实时音视频

语音聊天室（含 UI）

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

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

【商标声明】

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

【服务声明】

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

文档目录

版权所有：腾讯云计算（北京）有限责任公司

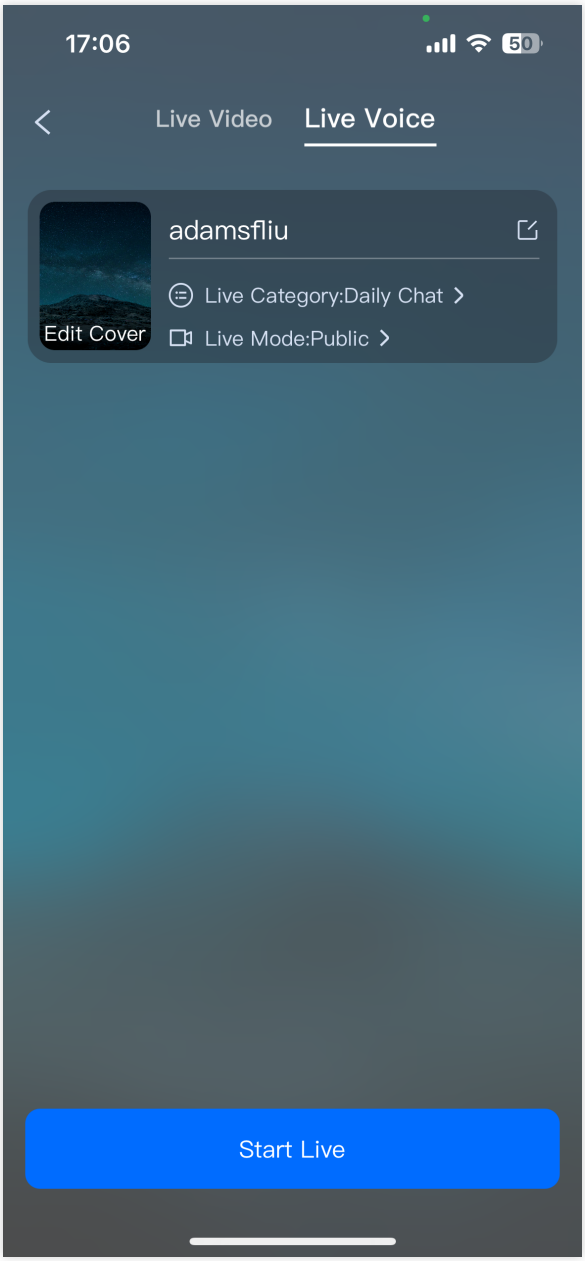
语音聊天室（含 UI）组件介绍（TUILiveKit）

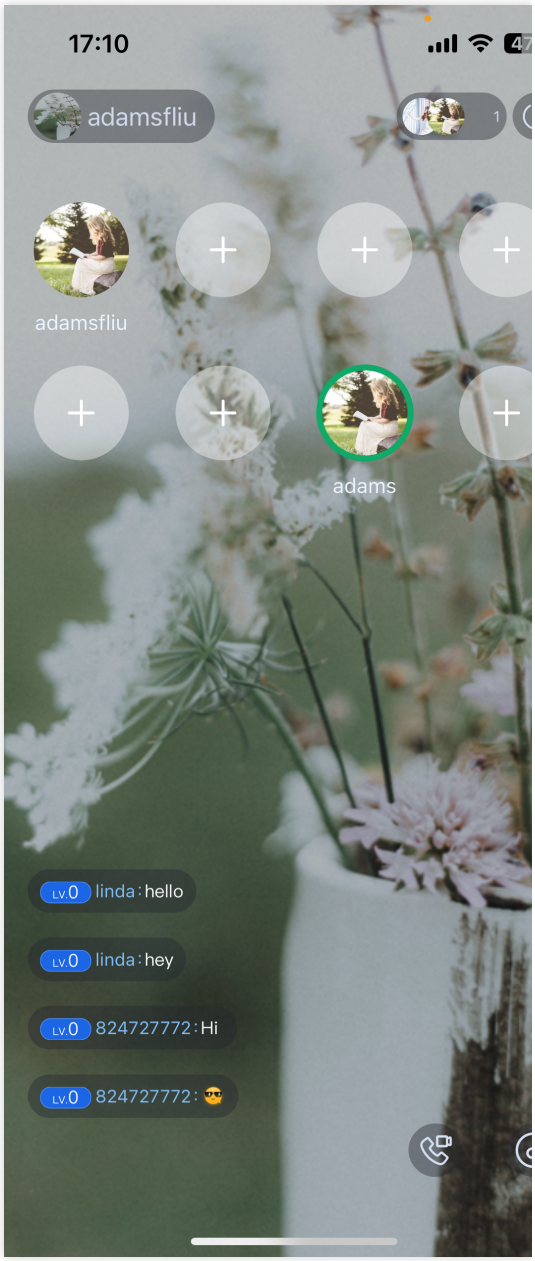
最近更新时间：2024-05-17 11:20:40

组件介绍

TUILiveKit 是一款适用于社交娱乐、游戏互动、实时合唱，语音电台等视频互动直播和语音聊天场景的产品，通过集成该产品，仅需三步，30分钟内就可以为您的应用添加语聊房间、互动连麦、送礼、房间管理，麦位管理等功能，快速上线业务。基本功能展示如下图：

语音直播预览页	语音直播主播页





支持平台

平台	Android	iOS
是否支持		
支持语言/框架	Java Kotlin	Swift Objective-C

功能介绍

基本功能	高级功能	功能优势

语音聊天室 直播观看 观众连麦 麦位管理 查看观众列表 成员管理：拉黑禁言	聊天弹幕 心动点赞 互动礼物（全屏礼物） 音效&变声	**完善的 UI 交互** **专业级直播 3A 算法，更好的音质** **丰富的 REST API**

在线体验

平台	Android	iOS
demo 集成	Github: Andorid	Github: iOS

交流与反馈

如果您有任何需求或反馈，可以联系：info_rtc@tencent.com。

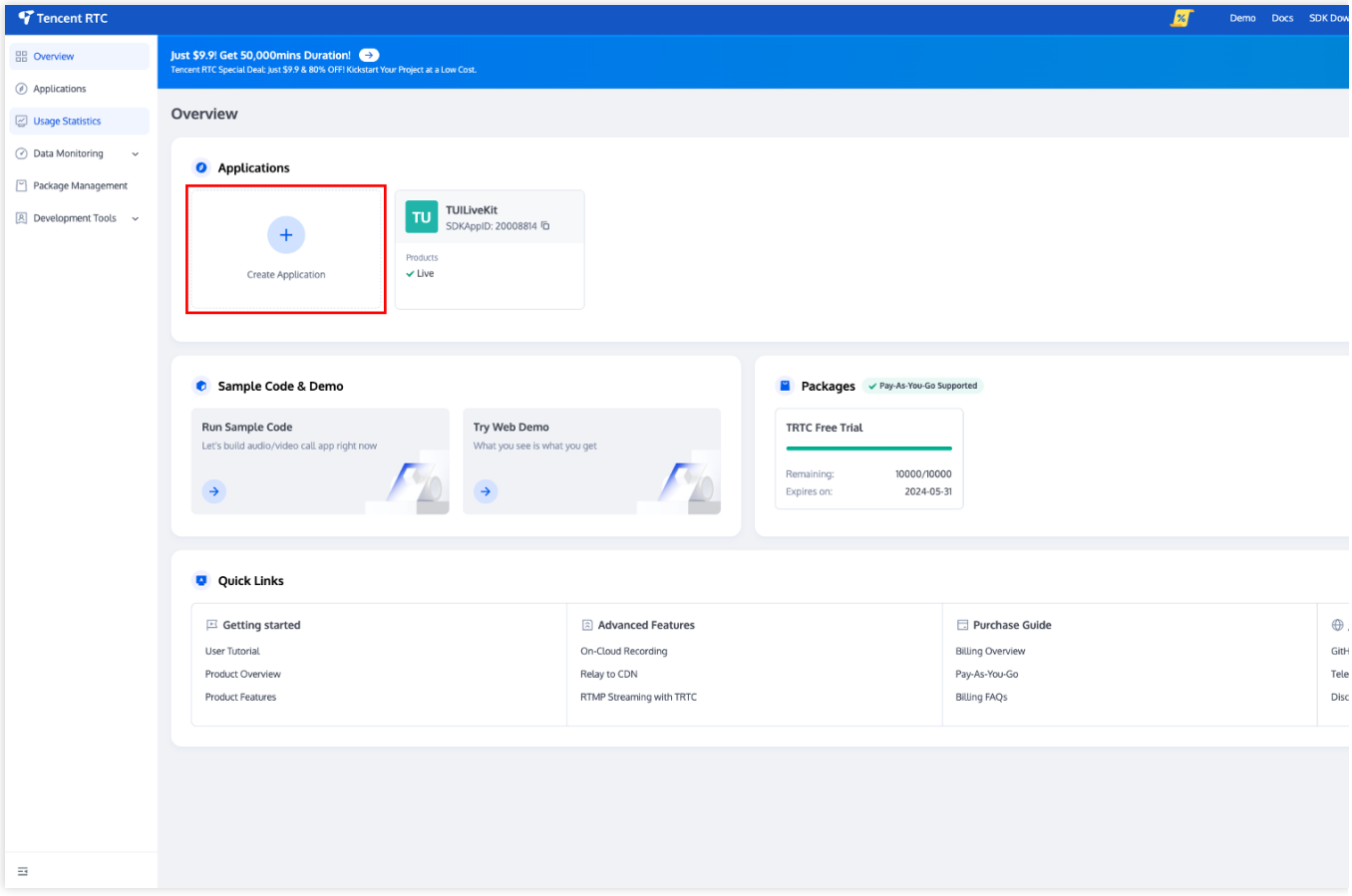
开通服务（TUILiveKit）

最近更新时间：2024-05-17 11:20:40

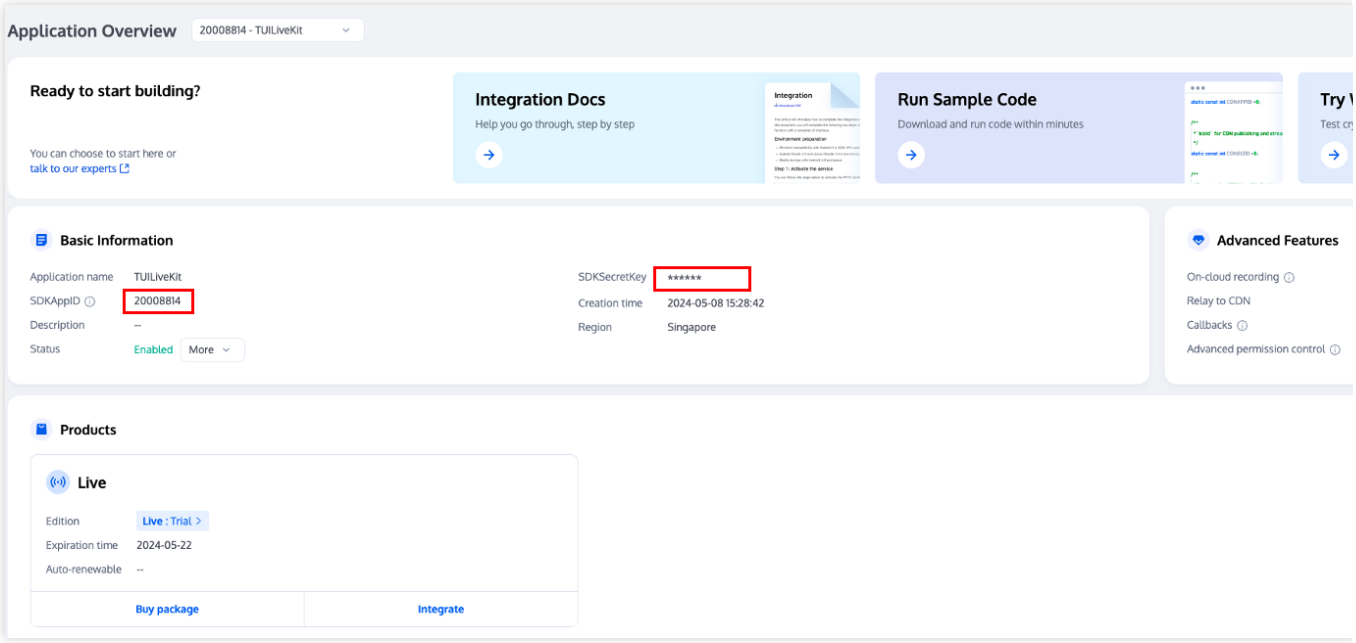
开通体验服务

为了您能更好地体验互动直播 Live 的功能，我们免费为每个 SDKAppID 提供了14天体验版（体验版不额外赠送通话时长），每个 SDKAppID 可免费体验2次，每次有效期均为14天，同时一个账号下所有 SDKAppID 的体验总次数为10次。

1. 登录到 实时音视频 TRTC 控制台，点击 **Create Application**。



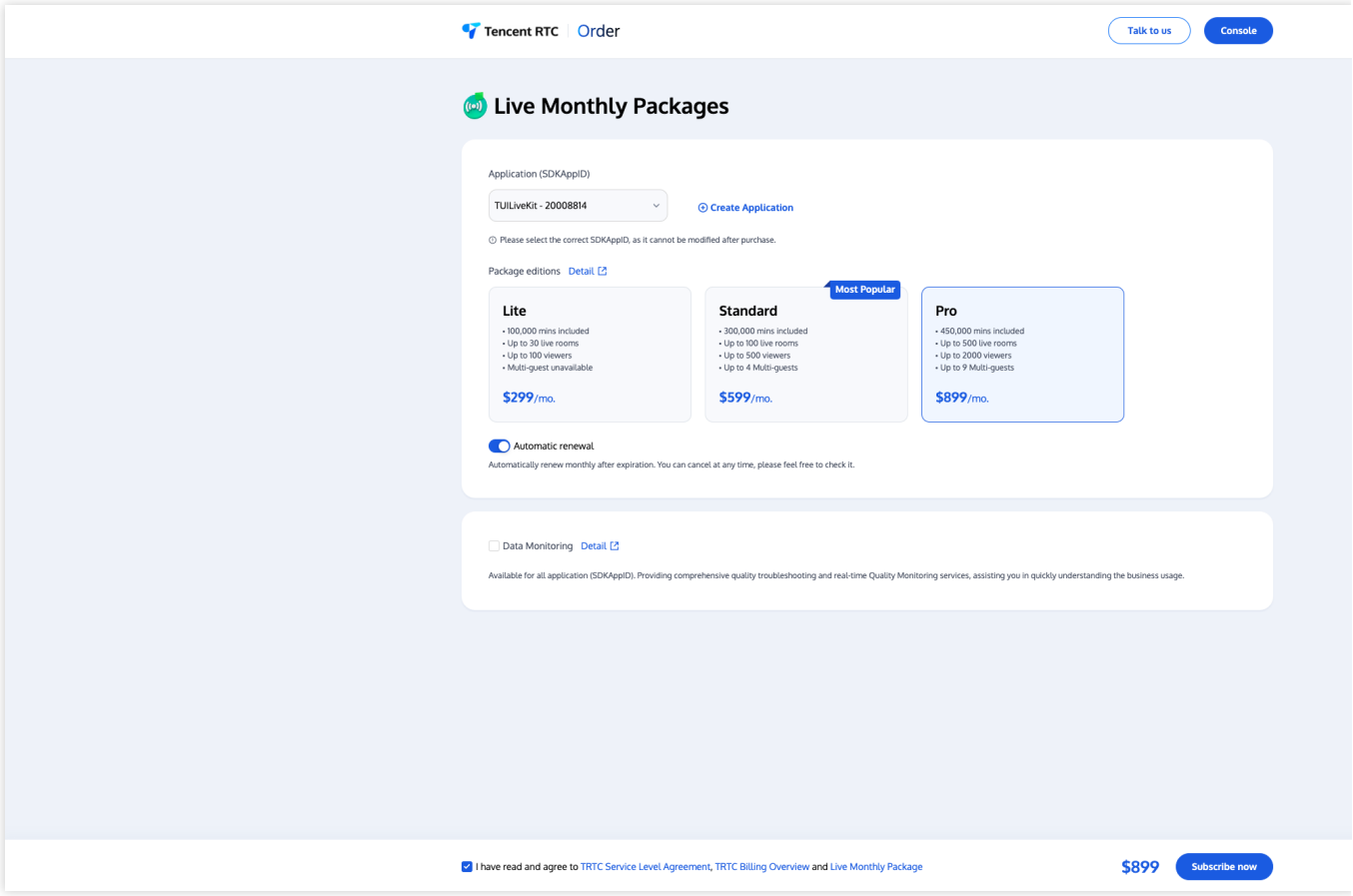
2. 在创建弹窗中输入**应用名称**，选择 **Live，** 并选择合适的 **Region**，点击 **Create**，即可完成体验版的创建。
3. 创建完成后，您就可以在当前页面查看应用的详细信息，比如应用的 `SDKAppID` 、 `SDKSecretKey` 等等信息，可前往 TRTC 控制台 查看应用版本信息，并可参见 快速接入（TUILiveKit） 进行集成。



购买正式版

在正式上线前，您需要购买正式版的 Live 包月套餐，Live 包月套餐的价格和功能对比详情可参见 Live 包月套餐计费说明，套餐包购买流程如下：

1. 访问 Live 购买页面，选择需要购买的应用（SDKAppID）及版本，**同时建议您开启自动续期以避免影响业务使用**。确认购买信息后，勾选协议内容并点击 **Subscribe now**。



说明：

针对语音聊天室业务，建议您购买 Pro 版本，支持的麦位更多，玩法更灵活！

2. 前往订单确认页确认商品信息，并完成支付即可。

快速接入（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:41

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频或语音直播功能。

环境准备

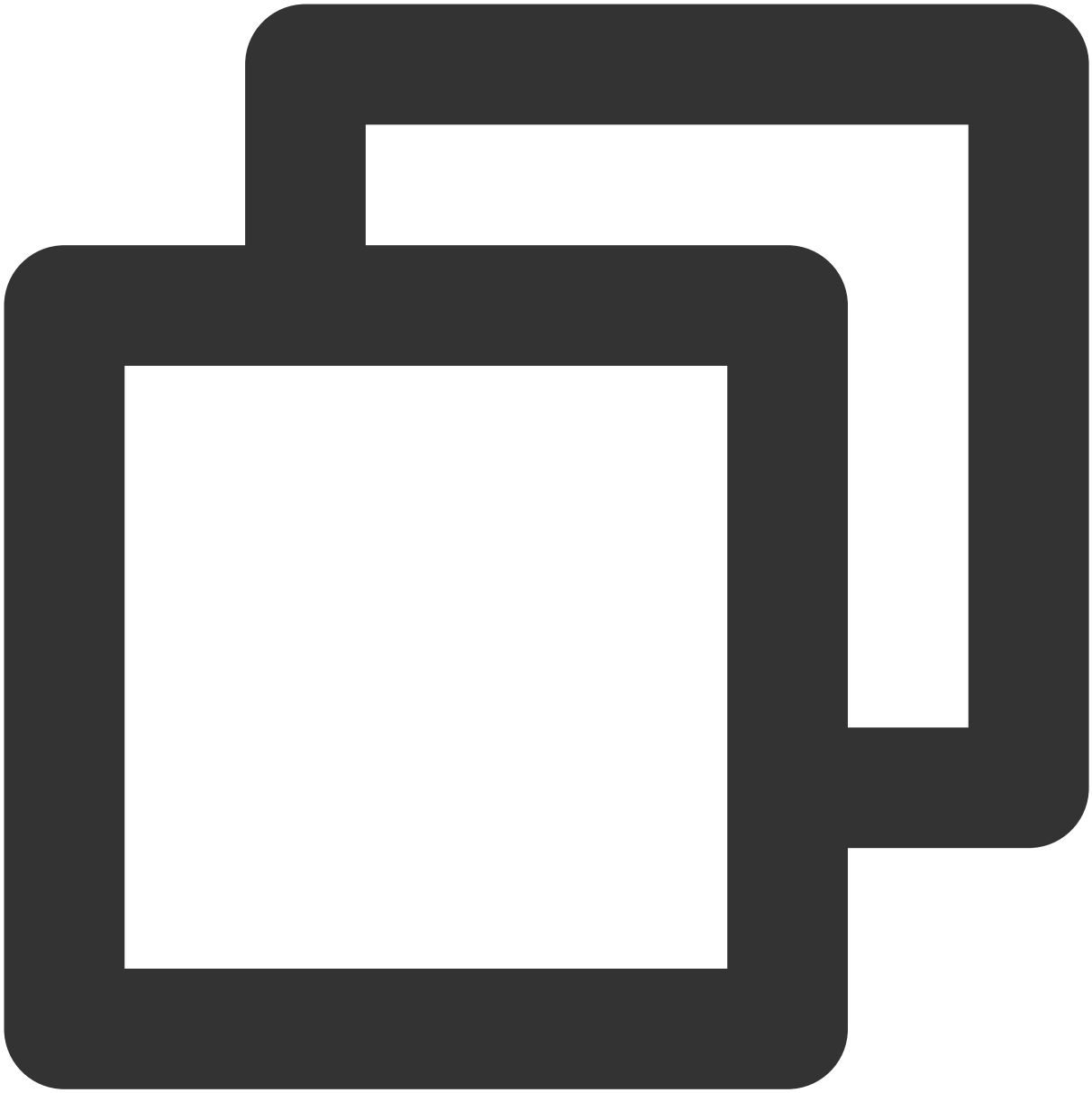
Xcode 13 及以上
iOS 13.0 及以上
CocoaPods环境安装，点击查看。
如果您的接入和使用中遇到问题，请参见 常见问题。

步骤一：开通服务

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

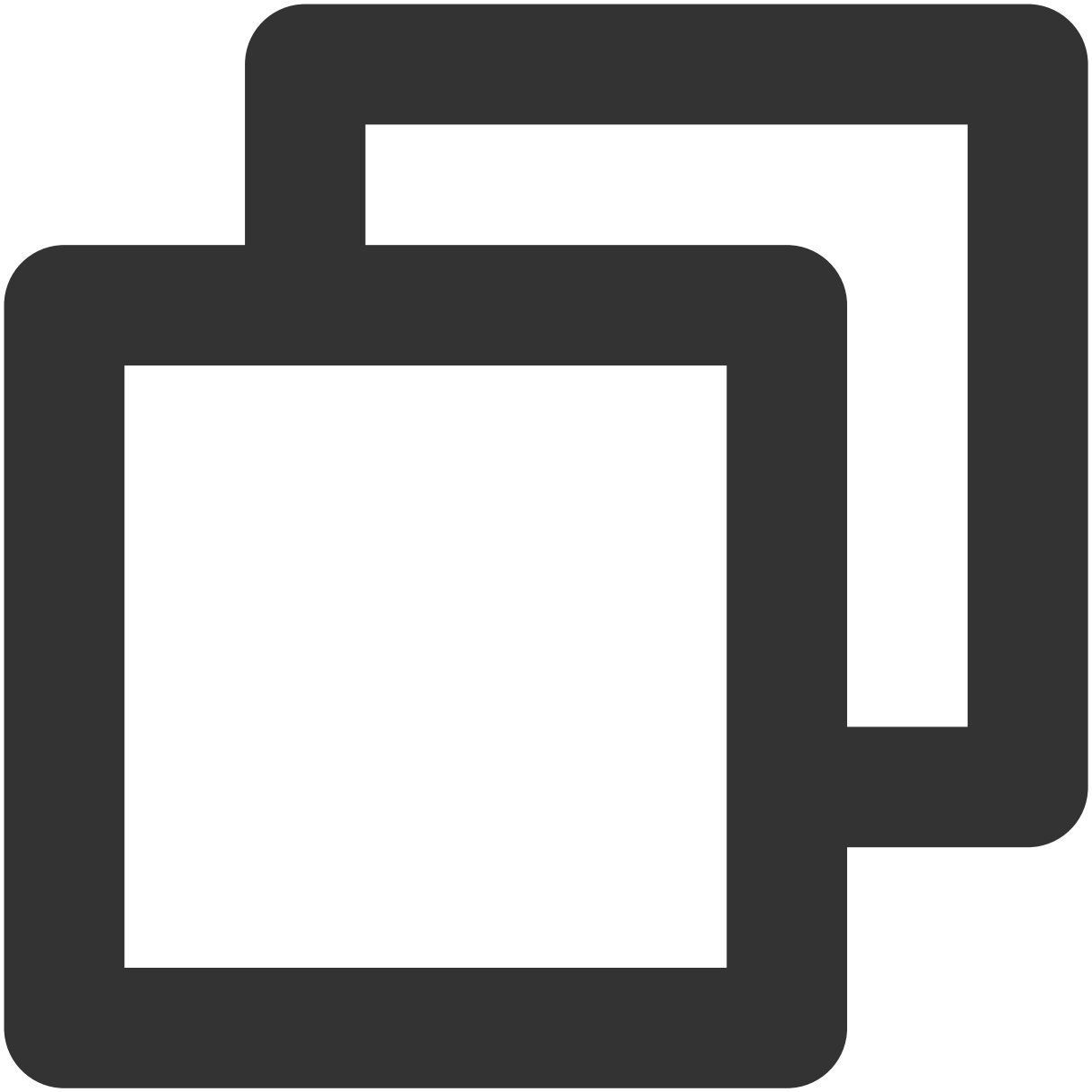
步骤二：导入 TUILiveKit 组件

使用 CocoaPods 导入组件，如果您遇到问题，请先参见 环境准备。导入组件具体骤如下：
1. 请在您的 `Podfile` 文件中添加 `pod 'TUILiveKit'` 依赖，如果您遇到任何问题，请参见 Example 工程。
Swift



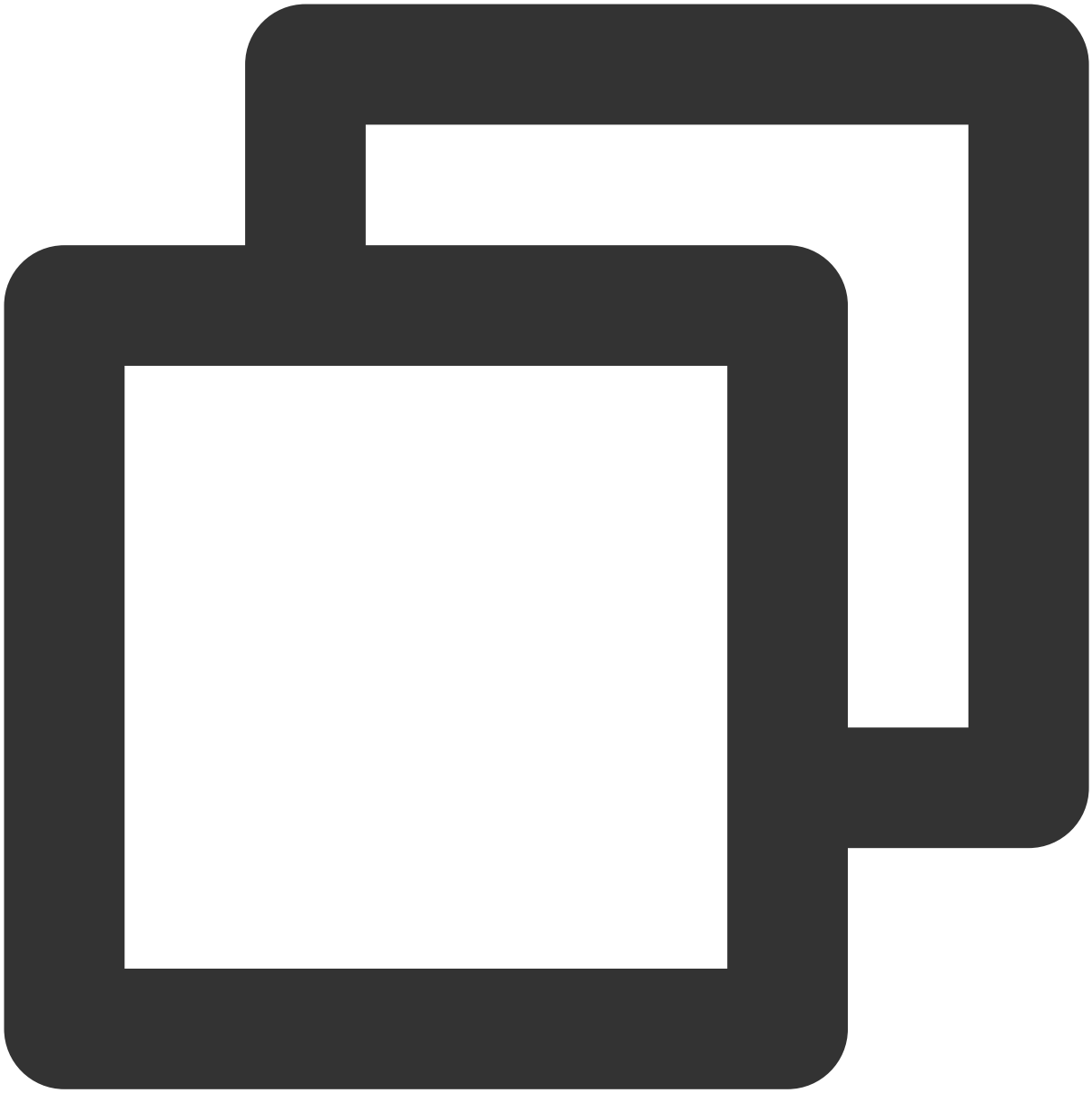
```
target 'xxxx' do
  ...
  ...
  pod 'TUILiveKit'
end
```

如果您没有 `Podfile` 文件，首先终端 `cd` 到 `xxxx.xcodeproj` 目录，然后通过以下命令创建：



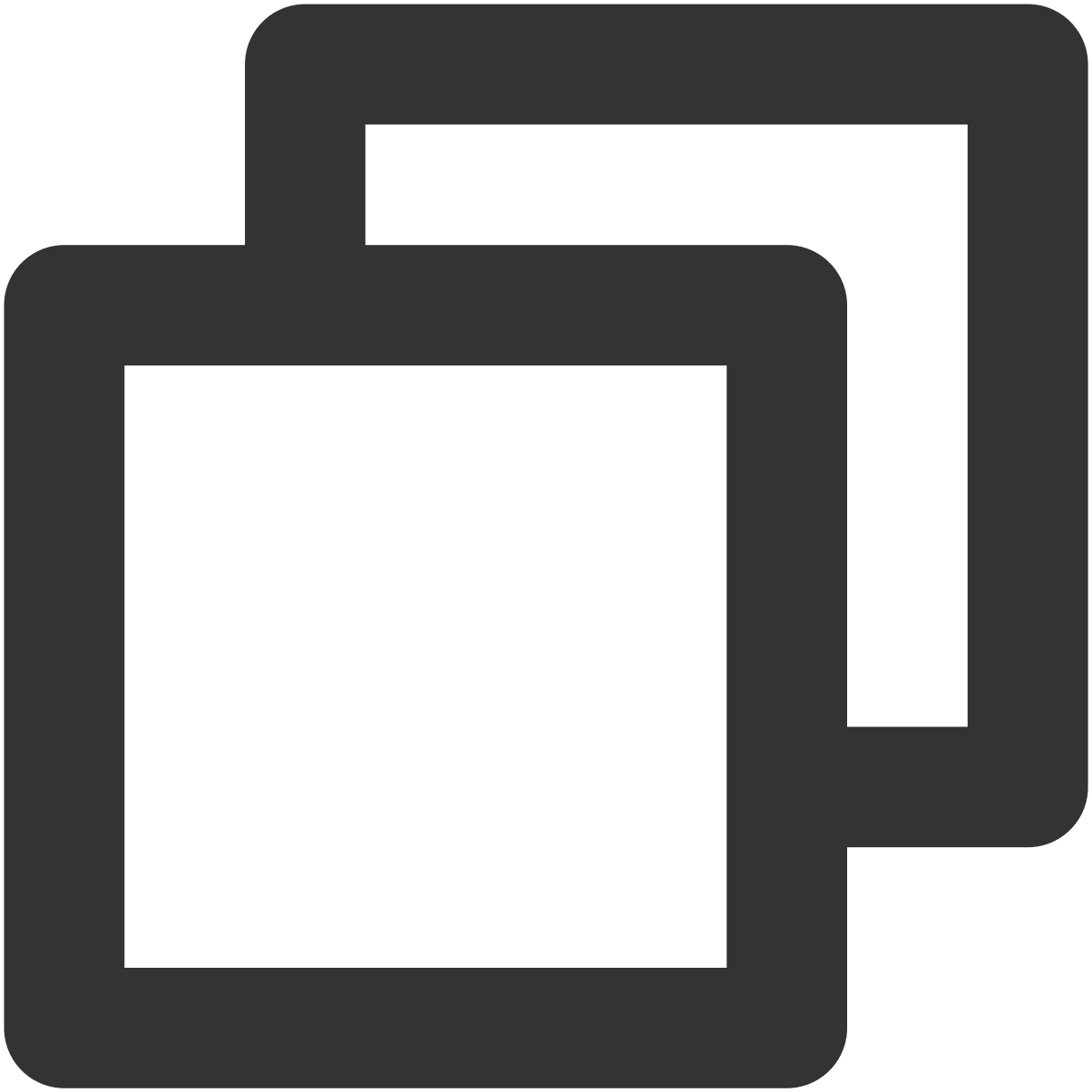
```
pod init
```

2. 在终端中，首先 `cd` 到 `Podfile` 目录下，然后执行以下命令，安装组件。



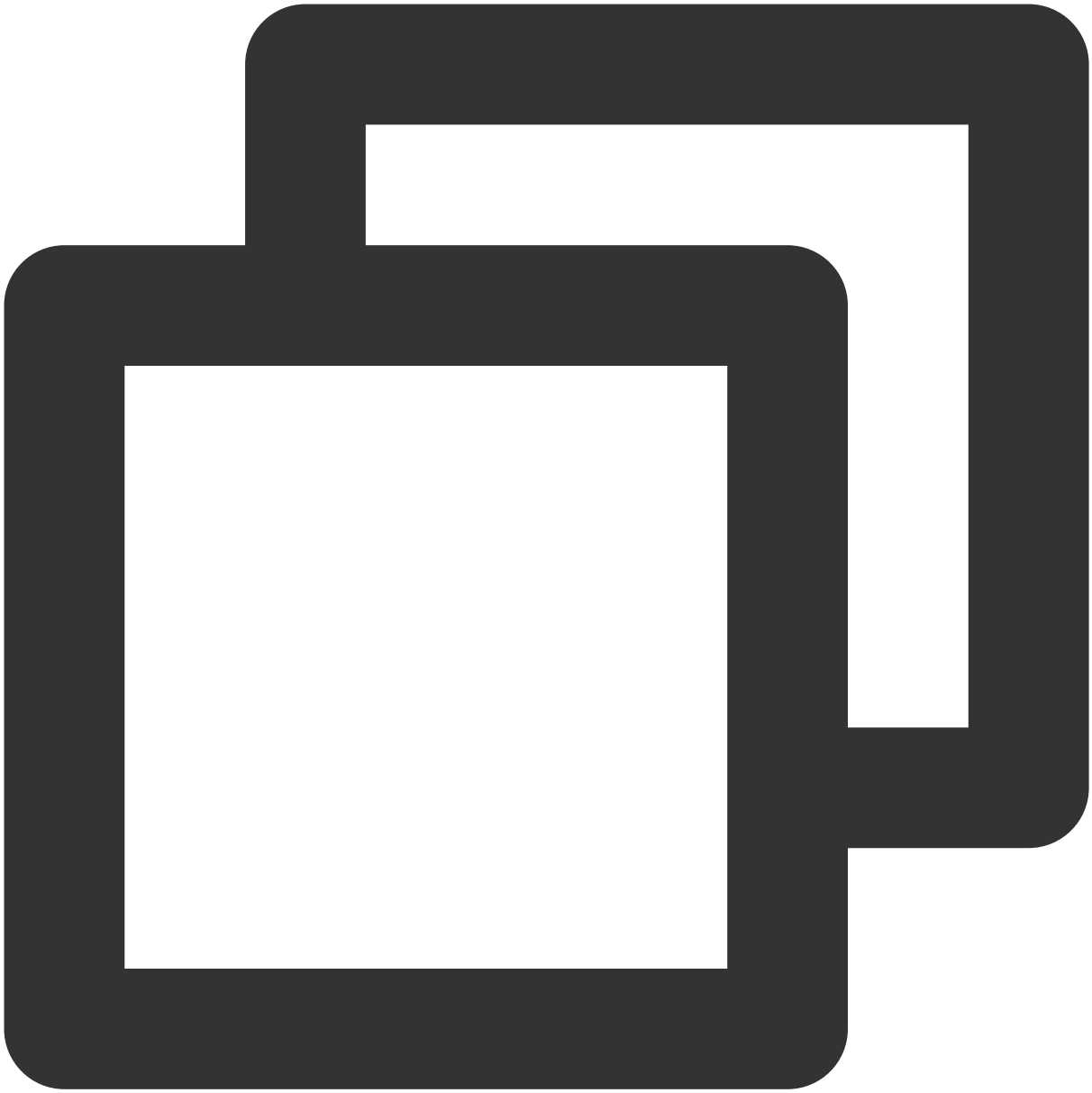
```
pod install
```

如果无法安装 TUILiveKit 最新版本，可以先删除**Podfile.lock**和**Pods**。然后执行以下命令更新本地的 CocoaPods 仓库列表。



```
pod repo update
```

之后执行以下命令，更新组件库的 Pod 版本。

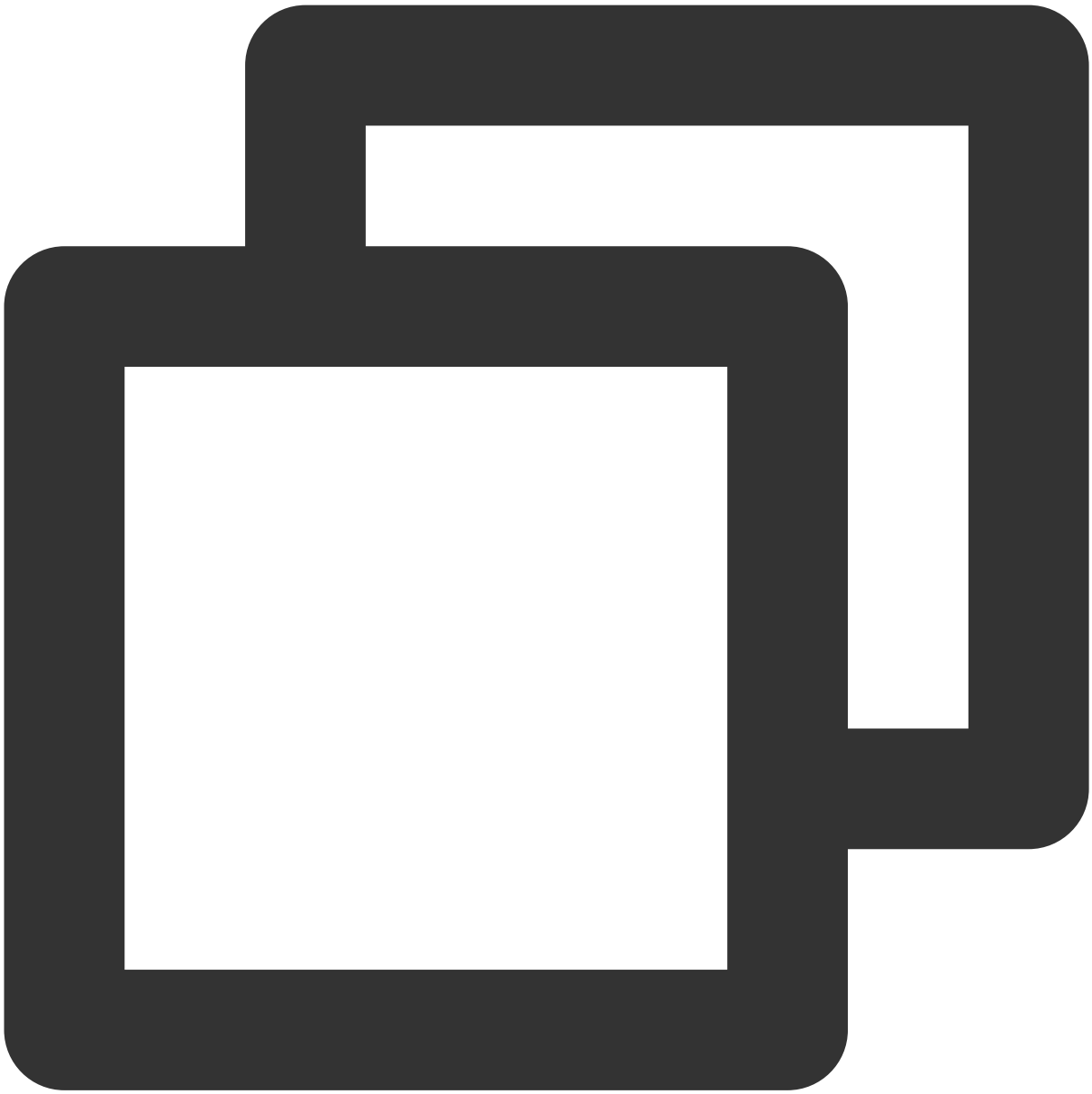


```
pod update
```

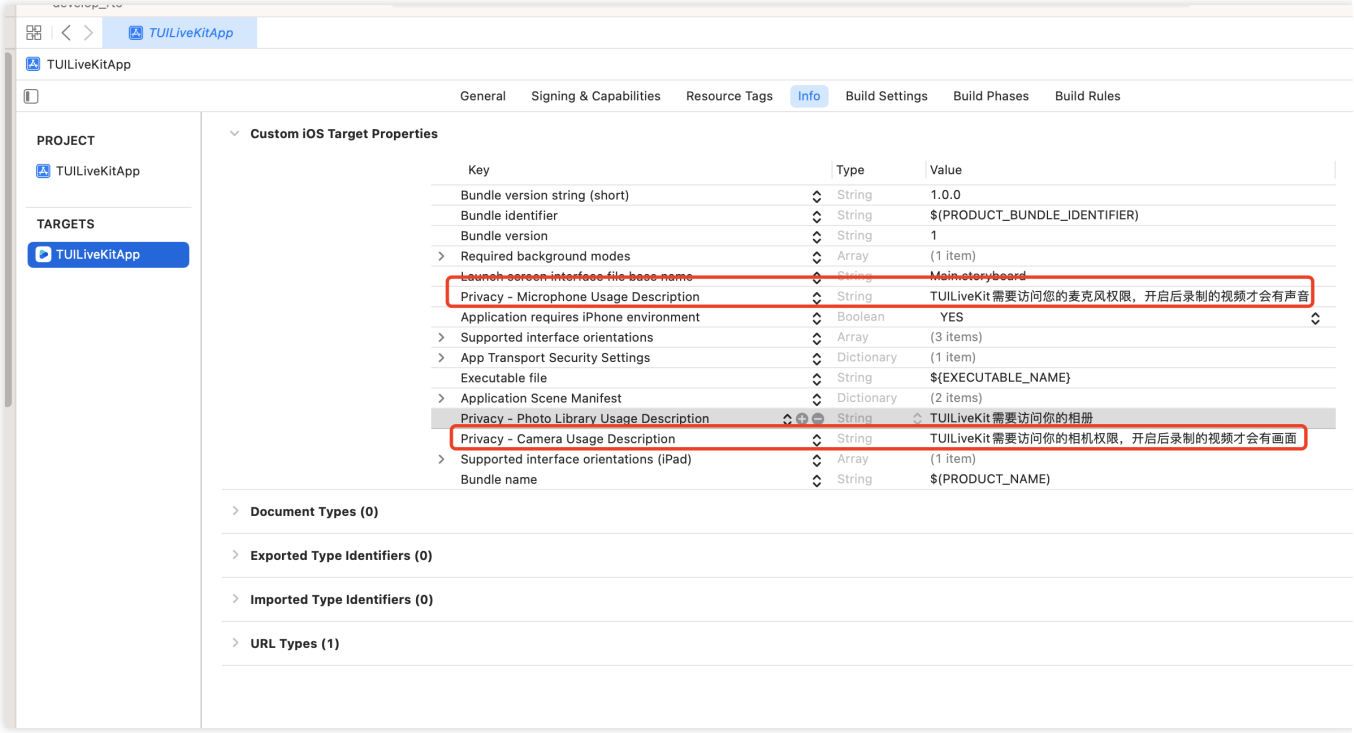
3. 可以先编译运行一下，如果遇到问题，请参见 常见问题。问题如果依然无法解决，可以先去跑一下我们的 Example工 程。您在接入和使用过程中遇到的任何问题，欢迎给我们 反馈。

步骤三：工程配置

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



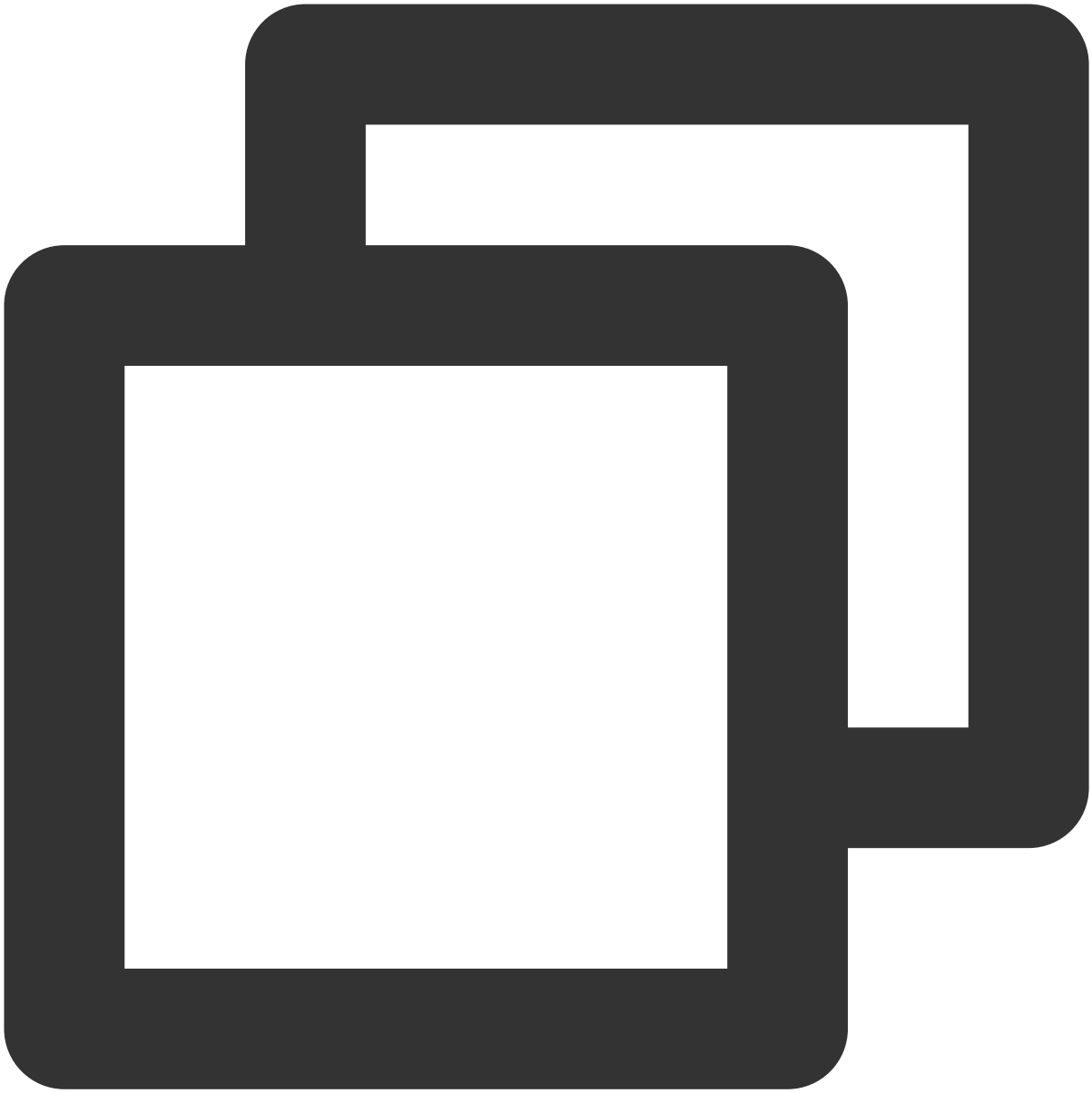
```
<key>NSCameraUsageDescription</key>
<string>TUILiveKit需要访问你的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUILiveKit需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```



步骤四：登录

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

Swift



```
//
//  AppDelegate.swift
//


import TUICore
func application(_ application: UIApplication, didFinishLaunchingWithOptions launch

    TUILogin.login(1400000001,                  // 请替换为步骤一取到的 SDKAppID
            userID: "denny",                    // 请替换为您的 UserID
            userSig: "xxxxxxxxxxx") {           // 您可以在控制台中计算一个 UserSig 并填在这
```

```
        print("login success")
    } fail: { (code, message) in
        print("login failed, code: \\(code), error: \\(message ?? "nil")")
    }

    return true
}
```

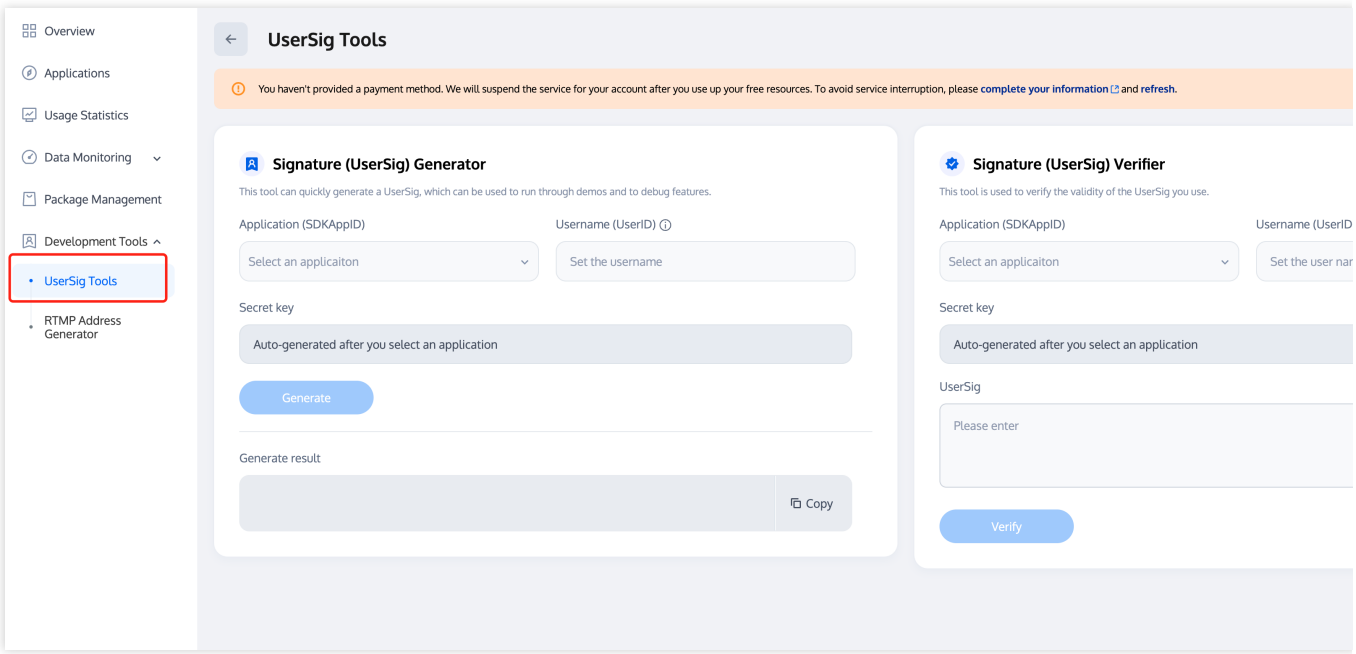
参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

SDKAppID：在 步骤一 中的最后一步中您已经获取到，这里不再赘述。

UserID：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

UserSig：使用 开通服务（TUILiveKit）获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台左侧项目栏中的 UserSig 工具，创建一个临时可用的 UserSig。



更多信息请参见 UserSig 相关。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：

SDKAppID 设置错误。

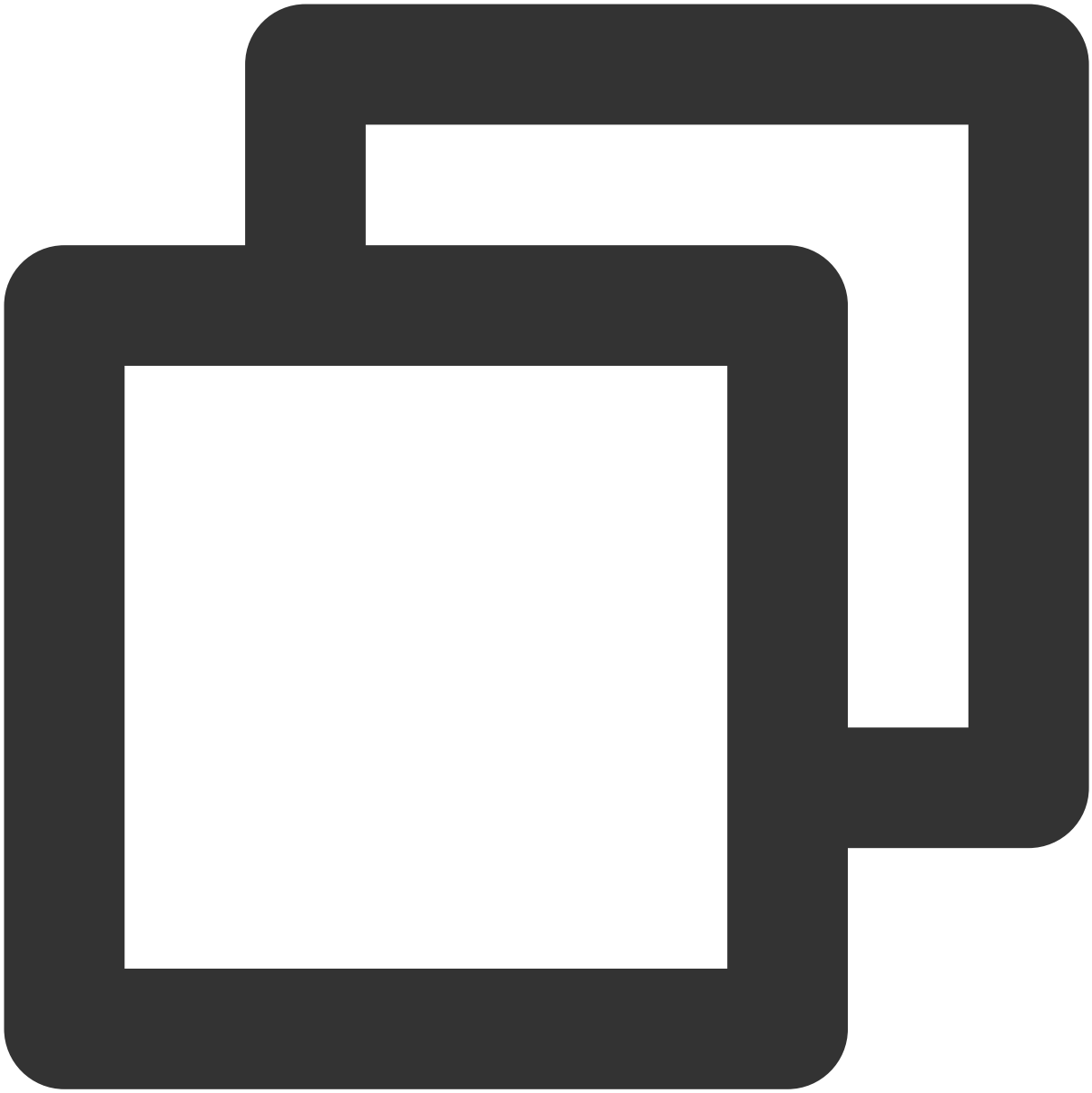
userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 userSig。

userSig 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的

时候设置一些辨识度高的 userID。
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TULiveKit 组件时向您的服务器请求实时计算出的 userSig。

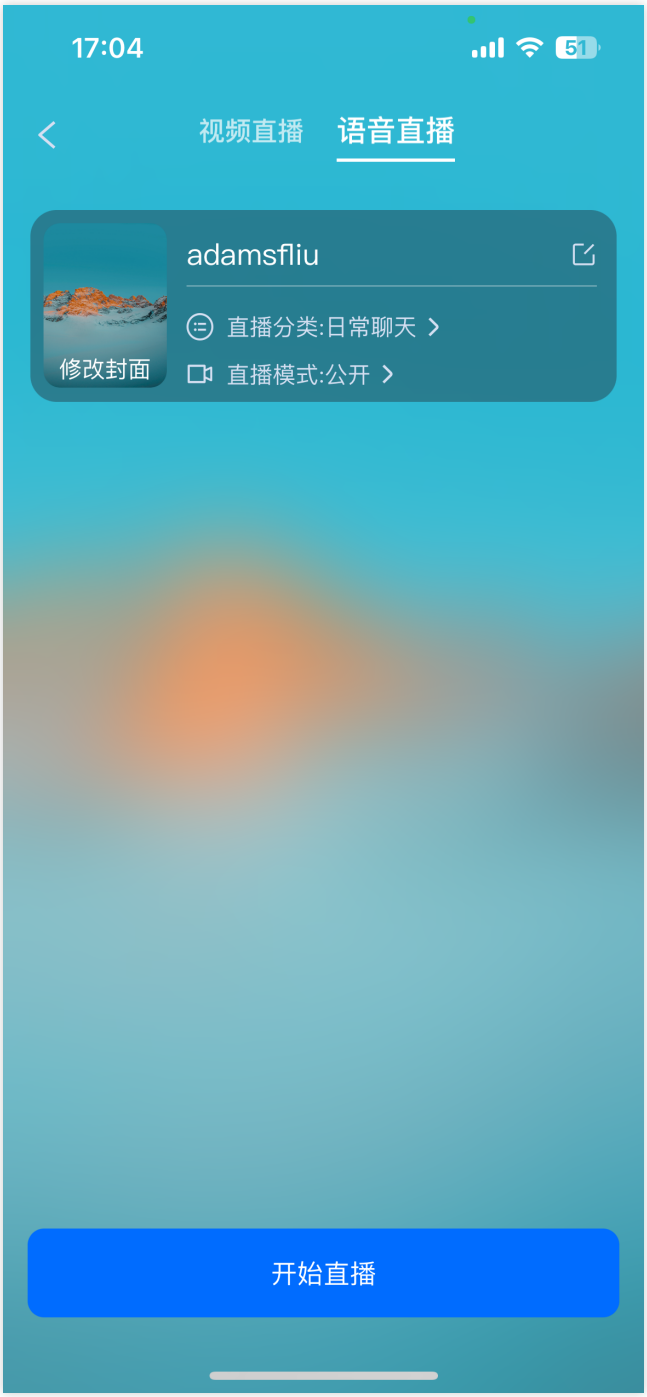
步骤五：进入语音聊天室预览画面

通过加载 TUILiveKit 的TUIVoiceRoomViewController页面，就可以拉起预览画面，点击“开始直播”即可创建语音聊天室。
Swift

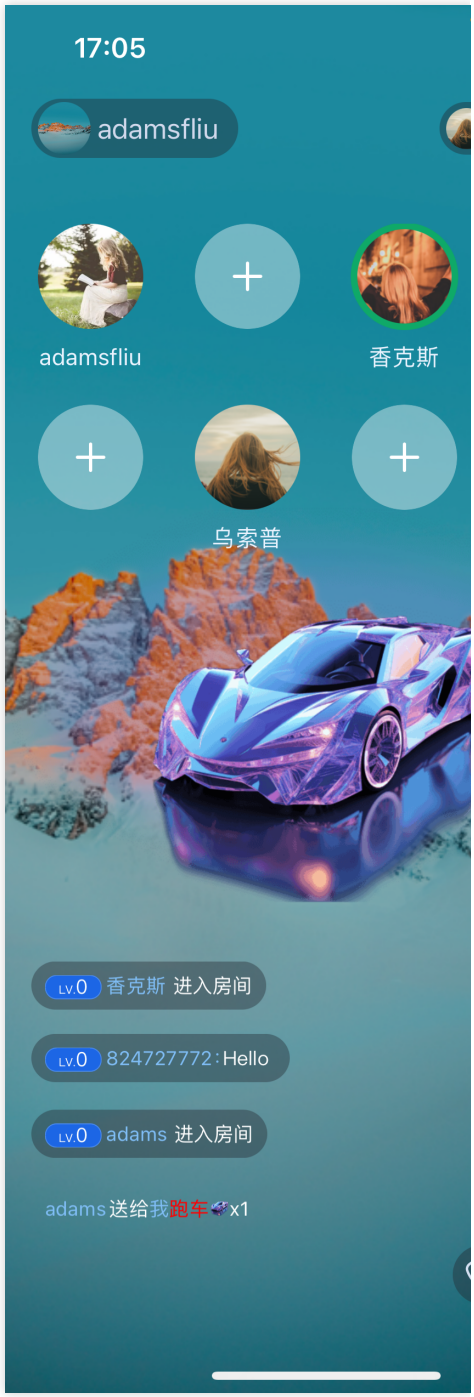


```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    //RoomId可以自定义生成
    let roomId = "123666";
```

```
    // 进入语音聊天室预览画面
    var roomParams = RoomParams()
    // 房间最大麦位数，默认套餐包支持的最大麦位数量
    roomParams.maxSeatCount = 0
    roomParams.seatMode = .applyToTake

    let voiceRoomController = TUIVoiceRoomViewController(roomId: roomId, behavior:
    self.navigationController?.pushViewController(voiceRoomController, animated: tr
}
```



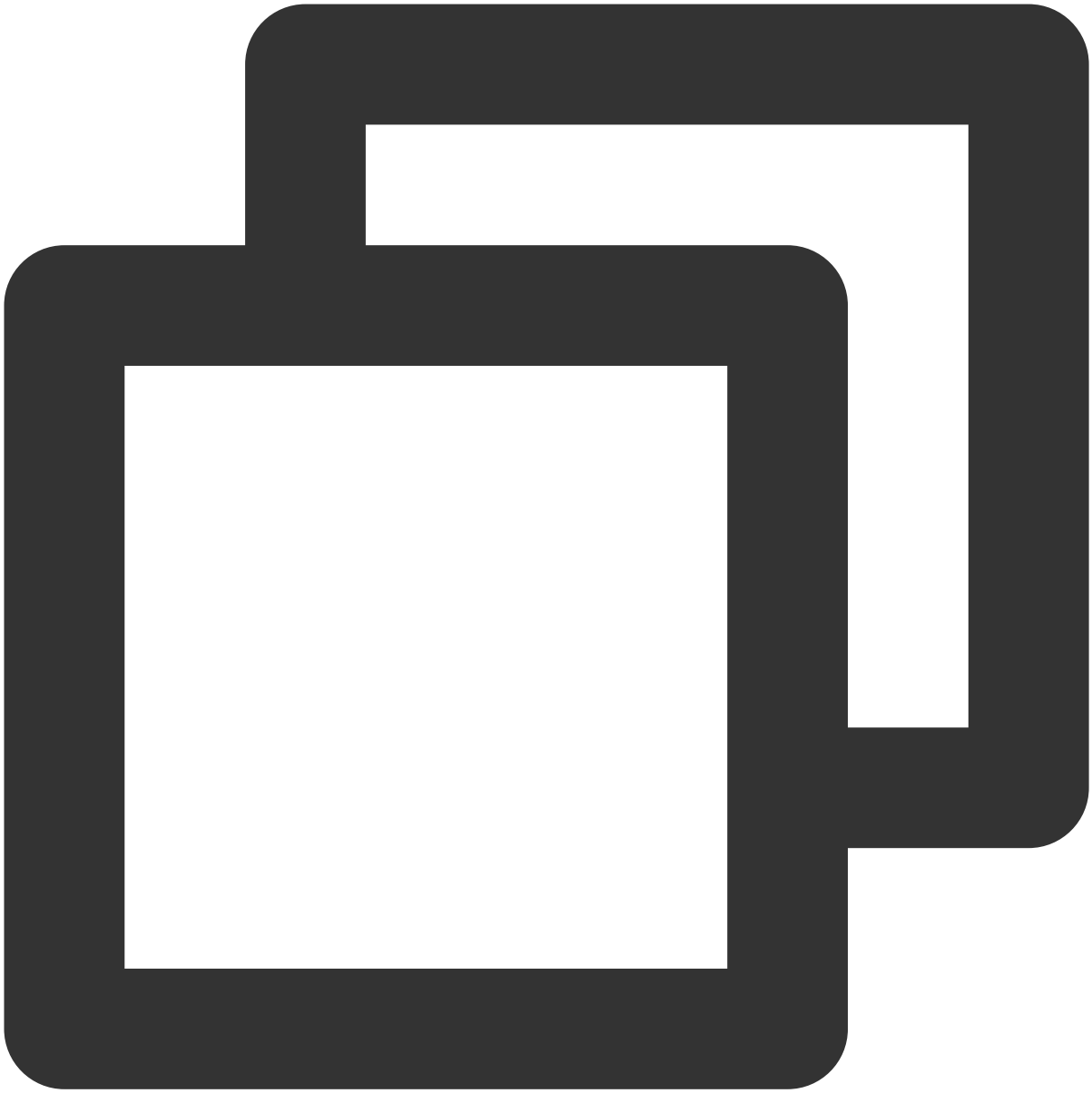
语音聊天室预览画面



语音聊天室房间内画面

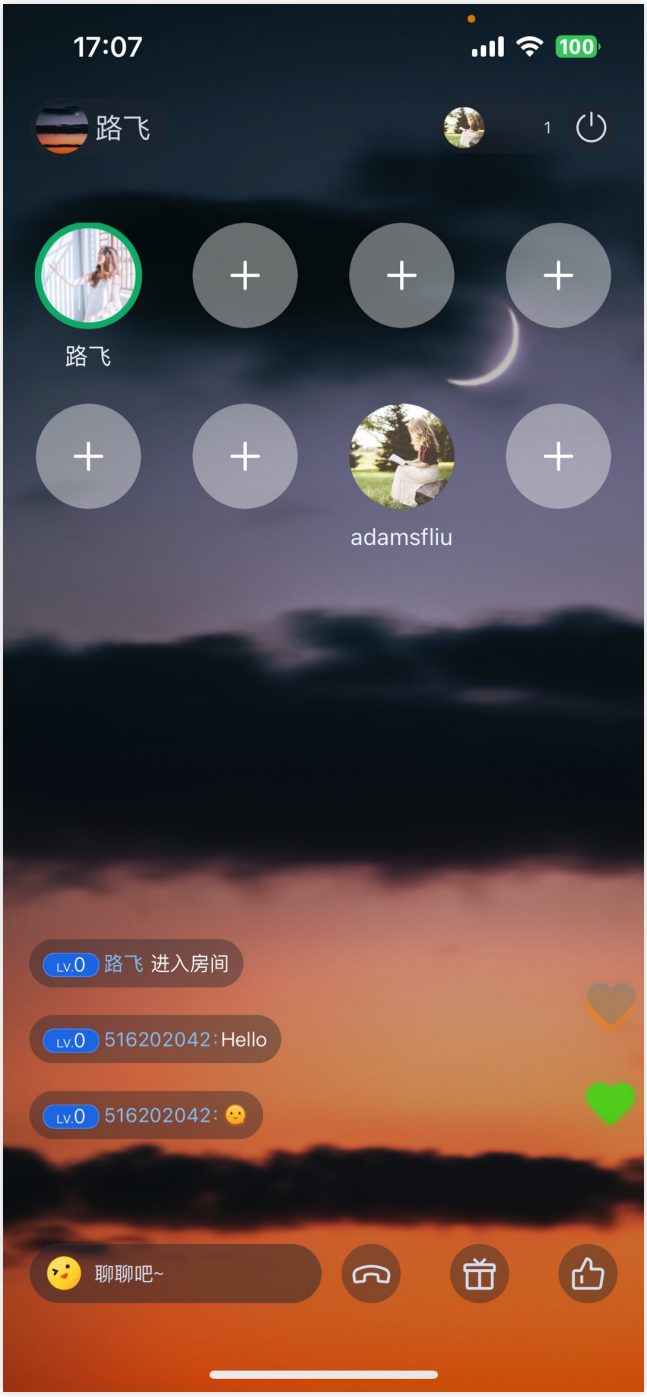
步骤六：观众进入直播间

Swift

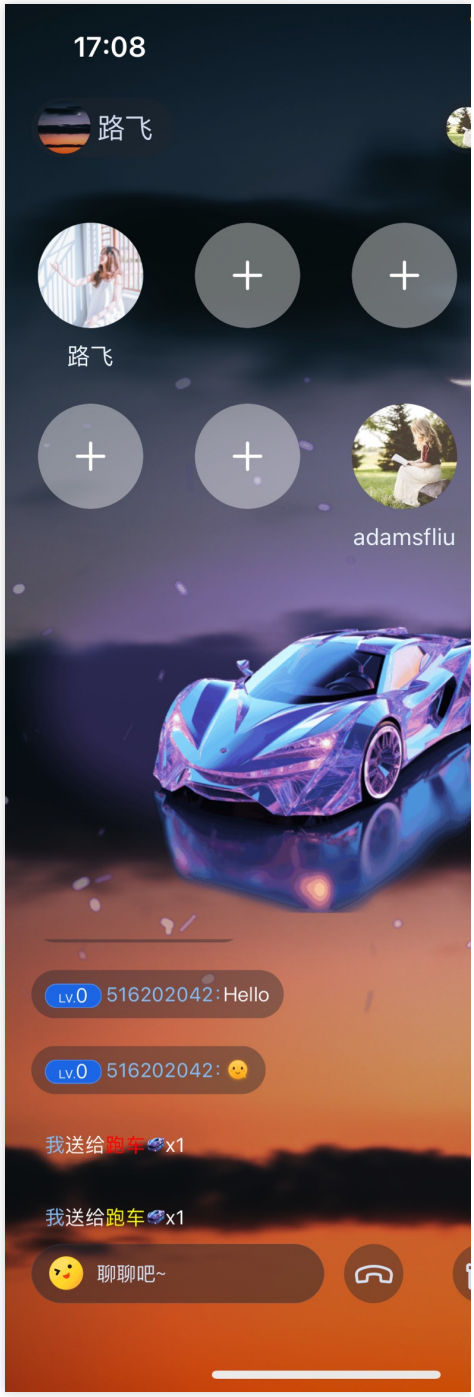


```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    //RoomId是创建语音聊天室的房间id
    let roomId = "123666";
```

```
    // 进入语音聊天室
    let viewController = TUIVoiceRoomViewController(roomId: roomId, behavior: .join
    self.navigationController?.pushViewController(viewController, animated: true)
}
```



语音聊天室



语音聊天室

更多特性

弹幕
礼物

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2024-05-17 11:20:40

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您可以在10分钟内完成接入工作，并最终得到一个包含完备 UI 界面的直播功能。

环境准备

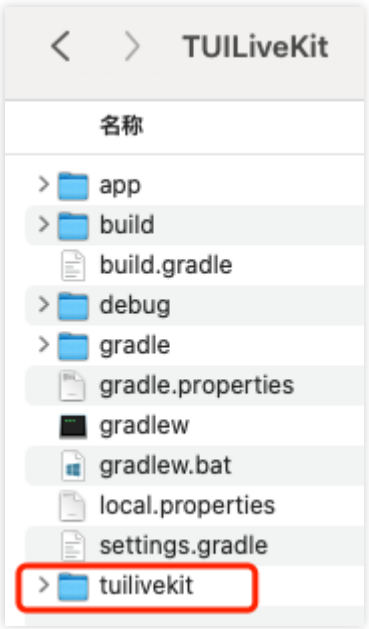
Android 5.0（SDK API Level 21）及以上版本。
Gradle 4.2.1 及以上的版本。
Android 5.0 及以上的手机设备。

步骤一：开通服务

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

步骤二：下载 TUILiveKit 组件

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



步骤三：工程配置

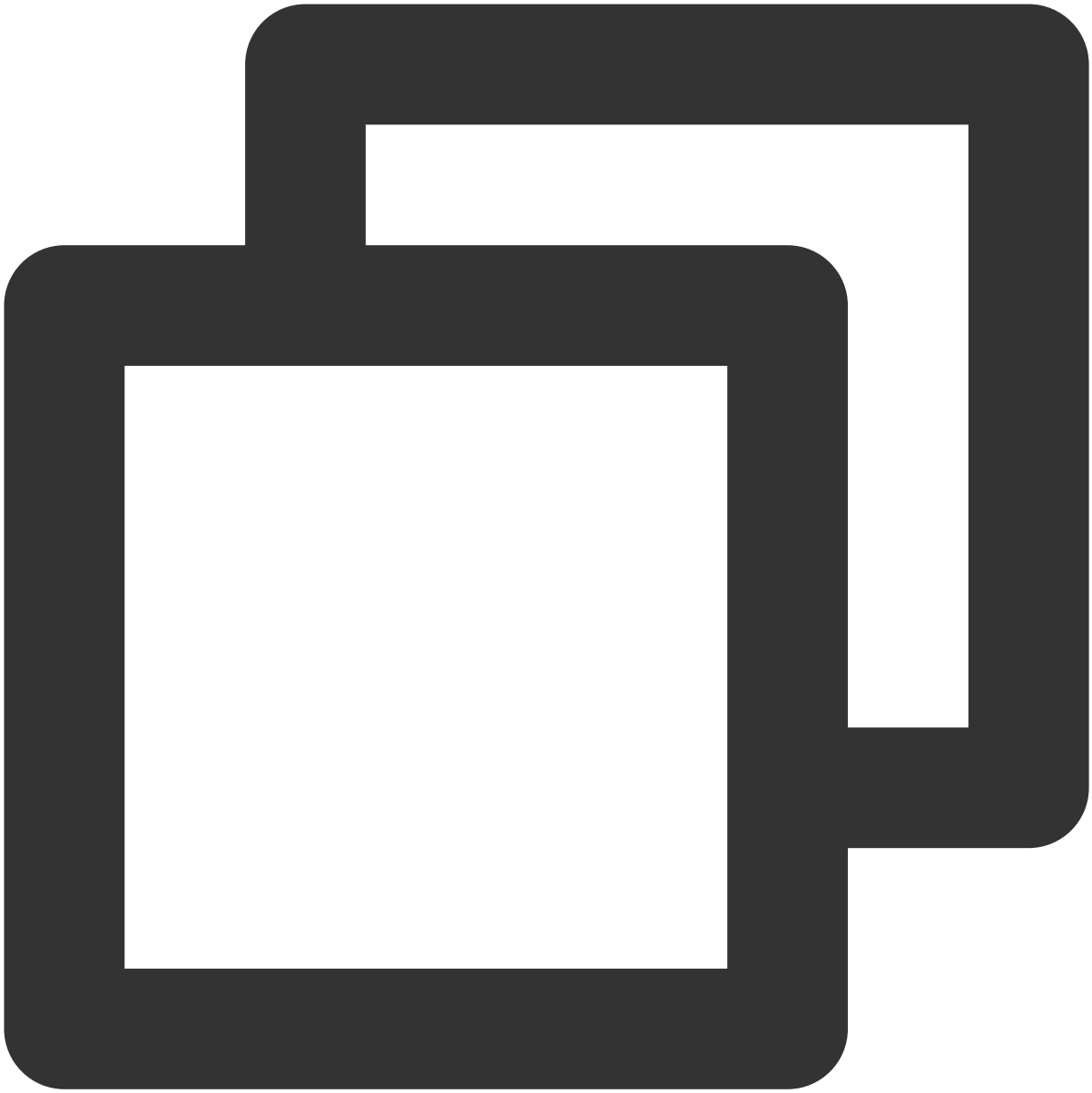
Java

1. 在工程中添加 `jitpack` 仓库依赖（下载播放礼物 svg 动画的三方库 `SVGAPlayer` ）：

gradle7.0以下版本

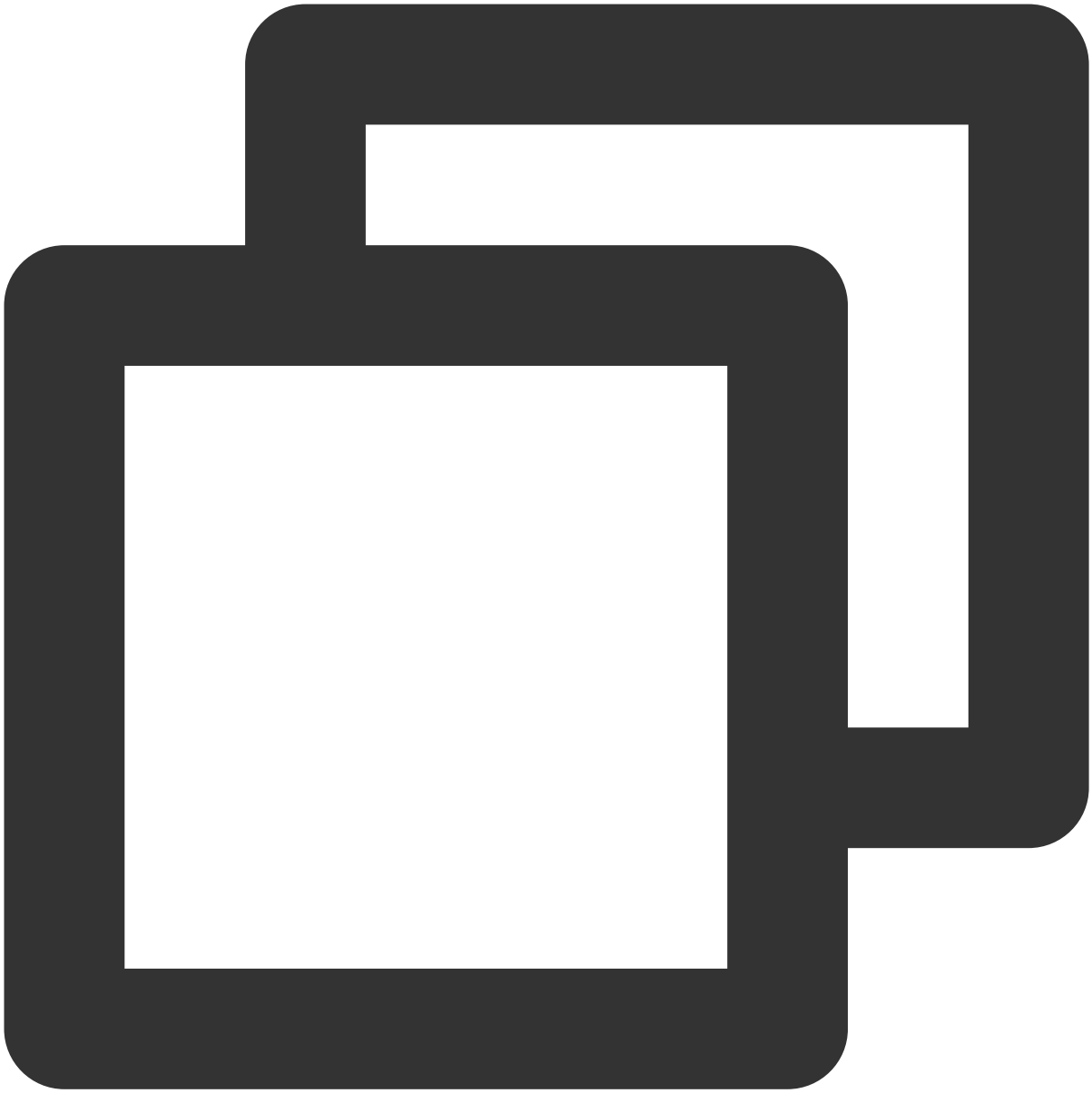
gradle7.0及以上版本

在工程根目录下的 build.gradle 文件中添加 jitpack 仓库地址：



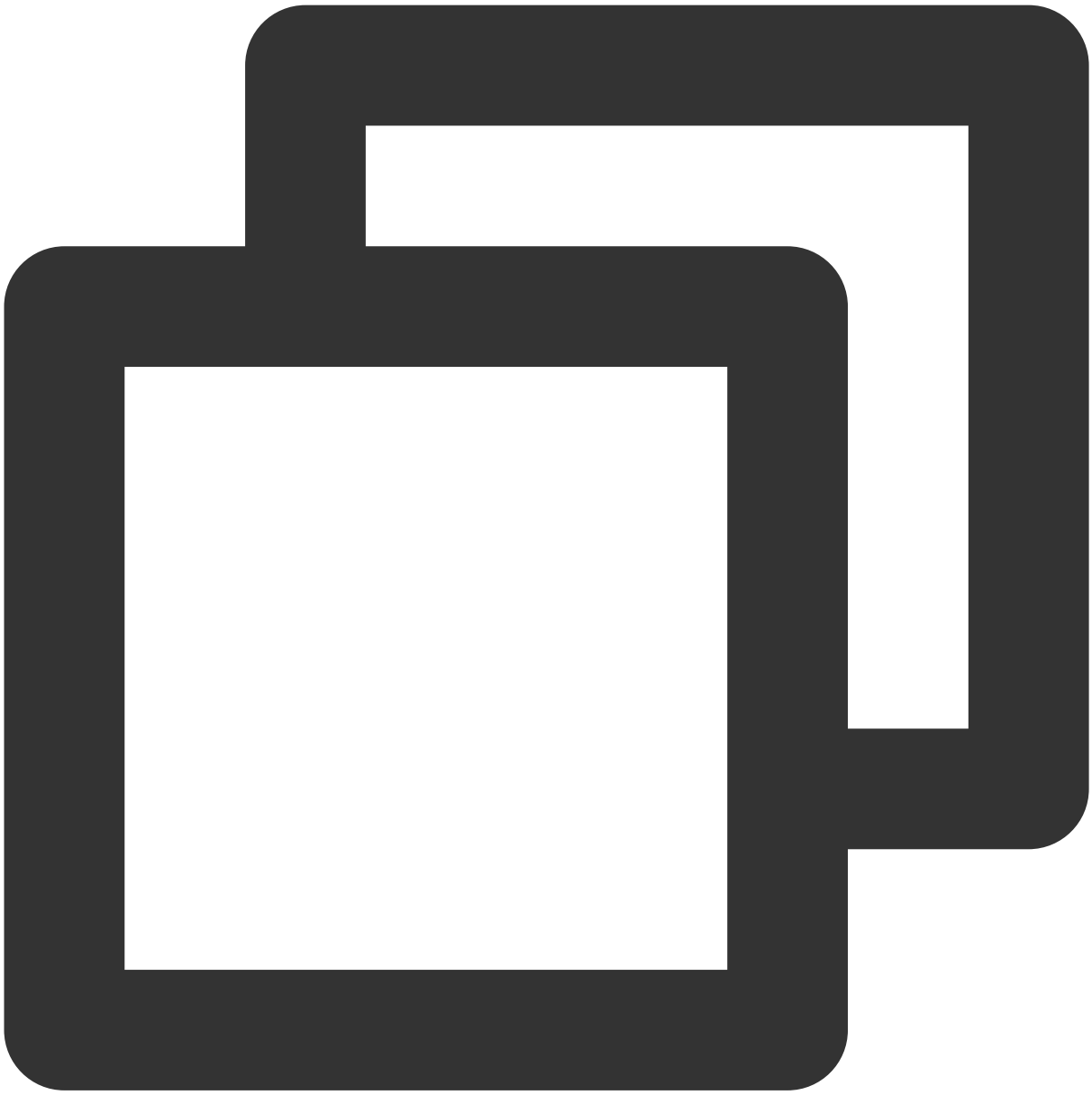
```
allprojects {
    repositories {
        google()
        mavenCentral()
        // 添加 jitpack 仓库地址
        maven { url 'https://jitpack.io' }
    }
}
```

在工程根目录下的 settings.gradle 文件中添加 jitpack 仓库地址：



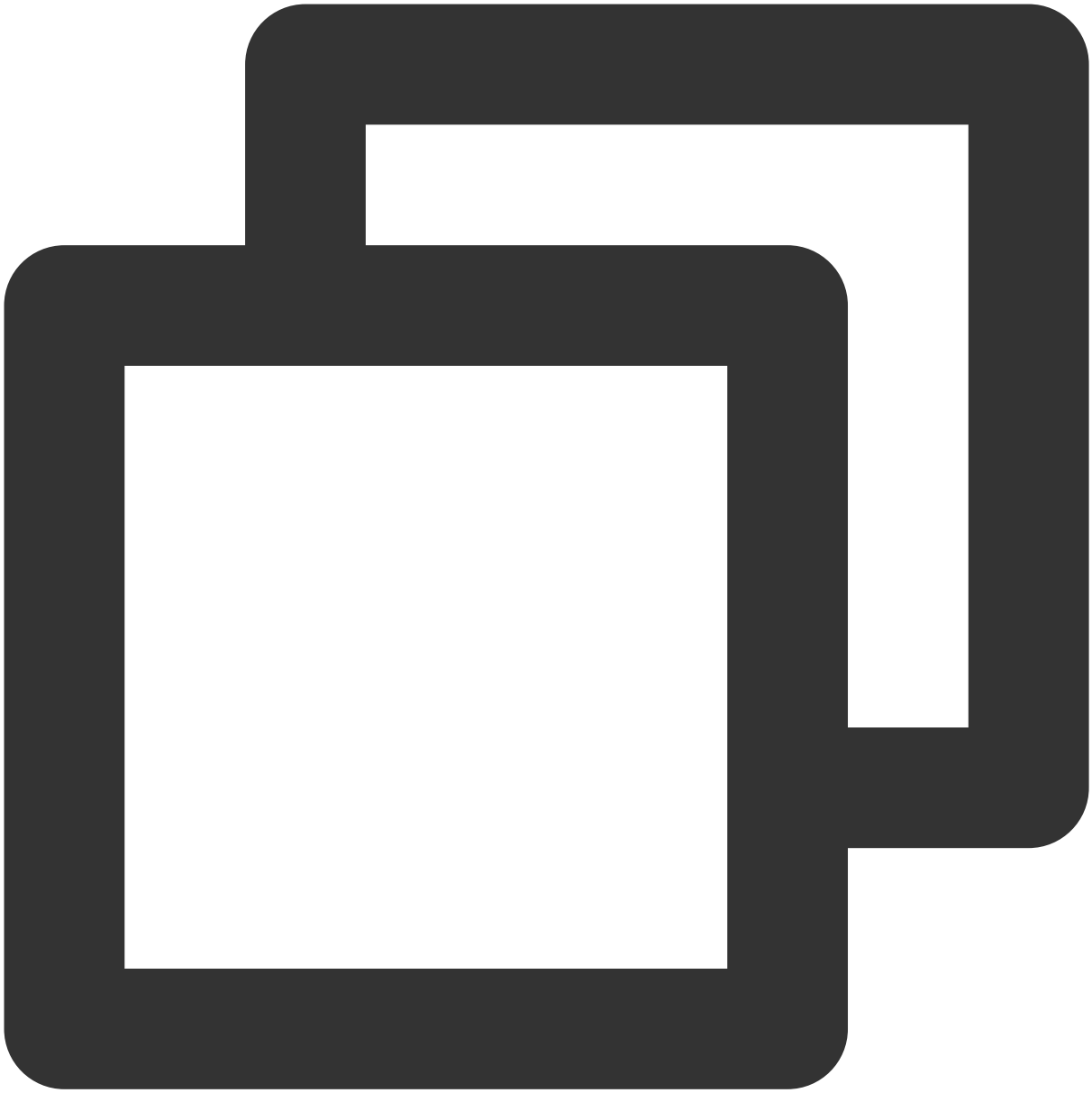
```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        // 添加 jitpack 仓库地址
        maven { url 'https://jitpack.io' }
    }
}
```

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



```
include ':tuilivekit'
```

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

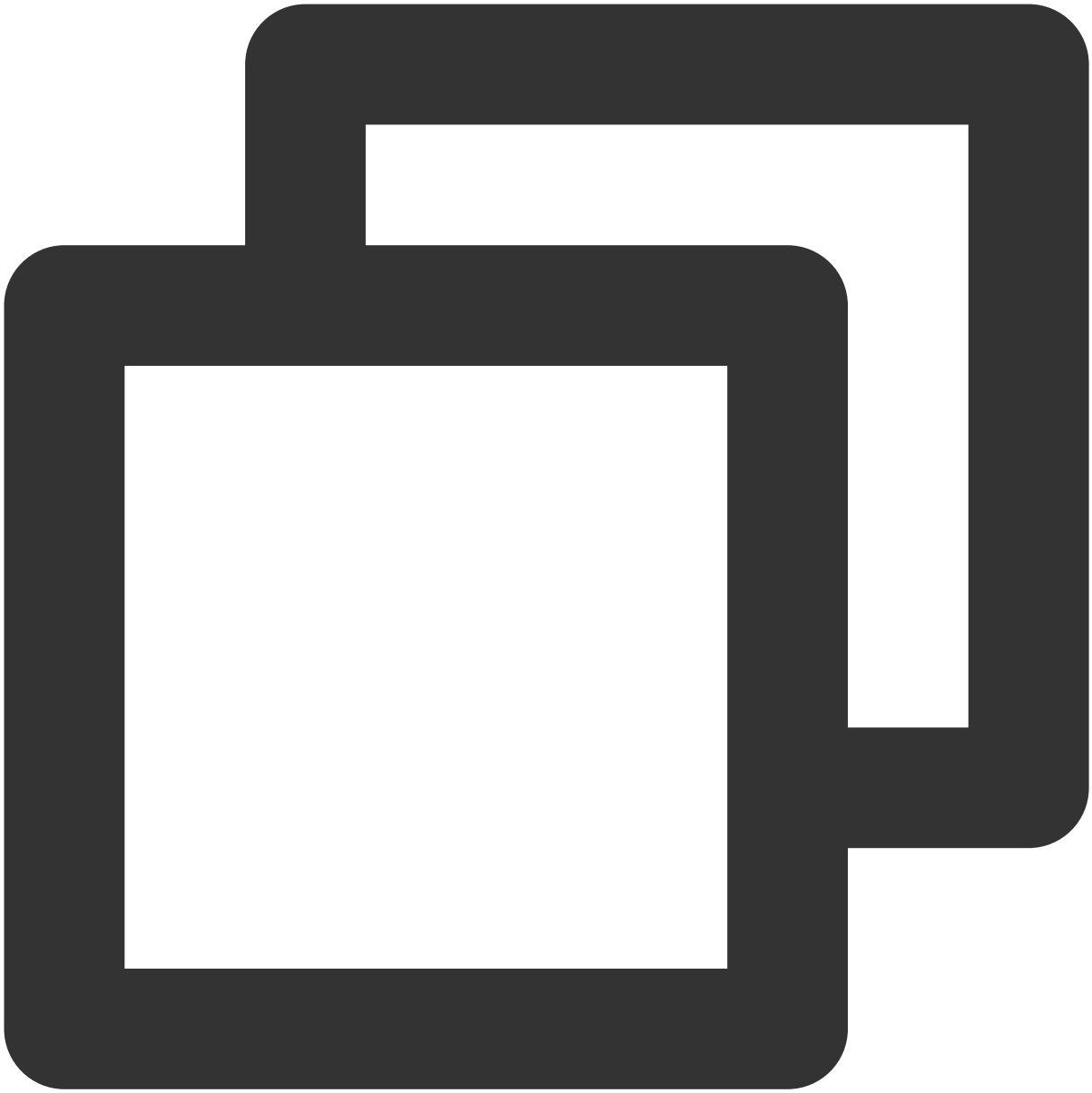


```
api project(':tuilivekit')
```

说明：

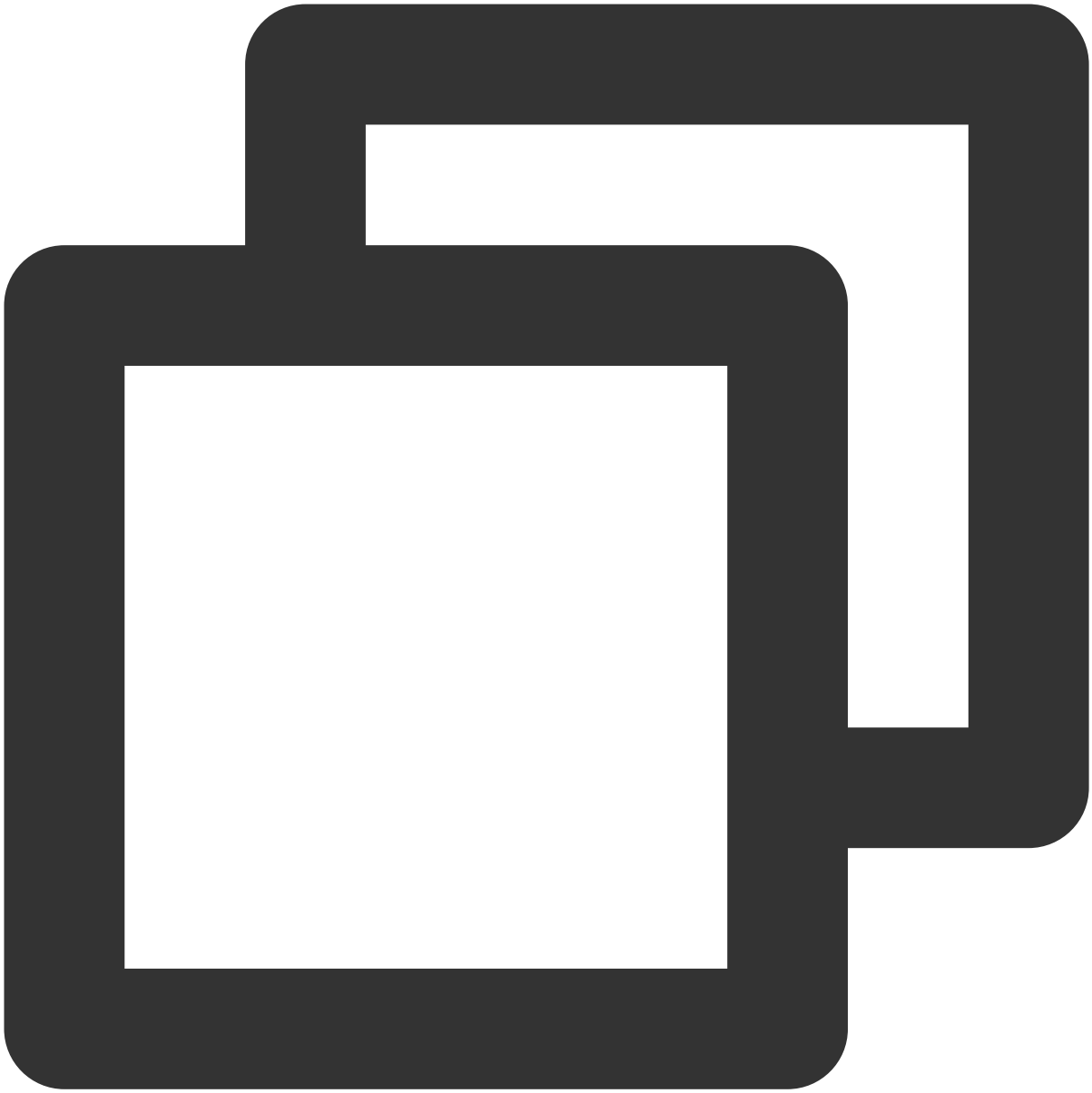
TUILiveKit 工程内部已经默认依赖： `TRTC SDK` 、 `IM SDK` 、 `tuiroomengine` 以及公共库 `tuicore` ，不需要开发者单独配置。如需进行版本升级，则修改 `tuilivekit/build.gradle` 文件即可。

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



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

5. 在 `AndroidManifest.xml` 里，给 `application` 的 `android:theme` 属性配置一个 `Theme.AppCompat` 主题：



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
        ...
    </application>
</manifest>
```

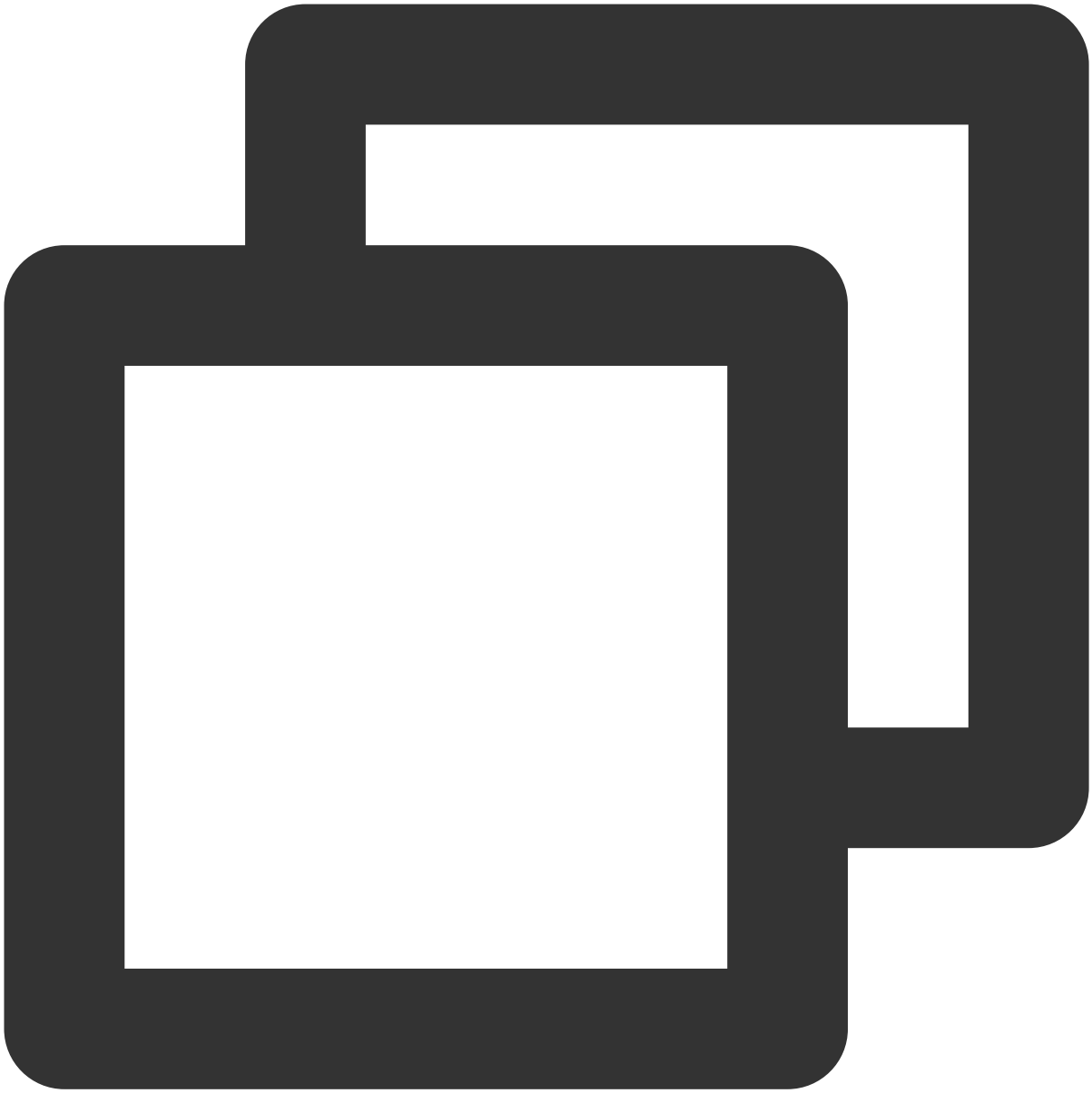
注意：

TUILiveKit 会在内部帮助您动态申请相机、麦克风、读取存储权限等，如果因为您的业务问题需要删减，可以请修改 `tuilivekit/src/main/AndroidManifest.xml` 。
如果遇到 `AndroidManifest.xml` 提示 `allowBackup` 相关异常，请参见 allowBackup 异常。
如果遇到 `Theme.AppCompat` 相关问题，请参见 Activity 主题问题。

步骤四：登录

在调用 TUILiveKit 组件各项功能之前，先要执行 TUI 组件的登录。在您的项目中，建议在您的业务登录场景里或者 App 的首个启动 Activity 里，添加执行如下登录代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤关键，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Java



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

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

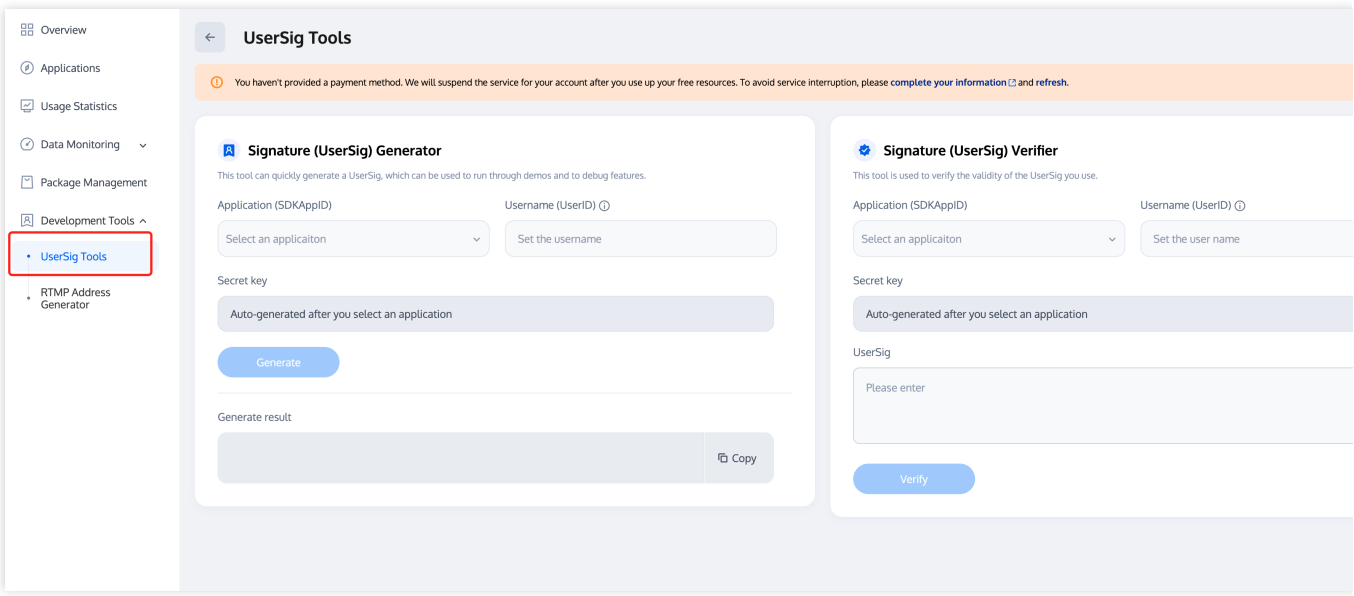
参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

SDKAppID：在 开通服务（TUILiveKit） 中您已经获取到，这里不再赘述。

UserID：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

UserSig：使用 开通服务（TUILiveKit） 获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台左侧项目栏中的 UserSig 工具，创建一个临时可用的 UserSig。



更多信息请参见 UserSig 相关。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误。

userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 userSig。

userSig 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userID。

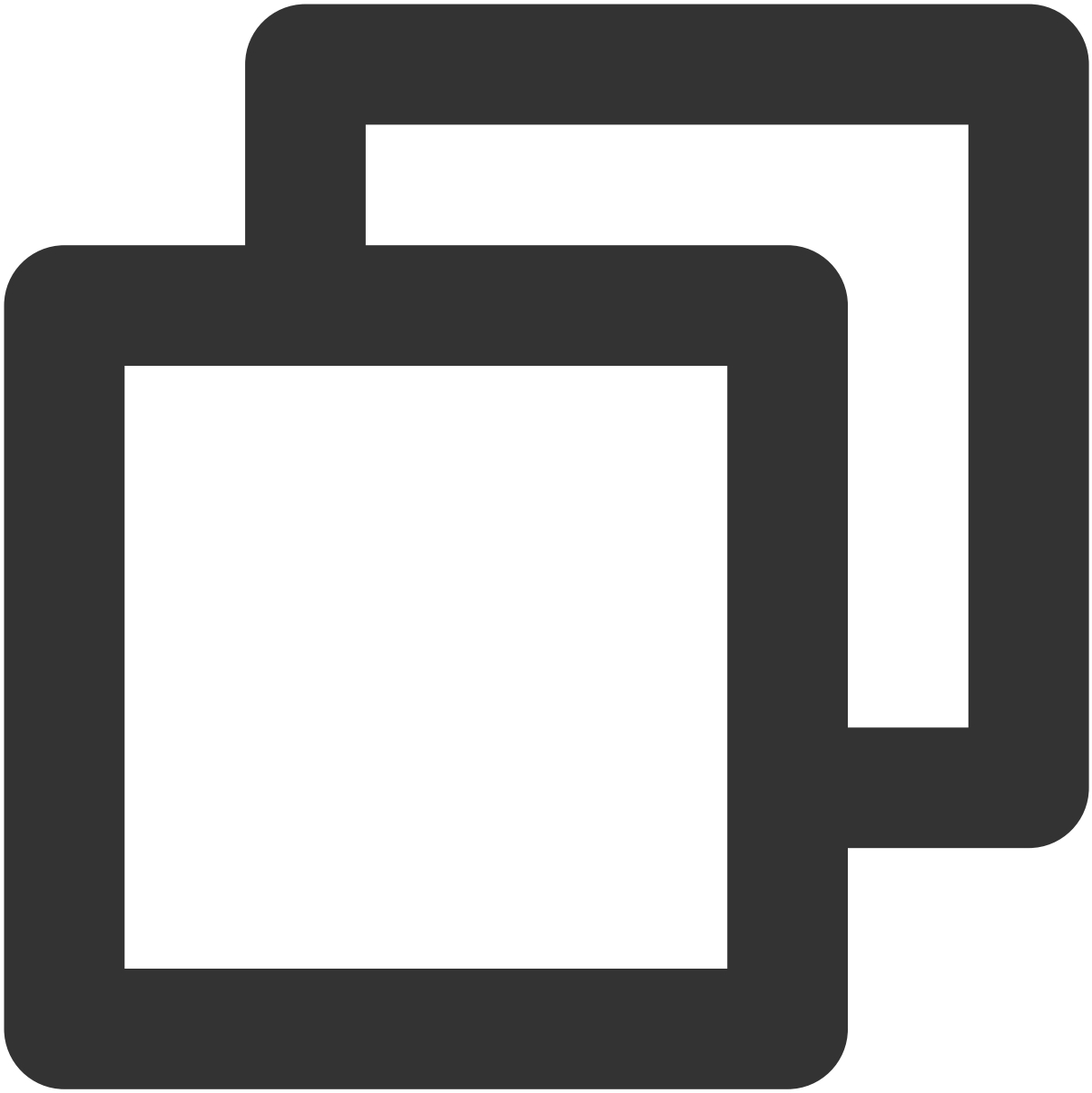
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUILiveKit 组件时向您的服务器请求实时计算出的 userSig。

步骤五：进入直播预览画面

注意：

请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。

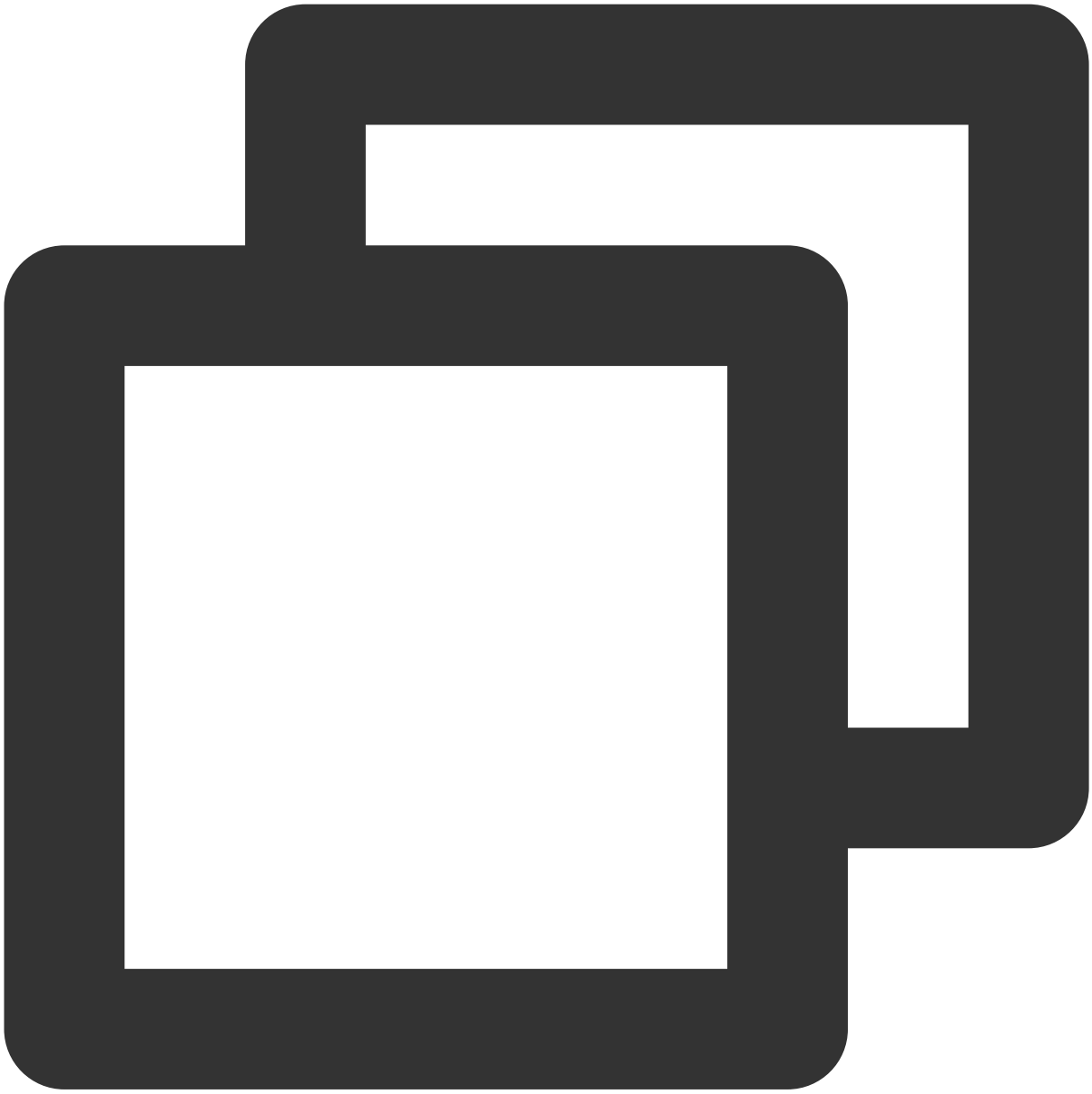
1. 新建 app_activity_anchor.xml 文件（默认路径：app/src/main/res/layout/app_activity_anchor.xml）。



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

2. 新建 AnchorActivity.java，并在 `AndroidManifest.xml` 里注册，通过加载 TUILiveKit 的 TUIVoiceRoomFragment 页面，就可以拉起预览画面。

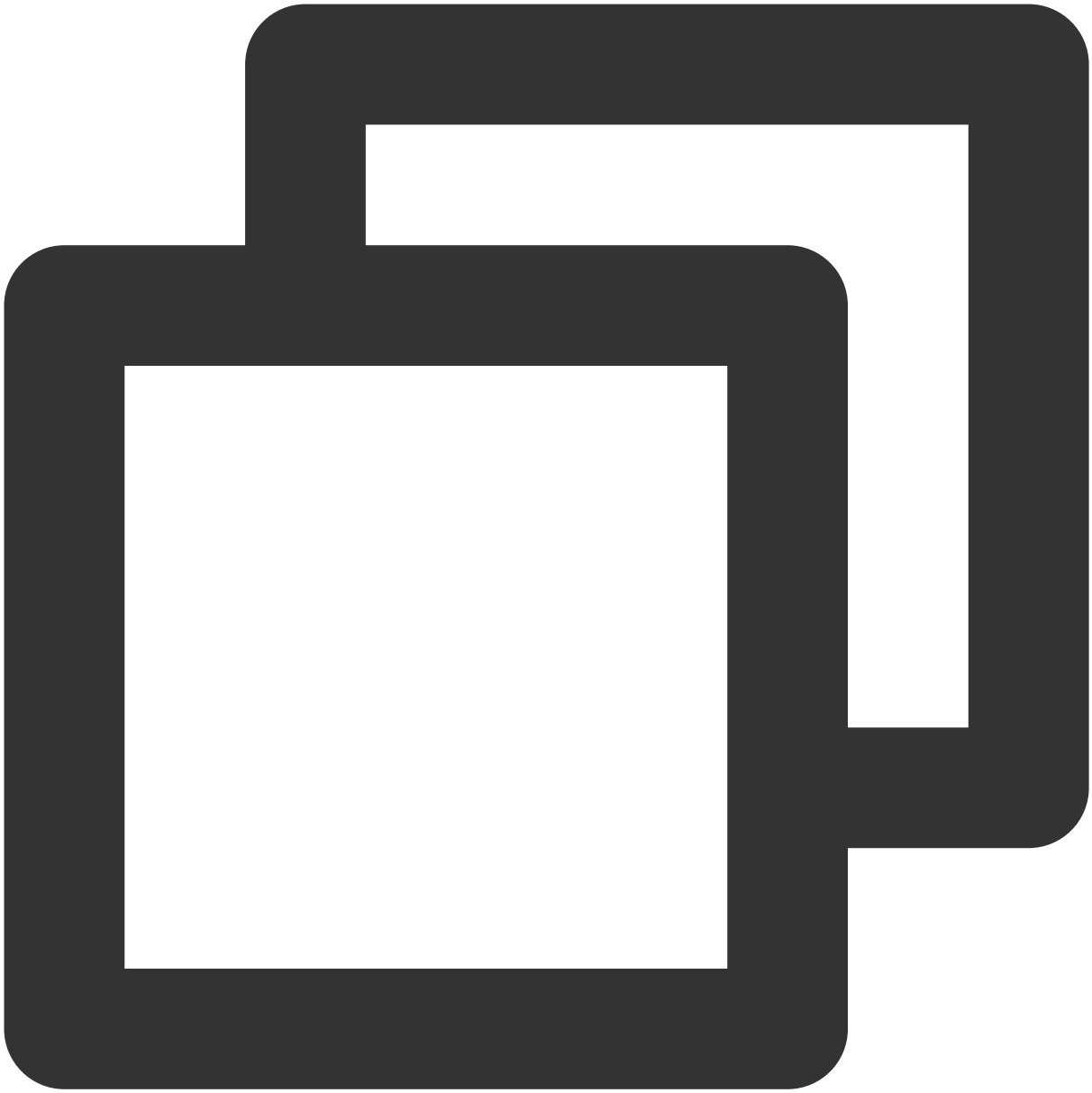
Java



```
public class AnchorActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_anchor);
        //主播的房间 ID
        String roomId = "123666";
```

```
        //创建房间的参数
        LiveDefine.RoomParams params = new LiveDefine.RoomParams();
        //房间最大麦位数，默认套餐包支持的最大麦位数量
        params.maxSeatCount = 0;
        //上麦模式
        params.seatMode = TUIRoomDefine.SeatMode.APPLY_TO_TAKE;

        //在 Activity 中显示 主播预览页面，点击预览页面的开始直播按钮，即可发起在线语音直播
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction(
        TUIVoiceRoomFragment fragment = new TUIVoiceRoomFragment(roomId,
        LiveDefine.RoomBehavior.PREPARE_CREATE, params);
        fragmentTransaction.add(R.id.fl_container, fragment);
        fragmentTransaction.commit();
    }
}
```

在 app 项目的 `AndroidManifest.xml` 里注册 `AnchorActivity` （请使用您的 `AnchorActivity` 实际包名）：



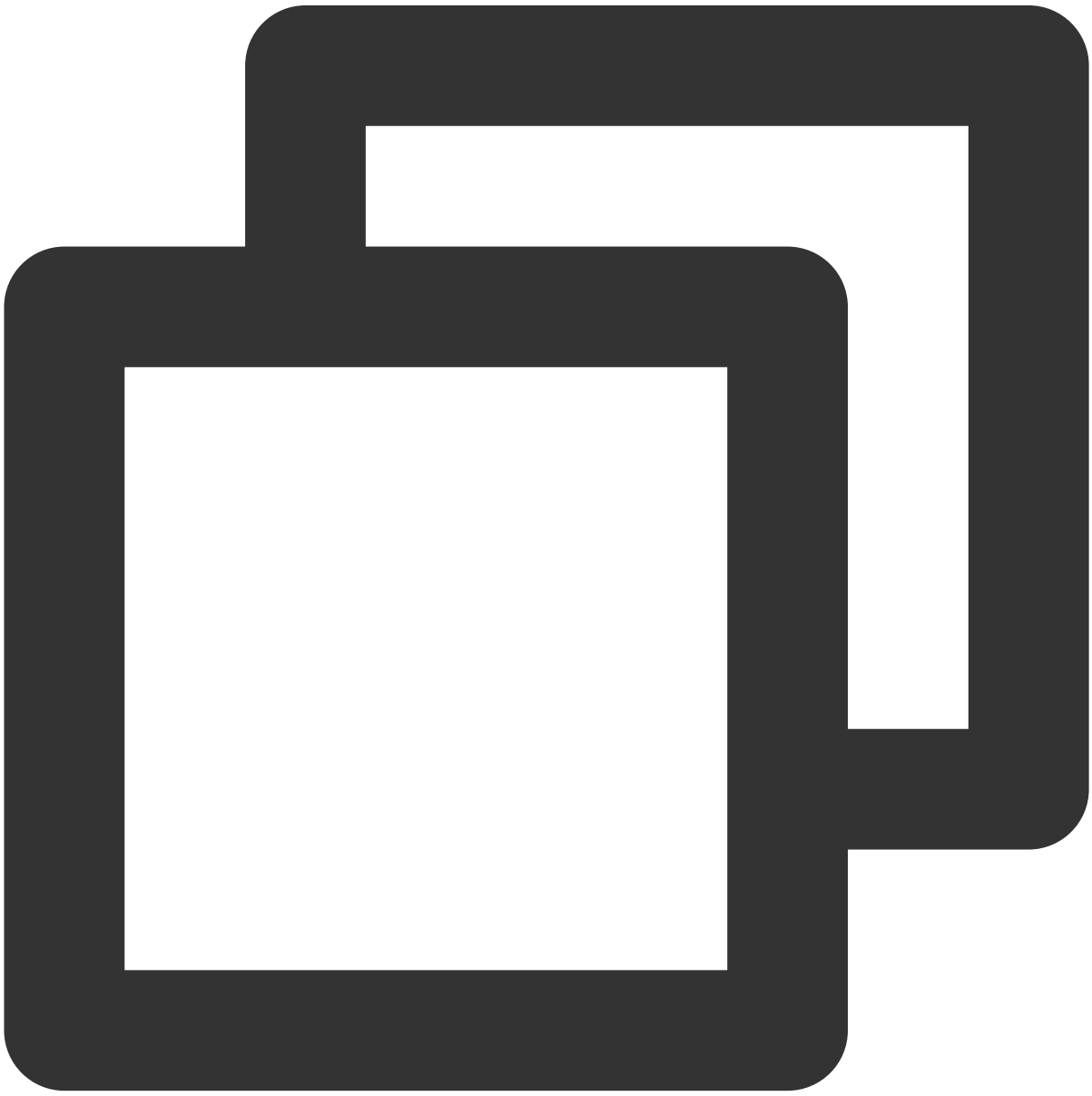
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application>
        ...
        <!-- 示例：注册AnchorActivity，请使用您的实际包名 -->
        <activity android:name="com.trtc.uikit.livekit.example.main.AnchorActivity"
                  android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"/>
        ...
    </application>
</manifest>
```

注意：

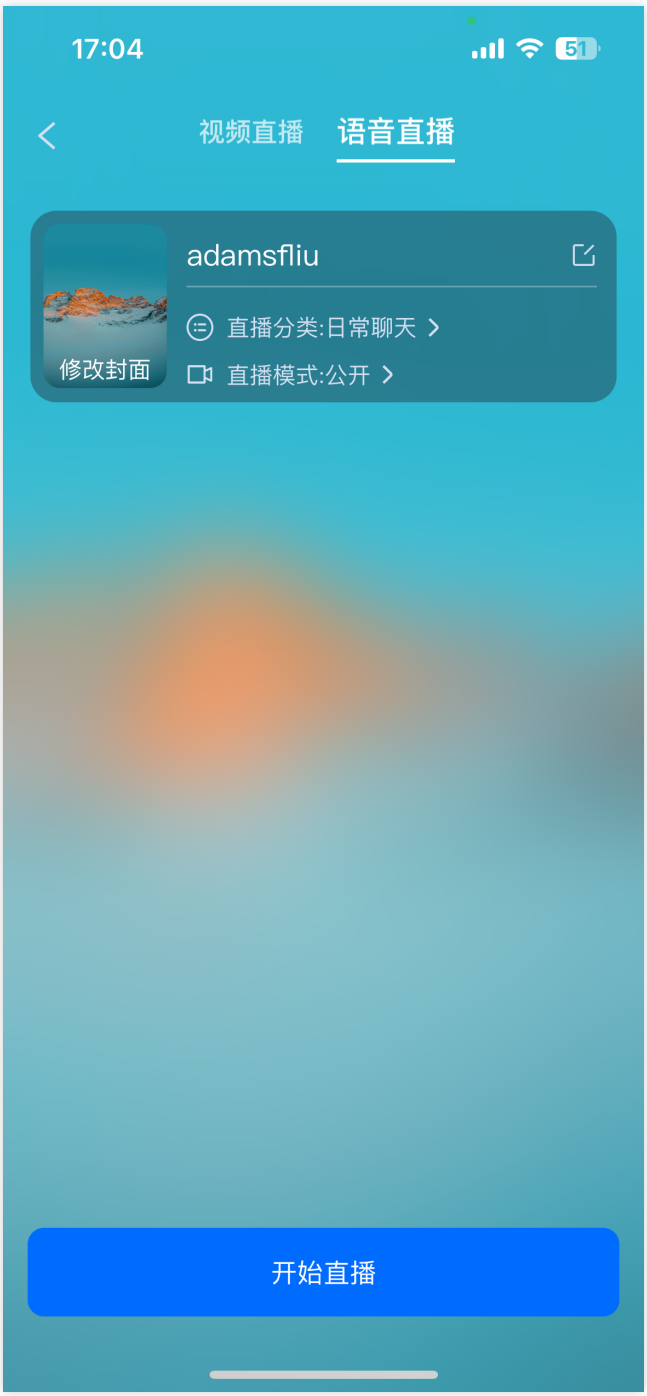
由于 `AnchorActivity` 继承自 `AppCompatActivity` ，所以要给 `AnchorActivity` 设置一个 `Theme.AppCompat` 主题。您可以修改成自己的 `Theme.AppCompat` 主题，如果遇到 Theme.AppCompat 相关问题，请参见 Activity 主题问题。

在您需要开启直播的地方（具体由您的业务决定，默认情况下可在 MainActivity 某点击事件里执行），执行如下操作，拉起主播开播页面：3、在您需要开启直播的地方（具体由您的业务决定，默认情况下可在 MainActivity 某点击事件里执行），执行如下操作，拉起主播开播页面：

Java



```
Intent intent = new Intent(context, AnchorActivity.class);
startActivity(intent);
```



语音聊天室预览画面

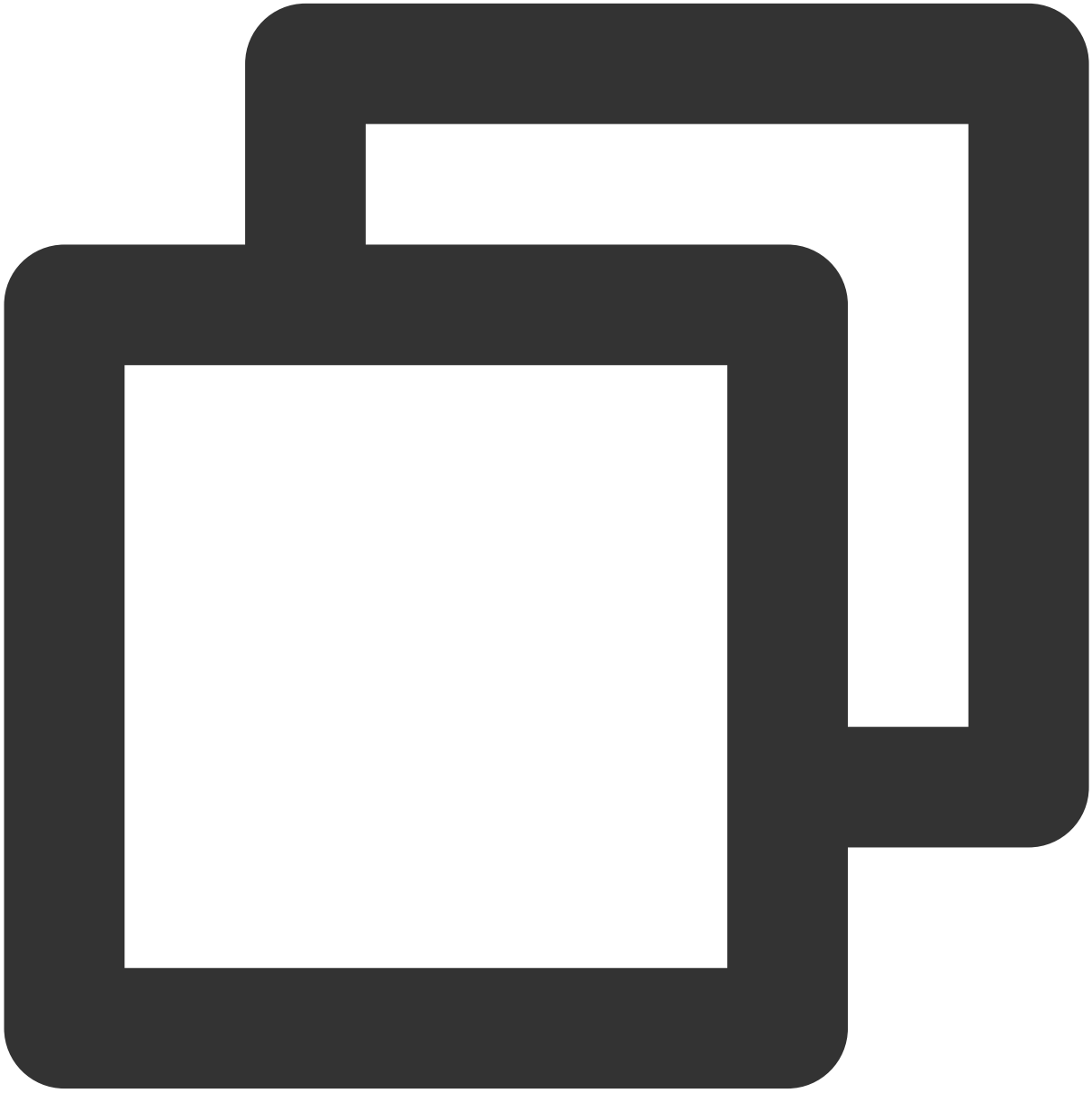
语音聊天室房间内画面

步骤六：观众进入直播间

注意：

请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后观众才能正常进入直播间。并且，登录的观众 `UserID` 不要与主播 `UserID` 相同。

1. 新建 app_activity_audience.xml 文件（默认路径：app/src/main/res/layout/app_activity_audience.xml）。

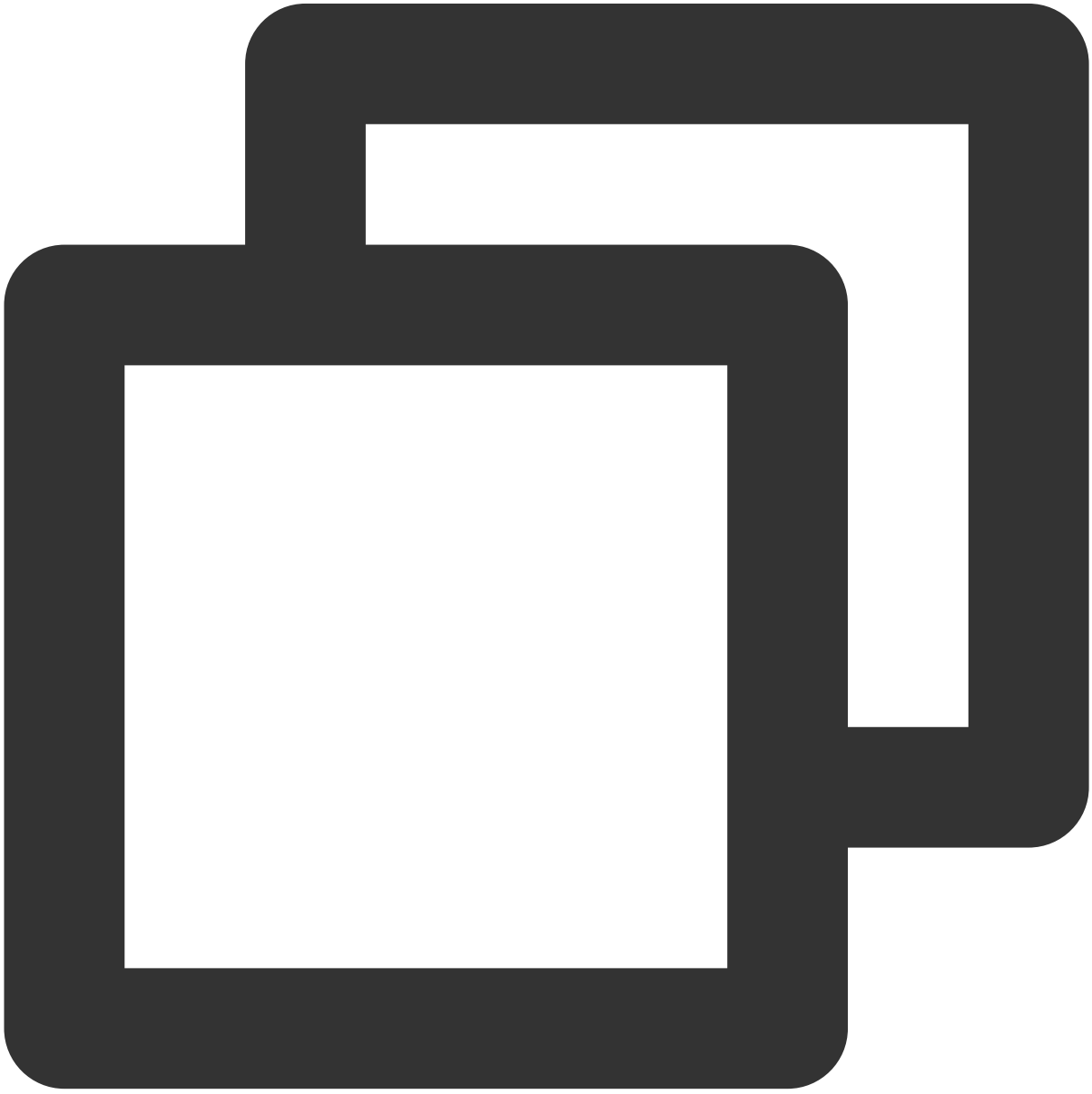


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

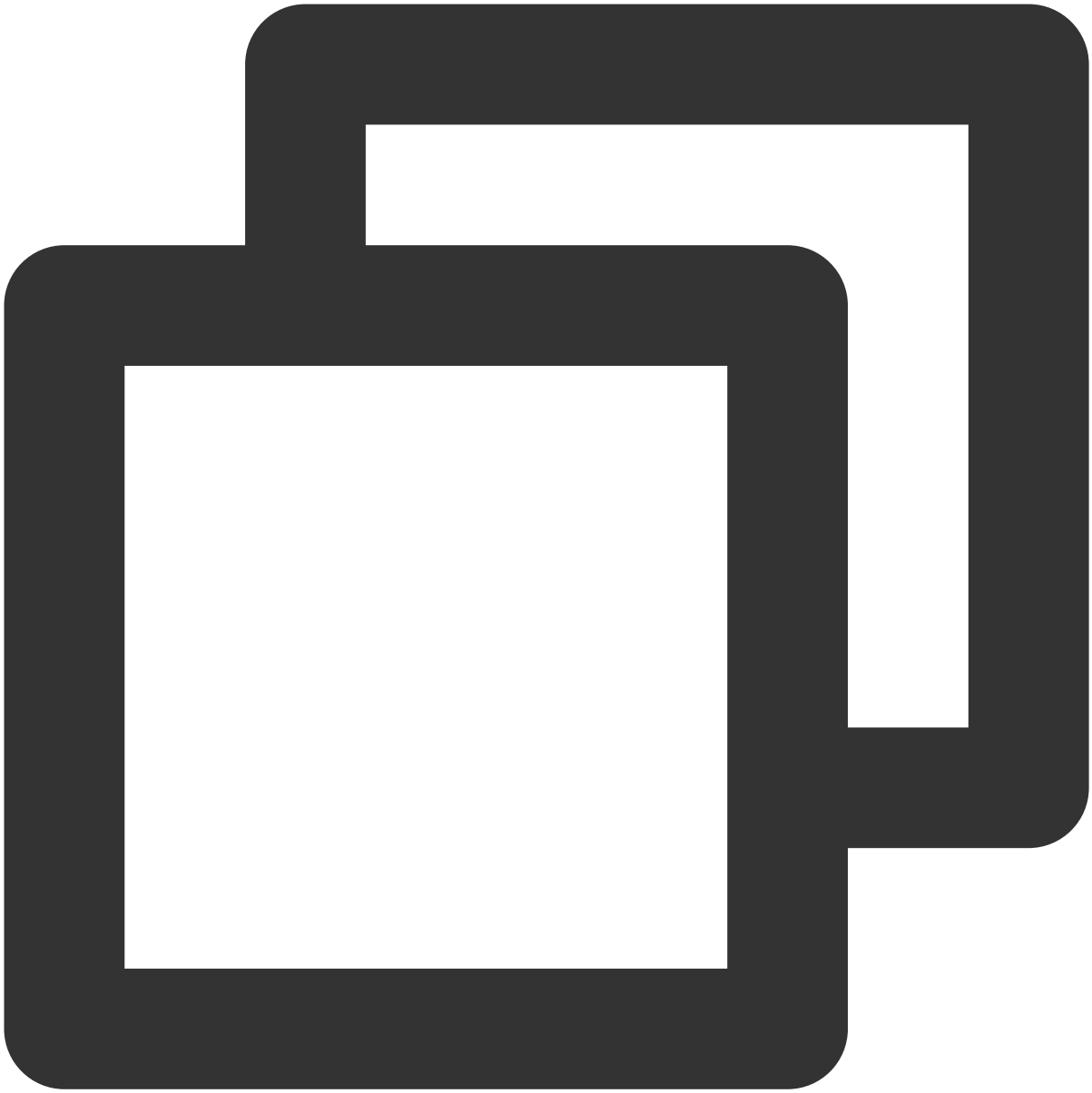
2. 新建 AudienceActivity.java，并在 `AndroidManifest.xml` 里注册， 通过加载 TUILiveKit 的 TUIVoiceRoomFragment 页面，进入直播间。

Java



```
public class AudienceActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_audience);
        //主播的房间 ID
        String roomId = "123666";

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction(
        TUIVoiceRoomFragment fragment = new TUIVoiceRoomFragment(roomId,
        LiveDefine.RoomBehavior.JOIN, null);
        fragmentTransaction.add(R.id.fl_container, fragment);
        fragmentTransaction.commit();
    }
}
```

在 app 项目的 `AndroidManifest.xml` 里注册 `AudienceActivity` （请使用您的 `AudienceActivity` 实际包名）：



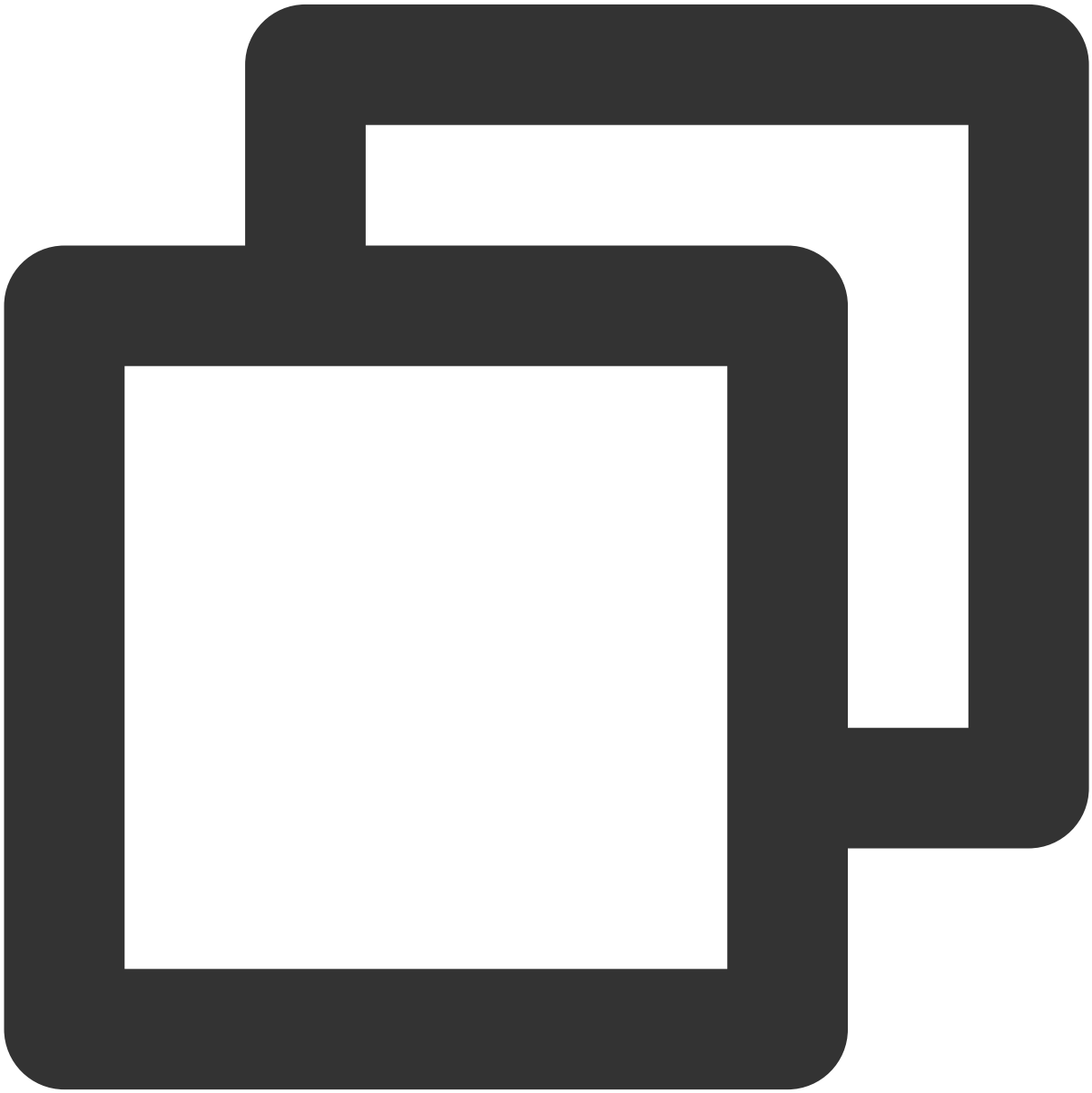
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application>
        ...
        <!-- 示例：注册AudienceActivity，请使用您的实际包名 -->
        <activity android:name="com.trtc.uikit.livekit.example.main.AudienceActivit
                  android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"/>
        ...
    </application>
</manifest>
```

注意：

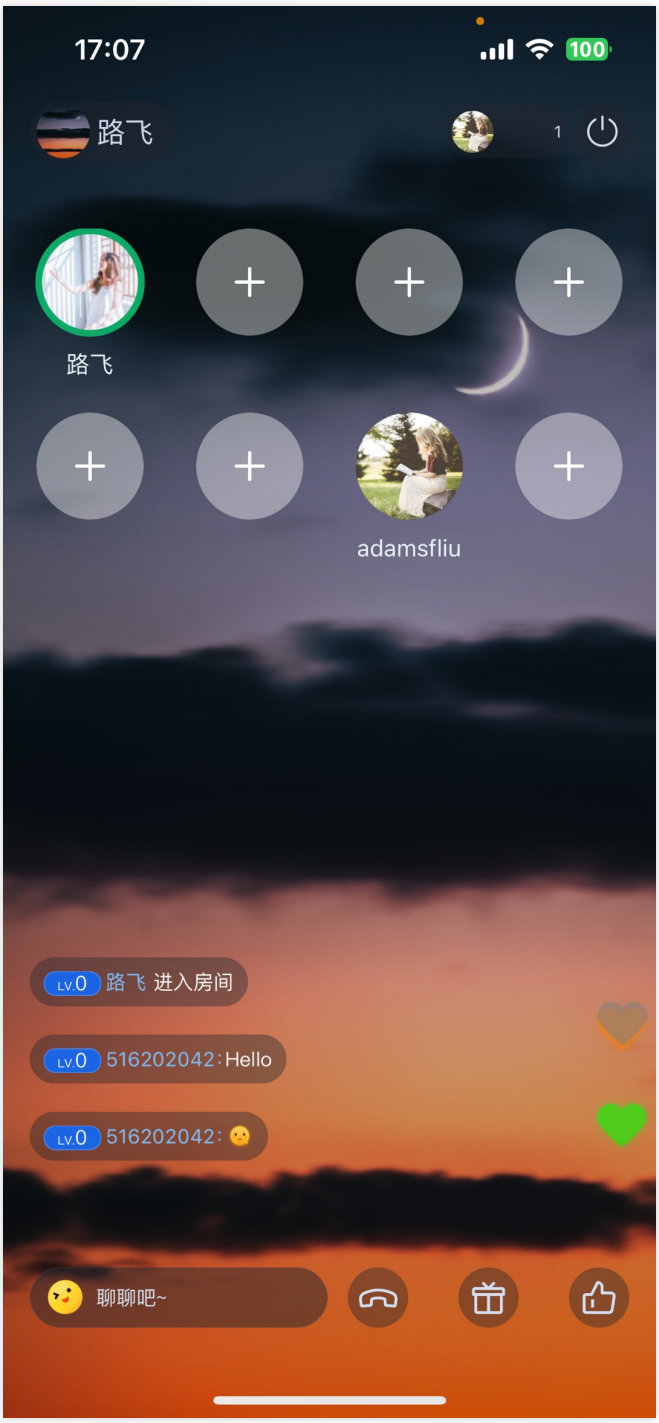
由于AudienceActivity继承自 `AppCompatActivity` ，所以要给 `AnchorActivity` 设置一个 `Theme.AppCompat` 主题。您可以修改成自己的 `Theme.AppCompat` 主题。如果遇到 Theme.AppCompat 相关问题，请参考 Activity 主题问题。

3. 在您需要观众进房的地方（具体由您的业务决定，默认情况下可在 MainActivity 某点击事件里执行），执行如下操作，拉起观众进房页面：

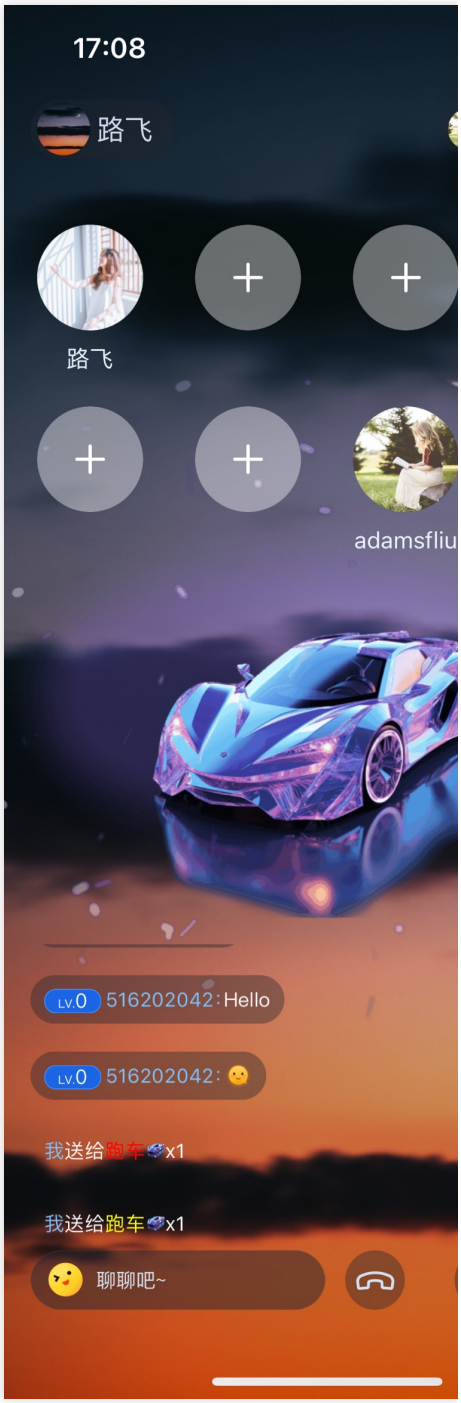
Java



```
Intent intent = new Intent(context, AudienceActivity.class);
startActivity(intent);
```



语音聊天室



语音聊天室

更多特性

弹幕
礼物

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

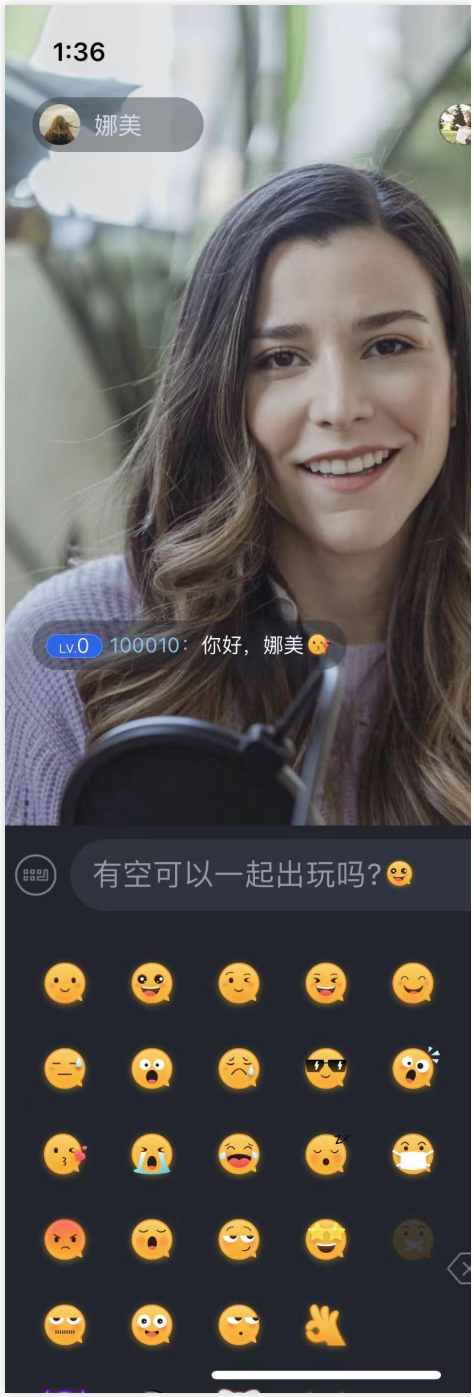
版权所有：腾讯云计算（北京）有限责任公司

互动弹幕（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:40

弹幕展示

弹幕显示



弹幕发送

说明：

支持**系统键盘**和**表情键盘**切换。

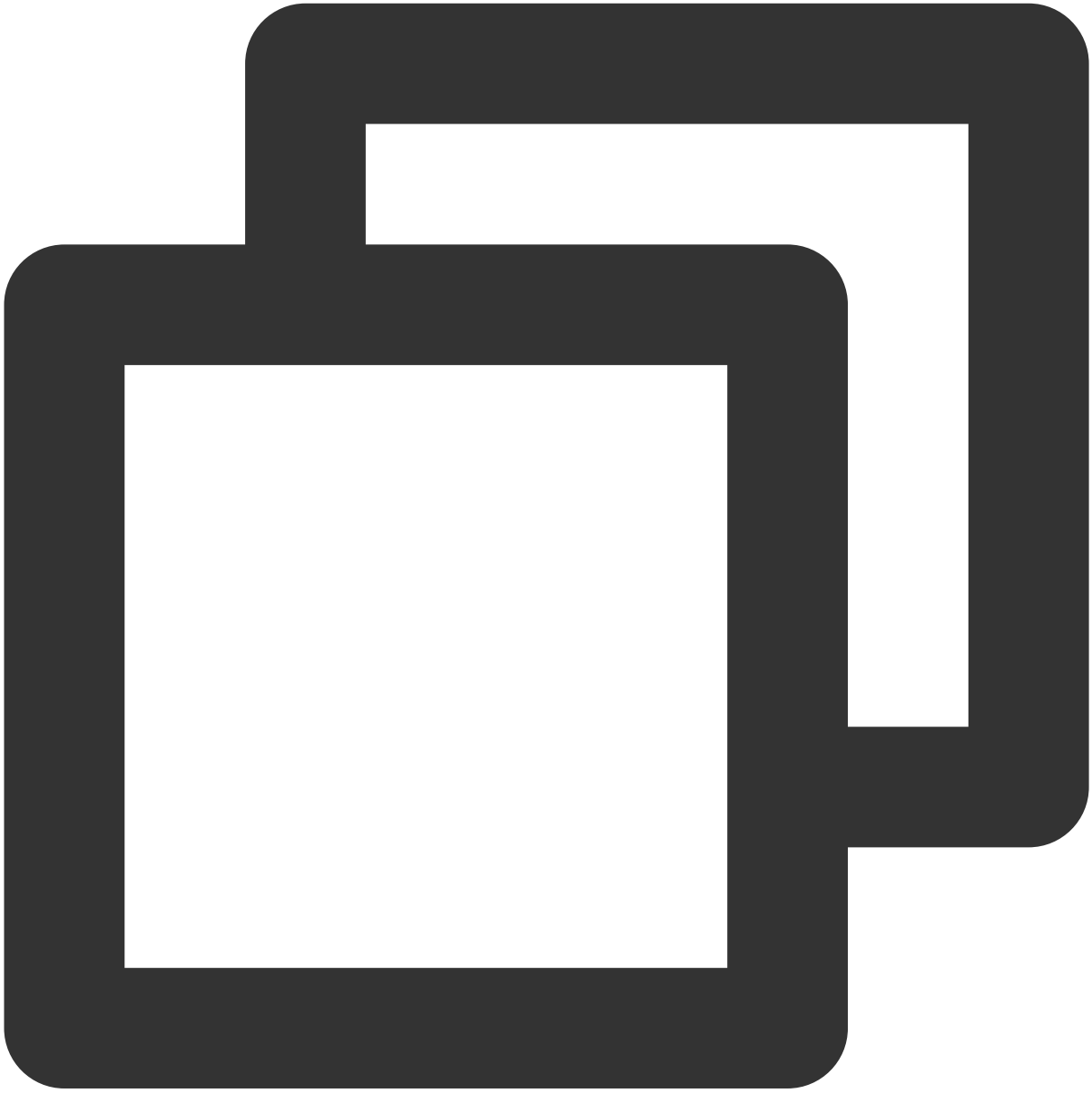
快速接入

弹幕组件主要包提供2个API：

`TUIBarrageButton` ：点击后可以拉起输入界面。

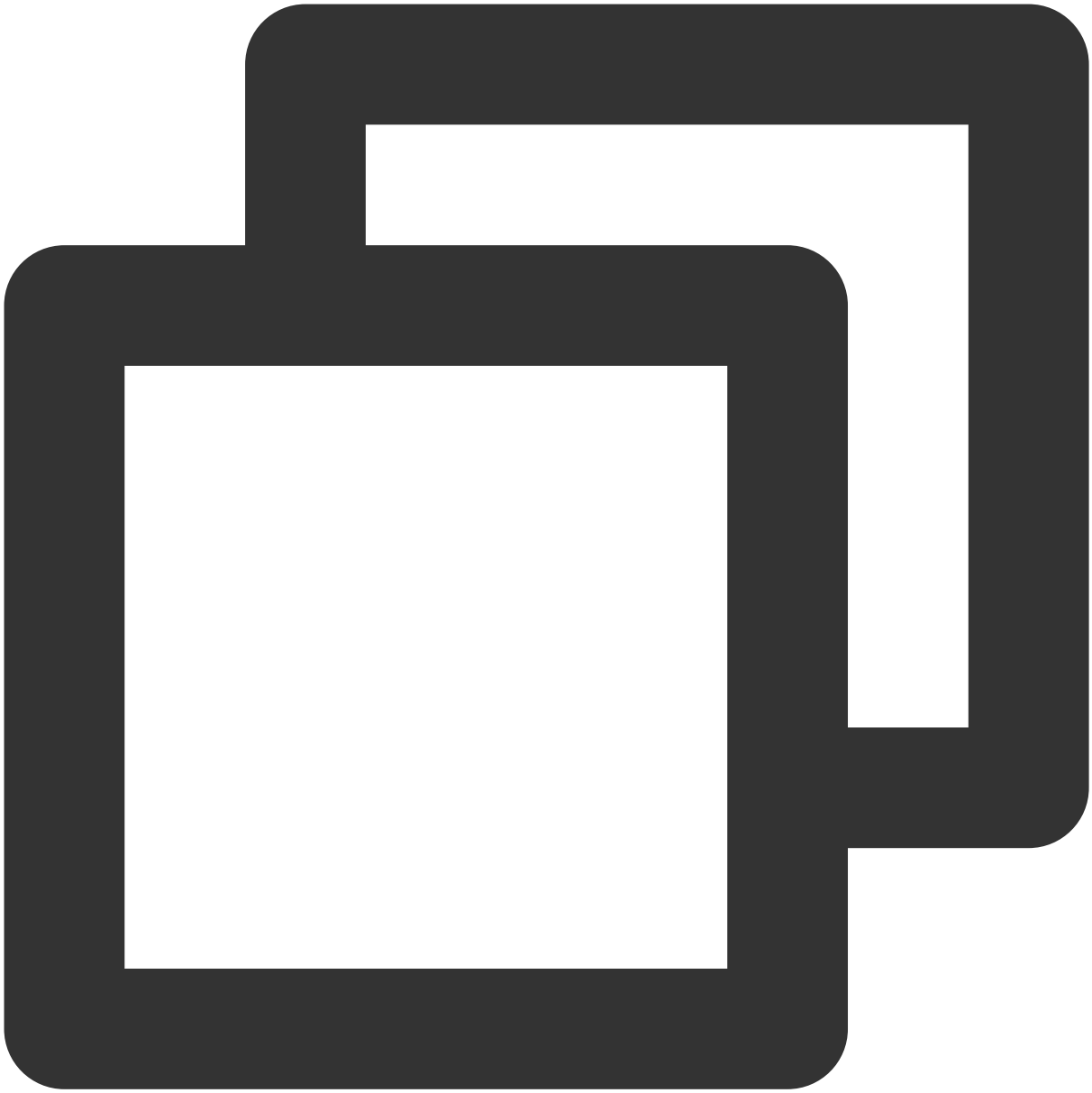
`TUIBarrageDisplayView` ：用于展示弹幕消息。

在需要发送弹幕的场景，创建 `TUIBarrageButton` ，点击后可以拉起输入界面：



```
let barrageButton: TUIBarrageButton = TUIBarrageButton(roomId: xxx)
view.addSubview(barrageButton)
//布局barrageButton
```

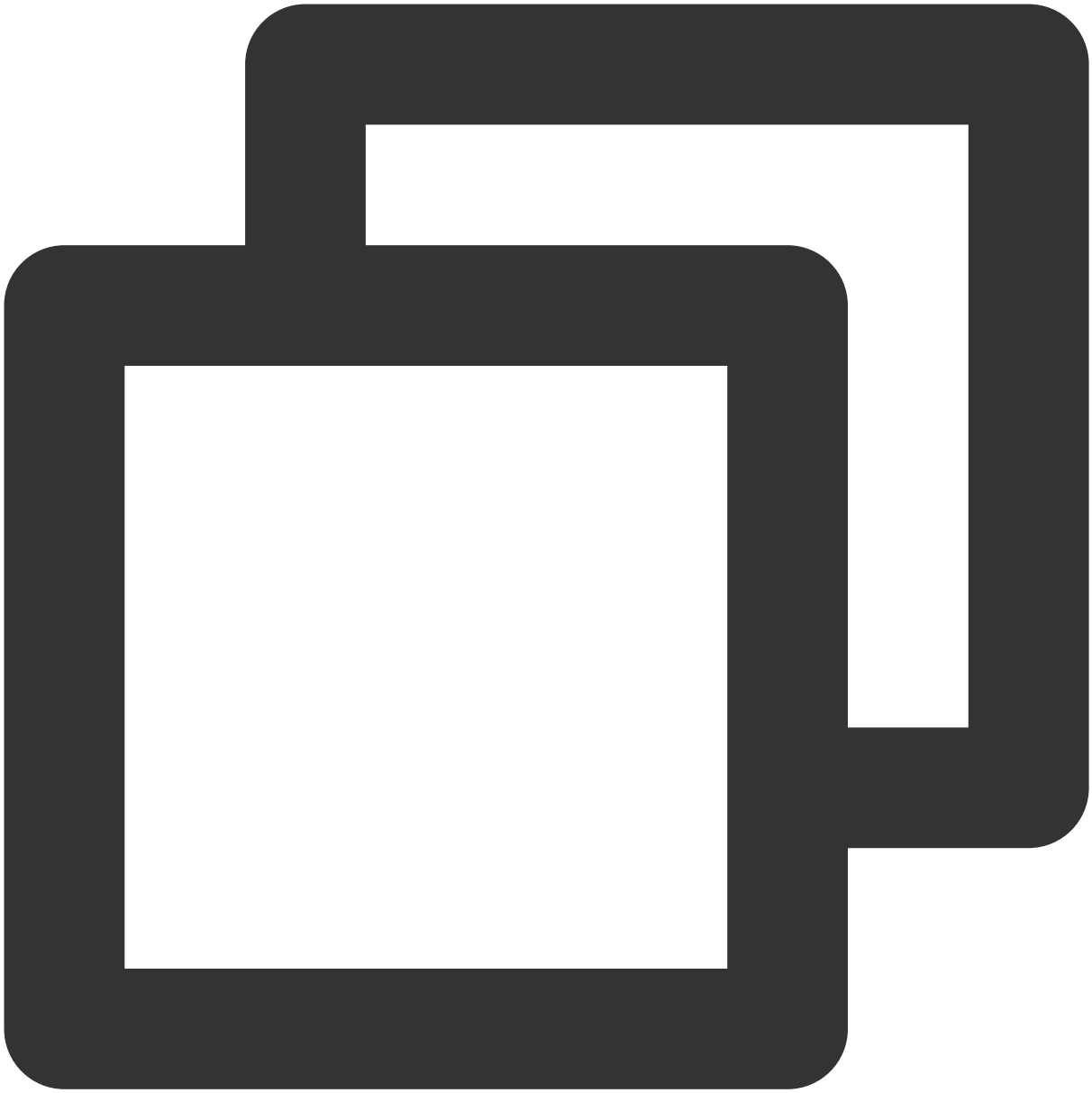
在需要展示弹幕的场景，使用 `TUIBarrageDisplayView` 来展示弹幕消息：



```
let barrageDisplayView: TUIBarrageDisplayView = TUIBarrageDisplayView(roomId: xxx)
view.addSubview(barrageDisplayView)
//布局barrageDisplayView
```

自定义消息样式

实现 `TUIBarrageDisplayView` 的代理 `TUIBarrageDisplayViewDelegate` 中的 `createCustomCell` 代理函数，用于自定义弹幕消息样式。



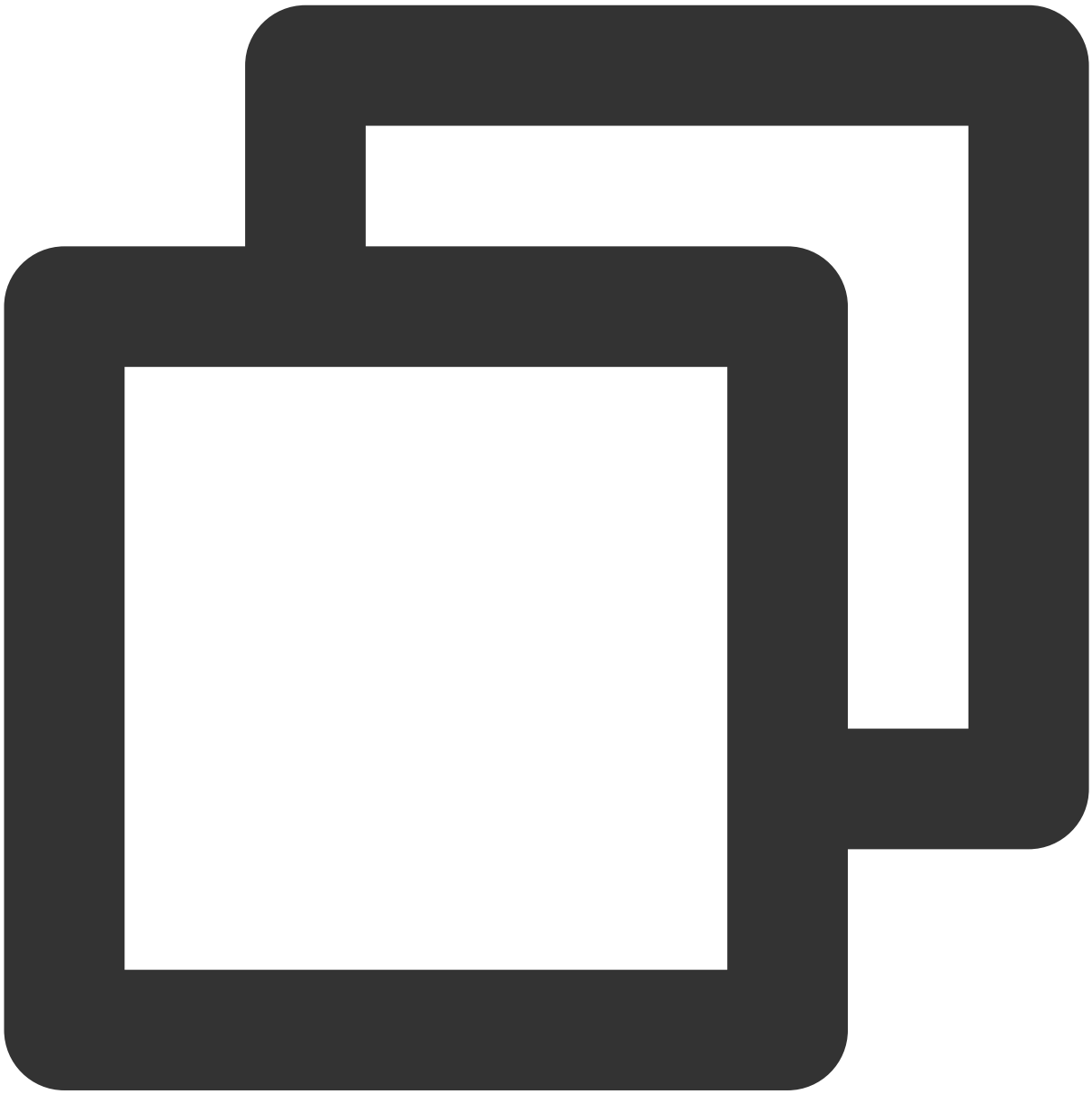
```
barrageDisplayView.delegate = self
extension UIViewController: TUIBarrageDisplayViewDelegate {
    func barrageDisplayView(_ barrageDisplayView: TUIBarrageDisplayView, createCusto
        //此处返回自定义弹幕UI
    }
}
```

说明：

`TUIBarrageDisplayView` 在展示消息时会先调用代理函数 `barrageDisplayView:createCustomCell` 用于获取用户对某条弹幕的自定义样式，若返回nil，则将采用 `TUIBarrageDisplayView` 的默认弹幕样式。

插入自定义消息

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息，通常自定义消息配合自定义样式，实现不一样的展示效果。



```
// 示例：在弹幕区插入一条礼物消息
let barrage = TUIBarrage()
barrage.content = "gift"
barrage.user.userId = sender.userId
barrage.user.userName = sender.userName
```

```
barrage.user.avatarUrl = sender.avatarUrl
barrage.user.level = sender.level
barrage.extInfo["xxx"] = "xxx"
barrageDisplayView.insertBarrages(barrage);
```

说明：

`TUIBarrage` 的 `extInfo` 是一个 `Map`，用于存放自定义数据。

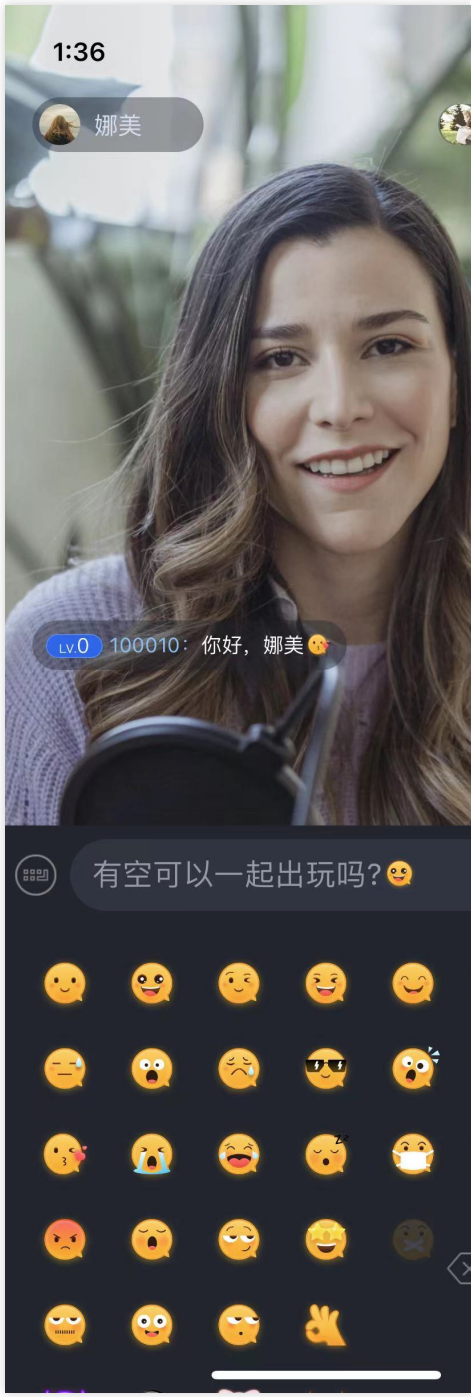
Android

最近更新时间：2024-05-17 11:20:41

互动弹幕功能，支持如下功能：发送弹幕消息、插入自定义消息、自定义消息样式等。弹幕消息支持表情输入，增加消息趣味性，让互动更愉悦。

弹幕展示

弹幕显示



弹幕发送

说明：

支持**系统键盘**和**表情键盘**切换。

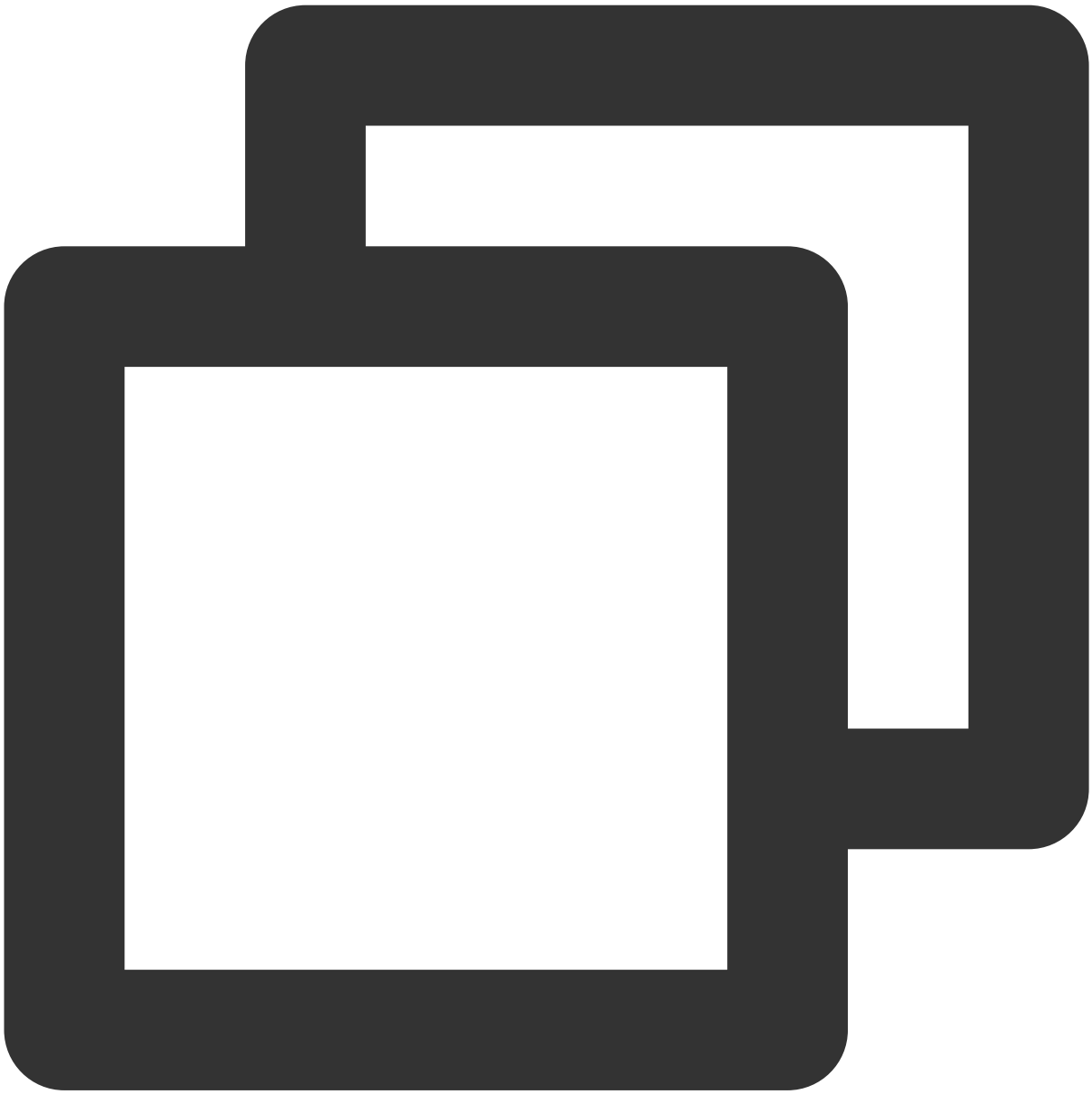
快速接入

弹幕组件主要包提供2个 API：

`TUIBarrageButton` ：点击后可以拉起输入界面。

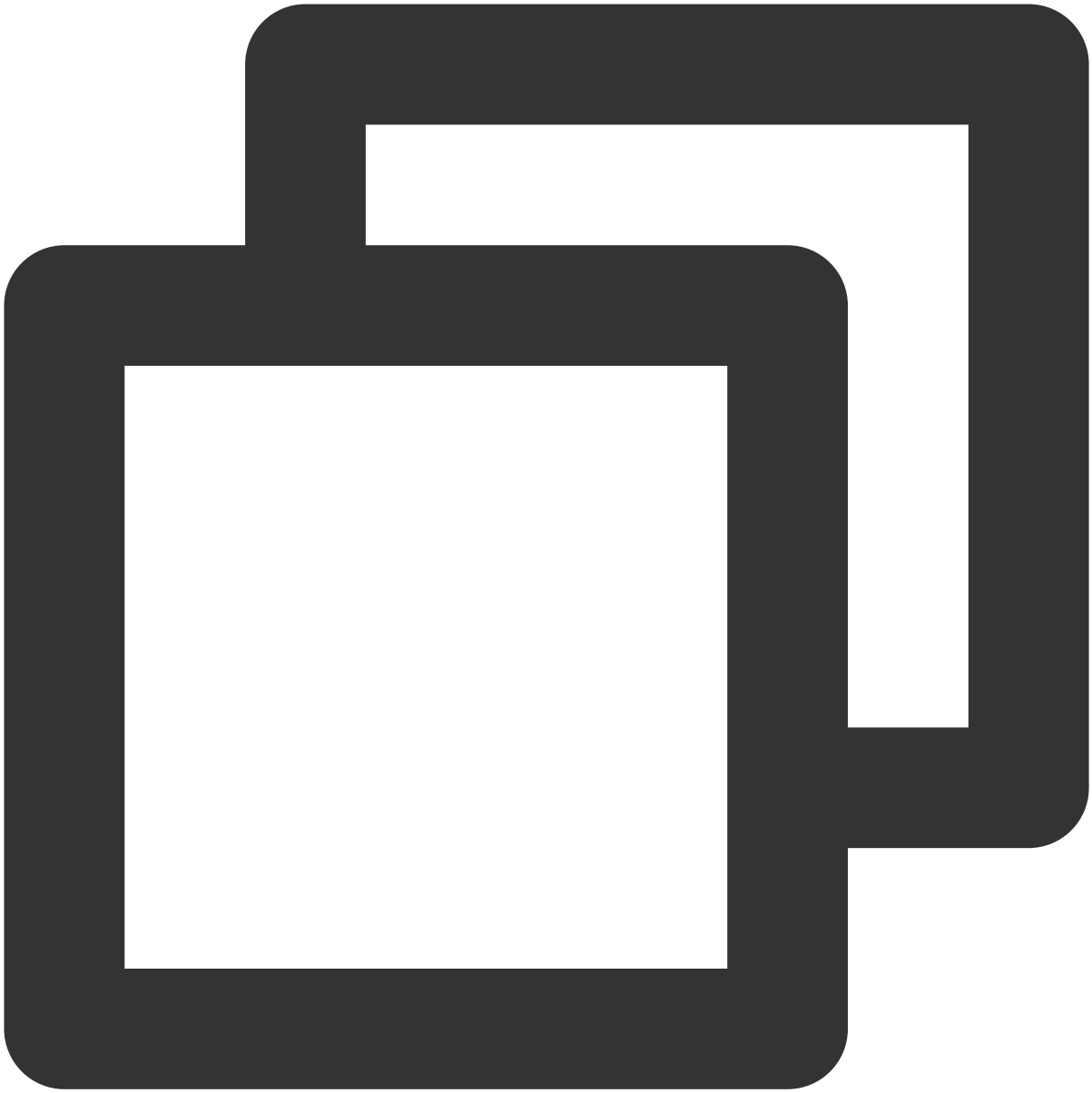
`TUIBarrageDisplayView` ：用于展示弹幕消息。

在需要发送弹幕的场景，创建 `TUIBarrageButton` ，点击后可以拉起输入界面：



```
TUIBarrageButton barrageButton = new TUIBarrageButton(mContext, roomId);
mBarrageButtonContainer.addView(barrageButton);
```

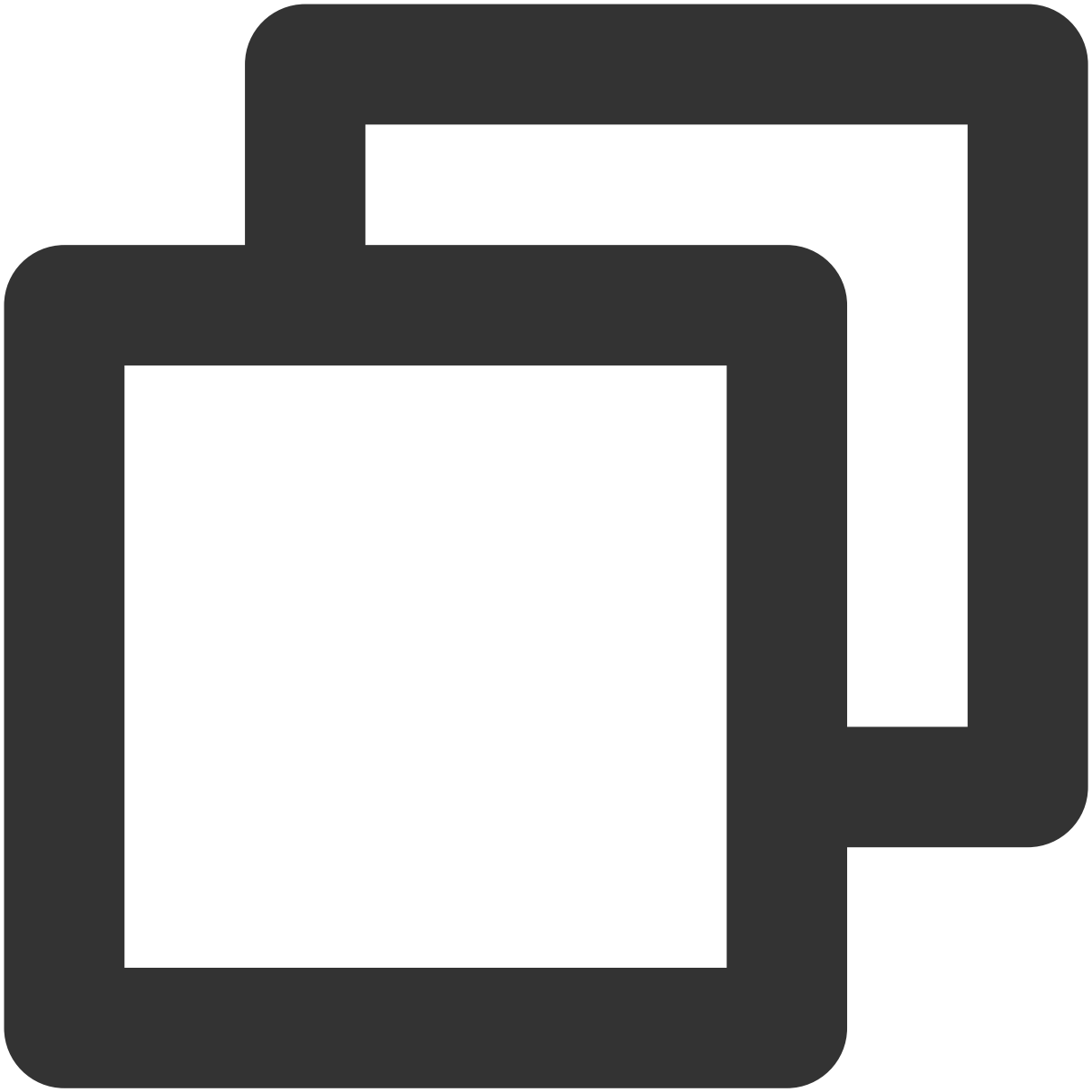
在需要展示弹幕的场景，使用 `TUIBarrageDisplayView` 来展示弹幕消息：



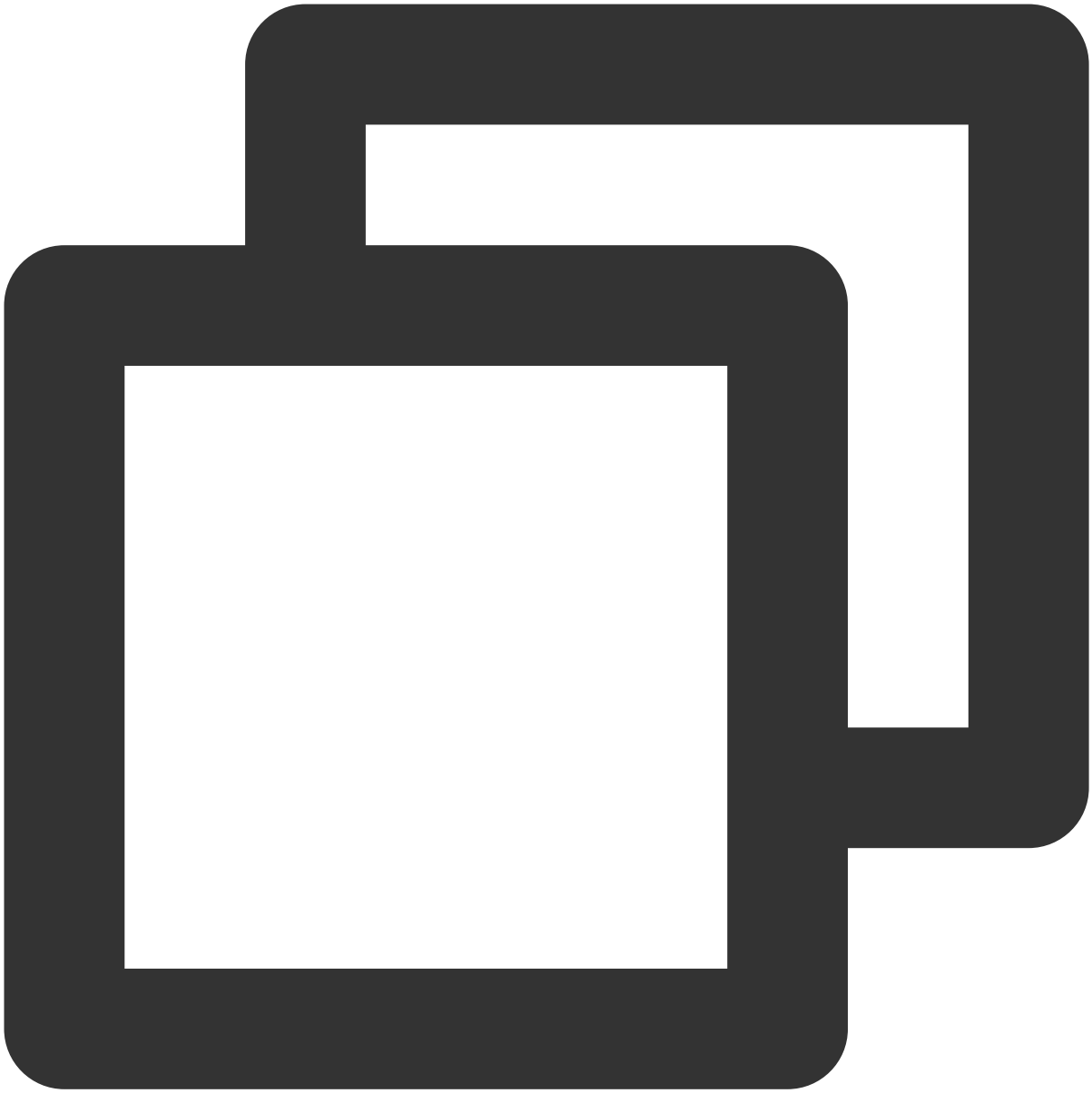
```
TUIBarrageDisplayView barrageDisplayView = new TUIBarrageDisplayView(mContext, room
mLayoutBarrageContainer.addView(barrageDisplayView);
```

自定义消息样式

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `setAdapter` 和 `TUIBarrageDisplayAdapter` ，用于自定义弹幕消息 `Item` 样式：



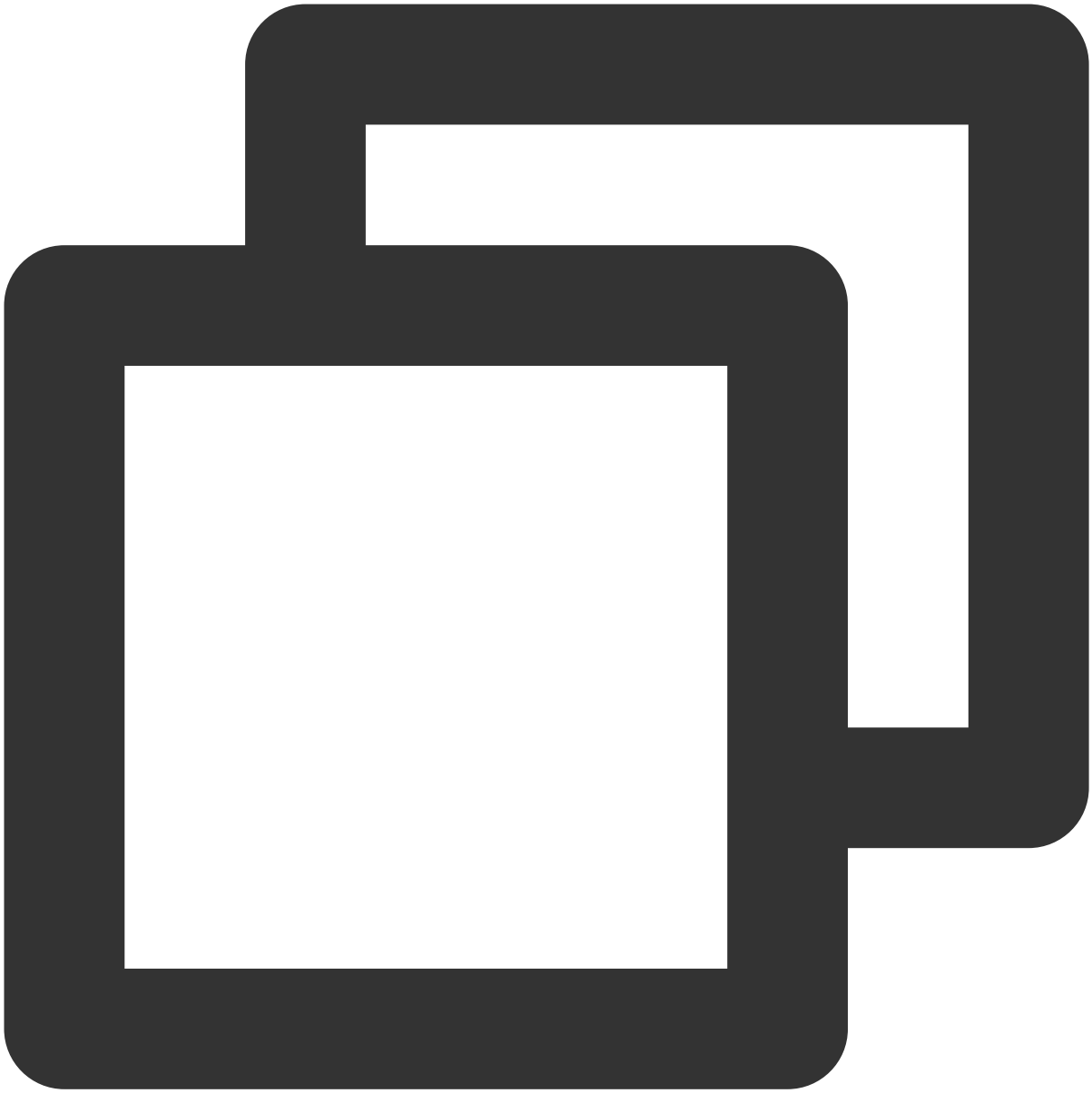
```
public interface TUIBarrageDisplayAdapter {
    RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType);

    void onBindViewHolder(RecyclerView.ViewHolder holder, TUIBarrage barrage);

    int getItemViewType(int position, TUIBarrage barrage);
}
```



```
public class GiftBarrageAdapter implements TUIBarrageDisplayAdapter {
    private final Context       mContext;

    public GiftBarrageAdapter(Context context) {
        mContext = context;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewTyp
        if (viewType == GIFT_VIEW_TYPE_1) {
```

```
            // 自定义样式1的处理
            LinearLayout ll = new LinearLayout(mContext);
            ll.addView(new TextView(mContext));
            return new GiftViewHolder(ll);
        }
        return null;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, TUIBarrage barrage
        if (holder instanceof GiftViewHolder) {
            GiftViewHolder viewHolder = (GiftViewHolder) holder;
            viewHolder.bind(barrage);
        }
    }

    @Override
    public int getItemViewType(int position, TUIBarrage barrage) {
        if (...) { // 如果当前 barrage 需要自定义 Item 样式，则返回对应样式type
            return GIFT_VIEW_TYPE_1;
        }
        return 0; // 0 表示使用默认样式
    }

    private static class GiftViewHolder extends RecyclerView.ViewHolder {

        public GiftViewHolder(View itemView) {
            super(itemView);
            // ...
        }

        public void bind(TUIBarrage barrage) {
            // ...
        }
    }
}

// 设置自定义 Adapter
barrageDisplayView.setAdapter(new GiftBarrageAdapter(mContext));
```

`TUIBarrage` 定义如下：



```
public class TUIBarrage {
    public final TUIBarrageUser    user = new TUIBarrageUser(); //发送内容
    public String                  content;                     //扩展信息
    public HashMap<String, Object> extInfo = new HashMap<>();   //扩展信息
}

public class TUIBarrageUser {
    public String userId;
    public String userName;
    public String avatarUrl;
```

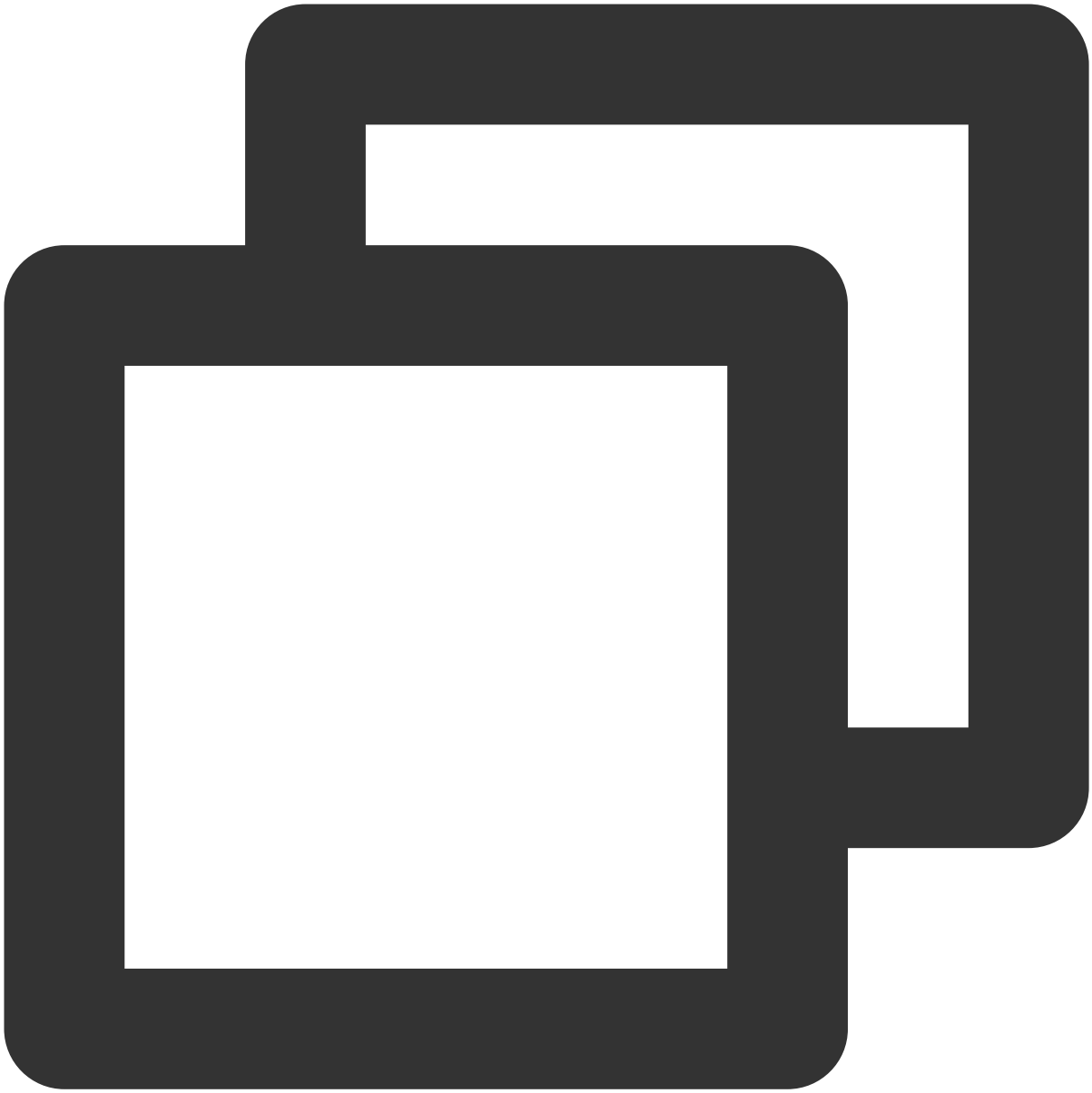
```
    public String level;
}
```

说明：

支持多种自定义样式（通过 `getItemViewType` 指定多种返回值），0 表示默认样式。

插入自定义消息

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息，通常自定义消息配合自定义样式，实现不一样的展示效果。



```
// 示例：在弹幕区插入一条礼物消息
TUIBarrage barrage = new TUIBarrage();
barrage.content = "gift";
barrage.user.userId = sender.userId;
barrage.user.userName = sender.userName;
barrage.user.avatarUrl = sender.avatarUrl;
barrage.user.level = sender.level;
barrage.extInfo.put(Constants.GIFT_VIEW_TYPE, GIFT_VIEW_TYPE_1);
barrage.extInfo.put(GIFT_NAME, barrage.giftName);
barrage.extInfo.put(GIFT_COUNT, giftCount);
barrage.extInfo.put(GIFT_ICON_URL, barrage.imageUrl);
```

版权所有：腾讯云计算（北京）有限责任公司

```
barrage.extInfo.put(GIFT_RECEIVER_USERNAME, receiver.userName);
barrageDisplayView.insertBarrages(barrage);
```

说明：

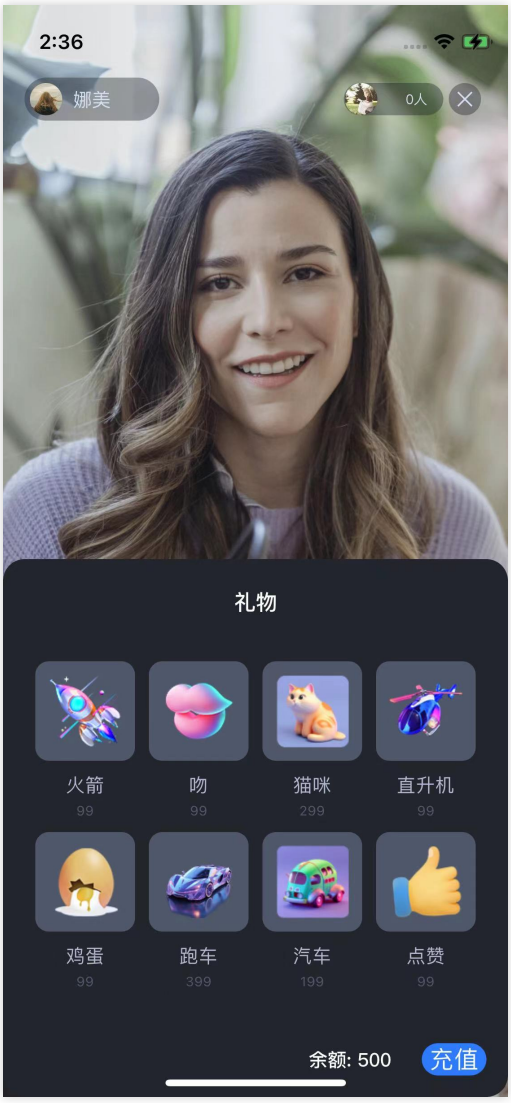
`TUIBarrage` 的 `extInfo` 是一个 `Map`，用于存放自定义数据。

互动礼物（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:41

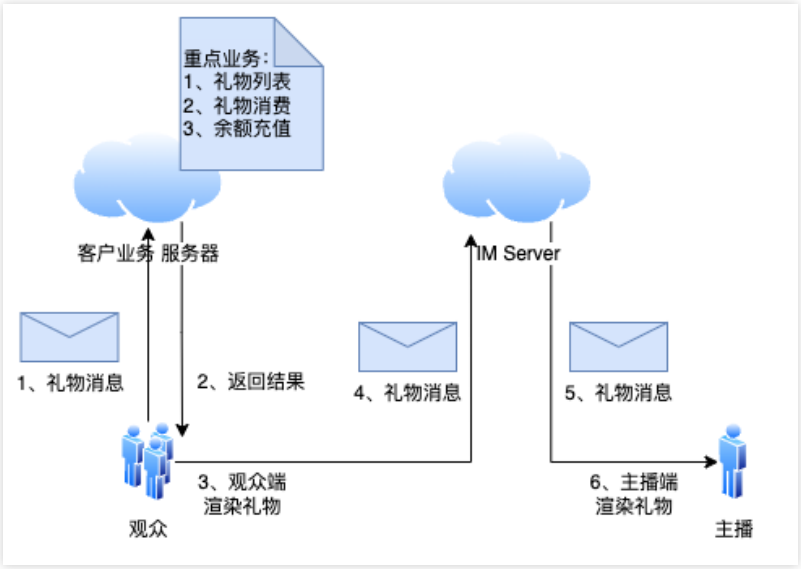
互动礼物组件是一款虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。借助这一组件，用户能够向自己欣赏的直播主播送上虚拟礼物，以此展示他们的赞赏、喜爱以及支持。
互动礼物组件支持设置**礼物素材、余额显示、普通礼物播放与全屏礼物播放、充值按钮**等。

礼物展示

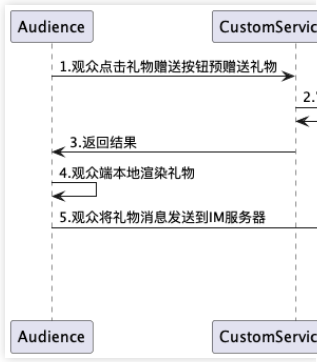


礼物展示面板	普通礼物播放效果

礼物系统



礼物系统结构图



礼物系统时序图

快速接入

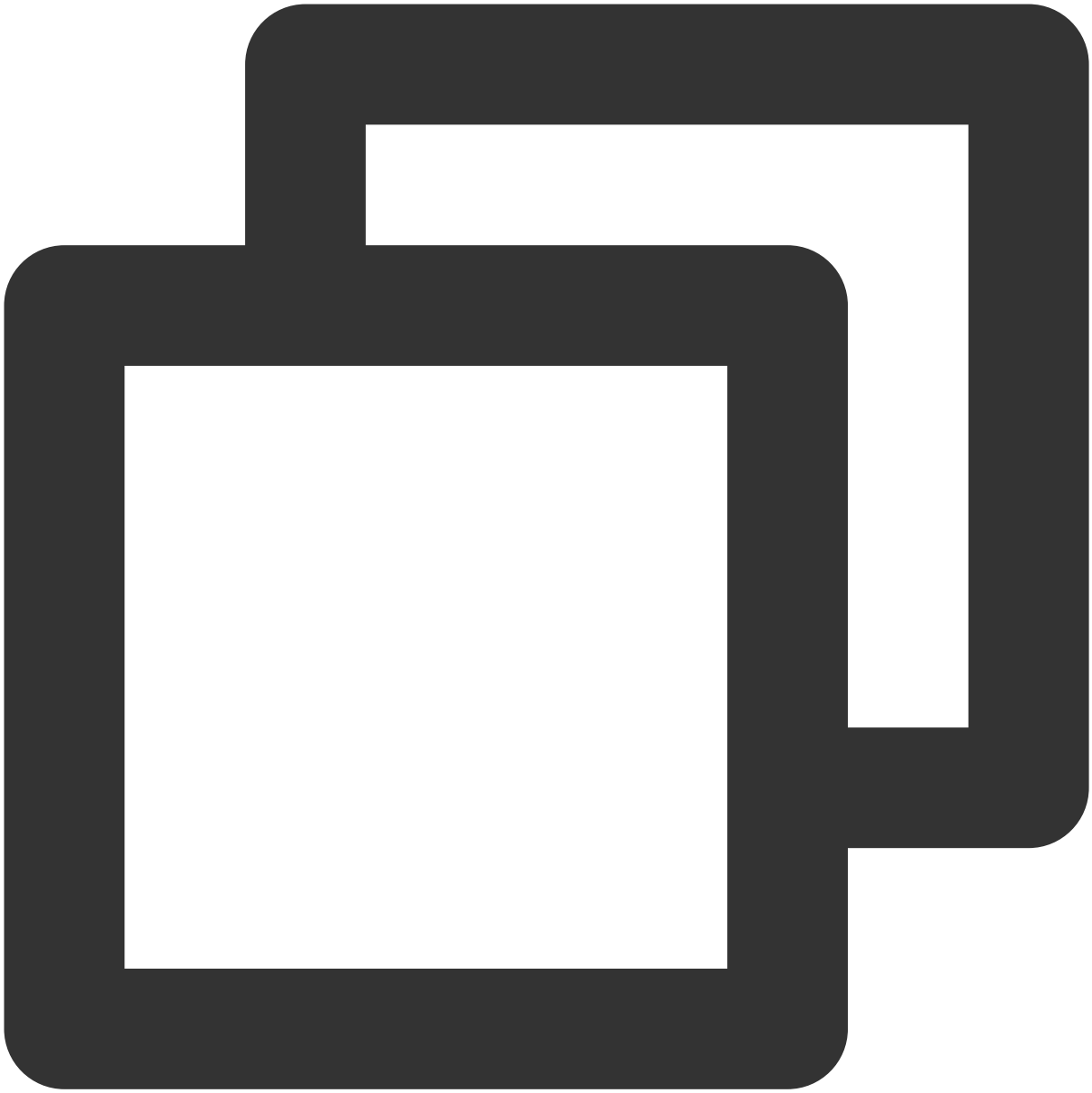
礼物组件主要提供2个 API：

`TUIGiftListView` ：礼物面板，呈现礼物列表，发送礼物及充值。

`TUIGiftPlayView` ：播放礼物的面板，自动监听礼物消息。

设置礼物素材

礼物面板组件 `TUIGiftListView` 提供了 `setGiftList` 接口，可用于设置礼物素材。



```
let giftListView: TUIGiftListView = TUIGiftListView(groupId: xxx) //生成礼物面板对象
let giftList: [TUIGift] = ... //这里可替换成自己的礼物素材数组
giftListView.setGiftList(giftList) //设置礼物面板的素材
```

说明：

`TUIGift` 的参数及说明如下：

`giftId: String`	礼物ID
`giftName: String`	礼物名称
`imageUrl: String`	礼物面板展示图像

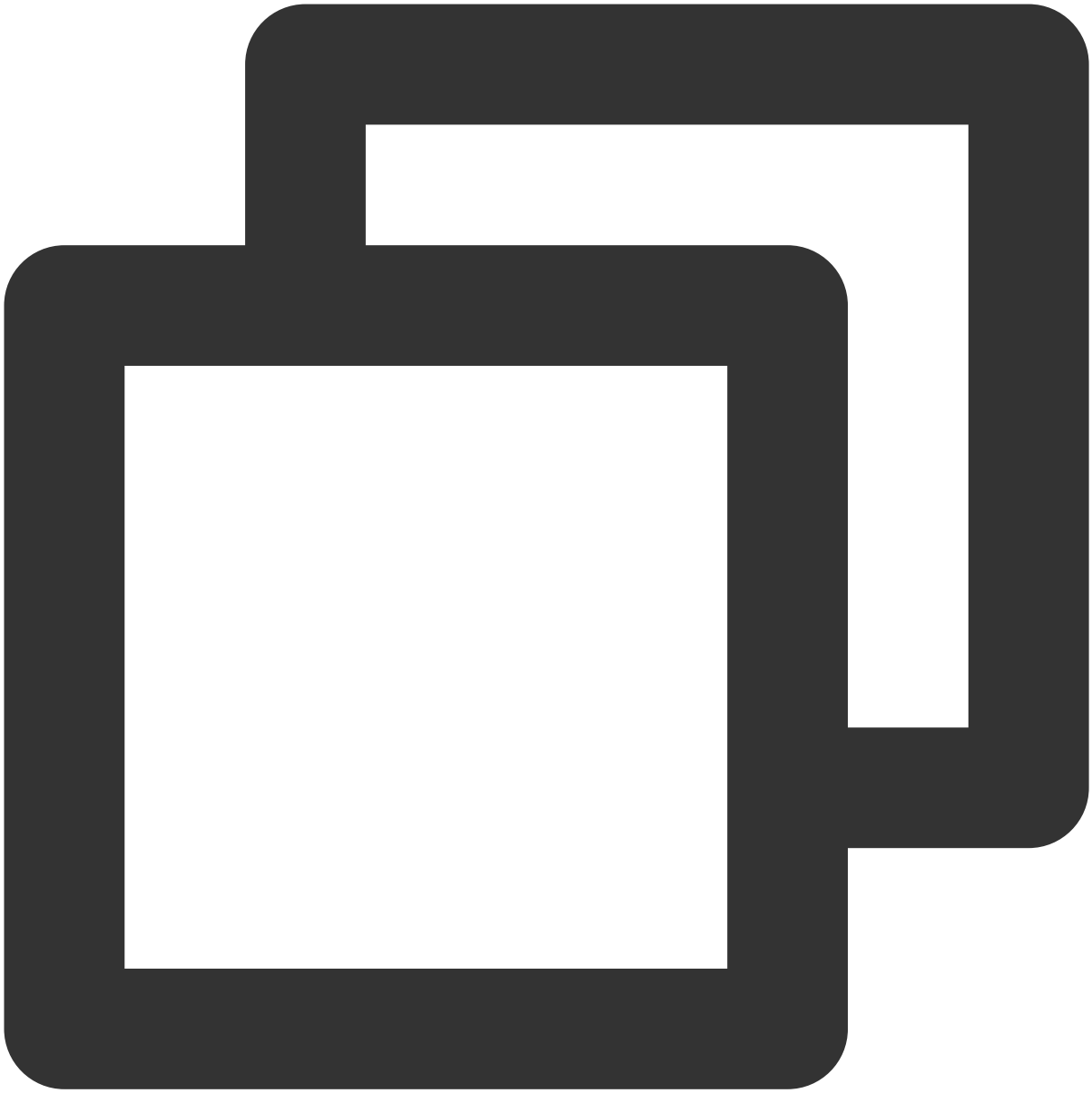
`animationUrl: String`	SVGA动画Url
`price: Int`	礼物价格
`extInfo: [String: AnyCodable]`	自定义扩展信息

互动礼物组件支持设置自己的礼物素材。若 `animationUrl` 为空，则礼物播放效果为普通播放，播放的内容为imageUrl所链接的图像；若 `animationUrl` 不为空，则播放效果为全屏播放，播放的内容为对应的svga动画。

赠送礼物

实现 `TUIGiftListView` 的代理 `TUIGiftListViewDelegate` 中的 `onSendGift` 代理函数，获取礼物个数和礼物信息，在预处理完后可调用 `TUIGiftListView` 的 `sendGift` 函数用于礼物的实际发送。

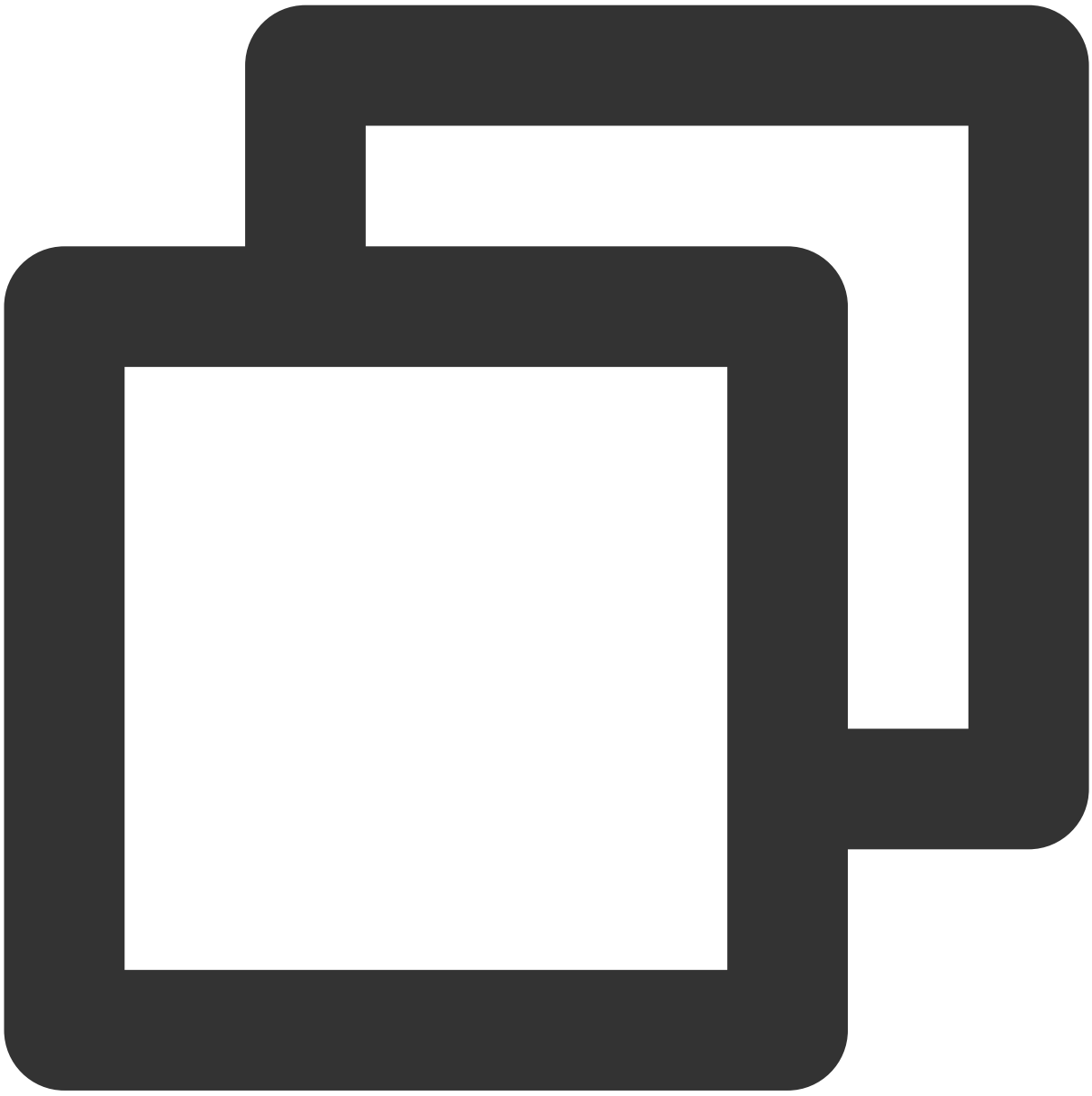
版权所有：腾讯云计算（北京）有限责任公司



```
giftListView.delegate = self

extension ViewController: TUIGiftListViewDelegate {
    func onSendGift(giftListView view: TUIGiftListView, giftModel: TUIGift, giftCou
        //...此处为预处理，如校验当前用户的余额等操作
        let receiver = TUIGiftUser()
        //...此处设置礼物接受者信息
        view.sendGift(giftModel: giftModel, giftCount: giftCount, receiver: receive
    }
}
```

接收礼物

礼物展示组件 `TUIGiftPlayView` 自身会接收并播放礼物消息。



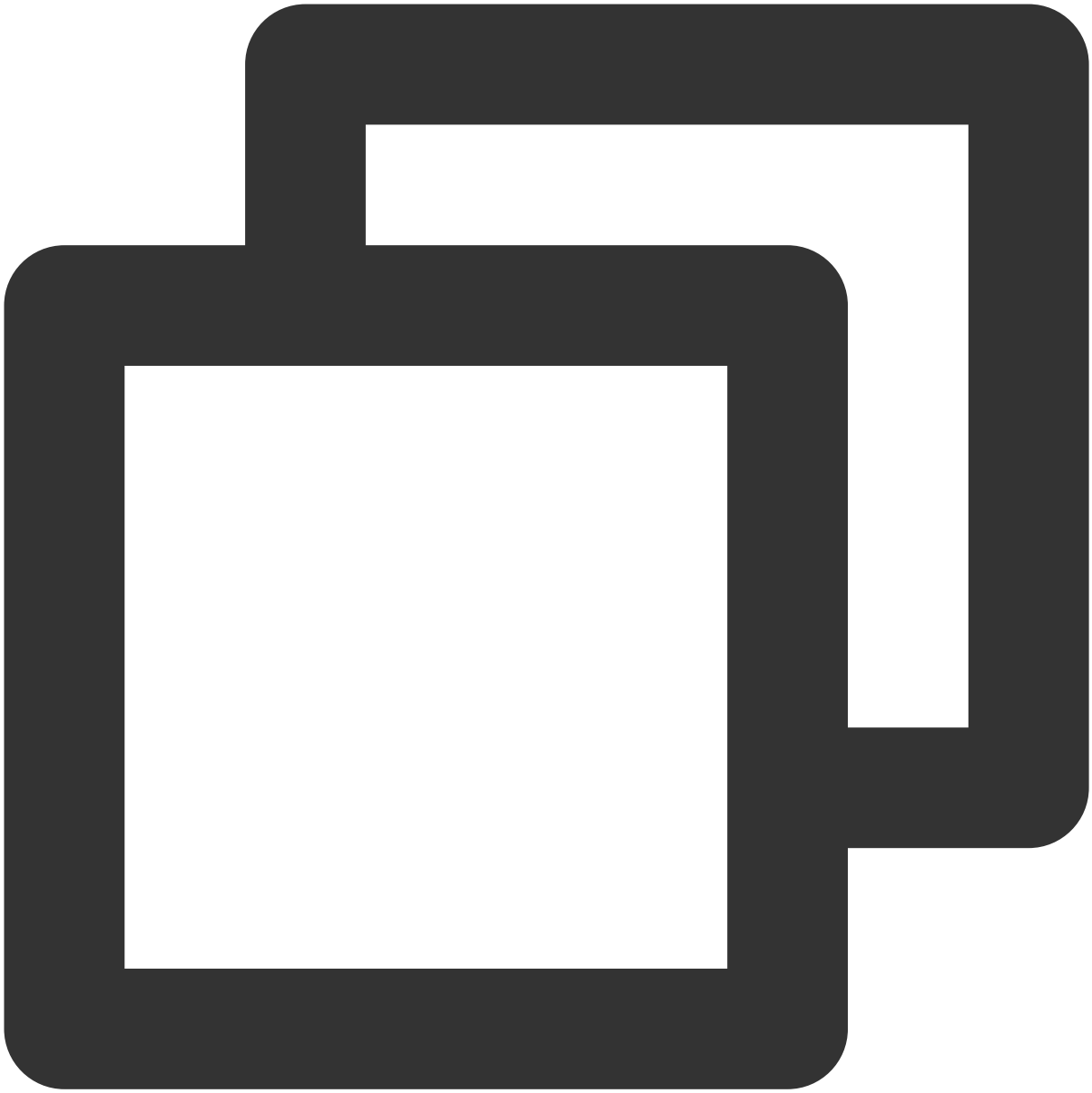
```
let giftDisplayView: TUIGiftPlayView = TUIGiftPlayView(groupId:xxx)
```

说明：

`TUIGiftPlayView` 需要全屏接入。

若需要获取接收礼物的回调信息，可将实现 `TUIGiftPlayView` 的代理 `TUIGiftPlayViewDelegate` 中的 `giftPlayView:onPlayGift` 代理函数。

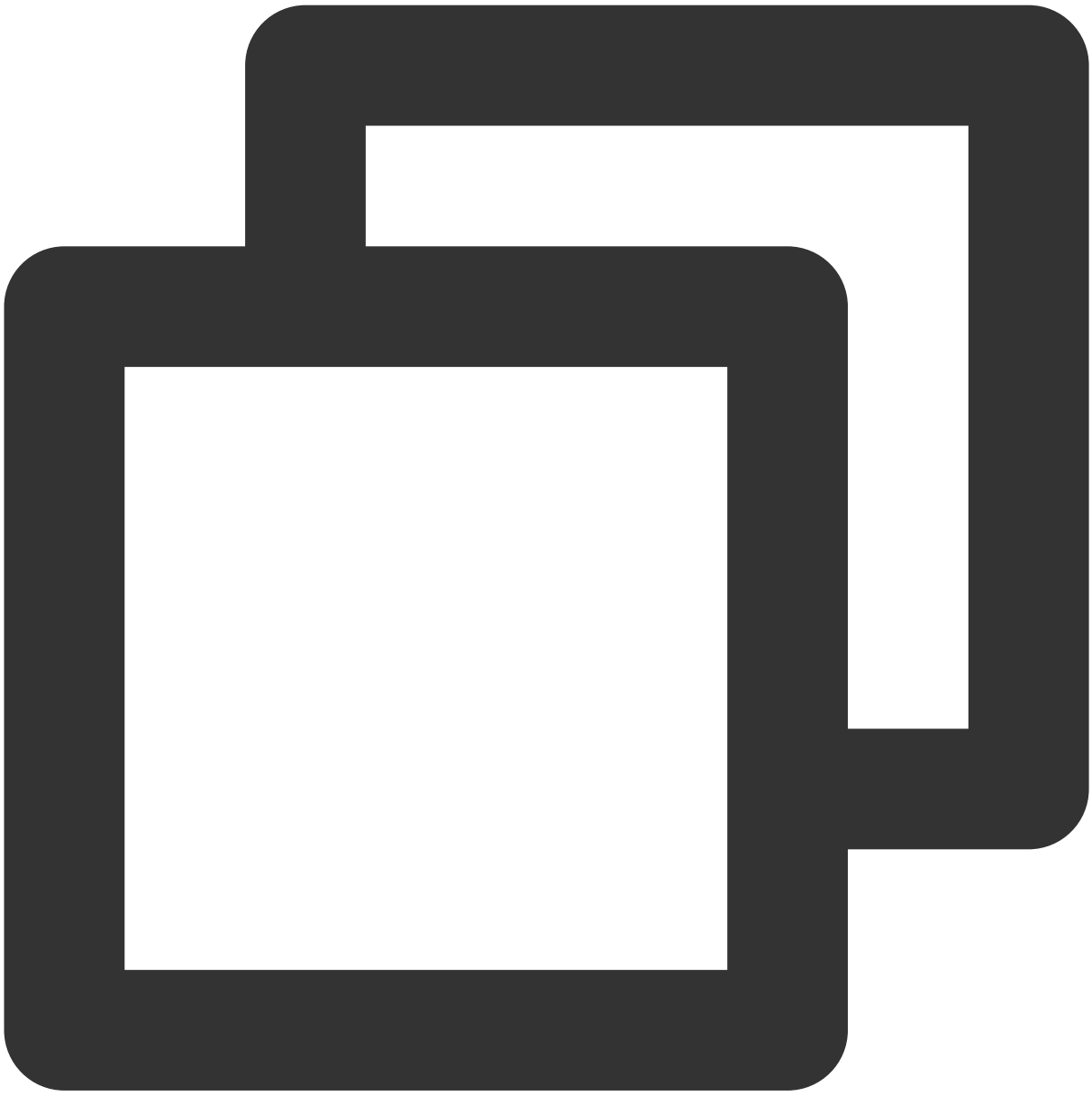
版权所有：腾讯云计算（北京）有限责任公司



```
extension ViewController: TUIGiftPlayViewDelegate {
    func giftPlayView(_ giftPlayView: TUIGiftPlayView, onPlayGift gift: TUIGift, gi
     //...可在此处自行处理
    }
}
```

设置余额

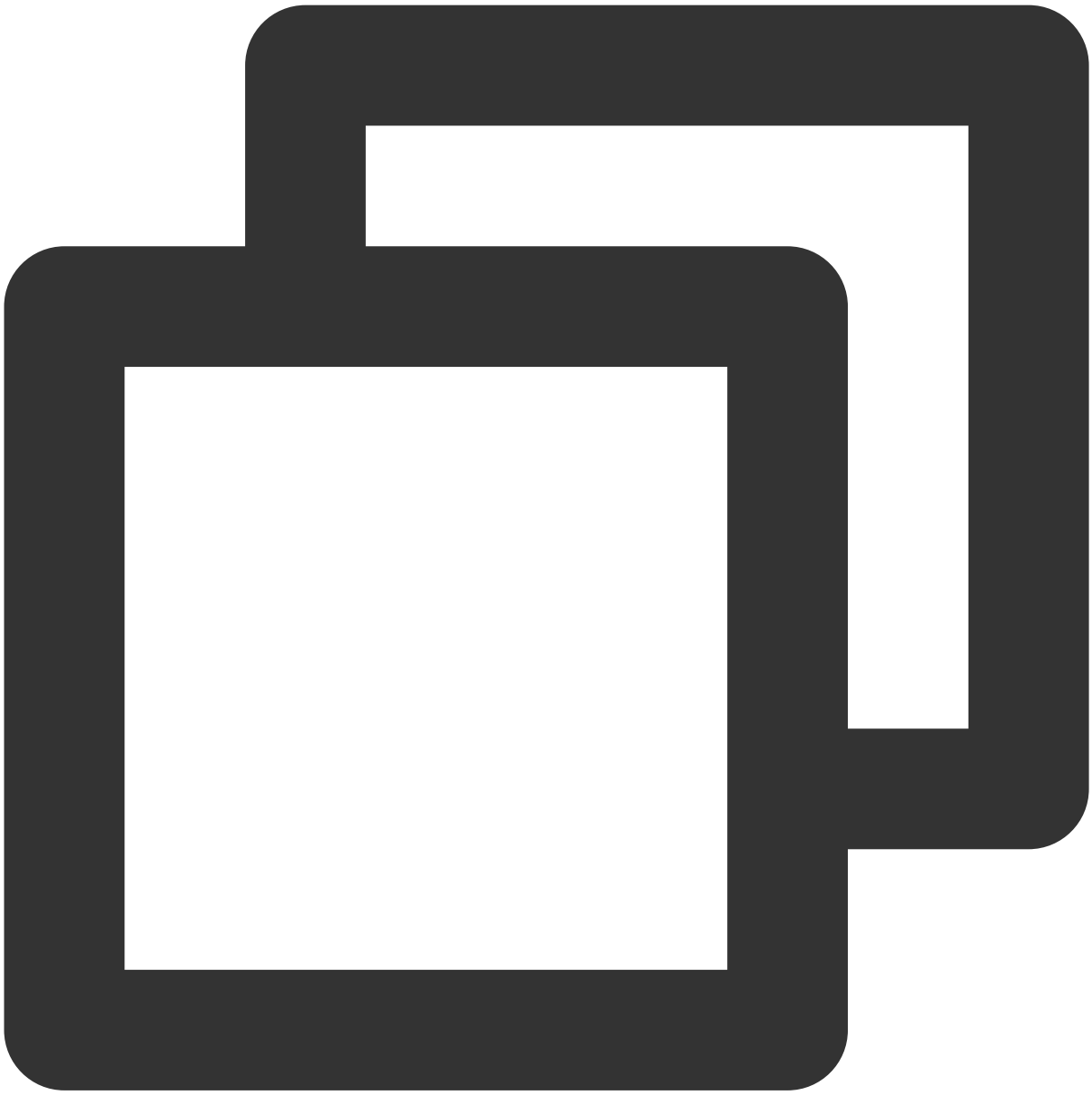
礼物面板组件 `TUIGiftListView` 提供了 `setBalance` 接口，可用于设置礼物面板上显示的余额值。



```
giftListView.setBalance(xxx)
```

充值

实现 `TUIGiftListView` 的代理 `TUIGiftListViewDelegate` 中的 `onRecharge` 代理函数可用于接收礼物展示面板抛出的充值按钮点击事件，在这里对接自己的充值系统。



```
extension ViewController: TUIGiftListViewDelegate {
    func onRecharge(giftListView view: TUIGiftListView) {
        //...此处可用于对接自己的充值系统，充值完毕后，可调用view.setBalance(xxx)设置礼物展示
    }
}
```

注意：

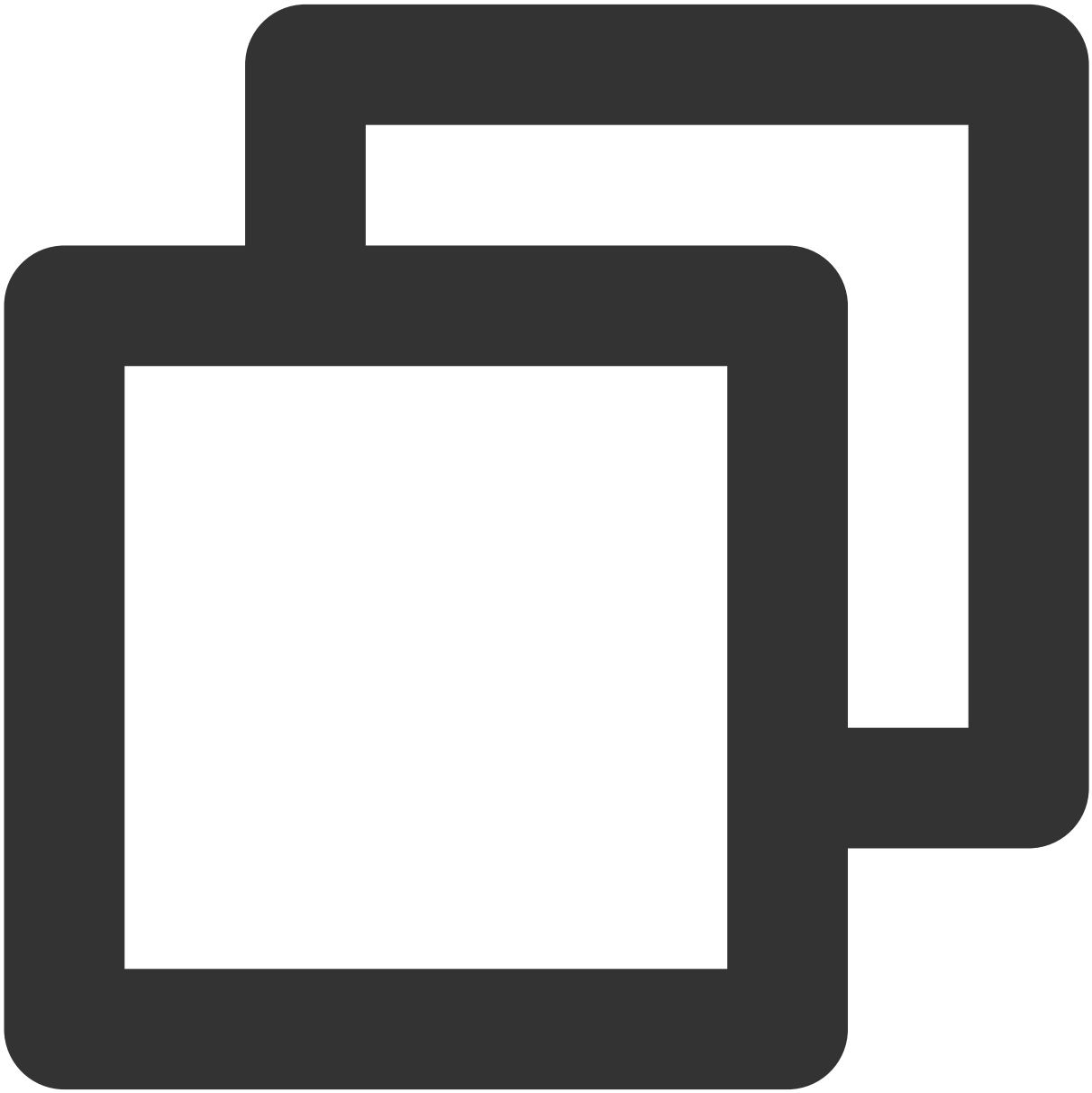
礼物余额是个虚拟币的概念，并不是真实货币。

礼物充值逻辑，由外部实现，客户可以接入自己的充值系统。充值完毕后再更新礼物余额。

修改源码，自定义处理

1、自定义礼物列表

修改观众端礼物面板的礼物列表：



```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw
```

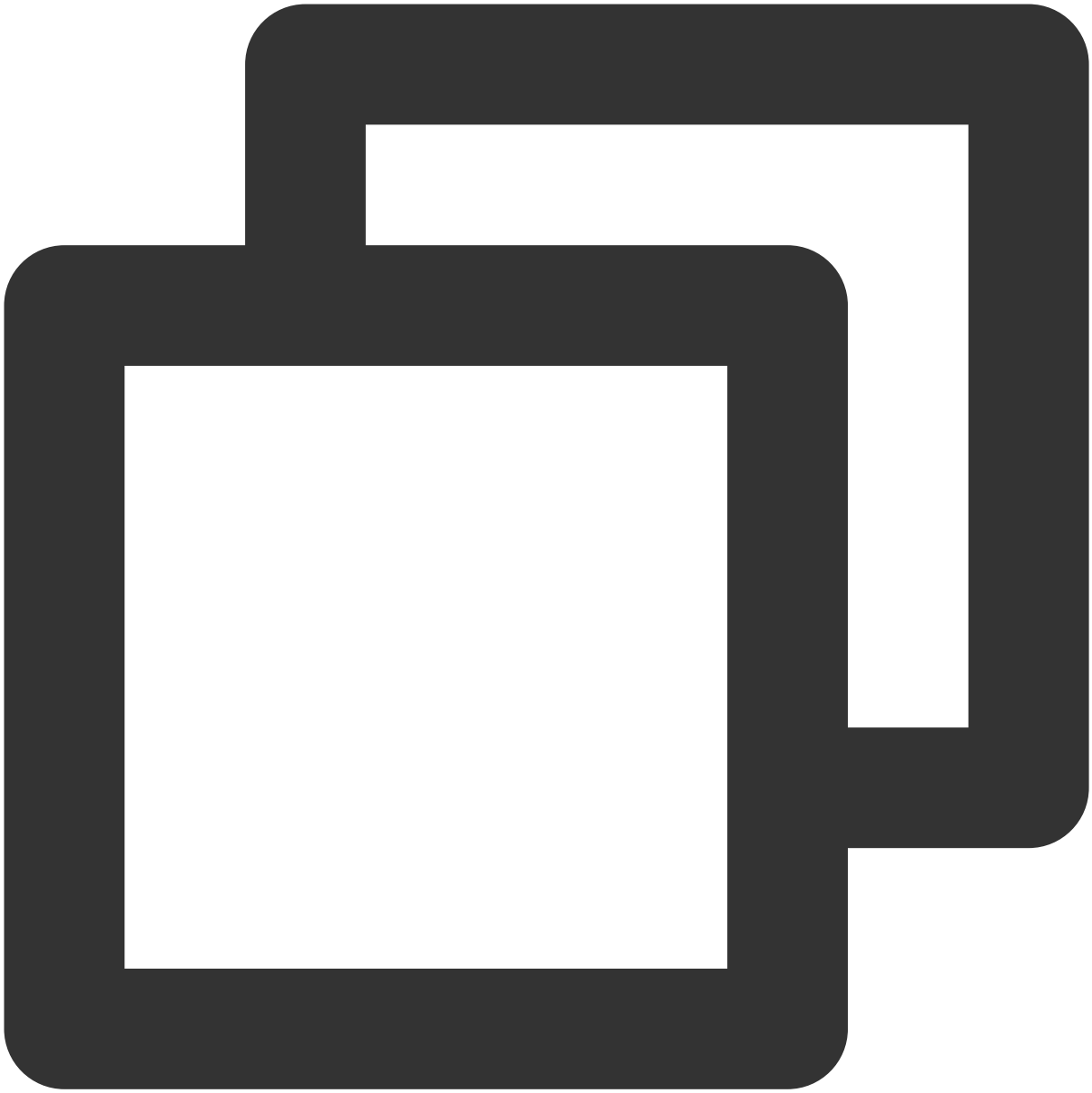
```
private lazy var giftPanelView: TUIGiftListView = {
    let view = TUIGiftListView(groupId: liveRoomInfo.roomId.value)
    giftCloudServer.queryGiftInfoList { [weak self] error, giftList in
        guard let self = self else { return }
        DispatchQueue.main.async {
            if error == .noError {
                view.setGiftList(giftList)
            } else {
                self.makeToast("query gift list error, code = \\(error)")
            }
        }
    }
    return view
}()
```

说明：

客户自行实现 `giftCloudServer.queryGiftInfoList` 的逻辑，得到一个自定义的礼物列表 `[TUIGift]`，通过 `view.setGiftList` 设置礼物列表即可。

礼物的 `animationUrl` 要求是一个SVGA动画。

2、自定义礼物余额充值



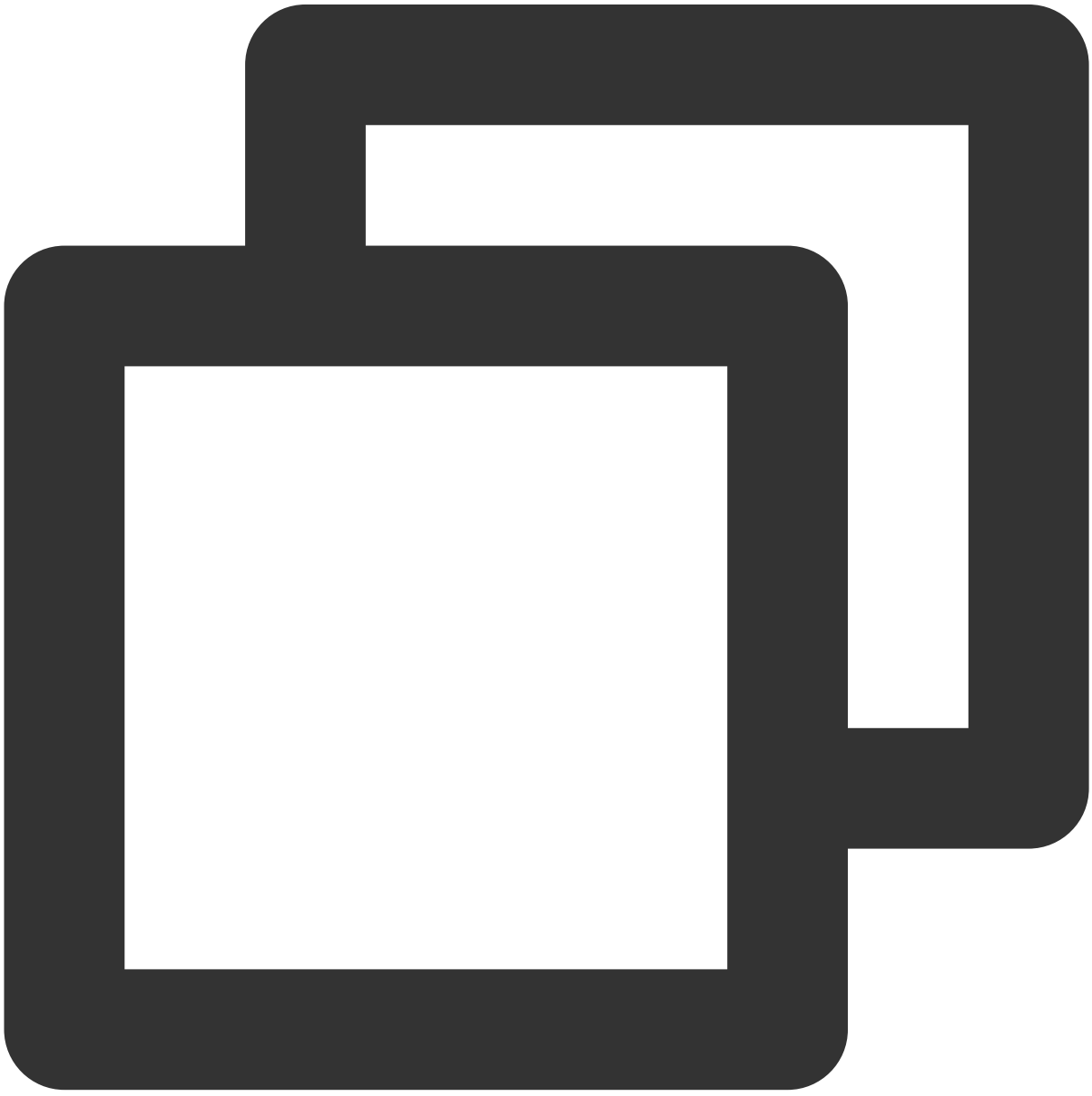
```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw

private lazy var giftPanelView: TUIGiftListView = {
    let view = TUIGiftListView(groupId: liveRoomInfo.roomId.value)
    giftCloudServer.queryBalance { [weak self] error, balance in
        guard let self = self else { return }
        DispatchQueue.main.async {
            if error == .noError {
                view.setBalance(balance)
            } else {
                self.makeToast("query balance error, code = \\(error)")
```

```
                }
            }
        }
        return view
    }()
```

说明：

客户自行实现 `giftCloudServer.queryBalance` 的逻辑，得到礼物余额，通过 `view.setBalance` 更新礼物余额即可。

3、自定义礼物消费逻辑



```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw

func onSendGift(giftListView view: TUIGiftListView, giftModel: TUIGift, giftCount:
    let receiver = TUIGiftUser()
    receiver.userId = liveRoomInfo.anchorInfo.value.userId
    receiver.userName = liveRoomInfo.anchorInfo.value.name.value
    receiver.avatarUrl = liveRoomInfo.anchorInfo.value.avatarUrl.value
    receiver.level = "0"
    giftCloudServer.sendGift(sender: TUILogin.getUserID() ?? "",

                             receiver: receiver.userId,

                             giftModel: giftModel,

                             giftCount: giftCount) { [weak self] error, balance in
        guard let self = self else { return }
        if error == .noError {
            view.sendGift(giftModel: giftModel, giftCount: giftCount, receiver: rec
            view.setBalance(balance)
        } else {
            self.makeToast(.balanceInsufficientText)
        }
    }
}
```

说明：

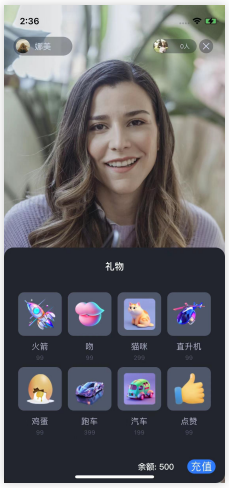
客户自行实现 `giftCloudServer.sendGift` 的逻辑，可以消费礼物的话，则通过 `GiftListView` 的 `sendGift` 发送礼物消息，之后再通过 `setBalance` 更新礼物余额。

Android

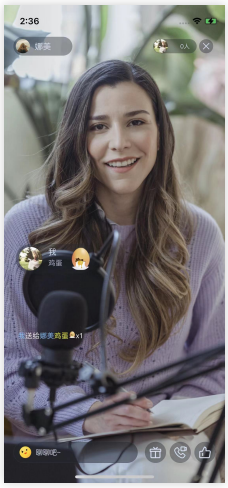
最近更新时间：2024-05-17 11:20:40

互动礼物组件是一款虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。借助这一组件，用户能够向自己欣赏的直播主播送上虚拟礼物，以此展示他们的赞赏、喜爱以及支持。
互动礼物组件支持设置**礼物素材、余额显示、普通礼物播放与全屏礼物播放、充值按钮**等。

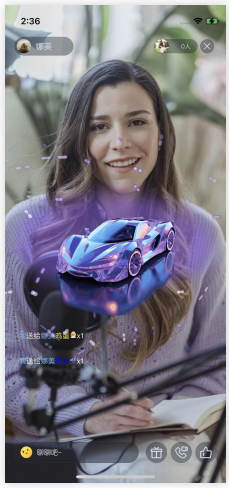
礼物展示



礼物展示面板

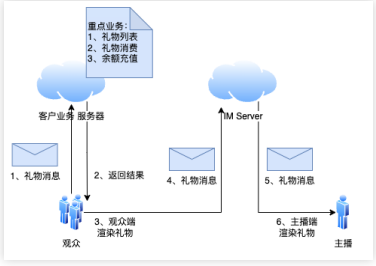


普通礼物播放效果

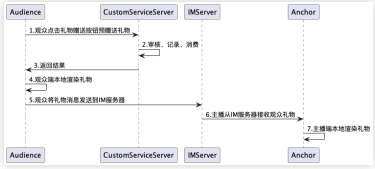


全屏礼物播放效果

礼物系统



礼物系统结构图



礼物系统时序图

快速接入

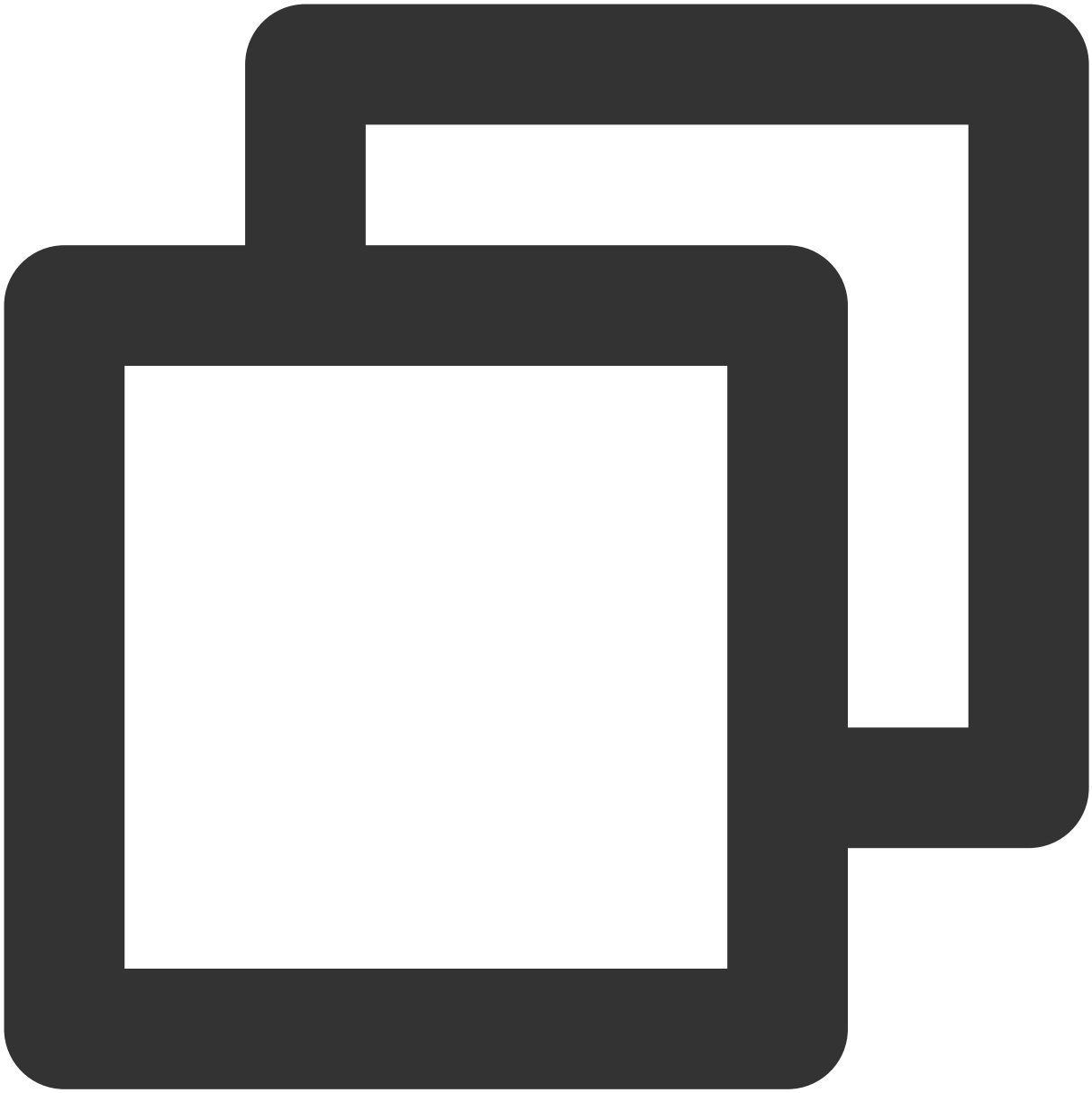
礼物组件主要提供2个 API：

`TUIGiftListView` ：礼物面板，呈现礼物列表，发送礼物及充值。

`TUIGiftPlayView` ：播放礼物的面板，自动监听礼物消息。

设置礼物素材

礼物面板组件 `TUIGiftListView` 提供了 `setGiftList` 接口，可用于设置礼物素材。



```
TUIGiftListView giftListView = new TUIGiftListView(mContext, roomId); //生成礼物面板对
List<GiftModel> giftList = new ArrayList<>() //这里可替换成自己的礼物素材数组
giftListView.setGiftList(giftList) //设置礼物面板的素材
```

说明：

`TUIGift` 的参数及说明如下：

`giftId: String` ： 礼物 ID

`giftName: String` ： 礼物名称

`imageUrl: String` ： 礼物面板展示图像

`animationUrl: String` ： SVGA 动画 Url

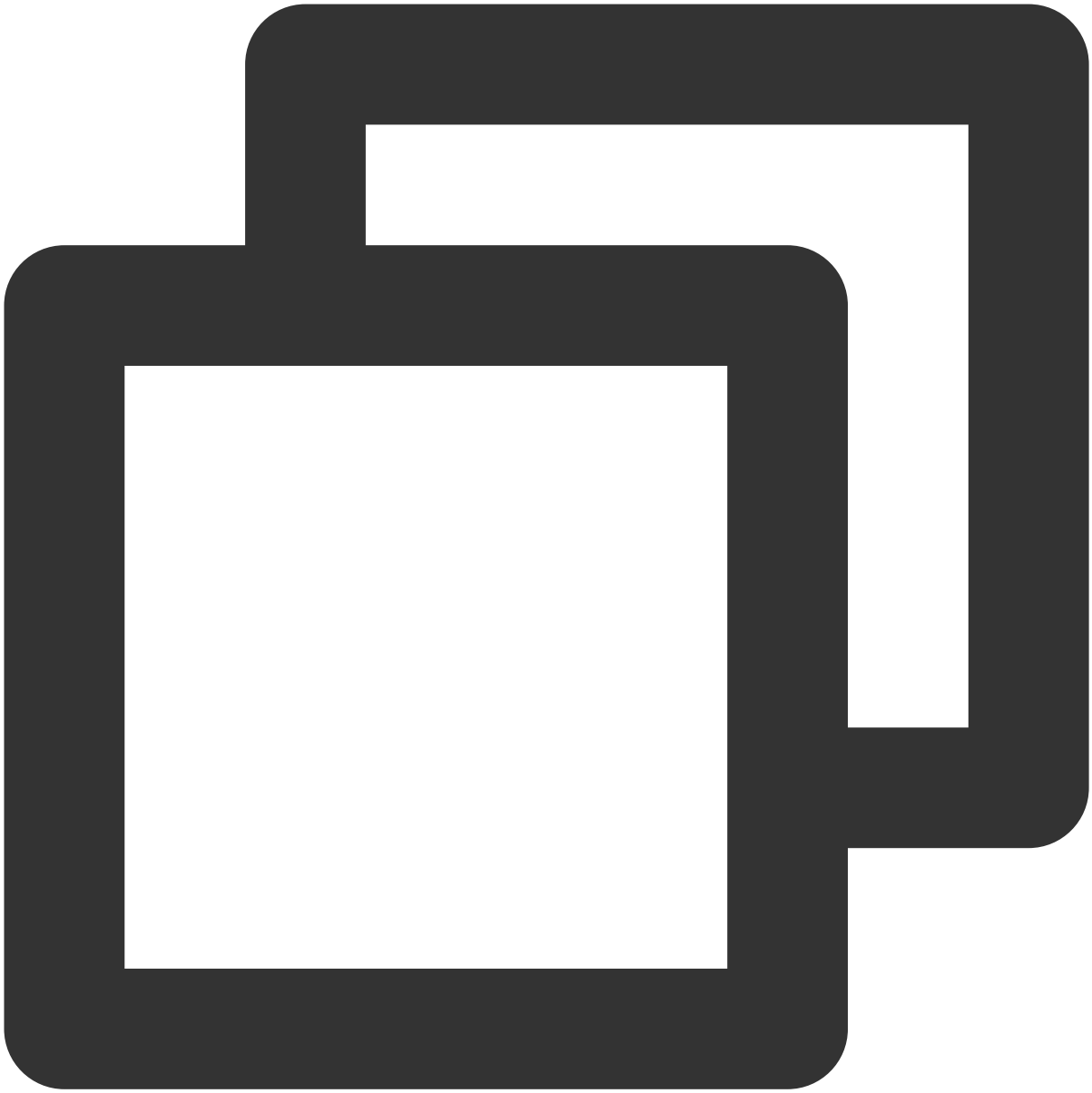
`price: int` ： 礼物价格

`extInfo: <String, Object>` ： 自定义扩展信息

互动礼物组件支持设置自己的礼物素材。若 `animationUrl` 为空，则礼物播放效果为普通播放，播放的内容为 imageUrl 所链接的图像，效果如图2；若 `animationUrl` 不为空，则播放效果为全屏播放，播放的内容为对应的 svga 动画，效果如图3。

赠送礼物

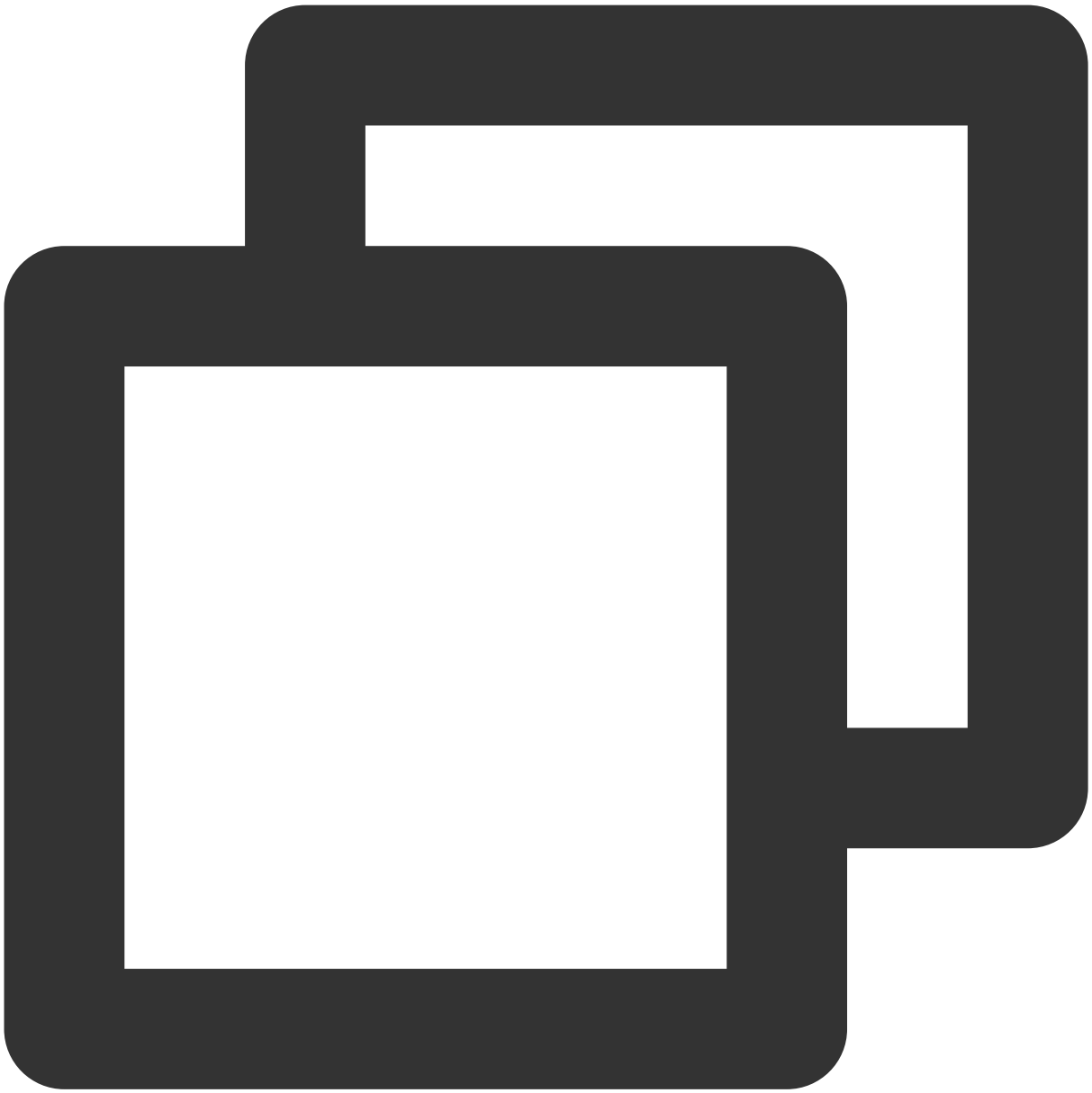
实现 `TUIGiftListView` 的 `OnGiftListener` 中的 `onSendGift` 回调，获取礼物个数和礼物信息，在预处理完后可调用 `TUIGiftListView` 的 `sendGift` 函数用于礼物的实际发送。



```
public void onSendGift(TUIGiftListView view, TUIGift gift, int giftCount) {
    //...此处为预处理，如校验当前用户的余额等操作
    TUIGiftUser receiver = new TUIGiftUser();
    //...此处设置礼物接受者信息
    view.sendGift(gift, giftCount, receiver);
}
```

接收礼物

礼物展示组件 `TUIGiftPlayView` 自身会接收并播放礼物消息。

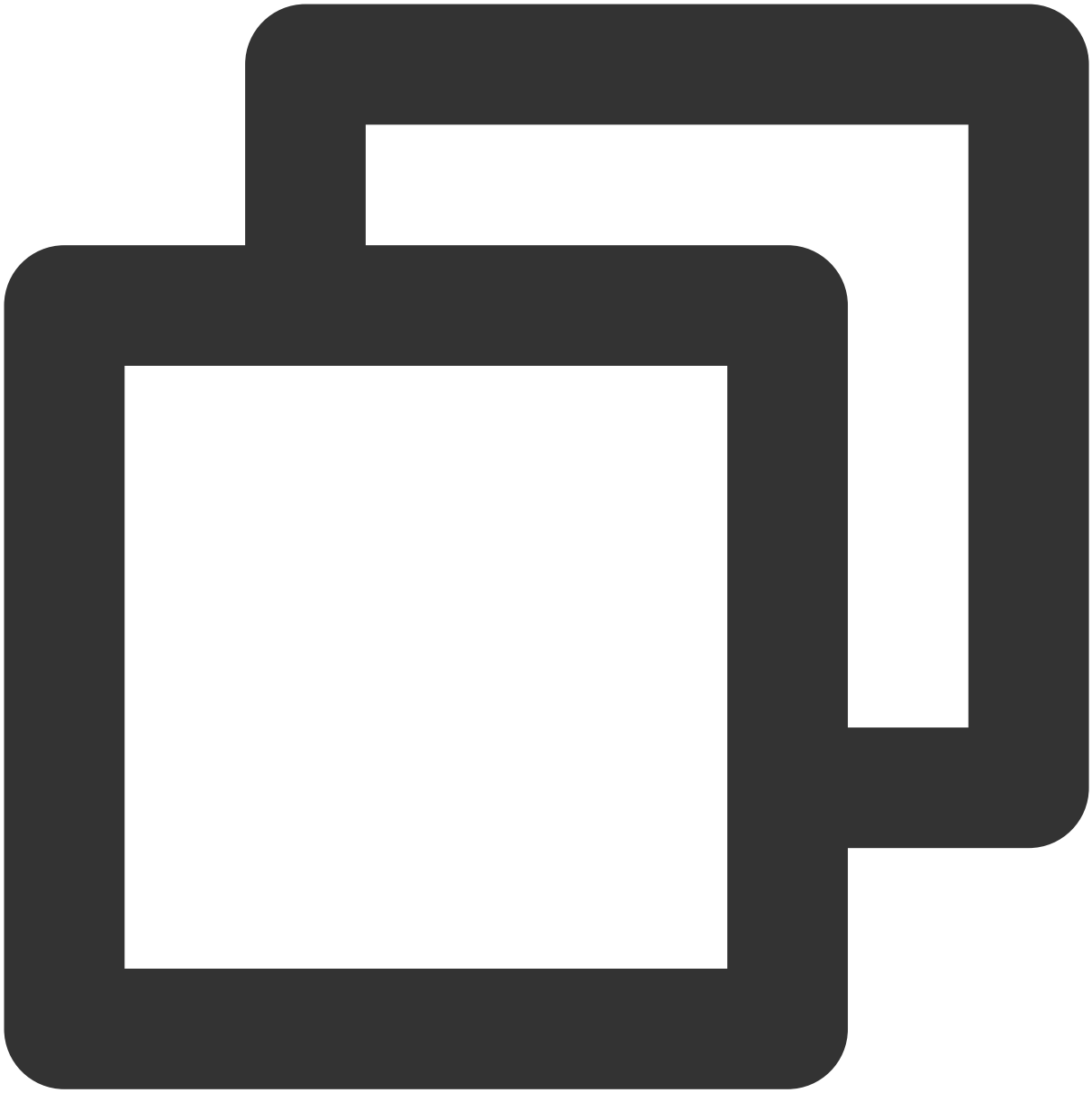


```
TUIGiftPlayView giftPlayView = new TUIGiftPlayView(mContext, roomId);
```

说明：

`TUIGiftPlayView` 需要全屏接入。

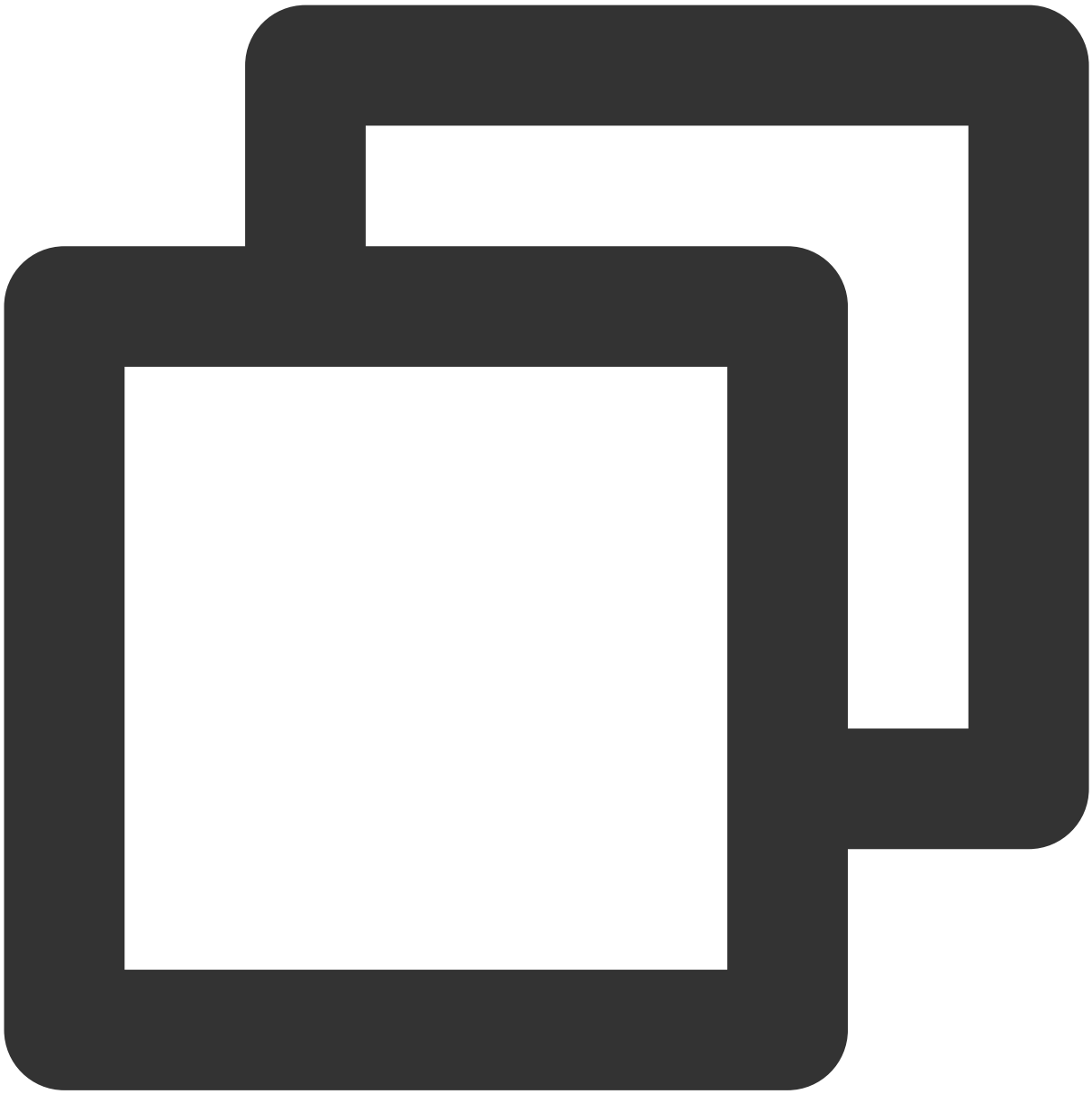
若需要获取接收礼物的回调信息，可将实现 `TUIGiftPlayView` 的 `TUIGiftPlayViewListener` 中的 `onReceiveGift` 函数。



```
public interface TUIGiftPlayViewListener {
    void onReceiveGift(TUIGift gift, TUIGiftUser sender, TUIGiftUser receiver, int
    //...
}
```

播放礼物动画

需要主动调用 `TUIGiftPlayView` 的 `playGiftAnimation` 进行动画播放，调用时机是在收到 `TUIGiftPlayViewListener` 的 `onPlayGiftAnimation` 回调。



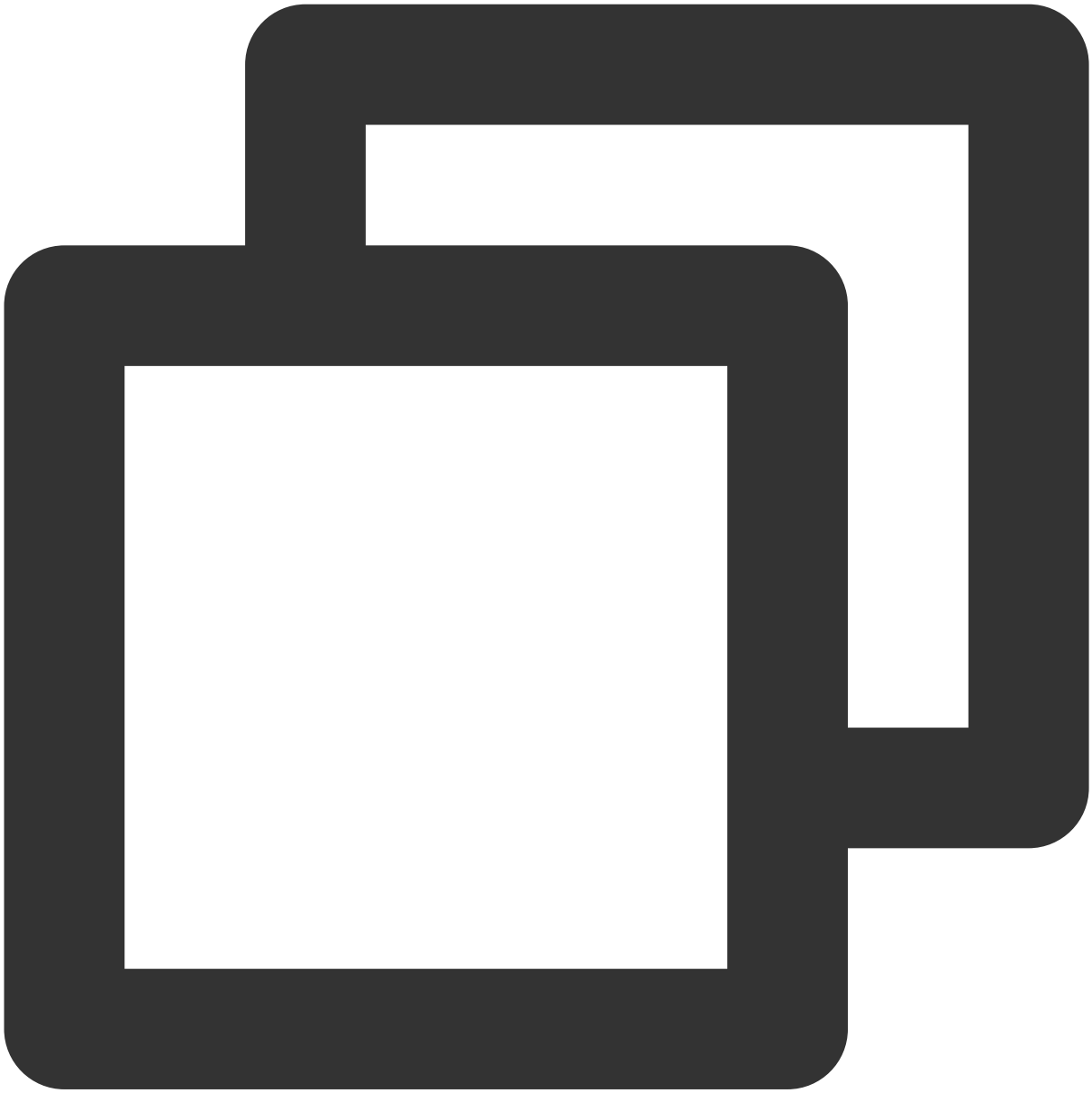
```
public interface TUIGiftPlayViewListener {
    void onPlayGiftAnimation(TUIGiftPlayView view, TUIGift gift);
    //...
}
```

说明：

仅支持 SVGA 动画。

设置余额

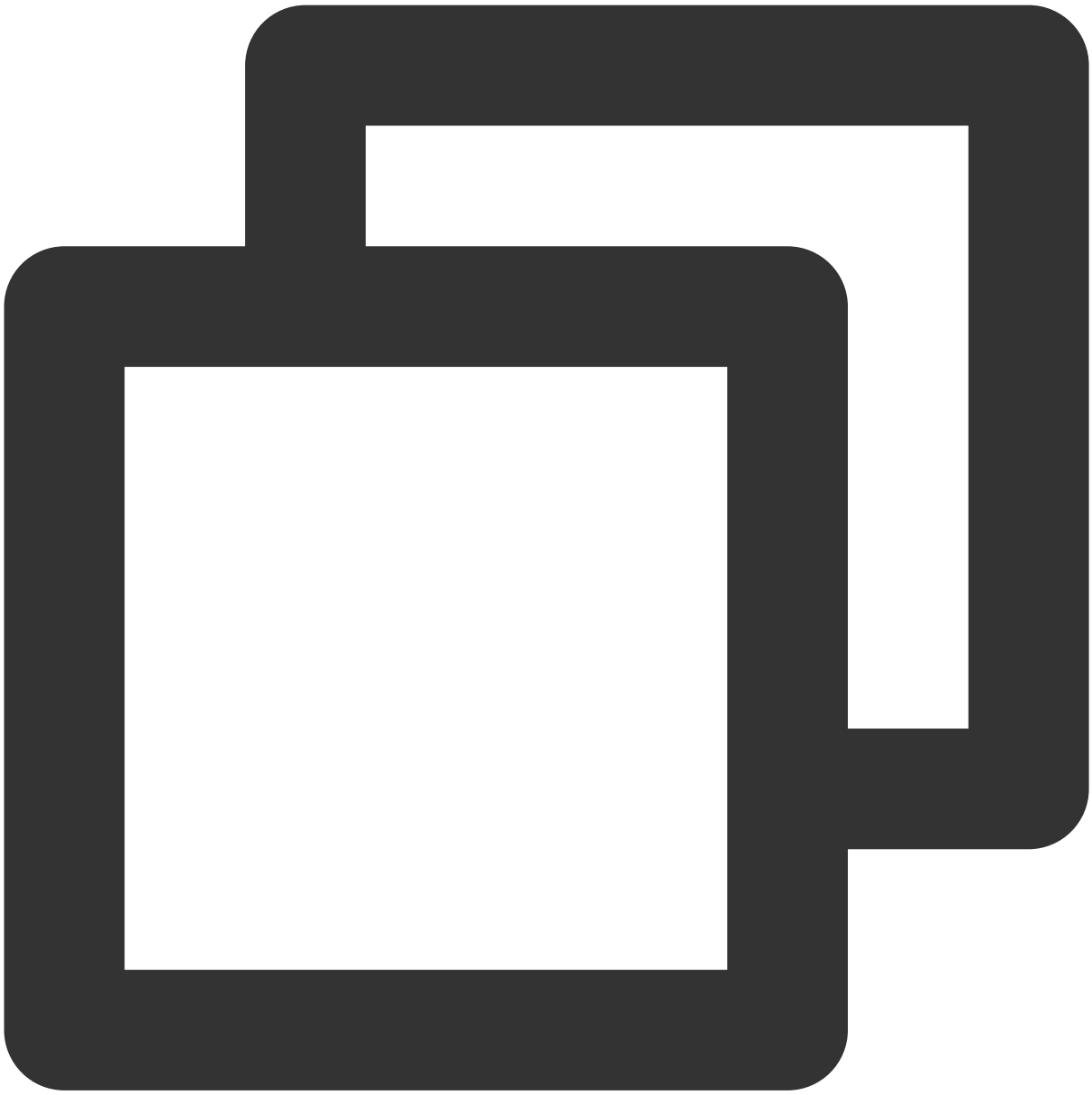
礼物面板组件 `TUIGiftListView` 提供了 `setBalance` 接口，可用于设置礼物面板上显示的余额值。



```
giftListView.setBalance(xxx);
```

充值

实现 `TUIGiftListView` 的 `OnGiftListener` 中的 `onRecharge` 回调可用于接收礼物展示面板抛出的充值按钮点击事件，在这里对接自己的充值系统。



```
public void onRecharge(TUIGiftListView view) {
    //...去充值
    //设置最新的余额
    giftListView.setBalance(balance);
}
```

注意：

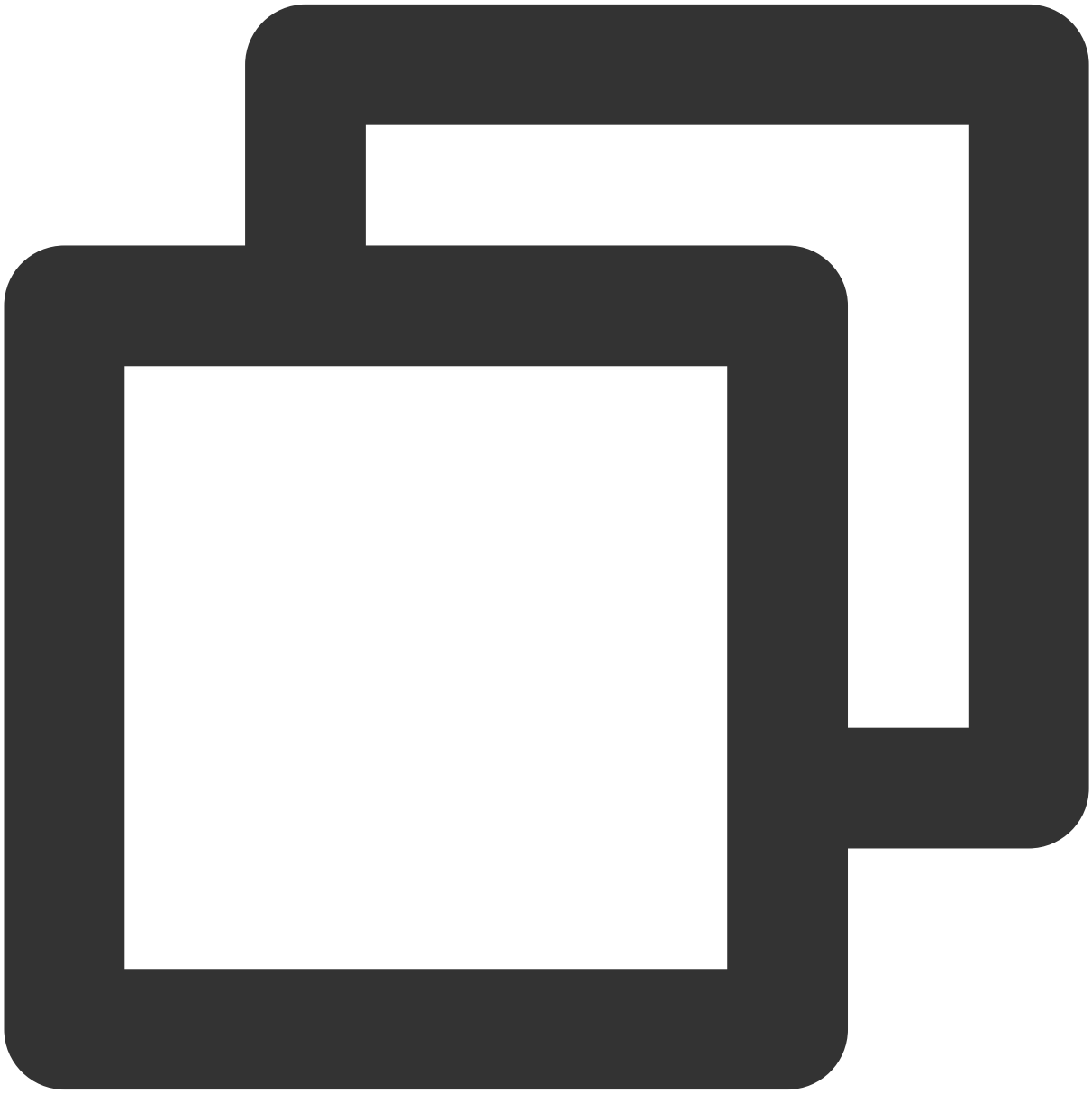
礼物余额是个虚拟币的概念，并不是真实货币。

礼物充值逻辑，由外部实现，客户可以接入自己的充值系统。充值完毕后再更新礼物余额。

修改源码，自定义处理

1、自定义礼物列表

修改观众端礼物面板的礼物列表：



```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c
```

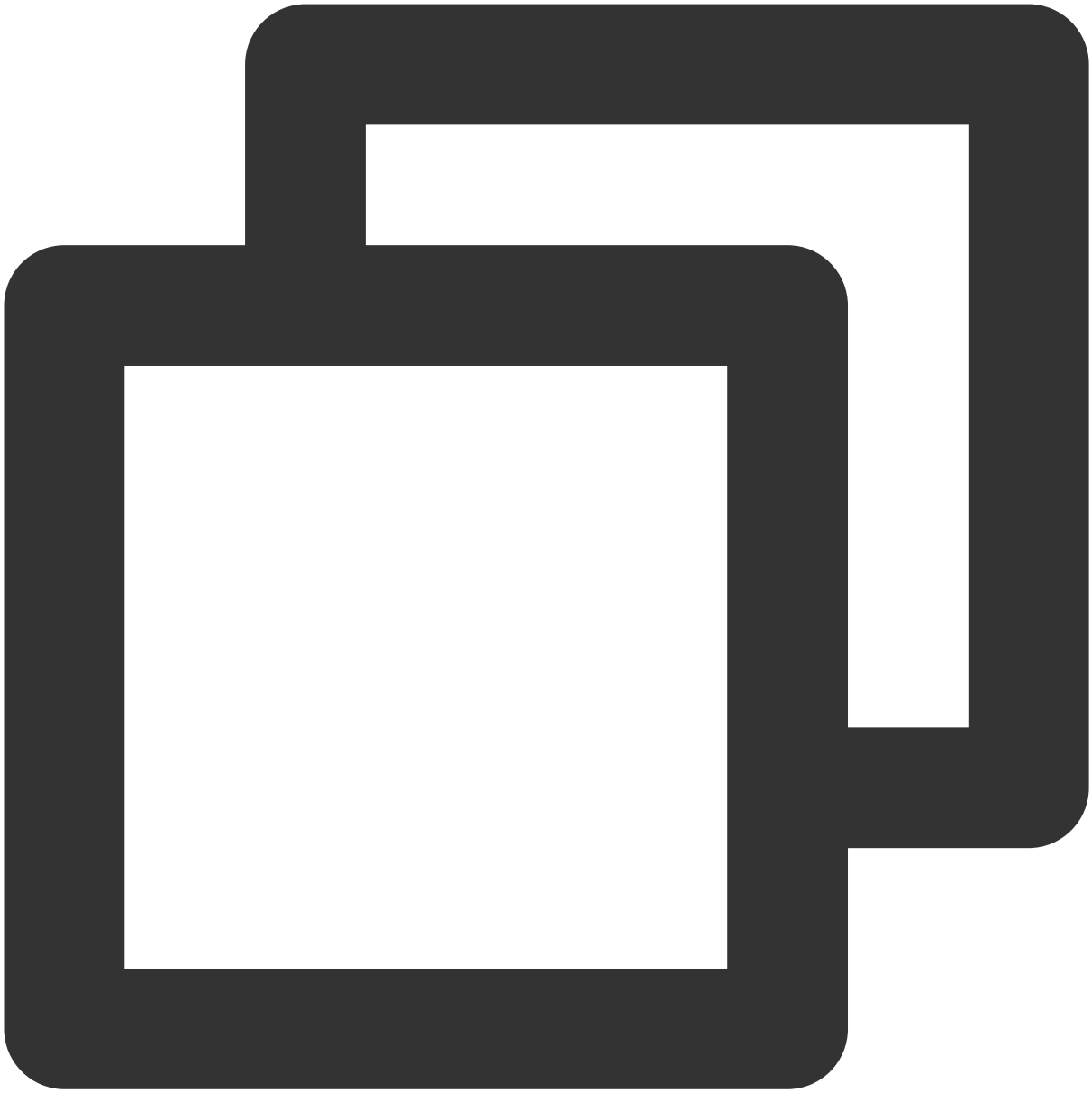
```
mGiftCloudServer.queryGiftInfoList((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setGiftList(result);
    } else {
        ToastUtil.toastLongMessage("query gift list error, code = " + error);
    }
}));
```

说明：

客户自行实现 `mGiftCloudServer.queryGiftInfoList` 的逻辑，得到一个自定义的礼物列表 `List<TUIGift>` ，通过 `GiftListView.setGiftList` 设置礼物列表即可。

礼物的 `animationUrl` 要求是一个 `SVGA` 动画。

2、自定义礼物余额充值

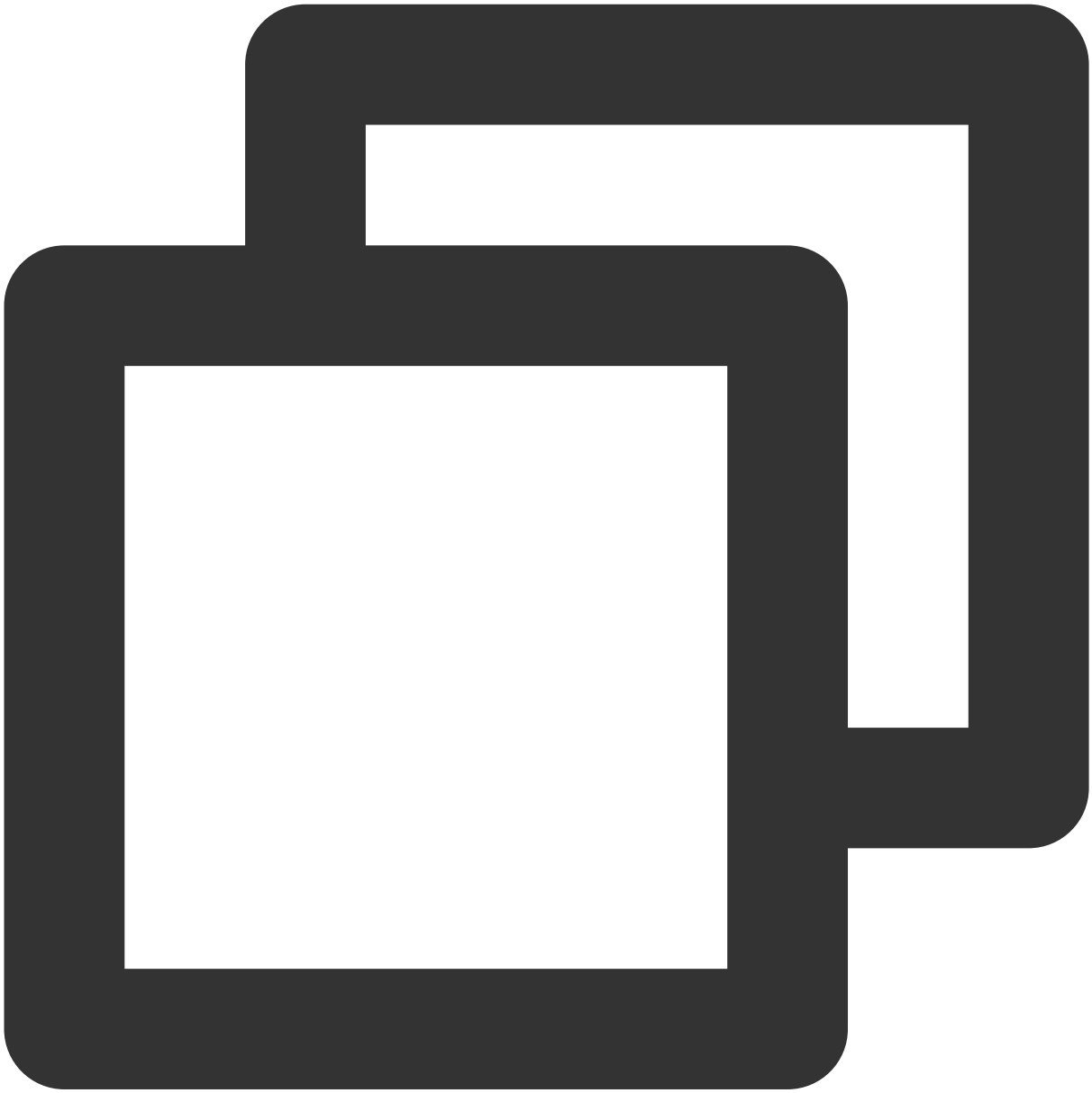


```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c

mGiftCloudServer.queryBalance((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setBalance(result);
    } else {
        ToastUtil.toastLongMessage("query balance error, code = " + error);
    }
}));
```

说明：

客户自行实现 `mGiftCloudServer.queryBalance` 的逻辑，得到礼物余额，通过 `GiftListView.setBalance` 更新礼物余额即可。

3、自定义礼物消费逻辑



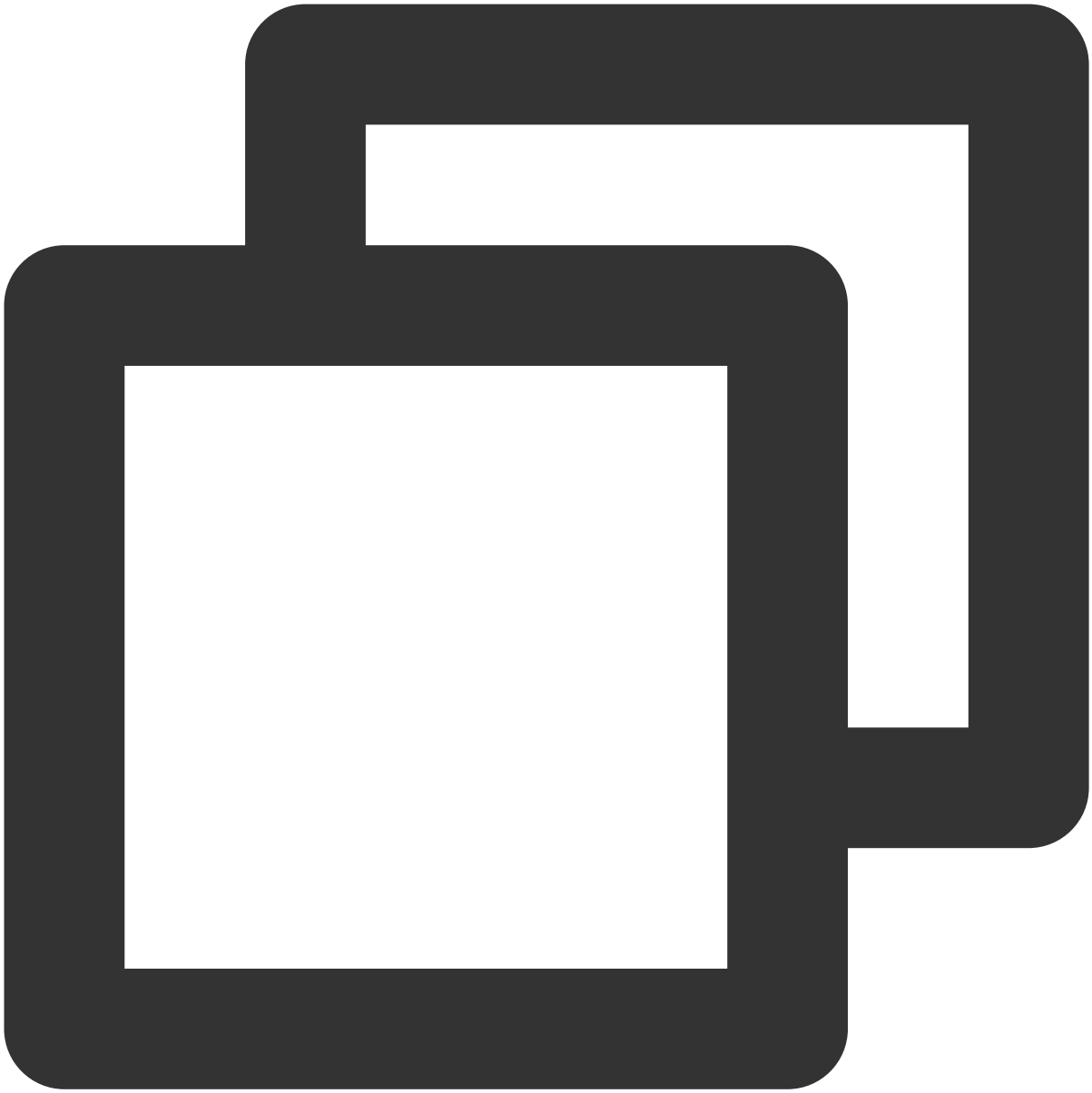
```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c

@Override
public void onSendGift(TUIGiftListView view, TUIGift gift, int giftCount) {
    TUIGiftUser receiver = new TUIGiftUser();
    receiver.userId = mLiveRoomInfo.anchorInfo.userId;
```

```
    receiver.userName = mLiveRoomInfo.anchorInfo.name.get();
    receiver.avatarUrl = mLiveRoomInfo.anchorInfo.avatarUrl.get();
    receiver.level = "0";
    mGiftCloudServer.sendGift(TUILogin.getUserId(), receiver.userId, gift, giftCoun
            (error, result) -> post(() -> {
        if (error == Error.NO_ERROR) {
            view.sendGift(gift, giftCount, receiver);
            view.setBalance(result);
        } else {
            if (error == Error.BALANCE_INSUFFICIENT) {
                String info = getResources().getString(R.string.livekit_gift_balance
                ToastUtil.toastLongMessage(info);
            } else {
                ToastUtil.toastLongMessage("send gift error, code = " + error);
            }
        }
    }));
}
```

说明：

客户自行实现 `mGiftCloudServer.sendGift` 的逻辑，可以消费礼物的话，则通过 `GiftListView` 的 `sendGift` 发送礼物消息，之后再通过 `setBalance` 更新礼物余额。

4、自定义加载礼物动画并播放



```
// 源码路径：
// tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/component
// tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/anchor/component/l

@Override
public void onPlayGiftAnimation(TUIGiftPlayView view, TUIGift gift) {
    mGiftCacheService.request(gift.animationUrl, (error, result) -> {
        if (error == 0) {
            view.playGiftAnimation(result);
        }
```

```
        });
    }
```

说明：

客户自行实现 `mGiftCacheService.request` 的逻辑，加载动画成功得到 `result` （ `InputStream` 类型），然后通过 `TUIGiftPlayView` 的 `playGiftAnimation` 播放礼物动画。

客户端API（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:40

简介

TUILiveKit 是语音聊天室的 UI 开源套件，目前 iOS 平台支持 Swift 语言，通过简单 API 调用即可唤起直播 UI。

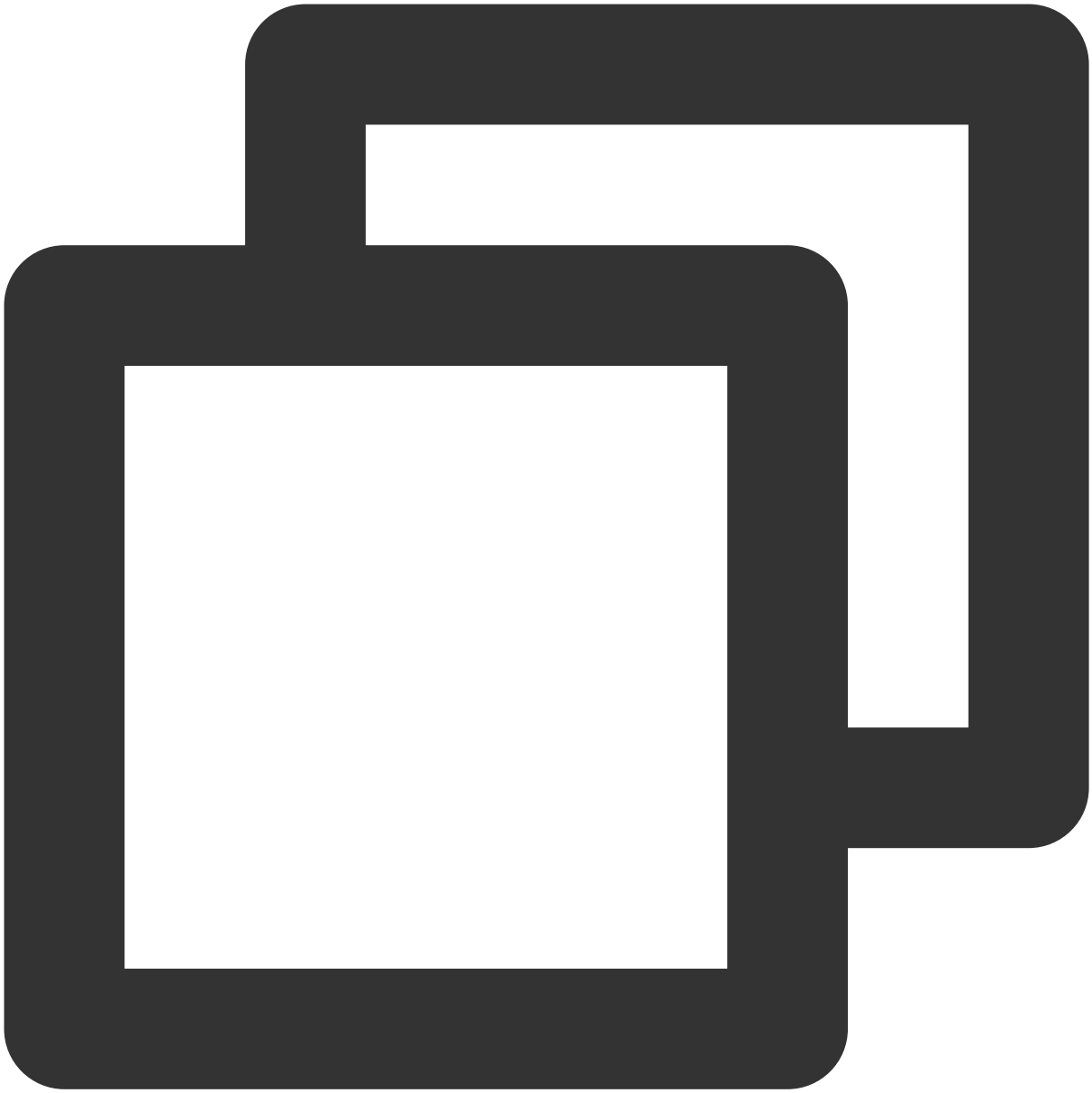
TUIVoiceRoomViewController

API	描述
init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)	构造 语音聊天室控制器 对象

init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)

初始化 TUIVoiceRoomViewController 对象。

版权所有：腾讯云计算（北京）有限责任公司



```
public init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)
```

参数如下表所示：

参数	类型	含义
roomId	String	语音聊天室房间 Id
behavior	RoomBehavior	初始化语音聊天室类型
roomParams	RoomParams	创建语音聊天室必传参数，加入房间（behavior为join）参数可为

		空

RoomBehavior

语音聊天室类型

类型	描述
autoCreate	直接创建语音聊天室
prepareCreate	预览页创建语音聊天室
join	加入语音聊天室

RoomParams

语音聊天室房间参数

类型	描述
maxSeatCount	房间最大麦位数，默认为套餐包支持的最大麦位数量
seatMode	上麦：自由上麦、申请上麦（默认申请上麦）

Android

最近更新时间：2024-05-17 11:20:40

简介

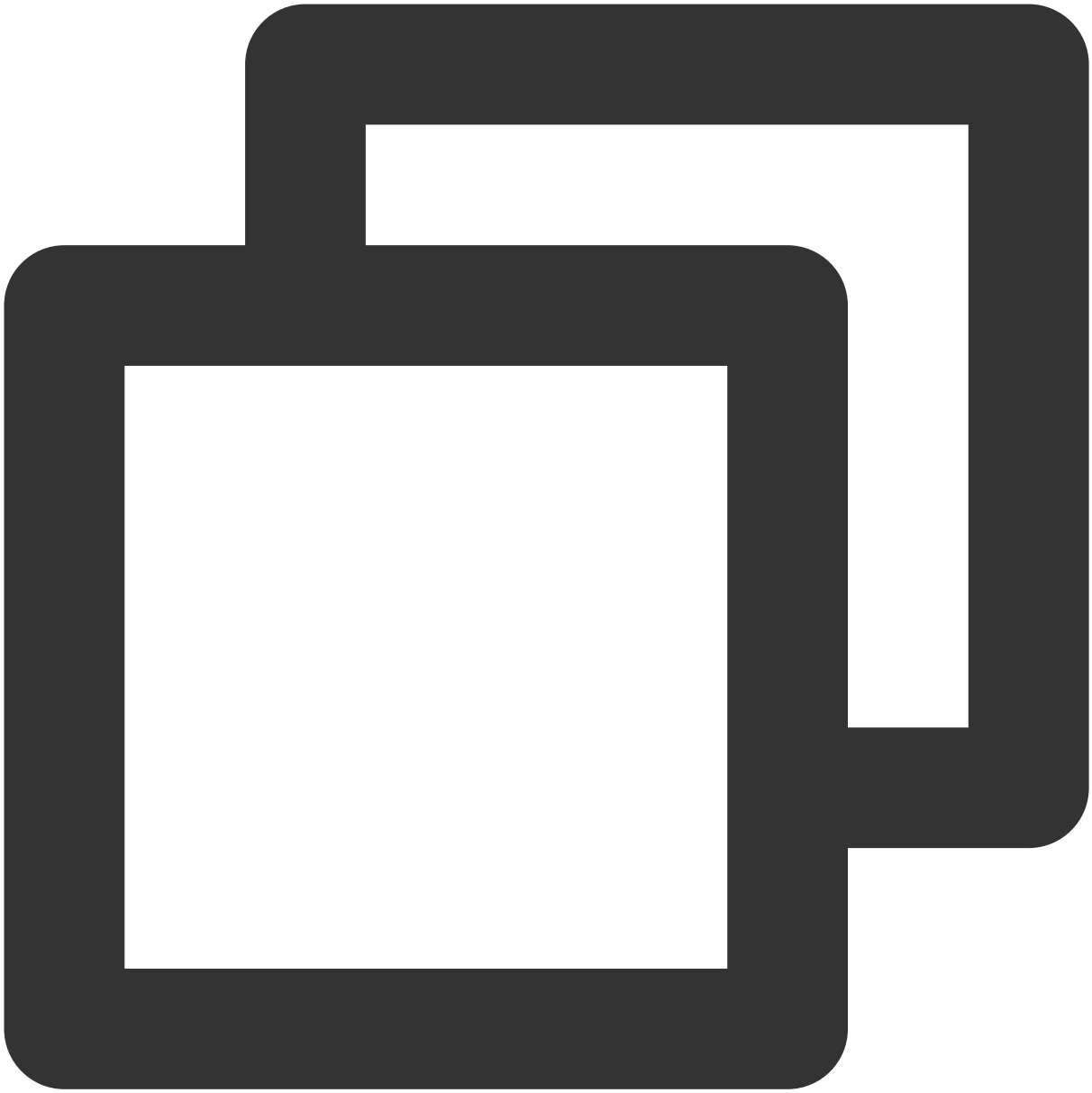
TUILiveKit 是语音聊天室的 UI 开源套件，目前 Android 平台仅支持 Java 语言，通过简单 API 调用即可唤起直播 UI。

TUIVoiceRoomFragment 类

API	描述
TUIVoiceRoomFragment(String roomId, LiveDefine.RoomBehavior behavior, LiveDefine.RoomParams params)	构造语音聊天室主界面对象

TUIVoiceRoomFragment

构造 语音聊天室主界面 对象



```
public TUIVoiceRoomFragment(String roomId, LiveDefine.RoomBehavior behavior, LiveDe
```

参数如下表所示：

参数	类型	含义
roomId	String	语音聊天室房间 Id
behavior	RoomBehavior	初始化语音聊天室类型
params	RoomParams	创建语音聊天室必传参数，加入房间（behavior 为

		JOIN）参数可为空

RoomBehavior

语音聊天室类型

类型	描述
AUTO_CREATE	直接创建语音聊天室
PREPARE_CREATE	预览页创建语音聊天室
JOIN	加入语音聊天室

RoomParams

语音聊天室房间参数

类型	描述
maxSeatCount	房间最大麦位数，默认为套餐包支持的最大麦位数量
seatMode	上麦：自由上麦、申请上麦（默认申请上麦）

错误码（TUILiveKit）

最近更新时间：2024-05-17 11:20:40

客户端错误码

通用错误码

错误码	描述
0	操作成功
-1	暂未归类的通用错误
-2	请求被限频，请稍后重试
-1000	未找到 SDKAppID，请在 腾讯云视立方 SDK 控制台 确认应用信息
-1001	调用 API 时，传入的参数不合法，检查入参是否合法
-1002	未登录,请调用 Login 接口
-1003	获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限
-1004	该功能需要开通额外的套餐，请在 腾讯云视立方 SDK 控制台 按需开通对应套餐

本地用户渲染，视频管理，音频管理 API 回调错误定义

错误码	描述
-1100	系统问题，打开摄像头失败。检查摄像头设备是否正常
-1101	摄像头没有系统授权, 检查系统授权
-1102	摄像头被占用，检查是否有其他进程使用摄像头
-1103	当前无摄像头设备，请插入摄像头设备解决该问题
-1104	系统问题，打开麦克风失败。检查麦克风设备是否正常
-1105	麦克风没有系统授权，检查系统授权
-1106	麦克风被占用
-1107	当前无麦克风设备

-1108	获取屏幕分享对象失败，检查屏幕录制权限
-1109	开启屏幕分享失败，检查房间内是否有人正在屏幕分享

房间管理相关 API 回调错误定义

错误码	描述
-2100	进房时房间不存在，或许已被解散
-2101	需要进房后才可使用此功能
-2102	房主不支持退房操作，Conference（会议）房间类型: 可以先转让房主，再退房。LivingRoom（直播）房间类型: 房主只能解散房间
-2103	当前房间类型下不支持该操作
-2104	当前发言模式下不支持该操作
-2105	创建房间 ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
-2106	房间 ID 已被使用，请选择别的房间 ID
-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
-2108	当前用户已在别的房间内，单个 roomEngine 实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的 roomEngine 实例

房间内用户信息 API 回调错误定义

错误码	描述
-2200	未找到该用户
-2201	房间内未找到该用户

房间内用户发言管理 API 回调错误定义&房间内麦位管理 API 回调错误定义

错误码	描述
-2300	需要房主权限才能操作
-2301	需要房主或者管理员权限才能操作
-2310	信令请求无权限，比如取消非自己发起的邀请。
-2311	信令请求 ID 无效或已经被处理过。

版权所有：腾讯云计算（北京）有限责任公司

-2340	最大麦位超出套餐包数量限制
-2341	当前用户已经在麦位上
-2342	当前麦位已经有人了
-2343	当前麦位被锁
-2344	麦位编号不存在
-2345	当前用户没有在麦上
-2346	上麦人数已满
-2360	当前麦位音频被锁
-2361	需要向房主或管理员申请后打开麦克风
-2370	当前麦位视频被锁, 需要由房主解锁麦位后，才能打开摄像头
-2371	需要向房主或管理员申请后打开摄像头
-2380	当前房间已开启全员禁言
-2381	当前房间内，您已被已禁言

服务端错误码

错误码	描述
83007	请求被限频
84002	房间不存在
84003	房间内人数已超出最大套餐包限制
84004	房间不存在该用户
84005	房间已存在
84006	房主不能退房
85001	麦位已满
85002	麦位被锁定

版权所有：腾讯云计算（北京）有限责任公司

85003	用户已上麦
85004	麦位被使用
85005	麦位号不存在
85006	麦位音频被锁定
85007	麦位视频被锁定
85008	非直播场景不能使用麦位
87001	不允许修改成员身份或房间信息
87002	权限不足，需要房主或管理员身份
87003	IM 群属性更新错误
87005	IM 群属性获取异常
87006	权限不足，需要房主才可以操作
87007	权限不足，需要管理员才可以操作
87008	不允许销毁 IM 房间
41001	创建 IM 群组错误
41002	销毁 IM 群组错误
41003	添加 IM 成员错误
41004	删除 IM 成员错误
41005	IM 改变角色错误
42001	预定会议错误
42002	预定会议无效
42003	预定会议必须是会议场景
42004	预定会议未开始
42005	预定会议已经开始
42006	预定会议邀请成员太多，超出数量限制

版权所有：腾讯云计算（北京）有限责任公司

发布日志（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:41

2024年5月

发布动态	描述	发布时间
Version 1.0.0	支持自定义礼物 支持自定义弹幕 支持音效&混响 支持点赞 支持麦控	2024.05.17

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2024-05-17 11:20:40

2024年4月

发布动态	描述	发布时间
Version 1.0.0	支持自定义礼物 支持自定义弹幕 支持音效&混响 支持点赞 支持麦控	2024.05.17

常见问题（TUILiveKit）

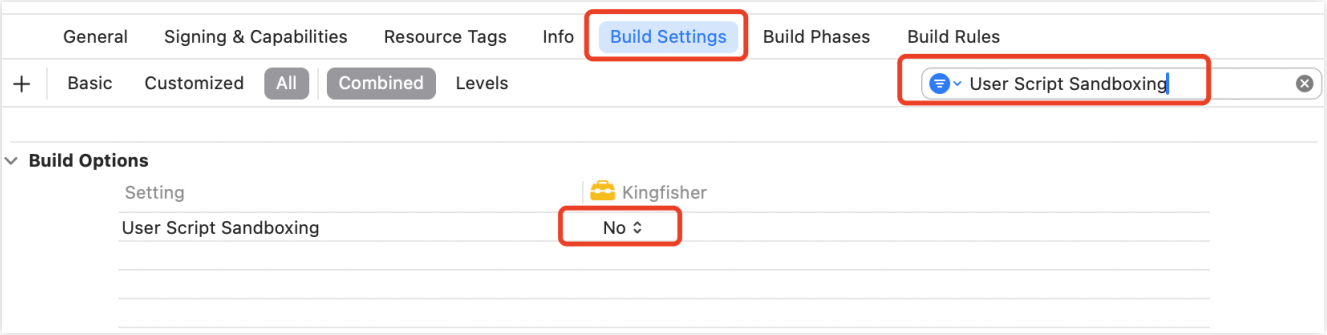
iOS

最近更新时间：2024-05-17 11:48:33

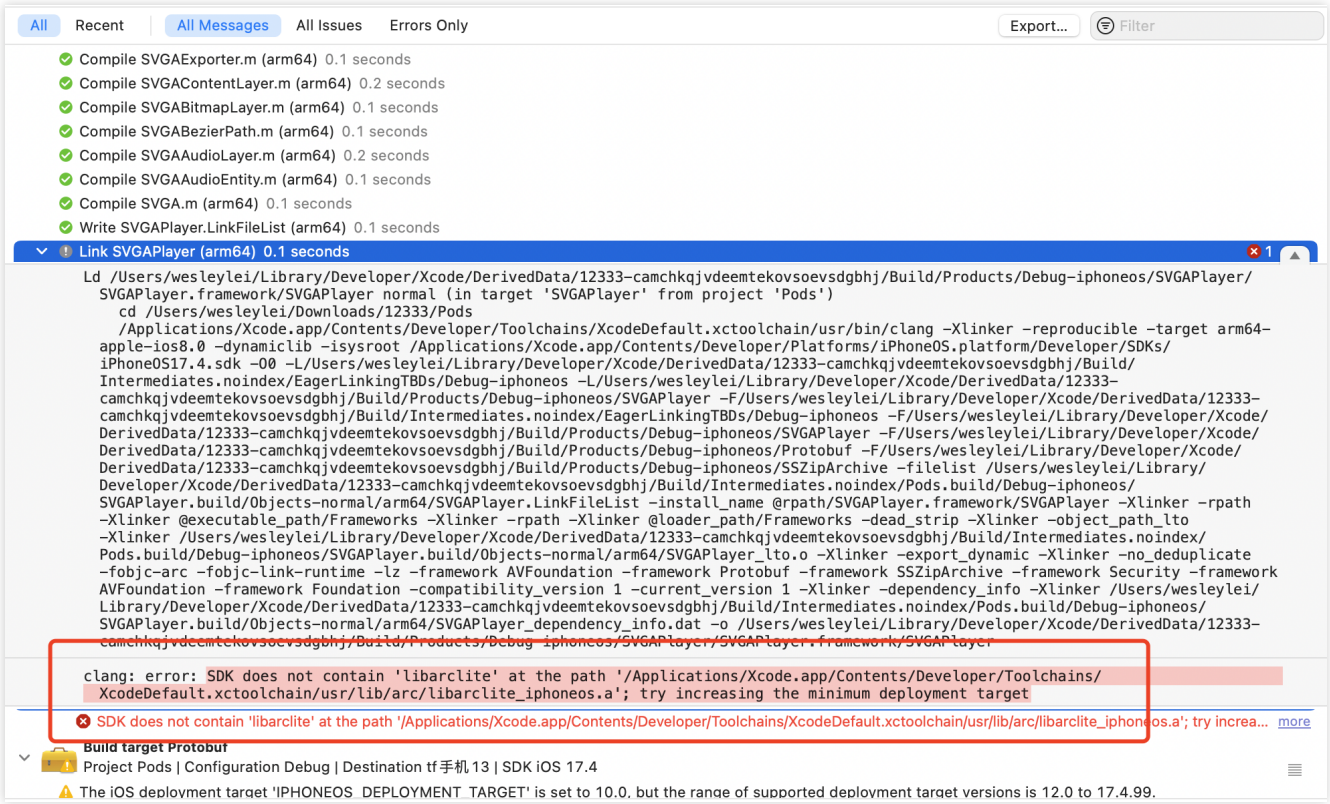
Xcode 15 编译报错？

1. 出现 **Sandbox: rsync**，编译报错截图：

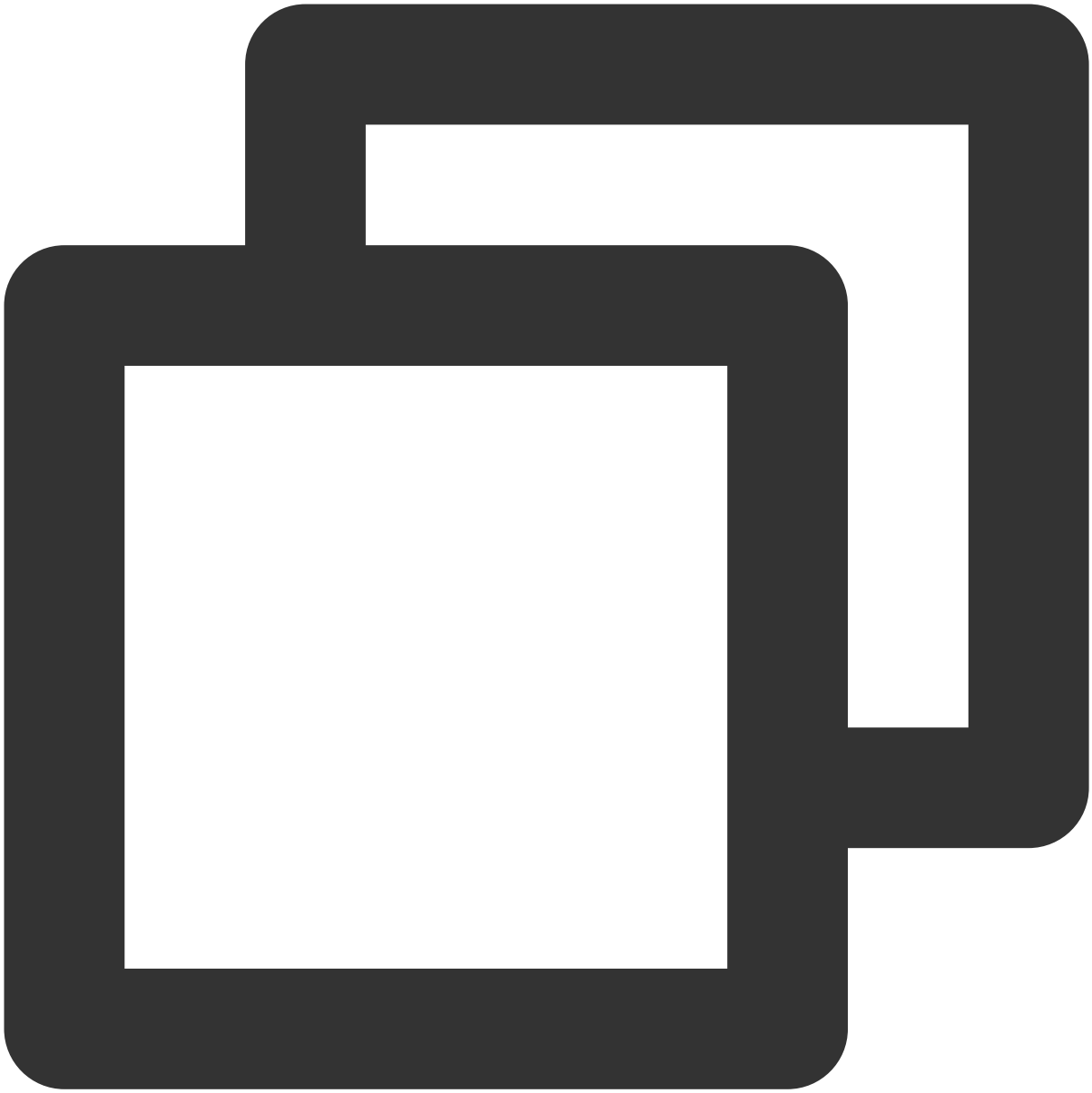
可以在 **Build Settings** 中把 **User Script Sandboxing** 设置为 **NO**：



2. 如果出现 **SDK does not contain**，编译报错截图：

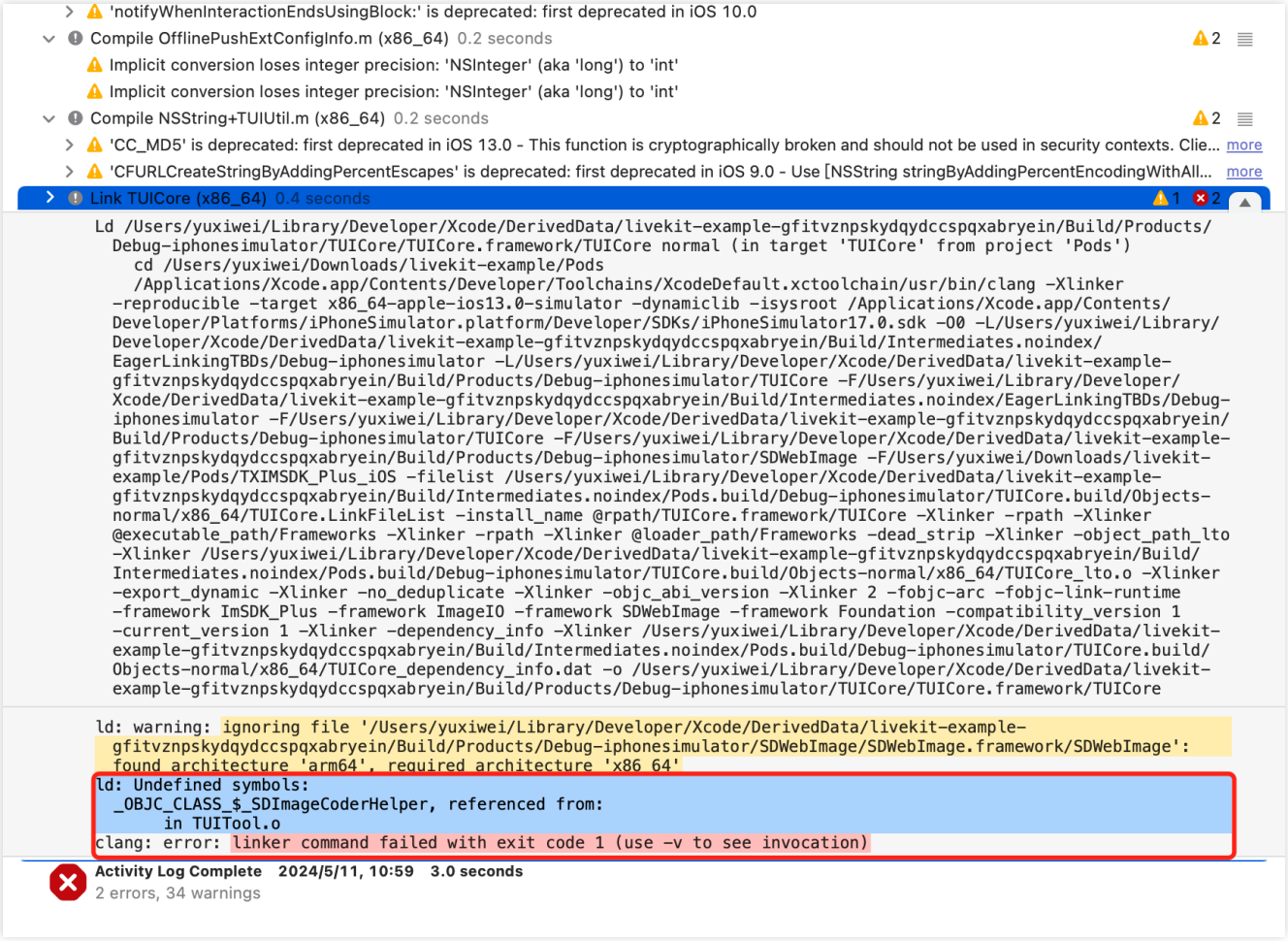


请在 Podfile 添加如下代码：

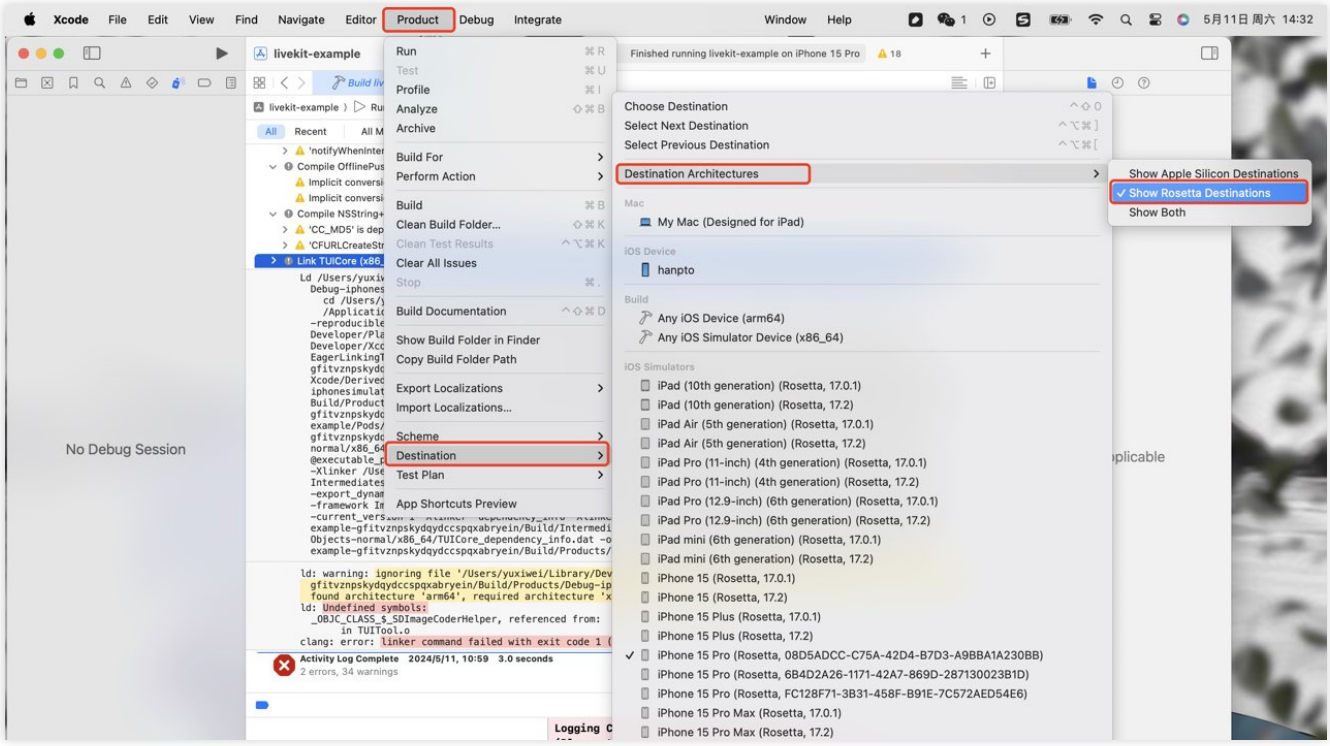


```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '13.0'
    end
  end
end
```

3. 如果在 **M 系列电脑**上运行模拟器，可能会出现 **Linker command failed with exit code 1 (use -v to see invocation)**，编译报错截图：



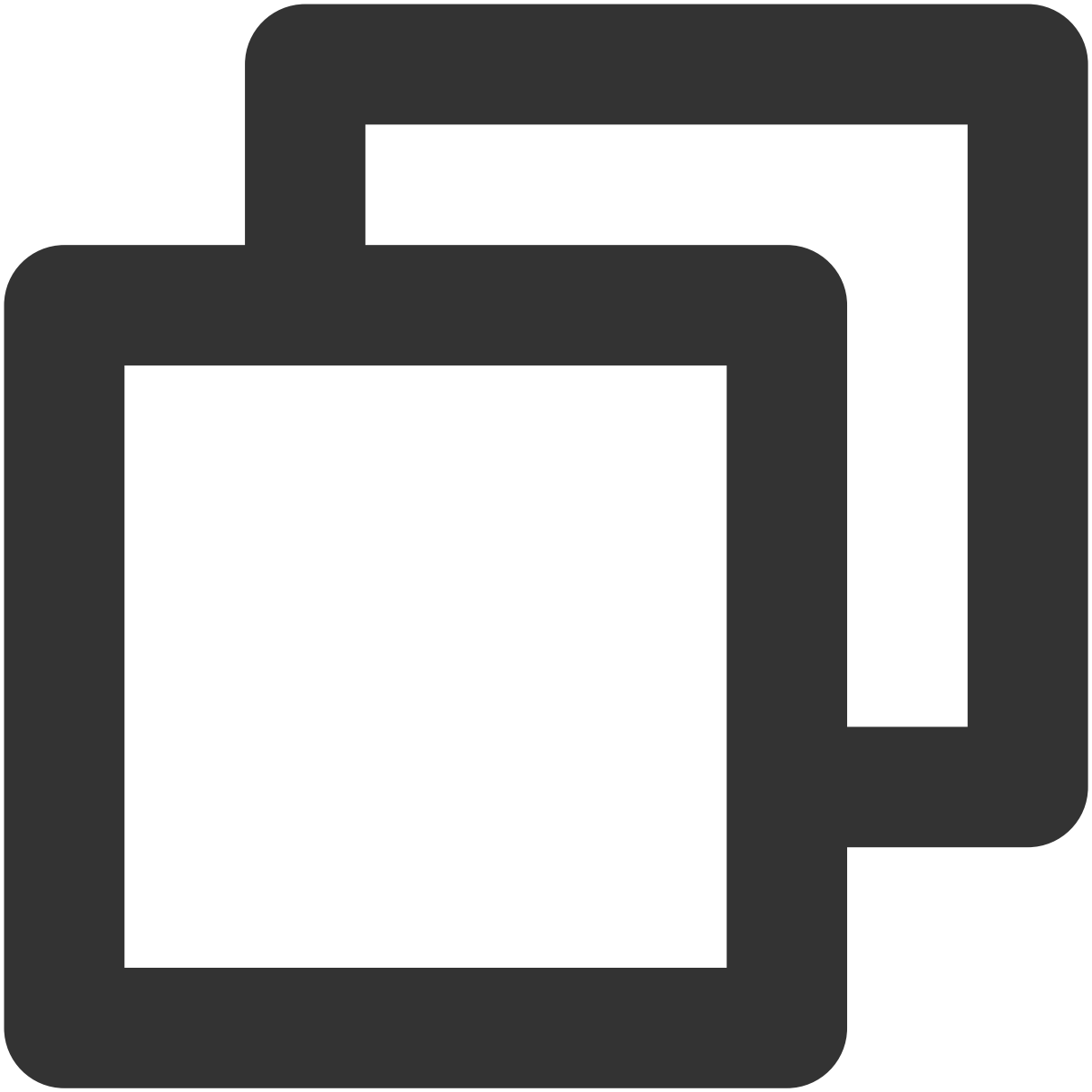
需要修改 xcode 配置。xcode 打开项目 > Product > Destination > **Destination Architectures** 可以选择用哪种模式的模拟器打开，需要选择 （**Rosetta**）结尾的模拟器。



TUILiveKit 和自己集成的音视频库冲突了？

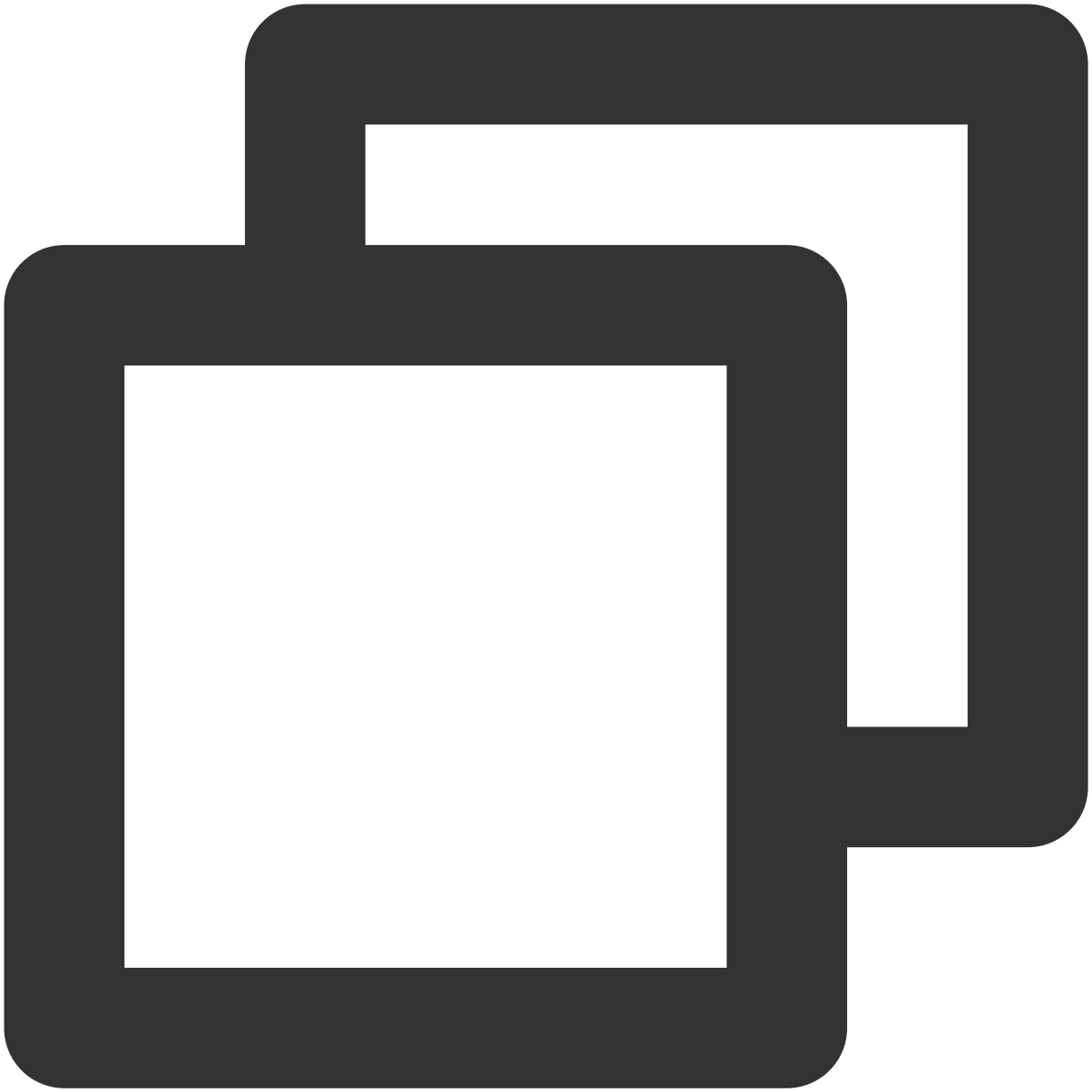
腾讯云的 `音视频库` 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。

1. 如果您使用了 `TXLiteAVSDK_TRTC` 库，不会发生符号冲突。可直接在 `Podfile` 文件中添加依赖，



```
pod 'TUILiveKit'
```

2. 如果您使用了 `TXLiteAVSDK_Professional` 库，会产生符号冲突。您可在 `Podfile` 文件中添加依赖，



```
pod 'TUILiveKit/Professional'
```

3. 如果您使用了 `TXLiteAVSDK_Enterprise` 库，会产生符号冲突。建议升级到 `TXLiteAVSDK_Professional` 后使用 `TUILiveKit/Professional` 。

如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 .xlog。日志是否加密是可以通过 setLogCompressEnabled 来控制，生成的文件名里面含 C(compressed) 的就是加密压缩的，含 R(raw) 的就是明文的。

iOS： `sandbox的Documents/log`

说明：

查看 .xlog 文件需要下载 解密工具，在 Python 2.7环境中放到 xlog 文件同目录下直接使用 `python decode_mars_log_file.py` 运行即可。

查看 .clog 文件（9.6 版本以后新的日志格式）需要下载 解密工具，在 Python 2.7 环境中放到 clog 文件同目录下直接使用 `python decompress_clog.py` 运行即可。

Android

最近更新时间：2024-05-17 11:48:33

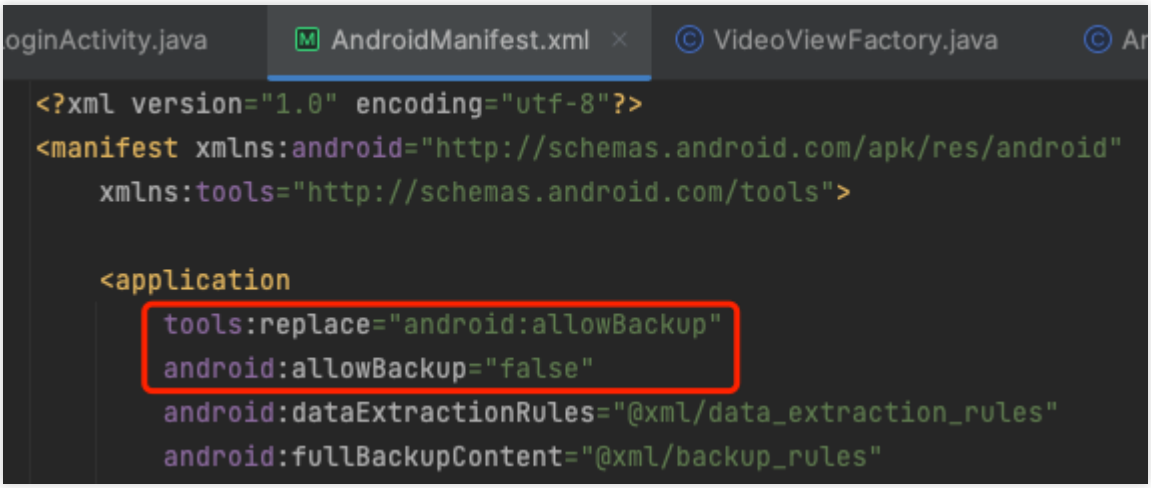
TUILiveKit 是否可以不引入 IM SDK，只使用 TRTC？

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，例如创建房间信令、连麦信令等核心逻辑都使用 IM服务，如果您已经购买有其他 IM 产品，也可以参照 TUILiveKit 逻辑进行适配。

运行异常：allowBackup 异常，如何处理？

```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-3
    is also present at [com.github.yyued:SVGAPlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(t
    Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.x
```

问题原因：多个模块的 `AndroidManifest.xml` 中都配置了 `allowBackup` 属性，造成冲突。
解决方法：您可以在您工程的 `AndroidManifest.xml` 文件中删除 allowBackup 属性或将该属性改为 false，表示关闭备份和恢复功能；并在 `AndroidManifest.xml` 文件的 application 节点中添加 `tools:replace="android:allowBackup"` 表示覆盖其他模块的设置，使用您自己的设置。修复示例如图所示：



Activity 主题问题：Activity need to use a Theme.AppCompat theme，如何处理？

```
FATAL EXCEPTION: main
Process: com.trtc.uikit.livekit.example, PID: 15190
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.trtc.uikit.livekit.example/com.trtc.u
 .LoginActivity}: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) w
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3730)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3885)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:101)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2332)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:230)
    at android.app.ActivityThread.main(ActivityThread.java:8115) <1 internal line>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:526)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1034)
```

问题原因：由于 `LoginActivity` 继承自 `AppCompatActivity` ，所以要给 `LoginActivity` 设置一个 `Theme.AppCompat` 主题。

解决方法：您可以在您工程的 `AndroidManifest.xml` 文件中 `LoginActivity` 的配置里，增加一个 `Theme.AppCompat` 主题。您也可以使用自己的 `Theme.AppCompat` 主题。修复示例如图所示：

```
<activity
    android:name=".login.LoginActivity"
    android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
</activity>
```

集成 TUIVoiceRoom (Android)

最近更新时间：2022-09-06 14:19:19

组件介绍

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

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

组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

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

- 在 `setting.gradle` 中完成导入，参考如下：

```
include ':Source'
```

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

```
api project(':Source')
```

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

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

步骤二：配置权限及混淆规则

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

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

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

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

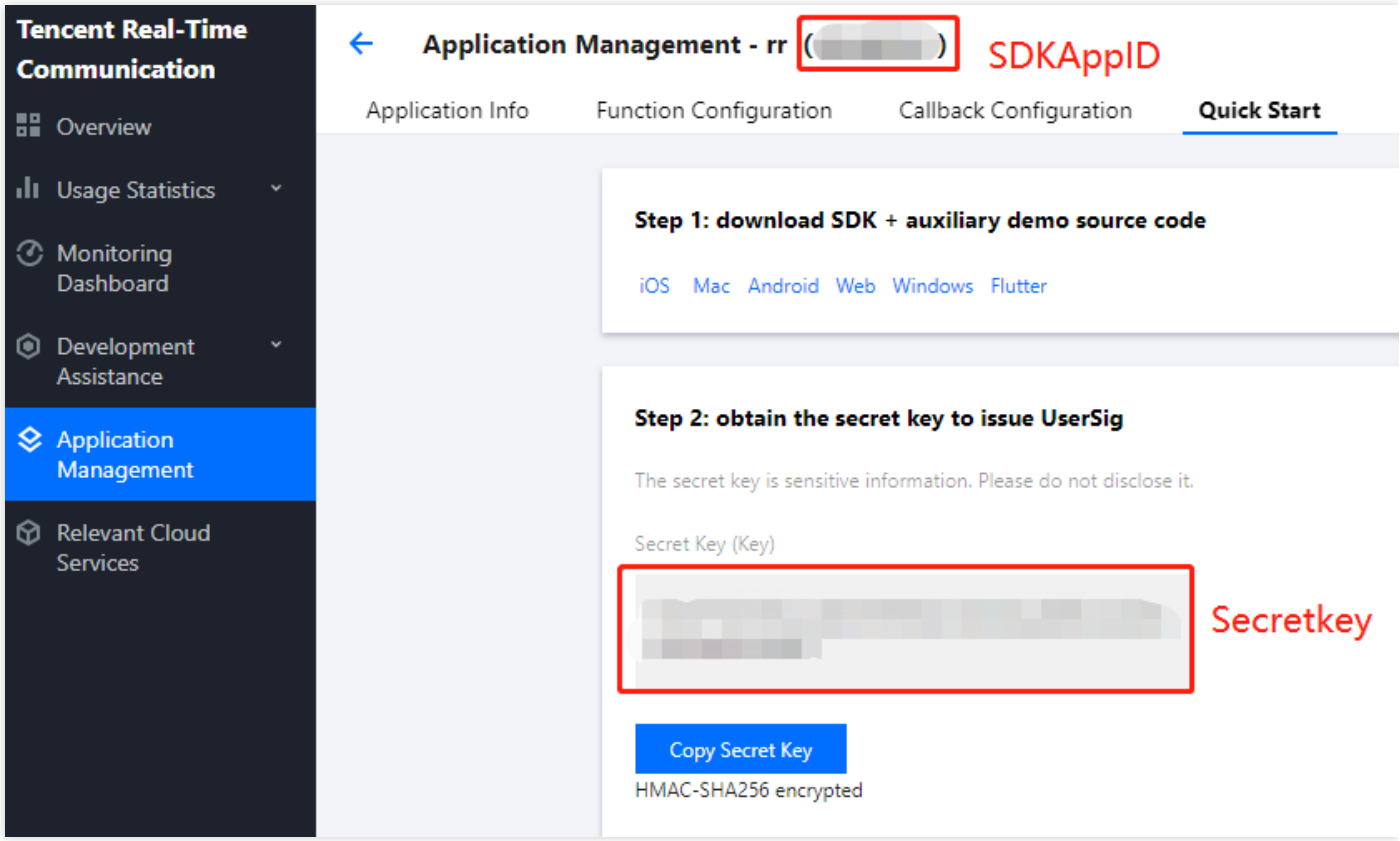
步骤三：初始化并登录

```
// 1.初始化,
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
);
// 2.登录,
mTRTCVoiceRoom.login(SDKAppID, userId, userSig, new TRTCVoiceRoomCallback.ActionC
```

```
allback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//登录成功
}
}
});
```

参数说明：

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



- **Secretkey**：**TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，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：实现语音聊天房间

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

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

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

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

3. 实现听众主动上麦 TRTCVoiceRoom#enterSeat

```
// 1: 听众调用上麦
int seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex, new TRTCVoiceRoomCallback.ActionCallback()
{
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//操作成功
}
}
});
// 2.收到 onSeatListChange 回调，刷新您的麦位列表
```

```
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoLis
t) {
}
```

4. 实现房主抱人上麦 TRTCVoiceRoom#pickSeat

```
// 1: 房主调用抱人麦位
int seatIndex = 2; //麦位的index
String userId = "123"; //需要上麦的用户id
mTRTCVoiceRoom.pickSeat(1, userId, new TRTCVoiceRoomCallback.ActionCallback() {
@Override
public void onCallback(int code, String msg) {
if (code == 0) {
//操作成功
}
}
});
// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoLis
t) {
}
```

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

```
// 听众端视角
// 1.听众调用申请上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("takeSeat", userId, seatIndex,
null);
// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
if(id.equals(inviteId)) {
mTRTCVoiceRoom.enterSeat(index, null);
}
}
// 房主端视角
// 1.房主收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd,
final String content) {
```

```
if (cmd.equals("takeSeat")) {
// 2.房主同意听众请求
mTRTCVoiceRoom.acceptInvitation(id, null);
}
}
```

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

```
// 房主端视角
// 1.房主调用请求抱人上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("pickSeat", userId, seatIndex,
null);
// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
if(id.equals(inviteId)) {
mTRTCVoiceRoom.pickSeat(index, null);
}
}
// 听众端视角
// 1.听众收到请求
@Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd,
final String content) {
if (cmd.equals("pickSeat")) {
// 2.听众同意房主请求
mTRTCVoiceRoom.acceptInvitation(id, null);
}
}
```

7. 实现文字聊天 TRTCVoiceRoom#sendRoomTextMsg

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

8. 实现弹幕消息 TRTCVoiceRoom#sendRoomCustomMsg

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
@Override
public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.Us
erInfo userInfo) {
if ("CMD_DANMU".equals(cmd)) {
// 收到弹幕消息
Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
} else if ("CMD_LIKE".equals(cmd)) {
// 收到点赞消息
Log.d(TAG, userInfo.userName + "给您点了个赞！");
}
}
});
```

常见问题

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

集成 TUIVoiceRoom (iOS)

最近更新时间：2022-09-06 14:15:51

组件介绍

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

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

组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

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

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

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

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

```
pod install
```

步骤二：配置权限及混淆规则

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

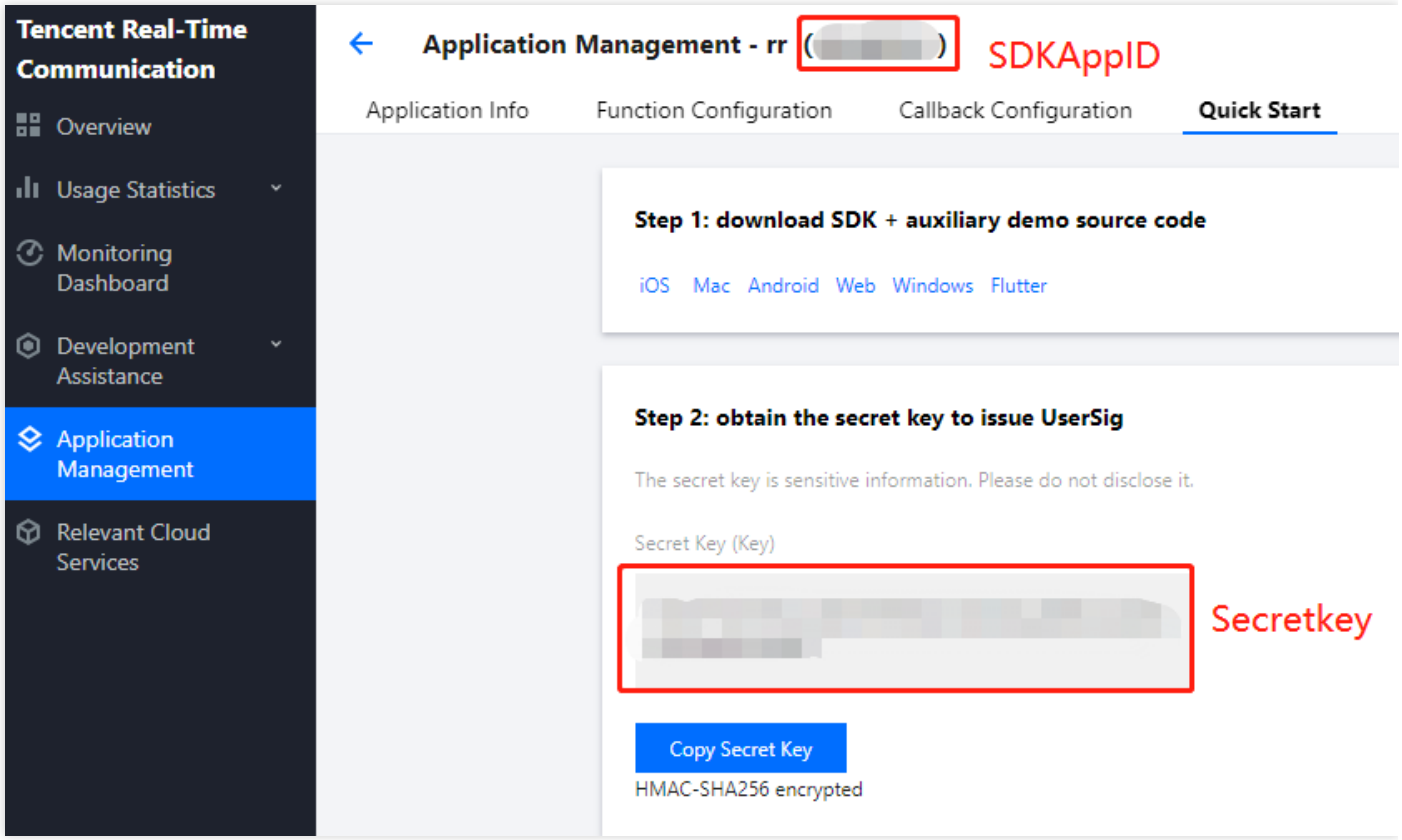
```
<key>NSMicrophoneUsageDescription</key>
<string>VoiceRoomApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```

步骤三：初始化并登录

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

参数说明：

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



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

步骤四：实现语音聊天房间

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

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

```
// 房主端创建房间
mTRTCVoiceRoom.createRoom(roomID: yourRoomID, roomParam: roomParam) { (code, me
ssage) in
if code == 0 {
// 创建成功
}
}
```

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

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

3. 实现听众主动上麦 TRTCVoiceRoom#enterSeat

```
// 1: 听众调用上麦
let seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex: 2) { (code, message) in
if code == 0 {
// 上麦成功
}
}
// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
// 刷新的麦位列表
}
```

4. 实现房主抱人上麦 TRTCVoiceRoom#pickSeat

```
// 1: 房主调用抱人麦位
let seatIndex = 2; //麦位的index
let userId = "123"; //需要上麦的用户id
mTRTCVoiceRoom.pickSeat(seatIndex: 1, userId: "123") { (code, message) in
if code == 0 {
}
}
```

版权所有：腾讯云计算（北京）有限责任公司

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

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

```
// 听众端视角
// 1.听众调用申请上麦
let seatIndex = "1"; //麦位的index
let userId = "123"; //用户id
let inviteId = mTRTCVoiceRoom.sendInvitation(cmd: "takeSeat", userId: ownerUser
Id, content: "1") { (code, message) in
// 发送结果回调
}
// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
self.mTRTCVoiceRoom.enterSeat(seatIndex: ) { (code, message) in
// 上麦结果回调
}
}
}
// 房主端视角
// 1.房主收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, c
ontent: String) {
if cmd == "takeSeat" {
// 2.房主同意听众请求
self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

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

```
// 房主端视角
// 1.房主调用 sendInvitation，请求抱听众“123”上2号麦
let inviteId = self.mTRTCVoiceRoom.sendInvitation(cmd: "pickSeat", userId: owne
rUserId, content: "2") { (code, message) in
// 发送结果回调
}
// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
if identifier == selfID {
```

```
self.mTRTCVoiceRoom.pickSeat(seatIndex: ) { (code, message) in
// 上麦结果回调
}
}
}
// 听众端视角
// 1.听众收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, c
ontent: String) {
if cmd == "pickSeat" {
// 2.听众同意房主请求
self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
}
}
```

7. 实现文字聊天 TRTCVoiceRoom#sendRoomTextMsg

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

}
// 接收端：监听文本消息
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
//收到的message信息处理方法
}
```

8. 实现弹幕消息 TRTCVoiceRoom#sendRoomCustomMsg

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息, "CMD_LIKE"表示点赞消息
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: “CMD_DANMU”, message: "hello world",
callback: nil)
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: n
il)
// 接收端：监听自定义消息
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserI
nfo) {
if cmd == "CMD_DANMU" {
// 收到弹幕消息
}
if cmd == "CMD_LIKE" {
// 收到点赞消息
}
}
```

常见问题

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

版权所有：腾讯云计算（北京）有限责任公司

TUIVoiceRoom API 查询
TRTCVoiceRoom (iOS)

最近更新时间：2022-07-22 15:13:44

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

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

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

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

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

TRTCVoiceRoom API 概览

SDK 基础函数

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

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

房间相关接口函数

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

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位 (麦上主播端可调用) 。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

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

消息发送相关接口

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

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

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

房间事件回调

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

麦位变更回调

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

版权所有：腾讯云计算（北京）有限责任公司

听众进出事件回调

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

消息事件回调

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

信令事件回调

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

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

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

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

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

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

setDelegate

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

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

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

setDelegateQueue

设置事件回调所在的线程队列，默认发送动主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
```

```
* - parameter queue 队列，TRTCVoiceRoom 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
userId:(NSString *)userId
userSig:(NSString *)userSig
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback:));
```

参数如下表所示：

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

logout

登出。

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

参数如下表所示：

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

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

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

房间相关接口函数

createRoom

创建房间（房主调用）。

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

参数如下表所示：

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

房主开播的正常调用流程如下：

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

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

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

参数如下表所示：

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

enterRoom

进入房间（听众调用）。

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

参数如下表所示：

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

听众进房收听的正常调用流程如下：

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

版权所有：腾讯云计算（北京）有限责任公司

exitRoom

退出房间。

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

参数如下表所示：

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

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(VoiceRoomInfoCallback _Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

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

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(VoiceRoomUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userIdList	NSArray<NSString>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

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

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

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

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

moveSeat

移动麦位 (麦上主播端可调用)。

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

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

参数如下表所示：

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

返回值：

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

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

leaveSeat

主动下麦（主播调用）。

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

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

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

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

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

参数如下表所示：

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

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

kickSeat

踢人下麦（房主调用）。

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

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

参数如下表所示：

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

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

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

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

参数如下表所示：

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

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

closeSeat

封禁/解禁某个麦位（房主调用）。

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

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

参数如下表所示：

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

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

本地音频操作接口

startMicrophone

开启麦克风采集。

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

stopMicrophone

停止麦克风采集。

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

setAudioQuality

设置音质。

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

参数如下表所示：

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

muteLocalAudio

静音/取消静音本地的音频。

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

参数如下表所示：

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

setSpeaker

设置开启扬声器。

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

参数如下表所示：

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

setAudioCaptureVolume

设置麦克风采集音量。

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

参数如下表所示：

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

setAudioPlayoutVolume

设置播放音量。

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

参数如下表所示：

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

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	指定的用户 ID。
mute	BOOL	YES：开启静音；NO：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	YES：开启静音；NO：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enable:));
```

参数如下表所示：

参数	类型	含义
enable	BOOL	YES：开启耳返；NO：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

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

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

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

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
userId:(NSString *)userId
content:(NSString *)content
```

```
callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendInvitation(cmd:user
Id:content:callback:));
```

参数如下表所示：

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

返回值：

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

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullabl
e)callback NS_SWIFT_NAME(acceptInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullabl
e)callback NS_SWIFT_NAME(rejectInvitation(id:callback:));
```

参数如下表所示：

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(cancelInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

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

```
- (void)onError:(int)code
message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义

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

onWarning

警告回调。

```
- (void)onWarning:(int)code
message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

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

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	NSString	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)roomId
NS_SWIFT_NAME(onRoomDestroy(roomId:));
```

参数如下表所示：

参数	类型	含义
roomId	NSString	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(VoiceRoomInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
roomInfo	VoiceRoomInfo	房间信息。

onUserMicrophoneMute

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

```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
mute	BOOL	YES：静音麦位； NO：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:
(NSInteger)totalVolume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义

参数	类型	含义
userVolumes	NSArray	用户列表。
totalVolume	NSInteger	音量大小，取值：0 - 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	NSArray<VoiceRoomSeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

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

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
user:(VoiceRoomUserInfo *)user
```

版权所有：腾讯云计算（北京）有限责任公司

```
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

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

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	操作的麦位。
isMute	BOOL	YES：静音麦位； NO：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

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

听众进出事件回调

onAudienceEnter

收到听众进房通知。

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

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

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

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

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

参数如下表所示：

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

onRecvRoomCustomMsg

收到自定义消息。

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

参数如下表所示：

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

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

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

参数如下表所示：

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

onInviteeAccepted

被邀请者接受邀请。

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

参数如下表所示：

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

onInviteeRejected

被邀请者拒绝邀请。

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

参数如下表所示：

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

onInvitationCancelled

邀请人取消邀请。

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

参数如下表所示：

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

TRTCVoiceRoom (Android)

最近更新时间：2022-07-22 15:13:44

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

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

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

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

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

TRTCVoiceRoom API 概览

SDK 基础函数

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

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

房间相关接口函数

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

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位（麦上主播端可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

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

消息发送相关接口

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

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

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

房间事件回调

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

麦位变更回调

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

版权所有：腾讯云计算（北京）有限责任公司

听众进出事件回调

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

消息事件回调

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

信令事件回调

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

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

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

参数如下表所示：

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

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

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

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

setDelegate

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

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

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

setDelegateHandler

设置事件回调所在的线程。

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

参数如下表所示：

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

login

登录。

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

参数如下表所示：

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

logout

登出。

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

参数如下表所示：

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

setSelfProfile

修改个人信息。

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

参数如下表所示：

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

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

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

房主开播的正常调用流程如下：

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

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

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

参数如下表所示：

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

enterRoom

进入房间（听众调用）。

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

参数如下表所示：

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

听众进房收听的正常调用流程如下：

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

exitRoom

退出房间。

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

参数如下表所示：

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

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> 说明：
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCVoiceRoomCallback.RoomInfoCallback callback);
```

参数如下表所示：

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

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCVoiceRoomCallback.UserListCallback userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

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

```
public abstract void enterSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

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

moveSeat

移动麦位 (麦上主播端可调用)。

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

```
public abstract int moveSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

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

返回值：

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

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

leaveSeat

主动下麦（主播调用）。

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

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

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

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

```
public abstract void pickSeat(int seatIndex, String userId, TRTCVoiceRoomCallbac
k.ActionCallback callback);
```

参数如下表所示：

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

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

kickSeat

踢人下麦（房主调用）。

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

```
public abstract void kickSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

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

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

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

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

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

closeSeat

封禁/解禁某个麦位（房主调用）。

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

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

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

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

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

muteLocalAudio

静音/取消静音本地的音频。

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

参数如下表所示：

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

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

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

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

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

muteRemoteAudio

静音/解除静音指定成员。

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

参数如下表所示：

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

muteAllRemoteAudio

静音/解除静音所有成员。

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

参数如下表所示：

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

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager。

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

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCVoiceRoomCallback.Action
Callback callback);
```

参数如下表所示：

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

sendRoomCustomMsg

发送自定义文本消息。

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

参数如下表所示：

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

邀请信令相关接口

sendInvitation

向用户发送邀请。

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

参数如下表所示：

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

返回值：

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

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCall
back callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

版权所有：腾讯云计算（北京）有限责任公司

```
public abstract void rejectInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

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

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

参数如下表所示：

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

onWarning

警告回调。

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

参数如下表所示：

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

onDebugLog

Log 回调。

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

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

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

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCVoiceRoomDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

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

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

参数如下表所示：

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

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

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

参数如下表所示：

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

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

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

参数如下表所示：

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

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

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

参数如下表所示：

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

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

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

参数如下表所示：

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

onSeatMute

房主禁麦。

版权所有：腾讯云计算（北京）有限责任公司

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCVoiceRoomDef.UserInfo userInfo);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo us
erInfo);
```

参数如下表所示：

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

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

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

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

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

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

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

版权所有：腾讯云计算（北京）有限责任公司

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
</trtcclouddef.trtcvolumeinfo>		