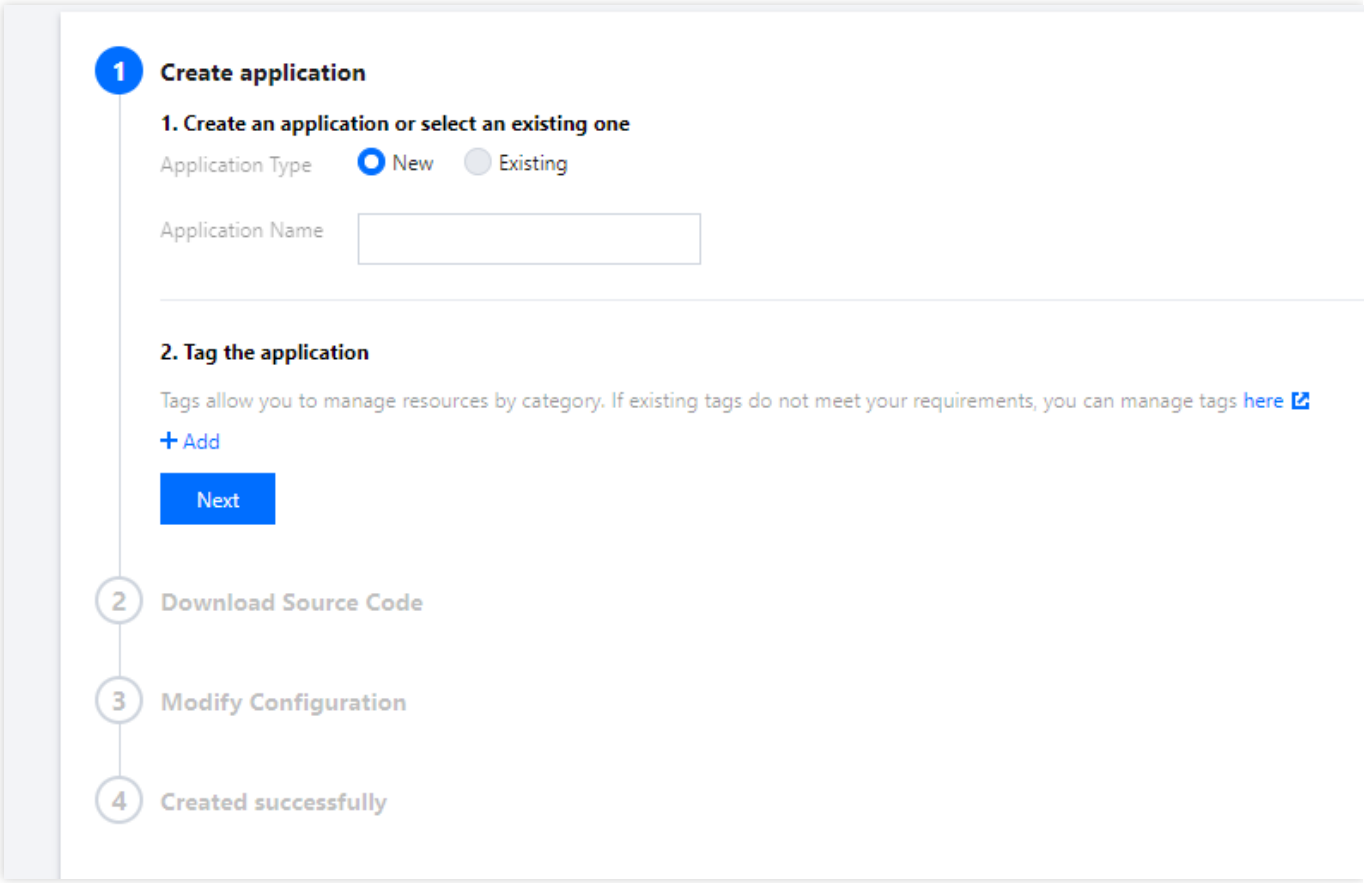
组件集成

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。

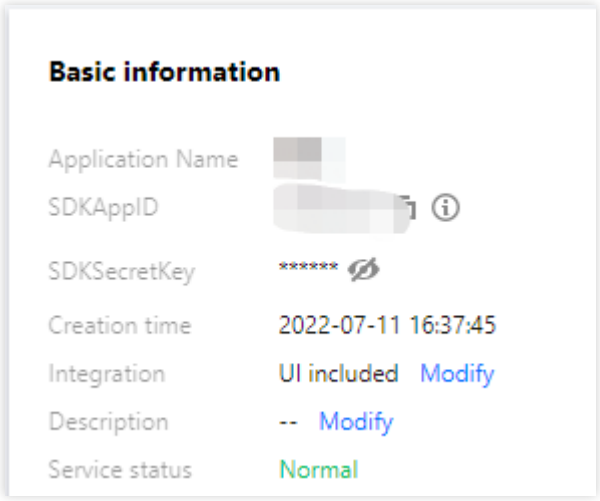
步骤一： 开通腾讯云实时音视频及即时通信服务

TUIRoom 基于腾讯云实时音视频和即时通信服务进行开发。

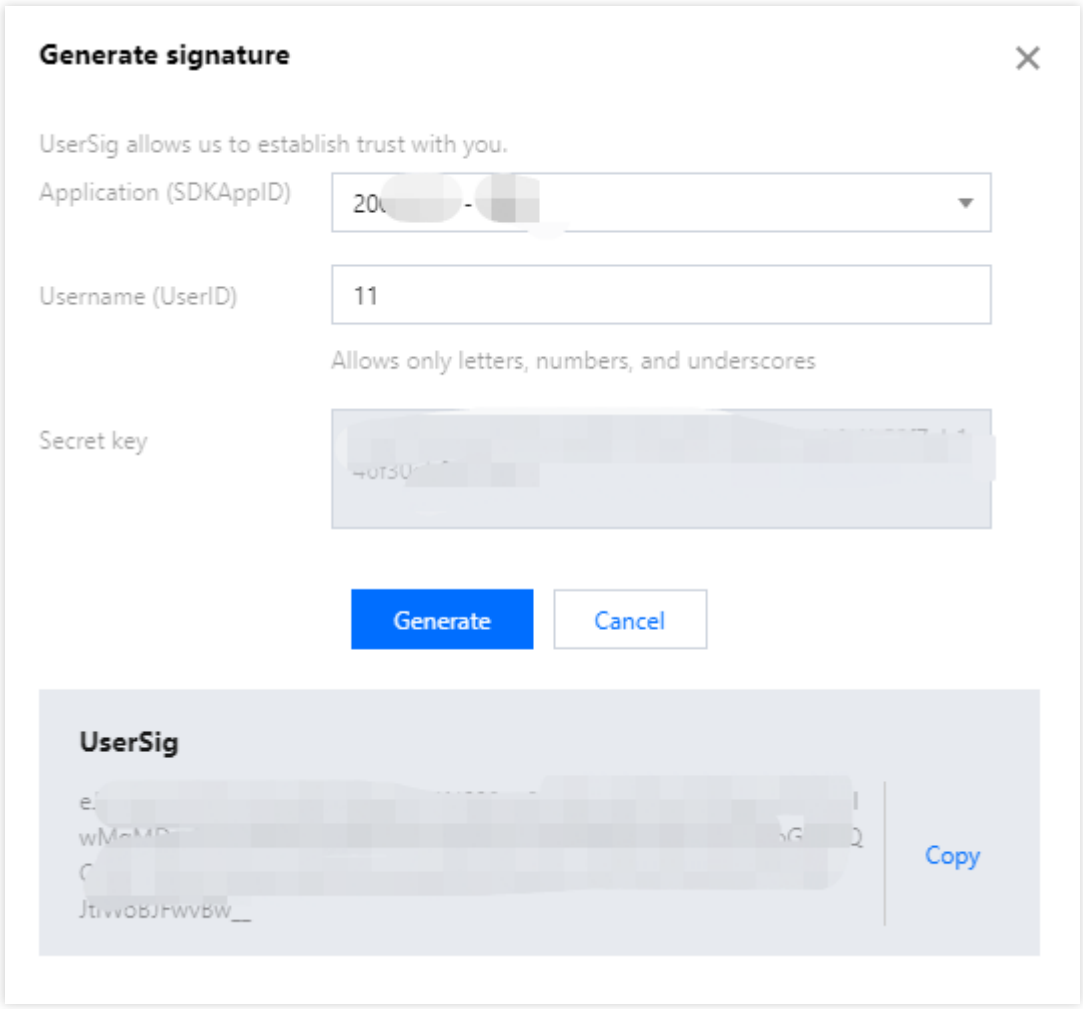
1. **创建实时音视频 TRTC 应用**
   - 如果您还没有腾讯云账号，请 注册腾讯云账号 ，并完成 实名认证。
   - 在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。



2. **获取 TRTC 应用及密钥信息**
3. 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；



4. 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。



5. **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,可通过此模版 Github 生成 Electron + Vue3 + TS 的模板工程。

> 注意：
>
> - 本文档介绍的集成步骤基于 electron-vite-vue 模版工程1.0.0版本。
> - electron-vite-vue 模版工程最新版本目录结构有调整，如需使用最新版本，可参照本文档自行调整目录和配置。

2. 成功生成模板工程后，执行以下脚本：

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

3. 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.tencentcloud.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 tsignal
ing -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Pinia
createApp(App)
.use(createPinia())
.mount('#app')
.$nextTick(window.removeLoading)
```

3. 配置 element-plus 按需引入

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

> 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';
const path = require('path');
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'
```

4. **引入 trtc-electron-sdk**

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

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

5. **env.d.ts文件配置**

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

> 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;
}
```

6. **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

参数	类型	含义
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
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

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

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：2022-10-31 14:41:31

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

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

> 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）。

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。

API	描述
sendCustomMessage	发送自定义消息。

场控相关接口

API	描述
muteUserMicrophone	禁用/恢复某用户的麦克风。
muteAllUsersMicrophone	禁用/恢复所有用户的麦克风，并且状态会同步到房间信息中。
muteUserCamera	禁用/恢复某用户的摄像头。
muteAllUsersCamera	禁用/恢复所有用户的摄像头，并且状态会同步到房间信息中。
muteChatRoom	开启/停止聊天室禁言（主持人调用）。
kickOffUser	移除房间内的某人（主持人调用）。
startCallingRoll	主持人开始点名。
stopCallingRoll	主持人结束点名。
replyCallingRoll	参会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	参会成员同意/拒绝主持人的申请发言。
sendSpeechApplication	参会成员申请发言。
replySpeechApplication	主持人同意/拒绝参会成员的申请发言。
forbidSpeechApplication	主持人禁止申请发言。
sendOffSpeaker	主持人令参会成员停止发言。
sendOffAllSpeakers	主持人令全体停止发言。
exitSpeechState	参会成员停止发言，转变为观众。

屏幕分享接口

API	描述

API	描述
startScreenCapture	启动屏幕分享。
stopScreenCapture	停止屏幕采集。

美颜滤镜相关接口函数

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

相关设置接口

API	描述
setVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
getSDKVersion	获取 SDK 版本。

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。
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	参会成员被请求停止发言的回调。
onCallingRollStarted	主持人开始点名，参会成员收到的回调。
onCallingRollStopped	主持人结束点名，参会成员收到的回调。
onMemberReplyCallingRoll	参会成员回复点名，主持人收到的回调。

API	描述
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	参会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（参会成员调用）。

```
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 roomId 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）。

```
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.Action
Callback 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 callbac
k);
```

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

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
isFront	boolean	true：前置摄像头；false：后置摄像头。

发送消息接口

sendChatMessage

在房间中广播文本消息，一般用于文本聊天。

```
void sendChatMessage(String message, TUIRoomCoreCallback.ActionCallback callback)
;
```

参数如下表所示：

参数	类型	含义
message	String	消息内容。
callback	TUIRoomCoreCallback.ActionCallback	发送结果回调。

sendCustomMessage

发送自定义消息。

```
void sendCustomMessage(String data, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
data	String	消息内容。
callback	TUIRoomCoreCallback.ActionCallback	发送结果回调。

场控相关接口

muteUserMicrophone

禁用/恢复某用户的麦克风。

```
void muteUserMicrophone(String userId, boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteAllUsersMicrophone

禁用/恢复所有用户的麦克风。

```
void muteAllUsersMicrophone(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteUserCamera

禁用/恢复某用户的摄像头。

```
void muteUserCamera(String userId, boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteAllUsersCamera

禁用/恢复所有用户的摄像头。

```
void muteAllUsersCamera(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

muteChatRoom

禁言/恢复文字聊天。

```
void muteChatRoom(boolean mute, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
mute	boolean	是否禁用。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

kickOffUser

主持人踢人。

```
void kickOffUser(String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

startCallingRoll

主持人开始点名。

```
void startCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

stopCallingRoll

主持人结束点名。

```
void stopCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

replyCallingRoll

参会成员回复主持人点名。

```
void replyCallingRoll(TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

sendSpeechInvitation

主持人邀请参会成员发言。

```
void sendSpeechInvitation(String userId, TUIRoomCoreCallback.InvitationCallback c
allback);
```

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

主持人取消邀请参会成员发言。

```
void cancelSpeechInvitation(String userId, TUIRoomCoreCallback.ActionCallback cal
lback);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

参数	类型	含义
callback	TUIRoomCoreCallback.ActionCallback	结果回调。

replySpeechInvitation

参会成员同意/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

参会成员申请发言。

```
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

参会成员取消申请发言。

```
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/拒绝参会成员的申请发言。

```
void replySpeechApplication(boolean agree, String userId, TUIRoomCoreCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
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

主持人令参会成员停止发言。

```
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

参会成员停止发言，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.TRTCScreenShareParams 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
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。

参数	类型	含义
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
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

参会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，参会成员收到的回调。

```
void onCallingRollStarted(String userId);
```

onCallingRollStopped

主持人结束点名，参会成员收到的回调。

```
void onCallingRollStopped(String userId);
```

onMemberReplyCallingRoll

参会成员回复点名，主持人收到的回调。

```
void onMemberReplyCallingRoll(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

onChatRoomMuted

主持人更改聊天室是否禁言回调。

```
void onChatRoomMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

onMicrophoneMuted

主持人设置禁用麦克风回调。

```
void onMicrophoneMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

onCameraMuted

主持人设置禁用摄像头回调。

```
void onCameraMuted(boolean muted);
```

参数如下表所示：

参数	类型	含义
muted	boolean	是否禁用。

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 (iOS)

最近更新时间：2022-11-01 10:17:52

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）。

API	描述
enterRoom	进入房间（参会成员调用）。
leaveRoom	离开房间（参会成员调用）。
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

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	参会成员回复主持人点名。
sendSpeechInvitation	主持人邀请参会成员发言。
cancelSpeechInvitation	主持人取消邀请参会成员发言。
replySpeechInvitation	参会成员同意/拒绝主持人的申请发言。
sendSpeechApplication	参会成员申请发言。
replySpeechApplication	主持人同意/拒绝参会成员的申请发言。
forbidSpeechApplication	主持人禁止申请发言。
sendOffSpeaker	主持人令参会成员停止发言。
sendOffAllSpeakers	主持人令全体停止发言。
exitSpeechState	参会成员停止发言，转变为观众。

屏幕分享接口

API	描述
startScreenCapture	启动屏幕分享。
stopScreenCapture	停止屏幕采集。

美颜滤镜相关接口函数

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

相关设置接口

API	描述
setVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
getSDKVersion	获取 SDK 版本。

TUIRoomCoreDelegate API 概览

错误事件回调

API	描述
onError	错误回调。

基础事件回调

API	描述
onDestroyRoom	房间解散回调。
onUserVoiceVolume	音量大小回调回调。
onRoomMasterChanged	主持人更改回调。

远端用户事件回调

API	描述
onRemoteUserEnter	远端用户进入房间回调。
onRemoteUserLeave	远端用户离开房间回调。

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。

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	参会成员被请求停止发言的回调。
onCallingRollStarted	主持人开始点名，参会成员收到的回调。
onCallingRollStopped	主持人结束点名，参会成员收到的回调。
onMemberReplyCallingRoll	参会成员回复点名，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	参会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。

```
+ (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（参会成员调用）。

```
- (void)enterRoom:(NSString *)roomId
callback:(TUIRoomActionCallback)callback;
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
roomId	NSString	房间标识。
callback	TUIRoomActionCallback	结果回调。

加入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 roomId 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）。

```
- (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
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

参会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

主持人邀请参会成员发言。

```
- (void)sendSpeechInvitation:(NSString *)userId
callback:(TUIRoomInviteeCallback)callback
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
callback	TUIRoomInviteeCallback	结果回调。

cancelSpeechInvitation

主持人取消邀请参会成员发言。

```
- (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

参会成员同意/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

参会成员申请发言。

```
- (void)sendSpeechApplication:(TUIRoomInviteeCallback)callback;
```

参数如下表所示：

参数	类型	含义
callback	TUIRoomInviteeCallback	结果回调。

cancelSpeechApplication

参会成员取消申请发言。

```
- (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/拒绝参会成员的申请发言。

```
- (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

主持人令参会成员停止发言。

```
- (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

参会成员停止发言，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

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。
currentUserId	NSString	更改后的主持人用户 ID。

远端用户回调事件

onRemoteUserEnter

远端用户进入房间回调。

```
- (void)onRemoteUserEnter:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserLeave

远端用户离开房间回调。

```
- (void)onRemoteUserLeave:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserCameraAvailable

远端用户是否开启摄像头视频。

```
- (void)onRemoteUserCameraAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	YES：有视频流数据；NO：无视频流数据。

onRemoteUserScreenVideoAvailable

成员**开启/关闭**视频分享的通知。

```
- (void)onRemoteUserScreenVideoAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	是否有屏幕分享流数据。

onRemoteUserAudioAvailable

远端用户是否开启音频上行回调。

```
- (void)onRemoteUserAudioAvailable:(NSString *)userId
available:(BOOL)available;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
available	BOOL	是否有音频数据。

onRemoteUserEnterSpeechState

远端用户开始发言。

```
- (void)onRemoteUserEnterSpeechState:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onRemoteUserExitSpeechState

远端用户结束发言。

```
- (void)onRemoteUserExitSpeechState:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

聊天室消息事件回调

onReceiveChatMessage

收到文本消息。

```
- (void)onReceiveChatMessage:(NSString *)userId message:(NSString *)message;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
message	NSString	文本消息。

场控消息回调

onReceiveSpeechInvitation

用户收到主持人发言邀请回调。

```
- (void)onReceiveSpeechInvitation:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

onReceiveInvitationCancelled

用户收到主持人取消发言邀请回调。

```
- (void)onReceiveInvitationCancelled:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

OnReceiveSpeechApplication

主持人收到用户发言申请的回调。

```
void onReceiveSpeechApplication(String userId);
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onSpeechApplicationCancelled

用户取消申请发言回调。

```
- (void)onSpeechApplicationCancelled:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onSpeechApplicationForbidden

主持人禁止申请发言回调。

```
- (void)onSpeechApplicationForbidden:(BOOL)isForbidden userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
isForbidden	BOOL	是否禁止。
userId	NSString	用户 ID。

onOrderedToExitSpeechState

参会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，参会成员收到的回调。

```
- (void)onCallingRollStarted:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人用户 ID。

onCallingRollStopped

主持人结束点名，参会成员收到的回调。

```
- (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

参会成员回复点名，主持人收到的回调。

```
- (void)onMemberReplyCallingRoll:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。

onChatRoomMuted

主持人更改聊天室是否禁言回调。

```
- (void)onChatRoomMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onMicrophoneMuted

主持人设置禁用麦克风回调。

```
- (void)onMicrophoneMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onCameraMuted

主持人设置禁用摄像头回调。

```
- (void)onCameraMuted:(BOOL)muted userId:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
muted	BOOL	是否禁用。
userId	NSString	主持人用户 ID。

onReceiveKickedOff

主持人踢人的回调。

```
- (void)onReceiveKickedOff:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	主持人/管理员 用户 ID。

统计和质量回调

onStatistics

技术指标统计回调。

```
- (void)onStatistics:(TRTCStatistics *)statistics;
```

参数如下表所示：

参数	类型	含义
statis	TRTCStatistics	统计数据。

onNetworkQuality

网络状况回调。

```
- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<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：2022-08-15 10:59:44

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。

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

API	描述
sendRoomTextMsg	在会议中广播文本消息，一般用于聊天。
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。

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

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。
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。
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
volume	int	采集音量，取值0 - 100，默认值为100。

setAudioPlayoutVolume

设置播放音量。

```
Future<void> setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，取值0 - 100，默认值100。

startAudioRecording

开始录音。

```
Future<int?> startAudioRecording(String filePath);
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
filePath	String	录音文件的保存路径，该路径需要用户自行指定，请确保路径存在且可写。该路径需精确到文件名及格式后缀，格式后缀决定录音文件的格式，目前支持的格式有 PCM、WAV 和 AAC。

stopAudioRecording

停止录音。

```
Future<void> stopAudioRecording();
```

enableAudioVolumeEvaluation

启用音量大小提示。

```
Future<void> enableAudioVolumeEvaluation(int intervalMs);
```

参数如下表所示：

参数	类型	含义
intervalMs	int	决定了 onUserVoiceVolume 回调的触发间隔，单位为ms，最小间隔为100ms，如果小于等于0则会关闭回调，建议设置为300ms。

录屏接口

startScreenCapture

启动屏幕分享。

```
Future<void> startScreenCapture({
int videoFps,
int videoBitrate,
int videoResolution,
int videoResolutionMode,
String appGroup,
});
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
videoFps	int	视频采集帧率。
videoBitrate	int	码率，SDK 会按照目标码率进行编码，只有在网络不佳的情况下才会主动降低视频码率。
videoResolution	int	视频分辨率。
videoResolutionMode	int	分辨率模式。
appGroup	String	该参数仅仅在iOS端有效，Android端不需要关注这个参数。该参数是主 App 与 Broadcast 共享的 Application Group Identifier。

说明：
详情请参见 TRTC SDK。

stopScreenCapture

停止屏幕采集。

```
Future<void> stopScreenCapture();
```

pauseScreenCapture

暂停屏幕采集。

```
Future<void> pauseScreenCapture();
```

resumeScreenCapture

恢复屏幕采集。

```
Future<void> resumeScreenCapture();
```

获取相关管理对象接口

getDeviceManager

获取设备管理对象 TXDeviceManager。

```
getDeviceManager();
```

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口

sendRoomTextMsg

在会议中广播文本消息，一般用于聊天。

```
Future<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。

参数	类型	含义
message	String	文本消息。

TRTCMeetingDelegate事件回调

通用事件回调

onError

错误回调。

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

参数如下表所示：

参数	类型	含义
errCode	int	错误码。
errMsg	String	错误信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onWarning

警告回调。

参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。
extraInfo	String	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。

onKickedOffline

其他用户登录了同一账号，被踢下线。

会议房间事件回调

onRoomDestroy

会议房间被销毁的回调。主持人退房时，房间内的所有用户都会收到此通知。

参数如下表所示：

参数	类型	含义
roomId	String	会议房间 ID。

onNetworkQuality

网络状态回调。

参数如下表所示：

参数	类型	含义
localQuality	TRTCCloudDef.TRTCQuality	上行网络质量。
remoteQuality	List<TRTCCloudDef.TRTCQuality>	下行网络质量。

onUserVolumeUpdate

用户通话音量回调。

参数如下表所示：

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

成员进出事件回调

onEnterRoom

本地进会回调。

参数如下表所示：

参数	类型	含义
result	int	大于0时为进会耗时（ms），小于0时为进会错误码。

onLeaveRoom

本地退会回调。

参数如下表所示：

参数	类型	含义
reason	int	离开会议原因，0：主动调用 leaveMeeting 退会；1：被服务器踢出当前会议；2：当前会议整个被解散。

onUserEnterRoom

新成员进会回调。

参数如下表所示：

参数	类型	含义
userId	String	新进会成员的用户 ID。

onUserLeaveRoom

成员退会回调。

参数如下表所示：

参数	类型	含义
userId	String	退会成员的用户 ID。

成员音视频事件回调

onUserAudioAvailable

成员开启/关闭麦克风的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开麦克风；false：用户关闭麦克风。

onUserVideoAvailable

成员开启/关闭摄像头的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开摄像头；false：用户关闭摄像头。

onUserSubStreamAvailable

成员开启/关闭辅路画面的回调。

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
available	boolean	true：用户打开辅路画面；false：用户关闭辅路画面。

消息事件回调

onRecvRoomTextMsg

收到文本消息的回调。

参数如下表所示：

参数	类型	含义
message	String	文本消息。
sendId	String	发送者用户 ID。
userAvatar	String	发送者用户头像。

参数	类型	含义
userName	String	发送者用户昵称。

onRecvRoomCustomMsg

收到自定义消息的回调。

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
sendId	String	发送者用户 ID。
userAvatar	String	发送者用户头像。
userName	String	发送者用户昵称。

录屏事件回调

onScreenCaptureStarted

录屏开始回调。

onScreenCapturePaused

录屏暂停回调。

onScreenCaptureResumed

录屏恢复回调。

onScreenCaptureStoped

录屏停止回调。

参数如下表所示：

参数	类型	含义
reason	int	停止原因，0：用户主动停止；1：屏幕窗口关闭导致停止。

TUIRoom(Windows&Mac)

最近更新时间：2022-10-31 14:41:31

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。

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（参会成员调用）。
LeaveRoom	离开房间（参会成员调用）。
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。
UpdateCameraPreview	改变本地视频渲染窗口。
StartLocalAudio	开启麦克风采集。
StopLocalAudio	停止麦克风采集。
StartSystemAudioLoopback	开启/关闭系统声音的采集。
StopSystemAudioLoopback	开启/关闭系统声音的采集。
SetVideoMirror	设置本地画面镜像预览模式。

远端用户相关接口

API	描述
StartRemoteView	订阅并播放指定成员的远端视频画面。
StopRemoteView	取消订阅并停止播放远端视频画面。
UpdateRemoteView	改变远端用户的视频渲染窗口。

发送聊天消息接口

API	描述
SendChatMessage	发送聊天消息。
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	主持人邀请参会成员发言。
CancelSpeechInvitation	主持人取消邀请参会成员发言。
ReplySpeechInvitation	参会成员同意/拒绝主持人的申请发言。
SendSpeechApplication	参会成员申请发言。
CancelSpeechApplication	参会成员取消申请发言。
ReplySpeechApplication	主持人同意/拒绝参会成员的申请发言。
ForbidSpeechApplication	主持人禁止申请发言。
SendOffSpeaker	主持人令参会成员停止发言。
SendOffAllSpeakers	主持人令全体停止发言。

API	描述
ExitSpeechState	参会成员停止发言，转变为观众。

基础组件接口函数

API	描述
GetDeviceManager	获取本地设置管理对象 ITXDeviceManager。
GetScreenShareManager	获取屏幕分享管理对象 IScreenShareManager。

云录制接口函数

API	描述
StartCloudRecord	开始云录制 。
StopCloudRecord	停止云录制 。

美颜相关接口函数

API	描述
SetBeautyStyle	美颜设置 。

相关设置接口

API	描述
SetVideoQosPreference	设置网络流控相关参数。

获取 SDK 版本接口函数

API	描述
GetSDKVersion	获取 SDK 版本。

TUIRoomCoreCallback API 概览

错误事件回调

API	描述
OnError	错误回调。

基础事件回调

API	描述
OnLogin	登录回调。
OnLogout	登出回调。
OnCreateRoom	创建房间回调。
OnDestroyRoom	房间解散回调。
OnEnterRoom	进入房间回调。
OnExitRoom	退出房间回调。
OnFirstVideoFrame	首帧画面回调。
OnUserVoiceVolume	音量大小回调回调。
OnRoomMasterChanged	主持人更改回调。

远端用户事件回调

API	描述
OnRemoteUserEnter	远端用户进入房间回调。
OnRemoteUserLeave	远端用户离开房间回调。
OnRemoteUserCameraAvailable	远端用户是否开启摄像头视频回调。
OnRemoteUserScreenAvailable	远端用户是否开启屏幕分享回调。
OnRemoteUserAudioAvailable	远端用户是否开启麦克风回调。
OnRemoteUserEnterSpeechState	远端用户开始发言回调。
OnRemoteUserExitSpeechState	远端用户结束发言回调。

消息事件回调

API	描述
OnReceiveChatMessage	收到文本消息回调。
OnReceiveCustomMessage	收到文本消息回调。

场控事件回调

API	描述
OnReceiveSpeechInvitation	用户收到主持人发言邀请回调。
OnReceiveInvitationCancelled	用户收到主持人取消发言邀请回调。
OnReceiveReplyToSpeechInvitation	主持人收到用户同意邀请发言的回调。
OnReceiveSpeechApplication	主持人收到用户发言申请的回调。
OnSpeechApplicationCancelled	用户取消申请发言回调。
OnReceiveReplyToSpeechApplication	主持人同意发言申请回调。
OnSpeechApplicationForbidden	主持人禁止申请发言回调。
OnOrderedToExitSpeechState	参会成员被请求停止发言的回调。
OnCallingRollStarted	主持人开始点名，参会成员收到的回调。
OnCallingRollStopped	主持人结束点名，参会成员收到的回调。
OnMemberReplyCallingRoll	参会成员回复点名，主持人收到的回调。
OnChatRoomMuted	主持人更改聊天室是否禁言回调。
OnMicrophoneMuted	主持人设置禁用麦克风回调。
OnCameraMuted	主持人设置禁用摄像头回调。

统计和质量回调

API	描述
OnStatistics	技术指标统计回调。
OnNetworkQuality	网络质量回调。

屏幕分享相关回调

API	描述
OnScreenCaptureStarted	开始屏幕分享回调。
OnScreenCaptureStopped	停止屏幕分享回调。

视频录制回调

API	描述
OnRecordError	录制错误回调。
OnRecordComplete	录制完成回调。
OnRecordProgress	录制进度回调。

本地设备测试回调

API	描述
OnTestSpeakerVolume	扬声器大小回调。
OnTestMicrophoneVolume	麦克风大小回调。
OnAudioDeviceCaptureVolumeChanged	调节系统采集音量回调。
OnAudioDevicePlayoutVolumeChanged	调节系统播放音量回调。

TUIRoomCore 基础函数

GetInstance

获取 TUIRoomCore 单例对象。

```
static TUIRoomCore* GetInstance();
```

DestroyInstance

```
static void DestroyInstance();
```

SetCallback

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

```
virtual void SetCallback(const TUIRoomCoreCallback* callback) = 0;
```

Login

登录。

```
virtual int Login(int sdk_appid, const std::string& user_id, const std::string& user_sig) = 0;
```

参数如下表所示：

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

Logout

登出。

```
virtual int Logout() = 0;
```

CreateRoom

创建房间（主持人调用）。

```
virtual int CreateRoom(const std::string& room_id, TUISpeechMode speech_mode) = 0;
```

参数如下表所示：

参数	类型	含义
room_id	string	房间标识，需要由您分配并进行统一管理。
speech_mode	TUISpeechMode	发言模式。

主持人正常调用流程如下：

1. **主持人**调用 `CreateRoom()` 创建房间，房间创建成功与否会通过 OnCreateRoom 通知给主持人。

2. **主持人**调用 `EnterRoom()` 进入房间。
3. **主持人**调用 `StartCameraPreview()` 打开摄像头采集和预览。
4. **主持人**调用 `StartLocalAudio()` 打开本地麦克风。

DestroyRoom

销毁房间房间（主持人调用）。主持人在创建房间后，可以调用该函数来销毁房间。

```
virtual int DestroyRoom() = 0;
```

EnterRoom

进入房间（参会成员调用）。

```
virtual int EnterRoom(const std::string& room_id) = 0;
```

参数如下表所示：

参数	类型	含义
room_id	string	房间标识。

参会成员进入房间的正常调用流程如下：

1. **参会成员**调用 `EnterRoom` 并传入 room_id 即可进入房间房间。
2. **参会成员**调用 `startCameraPreview()` 打开摄像头预览，调用 `StartLocalAudio()` 打开麦克风采集。
3. **参会成员**收到 `OnRemoteUserCameraAvailable` 的事件，调用 `StartRemoteView()` 开始播放视频。

LeaveRoom

离开房间（参会成员调用）。

```
virtual int LeaveRoom() = 0;
```

GetRoomInfo

获取房间信息。

```
virtual TUIRoomInfo GetRoomInfo() = 0;
```

GetRoomUsers

获取房间所有成员信息。

```
virtual std::vector<TUIUserInfo> GetRoomUsers() = 0;
```

GetUserInfo

获取成员信息。

```
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& avata
r_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& vie
w,
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, lite
av::TXView& view) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
type	TUIStreamType	流类型。
view	liteav::TXView	渲染窗口句柄。

发送消息接口

SendChatMessage

发送文本消息。

```
virtual int SendChatMessage(const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
message	string	消息内容。

SendCustomMessage

发送自定义消息。

```
virtual int SendCustomMessage(const std::string& message) = 0;
```

参数如下表所示：

参数	类型	含义
message	string	消息内容。

场控相关接口

MuteUserMicrophone

禁用/恢复某用户的麦克风。

```
virtual int MuteUserMicrophone(const std::string& user_id, bool mute, Callback ca
llback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
mute	bool	是否禁用。
callback	Callback	接口回调。

MuteAllUsersMicrophone

禁用/恢复所有用户的麦克风。

```
virtual int MuteAllUsersMicrophone(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

MuteUserCamera

禁用/恢复某用户的摄像头。

```
virtual int MuteUserCamera(const std::string& user_id, bool mute, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
mute	bool	是否禁用。
callback	Callback	接口回调。

MuteAllUsersCamera

禁用/恢复所有用户的摄像头。

```
virtual int MuteAllUsersCamera(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

MuteChatRoom

开启/停止聊天室禁言。

```
virtual int MuteChatRoom(bool mute) = 0;
```

参数如下表所示：

参数	类型	含义
mute	bool	是否禁用。

KickOffUser

主持人踢人。

```
virtual int KickOffUser(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
callback	Callback	接口回调。

StartCallingRoll

主持人开始点名。

```
virtual int StartCallingRoll() = 0;
```

StopCallingRoll

主持人结束点名。

```
virtual int StopCallingRoll() = 0;
```

ReplyCallingRoll

参会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

主持人邀请参会成员发言。

```
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

主持人取消邀请参会成员发言。

```
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

参会成员同意/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

参会成员申请发言。

```
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

参会成员取消申请发言。

```
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/拒绝参会成员的申请发言。

```
virtual int ReplySpeechApplication(const std::string& user_id, bool agree, Callba
ck 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

参数	类型	含义
forbid	bool	是否禁止。

SendOffSpeaker

主持人令参会成员停止发言。

```
virtual int SendOffSpeaker(const std::string& user_id, Callback callback) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
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

参会成员停止发言，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。

参数	类型	含义
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 st
ream_type) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
stream_type	TUIStreamType	流类型。

OnUserVoiceVolume

用户音量大小回调。

```
virtual void OnUserVoiceVolume(const std::string& user_id, int volume)
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
volume	int	用户的音量大小，取值范围 0 - 100。

OnRoomMasterChanged

主持人更改回调。

```
virtual void OnRoomMasterChanged(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

远端用户回调事件

OnRemoteUserEnter

远端用户进入房间回调。

```
virtual void OnRemoteUserEnter(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnRemoteUserLeave

远端用户离开房间回调。

```
virtual void OnRemoteUserLeave(const std::string& user_id) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。

OnRemoteUserCameraAvailable

远端用户是否开启摄像头视频。

```
virtual void OnRemoteUserCameraAvailable(const std::string& user_id, bool availab
le) = 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 availab
le) = 0;
```

参数如下表所示：

参数	类型	含义
user_id	string	用户 ID。
available	bool	true：有视频流数据；false：无视频流数据。

OnRemoteUserAudioAvailable

远端用户是否开启麦克风。

```
virtual void OnRemoteUserAudioAvailable(const std::string& user_id, bool availabl
e) = 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

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

参会成员被请求停止发言的回调。

```
virtual void OnOrderedToExitSpeechState() = 0;
```

OnCallingRollStarted

主持人开始点名，成员收到的回调。

```
virtual void OnCallingRollStarted() = 0;
```

OnCallingRollStopped

主持人结束点名，参会成员收到的回调。

```
virtual void OnCallingRollStopped() = 0;
```

OnMemberReplyCallingRoll

参会成员回复点名，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, litea
v::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& mes
sgae) {}
```

参数如下表所示：

参数	类型	含义
error	TXLiteAVLocalRecordError	错误信息。
messgae	string	错误描述。

OnRecordComplete

录制完成回调。

```
virtual void OnRecordComplete(const std::string& path) {}
```

参数如下表所示：

参数	类型	含义
path	string	错误描述。

OnRecordProgress

录制进度回调。

```
virtual void OnRecordProgress(int duration, int file_size) {}
```

参数如下表所示：

参数	类型	含义
duration	int	文件的时长。
file_size	int	文件的大小。

本地设备测试回调

OnTestSpeakerVolume

扬声器音量大小回调。

```
virtual void OnTestSpeakerVolume(uint32_t volume) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。

OnTestMicrophoneVolume

麦克风音量大小回调。

```
virtual void OnTestMicrophoneVolume(uint32_t volume) {}
```

参数如下表所示：

参数	类型	含义
volume	uint32_t	音量大小。

OnAudioDeviceCaptureVolumeChanged

调节系统采集音量回调。

```
virtual void OnAudioDeviceCaptureVolumeChanged(uint32_t volume, bool muted) {}
```

参数如下表所示：

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

OnAudioDevicePlayoutVolumeChanged

调节系统播放音量回调。

```
virtual void OnAudioDevicePlayoutVolumeChanged(uint32_t volume, bool muted) {}
```

参数如下表所示：

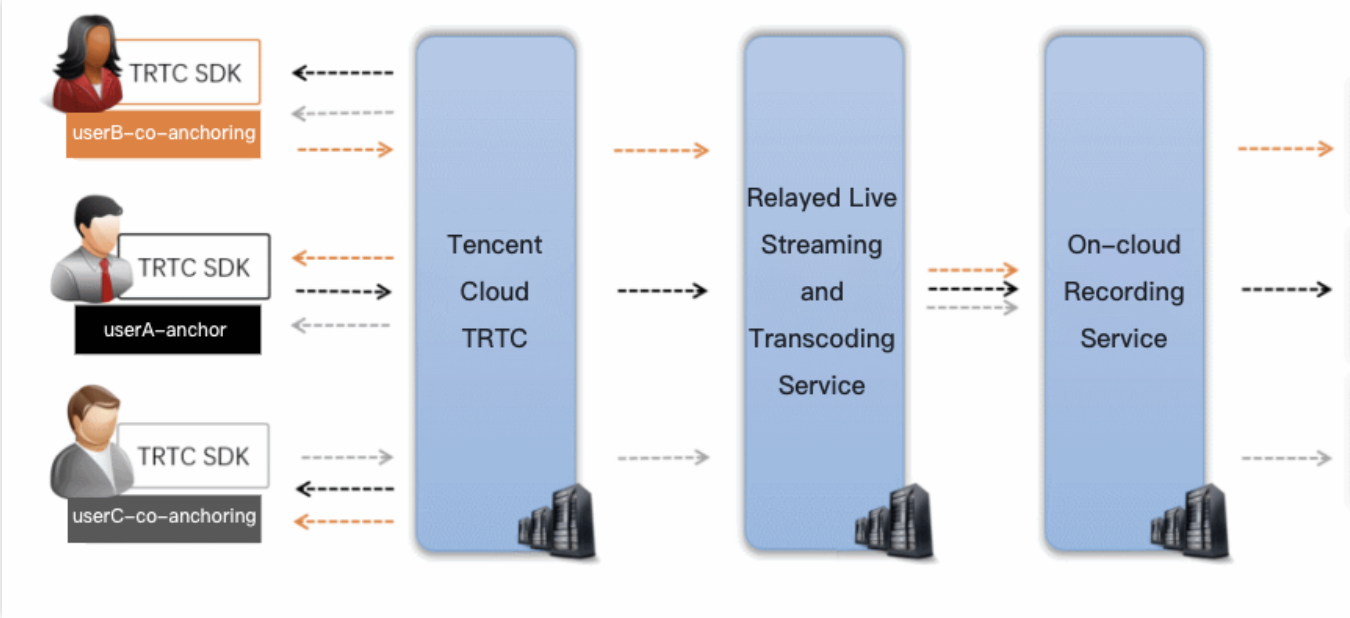
参数	类型	含义
volume	uint32_t	音量大小。
muted	bool	是否被禁用。

实现云端录制与回放（旧）

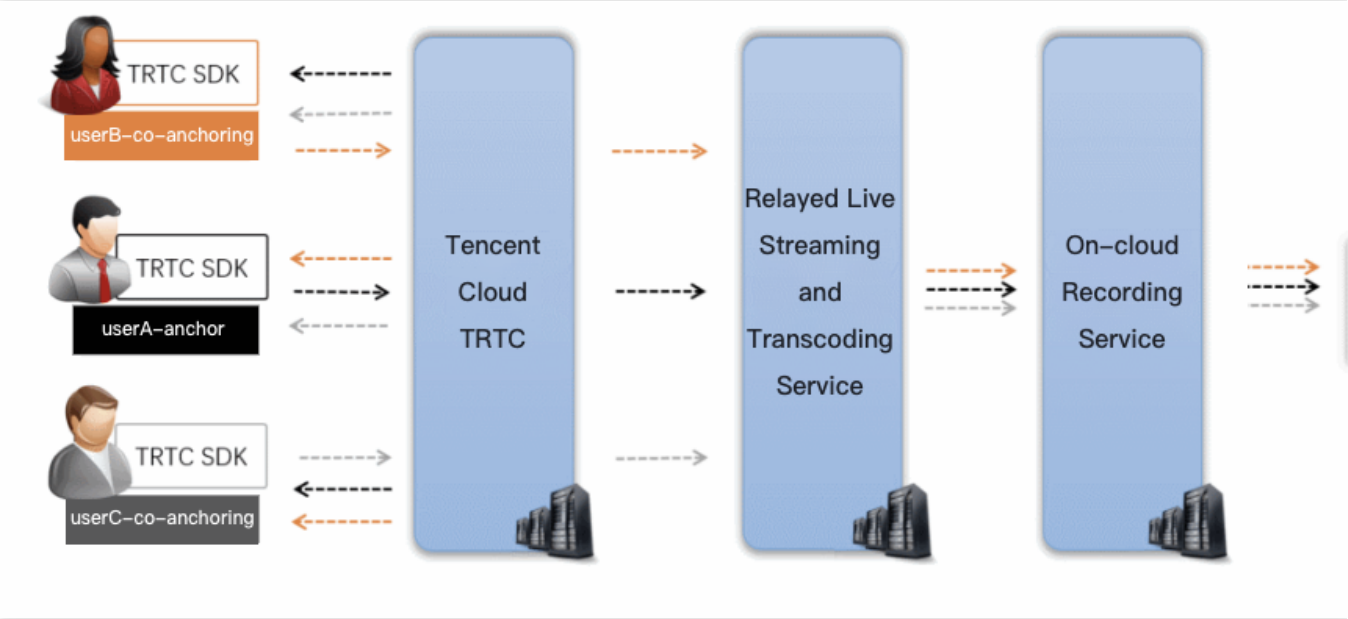
最近更新时间：2023-05-12 17:43:06

适用场景

在远程教育、秀场直播、视频会议、远程定损、金融双录、在线医疗等应用场景中，考虑取证、质检、审核、存档和回放等需求，常需要将整个视频通话或互动直播过程录制和存储下来。
TRTC 的云端录制，可以将房间中的每一个用户的音视频流都录制成一个独立的文件



也可以将房间中的多路音视频先进行 云端混流，再将混合后的音视频流录制成一个文件：



注意：

应用的 SDKAppID 为140 开头用户可参考本篇云端录制与回放操作指引接入使用。若您应用的 SDKAppID 为 200 开头，请参考新版云端录制发起云端录制功能。

控制台指引

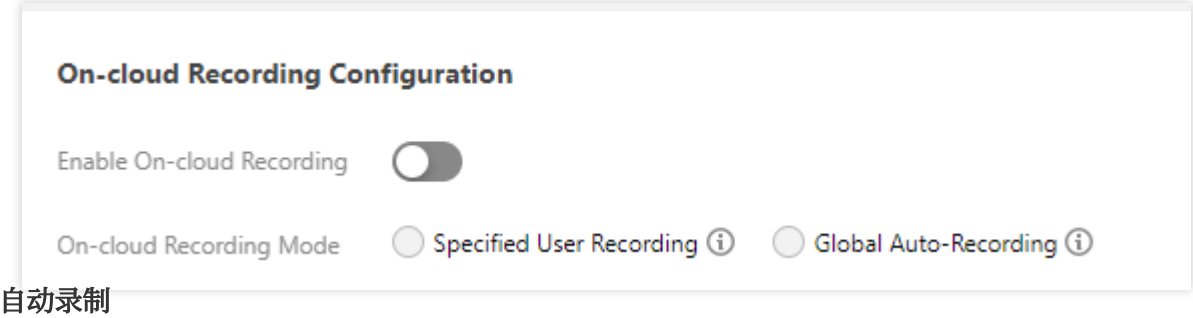
开通录制服务

1. 登录实时音视频控制台，在左侧导航栏选择 应用管理。
2. 单击目标应用所在行的**功能配置**，进入功能配置页卡。如果您还没有创建过应用，可以单击**创建应用**，填写应用名称，单击**确定**创建一个新的应用。
3. 单击**启用云端录制**右侧的

，会弹出云端录制的设置页面。

选择录制形式

TRTC 的云端录制服务提供了两种不同的录制形式：“全局自动录制”和“指定用户录制”：



全局自动录制

每一个 TRTC 房间中的每个用户的音视频上行流都会被自动录制下来，录制任务的启动和停止都是自动的，不需要您额外操心，比较简单和易用。具体的使用方法请阅读 方案一：全局自动录制。

指定用户录制

您可以指定只录制一部分用户的音视频流，这需要您通过客户端的 SDK API 或者服务端的 REST API 进行控制，需要额外的开发工作量。具体的使用方法请阅读 方案二：指定用户录制（SDK API） 和 方案三：指定用户录制（REST API）。

选择文件格式

云端录制支持 HLS、MP4、FLV 和 AAC 四种不同的文件格式，我们以表格的形式列出四种不同格式的差异和适用场景，您可以结合自身业务的需要进行选择：

参数	参数说明
文件类型	支持以下文件类型： HLS：该文件类型支持绝大多数浏览器在线播放，适合视频回放场景。选择该文件类型时，支持断点续录且不限制单个文件最大时长。 FLV：该文件类型不支持在浏览器在线播放，但该格式简单容错性好。如果无需将录制文件存储在云点播平台，可以选择该文件类型，录制完成后立刻下载录制文件并删除源文件。 MP4：该文件类型支持在 Web 浏览器在线播放，但此格式容错率差，视频通话过程中的任何丢包都会影响最终文件的播放质量。 AAC：如果只需录制音频，可以选择该文件类型。
单个文件的最大时长（分钟）	根据实际业务需求设置单个视频文件的最大时长限制，超过长度限制后系统将会自动拆分视频文件。单位为分钟，取值范围1 - 120。 当文件类型设置为HLS时，不限制单个文件的最大时长，即该参数无效。
文件保存时长（天）	根据实际业务需求设置视频文件存储在云点播平台的天数。单位为天，取值范围0 - 1500，到期后文件将被点播平台自动删除且无法找回， 0表示永久存储。
续录超时时长（秒）	默认情况下，若通话（或直播）过程因网络波动或其他原因被打断，录制文件会被切断成多个文件。 如果需要实现“一次通话（或直播）只产生一个回放链接”，可以根据实际情况设置续录超时时长，当打断间隔不超过设定的续录超时时长时，一次通话（或直播）只会生成一个文件，但需要等待续录时间超时后才能收到录制文件。 单位为秒，取值范围1 - 1800，0表示断点后不续录。

说明

HLS 支持最长三十分钟的续录，可以做到“一堂课只产生一个回放链接”，且支持绝大多数浏览器的在线观看，非常适合在线教育场景中的视频回放场景。

选择存储位置

TRTC 云端录制文件会默认存储于腾讯云点播服务上，如果您的项目中多个业务公用一个腾讯云点播账号，可能会有录制文件隔离的需求。您可以通过腾讯云点播的“子应用”能力，将 TRTC 的录制文件与其他业务区分开。

什么是点播主应用和子应用？

主应用和子应用是云点播上的一种资源划分的方式。主应用相当于云点播的主账号，子应用可以创建多个，每一个子应用相当于主账号下面的一个子账号，拥有独立的资源管理，存储区域可以跟其他的子应用相互隔离。

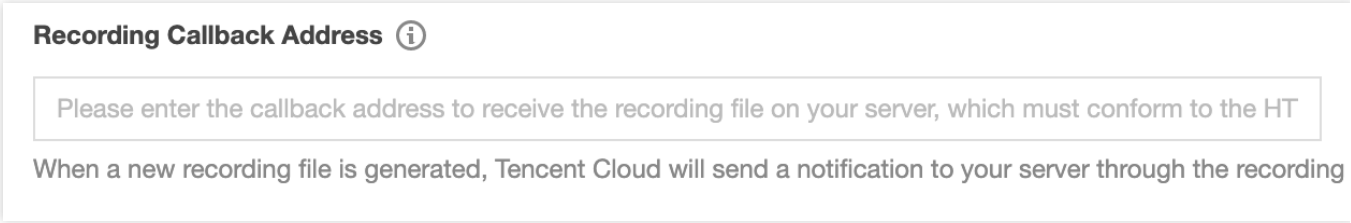
如何开启点播服务的子应用？

您可以根据文档 “如何开启点播子应用” 添加新的子应用，这一步需要您跳转到 点播控制台 中进行操作。

设置录制回调

录制回调地址：

如果您需要实时接收到新文件的 落地通知，可在此处填写您的服务器上用于接收录制文件的回调地址，该地址需符合 HTTP（或 HTTPS）协议。当新的录制文件生成后，腾讯云会通过该地址向您的服务器发送通知。



录制回调密钥：

回调密钥用于在接收回调事件时生成签名鉴权，该密钥需由大小写字母及数字组成，且不得超过32个字符。相关使用请参见 录制事件参数说明。

说明

详细的录制回调接收和解读方案请参考文档后半部分的：接收录制文件。

录制控制方案

TRTC 提供了三种云端录制的控制方案，分别是 全局自动录制、指定用户录制（由 SDK API 控制） 和 指定用户录制（由 REST API 控制）。对于其中的每一种方案，我们都会详细介绍：

如何在控制台中设定使用该方案？

如何开始录制任务？

如何结束录制任务？

如何将房间中的多路画面混合成一路？

录制下来的文件会以什么格式命名？
支持该种方案的平台有哪些？

方案一：全局自动录制

控制台中的设定

要使用该种录制方案，请在控制台中 选择录制形式 时，设定为“全局自动录制”。

录制任务的开始

TRTC 房间中的每一个用户的音视频流都会被自动录制成文件，无需您的额外操作。

录制任务的结束

自动停止。即每个主播在停止音视频上行后，该主播的云端录制即会自行停止。如果您在 选择文件格式 时设置了“续录时间”，则需要等待续录时间超时后才能收到录制文件。

多路画面的混合

全局自动录制模式下的云端混流有两种方案，即“服务端 REST API 方案” 和 “客户端 SDK API 方案”，两套方案请勿混合使用：

服务端 REST API 混流方案：需要由您的服务器发起 API 调用，不受客户端平台版本的限制。

客户端 SDK API 混流方案：可以直接在客户端发起混流，目前支持 iOS、Android、Windows、Mac 、Web 等平台，暂不支持微信小程序。

录制文件的命名

如果主播在进房时指定了 userDefineRecordId 参数，则录制文件会以 `userDefineRecordId_streamType_开始时间_结束时间` 来命名（streamType 有 main 和 aux 两个取值，main 代表主路，aux 代表辅路，辅路通常被用作屏幕分享）；

如果主播在进房时没有指定 userDefineRecordId 参数，但指定了 streamId 参数，则录制文件会以 `streamId_开始时间_结束时间` 来命名；

如果主播在进房时既没有指定 userDefineRecordId 参数，也没有指定 streamId 参数，则录制文件会以 `sdkappid_roomid_userid_streamType_开始时间_结束时间` 来命名（streamType 有 main 和 aux 两个取值，main 代表主路，aux 代表辅路，辅路通常被用作屏幕分享）。

已经支持的平台

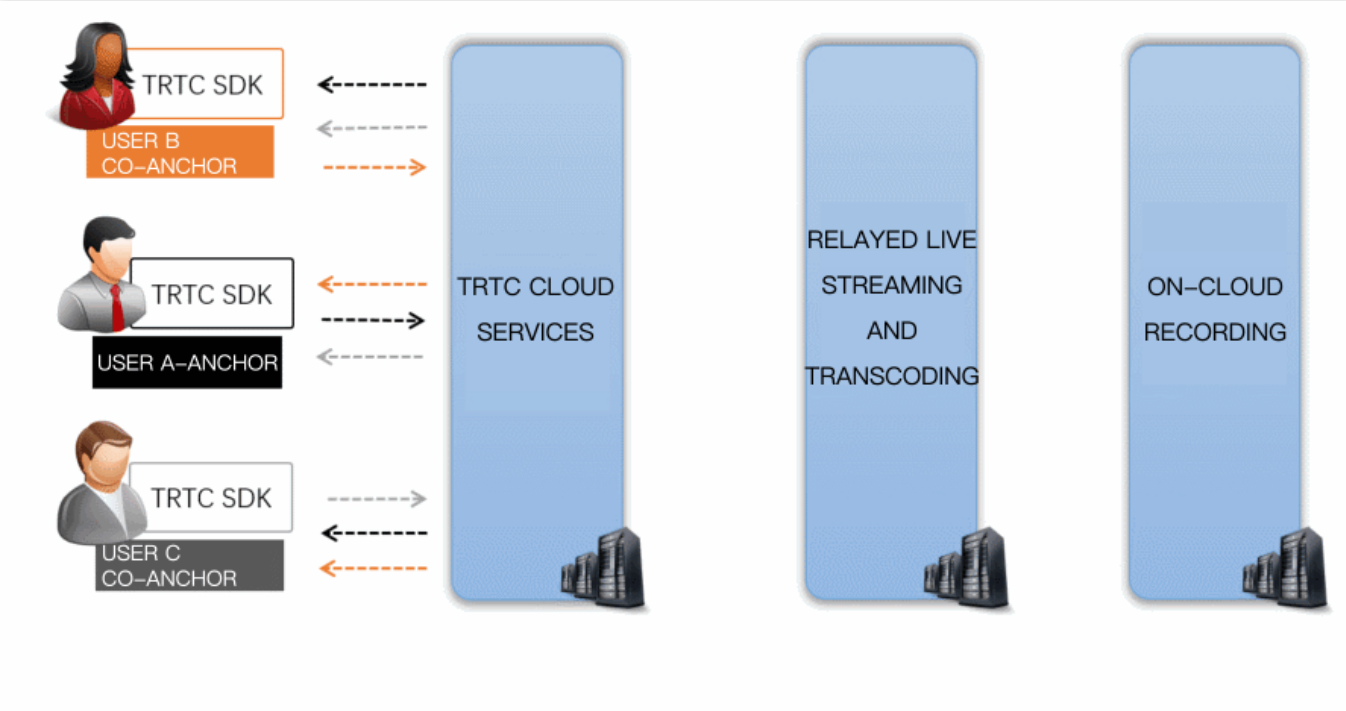
由您的服务端控制，不受客户端平台限制。

方案二：指定用户录制（SDK API）

通过调用 TRTC SDK 提供的一些 API 接口和参数，即可实现云端混流、云端录制和旁路直播三个功能：

云端能力	如何开始？	如何停止？
云端录制	进房时指定参数 `TRTCParams` 中的 `userDefineRecordId` 字段	主播退房时自动停止
云端混流	调用 SDK API setMixTranscodingConfig() 启动云端混流	发起混流的主播退房后，混流会自动停止，或中途调用 setMixTranscodingConfig() 并将参数设置

		为 `null/nil` 手动停止
旁路直播	进房时指定参数 `TRTCParams` 中的 `streamId` 字段	主播退房时自动停止

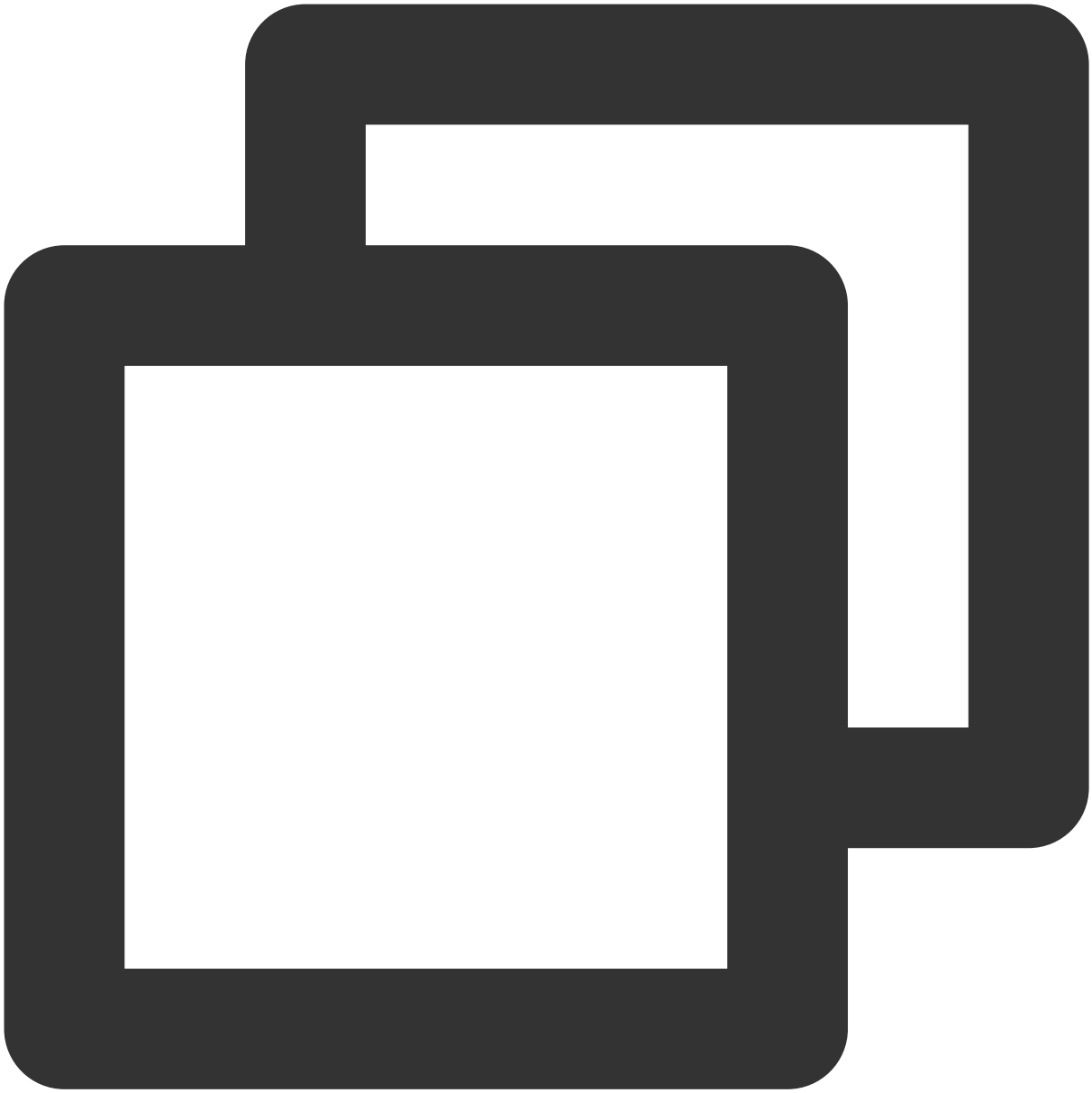


控制台中的设定

要使用该种录制方案，请在控制台中 选择录制形式 时，设定为“指定用户录制”。

录制任务的开始

主播在进房时指定进房参数 TRTCParams 中的 userDefineRecordId 字段，之后该主播的上行音视频数据即会被云端录制下来，不指定该参数的主播不会触发录制任务。



```
// 示例代码：指定录制用户 rexchang 的音视频流，文件 id 为 1001_rexchang
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
TRTCParams *param = [[TRTCParams alloc] init];
param.sdkAppId = 1400000123;      // TRTC 的 SDKAppID，创建应用后可获得
param.roomId   = 1001;            // 房间号
param.userId   = @"rexchang";     // 用户名
param.userSig  = @"xxxxxxxx";     // 登录签名
param.role     = TRTCRoleAnchor; // 角色：主播
param.userDefineRecordId = @"1001_rexchang";  // 录制 ID，即指定开启该用户的录制。
[trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE]; // 请使用 LIVE 模式
```

录制任务的结束

自动停止，当进房时指定 userDefineRecordId 参数的主播在停止音视频上行后，云端录制会自行停止。如果您在 选择文件格式 时设置了“续录时间”，则需要等待续录时间超时后才能收到录制文件。

多路画面的混合

您可以通过调用 SDK API setMixTranscodingConfig() 将房间中其它用户的画面和声音混合到当前用户的这一路音视频流上。关于这一部分详细介绍，可以阅读文档：云端混流转码。

注意

在一个 TRTC 房间中，只由一个主播（推荐是开播的主播）来调用 `setMixTranscodingConfig` 即可，多个主播调用可能会出现状态混乱的错误。

录制文件的命名

录制文件会以 `userDefineRecordId_开始时间_结束时间` 的格式来命名。

已经支持的平台

支持 iOS、Android、Windows、Mac、Electron、Web 等终端发起录制控制，暂不支持微信小程序端发起控制。

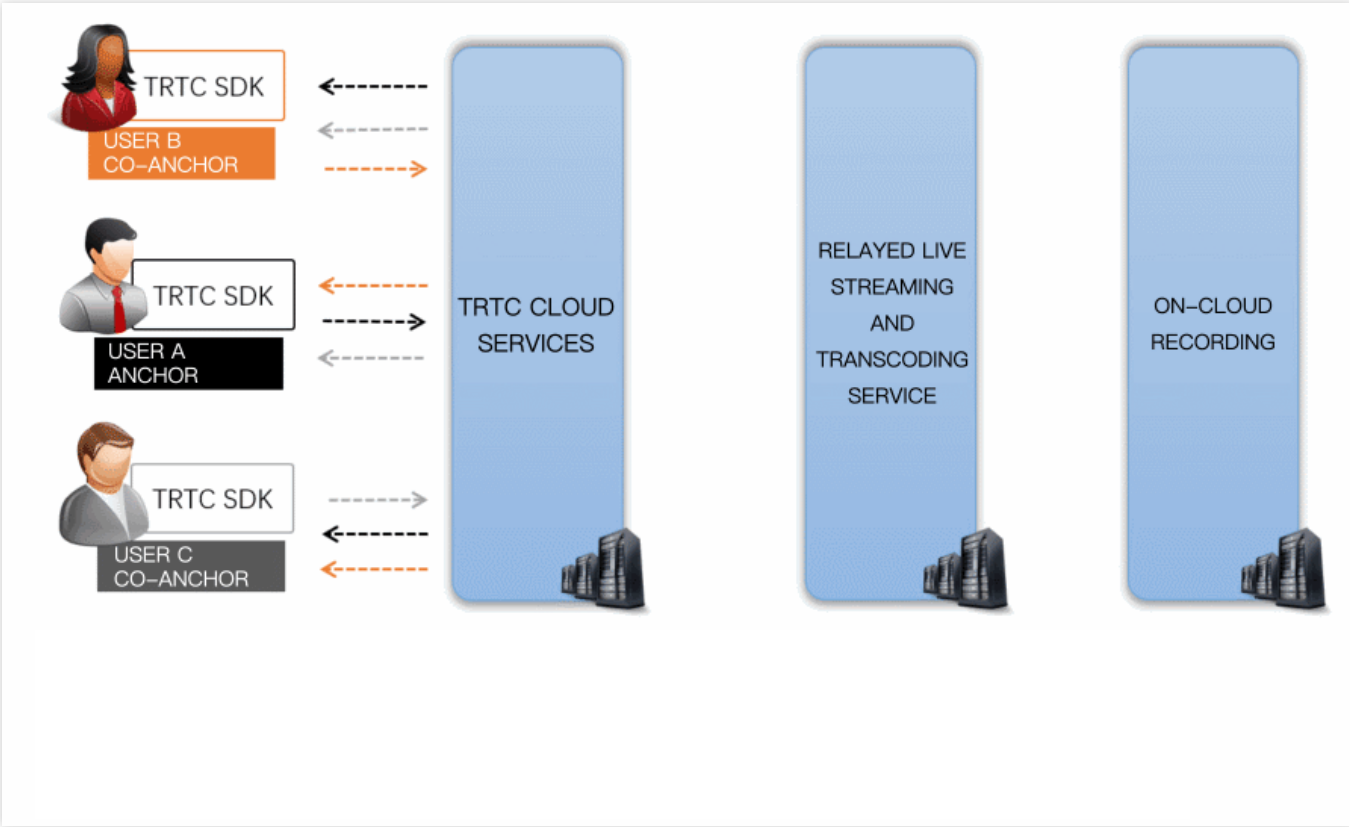
方案三：指定用户录制（REST API）

TRTC 的服务端提供了一对 REST API（ StartMCUMixTranscode 和 StopMCUMixTranscode）用于实现云端混流、云端录制和旁路直播三个功能：

云端能力	如何开始？	如何停止？
云端录制	调用 StartMCUMixTranscode 时指定 `OutputParams.RecordId` 参数即可开始录制	自动停止，或中途调用 StopMCUMixTranscode 停止
云端混流	调用 StartMCUMixTranscode 时指定 `LayoutParams` 参数可设置布局模板和布局参数	所有用户退房后自动停止，或中途调用 StopMCUMixTranscode 手动停止
旁路直播	调用 StartMCUMixTranscode 时指定 `OutputParams.StreamId` 参数可启动到 CDN 的旁路直播	自动停止，或中途调用 StopMCUMixTranscode 停止

说明

由于这对 REST API 控制的是 TRTC 云服务中的核心混流模块 MCU，并将 MCU 混流后的结果输送给录制系统和直播 CDN，因此 API 的名字被称为 `Start/StopMCUMixTranscode` 。因此，从功能角度上来说， `Start/StopMCUMixTranscode` 不仅仅可以实现混流的功能，也可以实现云端录制和旁路直播 CDN 的功能。

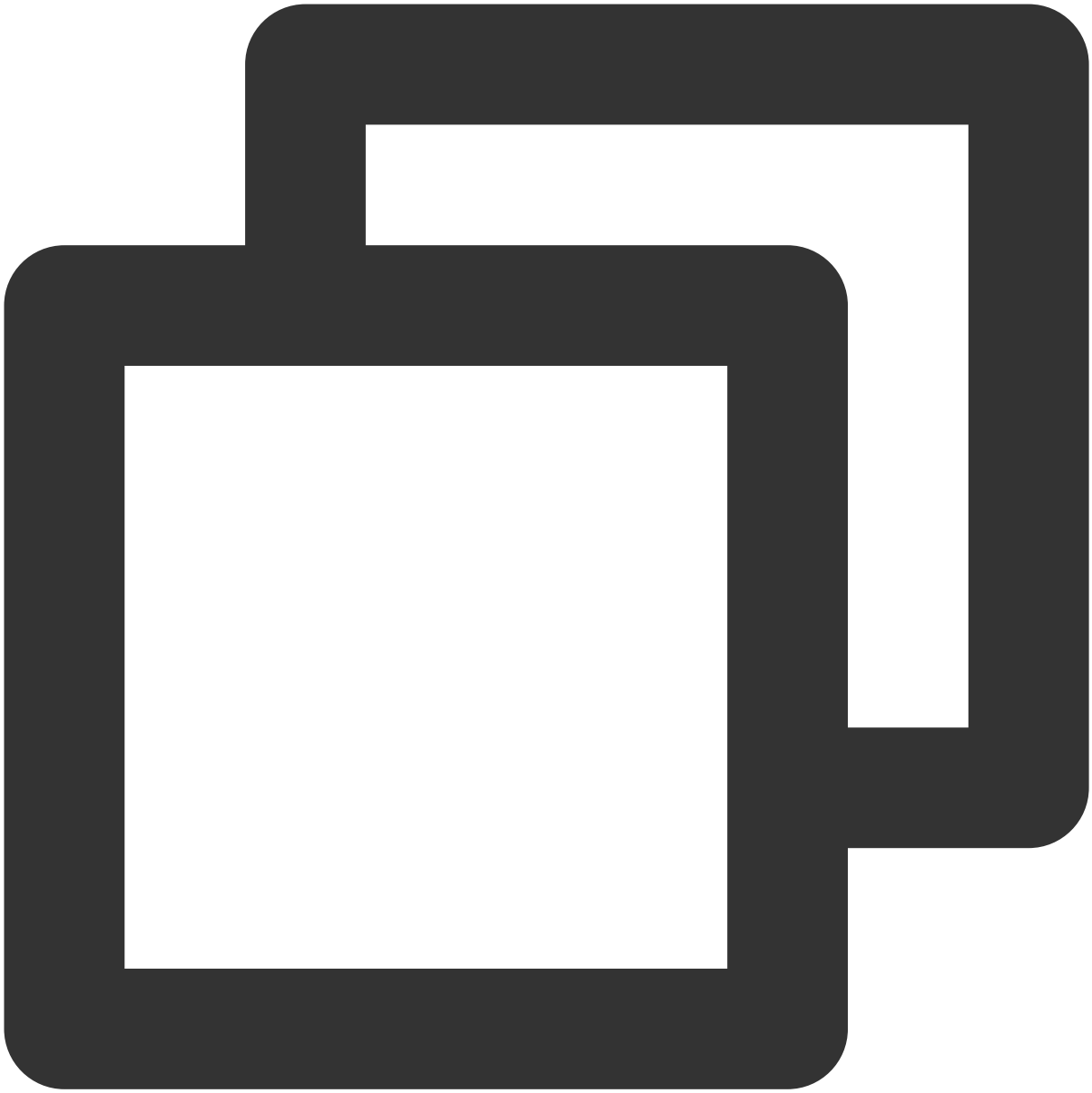


控制台中的设定

要使用该种录制方案，请在控制台中 选择录制形式 时，设定为“指定用户录制”。

录制任务的开始

由您的服务器调用 StartMCUMixTranscode ，并指定 `OutputParams.RecordId` 参数即可启动混流和录制。



```
// 代码示例：通过 REST API 启动云端混流和云端录制任务
https://trtc.tencentcloudapi.com/?Action=StartMCUMixTranscode
&SdkAppId=1400000123
&RoomId=1001
&OutputParams.RecordId=1400000123_room1001
&OutputParams.RecordAudioOnly=0
&EncodeParams.VideoWidth=1280
&EncodeParams.VideoHeight=720
&EncodeParams.VideoBitrate=1560
&EncodeParams.VideoFramerate=15
&EncodeParams.VideoGop=3
```

```
&EncodeParams.BackgroundColor=0
&EncodeParams.AudioSampleRate=48000
&EncodeParams.AudioBitrate=64
&EncodeParams.AudioChannels=2
&LayoutParams.Template=1
&<公共请求参数>
```

注意

该 REST API 需要在房间中至少有一个用户进房（enterRoom）成功后才有效。

使用 REST API 不支持单流录制，如果您需要单流录制，请选择 方案一 或者 方案二。

录制任务的结束

自动停止，您也可以中途调用 StopMCUMixTranscode 停止混流和录制任务。

多路画面的混合

在调用 StartMCUMixTranscode 时同时指定 `LayoutParams` 参数即可实现云端混流。该 API 支持在整个直播期间多次调用，即您可以根据需要修改 `LayoutParams` 参数并再次调用该 API 来调整混合画面的布局。但需要注意的是，您需要保持参数 `OutputParams.RecordId` 和 `OutputParams.StreamId` 在多次调用中的一致性，否则会导致断流并产生多个录制文件。

说明

关于云端混流的详细介绍，具体请参见 云端混流转码。

录制文件的命名

录制文件会以调用 StartMCUMixTranscode 时指定的 `OutputParams.RecordId` 参数来命名，命名格式为 `OutputParams.RecordId_开始时间_结束时间` 。

已经支持的平台

由您的服务端控制，不受客户端平台的限制。

查找录制文件

在开启录制功能以后，TRTC 系统中录制下来的文件就能在腾讯云点播服务中找到。您可以直接在云点播控制台手动查找，也可以由您的后台服务器使用 REST API 进行定时筛选：

方式一：在点播控制台手动查找

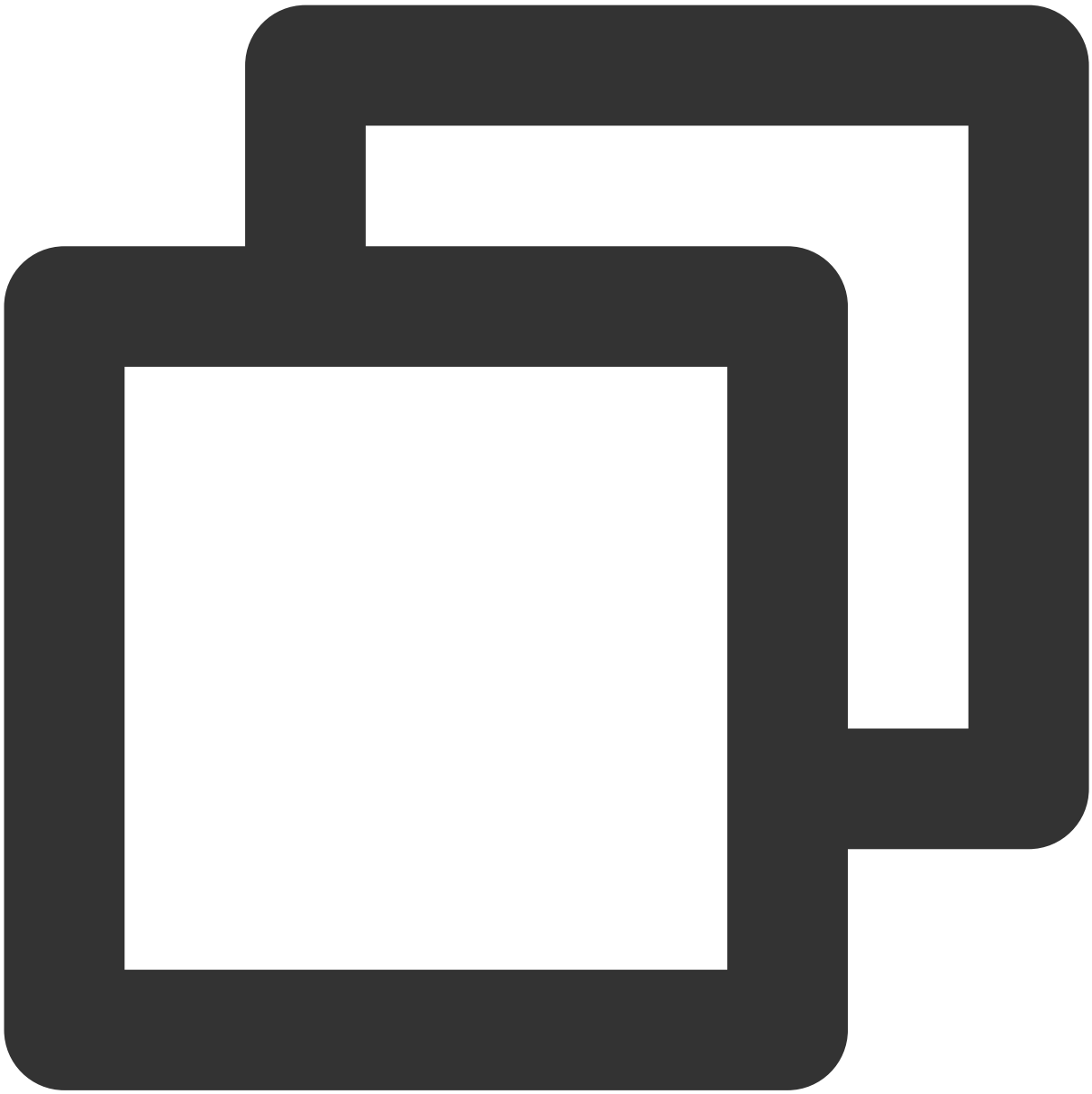
1. 登录 云点播控制台，在左侧导航栏选择**媒资管理**。
2. 单击列表上方的**前缀搜索**，选择**前缀搜索**，在搜索框输入关键词，例如 `1400000123_1001_rexchang_main` ，单击搜索按钮，将展示视频名称前缀相匹配的视频文件。
3. 您可以根据创建时间筛选所需的目标文件。

方式二：通过点播 REST API 查找

腾讯云点播系统提供了一系列 REST API 来管理其上的音视频文件，您可以通过 搜索媒体信息 这个 REST API 来查询您在点播系统上的文件。您可以通过请求参数表中的 `Text` 参数进行模糊匹配，也可以根据 `StreamId` 参数

进行精准查找。

REST 请求示例：



```
https://vod.tencentcloudapi.com/?Action=SearchMedia
&StreamId=stream1001
&Sort.Field=CreateTime
&Sort.Order=Desc
&<公共请求参数>
```

接收录制文件

除了 查找录制文件，您还可以通过配置回调地址，让腾讯云主动把新录制文件的消息推送给您的服务器。
房间里的最后一路音视频流退出后，腾讯云会结束录制并将文件转存到云点播平台，该过程大约默认需要30秒至2分钟（若您设置了续录时间为300秒，则等待时间将在默认基础上叠加300秒）。转存完成后，腾讯云会通过您在 设置录制回调 中设置的回调地址（HTTP/HTTPS）向您的服务器发送通知。
腾讯云会将录制和录制相关的事件都通过您设置的回调地址推送给您的服务器，回调消息示例如下图所示：


```
{
    "app": "8888.livepush.myqcloud.com",
    "appid": 1234567890,
    "appname": "trtc_1400000123",
    "channel_id": "1400000123_1001_rexchang_main",
    "duration": 39,
    "end_time": 1581926501,
    "end_time_usec": 817545,
    "event_type": 100,
    "file_format": "mp4",
    "file_id": "5285890798833426017",
    "file_size": 3337482,
    "media_start_time": 0,
    "record_bps": 0,
    "record_file_id": "5285890798833426017",
    "start_time": 1581926464,
    "start_time_usec": 588890,
    "stream_id": "1400000123_1001_rexchang_main",
    "stream_param": "txSecret=ecd19683f6cf1a7615f13670e0834de1&txTime=70d4653d&from=interactive&cli
                     sdkappid=1400000123&sdkapptype=1&groupid=1001&userid=cmV4Y2hhbmc=&ts=5e4a483c&
                     testfile&tinyid=1441152145405491 89&roomid=2294546&cliRecoId=0&trtcclientip=222
                     ayer=1",
    "task_id": "18025695036262267 07",
    "video_id": "1234567890_46ac1271218249118752d57423120300",
    "video_url": "http://1234567890.vod2.myqcloud.com/5022b150vodcq1234567890/39e578f12280795898833
}
```


您可以通过下表中的字段来确定当前回调是对应的哪一次通话（或直播）：

序号	字段名	说明
1	event_type	消息类型，当 event_type 为100时，表示该回调消息为录制文件生成的消息。
2	stream_id	即直播 CDN 的 streamId，您可以在进房时通过设置 TRTCParams 中的 streamId 字段指定（推荐），也可以在调用 TRTCCloud 的 startPublishing 接口时通过参数 streamId 来指定。
3	stream_param.userid	用户名的 Base64 编码。
	stream_param.userdefinerecordid	自定义字段，您可以通过设置 TRTCParams 中的 userDefineRecordId 字段指定。

4 5	video_url	录制文件的观看地址，可以用于 点播回放。

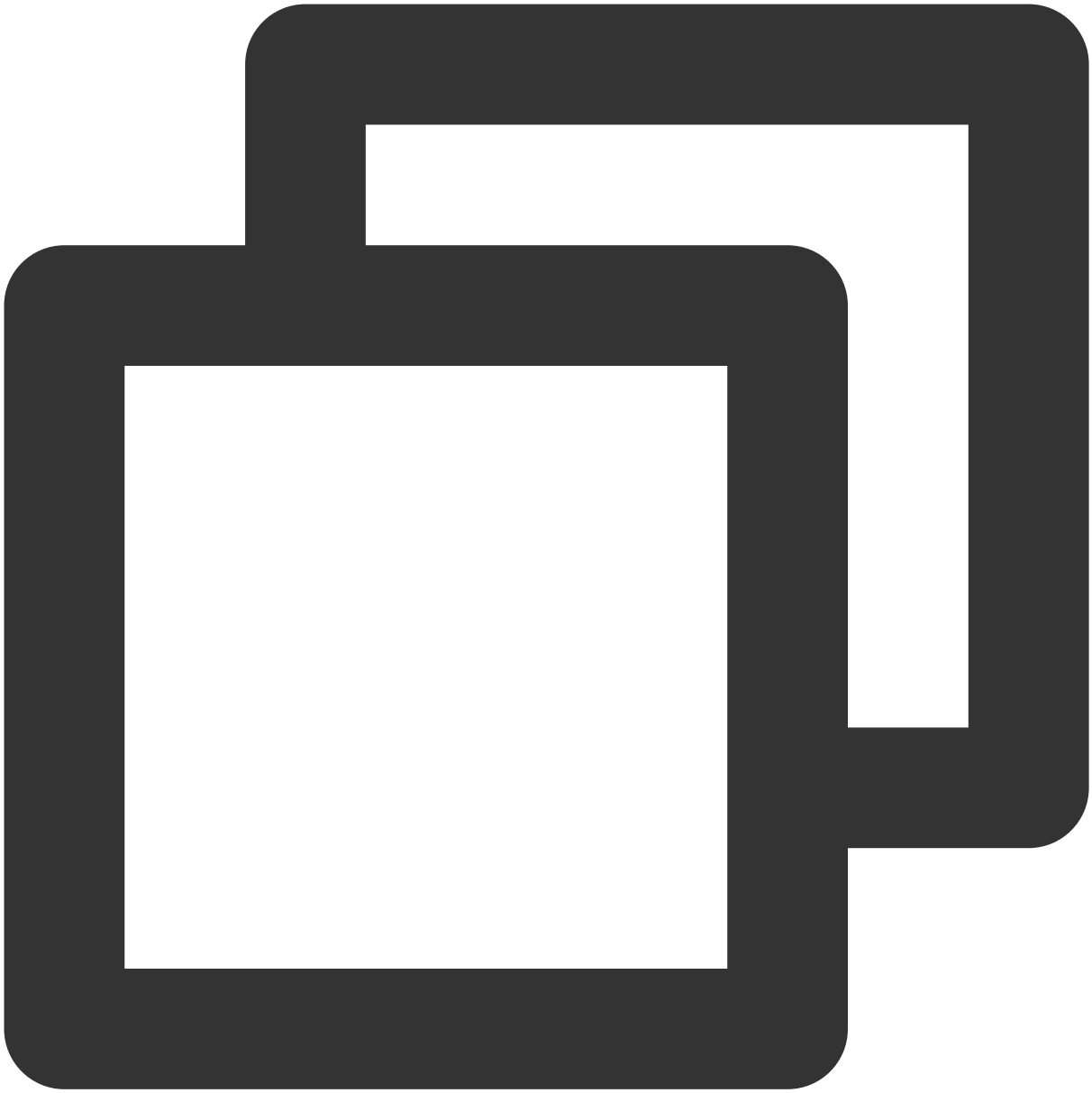
说明

更多回调字段说明，请参见 云直播-录制事件通知。

删除录制文件

腾讯云点播系统提供了一系列 REST API 来管理其上的音视频文件，您可以通过 删除媒体 API 删除某个指定的文件。

REST 请求示例：



```
https://vod.tencentcloudapi.com/?Action=DeleteMedia
&FileId=5285890798866415058 7
&<公共请求参数>
```

回放录制文件

在线教育等场景中，通常需要在直播结束后多次回放录制文件，以便充分利用教学资源。

选择文件格式（HLS）

在 设置录制格式 中选择文件格式为 HLS。
HLS 支持最长三十分钟的断点续录，可以做到“一场直播（或一堂课）只产生一个回放链接”，且 HLS 文件支持绝大多数浏览器在线播放，非常适合视频回放场景。

获取点播地址（video_url）

在 接收录制文件 时，可以获取回调消息中 **video_url** 字段，该字段为当前录制文件在腾讯云的点播地址。

对接点播播放器

根据使用平台对接点播播放器，具体操作参考如下：
iOS 平台
Android 平台
Web 浏览器

注意

建议使用 专业版 TRTC SDK，专业版集合了 超级播放器（Player+）、移动直播（MLVB） 等功能，由于底层模块的高度复用，集成专业版的体积增量要小于同时集成两个独立的 SDK，并且可以避免符号冲突（symbol duplicate）的困扰。

相关费用

云端录制与回放功能使用到的功能包括：云服务资源（包括云端录制服务、云点播的回放文件存储与处理、云点播的播放服务等），和终端 SDK 播放点播视频的能力。可能会根据实际需求产生以下费用。

云服务消耗

云服务消耗包括了云端录制产生的费用和回放文件产生的消耗费用。您可根据实际需求进行使用。

说明

本文中的价格为示例，仅供参考。若价格与实际不符，请以 云端录制计费说明、云直播 和 云点播 的定价为准。

录制费用：转码或转封装产生的计算费用

由于录制需要进行音视频流的转码或转封装，会产生服务器计算资源的消耗，因此需要按照录制业务对计算资源的占用成本进行收费。

注意

自2020年7月1日起首次在 TRTC 控制台创建应用的腾讯云账号，使用云端录制功能后产生的录制费用以 云端录制计费说明 为准。
在2020年7月1日之前已经在 TRTC 控制台创建过应用的腾讯云账号，无论是在2020年7月1日之前还是之后创建的应用，使用云端录制功能产生的录制费用均默认继续延用**直播录制**的计费规则。
直播录制计费的计算方法是按照并发录制的路数进行收费，并发数越高录制费用越高，具体计费说明请参见 云直播 > 直播录制。

计算公式：
录制有效天数占比 = 一个自然月使用录制功能的天数 / 当月总天数。
录制费用 = 一个自然月录制并发路数峰值 × 录制有效天数占比 × 录制路数单价。

例如，您4月份有1000个主播，如果在晚高峰时，最多同时有500路主播的音视频流需要录制，同时在4月份共有6天使用了录制功能（有效天数占比为6/30）。假设录制单价为5.2941美元/路/月，那么4月份总录制费用为 500路 × 5.2941美元/路/月 = 2647.05美元/月。
如果您在 设置录制格式 时同时选择了两种录制文件，录制费用和存储费用都会 × 2，同理，选择三种文件时录制费用和存储费用会 × 3。 如非必要，建议只选择需要的一种文件格式，可以大幅节约成本。

转码费用：如开启混流录制则会产生该费用

如果您启用了混流录制，由于混流本身需要进行解码和编码，所以还会产生额外的混流转码费用。 混流转码根据分辨率大小和转码时长进行计费，主播用的分辨率越高，连麦时间（通常在连麦场景才需要混流转码）越长，费用越高，具体费用计算请参见 直播转码。

例如，您通过 setVideoEncoderParam() 设置主播的码率（videoBitrate）为1500kbps，分辨率为720P。如果有一位主播跟观众连麦了一个小时，连麦期间开启了 云端混流，那么产生的转码费用为 0.0057 美元/分钟 × 60分钟 = 0.342美元。

回放文件存储费用：云端视频存储服务产生的存储费用

录制出的视频文件存放于云点播服务，会使用云点播的云端存储服务。由于存储本身会产生磁盘资源的消耗，因此需要按照存储的资源占用进行收费。存储的时间越久费用也就越高，因此如无特殊需要，您可以将文件的存储时间设置的短一些来节省费用，或者将文件存放在自己的服务器上。存储费用可以选择 视频存储（日结）价格 进行日结计算。

例如，您通过 setVideoEncoderParam() 设置主播的码率（videoBitrate）为1000kbps，录制该主播的直播视频（选择一种文件格式），录制一小时大约会产生一个 `(1000 / 8)KBps × 3600秒 = 450000KB = 0.45GB` 大小的视频文件，该文件每天产生的存储费用约为 0.45GB × 0.0009 美元/GB/日 = 0.000405美元。

观看回放费用：云点播视频进行分发播放产生的播放费用

如果您录制的视频文件要被用于回看播放，会使用云点播的 CDN 播放功能。由于观看本身会产生 CDN 流量消耗，因此需要按照点播的价格进行计费，默认按流量收费。观看的人数越多费用越高，观看费用可以选择 视频加速（日结）价格 进行日结计算。

例如，您通过云端录制产生了一个1GB大小的文件，且有1000位观众从头到尾完整地观看了视频，大约会产生1TB的点播观看流量，那么按照阶梯价格表，1000位观众就会产生 1000 × 1GB ×0.0794美元/GB = 79.4美元 的费用，按照流量套餐包则是24.5美元。
如果您选择从腾讯云下载文件到您的服务器上，也会产生一次很小的点播流量消耗，并且会在您的月度账单中有所体现。

SDK 播放授权

实时音视频（TRTC）全功能版本 SDK 提供了功能全面性能强大的视频播放能力，可轻松配合云点播实现视频播放功能。移动端 SDK 在10.1及以上的版本可通过获取指定 License 以解锁视频播放能力。

注意

TRTC 的播放能力无需 License 授权。

含 UI 在线直播（旧）
集成 TUILiveRoom (Android)

最近更新时间：2024-06-07 11:20:49

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：

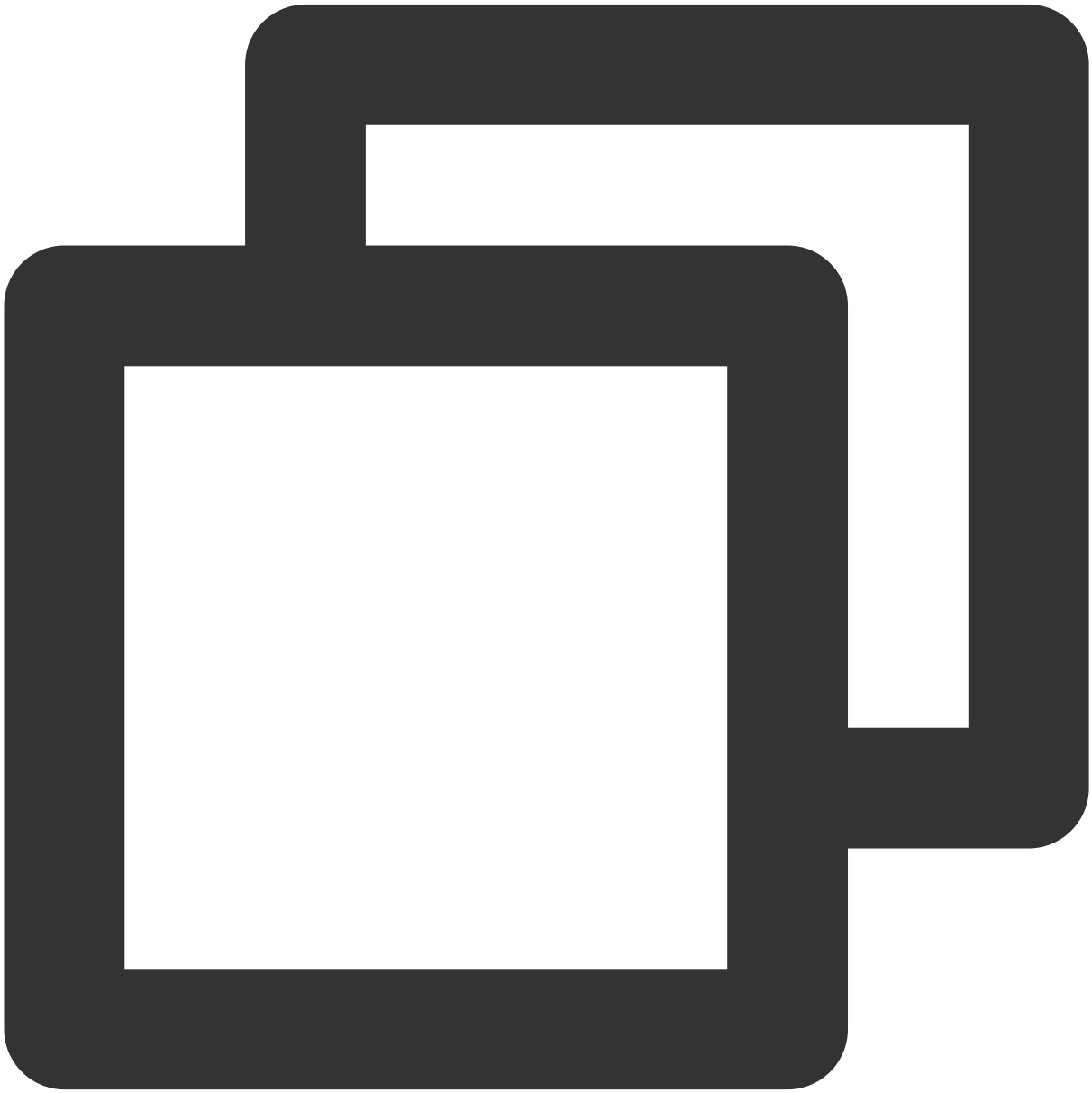
说明

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

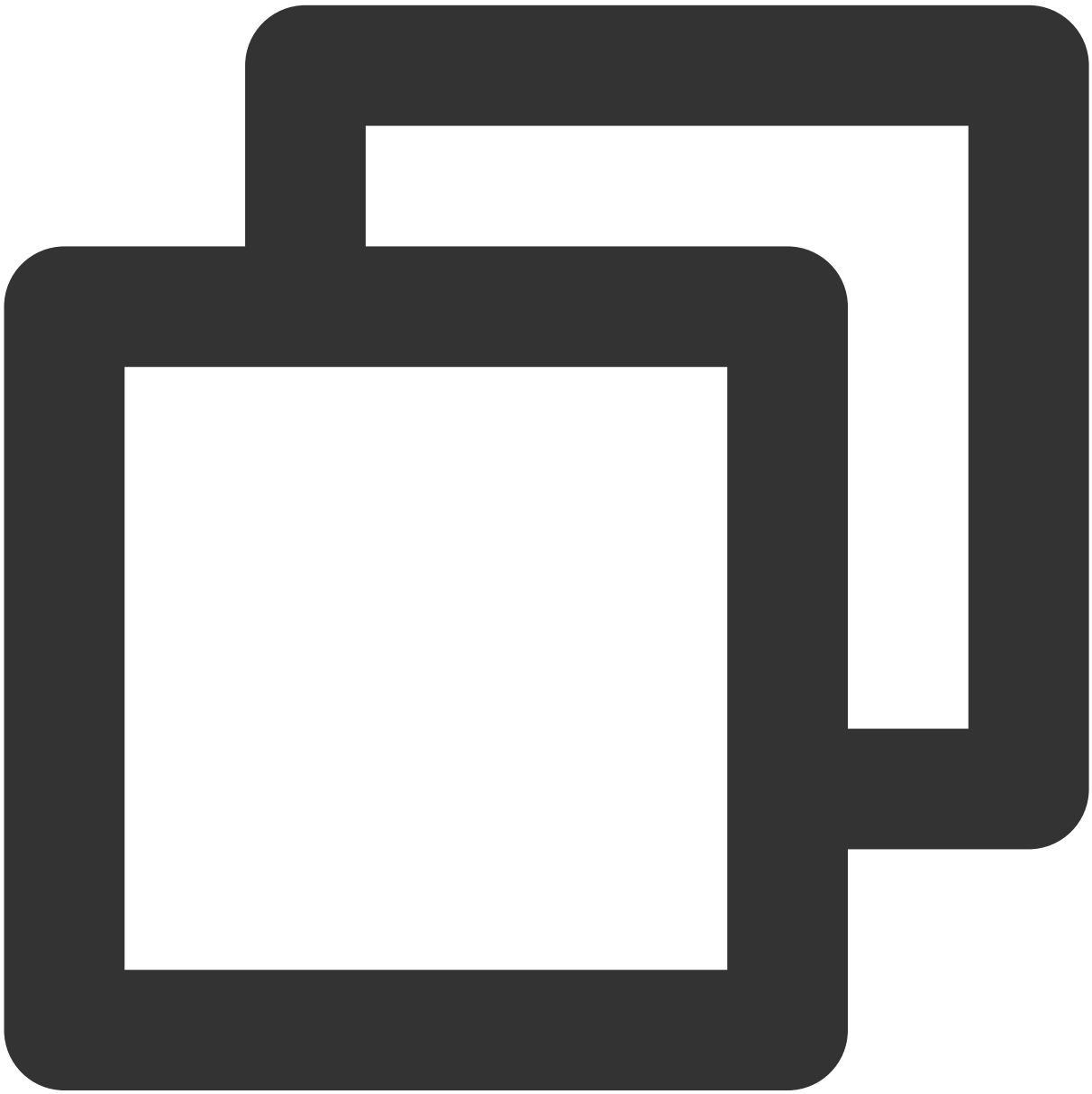
单击进入 Github，选择克隆/下载代码，然后拷贝 `Android/debug` ， `Android/tuiaudioeffect` ， `Android/tuibarrage` ， `Android/tuibeauty` ， `Android/tuigift` 和 `Android/tuiliveroom` 目录到您的工程中，并完成如下导入动作：
在 `setting.gradle` 中完成导入，参考如下：



```
include ':debug'
```

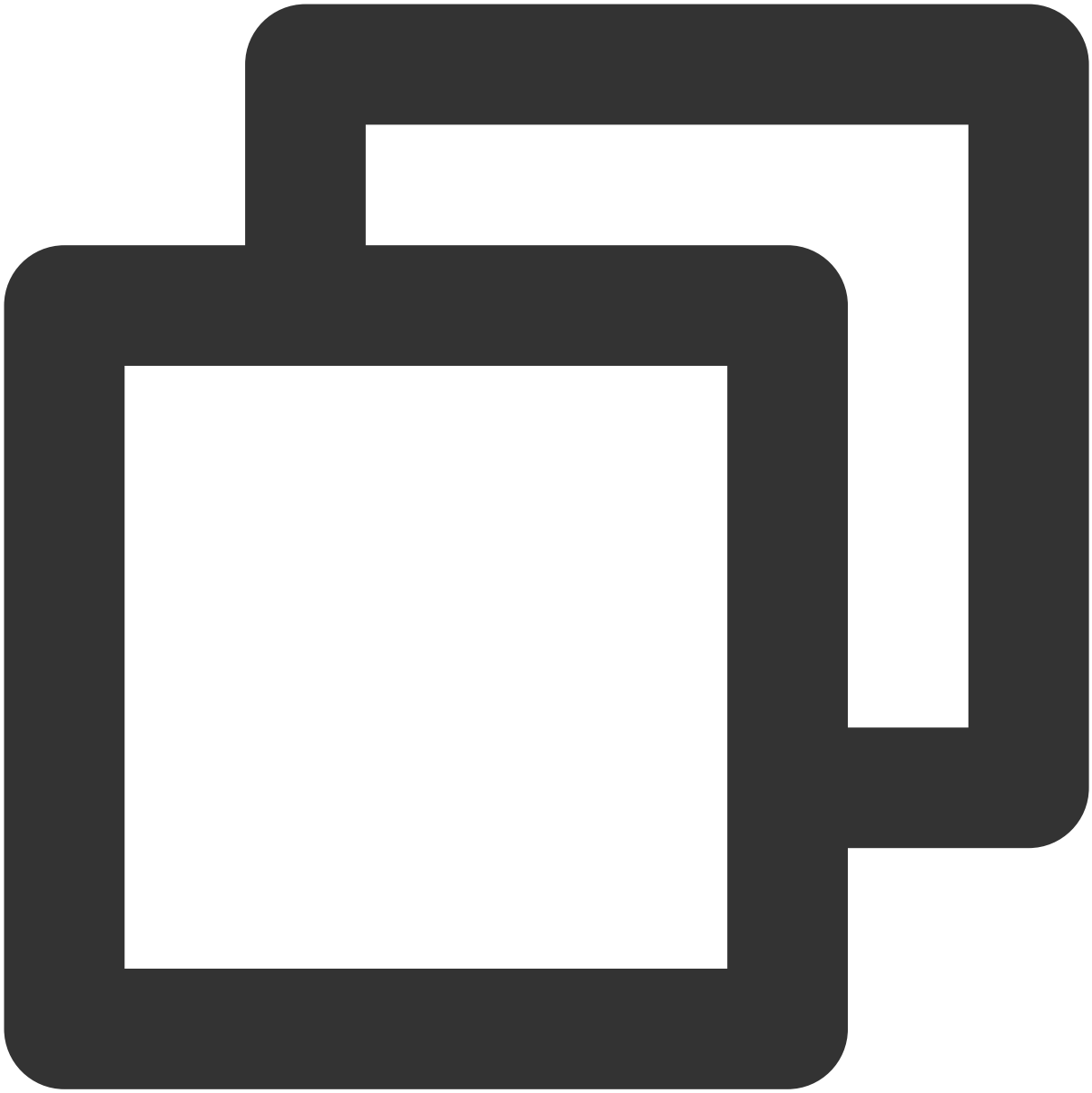
```
include ':tuibeauty'
include ':tuibarrage'
include ':tuiaudioeffect'
include ':tuigift'
include ':tuiliveroom'
```

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



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

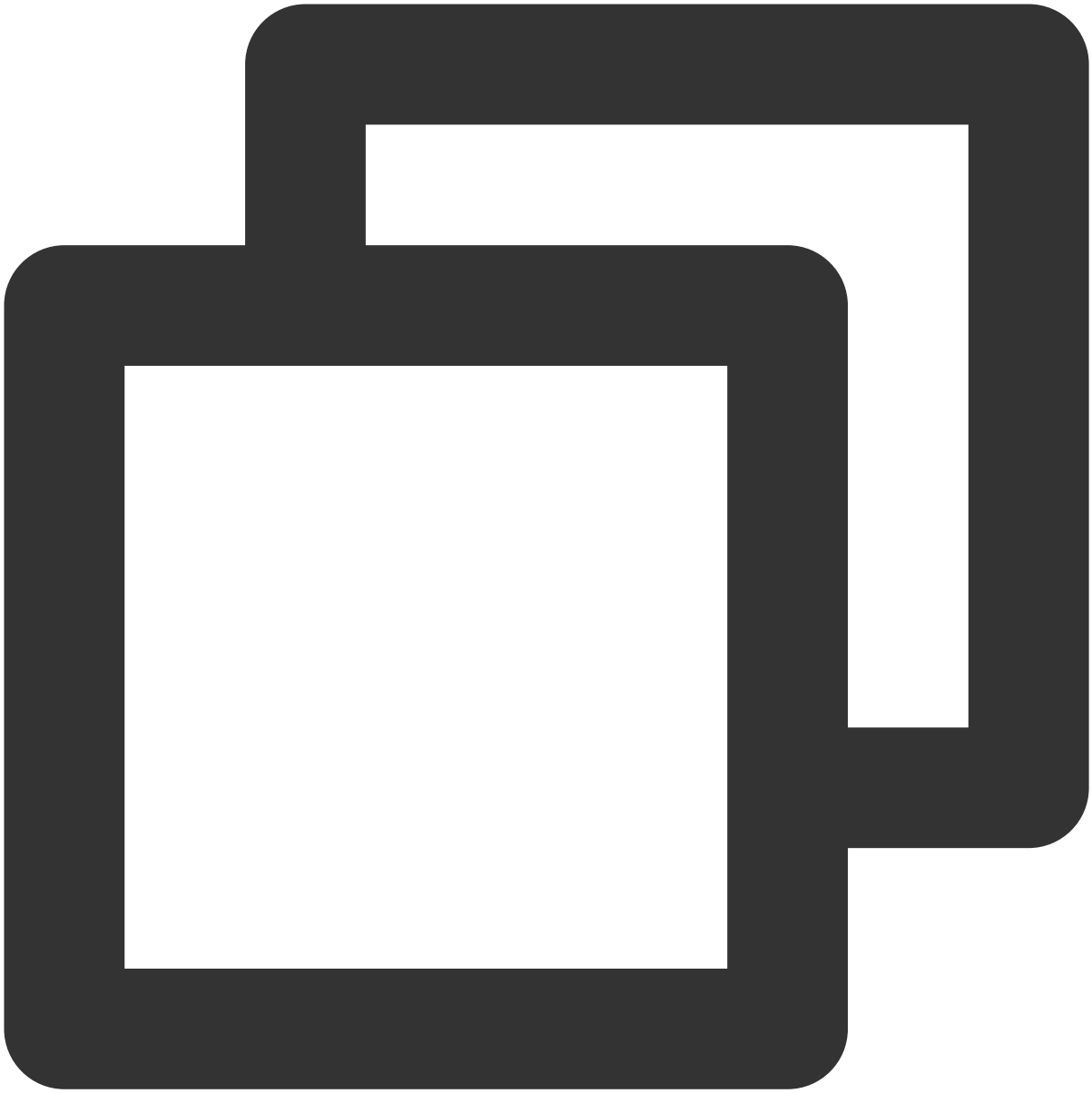
在根目录的 `build.gradle` 文件中添加 `TRTC SDK` 和 `IM SDK` 的依赖：



```
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

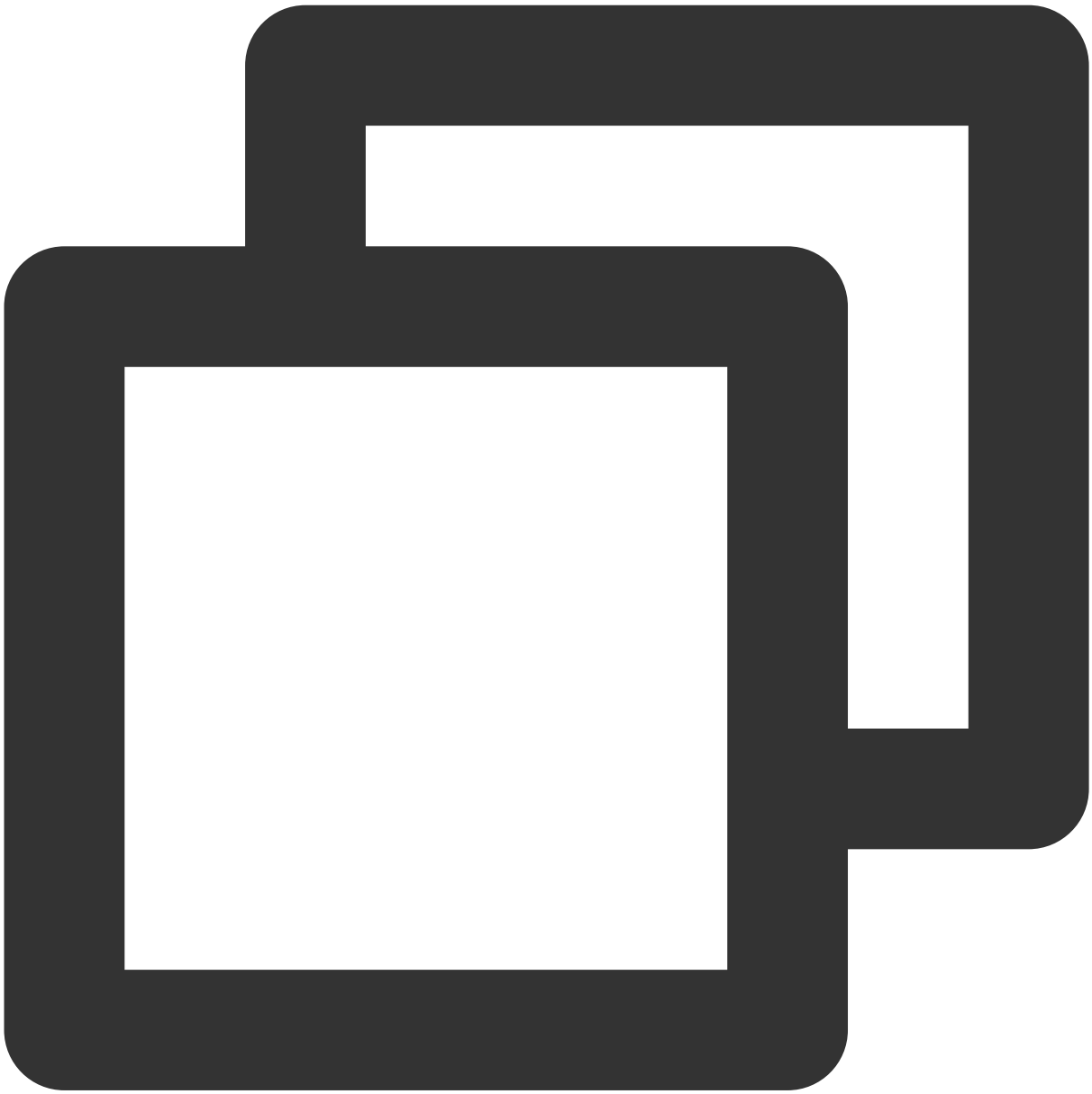
1. 在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：



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

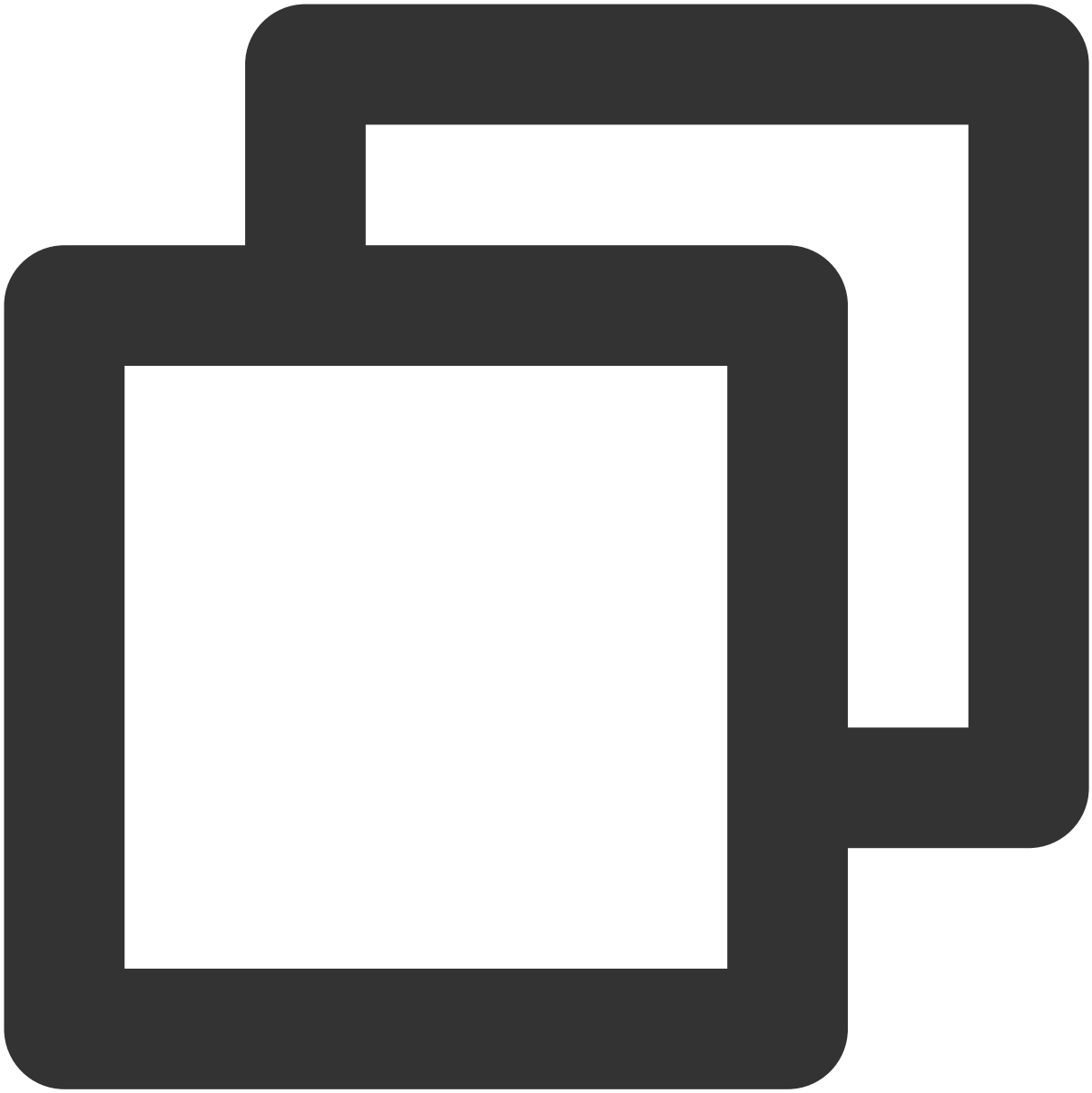
```
<uses-feature android:name="android.hardware.camera.autofocus" />
```

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 TUICal
    @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：

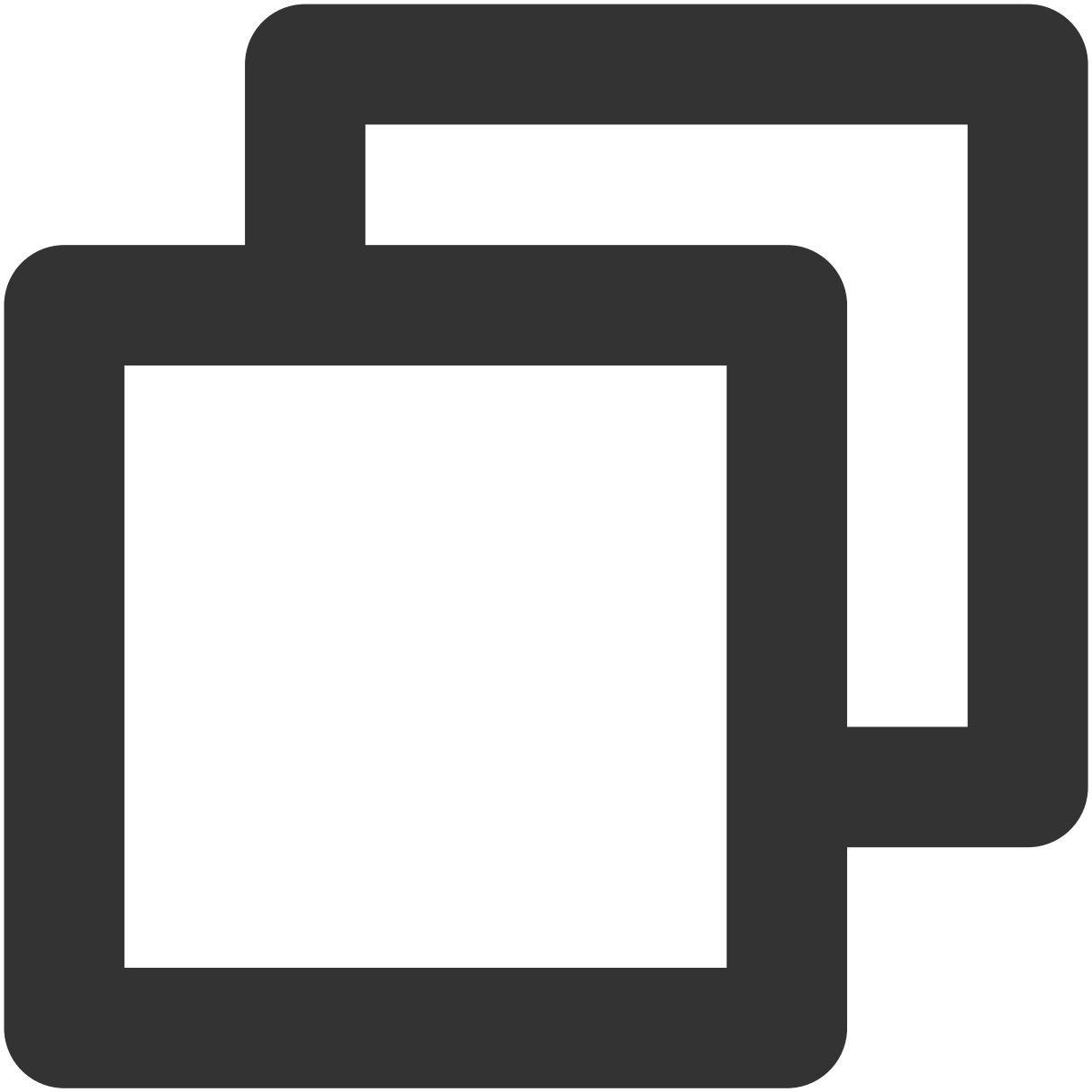
Secretkey：**TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。

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。

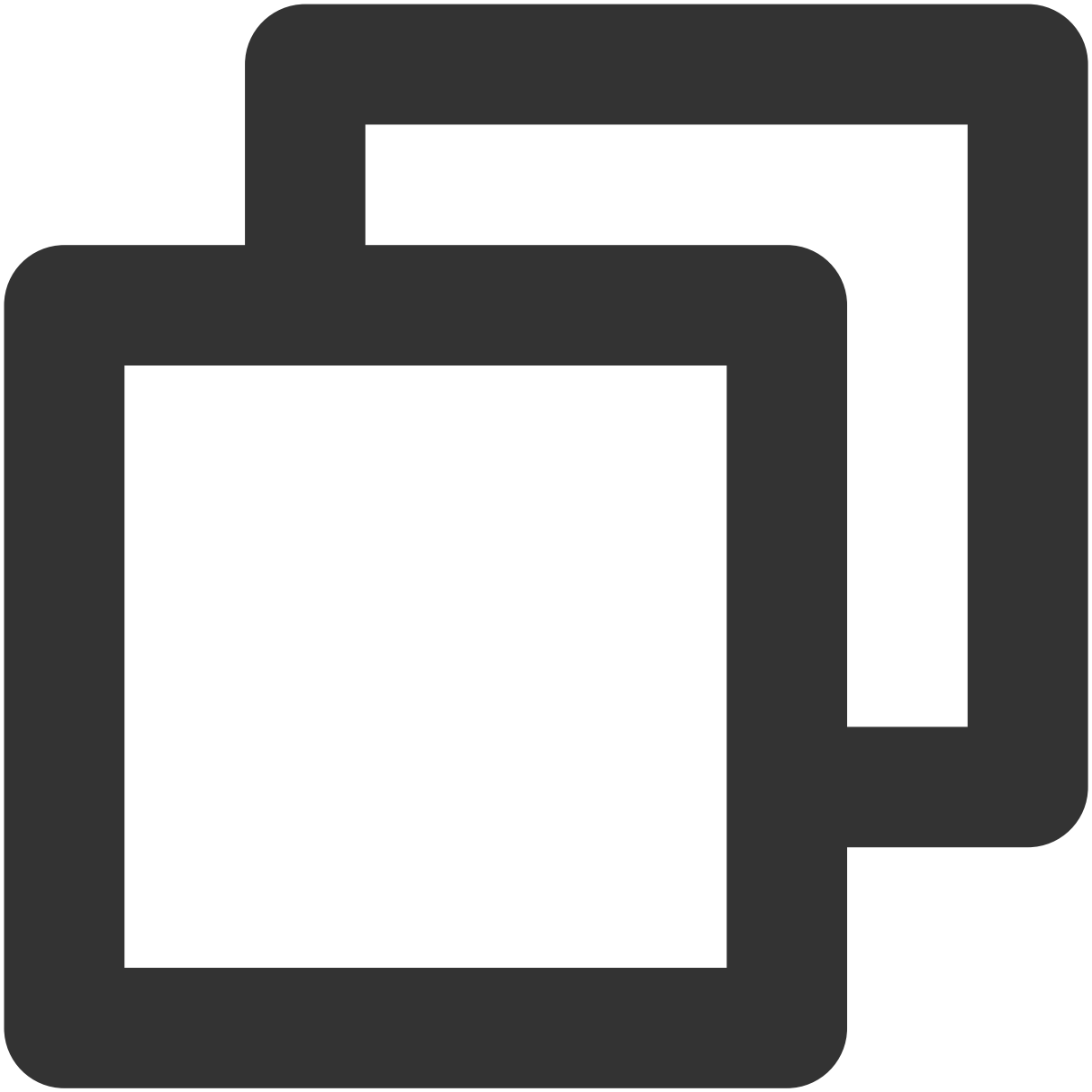
步骤四：实现视频互动直播间

1. **主播端开播**



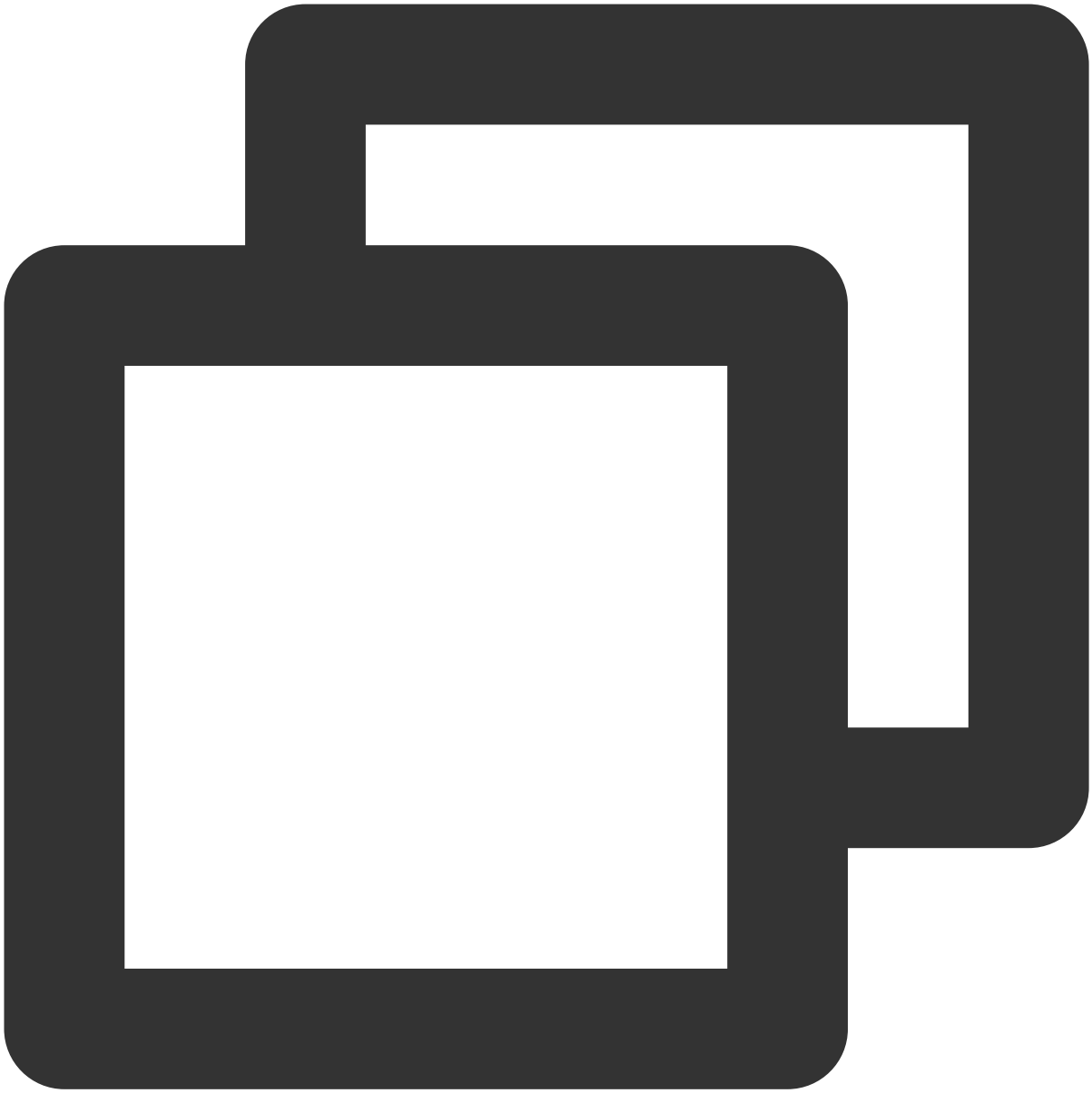
```
mLiveRoom.createRoom(int roomId, String roomName, String coverUrl);
```

2. **观众端观看**



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

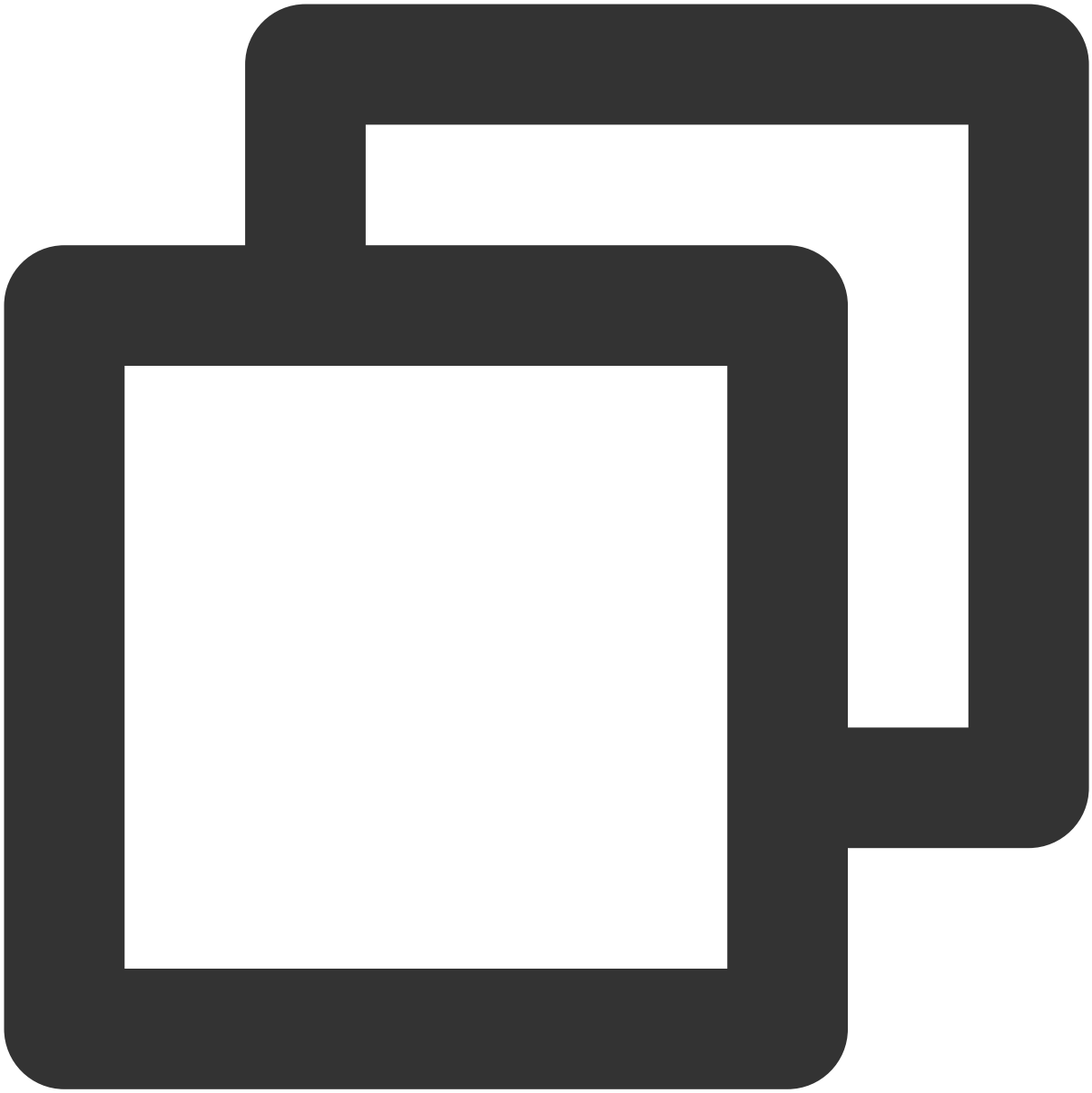
3. **观众与主播连麦** TRTCLiveRoom#requestJoinAnchor



```
// 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

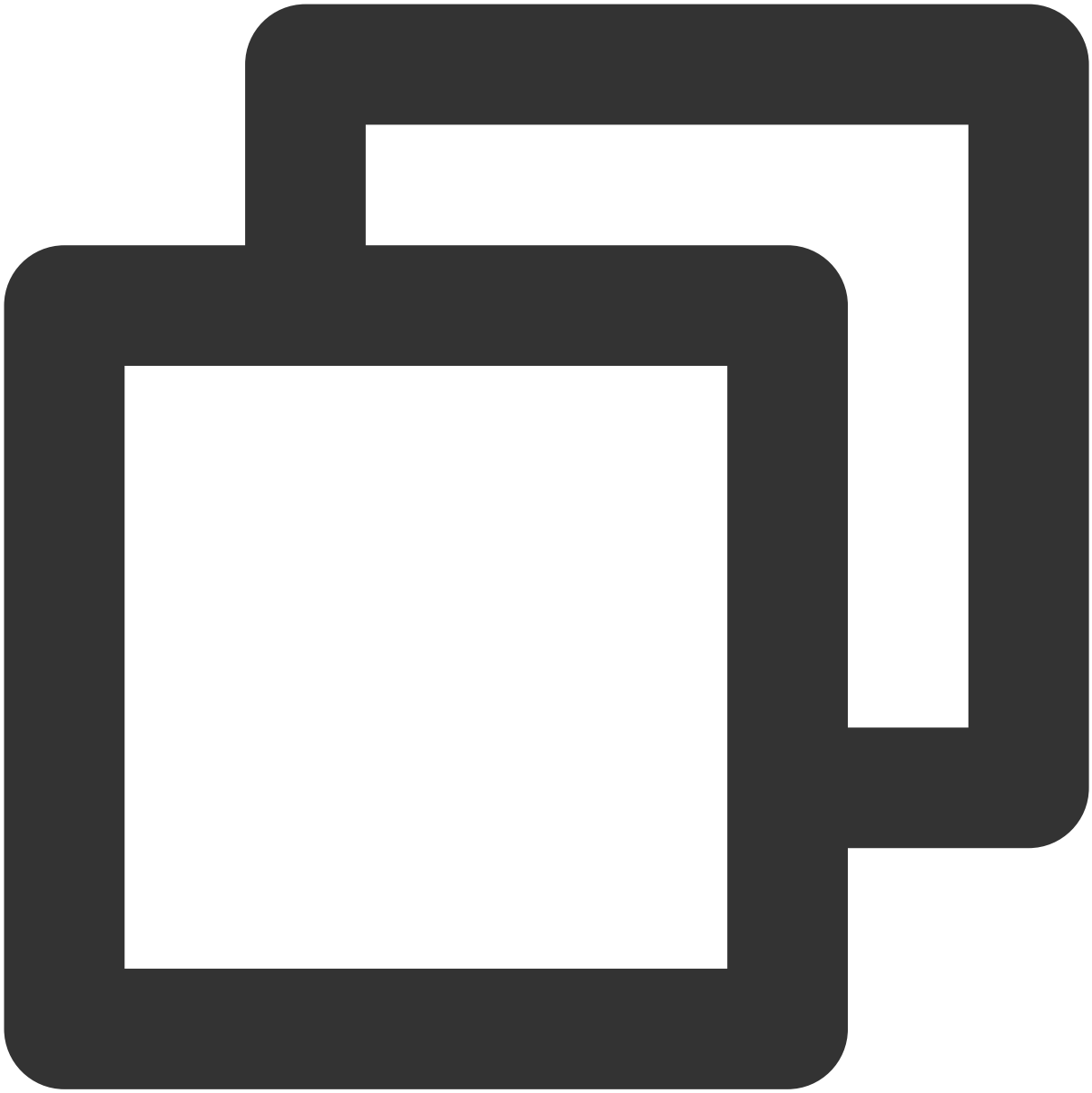


```
// 主播 A 创建12345的房间
mLiveRoom.createRoom(12345, "roomA", "Your coverUrl");
// 主播 B 创建54321的房间
mLiveRoom.createRoom(54321, "roomB", "Your coverUrl");

// 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”, “XMagicLic
```

交流与反馈

如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

集成 TUILiveRoom (iOS)

最近更新时间：2023-11-20 17:51:47

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

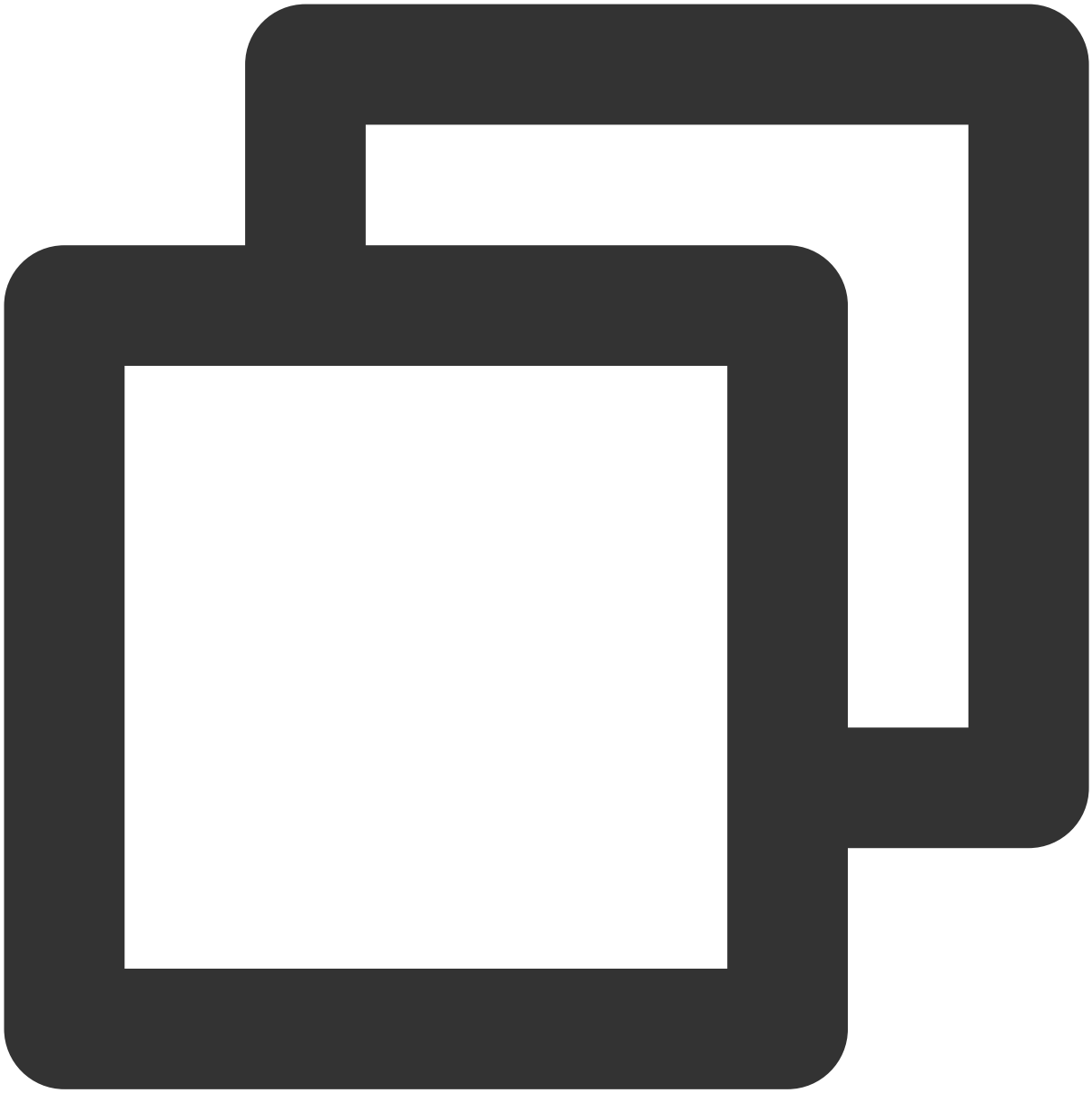
TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 - 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

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` 、 `TUIKitCommon` 文件夹 和 `TUILiveRoom.podspec` 文件拷贝到您在 `步骤1` 创建的 TUILiveRoom 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖，之后执行 `pod install` 命令，完成导入。



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

```
pod 'TUIGift', :path => "./TUILiveRoom/TUIGift/"
```

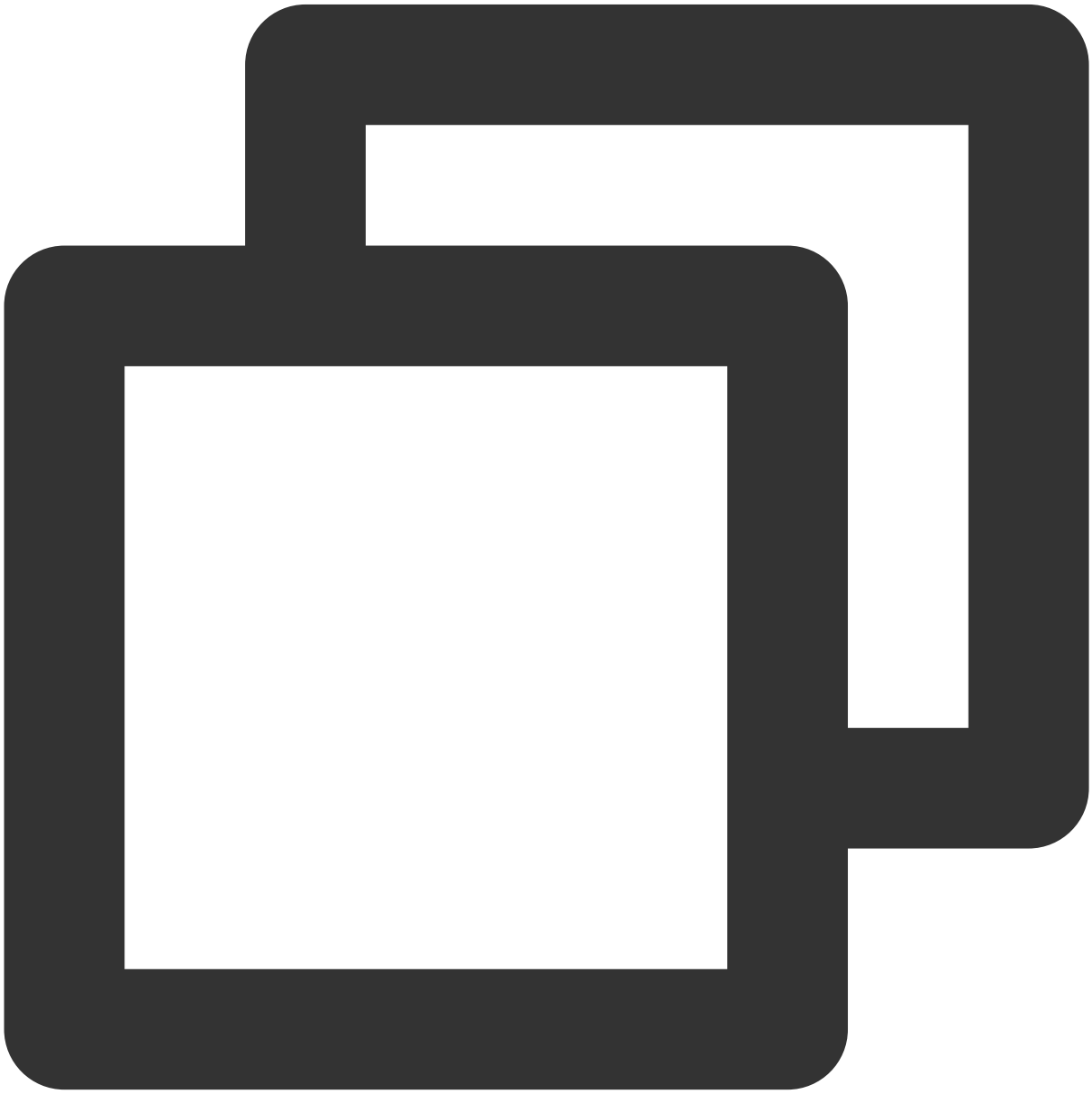
注意

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

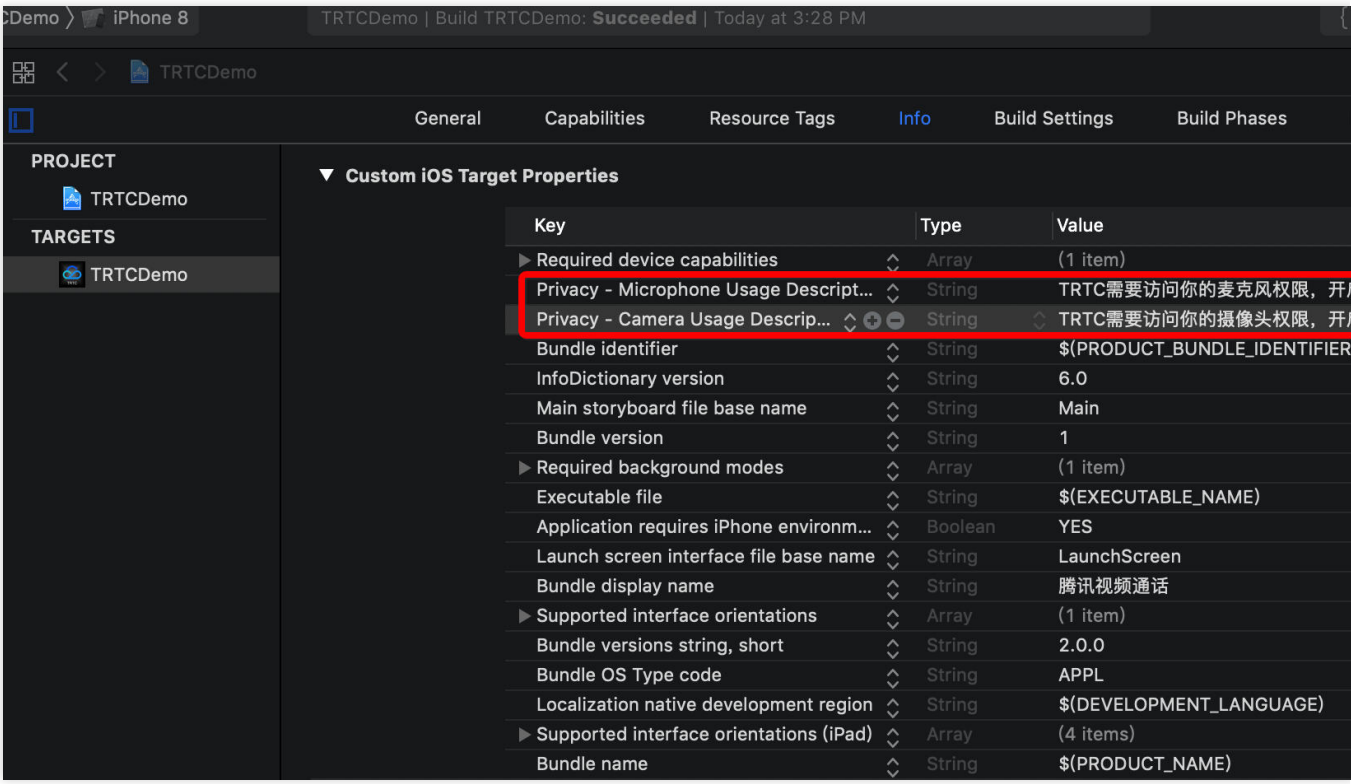
TUIKitCommon.podspec 在 TUIKitCommon文件夹下。

步骤二：配置权限

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



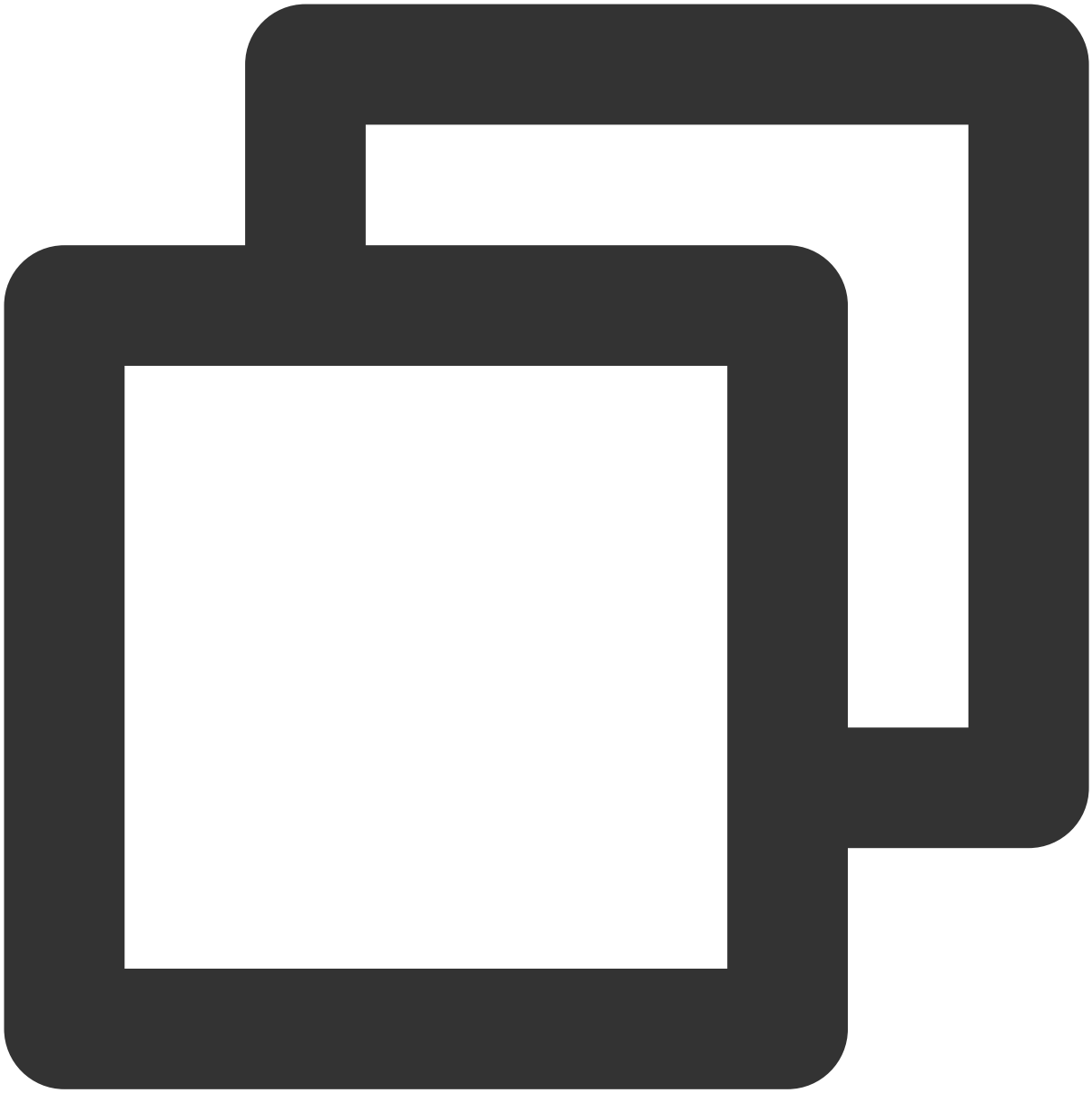
```
<key>NSCameraUsageDescription</key>
<string>RoomApp需要访问您的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>RoomApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```



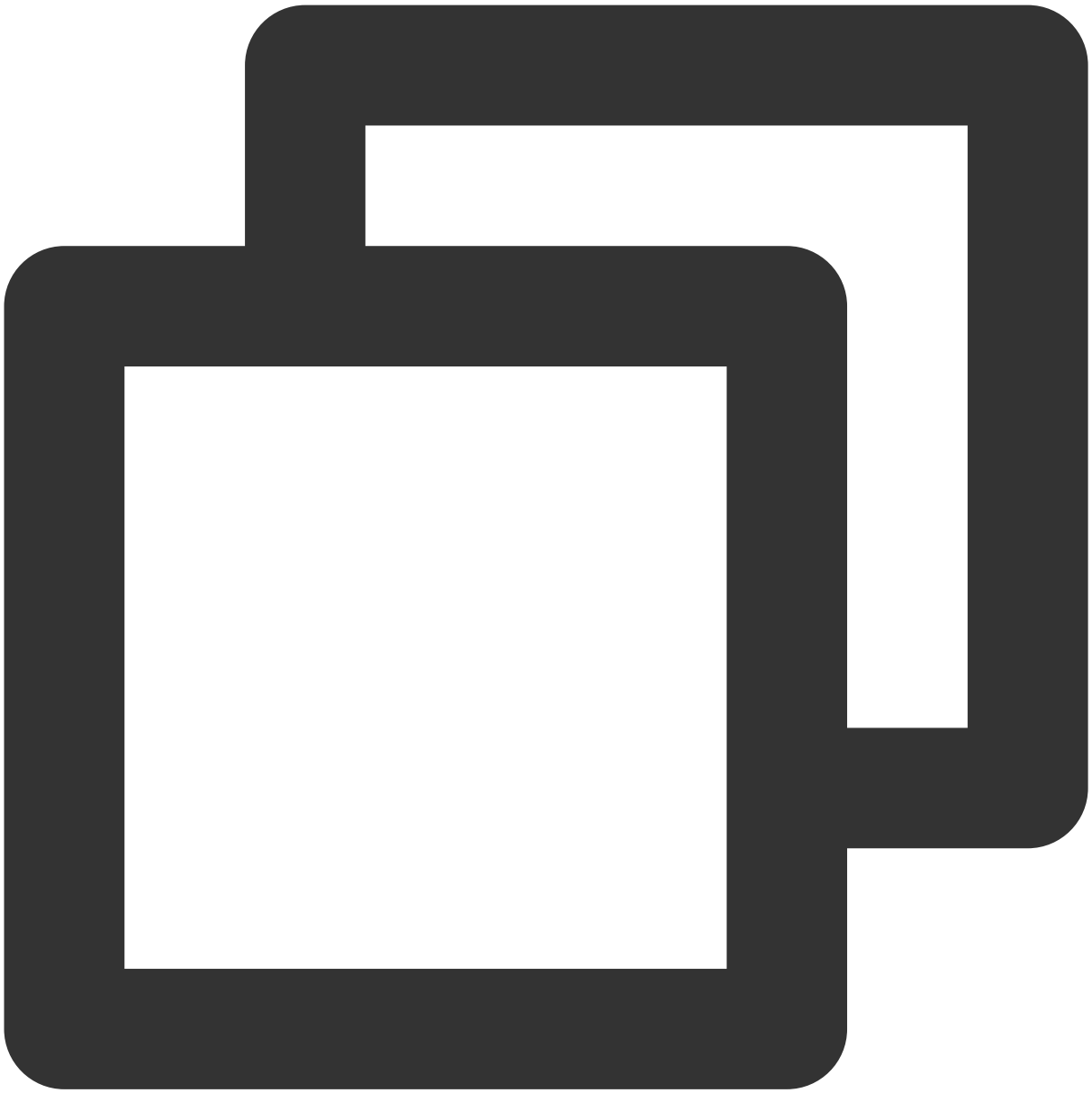
步骤三：初始化并登录组件

Objective-C

Swift



```
@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];
```



```
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。

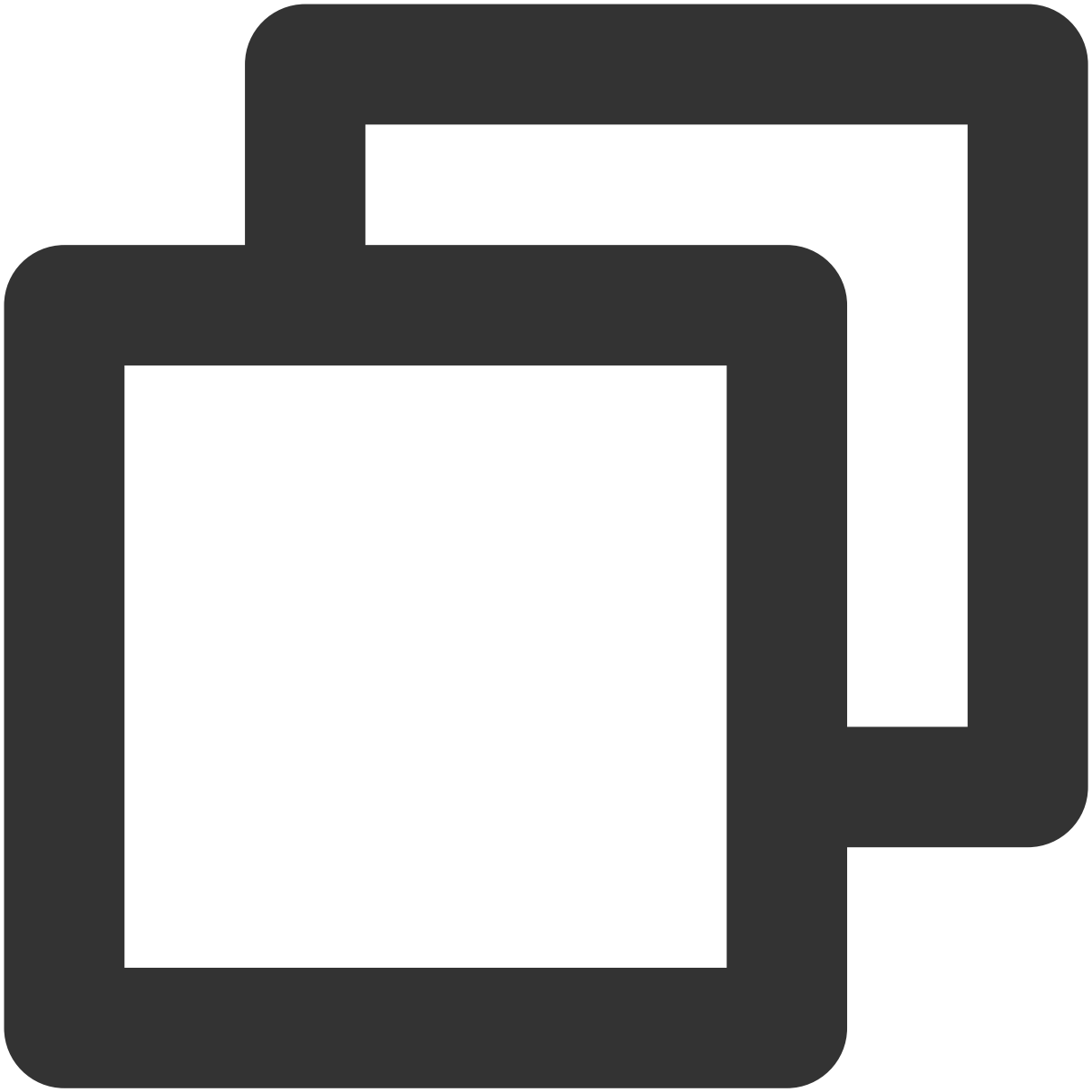
UserSig：根据 SDKAppId、UserID，SecretKey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUILiveRoom 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现视频互动直播间

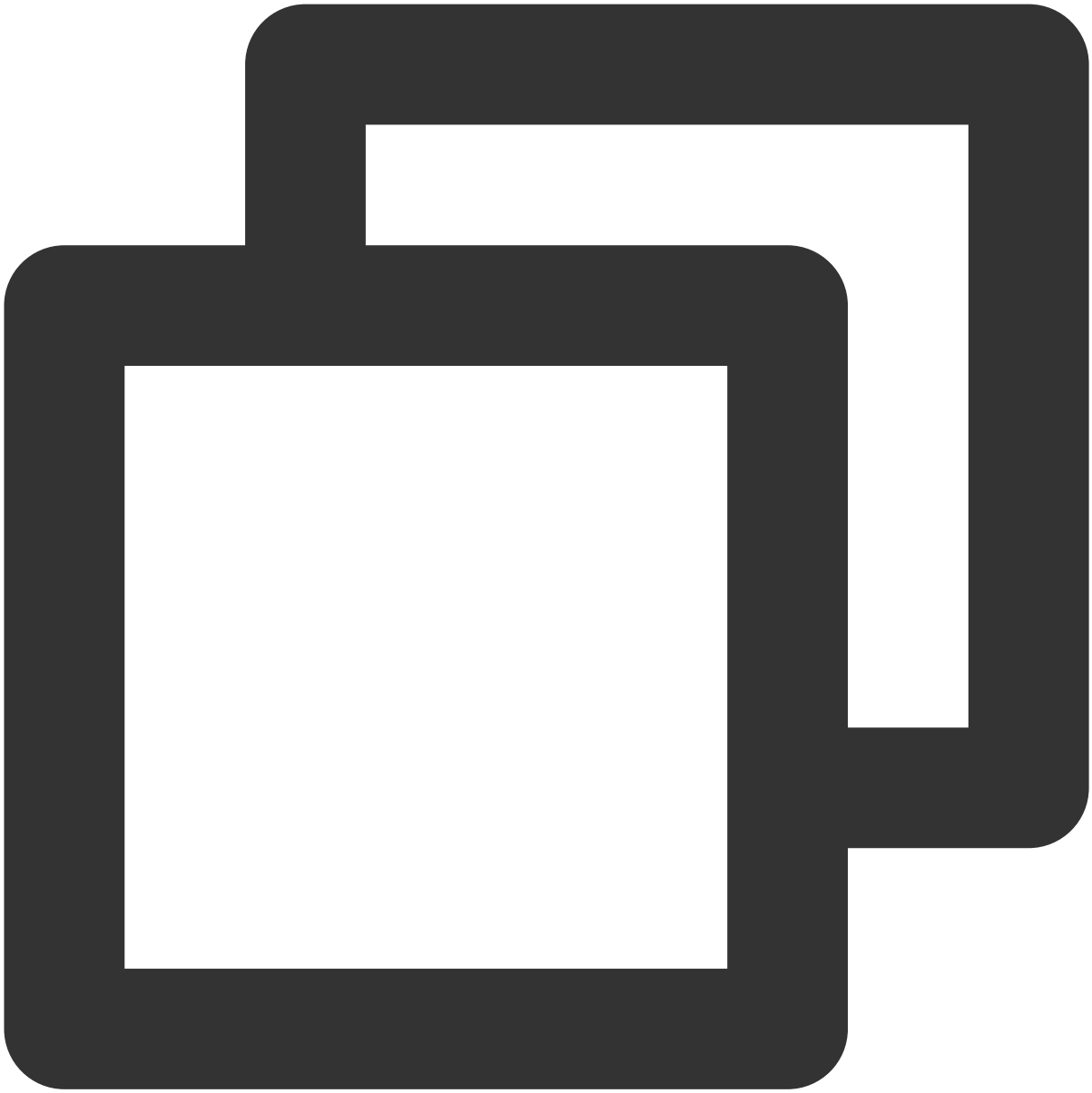
1. **主播端开播**

Objective-C

Swift



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

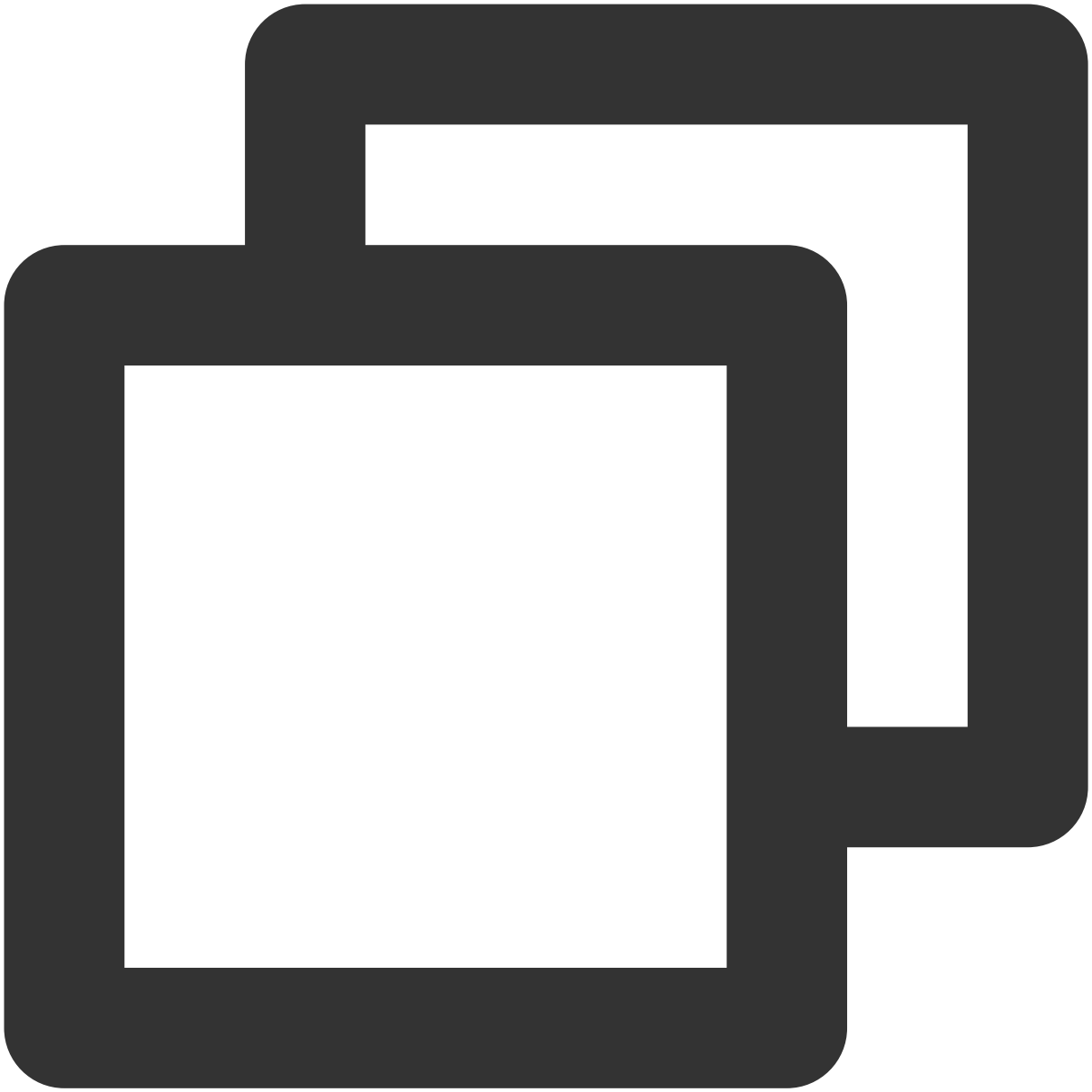


```
mLiveRoom.createRoom(roomId: 123, roomName: "test room", coverUrl:"")
```

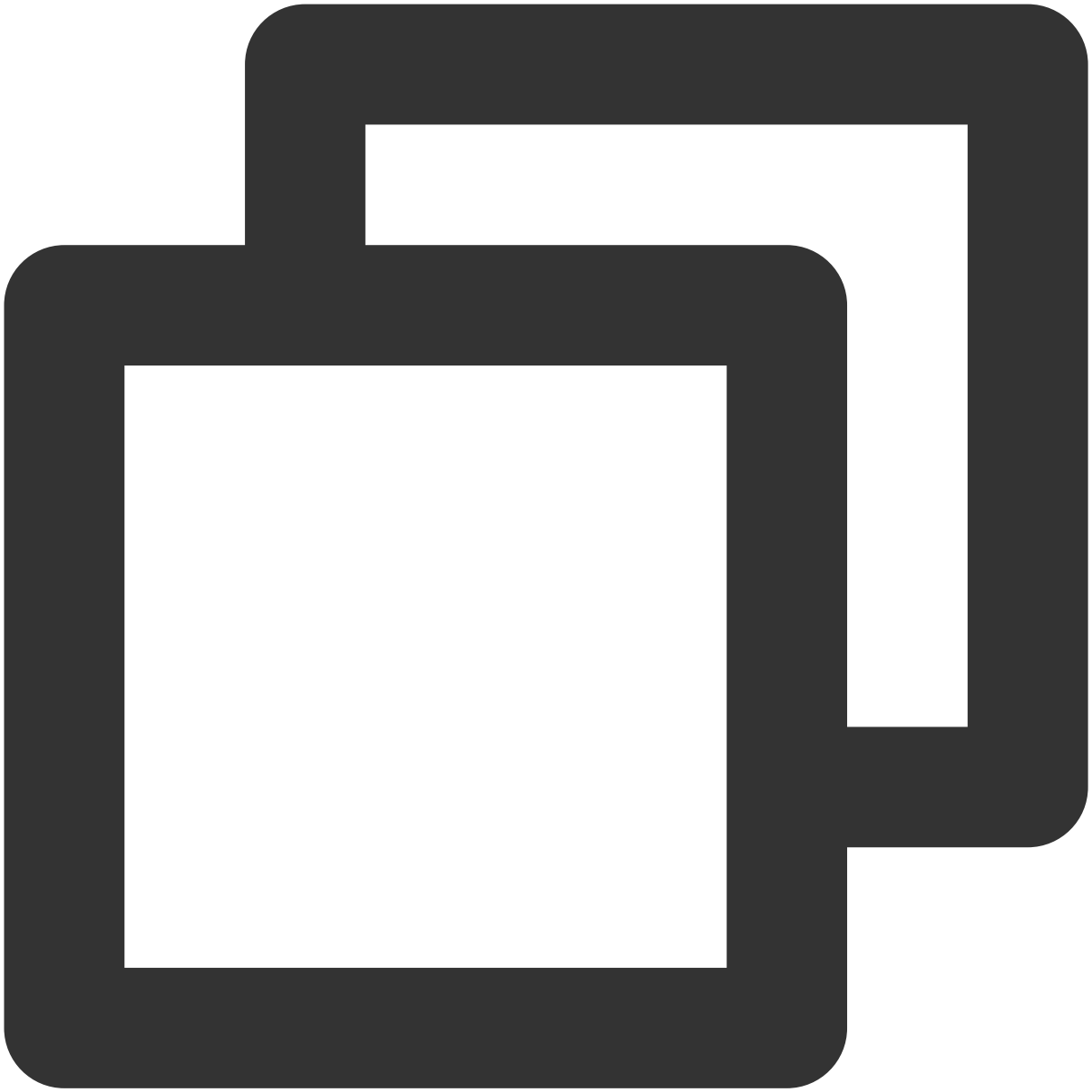
2. **观众端观看**

Objective-C

Swift



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

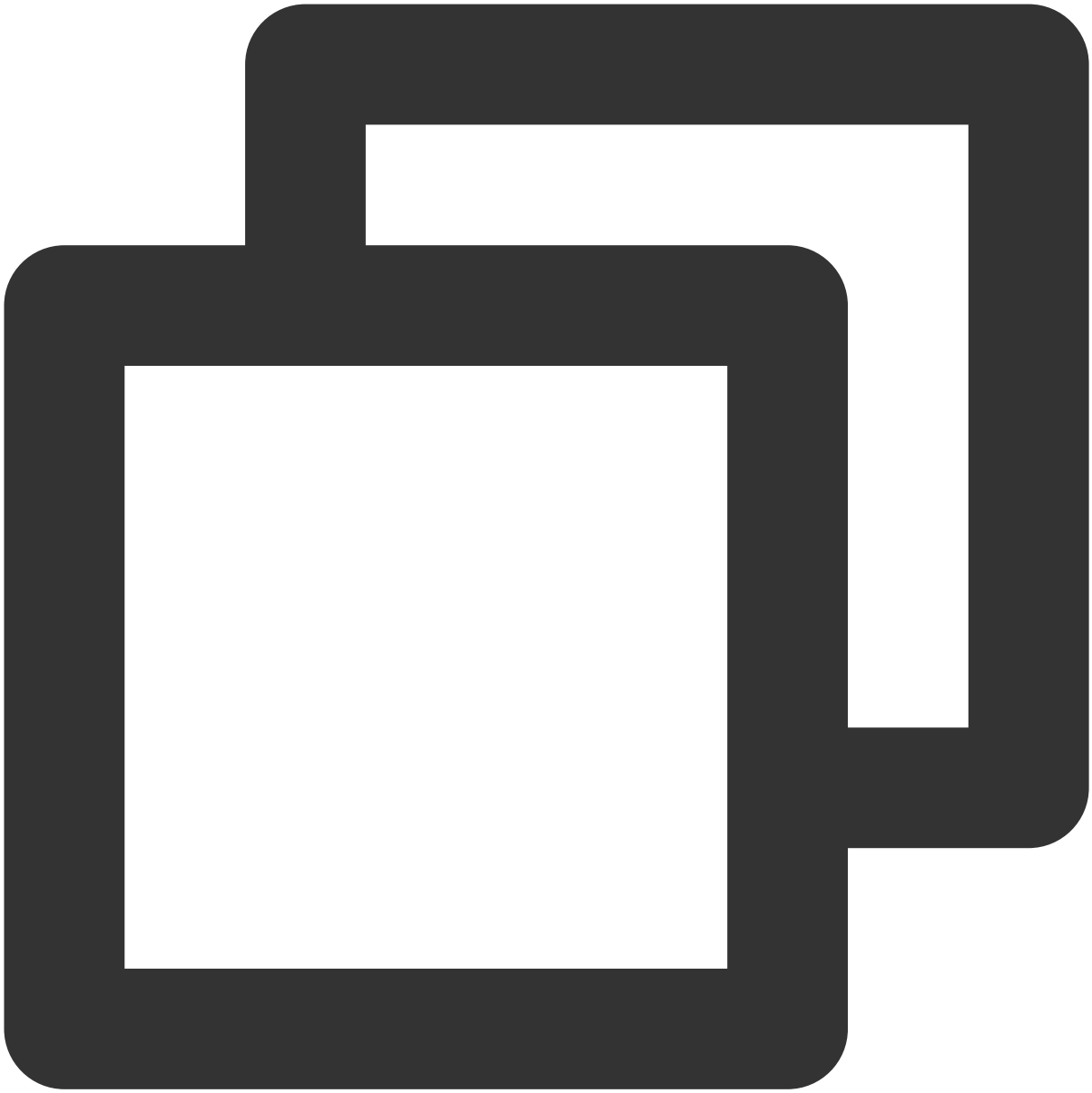


```
mLiveRoom.createRoom(roomId: 123)
```

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

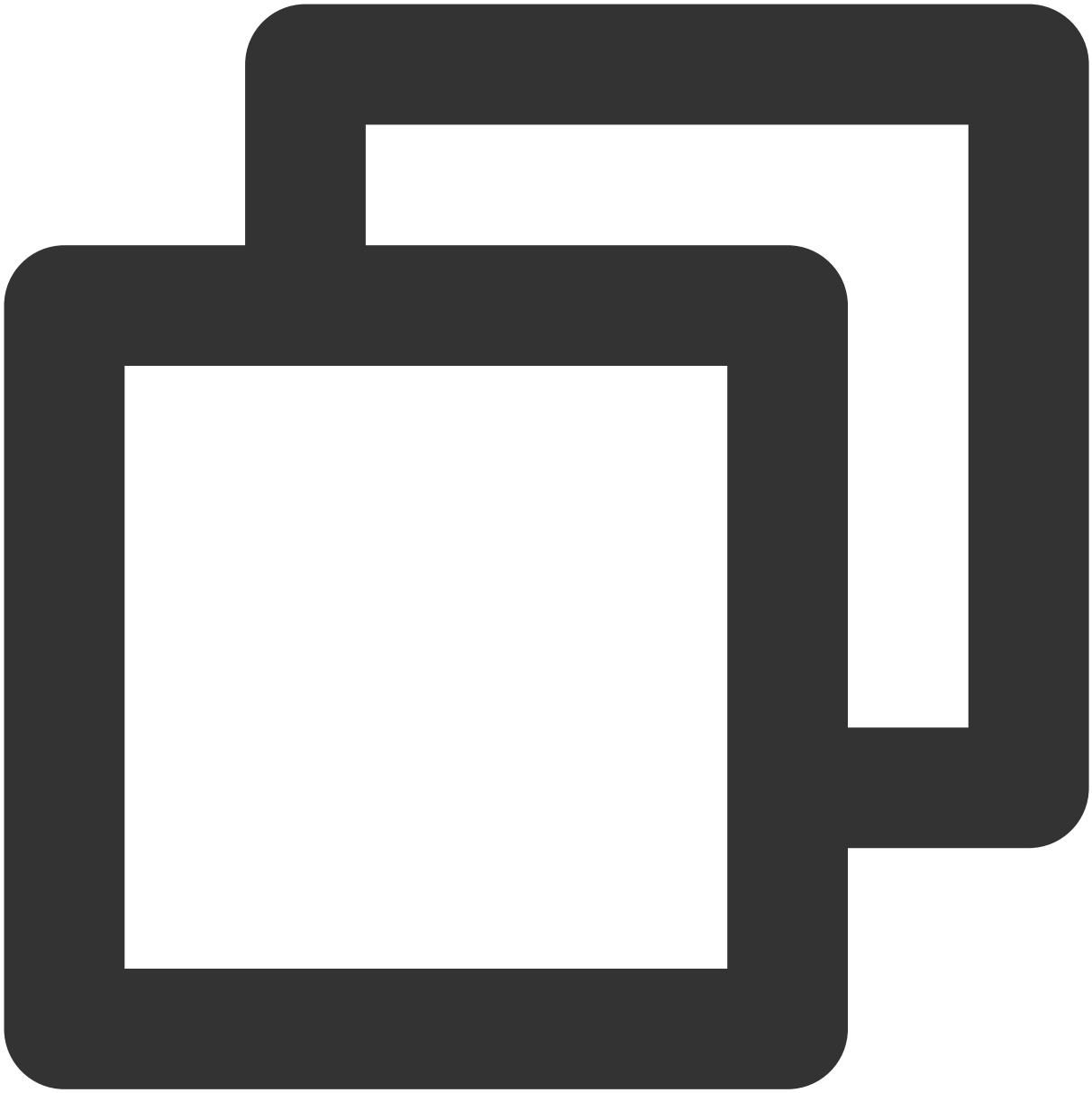
Objective-C

Swift



```
// 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
    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:play
```

```
        [[TRTCLiveRoom shareInstance] startPublishWithStreamID:[NSString stringWithFo
    }
}];

// 2.主播端收到连麦请求
#pragma mark - TRTCLiveRoomDelegate
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onRequestJoinAnchor:(TRTCLiveUser
    // 同意对方的连麦请求
    [[TRTCLiveRoom shareInstance] responseJoinAnchor:user.userId agree:YES reason:@
}

- (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
}
```



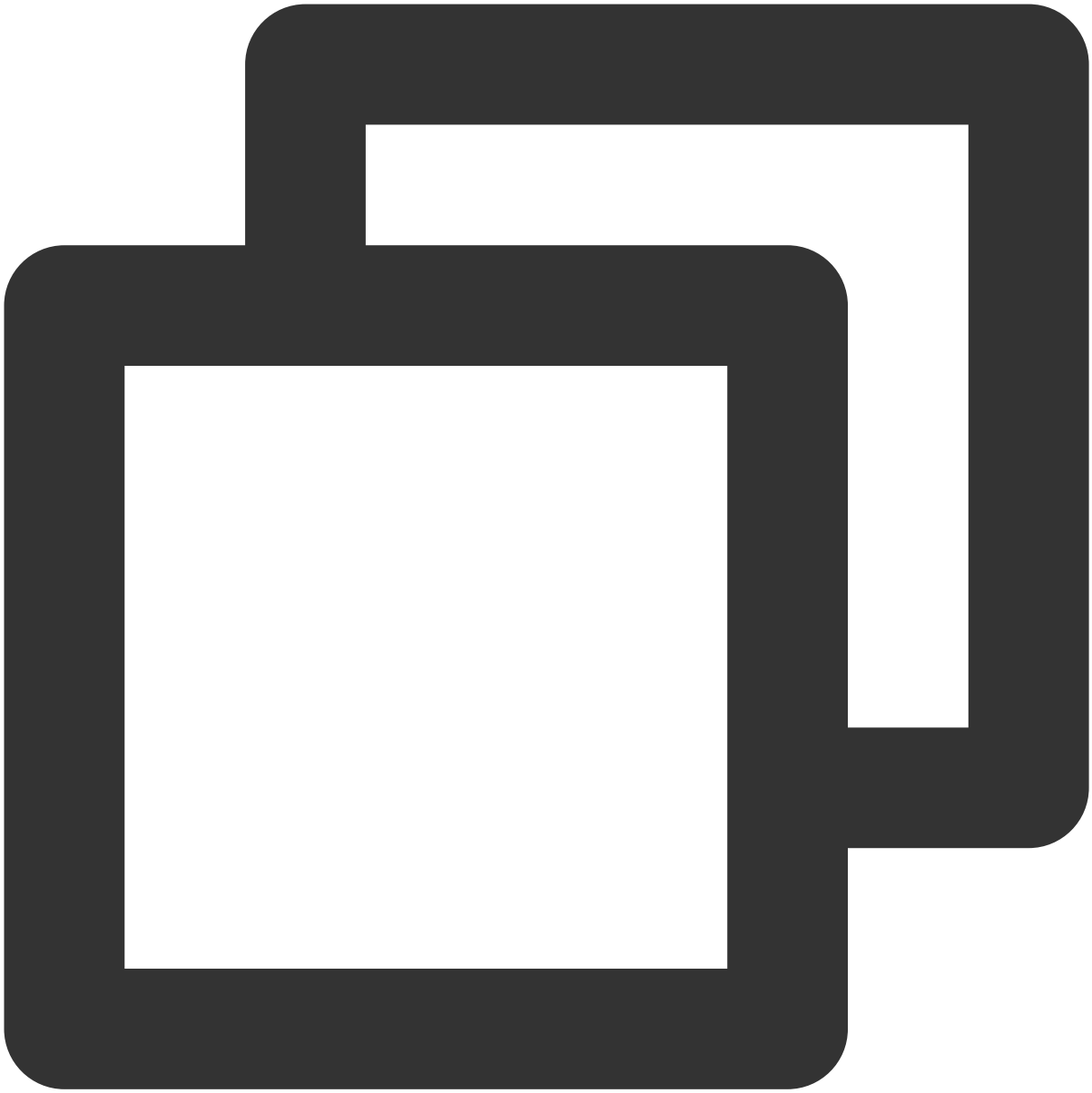
```
// 1.观众端发起连麦请求
TRTCLiveRoom.shareInstance().delegate = self
let mSelfUserId = "1314"
TRTCLiveRoom.shareInstance().requestJoinAnchor(reason: mSelfUserId + "请求和您连麦",
    guard let self = self else { return }
  if agree {
        // 主播接受了观众的请求
        let playView = UIView()
        self.view.addSubView(playView)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

```
            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: TRTCL
        // 同意对方的连麦请求
        TRTCLiveRoom.shareInstance().responseRoomPK(userID: user.userId, agree: tru
    }

    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

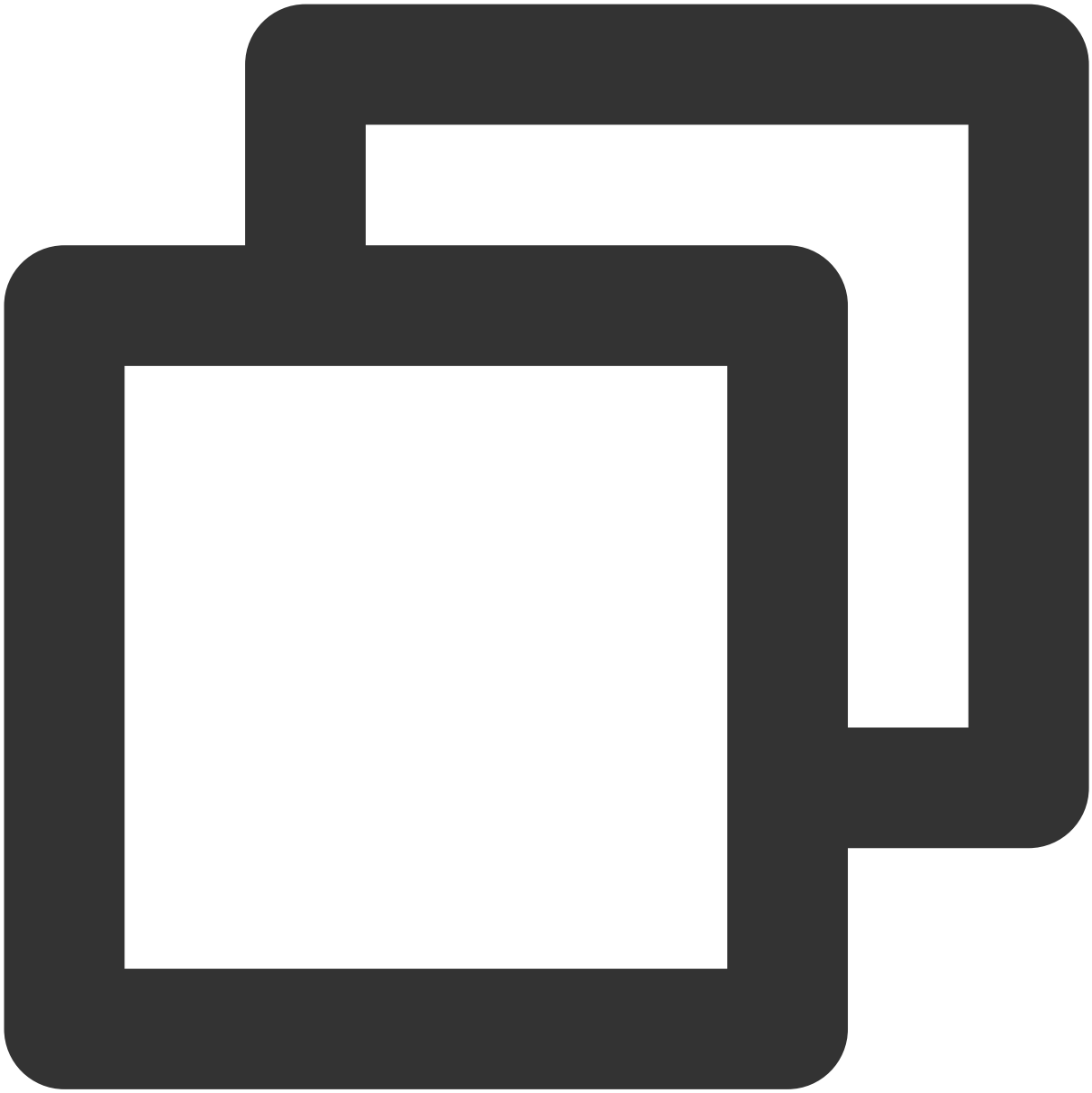
Objective-C

Swift



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

// 主播 A
// 主播 A 向 主播 B 发起PK请求
[[TRTCLiveRoom shareInstance] requestRoomPKWithRoomID:543321 userID:@"roomB userId"
    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
    // 3.主播 B 回复主播 A 接受请求
    [[TRTCLiveRoom shareInstance] responseRoomPKWithUserID:user.userId agree:YES re
}

- (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 callbac
}
```



```
// 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",
     guard let self = self else { return }
    if agreed {
        // 用户B接受
```

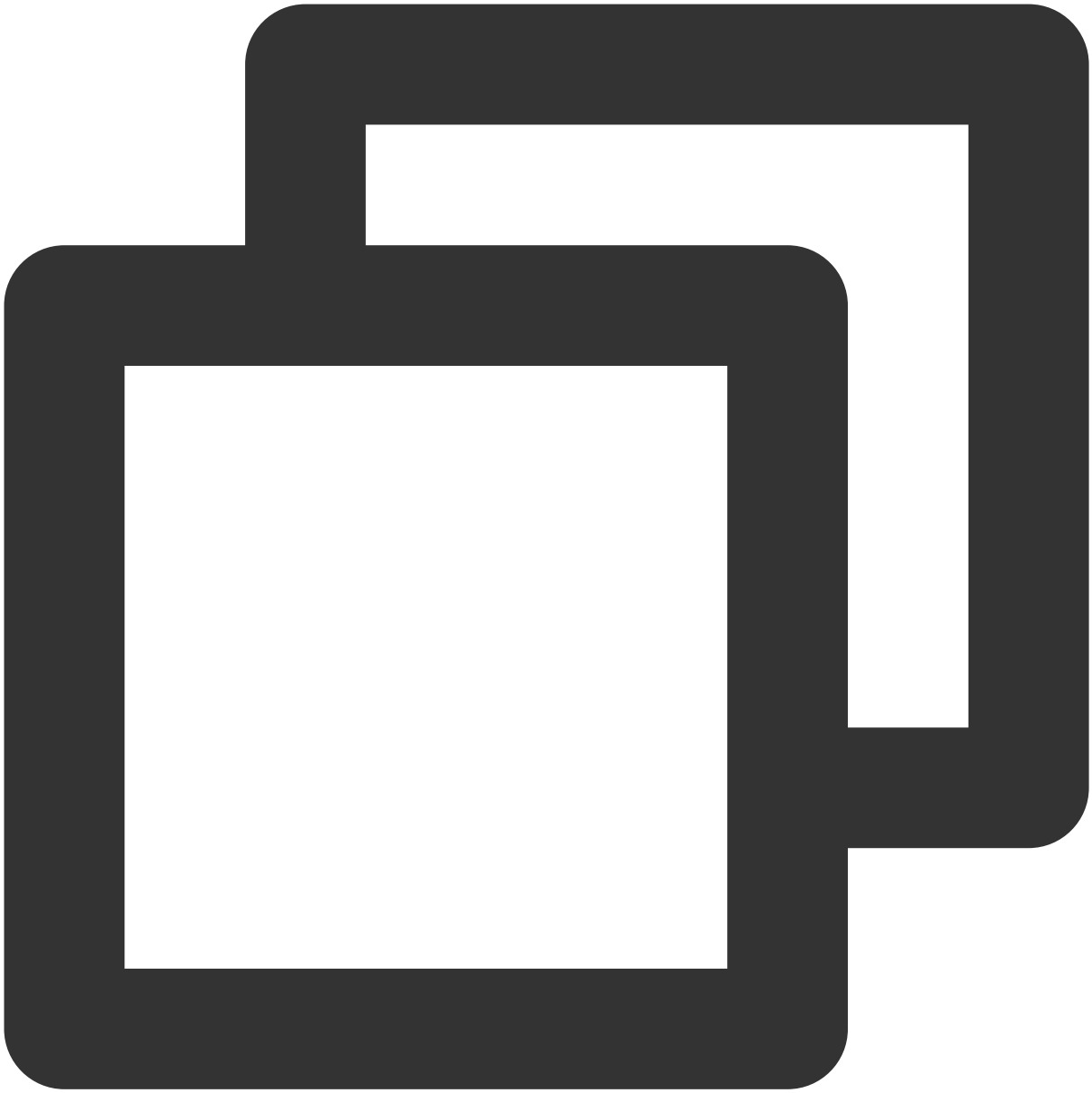
```
    } else {
        // 用户B拒绝
    }
}

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

    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）

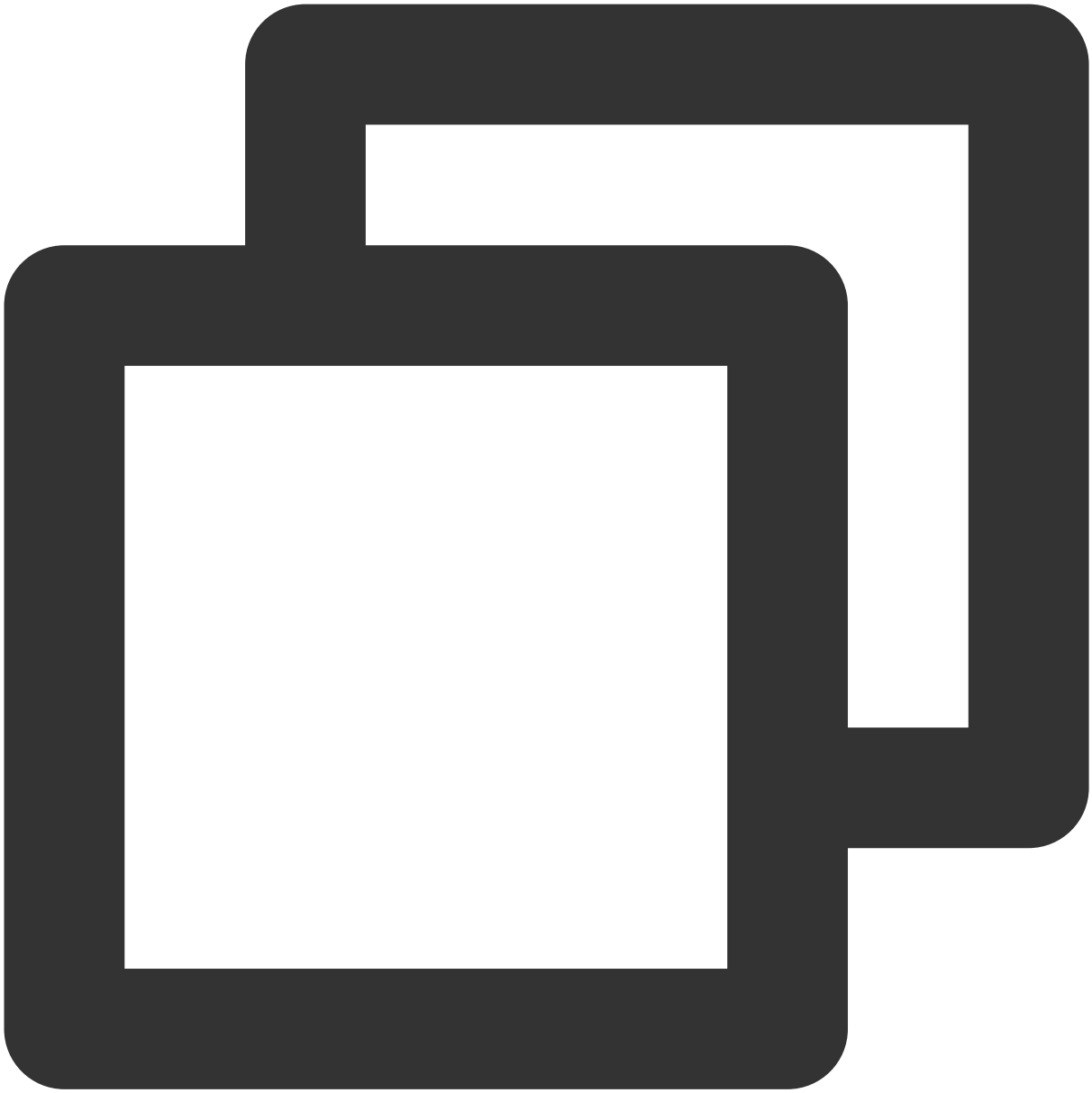
TUILiveRoom 美颜使用了 腾讯特效 SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License 申请指引。

Objective-C

Swift



```
@import TUIBeauty;

- (void)setXMagicLicense {
    [[TUIBeautyView getBeautyService] setLicenseUrl:@"XMagicLicenseURL" key:@"XMagi

    }];
}
```



```
import TUIBeauty

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

交流与反馈

如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

集成 TUIPusher&TUIPlayer (Web)

最近更新时间：2022-09-06 14:22:23

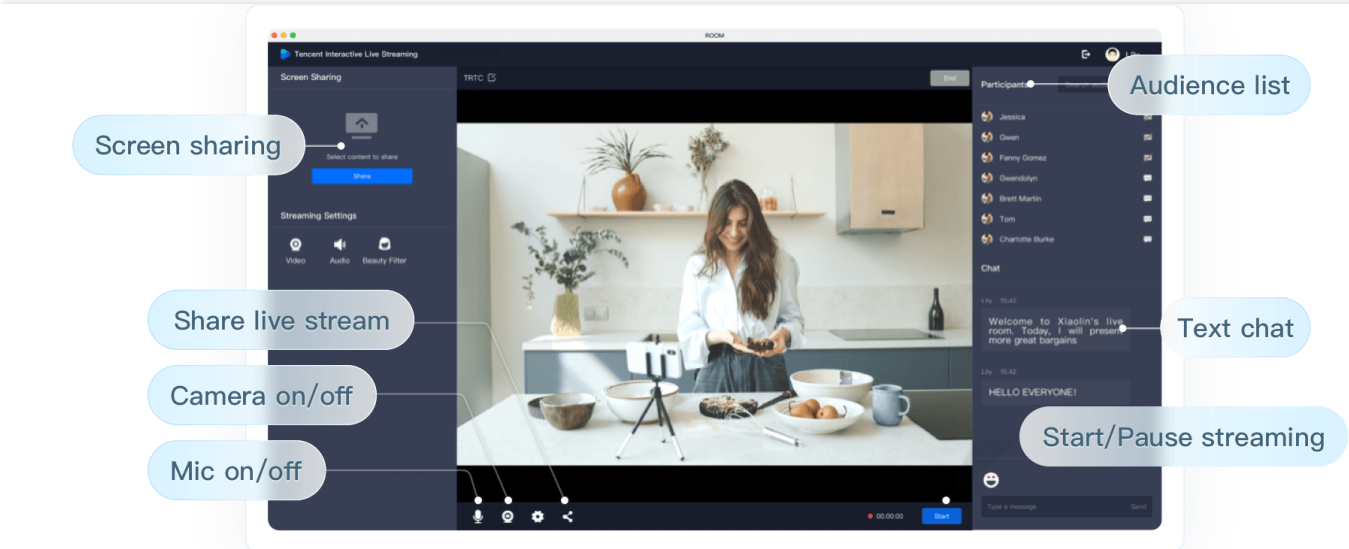
组件介绍

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

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

TUIPusher & TUIPlayer 的优势：

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



快速体验

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

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

TUIPusher 推流组件功能介绍

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

TUIPlayer 拉流组件功能介绍

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

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

组件集成

步骤一：开通腾讯云服务

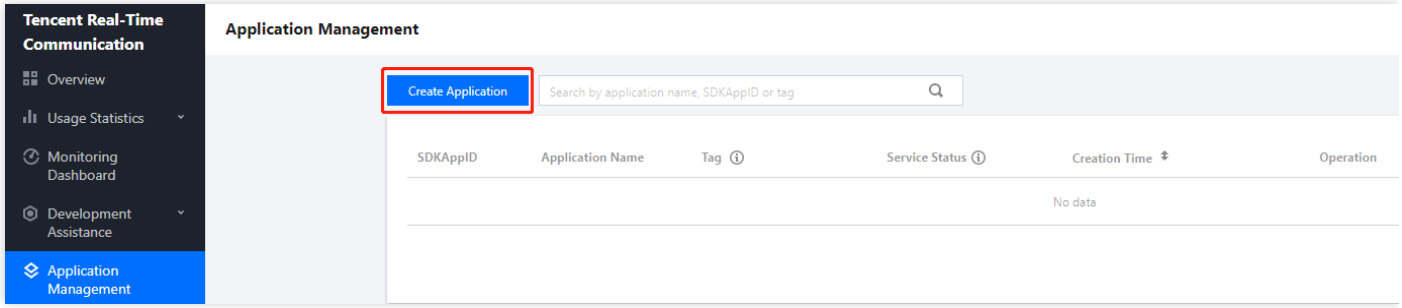
> 注意：

- TUIPusher & TUIPlayer 基于腾讯云实时音视频和即时通信服务进行开发。实时音视频 TRTC 应用与 即时通信 IM 应用的 SDKAppID 一致，才能复用账号与鉴权。
- 即时通信 IM 应用针对文本消息，提供基础版本的 安全打击 能力，如果希望使用自定义不雅词功能，可以单击 **升级** 。
- 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
- 方式2：基于即时通信 IM

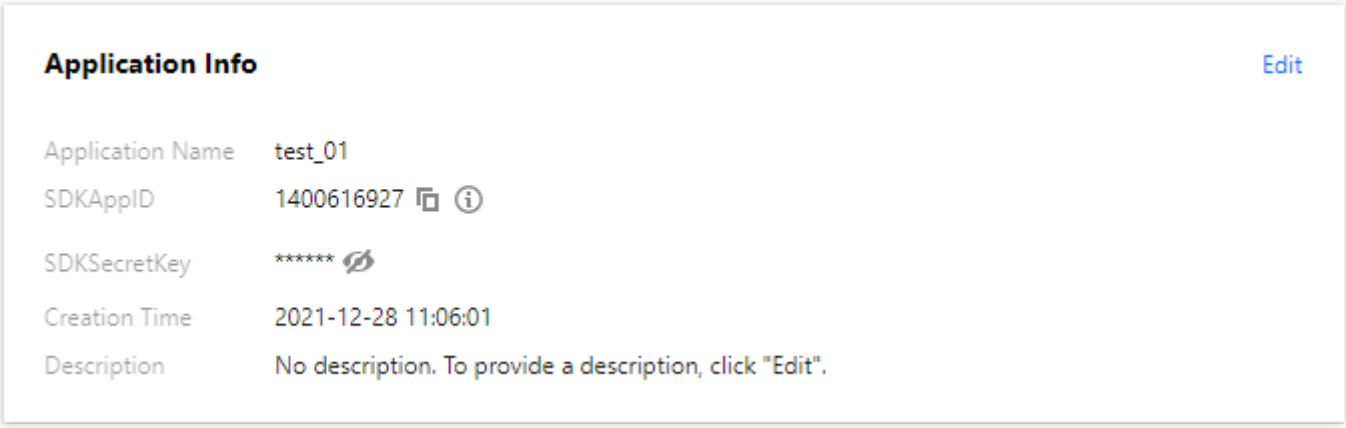
步骤1：创建实时音视频 TRTC 应用

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

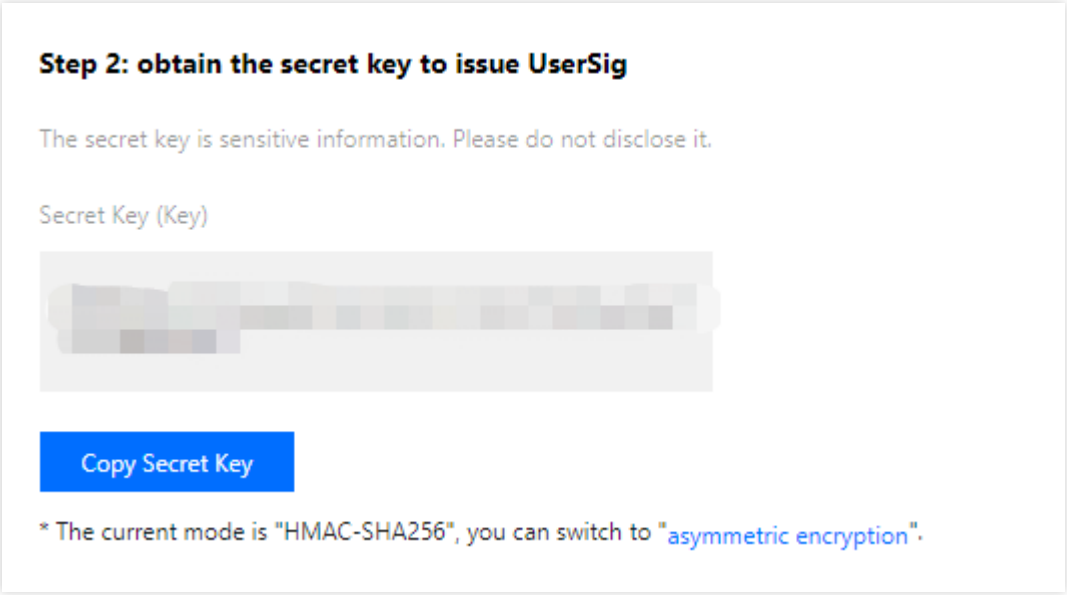


步骤2: 获取 TRTC 密钥信息

1. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。



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



说明

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

步骤二：项目准备

1. 在 GitHub 下载 TUIPusher & TUIPlayer 代码。

2. 为 TUIPusher & TUIPlayer 安装依赖。

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

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

4. 本地开发环境运行 TUIPusher & TUIPlayer。

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

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

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

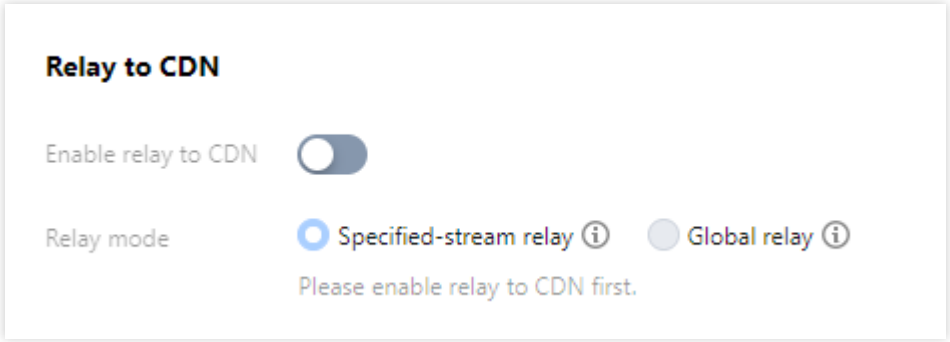
注意：

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

步骤三：旁路直播

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

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



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

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

步骤四：生产环境应用

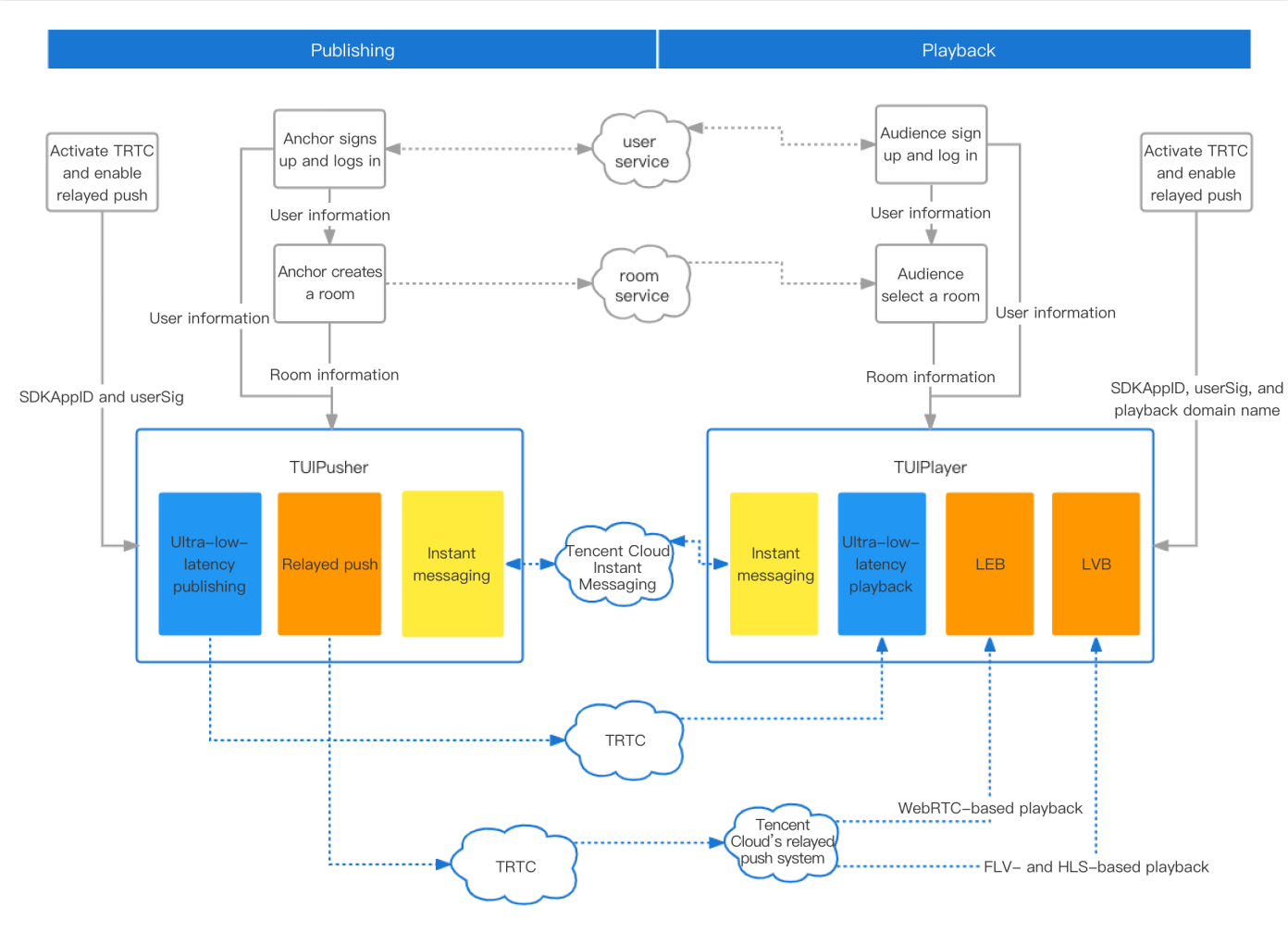
当您将 TUIPusher & TUIPlayer 用于生产应用时，在接入 TUIPusher & TUIPlayer 之外，您需要：

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

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

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

有功能。详细业务流程参见下图：



相关问题

Web 端如何实现美颜功能？

请参见 开启美颜。

Web 端如何实现屏幕共享？

请参见 屏幕分享。

Web 端如何实现云端录制?

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

Web 端如何实现推流到 CDN ？

Web 端推流到 CDN 请参见 实现推流到 CDN。

Web 端如何实现快直播拉流？

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

注意事项

平台支持

操作系统	浏览器类型	浏览器最低版本要求	TUIPlayer	TUIPusher	TUIPusher 屏幕分享
Mac OS	桌面版 Safari 浏览器	11+	支持	支持	支持（需要 Safari13+ 版本）
Mac OS	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Mac OS	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Mac OS	桌面版 Edge 浏览器	80+	支持	支持	支持
Mac OS	桌面版微信内嵌网页	-	支持	不支持	不支持
Mac OS	桌面版企业微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Windows	桌面版 QQ 浏览器（极速内核）	10.4+	支持	支持	不支持
Windows	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Windows	桌面版 Edge 浏览器	80+	支持	支持	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
Windows	桌面版微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版企业微信内嵌网页	-	支持	不支持	不支持
iOS	微信内嵌浏览器	-	支持	不支持	不支持
iOS	企业微信内嵌浏览器	-	支持	不支持	不支持
iOS	移动版 Safari 浏览器	-	支持	不支持	不支持
iOS	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	微信内嵌浏览器	-	支持	不支持	不支持
Android	企业微信浏览器	-	支持	不支持	不支持
Android	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	移动版 QQ 浏览器	-	支持	不支持	不支持
Android	移动版 Firefox 浏览器	-	支持	不支持	不支持
Android	移动端 UC 浏览器	-	支持（仅支持标准直播观看）	不支持	不支持

域名要求

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

> 注意：
> 本地开发可以通过 `http://localhost` 协议进行访问。

URL 域名及协议支持情况请参考如下表格：

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

防火墙限制

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

结语

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

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

TUILiveRoom API 查询

TRTCLiveRoom API (iOS)

最近更新时间：2022-07-22 15:13:44

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（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。

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.tencentcloud.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://www.tencentcloud.com/document/product/647/35166)。
/// - config: 全局配置信息，请在登录时初始化，登录之后不可变更。 isAttachedTUIKit 项目中是否引入并使用TUIKit
```

```
/// - callback: 登录回调，成功时 code 为0。
/// - Note:
/// - userSig 建议设定 7 天，能够有效规避 usersign 过期导致的 IM 收发消息失败、TRTC 连麦失败等情况
- (void)loginWithSdkAppID:(int)sdkAppID
userID:(NSString *)userID
userSig:(NSString *)userSig
config:(TRTCLiveRoomConfig *)config
callback:(Callback _Nullable)callback
NS_SWIFT_NAME(login(sdkAppID:userID:userSig:config:callback:));
```

参数如下表所示：

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

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
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。
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

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
/// 切换前后摄像头
- (void)switchCamera;
```

setMirror

设置是否镜像展示。

```
/// 设置是否镜像展示
/// - Parameter isMirror: 开启/关闭镜像。
- (void)setMirror:(BOOL)isMirror
NS_SWIFT_NAME(setMirror(isMirror:));
```

参数如下表所示：

参数	类型	含义
isMirror	Bool	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
/// 静音本地音频。
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteLocalAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteLocalAudio(isMuted:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
/// 静音远端音频
/// - Parameters:
/// - userID: 远端的用户ID。
/// - isMuted: true：开启静音；false：关闭静音。
- (void)muteRemoteAudioWithUserID:(NSString *)userID isMuted:(BOOL)isMuted
NS_SWIFT_NAME(muteRemoteAudio(userID:isMuted:));
```

参数如下表所示：

参数	类型	含义
userID	String	远端的用户 ID。
isMuted	Bool	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
/// 静音所有远端音频
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteAllRemoteAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteAllRemoteAudio(_:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

setAudioQuality

设置音频质量

```
/// 设置音频质量，支持的值为1 2 3，代表低中高
/// - Parameter quality 音频质量
- (void)setAudioQuality:(NSInteger)quality
NS_SWIFT_NAME(setAudioiQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	1：语音；2：标准； 3：音乐。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
/// 获取音效管理对象
- (TXAudioEffectManager *)getAudioEffectManager;
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。

```
/* 获取美颜管理对象 TXBeautyManager
*
* 通过美颜管理，您可以使用以下功能：
* - 设置"美颜风格"、“美白”、“红润”、“大眼”、“瘦脸”、“v脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
* - 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”
* - 设置人脸挂件（素材）等动态效果
* - 添加美妆
* - 进行手势识别
*/
- (TXBeautyManager *)getBeautyManager;
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
/// 发送文本消息，房间内所有成员都可见
/// - Parameters:
/// - message: 文本消息。
/// - callback: 发送回调。
- (void)sendRoomTextMsg:(NSString *)message callback:(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
/// 发送自定义消息
/// - Parameters:
/// - command: 命令字，由开发者自定义，主要用于区分不同消息类型
/// - message: 本文消息。
/// - callback: 发送回调。
- (void)sendRoomCustomMsgWithCommand:(NSString *)command message:(NSString *)mess
age 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。

参数	类型	含义
code	Int	错误码。
message	String?	错误信息。

onWarning

警告回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onWarning:(NSInteger)code
message:(NSString *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onWarning:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码 TRTCWarningCode。
message	String?	警告信息。

onDebugLog

Log 回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onDebugLog:(NSString *)log
NS_SWIFT_NAME(trtcLiveRoom(_:onDebugLog:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
log	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRoomDestroy:(NSString *)roomID
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomDestroy:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
roomID	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRoomInfoChange:(TRTCLiveRoomInfo *)info
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomInfoChange:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
info	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAnchorEnter:(NSString *)userID
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	新进房用户 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAnchorExit:(NSString *)userID
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAudienceEnter:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onAudienceExit:(TRTCLiveUserInfo *)user
```

```
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRequestJoinAnchor:(TRTCLiveUserInfo *)user
reason:(NSString * _Nullable)reason
timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestJoinAnchor:reason:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	请求连麦观众信息。
reason	String?	连麦原因描述。
timeout	Double	处理请求的超时时间。

onKickoutJoinAnchor

连麦观众收到被踢出连麦的通知。连麦观众收到被主播踢除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

```
- (void)trtcLiveRoomOnKickoutJoinAnchor:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnKickoutJoinAnchor(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRequestRoomPK:(TRTCLiveUserInfo *)user
timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestRoomPK:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	发起跨房连麦的主播信息。
timeout	Double	处理请求的超时时间。

onQuitRoomPK

收到断开跨房 PK 通知。

```
- (void)trtcLiveRoomOnQuitRoomPK:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnQuitRoomPK(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRecvRoomTextMsg:(NSString *)message
fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomTextMsg:fromUser:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRecvRoomCustomMsgWithCommand:(NSString *)command
message:(NSString *)message
fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomCustomMsg:message:fromUser:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

TRTCLiveRoom API (Android)

最近更新时间：2024-03-11 10:27:03

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
requestRoomPK	主播请求跨房 PK。

responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager。

美颜滤镜相关接口函数

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

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

调试相关接口函数

API	描述
showVideoDebugLog	是否在界面中展示 debug 信息。

TRTCLiveRoomDelegate API 概览

通用事件回调

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

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	直播房间信息变更回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述
onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onKickoutJoinAnchor	连麦观众收到被踢出连麦的通知。

主播 PK 事件回调

API	描述

onRequestRoomPK	收到请求跨房 PK 通知。
onQuitRoomPK	收到断开跨房 PK 通知。

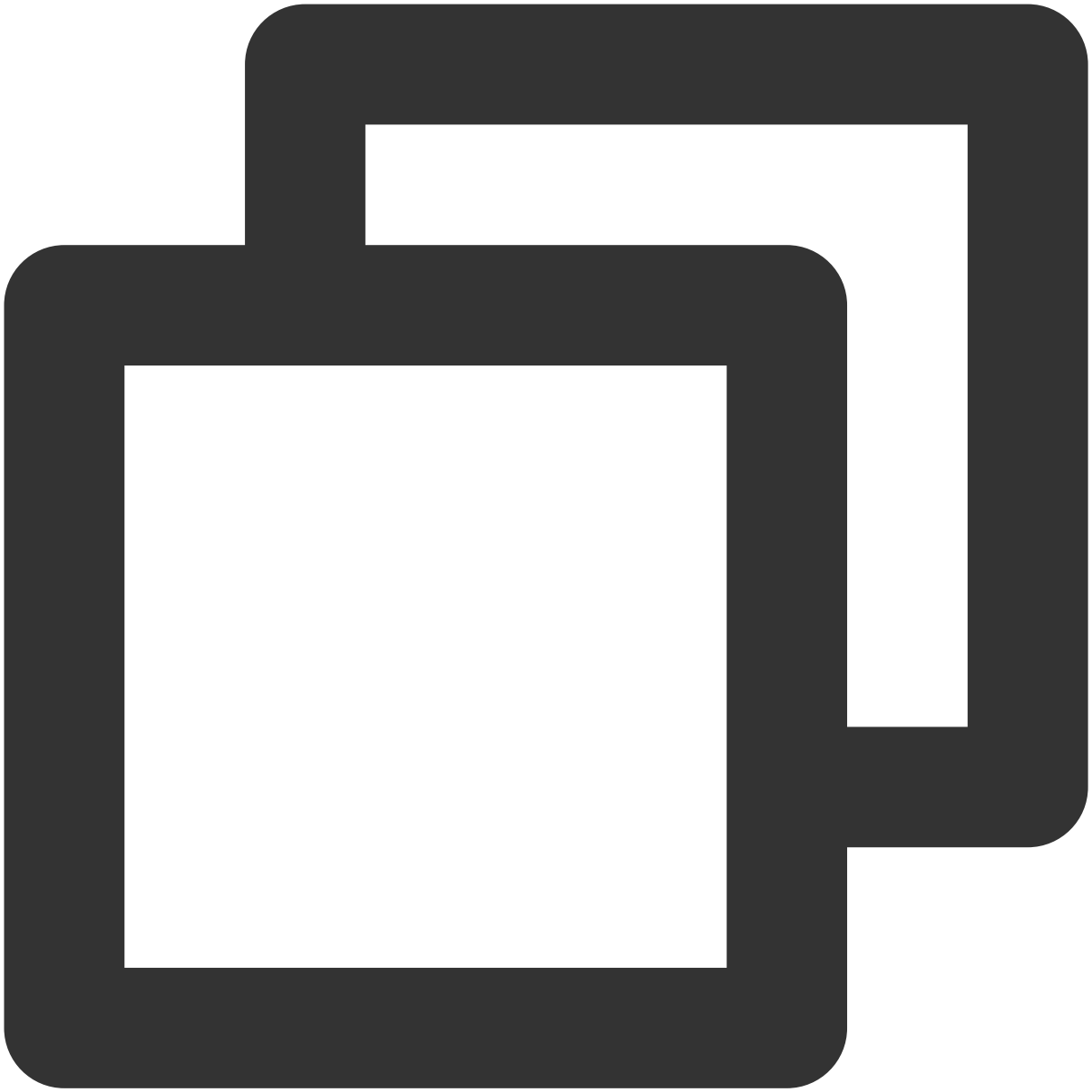
消息事件回调

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

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom 单例对象。



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

参数如下表所示：

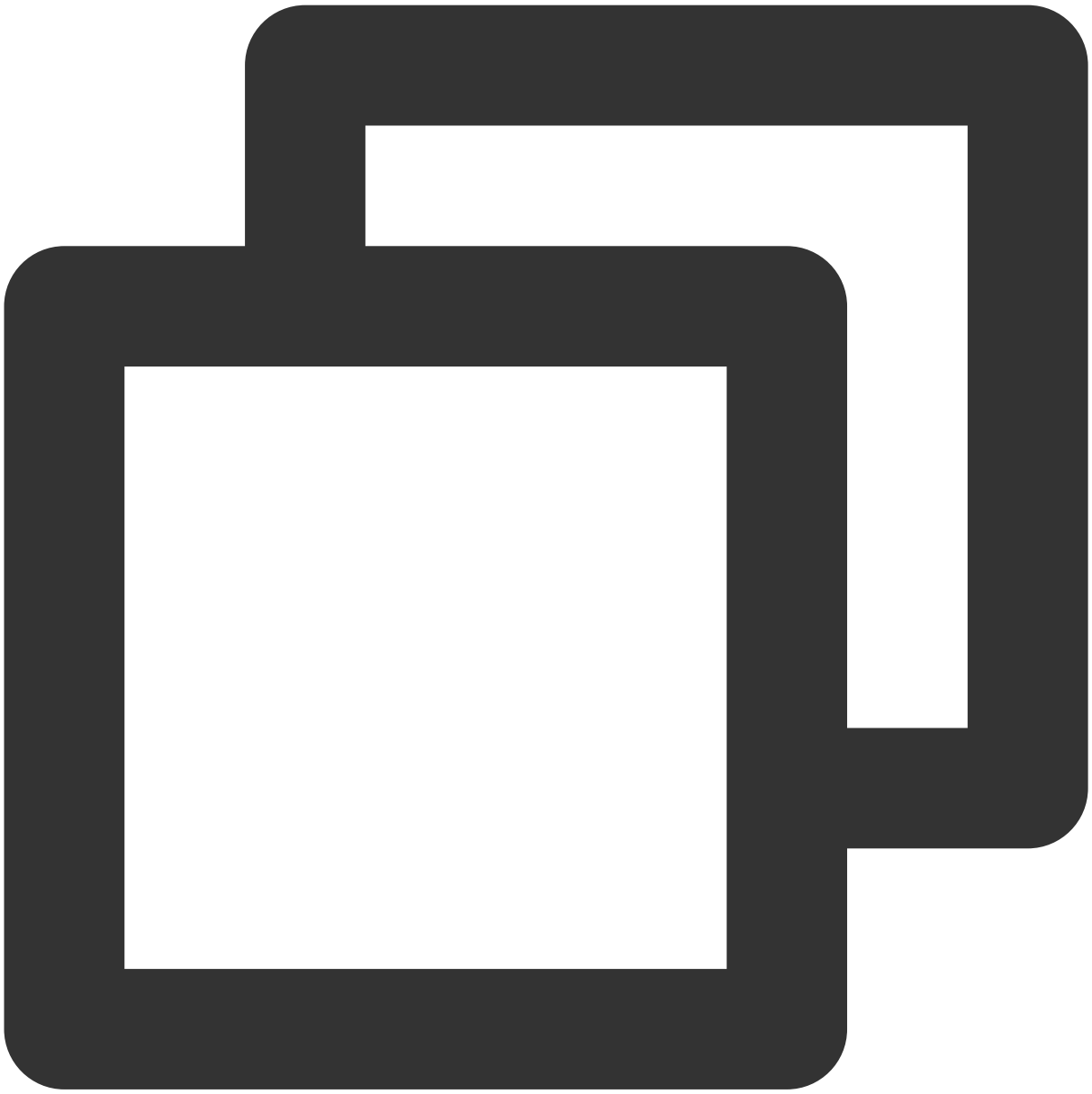
参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

说明：

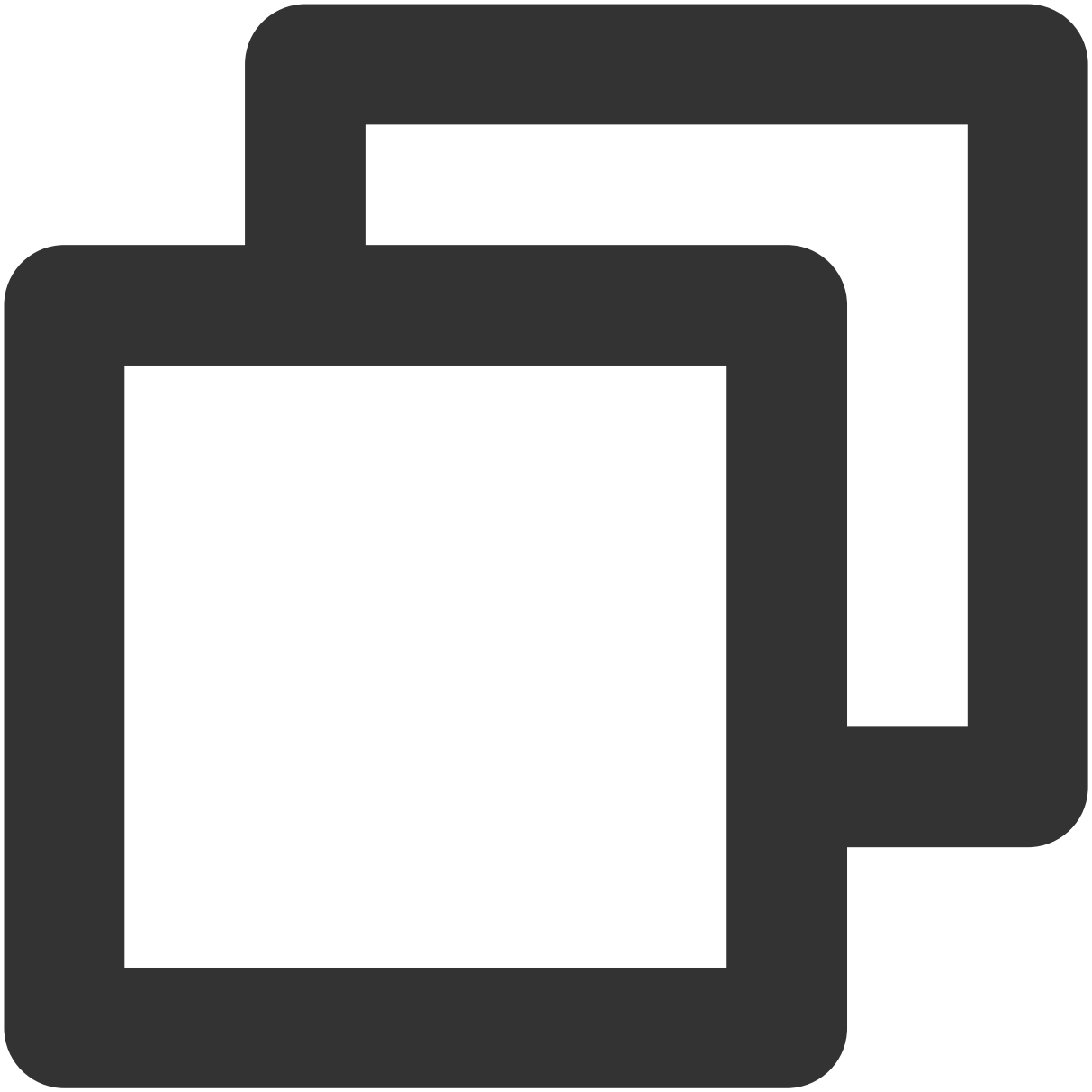
销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。



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

setDelegate

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



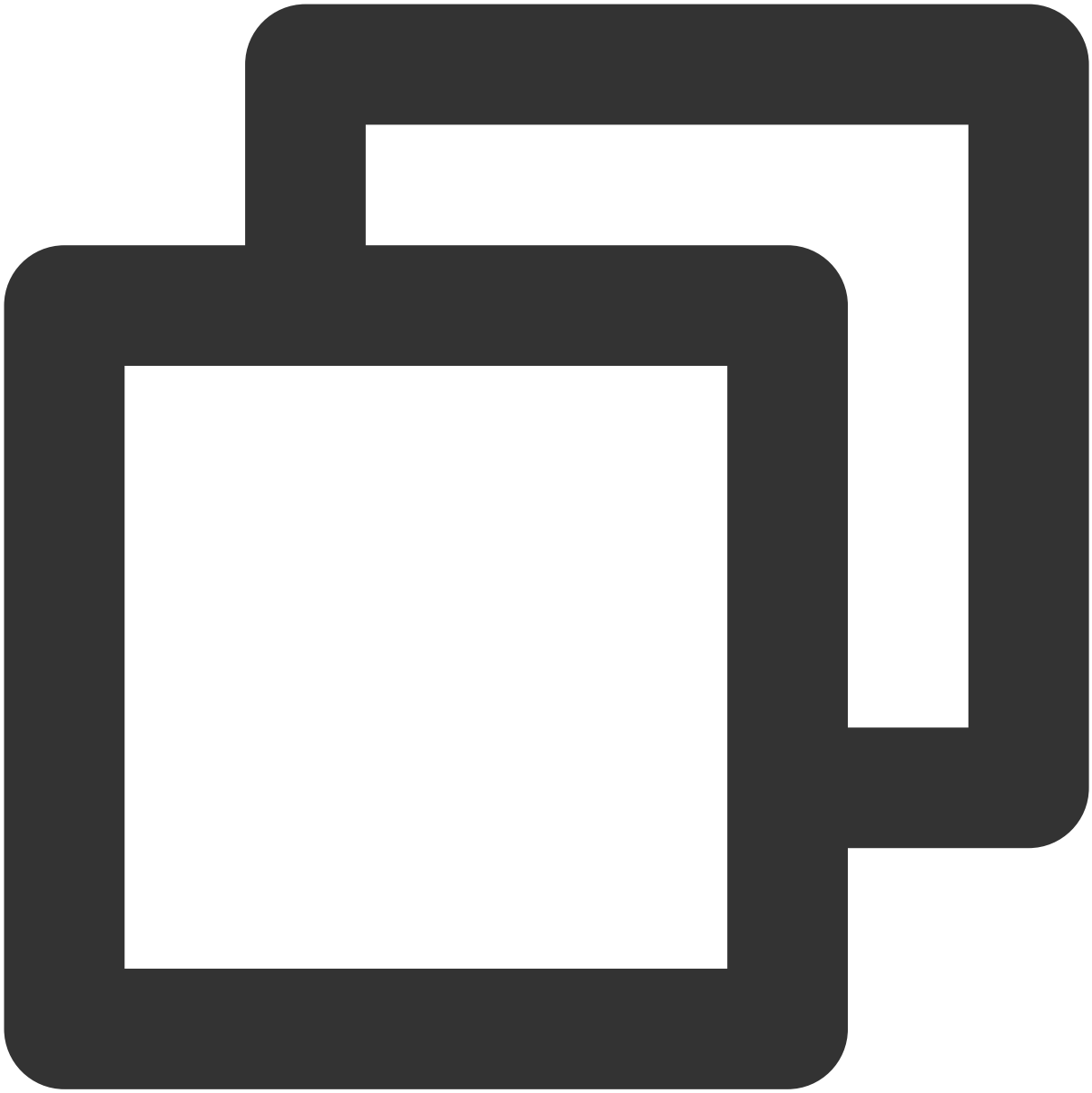
```
public abstract void setDelegate(TRTCLiveRoomDelegate delegate);
```

说明：

setDelegate 是 TRTCLiveRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。



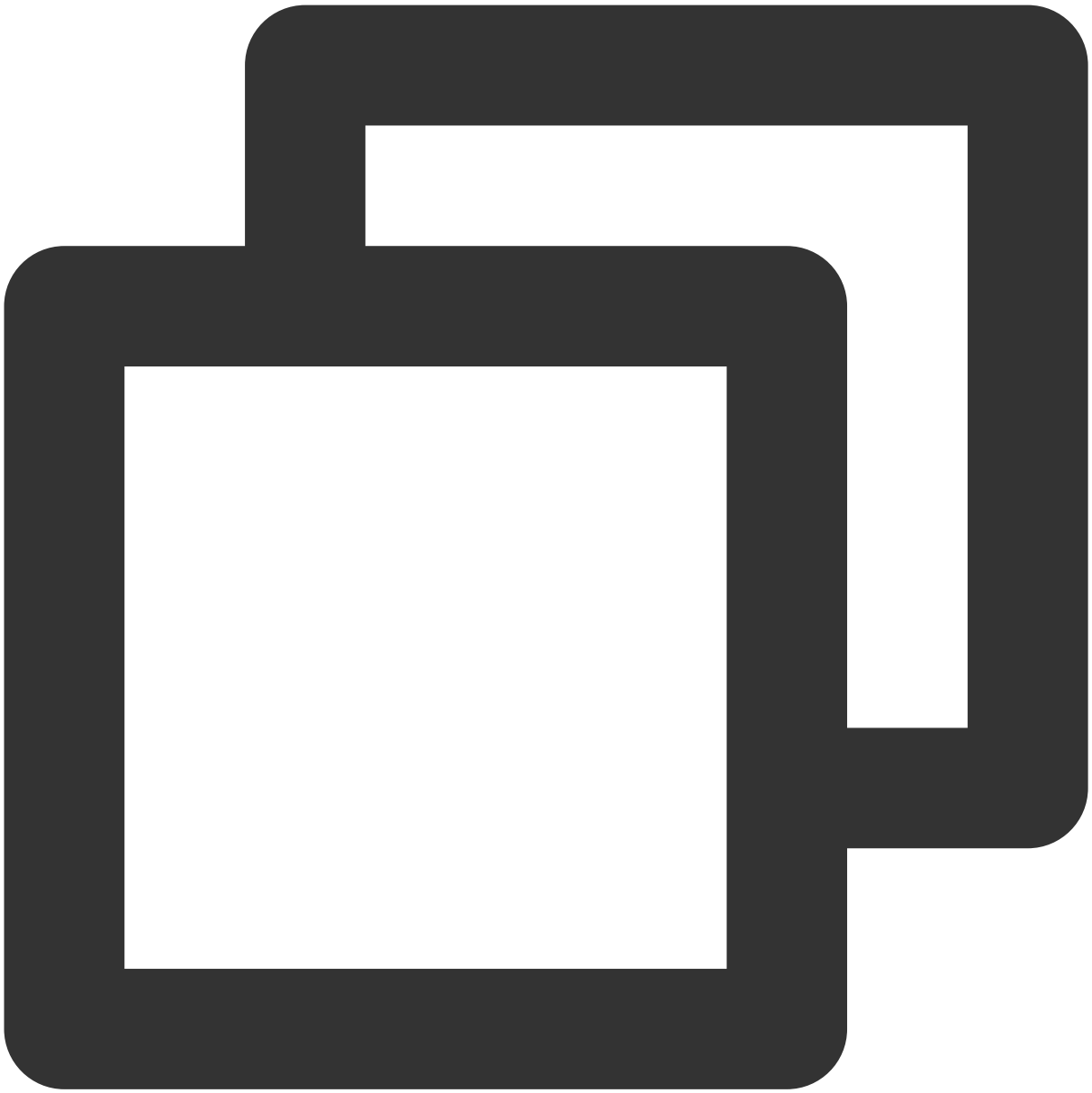
```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCLiveRoom 中的各种状态通知回调会通过该 handler 通知给您，请勿与 setDelegate 混用。

login

登录。



```
public abstract void login(int sdkAppId,
 String userId, String userSig,
 TRTCLiveRoomDef.TRTCLiveRoomConfig config,
 TRTCLiveRoomCallback.ActionCallback callback);
```

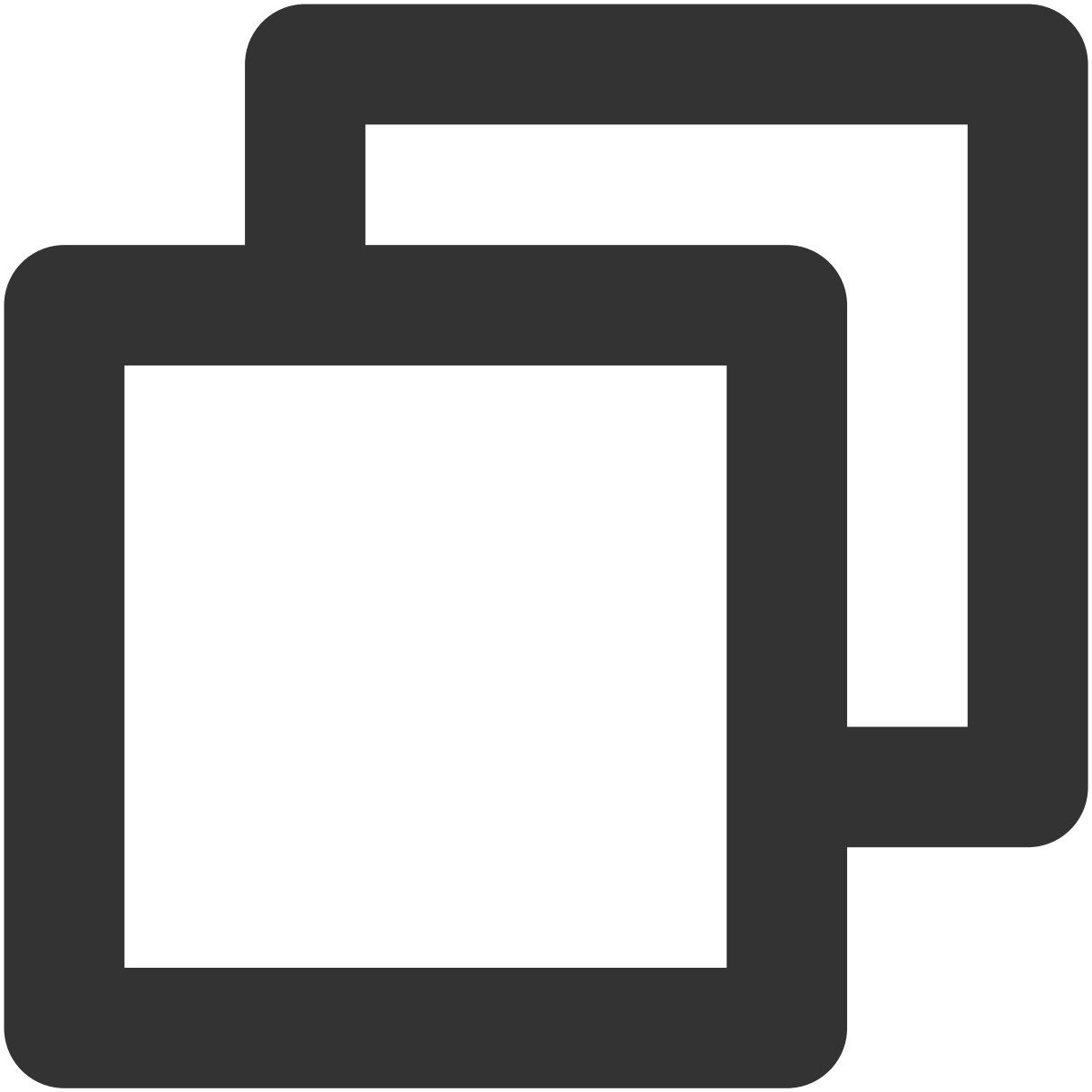
参数如下表所示：

参数	类型	含义

sdkAppId	int	您可以在实时音视频控制台 > 应用管理 > 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 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。



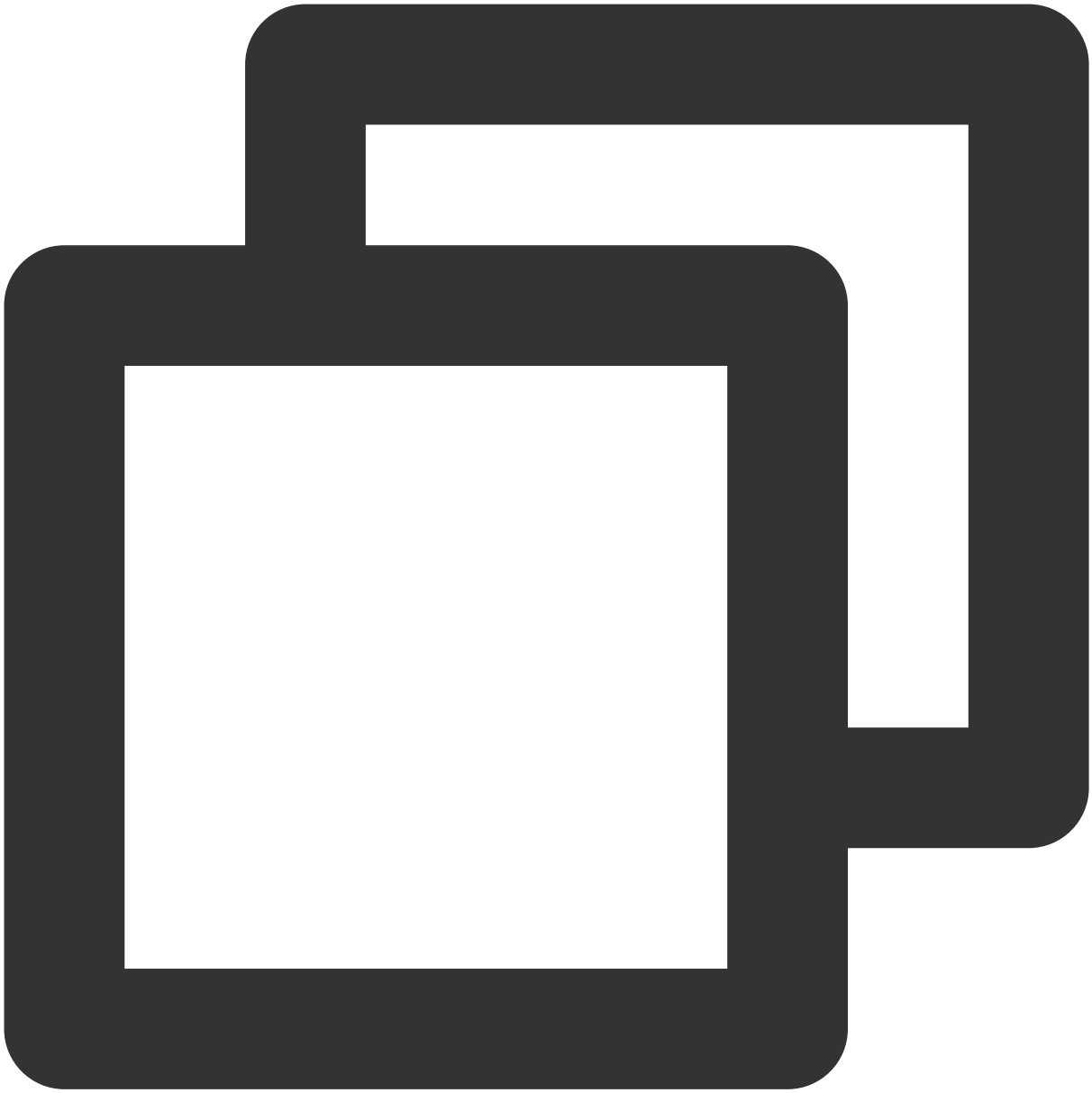
```
public abstract void logout(TRTCLiveRoomCallback.ActionCallback callback);
```

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, TRTCLiveRoom
```

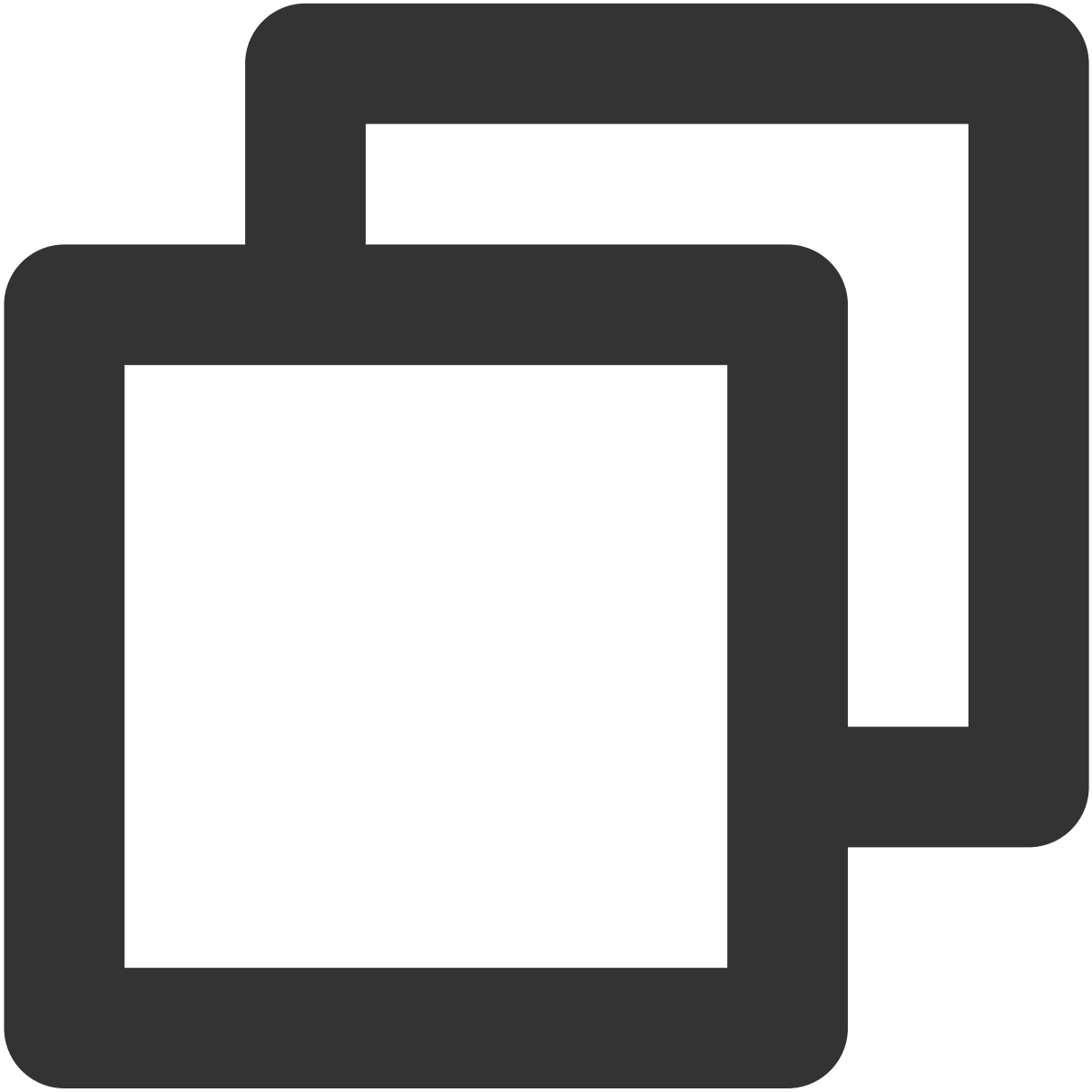
参数如下表所示：

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

房间相关接口函数

createRoom

创建房间（主播调用）。



```
public abstract void createRoom(int roomId, TRTCLiveRoomDef.TRTCCreateRoomParam roo
```

参数如下表所示：

参数	类型	含义

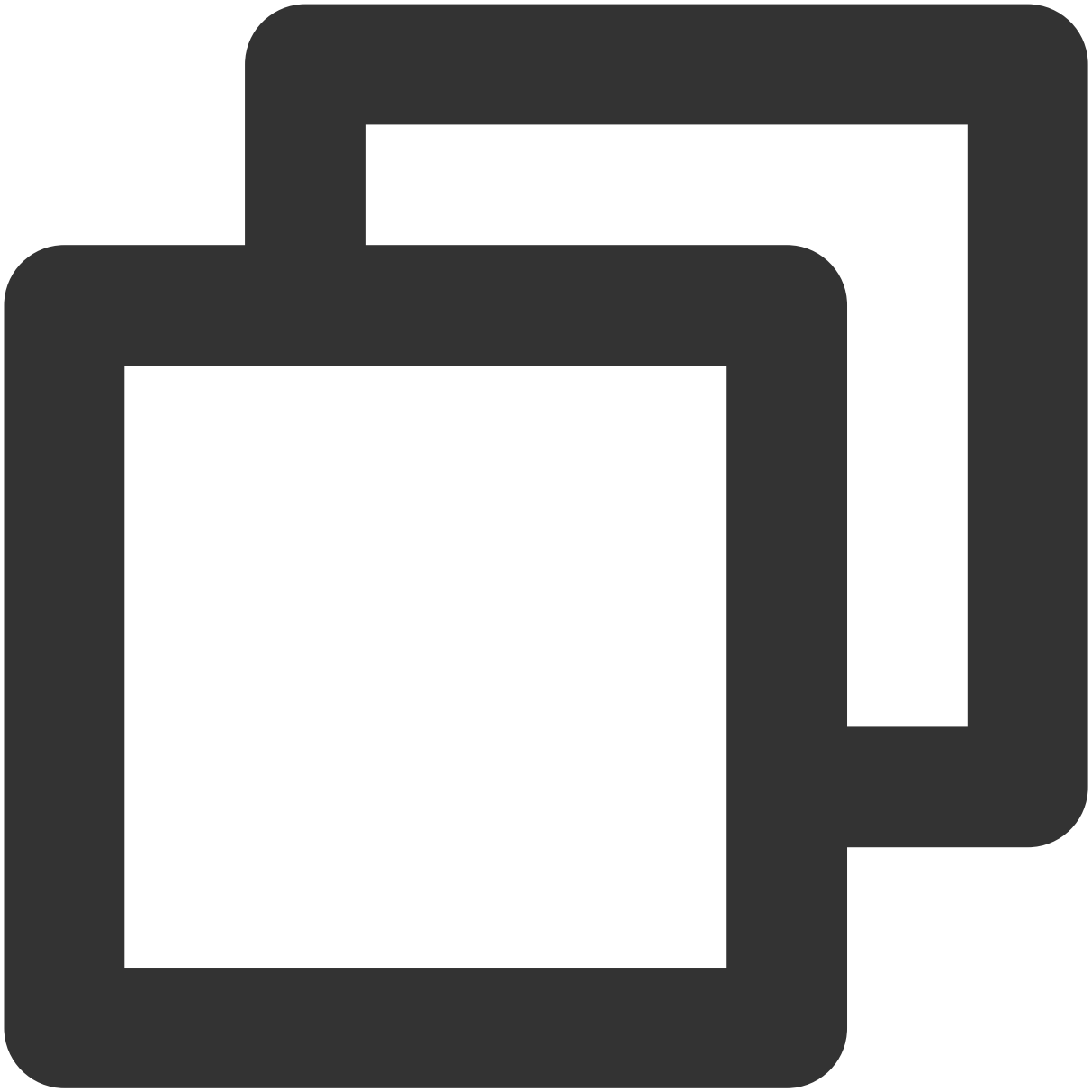
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

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。



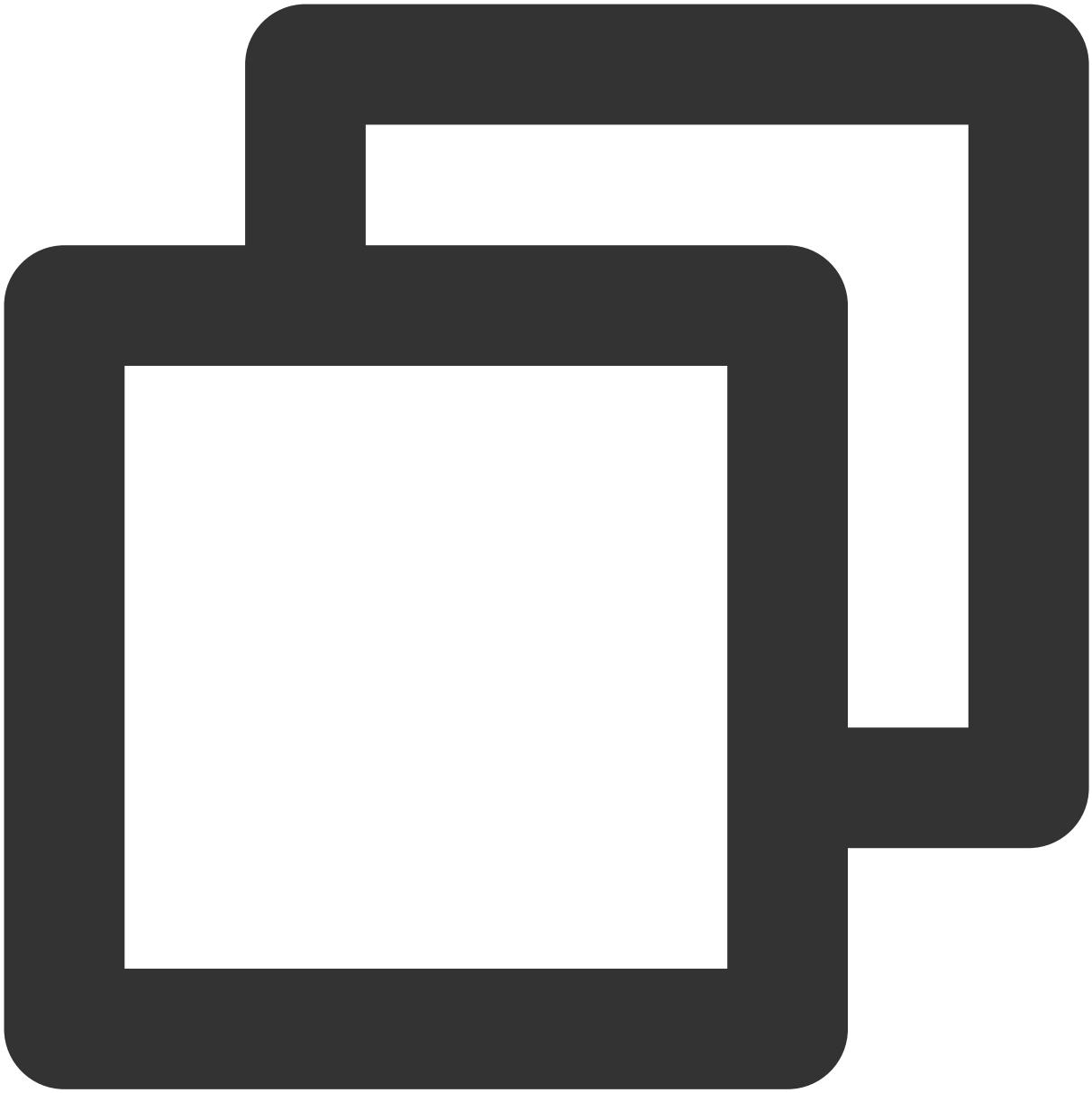
```
public abstract void destroyRoom(TRTCLiveRoomCallback.ActionCallback callback);
```

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 call
```

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：

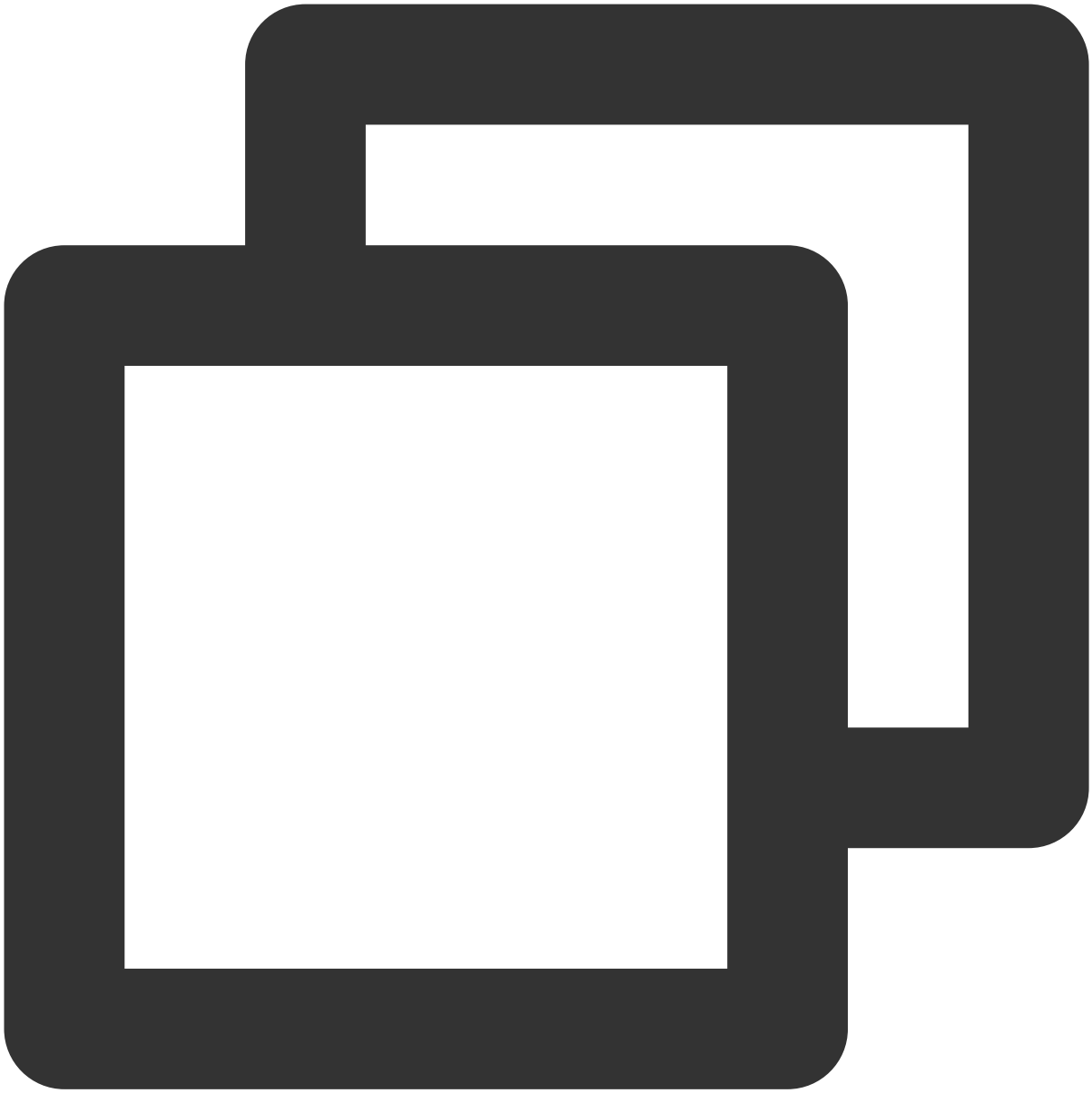
1. 【观众】向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 【观众】观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. 【观众】调用 `startPlay(userId)` 并传入主播的 userId 开始播放。

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：

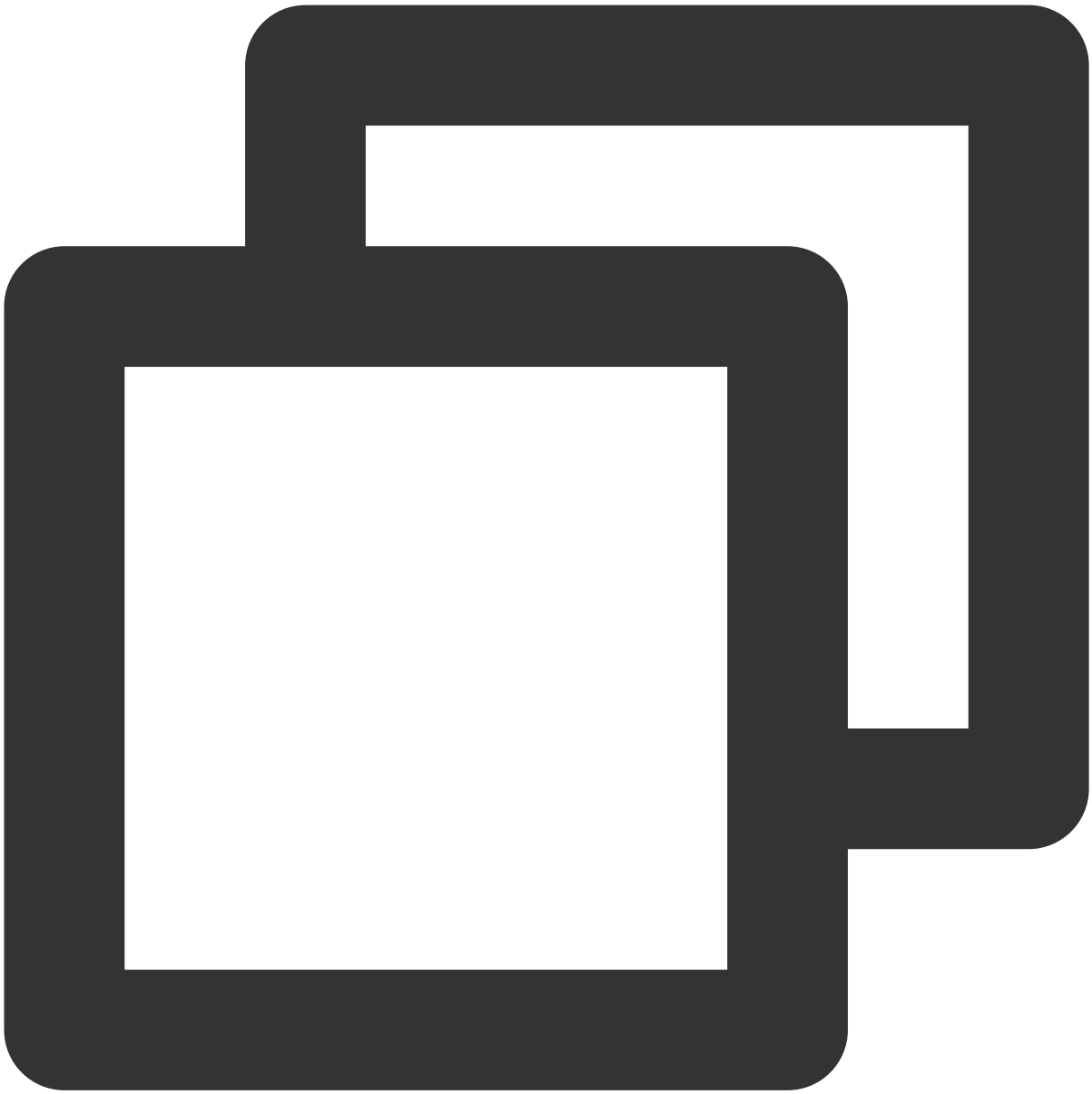
参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

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。



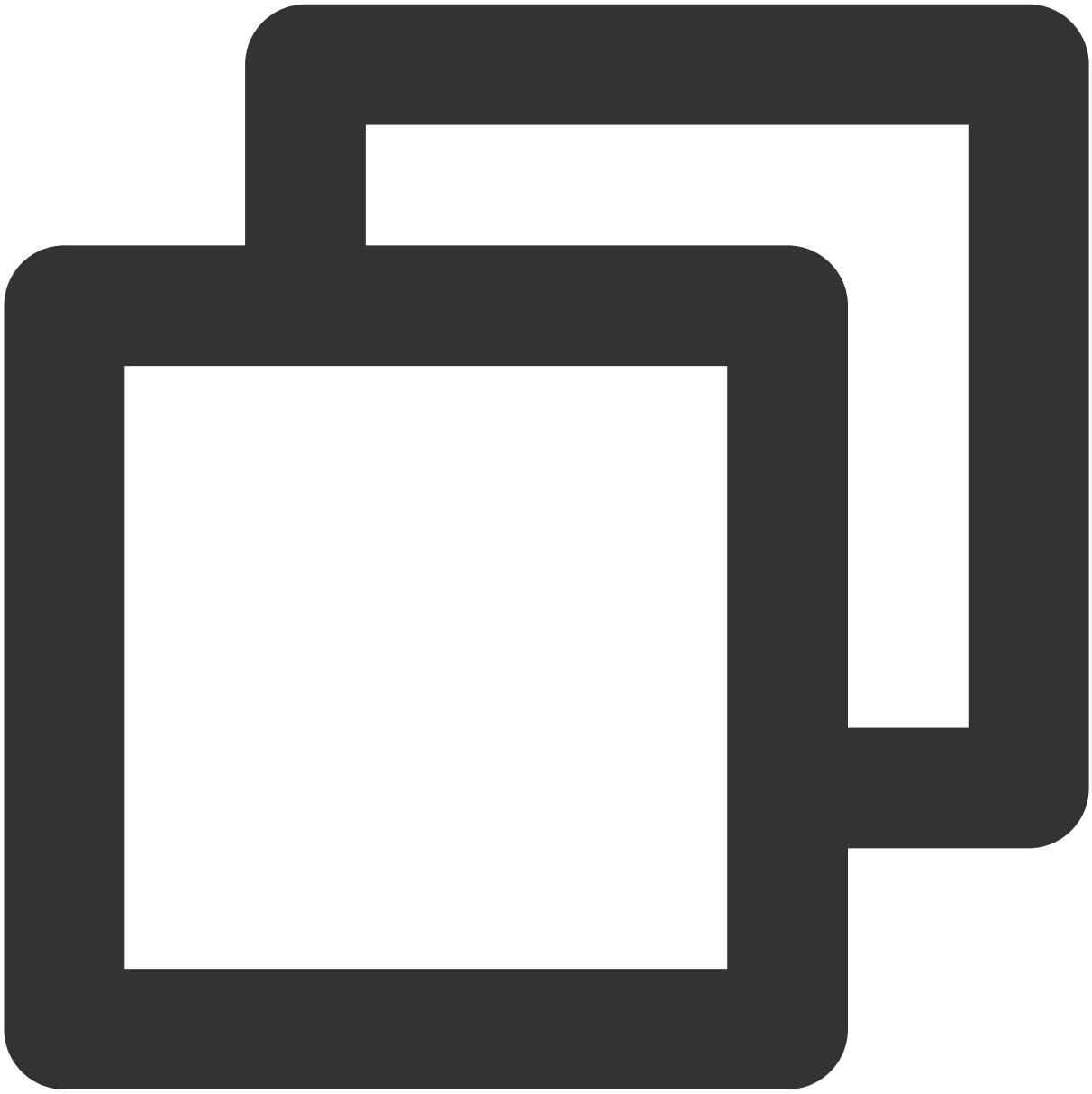
```
public abstract void getRoomInfos(List<Integer> roomIdList, TRTCLiveRoomCallback.Ro
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getAnchorList

获取房间内所有的主播列表，`enterRoom()` 成功后调用才有效。



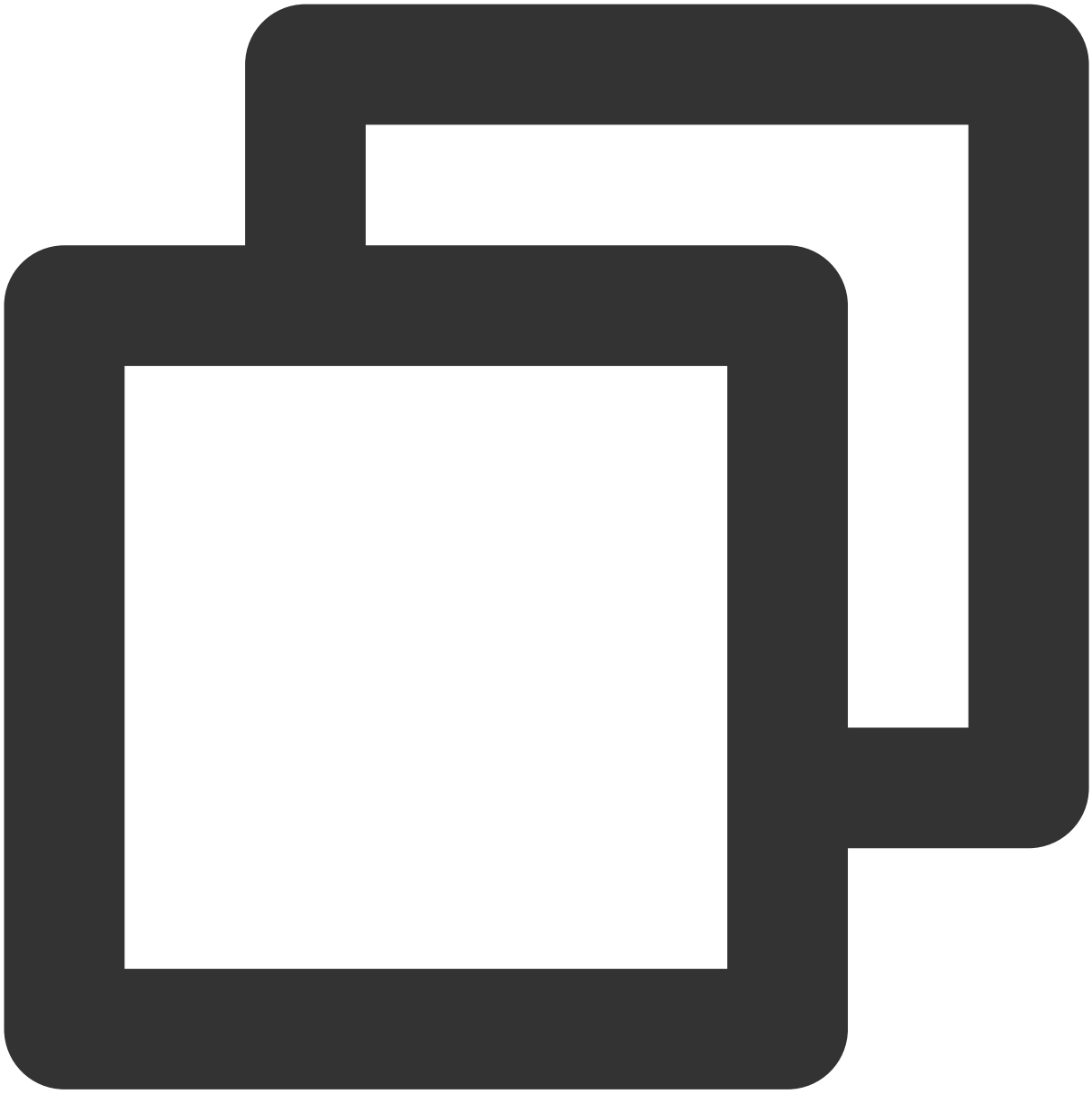
```
public abstract void getAnchorList(TRTCLiveRoomCallback.UserListCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

getAudienceList

获取房间内所有的观众信息，`enterRoom()` 成功后调用才有效。



```
public abstract void getAudienceList(TRTCLiveRoomCallback.UserListCallback callback
```

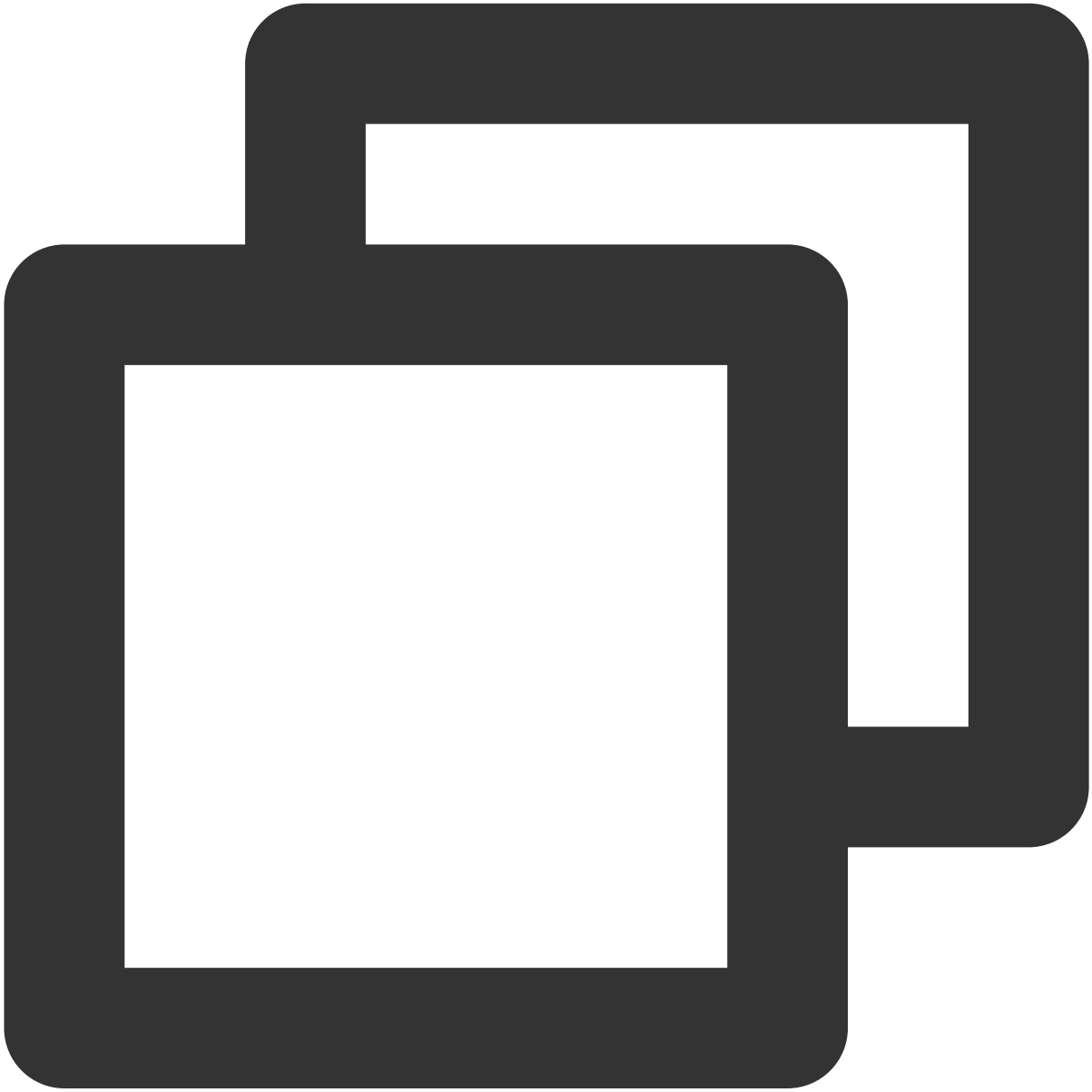
参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。



```
public abstract void startCameraPreview(boolean isFront, TXCloudVideoView view, TRT
```

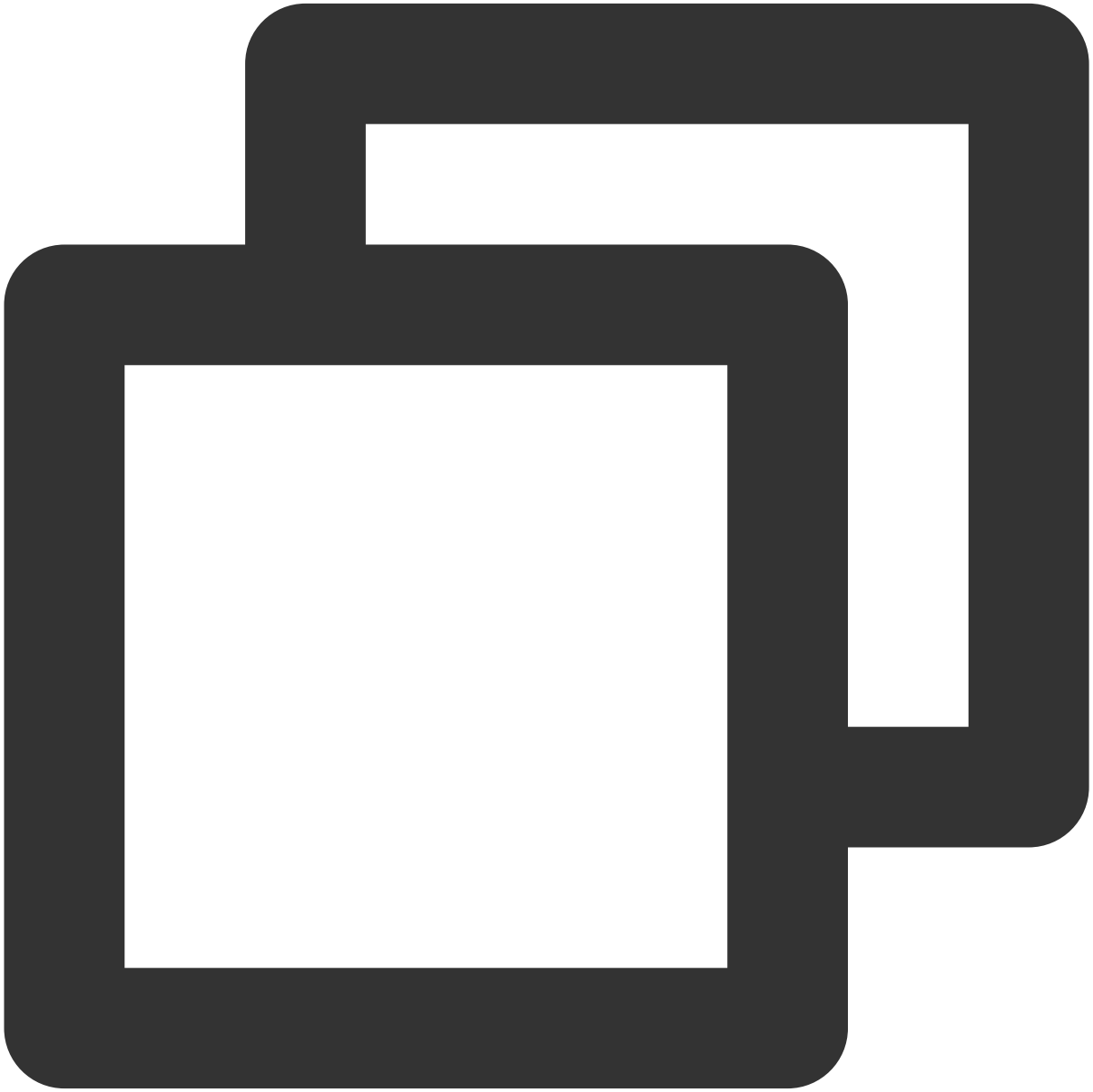
参数如下表所示：

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

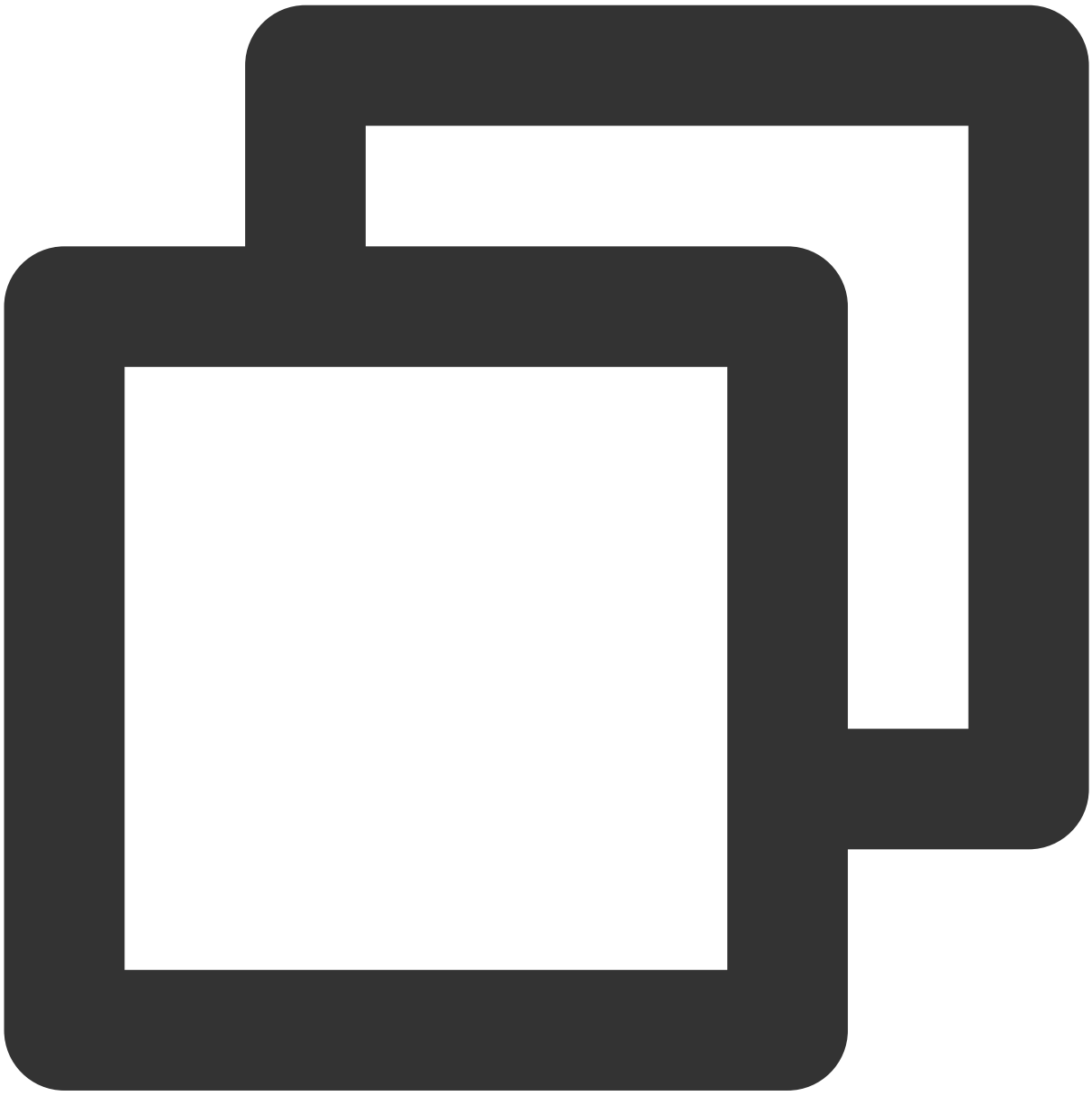
停止本地视频采集及预览。



```
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.ActionCallb
```

参数如下表所示：

参数	类型	含义

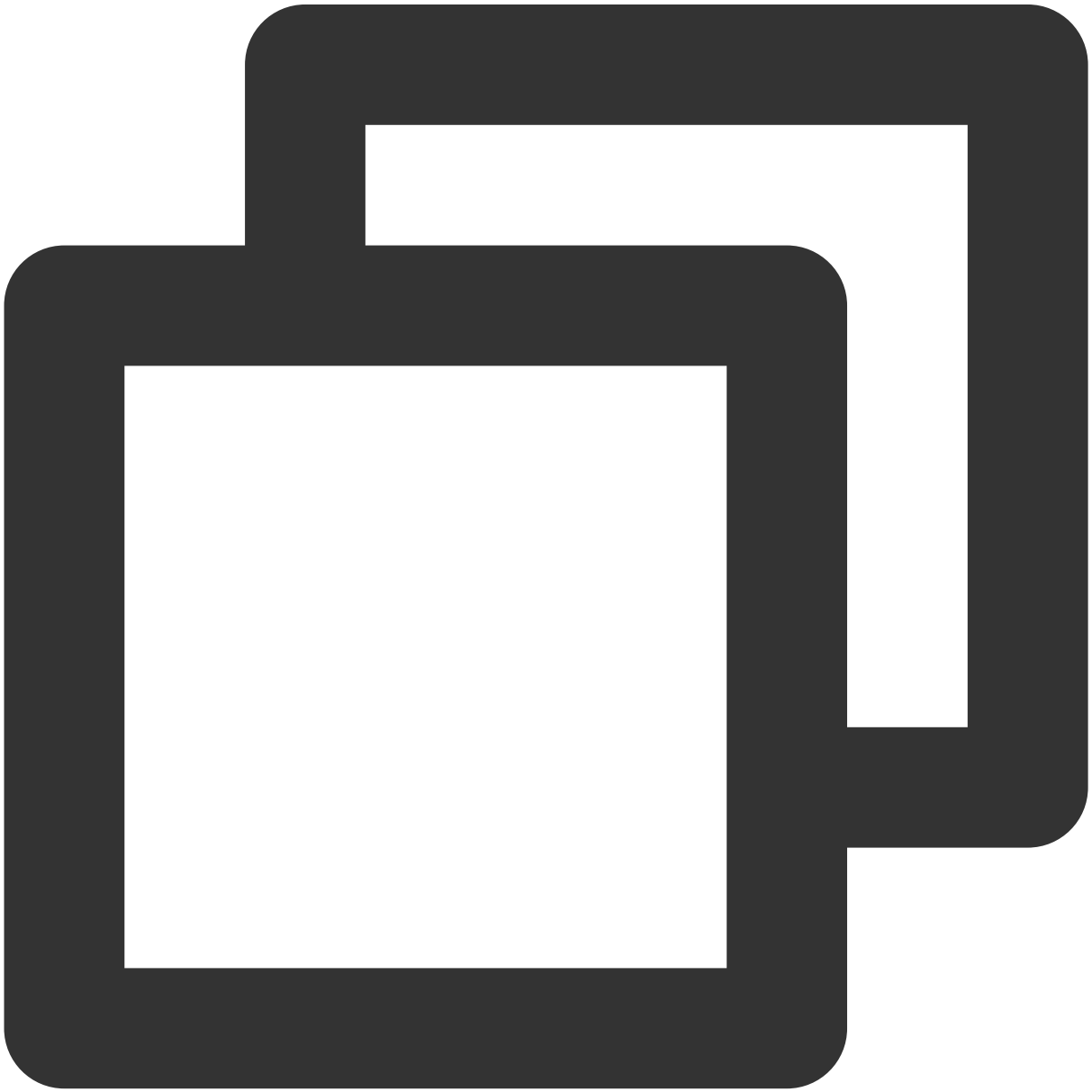
streamId	String	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
callback	ActionCallback	操作回调。

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



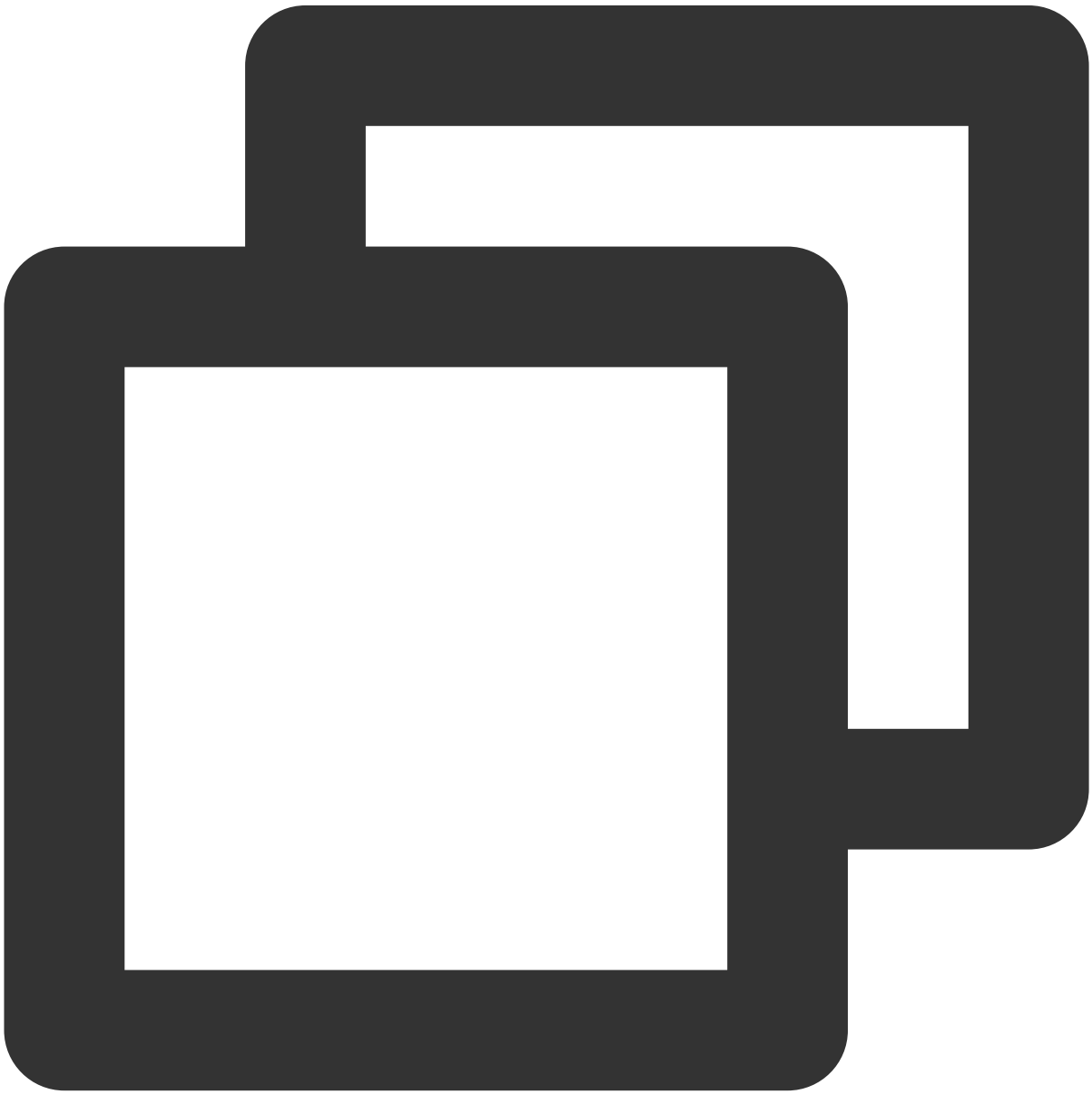
```
public abstract void stopPublish(TRTCLiveRoomCallback.ActionCallback callback);
```

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, TRTCLiveRoomCa
```

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。

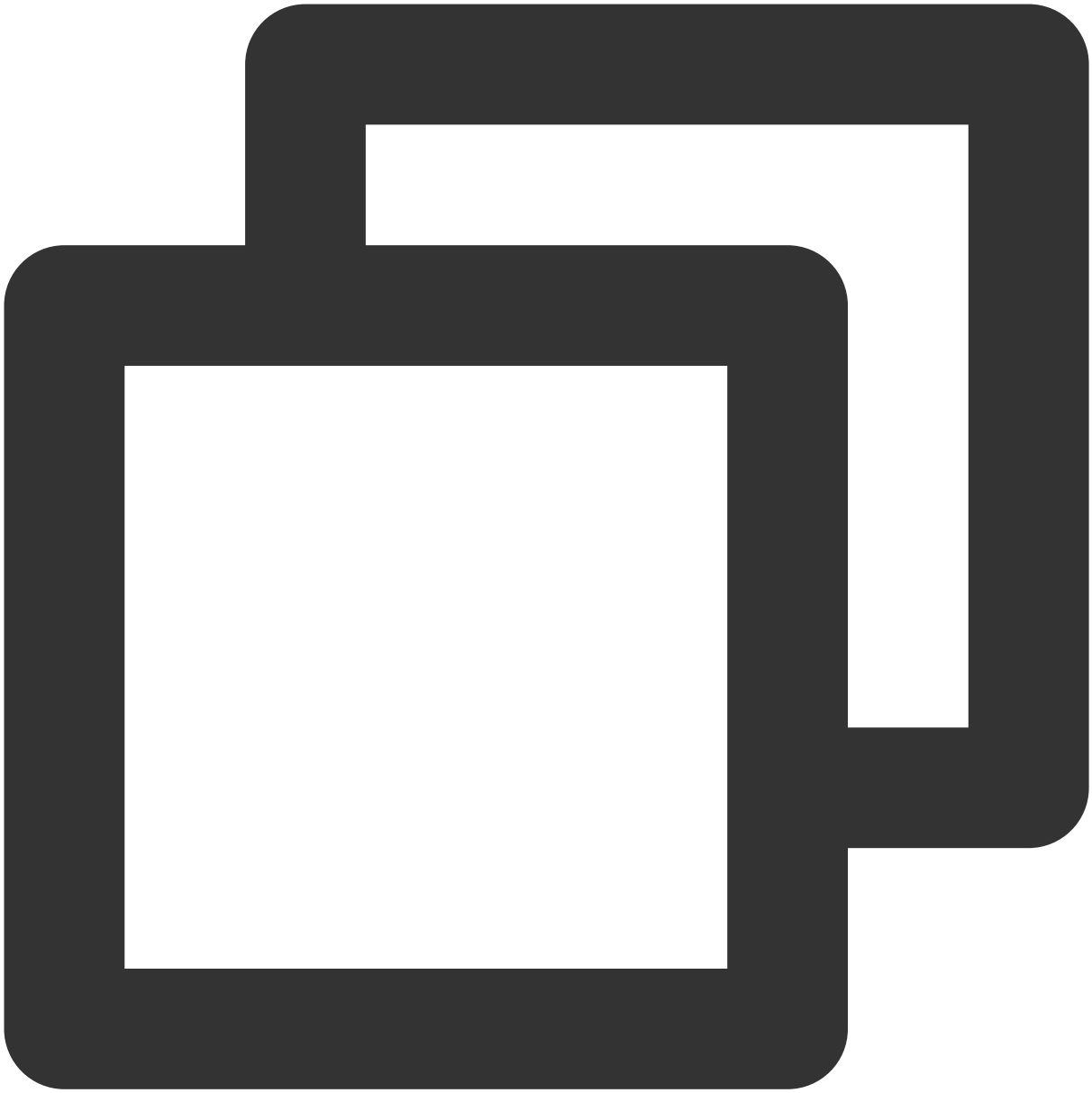
若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放主播的画面。

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 ca
```

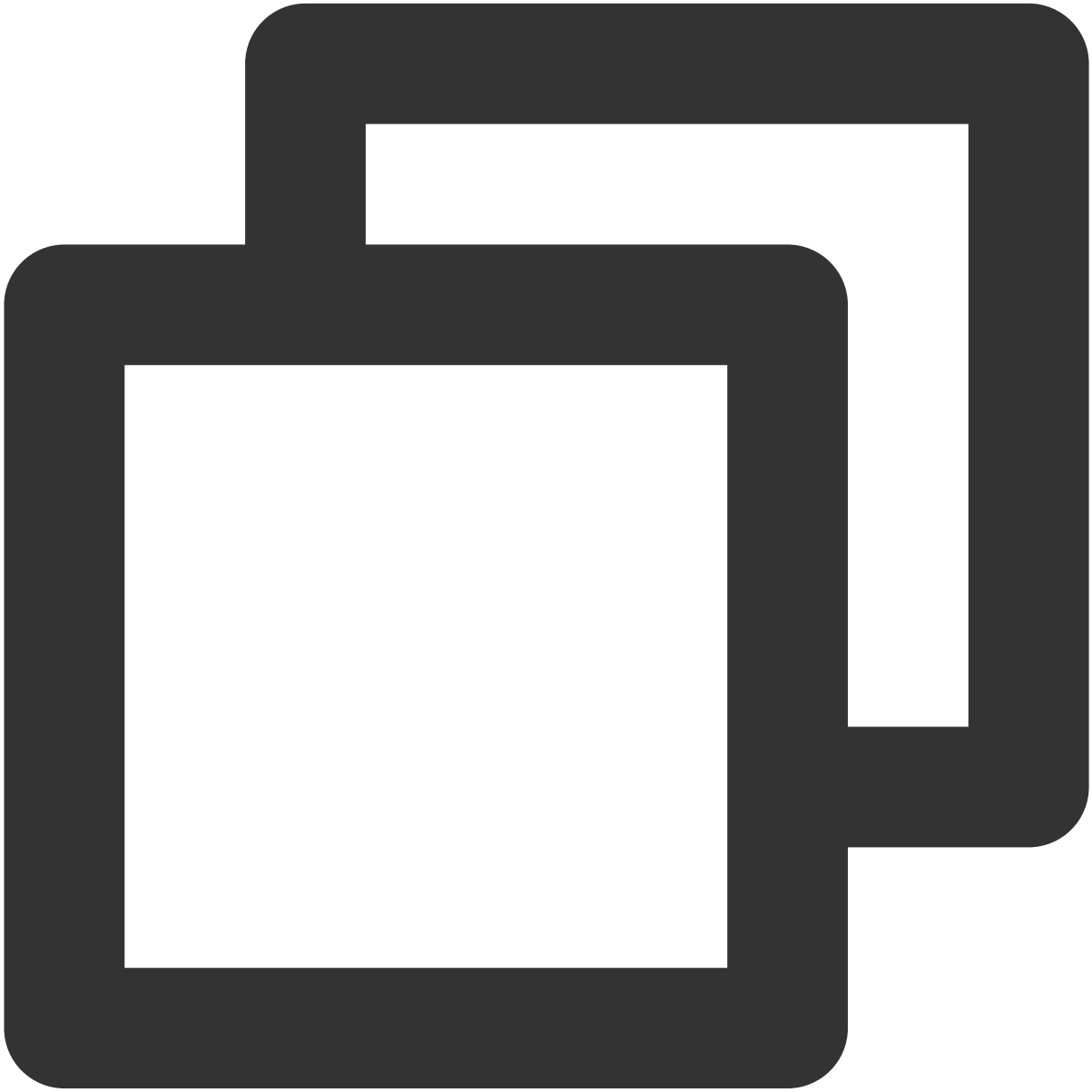
参数如下表所示：

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, TRTCLiveRoomCall
```

参数如下表所示：

参数	类型	含义

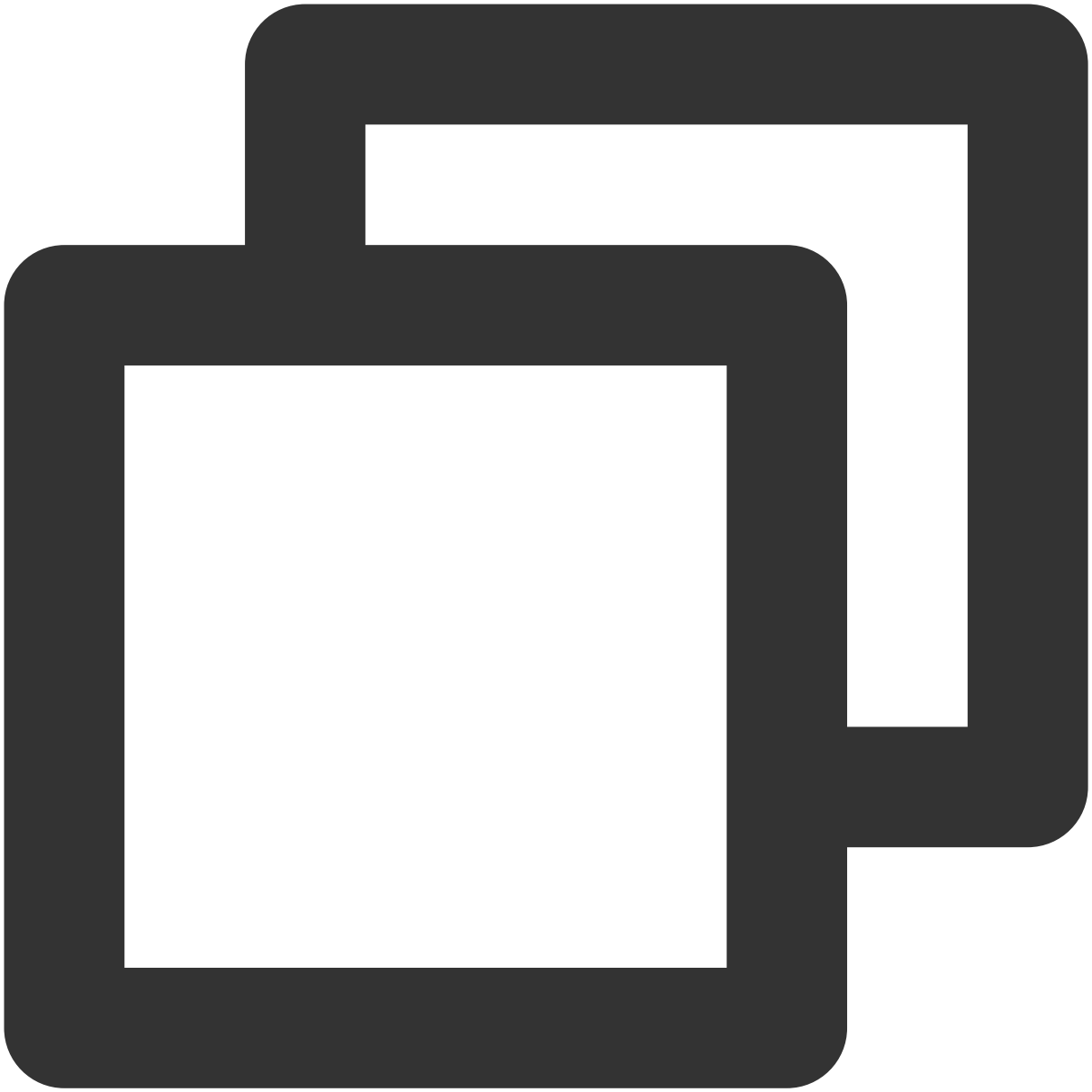
reason	String	连麦原因。
timeout	int	超时时间。
callback	ActionCallback	主播响应回调。

主播和观众的连麦流程如下：

1. 【观众】调用 `requestJoinAnchor()` 向主播发起连麦请求。
2. 【主播】会收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调通知。
3. 【主播】调用 `responseJoinAnchor()` 决定是否接受来自观众的连麦请求。
4. 【观众】会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 【观众】如果请求被同意，则调用 `startCameraPreview()` 开启本地摄像头。
6. 【观众】然后调用 `startPublish()` 正式进入推流状态。
7. 【主播】一旦观众进入连麦状态，主播会收到 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知。
8. 【主播】主播调用 `startPlay()` 即可看到连麦观众的视频画面。
9. 【观众】如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 `onAnchorEnter()` 通知，调用 `startPlay()` 播放其他连麦者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调后需要调用此接口来处理观众的连麦请求。



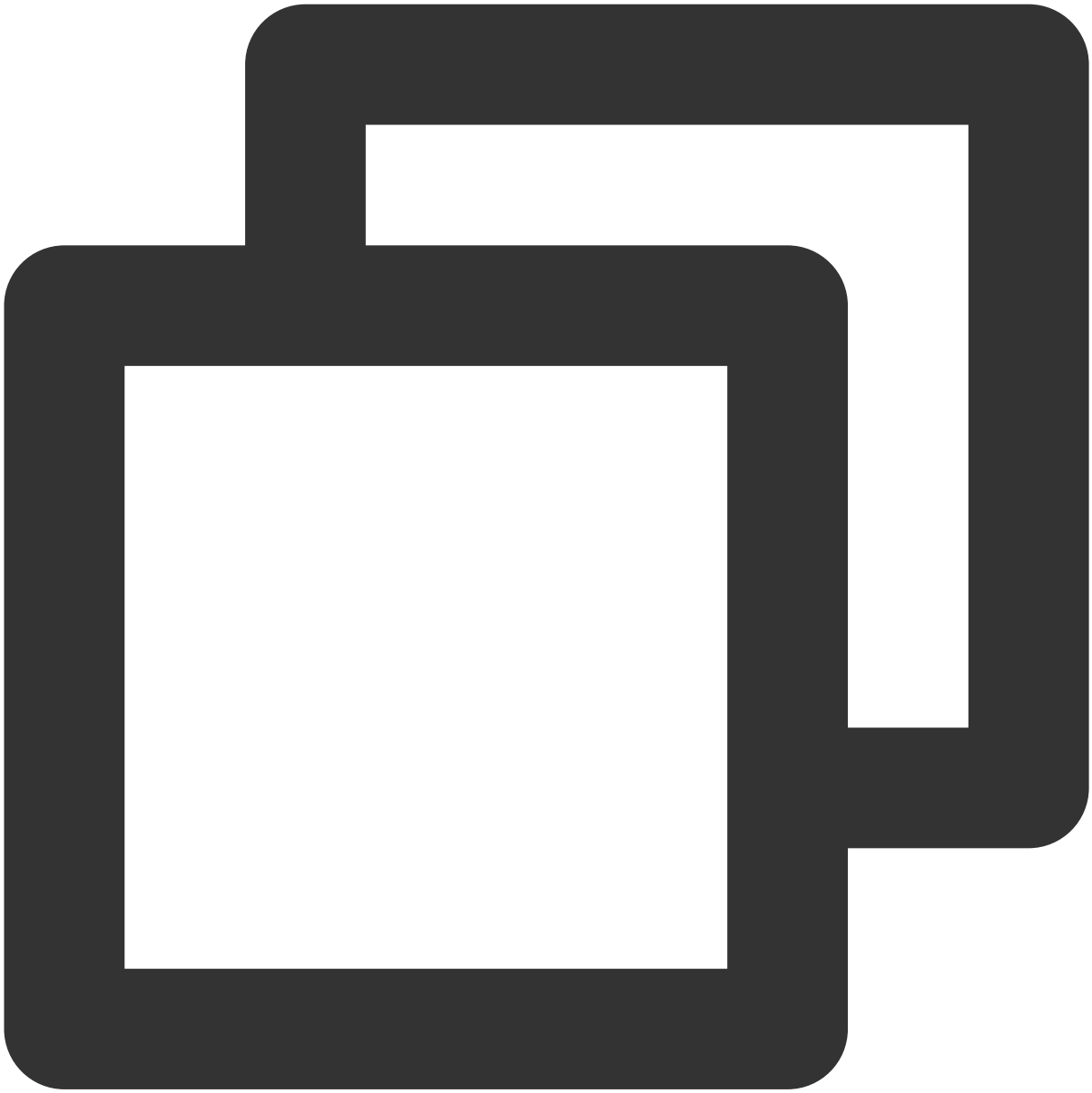
```
public abstract void responseJoinAnchor(String userId, boolean agree, String reason
```

参数如下表所示：

参数	类型	含义
userId	String	观众 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝连麦的原因描述。

kickoutJoinAnchor

主播踢除连麦观众。主播调用此接口踢除连麦观众后，被踢连麦观众会收到 `TRTCLiveRoomDelegate` 的 `onKickoutJoinAnchor()` 回调通知。



```
public abstract void kickoutJoinAnchor(String userId, TRTCLiveRoomCallback.ActionCa
```

参数如下表所示：

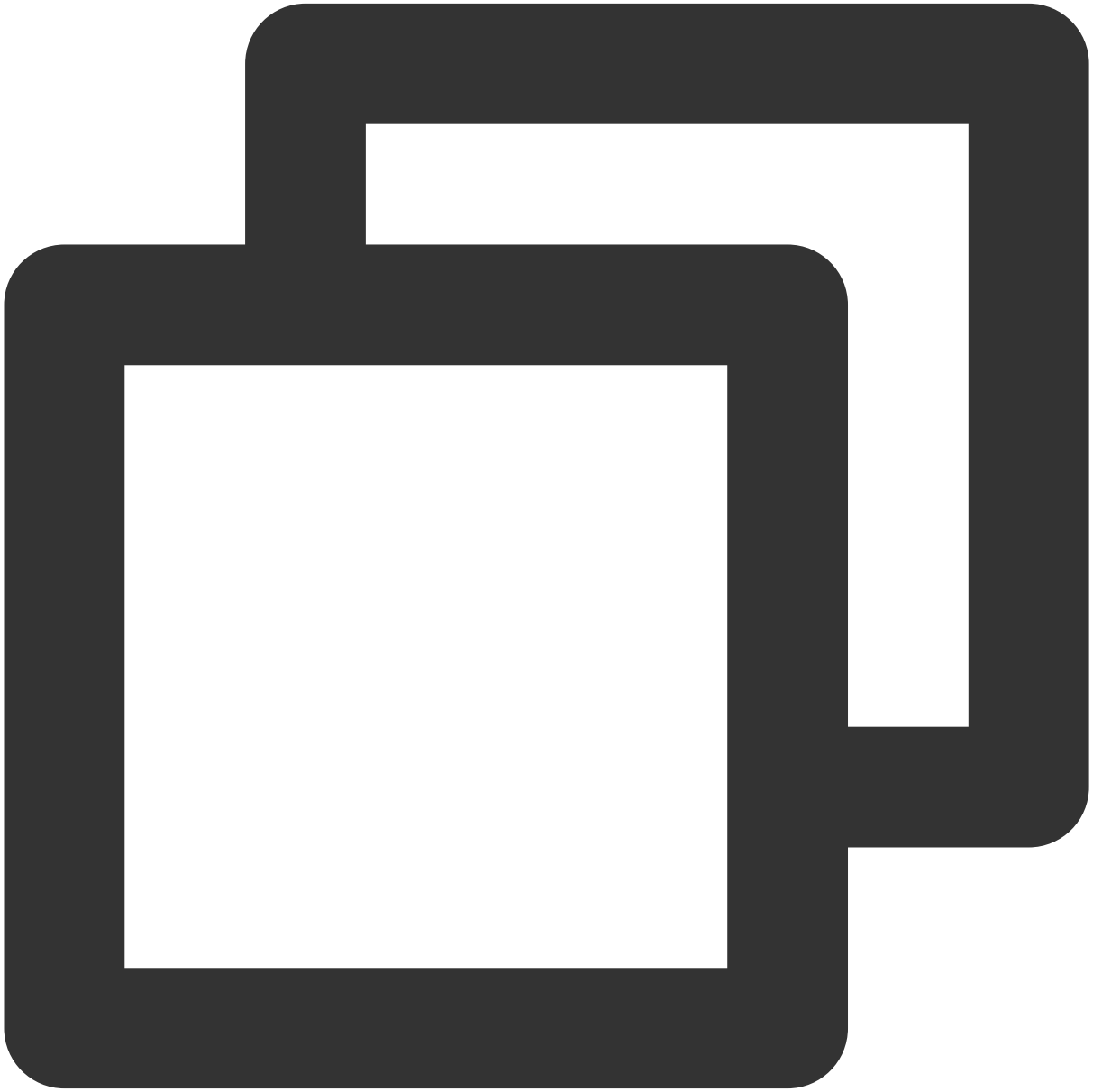
参数	类型	含义
userId	String	连麦观众 ID。

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.
```

参数如下表所示：

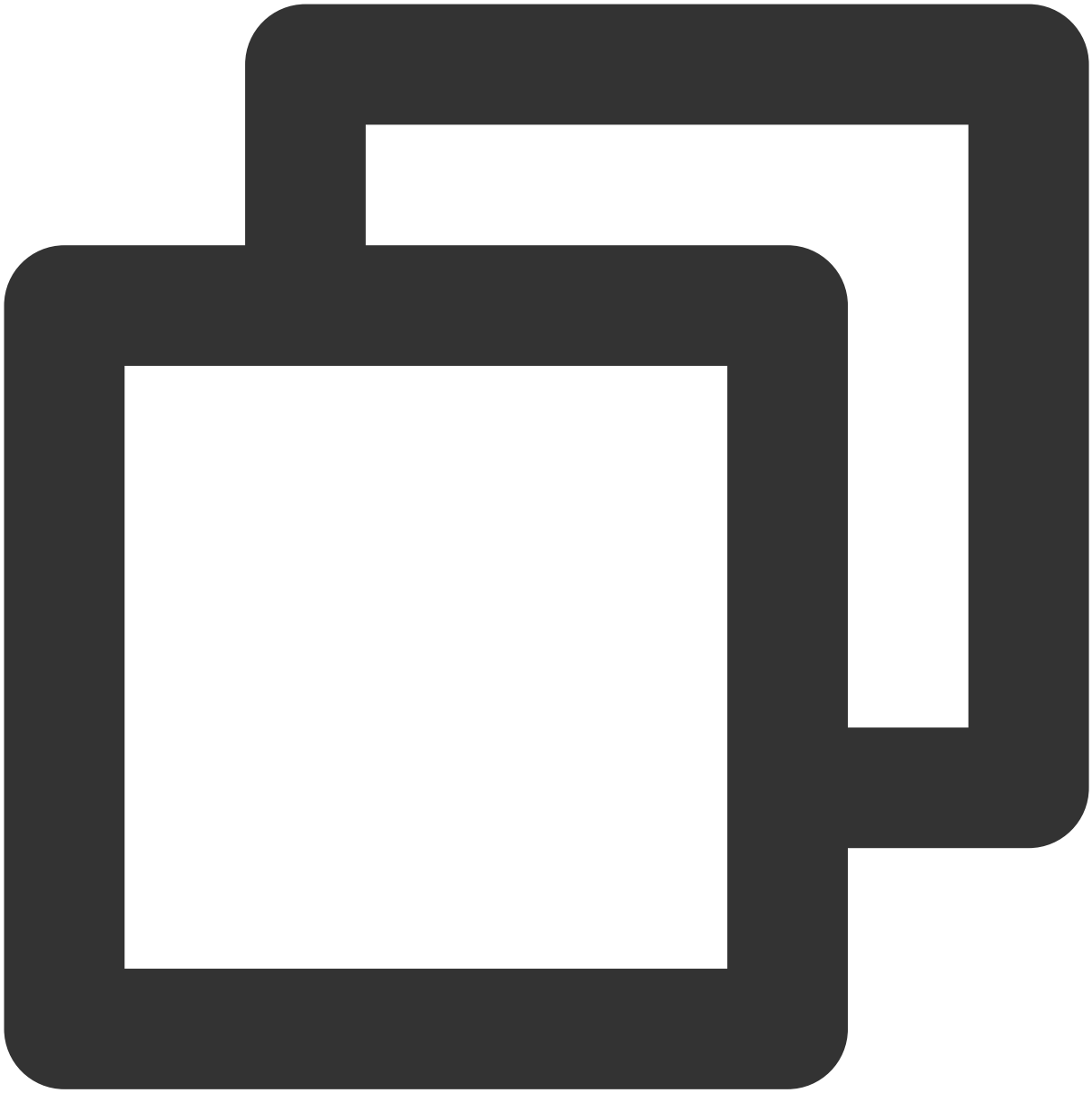
参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。
callback	ActionCallback	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 【主播 A】调用 `requestRoomPK()` 向主播 B 发起连麦请求。
2. 【主播 B】会收到 `TRTCLiveRoomDelegate` 的 `onRequestRoomPK()` 回调通知。
3. 【主播 B】调用 `responseRoomPK()` 决定是否接受主播 A 的 PK 请求。
4. 【主播 B】如果接受主播 A 的要求，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来显示主播 A 的视频画面。
5. 【主播 A】会收到 `responseCallback` 回调通知，该通知会携带来自主播 B 的处理结果。
6. 【主播 A】如果请求被同意，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调。



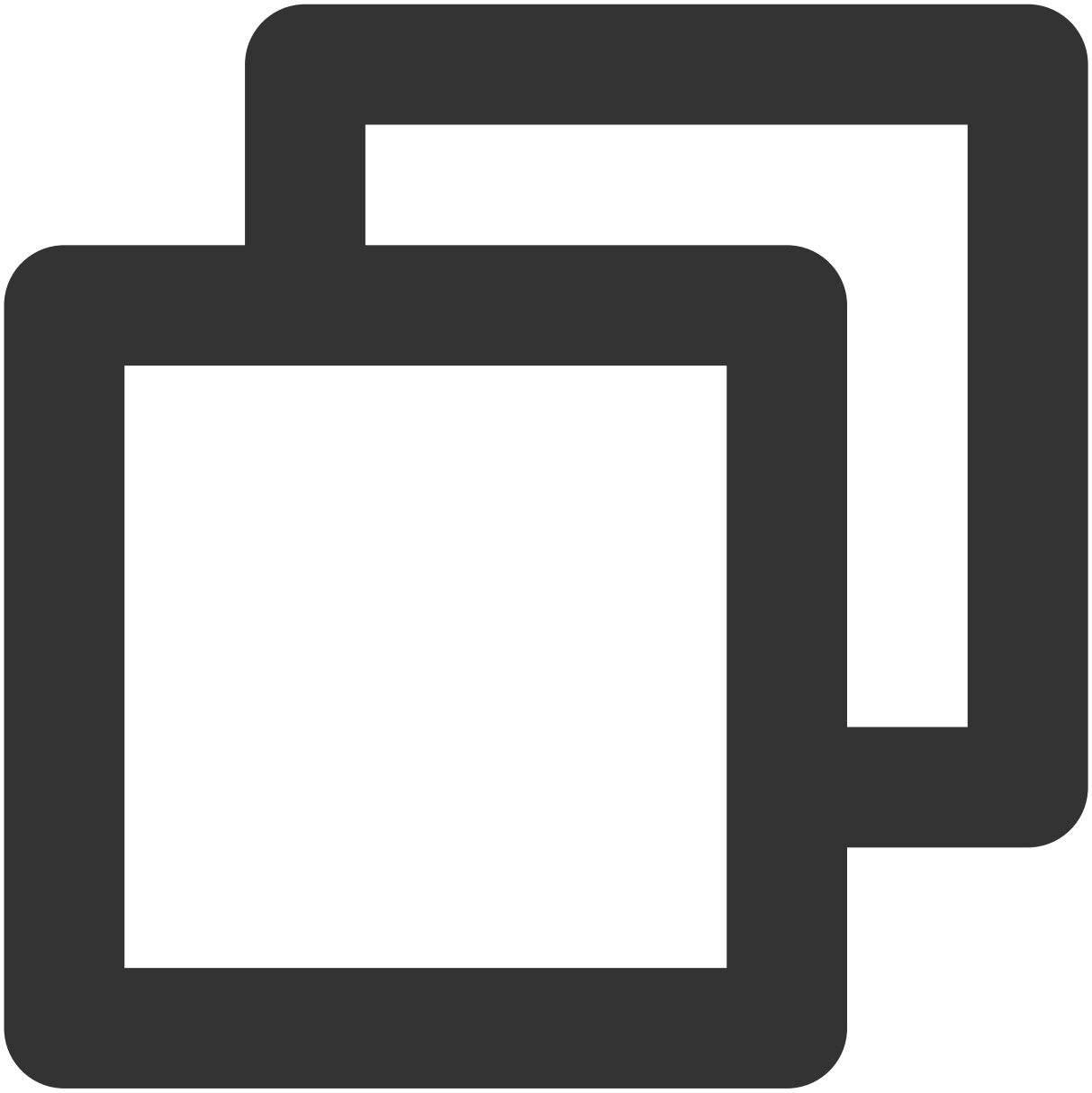
```
public abstract void responseRoomPK(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `onQuitRoomPk()` 回调通知。



```
public abstract void quitRoomPK(TRTCLiveRoomCallback.ActionCallback callback);
```

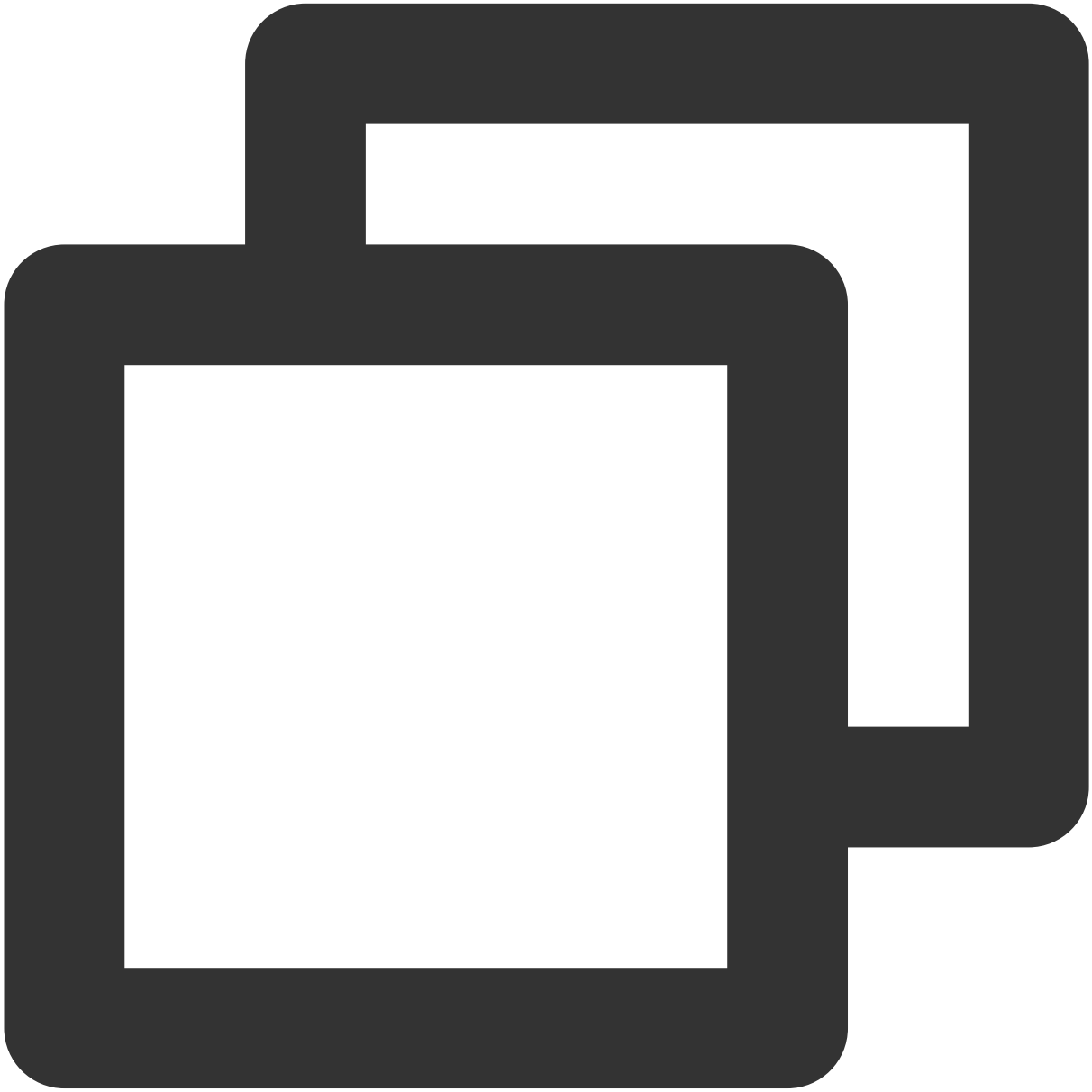
参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

音视频控制相关接口函数

switchCamera

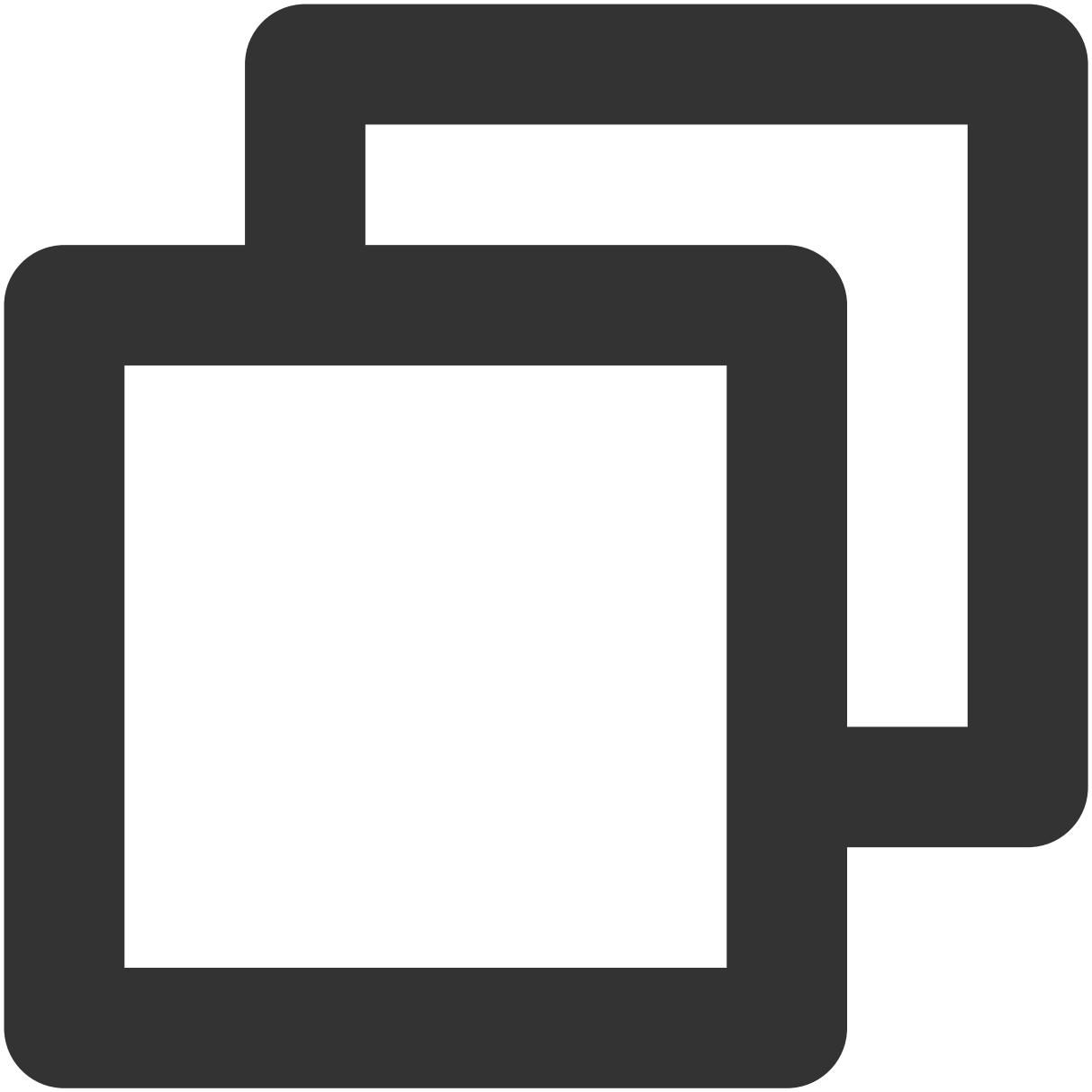
切换前后摄像头。



```
public abstract void switchCamera();
```

setMirror

设置是否镜像展示。



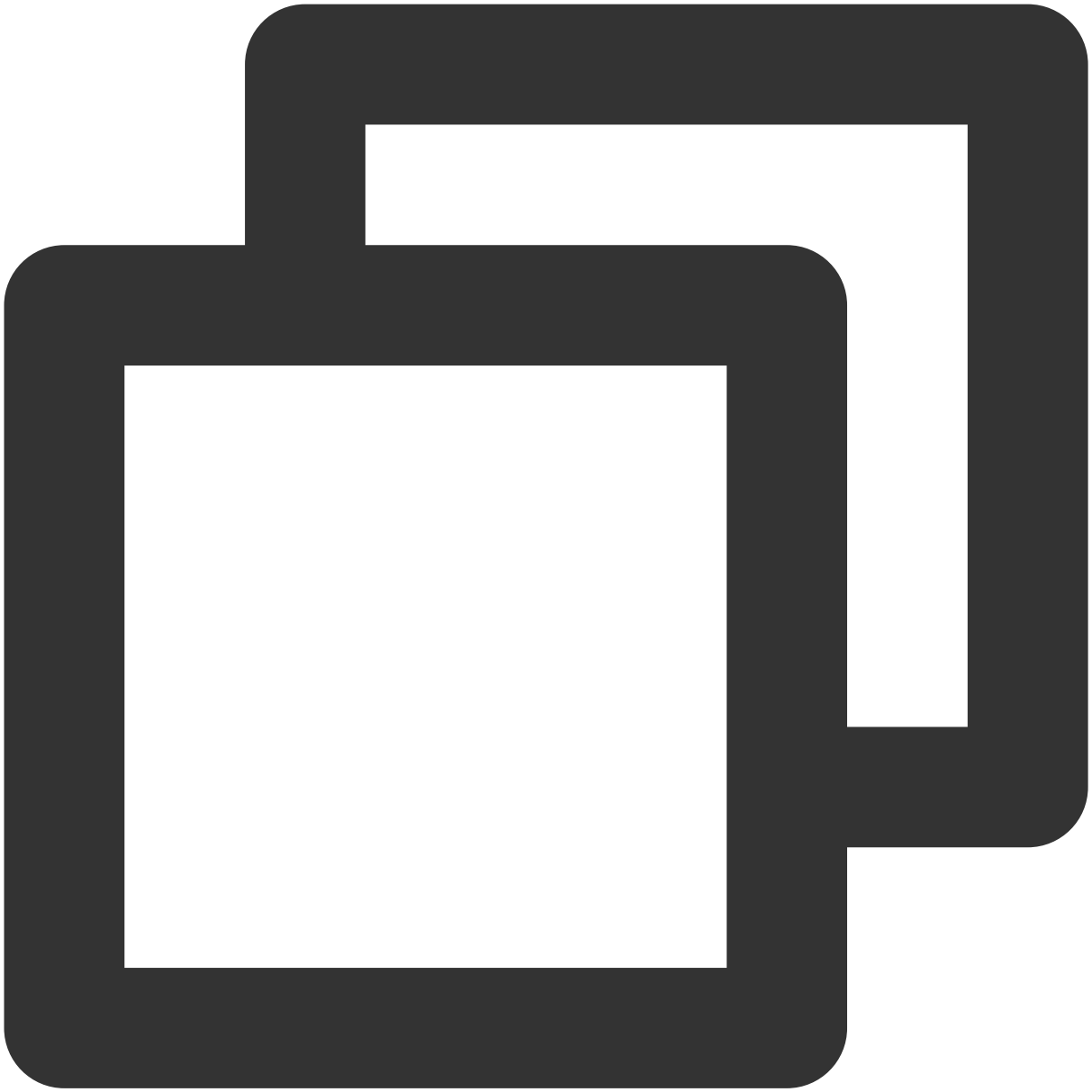
```
public abstract void setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	boolean	开启/关闭镜像。

muteLocalAudio

静音本地音频。



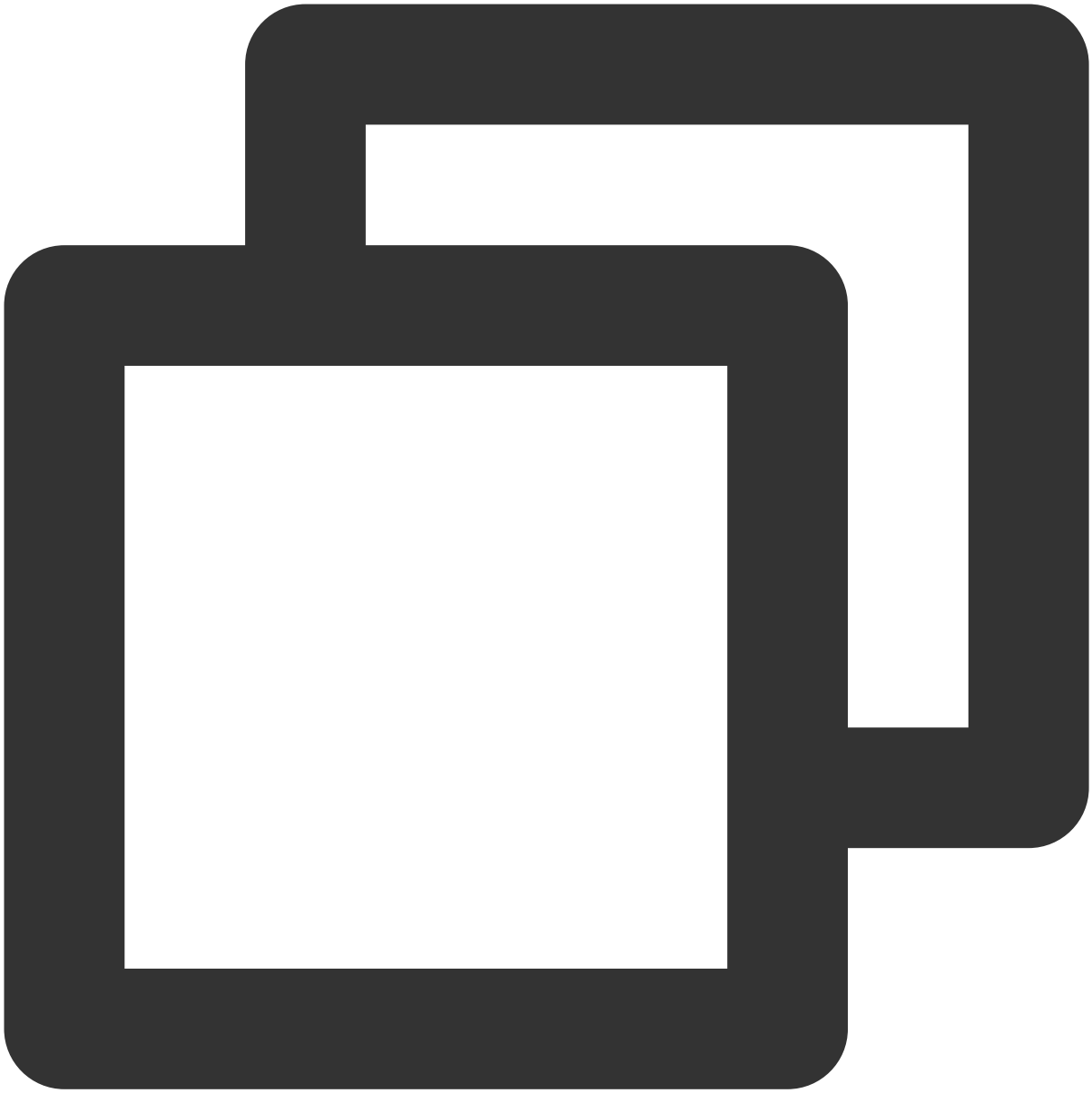
```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。



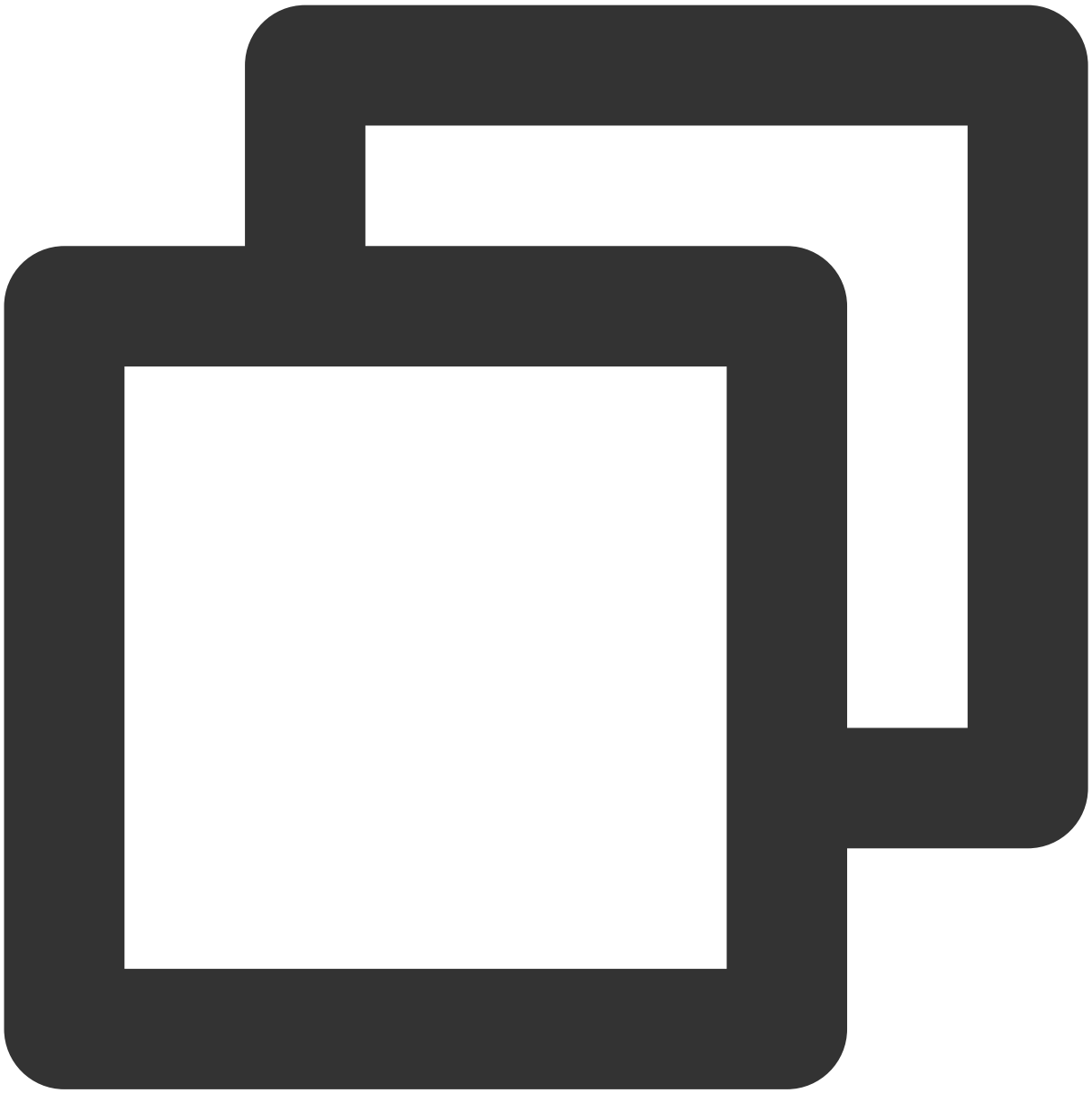
```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。



```
public abstract void muteAllRemoteAudio(boolean mute);
```

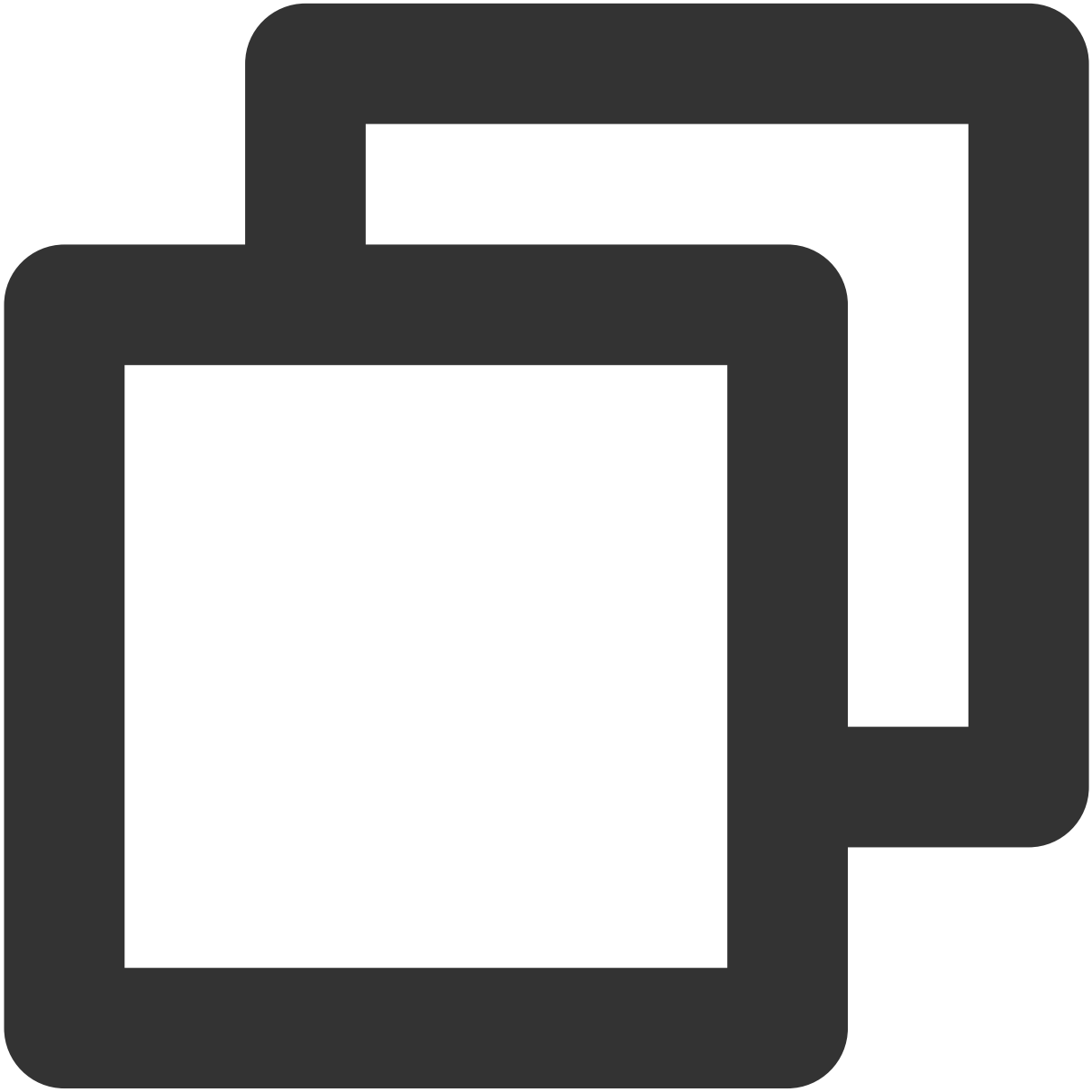
参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

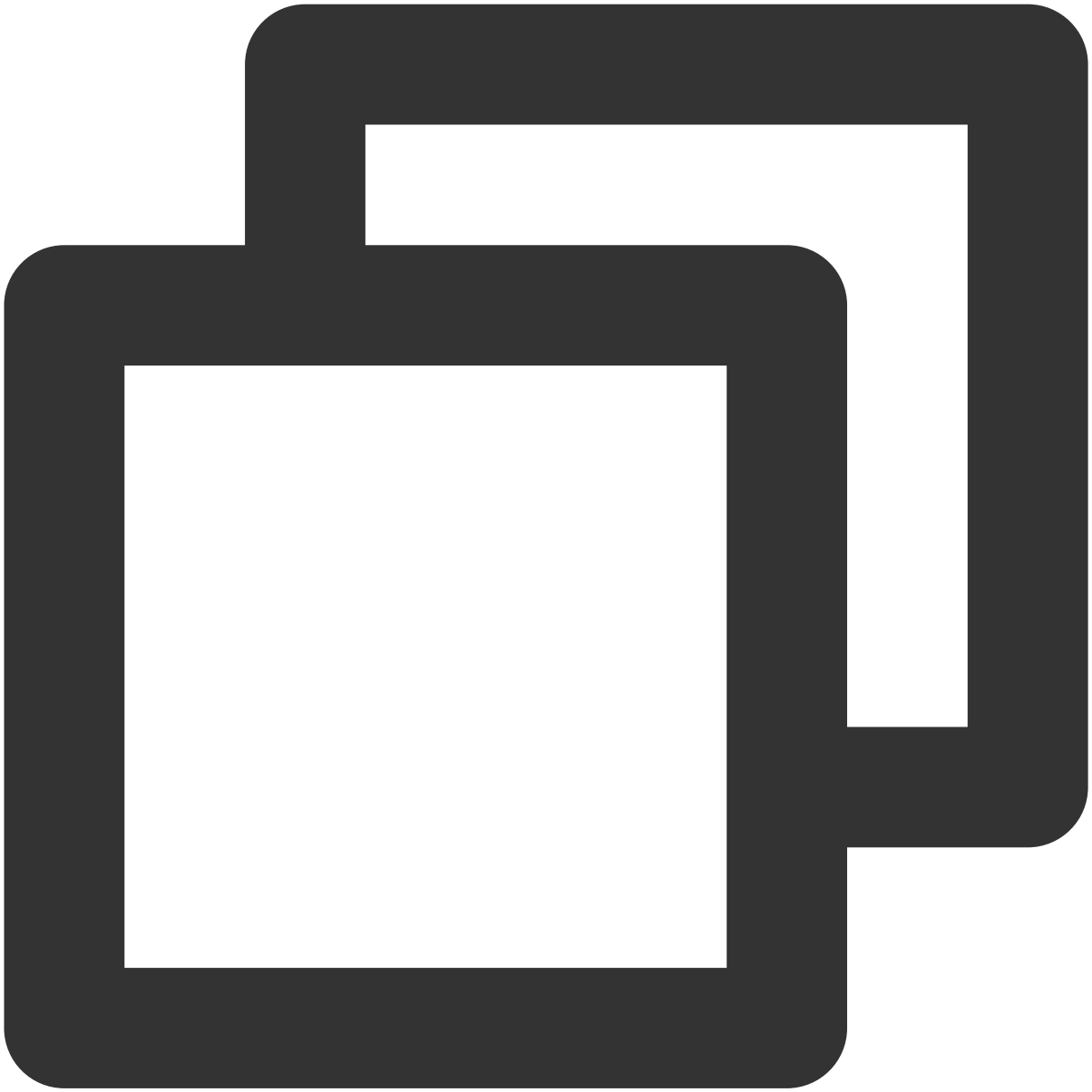


```
public abstract TXAudioEffectManager getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager。



```
public abstract TXBeautyManager getBeautyManager();
```

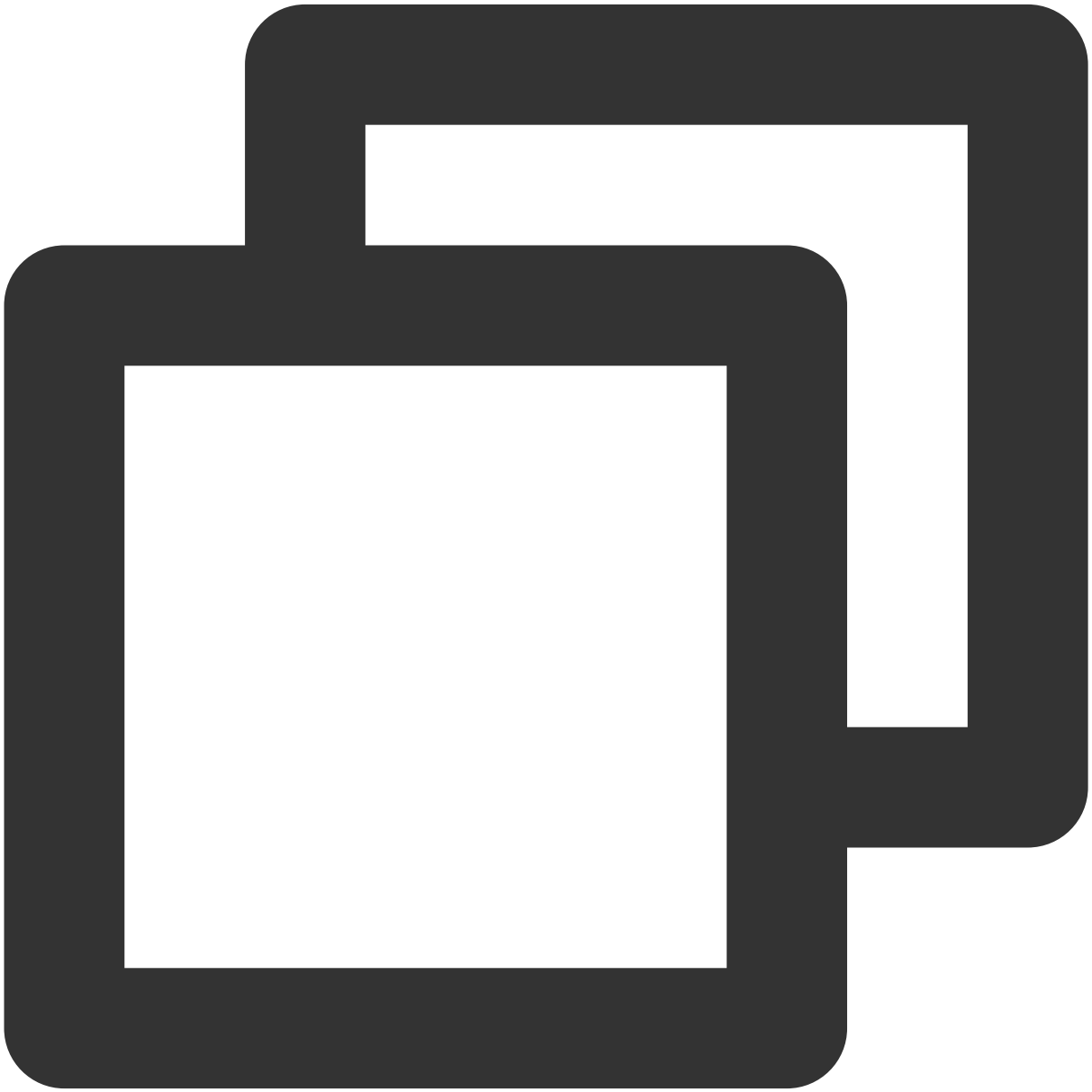
通过美颜管理，您可以使用以下功能：
设置“美颜风格”、“美白”、“红润”、“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等美容效果。
调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
设置人脸挂件（素材）等动态效果。

添加美妆。
进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。



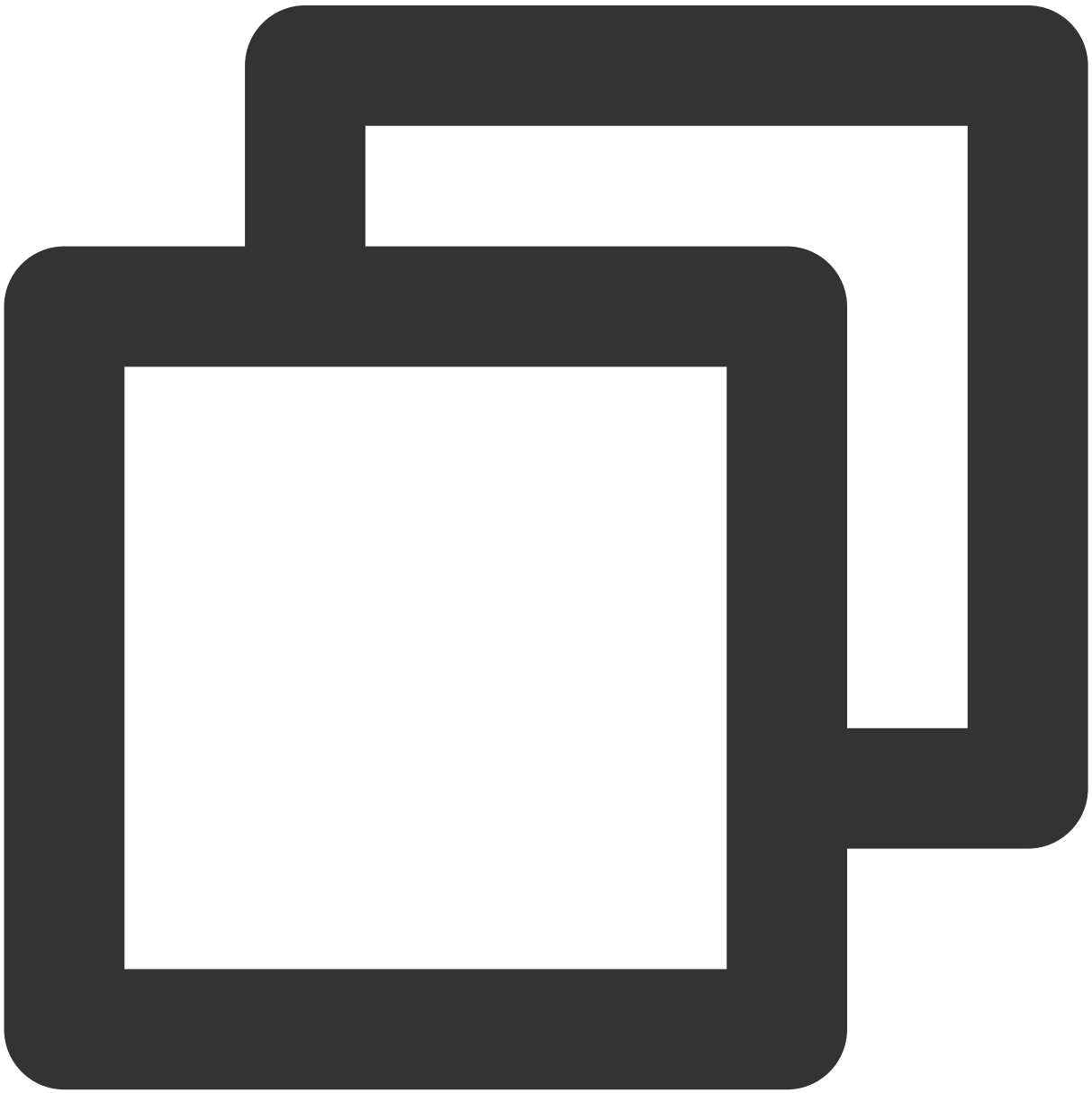
```
public abstract void sendRoomTextMsg(String message, TRTCLiveRoomCallback.ActionCal
```

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, TRTCLiveRoomCall
```

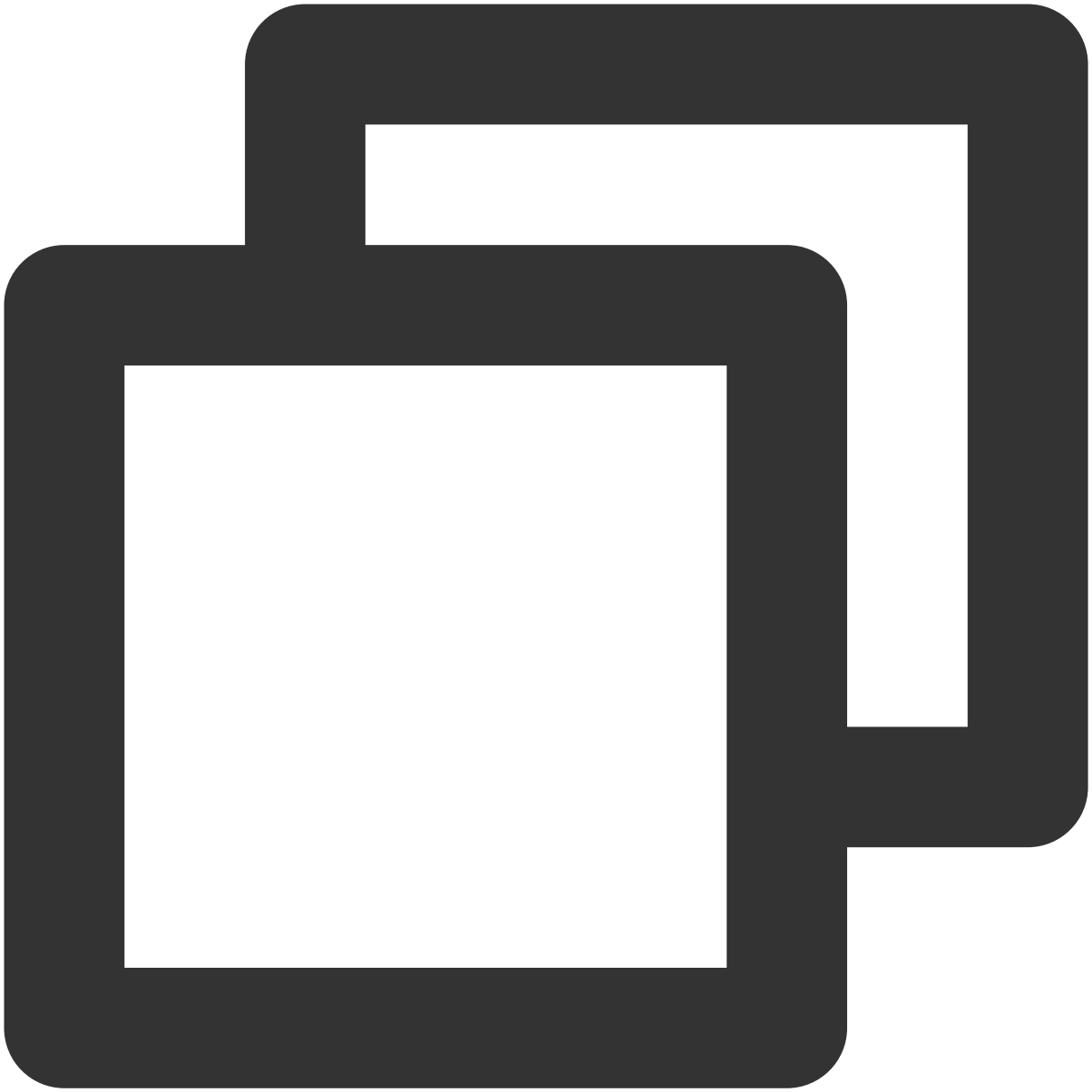
参数如下表所示：

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

调试相关接口函数

showVideoDebugLog

是否在界面中展示debug信息。



```
public abstract void showVideoDebugLog(boolean isShow);
```

参数如下表所示：

参数	类型	含义
isShow	boolean	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

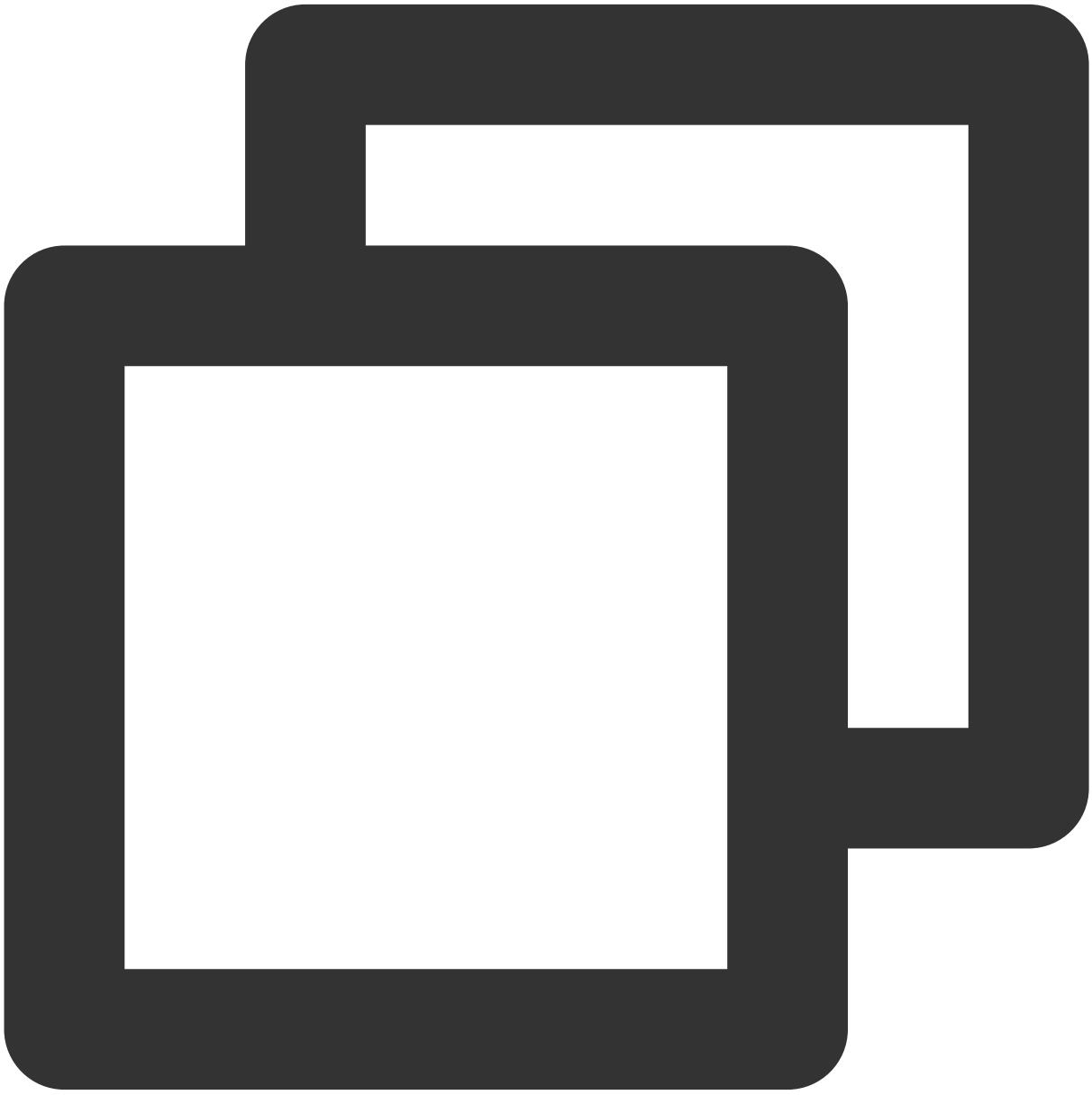
通用事件回调

onError

错误回调。

说明：

SDK 不可恢复的错误，一定要监听，并分情况给用户适当的界面提示。



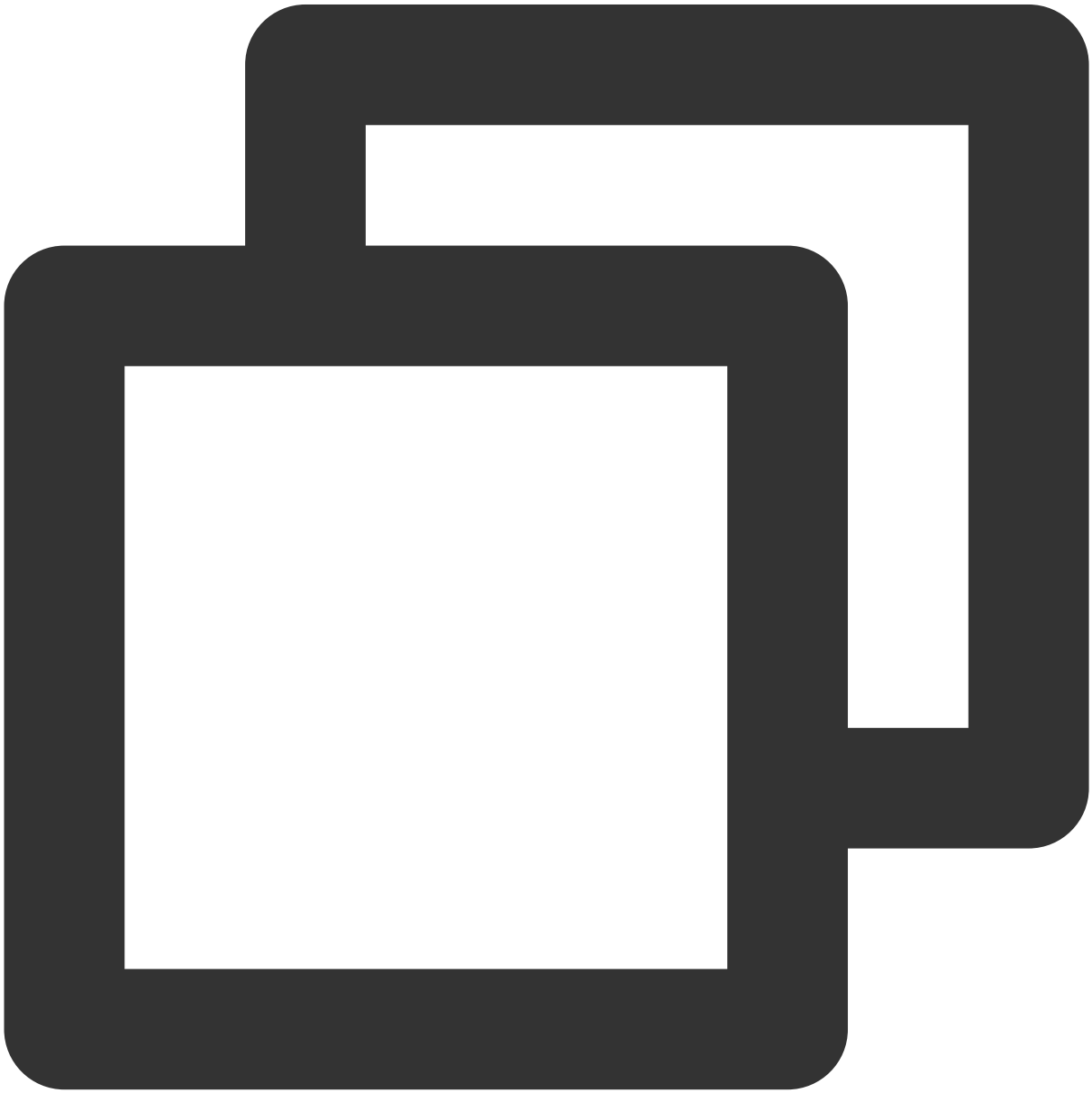
```
void onError(int code, String message);
```

参数如下表所示：

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

onWarning

警告回调。



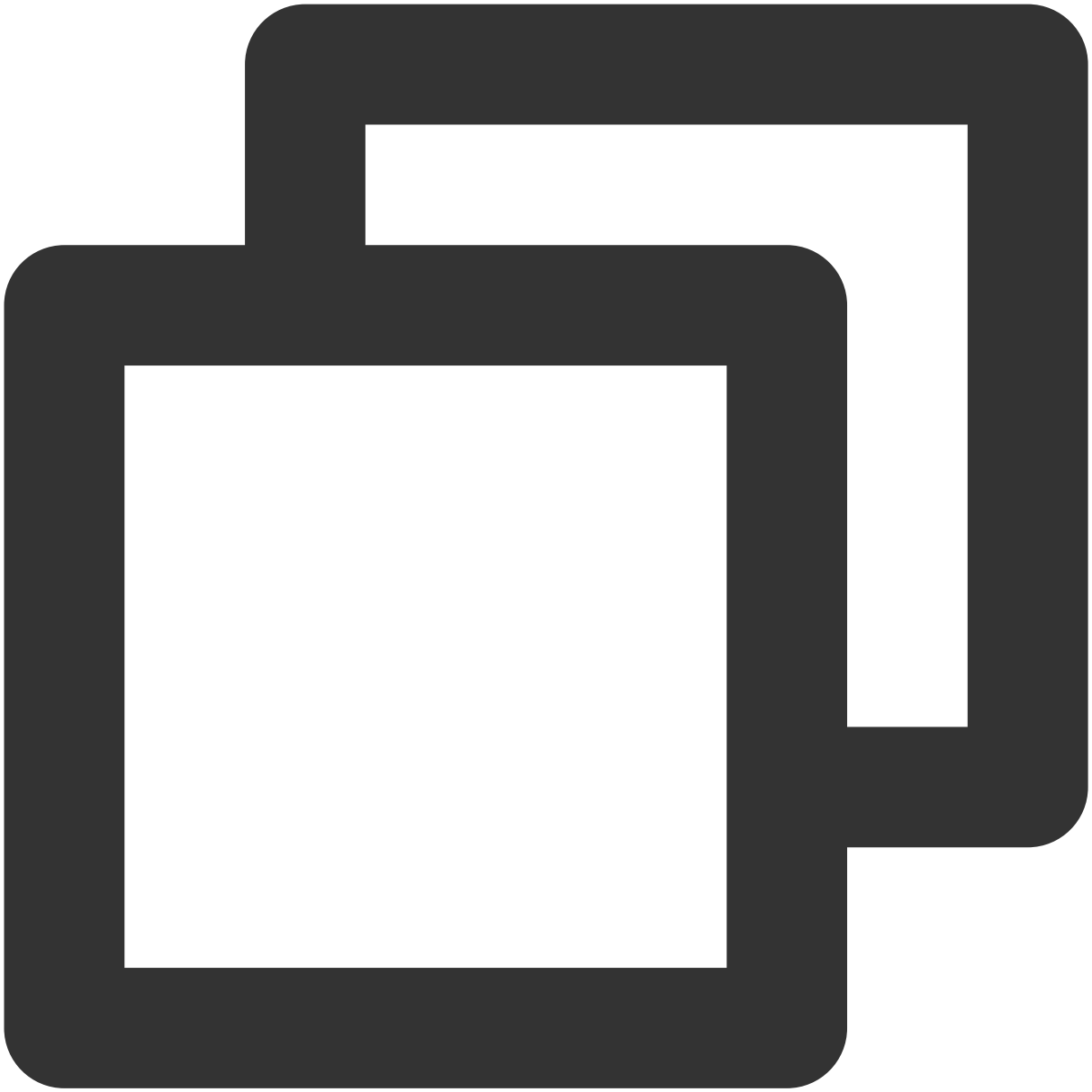
```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。



```
void onDebugLog(String message);
```

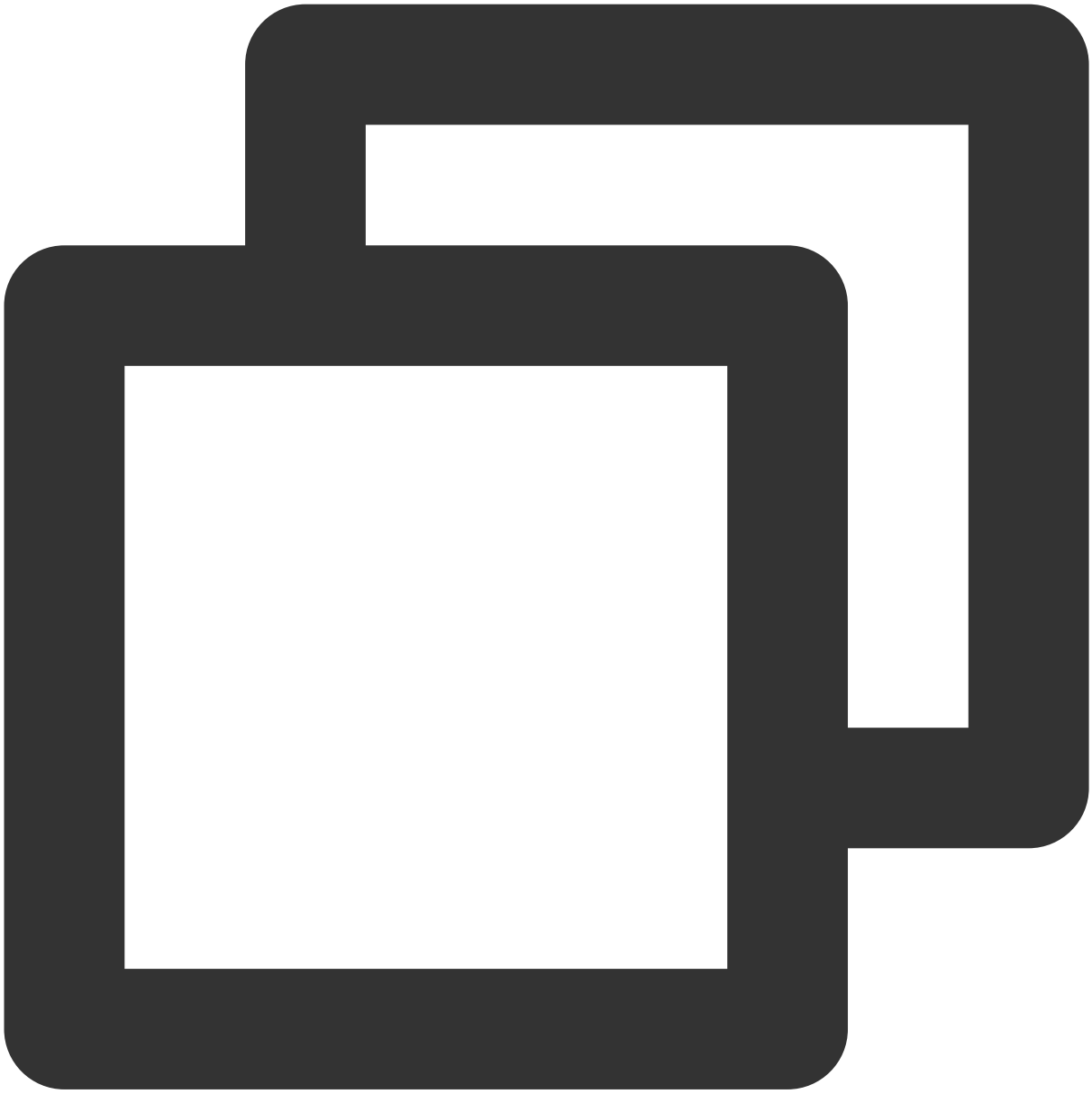
参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。



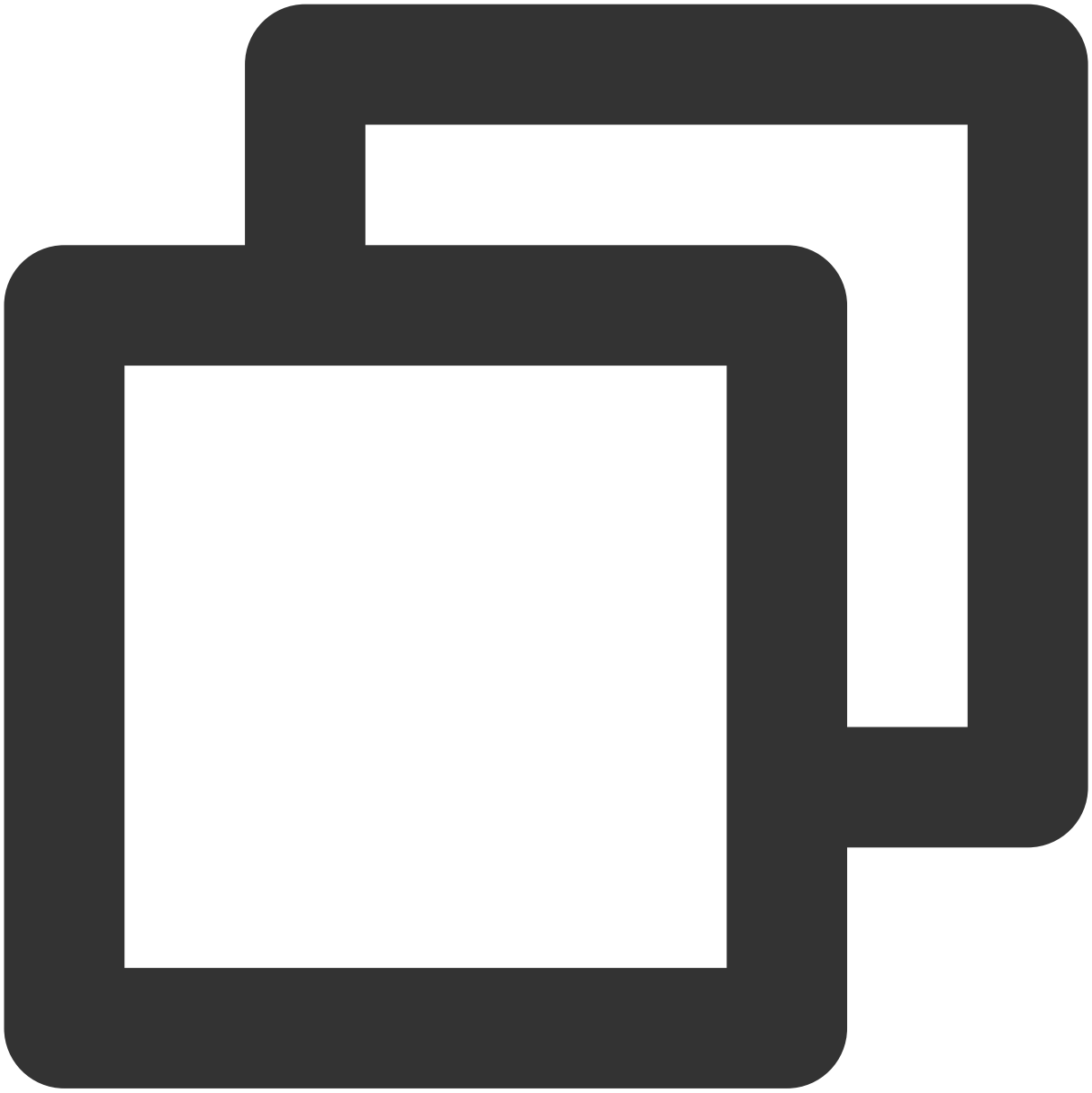
```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。



```
void onRoomInfoChange(TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo);
```

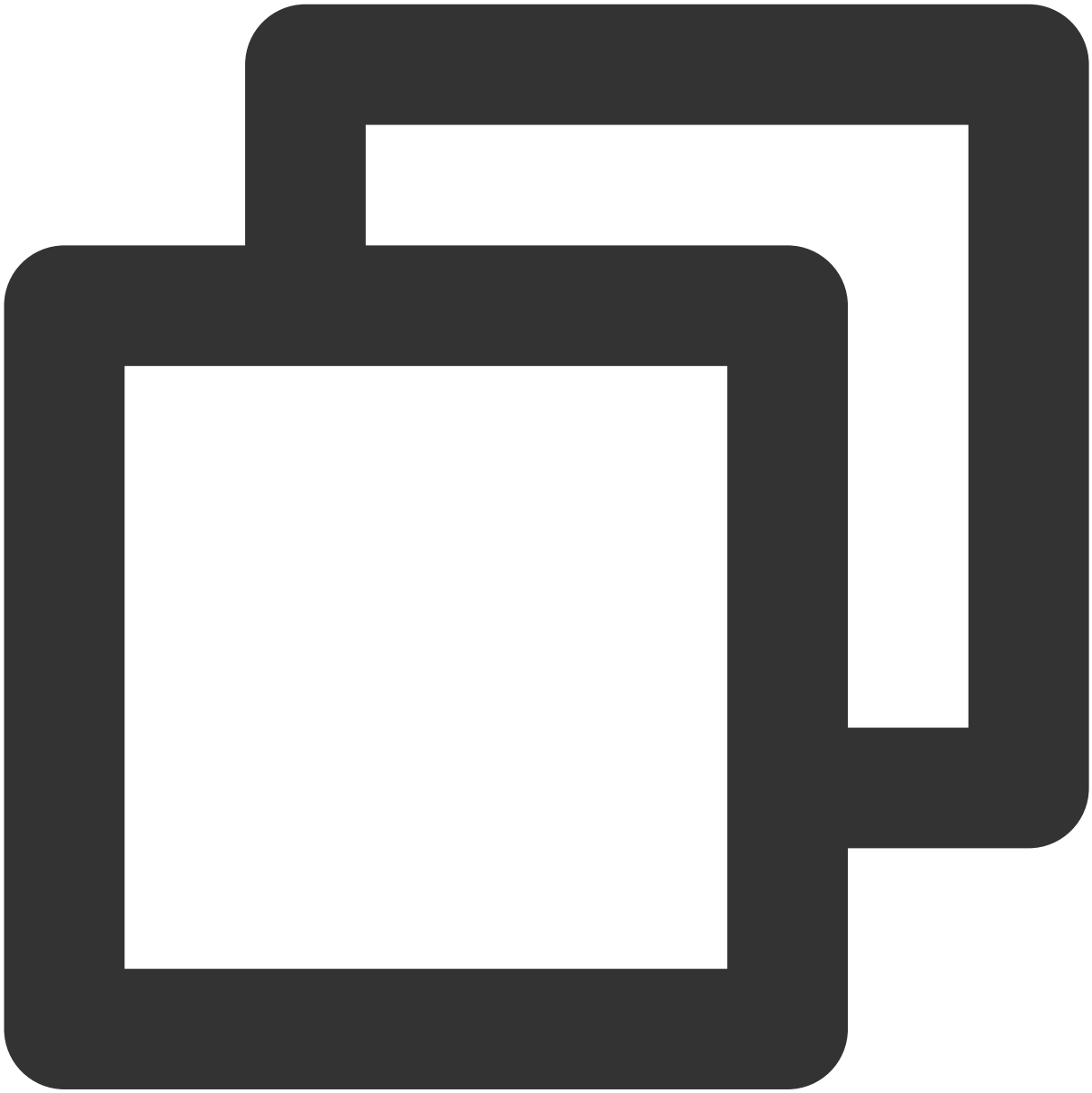
参数如下表所示：

参数	类型	含义
roomInfo	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。



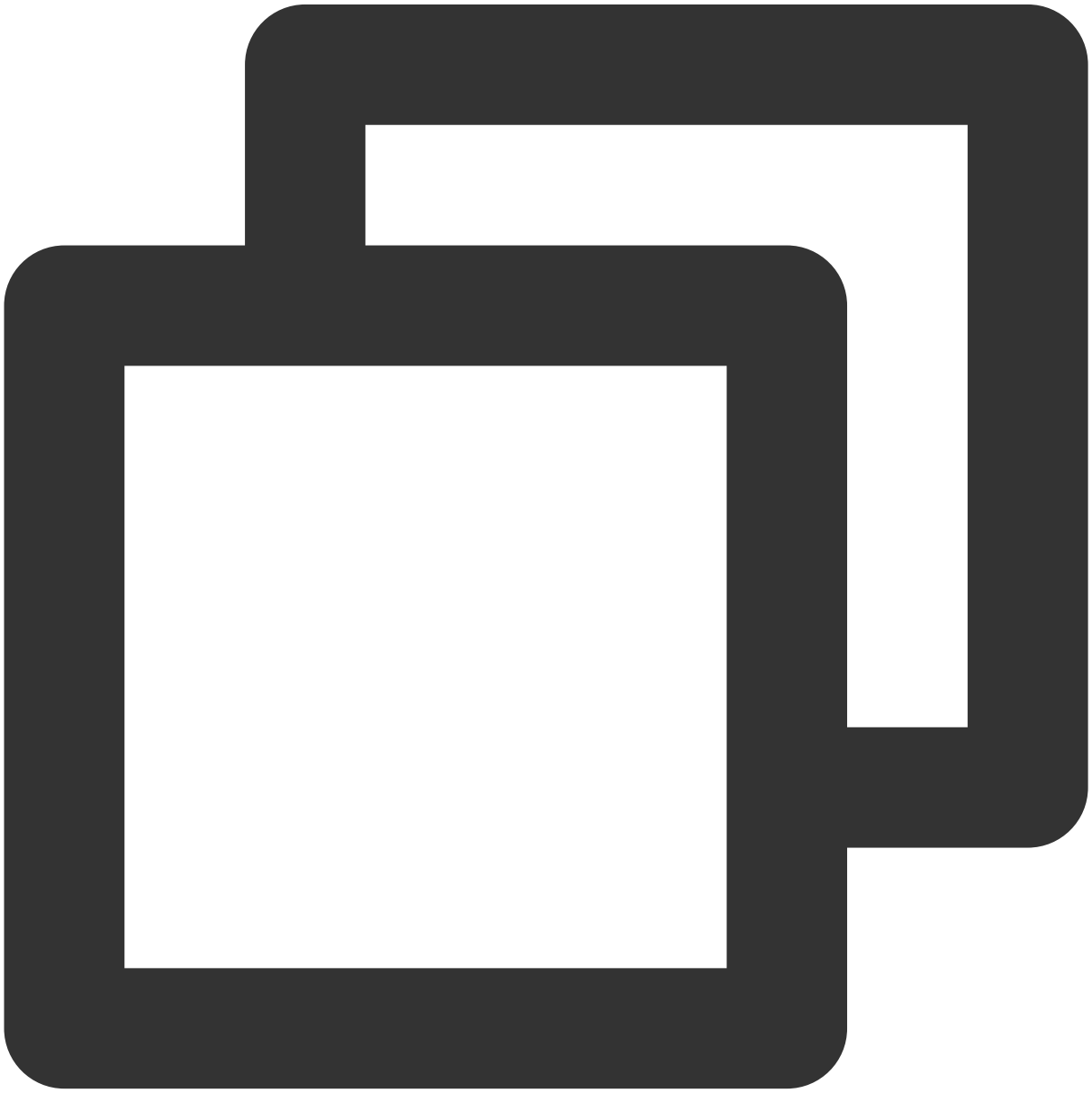
```
void onAnchorEnter(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。



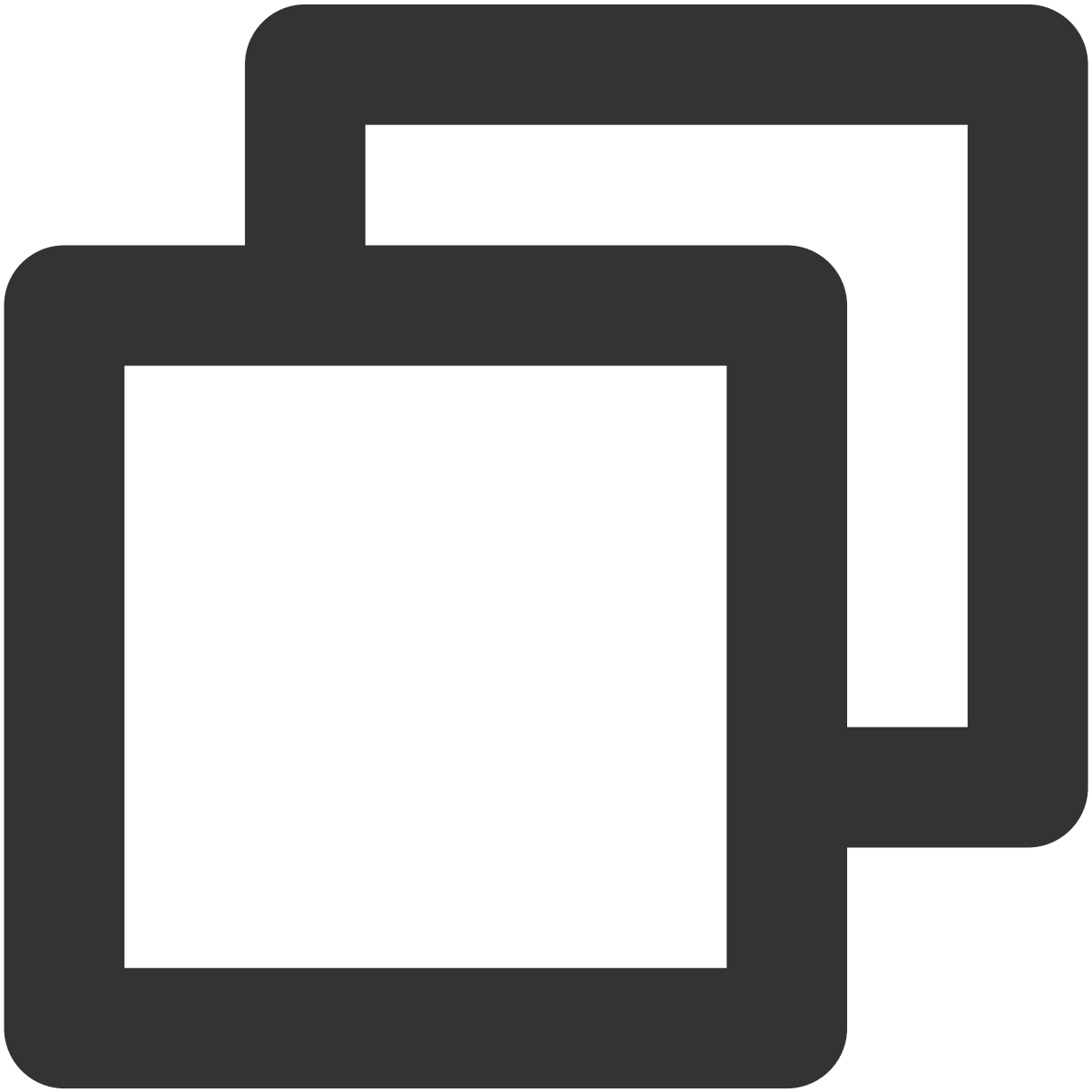
```
void onAnchorExit(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。



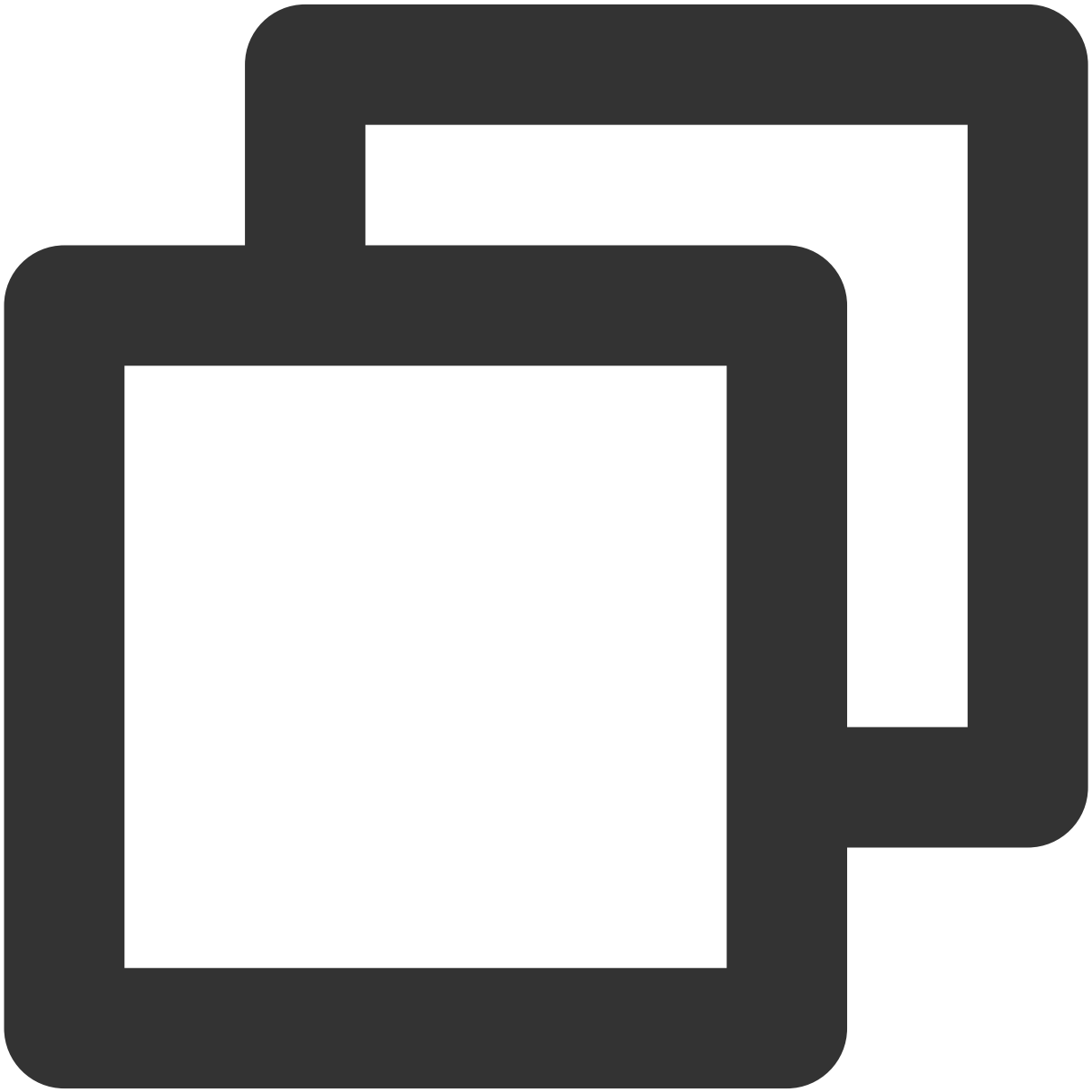
```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。



```
void onAudienceExit(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

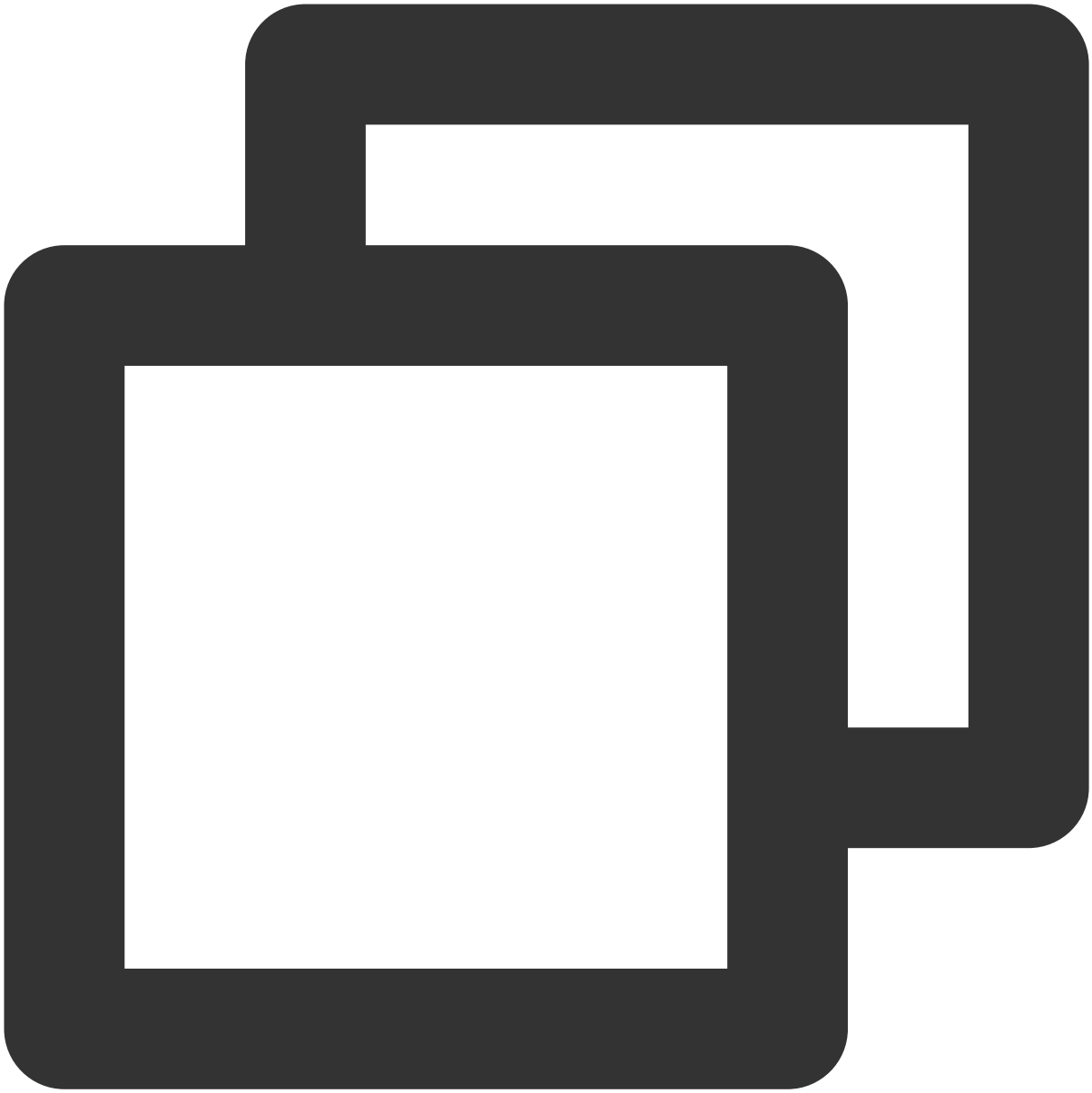
参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。



```
void onRequestJoinAnchor(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, String reason,
```

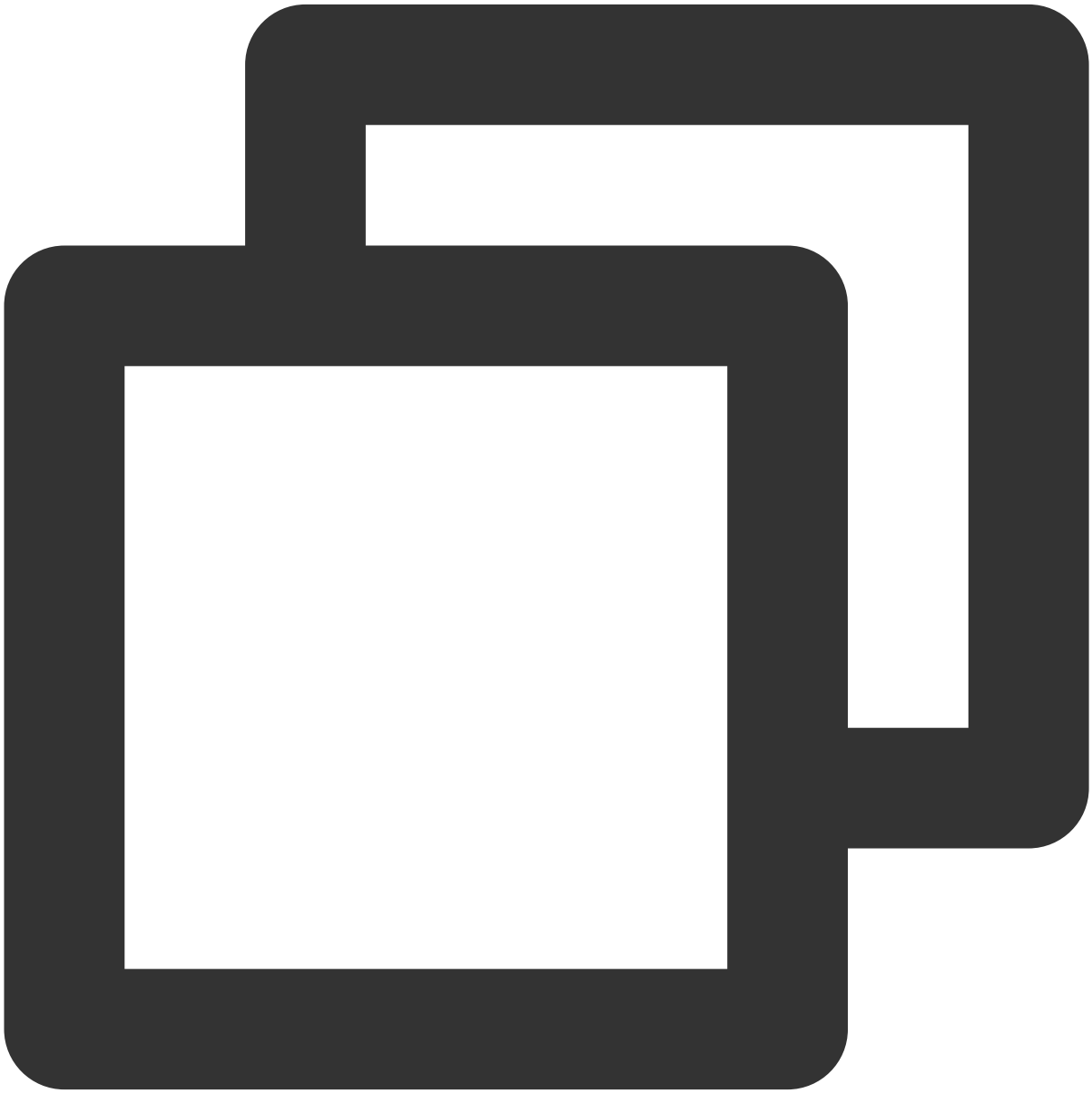
参数如下表所示：

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。

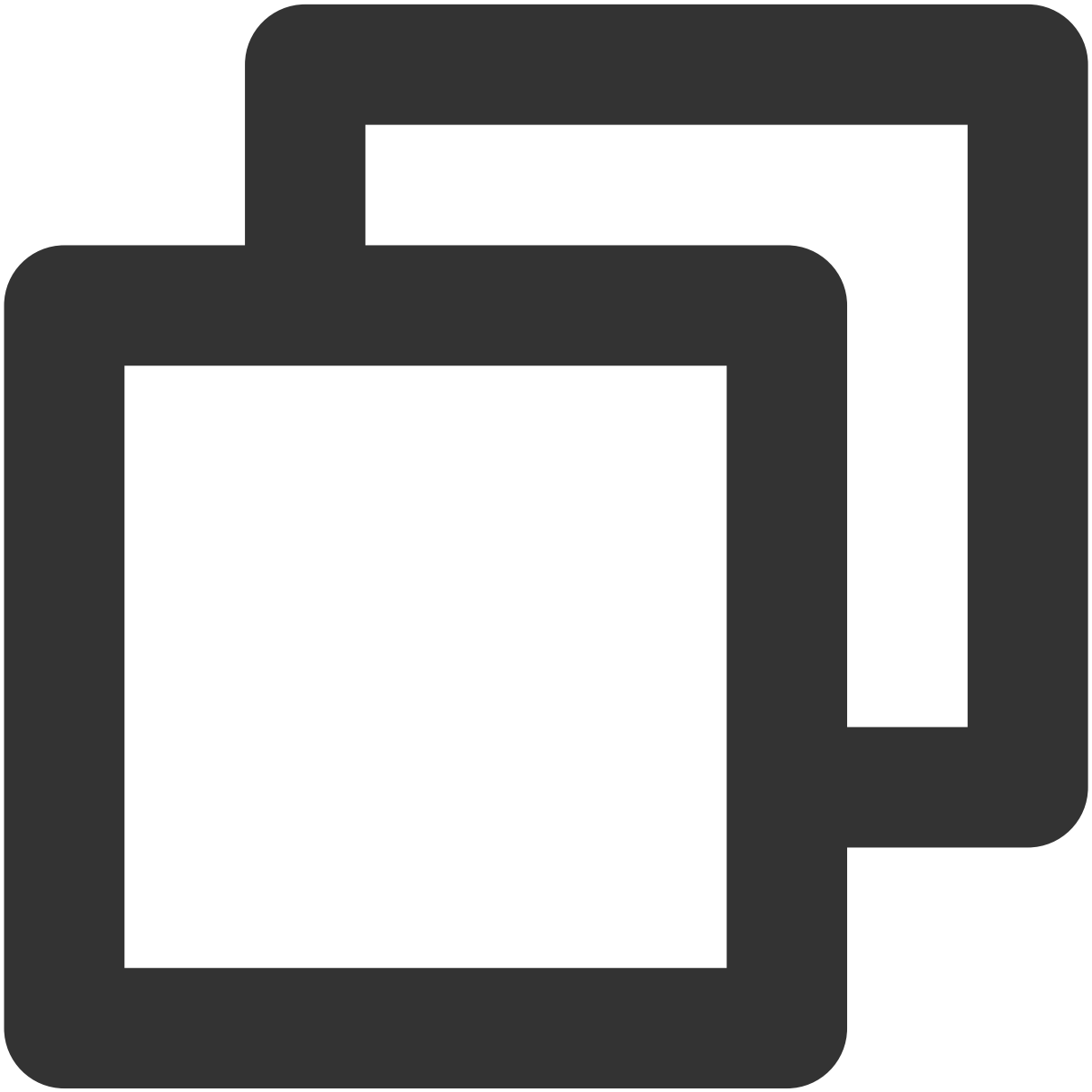


```
void onKickoutJoinAnchor();
```

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。



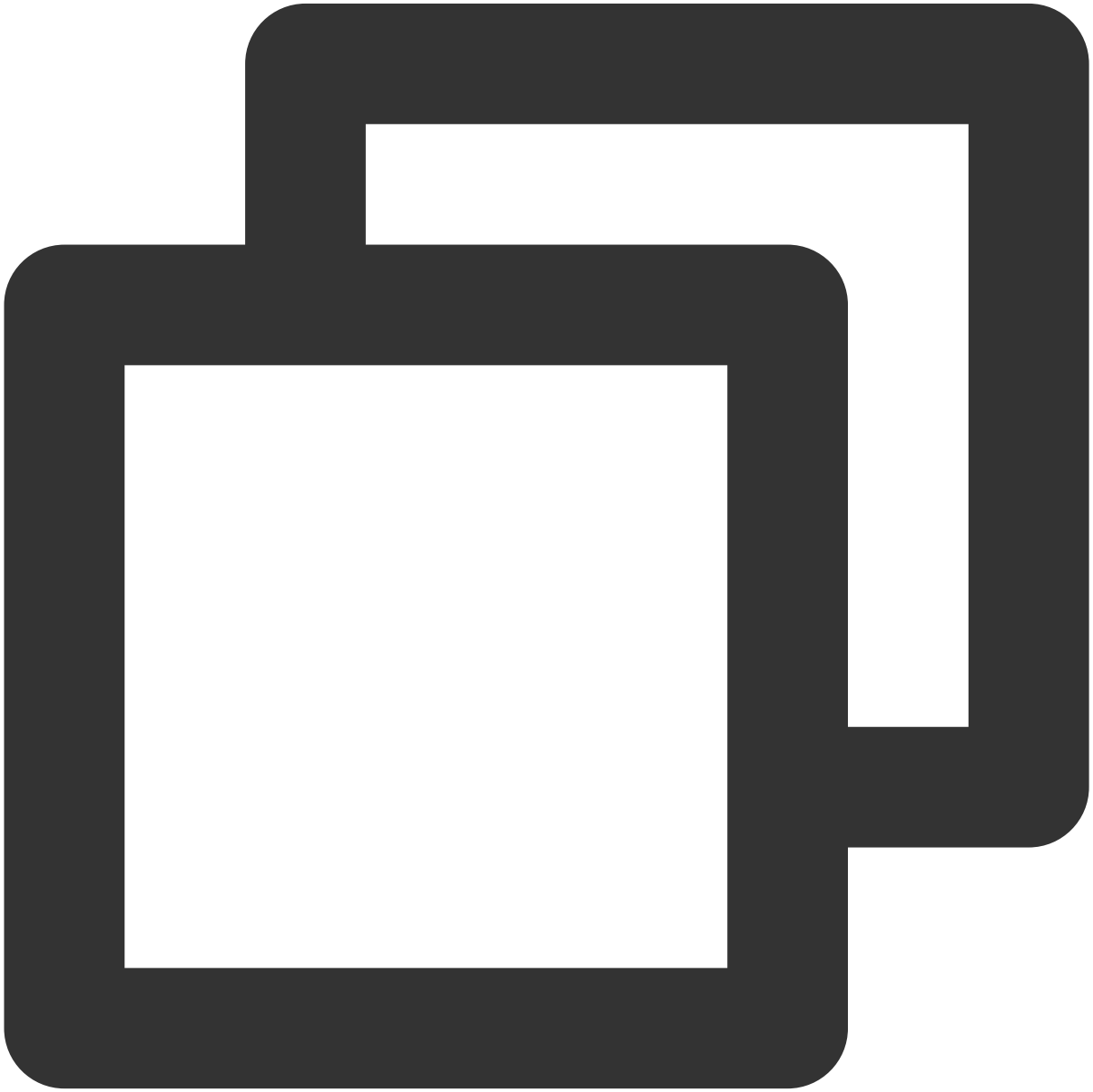
```
void onRequestRoomPK(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, int timeout);
```

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。

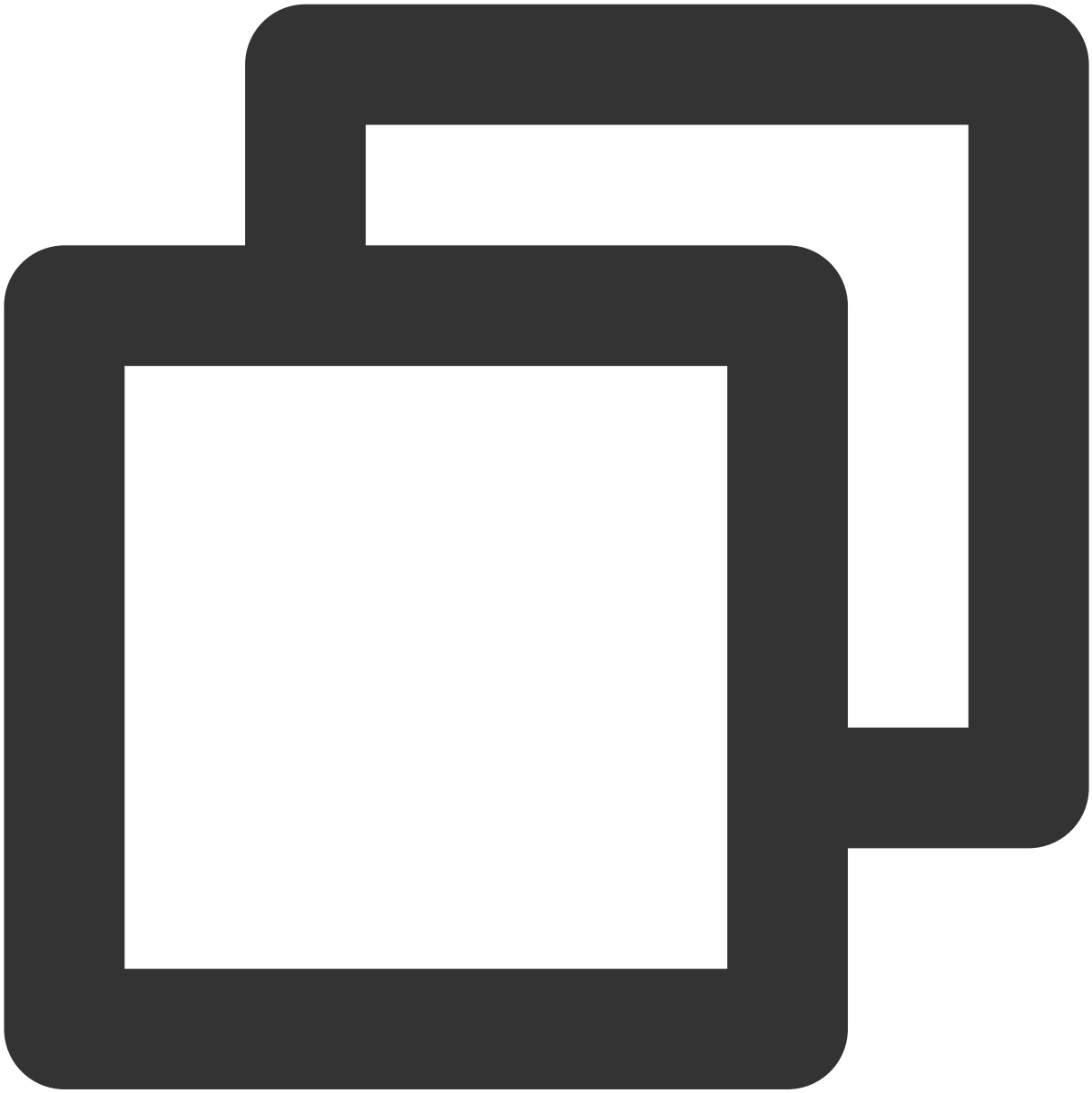


```
void onQuitRoomPK();
```

消息事件回调

onRecvRoomTextMsg

收到文本消息。



```
void onRecvRoomTextMsg(String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

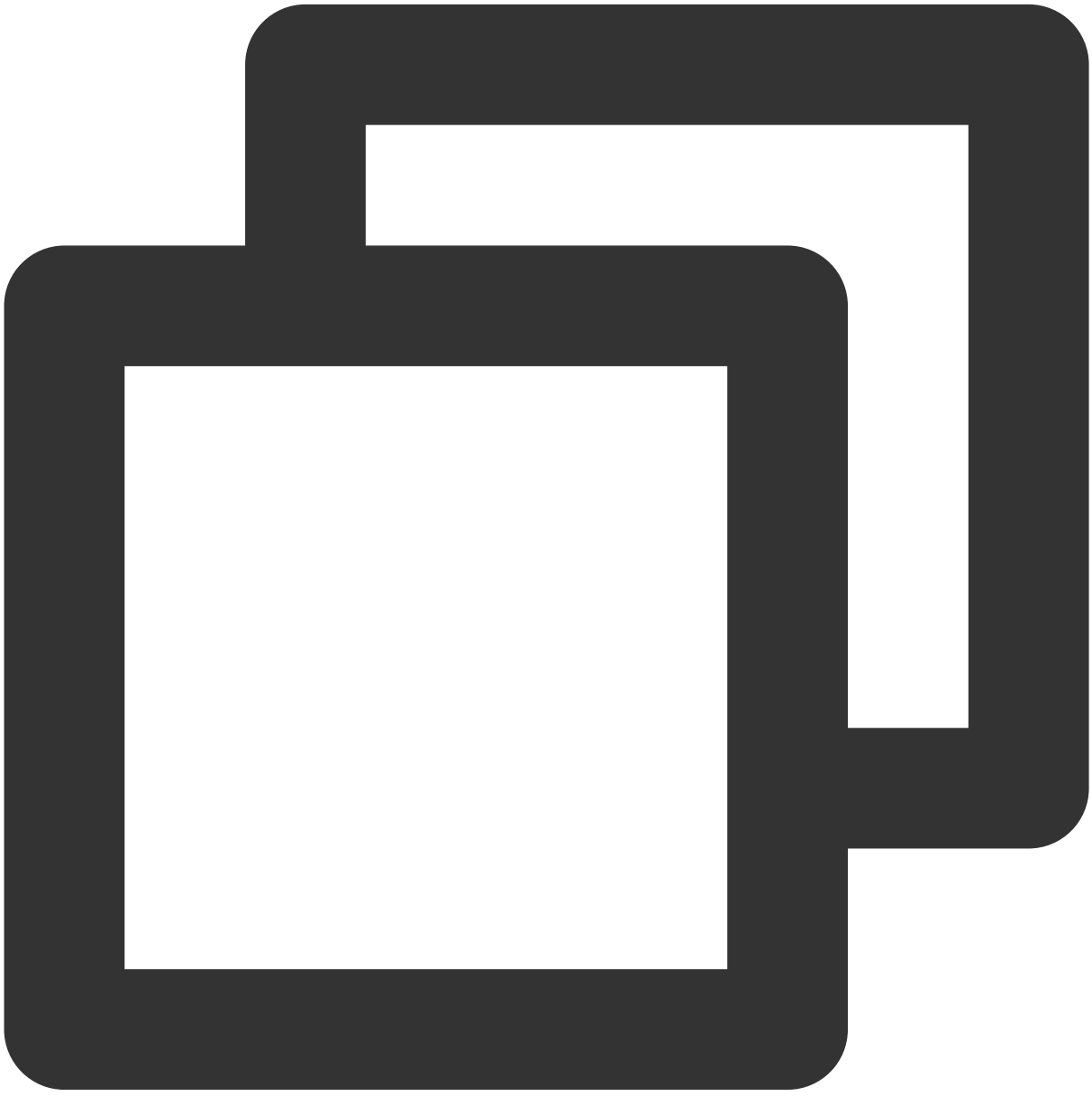
参数如下表所示：

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.TRTCLiveUserIn
```

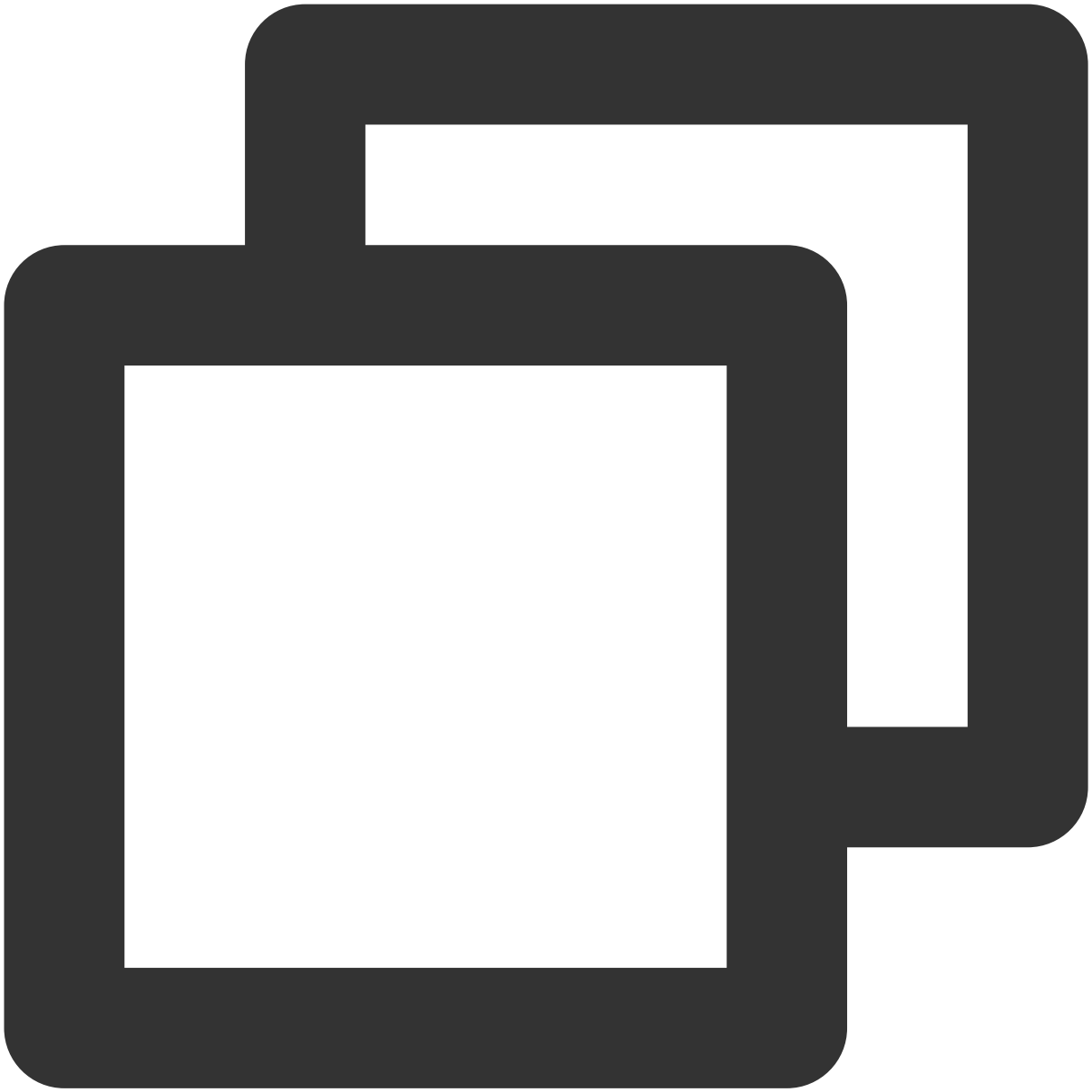
参数如下表所示：

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

TRTCAudioEffectManager

playBGM

播放背景音乐。



```
void playBGM(String url, int loopTimes, int bgmVol, int micVol, TRTCCloud.BGMNotify
```

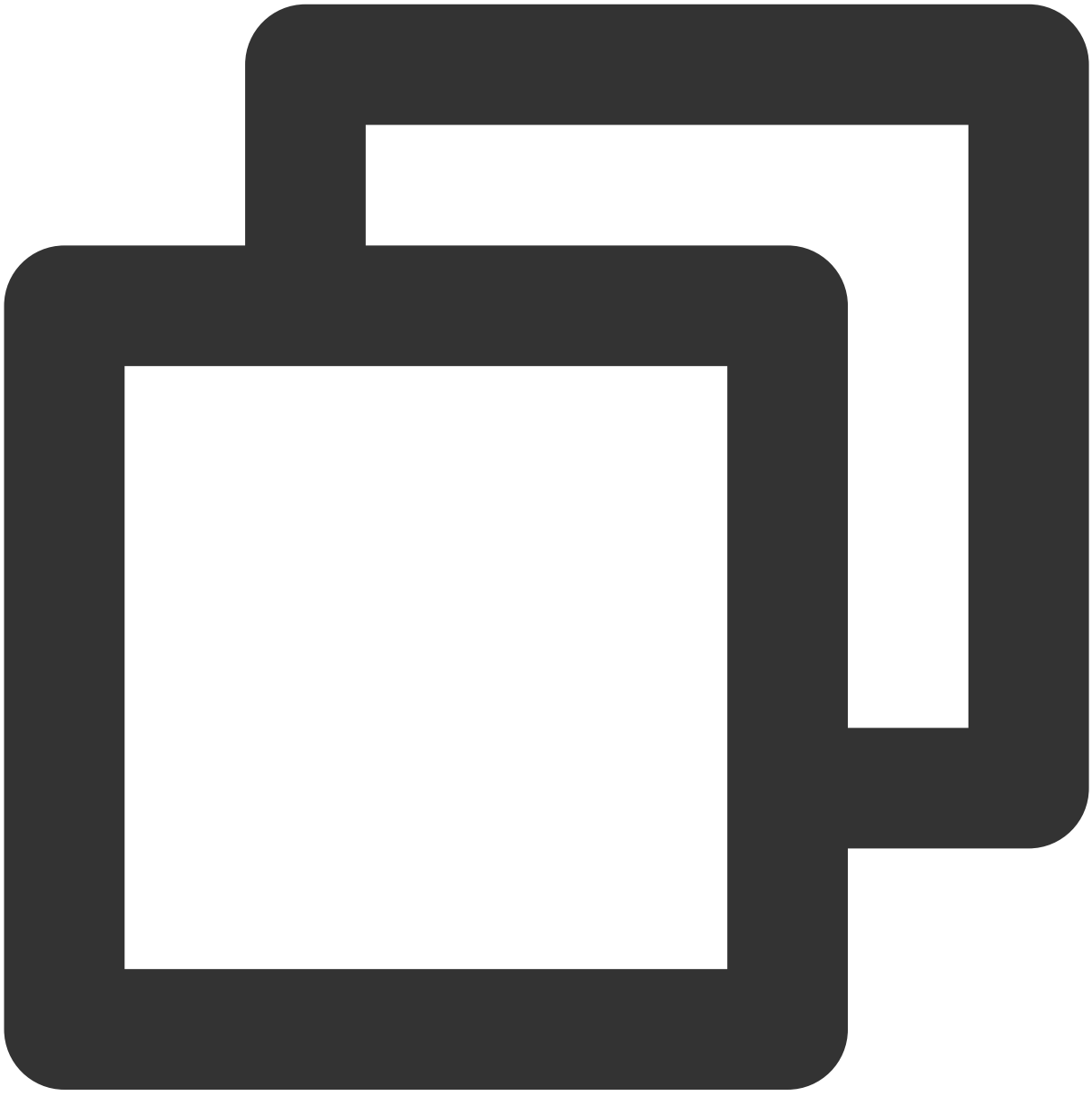
参数如下表所示：

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

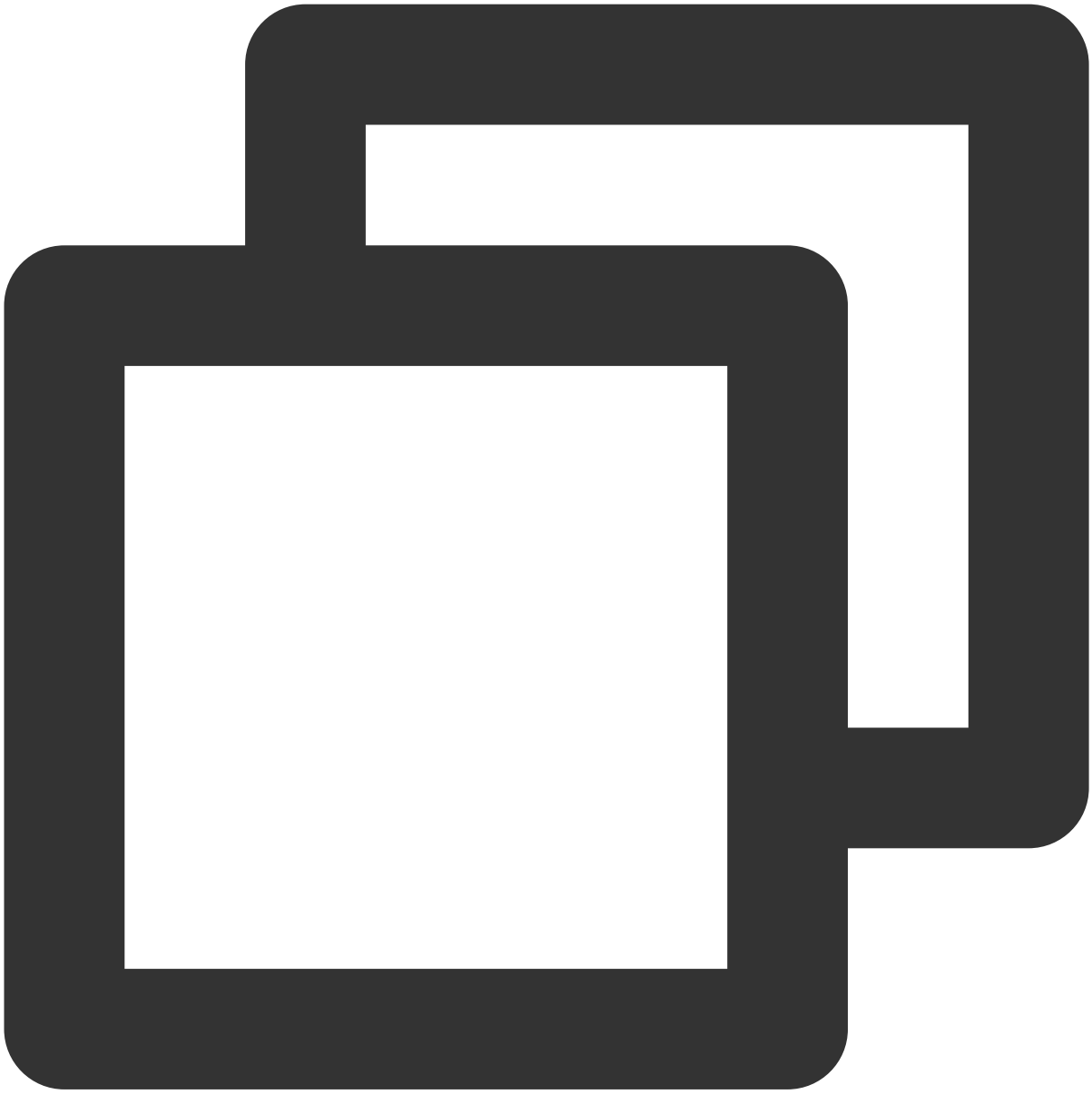
停止播放背景音乐。



```
void stopBGM();
```

pauseBGM

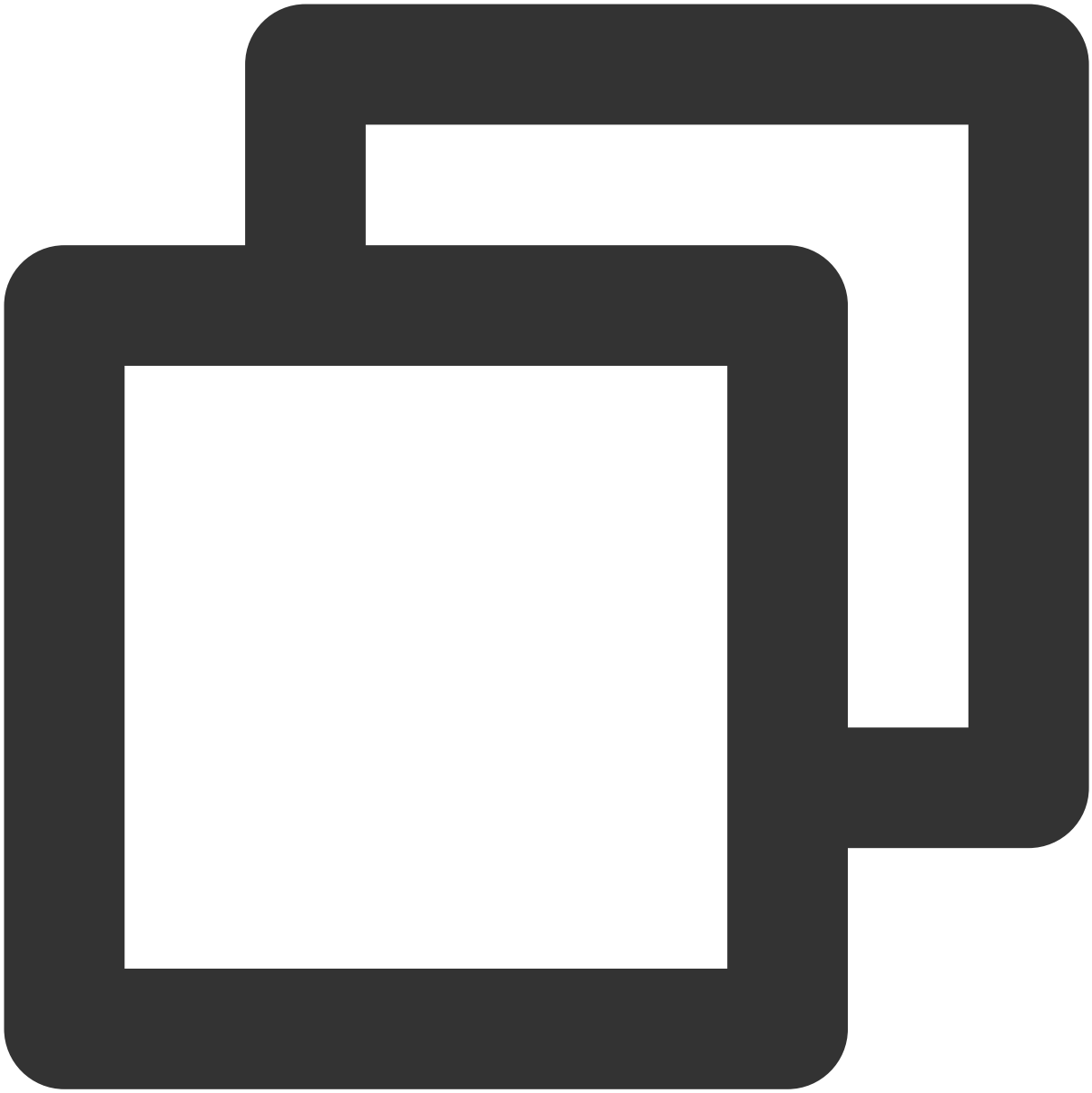
暂停播放背景音乐。



```
void pauseBGM();
```

resumeBGM

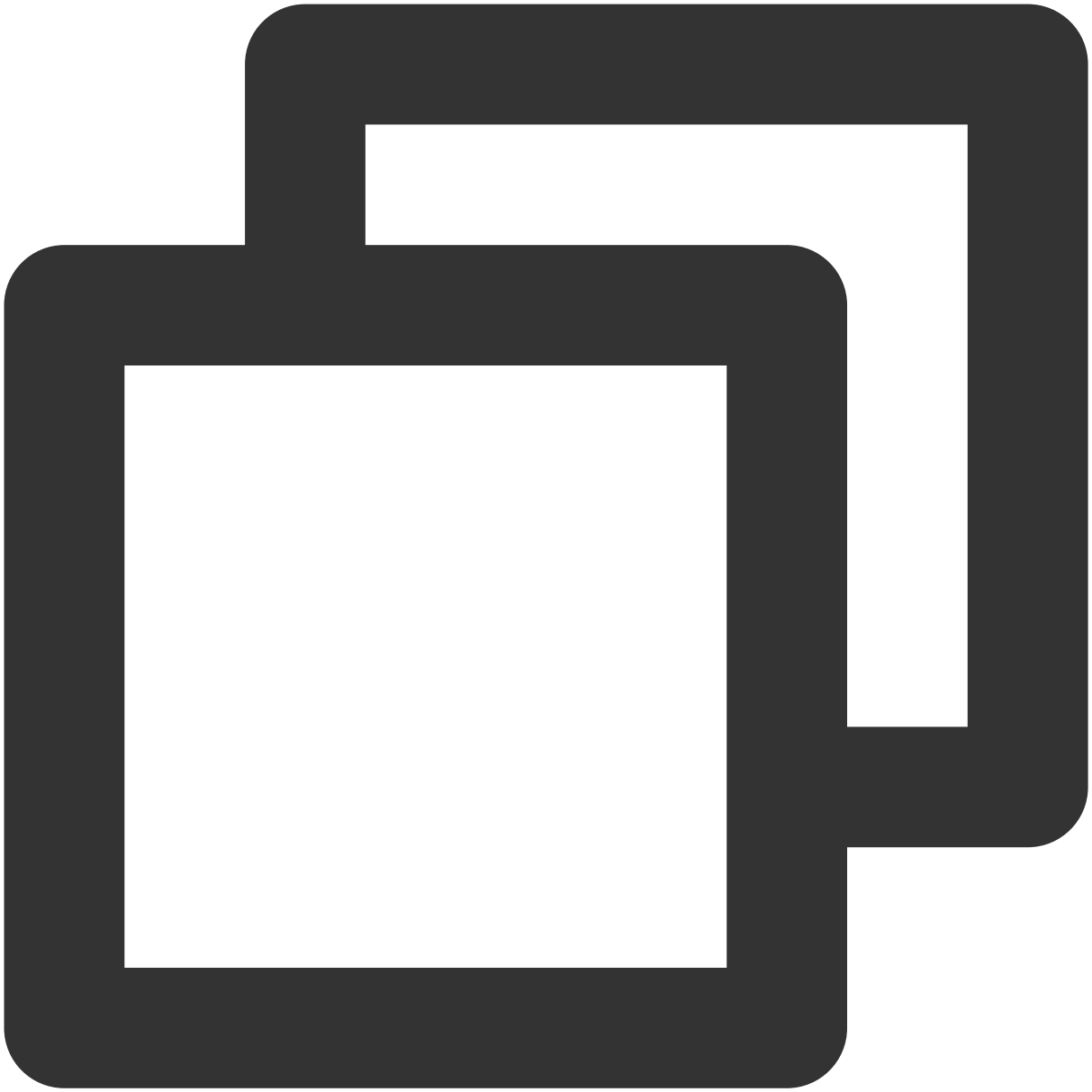
继续播放背景音乐。



```
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。



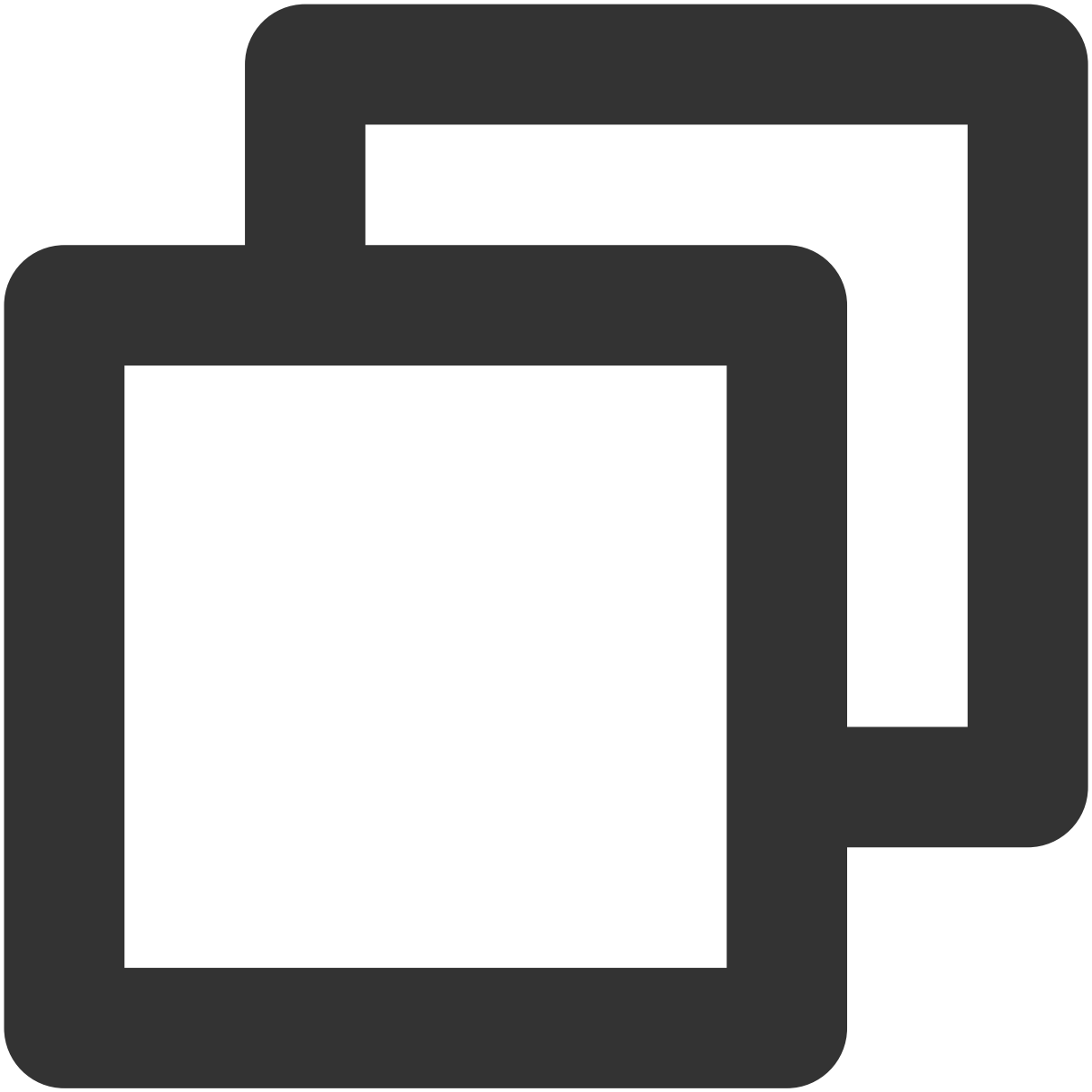
```
void setBGMVolume(int volume);
```

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：

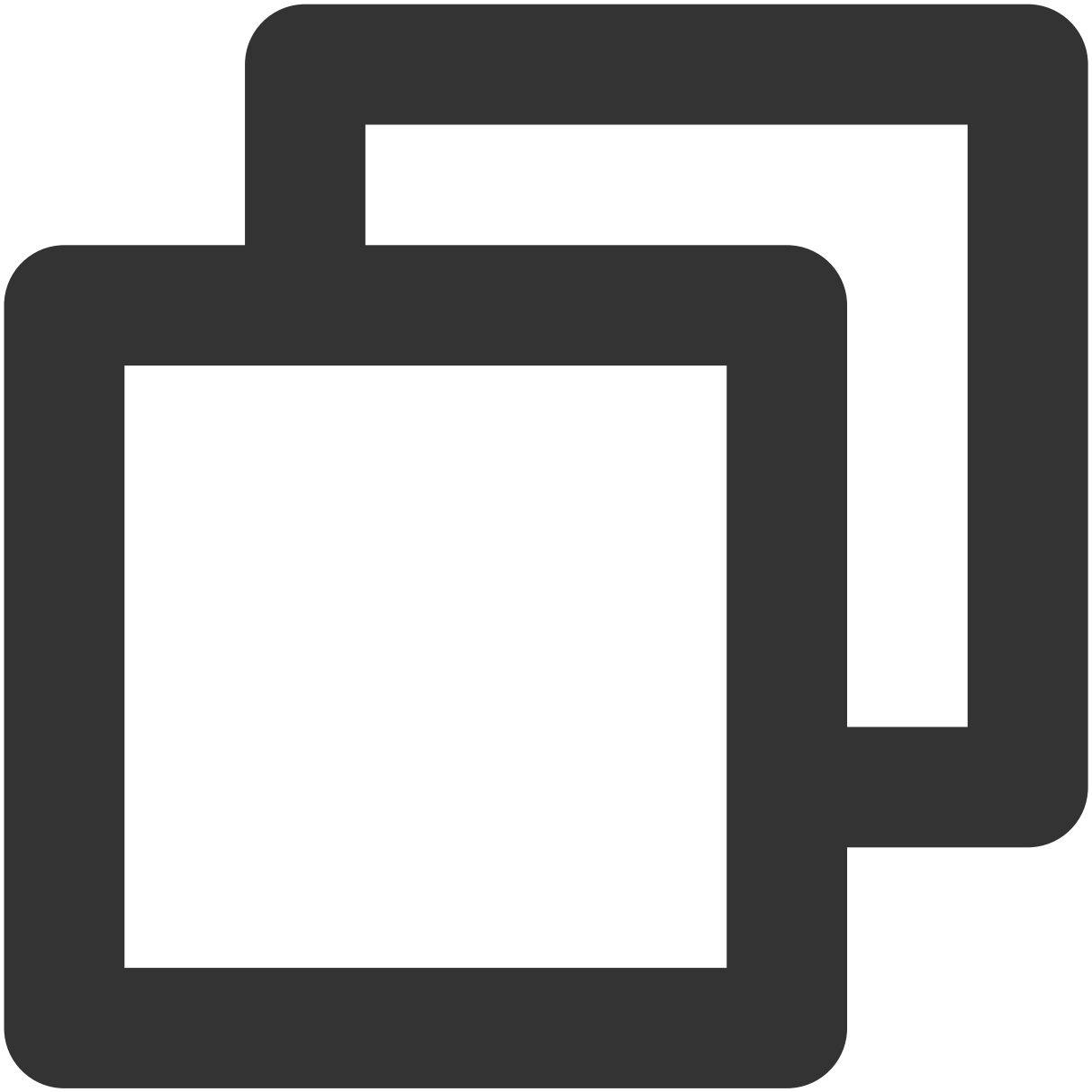
参数	类型	含义
position	int	背景音乐播放进度，单位为毫秒（ms）。

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。



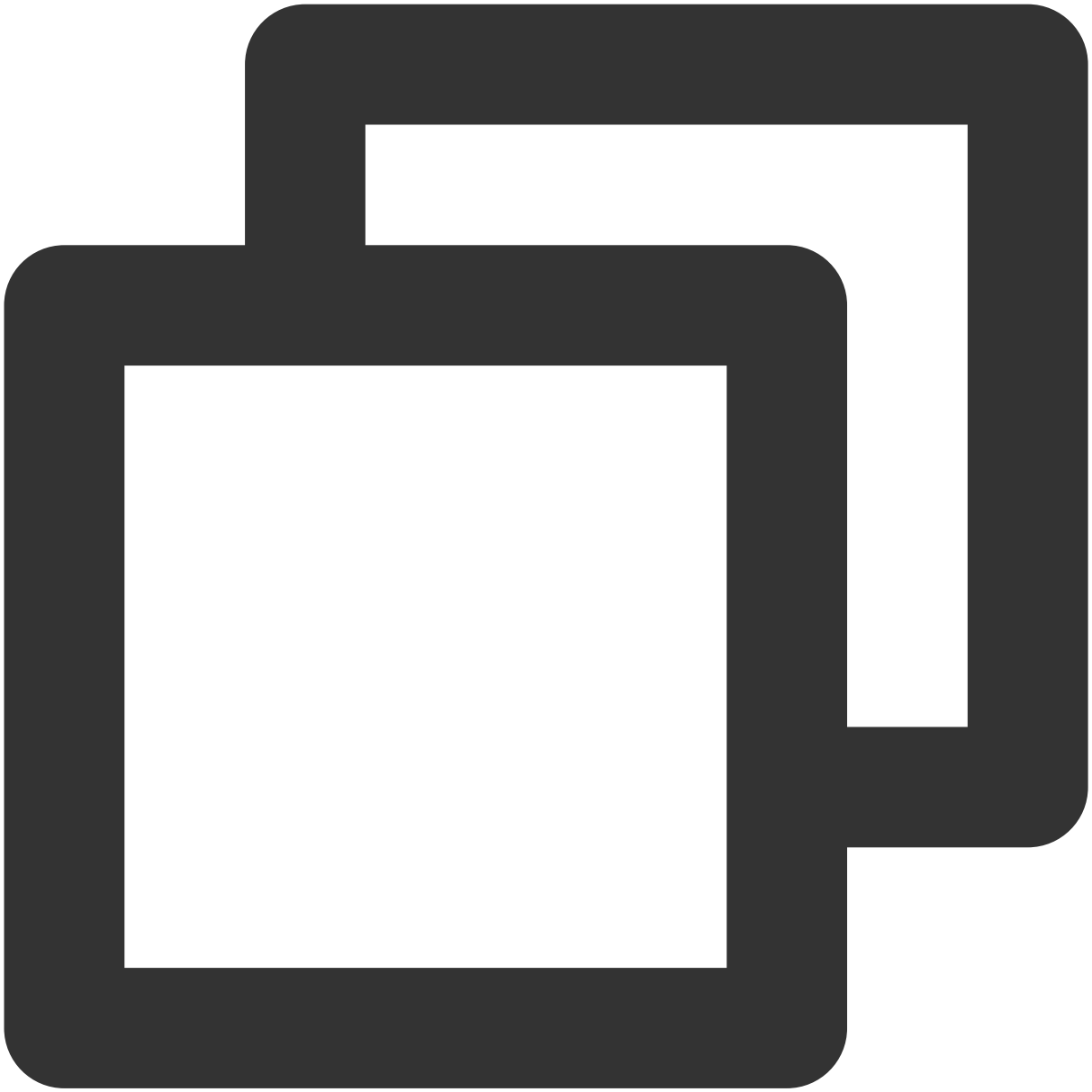
```
void setMicVolume(int volume);
```

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。



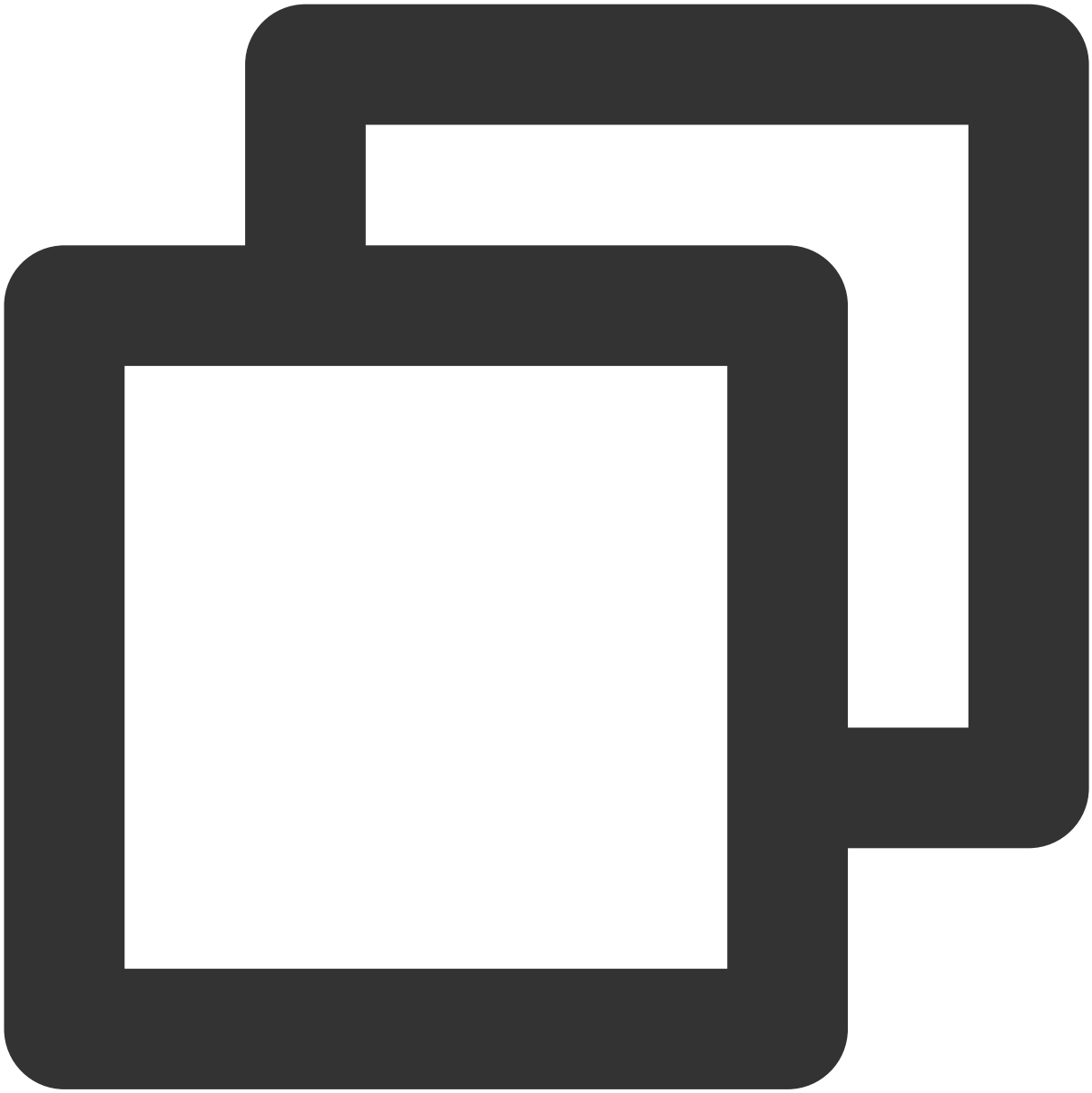
```
void setReverbType(int reverbType);
```

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。



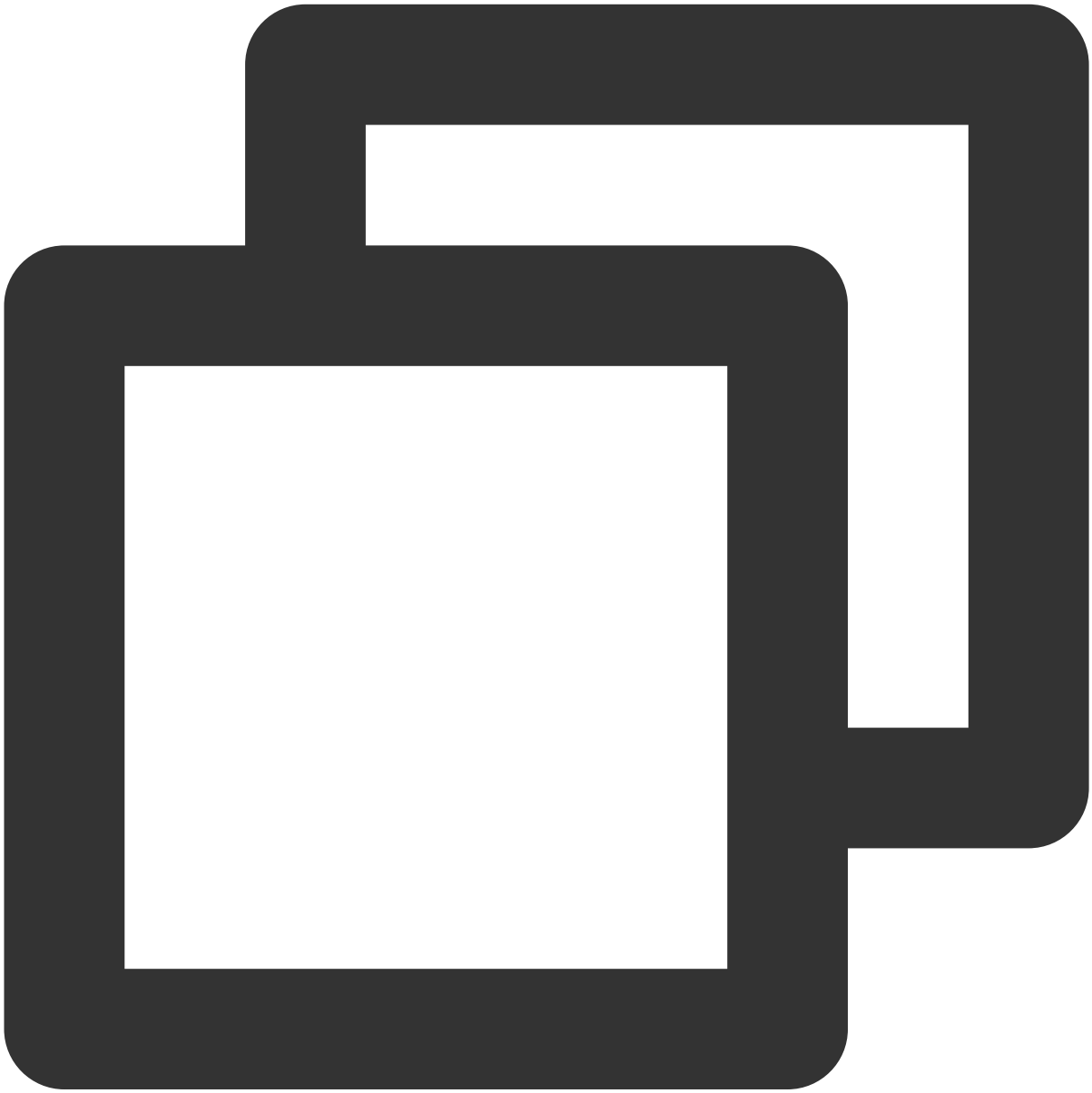
```
void setVoiceChangerType(int type);
```

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 vol
```

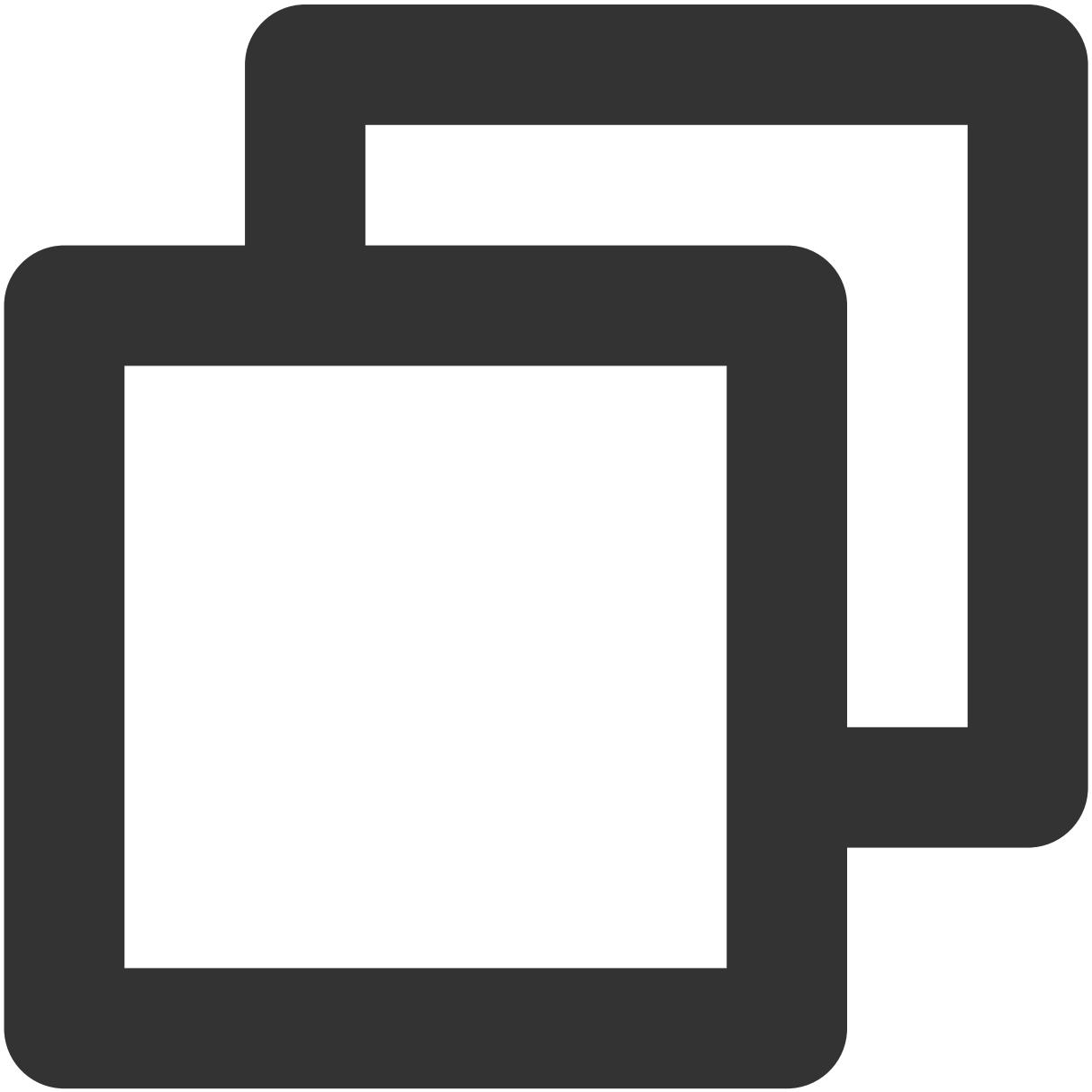
参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

path	String	音效路径。
count	int	循环次数。
publish	boolean	是否推送 / true 推送给观众, false 本地预览。
volume	int	音量大小，取值范围为0 - 100，默认值为100。

pauseAudioEffect

暂停音效播放。



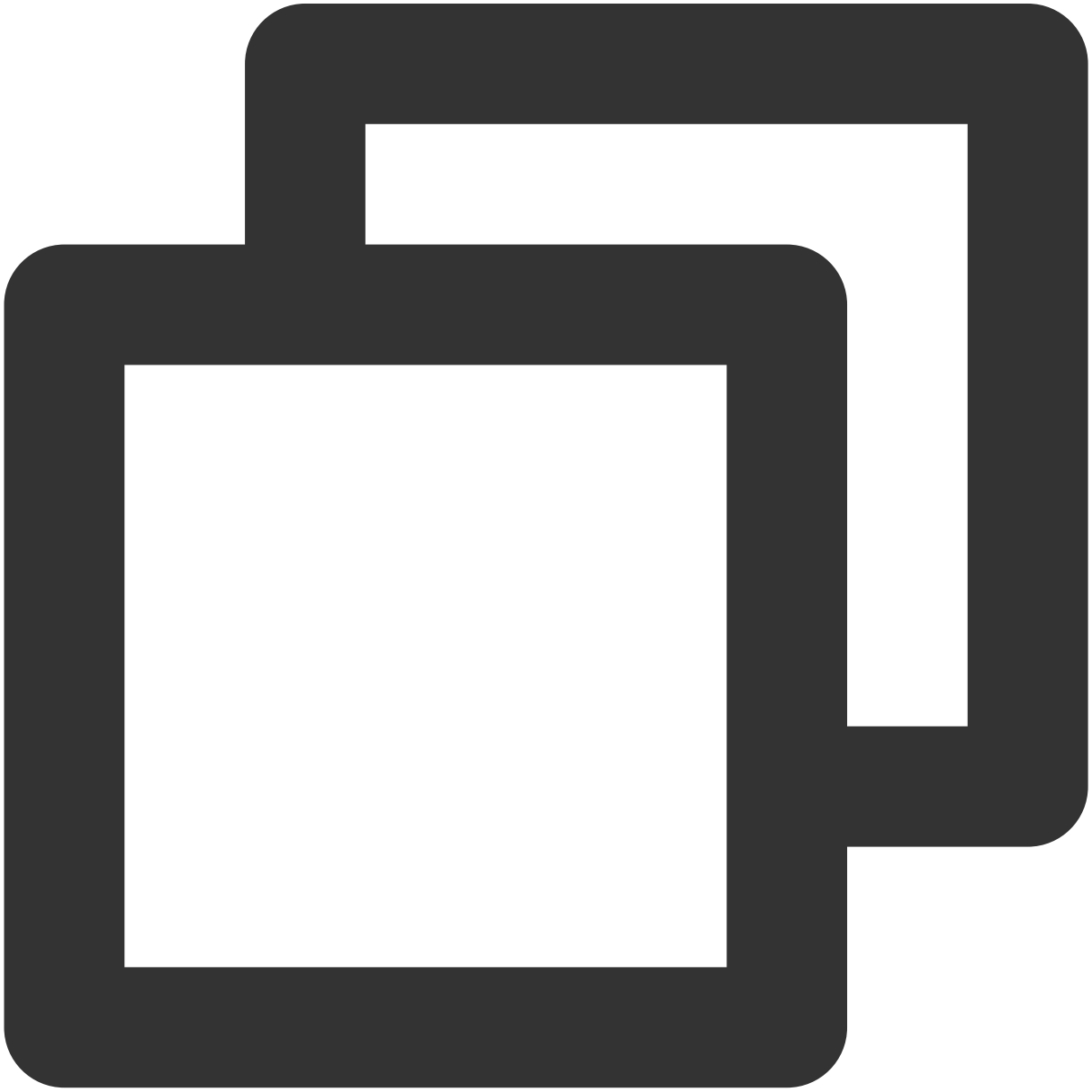
```
void pauseAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

resumeAudioEffect

恢复音效播放。



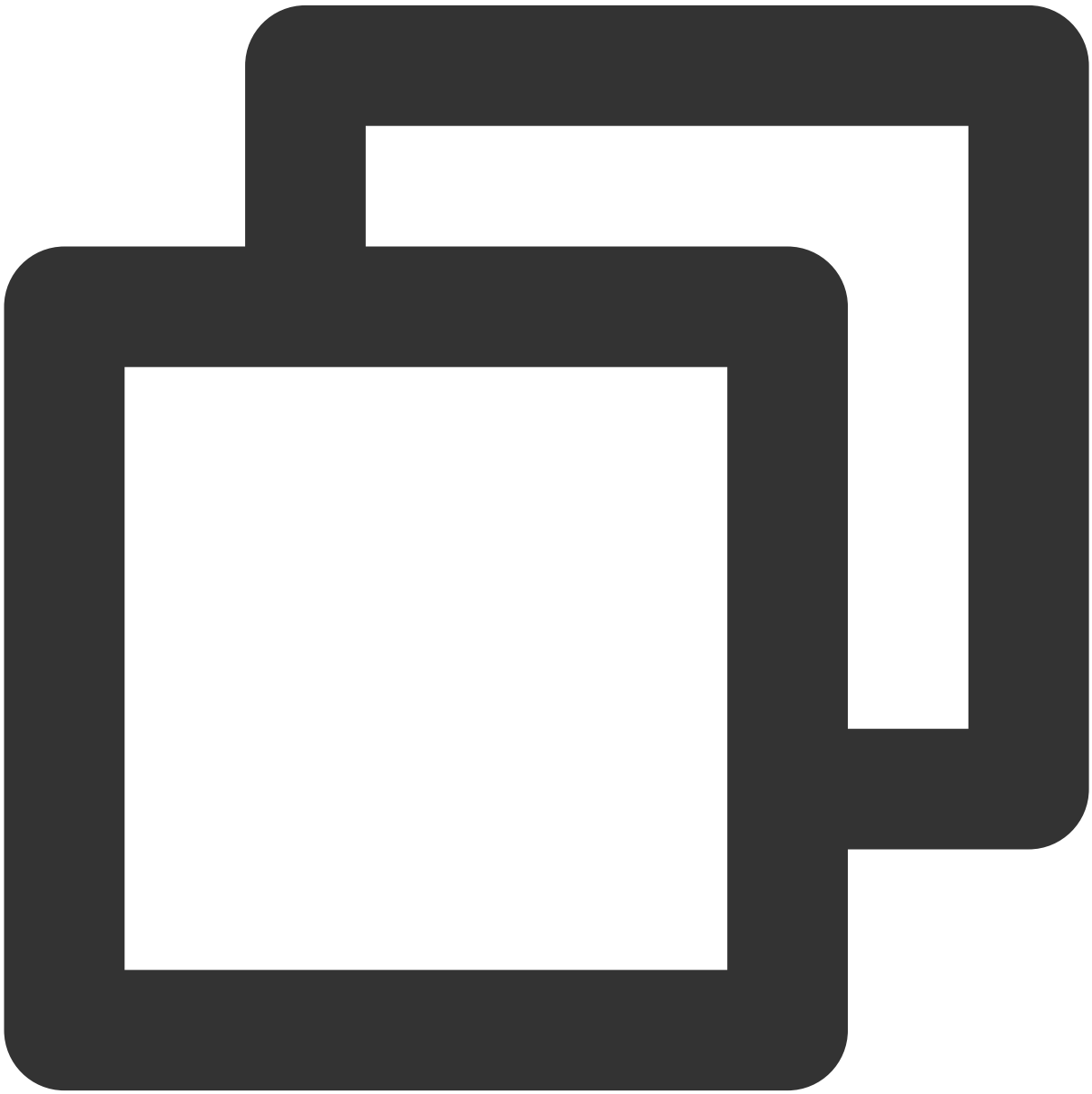
```
void resumeAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

stopAudioEffect

停止音效播放。



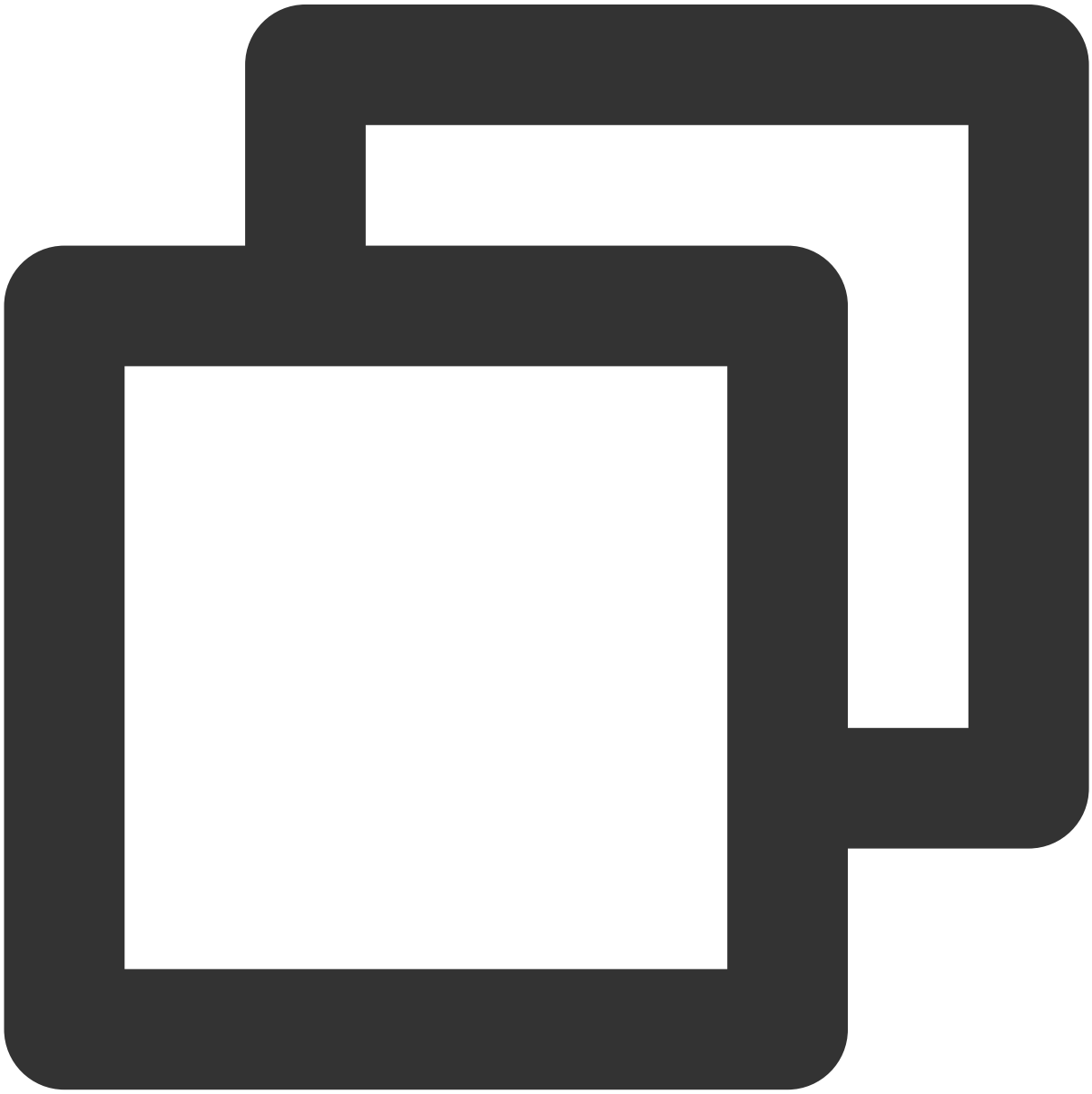
```
void stopAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

stopAllAudioEffects

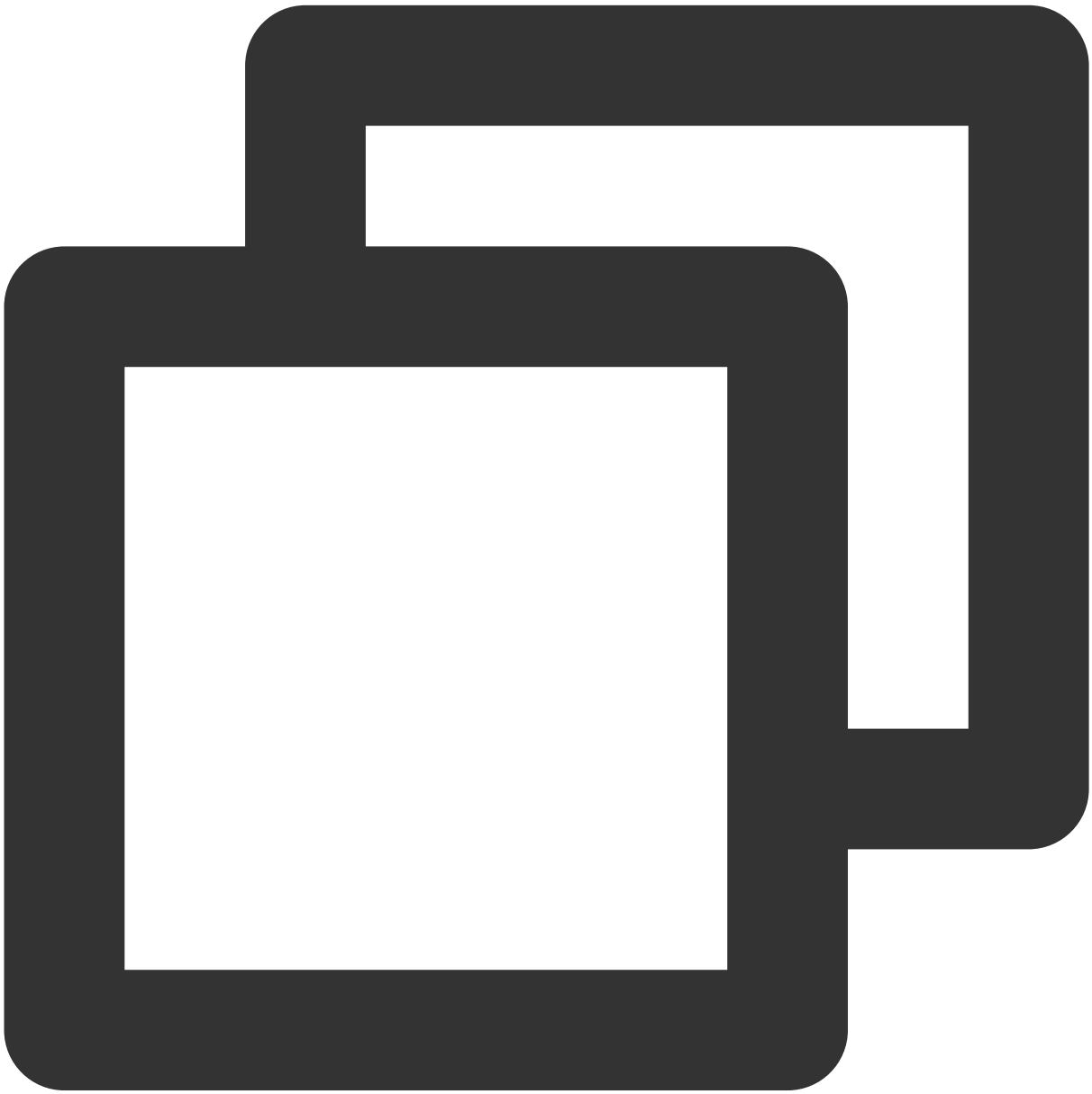
停止全部音效播放。



```
void stopAllAudioEffects();
```

setAudioEffectVolume

设置音效音量。



```
void setAudioEffectVolume(int effectId, int volume);
```

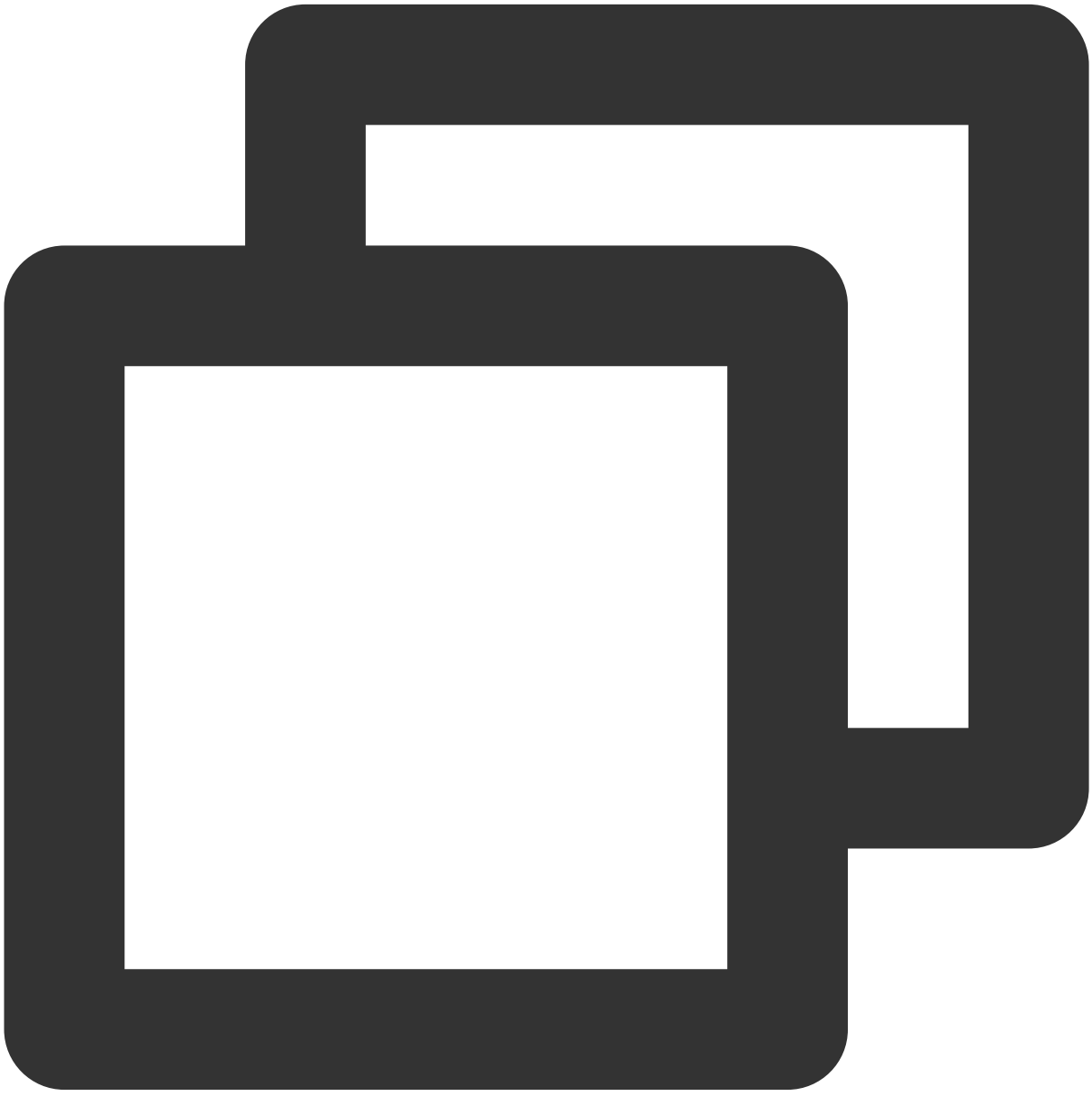
参数如下表所示：

参数	类型	含义

effectId	int	音效 ID。
volume	int	音量大小，取值范围为0 - 100，默认值为100。

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。