即时通信 IM

视频通话（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默示的承诺或保证。

文档目录

- 视频通话（含 UI）
  - 组件介绍（TUICallKit）
  - 开通服务（TUICallKit）
  - 快速接入（TUICallKit）
    - Android
    - iOS
    - Web&H5
    - Flutter
  - 界面定制（TUICallKit）
    - Android
    - Web
    - Flutter
  - 更多特性（TUICallKit）
    - 群组通话
      - Android&iOS&Flutter
      - Web&H5
    - 设置昵称、头像（全平台）
    - 设置分辨率和填充模式
    - 悬浮窗
      - Android&iOS&Flutter
      - Web&H5
    - 美颜特效（TUICallKit）
      - Flutter
    - 自定义铃声
      - Android
      - iOS
      - Web&H5
      - Flutter
    - 监听通话状态
      - Android&iOS&Flutter
      - Web&H5
  - 客户端 API（TUICallKit）
    - Android
      - API 概览
      - TUICallKit

- TUICallEngine
- TUICallObserver
- 类型定义
- 错误码表
- iOS
  - API 概览
  - TUICallKit
  - TUICallEngine
  - TUICallObserver
  - 类型定义
  - 错误码表
- Web
  - API 概览
  - TUICallKit
  - TUICallEngine
  - TUICallEvent
- Flutter
  - API 概览
  - TUICallKit
  - TUICallEngine
  - TUICallObserver
  - 类型定义
  - 错误码表
- 发布日志（TUICallKit）
  - Web(Vue)
  - Android & iOS
  - Flutter

视频通话（含 UI）组件介绍（TUICallKit）

最近更新时间：2024-05-08 11:30:04

组件介绍

TUICallKit 是腾讯云推出一款音视频通话 UI 组件，通过集成该组件，您只需要编写几行代码就可以为您的应用添加音视频通话功能。

支持平台

	Android	iOS	Web	uni-app 小程序	Flutter	uni-app 客户端	微信小程序
是否支持							
支持语言/框架	Kotlin Java	Swift Objective-C	Vue3 Vue2 React	Vue3 Vue2	Dart	Vue3 Vue2	-

功能介绍

基本功能	高级功能	功能优势
1v1 语音/视频通话 **群组通话**、中途邀请他人、中途加入 自定义来电铃声 自定义昵称、头像 **开启/关闭悬浮窗** 开启/关闭来电铃声	**离线推送** **虚拟背景** 云端录制 AI 降噪 全球互通 弱网卡顿优化 通话记录	**完善的 UI 交互** **支持多平台互联互通** **支持多设备登录**

适用场景

在线社交	在线问诊

在线教育	在线销售

在线体验

	Web	Android	iOS	其它
在线体验				/
demo 集成	Github: web	Github: Andorid	Github: iOS	Github: TUICallKit Demo

交流与反馈

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

开通服务（TUICallKit）

最近更新时间：2024-07-17 11:34:11

本文介绍如何开通音视频通话服务。

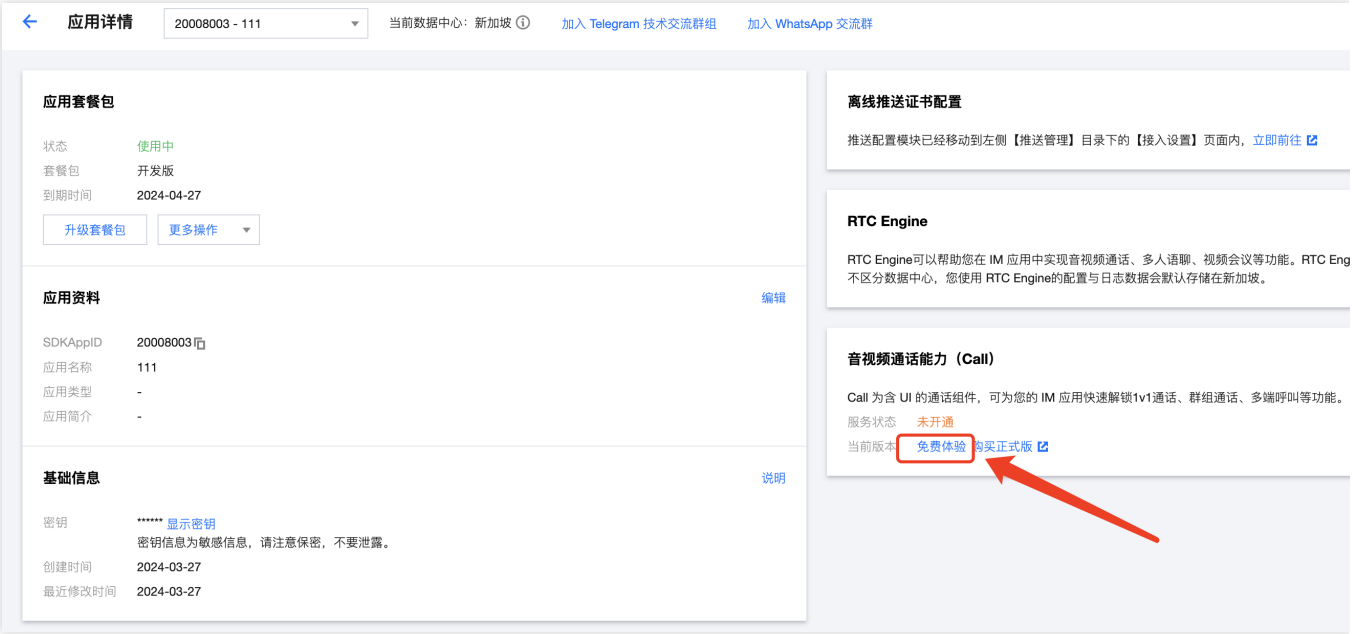
开通体验版

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。为了您能更好地体验音视频通话 Call 功能，我们免费为每个 SDKAppID 提供了7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7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。

您可以在 即时通信 IM 控制台中开通 Call 体验版，具体操作步骤如下：

1. 访问 IM 控制台，选择数据中心并新建应用。若您已有应用可跳过该步骤。

2. 单击目标应用卡片，进入应用的基础配置页面。
3. 找到**音视频通话能力（Call）**卡片，单击**免费体验**。



4. 确认弹窗内容后，单击**立即体验**，即可成功开通音视频通话体验版。

免费体验音视频通话能力（Call）

音视频通话能力（Call）为含 UI 的通话组件，可为您的 IM 应用快速解锁1v1通话、群组通话、多端呼叫等功能。

- 音视频通话能力提供**有效期为 7 天的体验版**，可免费体验音视频通话的全部功能。
- 音视频通话能力基于即时通信和实时音视频服务实现，因此开通音视频通话将为您默认开通实时音视频服务并创建应用。
- 音视频通话能力体验版**仅为您免费提供上述功能的使用权，您在使用过程中产生的通话时长用量将按** 实时音视频计费说明 计费，即
信费用按 即时通信计费说明 计费。
- 开通音视频通话能力的腾讯云账号（UIN）将每月获赠一个 10,000 分钟的 免费时长包 ，可用于抵扣使用过程中产生的通话时长用量
多说明参见 音视频通话 。

免费开通

5. 如果您的业务上线，需要购买正式版本，可前往控制台进行购买，请参见 购买正式版。

功能与计费说明

TUICallKit 由实时音视频 TRTC 和即时通信 IM 提供底层技术支持，**您需要购买指定套餐后方可使用 TUICallKit，**其中免费体验版的开通方式请参见 开通 TUICallKit 体验版 。

下表为 TUICallKit 推荐套餐对应的功能和价格，您也可以自由组合 TRTC 包月套餐和 IM 版本以获得对应版本的 TUICallKit。

对比项		体验版	1v1 通话版	群组通话版
定价		0 USD/7天	199 USD/月 立即选购	597 USD/月 立即选购
赠送资源	免费时长	10,000 分钟/月	10,000 分钟/月	10,000 分钟/月

	套餐赠送音视频通话时长	-	100,000 分钟/月	300,000 分钟/月
	免费月活跃用户数（MAU）	100/月	5,000/月	10,000/月
	超量后开启后付费（Call 版本有效期内）	超量后服务不可用	✓	✓
功能点	音频、视频通话	✓	✓	✓
	完整聊天应用 UI	✓	✓	✓
	通话状态展示	✓	✓	✓
	呼叫通知和离线推送（应用未打开时，通过消息推送来通知被呼叫信息）	✓	✓	✓
	通话悬浮窗（通话界面收起后，可以浮窗样式悬浮在应用界面上）	✓	✓	✓
	自定义呼叫铃声	✓	✓	✓
	通话呼叫/接听/拒绝/挂断	✓	✓	✓
	视频通话切换语音通话	✓	✓	✓
	1v1 通话	✓	✓	✓
	群组通话	✓	-	✓
	虚拟背景	✓	-	✓
	中途呼叫/加入三方通话	✓	-	✓
	通话记录 API（支持使用回调和 REST API 的方式获	✓	-	✓

	取通话记录数据）			
	多端登录通话（任一平台接通，自动停止其他端的接入请求）	✓	-	✓
	同平台多端登录通话（支持同一平台，如多个 iOS 设备登录，来电后任一设备接通，自动停止其他设备的接入请求）	✓（配合 IM 进阶版使用）	-	✓
	AI 降噪（借助AI能力有效过滤背景噪声，还原通话效果）	✓	-	✓
	户外通话卡顿优化（针对户外等弱网环境下优化卡顿率，秒开速度更快）	✓	-	✓
支持平台		iOS、Android、Web、Flutter	iOS、Android、Web、Flutter	iOS、Android、Web、Flutter

说明

：

1. 赠送资源：可用于抵扣使用 Call 过程中产生的用量，包括时长和免费月活跃用户数（MAU）。Call 体验版用量超出套餐时长和免费月活跃用户数后将无法继续使用；1v1通话版和群组通话版在套餐有效期内产生超出套餐时长和免费月活跃用户数（MAU）部分的用量将默认启用后付费模式计费。
2. 免费时长：所有开通 Call 的腾讯云账号（UIN）每月可获赠一个10,000分钟的免费时长包，可用于抵扣使用通话专属功能时产生的音视频通话时长，以及使用 TRTC 通用功能产生的云端录制和混流转码时长，更多说明请参见 免费时长说明。
3. 套餐赠送音视频通话时长：可用于抵扣使用通话专属功能时产生的音视频通话时长，有效期一个月，到期余量失效。
4. 免费月活跃用户数（MAU）：单个用户当月登录即时通信 IM 计为1个 MAU，同一用户重复登录时，MAU 不累加。

5. TRTC 通用功能：您在使用通话专属功能的同时，也可使用 TRTC 提供的通用功能，使用过程中将产生对应的用量，计费标准请参见 云端录制、混流转推。

快速接入（TUICallKit）
Android

最近更新时间：2024-05-15 17:01:07

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，跟随本文档，您将在 10 分钟内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的音视频通话功能。

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1：开通服务

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

步骤2：下载并导入组件

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

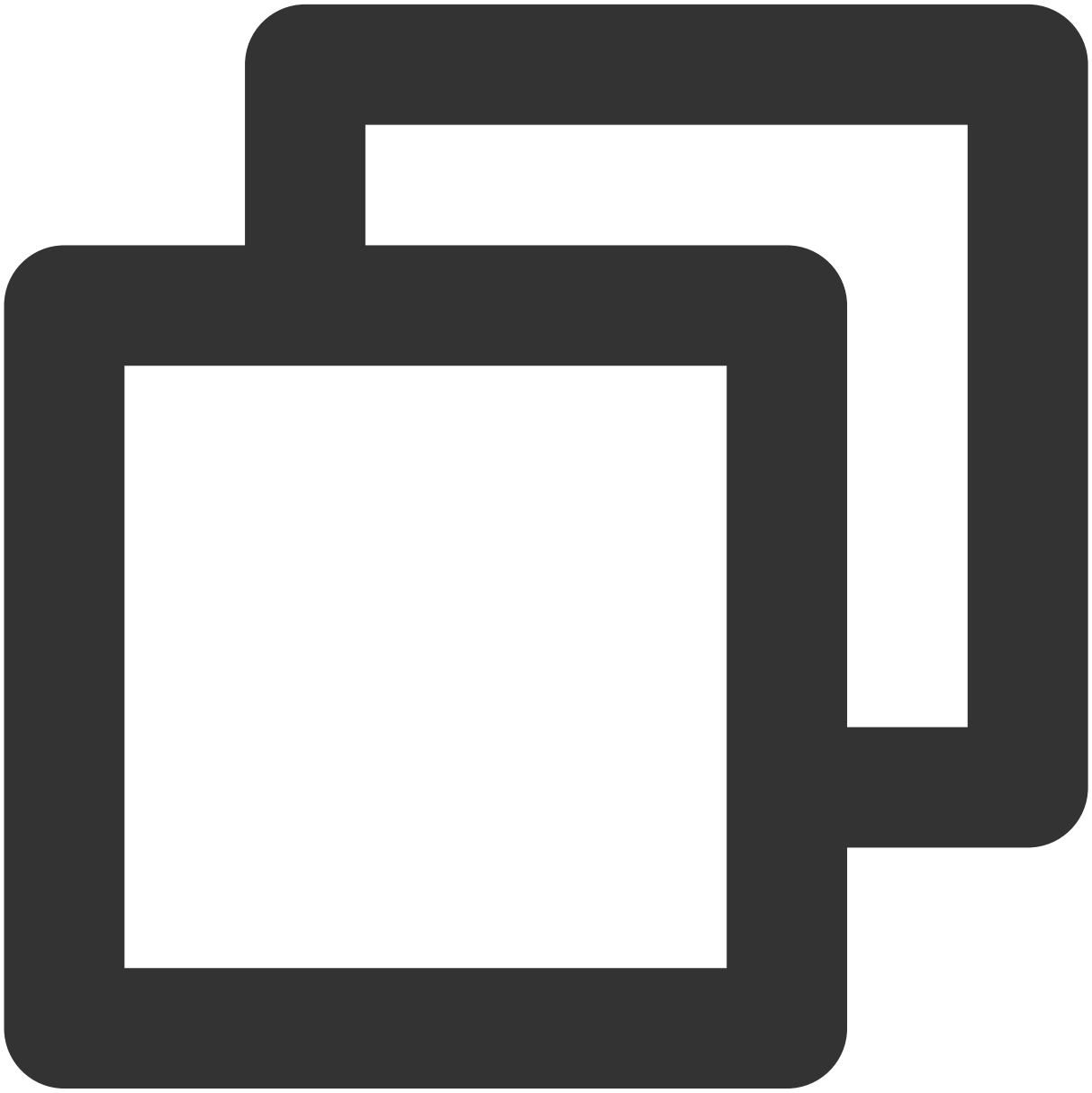
版权所有：腾讯云计算（北京）有限责任公司

步骤3：完成工程配置

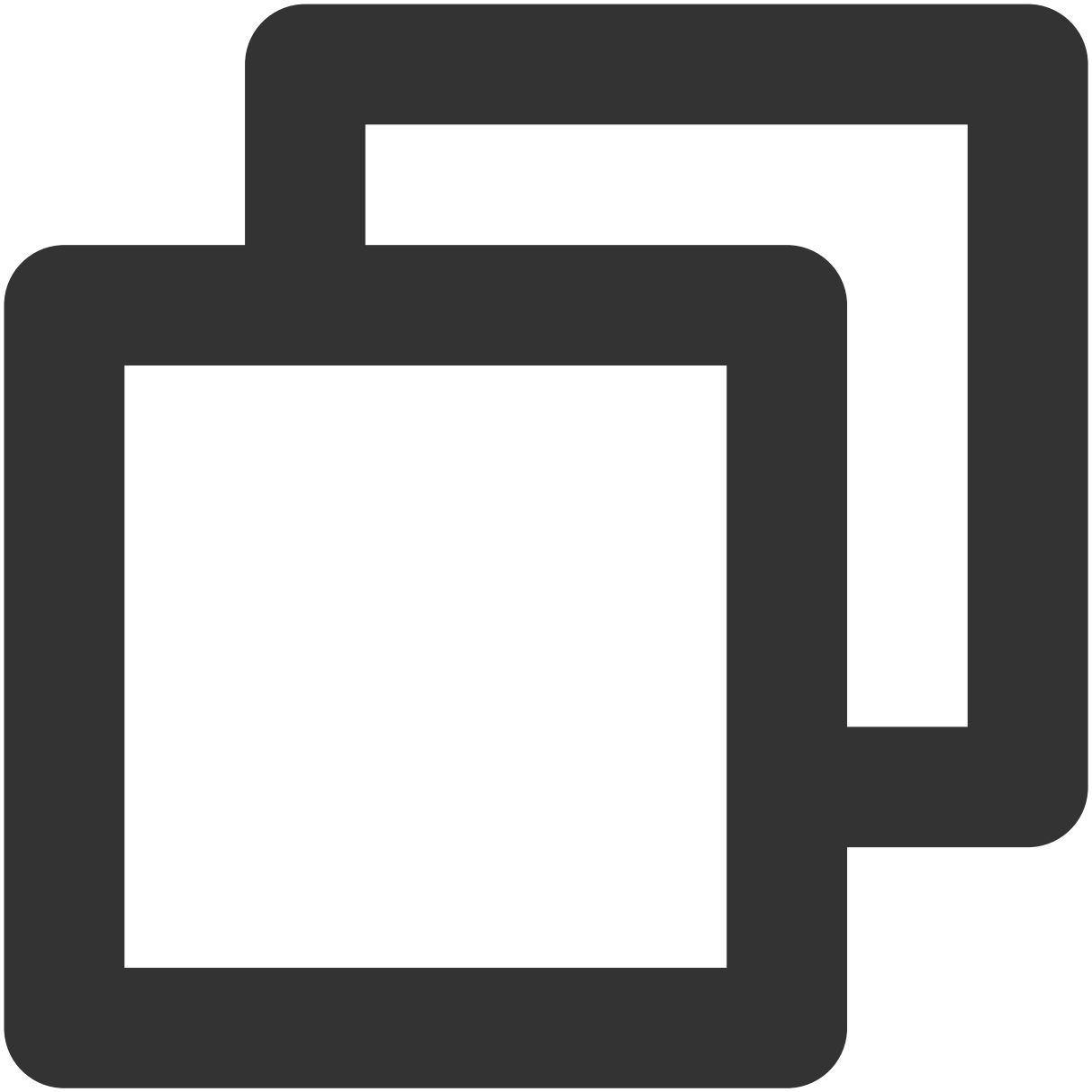
1. 在工程根目录下找到 `settings.gradle（或settings.gradle.kts）` 文件，在其中增加如下代码，它的作用是将 步骤二 中下载的组件导入到您当前的项目中：

settings.gradle

setting.gradle.kts



```
include ':tuicallkit-kt'
```

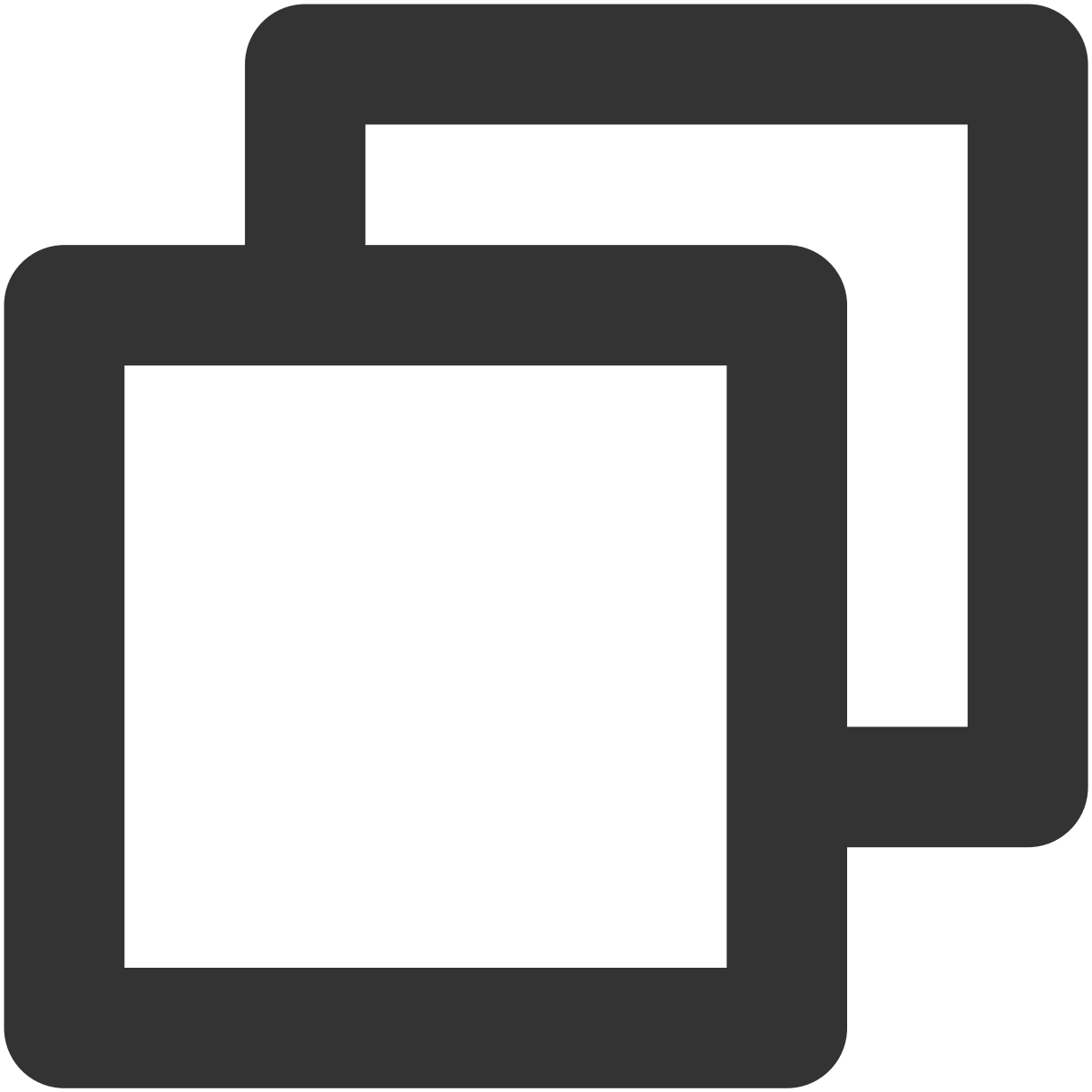


```
include(":tuicallkit-kt")
```

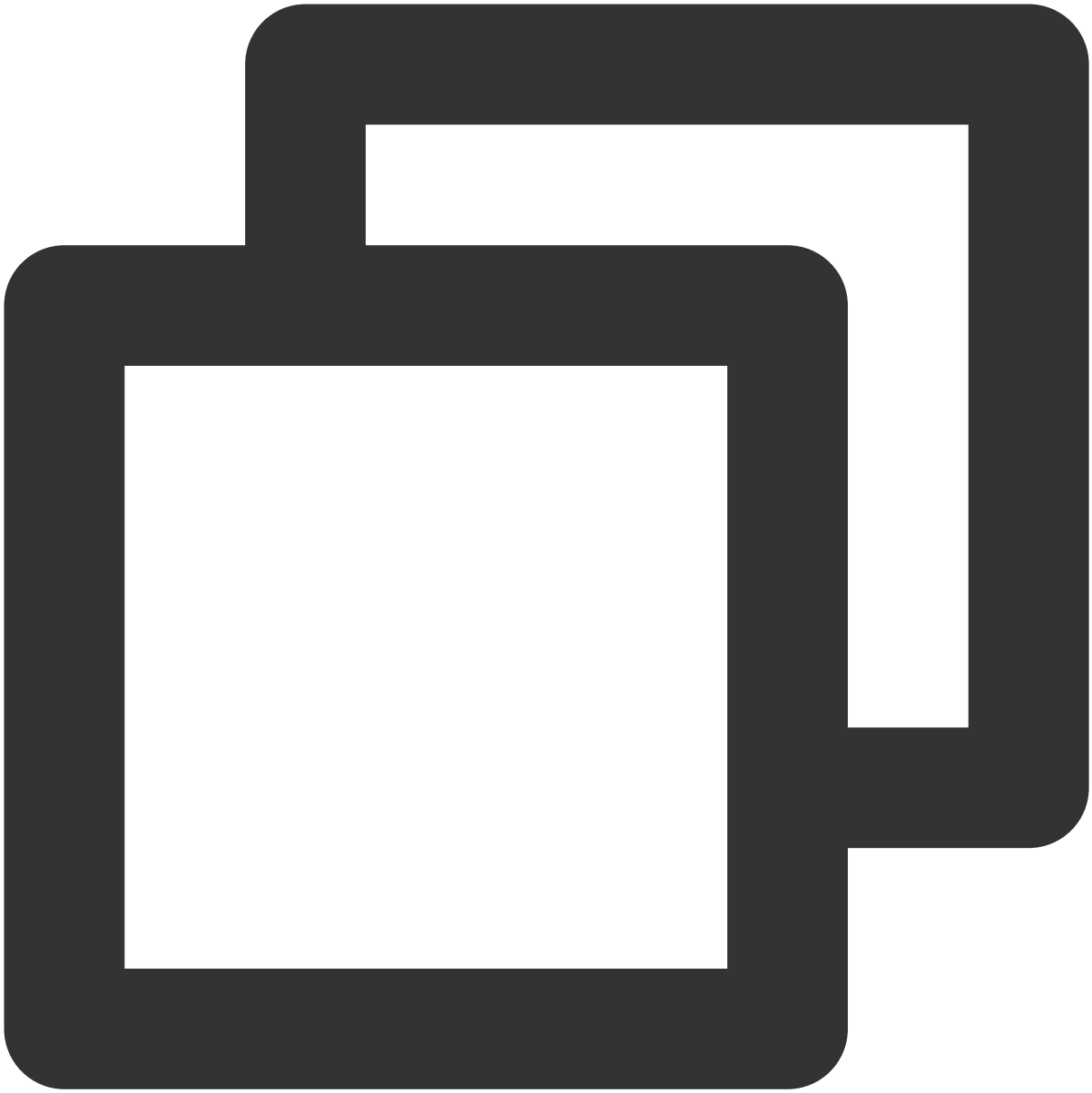
2. 在 app 目录下找到 `build.gradle(或build.gradle.kts)` 文件，增加如下代码，它的作用是声明当前 app 对新加入的组件的依赖：

build.gradle

build.gradle.kts



```
api project(':tuicallkit-kt')
```

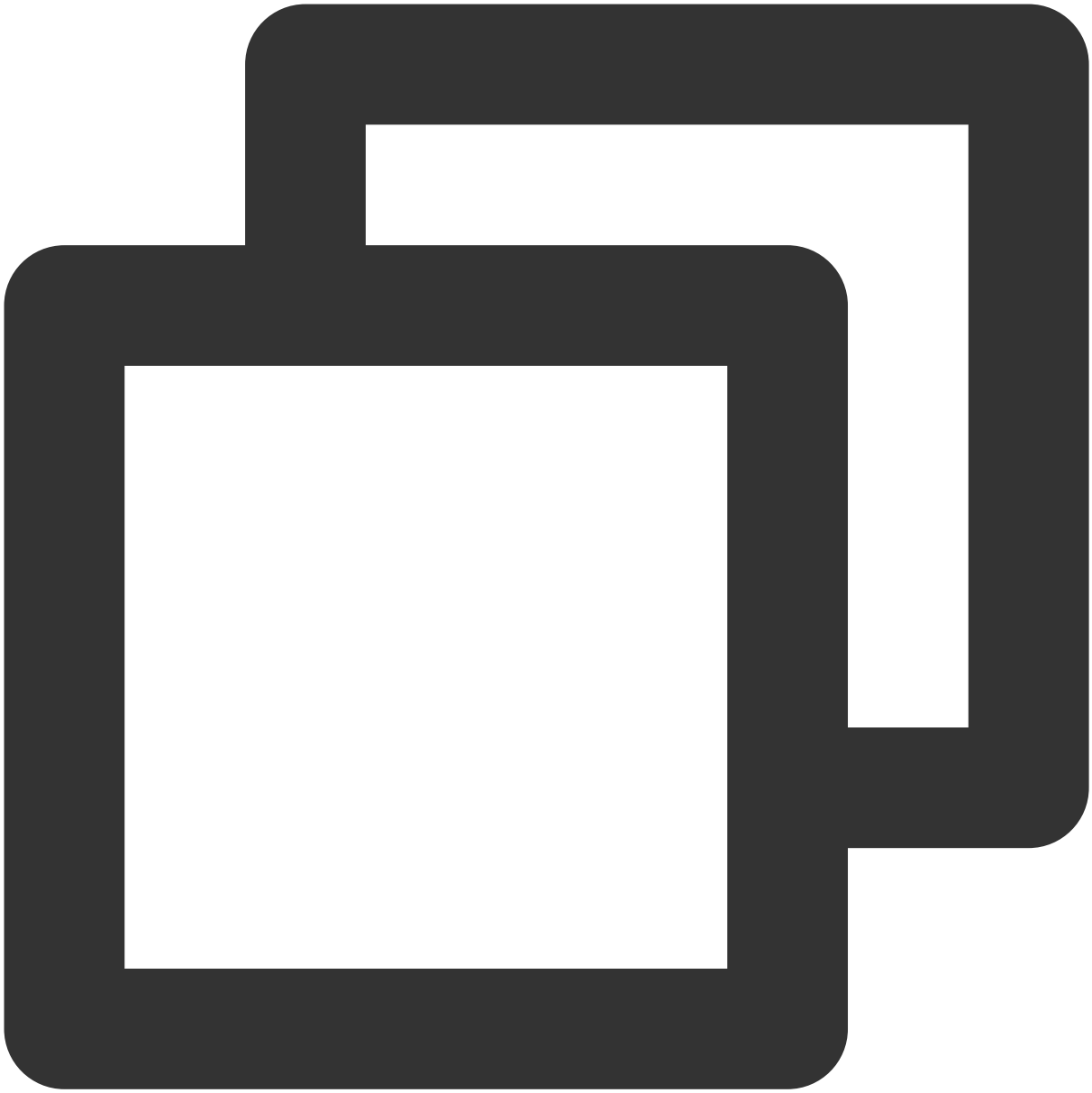


```
api(project(":tuicallkit-kt"))
```

说明

TUICallKit 工程内部已经默认依赖： `TRTC SDK` 、 `IM SDK` 、 `tuicallengine` 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 `tuicallkit-kt/build.gradle` 文件即可。

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



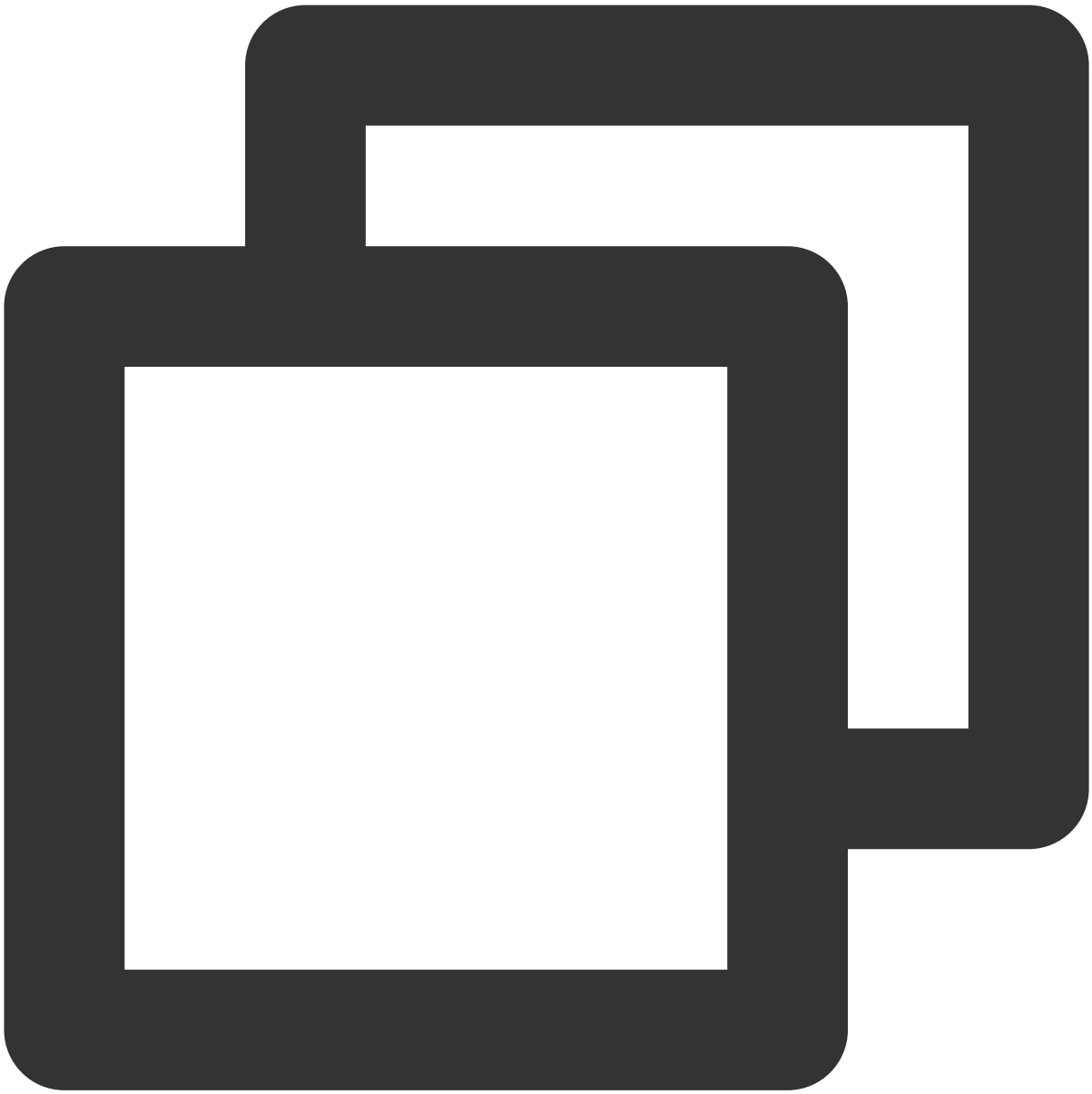
```
-keep class com.tencent.** { *; }
```

注意

TUICallKit 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 `tuicallkit-kt/src/main/AndroidManifest.xml` 。

步骤4：登录 TUI 组件

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。这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：



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

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

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 步骤一 的第 2 步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 辅助工具 生成一个临时可用的 UserSig。

更多信息请参见 如何计算及使用 UserSig。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。

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

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

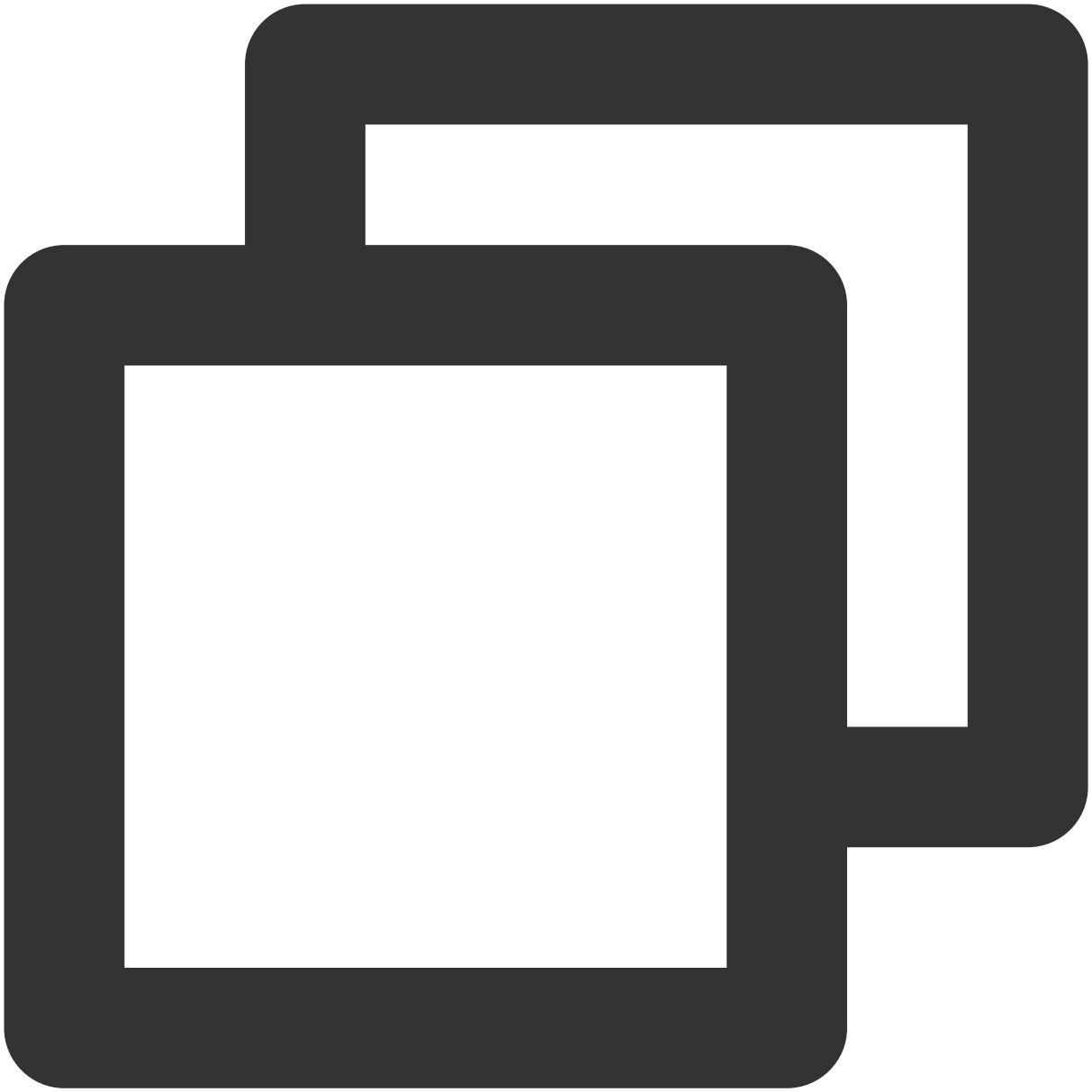
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤5：拨打您的第一通电话

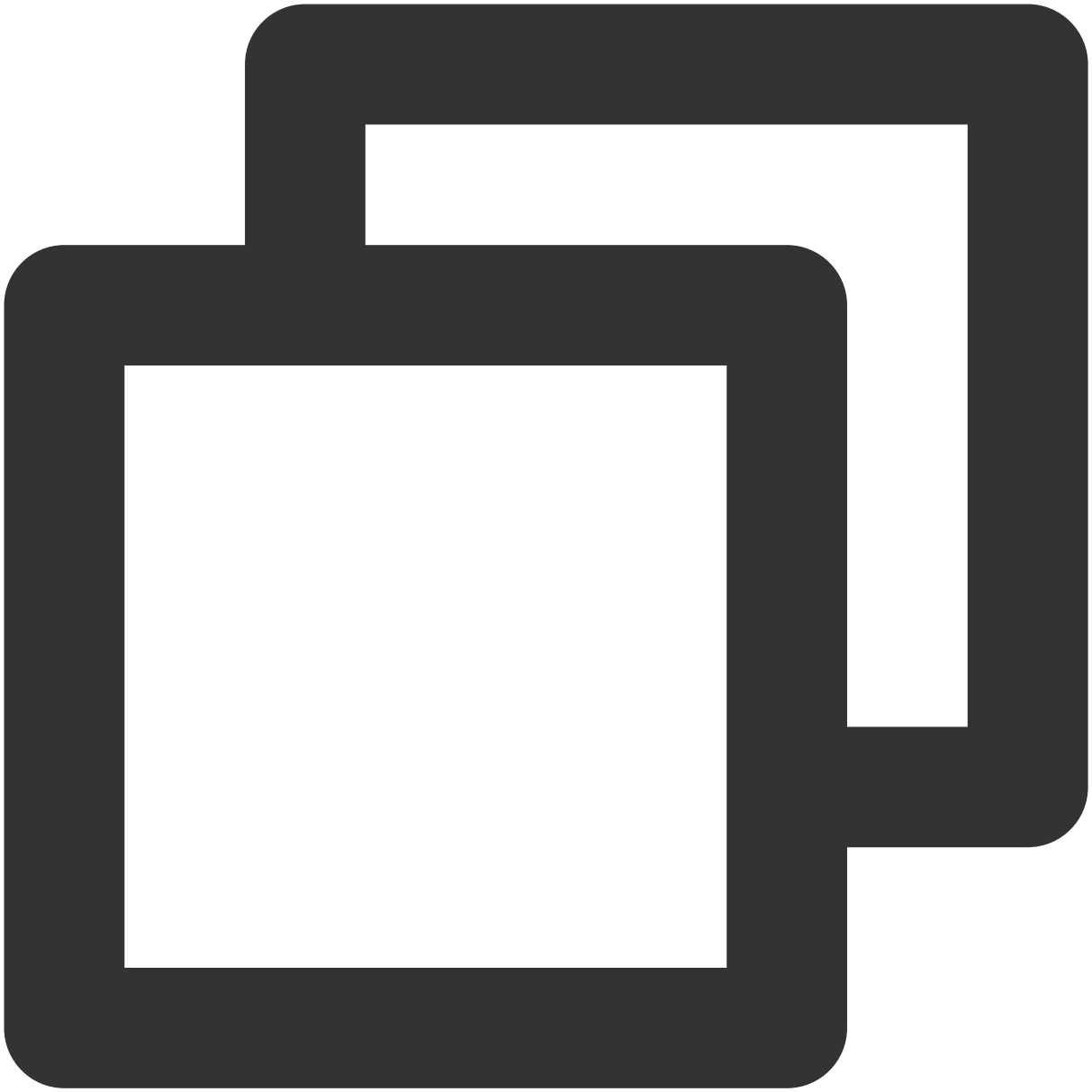
主叫方与被叫方登录成功后，主叫方通过调用 TUICallKit 的 call 方法并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话，被叫方此时就可接受到来电邀请。

Kotlin

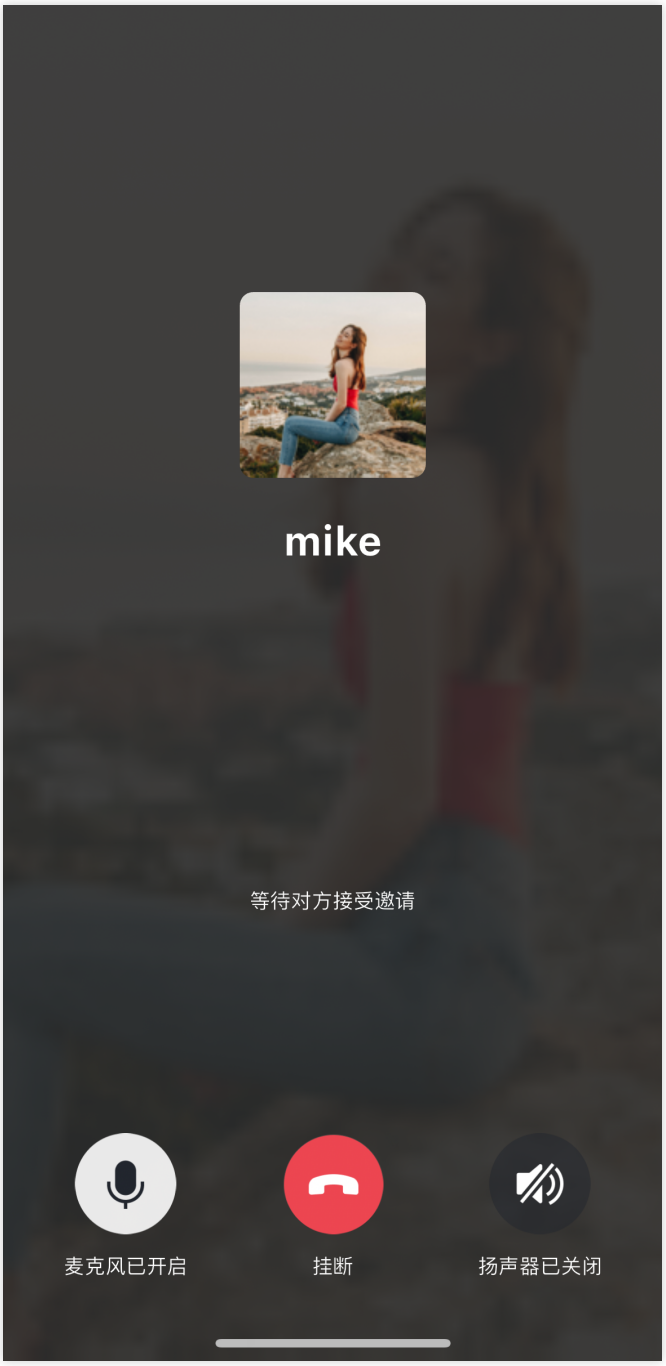
Java



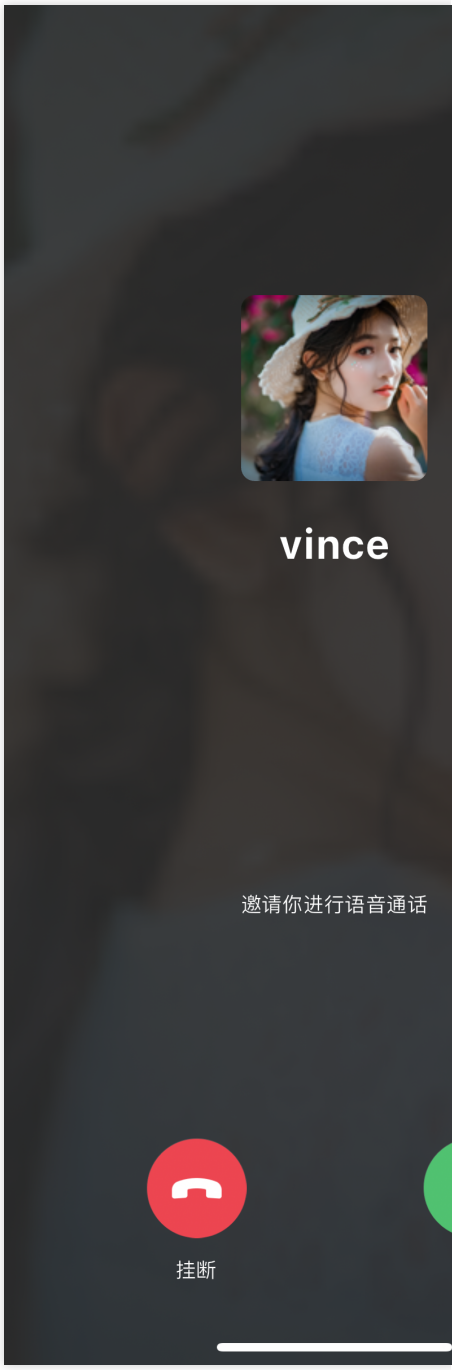
```
// 发起1对1视频通话(假设 UserID 为 mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video)
```



```
// 发起1对1视频通话(假设 UserID 为 mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video);
```



主叫方



被叫方

更多特性

界面定制
群组通话
悬浮窗
自定义铃声

监听通话状态
云端录制

常见问题

1、错误提示“The package you purchased does not support this ability”？

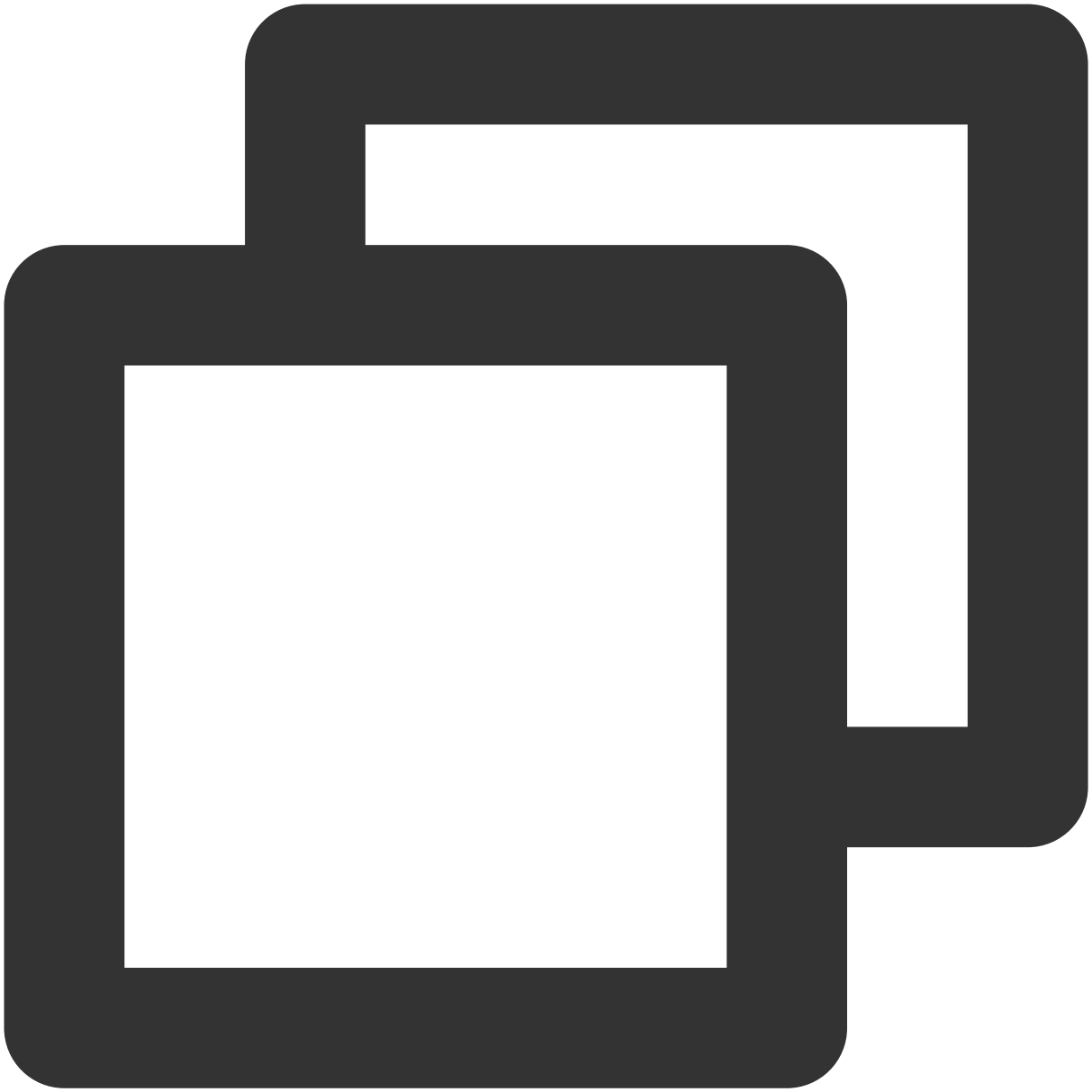
如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2、如何购买套餐？

请参考购买链接 购买正式版。

3、TUICallKit 崩溃，崩溃日志："No implementation found for xxxx"？

堆栈信息如下：



```
java.lang.UnsatisfiedLinkError: No implementation found for void com.tencent.liteav
        at com.tencent.liteav.base.Log.nativeWriteLogToNative(Native Method)
        at com.tencent.liteav.base.Log.i(SourceFile:177)
        at com.tencent.liteav.basic.log.TXCLog.i(SourceFile:36)
        at com.tencent.liteav.trtccalling.model.impl.base.TRTCLogger.i(TRTCLogger.ja
        at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.init(Service
        at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.onCreate(Ser
        at android.content.ContentProvider.attachInfo(ContentProvider.java:2097)
        at android.content.ContentProvider.attachInfo(ContentProvider.java:2070)
        at android.app.ActivityThread.installProvider(ActivityThread.java:8168)
        at android.app.ActivityThread.installContentProviders(ActivityThread.java:77
```

```
        at android.app.ActivityThread.handleBindApplication(ActivityThread.java:7573
        at android.app.ActivityThread.access$2600(ActivityThread.java:260)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2435)
        at android.os.Handler.dispatchMessage(Handler.java:110)
        at android.os.Looper.loop(Looper.java:219)
        at android.app.ActivityThread.main(ActivityThread.java:8668)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.j
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1109)
```

因为 TUICallKit 依赖的 TRTC 等 SDK 部分接口使用了 JNI 实现，Android Studio 在某些条件进行编译 APK 的时候可能并不会把 Native 的 .so 库打包进去，导致出现这类报错，重新 Clean 一下即可。

交流与反馈

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

iOS

最近更新时间：2024-05-15 17:01:46

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，跟随本文档，您将在一个小时的时间内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

环境准备

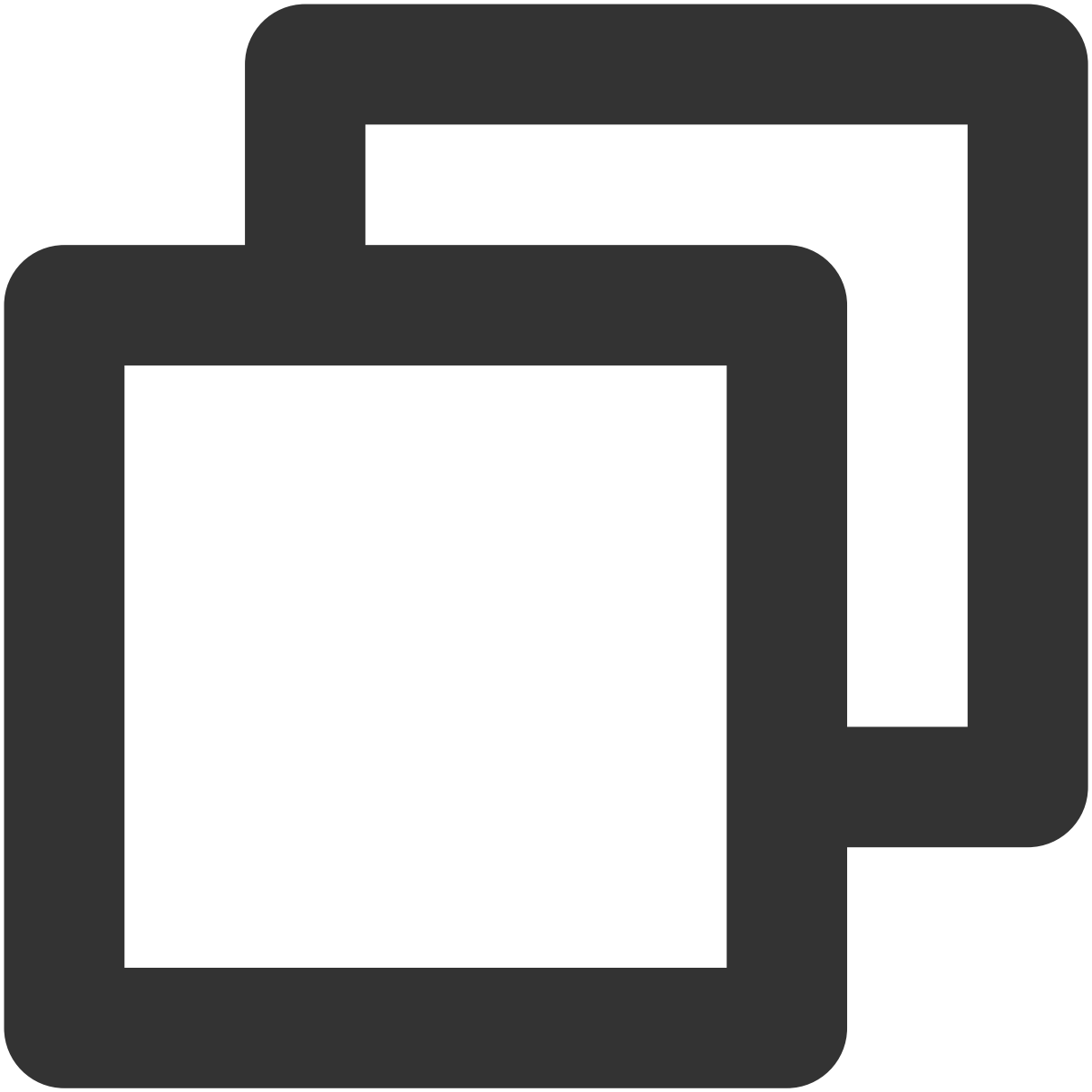
Xcode 13 及以上
iOS 12.0 及以上

步骤1：开通服务

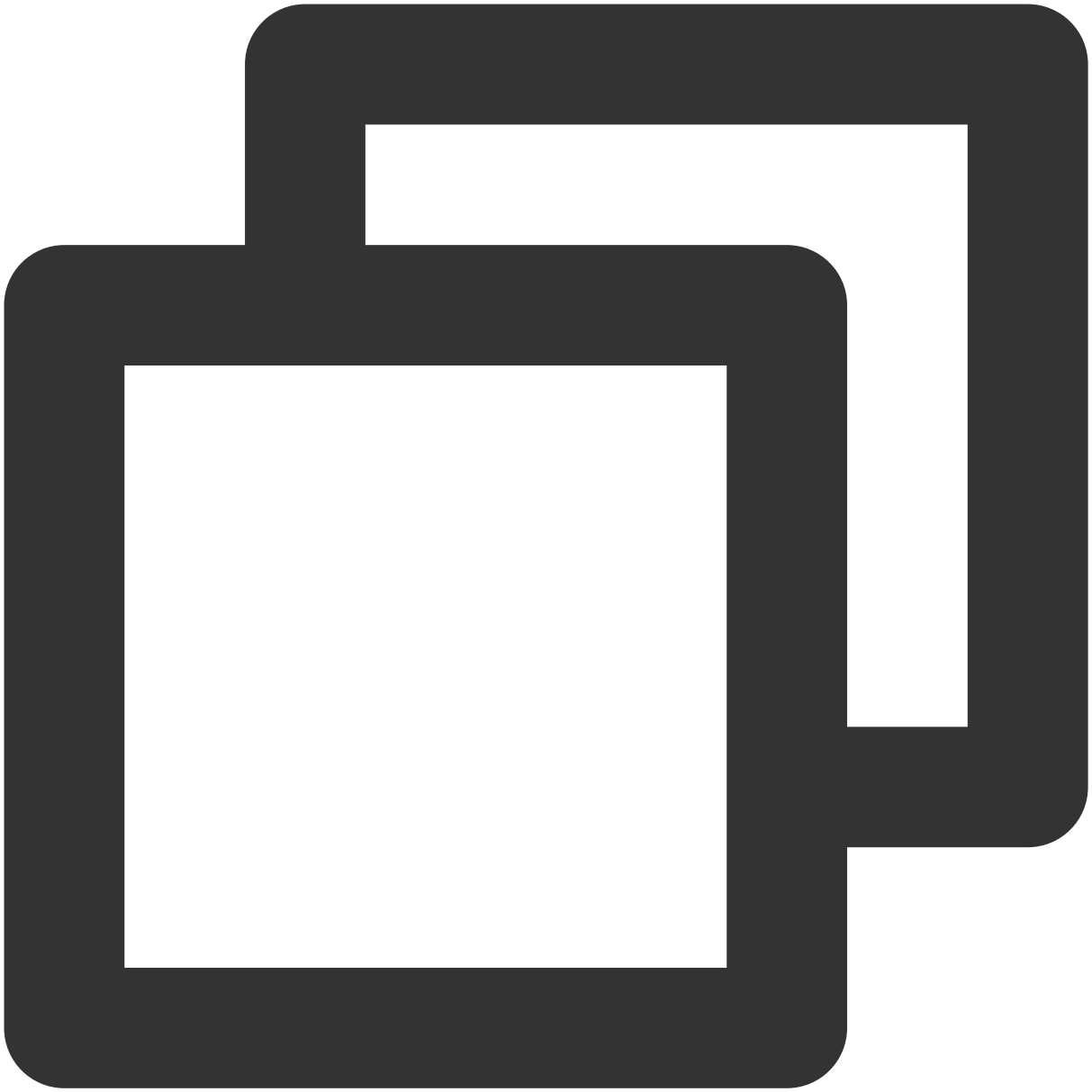
在使用腾讯云提供的音视频服务前，您需要前往控制台，为应用开通音视频服务。具体步骤请参考开通服务。

步骤2：导入组件

使用 CocoaPods 导入组件，具体步骤如下：
1. 在您的 `Podfile` 文件中添加以下依赖。
Swift
Objective-C

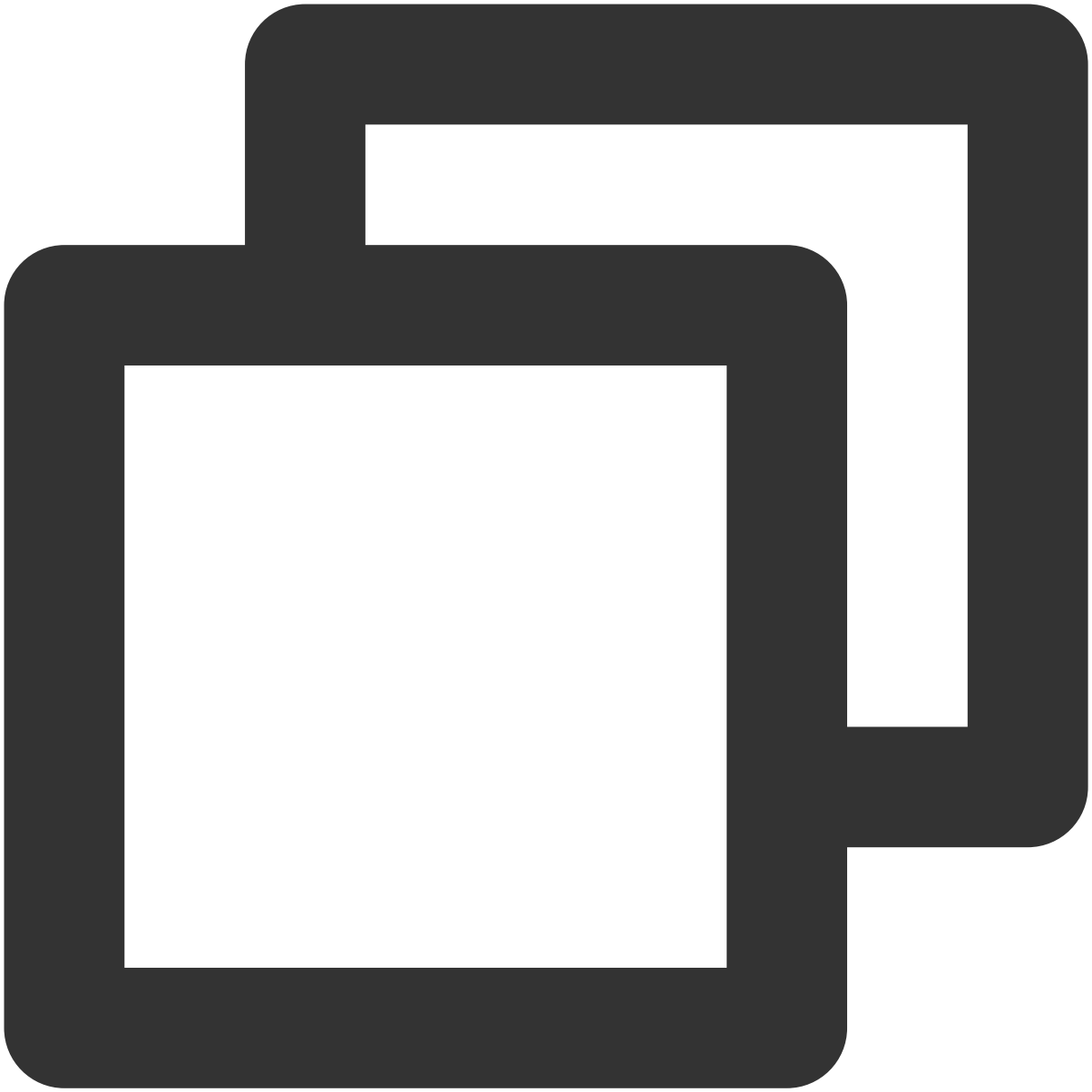


```
pod 'TUICallKit_Swift'
```



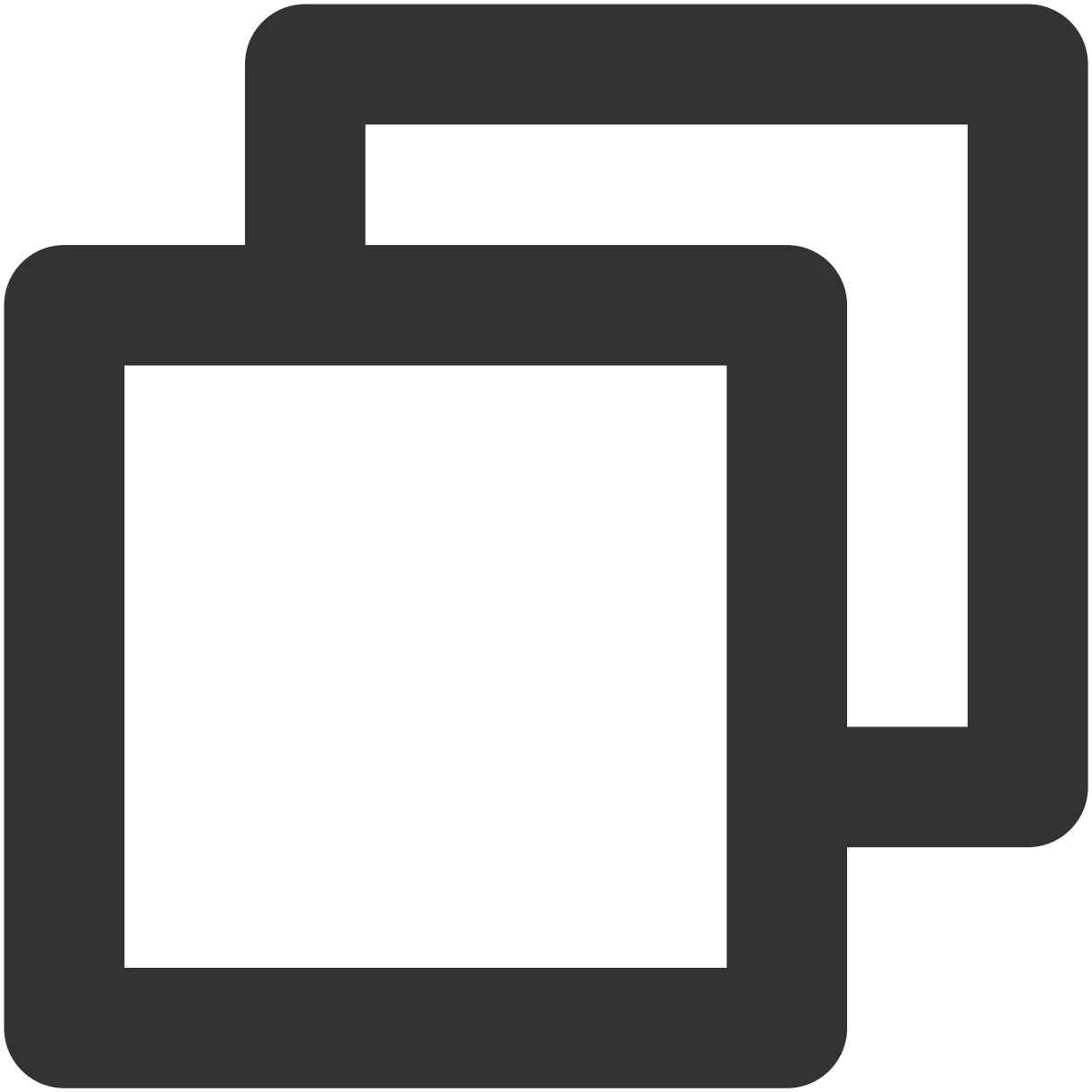
```
pod 'TUICallKit'
```

2. 执行以下命令，安装组件。



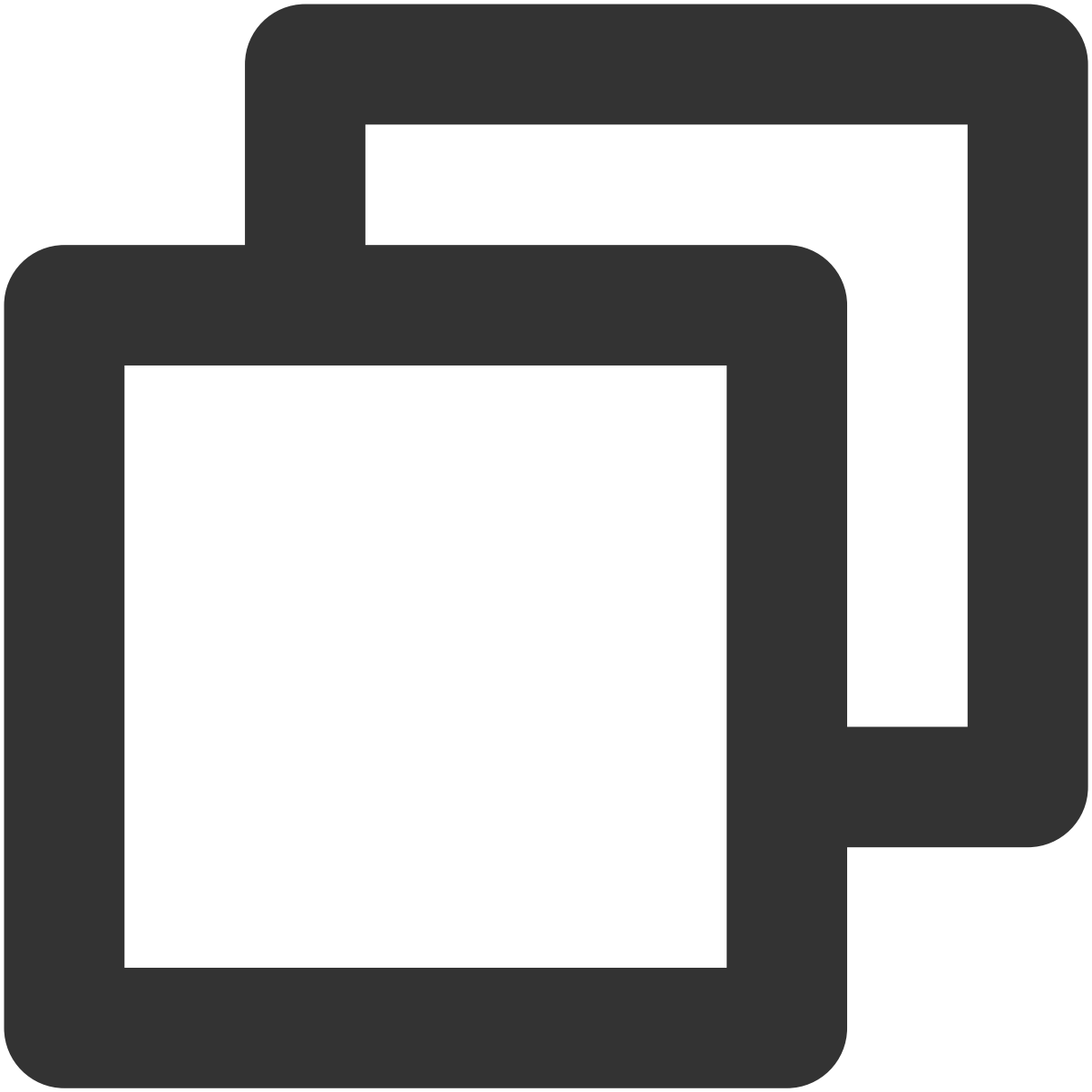
```
pod install
```

如果无法安装 TUICallKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。



```
pod repo update
```

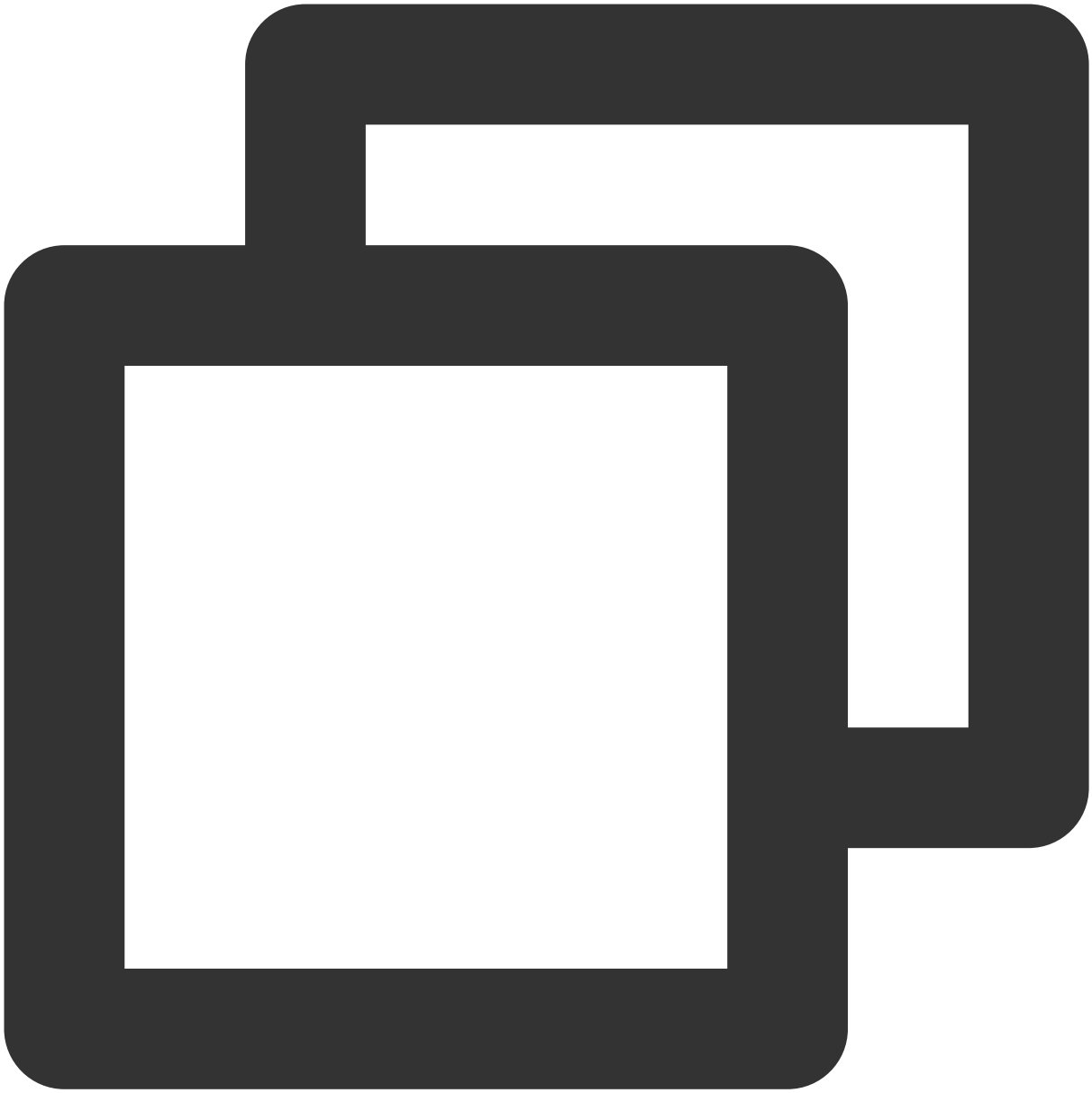
之后执行以下命令，更新组件库的 Pod 版本。



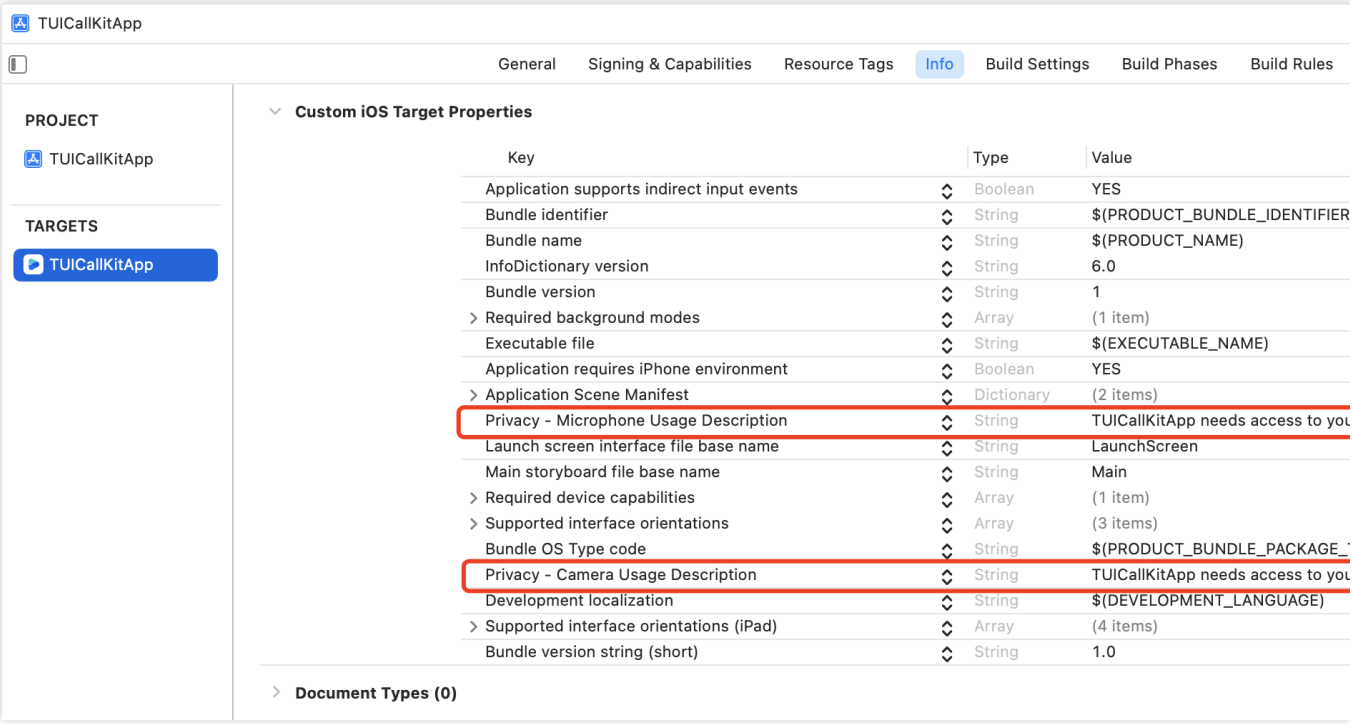
```
pod update
```

步骤3：完成工程配置

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

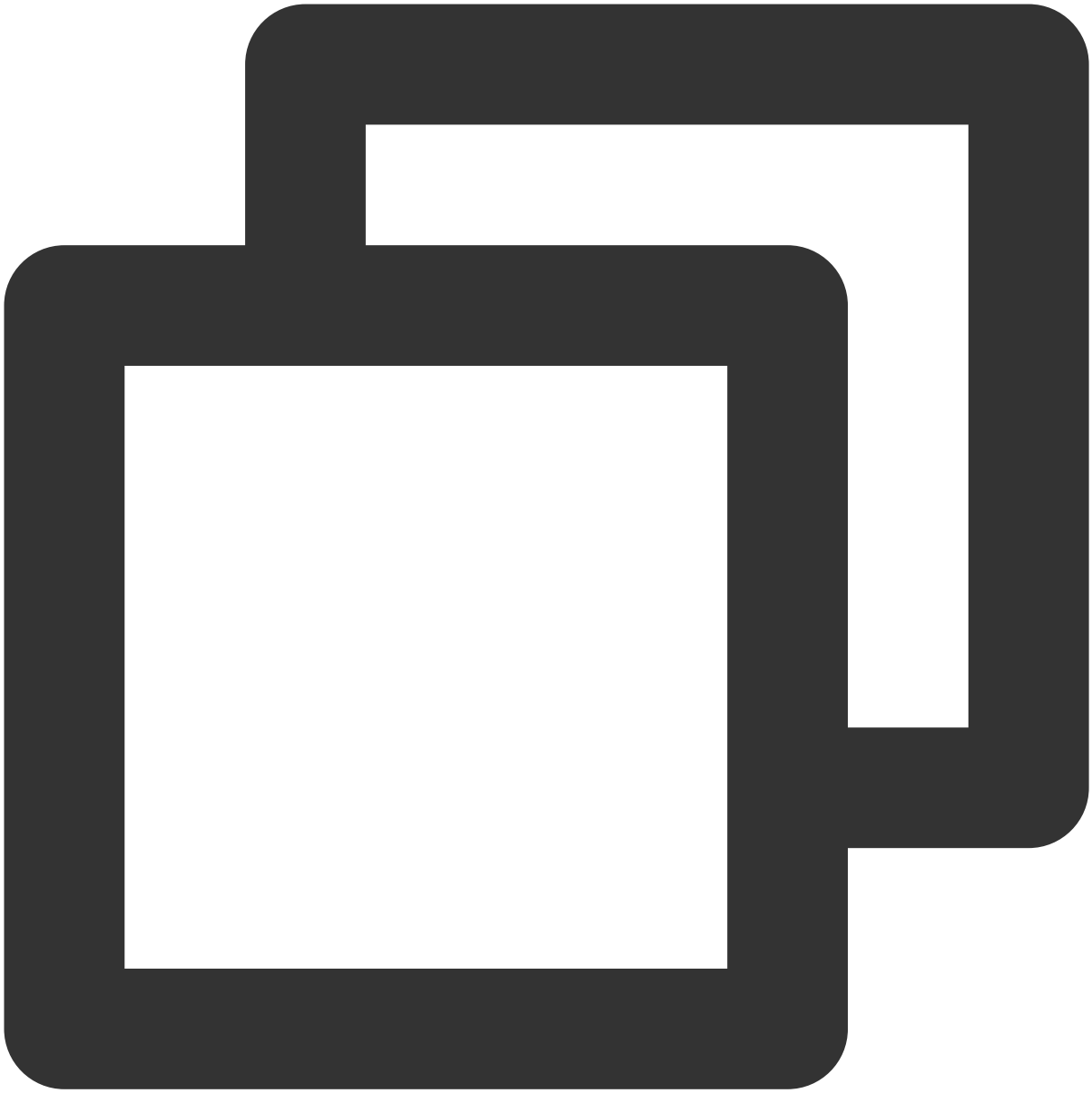


步骤4：登录 TUI 组件

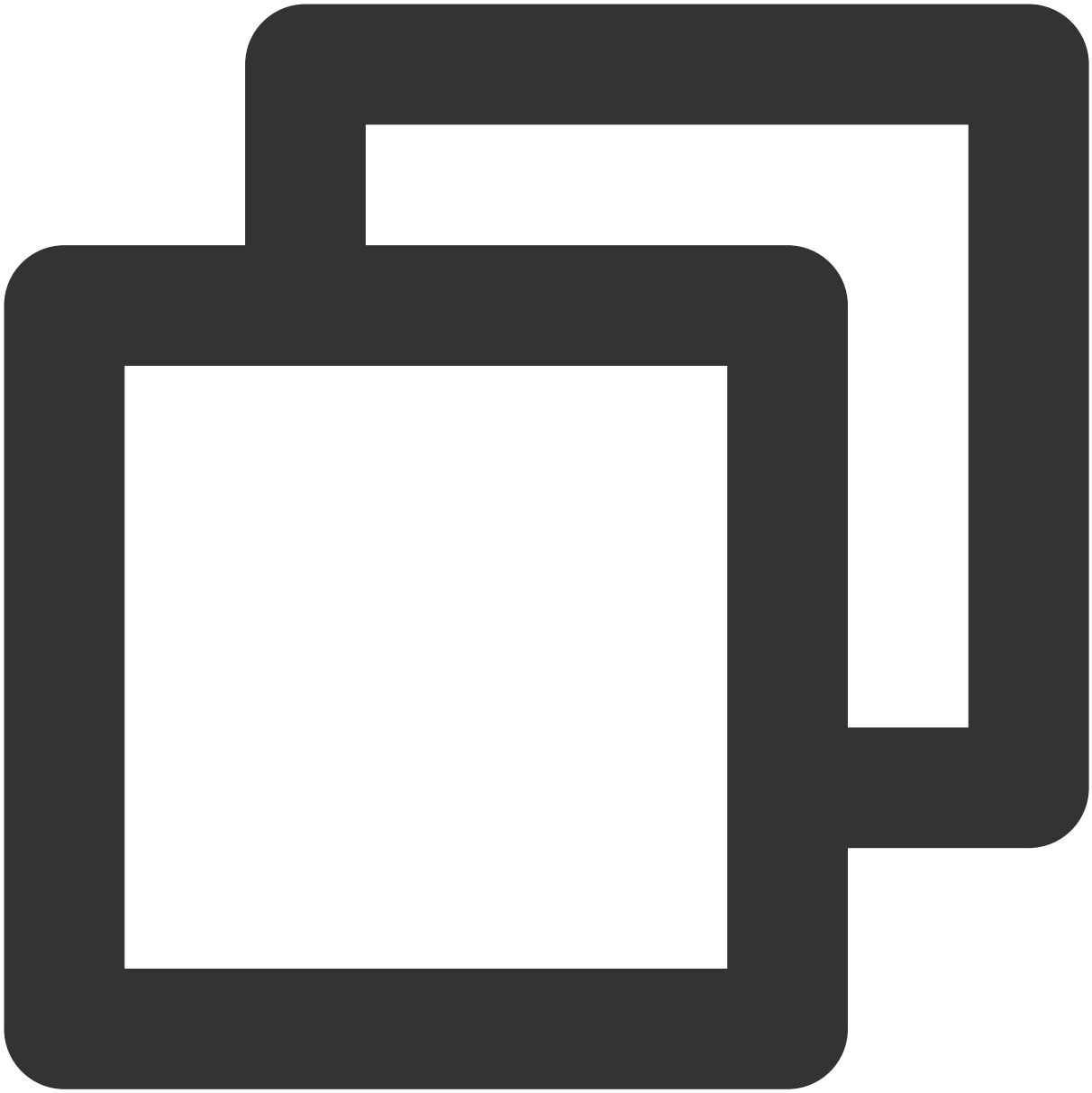
在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Swift

Objective-C



```
import TUICore

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")")
}
```



```
#import <TUICore/TUILogin.h>

[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 并填在这个位置
           succ:^{
    NSLog(@"login success");
} fail:^(int code, NSString *msg) {
    NSLog(@"login failed, code: %d, error: %@", code, msg);
}
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

SDKAppID：在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 步骤一 的第 2 步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 辅助工具 生成一个临时可用的 UserSig。

更多信息请参见 如何计算及使用 UserSig。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。

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

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

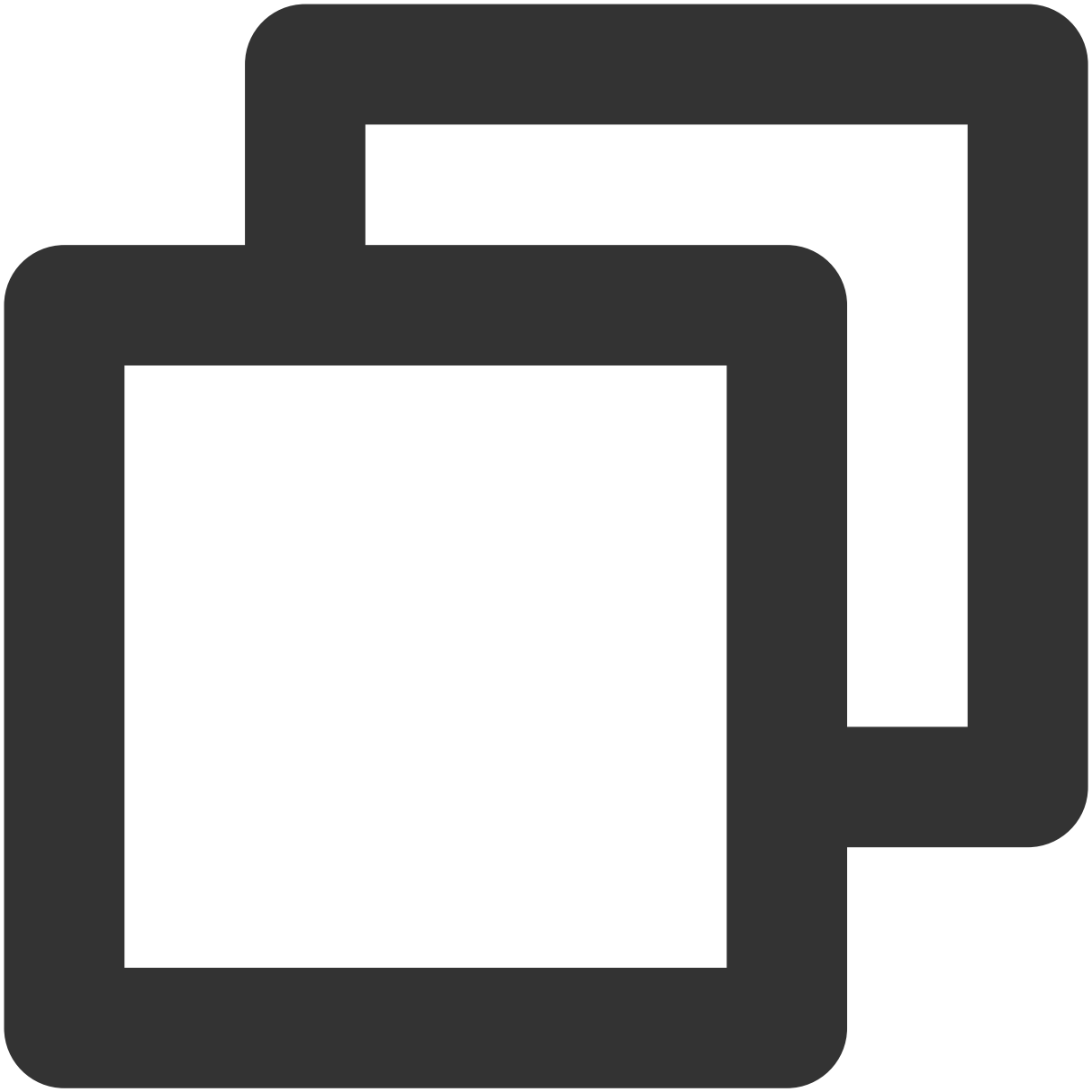
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤5：设置昵称和头像

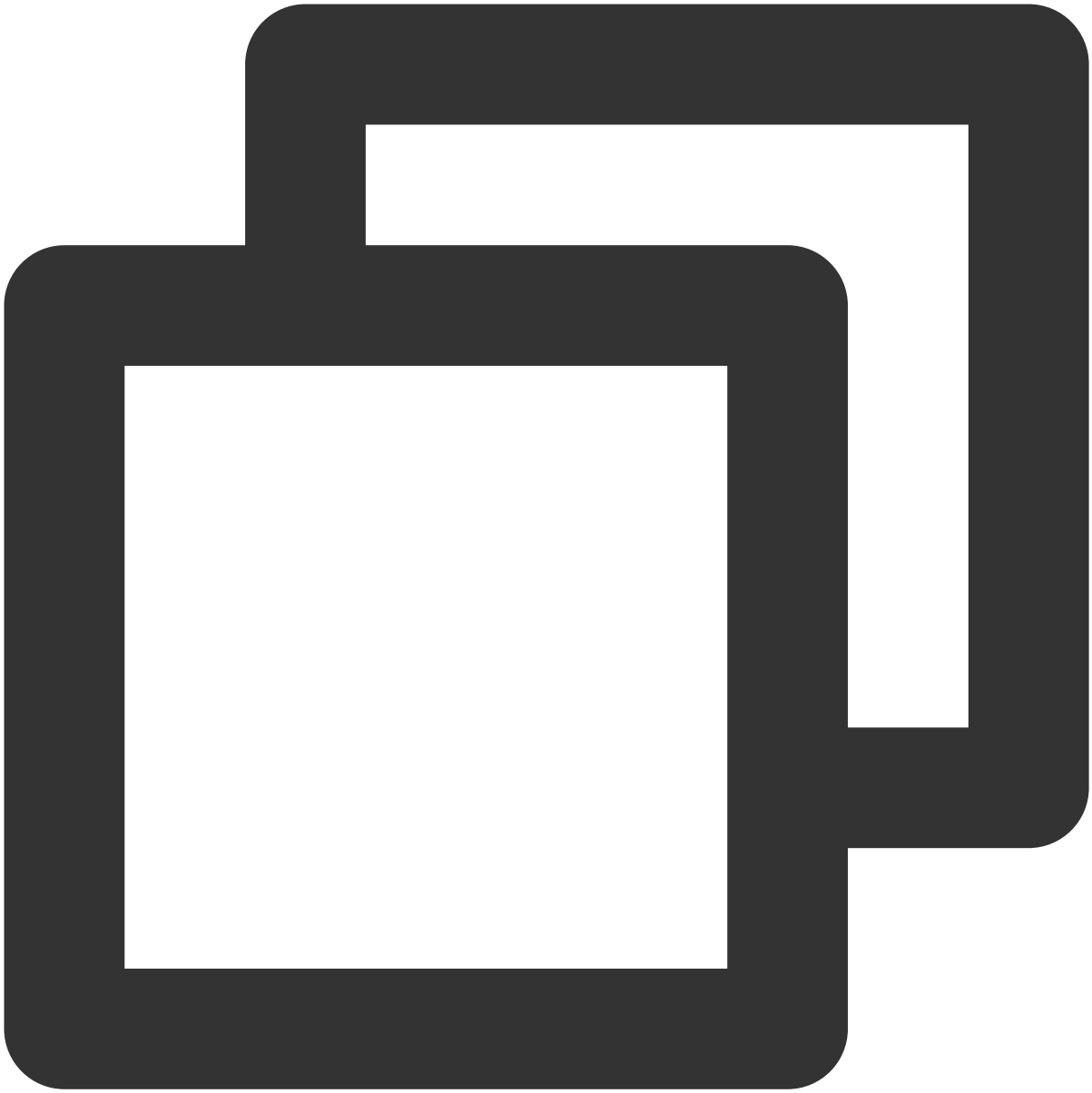
如果您需要自定义昵称或头像，可以使用如下接口进行更新：

Swift

Objective-C



```
TUICallKit.createInstance().setSelfInfo(nickname: "昵称", avatar: "头像url") {

} fail: { code, message in

}
```



```
[[TUICallKit createInstance] setSelfInfo:@"昵称" avatar:@"头像url" succ:^{

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

}];
```

注意：

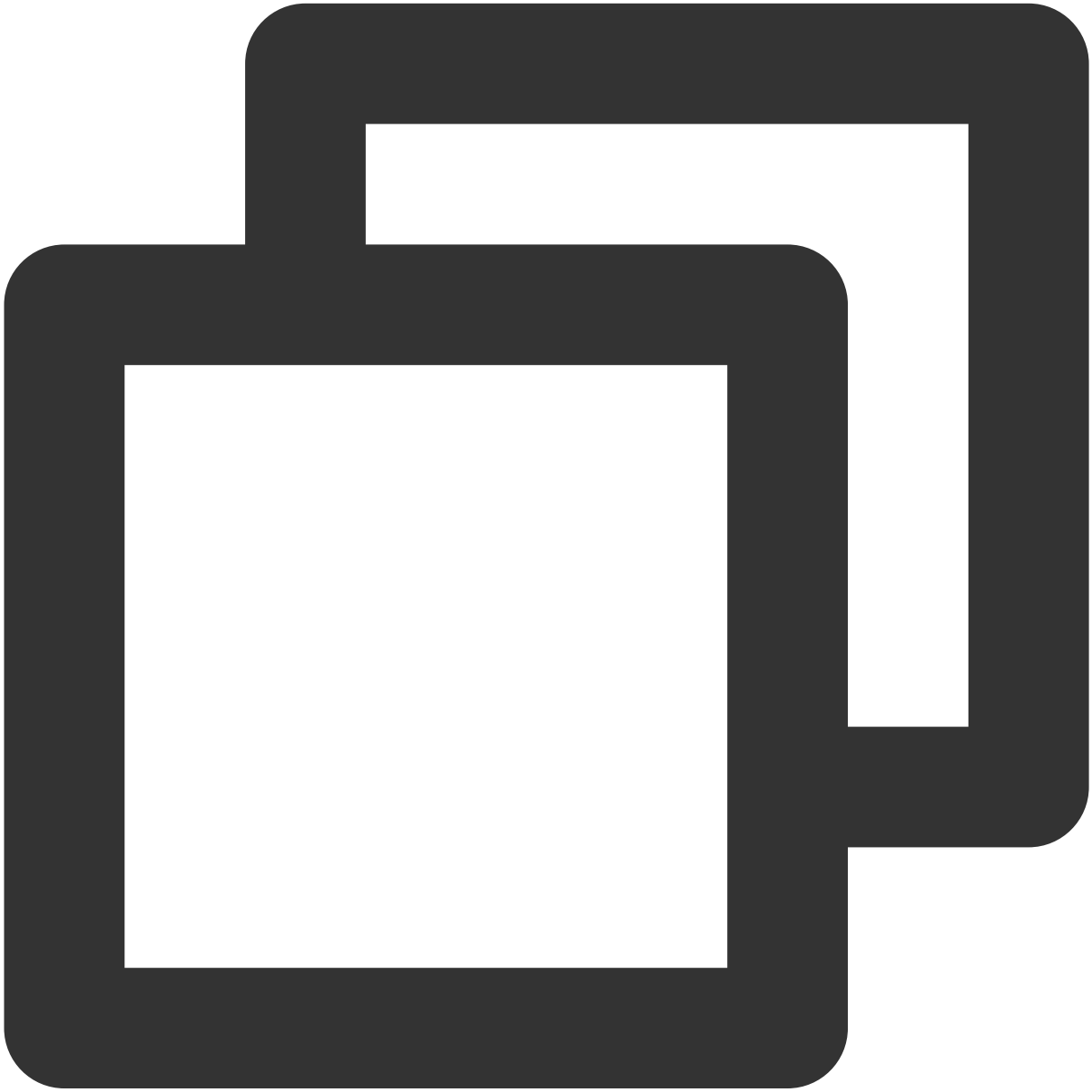
因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

步骤6：拨打您的第一通电话

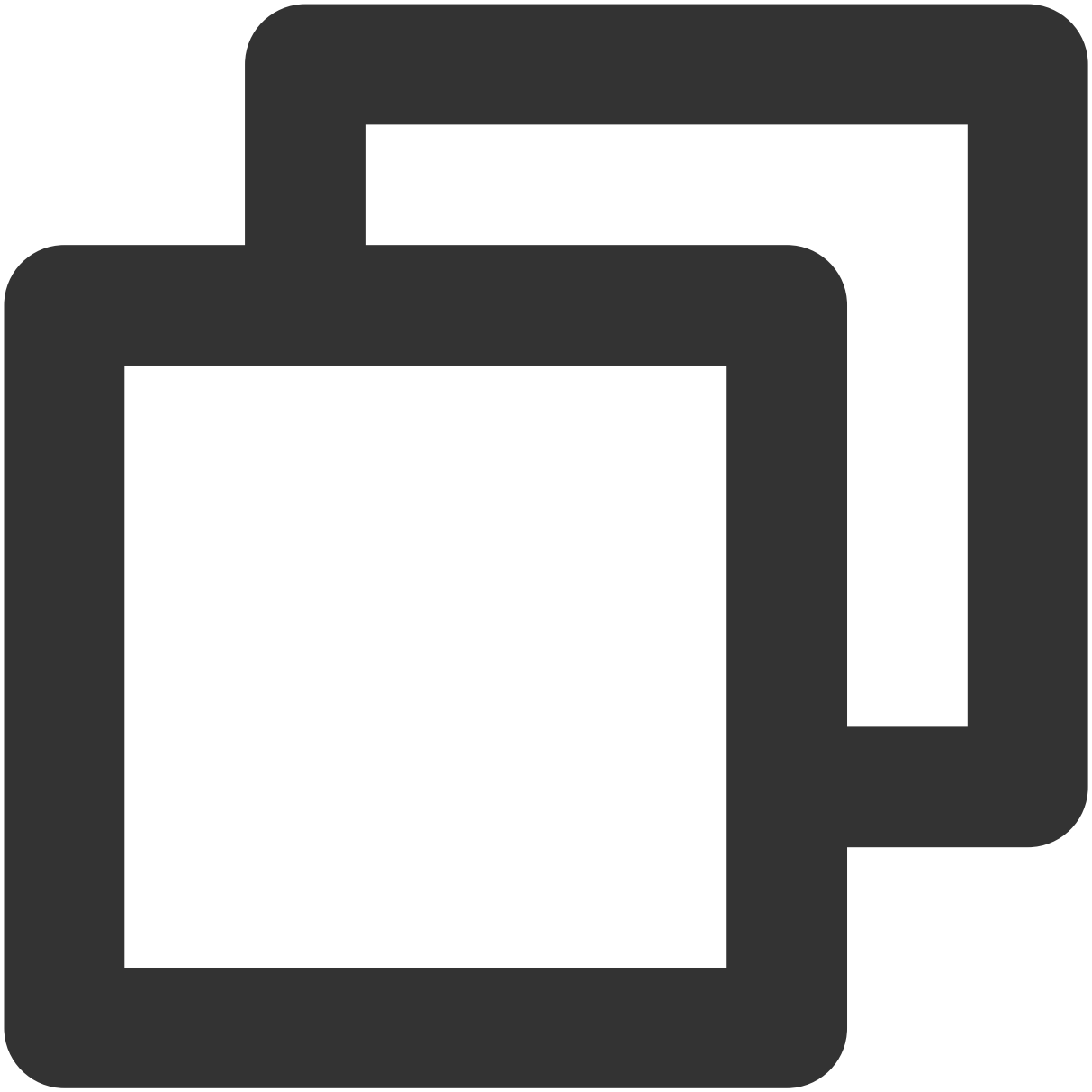
通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

Swift

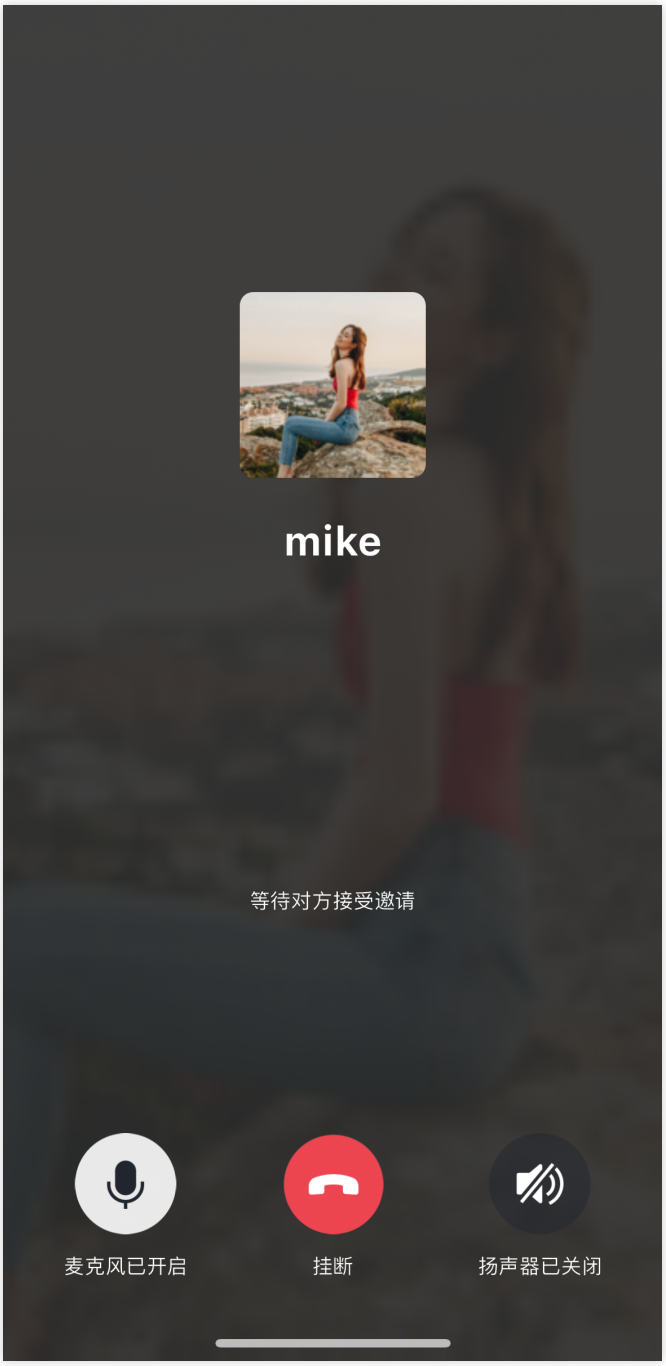
Objective-C



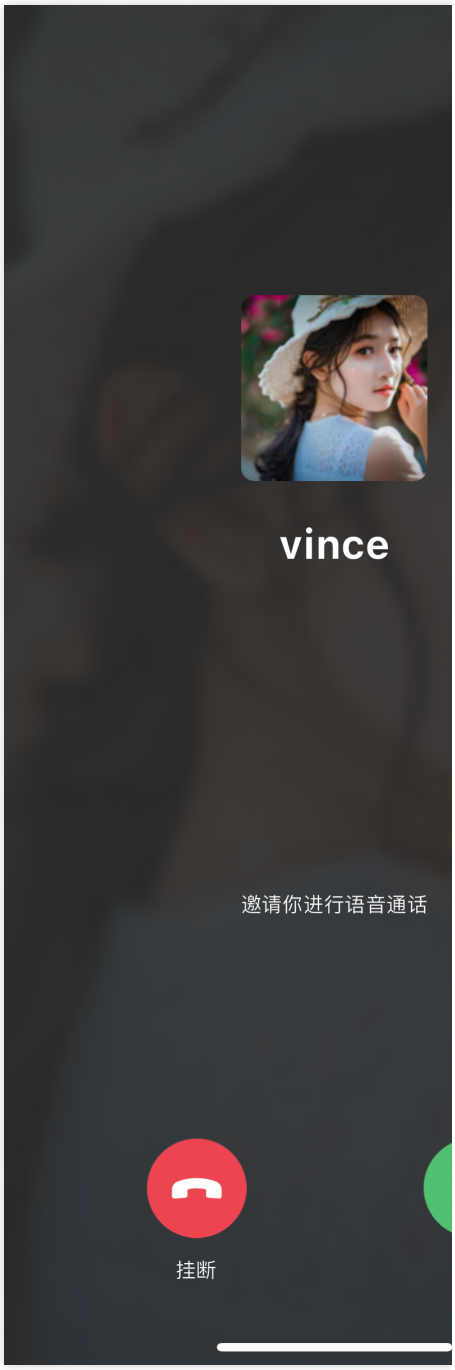
```
import TUICallKit

// 发起1对1视频通话(假设 userId 为 mike)
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video)
```



```
#import <TUICallKit/TUICallKit.h>

// 发起1对1视频通话(假设 userId 为 mike)
[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```



主叫方



被叫方

更多特性

界面定制
群组通话
悬浮窗
自定义铃声

监听通话状态
云端录制

常见问题

1、错误提示“The package you purchased does not support this ability”怎么处理？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2、如何购买套餐？

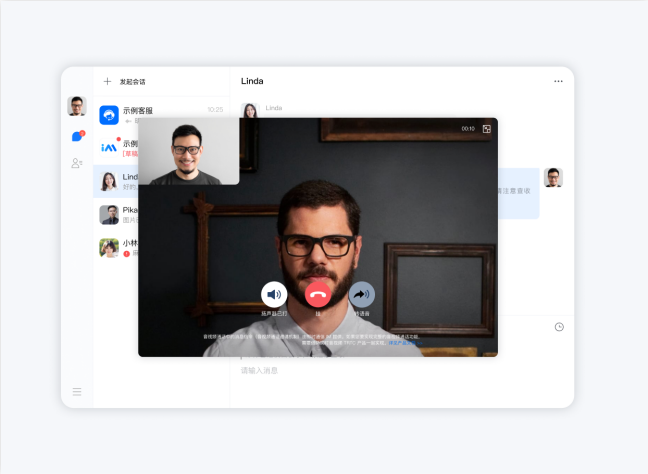
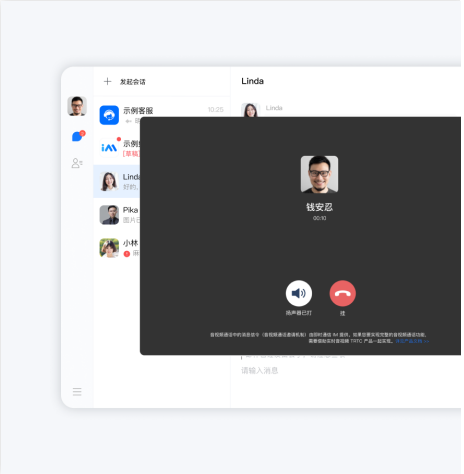
请参考购买链接 购买正式版。

交流与反馈

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

Web&H5

最近更新时间：2024-07-23 18:04:12

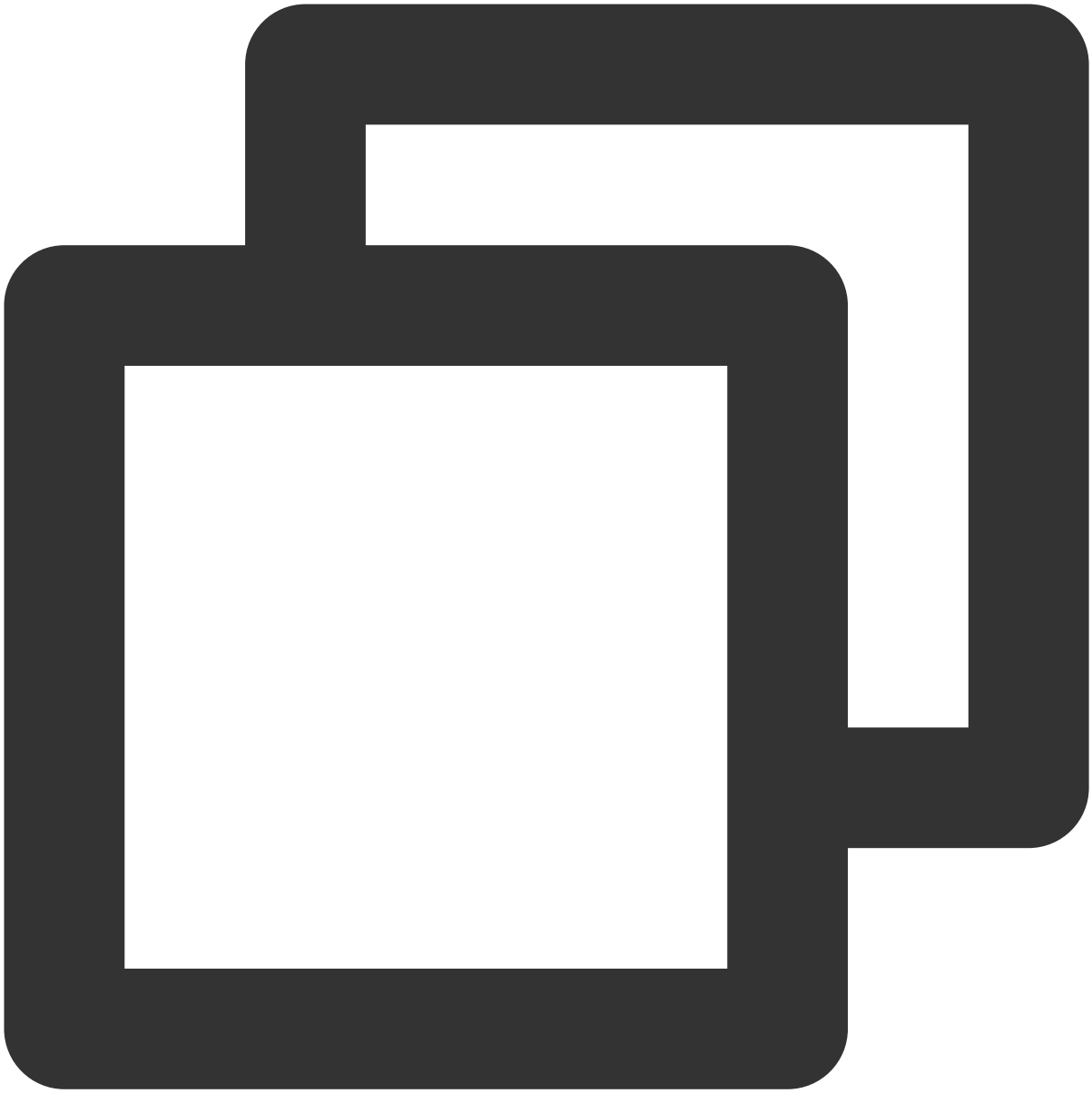
视频通话	语音通话
	

步骤1：开通音视频通话能力

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。具体步骤请参考开通服务。

步骤2：下载 TUICallKit 组件

通过 npm 方式下载 TUICallKit 组件：

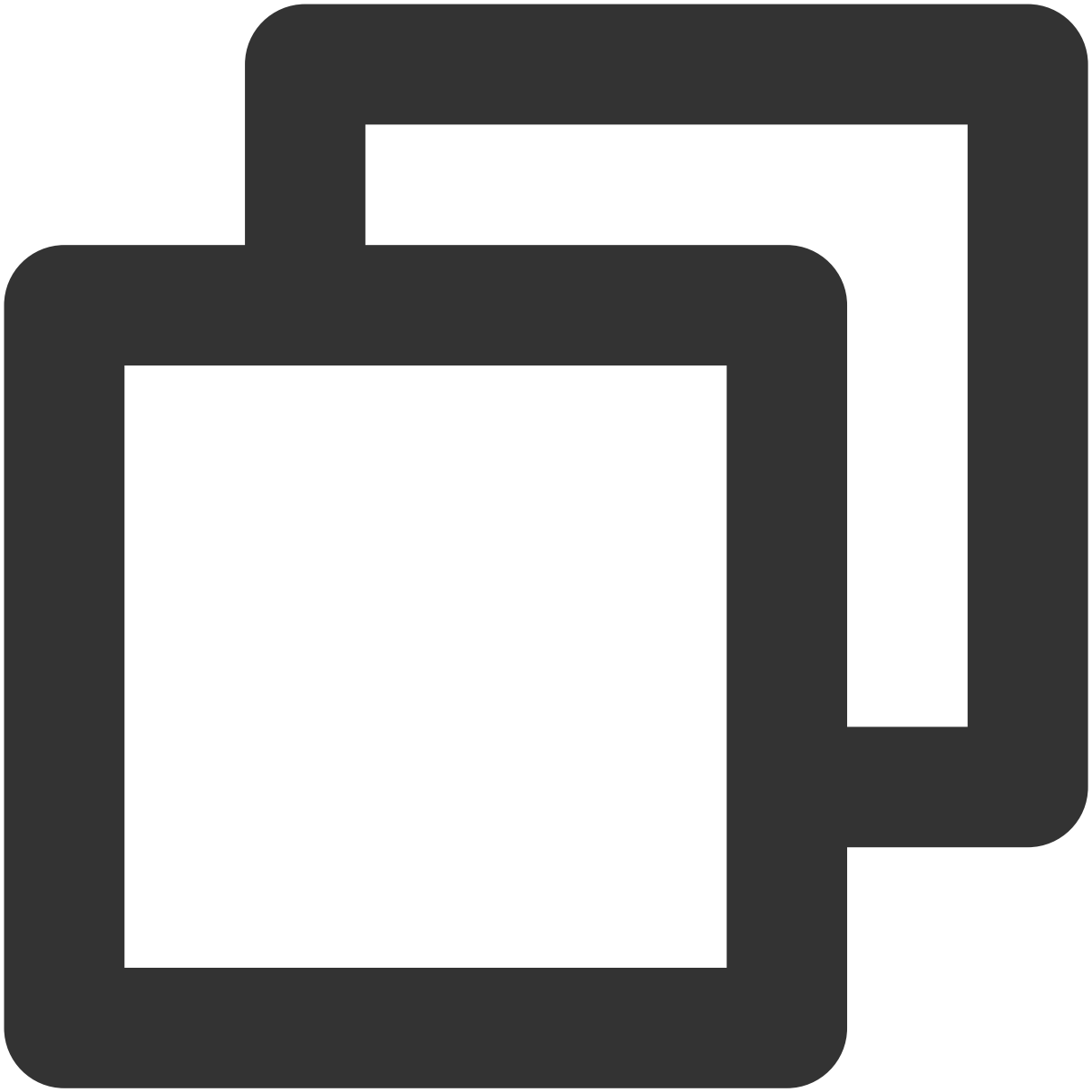


```
npm i @tencentcloud/call-uikit-vue
```

步骤3：引入 TUICallKit 组件

在 main.ts 中，仅需添加两行代码，即可体验通话功能。

引入 TUICallKit，并将其挂载到 TUIKit 中。

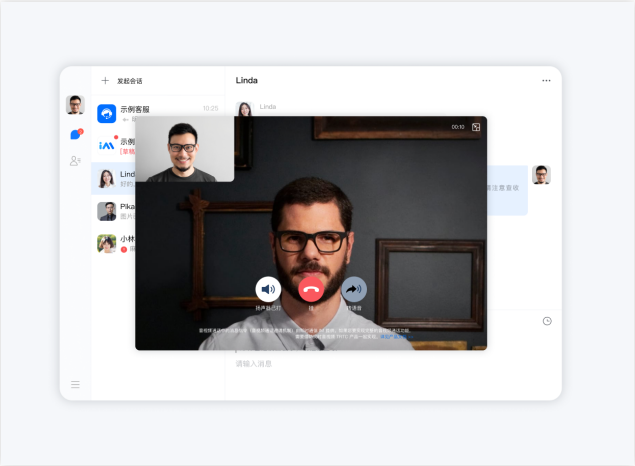


```
// import TUICallKit
import { TUICallKit } from '@tencentcloud/call-uikit-vue';
// TUIKit add TUICallKit
TUIKit.use(TUICallKit);
```

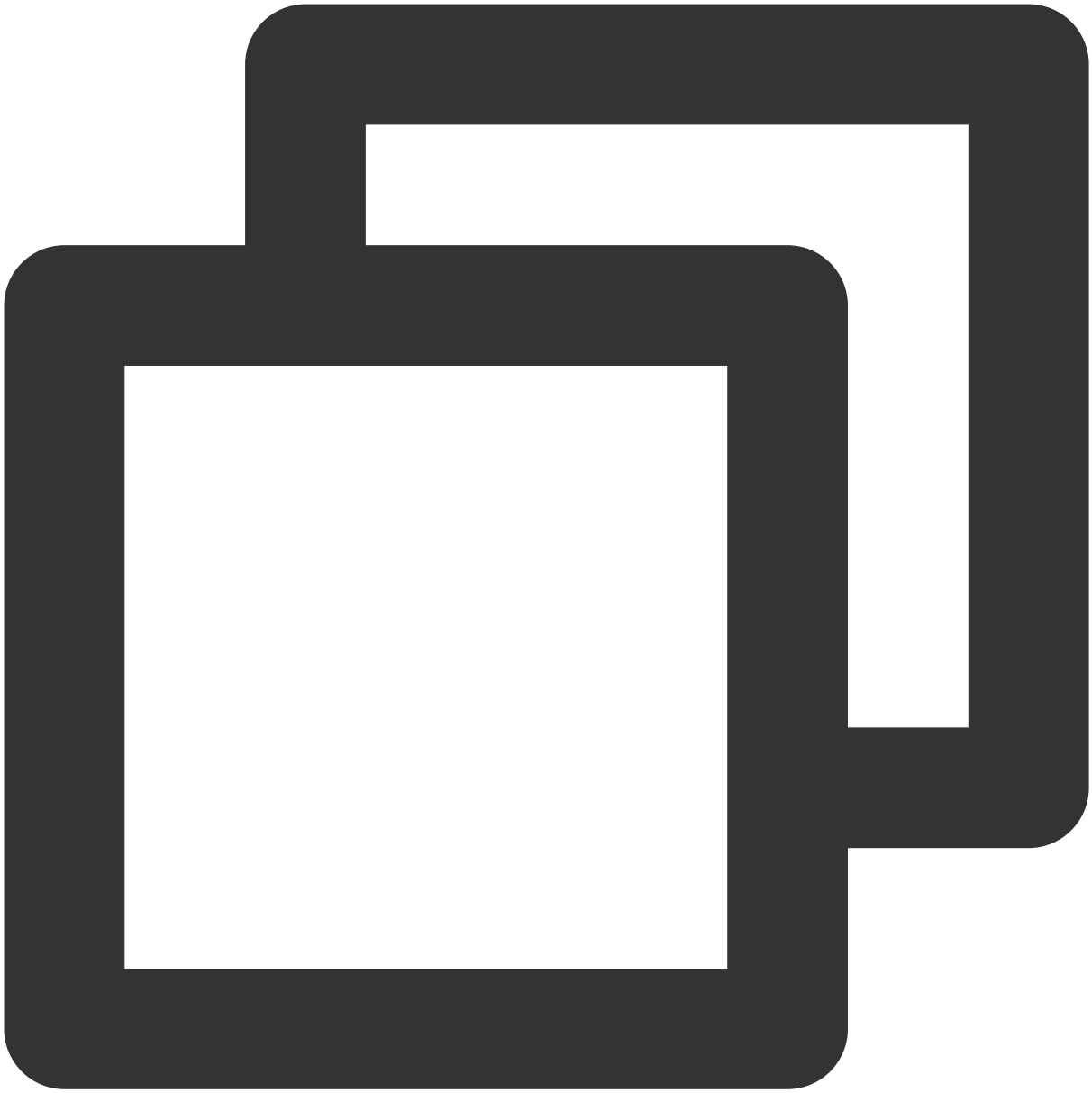
步骤4：调用 TUICallKit 组件

注意：

TUICallKit 组件需要放到一个 dom 节点里，用于显示并控制 TUICallKit 的位置、宽高等样式。
在需要展示的页面，调用 TUICallKit 的组件即可使用。TUICallKit 主要组件包括：

`<TUICallKit/>` ：通话 UI 组件主体	`<TUICallKitMini/>` ：通话 UI 悬浮窗能
	

例如：在 App.vue 页面中，在上文已搭建的聊天界面基础上，使用 TUICallKit 快速搭建通话界面。



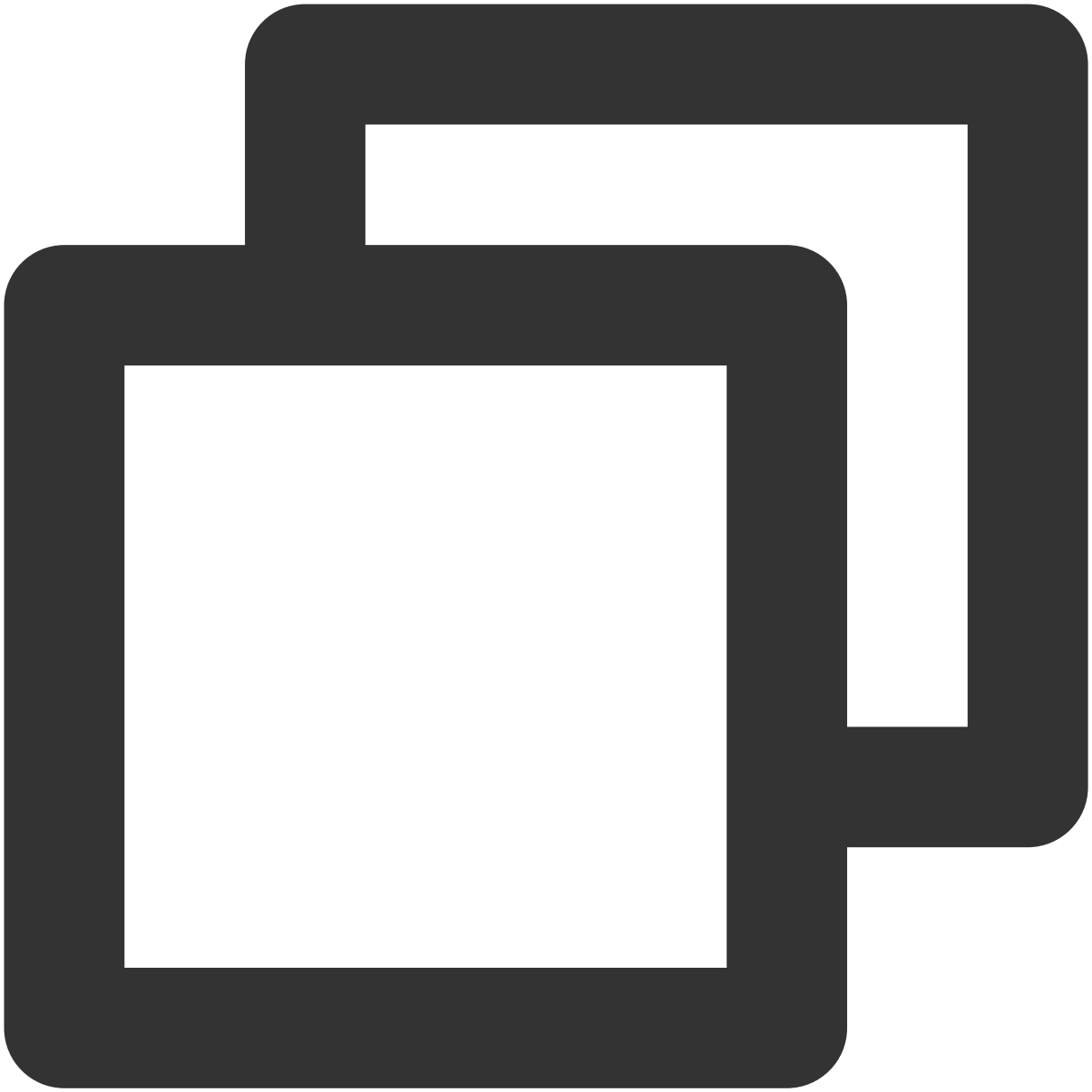
```
<template>
  <div class="home-TUIKit-main">
    <div
      :class="env?.isH5 ? 'conversation-h5' : 'conversation'"
      v-show="!env?.isH5 || currentModel === 'conversation'"
    >
      <TUISearch class="search" />
      <TUIConversation @current="handleCurrentConversation" />
    </div>
    <div class="chat" v-show="!env?.isH5 || currentModel === 'message'">
      <TUIChat>
```

```
        <h1>欢迎使用腾讯云即时通信IM</h1>
      </TUIChat>
    </div>
    <!-- TUICallKit 组件：通话 UI 组件主体 -->
    <TUICallKit
      :class="!showCallMini ? 'callkit-drag-container' : 'callkit-drag-container-mi
      :allowedMinimized="true"
      :allowedFullScreen="false"
      :beforeCalling="beforeCalling"
      :afterCalling="afterCalling"
      :onMinimized="onMinimized"
      :onMessageSentByMe="onMessageSentByMe"
    />
  </div>
</template>

<script lang="ts">
import { defineComponent, reactive, toRefs } from "vue";
import { TUIEnv } from "./TUIKit/TUIPlugin";
import { handleErrorPrompts } from "./TUIKit/TUIComponents/container/utils";

export default defineComponent({
  name: "App",
  setup() {
    const data = reactive({
      env: TUIEnv(),
      currentModel: "conversation",
      showCall: false,
      showCallMini: false,
    });
    const TUIServer = (window as any)?.TUIKitTUICore?.TUIServer;
    const handleCurrentConversation = (value: string) => {
      data.currentModel = value ? "message" : "conversation";
    };
    // beforeCalling：在拨打电话前与收到通话邀请前执行
    const beforeCalling = (type: string, error: any) => {
      if (error) {
        handleErrorPrompts(error, type);
        return;
      }
      data.showCall = true;
    };
    // afterCalling：结束通话后执行
    const afterCalling = () => {
      data.showCall = false;
      data.showCallMini = false;
    };
```

```
    // onMinimized：组件切换最小化状态时执行
    const onMinimized = (
      oldMinimizedStatus: boolean,
      newMinimizedStatus: boolean
    ) => {
      data.showCall = !newMinimizedStatus;
      data.showCallMini = newMinimizedStatus;
    };
    // onMessageSentByMe：在整个通话过程内发送消息时执行
    const onMessageSentByMe = async (message: any) => {
      TUIServer?.TUIChat?.handleMessageSentByMeToView(message);
      return;
    };
    return {
      ...toRefs(data),
      handleCurrentConversation,
      beforeCalling,
      afterCalling,
      onMinimized,
      onMessageSentByMe,
    };
  },
});
</script>
<style scoped>
.home-TUIKit-main {
  display: flex;
  height: 100vh;
  overflow: hidden;
}
.search {
  padding: 12px;
}
.conversation {
  min-width: 285px;
  flex: 0 0 24%;
  border-right: 1px solid #f4f5f9;
}
.conversation-h5 {
  flex: 1;
  border-right: 1px solid #f4f5f9;
}
.chat {
  flex: 1;
  height: 100%;
  position: relative;
}
```

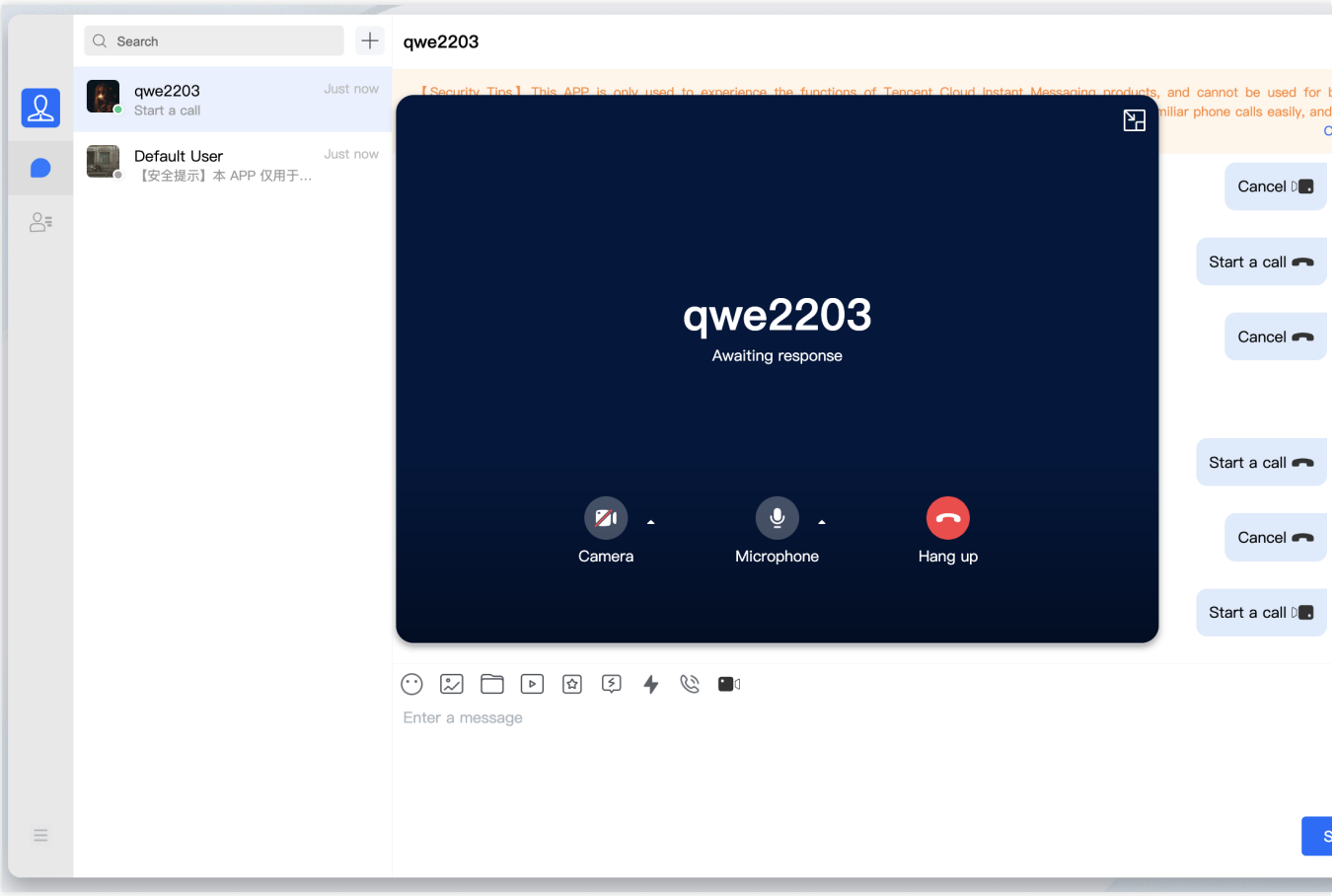
```
.callkit-drag-container {
  position: fixed;
  left: calc(50% - 25rem);
  top: calc(50% - 18rem);
  width: 50rem;
  height: 36rem;
  border-radius: 16px;
  box-shadow: rgba(0, 0, 0, 0.16) 0px 3px 6px, rgba(0, 0, 0, 0.23) 0px 3px 6px;
}
.callkit-drag-container-mini {
  position: fixed;
  width: 168px;
  height: 56px;
  right: 10px;
  top: 70px;
}
</style>
```

步骤5：启动项目



```
npm run serve
```

步骤6：进行您的第一次通话



技术咨询

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

Flutter

最近更新时间：2024-03-20 16:54:30

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，跟随本文档，您可以在 10 分钟内完接入工作，并最终得到一个包含完备 UI 界面的音视频通话功能。

环境准备

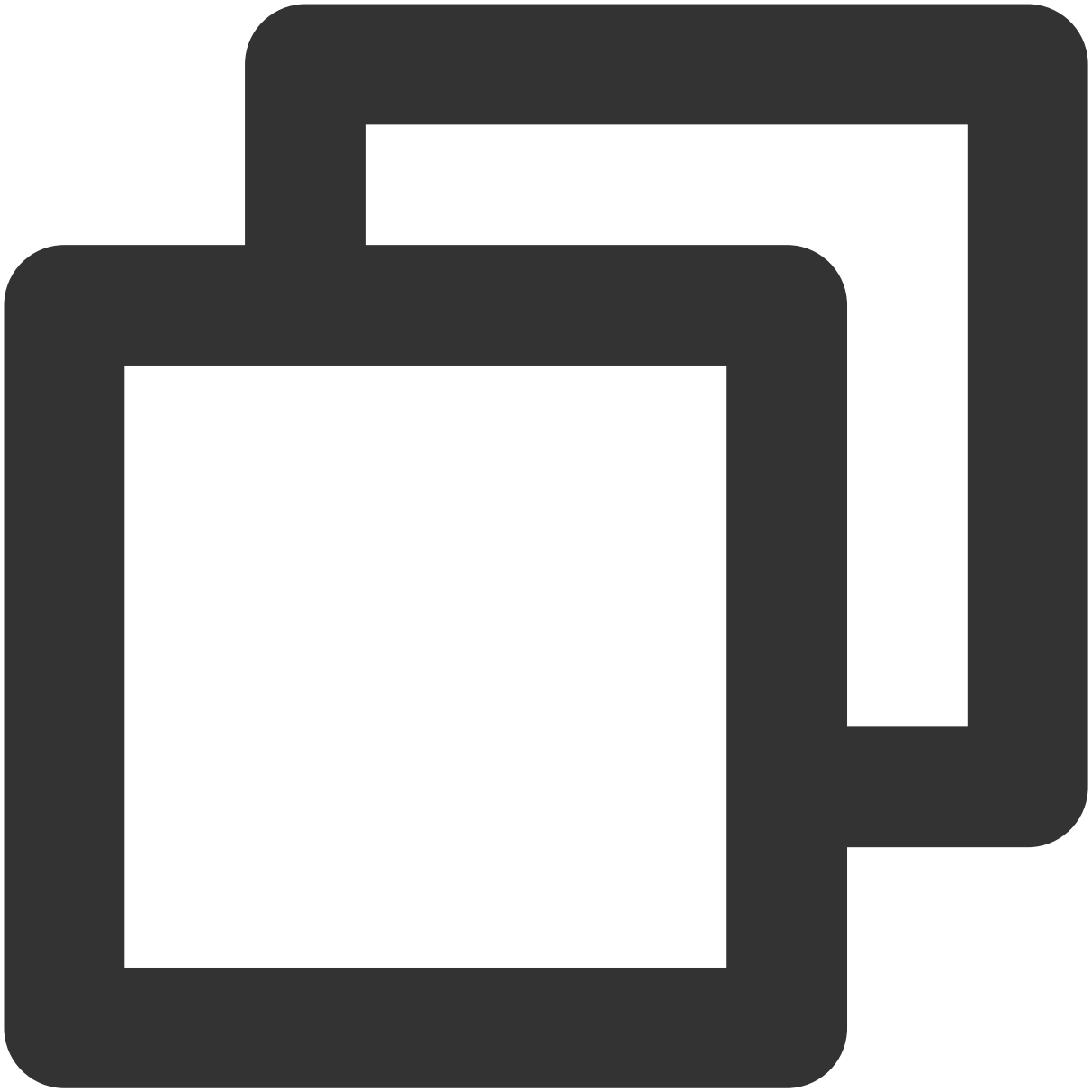
Flutter 3.0及更高版本。

步骤1：开通服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。为了您能更好地体验音视频通话 Call 功能，我们免费为每个 SDKAppID 提供了7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7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。
您可以在 即时通信 IM 控制台中开通 Call 体验版，具体操作步骤如下：
1. 访问 IM 控制台，选择数据中心并新建应用。若您已有应用可跳过该步骤。
2. 单击目标应用卡片，进入应用的基础配置页面。
3. 找到**音视频通话能力（Call）**卡片，单击**免费体验。**
4. 确认弹窗内容后，单击**立即体验**，即可成功开通音视频通话体验版。开通完成后即可根据本文档进行集成。
5. 如果您的业务上线，需要购买正式版本，可前往控制台进行购买，请参考：购买正式版。

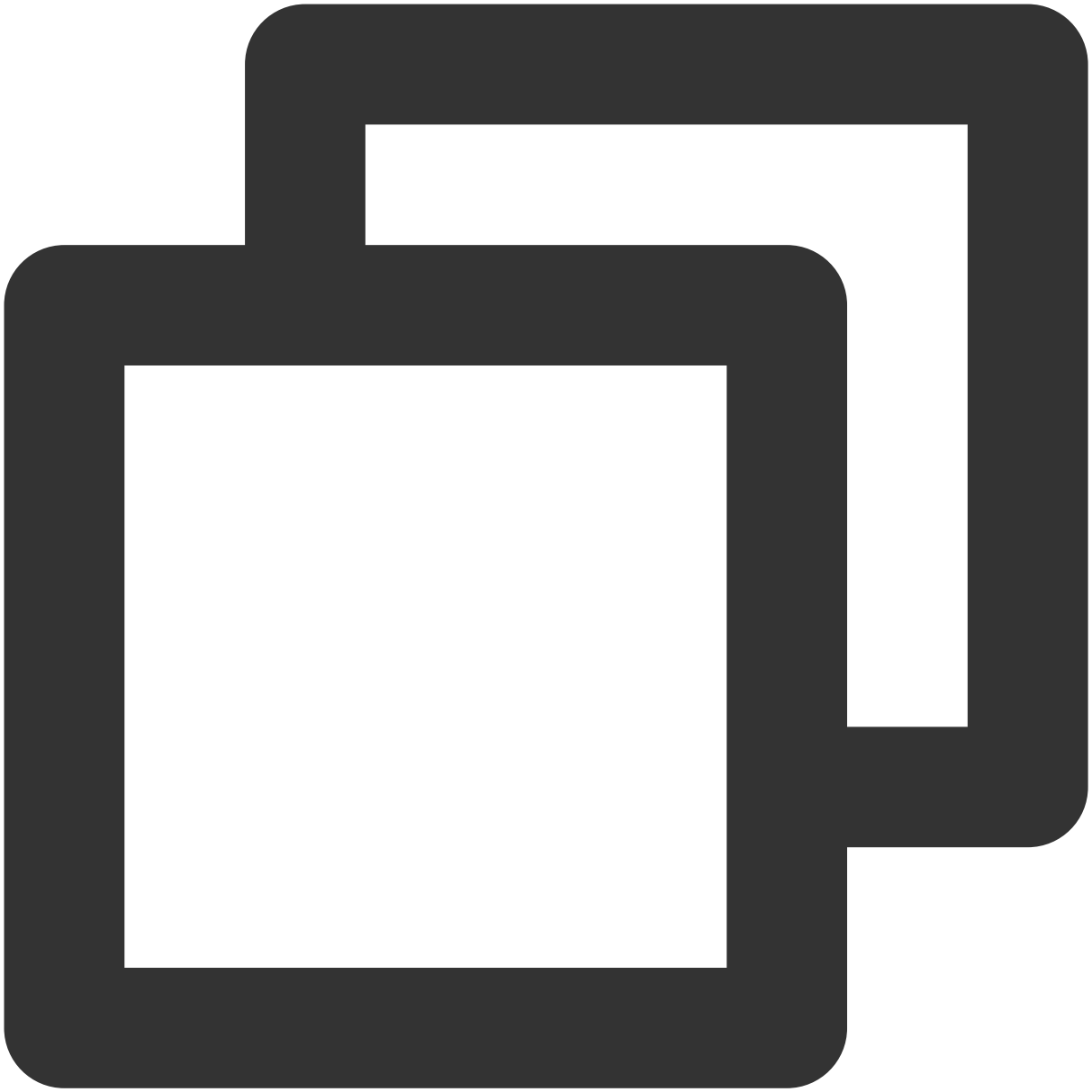
步骤2：导入TUICallKit组件

使用 Flutter 导入组件，具体步骤如下：
1. 在您的 pubspec.yaml 文件中添加 tencent_calls_uikit 插件依赖，点击插件超链接切换到Versions目录下查询最新版本。



```
dependencies:
  tencent_calls_uikit: 最新版本
```

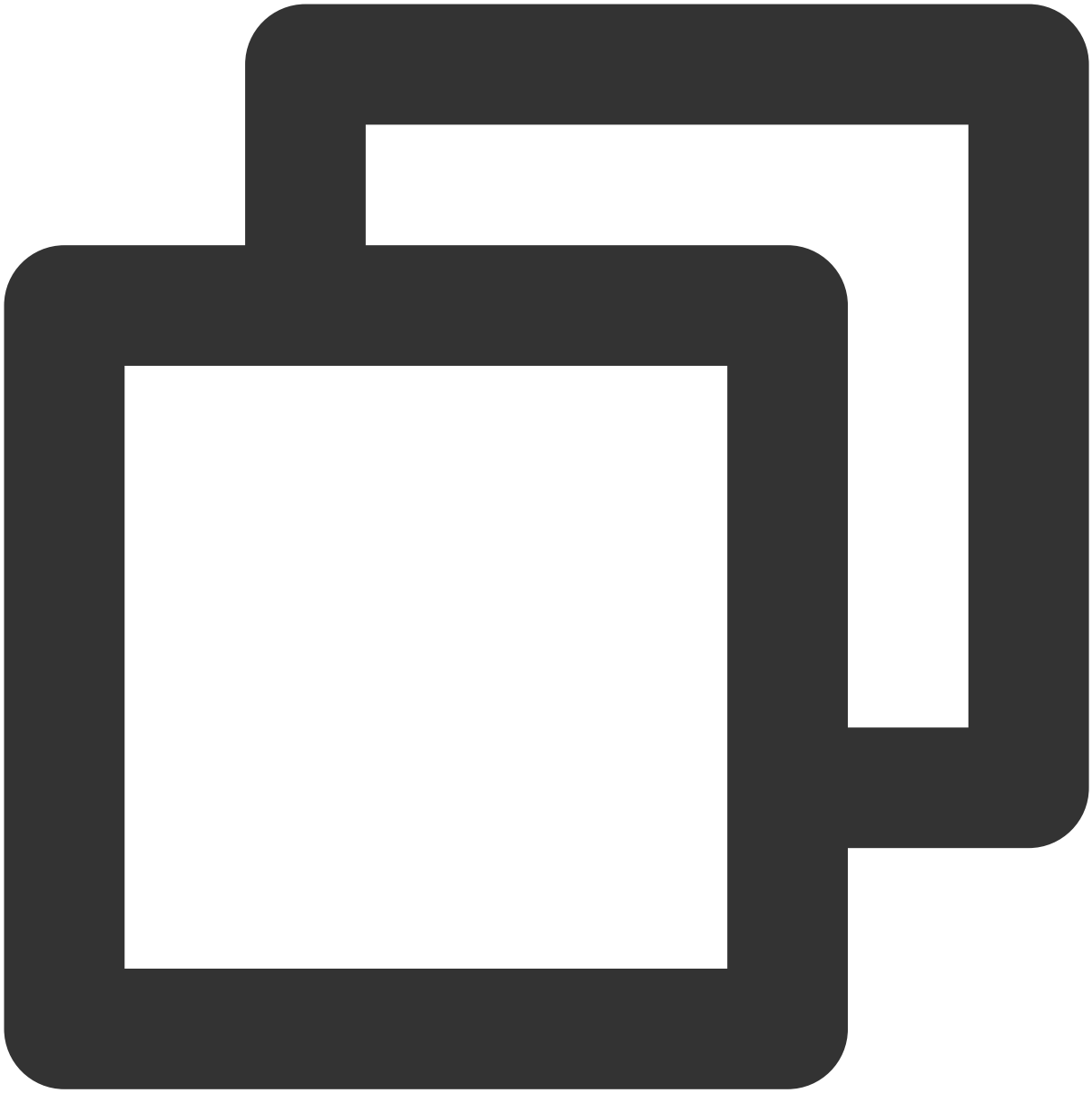
2. 执行以下命令安装组件：



```
flutter pub get
```

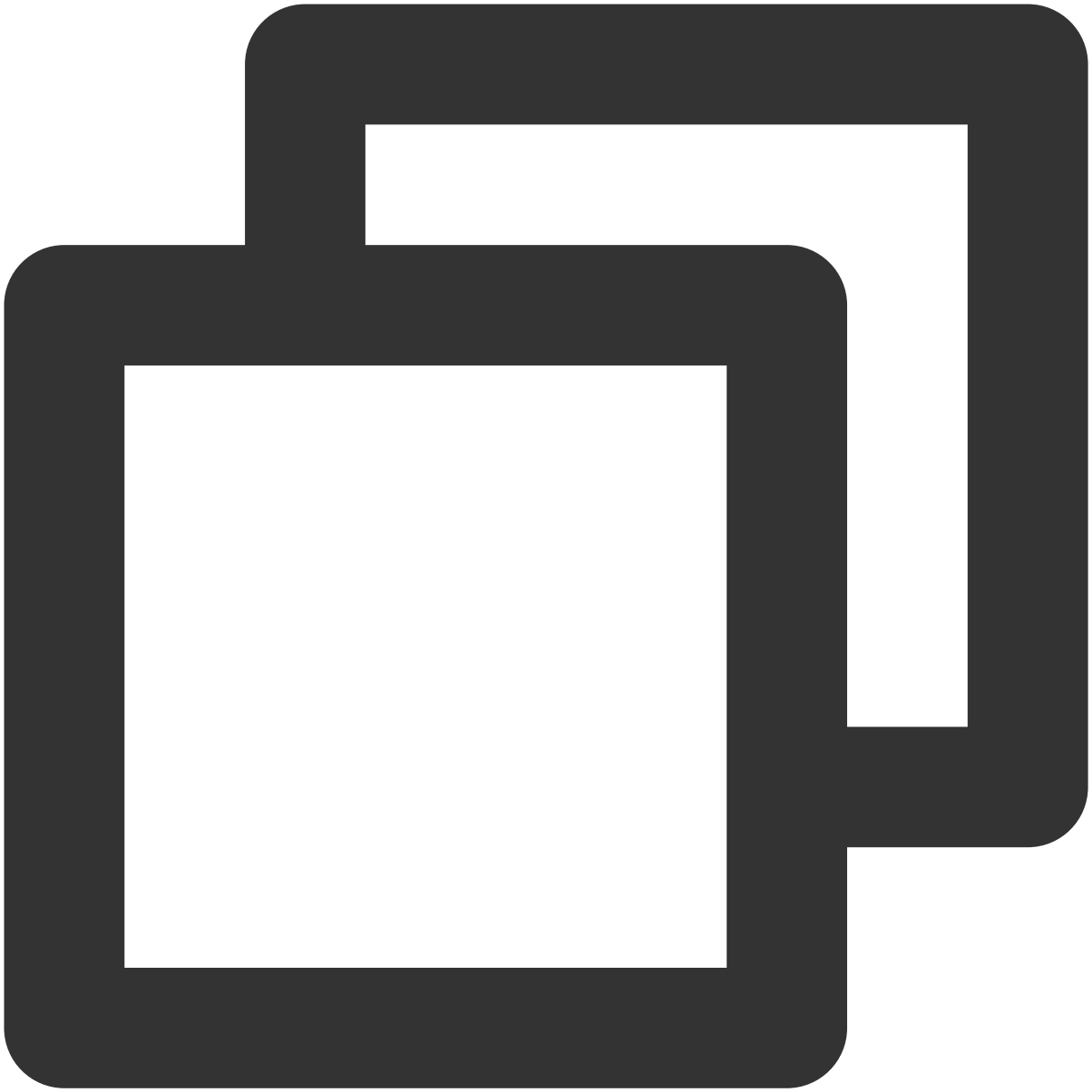
步骤3：完成工程配置

1. 将 TUICallKit的navigatorObserver 添加到 App组件中，以MateriaApp为例，代码如下：



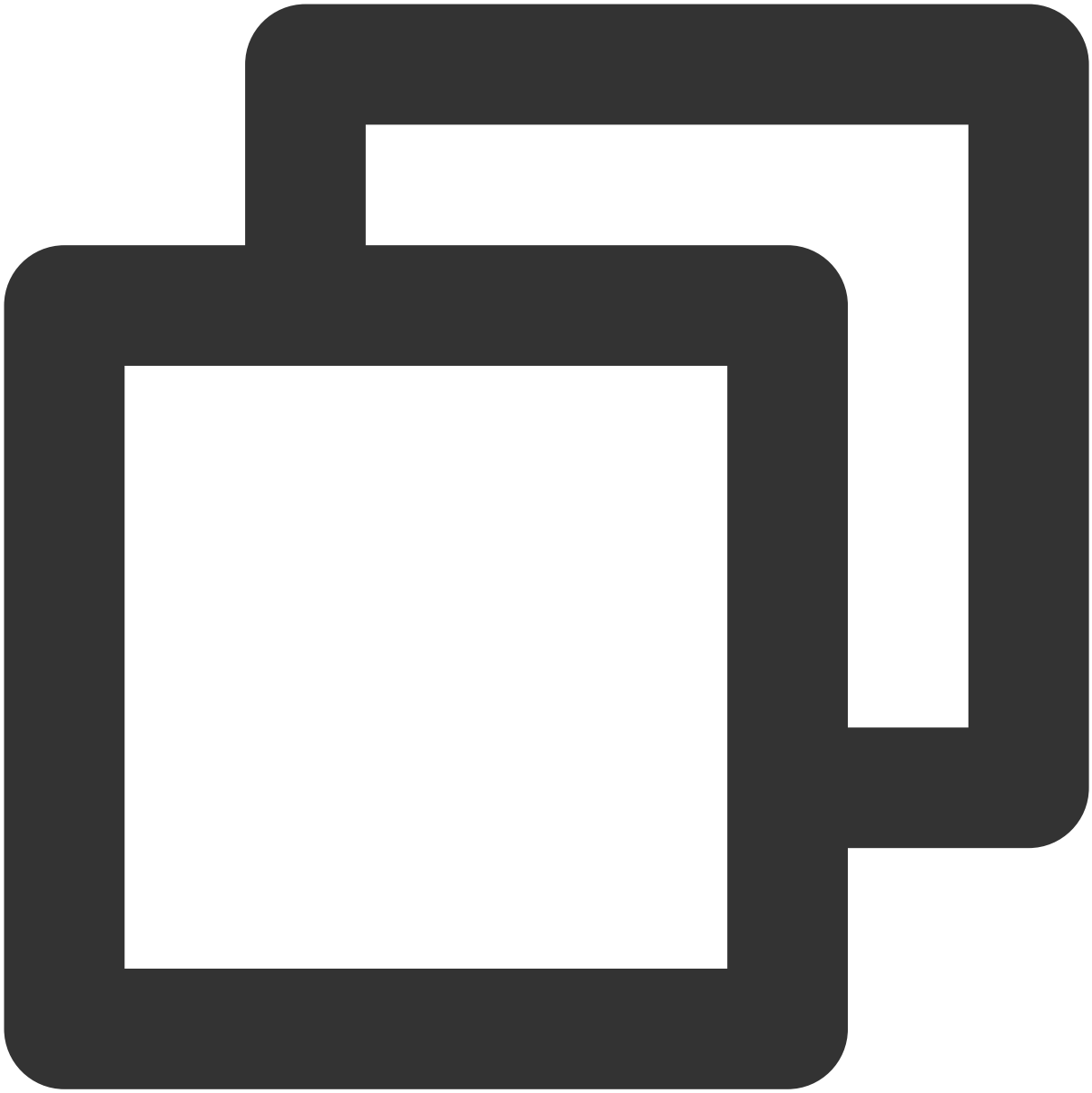
```
import 'package:tencent_calls_uikit/tuicall_kit.dart';

MaterialApp (
     navigatorObservers：[TUICallKit.navigatorObserver],
     ...
)
```

2. 如果您需要编译运行在 Android 平台，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.** { *; }
```

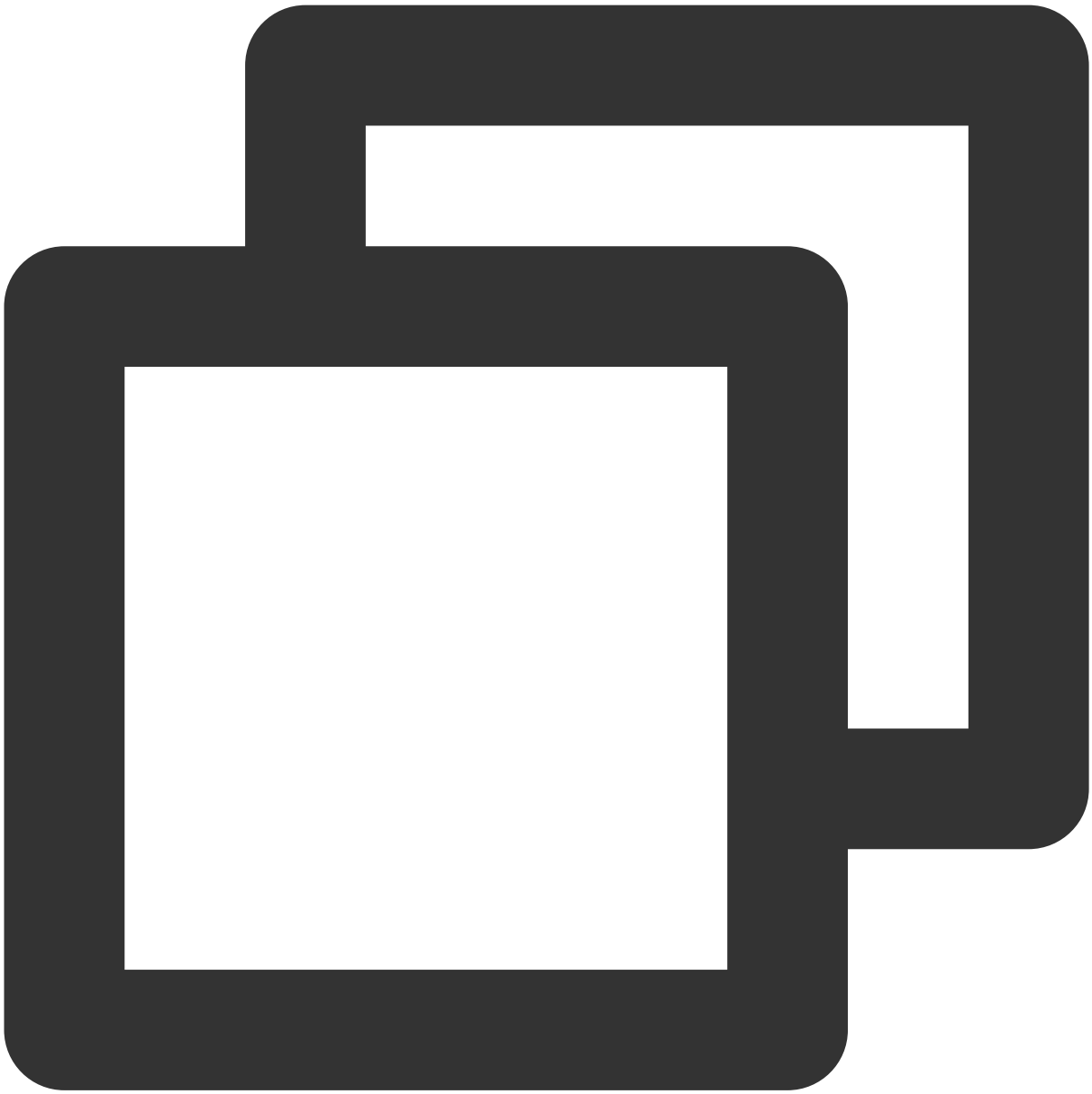
3. 如果您的工程需要在iOS 模拟器上调试，您需要在工程的 `/ios/Podfile` 文件中添加以下代码：



```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['VALID_ARCHS'] = 'arm64 arm64e x86_64'
      config.build_settings['VALID_ARCHS[sdk=iphonesimulator*]'] = 'x86_64'
    end
  end
end
```

4. 如果您需要使用 iOS 的音视频功能，您需要授权麦克风和摄像头的使用权限。

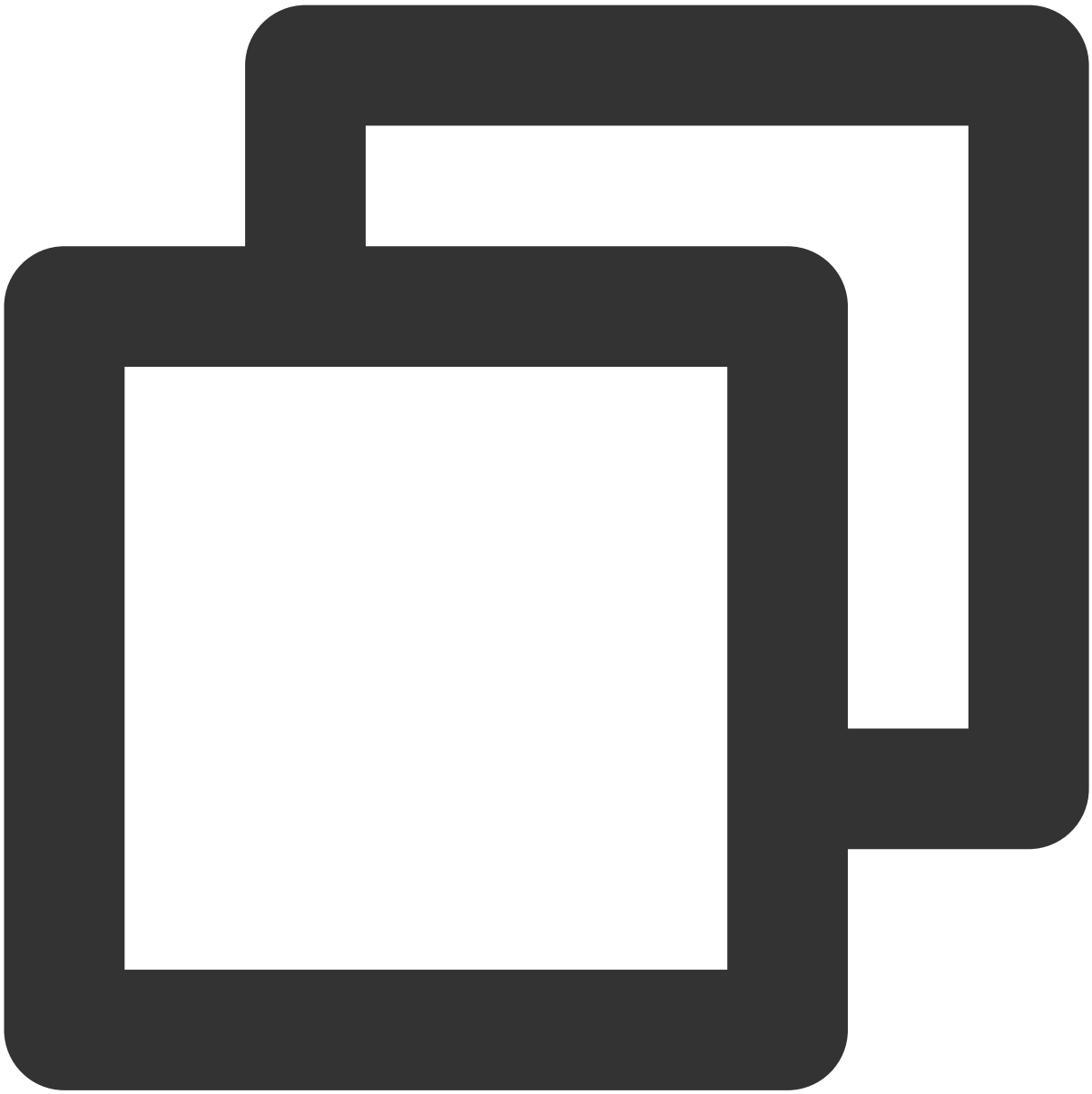
授权操作方法：在您的 iOS 工程的 Info.plist 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。



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

步骤4：登录 TUICallKit 组件

在您的项目中添加如下代码，这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：



```
TUIResult result = TUICallKit.instance.login(SDKAppID,        // 请替换为第一步得到的SDI
                                             'userId',        // 请替换为您的User ID
                                             'userSig');      // 您可以在控制台计算一个u
```

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 步骤一 的第 2 步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 辅助工具 生成一个临时可用的 UserSig。

更多信息请参见 如何计算及使用 UserSig。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。

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

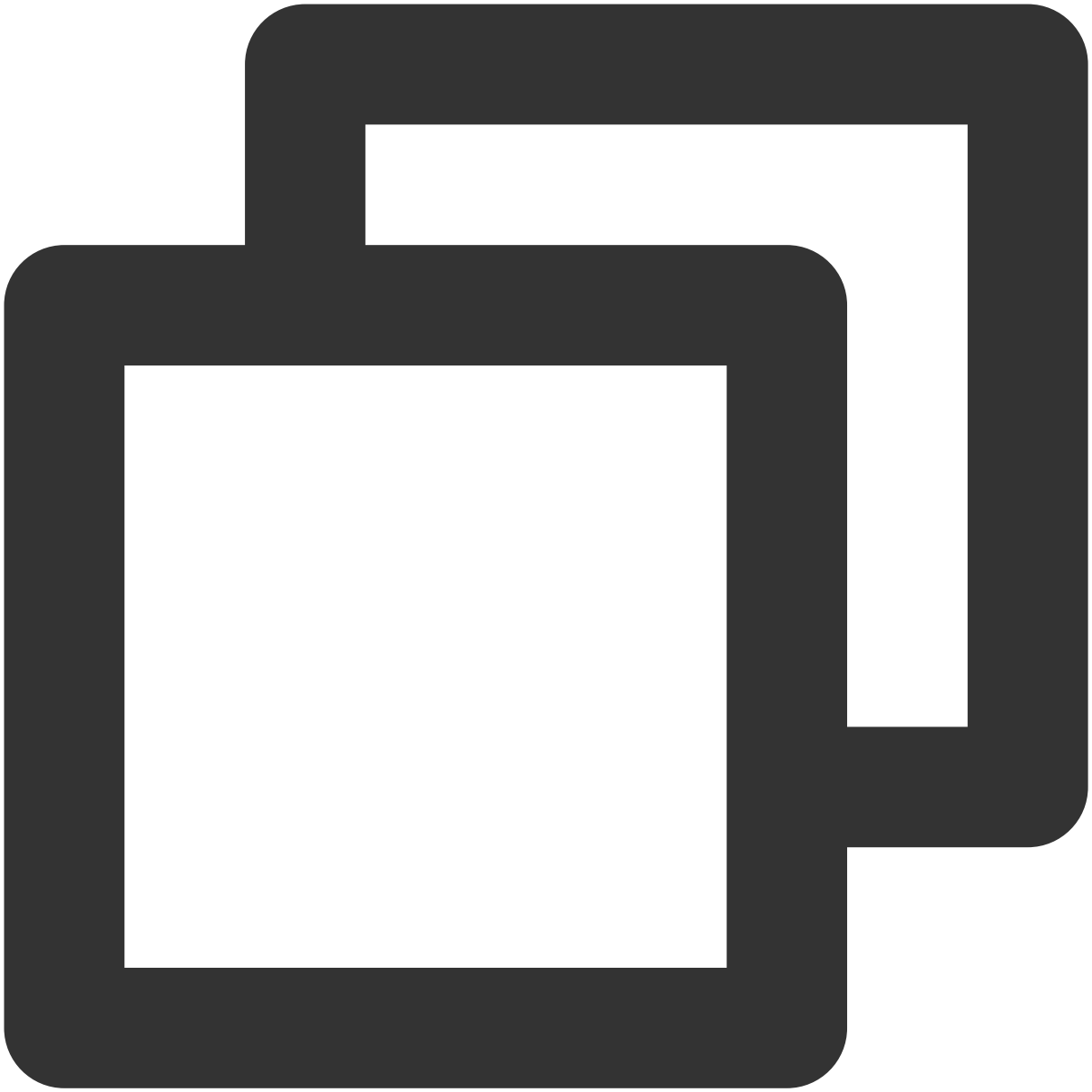
UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。

Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 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 TUICallKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤5：拨打通话

1对1视频通话

通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

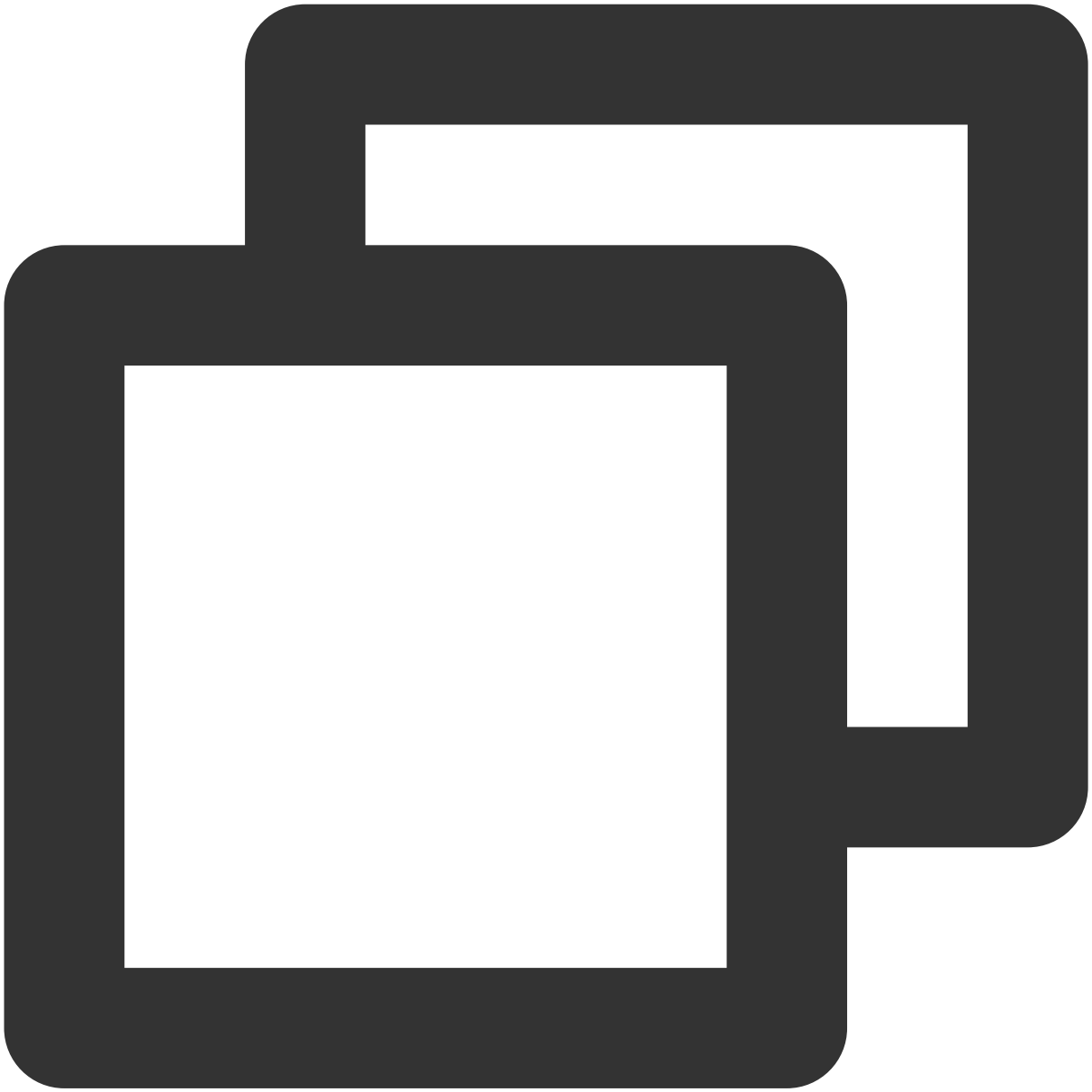


```
// 假设给mike拨打视频电话
TUICallKit.instance.call('mike', TUICallMediaType.video);
```

参数	类型	含义
userId	String	目标用户的 UserID： `"mike"`
callMediaType	TUICallMediaType	通话的媒体类型，示例： `TUICallMediaType.video` 或 `TUICallMediaType.audio`

群内视频通话

通过调用 TUICallKit 的 groupCall 函数并指定通话类型和被叫方的 UserID 列表，就可以发起群内的语音或者视频通话。



```
// 假设在groupId:0001的群组内，向denny，mike 和 tommy 发起视频通话
TUICallKit.instance.groupCall('0001', ['denny', 'mike', 'tommy'], TUICallMediaType.
```

参数	类型	含义
groupId	String	群组 ID

userIdList	Array	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

说明

群组的创建详见：IM 群组管理，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈：TUIKit 需求收集表。

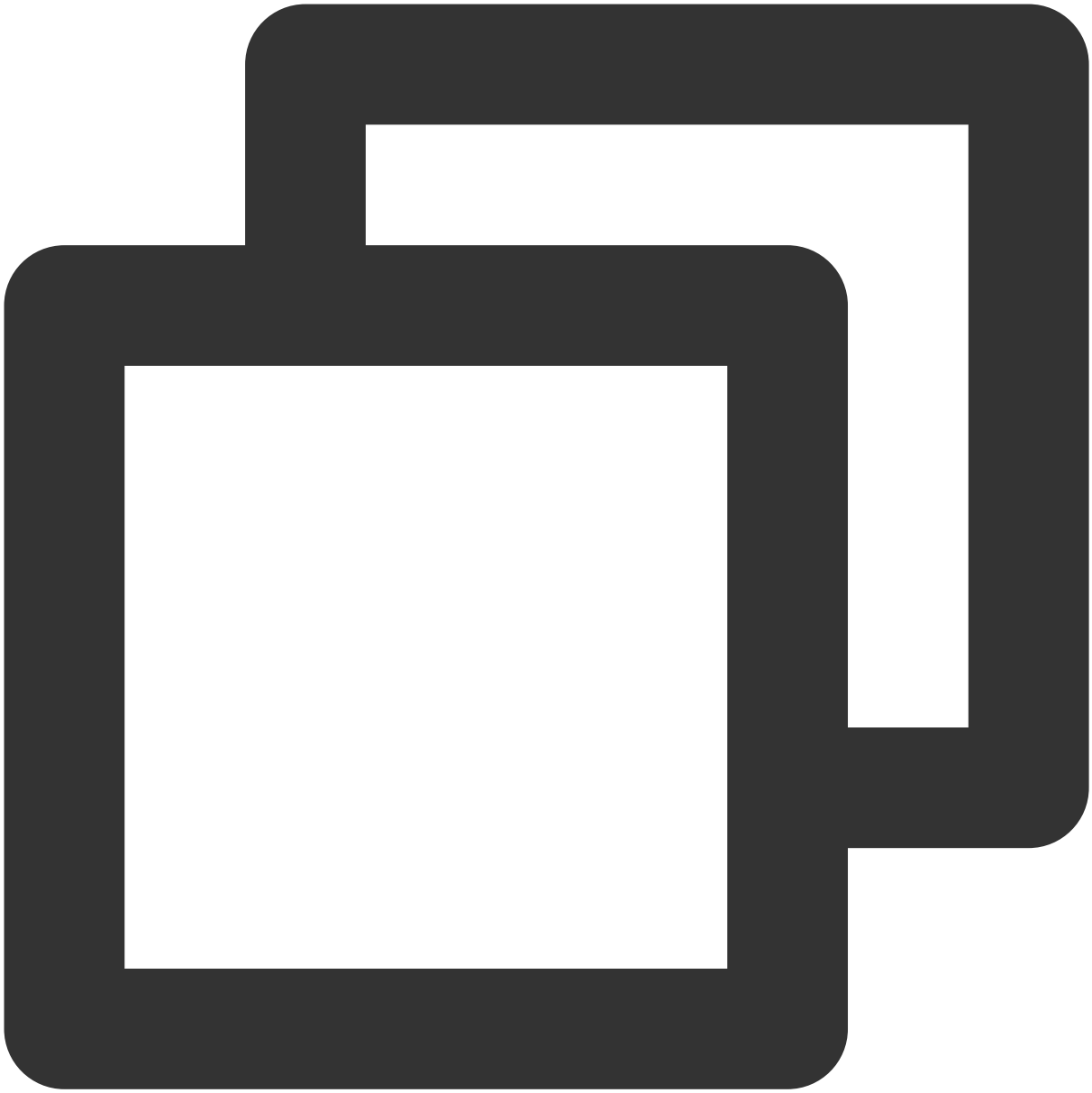
步骤6：接听通话

在 步骤四 完成后，收到来电请求后，TUICallKit 组件会自动启动相应的接听界面。

步骤7：更多特性

一、设置昵称&头像

如果您需要自定义昵称或头像，可以使用如下接口进行更新：



```
// 函数定义为: Future<TUIResult> setSelfInfo(String nickname, String avatar)
TUIResult result = TUICallKit.instance.setSelfInfo('userName', 'url:********');
```

注意

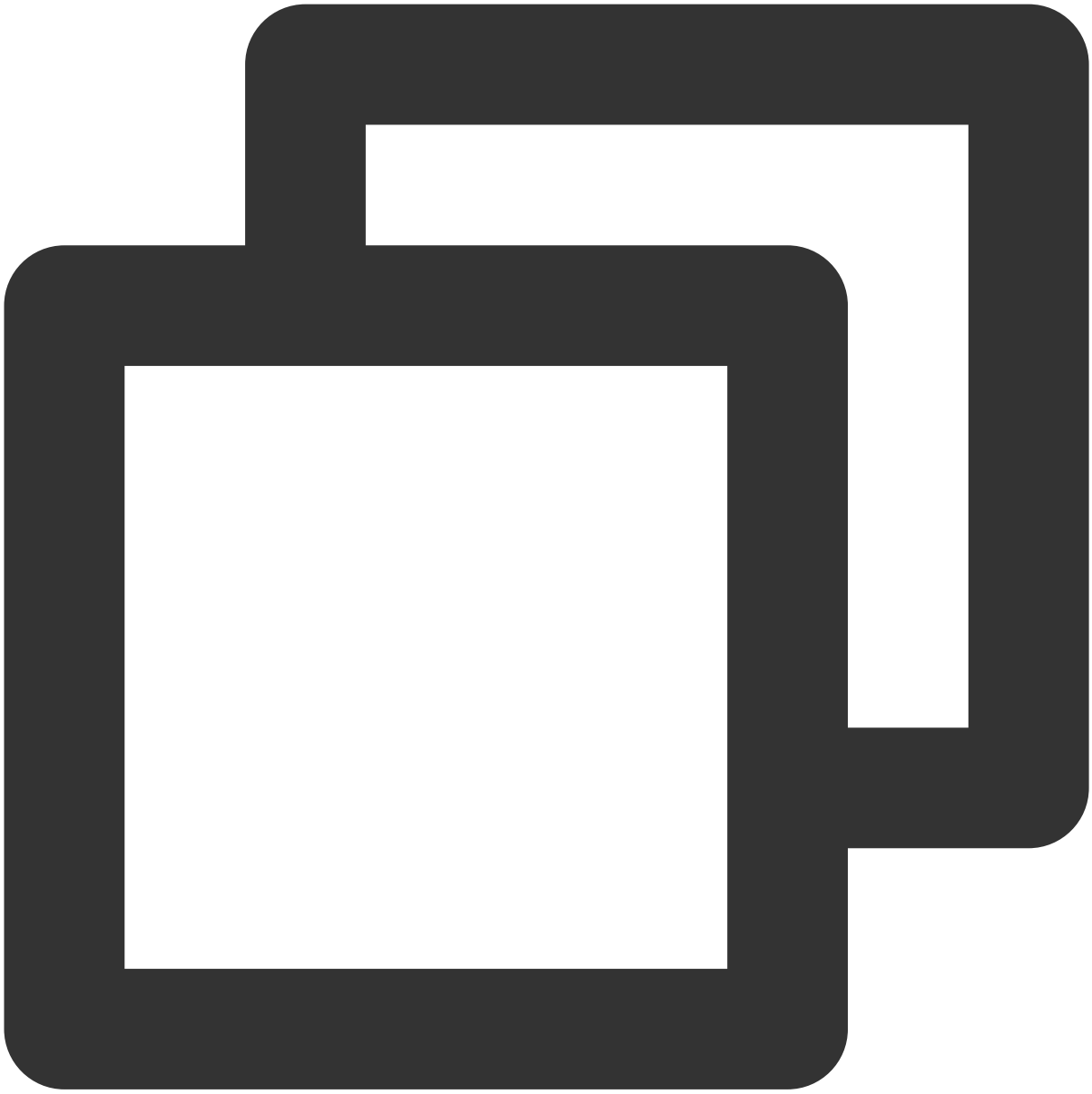
因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

二、离线唤醒

完成以上步骤，就可以实现音视频通话的拨打和接通，但如果您的业务场景需要在 `应用的进程被杀死后` 或者 `应用退到后台后` ，还可以正常接收到音视频通话请求，就需要增加离线唤醒功能，详情见 离线唤醒（Flutter）。

三、悬浮窗功能

如果您的业务需要开启悬浮窗功能，您可以在 TUICallKit 组件初始化时调用以下接口开启该功能：



```
TUICallKit.instance.enableFloatWindow(true);
```

四. 通话状态监听

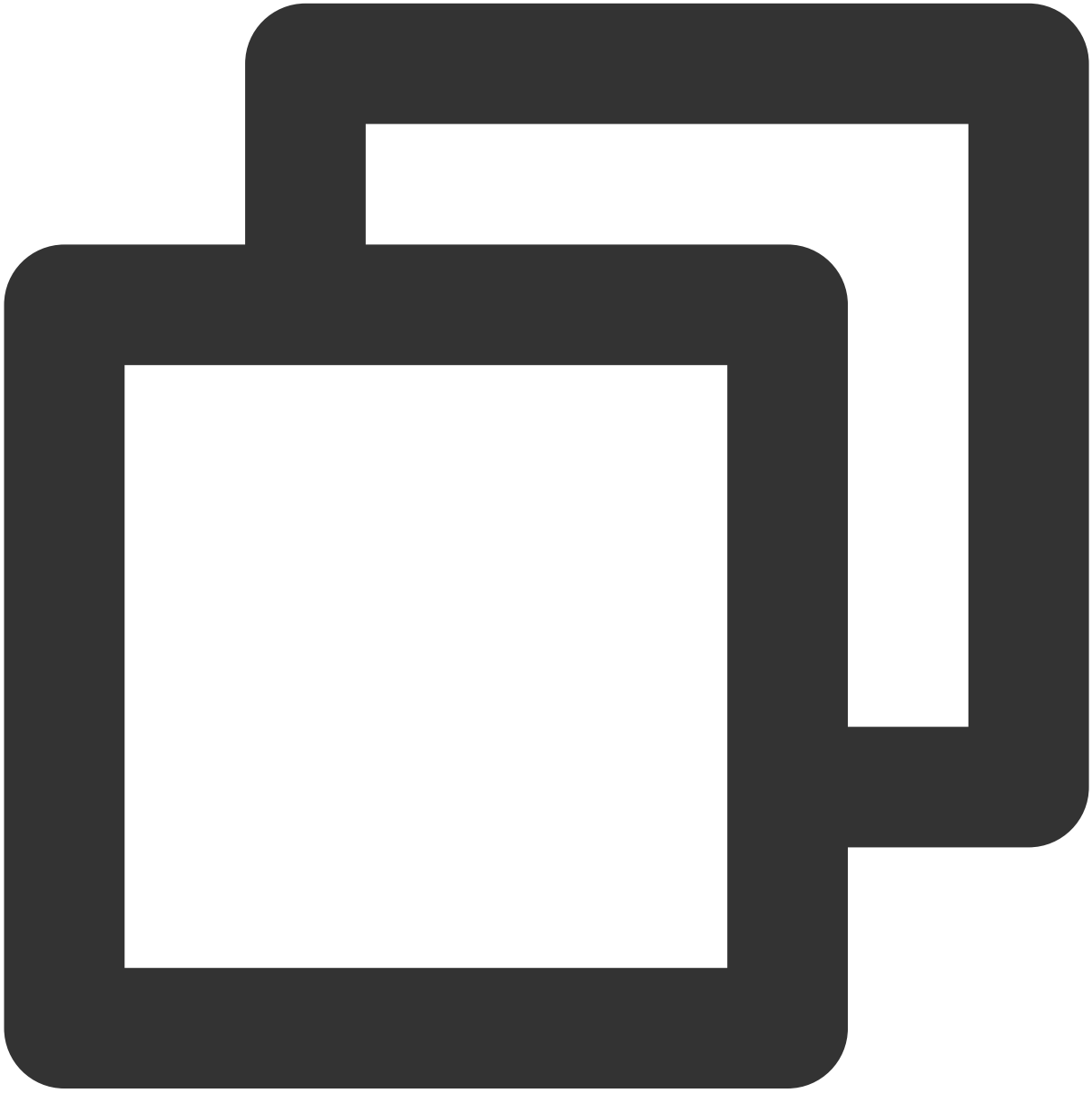
如果您的业务需要 **监听通话的状态**，例如通话开始、结束，以及通话过程中的网络质量等等，可以监听以下事件：

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {
      //您的回调处理逻辑
    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole
      //您的回调处理逻辑
    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
      //您的回调处理逻辑
    },, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityLis
      //您的回调处理逻辑
    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,
      //您的回调处理逻辑
```

```
    }
));
```

五、自定义铃音

如果您需要自定义来电铃音，可以通过如下接口进行设置：



```
TUICallKit.instance.setCallingBell('flie path');
```

常见问题

1、错误提示“The package you purchased does not support this ability”？

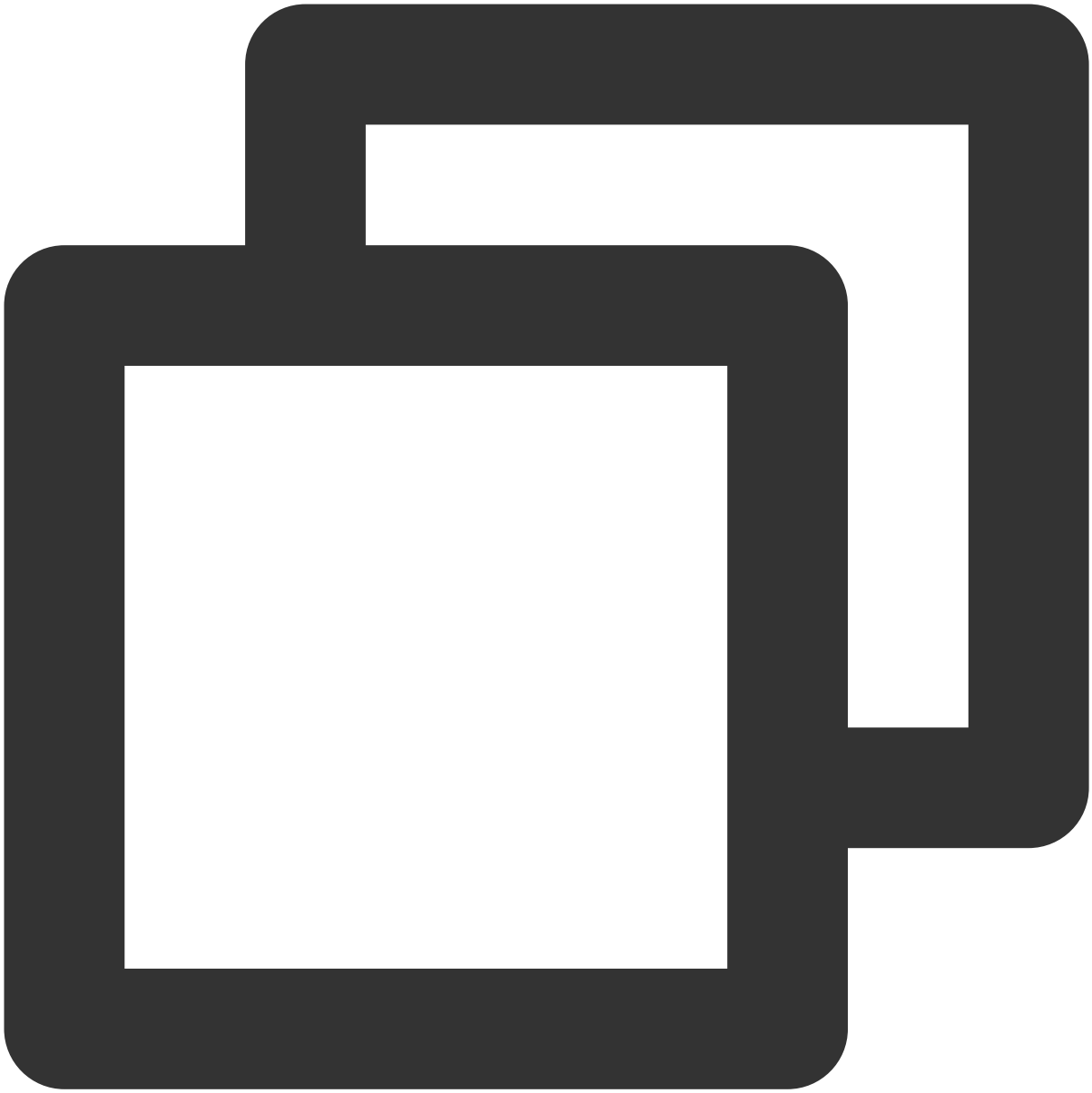
如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，见步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2、如何购买套餐？

请参考购买链接 购买正式版。

3、Android 编译报错：uses-sdk:minSdkVersion 16 cannot be smaller than version 19.

详细的报错信息如下：



```
uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:t
Suggestion: use a compatible library with a minSdk of at most 16,
                or increase this project's minSdk version to at least 19,
                or use tools:overrideLibrary="com.tencent.cloud.tuikit.flutter.tuic
```

这是因为 `tencent_call_uikit` Plugin Android 端支持的 minSdkVersion 是 19，可以将您工程中的 `android/app/build.gradle` 文件中的 `minSdkVersion` 修改为 19 及其以上即可。

交流与反馈

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

界面定制（TUICallKit）
Android

最近更新时间：2024-05-15 16:54:16

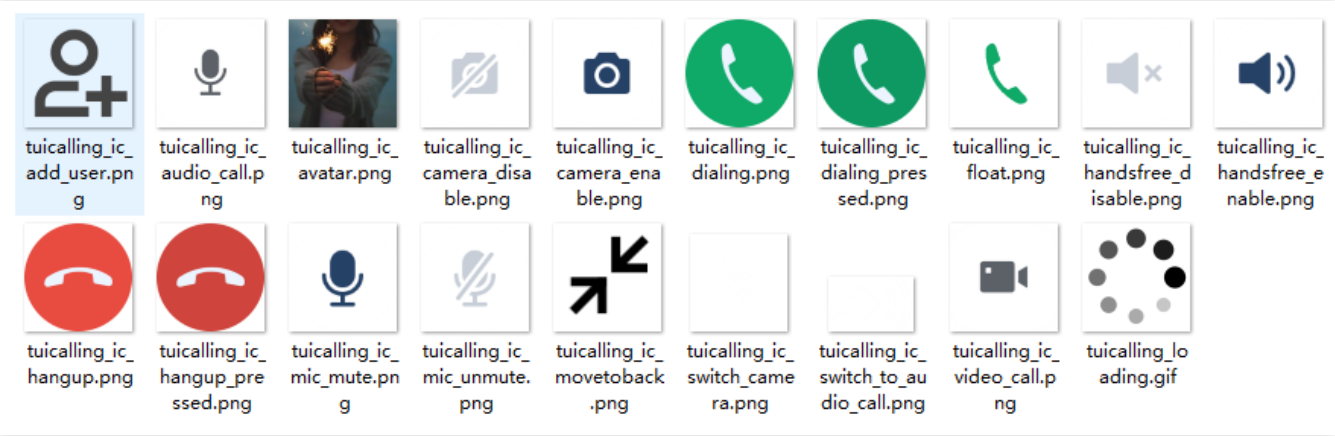
本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUICallKit 的用户界面进行调整，TUICallKit 的界面源代码位于 Github 中的 `Android/tuicallkit` 文件夹下面：

替换图标

您可以直接替换 `res\\drawable-xxhdpi` 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 `res\\raw` 文件夹下的三个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_hangup.mp3	被挂断的声音
phone_ringing.mp3	接到呼叫时的声音

替换文案

您可以通过修改 values-zh 和 values-en 中的 `strings.xml` 文件来修改视频通话界面中的字符串内容。

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI 组件实现的，您可以删掉 tuicallkit 文件夹，完全基于 TUICallEngine 实现一套自己的 UI 界面。

TUICallEngine

TUICallEngine 是整个通话组件的底层接口，主要提供了1对1音视频通话和群组内通话的发起、接听、拒绝、结束以及设备操作等关键接口。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，例如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头

closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式 （尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调

onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

关键类型定义

API	描述
TUICallDefine.MediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallDefine.Role	通话的角色，枚举类型：主叫、被叫
TUICallDefine.Status	通话的状态，枚举类型：空闲、待接听、接听中
TUICallDefine.CallParams	通话扩展参数：离线推送消息显示内容
TUICommonDefine.RoomId	音视频房间Id，支持数字、字符串两种类型
TUICommonDefine.Camera	摄像头Id参数，枚举类型：前摄、后摄
TUICommonDefine.AudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUICommonDefine.NetworkQualityInfo	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

Web

最近更新时间：2024-08-09 17:45:54

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

方案1：界面微调方案

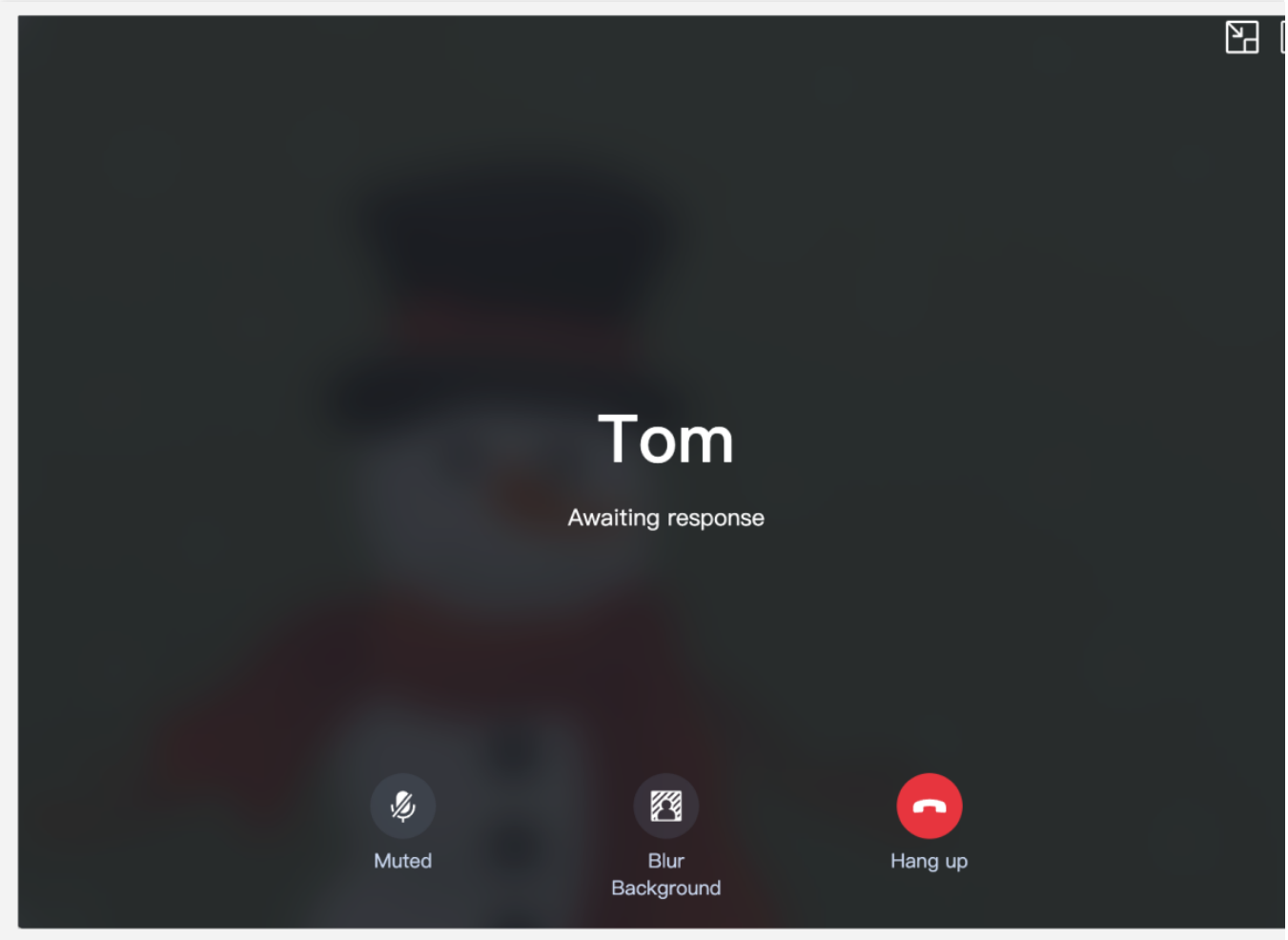
按钮隐藏

调用 hideFeatureButton 接口隐藏按钮，目前支持 Camera，Microphone，SwitchCamera，具体看枚举类型 FeatureButton。

注意：

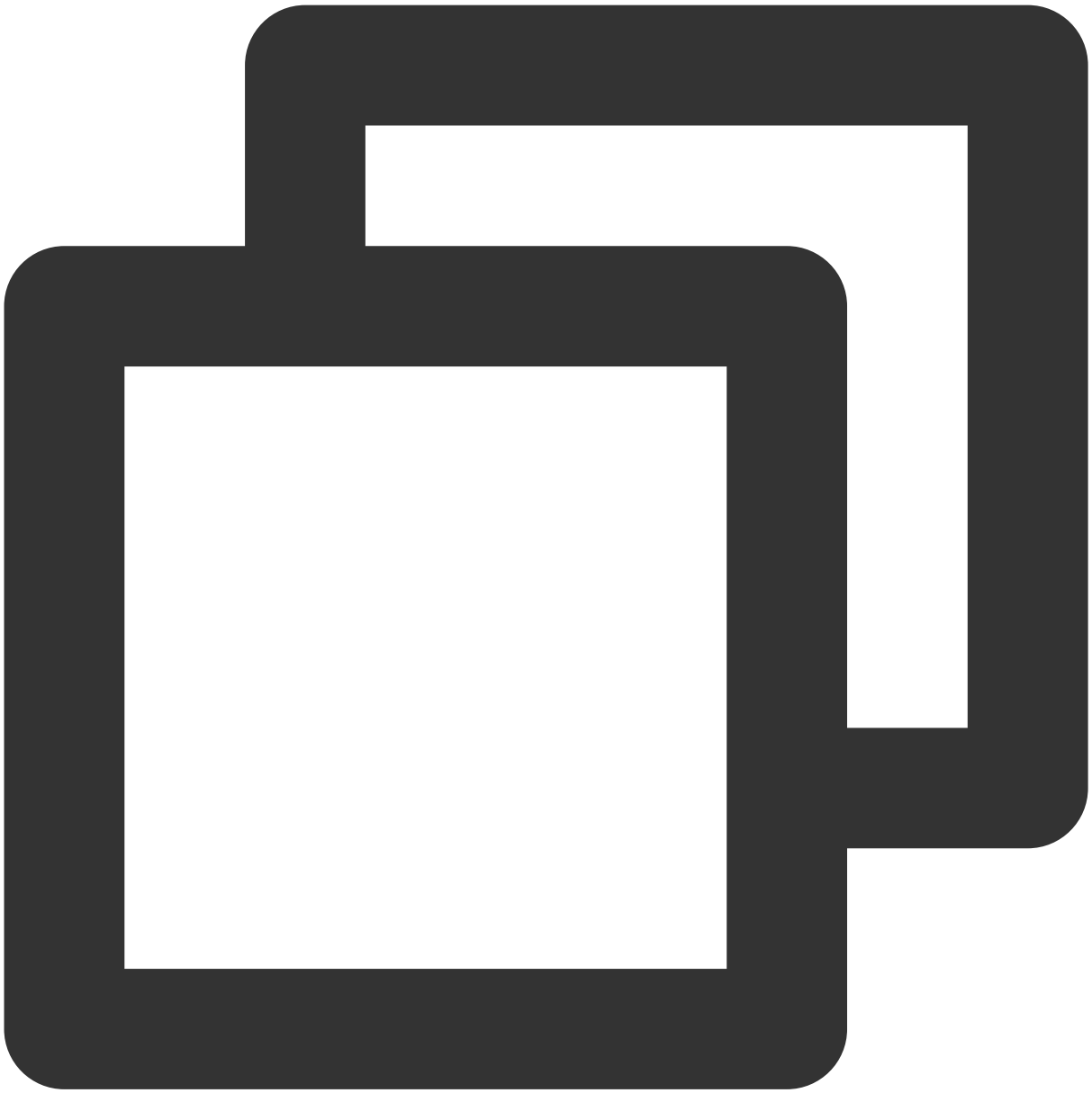
v3.2.9+ 支持。

以隐藏**摄像头按钮**为例。

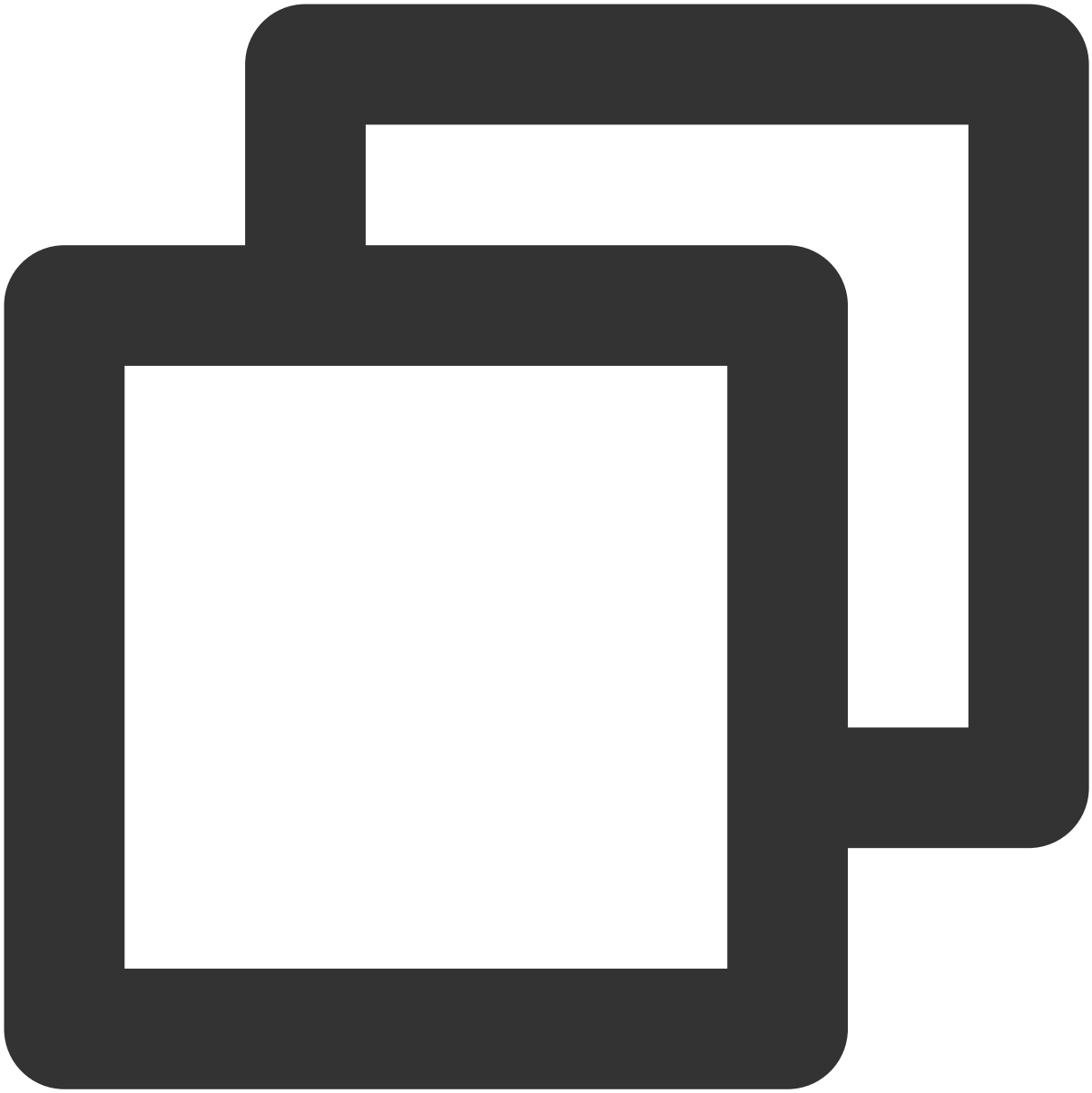


Vue3

React



```
import { TUICallKitServer, FeatureButton } from "@tencentcloud/call-uikit-vue";

TUICallKitServer.hideFeatureButton(FeatureButton.Camera);
```



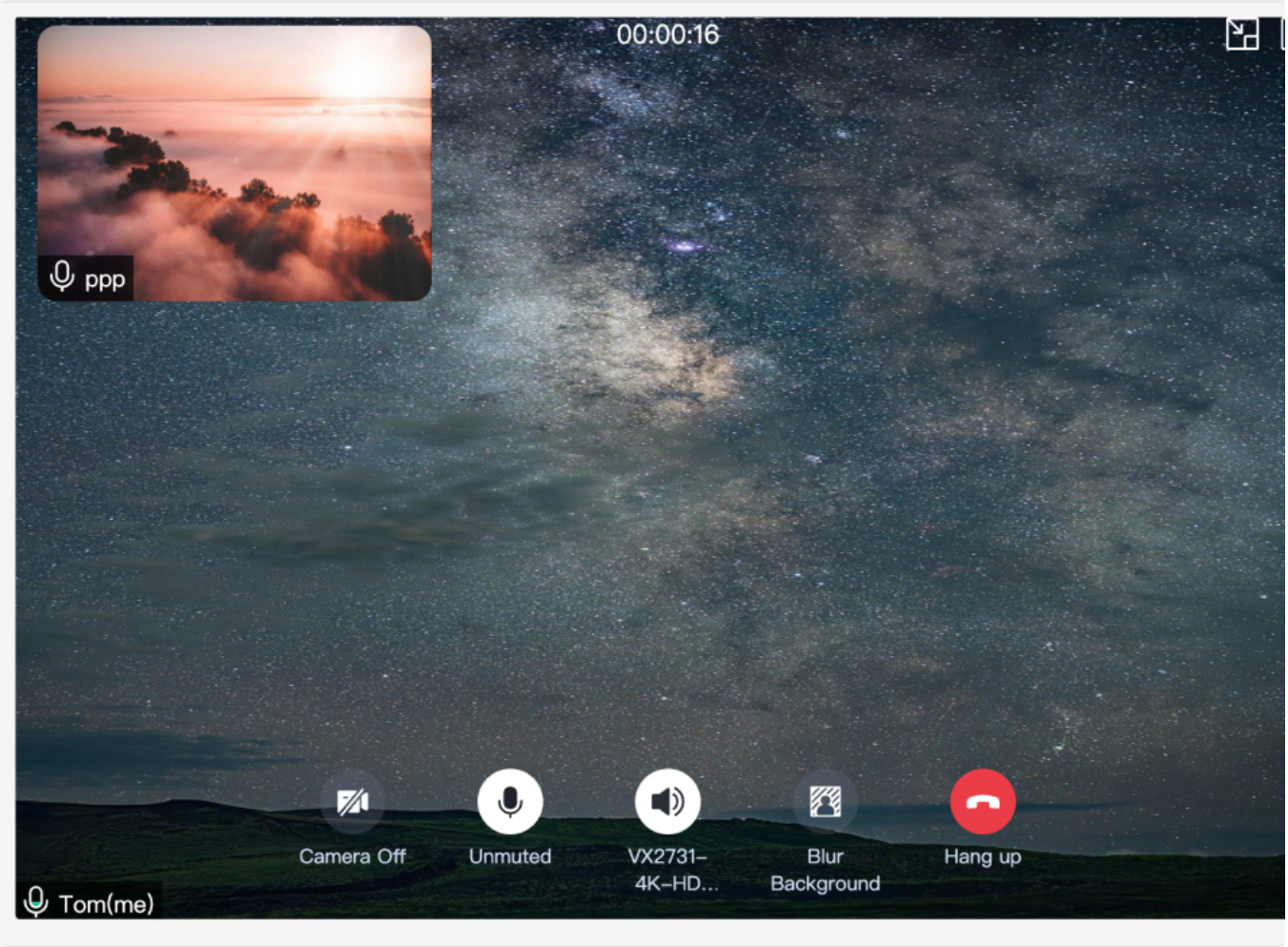
```
import { TUICallKitServer, FeatureButton } from "@tencentcloud/call-uikit-react";

TUICallKitServer.hideFeatureButton(FeatureButton.Camera);
```

自定义通话背景图

通话背景图会在语音通话或者视频通话关闭摄像头后出现，通过调用 setLocalViewBackgroundImage 修改本地用户通话界面背景图，setRemoteViewBackgroundImage 修改远端用户通话界面背景图。

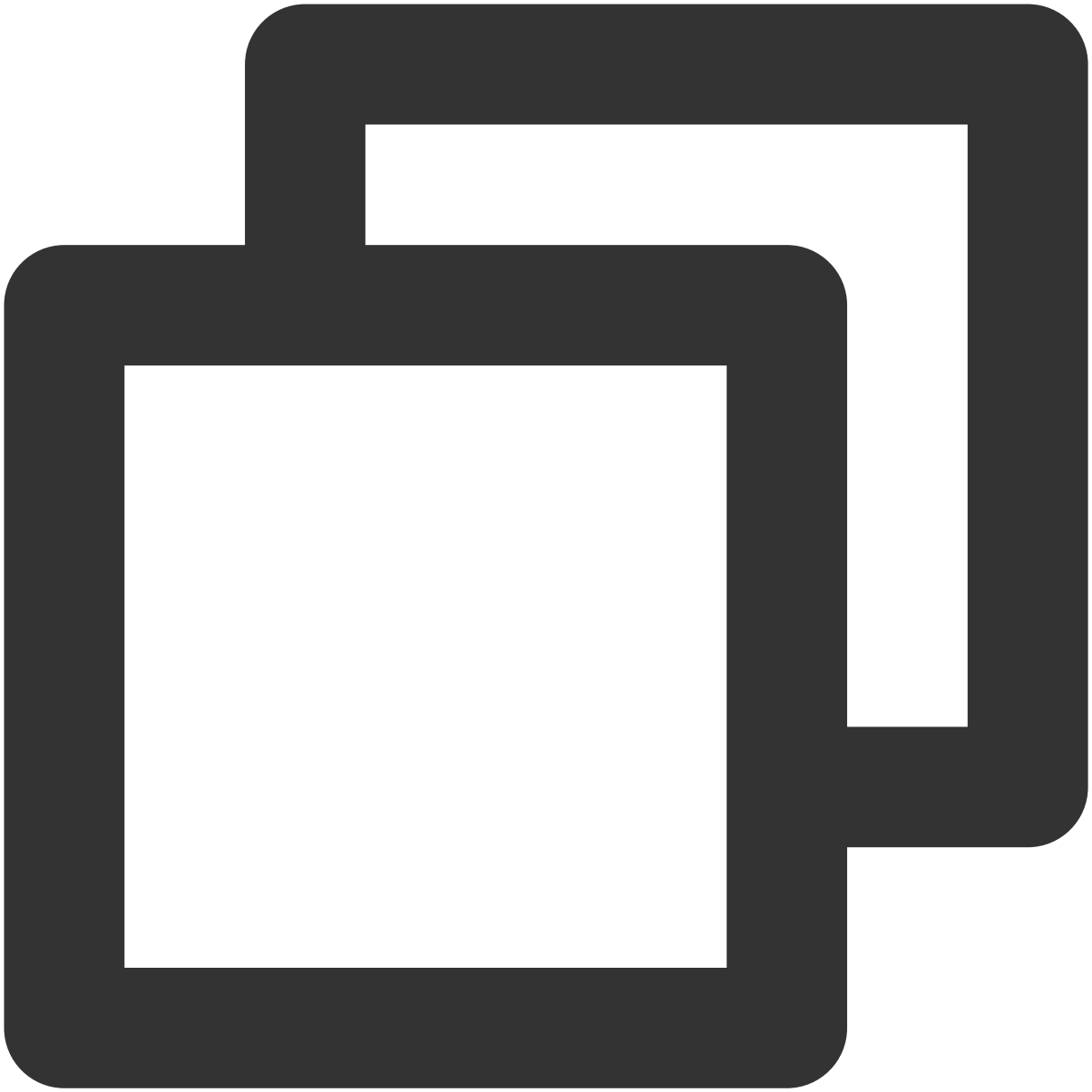
注意：

v3.2.9+ 支持。

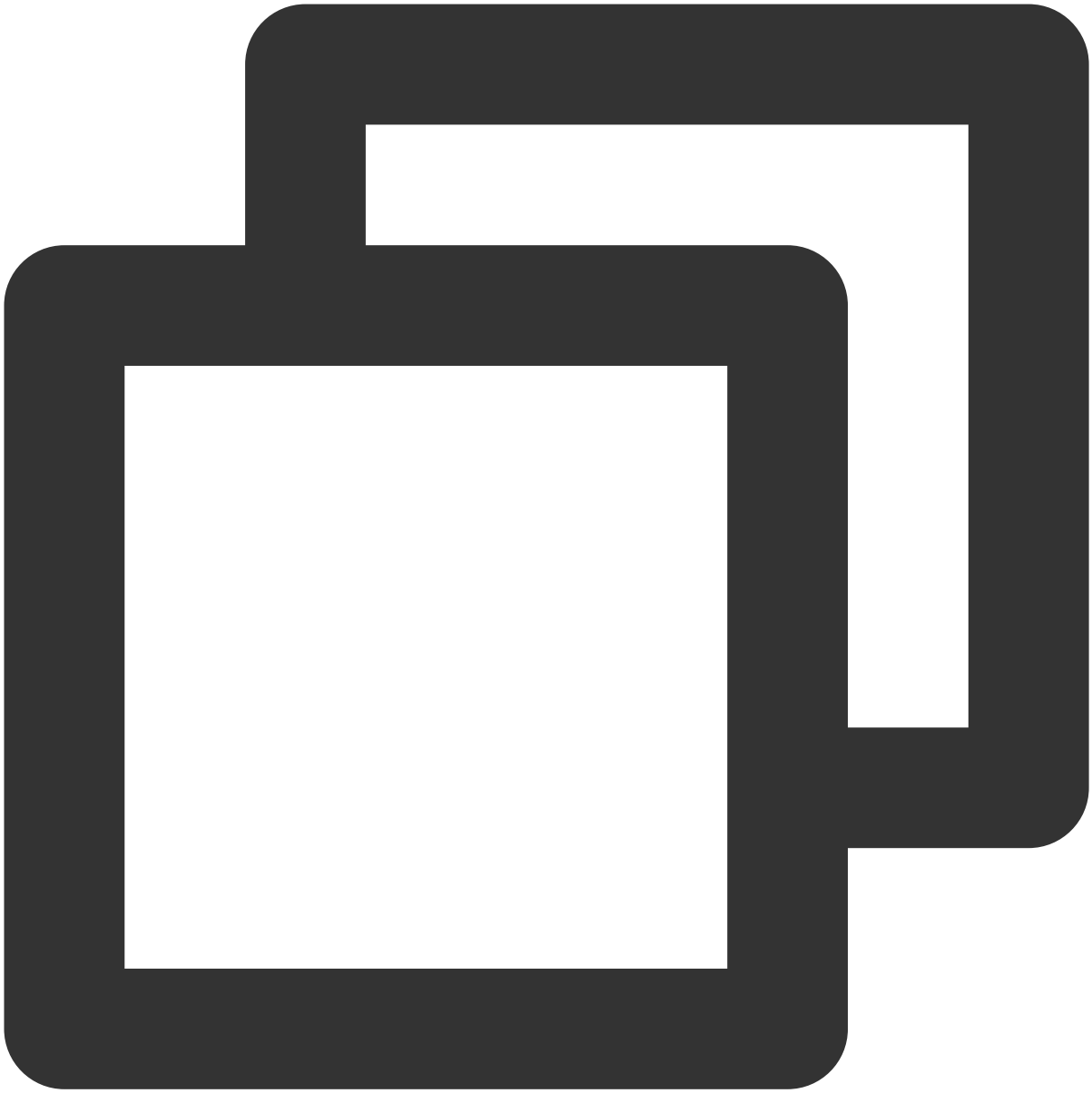


Vue3

React



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-vue";

TUICallKitServer.setLocalViewBackgroundImage('http://xxx.png');
TUICallKitServer.setRemoteViewBackgroundImage('remoteUserId', 'http://xxx.png');
```



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";

TUICallKitServer.setLocalViewBackgroundImage('http://xxx.png');
TUICallKitServer.setRemoteViewBackgroundImage('remoteUserId', 'http://xxx.png');
```

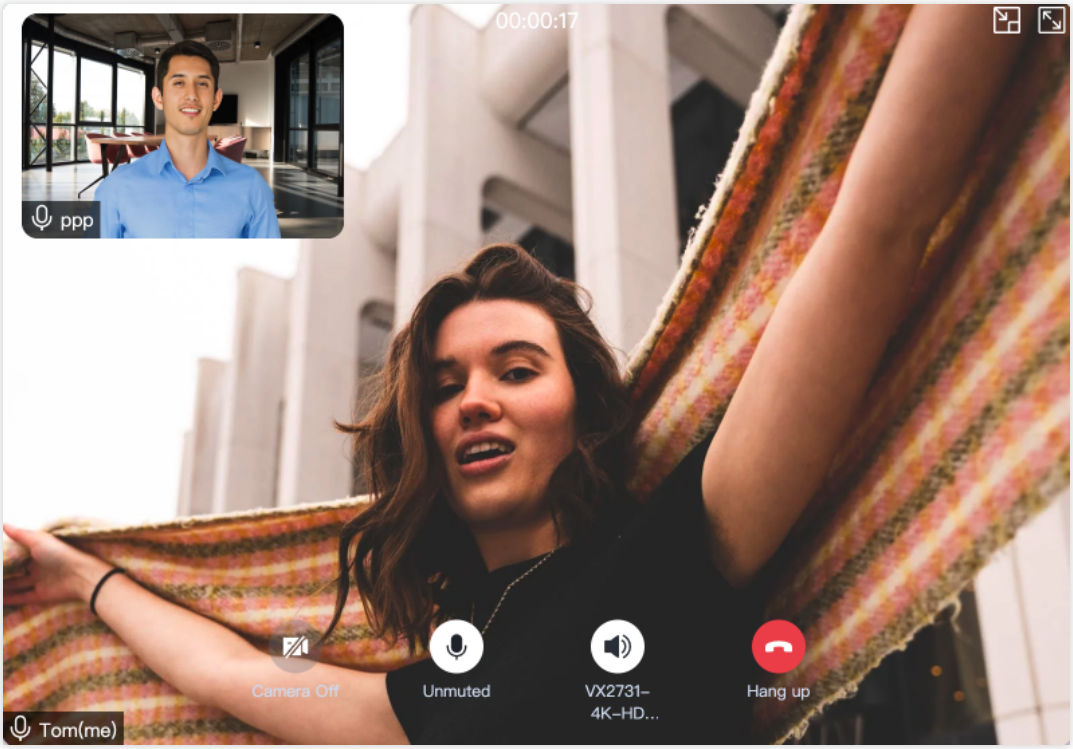
设置布局

注意：

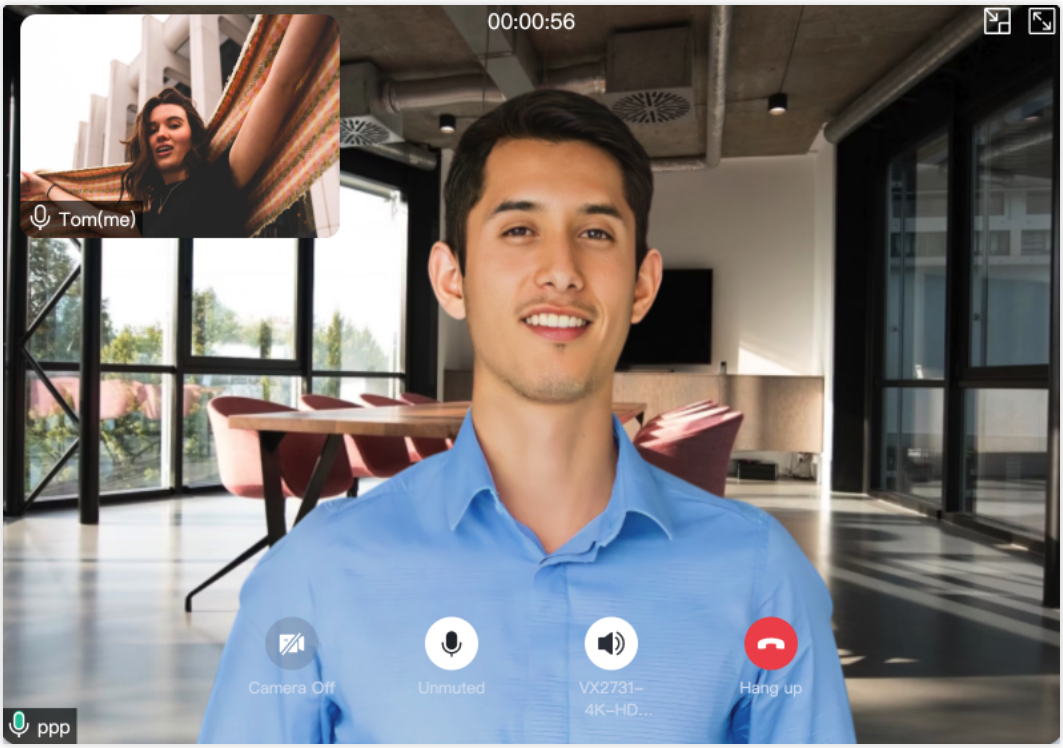
仅 1V1 视频通话可用，v3.3.0+ 支持。

调用 setLayoutMode 设置通话界面布局，目前仅支持 LocalInLargeView，RemoteInLargeView，具体看枚举类型 LayoutMode。

1. LocalInLargeView layout，本地用户在大窗：

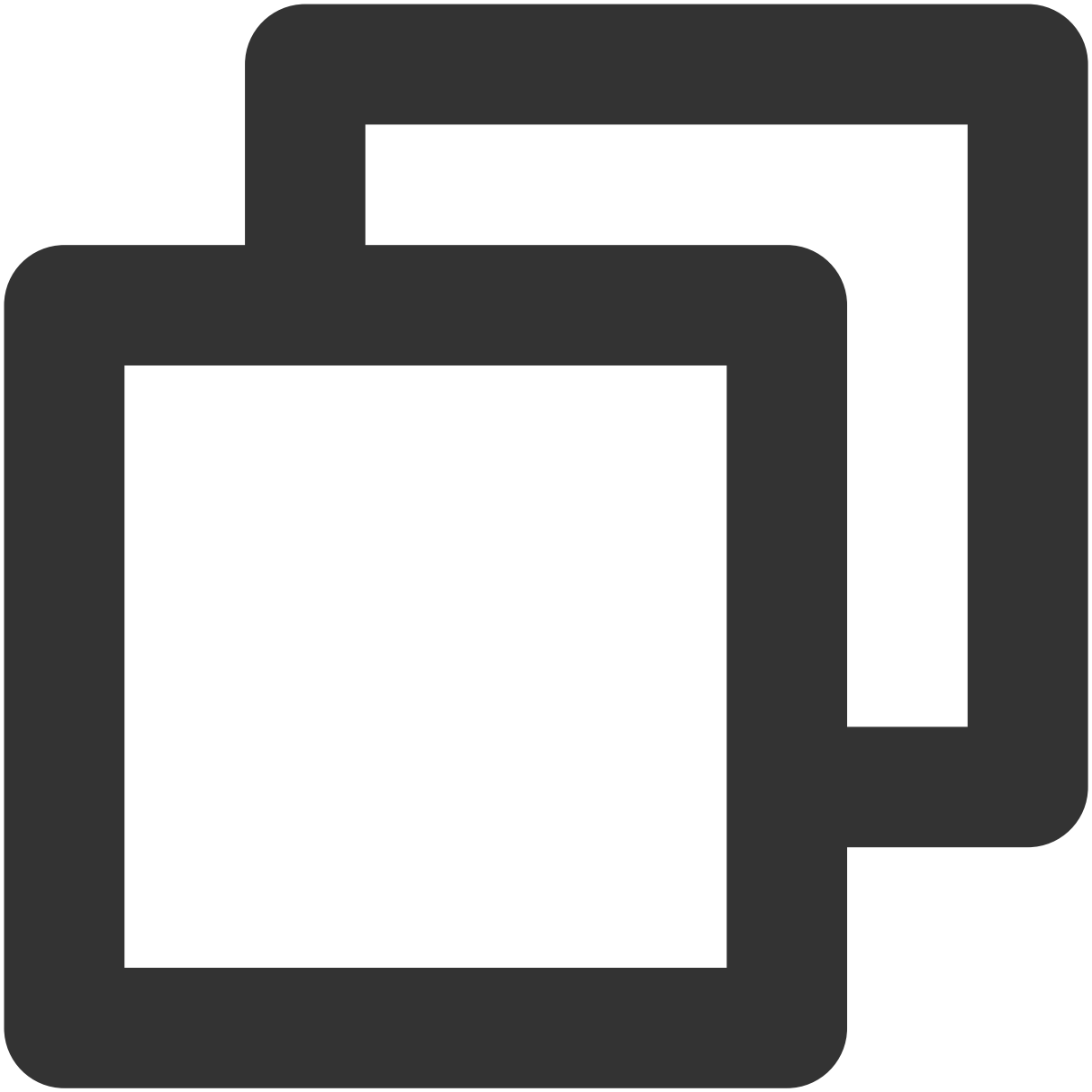


2. RemoteInLargeView layout，远端用户在大窗：

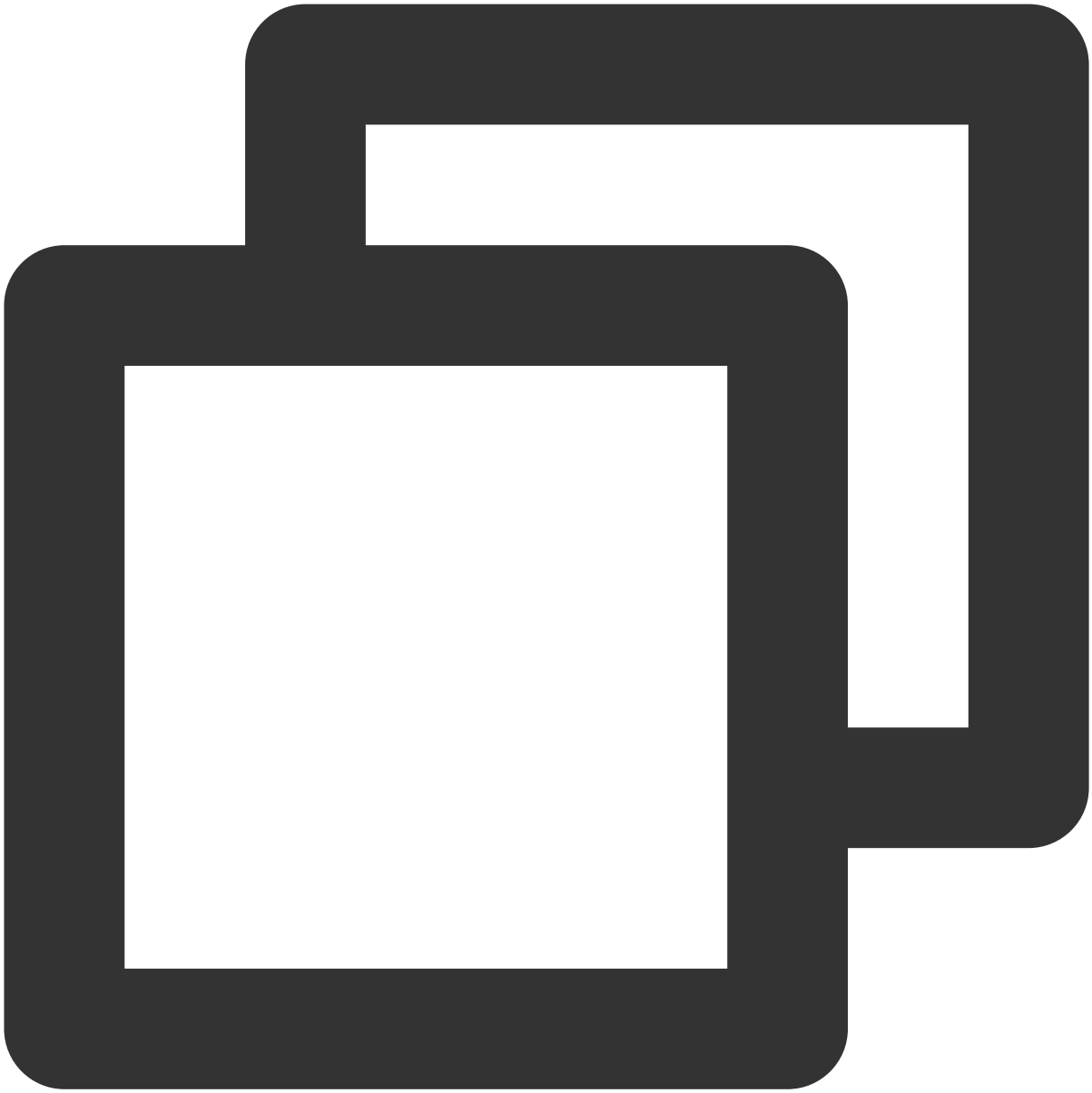


Vue3

React



```
import { TUICallKitServer, LayoutMode } from "@tencentcloud/call-uikit-vue";

TUICallKitServer.setLayoutMode(LayoutMode.LocalInLargeView);
```



```
import { TUICallKitServer, LayoutMode } from "@tencentcloud/call-uikit-react";

TUICallKitServer.setLayoutMode(LayoutMode.LocalInLargeView);
```

设置摄像头初始状态

注意：

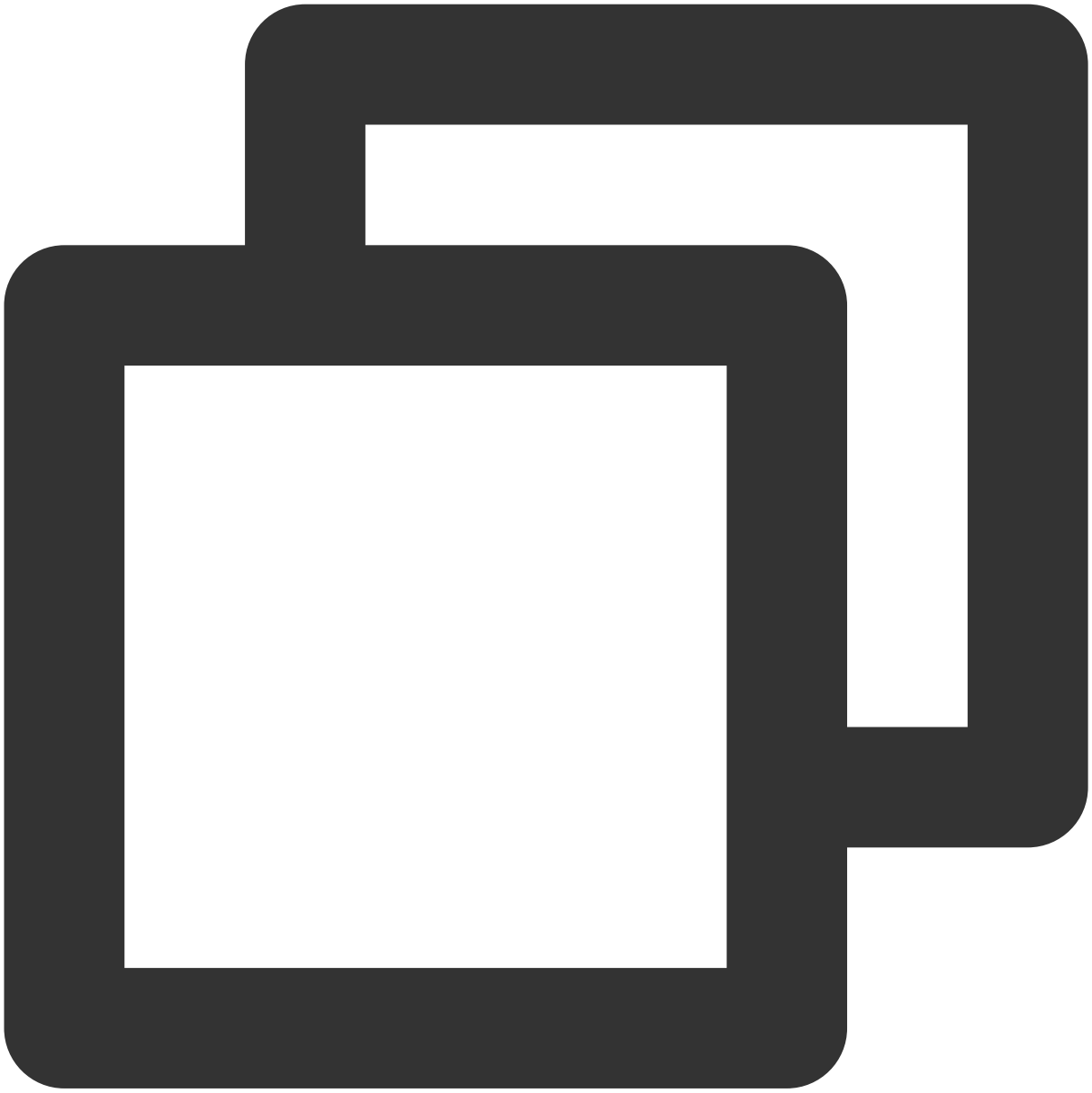
v3.3.0+ 支持。

调用 setCameraDefaultState 设置摄像头按钮初始状态，目前支持开启和关闭。

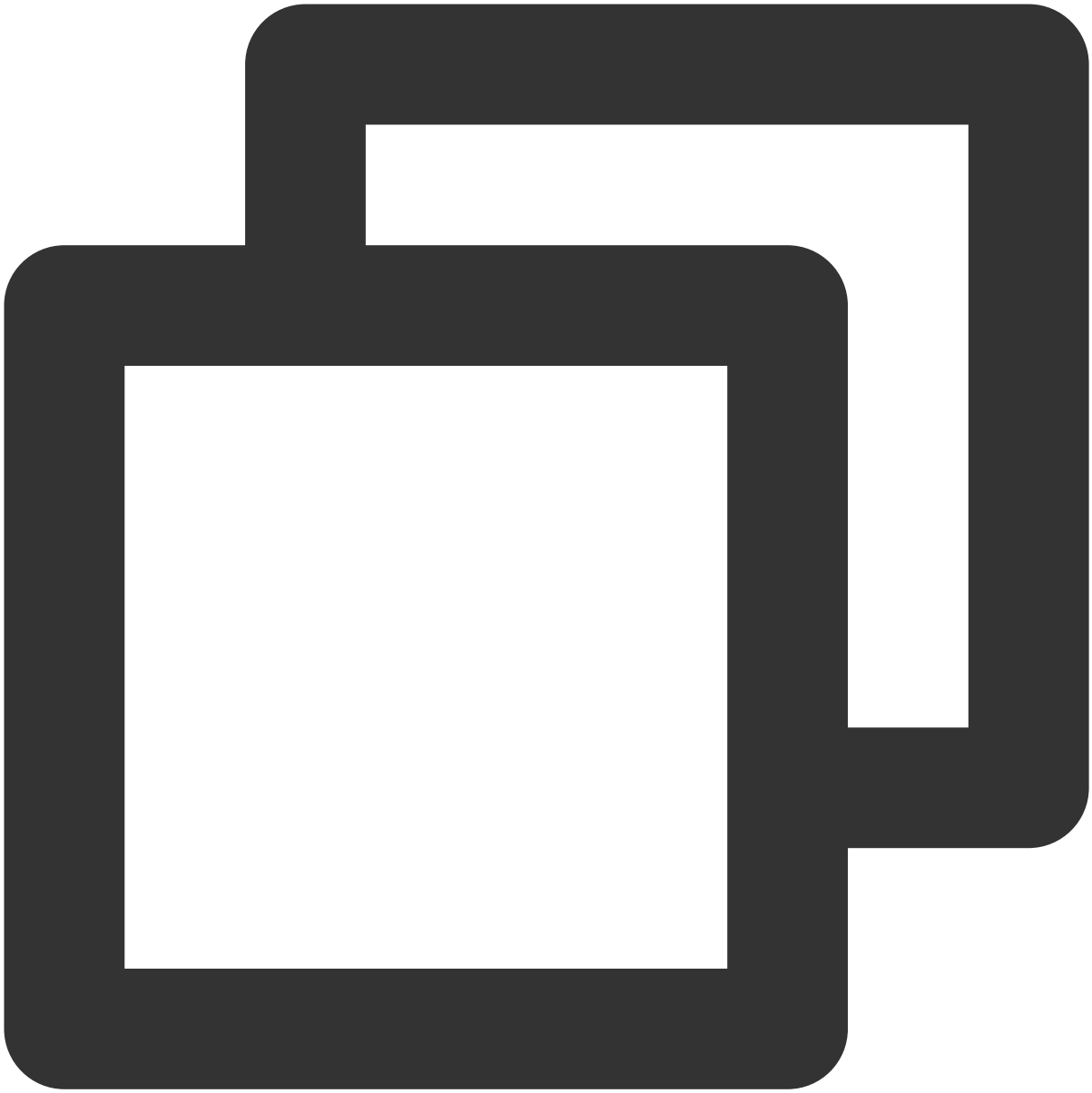
以默认关闭摄像头为例：

Vue3

React



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-vue";

TUICallKitServer.setCameraDefaultState(false);
```



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";

TUICallKitServer.setCameraDefaultState(false);
```

替换图标

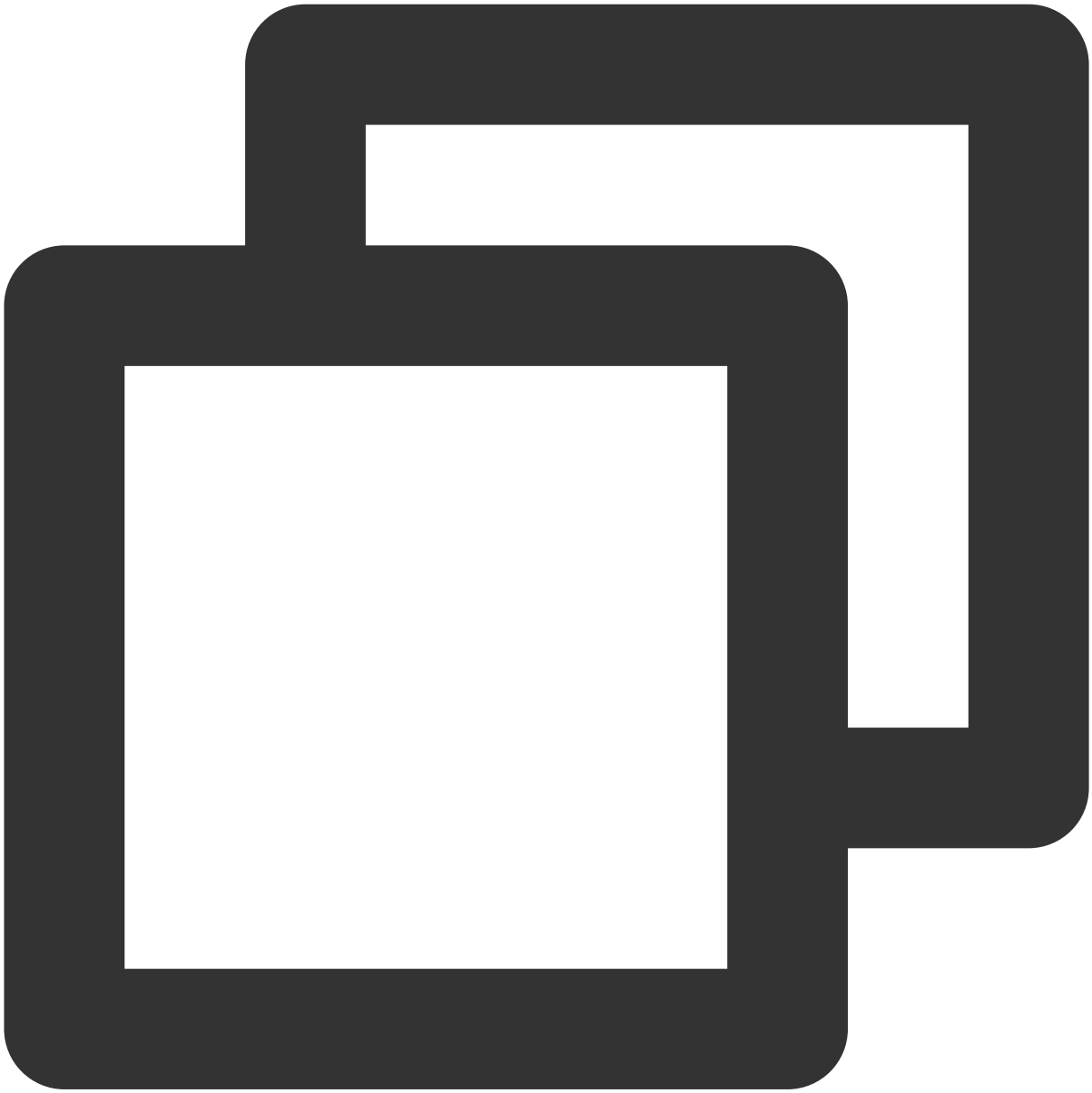
替换图标需要先源码引入，将组件拷贝到您的项目中（源码为 TypeScript 版本）。

注意：

替换图标适用于 `Vue + Typescript` 项目且 `Callkit` 版本号 ≥ 3.2.2，若您采用其他语言或者技术栈，请使用自实现 UI 方案。

1. **下载源码**

Vue3

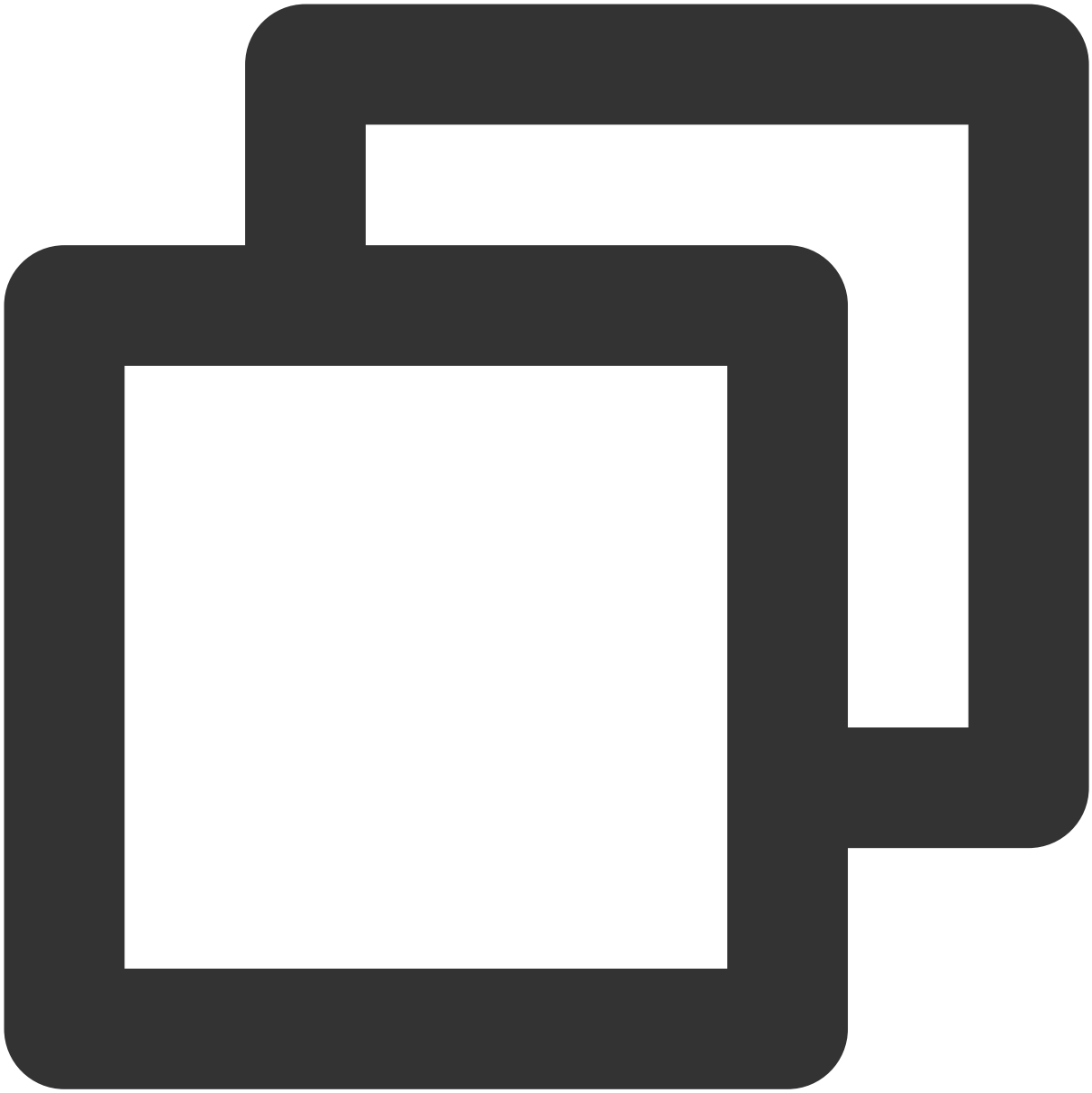


```
npm install @tencentcloud/call-uikit-vue
```

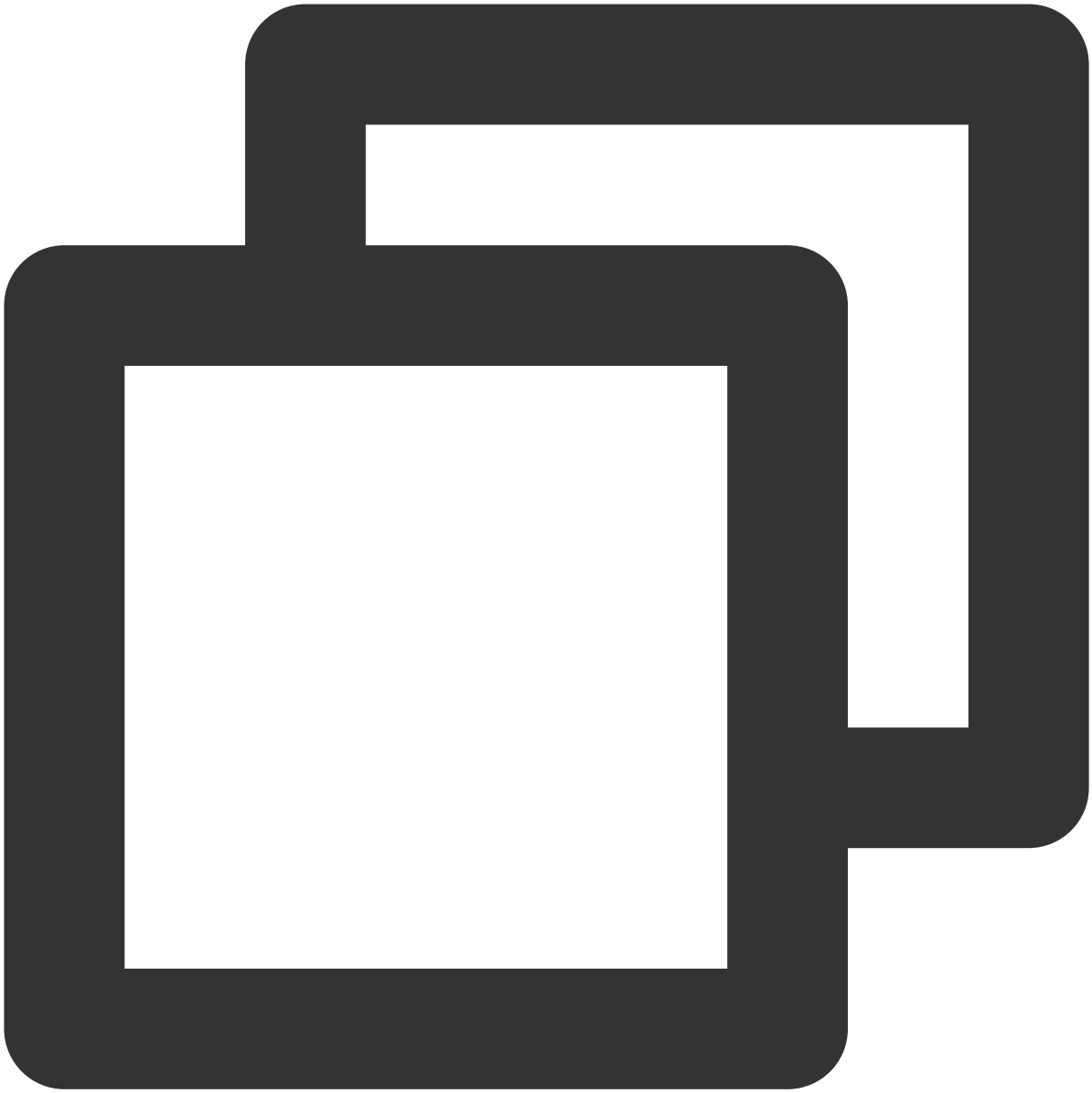
2. **将源码拷贝到自己的项目中，以拷贝到 `src/components/` 目录为例：**

macOS + Vue3

Windows + Vue3



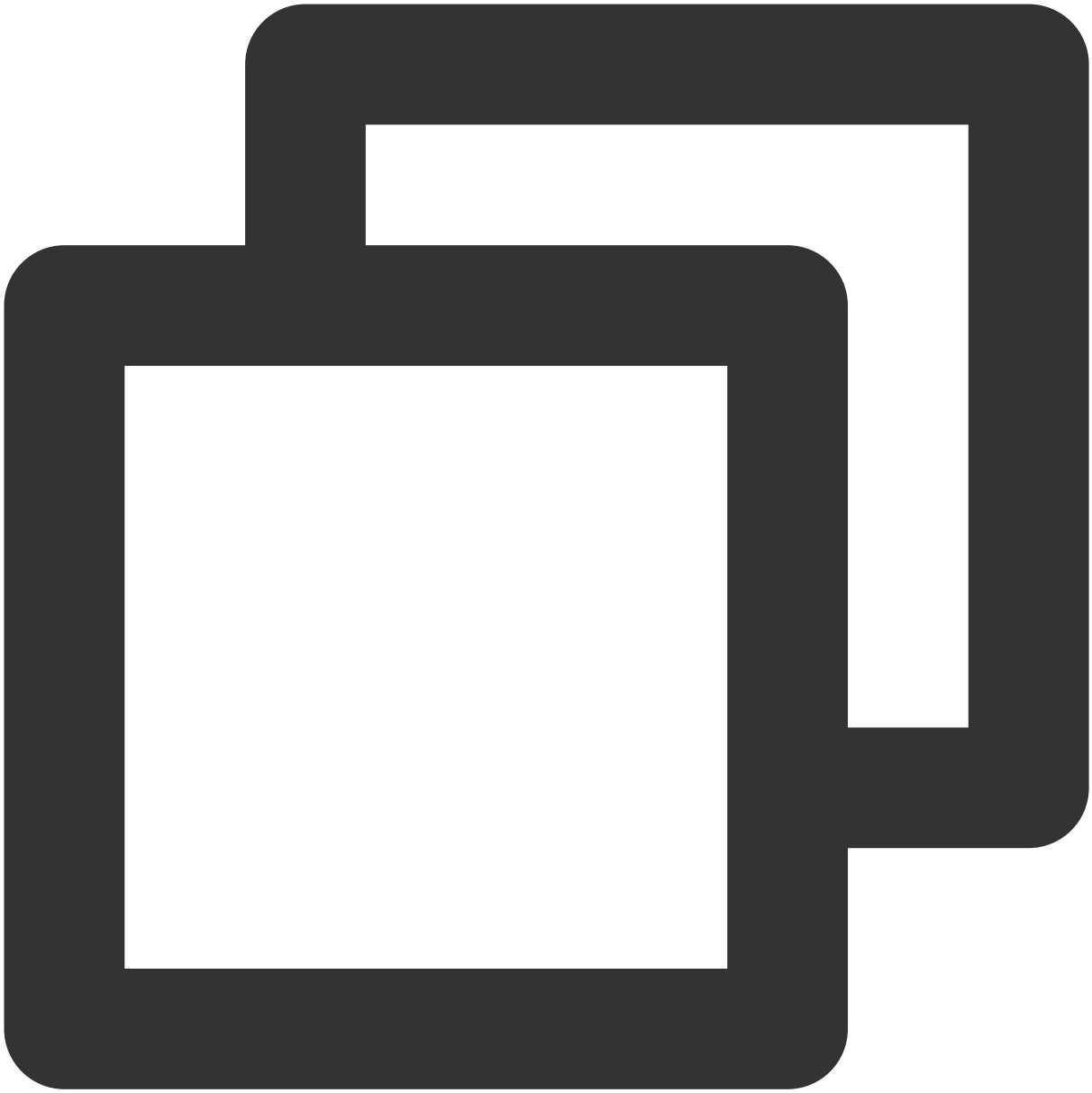
```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uik
```



```
xcopy .\\node_modules\\@tencentcloud\\call-uikit-vue  .\\src\\components\\TUICallKi
```

3. **修改引入路径**

需要将 CallKit 改为从本地文件中引入，如下方代码。其他用法细节可参考 TUICallKit 快速接入。



```
import { TUICallKit, TUICallKitServer, TUICallType } from "./components/TUICallKit/
```

4.

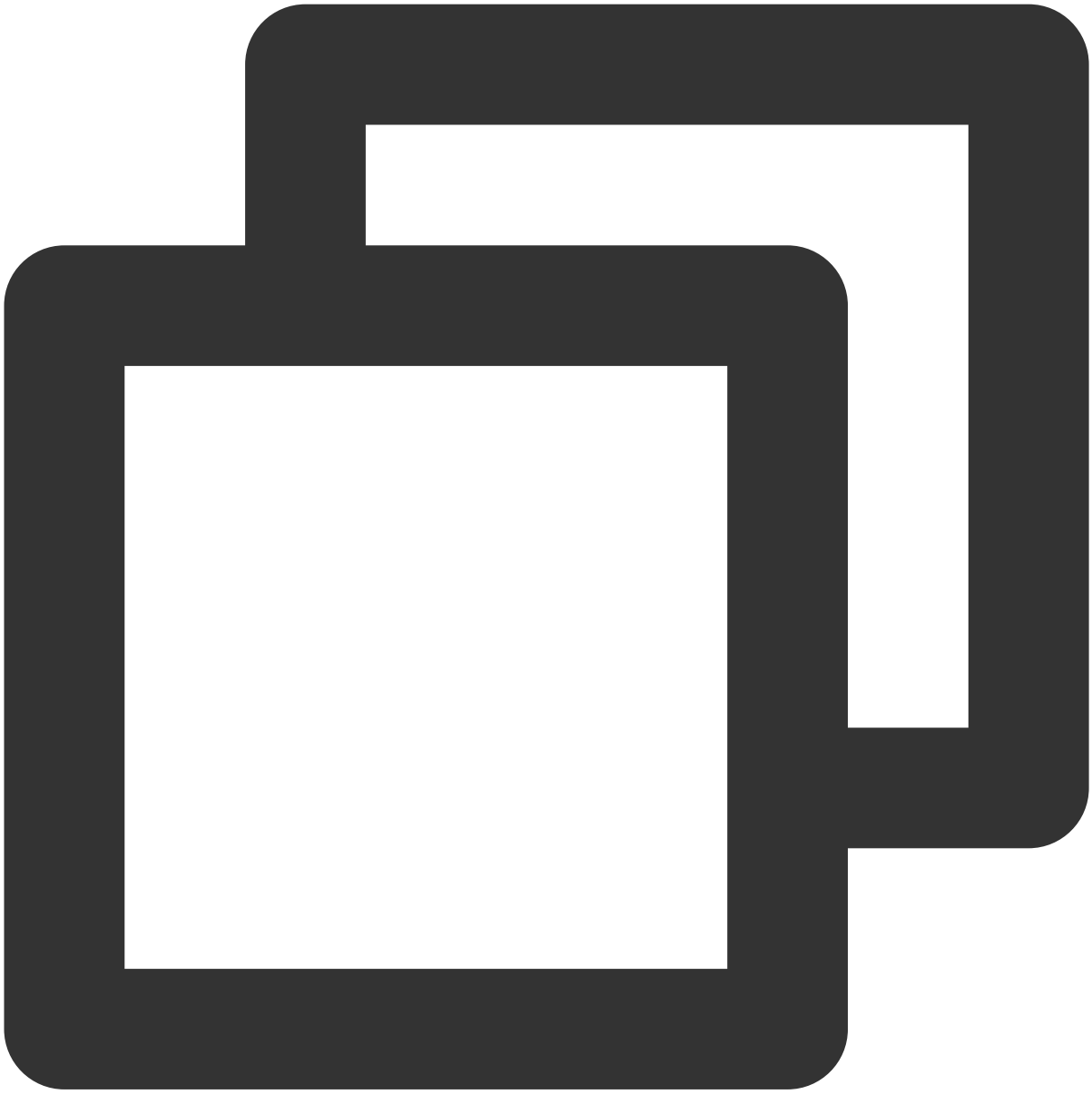
解决源码拷贝可能导致的报错

如果您在使 TUICallKit 组件时遇到了报错，请不要担心，大多数情况下这是由于 ESLint 和 TSConfig 配置不一致造成的。您可以查阅文档，按照要求正确配置即可。如果您需要帮助，请随时联系我们，我们将确保您能够成功地使用此组件。以下是几个常见的问题：

ESLint 报错

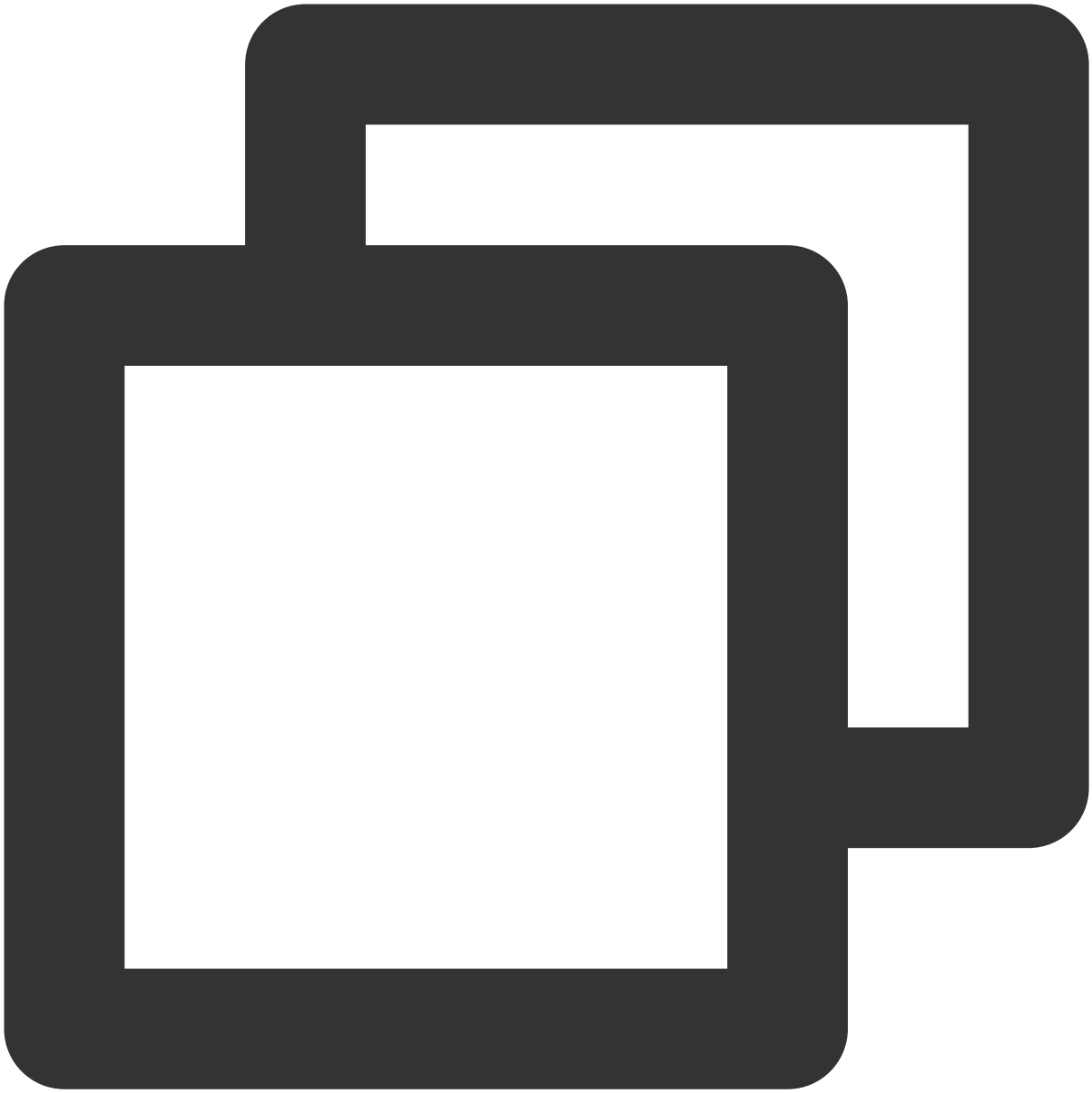
TypeScript 报错

若 TUICallKit 与您项目的代码风格不一致导致报错，可将本组件目录屏蔽，如在项目根目录增加 `.eslintignore` 文件，如：



```
# .eslintignore
src/components/TUICallKit
```

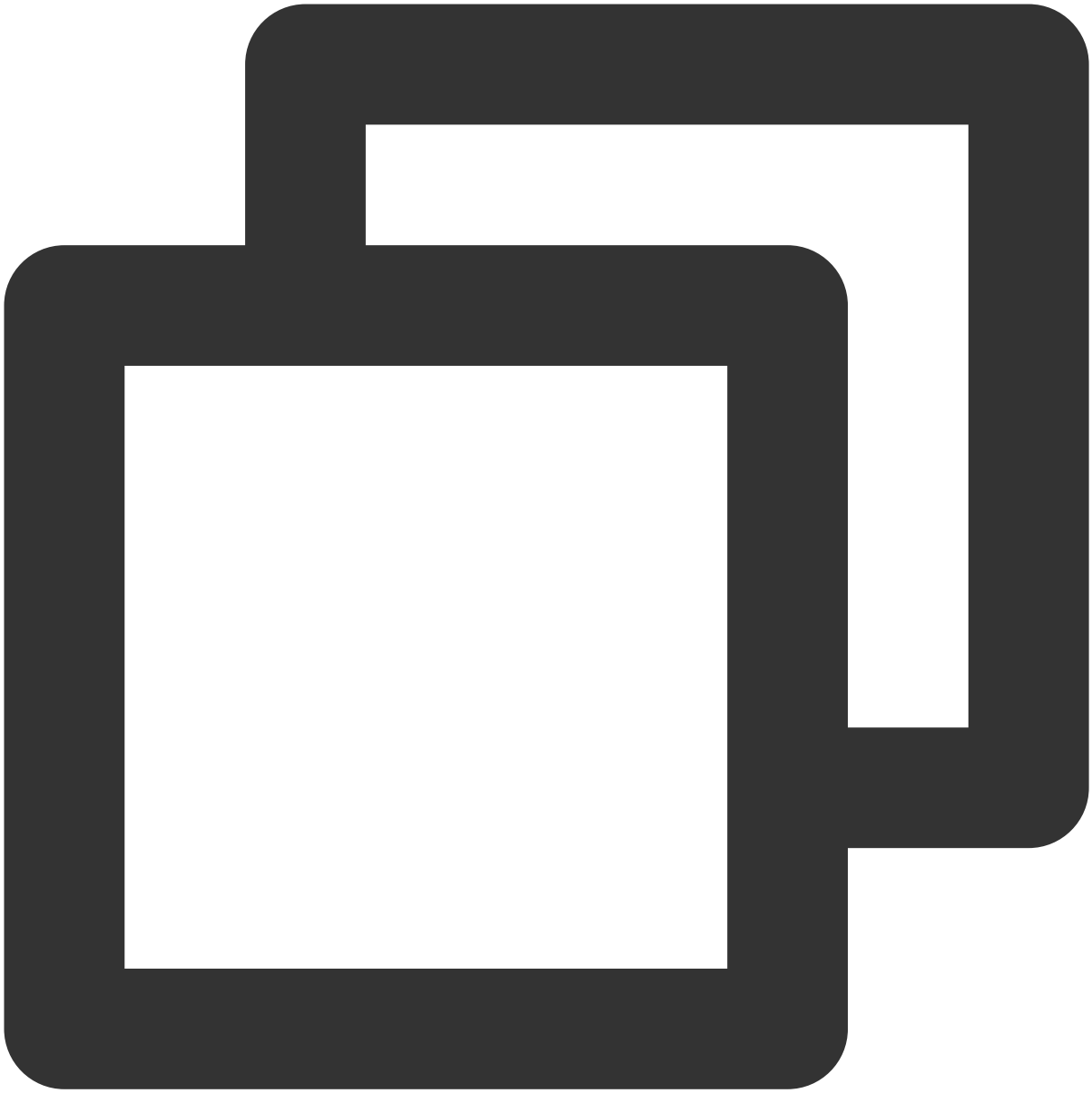
1. 如遇 Cannot find module '../package.json' 报错，是因为 TUICallKit 内引用了 JSON 文件，可在 tsconfig.json 中添加相关配置，示例：



```
{
  "compilerOptions": {
    "resolveJsonModule": true
  }
}
```

其他 TSConfig 问题请参见 TSConfig Reference。

2. 如遇 Uncaught SyntaxError: Invalid or unexpected token 报错，是因为 TUICallKit 使用了装饰器，可在 tsconfig.json 中添加相关配置，示例：



```
{
  "compilerOptions": {
    "experimentalDecorators": true
  }
}
```

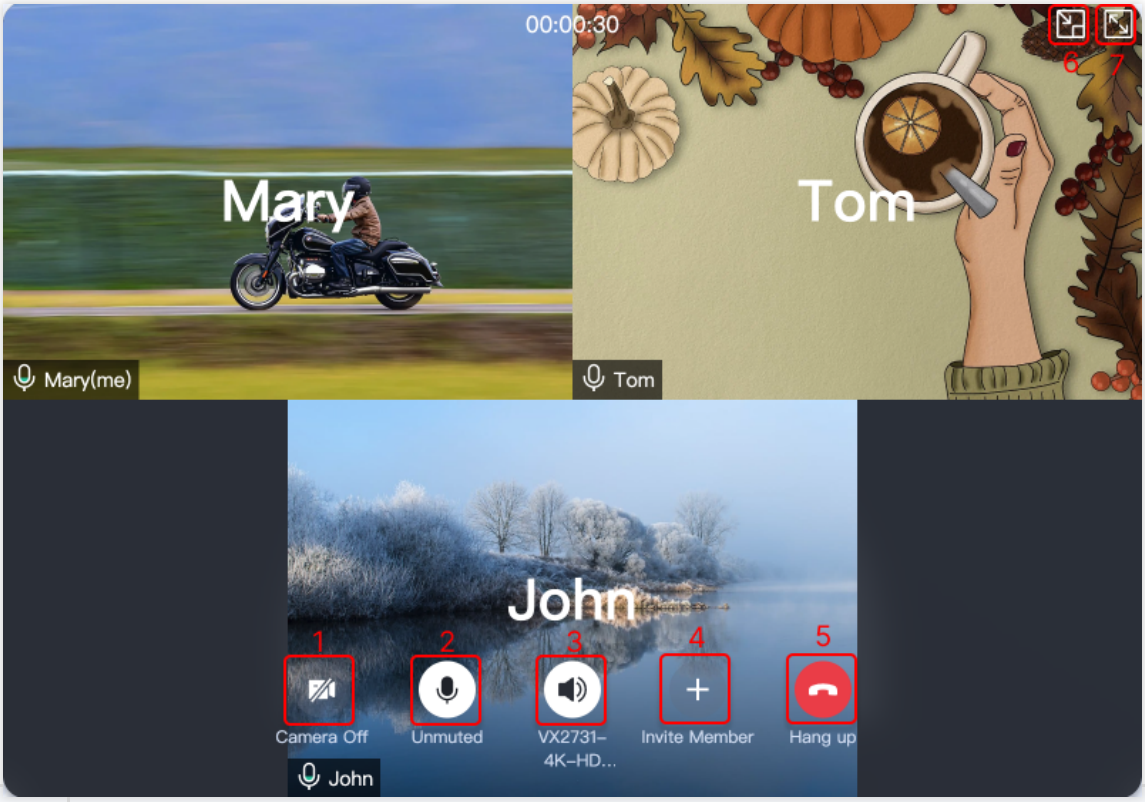
5. **修改 TUICallKit/Components/assets 文件夹下的图标组件**

注意：

为了确保整个应用中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。

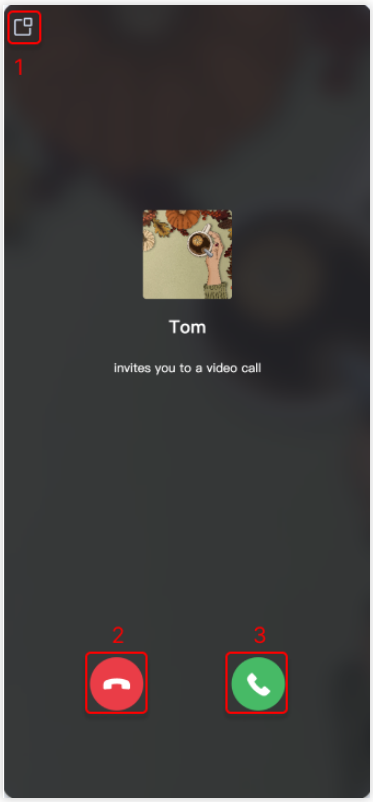
桌面端

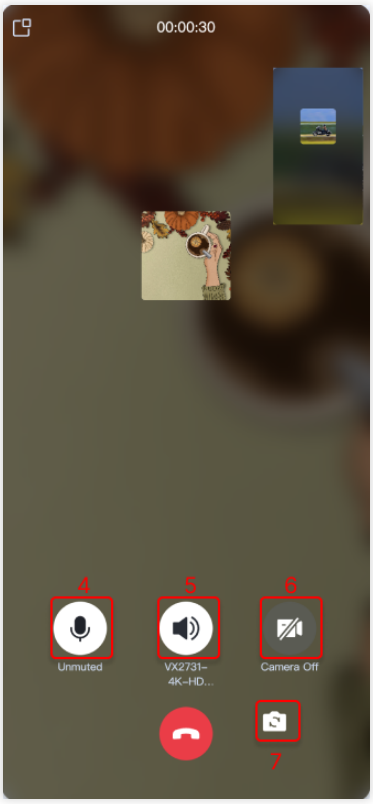
移动端



序号	资源路径
1	/TUICallKit/Components/assets/button/camera-close.svg
2	/TUICallKit/Components/assets/button/microphone-open.svg
3	/TUICallKit/Components/assets/button/speaker-open.svg
4	/TUICallKit/Components/assets/button/desktop/inviteUser.svg
5	/TUICallKit/Components/assets/button/hangup.svg
6	/TUICallKit/Components/assets/button/desktop/minimize.svg
7	/TUICallKit/Components/assets/button/desktop/fullScreen.svg

版权所有：腾讯云计算（北京）有限责任公司


1
Tom
invites you to a video call
2
3


00:00:30
4
Unmuted
5
VX2731-
4K-HD...
6
Camera Off
7

序号	资源路径
1	/TUICallKit/Components/assets/button/mobile/minimize.svg
2	/TUICallKit/Components/assets/button/hangup.svg
3	/TUICallKit/Components/assets/button/accept.svg
4	/TUICallKit/Components/assets/button/microphone-open.svg
5	/TUICallKit/Components/assets/button/speaker-open.svg
6	/TUICallKit/Components/assets/button/camera-close.svg
7	/TUICallKit/Components/assets/button/switchCamera.svg

方案2：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI SDK 实现的，您可以完全基于 TUICallEngine 实现一套自己的 UI 界面。详情可见：

TUICallEngine 接入指引

TUICallEngine API 接口地址

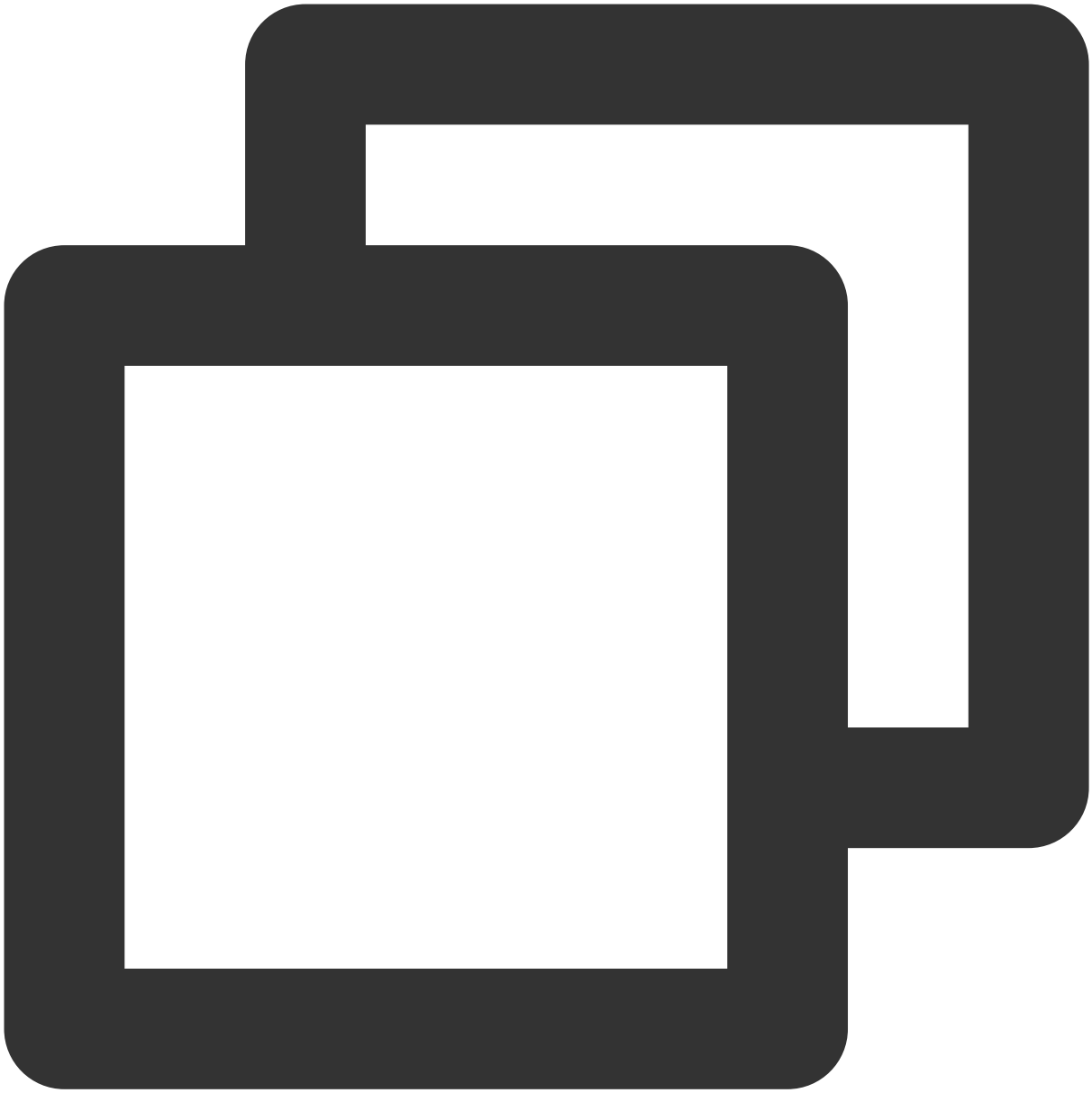
Flutter

最近更新时间：2023-08-23 17:22:05

本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。
注意：页面定制方案需要使用 tencent_calls_uikit 插件 1.8.0 以上的版本。

方案一：界面微调方案

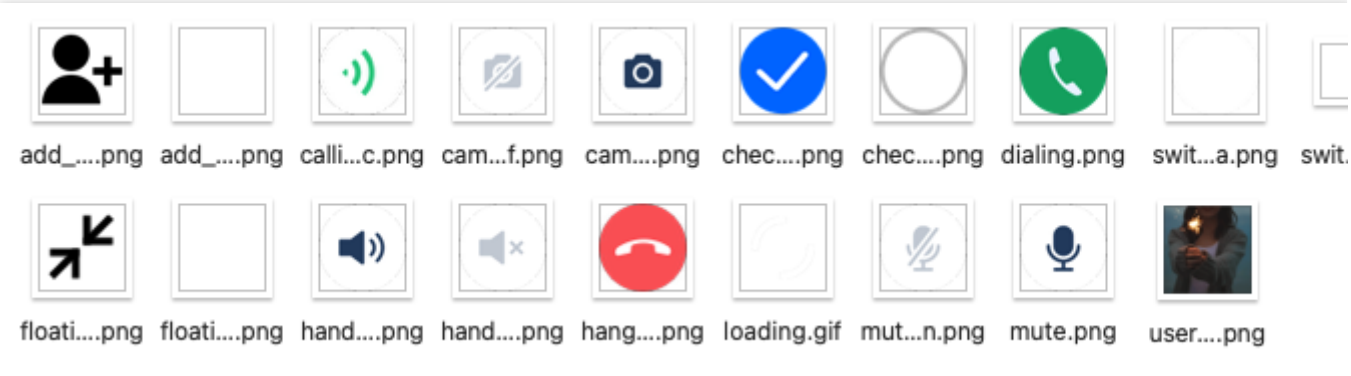
你可以下载 tencent_calls_uikit 插件的最新版本到本地，之后在你的工程中使用本地依赖的方式接入插件，本地依赖方式如下：
在工程 **pubspec.yaml** 文件中的 **dependencies** 节点下，添加 **tencent_calls_uikit** 插件依赖，示例如下



```
dependencies:
    tencent_calls_uikit:
      path: 插件目录
```

替换图标

您可以直接替换 `assets\\images` 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 **assets\\audios** 文件夹下的三个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_hangup.mp3	被挂断的声音
phone_ringing.mp3	接到呼叫时的声音

替换文案

您可以通过修改 **lib\\src\\i18n** 目录下 **strings.g.dart** 文件中的的字符串来修改视频通话界面中的字符串内容。

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI 组件实现的，您可以不依赖 **tencent_calls_uikit** 插件，而直接依赖 tencent_calls_engine 插件，完全基于 TUICallEngine 实现一套自己的 UI 界面。

TUICallEngine

TUICallEngine 是整个通话组件的底层接口，主要提供了1对1音视频通话和群组内通话的发起、接听、拒绝、结束以及设备操作等关键接口。

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，收取自己感兴趣的回调事件。

版权所有：腾讯云计算（北京）有限责任公司

更多特性（TUICallKit）
群组通话
Android&iOS&Flutter

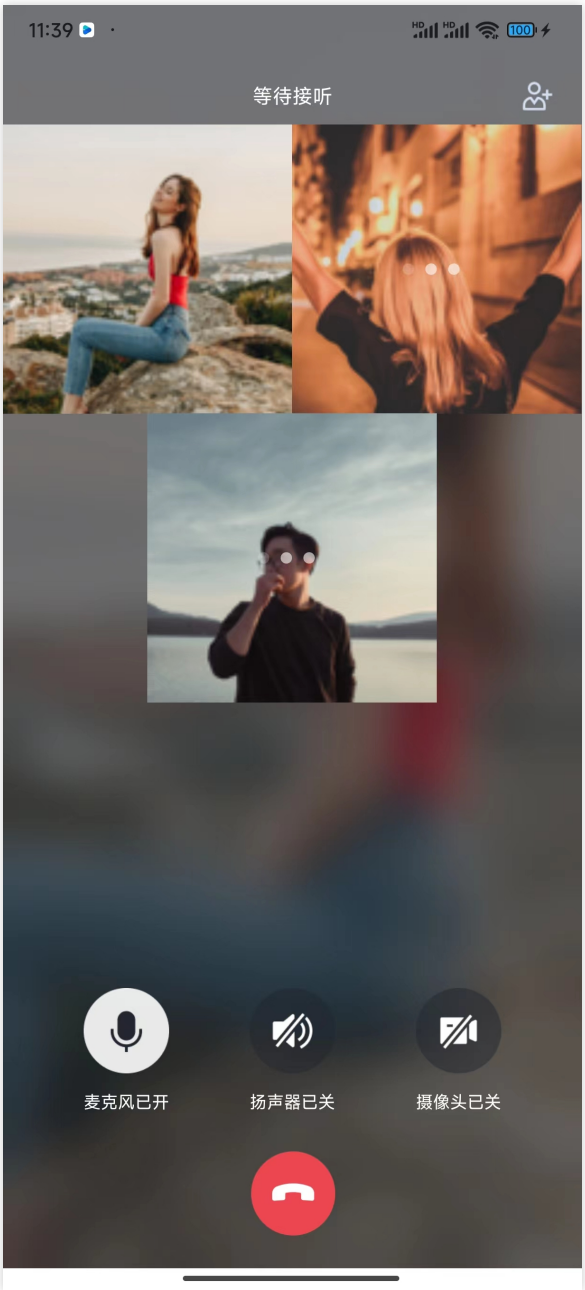
最近更新时间：2024-05-08 11:37:24

本文介绍群组通话功能的使用，如发起群组通话、加入群组通话。

预期效果

TUICallKit 支持群组通话，预期效果见下图。

发起群组通话	收到群组通话请求



创建 groupID

使用群组通话功能前，需要先创建群组，在已存在的群组中发起群组通话。

方式一：通过调用 IM API 创建群，详见 IM 群组管理。

方式二：通过控制台手动创建群组，详见 控制台群组管理。

群组通话

发起群组通话

调用 groupCall API 发起群通话。

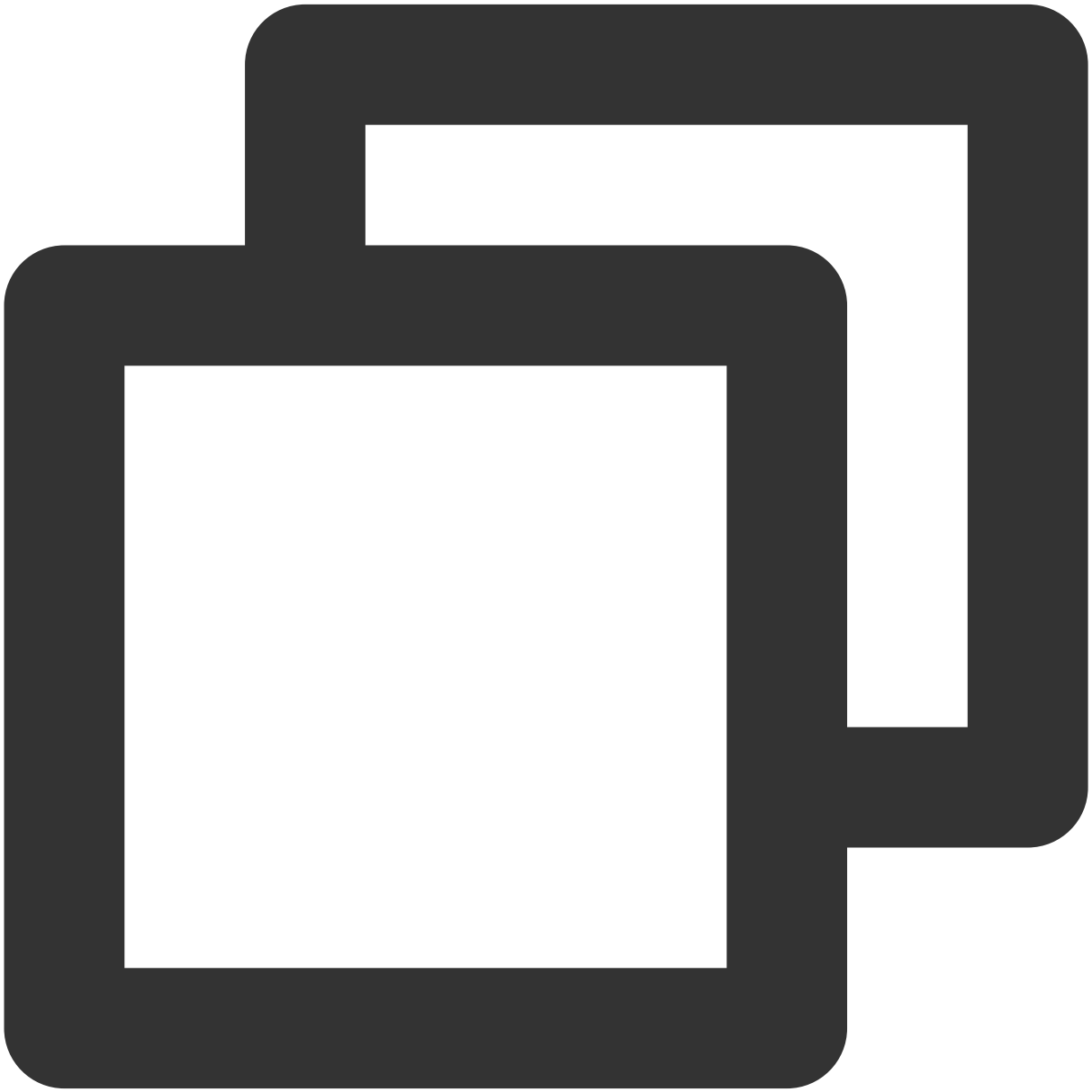
Android（Kotlin）

Android（Java）

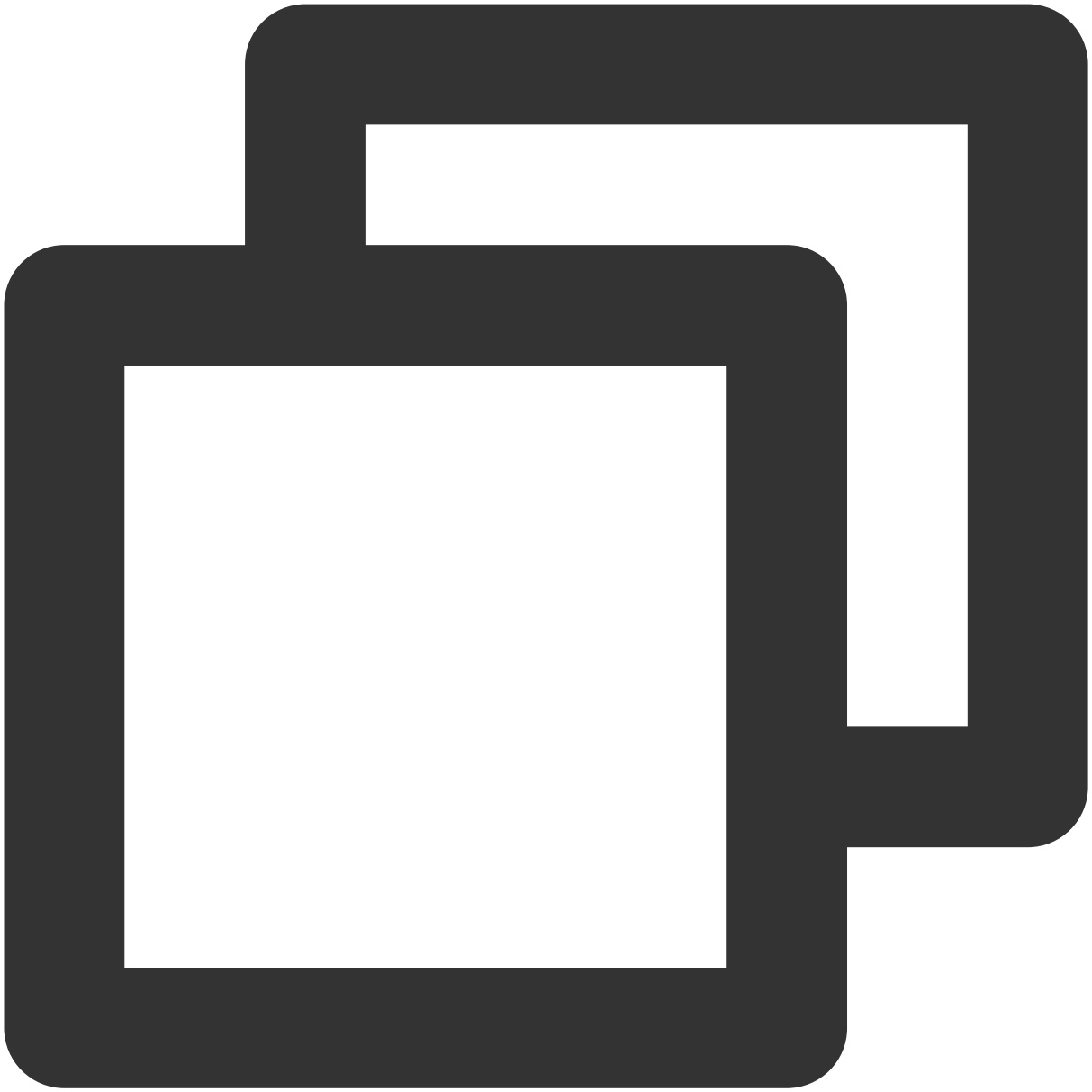
iOS（Swift）

iOS（Objective-C）

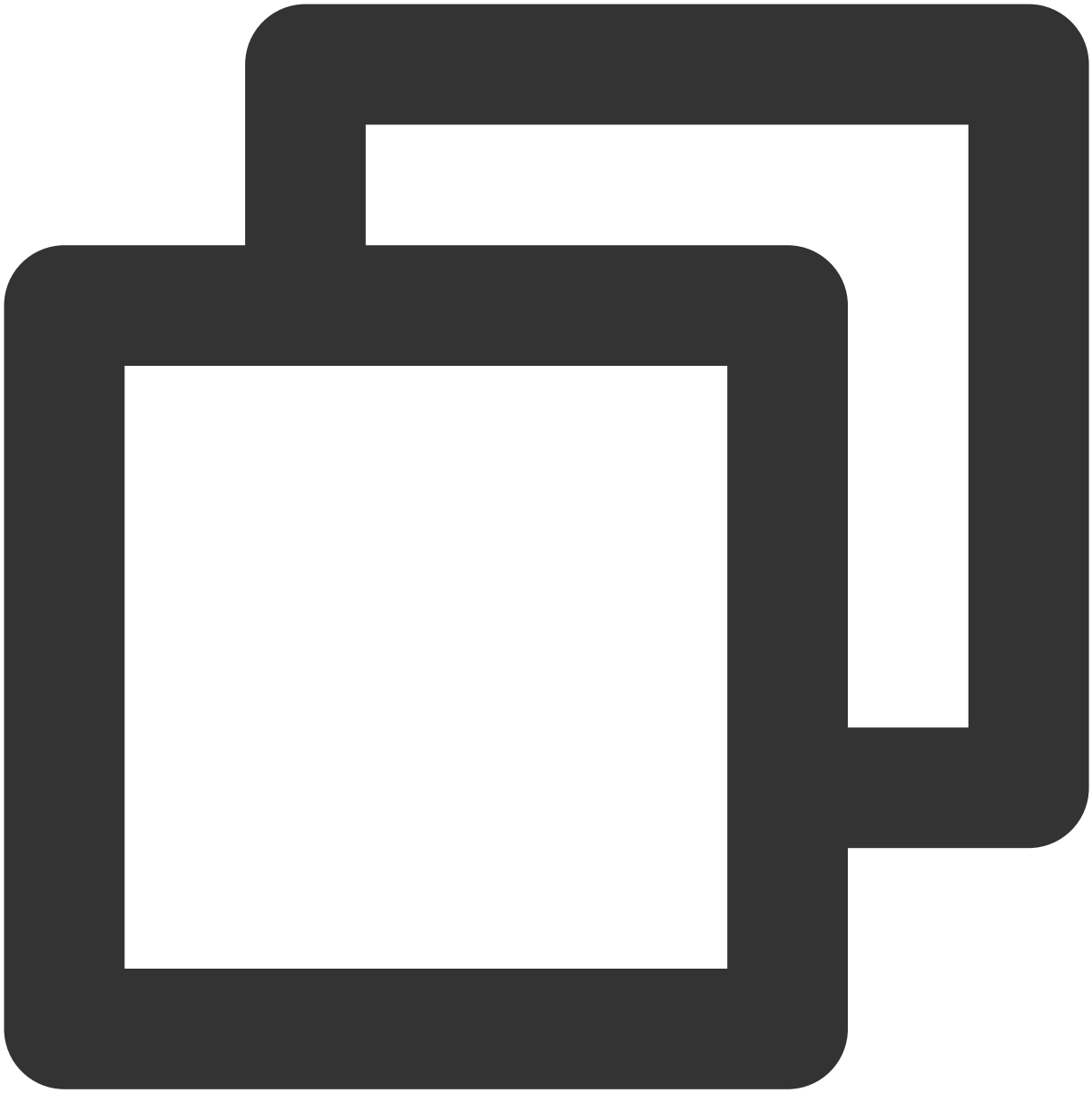
Flutter（Dart）



```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mik
```

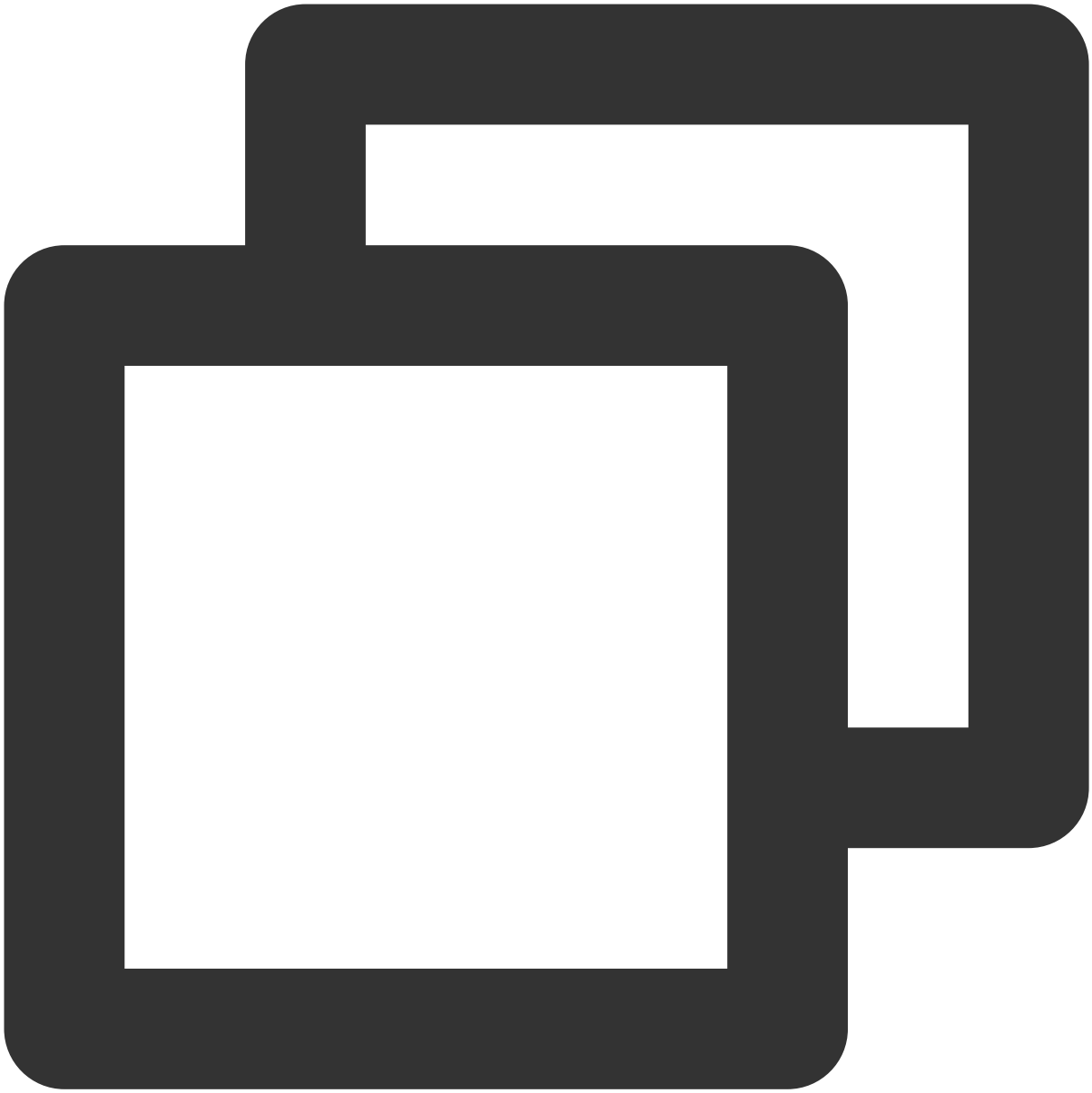


```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mik
```



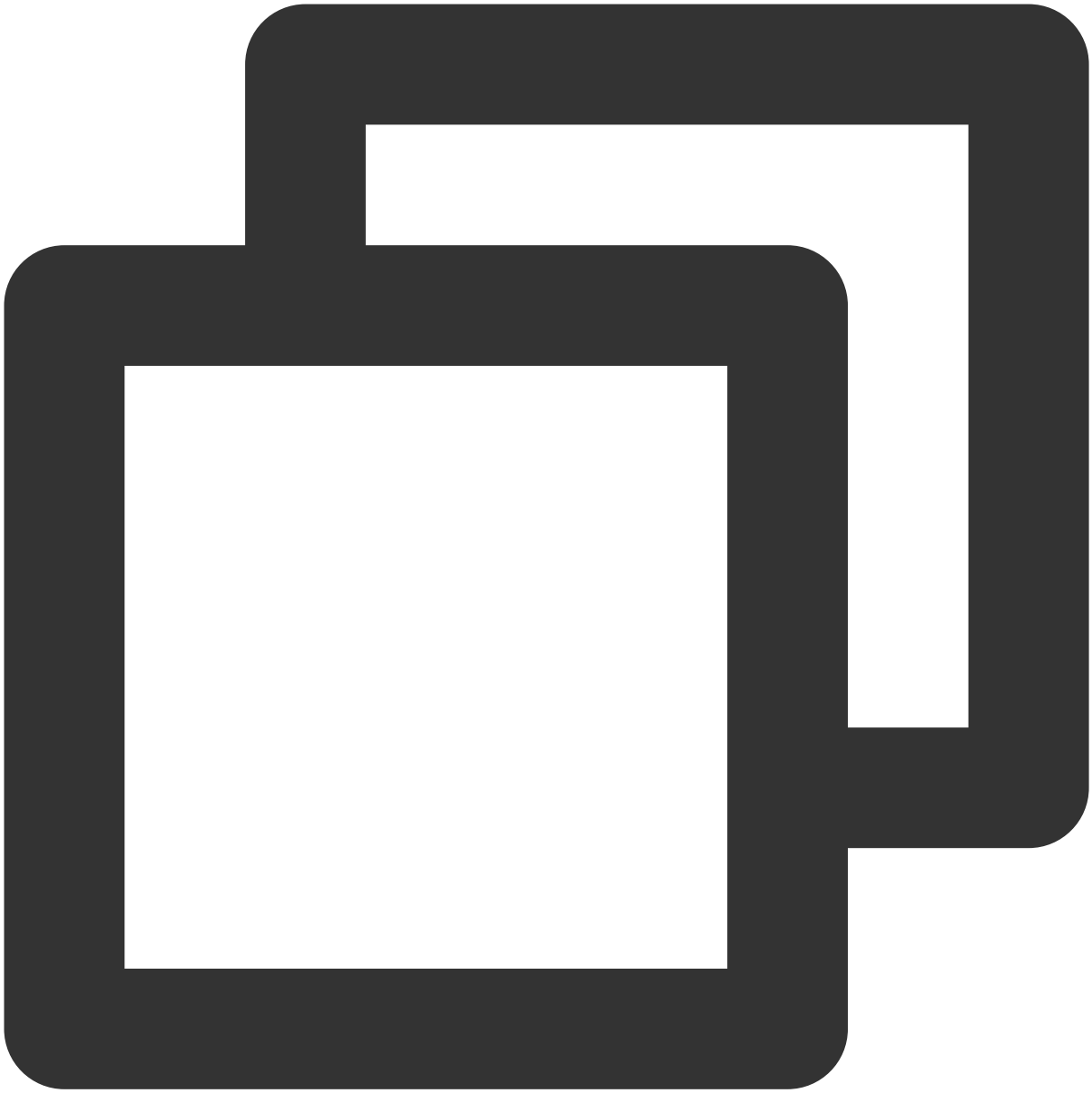
```
import TUICallKit

TUICallKit.createInstance().groupCall(groupId: "12345678",
                                      userIdList: ["denny", "mike", "tommy"],
                                      callMediaType: .video)
```

版权所有：腾讯云计算（北京）有限责任公司



```
#import <TUICallKit/TUICallKit.h>

[[TUICallKit createInstance] groupCall:@"12345678"
                              userIdList:@[@"denny", @"mike", @"tommy"]
                           callMediaType:TUICallMediaTypeVideo];
```

版权所有：腾讯云计算（北京）有限责任公司



```
TUICallKit.instance.groupCall('0001', ['denny', 'mike', 'tommy'], TUICallMediaType.
```

加入群组通话

调用 joinInGroupCall API 主动加入群组中已有的音视频通话。

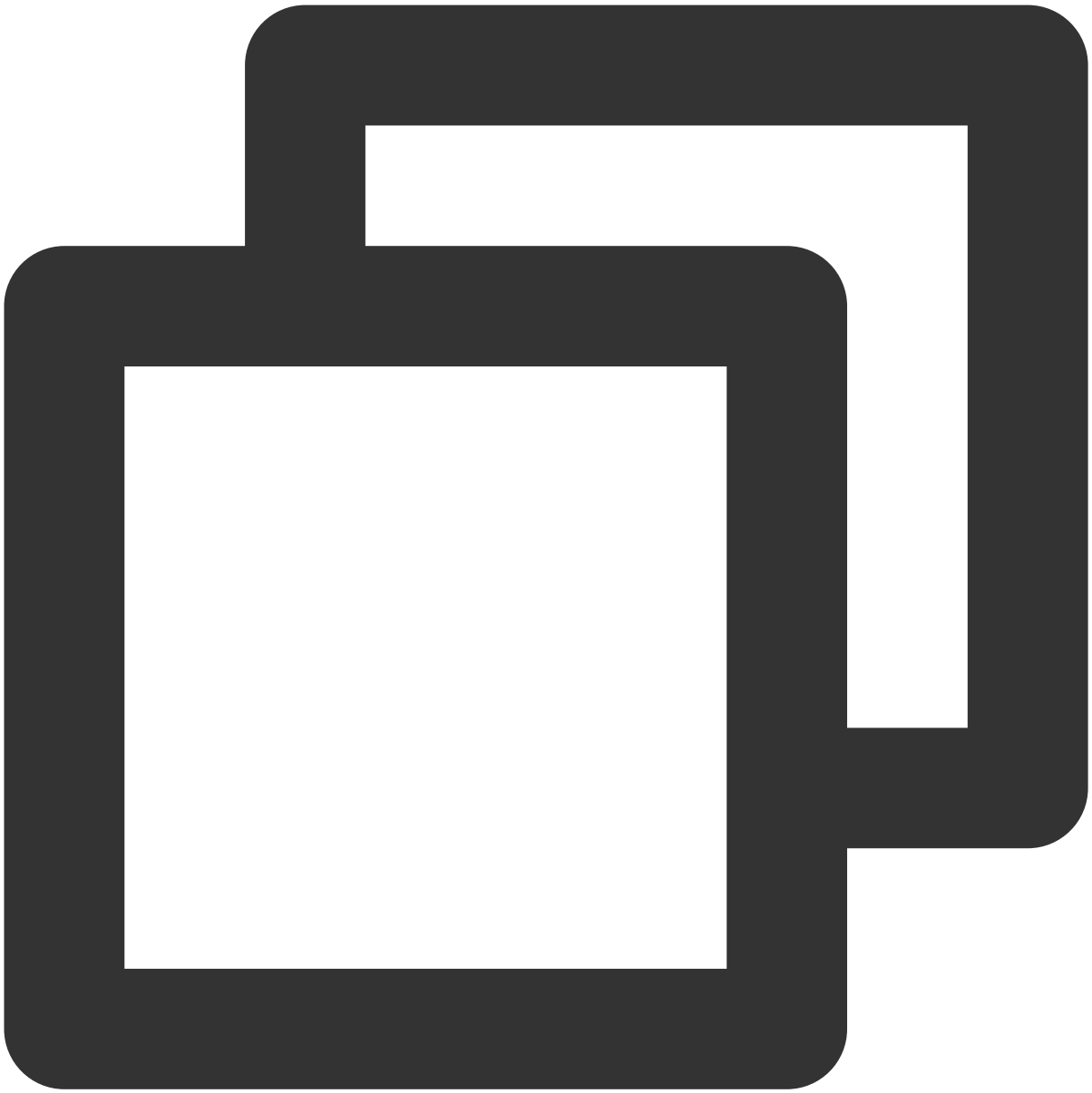
Android（Kotlin）

Android（Java）

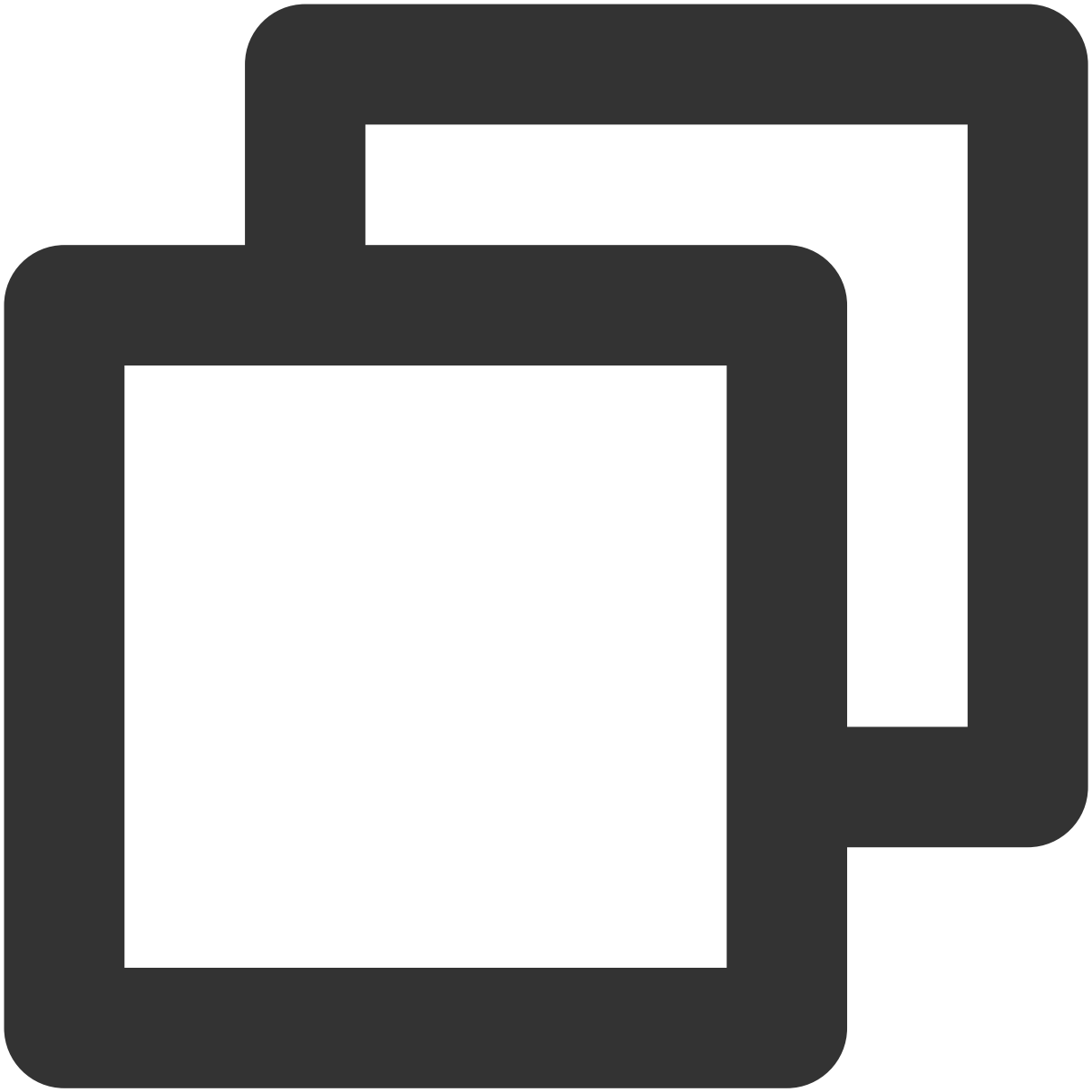
iOS（Swift）

iOS（Objective-C）

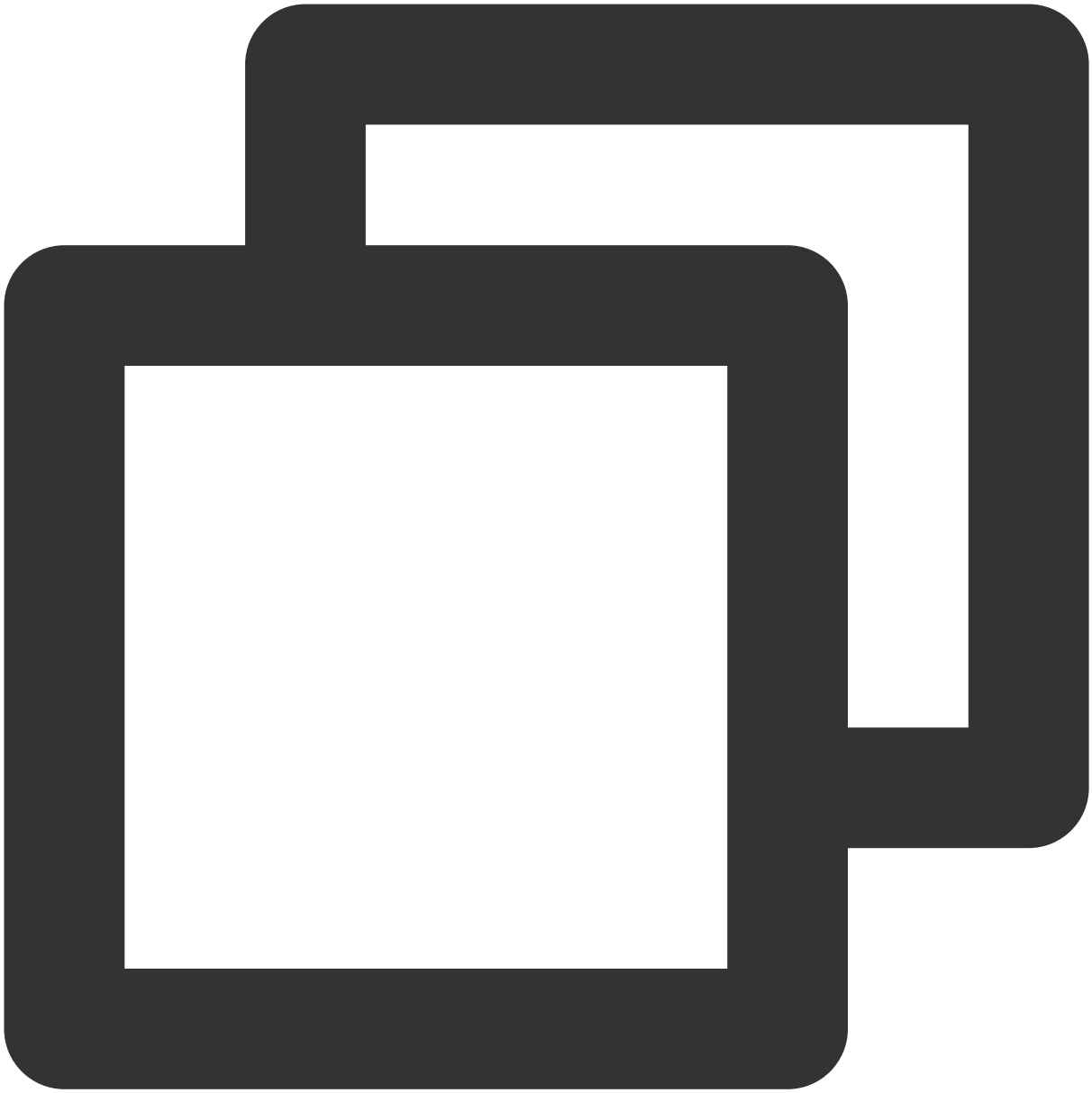
Flutter（Dart）



```
RoomId roomId = RoomId();
roomId.intRoomId = 123321;
TUICallKit.createInstance(context).joinInGroupCall(roomId, "12345678", TUICallDefin
```

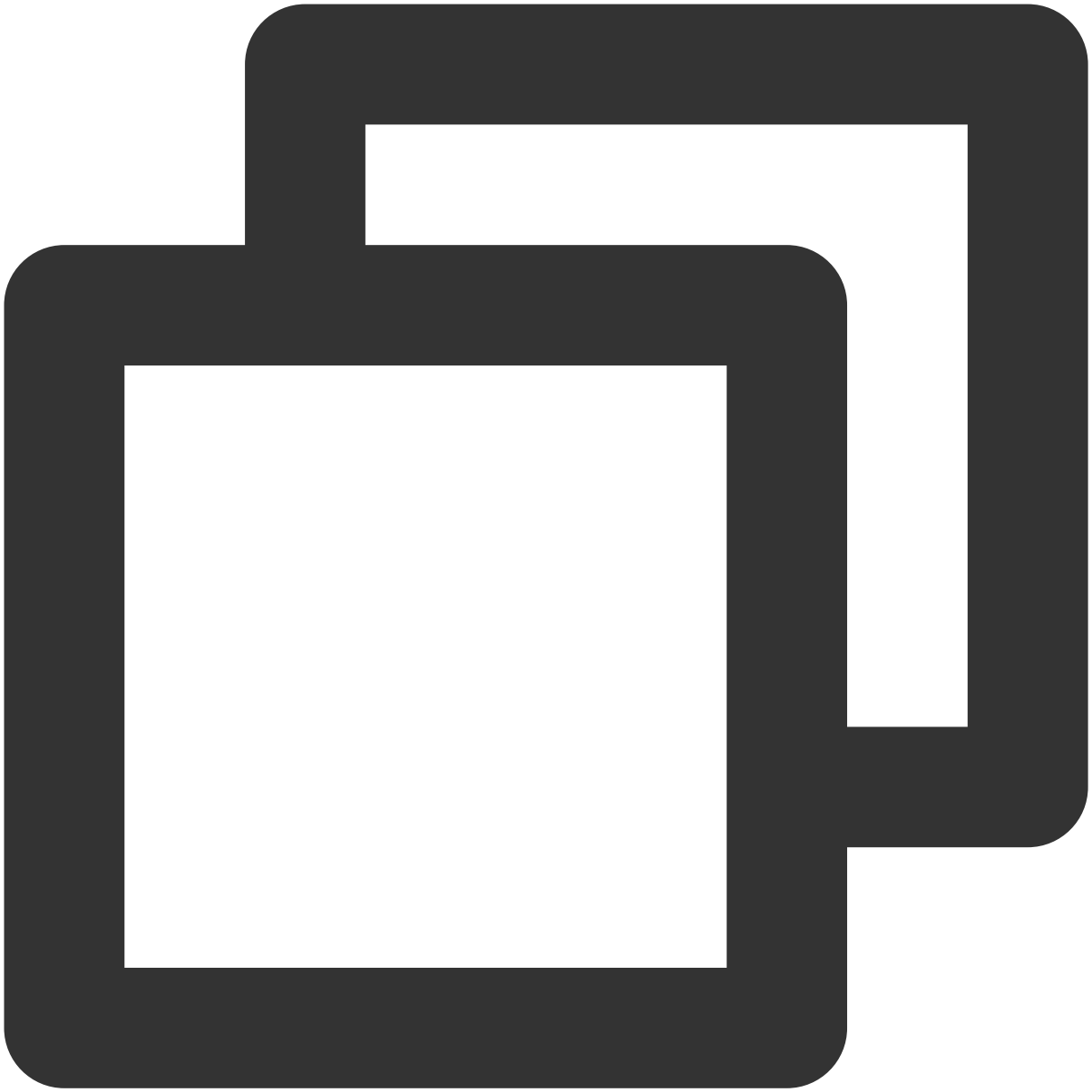


```
RoomId roomId = new RoomId();
roomId.intRoomId = 123321;
TUICallKit.createInstance(context).joinInGroupCall(roomId , "12345678", TUICallDefi
```



```
import TUICallKit
let roomId = TUIRoomId()
roomId.intRoomId = 123321
TUICallKit.createInstance().joinInGroupCall(roomId: roomId,
                                            groupId: "1234567",
                                            callMediaType: .video)
```

```
#import <TUICallKit/TUICallKit.h>

TUIRoomId *roomId = [[TUIRoomId alloc] init];
roomId.intRoomId = 123321;

[[TUICallKit createInstance] roomId: roomId
                            groupId:@"223344"
                         callMediaType:TUICallMediaTypeVideo];
```



```
TUIRoomId roomId = TUIRoomId()
roomId.intRoomId = 123321;
TUICallKit.instance.joinInGroupCall(roomId, '1234567', TUICallMediaType.video);
```

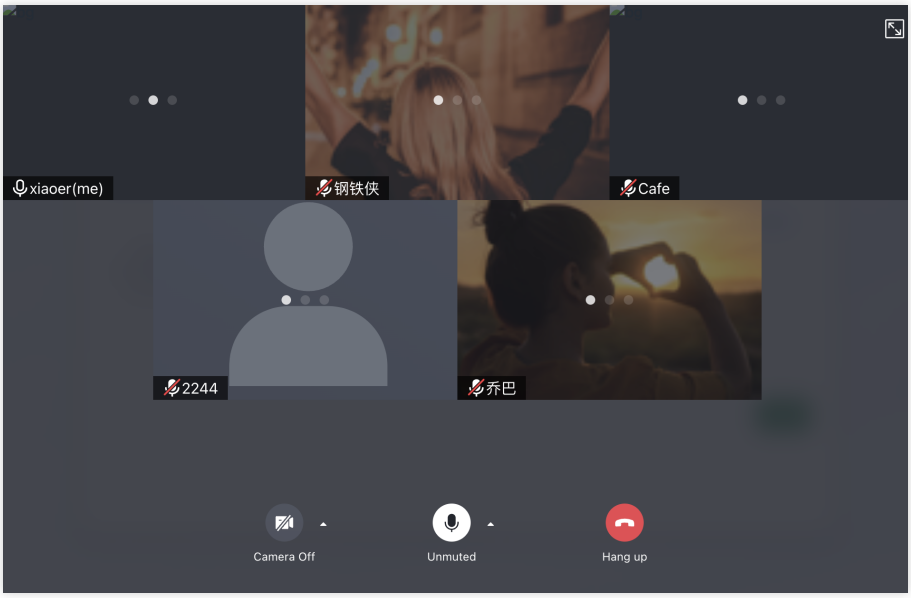
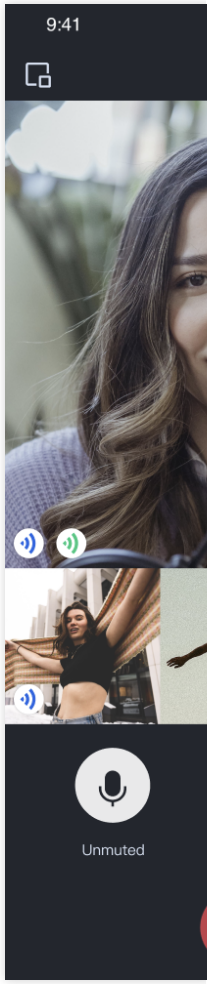
Web&H5

最近更新时间：2024-05-08 11:37:24

本文介绍群组通话功能的使用，如发起群组通话、加入群组通话。

预期效果

TUICallKit 支持群组通话，预期效果见下图。

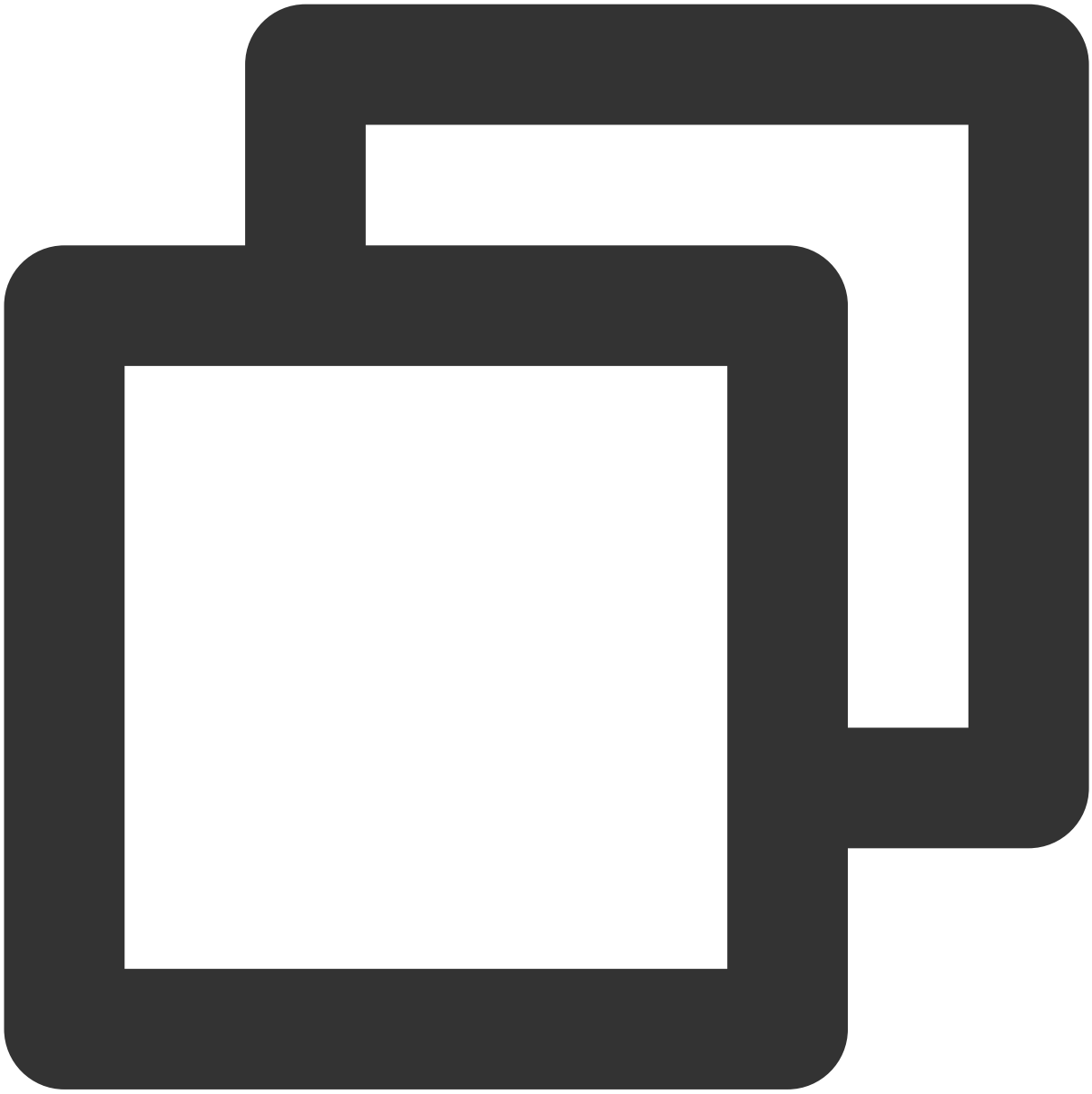
Web 端	移动端
	

创建 groupID

使用群组通话功能前，需要先创建群组，在已存在的群组中发起群组通话。

方式一：通过调用 IM API 创建群，详见 IM 群组管理。

方式二：通过控制台手动创建群组，详见 控制台群组管理。



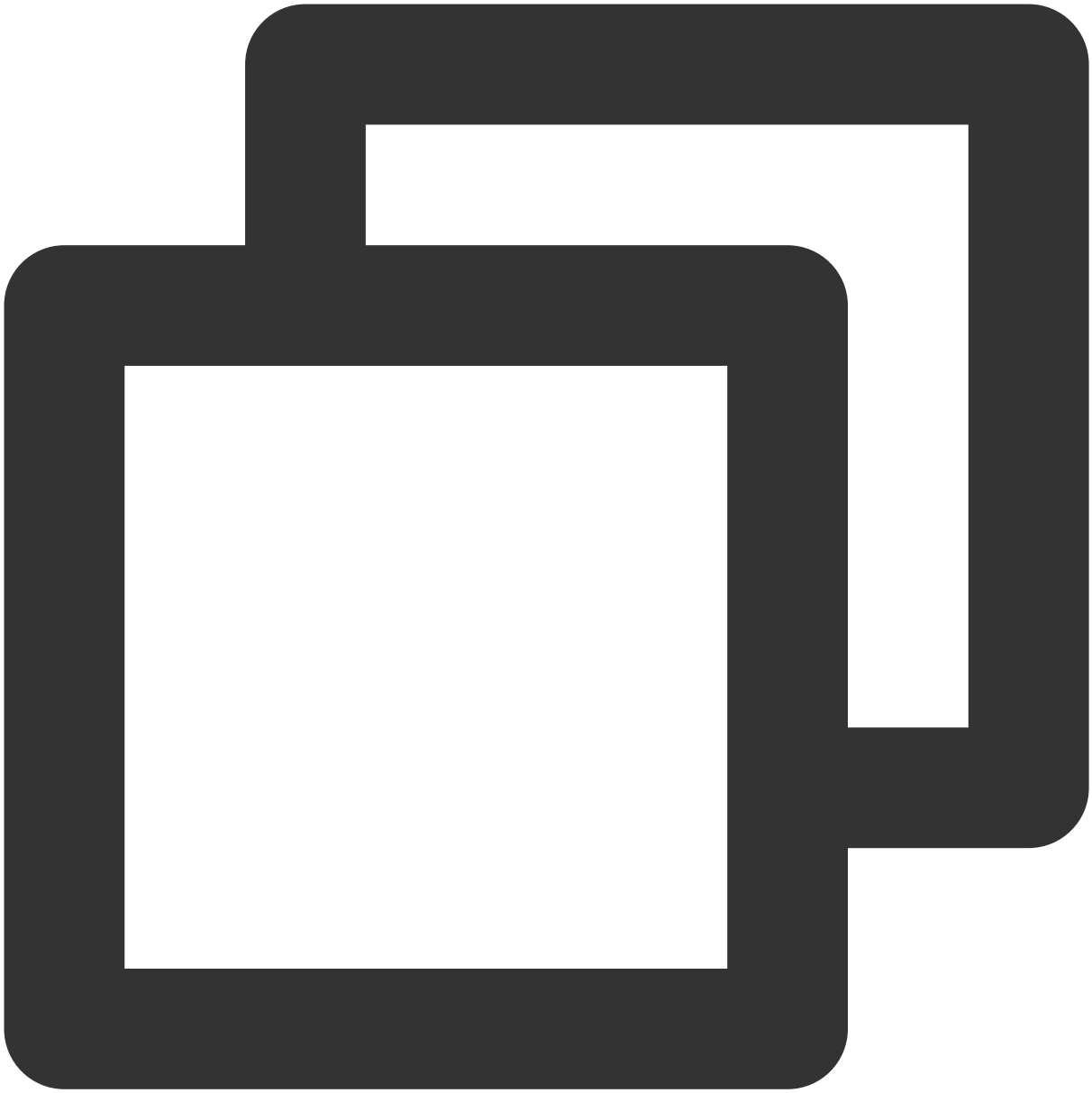
```
import TIM from "@tencentcloud/chat";   // npm i @tencentcloud/chat

const userIDList: string[] = ['user1', 'user2'];
async function createGroupID() {
  const tim = TIM.create({ SDKAppID });
  const memberList: any[] = userIDList.map(userId => ({ userID: userId }));
  const res = await tim.createGroup({
```

```
    type: TIM.TYPES.GRP_PUBLIC, // Must be a public group
    name: 'WebSDK',
    memberList
  });
  return res.data.group.groupID;
}
```

群组通话

发起群组通话

调用 `groupCall` API 发起群通话。



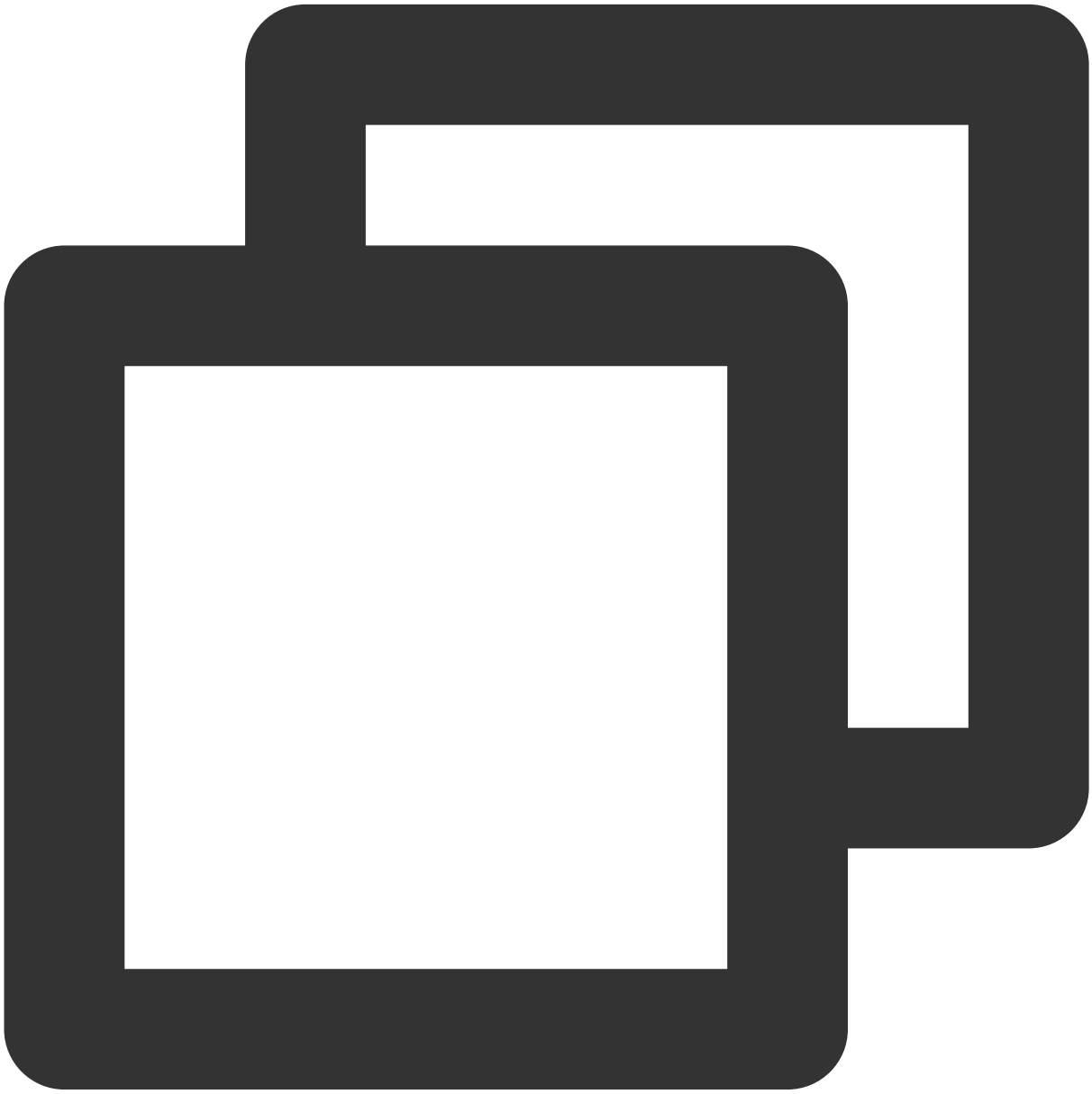
```
import { TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-react";
// Replace it with the call-uikit npm package you are currently using

try {
   const params = {
     userIDList: ['user1', 'user2'],
     groupID: 'xxx',
     type: TUICallType.VIDEO_CALL,
   }
  await TUICallKitServer.groupCall(params);
} catch (error: any) {
```

```
  alert(`[TUICallKit] groupCall failed. Reason:${error}`);
}
```

加入群组通话

调用 `joinInGroupCall` API 主动加入群组中已有的音视频通话。

说明：Vue ≥ v3.1.2 支持



```
import { TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-react";
// Replace it with the call-uikit npm package you are currently using
```

```
try {
  const params = {
    type: TUICallType.VIDEO_CALL,
    groupID: "xxx",
    roomID: 0,
  };
  await TUICallKitServer.joinInGroupCall(params);
} catch (error: any) {
  alert(`[TUICallKit] joinInGroupCall failed. Reason: ${error}`);
}
```

设置昵称、头像（全平台）

最近更新时间：2024-05-09 14:25:24

本文介绍如何设置用户的头像和昵称。

设置头像、昵称

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

Android（Kotlin）

Android（Java）

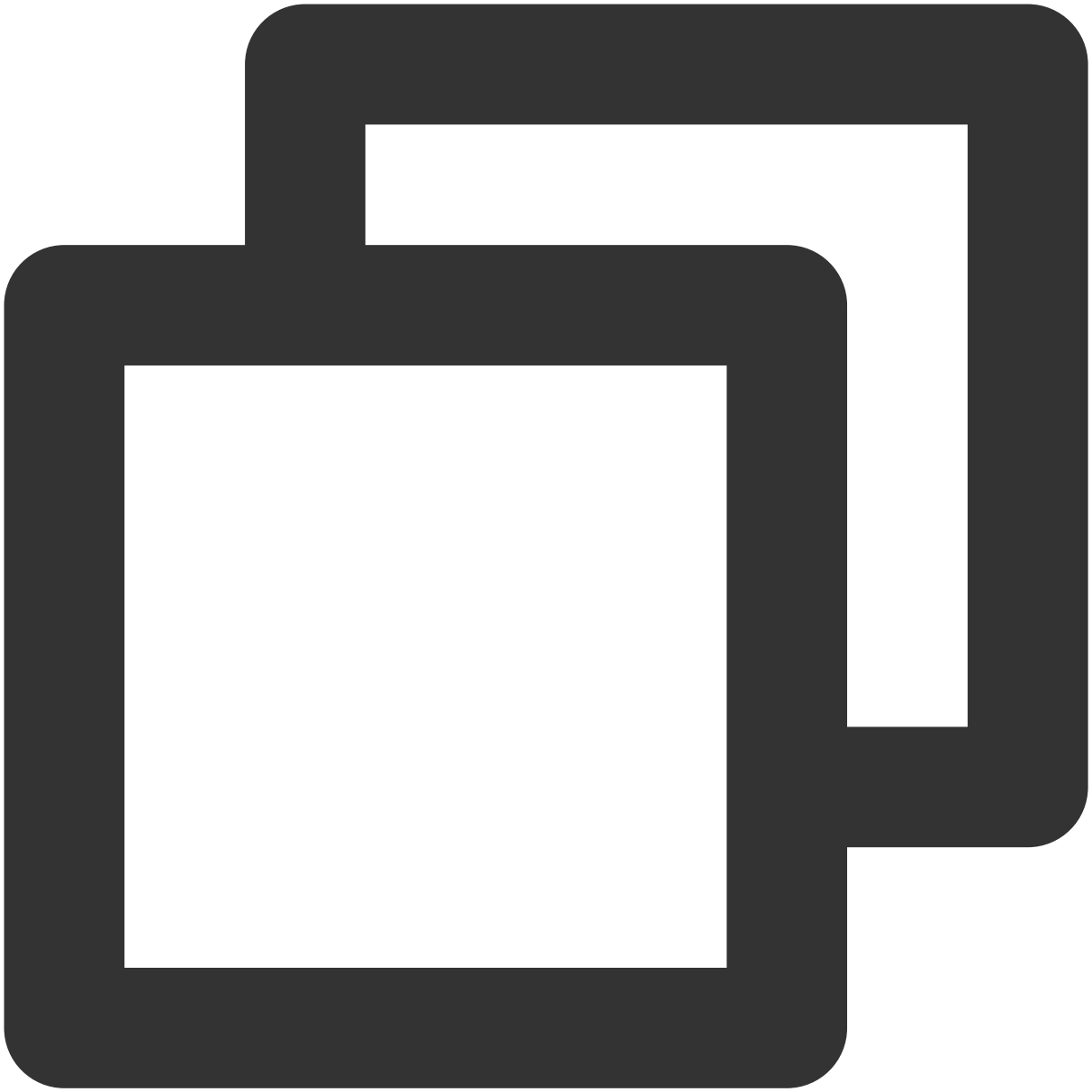
iOS（Objective-C）

iOS（Swift）

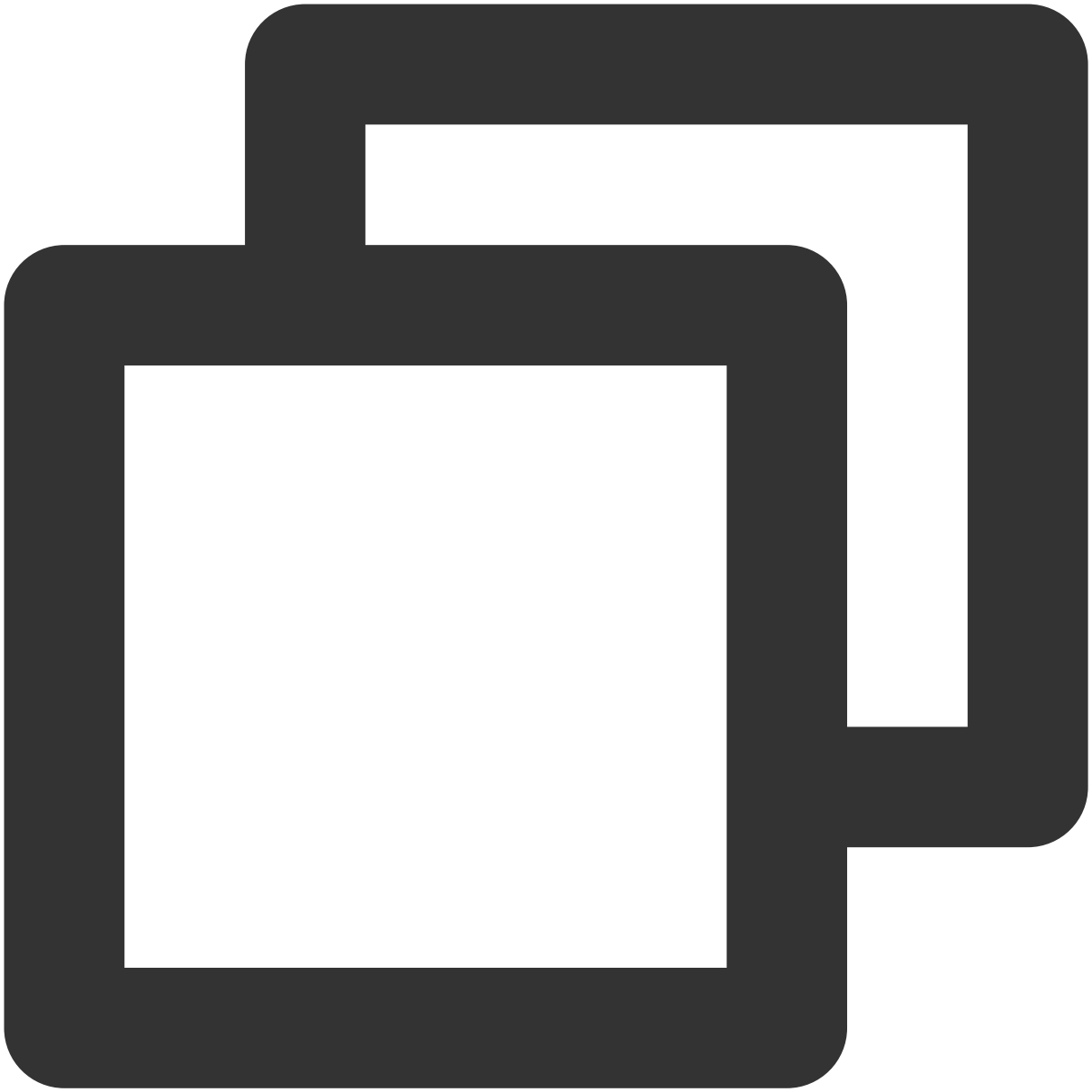
Flutter（Dart）

uni-app(Andorid&iOS)

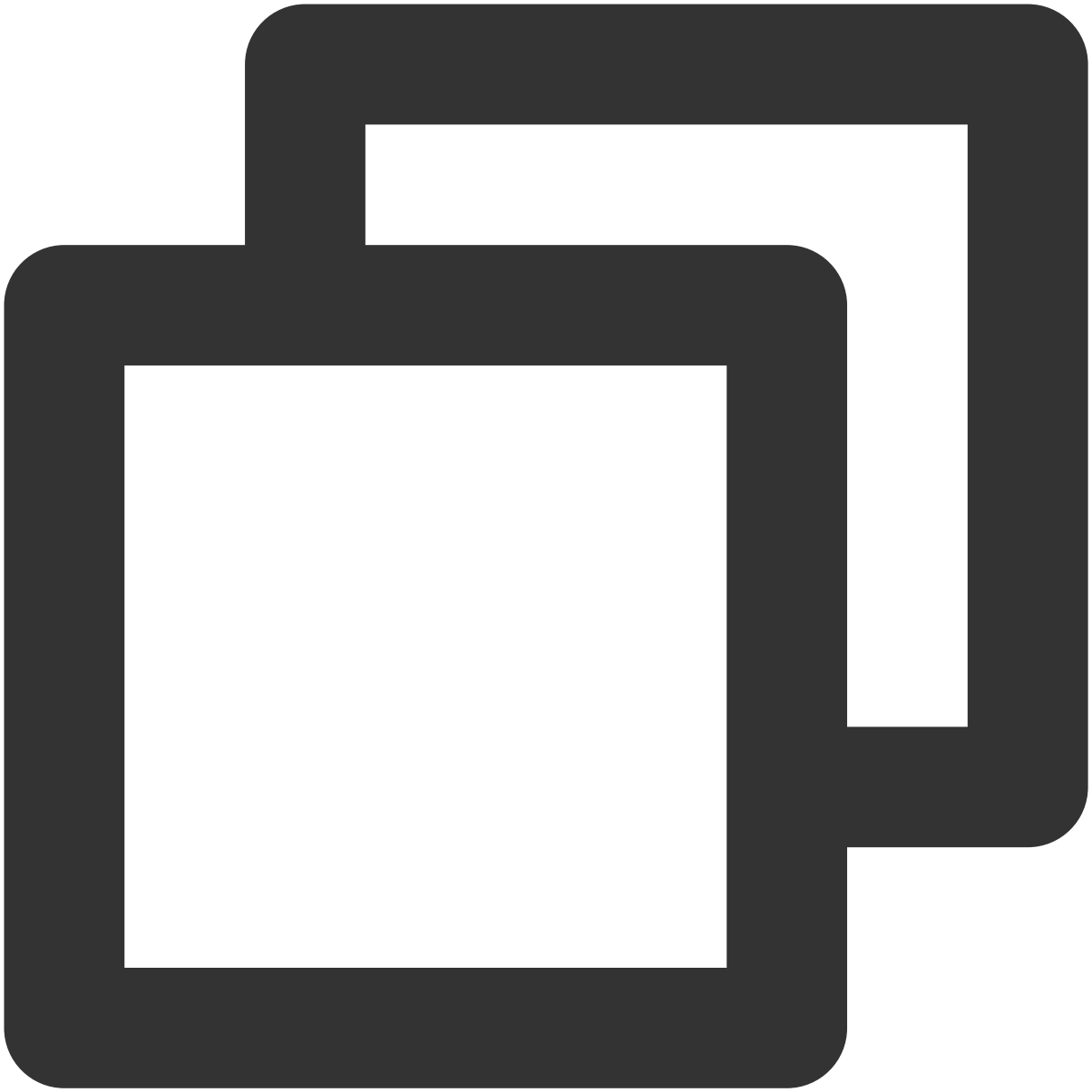
Web&H5



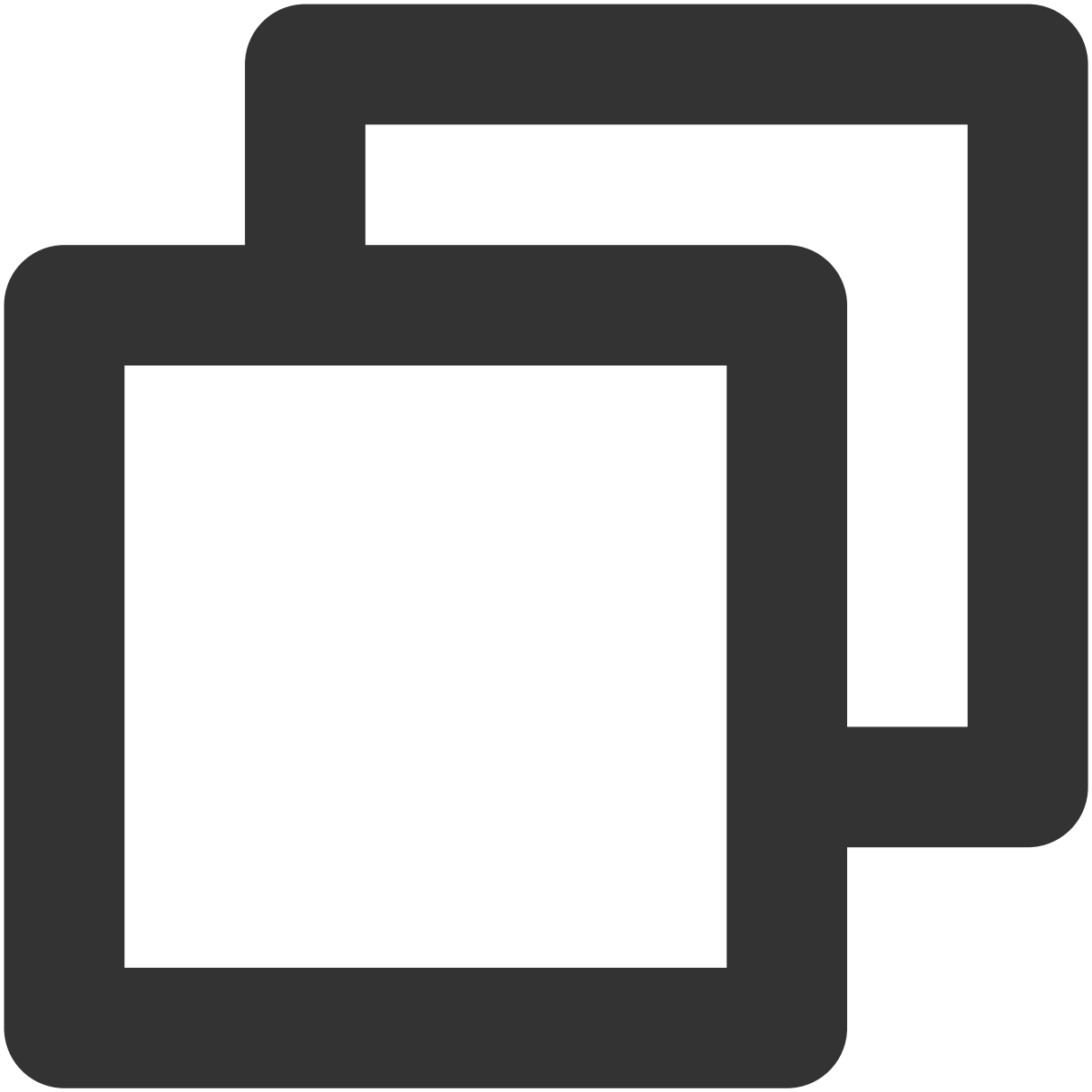
```
TUICallKit.createInstance(context).setSelfInfo("jack", "https:/****/user_avatar.png
```



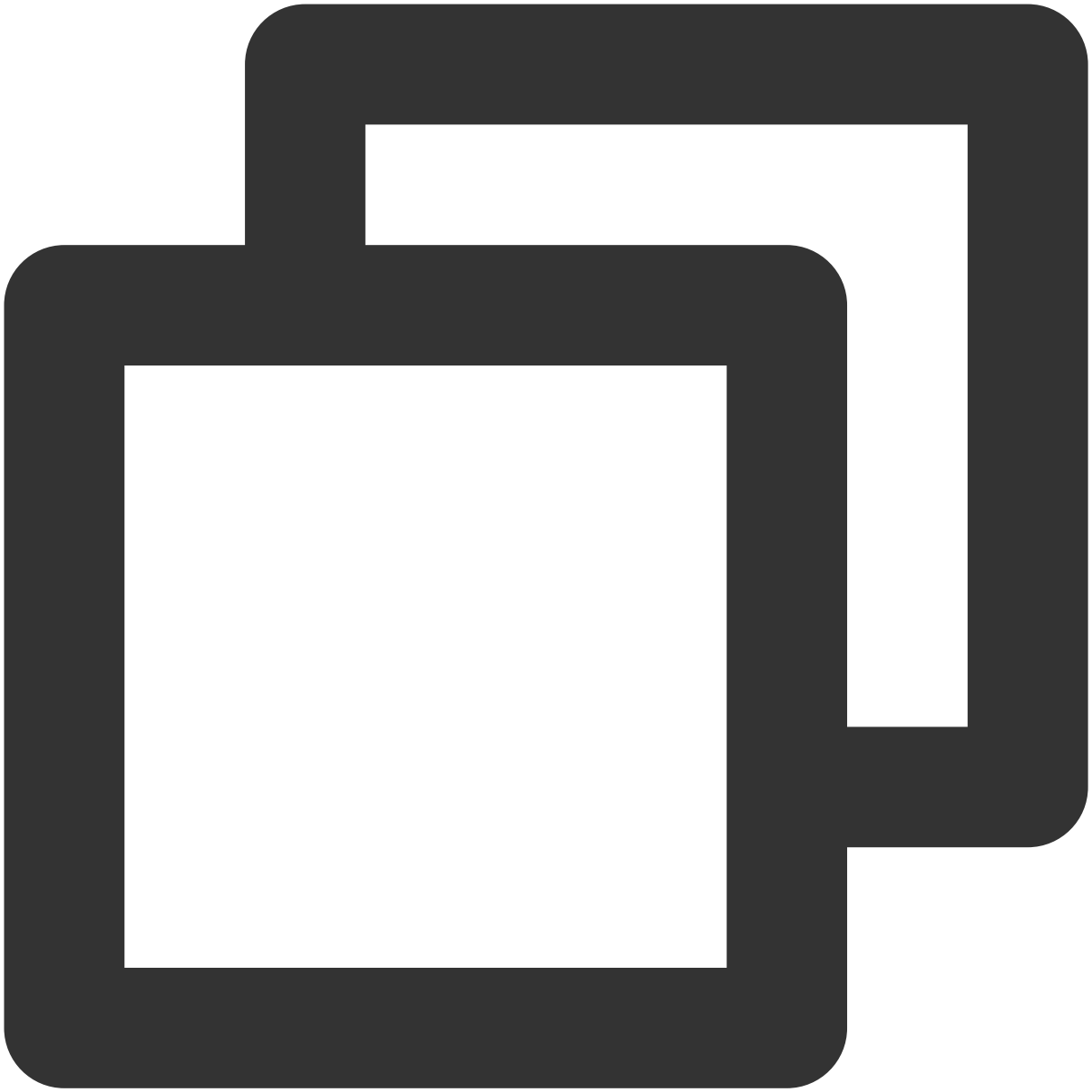
```
TUICallKit.createInstance(context).setSelfInfo("jack", "https:/****/user_avatar.png
```



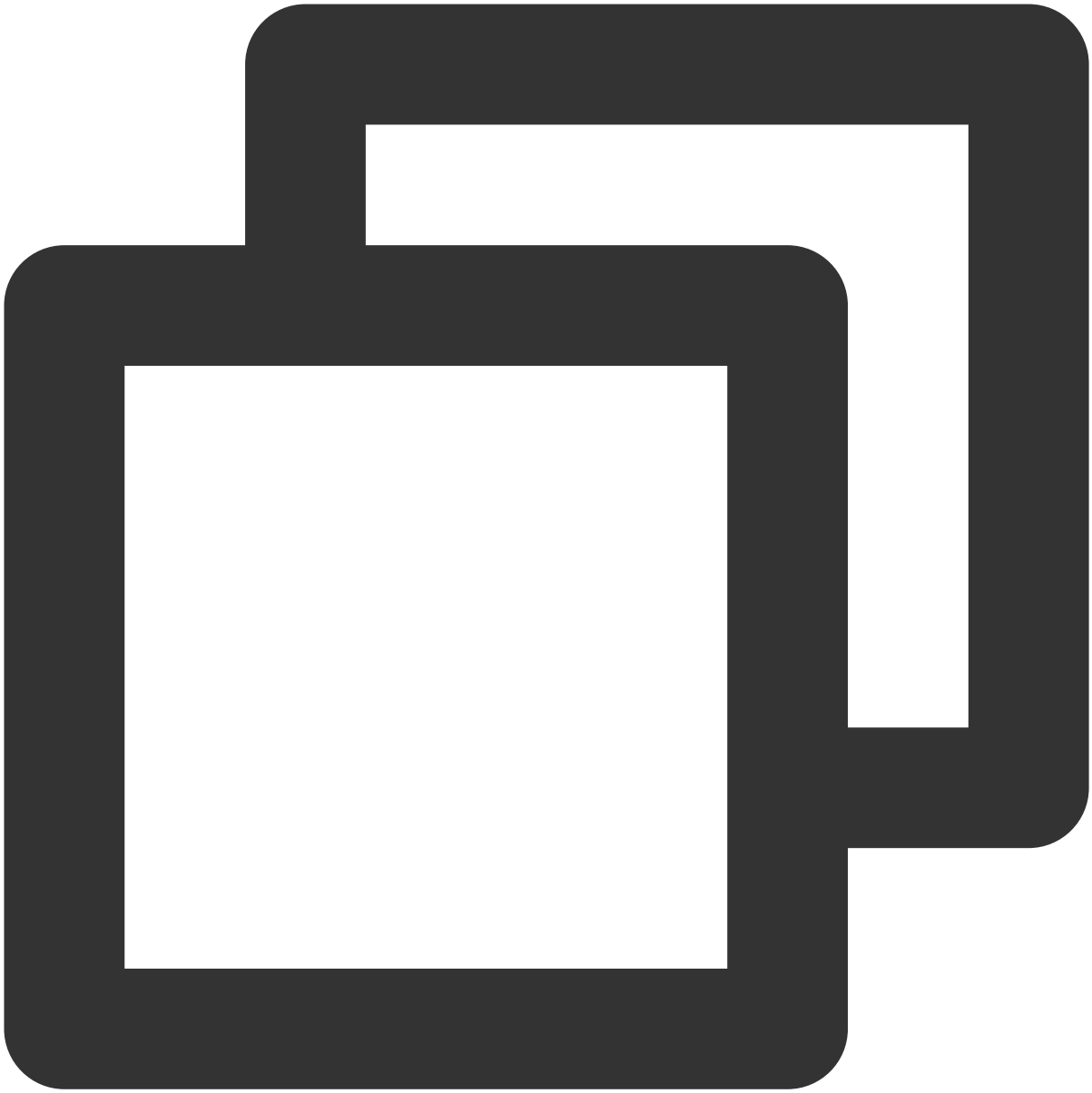
```
[[TUICallKit createInstance] setSelfInfo:@"" avatar:@"" succ:^{

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

}];
```



```
TUICallKit.createInstance().setSelfInfo(nickname: "", avatar: "") {

} fail: { code, message in

}
```



```
TUIResult result = TUICallKit.instance.setSelfInfo('userName', 'url:********');
```



```
const options = {
    nickName: 'jack',
    avatar: 'https:/****/user_avatar.png'
}
TUICallKit.setSelfInfo(options, (res) => {
    if (res.code === 0) {
        console.log('setSelfInfo success');
    } else {
        console.log(`setSelfInfo failed, error message = ${res.msg}`);
    }
});
```

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "jack", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`);
}
```

注意：

因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

设置分辨率和填充模式

最近更新时间：2024-05-08 11:37:24

本文介绍如何设置分辨率、填充模式。

设置分辨率、填充模式

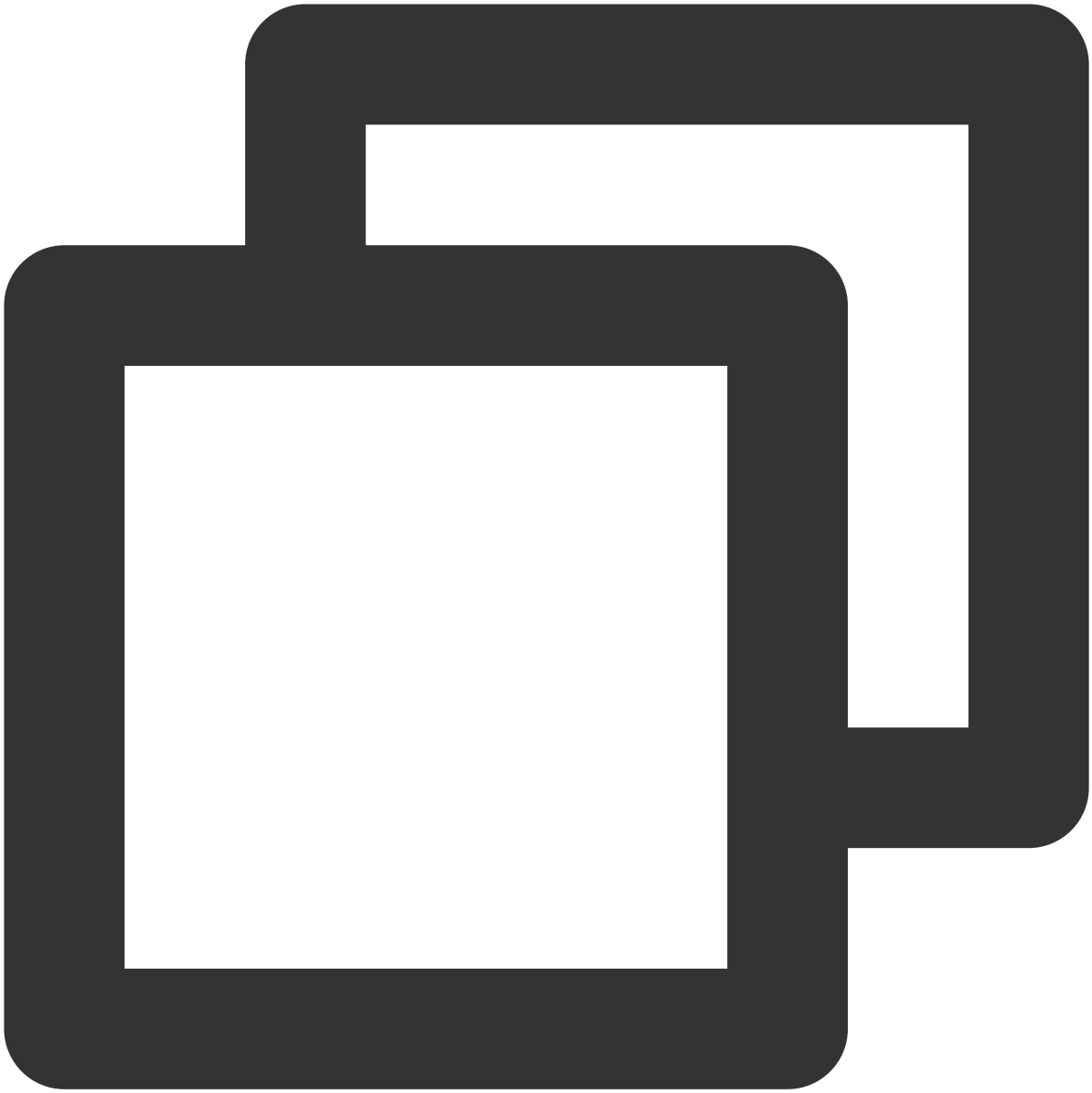
TUICallKit 组件提供了若干个功能开关，可以根据需要选择开启或关闭，具体参见 TUICallKit 属性介绍。

`videoDisplayMode` ：画面显示模式。

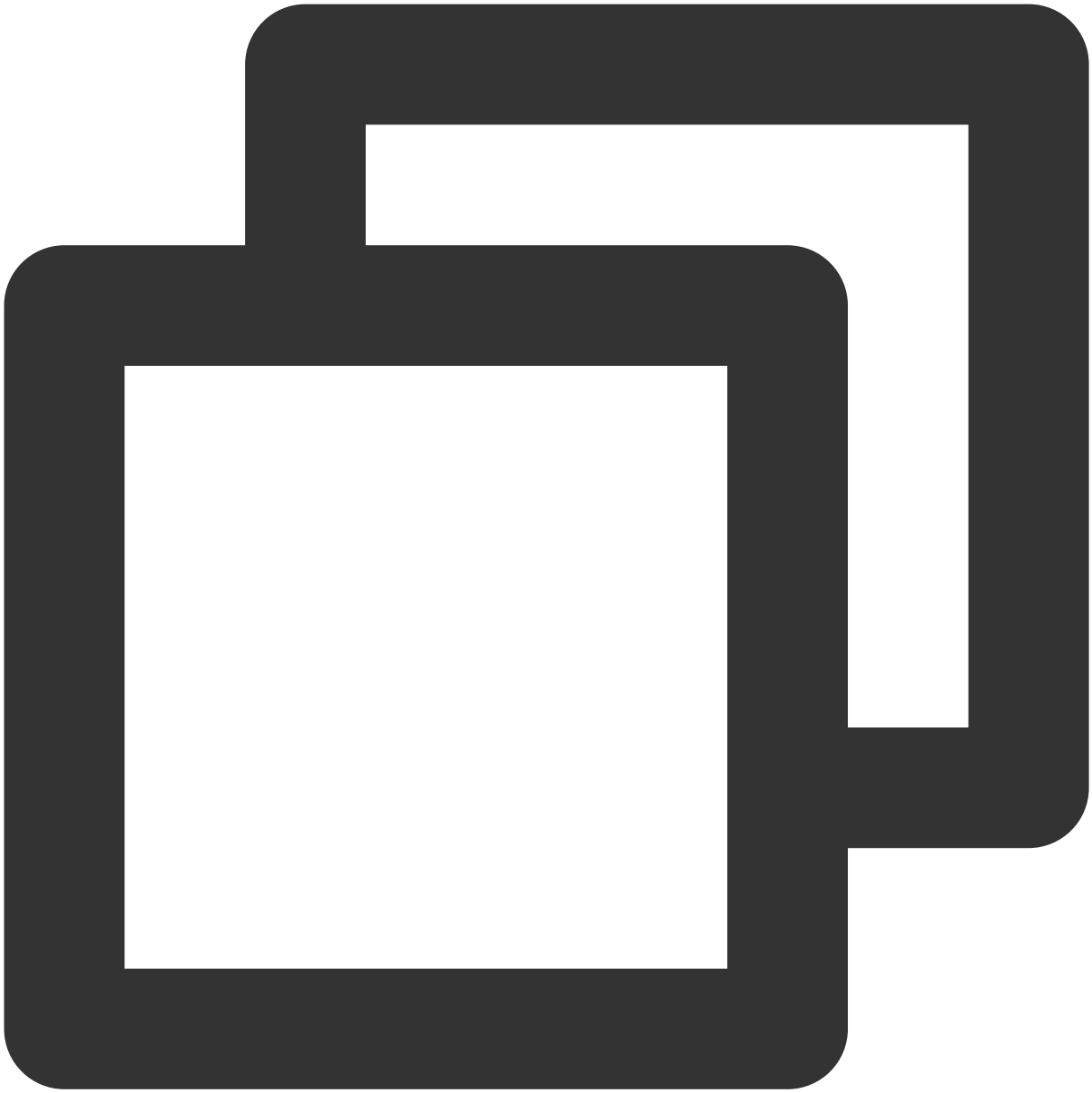
`videoResolution` ：通话分辨率。

React

Vue



```
import { VideoDisplayMode, VideoResolution } from "@tencentcloud/call-uikit-react";

<TUICallKit
    videoDisplayMode={VideoDisplayMode.CONTAIN}
    videoResolution={VideoResolution.RESOLUTION_1080P} />
```



```
import { VideoDisplayMode, VideoResolution } from "@tencentcloud/call-uikit-vue";

<TUICallKit
    :videoDisplayMode="VideoDisplayMode.CONTAIN"
    :videoResolution="VideoResolution.RESOLUTION_1080P" />
```

videoDisplayMode

画面显示模式 `videoDisplayMode` 属性有三个值：

`VideoDisplayMode.CONTAIN`

`VideoDisplayMode.COVER`

`VideoDisplayMode.FILL` ，默认值是 `VideoDisplayMode.COVER` 。

属性	值	描述
videoDisplayMode	VideoDisplayMode.CONTAIN	优先保证视频内容全部显示。 视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。 如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
	VideoDisplayMode.COVER	优先保证视窗被填满。 视频尺寸等比缩放，直至整个视窗被视频填满。 如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
	VideoDisplayMode.FILL	保证视窗被填满的同时保证视频内容全部显示，但是不保证视频尺寸比例不变。 视频的宽高会被拉伸至和视窗尺寸一致。

videoResolution

分辨率 `videoResolution` 有三个值：

`VideoResolution.RESOLUTION_480P`

`VideoResolution.RESOLUTION_720P`

`VideoResolution.RESOLUTION_1080P` ，默认值是 `VideoResolution.RESOLUTION_480P` 。

分辨率说明：

视频 Profile	分辨率（宽 × 高）	帧率（fps）	码率（kbps）
480p	640 × 480	15	900
720p	1280 × 720	15	1500
1080p	1920 × 1080	15	2000

常见问题：

iOS 13&14不支持编码高于 720P 的视频，建议在这两个系统版本限制最高采集 720P。参见 iOS Safari 已知问题 case 12。

Firefox 不支持自定义视频帧率（默认为 30fps）。

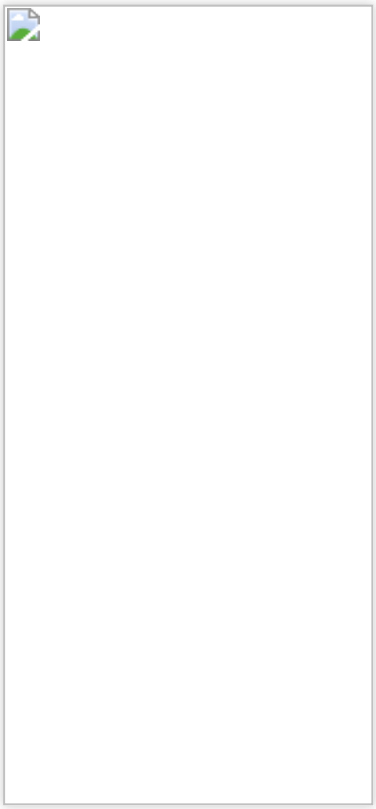
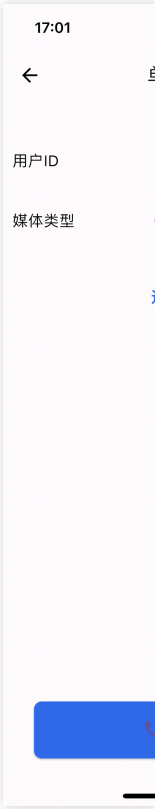
受系统性能占用，摄像头采集能力和浏览器限制等因素的影响，视频分辨率、帧率、码率的实际值不一定能够完全匹配设定值，在这种情况下，浏览器会自动调整 Profile 尽可能匹配设定值。

悬浮窗
Android&iOS&Flutter

最近更新时间：2024-05-08 11:37:24

本文介绍如何使用悬浮窗功能。

预期效果

开启悬浮窗按钮	语音通话悬浮窗	视频通话悬浮窗
		

悬浮窗功能

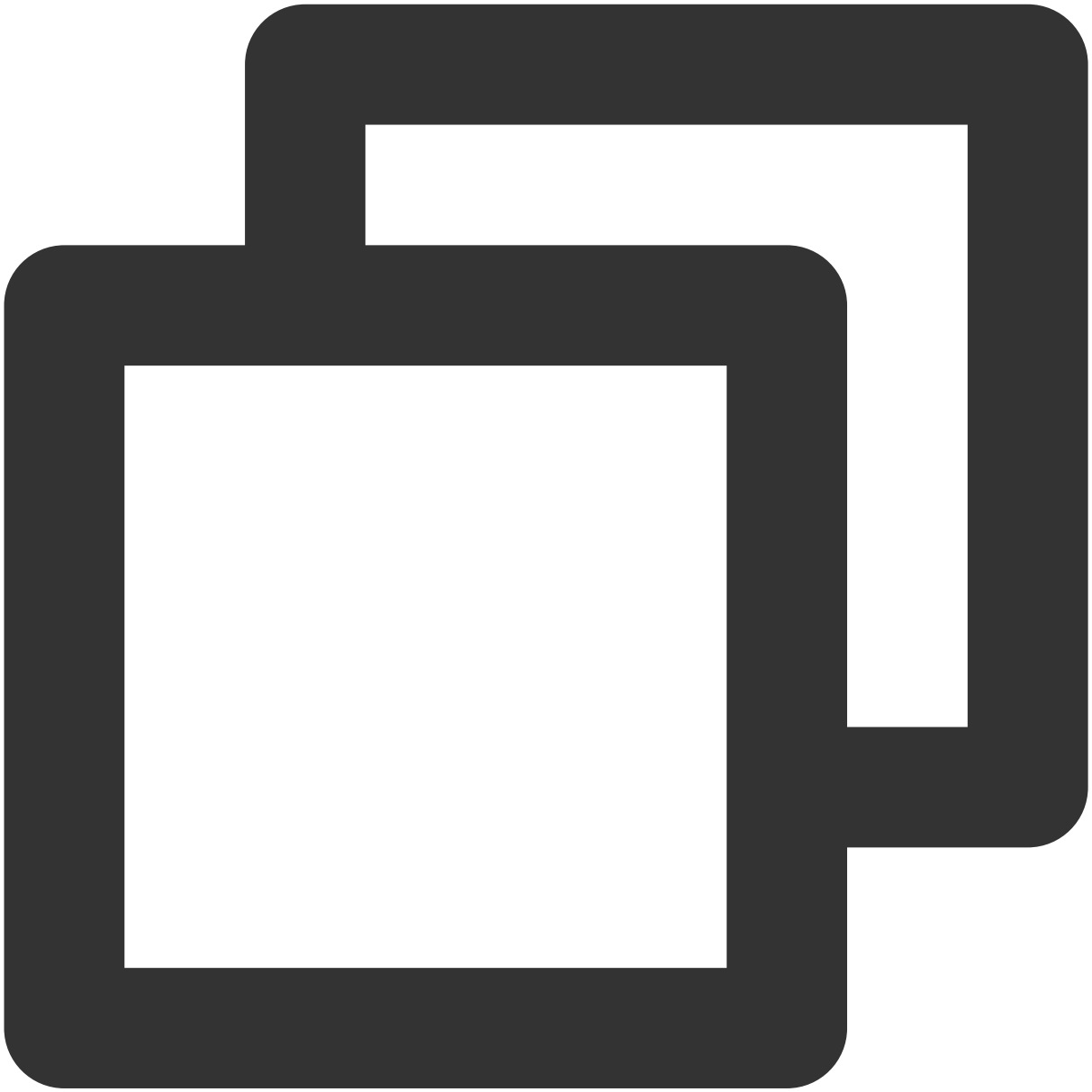
TUICallKit 允许用户在通话时使用通话界面左上角的悬浮窗按钮将通话界面缩小为悬浮窗。
您的业务如果需要启用该功能，可以使用 enableFloatWindow 方法，在 TUICallKit 组件初始化时开启该功能：
Android（Kotlin）

Android（Java）

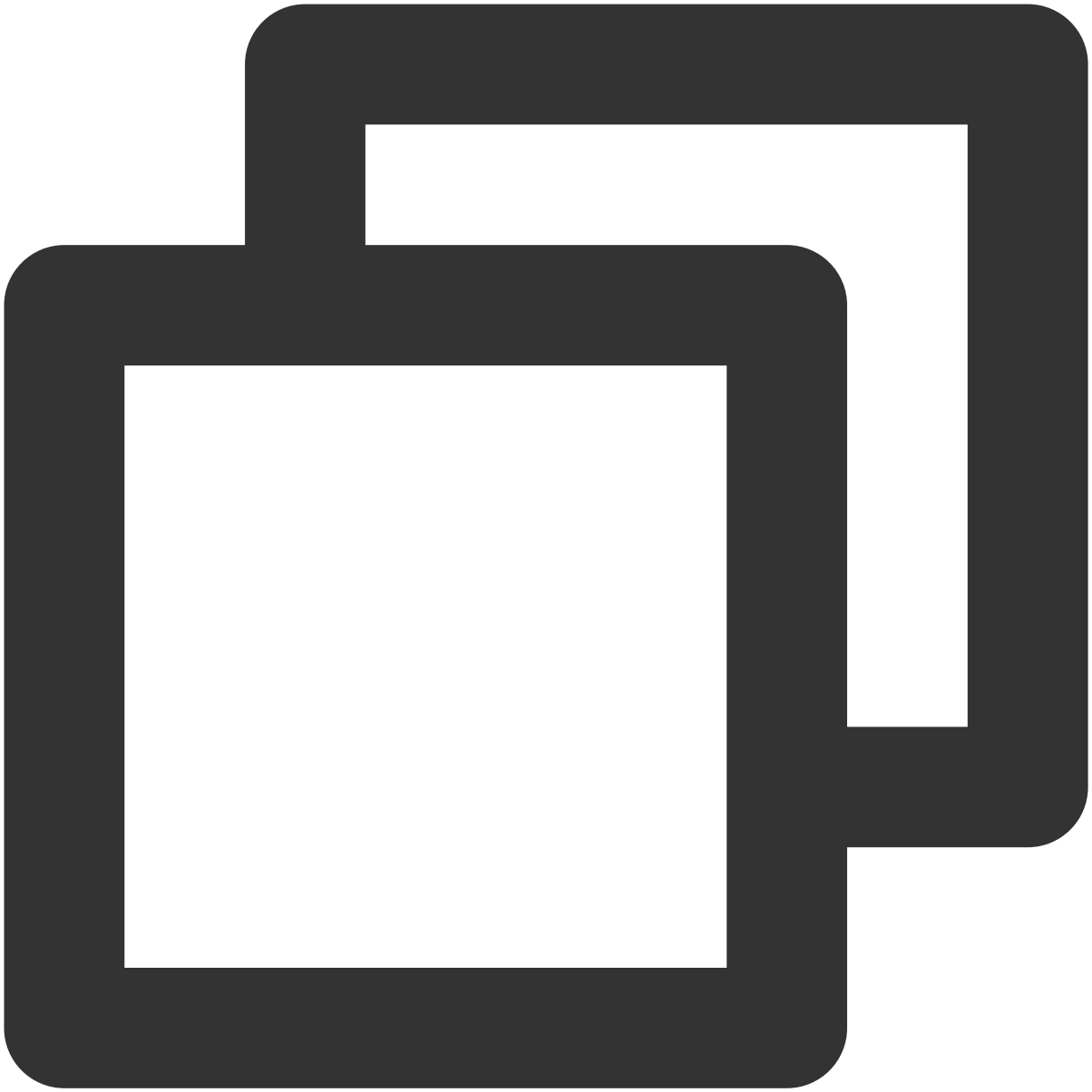
iOS（Swift）

iOS（Objective-C）

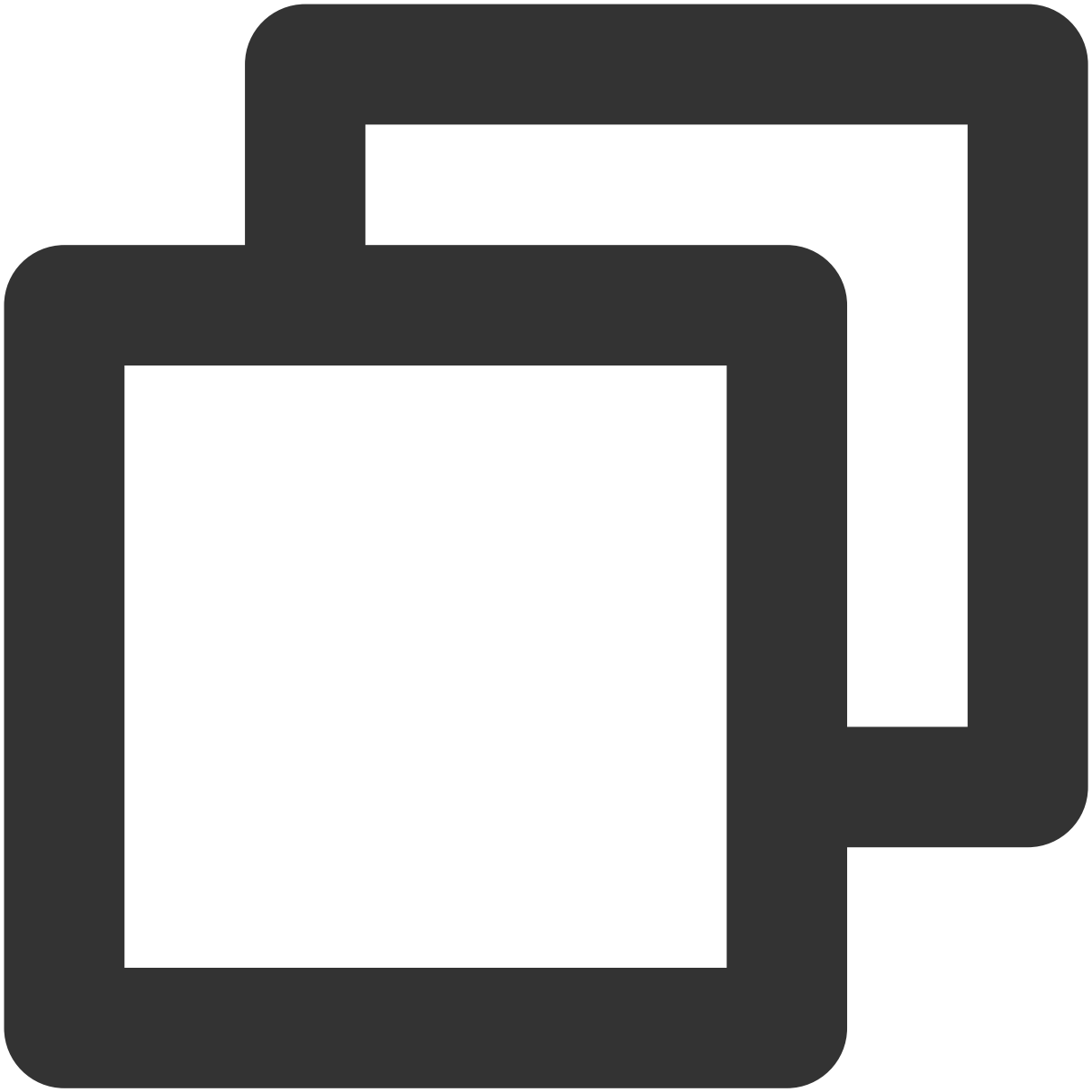
Flutter（Dart）



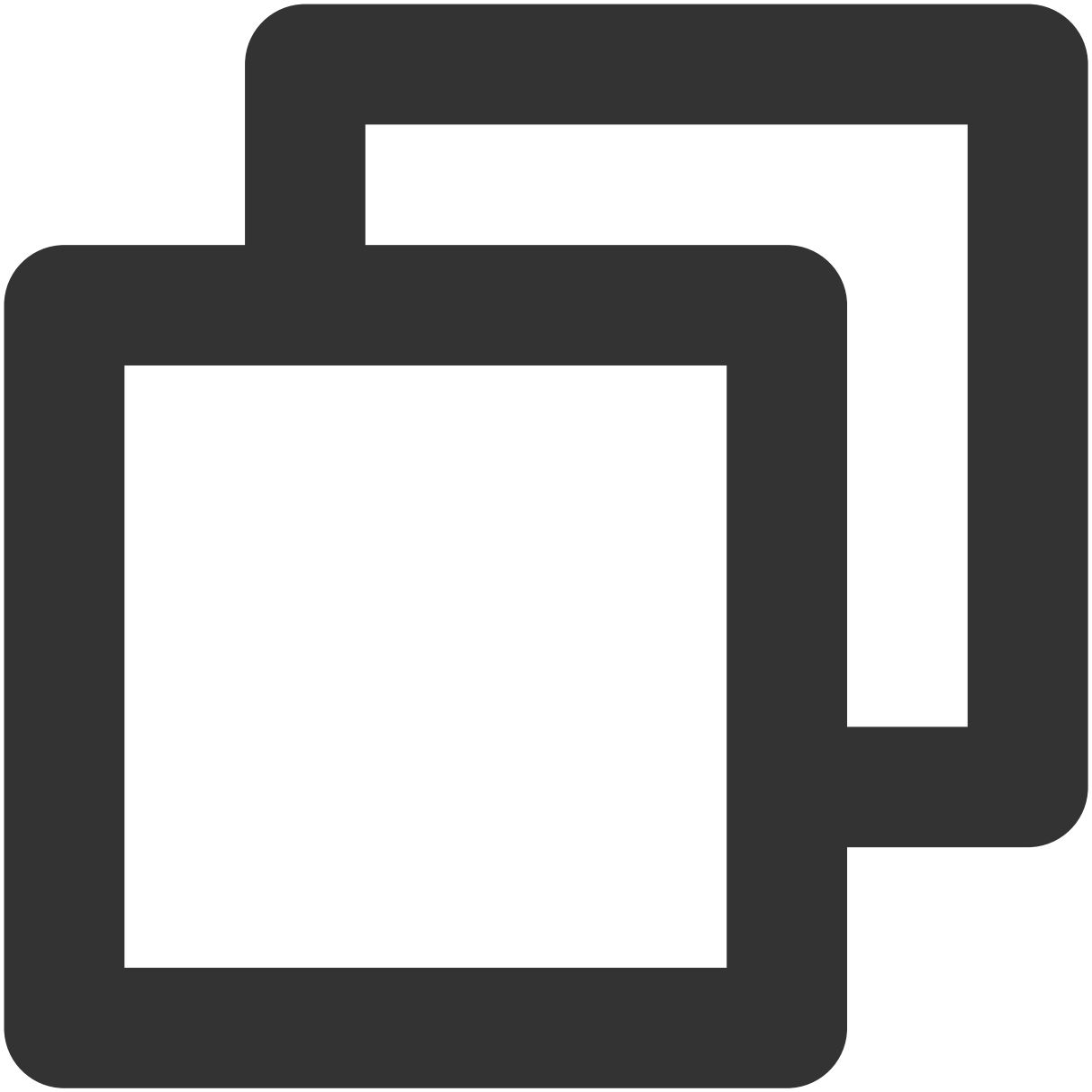
```
TUICallKit.createInstance(context).enableFloatWindow(true)
```



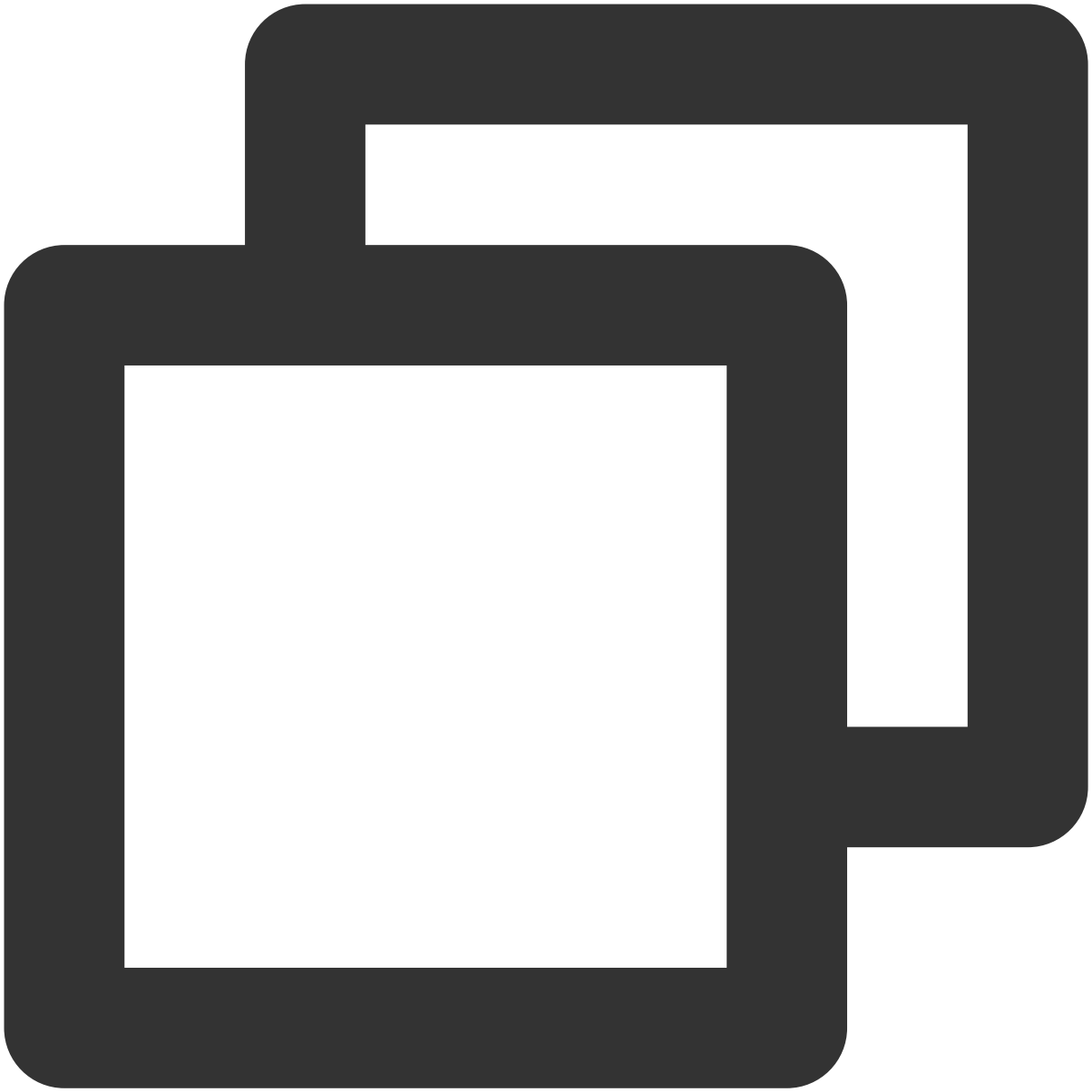
```
TUICallKit.createInstance(context).enableFloatWindow(true);
```



```
TUICallKit.createInstance().enableFloatWindow(true)
```



```
[[TUICallKit createInstance] enableFloatWindow:YES];
```



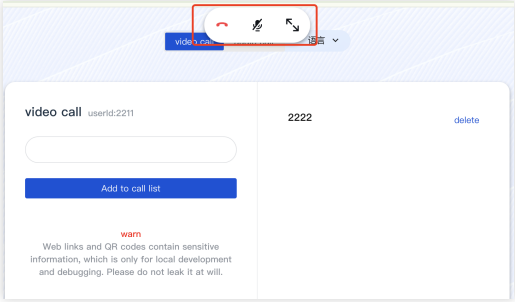
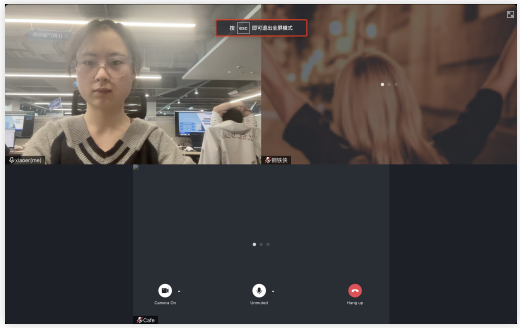
```
TUICallKit.instance.enableFloatWindow(true);
```

Web&H5

最近更新时间：2024-05-08 11:37:24

本文将介绍悬浮窗功能的使用。

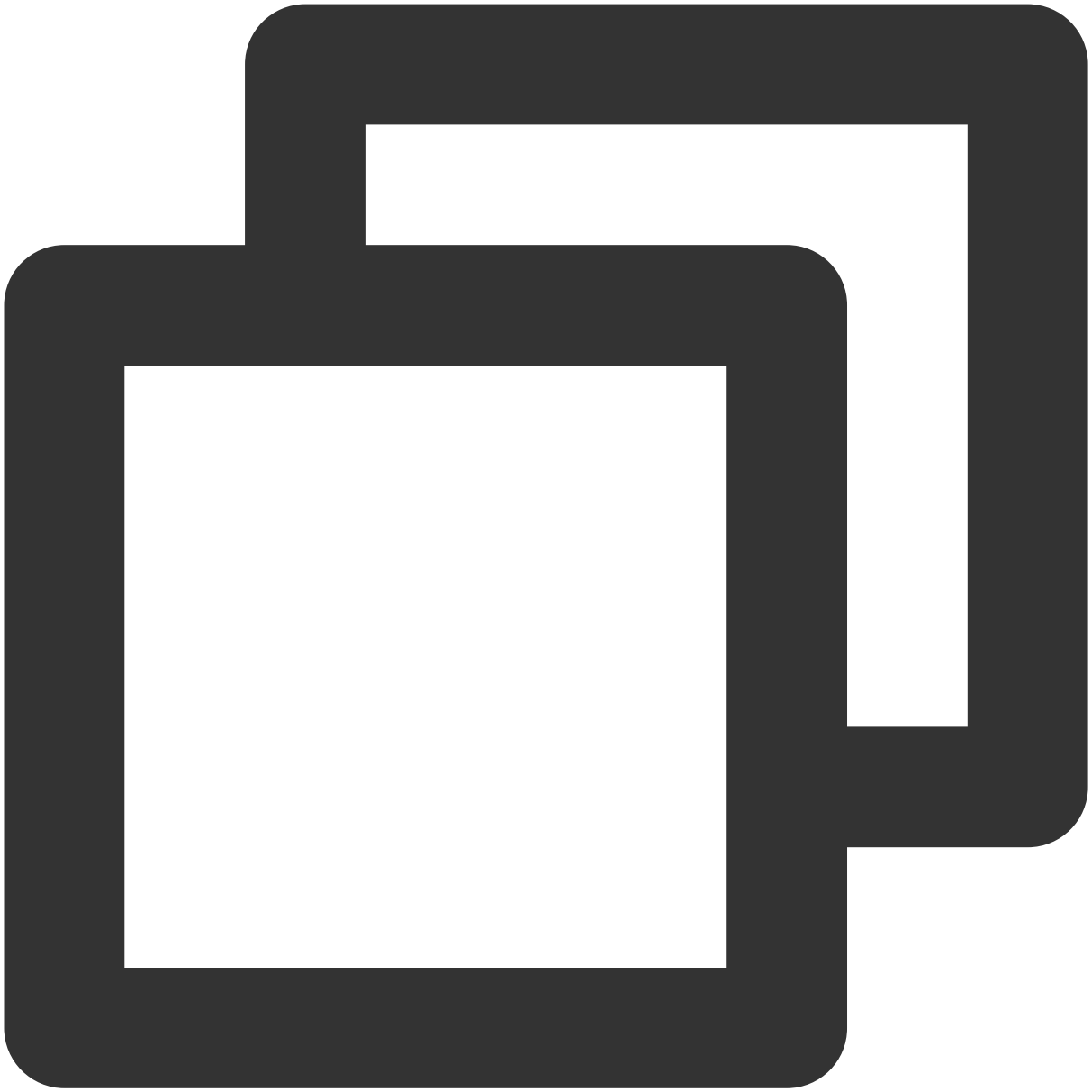
预期效果

Web 端悬浮窗	web 端全屏
	

悬浮窗功能

方式一：调用 `enableFloatWindow(enable: boolean)` API 开启/关闭悬浮窗。

说明：Vue ≥ v3.1.0 支持



```
try {
  await TUICallKitServer.enableFloatWindow(enable: Boolean)
} catch (error: any) {
  alert(`[TUICallKit] enableFloatWindow failed. Reason: ${error}`);
}
```

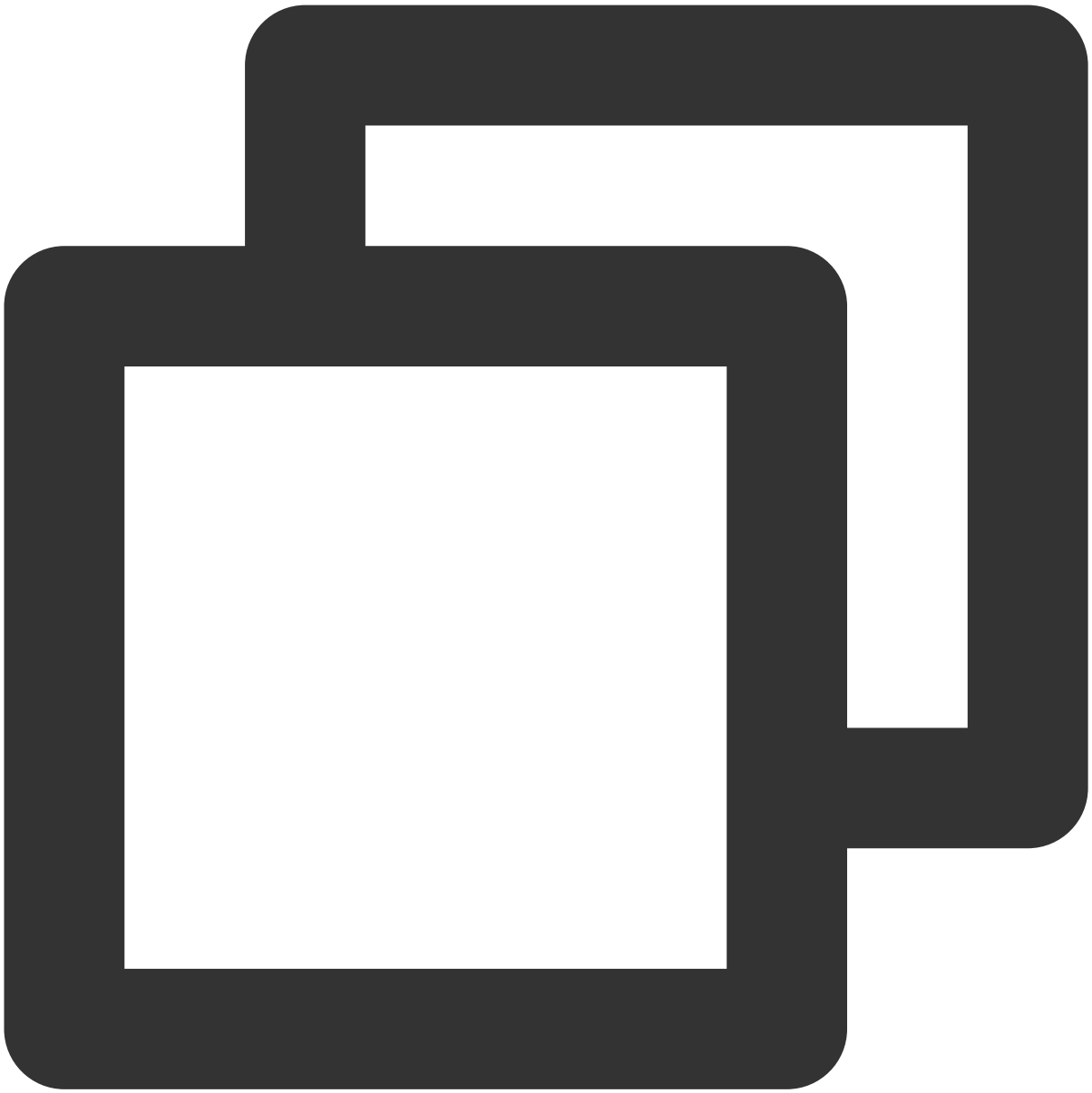
方式二：通过属性控制悬浮窗、全屏的开启/关闭。

`allowedMinimized` 属性控制控制悬浮窗开启/关闭。

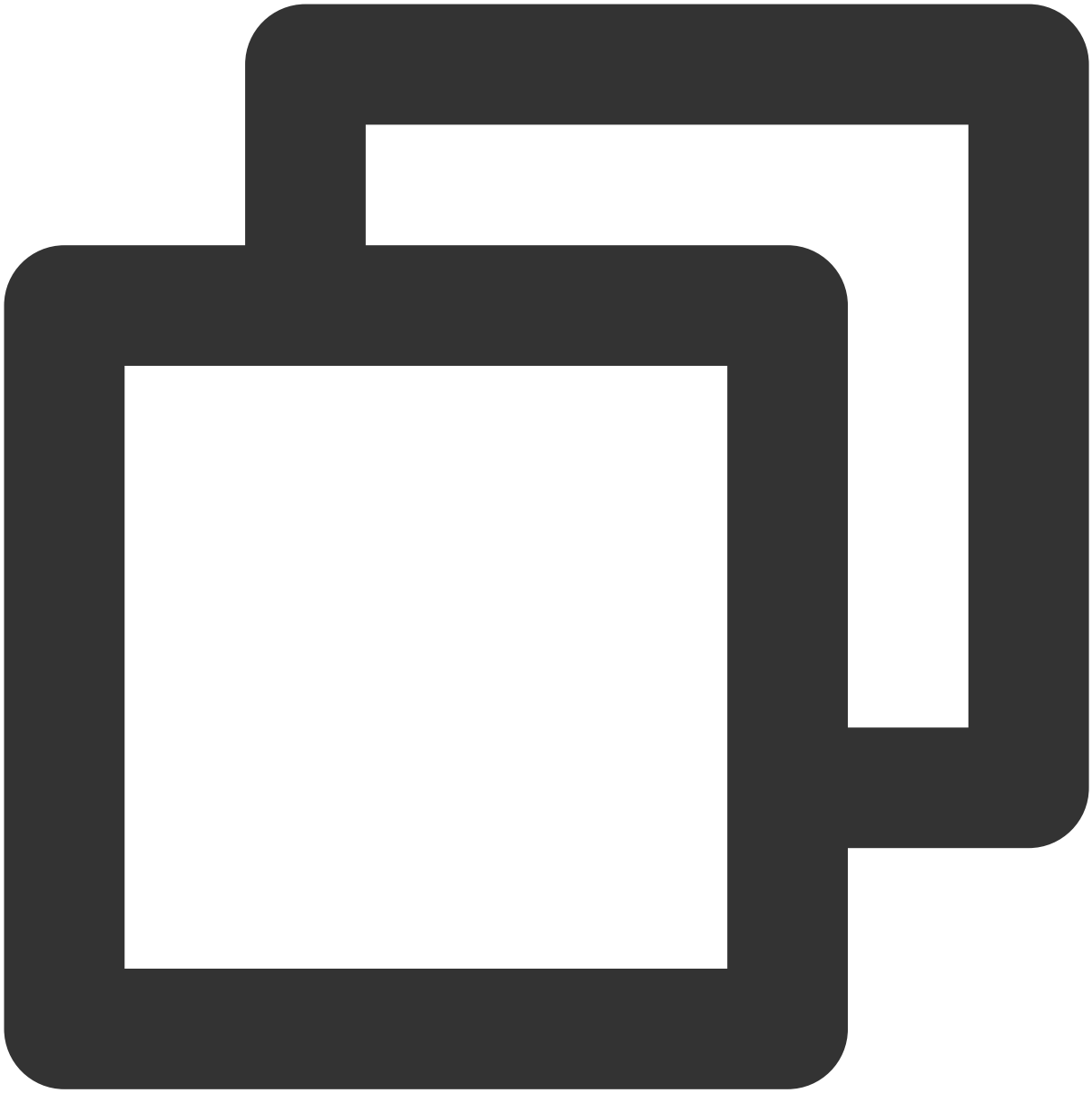
`allowedMinimized` 属性控制全屏的开启/关闭。

React

Vue



```
<TUICallKit
     allowedMinimized={true}
     allowedFullScree={true}
    />
```

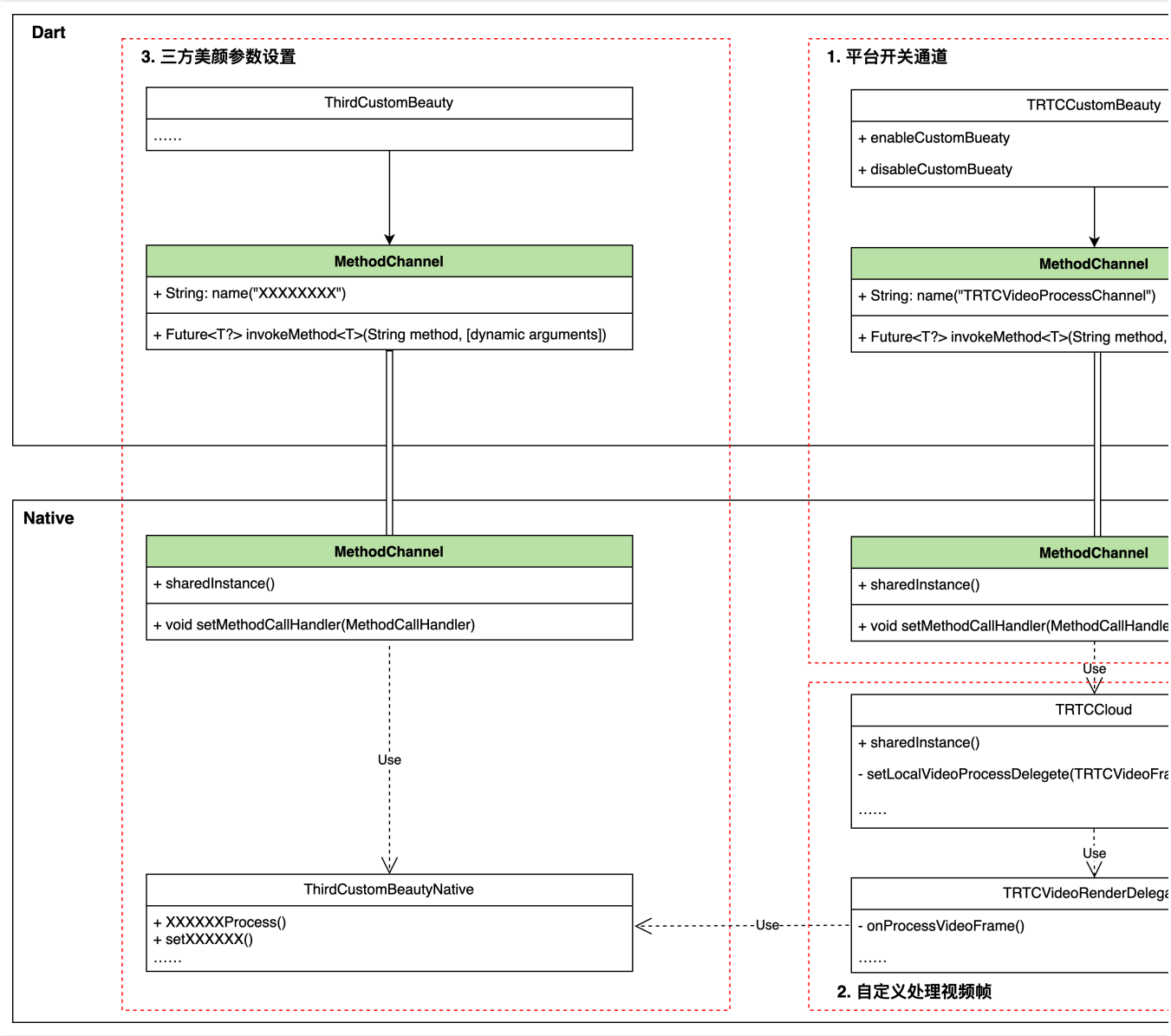


```
<TUICallKit
    :allowedMinimized="true"
    :allowedFullScreen="true"
   />
```

美颜特效（TUICallKit）
Flutter

最近更新时间：2024-03-20 16:26:15

本篇文档主要介绍在 TUICallKit 中接入美颜特效的方法。
在 Flutter 中完成自定义美颜处理，需要通过 TRTC 自定义视频渲染完成。由于 Flutter 不善于进行大量实时数据传输的特性，涉及 TRTC 的自定义视频渲染部分需要在 Native 部分完成。 具体的方案如下:



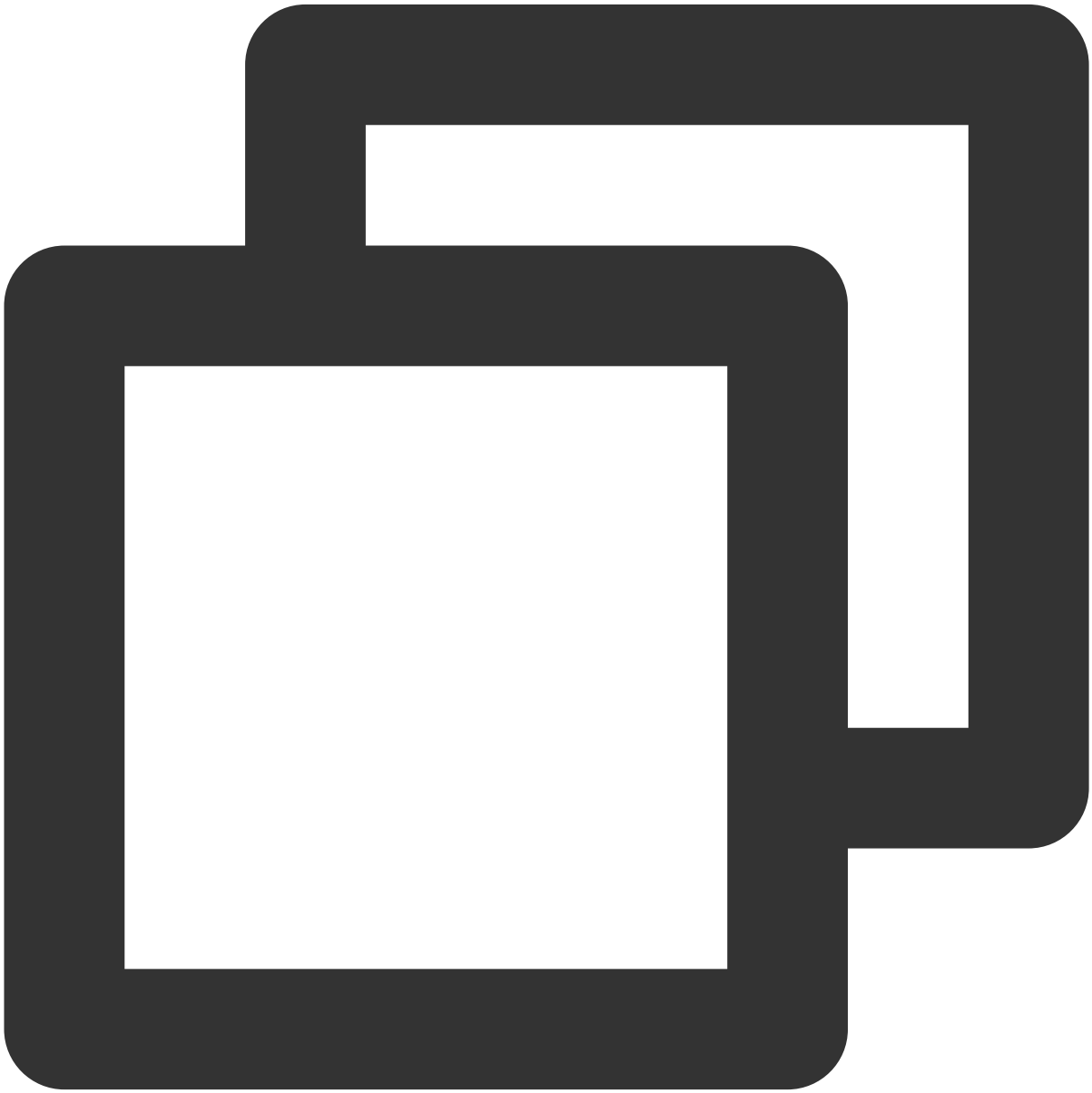
接入方案分为 3 步:
第一步：通过 MethodChannel 通道 开启/关闭 TRTC 自定义渲染逻辑；
第二步：在 TRTC 自定义渲染处理逻辑 onProcessVideoFrame() 中使用**美颜处理模块**对原始视频帧进行处理；

第三步：客户的美颜处理模块也需要在 Dart 通过接口设置当前美颜的参数，客户可以通过 MethodChannel 的方式设置美颜参数。这部分客户可根据自己的需求和使用的美颜自行定制。

接入第三方美颜特效

步骤1：实现 Dart 层到 Native 开始/关闭美颜的控制接口

实现 Dart 层接口:



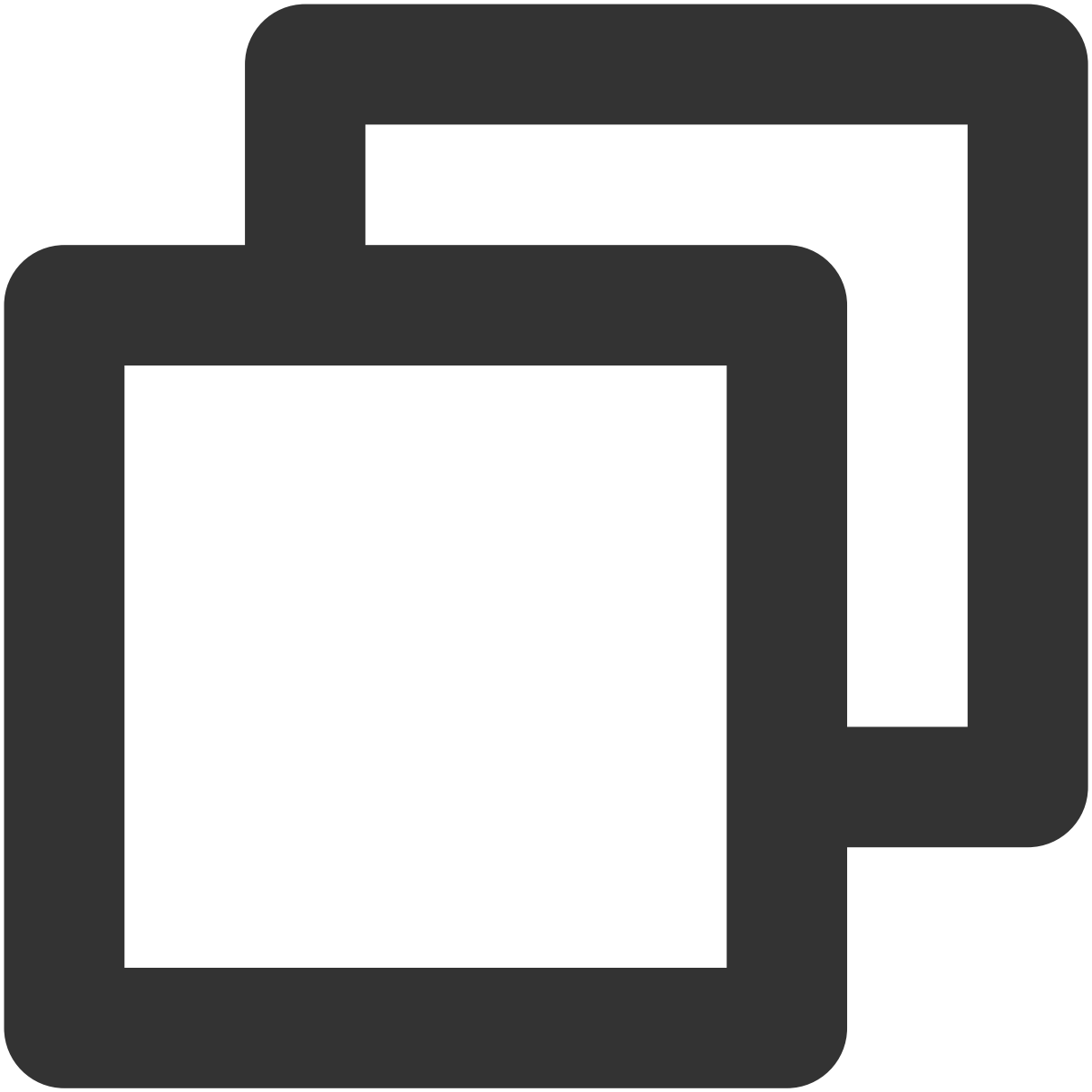
```
final channel = MethodChannel('TUICallKitCustomBeauty');
```

```
void enableTUICallKitCustomBeauty() async {
   await channel.invokeMethod('enableTUICallKitCustomBeauty');
}

void disableTUICallKitCustomBeauty() async {
   await channel.invokeMethod('disableTUICallKitCustomBeauty');
}
```

实现对应 Native 层接口:

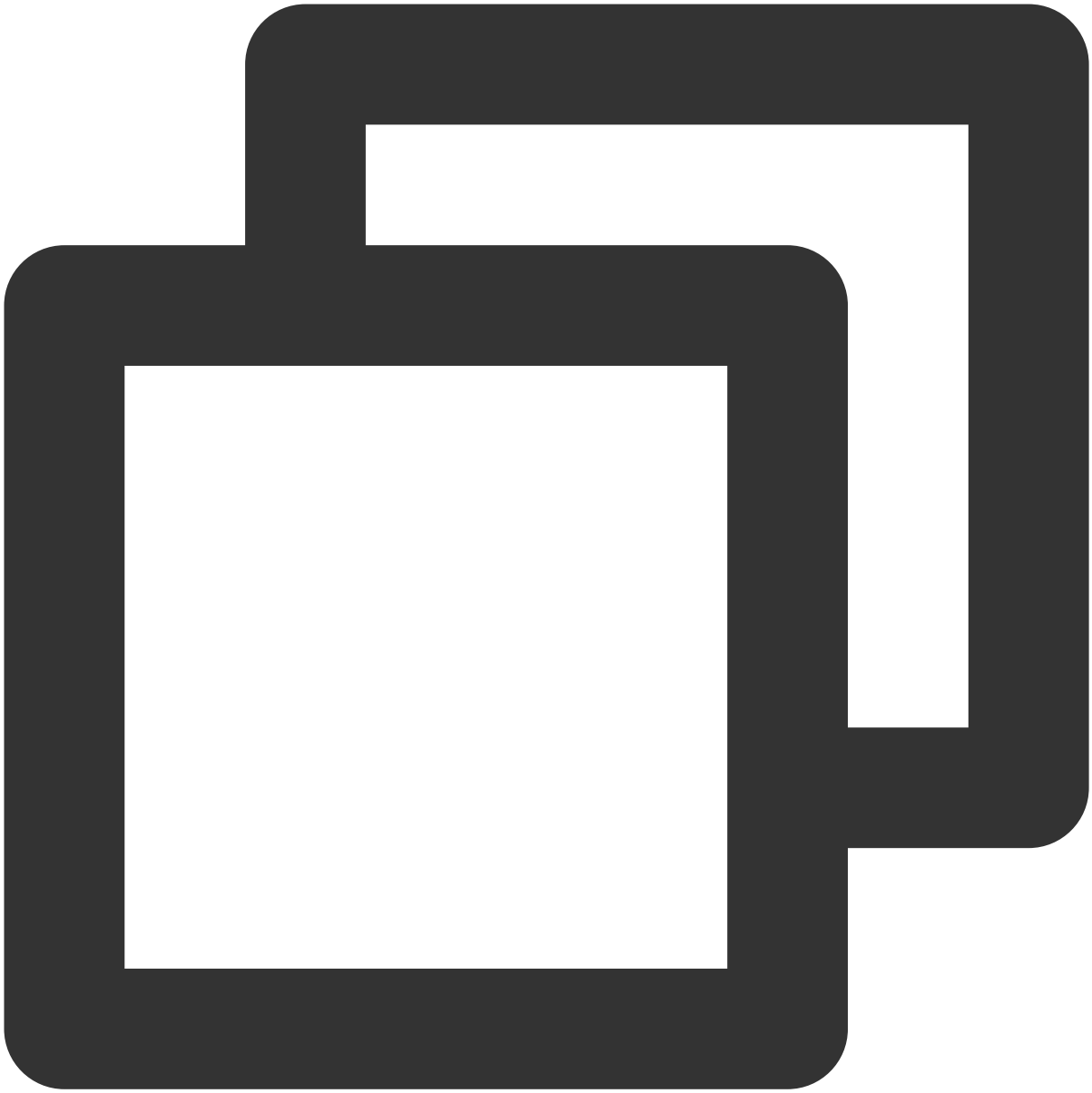
Java

Swift



```
public class MainActivity extends FlutterActivity {
    private static final String channelName = "TUICallKitCustomBeauty";

    private MethodChannel channel;

    @Override
    public void configureFlutterEngine(@NonNull FlutterEngine flutterEngine) {
        super.configureFlutterEngine(flutterEngine);

        channel = new MethodChannel(flutterEngine.getDartExecutor().getBinaryMessen
        channel.setMethodCallHandler(((call, result) -> {
```

```
            switch (call.method) {
                case "enableTUICallKitCustomBeauty":
                    enableTUICallKitCustomBeauty();
                    break;
                case "disableTUICallKitCustomBeauty":
                    disableTUICallKitCustomBeauty();
                    break;
                default:
                    break;
            }
            result.success("");
        }));
    }

    public void enableTUICallKitCustomBeauty() {

    }

    public void disableTUICallKitCustomBeauty() {

    }
}J
```



```
@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
    var channel: FlutterMethodChannel?

    override func application(_ application: UIApplication,
                              didFinishLaunchingWithOptions launchOptions: [UIAppli
        GeneratedPluginRegistrant.register(with: self)

        guard let controller = window?.rootViewController as? FlutterViewController
            fatalError("Invalid root view controller")
        }
```

```
        channel = FlutterMethodChannel(name: "TUICallKitCustomBeauty", binaryMessen
        channel?.setMethodCallHandler({ [weak self] call, result in
            guard let self = self else { return }
            switch (call.method) {
            case "enableTUICallKitCustomBeauty":
                self.enableTUICallKitCustomBeauty()
                break
            case "disableTUICallKitCustomBeauty":
                self.disableTUICallKitCustomBeauty()
                break
            default:
                break
            }
        })
        result(nil)
        return super.application(application, didFinishLaunchingWithOptions: launch
    }

    func enableTUICallKitCustomBeauty() {

    }

    func disableTUICallKitCustomBeauty() {

    }
}S
```

步骤2：在 Native 的 TRTC 自定义渲染逻辑中完成美颜处理

注意：

Android 在接入美颜过程中需要先依赖 **LiteAVSDK_Professional，** **在 Android 工程的 app/build.gradle 中添加一下依赖：**

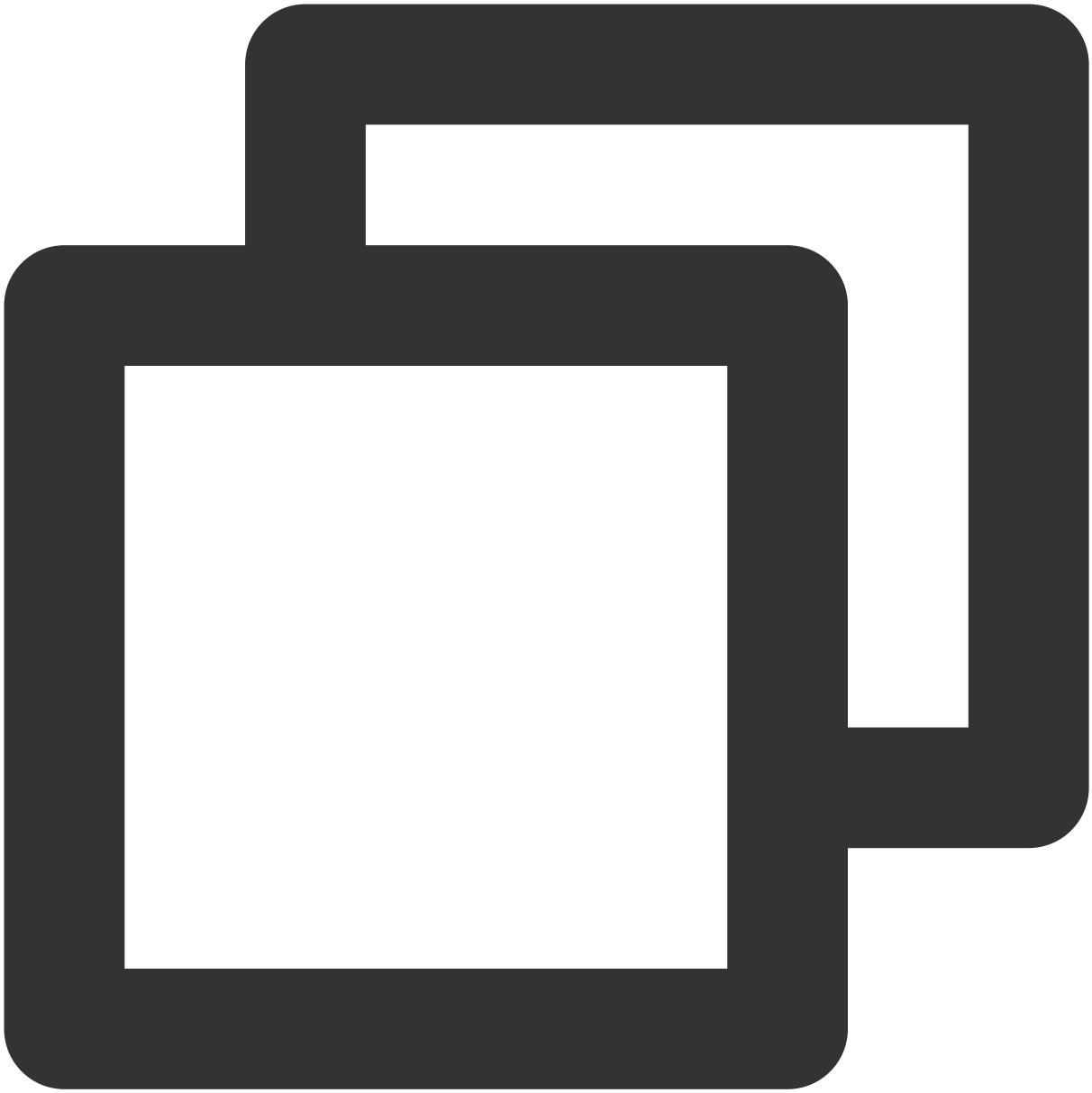
```
dependencies{
    api "com.tencent.liteav:LiteAVSDK_Professional:latest.release"
}
```

Java

Swift

```
Jvoid enableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance(getApplicationContext()).
                setLocalVideoProcessListener(TRTC_VIDEO_PIXEL_FORMAT_Texture_2D,
                        TRTC_VIDEO_BUFFER_TYPE_TEXTURE, new VideoFrameListerer());

}

void disableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance(getApplicationContext()).
                setLocalVideoProcessListener(TRTC_VIDEO_PIXEL_FORMAT_Texture_2D,
                        TRTC_VIDEO_BUFFER_TYPE_TEXTURE, null);
```

```
}

class VideoFrameListerer implements TRTCCloudListener.TRTCVideoFrameListener {
    private XXXBeautyModel  mBeautyModel = XXXBeautyModel.sharedInstance();

    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame trtcVideoFrame,
                                   TRTCCloudDef.TRTCVideoFrame trtcVideoFrame1) {
        // 美颜处理逻辑
        mBeautyModel.process(trtcVideoFrame, trtcVideoFrame1);
        ……

        return 0;
    }

    @Override
    public void onGLContextCreated() {
    }

    @Override
    public void onGLContextDestory() {
    }
}
```



```
let videoFrameListener: TRTCVideoFrameListener = TRTCVideoFrameListener()

func enableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance().setLocalVideoProcessDelegete(videoFrameListener, pix
}

func disableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance().setLocalVideoProcessDelegete(nil, pixelFormat: ._Tex
}

class TRTCVideoFrameListener: NSObject, TRTCVideoFrameDelegate {
```

```
    let bueutyModel = XXXXBeautyModel.shareIntance()
    func onProcessVideoFrame(_ srcFrame: TRTCVideoFrame, dstFrame: TRTCVideoFrame)
        // 美颜处理逻辑
        bueutyModel.onProcessVideoFrame(srcFrame, dstFrame)
        ……

        return 0
    }
}S
```

步骤3：客户自定义第三方美颜参数控制逻辑

此部分客户可以根据自己的需求使用具体的美颜模块，参见 步骤1 实现美颜参数的设置功能，具体实现根据具体使用情况决定。

接入腾讯美颜特效

腾讯美颜特效的接入方法也同样遵循上述方法，现在以腾讯美颜特效为例，详细介绍接入方法：

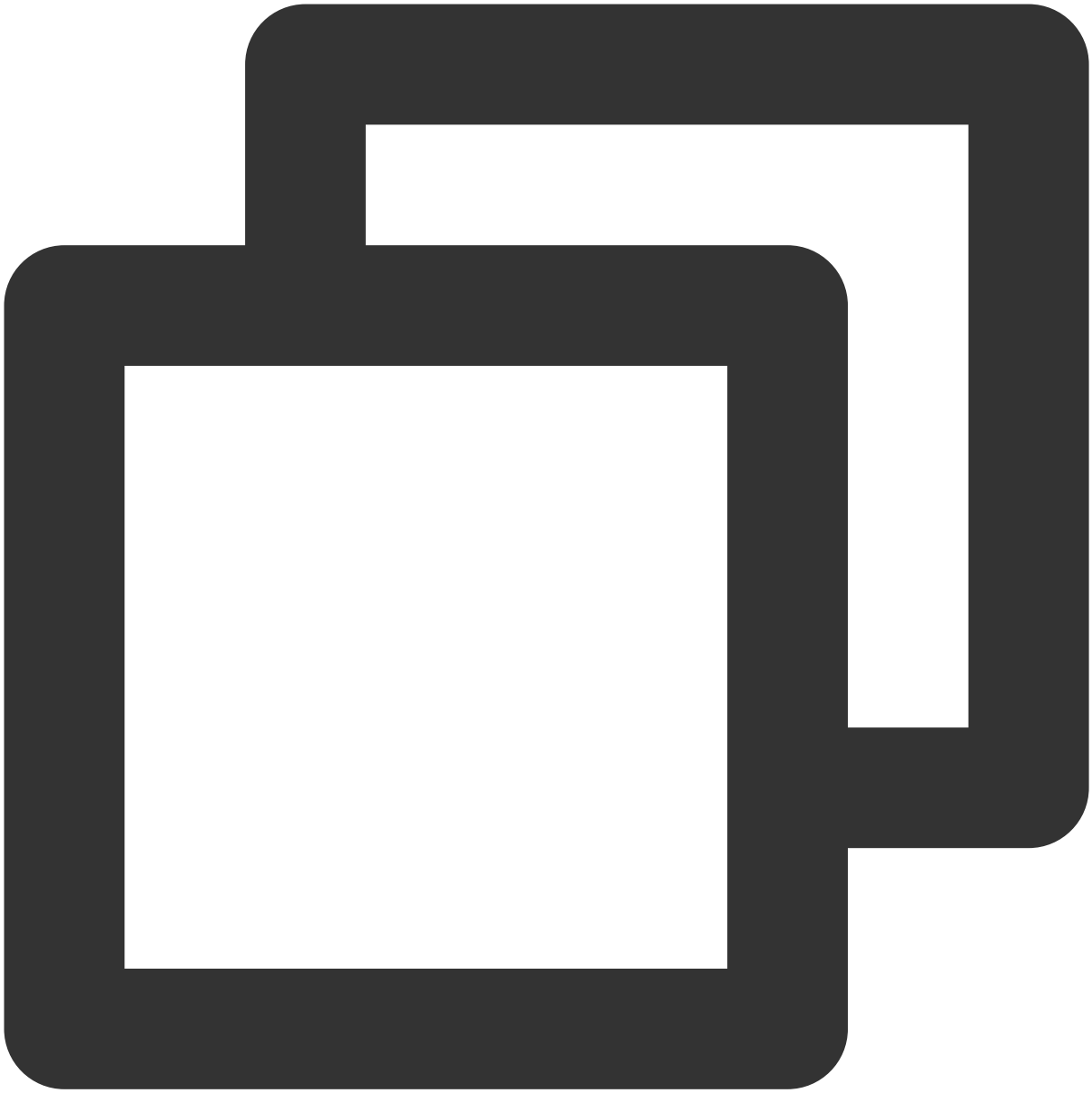
步骤1：美颜资源下载与集成

1. 根据您购买的套餐下载 SDK。
2. 添加文件到自己的工程中：

Android

iOS

1. 在 app 模块下找到 build.gradle 文件，添加您对应套餐的 maven 引用地址，例如您选择的是 S1-04 套餐，则添加如下代码：

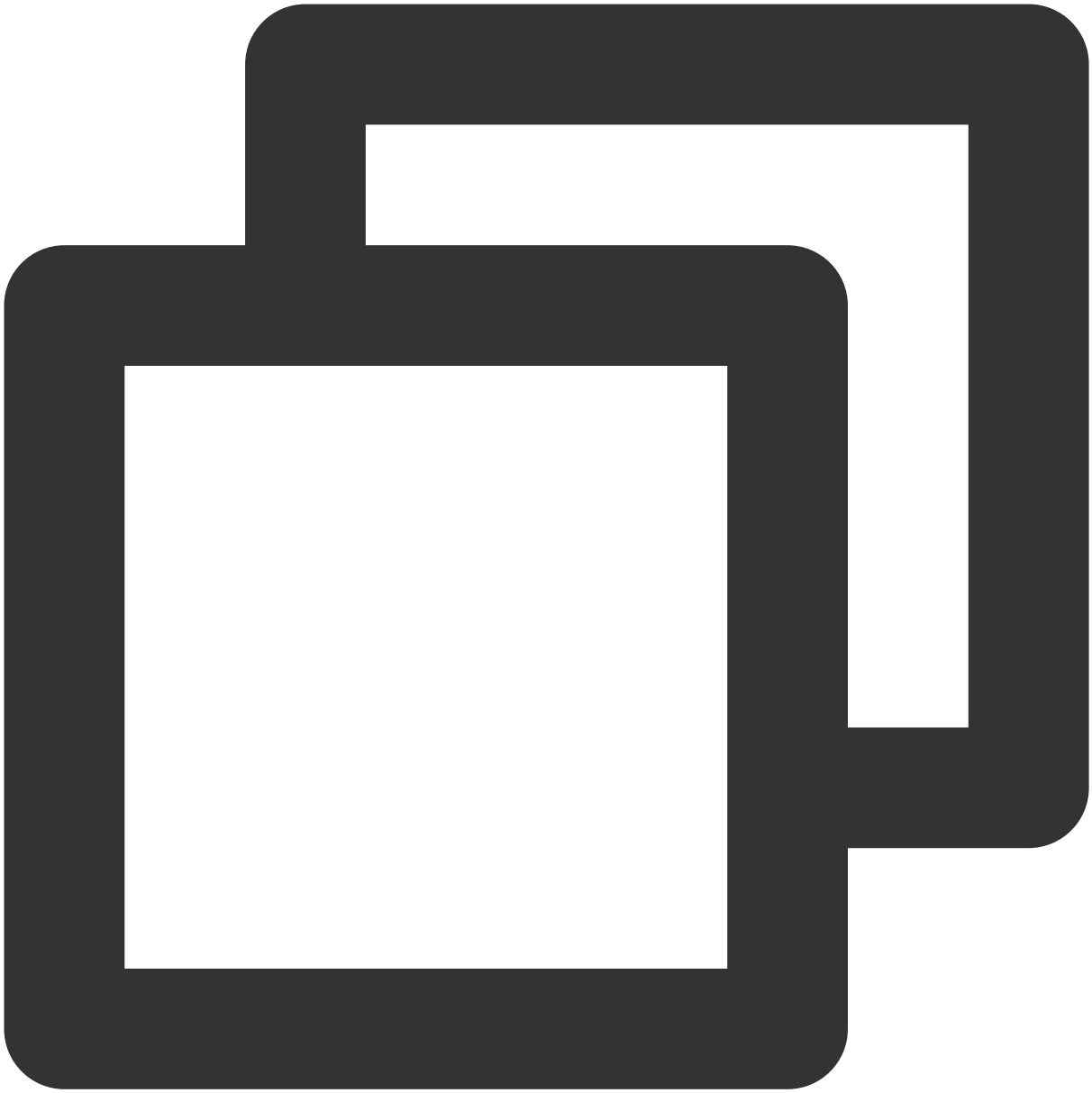


```
dependencies {
   implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
}
```

各套餐对应的 maven 地址，请参见文档。

2. 在 app 模块下找到 src/main/assets 文件夹，如果没有则创建，检查下载的 SDK 包中是否有 MotionRes 文件夹，如果有则将此文件夹拷贝到 `../src/main/assets` 目录下。

3. 在 app 模块下找到 AndroidManifest.xml 文件，在 application 表填内添加如下标签。



```
<uses-native-library
       android:name="libOpenCL.so"
       android:required="true" />
       //此处的true 表示如果没有此库，则应用将无法正常运行。系统不允许在没有此库的设备上安装应用
       //false表示应用可以使用此库（如果存在），但专门在没有此库的情况下运行（如果有必要）。系统
       //Android 官网介绍：https://developer.android.com/guide/topics/manifest/uses-
```

添加后如下图：

```
<application
    android:name="${applicationName}"
    android:icon="@mipmap/ic_launcher"
    android:label="tencent_effect_flutter_example"
    tools:replace="android:label">
    <uses-native-library
        android:name="libOpenCL.so"
        android:required="true" />
    <activity
        android:name=".MainActivity"
        android:configChanges="orientation|keyboardHidden|keybo
        android:exported="true"
        android:hardwareAccelerated="true"
        android:launchMode="singleTop"
        android:theme="@style/LaunchTheme"
        android:windowSoftInputMode="adjustResize">
        <!-- Specifies an Android theme to apply to this Activi
```

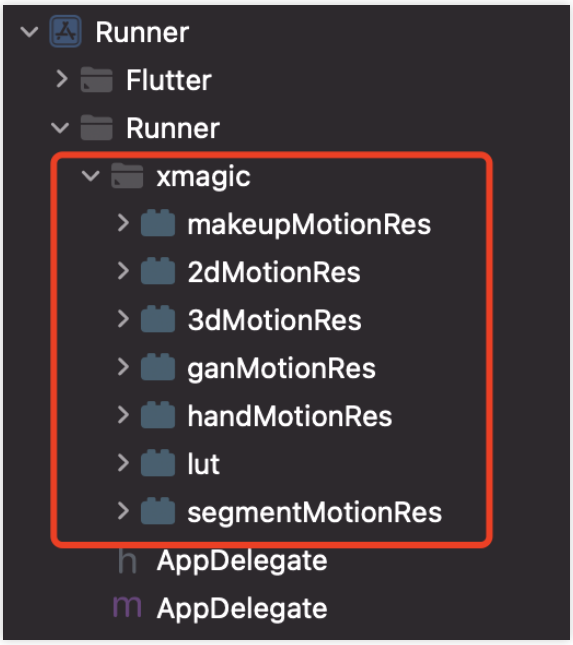
4. 混淆配置

如果您在打 release 包时，启用了编译优化（把 minifyEnabled 设置为 true ），会裁掉一些未在 Java 层调用的代码，而这些代码有可能会被 Native 层调用，从而引起 `no xxx method` 的异常。

如果您启用了这样的编译优化，那就要添加这些 keep 规则，防止 xmagic 的代码被裁掉：

```
-keep class com.tencent.xmagic.** { *;}
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class com.tencent.cloud.iai.lib.** { *;}
-keep class com.tencent.beacon.** { *;}
-keep class com.tencent.qimei.** { *;}
-keep class androidx.exifinterface.** { *;}
```

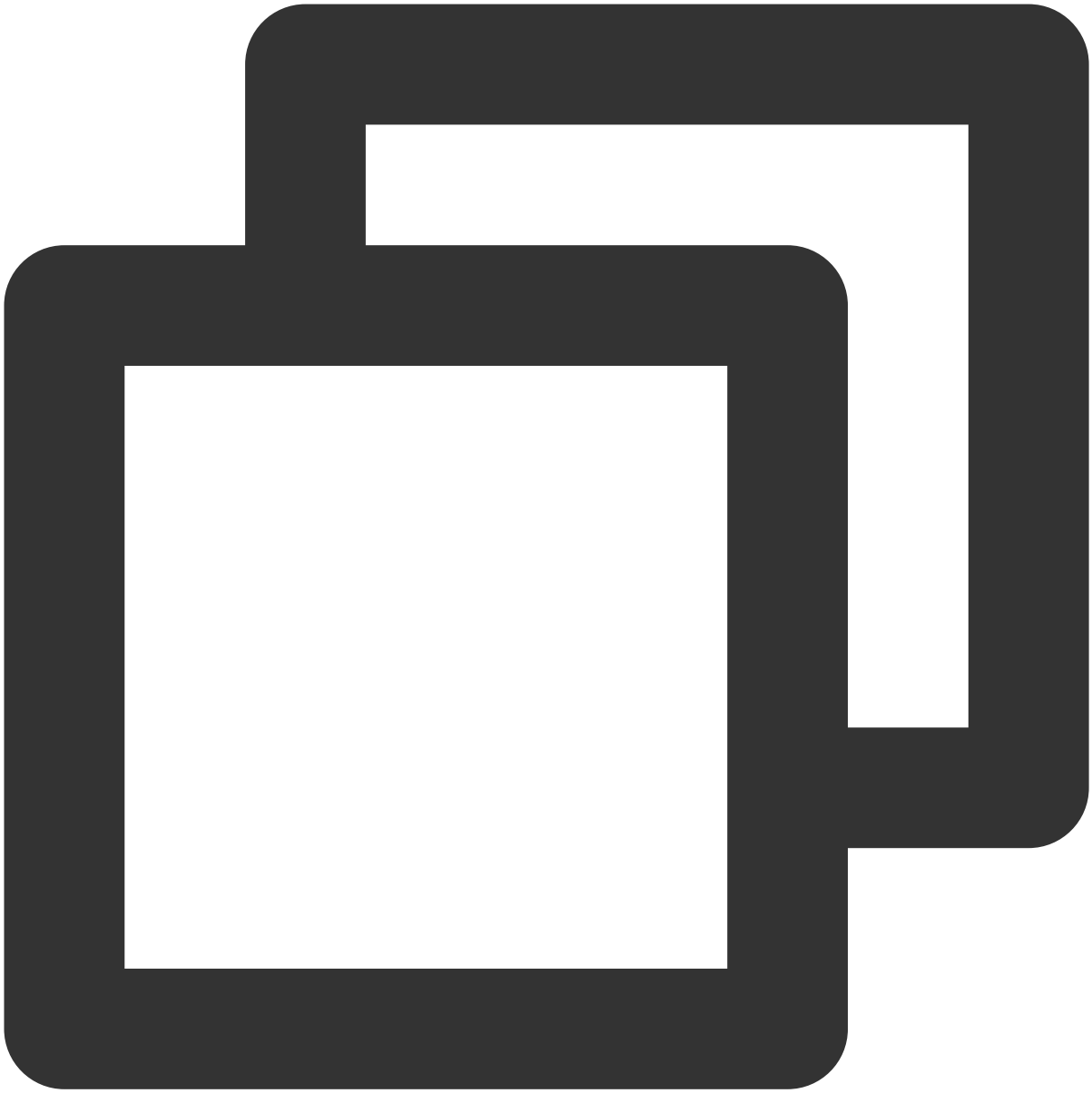
1. 添加美颜资源到您的工程。添加后如下图（您的资源种类跟下图不完全一致）：



2. 在 demo 中把 demo/lib/producer 里面的4个类：BeautyDataManager、BeautyPropertyProducer、BeautyPropertyProducerAndroid 和 BeautyPropertyProducerIOS 复制添加到自己的 Flutter 工程中，这 4 个类是用来配置美颜资源，把美颜类型展示在美颜面板中。

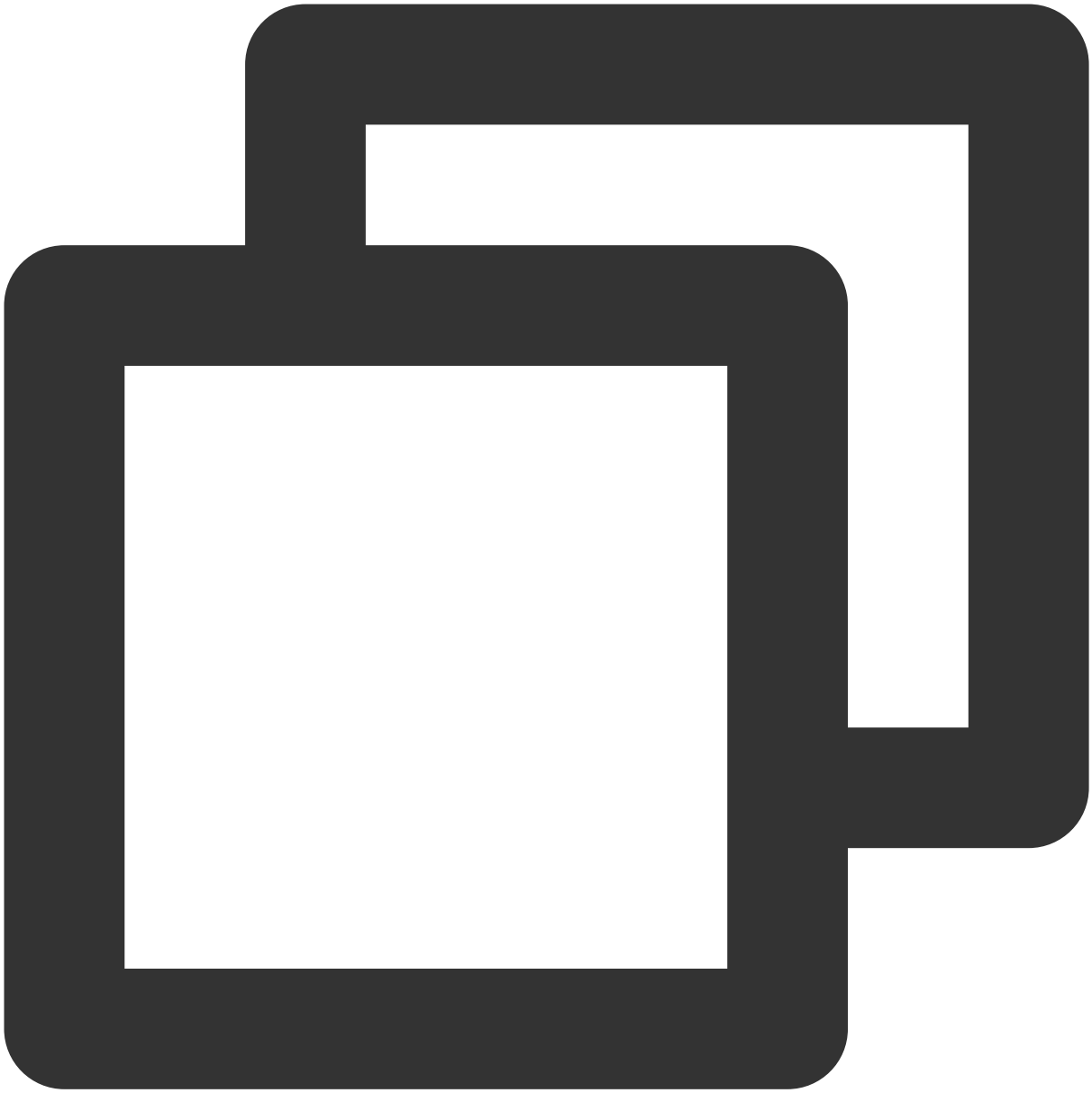
步骤2：引用 Flutter 版本 SDK

GitHub 引用：在工程的 pubspec.yaml 文件中添加如下引用：



```
tencent_effect_flutter:
  git:
    url: https://github.com/TencentCloud/tencenteffect-sdk-flutter
```

本地引用：从 tencent_effect_flutter 下载最新版本的 tencent_effect_flutter，然后把文件夹 `android` 、 `ios` 、 `lib` 和文件 `pubspec.yaml` 、 `tencent_effect_flutter.iml` 添加到工程目录下，然后在工程的 pubspec.yaml 文件中添加如下引用：（可参考demo ）



```
tencent_effect_flutter:
    path: ../
```

tencent_effect_flutter 只是提供一个桥接，里面依赖的 XMagic 默认是最新版的，真正实现美颜是 XMagic。

如果要使用最新版本的美颜 SDK，您可以通过以下步骤进行 SDK 升级：

Android

iOS

在工程目录下执行命令：`flutter pub upgrade` 或者在 `subspec.yaml` 页面的右上角单击 **Pub upgrade**。

在工程目录下执行命令：flutter pub upgrade，然后在 iOS 目录下执行命令： `pod update` 。

步骤3：实现 Dart 层到 Native 开始/关闭美颜的控制接口

此部分可以参见 接入第三方美颜特效-步骤1。

步骤4：在 Native 的 TRTC 自定义渲染逻辑中完成美颜处理

Android

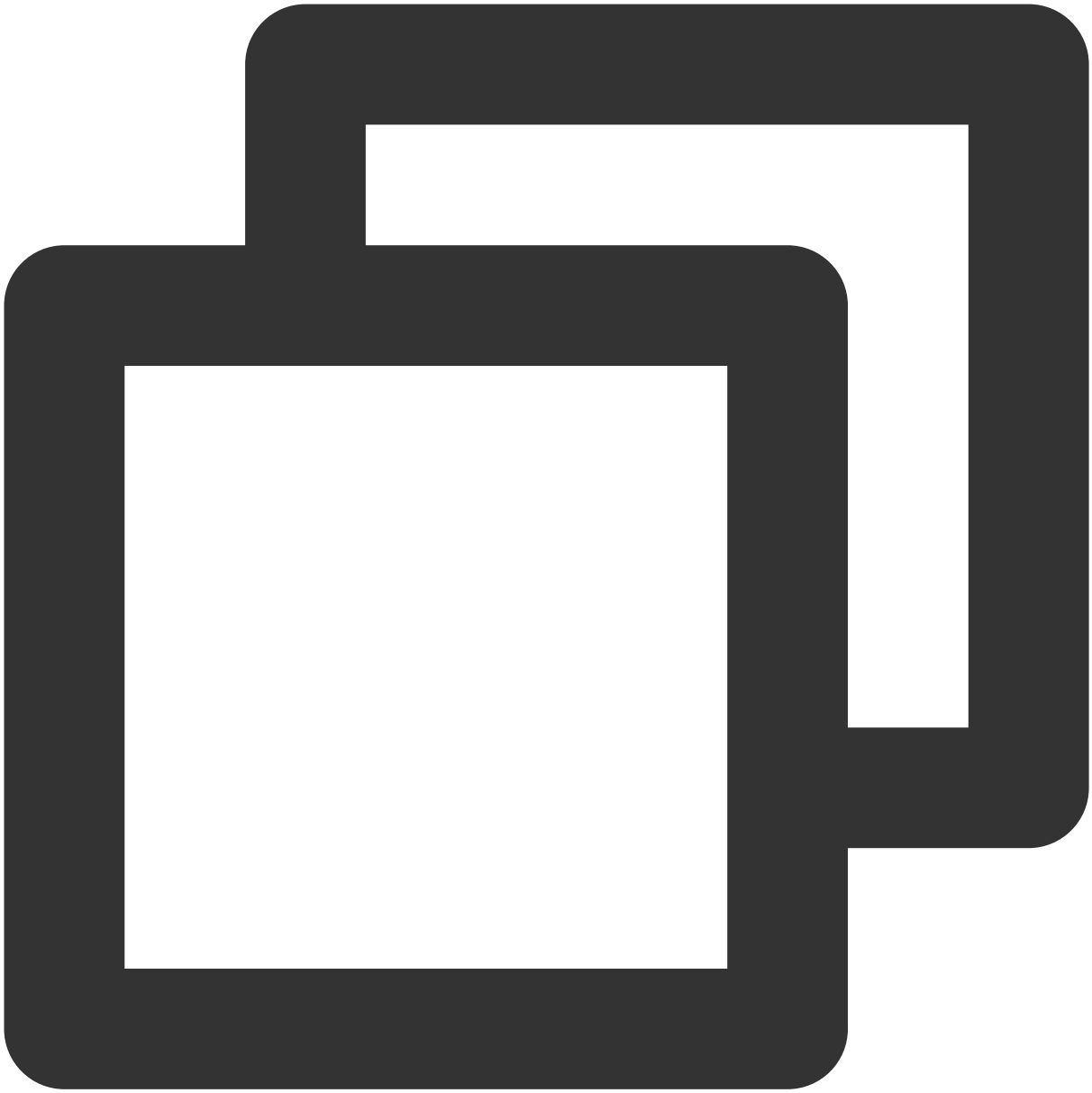
iOS

Android 在接入美颜过程中需要先依赖 **LiteAVSDK_Professional，** **在 Android 工程的 app/build.gradle 中添加以下依赖：**

```
dependencies{
    api "com.tencent.liteav:LiteAVSDK_Professional:latest.release"
}
```

```
void enableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance(getApplicationContext()).
                setLocalVideoProcessListener(TRTC_VIDEO_PIXEL_FORMAT_Texture_2D,
                        TRTC_VIDEO_BUFFER_TYPE_TEXTURE, new VideoFrameListerer());

}

void disableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance(getApplicationContext()).
                setLocalVideoProcessListener(TRTC_VIDEO_PIXEL_FORMAT_Texture_2D,
                        TRTC_VIDEO_BUFFER_TYPE_TEXTURE, null);
```

```
}

class VideoFrameListerer implements TRTCCloudListener.TRTCVideoFrameListener {
    private XXXBeautyModel  mBeautyModel = XXXBeautyModel.sharedInstance();

    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame trtcVideoFrame,
                                   TRTCCloudDef.TRTCVideoFrame trtcVideoFrame1) {
       trtcVideoFrame1.texture.textureId = XmagicApiManager.getInstance()
        .process(trtcVideoFrame.texture.textureId, trtcVideoFrame.width, trtcVideoF
        return 0;
    }

    @Override
    public void onGLContextCreated() {
    }

    @Override
    public void onGLContextDestory() {
    }
}
```



```
import TXLiteAVSDK_Professional
import tencent_effect_flutter

let videoFrameListener: TRTCVideoFrameListener = TRTCVideoFrameListener()

func enableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance().setLocalVideoProcessDelegete(videoFrameListener, pix
}

func disableTUICallKitCustomBeauty() {
    TRTCCloud.sharedInstance().setLocalVideoProcessDelegete(nil, pixelFormat: ._Tex
```

```
}

class TRTCVideoFrameListener: NSObject, TRTCVideoFrameDelegate {
    func onProcessVideoFrame(_ srcFrame: TRTCVideoFrame, dstFrame: TRTCVideoFrame)
        dstFrame.textureId = GLuint(XmagicApiManager.shareSingleton().getTextureId(
        return 0
    }
}


public class ConvertBeautyFrame: NSObject {
    public static func convertToTRTCPixelFormat(beautyPixelFormat: ITXCustomBeautyP
        switch beautyPixelFormat {
        case .Unknown:
            return ._Unknown
        case .I420:
            return ._I420
        case .Texture2D:
            return ._Texture_2D
        case .BGRA:
            return ._32BGRA
        case .NV12:
            return ._NV12
        }
    }

    public static func convertTRTCVideoFrame(trtcVideoFrame: TRTCVideoFrame) -> ITX
        let beautyVideoFrame = ITXCustomBeautyVideoFrame()
        beautyVideoFrame.data = trtcVideoFrame.data
        beautyVideoFrame.pixelBuffer = trtcVideoFrame.pixelBuffer
        beautyVideoFrame.width = UInt(trtcVideoFrame.width)
        beautyVideoFrame.height = UInt(trtcVideoFrame.height)
        beautyVideoFrame.textureId = trtcVideoFrame.textureId
        switch trtcVideoFrame.rotation {
        case ._0:
            beautyVideoFrame.rotation = .rotation_0
        case ._90:
            beautyVideoFrame.rotation = .rotation_90
        case ._180:
            beautyVideoFrame.rotation = .rotation_180
        case ._270:
            beautyVideoFrame.rotation = .rotation_270
        default:
            beautyVideoFrame.rotation = .rotation_0
        }
        switch trtcVideoFrame.pixelFormat {
        case ._Unknown:
```

```
            beautyVideoFrame.pixelFormat = .Unknown
        case ._I420:
            beautyVideoFrame.pixelFormat = .I420
        case ._Texture_2D:
            beautyVideoFrame.pixelFormat = .Texture2D
        case ._32BGRA:
            beautyVideoFrame.pixelFormat = .BGRA
        case ._NV12:
            beautyVideoFrame.pixelFormat = .NV12
        default:
            beautyVideoFrame.pixelFormat = .Unknown
        }
        beautyVideoFrame.bufferType = ITXCustomBeautyBufferType(rawValue: trtcVideo
        beautyVideoFrame.timestamp = trtcVideoFrame.timestamp
        return beautyVideoFrame
    }
}
```

步骤5：开启美颜并设置美颜参数

完成上述配置后，可以通过 `enableTUICallKitCustomBeauty()/disableTUICallKitCustomBeauty()` 开启/关闭美颜。可以通过腾讯特效美颜 Flutter 接口设置美颜参数。

自定义铃声

Android

最近更新时间：2024-05-08 11:37:24

本文介绍如何替换 TUICallKit 的来电铃声，来电铃声分为**应用铃声**和**离线推送铃声**。

一、 设置应用铃声

设置应用铃声有两种方式：替换铃声音频、调用设置铃声接口。

1、替换音频文件

如果您通过源码依赖 TUICallKit 组件，您可以替换 `res\\raw` 文件夹下的三个音频文件来达到替换铃声的目的：

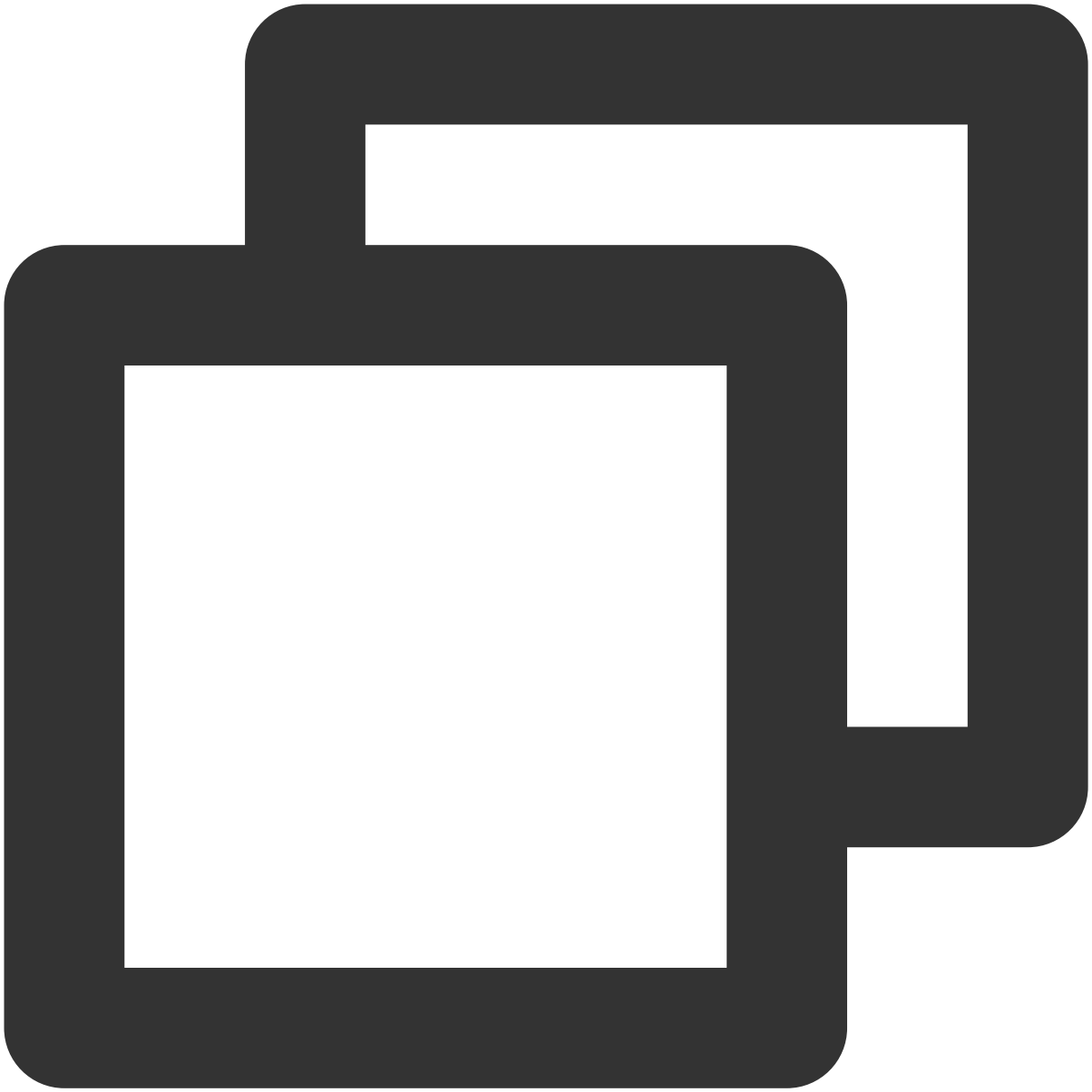
文件名	用途
phone_dialing.mp3	发起呼叫时的铃音
phone_hangup.mp3	被挂断的铃音
phone_ringing.mp3	接到呼叫时的铃音

2、设置铃声接口

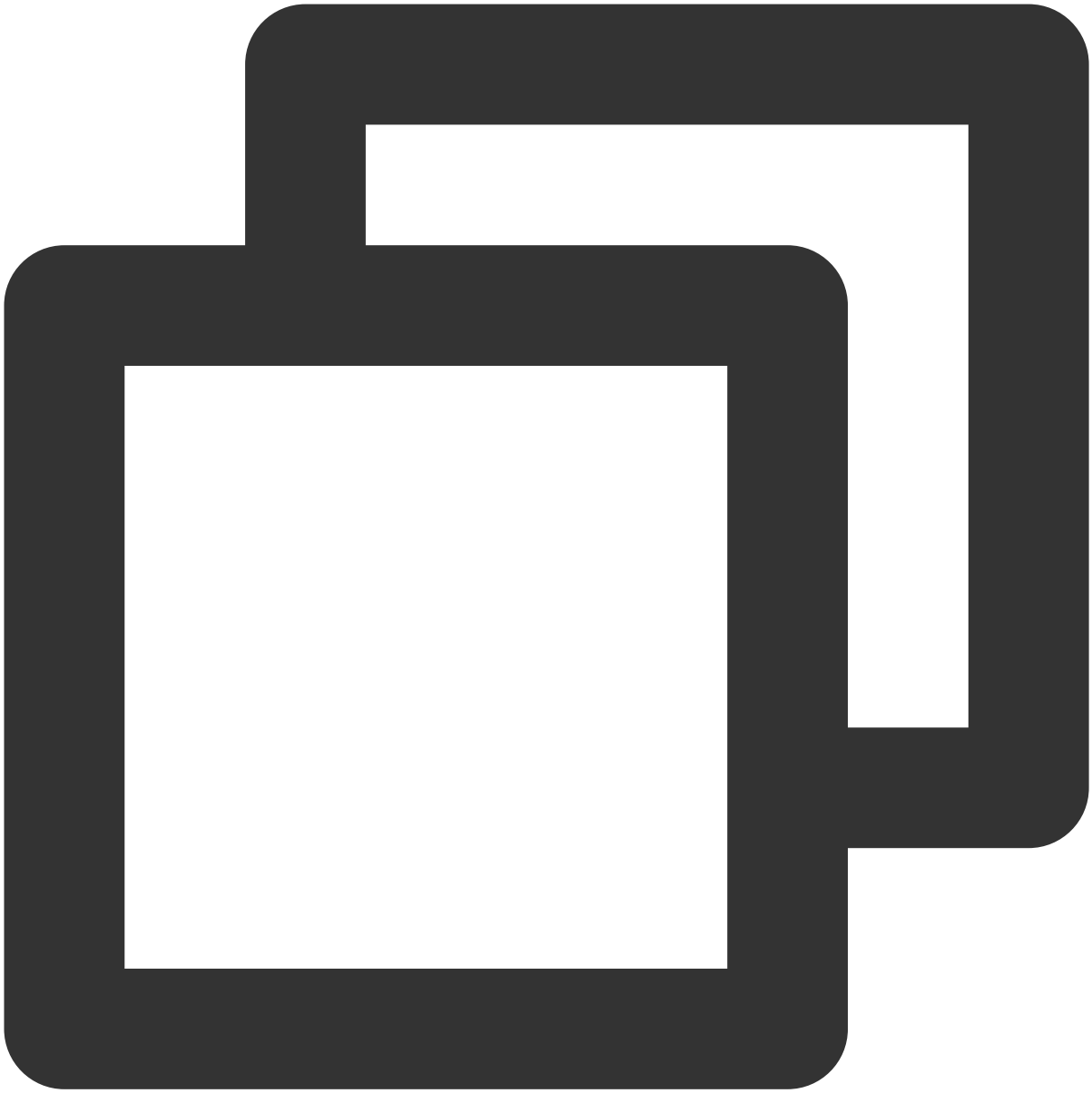
您也可以通过 setCallingBell 接口设置来电铃声。

Kotlin

Java



```
TUICallKit.createInstance(context).setCallingBell(filePath)
```



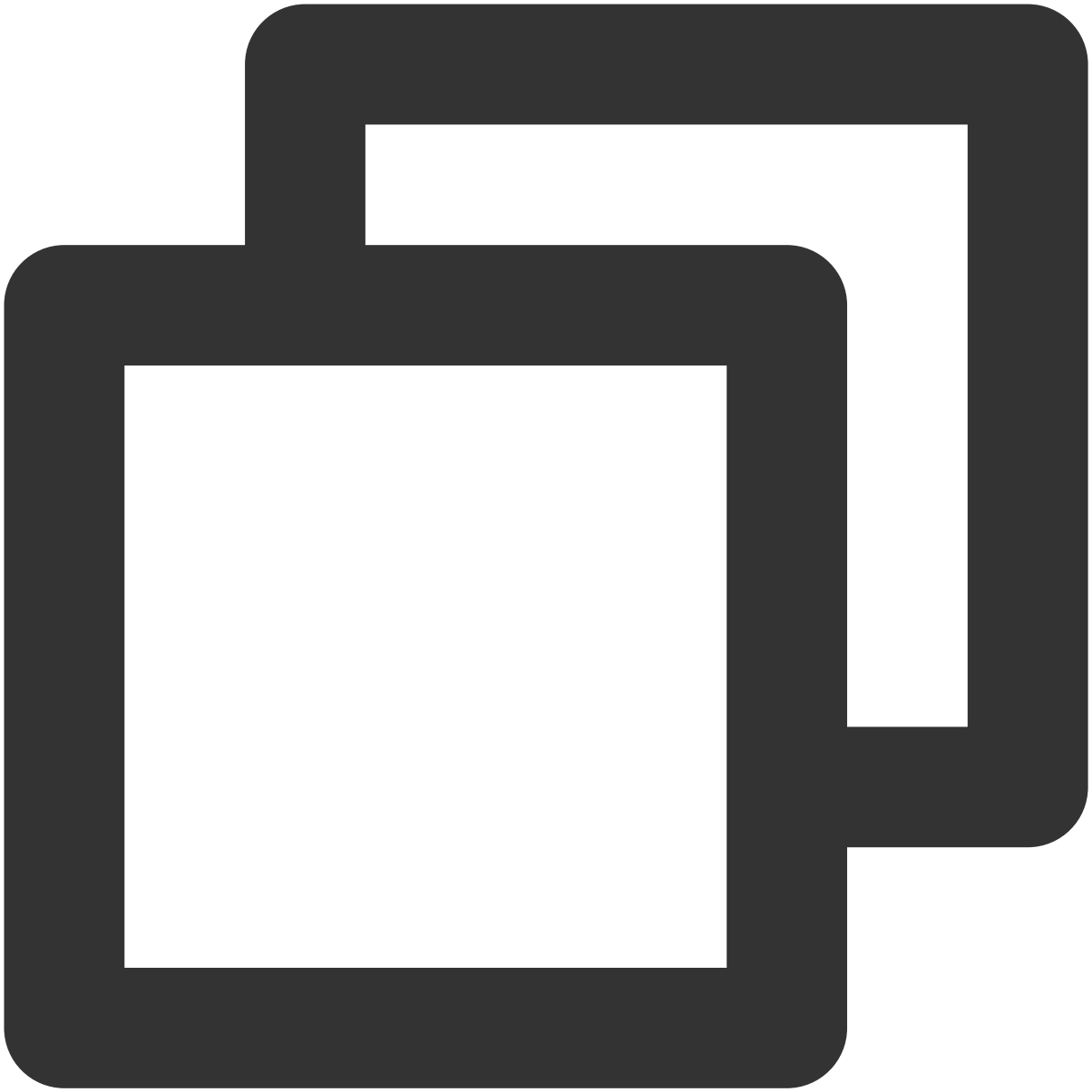
```
TUICallKit.createInstance(context).setCallingBell(filePath);
```

3. 设置静音模式

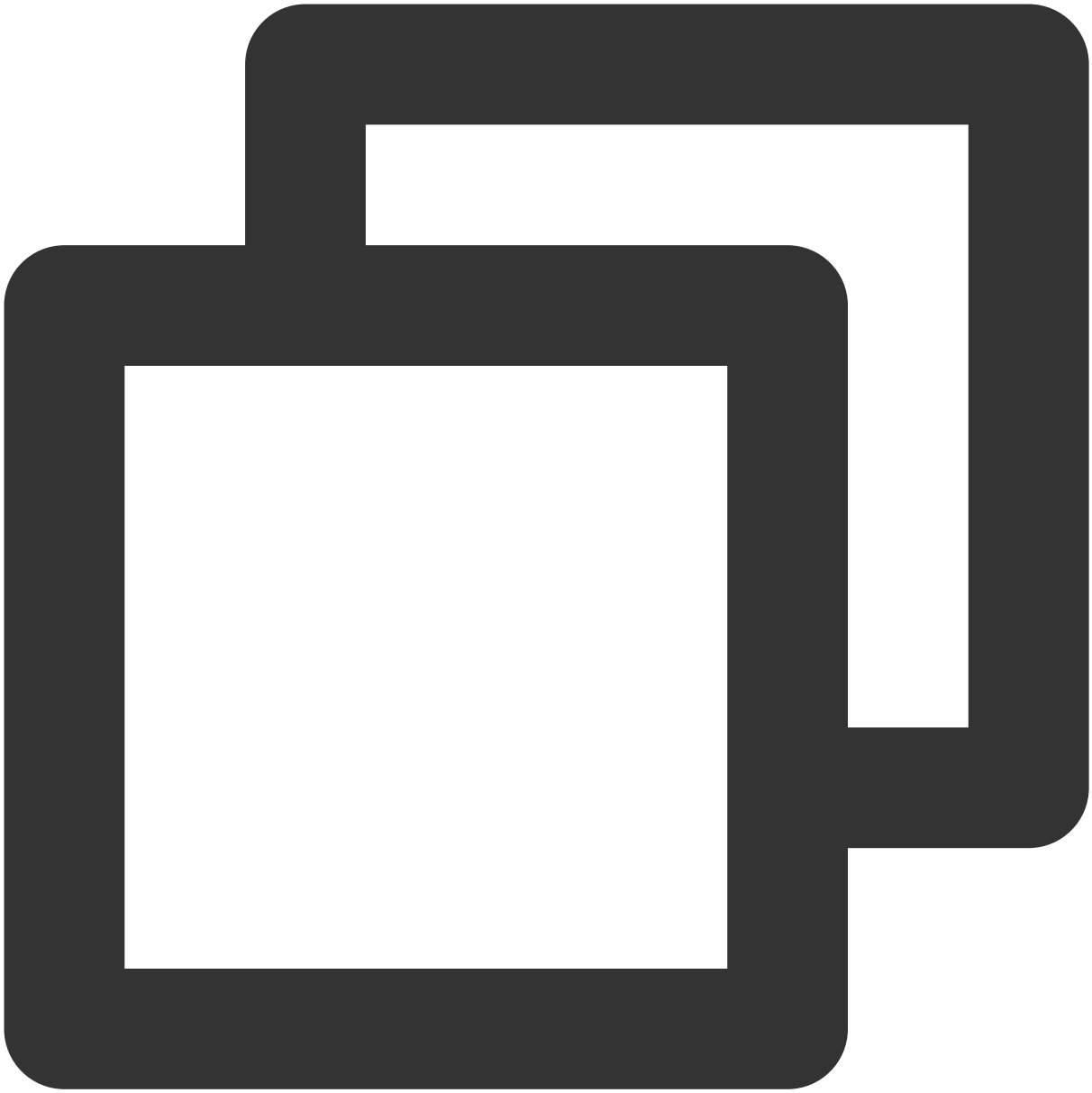
如果您不需要响铃，您可以通过 enableMuteMode 设置静音模式。

Kotlin

Java



```
TUICallKit.createInstance(context).enableMuteMode(true)
```



```
TUICallKit.createInstance(context).enableMuteMode(true);
```

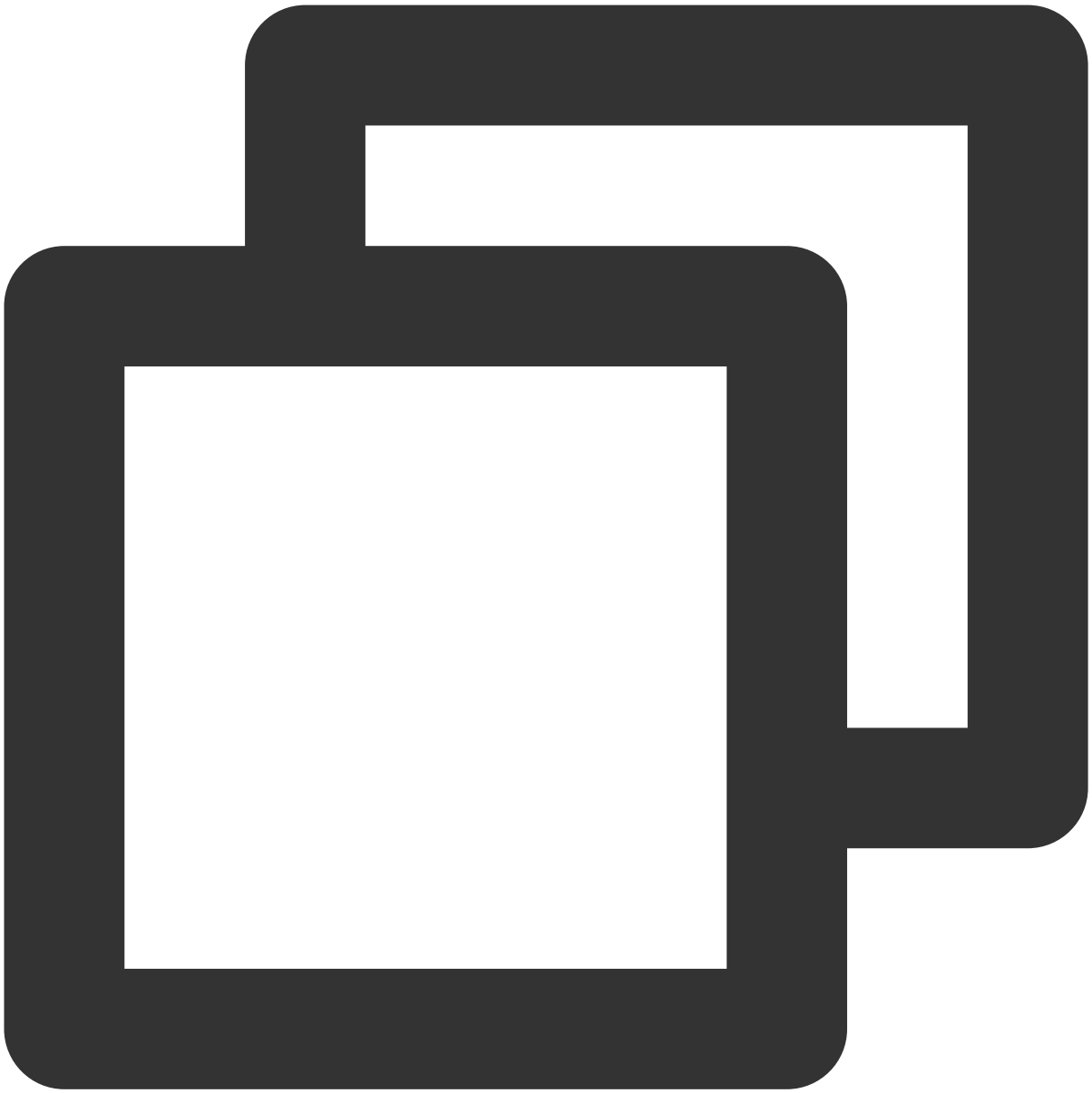
二、设置离线推送铃音

离线通知铃声仅支持以下厂商自定义：**华为、小米、FCM 和 APNs**，其他厂商 OPPO、VIVO、荣耀等暂不支持。

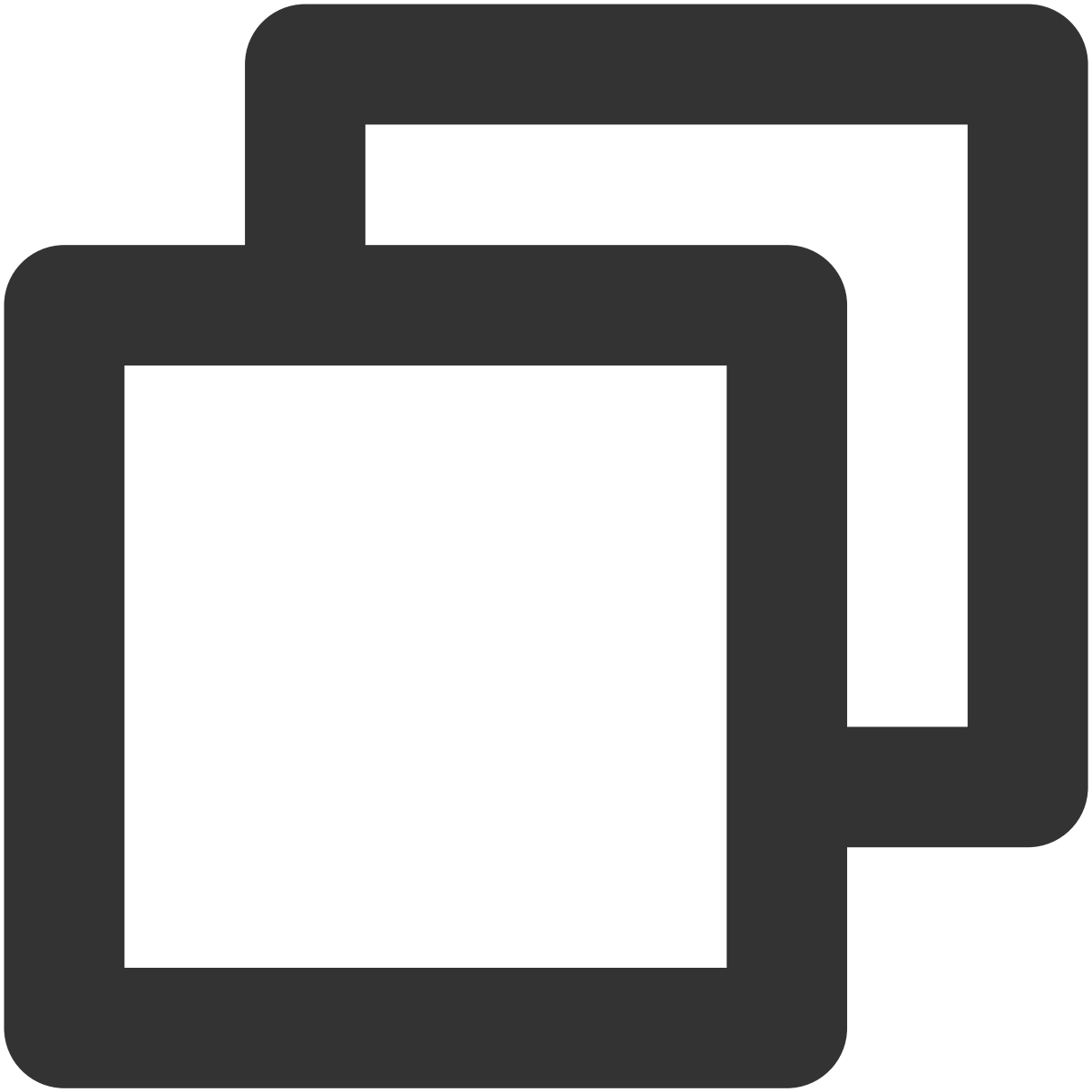
TUICallKit 引入了 TUIOfflinePushPush 组件，需在应用启动时，调用以下方法开启离线通知自定义铃音能力。

Kotlin

Java



```
class DemoApplication : Application() {
    override fun onCreate() {
        TUIOfflinePushConfig.getInstance().isAndroidPrivateRing = true
    }
}
```



```
public class DemoApplication extends Application {
    @Override
    public void onCreate() {
        TUIOfflinePushConfig.getInstance().setAndroidPrivateRing(true);
    }
}
```

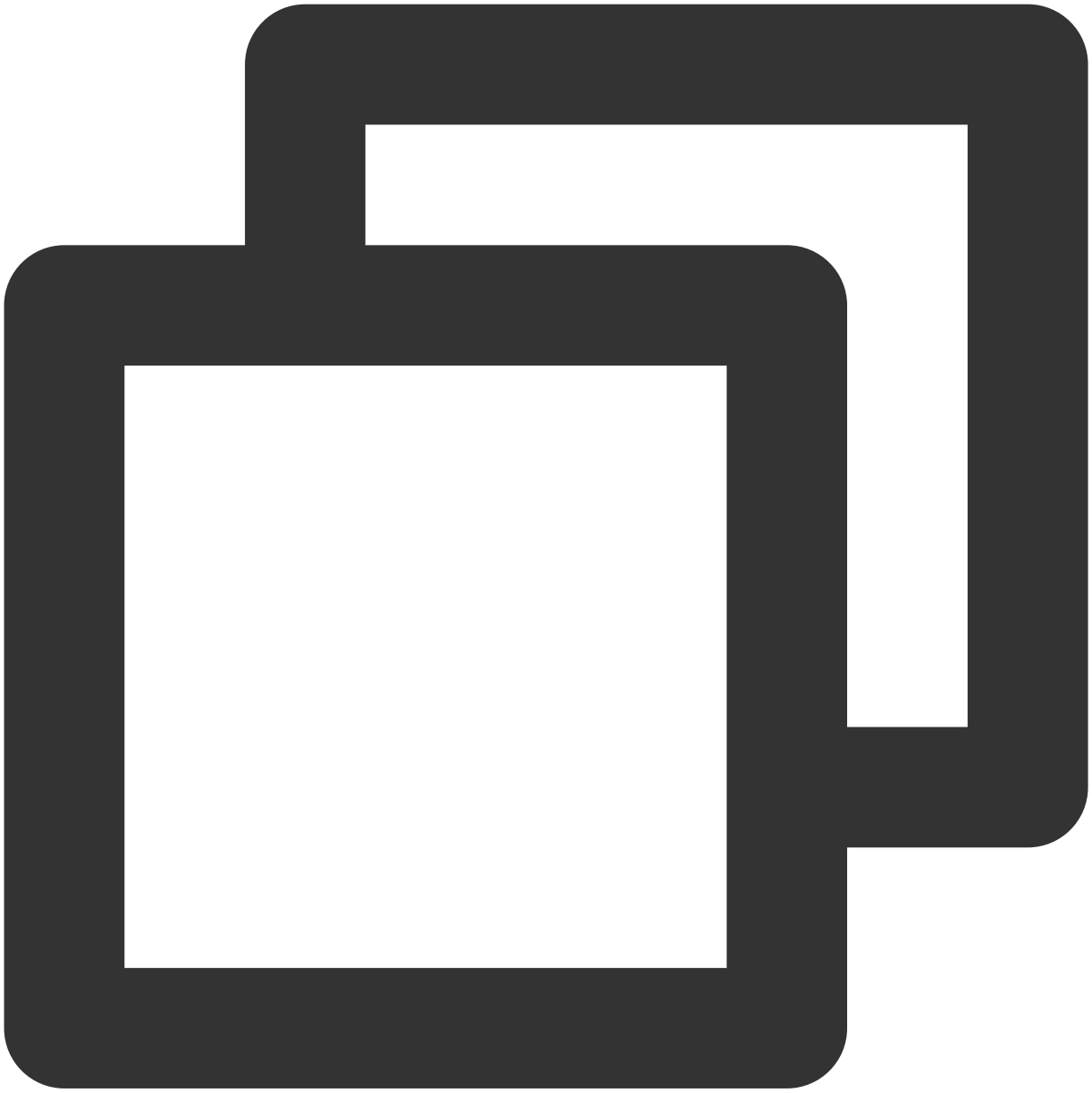
1、华为 & APNs

调用接口 setAndroidSound() 和 setIOSSound()。

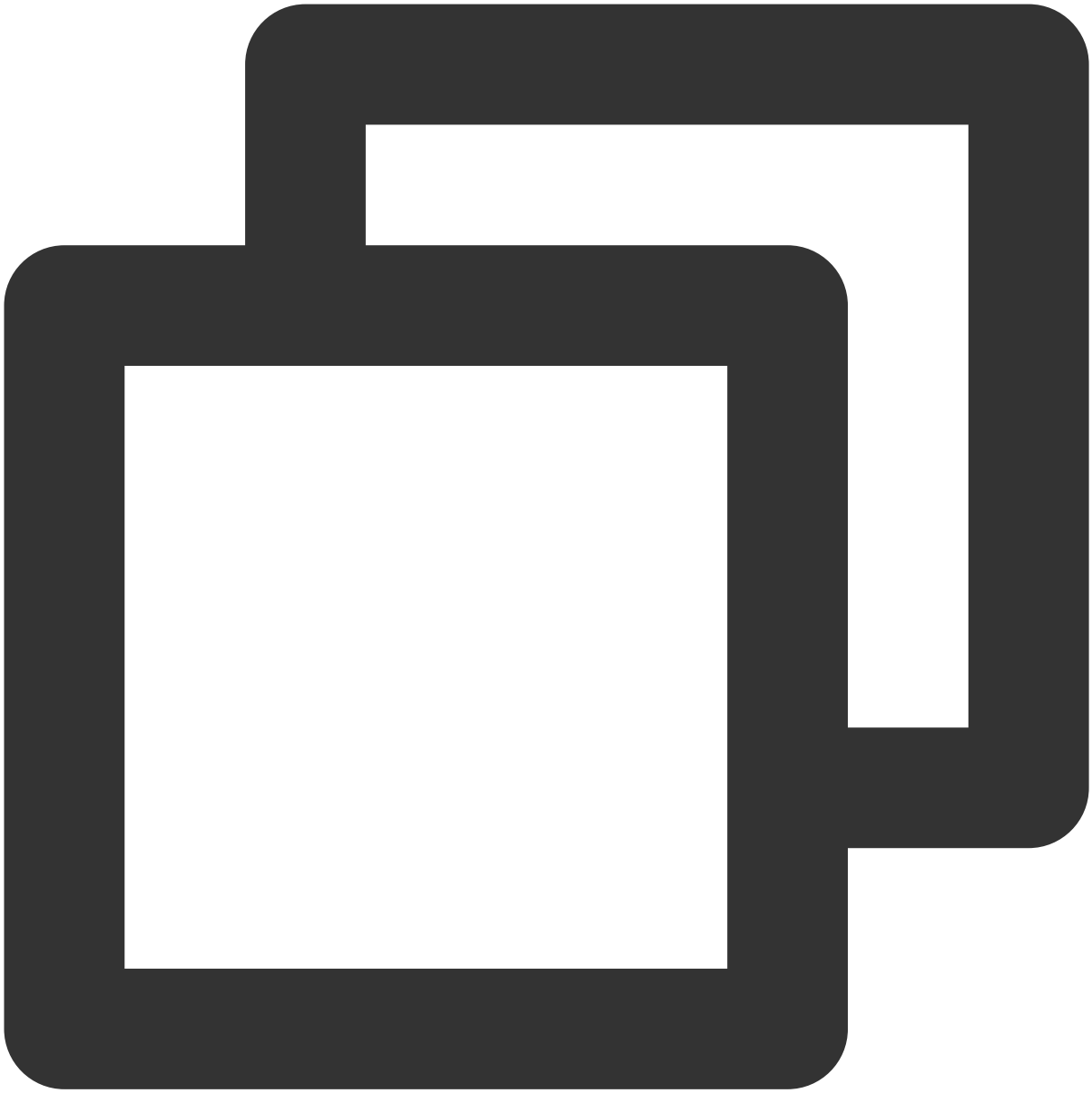
定制铃音资源文件，将铃音文件添加到本地 Android Studio 工程 res/raw 目录下。

Kotlin

Java



```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
```

注意

IMSDK 6.1.2155 及以上版本支持。

华为推送消息的铃声为首次创建通知渠道时的铃声，之后再次设置其他铃声无效，若想更新自定义通知铃音需卸载重装，详情可参见：华为自定义铃声。

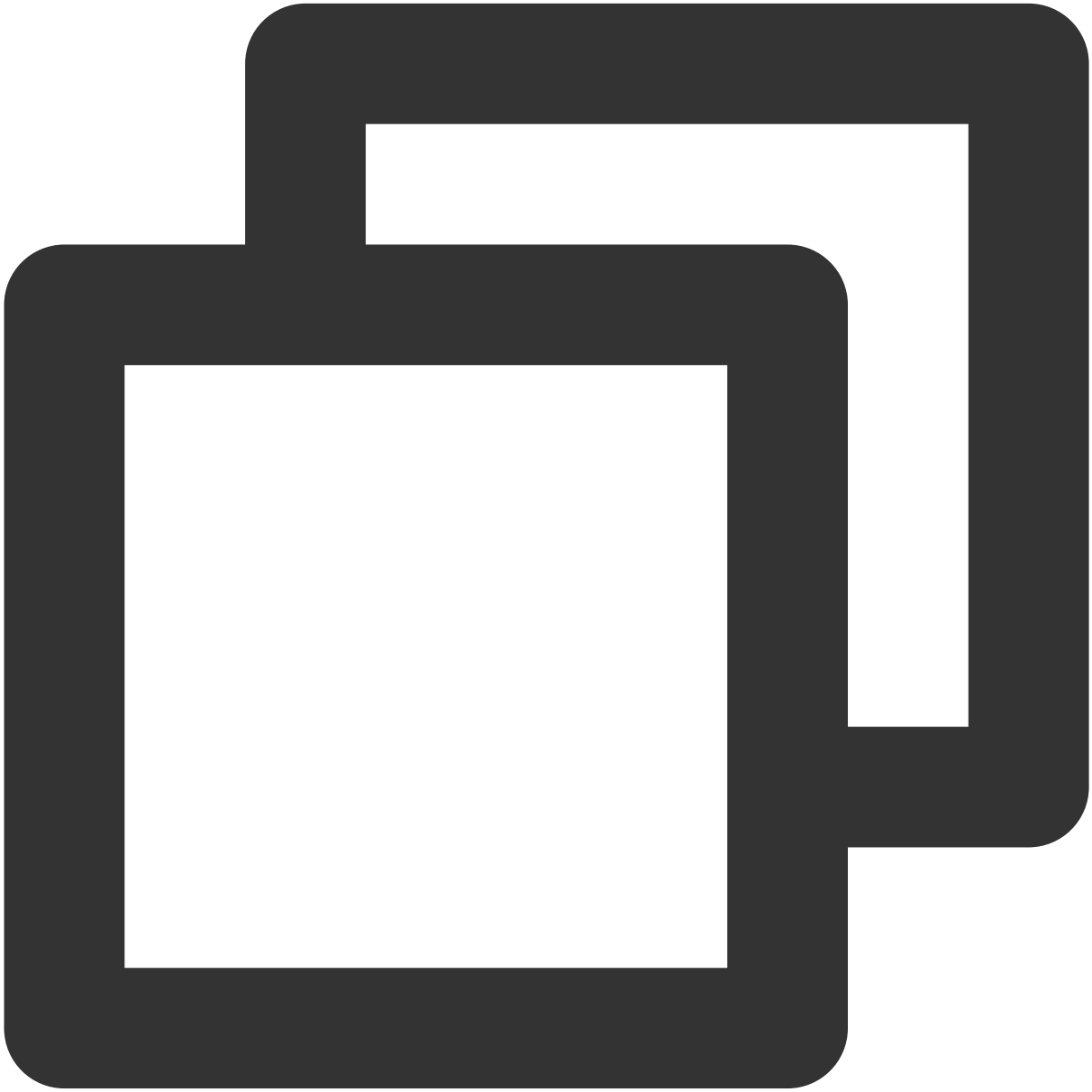
2、小米

（1）Android 8.0 之前，调用setAndroidSound() 和 setIOSSound()。定制铃音资源文件，将铃音文件添加到工程 res/raw 目录下（参考上述华为调用）。

（2）Android 8.0 之后，还需要登录小米厂商控制台并创建channel并做好配置，其中铃音文件需要添加到本地 Android Studio 工程的 res/raw 目录下。

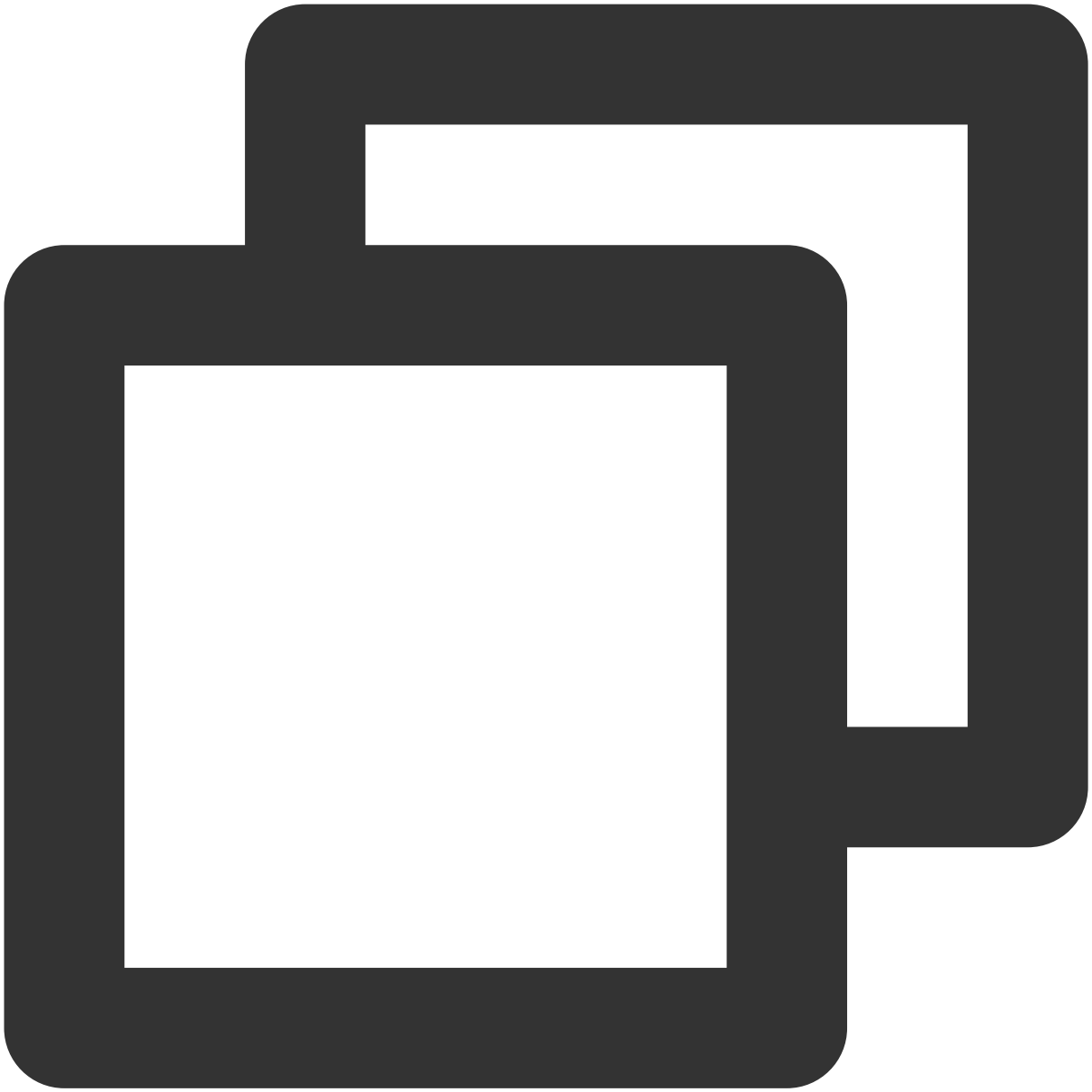
Kotlin

Java



```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
```

```
pushInfo.androidXiaoMiChannelID = "厂商申请的channelID"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
pushInfo.setAndroidXiaoMiChannelID("厂商申请的channelID");
```

注意

Android 8.0之前的方式，IMSDK 6.1.2155 及以上版本支持。

Android 8.0之后的方式，IMSDK 7.0.3754 及以上版本支持。

3、FCM

（1）Android 8.0 之前，调用 setAndroidSound() 和 setIOSSound()。定制铃音资源文件，将铃音文件添加到工程 res/raw 目录下（参考上述华为调用）。

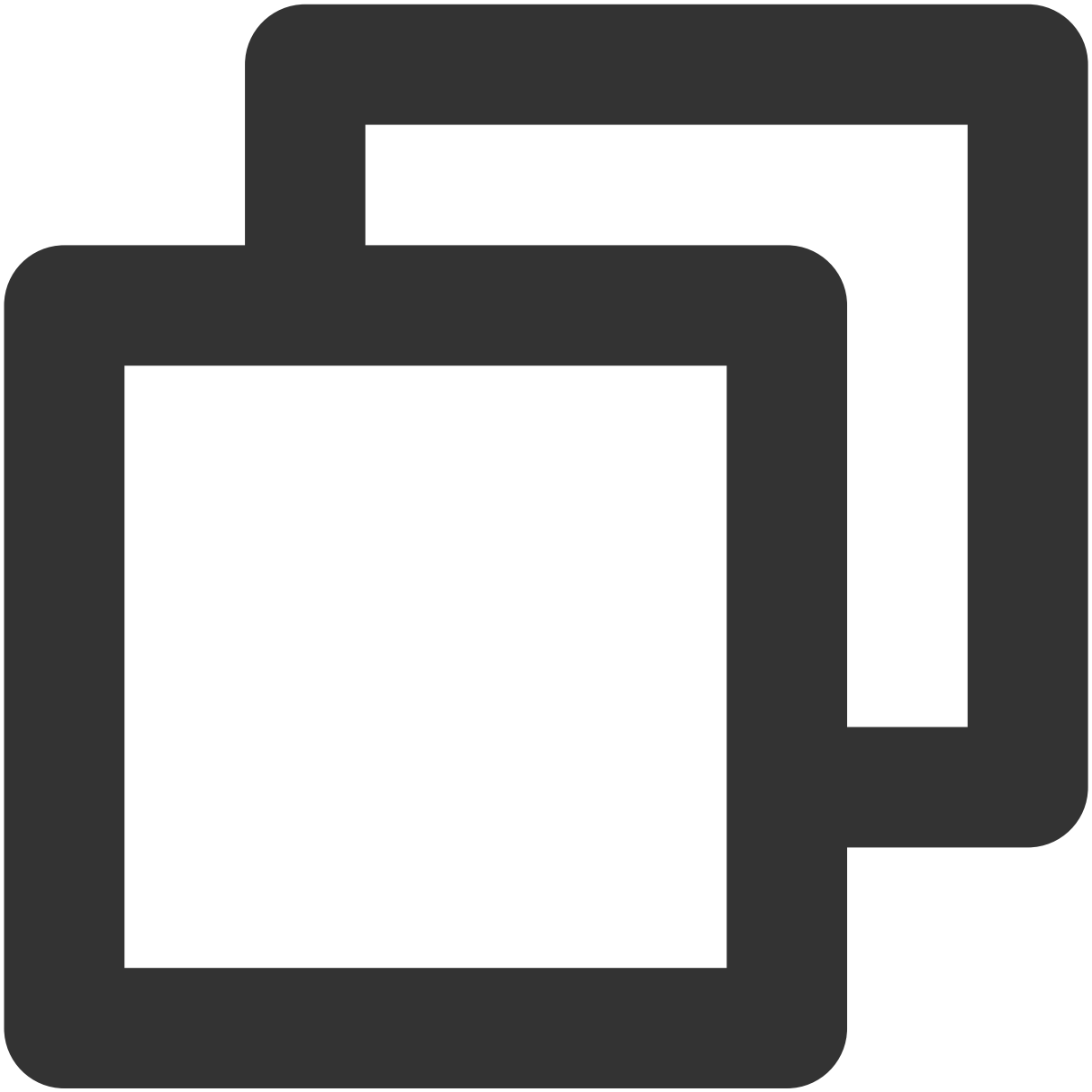
（2）Android 8.0之后，FCM 需要配置 channelID 以及铃音资源，TUICallKit 引入的 tuiofflinepush 组件已经处理了自定义铃声的播放。铃音文件需要添加到您本地 Android Studio 工程的 res/raw 目录下，并指定铃音名称和 channel ID 的名称，详见 PrivateConstants。

```
public class PrivateConstants {
    // FCM 通道指定 channel ID
    public static String fcmPushChannelId = "FCM ChannelID";
```

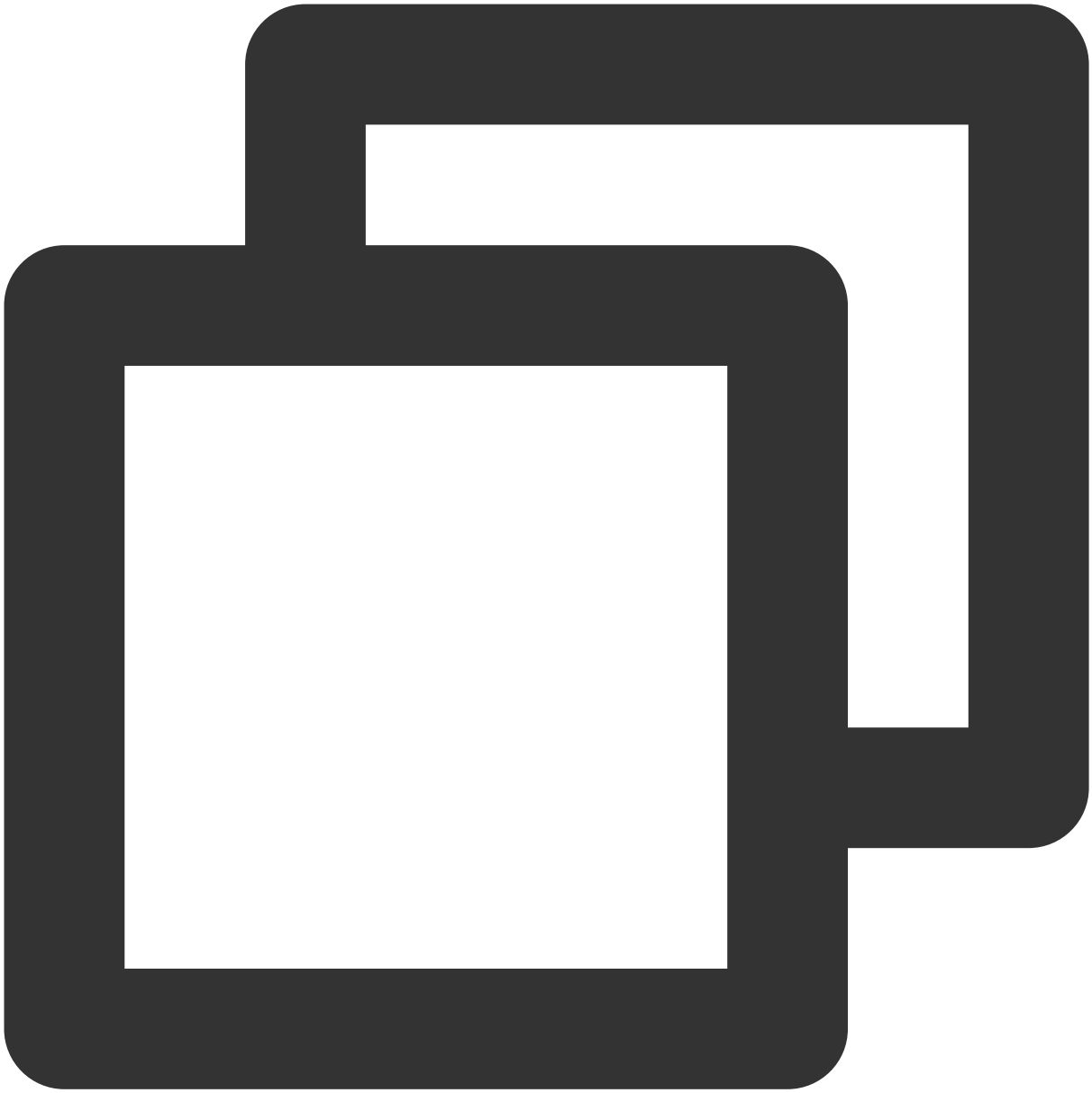
```
    // FCM 需指定通道的推送铃音名称，与AndroidStudio 工程里 res/raw 目录中的铃音文件名一致，不
    public static String fcmPushChannelSoundName = "铃声名称";
}
```

Kotlin

Java



```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
pushInfo.androidFCMChannelID = "厂商申请的channelID"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
pushInfo.setAndroidFCMChannelID("厂商申请的channelID");
```

注意

Android 8.0之前的方式，IMSDK 6.1.2155 及以上版本支持。

Android 8.0之后的方式，IMSDK 7.0.3754 及以上版本支持。

FCM 自定义铃声或者设置 ChannelID 仅支持证书模式。

版权所有：腾讯云计算（北京）有限责任公司

iOS

最近更新时间：2024-05-08 11:37:24

本文介绍如何替换 TUICallKit 的来电铃声，来电铃声分为**应用铃声**和**离线推送铃声**。

一、 设置应用铃声

设置应用铃声有两种方式：替换铃声音频、调用设置铃声接口。

1、替换音频文件

如果您通过源码依赖 TUICallKit 组件，您可以替换 `Resources\\AudioFile` 文件夹下的三个音频文件来达到替换铃声的目的：

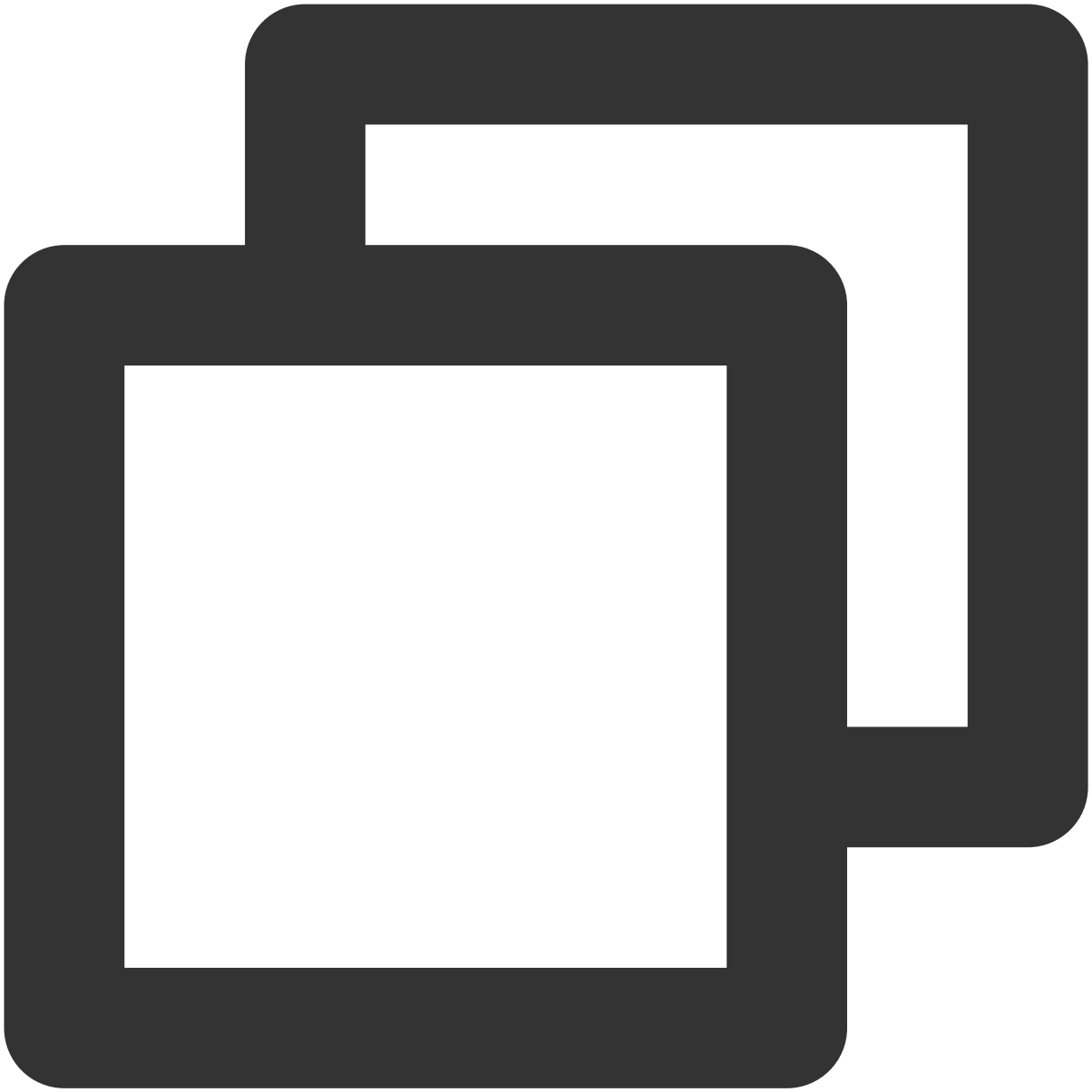
文件名	用途
phone_dialing.mp3	发起呼叫时的铃音
phone_hangup.mp3	被挂断的铃音
phone_ringing.mp3	接到呼叫时的铃音

2、设置铃声接口

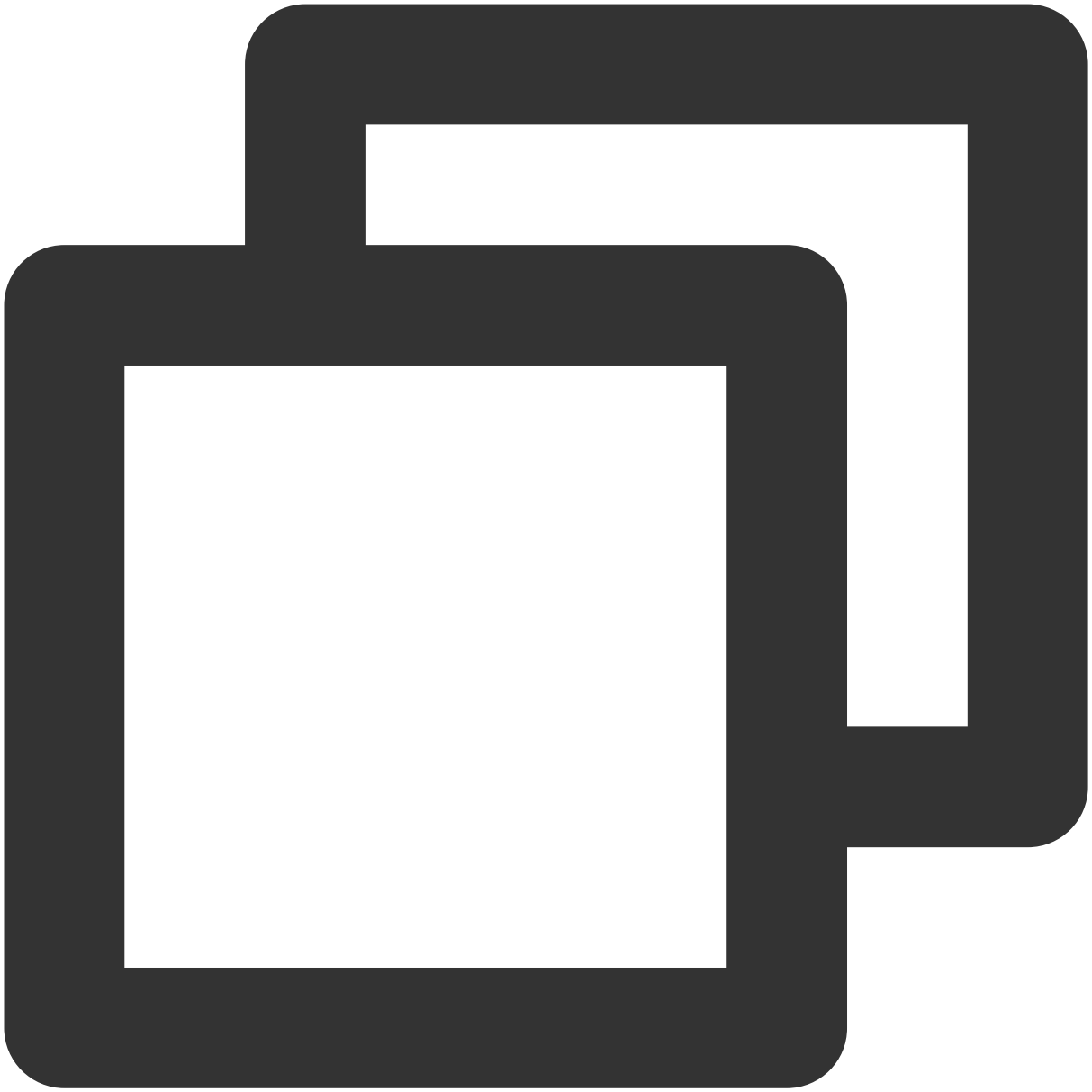
您也可以通过 setCallingBell 接口设置来电铃声。

Swift

Objective-C



```
TUICallKit.createInstance().setCallingBell(filePath: " ")
```



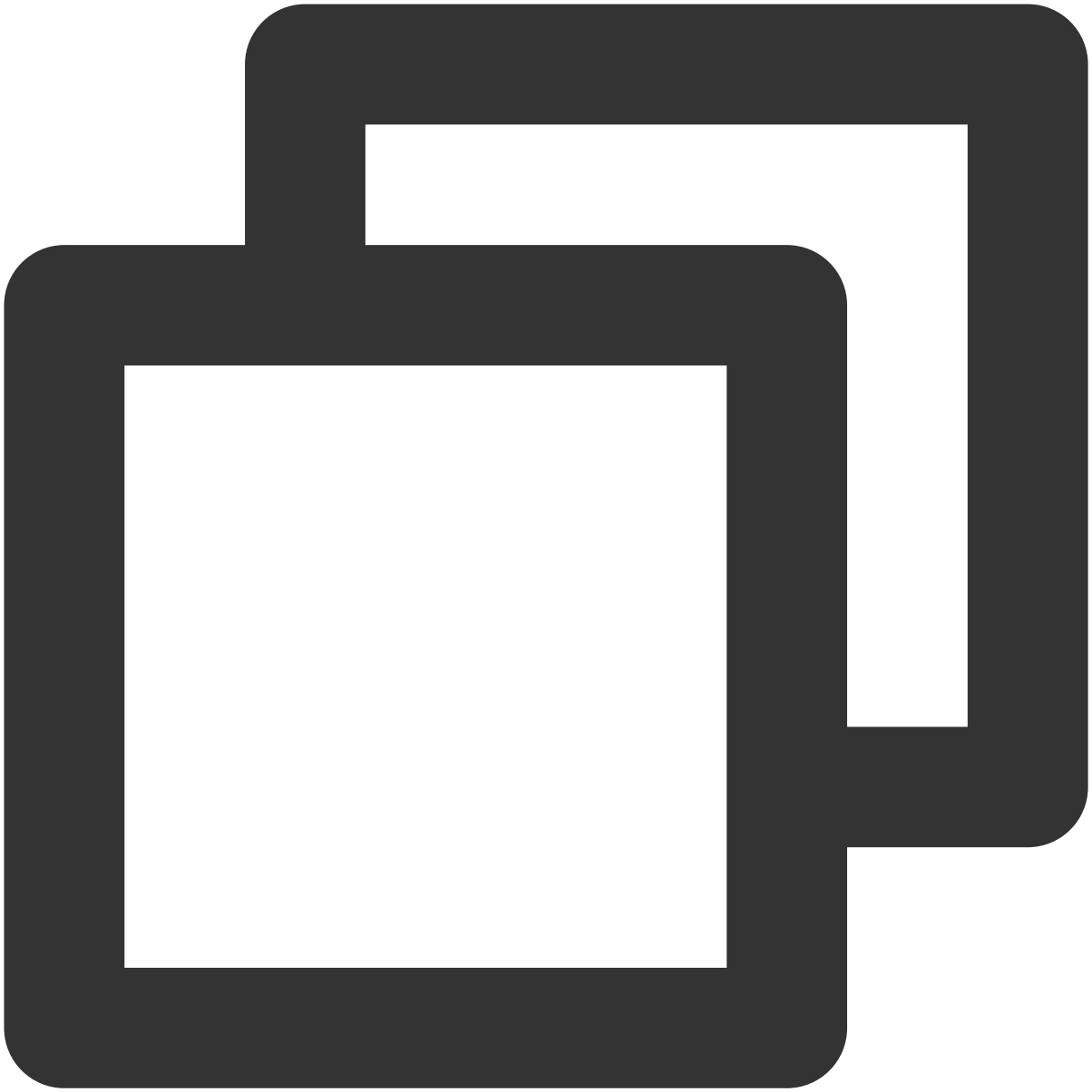
```
[[TUICallKit createInstance] setCallingBell:@" "];
```

3、设置静音模式

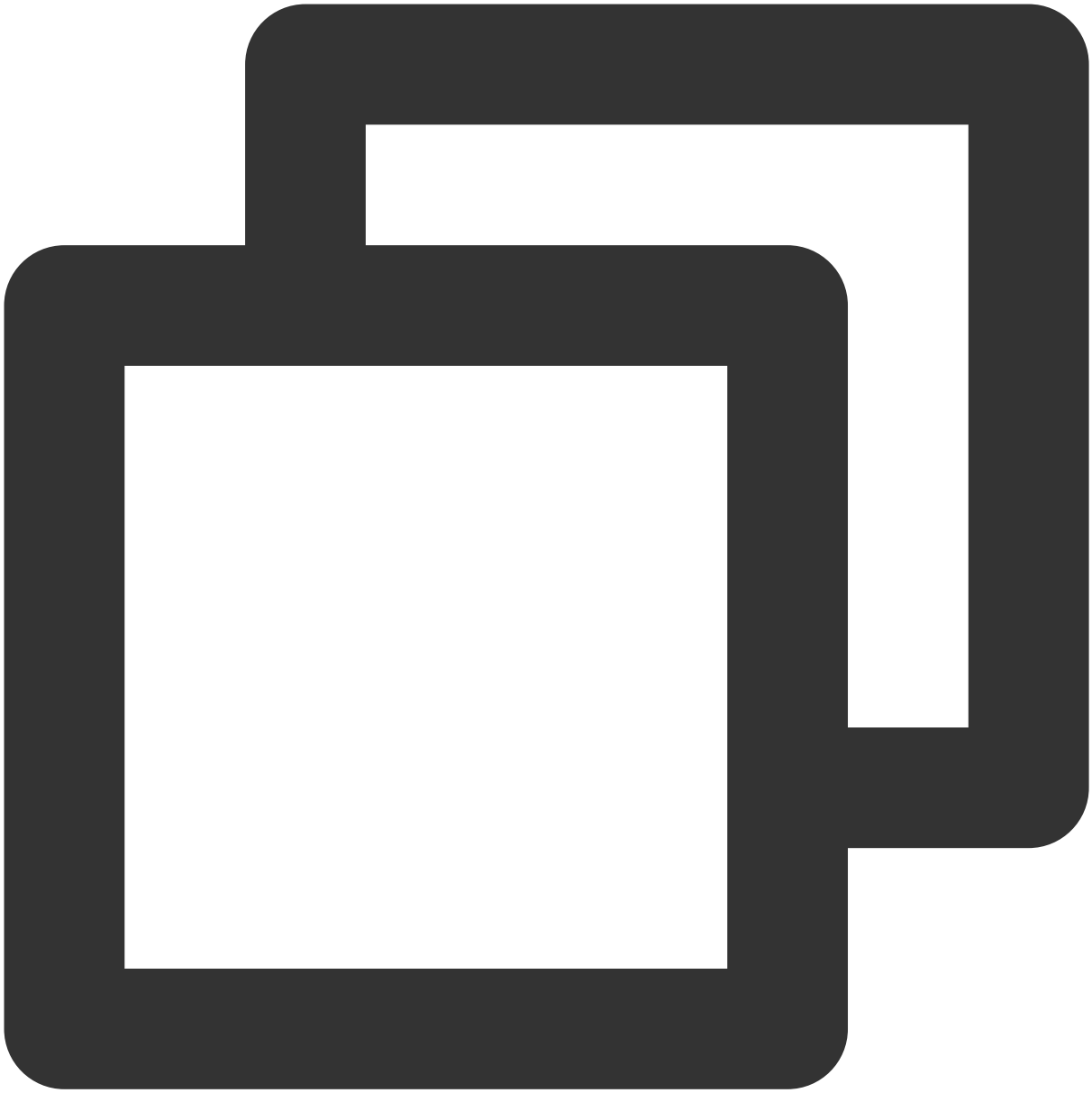
如果您不需要响铃，您可以通过 enableMuteMode 设置静音模式。

Swift

Objective-C



```
TUICallKit.createInstance().setCallingBell(enable: true)
```



```
[[TUICallKit createInstance] enableMuteMode: YES];
```

二、设置离线推送铃音

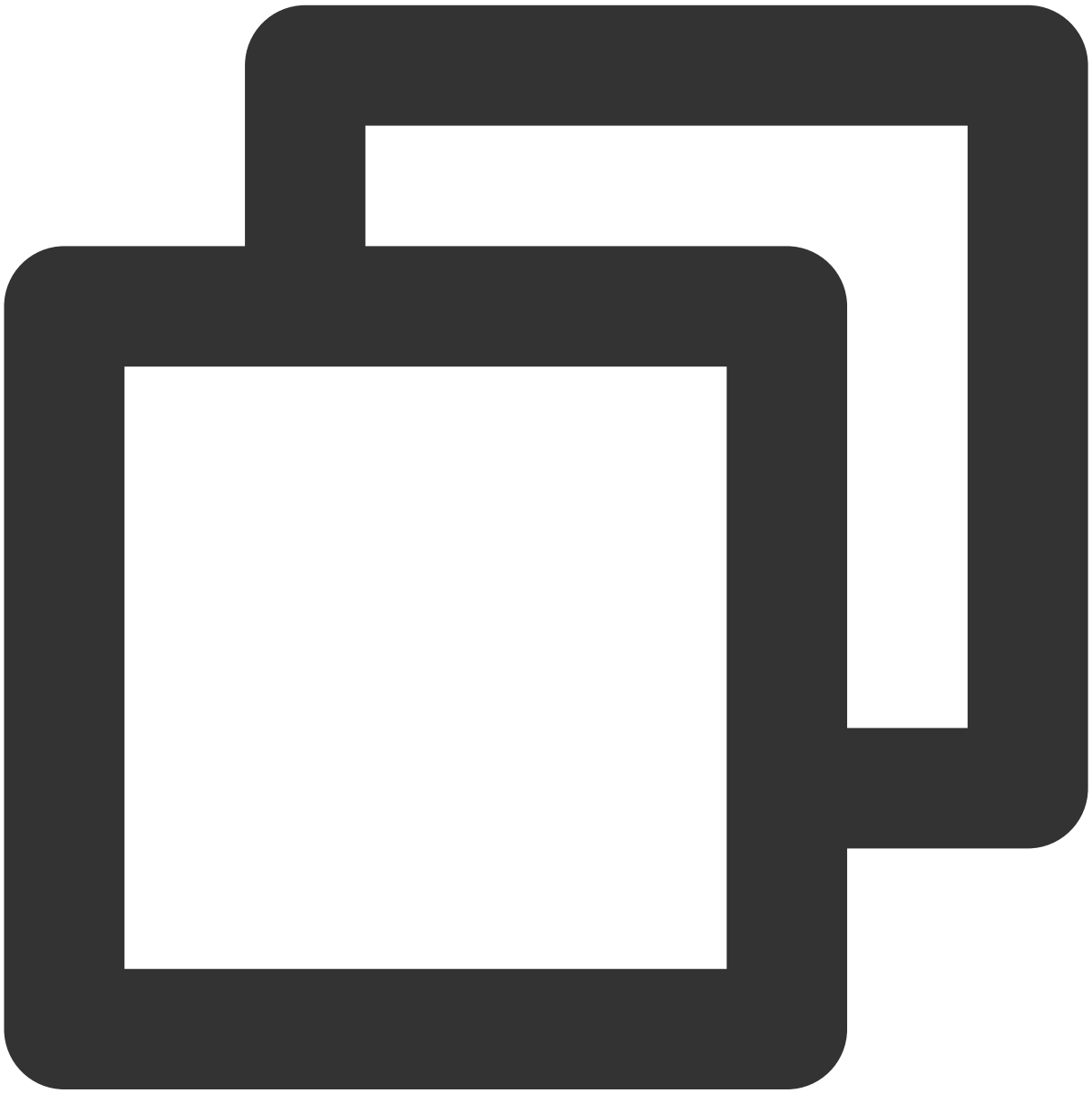
Voip 推送不支持自定义推送铃声。 APNs 推送可以在调用 call 接口拨打电话时设置 params 的 offlinePushInfo 中的 iOSSound 字段， iOSSound 传语音文件名。

注意：

离线推送声音设置（仅对 iOS 生效），如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀名）设置给 iOSSound。
铃声时长应小于30s。

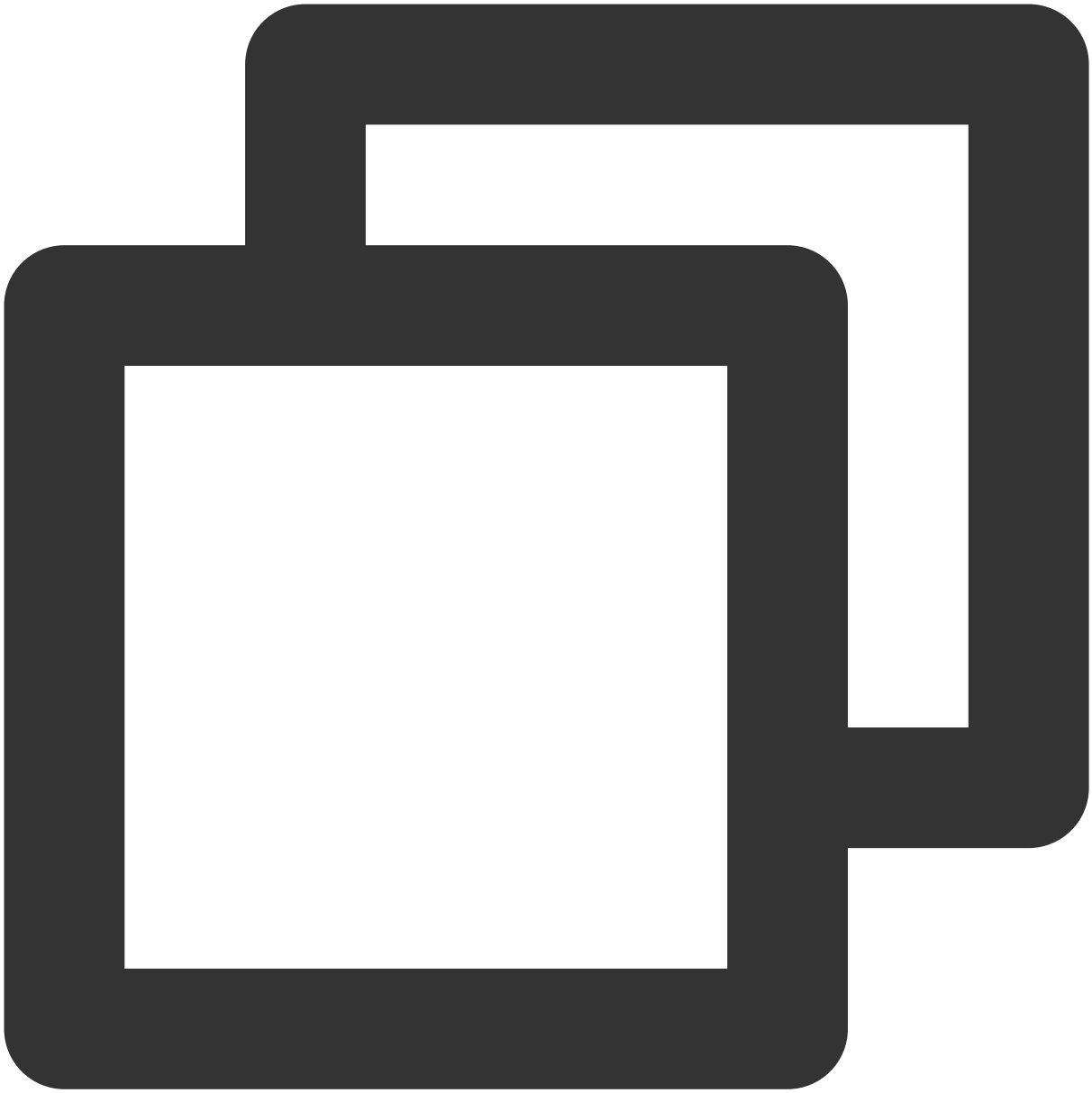
Objective-C

Swift



```
[[TUICallKit createInstance] call:@"mike 的 id" params:[self getCallParams] callMed

- (TUICallParams *)getCallParams {
```

```
    TUIOfflinePushInfo *offlinePushInfo = [self createOfflinePushInfo];
    TUICallParams *callParams = [TUICallParams new];
    callParams.offlinePushInfo = offlinePushInfo;
    callParams.timeout = 30;
    return callParams;
}

+ (TUIOfflinePushInfo *)createOfflinePushInfo {
    TUIOfflinePushInfo *pushInfo = [TUIOfflinePushInfo new];
    pushInfo.title = @"";
    pushInfo.desc = TUICallingLocalize(@"TUICallKit.have.new.invitation");
    pushInfo.iOSPushType = TUICallIOSOfflinePushTypeAPNs;
    pushInfo.ignoreIOSBadge = NO;
    pushInfo.iOSSound = @"phone_ringing.mp3";
    pushInfo.AndroidSound = @"phone_ringing";
    // OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
    // OPPO must set a ChannelID to receive push messages. This channelID needs to
    pushInfo.AndroidOPPOChannelID = @"tuikit";
    // FCM channel ID, you need change PrivateConstants.java and set "fcmPushChanne
    pushInfo.AndroidFCMChannelID = @"fcm_push_channel";
    // VIVO message type: 0-push message, 1-System message(have a higher delivery r
    pushInfo.AndroidVIVOClassification = 1;
    // HuaWei message type: https://developer.huawei.com/consumer/cn/doc/developmen
    pushInfo.AndroidHuaWeiCategory = @"IM";
    return pushInfo;
}
```



```
let params = TUICallParams()
let pushInfo: TUIOfflinePushInfo = TUIOfflinePushInfo()
pushInfo.title = "TUICallKit"
pushInfo.desc = "TUICallKit.have.new.invitation"
pushInfo.iOSPushType = .apns
pushInfo.ignoreIOSBadge = false
pushInfo.iOSSound = "phone_ringing.mp3"
pushInfo.androidSound = "phone_ringing"
// OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
// OPPO must set a ChannelID to receive push messages. This channelID needs to be t
pushInfo.androidOPPOChannelID = "tuikit"
```

```
// FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
pushInfo.androidFCMChannelID = "fcm_push_channel"
// VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
pushInfo.androidVIVOClassification = 1
// HuaWei message type: https://developer.huawei.com/consumer/cn/doc/development/HM
pushInfo.androidHuaWeiCategory = "IM"
params.userData = "User Data"
params.timeout = 30
params.offlinePushInfo = pushInfo
TUICallKit.createInstance().call(userId: "123456", callMediaType: .audio, params: p

} fail: {
    code, message in
}
```

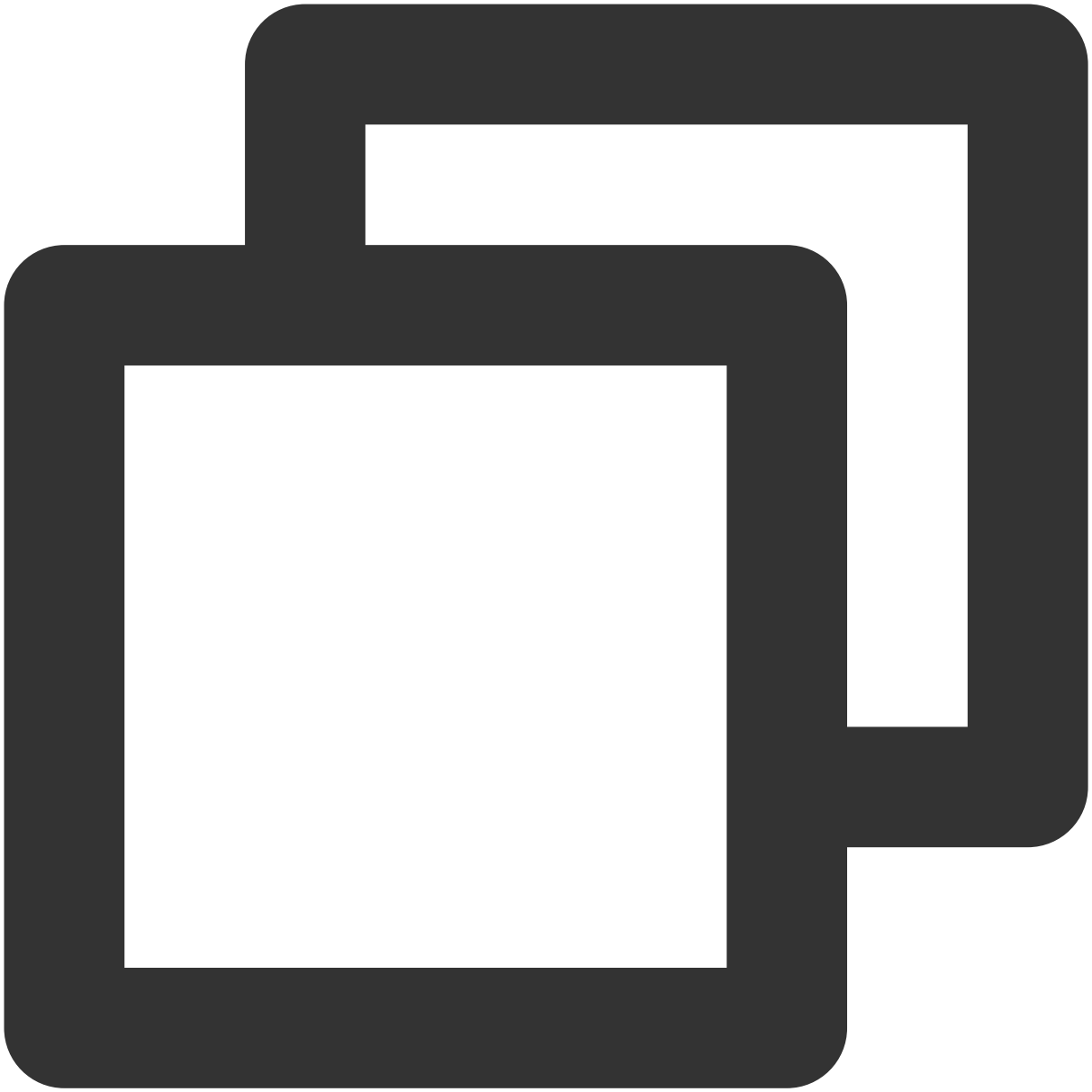
Web&H5

最近更新时间：2024-05-08 11:37:24

本文介绍自定义铃声、静音来电铃声功能的使用。

设置来电铃声

仅限传入本地 MP3 格式的文件地址，确保该文件可以访问的。
如需恢复默认铃声 filePath 传空即可。
使用 ES6 import 方式引入铃声文件。
说明：
Vue ≥ v3.0.0 支持



```
import filePath from '../public/ring.mp3';

try {
  await TUICallKitServer.setCallingBell(filePath?: string);
} catch (error: any) {
  alert(`[TUICallKit] setCallingBell API failed. Reason: ${error}`);
}
```

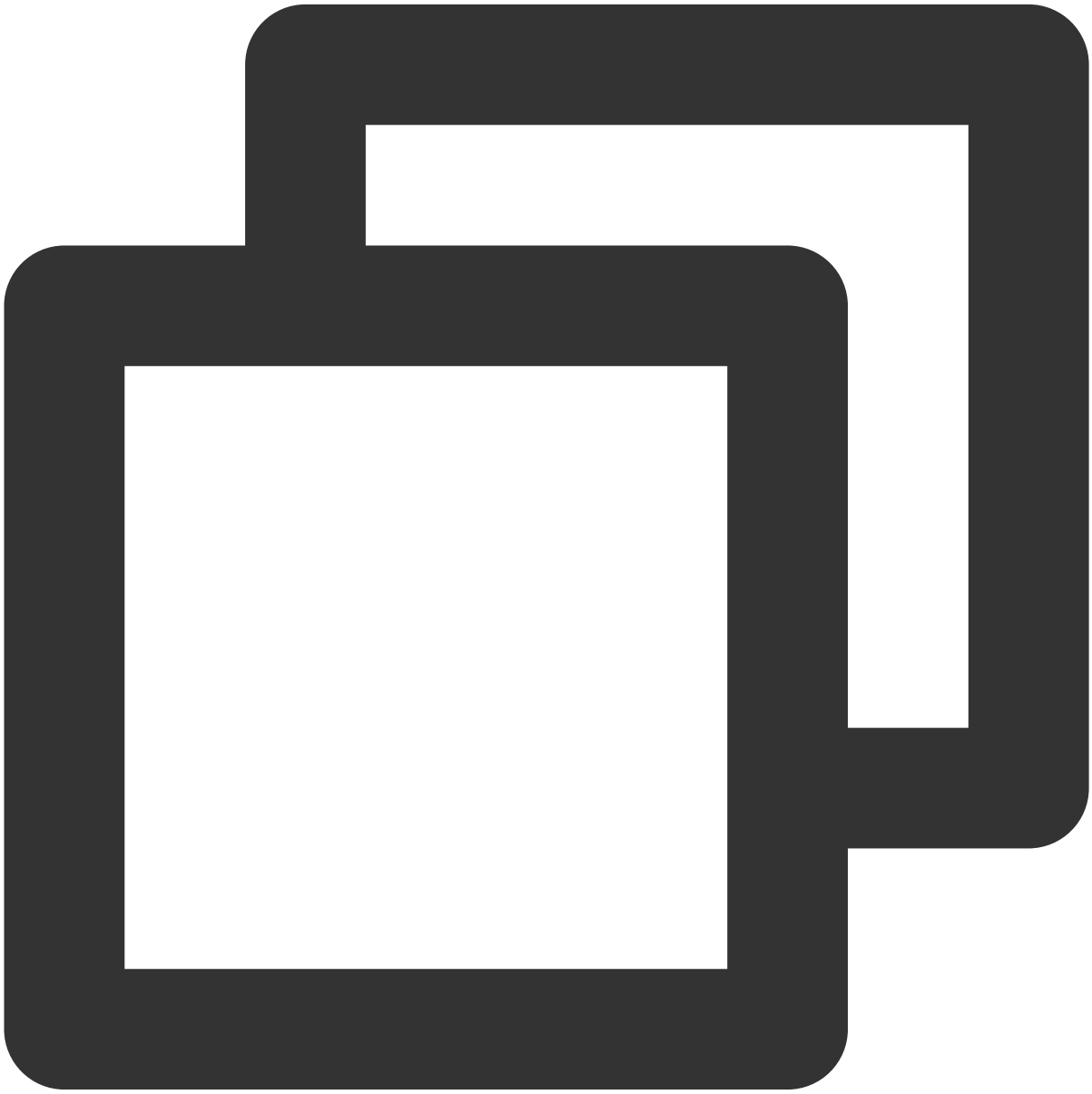
静音来电铃声

开启/关闭来电铃声。

开启后，收到通话请求时，不会播放来电铃声。

说明：

Vue ≥ v3.1.2 支持



```
try {
  await TUICallKitServer.enableMuteMode(enable: boolean);
} catch (error: any) {
```

```
  alert(`[TUICallKit] enableMuteMode API failed. Reason: ${error}`);
}
```

Flutter

最近更新时间：2024-05-08 11:37:24

本文介绍如何替换 TUICallKit 的来电铃声，来电铃声分为**应用铃声**和**离线推送铃声**。

一、 设置应用铃声

设置应用铃声有两种方式：替换铃声音频、调用设置铃声接口。

1、替换音频文件

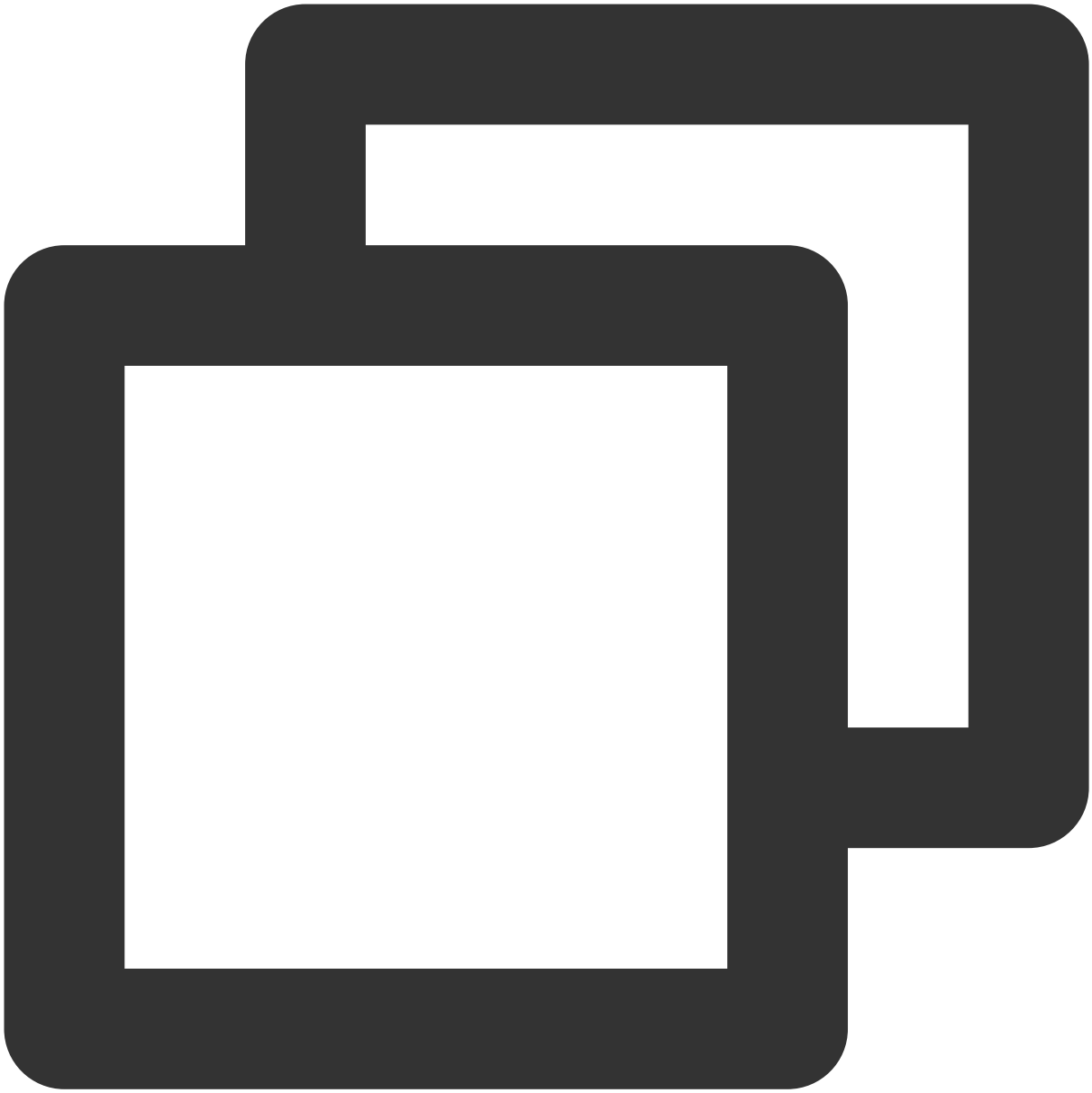
如果您通过源码依赖 TUICallKit 组件，您可以替换 **assets\\audios** 文件夹下的三个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的铃音
phone_hangup.mp3	被挂断的铃音
phone_ringing.mp3	接到呼叫时的铃音

2、设置铃声接口

您也可以通过 setCallingBell 接口设置来电铃声。

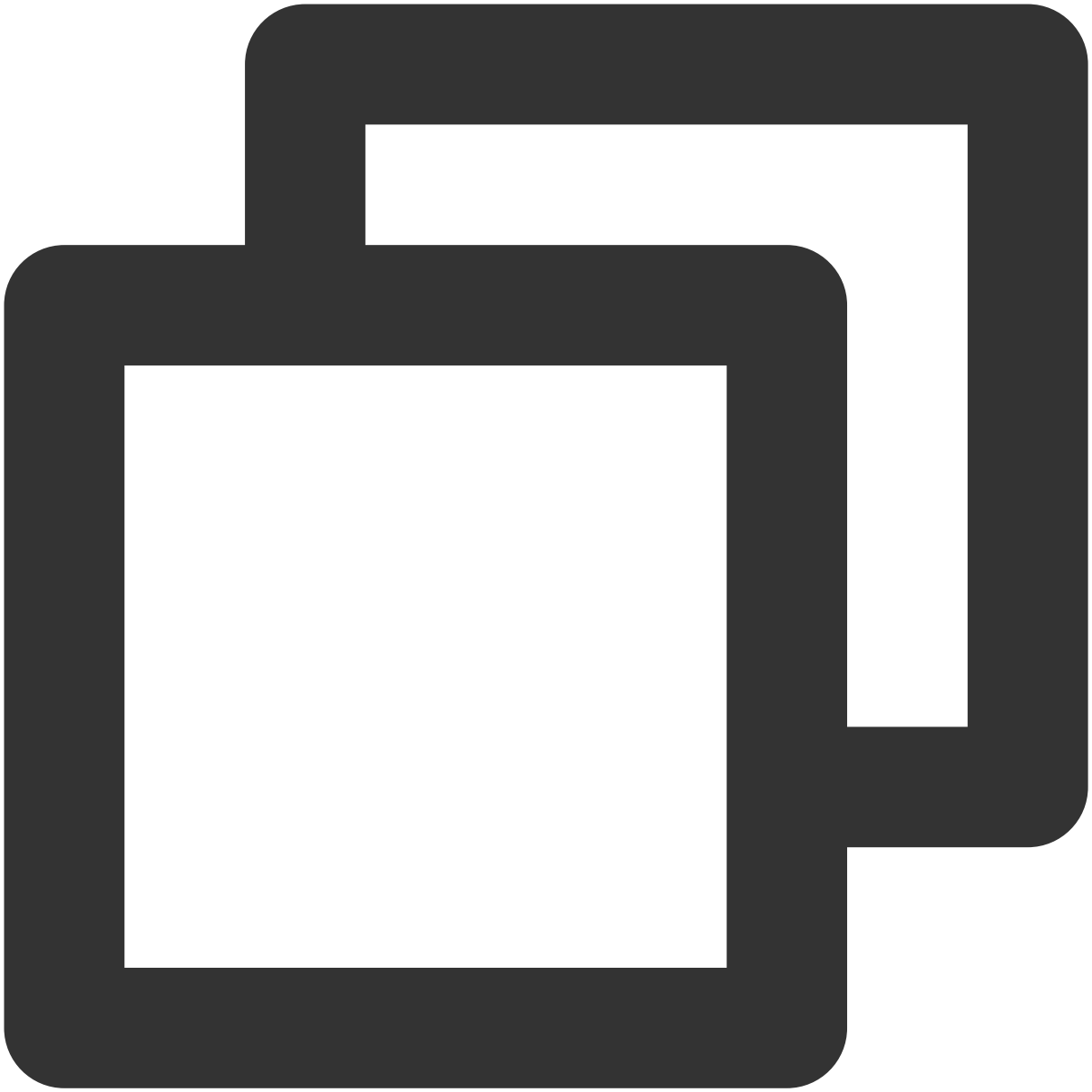
版权所有：腾讯云计算（北京）有限责任公司



```
TUICallKit.instance.setCallingBell('flie path');
```

3、 设置静音模式

如果您不需要响铃，您可以通过 enableMuteMode 设置静音模式。



```
TUICallKit.instance.enableMuteMode(true);
```

二、设置离线推送铃音

1、 iOS

Voip 推送不支持自定义推送铃声。 APNs 推送可以修改 `call` 和 `groupcall` 接口中的参数 `params` 中的 `TUIOfflinePushInfo.iOSSound` 设置 `iOS` 平台上的离线消息铃声。

2、 Android

注意：

接口支持华为、小米、FCM 和 APNS。

FCM 的推送铃声为设置的应用铃声。

华为、小米 和 APNS 推送铃声设置请在调用 Call 和 GroupCall 的时候设置 `TUIOfflinePushInfo.iOSSound` 的 `iOSSound` 及 `androidSound` 字段。具体详情请参见 自定义铃音。

监听通话状态
Android&iOS&Flutter

最近更新时间：2024-05-08 11:37:24

本文介绍 TUICallKit 组件通话状态回调的使用。

通话状态监听

如果您的业务需要**监听通话的状态**，例如通话开始、结束等通话过程中的事件，可以参考如下代码：

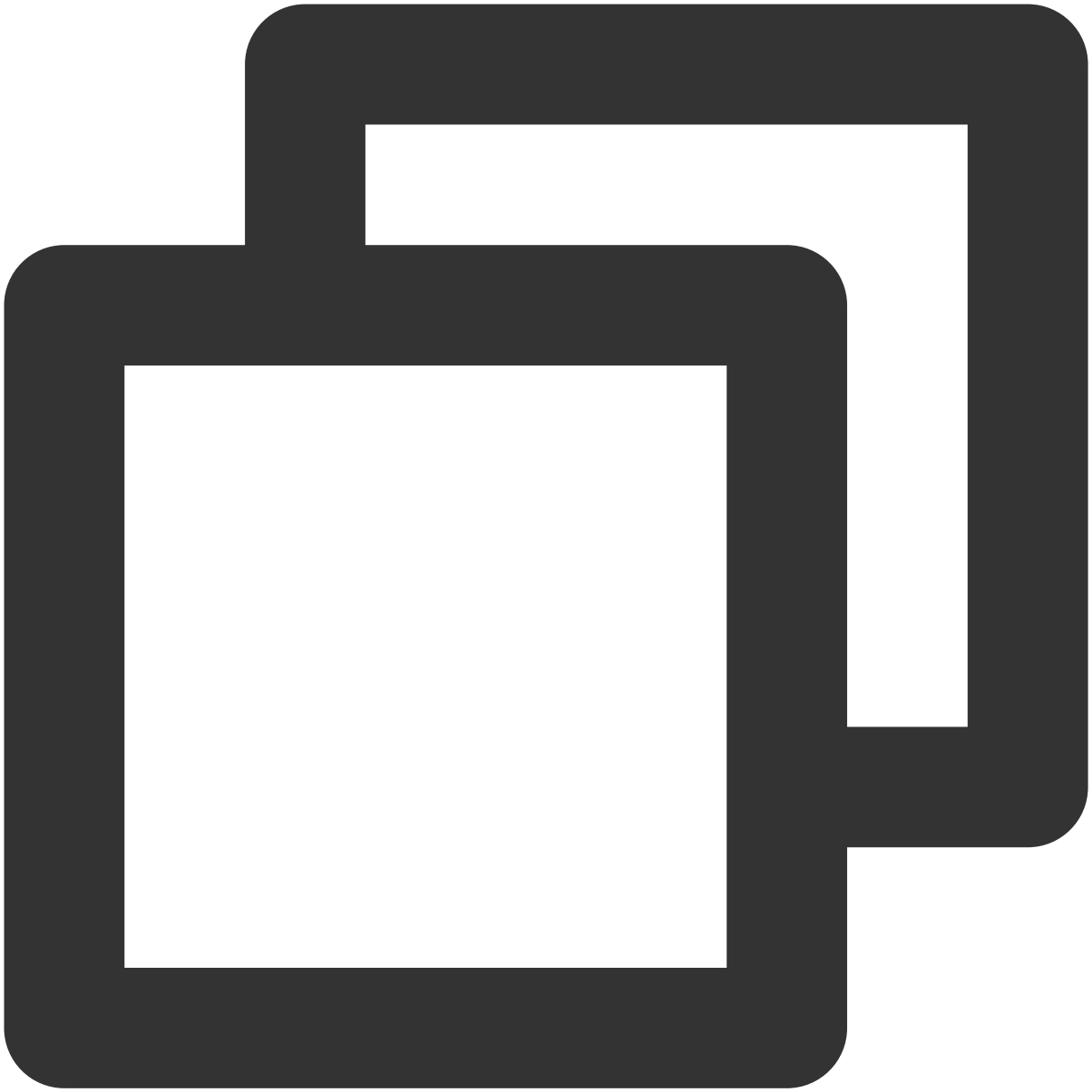
Android（Kotlin）

Android（Java）

iOS（Swift）

iOS（Objective-C）

Flutter（Dart）



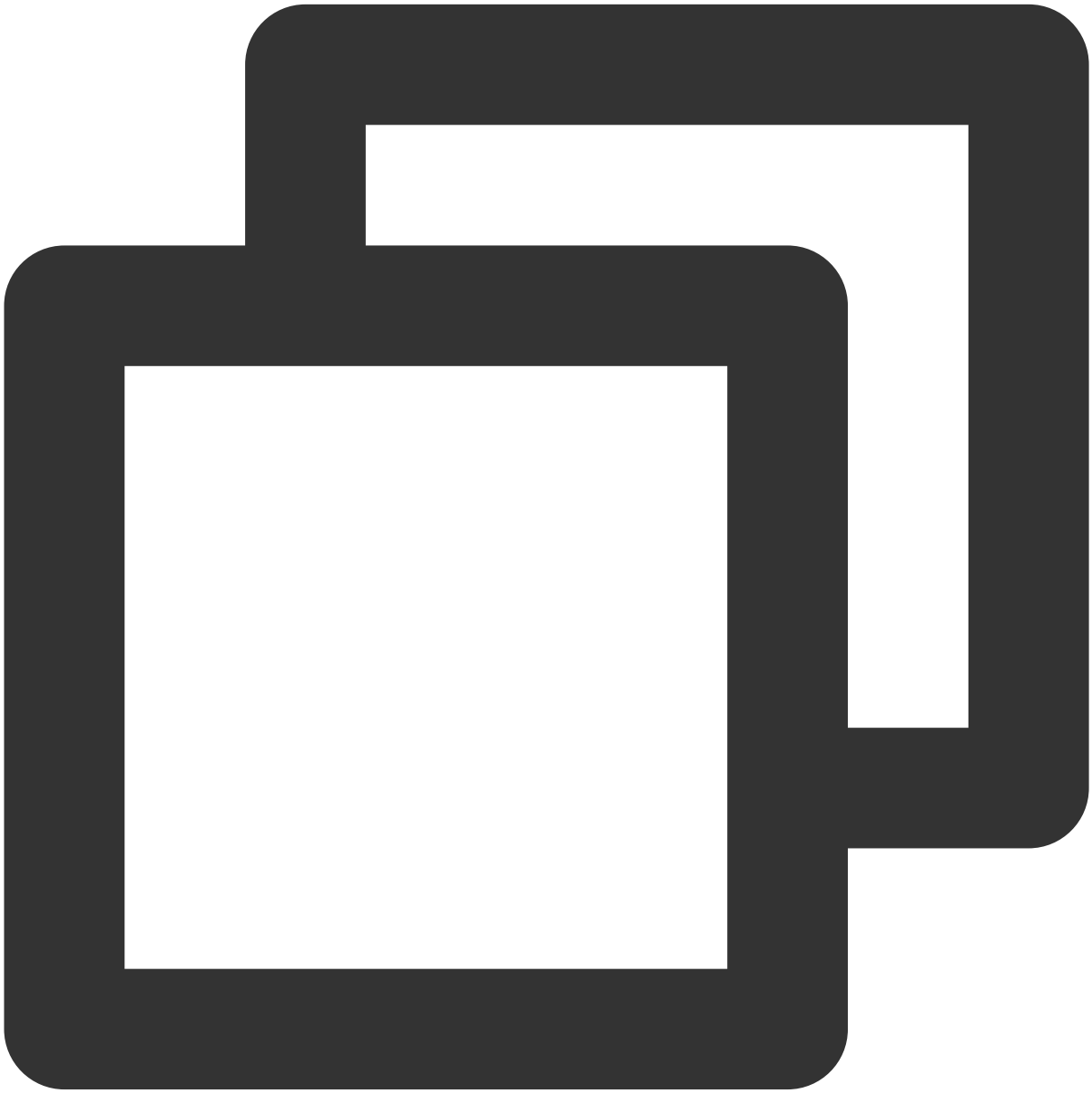
```
private val observer: TUICallObserver = object : TUICallObserver() {
    override fun onCallBegin(roomId: TUICommonDefine.RoomId?, callMediaType: TUICal
    }
    override fun onCallEnd(roomId: TUICommonDefine.RoomId?, callMediaType: TUICallD
    }
    override fun onUserNetworkQualityChanged(networkQualityList: MutableList<TUICom
    }
}
private fun initData() {
    TUICallEngine.createInstance(context).addObserver(observer)
}
```

```
private TUICallObserver observer = new TUICallObserver() {
    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType
    }
    public void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType ca
    }
    public void onUserNetworkQualityChanged(List<TUICommonDefine.NetworkQualityInfo
    }
};
```

```
private void initData(){
    TUICallEngine.createInstance(context).addObserver(observer);
}
```

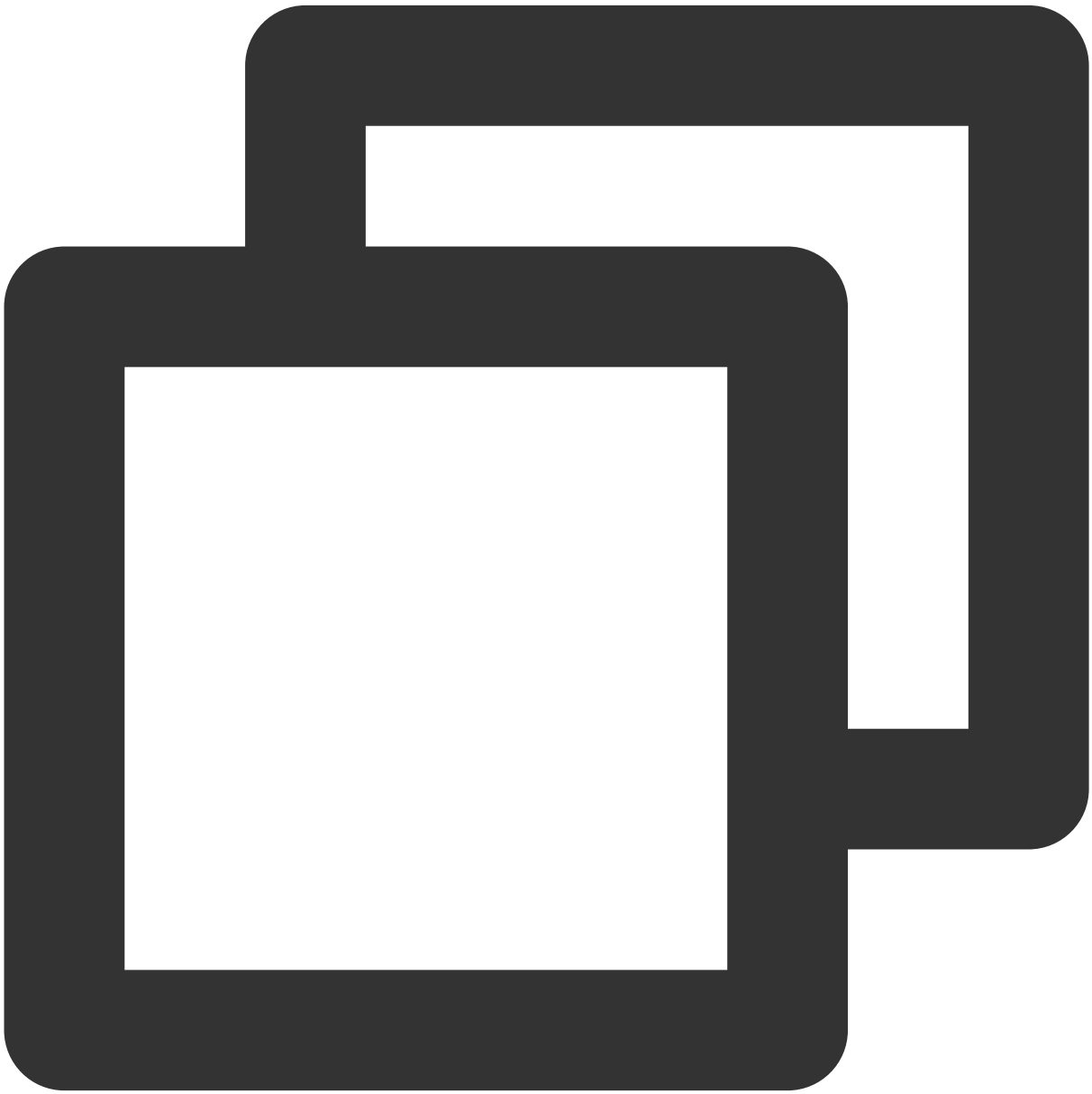
```
import TUICallEngine

TUICallEngine.createInstance().addObserver(self)

public func onCallBegin(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRol

}
```

```
public func onCallEnd(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRole:

}
public func onUserNetworkQualityChanged(networkQualityList: [TUINetworkQualityInfo]

}
```



```
#import <TUICallEngine/TUICallEngine.h>

[[TUICallEngine createInstance] addObserver:self];
```

```
- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaTy

}
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType

}
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual

}
```



```
TUICallObserver observer = TUICallObserver(
```

```
    onError: (int code, String message) {
      //您的回调处理逻辑
    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole
      //您的回调处理逻辑
    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
      //您的回调处理逻辑
    },, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityLis
      //您的回调处理逻辑
    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,
      //您的回调处理逻辑
    }
    ……
))

TUICallEngine.instance.addObserver(observer);
```

注意：

在 Android 平台设置 TUICallObserver 监听回调，需保证回调所在类不会被清除，举例：不建议在 LoginActivity 中添加监听，LoginActivity 销毁的时候，回调也会被清除；建议在应用的 Application 中或者应用主界面进行监听。

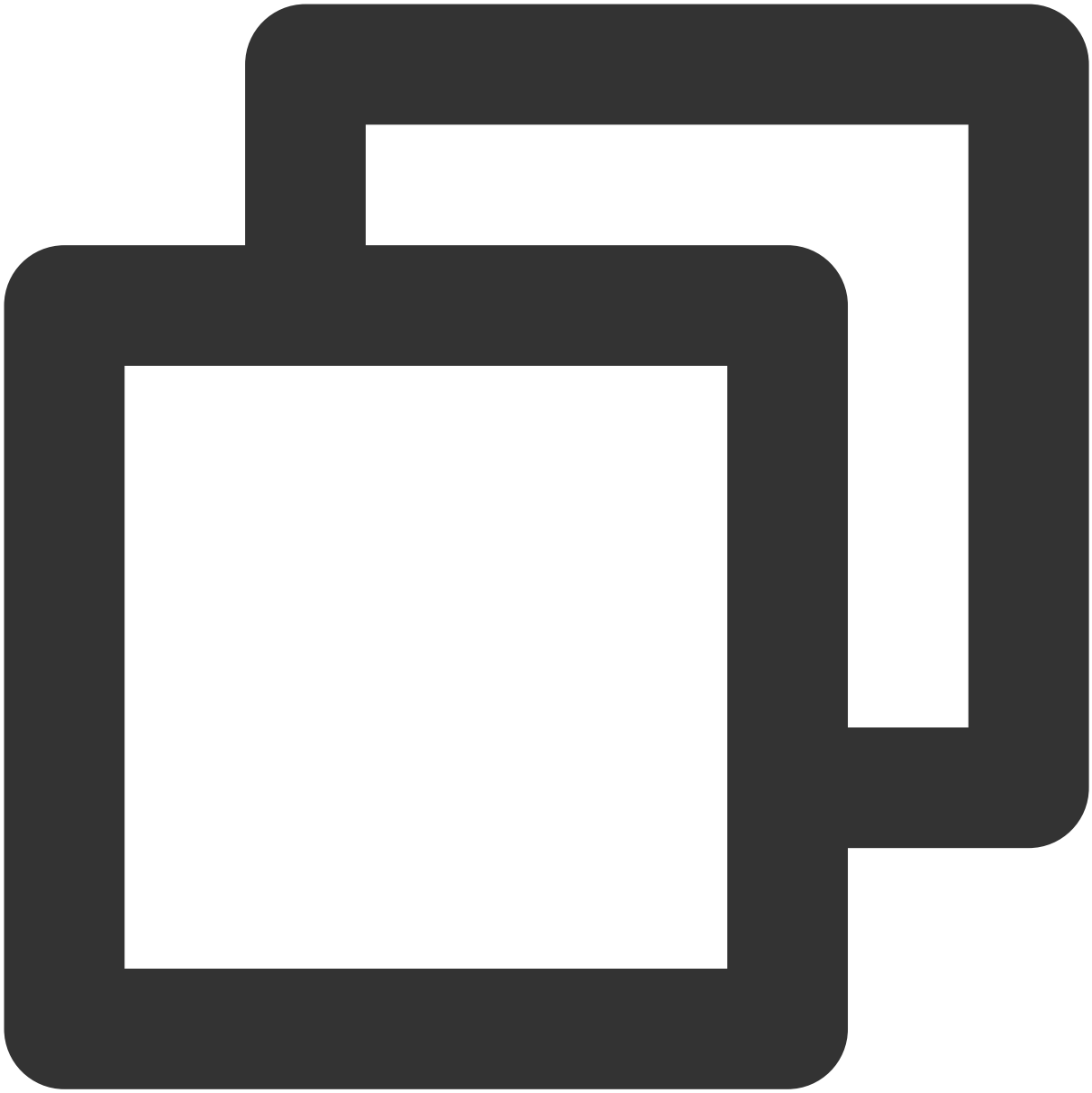
Web&H5

最近更新时间：2024-05-08 11:37:24

本文介绍 TUICallKit 组件通话状态回调的使用。

通话状态监听

如果您的业务需要**监听通话的状态**，例如通话开始、结束等通话过程中的事件（详见 TUICallEvent），可以参见如下代码：



```
import { TUICallEvent } from 'tuicall-engine-webrtc';

let handleUserEnter = function(event) {
    console.log('TUICallEvent.USER_ENTER: ', event);
};

TUICallKitServer.getTUICallEngineInstance().on(TUICallEvent.USER_ENTER, handleUserE
TUICallKitServer.getTUICallEngineInstance().off(TUICallEvent.USER_ENTER, handleUser
```

组件回调事件

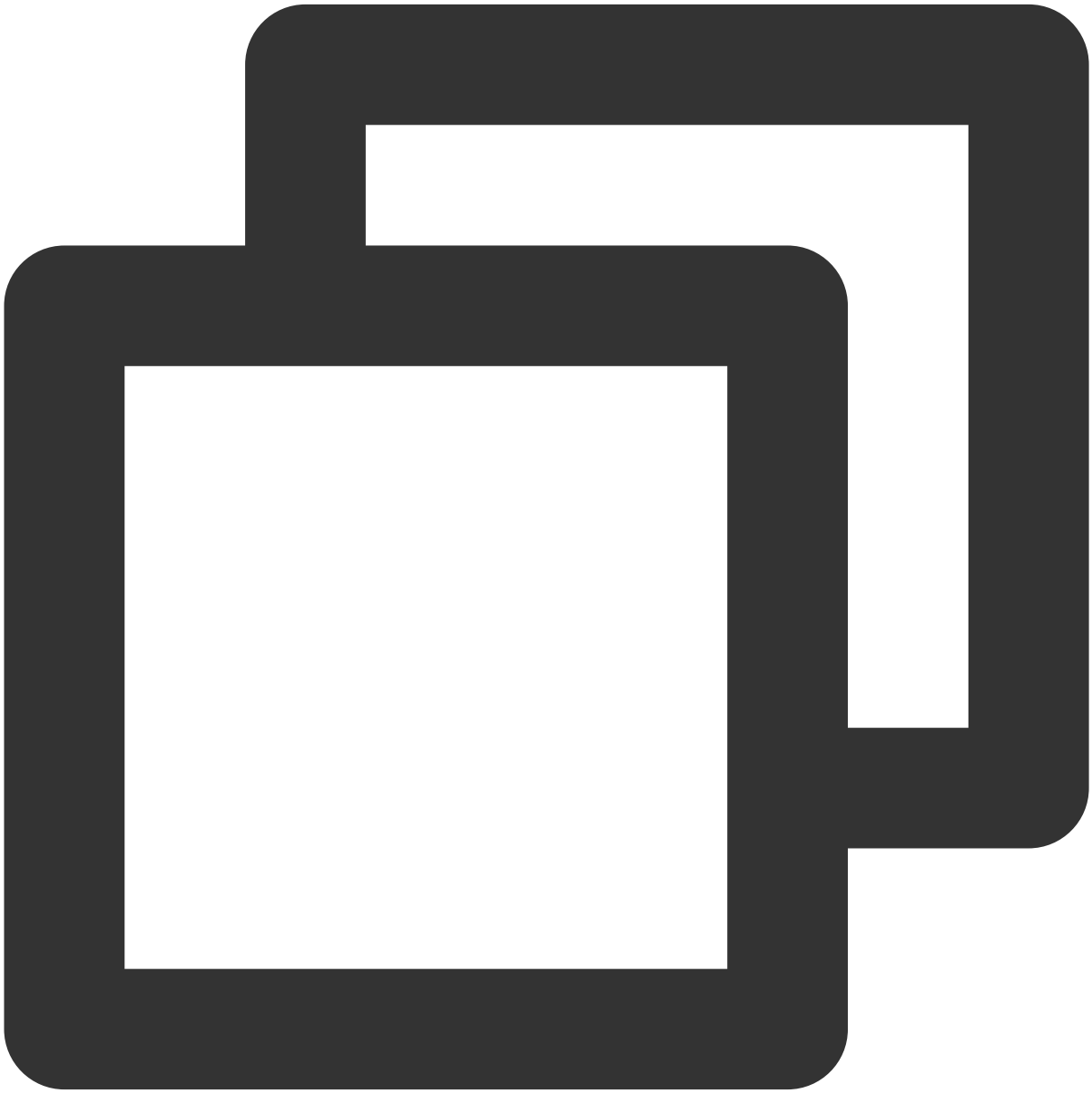
TUICallKit 组件提供了通话状态回调，可以用于业务侧实现更多交互逻辑，具体参见 TUICallKit 属性介绍。

`beforeCalling`：通话开始前会执行。

`afterCalling`：通话完成后执行。

React

Vue



```
function App() {
  const handleBeforeCalling = () => {
```

```
    console.log("[TUICallKit Demo] beforeCalling");
  };
  const handleAfterCalling = () => {
    console.log("[TUICallKit Demo] afterCalling");
  };
  return (
    <TUICallKit
      beforeCalling={handleBeforeCalling}
      afterCalling={handleAfterCalling} />
  )
}
```

```
<template>
  <TUICallKit
    :beforeCalling="handleBeforeCalling"
    :afterCalling="handleAfterCalling" />
</template>
<script setup>
function handleBeforeCalling() {
  console.log("[TUICallKit Demo] beforeCalling");
}
function handleAfterCalling() {
  console.log("[TUICallKit Demo] afterCalling");
}
</script>
```

客户端 API（TUICallKit）
Android
API 概览

最近更新时间：2024-07-30 11:20:02

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的头像、昵称
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
enableIncomingBanner	开启/关闭来电横幅显示

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述

createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，例如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（群组通话版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式

setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

版权所有：腾讯云计算（北京）有限责任公司

关键类型定义

API	描述
TUICallDefine.MediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallDefine.Role	通话的角色，枚举类型：主叫、被叫
TUICallDefine.Status	通话的状态，枚举类型：空闲、待接听、接听中
TUICallDefine.CallParams	通话扩展参数：离线推送消息显示内容
TUICommonDefine.RoomId	音视频房间Id，支持数字、字符串两种类型
TUICommonDefine.Camera	摄像头Id参数，枚举类型：前摄、后摄
TUICommonDefine.AudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUICommonDefine.NetworkQualityInfo	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2024-05-27 17:40:53

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入TUICallKit。

API 概览

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
enableIncomingBanner	开启/关闭来电横幅显示

API 详情

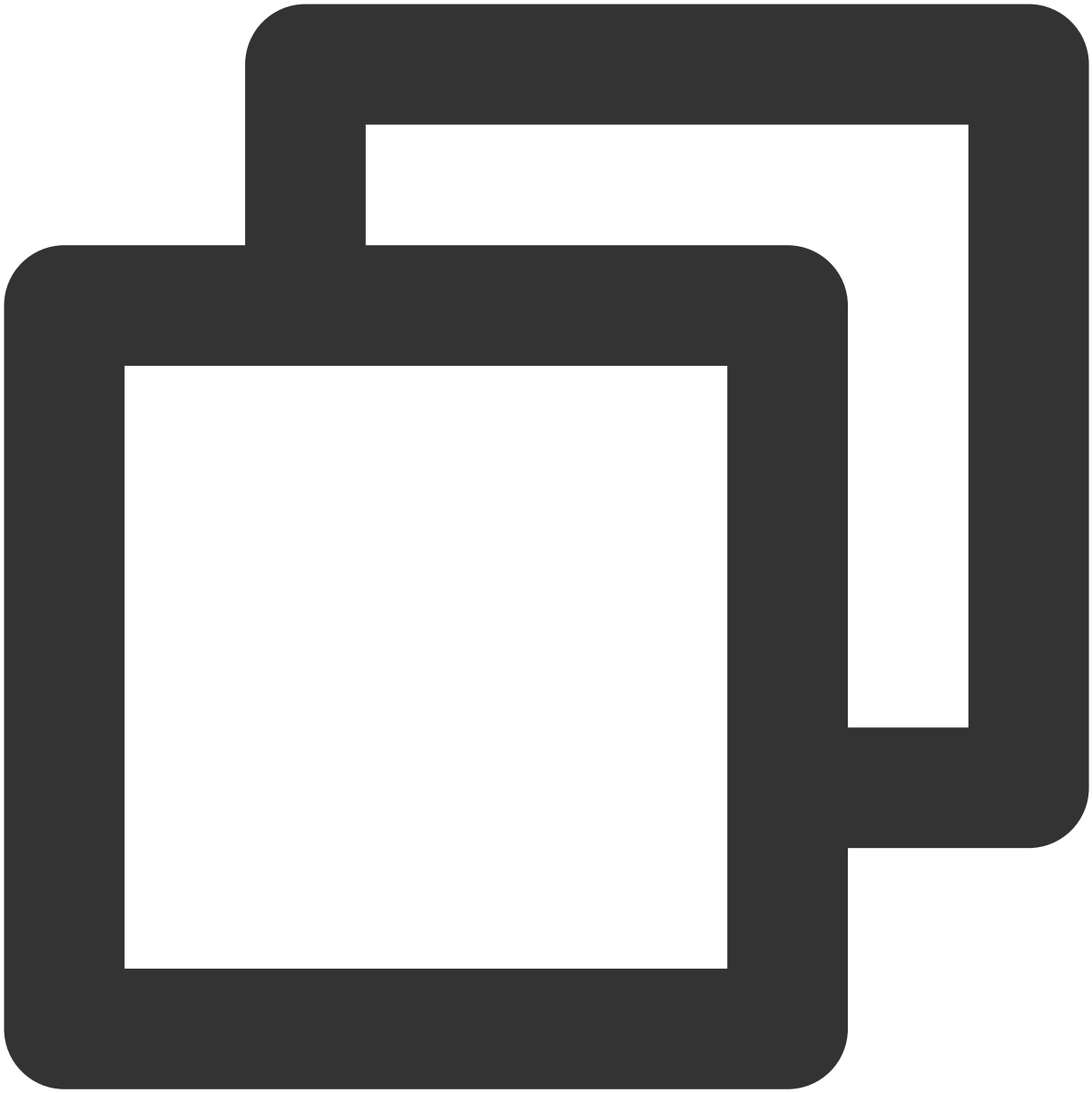
createInstance

创建 TUICallKit 的单例。

Kotlin

Java

```
fun createInstance(context: Context): TUICallKit
```



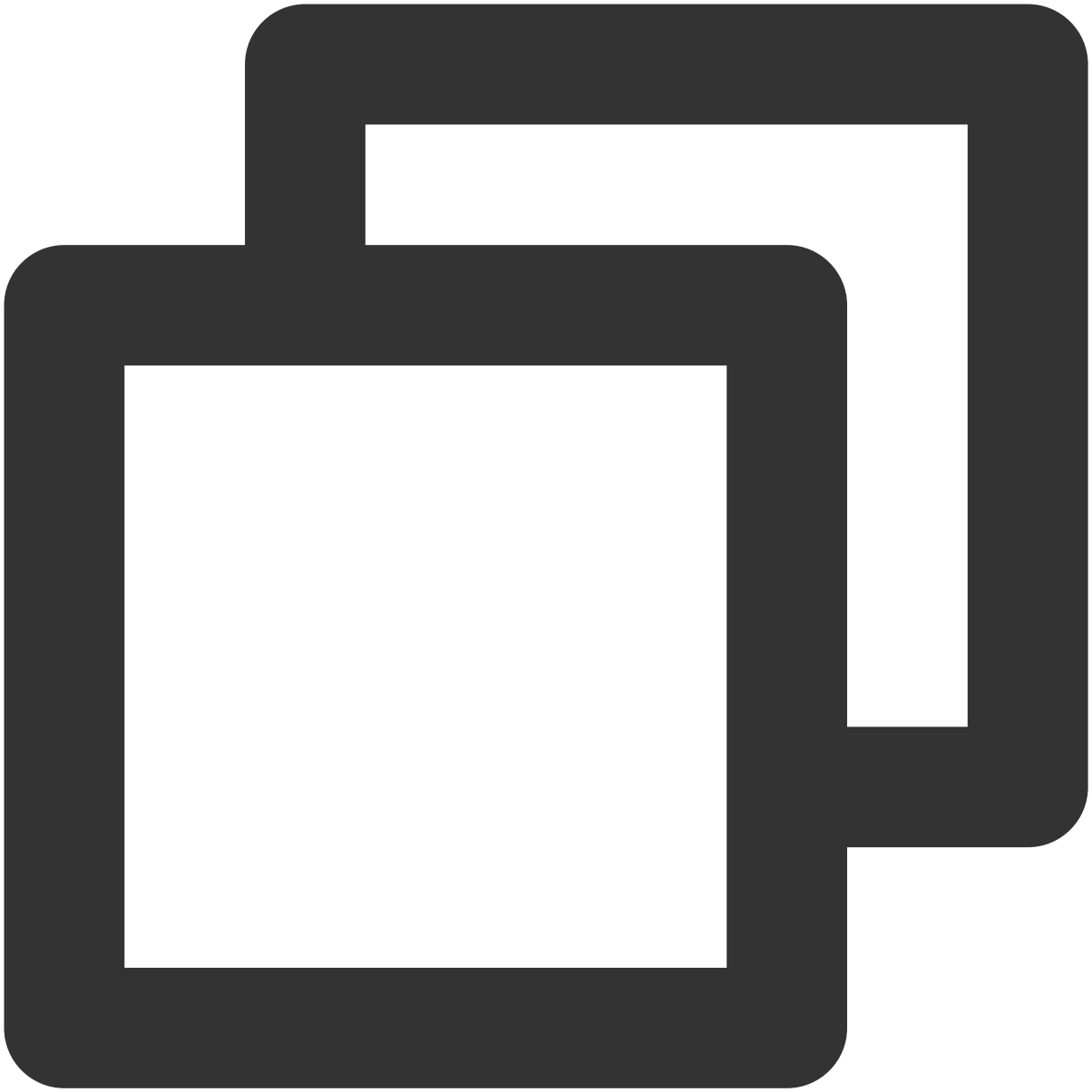
```
TUICallKit createInstance(Context context)
```

setSelfInfo

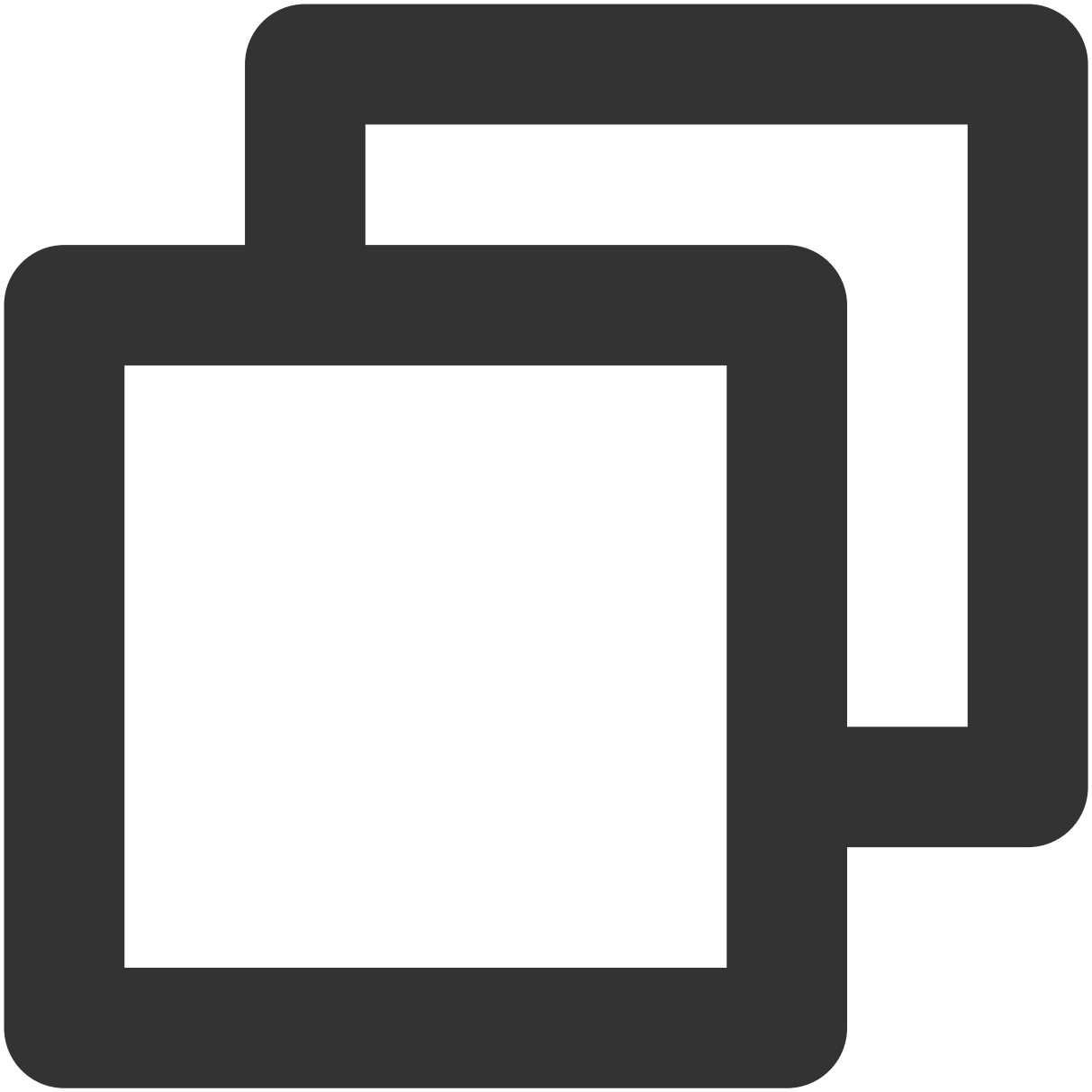
设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是 URL 格式。

Kotlin

Java



```
fun setSelfInfo(nickname: String?, avatar: String?, callback: TUICommonDefine.Callb
```



```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

参数如下表所示：

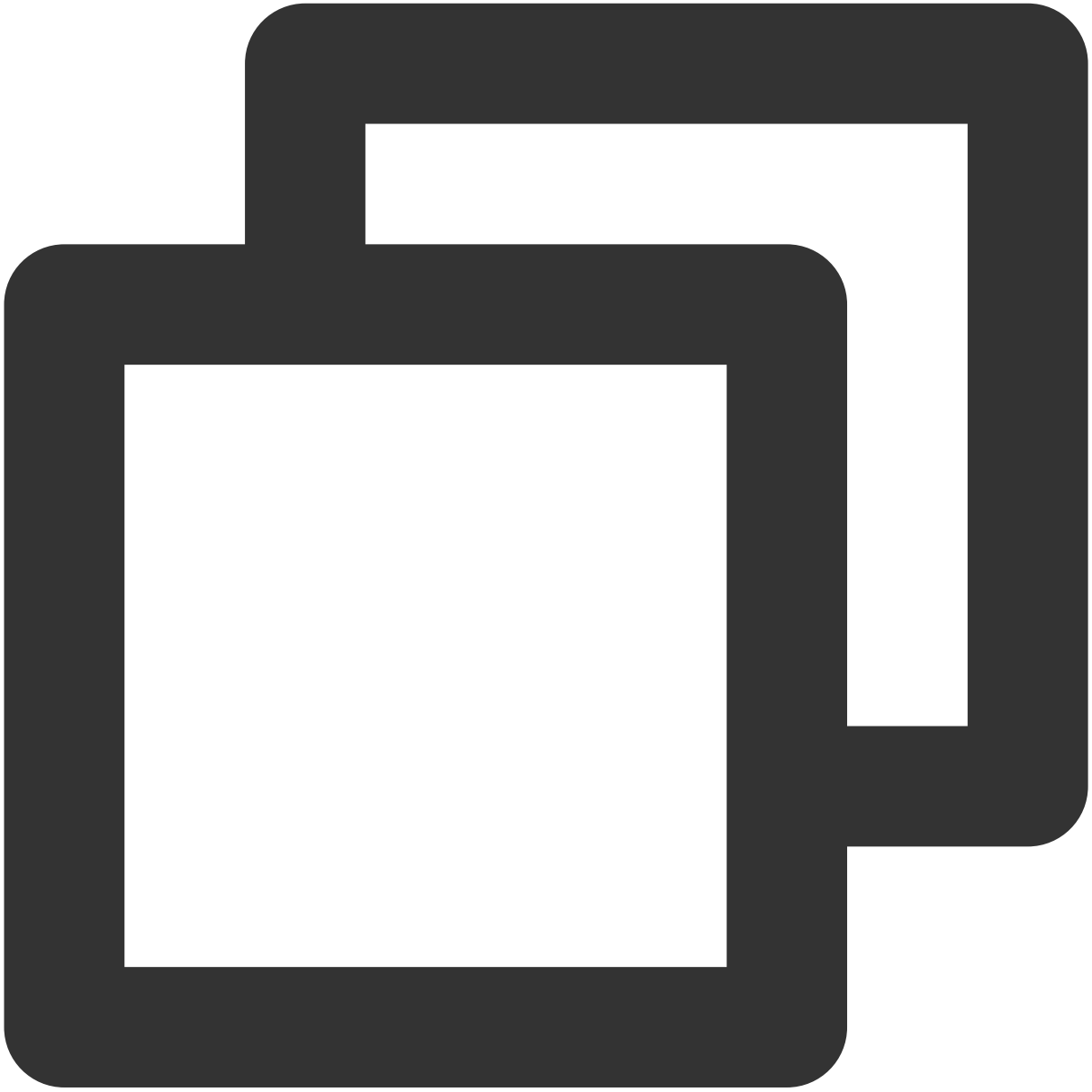
参数	类型	含义
nickname	String	目标用户的昵称
avatar	String	目标用户的头像

call

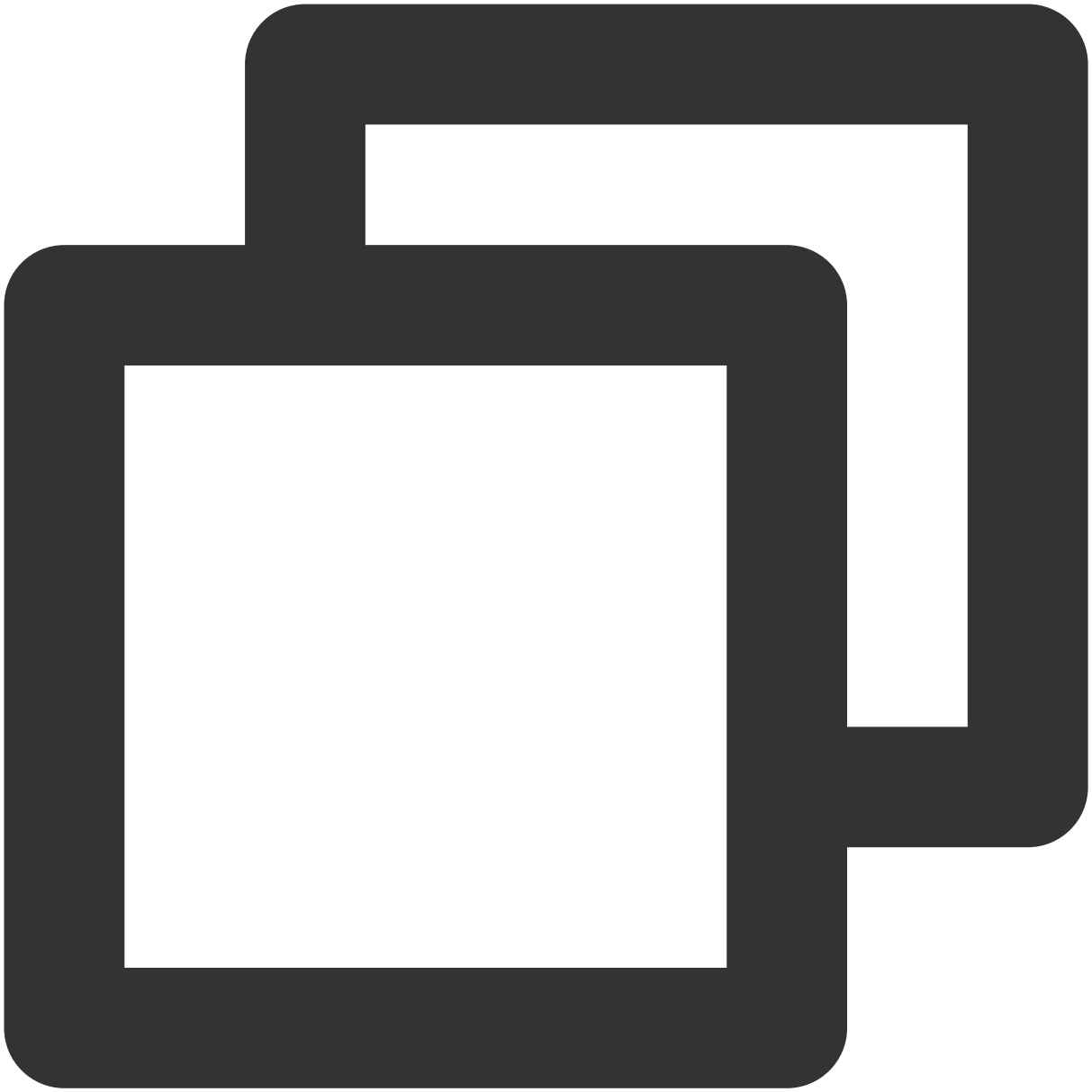
拨打电话（1v1通话）。

Kotlin

Java



```
fun call(userId: String, callMediaType: TUICallDefine.MediaType)
```



```
void call(String userId, TUICallDefine.MediaType callMediaType)
```

参数如下表所示：

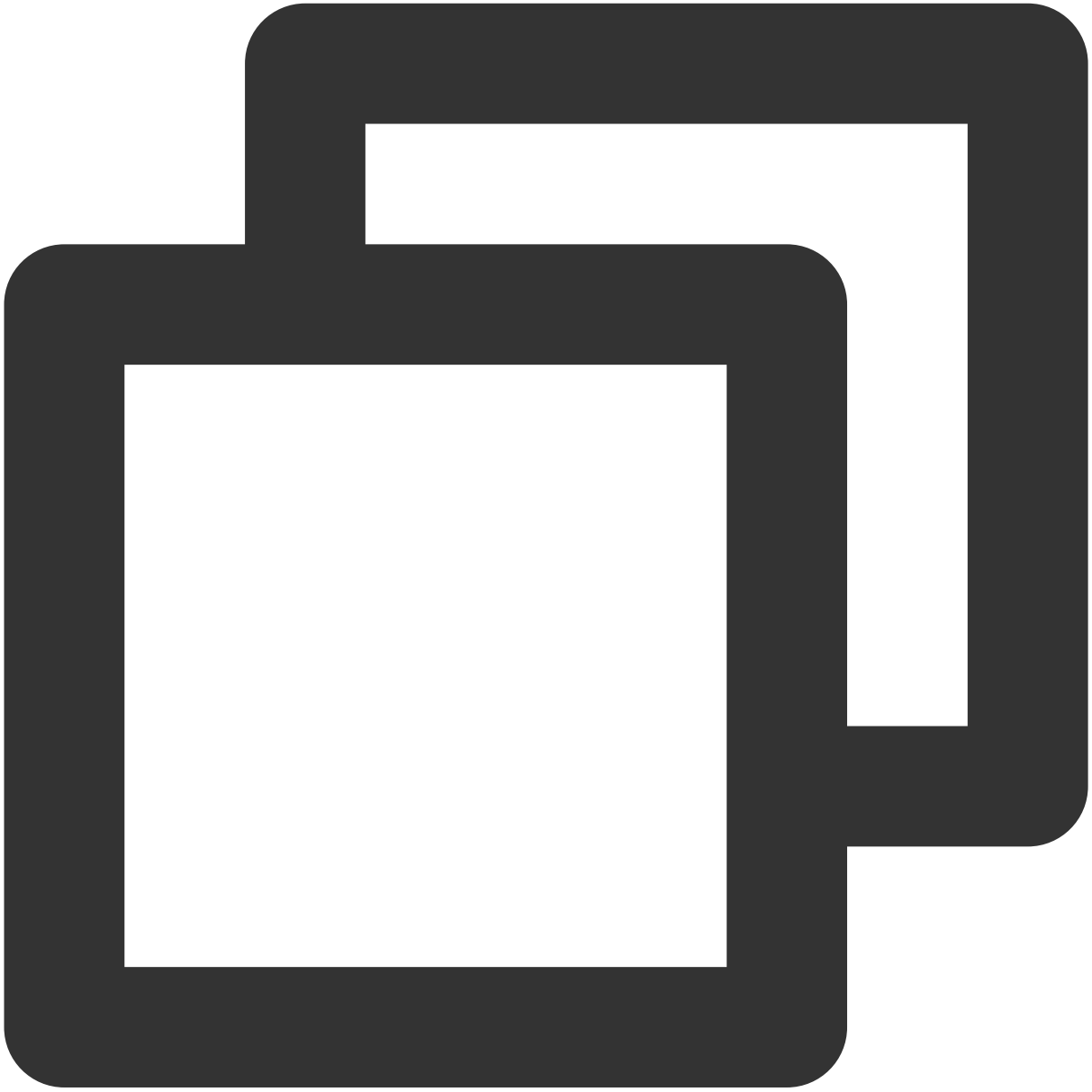
参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

call

拨打电话（1v1通话），支持自定义房间号、通话邀请超时时间，离线推送内容等。

Kotlin

Java



```
fun call(
    userId: String, callMediaType: TUICallDefine.MediaType,
    params: CallParams?, callback: TUICommonDefine.Callback?
)
```

```
void call(String userId, TUICallDefine.MediaType callMediaType,
          TUICallDefine.CallParams params, TUICommonDefine.Callback callback)
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

params	TUICallDefine.CallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话。

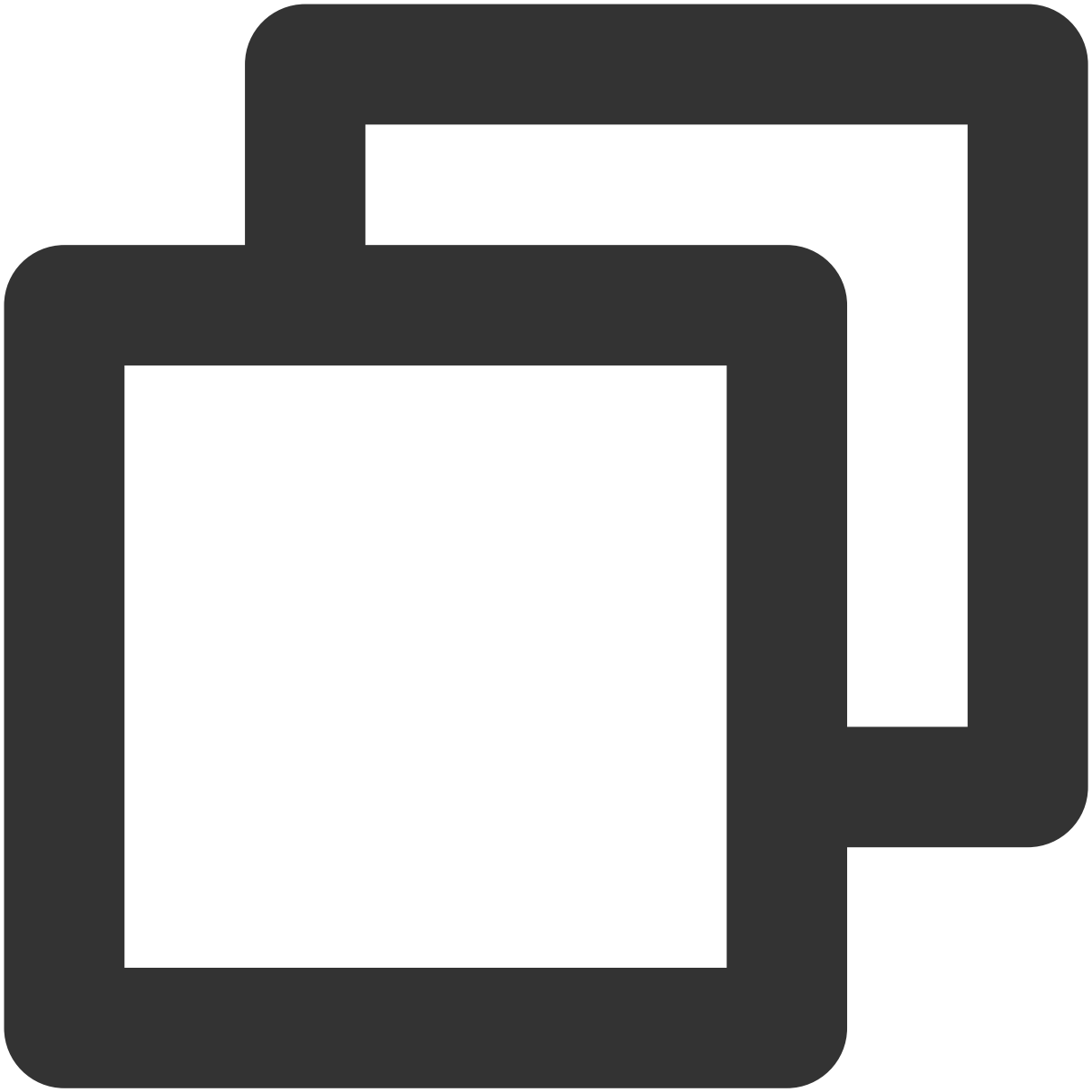
注意

使用群组通话前需要创建IM 群组，如果已经创建，请忽略。

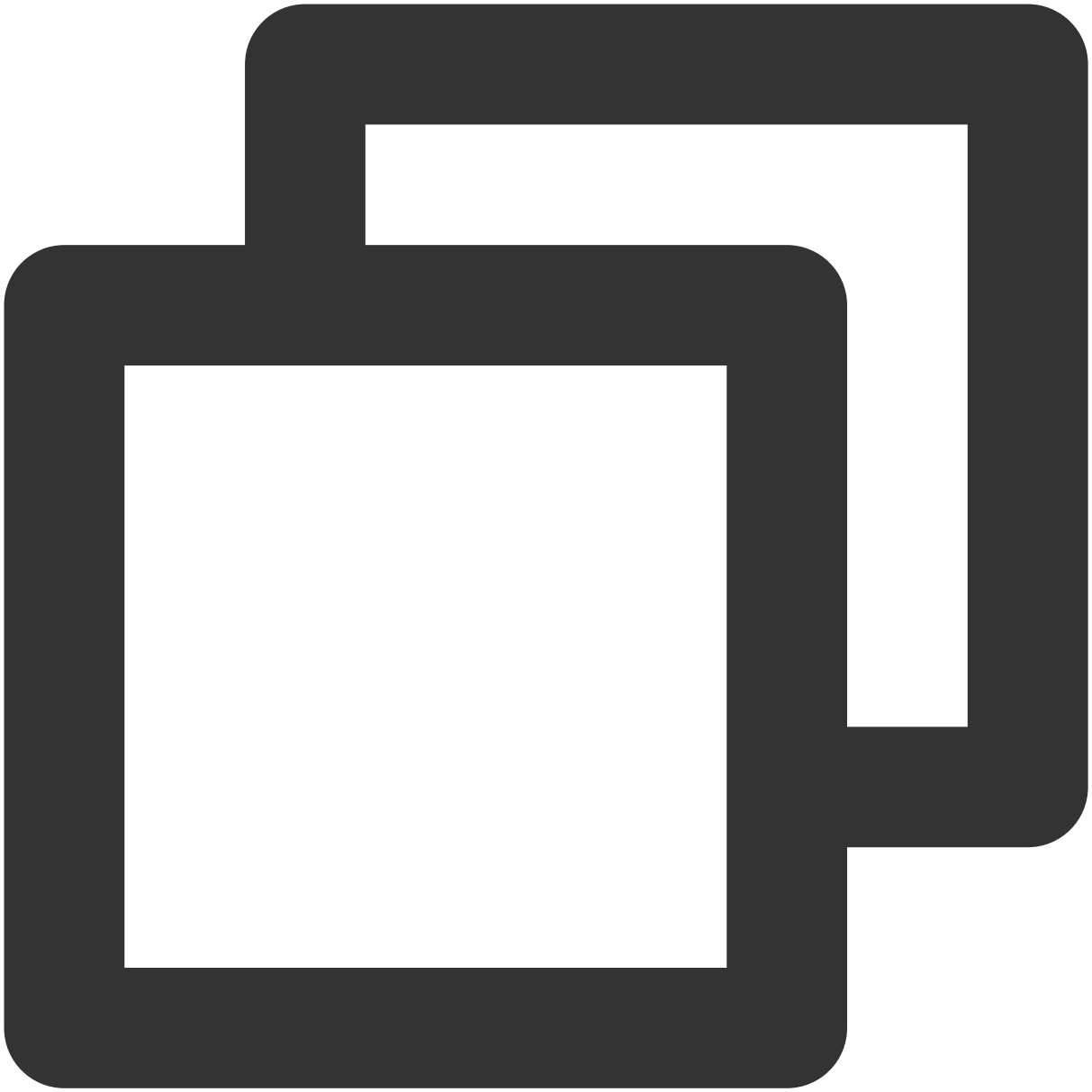
群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Kotlin

Java



```
fun groupCall(groupId: String, userIdList: List<String?>?, callMediaType: TUICallDe
```



```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType cal
```

参数如下表所示：

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

groupCall

发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。

注意

使用群组通话前需要创建IM 群组，如果已经创建，请忽略。

群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Kotlin

Java

```
fun groupCall(
    groupId: String, userIdList: List<String?>?,
```

```
    callMediaType: TUICallDefine.MediaType, params: CallParams?,
    callback: TUICommonDefine.Callback?
)
```

```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType cal
               TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义

groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallDefine.CallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

joinInGroupCall

加入群组中已有的音视频通话。

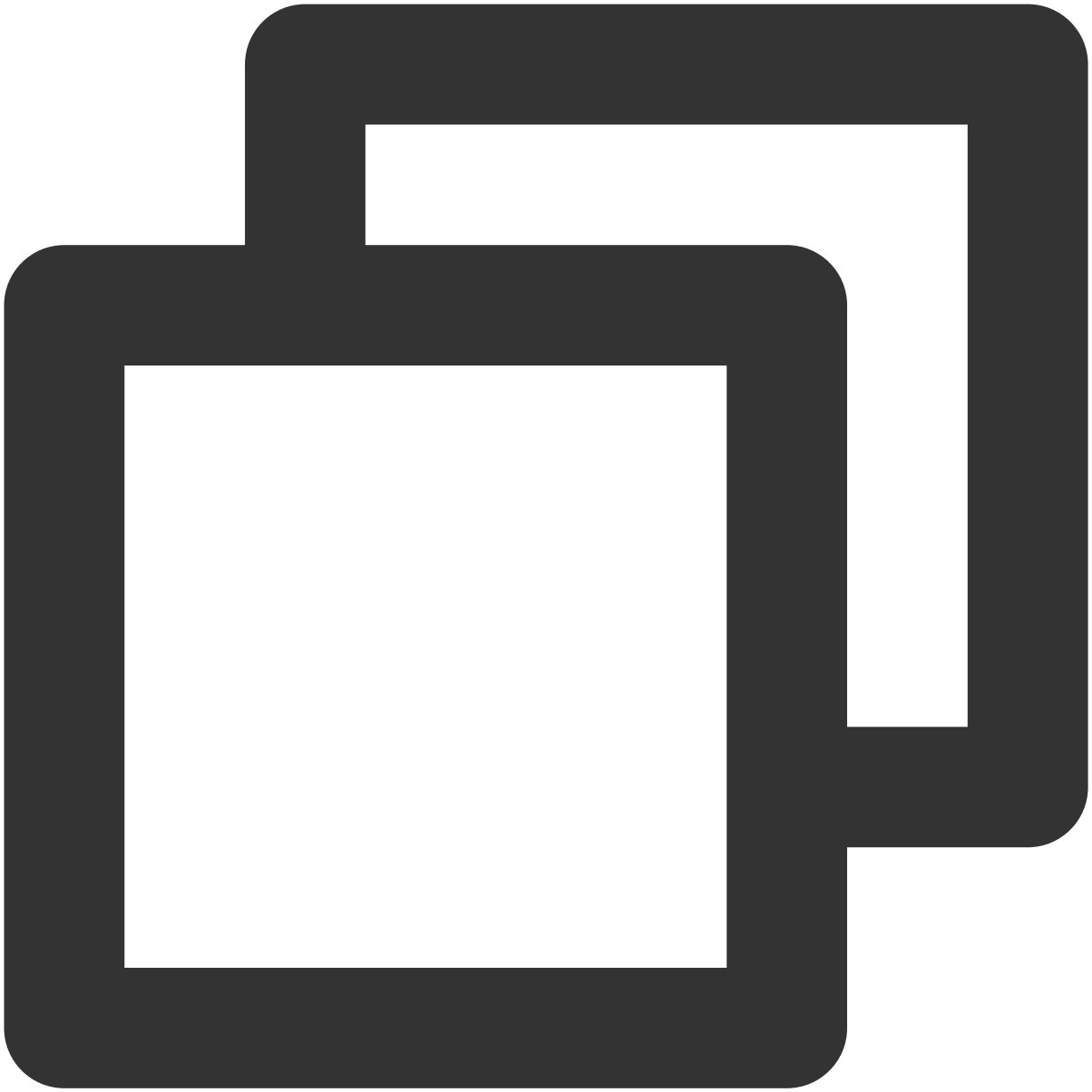
注意

加入群组中已有的音视频通话前，需要提前创建或加入IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。

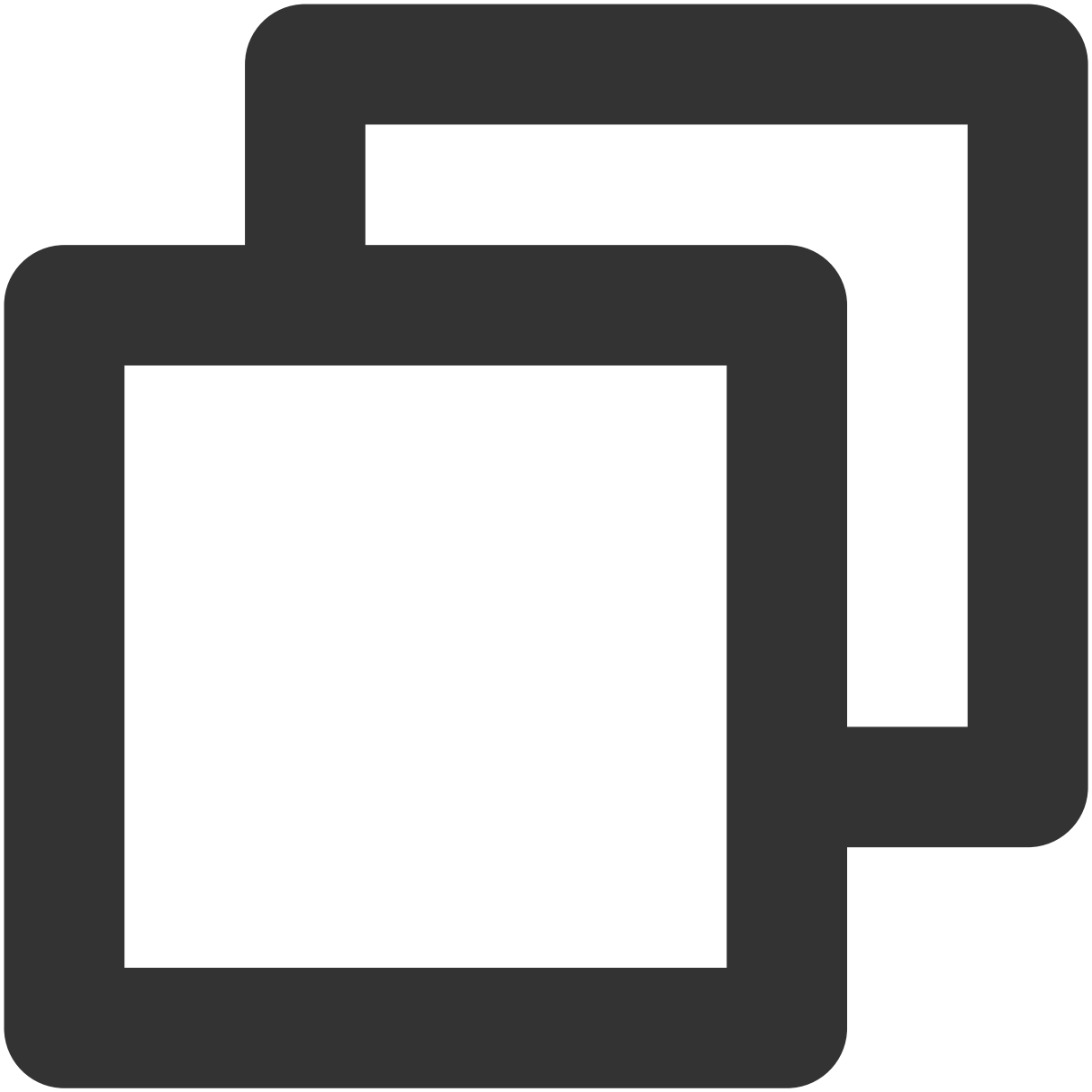
群组的创建详见：IM 群组管理，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Kotlin

Java



```
fun joinInGroupCall(roomId: RoomId?, groupId: String?, callMediaType: TUICallDefine
```



```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId, TUICallDefine.M
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

setCallingBell

设置自定义来电铃音。

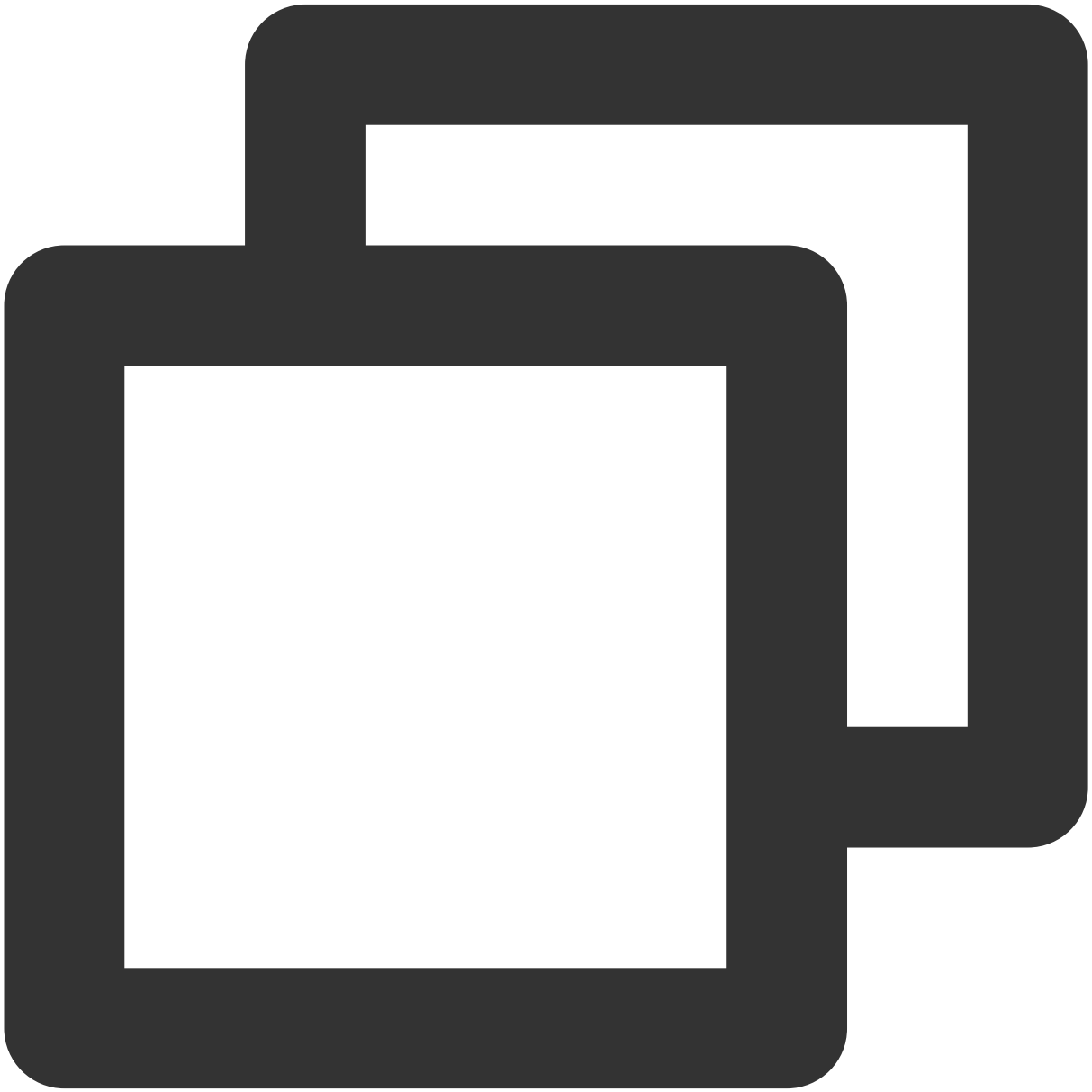
这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

铃声设置后与设备绑定，更换用户，铃声依旧会生效。

如需恢复默认铃声， `filePath` 传空即可。

Kotlin

Java



```
fun setCallingBell(filePath: String?)
```

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

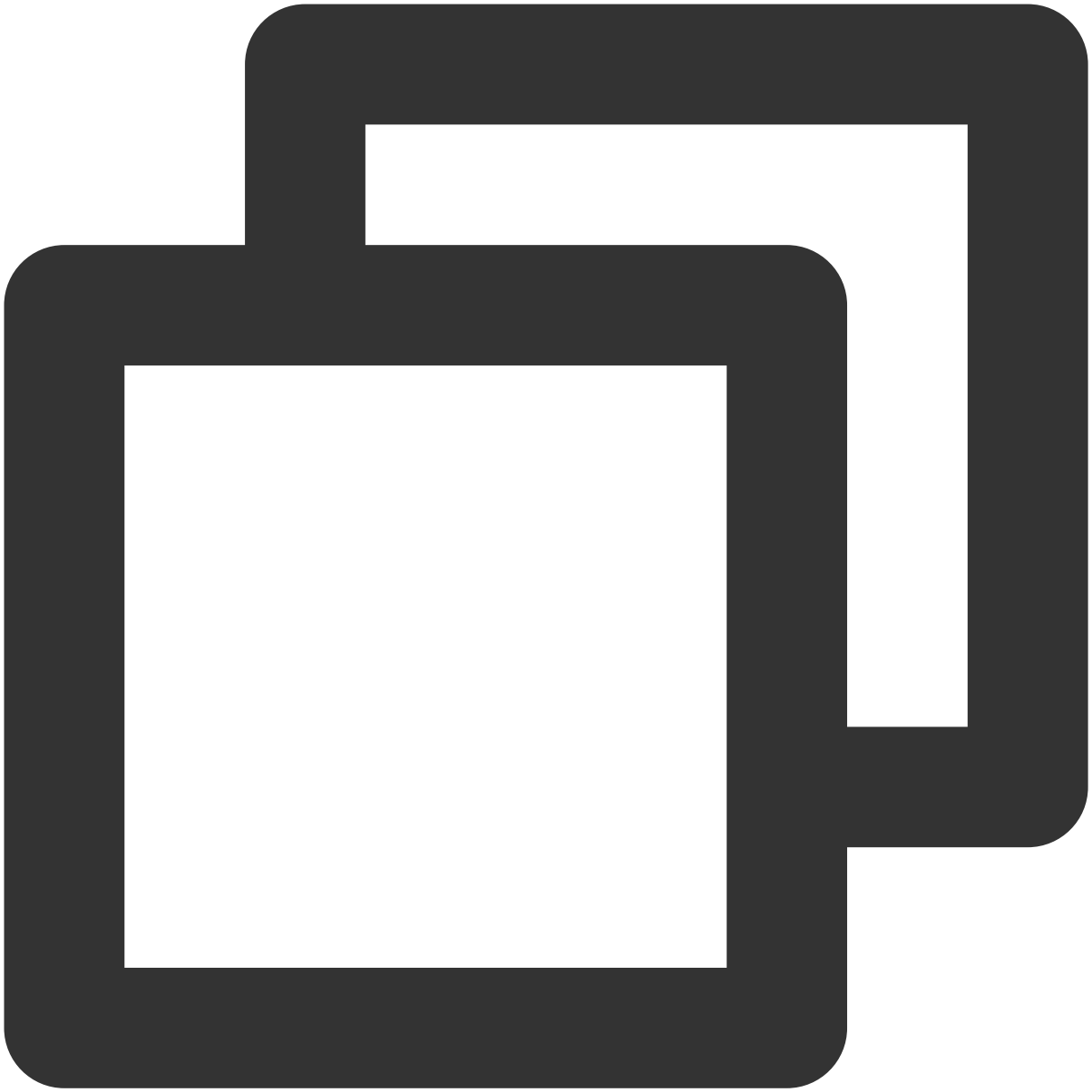
enableMuteMode

开启/关闭静音模式。

Kotlin

Java

```
fun enableMuteMode(enable: Boolean)
```



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

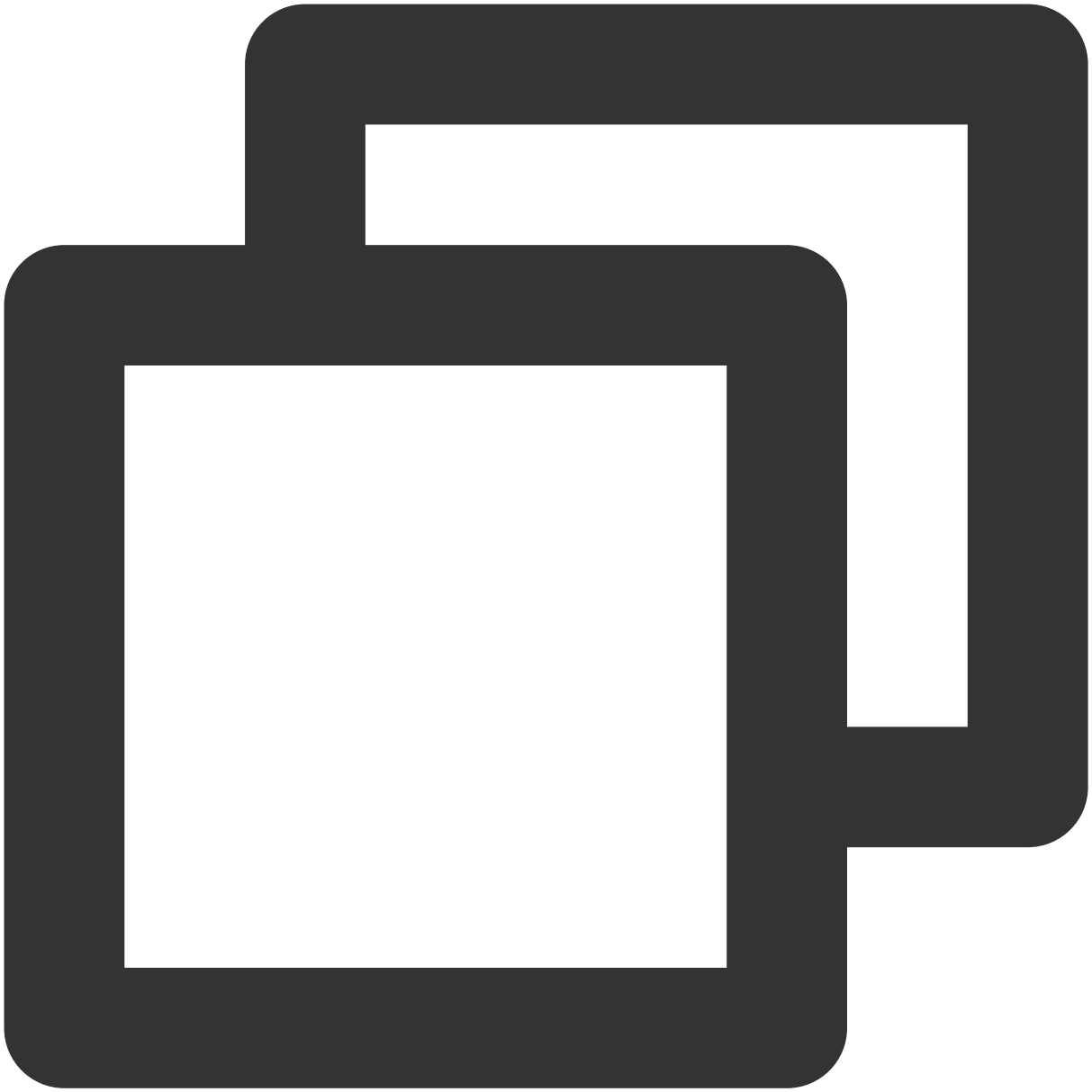
enableFloatWindow

开启/关闭悬浮窗功能。

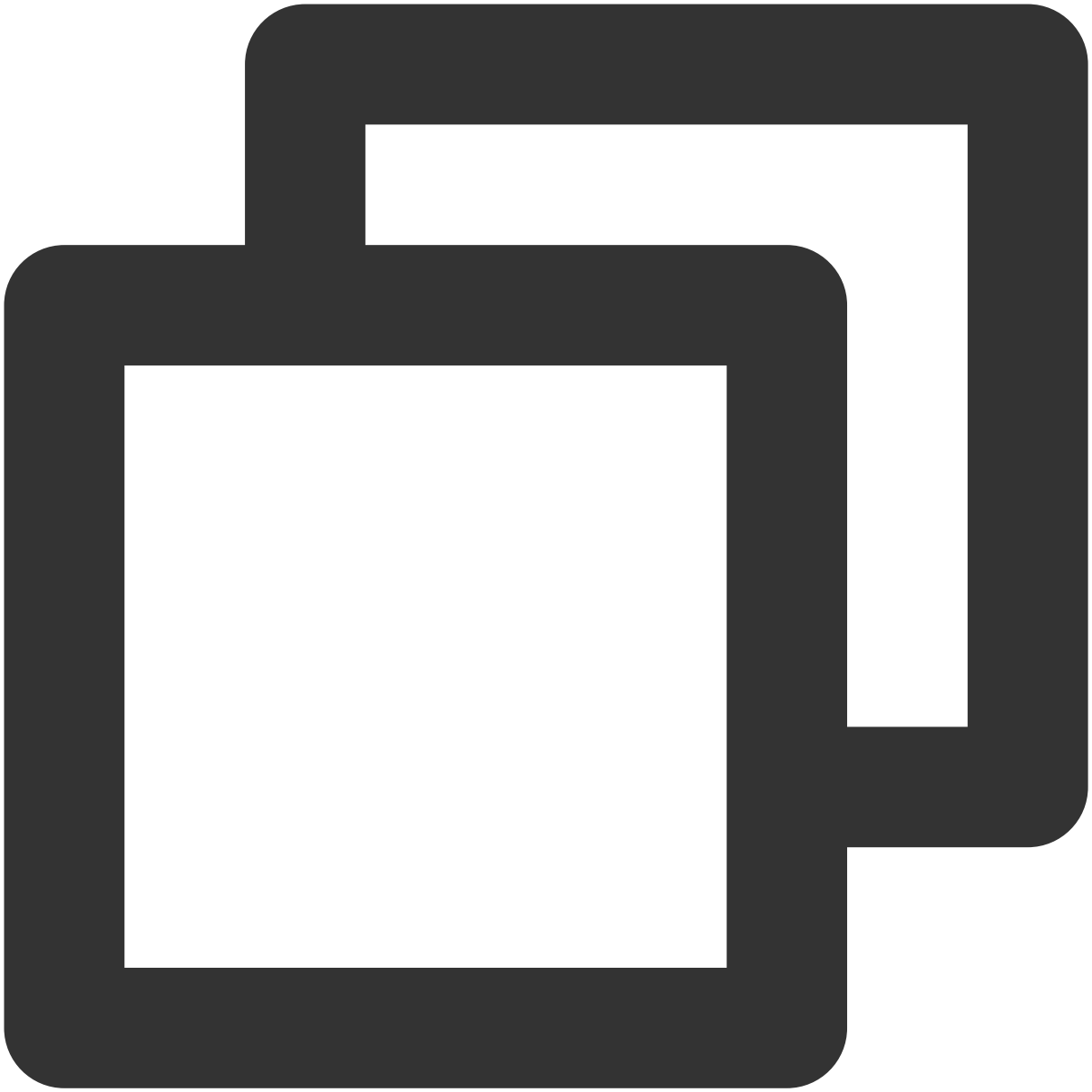
默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

Kotlin

Java



```
fun enableFloatWindow(enable: Boolean)
```

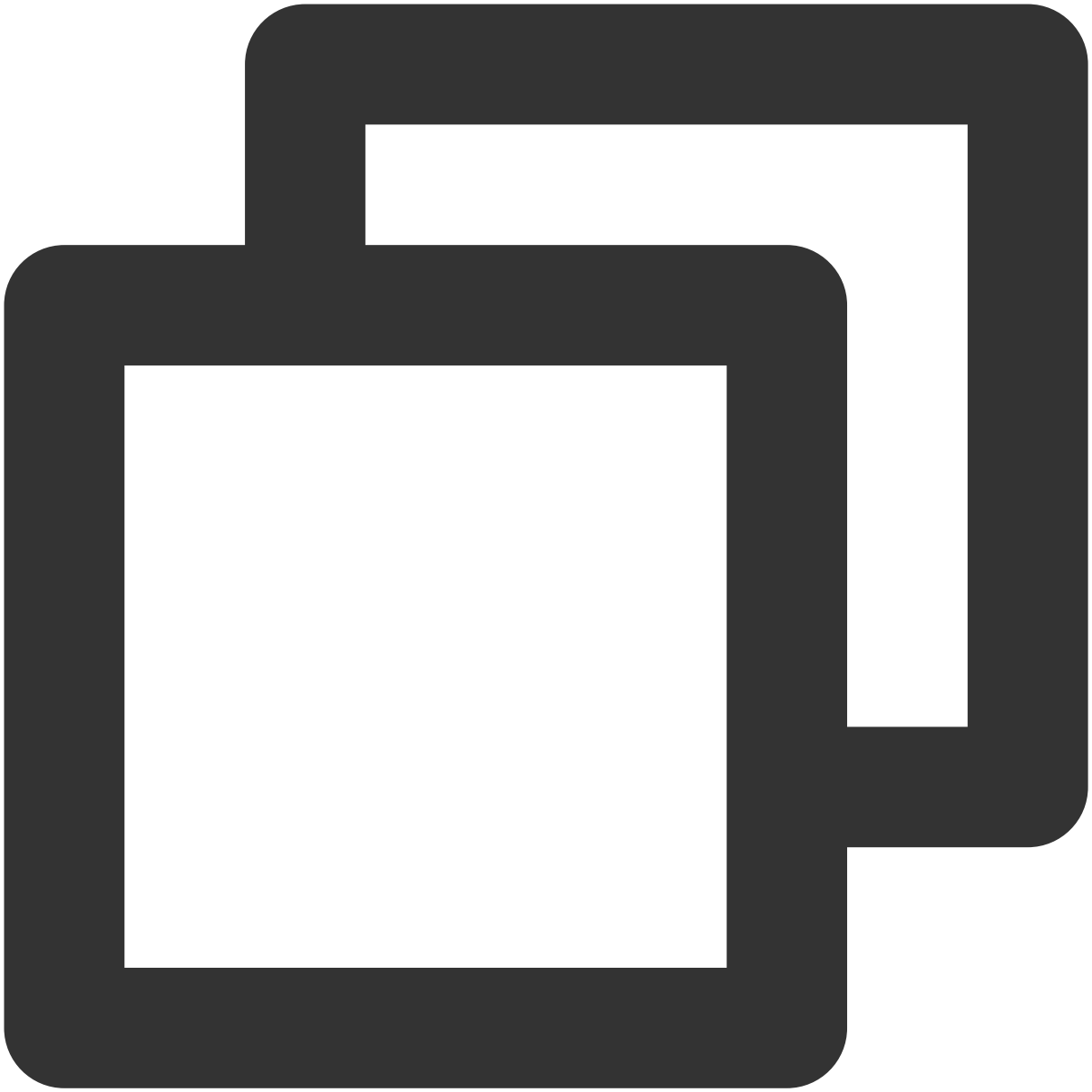


```
void enableFloatWindow(boolean enable);
```

enableIncomingBanner

开启/关闭来电横幅显示。

默认为 `false` ，被叫端收到邀请后默认弹出全屏通话等待界面，开启后先展示一个横幅，然后根据需要拉起全屏通话界面。



```
fun enableIncomingBanner(enable: Boolean)
```

TUICallEngine

最近更新时间：2024-07-30 11:21:12

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，例如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流

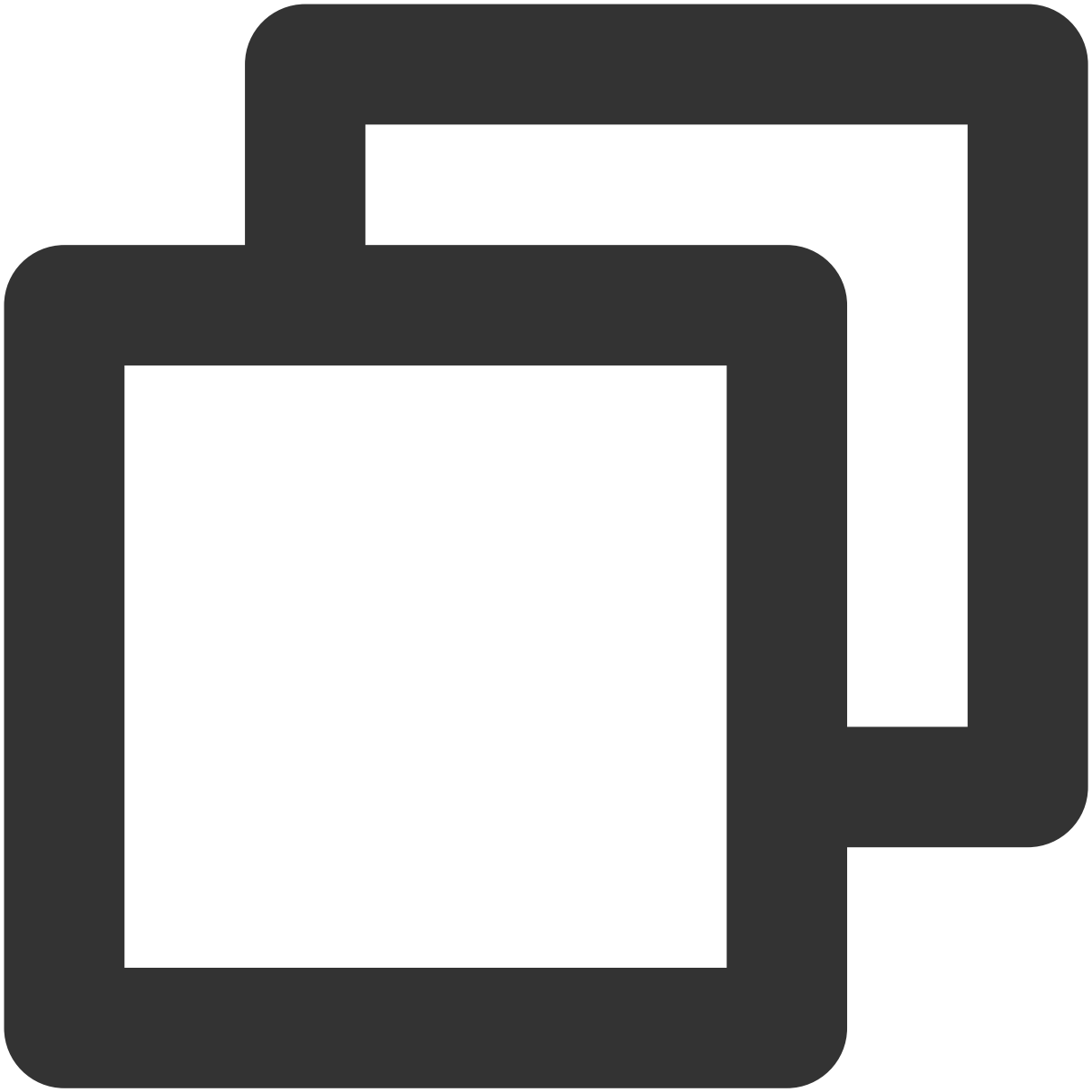
版权所有：腾讯云计算（北京）有限责任公司

openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（群组通话版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

API 详情

createInstance

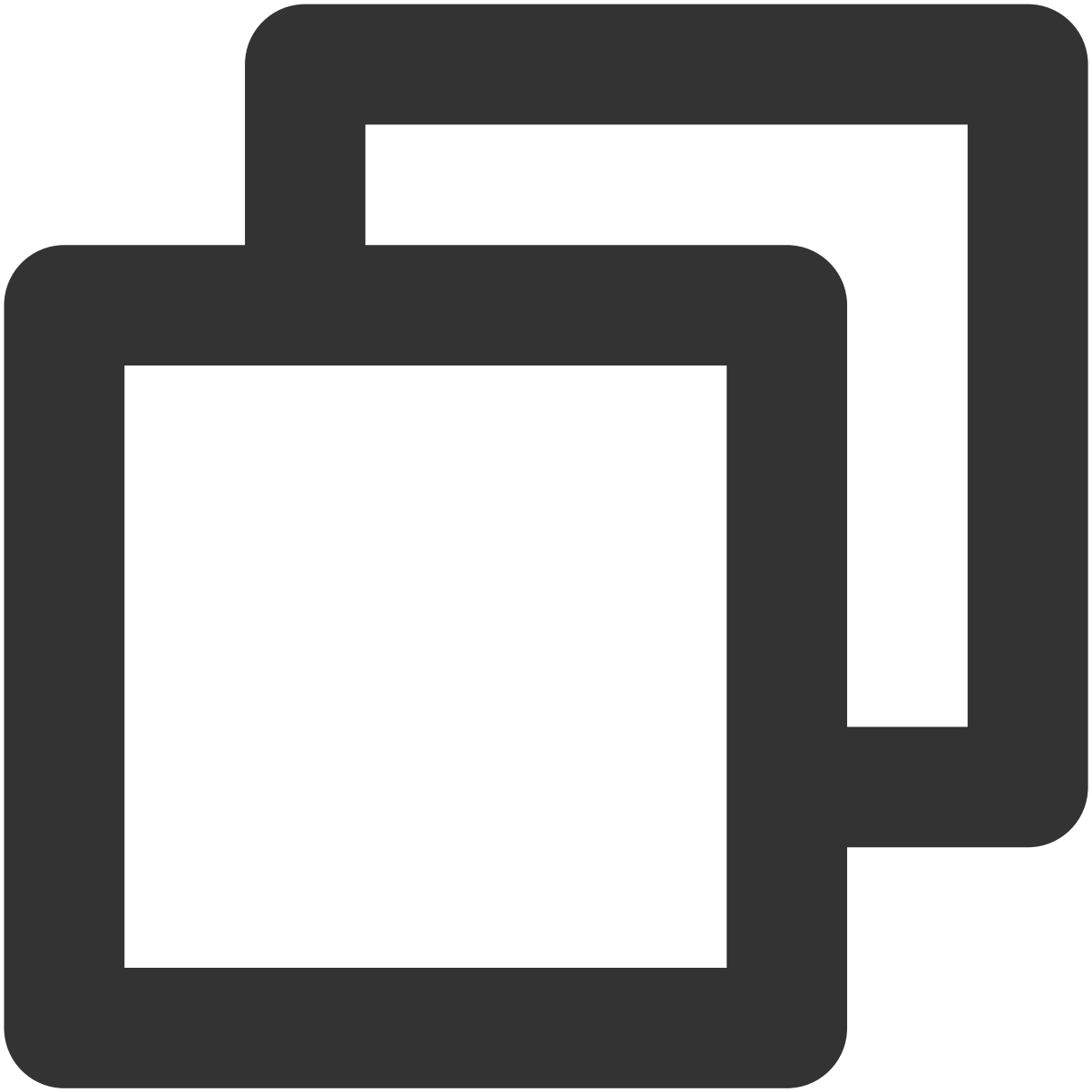
创建 TUICallEngine 的单例。



```
TUICallEngine createInstance(Context context);
```

destroyInstance

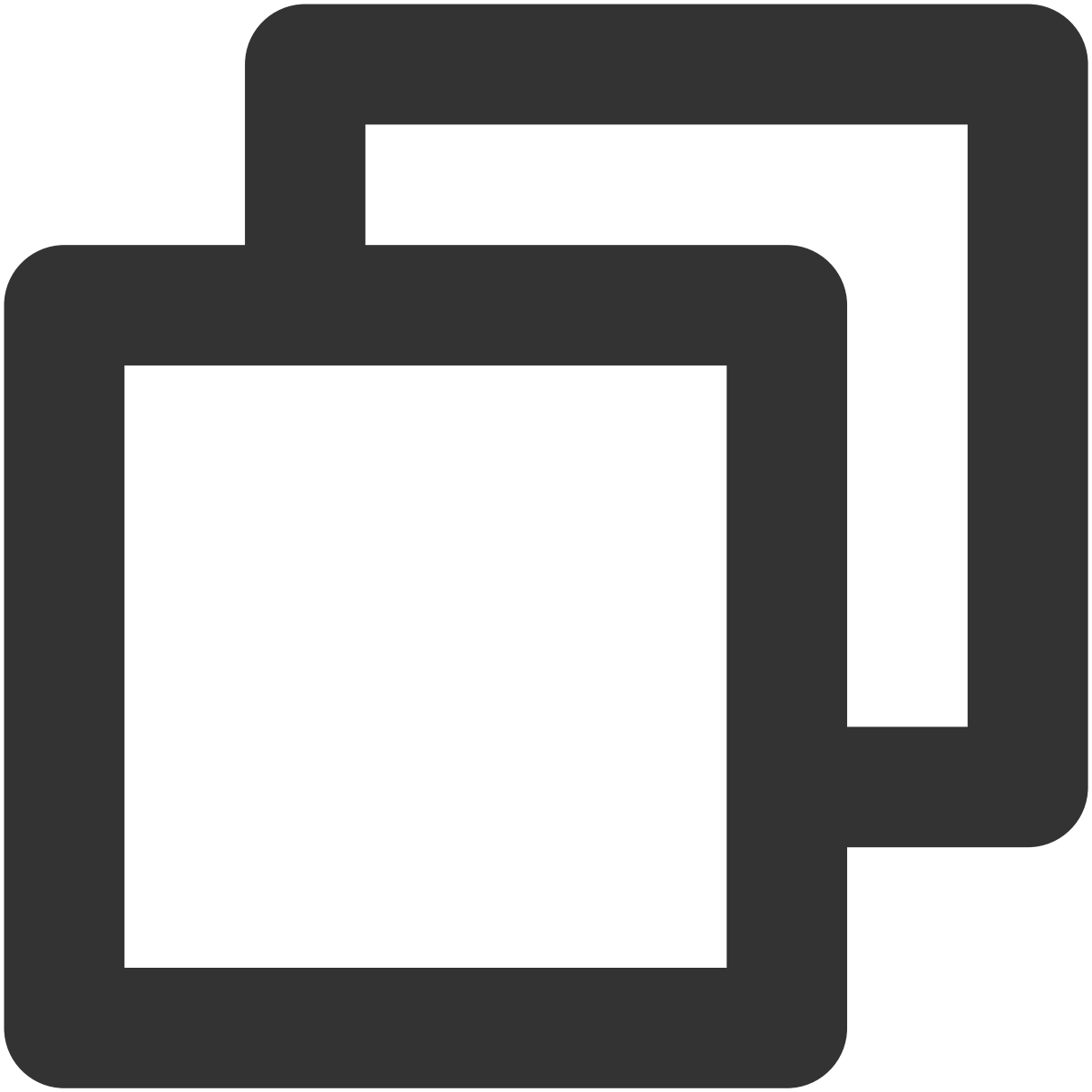
销毁 TUICallEngine 的单例。



```
void destroyInstance();
```

init

初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。



```
void init(int sdkAppId, String userId, String userSig, TUICommonDefine.Callback cal
```

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	TUICommonDefine.Callback	初始化回调， `onSuccess` 表示初始化成功

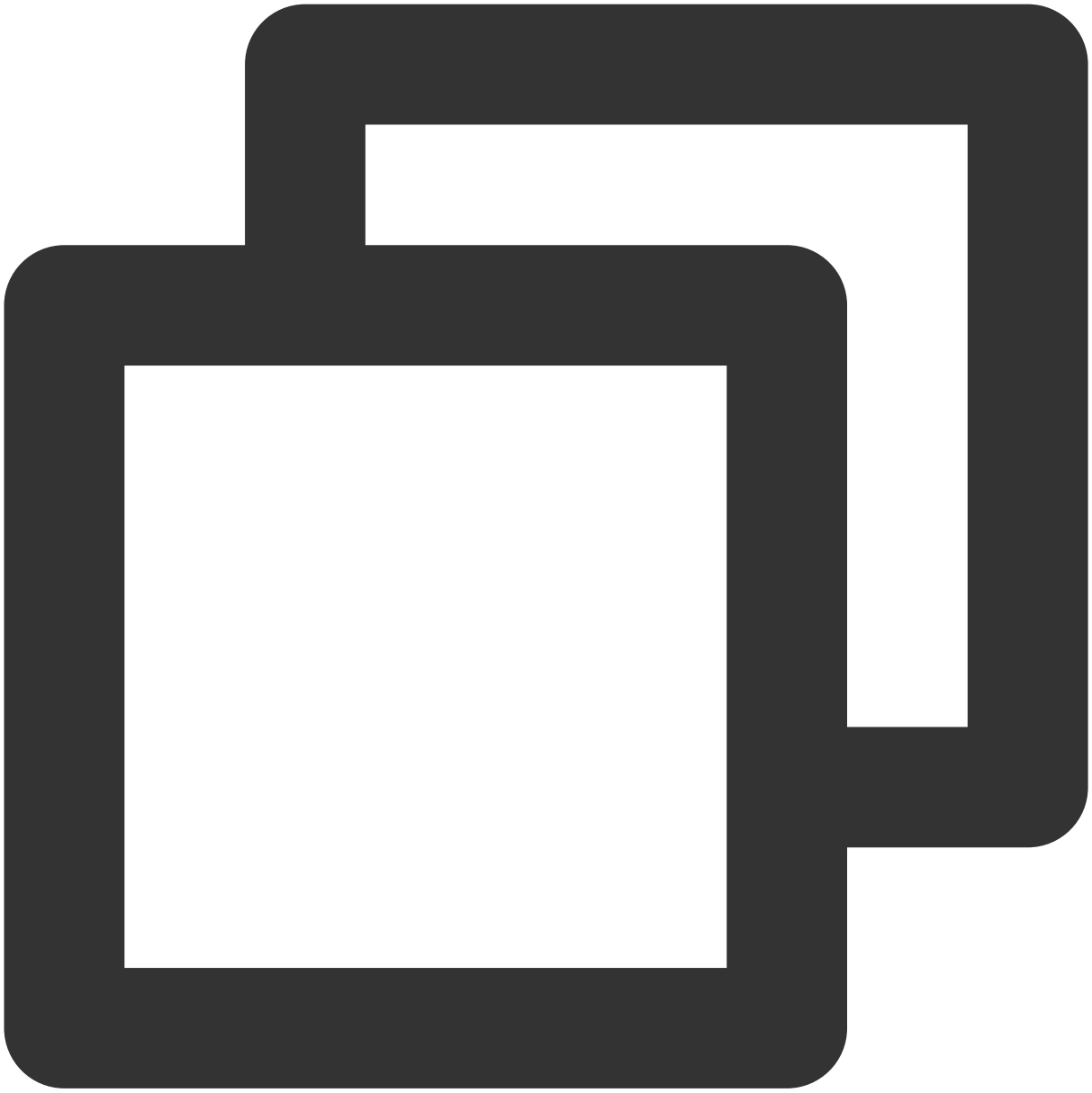
addObserver

添加回调接口，您可以通过这个接口，监听 `TUICallObserver` 相关的事件回调。

```
void addObserver(TUICallObserver observer);
```

removeObserver

移除回调接口。



```
void removeObserver(TUICallObserver observer);
```

call

拨打电话（1v1通话）。

```
void call(String userId, TUICallDefine.MediaType callMediaType,
          TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如：视频通话、语音通话

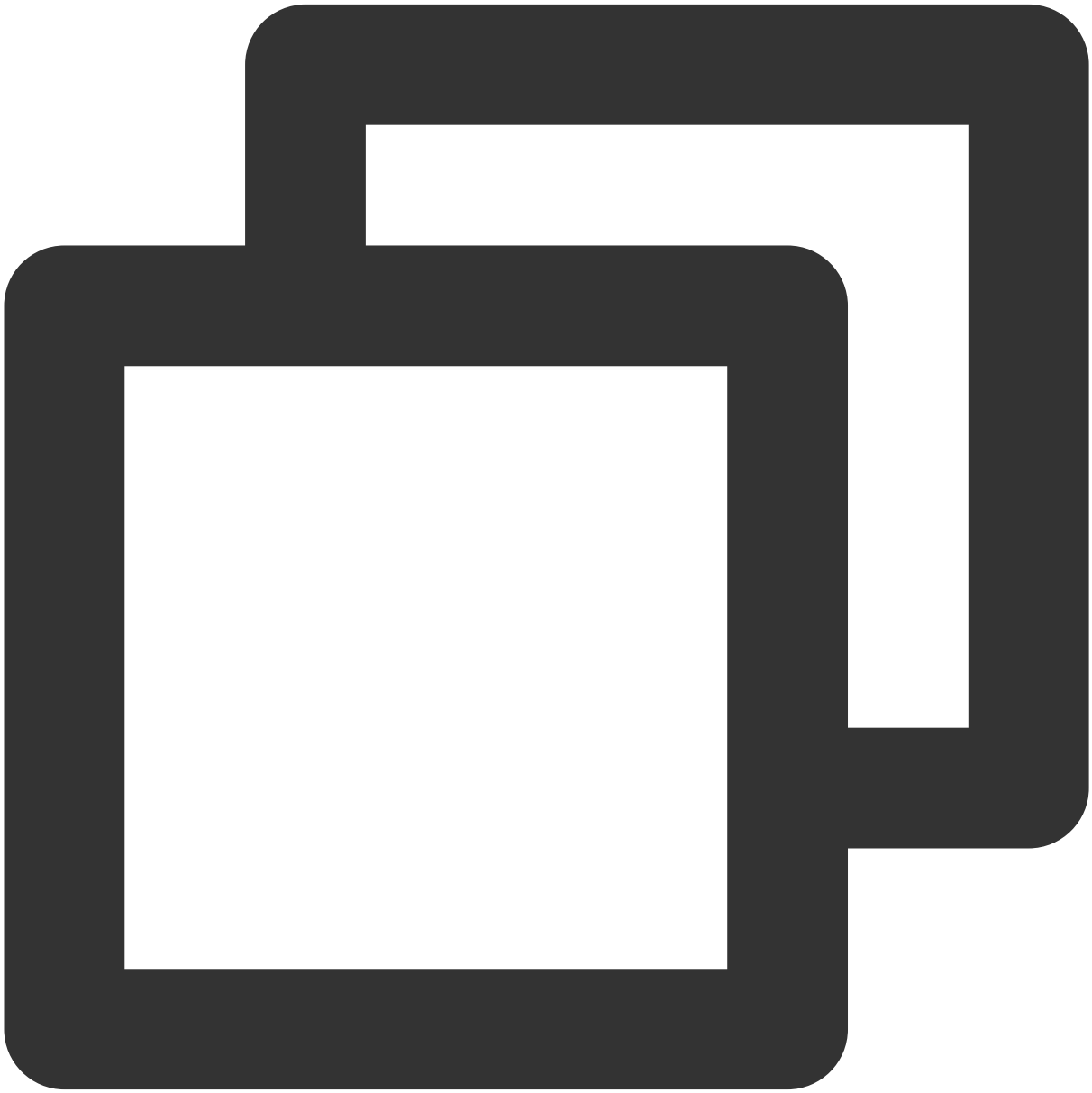
params	TUICallDefine.CallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话。

注意：

使用群组通话前需要创建IM 群组，如果已经创建，请忽略。



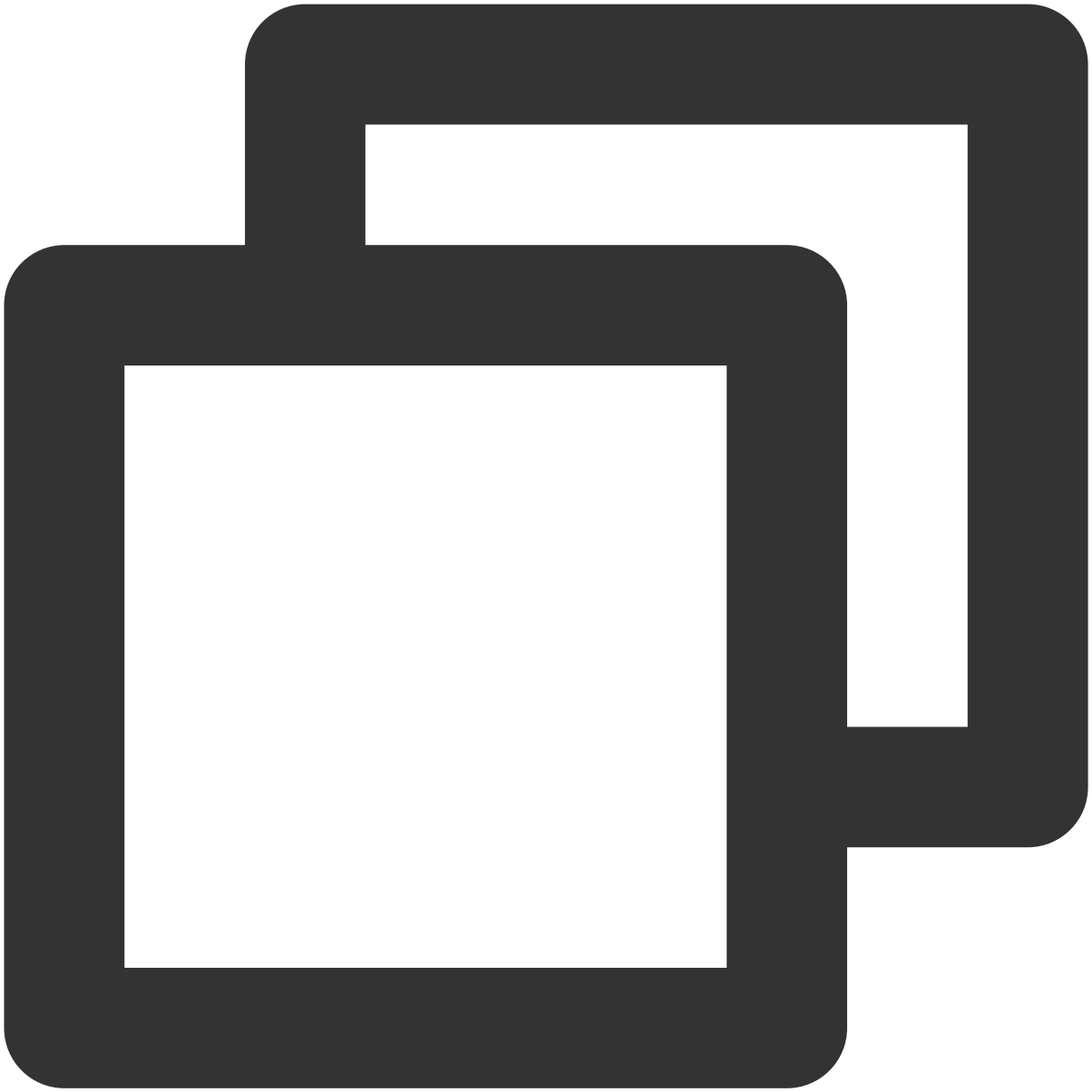
```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType cal
               TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如：视频通话、语音通话
params	TUICallDefine.CallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

accept

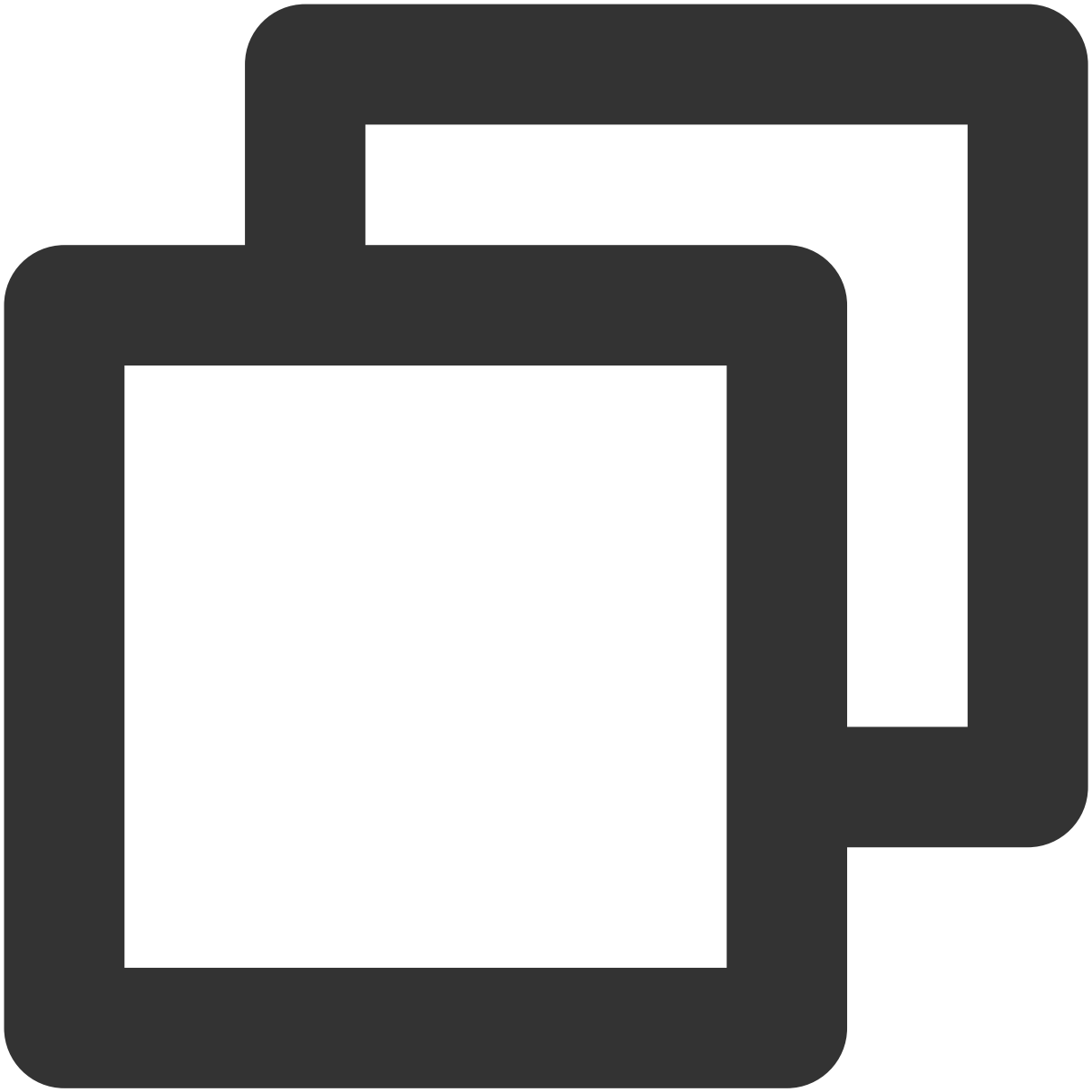
接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。



```
void accept(TUICommonDefine.Callback callback);
```

reject

拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。

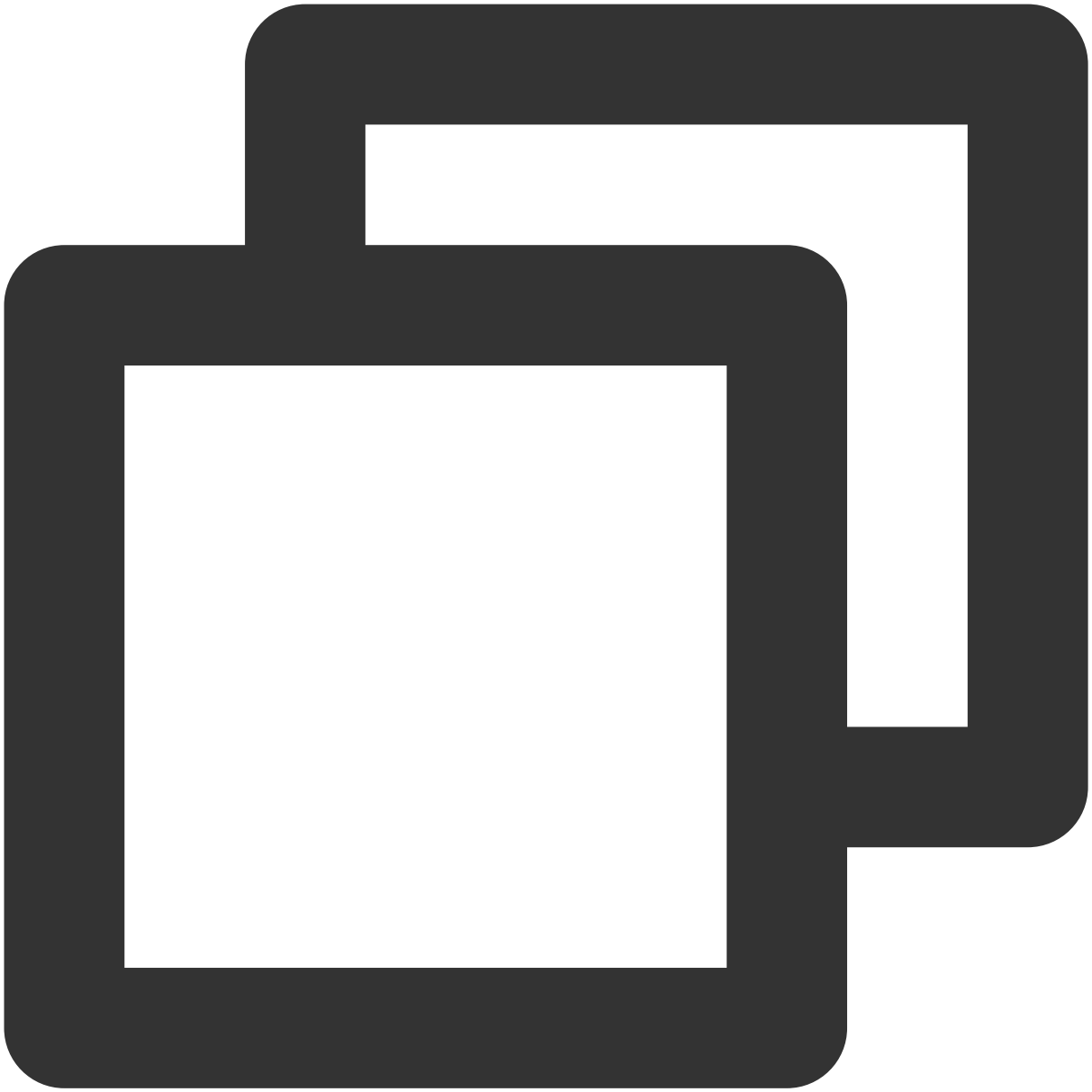


```
void reject(TUICommonDefine.Callback callback);
```

ignore

忽略当前通话，当您作为被叫收到 onCallReceived() 的回调时，可以调用该函数忽略来电，此时主叫会收到 onUserLineBusy 的回调。

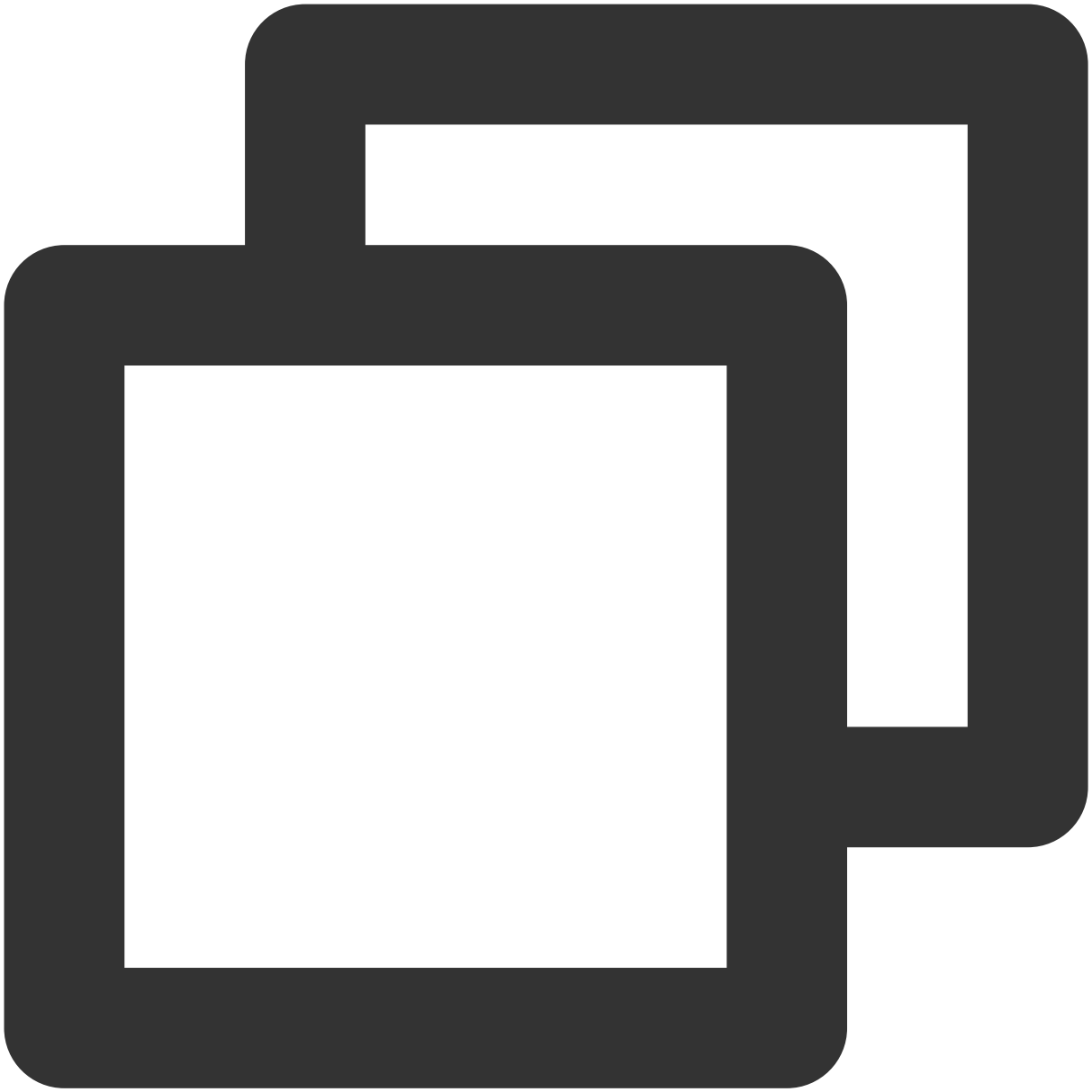
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。



```
void ignore(TUICommonDefine.Callback callback);
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

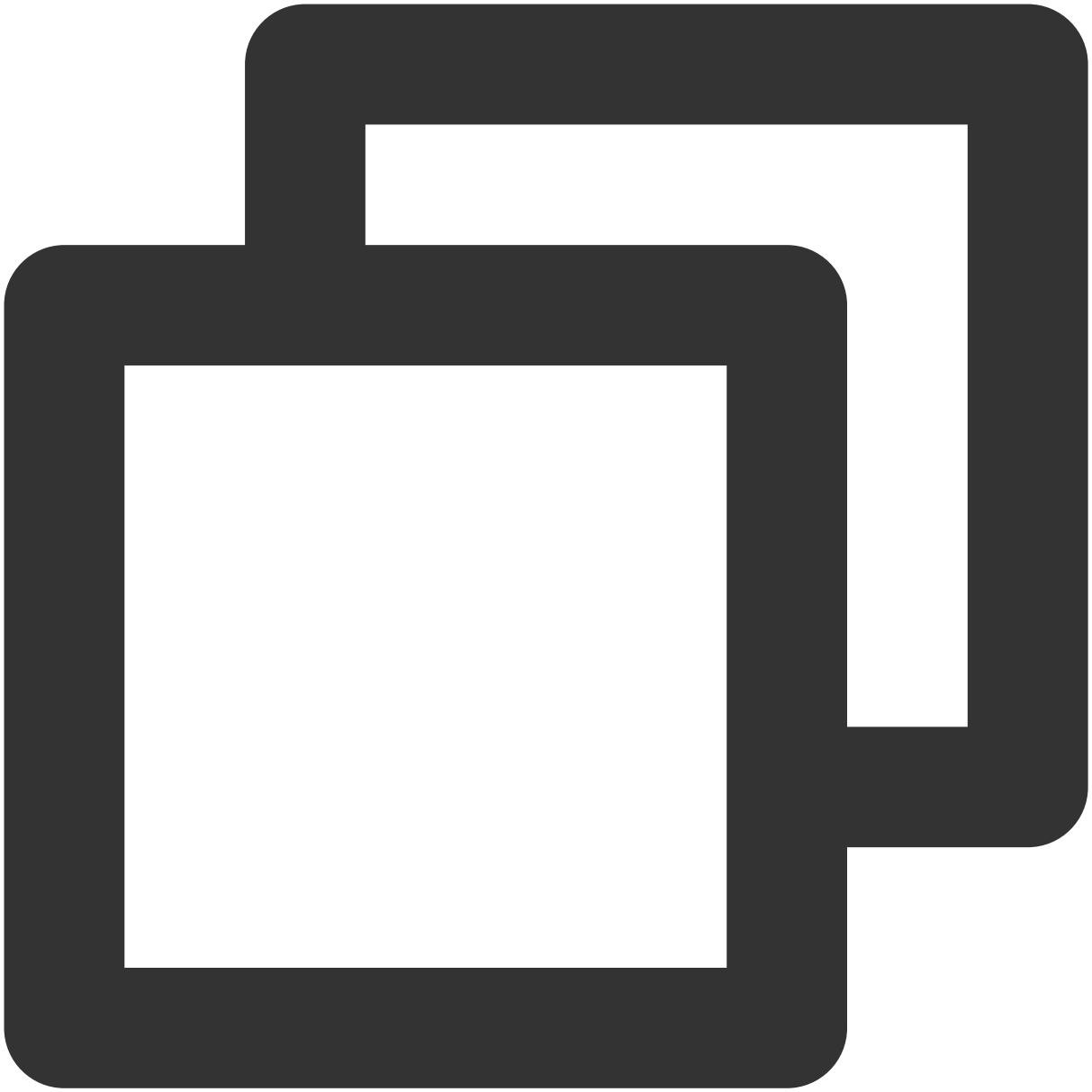


```
void hangup(TUICommonDefine.Callback callback);
```

inviteUser

邀请用户加入此次群组通话。

使用场景：一个群组通话中的用户主动邀请其他人时使用。



```
void inviteUser(List<String> userIdList, TUICallDefine.CallParams params,
                TUICommonDefine.ValueCallback callback);
```

参数如下表所示：

参数	类型	含义
userIdList	List	目标用户的 userId 列表
params	TUICallDefine.CallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

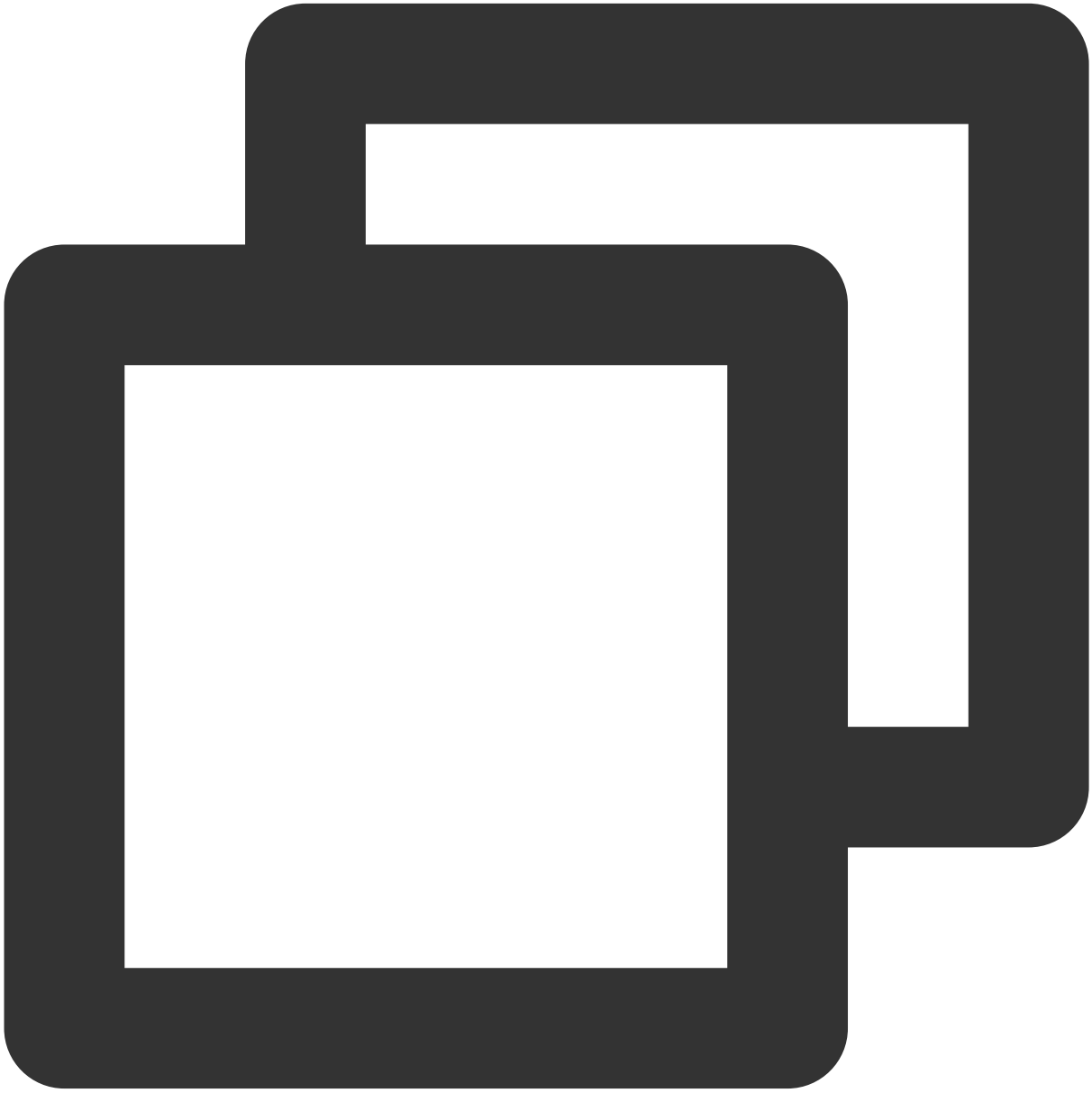
注意：

邀请他人进入当前通话，该场景下，自定义RoomId无效，SDK会邀请他人加入当前邀请者所在的房间。

joinInGroupCall

主动加入此次群组通话。

使用场景：群组内用户主动加入此次群组通话使用。



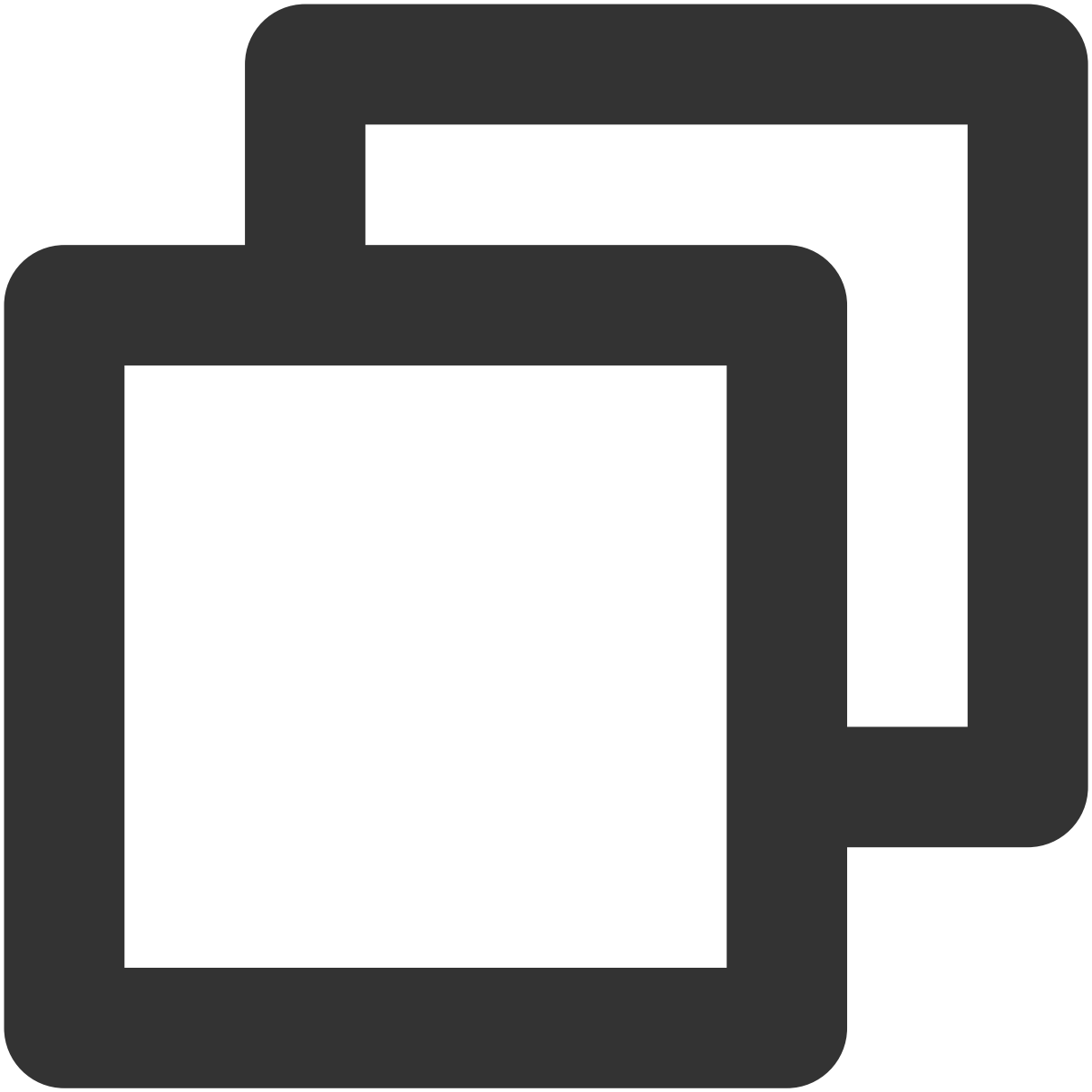
```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId,
                     TUICallDefine.MediaType callMediaType, TUICommonDefine.Callbac
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如视频通话、语音通话

switchCallMediaType

切换视频通话到语音通话。



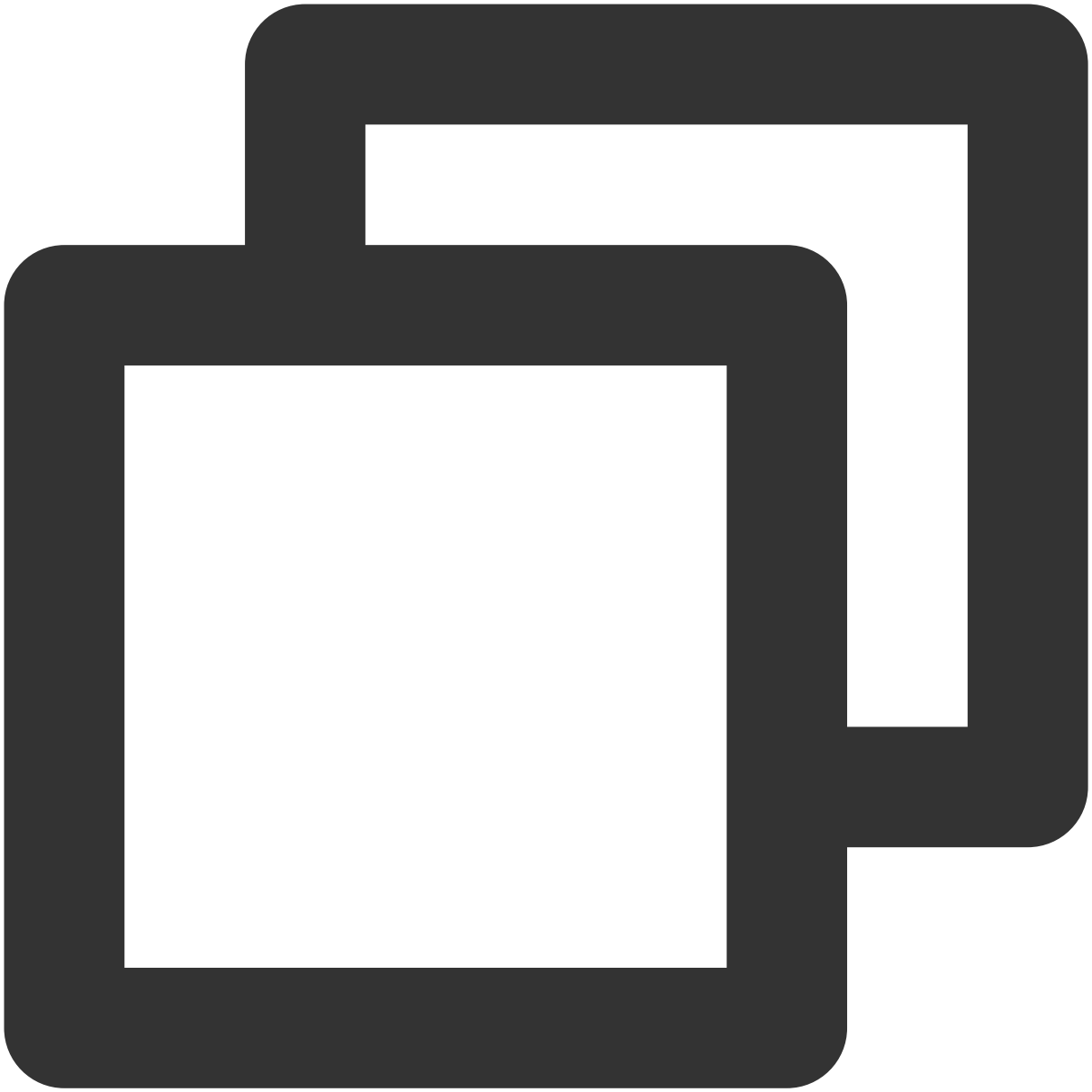
```
void switchCallMediaType(TUICallDefine.MediaType callMediaType);
```

参数如下表所示：

参数	类型	含义
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如视频通话、语音通话

startRemoteView

开始订阅远端用户的视频数据。



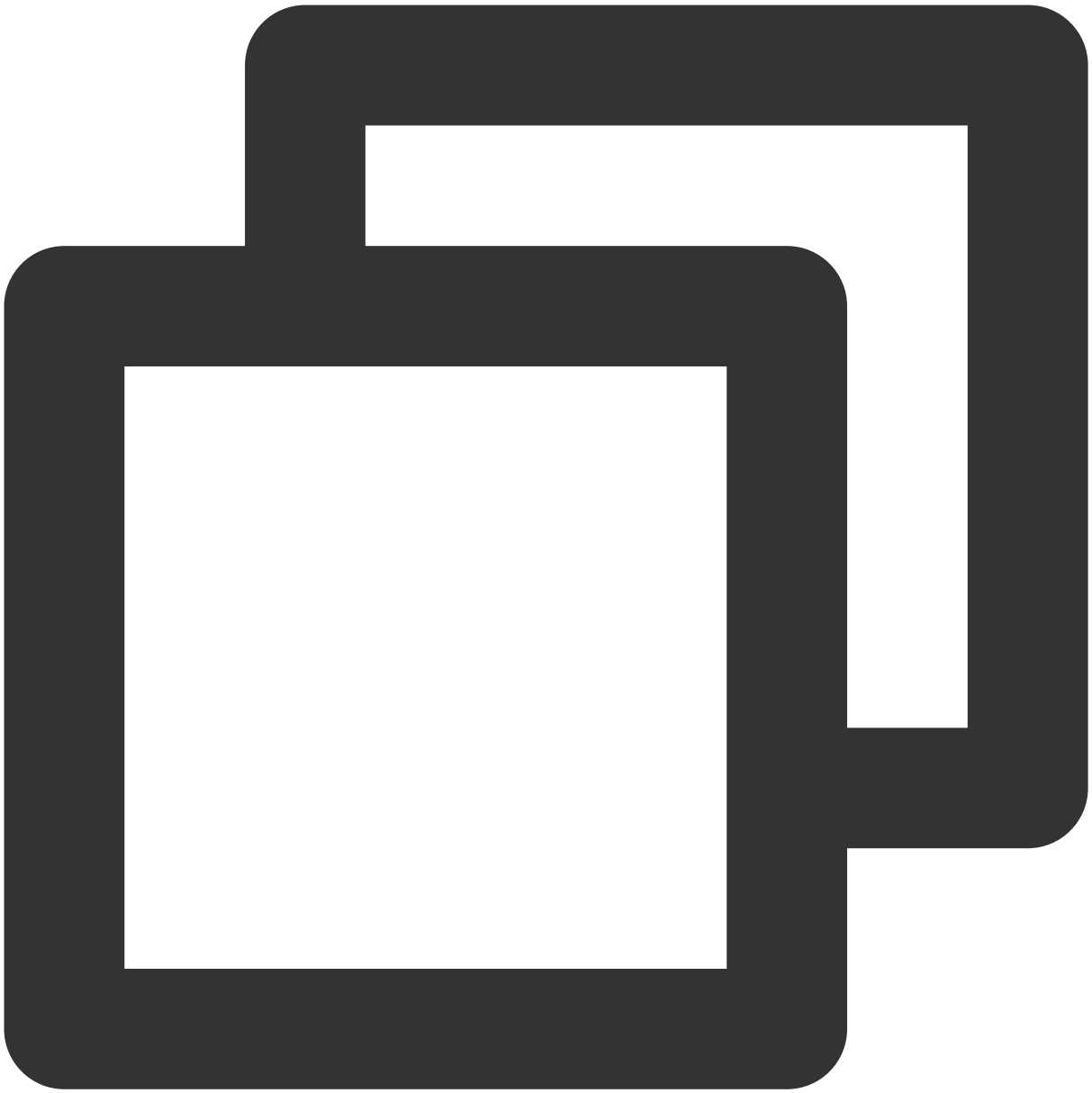
```
void startRemoteView(String userId, TUIVideoView videoView, TUICommonDefine.PlayCal
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
videoView	TUIVideoView	待渲染的视图

stopRemoteView

停止订阅远端用户的视频数据。



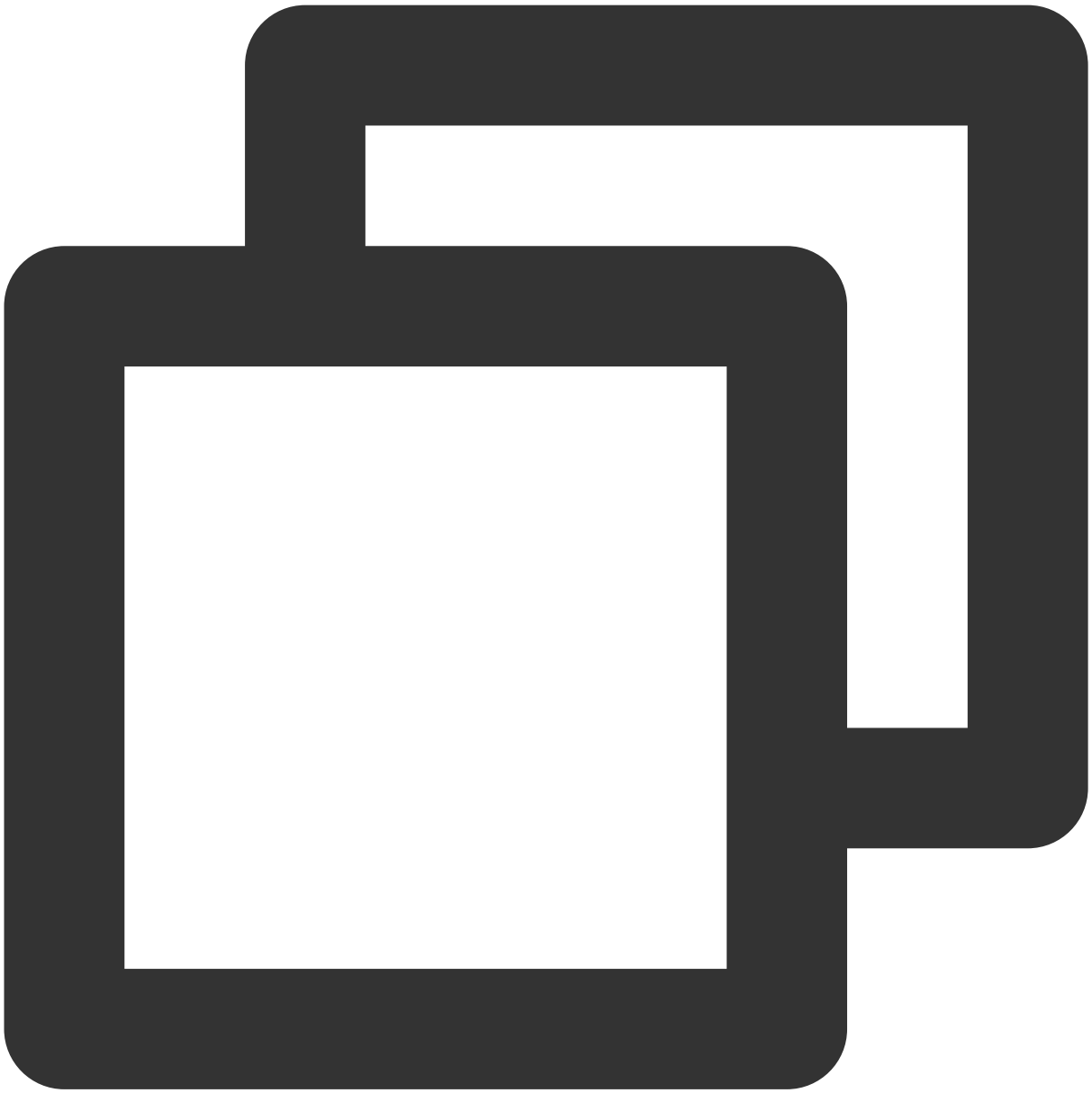
```
void stopRemoteView(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId

openCamera

开启摄像头。



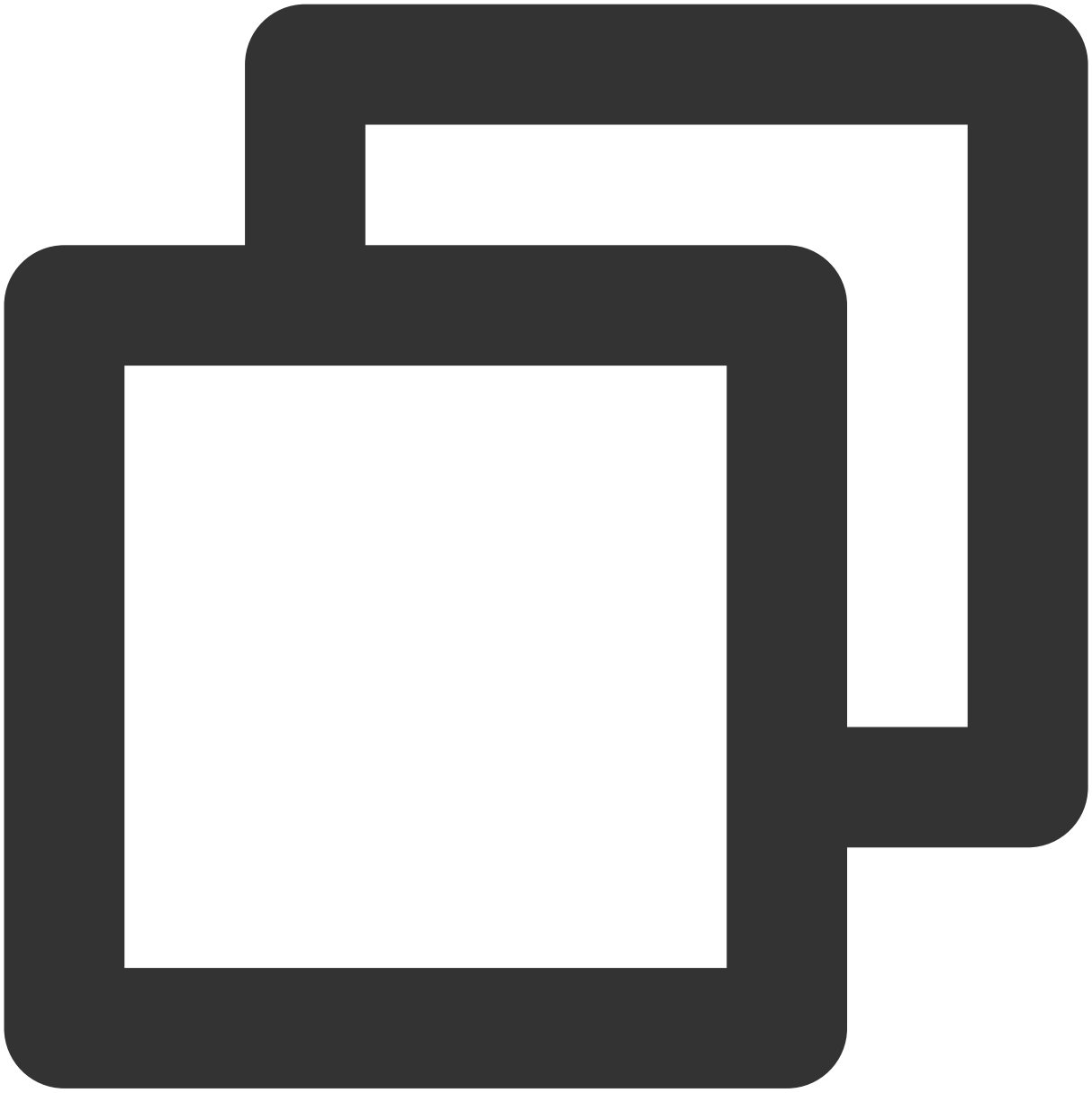
```
void openCamera(TUICommonDefine.Camera camera, TUIVideoView videoView, TUICommonDef
```

参数如下表所示：

参数	类型	含义
camera	TUICommonDefine.Camera	前置/后置 摄像头
videoView	TUIVideoView	待渲染的视图

closeCamera

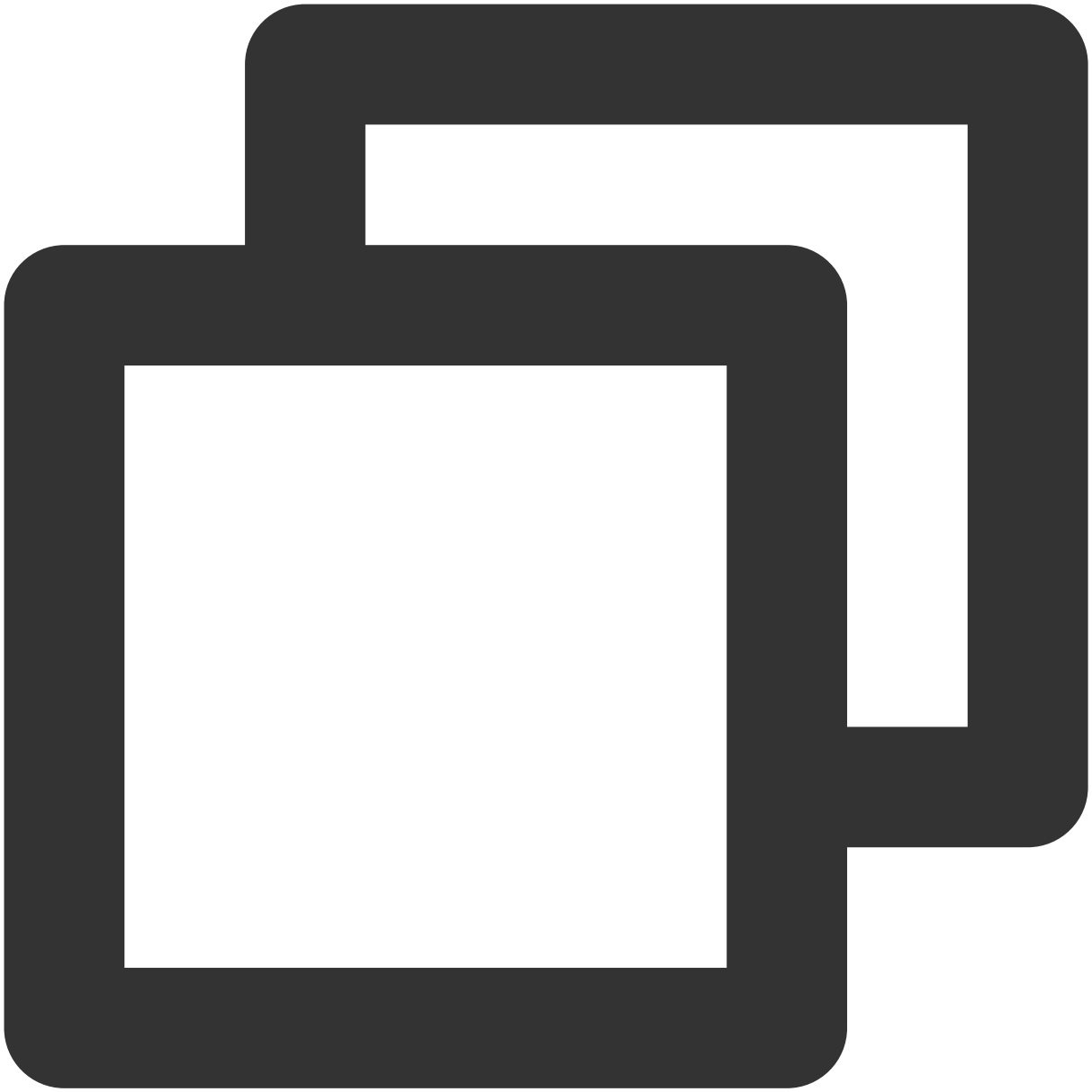
关闭摄像头。



```
void closeCamera();
```

switchCamera

切换前后摄像头。



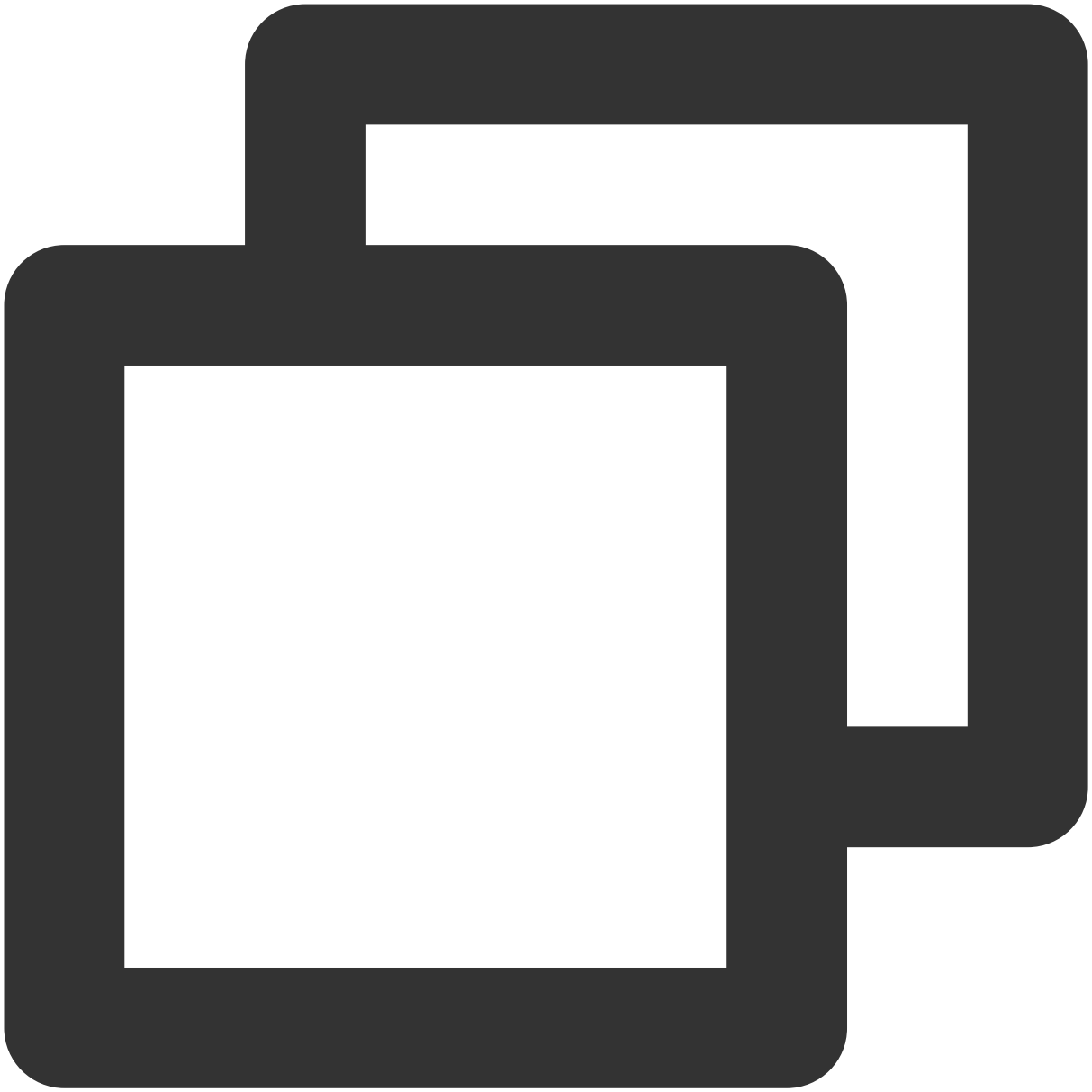
```
void switchCamera(TUICommonDefine.Camera camera);
```

参数如下表所示：

参数	类型	含义
camera	TUICommonDefine.Camera	前置/后置 摄像头

openMicrophone

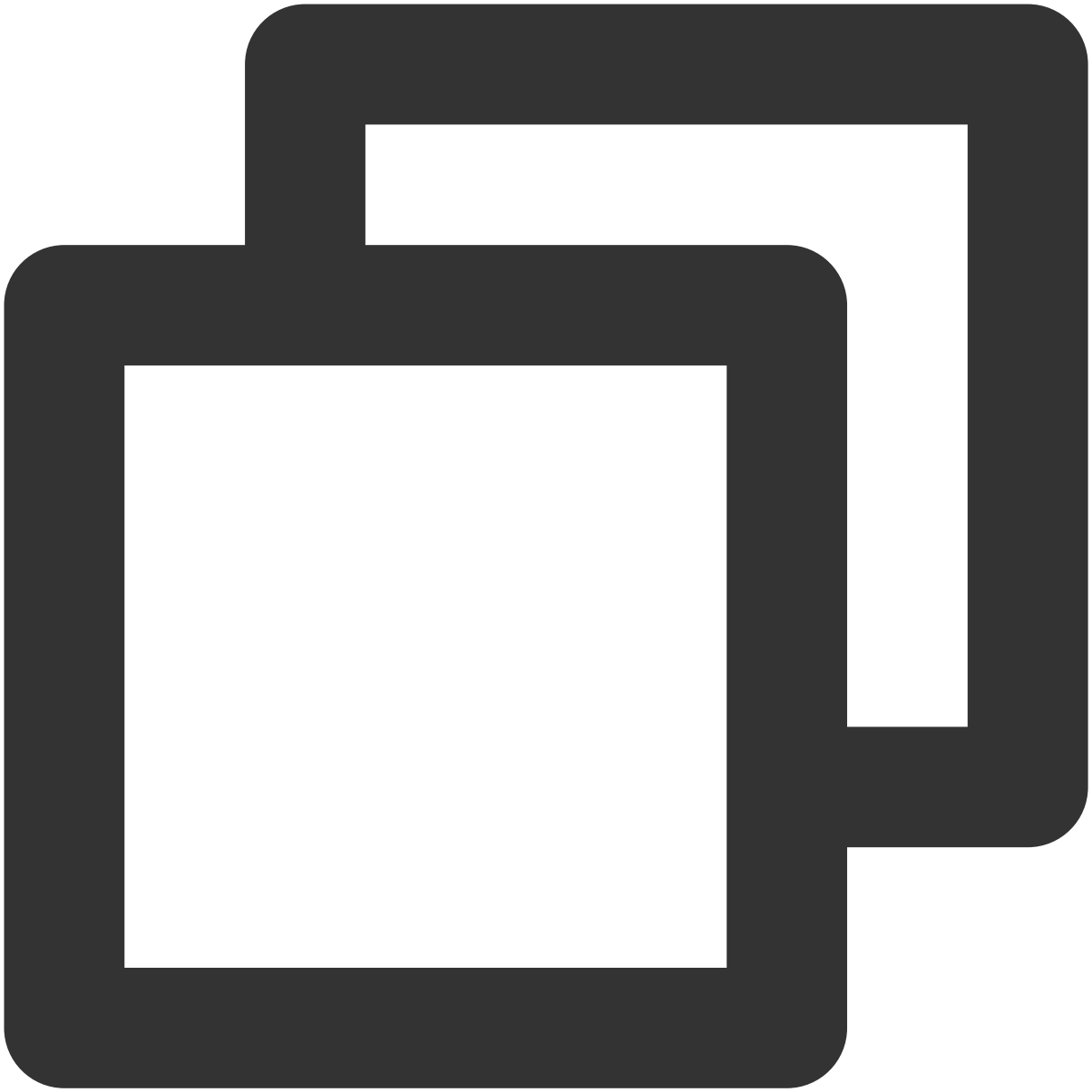
打开麦克风。



```
void openMicrophone(TUICommonDefine.Callback callback);
```

closeMicrophone

关闭麦克风。



```
void closeMicrophone();
```

selectAudioPlaybackDevice

选择音频播放设备。

目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。

```
void selectAudioPlaybackDevice(TUICommonDefine.AudioPlaybackDevice device);
```

参数如下表所示：

参数	类型	含义
device	TUICommonDefine.AudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。

用户昵称不能超过500字节，用户头像必须是 URL 格式。

```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

参数如下表所示：

参数	类型	含义
nickname	String	用户昵称
avatar	String	用户头像（格式为 URL）

enableMultiDeviceAbility

开启/关闭 TUICallEngine 的多设备登录模式（群组通话版套餐支持）。

```
void enableMultiDeviceAbility(boolean enable, TUICommonDefine.Callback callback);
```

setVideoRenderParams

设置用户视频画面的渲染模式。

```
void setVideoRenderParams(String userId, TUICommonDefine.VideoRenderParams params,
```

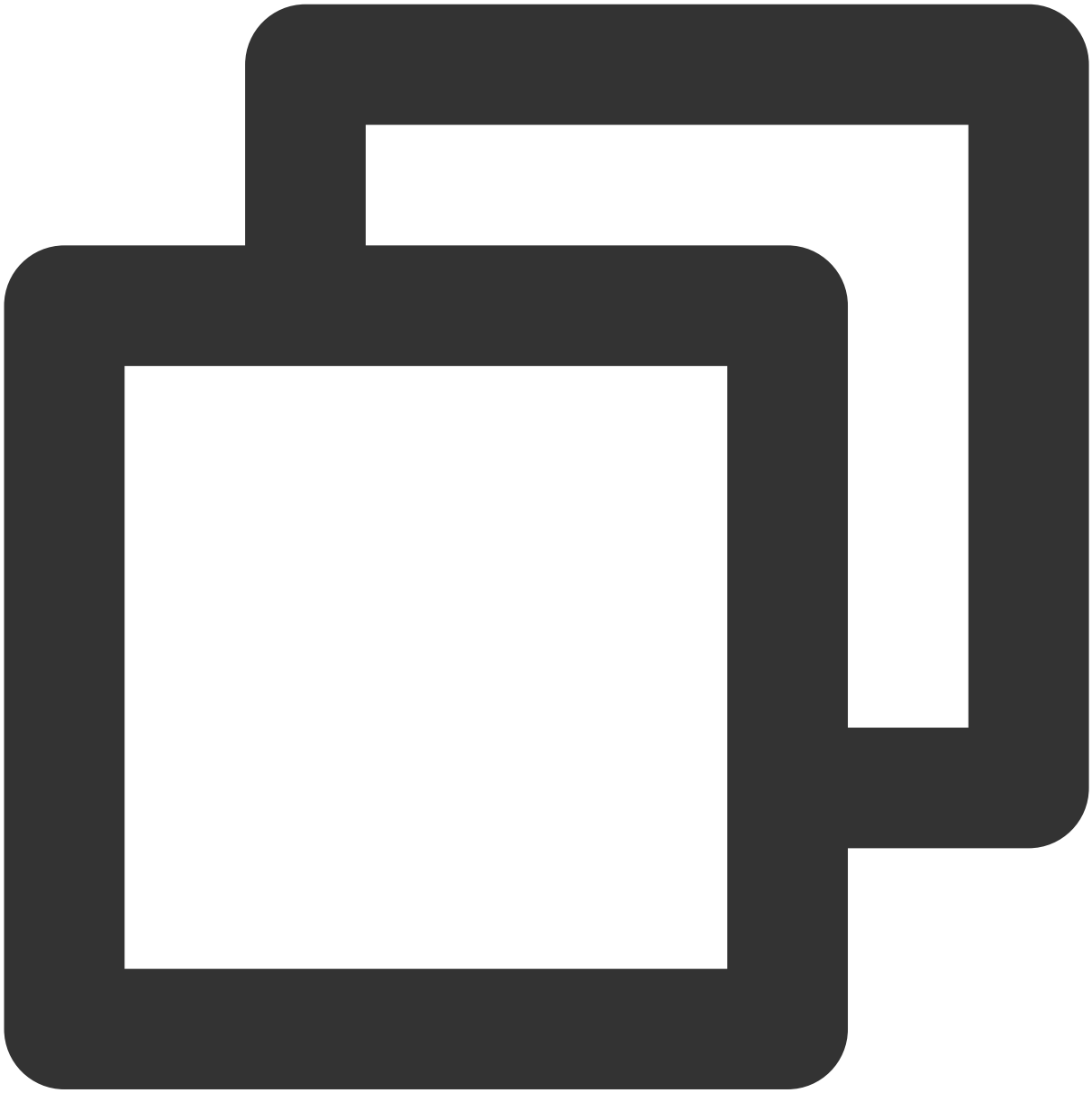
参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
params	TUICommonDefine.VideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。



```
void setVideoEncoderParams(TUICommonDefine.VideoEncoderParams params, TUICommonDefi
```

参数如下表所示：

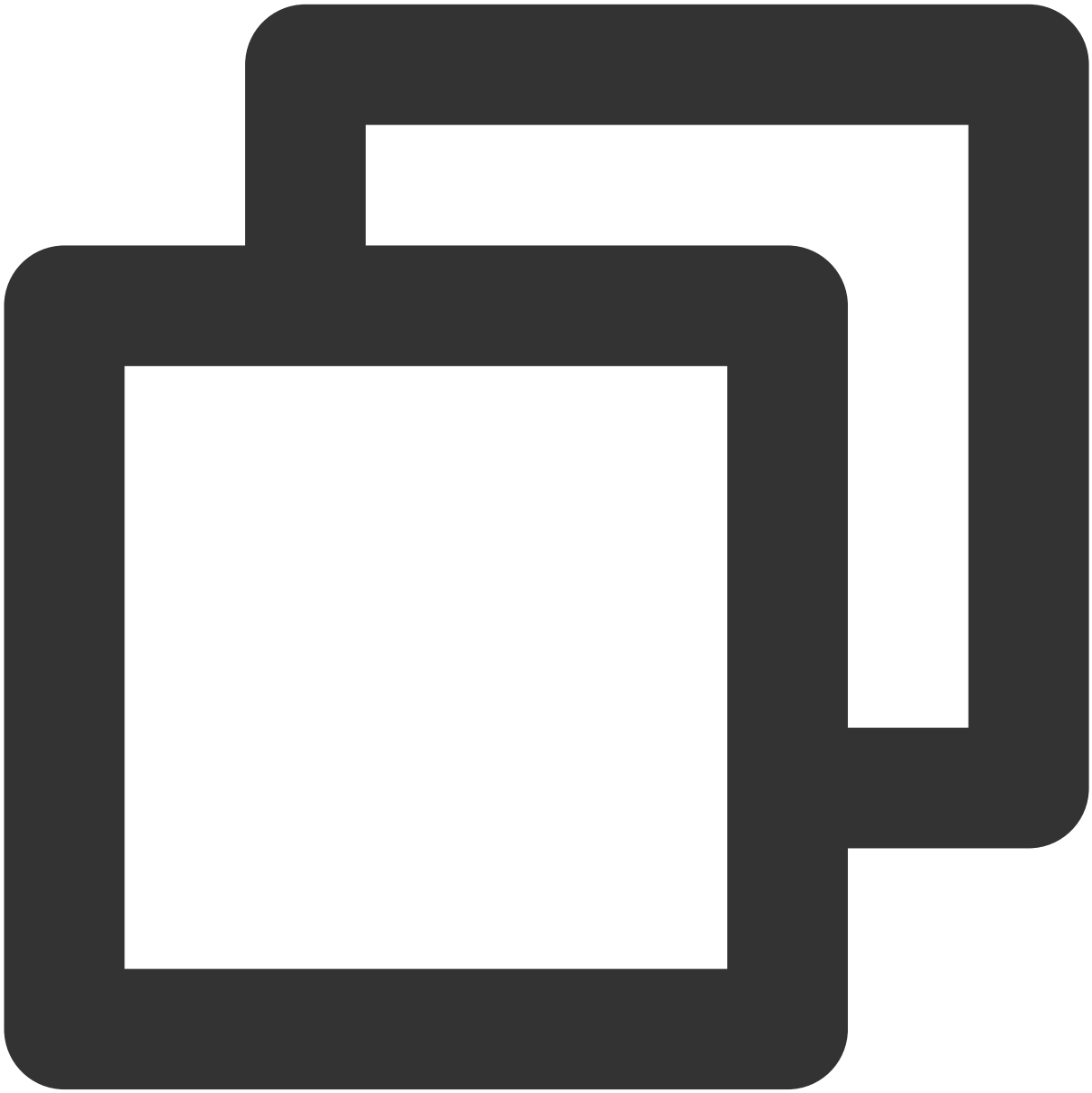
参数	类型	含义
params	TUICommonDefine.VideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高

		比

getTRTCCloudInstance

高级接口。

用于接入第三方美颜等。



```
TRTCCloud getTRTCCloudInstance();
```

setBeautyLevel

设置美颜级别，支持关闭默认美颜。

```
void setBeautyLevel(float level, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
level	float	美颜级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

TUICallObserver

最近更新时间：2024-01-25 14:24:20

TUICallObserver API 简介

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线

版权所有：腾讯云计算（北京）有限责任公司

onUserSigExpired	在线时票据过期

回调事件详情

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。

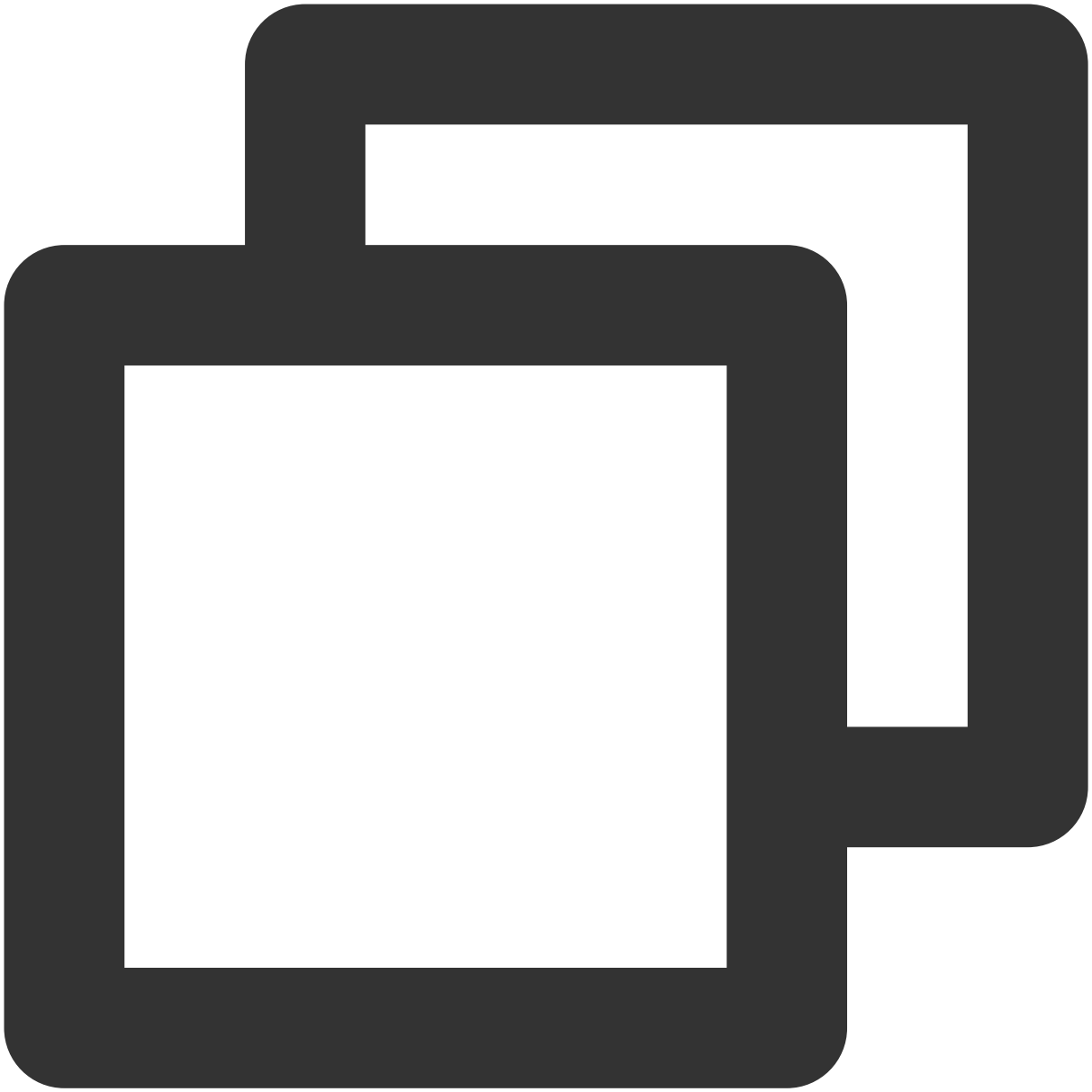
```
void onError(int code, String msg);
```

参数如下表所示：

参数	类型	含义
code	int	错误码
msg	String	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。



```
void onCallReceived(String callerId, List<String> calleeIdList, String groupId,
                    TUICallDefine.MediaType callMediaType, String userData);
```

参数如下表所示：

参数	类型	含义
callerId	String	主叫 ID（邀请方）
calleeIdList	List	被叫 ID 列表（被邀请方）

groupId	String	群组通话 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如视频通话、语音通话
userData	String	用户增加的扩展字段，详见 TUICallDefine.CallParams

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）

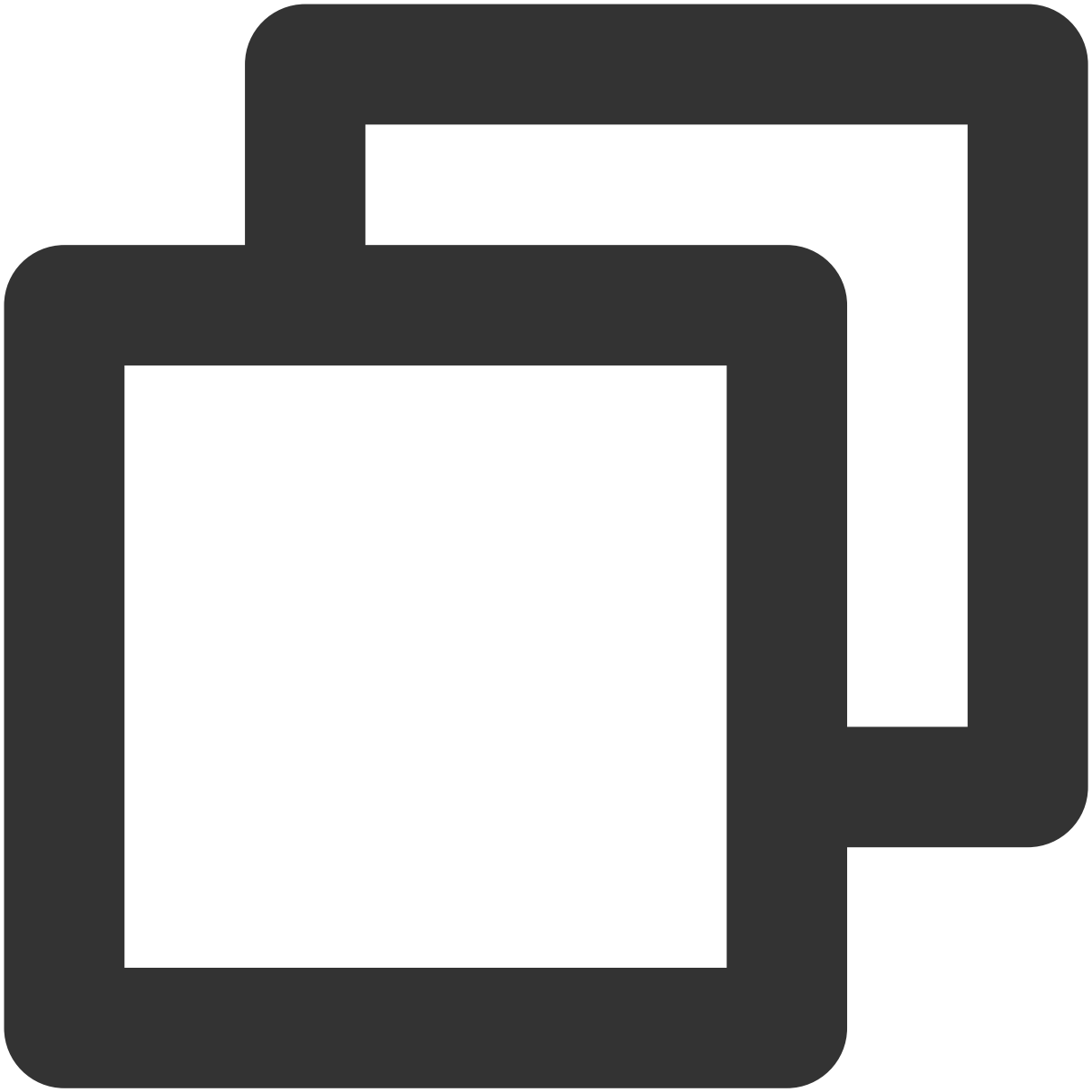
被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）

被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）

被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）

异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）

版权所有：腾讯云计算（北京）有限责任公司



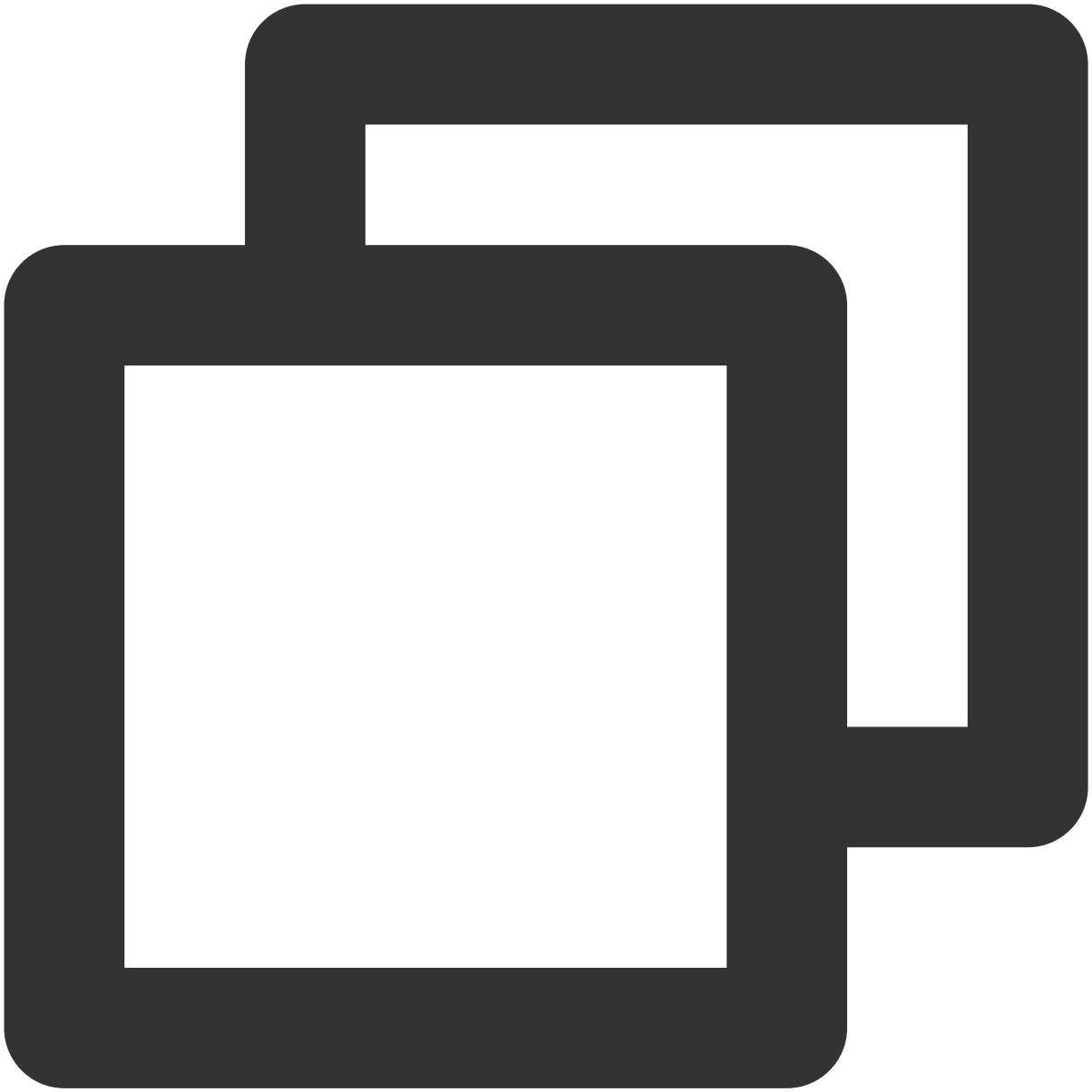
```
void onCallCancelled(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	取消用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。



```
void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaTy
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
callMediaType	TUICallDefine.MediaType	通话的媒体类型，视频通话、语音通话

callRole	TUICallDefine.Role	角色，枚举类型：主叫、被叫

onCallEnd

表示通话挂断，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType
```

参数如下表所示：

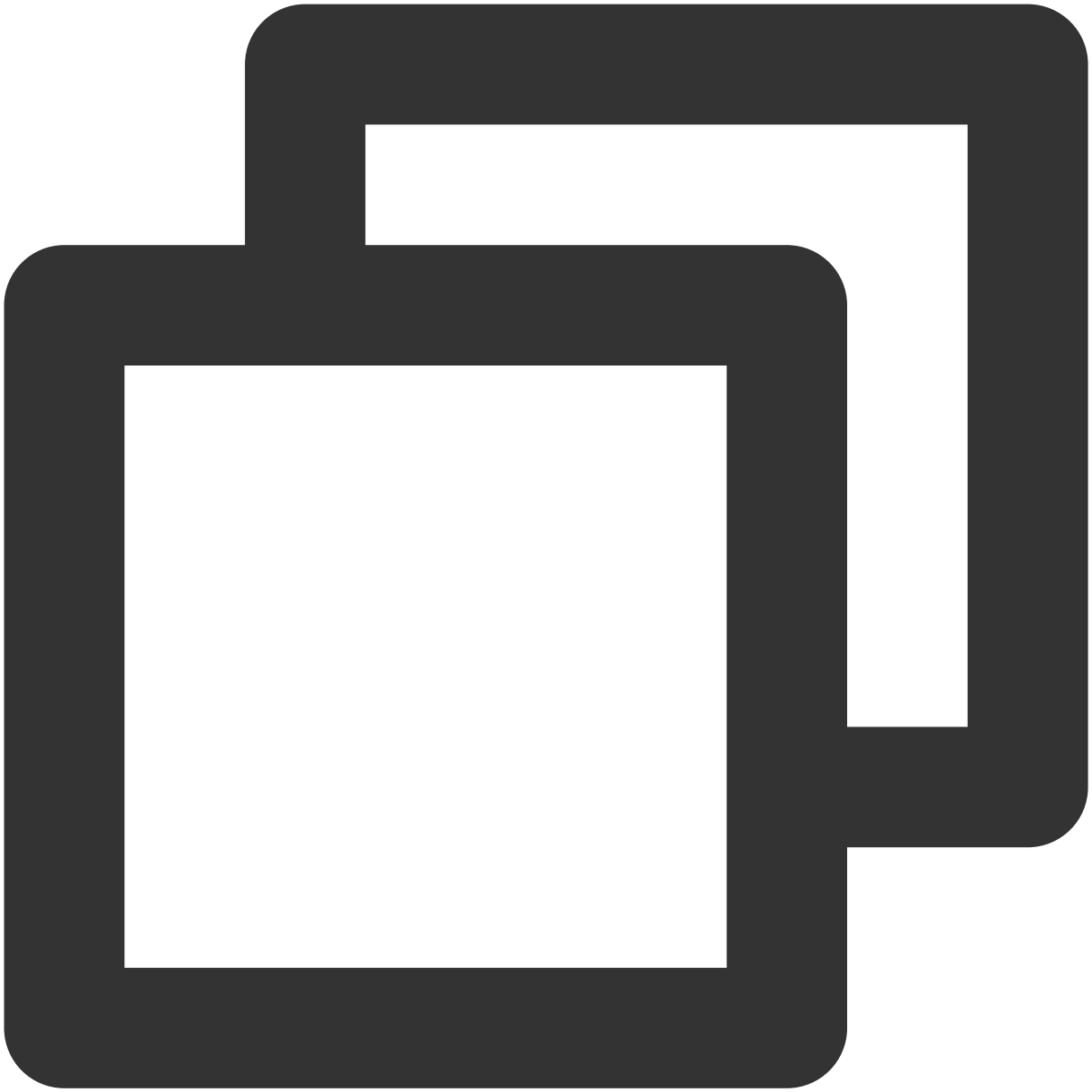
参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
callMediaType	TUICallDefine.MediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallDefine.Role	角色，枚举类型：主叫、被叫
totalTime	long	此次通话的时长；单位：秒

注意

客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用REST API来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。



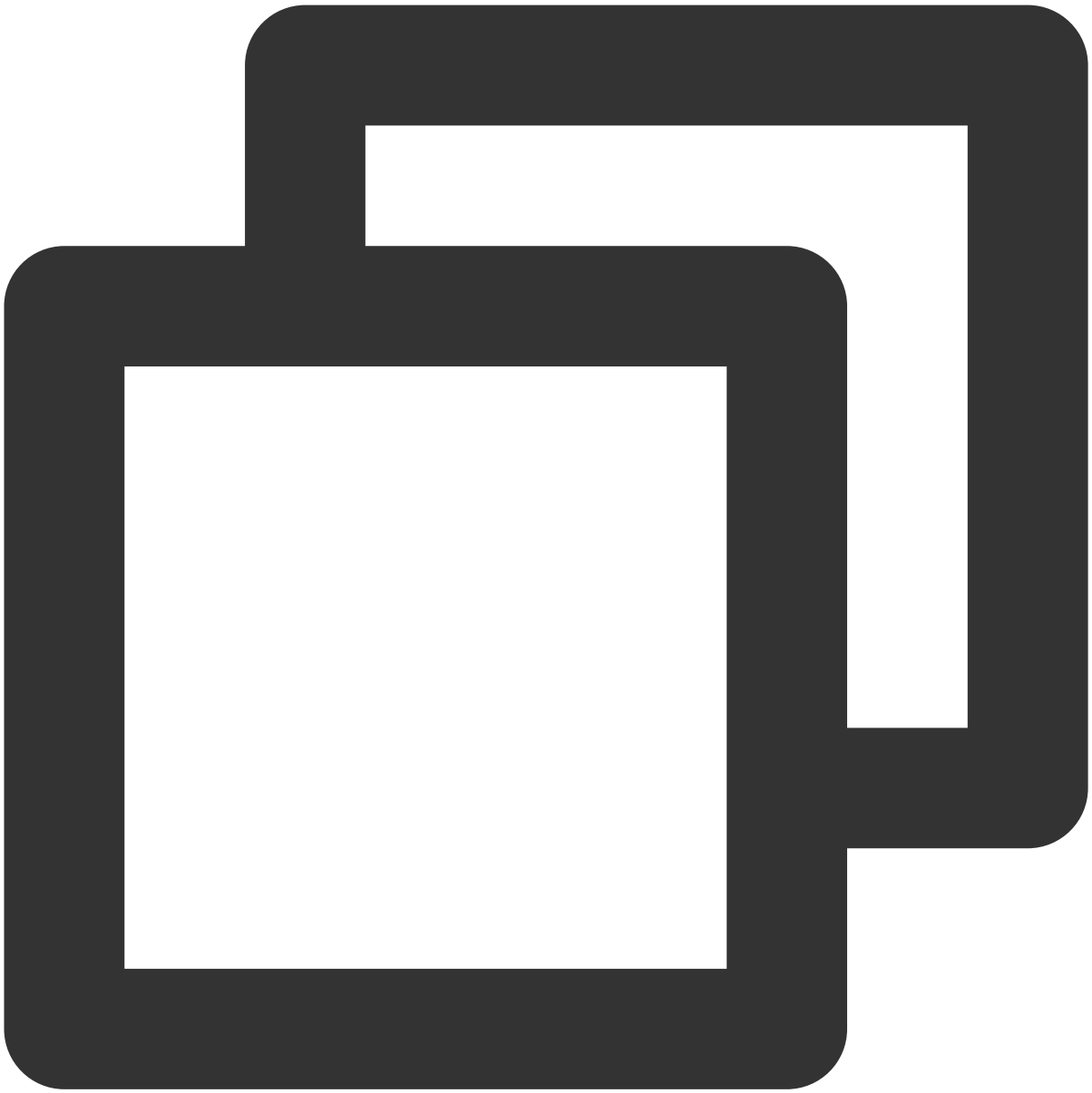
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType,TUICallDefine.
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallDefine.MediaType	旧的通话类型
newCallMediaType	TUICallDefine.MediaType	新的通话类型

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。



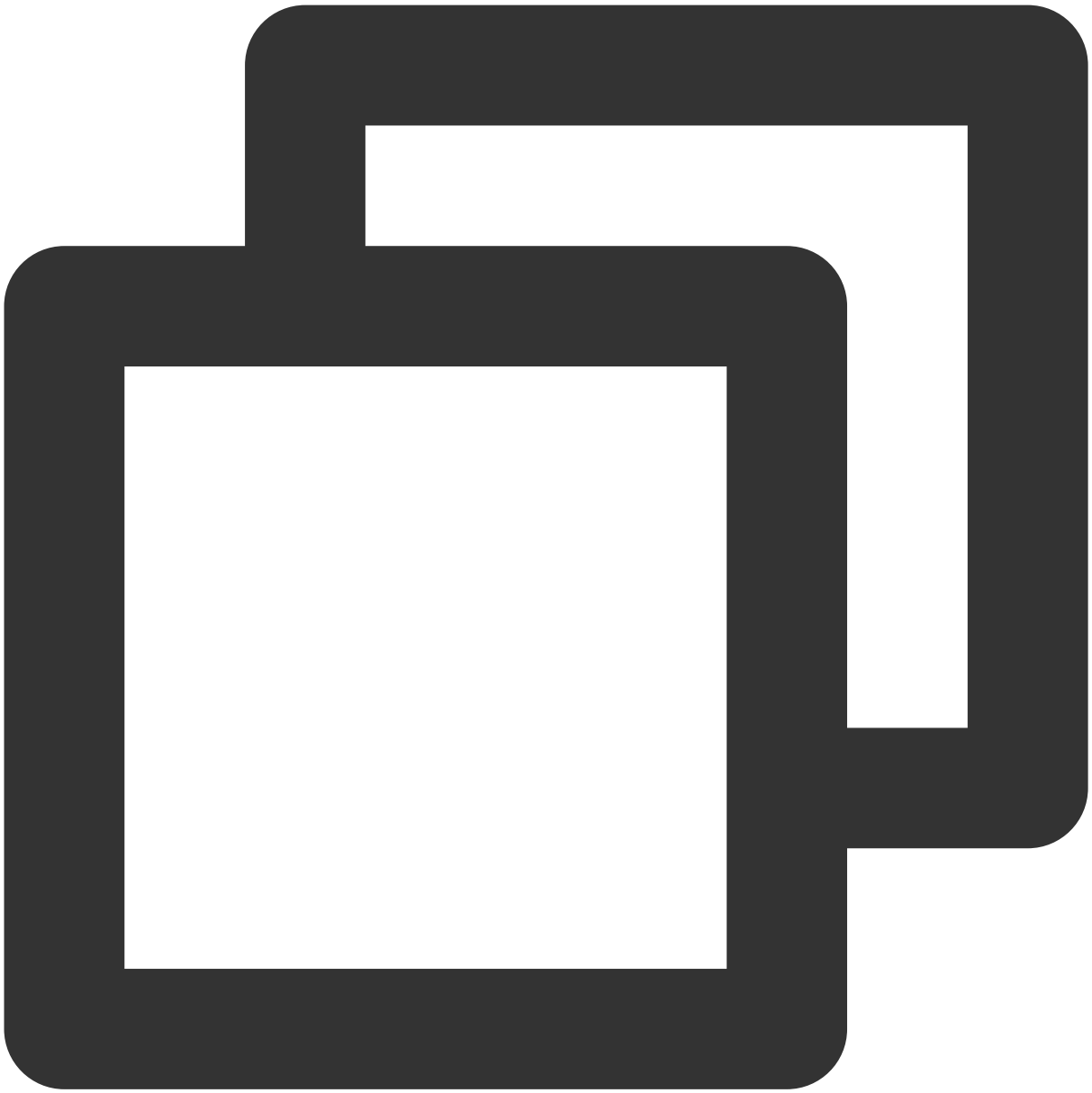
```
void onUserReject(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。



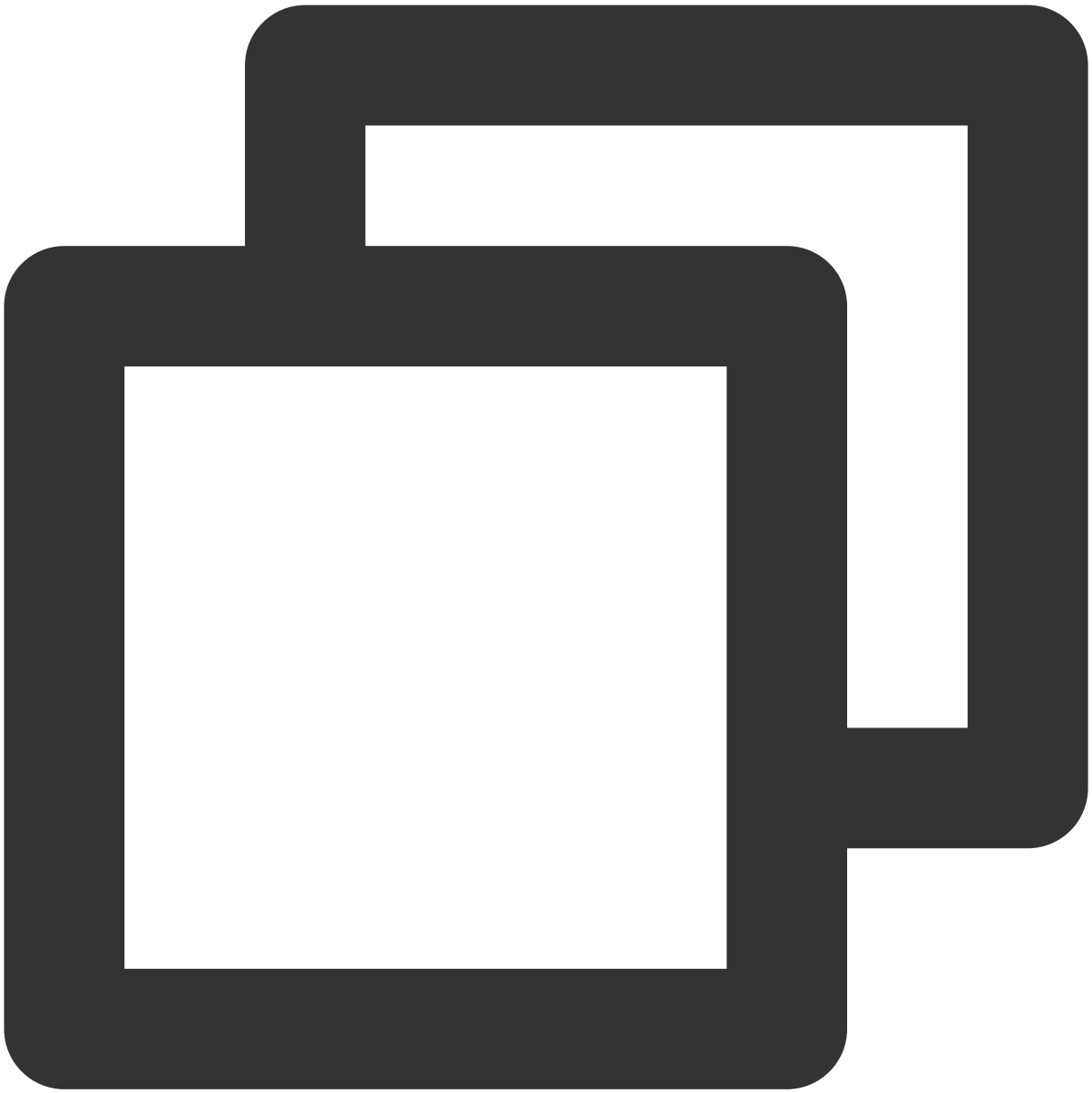
```
void onUserNoResponse(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	无响应用户的 ID

onUserLineBusy

通话忙线回调。



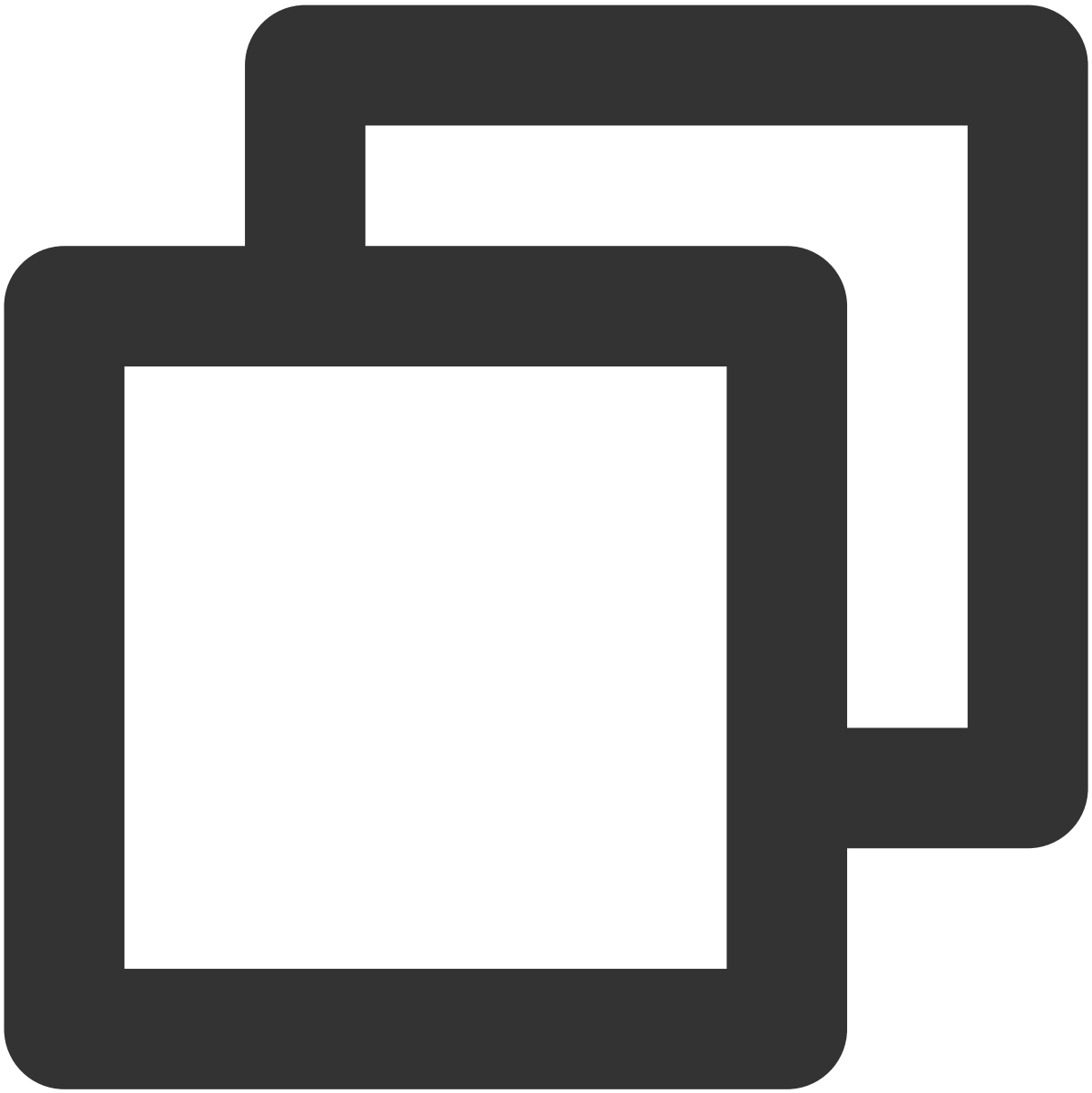
```
void onUserLineBusy(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID

onUserJoin

有用户进入此次通话的回调。



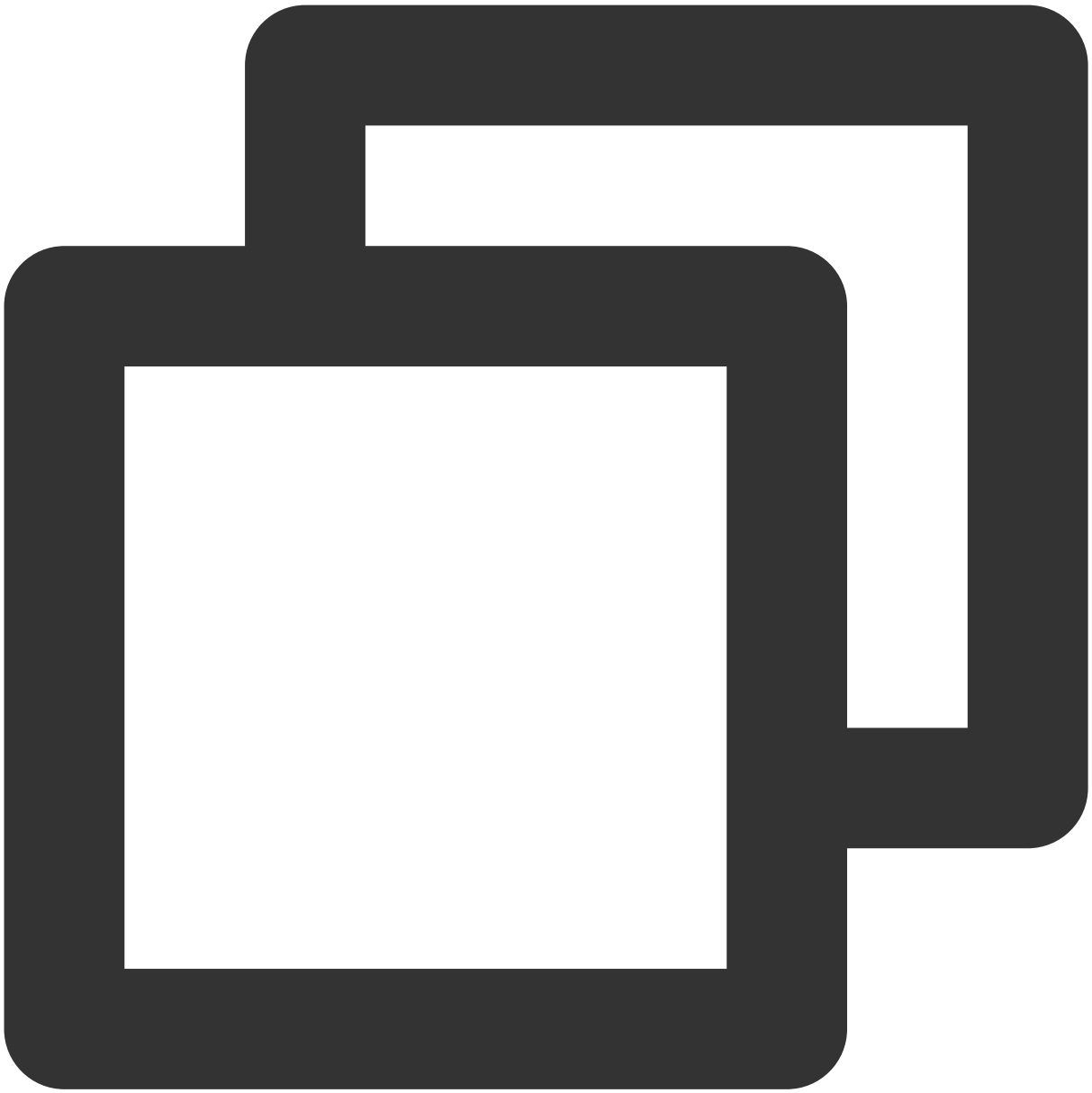
```
void onUserJoin(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	加入当前通话的用户 ID

onUserLeave

有用户离开此次通话的回调。



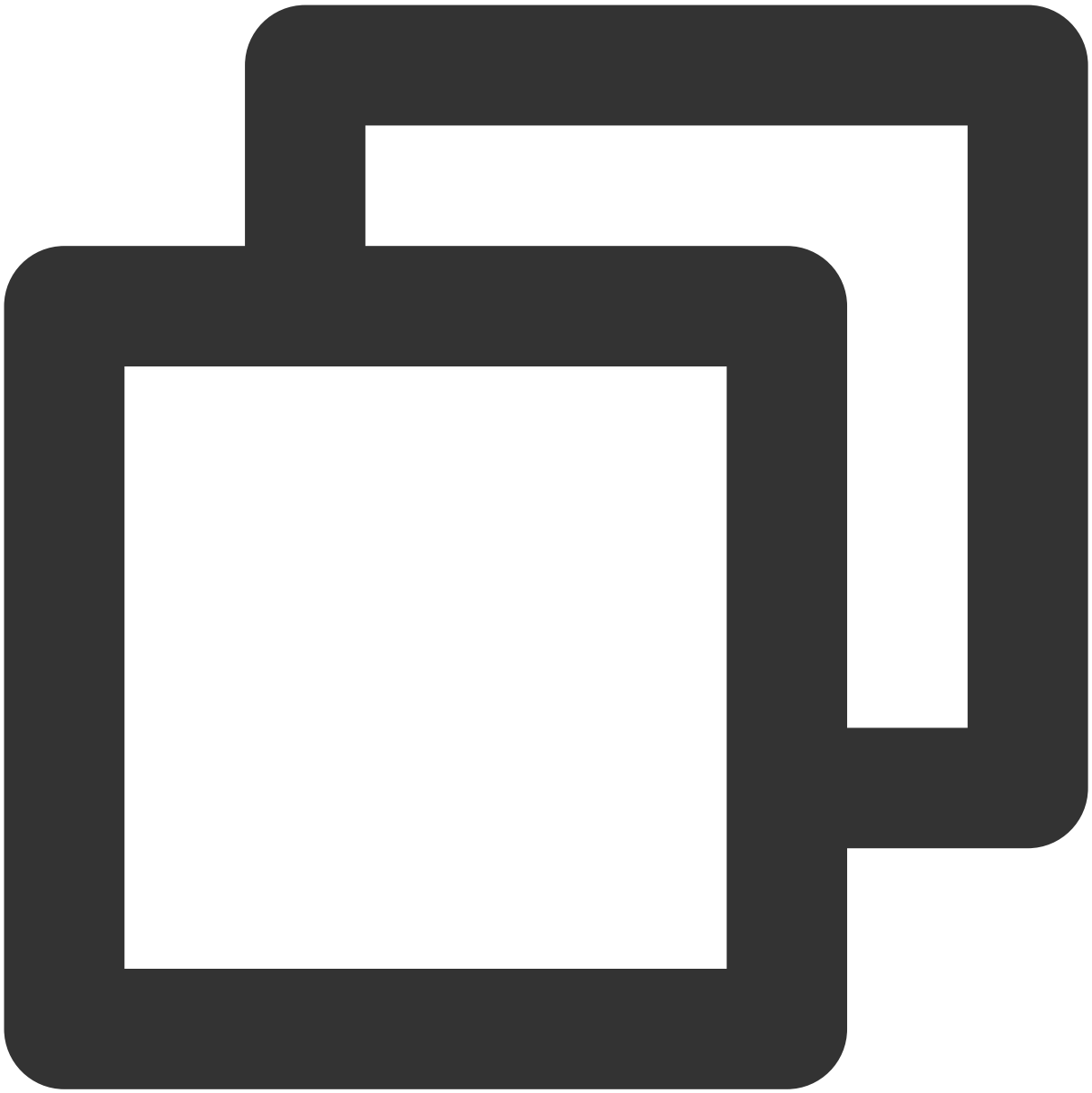
```
void onUserLeave(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	离开当前通话的用户 ID

onUserVideoAvailable

用户是否开启视频上行回调。



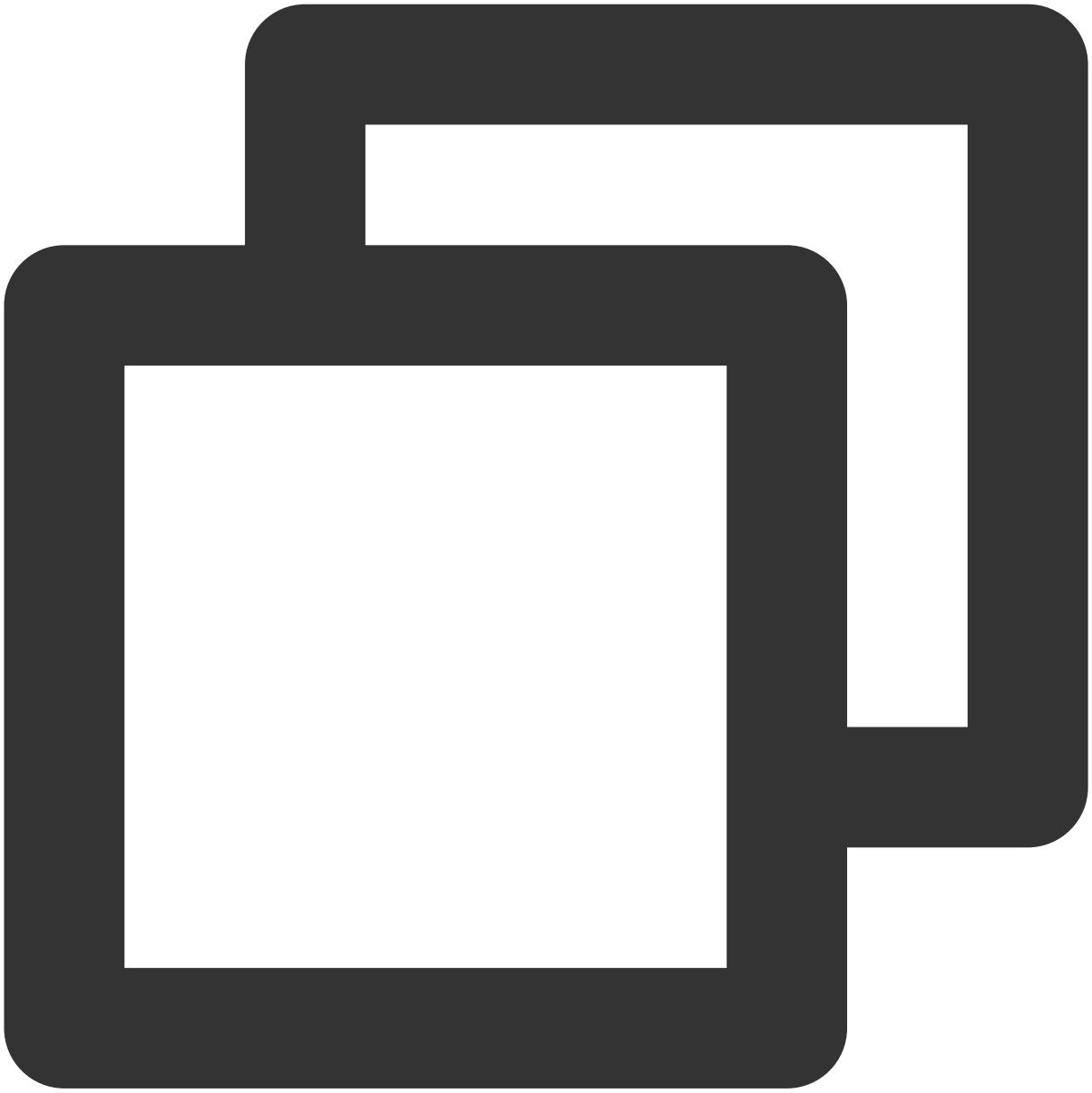
```
void onUserVideoAvailable(String userId, boolean isVideoAvailable);
```

参数如下表所示：

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

onUserAudioAvailable

用户是否开启音频上行回调。



```
void onUserAudioAvailable(String userId, boolean isAudioAvailable);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID
isAudioAvailable	boolean	用户音频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

```
void onUserVoiceVolumeChanged(Map<String, Integer> volumeMap);
```

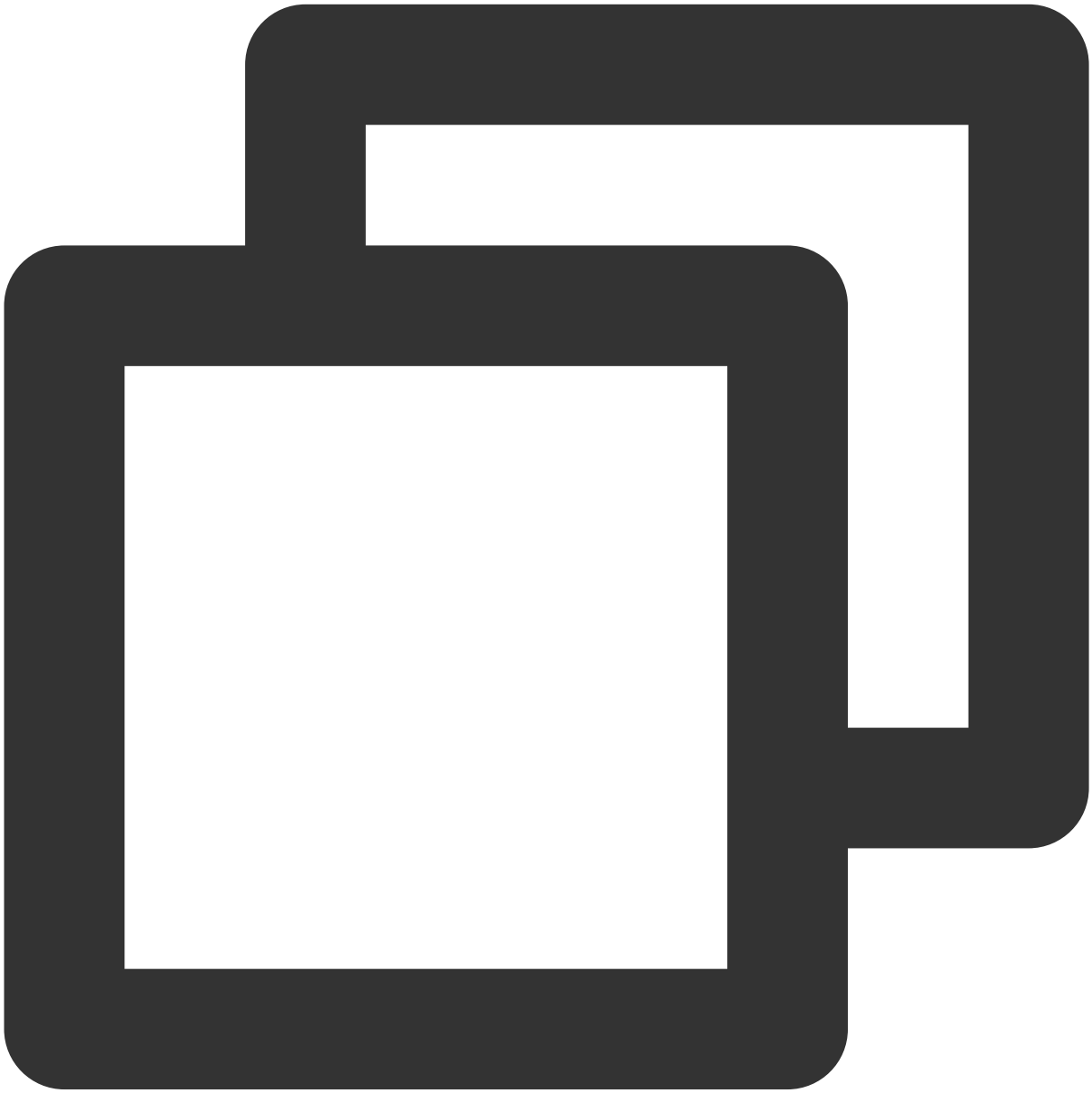
参数如下表所示：

参数	类型	含义
volumeMap	Map<String,	音量表，根据每个 userId 可以获取对应用户的音量大小，音量最小

	Integer>	值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。



```
void onUserNetworkQualityChanged(List<TUICallDefine.NetworkQualityInfo> networkQual
```

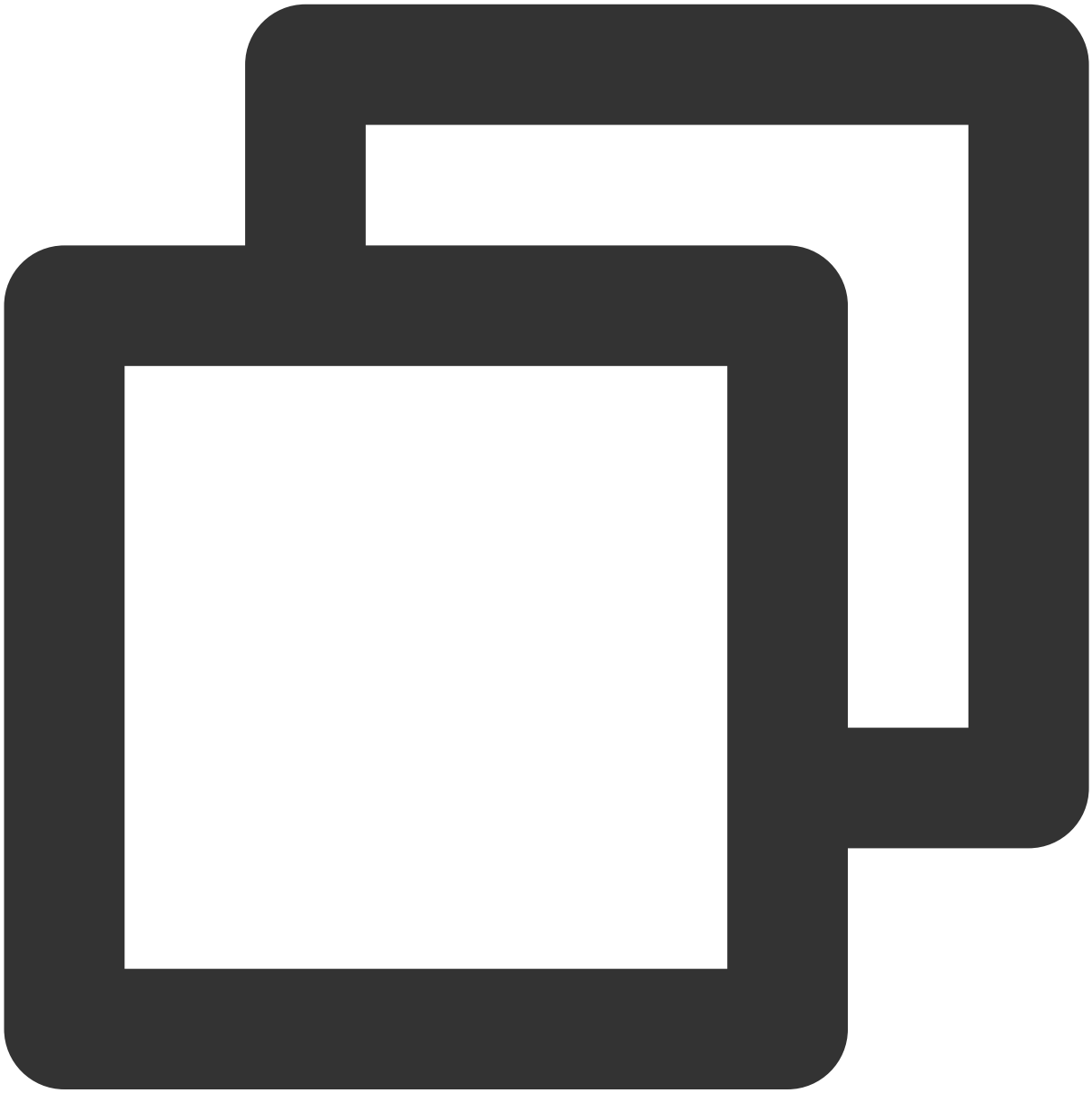
参数如下表所示：

参数	类型	含义

networkQualityList	List	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。



```
void onKickedOffline();
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。

```
void onUserSigExpired();
```

类型定义

最近更新时间：2023-06-30 15:41:34

常用结构

TUICallDefine

类型	描述
CallParams	通话参数
OfflinePushInfo	离线推送厂商配置信息

TUICommonDefine

类型	描述
RoomId	通话中音视频房间 ID
NetworkQualityInfo	网络质量
VideoRenderParams	视频画面的渲染参数
VideoEncoderParams	视频编码参数

枚举定义

TUICallDefine

类型	说明
MediaType	通话类型
Role	通话角色
Status	通话状态
Scene	通话场景
IOSOfflinePushType	iOS 离线推送类型

TUICommonDefine

类型	说明

版权所有：腾讯云计算（北京）有限责任公司

AudioPlaybackDevice	音频设备
Camera	摄像头类型
NetworkQuality	网络质量
FillMode	视频画面填充模式
Rotation	视频画面旋转方向
ResolutionMode	视频宽高比模式
Resolution	视频分辨率

CallParams

通话参数

类型	类型	描述
offlinePushInfo	OfflinePushInfo	厂商离线推送配置信息
timeout	int	通话超时时间，默认：30s，单位：秒
userData	String	发起通话时自定义扩展字段，被叫端在 onCallReceived 中回调

OfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

类型	类型	描述
title	String	离线推送展示通知栏标题
desc	String	离线推送展示通知栏内容
ignoreIOSBadge	boolean	离线推送忽略 badge 计数（仅对 iOS 生效）， 如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加。
iOSSound	String	离线推送声音设置（仅对 iOS 生效）。 当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。 当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。 如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀）设置给 iOSSound

androidSound	String	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。 只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声。 另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	int	VIVO 推送消息分类 (待废弃接口，VIVO 推送服务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	String	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	String	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	String	华为推送消息分类，详见：华为消息分类标准
isDisablePush	boolean	是否关闭推送（默认开启推送）
iOSPushType	IOSOfflinePushType	iOS 离线推送类型，默认：APNs

RoomId

通话中音视频房间ID

特别说明

(1) intRoomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 intRoomId 需要填写为 0。若两者都填，SDK 将优先选用 intRoomId。

(2) 不要混用 intRoomId 和 strRoomId，因为它们之间是不互通的，比如数字123和字符串123是两个完全不同的房间。

类型	类型	描述
intRoomId	int	数字房间号 **取值范围：**1 - 2147483647(2^31-1)
strRoomId	String	字符串房间号 **推荐取值：** 限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空 格、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`、`=`、

注意：

字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！

NetworkQualityInfo

用户网络质量

类型	类型	描述
userId	String	用户 ID
quality	NetworkQuality	网络质量

VideoRenderParams

视频画面的渲染参数

类型	类型	描述
fillMode	VideoRenderParams.FillMode	视频画面填充模式
rotation	VideoRenderParams.Rotation	视频画面旋转方向

VideoEncoderParams

视频编码参数

类型	类型	描述
resolution	VideoEncoderParams.Resolution	视频分辨率
resolutionMode	VideoEncoderParams.ResolutionMode	视频宽高比

MediaType

通话类型

类型	取值	描述
Unknown	0	未知类型
Audio	1	语音通话
Video	2	视频通话

Role

通话角色

类型	取值	描述
None	0	未知类型
Caller	1	主叫（邀请方）
Called	2	被叫（被邀请方）

Status

通话状态

类型	取值	描述
None	0	未定义
Waiting	1	通话等待中
Accept	2	通话已接听

Scene

通话场景

类型	取值	描述
GROUP_CALL	0	群组通话
MULTI_CALL	1	匿名群组通话（暂不支持，敬请期待）
SINGLE_CALL	2	单人通话

IOSOfflinePushType

iOS离线推送类型

类型	取值	描述
APNs	0	普通推送
VoIP	1	VoIP 推送

AudioPlaybackDevice

音频设备

类型	取值	描述
Earpiece	0	听筒

版权所有：腾讯云计算（北京）有限责任公司

Speakerphone	1	扬声器

Camera

前置/后置 摄像头

类型	取值	描述
Front	0	前置摄像头
Back	1	后置摄像头

NetworkQuality

网络质量

类型	取值	描述
Unknown	0	未定义
Excellent	1	当前网络非常好
Good	2	当前网络比较好
Poor	3	当前网络一般
Bad	4	当前网络较差
Vbad	5	当前网络很差
Down	6	当前网络不满足通话的最低要求

FillMode

视频画面填充模式

类型	取值	描述
Fill	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完整。
Fit	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边。

Rotation

视频画面旋转方向

类型	取值	描述
Rotation_0	0	不旋转
Rotation_90	1	顺时针旋转90度
Rotation_180	2	顺时针旋转180度
Rotation_270	3	顺时针旋转270度

ResolutionMode

视频宽高比模式

类型	取值	描述
Landscape	0	横屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360。
Portrait	1	竖屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640。

Resolution

视频分辨率

类型	取值	描述
Resolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps
Resolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
Resolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
Resolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
Resolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
Resolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

版权所有：腾讯云计算（北京）有限责任公司

错误码表

最近更新时间：2023-06-30 15:43:34

通知用户在使用音视频通话过程中出现的警告和错误。

TUICallDefine 错误码

定义	取值	描述
ERROR_PACKAGE_NOT_PURCHASED	-1001	您当前未购买音视频通话能力套餐，请前往IM控制台 开通免费体验 或 加购正式版
ERROR_PACKAGE_NOT_SUPPORTED	-1002	您当前购买的音视频通话能力套餐不支持该功能，建议您前往IM控制台 升级套餐类型
ERROR_TIM_VERSION_OUTDATED	-1003	IM SDK 版本过低，请升级 IM SDK 版本 >= 6.6；在build.gradle文件中查找并修改："com.tencent.imsdk:imsdk-plus:7.1.3925"
ERROR_PERMISSION_DENIED	-1101	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
ERROR_GET_DEVICE_LIST_FAIL	-1102	获取设备列表失败（仅 web 平台支持）
ERROR_INIT_FAIL	-1201	未调用 init 方法进行初始化，TUICallEngine API 需在初始化之后使用
ERROR_PARAM_INVALID	-1202	参数错误
ERROR_REQUEST_REFUSED	-1203	当前状态不支持调用
ERROR_REQUEST_REPEATED	-1204	当前方法正在执行中，请勿重复调用
ERROR_SCENE_NOT_SUPPORTED	-1205	当前通话场景，不支持该功能
ERROR_SIGNALING_SEND_FAIL	-1406	信令发送失败，具体失败原因可查看具体方法回调中的 errorMsg

IM 错误码

版权所有：腾讯云计算（北京）有限责任公司

音视频通话使用了腾讯云 IM SDK 作为通信的基础服务，比如通话拨打信令、通话忙线信令等核心逻辑。常见错误码如下：

错误码	描述
6014	IM SDK 未登录，请先登录，成功回调之后重试，或者已被踢下线，可使用 TIMManager getLoginUser 检查当前是否在线。
6017	参数无效，请检查参数是否符合要求，具体可查看错误信息进一步定义哪个字段。
6206	UserSig 过期，请重新获取有效的 UserSig 后再重新登录，获取方法请参见 生成 UserSig。
7013	套餐包不支持该接口的使用，请升级到旗舰版套餐。
8010	信令请求 ID 无效或已经被处理过。

说明：

更多 IM SDK 错误码详见：IM错误码

TRTC 错误码

音视频通话使用了腾讯云 TRTC SDK 作为音频和视频的基础服务，比如开关摄像头、开关麦克风等核心逻辑。常见错误码如下：

枚举	取值	描述
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了

ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_TRTC_ENTER_ROOM_FAILED	-3301	进入房间失败，请查看 onError 中的 -3301 对应的 msg 提示确认失败原因

说明：

更多 TRTC SDK 错误码详见：TRTC错误码

iOS
API 概览

最近更新时间：2024-07-30 11:22:04

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的头像、昵称
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
enableIncomingBanner	开启/关闭来电横幅显示

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述

createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（群组通话版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式

版权所有：腾讯云计算（北京）有限责任公司

setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

版权所有：腾讯云计算（北京）有限责任公司

关键类型定义

API	描述
TUICallMediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallRole	通话的角色，枚举类型：主叫、被叫
TUICallStatus	通话的状态，枚举类型：空闲、待接听、接听中
TUICallParams	通话扩展参数：离线推送消息显示内容
TUIRoomId	音视频房间Id，支持数字、字符串两种类型
TUICamera	摄像头Id参数，枚举类型：前摄、后摄
TUIAudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUINetworkQualityInfo	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2024-05-27 17:18:51

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入TUICallKit。

API 概览

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义离线推送消息显示内容
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义离线推送消息显示内容
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
enableIncomingBanner	开启/关闭来电横幅显示

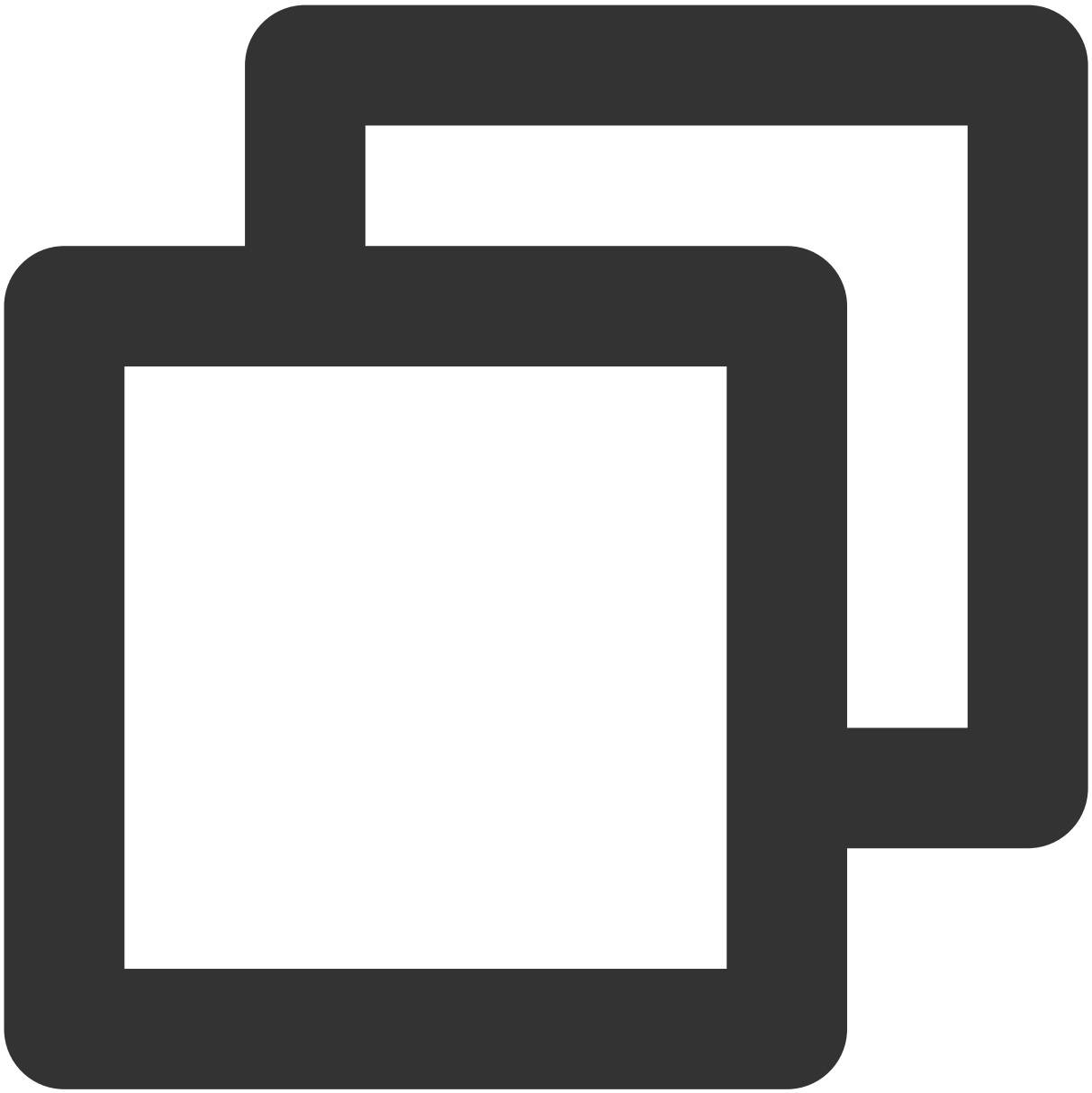
API 详情

createInstance

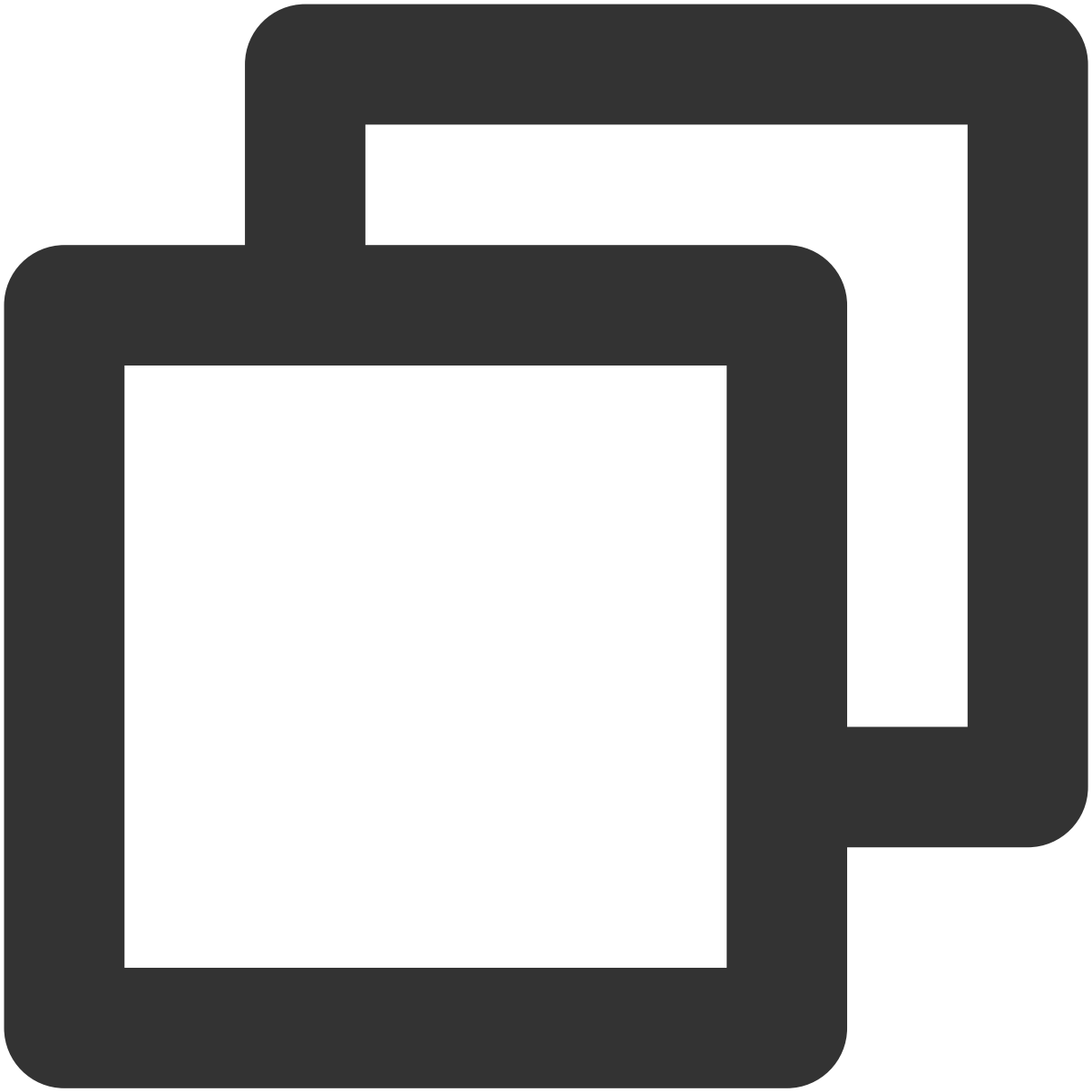
创建 TUICallKit 的单例。

Objective-C

Swift



```
- (instancetype)createInstance;
```



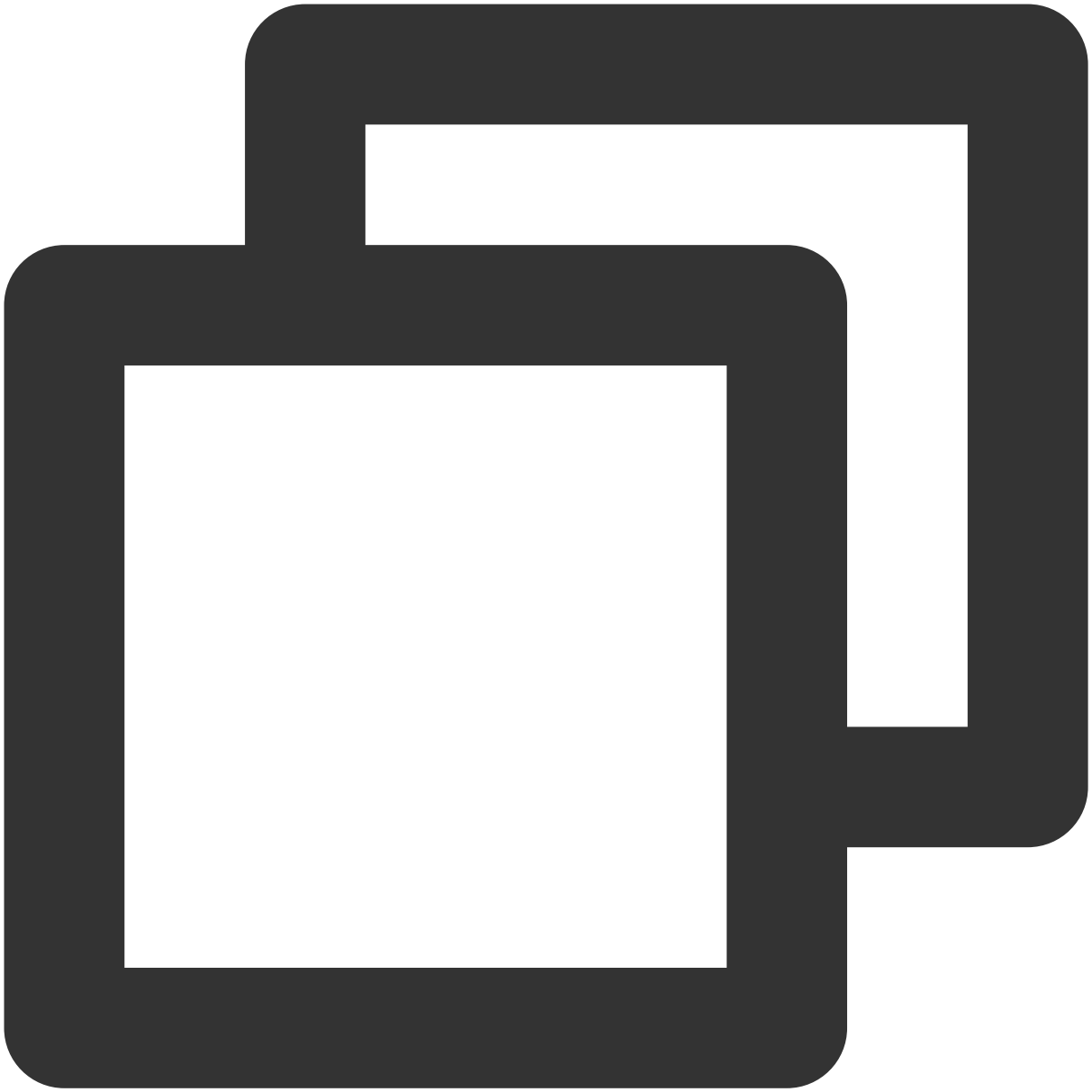
```
public static func createInstance() -> TUICallKit
```

setSelfInfo

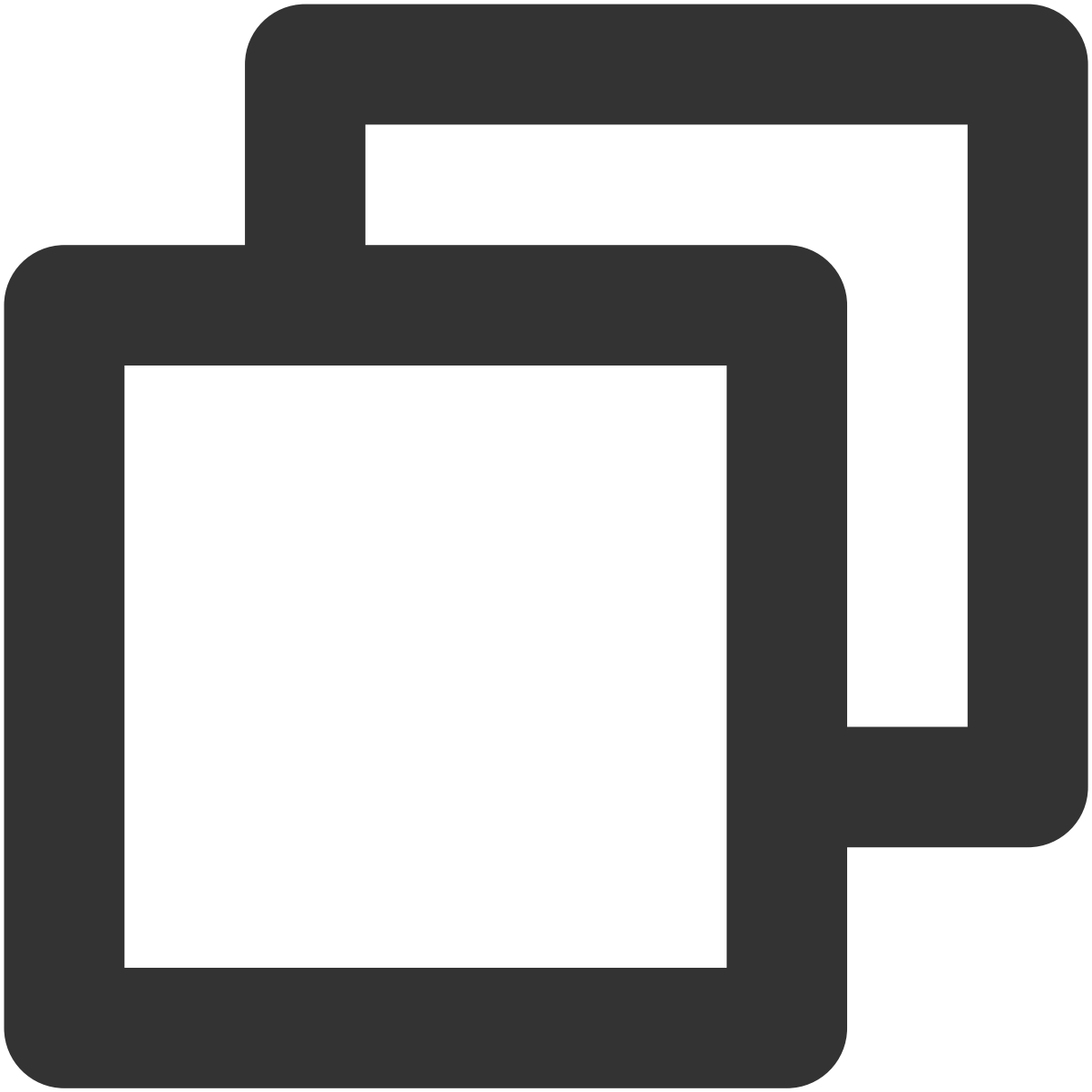
设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是 URL 格式。

Objective-C

Swift



```
- (void)setSelfInfo:(NSString * _Nullable)nickname avatar:(NSString * _Nullable)ava
```



```
public func setSelfInfo(nickname: String, avatar: String, succ:@escaping TUICallSuc
```

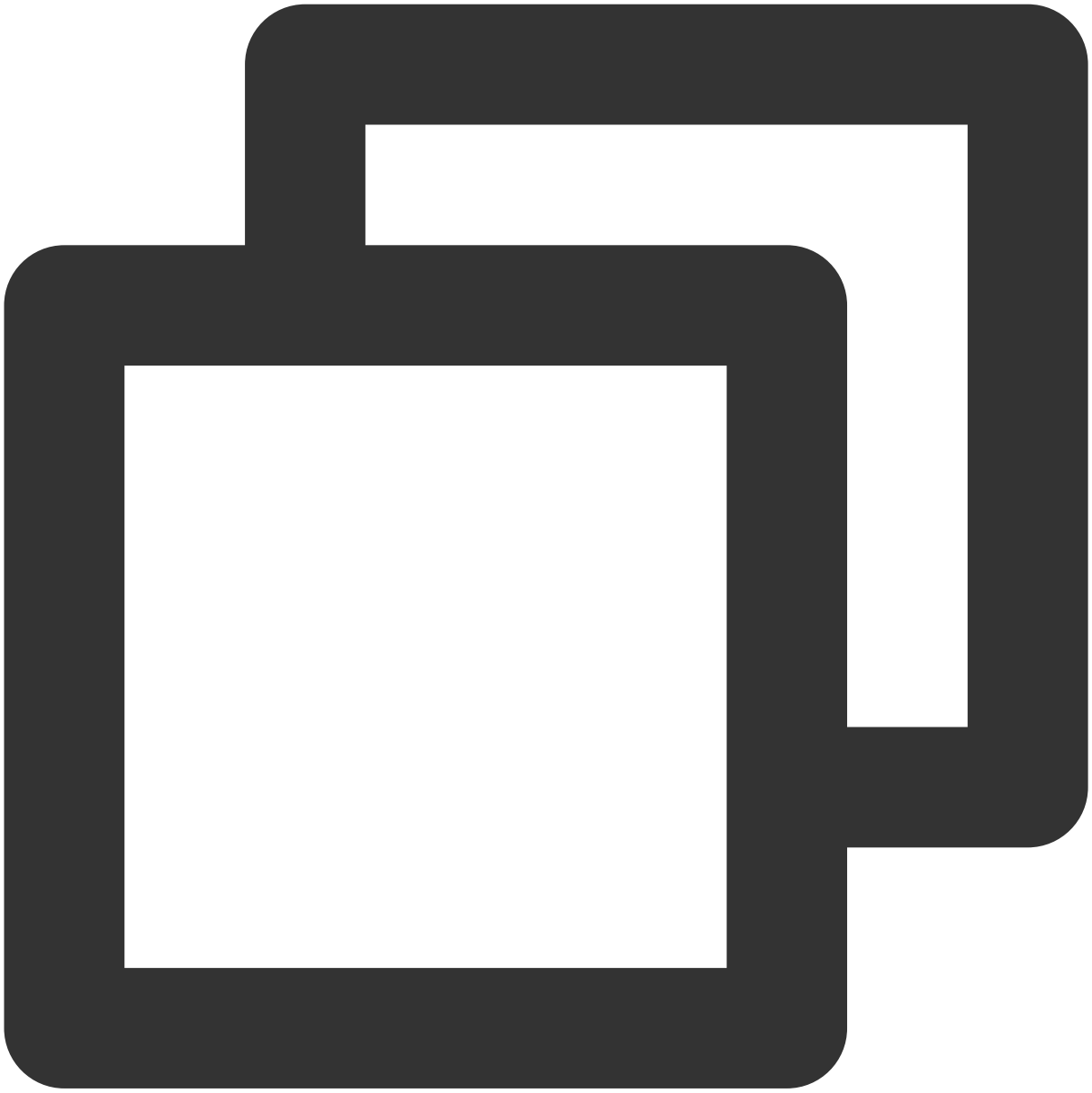
参数	类型	含义
nickname	NSString	用户昵称
avatar	NSString	用户头像（格式为 URL）

call

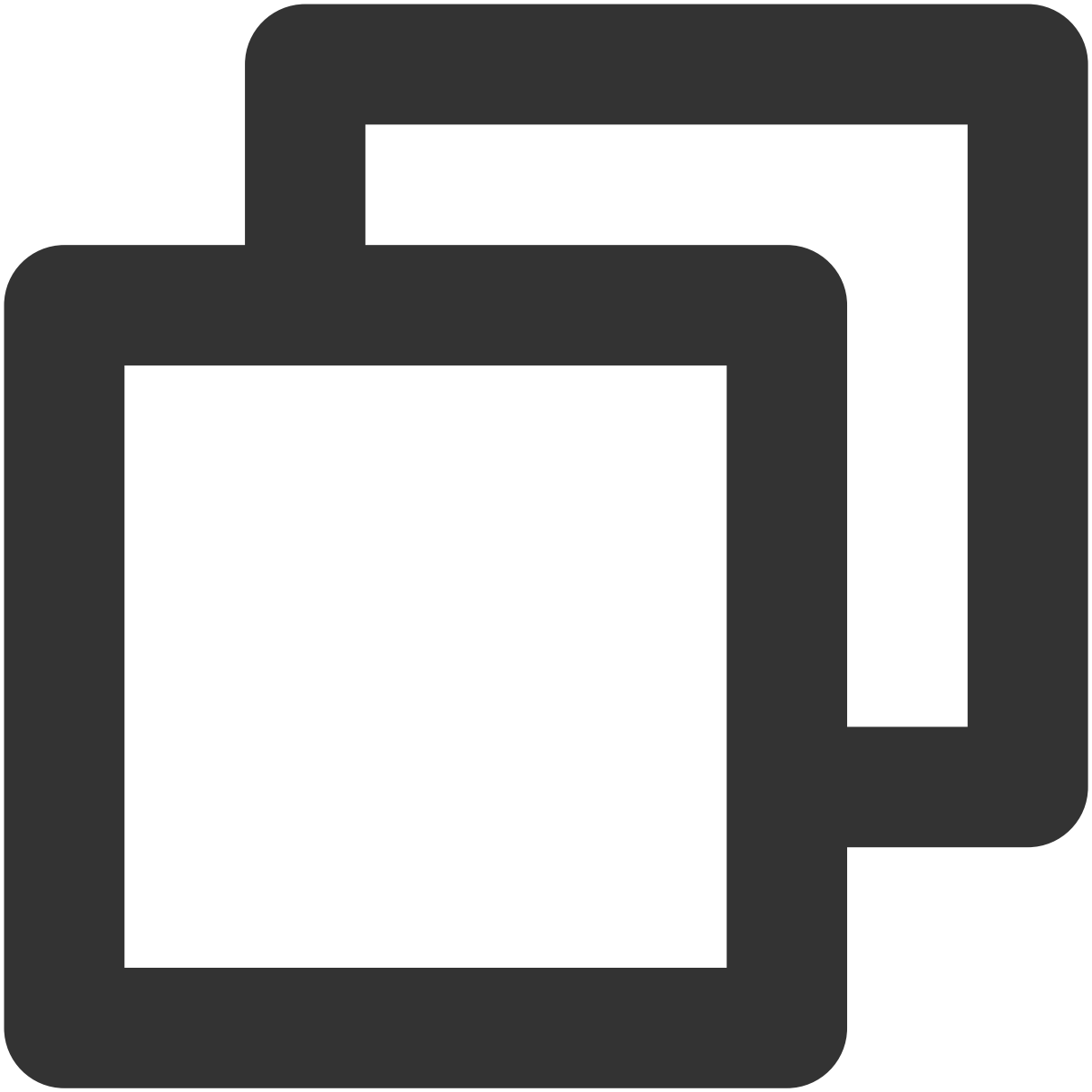
拨打电话（1v1通话）

Objective-C

Swift



```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType;
```



```
public func call(userId: String, callMediaType: TUICallMediaType)
```

参数如下表所示：

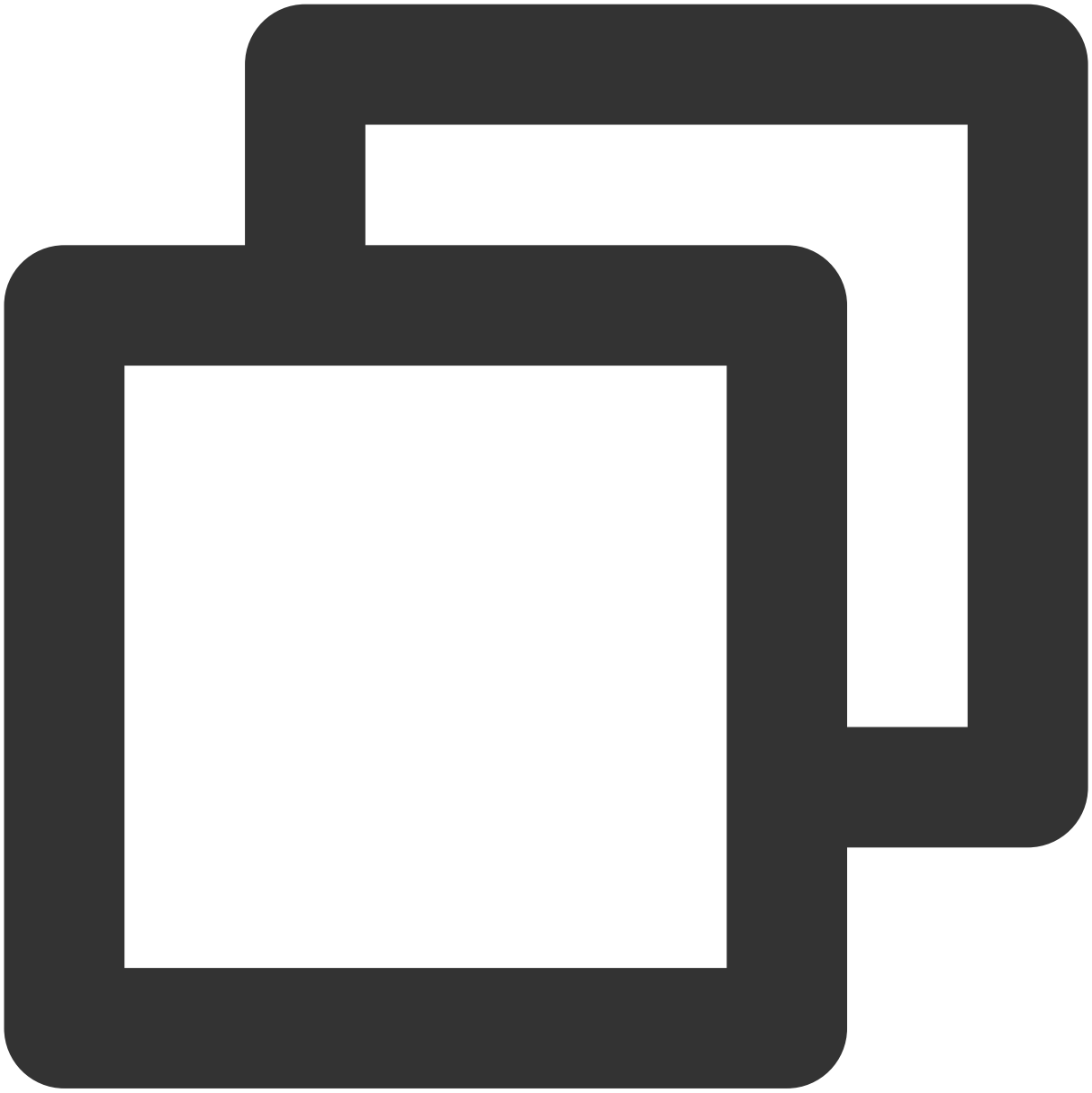
参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

call

拨打电话（1v1通话），支持自定义房间号、通话邀请超时时间，离线推送内容等。

Objective-C

Swift



```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType param
```

```
public func call(userId: String, callMediaType: TUICallMediaType, params: TUICallPa
                     succ: @escaping TUICallSucc, fail: @escaping TUICallFail)
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话

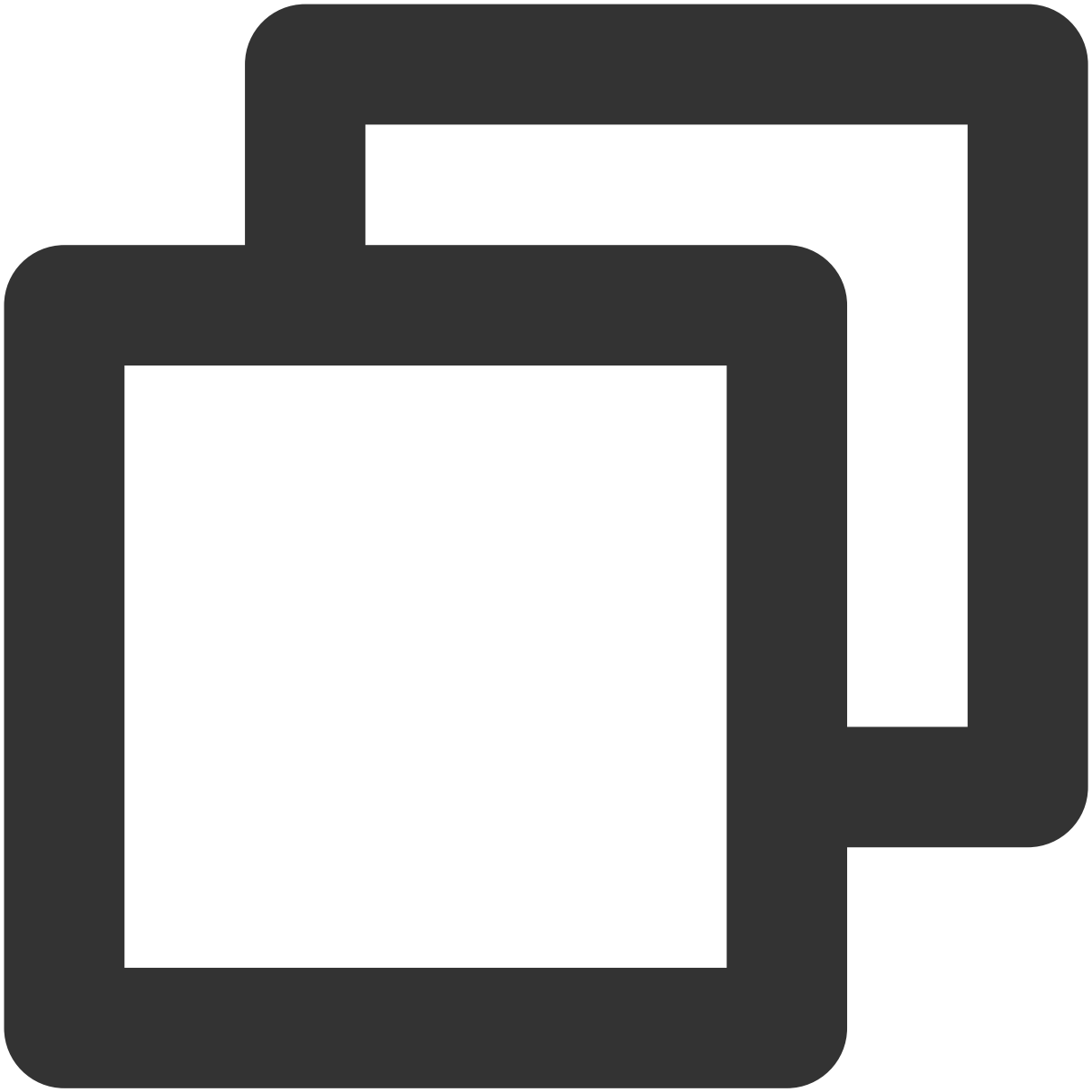
注意：

使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。

群组的创建详见：IM 群组管理，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Objective-C

Swift



```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray<NSString *> *)userIdList
```

```
public func groupCall(groupId: String, userIdList: [String], callMediaType: TUICall
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

groupCall

发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。

注意：

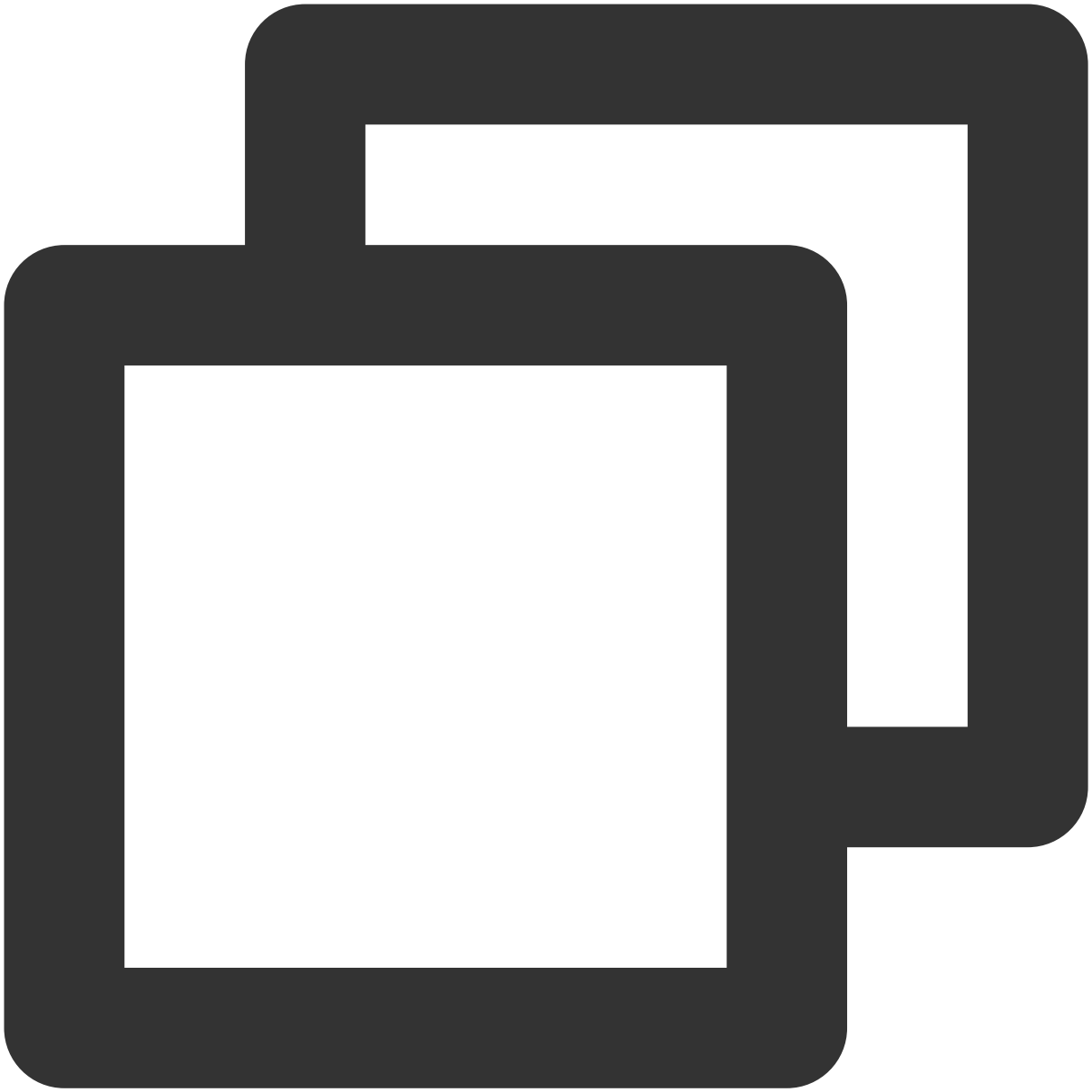
使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。

群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Objective-C

Swift

```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray<NSString *> *)userIdList
```



```
public func groupCall(groupId: String, userIdList: [String], callMediaType: TUICall
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时

版权所有：腾讯云计算（北京）有限责任公司

		间，离线推送自定义内容等

joinInGroupCall

发起群组通话

注意：

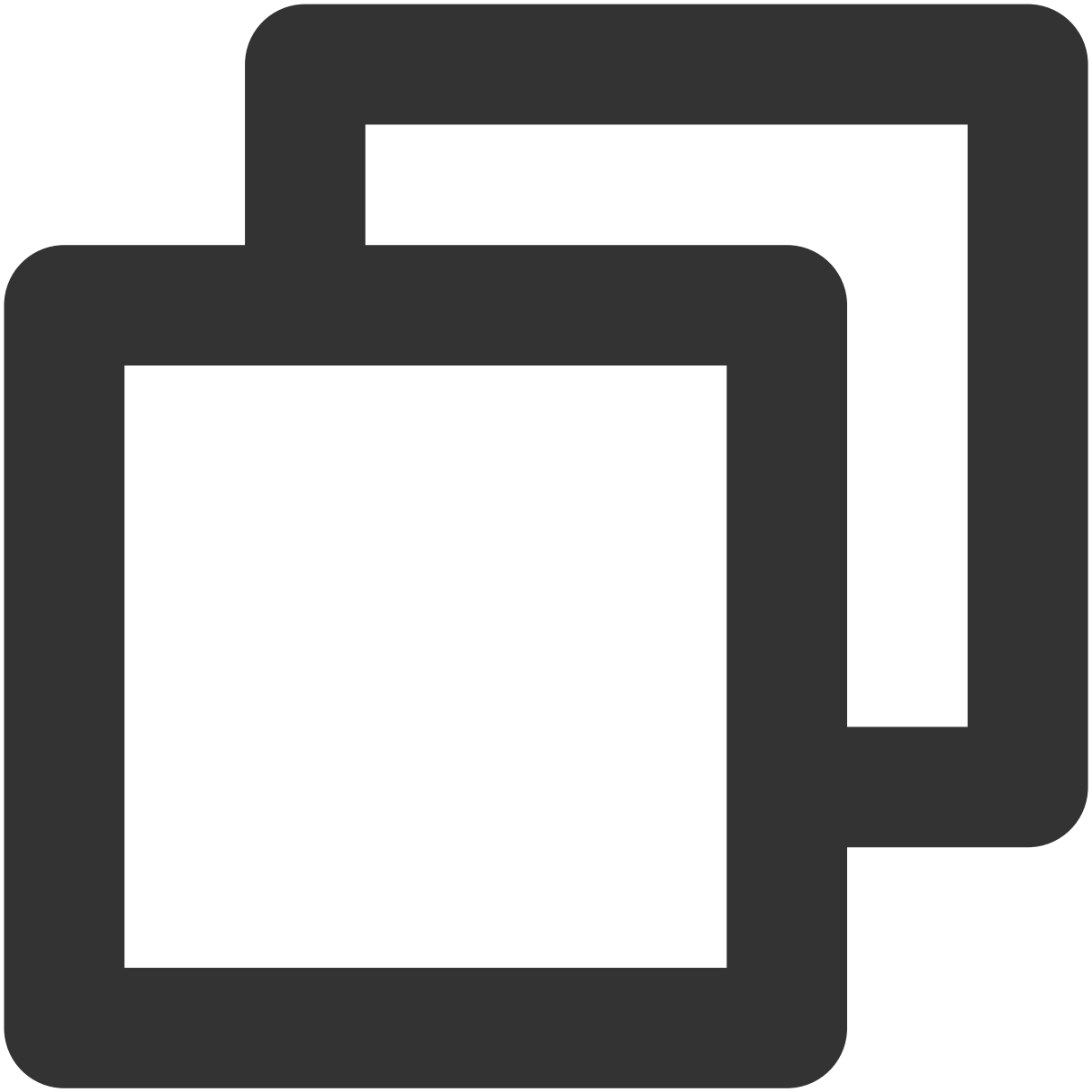
加入群组中已有的音视频通话前，需要提前创建或加入 IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。

群组的创建详见：IM 群组管理，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Objective-C

Swift

```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```



```
public func joinInGroupCall(roomId: TUIRoomId, groupId: String, callMediaType: TUIC
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
groupId	NSString	此次群组通话的群 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

setCallingBell

设置自定义来电铃音。

这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

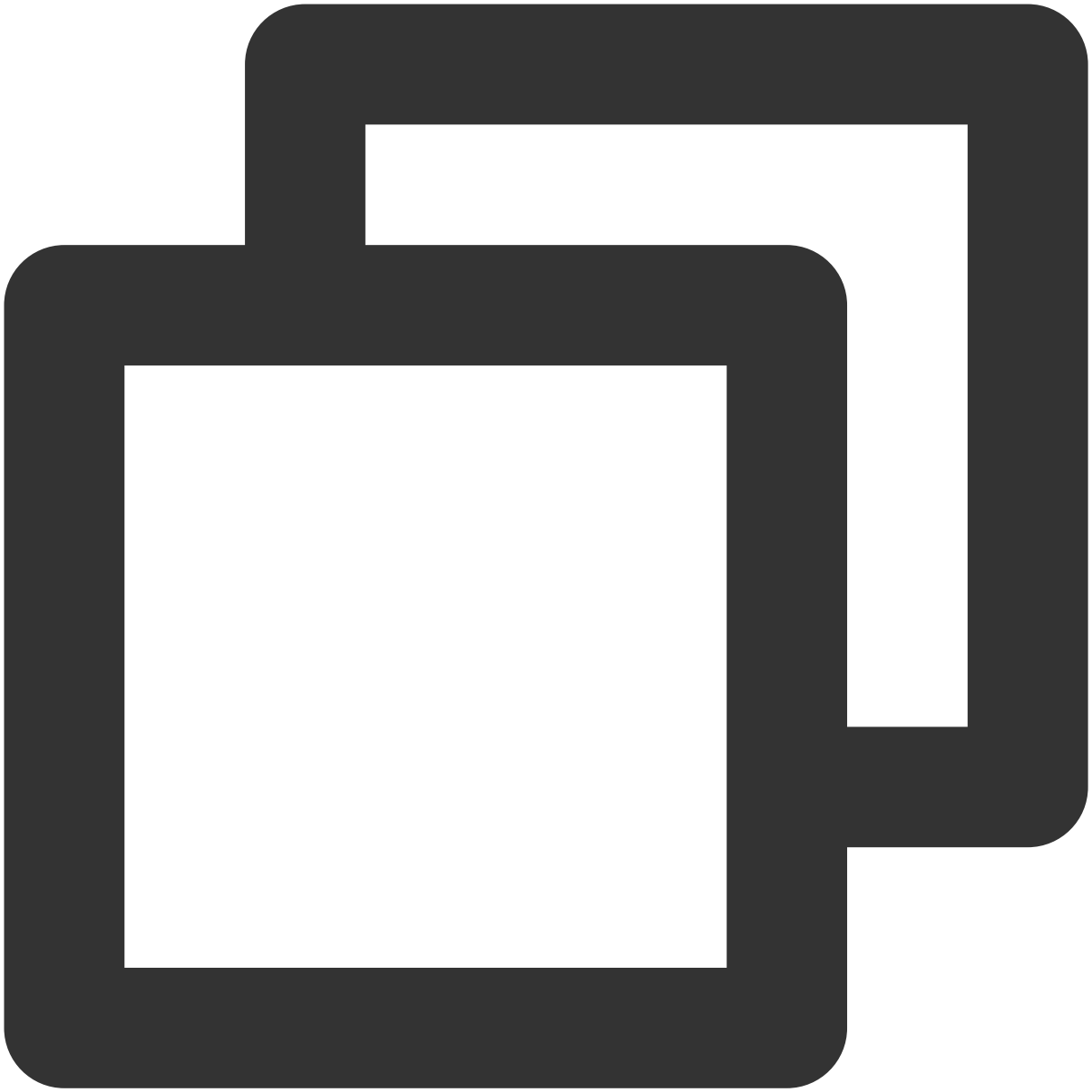
铃声设置后与设备绑定，更换用户，铃声依旧会生效。

如需恢复默认铃声， `filePath` 传空即可。

Objective-C

Swift

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



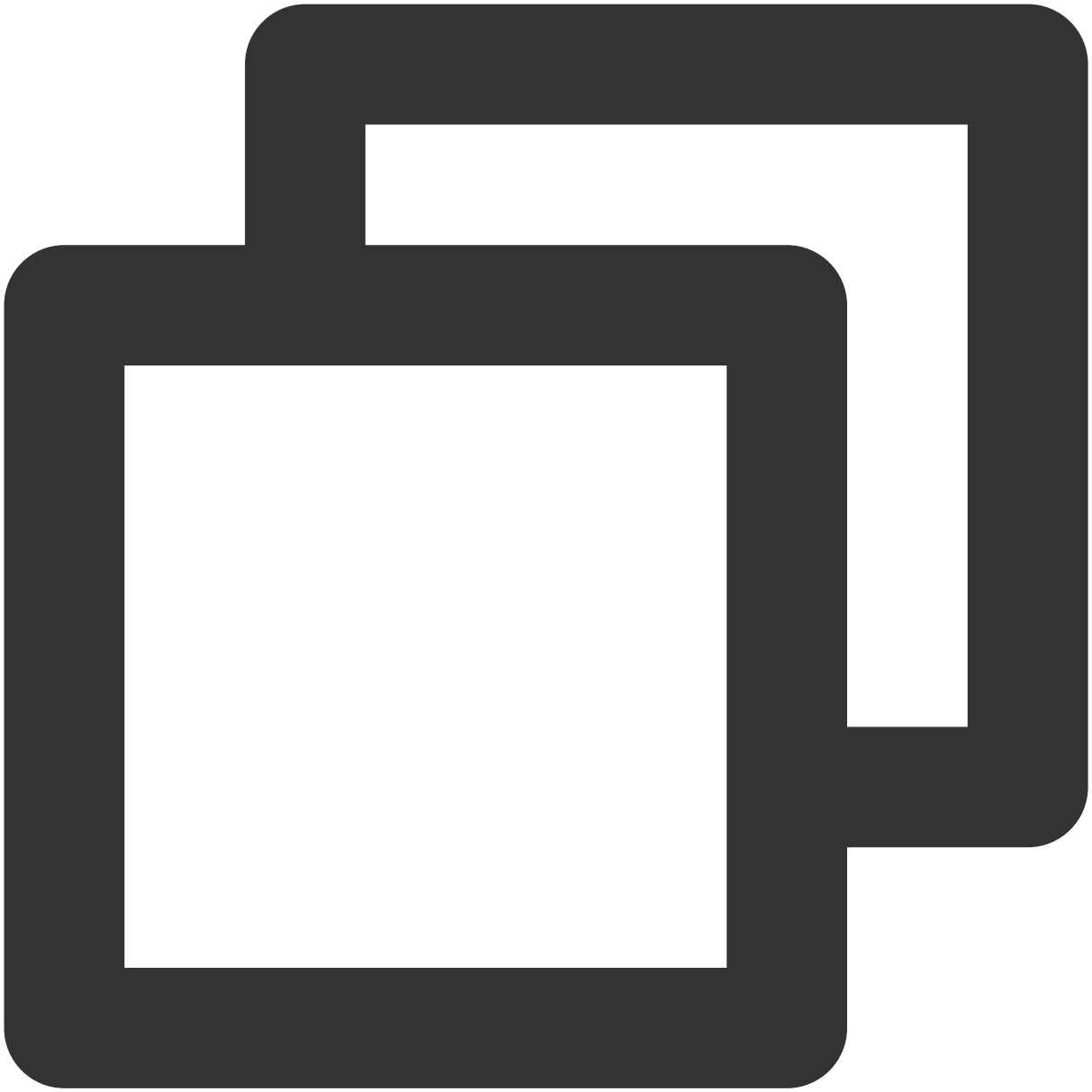
```
public func setCallingBell(filePath: String)
```

enableMuteMode

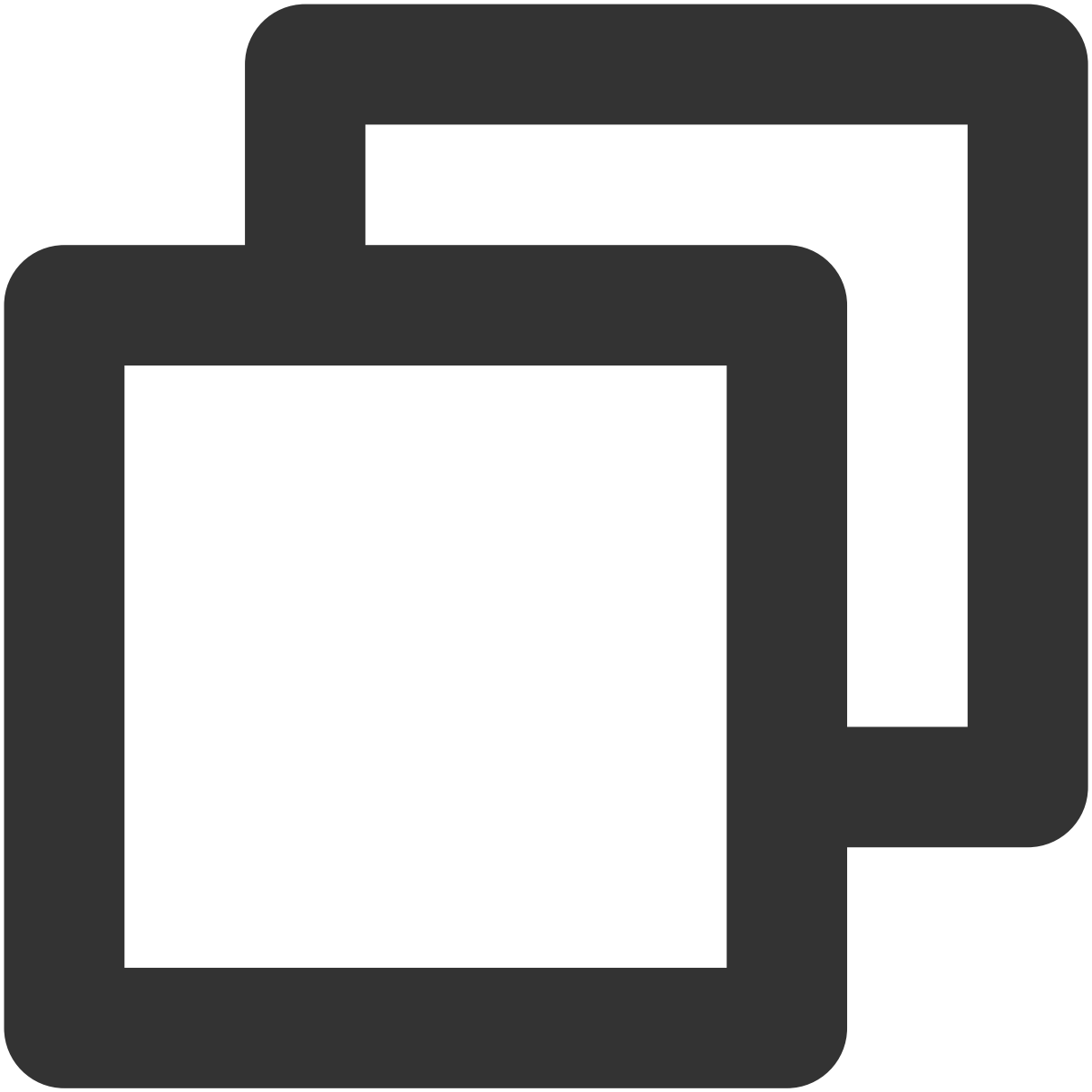
开启/关闭静音模式。

Objective-C

Swift



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



```
public func enableMuteMode(enable: Bool)
```

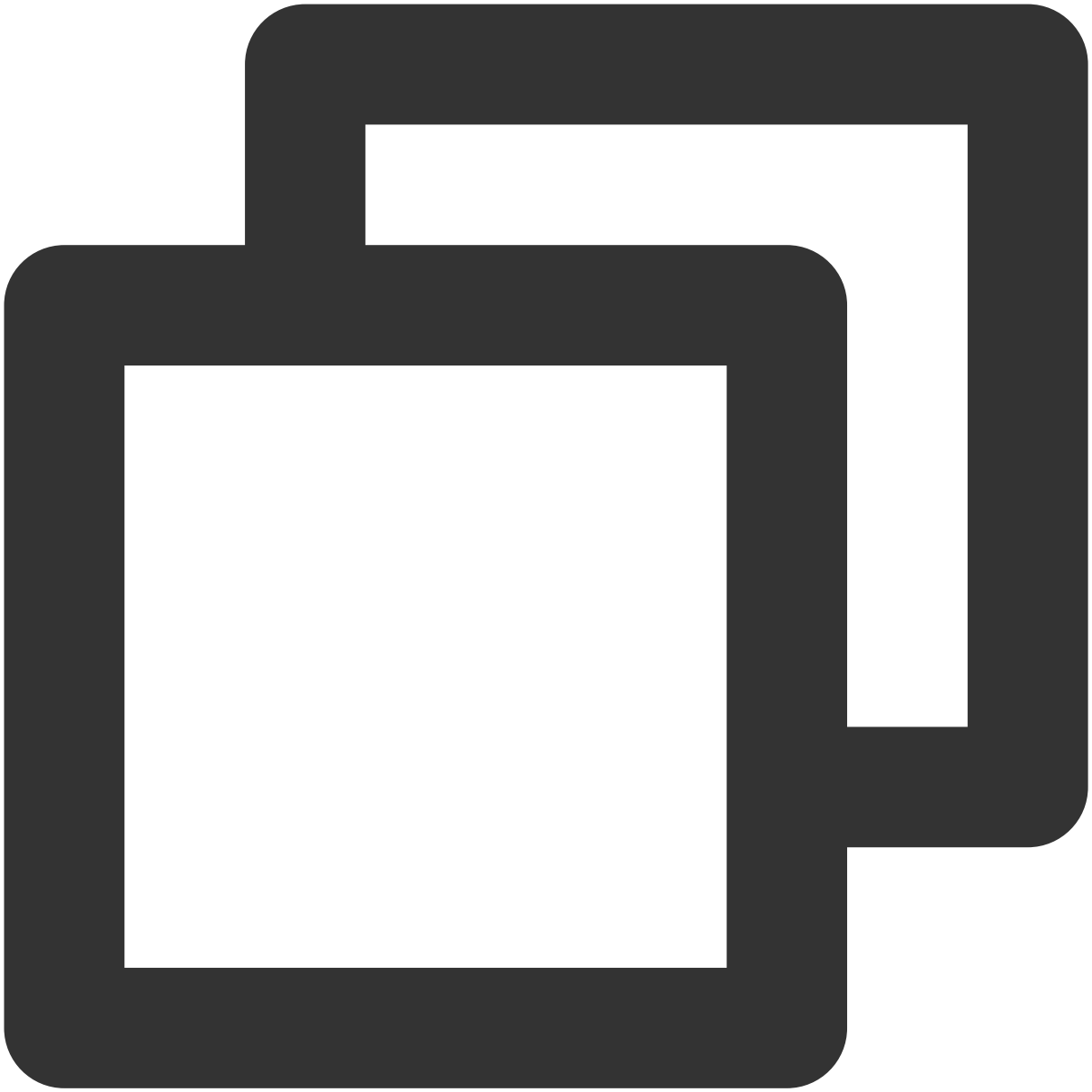
enableFloatWindow

开启/关闭悬浮窗功能。

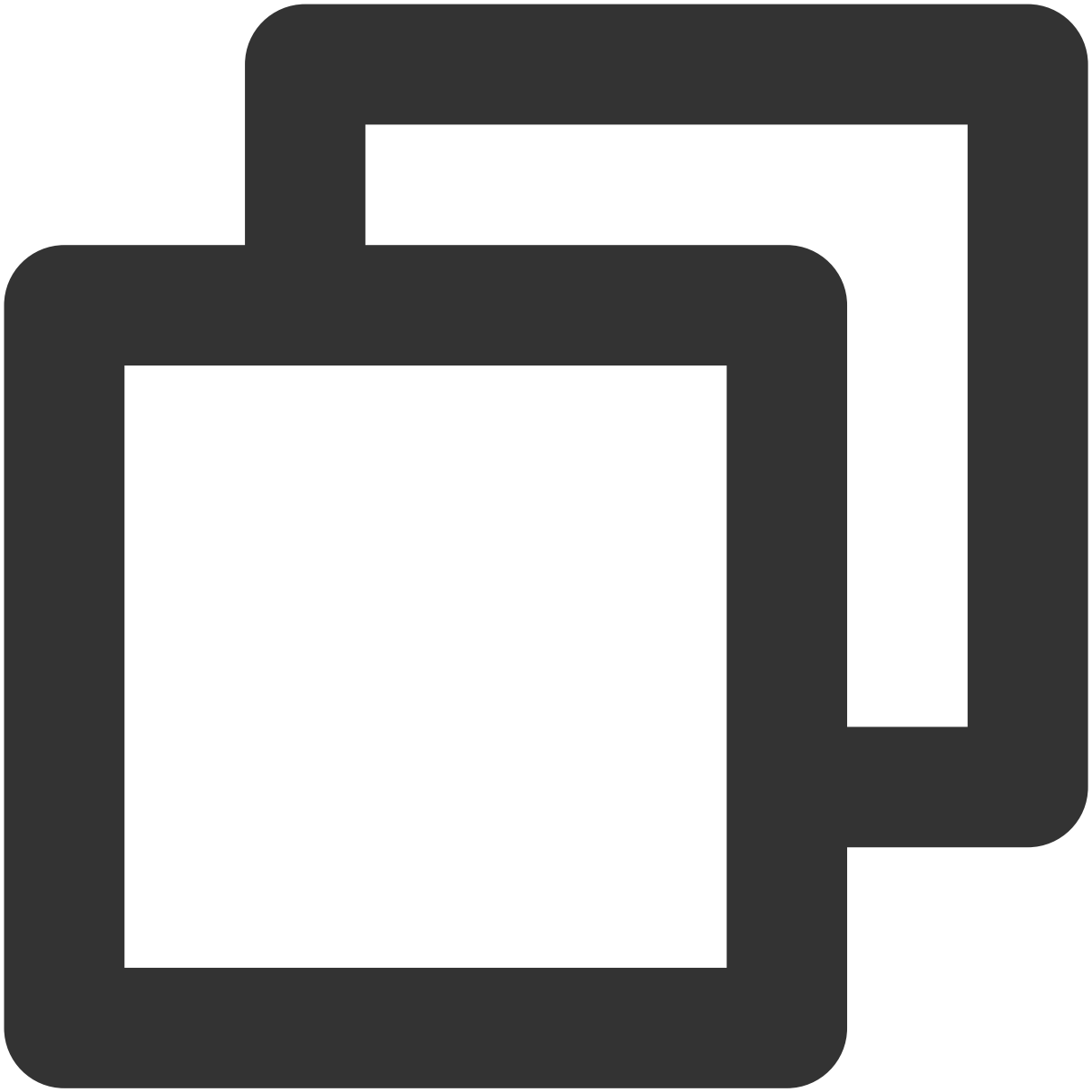
默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

Objective-C

Swift



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

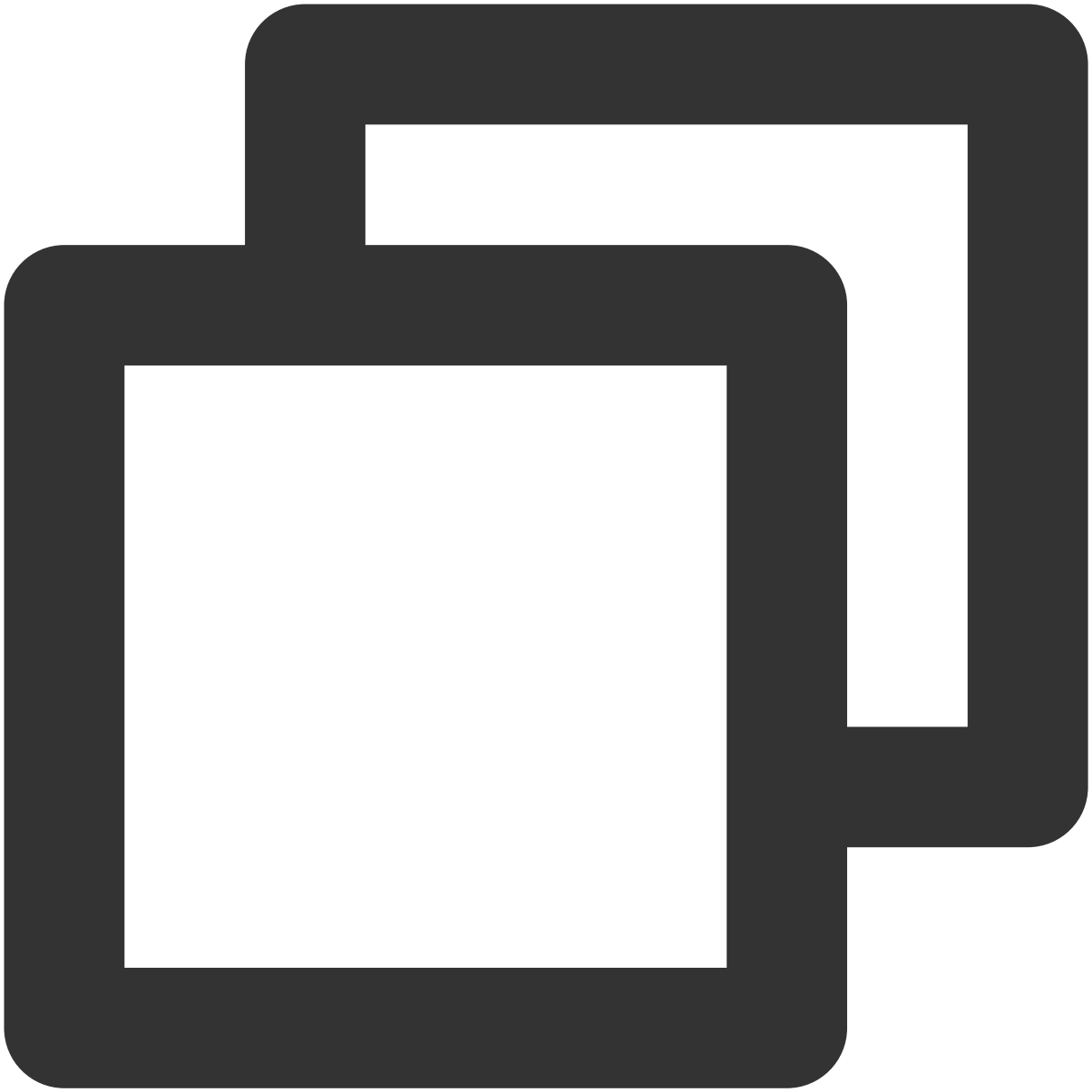


```
public func enableFloatWindow(enable: Bool)
```

enableIncomingBanner

开启/关闭来电横幅显示。

默认为 `false` ，被叫端收到邀请后默认弹出全屏通话等待界面，开启后先展示一个横幅，然后根据需要拉起全屏通话界面。



```
public func enableIncomingBanner(enable: Bool)
```

TUICallEngine

最近更新时间：2024-07-30 11:23:09

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流

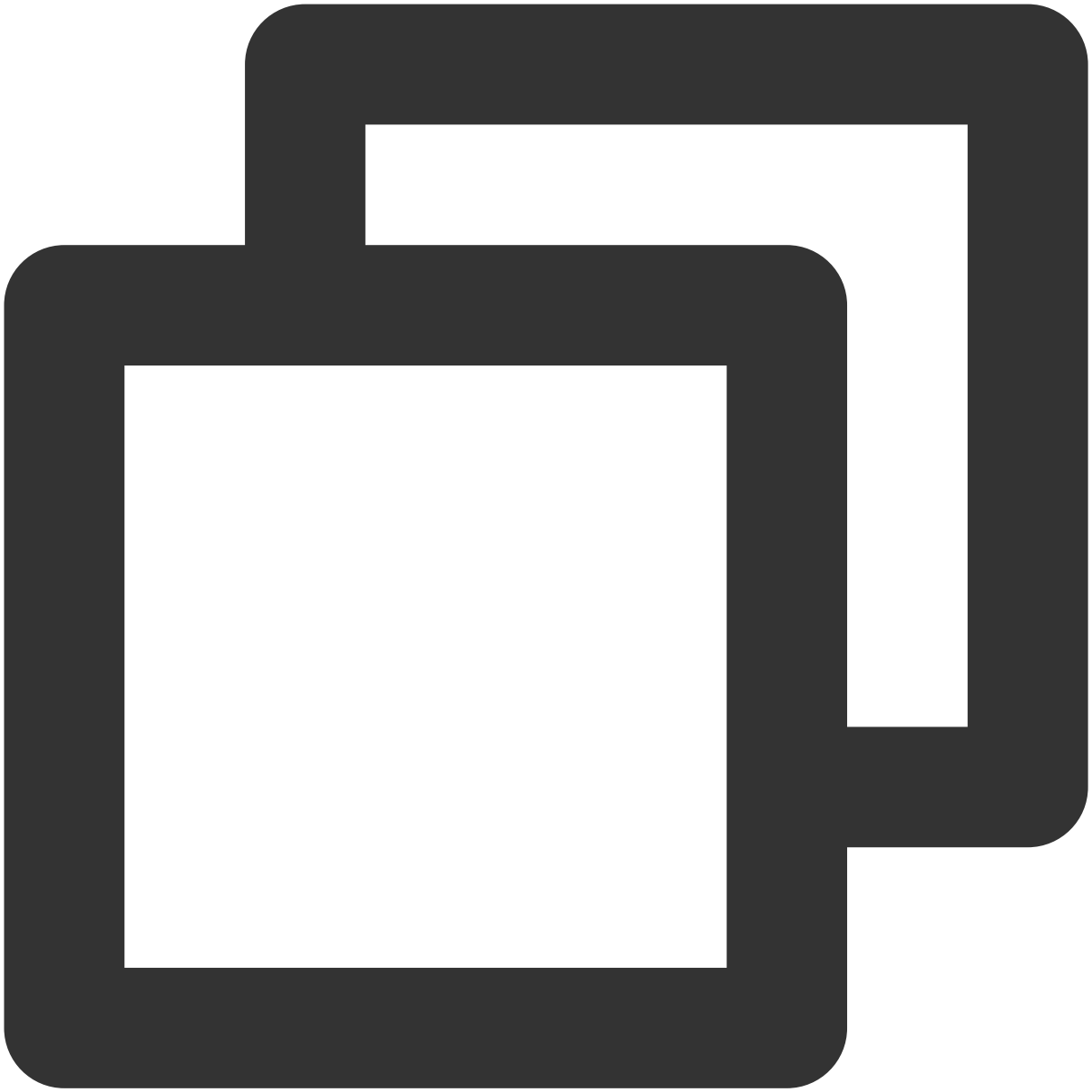
版权所有：腾讯云计算（北京）有限责任公司

openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式 （群组通话版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

API 详情

createInstance

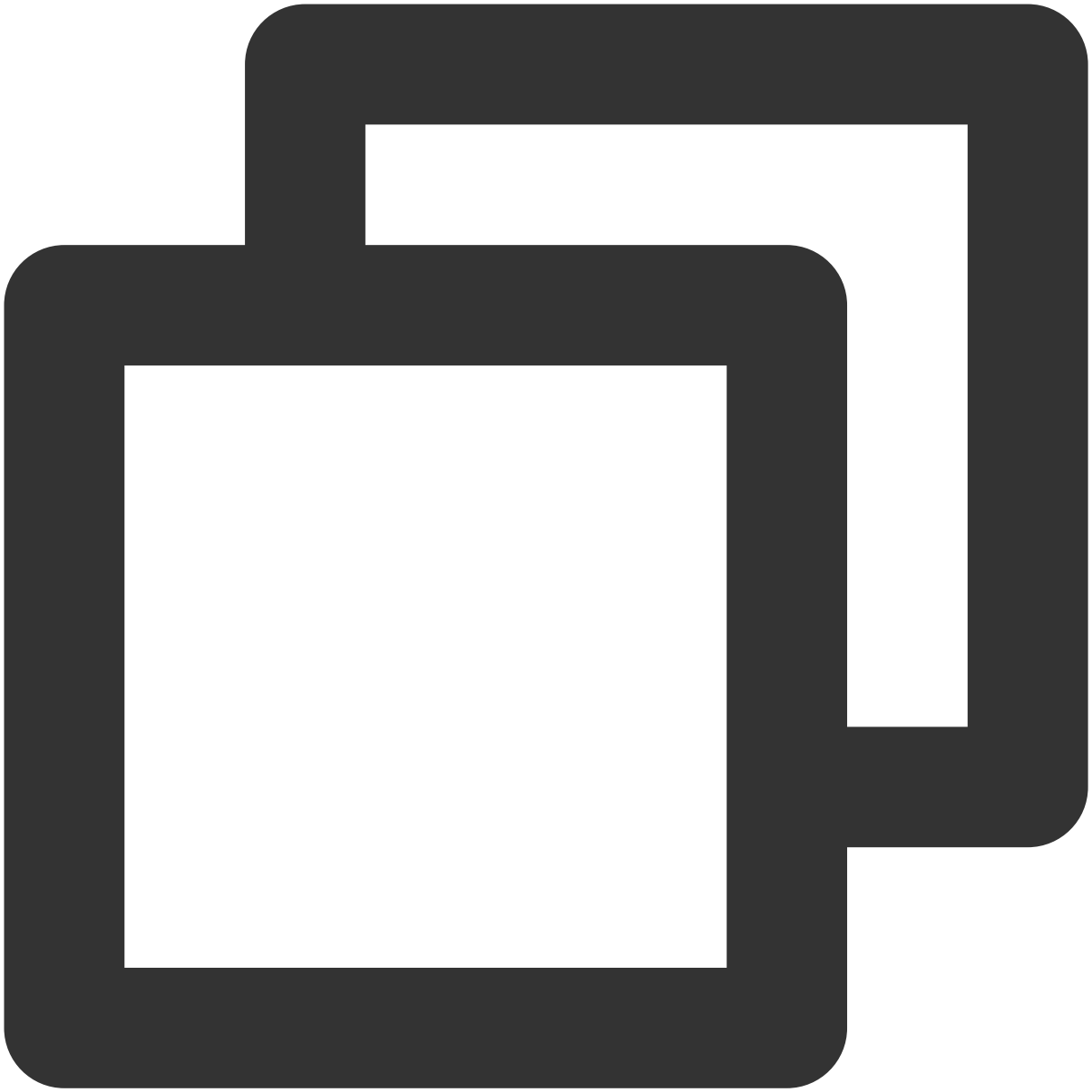
创建 TUICallEngine 的单例。



```
- (TUICallEngine *)createInstance;
```

destroyInstance

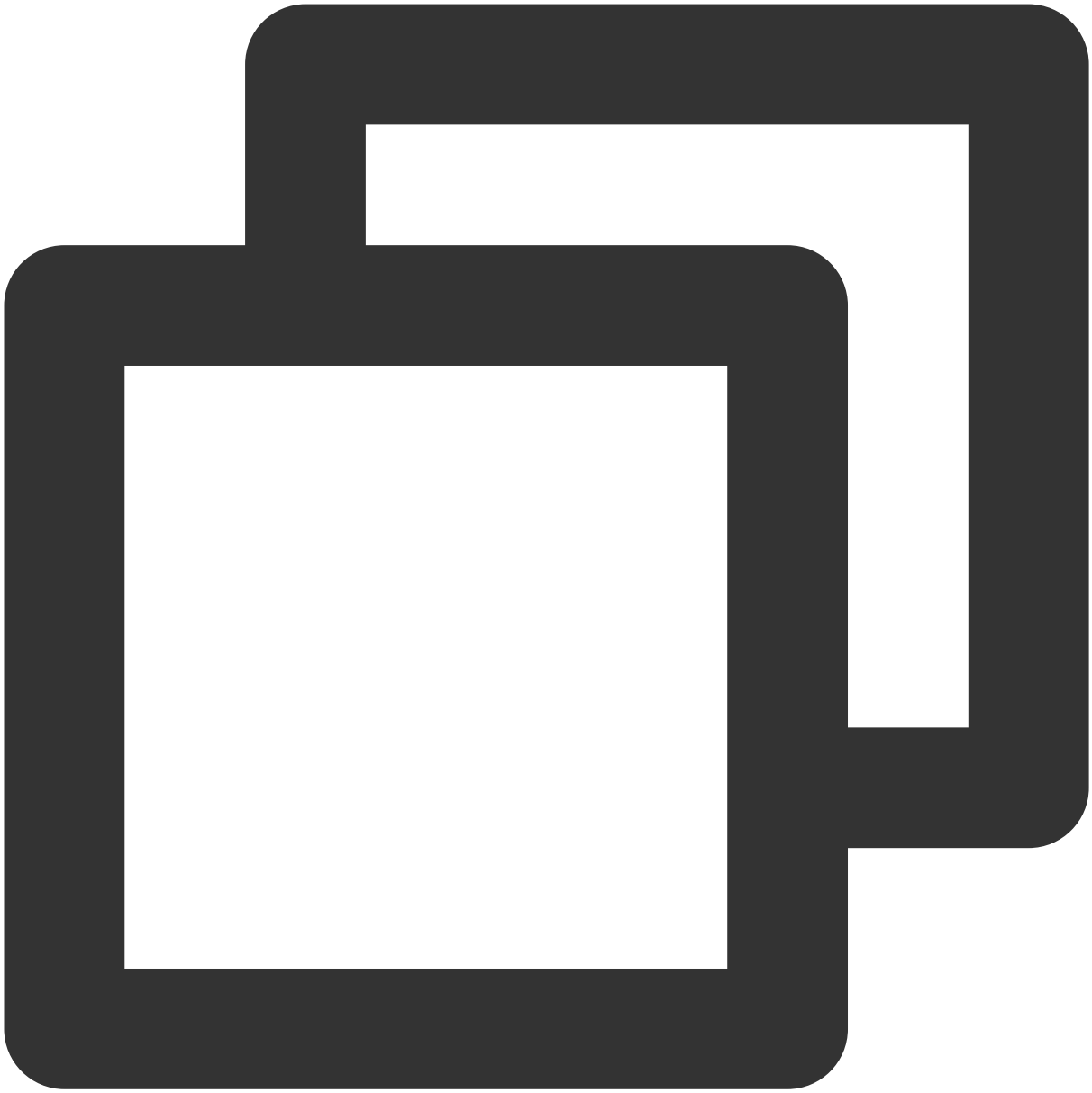
销毁 TUICallEngine 的单例。



```
- (void)destroyInstance;
```

init

初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。



```
- (void)init:(NSString *)sdkAppID userId:(NSString *)userId userSig:(NSString *)use
```

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

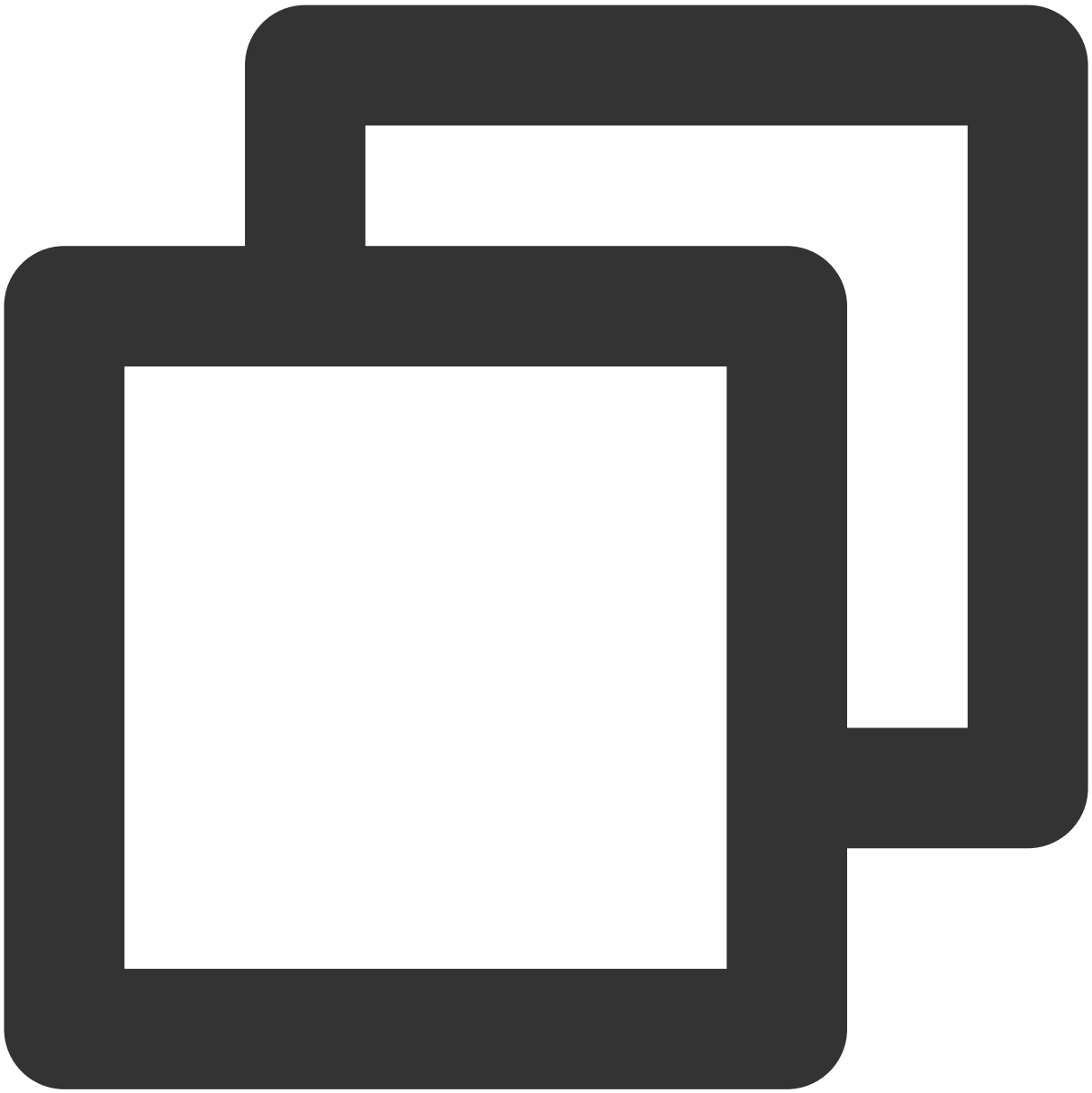
addObserver

添加回调接口，您可以通过这个接听，监听 TUICallObserver 相关的事件回调。

```
- (void)addObserver:(id<TUICallObserver>)observer;
```

removeObserver

移除回调接口。



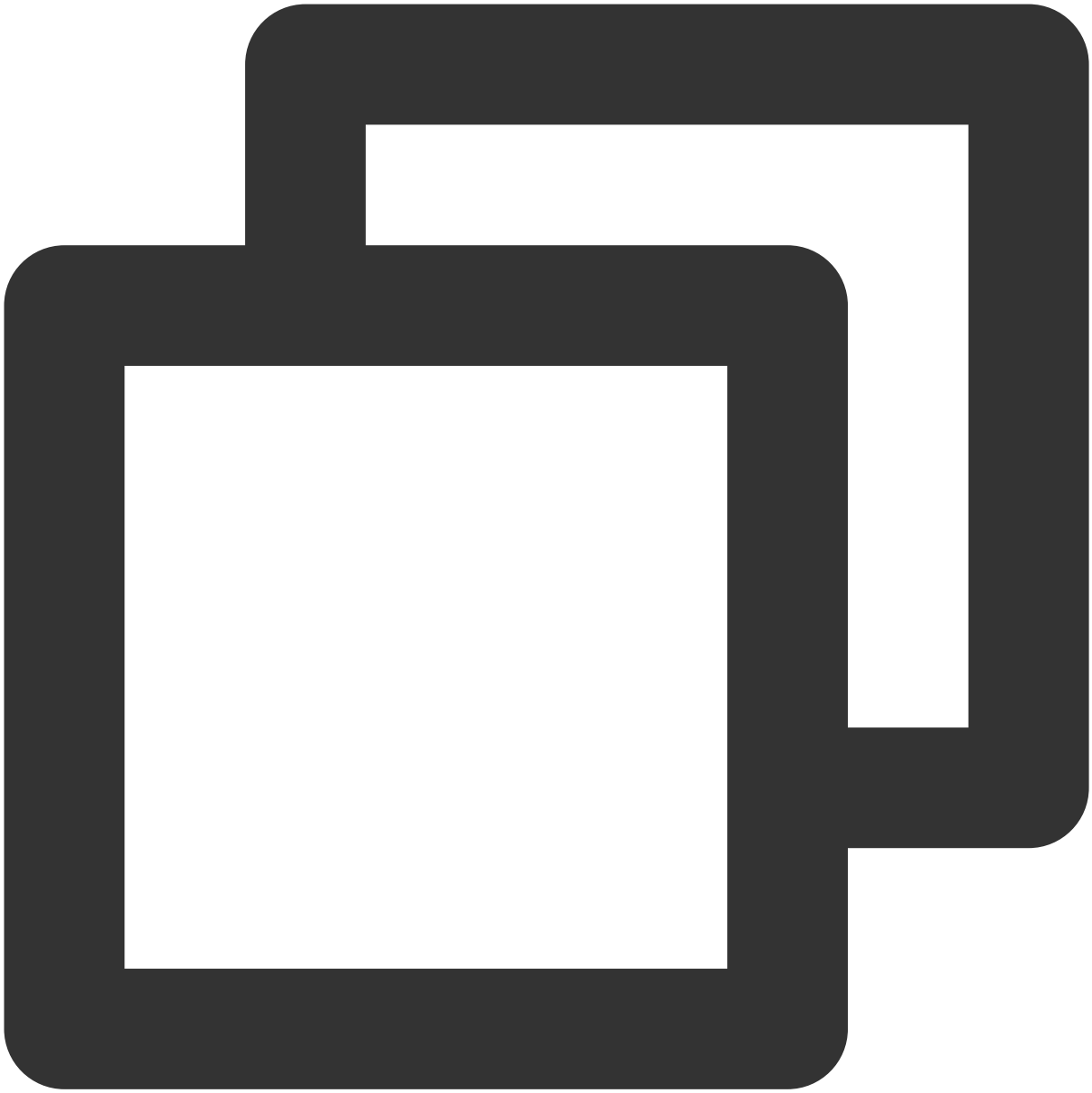
```
- (void)removeObserver:(id<TUICallObserver>)observer;
```

call

拨打电话（1v1通话）。

注意：

字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！



```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType param
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线

		推送自定义内容等

groupCall

发起群组通话。

注意：

使用群组通话前需要创建 IM 群组，如果已经创建，请忽略；

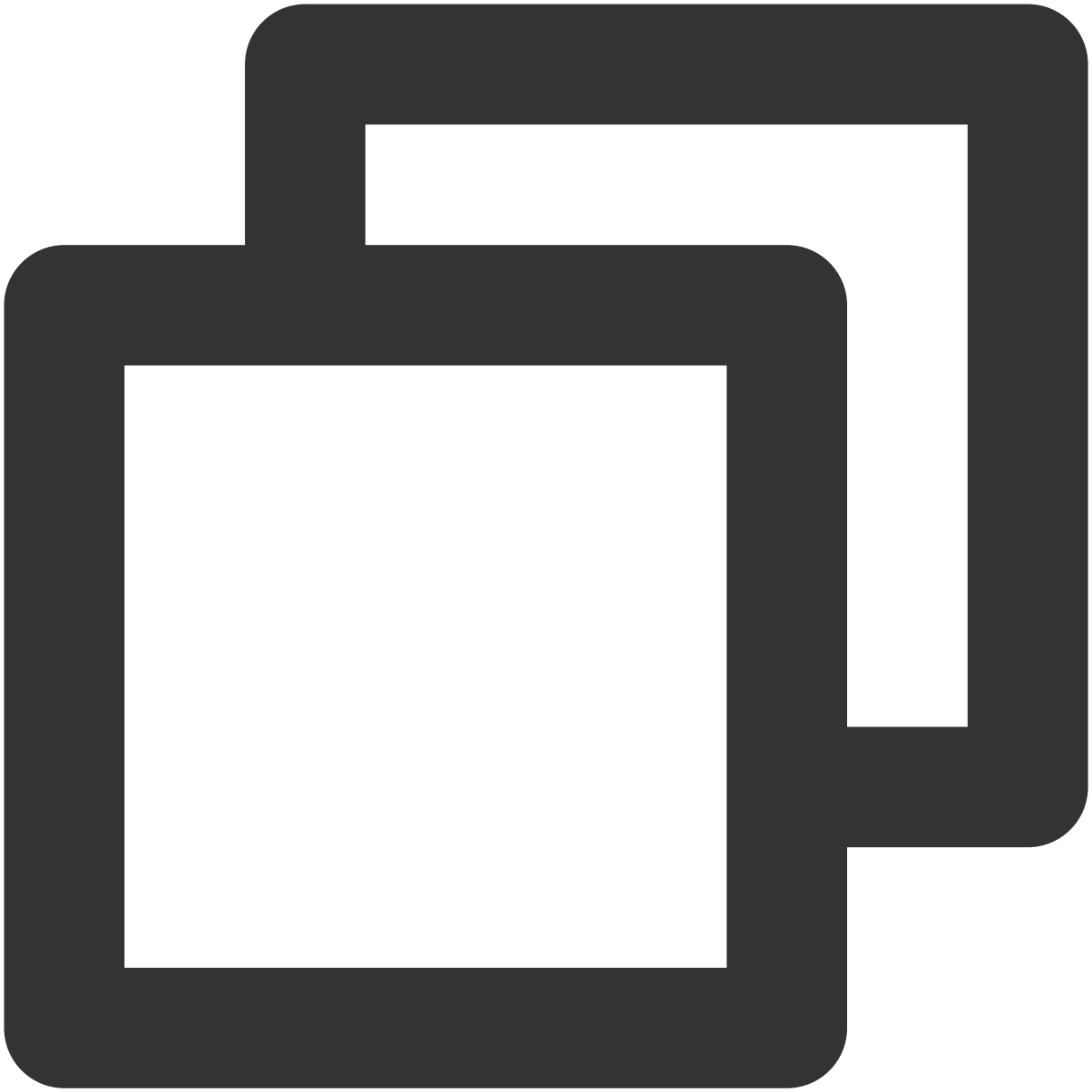
字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！

```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray <NSString *> *)userIdList
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

accept

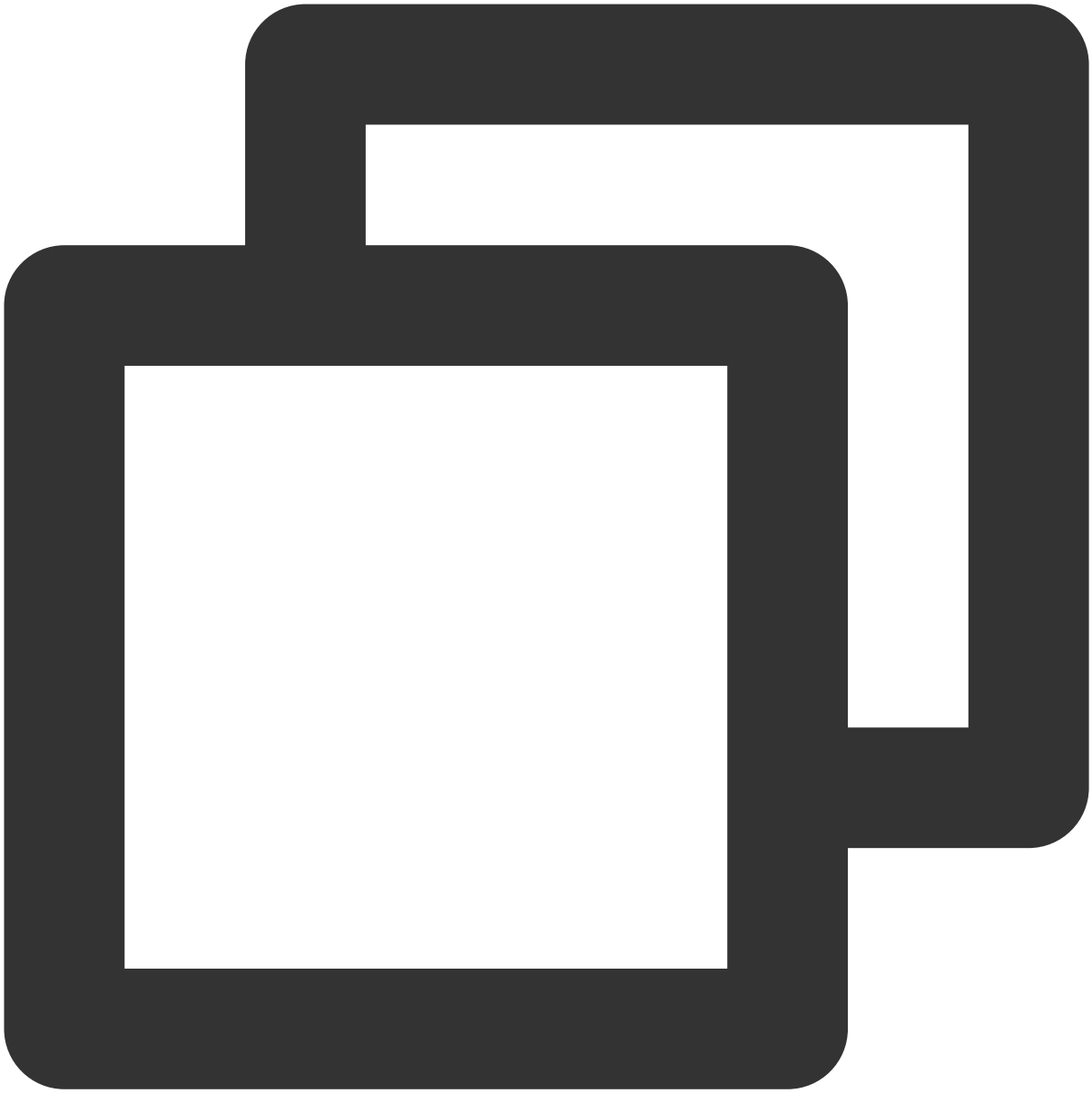
接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。



```
- (void)accept:(TUICallSucc)succ fail:(TUICallFail)fail;
```

reject

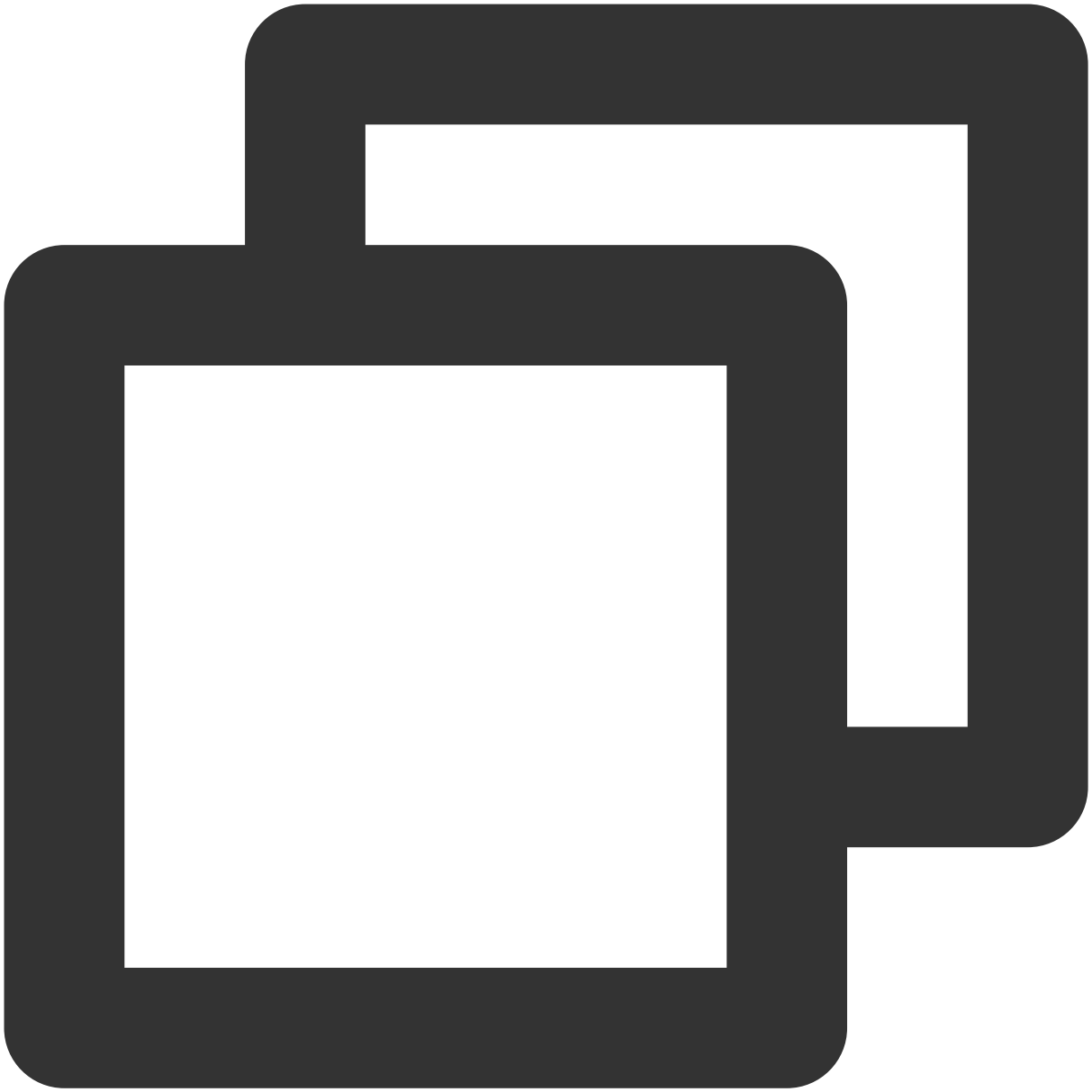
拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。



```
- (void)reject:(TUICallSucc)succ fail:(TUICallFail)fail;
```

ignore

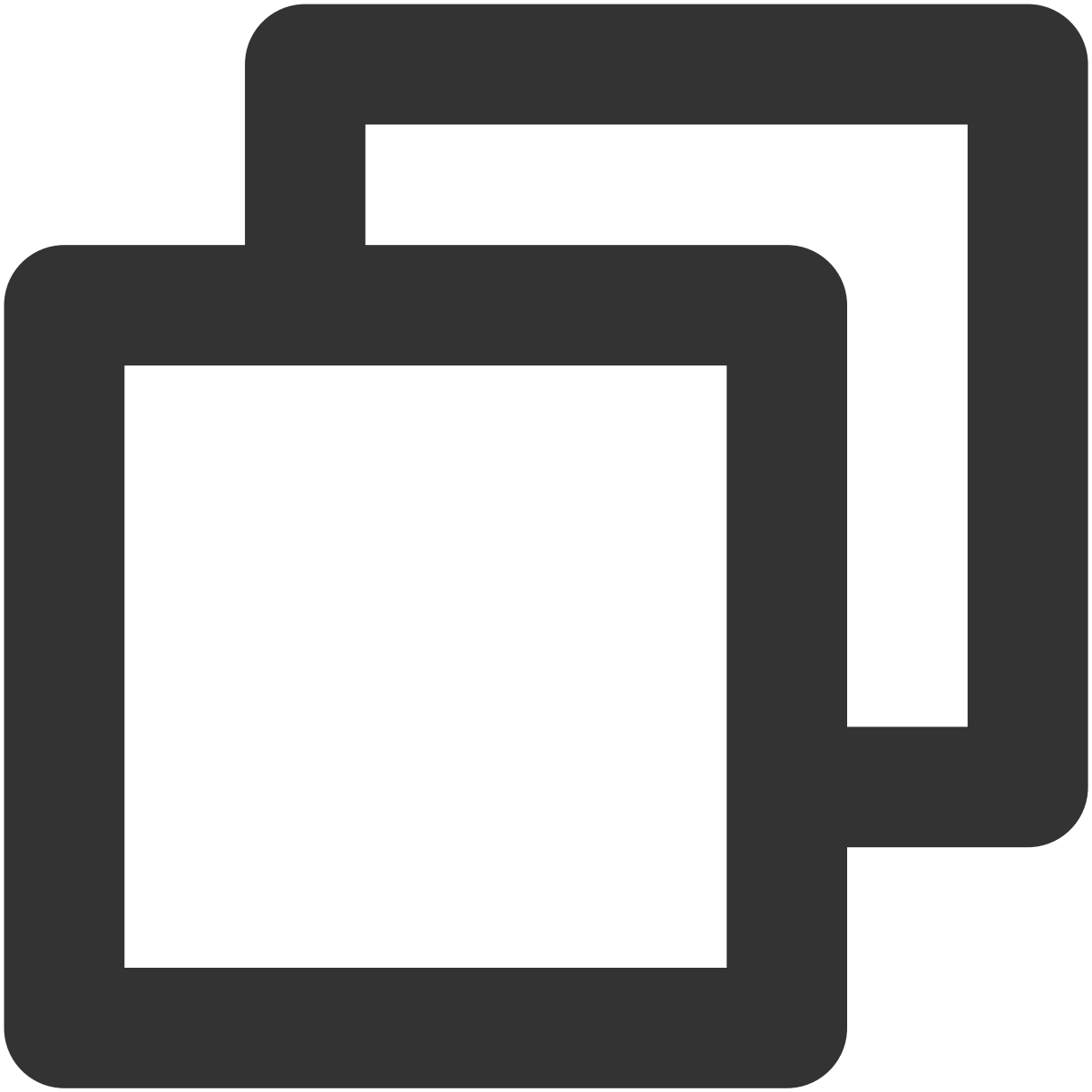
忽略当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数忽略来电，此时主叫会收到 `onUserLineBusy` 的回调。
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。



```
- (void)ignore:(TUICallSucc)succ fail:(TUICallFail)fail;
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

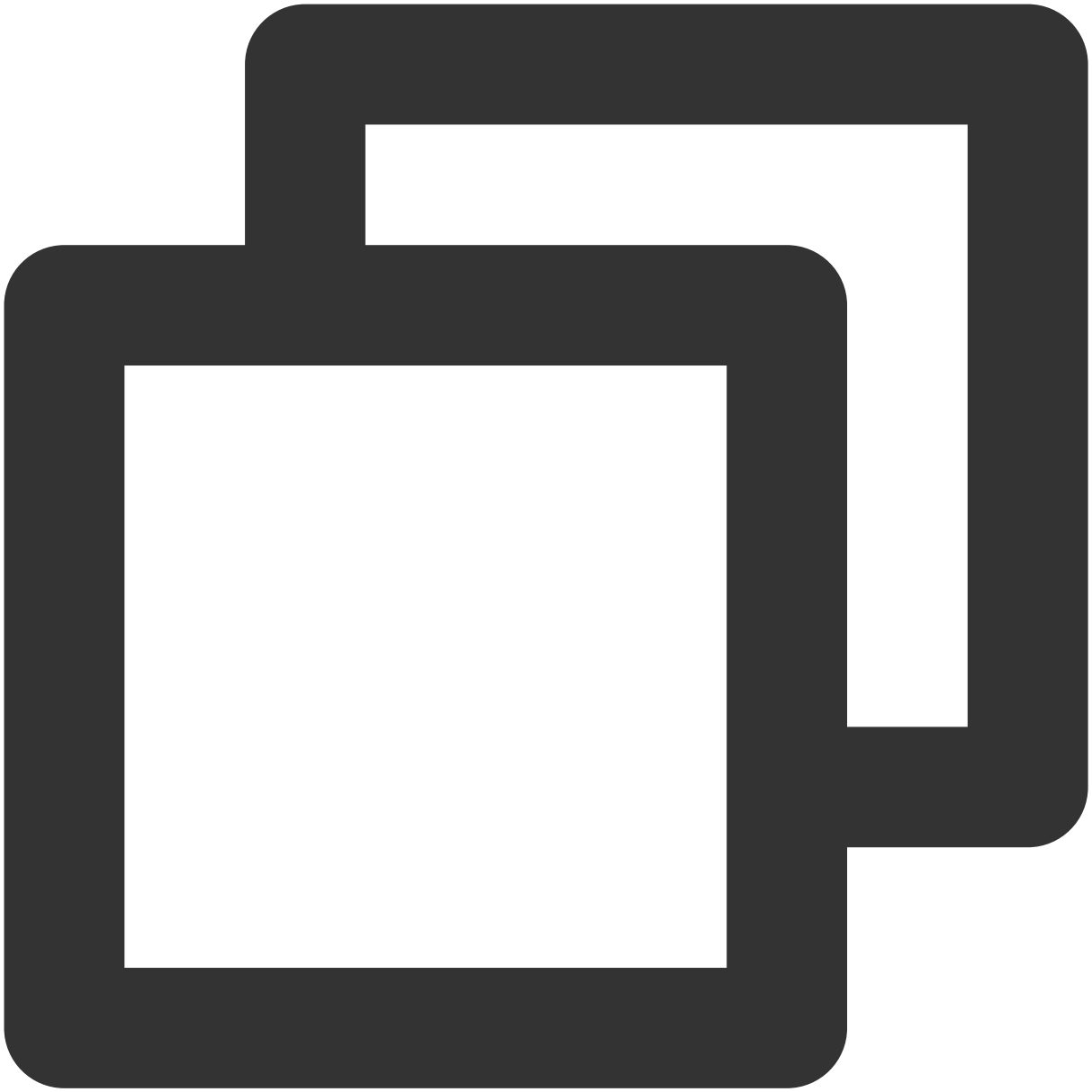


```
- (void)hangup:(TUICallSucc)succ fail:(TUICallFail)fail;
```

inviteUser

邀请用户加入此次群组通话。

使用场景：一个群组通话中的用户主动邀请其他人时使用。



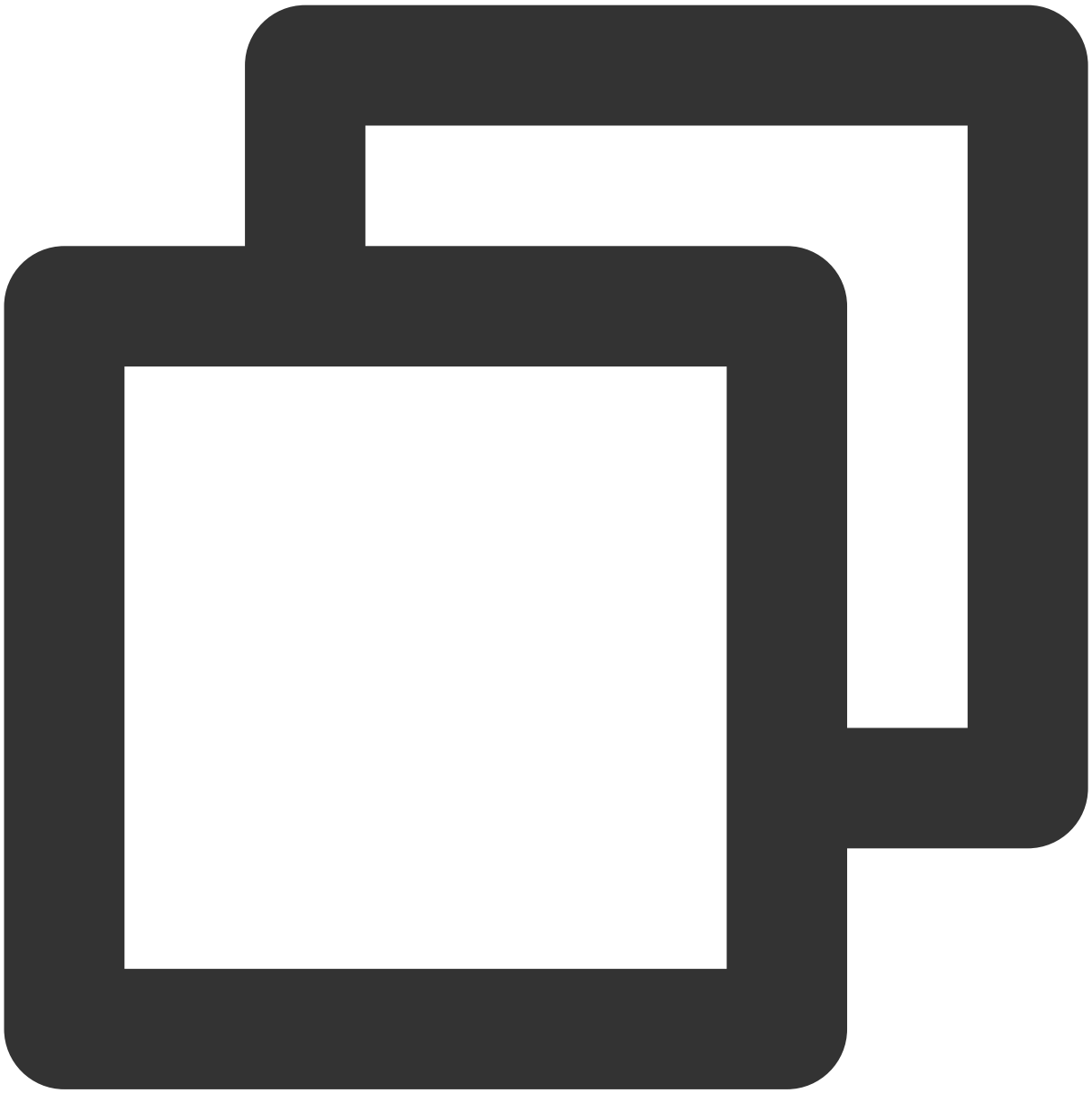
```
- (void)inviteUser:(NSArray<NSString *> *)userIdList params:(TUICallParams *)params
```

参数	类型	含义
userIdList	NSArray	目标用户的 userId 列表
params	TUICallParams	通话参数扩展字段，例如：离线推送自定义内容

joinInGroupCall

主动加入此次群组通话。

使用场景：群组内用户主动加入此次群组通话使用。



```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
groupId	NSString	此次群组通话的群 ID

callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

switchCallMediaType

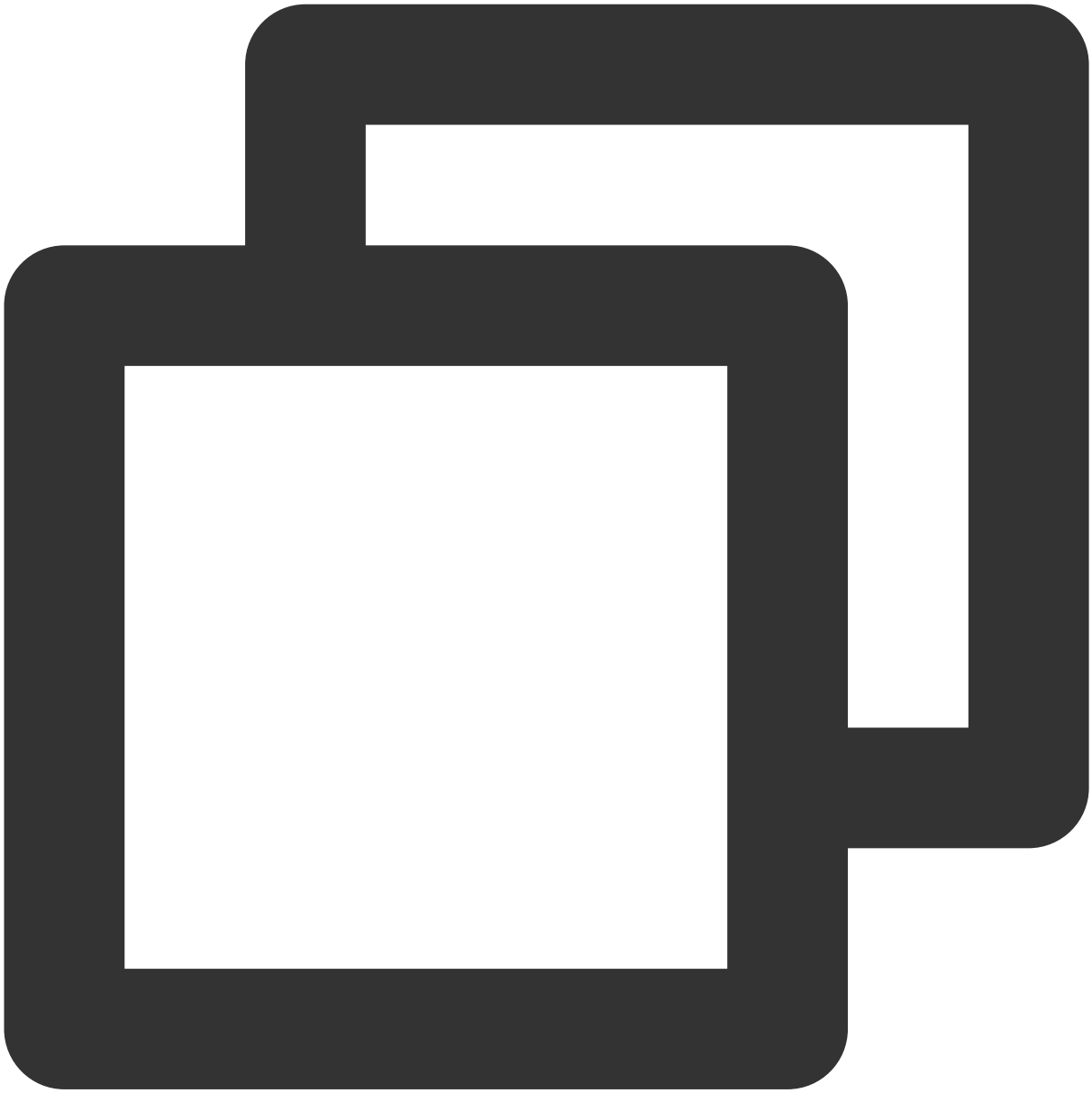
切换视频通话到语音通话。

```
- (void)switchCallMediaType:(TUICallMediaType)newType;
```

参数	类型	含义
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

startRemoteView

设置显示视频画面的 View 对象。



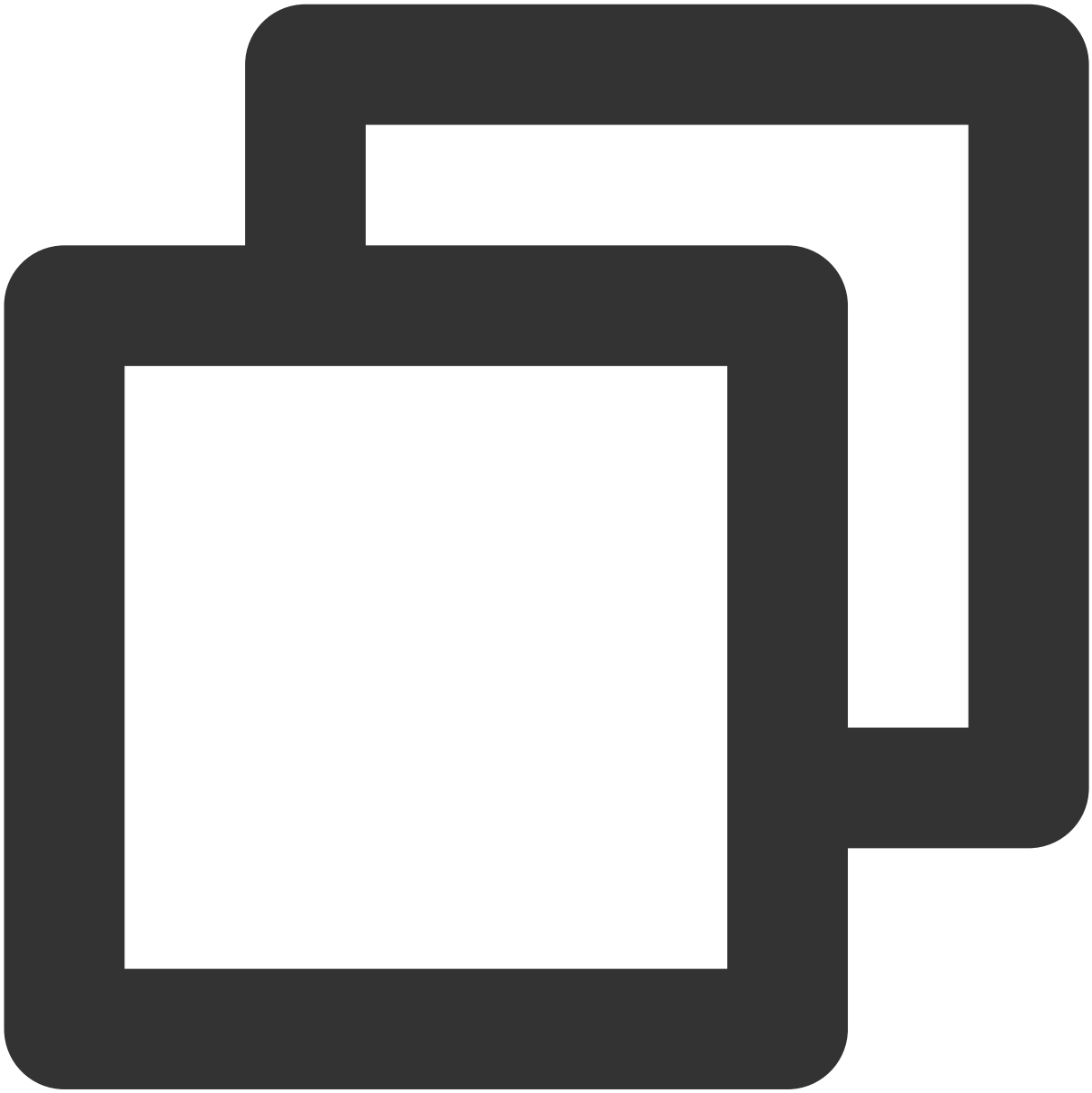
```
- (void)startRemoteView:(NSString *)userId videoView:(TUIVideoView *)videoView onPl
```

参数	类型	含义
userId	NSString	目标用户的 userId

videoView	TUIVideoView	待渲染的视图

stopRemoteview

停止订阅远端用户的视频数据。

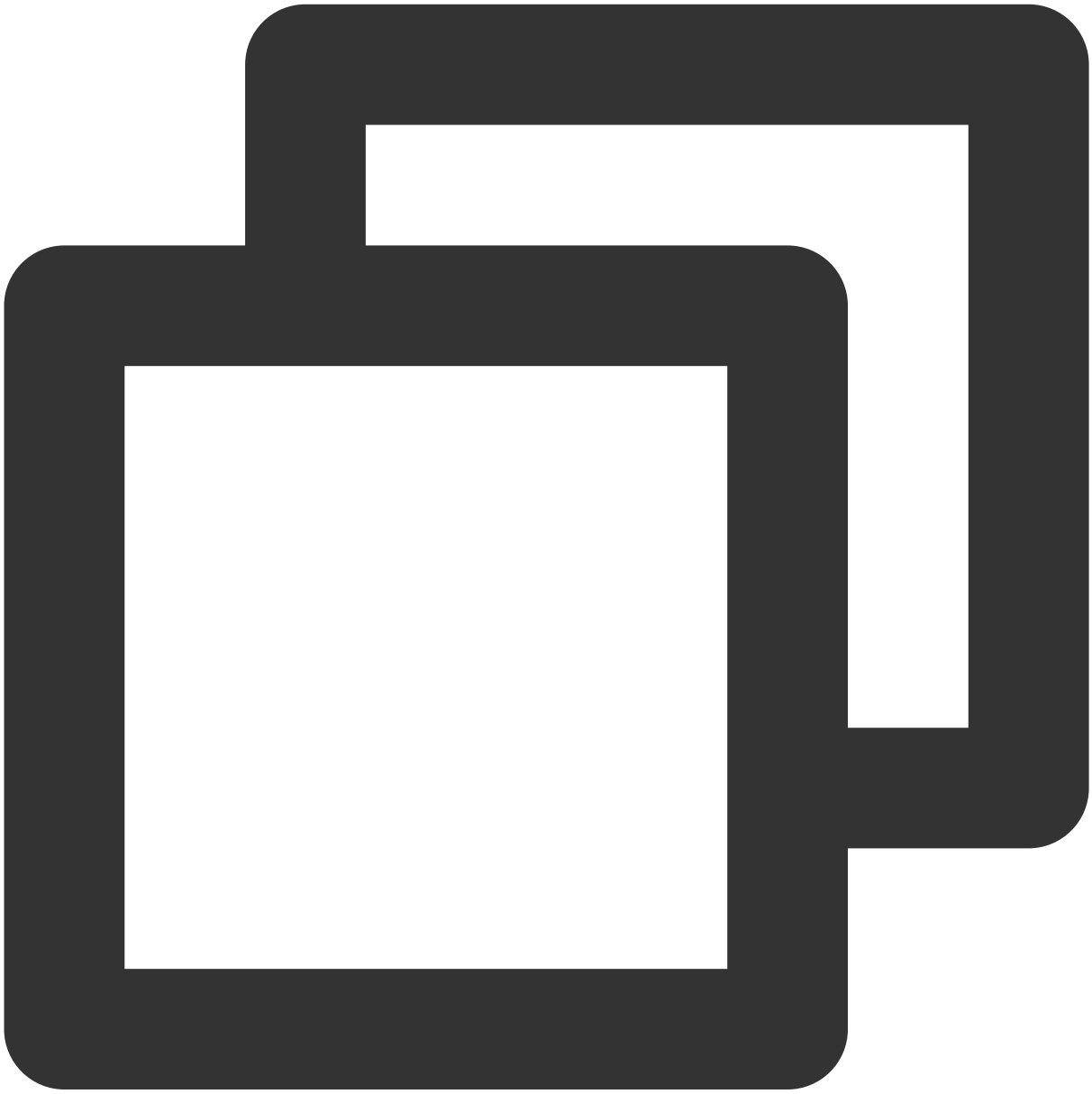


```
- (void)stopRemoteView:(NSString *)userId;
```

参数	类型	含义
userId	NSString	目标用户的 userId

openCamera

开启摄像头。

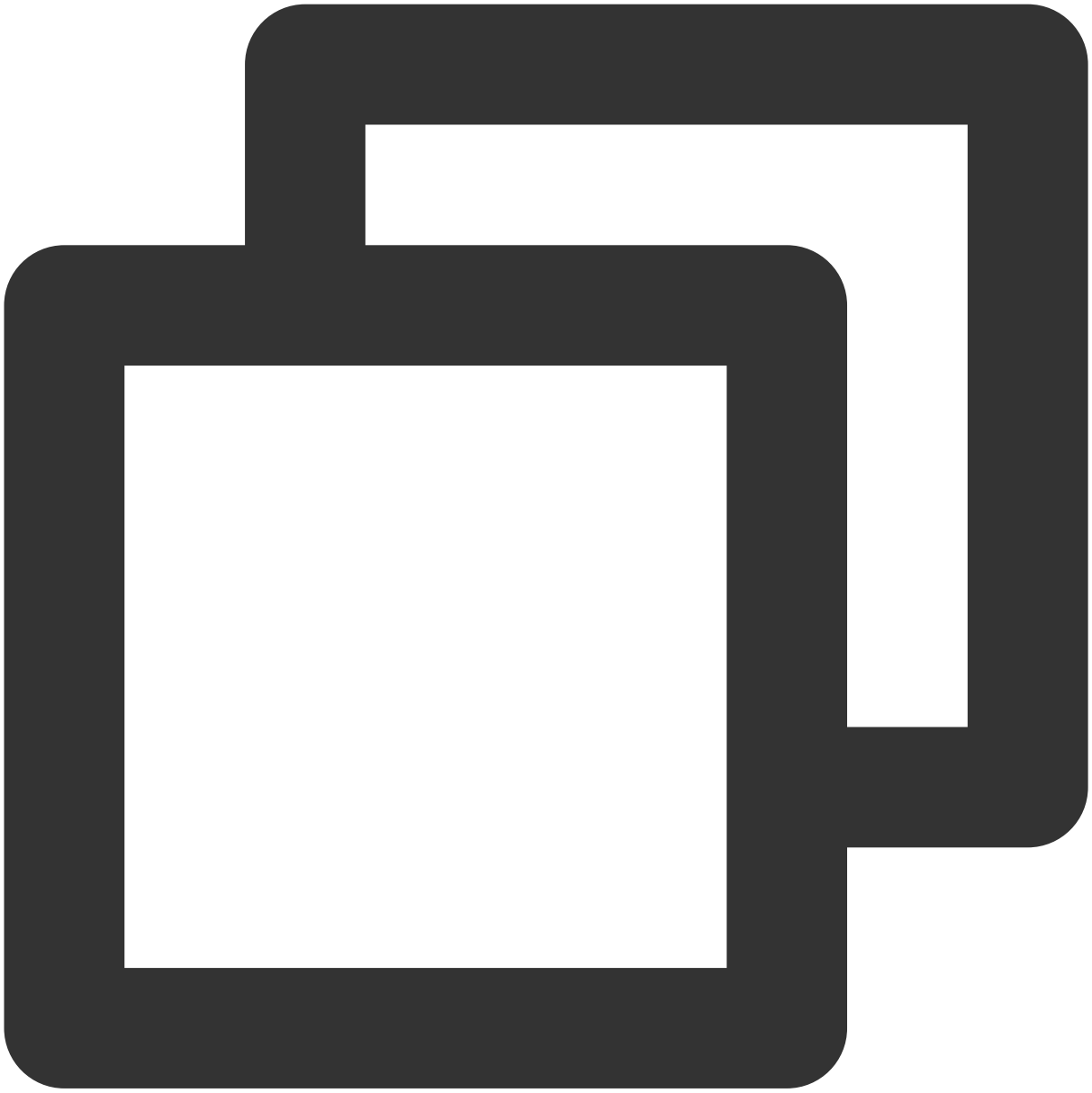


```
- (void)openCamera:(TUICamera)camera videoView:(TUIVideoView *)videoView succ:(TUIC
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头
videoView	TUIVideoView	待渲染的视图

closeCamera

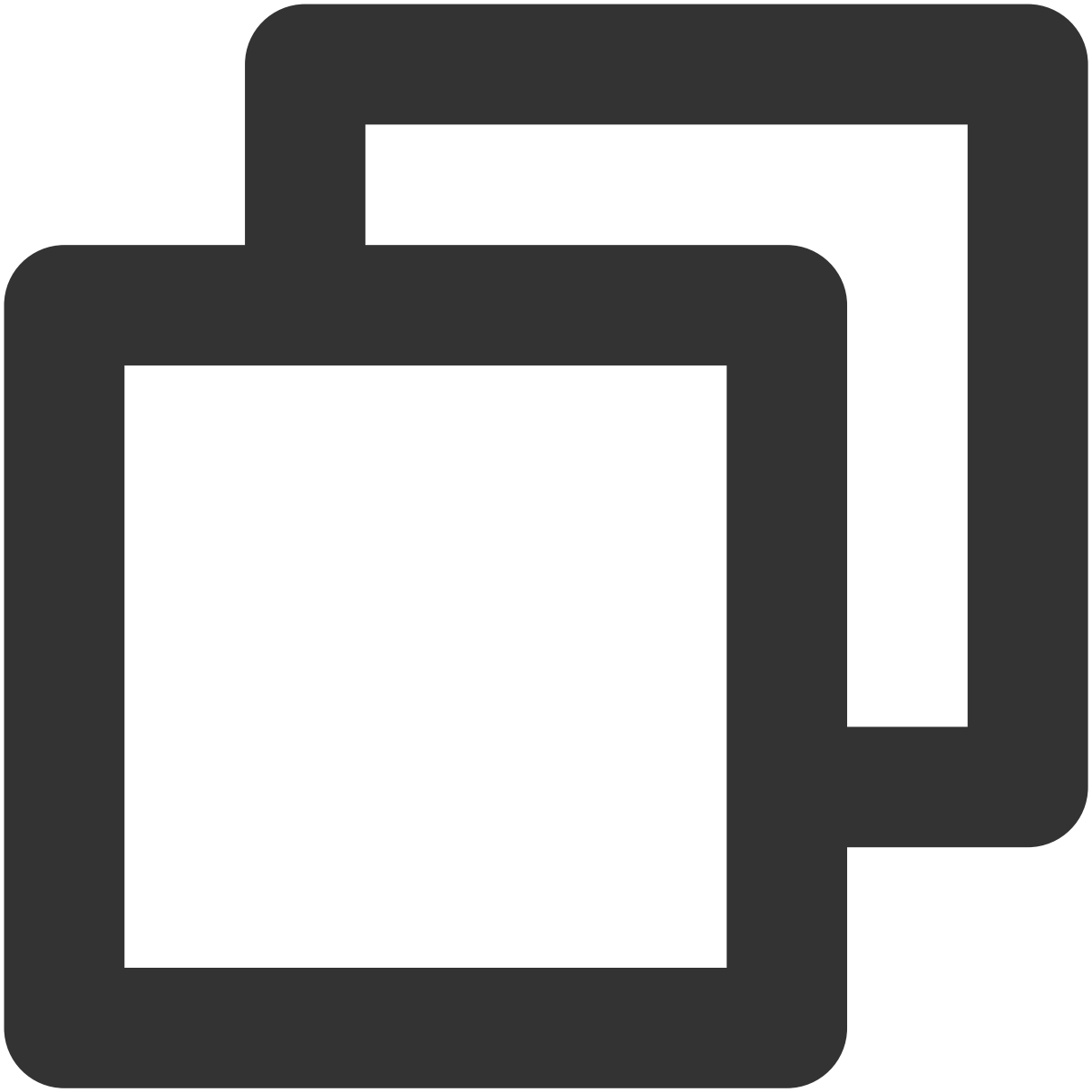
关闭摄像头。



```
- (void)closeCamera;
```

switchCamera

切换前后摄像头。

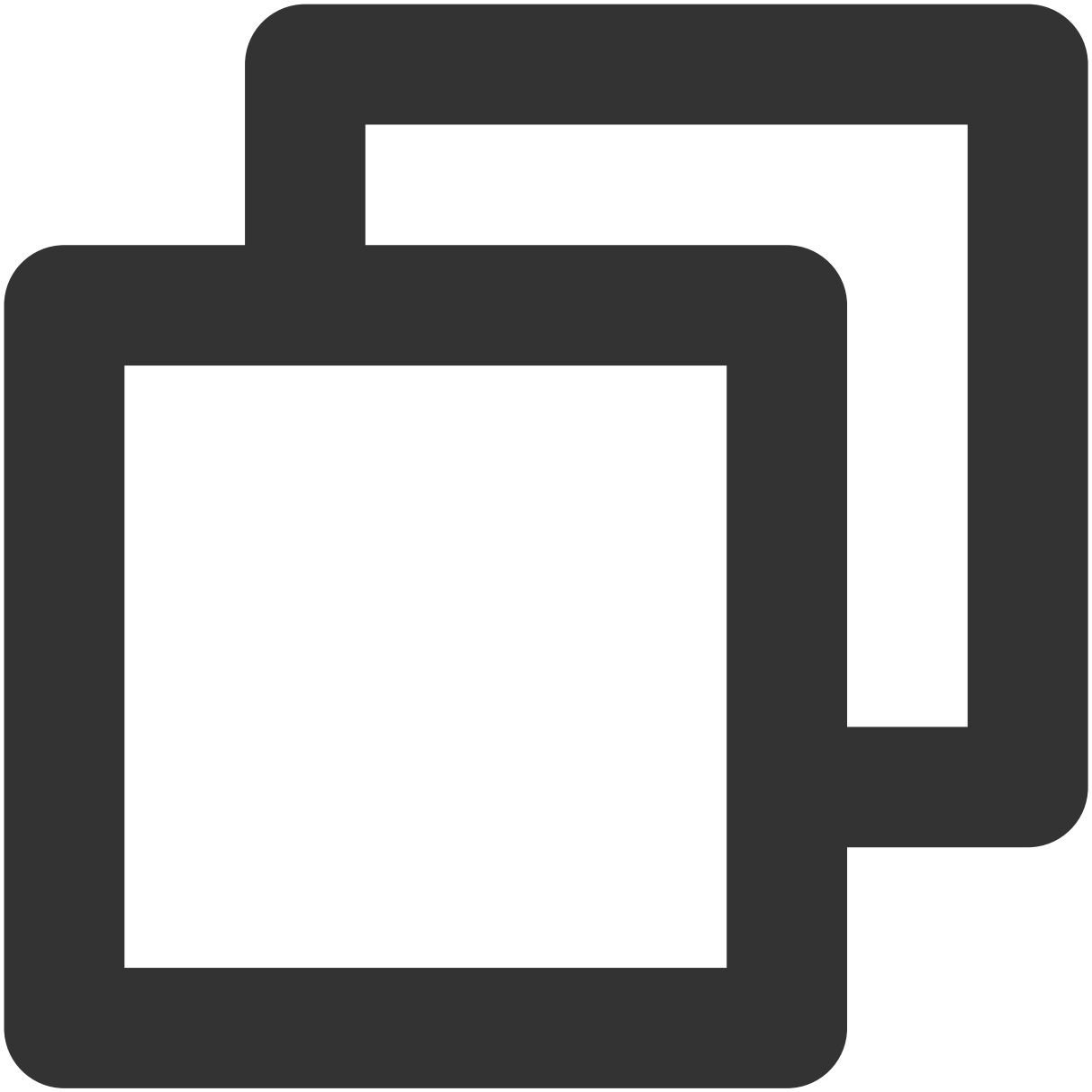


```
- (void)switchCamera:(TUICamera)camera;
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头

openMicrophone

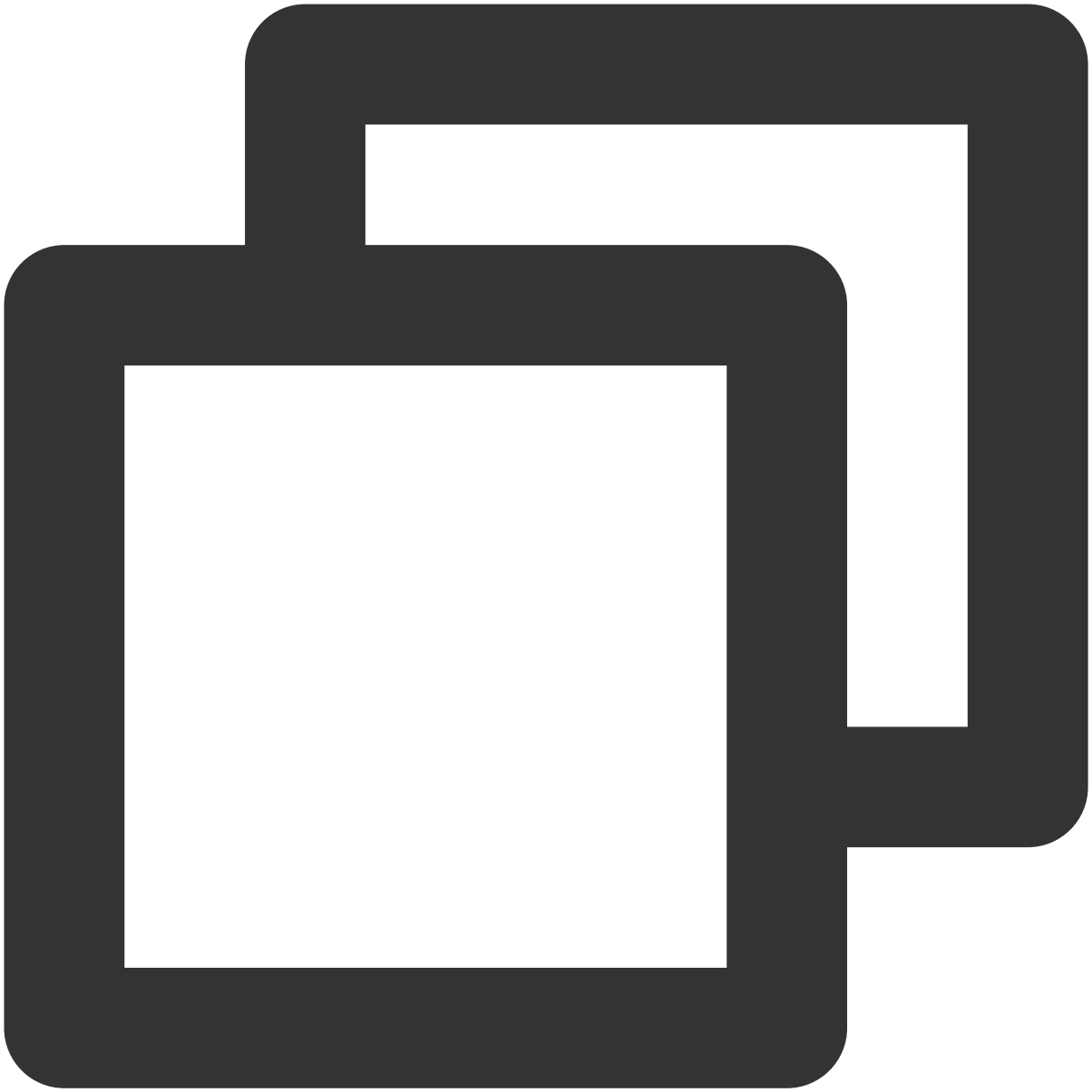
打开麦克风。



```
- (void)openMicrophone:(TUICallSucc)succ fail:(TUICallFail)fail;
```

closeMicrophone

关闭麦克风。

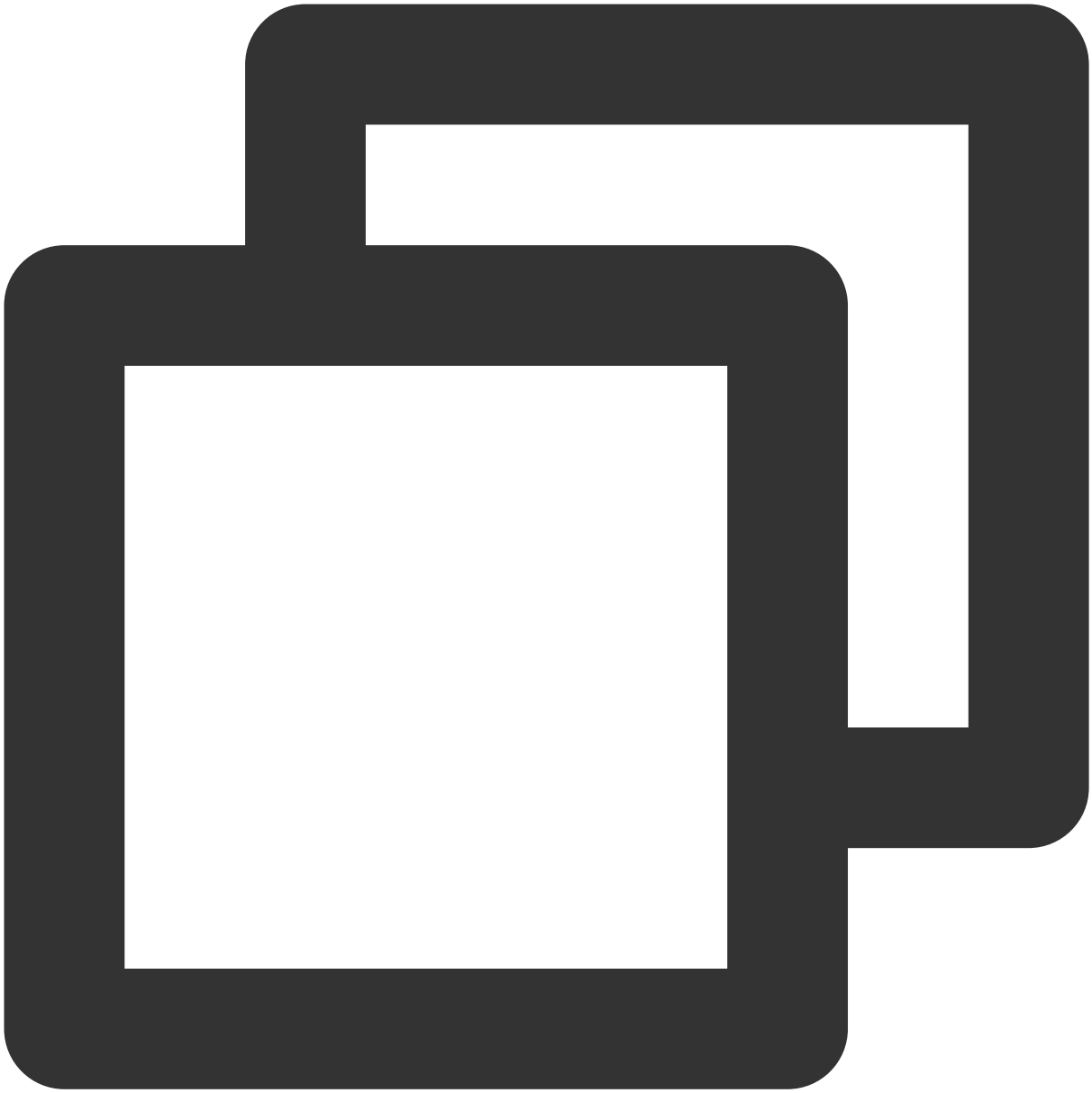


```
- (void)closeMicrophone;
```

selectAudioPlaybackDevice

选择音频播放设备。

目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。



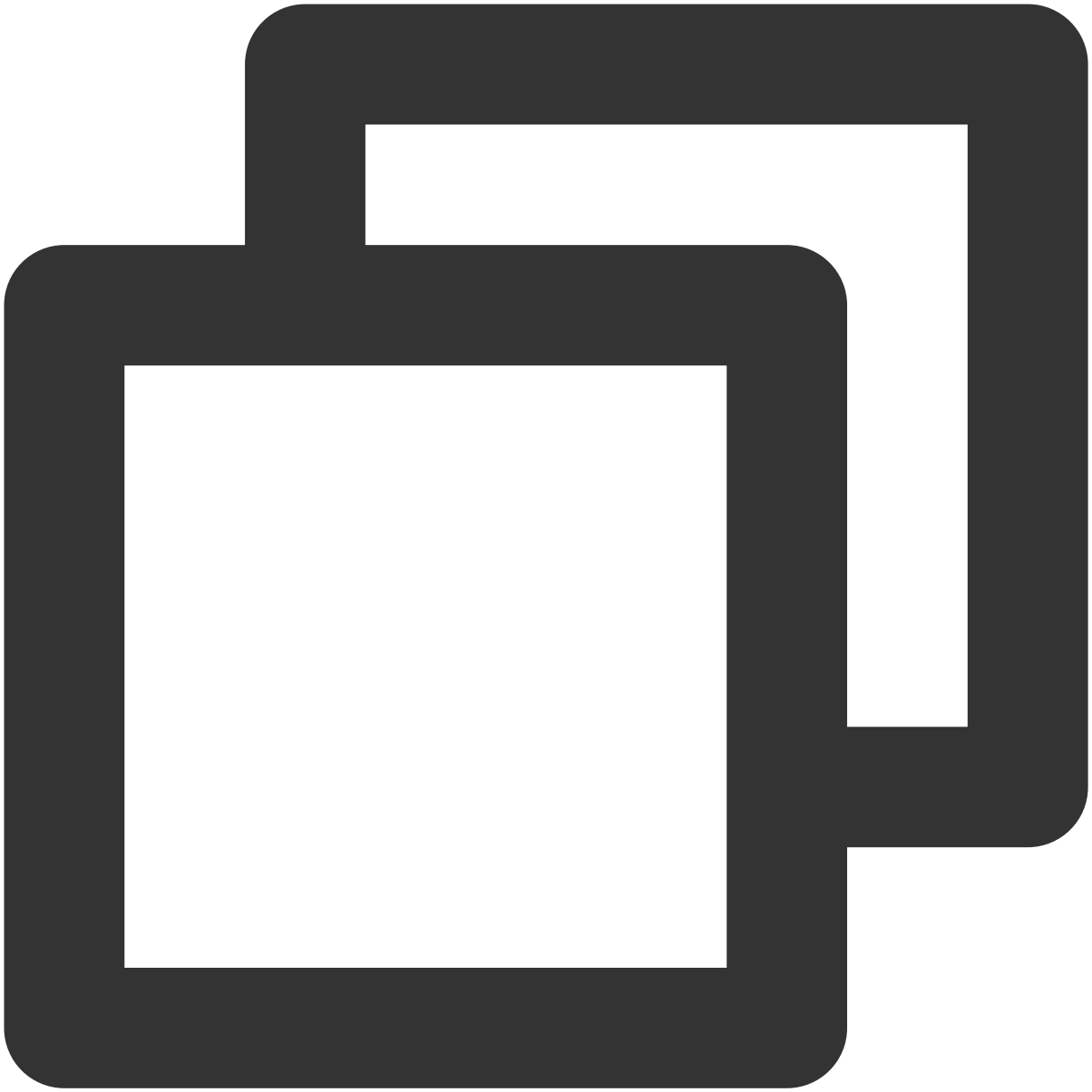
```
- (void)selectAudioPlaybackDevice:(TUIAudioPlaybackDevice)device;
```

参数	类型	含义
device	TUIAudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。

用户昵称不能超过500字节，用户头像必须是 URL 格式。

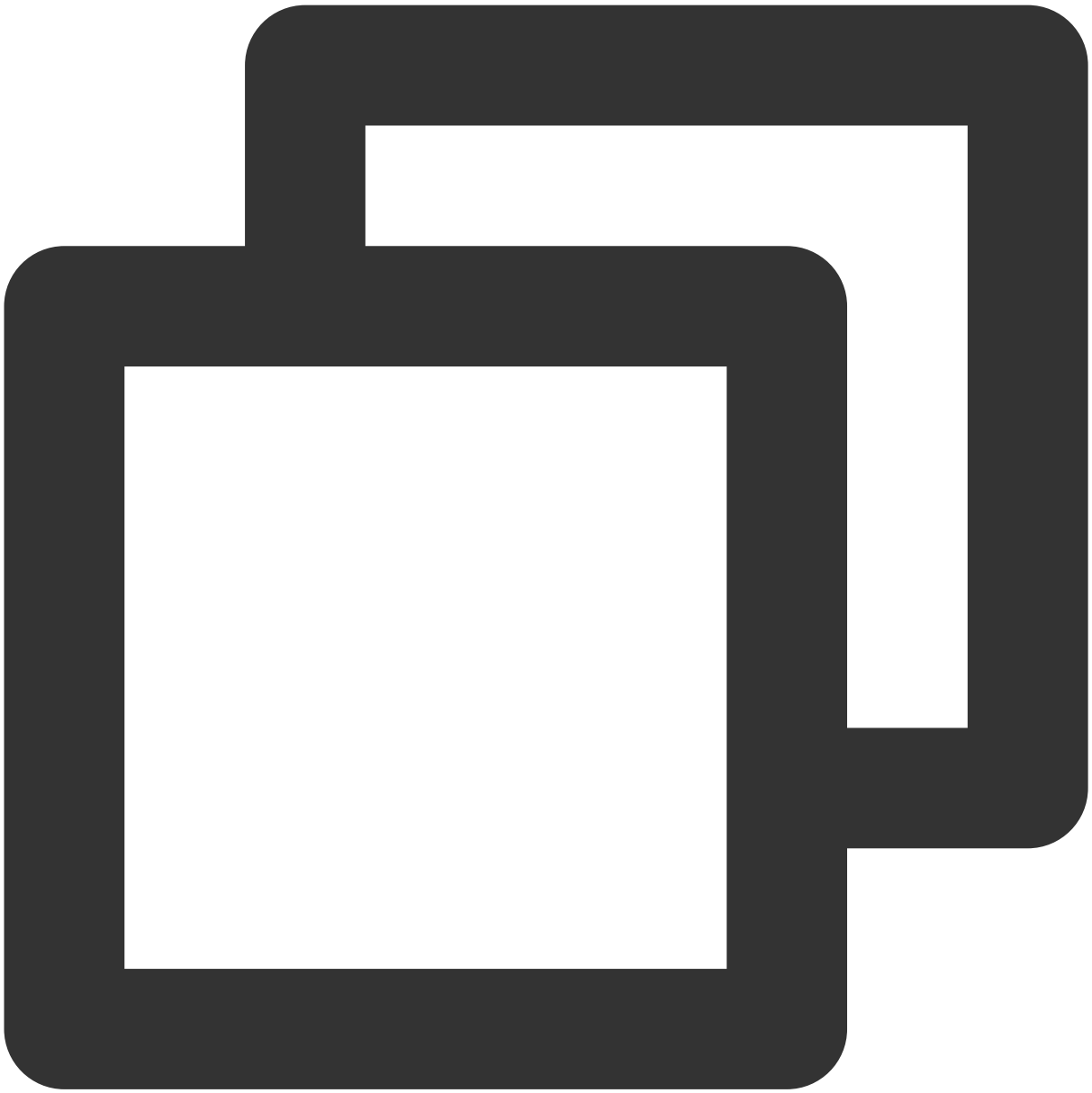


```
- (void)setSelfInfo:(NSString * _Nullable)nickname avatar:(NSString * _Nullable)ava
```

参数	类型	含义
nickname	NSString	用户昵称
avatar	NSString	用户头像（格式为 URL）

enableMultiDeviceAbility

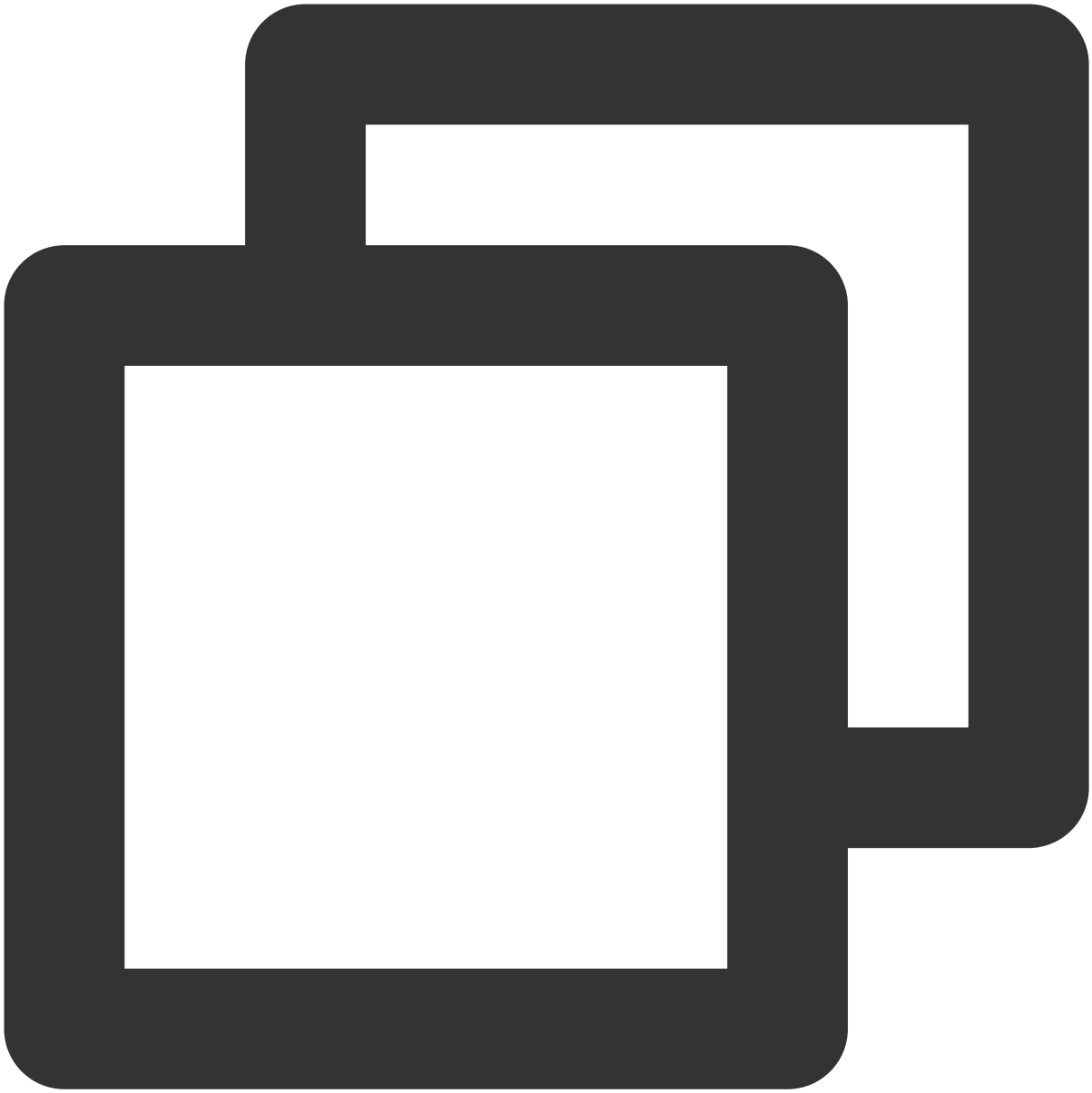
开启/关闭 TUICallEngine 的多设备登录模式（群组通话版套餐支持）。



```
- (void)enableMultiDeviceAbility:(BOOL)enable succ:(TUICallSucc)succ fail:(TUICallF
```

setVideoRenderParams

设置用户视频画面的渲染模式。



```
- (void)setVideoRenderParams:(NSString *)userId params:(TUIVideoRenderParams *)para
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
params	TUIVideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。

```
- (void)setVideoEncoderParams:(TUIVideoEncoderParams *)params succ:(TUICallSucc)suc
```

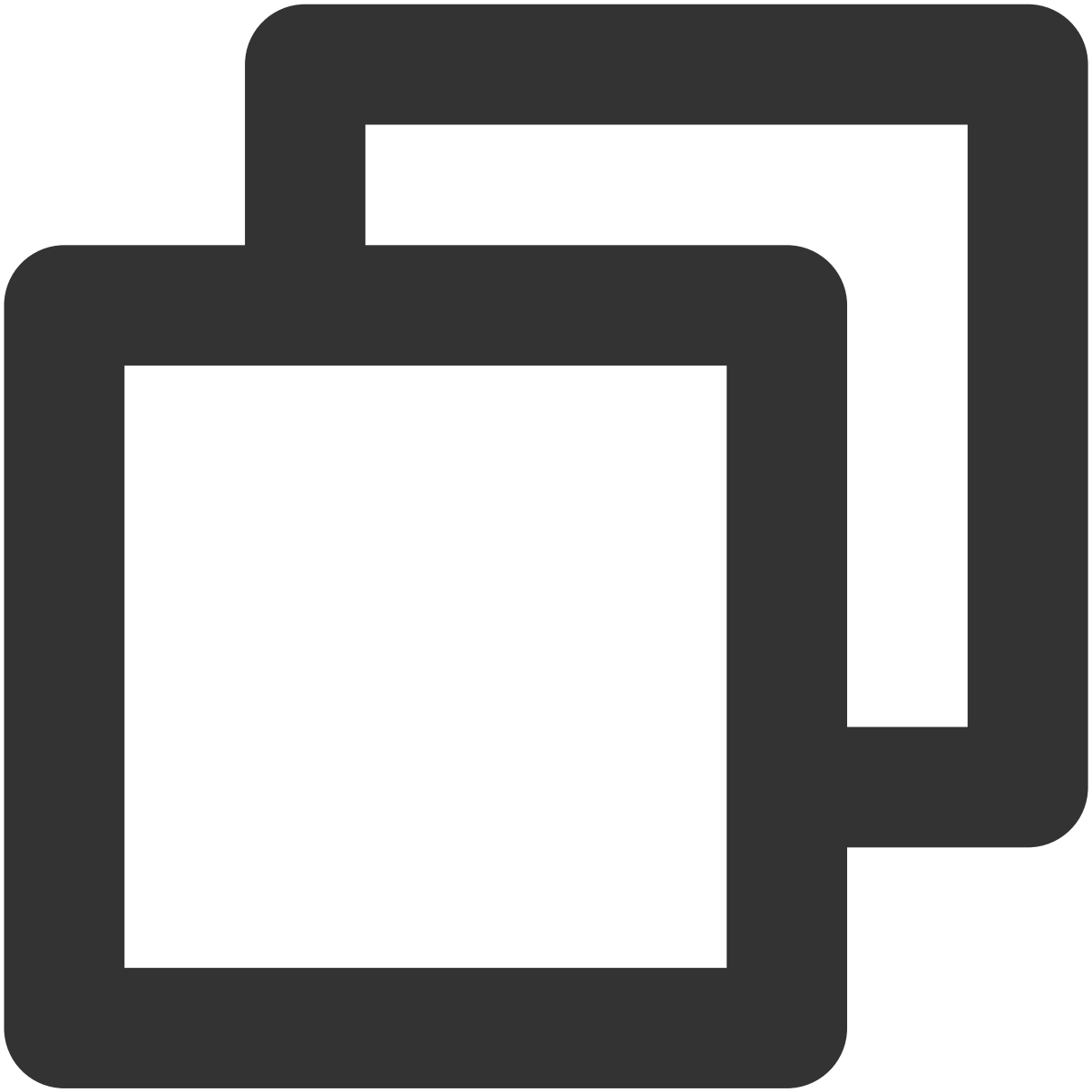
参数如下表所示：

参数	类型	含义
params	TUIVideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高比

getTRTCCloudInstance

高级接口。

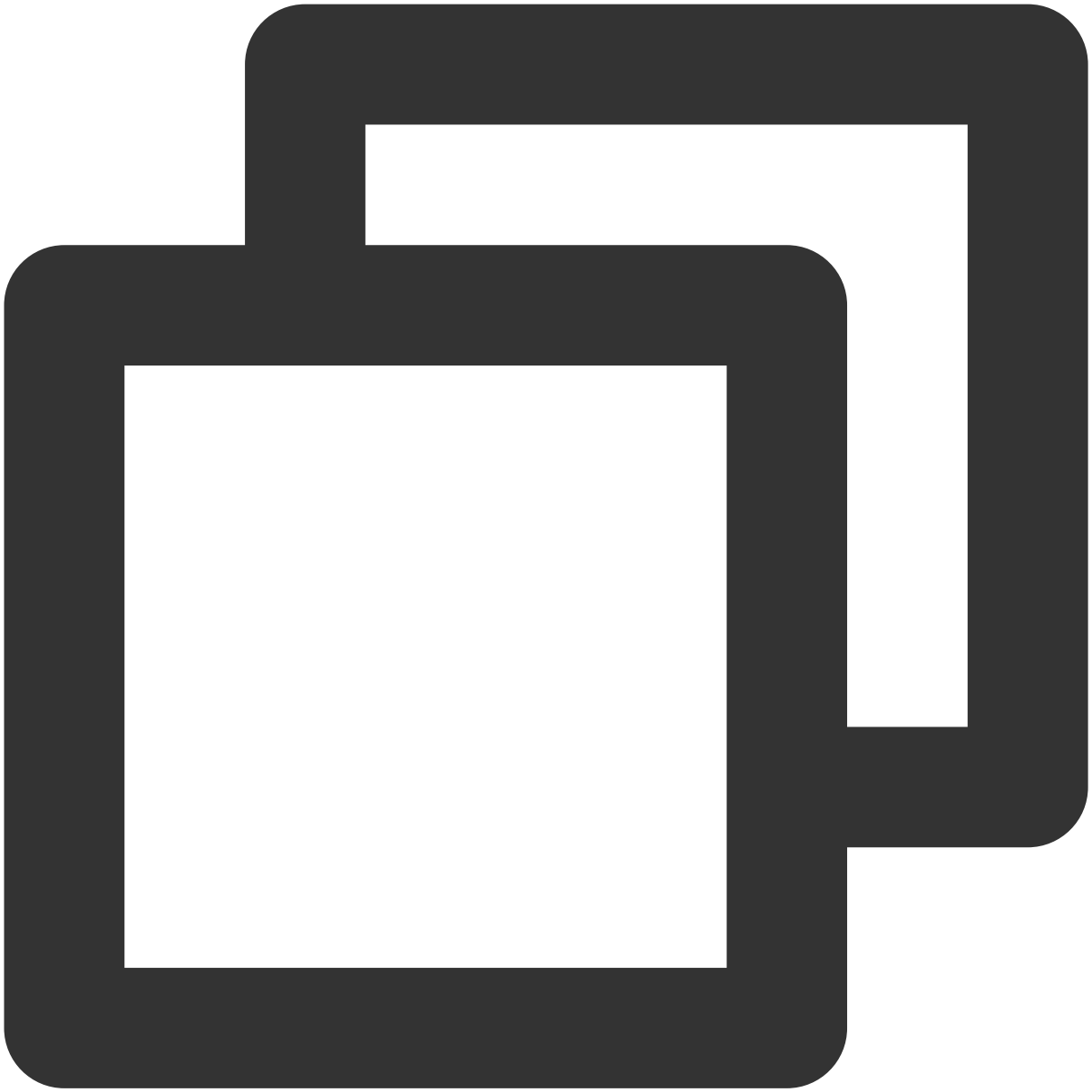
用于接入第三方美颜等。



```
- (TRTCCloud *)getTRTCCloudInstance;
```

setBeautyLevel

设置美颜级别，支持关闭默认美颜。



```
- (void)setBeautyLevel:(CGFloat)level succ:(TUICallSucc)succ fail:(TUICallFail)fail
```

参数如下表所示：

参数	类型	含义
level	CGFloat	美颜级别，美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显

版权所有：腾讯云计算（北京）有限责任公司

TUICallObserver

最近更新时间：2024-01-25 14:39:39

TUICallObserver API 简介

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线

版权所有：腾讯云计算（北京）有限责任公司

onUserSigExpired	在线时票据过期

回调事件详情

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。

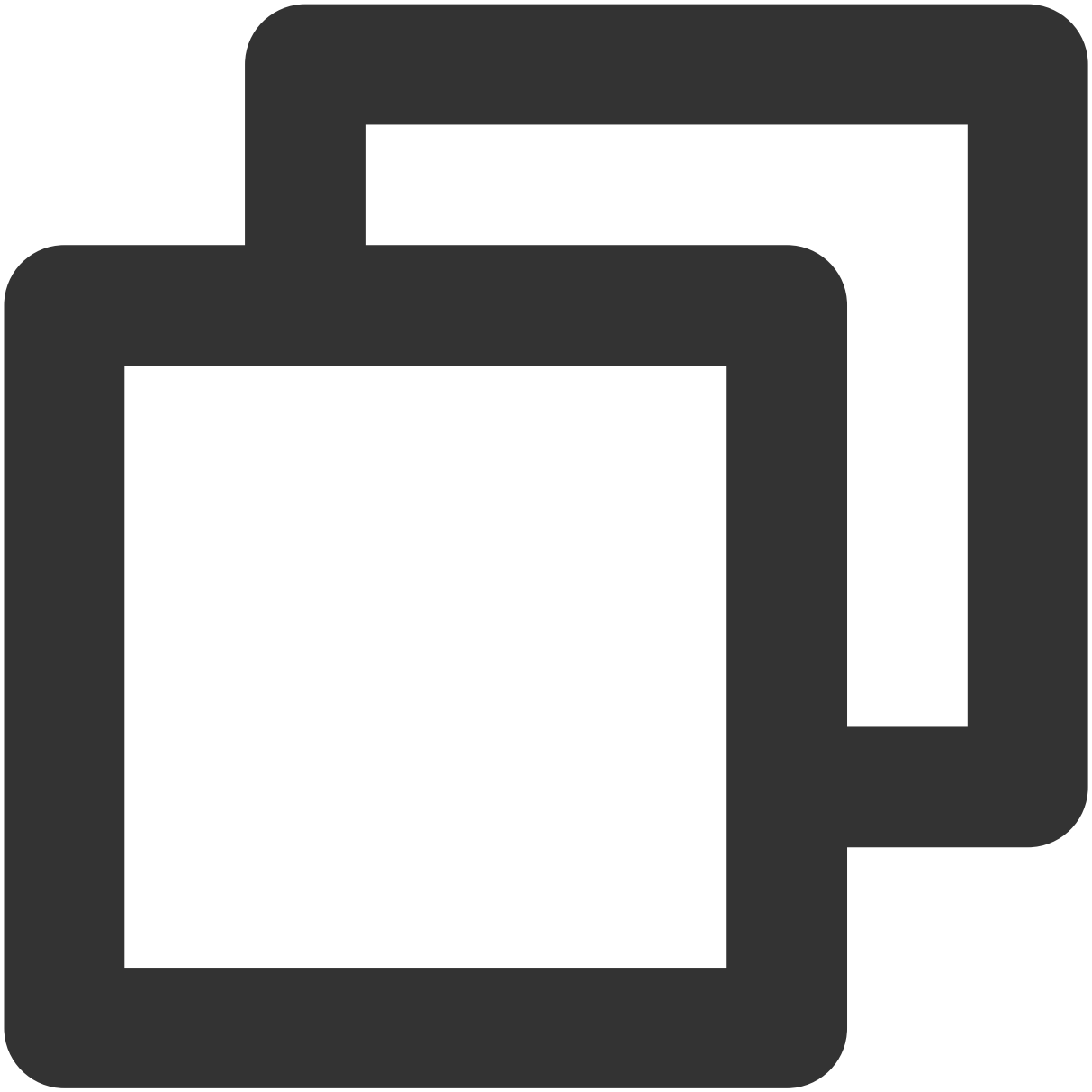
```
- (void)onError:(int)code message:(NSString * _Nullable)message;
```

参数如下表所示：

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

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。



```
- (void)onCallReceived:(NSString *)callerId calleeIdList:(NSArray<NSString *> *)cal
```

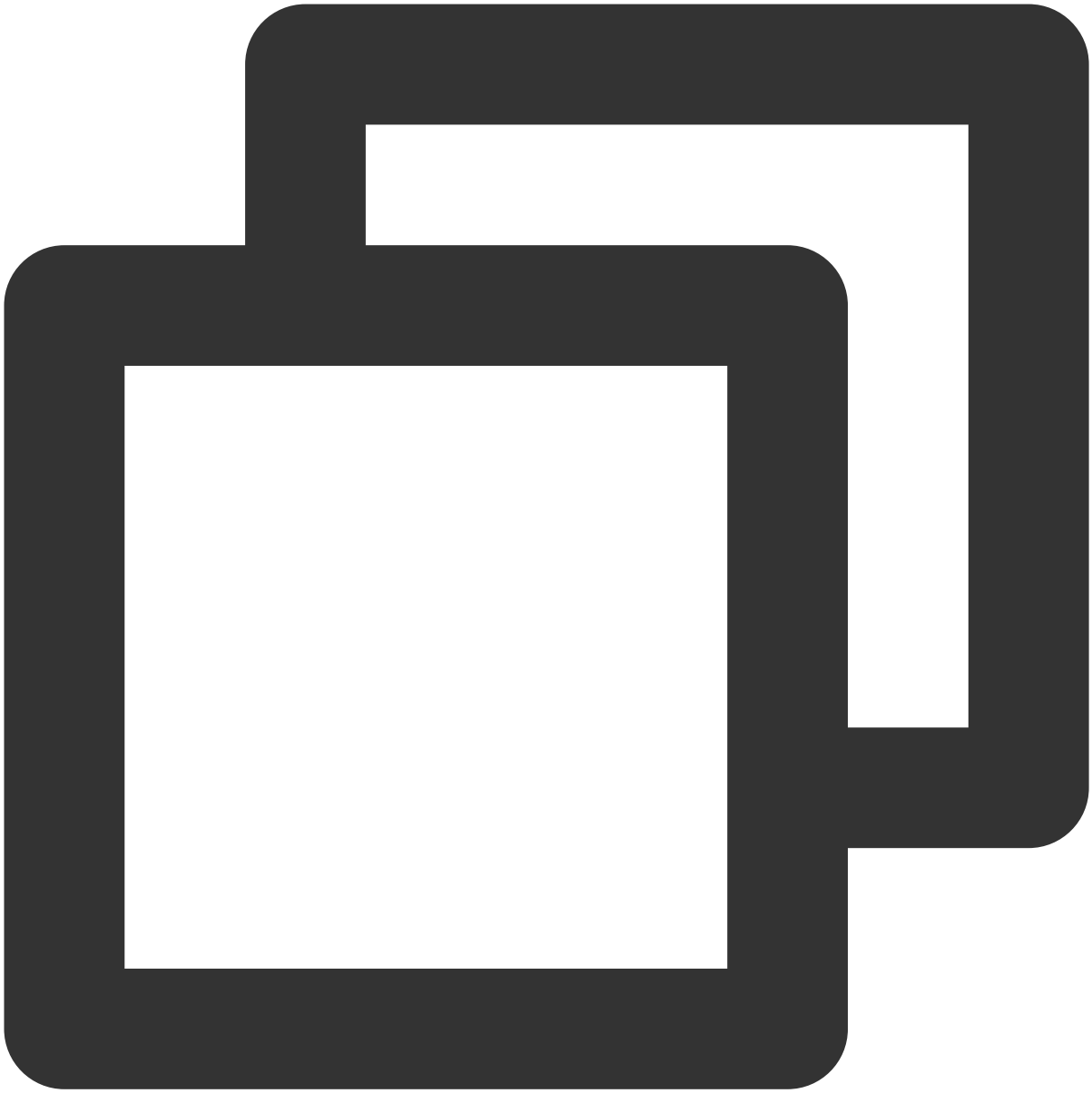
参数如下表所示：

参数	类型	含义
callerId	NSString	主叫 ID（邀请方）
calleeIdList	NSArray	被叫 ID 列表（被邀请方）

groupId	NSString	群组通话 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
userData	NSString	用户增加的扩展字段，详见:TUICallParams

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。
主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）
被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）
被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）
被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）
异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）



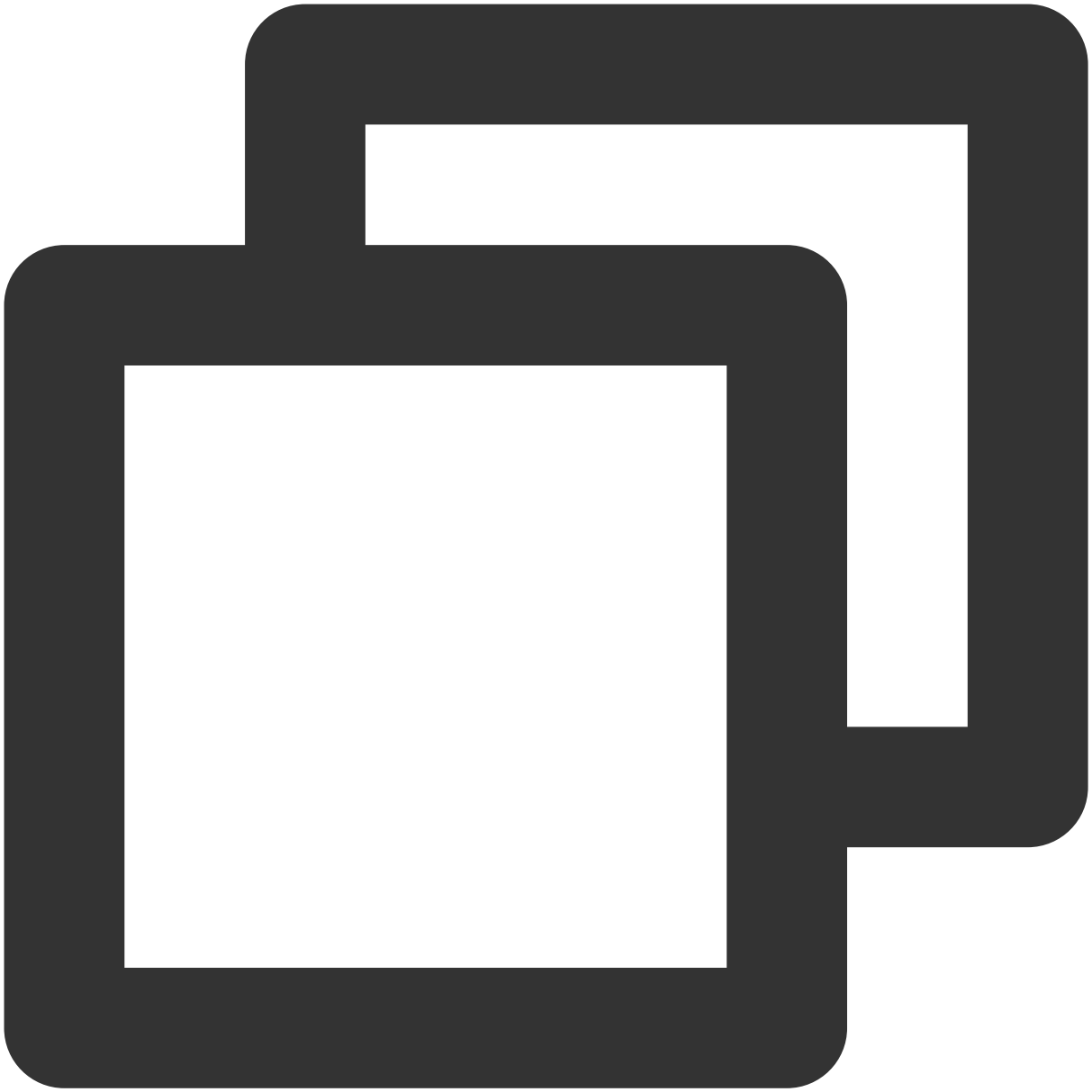
```
- (void)onCallCancelled:(NSString *)callerId;
```

参数如下表所示：

参数	类型	含义
callerId	NSString	用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。



```
- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaTy
```

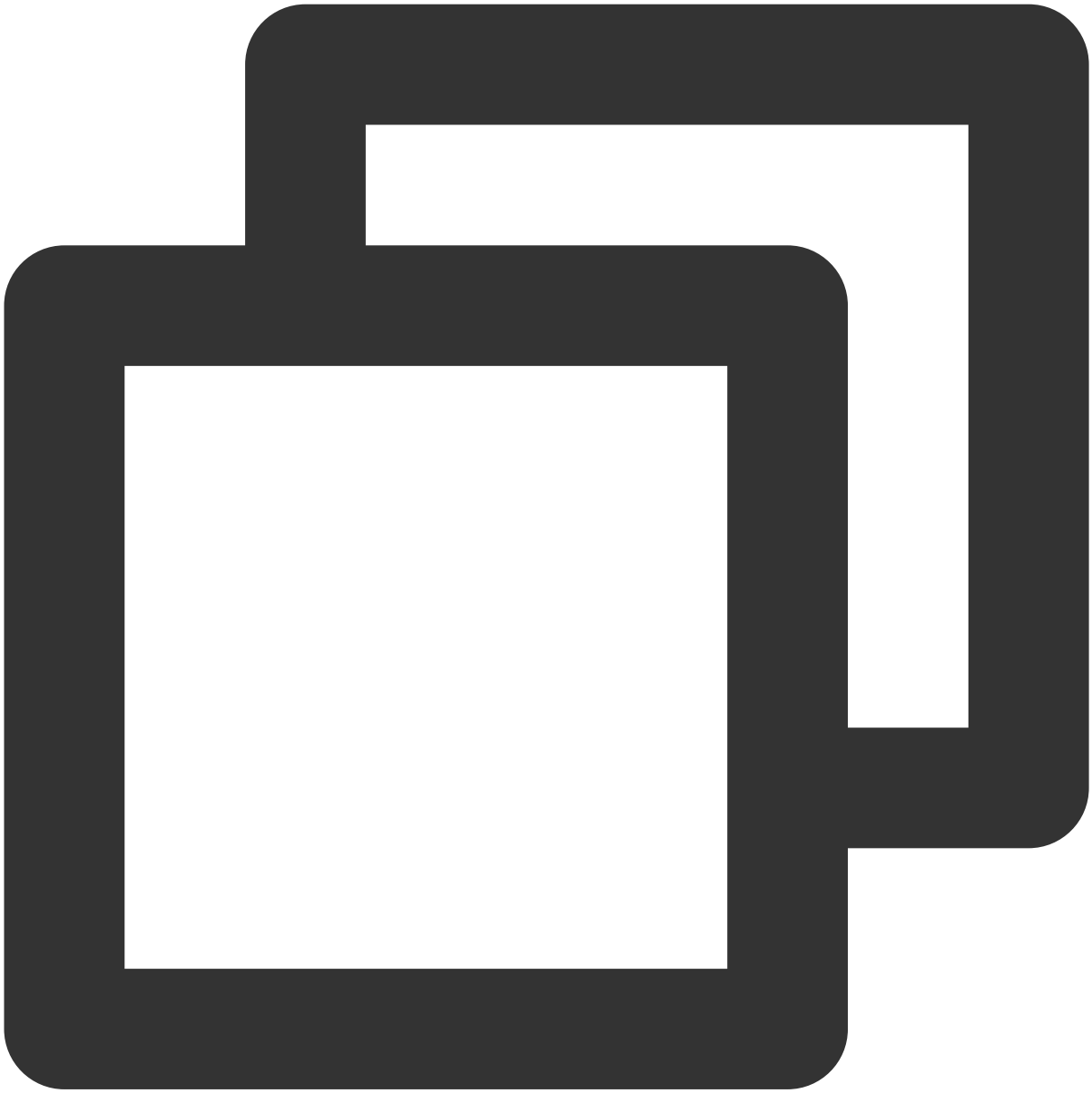
参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
callMediaType	TUICallMediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallRole	角色，枚举类型：主叫、被叫

版权所有：腾讯云计算（北京）有限责任公司

onCallEnd

表示通话挂断，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。



```
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID

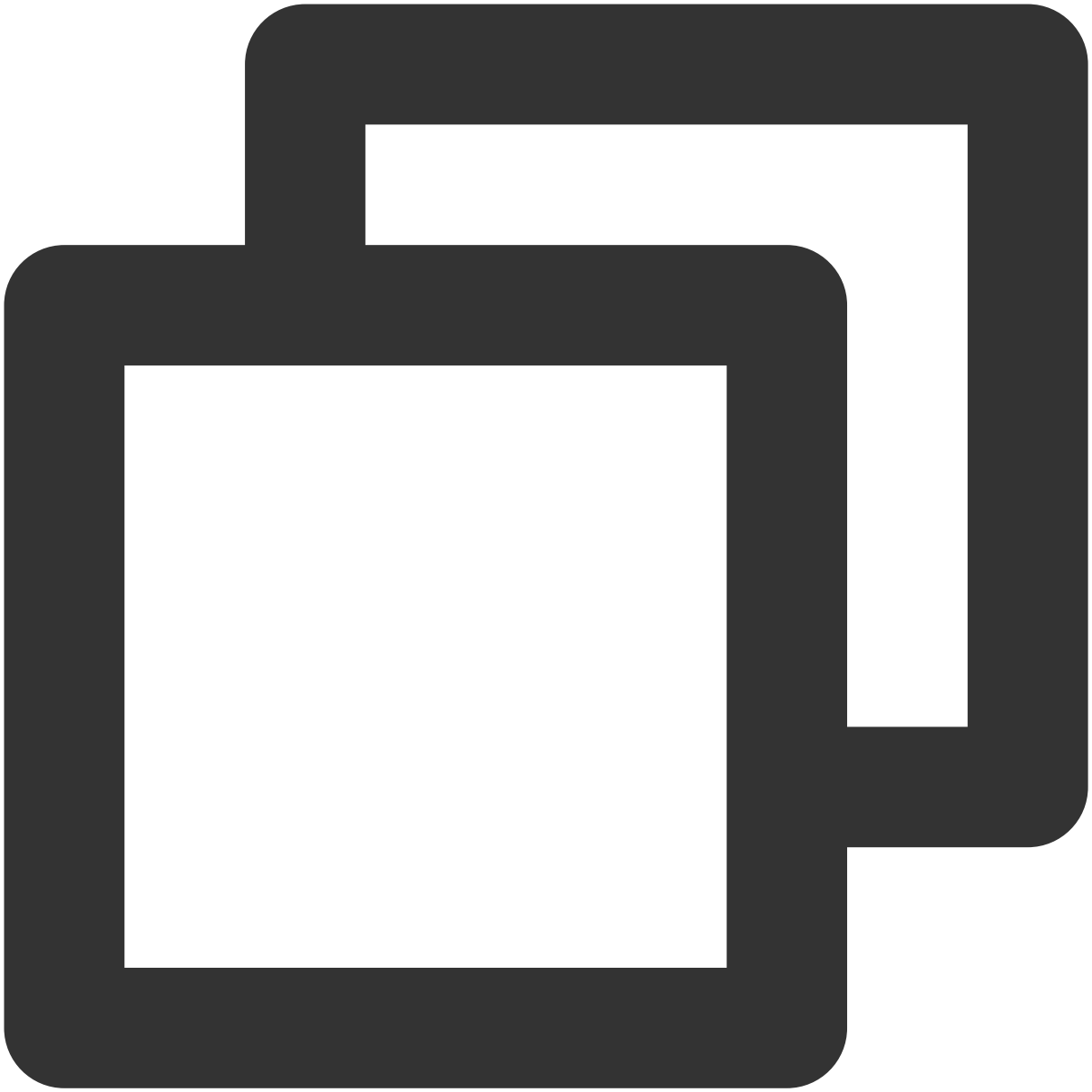
callMediaType	TUICallMediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallRole	角色，枚举类型：主叫、被叫
totalTime	float	此次通话的时长

注意

客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用 REST API 来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。



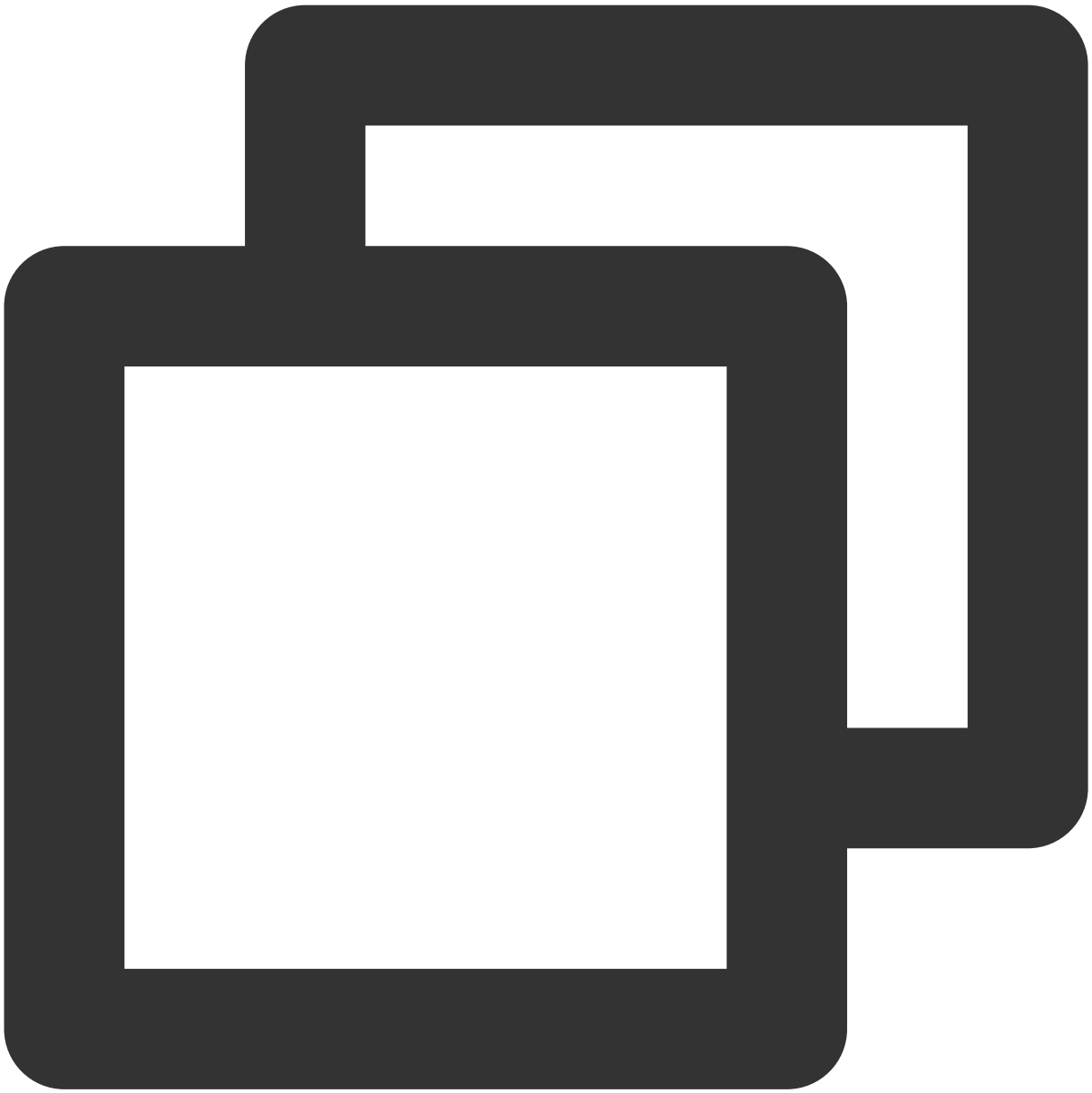
```
- (void)onCallMediaTypeChanged:(TUICallMediaType)oldCallMediaType newCallMediaType:
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallMediaType	旧的通话类型
newCallMediaType	TUICallMediaType	新的通话类型

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。



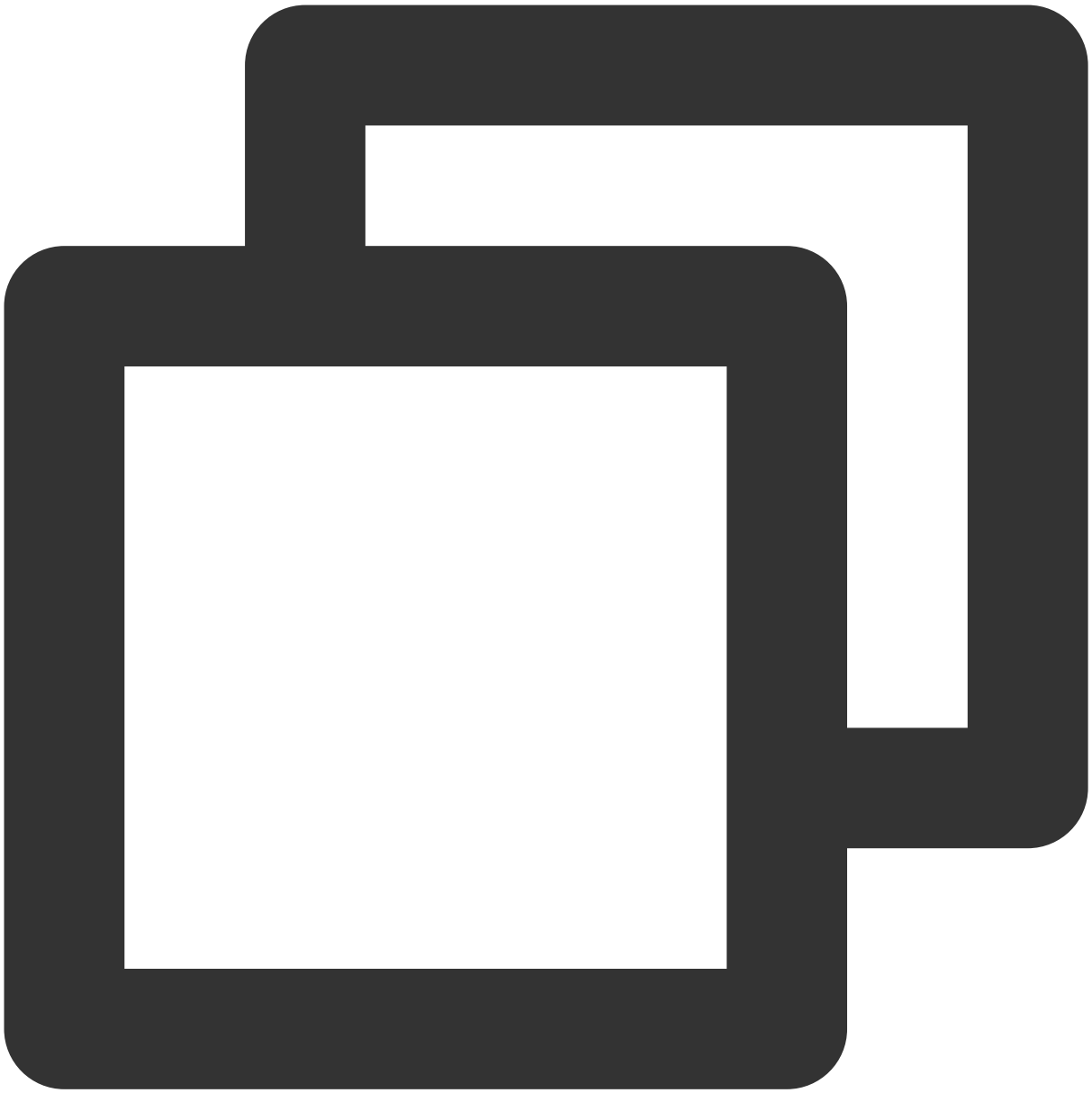
```
- (void)onUserReject:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。



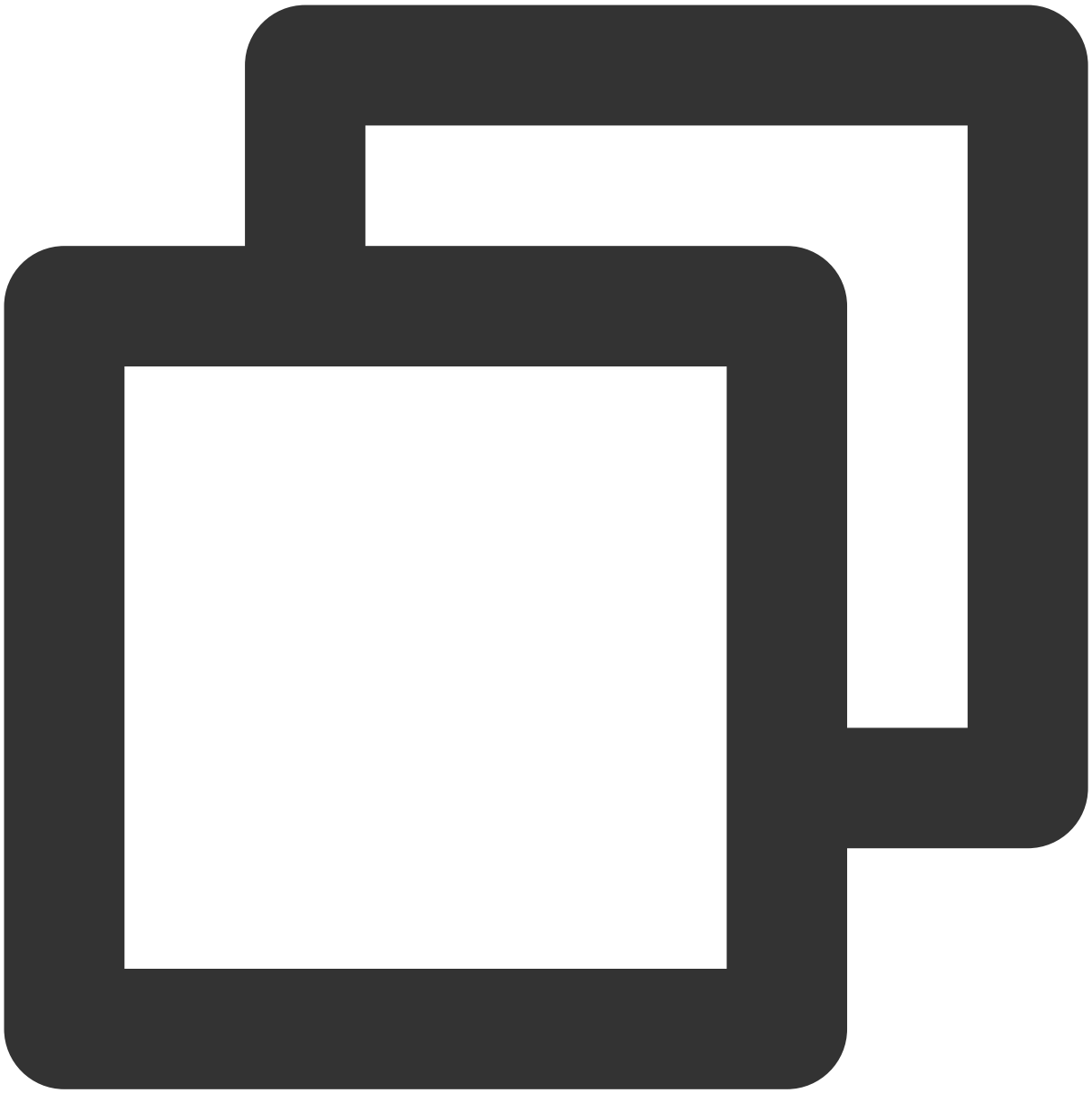
```
- (void)onUserNoResponse:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	无响应用户的 ID

onUserLineBusy

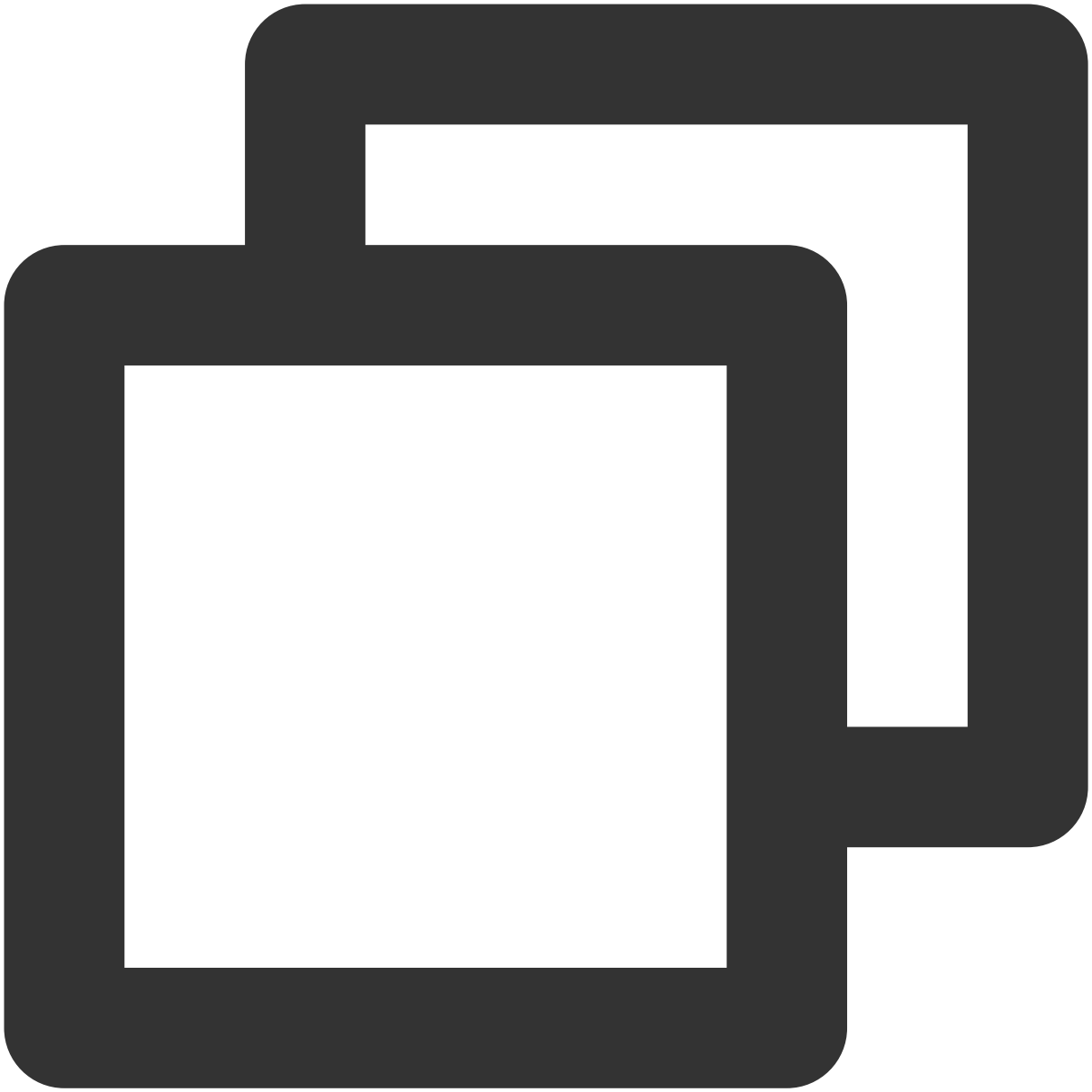
通话忙线回调。



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

onUserJoin

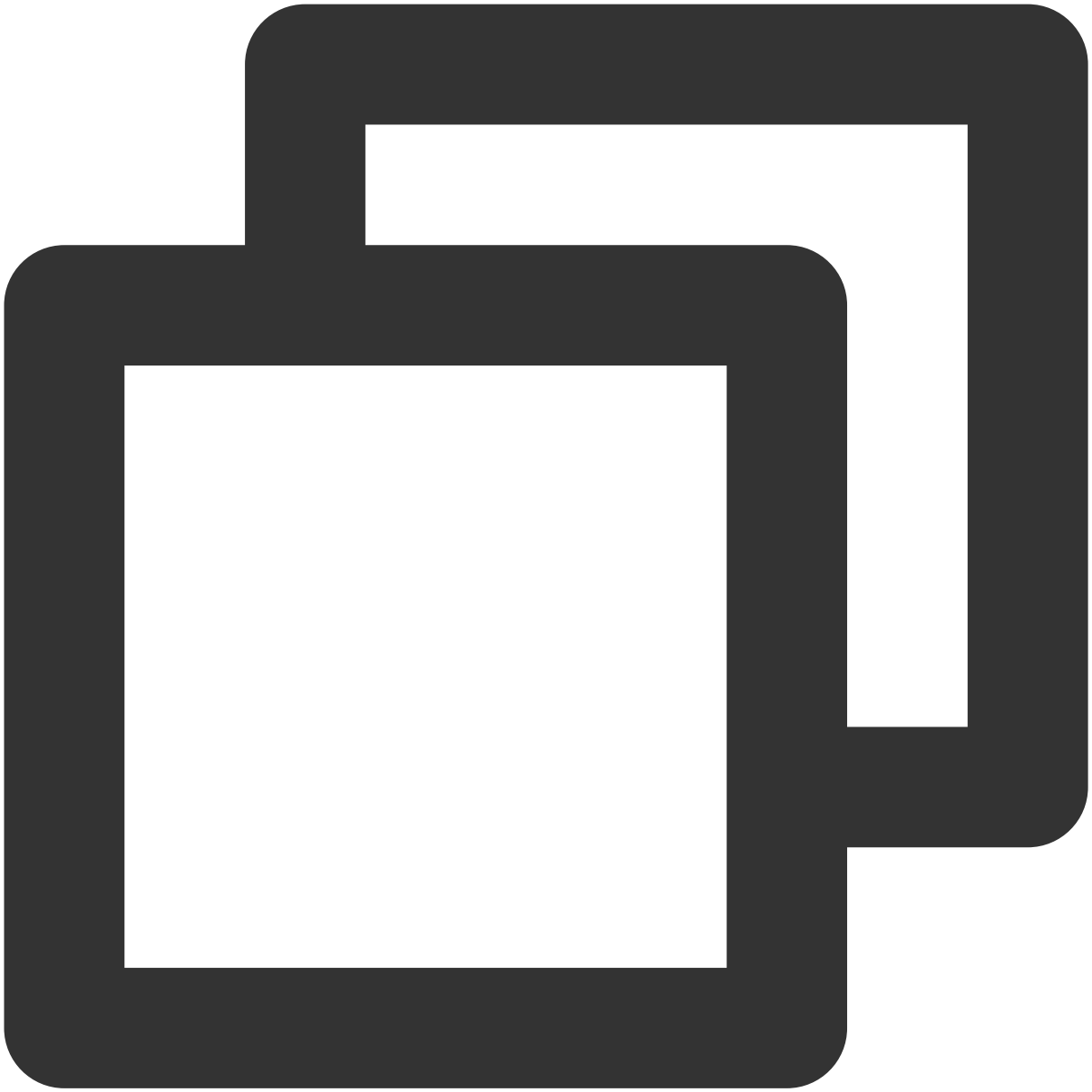
有用户进入此次通话的回调。



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

onUserLeave

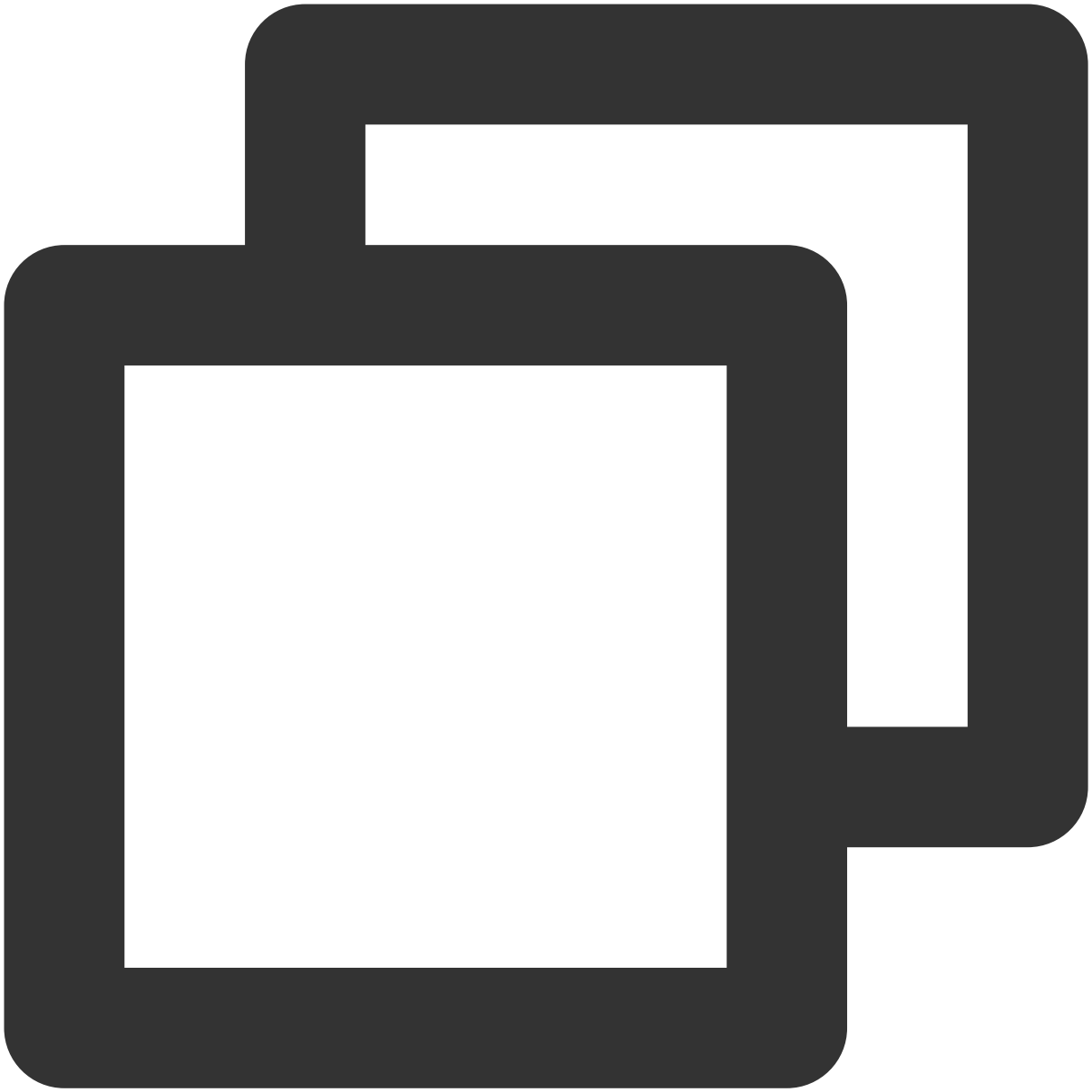
有用户离开此次通话的回调。



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

onUserAudioAvailable

用户是否开启音频上行回调。



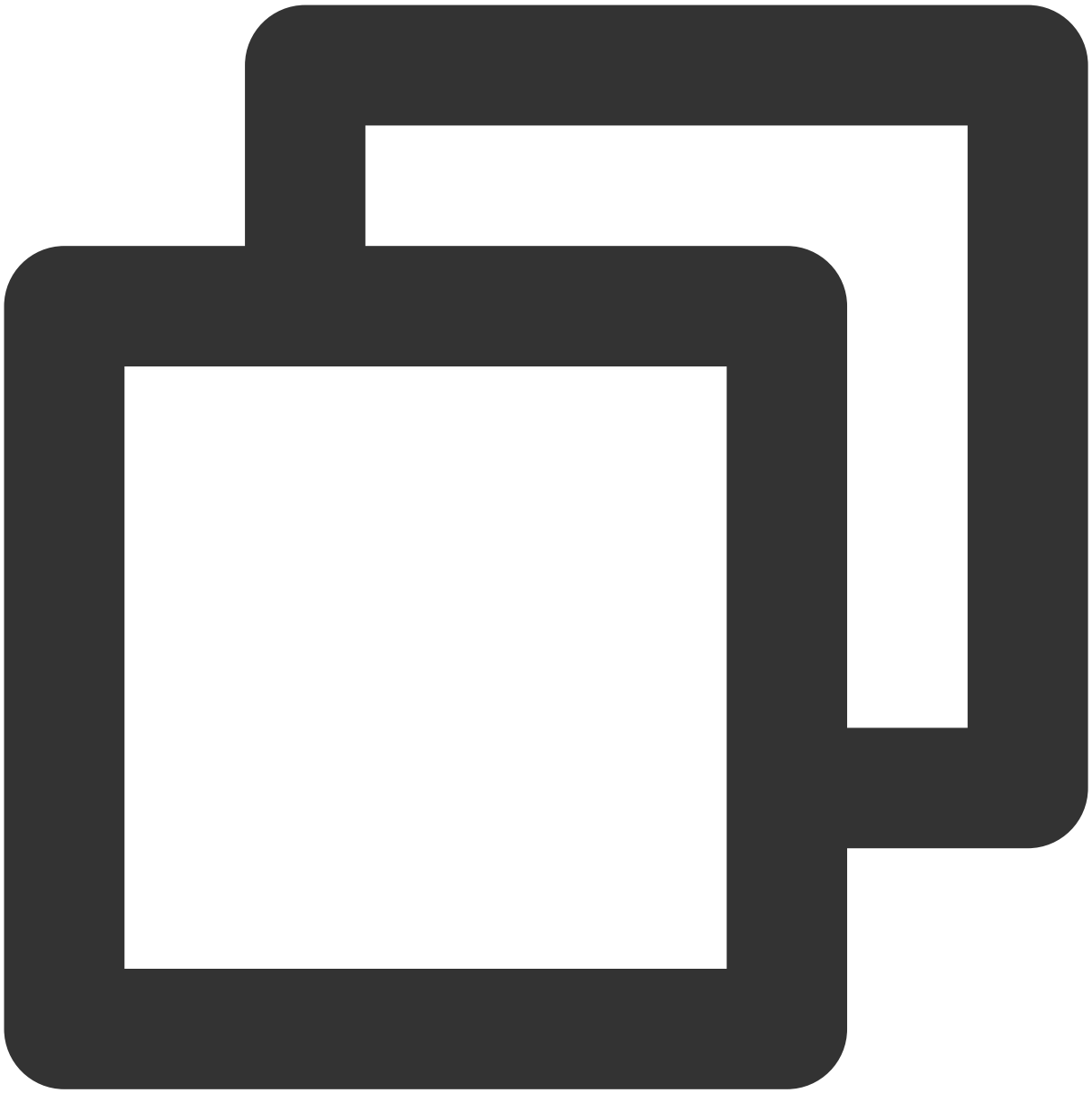
```
- (void)onUserAudioAvailable:(NSString *)userId isAudioAvailable:(BOOL)isAudioAvail
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID
isAudioAvailable	BOOL	用户音频是否可用

onUserVideoAvailable

用户是否开启视频上行回调。



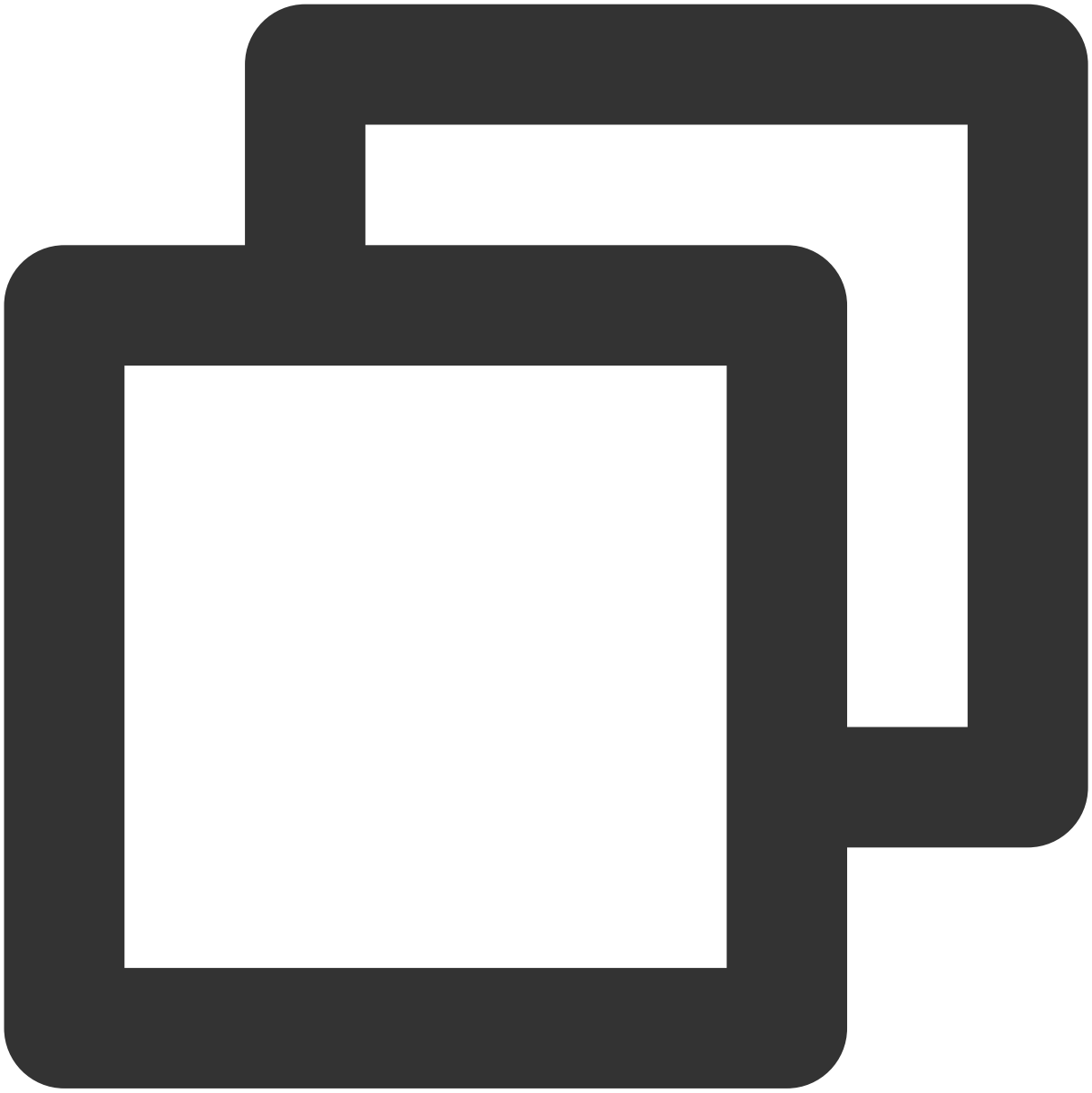
```
- (void)onUserVideoAvailable:(NSString *)userId isVideoAvailable:(BOOL)isVideoAvail
```

参数如下表所示：

参数	类型	含义
userId	NSString	通话用户 ID
isVideoAvailable	BOOL	用户视频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。



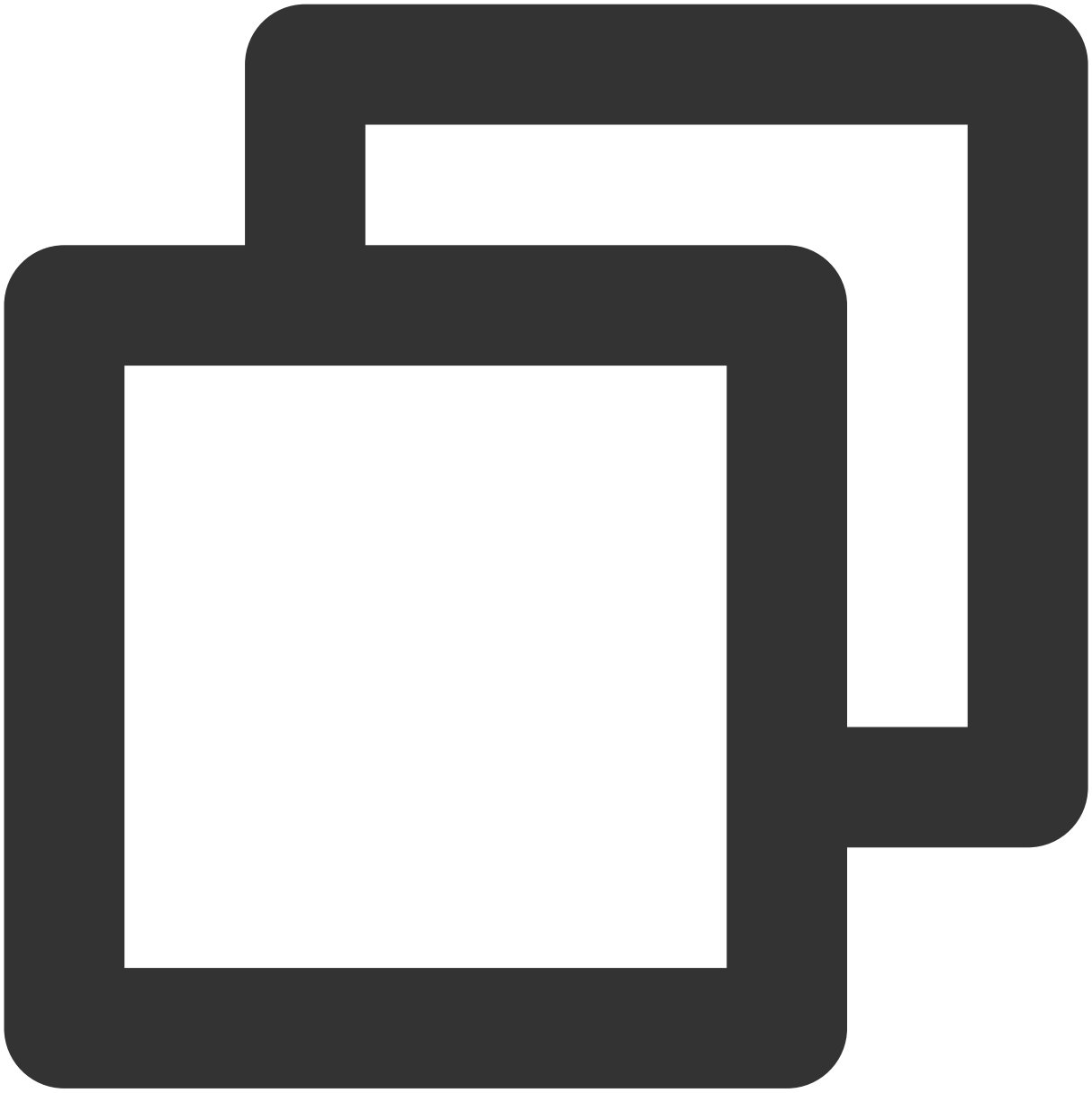
```
- (void)onUserVoiceVolumeChanged:(NSDictionary <NSString *, NSNumber *> *)volumeMap
```

参数如下表所示：

参数	类型	含义
volumeMap	NSDictionary	音量表，根据每个 userId 可以获取对应的音量大小，音量最小值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。



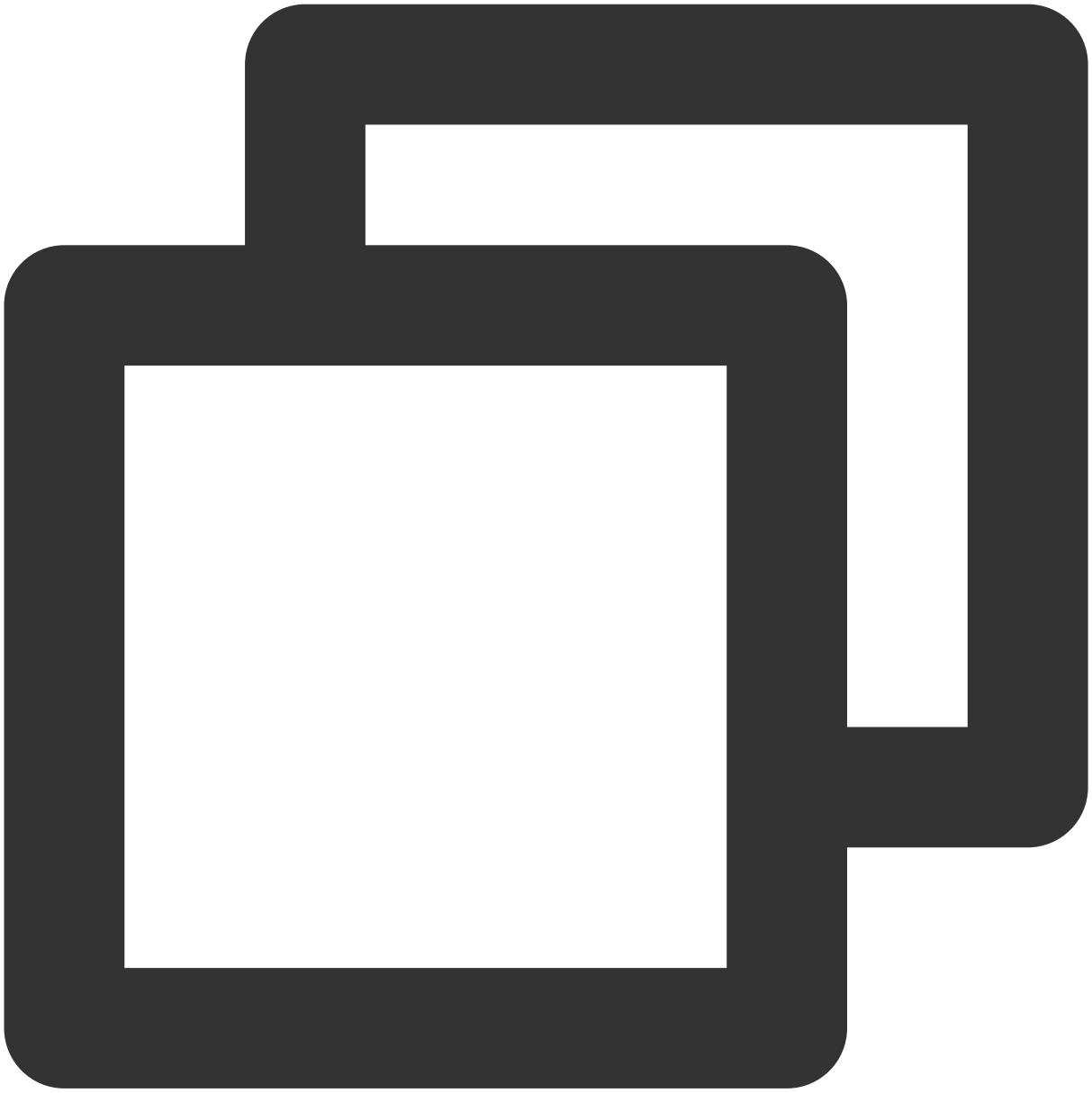
```
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual
```

参数如下表所示：

参数	类型	含义
networkQualityList	NSArray	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

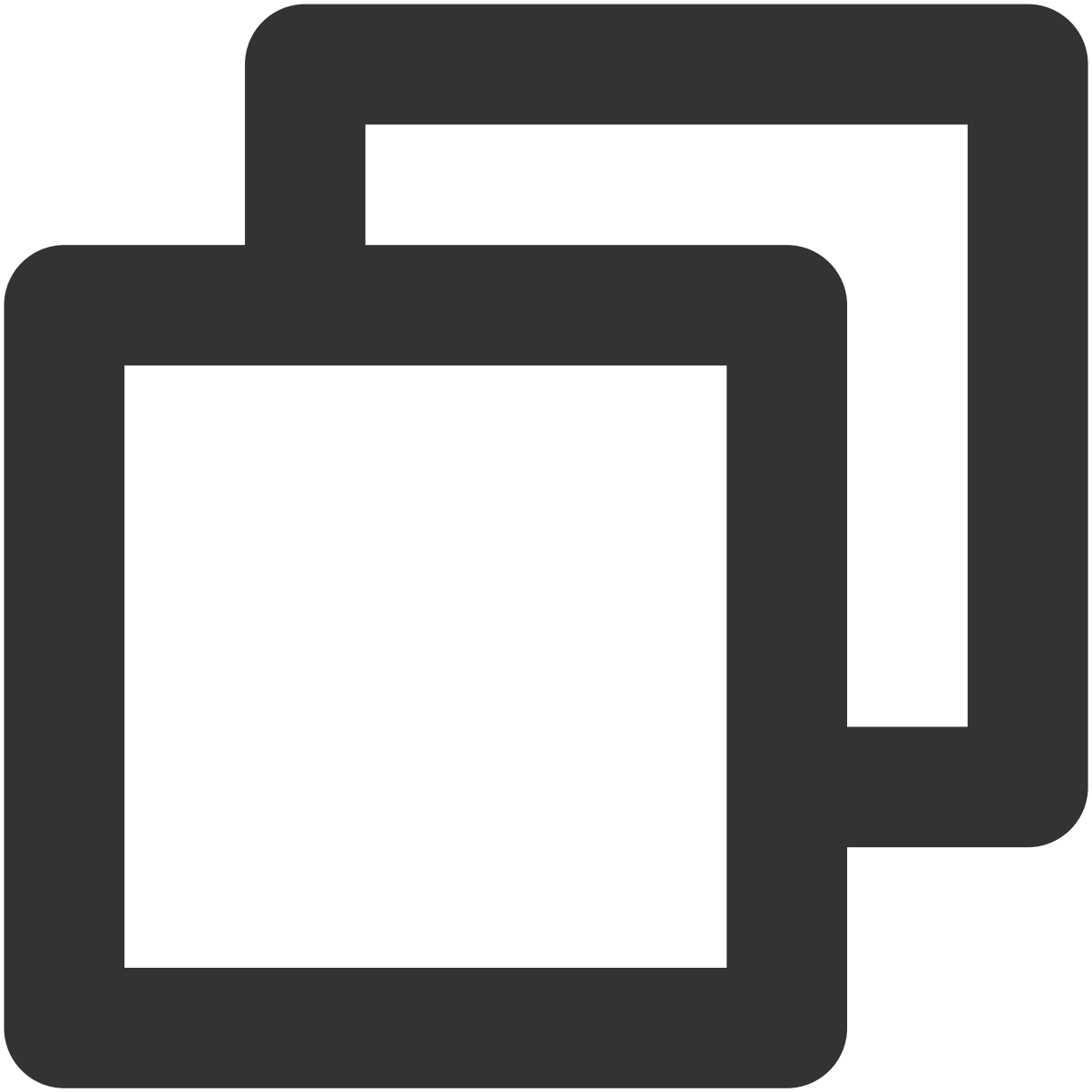
当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。



```
- (void)onKickedOffline;
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。



```
- (void)onUserSigExpired;
```

类型定义

最近更新时间：2023-07-04 14:51:04

常用结构

TUICallDefine

类型	描述
TUICallParams	通话参数
TUIOfflinePushInfo	离线推送厂商配置信息

TUICommonDefine

类型	描述
TUIRoomId	通话中音视频房间 ID
TUINetworkQuality	网络质量
TUIVideoRenderParams	视频画面的渲染参数
TUIVideoEncoderParams	视频编码参数

枚举定义

TUICallDefine

类型	说明
TUICallMediaType	通话类型
TUICallRole	通话角色
TUICallStatus	通话状态
TUICallScene	通话场景
TUICallIOSOfflinePushType	iOS 离线推送类型

TUICommonDefine

类型	说明

版权所有：腾讯云计算（北京）有限责任公司

TUIAudioPlaybackDevice	音频设备
TUICamera	摄像头类型
TUINetworkQuality	网络质量
TUIVideoRenderParamsFillMode	视频画面填充模式
TUIVideoRenderParamsRotation	视频画面旋转方向
TUIVideoEncoderParamsResolutionMode	视频宽高比模式
TUIVideoEncoderParamsResolution	视频分辨率

TUICallParams

通话参数

参数	类型	描述
roomId	TUIRoomId	通话中音视频房间ID
offlinePushInfo	TUIOfflinePushInfo	厂商离线推送配置信息
timeout	int	通话超时时间，默认：30s，单位：秒
userData	NSString	发起通话时自定义扩展字段，被叫端在 onCallReceived 中回调

TUIOfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

参数	类型	描述
title	NSString	离线推送展示通知栏标题
desc	NSString	离线推送展示通知栏内容
ignoreIOSBadge	BOOL	离线推送忽略 badge 计数（仅对 iOS 生效），如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加。
iOSSound	NSString	离线推送声音设置（仅对 iOS 生效）。 当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。 当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。 如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工

		程，然后把语音文件名（带后缀）设置给 iOSSound
androidSound	NSString	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。 只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声 另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	NSString	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	NSInteger	VIVO 推送消息分类 (待废弃接口，VIVO 推送服务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。 0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	NSString	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	NSString	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	NSString	华为推送消息分类，详见：华为消息分类标准
isDisablePush	BOOL	是否关闭推送（默认开启推送）
iOSPushType	TUICallIOSOfflinePushType	iOS 离线推送类型，默认：APNs

TUIRoomId

通话中音视频房间ID

(1) `intRoomId` 与 `strRoomId` 是互斥的，若您选用 `strRoomId` ，则 `intRoomId` 需要填写为 0。若两者都填，SDK 将优先选用 `intRoomId` 。

(2) 不要混用 `intRoomId` 和 `strRoomId` ，因为它们之间是不互通的，比如数字123和字符串123是两个完全不同的房间。

参数	类型	描述
intRoomId	UInt32	数字房间号 **取值范围** 1 - 2147483647(2^31-1)
strRoomId	NSString	字符串房间号 **推荐取值** 限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）：

		大小写英文字母（a-zA-Z）； 数字（0-9）； 空 格、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`、`=`

注意：

字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！

TUINetworkQualityInfo

用户网络质量

类型	类型	描述
userId	NSString	用户 ID
quality	TUINetworkQuality	网络质量

TUIVideoRenderParams

视频画面的渲染参数

参数	类型	描述
fillMode	TUIVideoRenderParamsFillMode	视频画面填充模式
rotation	TUIVideoRenderParamsRotation	视频画面旋转方向

TUIVideoEncoderParams

视频编码参数

参数	类型	描述
resolution	TUIVideoEncoderParamsResolution	视频分辨率
resolutionMode	TUIVideoEncoderParamsResolutionMode	视频宽高比

TUICallMediaType

通话类型

类型	取值	描述
TUICallMediaTypeUnknown	0	未知类型
TUICallMediaTypeAudio	1	语音通话

TUICallMediaTypeVideo	2	视频通话

TUICallRole

通话角色

类型	取值	描述
TUICallRoleNone	0	未知类型
TUICallRoleCall	1	主叫（邀请方）
TUICallRoleCalled	2	被叫（被邀请方）

TUICallStatus

通话状态

类型	取值	描述
TUICallStatusNone	0	未定义
TUICallStatusWaiting	1	通话等待中
TUICallStatusAccept	2	通话已接听

TUICallScene

通话场景

类型	取值	描述
TUICallSceneGroup	0	群组通话
TUICallSceneMulti	1	匿名群组通话（暂不支持，敬请期待）
TUICallSceneSingle	2	单人通话

TUICallIOSOfflinePushType

iOS离线推送类型

类型	取值	描述
TUICallIOSOfflinePushTypeAPNs	0	普通推送

TUICallIOSOfflinePushTypeVoIP	1	VoIP 推送

TUIAudioPlaybackDevice

音频设备

类型	取值	描述
TUIAudioPlaybackDeviceSpeakerphone	0	扬声器
TUIAudioPlaybackDeviceEarpiece	1	听筒

TUICamera

前置/后置 摄像头

类型	取值	描述
TUICameraFront	0	前置摄像头
TUICameraBack	1	后置摄像头

TUINetworkQuality

网络质量

类型	取值	描述
TUINetworkQualityUnknown	0	未定义
TUINetworkQualityExcellent	1	当前网络非常好
TUINetworkQualityGood	2	当前网络比较好
TUINetworkQualityPoor	3	当前网络一般
TUINetworkQualityBad	4	当前网络较差
TUINetworkQualityVbad	5	当前网络很差
TUINetworkQualityDown	6	当前网络不满足通话的最低要求

TUIVideoRenderParamsFillMode

视频画面填充模式

类型	取值	描述

TUIVideoRenderParamsFillModeFill	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完整
TUIVideoRenderParamsFillModeFit	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边

TUIVideoRenderParamsRotation

视频画面旋转方向

类型	取值	描述
TUIVideoRenderParamsRotation_0	0	不旋转
TUIVideoRenderParamsRotation_90	1	顺时针旋转90度
TUIVideoRenderParamsRotation_180	2	顺时针旋转180度
TUIVideoRenderParamsRotation_270	3	顺时针旋转270度

TUIVideoEncoderParamsResolutionMode

视频宽高比模式

类型	取值	描述
TUIVideoEncoderParamsResolutionModeLandscape	0	横屏分辨率，例如：TUIVideoEncoderParamsResolution_640_360 + TUIVideoEncoderParamsResolutionModeLandsca = 640 × 360
TUIVideoEncoderParamsResolutionModePortrait	1	竖屏分辨率，例如：TUIVideoEncoderParamsResolution_640_360 + TUIVideoEncoderParamsResolutionModePortrait = 360 × 640

TUIVideoEncoderParamsResolution

视频分辨率

类型	取值	描述
TUIVideoEncoderParamsResolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps

TUIVideoEncoderParamsResolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
TUIVideoEncoderParamsResolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
TUIVideoEncoderParamsResolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
TUIVideoEncoderParamsResolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
TUIVideoEncoderParamsResolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

版权所有：腾讯云计算（北京）有限责任公司

错误码表

最近更新时间：2023-07-04 15:46:44

通知用户在使用音视频通话过程中出现的警告和错误。

TUICallDefine

定义	取值	描述
ERROR_PACKAGE_NOT_PURCHASED	-1001	您当前未购买音视频通话能力套餐，请前往IM控制台 开通免费体验 或 加购正式版
ERROR_PACKAGE_NOT_SUPPORTED	-1002	您当前购买的音视频通话能力套餐不支持该功能，建议您前往IM控制台 升级套餐类型
ERROR_TIM_VERSION_OUTDATED	-1003	IM SDK 版本过低，请升级 IM SDK 版本 >= 6.6；在build.gradle文件中查找并修改："com.tencent.imsdk:imsdk-plus:7.1.3925"
ERROR_PERMISSION_DENIED	-1101	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
ERROR_INIT_FAIL	-1201	未调用 init 方法进行初始化，TUICallEngine API 需在初始化之后使用
ERROR_PARAM_INVALID	-1202	参数错误
ERROR_REQUEST_REFUSED	-1203	当前状态不支持调用
ERROR_REQUEST_REPEATED	-1204	当前方法正在执行中，请勿重复调用
ERROR_SCENE_NOT_SUPPORTED	-1205	当前通话场景，不支持该功能
ERROR_SIGNALING_SEND_FAIL	-1406	信令发送失败，具体失败原因可查看具体方法回调中的errorMsg

IM错误码

音视频通话使用了腾讯云 IM SDK 作为通信的基础服务，比如通话拨打信令、通话忙线信令等核心逻辑。常见错误码如下：

错误码	描述
6014	IM SDK 未登录，请先登录，成功回调之后重试，或者已被踢下线，可使用 TIMManager getLoginUser 检查当前是否在线

6017	参数无效，请检查参数是否符合要求，具体可查看错误信息进一步定义哪个字段
6206	UserSig 过期，请重新获取有效的 UserSig 后再重新登录，获取方法请参见 生成 UserSig
7013	套餐包不支持该接口的使用，请升级到旗舰版套餐
8010	信令请求 ID 无效或已经被处理过

说明：

更多 IM SDK 错误码详见：IM错误码

TRTC错误码

音视频通话使用了腾讯云 TRTC SDK 作为音频和视频的基础服务，比如开关摄像头、开关麦克风等核心逻辑。常见错误码如下：

枚举	取值	描述
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_TRTC_ENTER_ROOM_FAILED	-3301	进入房间失败，请查看 onError 中的 -3301 对应的 msg 提示确认失败原因

说明：

更多 TRTC SDK 错误码详见：TRTC错误码

版权所有：腾讯云计算（北京）有限责任公司

Web
API 概览

最近更新时间：2024-08-09 17:05:01

TUICallKit（含 UI 组件）

TUICallKit 是音视频通话组件的**含 UI 组件**，您可以通过该组件快速实现一个类微信的音视频通话场景。

API	描述
init	初始化 TUICallKit 组件实例
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中，**v3.1.2+ 支持**
setCallingBell	设置自定义来电铃音，**v3.0.0+ 支持**
setSelfInfo	设置用户昵称和头像，**v2.2.0+ 支持**
enableMuteMode	开启/关闭来电铃声，**v3.1.2+ 支持**
enableFloatWindow	开启/关闭悬浮窗功能，**v3.1.0+ 支持**
enableVirtualBackground	开启/关闭模糊背景的功能按钮，**v3.2.4+ 支持**
setLanguage	设置 TUICallKit 组件通话语言
hideFeatureButton	隐藏按钮，**v3.2.9+ 支持**
setLocalViewBackgroundImage	设置本地用户通话界面背景图，**v3.2.9+ 支持**
setRemoteViewBackgroundImage	设置远端用户通话界面背景图，**v3.2.9+ 支持**
setLayoutMode	设置通话界面布局模式，**v3.3.0+ 支持**
setCameraDefaultState	设置摄像头是否默认打开，**v3.3.0+ 支持**
destroyed	销毁 TUICallKit 组件实例

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
on	监听事件
off	取消监听事件
login	登录接口
logout	登出接口
setSelfInfo	设置用户昵称和头像
call	发起 1v1 通话
groupCall	群聊邀请通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
switchCallMediaType	音视频通话切换
startRemoteView	启动远端画面渲染
stopRemoteView	停止远端画面渲染
startLocalView	启动本地画面渲染，**注意：后续将废弃，建议使用 openCamera**
stopLocalView	停止本地画面渲染，**注意：后续将废弃，建议使用 closeCamera**
openCamera	开启摄像头
closeCamara	关闭摄像头
switchCamera	切换前后置摄像头，**注意：仅移动端支持。v3.0.0+ 支持**
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
setVideoQuality	设置视频质量

版权所有：腾讯云计算（北京）有限责任公司

getDeviceList	获取设备列表
switchDevice	切换摄像头或麦克风设备
enableAIVoice	开启/关闭 AI 降噪
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式。**v2.1.1+ 支持**
setBlurBackground	开关/设置背景虚化，**v3.0.6+ 支持**
setVirtualBackground	开关/设置图片背景虚化，**v3.0.6+ 支持**

事件类型定义

TUICallEvent 是 TUICallEngine 对应的事件类，您可以通过监听如下事件，来监听自己感兴趣的事件。

EVENT	描述
TUICallEvent.ERROR	通话过程中错误事件
TUICallEvent.SDK_READY	SDK 进入 ready 状态时收到该事件
TUICallEvent.KICKED_OUT	用户重复登录，被踢下线
TUICallEvent.USER_ACCEPT	xxxx 用户接通的事件
TUICallEvent.USER_ENTER	xxxx 用户加入通话的事件
TUICallEvent.USER_LEAVE	xxxx 用户离开通话的事件
TUICallEvent.REJECT	xxxx 用户拒绝通话的事件
TUICallEvent.NO_RESP	xxxx 用户不响应的事件
TUICallEvent.LINE_BUSY	xxxx 用户忙线的事件
TUICallEvent.USER_VIDEO_AVAILABLE	xxxx 用户是否有视频流的事件
TUICallEvent.USER_AUDIO_AVAILABLE	xxxx 用户是否有音频流的事件
TUICallEvent.USER_VOICE_VOLUME	所有用户音量大小的反馈事件
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	群聊更新邀请列表收到该事件
TUICallEvent.ON_CALL_BEGIN	通话接通的事件，**v1.4.6+ 支持**
TUICallEvent.INVITED	通话请求的事件， **后续废弃，建议使用**

	TUICallEvent.ON_CALL_RECEIVED
TUICallEvent.ON_CALL_RECEIVED	通话请求的事件，**v1.4.6+ 支持**
TUICallEvent.CALLING_CANCEL	通话取消的事件， **后续废弃，建议使用 TUICallEvent.ON_CALL_CANCELED**
TUICallEvent.ON_CALL_CANCELED	通话取消的事件，**v1.4.6+ 支持**
TUICallEvent.ON_CALL_BEGIN	通话接通时抛出的事件
TUICallEvent.CALLING_END	通话结束的事件
TUICallEvent.DEVICED_UPDATED	设备列表更新的事件
TUICallEvent.CALL_TYPE_CHANGED	通话的媒体类型发生改变的回调
TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED	所有用户网络质量的事件，**v3.0.7+ 支持**

文档链接

TUICallEngine
TUICallEvent

TUICallKit

最近更新时间：2024-08-09 17:37:54

API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详见：快速接入 TUICallKit。

API 概览

TUICallKit 是音视频通话组件的**含 UI 组件**，您可以通过该组件快速实现一个类微信的音视频通话场景。
<TUICallKit/>：UI 通话组件主体。
TUICallKitServer 是通话实例，提供的 API 接口如下。

API	描述
init	初始化 TUICallKit 组件实例
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中，**v3.1.2+ 支持**
setCallingBell	设置自定义来电铃音，**v3.0.0+ 支持**
setSelfInfo	设置用户昵称和头像，**v2.2.0+ 支持**
enableMuteMode	开启/关闭来电铃声，**v3.1.2+ 支持**
enableFloatWindow	开启/关闭悬浮窗功能，**v3.1.0+ 支持**
enableVirtualBackground	开启/关闭模糊背景的功能按钮，**v3.2.4+ 支持**
setLanguage	设置 TUICallKit 组件通话语言
hideFeatureButton	隐藏按钮，**v3.2.9+ 支持**
setLocalViewBackgroundImage	设置本地用户通话界面背景图，**v3.2.9+ 支持**
setRemoteViewBackgroundImage	设置远端用户通话界面背景图，**v3.2.9+ 支持**
setLayoutMode	设置通话界面布局模式，**v3.3.0+ 支持**

版权所有：腾讯云计算（北京）有限责任公司

setCameraDefaultState	设置摄像头是否默认打开，**v3.3.0+ 支持**
destroyed	销毁 TUICallKit 组件实例

`<TUICallKit/>` 属性介绍

属性概览

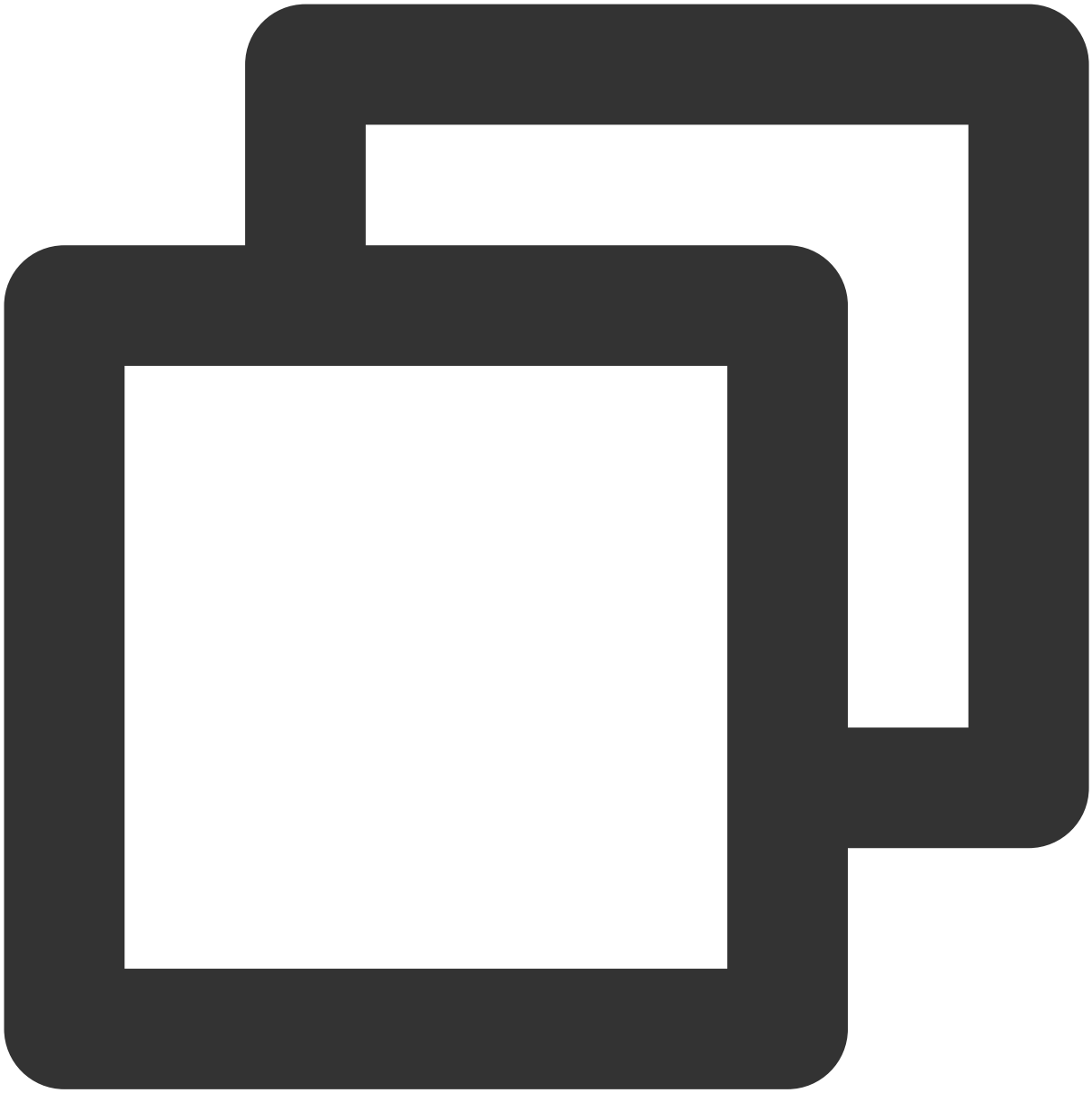
属性	描述	类型	是否必填	默认值
allowedMinimized	是否允许悬浮窗	boolean	否	false
allowedFullScreen	是否允许通话界面全屏	boolean	否	true
videoDisplayMode	通话界面显示模式	VideoDisplayMode	否	VideoDisplayMode.COVER
videoResolution	通话分辨率	VideoResolution	否	VideoResolution.RESOLUTION_480P
beforeCalling	拨打电话前与收到通话邀请前会执行此函数	function(type, error)	否	-
afterCalling	结束通话后会执行此函数	function()	否	-
onMinimized	组件切换最小化状态时会执行此函数，STATUS 值说明	function(oldStatus, newStatus)	否	-
kickedOut	组件抛出的事件，当前登录用户被踢出登录时会触发该事件，通话也会自动结束	function()	否	-

statusChanged	组件抛出的事件，当通话状态发生变化时，会触发该事件，通话状态种类详见，STATUS 值说明	function({oldStatus, newStatus})	否	-

示例代码

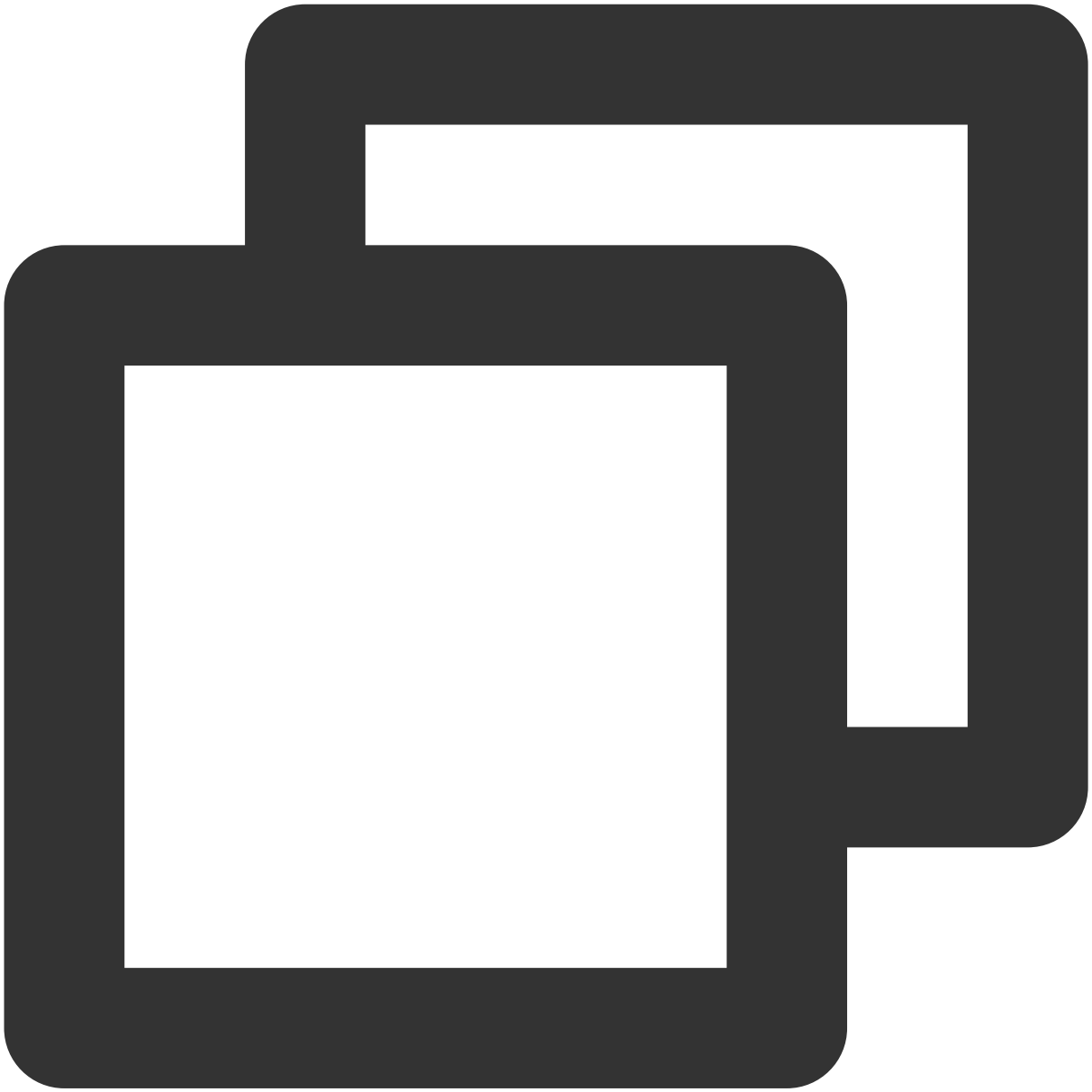
React

Vue



```
import { TUICallKit, VideoDisplayMode, VideoResolution } from "@tencentcloud/call-u

<TUICallKit
    videoDisplayMode={VideoDisplayMode.CONTAIN}
    videoResolution={VideoResolution.RESOLUTION_1080P}
    beforeCalling={handleBeforeCalling}
    afterCalling={handleAfterCalling}
/>

function handleBeforeCalling(type: string, error: any) {
  console.log("[TUICallkit Demo] handleBeforeCalling:", type, error);
```

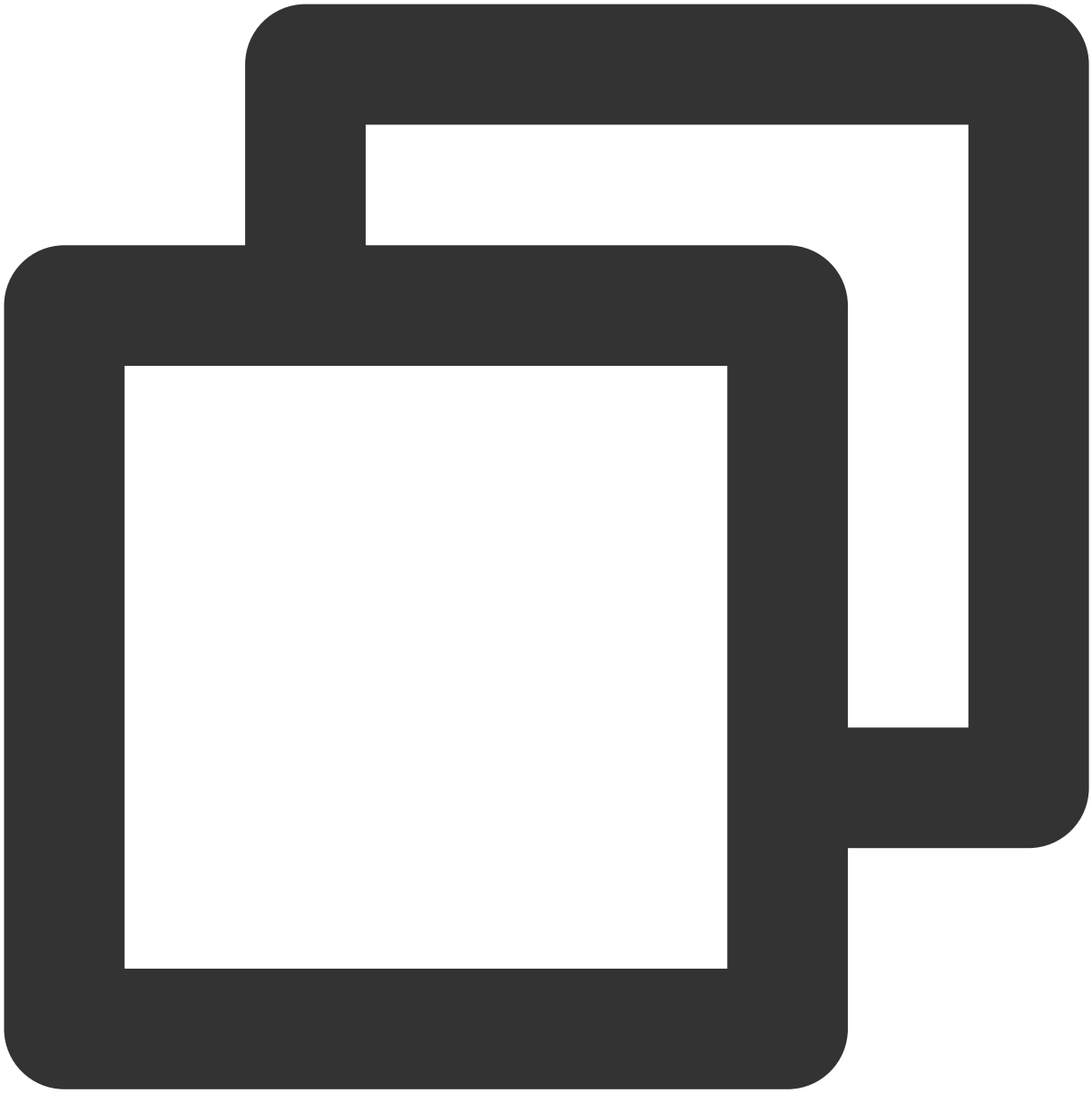
```
}
function handleAfterCalling() {
  console.log("[TUICallkit Demo] handleAfterCalling");
}
```



```
<template>
   <TUICallKit
    :allowedMinimized="true"
    :allowedFullScreen="true"
    :videoDisplayMode="VideoDisplayMode.CONTAIN"
    :videoResolution="VideoResolution.RESOLUTION_1080P"
```

```
    :beforeCalling="beforeCalling"
    :afterCalling="afterCalling"
    :onMinimized="onMinimized"
    :kickedOut="handleKickedOut"
    :statusChanged="handleStatusChanged"
  />
</template>
<script lang="ts" setup>
import { TUICallKit, TUICallKitServer, VideoDisplayMode, VideoResolution, STATUS }

function beforeCalling(type: string, error: any) {
  console.log("[TUICallkit Demo] beforeCalling:", type, error);
}
function afterCalling() {
  console.log("[TUICallkit Demo] afterCalling");
}
function onMinimized(oldStatus: string, newStatus: string) {
  console.log("[TUICallkit Demo] onMinimized: " + oldStatus + " -> " + newStatus);
}
function kickedOut() {
  console.log("[TUICallkit Demo] kickedOut");
}
function statusChanged(args: { oldStatus: string; newStatus: string; }) {
  const { oldStatus, newStatus } = args;
  if (newStatus === STATUS.CALLING_C2C_VIDEO) {
    console.log(`[TUICallkit Demo] statusChanged: ${oldStatus} -> ${newStatus}`);
  }
}
</script>
```

TUICallKitServer API 详情



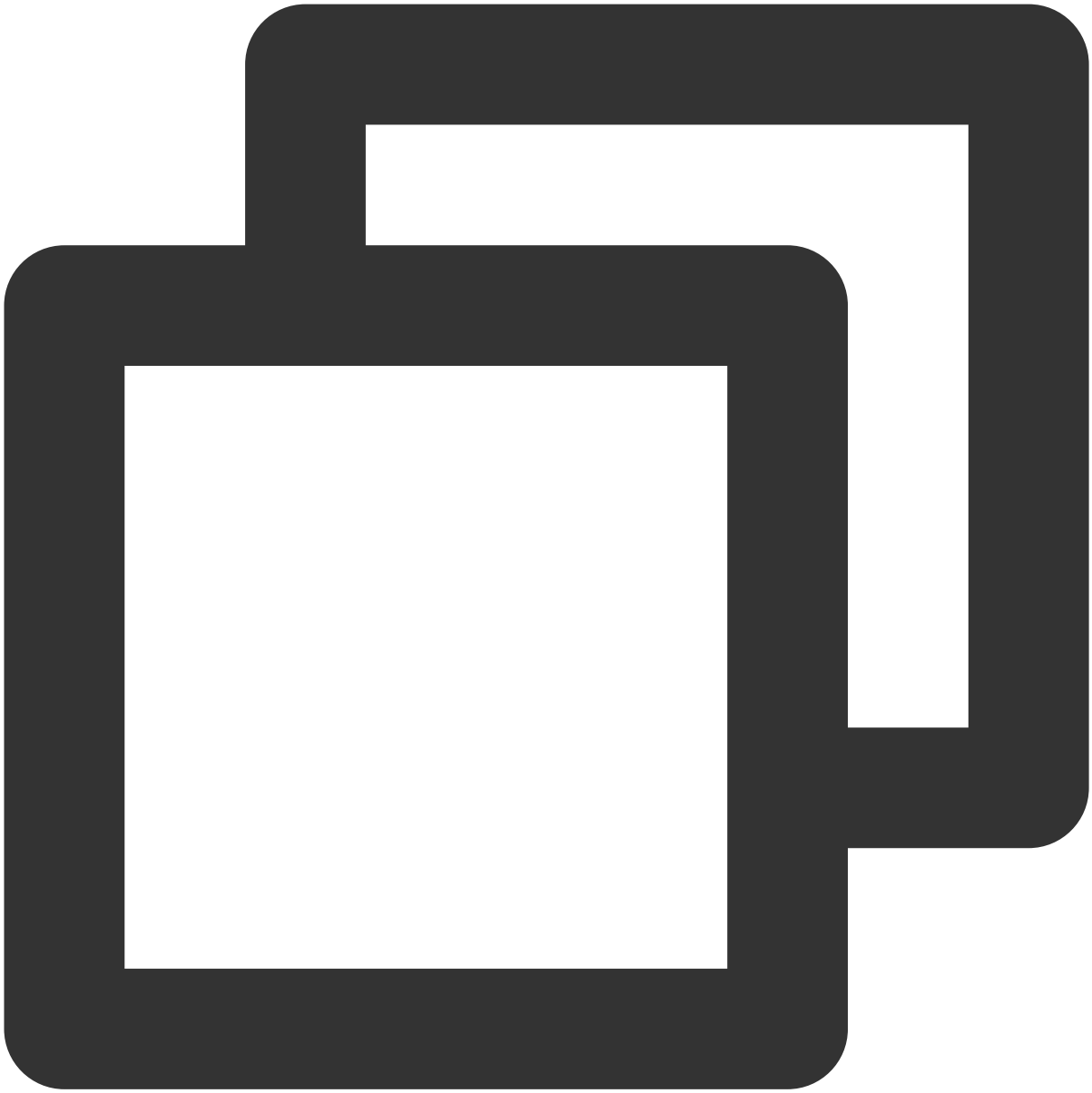
```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";
// Replace it with the call-uikit npm package you are currently using
```

init

初始化 TUICallKit。

注意：

需完成 init 初始化，才能使用 call、groupCall 等功能。



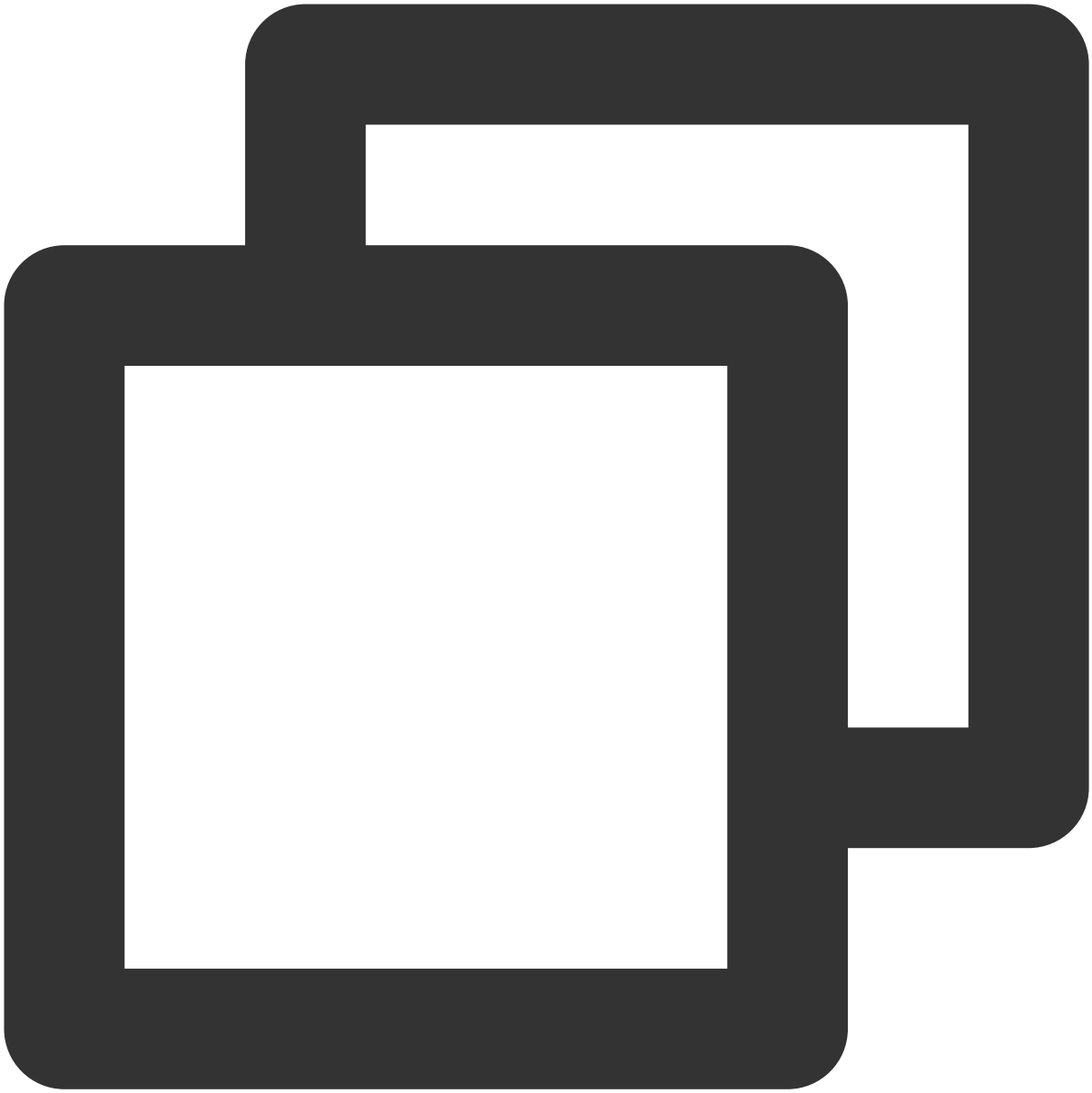
```
try {
  await TUICallKitServer.init({ SDKAppID, userID, userSig });
  // If you already have a tim instance in your project, you need to pass it in her
  // await TUICallKitServer.init({ tim, SDKAppID, userID, userSig});
  console.log("[TUICallKit] Initialization succeeds.");
} catch (error: any) {
  console.error(`[TUICallKit] Initialization failed. Reason: ${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
SDKAppID	Number	是	应用的 SDKAppID，您可以在实时音视频控制台中找到您的 SDKAppID。具体详见 开通服务
userID	String	是	即用户名，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	是	使用 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，获取方式请参考 如何计算 UserSig
tim	TencentCloudChat	否	tim 是 TencentCloudChat SDK 的实例

call

拨打 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。



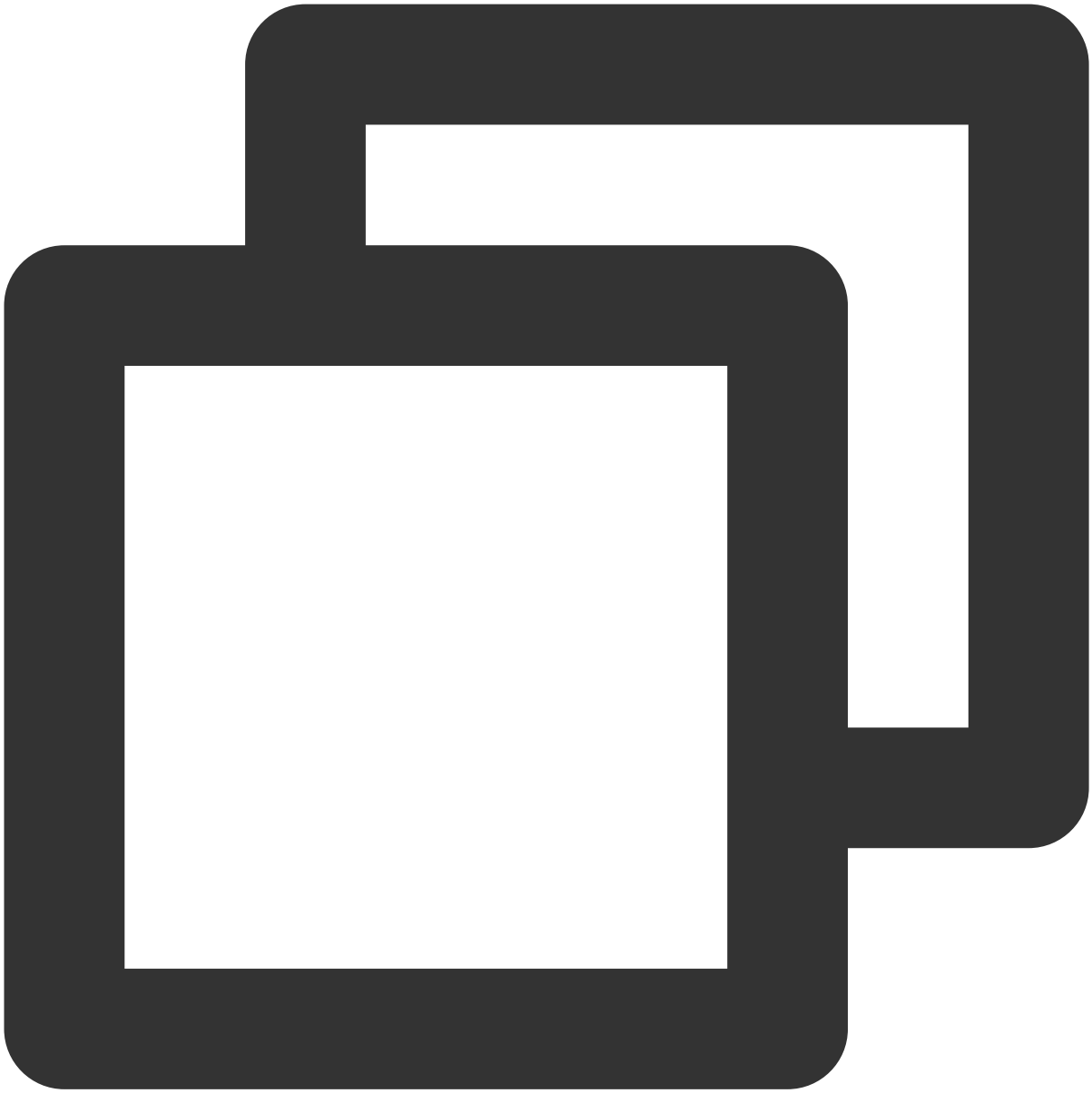
```
import { TUICallKitServer, TUICallType } from '@tencentcloud/call-uikit-react';
try {
  await TUICallKitServer.call({
    userID: 'mike', // username being called
    type: TUICallType.VIDEO_CALL,
  });
} catch (error: any) {
  console.error(`[TUICallKit] Call failed. Reason: ${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
userID	String	是	被呼叫用户的用户名
type	TUICallType	是	通话的媒体类型，参数值说明参考 TUICallType 通话类型
roomID	Number	否	数字房间号，范围 [1, 2147483647]
strRoomID	String	否	字符串房间号。**v3.3.1+ 支持** **范围：** 长度限制为64个字节。支持的字符集范围如下（共89个字符）： 小写和大写英文字母。（a-zA-Z）； 数字（0-9）； `空` 格、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、 1. roomID 与 strRoomID 是互斥的，若您选用 strRoomID，则 roomID 需 2. 不要混用 roomID 和 strRoomID，因为它们之间是不互通的，比如数字
timeout	Number	否	通话超时时间，默认：30s，单位：秒。timeout = 0，设置为不超时
userData	String	否	发起通话时自定义扩展字段，被呼叫用户在 ON_CALL_RECEIVED 事件
offlinePushInfo	Object	否	自定义离线消息推送参数

groupCall

发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。



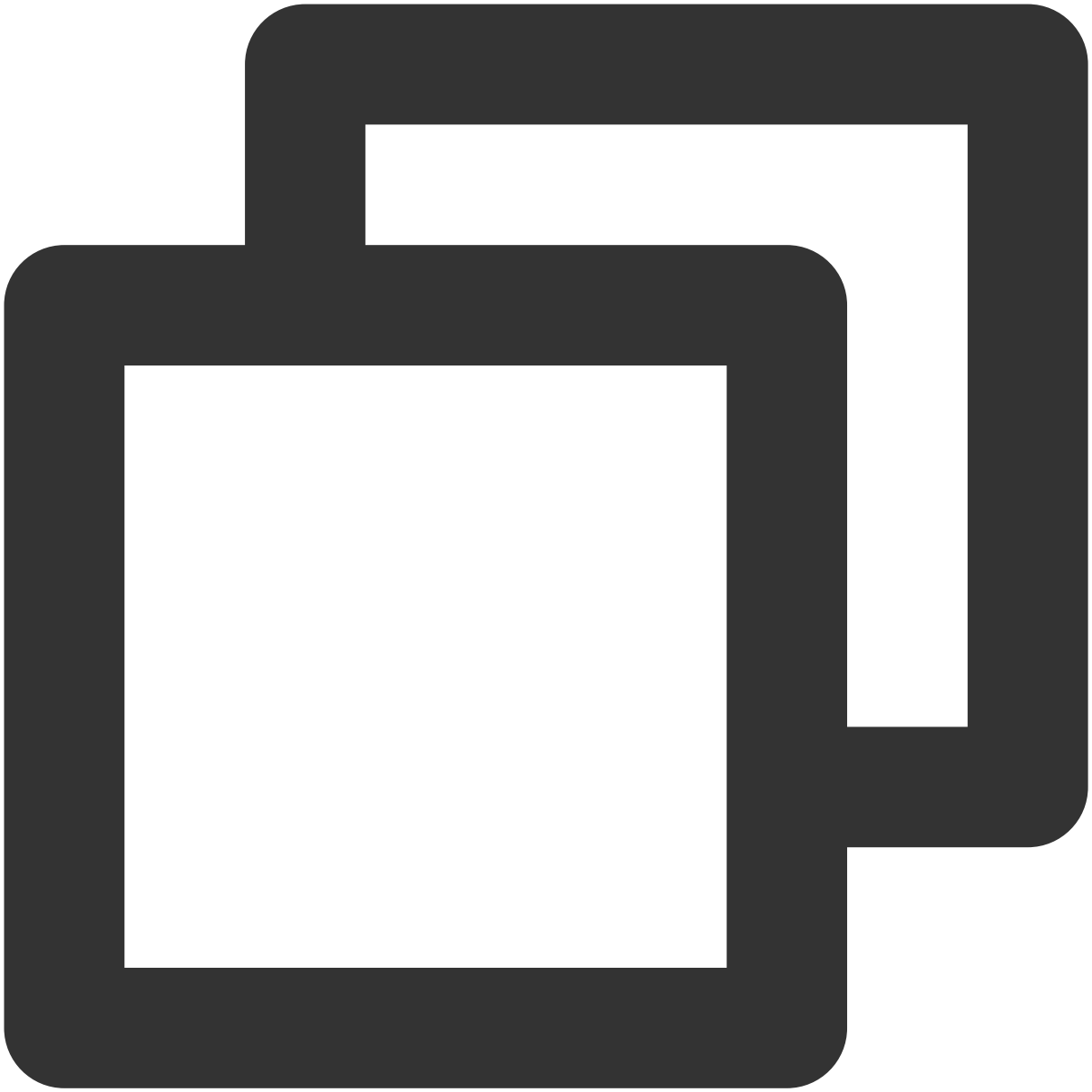
```
import { TUICallKitServer, TUICallType } from '@tencentcloud/call-uikit-react';
try {
  await TUICallKitServer.groupCall({
    userIDList: ['jack', 'tom'],
    groupID: 'xxx',
    type: TUICallType.VIDEO_CALL
  });
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the groupCall API. Reason:${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
userIDList	Array<String>	是	被呼叫的用户列表
type	TUICallType	是	通话的媒体类型，参数值说明参考TUICallType 通话类型
groupID	String	是	此次群组通话的群 ID，groupID 的创建可参考 chat-createGroup API
roomID	Number	否	数字房间号，范围 [1, 2147483647]
strRoomID	String	否	字符串房间号。**v3.3.1+ 支持** **范围：** 长度限制为64个字节。支持的字符集范围如下（共89个字符）： 小写和大写英文字母。（a-zA-Z）； 数字（0-9）； `空格`、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、 1. roomID 与 strRoomID 是互斥的，若您选用 strRoomID，则 roomID 需 2. 不要混用 roomID 和 strRoomID，因为它们之间是不互通的，比如数
timeout	Number	否	通话超时时间，默认：30s，单位：秒。timeout = 0，设置为不超时
userData	String	否	发起通话时自定义扩展字段，被呼叫用户在 ON_CALL_RECEIVED 事
offlinePushInfo	Object	否	自定义离线消息推送参数

setLanguage

设置语言，目前支持：中文、英文、日文。



```
TUICallKitServer.setLanguage("zh-cn"); // "en" | "zh-cn" | "ja_JP"
```

参数如下表所示：

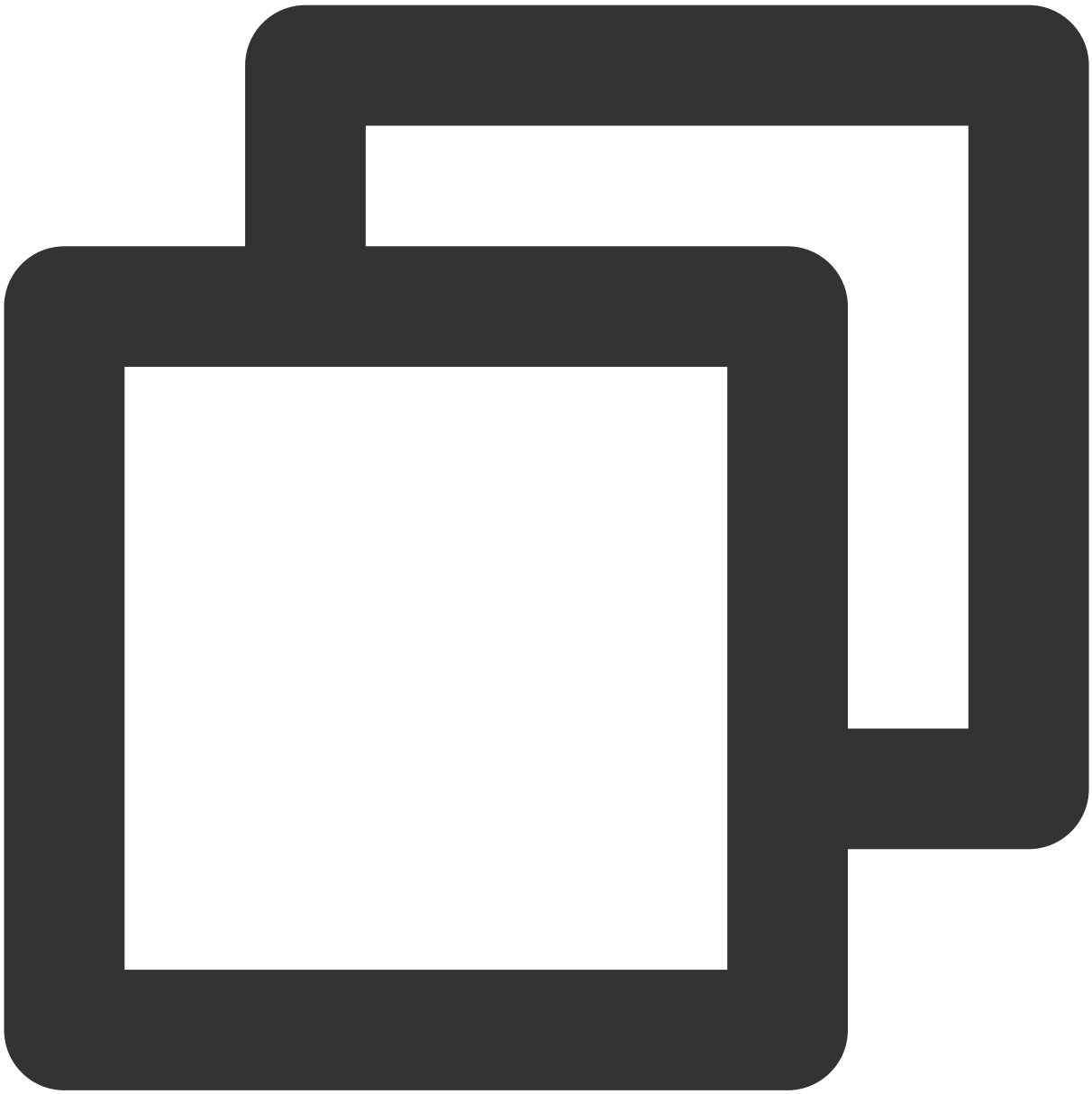
参数	类型	是否必填	含义
lang	String	是	语言类型 `en` 、 `zh-cn` 和 `ja_JP` 。

setSelfInfo

设置自己的昵称和头像。

注意：

v2.2.0+ 支持。通话中使用该接口修改用户信息，UI 不会立即更新，需要等到下次通话才能看到变化。



```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "xxx", avatar: "http://xxx" });
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`
}
```

参数如下表所示：

参数	类型	是否必填	含义

nickName	String	是	自己的昵称
avatar	String	是	自己头像地址

setCallingBell

注意：

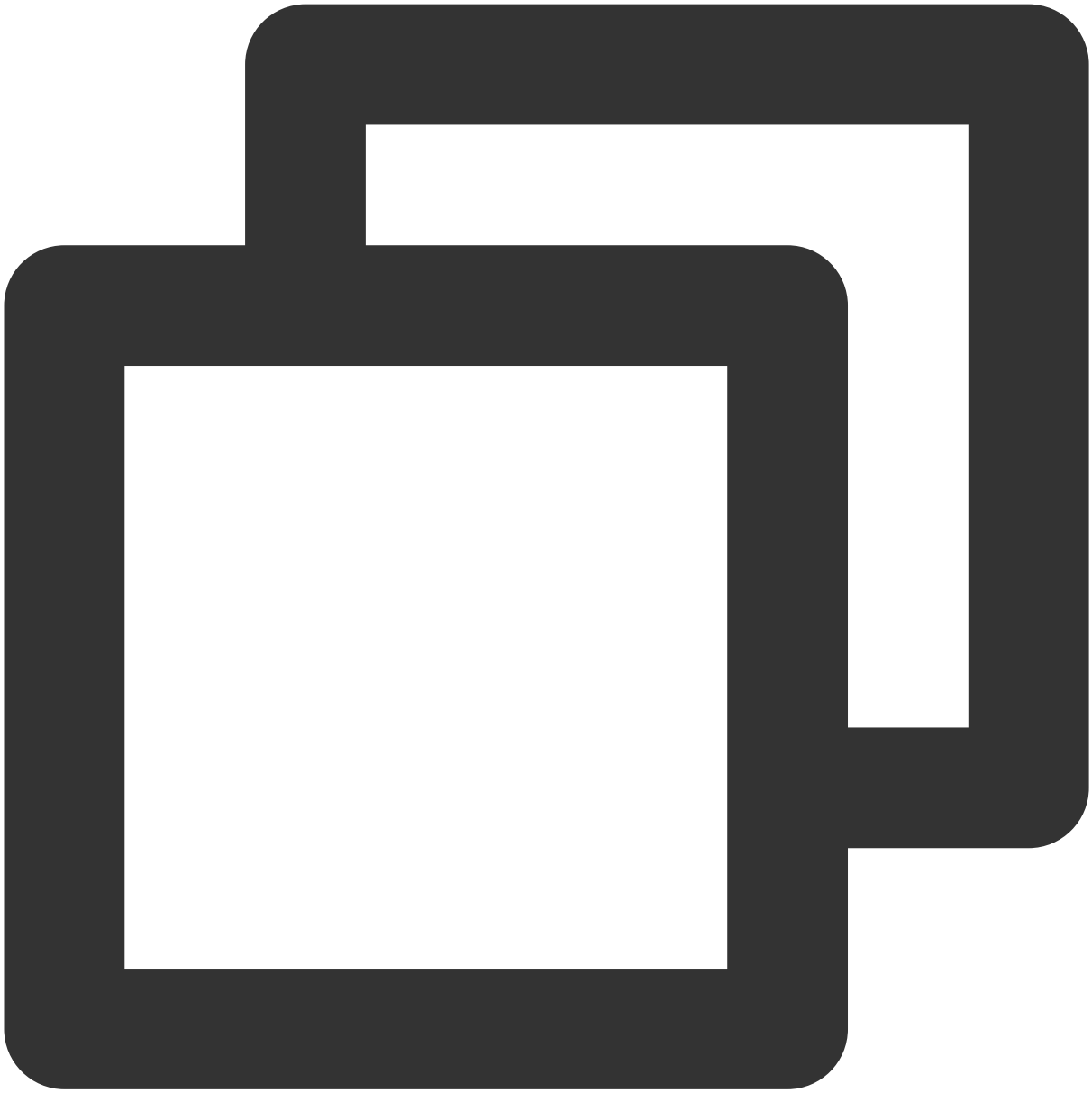
v3.0.0+ 支持。

自定义用户的来电铃音。

仅限传入本地 MP3 格式的文件地址，需要确保该文件目录是应用可以访问的。

使用 import 方式引入铃声文件。

如需恢复默认铃声，`filePath` 传空即可。



```
import filePath from '../assets/phone_ringing.mp3';
try {
  await TUICallKitServer.setCallingBell(filePath)
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the setCallingBell API. Reason: ${erro
}
```

参数如下表所示：

参数	类型	是否必填	含义

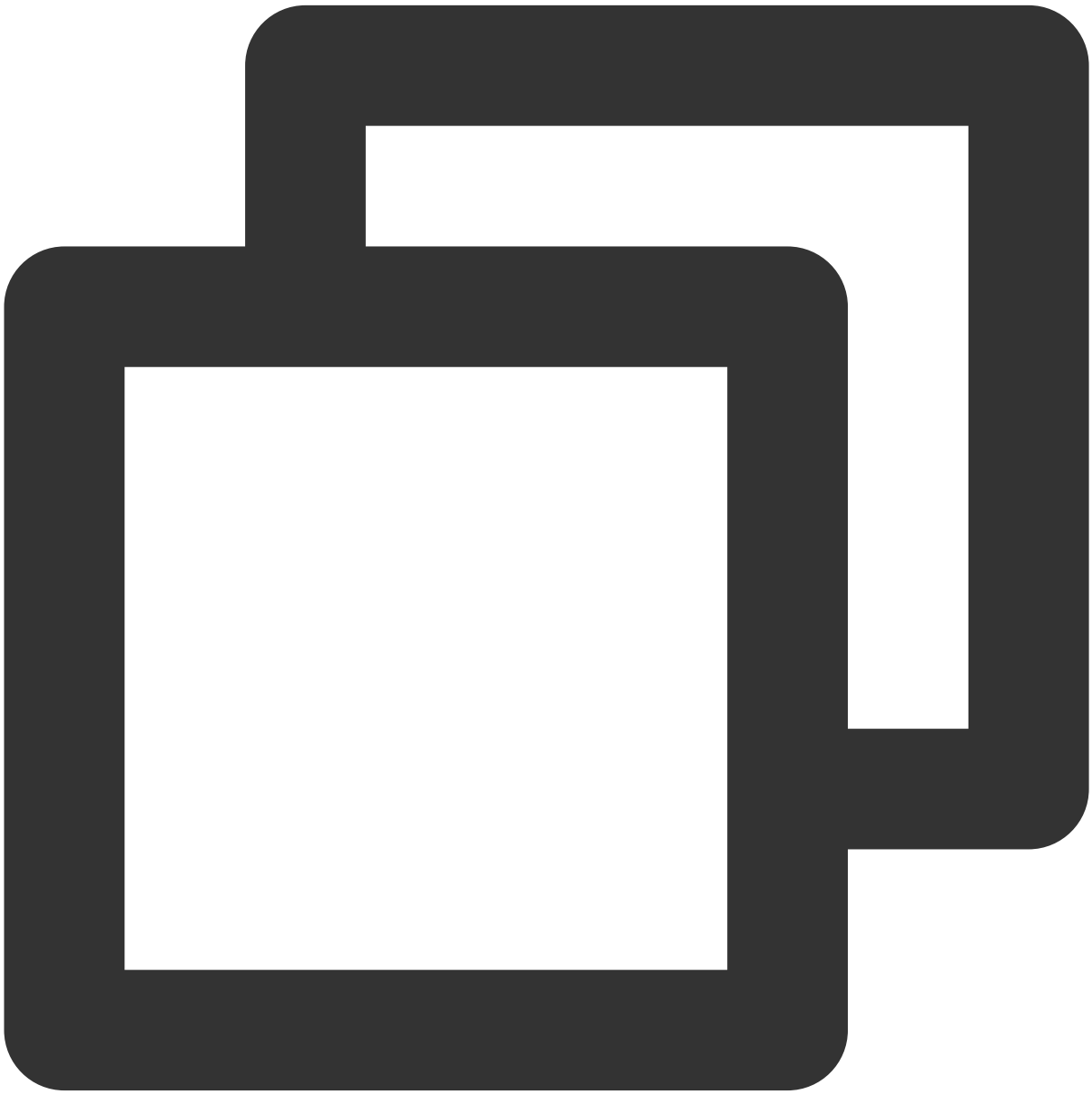
filePath	String	是	铃声文件地址

enableFloatWindow

开启/关闭悬浮窗功能。默认为false，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

注意：

v3.1.0+ 支持。



```
try {
  const enable = true;
  await TUICallKitServer.enableFloatWindow(enable);
```

```
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the enableFloatWindow API. Reason: ${e
}
```

参数如下表所示：

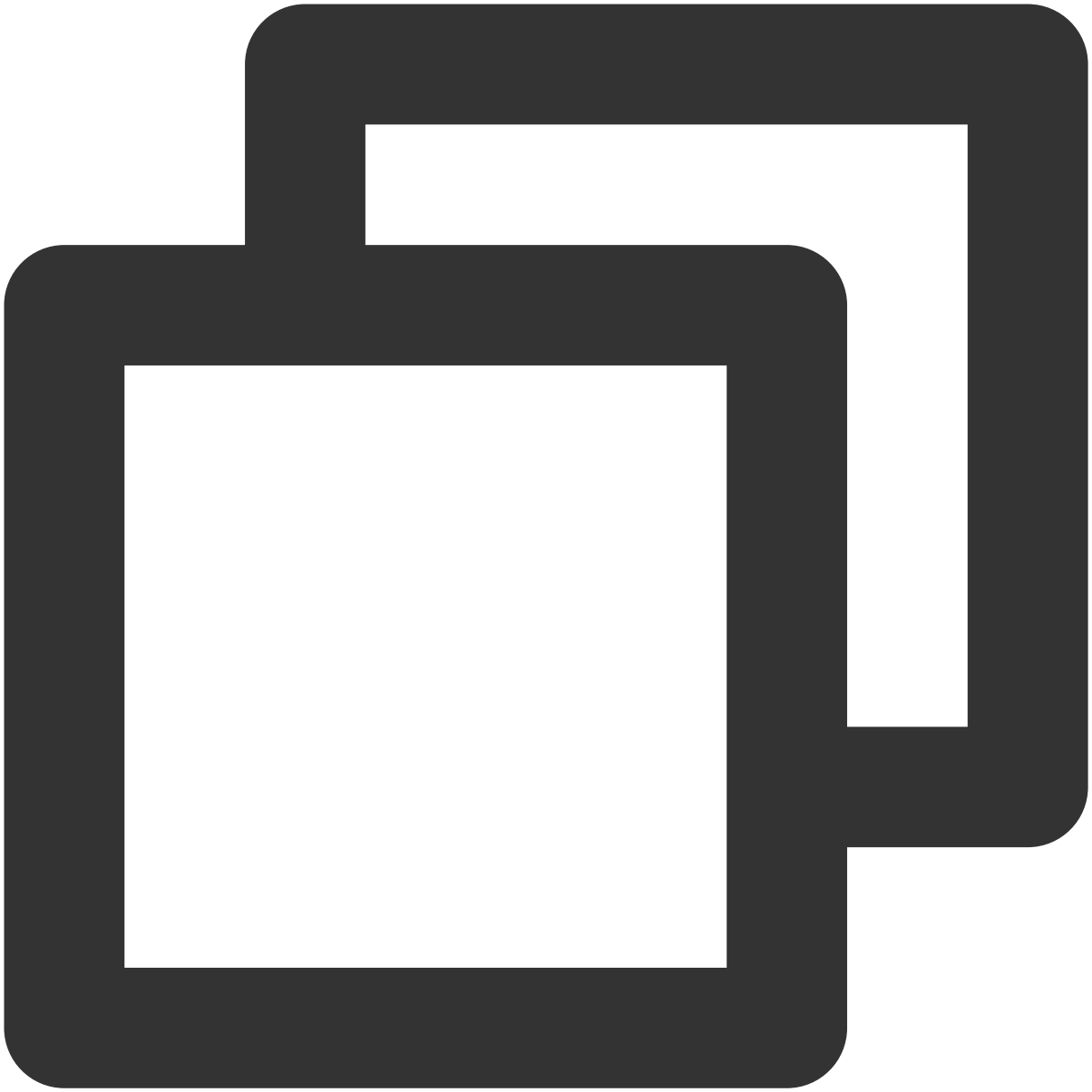
参数	类型	是否必填	含义
enable	Boolean	是	开启/关闭悬浮窗功能

enableMuteMode

开启/关闭来电铃声。开启后，收到通话请求时，不会播放来电铃声。

注意：

v3.1.2+ 支持



```
try {
  const enable = true;
  await TUICallKitServer.enableMuteMode(enable);
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${erro
}
```

参数如下表所示：

参数	类型	是否必填	含义

enable	Boolean	是	开启/关闭来电铃声。默认 false。

joinInGroupCall

加入群组中已有的音视频通话。

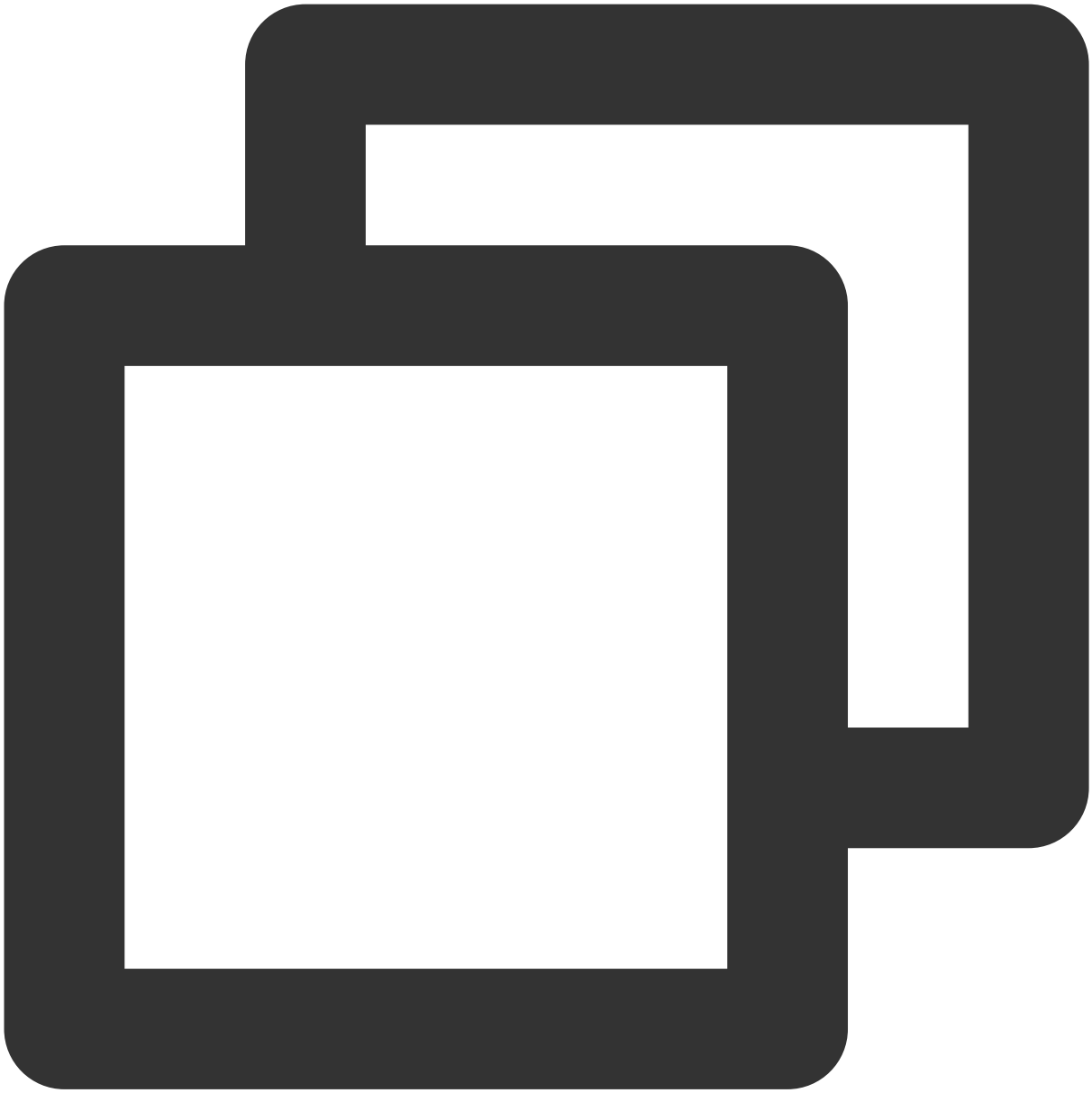
注意：

v3.1.2+ 支持。

说明：

加入群组中已有的音视频通话前，需要提前创建或加入IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。

群组的创建详见：IM 群组管理，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。



```
import { TUICallKitServer, TUICallType } from '@tencentcloud/call-uikit-react';
try {
  const params = {
    type: TUICallType.VIDEO_CALL,
    groupID: "xxx",
    roomID: 234,
  };
  await TUICallKitServer.joinInGroupCall(params);
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${erro
}
```

参数列表：

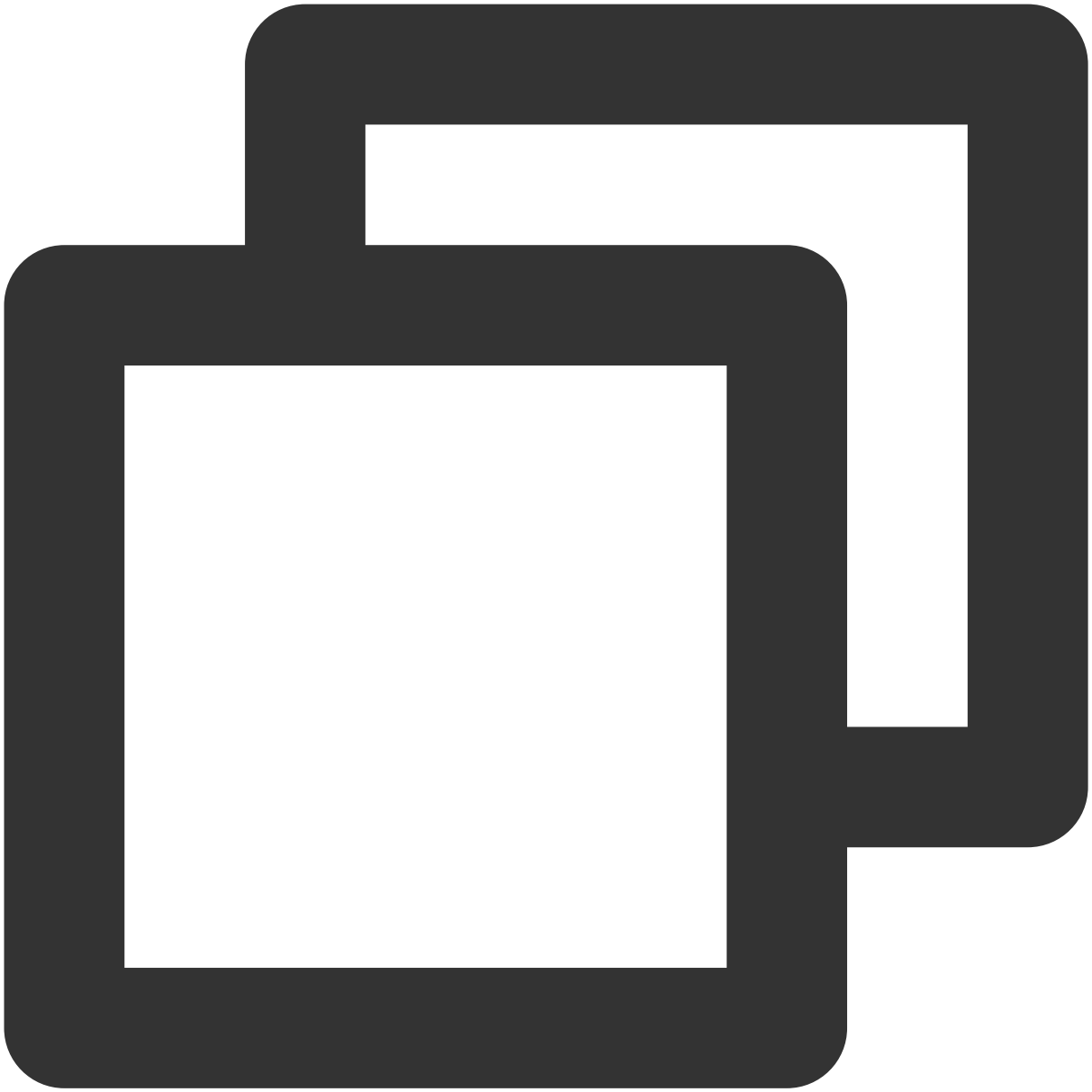
参数	类型	是否必填	含义
type	TUICallType	是	通话的媒体类型，例如视频通话、语音通话
groupID	string	是	此次群组通话的群 ID
roomID	number	是	此次通话的音视频房间 ID

enableVirtualBackground

开启/关闭模糊背景功能。如果想设置图片背景模糊参见 Web。通过调用接口，您可以在 UI 上显示模糊背景的功能按钮，点击按钮可直接启用模糊背景功能。

注意：

v3.2.4+ 支持。



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";
const enable = true;
TUICallKitServer.enableVirtualBackground(enable);
```

参数列表：

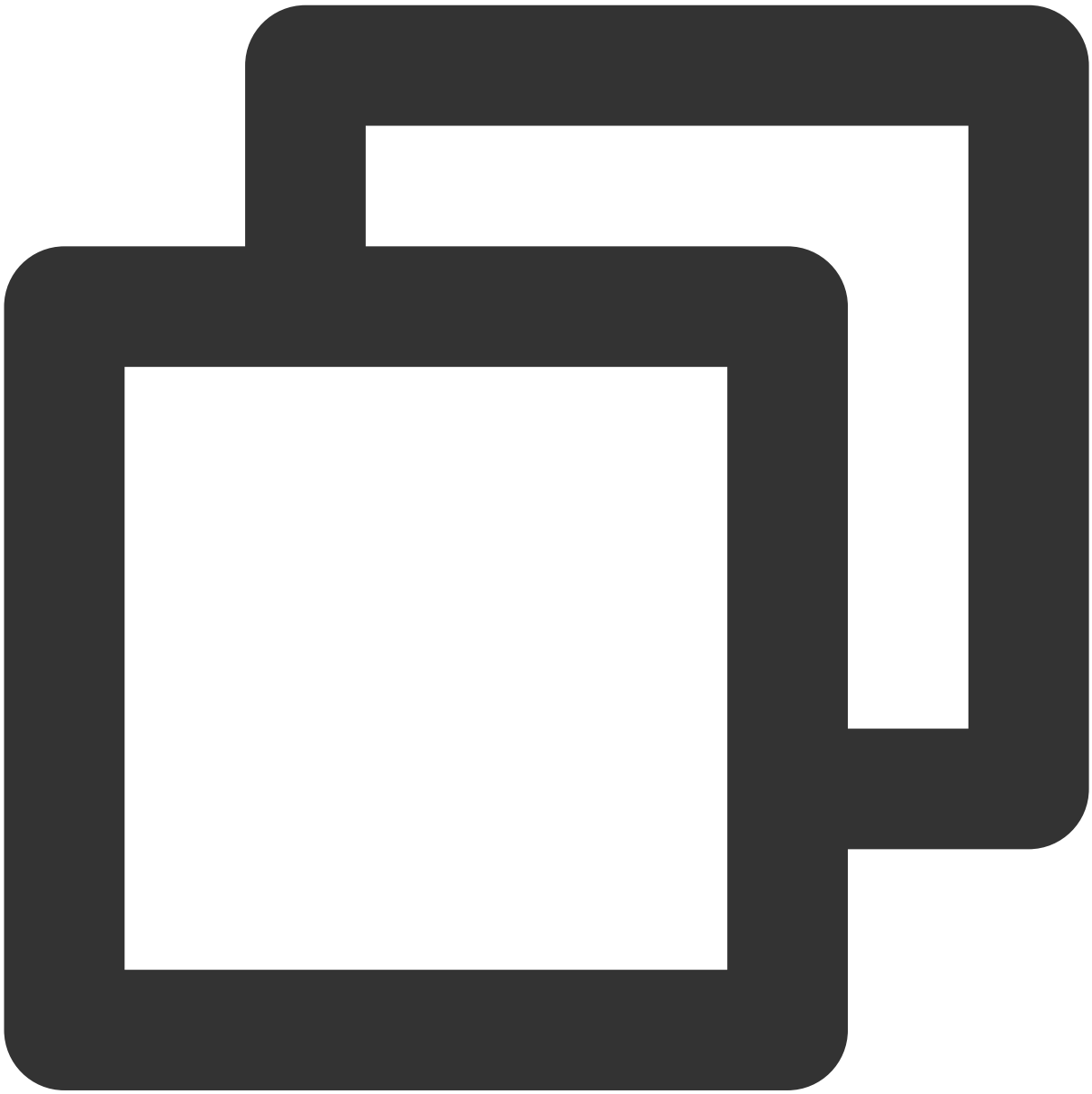
参数	类型	是否必填	含义
enable	boolean	是	enable = true 显示模糊背景按钮 enable = false 不显示模糊背景按钮

hideFeatureButton

隐藏功能按钮，目前仅支持 摄像头，麦克风和切换前后置按钮。

注意：

v3.2.9+ 支持



```
TUICallKitServer.hideFeatureButton(buttonName: FeatureButton);
```

参数列表：

参数	类型	是否必填	含义

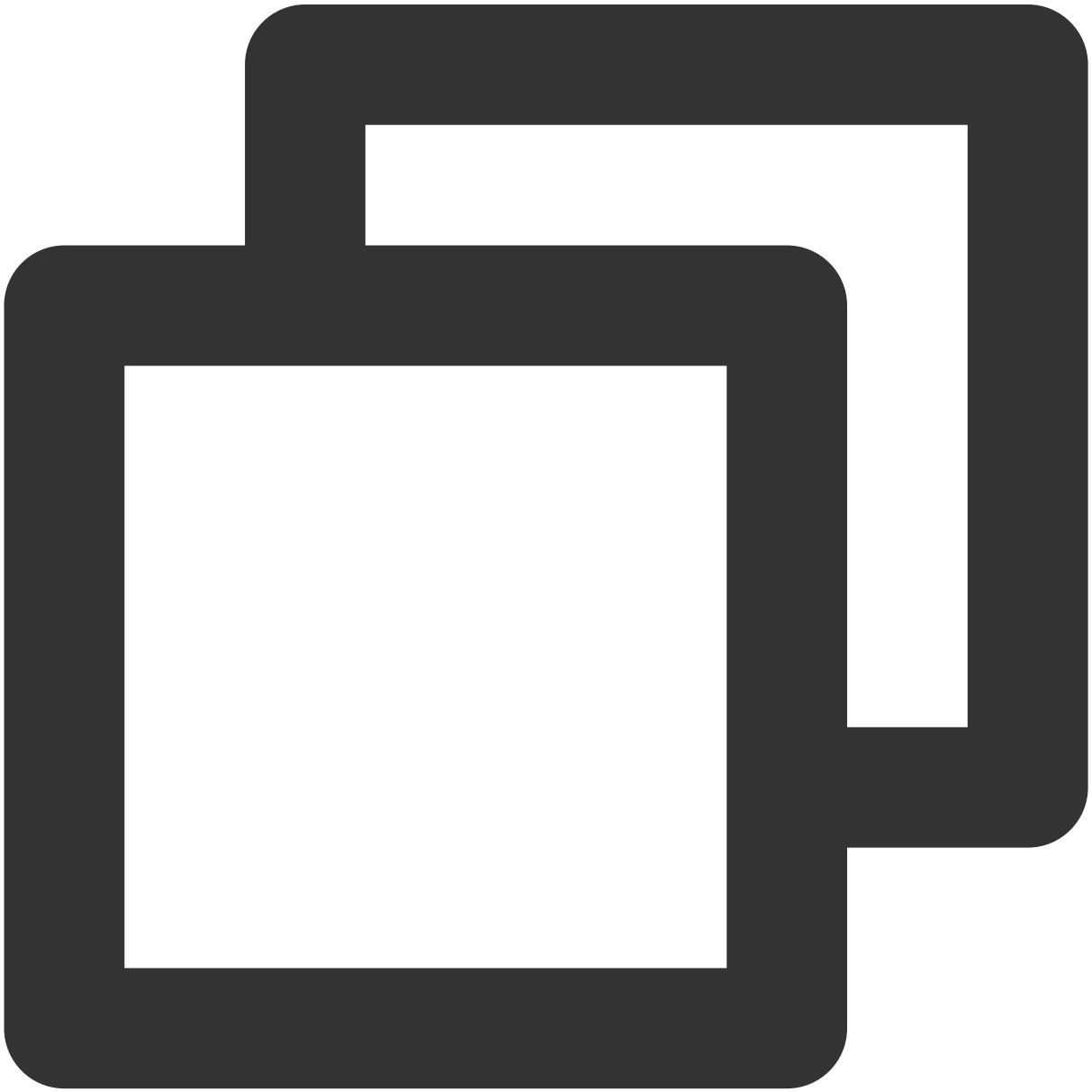
buttonName	FeatureButton	是	按钮名称

setLocalViewBackgroundImage

设置本地用户通话背景。

注意：

v3.2.9+ 支持



```
TUICallKitServer.setLocalViewBackgroundImage(url: string);
```

参数列表：

参数	类型	是否必填	含义
url	string	是	图片地址（支持本地路径和网络地址）

setRemoteViewBackgroundImage

设置远端用户通话背景。

注意：

v3.2.9+ 支持

```
TUICallKitServer.setRemoteViewBackgroundImage(userId: string, url: string);
```

参数列表：

参数	类型	是否必填	含义
userId	string	是	远端用户 userId，设置为 '*' 表示对所有远端用户生效
url	string	是	图片地址（支持本地路径和网络地址）

setLayoutMode

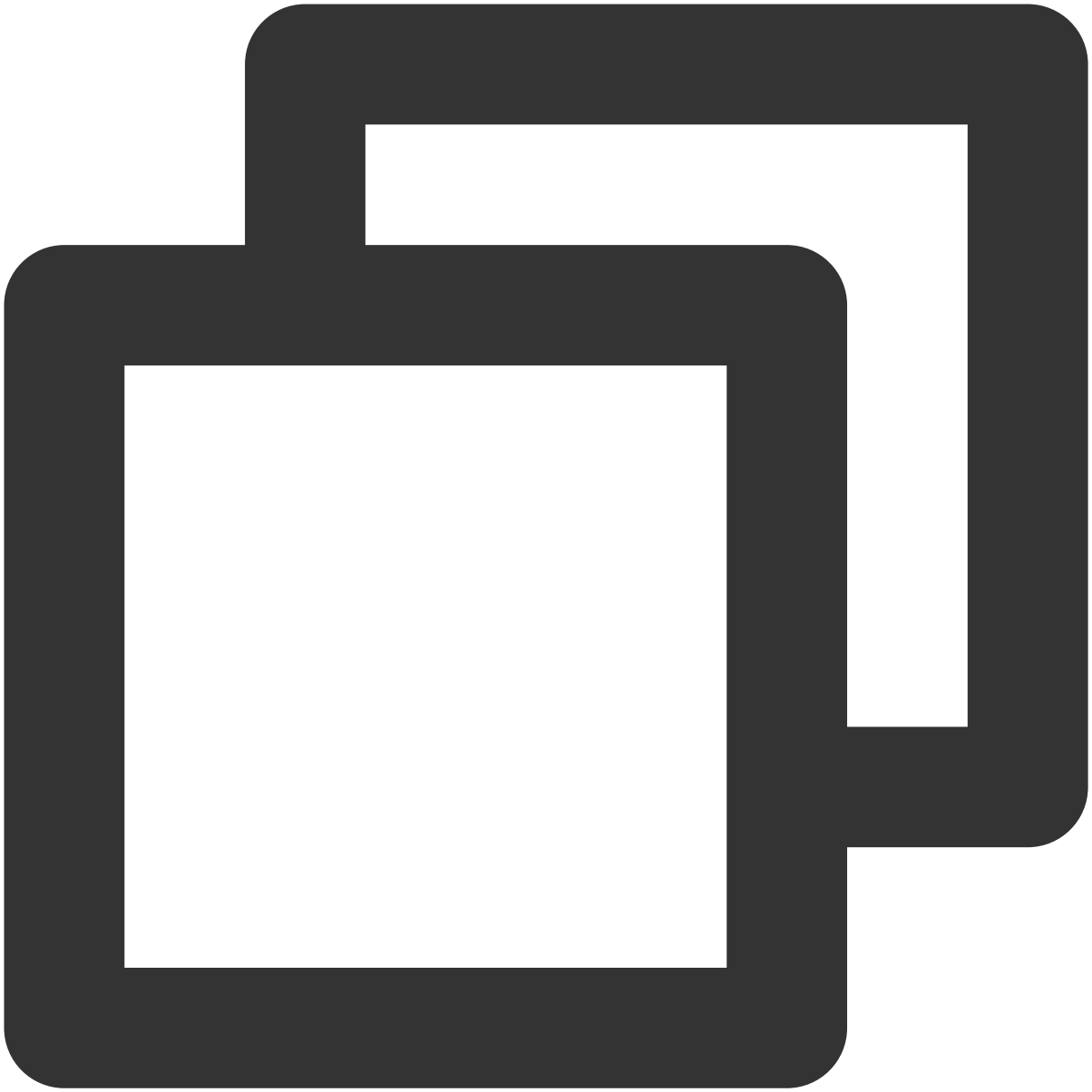
设置通话界面布局模式。

注意：

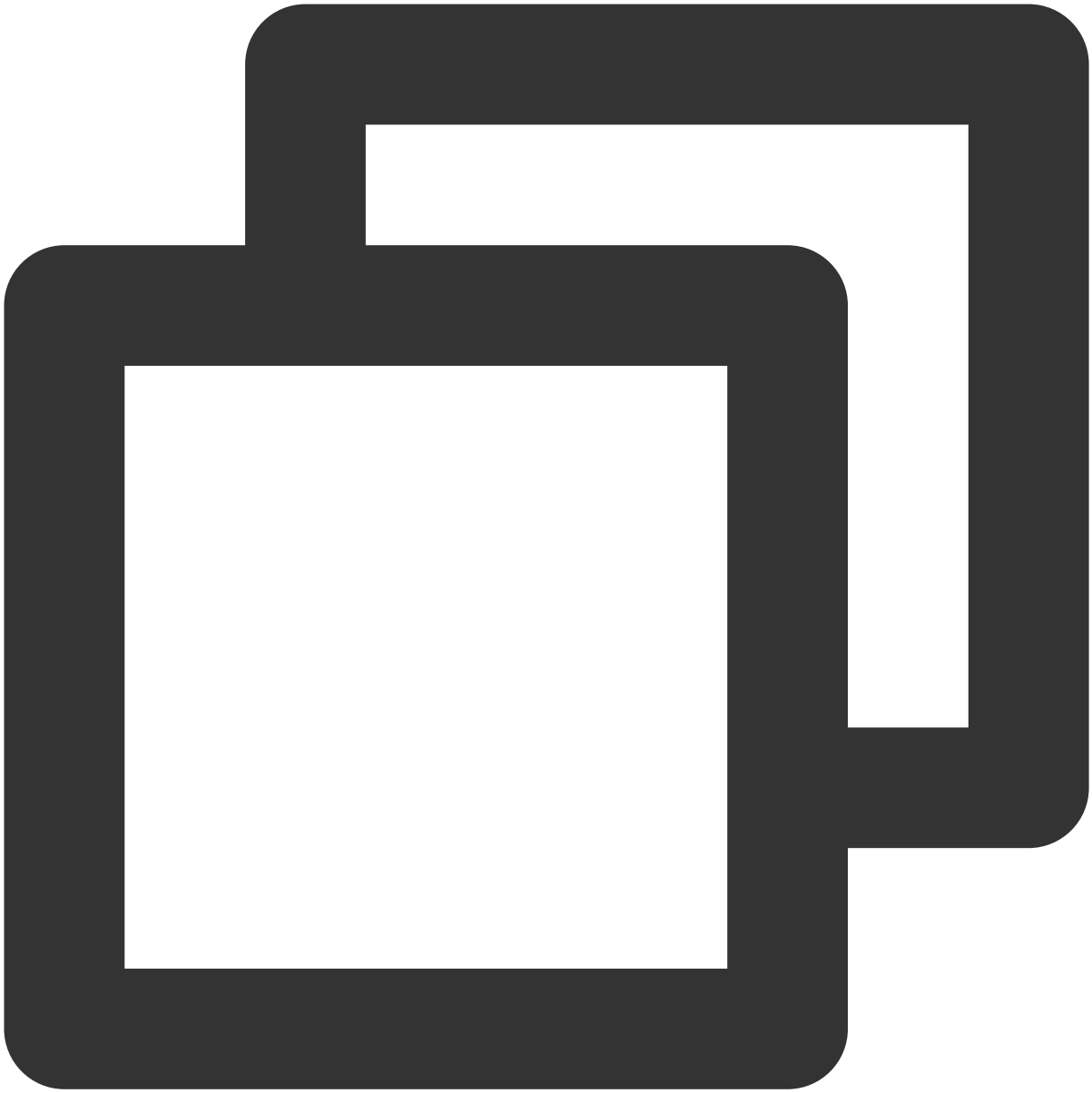
v3.3.0+ 支持。

Vue

React



```
import { TUICallKitServer, LayoutMode } from "@tencentcloud/call-uikit-vue";
TUICallKitServer.setLayoutMode(LayoutMode.LocalInLargeView);
```



```
import { TUICallKitServer, LayoutMode } from "@tencentcloud/call-uikit-react";
TUICallKitServer.setLayoutMode(LayoutMode.LocalInLargeView);
```

参数列表：

参数	类型	是否必填	含义
layoutMode	LayoutMode	是	用户流的布局模式

setCameraDefaultState

设置摄像头是否默认打开。

注意：

v3.3.0+ 支持。

```
TUICallKitServer.setCameraDefaultState(true);
```

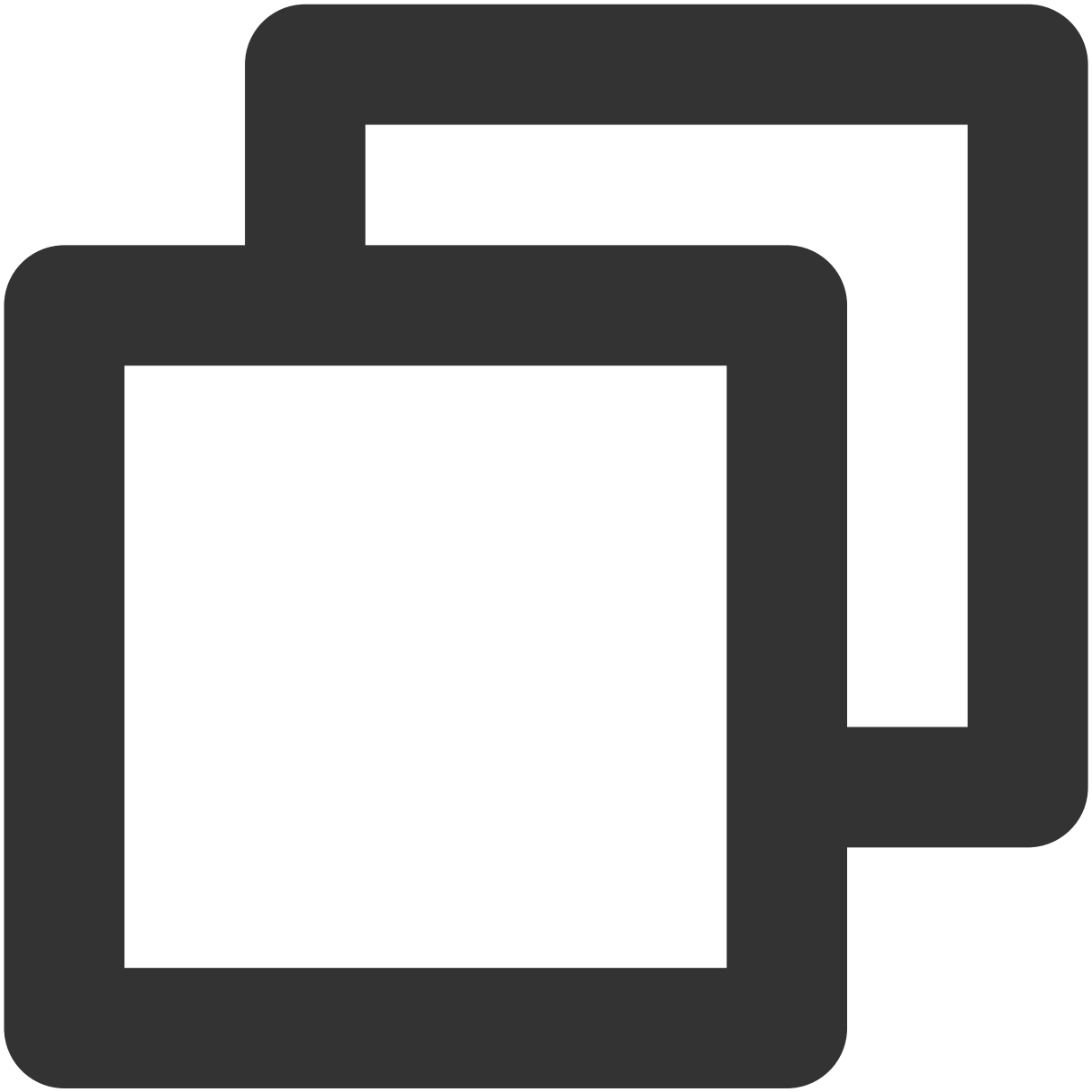
参数列表：

参数	类型	是否必填	含义
isOpen	boolean	是	是否开启摄像头

destroyed

销毁 TUICallKit 实例。

该方法不会自动退出 `tim` ，需要手动退出 `tim.logout();` 。



```
try {
  await TUICallKitServer.destroyed();
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the destroyed API. Reason: ${error}`);
}
```

TUICallKit 类型定义

videoDisplayMode

画面显示模式 `videoDisplayMode` 属性有三个值：

`VideoDisplayMode.CONTAIN`

`VideoDisplayMode.COVER`

`VideoDisplayMode.FILL` ，默认值是 `VideoDisplayMode.COVER` 。

属性	值	描述
videoDisplayMode	VideoDisplayMode.CONTAIN	优先保证视频内容全部显示。 视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。 如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
	VideoDisplayMode.COVER	优先保证视窗被填满。 视频尺寸等比缩放，直至整个视窗被视频填满。 如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
	VideoDisplayMode.FILL	保证视窗被填满的同时保证视频内容全部显示，但是不保证视频尺寸比例不变。 视频的宽高会被拉伸至和视窗尺寸一致。

videoResolution

分辨率 `videoResolution` 有三个值：

`VideoResolution.RESOLUTION_480P`

`VideoResolution.RESOLUTION_720P`

`VideoResolution.RESOLUTION_1080P` ，默认值是 `VideoResolution.RESOLUTION_480P` 。

分辨率说明：

视频 Profile	分辨率（宽 × 高）	帧率（fps）	码率（kbps）
480p	640 × 480	15	900
720p	1280 × 720	15	1500
1080p	1920 × 1080	15	2000

常见问题：

iOS 13&14不支持编码高于 720P 的视频，建议在这两个系统版本限制最高采集 720P。参见 iOS Safari 已知问题 case 12。

Firefox 不支持自定义视频帧率（默认为 30fps）。

受系统性能占用，摄像头采集能力和浏览器限制等因素的影响，视频分辨率、帧率、码率的实际值不一定能够完全匹配设定值，在这种情况下，浏览器会自动调整 Profile 尽可能匹配设定值。

STATUS

STATUS 属性值	描述
STATUS.IDLE	闲置状态
STATUS.BE_INVITED	收到通话邀请
STATUS.DIALING_C2C	正在 1v1 呼叫
STATUS.DIALING_GROUP	正在群组呼叫
STATUS.CALLING_C2C_AUDIO	正在 1v1 语音通话
STATUS.CALLING_C2C_VIDEO	正在 1v1 视频通话
STATUS.CALLING_GROUP_AUDIO	正在群组语音通话
STATUS.CALLING_GROUP_VIDEO	正在群组视频通话

TUICallType

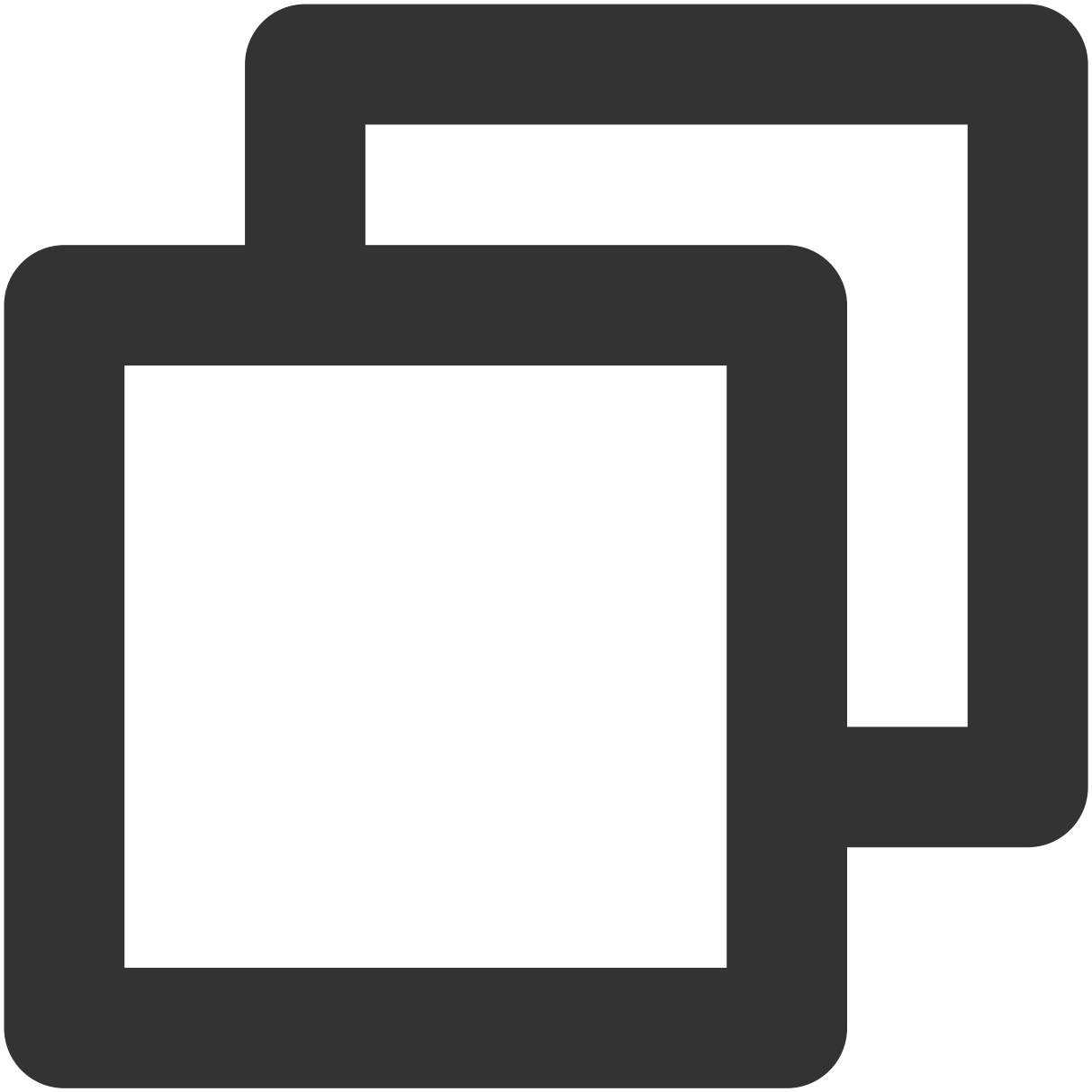
TUICallType 类型	描述
TUICallType.AUDIO_CALL	语音通话
TUICallType.VIDEO_CALL	视频通话

offlinePushInfo

参数	类型	是否必填	含义
offlinePushInfo.title	String	否	离线推送标题（选填）
offlinePushInfo.description	String	否	离线推送内容（选填）
offlinePushInfo.androidOPPOChannelID	String	否	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）

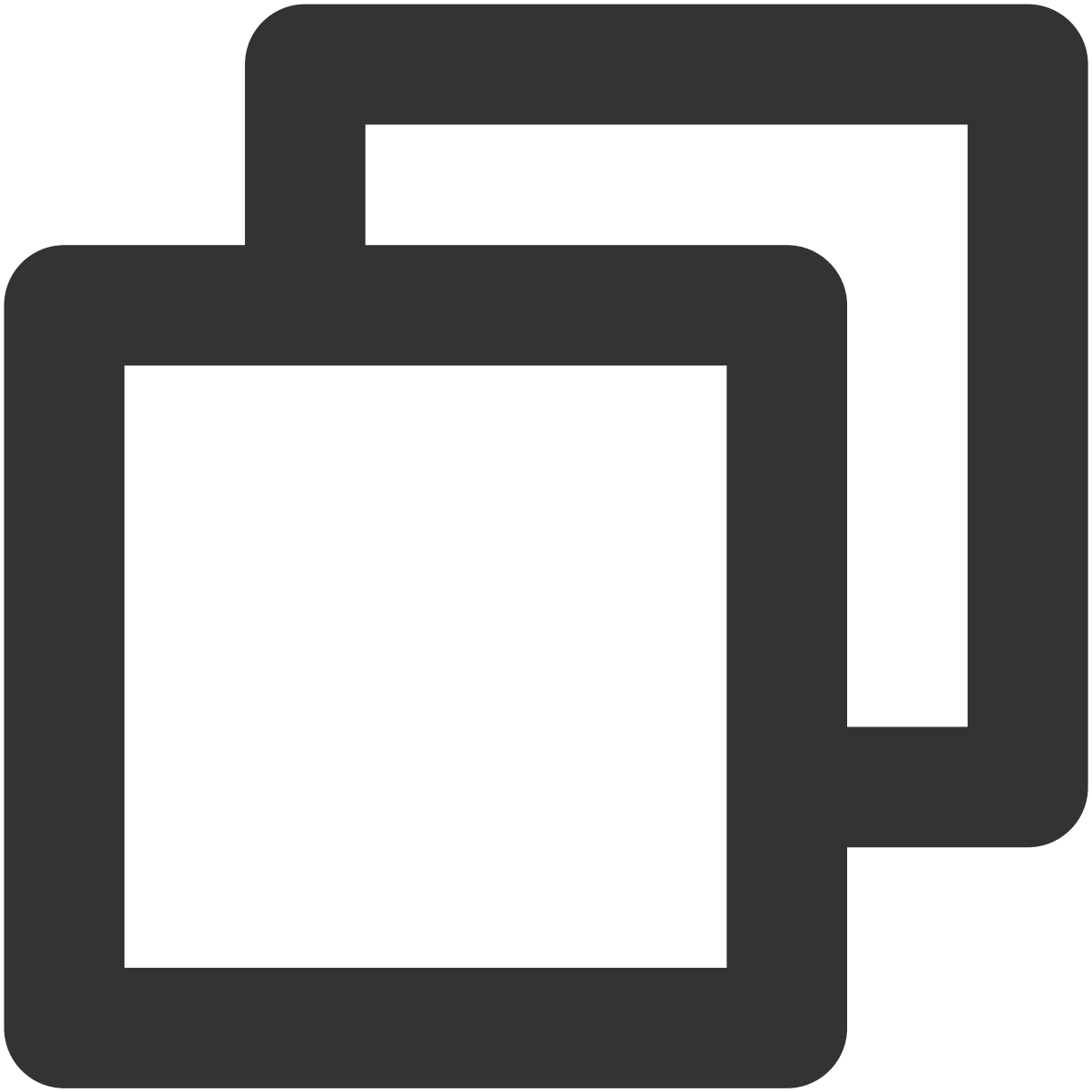
offlinePushInfo.extension	String	否	离线推送透传内容，可以用于设置 Android Notification 模式 和 VoIP 模式。**默认：** Notification 模式，会是系统的通知；VoIP 模式是需要传字段。
offlinePushInfo.ignoreIOSBadge	Boolean	否	离线推送忽略 badge 计数（仅对 iOS 生效）， 如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加 **v3.3.0+ 支持**
offlinePushInfo.iOSSound	String	否	离线推送声音设置(仅对 iOS 生效)。 **v3.3.0+ 支持**
offlinePushInfo.androidSound	String	否	离线推送声音设置（仅对 Android 生效）。 **v3.3.0+ 支持**
offlinePushInfo.androidVIVOClassification	Number	否	VIVO 推送消息分类（已弃用的接口，VIVO 推送服务将在 2023 年 4 月 3 日优化消息分类规则。建议使用 setAndroidVIVOCategory 设置消息类别）。0：运营消息，1：系统消息。默认值为 1。**v3.3.0+ 支持**
offlinePushInfo.androidXiaoMiChannelID	String	否	小米手机 8.0 系统及以上的渠道 ID。 **v3.3.0+ 支持**
offlinePushInfo.androidFCMChannelID	String	否	FCM 通道手机 8.0 系统及以上的渠道 ID。 **v3.3.0+ 支持**
offlinePushInfo.androidHuaWeiCategory	String	否	华为推送消息分类。**v3.3.0+ 支持**
offlinePushInfo.isDisablePush	Boolean	否	是否关闭推送（默认开启推送）。**v3.3.0+ 支持**
offlinePushInfo.iOSPushType	Number	否	iOS 离线推送类型。0-普通推送；1-VoIP推送。默认：0。**v3.3.0+ 支持**

Android Notification 模式



```
const extension = {
  timPushFeatures: {
    fcmPushType: 0, // 0, VoIP; 1, notification
  }
};
offlinePushInfo.extension = JSON.stringify(extension);
```

Android VoIP 模式



```
const extension = {
  timPushFeatures: {
    fcmPushType: 0, // 0, data; 1, notification
    fcmNotificationType: 1, // 0, TIMPush implementation； 1, business implementati
  }
};
offlinePushInfo.extension = JSON.stringify(extension);
```

FeatureButton

FeatureButton 类型	描述
FeatureButton.Camera	摄像头按钮
FeatureButton.Microphone	麦克风按钮
FeatureButton.SwitchCamera	切换前后置摄像头按钮

LayoutMode

LayoutMode 类型	描述
LayoutMode.LocalInLargeView	本地用户在大窗显示
LayoutMode.RemoteInLargeView	远端用户在大窗显示

版权所有：腾讯云计算（北京）有限责任公司

TUICallEngine

最近更新时间：2024-08-09 17:55:21

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
on	监听事件
off	取消监听事件
login	登录接口
logout	登出接口
setSelfInfo	设置用户昵称和头像
call	发起 1v1 通话
groupCall	群聊邀请通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
switchCallMediaType	音视频通话切换
startRemoteView	启动远端画面渲染
stopRemoteView	停止远端画面渲染
startLocalView	启动本地画面渲染，**注意：后续将废弃，建议使用 openCamera**

版权所有：腾讯云计算（北京）有限责任公司

stopLocalView	停止本地画面渲染，**注意：后续将废弃，建议使用 closeCamera**
openCamera	开启摄像头
closeCamara	关闭摄像头
switchCamera	切花前后置摄像头，**注意：仅移动端支持。v3.0.0+ 支持**
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
setVideoQuality	设置视频质量
getDeviceList	获取设备列表
switchDevice	切换摄像头或麦克风设备
enableAIVoice	开启/关闭 AI 降噪
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式。**v2.1.1+ 支持**
setBlurBackground	开关/设置背景虚化，**v3.0.6+ 支持**
setVirtualBackground	开关/设置图片背景虚化，**v3.0.6+ 支持**

版权所有：腾讯云计算（北京）有限责任公司

TUICallEvent

最近更新时间：2024-08-09 17:56:57

TUICallEvent API 简介

TUICallEvent API 是音视频通话组件的**事件接口**。

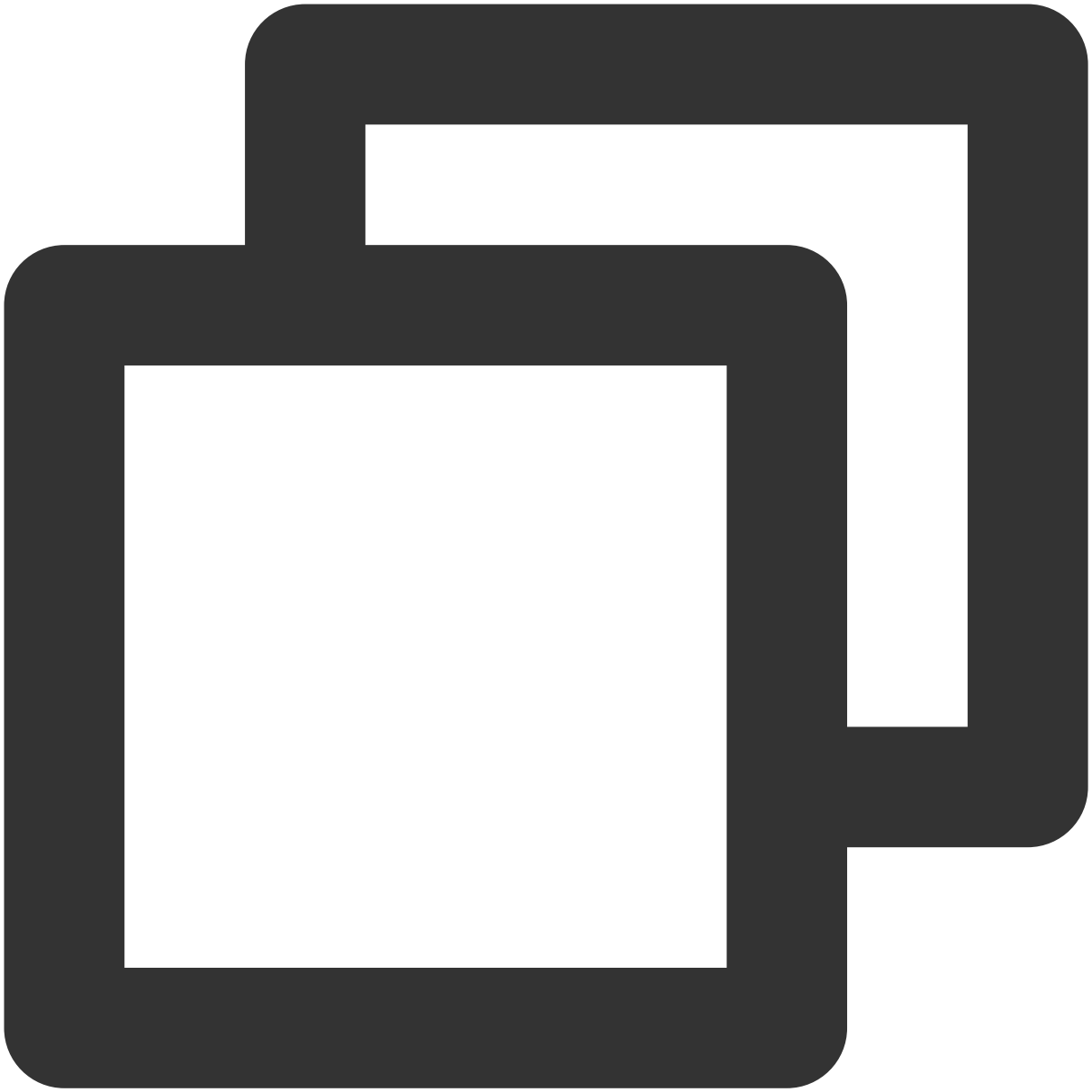
事件列表

EVENT	描述
TUICallEvent.ERROR	通话过程中错误事件
TUICallEvent.SDK_READY	SDK 进入 ready 状态时收到该事件
TUICallEvent.KICKED_OUT	用户重复登录，被踢下线
TUICallEvent.USER_ACCEPT	xxxx 用户接通的事件
TUICallEvent.USER_ENTER	xxxx 用户加入通话的事件
TUICallEvent.USER_LEAVE	xxxx 用户离开通话的事件
TUICallEvent.REJECT	xxxx 用户拒绝通话的事件
TUICallEvent.NO_RESP	xxxx 用户不响应的事件
TUICallEvent.LINE_BUSY	xxxx 用户忙线的事件
TUICallEvent.USER_VIDEO_AVAILABLE	xxxx 用户是否有视频流的事件
TUICallEvent.USER_AUDIO_AVAILABLE	xxxx 用户是否有音频流的事件
TUICallEvent.USER_VOICE_VOLUME	所有用户音量大小的反馈事件
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	群聊更新邀请列表收到该事件
TUICallEvent.ON_CALL_BEGIN	通话接通的事件，**v1.4.6+ 支持**
TUICallEvent.INVITED	通话请求的事件， **后续废弃，建议使用 TUICallEvent.ON_CALL_RECEIVED**

TUICallEvent.ON_CALL_RECEIVED	通话请求的事件，**v1.4.6+ 支持**
TUICallEvent.CALLING_CANCEL	通话取消的事件，**后续废弃，建议使用 TUICallEvent.ON_CALL_CANCELED**
TUICallEvent.ON_CALL_CANCELED	通话取消的事件，**v1.4.6+ 支持**
TUICallEvent.ON_CALL_BEGIN	通话接通时抛出的事件
TUICallEvent.CALLING_END	通话结束的事件
TUICallEvent.DEVICED_UPDATED	设备列表更新的事件
TUICallEvent.CALL_TYPE_CHANGED	通话的媒体类型发生改变的回调
TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED	所有用户网络质量的事件，**v3.0.7+ 支持**

ERROR

通话过程中错误事件，可以通过监听该事件，捕获通话过程中的内部错误。



```
let onError = function(error) {
  console.log(error.code, error.msg);
};
tuiCallEngine.on(TUICallEvent.ERROR, onError);
```

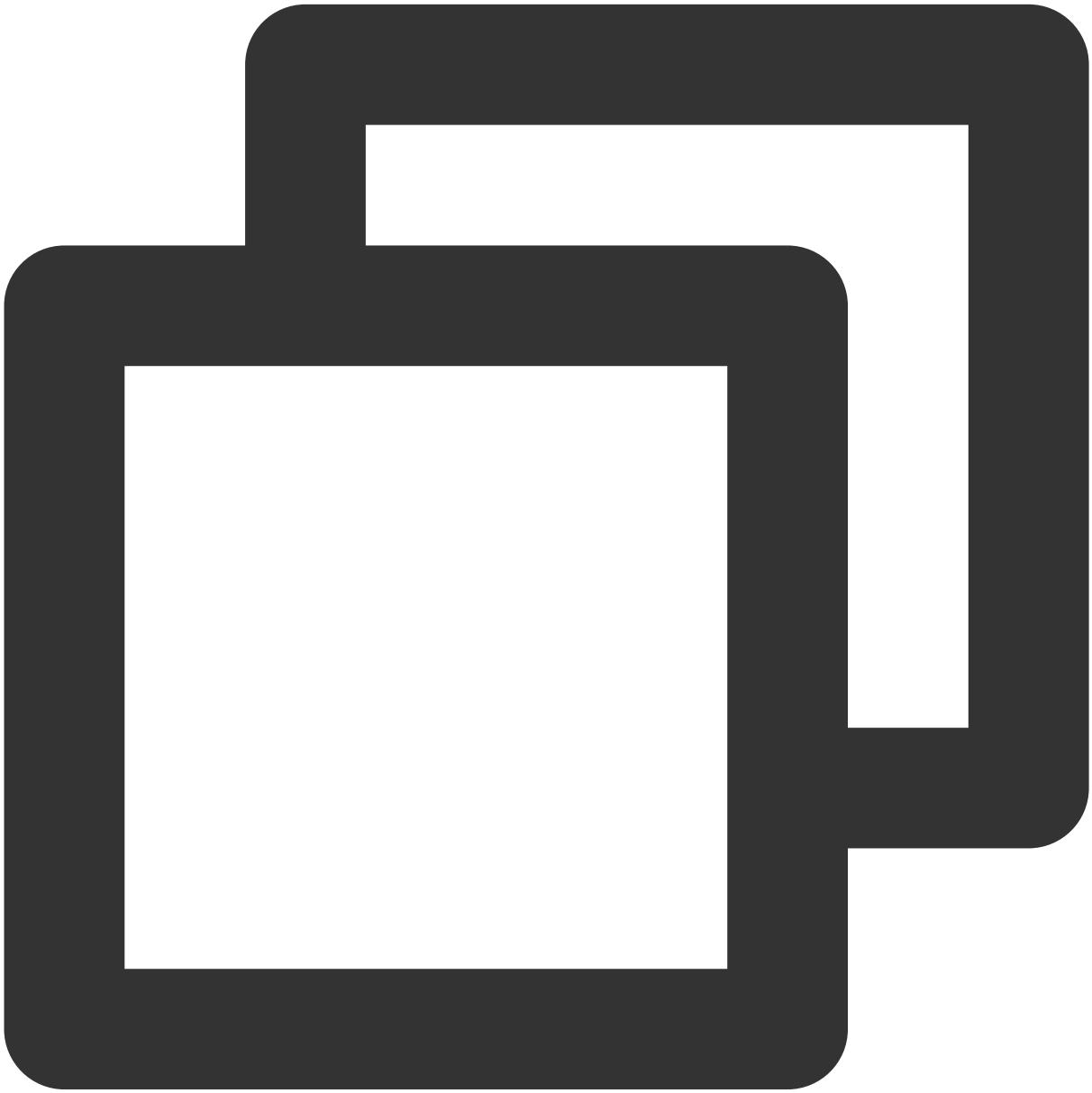
参数如下表所示：

参数	类型	含义
code	Number	错误码

msg	String	错误信息

SDK_READY

TUICallEngine 依赖 @tencentcloud/chat SDK，登录成功后才会出发 SDK_READY 事件，然后才能使用 SDK 的各项功能。

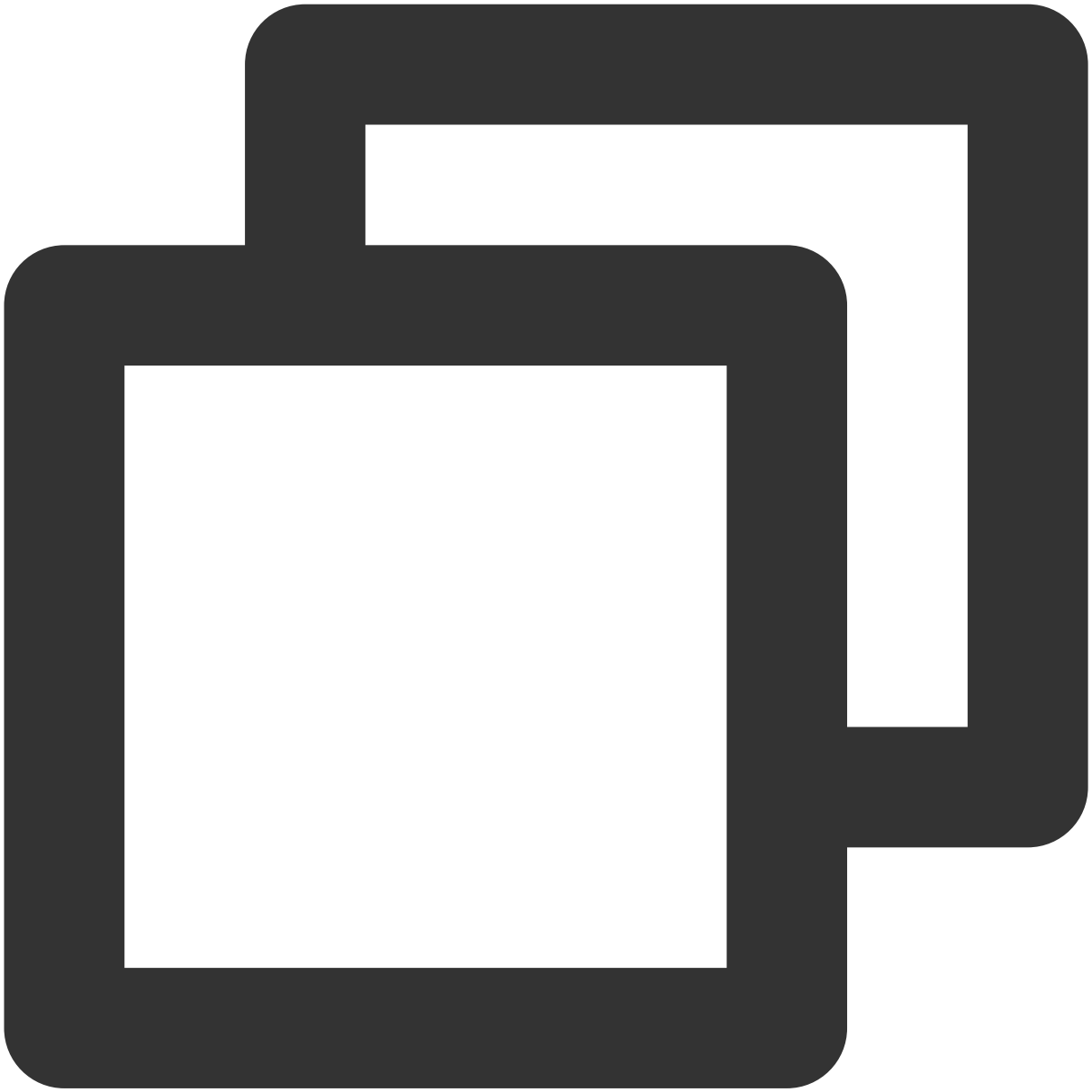


```
let onSDKReady = function(event) {
  console.log(event);
};
tuiCallEngine.on(TUICallEvent.SDK_READY, onSDKReady);
```

版权所有：腾讯云计算（北京）有限责任公司

KICKED_OUT

当前用户被踢下线，此时可以 UI 提示用户，并再次重新调用登录。



```
let handleOnKickedOut = function(event) {
  console.log(event);
};
tuiCallEngine.on(TUICallEvent.KICKED_OUT, handleOnKickedOut);
```

USER_ACCEPT

如果有用户接通，其他用户都会抛出该事件，userID 就是接通的用户名。

1. 1v1 通话下：被叫接通，主叫会抛出该事件。
2. 群组通话下：A 呼叫 B/C，B 接通，A/C 都会抛出该事件，事件的 userID 都是 B。同理 C 接通，A、B 都会抛出该事件，事件的 userID 都是 C。

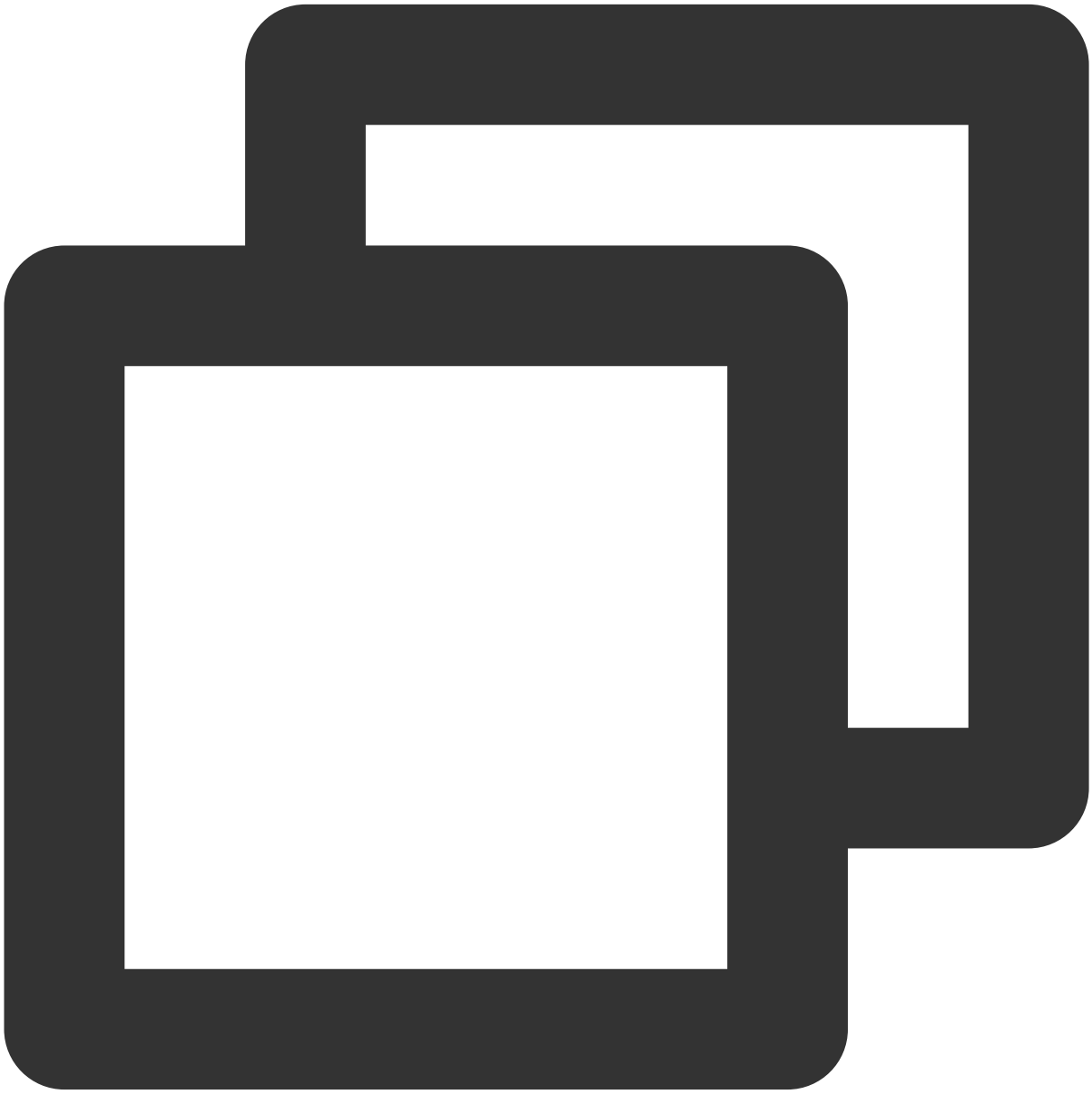
```
let handleUserAccept = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.USER_ACCEPT, handleUserAccept);
```

参数如下表所示：

参数	类型	含义
userID	String	接通通话的用户 ID

USER_ENTER

如果有用户进入通话，其他用户都会抛出该事件，userID 就是进入通话的用户名。



```
let handleUserEnter = function(event) {
  console.log(event.userID);
};
```

```
tuiCallEngine.on(TUICallEvent.USER_ENTER, handleUserEnter);
```

参数如下表所示：

参数	类型	含义
userID	String	进入通话的用户 ID

USER_LEAVE

有用户离开此次通话时，**通话中**的其他用户都会抛出该事件，userID 就是离开通话的用户名。

```
let handleUserLeave = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.USER_LEAVE, handleUserLeave);
```

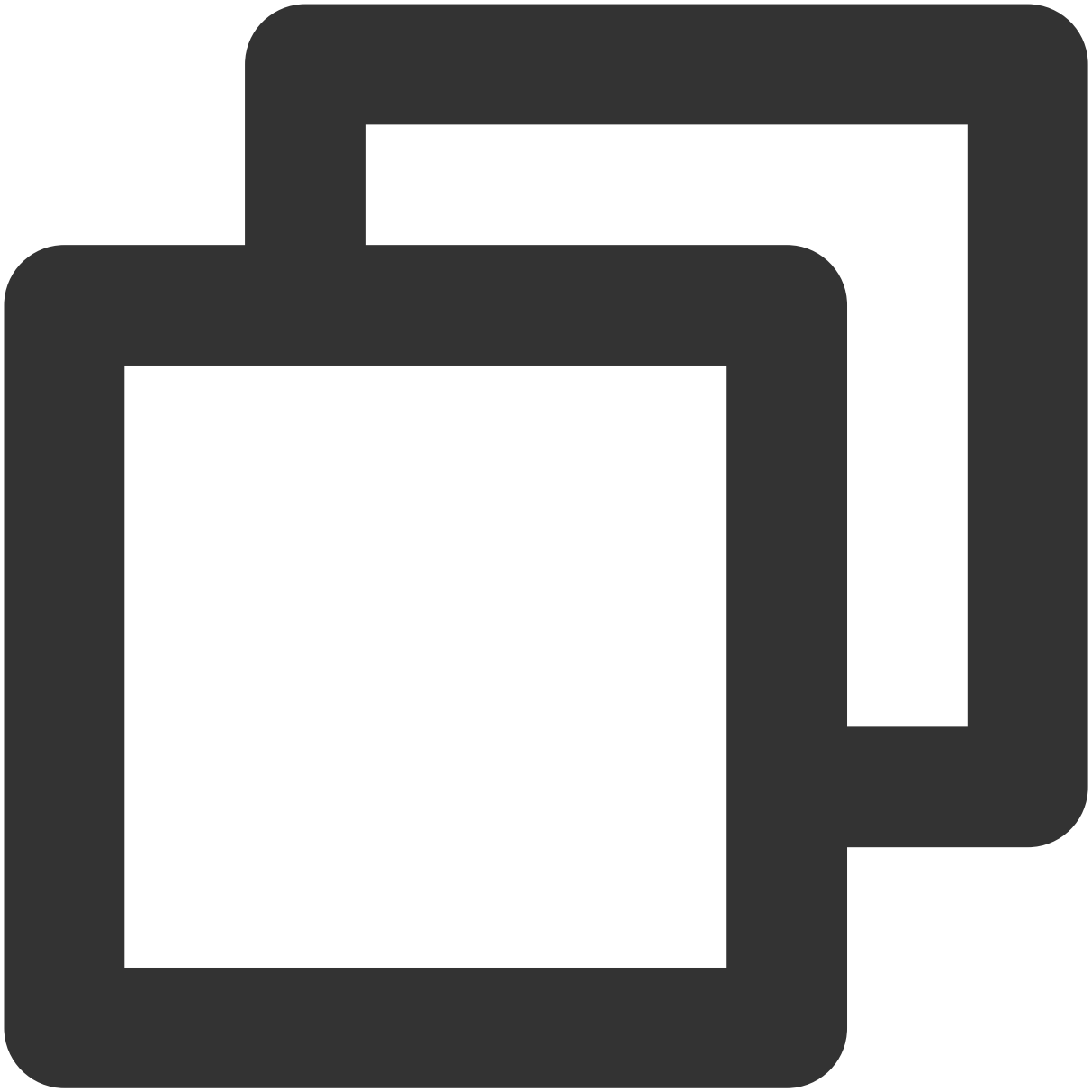
参数如下表所示：

参数	类型	含义
userID	String	离开当前通话的用户 ID

REJECT

通话被拒绝时抛出该事件

1. 在 1v1 通话中，只有主叫方会收到拒绝事件，userID 就是被叫用户名。
2. 在群组通话中，当有被邀请者拒绝通话时，群组通话的其他人都会抛出该事件，userID 就是拒绝通话的用户名。



```
let handleInviteeReject = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.REJECT, handleInviteeReject);
```

参数如下表所示：

参数	类型	含义
userID	String	拒绝通话的用户 ID

NO_RESP

被呼叫者没有回应时，其他通话用户都会抛出该事件。

在 1v1 通话中，只有主叫方会抛出无人应答的事件。例如 A 邀请 B，B 不应答，A 抛出该事件。

在群组通话中，当有被邀请者不回应时，群组通话的其他人都会抛出该事件。例如 A 邀请 B、C 进入通话，B 不应答，A、C 均会抛出该事件。

```
let handleNoResponse = function(event) {
console.log(event.sponsor, event.userIDList);
};
tuiCallEngine.on(TUICallEvent.NO_RESP, handleNoResponse);
```

参数如下表所示：

参数	类型	含义
sponsor	String	发起方的用户 ID
userIDList	Array<String>	无响应的用户 ID 列表

LINE_BUSY

通话忙线事件。例如：B 正在通话时，此时 A 呼叫 B，A 会抛出事件。

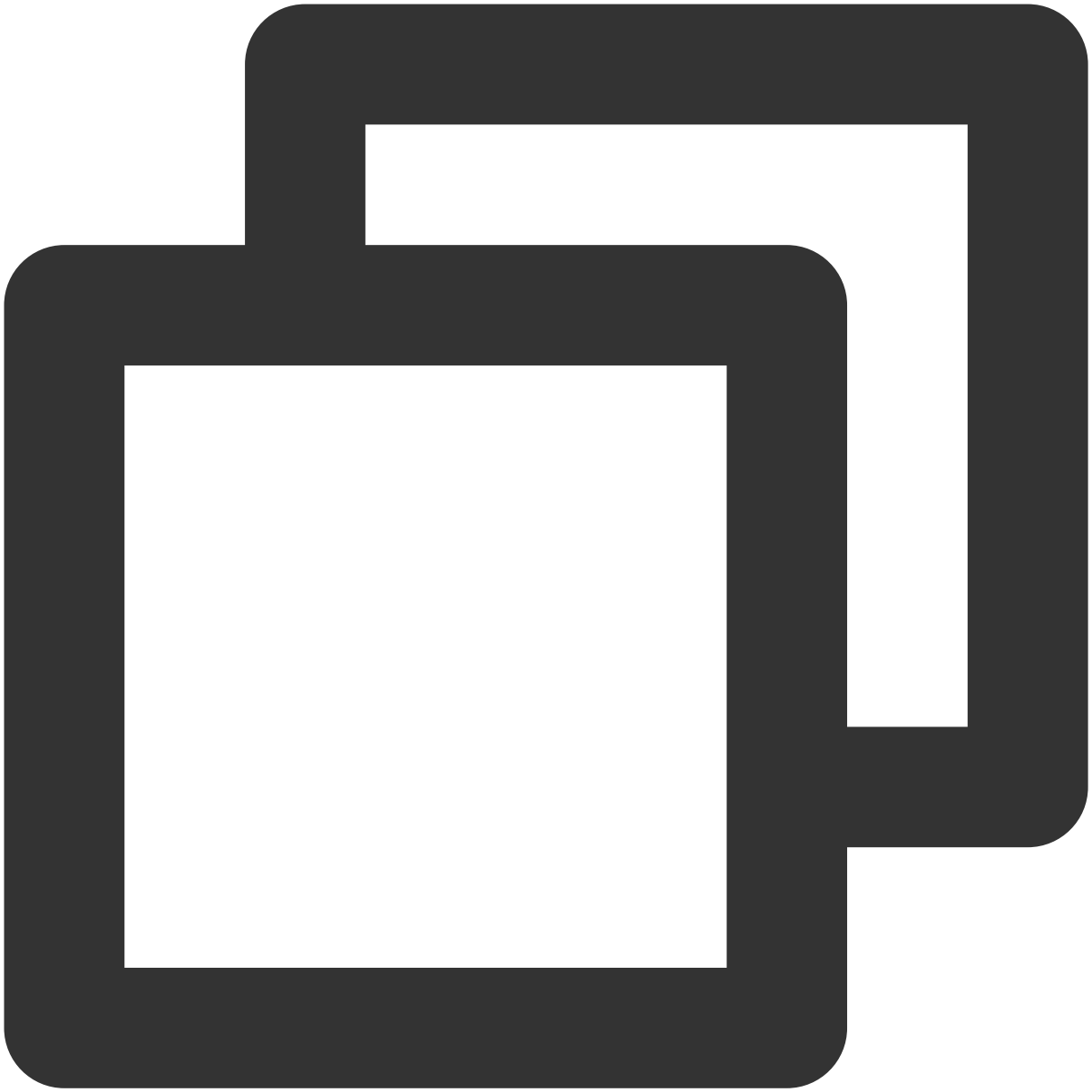
```
let handleLineBusy = function(event) {
  console.log(event);
};
tuiCallEngine.on(TUICallEvent.LINE_BUSY, handleLineBusy);
```

参数如下表所示：

参数	类型	含义
userID	String	忙线用户 ID

USER_VIDEO_AVAILABLE

视频通话中一个用户开启/关闭了摄像头, 通话的其他用户都会抛出该事件。例如：A 和 B 在进行视频通话，A 开/关摄像头，B 会抛出该事件。



```
let handleUserVideoChange = function(event) {
  console.log(event.userID, event.isVideoAvailable);
};
tuiCallEngine.on(TUICallEvent.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

参数如下表所示：

参数	类型	含义
userID	String	开/关摄像头的用户 ID

isVideoAvailable	Boolean	true 用户打开摄像头；false 用户关闭摄像头

USER_AUDIO_AVAILABLE

音视频通话中一个用户开启/关闭了麦克风, 通话的其他用户都会抛出该事件。例如：A 和 B 在进行音视频通话，A 开/关麦克风，B 会抛出该事件。

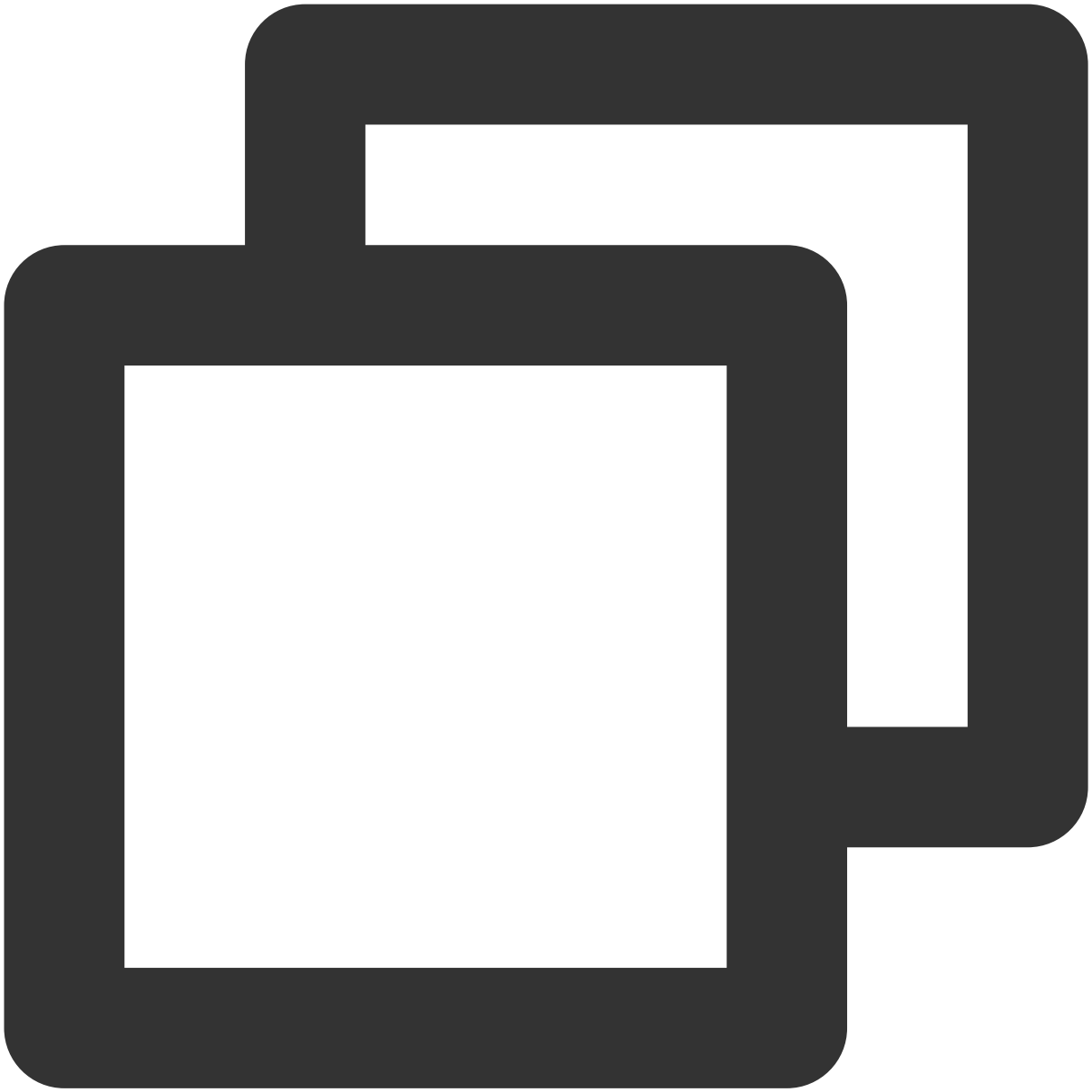
```
let handleUserAudioChange = function(event) {
  console.log(event.userID, event.isAudioAvailable);
};
tuiCallEngine.on(TUICallEvent.USER_AUDIO_AVAILABLE, handleUserAudioChange);
```

参数如下表所示：

参数	类型	含义
userID	String	开/关麦克风的用户 ID
isAudioAvailable	Boolean	true 用户打开麦克风；false 用户关闭麦克风

USER_VOICE_VOLUME

音视频通话中用户音量变化时，通话的其他用户都会抛出该事件。例如：A 和 B 在进行音视频通话，A 音量变化，B 会抛出该事件。



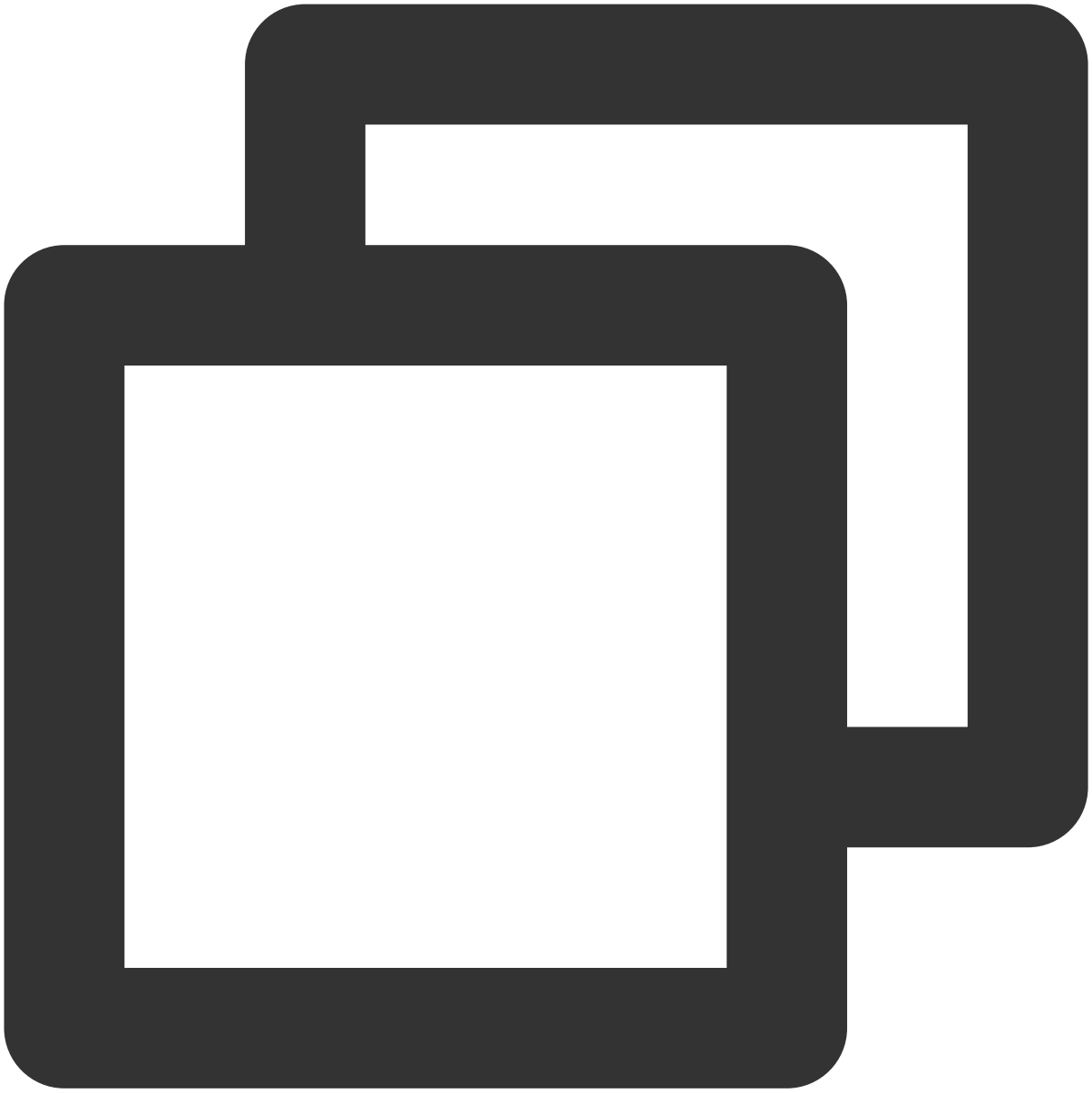
```
let handleUserVoiceVolumeChange = function(event) {
  console.log(event.volumeMap);
};
tuiCallEngine.on(TUICallEvent.USER_VOICE_VOLUME, handleUserVoiceVolumeChange);
```

参数如下表所示：

参数	类型	含义
volumeMap	Array<Object>	音量表，根据每个 userid 可以获取对应的音量大小，音量范围：[0，100]

GROUP_CALL_INVITEE_LIST_UPDATE

群聊更新邀请列表收到该事件。



```
let handleGroupInviteeListUpdate = function(event) {
  console.log(event.userIDList);
};
tuiCallEngine.on(TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeLis
```

参数如下表所示：

参数	类型	含义

userIDList	Array<String>	群组更新邀请列表

INVITED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

注意：

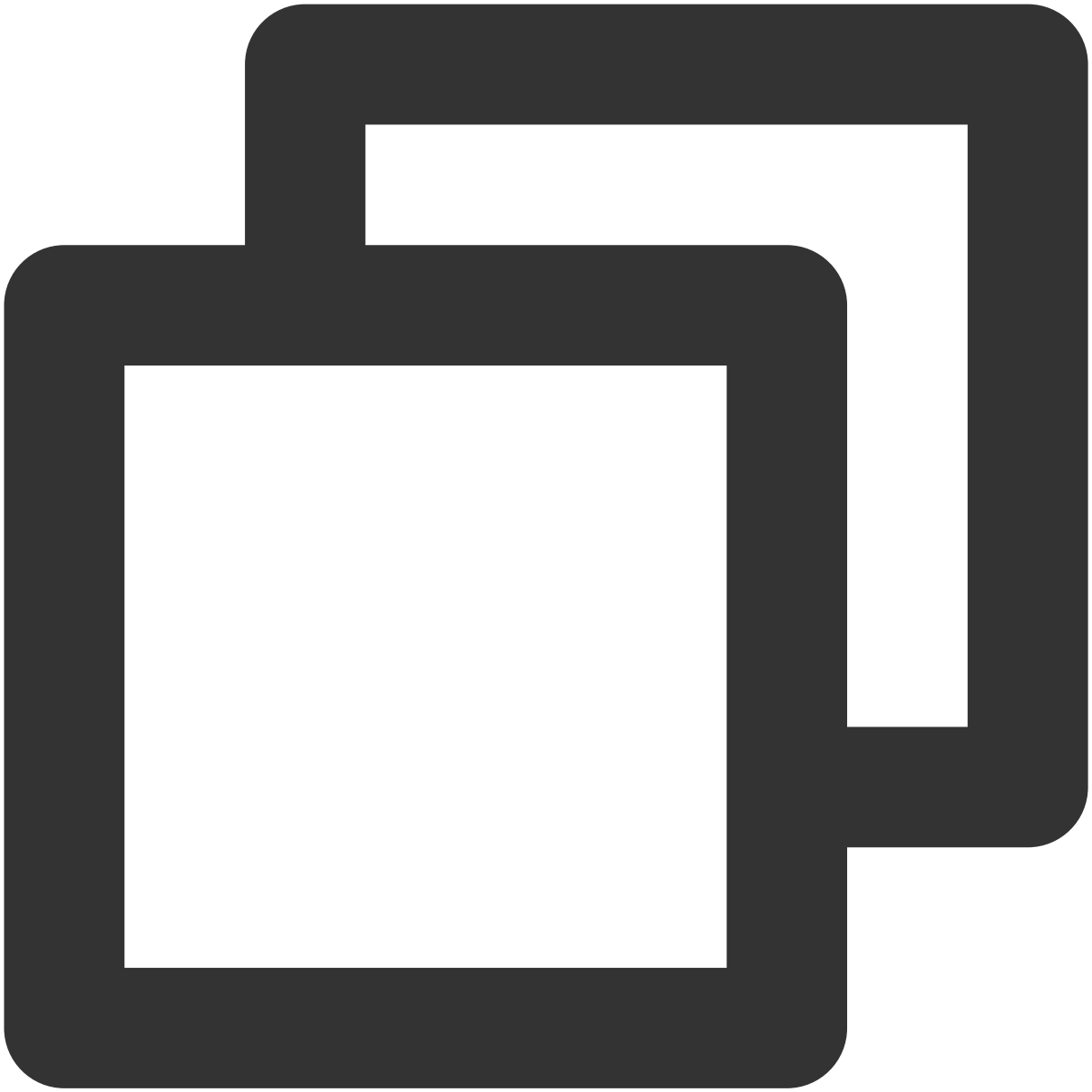
计划后续版本废弃，建议使用：ON_CALL_RECEIVED。

ON_CALL_RECEIVED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

注意：

v1.4.6+ 版本支持。



```
let handleOnCallReceived = function(event) {
    console.log(event);
};
tuiCallEngine.on(TUICallEvent.ON_CALL_RECEIVED, handleOnCallReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者

userIDList	Array<String>	同时还被邀请的人
isFromGroup	Boolean	是否是群组通话
inviteData	Object	呼叫数据
inviteID	String	邀请 ID，标识一次邀请
userData	String	扩展字段：用于在邀请信令中增加扩展信息
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫、被叫

CALLING_CANCEL

如果通话未建立，都会抛出该事件。您可以通过监听该事件来实现类似未接来电、重置 UI 状态等显示逻辑。通话未建立的场景如下：

注意：

计划后续版本废弃，建议使用：ON_CALL_CANCELED。

ON_CALL_CANCELED

如果通话未建立，都会抛出该事件。您可以通过监听该事件来实现类似未接来电、重置 UI 状态等显示逻辑。通话未建立的场景如下：

主叫取消：主叫抛出该事件，userID 为主叫；被叫也会抛出该事件，userID 为被叫；

被叫超时：主叫会同时抛出 NO_RESP 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

被叫拒接：主叫会同时抛出 REJECT 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

被叫忙线：主叫会同时抛出 LINE_BUSY 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

注意：

v1.4.6+ 版本支持 。

```
let handleOnCallCanceled = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.ON_CALL_CANCELED, handleOnCallCanceled);
```

参数如下表所示：

参数	类型	含义
userID	String	通话取消的用户 ID

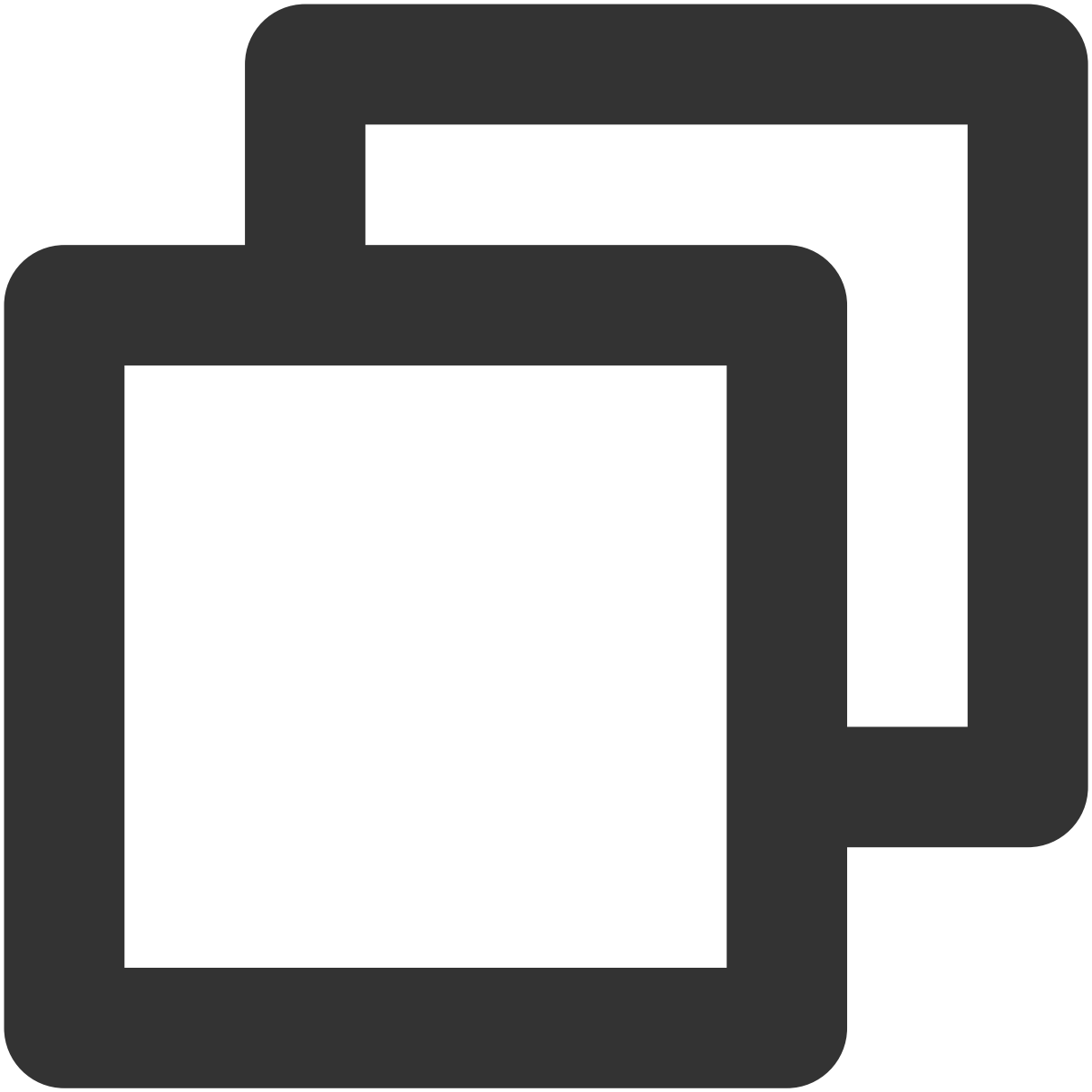
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫、被叫

ON_CALL_BEGIN

表示通话接通，主叫和被叫都会抛出该事件，您可以通过监听这个事件来开启云端录制、内容审核等流程。

注意：

v1.4.6+ 版本支持。



```
let handleOnCallBegin = function(event) {
    console.log(event);
};
tuiCallEngine.on(TUICallEvent.ON_CALL_BEGIN, handleOnCallBegin);
```

参数如下表所示：

参数	类型	含义
callId	String	本次通话的唯一 ID

roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，类型：主叫、被叫

CALLING_END

通话结束时，主叫和被叫都会抛出该事件。您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

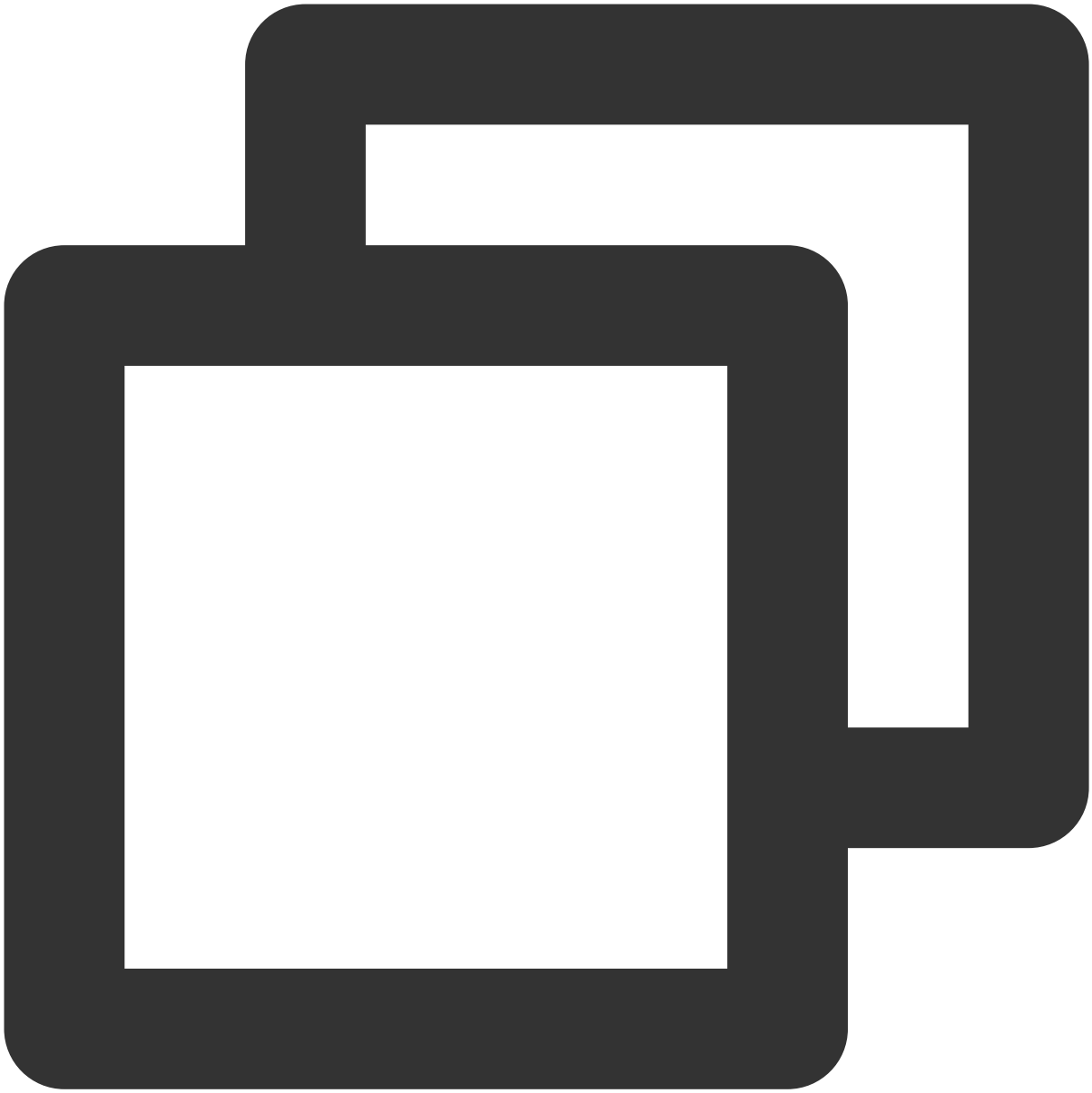
```
let handleCallingEnd = function(event) {
  console.log(event.userID, event.);
};
tuiCallEngine.on(TUICallEvent.CALLING_END, handleCallingEnd);
```

参数如下表所示：

参数	类型	含义
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫（'inviter'）、被叫（'invitee'）、未知（''）
totalTime	Number	此次通话的时长，单位: 秒
userID	String	通话结束的 userID。
callId	String	本次通话的唯一 ID。**v1.4.6+ 版本支持**
callEnd	Number	此次通话的时长（**后续将废弃，请使用 totalTime**），单位: 秒

DEVICED_UPDATED

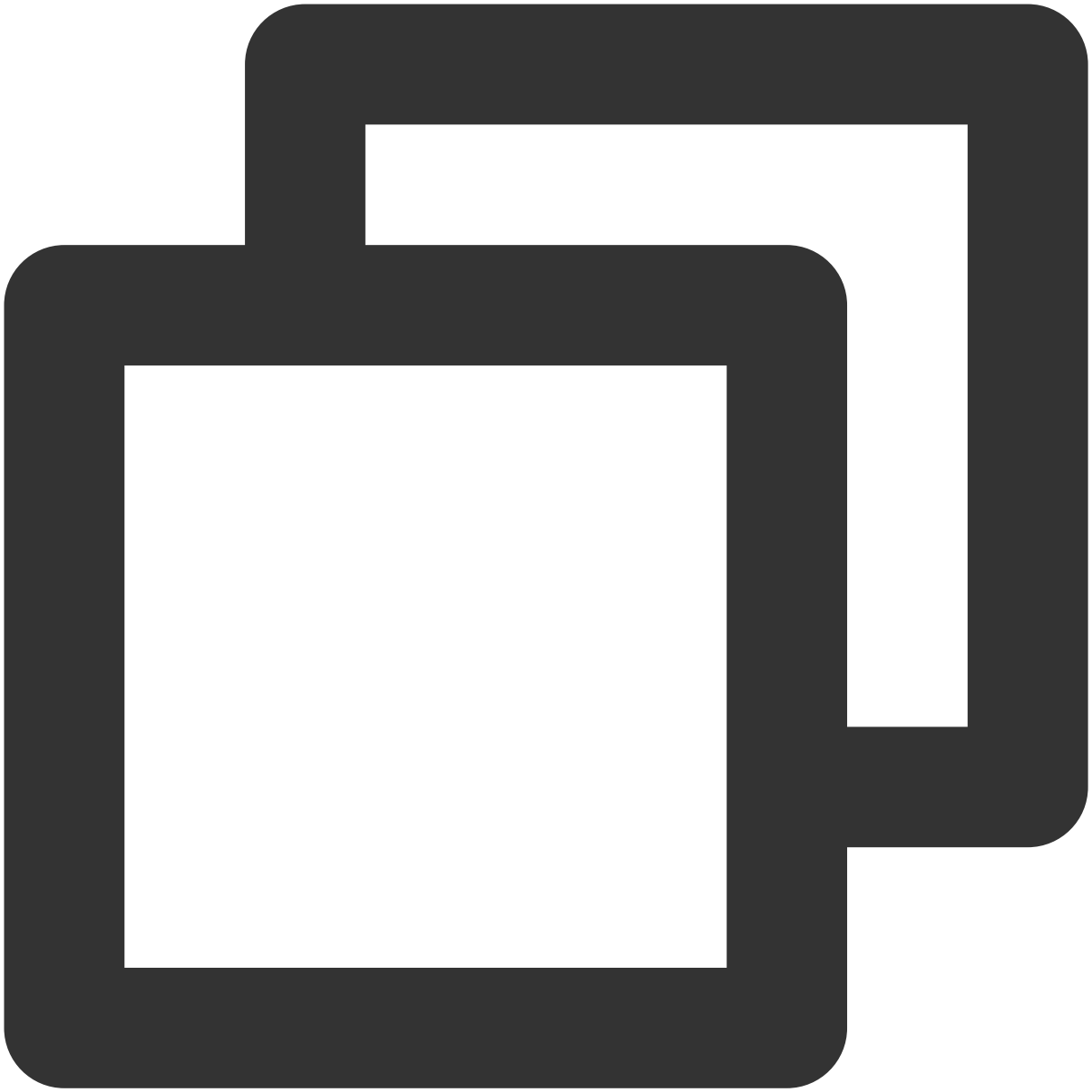
设备列表更新时抛出该事件。



```
let handleDeviceUpdated = function({ microphoneList, cameraList, currentMicrophoneI
  console.log(microphoneList, cameraList, currentMicrophoneID, currentCameraID);
};
tuiCallEngine.on(TUICallEvent.DEVICED_UPDATED, handleDeviceUpdated);
```

CALL_TYPE_CHANGED

通话类型切换时抛出该事件。



```
let handleCallTypeChanged = function({ oldCallType, newCallType }) {
  console.log(oldCallType, newCallType);
};
tuiCallEngine.on(TUICallEvent.CALL_TYPE_CHANGED, handleDeviceUpdated);
```

参数如下表所示：

参数	类型	含义
oldCallType	Number	旧通话类型

newCallType	Number	新通话类型

ON_USER_NETWORK_QUALITY_CHANGED

所有用户网络质量的事件

注意：

v3.0.7+ 支持。

```
let handleOnUserNetworkQualityChange = function(event) {
  console.log(event.networkQualityList);
};
```

```
tuiCallEngine.on(TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED, handleOnUserNetworkQ
```

参数如下表所示：

参数	类型	含义
networkQualityList	Array<Object>	网络状态，根据每个 userID 可以获取对应用户当前的网络质量

Flutter
API 概览

最近更新时间：2024-03-20 17:09:03

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
setCallingBell	自定义来电铃声

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
init	初始化SDK函数，完成包含通话服务鉴权在内初始化动作
unInit	反初始化函数，释放TUICallEngine使用到的相关资源。
addObserver	增加事件回调

removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式 （尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
queryRecentCalls	查询通话记录
deleteRecordCalls	删除通话记录
setBeautyLevel	设置美颜级别

版权所有：腾讯云计算（北京）有限责任公司

TUICallObserver

TUICallObserver 是回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2024-04-15 17:41:16

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入（TUICallKit）。

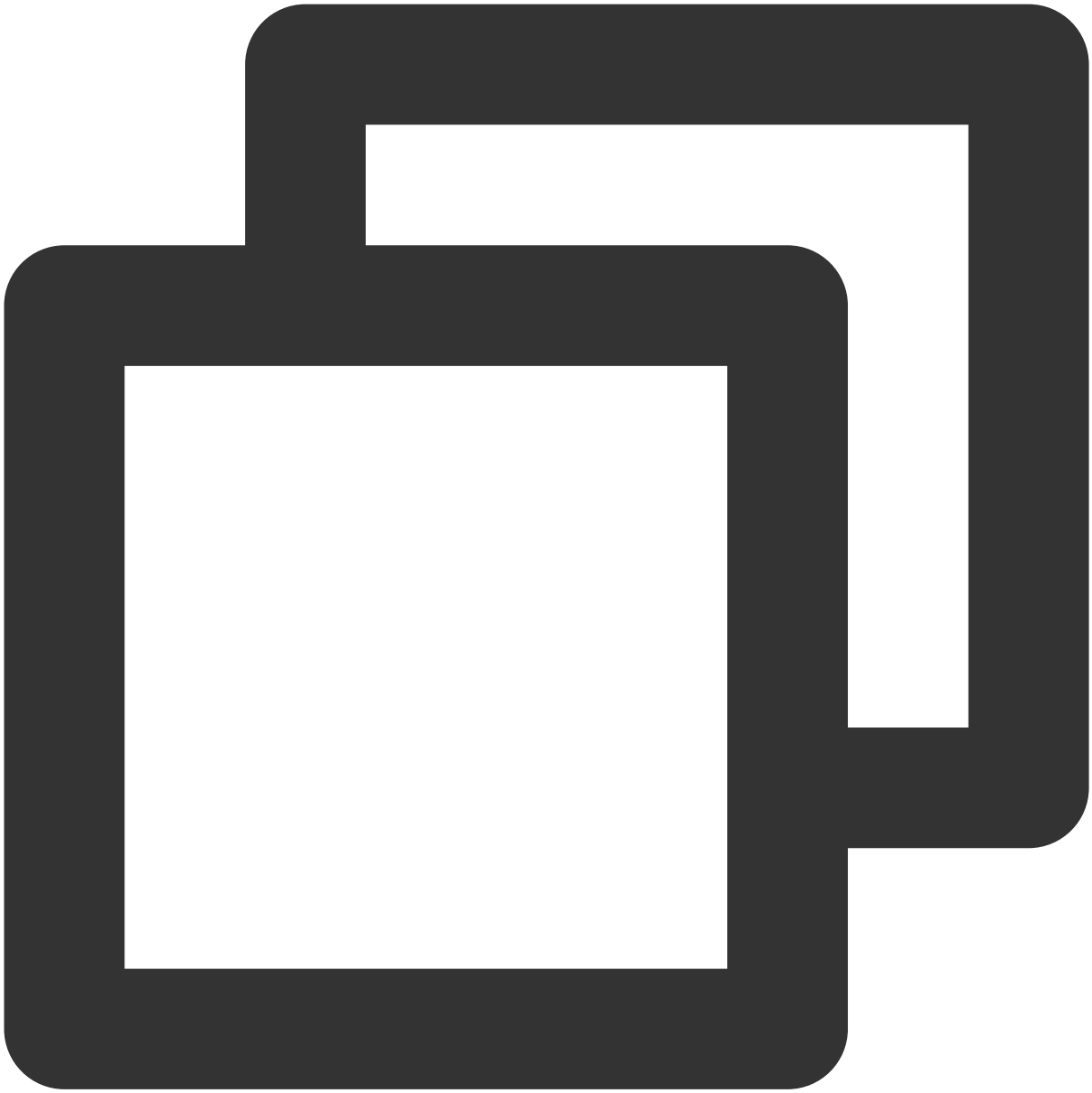
API 概览

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
setCallingBell	自定义来电铃声

API 详情

login

登录



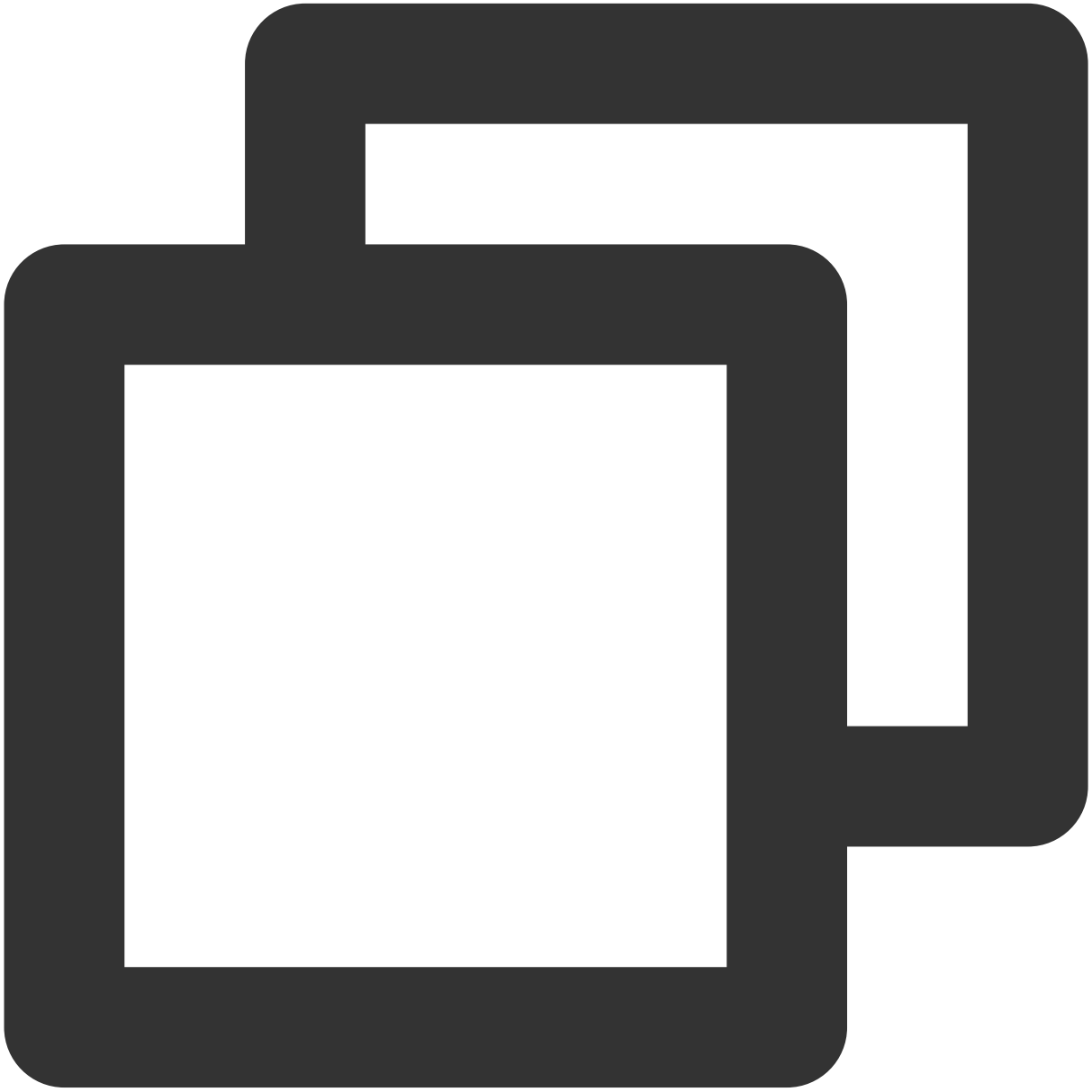
```
Future<TUIResult> login(int sdkAppId, String userId, String userSig);
```

参数	类型	含义
sdkAppId	int	用户 SDKAppID
userId	String	用户 ID
userSig	String	用户签名 userSig（计算方法详见 生成 UserSig）
返回值	TUIResult	包含 code 和 message 信息：

		code 为空 ("") 表示调用成功。 code 不为空 ("") 表示调用失败，失败原因见 message。

logout

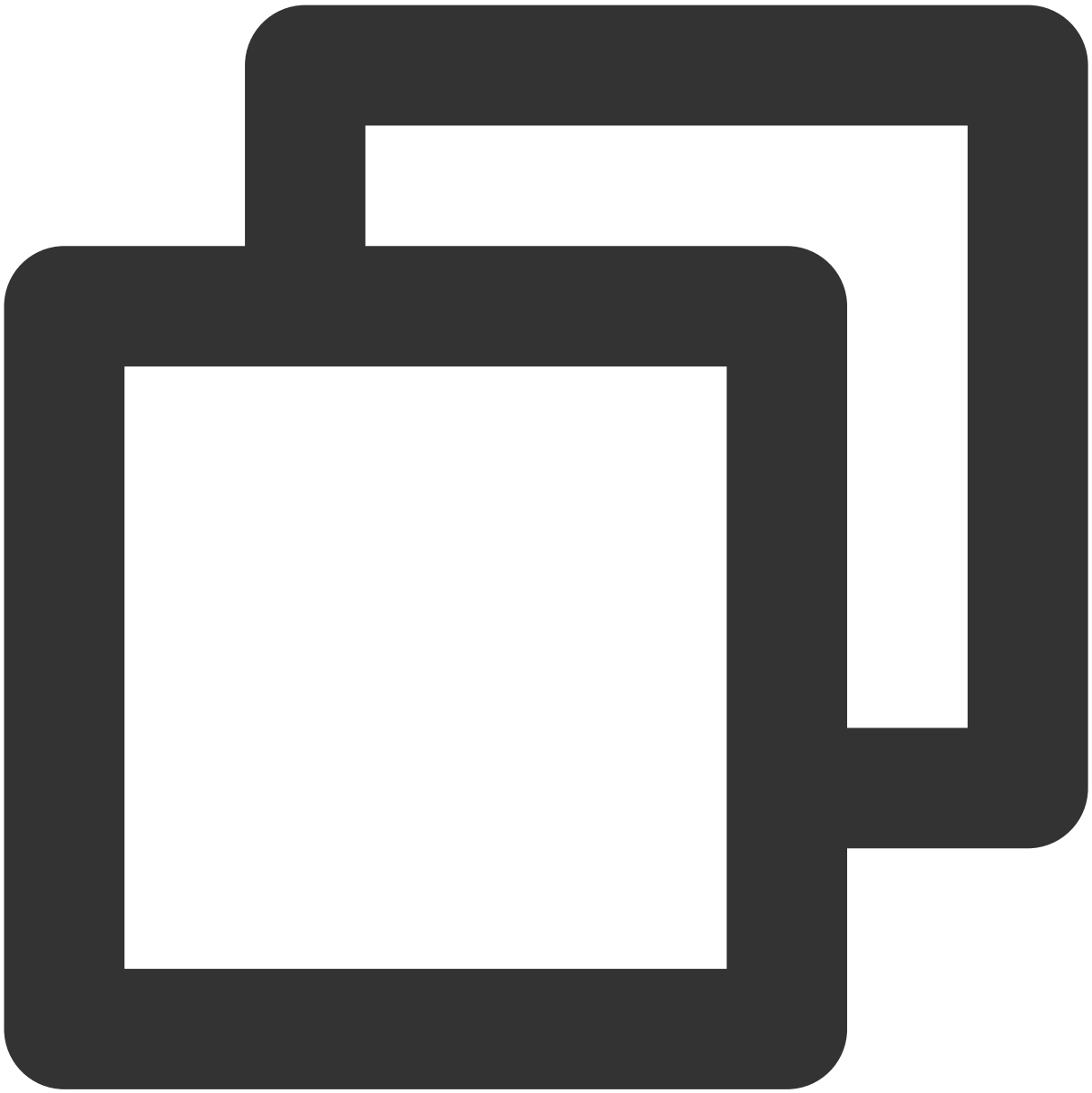
登出



```
Future<void> logout()
```

setSelfInfo

设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是URL格式。



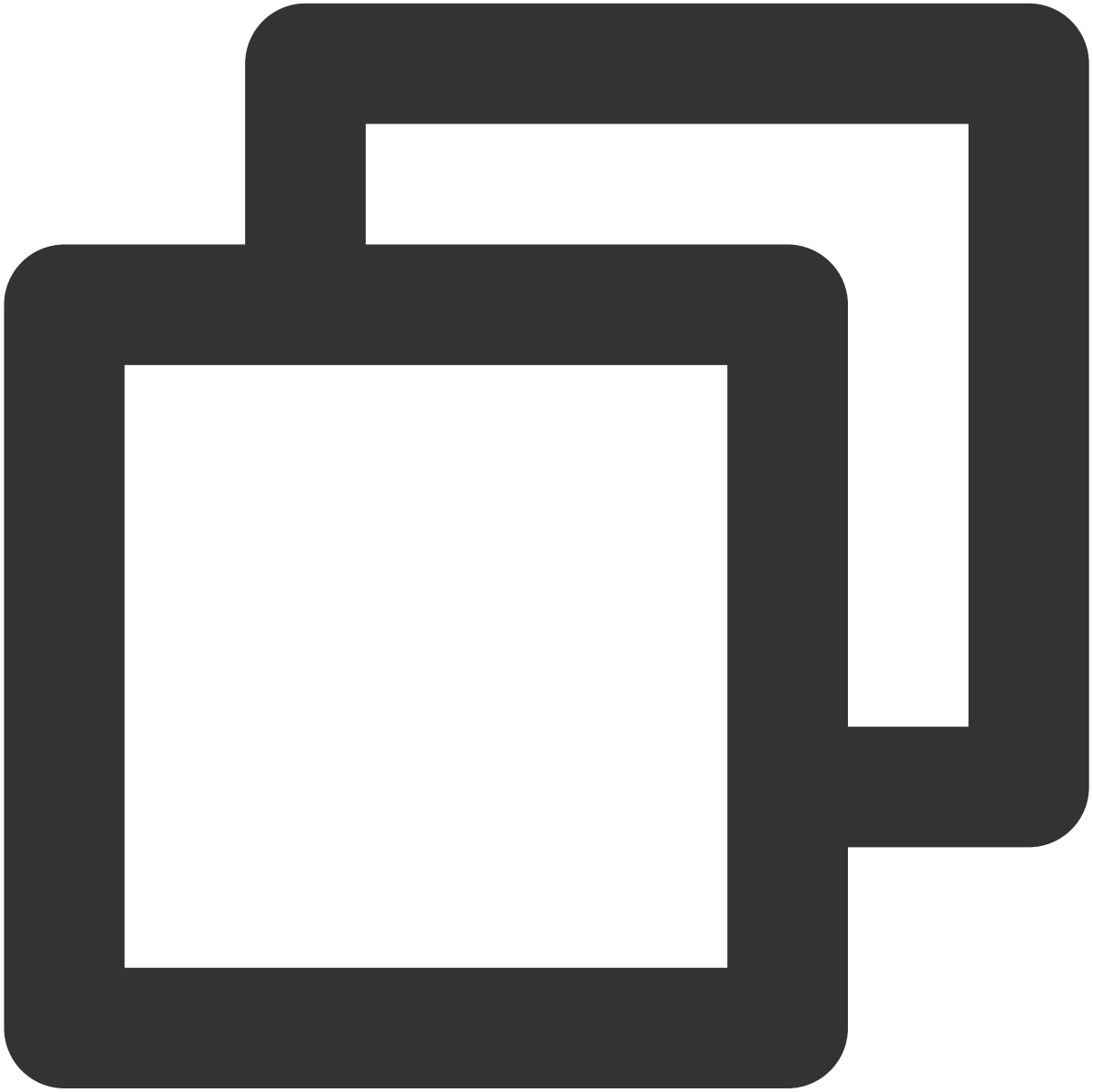
```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

参数	类型	含义
nickName	String	目标用户的昵称，非必填
avatar	String	目标用户的头像，非必填
返回值	TUIResult	包含 code 和 message 信息：

		code 为空 ("") 表示调用成功。 code 不为空 ("") 表示调用失败，失败原因见 message。

call

拨打电话（1v1通话）



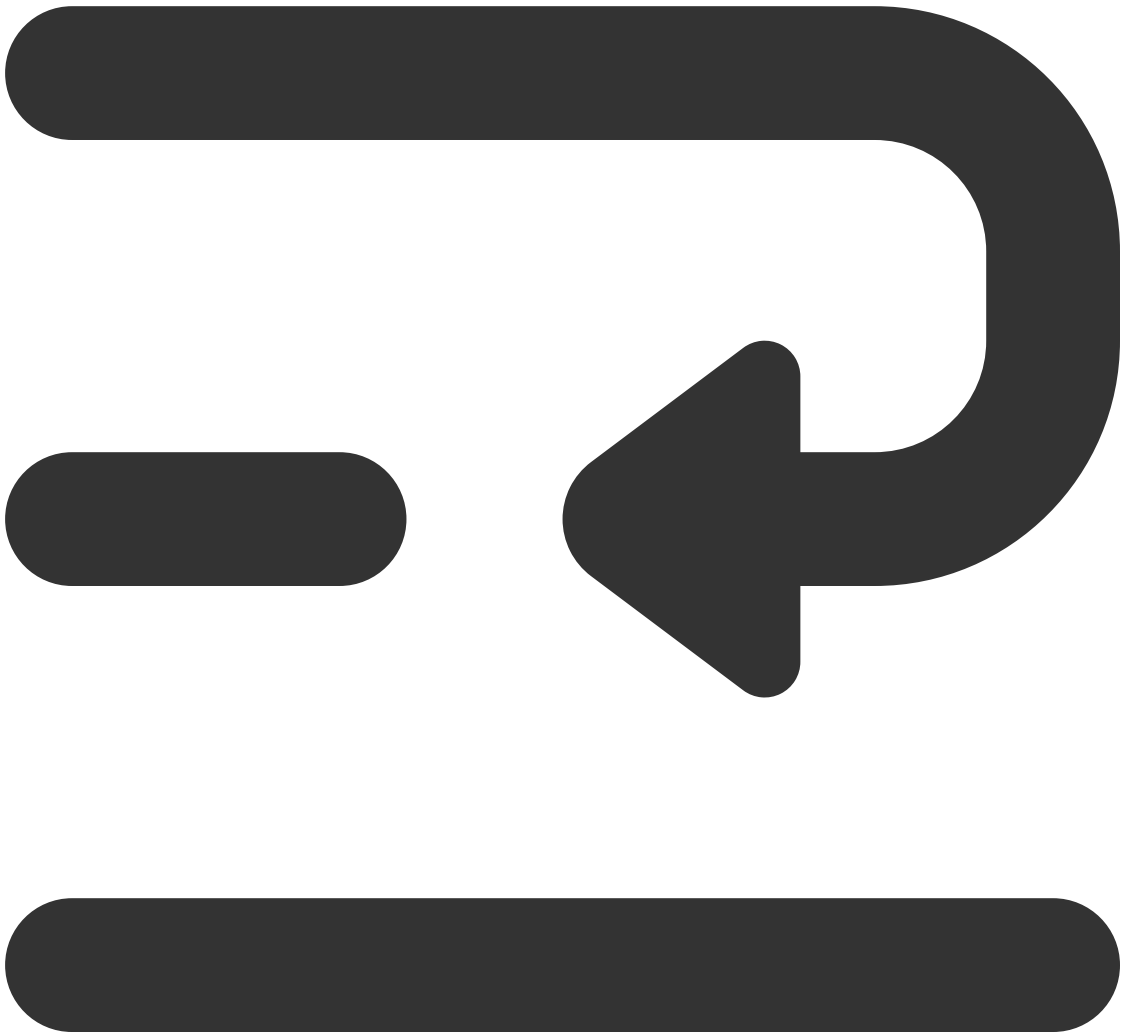
```
Future<void> call(String userId, TUICallMediaType callMediaType, [TUICallParams? pa
```

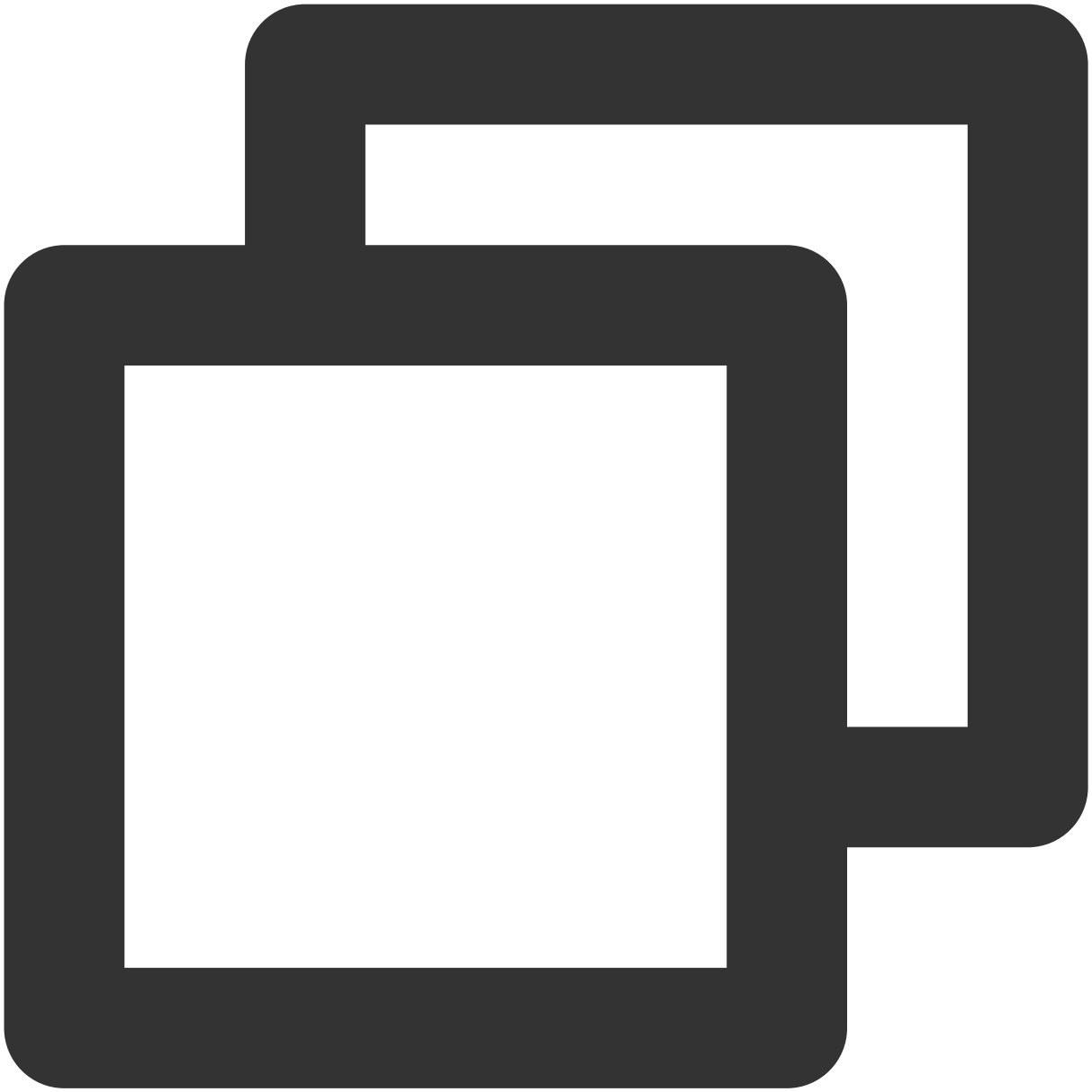
参数	类型	含义

userId	String	目标用户的 userID
callMediaType	TUICallMediaType	通话的媒体类型，例如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`
params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话，注意：使用群组通话前需要创建IM 群组，如果已经创建，请忽略。





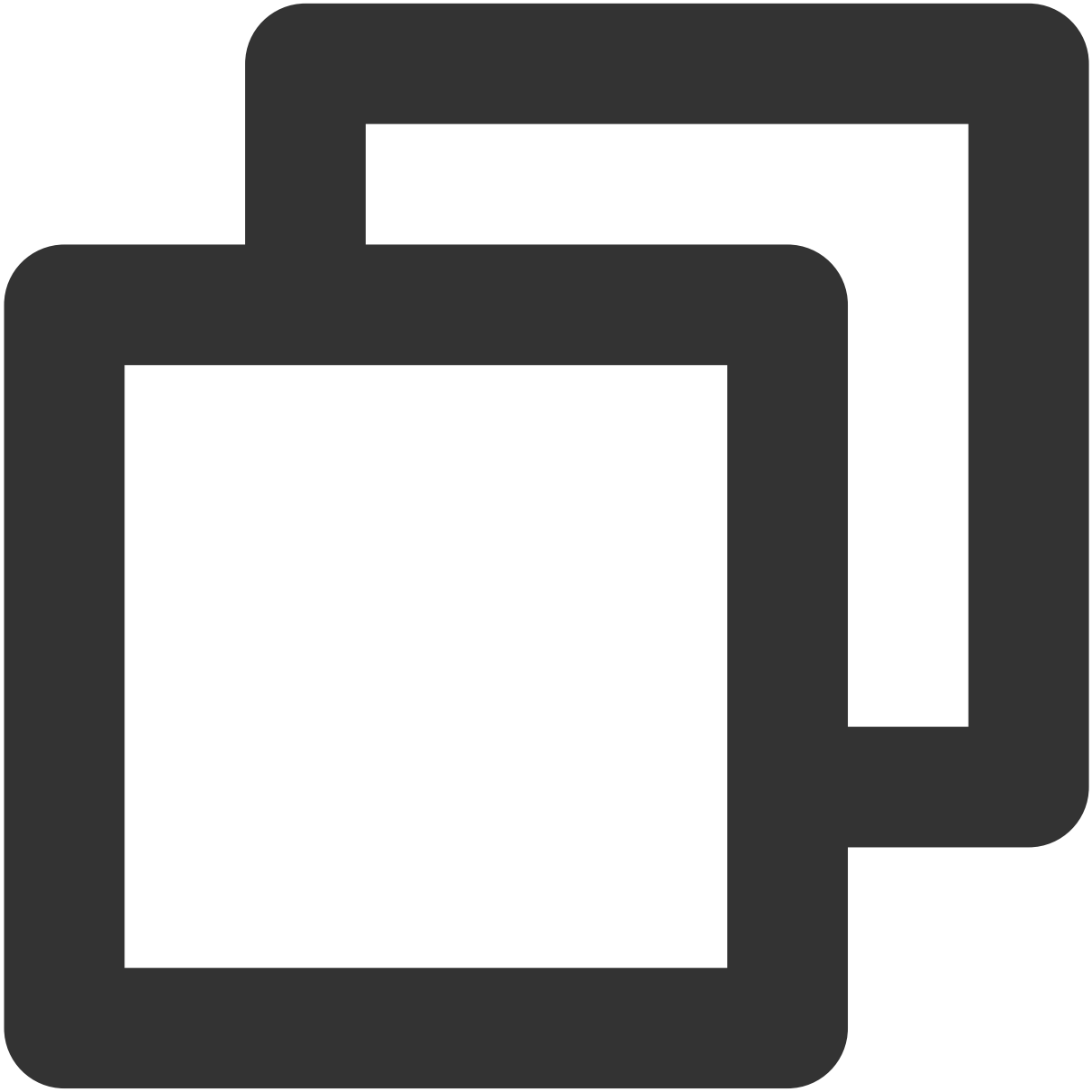
```
Future<void> groupCall(String groupId, List<String> userIdList, TUICallMediaType ca
```

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List<String>	目标用户的userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，例如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

joinInGroupCall

加入群组中已有的音视频通话。



```
Future<void> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaType cal
```

参数	类型	含义

roomId	TUIRoomID	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaType	TUICallMediaType	通话的媒体类型，例如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

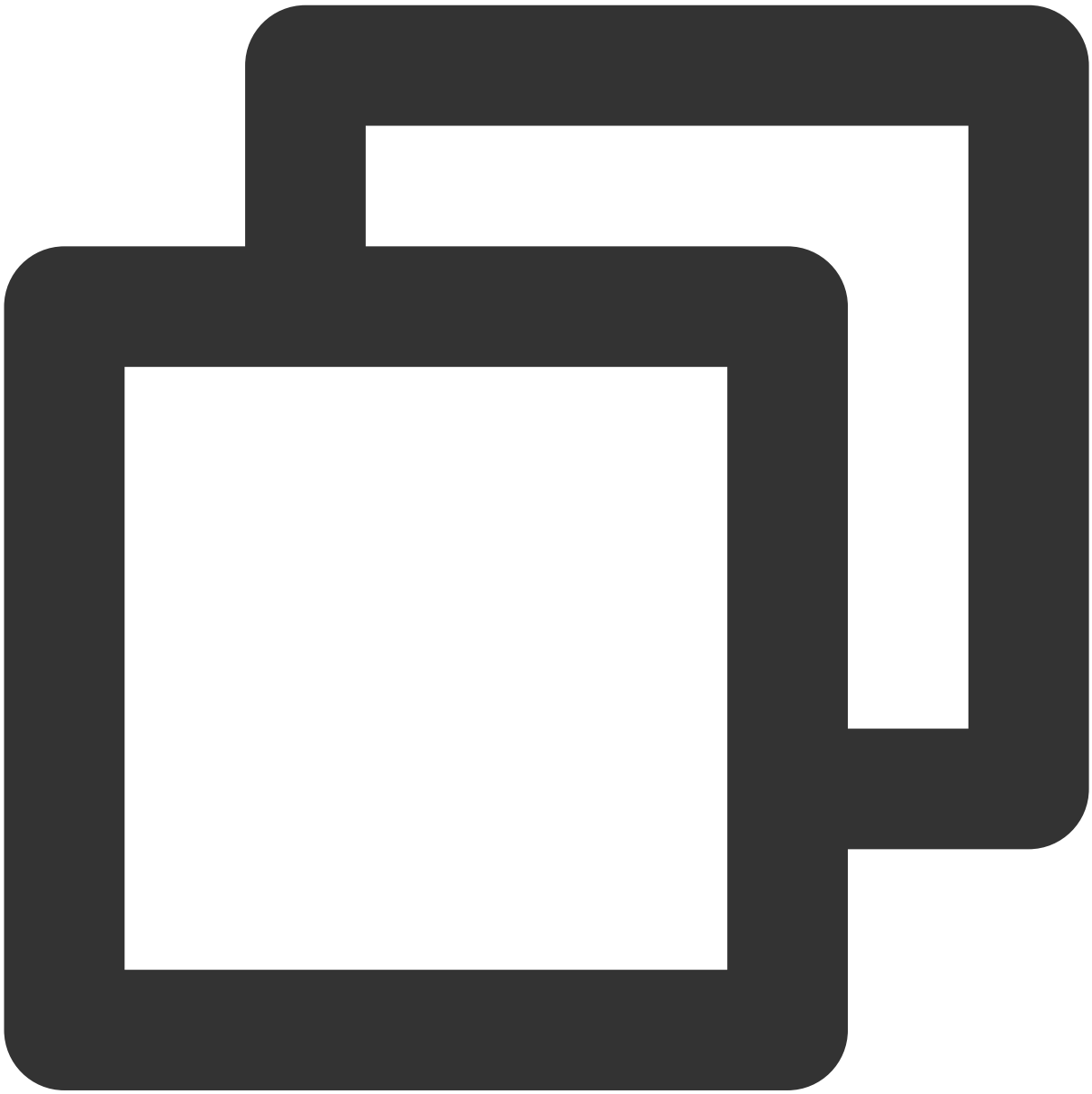
enableMuteMode

开启/关闭静音模式。

```
Future<void> enableMuteMode(bool enable)
```

enableFloatWindow

开启/关闭悬浮窗功能，设置为false后，通话界面左上角的悬浮窗按钮会隐藏。



```
Future<void> enableFloatWindow(bool enable)
```

setCallingBell

自定义来电铃声。

```
Future<void> setCallingBell(String assetName)
```

参数	类型	含义
assetName	String	铃声的路径，需要将铃声文件添加至主工程的 assets 资源中。

TUICallEngine

最近更新时间：2024-01-25 14:47:54

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
init	初始化SDK函数，完成包含通话服务鉴权在内初始化动作
unInit	反初始化函数，释放TUICallEngine使用到的相关资源。
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头

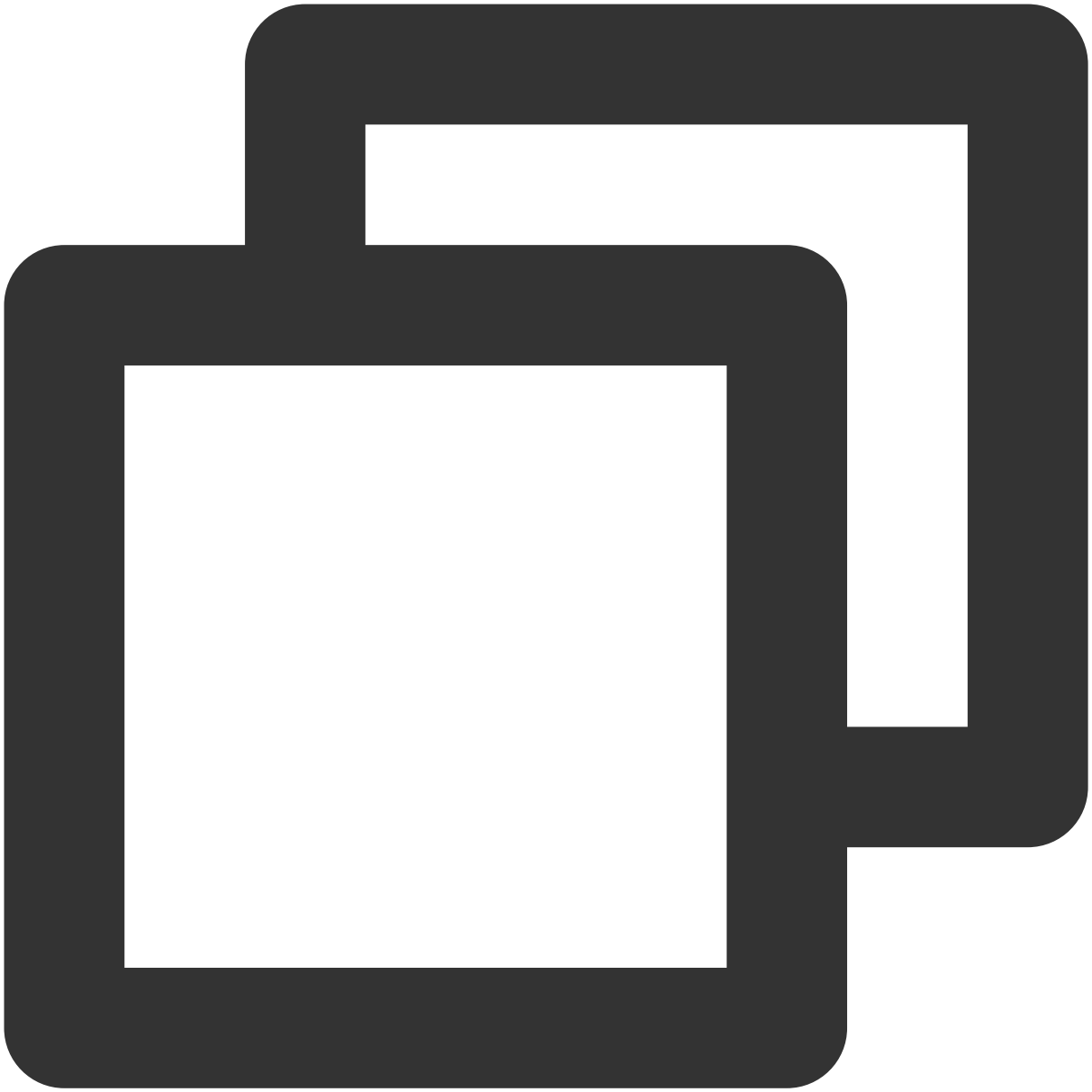
版权所有：腾讯云计算（北京）有限责任公司

closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式 （尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
queryRecentCalls	查询通话记录
deleteRecordCalls	删除通话记录
setBeautyLevel	设置美颜级别

API 详情

init

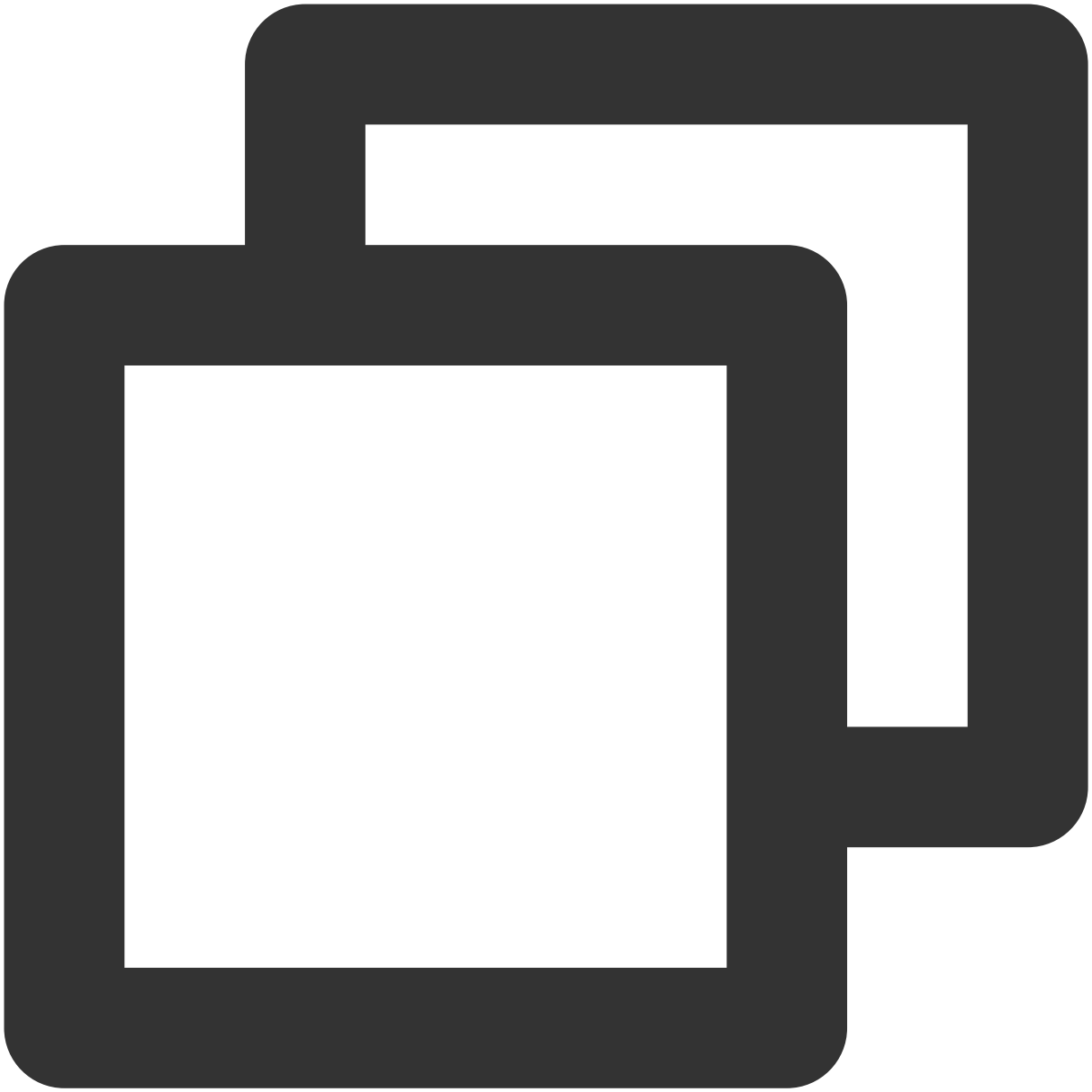
初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。



```
Future<TUIResult> init(int sdkAppID, String userId, String userSig)
```

unInit

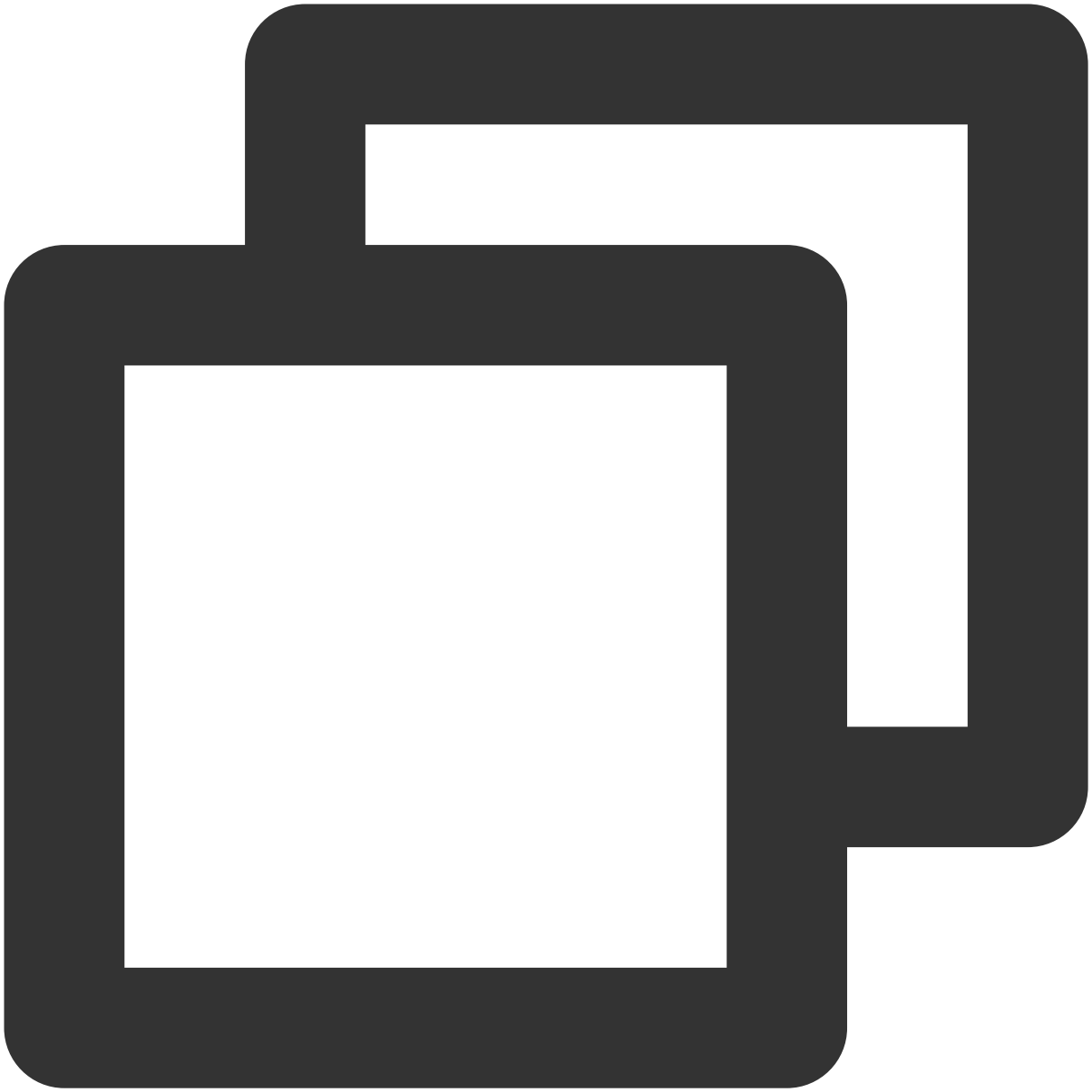
反初始化函数，释放 `TUICallEngine` 使用到的相关资源。



```
Future<TUIResult> unInit()
```

addObserver

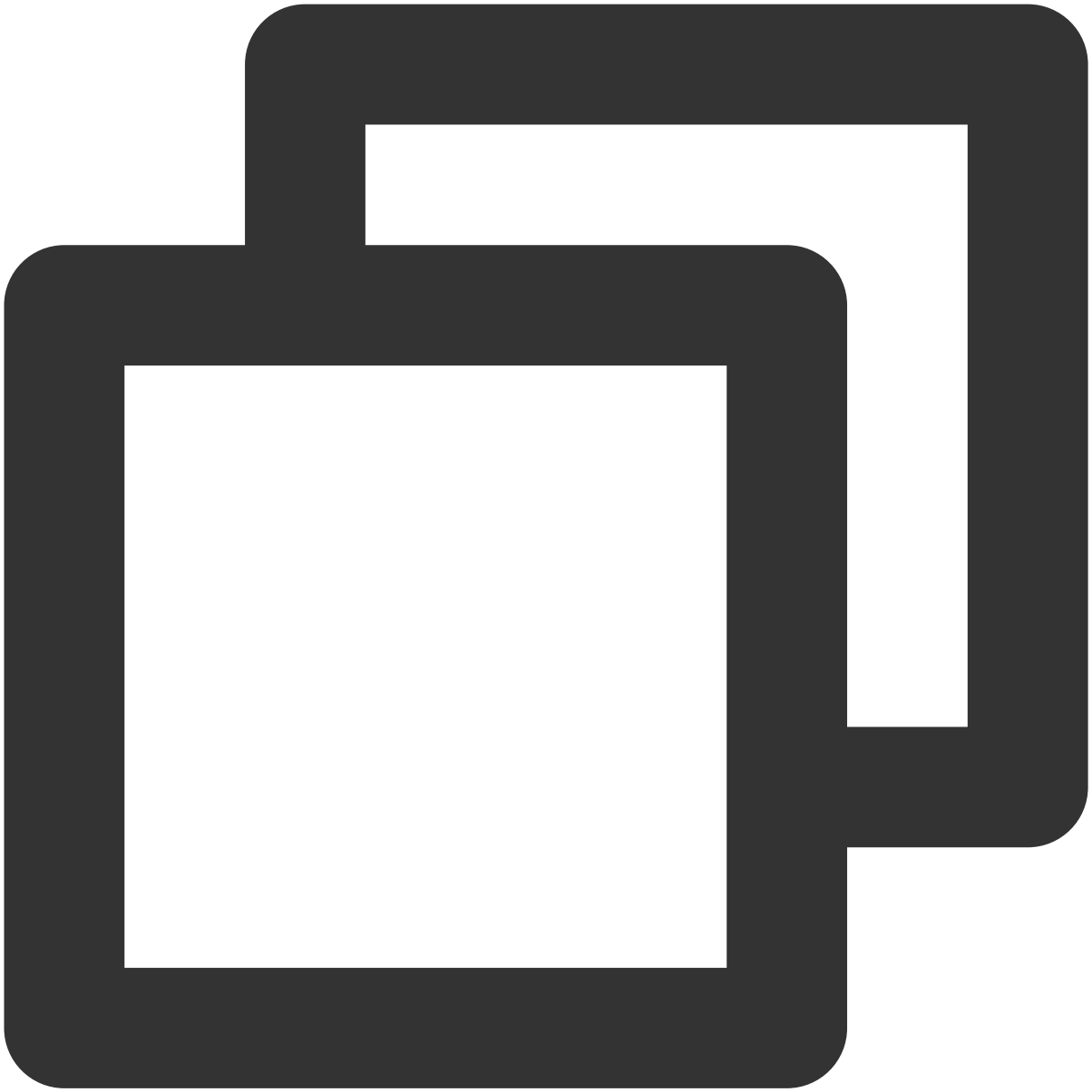
添加回调接口，您可以通过这个接口，监听 `TUICallObserver` 相关的事件回调。



```
Future<void> addObserver(TUICallObserver observer)
```

removeObserver

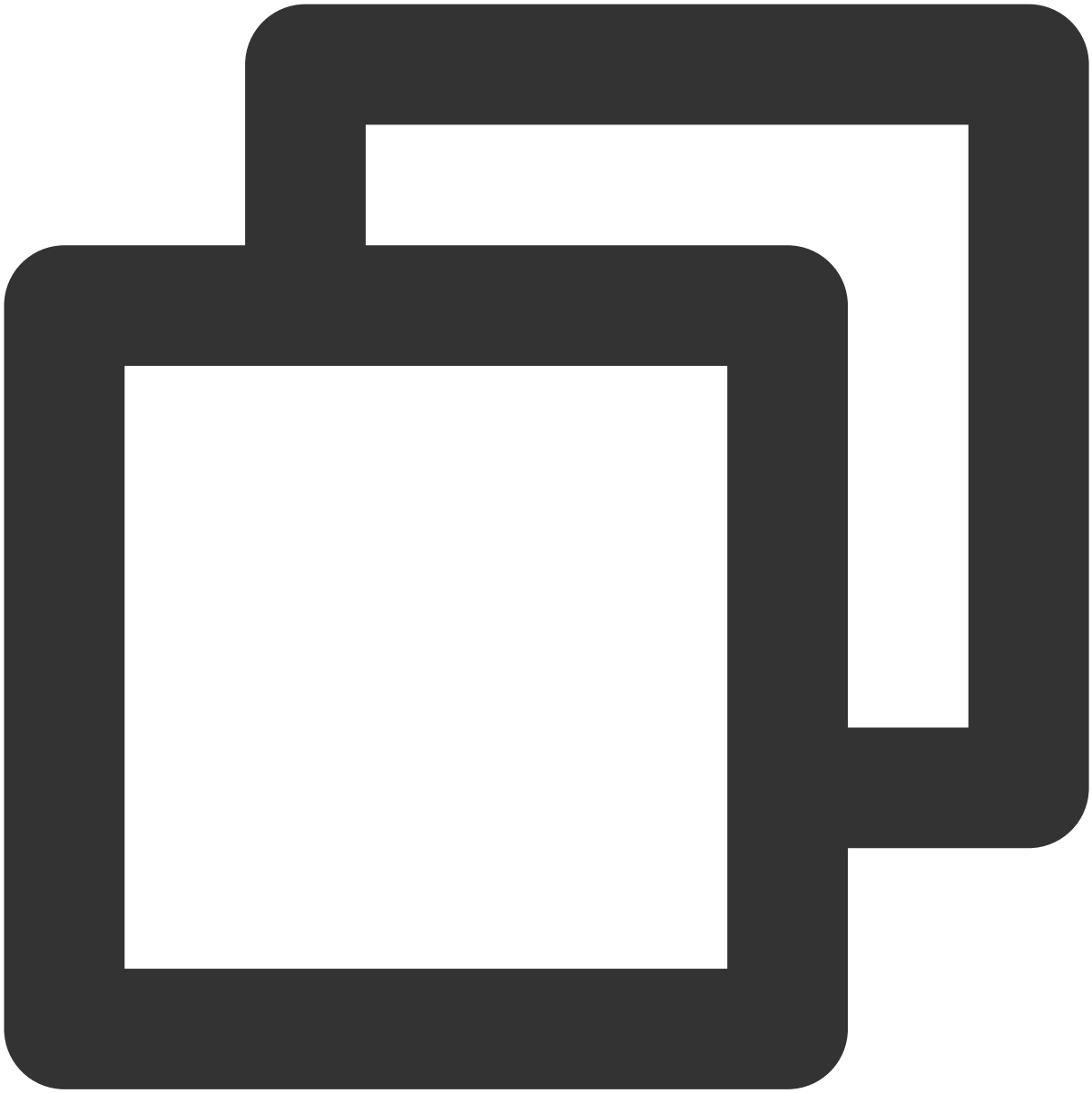
移除回调接口。



```
Future<void> removeObserver(TUICallObserver observer)
```

call

拨打电话（1v1通话）。



```
Future<TUIResult> call(String userId, TUICallMediaType mediaType, TUICallParams par
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定

		义内容等

groupCall

发起群组通话。

注意：

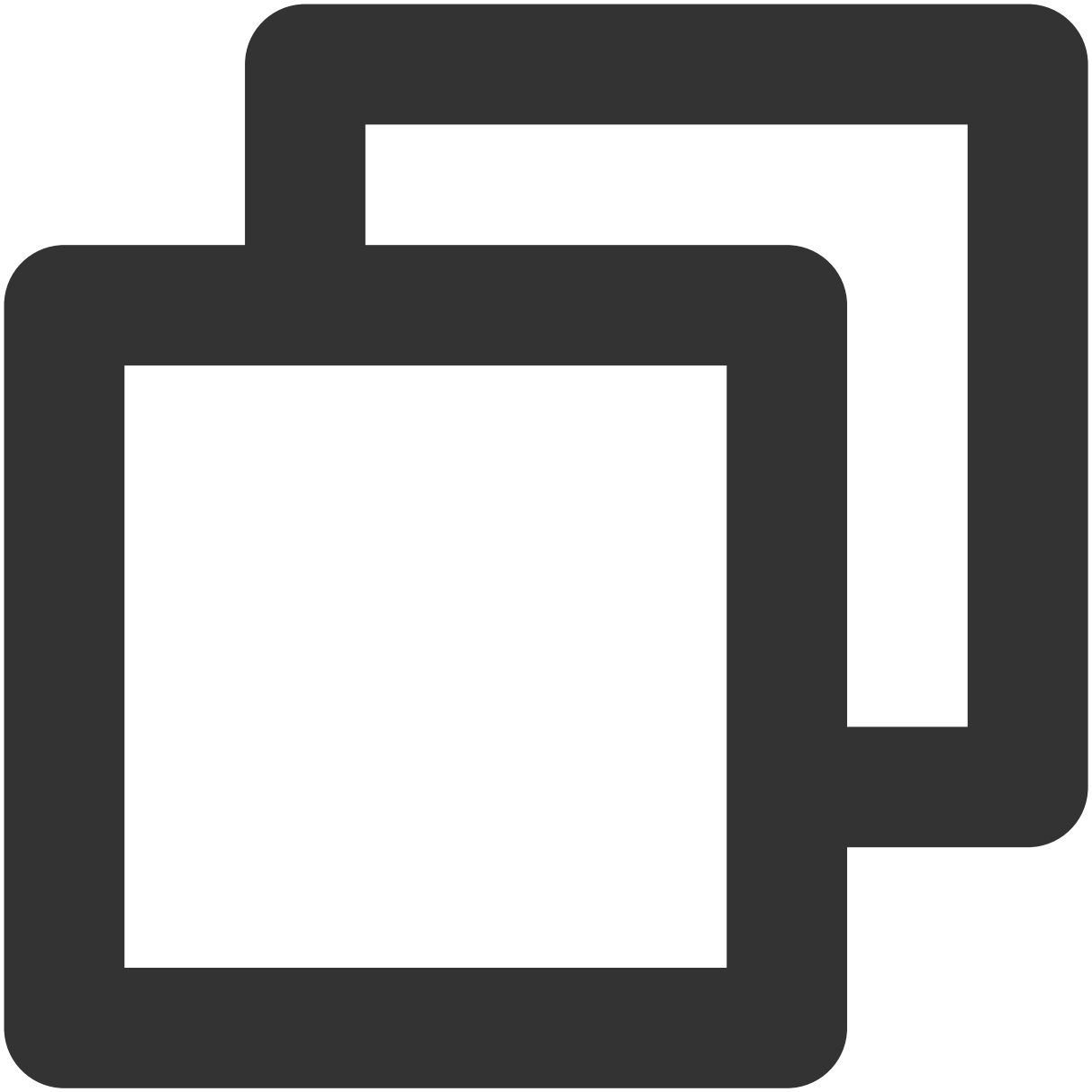
使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。

```
Future<TUIResult> groupCall(String groupId, List<String> userIdList, TUICallMediaTy
```


参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List<String>	目标用户的 userId 列表
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

accept

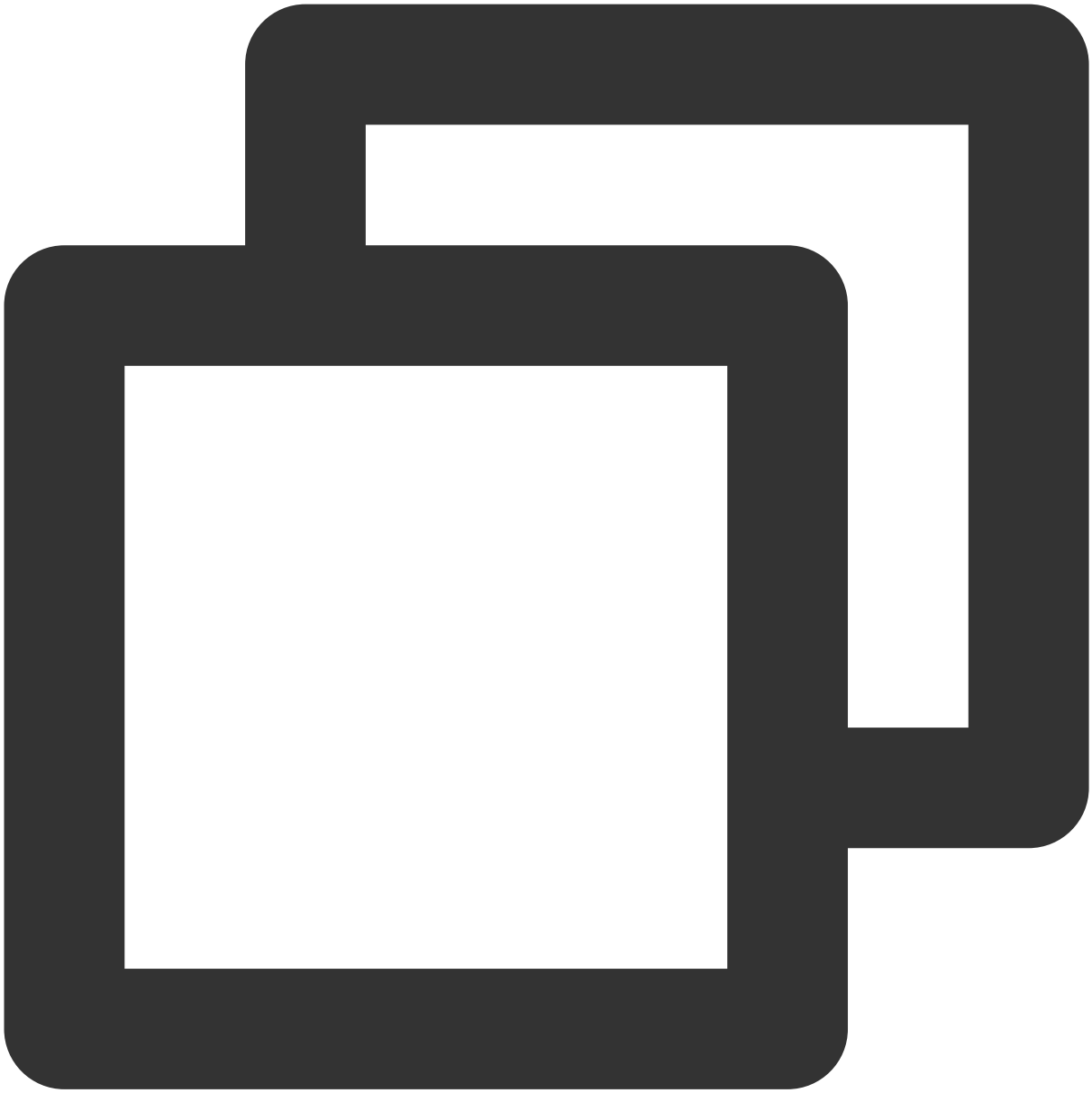
接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。



```
Future<TUIResult> accept()
```

reject

拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。

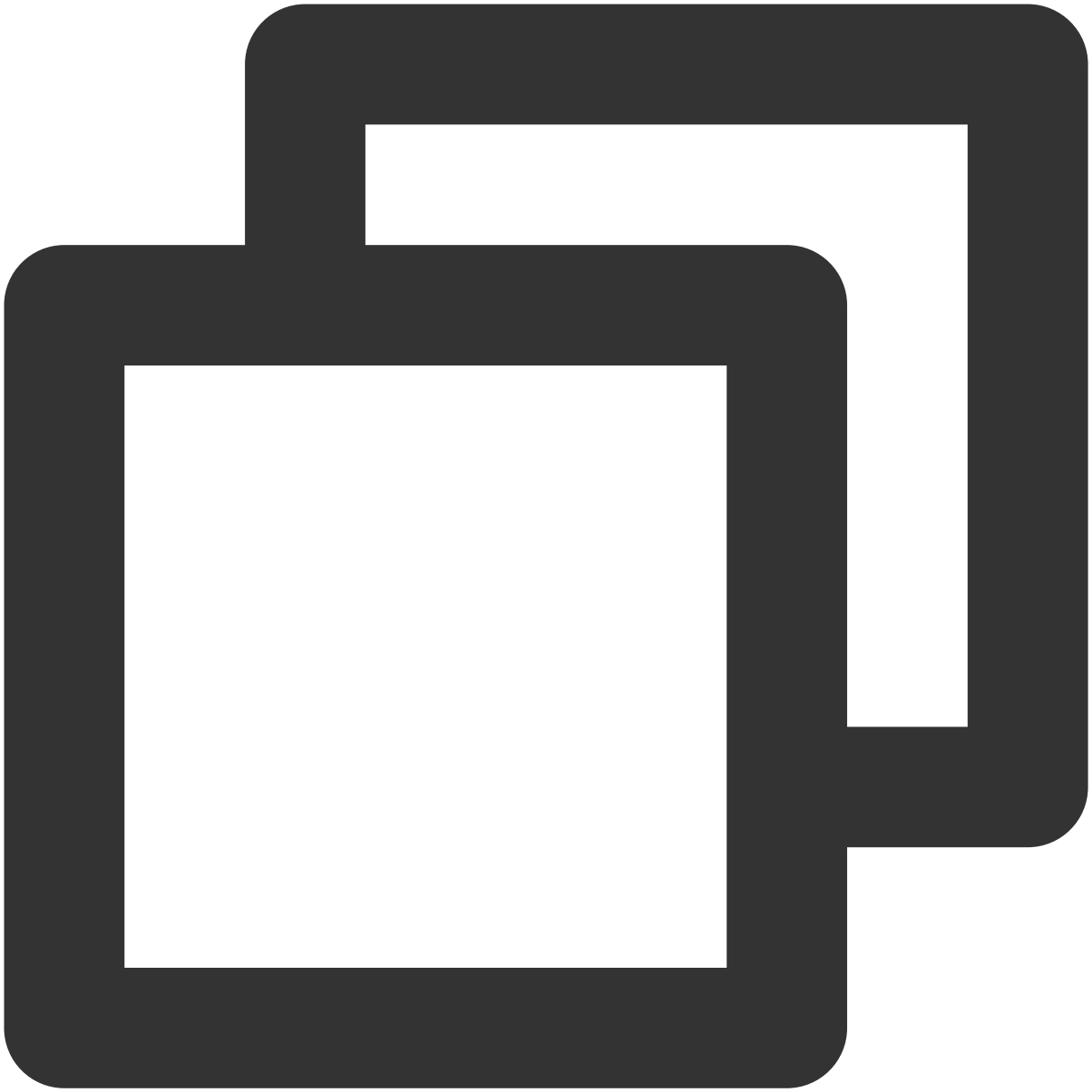


```
Future<TUIResult> reject()
```

ignore

忽略当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数忽略来电，此时主叫会收到 `onUserLineBusy` 的回调。

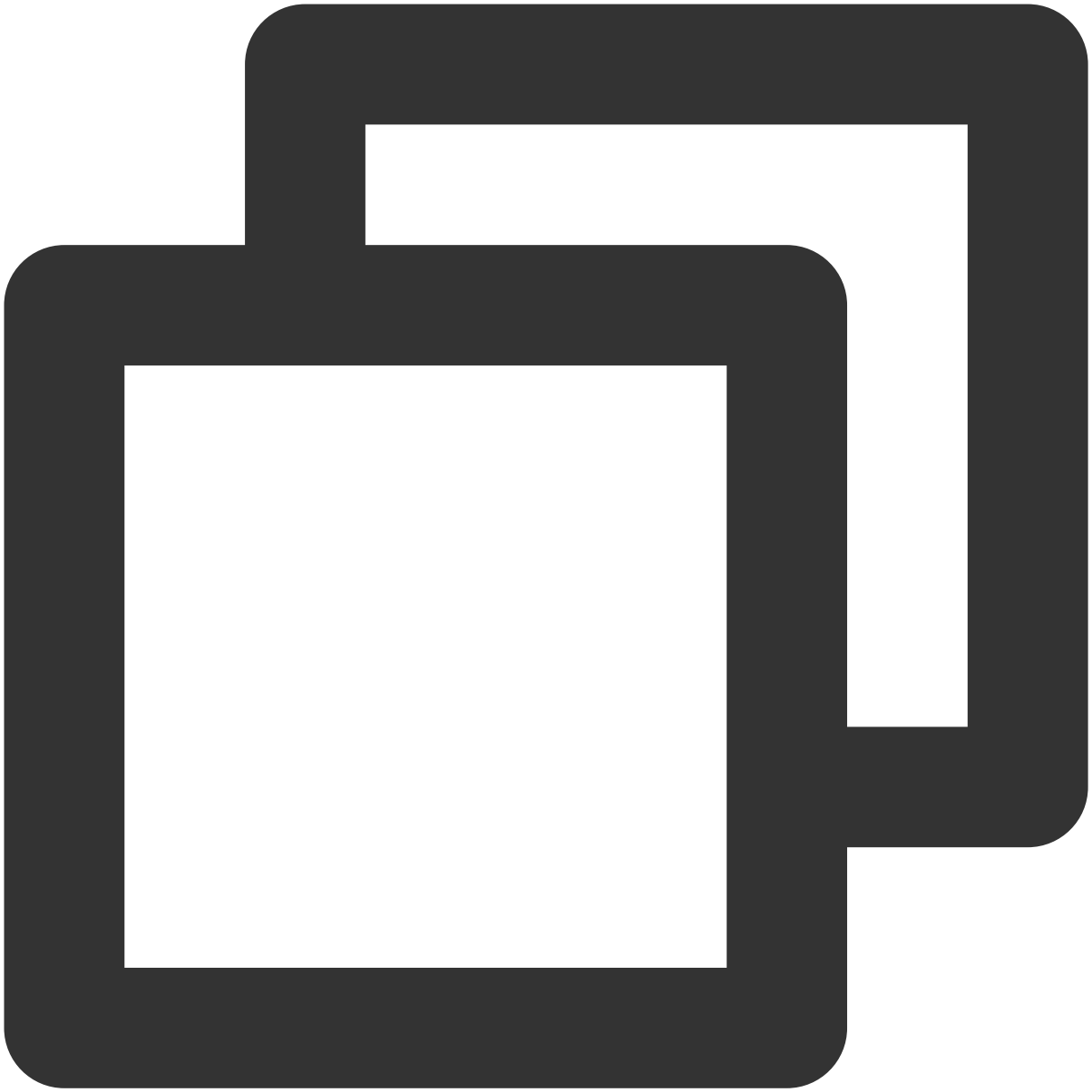
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。



```
Future<TUIResult> ignore()
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

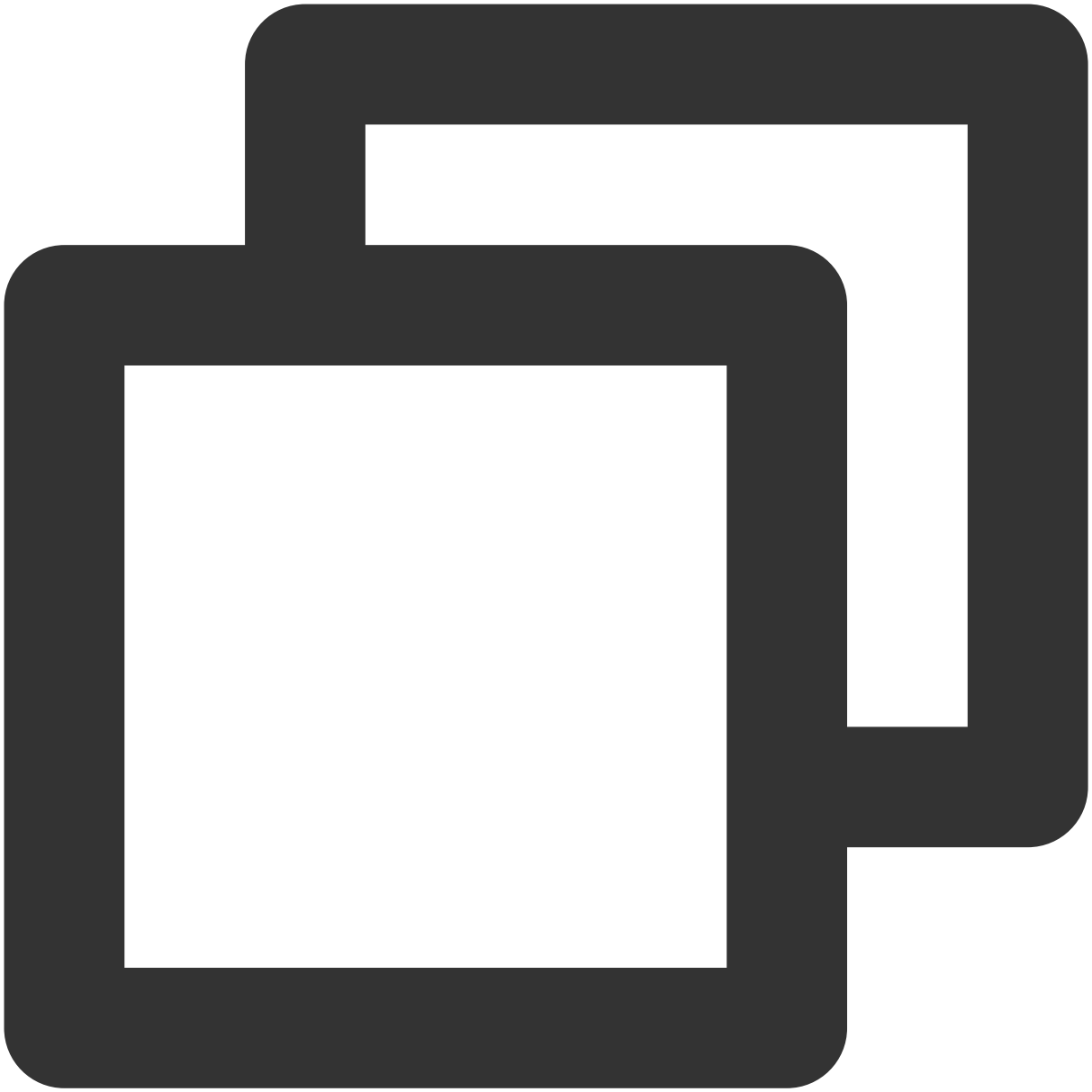


```
Future<TUIResult> hangup()
```

inviteUser

邀请用户加入此次群组通话。

使用场景：一个群组通话中的用户主动邀请其他人时使用。



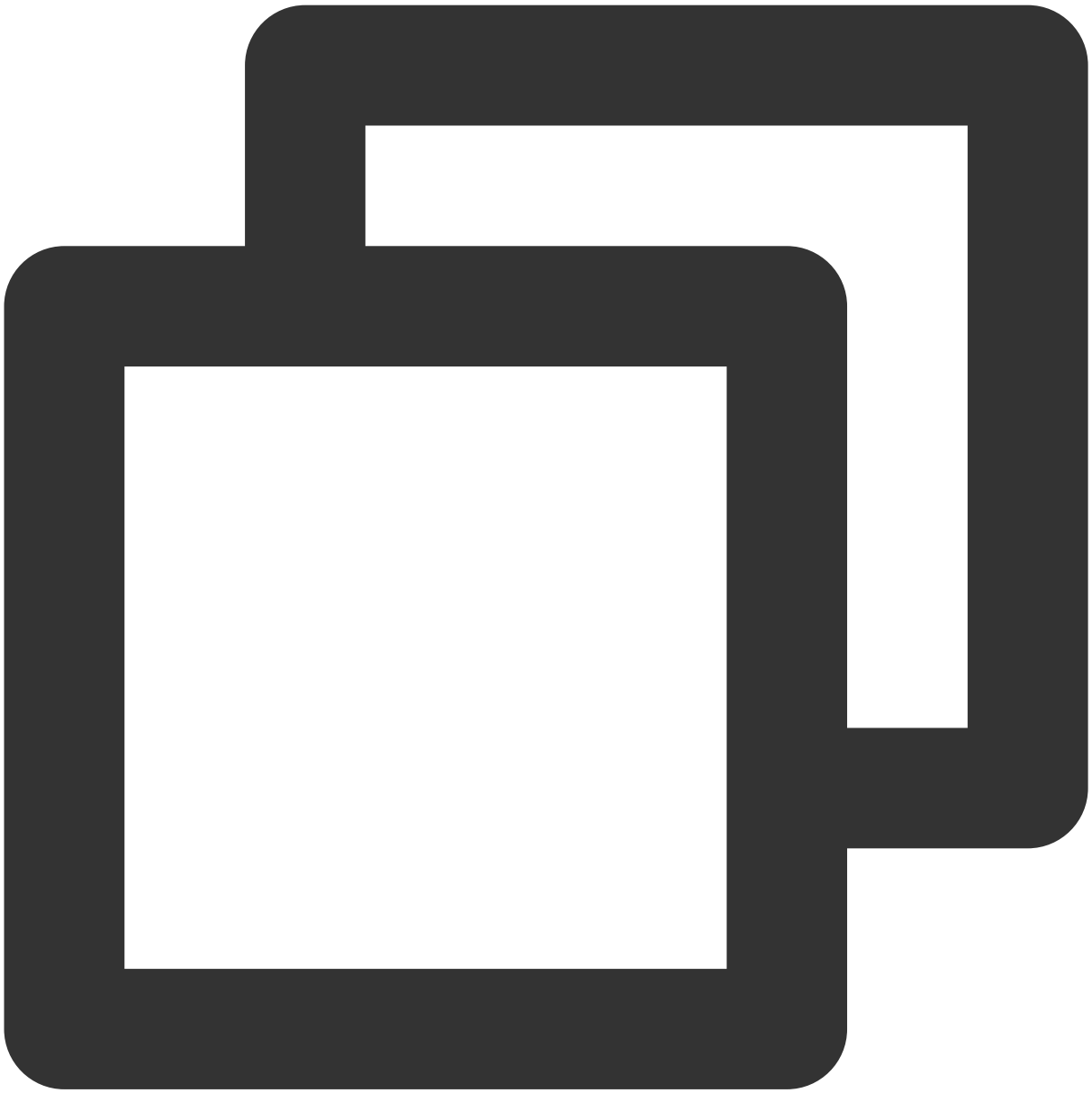
```
Future<void> iniviteUser(List<String> userIdList, TUICallParams params, TUIValueCal
```

参数	类型	含义
userIdList	List<String>	目标用户的 userId 列表
params	TUICallParams	通话参数扩展字段，例如：离线推送自定义内容

joinInGroupCall

主动加入此次群组通话。

使用场景：群组内用户主动加入此次群组通话使用。



```
Future<TUIResult> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaTyp
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID

mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

switchCallMediaType

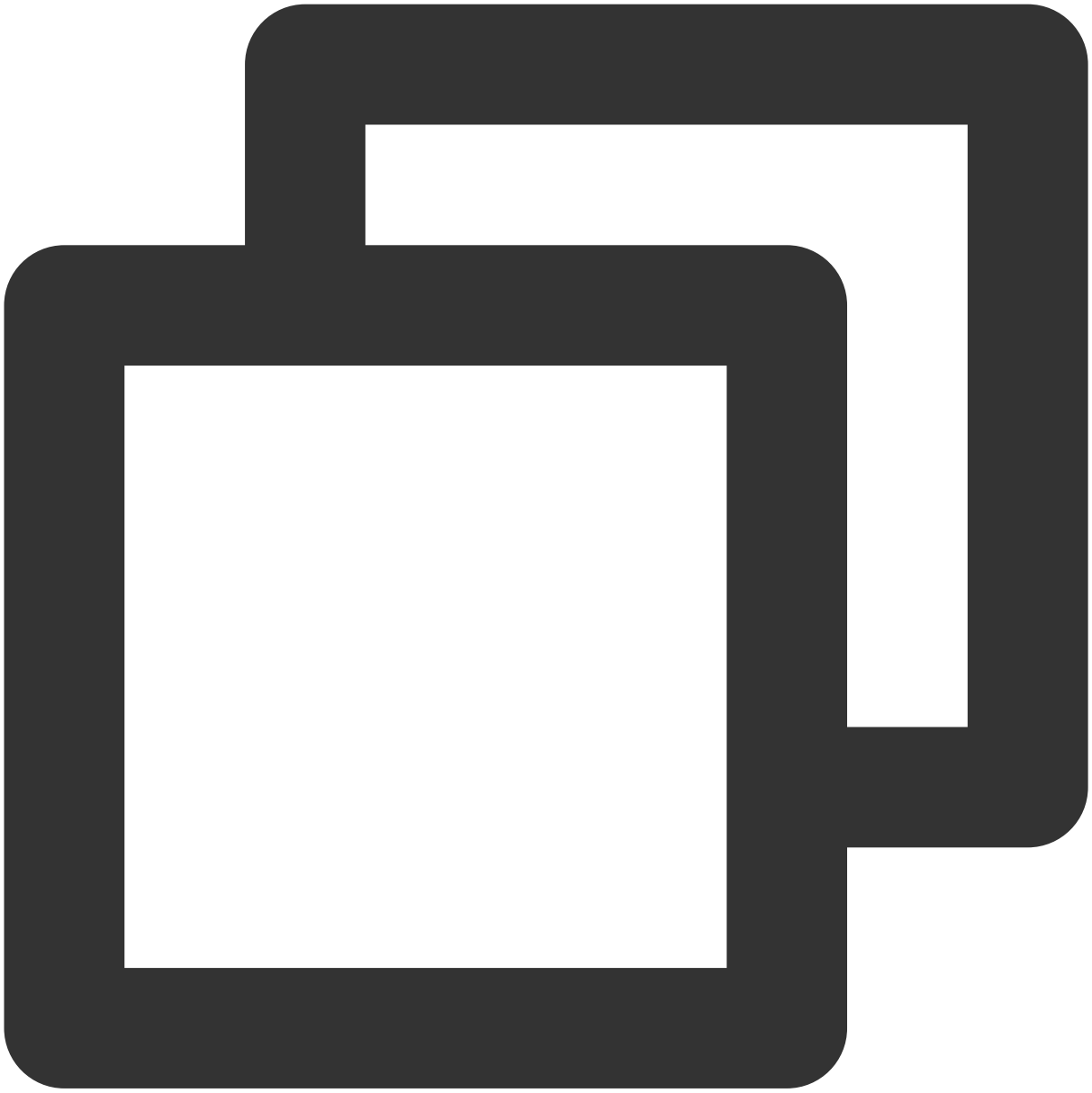
切换视频通话到语音通话。

```
Future<void> switchCallMediaType(TUICallMediaType mediaType)
```

参数	类型	含义
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

startRemoteView

设置显示视频画面的 View 对象。

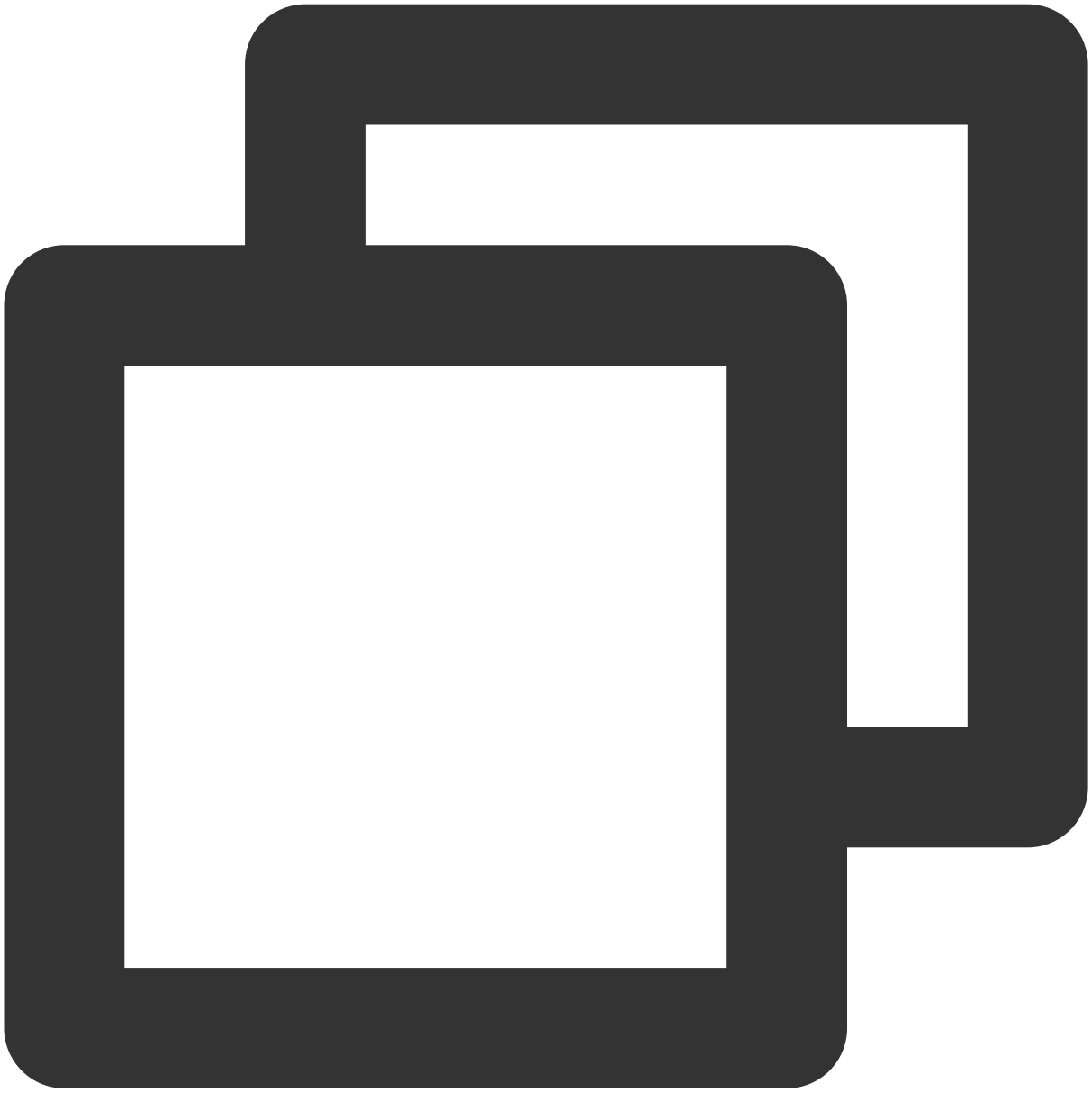


```
Future<void> startRemoteView(String userId, intviewId)
```

参数	类型	含义
userId	String	目标用户的 userId
intviewId	int？	显示视频画面的widget对应的Id

stopRemoteview

停止订阅远端用户的视频数据。

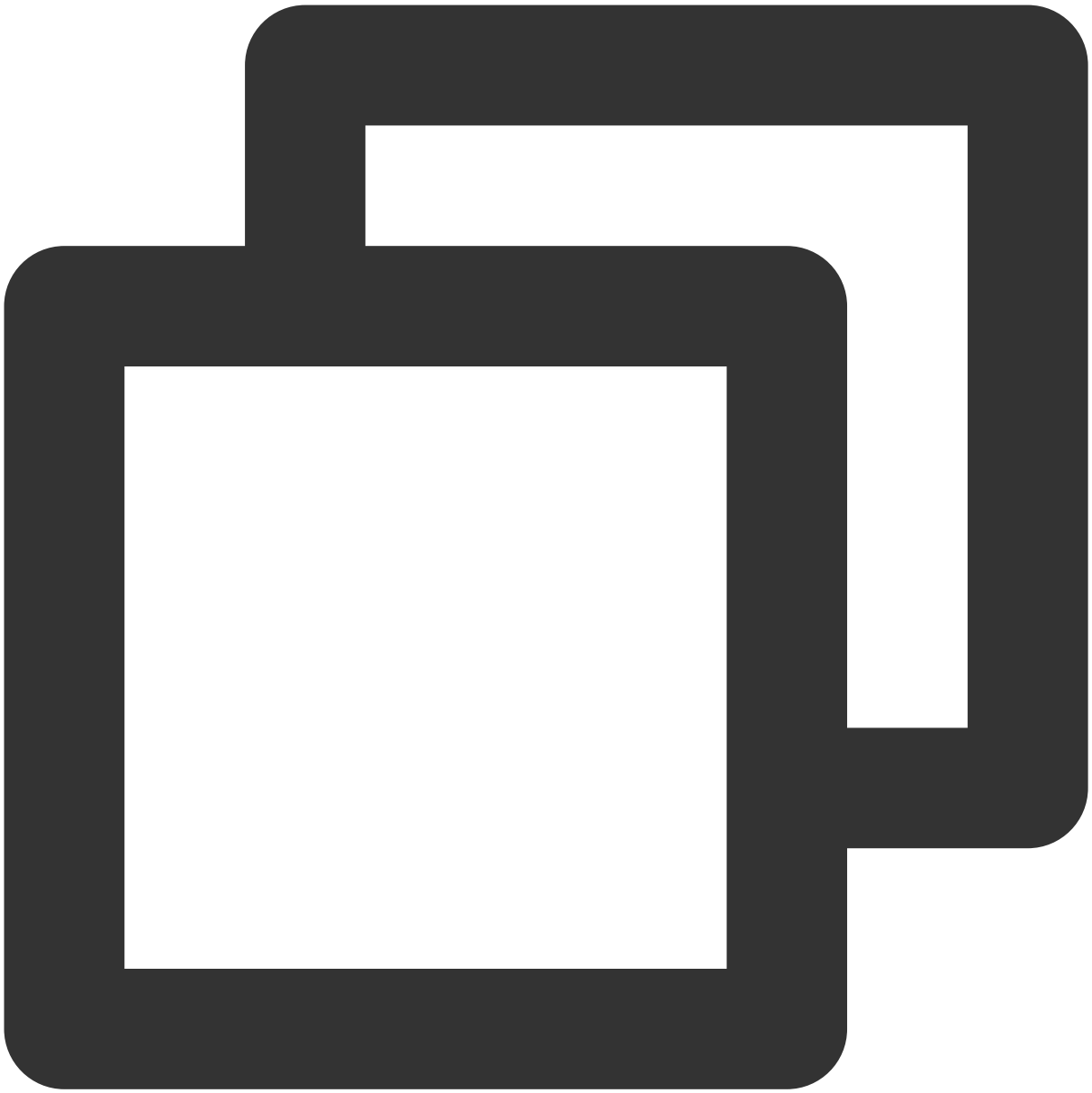


```
Future<void> stopRemoteView(String userId)
```

参数	类型	含义
userId	String	目标用户的 userId

openCamera

开启摄像头。

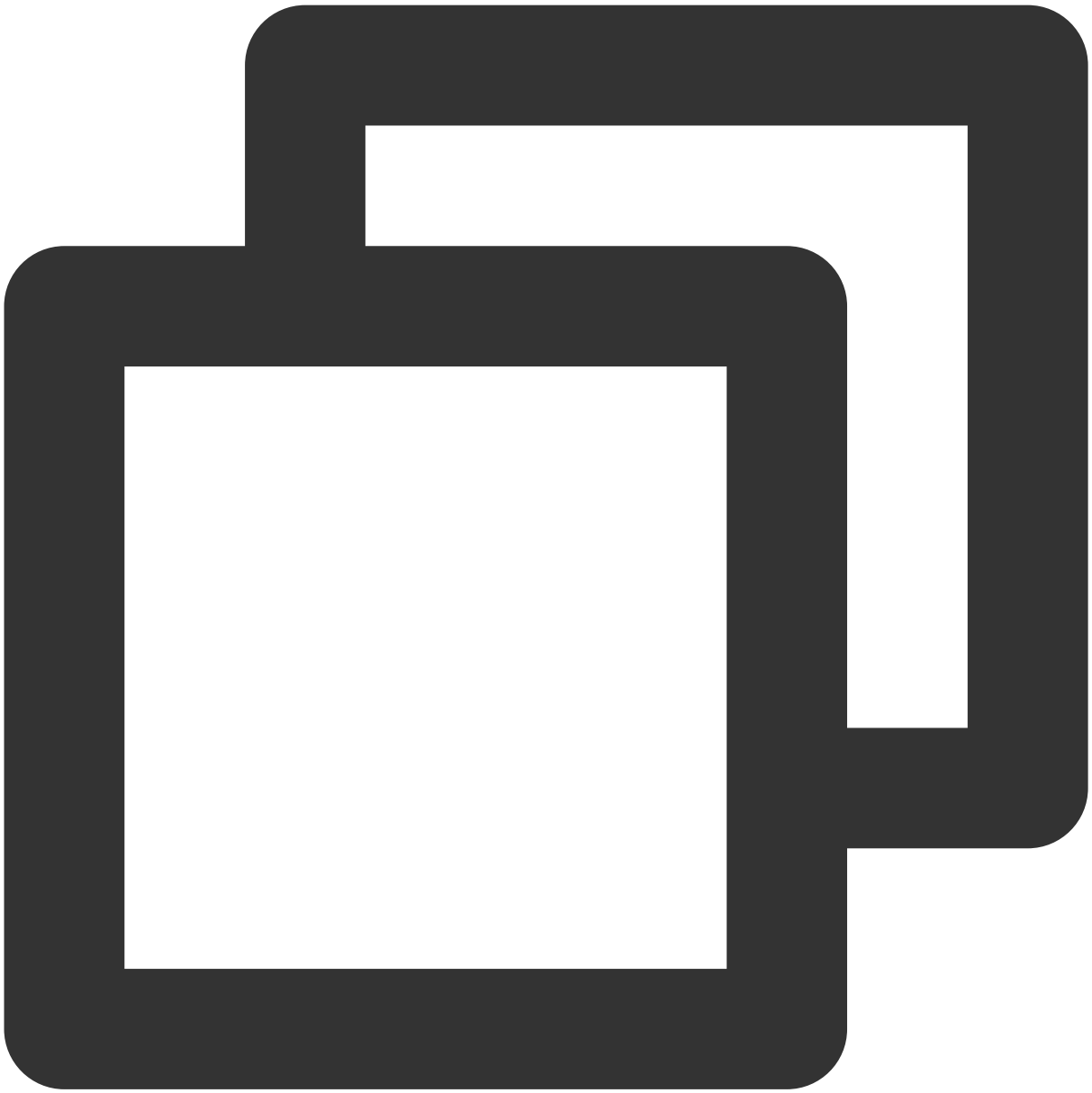


```
Future<TUIResult> openCamera(TUICamera camera, int? viewId)
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头
viewId	int?	显示视频画面的widget对应的Id

closeCamera

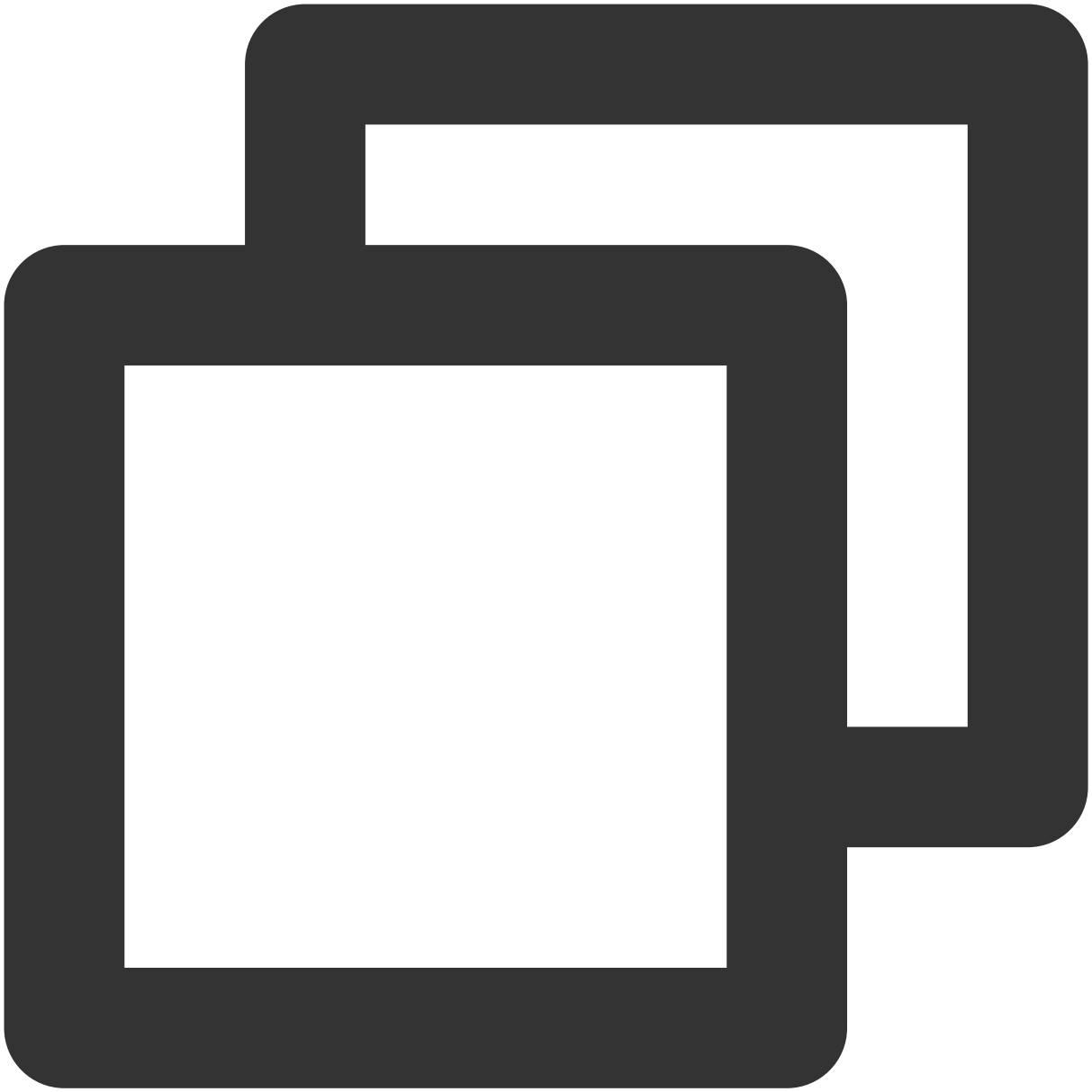
关闭摄像头。



```
Future<void> closeCamera()
```

switchCamera

切换前后摄像头。



```
Future<void> switchCamera(TUICamera camera)
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头

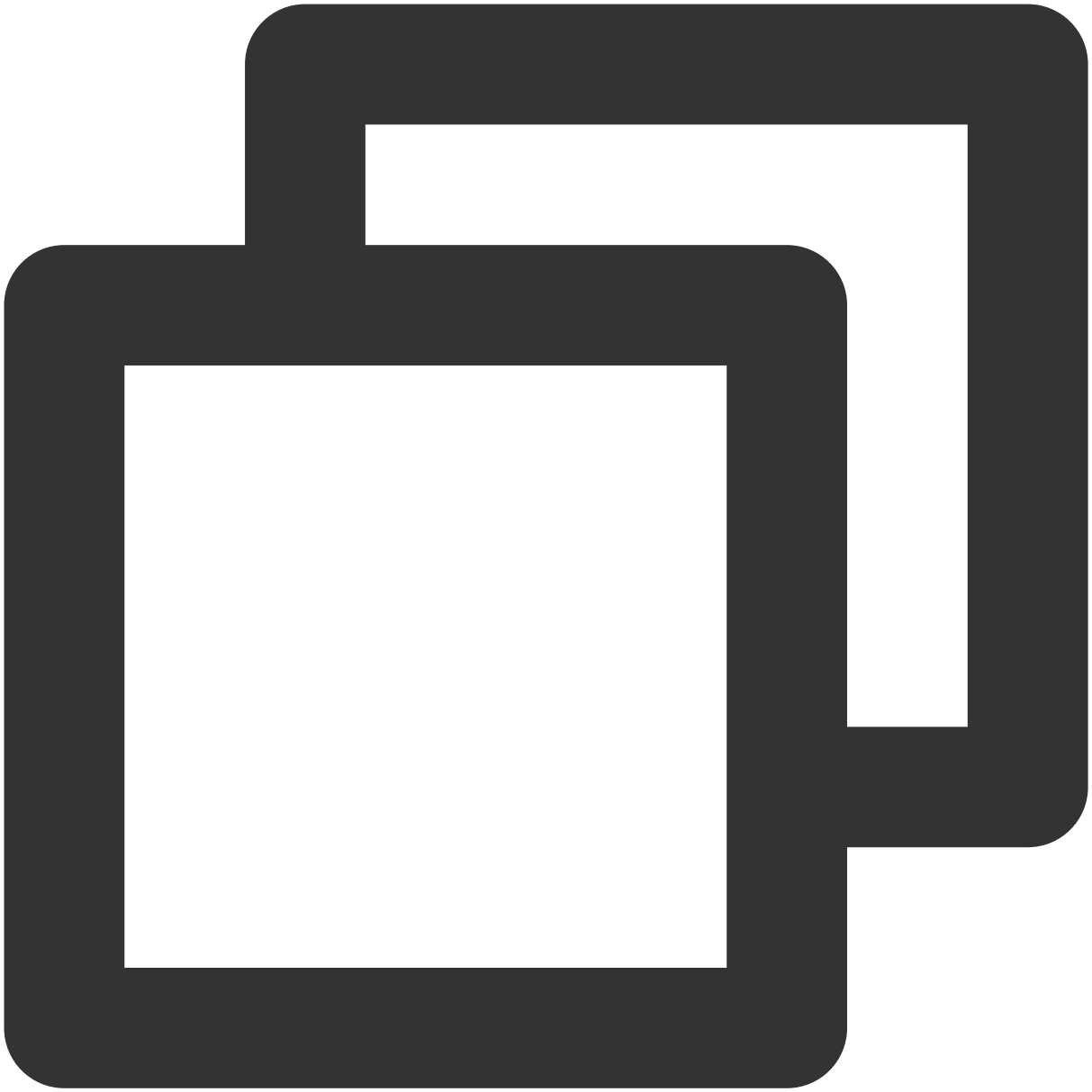
openMicrophone

打开麦克风。

```
Future<TUIResult> openMicrophone()
```

closeMicrophone

关闭麦克风。

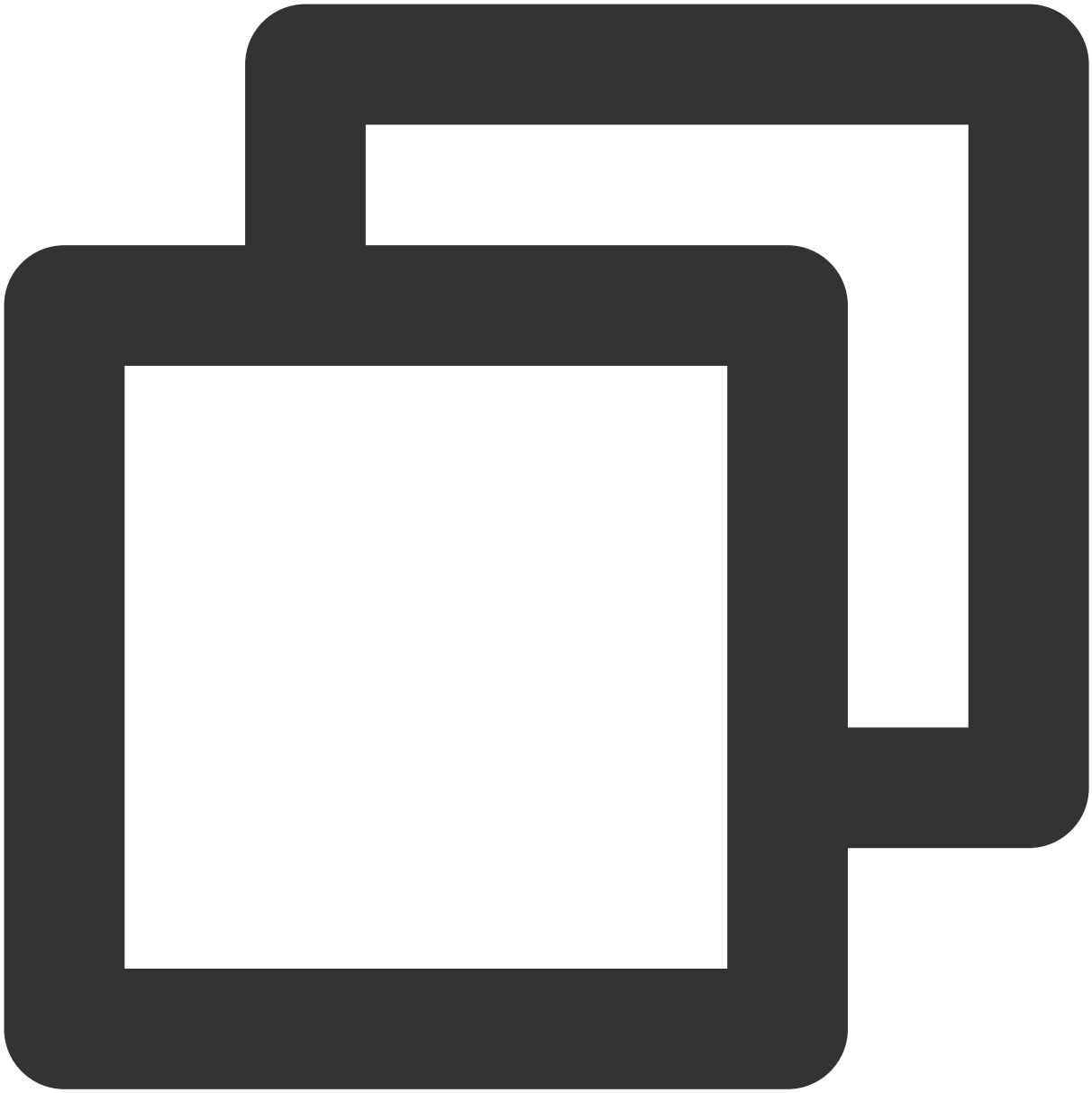


```
Future<void> closeMicrophone()
```

selectAudioPlaybackDevice

选择音频播放设备。

目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。



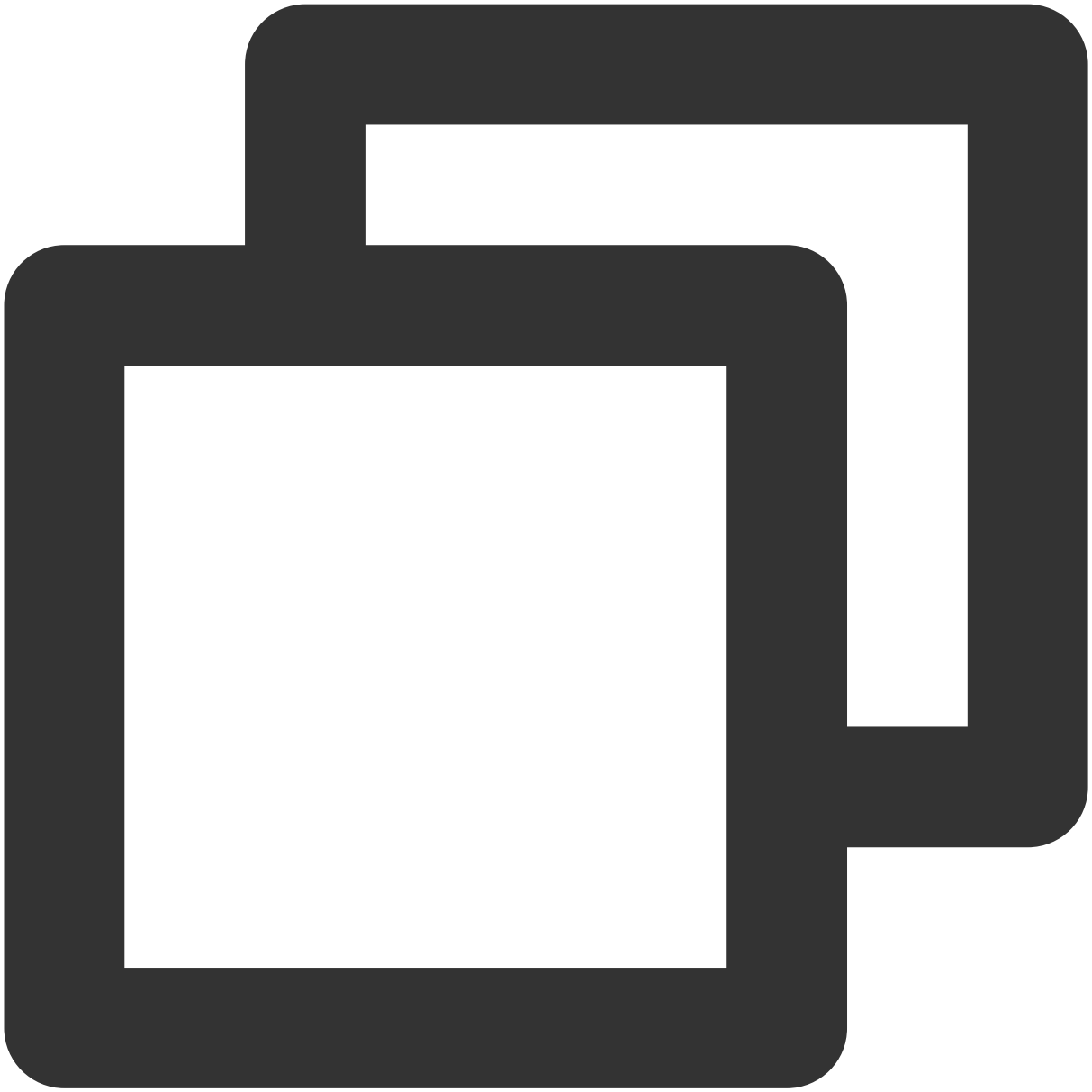
```
Future<void> selectAudioPlaybackDevice(TUIAudioPlaybackDevice device)
```

参数	类型	含义
device	TUIAudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。

用户昵称不能超过500字节，用户头像必须是 URL 格式。

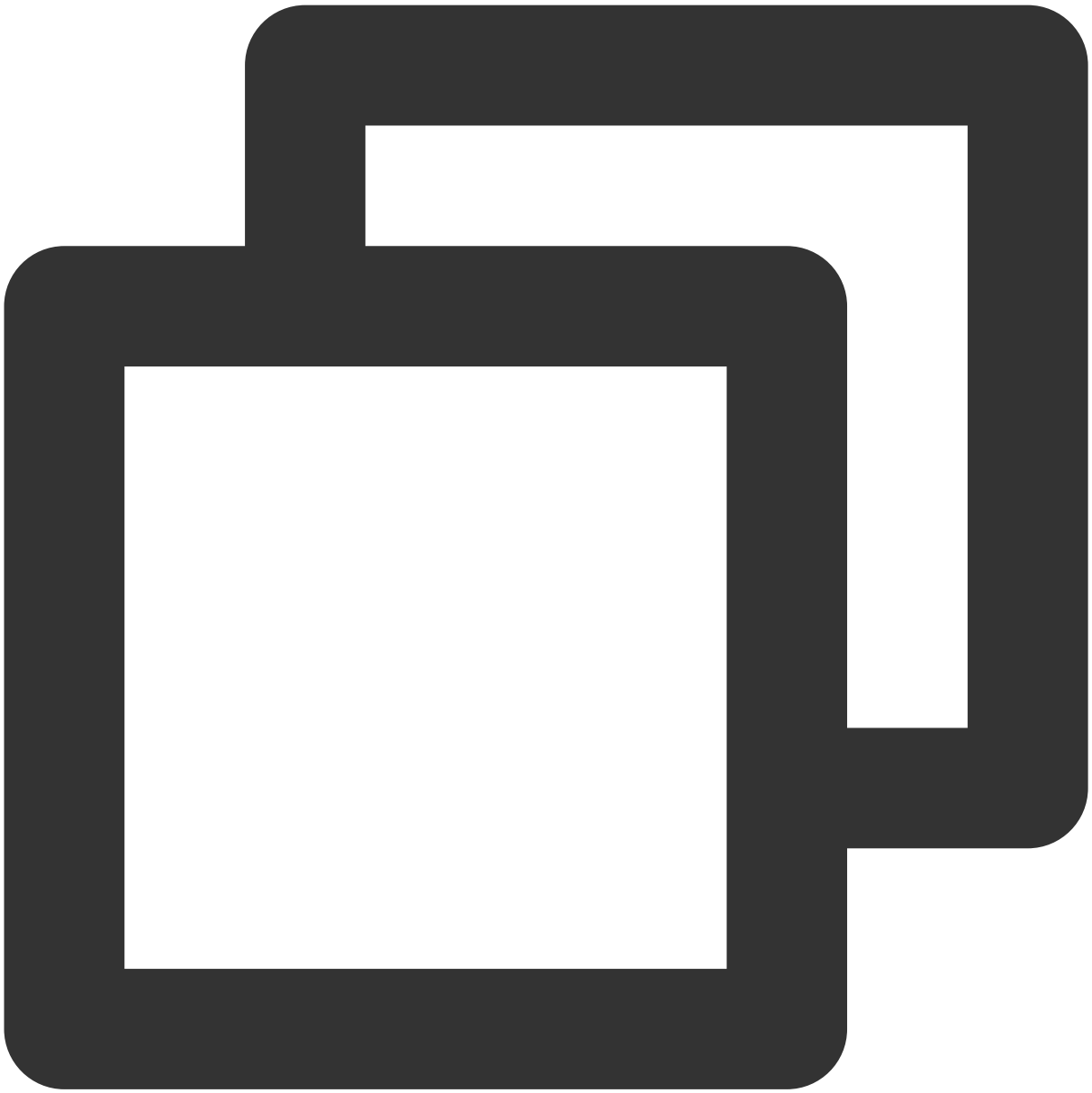


```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

参数	类型	含义
nickname	String	用户昵称
avatar	String	用户头像（格式为 URL）

enableMultiDeviceAbility

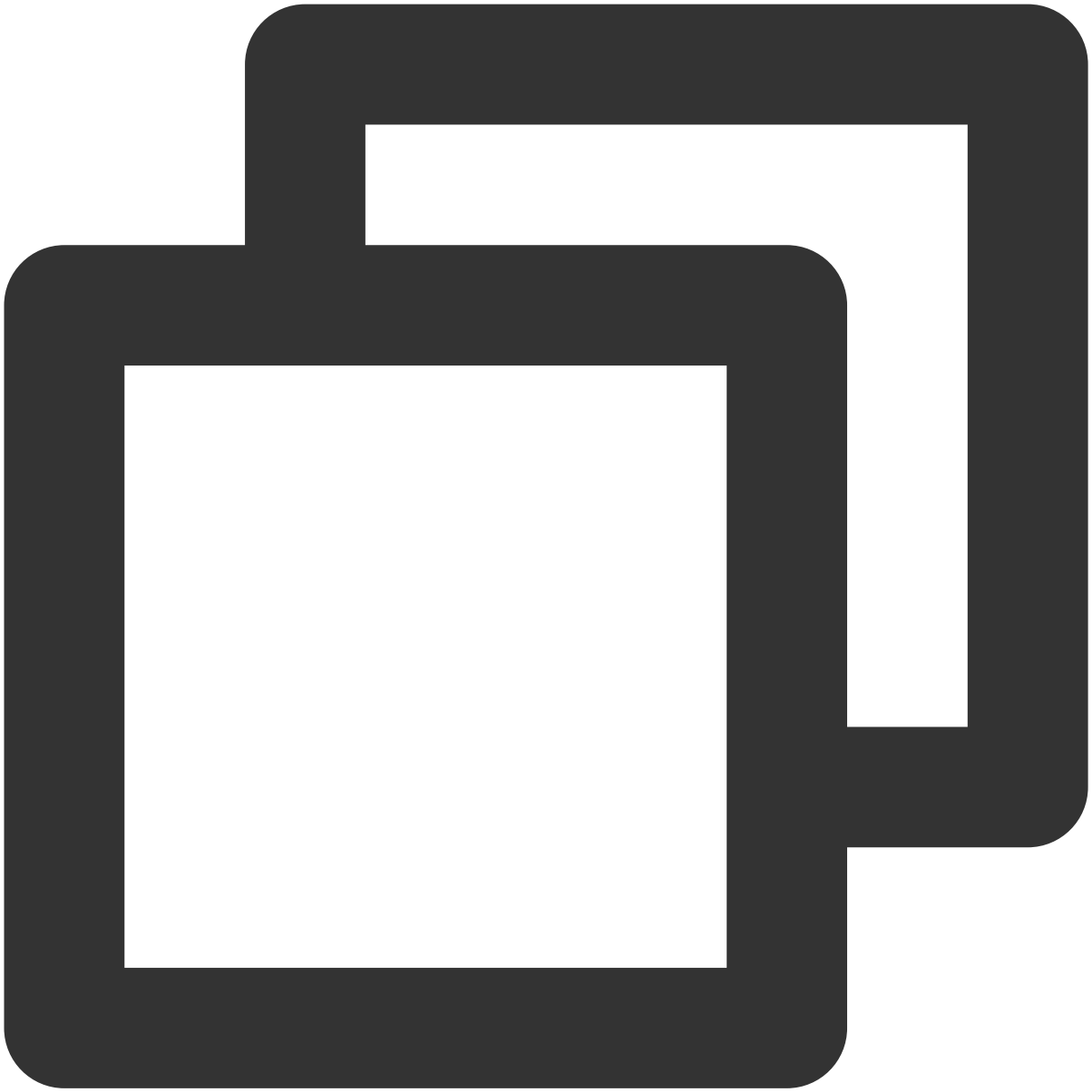
开启/关闭 TUICallEngine 的多设备登录模式 （尊享版套餐支持）。



```
Future<TUIResult> enableMultiDeviceAbility(bool enable)
```

setVideoRenderParams

设置用户视频画面的渲染模式。



```
Future<TUIResult> setVideoRenderParams(String userId, VideoRenderParams params)
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
params	VideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。

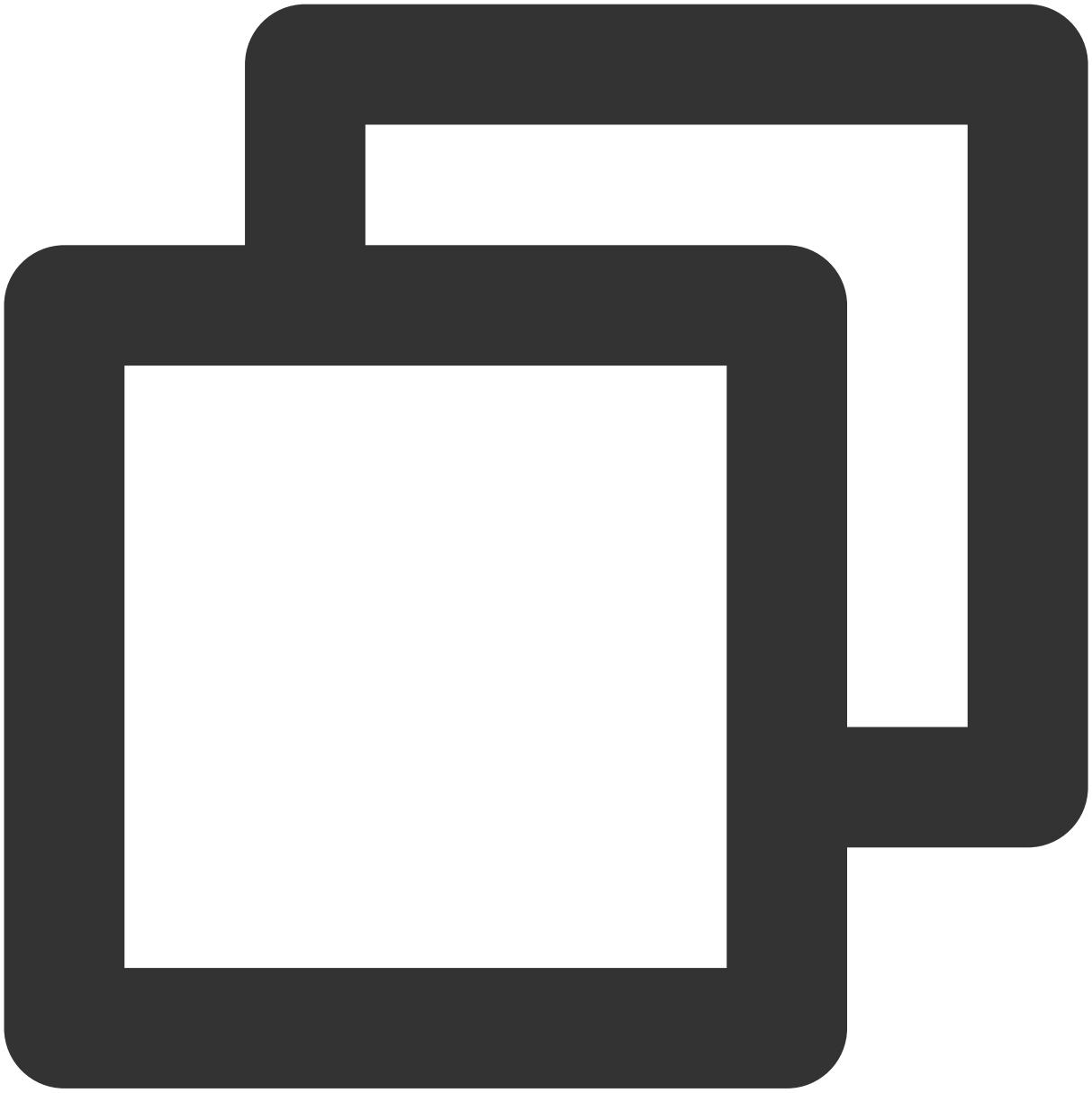
```
Future<TUIResult> setVideoEncoderParams(VideoEncoderParams params)
```

参数如下表所示：

参数	类型	含义
params	VideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高比

queryRecentCalls

查询通话记录。



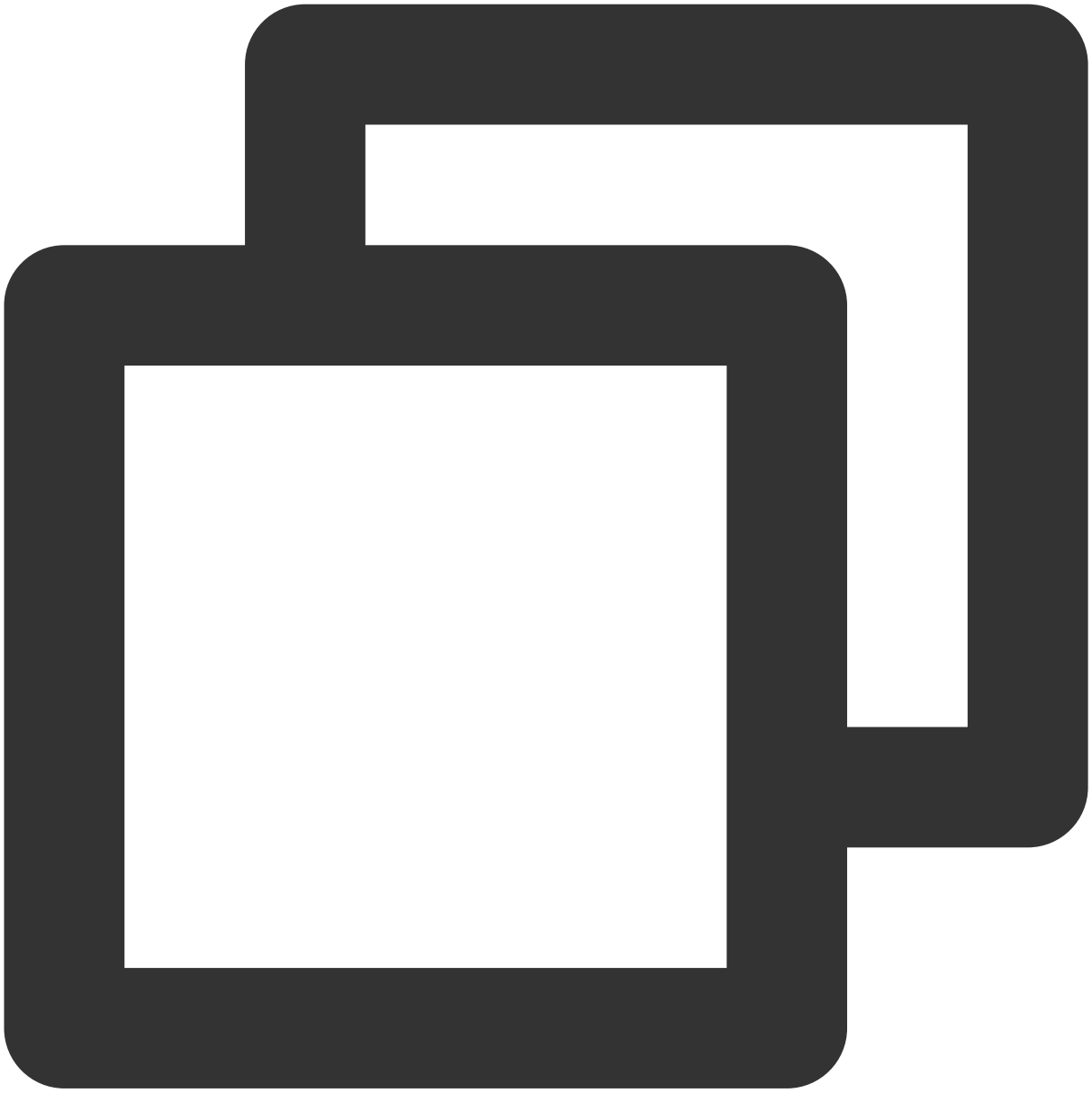
```
Future<void> queryRecentCalls(TUICallRecentCallsFilter filter, TUIValueCallback cal
```

参数如下表所示：

参数	类型	含义
filter	TUICallRecentCallsFilter	查询过滤条件

deleteRecordCalls

删除通话记录。



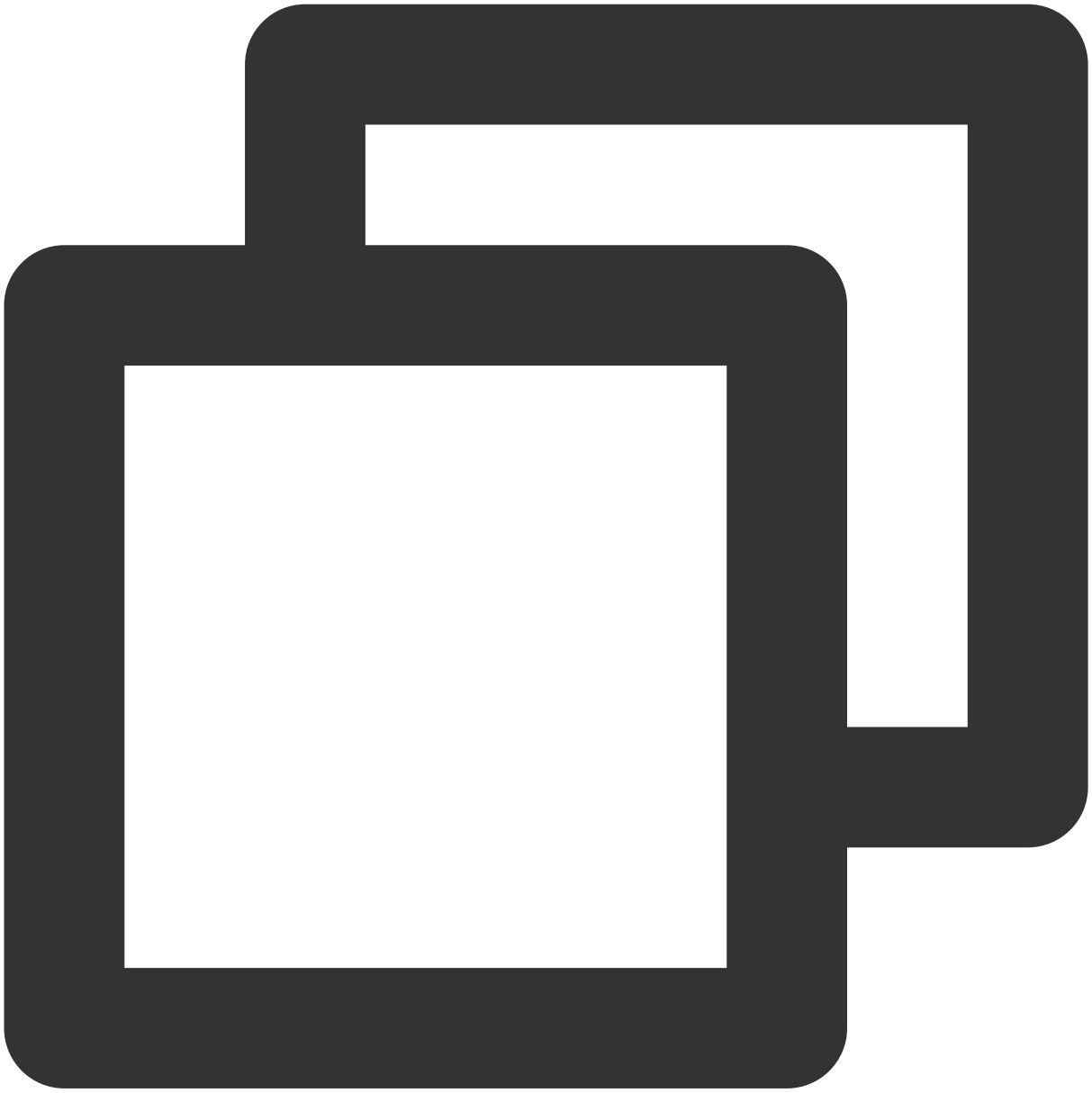
```
Future<void> deleteRecordCalls(List<String> callIdList, TUIValueCallback callback)
```

参数如下表所示：

参数	类型	含义
callIdList	List<String>	待删除记录的Id列表

setBeautyLevel

设置美颜级别。



```
Future<TUIResult> setBeautyLevel(double level)
```

参数如下表所示：

参数	类型	含义
level	double	美颜等级，使用范围0.0～10.0

版权所有：腾讯云计算（北京）有限责任公司

TUICallObserver

最近更新时间：2024-01-25 16:15:40

TUICallObserver API 简介

TUICallObserver 是 TUICallKit 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线

版权所有：腾讯云计算（北京）有限责任公司

onUserSigExpired	在线时票据过期

回调事件详情

通过 addObserver 监听Flutter插件抛出的事件。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {
      //您的回调处理逻辑
```

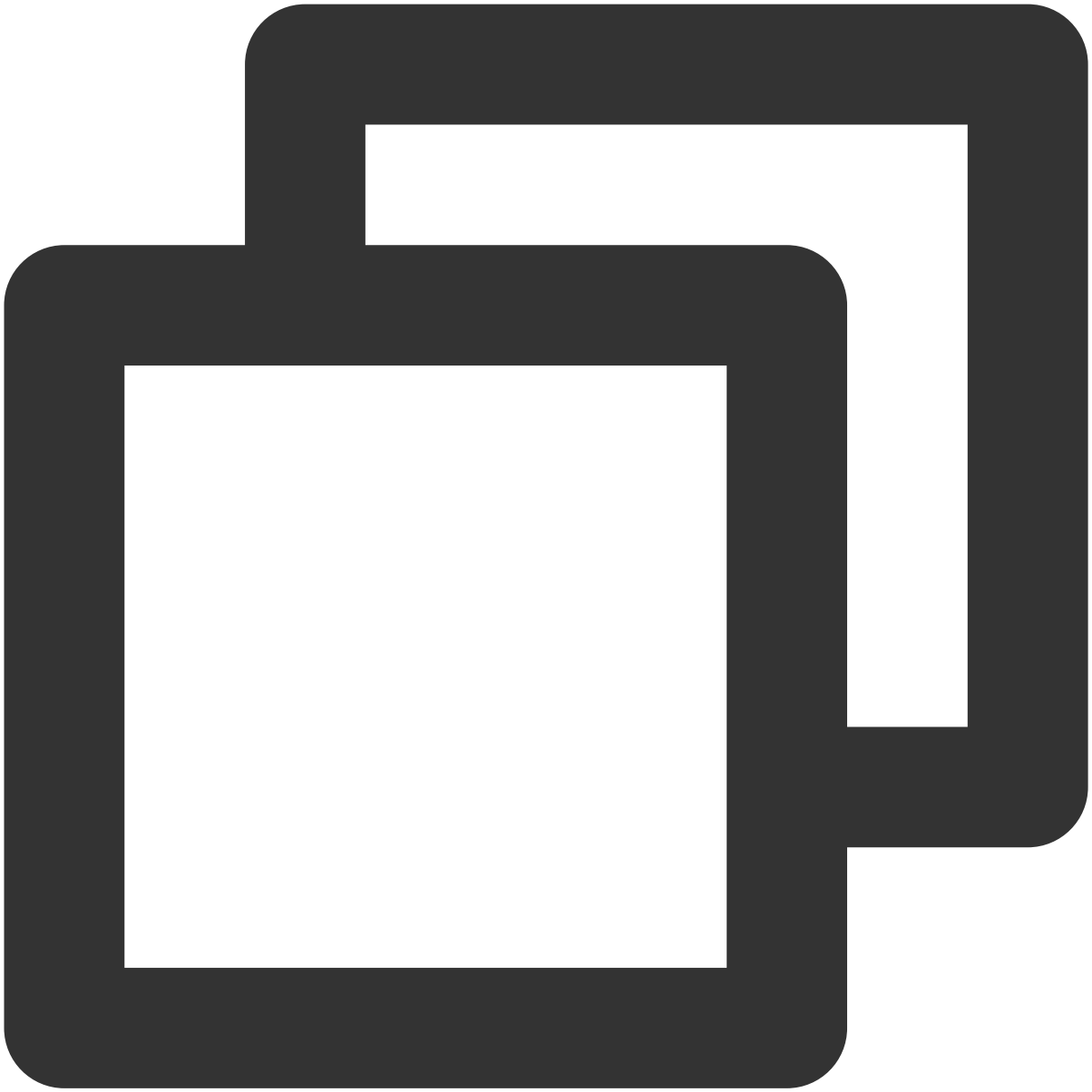
```
    },onCallCancelled: (String callerId) {
      //您的回调处理逻辑
    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole
      //您的回调处理逻辑
    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
      //您的回调处理逻辑
    }, onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType
      //您的回调处理逻辑
    }, onUserReject: (String userId) {
      //您的回调处理逻辑
    }, onUserNoResponse: (String userId) {
      //您的回调处理逻辑
    }, onUserLineBusy: (String onUserLineBusy) {
      //您的回调处理逻辑
    }, onUserJoin: (String userId) {
      //您的回调处理逻辑
    }, onUserLeave: (String userId) {
      //您的回调处理逻辑
    }, onUserVideoAvailable: (String userId, bool isVideoAvailable) {
      //您的回调处理逻辑
    }, onUserAudioAvailable: (String userId, bool isAudioAvailable) {
      //您的回调处理逻辑
    }, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList
      //您的回调处理逻辑
    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,
      //您的回调处理逻辑
    }, onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {
      //您的回调处理逻辑
    }, onKickedOffline: () {
      //您的回调处理逻辑
    }, onUserSigExpired: () {
      //您的回调处理逻辑
    }
));
```

onError

错误事件回调。

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。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {
      //您的回调处理逻辑
    }
));
```

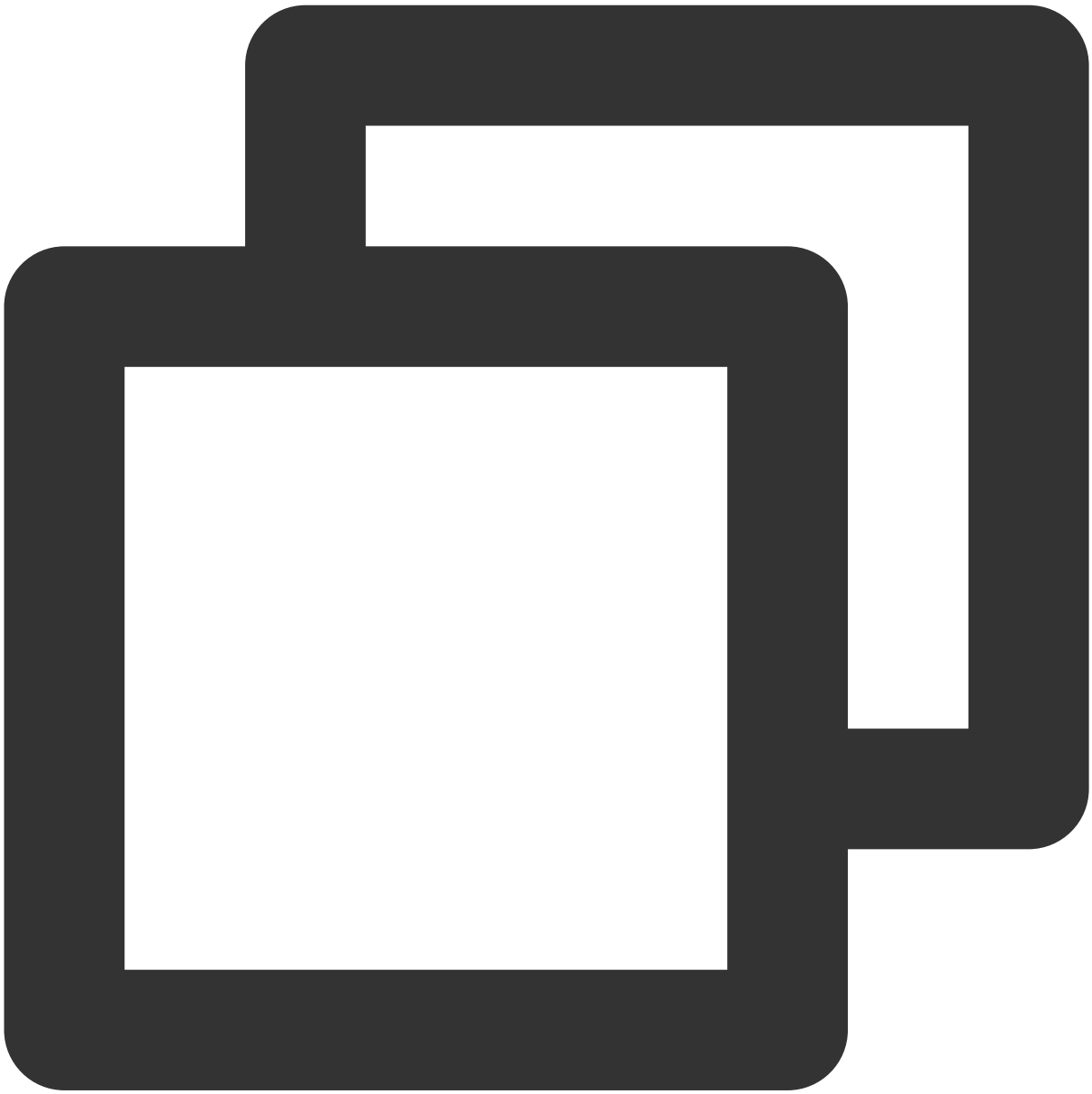
参数如下表所示：

参数	类型	含义
code	int	错误码

message	String	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallReceived: (String callerId, List<String> calleeIdList, String groupId, TU
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
callerId	String	主叫 ID（邀请方）
calleeIdList	List<String>	被叫 ID 列表（被邀请方）
groupId	String	群组ID
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）

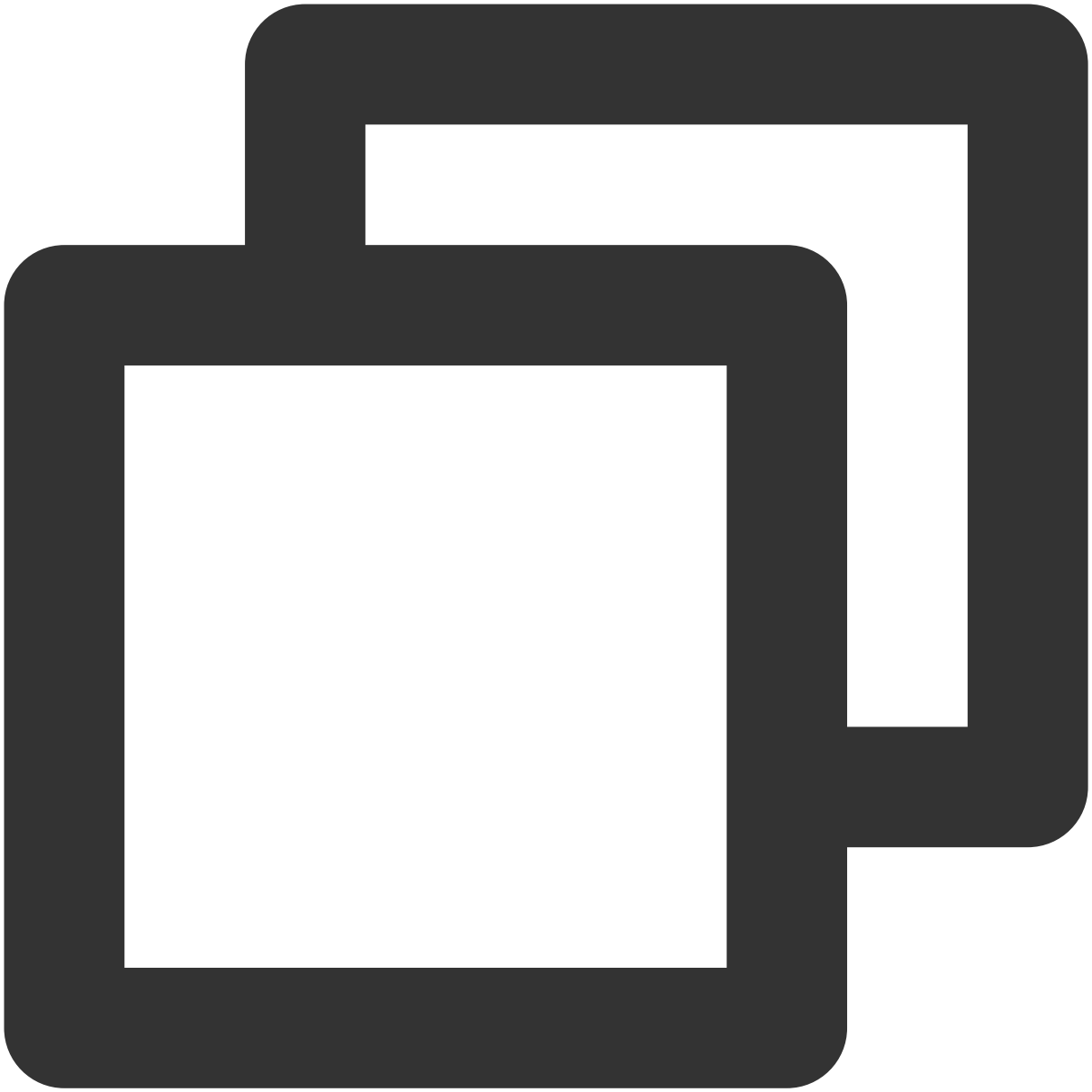
被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）

被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）

被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）

异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）

版权所有：腾讯云计算（北京）有限责任公司



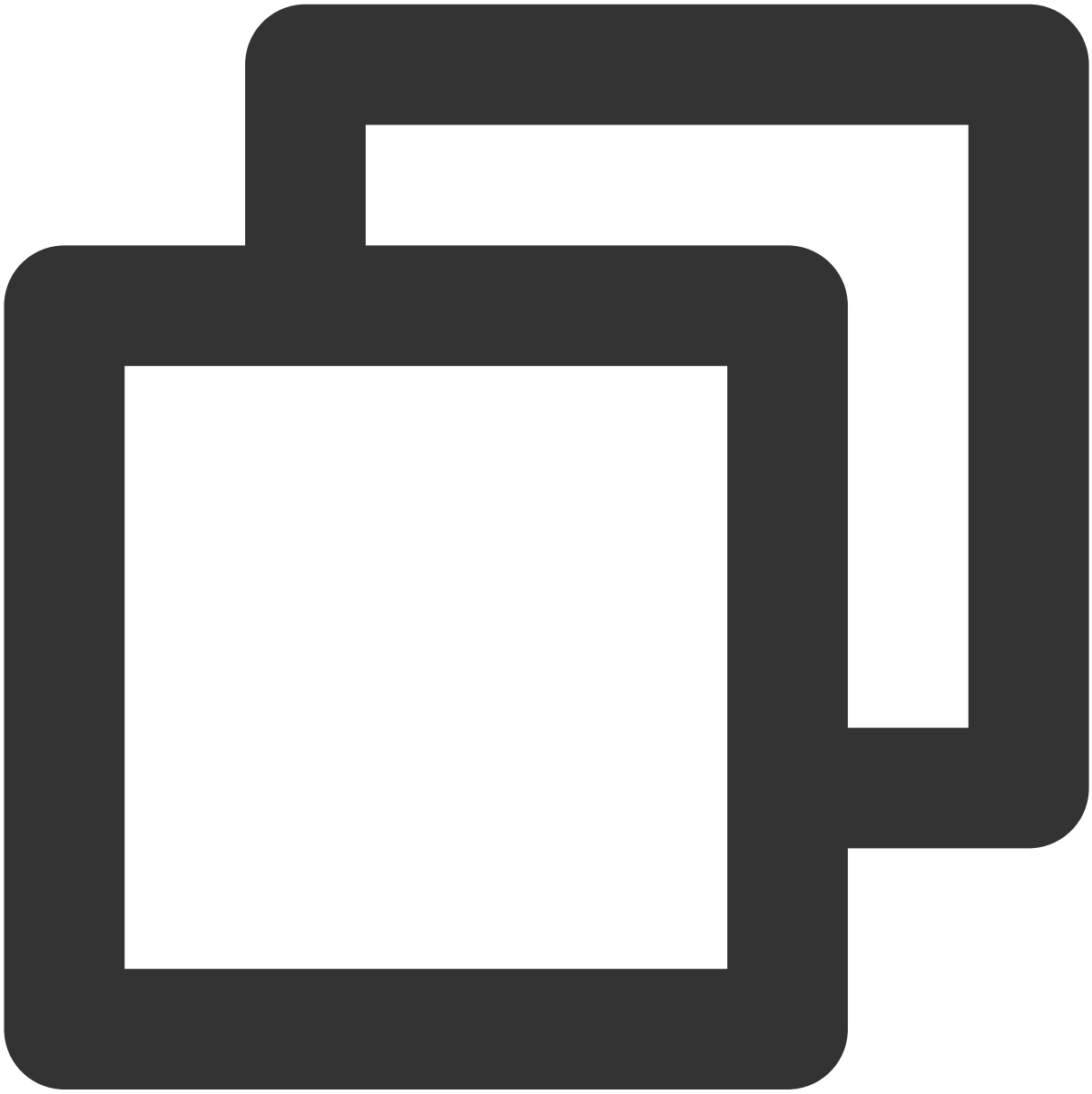
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallCancelled: (String callerId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
callerId	String	取消用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。



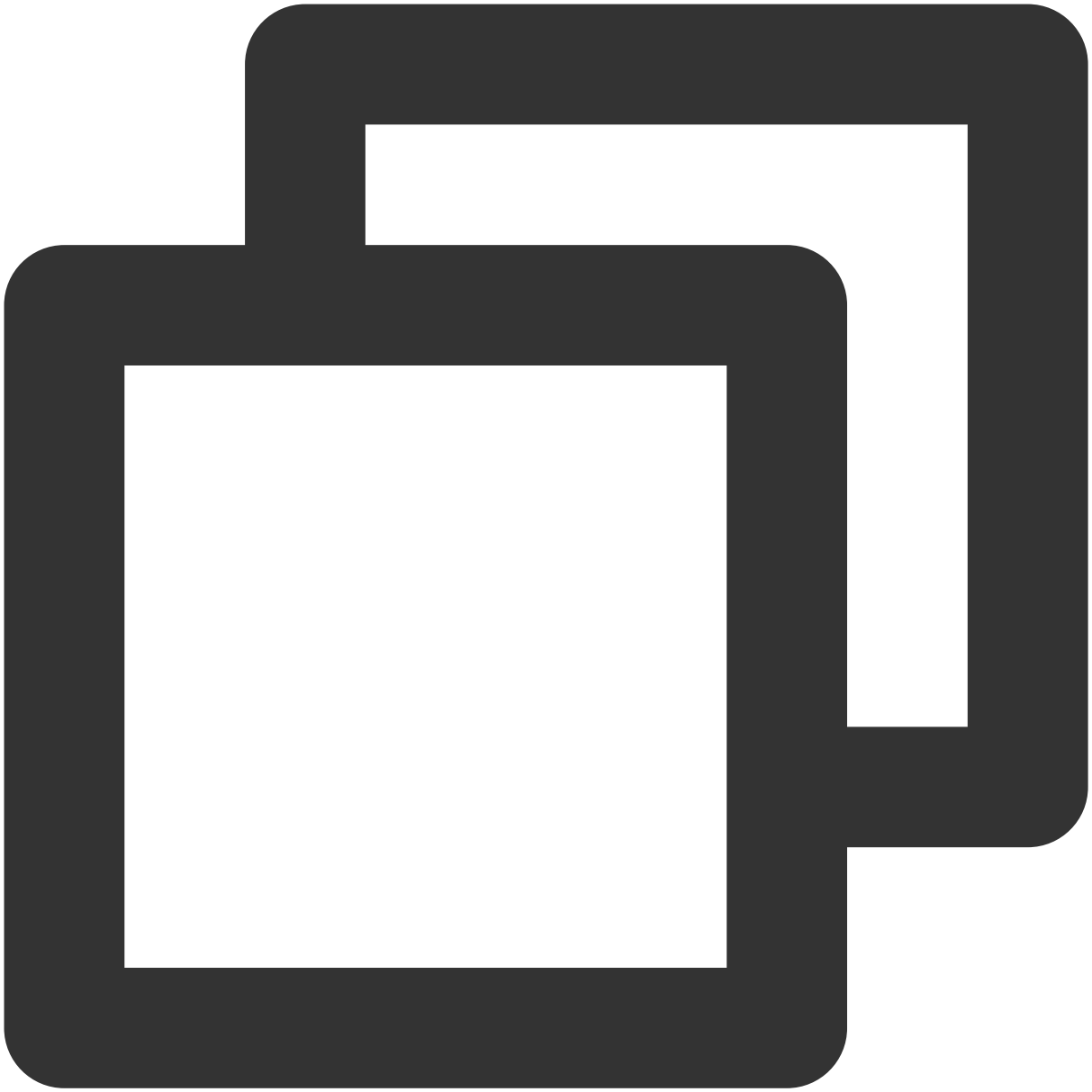
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole cal
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 Id
callMediaType	TUICallMediaType	通话的媒体类型，示例： `TUICallMediaType.video` 或 `TUICallMediaType.audio`
callRole	TUICallRole	角色，枚举类型：主叫(`TUICallRole.caller`)、被叫(`TUICallRole.called`)

onCallEnd

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callR
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 Id

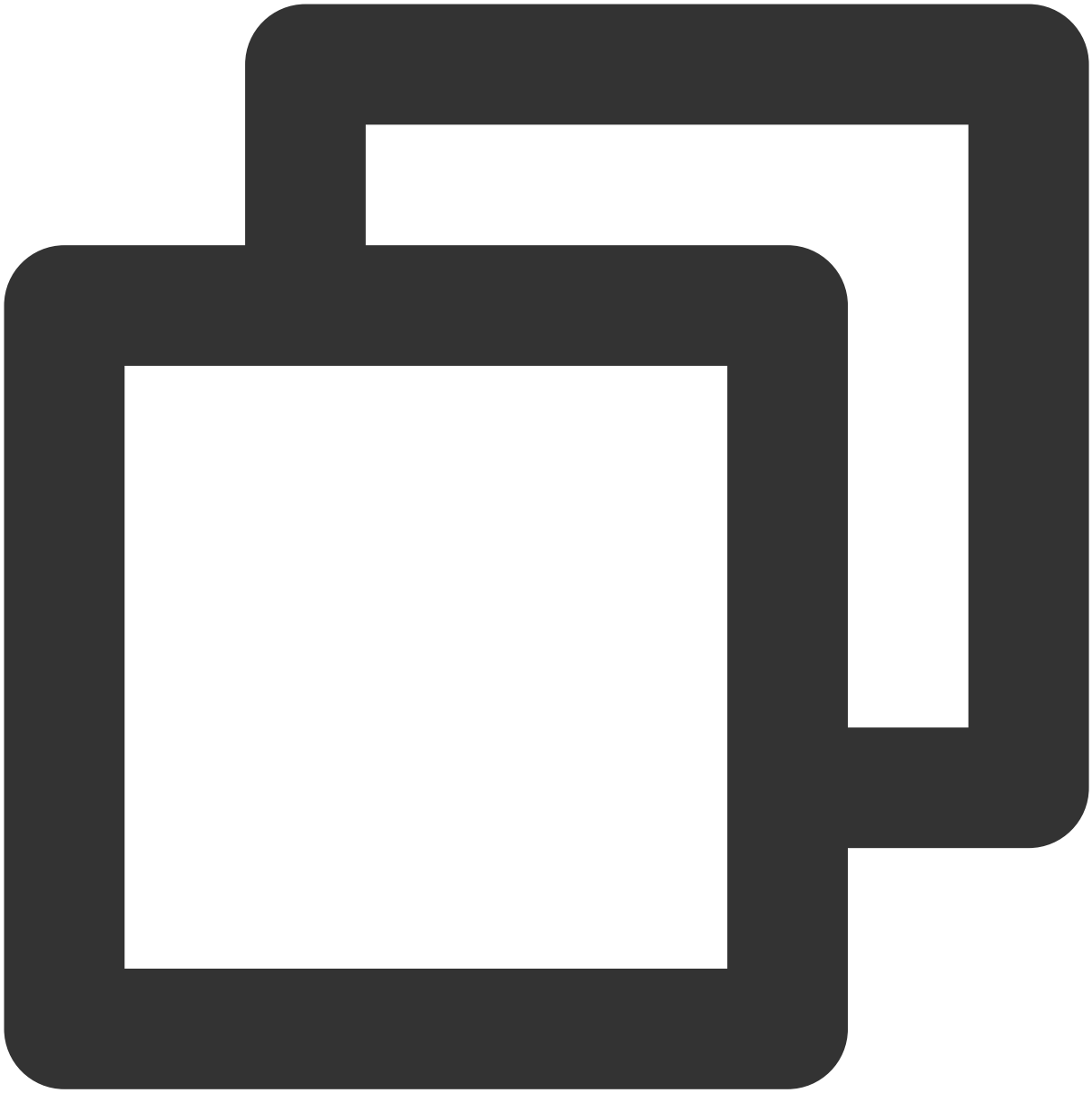
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`
callRole	Number	角色，枚举类型：主叫(callRole = 1)、被叫(callRole = 2)
totalTime	double	此次通话的时长，单位 ms

注意

客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用 REST API 来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType ne
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或

		`TUICallMediaType.audio`
newCallMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserReject: (String userId) {
```

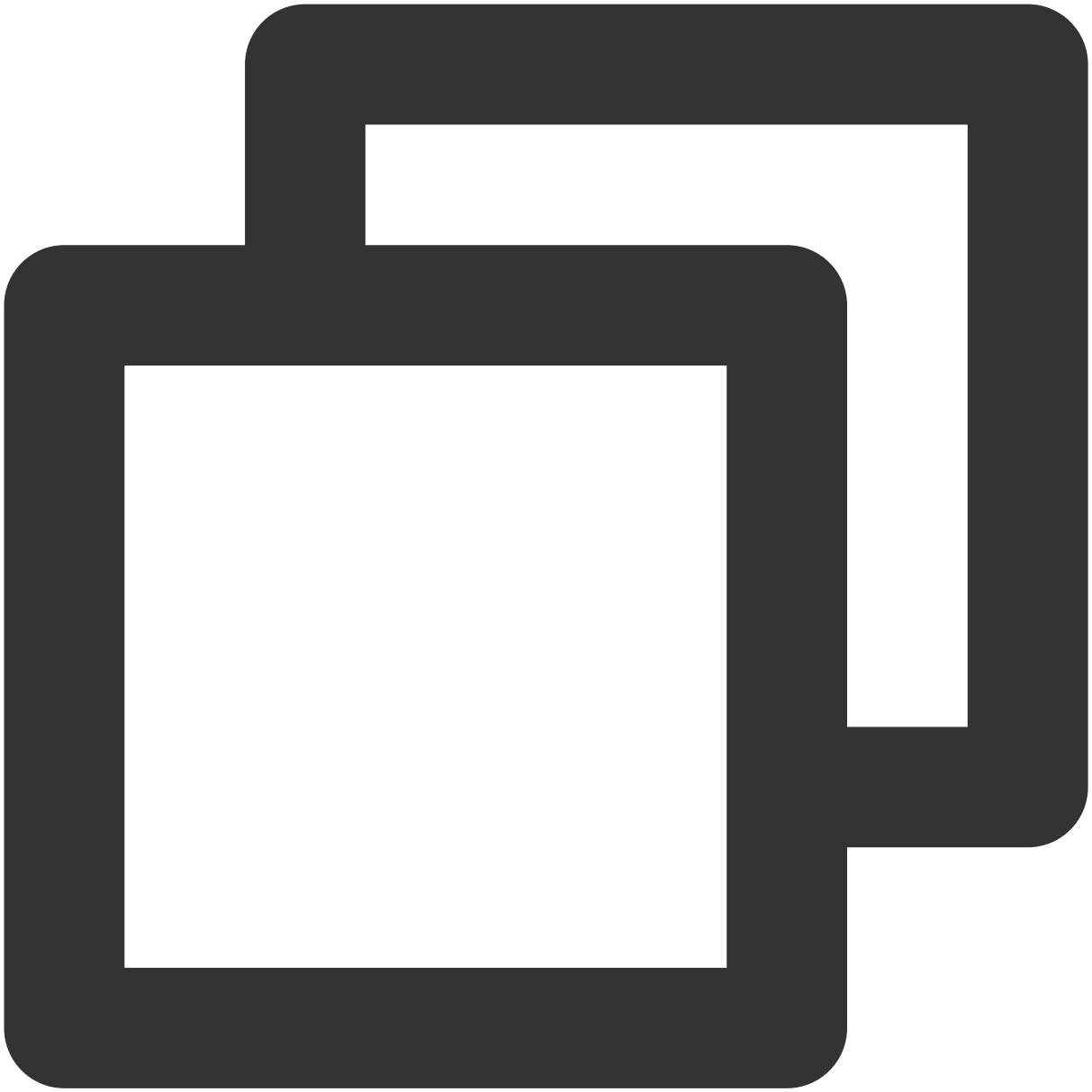
```
        //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
res.userId	String	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。



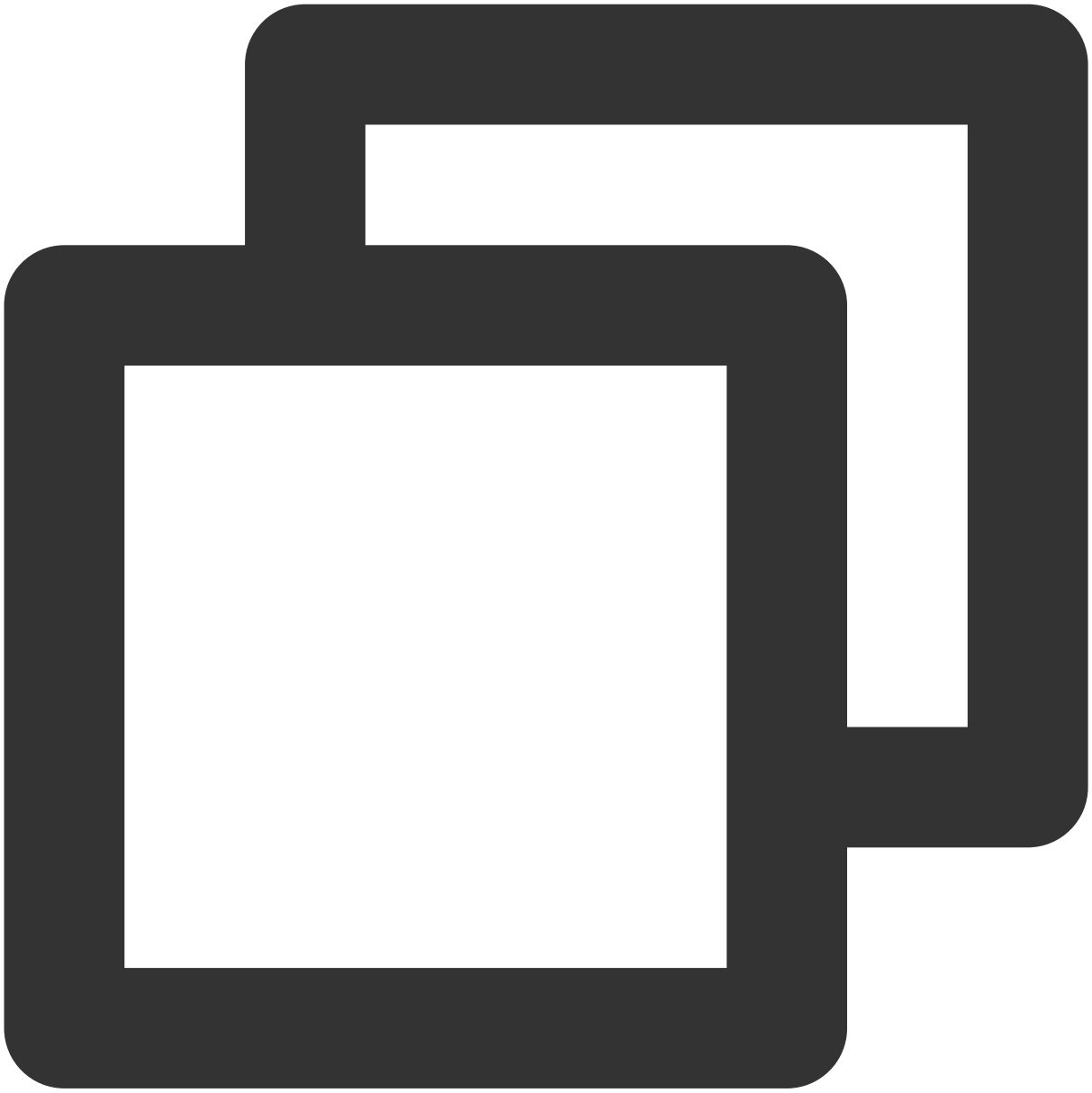
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNoResponse: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	无响应用户的 ID

onUserLineBusy

通话忙线回调。



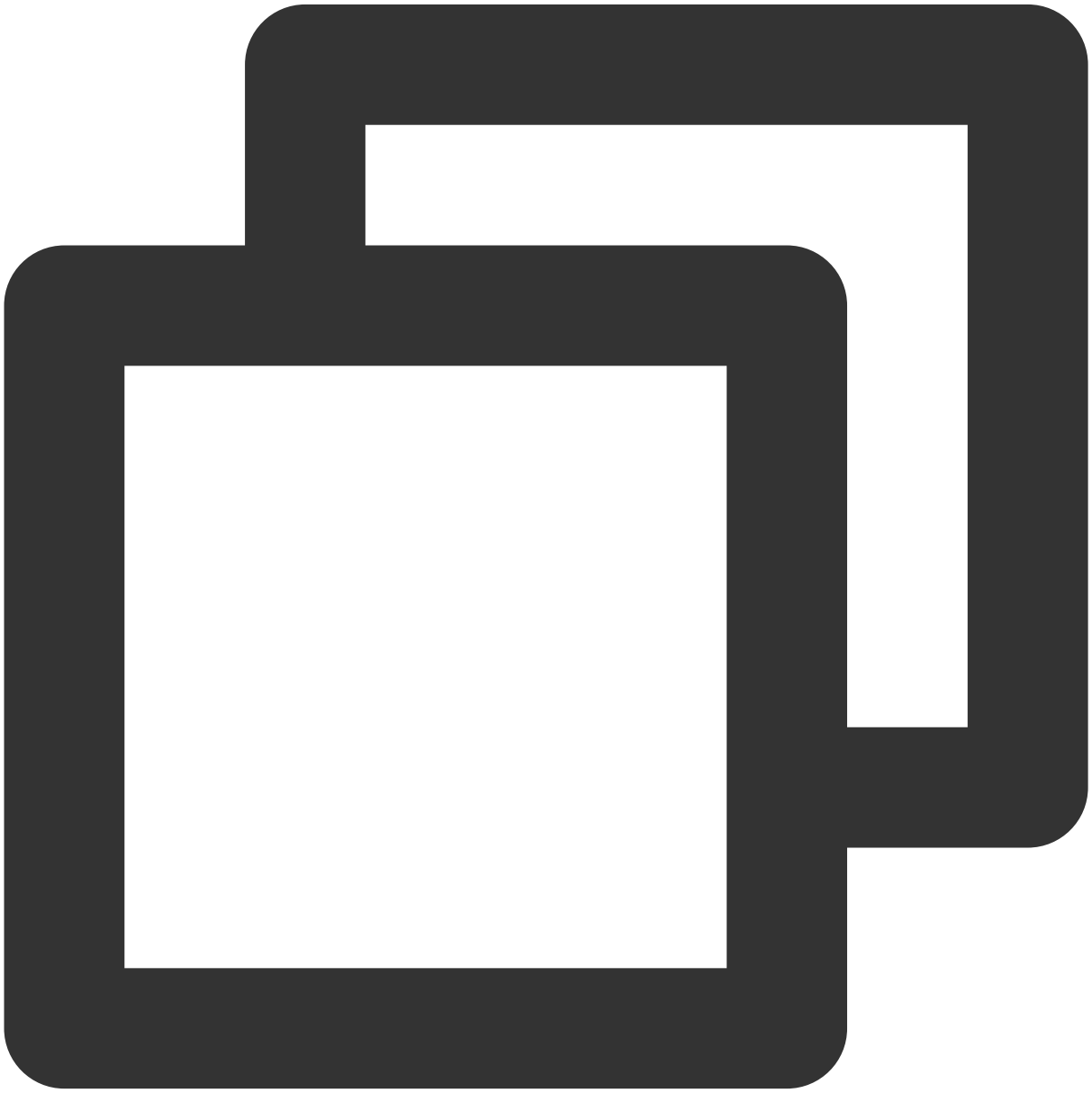
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLineBusy: (String onUserLineBusy) {
      //您的回调处理逻辑
    },
));
```

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID

onUserJoin

有用户进入此次通话的回调。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserJoin: (String userId) {
      //您的回调处理逻辑
```

```
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	加入当前通话的用户 ID

onUserLeave

有用户离开此次通话的回调。

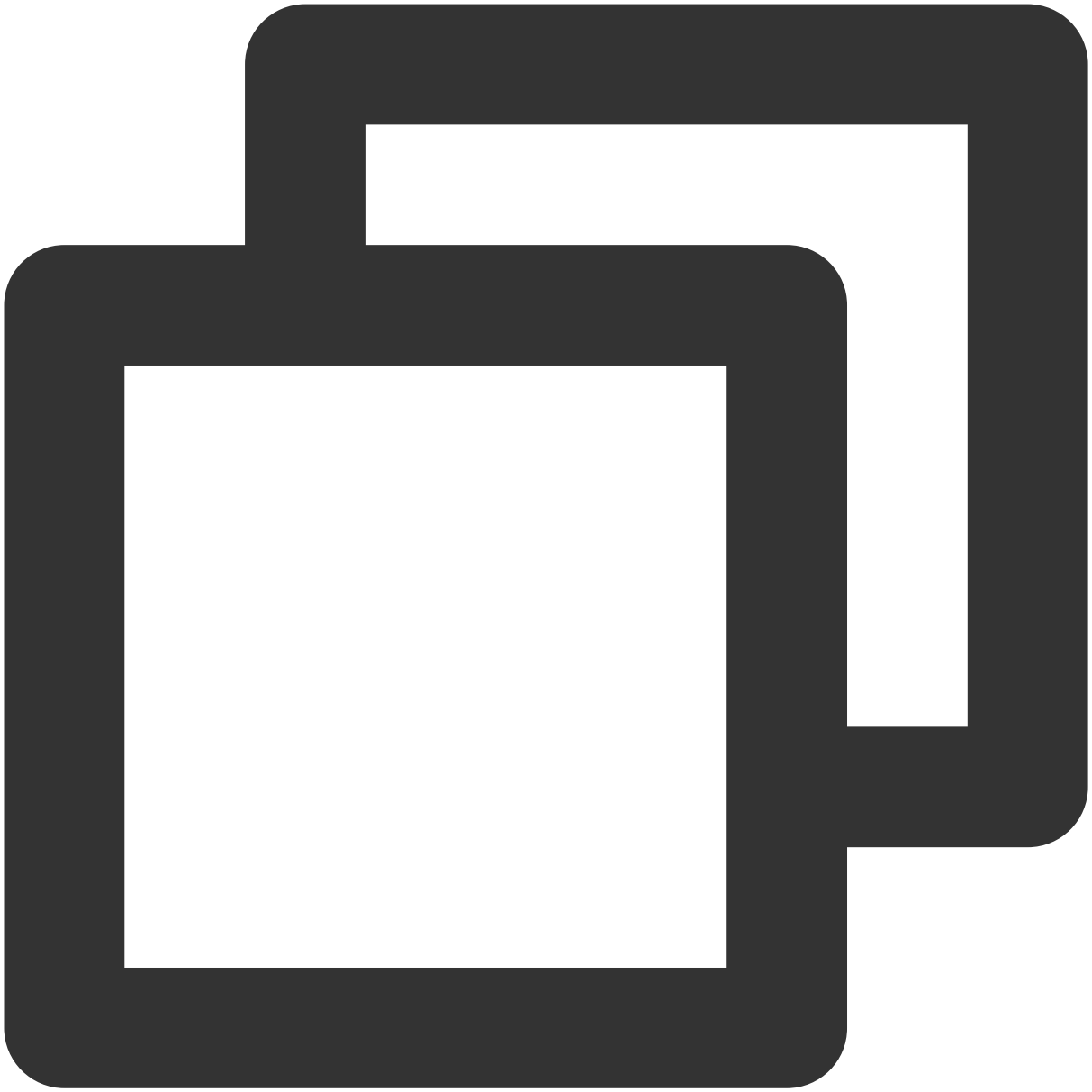
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLeave: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	离开当前通话的用户 ID

onUserVideoAvailable

用户是否开启视频上行回调。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVideoAvailable: (String userId, bool isVideoAvailable) {
      //您的回调处理逻辑
    }
));
```

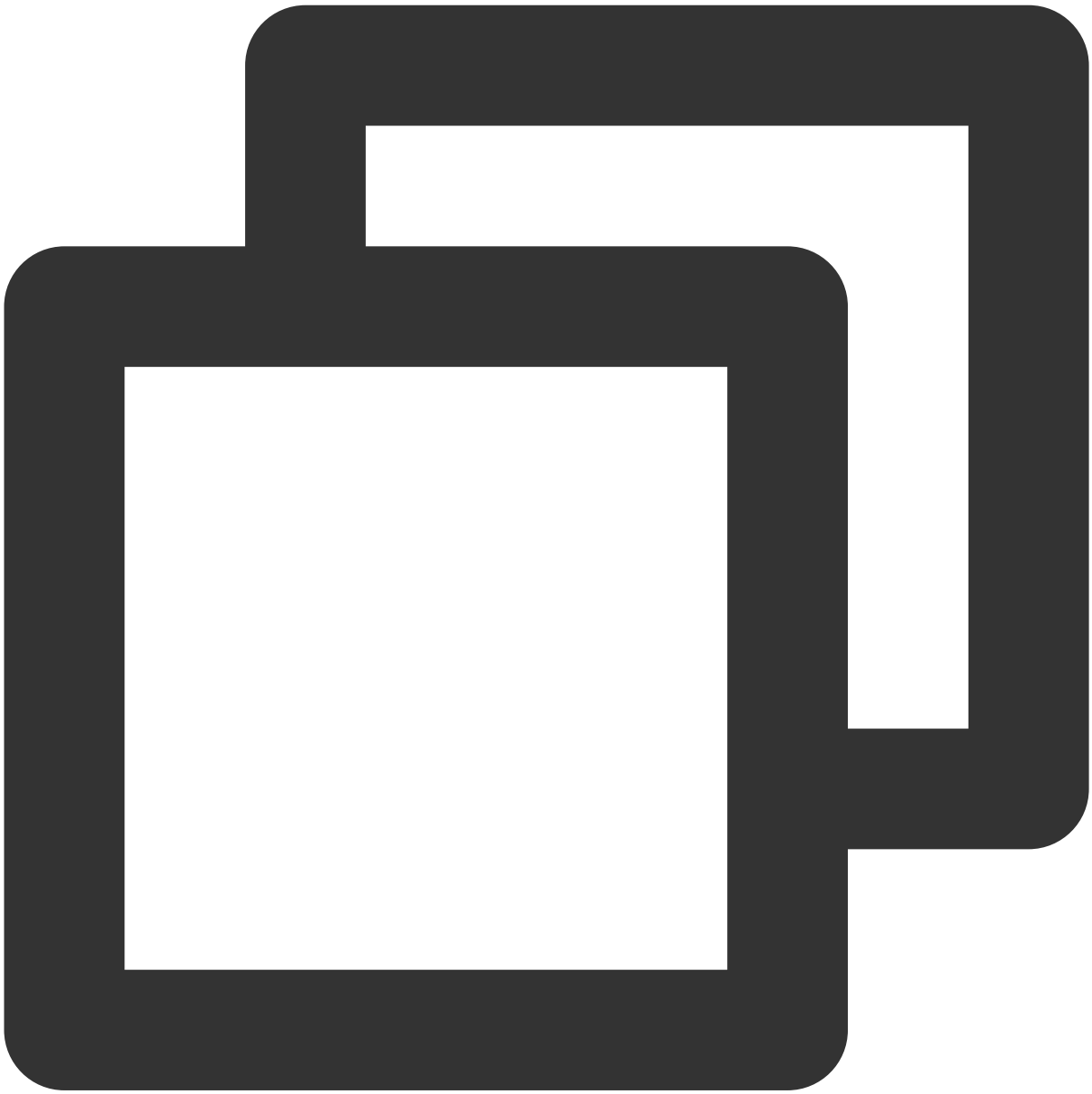
参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID

isVideoAvailable	bool	用户视频是否可用

onUserAudioAvailable

用户是否开启音频上行回调。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserAudioAvailable: (String userId, bool isAudioAvailable) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID
isAudioAvailable	bool	用户音频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

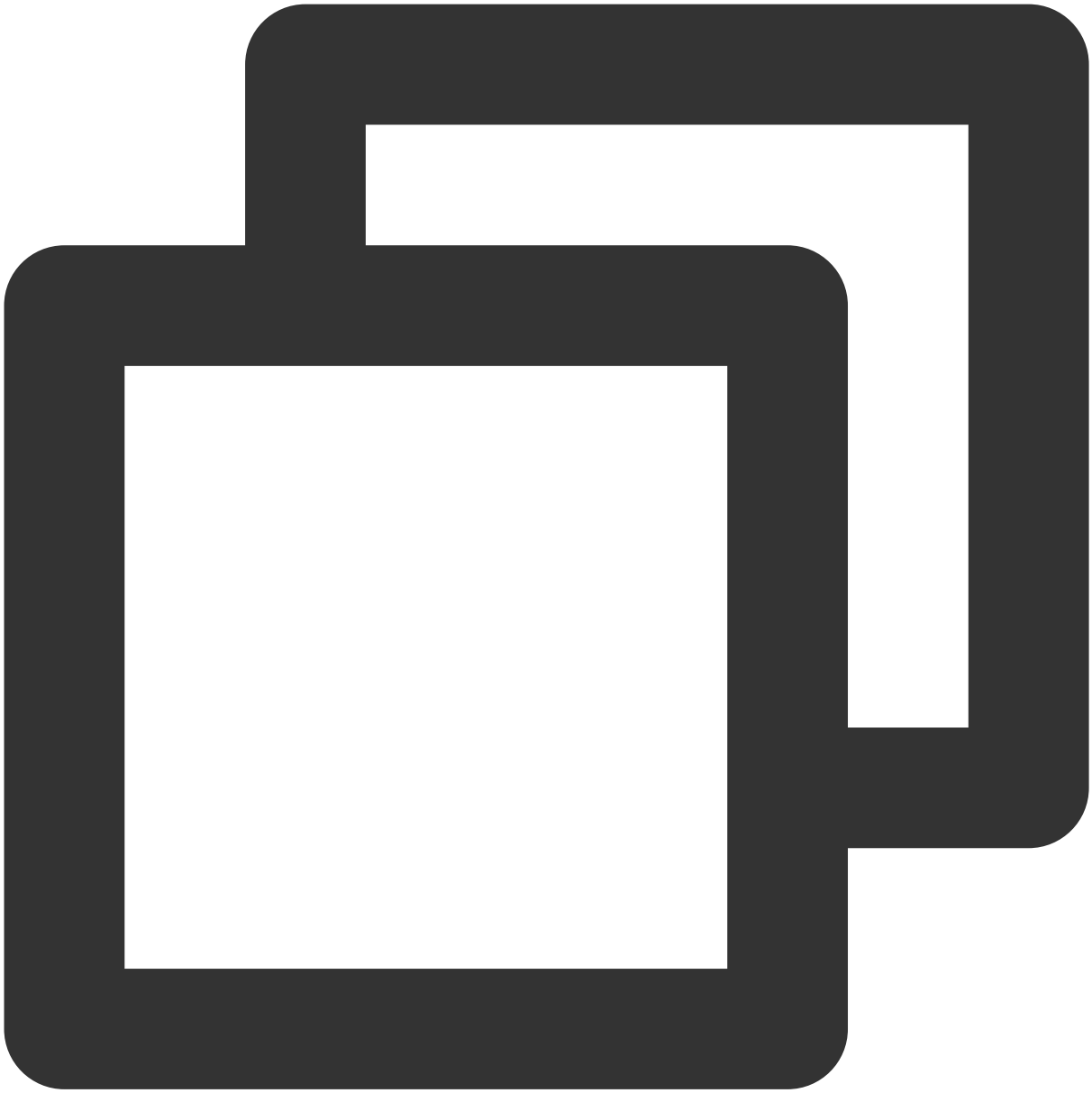
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
volumeMap	Map<String, int>	音量表，根据每个 userId 可以获取对应用户的音量大小，音量最小值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {
      //您的回调处理逻辑
    }
));

//TUINetworkQualityInfo的定义如下：
class TUINetworkQualityInfo {
    String userId;
    TUINetworkQuality quality;
    TUINetworkQualityInfo({required this.userId, required this.quality});
```

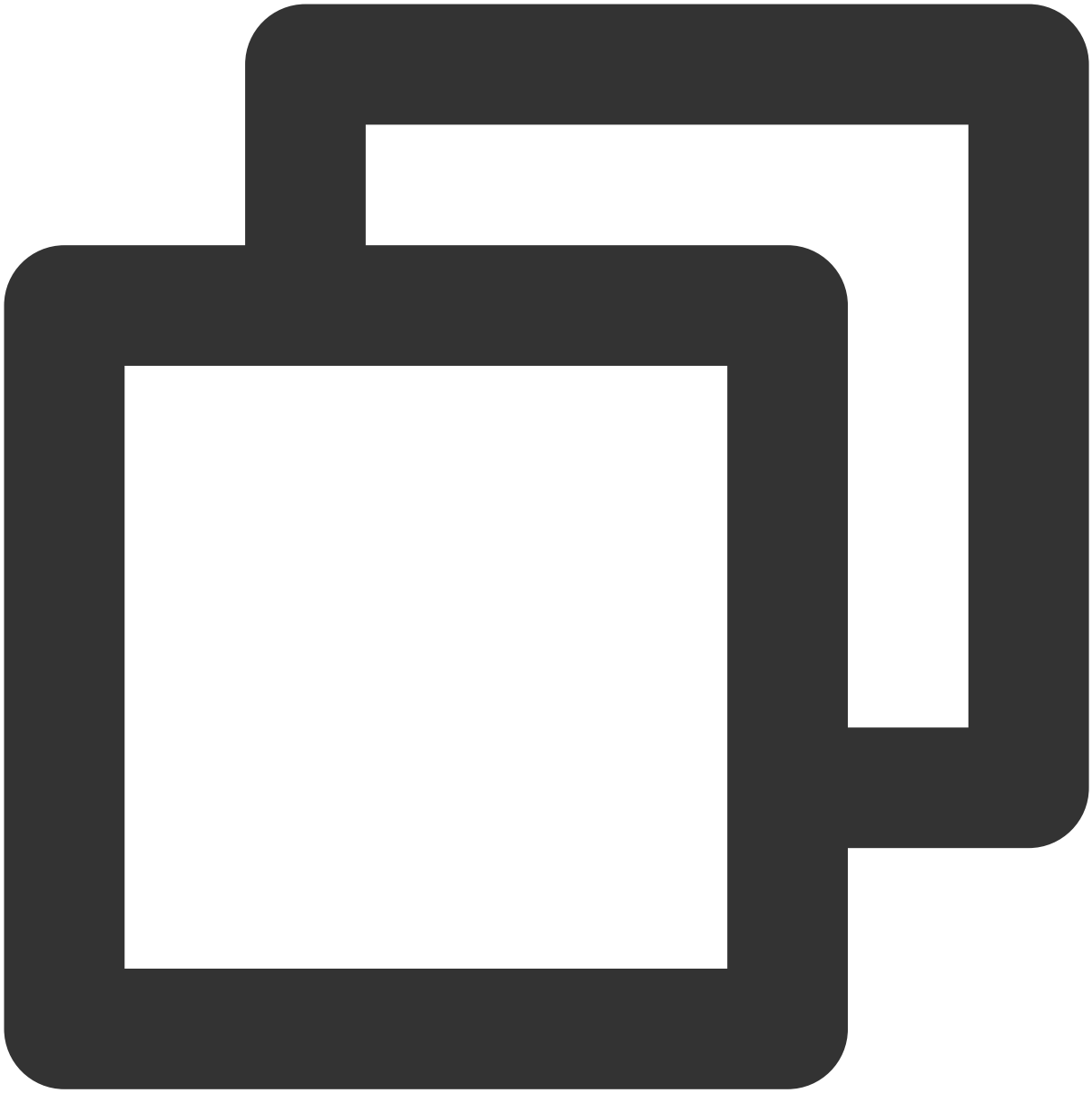
```
}
// TUINetworkQuality的定义如下：
enum TUINetworkQuality {
  unknown,
  excellent,
  good,
  poor,
  bad,
  vBad,
  down
}
```

参数如下表所示：

参数	类型	含义
networkQualityList	List<TUINetworkQualityInfo>	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

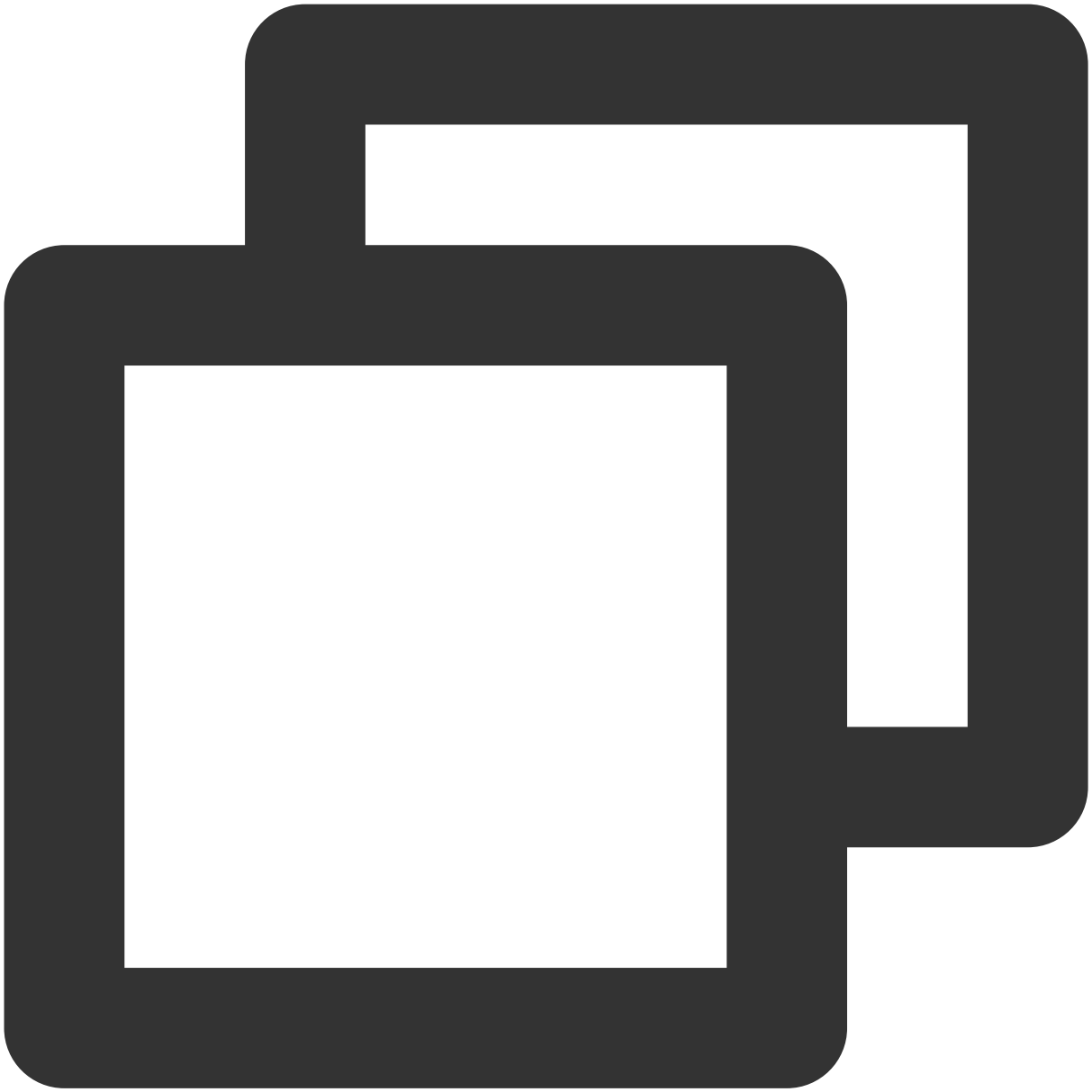
当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onKickedOffline: () {
      //您的回调处理逻辑
    }
));
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserSigExpired: () {
      //您的回调处理逻辑
    }
));
```

类型定义

最近更新时间：2023-08-22 10:13:23

常用结构

TUIResult

调用api的返回值

类型	类型	描述
code	String	code 为空 "" 表示调用成功,code 不为空 "" 表示调用失败
message	String?	失败原因

TUIRoomId

房间Id

类型	类型	描述
intRoomId	int	数字房间号
strRoomId	String	字符串房间号

VideoRenderParams

视频画面的渲染参数

类型	类型	描述
fillMode	FillMode	视频画面填充模式
rotation	Rotation	视频画面旋转方向

VideoEncoderParams

视频编码参数

类型	类型	描述
resolution	Resolution	视频分辨率
resolutionMode	ResolutionMode	视频宽高比模式

版权所有：腾讯云计算（北京）有限责任公司

TUICallParams

通话参数

类型	类型	描述
roomId	TUIRoomId	房间 ID
offlinePushInfo	TUIOfflinePushInfo	厂商离线推送配置信息
timeout	String	超时时常
userData	String	用户自定义数据

TUIOfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

类型	类型	描述
title	String	离线推送展示通知栏标题
desc	String	离线推送展示通知栏内容
ignoreIOSBadge	bool	离线推送忽略 badge 计数（仅对 iOS 生效），如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加。
iOSSound	String	离线推送声音设置（仅对 iOS 生效）。 当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。 当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。 如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀）设置给 iOSSound
androidSound	String	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。 只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声。 另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	int	VIVO 推送消息分类 (待废弃接口，VIVO 推送服

		务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	String	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	String	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	String	华为推送消息分类，详见：华为消息分类标准
isDisablePush	bool	是否关闭推送（默认开启推送）
iOSPushType	TUICallIOSOfflinePushType	iOS离线推送类型，默认：APNs

TUICallRecords

通话记录信息

类型	类型	描述
callId	String	通话记录ID
inviter	String	邀请者ID
inviteList	List<String>	被邀请用户ID列表
scene	TUICallScene	通话场景
mediaType	TUICallMediaType	通话媒体类型
groupId	String	群组ID
role	TUICallRole	通话角色
result	TUICallResultType	通话结果类型
beginTime	int	开始时间
totalTime	int	总时间

TUICallRecentCallsFilter

通话记录过滤条件

类型	类型	描述
begin	double	开始时间
end	double	结束时间

resultType	TUICallResultType	通话记录类型

枚举定义

TUICallMediaType

通话类型

类型	取值	描述
none	0	未知类型
audio	1	语音通话
video	2	视频通话

TUICallRole

通话角色

类型	取值	描述
none	0	未知类型
caller	1	主叫（邀请方）
called	2	被叫（被邀请方）

TUICallStatus

通话状态

类型	取值	描述
none	0	未知类型
waiting	1	通话等待中
accept	2	通话已接听

TUICallScene

通话场景

类型	取值	描述
groupCall	0	群组通话

版权所有：腾讯云计算（北京）有限责任公司

multiCall	1	匿名群组通话（暂不支持，敬请期待）
singleCall	2	单人通话

TUINetworkQuality

网络质量

类型	取值	描述
unknown	0	未定义
excellent	1	当前网络非常好
good	2	当前网络比较好
poor	3	当前网络一般
bad	4	当前网络较差
vBad	5	当前网络很差
down	6	当前网络不满足通话的最低要求

FillMode

视频画面填充模式

类型	取值	描述
fill	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完整。
fit	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边。

Rotation

视频画面旋转方向

类型	取值	描述
rotation_0	0	不旋转
rotation_90	1	顺时针旋转90度
rotation_180	2	顺时针旋转180度

版权所有：腾讯云计算（北京）有限责任公司

rotation_270	3	顺时针旋转270度

ResolutionMode

视频宽高比模式

类型	取值	描述
landscape	0	横屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360。
portrait	1	竖屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640。

Resolution

视频分辨率

类型	取值	描述
resolution_640_360	0	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps
resolution_640_480	1	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
resolution_960_540	2	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
resolution_960_720	3	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
resolution_1280_720	4	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
resolution_1920_1080	5	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

TUICallIOSOfflinePushType

iOS 离线推送类型

类型	取值	描述
APNs	0	普通推送
VoIP	1	VoIP推送

版权所有：腾讯云计算（北京）有限责任公司

TUICamera

摄像头类型类型

类型	取值	描述
front	0	前置摄像头
back	1	后置摄像头

TUIAudioPlaybackDevice

音频播放路由

类型	取值	描述
speakerphone	0	扬声器
earpiece	1	耳麦

TUICallResultType

通话记录类型

类型	取值	描述
unknown	0	未知
missed	1	未接
incoming	2	接入
outgoing	3	拨出

版权所有：腾讯云计算（北京）有限责任公司

错误码表

最近更新时间：2023-06-30 16:29:46

通知用户在使用音视频通话过程中出现的警告和错误。

TUICallDefine 错误码

定义	取值	描述
ERROR_PACKAGE_NOT_PURCHASED	-1001	您当前未购买音视频通话能力套餐，请前往IM控制台 开通免费体验 或 加购正式版
ERROR_PACKAGE_NOT_SUPPORTED	-1002	您当前购买的音视频通话能力套餐不支持该功能，建议您前往IM控制台 升级套餐类型
ERROR_TIM_VERSION_OUTDATED	-1003	IM SDK 版本过低，请升级 IM SDK 版本 >= 6.6；在build.gradle文件中查找并修改："com.tencent.imsdk:imsdk-plus:7.1.3925"
ERROR_PERMISSION_DENIED	-1101	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
ERROR_GET_DEVICE_LIST_FAIL	-1102	获取设备列表失败（仅 web 平台支持）
ERROR_INIT_FAIL	-1201	未调用 init 方法进行初始化，TUICallEngine API 需在初始化之后使用
ERROR_PARAM_INVALID	-1202	参数错误
ERROR_REQUEST_REFUSED	-1203	当前状态不支持调用
ERROR_REQUEST_REPEATED	-1204	当前方法正在执行中，请勿重复调用
ERROR_SCENE_NOT_SUPPORTED	-1205	当前通话场景，不支持该功能
ERROR_SIGNALING_SEND_FAIL	-1406	信令发送失败，具体失败原因可查看具体方法回调中的 errorMsg

IM 错误码

版权所有：腾讯云计算（北京）有限责任公司

音视频通话使用了腾讯云 IM SDK 作为通信的基础服务，比如通话拨打信令、通话忙线信令等核心逻辑。常见错误码如下：

错误码	描述
6014	IM SDK 未登录，请先登录，成功回调之后重试，或者已被踢下线，可使用 TIMManager getLoginUser 检查当前是否在线。
6017	参数无效，请检查参数是否符合要求，具体可查看错误信息进一步定义哪个字段。
6206	UserSig 过期，请重新获取有效的 UserSig 后再重新登录，获取方法请参见 生成 UserSig。
7013	套餐包不支持该接口的使用，请升级到旗舰版套餐。
8010	信令请求 ID 无效或已经被处理过。

说明：

更多 IM SDK 错误码详见：IM错误码

TRTC 错误码

音视频通话使用了腾讯云 TRTC SDK 作为音频和视频的基础服务，比如开关摄像头、开关麦克风等核心逻辑。常见错误码如下：

枚举	取值	描述
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了

ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_TRTC_ENTER_ROOM_FAILED	-3301	进入房间失败，请查看 onError 中的 -3301 对应的 msg 提示确认失败原因

说明：

更多 TRTC SDK 错误码详见：TRTC错误码

发布日志（TUICallKit）Web(Vue)

最近更新时间：2024-06-12 10:17:44

说明：

TUICallKit Vue3 Github 版本地址：Github TUICallKit Web。

Version 3.2.9 @2024.05.29

优化

UI 定制化接口增加日志上报

Version 3.2.8 @2024.05.27

修复

修复 SDK 引入 ref 路径错误问题

Version 3.2.7 @2024.05.17

新增

新增中途加入 UI 组件。
新增自定义 UI 接口，支持设置通话背景和按钮隐藏。
调整发起通话时的参数校验，支持字符串房间号。

Version 3.2.4 @2024.05.06

新增

视频通话支持背景虚化。

修复

修复 uni-app 打包 web 项目，image 加载 icon 失败问题。
修复群通话切换摄像头按钮问题。

版权所有：腾讯云计算（北京）有限责任公司

优化按钮点击后，应用卡住，再次点击按钮引起的异常问题。

Version 3.2.3 @2024.04.19

新增

新增群通话支持前后置摄像头切换。

优化

优化 TUICallKit SDK Readme。

Version 3.2.2 @2024.03.25

新增

新增全新的 UI 视效，功能更清晰，体验更好。

优化

优化使用 TUICallKit 的数据上报。

新增

language 日志上报。

Version 3.2.1 @2024.03.08

新增

language 日志上报。

Version 3.2.0 @2024.02.23

新增

增加默认的离线推送参数。

修复

修复群组通话没有昵称问题。

Version 3.1.9 @2024.01.30

修复

修复群组通话没有展示用户信息问题。
修复选人组件在没有可选成员时，“确认”按钮仍然可点击的问题。
修复通话中关闭麦克风，再次通话不上行音频流问题（升级 trtc-cloud-js-sdk 到 v2.2.7+）。

Version 3.1.8 @2024.01.19

修复

修复选人组件样式对页面的影响。

Version 3.1.7 @2024.01.12

修复

增加接口的重试机制，修复播放时因找不到 dom 节点导致播放失败问题。
修复 PC 上设备列表选择样式问题。

Version 3.1.6 @2023.12.29

优化

优化群组通话下的提示信息。
优化昵称过长时的显示问题。

修复

修复语音通话请求摄像头权限问题。
修复 destroyed 存在的问题。
修复不同呼叫场景下悬浮窗里的挂断问题。
修复主叫呼叫状态下显示远端问题。
修复 pc 上填充不全的样式问题。

Version 3.1.5 @2023.12.15

新增

优化获取设备权限的时机，初始化时不再获取设备权限，使用 call 时才获取。

修复

修复 @tencentcloud/call-uikit-vue2、@tencentcloud/call-uikit-vue2.6 组件没有声明文件问题。

Version 3.1.4 @2023.12.01

新增

融合到 Chat 中增加 isFromChat 上报。

修复

修复按钮 loading 下可点击的问题。

Version 3.1.3 @2023.11.17

新增

接口增加参数校验。

Version 3.1.2 @2023.11.03

新增

新增邀请他人 inviteUser 功能。
新增中途加人 joinInGroupCall 功能。
新增被叫来电铃声静音接口 enableMuteMode功能。

修复

修复远端流麦克风状态展示不对问题。

Version 3.1.0 @2023.10.20

新增

增加悬浮窗功能。
增加 enableFloatWindow 接口，用于开启/关闭悬浮窗功能。
桌面端支持摄像头、麦克风切换设备。

增加呼叫黑名单用户失败提示信息。
增加对日语的支持。

优化

视频通话下，大屏默认显示远端用户。

Version 3.0.8 @2023.10.10

新增

增加版本号、框架等信息上报。

Version 3.0.7 @2023.10.08

新增

增加桌面端视频通话下，通话时长展示。

优化

优化桌面端上视频流预览圆角黑边问题。
优化远端流用户信息显示优先级：备注 > 昵称 > userId。
优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 3.0.6 @2023.09.19

修复

修复融合到 TUIKit 中的消息上屏问题。

Version 3.0.5 @2023.09.15

优化

优化 TUICallKit 之间的相互引用，避免在 uniapp 打包小程序出现的栈溢出问题。

新增

新增桌面端设备无权限时的提示，引导客户如何对设备进行授权。

修复

修复 setCallingBell 被叫铃声被主叫铃声覆盖，导致铃声重复问题。
修复移动端上的样式问题。

Version 3.0.4 @2023.09.01

修复

修复 setCallingBell 作用对象为来电铃声（被叫铃声）。
修复 destroyed 报错问题。
修复错误弹框提示没有中英文。
修复 1v1 通话关闭摄像头后, 无法切换大小屏。

Version 3.0.3 @2023.8.25

新增

新增 @tencentcloud/call-uikit-vue2.6 适配 Vue2.6 版本。

优化

优化组件默认语言，调整为系统默认的语言。
优化打印的日志信息。
优化 tuicall-engine-webrtc 抛出错误的信息提示。
优化组件销毁后，对所用资源的清理。

修复

修复 videoDisplayMode、videoResolution 挂断后再次呼叫时，未生效问题。
修复通话中，statusChanged 未抛出问题。
修复 init 多次调用问题。
修复通话中，关闭摄像头无法切换大小屏问题。

Version 3.0.2 @2023.8.14

修复

修复 H5 端被叫组件的样式问题。
修复切换小窗后，再次通话出现的样式问题。

Version 3.0.1 @2023.8.8

修复

修复群通话主叫方本地预览失败的问题，修改组件层对数据层默认读取方式。

Version 3.0.0 @2023.8.4

Breaking Change

升级底层依赖 tuicall-engine-webrtc 至 ^2.0.0，不再支持 tim-js-sdk 创建 tim 实例，如需创建 tim 实例，请使用 @tencentcloud/chat 创建。

新增

新增自定义铃声功能 setCallingBell。

Version 2.4.2 @2023.11.03

新增

新增邀请他人 inviteUser 功能。
新增中途加人 joinInGroupCall 功能。
新增被叫来电铃声静音接口 enableMuteMode功能。

修复

修复远端流麦克风状态展示不对问题。

Version 2.4.0 @2023.10.20

新增

增加悬浮窗功能。
增加 enableFloatWindow 接口，用于开启/关闭悬浮窗功能。
桌面端支持摄像头、麦克风切换设备。
增加呼叫黑名单用户失败提示信息。
增加对日语的支持。

优化

视频通话下，大屏默认显示远端用户。

Version 2.3.9 @2023.10.10

新增

增加版本号、框架等信息上报。

Version 2.3.8 @2023.10.08

新增

增加桌面端视频通话下，通话时长展示。

优化

优化桌面端上视频流预览圆角黑边问题。
优化远端流用户信息显示优先级：备注 > 昵称 > userId。
优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 2.3.6 @2023.09.15

优化

优化 TUICallKit 之间的相互引用，避免在 uniapp 打包小程序出现的栈溢出问题。

新增

新增桌面端设备无权限时的提示，引导客户如何对设备进行授权。

修复

修复 setCallingBell 被叫铃声被主叫铃声覆盖，导致铃声重复问题。
修复移动端上的样式问题。

Version 2.3.5 @2023.9.5

修复

修复进房前摄像头、麦克风按钮默认打开的问题。

Version 2.3.4 @2023.9.1

新增

新增视频通话接听前可以关闭或者开启摄像头功能。

修复

修复 1v1 通话关闭摄像头后, 无法切换大小屏问题。
修复从视频通话切换到语音通话 statusChanged 未抛出问题。

Version 2.3.3 @2023.8.22

修复

修复 videoDisplayMode、videoResolution 挂断后再次呼叫时，未生效问题。
修复通话中，statusChanged 未抛出问题。

Version 2.3.2 @2023.7.26

Breaking Change

移除悬浮窗 TUICallKitMini 组件，合并到 TUICallKit 组件中。
抛出的 @kicked-out 事件调整为绑定回调 `:kickedOut` 的方式。
抛出的 @status-changed 事件调整为绑定回调 `:statusChanged` 的方式。

新增

新增通话页面出现时的动画效果。
新增 H5 上群组通话布局。

优化

优化通话中的问题提示信息、提示方式。
优化在 H5 页面的支持、交互。
优化通话界面唤起的时间。
优化 @tencentcloud/call-uikit-vue 包目录结构。

修复

修复接通后立刻挂断等边界操作下的通话问题。
修复 H5 在部分机型、浏览器上的样式问题。

修复重复点击引起的通话异常问题。

Version 2.2.1 @2023.7.7

新增

@tencentcloud/call-uikit-vue2 新增 Vue 版本的检测及提示。

修复

修复来电页面重复点击“接听”按钮导致接听失败。

Version 2.2.0 @2023.6.30

新增

call、groupCall 支持自定义数字房间号 roomID 参数。
call、groupCall 支持自定义扩展字段 userData 参数（用于在邀请信令中增加扩展信息）。
新增 setSelfInfo 接口，支持用户设置昵称和头像。

Version 2.1.0 @2023.4.14

新增

在 H5 模式语音通话下，呼叫时支持显示对方昵称。
发起通话失败时，会在呼叫页面显示“发起通话失败”。
接听通话失败时，会在来电页面显示“接听失败”。
支持监听当前用户是否被踢出（如被挤出登录），详见 TUICallKit 方法 - @kicked-out。
支持监听 TUICallKit 通话状态，详见 TUICallKit 方法 - @status-changed。
支持业务侧代码控制通话的接听、取消和挂断，详见 更多特性 - 通过接口设置自动接听。
Vue2 版本新增 TypeScript 类型声明文件，在 TypeScript 项目中可以正常编译类型。

修复

修复在初始化组件时，控制台出现的更新个人资料接口的警告。
修复在 H5 模式下，被叫接听按钮出现背景图片错位的问题。

接口变更

`TUICallKitServer.destroy()` 增加调用限制，仅支持在未通话状态时调用。

Version 2.0.1 @2023.03.31

新增

优化 1v1 通话与多人通话视频的渲染逻辑，提升性能与稳定性。
优化 UI 展示，支持在 `TUICallKitServer.call()` 执行过程中显示对应 UI，即可以实现点击通话按钮立刻展示 `<TUICallKit/>` UI 组件。

修复

修复多人通话中昵称显示错误的问题。
修复 CSS 未限制生效范围导致污染全局样式的问题。

Version 2.0.0 @2023.03.21

新增

支持从 npm 引入已打包的 CallKit 文件。
支持 JavaScript 版本的 Vue 项目。
支持 Vue2 所有版本的 Vue 项目，适用于 Vue2 的 npm 包：call-uikit-vue2。

修复

修复无摄像头设备或权限导致通话发起异常的问题。

Version 1.4.2 @2023.03.03

新增

支持设置通话分辨率，详见 API 文档。
支持修改画面显示模式，详见 API 文档。
优化了接入步骤。
优化了错误抛出。

Version 1.4.1 @2023.02.13

新增

优化了本地摄像头的预览逻辑。
优化了远端视频流的渲染逻辑。

Version 1.4.0 @2023.01.06

新增

支持 Vue2.7+ 项目引入。
通话界面默认显示昵称，设置昵称请参考 TIM#updateMyProfile。

Version 1.3.3 @2022.12.27

新增

在 Basic Demo 中拨打电话时，新增了通话列表的空值检测。
在拨打电话时，Basic demo 中增加了 loading 图标。
优化了 Basic Demo 中设备检测的逻辑，手动跳过后不再主动弹出。
优化了组件图标的引用方式。
修改了默认的包管理工具为 npm。
优化了视频的渲染方式，减少了重复渲染的次数。

修复

修复了由于 vue-CLI 过时依赖导致的 Basic Demo 报错。

Version 1.3.2 @2022.12.07

新增

支持语言切换，接口详情可见 setLanguage。
优化 Basic Demo 设备检测逻辑，手动跳过后不会再主动弹出。

修复

修复引入 `defineProps` 导致的警告。

Version 1.3.1 @2022.11.29

注意：

本版本依赖 SDK tuicall-engine-webrtc@1.2.1 版本，请及时更新。

新增

优化样式细节。

支持未接听对方通话时，监听对方修改通话类型。
basic demo 增加设备检测功能。

修复

修复在挂断电话时内部逻辑导致的报错。

Version 1.3.0 @2022.11.14

新增

支持手机 H5 使用时，自动切换为竖屏样式。
支持在拨打电话时预览本地摄像头。
basic demo 增加拨打电话前的设备检测。

修复

修复调用 TUICallKitServer.destroyed() 之后 tim 实例未完全退出的问题。
修复对方忙线时显示无应答的问题。
修复 vite 环境下 typescript 类型未成功打包的问题。

接口变更

主动调用 TUICallKitServer.call() 或 TUICallKitServer.groupCall() 时，如果遇到报错，不会再调用 beforeCalling 回调。请直接使用 try catch 捕获错误。

Version 1.2.0 @2022.11.03

新增

适配新版本 TUICallEngine SDK。

Version 1.1.0 @2022.10.21

新增

通话过程中，通话页面可以全屏。
通话过程中，可以使用 `<TUICallKitMini/>` 进行最小化。

修复

修复已知问题，提升稳定性。

Version 1.0.3 @2022.10.14

新增

basic demo 增加快捷复制 UserID、一键打开新窗口。

Version 1.0.2 @2022.09.30

新增

优化接入文档，增加演示图片和详细指引。

修复

修复首次进入房间时，设备状态位失效的问题。
修复在 webpack 工具打包时，图标偶现加载失败的问题。
修复已知样式问题。

Version 1.0.1 @2022.09.26

新增

在拨打电话时，隐藏对方麦克风图标。

修复

修复 basic demo SDKAppID 输入框应为数字的问题。

Version 1.0.0 @2022.09.23

TUICallKit demo 快速跑通
TUICallKit 快速接入
TUICallKit API
TUICallKit 界面定制指引
TUICallKit (Web) 常见问题

Android & iOS

最近更新时间：2024-05-14 15:31:17

新版本（1.1.0.103）涉及 TUICallEngine 接口变更：

如果您在开发过程中，因为 maven latest 更新机制导致的 Android 构建报错，可以使用如下两种方式解决：

升级 TUICallKit 到最新版本。

修改 tuicallkit/build.gradle 的 tuicallengine 依赖为 1.0 的固定版本：`com.tencent.liteav.tuikit:tuicallengine:1.0.0.53`。

如果您在开发过程中，因为 执行 pod update 导致 iOS 构建报错，可以使用如下两种方式解决：

升级 TUICallKit 到最新版本。

Podfile 文件中添加 `pod 'TUICallEngine', '1.0.0.53'`。

Version 2.3.0.915 @ 2024.4.15

功能优化

Android&iOS：支持在 TUIChat 的群组顶部显示通话状态，并允许群成员主动加入通话。

Android&iOS：优化来电弹出逻辑，默认展示横幅接听框。

Android&iOS：视频通话支持背景虚化。

缺陷修复

Android：修复通话记录编辑界面中，点击删除按钮无响应的问题。

iOS：修复在群通话中点击成员视图时，切换过程中出现重影的问题 。

iOS：修复音视频界面特定场景下不显示的问题。

Android&iOS：修复在呼叫忙线中的用户时，通话结束后缺少提示的问题。

Version 2.2.0.860 @ 2024.2.1

功能优化

Android&iOS：全新的UI视效，功能更清晰，体验更好。

缺陷修复

Android&iOS：修复在会议、直播场景下，接听来电后，麦克风和摄像头设备互相占用问题。

Version 2.1.0.810 @ 2023.12.19

功能优化

Android：优化未登录时调用 TUICallKit 接口异常的提示。
Android： 优化针对 Android 14平台(API 34) 的兼容性，详见：Android 14 行为变更。
Android&iOS：优化用户昵称的显示，按照如下顺序显示：用户备注 > 用户昵称 > 用户 Id，默认为 userId。

缺陷修复

iOS：修复群通话头像显示重叠问题。
iOS：修复视频通话时开启悬浮窗发消息再回到通话界面键盘无法收起问题。
iOS：修复视频通话中，先切换摄像头、关闭摄像头，再切换回原摄像头并重新打开时，无法切换和移动摄像头的问题。
Android：修复在阿拉伯语等从右至左布局模式(RTL 模式)下单人视频通话的小窗口无法移动的问题。

Version 2.0.0.750 @ 2023.11.03

功能优化

Android&iOS：UIKit 支持日语。
Android&iOS：优化通话昵称显示。
Android&iOS：调整铃声的默认音量，由60%调整到100%。

缺陷修复

iOS：修复Swift 版本加载图片过慢问题。
iOS：修复主叫端发起通话后退后台，通话邀请自动取消问题。

Version 1.9.0.680 @ 2023.09.27

功能优化

Android&iOS：支持阿拉伯语言。
Android&iOS：优化套餐购买提示，支持根据链接跳转到对应的套餐购买页。
Android&iOS：优化不同分辨率下的默认码率，高分辨率下画面更清晰。
Android&iOS：调整视频通话的默认码率为600kbps，美颜等级为4级。

缺陷修复

Android&iOS：对黑名单用户发起通话，通话拒绝提示与发送单聊消息拒绝提示不一致。
iOS：4人群视频通话界面，一名成员拒绝接听，发起人视频位置画面异常。
iOS：修复登录成功后立即设置美颜，分辨率会被重置问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.8.0.620 @ 2023.08.14

功能优化

Android&iOS：通话消息默认不计入未读数。
Android：优化小米手机悬浮窗权限跳转页面。

缺陷修复

iOS：修复被移下线后回调接口 onKickOffline 无效问题。
iOS： 修复通话记录未接界面清除通话后，回到所有通话界面，列表为空问题。

Version 1.7.2.570 @ 2023.07.20

功能优化

Android：默认关闭重力感应，优化在大屏、定制设备的通话体验。

缺陷修复

Android & iOS：A 呼叫 B(离线)后取消，A 再次呼叫 B，B 登录上线，B 的云端通话记录异常问题。
Android： 升级 TRTC SDK 版本到11.3后，TUICallKit 的崩溃问题。

Version 1.7.0.460 @ 2023.06.25

功能优化

Android&iOS：含UI 集成方案，优化示例工程，完善通话设置项。
Android：降低通话时系统保活的级别，只在状态栏显示保活提示，去掉通知和震动。

Version 1.6.1.410 @ 2023.05.22

新特性

Android&iOS：含UI接口 call 、groupCall 支持自定义房间号；
Android&iOS：支持发起通话时传入字符串型房间号，详见 CallParams；

缺陷修复

Android：修复通过 Arrays.asList 方式生成列表参数时，groupCall 报错问题；
Android：修复视频通话显示异常问题；

iOS：修复与 TUIRoom 组件冲突问题；
iOS：修复登录成功回调中，立刻发起通话崩溃问题；
iOS：修复点击通知消息进入 App，偶现不显示邀请页面问题；

Version 1.6.0.360 @ 2023.04.27

新特性

Android：TUICallKit 新增 Kotlin 语言版本；
iOS：TUICallKit 新增 Swift 语言版本；
Android&iOS：新增通话记录（本地）的显示页面。

功能优化

Android：优化单人视频通话头像显示；
Android&iOS：群组通话中，默认支持邀请其他群成员加入通话。

缺陷修复

Android：修复 Android 12及其以上 设备连接蓝牙后无声的问题；
Android：修复被叫端偶现静音设置不生效问题；
iOS：修复重新登录后偶现设备接收不到来电邀请的问题；
IOS：修复 TUICallKit 中 enableCustomViewRoute 接口失效问题；
IOS：修复 VoIP 推送页面昵称显示错误的问题。

Version 1.5.1.310 @ 2023.04.17

新特性

Android&iOS：新增 VoIP 消息推送功能，提供更好的通话接听体验。
Android&iOS：支持发起通话时自定义扩展字段，详见 call 方法的 TUICallDefine.CallParams 参数。

功能优化

Android&iOS：优化华为、小米、FCM 等厂商的离线推送能力，增加厂商消息分类及 channelID 设置。

缺陷修复

Android&iOS：修复 IM 自定义属性发起通话后被覆盖的问题。
Android：修复回调 onCallEnd 中 totalTime 的时间单位错误的问题

Version 1.5.0.305 @ 2023.03.09

功能优化

Android&iOS：优化上屏消息显示。
Android：贴耳息屏功能默认关闭。
Android：升级 gradle 插件及版本。
Android：优化铃声播放类，支持循环播放。

缺陷修复

Android&iOS：修复被叫接听通话失败，没有 onCallCancel 回调问题。
Android：修复被叫接听通话失败，主叫异常问题。
Android：修复首次通话检查权限时，主叫端取消通话，被叫端再次拉起界面问题。
Android：修复回调上层的网络质量 userId 为空问题。

Version 1.4.0.255 @ 2023.01.06

新特性

Android&iOS：支持自定义来电超时时间，详见 call 方法 TUICallDefine.CallParams 参数。

缺陷修复

Android&iOS：修复主动加入房间（joinInGroupCall）时通话异常结束问题。
Android：修复在音视频通话接听中退后台，再回到前台时，通话状态异常问题。

Version 1.3.0.205 @ 2022.11.30

新特性

Android&iOS：新增美颜设置接口 setBeautyLevel，支持关闭默认美颜。

功能优化

iOS: 优化 TUICallKit 的 Framework 体积。

缺陷修复

Android&iOS：修复服务端解散房间或踢出用户时，通话界面不消失问题。
Android：修复 A 呼叫离线用户 B，然后取消；A 再次呼叫 B，B 上线后通话界面不显示问题。

Version 1.2.0.153 @ 2022.11.14

新特性

Android&iOS：支持视频自定义编码分辨率。
Android&iOS：支持设置视频的渲染参数：渲染方向、填充模式。
Android&iOS：支持第三方美颜的接入。
Android&iOS：TUICallKit 新增重载接口 call()、groupCall()，支持自定义离线消息（详情请参见 API文档）。

功能优化

Android&iOS：优化部分 TUICallObserver 回调异常问题。
Android&iOS：优化视频切语音功能，支持离线状态下切换。
Android&iOS：完善 TUICallKit 的错误码及错误提示。
iOS：规范 TUICallEngine、TUICallKit Swift API 名称。

缺陷修复

Android&iOS：修复重新登录账号后，仍会收到之前已拒绝的来电问题。
Android：修复单聊中收到群聊邀请，通话异常挂断问题。
Android：多场景退出直播间无法发起通话异常问题。
Android：修复 A 呼叫 B，同时 B 呼叫 C，C 概率性进入 A 的房间问题。

Version 1.1.0.103 @ 2022.09.30

Android&iOS：优化群组通话过程中，邀请群成员加入功能。
Android&iOS：优化通话流程，避免通话接听前产生录制、审核等费用。
Android&iOS：支持自定义音视频通话的离线推送消息。
Android&iOS：部分 **TUICallEngine** 接口参数变更，详见 call()、groupCall()、inviteUser()、以及回调接口 onCallReceived()。
Android&iOS：修复群组通话过程中，偶现的少量回调异常问题。
Android&iOS：修复重复登录以及UserSig过期造成的状态异常问题。
iOS：修复 TUCallKit Objective-C 和 Swift 混合编译时，`init` 接口报错的问题。
Android：修复 Kotlin 集成悬浮窗功能时编译报错问题。

Version 1.0.0.53 @ 2022.08.15

全新发布

Android&iOS：支持1v1 音视频通话、群组音视频通话。
Android&iOS：支持主流厂商的离线唤醒。
Android&iOS：支持自定义头像、自定义昵称。
Android&iOS：支持通话过程中开启悬浮窗。
Android&iOS：支持设置自定义铃音。
Android&iOS：支持多平台登录状态下的来电服务。

Flutter

最近更新时间：2024-04-30 16:11:23

Version 2.3.2 @2024.04.24

新特性

Android&iOS：优化来电弹出逻辑，默认展示横幅接听框。
Android&iOS：视频通话支持背景虚化。

Version 2.3.1 @2024.04.22

问题修复

修复在非 iOS&Android 平台上出现的 Method Channel 报错问题。

Version 2.3.0 @2024.04.18

新特性

Android&iOS：支持在 tencent_cloud_chat_uikit 的群组顶部显示通话状态，并允许群成员主动加入通话。

功能优化

Android&iOS：优化与 tencent_cloud_chat_uikit 同时使用时的登录方式。

依赖描述

升级 tencent_cloud_uikit_core 版本为1.6.0。
升级依赖客户端 TUICallEngine SDK 版本为 2.3.0.915.

Version 2.2.3 @2024.03.08

依赖描述

升级 tencent_cloud_uikit_core 版本为1.5.2。

Version 2.2.2 @2024.03.07

版权所有：腾讯云计算（北京）有限责任公司

新特性

Android&iOS：TUICallObserver 的 onCallReceived 回调新增用户自定义参数 userData。

Version 2.2.1 @2024.02.06

问题修复

Android：修复了杀死进程后，应用程序接收到了 FCM 推送，引起的应用页面异常问题。

Version 2.2.0 @2024.02.03

新特性

Android：支持 FCM 推送（使用 FCM 推送，需要使用 tencent_cloud_chat_push）。

Version 2.1.1 @2024.01.06

问题修复

Android&iOS：修复了一些 UI 3.0相关的 UI 问题。

Version 2.1.0 @2024.01.04

新特性

Android&iOS：升级为 UI 3.0。

Version 2.0.6 @2023.12.15

问题修复

iOS：修复了 iOS 离线推送信息中缺少某些字段的问题。

Version 2.0.5 @2023.12.10

问题修复

iOS：修复了 VoIP 推送期间进入通话页面时铃声异常以及远程流拉取异常的问题。
Android：修复了在未获得悬浮窗权限的情况下，后台时来电页面显示异常的问题。

Version 2.0.4 @2023.12.04

问题修复

Android&iOS：修复了点击来电通知后有时不显示来电页面的偶现问题。

Version 2.0.2 @2023.11.27

问题修复

Android：修复工程中有多个 Flutter Engine 实例时导致 Method Channel 通信异常的问题。

Version 2.0.1 @2023.11.15

问题修复

Android&iOS：修复了在使用其它 TRTC 组件时导致 TRTC Observer 不兼容的问题。
Android：修复了与 TUIRoom 结合使用时的不兼容问题。

Version 2.0.0 @2023.11.06

依赖描述

升级依赖客户端 SDK 版本：
Android&iOS TUICore：7.6.5011。
Android&iOS TUICallEngine：2.0.0.750。

Version 1.9.1 @2023.10.21

新特性

iOS：支持 Voip。

问题修复

iOS：修复后台接听电话时通话页面异常的问题。

Version 1.9.0 @2023.10.09

新特性

Android&iOS：添加设置铃声的接口。

功能优化

Android&iOS：优化购买包的提示。
Android&iOS：优化不同分辨率的默认比特率，参见。

问题修复

iOS：修复了同一个 Observer 对象可以被注册两次的问题。

依赖描述

升级依赖客户端 SDK 版本：
Android&iOS TUICore：7.5.4852。
Android&iOS TUICallEngine：1.9.0.680。

Version 1.8.3 @2023.08.25

问题修复

Android&iOS：修复了使用 tencent_cloud_chat_uikit 时没有通话消息显示的问题。
Android&iOS：修复了在单人通话过程中偶尔拉起群通话页面的问题。
Android&iOS：修复了在通话过程中偶尔拉起通话页面两次的问题。
Android&iOS：修复了通话时间显示异常的问题。

功能优化

Android：优化了在后台接收到通话时无法拉起界面的问题。

依赖描述

升级 tencent_cloud_uikit_core 到版本 1.1.1。

Version 1.8.2 @2023.08.19

问题修复

iOS：修复了某些编译器由于使用了已弃用的 Swift 接口而无法编译的问题。

Version 1.8.1 @2023.08.18

缺陷修复

Android&iOS：修复了在进行群通话语音通话时，另一方的视频流显示的问题。

Version 1.8.0 @ 2023.08.17

新特性

Android&iOS：基于 Dart 语言构建全新的 TUICallkit，可以更方便的定制自己的 UI 风格。
Android&iOS：TUICallEngine 新增诸如 hangup、accept、reject 等多个业务接口。

Version 1.7.4 @ 2023.07.20

功能优化

Android：默认关闭重力感应，优化在大屏、定制设备的通话体验。

缺陷修复

Android&iOS：A呼叫B(离线)后取消，A再次呼叫B，B登录上线，B的云端通话记录异常问题。

Version 1.7.3 @ 2023.07.19

功能优化

Android: 支持使用模拟器进行开发&调试。

依赖说明

升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional: 11.3.0.13176。

Version 1.7.2 @ 2023.07.09

缺陷修复

iOS: 升级客户端 SDK版本，修复因为 Non-public API usage 问题导致的 AppStore 上架失败的问题。

Version 1.7.1 @ 2023.07.03

新特性

Android&iOS：新增云端通话记录，您可以在控制台开通服务进行体验查询。

功能优化

Android：降低通话时系统保活的级别，只在状态栏显示保活提示，去掉通知和震动。

Version 1.6.3 @ 2023.06.07

缺陷修复

iOS：修复调用 joinInGroupCall 后中途加人页面为空的问题。
iOS：修复调用 joinInGroupCall 后用户画面遮挡的问题。

Version 1.6.2 @ 2023.05.24

缺陷修复

Android：修复调用 joinInGroupCall api引起Crash问题 。

Version 1.6.1 @ 2023.05.14

缺陷修复

Android：修复在 Vivo 特定机型上申请悬浮窗权限时偶现的崩溃问题。
依赖说明
升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional: 11.1.0.13111、iOS TXLiteAVSDK_Professional: 11.1.14143。

Version 1.6.0 @ 2023.05.07

新特性

Android&iOS：增加hangup挂断接口。
Android&iOS：增加用户自定义字段、增加用户自定义通话超时时长。
Android&iOS：群组通话增加中途加人页面。

功能优化

Android：优化单人视频通话头像显示。
Android&iOS：群组通话中，默认支持邀请其他群成员加入通话。

缺陷修复

Android：修复 Android 12及其以上 设备连接蓝牙后无声的问题。
Android：修复被叫端偶现静音设置不生效问题。
iOS：修复重新登录后偶现设备接收不到来电邀请的问题。
iOS：修复 VoIP 推送页面昵称显示错误的问题。

依赖说明

升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional: 11.1.0.13111、iOS TXLiteAVSDK_Professional: 11.1.14143。

Version 1.5.4 @ 2023.04.14

新特性

Android&iOS：增加小米、华为、VIVO 离线推送参数。
iOS：支持 Voip 消息推送功能。
Android&iOS：增加分辨率设置功能。

功能优化

Android：优化 onCallEnd 回调中 totaltime 参数单位为毫秒。

缺陷修复

Android&iOS：修复 onCallReceived 回调异常问题。
iOS：修复屏幕旋转时通话页面显示不全的问题

Version 1.5.3 @ 2023.03.17

缺陷修复

Android：修复打包失败问题。
Android&iOS：修复回调方法未实现时抛出异常的问题。

Version 1.5.2 @ 2023.03.13

缺陷修复

Android：修复 `TUICallDefine.OfflinePushInfo` API 变更导致的编译报错。

Version 1.5.1 @ 2023.03.13

缺陷修复

Android&iOS：修复 `tencent_calls_engine` 版本依赖错误的问题。

Version 1.5.0 @ 2023.03.13

功能优化

Android：贴耳息屏功能默认关闭。
Android：升级 gradle 插件及版本。
Android：优化铃声播放类，支持循环播放。

缺陷修复

Android&iOS：修复被叫接听通话失败，没有 onCallCancel 回调问题。
Android：修复被叫接听通话失败，主叫异常问题。
Android：修复首次通话检查权限时，主叫端取消通话，被叫端再次拉起界面问题。
Android：修复回调上层的网络质量 userId 为空问题。
iOS：修复 Example 中 Observer 注册时机错误的问题。

Version 1.4.2 @ 2023.02.24

缺陷修复

Android&iOS：修复 Example 中错误的 Observer 注册时机导致的通话异常问题。
iOS：修复 `removeObserver` API 偶现设置无效的问题。

Version 1.4.1 @ 2023.02.20

缺陷修复

Android：修复通话结束后，`OnCallEnd`事件丢失的问题。

Version 1.4.0 @ 2023.01.16

缺陷修复：

Android&iOS：修复主动加入房间（joinInGroupCall）时通话异常结束问题。
Android：修复在音视频通话接听中退后台，再回到前台时，通话状态异常问题。
Android：修复同时集成 `tencent_cloud_chat_uikit` 插件时，登录状态导致的通话发起失败问题。
Android：修复发起群组通话中因为参数检查问题导致的通话发起失败问题。

Version 1.3.1 @ 2022.12.27

新特性：

Android&iOS: 支持通话时自定义离线推送消息。

缺陷修复：

Android：修复同时集成 tencent_cloud_chat_uikit 插件时，提示sdkappid is invalid的问题。

Version 1.3.0 @ 2022.12.02

功能优化：

iOS：优化 TUICallKit 的 Framework 体积。

缺陷修复：

Android&iOS：修复服务端解散房间或踢出用户时，通话界面不消失问题。
Android：修复A呼叫离线用户B，然后取消；A 再次呼叫 B，B 上线后通话界面不显示问题。

Version 1.2.2 @ 2022.11.17

缺陷修复：

iOS：修复静态库链接导致的编译问题。

Version 1.2.0 @ 2022.11.17

新特性：

支持1v1 音视频通话、群组音视频通话。

支持自定义头像、自定义昵称。

支持设置自定义铃音。

支持通话过程中开启悬浮窗。

支持多平台登录状态下的来电服务。

版权所有：腾讯云计算（北京）有限责任公司