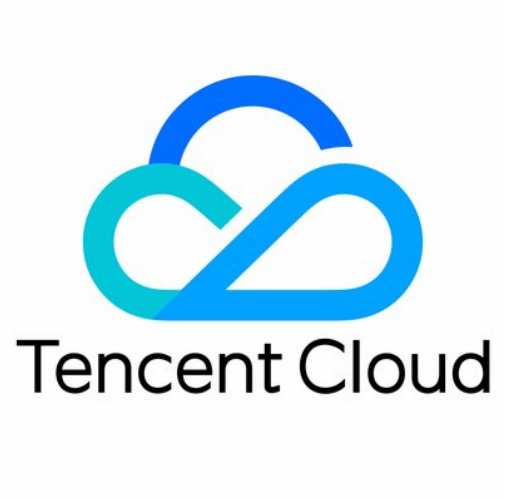
# Chat

# Video Call (UI Included)

# Product Documentation



# Contents

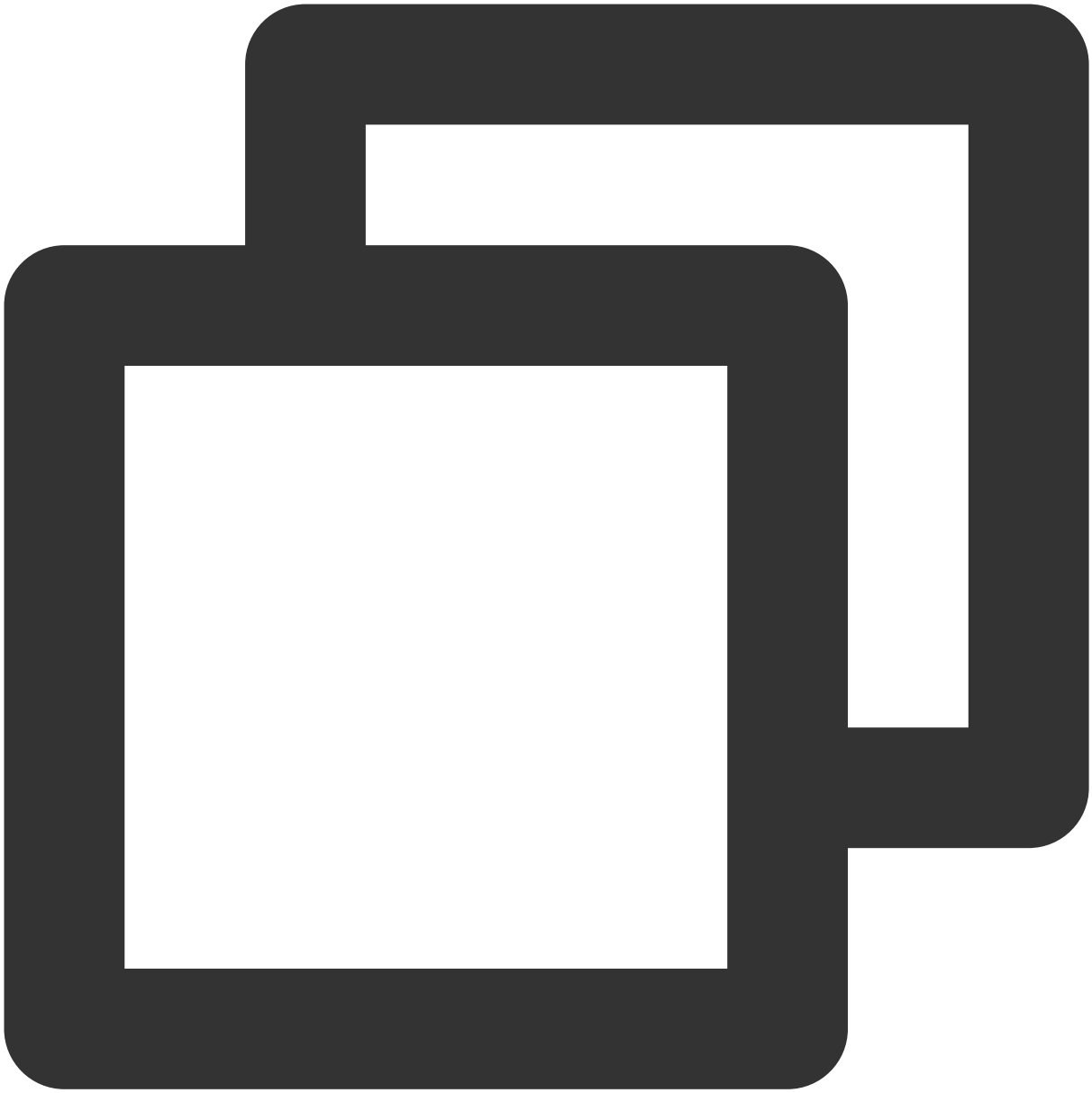
# Video Call (UI Included)
# UI Customization (TUICallKit)
# Flutter

Last updated：2024-03-13 16:25:05

This article will introduce how to customize the user interface of TUICallKit. We provide two solutions for you to choose: **interface fine-tuning solution** and **self-implementation UI** solution.
Note: The page customization solution needs to use the tencent_calls_uikit plugin version 1.8.0 or later.

## Scheme 1. Slight UI Adjustment

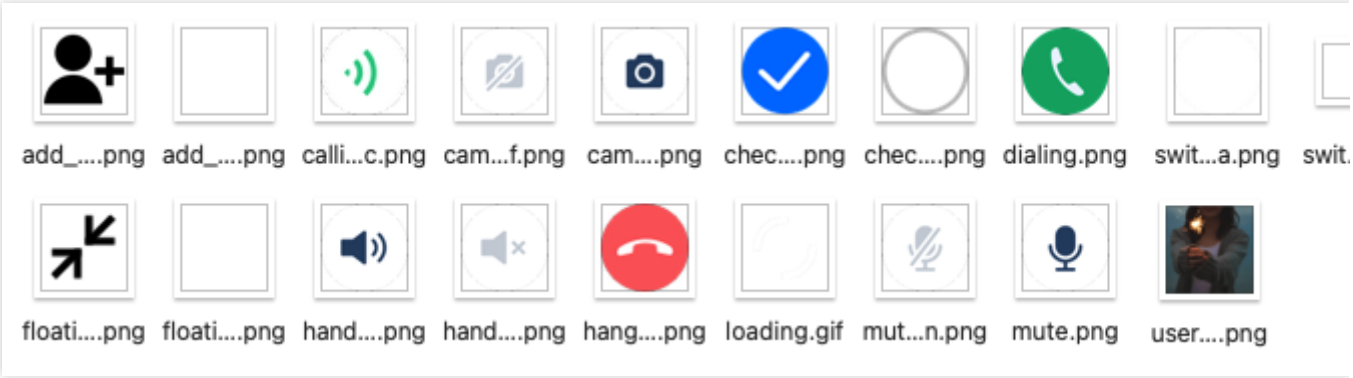
You can download the latest version of the tencent_calls_uikit plugin locally, and then use the local dependency method to access the plugin in your project. The local dependency method is as follows:
Under the **dependencies** node in the project **pubspec.yaml** file, add the **tencent_calls_uikit** plugin dependency, as shown below：



```
dependencies:
    tencent_calls_uikit:
      path: your file path
```

### replace icon

You can directly replace the icons under the **assets\\images** folder to ensure that the color tone of the icons in the entire app is consistent. Please keep the name of the icon file unchanged when replacing.




### replace ringtone

You can replace the three audio files in the **assets\\audios** folder to achieve the purpose of replacing the ringtone:

| Name | Purpose |
| --- | --- |
| phone_dialing.mp3 | The sound of making a call |
| phone_hangup.mp3 | The sound of being hung up |
| phone_ringing.mp3 | The ringtone for incoming calls |

### Replacing text

You can modify the string content in the video call interface by modifying the strings in the **strings.g.dart** file in the **lib\\src\\i18n** directory.

## Scheme 2. Custom UI Implementation

The entire call feature of `TUICallKit` is implemented based on the UI-less component `TUICallEngine`. You can delete the `tuicallkit` folder and implement your own UIs based entirely on `TUICallEngine`.

### TUICallEngine

TUICallEngine is the underlying API of the entire `TUICallKit` component. It provides key APIs such as APIs for making, answering, declining, and hanging up one-to-one audio/video and group calls and device operations.

### TUICallObserver

TUICallObserver is the callback even class of `TUICallEngine`. You can use it to listen on the desired callback events.

# Integration(TUICallKit)
# Android

Last updated：2024-03-21 15:36:32

This document describes how to quickly integrate the `TUICallKit` component. Performing the following key steps generally takes about ten minutes, after which you can implement the video call feature with complete UIs.

## Environment Preparations

Android 5.0 (SDK API level 21) or later.
Gradle 4.2.1 or later.
Mobile phone on Android 5.0 or later.

## Step 1. Activate the service

**TUICallKit** is an audio & video communication component built upon Tencent Cloud's paid PaaS services, **Chat** and **Real-time Communication (TRTC)** . In order for you to better experience the Call feature, we offer a 7-day free trial version for each SDKAppID (Note: no additional call duration is provided with the free trial version). Each SDKAppID can apply the free trial version twice, with each trial lasting for 7 days; at the same time, the total number of trial opportunities for all SDKAppID under one account (UIN) is 10.

You can activate the Call free trial version in the Chat console, with the specific operation steps as follows:

1. Log into the Chat console, select the data center, and create a new application. Skip this step if you already have an application.
2. Click on the target application card to enter the application's basic configuration page.
3. Locate the Call card, and click on "Free Trial".
4. After confirming the content in the popup window, click on "**Activate now**". Once activated, you can proceed with integration according to this document.
5. If you need to purchase the official version for your business to go live, you can proceed to the console to make the purchase. Please refer to Purchase Official Version.

## Step 2. Download and import the component

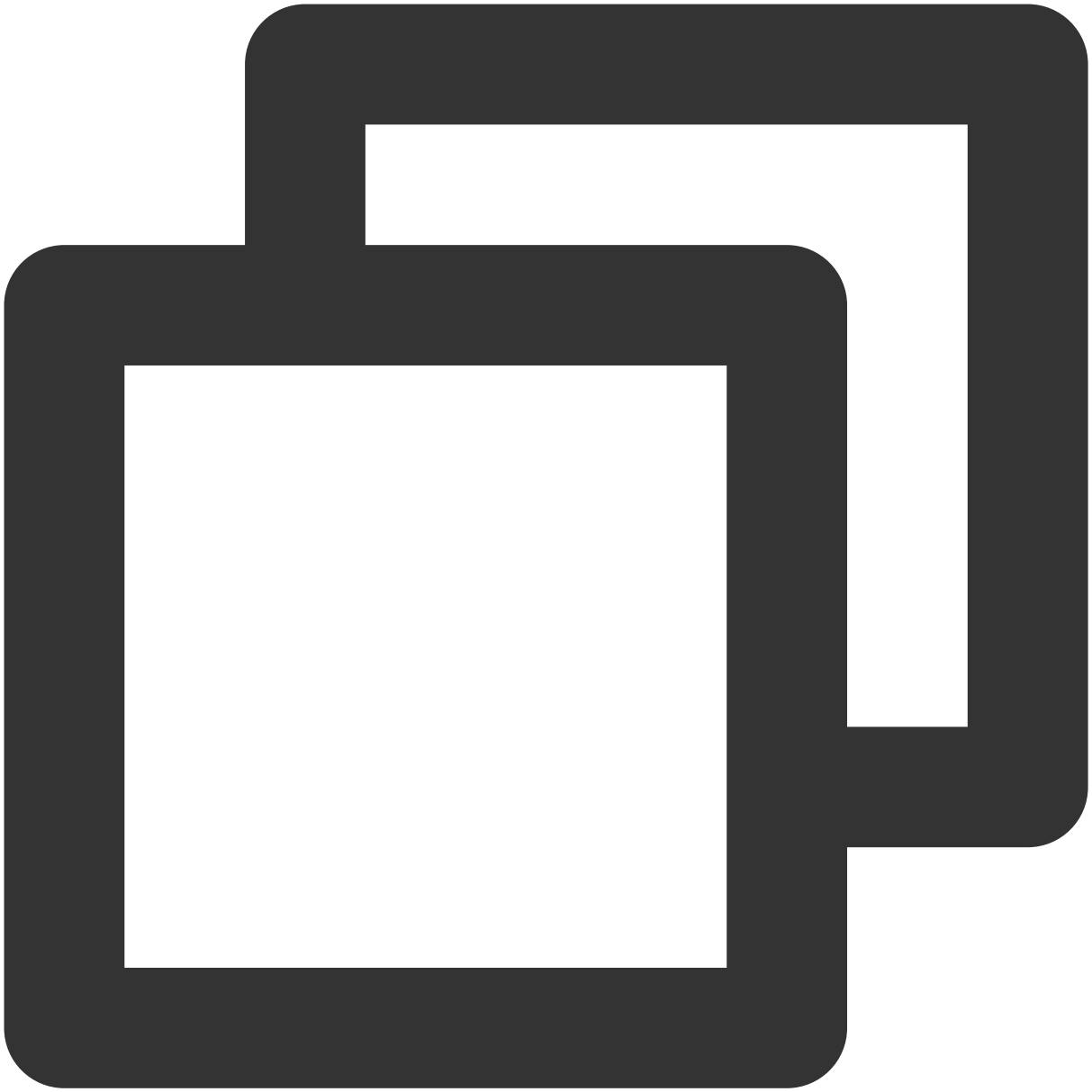
Go to GitHub, clone or download the code, and copy the `tuicallkit-kt` subdirectory in the `Android` directory to the directory at the same level as `app` in your current project, as shown below:


## Step 3. Configure the project

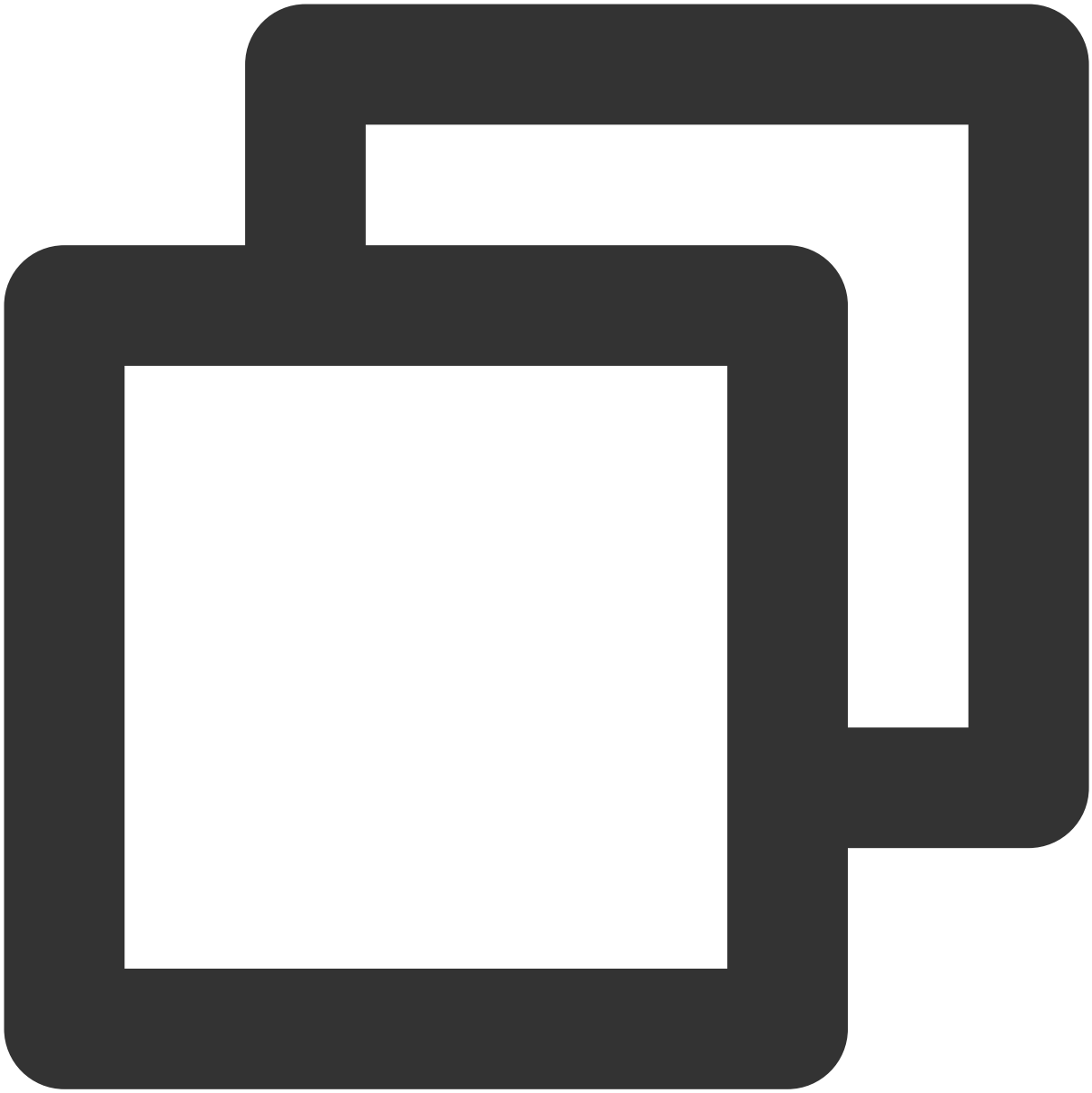
1. Find the `settings.gradle(or settings.gradle.kts)` file in the project root directory and add the following code to import the `TUICallKit` component downloaded in step 2 to your current project:

settings.gradle

settings.gradle.kts



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

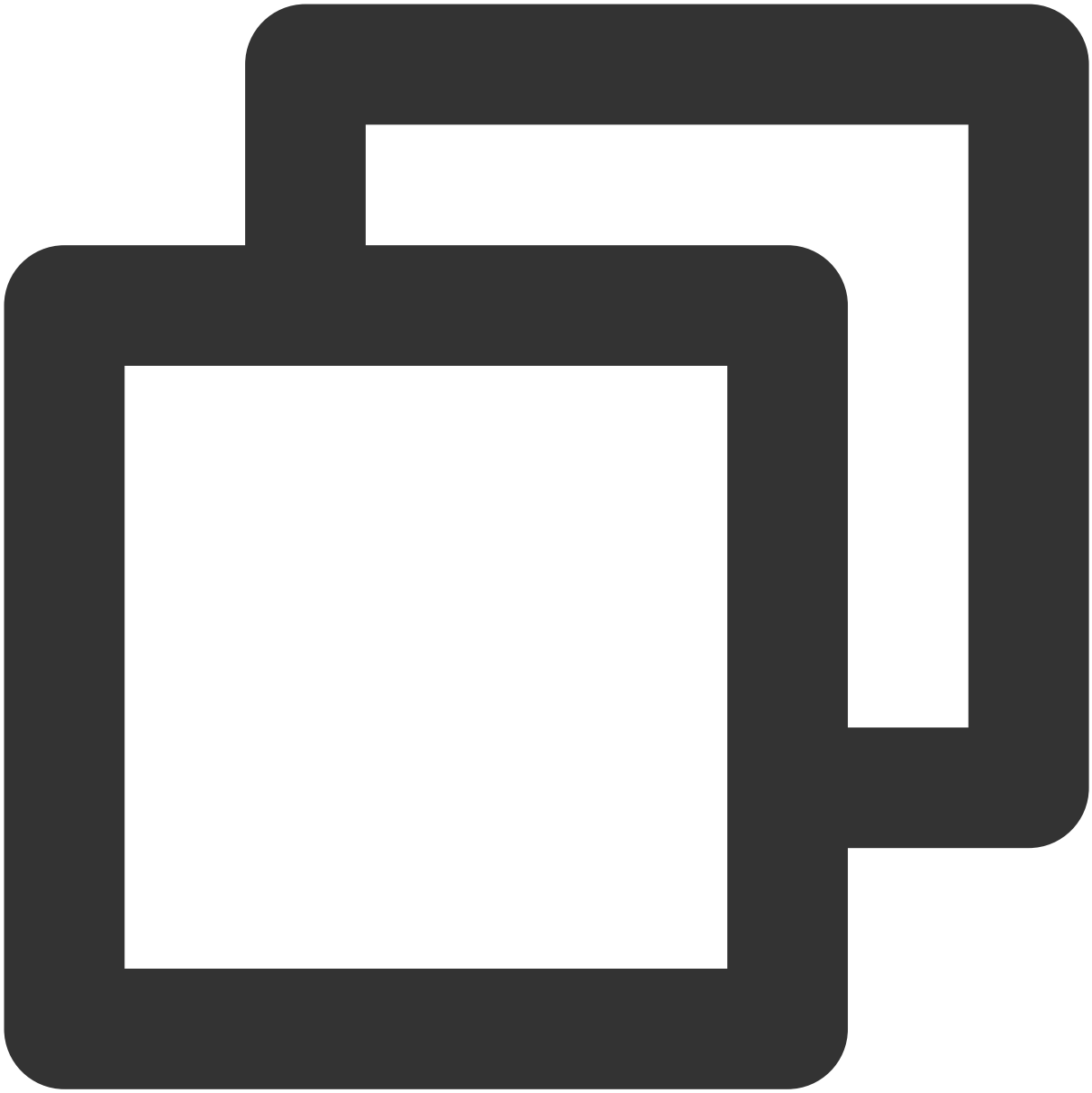


```
include(":tuicallkit-kt")
```

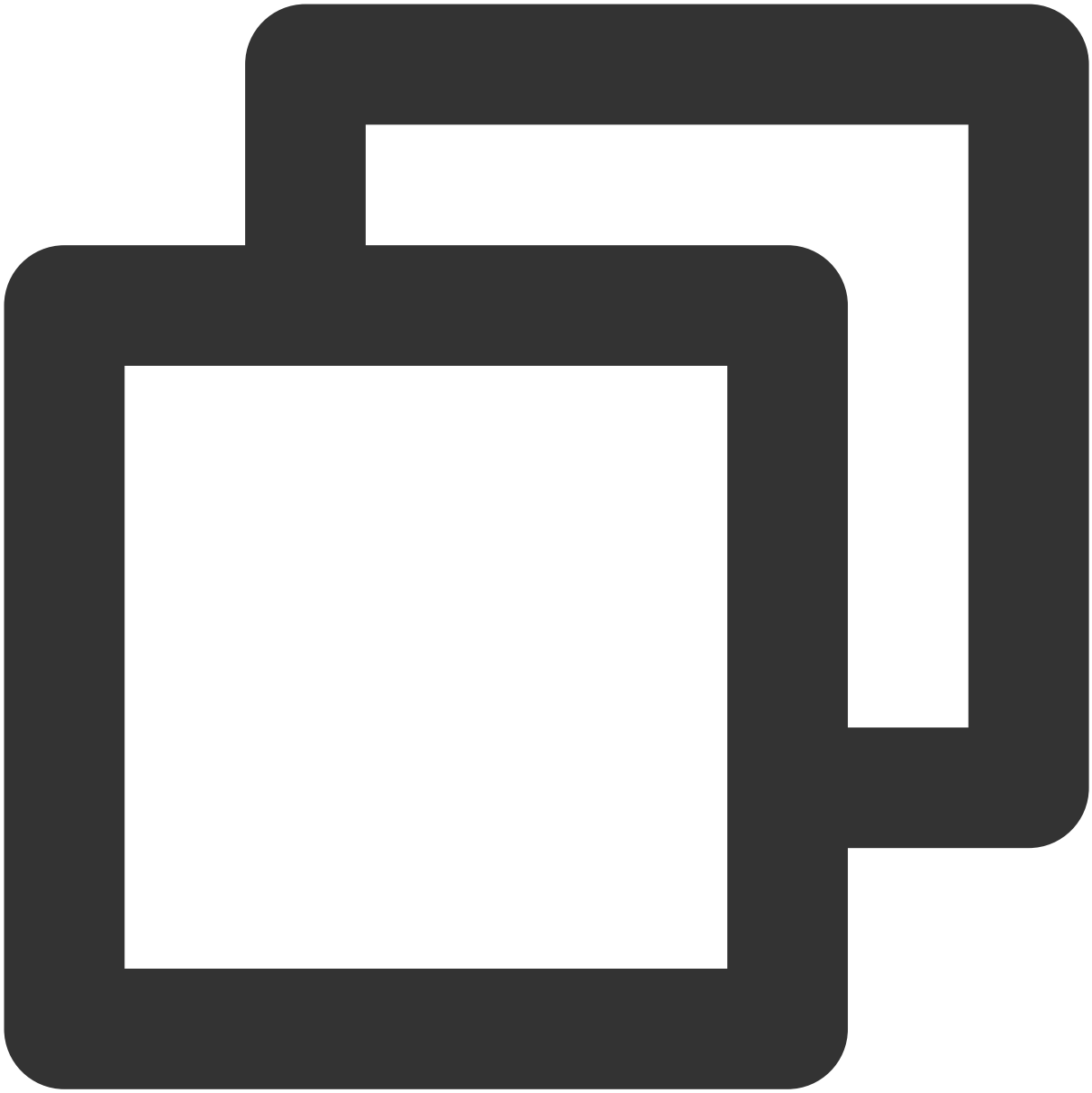
2. Find the `build.gradle(or build.gradle.kts)` file in the `app` directory and add the following code to declare the dependencies of the current application on the component just added:

build.gradle

build.gradle.kts



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

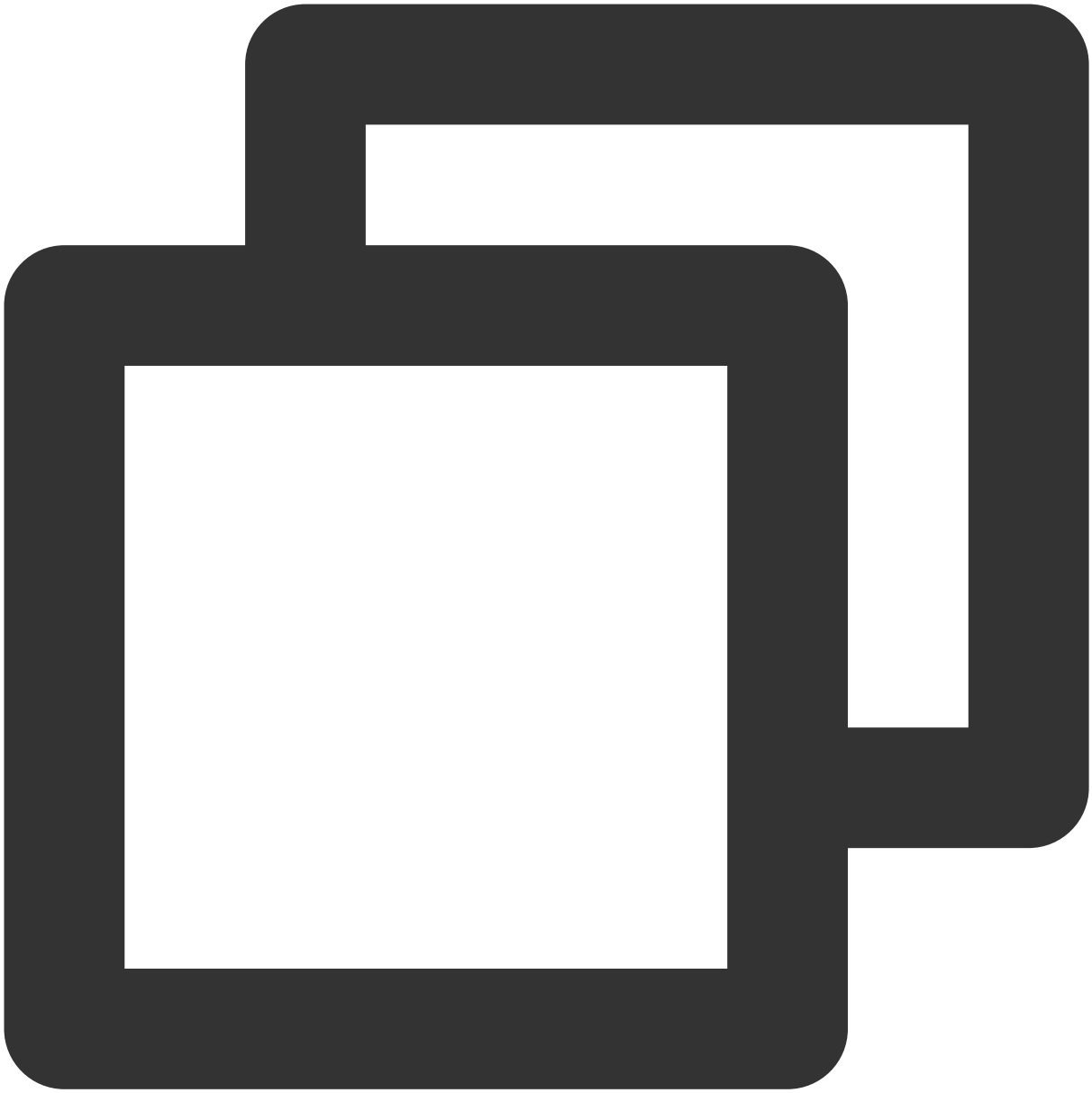


```
api(project(":tuicallkit-kt"))
```

**Note**

The `TUICallKit` project depends on `TRTC SDK`, `Chat SDK`, `tuicallengine`, and the `tuicore` public library internally by default with no need of additional configuration. To upgrade the version, modify the `tuicallkit-kt/build.gradle` file.

3. As the SDK uses Java's reflection feature internally, you need to add certain classes in the SDK to the obfuscation allowlist by adding the following code to the `proguard-rules.pro` file:



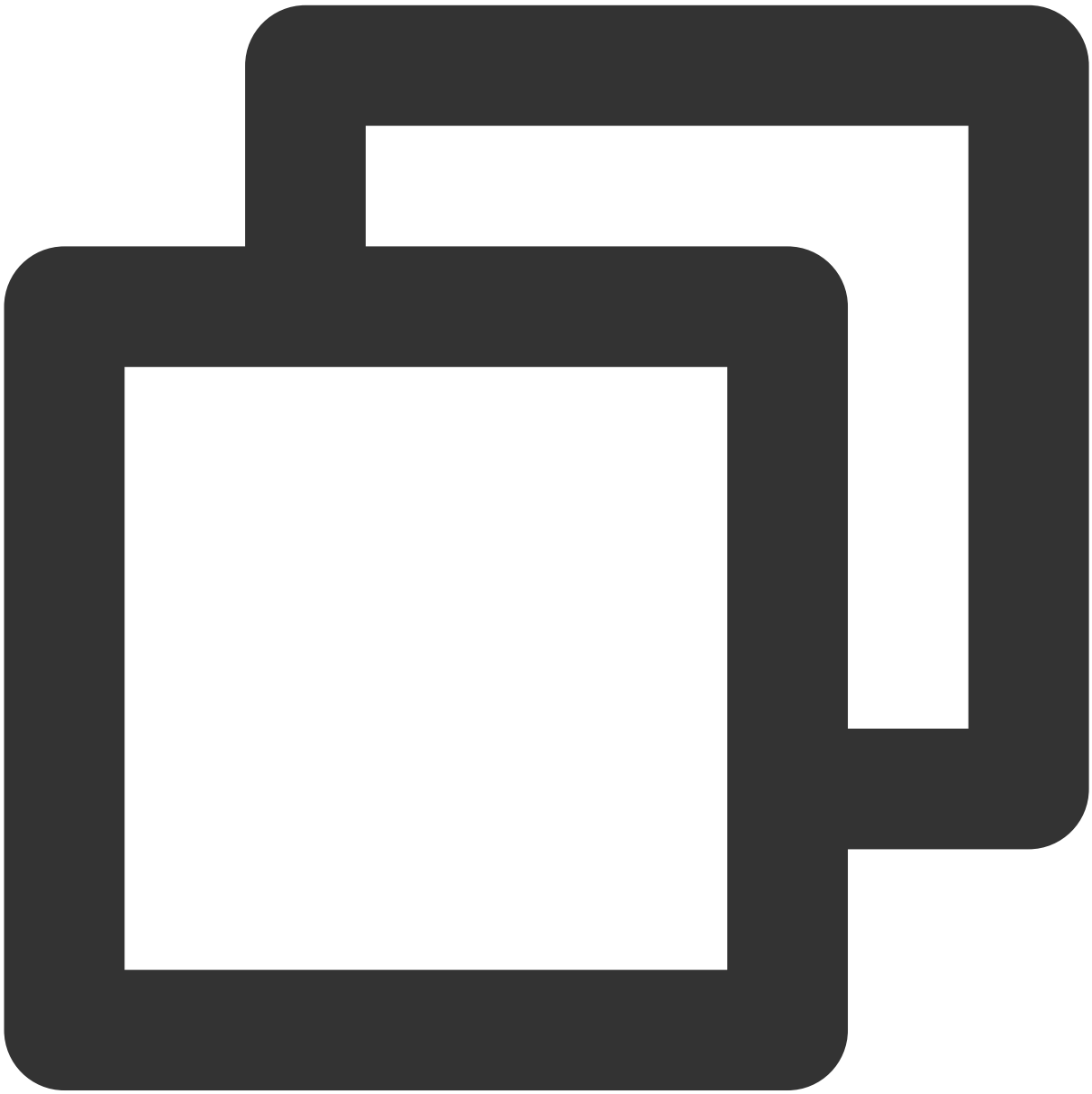
```
-keep class com.tencent.** { *;}
```

**Note**

`TUICallKit` helps you apply for camera, mic, and storage read/write permissions internally. If you need more or fewer permissions based on your actual business conditions, you can modify `tuicallkit-kt/src/main/AndroidManifest.xml` .

## Step 4. Log in to the `TUICallKit` component

Add the following code to your project to call the relevant APIs in `TUICore` to log in to the `TUICallKit` component. This step is very important, as the user can use the component features properly only after a successful login. Carefully check whether the relevant parameters are correctly configured:



```
TUILogin.login(context,
    1400000001,     // Replace it with the `SDKAppID` obtained in step 1.
    "denny",        // Replace it with your `UserID`.
    "xxxxxxxxxxx",  // You can calculate a `UserSig` in the console and enter it he
```

```
        object : TUICallback() {
            override fun onSuccess() {
            }

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

**Parameter description:** The key parameters used by the `login` function are as detailed below:

**SDKAppID**: Obtained in the last step in step 1 and not detailed here.

**UserID**: The ID of the current user, which is a string that can contain only letters (a–z and A–Z), digits (0–9), hyphens (-), or underscores (_).

**UserSig**: The authentication credential used by Tencent Cloud to verify whether the current user is allowed to use the TRTC service. You can get it by using the `SDKSecretKey` to encrypt the information such as `SDKAppID` and `UserID`. You can generate a temporary `UserSig` by clicking the UserSig Generate button in the console. For more information, see UserSig.

**Note**

**Many developers have contacted us with many questions regarding this step. Below are some of the frequently encountered problems:**

SDKAppID is invalid.

UserSig is set to the value of Secretkey mistakenly. The UserSig is generated by using the SecretKey for the purpose of encrypting information such as SDKAppID, UserID, and the expiration time. But the value of the UserSig that is required cannot be directly substituted with the value of the SecretKey.

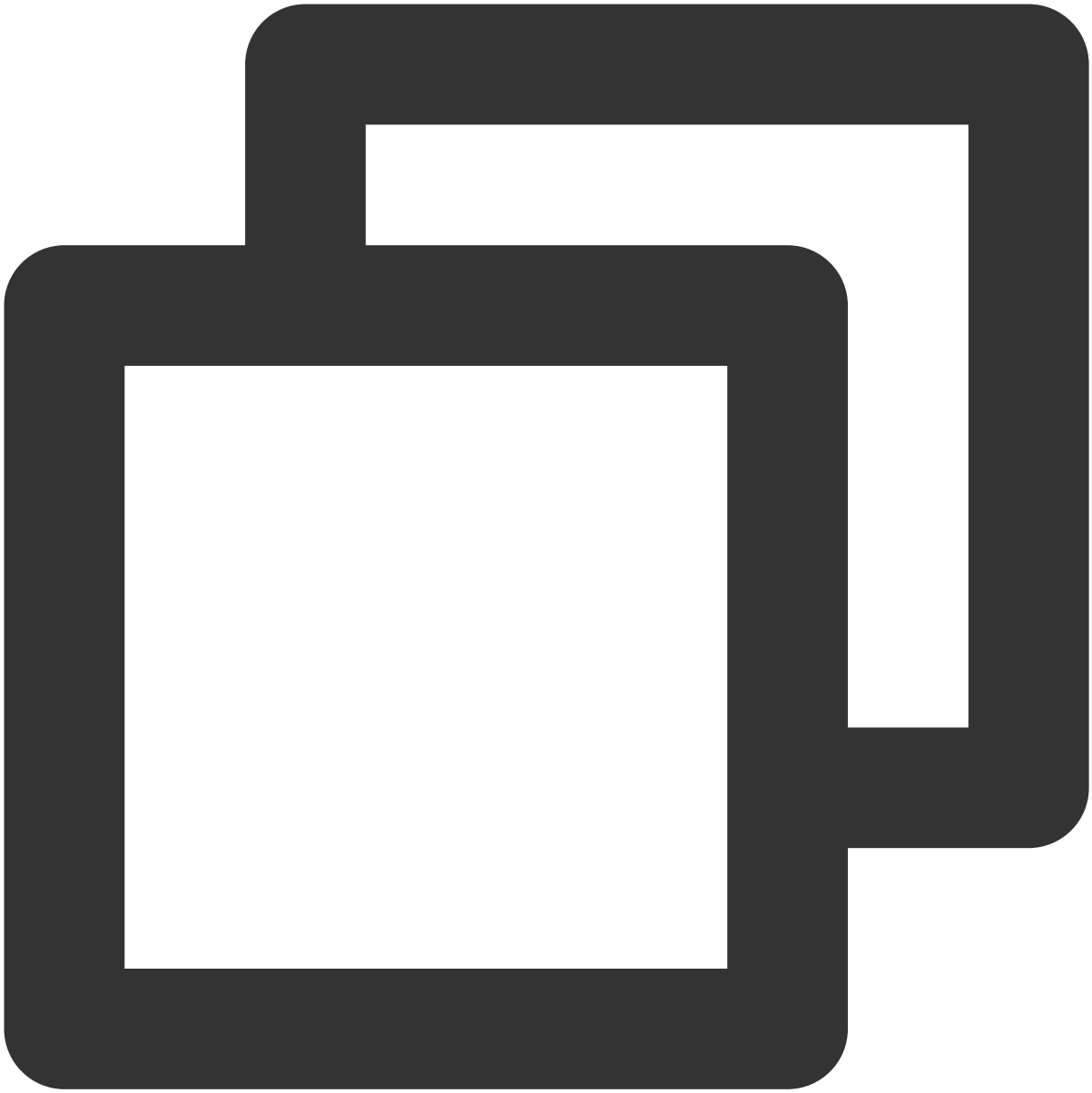
UserID is set to a simple string such as 1, 123, or 111, and your colleague may be using the same userId while working on a project simultaneously. In this case, login will fail as TRTC doesn't support login on multiple terminals with the same UserID. Therefore, we recommend you use some distinguishable userId values during debugging.

The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current integration process more quickly. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` subsequently. Therefore, we strongly recommend you run the `UserSig` calculation logic on the server and make the application request the `UserSig` calculated in real time every time the application uses the `TUICallKit` component from the server.

# Step 5. Make a call

### One-to-one video call

You can call the `call` function of `TUICallKit` and specify the call type and the callee's `userId` to make an audio/video call.

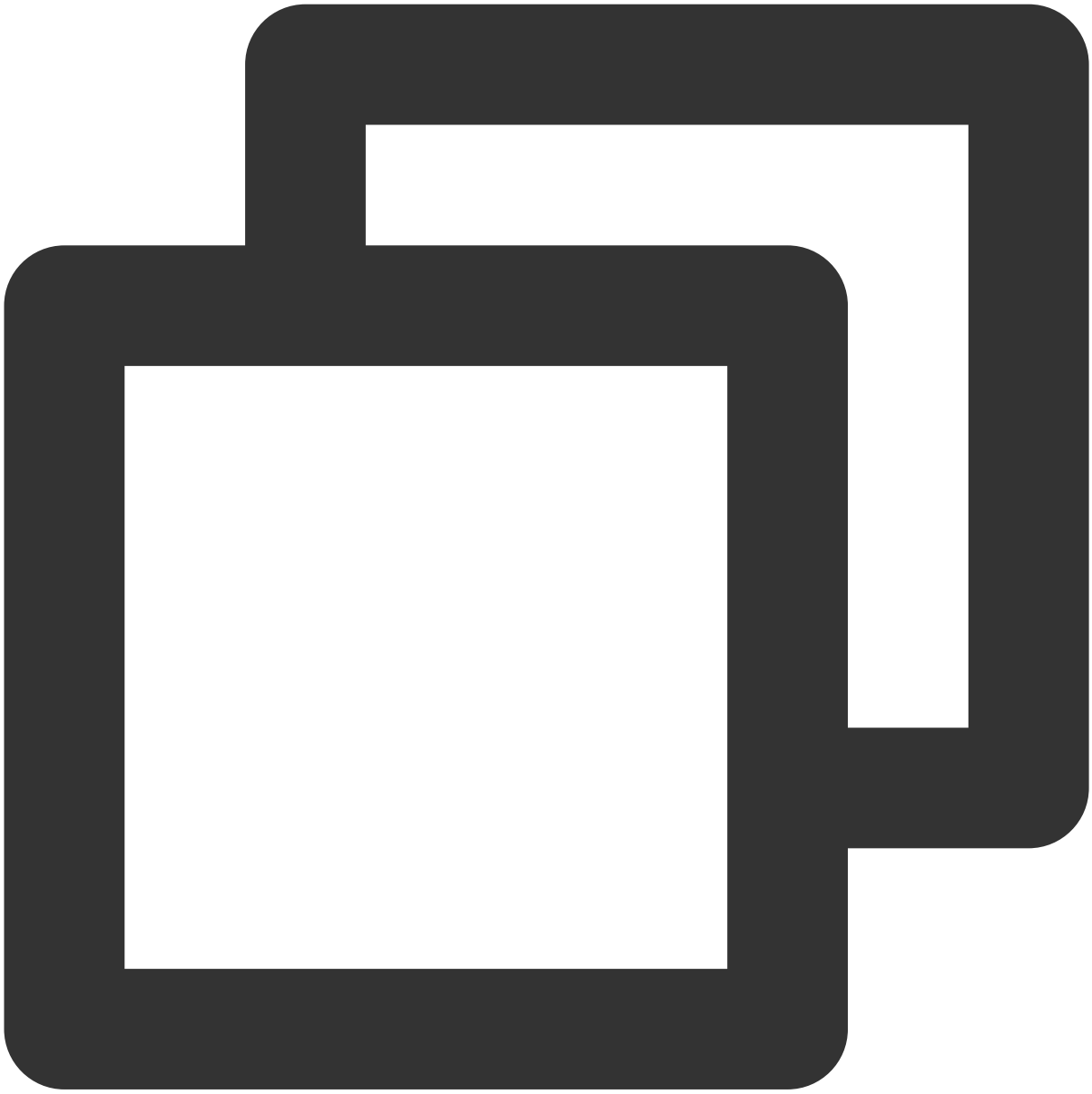


```
// Make a one-to-one video call. Suppose the `UserID` is `mike`.
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video)
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the target user, such as `"mike"`. |
| callMediaType | TUICallDefine.MediaType | The call media type, such as `TUICallDefine.MediaType.Video`. |

### Group video call

You can call the `groupCall` function of `TUICallKit` and specify the call type and the list of callees' `UserID` values to make a group audio/video call.



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

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID, such as `"12345678"` . |

| | | |
|---|---|---|
| userIdList | List | The list of `UserID` values of the target users, such as `{"jane", "mike", "tommy"}`. |
| callMediaType | TUICallDefine.MediaType | The call media type, such as `TUICallDefine.MediaType.Video`. |

**Note**

You can create a group as instructed in Android, iOS, and macOS. You can also use Chat TUIKit to integrate chat and call scenarios at one stop.

`TUICallKit` currently doesn't support making a multi-person video call among users not in a group. If you have such a need, contact info_rtc@tencent.com.

# Step 6. Answer a call

After receiving an incoming call, the `TUICallKit` component will automatically wake up the call answering UI. However, the wake effect varies by Android system permissions as follows:

If your application is in the foreground, it will pop up the call UI and play back the incoming call ringtone automatically when receiving an incoming call.

If your application is in the background and is granted the `Display over other apps` or `Display pop-up windows while running in the background` permission, it will still pop up the call UI and play back the incoming call ringtone automatically when receiving an incoming call.

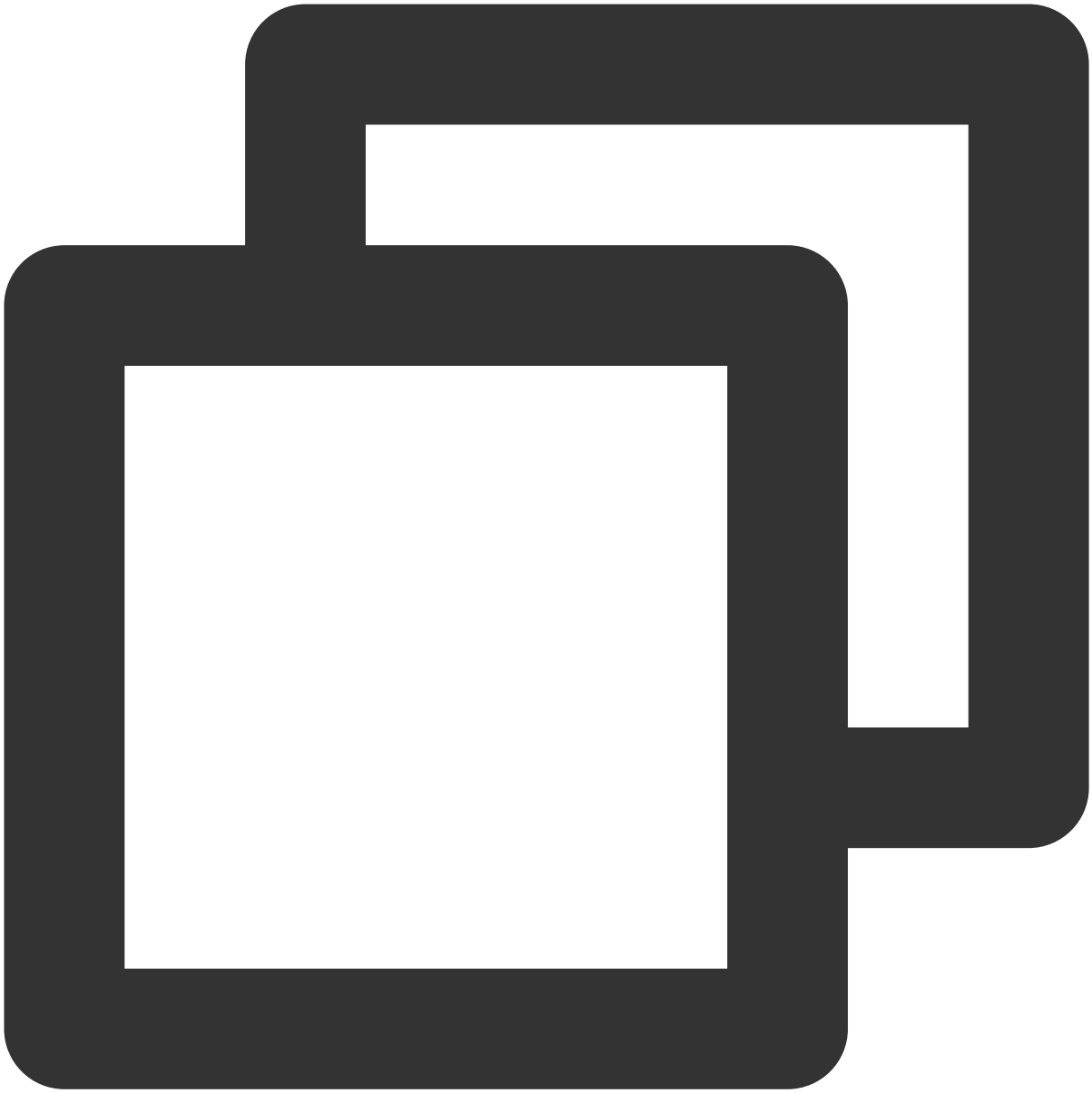
If your application is in the background but isn't granted with the `Display over other apps` or `Display pop-up windows while running in the background` permission, If you click the application icon and enter the application, It will pop up the call UI.

If the application process has been terminated, you can use the offline push feature as described in Offline Call Push (Android) to prompt the user to answer or decline the call through the status bar notification.

# Step 7. Implement more features

### 1. Nickname and profile photo settings

To customize the nickname or profile photo, use the following API for update:



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

**Note**

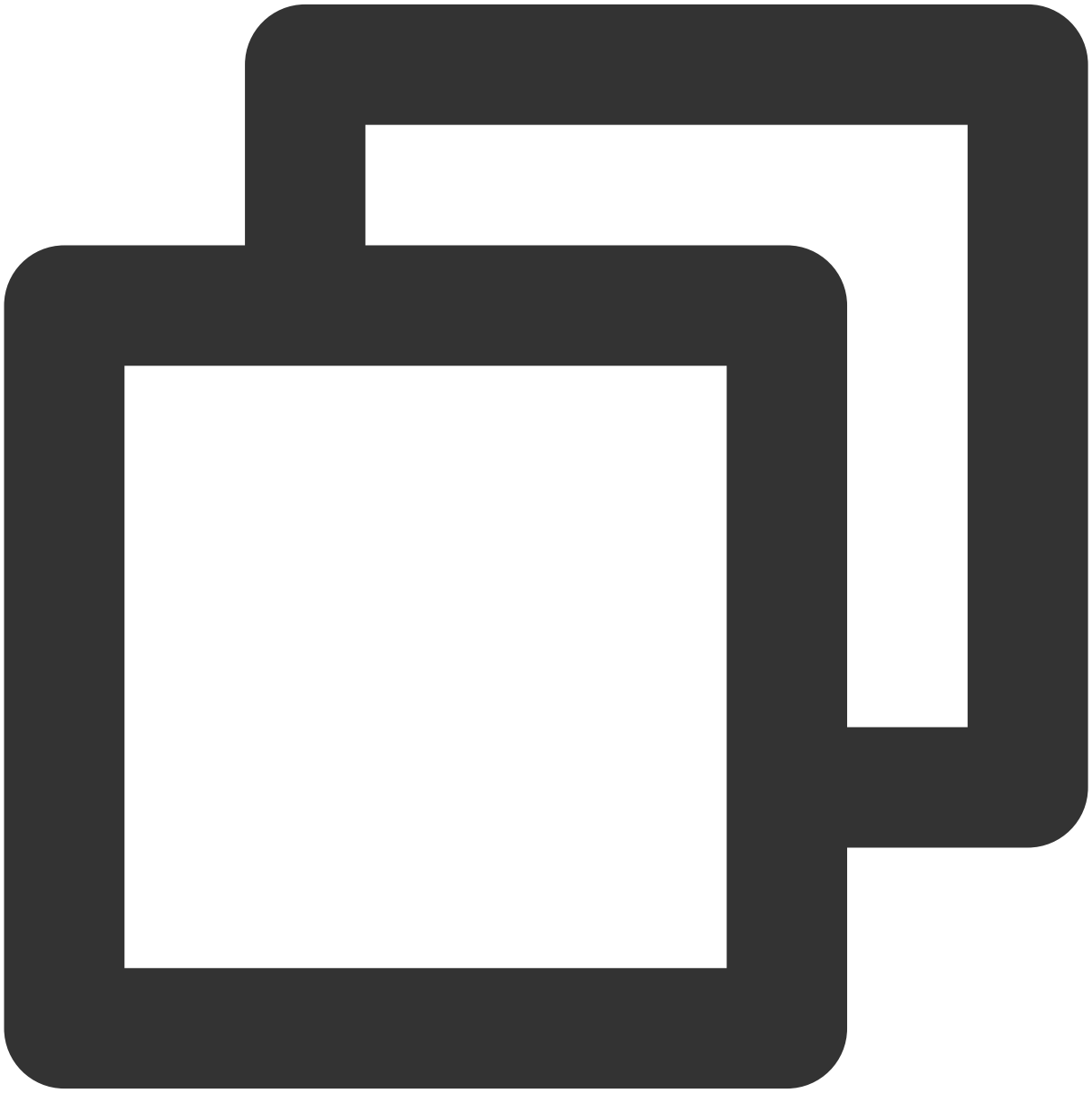
The update of the callee's nickname and profile photo may be delayed during a call between non-friend users due to the user privacy settings. After a call is made successfully, the information will also be updated properly in subsequent calls.

### 2. Offline call push

You can make/answer audio or video calls after completing the above steps. However, if you want your users to be able to receive call invitations even when your application is in the background or after it is closed, then you need to also implement the offline call push feature. For more information, see Offline Call Push (Android).

### 3. Floating window

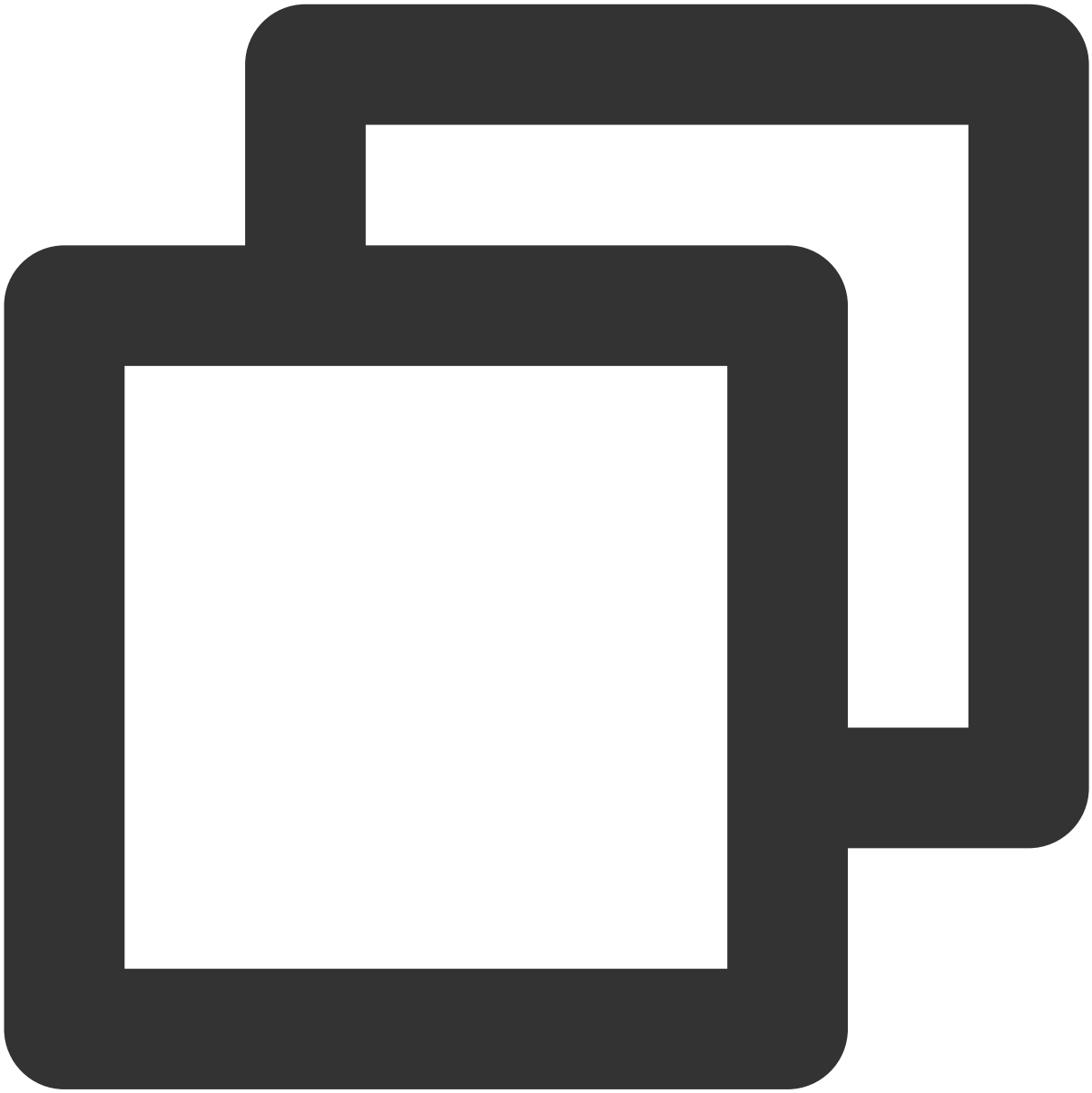
To implement the floating window feature in your application, call the following API when initializing the `TUICallKit` component:



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

### 4. Call status listening

To **listen on the call status** (for example, the start or end of a call or the call quality during a call), listen on the following events:

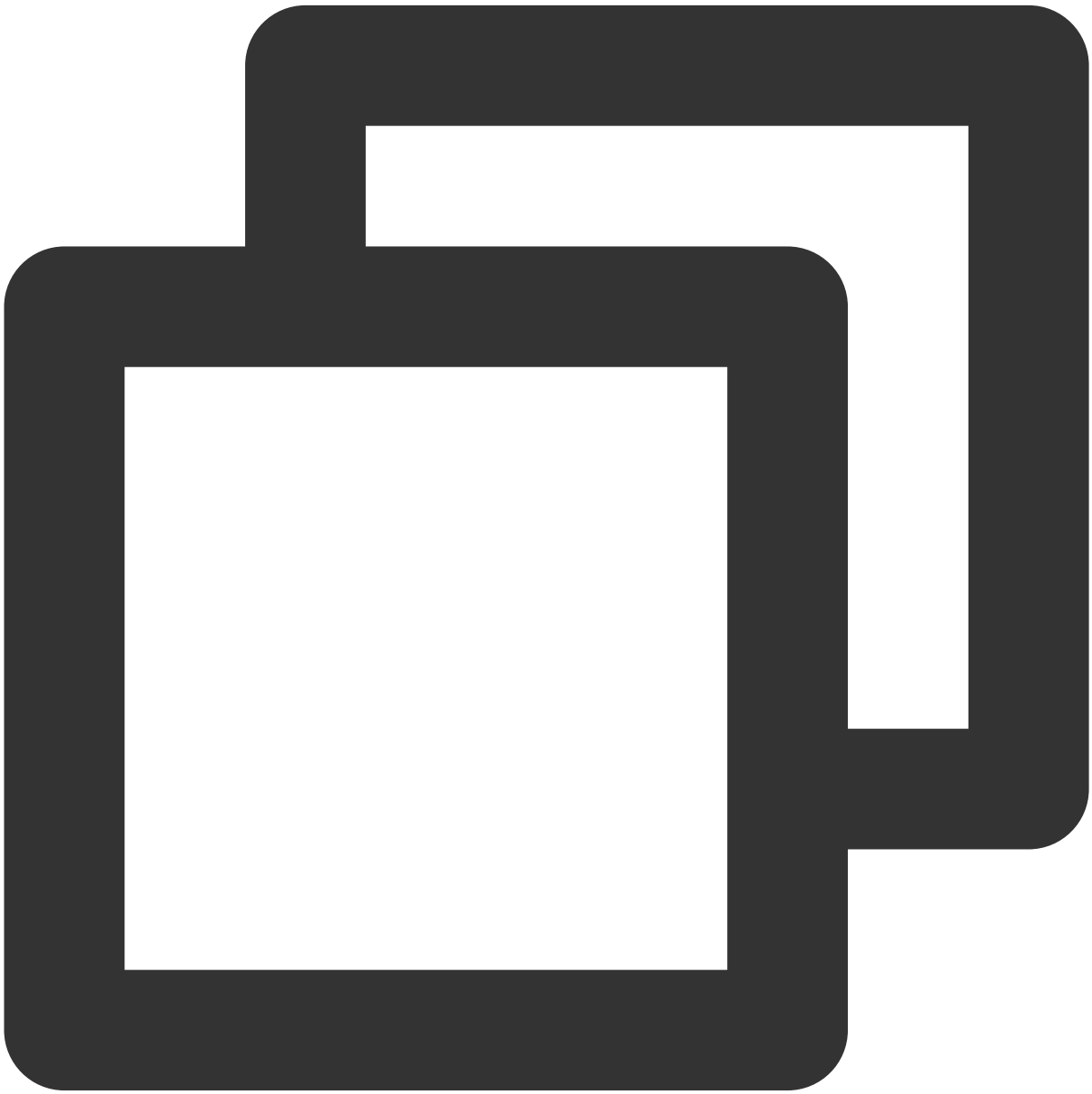


```
private val observer: TUICallObserver = object : TUICallObserver() {
    override fun onCallBegin(roomId: TUICommonDefine.RoomId?, callMediaType: TUICal
    }
    override fun onCallEnd(roomId: TUICommonDefine.RoomId?, callMediaType: TUICallD
    }
    override fun onUserNetworkQualityChanged(networkQualityList: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

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

### 5. Custom ringtone

You can use the following API to customize the ringtone:



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

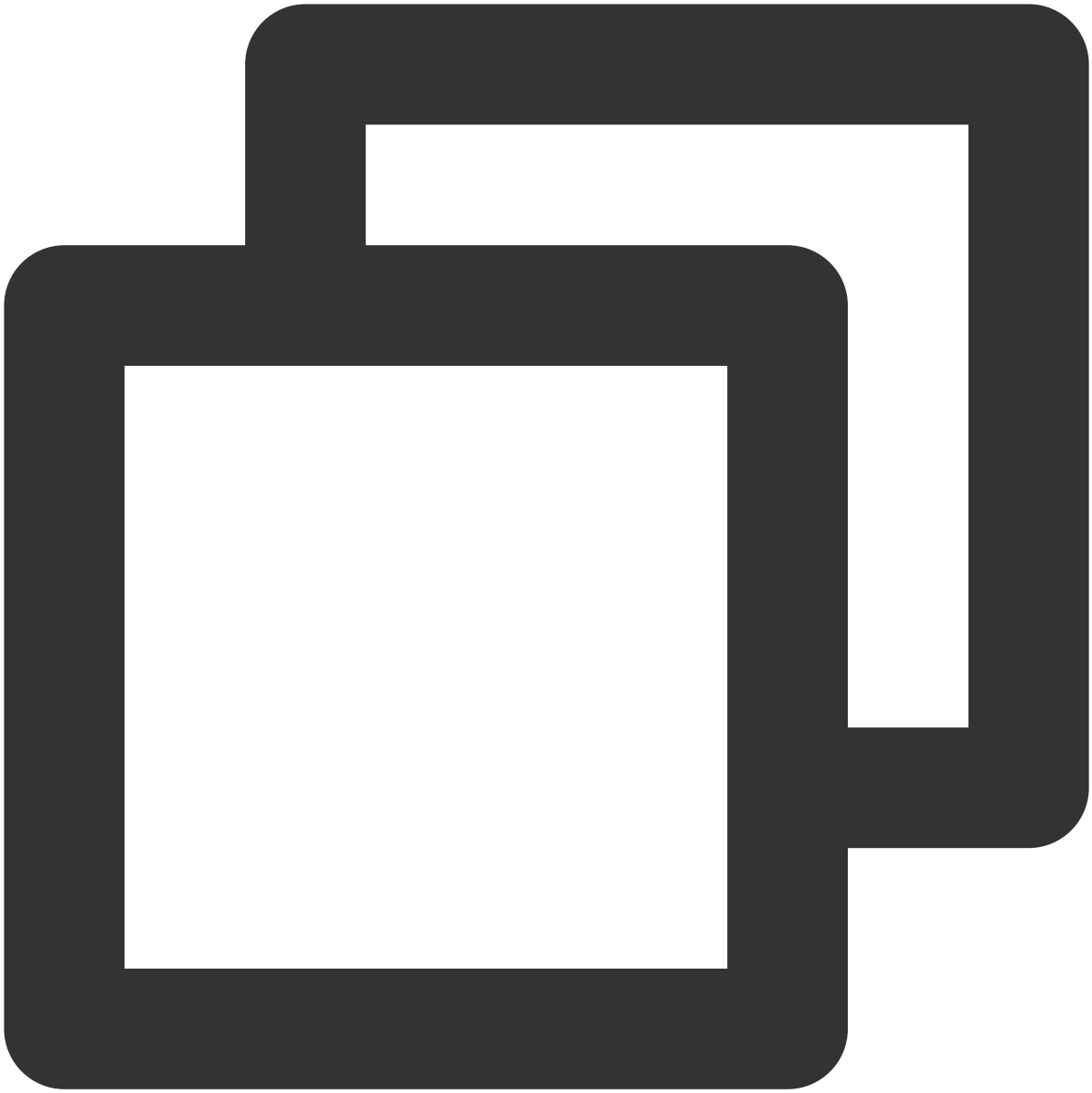
# FAQs

### 1. What should I do if I receive the error message "The package you purchased does not support this ability"?

The error message indicates that your application's audio/video call capability package has expired or is not activated. You can claim or activate the audio/video call capability as instructed in step 1 to continue using `TUICallKit`.

### 2. What should I do if `TUICallKit` crashes and outputs the log "No implementation found for xxxx"?

Below is the stack information:



```
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)
```

If the above exception occurs on a real device, it is because some APIs of SDKs such as the TRTC SDK depended on by `TUICallKit` are implemented through JNI, but Android Studio may not package native .so libraries when compiling the APK under some conditions. In this case, just clean the project again.

## Suggestions and Feedback

If you have any suggestions or feedback, please contact info_rtc@tencent.com.

# iOS

Last updated：2024-03-18 15:06:32

This document describes how to quickly integrate the `TUICallKit` component. Performing the following key steps generally takes about an hour, after which you can implement the video call feature with complete UIs.

## Environment Preparations

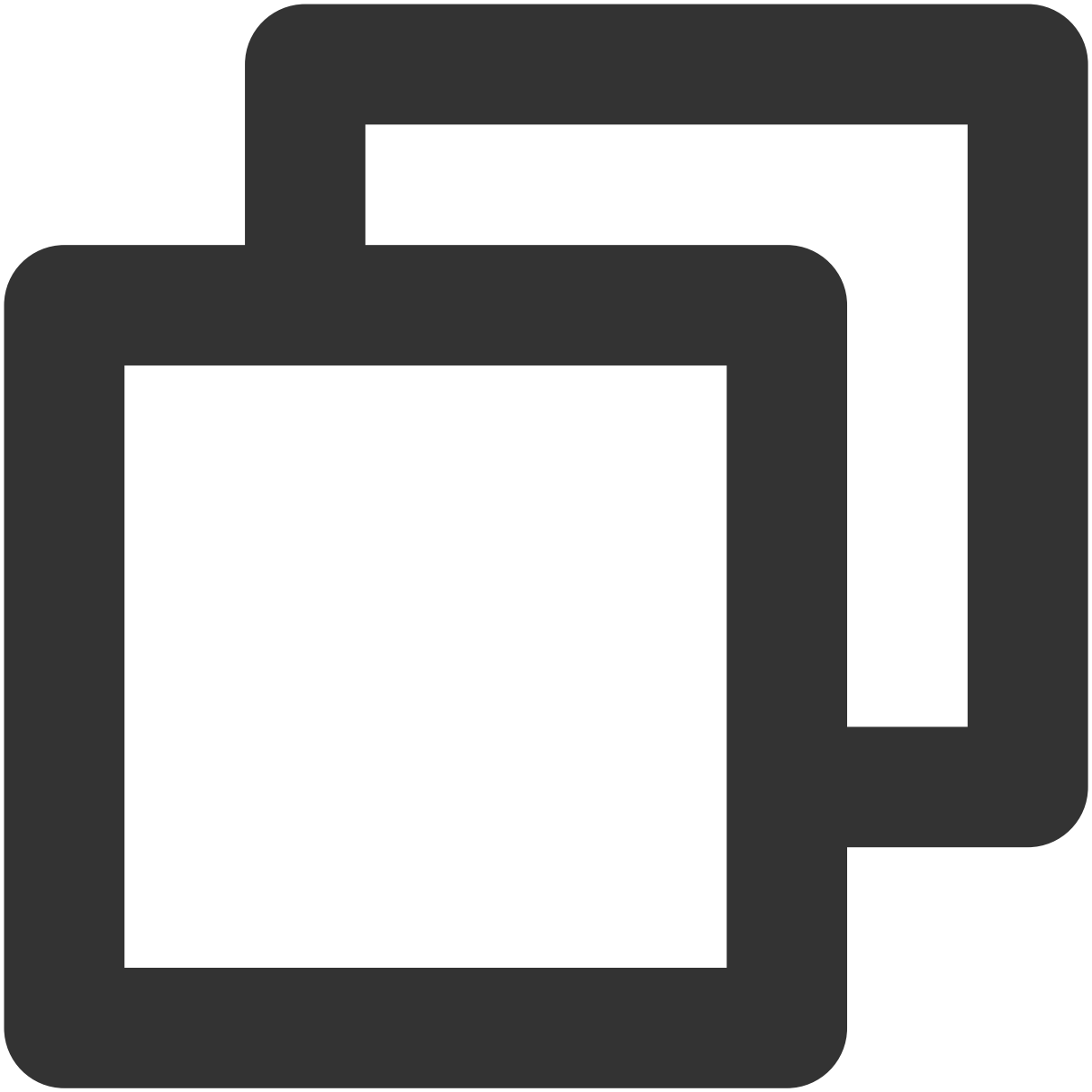
Xcode 13 or later.
iOS 12.0 or later.

## Step 1. Activate the service

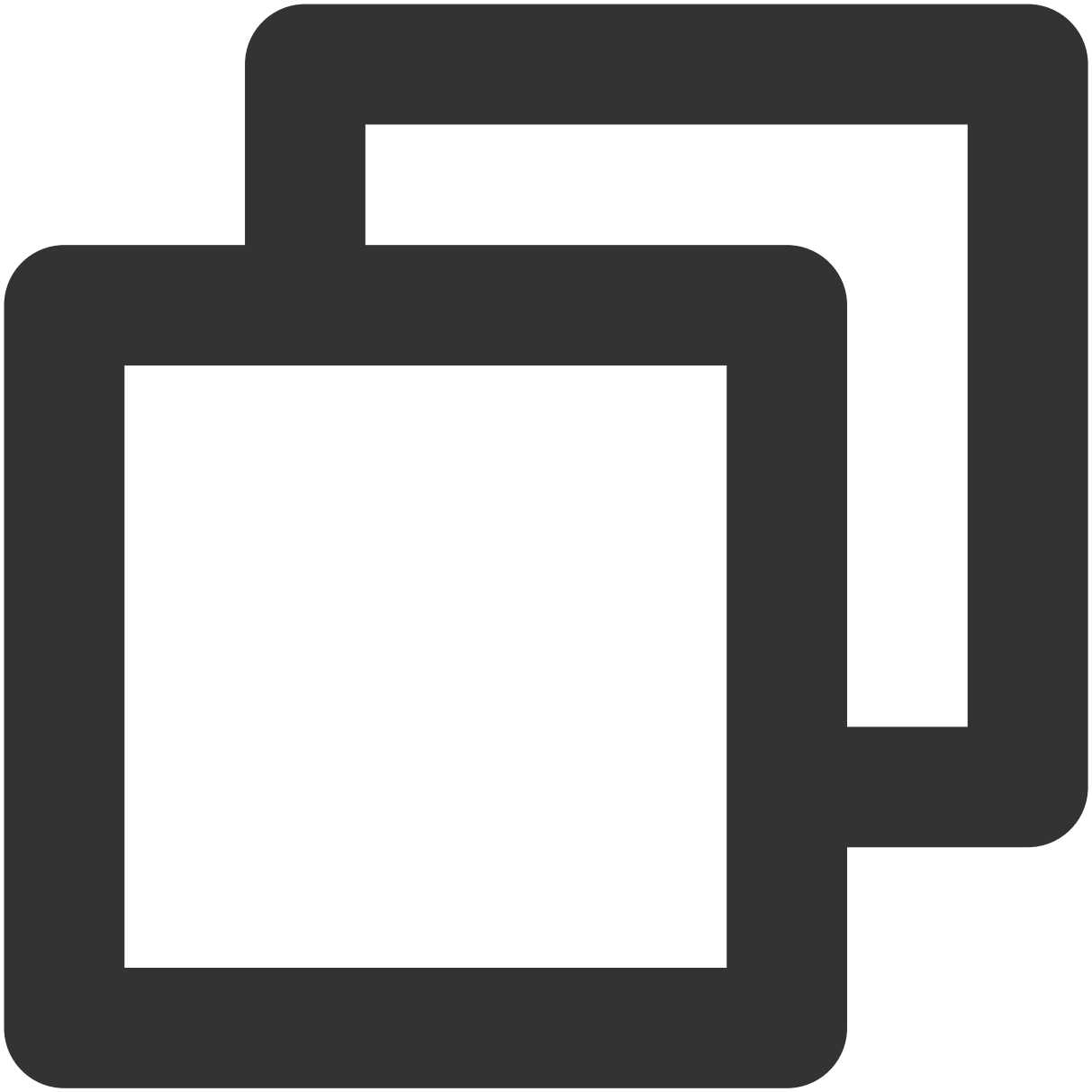
**TUICallKit** is an audio & video communication component built upon Tencent Cloud's paid PaaS services, **Chat** and **Real-time Communication (TRTC)** . In order for you to better experience the Call feature, we offer a 7-day free trial version for each SDKAppID (Note: no additional call duration is provided with the free trial version). Each SDKAppID can apply the free trial version twice, with each trial lasting for 7 days; at the same time, the total number of trial opportunities for all SDKAppID under one account (UIN) is 10.
You can activate the Call free trial version in the Chat console, with the specific operation steps as follows:
1. Log into the Chat console, select the data center, and create a new application. Skip this step if you already have an application.
2. Click on the target application card to enter the application's basic configuration page.
3. Locate the Call card, and click on "Free Trial".
4. After confirming the content in the popup window, click on "**Activate now**". Once activated, you can proceed with integration according to this document.
5. If you need to purchase the official version for your business to go live, you can proceed to the console to make the purchase. Please refer to Purchase Official Version.

## Step 2. Import the component

Use CocoaPods to import the component as follows:
1. Add the following dependency to your `Podfile` .
Swift
Objective-C

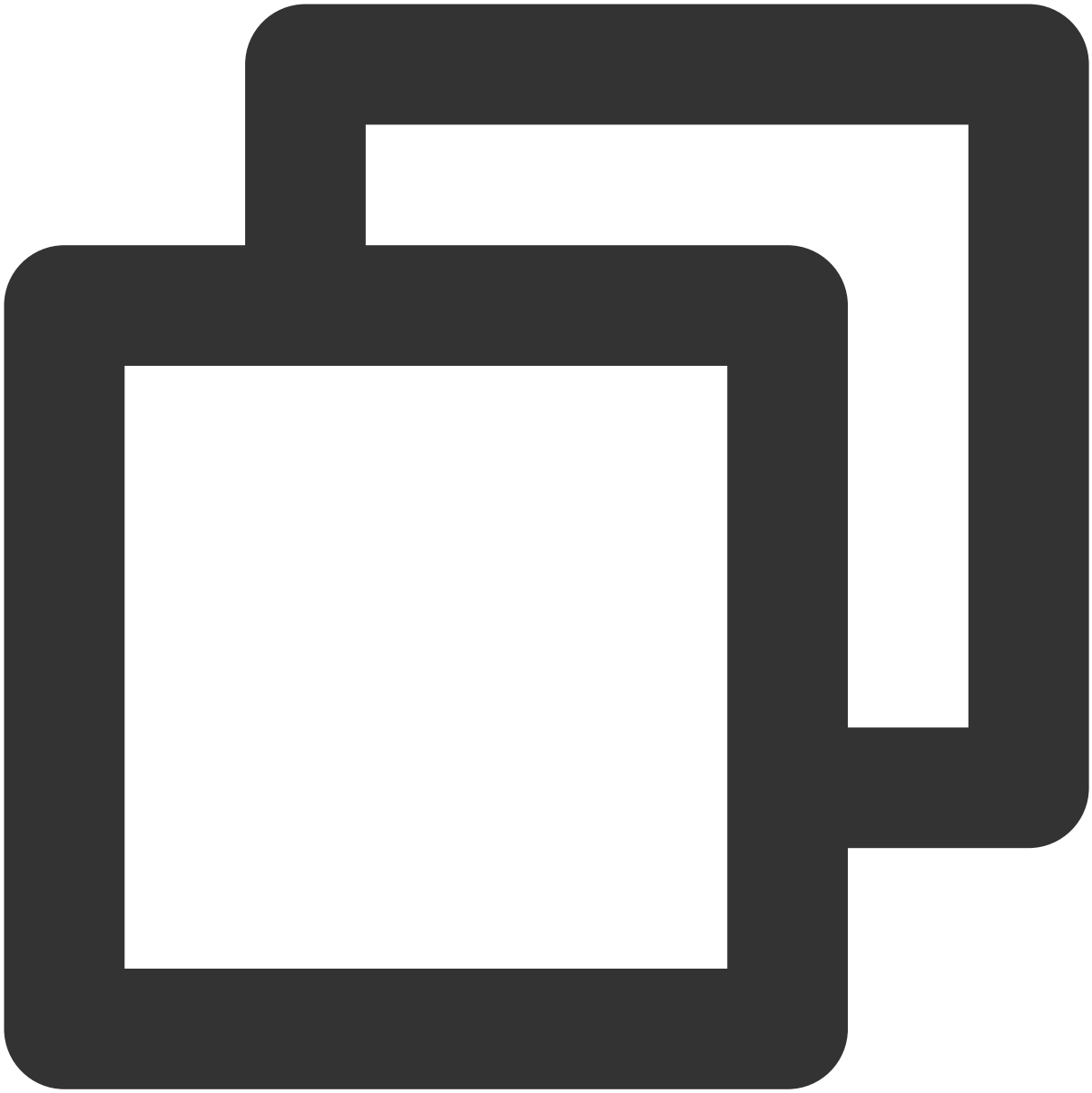


```
pod 'TUICallKit_Swift'
```



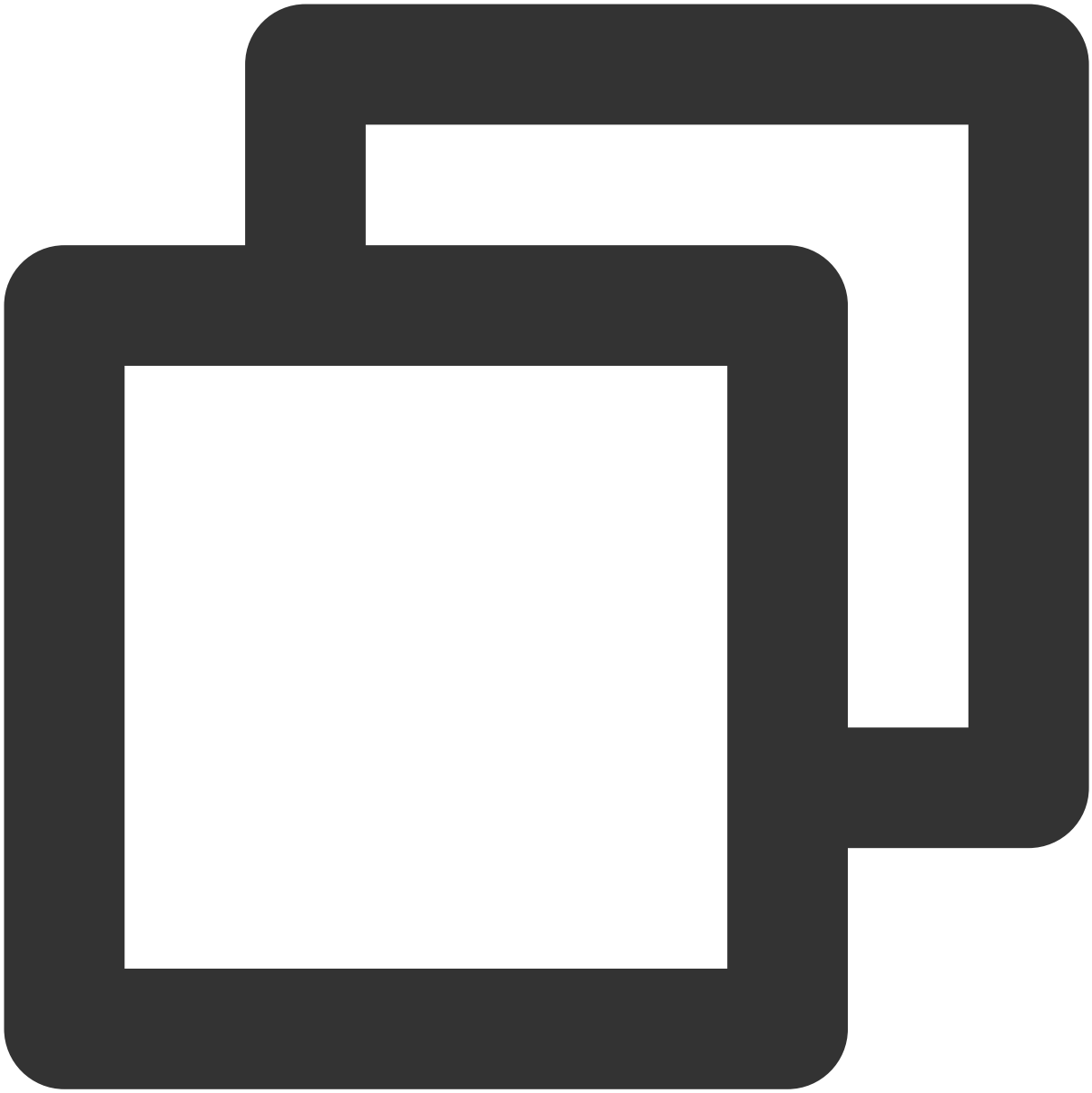
```
pod 'TUICallKit'
```

2. Run the following command to install the component:



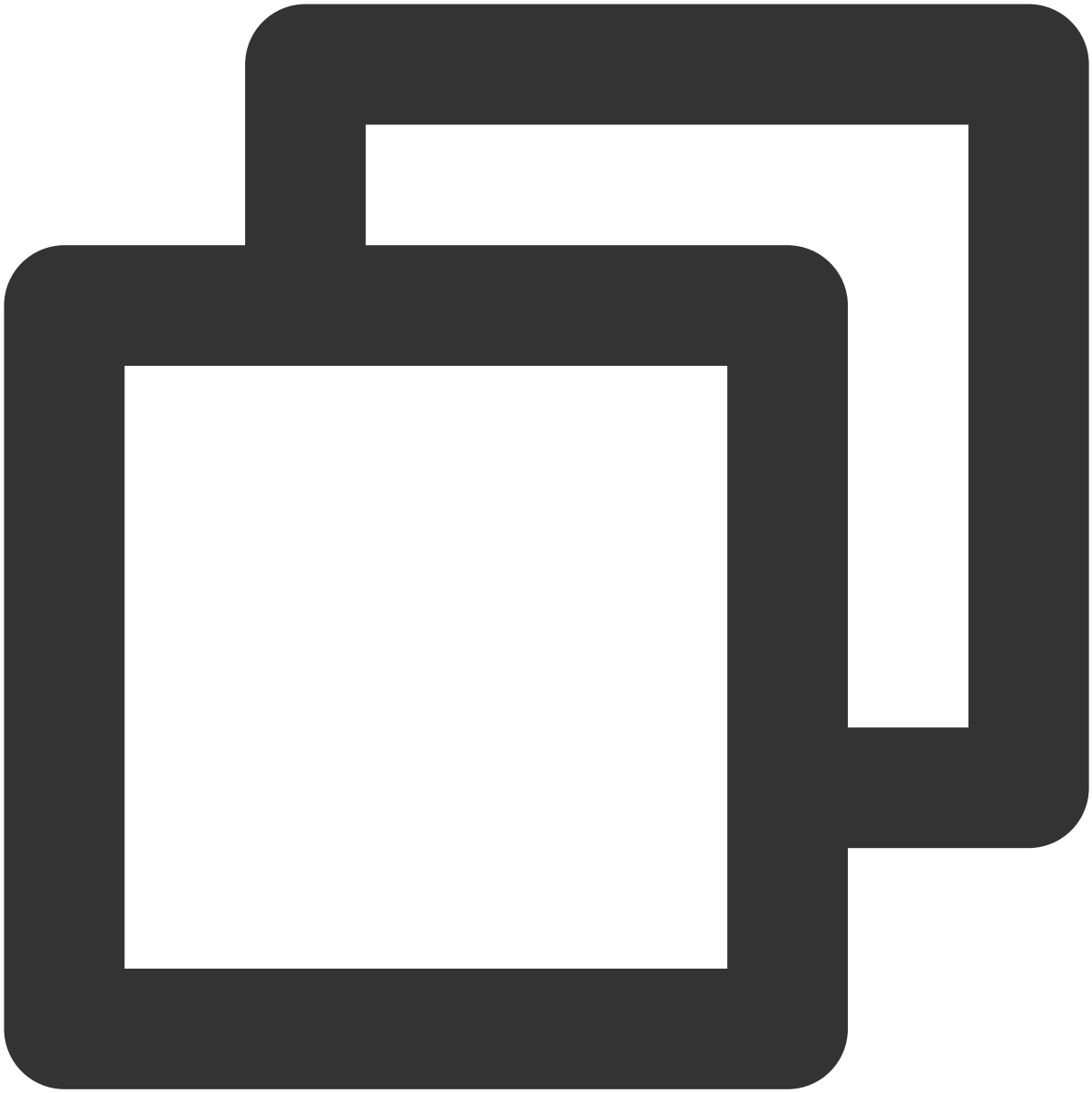
```
pod install
```

If the latest version of `TUICallKit` cannot be installed, run the following command to update the local CocoaPods repository list:



```
pod repo update
```

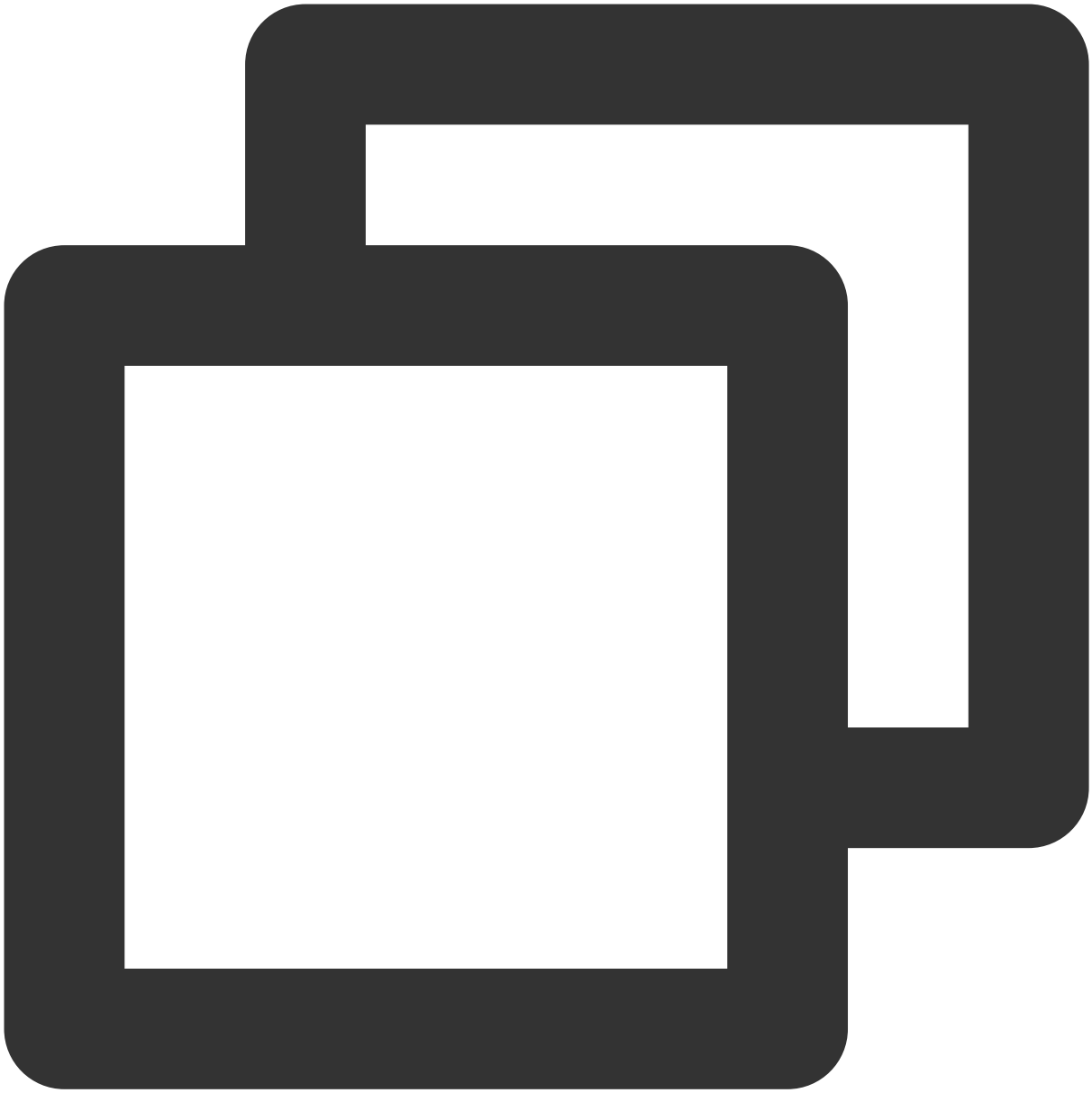
Afterwards, execute the following command to update the Pod version of the component library.



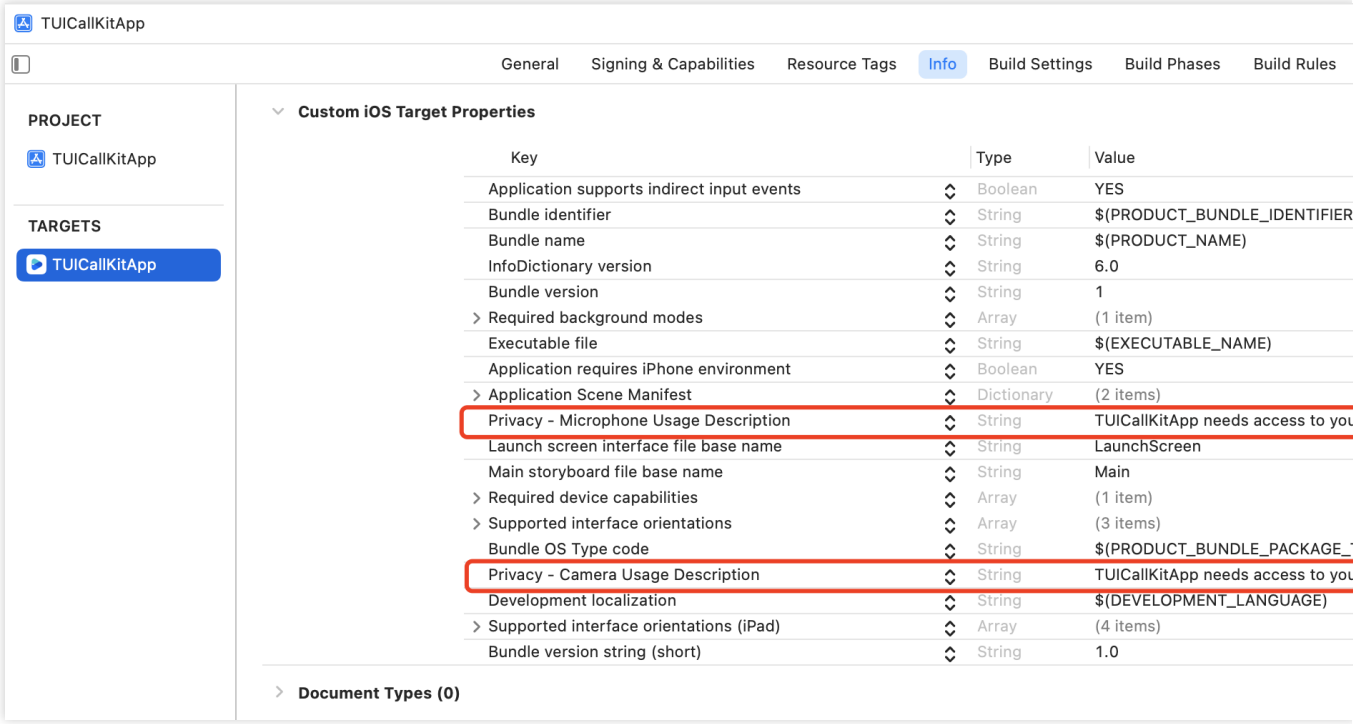
```
pod update
```

## Step 3. Configure the project

Your app needs mic and camera permissions to implement audio/video communication. Add the two items below to `Info.plist` of your app. Their content is what users see in the mic and camera access pop-up windows.



```
<key>NSCameraUsageDescription</key>
<string>CallingApp needs to access your camera to capture video.</string>
<key>NSMicrophoneUsageDescription</key>
<string>CallingApp needs to access your mic to capture audio.</string>
```

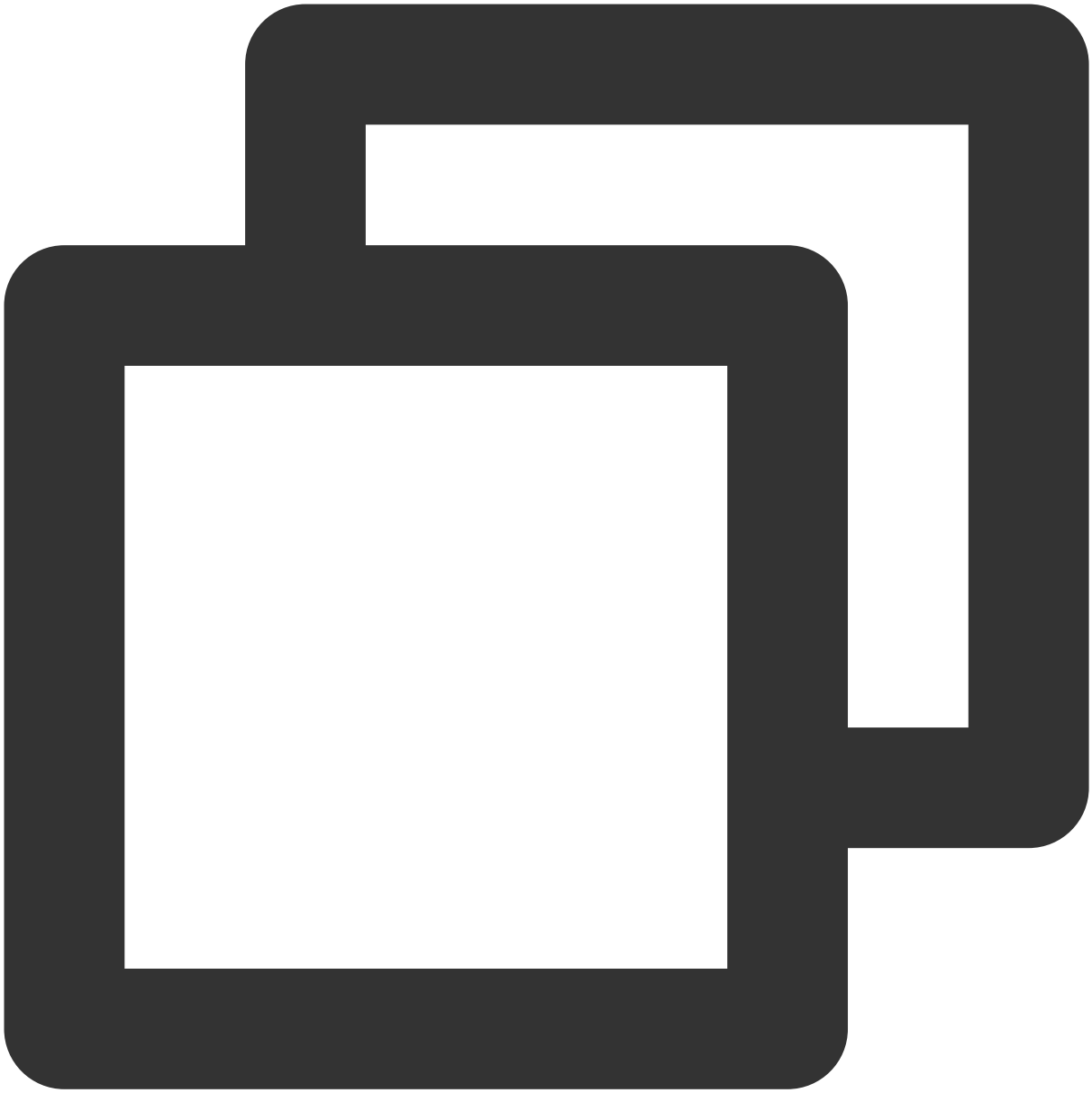



## Step 4. Log in to the `TUICallKit` component

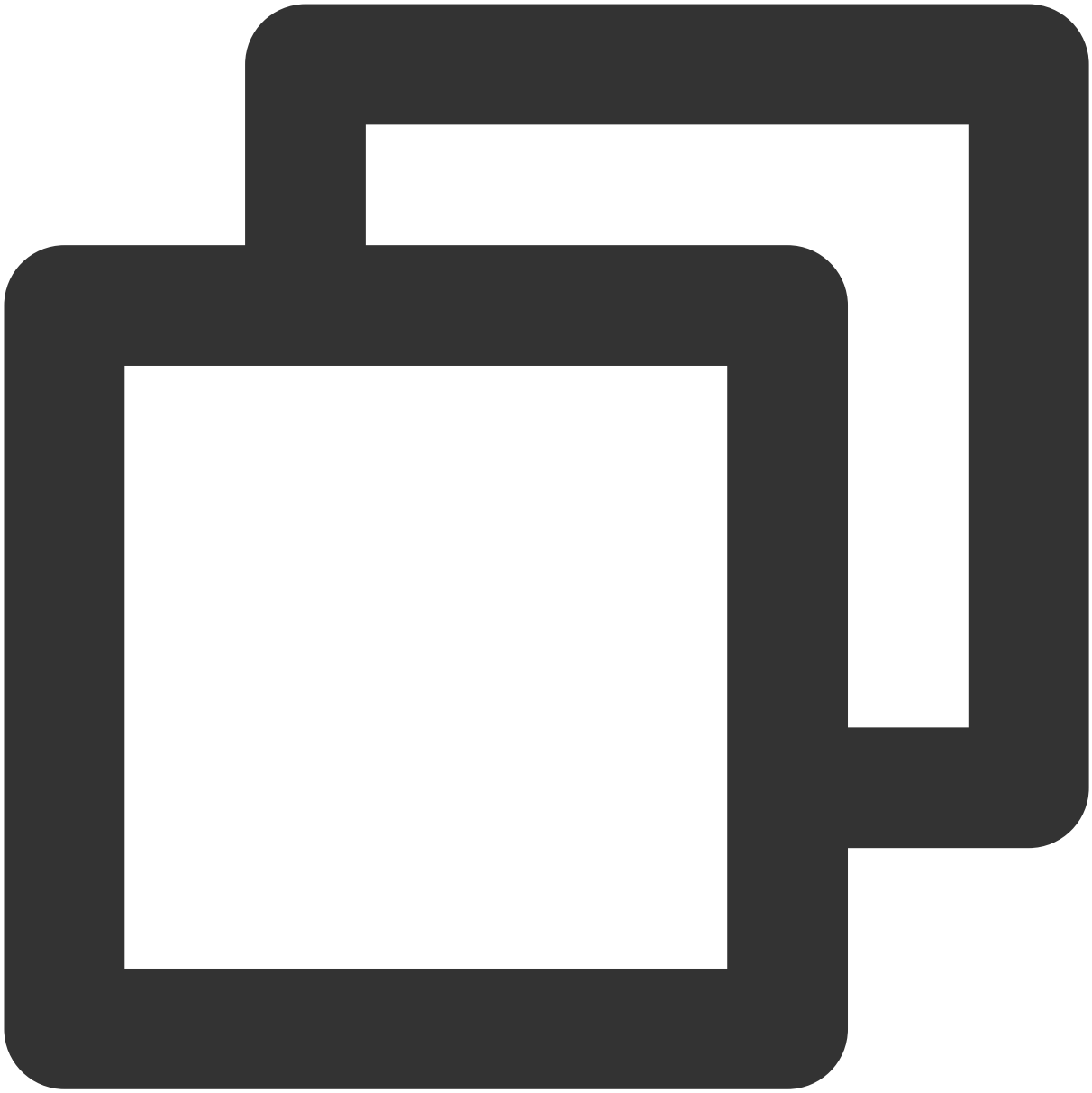
Add the following code to your project to call the relevant APIs in `TUICore` to log in to the `TUICallKit` component. This step is very important, as the user can use the component features properly only after a successful login. Carefully check whether the relevant parameters are correctly configured:

Swift

Objective-C



```
import TUICore

TUILogin.login(1400000001,                    // Replace it with the `SDKAppID` obta
            userID: "denny",                  // Replace it with your `UserID`.
            userSig: "xxxxxxxxxxx") {         // You can calculate a `UserSig` in th
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \\(code), error: \\(message ?? "nil")")
}
```



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

[TUILogin login:1400000001           // Replace it with the `SDKAppID` obtained in
         userID:@"denny"             // Replace it with your `UserID`.
        userSig:@"xxxxxxxxxxx"       // You can calculate a `UserSig` in the consol
           succ:^{
    NSLog(@"login success");
} fail:^(int code, NSString *msg) {
    NSLog(@"login failed, code: %d, error: %@", code, msg);
}
```

**Parameter description:** The key parameters used by the `login` function are as detailed below:

**SDKAppID**: Obtained in the last step in step 1 and not detailed here.

**UserID**: The ID of the current user, which is a string that can contain only letters (a–z and A–Z), digits (0–9), hyphens (-), or underscores (_).

**UserSig**: The authentication credential used by Tencent Cloud to verify whether the current user is allowed to use the TRTC service. You can get it by using the `SDKSecretKey` to encrypt the information such as `SDKAppID` and `UserID`. You can generate a temporary `UserSig` by clicking the UserSig Generate button in the console. For more information, see UserSig.

**Note:**

**Many developers have contacted us with many questions regarding this step. Below are some of the frequently encountered problems:**

SDKAppID is invalid.

UserSig is set to the value of Secretkey mistakenly. The UserSig is generated by using the SecretKey for the purpose of encrypting information such as SDKAppID, UserID, and the expiration time. But the value of the UserSig that is required cannot be directly substituted with the value of the SecretKey.

UserID is set to a simple string such as 1, 123, or 111, and your colleague may be using the same userId while working on a project simultaneously. In this case, login will fail as TRTC doesn't support login on multiple terminals with the same UserID. Therefore, we recommend you use some distinguishable userId values during debugging.

The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current integration process more quickly. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` subsequently. Therefore, we strongly recommend you run the `UserSig` calculation logic on the server and make the application request the `UserSig` calculated in real time every time the application uses the `TUICallKit` component from the server.

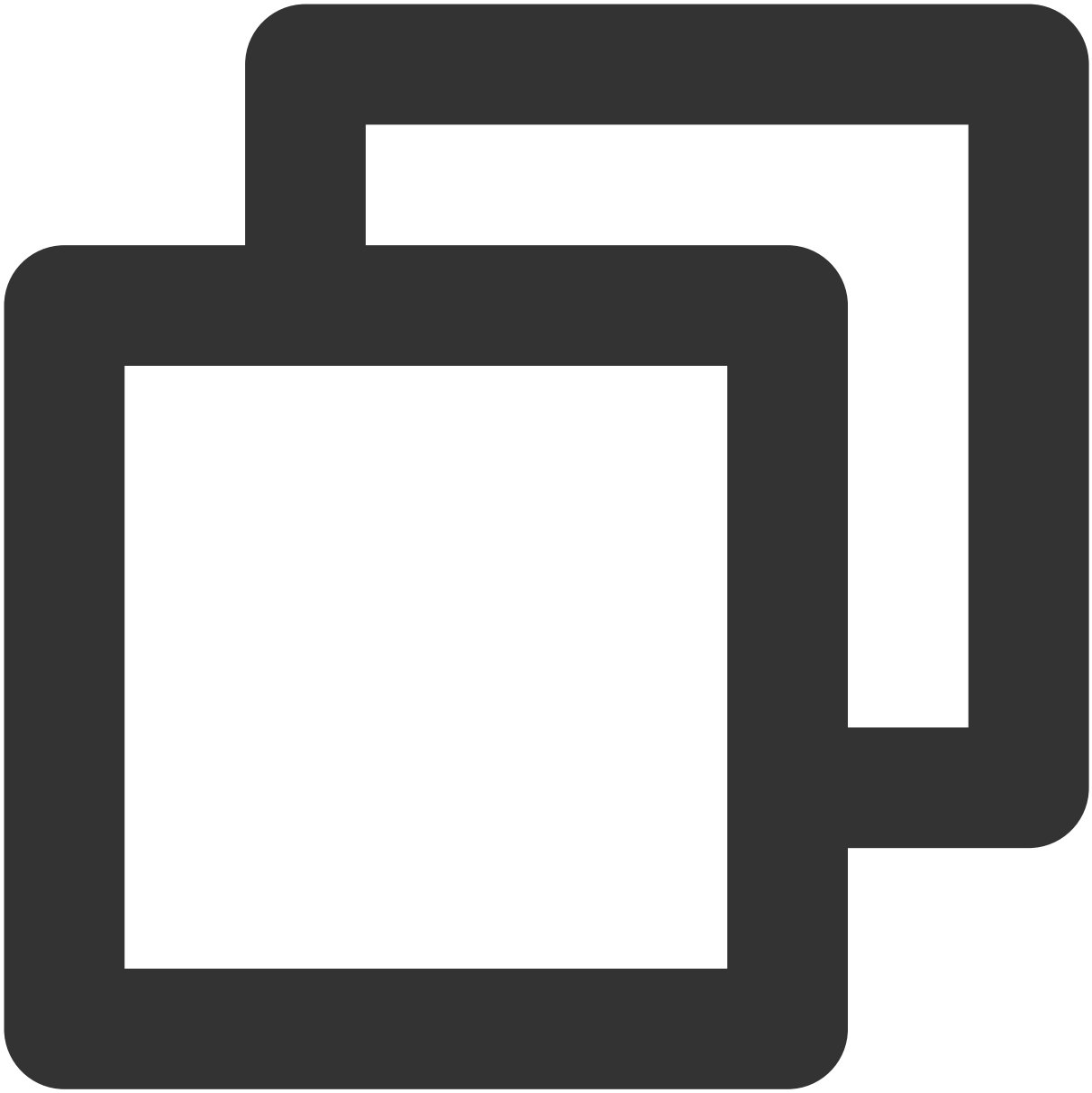
# Step 5. Make a call

### One-to-one video call

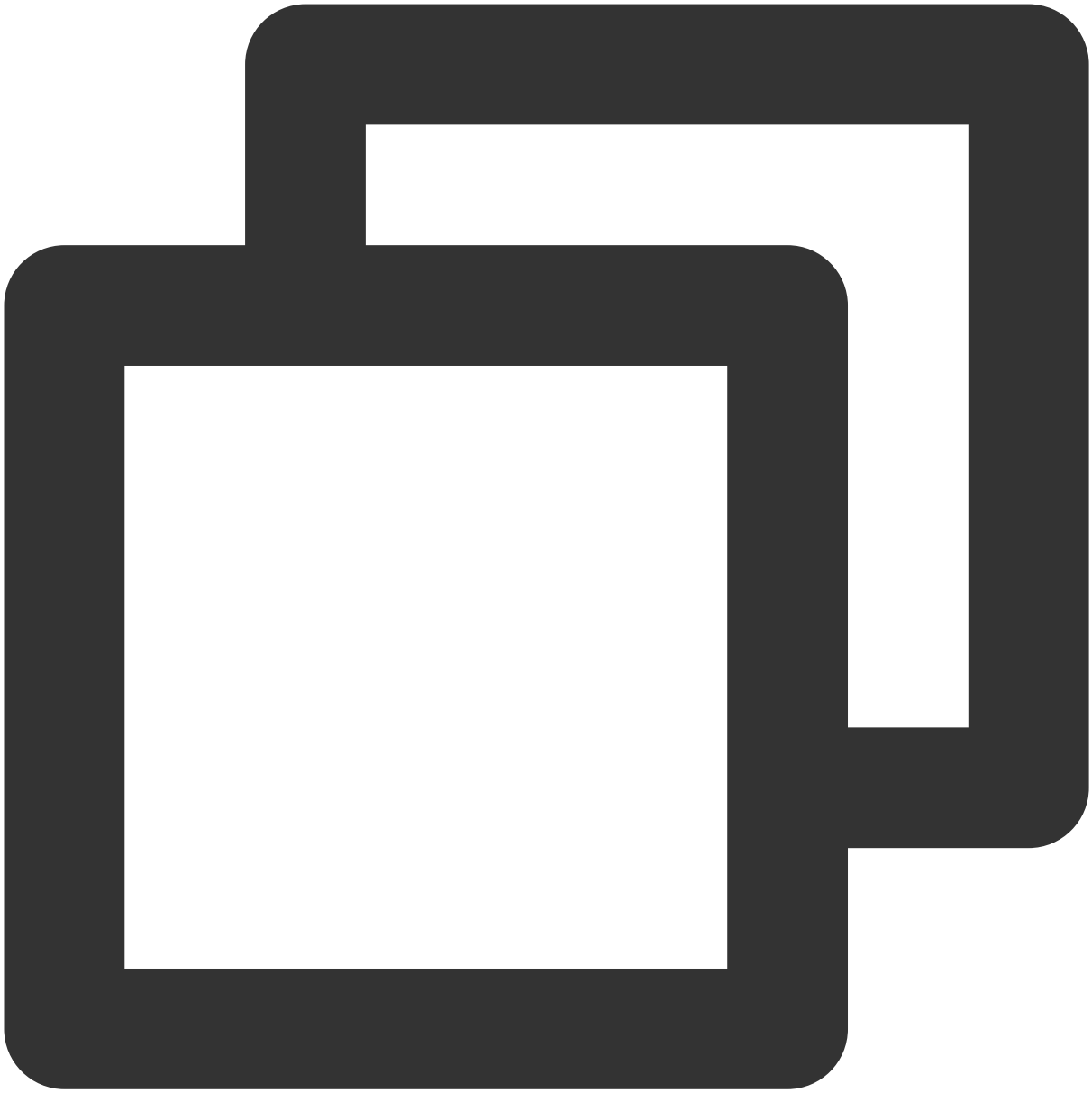
You can call the `call` function of `TUICallKit` and specify the call type and the callee's `userId` to make an audio/video call.

Swift

Objective-C



```
import TUICallKit

// Make a one-to-one video call. Suppose the `userId` is `mike`.
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video)
```



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

// Make a one-to-one video call. Suppose the `userId` is `mike`.
[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```

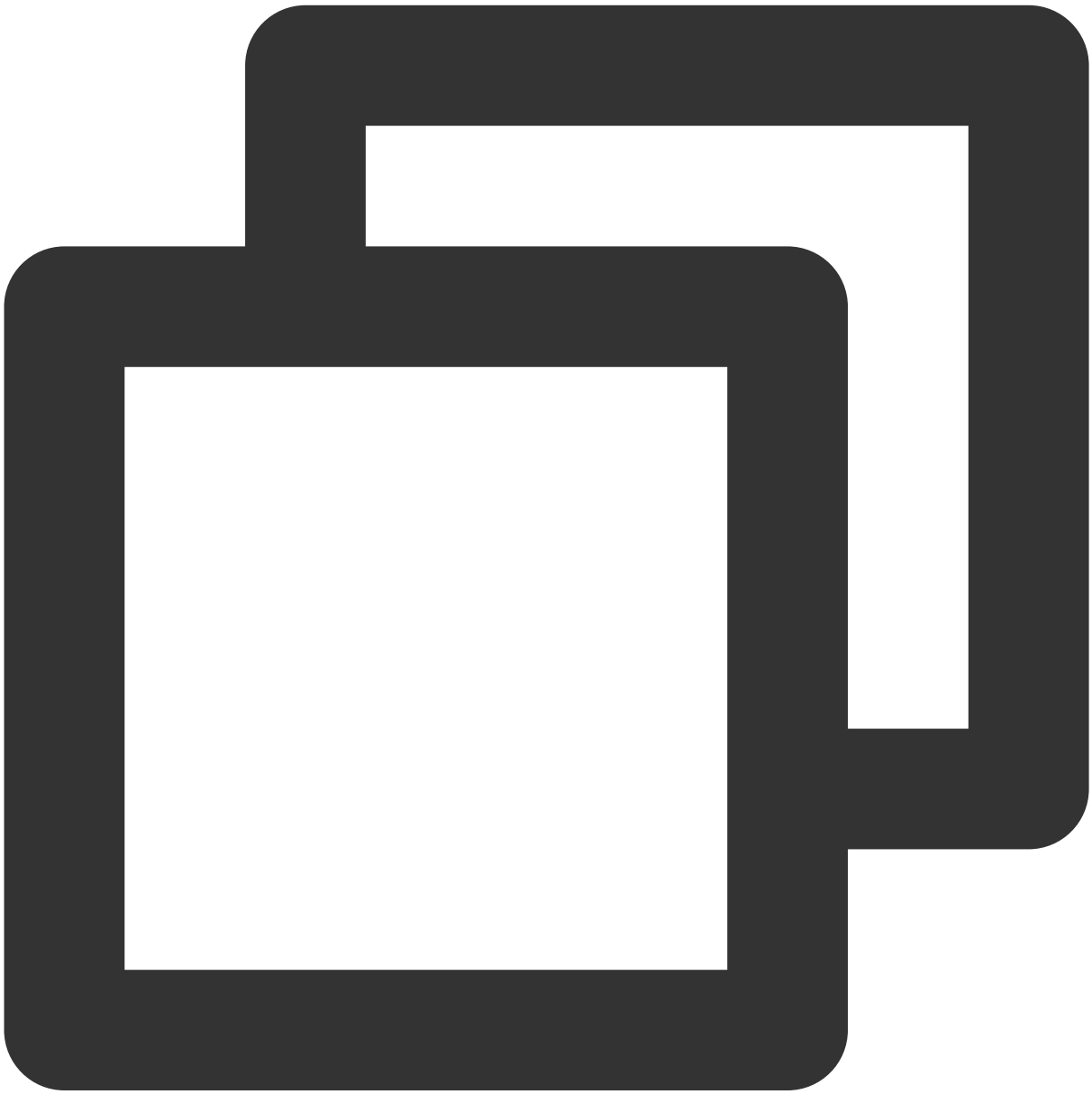
| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the target user, such as `"mike"`. |
| callMediaType | TUICallMediaType | The call media type, such as `TUICallMediaTypeVideo`. |

### Group video call

You can call the `groupCall` function of `TUICallKit` and specify the call type and the list of callees' `UserID` values to make a group audio/video call.

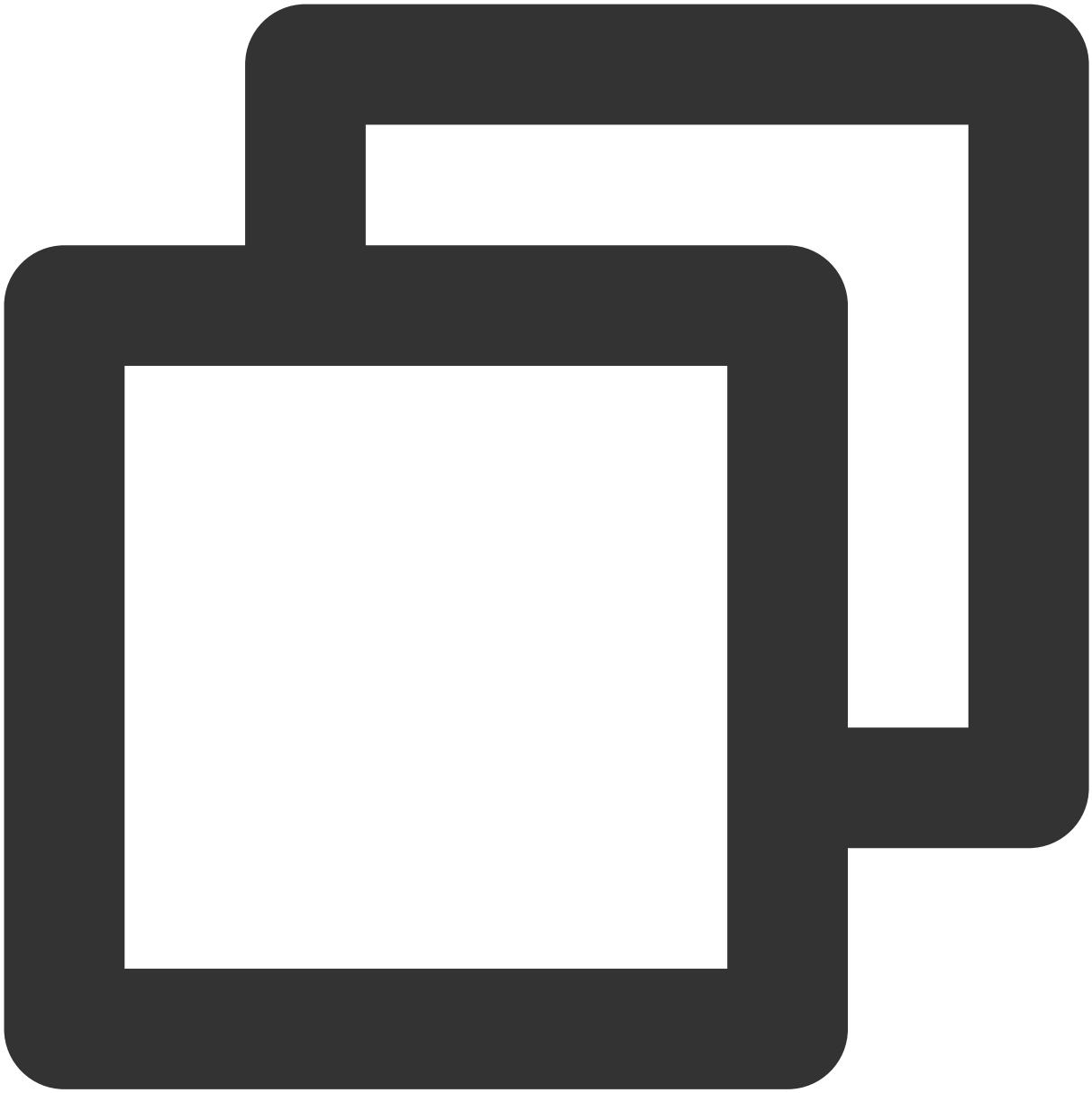
Swift

Objective-C



```
import TUICallKit

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

```
                                        callMediaType: .video)
```



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

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

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID, such as `"12345678"`. |

| | | |
|---|---|---|
| userIdList | Array | The list of `userId` values of the target users, such as `["denny", "mike", "tommy"]` |
| callMediaType | TUICallMediaType | The call media type, such as `TUICallMediaTypeVideo`. |

**Note**

You can create a group as instructed in Android, iOS, and macOS. You can also use Chat TUIKit to integrate chat and call scenarios at one stop.

`TUICallKit` currently doesn't support making a group video call among users not in a group. If you have such a need, please contact info_rtc@tencent.com.

# Step 6. Answer a call

After **step 4**, when receiving an incoming call, the `TUICallKit` component will automatically display the call answering UI.

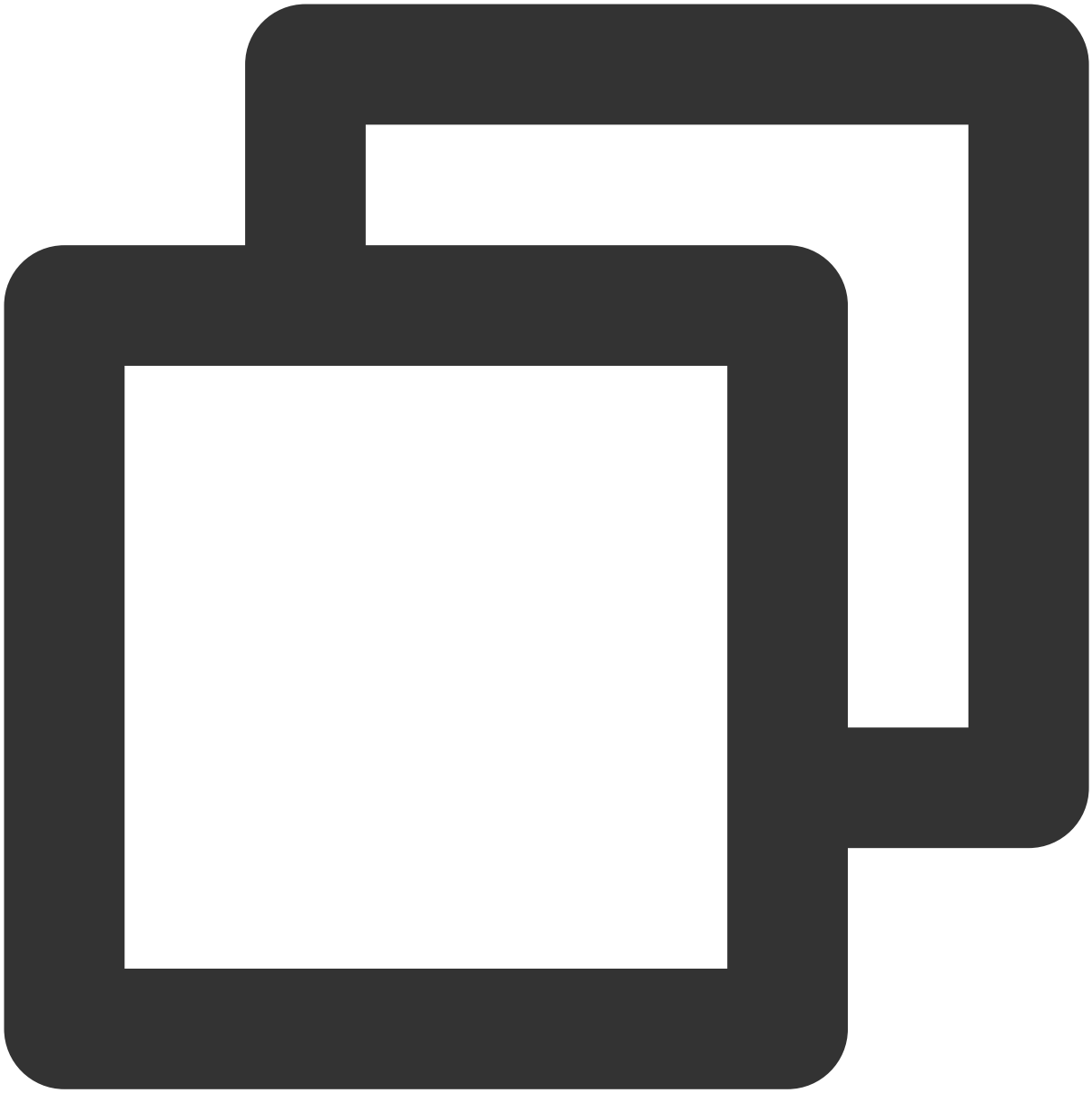
# Step 7. Implement more features

### 1. Nickname and profile photo settings

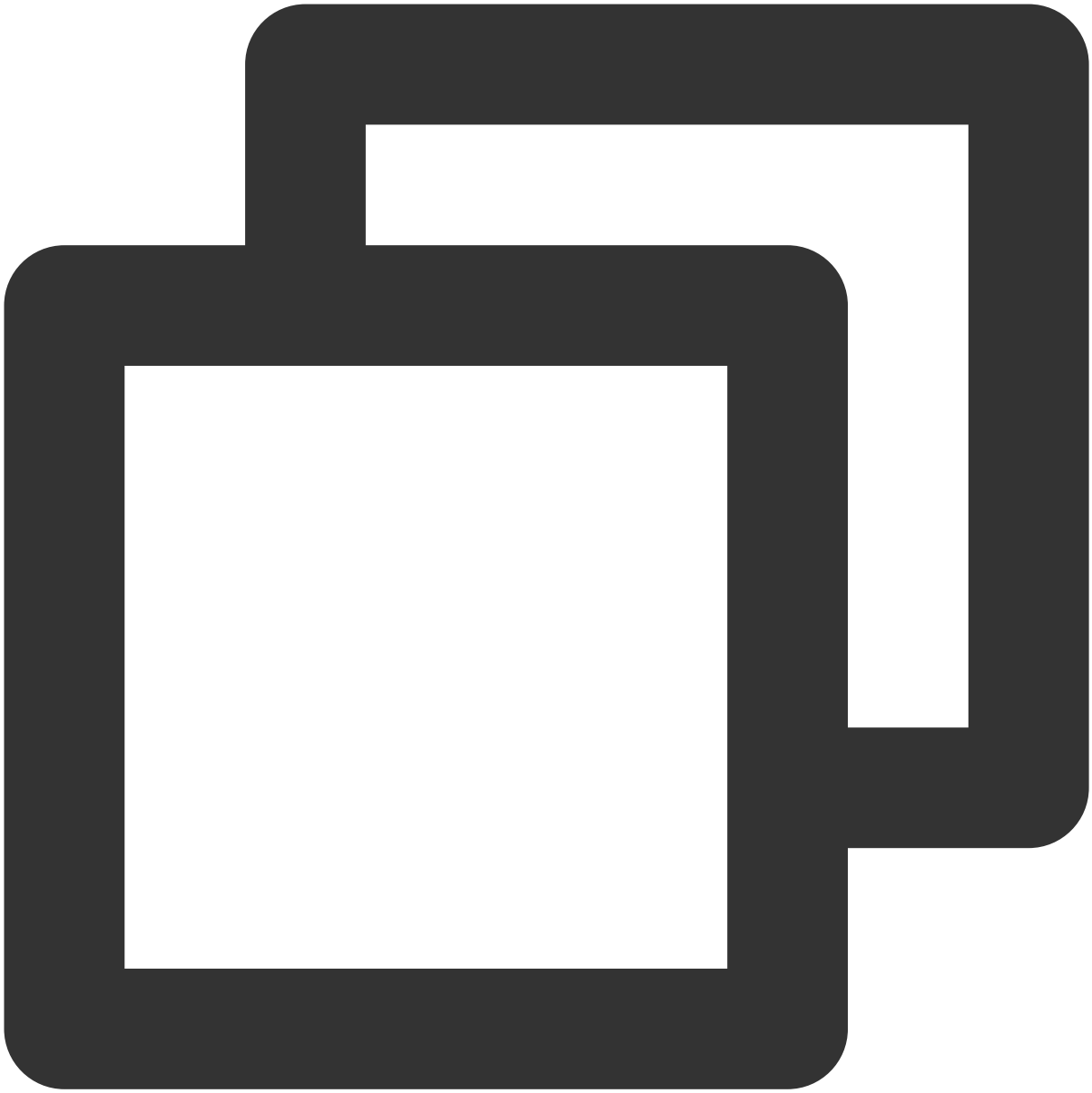
To customize the nickname or profile photo, use the following API for update:

Swift

Objective-C



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

} fail: { code, message in

}
```



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

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

}];
```

**Caution**

The update of the callee's nickname and profile photo may be delayed during a call between non-friend users due to the user privacy settings. After a call is made successfully, the information will also be updated properly in subsequent calls.

### 2. Offline call push

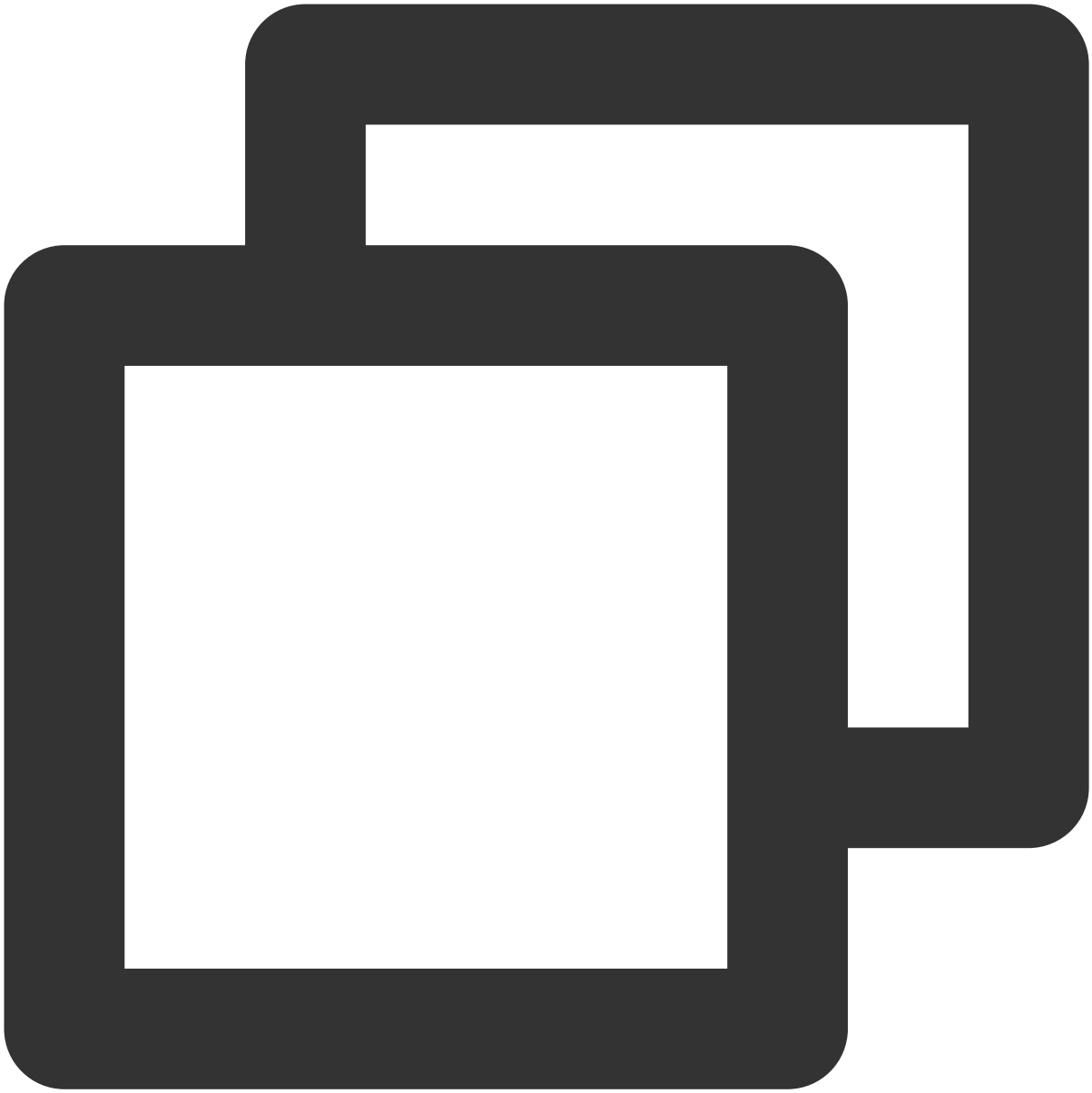
You can make/answer audio or video calls after completing the above steps. However, if you want your users to be able to receive call invitations even when your application is in the background or after it is closed, then you need to also implement the offline call push feature. For more information, see Offline Call Push (iOS).

### 3. Floating window

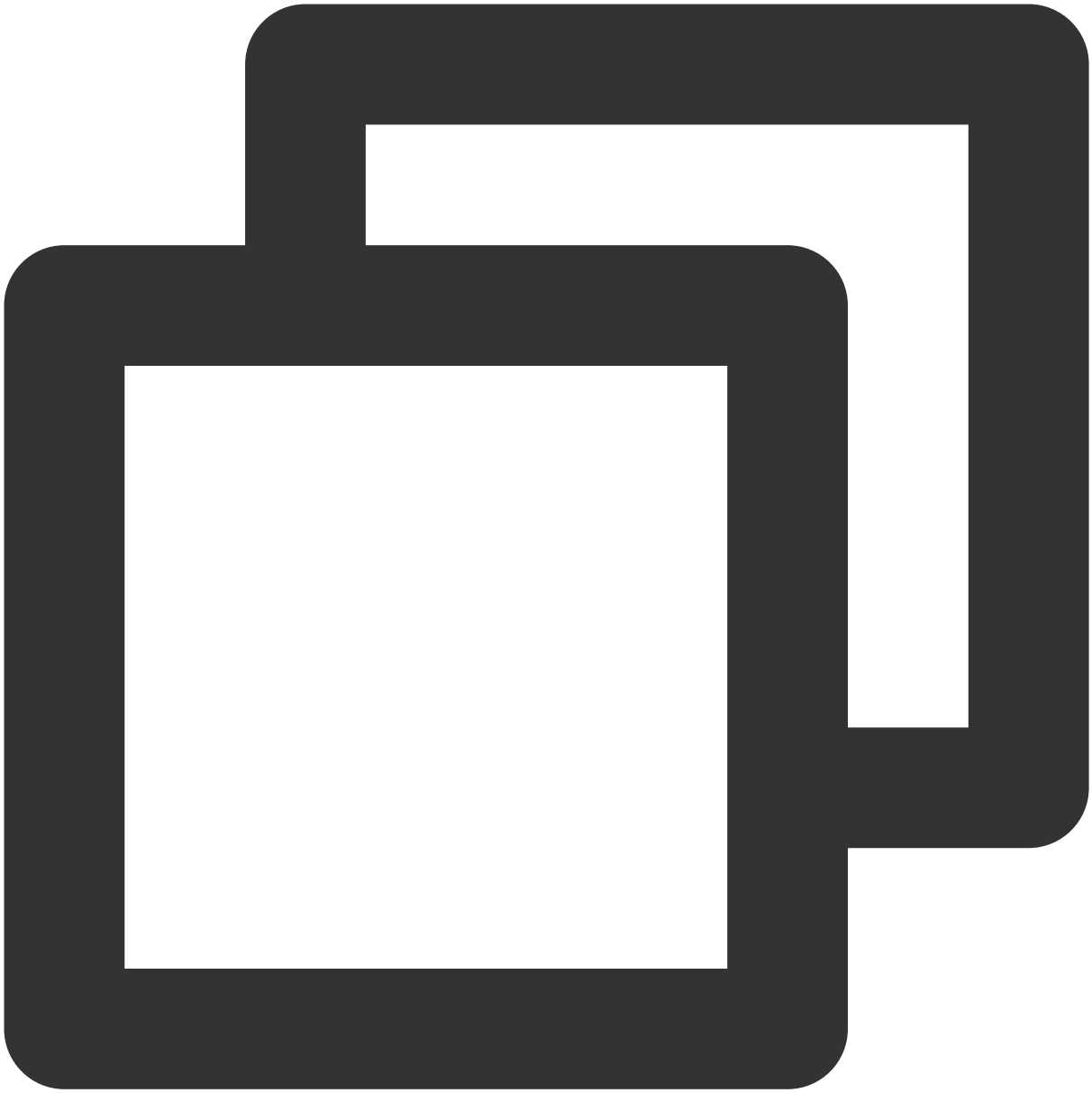
To implement the floating window feature in your application, call the following API when initializing the `TUICallKit` component:

Swift

Objective-C



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



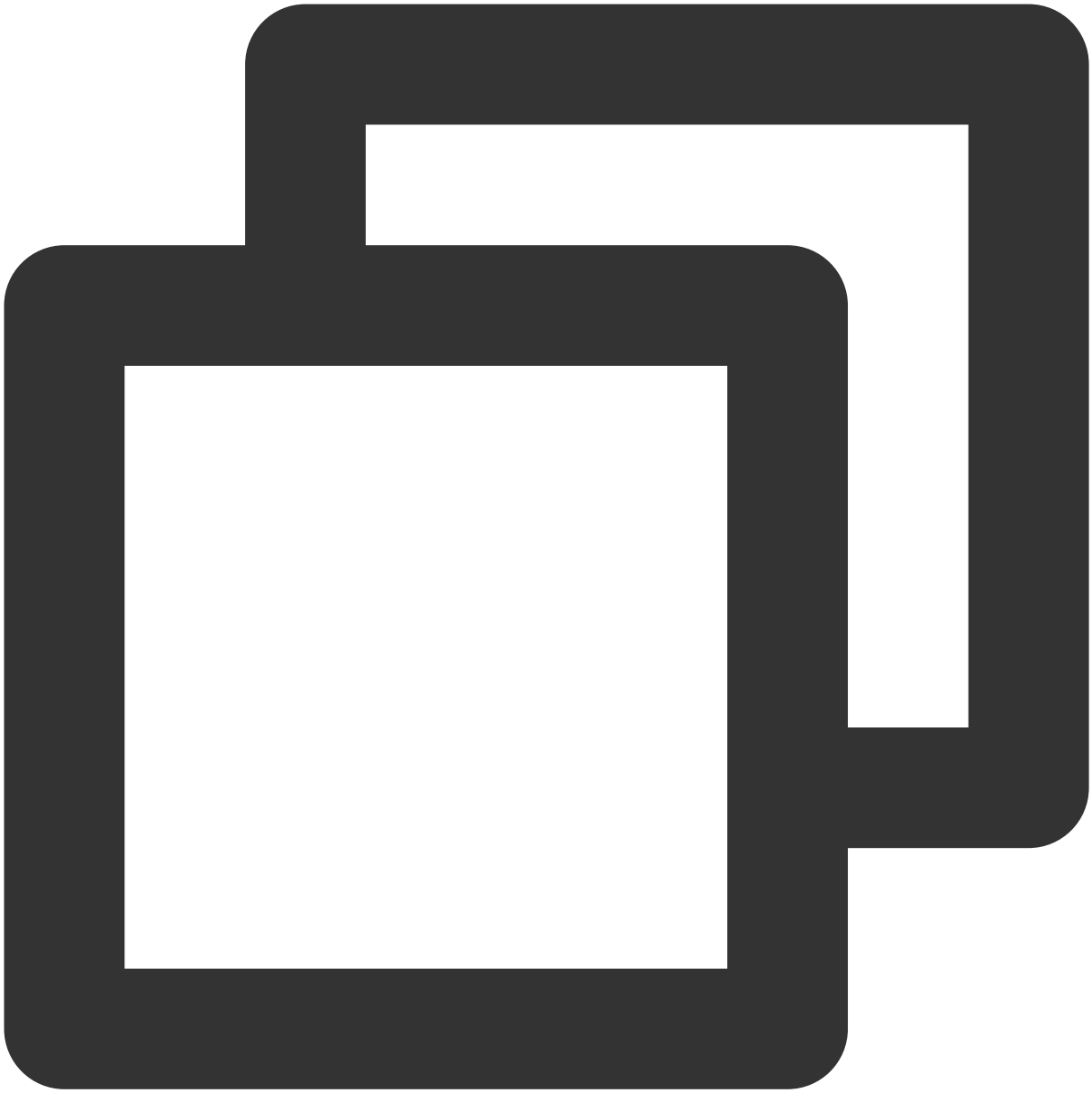
```
[[TUICallKit createInstance] enableFloatWindow:YES];
```

### 4. Listening on the call status

To **listen on the call status** (for example, the start or end of a call or the call quality during a call), listen on the following events:

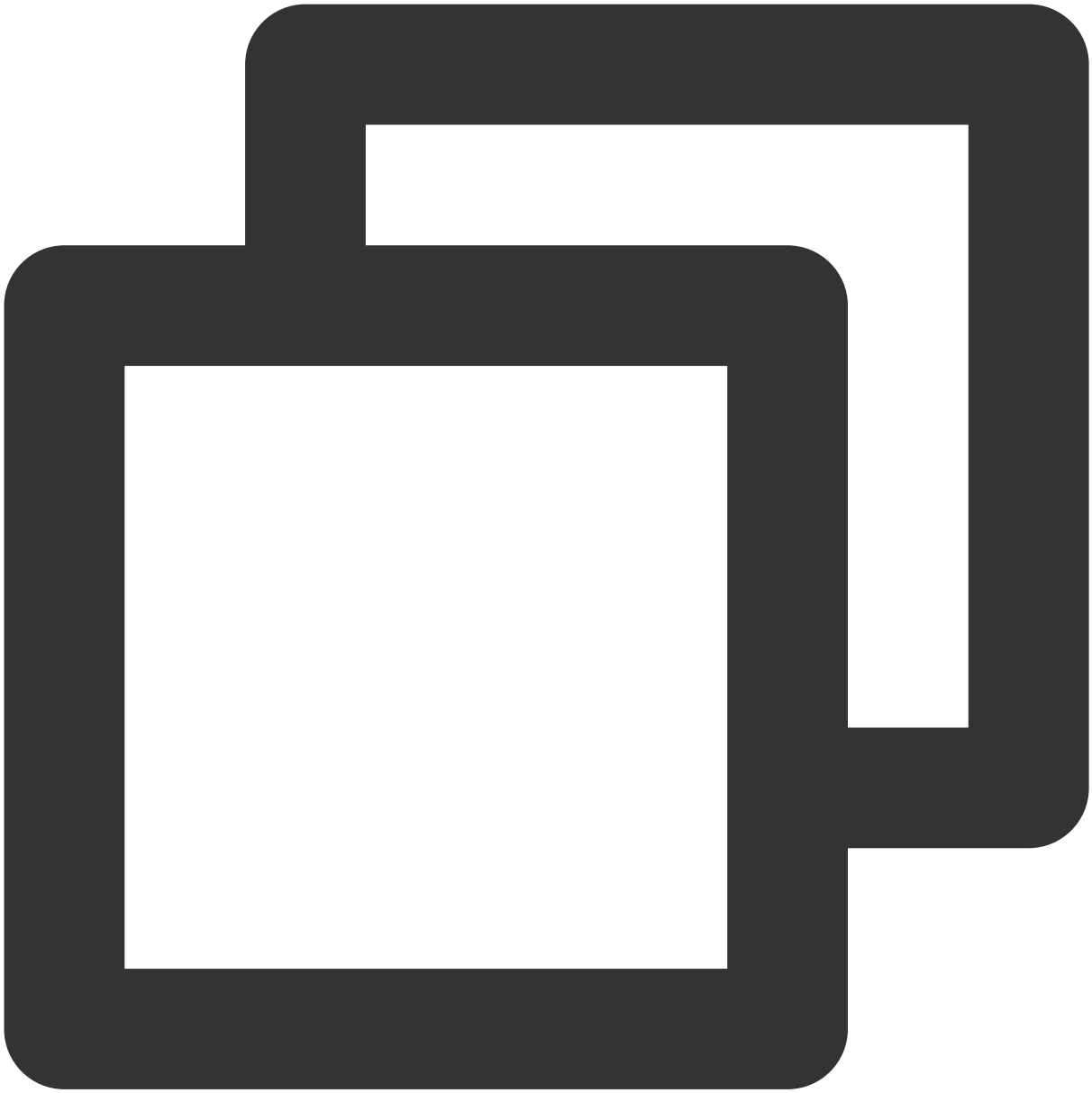
Swift

Objective-C



```
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
```

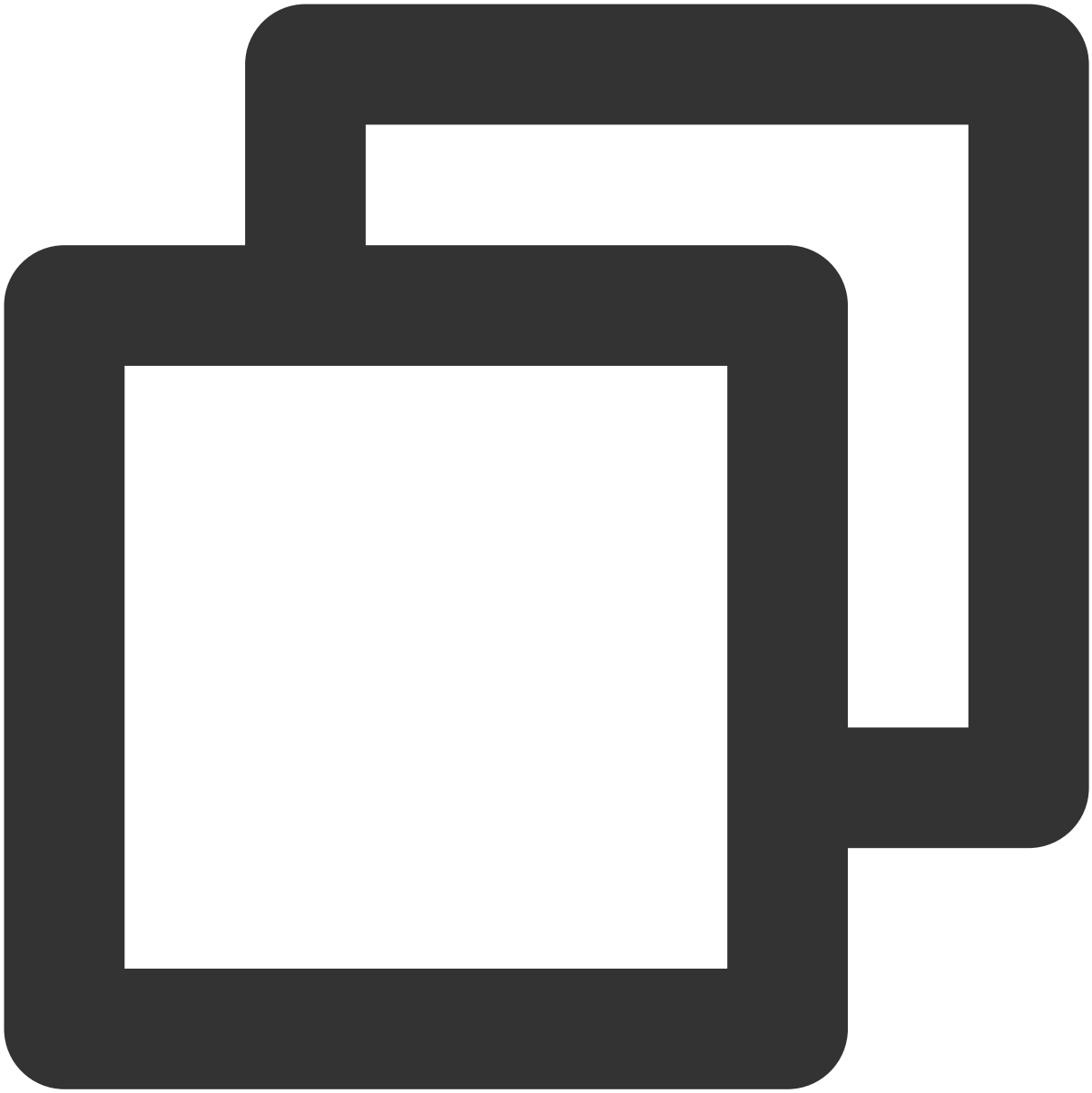
```
}
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual

}
```

### 5. Custom ringtone

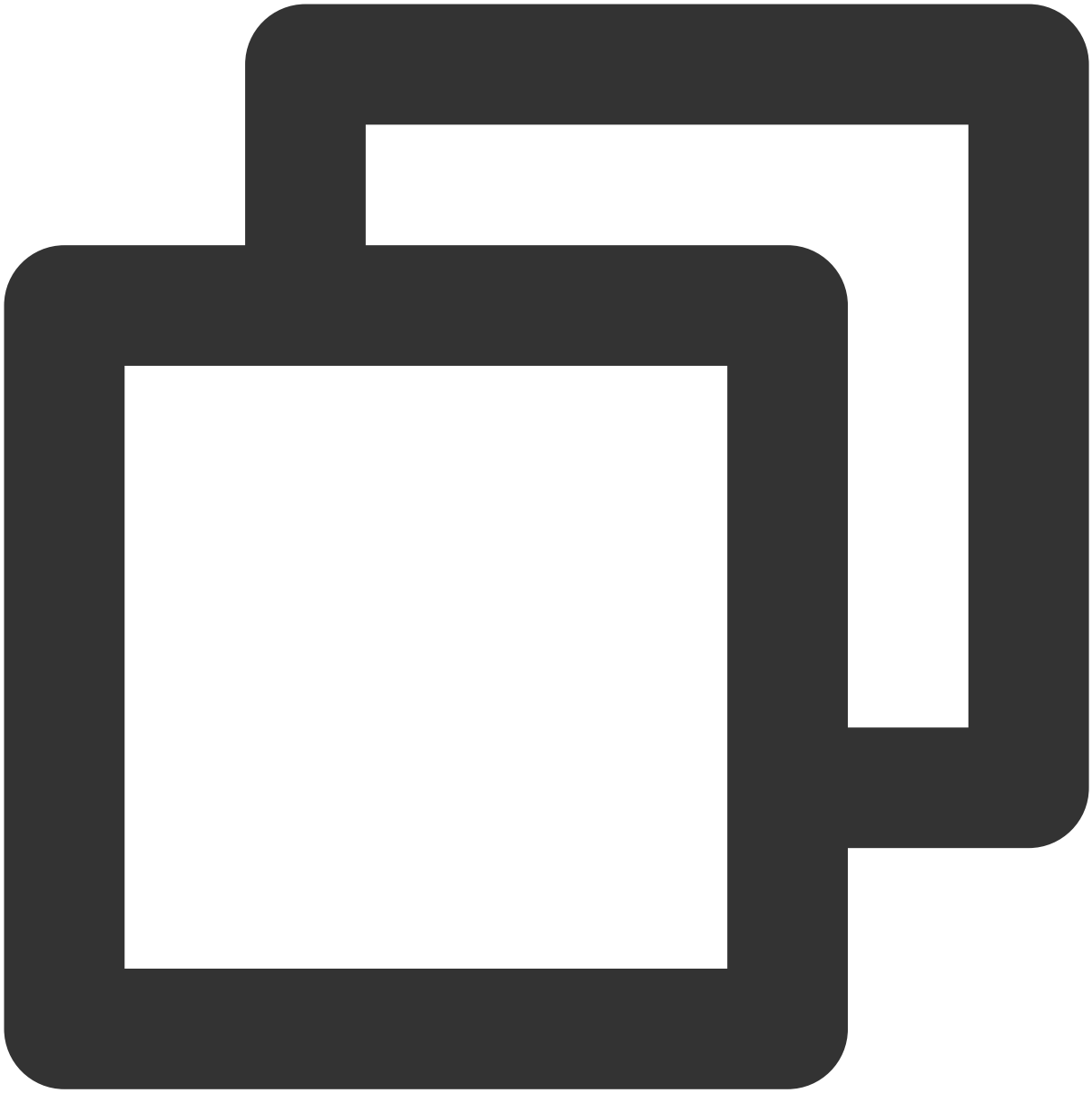
You can use the following API to customize the ringtone:

Swift

Objective-C



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



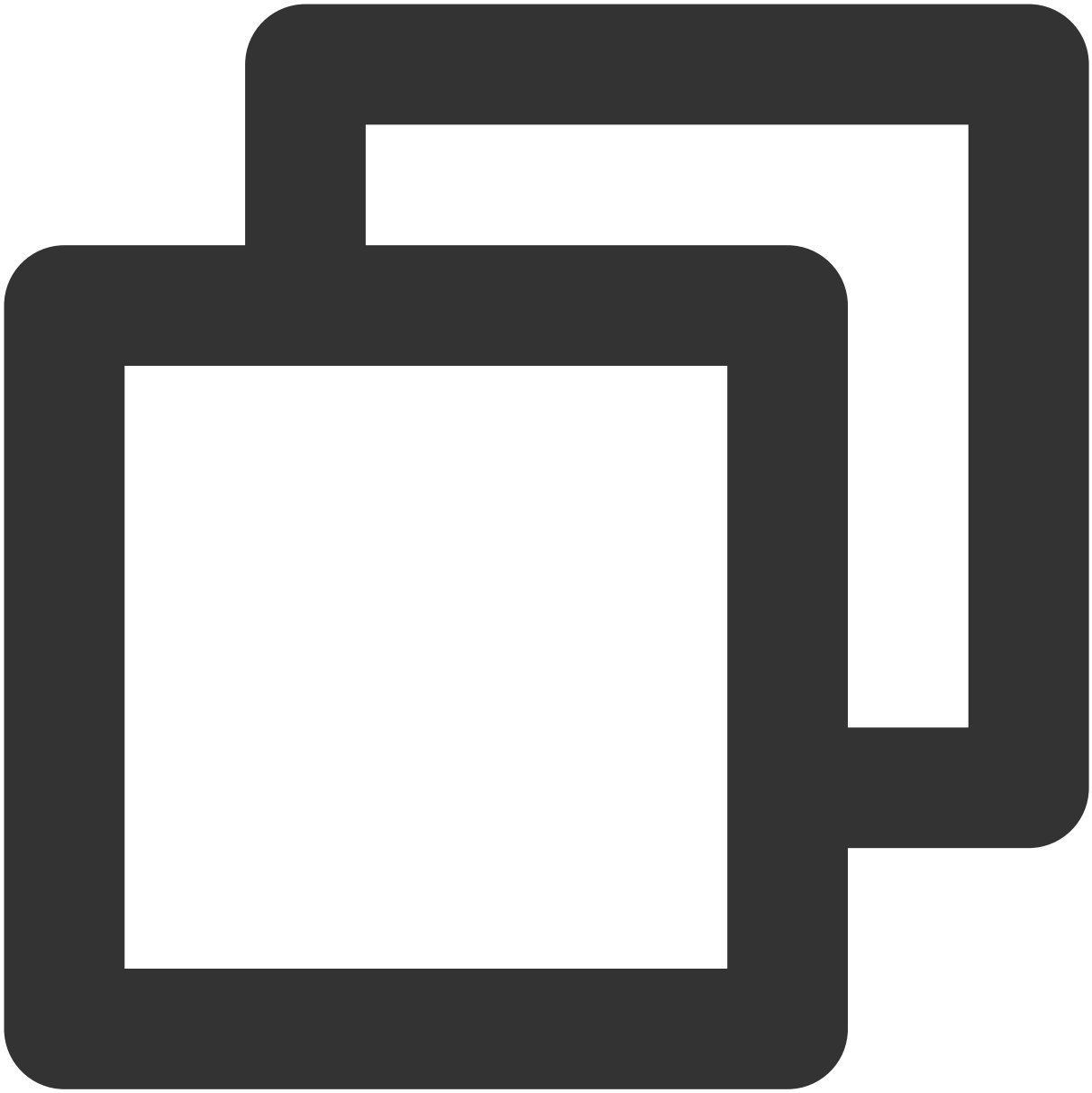
```
[[TUICallKit createInstance] setCallingBell:@" "];
```

### 6. Custom call timeout period

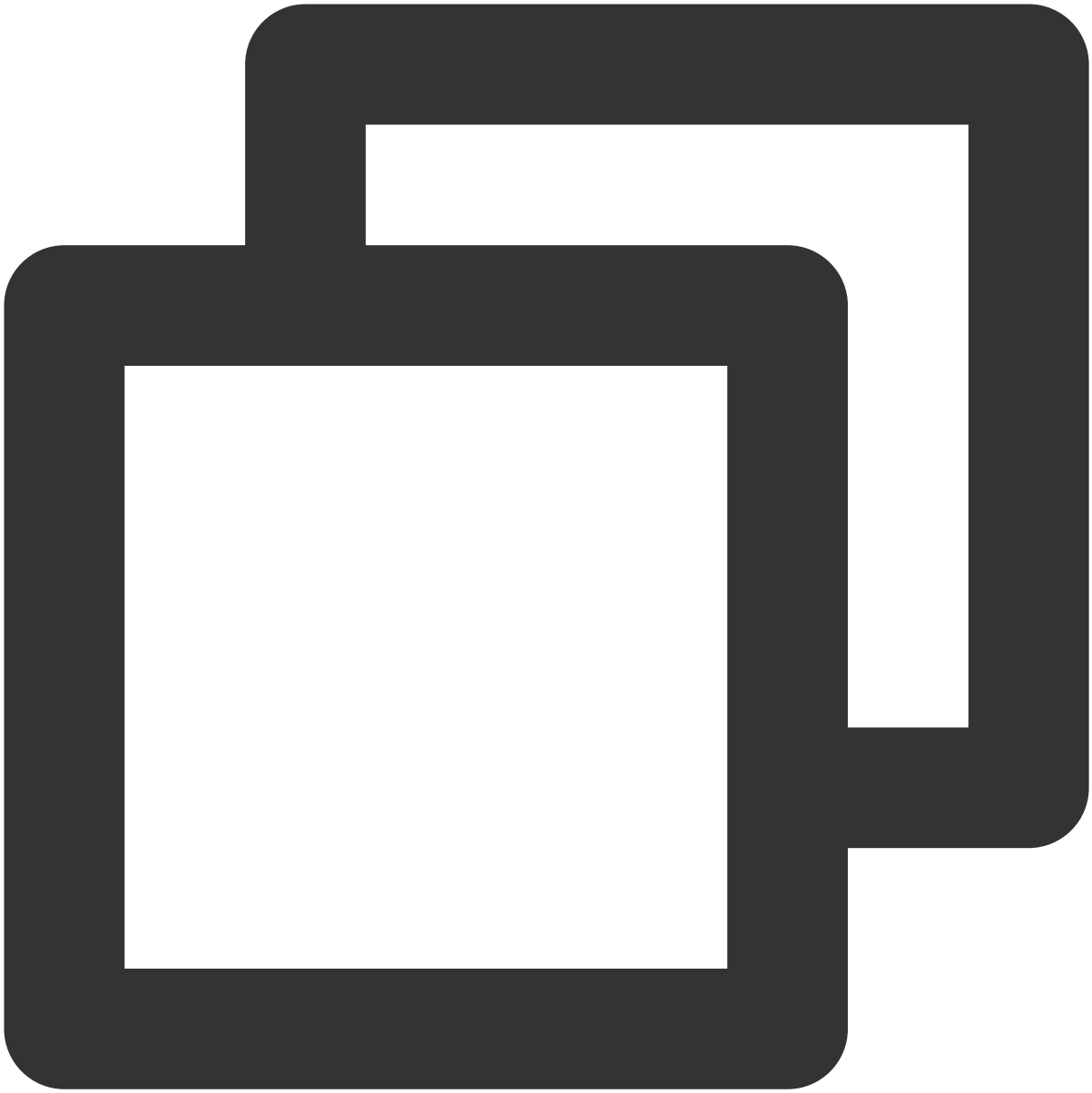
You can set callParams when invoke call, groupCall,inviteUser to change your call timeout period.

Swift

Objective-C



```
let callParams = TUICallParams()
let offlinePushInfo = TUIOfflinePushInfo()
callParams.offlinePushInfo = offlinePushInfo
callParams.timeout = 30
callParams.userData = "user data"
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video, params: cal

} fail: { code, message in

}
```



```
TUICallParams *callParams = [TUICallParams new];
callParams.offlinePushInfo = [TUIOfflinePushInfo new];
callParams.timeout = 30;
callParams.userData = @"user data";
[[TUICallKit createInstance] call:@"mike"
                    callMediaType:TUICallMediaTypeVideo
                           params:callParams
                             succ:^{

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

```
}];
```

# FAQs

### 1、What should I do if I receive the error message "The package you purchased does not support this ability"?

The error message indicates that your application's audio/video call capability package has expired or is not activated. You can claim or activate the audio/video call capability as instructed in step 1 to continue using `TUICallKit`.

### 2、How to purchase official version？

please refer to Purchase Official Version.

# Suggestions and Feedback

If you have any suggestions or feedback, please contact info_rtc@tencent.com.

# Web

Last updated：2023-06-07 15:08:40

# Integrating TUICallKit

This document shows you how to quickly integrate the `TUICallKit` component to build a video call application with ready-made UI.

To try out the component, see TUICallKit basic demo.

Before you start, check whether your desktop browser supports TRTC. For details, see **Environment requirements** below.

## Contents



## Step 1. Activate the service

**TUICallKit** is an audio & video communication component built upon Tencent Cloud's paid PaaS services, **Chat** and **Real-time Communication (TRTC)** . In order for you to better experience the Call feature, we offer a 7-day free trial version for each SDKAppID (Note: no additional call duration is provided with the free trial version). Each SDKAppID can apply the free trial version twice, with each trial lasting for 7 days; at the same time, the total number of trial opportunities for all SDKAppID under one account (UIN) is 10.

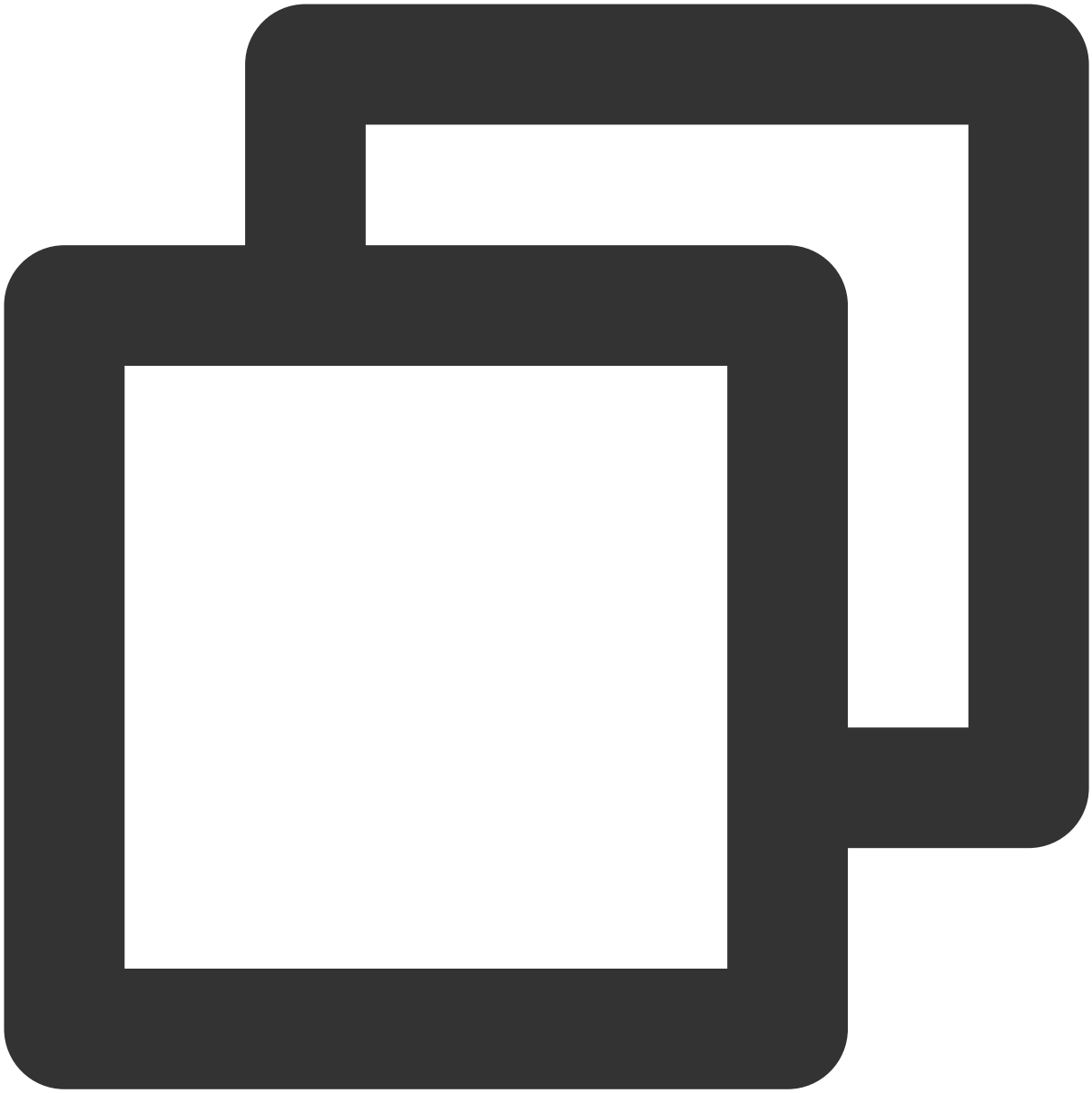
You can activate the Call free trial version in the Chat console, with the specific operation steps as follows:

1. Log into the Chat console, select the data center, and create a new application. Skip this step if you already have an application.
2. Click on the target application card to enter the application's basic configuration page.
3. Locate the Call card, and click on "Free Trial".
4. After confirming the content in the popup window, click on "**Activate now**". Once activated, you can proceed with integration according to this document.
5. If you need to purchase the official version for your business to go live, you can proceed to the console to make the purchase. Please refer to Purchase Official Version.

# Step 2: Integrate TUICallKit

### Method 1: Integrate using npm

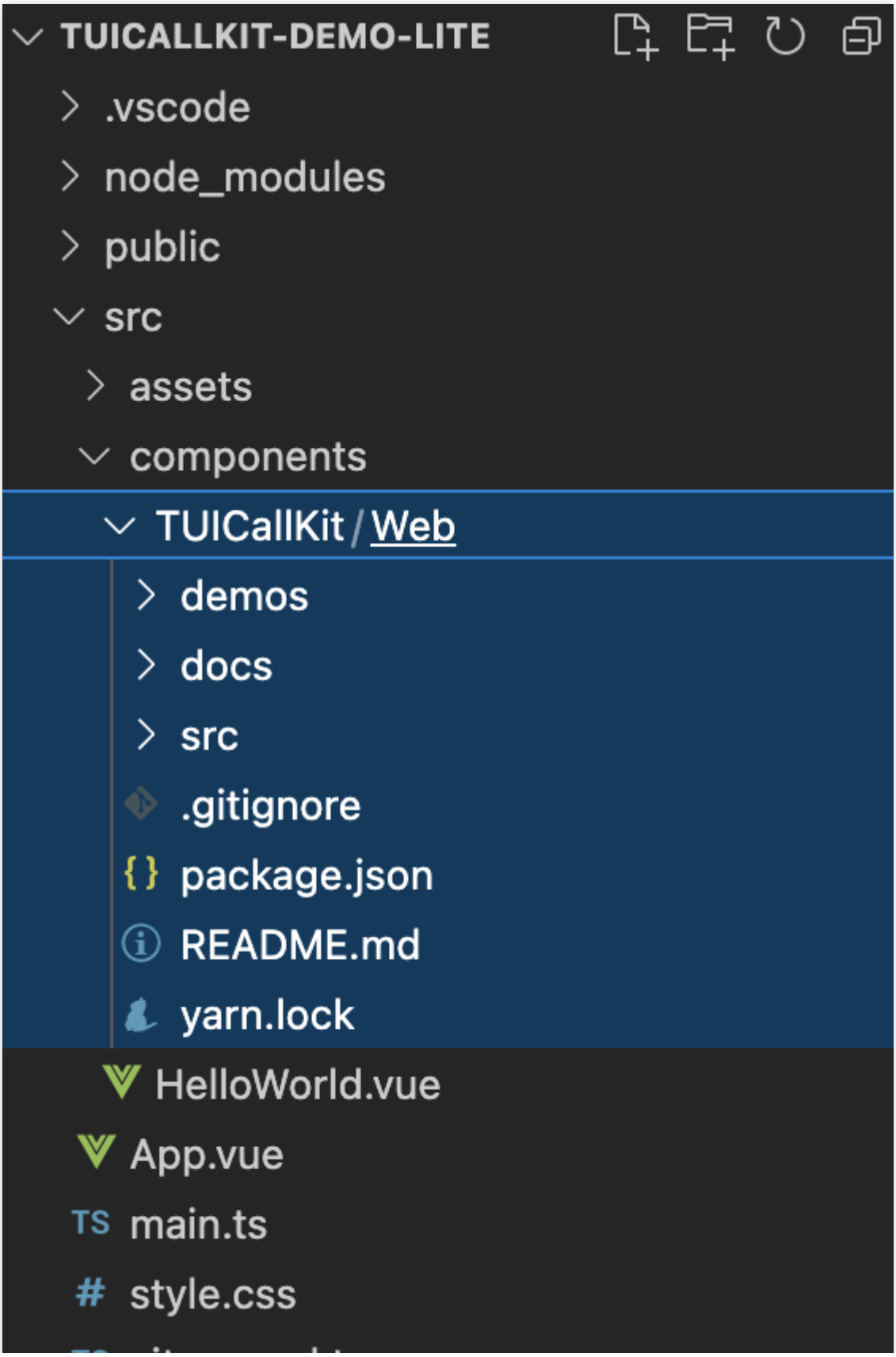
Download the `TUICallKit` component using npm. In case you need to scale your business in the future, we recommend you copy `TUICallKit` to the `src/components/` directory of your project.



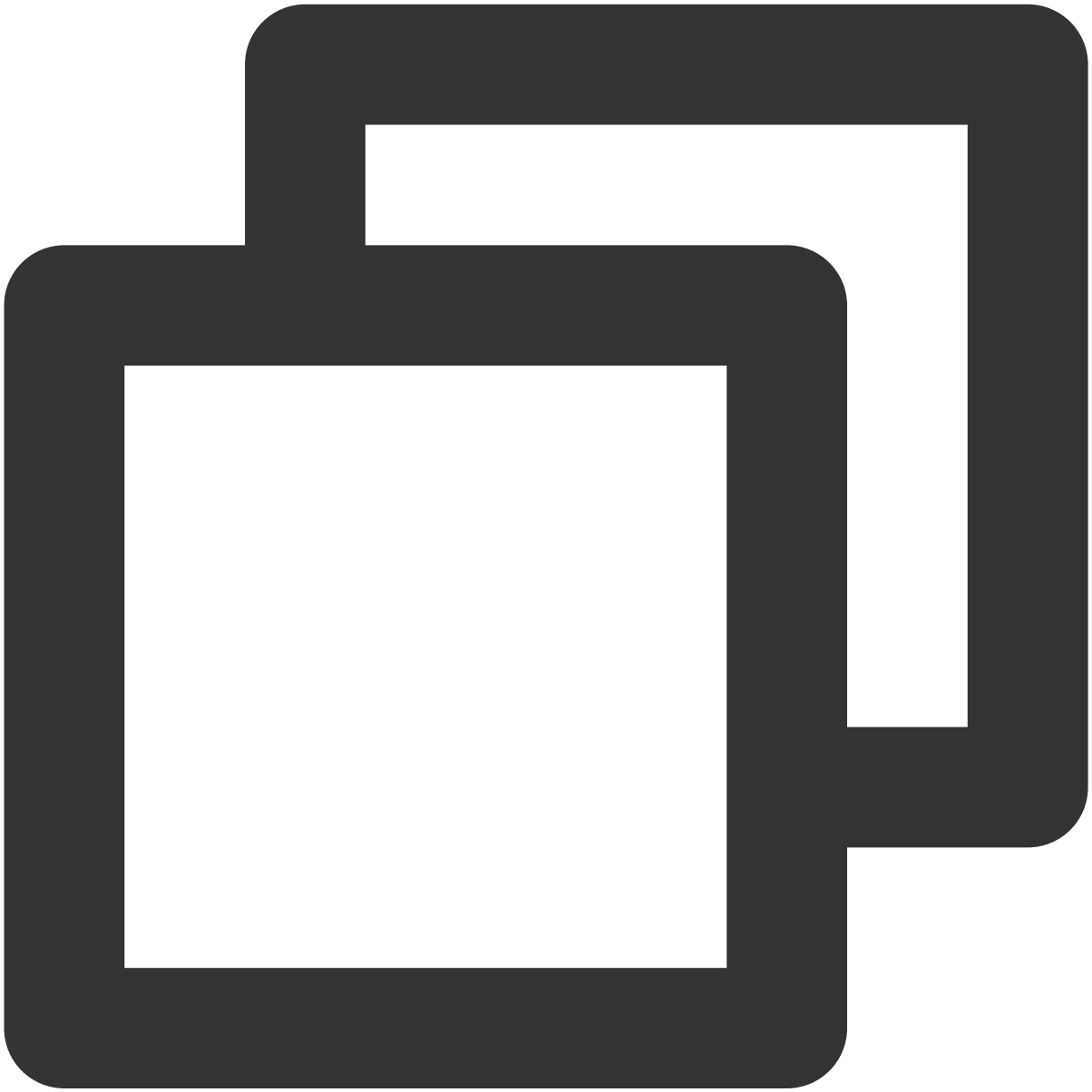
```
# macOS
yarn add @tencentcloud/call-uikit-vue  # If you haven't installed Yarn, execute `np
mkdir -p ./src/components/TUICallKit/Web && cp -r ./node_modules/@tencentcloud/call

# Windows
yarn add @tencentcloud/call-uikit-vue  # If you haven't installed Yarn, execute `np
xcopy .\\node_modules\\@tencentcloud\\call-uikit-vue  .\\src\\components\\TUICallKi
```

### Method 2: Integrate the source code

1. Download the `TUICallKit` source code from GitHub and copy the `TUICallKit/Web` folder to the `src/components` folder of your project.




2. Go to the above folder and install the required dependencies.

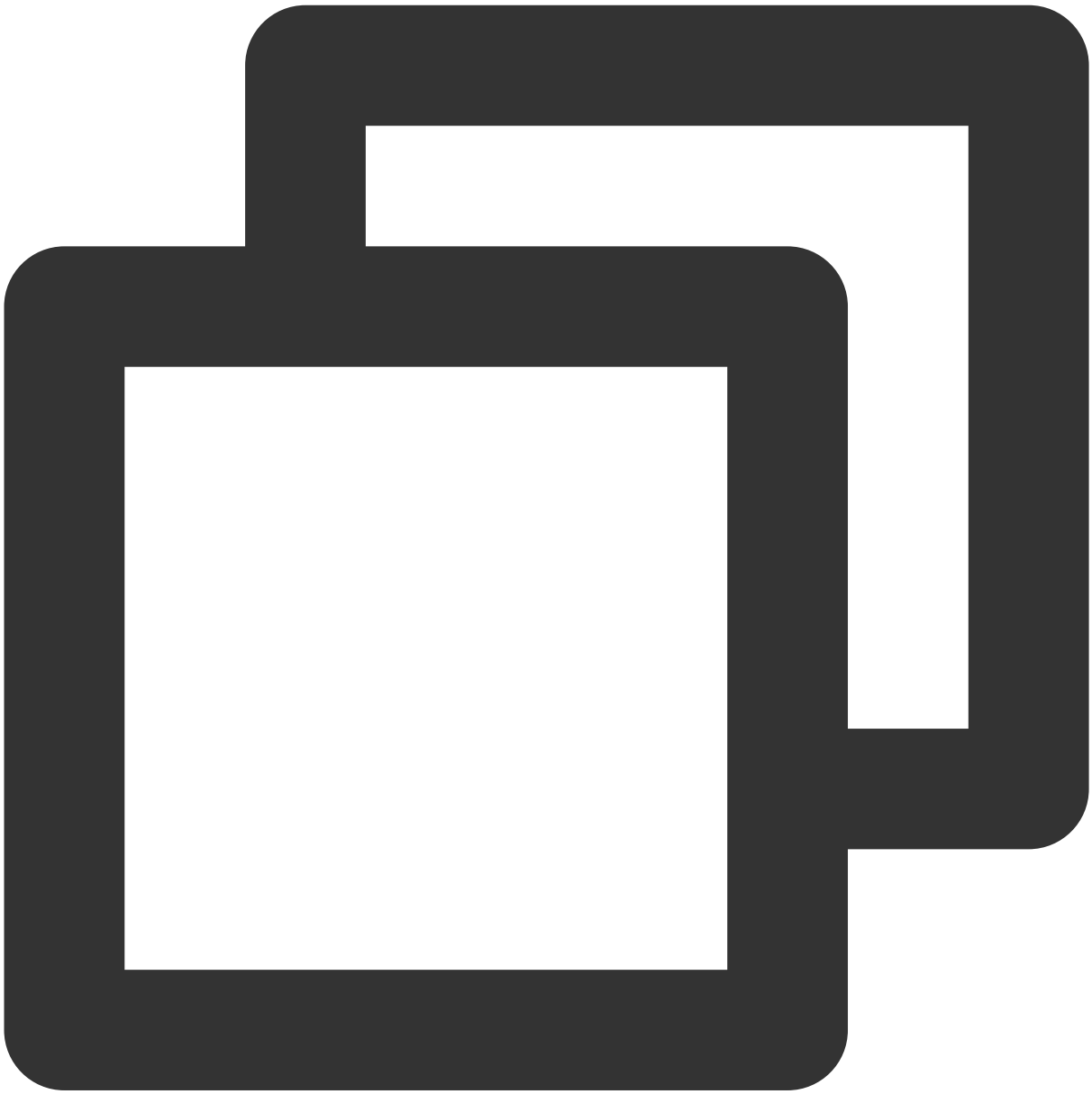


```
cd ./src/components/TUICallKit/Web
yarn         # If you haven't installed Yarn, execute `npm install -g yarn` first.
```

## Step 3. Call `TUICallKit`

Call `TUICallKit` on a page to show the call view.

1. Import the UI. Below is an example:

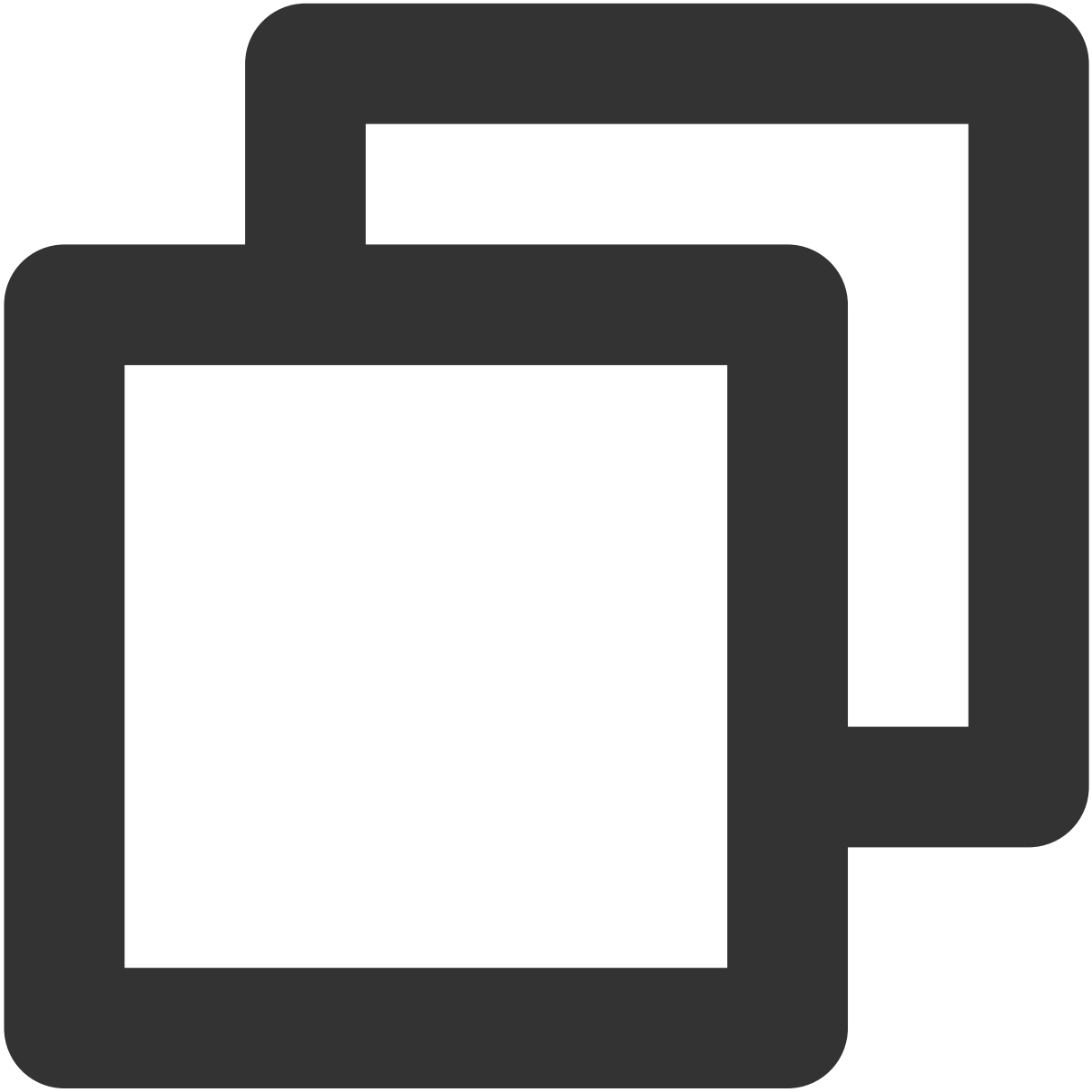


```
<script lang="ts" setup>
import { TUICallKit } from "./components/TUICallKit/Web";
</script>

<template>
```

```
  <TUICallKit />
</template>
```

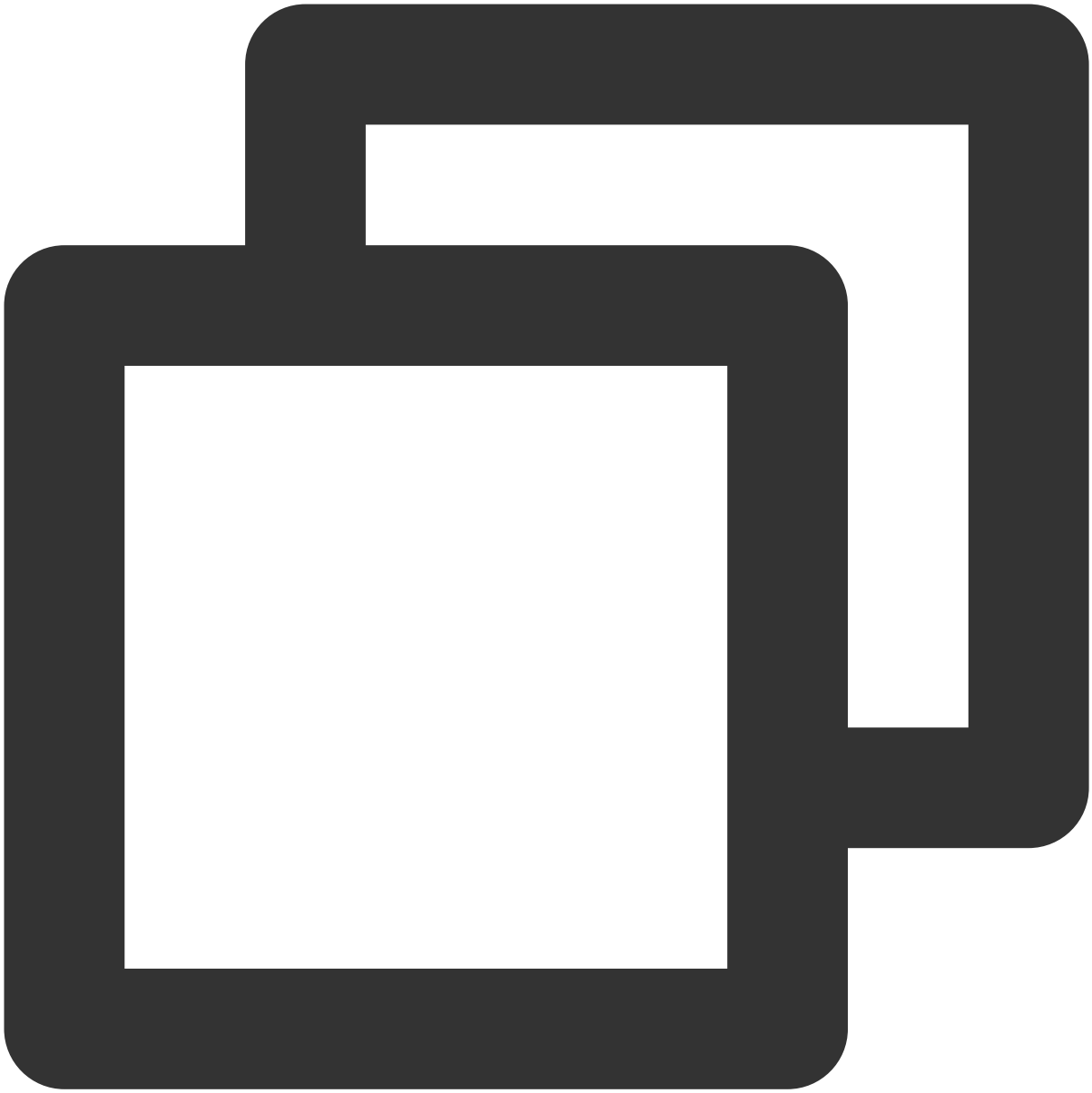
2. Log in and make a call.

2.1 If you are already using a TUIKit component, add the code below to declare `TUICallKit` as a plugin. If you are not already using a TUIKit component, skip this.



```
import { TUICallKit } from './components/TUICallKit/Web';
TUIKit.use(TUICallKit);
```

2.2 Initialize the component before making a call.



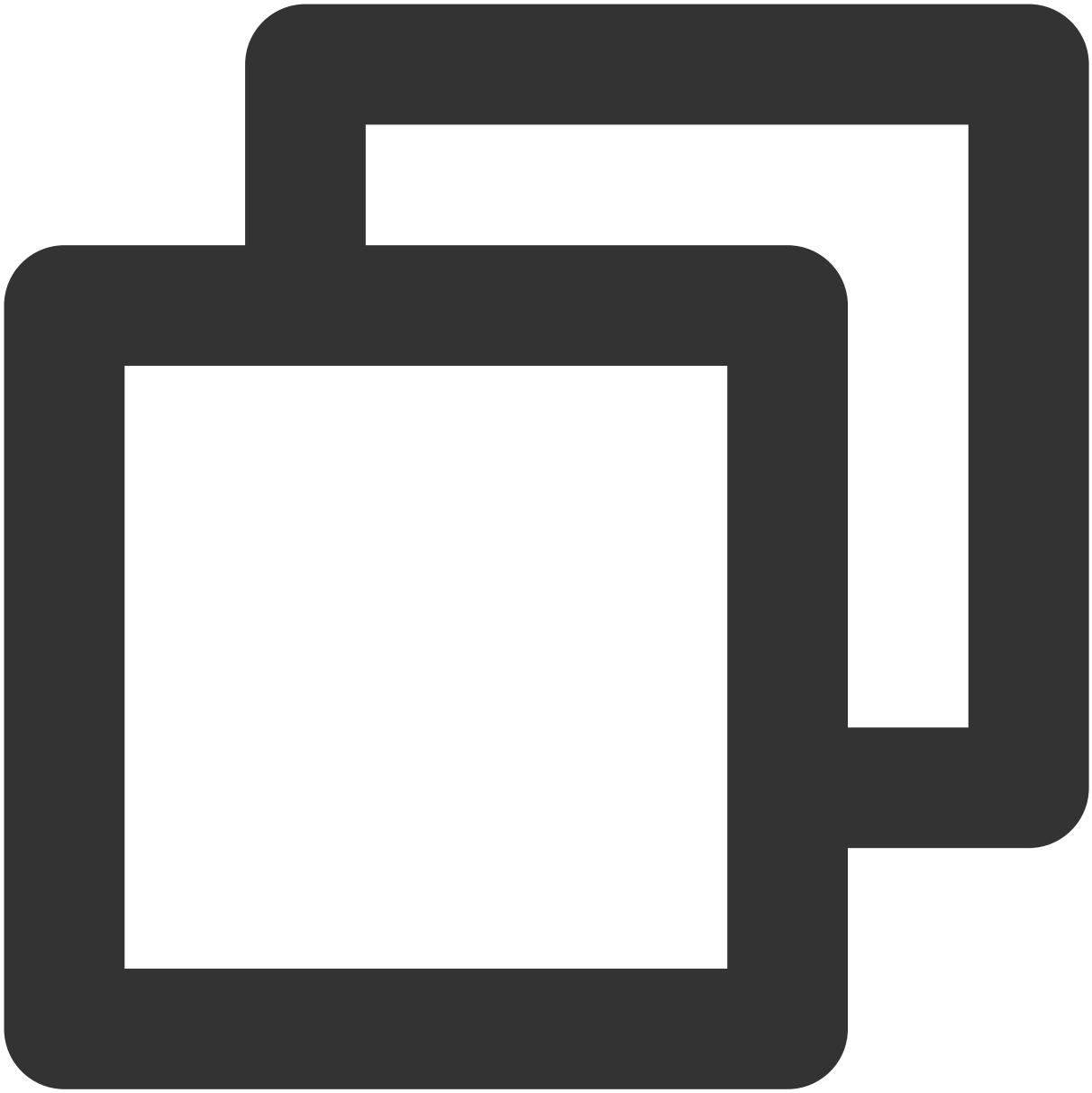
```
import { TUICallKitServer } from './components/TUICallKit/Web';
TUICallKitServer.init({ SDKAppID, userID, userSig });
```

**Note**

If you use Vite to start your project, you also need to perform these steps.

For how to get the `SDKAppID` and `SecretKey`, see step 1.

**As a temporary method**, you can use `genTestUserSig(userID)` in `GenerateTestUserSig.js` to generate `userSig`:



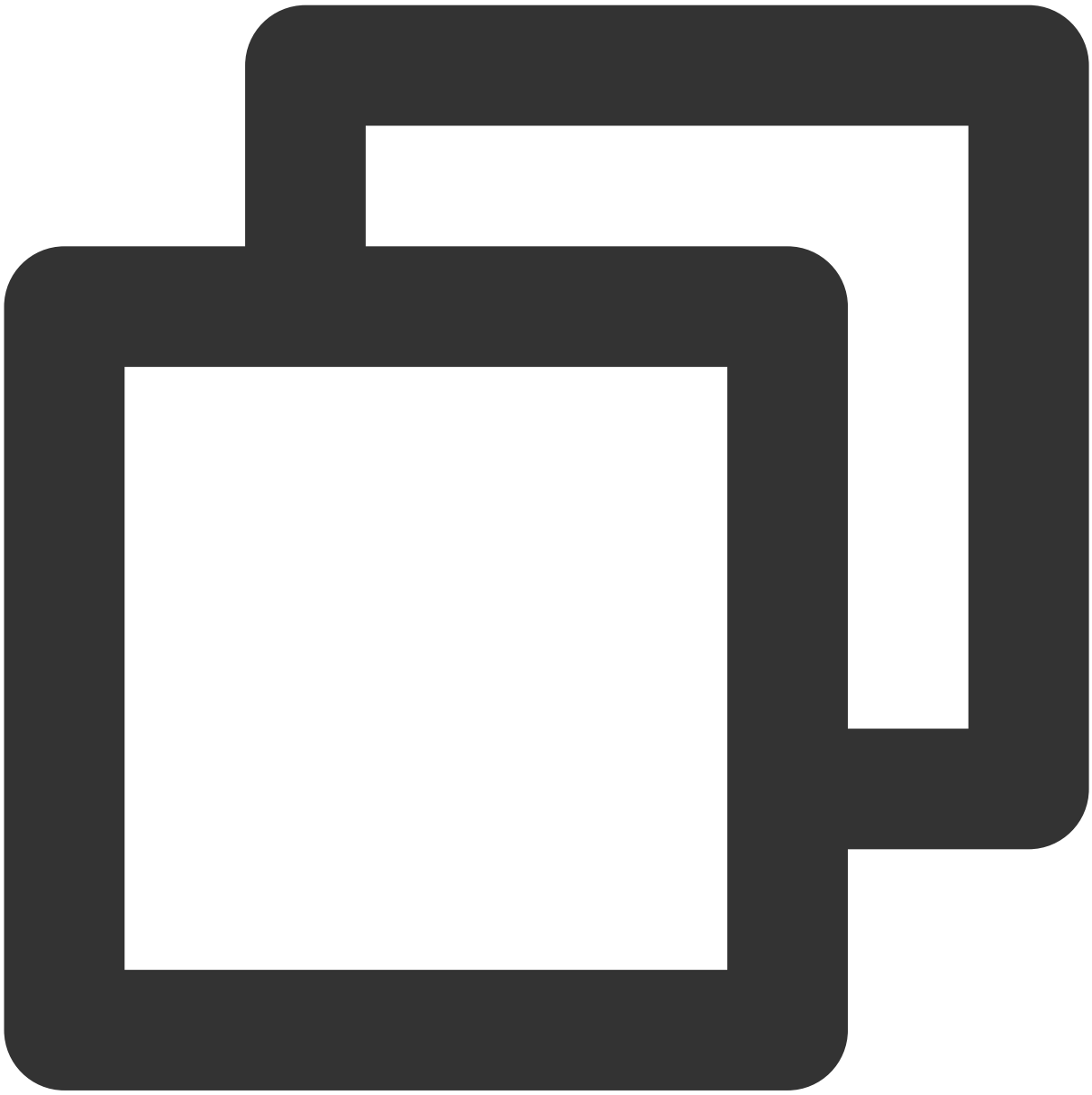
```
import * as GenerateTestUserSig from "./components/TUICallKit/Web/demos/basic/publi
const { userSig } = GenerateTestUserSig.genTestUserSig(userID, SDKAppID, SecretKey)
```

### Caution

In this document, the method to obtain `UserSig` is to configure a `SECRETKEY` in the client code. In this method, the `SECRETKEY` is vulnerable to decompilation and reverse engineering. Once your `SECRETKEY` is disclosed, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for locally running and debugging a demo project**. The correct `UserSig` distribution method is to integrate the calculation code of

`UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to your server for a dynamic `UserSig` . For more information, see Generating UserSig.

2.3 Run the following code where call making needs to be implemented:



```
import { TUICallKitServer } from './components/TUICallKit/Web';
TUICallKitServer.call({ userID: "123", type: 2 }); // One-to-one call
TUICallKitServer.groupCall({ userIDList: ["xxx"], groupID: "xxx", type: 2 }); // Gr
```

You can now make your first call. For information on API parameters, see API Documentation.

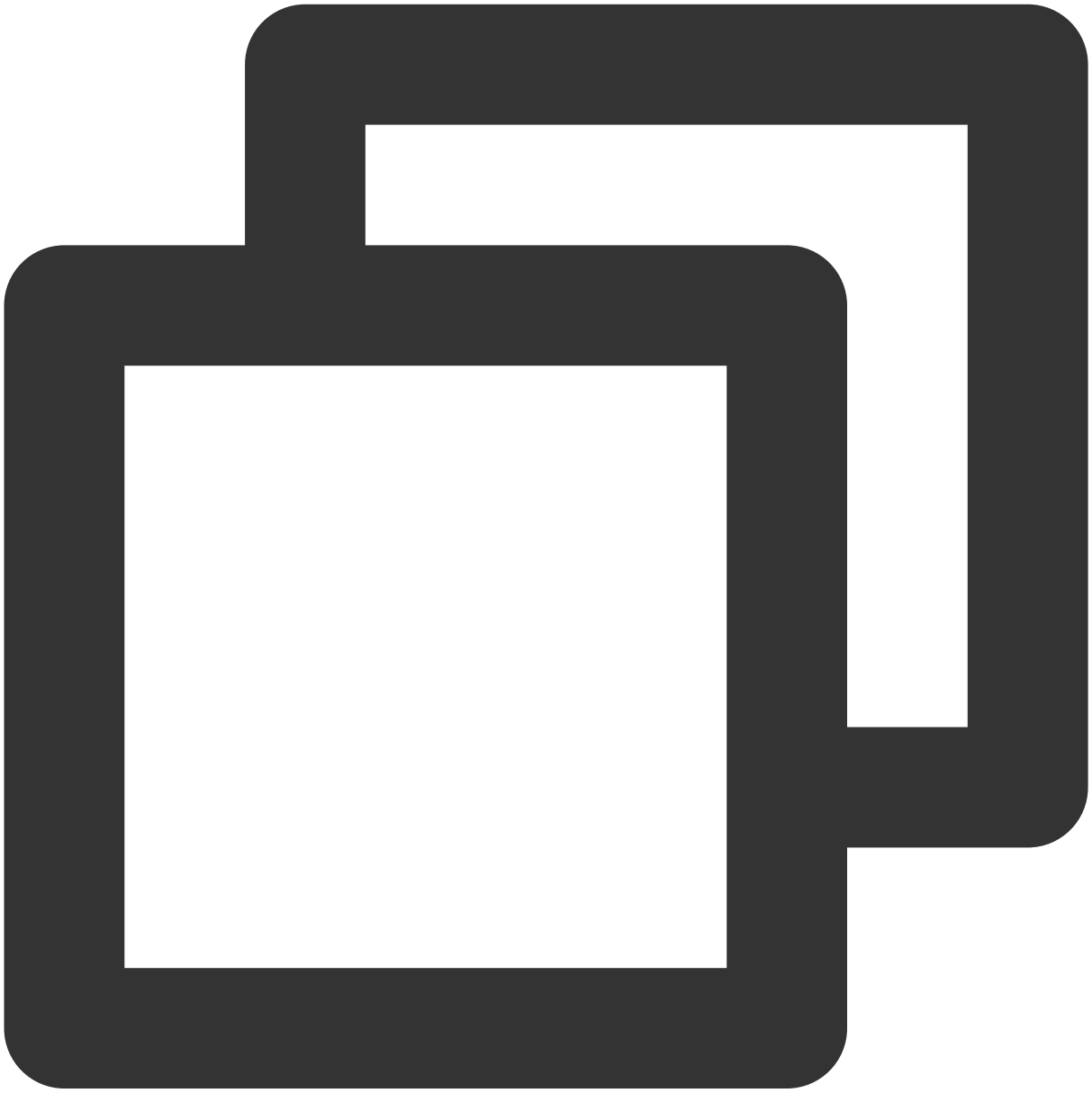
3. Advanced APIs

`TUICallKit` offers the `beforeCalling` and `afterCalling` callbacks to send you notifications about call status.

`beforeCalling` : Returned before a call.

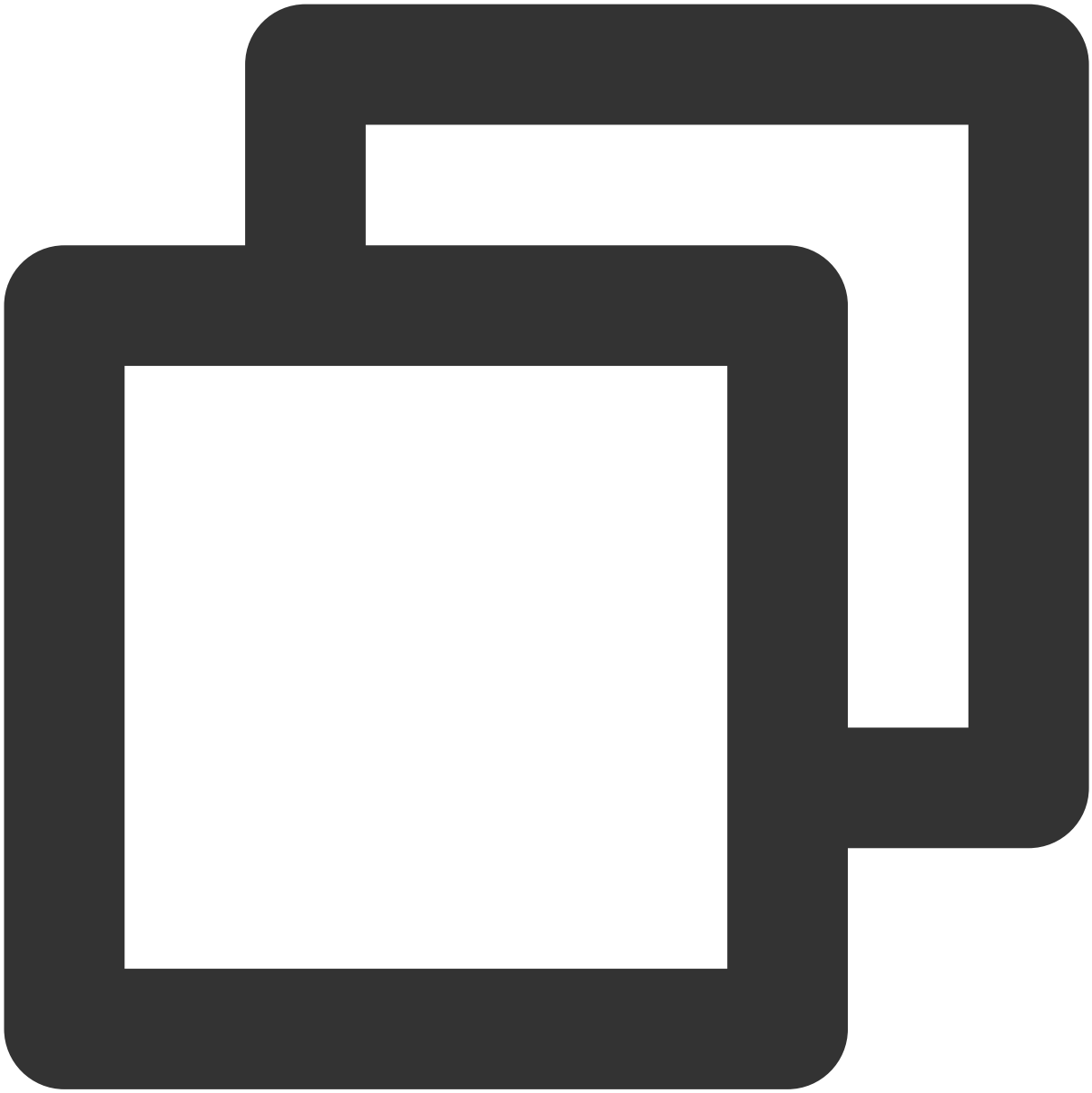
`afterCalling` : Returned after a call.

For example, you can use them to expand and collapse the `<TUICallKit />` component (as shown in the demo).



```
function beforeCalling() {
  console.log("This is executed before a call.");
}
function afterCalling() {
```

```
  console.log("This is executed after a call.");
}
```



```
<TUICallKit :beforeCalling="beforeCalling" :afterCalling="afterCalling"/>
```

## Related documents

TUICallKit

TUICallKit basic demo
UI Customization (TUICallKit)
FAQs (Web)

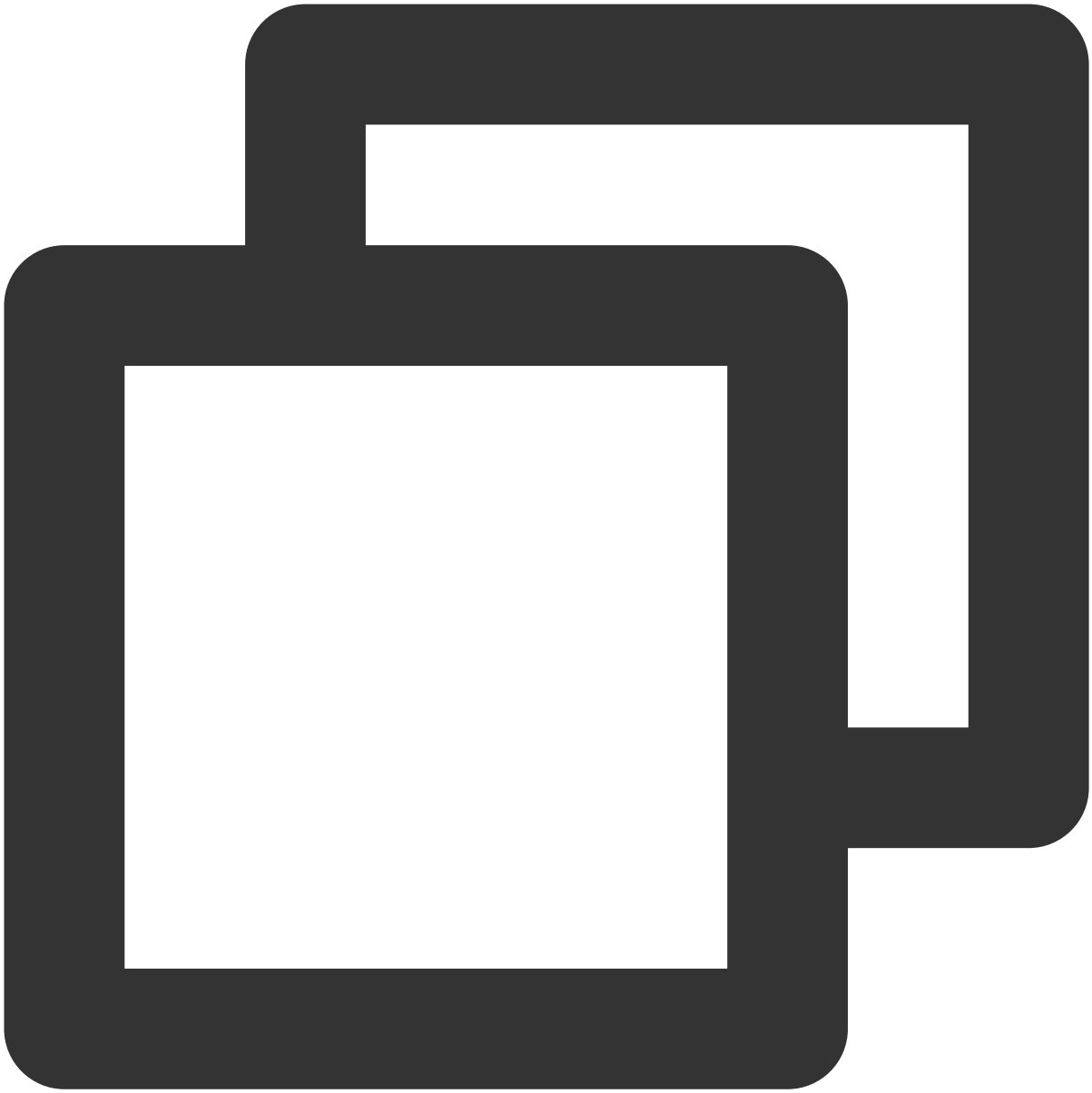
# FAQs

### 1. How do I generate a UserSig?

The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide a project-oriented API. When `UserSig` is needed, your project can send a request to your server for a dynamic `UserSig` . For more information, see Generating UserSig.

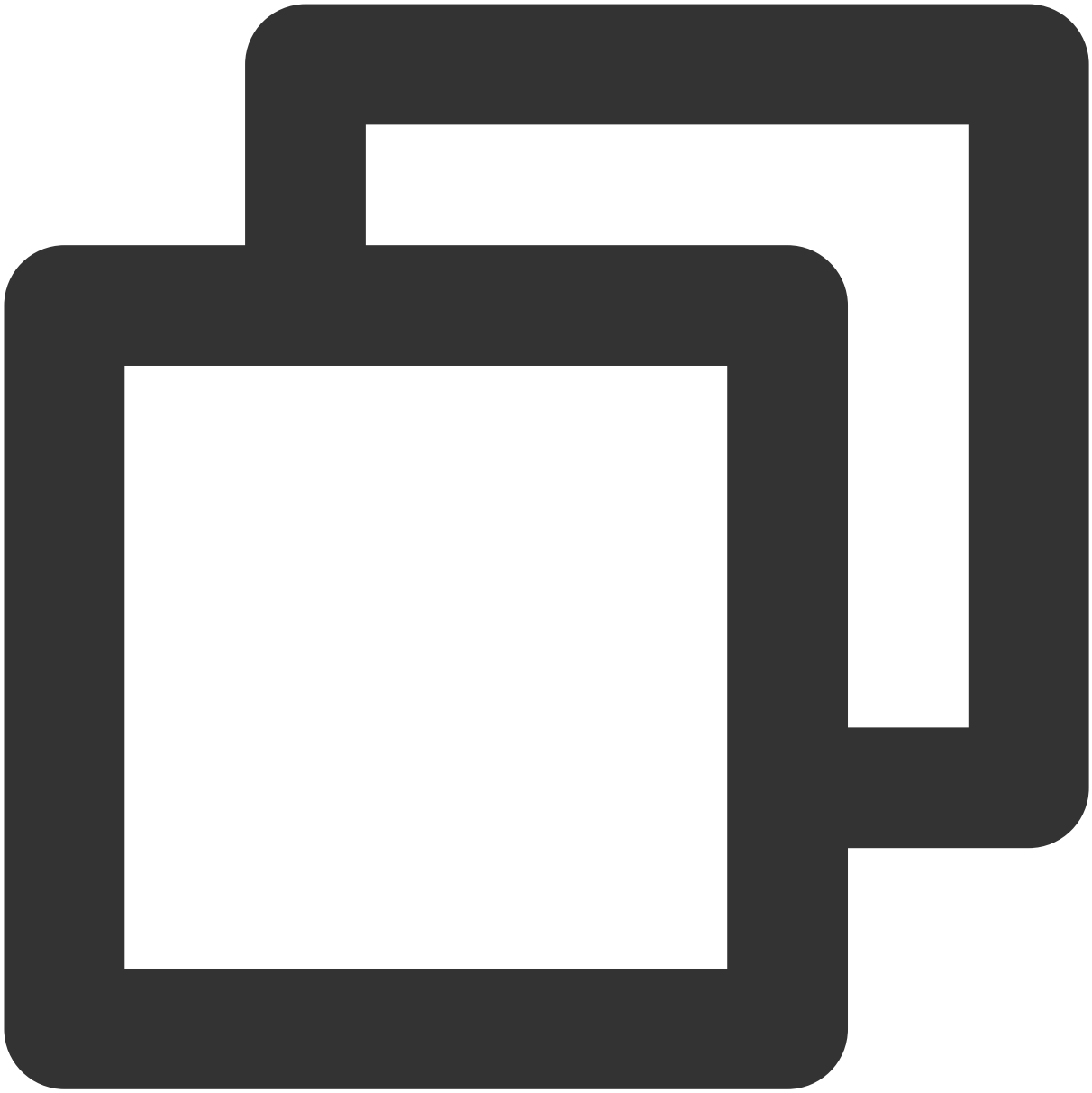
### 2. Vite import

If you created your project using Vite, because Vite uses a different file packaging method, you need to import `lib-generate-test-usersig.min.js` in `index.html` .



```
// index.html
<script src="./src/components/TUICallKit/Web/demos/basic/public/debug/lib-generate-
```

Comment out the import method in `GenerateTestUserSig.js` .



```
// import * as LibGenerateTestUserSig from './lib-generate-test-usersig.min.js'
```

## 3. Environment requirements

### Browser versions

We recommend you use the latest version of Chrome. The table below shows the support for `TUICallKit` by mainstream browsers:

| OS | Browser | Minimum Browser Version Requirement |
| --- | --- | --- |
| | | |

| | | |
|---|---|---|
| macOS | Safari (desktop) | 11+ |
| macOS | Chrome (desktop) | 56+ |
| macOS | Firefox (desktop) | 56+ |
| macOS | Edge (desktop) | 80+ |
| Windows | Chrome (desktop) | 56+ |
| Windows | QQ Browser (desktop, WebKit core) | 10.4+ |
| Windows | Firefox (desktop) | 56+ |
| Windows | Edge (desktop) | 80+ |

**Note**

For more information on browser compatibility, see Browsers Supported. You can also run an online test using the TRTC compatibility check page.

**Network**

Audio/Video calls may fail due to firewall restrictions. To avoid this, add the ports and domain names used by `TUICallKit` to your firewall's allowlist.

**Website domain protocol**

For security and privacy reasons, only HTTPS URLs can access all features of the component. Please use the HTTPS protocol for your website in production environments.

| Scenario | Protocol | Receive (Playback) | Send (Publish) | Share Screen | Remarks |
|---|---|---|---|---|---|
| Production | HTTPS | Supported | Supported | Supported | Recommended |
| Production | HTTP | Supported | Not supported | Not supported | - |
| Local development | http://localhost | Supported | Supported | Supported | Recommended |
| Local development | http://127.0.0.1 | Supported | Supported | Supported | - |
| Local development | http://[local IP] | Supported | Not supported | Not supported | - |

| | | | | | |
|---|---|---|---|---|---|
| Local development | file:/// | Supported | Supported | Supported | - |

# Flutter

Last updated：2024-03-20 16:50:27

This document describes how to quickly integrate the TUICallKit component. Performing the following key steps generally takes about 10 minutes, after which you can enjoy the audio and video call feature with a complete UI.

## Environment Preparations

Flutter 3.0 or higher version.

## Step 1. Activate the service

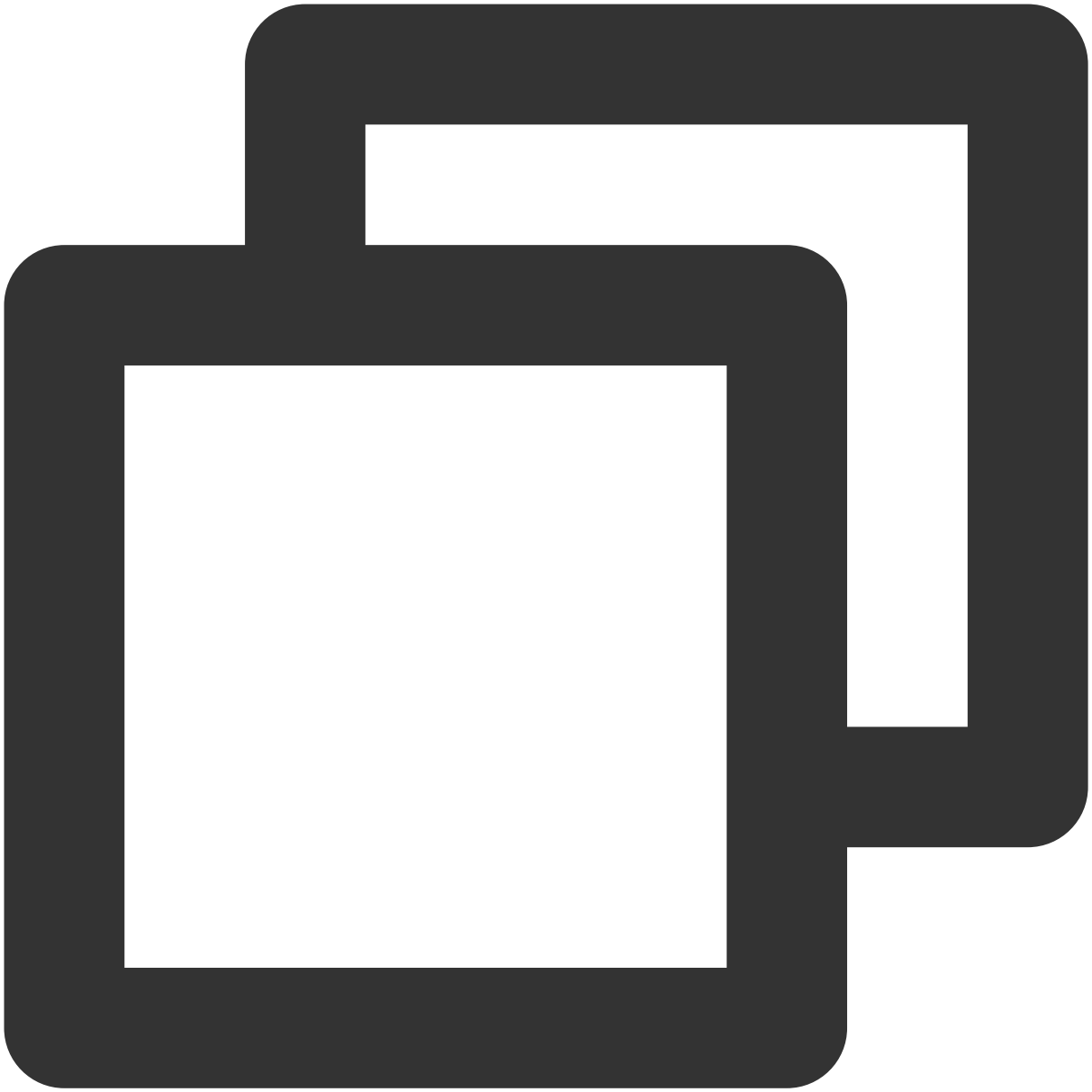
**TUICallKit** is an audio & video communication component built upon Tencent Cloud's paid PaaS services, **Chat** and **Real-time Communication (TRTC)** . In order for you to better experience the Call feature, we offer a 7-day free trial version for each SDKAppID (Note: no additional call duration is provided with the free trial version). Each SDKAppID can apply the free trial version twice, with each trial lasting for 7 days; at the same time, the total number of trial opportunities for all SDKAppID under one account (UIN) is 10.

You can activate the Call free trial version in the Chat console, with the specific operation steps as follows:

1. Log into the Chat console, select the data center, and create a new application. Skip this step if you already have an application.
2. Click on the target application card to enter the application's basic configuration page.
3. Locate the Call card, and click on "Free Trial".
4. After confirming the content in the popup window, click on "**Activate now**". Once activated, you can proceed with integration according to this document.
5. If you need to purchase the official version for your business to go live, you can proceed to the console to make the purchase. Please refer to Purchase Official Version.

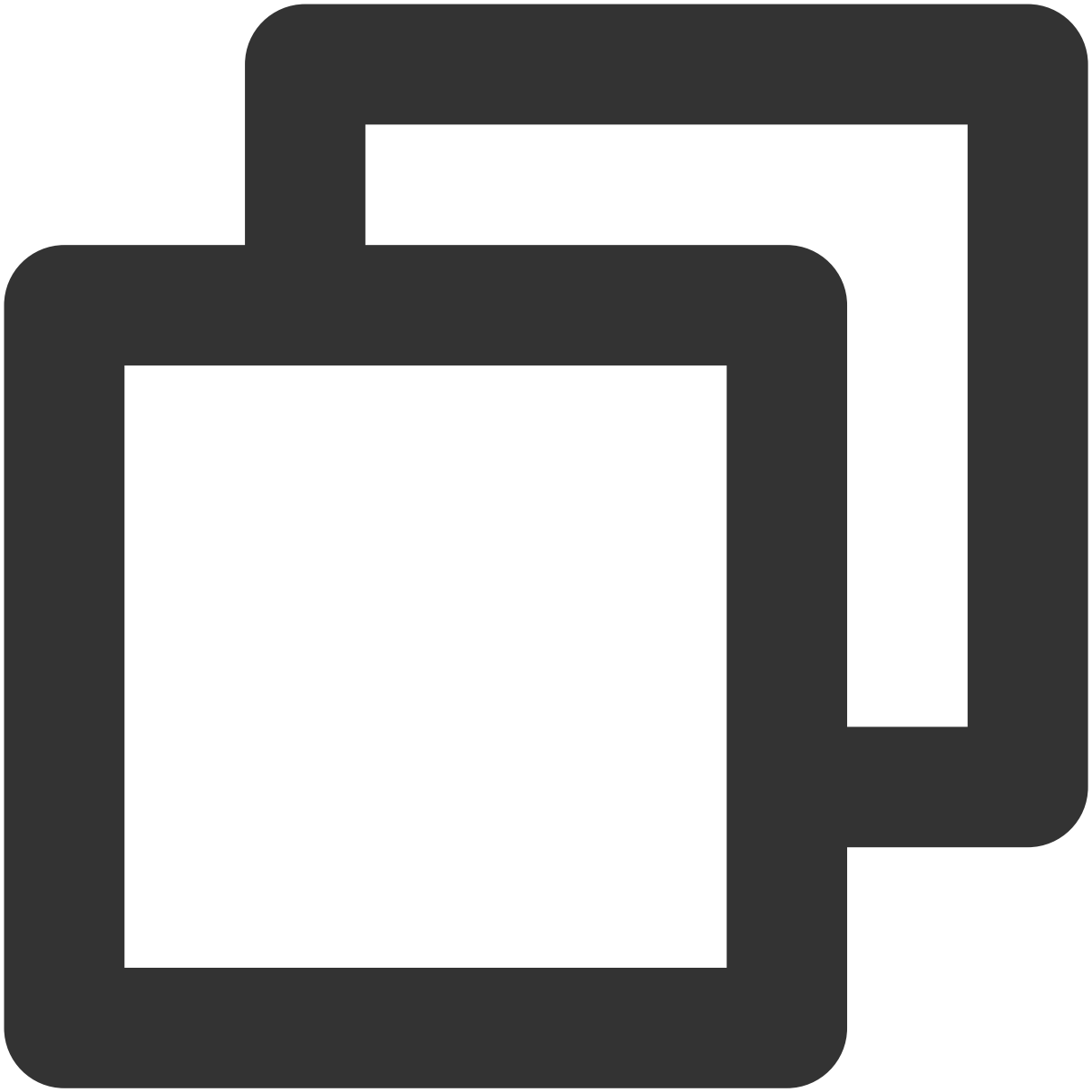
## Step 2. Import the component

Import Flutter TUICallKit, the steps are as follows:
1. Add the tencent_calls_uikit plugin dependency in your pubspec.yaml file, and click the plugin hyperlink to switch to the Versions directory to check for the latest version.



```
dependencies:
  tencent_calls_uikit: The latest version
```

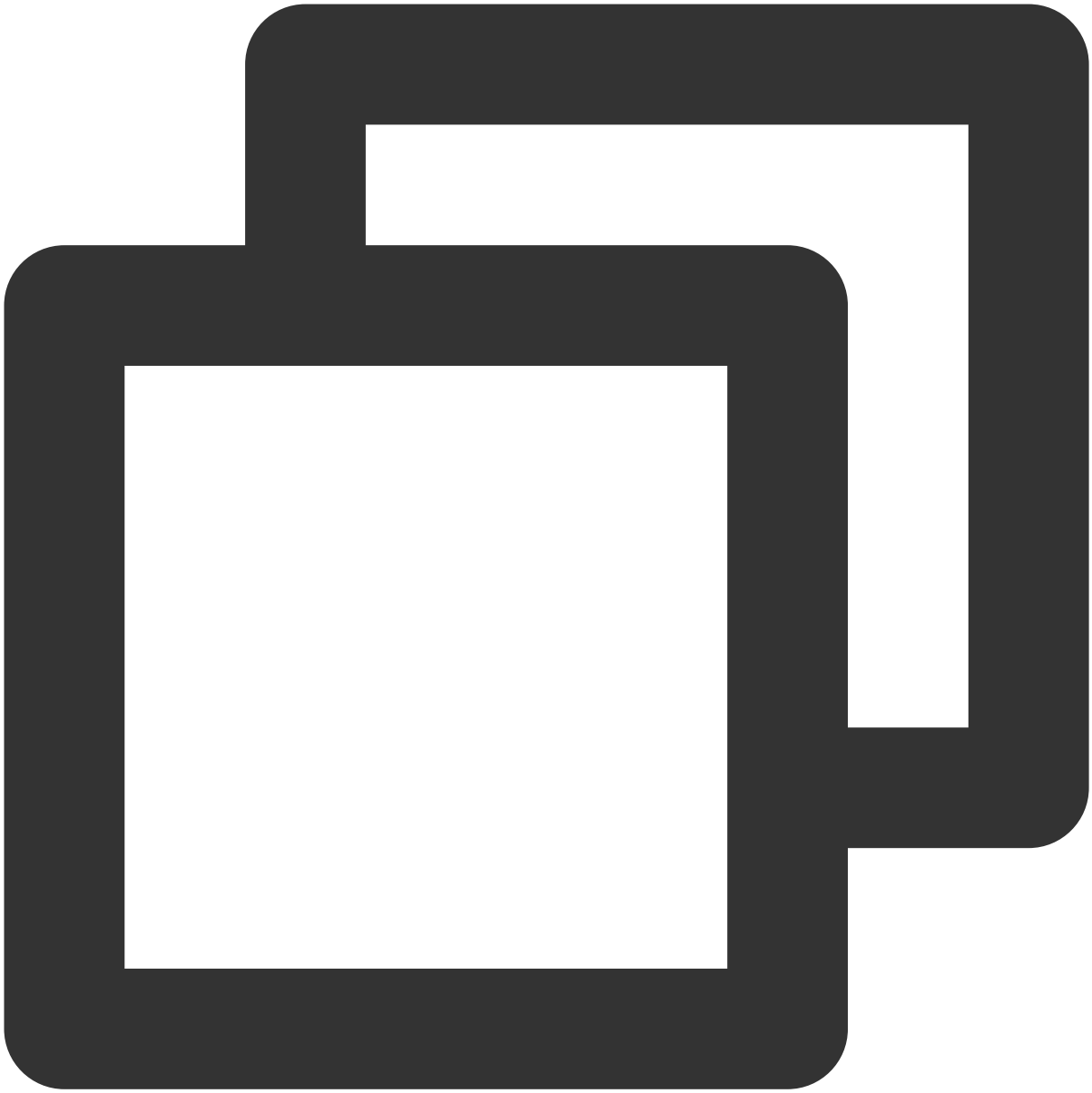
2. Execute the following command to install the components.



```
flutter pub get
```

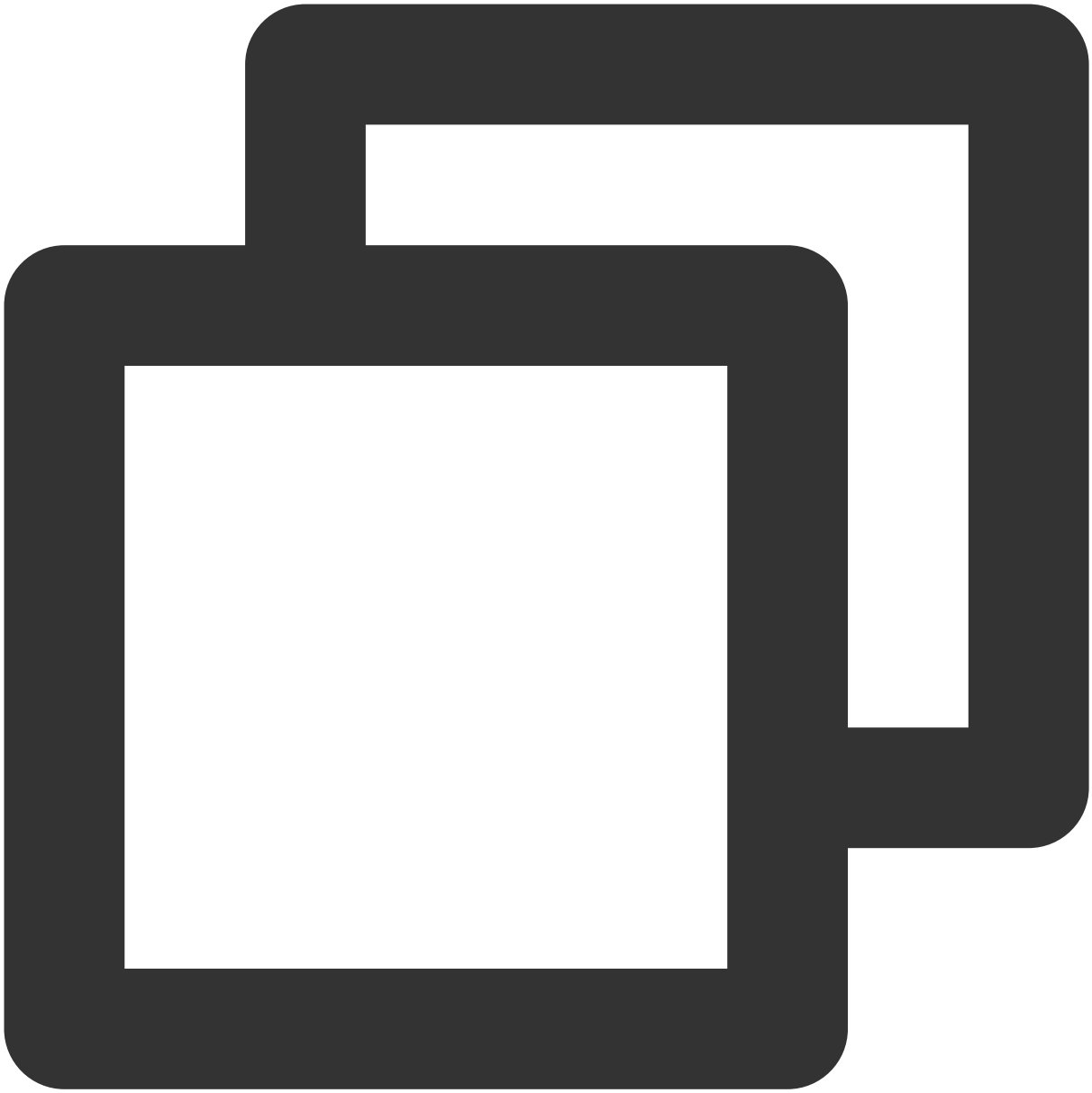
## Step 3. Configure the project

1. Add the navigatorObserver of TUICallKit to the App component, taking MateriaApp as an example, the code is as follows:



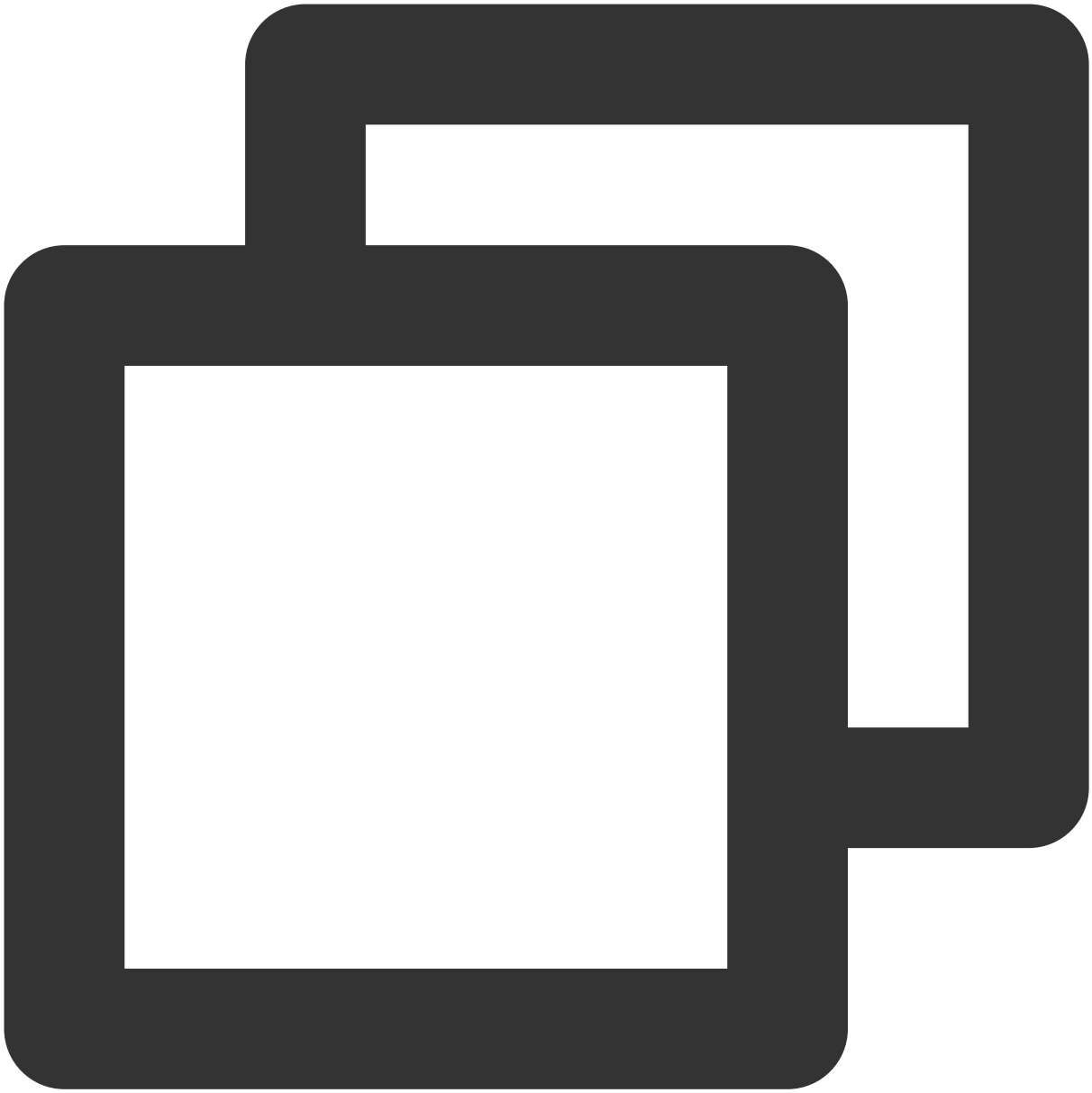
```
import 'package:tencent_calls_uikit/tuicall_kit.dart';

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

2. If you need to compile and run on the Android platform, since we use the reflection feature of Java inside the SDK, some classes in the SDK need to be added to the non-confusing list, so you need to add the following code to the `proguard-rules.pro` file:



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

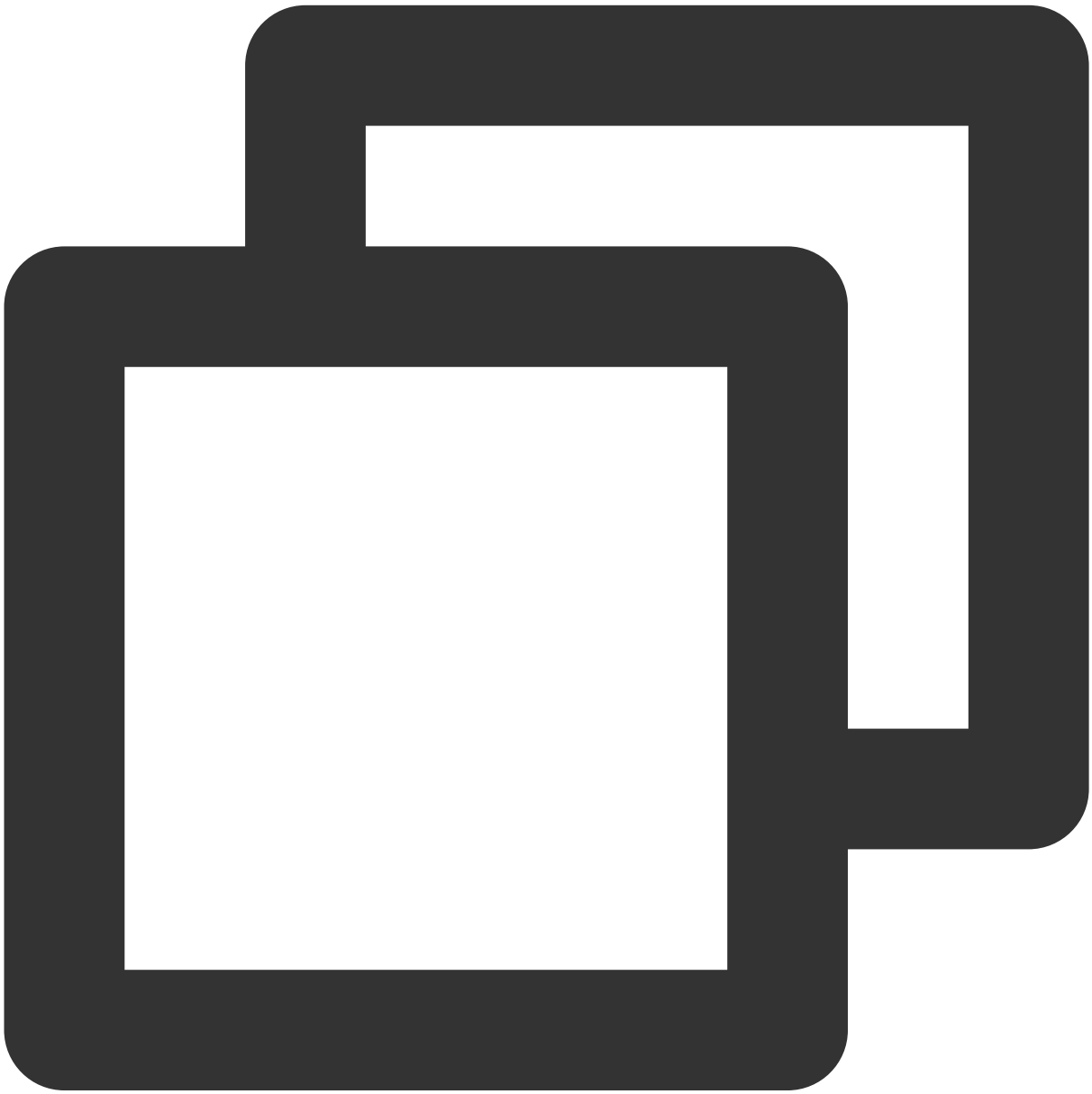
3. If your project needs to be debugged on the iOS simulator, you need to add the following code to the project's `/ios/Podfile` :



```
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
```

## Step 4. Log in to the `TUICallKit` component

Add the following code to your project to call the relevant APIs in `TUICore` to log in to the `TUICallKit` component. This step is very important, as the user can use the component features properly only after a successful login. Carefully check whether the relevant parameters are correctly configured:



```
TUIResult result = TUICallKit.instance.login(SDKAppID,     // Please replace it wit
                                             'userId',       // Please replace with
                                             'userSig');     // You can get a UserS
```

**Parameter description:** The key parameters used by the `login` function are as detailed below:

**SDKAppID**: Obtained in the last step in step 1 and not detailed here.

**UserID**: The ID of the current user, which is a string that can contain only letters (a–z and A–Z), digits (0–9), hyphens (-), or underscores (_).

**UserSig**: The authentication credential used by Tencent Cloud to verify whether the current user is allowed to use the TRTC service. You can get it by using the `SDKSecretKey` to encrypt the information such as `SDKAppID` and `UserID`. You can generate a temporary `UserSig` by clicking the UserSig Generate button in the console. For more information, see UserSig.

**Note**

**Many developers have contacted us with many questions regarding this step. Below are some of the frequently encountered problems:**

SDKAppID is invalid.

userSig is set to the value of Secretkey mistakenly. The userSig is generated by using the SecretKey for the purpose of encrypting information such as sdkAppId, userId, and the expiration time. But the value of the userSig that is required cannot be directly substituted with the value of the SecretKey.

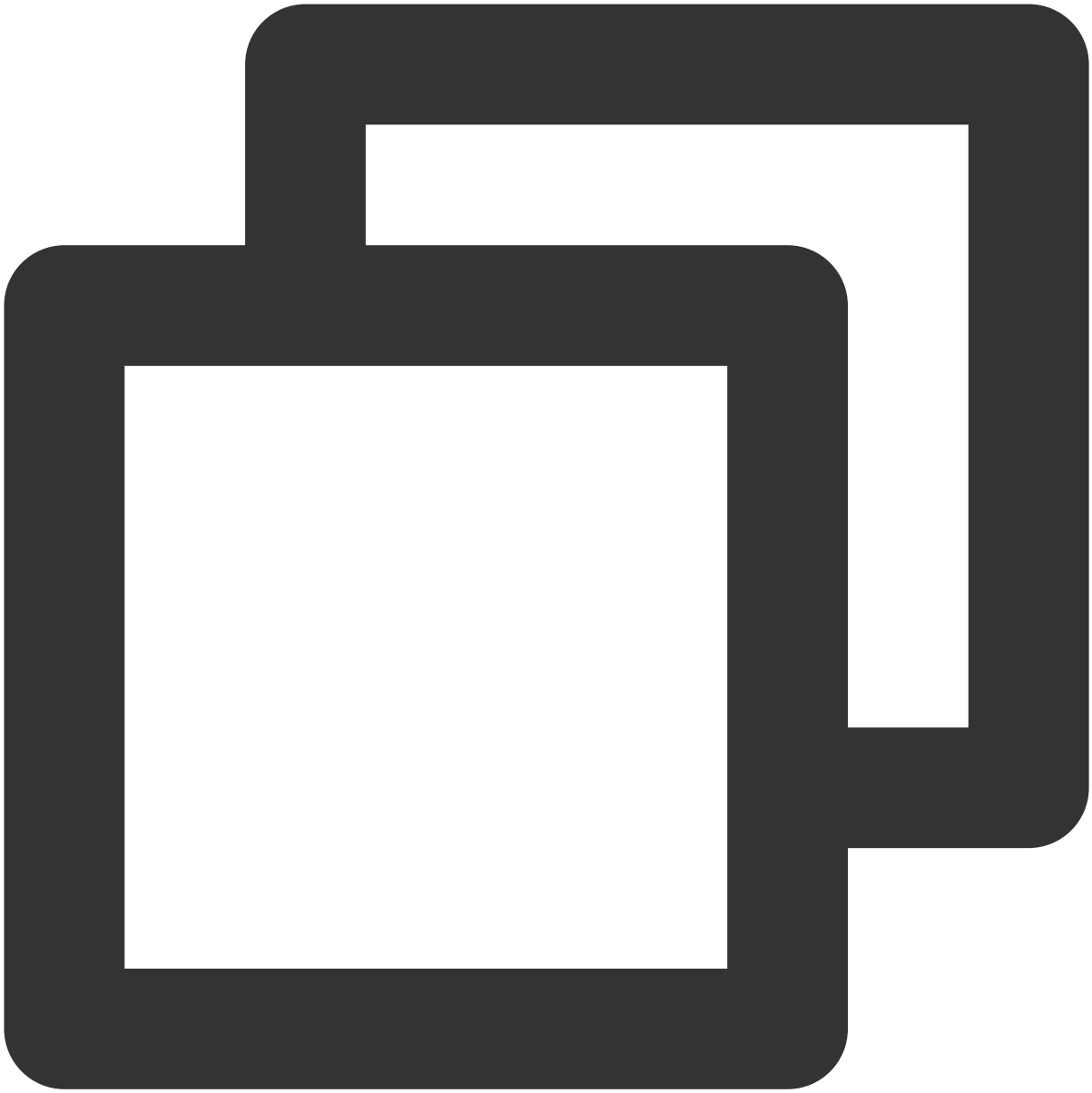
userId is set to a simple string such as 1, 123, or 111, and your colleague may be using the same userId while working on a project simultaneously. In this case, login will fail as TRTC doesn't support login on multiple terminals with the same UserID. Therefore, we recommend you use some distinguishable userId values during debugging.

The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current integration process more quickly. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` subsequently. Therefore, we strongly recommend you run the `UserSig` calculation logic on the server and make the application request the `UserSig` calculated in real time every time the application uses the `TUICallKit` component from the server.

# Step 5. Make a call

### One-to-one video call

You can call the `call` function of `TUICallKit` and specify the call type and the callee's `userId` to make an audio/video call.

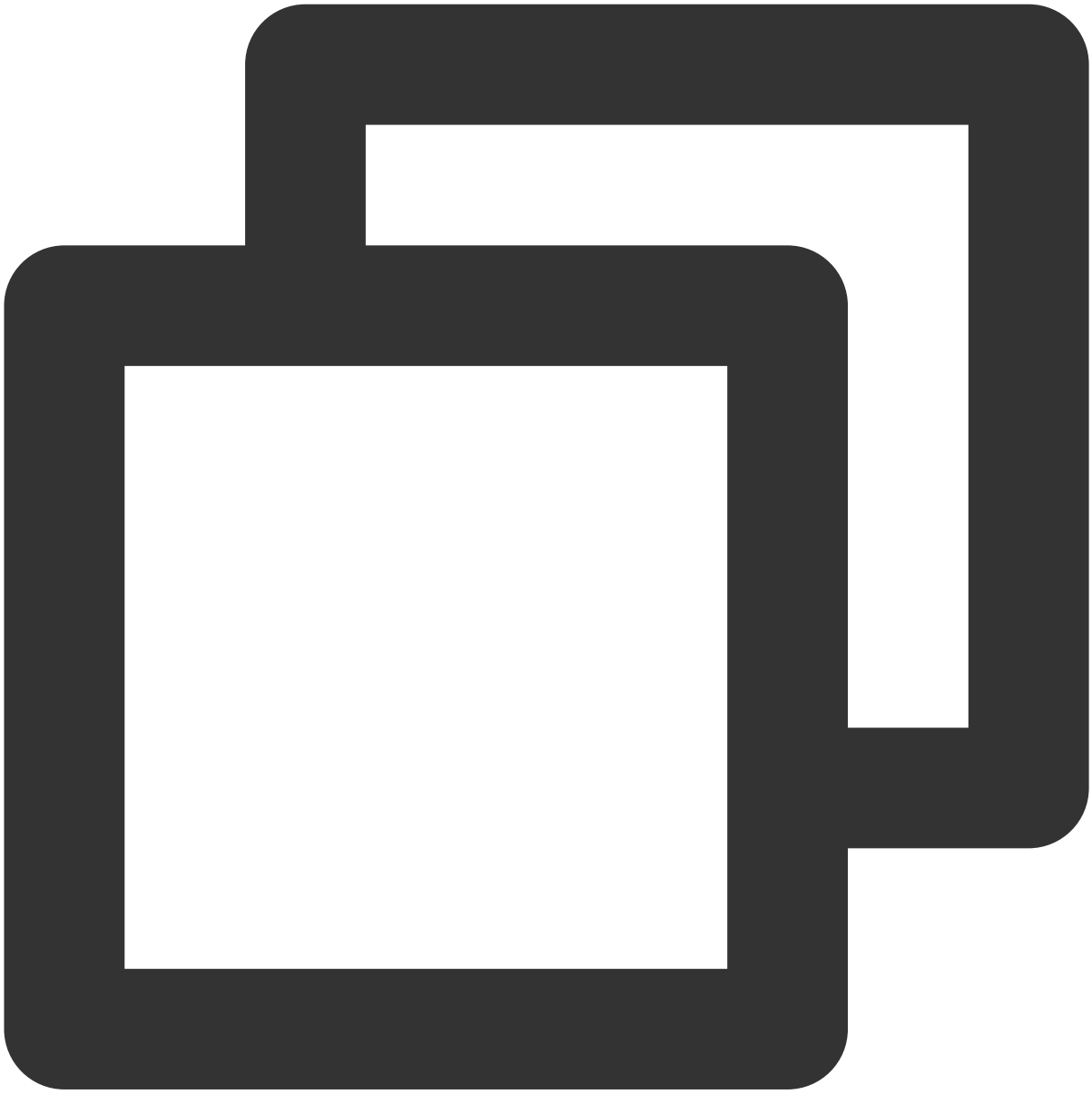


```
// make a video call to Mike
TUICallKit.instance.call('mike', TUICallMediaType.video);
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the target user, such as `"mike"`. |
| callMediaType | TUICallDefine.MediaType | The call media type, such as `TUICallDefine.MediaType.Video`. |

### Group video call

You can call the `groupCall` function of `TUICallKit` and specify the call type and the list of callees' `UserID` values to make a group audio/video call.



```
// in the group(groupId:0001), make a video call to denny, mike and tommy
TUICallKit.instance.groupCall('0001', ['denny', 'mike', 'tommy'], TUICallMediaType.
```

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID, such as `"12345678"`. |

| | | |
|---|---|---|
| userIdList | List | The list of `UserID` values of the target users, such as `{"jane", "mike", "tommy"}`. |
| callMediaType | TUICallDefine.MediaType | The call media type, such as `TUICallDefine.MediaType.Video`. |

**Note**

You can create a group as instructed in Android, iOS, and macOS. You can also use Chat TUIKit to integrate chat and call scenarios at one stop.

`TUICallKit` currently doesn't support making a multi-person video call among users not in a group. If you have such a need, contact colleenyu@tencent.com.

# Step 6. Answer a call

After receiving an incoming call, the `TUICallKit` component will automatically wake up the call answering UI. However, the wake effect varies by Android system permissions as follows:

If your application is in the foreground, it will pop up the call UI and play back the incoming call ringtone automatically when receiving an incoming call.

If your application is in the background and is granted the `Display over other apps` or `Display pop-up windows while running in the background` permission, it will still pop up the call UI and play back the incoming call ringtone automatically when receiving an incoming call.

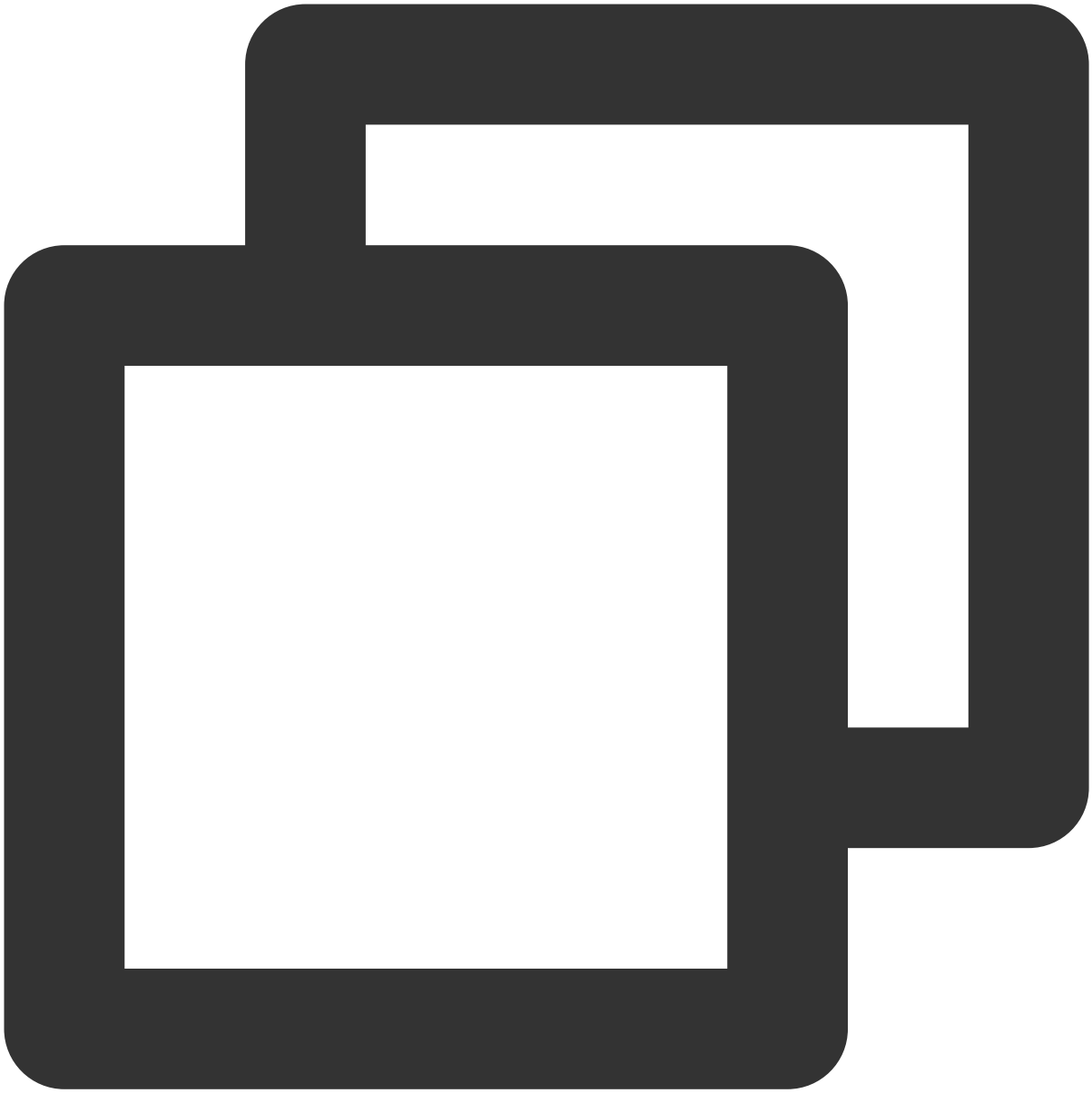
If your application is in the background but isn't granted with the `Display over other apps` or `Display pop-up windows while running in the background` permission, `TUICallKit` will play back the incoming call ringtone to prompt the user to answer or decline the call.

If the application process has been terminated, you can use the offline push feature as described in Offline Call Push (Android) to prompt the user to answer or decline the call through the status bar notification.

# Step 7. Implement more features

### 1. Nickname and profile photo settings

To customize the nickname or profile photo, use the following API for update:



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

**Note**

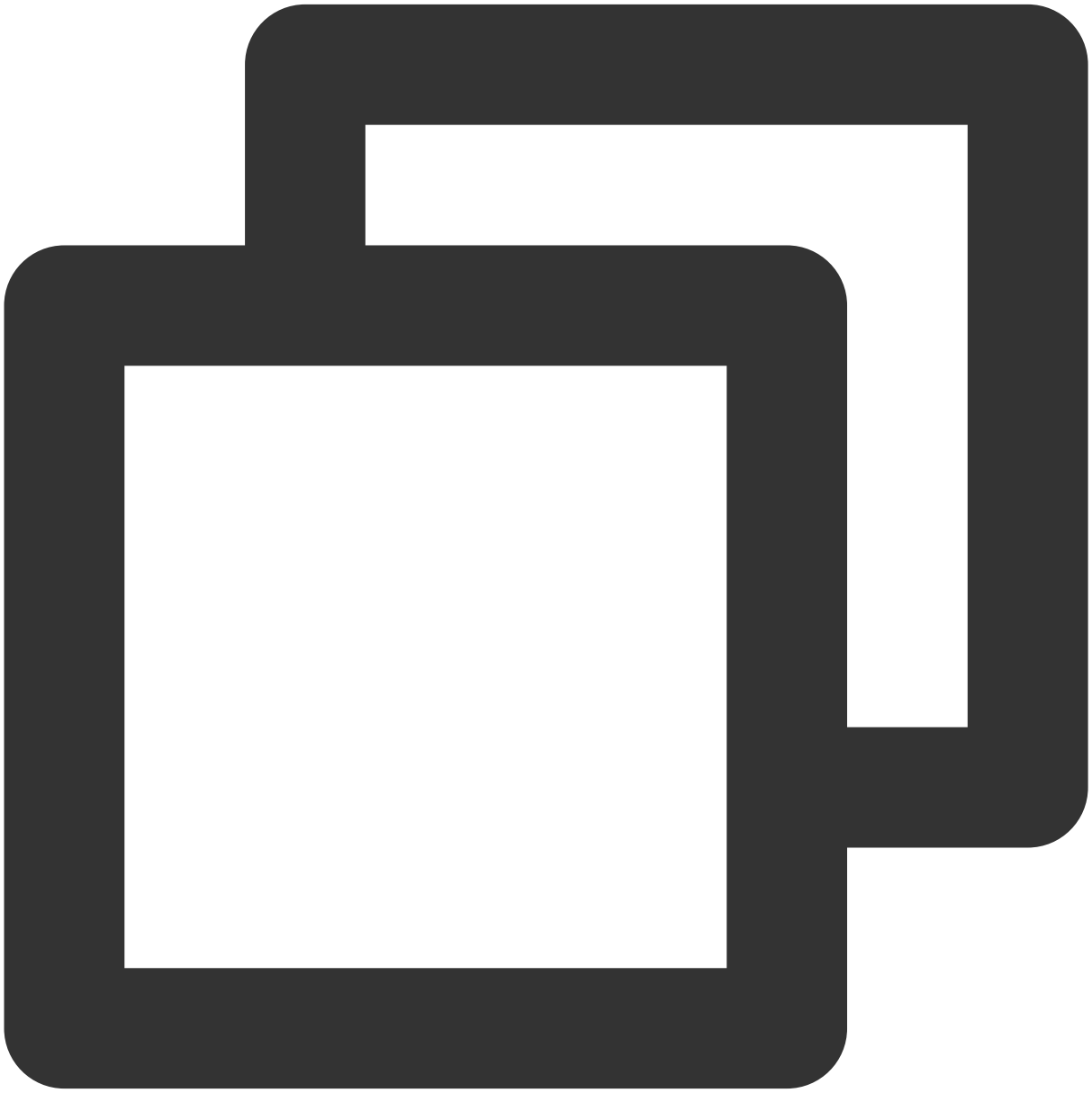
The update of the callee's nickname and profile photo may be delayed during a call between non-friend users due to the user privacy settings. After a call is made successfully, the information will also be updated properly in subsequent calls.

### 2. Offline call push

After completing the above steps, you can dial and connect audio and video calls. However, if your business scenario needs to receive audio and video call requests normally after the application process is killed or the application retreats to the background, you need to Added offline wakeup function, see offline wakeup (Flutter) for details.

### 3. Floating window

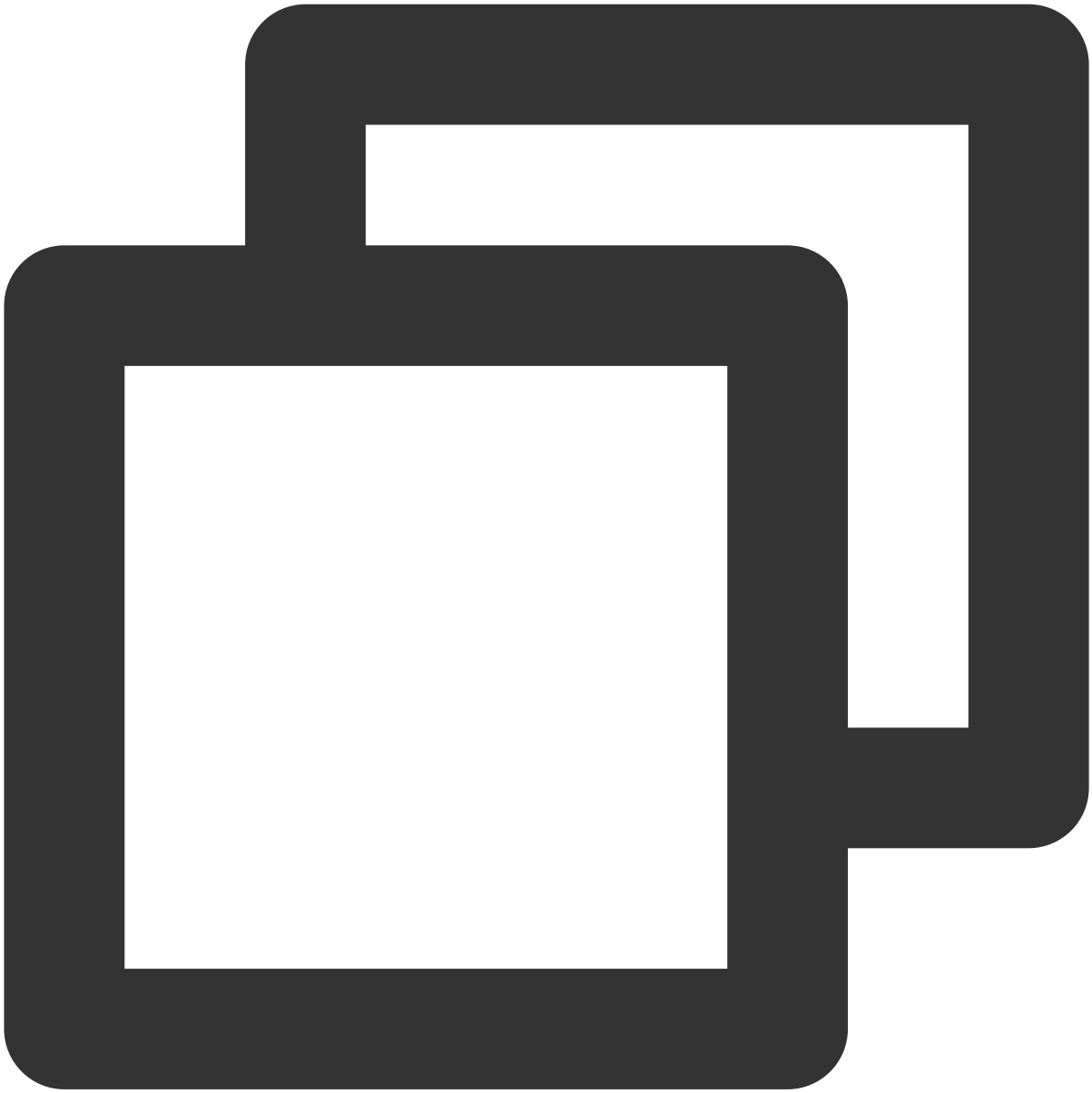
To implement the floating window feature in your application, call the following API when initializing the `TUICallKit` component:



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

### 4. Call status listening

To **listen on the call status** (for example, the start or end of a call or the call quality during a call), listen on the following events:

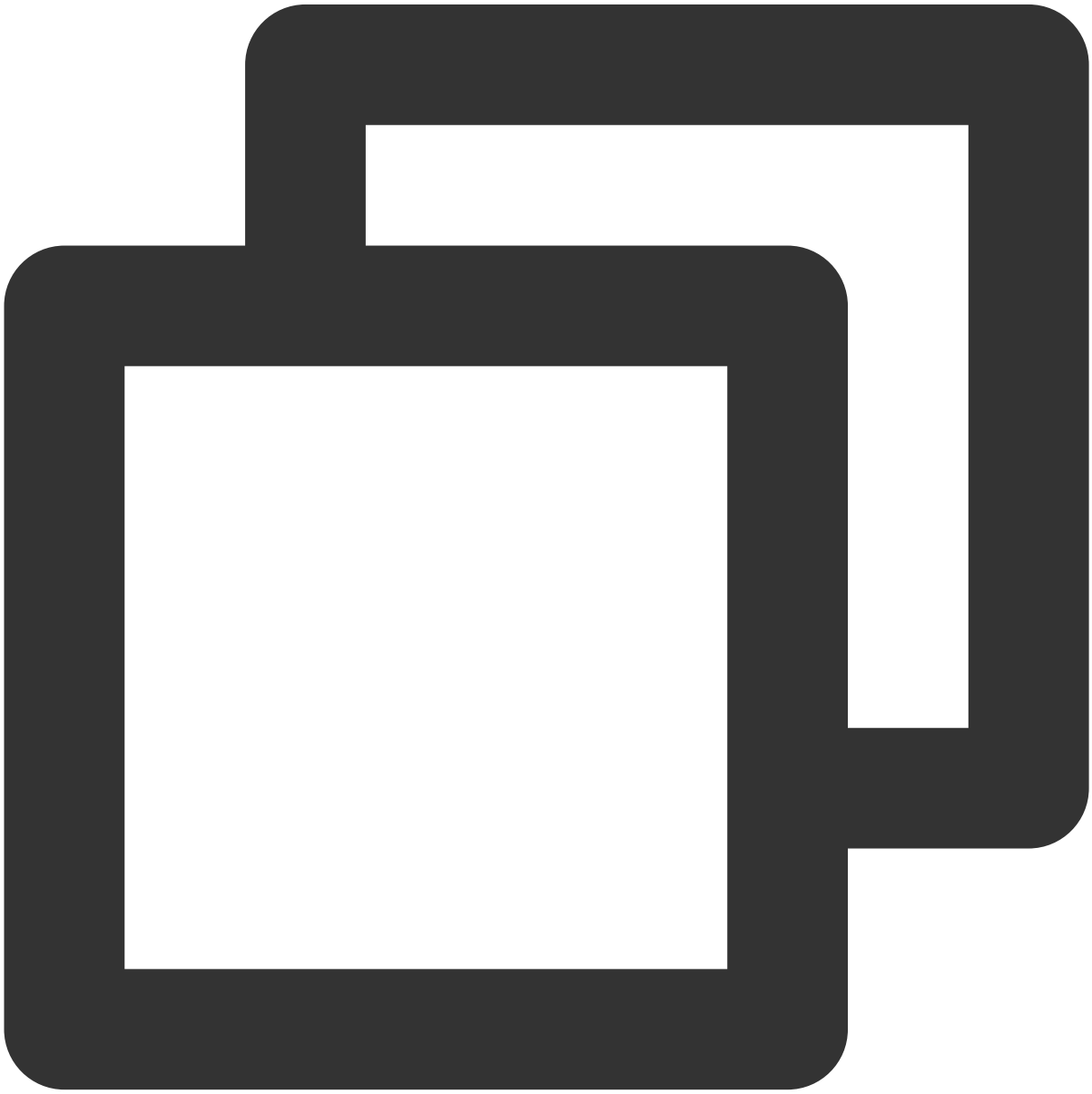


```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {

    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole

    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
```

```
    },, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityLis

    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,

    }
));
```

### 5. Custom ringtone

You can use the following API to customize the ringtone:



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

# FAQs

### 1. What should I do if I receive the error message "The package you purchased does not support this ability"?

The error message indicates that your application's audio/video call capability package has expired or is not activated. You can claim or activate the audio/video call capability as instructed in step 1 to continue using `TUICallKit`.

# Suggestions and Feedback

If you have any suggestions or feedback, please contact colleenyu@tencent.com.

# API Documentation(TUICallKit)
# Android
# API Overview

Last updated：2023-07-13 16:40:03

## TUICallKit (UI Included)

TUICallKit is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat.

| API | Description |
| --- | --- |
| createInstance | Creates a TUICallKit instance (singleton mode). |
| setSelfInfo | Sets the user nickname and profile photo. |
| call | Makes a one-to-one call. |
| call | Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc |
| groupCall | Makes a group call. |
| groupCall | Makes a group call, Support for custom room ID, call timeout, offline push content, etc |
| joinInGroupCall | Joins a group call. |
| setCallingBell | Sets the ringtone. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |

## TUICallEngine (No UI)

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

| API | Description |
| --- | --- |

| | |
|---|---|
| createInstance | Creates a `TUICallEngine` instance (singleton). |
| destroyInstance | Terminates a `TUICallEngine` instance (singleton). |
| init | Authenticates the basic audio/video call capabilities. |
| addObserver | Registers an event listener. |
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Answers a call. |
| reject | Declines a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Switches the call media type, such as from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |
| openCamera | Turns the camera on. |
| closeCamera | Turns the camera off. |
| switchCamera | Switches the camera. |
| openMicrophone | Enables the mic. |
| closeMicrophone | Disables the mic. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver/speaker). |
| setSelfInfo | Sets the user nickname and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for `TUICallEngine` (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |

| | |
|---|---|
| setVideoEncoderParams | Set the encoding parameters of video encoder. |
| getTRTCCloudInstance | Advanced features. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

# TUICallObserver

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

| API | Description |
|---|---|
| onError | An error occurred during the call. |
| onCallReceived | A call was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user has a video stream. |
| onUserAudioAvailable | Whether a user has an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | The current user was kicked offline. |
| onUserSigExpired | The user sig is expired. |

# Definitions of Key Types

| API | Description |
| --- | --- |
| TUICallDefine.MediaType | The call type. Enumeration: Video call and audio call. |
| TUICallDefine.Role | The call role. Enumeration: Caller and callee. |
| TUICallDefine.Status | The call status. Enumeration: Idle, waiting, and answering. |
| TUICommonDefine.RoomId | The room ID, which can be a number or string. |
| TUICommonDefine.Camera | The camera type. Enumeration: Front camera and rear camera. |
| TUICommonDefine.AudioPlaybackDevice | The audio playback device type. Enumeration: Speaker and receiver. |
| TUICommonDefine.NetworkQualityInfo | The current network quality. |

# TUICallKit

Last updated：2023-06-28 16:11:11

## TUICallKit APIs

`TUICallKit` is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat. For directions on integration, see Integrating TUICallKit.

## API Overview

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallKit` instance (singleton mode). |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| call | Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc |
| groupCall | Makes a group call. |
| groupCall | Makes a group call, Support for custom room ID, call timeout, offline push content, etc |
| joinInGroupCall | Joins a group call. |
| setCallingBell | Sets the ringtone. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |

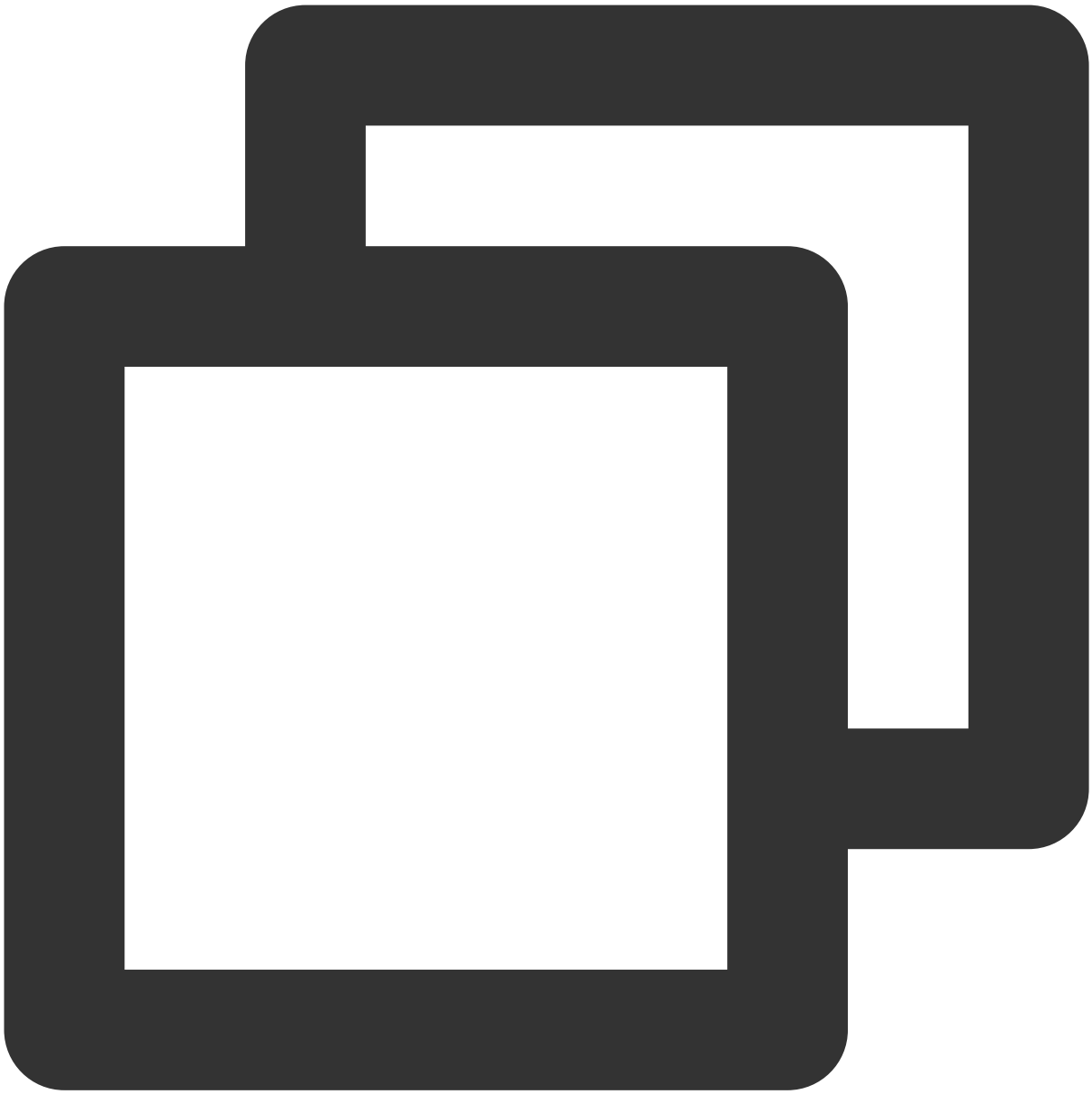
## Details

### createInstance

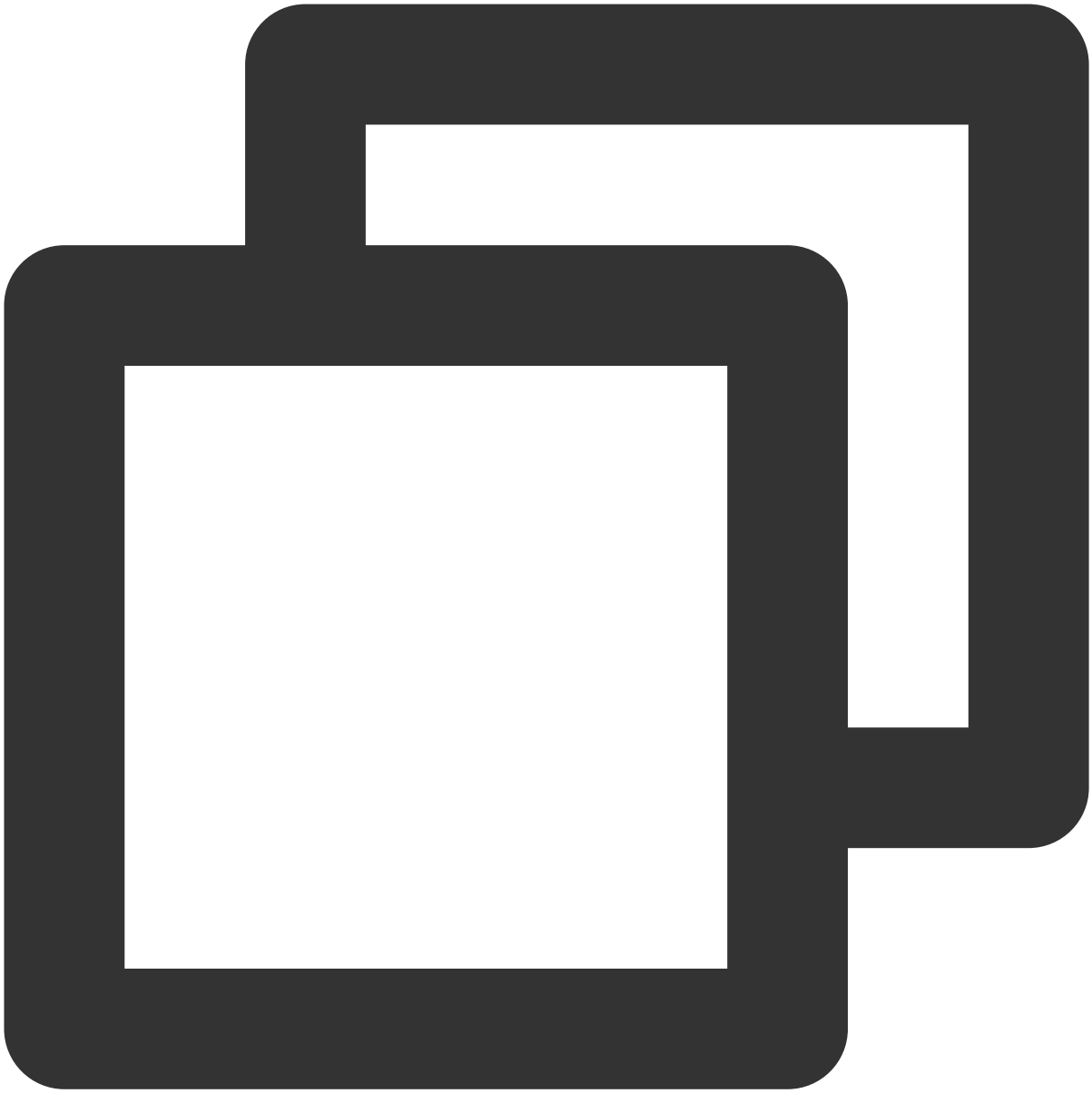
This API is used to create a `TUICallKit` singleton.

Kotlin

Java



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



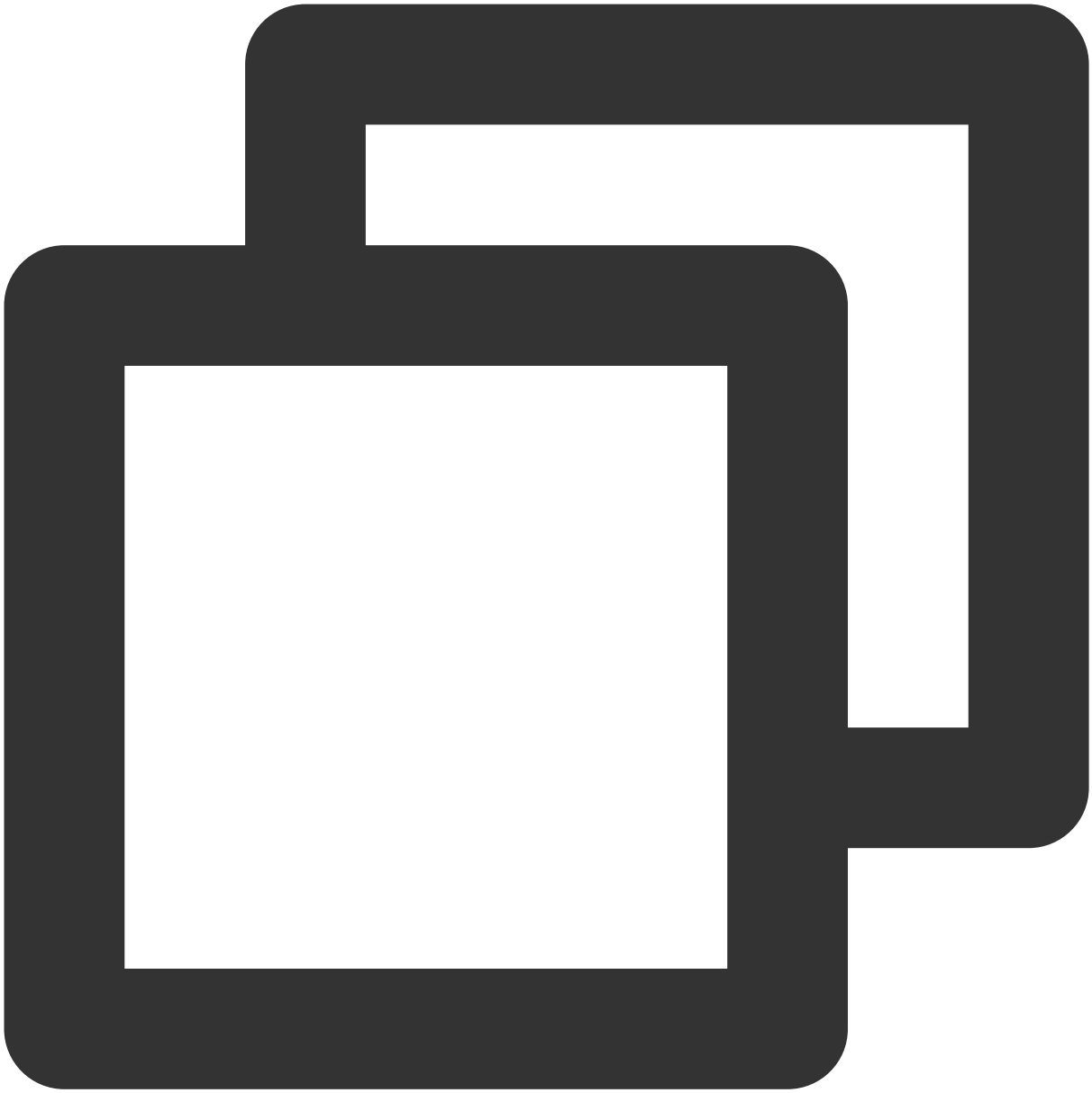
```
TUICallKit createInstance(Context context)
```

### setSelfInfo

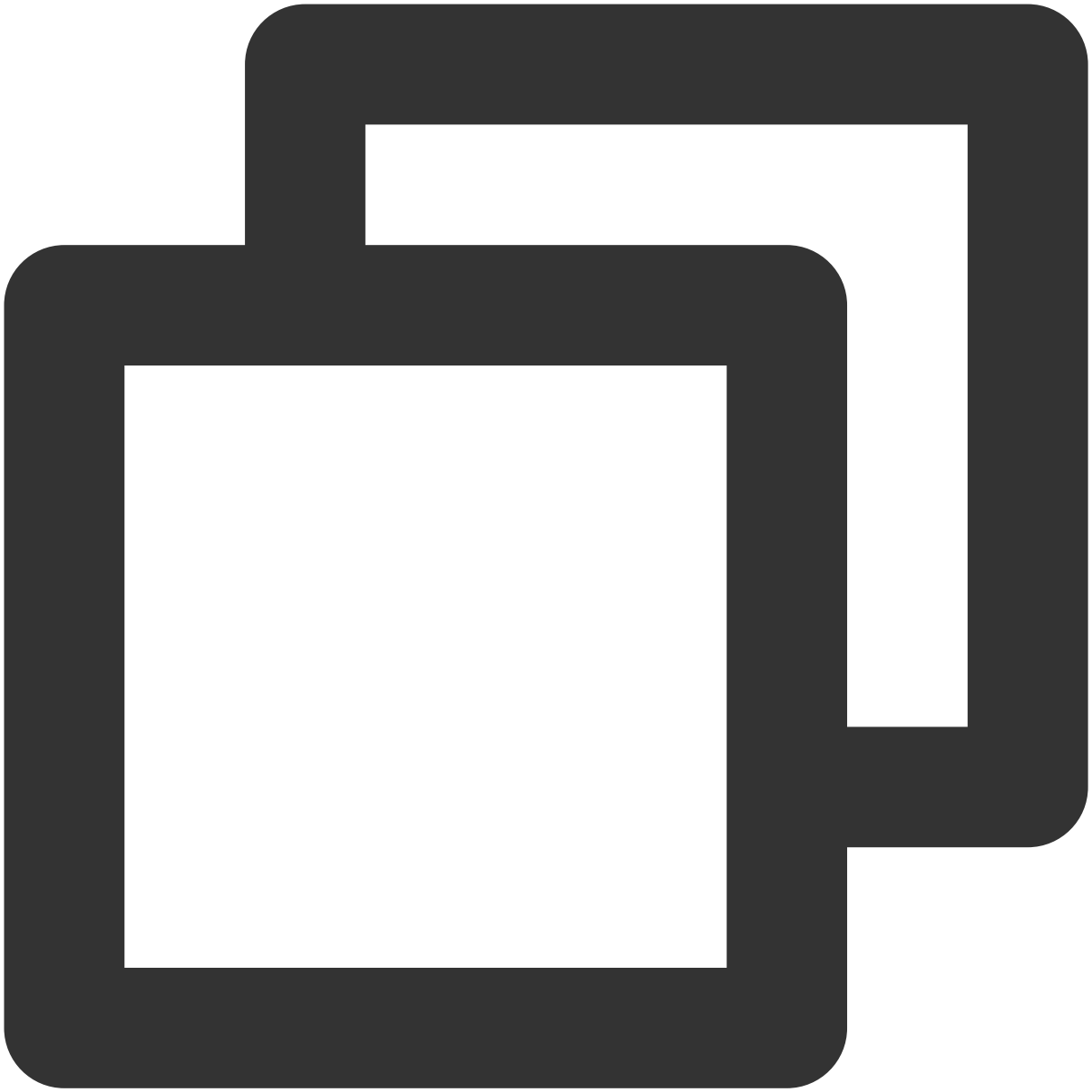
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.

Kotlin

Java



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



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

The parameters are described below:

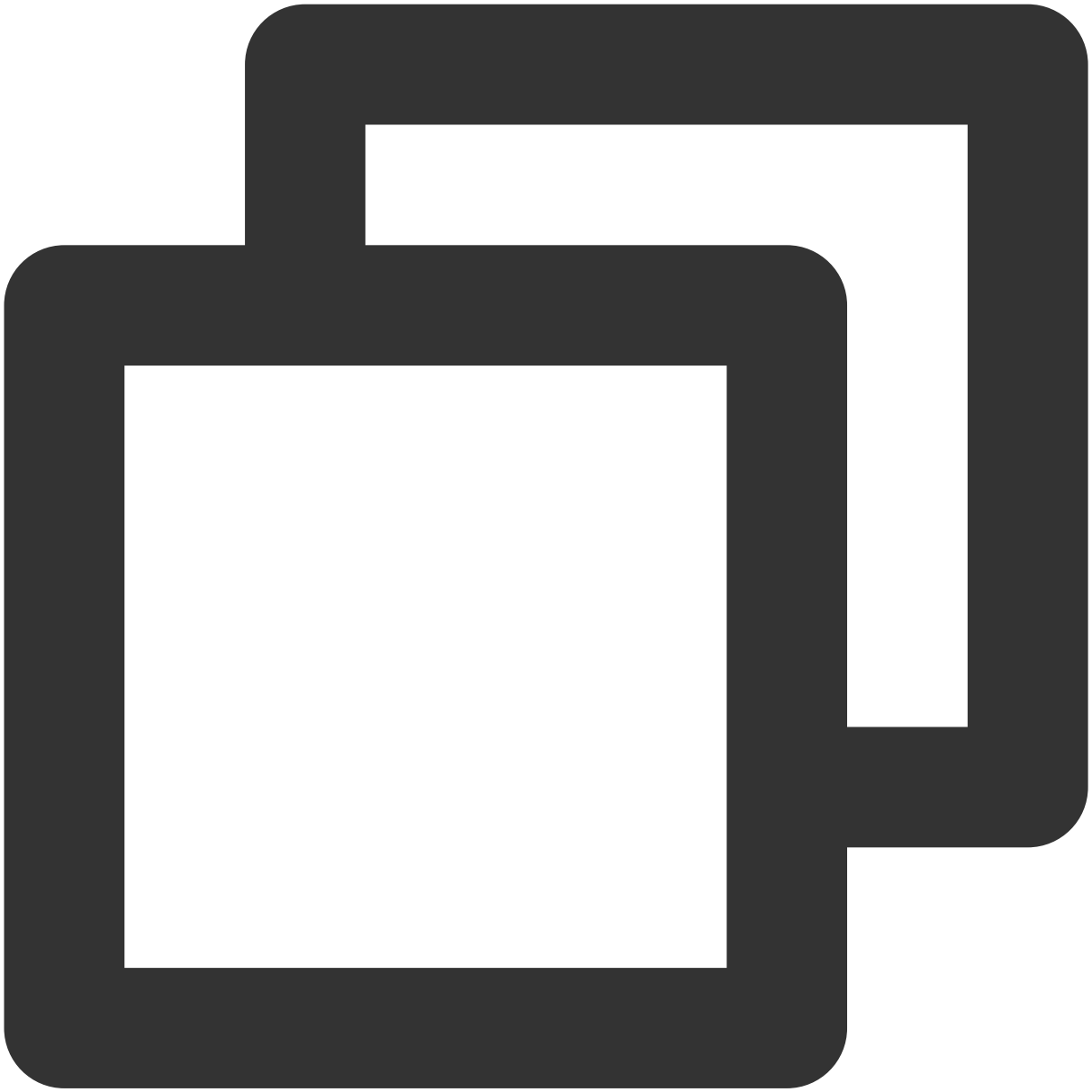
| Parameter | Type | Description |
| --- | --- | --- |
| nickname | String | The alias. |
| avatar | String | The profile photo. |

### call

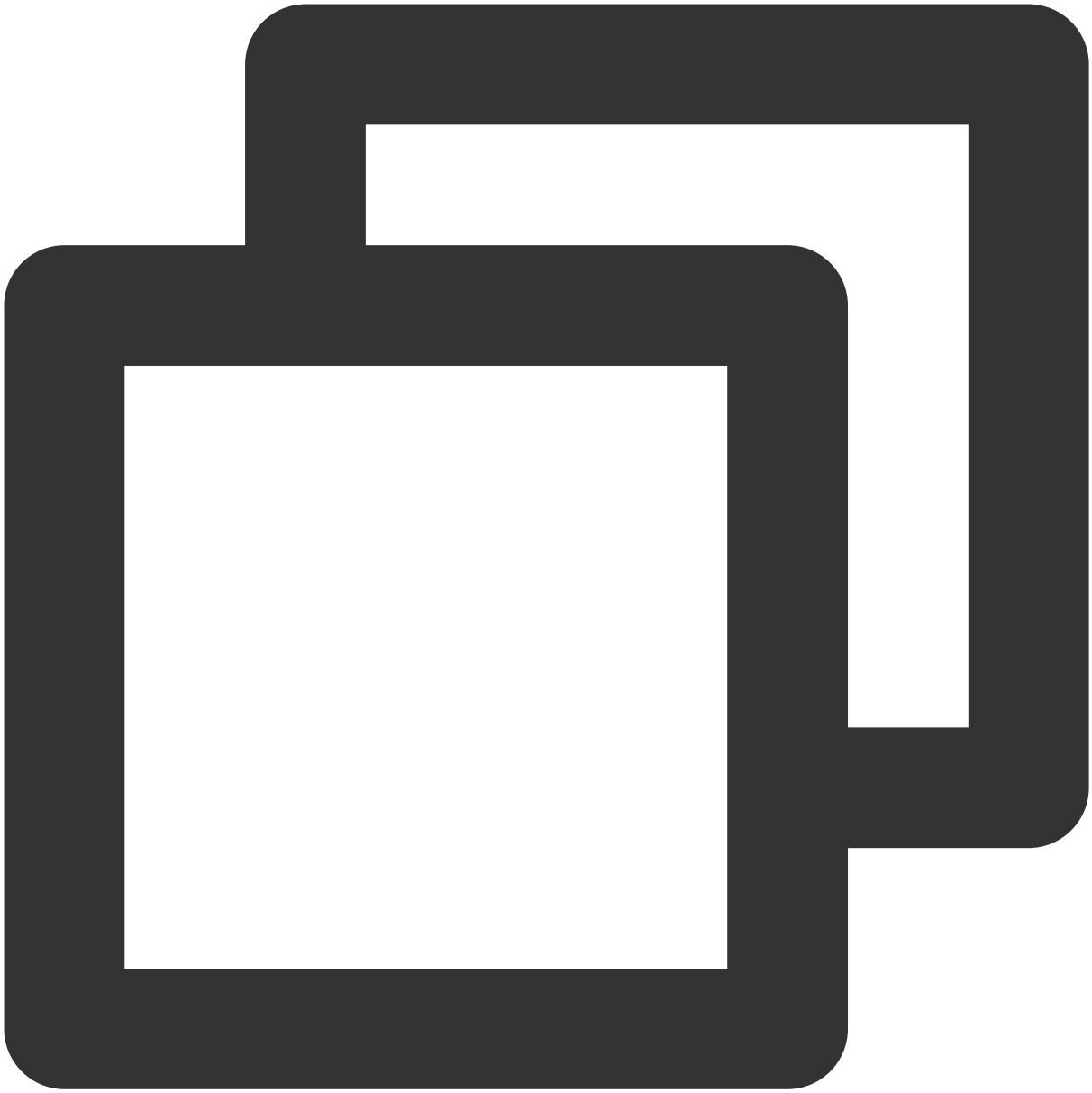
This API is used to make a (one-to-one) call.

Kotlin

Java



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



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

The parameters are described below:

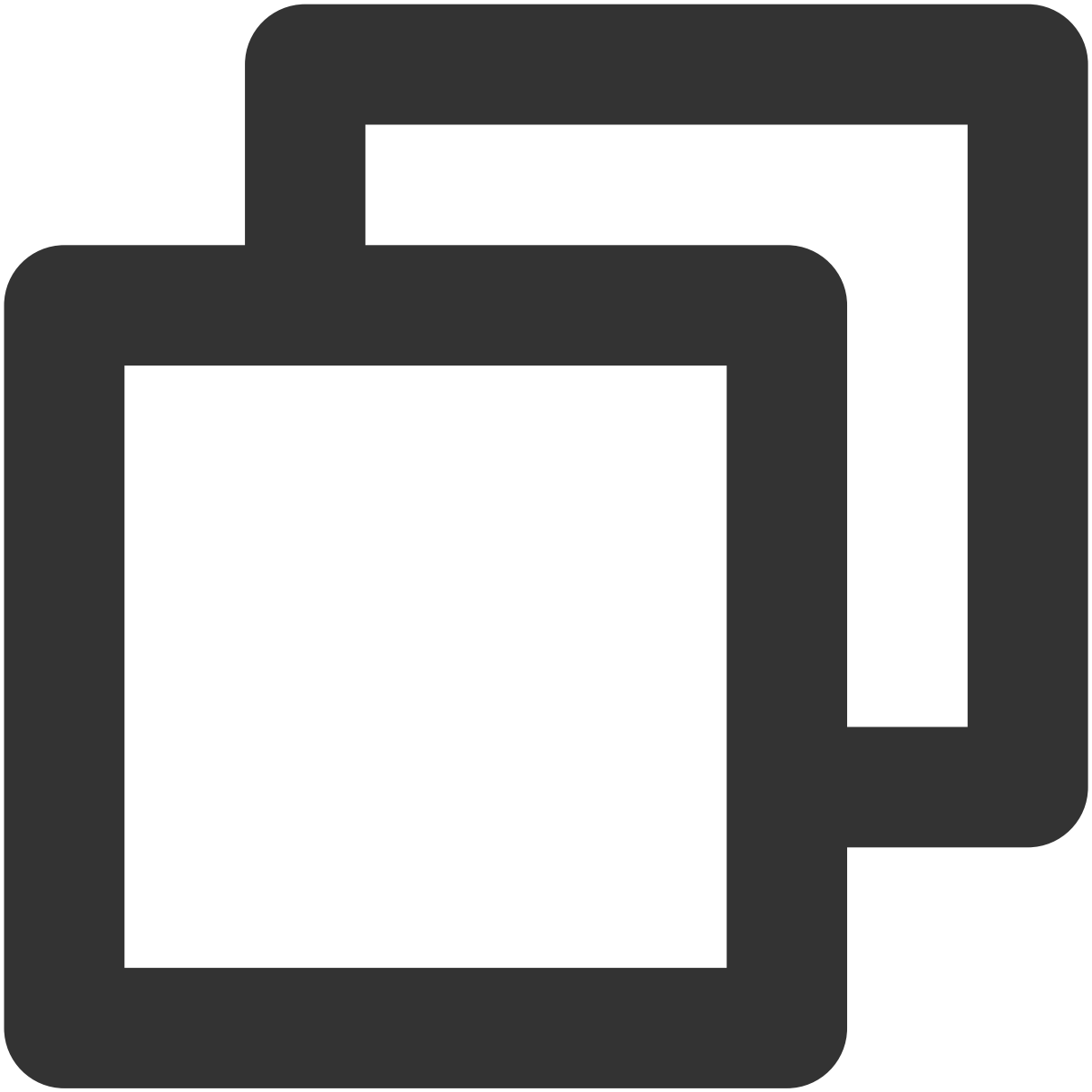
| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

### call

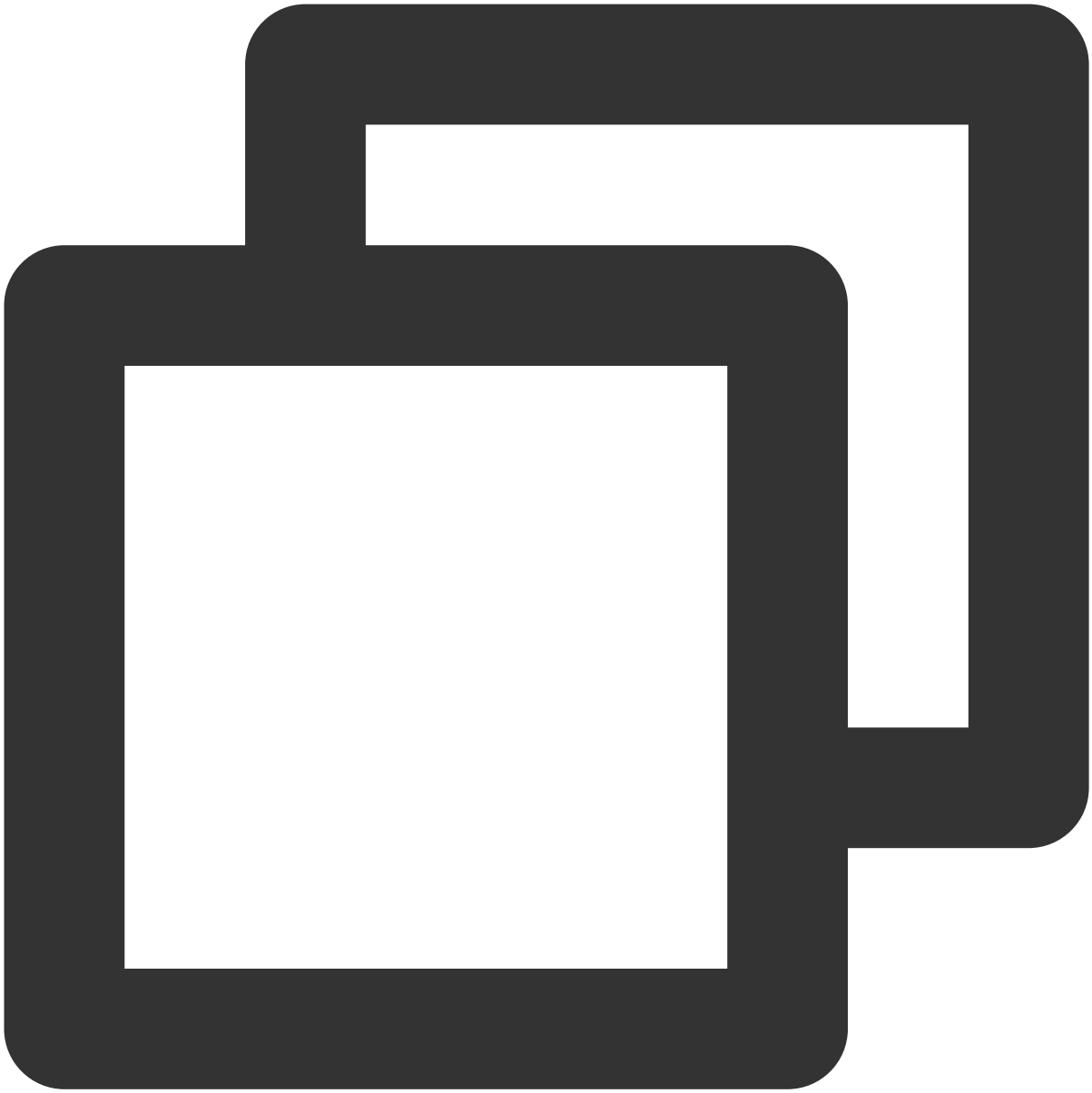
This API is used to make a (one-to-one) call, Support for custom room ID, call timeout, offline push content, etc.

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)
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |
| | | |

| | | |
|---|---|---|
| params | TUICallDefine.CallParams | Call extension parameters, such as roomID, call timeout, offline push info,etc |

### groupCall

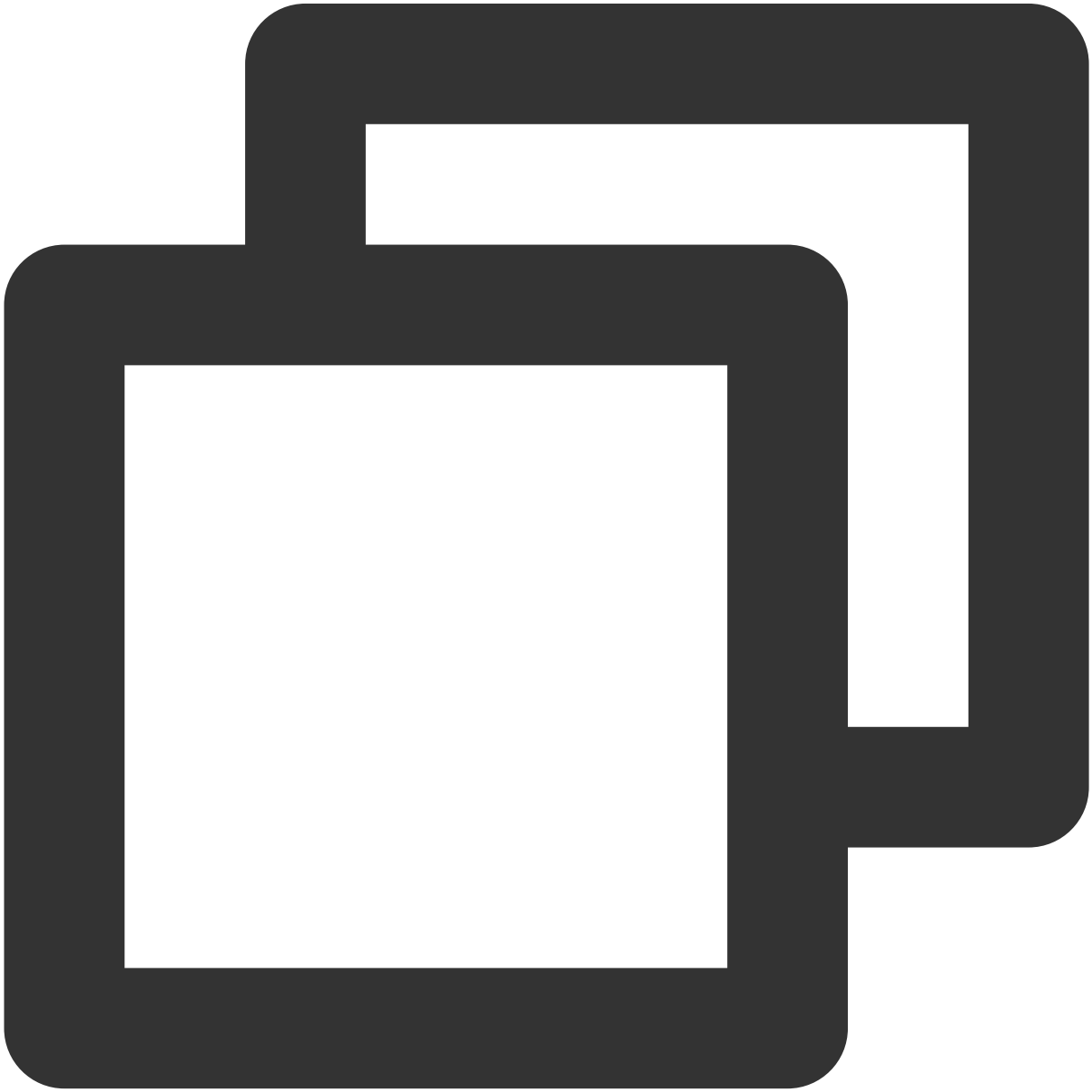
This API is used to make a group call.

**Notice:**

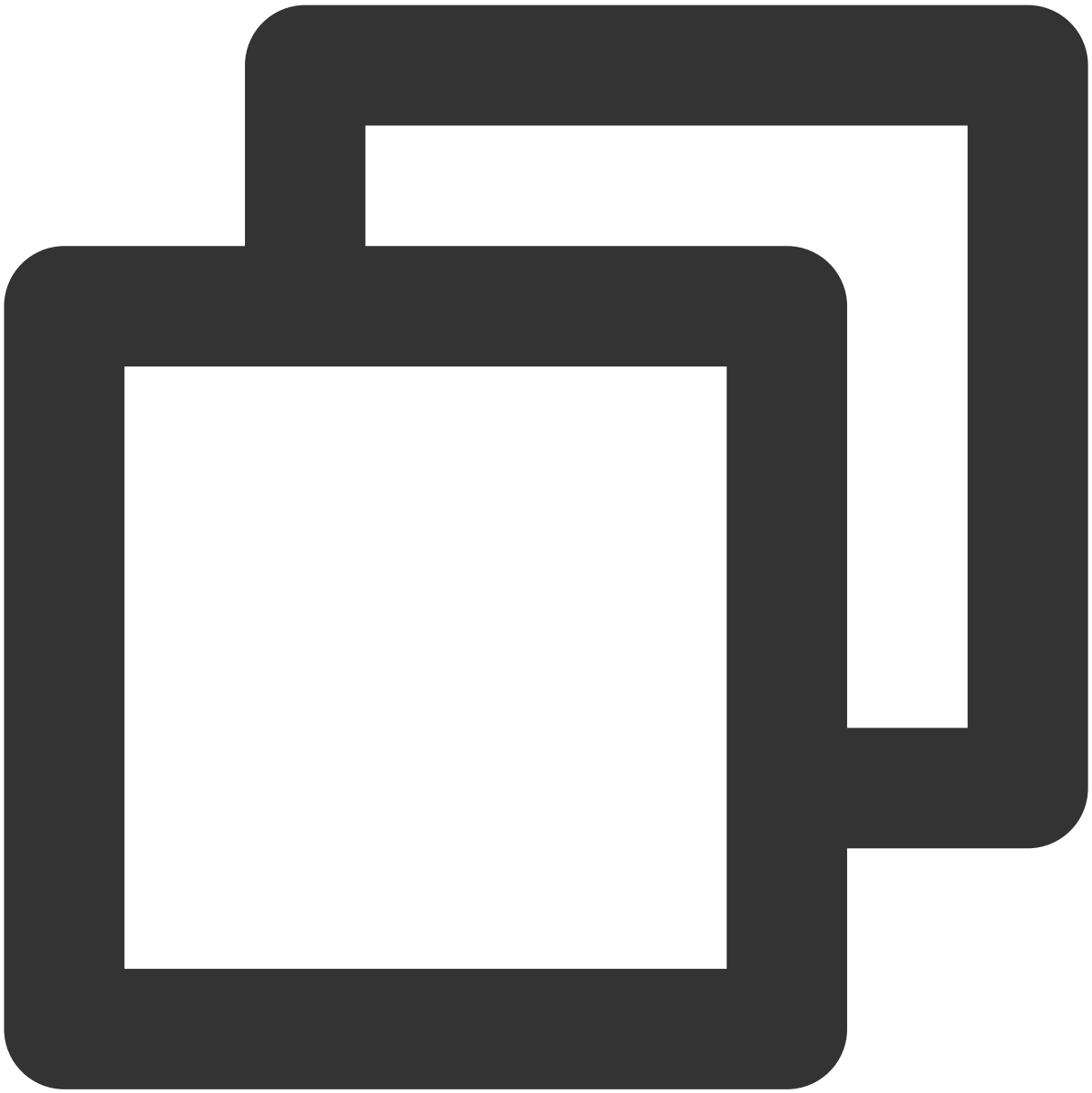
Before making a group call, you need to create an IM group first.

Kotlin

Java



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



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

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID. |
| userIdList | List | The target user IDs. |

| | | |
|---|---|---|
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

### groupCall

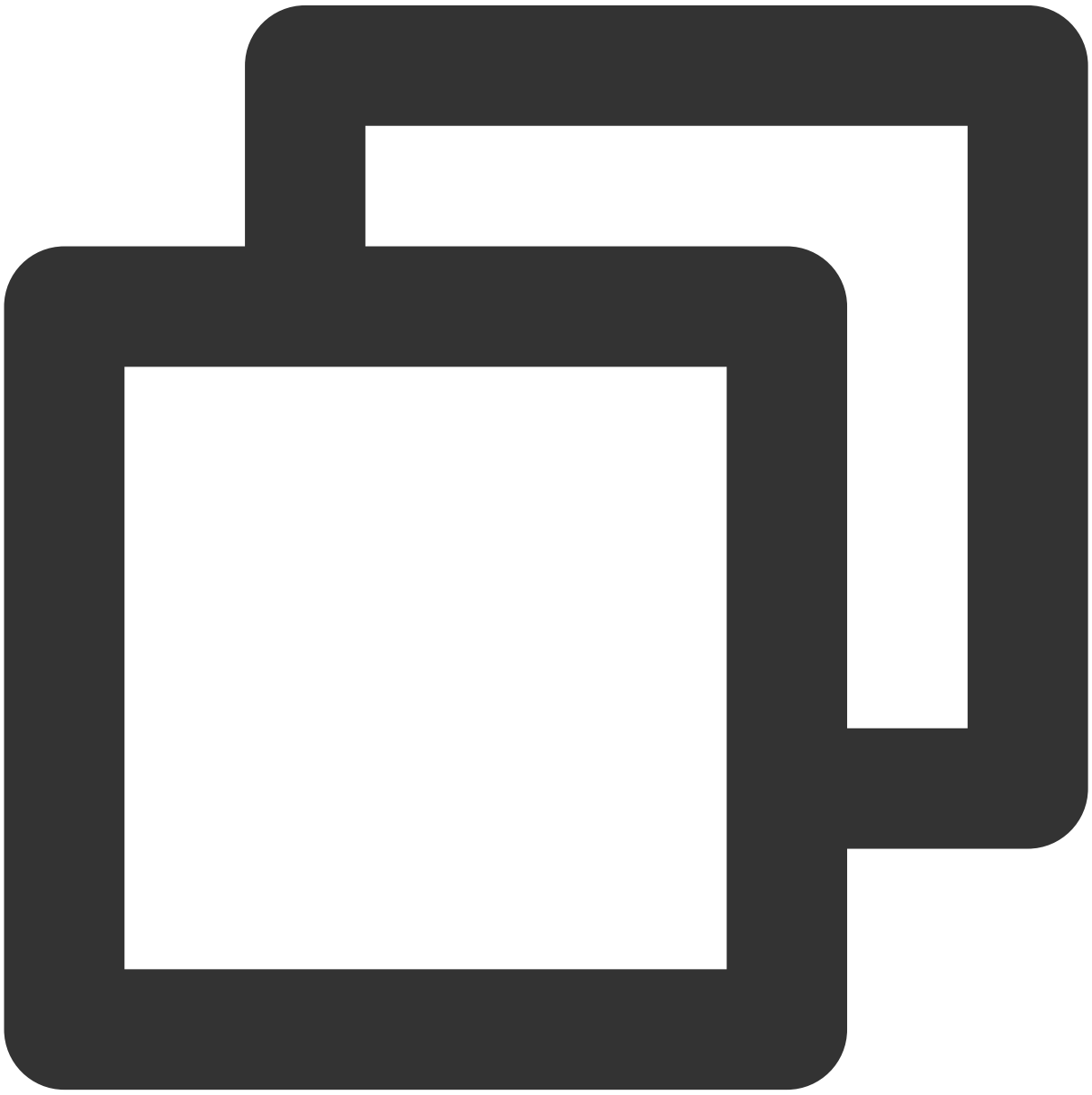
This API is used to make a group call, Support for custom room ID, call timeout, offline push content, etc.

**Notice:**

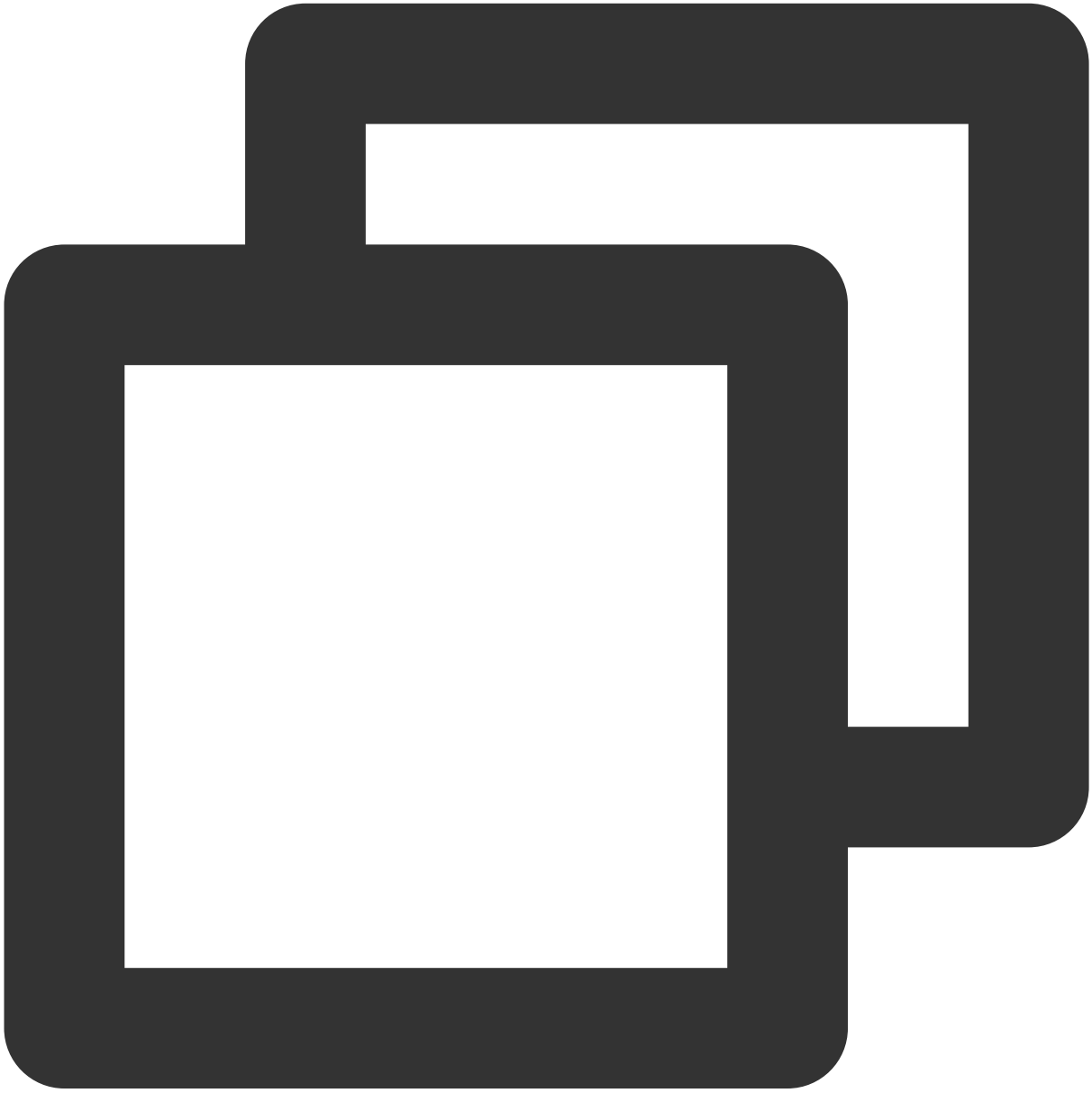
Before making a group call, you need to create an IM group first.

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);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID. |
| userIdList | List | The target user IDs. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |
| params | TUICallDefine.CallParams | Call extension parameters, such as roomID, call timeout, offline push info,etc |

### joinInGroupCall

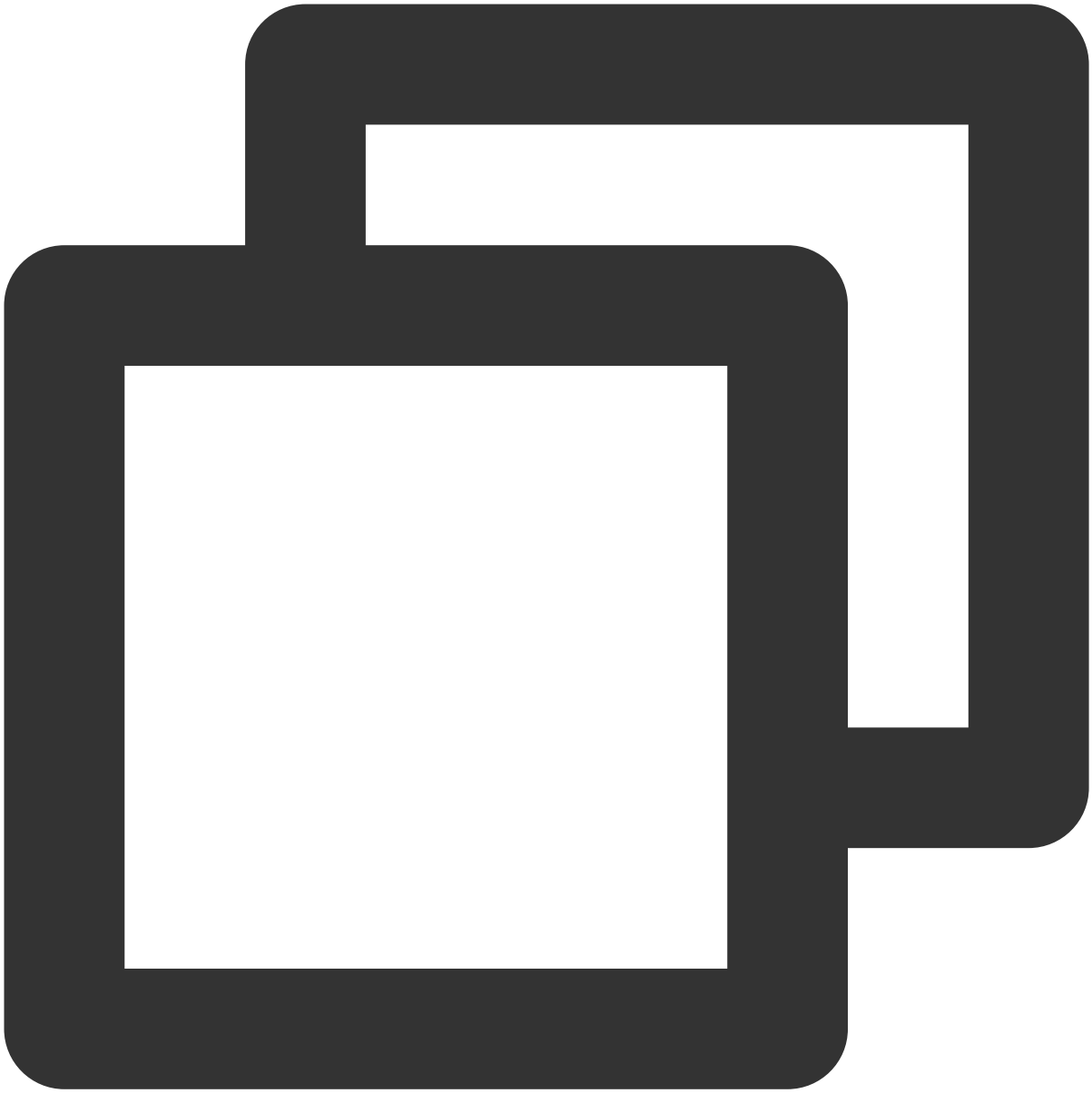
This API is used to join a group call.

**Notice:**

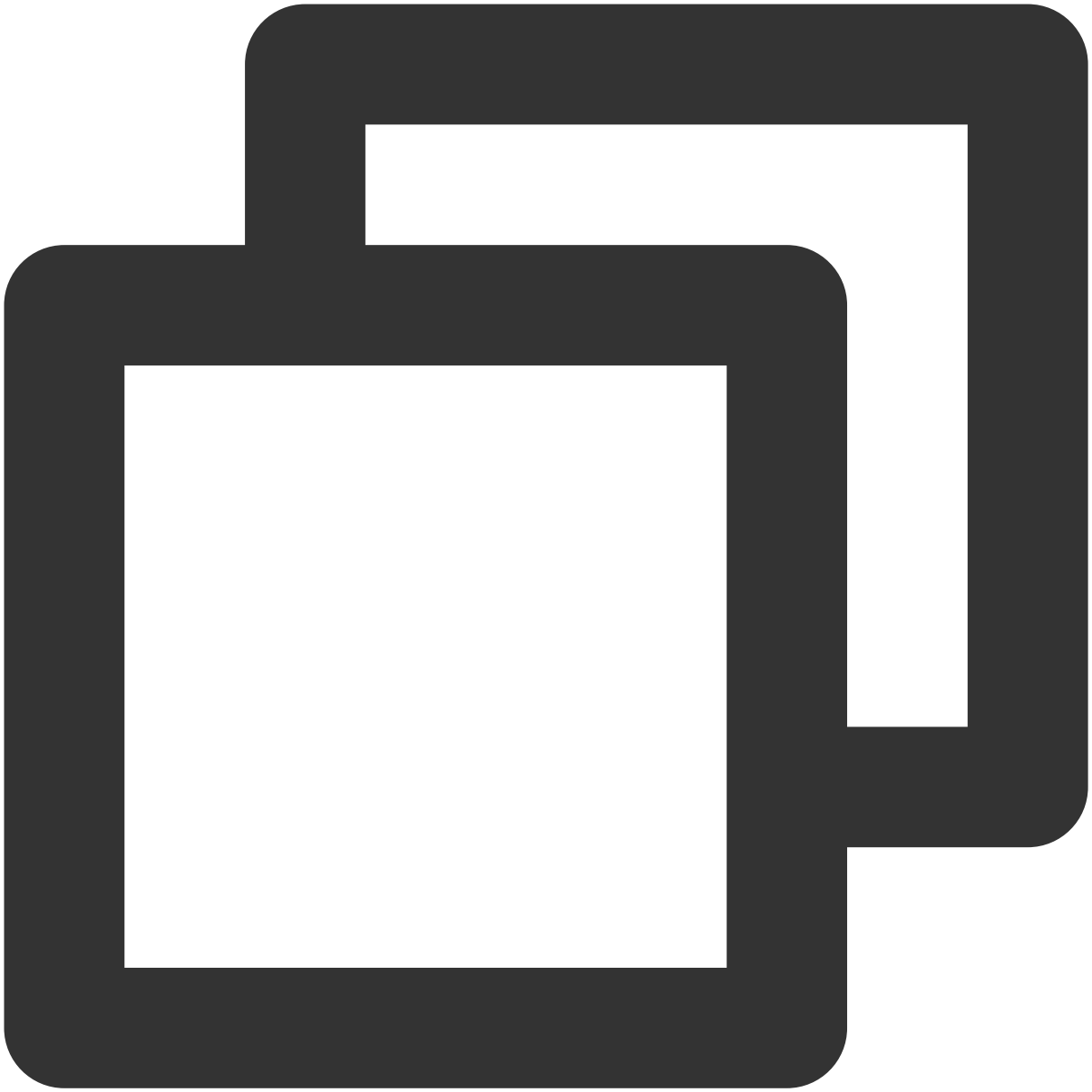
Before joining a group call, you need to create or join an IM group in advance, and there are already users in the group who are in the call.

Kotlin

Java



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



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

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUICommonDefine.RoomId | The room ID. |
| groupId | String | The group ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

### setCallingBell

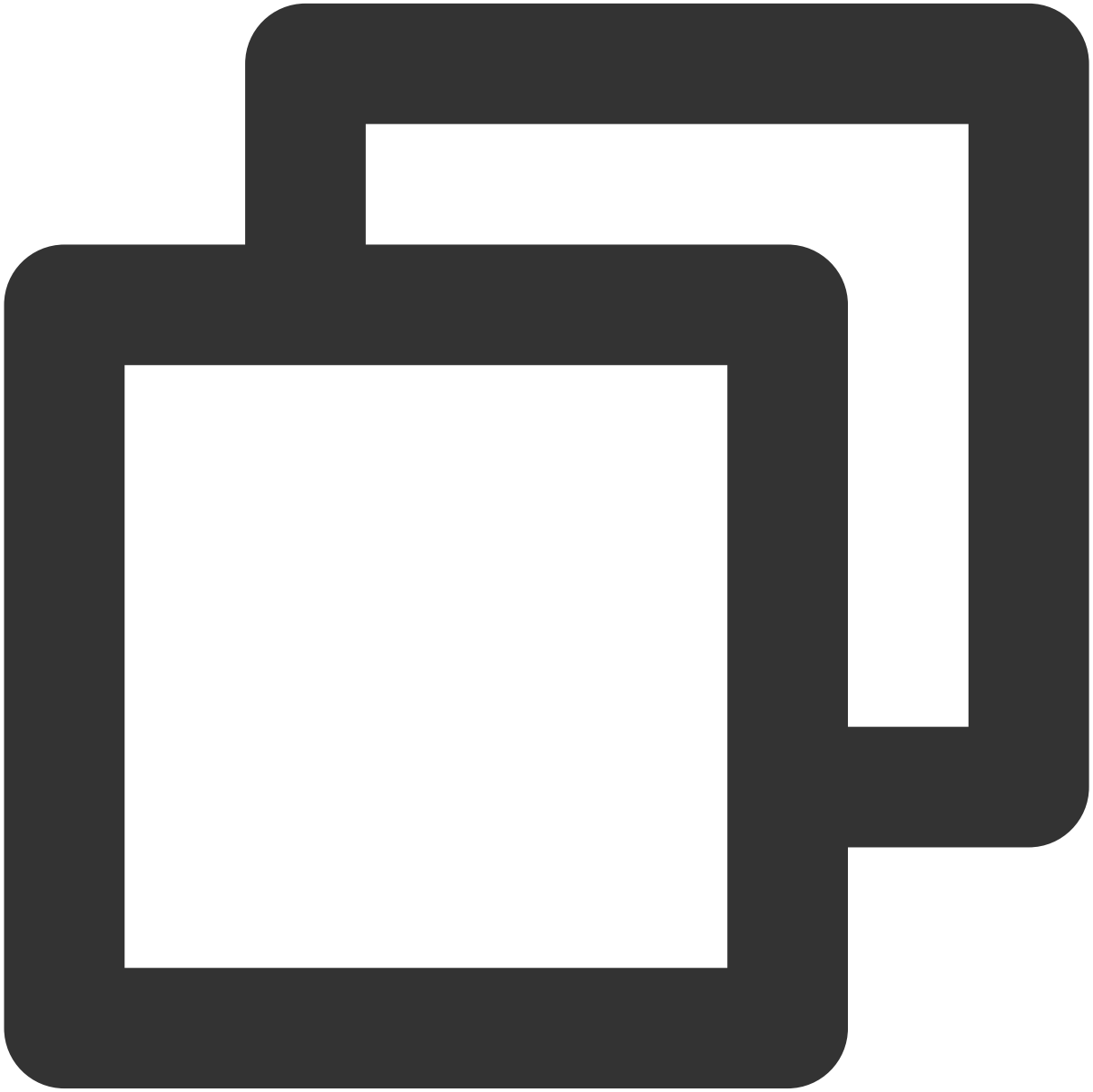
This API is used to set the ringtone. `filePath` must be an accessible local file URL.

The ringtone set is associated with the device and does not change with user.

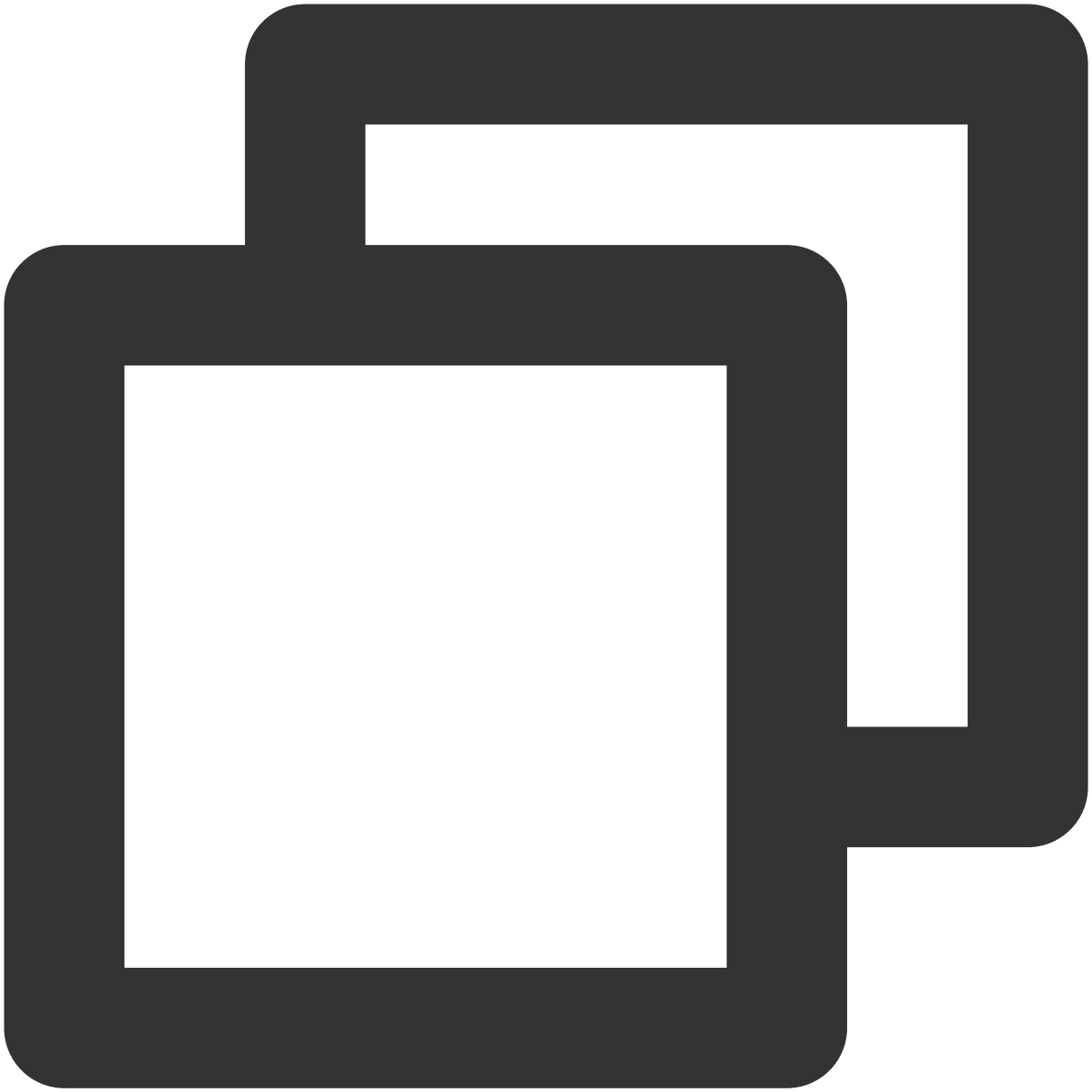
To reset the ringtone, pass in an empty string for `filePath` .

Kotlin

Java



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



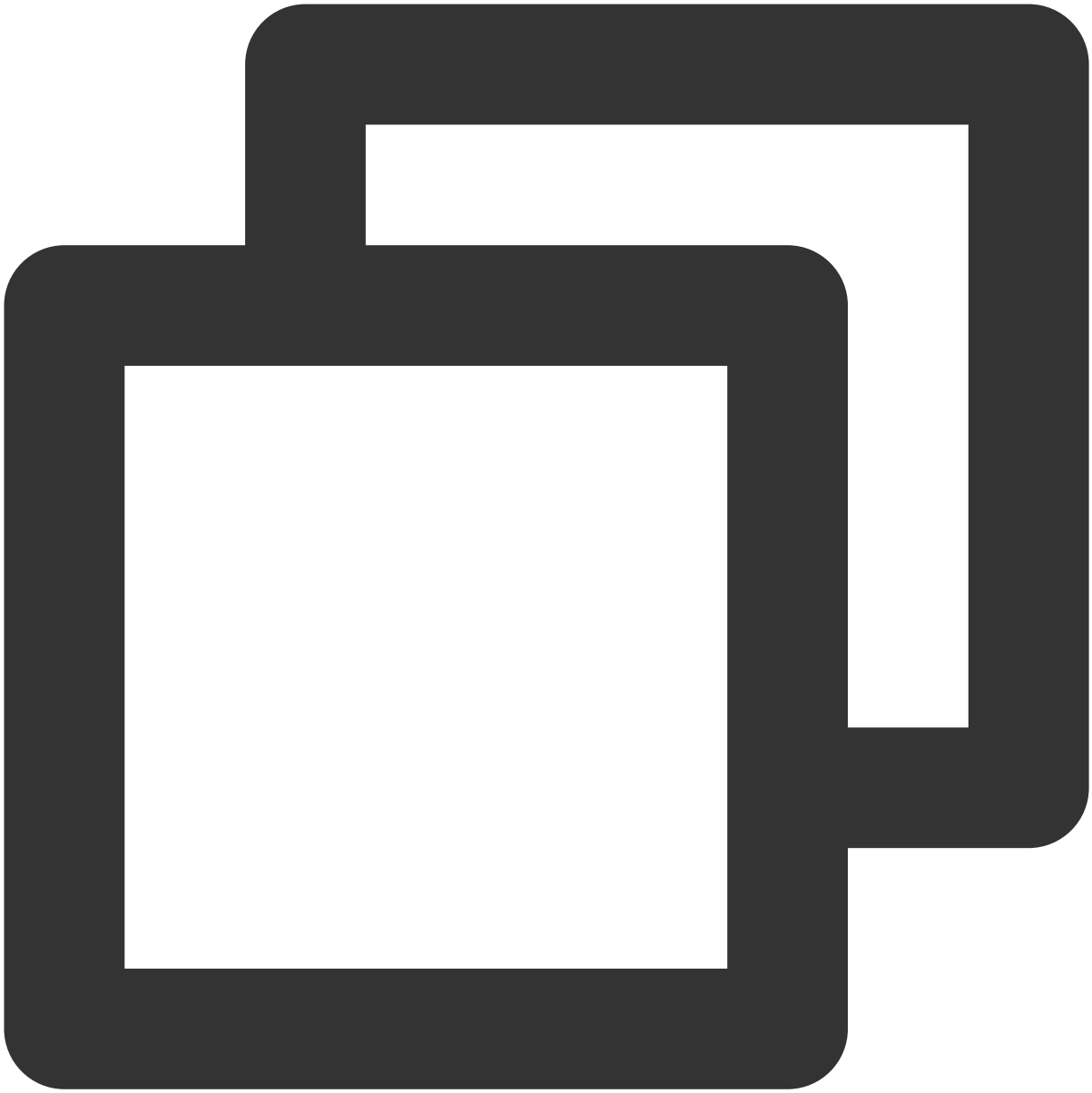
```
void setCallingBell(String filePath);
```

### enableMuteMode

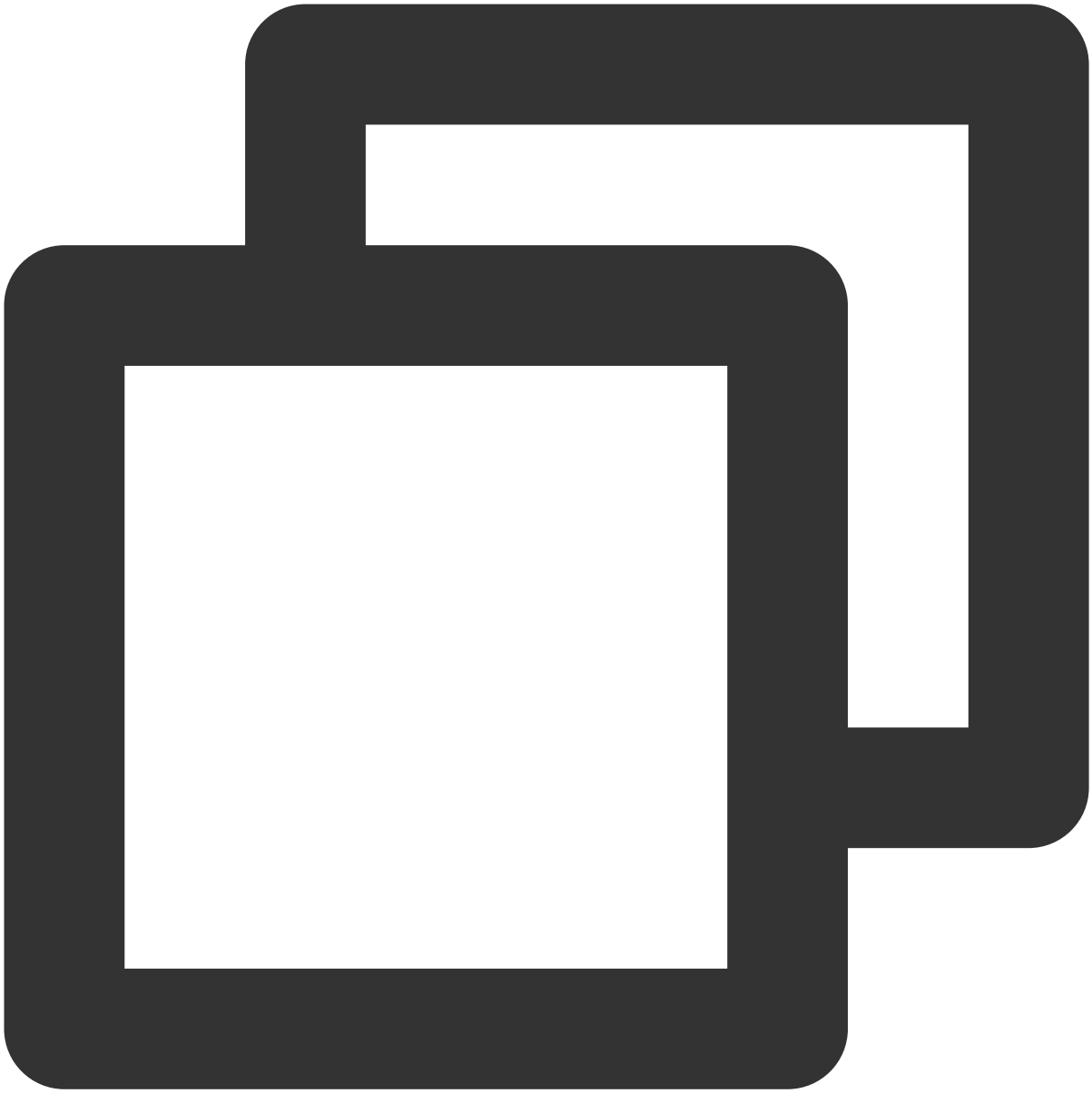
This API is used to set whether to turn on the mute mode.

Kotlin

Java



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



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

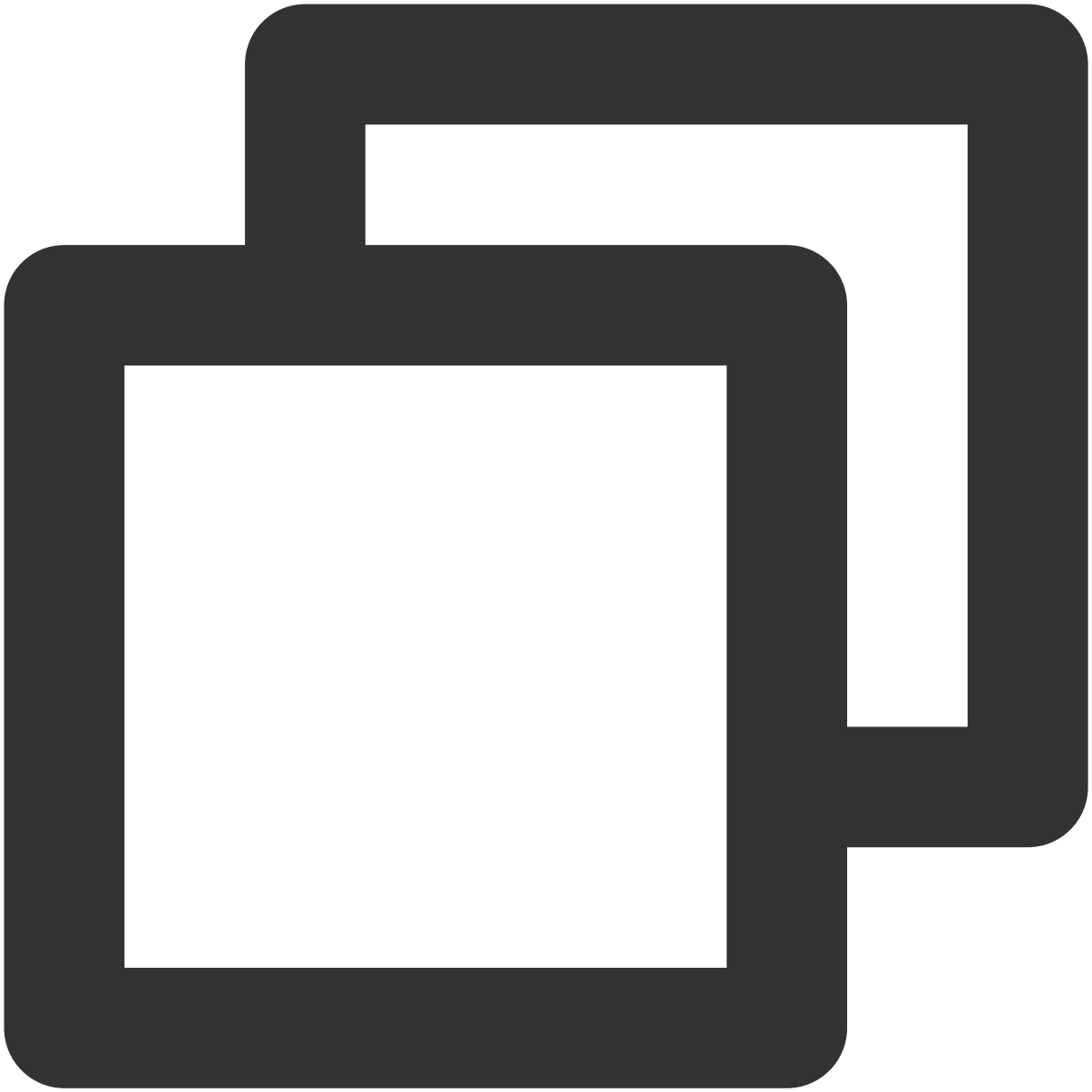
### enableFloatWindow

This API is used to set whether to enable floating windows.

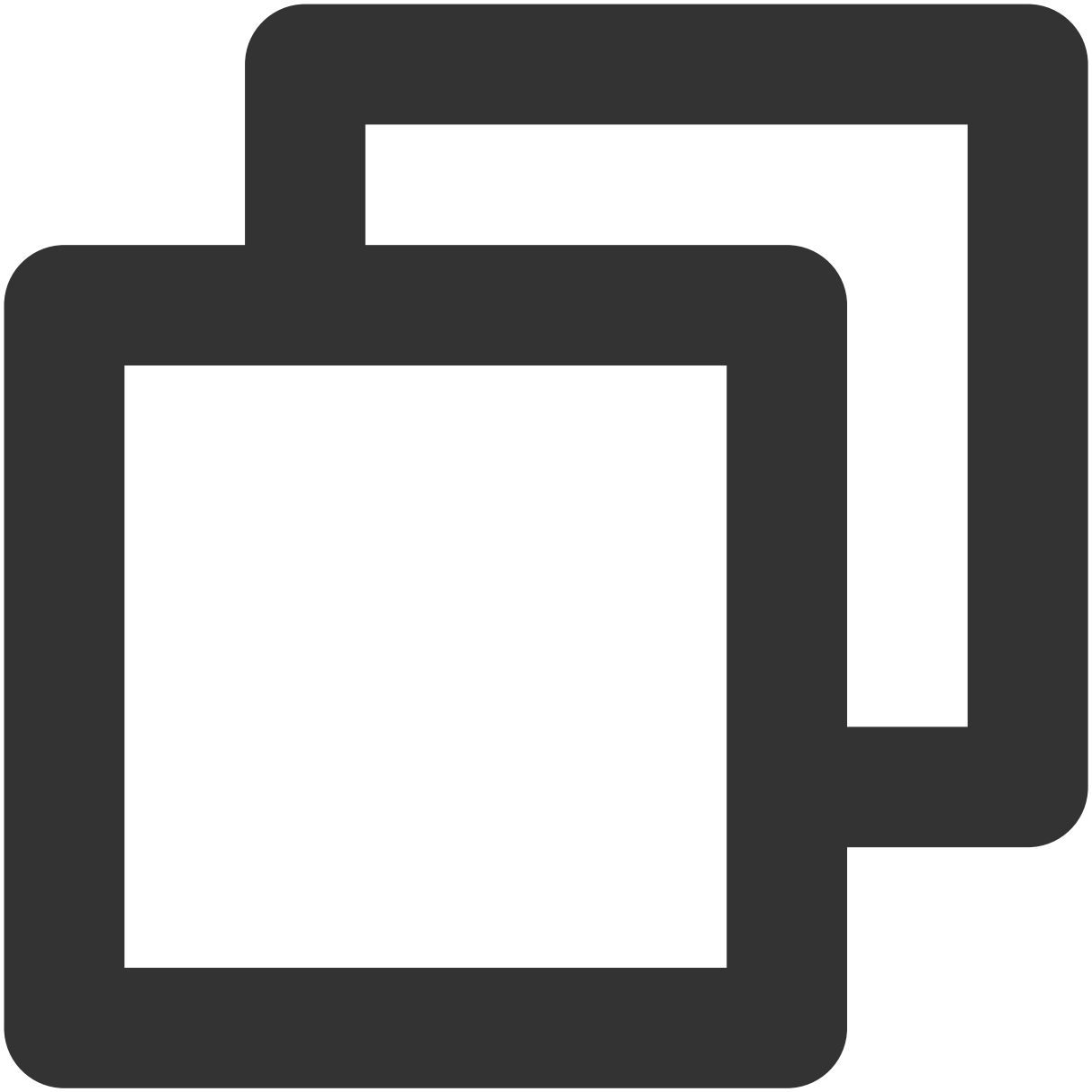
The default value is `false`, and the floating window button in the top left corner of the call view is hidden. If it is set to `true`, the button will become visible.

Kotlin

Java



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



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

# TUICallEngine

Last updated：2023-08-11 15:32:40

## TUICallEngine APIs

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

## Overview

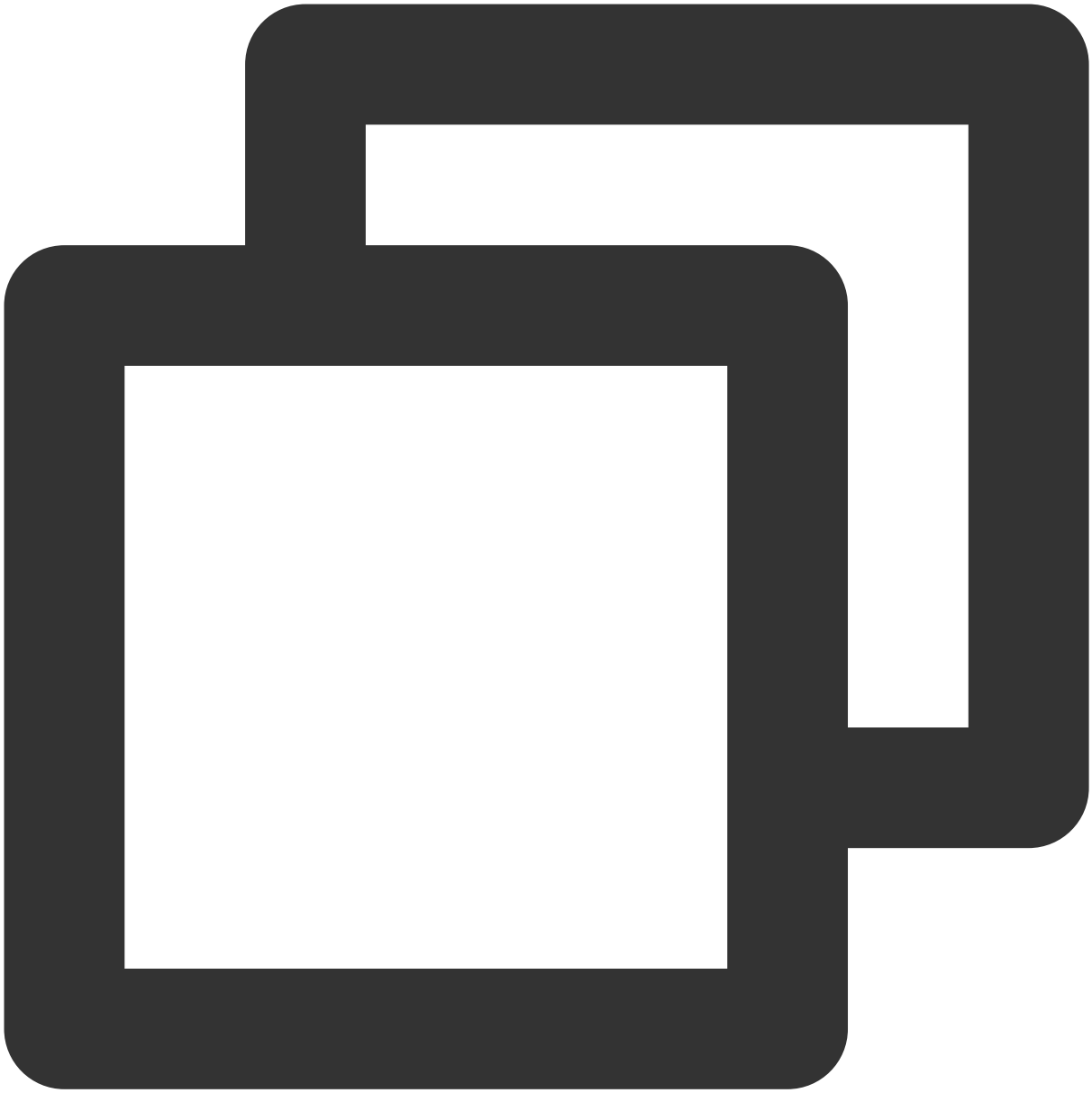
| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallEngine` instance (singleton mode). |
| destroyInstance | Terminates a `TUICallEngine` instance (singleton mode). |
| init | Authenticates the basic audio/video call capabilities. |
| addObserver | Registers an event listener. |
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Changes the call type, for example, from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |

| | |
|---|---|
| openCamera | Turns the camera on. |
| closeCamera | Turns the camera off. |
| switchCamera | Switches between the front and rear cameras. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver or speaker). |
| setSelfInfo | Sets the alias and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for `TUICallEngine` (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |
| setVideoEncoderParams | Set the encoding parameters of video encoder. |
| getTRTCCloudInstance | Advanced features. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

# Details

### createInstance

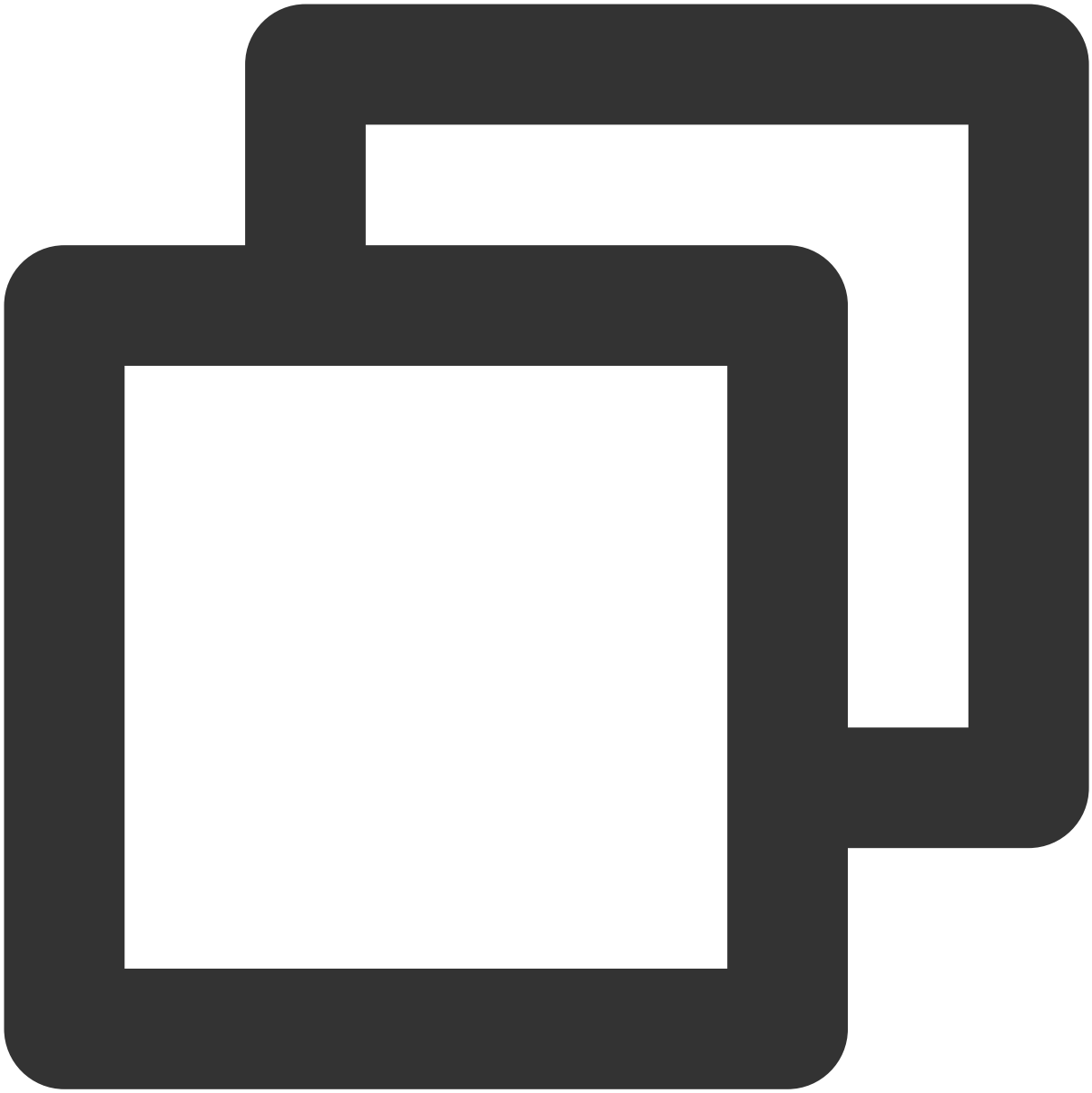
This API is used to create a `TUICallEngine` singleton.



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

### destroyInstance

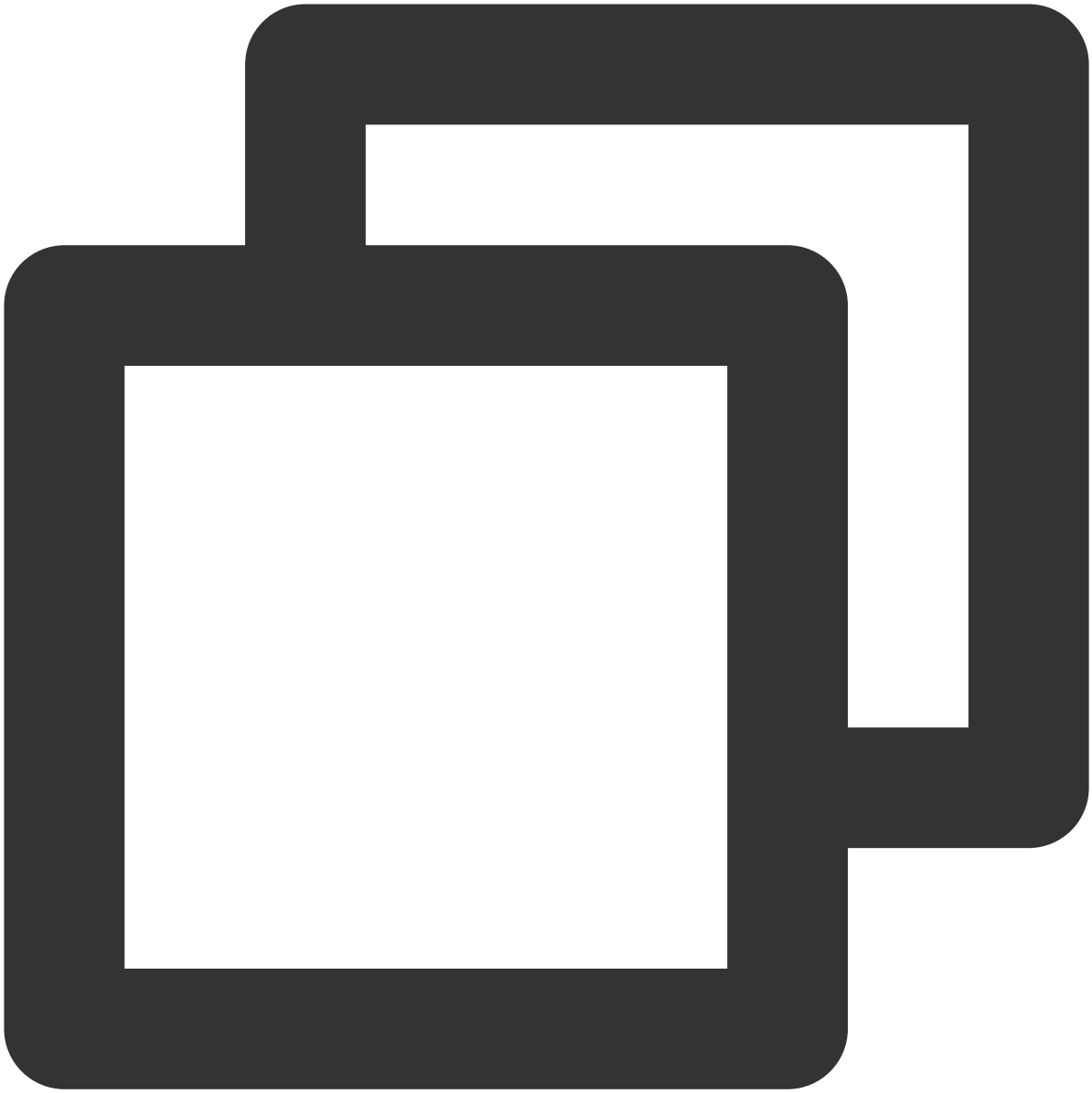
This API is used to terminate a `TUICallEngine` singleton.



```
void destroyInstance();
```

### Init

This API is used to initialize `TUICallEngine` . Call it to authenticate the call service and perform other required actions before you call other APIs.



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

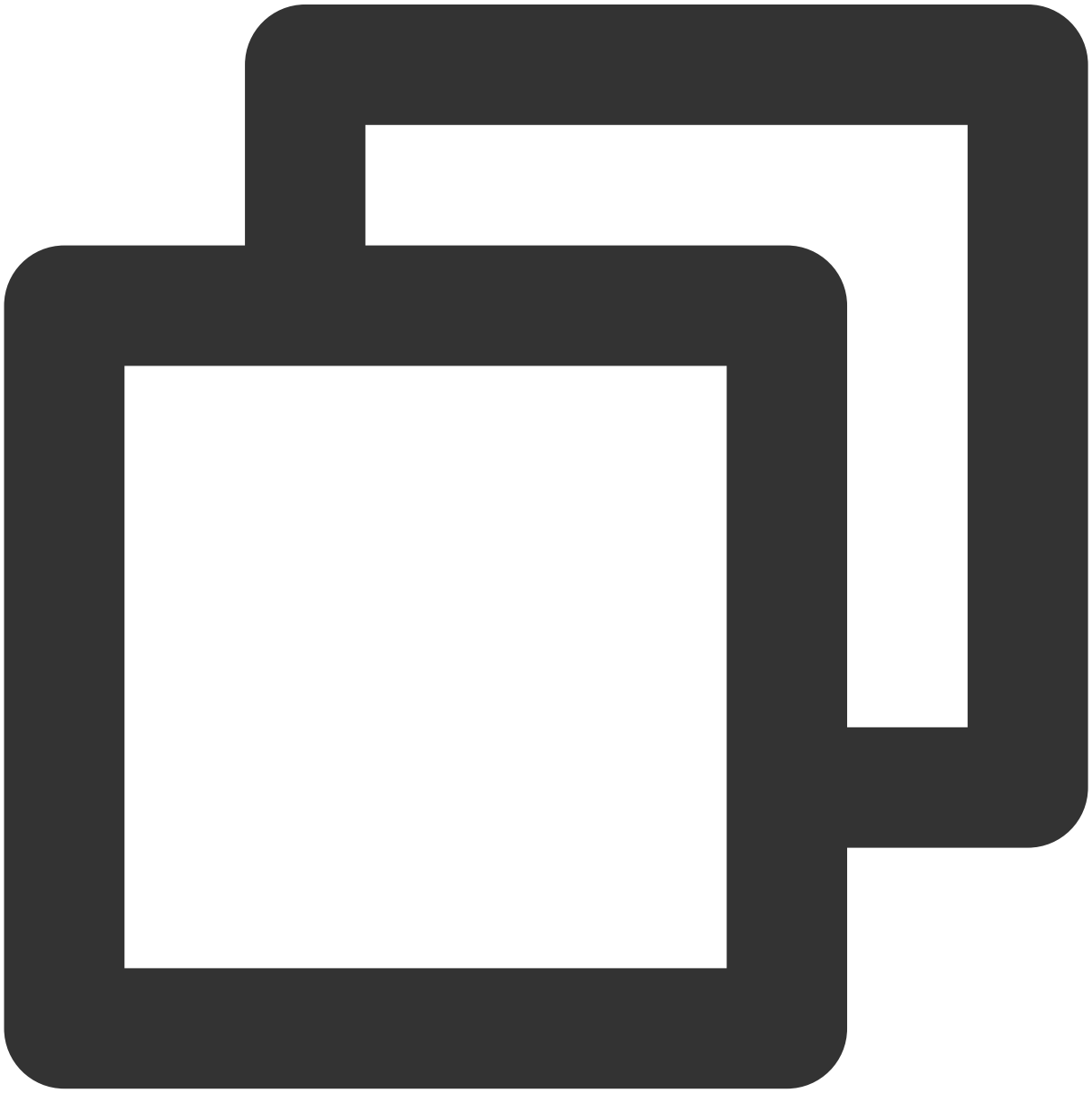
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| sdkAppId | int | You can view `SDKAppID` in Application Management > **Application Info** of the TRTC console. |
| userId | String | The ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and |

| | | |
|---|---|---|
| | | underscores (_). |
| userSig | String | Tencent Cloud's proprietary security signature. For how to calculate and use it, see UserSig. |
| callback | TUICommonDefine.Callback | The initialization callback. `onSuccess` indicates initialization is successful. |

### addObserver

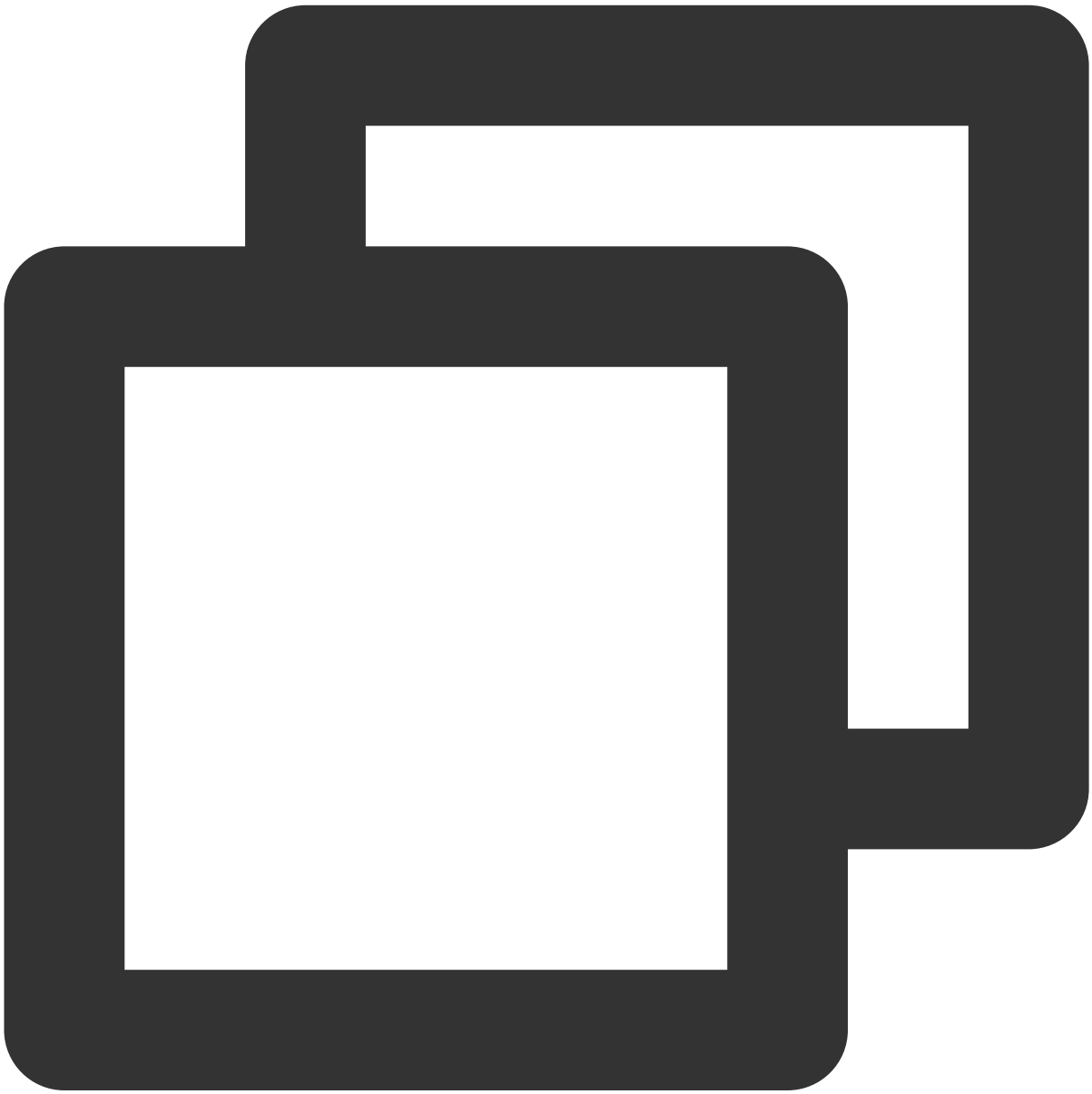
This API is used to register an event listener to listen for `TUICallObserver` events.



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

### removeObserver

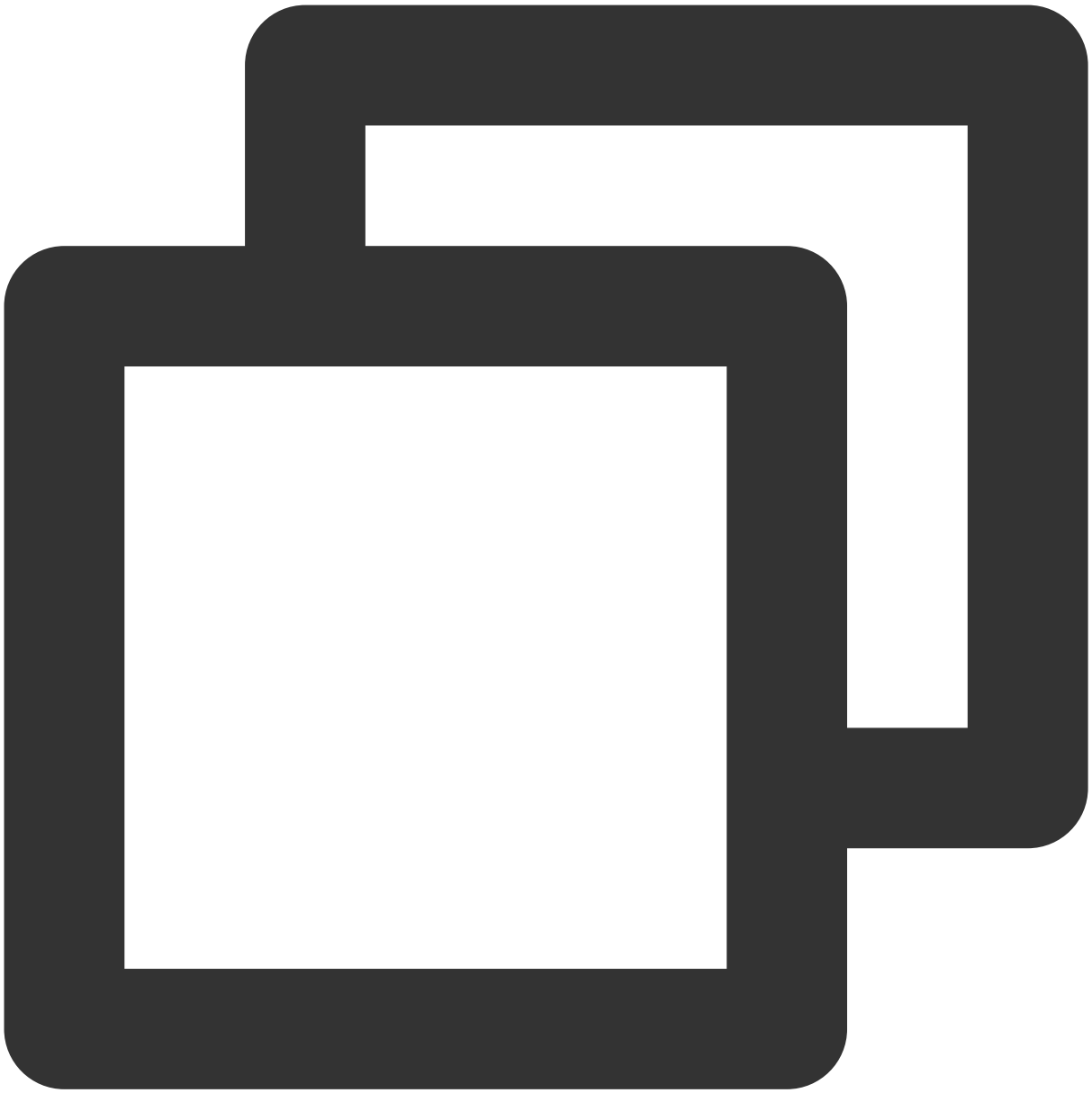
This API is used to unregister an event listener.



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

### call

This API is used to make a (one-to-one) call.



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

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

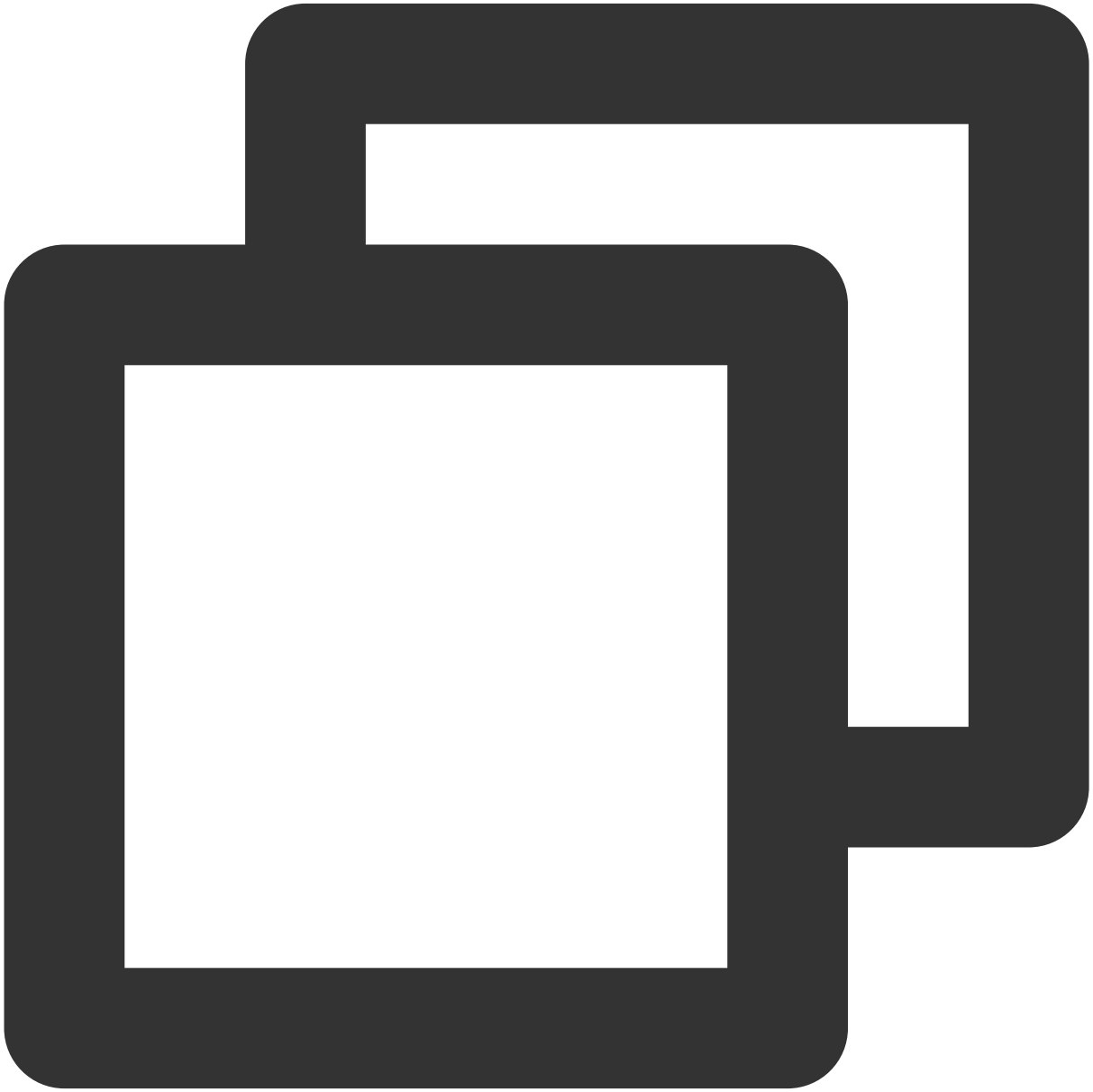
| | | |
|---|---|---|
| params | TUICallDefine.CallParams | An additional parameter, such as roomID, call timeout, offline push info,etc |

### groupCall

This API is used to make a group call.

**Notice：**

Before making a group call, you need to create an IM group first.



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

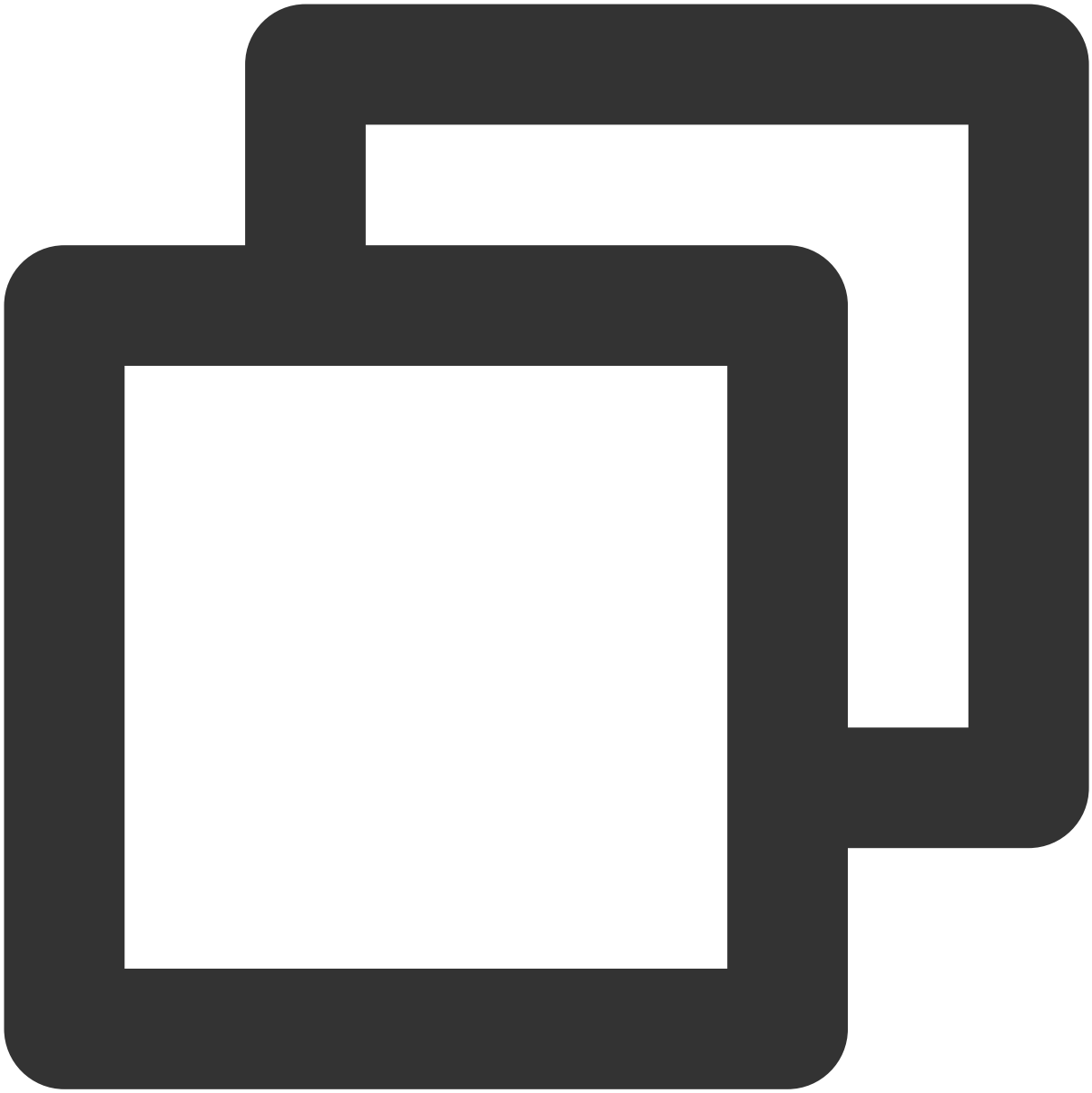
```
                TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID. |
| userIdList | List | The target user IDs. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |
| params | TUICallDefine.CallParams | An additional parameter. such as roomID, call timeout, offline push info,etc |

### accept

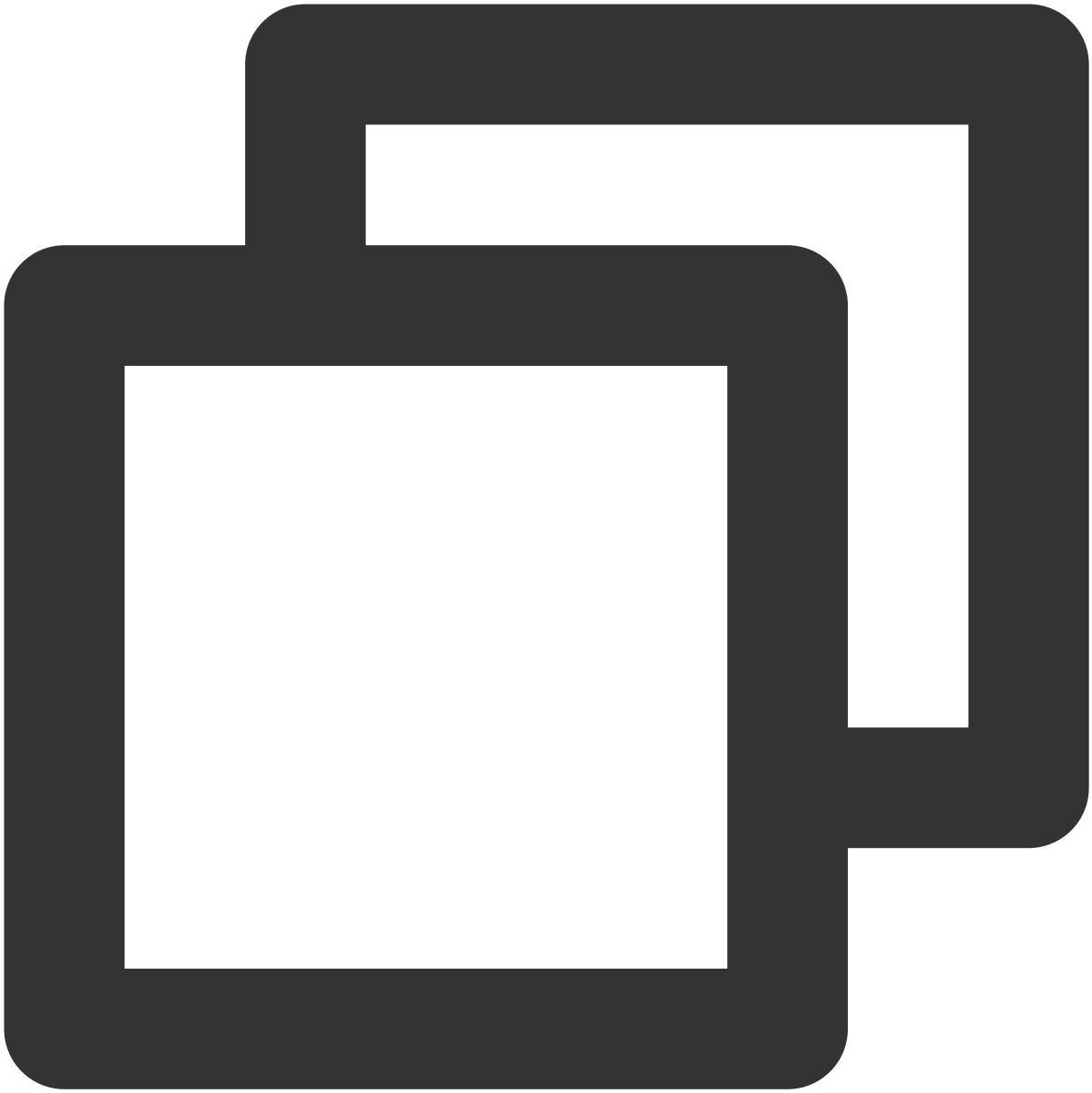
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



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

### reject

This API is used to reject a call. After receiving the `onCallReceived()` callback, you can call this API to reject the call.

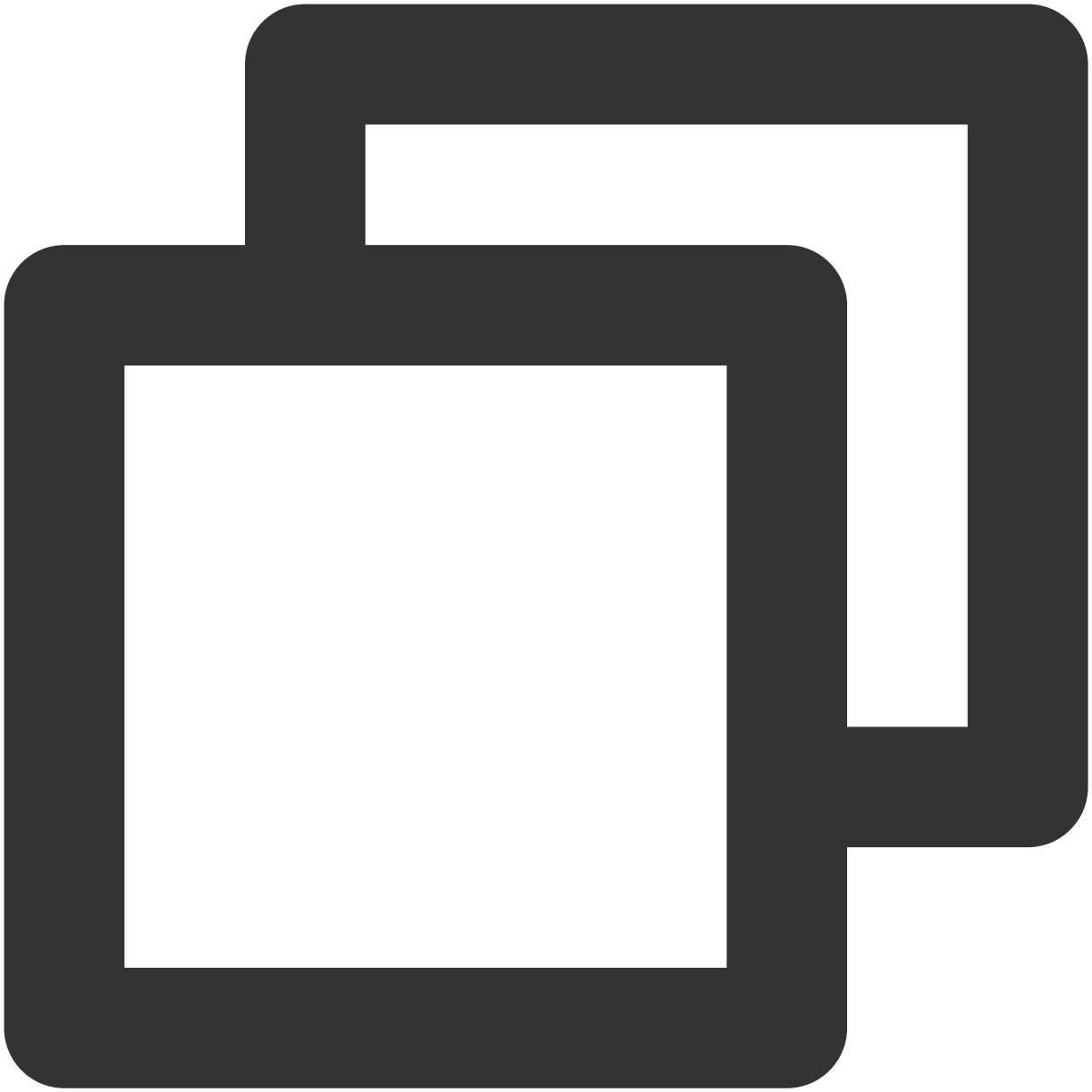


```
void reject(TUICommonDefine.Callback callback);
```

### ignore

This API is used to ignore a call. After receiving the `onCallReceived()` , you can call this API to ignore the call. The caller will receive the `onUserLineBusy` callback.

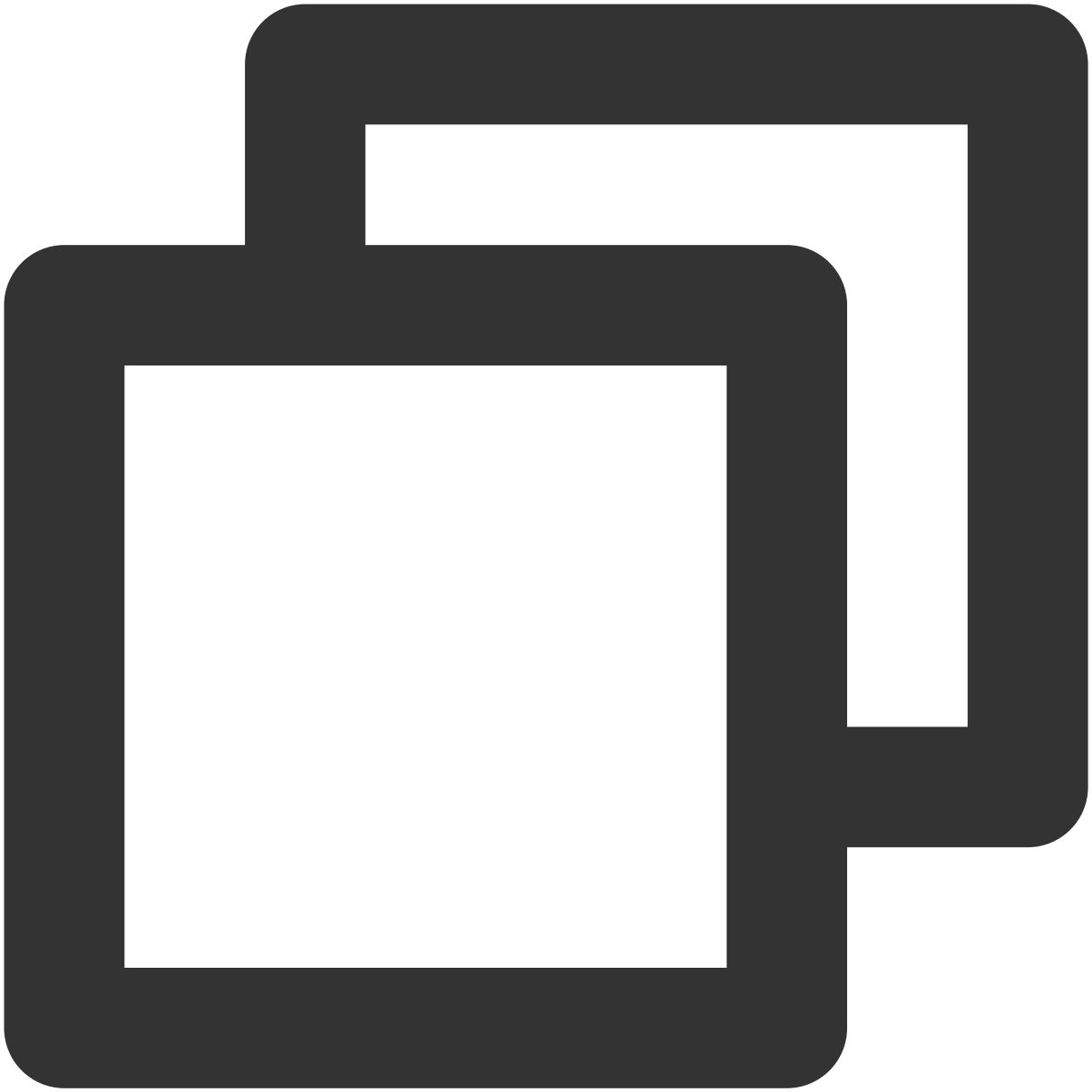
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



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

### hangup

This API is used to end a call.

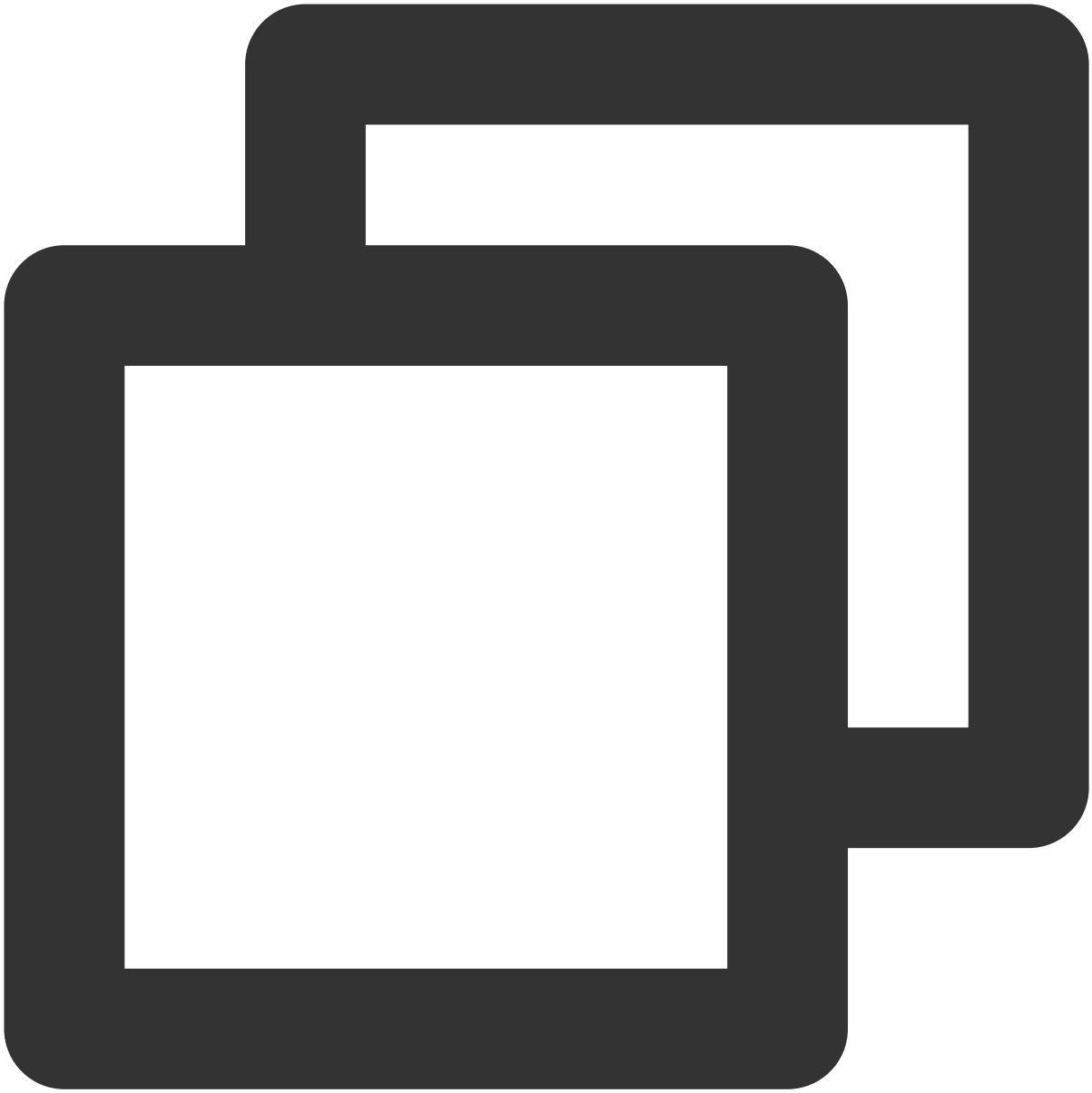


```
void hangup(TUICommonDefine.Callback callback);
```

### inviteUser

This API is used to invite users to the current group call.

This API is called by a participant of a group call to invite new users.



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

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userIdList | List | The target user IDs. |
| params | TUICallDefine.CallParams | An additional parameter. such as roomID, call timeout, offline push info,etc |

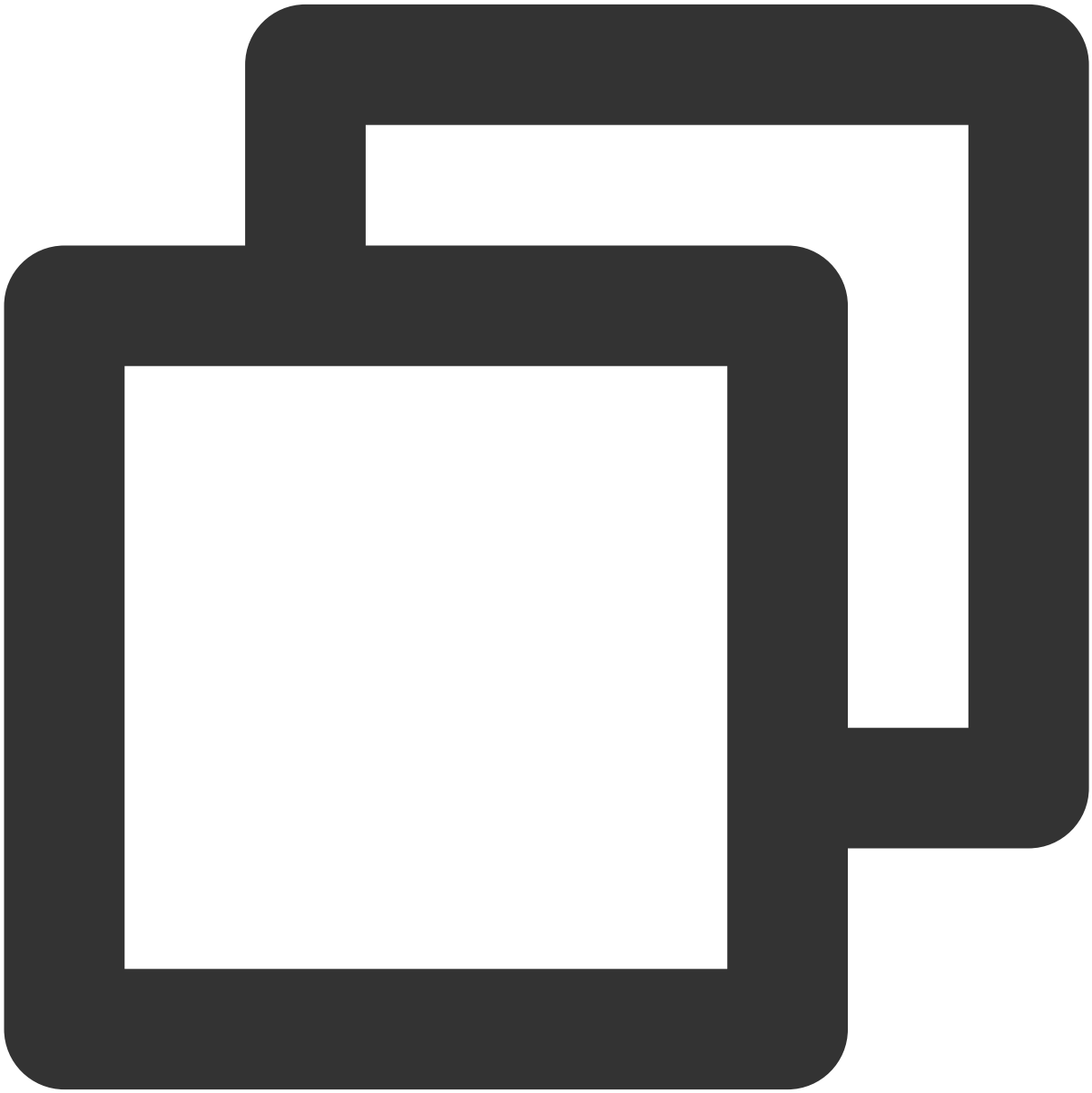
**Notice：**

In this case, the custom RoomId is invalid. The SDK will invite others to join the room where the inviter is currently located.

### joinInGroupCall

This API is used to join a group call.

This API is called by a group member to join the group's call.



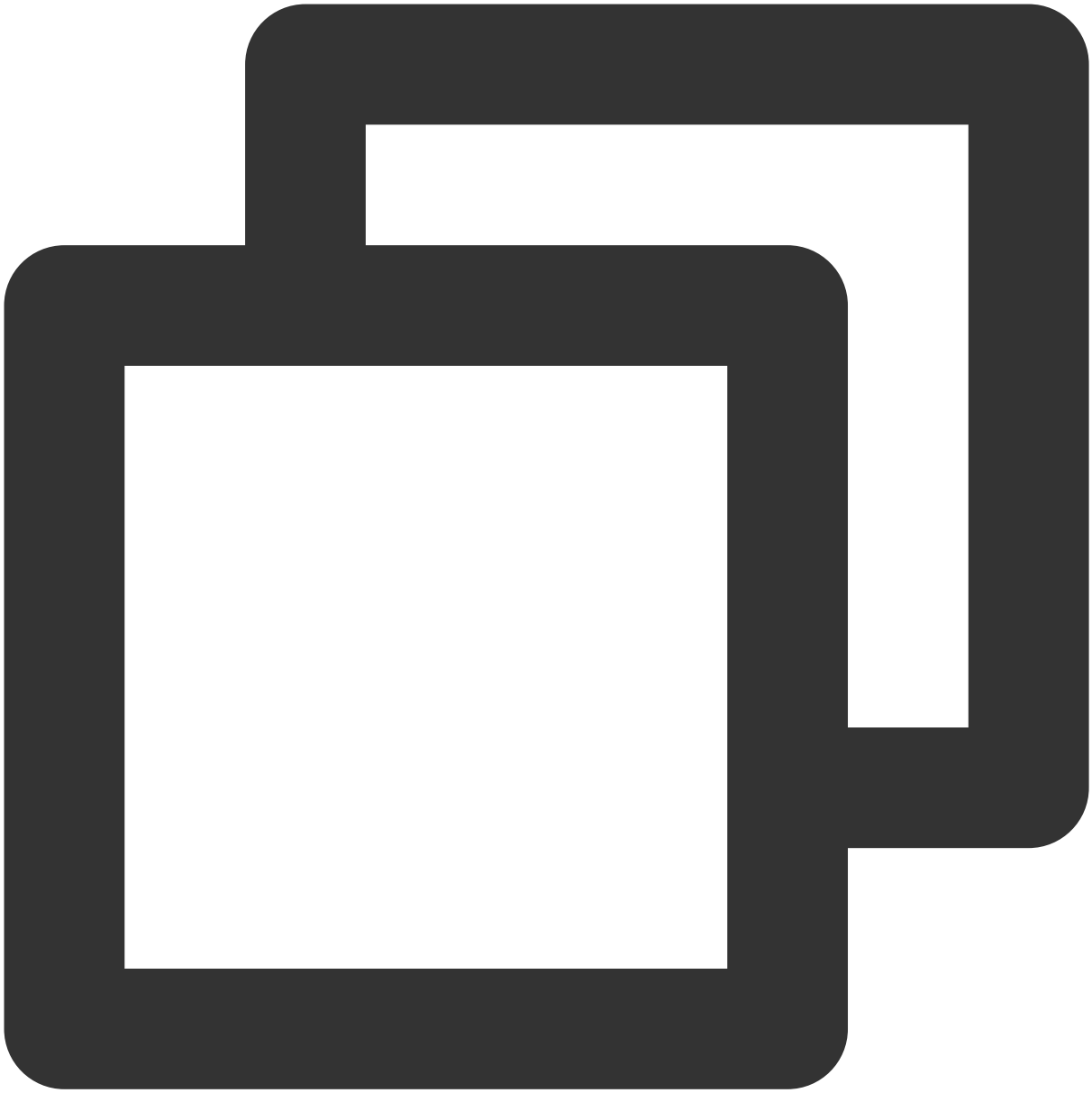
```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId,
                     TUICallDefine.MediaType callMediaType, TUICommonDefine.Callbac
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUICommonDefine.RoomId | The room ID. |
| groupId | String | The group ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

### switchCallMediaType

This API is used to change the call type.



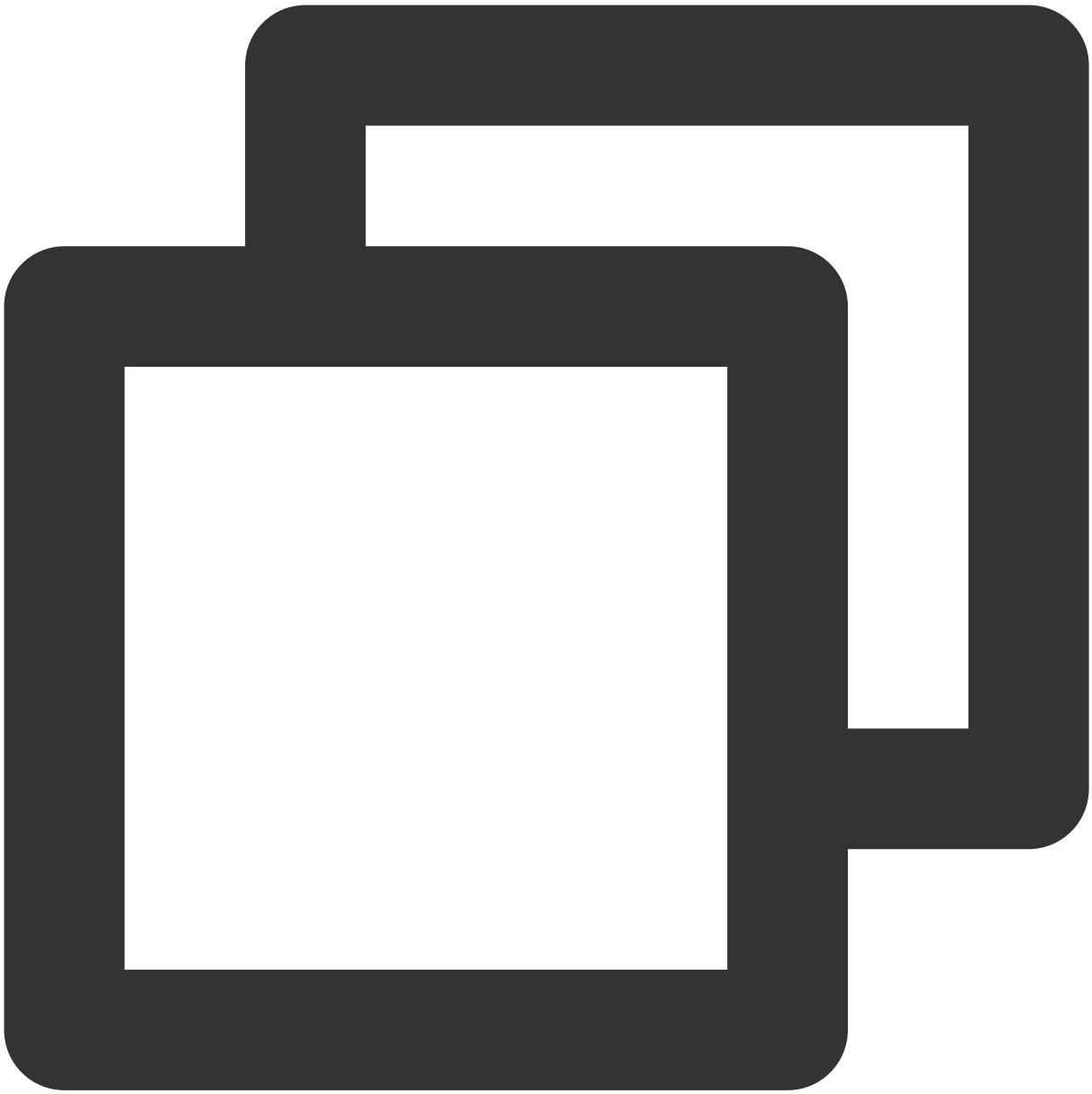
```
void switchCallMediaType(TUICallDefine.MediaType callMediaType);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

### startRemoteView

This API is used to subscribe to the video stream of a remote user. For it to work, make sure you call it after `setRenderView` .



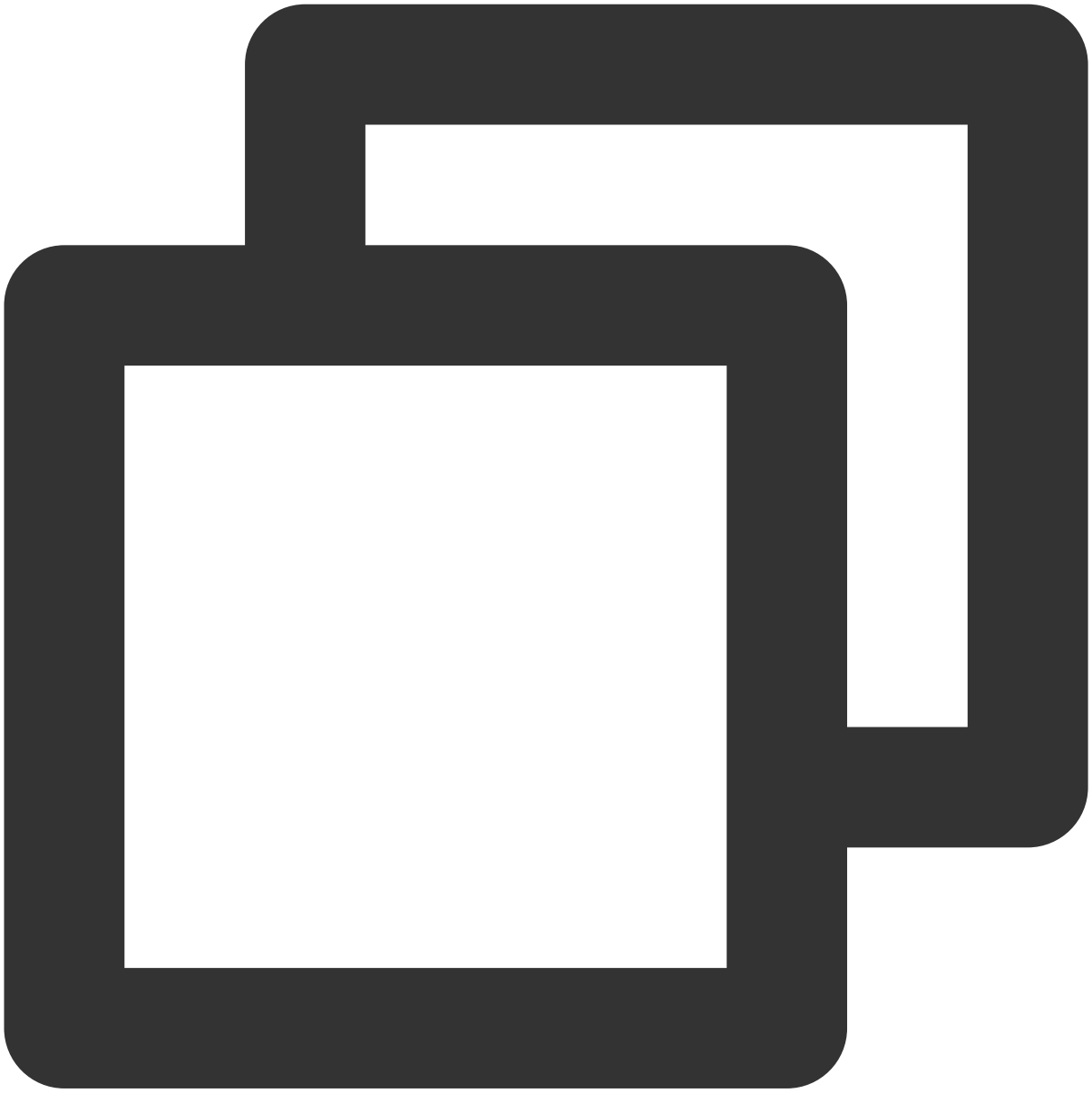
```
void startRemoteView(String userId, TUIVideoView videoView, TUICommonDefine.PlayCal
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| videoView | TUIVideoView | The view to be rendered. |

### stopRemoteView

This API is used to unsubscribe from the video stream of a remote user.



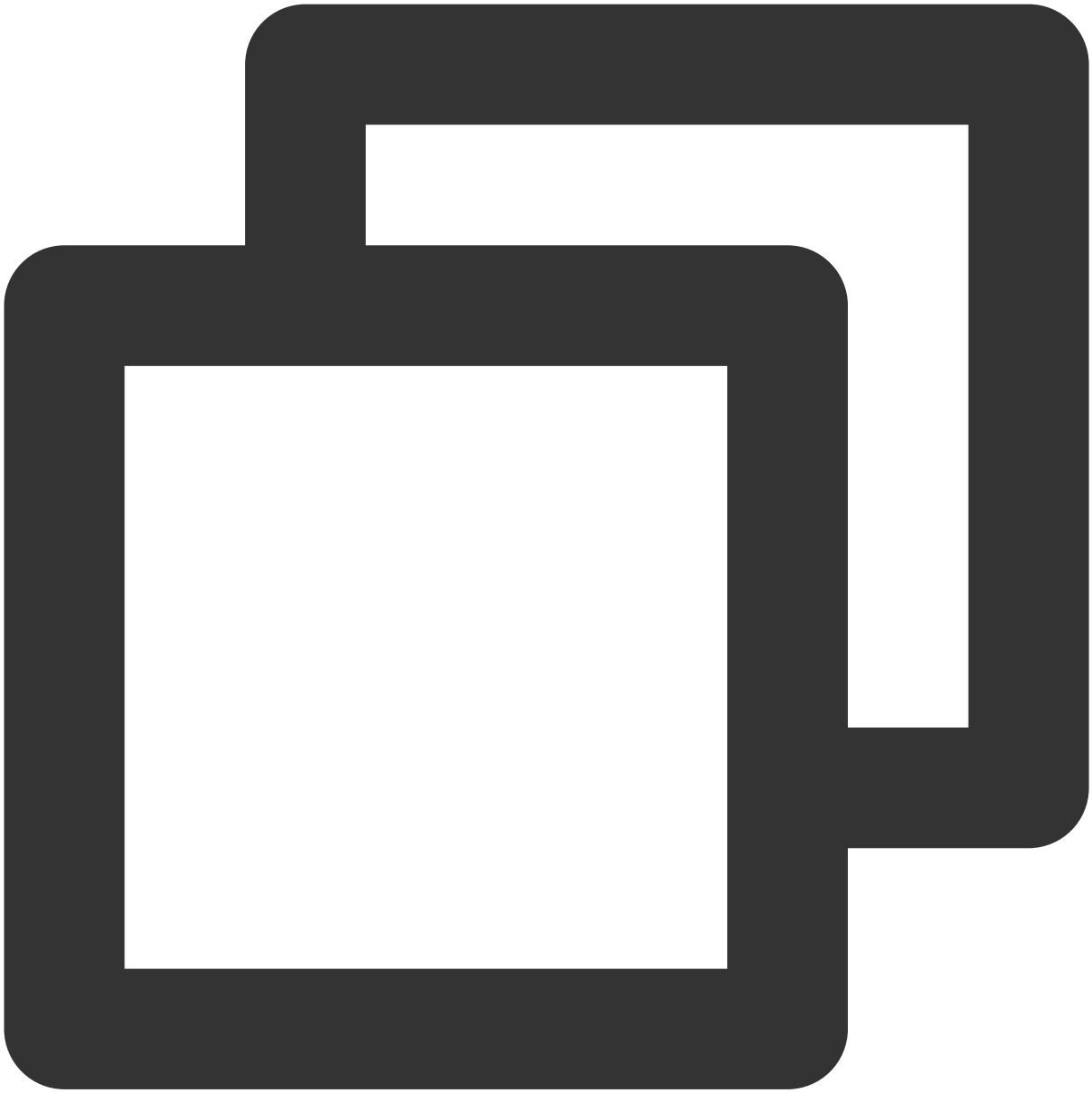
```
void stopRemoteView(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |

### openCamera

This API is used to turn the camera on.



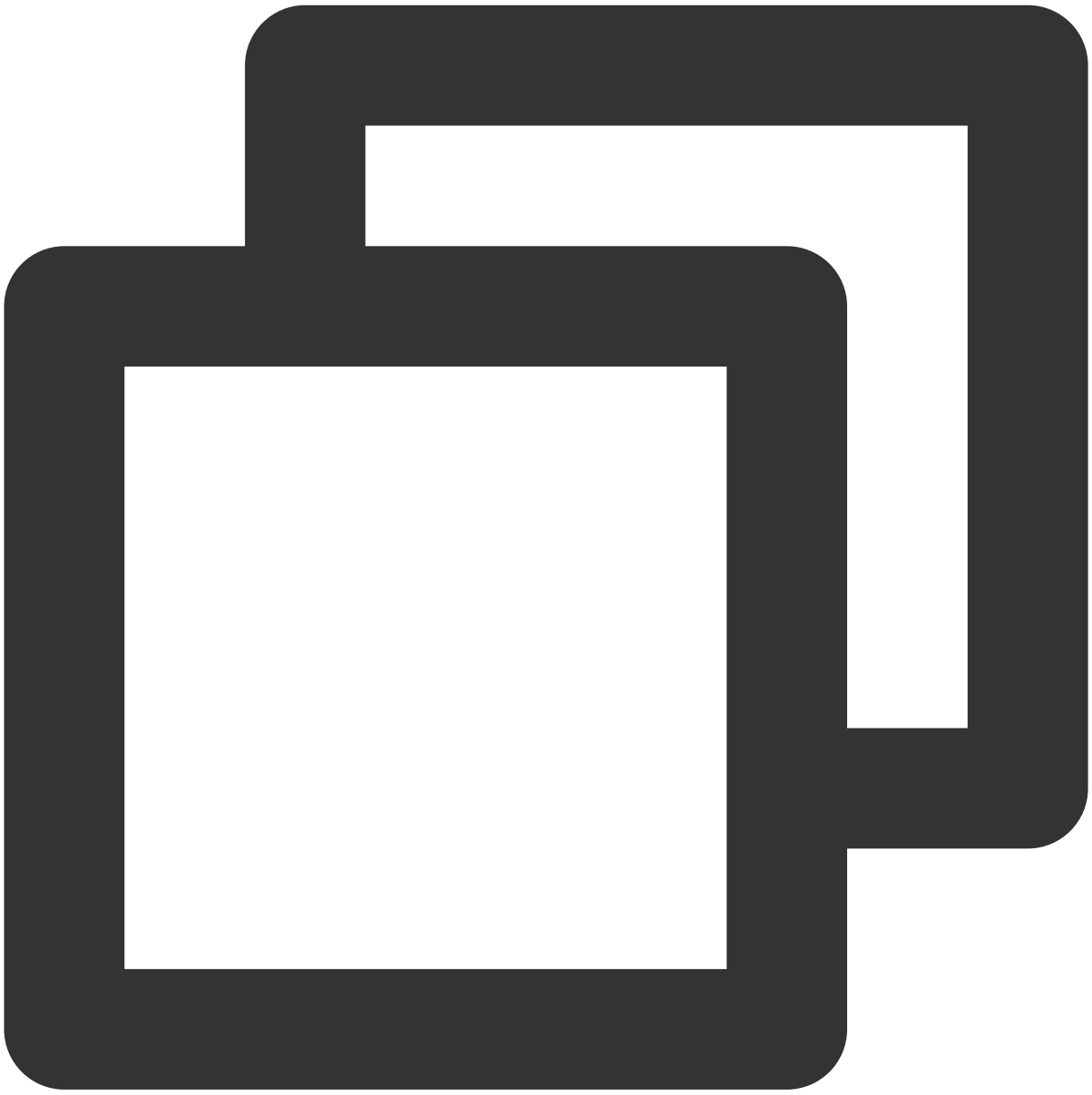
```
void openCamera(TUICommonDefine.Camera camera, TUIVideoView videoView, TUICommonDef
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| camera | TUICommonDefine.Camera | The front or rear camera. |
| videoView | TUIVideoView | The view to be rendered. |

### closeCamera

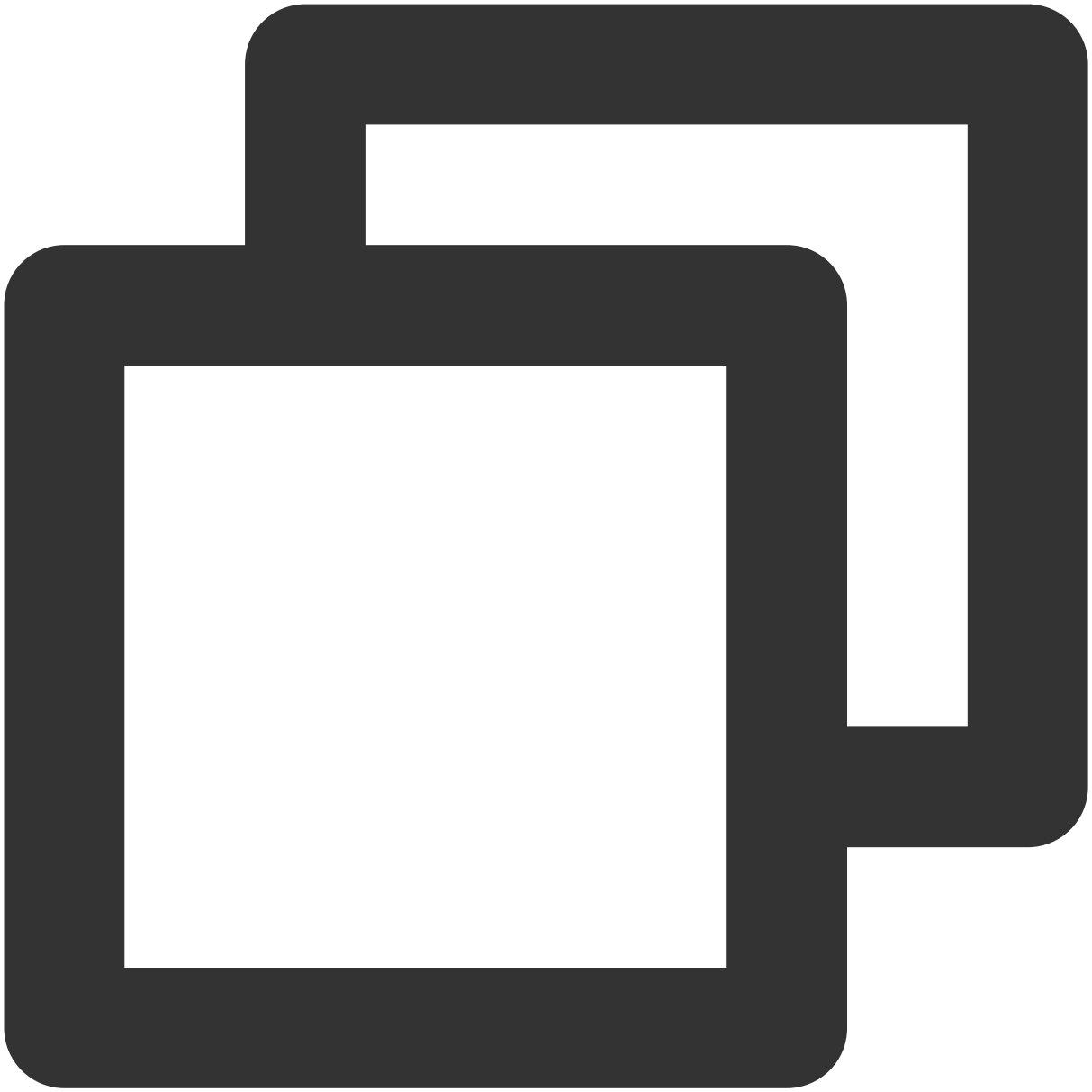
This API is used to turn the camera off.



```
void closeCamera();
```

### switchCamera

This API is used to switch between the front and rear cameras.



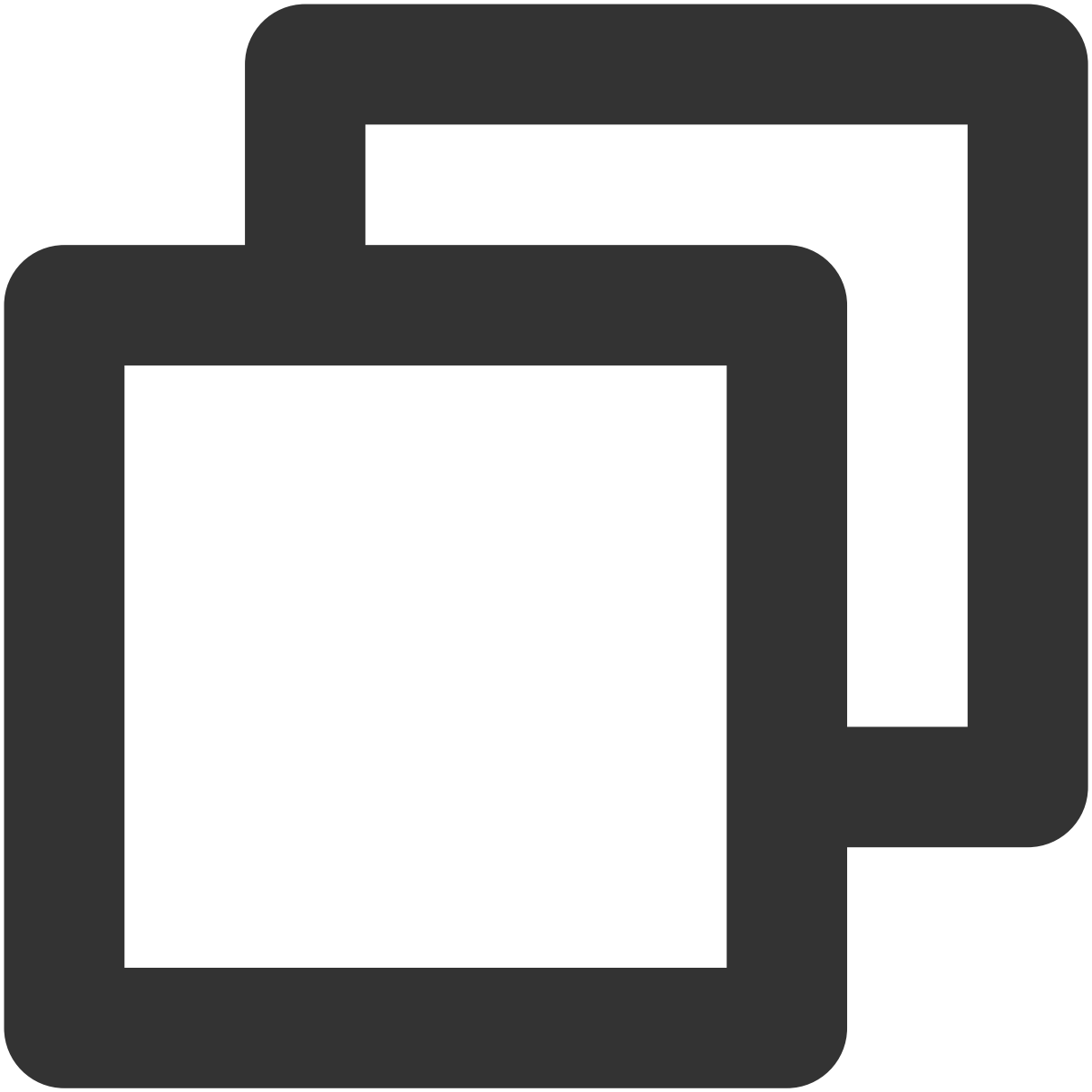
```
void switchCamera(TUICommonDefine.Camera camera);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| camera | TUICommonDefine.Camera | The front or rear camera. |

### openMicrophone

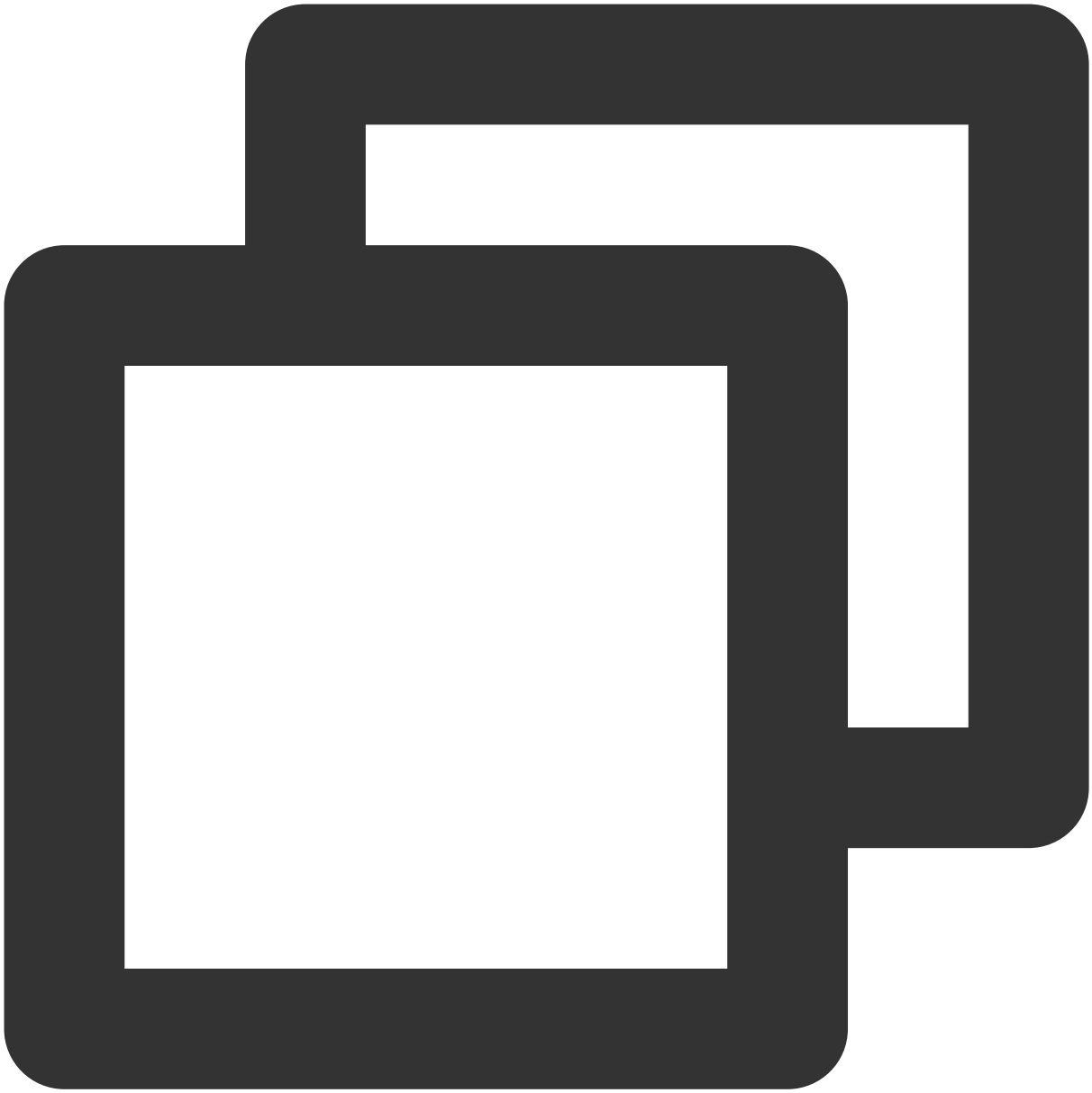
This API is used to turn the mic on.



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

### closeMicrophone

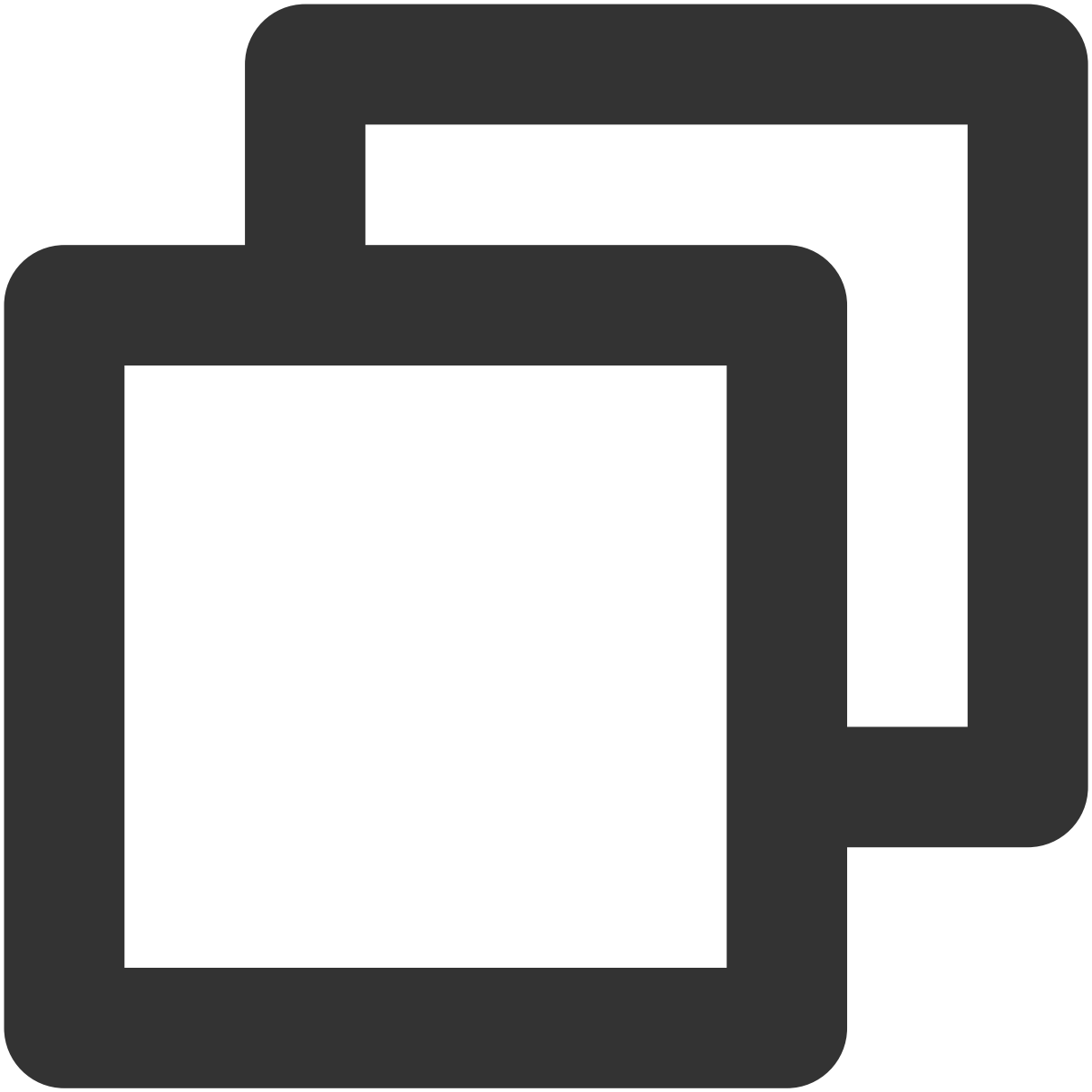
This API is used to turn the mic off.



```
void closeMicrophone();
```

### selectAudioPlaybackDevice

This API is used to select the audio playback device (receiver or speaker). In call scenarios, you can use this API to turn on/off hands-free mode.



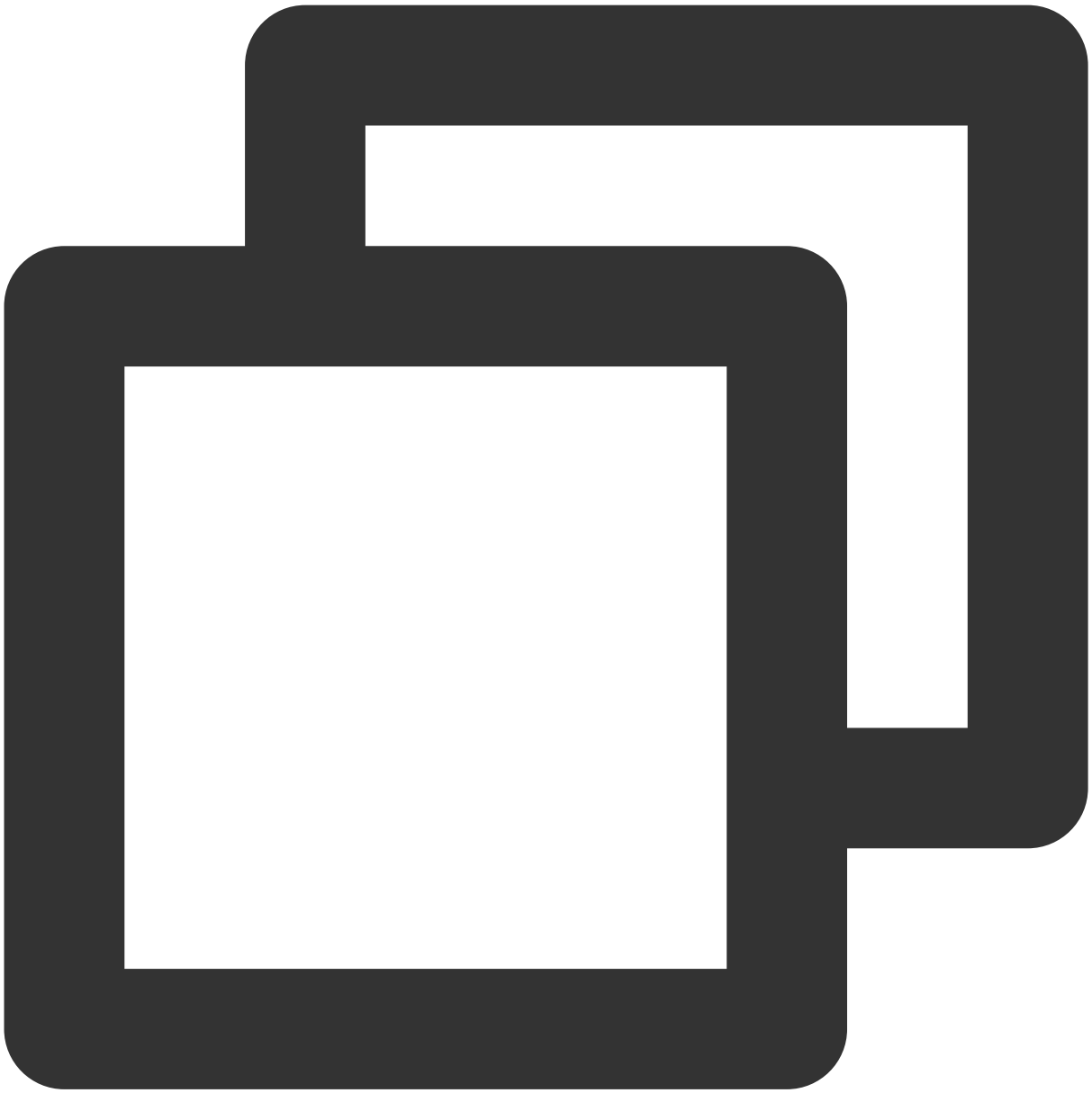
```
void selectAudioPlaybackDevice(TUICommonDefine.AudioPlaybackDevice device);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| device | TUICommonDefine.AudioPlaybackDevice | The speaker or receiver. |

### setSelfInfo

This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



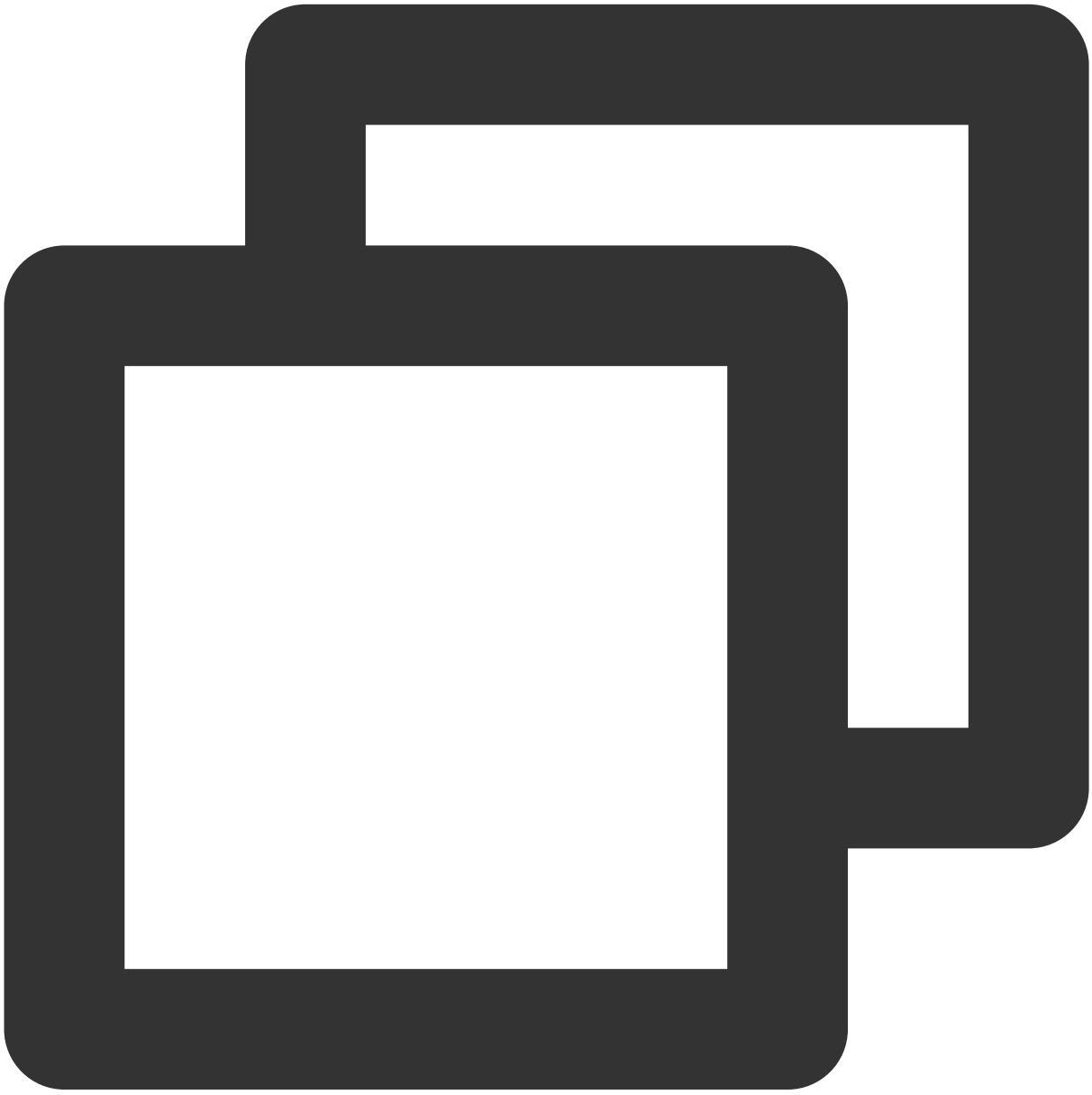
```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| nickname | String | The alias. |
| avatar | String | The URL of the profile photo. |

### enableMultiDeviceAbility

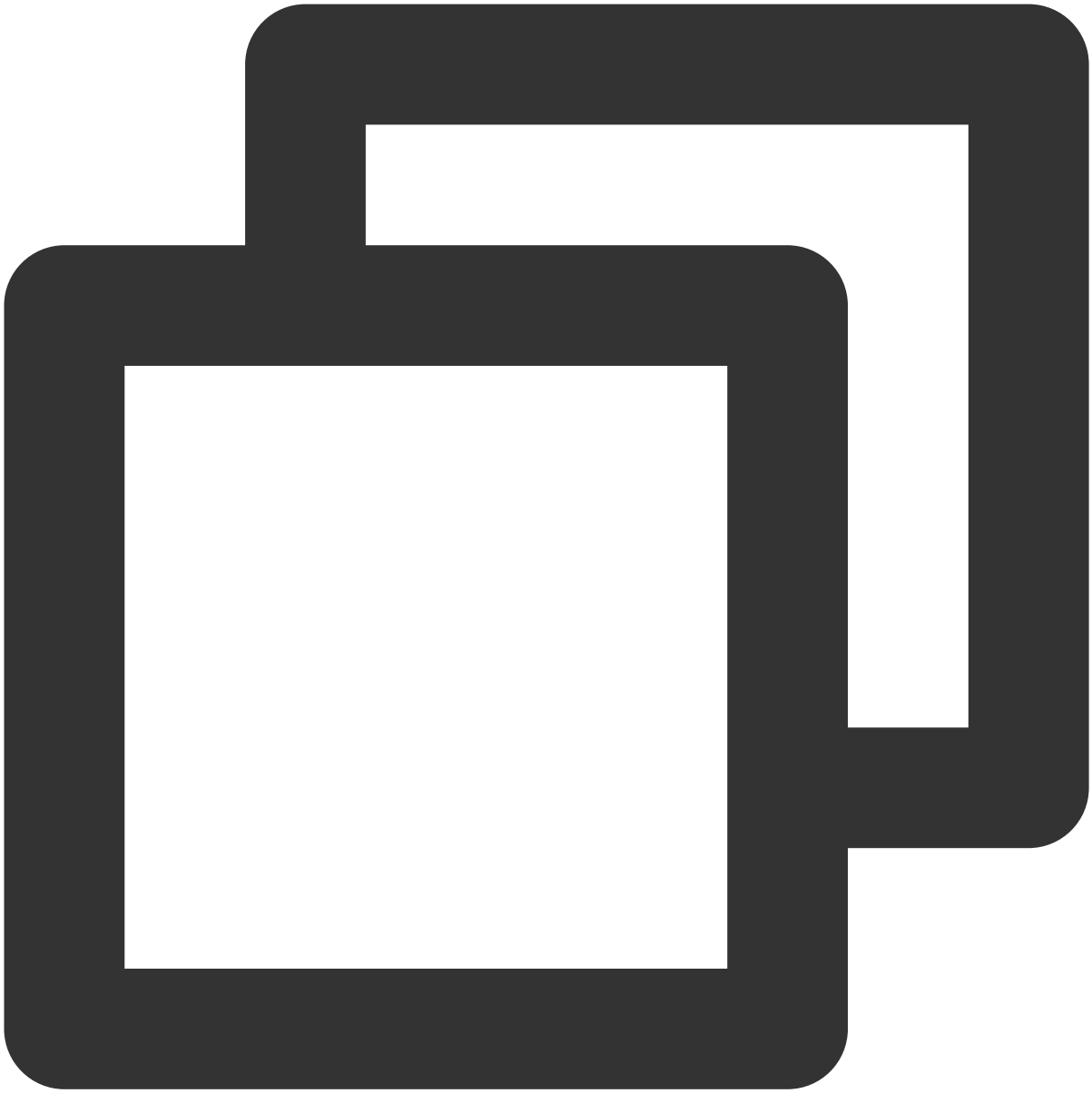
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the premium package).



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

### setVideoRenderParams

Set the rendering mode of video image.



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

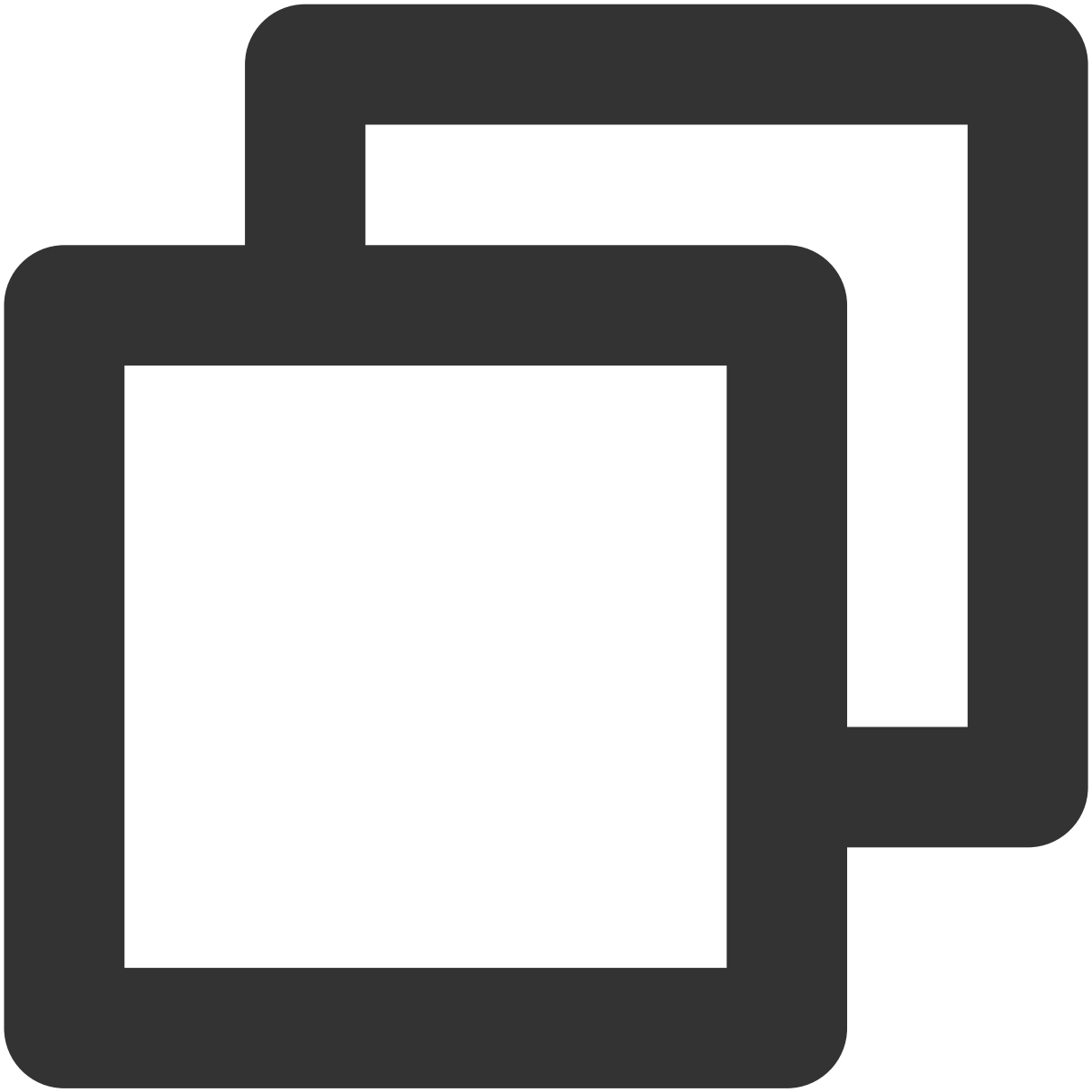
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| params | TUICommonDefine.VideoRenderParams | Video render parameters. |

### setVideoEncoderParams

Set the encoding parameters of video encoder.

This setting can determine the quality of image viewed by remote users, which is also the image quality of on-cloud recording files.



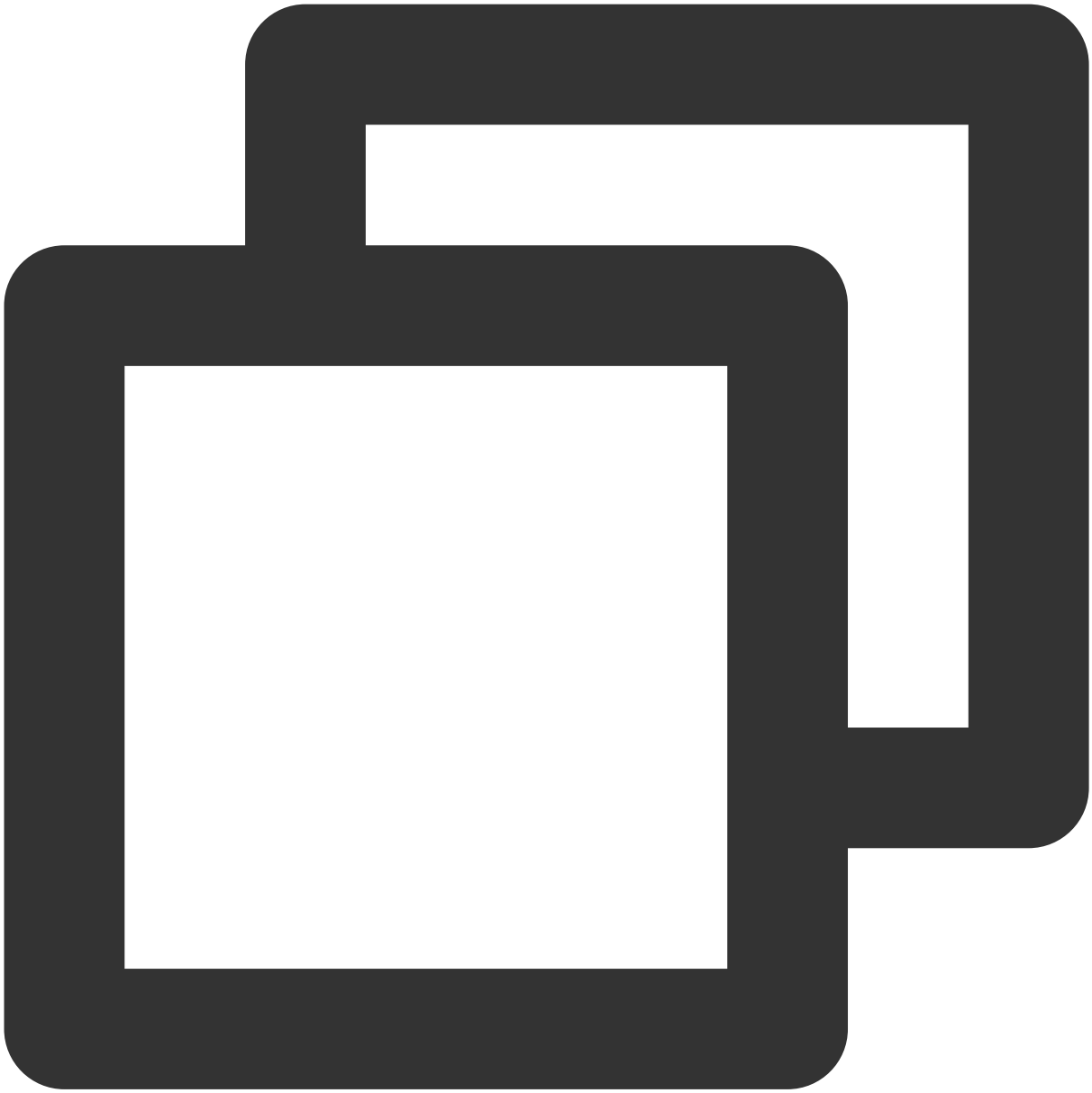
```
void setVideoEncoderParams(TUICommonDefine.VideoEncoderParams params, TUICommonDefi
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| params | TUICommonDefine.VideoEncoderParams | Video encoding parameters |

### getTRTCCloudInstance

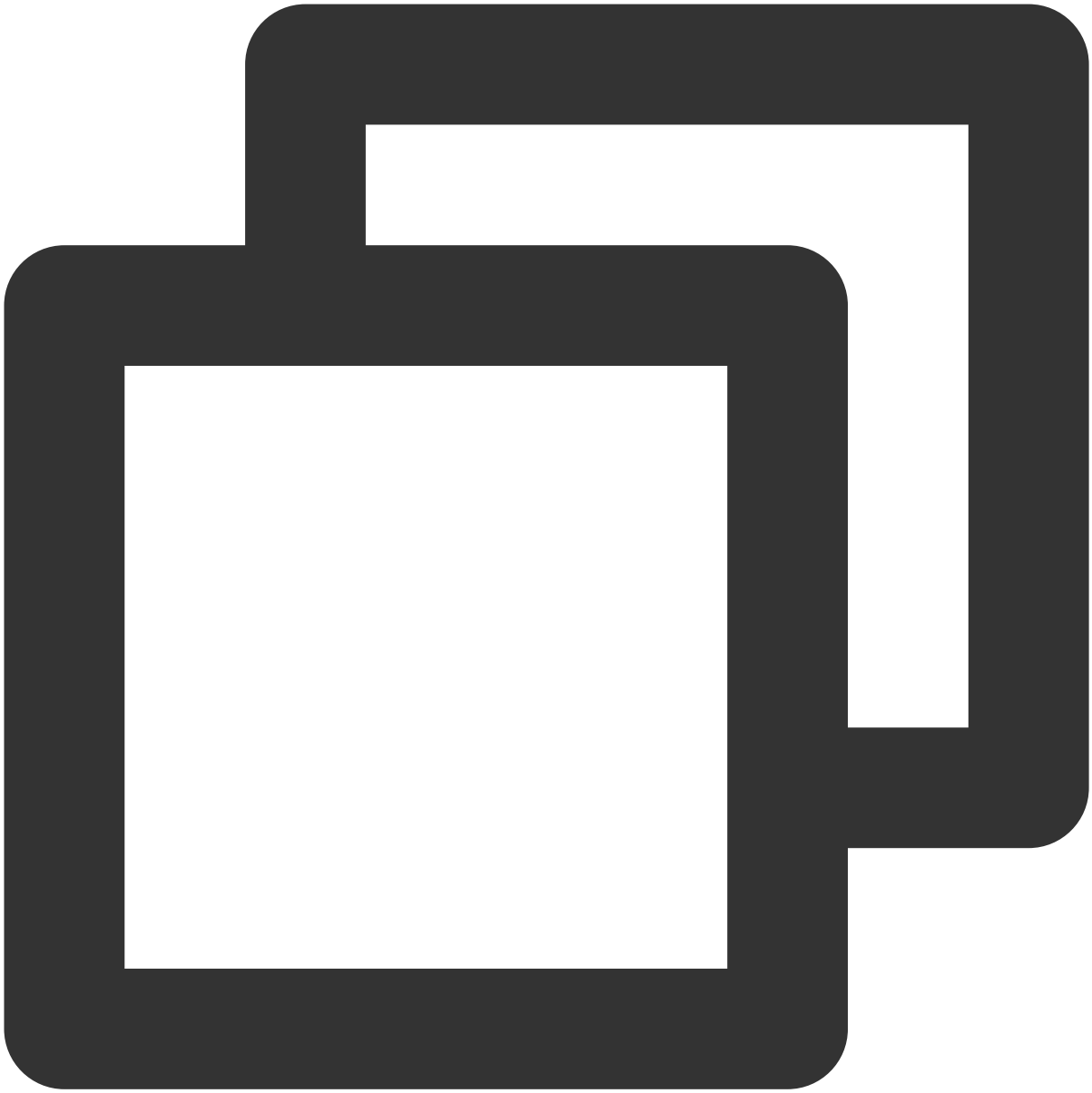
Advanced features.



```
TRTCCloud getTRTCCloudInstance();
```

### setBeautyLevel

Set beauty level, support turning off default beauty.



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

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| level | float | Beauty level，range: 0 - 9；<br>0 means turning off the effect, 9 means the most obvious effect. |

# TUICallObserver

Last updated：2024-01-25 14:24:04

## TUICallObserver APIs

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

## Overview

| API | Description |
| --- | --- |
| onError | A call occurred during the call. |
| onCallReceived | A call invitation was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user had a video stream. |
| onUserAudioAvailable | Whether a user had an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | The current user was kicked offline. |
| | |

| | |
|---|---|
| onUserSigExpired | The user sig is expired. |

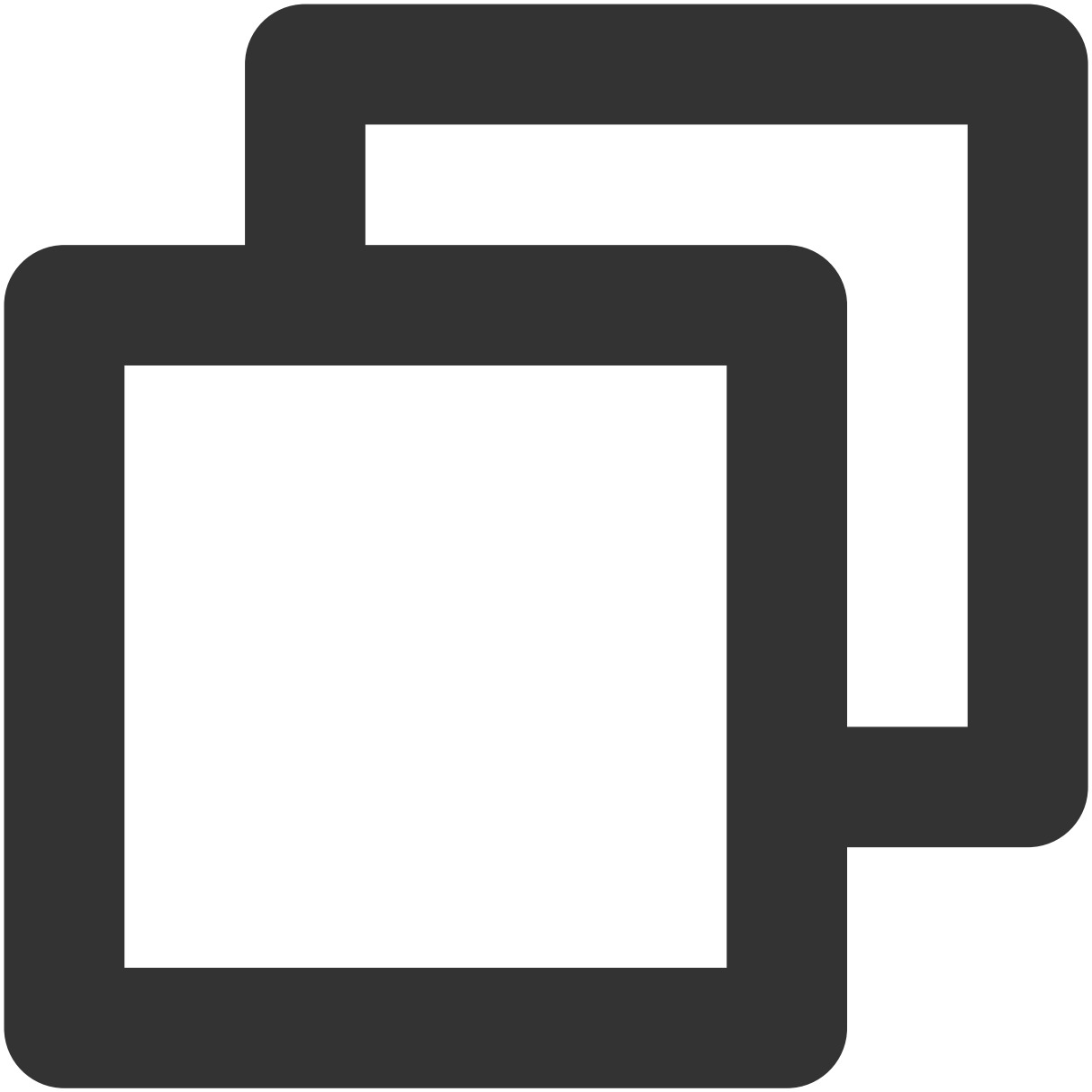
# Details

### onError

An error occurred.

**explain**

This callback indicates that the SDK encountered an unrecoverable error. Such errors must be listened for, and UI reminders should be sent to users if necessary.



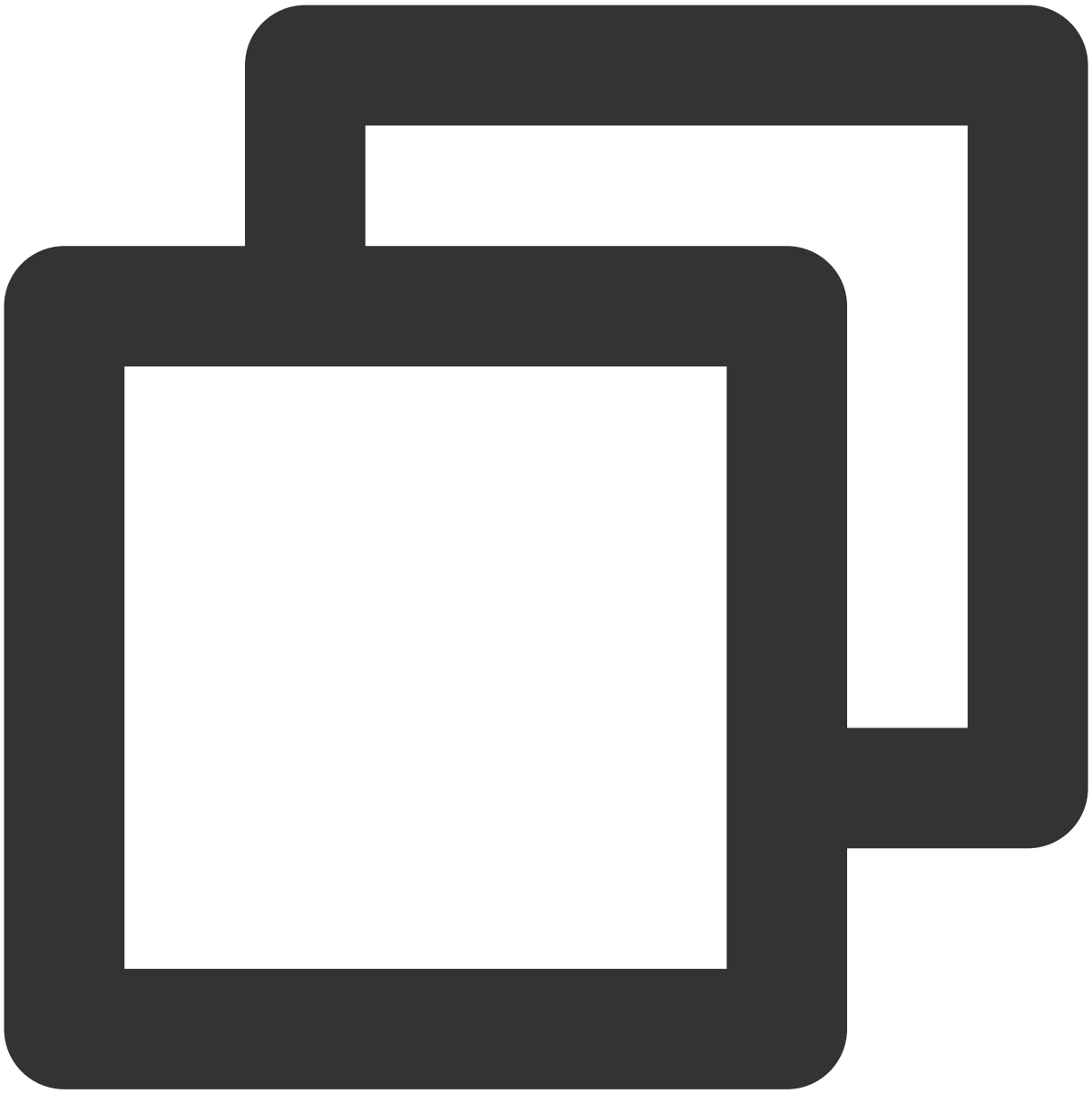
```
void onError(int code, String msg);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| code | int | The error code. |
| msg | String | The error message. |

### onCallReceived

A call invitation was received. This callback is received by an invitee. You can listen for this event to determine whether to display the incoming call view.



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

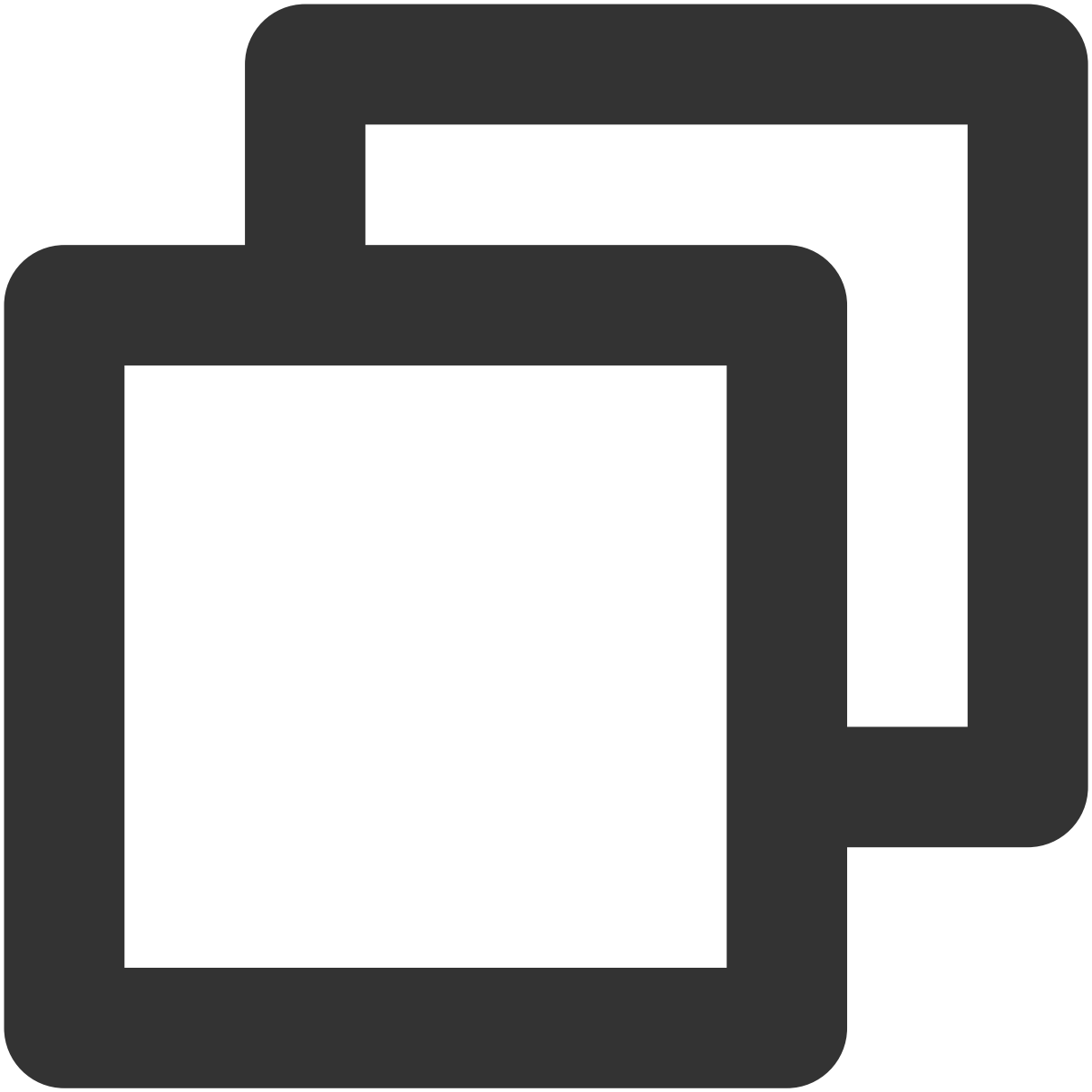
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| callerId | String | The user ID of the inviter. |
| | | |

| | | |
|---|---|---|
| calleeIdList | List | The invitee list. |
| groupId | String | The group ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |
| userData | String | User-added extended fields., Please refer to: TUICallDefine.CallParams |

### onCallCancelled

The call was canceled by the inviter or timed out. This callback is received by an invitee. You can listen for this event to determine whether to show a missed call message.
This indicates that the call was canceled by the caller, timed out by the callee, rejected by the callee, or the callee was busy. There are multiple scenarios involved. You can listen to this event to achieve UI logic such as missed calls and resetting UI status.
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller).
Callee timeout: the caller will simultaneously receive the onUserNoResponse and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID).
Callee rejection: The caller will simultaneously receive the onUserReject and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID),
Callee busy: The caller will simultaneously receive the onUserLineBusy and onCallCancelled callbacks (userId is his own ID),
Abnormal interruption: The callee failed to receive the call , he receives this callback (userId is his own ID).



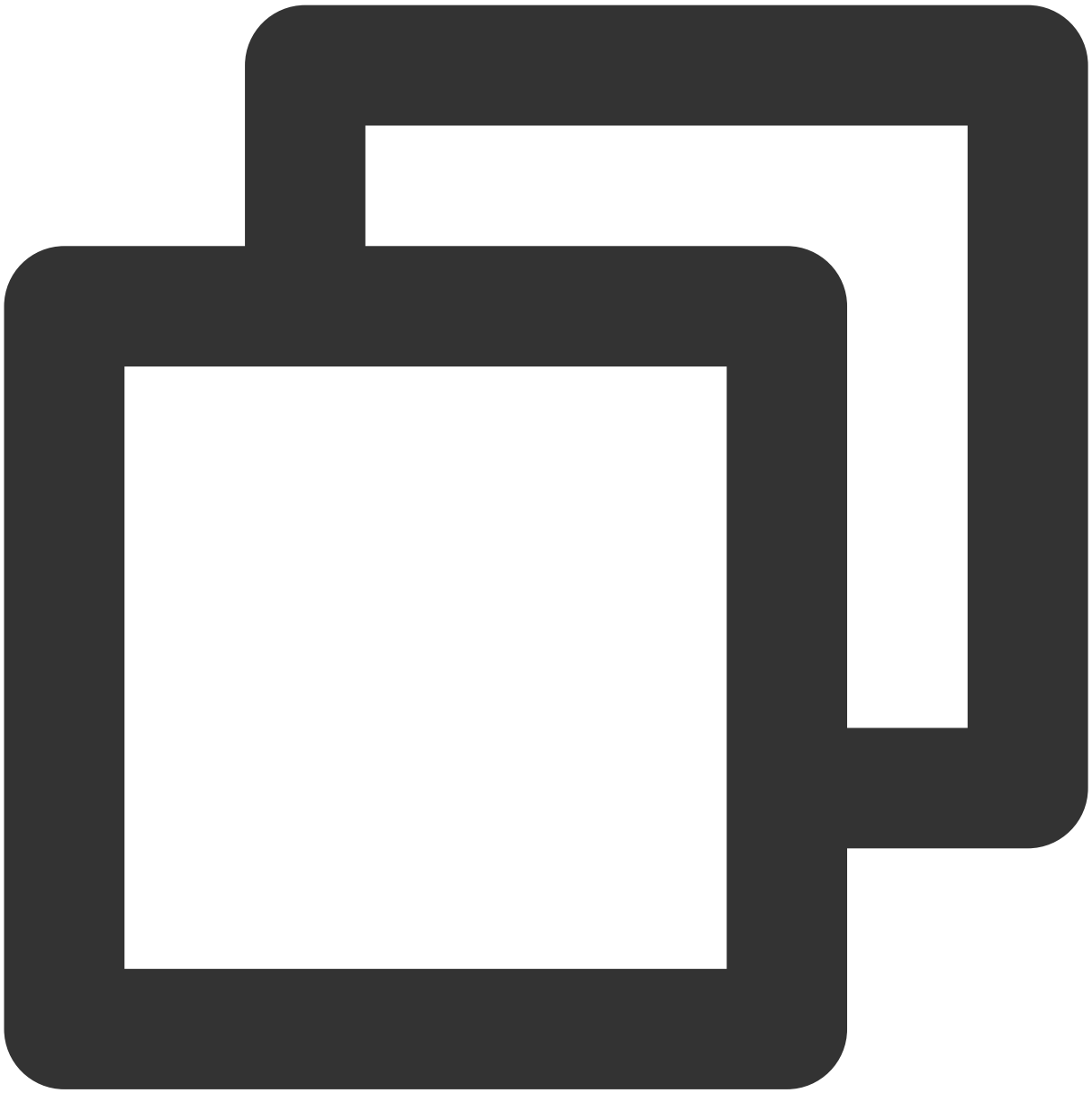
```
void onCallCancelled(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the inviter. |

### onCallBegin

The call was connected. This callback is received by both the inviter and invitees. You can listen for this event to determine whether to start on-cloud recording, content moderation, or other tasks.



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

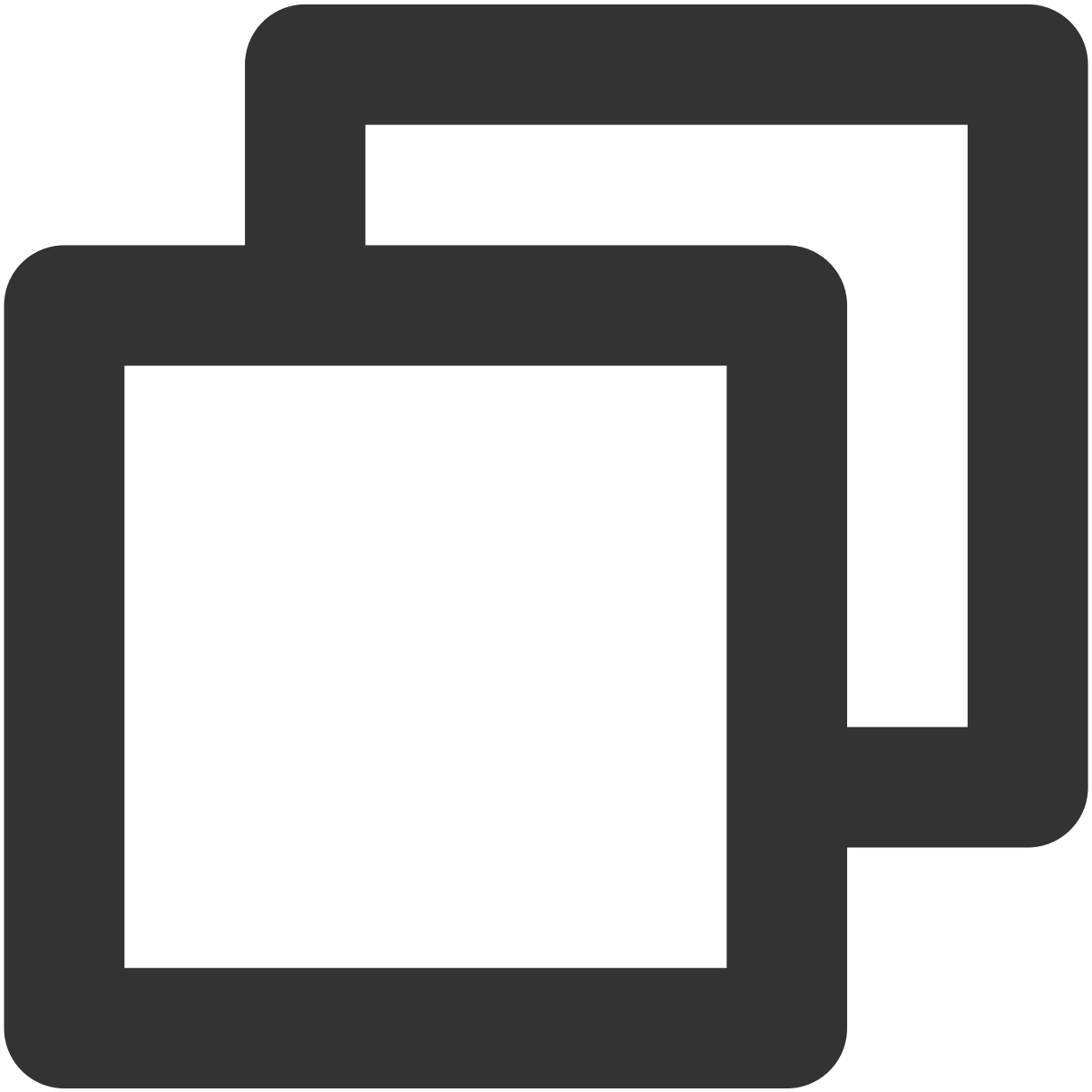
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUICommonDefine.RoomId | The room ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |

| | | |
|---|---|---|
| callRole | TUICallDefine.Role | The role, which can be caller or callee. |

### onCallEnd

The call ended. This callback is received by both the inviter and invitees. You can listen for this event to determine when to display call information such as call duration and call type, or stop on-cloud recording.



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

The parameters are described below:

| | | |
|---|---|---|

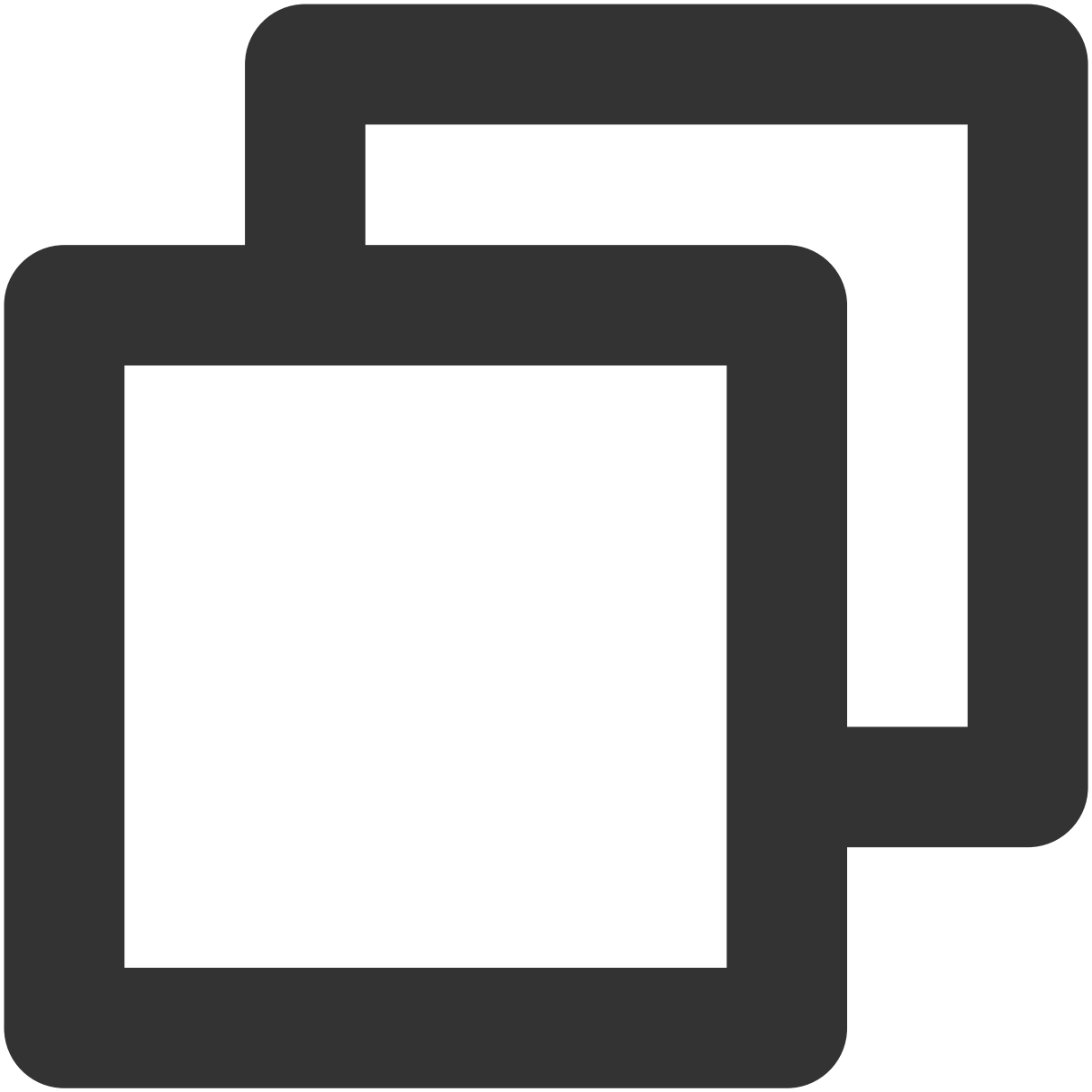
| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUICommonDefine.RoomId | The room ID. |
| callMediaType | TUICallDefine.MediaType | The call type, which can be video or audio. |
| callRole | TUICallDefine.Role | The role, which can be caller or callee. |
| totalTime | long | The call duration. |

**Notice：**

Client-side callbacks are often lost when errors occur, for example, when the process is closed. If you need to measure the duration of a call for billing or other purposes, we recommend you use the RESTful API.

### onCallMediaTypeChanged

The call type changed.



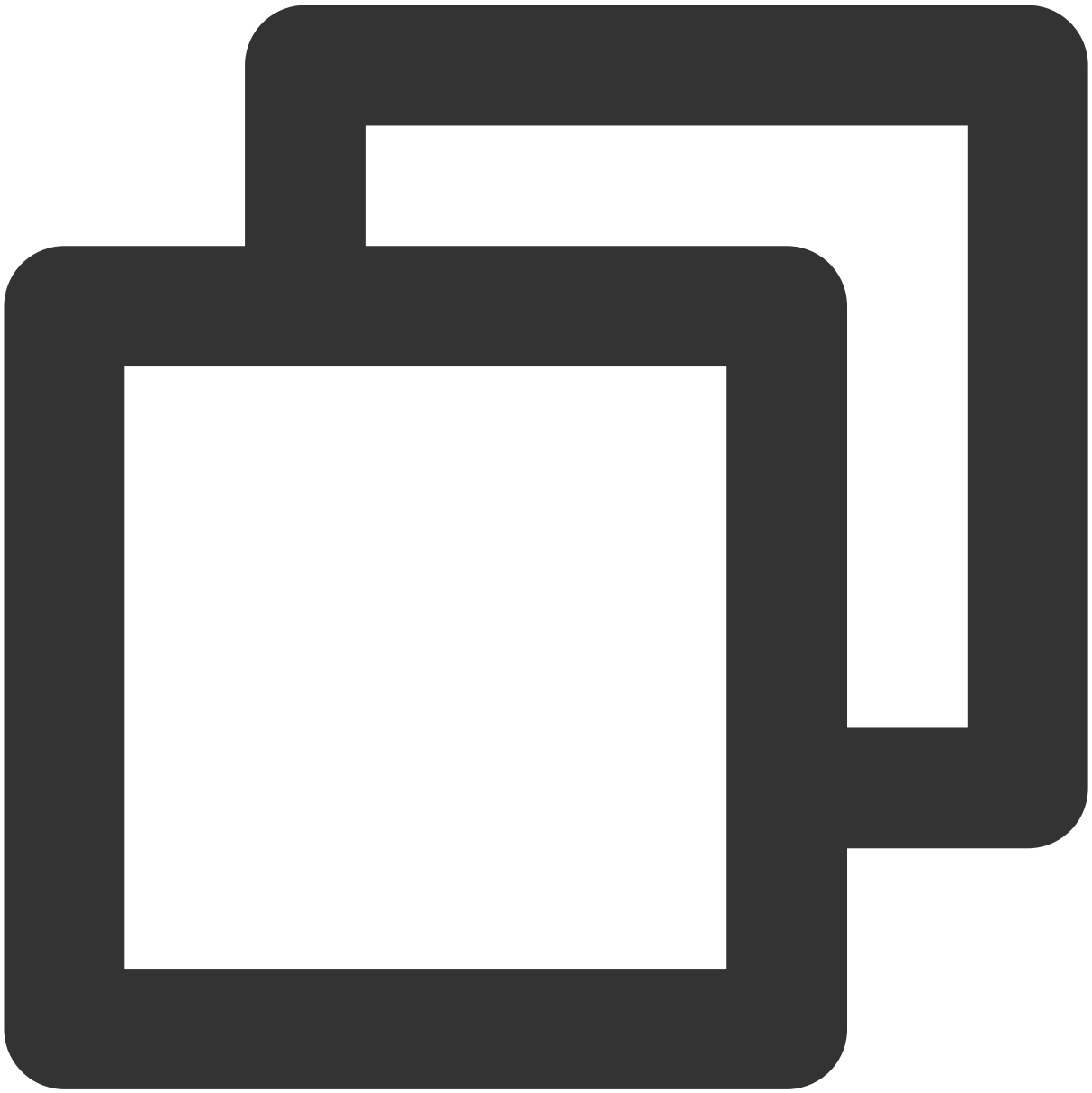
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType,TUICallDefine.
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| oldCallMediaType | TUICallDefine.MediaType | The call type before the change. |
| newCallMediaType | TUICallDefine.MediaType | The call type after the change. |

### onUserReject

The call was rejected. In a one-to-one call, only the inviter will receive this callback. In a group call, all invitees will receive this callback.



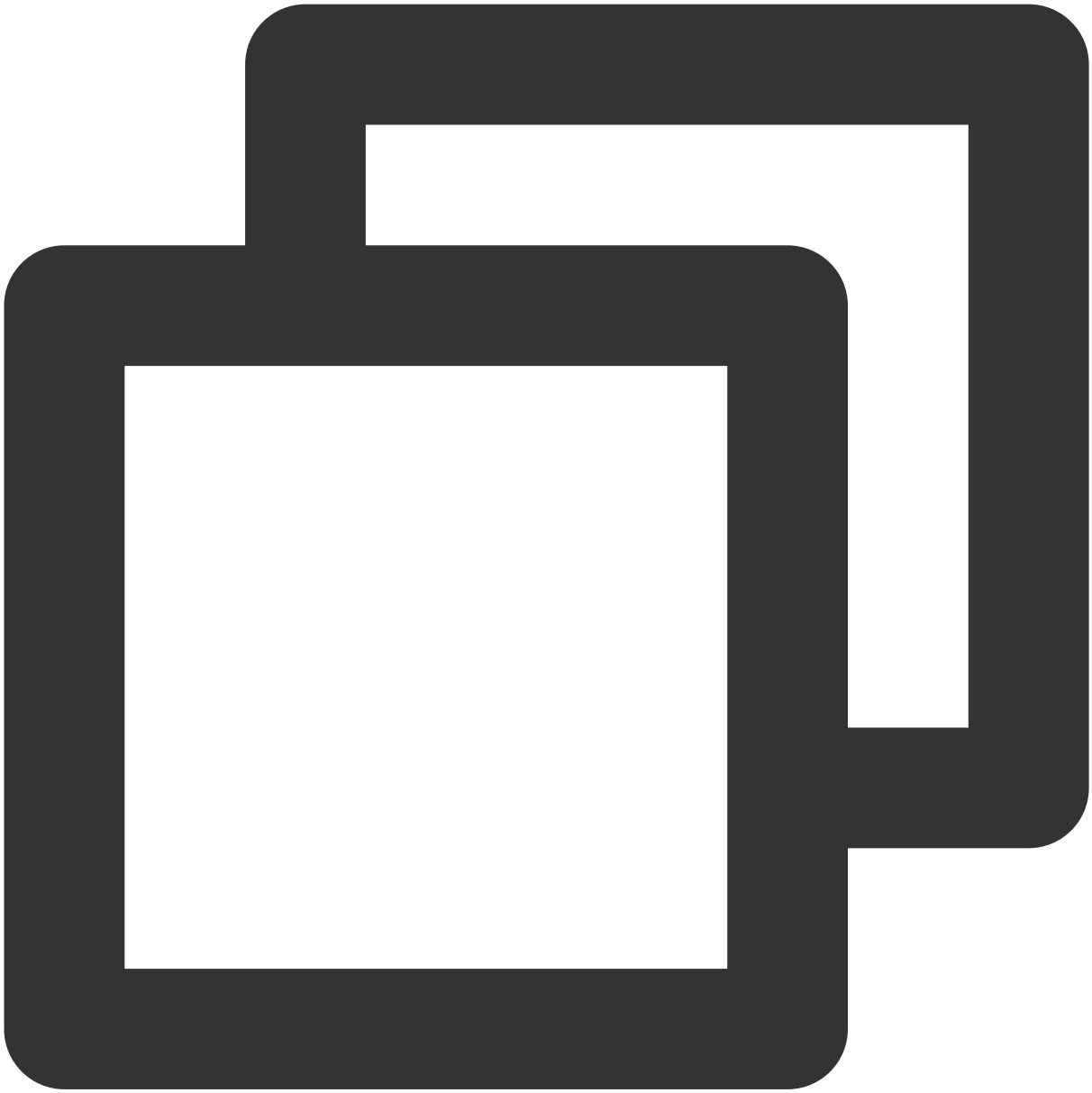
```
void onUserReject(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the invitee who rejected the call. |

### onUserNoResponse

A user did not respond.



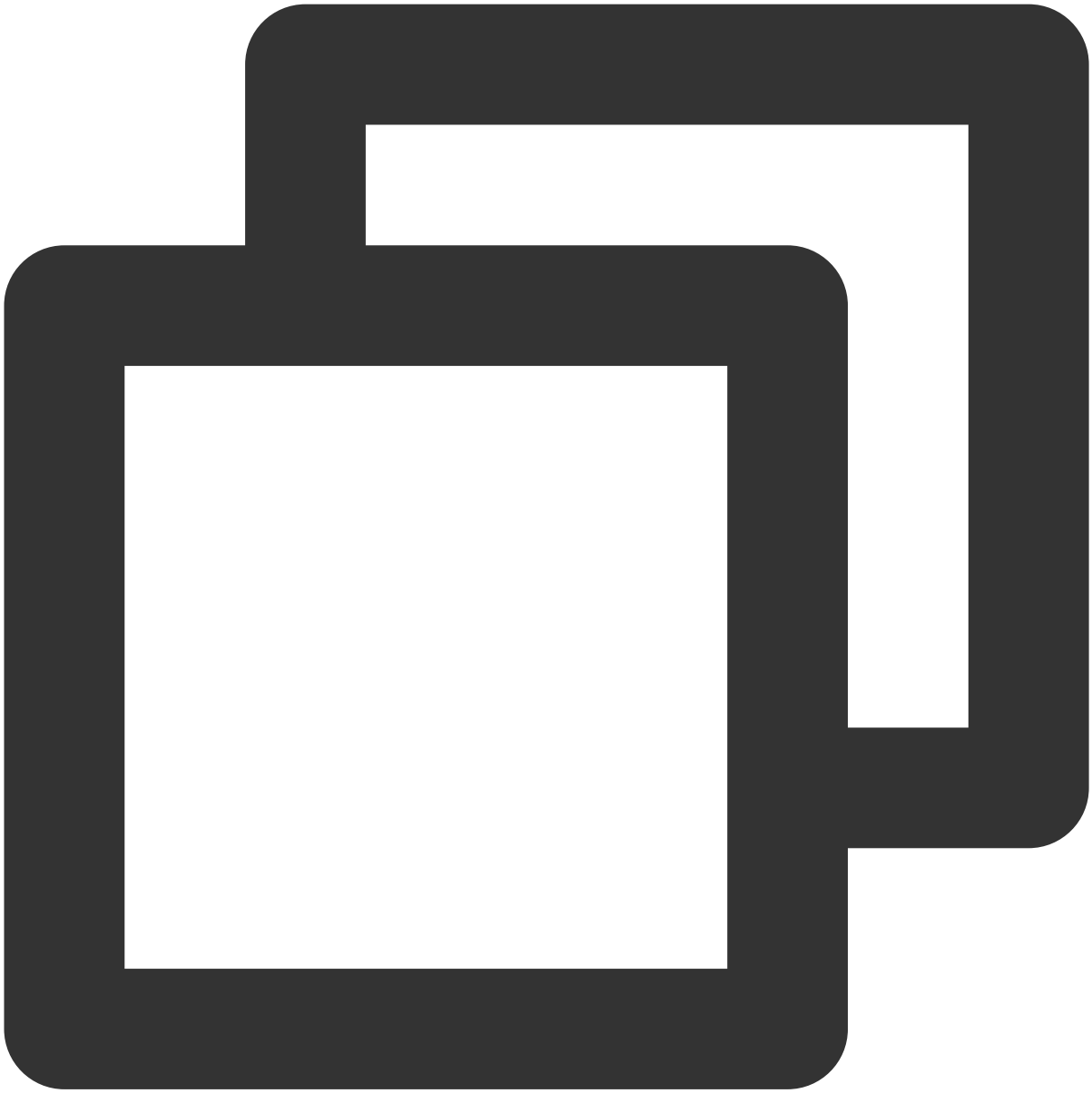
```
void onUserNoResponse(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the invitee who did not answer. |

### onUserLineBusy

A user is busy.



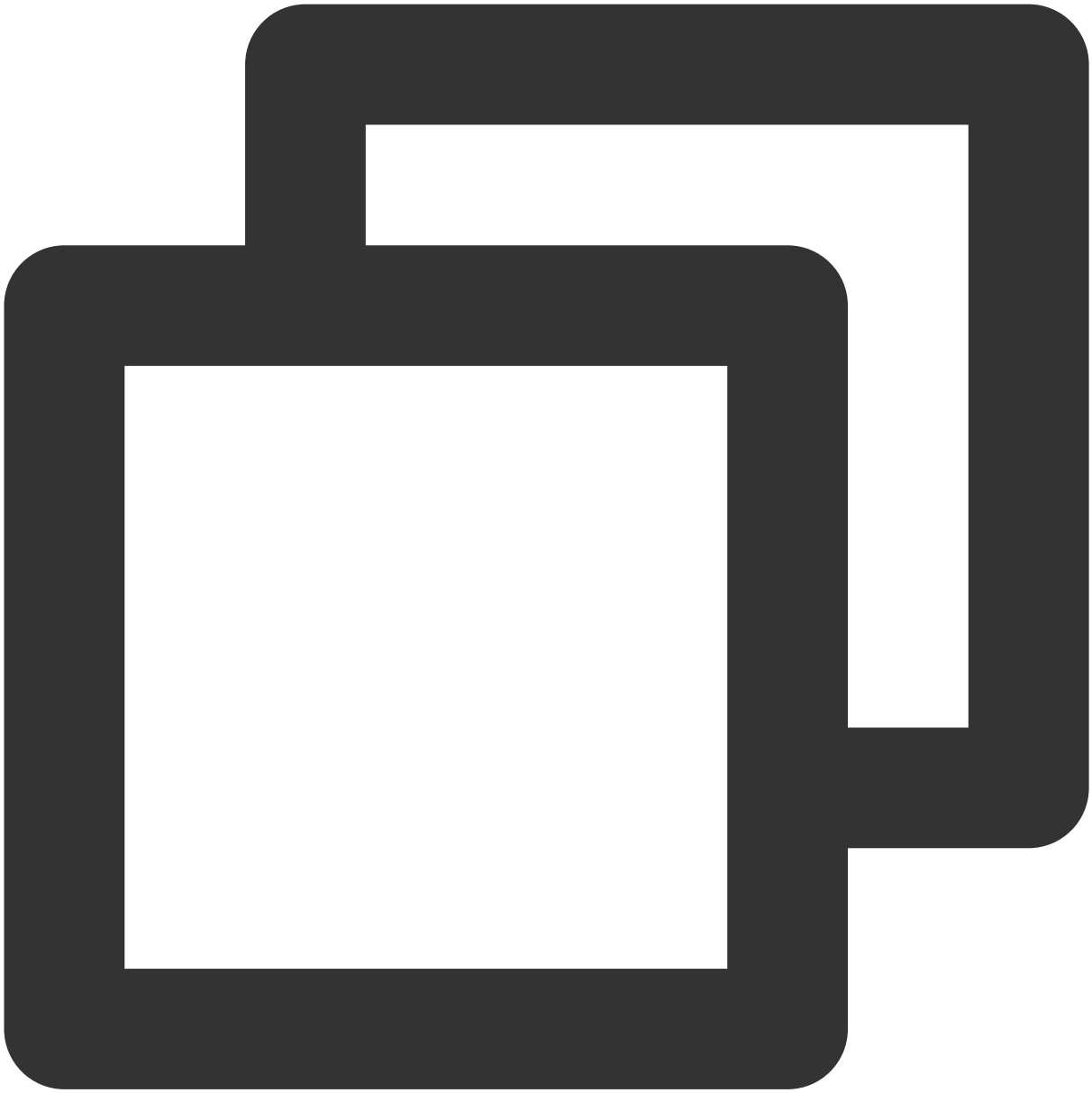
```
void onUserLineBusy(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the invitee who is busy. |

### onUserJoin

A user joined the call.



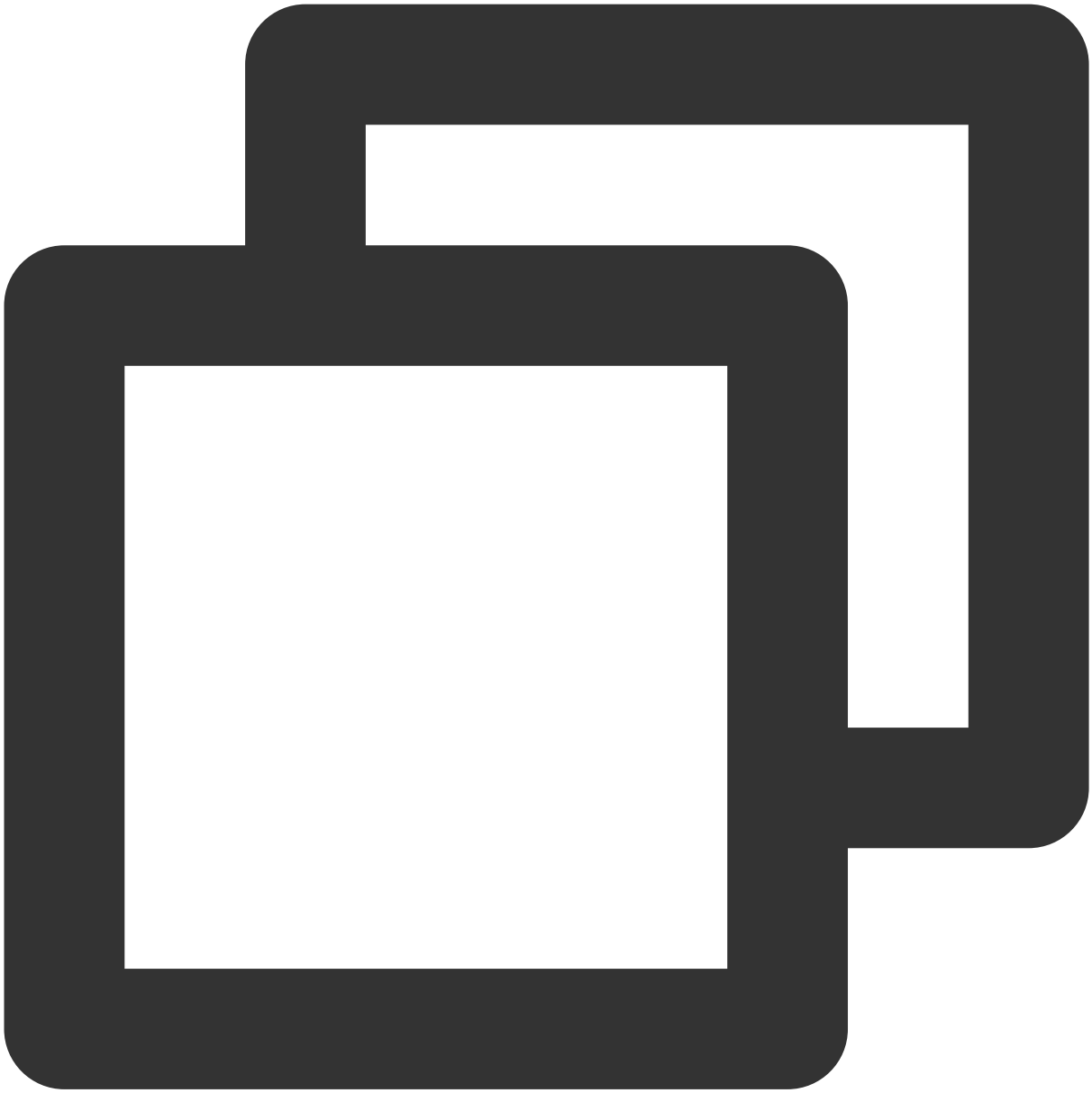
```
void onUserJoin(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the user who joined the call. |

### onUserLeave

A user left the call.



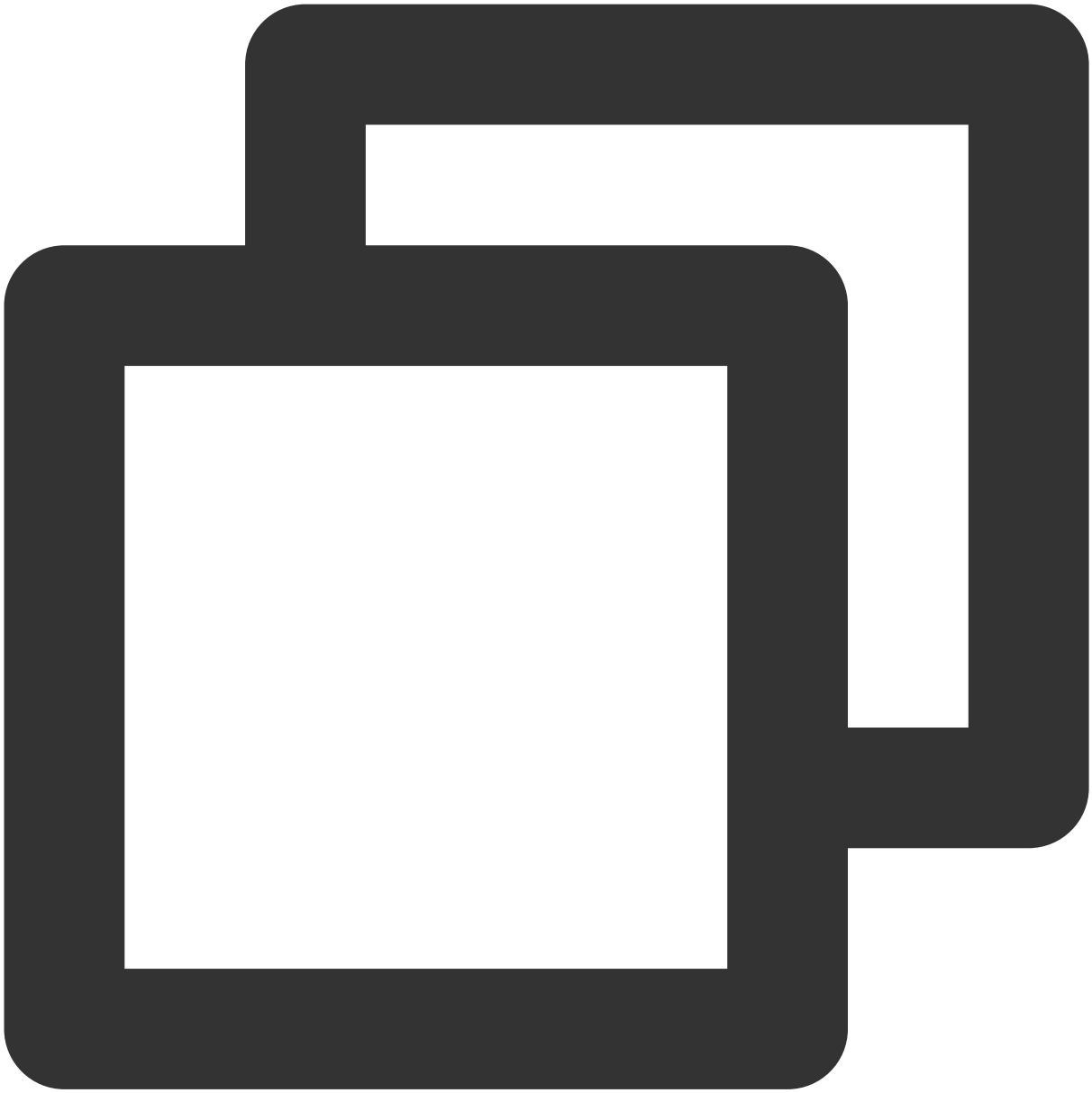
```
void onUserLeave(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the user who left the call. |

### onUserVideoAvailable

Whether a user is sending video.



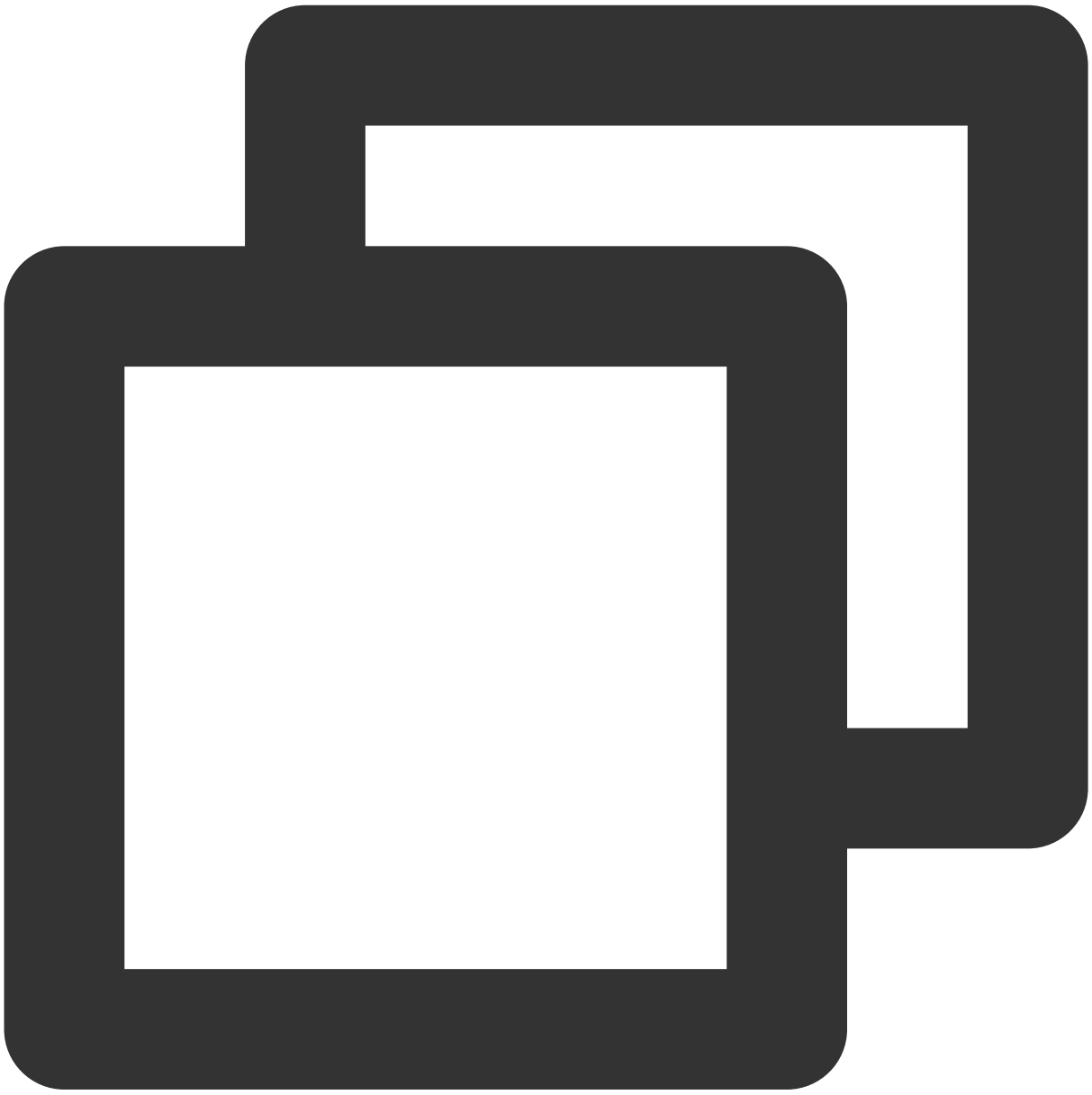
```
void onUserVideoAvailable(String userId, boolean isVideoAvailable);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID. |
| isVideoAvailable | boolean | Whether the user has video. |

### onUserAudioAvailable

Whether a user is sending audio.



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

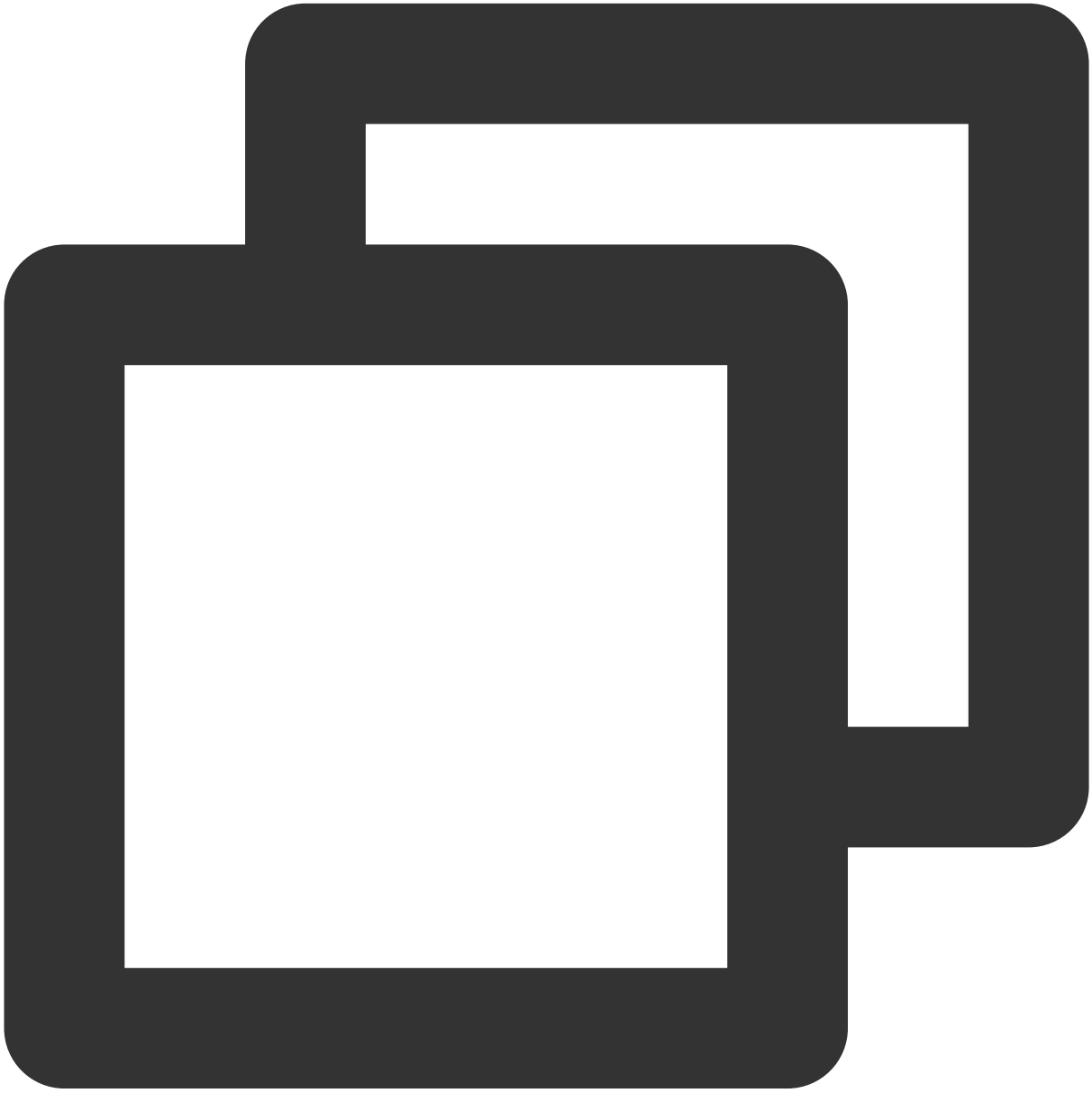
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID. |

| | | |
|---|---|---|
| isAudioAvailable | boolean | Whether the user has audio. |

### onUserVoiceVolumeChanged

The volumes of all users.



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

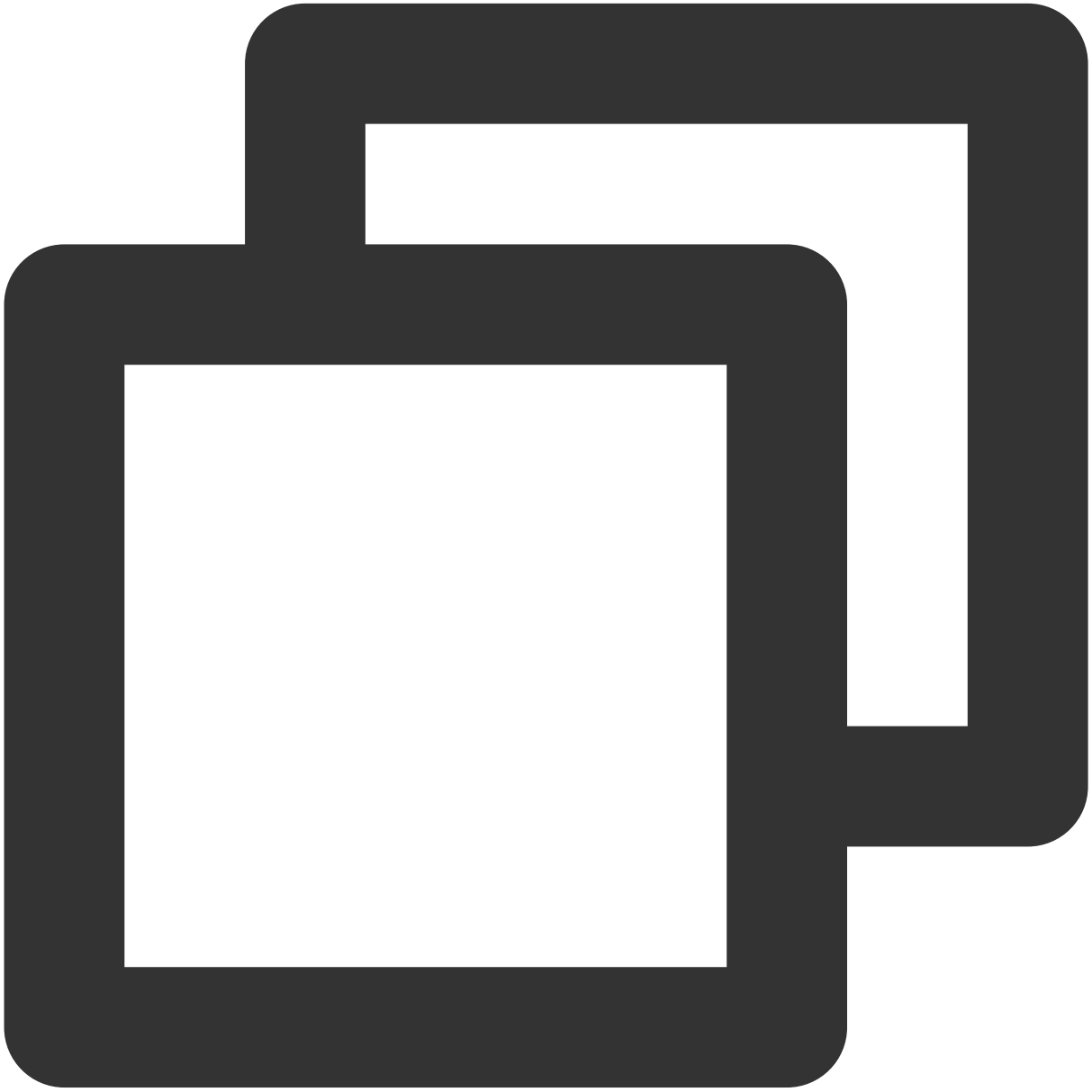
The parameters are described below:

| Parameter | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| volumeMap | Map<String, Integer> | The volume table, which includes the volume of each user ( `userId` ). Value range: 0-100. |

### onUserNetworkQualityChanged

The network quality of all users.



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

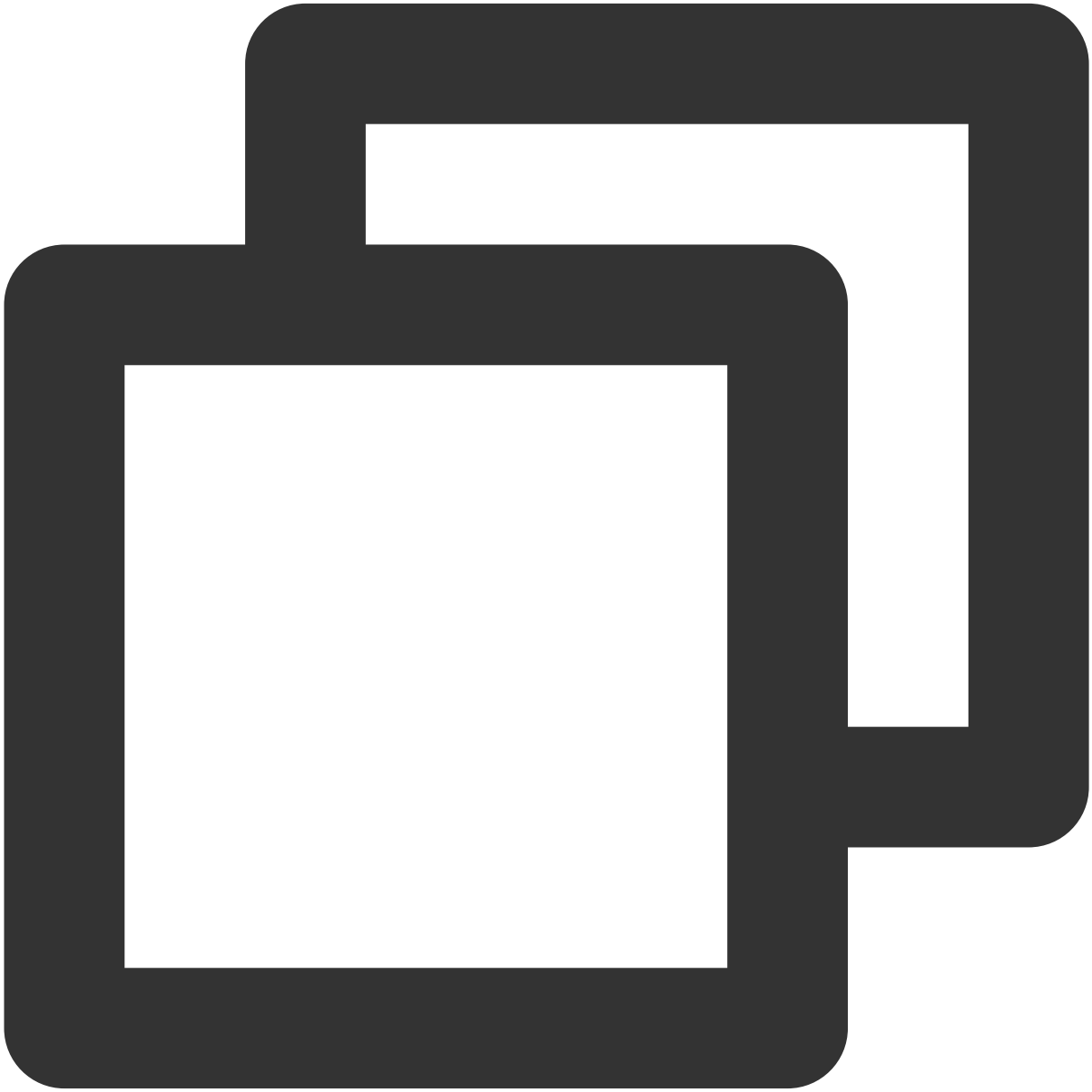
The parameters are described below:

| | | |
|---|---|---|

| Parameter | Type | Description |
| --- | --- | --- |
| networkQualityList | List | The current network conditions for all users ( `userId` ). |

### onKickedOffline

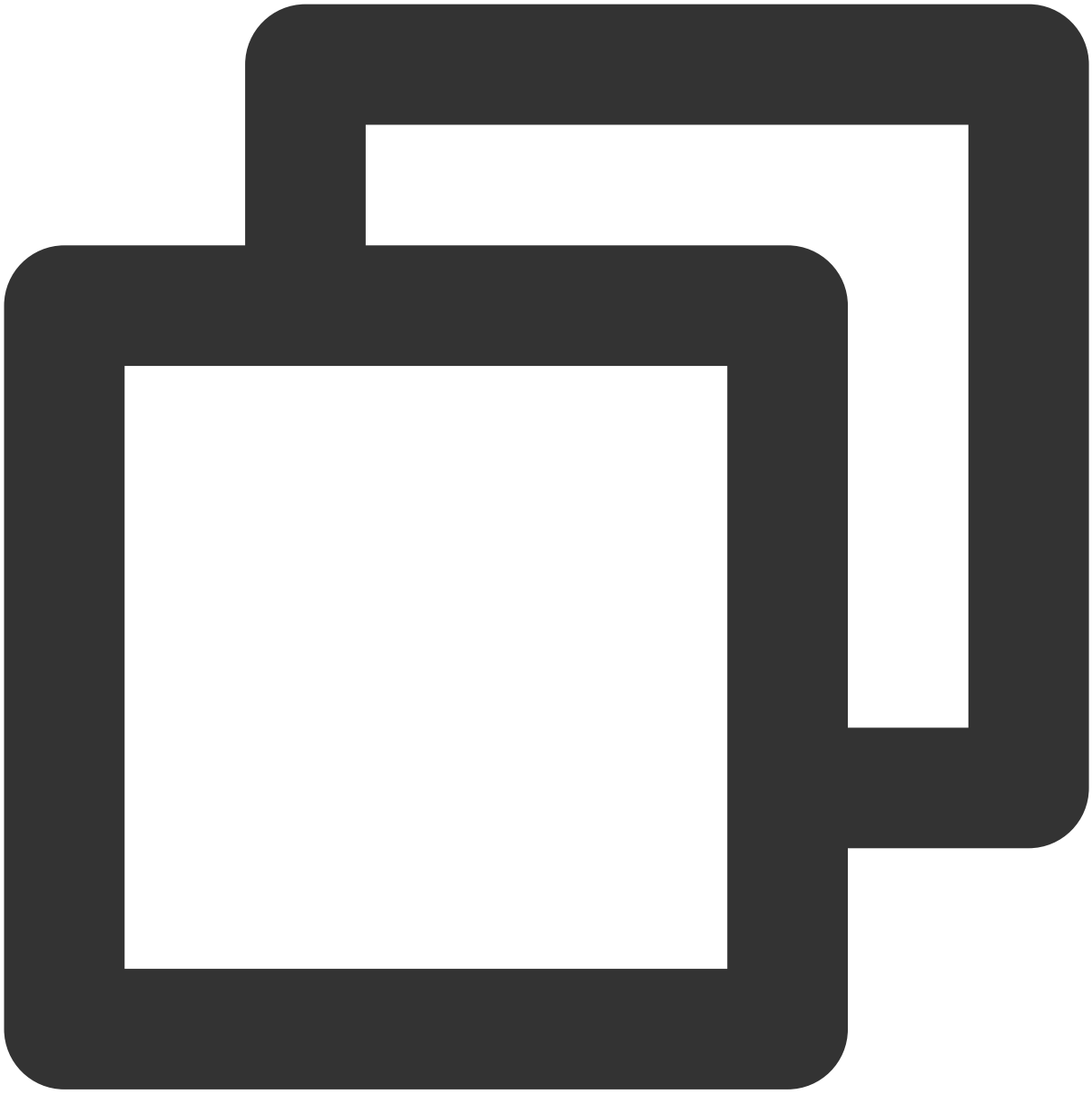
The current user was kicked offline：At this time, you can prompt the user with a UI message and then invoke `init` again.



```
void onKickedOffline();
```

### onUserSigExpired

The user sig is expired：At this time, you need to generate a new `userSig` , and then invoke `init` again.



```
void onUserSigExpired();
```

# Type Definition

Last updated：2023-06-30 15:41:06

## Common structures

### TUICallDefine

| Type | Description |
| --- | --- |
| CallParams | An additional parameter. |
| OfflinePushInfo | Offline push vendor configuration information. |

### TUICommonDefine

| Type | Description |
| --- | --- |
| RoomId | Room ID for audio and video in a call. |
| NetworkQualityInfo | Network quality information |
| VideoRenderParams | Video render parameters |
| VideoEncoderParams | Video encoding parameters |

## Enum definition

### TUICallDefine

| Type | Description |
| --- | --- |
| MediaType | Media type in a call |
| Role | Roles of individuals in a call. |
| Status | The call status |
| Scene | The call scene |
| IOSOfflinePushType | iOS offline push type |

### TUICommonDefine

| Type | Description |
| --- | --- |
| | |

| | |
|---|---|
| AudioPlaybackDevice | Audio route |
| Camera | Camera type |
| NetworkQuality | Network quality |
| FillMode | Video image fill mode |
| Rotation | Video image rotation direction |
| ResolutionMode | Video aspect ratio mode |
| Resolution | Video resolution |

### CallParams

Call params

| Value | Type | Description |
|---|---|---|
| offlinePushInfo | OfflinePushInfo | Offline push vendor configuration information. |
| timeout | int | Call timeout period, default: 30s, unit: seconds. |
| userData | String | An additional parameter. Callback when the callee receives onCallReceived |

### OfflinePushInfo

Offline push vendor configuration information, please refer to：Offline call push

| Value | Type | Description |
|---|---|---|
| title | String | offlinepush notification title |
| desc | String | offlinepush notification description |
| ignoreIOSBadge | boolean | Ignore badge count for offline push (only for iOS), if set to true, the message will not increase the unread count of the app icon on the iOS receiver's side. |
| iOSSound | String | Offline push sound setting (only for iOS). When `sound = IOS_OFFLINE_PUSH_NO_SOUND`, there will be no sound played when the message is received. When `sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND`, the system sound will be played when the message is received. If you want to customize the iOSSound, you need to link |

| | | |
|---|---|---|
| | | the audio file into the Xcode project first, and then set the audio file name (with extension) to the iOSSound. |
| androidSound | String | Offline push sound setting (only for Android, supported by IMSDK 6.1 and above). Only Huawei and Google phones support setting sound prompts. For Xiaomi phones, please refer to: Xiaomi custom ringtones. In addition, for Google phones, in order to set sound prompts for FCM push on Android 8.0 and above systems, you must call `setAndroidFCMChannelID` to set the channelID for it to take effect. |
| androidOPPOChannelID | String | Set the channel ID for OPPO phones with Android 8.0 and above systems. |
| androidVIVOClassification | int | Classification of VIVO push messages (deprecated interface, VIVO push service will optimize message classification rules on April 3, 2023. It is recommended to use setAndroidVIVOCategory to set the message category). 0: Operational messages, 1: System messages. The default value is 1. |
| androidXiaoMiChannelID | String | Set the channel ID for Xiaomi phones with Android 8.0 and above systems. |
| androidFCMChannelID | String | Set the channel ID for google phones with Android 8.0 and above systems. |
| androidHuaWeiCategory | String | Classification of Huawei push messages, please refer to: Huawei message classification standard. |
| isDisablePush | boolean | Whether to turn off push notifications (default is on). |
| iOSPushType | IOSOfflinePushType | iOS offline push type， default is APNs |

### RoomId

Room ID for audio and video in a call.

**Note：**

(1) `intRoomId` and `strRoomId` are mutually exclusive. If you choose to use `strRoomId`, `intRoomId` needs to be filled in as 0. If both are filled in, the SDK will prioritize `intRoomId`.

(2) Do not mix `intRoomId` and `strRoomId` because they are not interchangeable. For example, the number 123 and the string "123" represent two completely different rooms.

| Value | Type | Description |
| --- | --- | --- |
| intRoomId | int | Numeric room ID.<br>**range**：1 - 2147483647(2^31-1) |
| strRoomId | String | String room ID.<br>**range**：<br>Limited to 64 bytes in length. The supported character set range is as follows (a total of 89<br>Lowercase and uppercase English letters.（a-zA-Z）；<br>Number（0-9）；<br>`Spaces`、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`<br>。 |

**Note：**

Currently, string room number is only supported on Android and iOS platforms. Support for other platforms such as Web, Mini Programs, Flutter, and Uniapp will be available in the future. Please stay tuned!

### NetworkQualityInfo

User network quality information

| Value | Type | Description |
| --- | --- | --- |
| userId | String | user ID |
| quality | NetworkQuality | network quality |

### VideoRenderParams

Video render parameters

| Value | Type | Description |
| --- | --- | --- |
| fillMode | VideoRenderParams.FillMode | Video image fill mode |
| rotation | VideoRenderParams.Rotation | Video image rotation direction |

### VideoEncoderParams

Video encoding parameters

| Value | Type | Description |
| --- | --- | --- |
| resolution | VideoEncoderParams.Resolution | Video resolution |

| | | |
|---|---|---|
| resolutionMode | VideoEncoderParams.ResolutionMode | Video aspect ratio mode |

### MediaType

Call media type

| Value | Type | Description |
|---|---|---|
| Unknown | 0 | Unknown |
| Audio | 1 | Audio call |
| Video | 2 | Video call |

### Role

Call role

| Value | Type | Description |
|---|---|---|
| None | 0 | Unknown |
| Caller | 1 | Caller（inviter） |
| Called | 2 | Callee（invitee） |

### Status

Call status

| Value | Type | Description |
|---|---|---|
| None | 0 | Unknown |
| Waiting | 1 | The call is currently waiting |
| Accept | 2 | The call has been accepted |

### Scene

Call scene

| Value | Type | Description |
|---|---|---|
| GROUP_CALL | 0 | Group call |
| MULTI_CALL | 1 | Anonymous group calling (not supported at this moment, please stay tuned). |

| | | |
|---|---|---|
| SINGLE_CALL | 2 | one to one call |

### IOSOfflinePushType

iOS offline push type

| Type | Value | Description |
|---|---|---|
| APNs | 0 | APNs |
| VoIP | 1 | VoIP |

### AudioPlaybackDevice

Audio route

| Type | Value | Description |
|---|---|---|
| Earpiece | 0 | Earpiece |
| Speakerphone | 1 | Speakerphone |

### Camera

Front/Back camera

| Type | Value | Description |
|---|---|---|
| Front | 0 | Front camera |
| Back | 1 | Back camera |

### NetworkQuality

Network quality

| Type | Value | Description |
|---|---|---|
| Unknown | 0 | Unknown |
| Excellent | 1 | Excellent |
| Good | 2 | Good |
| Poor | 3 | Poor |
| Bad | 4 | Bad |
| Vbad | 5 | Vbad |

| | | |
|---|---|---|
| Down | 6 | Down |

### FillMode

If the aspect ratio of the video display area is not equal to that of the video image, you need to specify the fill mode:

| Type | Value | Description |
|---|---|---|
| Fill | 0 | Fill mode: the video image will be centered and scaled to fill the entire display area, where parts that exceed the area will be cropped. The displayed image may be incomplete in this mode. |
| Fit | 1 | Fit mode: the video image will be scaled based on its long side to fit the display area, where the short side will be filled with black bars. The displayed image is complete in this mode, but there may be black bars. |

### Rotation

We provides rotation angle setting APIs for local and remote images. The following rotation angles are all clockwise.

| Type | Value | Description |
|---|---|---|
| Rotation_0 | 0 | No rotation |
| Rotation_90 | 1 | Clockwise rotation by 90 degrees |
| Rotation_180 | 2 | Clockwise rotation by 180 degrees |
| Rotation_270 | 3 | Clockwise rotation by 0 degrees |

### ResolutionMode

Video aspect ratio mode

| Type | Value | Description |
|---|---|---|
| Landscape | 0 | Landscape resolution, such as Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360. |
| Portrait | 1 | Portrait resolution, such as Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640. |

### Resolution

Video resolution

| Type | Value | Description |
| --- | --- | --- |
| Resolution_640_360 | 108 | Aspect ratio: 16:9；resolution: 640x360；recommended bitrate: 500kbps |
| Resolution_640_480 | 62 | Aspect ratio: 4:3；resolution: 640x480；recommended bitrate: 600kbps |
| Resolution_960_540 | 110 | Aspect ratio: 16:9；resolution: 960x540；recommended bitrate: 850kbps |
| Resolution_960_720 | 64 | Aspect ratio: 4:3；resolution: 960x720；recommended bitrate: 1000kbps |
| Resolution_1280_720 | 112 | Aspect ratio: 16:9；resolution: 1280x720；recommended bitrate: 1200kbps |
| Resolution_1920_1080 | 114 | Aspect ratio: 16:9；resolution: 1920x1080；recommended bitrate: 2000kbps |

# ErrorCode

Last updated：2023-06-30 15:42:26

Notify users of warnings and errors that occur during audio and video calls.

## TUICallDefine Error Code

| Definition | Value | Description |
| --- | --- | --- |
| ERROR_PACKAGE_NOT_PURCHASED | -1001 | You do not have TUICallKit package, please open the free experience in the console or purchase the official package. |
| ERROR_PACKAGE_NOT_SUPPORTED | -1002 | The package you purchased does not support this ability. You can refer to console to purchase Upgrade package. |
| ERROR_TIM_VERSION_OUTDATED | -1003 | The IM SDK version is too low, please upgrade the IM SDK version to >= 6.6; Find and modify in the build.gradle file.："com.tencent.imsdk:imsdk-plus:7.1.3925" |
| ERROR_PERMISSION_DENIED | -1101 | Failed to obtain permission. The audio/video permission is not authorized. Check if the device permission is enabled. |
| ERROR_GET_DEVICE_LIST_FAIL | -1102 | Failed to get the device list (only supported on web platform). |
| ERROR_INIT_FAIL | -1201 | The init method has not been called for initialization. The TUICallEngine API should be used after initialization. |
| ERROR_PARAM_INVALID | -1202 | param is invalid. |
| ERROR_REQUEST_REFUSED | -1203 | The current status can't use this function. |
| ERROR_REQUEST_REPEATED | -1204 | The current status is waiting/accept, please do not call it repeatedly. |
| ERROR_SCENE_NOT_SUPPORTED | -1205 | The current calling scene does not support this feature. |
| ERROR_SIGNALING_SEND_FAIL | -1406 | Failed to send signaling. You can check the specific |

| | | |
|---|---|---|
| | | error message in the callback of the method. |

## IM Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for communication, such as the core logic of call signaling and busy signaling. Common error codes are as follows:

| Error Code | Description |
|---|---|
| 6014 | You have not logged in to the Chat SDK or have been forcibly logged out. Log in to the Chat SDK first and try again after a successful callback. To check whether you are online, use TIMManager getLoginUser. |
| 6017 | Invalid parameter. Check the error information to locate the invalid parameter. |
| 6206 | UserSig has expired. Get a new valid UserSig and log in again. For more information about how to get a UserSig, see Generating UserSig. |
| 7013 | The current package does not support this API. Please upgrade to the Flagship Edition package. |
| 8010 | The signaling request ID is invalid or has been processed. |

Explanation：
More IM SDK error codes are available at：IM Error Code

## TRTC Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for calling, such as the core logic of switching camera and microphone on or off. Common error codes are as follows:

| Enum | Value | Description |
|---|---|---|
| ERR_CAMERA_START_FAIL | -1301 | Failed to turn the camera on. This may occur when there is a problem with the camera configuration program (driver) on Windows or macOS. Disable and reenable the camera, restart the camera, or update the configuration program. |
| ERR_CAMERA_NOT_AUTHORIZED | -1314 | No permission to access to the camera. This usually occurs on mobile devices and may be because the user denied access. |
| | | |

| | | |
|---|---|---|
| ERR_CAMERA_SET_PARAM_FAIL | -1315 | Incorrect camera parameter settings (unsupported values or others). |
| ERR_CAMERA_OCCUPY | -1316 | The camera is being used. Try another camera. |
| ERR_MIC_START_FAIL | -1302 | Failed to turn the mic on. This may occur when there is a problem with the mic configuration program (driver) on Windows or macOS. Disable and reenable the mic, restart the mic, or update the configuration program. |
| ERR_MIC_NOT_AUTHORIZED | -1317 | No permission to access to the mic. This usually occurs on mobile devices and may be because the user denied access. |
| ERR_MIC_SET_PARAM_FAIL | -1318 | Failed to set mic parameters. |
| ERR_MIC_OCCUPY | -1319 | The mic is being used. The mic cannot be turned on when, for example, the user is having a call on the mobile device. |
| ERR_TRTC_ENTER_ROOM_FAILED | -3301 | Failed to enter the room. For the reason, refer to the error message for -3301. |

Explanation：

More TRTC SDK error codes are available at：TRTC Error Code

# iOS
# API Overview

Last updated：2023-07-06 15:57:03

## TUICallKit (UI Included)

TUICallKit is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat.

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallKit` instance (singleton mode). |
| setSelfInfo | Sets the user nickname and profile photo. |
| call | Makes a one-to-one call. |
| call | Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc |
| groupCall | Makes a group call. |
| groupCall | Makes a group call, Support for custom room ID, call timeout, offline push content, etc |
| joinInGroupCall | Joins a group call. |
| setCallingBell | Sets the ringtone. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |

## TUICallEngine (No UI)

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallEngine` instance (singleton). |

| | |
|---|---|
| destroyInstance | Terminates a `TUICallEngine` instance (singleton). |
| init | Authenticates the basic audio/video call capabilities. |
| addObserver | Registers an event listener. |
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Answers a call. |
| reject | Declines a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Switches the call media type, such as from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |
| openCamera | Turns the camera on. |
| closeCamera | Turns the camera off. |
| switchCamera | Switches the camera. |
| openMicrophone | Enables the mic. |
| closeMicrophone | Disables the mic. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver/speaker). |
| setSelfInfo | Sets the user nickname and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for `TUICallEngine` (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |
| setVideoEncoderParams | Set the encoding parameters of video encoder. |

| | |
|---|---|
| getTRTCCloudInstance | Advanced features. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

### TUICallObserver

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

| API | Description |
|---|---|
| onError | An error occurred during the call. |
| onCallReceived | A call was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user had a video stream. |
| onUserAudioAvailable | Whether a user had an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | The current user was kicked offline. |
| onUserSigExpired | The user sig is expired. |

## Definitions of Key Types

| | |
|---|---|

| API | Description |
| --- | --- |
| TUICallMediaType | The call type. Enumeration: Video call and audio call. |
| TUICallRole | The call role. Enumeration: Caller and callee. |
| TUICallStatus | The call status. Enumeration: Idle, waiting, and answering. |
| TUIRoomId | The room ID, which can be a number or string. |
| TUICallCamera | The camera type. Enumeration: Front camera and rear camera. |
| TUIAudioPlaybackDevice | The audio playback device type. Enumeration: Speaker and receiver. |
| TUINetworkQualityInfo | The current network quality. |

# TUICallKit

Last updated：2023-07-04 14:45:29

## TUICallKit APIs

`TUICallKit` is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat. For directions on integration, see Integrating TUICallKit.

## Overview

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallKit` instance (singleton mode). |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| call | Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc |
| groupCall | Makes a group call. |
| groupCall | Makes a group call, Support for custom room ID, call timeout, offline push content, etc |
| joinInGroupCall | Joins a group call. |
| setCallingBell | Sets the ringtone. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |

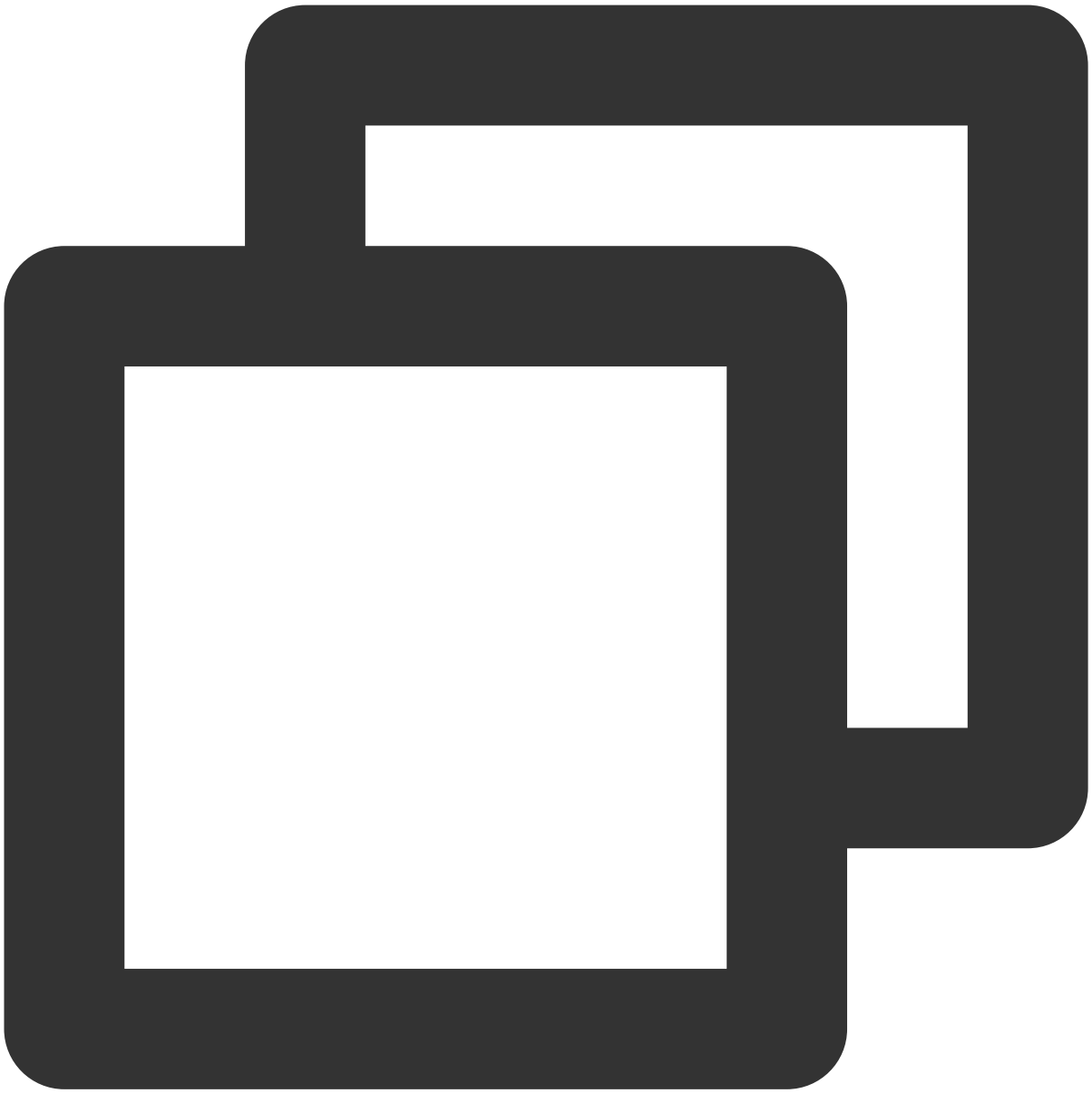
## Details

### createInstance

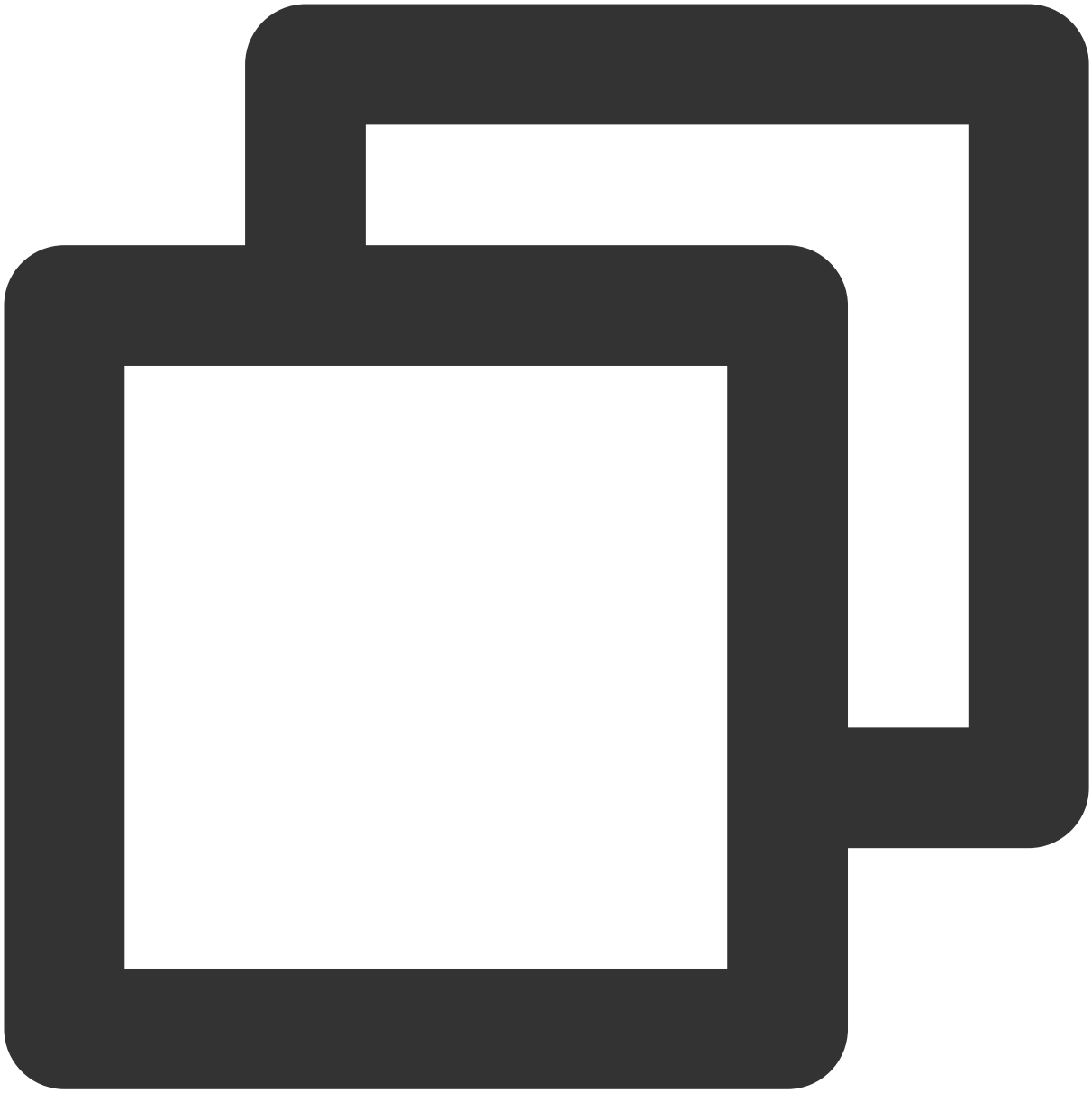
This API is used to create a `TUICallKit` singleton.

Objective-C

Swift



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

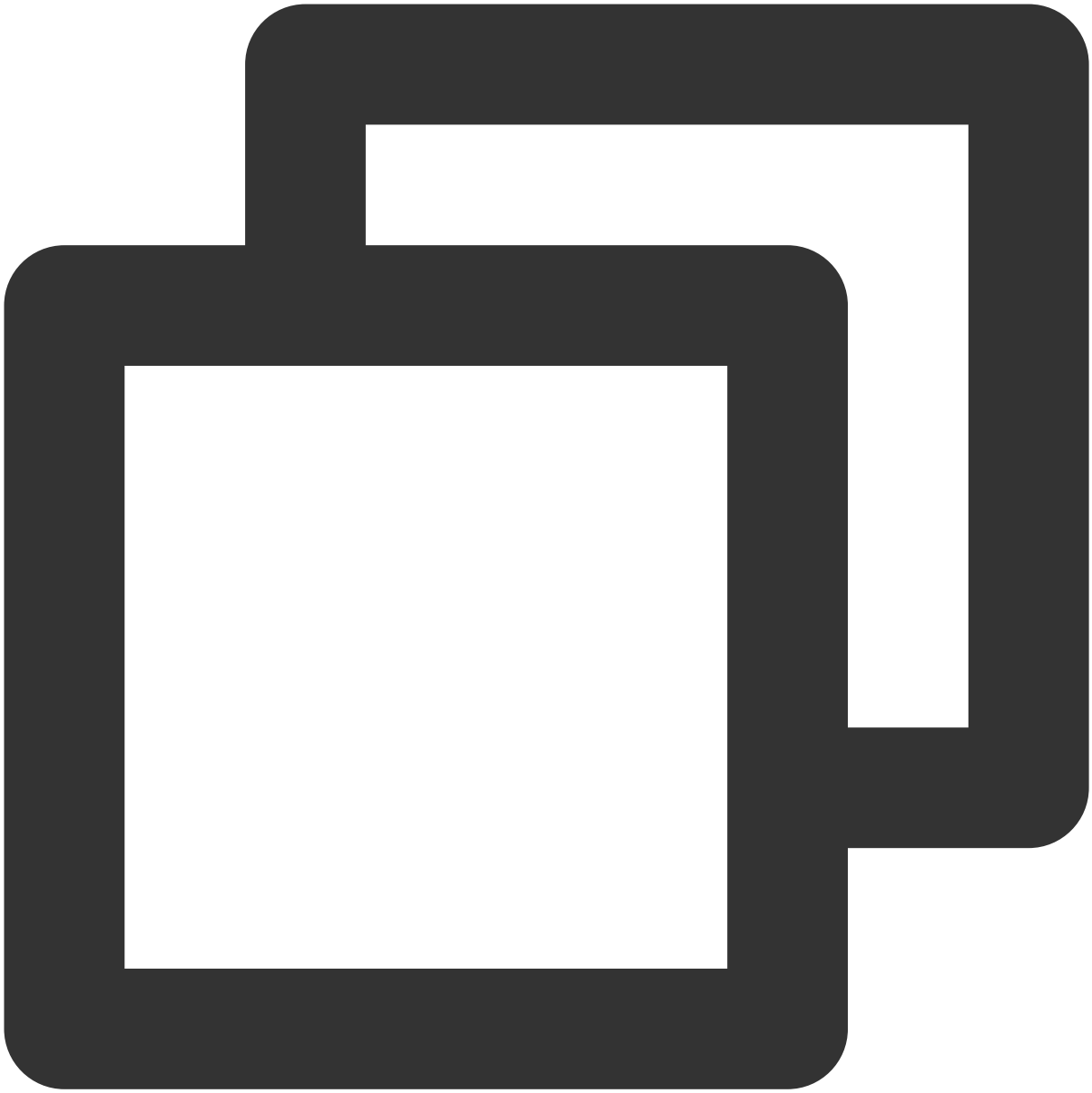


```
public static func createInstance() -> TUICallKit
```

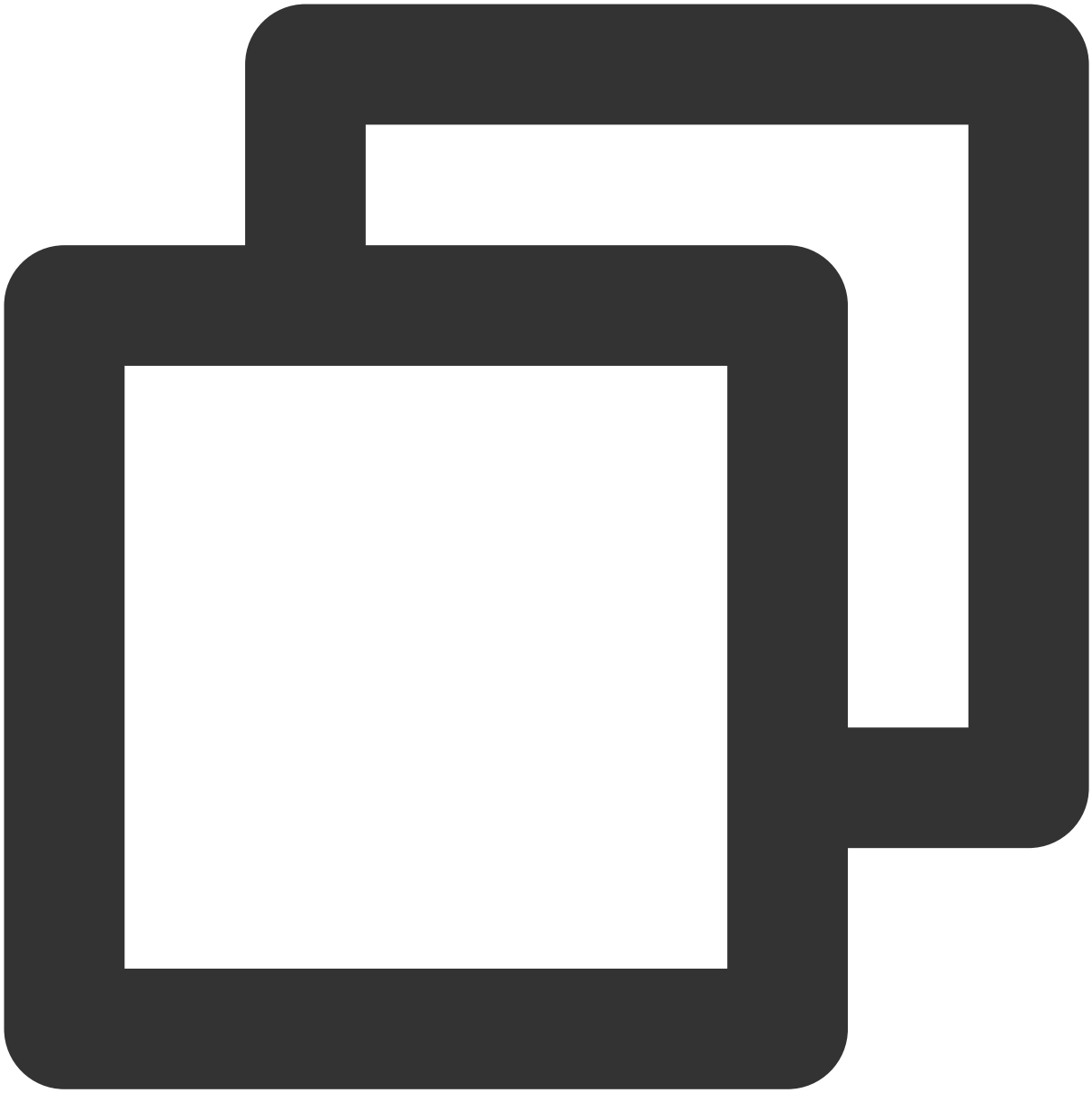
### setSelfInfo

This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.

Objective-C
Swift



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



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

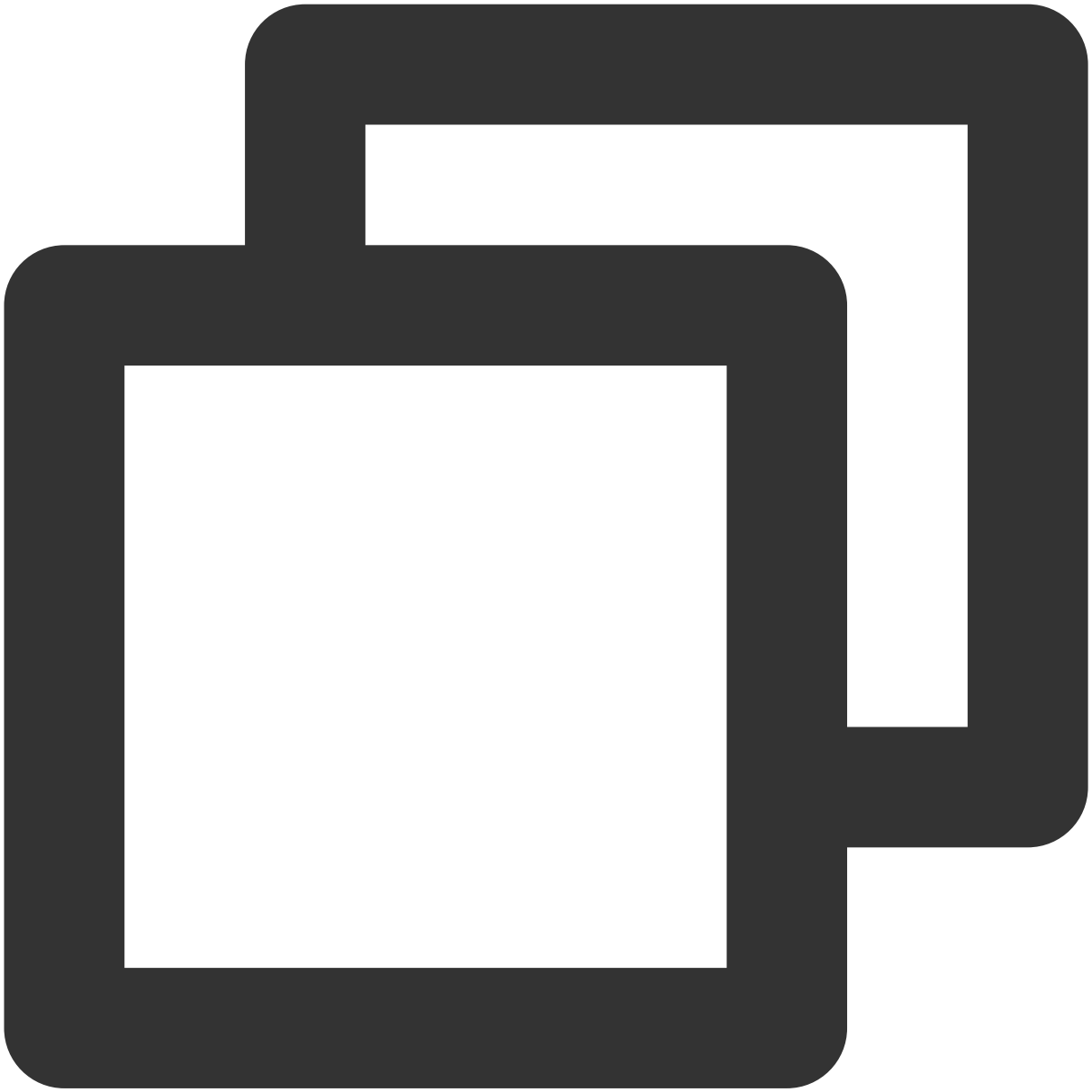
| Parameter | Type | Description |
| --- | --- | --- |
| nickName | NSString | The alias. |
| avatar | NSString | The profile photo. |

### call

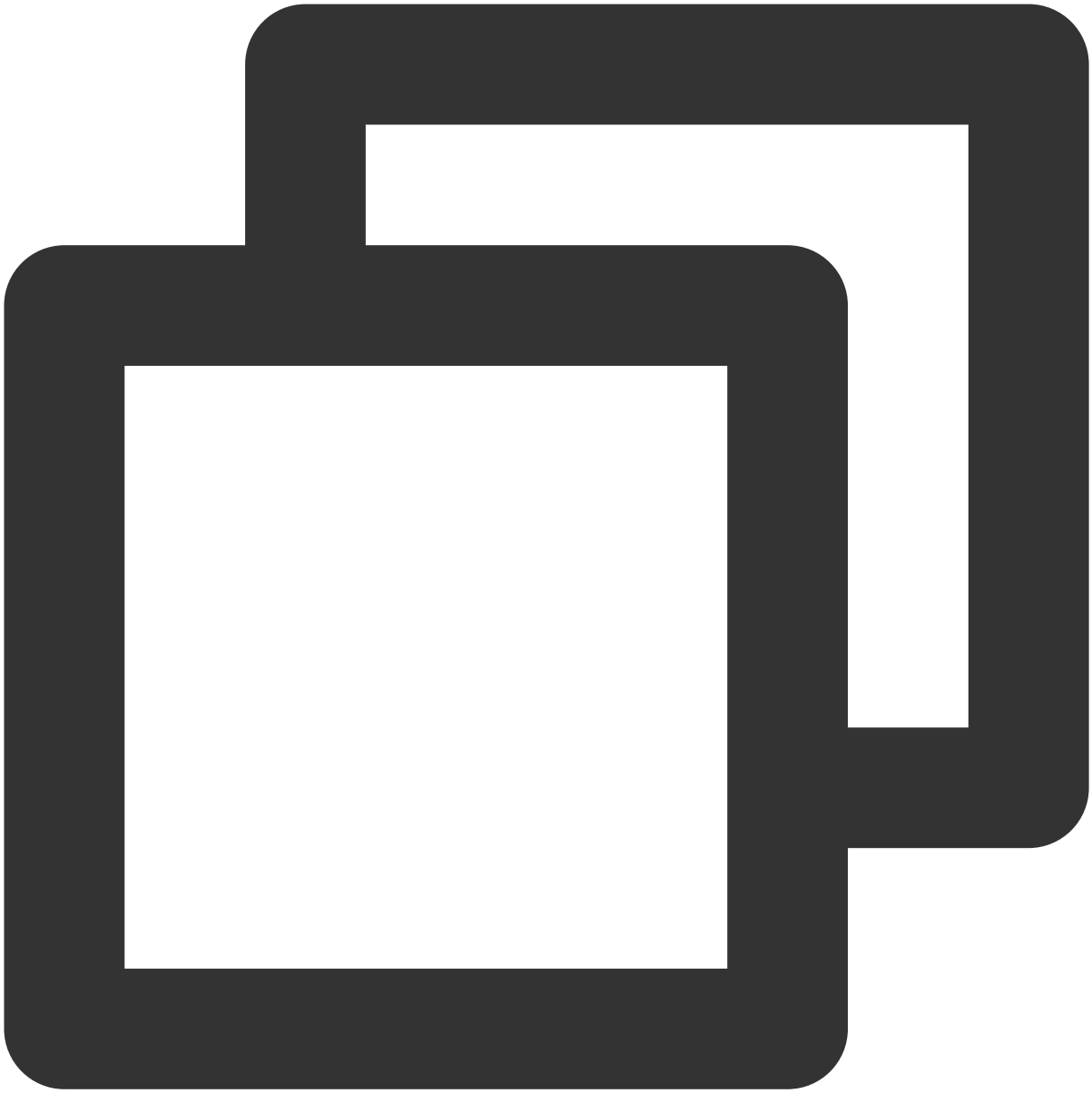
This API is used to make a (one-to-one) call.

Objective-C

Swift



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



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

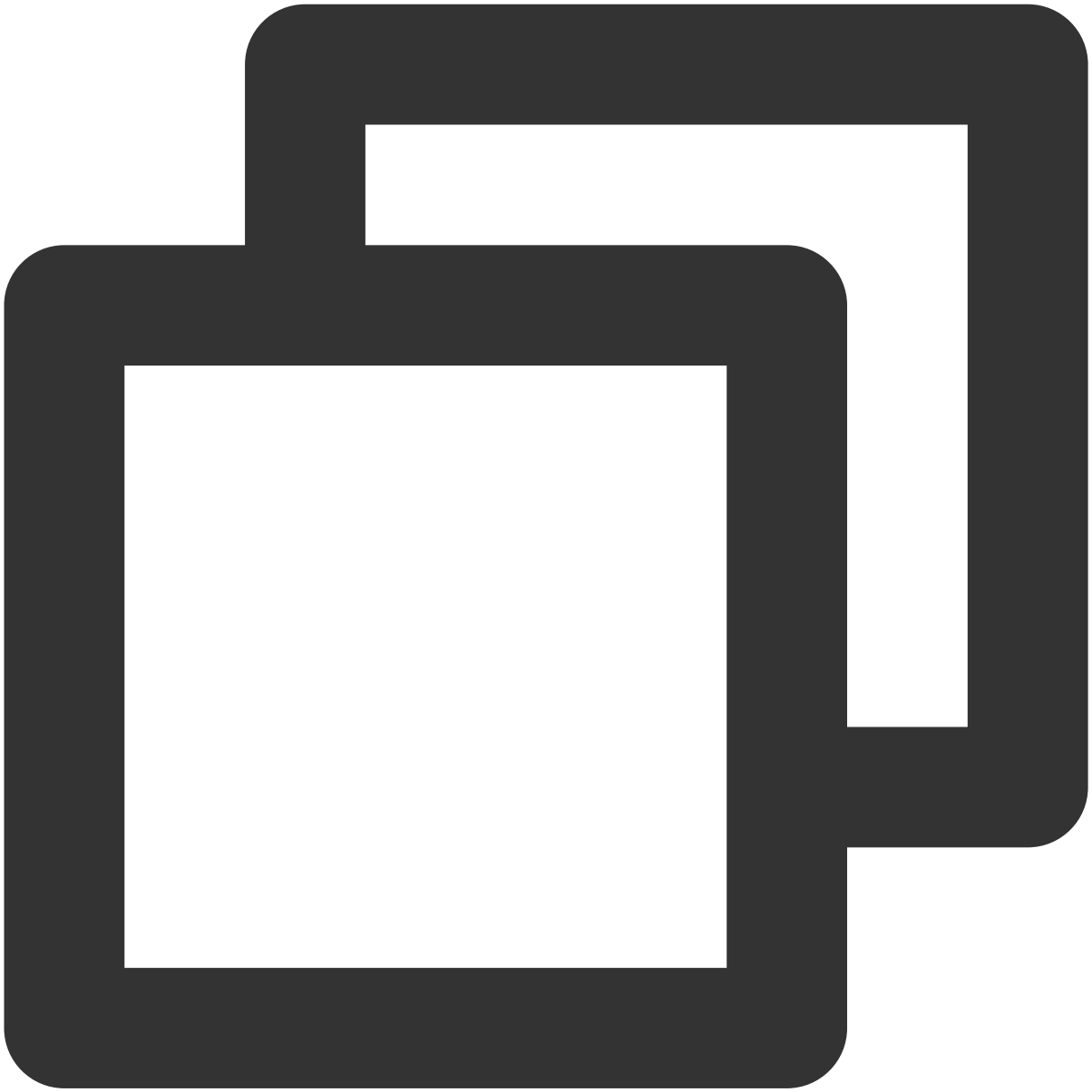
| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### call

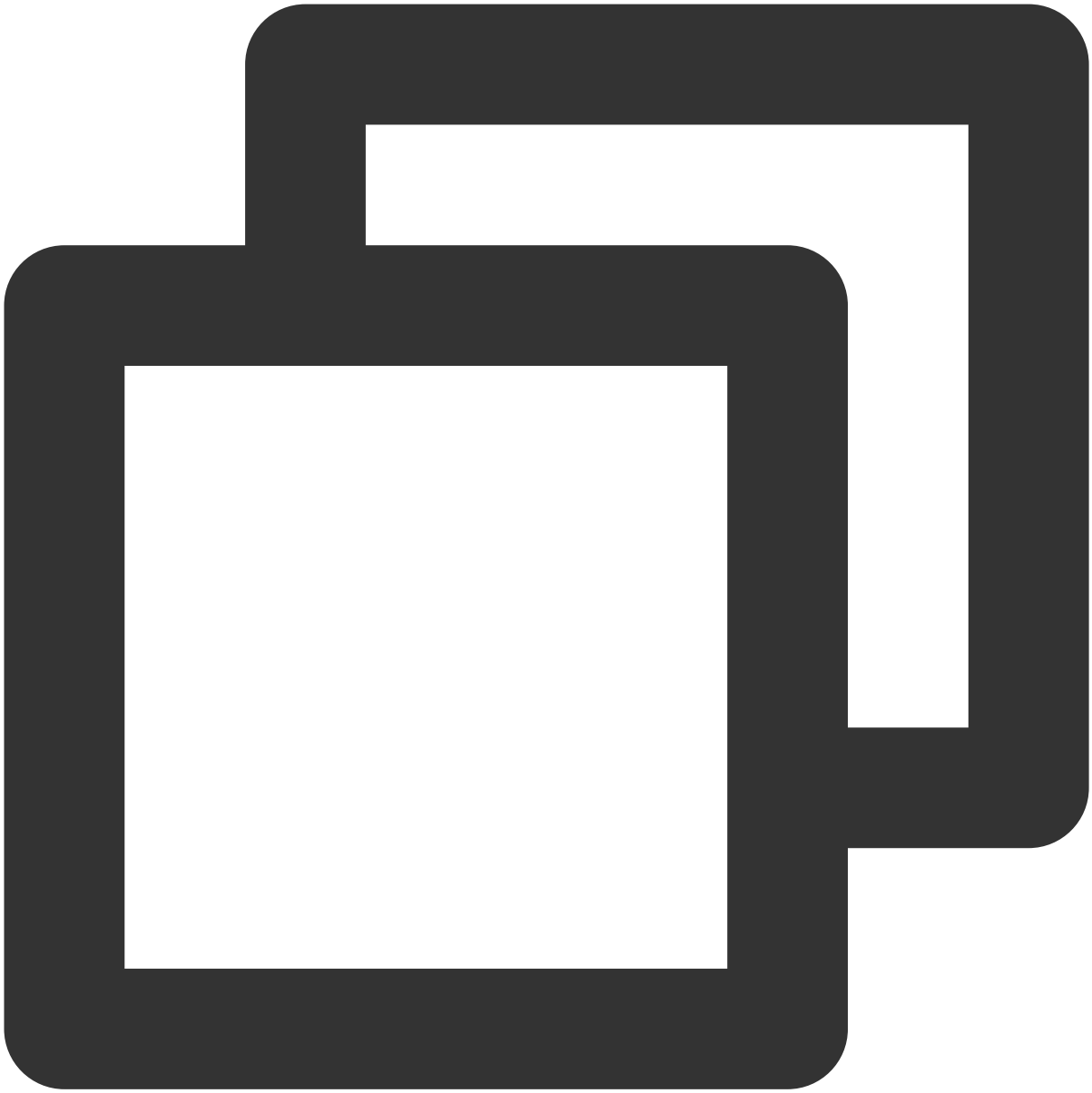
This API is used to make a (one-to-one) call, Support for custom room ID, call timeout, offline push content, etc.

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)
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | Call extension parameters, such as roomID, call timeout, offline push info,etc |

### groupCall

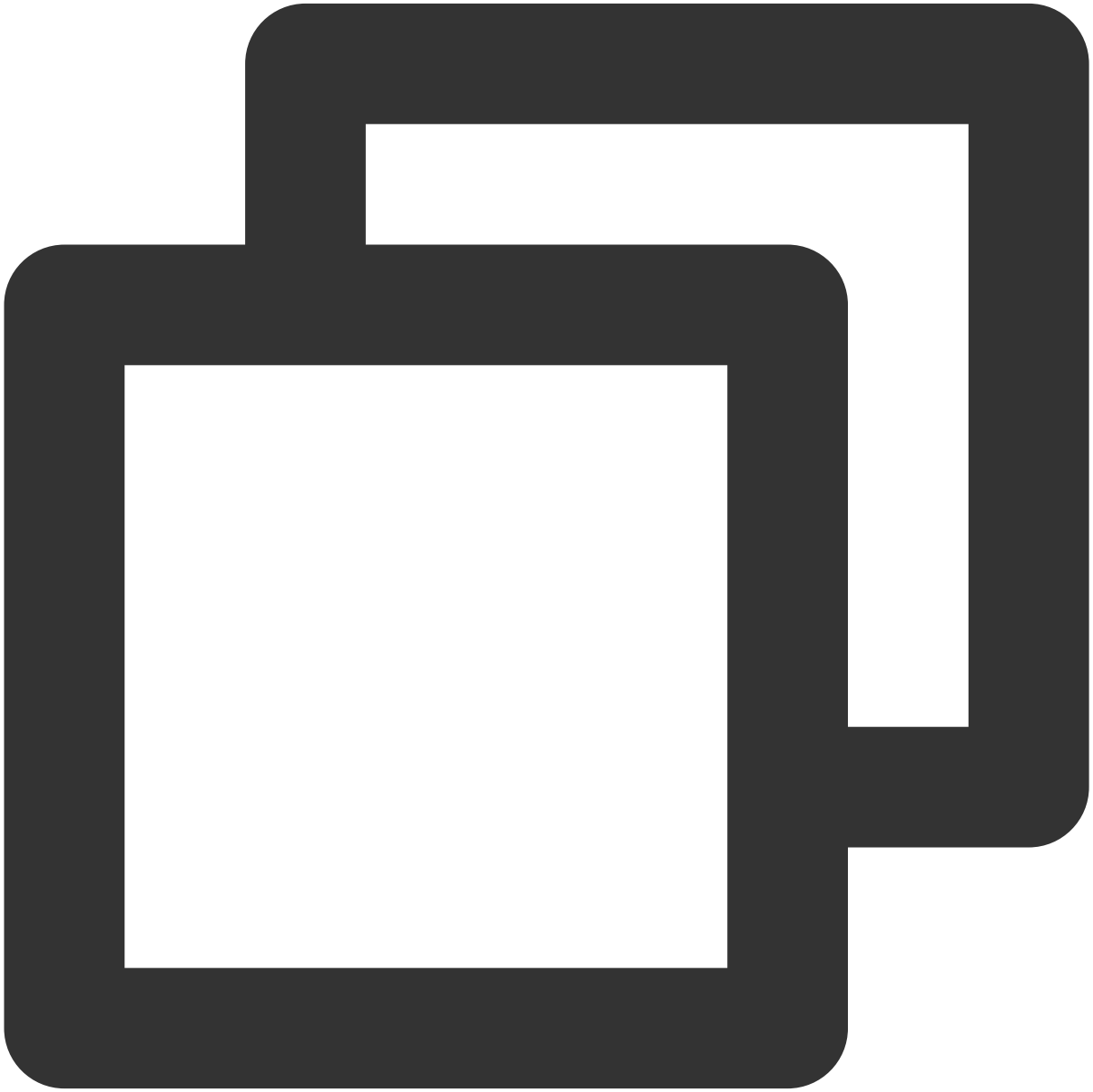
This API is used to make a group call.

注意：

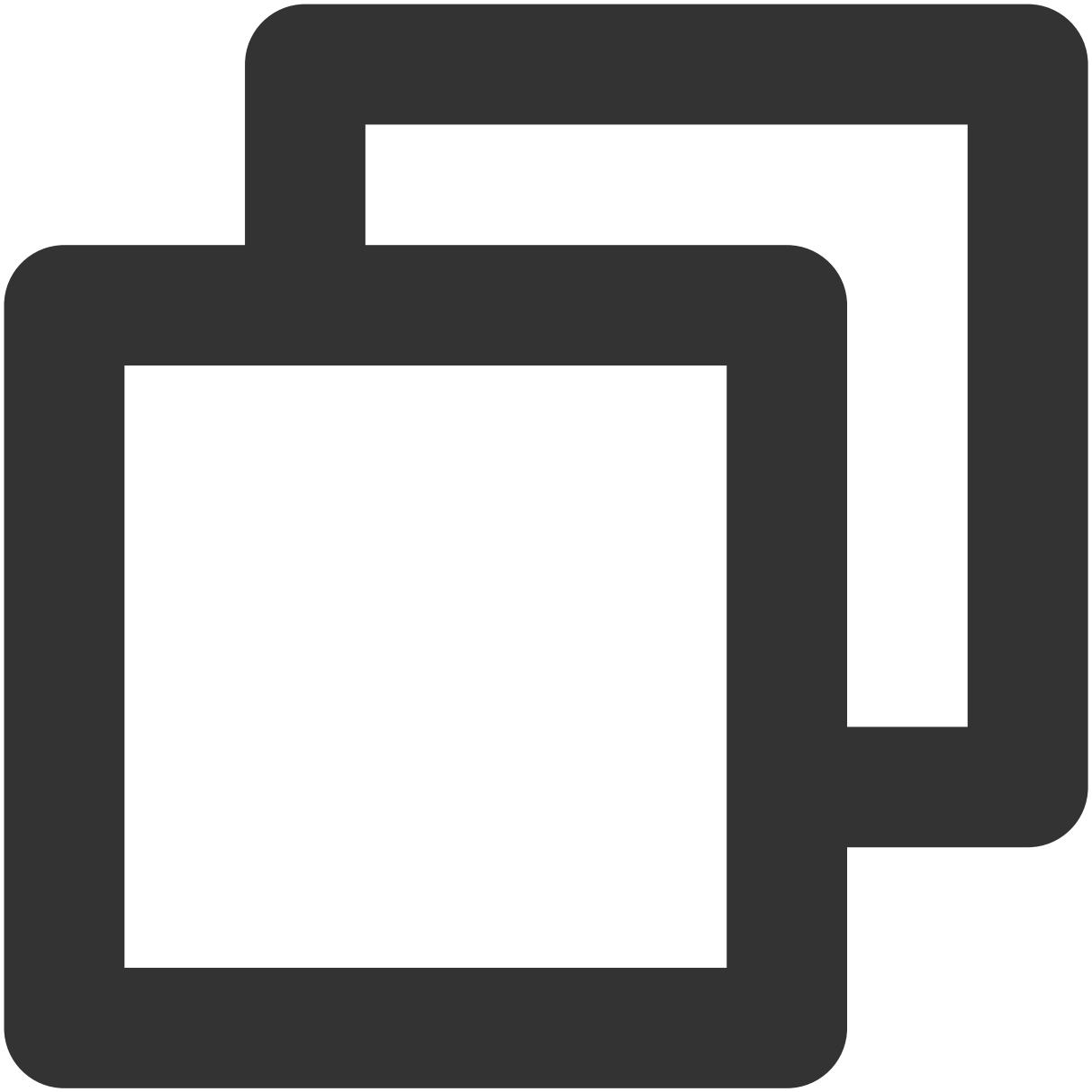
Before making a group call, you need to create an IM group first.

Objective-C

Swift



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



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

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | NSString | The group ID. |
| userIdList | NSArray | The target user IDs. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### groupCall

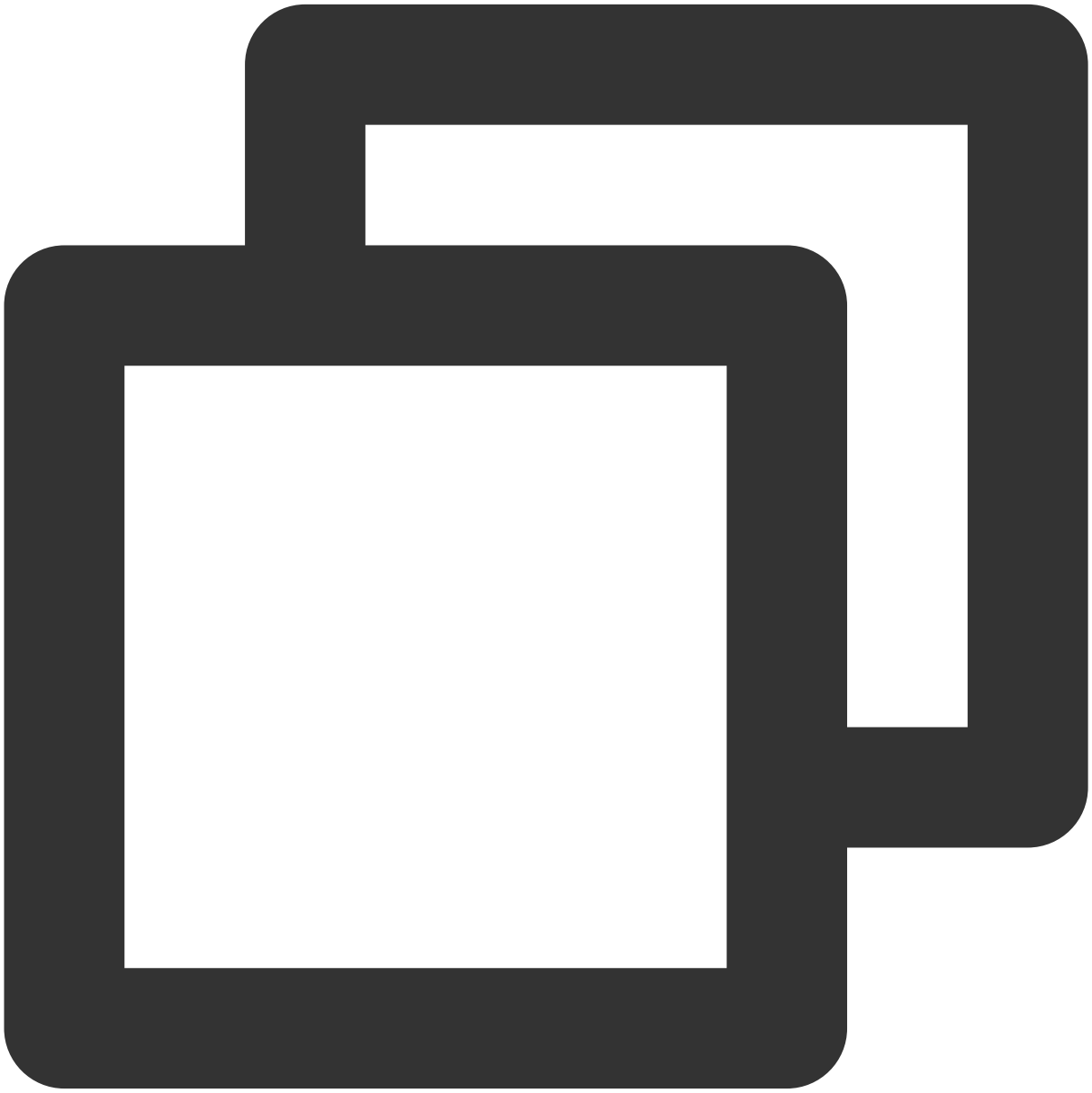
This API is used to make a group call, Support for custom room ID, call timeout, offline push content, etc.

**Notice：**

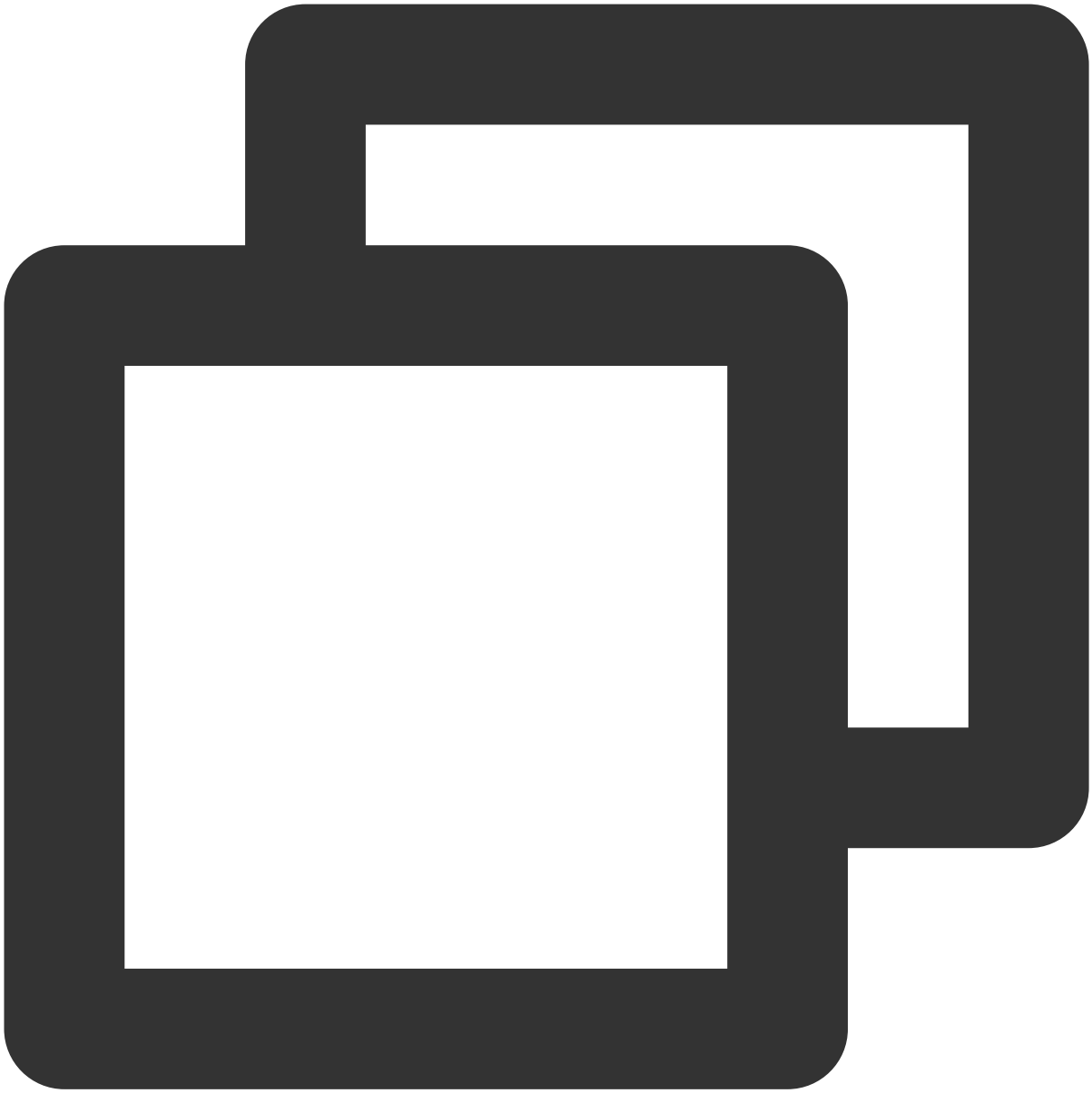
Before making a group call, you need to create an IM group first.

Objective-C

Swift



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



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

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | NSString | The group ID. |
| userIdList | NSArray | The target user IDs. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | Call extension parameters, such as roomID, call timeout, offline |

| | | |
|---|---|---|
| | | push info, etc |

### joinInGroupCall

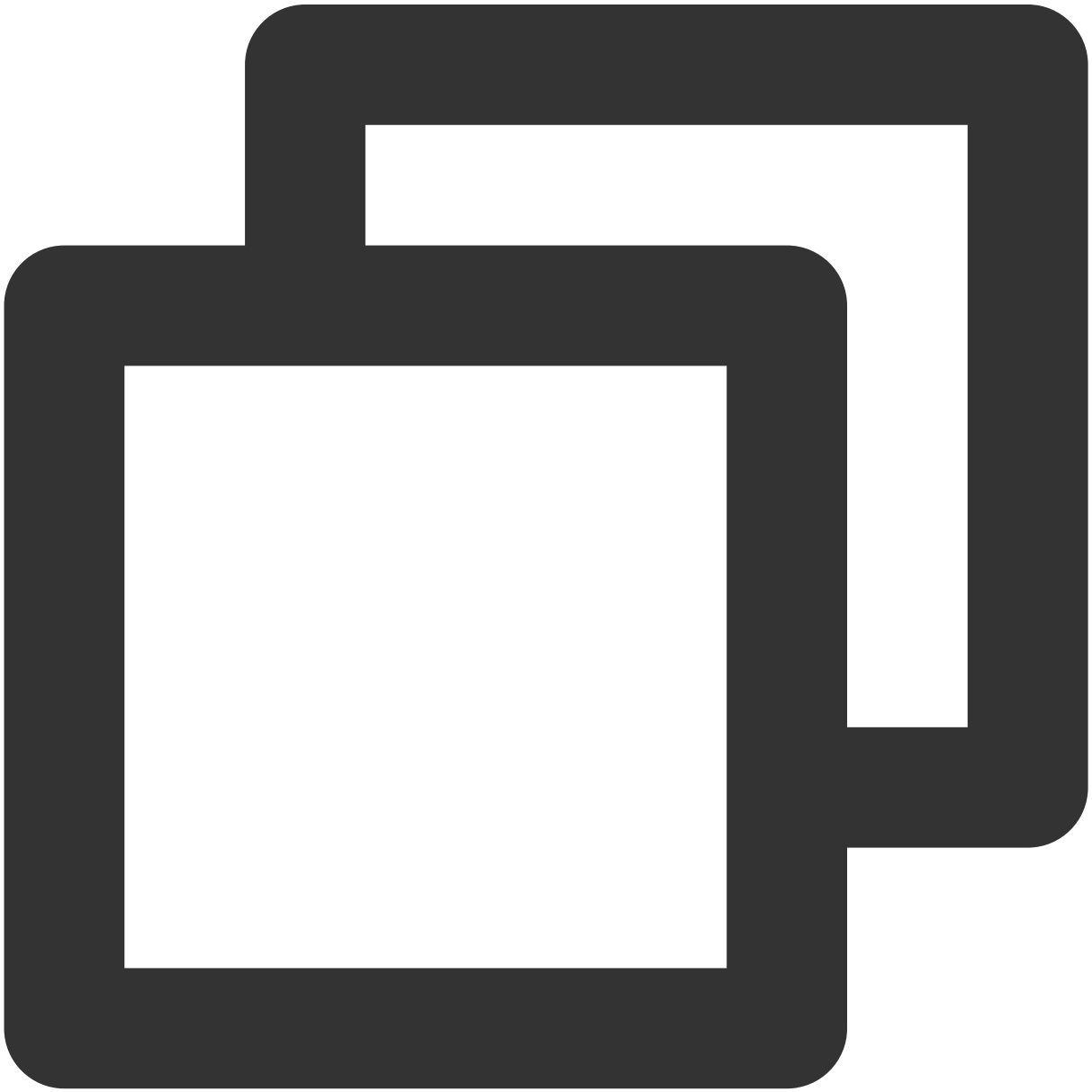
This API is used to join a group call.

**Notice:**

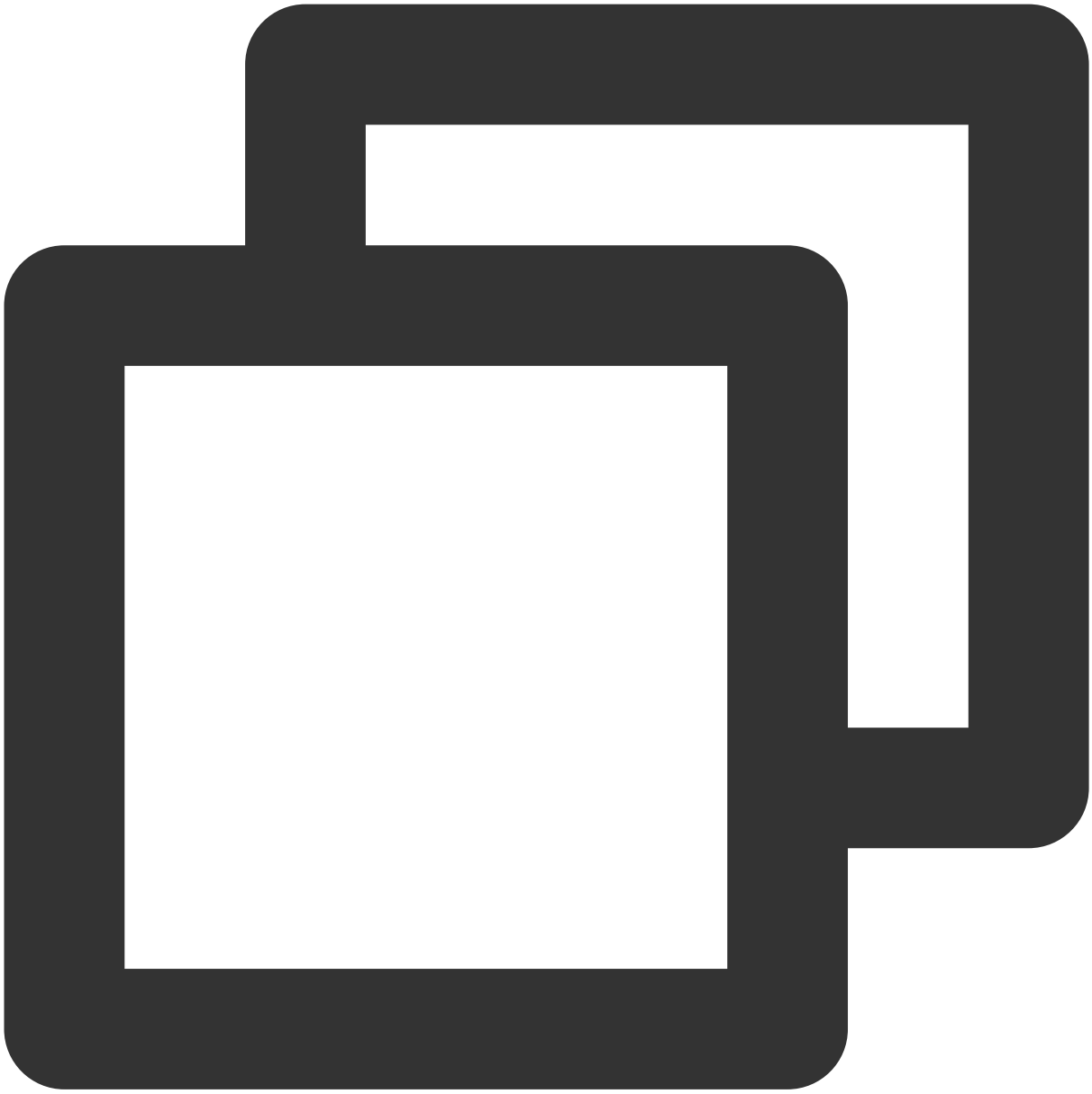
Before joining a group call, you need to create or join an IM group in advance, and there are already users in the group who are in the call.

Objective-C

Swift



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



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

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |
| groupId | NSString | The group ID. |
| | | |

| | | |
|---|---|---|
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### setCallingBell

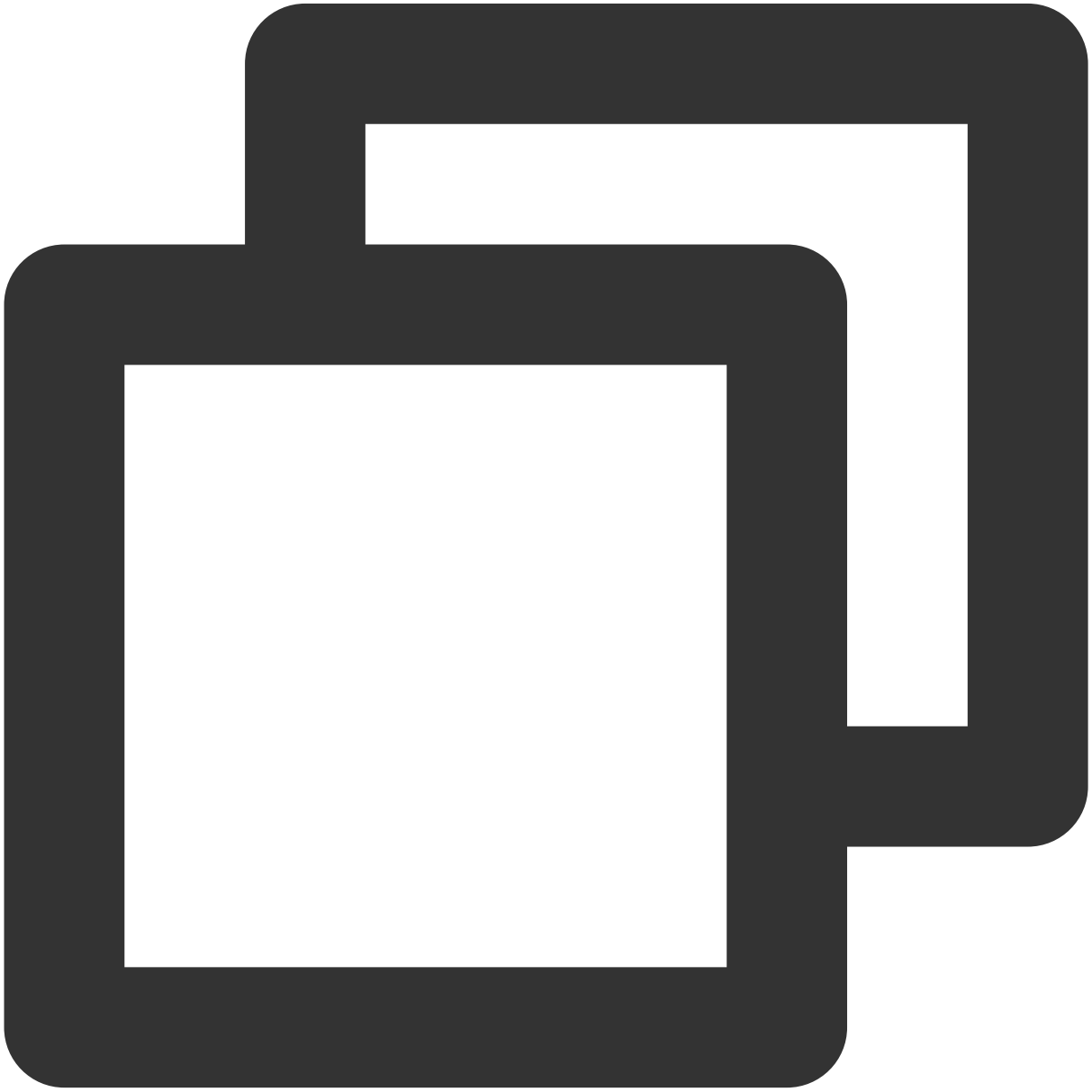
This API is used to set the ringtone. `filePath` must be an accessible local file URL.

The ringtone set is associated with the device and does not change with user.

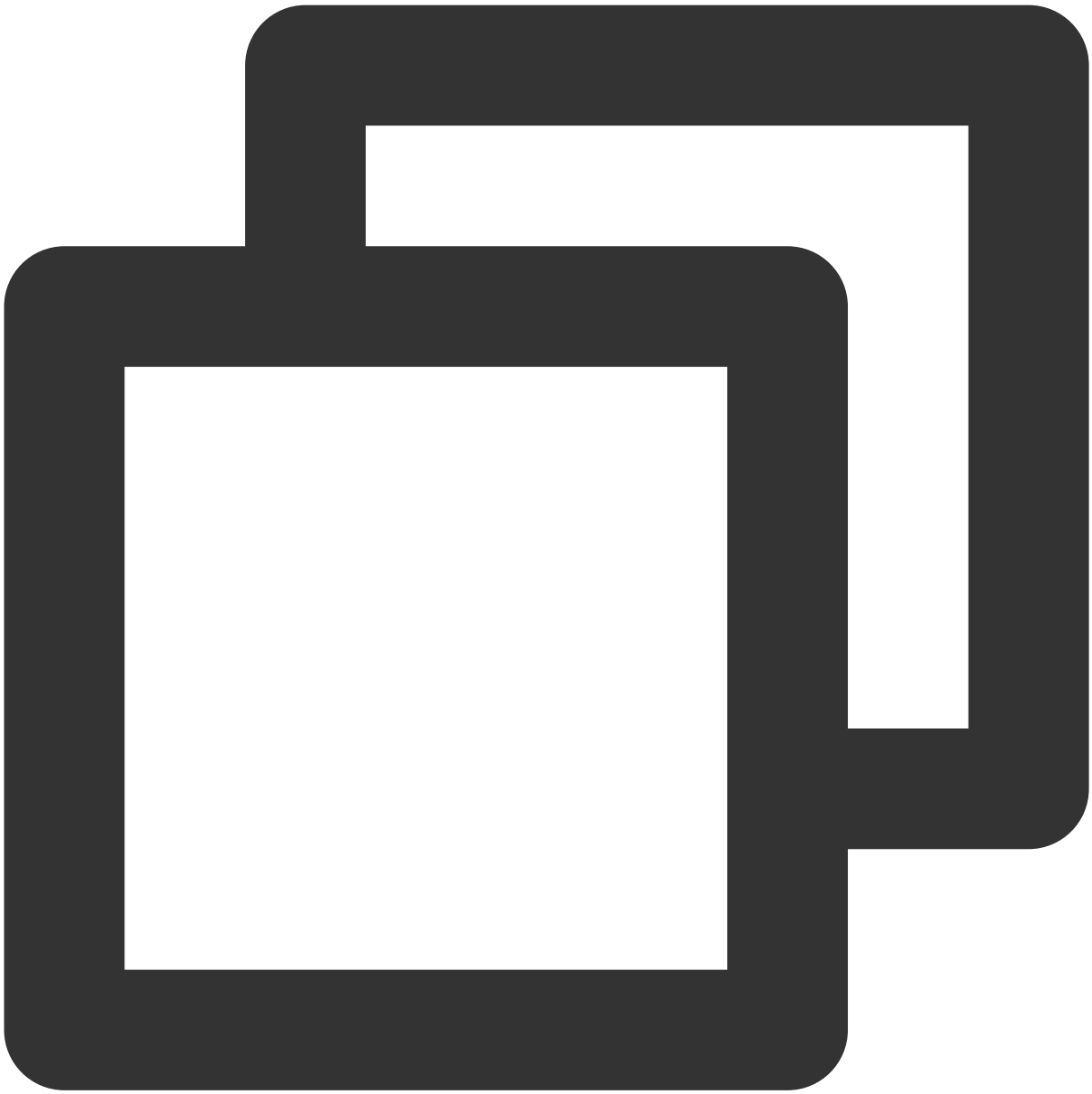
To reset the ringtone, pass in an empty string for `filePath` .

Objective-C

Swift



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



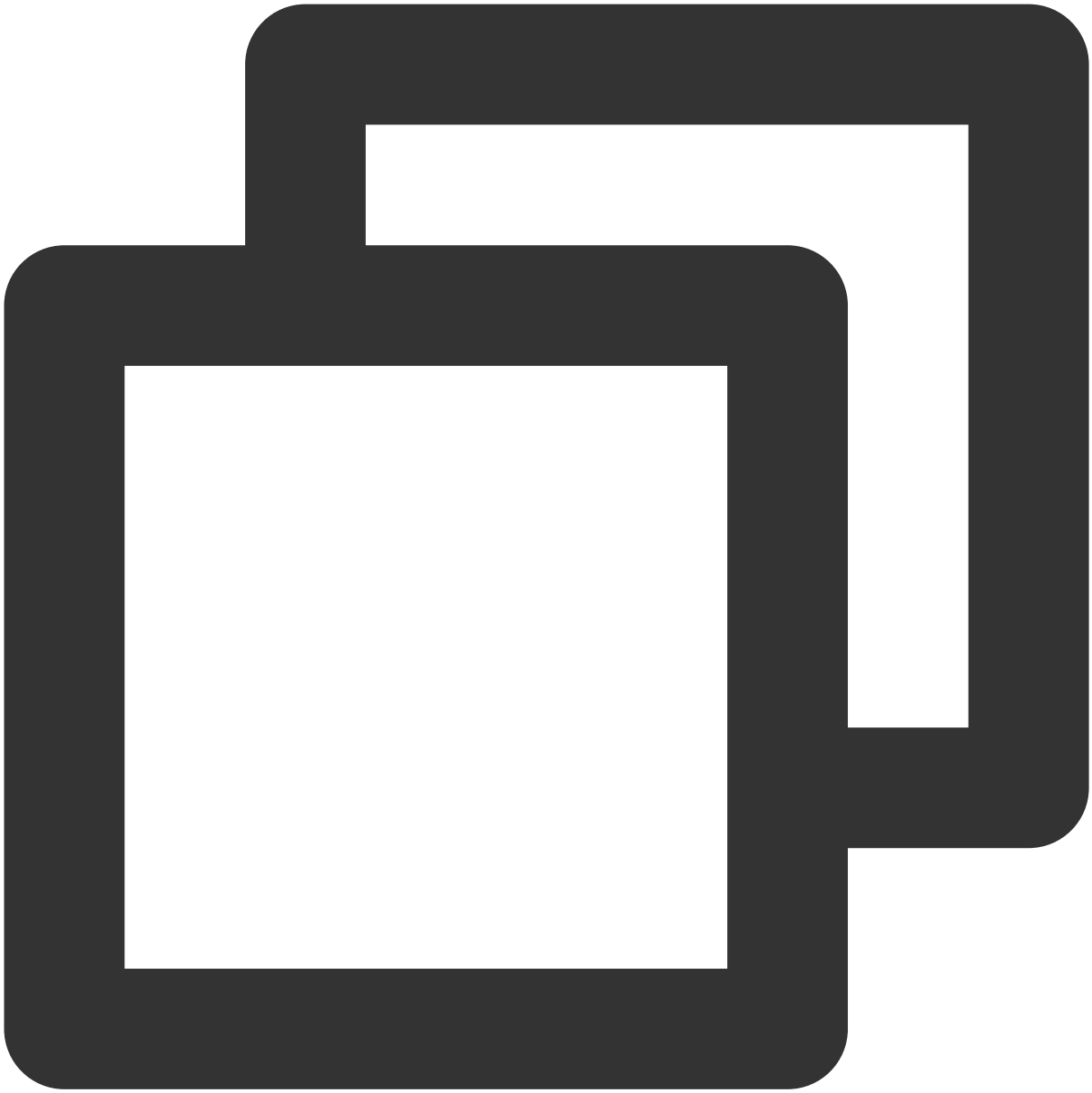
```
public func setCallingBell(filePath: String)
```

### enableMuteMode

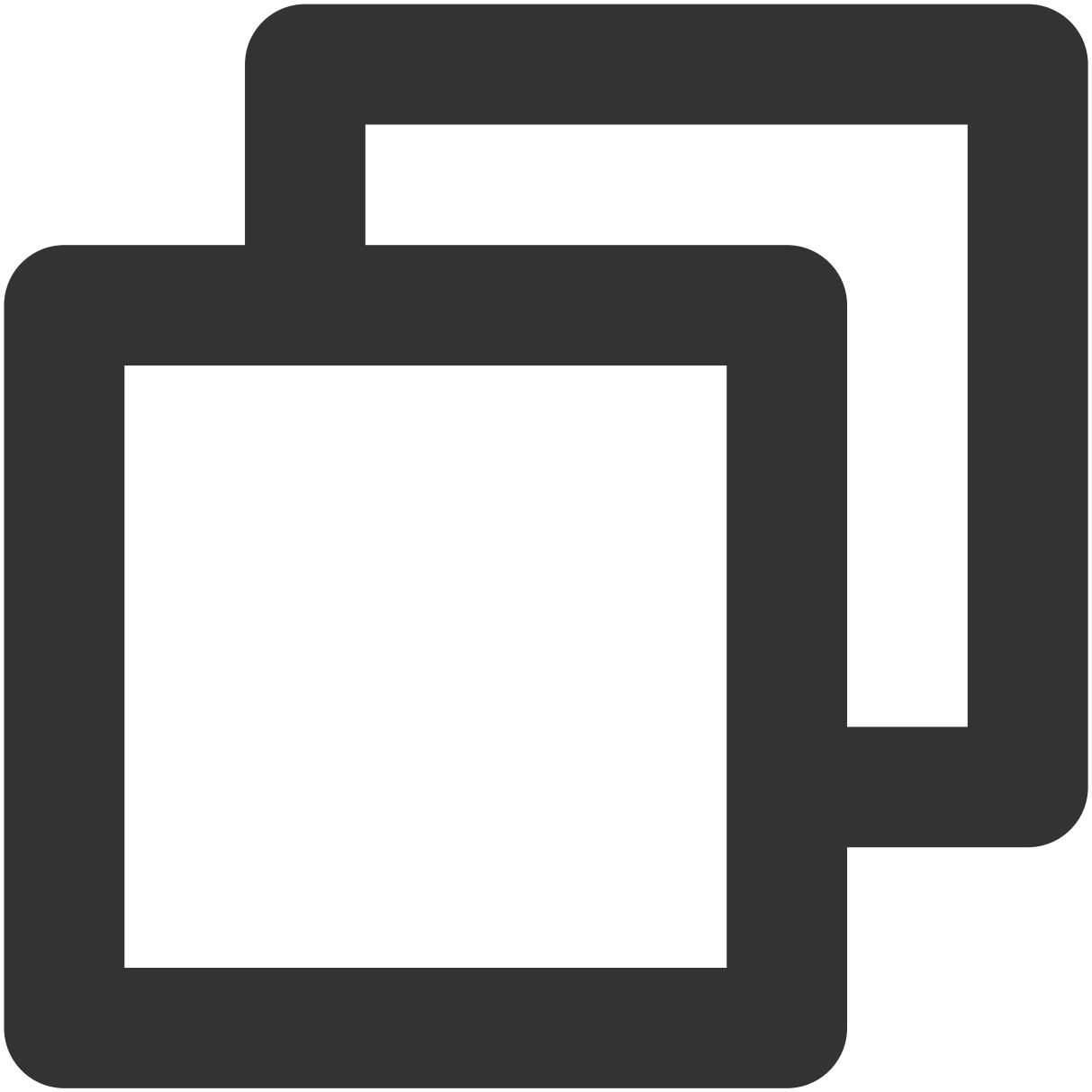
This API is used to set whether to turn on the mute mode.

Objective-C

Swift



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



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

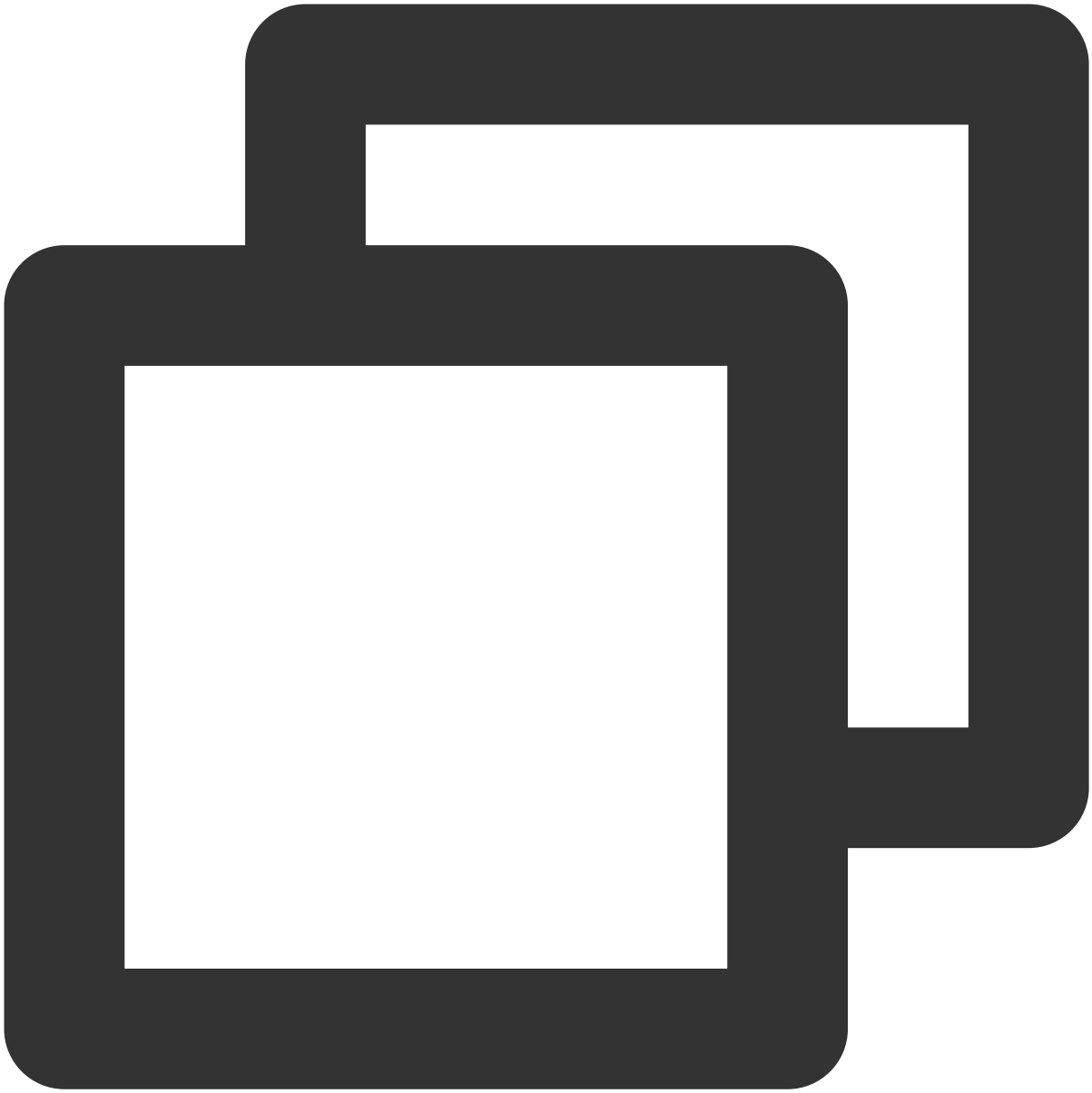
### enableFloatWindow

This API is used to set whether to enable floating windows.

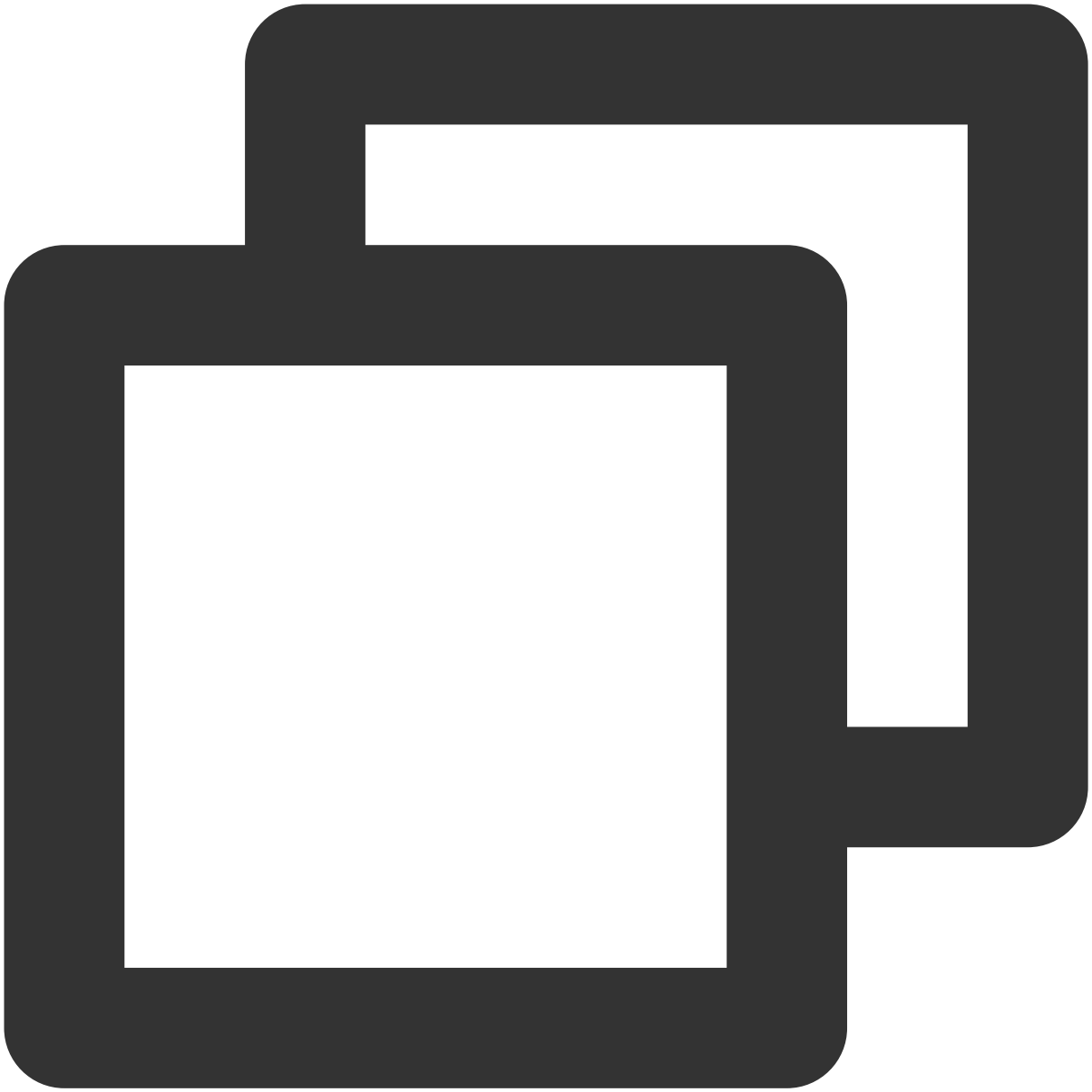
The default value is `false`, and the floating window button in the top left corner of the call view is hidden. If it is set to `true`, the button will become visible.

Objective-C

Swift



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



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

# TUICallEngine

Last updated：2024-03-26 14:39:11

## TUICallEngine APIs

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

## Overview

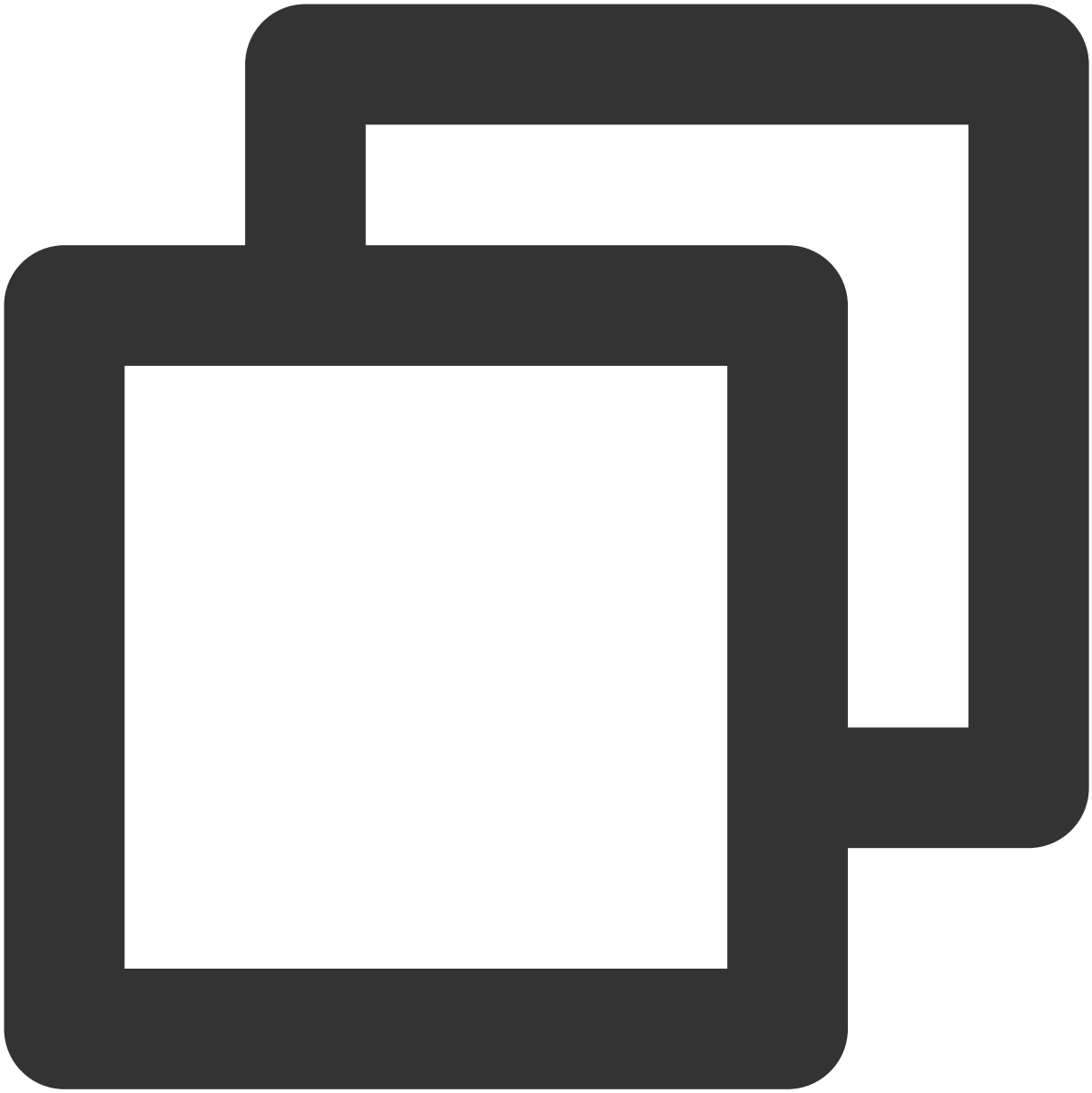
| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallEngine` instance (singleton mode). |
| destroyInstance | Terminates a `TUICallEngine` instance (singleton mode). |
| init | Authenticates the basic audio/video call capabilities. |
| addObserver | Registers an event listener. |
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Changes the call type, for example, from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |

| | |
|---|---|
| openCamera | Turns the camera on. |
| closeCamera | Turns the camera off. |
| switchCamera | Switches between the front and rear cameras. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver or speaker). |
| setSelfInfo | Sets the alias and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for `TUICallEngine` (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |
| setVideoEncoderParams | Set the encoding parameters of video encoder. |
| getTRTCCloudInstance | Advanced features. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

# Details

### createInstance

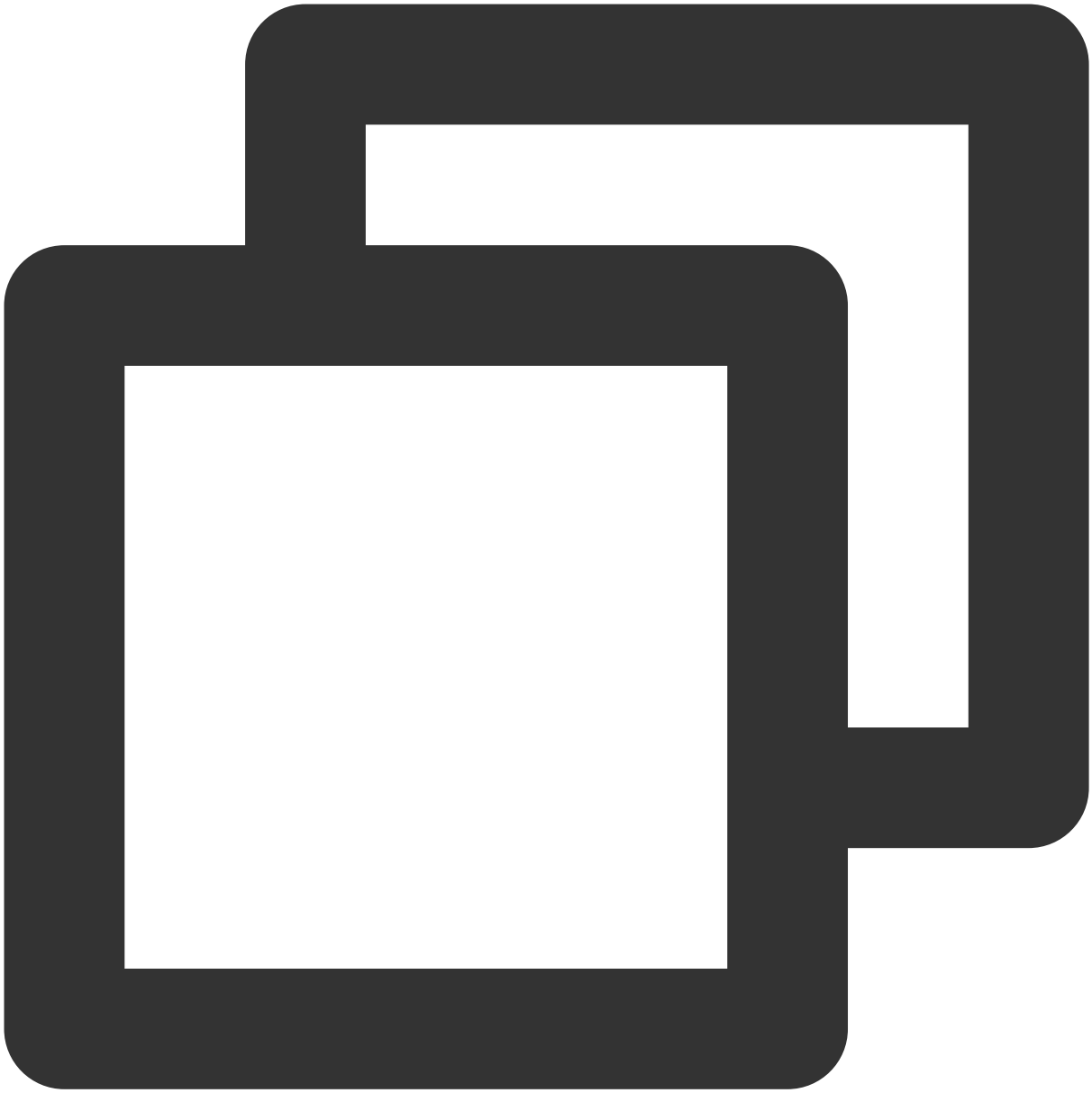
This API is used to create a `TUICallEngine` singleton.



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

### destroyInstance

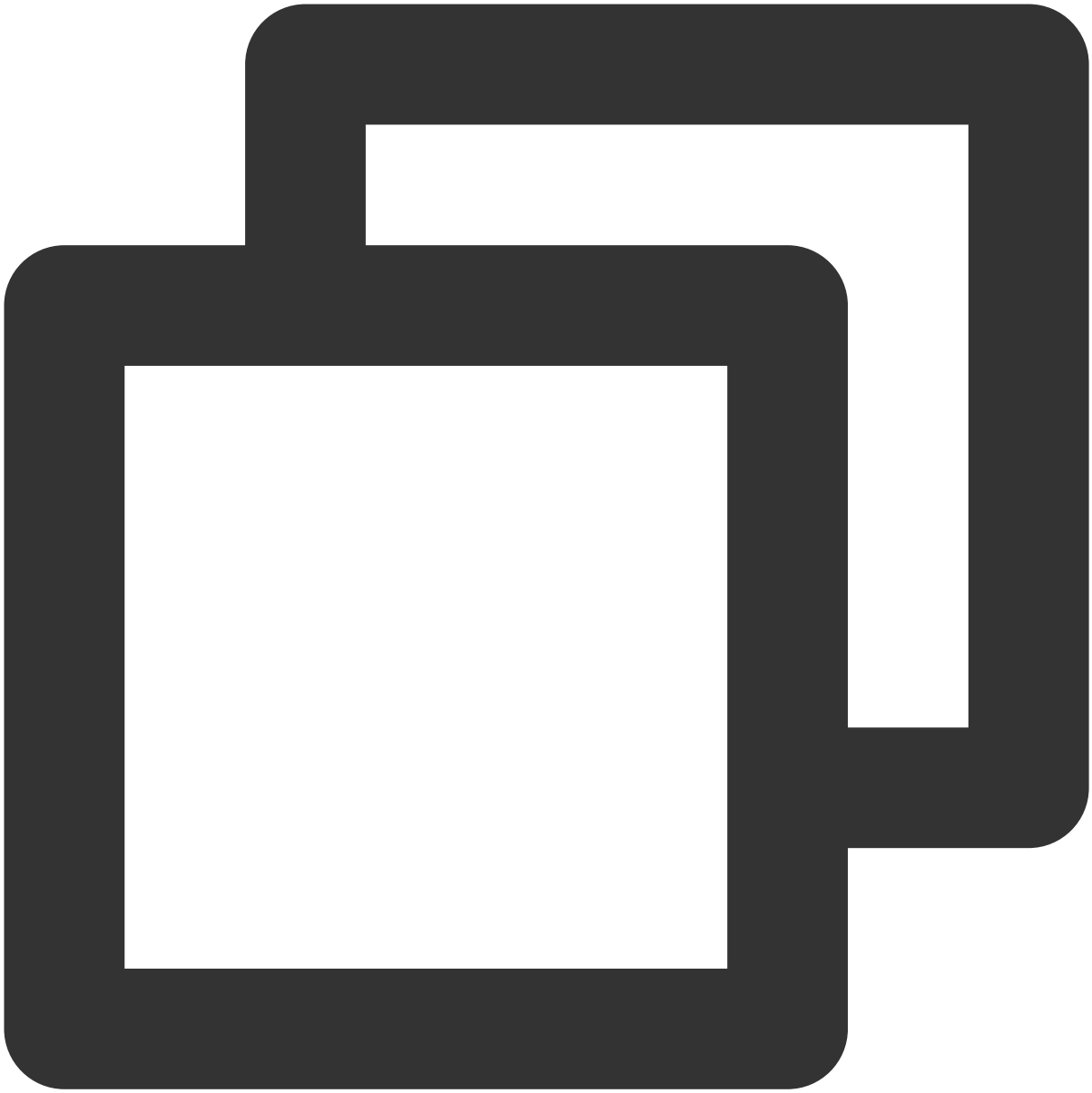
This API is used to terminate a `TUICallEngine` singleton.



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

### Init

This API is used to initialize `TUICallEngine`. Call it to authenticate the call service and perform other required actions before you call other APIs.



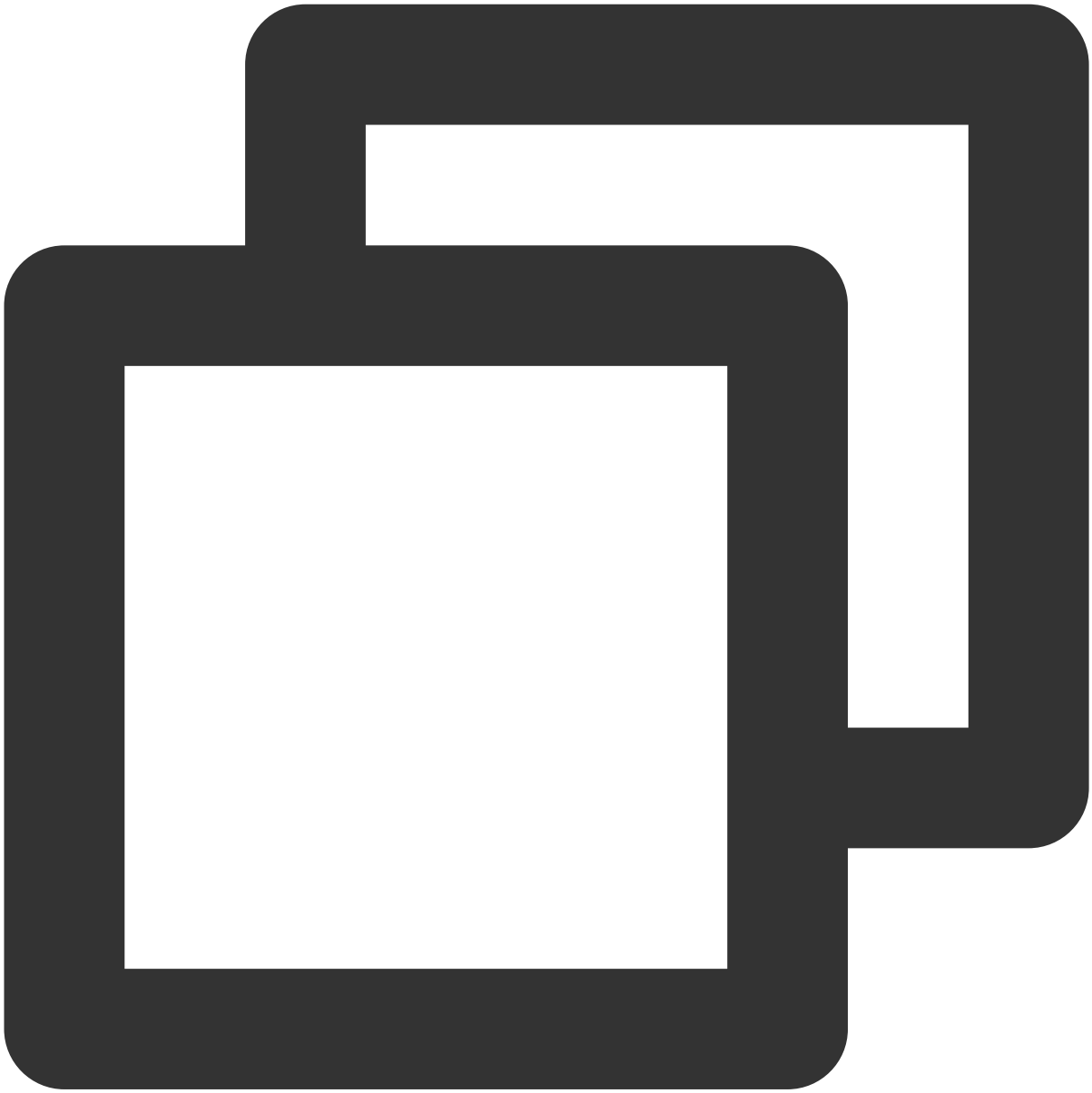
```
- (void)init:(NSString *)sdkAppID userId:(NSString *)userId userSig:(NSString *)use
```

| Parameter | Type | Description |
| --- | --- | --- |
| sdkAppID | NSString | You can view `SDKAppID` in Application Management > **Application Info** of the TRTC console. |
| userId | NSString | The ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_). |
| userSig | NSString | Tencent Cloud's proprietary security signature. For how to calculate |

| | | |
|---|---|---|
| | | and use it, see UserSig. |

### addObserver

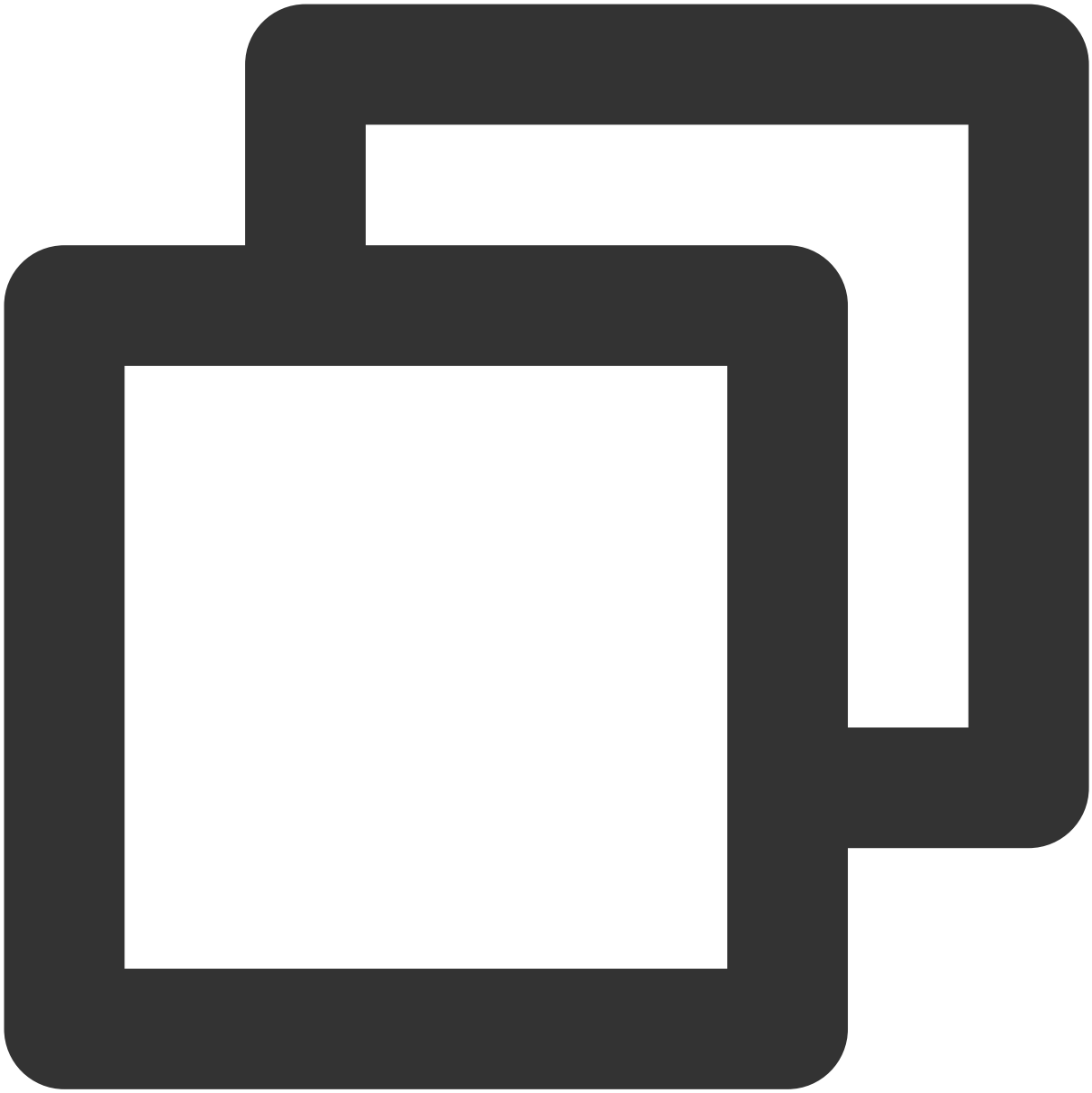
This API is used to register an event listener to listen for `TUICallObserver` events.



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

### removeObserver

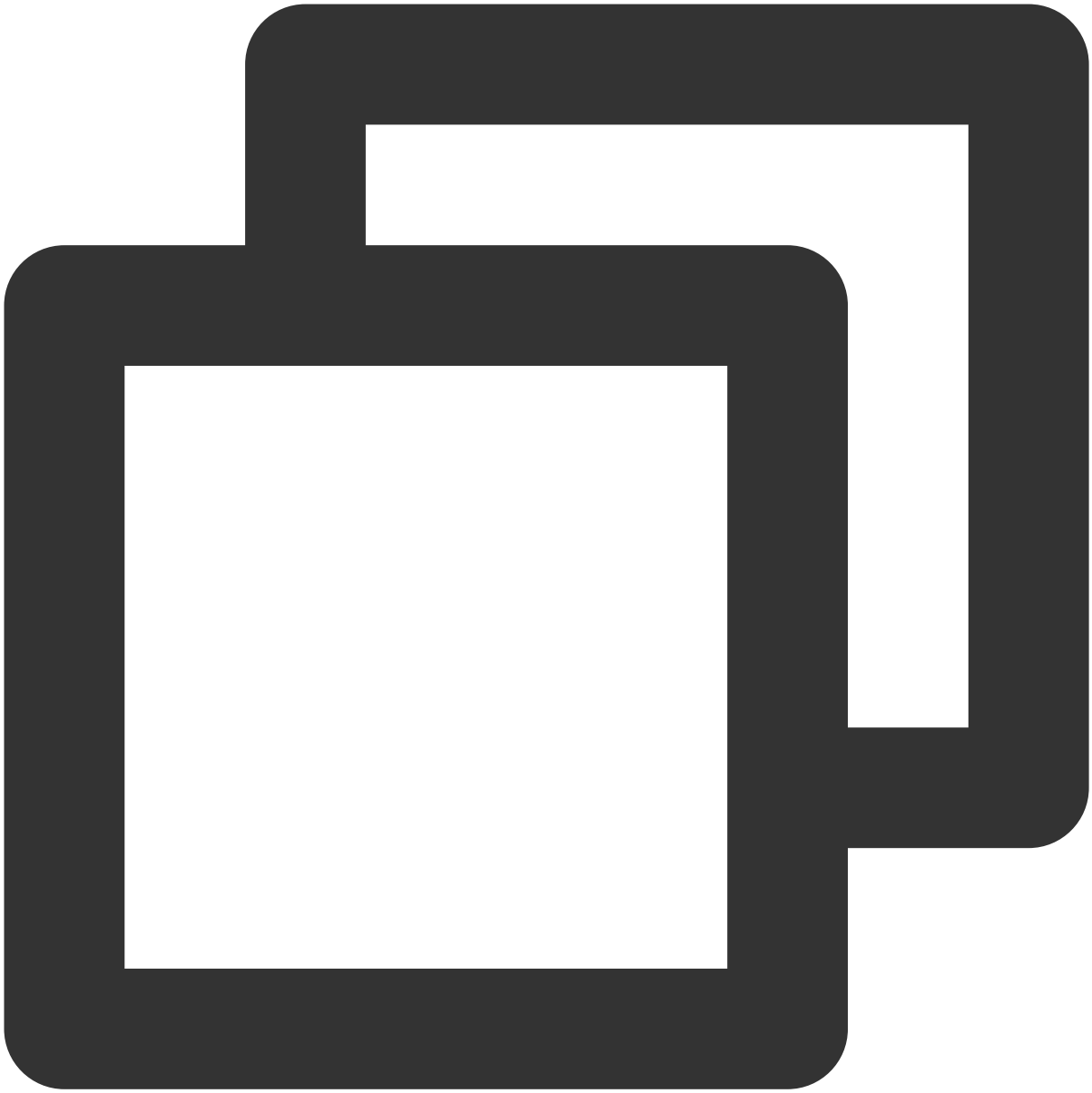
This API is used to unregister an event listener.



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

### call

This API is used to make a (one-to-one) call.



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

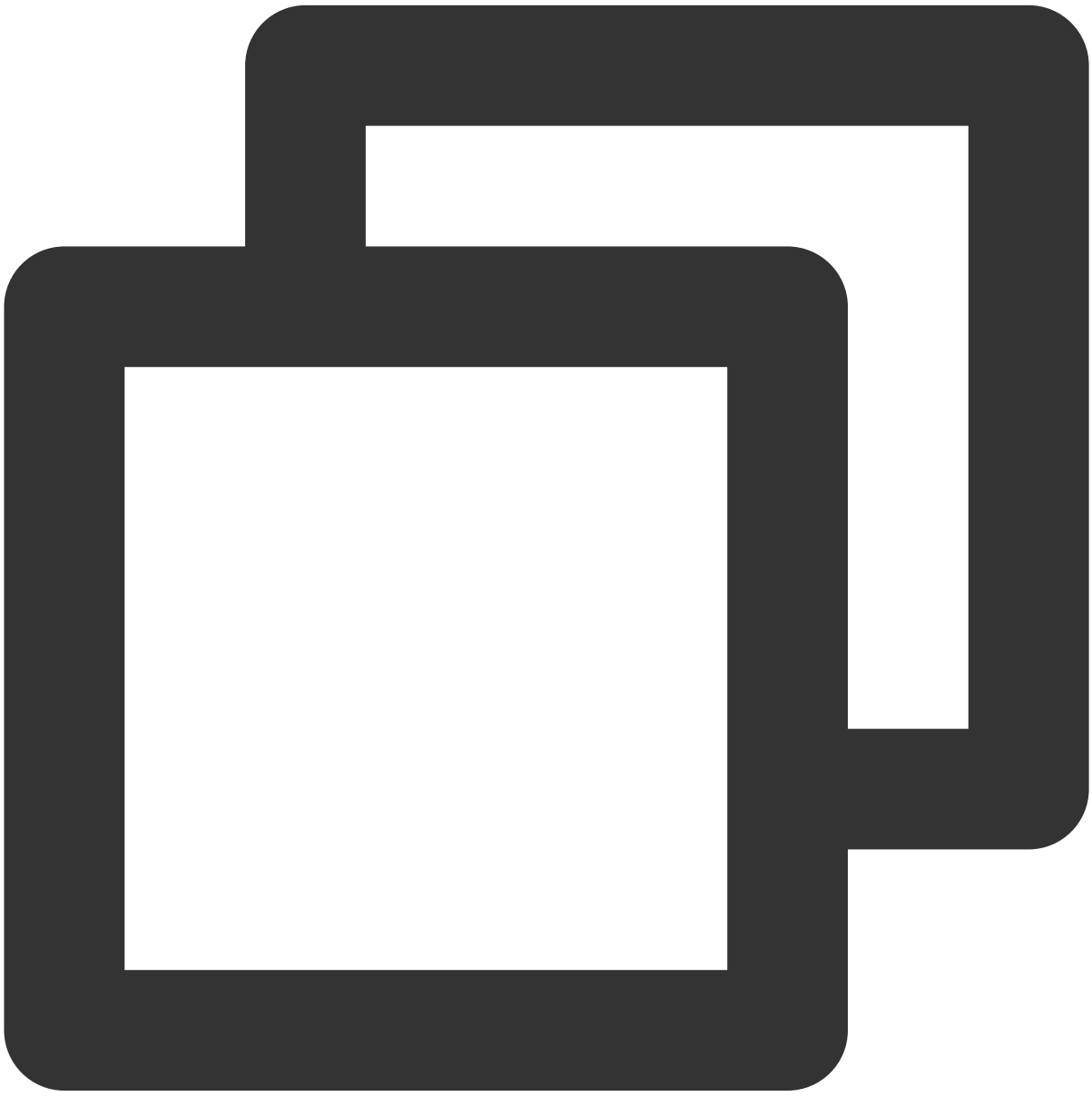
| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | An additional parameter, such as roomID, call timeout, offline push info, etc |

### groupCall

This API is used to make a group call.

**Notice：**

Before making a group call, you need to create an IM group first.



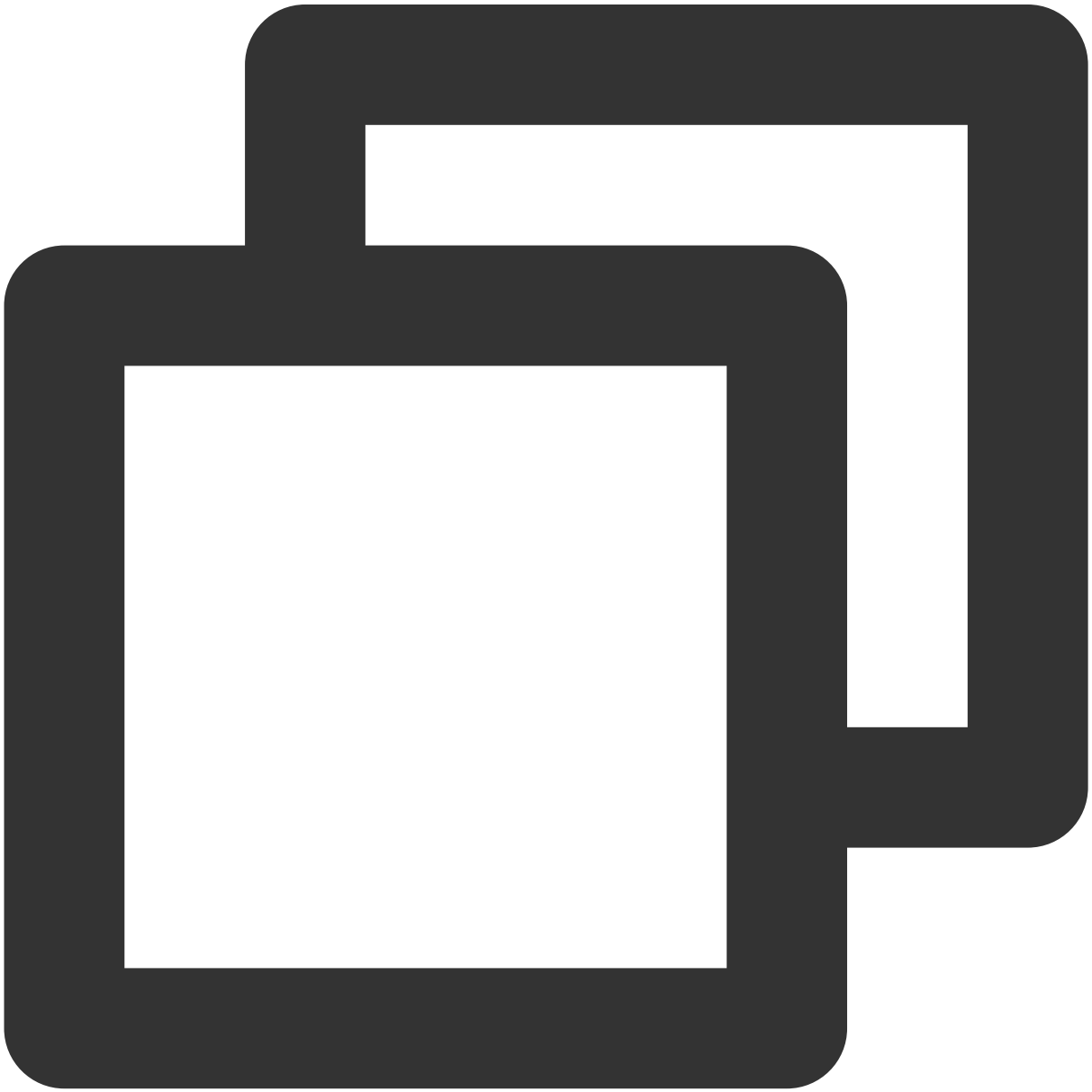
```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray <NSString *> *)userIdList
```

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | NSString | The group ID. |

| | | |
|---|---|---|
| userIdList | NSArray | The target user IDs. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | An additional parameter. such as roomID, call timeout, offline push info, etc |

### accept

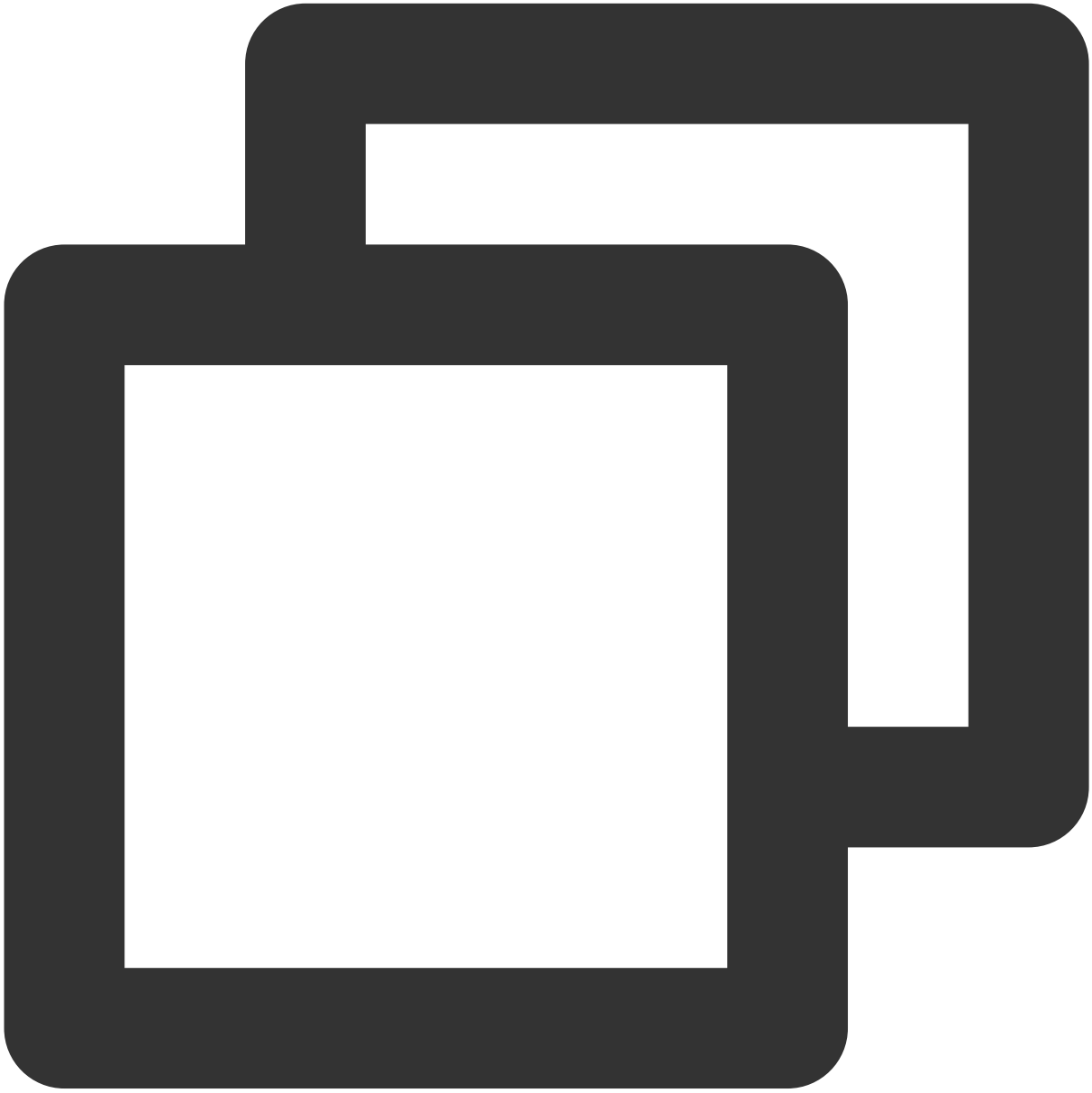
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



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

### reject

This API is used to reject a call. After receiving the `onCallReceived()` callback, you can call this API to reject the call.

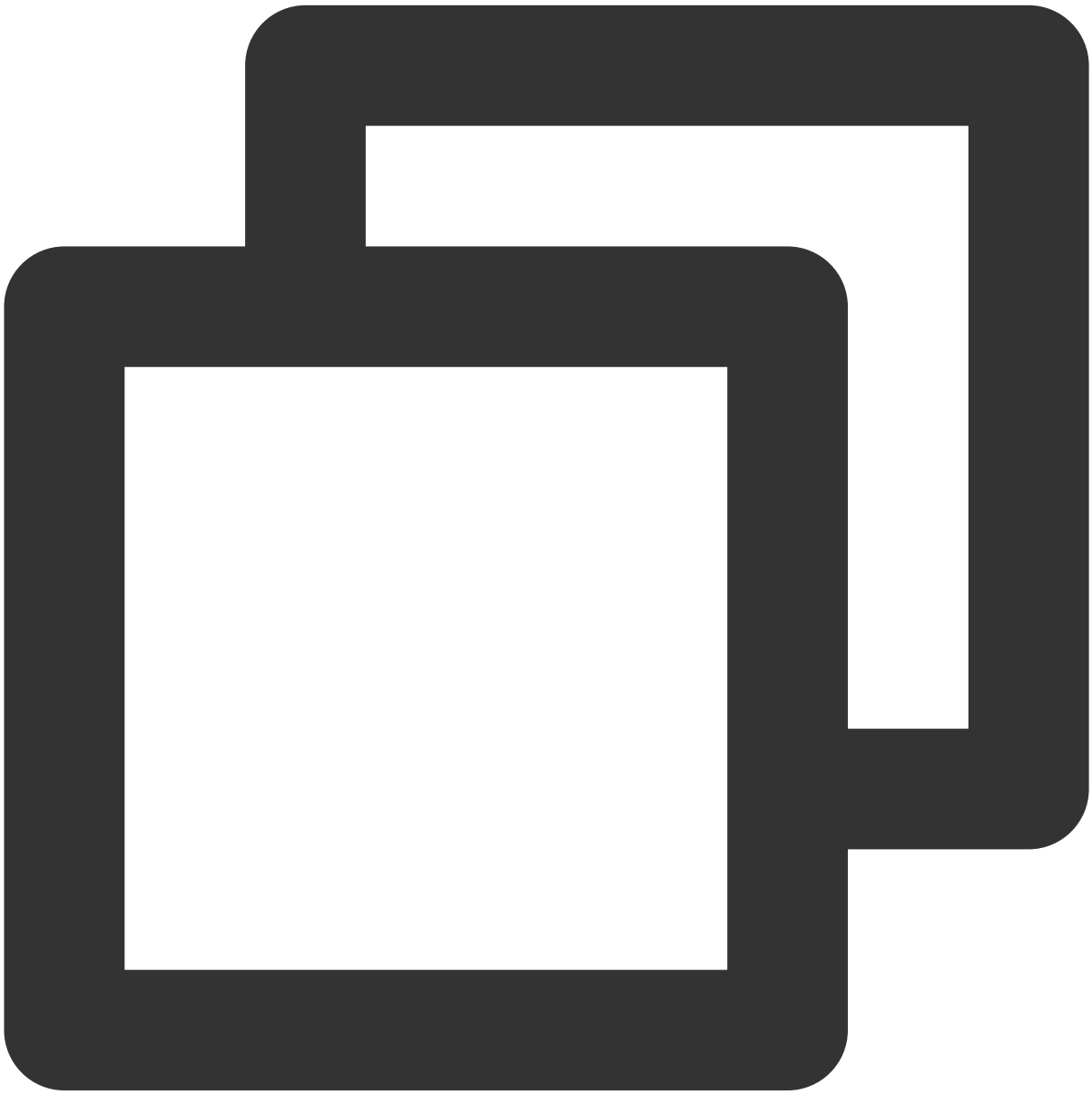


```
- (void)reject:(TUICallSucc)succ fail:(TUICallFail)fail;
```

### ignore

This API is used to ignore a call. After receiving the `onCallReceived()` , you can call this API to ignore the call. The caller will receive the `onUserLineBusy` callback.

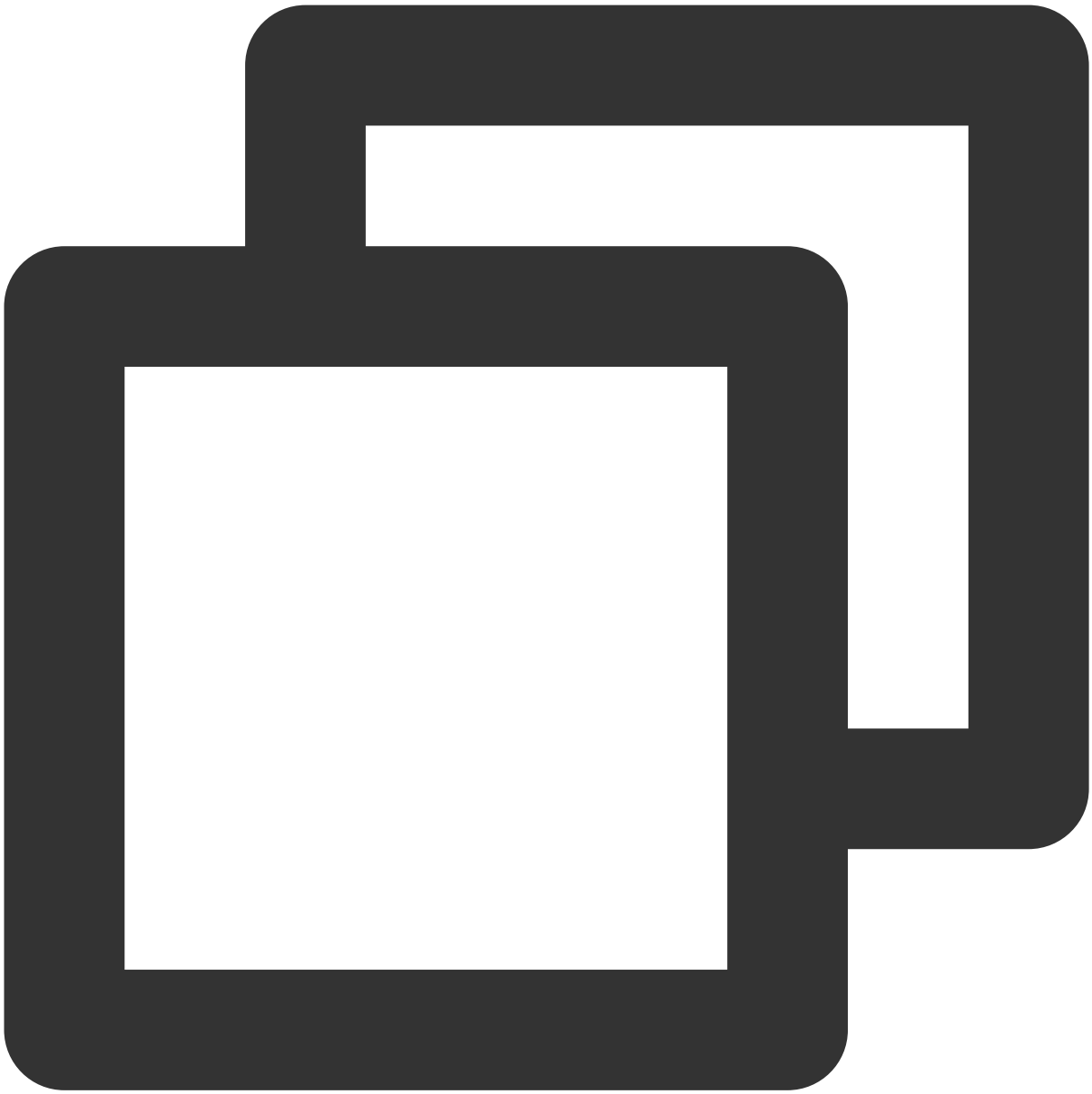
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



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

### hangup

This API is used to end a call.

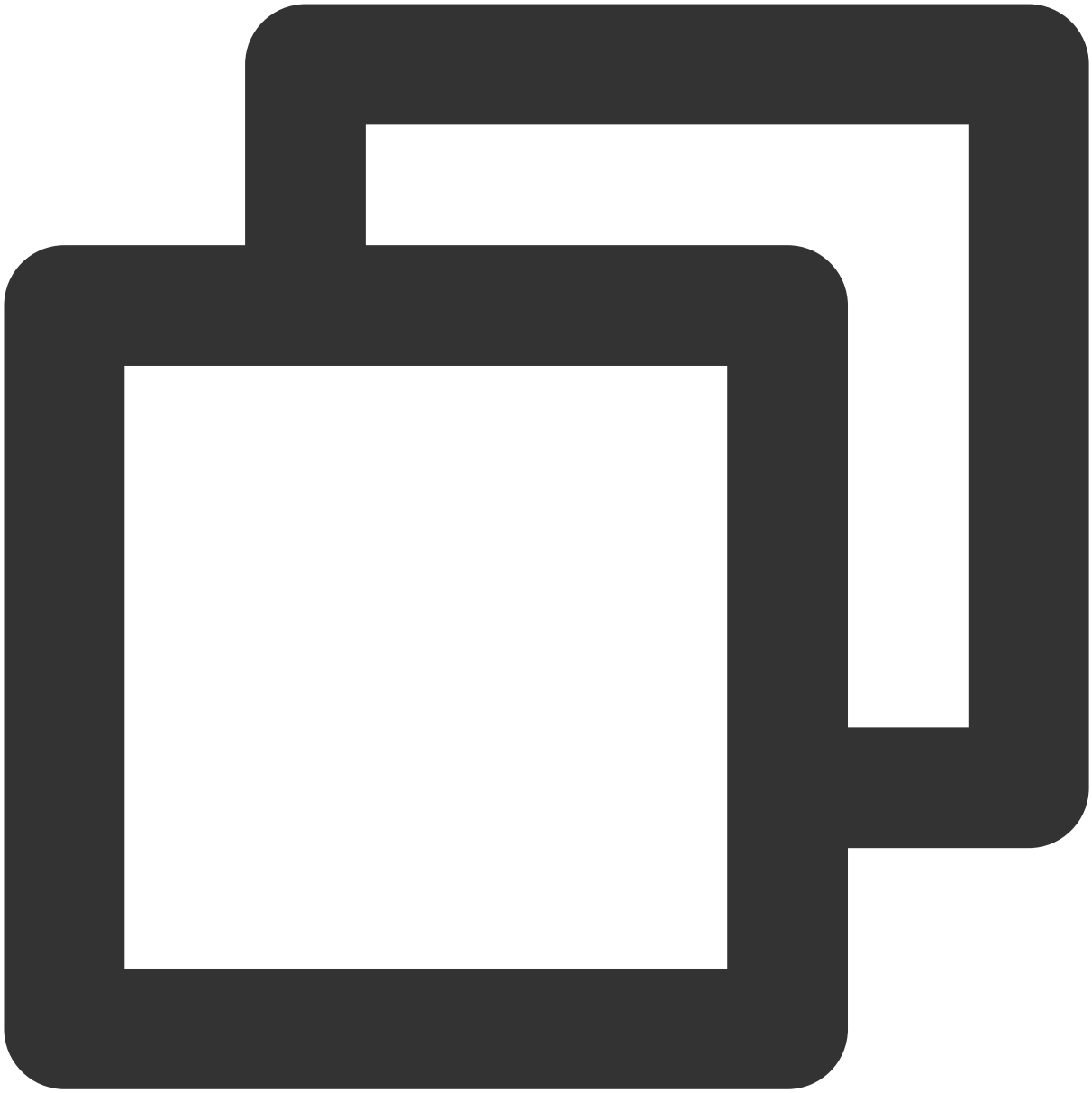


```
- (void)hangup:(TUICallSucc)succ fail:(TUICallFail)fail;
```

### inviteUser

This API is used to invite users to the current group call.

This API is called by a participant of a group call to invite new users.



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

| Parameter | Type | Description |
| --- | --- | --- |
| userIdList | NSArray | The target user IDs. |
| params | TUICallParams | An additional parameter. such as roomID, call timeout, offline push info, etc |

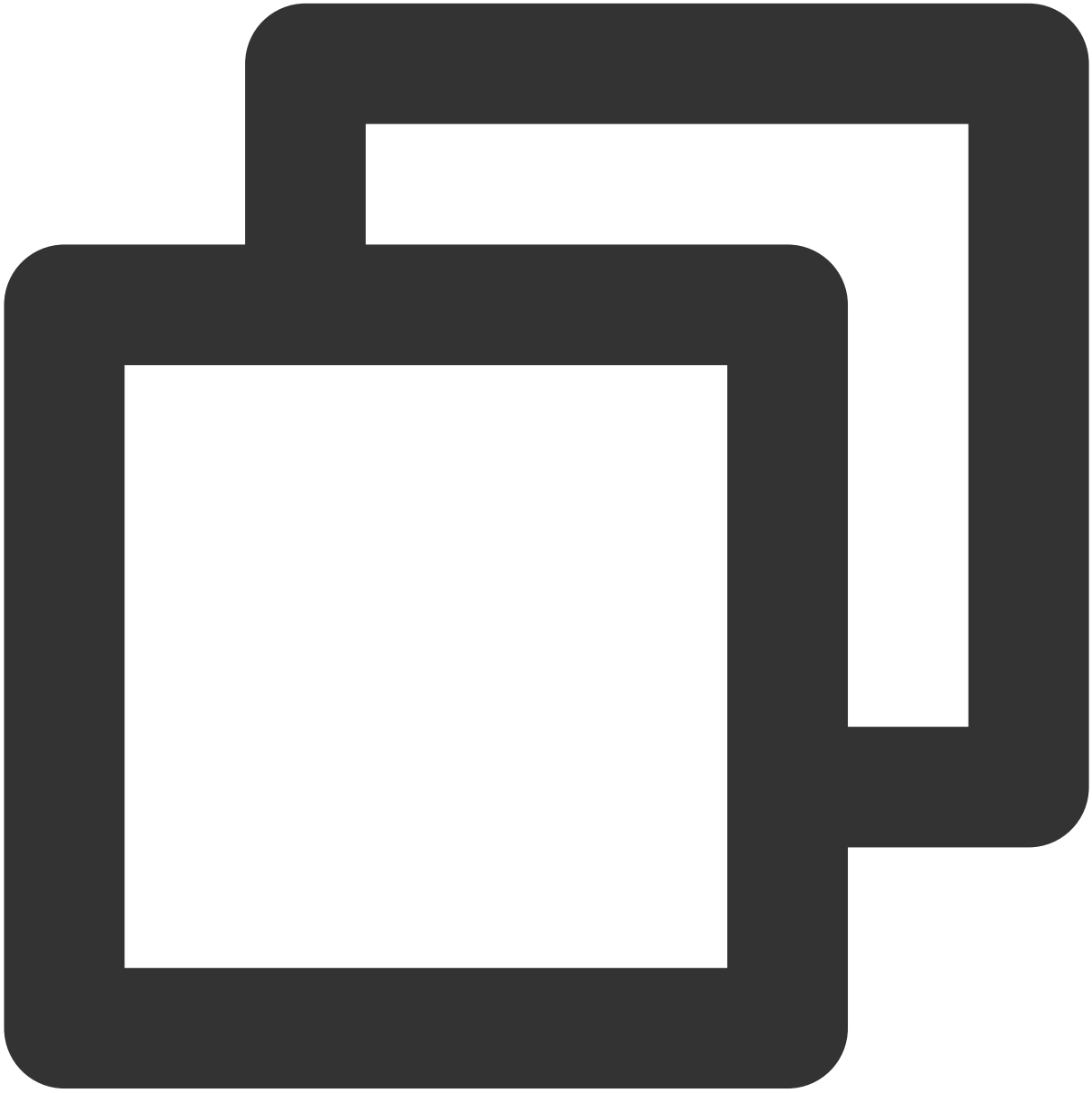
**Notice：**

In this case, the custom RoomId is invalid. The SDK will invite others to join the room where the inviter is currently located.

### joinInGroupCall

This API is used to join a group call.

This API is called by a group member to join the group's call.



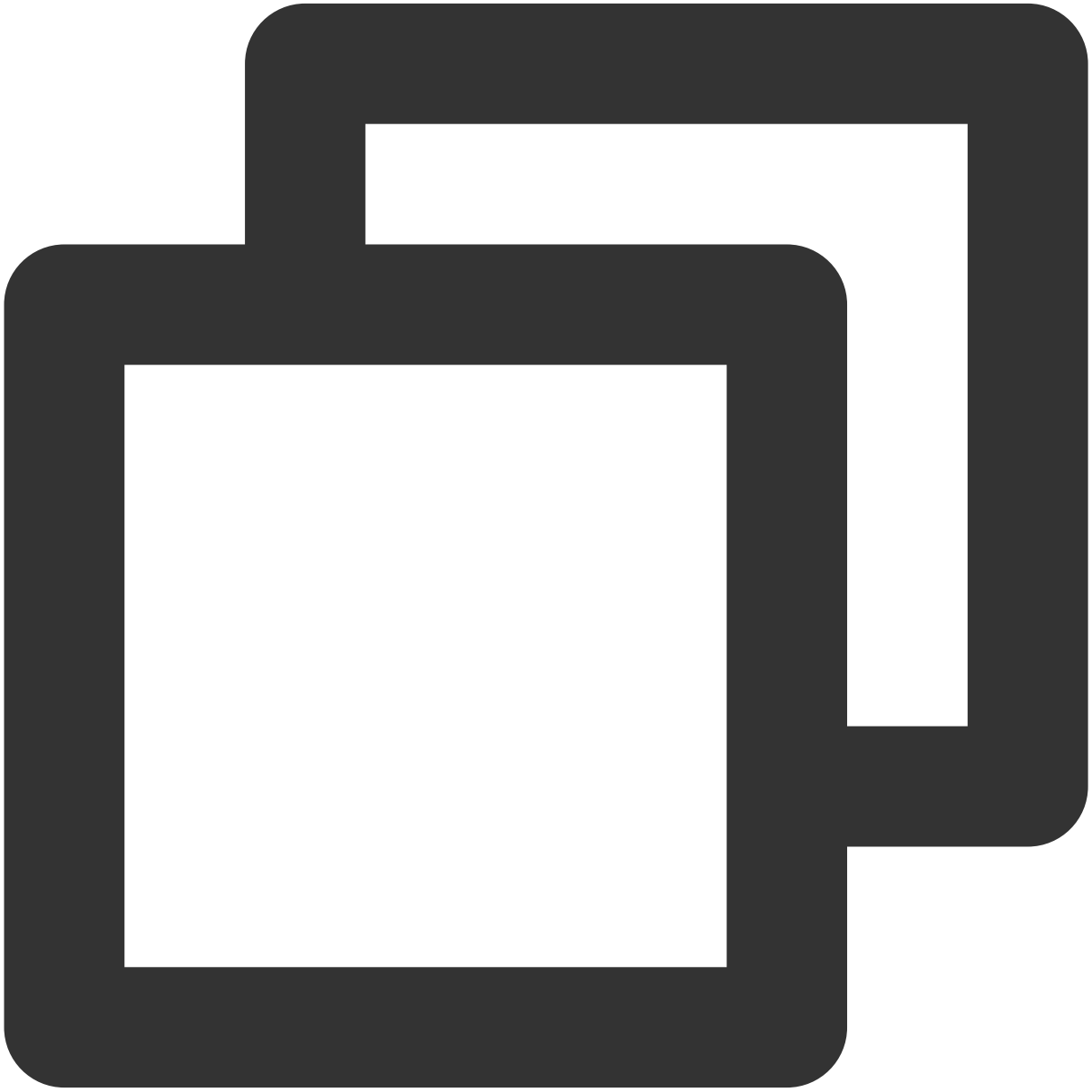
```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```

| Parameter | Type | Description |
| --- | --- | --- |

| | | |
|---|---|---|
| roomId | TUIRoomId | The room ID. |
| groupId | NSString | The group ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### switchCallMediaType

This API is used to change the call type.

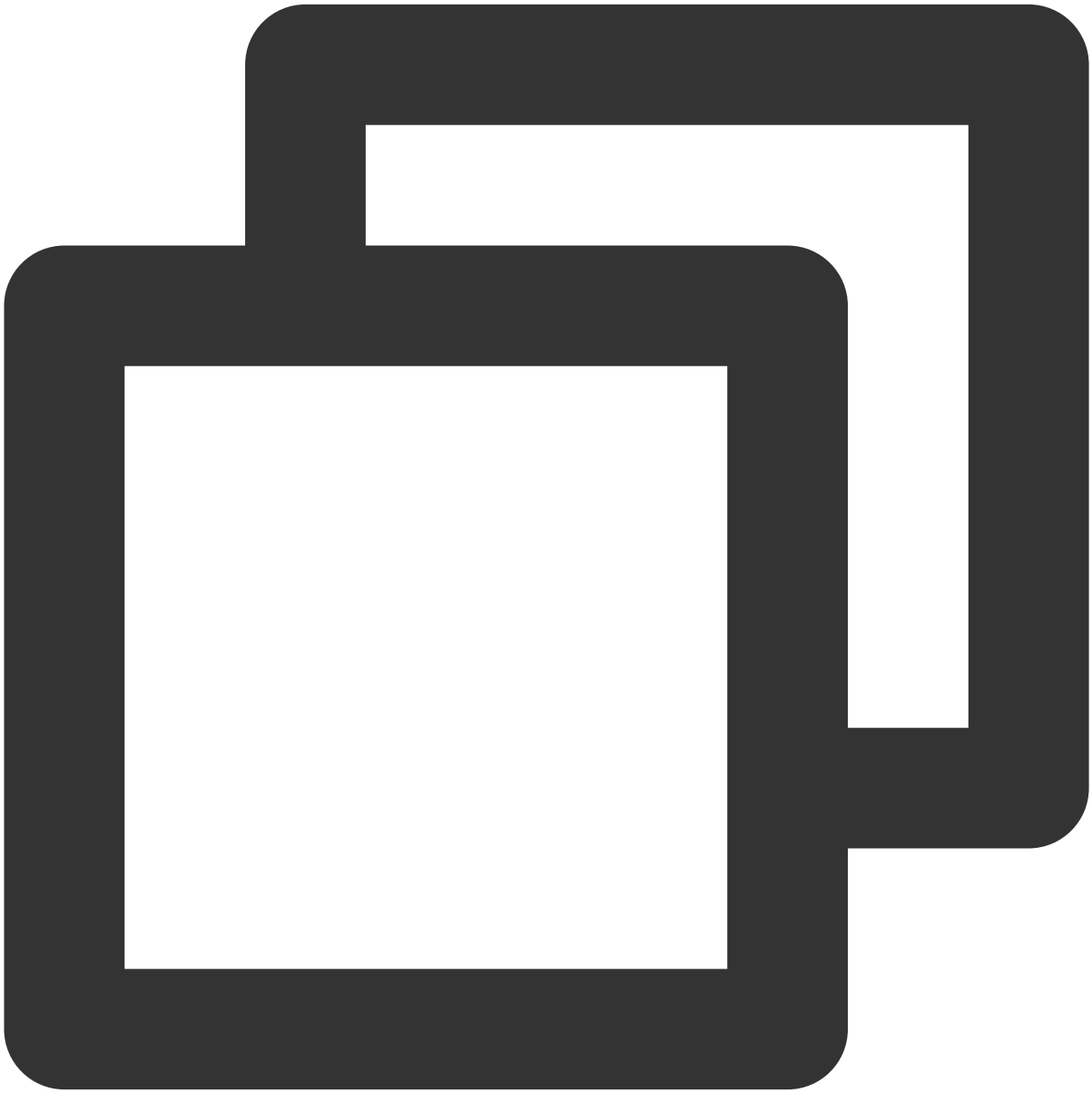


```
- (void)switchCallMediaType:(TUICallMediaType)newType;
```

| Parameter | Type | Description |
| --- | --- | --- |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### startRemoteView

This API is used to set the view object to display a remote video.



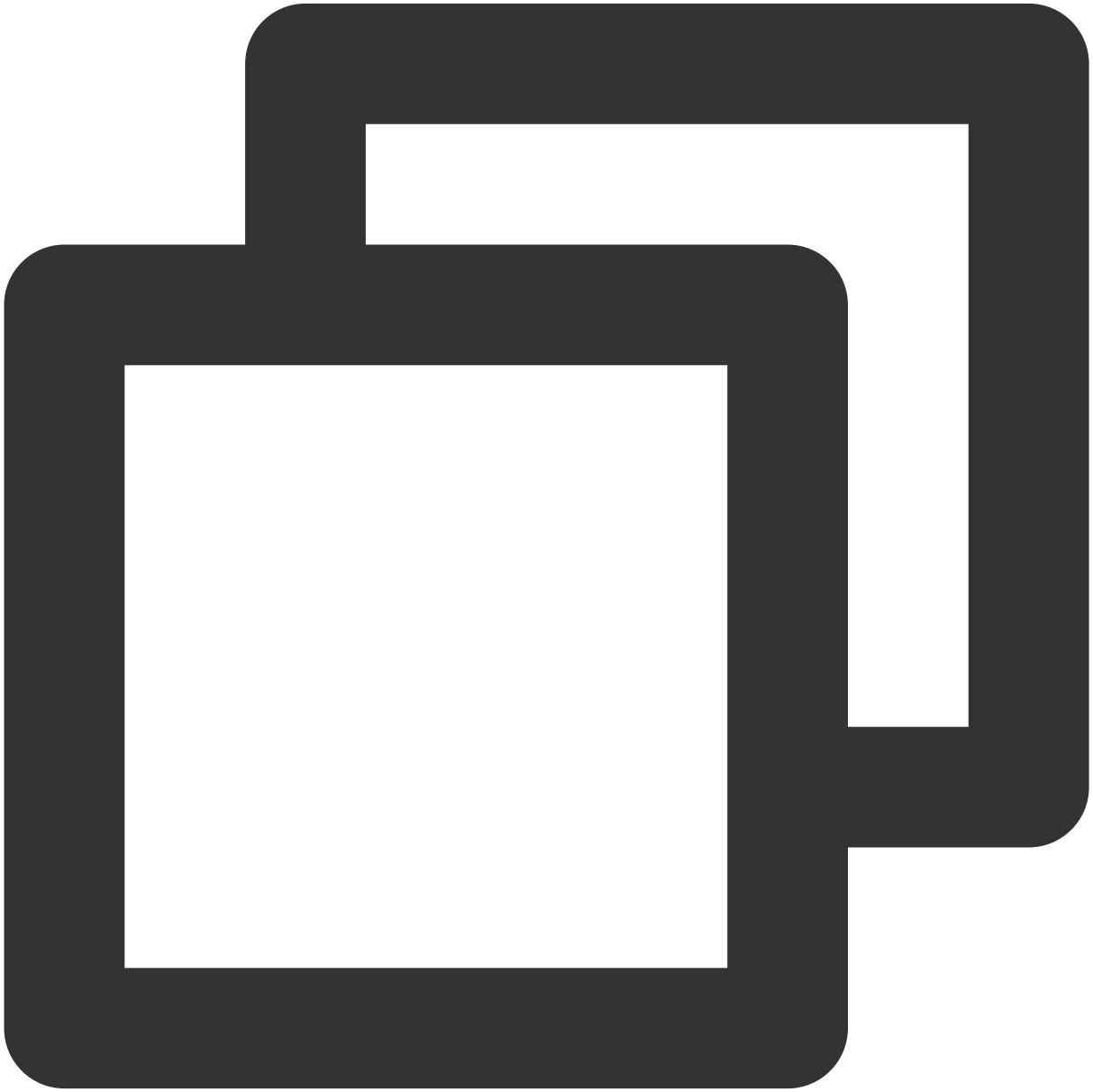
```
- (void)startRemoteView:(NSString *)userId videoView:(TUIVideoView *)videoView onPl
```

| | | |
| --- | --- | --- |

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |
| videoView | TUIVideoView | The view to be rendered. |

### stopRemoteView

This API is used to unsubscribe from the video stream of a remote user.

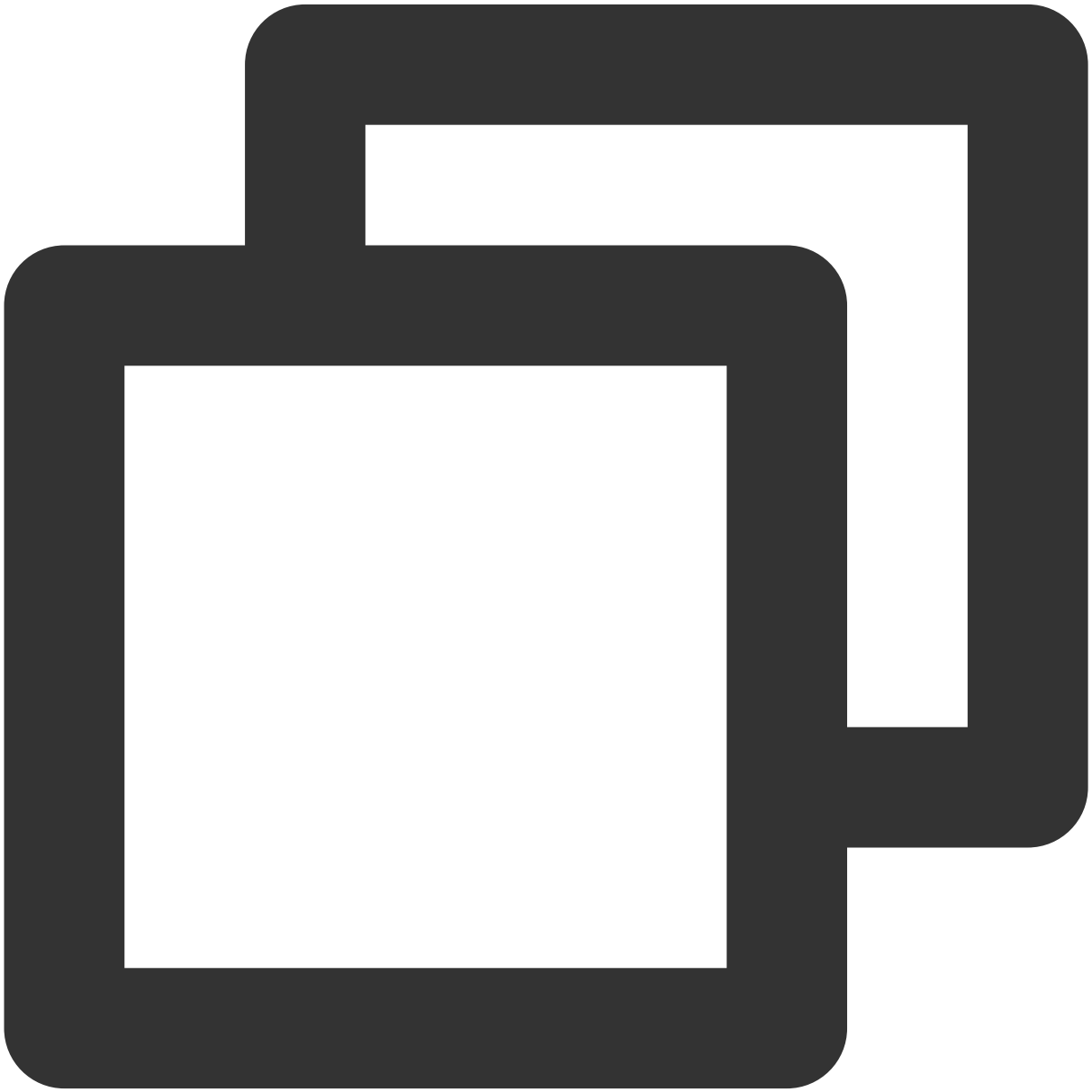


```
- (void)stopRemoteView:(NSString *)userId;
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |

### openCamera

This API is used to turn the camera on.



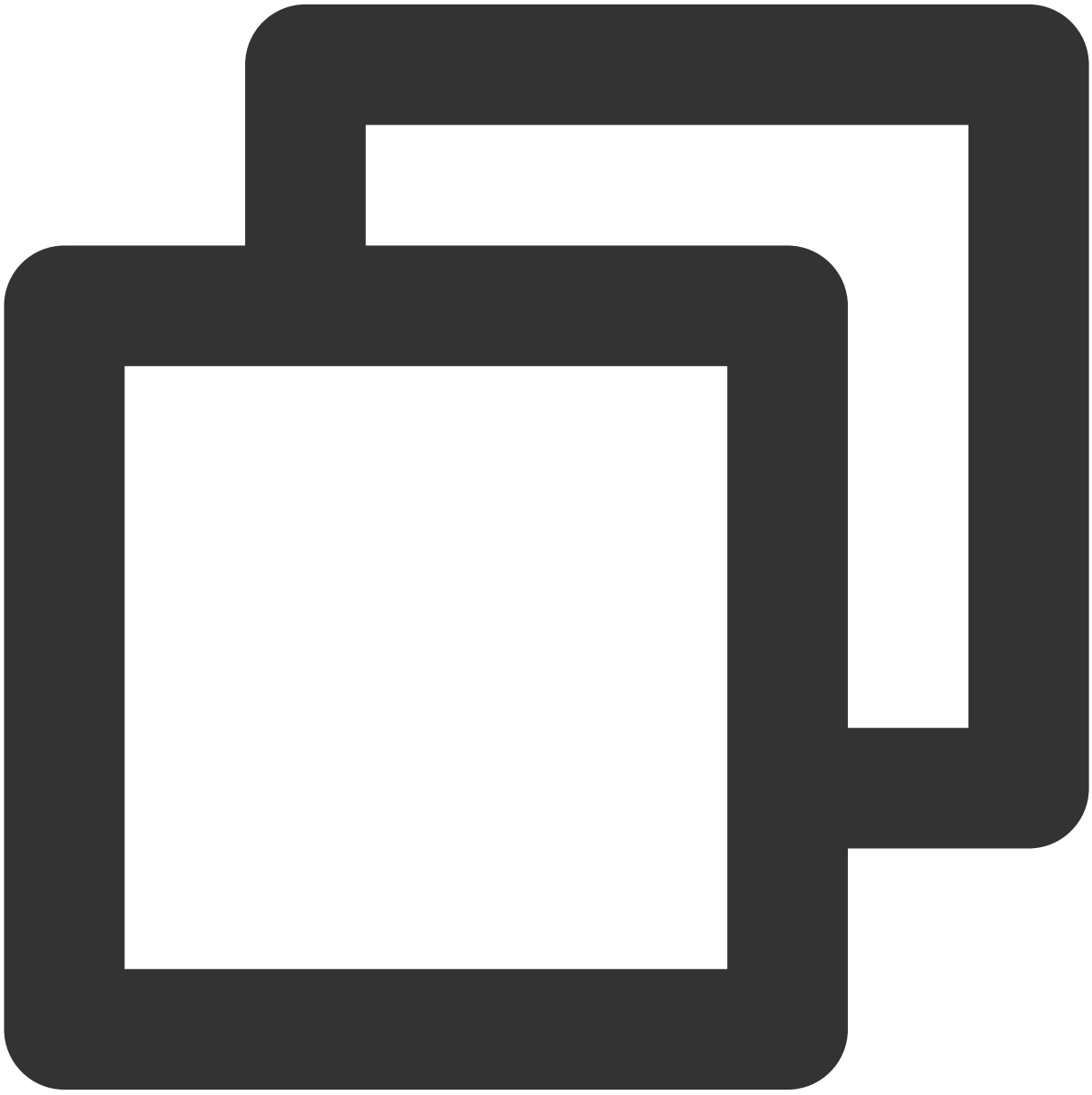
```
- (void)openCamera:(TUICamera)camera videoView:(TUIVideoView *)videoView succ:(TUIC
```

| Parameter | Type | Description |
| --- | --- | --- |

| | | |
|---|---|---|
| camera | TUICamera | The front or rear camera. |
| videoView | TUIVideoView | The view to be rendered. |

### closeCamera

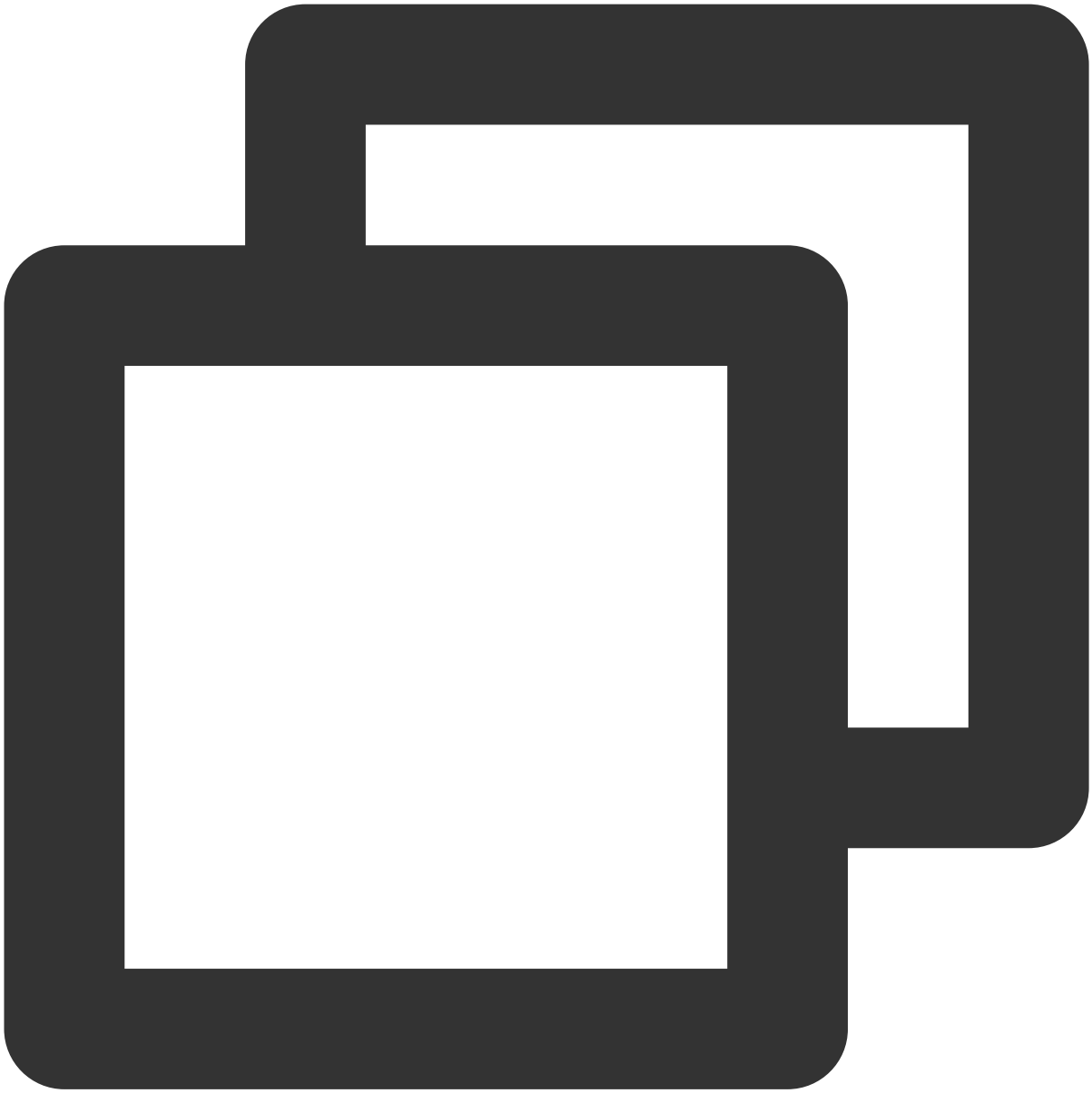
This API is used to turn the camera off.



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

### switchCamera

This API is used to switch between the front and rear cameras.

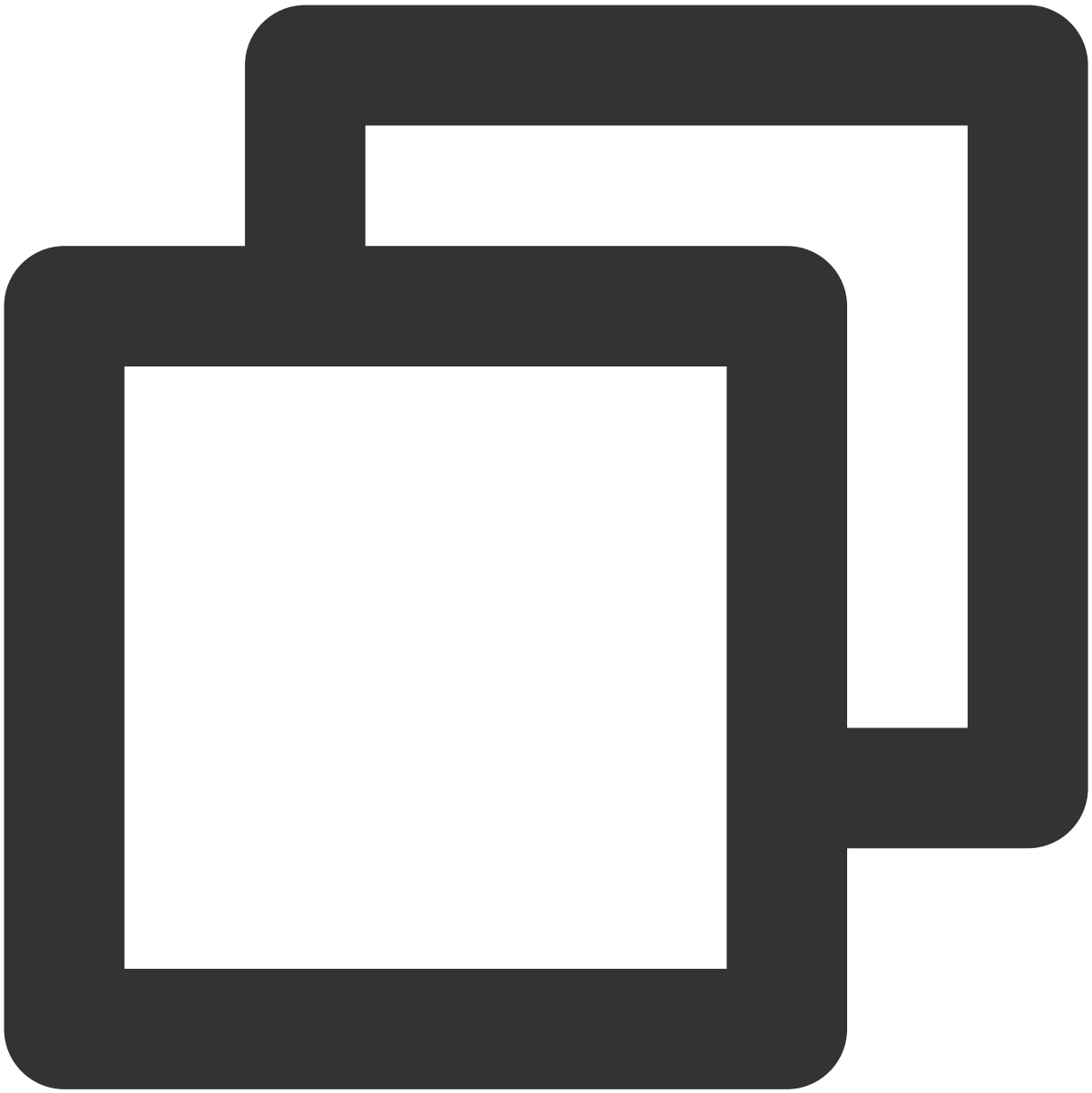


```
- (void)switchCamera:(TUICamera)camera;
```

| Parameter | Type | Description |
| --- | --- | --- |
| camera | TUICamera | The front or rear camera. |

### openMicrophone

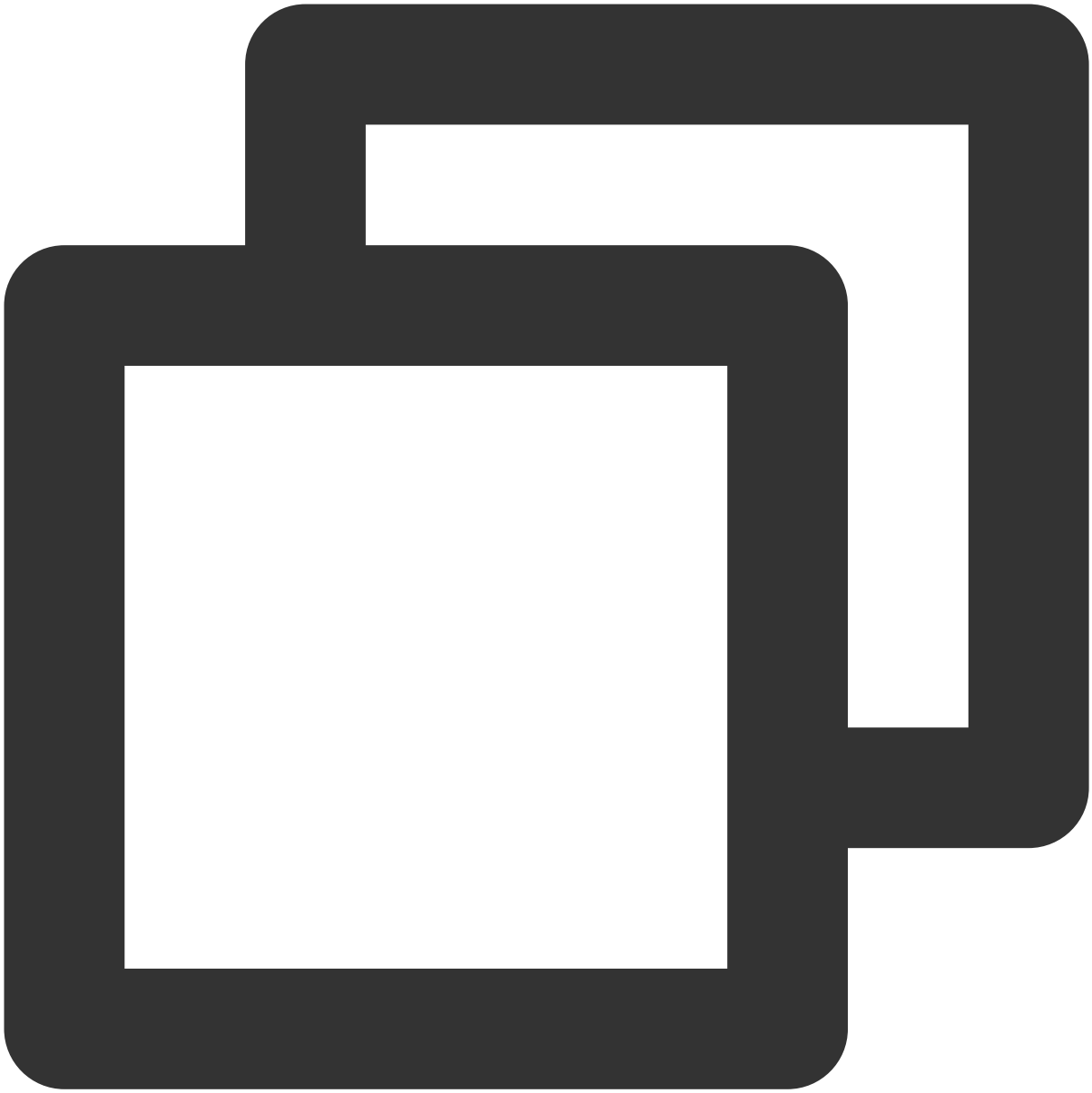
This API is used to turn the mic on.



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

### closeMicrophone

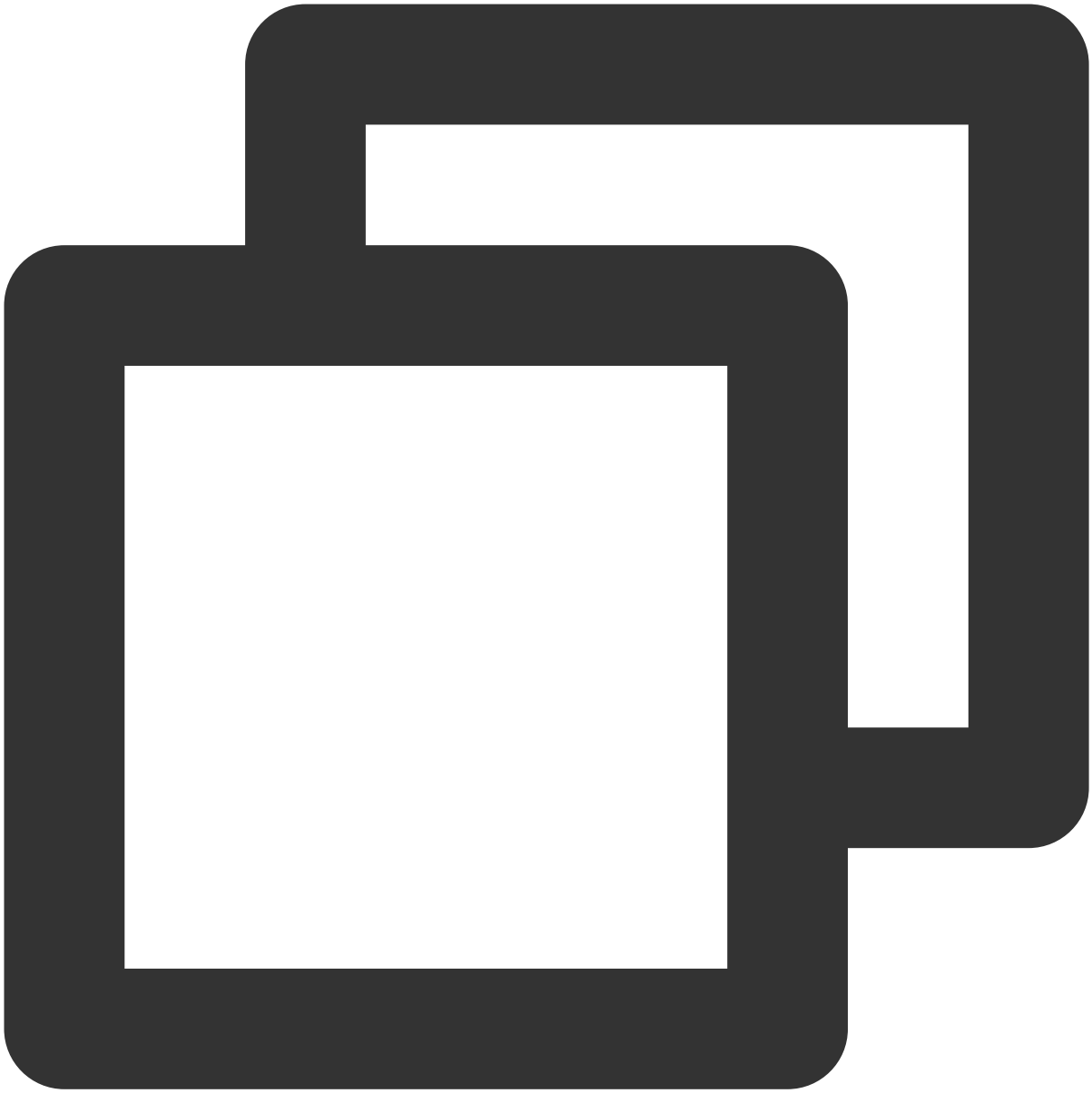
This API is used to turn the mic off.



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

### selectAudioPlaybackDevice

This API is used to select the audio playback device (receiver or speaker). In call scenarios, you can use this API to turn on/off hands-free mode.

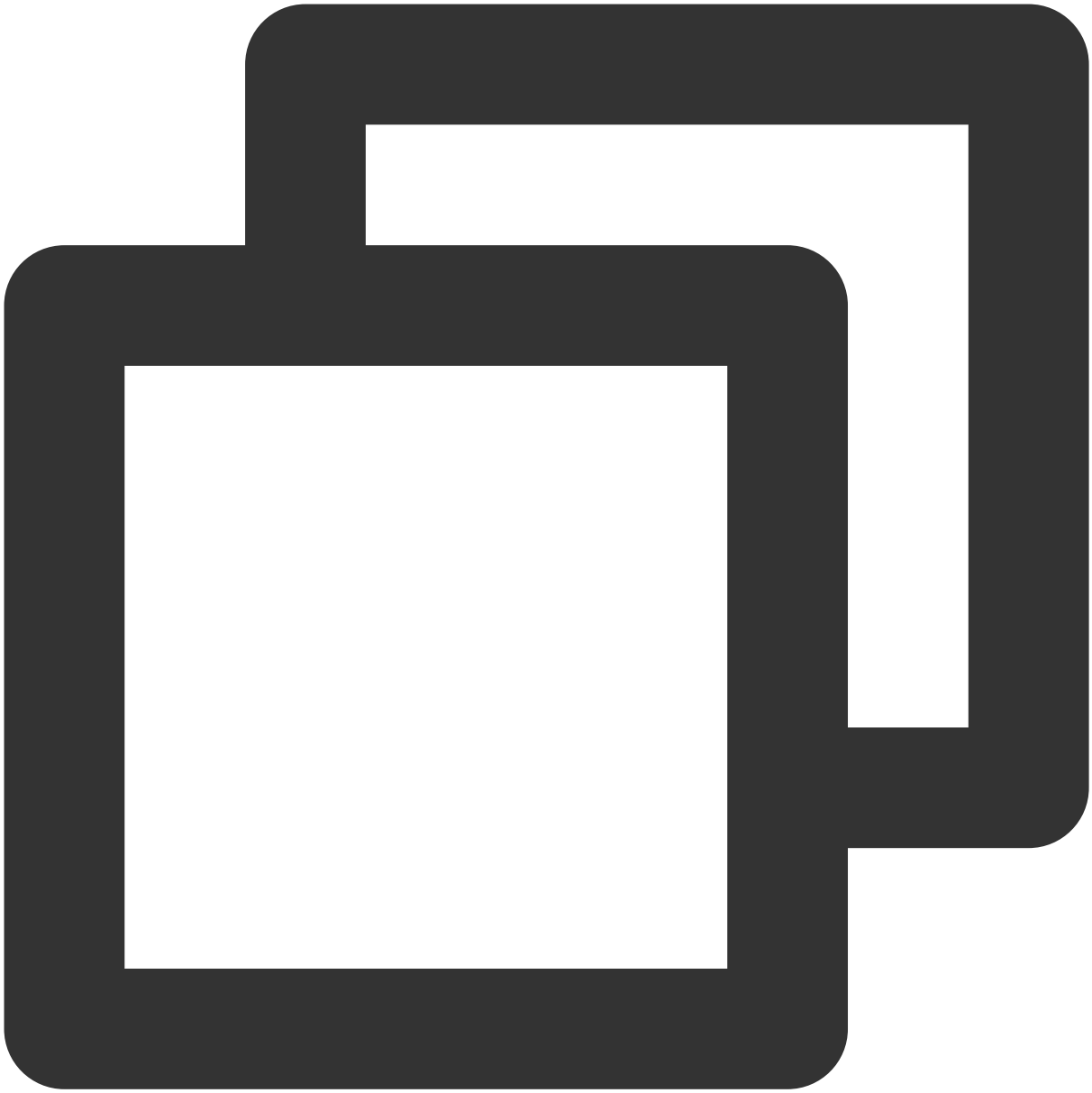


```
- (void)selectAudioPlaybackDevice:(TUIAudioPlaybackDevice)device;
```

| Parameter | Type | Description |
| --- | --- | --- |
| device | TUIAudioPlaybackDevice | The speaker or receiver. |

### setSelfInfo

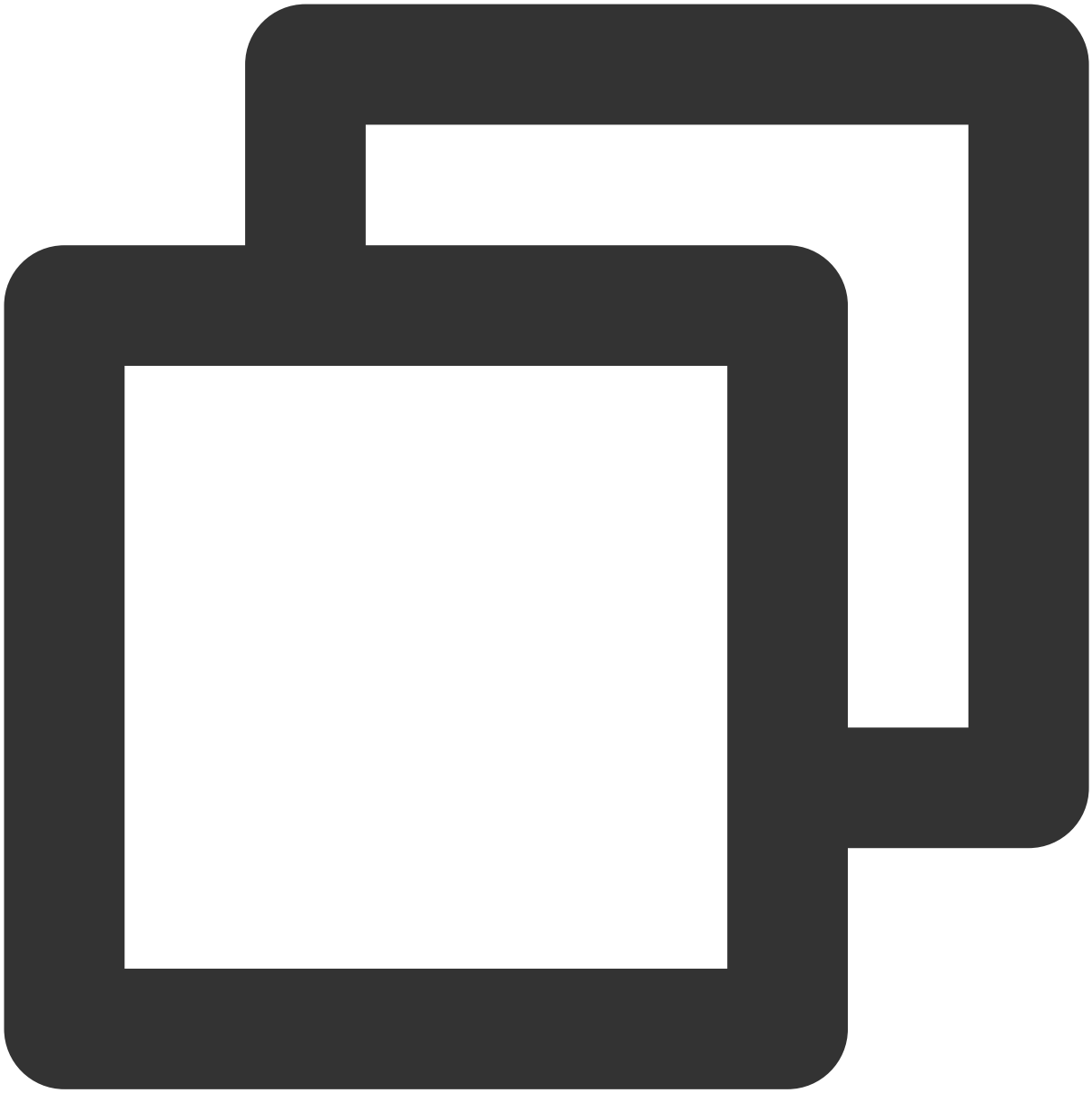
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



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

### enableMultiDeviceAbility

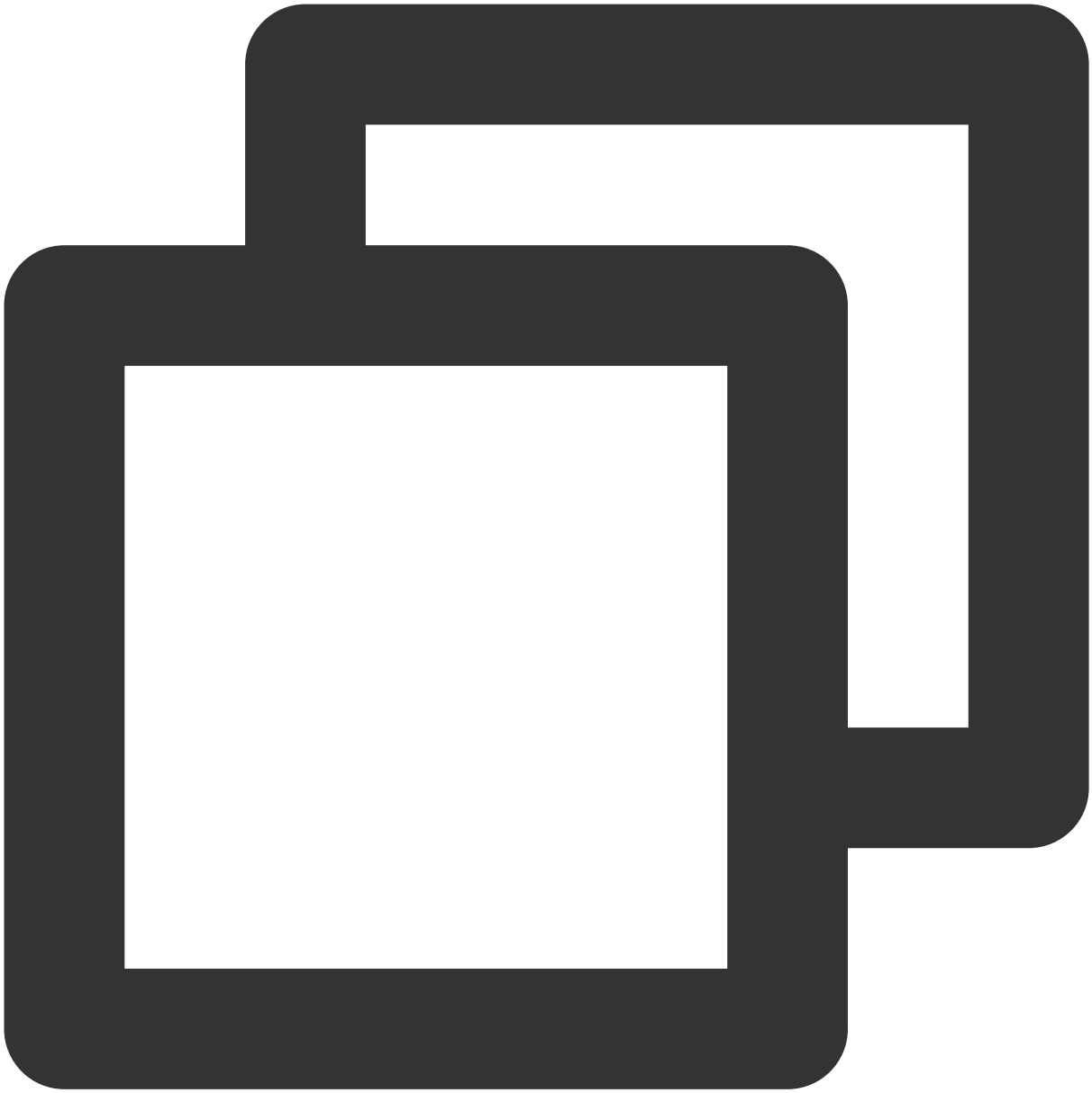
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the premium package).



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

### setVideoRenderParams

Set the rendering mode of video image.



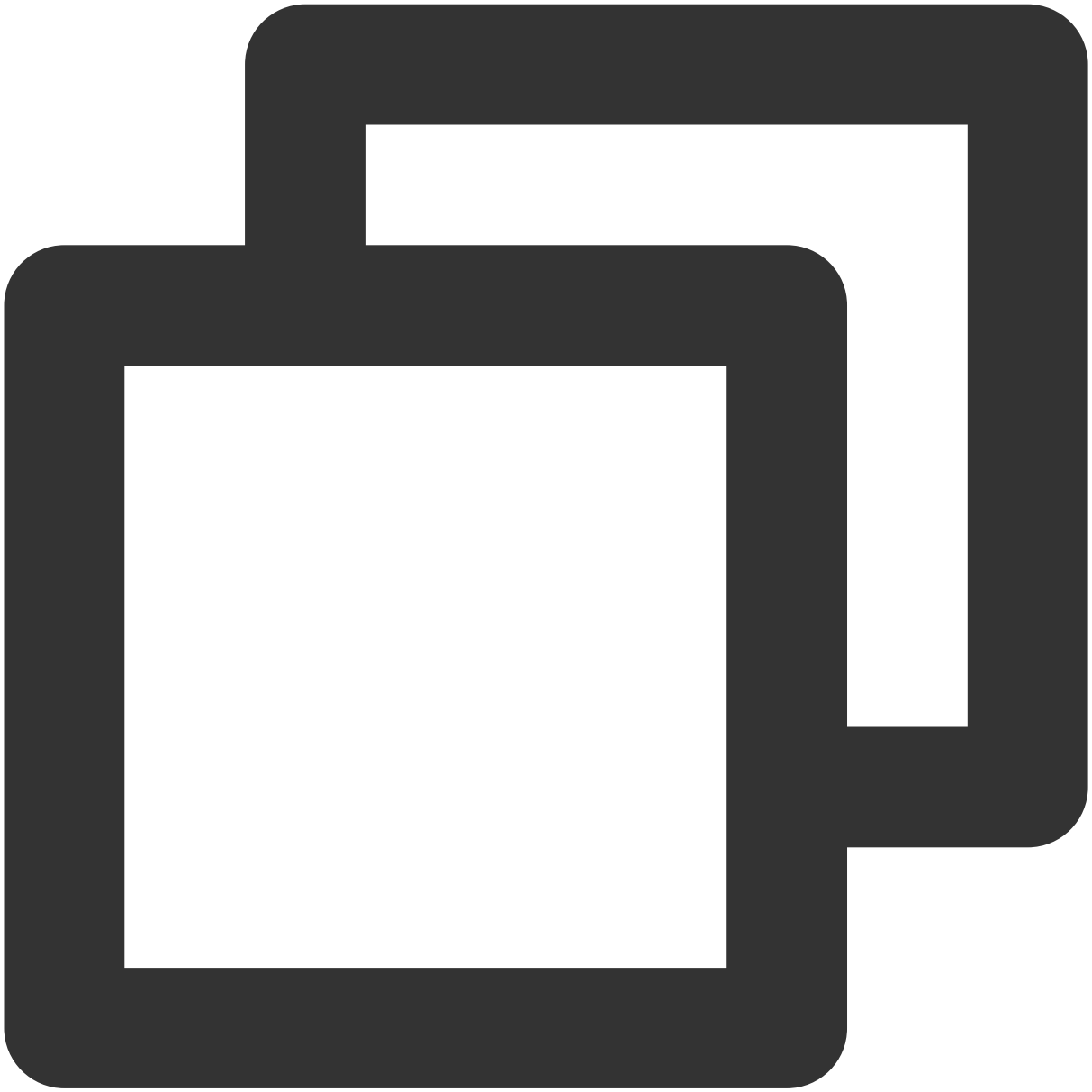
```
- (void)setVideoRenderParams:(NSString *)userId params:(TUIVideoRenderParams *)para
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The target user ID. |
| params | TUIVideoRenderParams | Video render parameters. |

### setVideoEncoderParams

Set the encoding parameters of video encoder.

This setting can determine the quality of image viewed by remote users, which is also the image quality of on-cloud recording files.

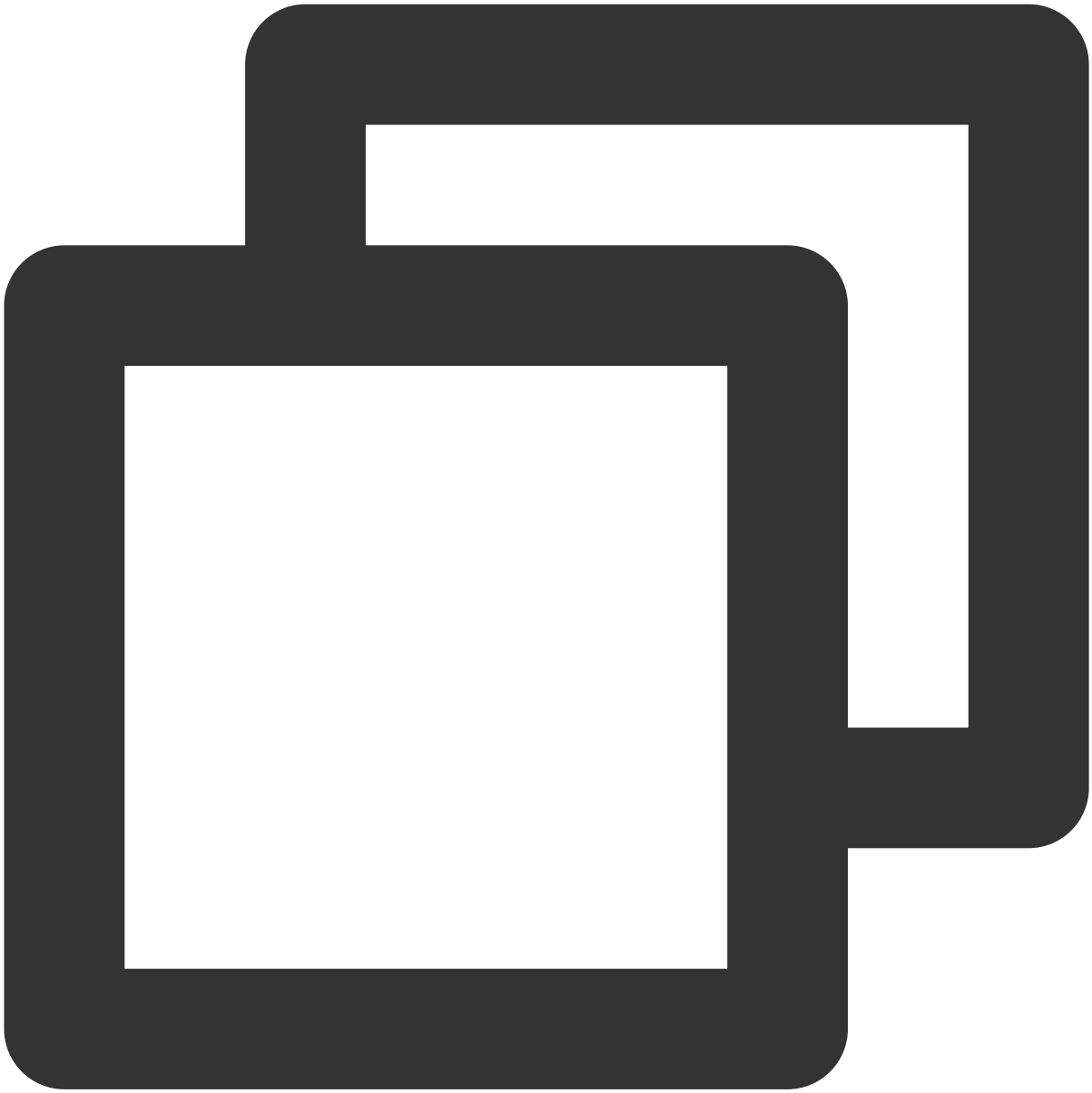


```
- (void)setVideoEncoderParams:(TUIVideoEncoderParams *)params succ:(TUICallSucc)suc
```

| Parameter | Type | Description |
| --- | --- | --- |
| params | TUIVideoEncoderParams | Video encoding parameters |

### getTRTCCloudInstance

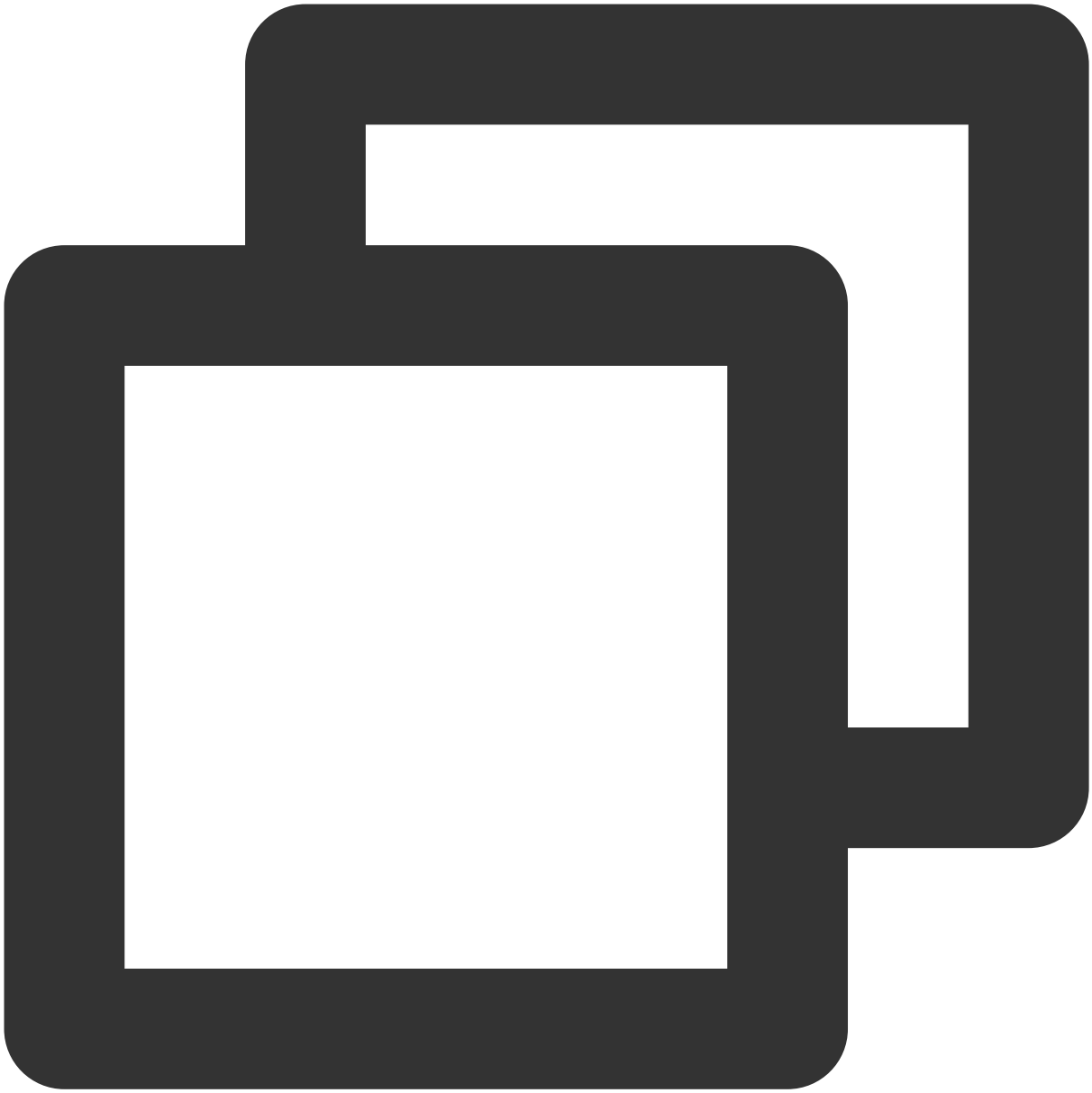
Advanced features.



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

### setBeautyLevel

Set beauty level, support turning off default beauty.



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

| Parameter | Type | Description |
| --- | --- | --- |
| level | CGFloat | Beauty level，range: 0 - 9；<br>0 means turning off the effect, 9 means the most obvious effect. |

# TUICallObserver

Last updated：2024-01-25 14:39:23

## TUICallObserver APIs

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

## Overview

| API | Description |
| --- | --- |
| onError | A call occurred during the call. |
| onCallReceived | A call invitation was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user had a video stream. |
| onUserAudioAvailable | Whether a user had an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | The current user was kicked offline. |
| | |

| | |
|---|---|
| onUserSigExpired | The user sig is expired. |

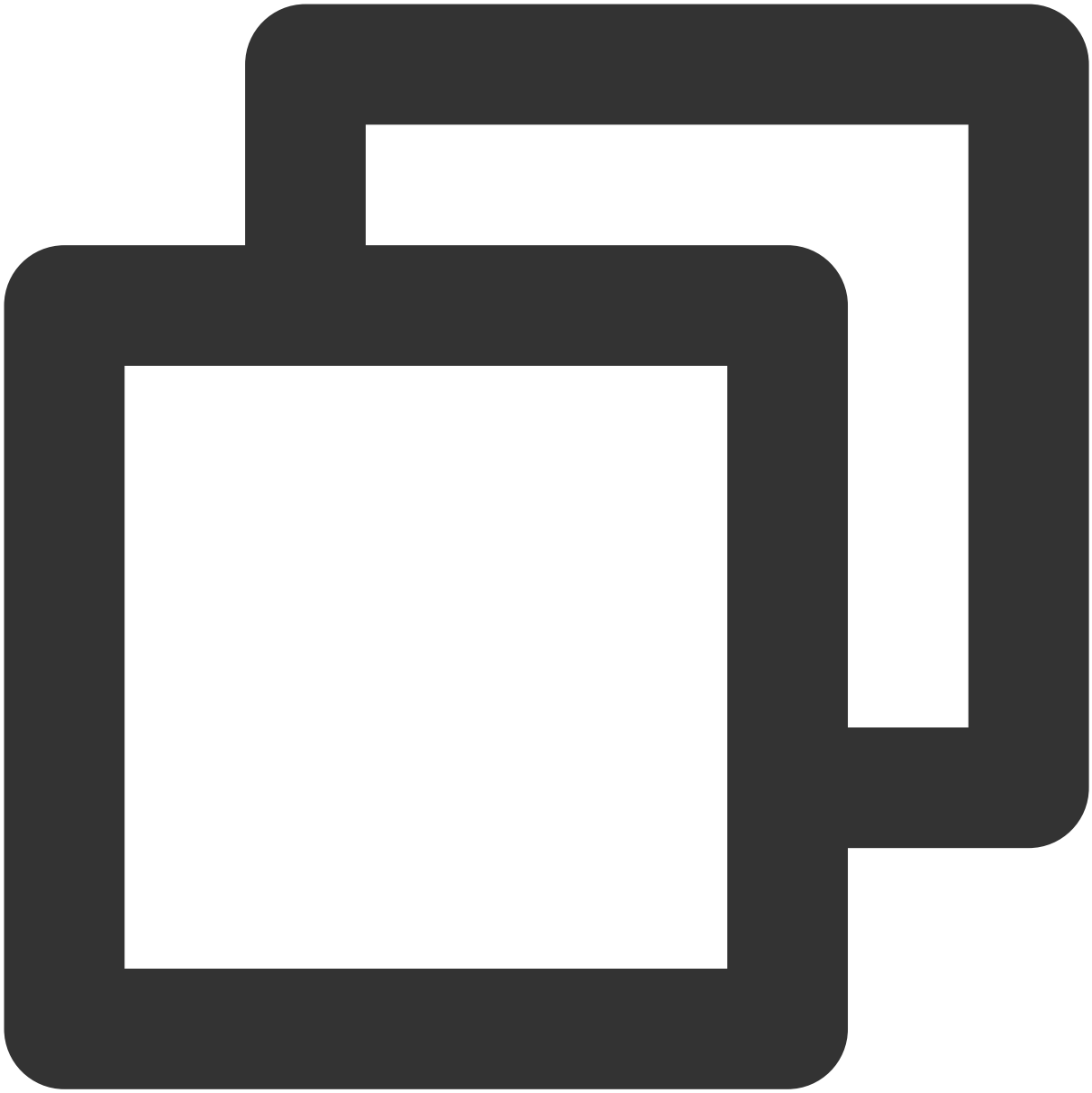
# Details

### onError

An error occurred.

**explain**

This callback indicates that the SDK encountered an unrecoverable error. Such errors must be listened for, and UI reminders should be sent to users if necessary.



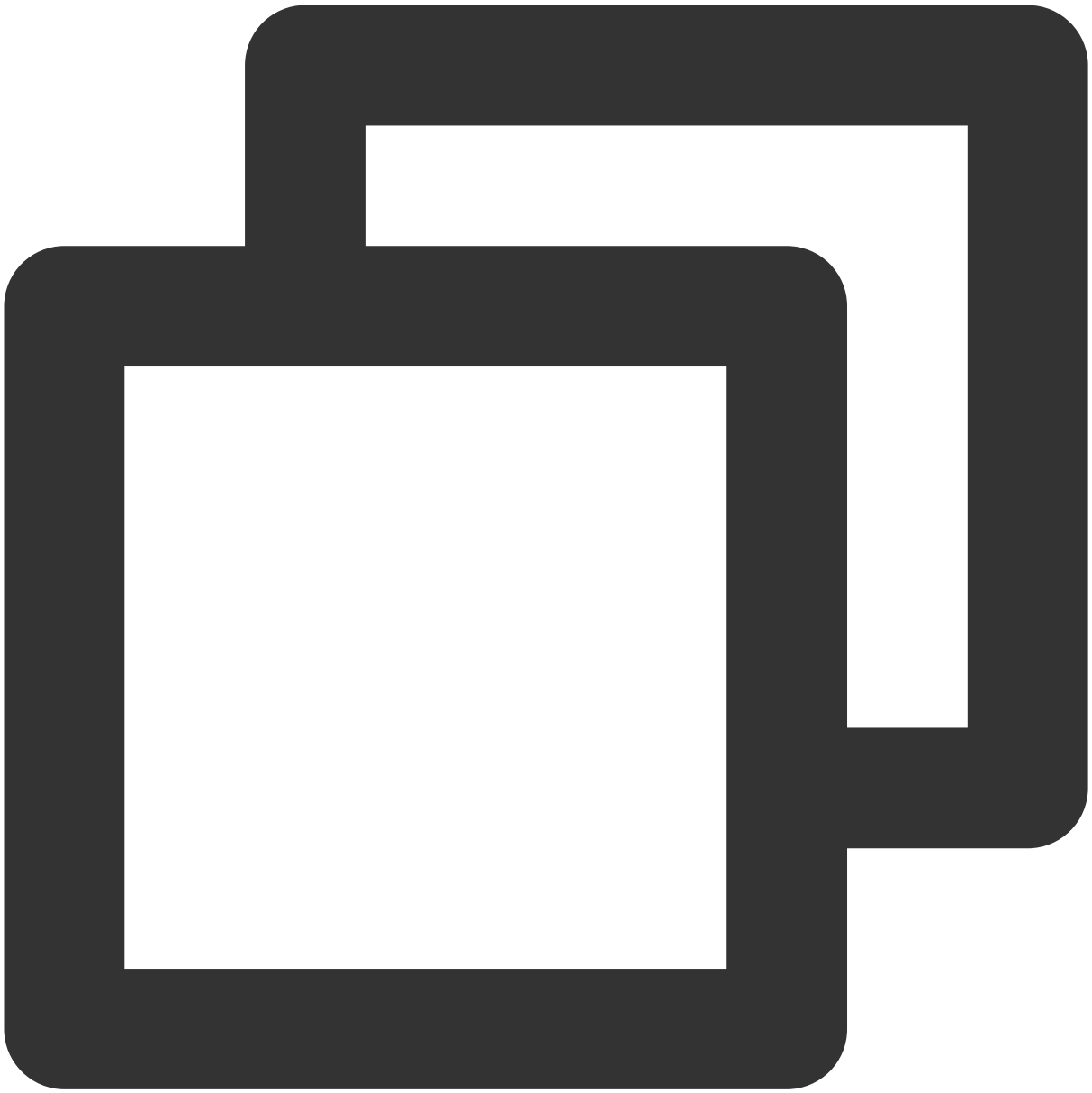
```
void onError(int code, String msg);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| code | int | The error code. |
| msg | NSString | The error message. |

### onCallReceived

A call invitation was received. This callback is received by an invitee. You can listen for this event to determine whether to display the incoming call view.



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

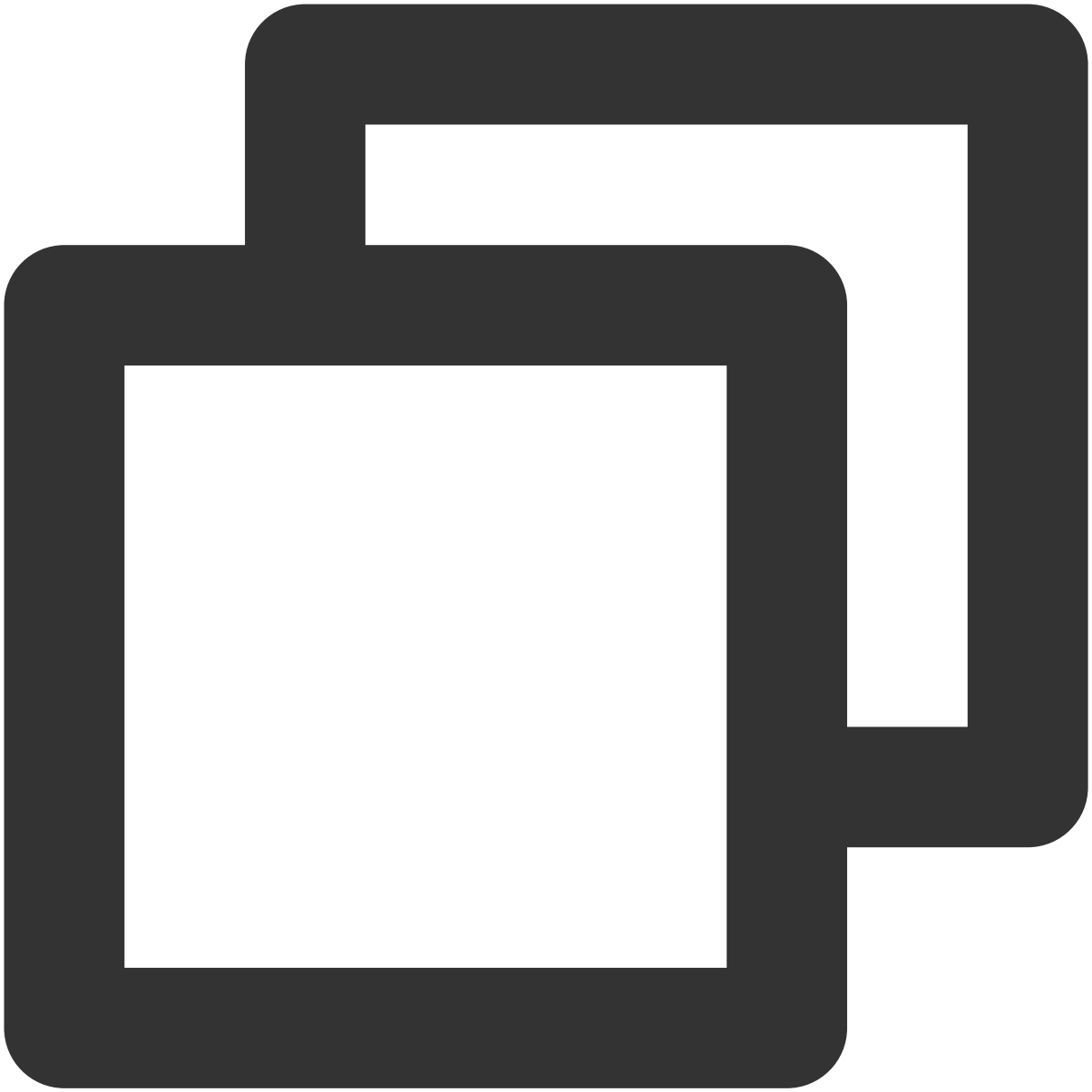
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| callerId | NSString | The user ID of the inviter. |
| | | |

| | | |
|---|---|---|
| calleeIdList | NSArray | The invitee array. |
| groupId | NSString | The group ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| userData | NSString | User-added extended fields., Please refer to: TUICallParams |

### onCallCancelled

The call was canceled by the inviter or timed out. This callback is received by an invitee. You can listen for this event to determine whether to show a missed call message.
This indicates that the call was canceled by the caller, timed out by the callee, rejected by the callee, or the callee was busy. There are multiple scenarios involved. You can listen to this event to achieve UI logic such as missed calls and resetting UI status.
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller)
Callee timeout: the caller will simultaneously receive the onUserNoResponse and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID).
Callee rejection: The caller will simultaneously receive the onUserReject and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID).
Callee busy: The caller will simultaneously receive the onUserLineBusy and onCallCancelled callbacks (userId is his own ID).
Abnormal interruption: The callee failed to receive the call , he receives this callback (userId is his own ID).



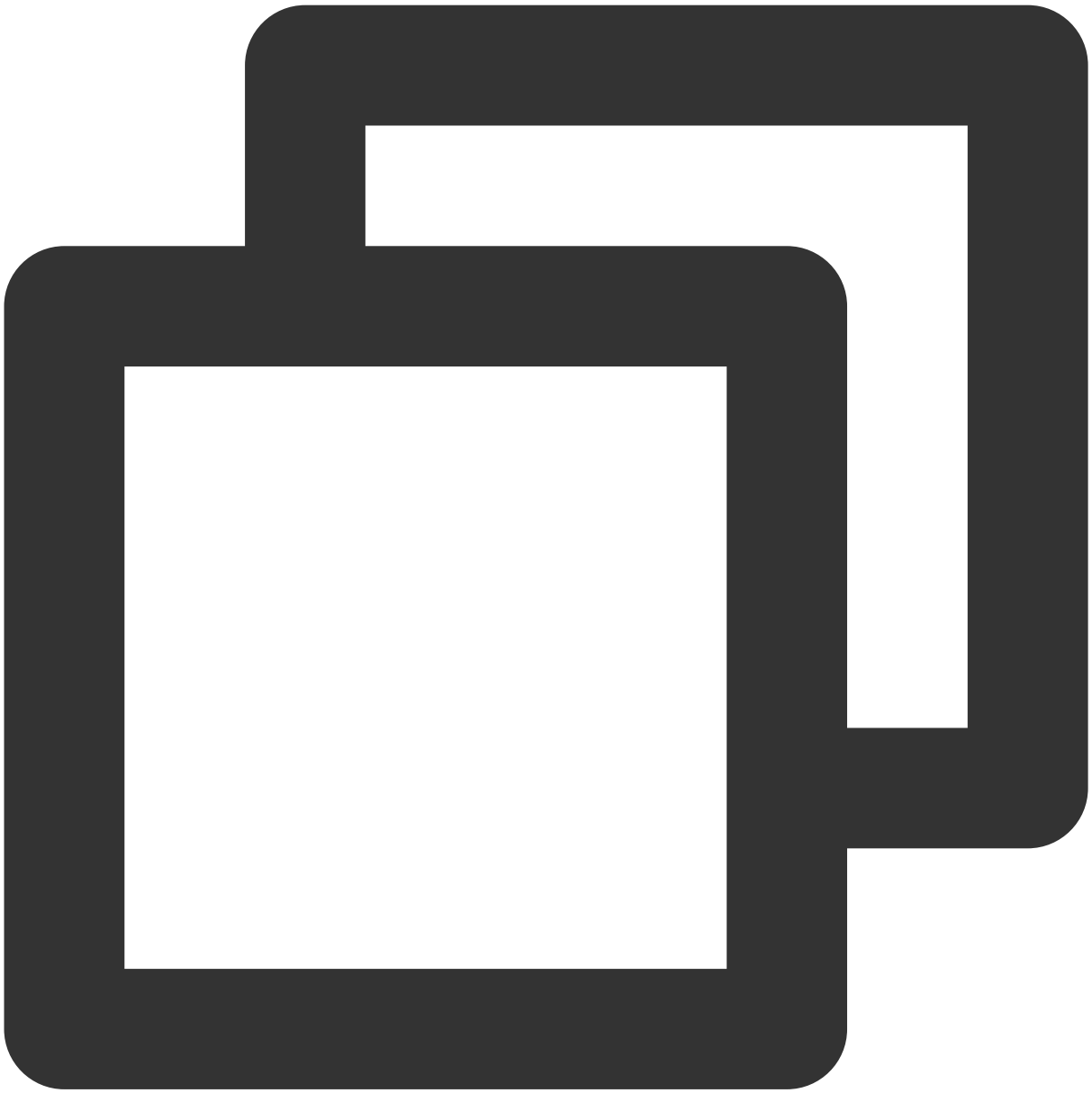
```
void onCallCancelled(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The user ID of the inviter. |

### onCallBegin

The call was connected. This callback is received by both the inviter and invitees. You can listen for this event to determine whether to start on-cloud recording, content moderation, or other tasks.



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

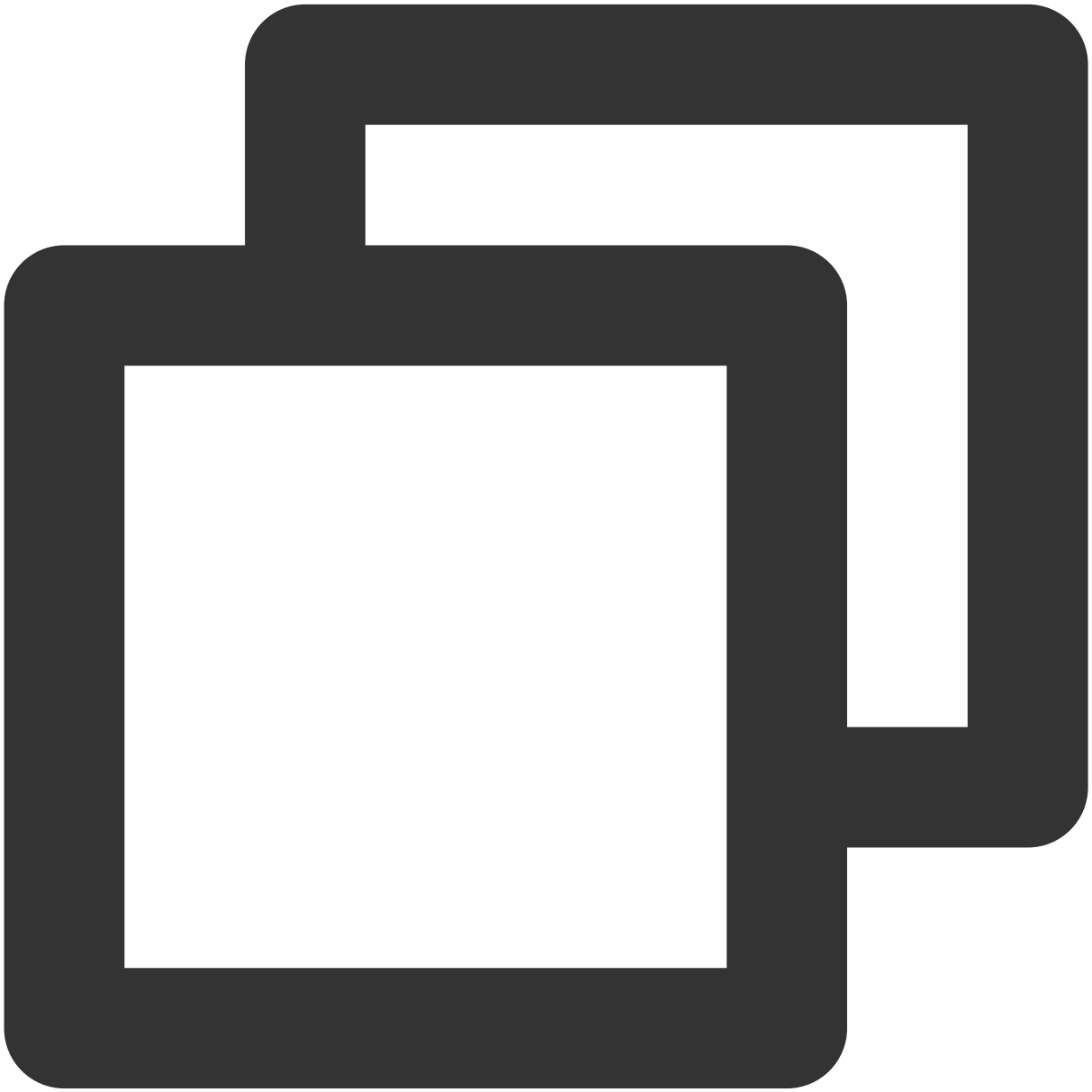
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

| | | |
|---|---|---|
| callRole | TUICallRole | The role, which can be caller or callee. |

### onCallEnd

The call ended. This callback is received by both the inviter and invitees. You can listen for this event to determine when to display call information such as call duration and call type, or stop on-cloud recording.



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

The parameters are described below:

| | | |
|---|---|---|

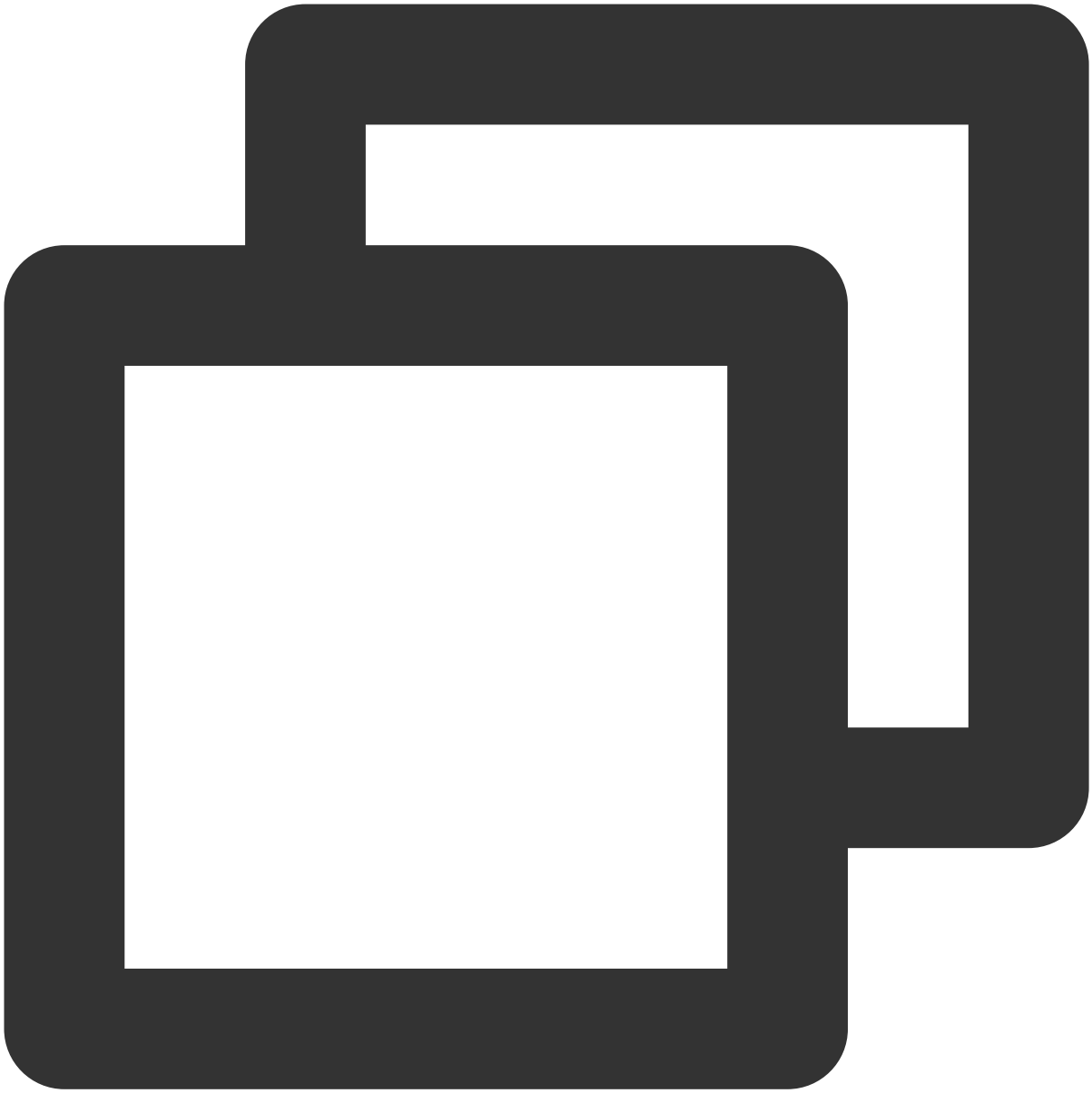
| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| callRole | TUICallRole | The role, which can be caller or callee. |
| totalTime | Float | The call duration. |

**Notice：**

Client-side callbacks are often lost when errors occur, for example, when the process is closed. If you need to measure the duration of a call for billing or other purposes, we recommend you use the RESTful API.

### onCallMediaTypeChanged

The call type changed.



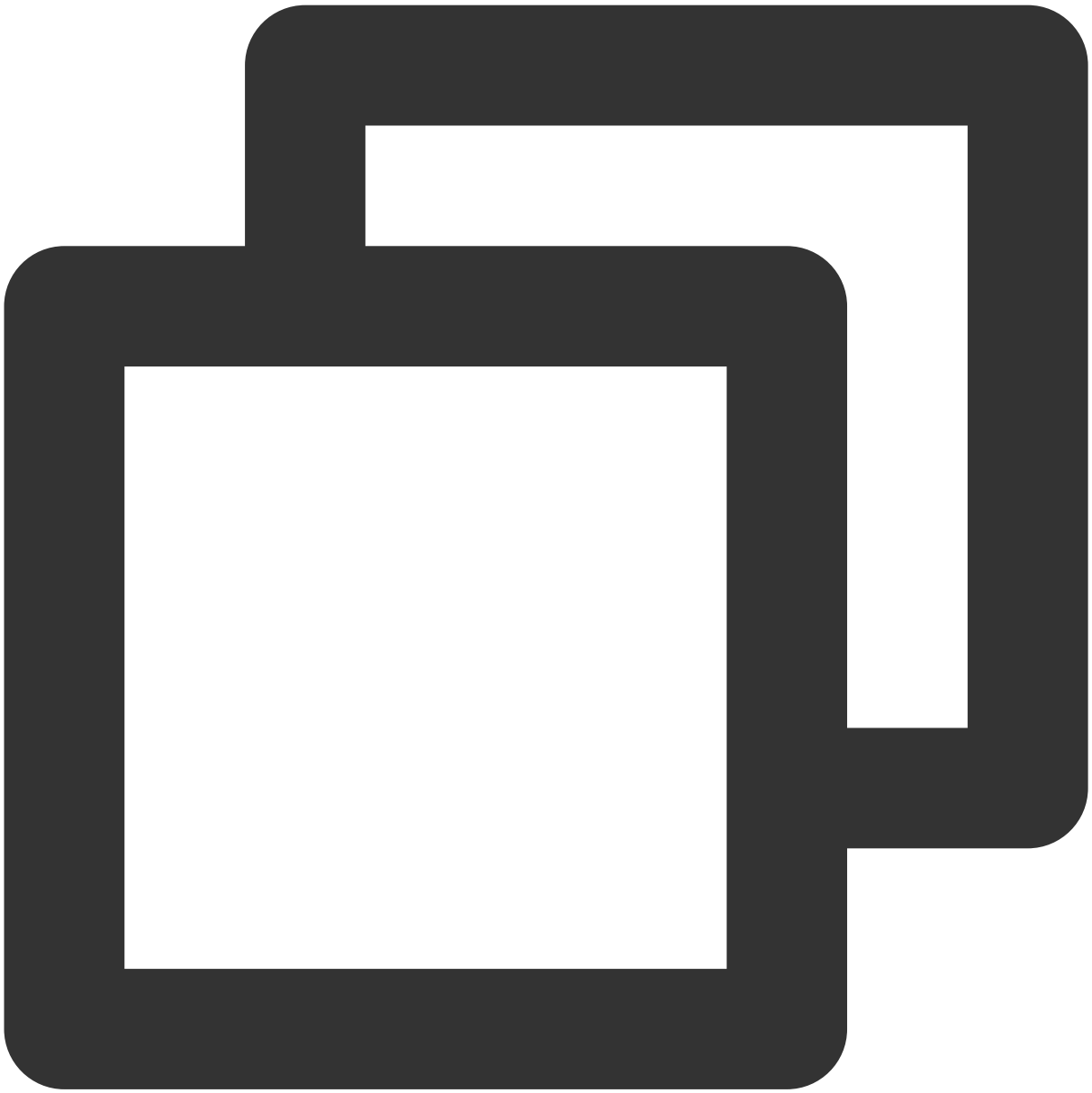
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType,TUICallDefine.
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| oldCallMediaType | TUICallMediaType | The call type before the change. |
| newCallMediaType | TUICallMediaType | The call type after the change. |

### onUserReject

The call was rejected. In a one-to-one call, only the inviter will receive this callback. In a group call, all invitees will receive this callback.



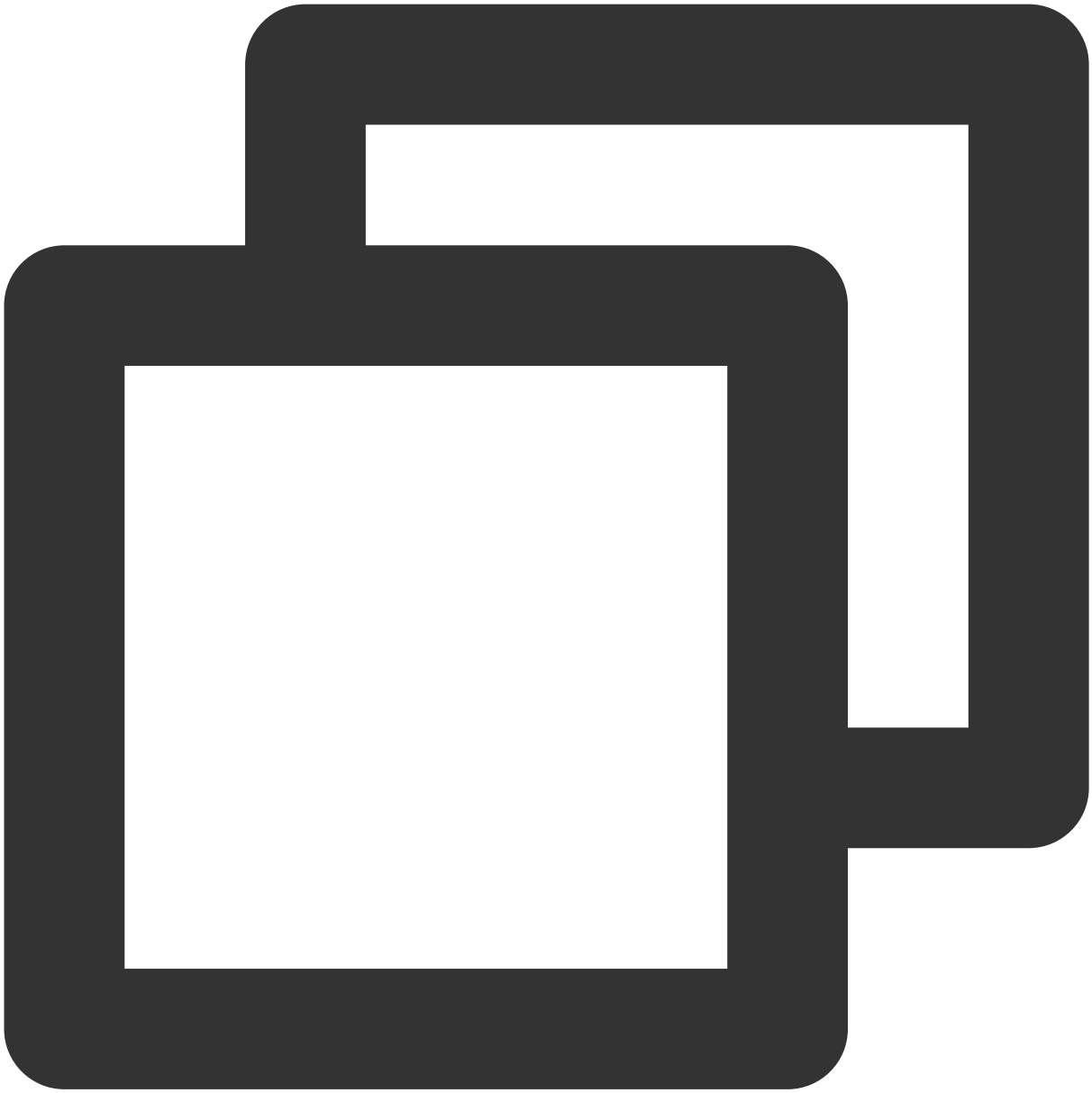
```
void onUserReject(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The user ID of the invitee who rejected the call. |

### onUserNoResponse

A user did not respond.



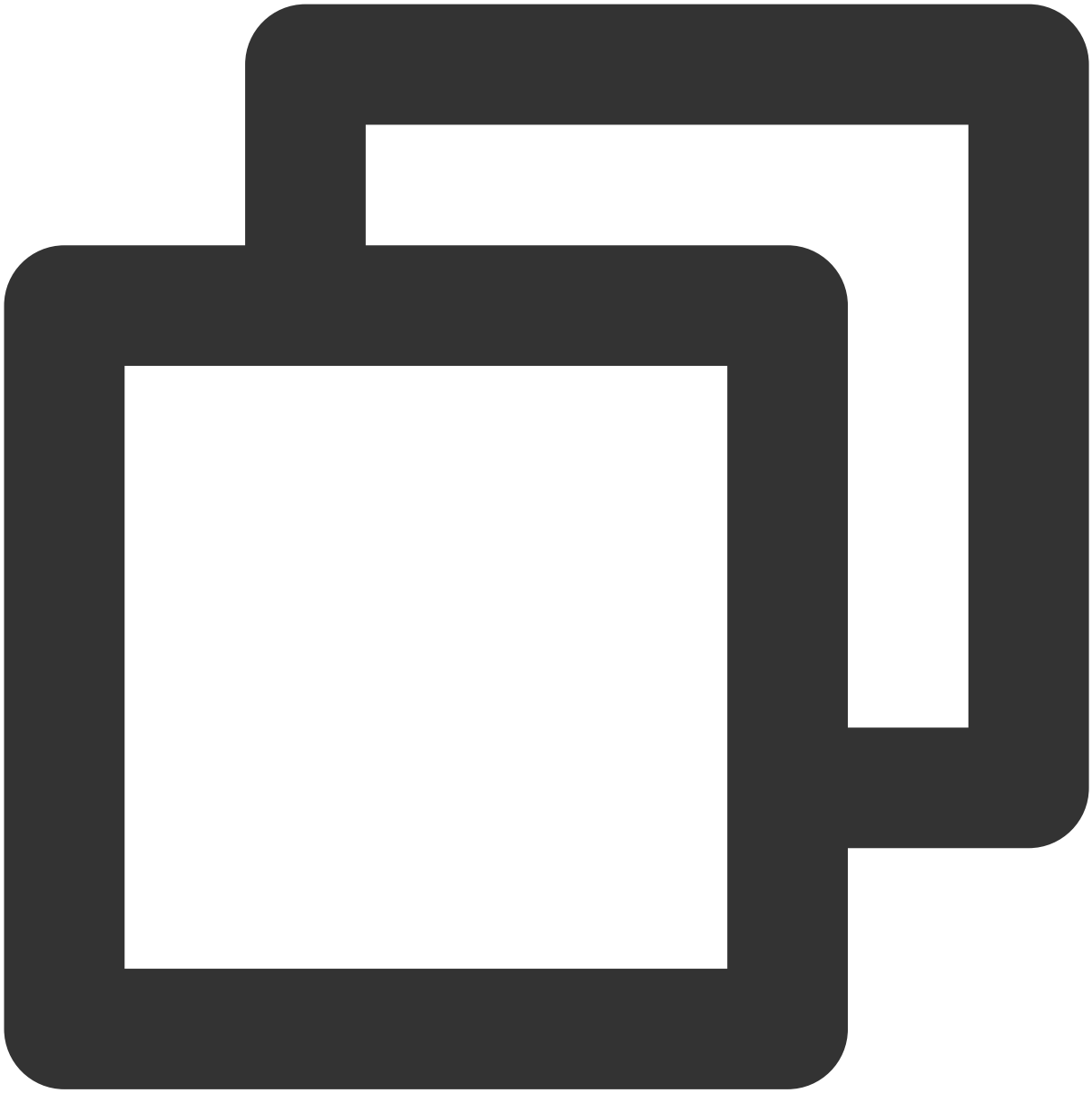
```
void onUserNoResponse(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The user ID of the invitee who did not answer. |

### onUserLineBusy

A user is busy.



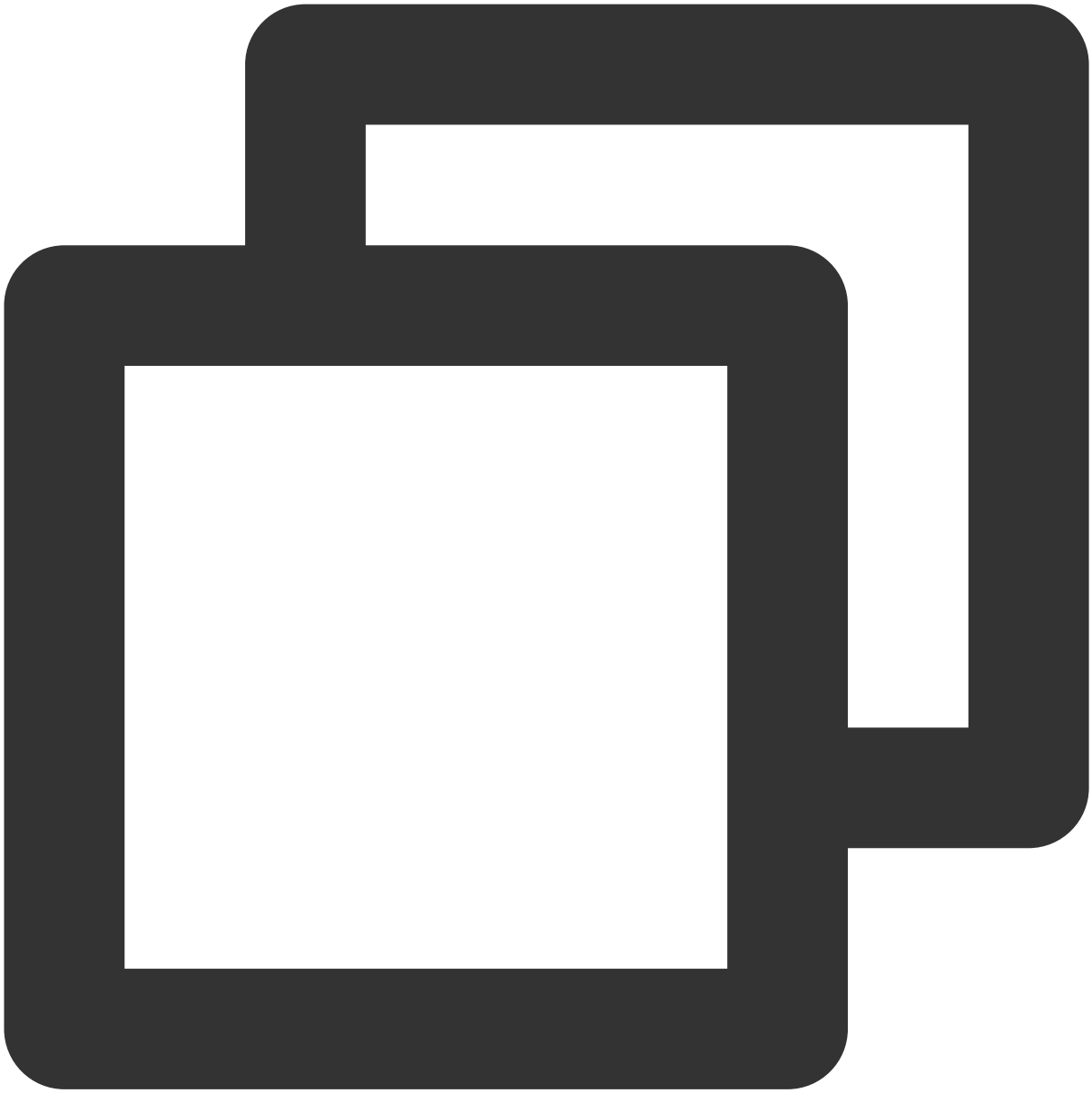
```
void onUserLineBusy(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The user ID of the invitee who is busy. |

### onUserJoin

A user joined the call.



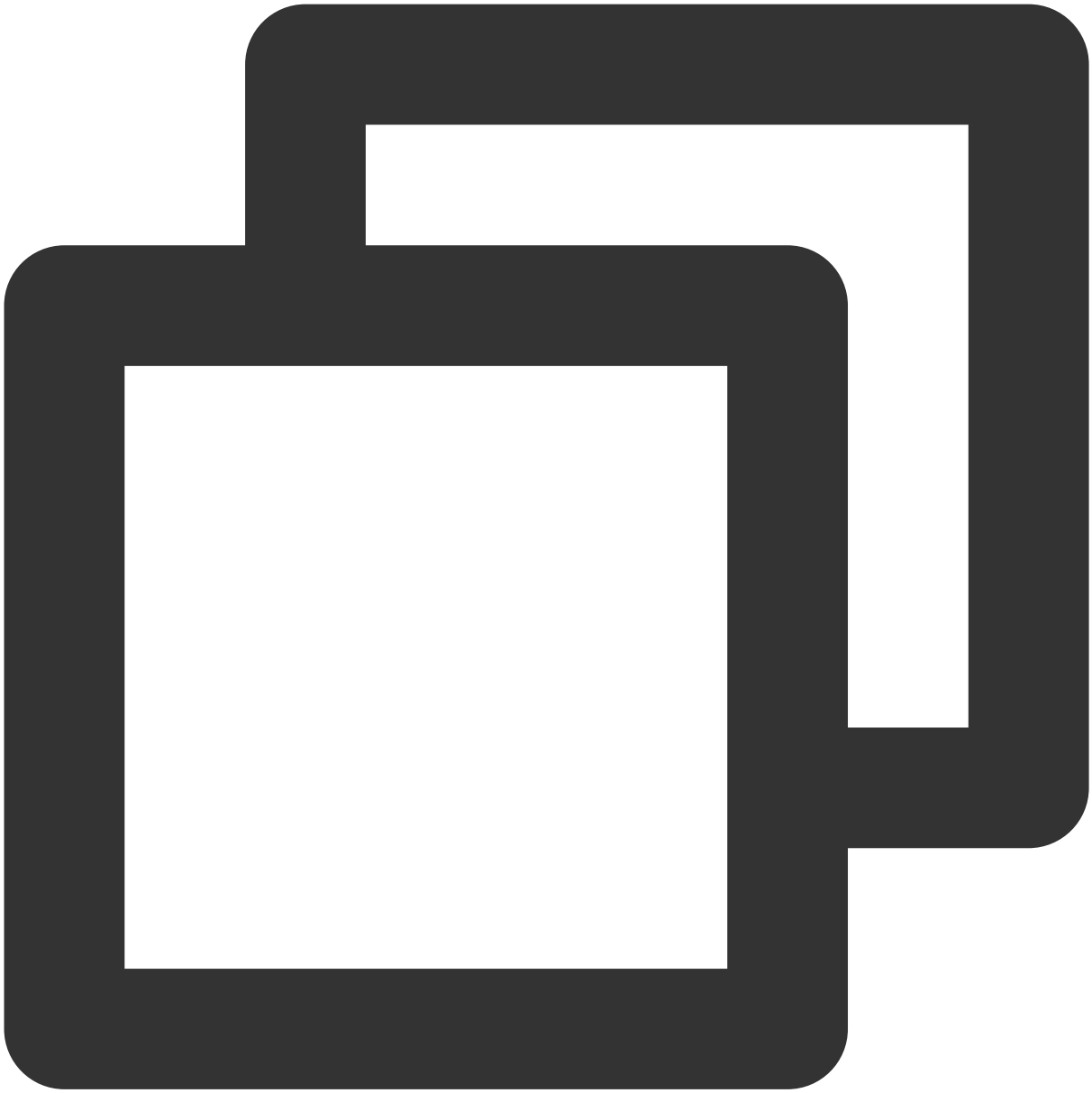
```
void onUserJoin(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The ID of the user who joined the call. |

### onUserLeave

A user left the call.



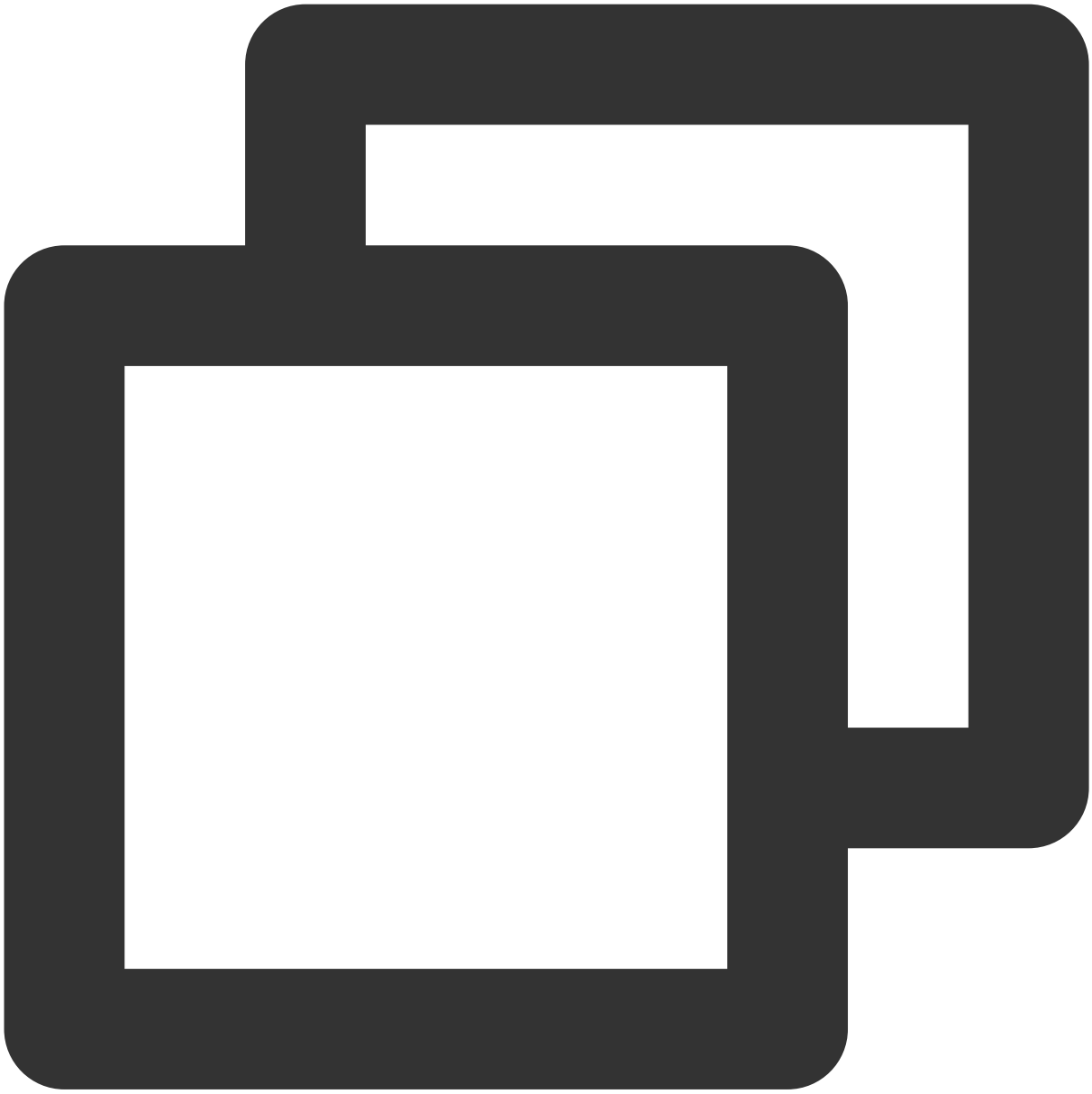
```
void onUserLeave(String userId);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The ID of the user who left the call. |

### onUserVideoAvailable

Whether a user is sending video.



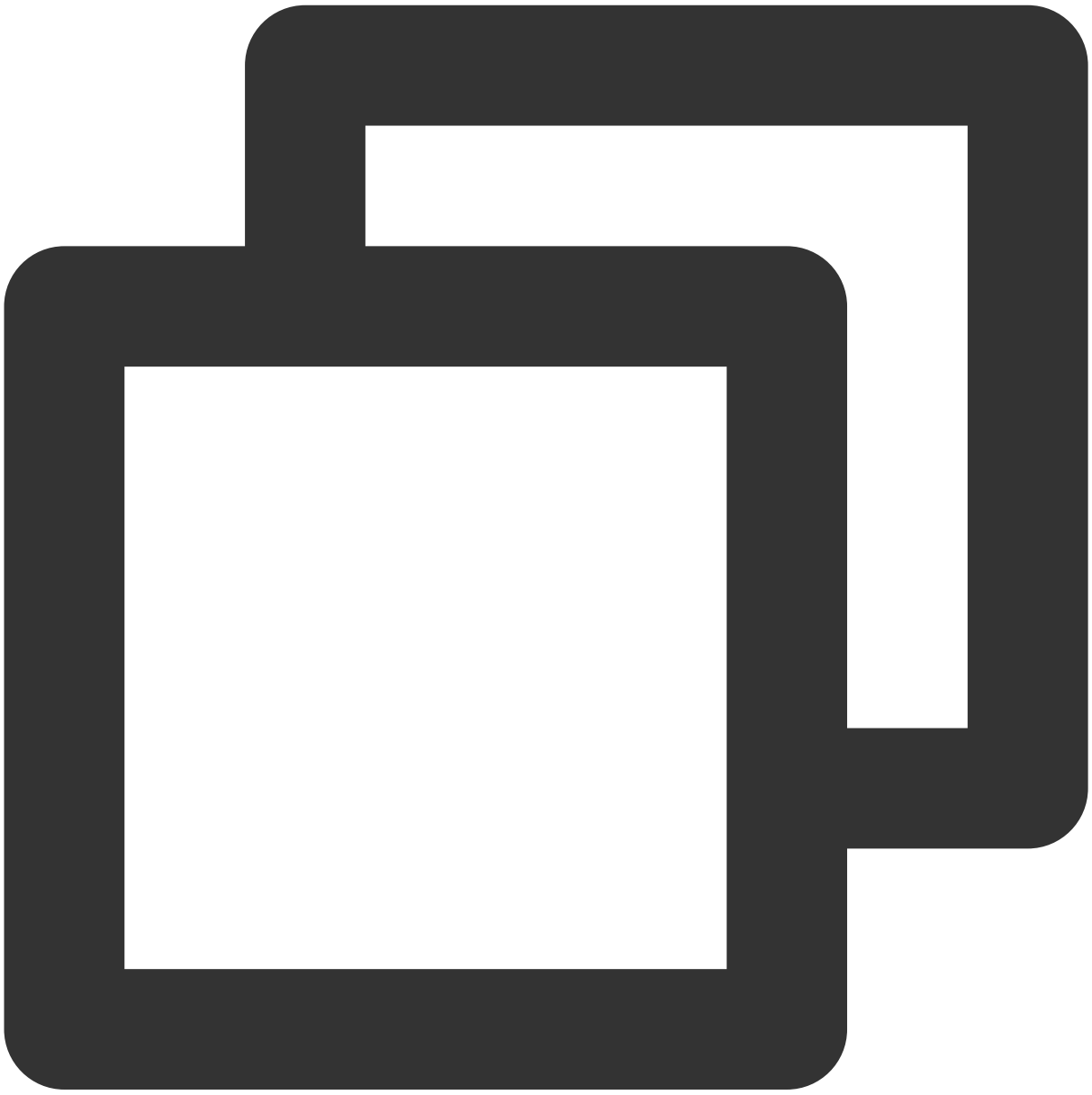
```
void onUserVideoAvailable(String userId, boolean isVideoAvailable);
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | |
| isVideoAvailable | BOOL | Whether the user has video. |

### onUserAudioAvailable

Whether a user is sending audio.



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

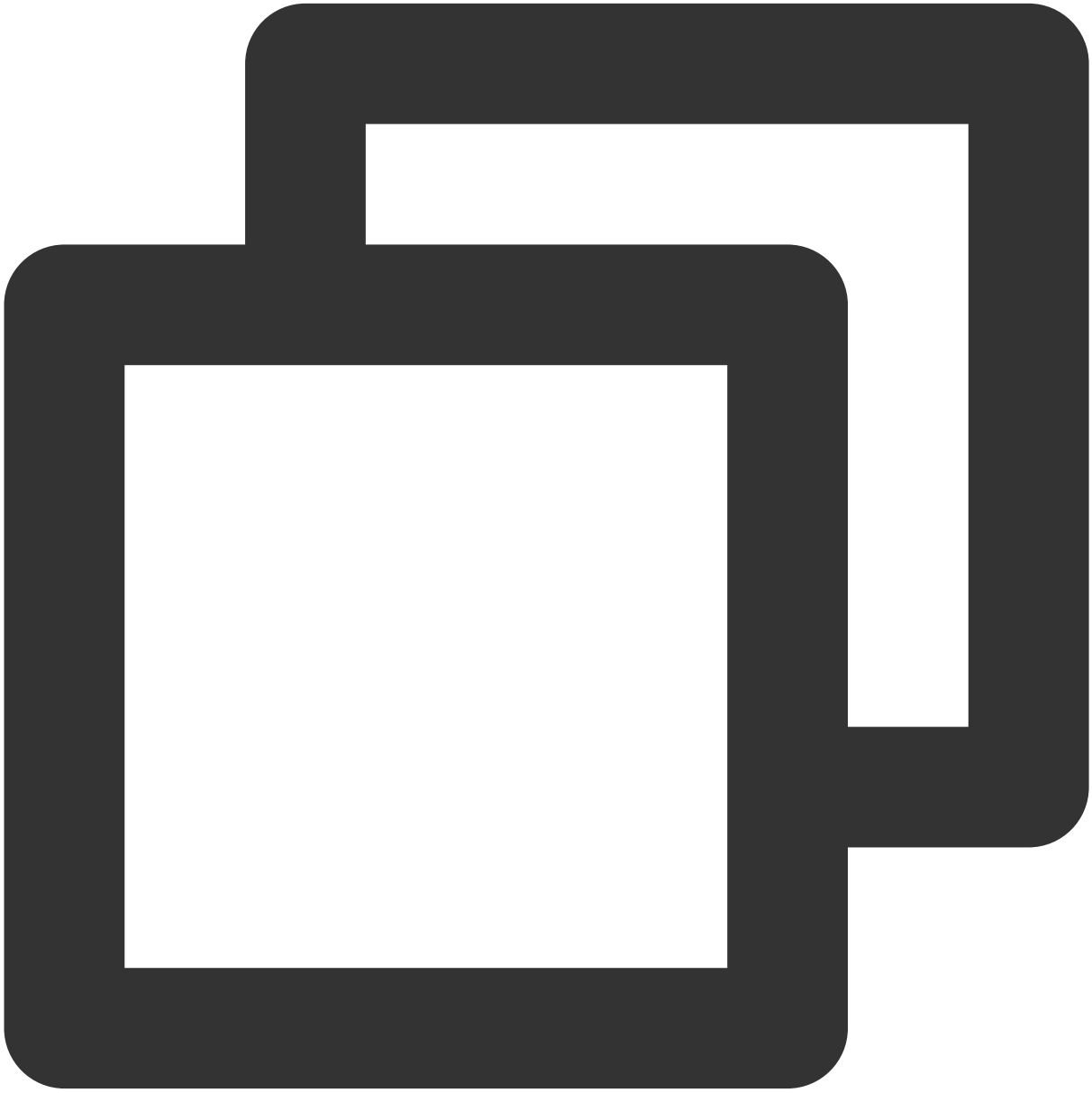
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | NSString | The user ID. |

| | | |
|---|---|---|
| isAudioAvailable | BOOL | Whether the user has audio. |

### onUserVoiceVolumeChanged

The volumes of all users.



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

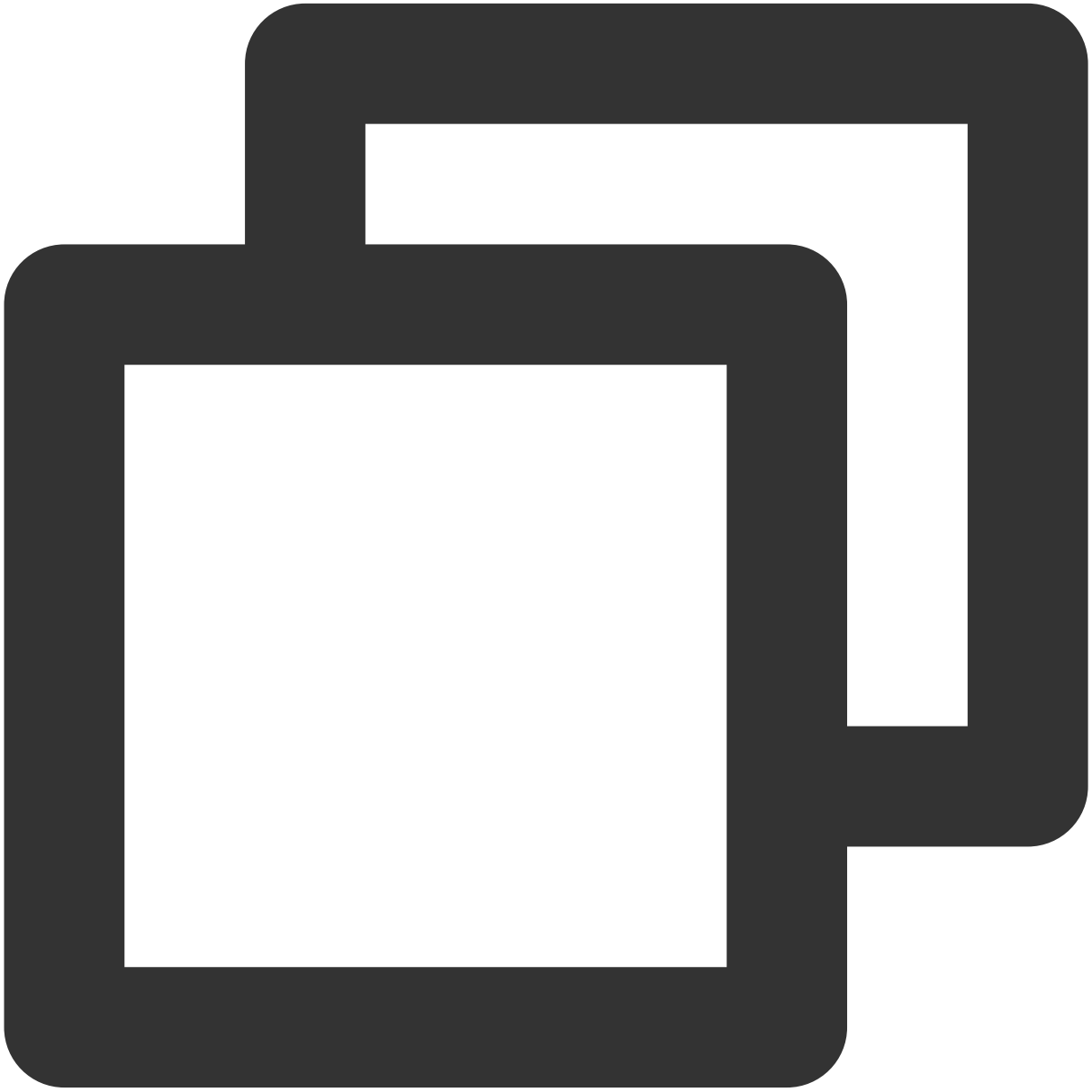
The parameters are described below:

| Parameter | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| volumeMap | NSDictionary <NSString *, NSNumber *> | The volume table, which includes the volume of each user ( `userId` ). Value range: 0-100. |

### onUserNetworkQualityChanged

The network quality of all users.



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

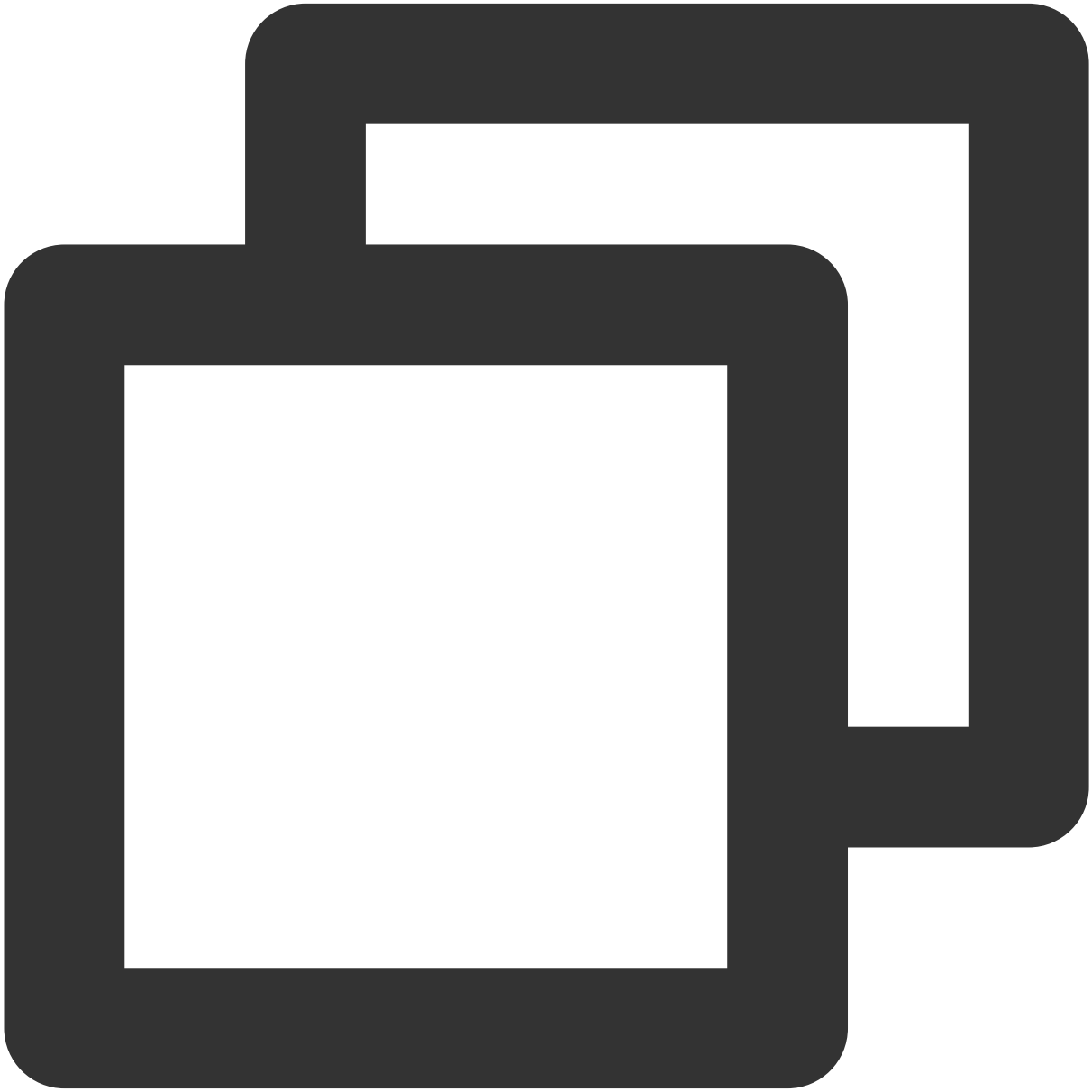
The parameters are described below:

| | | |
|---|---|---|

| Parameter | Type | Description |
| --- | --- | --- |
| networkQualityList | NSArray | The current network conditions for all users ( `userId` ). |

### onKickedOffline

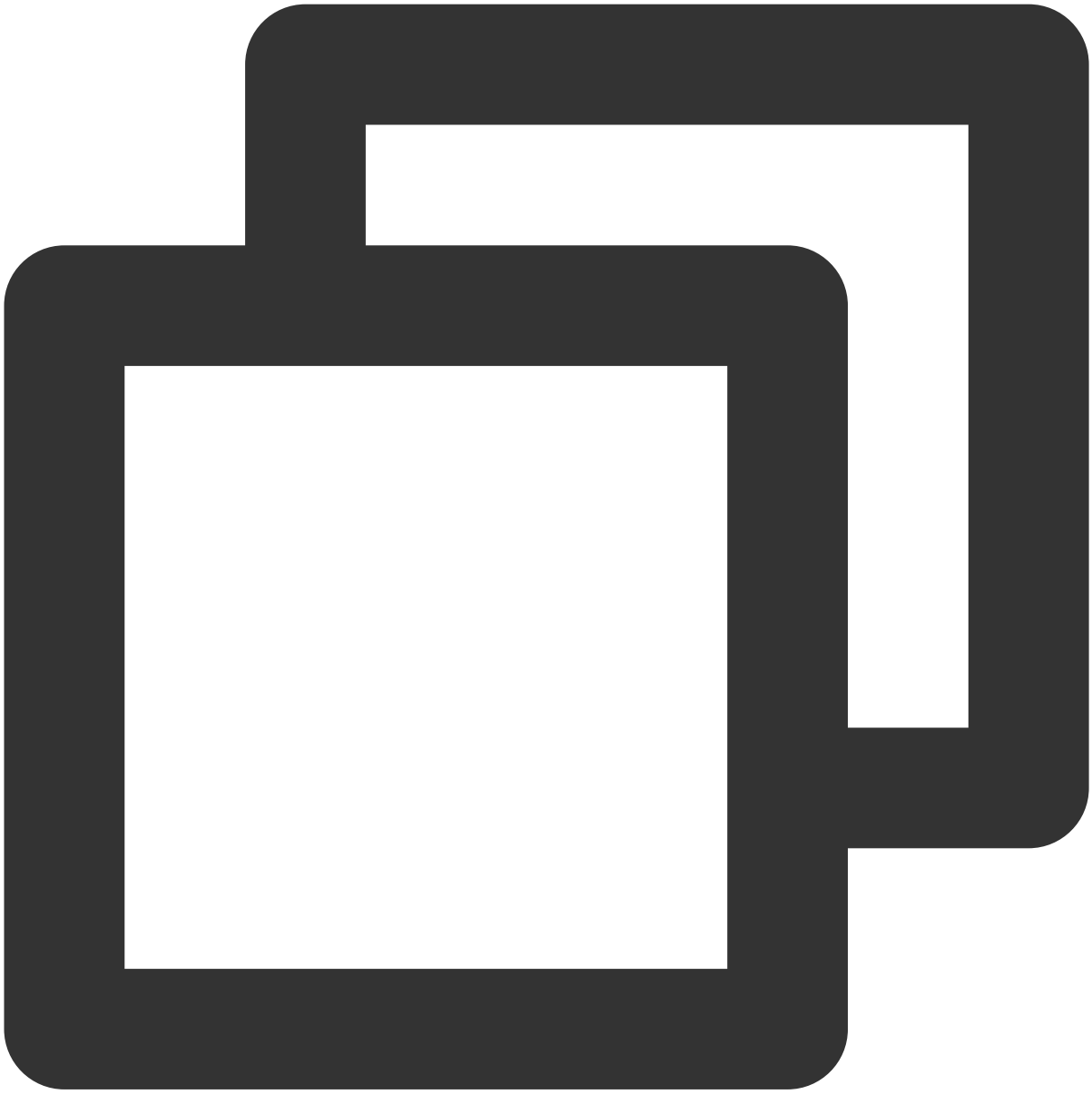
The current user was kicked offline：At this time, you can prompt the user with a UI message and then invoke `init` again.



```
void onKickedOffline();
```

### onUserSigExpired

The user sig is expired：At this time, you need to generate a new `userSig` , and then invoke `init` again.



```
void onUserSigExpired();
```

# Type Definition

Last updated：2023-07-04 14:51:46

## Common structures

### TUICallDefine

| Type | Description |
| --- | --- |
| TUICallParams | An additional parameter. |
| TUIOfflinePushInfo | Offline push vendor configuration information. |

### TUICommonDefine

| Type | Description |
| --- | --- |
| TUIRoomId | Room ID for audio and video in a call. |
| TUINetworkQuality | Network quality information |
| TUIVideoRenderParams | Video render parameters |
| TUIVideoEncoderParams | Video encoding parameters |

## Enum definition

### TUICallDefine

| Type | Description |
| --- | --- |
| TUICallMediaType | Media type in a call |
| TUICallRole | Roles of individuals in a call. |
| TUICallStatus | The call status |
| TUICallScene | The call scene |
| TUICallIOSOfflinePushType | iOS offline push type |

### TUICommonDefine

| Type | Description |
| --- | --- |
| | |

| | |
|---|---|
| TUIAudioPlaybackDevice | Audio route |
| TUICamera | Camera type |
| TUINetworkQuality | Network quality |
| TUIVideoRenderParamsFillMode | Video image fill mode |
| TUIVideoRenderParamsRotation | Video image rotation direction |
| TUIVideoEncoderParamsResolutionMode | Video aspect ratio mode |
| TUIVideoEncoderParamsResolution | Video resolution |

### TUICallParams

Call params

| 参数 | Type | Description |
|---|---|---|
| roomId | TUIRoomId | Room ID for audio and video in a call. |
| offlinePushInfo | TUIOfflinePushInfo | Offline push vendor configuration information. |
| timeout | int | Call timeout period, default: 30s, unit: seconds. |
| userData | NSString | An additional parameter. Callback when the callee receives onCallReceived |

### TUIOfflinePushInfo

Offline push vendor configuration information， please refer to：Offline call push

| Value | Type | Description |
|---|---|---|
| title | NSString | offlinepush notification title |
| desc | NSString | offlinepush notification description |
| ignoreIOSBadge | BOOL | Ignore badge count for offline push (only for iOS), if set to true, the message will not increase the unread count of the app icon on the iOS receiver's side. |
| iOSSound | NSString | Offline push sound setting (only for iOS). When sound = **IOS_OFFLINE_PUSH_NO_SOUND**, there will be no sound played when the message is received. When sound = **IOS_OFFLINE_PUSH_DEFAULT_SOUND**, |

| | | |
|---|---|---|
| | | the system sound will be played when the message is received. If you want to customize the iOSSound, you need to link the audio file into the Xcode project first, and then set the audio file name (with extension) to the iOSSound. |
| androidSound | NSString | Offline push sound setting (only for Android, supported by IMSDK 6.1 and above). Only Huawei and Google phones support setting sound prompts. For Xiaomi phones, please refer to: Xiaomi custom ringtones. In addition, for Google phones, in order to set sound prompts for FCM push on Android 8.0 and above systems, you must call setAndroidFCMChannelID to set the channelID for it to take effect. |
| androidOPPOChannelID | NSString | Set the channel ID for OPPO phones with Android 8.0 and above systems. |
| androidVIVOClassification | NSInteger | Classification of VIVO push messages (deprecated interface, VIVO push service will optimize message classification rules on April 3, 2023. It is recommended to use setAndroidVIVOCategory to set the message category). 0: Operational messages, 1: System messages. The default value is 1. |
| androidXiaoMiChannelID | NSString | Set the channel ID for Xiaomi phones with Android 8.0 and above systems. |
| androidFCMChannelID | NSString | Set the channel ID for google phones with Android 8.0 and above systems. |
| androidHuaWeiCategory | NSString | Classification of Huawei push messages, please refer to: Huawei message classification standard. |
| isDisablePush | BOOL | Whether to turn off push notifications (default is on). |
| iOSPushType | TUICallIOSOfflinePushType | iOS offline push type， default is APNs |

### TUIRoomId

Room ID for audio and video in a call.

**Note：**

(1) `intRoomId` and `strRoomId` are mutually exclusive. If you choose to use `strRoomId`, `intRoomId` needs to be filled in as 0. If both are filled in, the SDK will prioritize `intRoomId`.
(2) Do not mix `intRoomId` and `strRoomId` because they are not interchangeable. For example, the number 123 and the string "123" represent two completely different rooms.

| Value | Type | Description |
| --- | --- | --- |
| intRoomId | UInt32 | Numeric room ID.<br>**range:** 1 - 2147483647(2^31-1) |
| strRoomId | NSString | String room ID.<br>**range**：Limited to 64 bytes in length. The supported character set range is as follows (<br>Lowercase and uppercase English letters. (a-zA-Z)<br>Number (0-9)<br>Spaces、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`、 |

**Note：**
Currently, string room number is only supported on Android and iOS platforms. Support for other platforms such as Web, Mini Programs, Flutter, and Uniapp will be available in the future. Please stay tuned!

### TUIVideoRenderParams

Video render parameters

| Value | Type | Description |
| --- | --- | --- |
| fillMode | TUIVideoRenderParamsFillMode | Video image fill mode |
| rotation | TUIVideoRenderParamsRotation | Video image rotation direction |

### TUINetworkQualityInfo

User network quality information

| Value | Type | Description |
| --- | --- | --- |
| userId | NSString | user ID |
| quality | NetworkQuality | network quality |

### TUIVideoEncoderParams

Video encoding parameters

| Value | Type | Description |
| --- | --- | --- |
| resolution | TUIVideoEncoderParamsResolution | Video resolution |
| resolutionMode | TUIVideoEncoderParamsResolutionMode | Video aspect ratio mode |

### TUICallMediaType

Call media type

| Type | Value | Description |
| --- | --- | --- |
| TUICallMediaTypeUnknown | 0 | Unknown |
| TUICallMediaTypeAudio | 1 | Audio call |
| TUICallMediaTypeVideo | 2 | Video call |

### TUICallRole

Call role

| Type | Value | Description |
| --- | --- | --- |
| TUICallRoleNone | 0 | Unknown |
| TUICallRoleCall | 1 | Caller（inviter） |
| TUICallRoleCalled | 2 | Callee（invitee） |

### TUICallStatus

Call status

| Type | Value | Description |
| --- | --- | --- |
| TUICallStatusNone | 0 | Unknown |
| TUICallStatusWaiting | 1 | The call is currently waiting |
| TUICallStatusAccept | 2 | The call has been accepted |

### TUICallScene

Call scene

| Type | Value | Description |
| --- | --- | --- |
| | | |

| | | |
|---|---|---|
| TUICallSceneGroup | 0 | Group call |
| TUICallSceneMulti | 1 | Anonymous group calling (not supported at this moment, please stay tuned). |
| TUICallSceneSingle | 2 | one to one call |

### TUICallIOSOfflinePushType

iOS offline push type

| Type | Value | Description |
|---|---|---|
| TUICallIOSOfflinePushTypeAPNs | 0 | APNs |
| TUICallIOSOfflinePushTypeVoIP | 1 | VoIP |

### TUIAudioPlaybackDevice

Audio route

| Type | Value | Description |
|---|---|---|
| TUIAudioPlaybackDeviceSpeakerphone | 0 | Speakerphone |
| TUIAudioPlaybackDeviceEarpiece | 1 | Earpiece |

### TUICamera

Front/Back camera

| Type | Value | Description |
|---|---|---|
| TUICameraFront | 0 | Front camera |
| TUICameraBack | 1 | Back camera |

### TUINetworkQuality

Network quality

| Type | Value | Description |
|---|---|---|
| TUINetworkQualityUnknown | 0 | Unknown |
| TUINetworkQualityExcellent | 1 | Excellent |

| | | |
|---|---|---|
| TUINetworkQualityGood | 2 | Good |
| TUINetworkQualityPoor | 3 | Poor |
| TUINetworkQualityBad | 4 | Bad |
| TUINetworkQualityVbad | 5 | Vbad |
| TUINetworkQualityDown | 6 | Down |

### TUIVideoRenderParamsFillMode

If the aspect ratio of the video display area is not equal to that of the video image, you need to specify the fill mode:

| Type | Value | Description |
|---|---|---|
| TUIVideoRenderParamsFillModeFill | 0 | Fill mode: the video image will be centered and scaled to fill the entire display area, where parts that exceed the area will be cropped. The displayed image may be incomplete in this mode. |
| TUIVideoRenderParamsFillModeFit | 1 | Fit mode: the video image will be scaled based on its long side to fit the display area, where the short side will be filled with black bars. The displayed image is complete in this mode, but there may be black bars. |

### TUIVideoRenderParamsRotation

We provides rotation angle setting APIs for local and remote images. The following rotation angles are all clockwise.

| Type | Value | Description |
|---|---|---|
| TUIVideoRenderParamsRotation_0 | 0 | No rotation |
| TUIVideoRenderParamsRotation_90 | 1 | Clockwise rotation by 90 degrees |
| TUIVideoRenderParamsRotation_180 | 2 | Clockwise rotation by 180 degrees |
| TUIVideoRenderParamsRotation_270 | 3 | Clockwise rotation by 0 degrees |

### TUIVideoEncoderParamsResolutionMode

Video aspect ratio mode

| Type | Value | Description |
|---|---|---|
| | | |

| | | |
|---|---|---|
| TUIVideoEncoderParamsResolutionModeLandscape | 0 | Landscape resolution, such as：TUIVideoEncoderParamsResolution_640_360 - TUIVideoEncoderParamsResolutionModeLands = 640 × 360 |
| TUIVideoEncoderParamsResolutionModePortrait | 1 | Portrait resolution, such as：TUIVideoEncoderParamsResolution_640_360 - TUIVideoEncoderParamsResolutionModePortra 360 × 640 |

### TUIVideoEncoderParamsResolution

Video resolution

| Type | Value | Description |
|---|---|---|
| TUIVideoEncoderParamsResolution_640_360 | 108 | Aspect ratio: 16:9；resolution: 640x360；recommended bitrate: 500kbps |
| TUIVideoEncoderParamsResolution_640_480 | 62 | Aspect ratio: 4:3；resolution: 640x480；recommended bitrate: 600kbps |
| TUIVideoEncoderParamsResolution_960_540 | 110 | Aspect ratio: 16:9；resolution: 960x540；recommended bitrate: 850kbps |
| TUIVideoEncoderParamsResolution_960_720 | 64 | Aspect ratio: 4:3；resolution: 960x720；recommended bitrate: 1000kbps |
| TUIVideoEncoderParamsResolution_1280_720 | 112 | Aspect ratio: 16:9；resolution: 1280x720；recommended bitrate: 1200kbps |
| TUIVideoEncoderParamsResolution_1920_1080 | 114 | Aspect ratio: 16:9；resolution: 1920x1080；recommended bitrate: 2000kbps |

# ErrorCode

Last updated：2023-07-04 15:46:19

Notify users of warnings and errors that occur during audio and video calls.

### TUICallDefine

| Definition | Value | Description |
| --- | --- | --- |
| ERROR_PACKAGE_NOT_PURCHASED | -1001 | You do not have TUICallKit package, please open the free experience in the console or purchase the official package |
| ERROR_PACKAGE_NOT_SUPPORTED | -1002 | The package you purchased does not support this ability. You can refer to console to purchase Upgrade package |
| ERROR_TIM_VERSION_OUTDATED | -1003 | The IM SDK version is too low, please upgrade the IM SDK version to >= 6.6; |
| ERROR_PERMISSION_DENIED | -1101 | Failed to obtain permission. The audio/video permission is not authorized. Check if the device permission is enabled. |
| ERROR_INIT_FAIL | -1201 | The init method has not been called for initialization. The TUICallEngine API should be used after initialization. |
| ERROR_PARAM_INVALID | -1202 | param is invalid. |
| ERROR_REQUEST_REFUSED | -1203 | The current status can't use this function. |
| ERROR_REQUEST_REPEATED | -1204 | The current status is waiting/accept, please do not call it repeatedly. |
| ERROR_SCENE_NOT_SUPPORTED | -1205 | The current calling scene does not support this feature. |
| ERROR_SIGNALING_SEND_FAIL | -1406 | Failed to send signaling. You can check the specific error message in the callback of the method. |

### IM Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for communication, such as the core logic of call signaling and busy signaling. Common error codes are as follows:

| 错误码 | Description |
| --- | --- |
| 6014 | You have not logged in to the Chat SDK or have been forcibly logged out. Log in to the Chat SDK first and try again after a successful callback. To check whether you are online, use TIMManager getLoginUser. |
| 6017 | Invalid parameter. Check the error information to locate the invalid parameter. |
| 6206 | UserSig has expired. Get a new valid UserSig and log in again. For more information about how to get a UserSig, see Generating UserSig. |
| 7013 | The current package does not support this API. Please upgrade to the Flagship Edition package. |
| 8010 | The signaling request ID is invalid or has been processed. |

**Explanation：**

More IM SDK error codes are available at：IM Error Code

## TRTC Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for calling, such as the core logic of switching camera and microphone on or off. Common error codes are as follows:

| 枚举 | Value | Description |
| --- | --- | --- |
| ERR_CAMERA_START_FAIL | -1301 | Failed to turn the camera on. This may occur when there is a problem with the camera configuration program (driver) on Windows or macOS. Disable and reenable the camera, restart the camera, or update the configuration program. |
| ERR_CAMERA_NOT_AUTHORIZED | -1314 | Failed to turn the camera on. This may occur when there is a problem with the camera configuration program (driver) on Windows or macOS. Disable and reenable the camera, restart the camera, or update the configuration program. |
| ERR_CAMERA_SET_PARAM_FAIL | -1315 | Incorrect camera parameter settings (unsupported values or others). |
| ERR_CAMERA_OCCUPY | -1316 | The camera is being used. Try another camera. |
| ERR_MIC_START_FAIL | -1302 | Failed to turn the mic on. This may occur when there is a problem with the mic configuration program (driver) on Windows or macOS. Disable and |

| | | |
|---|---|---|
| | | reenable the mic, restart the mic, or update the configuration program. |
| ERR_MIC_NOT_AUTHORIZED | -1317 | No permission to access to the mic. This usually occurs on mobile devices and may be because the user denied access. |
| ERR_MIC_SET_PARAM_FAIL | -1318 | Failed to set mic parameters. |
| ERR_MIC_OCCUPY | -1319 | The mic is being used. The mic cannot be turned on when, for example, the user is having a call on the mobile device. |
| ERR_TRTC_ENTER_ROOM_FAILED | -3301 | Failed to enter the room. For the reason, refer to the error message for -3301. |

**Explanation：**

More TRTC SDK error codes are available at: TRTC Error Code

# Web
# API Overview

Last updated：2023-07-13 16:55:53

## TUICallKit（UI Component）

`<TUICallKit/>` : The call component.

`<TUICallKitMini/>` : The floating call window. If `<TUICallKit/>allowedMinimized` is `true` , `<TUICallKitMini/>` is required.

`TUICallKitServer` : The call instance, which has the following APIs:

init: Initializes `TUICallKit` .

call: Makes a one-to-one call.

groupCall: Makes a group call.

destroyed: Terminates `TUICallKit` .

## TUICallEngine (No UI)

`TUICallEngine` is an audio/video call component that **does not include UI elements**. You can use its APIs to customize your project.

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallEngine` instance (singleton mode). |
| destroyInstance | Terminates a `TUICallEngine` instance (singleton mode). |
| on | Listens for events. |
| off | Stops listening for events. |
| login | Logs in. |
| logout | Logs out. |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |

| | |
|---|---|
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| switchCallingType | Changes the call type. |
| startRemoteView | Starts rendering a remote video. |
| stopRemoteView | Stops rendering a remote video. |
| startLocalView | Starts rendering the local video. |
| stopLocalView | Stops rendering the local video. |
| openCamera | Turns the camera on. |
| closeCamara | Turns the camera off. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| setMicMute | Sets whether to mute the mic. |
| setVideoQuality | Sets the video quality. |
| getDeviceList | Gets the device list. |
| switchDevice | Changes to a different camera/mic. |
| enableAIVoice | open/close AI noise reduction |

## Event Types

`TUICallEvent` is the callback class of `TUICallEngine`. You can use it to listen for events.

| Event | Description |
|---|---|
| TUICallEvent.ERROR | An internal error occurred. |
| TUICallEvent.SDK_READY | The SDK is ready. |
| TUICallEvent.KICKED_OUT | The current user was removed from the room due to repeated login. |
| TUICallEvent.USER_ACCEPT | A user accepted the call. |

| | |
|---|---|
| TUICallEvent.USER_ENTER | A user agreed to join the call. |
| TUICallEvent.USER_LEAVE | A user agreed to leave the call. |
| TUICallEvent.REJECT | A user rejected the call. |
| TUICallEvent.NO_RESP | The invited user did not answer. |
| TUICallEvent.LINE_BUSY | The line is busy. |
| TUICallEvent.USER_VIDEO_AVAILABLE | A remote user turned on/off their camera. |
| TUICallEvent.USER_AUDIO_AVAILABLE | A remote user turned on/off their mic. |
| TUICallEvent.USER_VOICE_VOLUME | A remote user adjusted their call volume. |
| TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE | The invitation list for a group call was updated. |
| TUICallEvent.INVITED | You were invited to a call. |
| TUICallEvent.CALLING_CANCEL | The call was canceled (received by an invitee). |
| TUICallEvent.CALLING_END | The call ended. |
| TUICallEvent.DEVICED_UPDATED | The device list was updated. |
| TUICallEvent.CALL_TYPE_CHANGED | The call type changed. |

## Document

TUICallEngine
TUICallEvent
TUICallEngine API document

# TUICallKit

Last updated：2023-06-30 16:16:41

`TUICallKit` is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat. For directions on integration, see Integrating TUICallKit.

## Overview

`<TUICallKit/>` : The call component.

`<TUICallKitMini/>` : The floating call window. If `<TUICallKit/>allowedMinimized` is `true` , `<TUICallKitMini/>` is required.

`TUICallKitServer` : The call instance, which has the following APIs:

init: Initializes `TUICallKit` .

call: Makes a one-to-one call.

groupCall: Makes a group call.

destroyed: Terminates `TUICallKit` .

## `<TUICallKit/>` API details

### Attributes

| Parameter | Description | Type | Required | Default Value |
| --- | --- | --- | --- | --- |
| allowedMinimized | Whether to allow window minimization. If this is `true` , **TUICallKitMini** is required. If this is `false` , the minimize button will be hidden. | boolean | No | false |
| allowedFullScreen | Whether to allow full screen. If this is `false` , the full screen | boolean | No | true |

| | | | | |
|---|---|---|---|---|
| | button will be hidden. | | | |
| videoDisplayMode | Screen display mode | VideoDisplayMode | No | VideoDisplayMode.( |
| videoResolution | Call resolution | VideoResolution | No | VideoResolution.RE |
| beforeCalling | This is executed when the user makes or receives a call. | function(type, error) | No | - |
| afterCalling | This is executed after a call ends. | function() | No | - |
| onMinimized | This is executed when the component is minimized. | function(oldStatus, newStatus) | No | - |
| @kicked-out | The event thrown by the component will be triggered when the currently logged in user is kicked out of the login, and the call will be automatically ended | @kicked-out="handleKickedOut" | No | - |
| @status-changed | The event thrown by the component will be triggered when the call status changes, see note below for details of the call status | @status-changed="handleStatusChanged" | No | - |

**Remarks on parameters:**

1. The screen display mode `videoDisplayMode` has three values：`VideoDisplayMode.CONTAIN`、`VideoDisplayMode.COVER`、`VideoDisplayMode.FILL`, default is `VideoDisplayMode.COVER`.

`VideoDisplayMode.CONTAIN` Prioritize to ensure that the video content is displayed in its entirety. Scale the video size isometrically until one side of the video window is aligned with the viewport border. If the video size does not match the size of the display window, the video will be scaled to fill the window while maintaining the aspect ratio, and the scaled video will have a black border around it.

`VideoDisplayMode.COVER` Prioritize to ensure that the viewport is filled. The video is scaled equally in size until the entire viewport is filled with video. If the video is different in length and width from the display window, the video stream will fill the window with a peripheral crop or image stretch in the same proportion as the display window.

`VideoDisplayMode.FILL` Ensures that the viewport is filled and the video content is displayed in its entirety, but does not guarantee that the video size remains proportional. The width and height of the video will be stretched to match the size of the viewport.

2. `videoResolution` has three values：`VideoResolution.RESOLUTION_480P`、`VideoResolution.RESOLUTION_720P`、`RESOLUTION_1080P.FILL`, default is `VideoResolution.RESOLUTION_480P`.

| video Profile | resolution（width × height） | Frame Rate（fps） | Code Rate（kbps） |
|---|---|---|---|
| 480p | 640 × 480 | 15 | 900 |
| 720p | 1280 × 720 | 15 | 1500 |
| 1080p | 1920 × 1080 | 15 | 2000 |

Firefox：No support for custom video frame rates（default is 30fps）。

The actual values of video resolution, frame rate and bit rate may not always match the set values due to system performance, camera capture capability and browser limitations, in which case the browser will automatically adjust the Profile to match the set values as closely as possible.

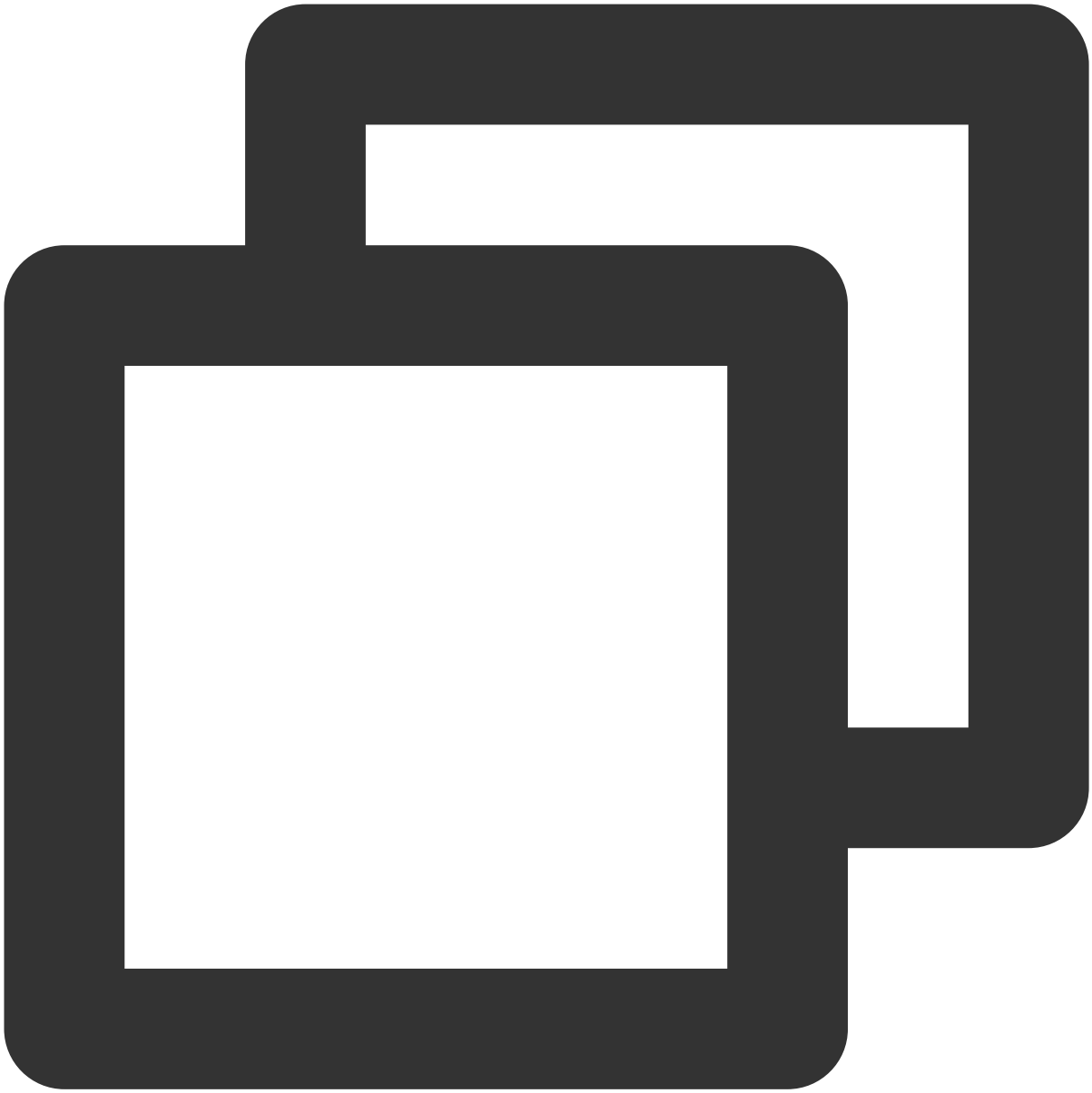
3. `@status-changed` The state inside the call component can be thrown, `handleStatusChanged({ oldStatus, newStatus })` receive the new state and the old state, the use of reference to the following sample code, where the state has the following kinds of:

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

| Constants | Physical meaning |
|---|---|
| STATUS.IDLE | Idle status |
| STATUS.BE_INVITED | Receive a call invitation |
| | |

| | |
|---|---|
| STATUS.DIALING_C2C | Calling one-to-one |
| STATUS.DIALING_GROUP | Group call in progress |
| STATUS.CALLING_C2C_AUDIO | Ongoing one-to-one audio calls |
| STATUS.CALLING_C2C_VIDEO | Ongoing one-to-one video calls |
| STATUS.CALLING_GROUP_AUDIO | Group audio call in progress |
| STATUS.CALLING_GROUP_VIDEO | Group video call in progress |

### Sample Code



```
<TUICallKit
  :allowedMinimized="true"
  :allowedFullScreen="true"
  :videoDisplayMode="VideoDisplayMode.CONTAIN"
  :videoResolution="VideoResolution.RESOLUTION_1080P"
  :beforeCalling="beforeCalling"
  :afterCalling="afterCalling"
  :onMinimized="onMinimized"
  @kicked-out="handleKickedOut"
  @status-changed="handleStatusChanged"
/>
```

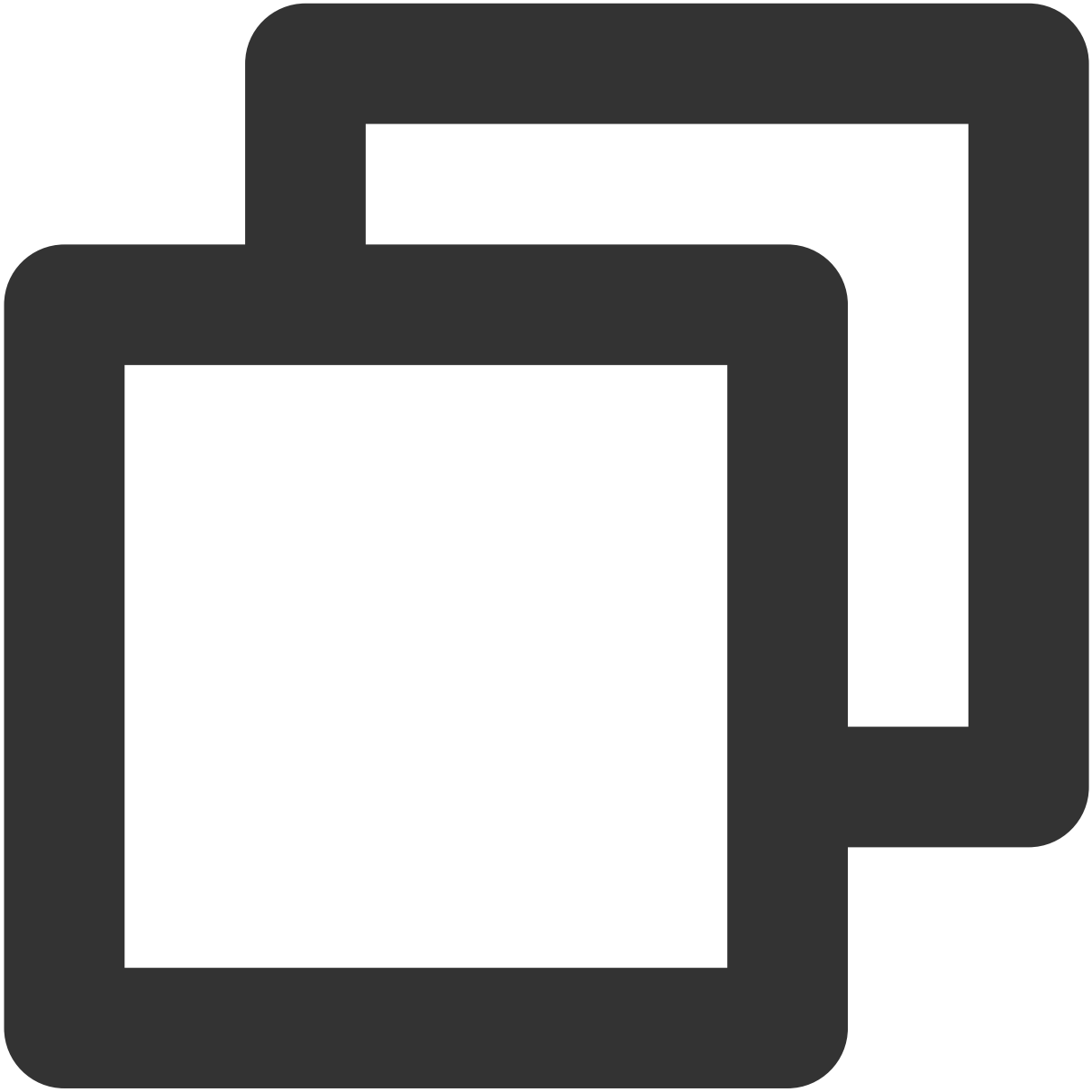
```
<TUICallKitMini />
```

```javascript
import { TUICallKit, TUICallKitMini, TUICallKitServer, VideoDisplayMode, VideoResol

/**
* beforeCalling
 * @param { string } type value is: "invited" | "call" | "groupCall",
 * @param { number } error.code
 * @param { string } error.type
 * @param { string } error.code
 */
function beforeCalling(type, error) {
  console.log(type, error);
}

function afterCalling() {
  console.log("--");
}

/**
 * onMinimized
 * @param { boolean } oldStatus
 * @param { boolean } newStatus
 */
function onMinimized(oldStatus, newStatus) {
  if (newStatus === true) {
    console.log("TUICallKit enter minimized state");
  } else {
    console.log("TUICallKit exit minimized state");
  }
}

function handleKickedOut() {
  console.error("The user has been kicked out");
}

function handleStatusChanged(args: { oldStatus: string; newStatus: string; }) {
  const { oldStatus, newStatus } = args;
  if (newStatus === STATUS.CALLING_C2C_VIDEO) {
    console.log(`[Call Demo] state change: ${oldStatus} -> ${newStatus}`);
  }
}
```

# TUICallKitServer API details

### import `TUICallKitServer`

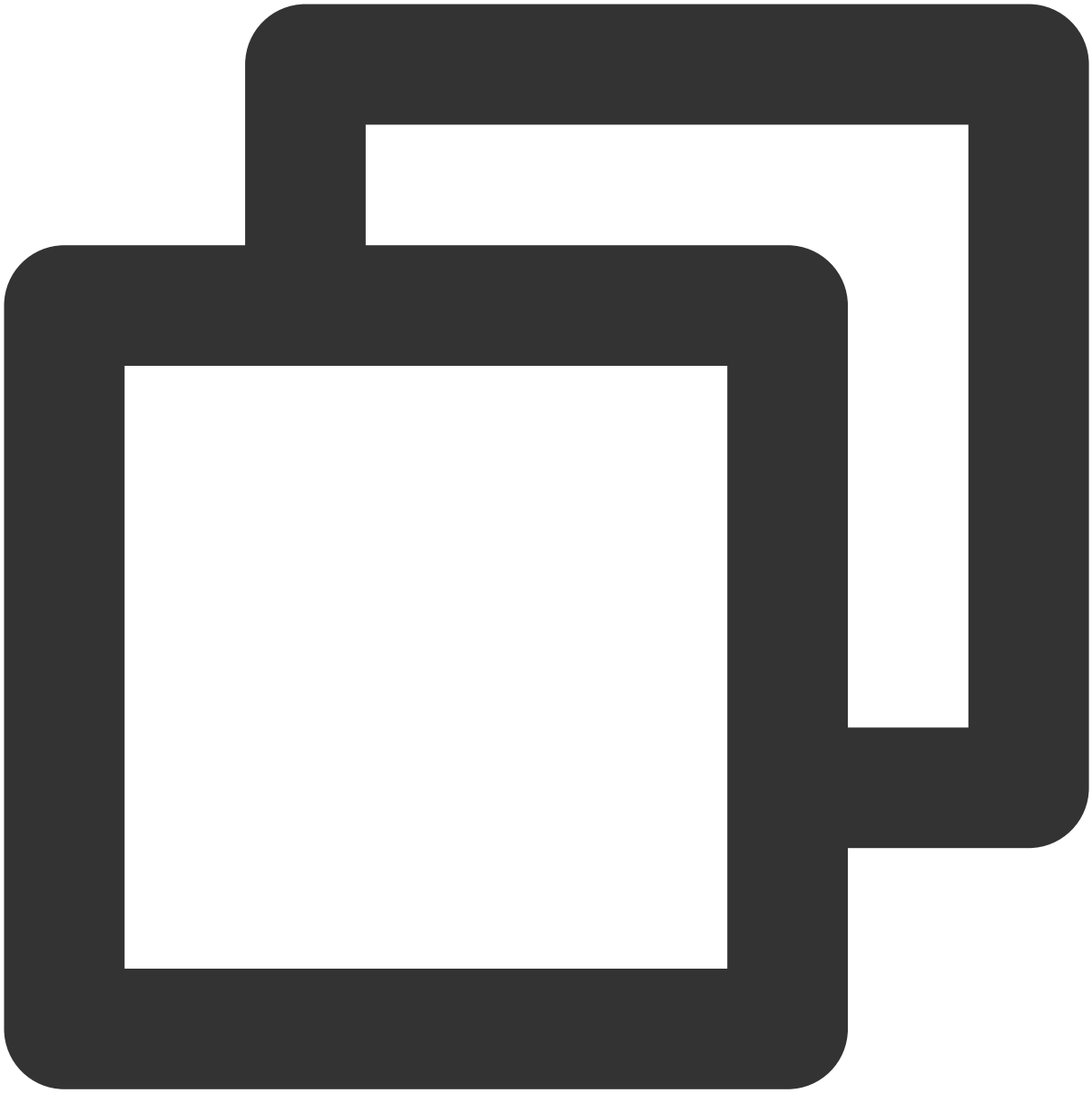
vue3



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

### init

This API is used to initialize `TUICallKit` . It must be called before `call` and `groupCall` .



```
try {
  await TUICallKitServer.init({ SDKAppID, userID, userSig });
  // await TUICallKitServer.init({ tim, SDKAppID, userID, userSig});
  alert("init finished");
} catch (error: any) {
  alert(`init failed, reason is：${error}`);
}
```

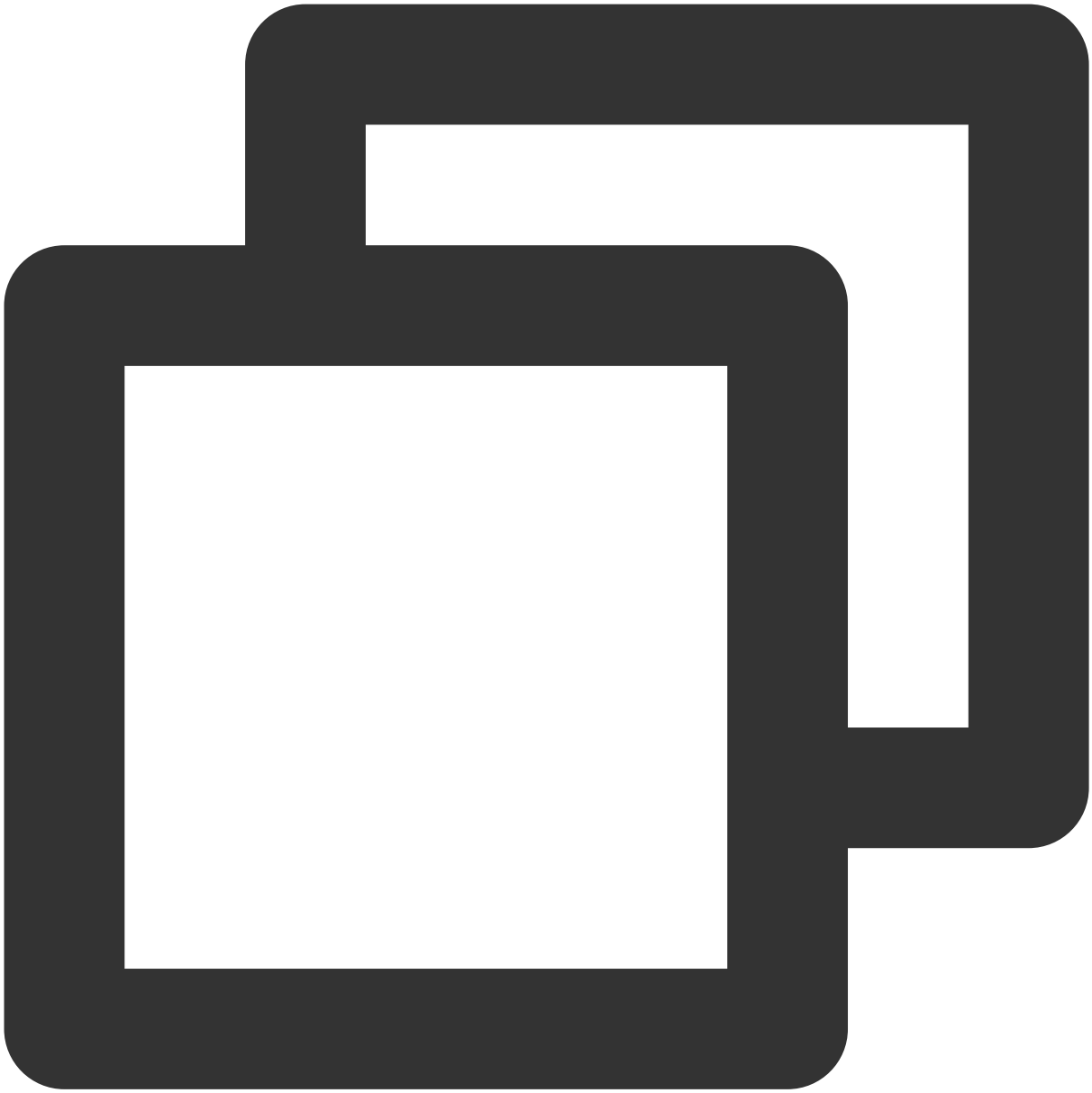
The parameters are described below:

| Parameter | Type | Required | Description |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| SDKAppID | Number | Yes | The SDKAppID of the IM application. |
| userID | String | Yes | The ID of the current user, which is a string that can contain only letters (a-z and A-Z), numbers (0-9), hyphens (-), and underscores (_). |
| userSig | String | Yes | Tencent Cloud's proprietary security signature. For how to obtain it, see UserSig. |
| TIM instance | Any | No | If you already have TIM instances, you can use this parameter to guarantee the uniqueness of a TIM instance. |

### call

This API is used to make a one-to-one call.



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.call({
  userID: 'jack',
  type: 1,
});
```

The parameters are described below:

| Parameter | Type | Required | Description |
| --- | --- | --- | --- |
| userID | String | Yes | The ID of the user to call. |

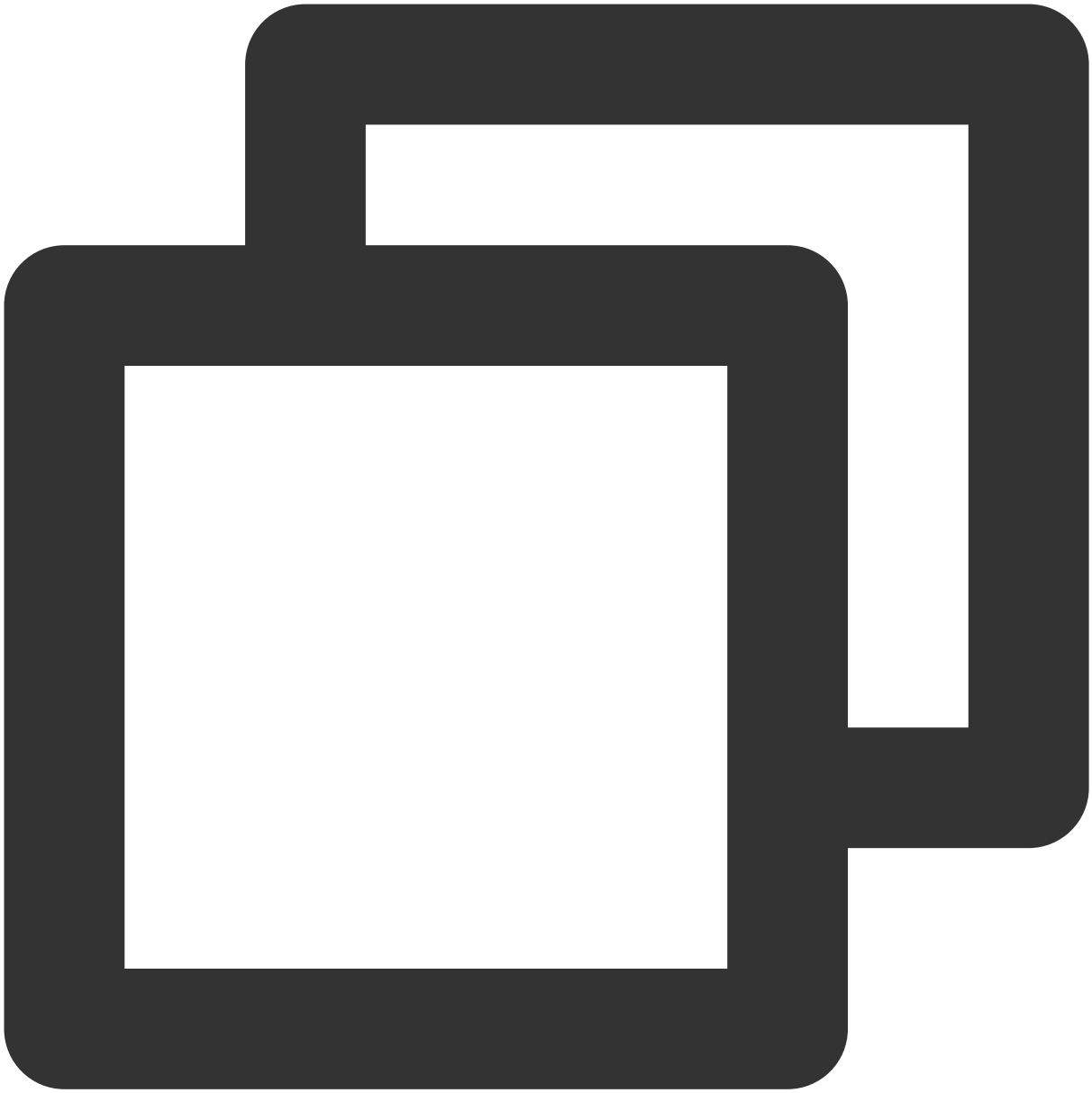
| type | Number | Yes | The call type. 1: Audio call; 2: Video call. |
| --- | --- | --- | --- |
| timeout | Number | No | The timeout period (seconds). `0` indicates the call will not time out. The default value is `30`. |
| offlinePushInfo | Object | No | The custom offline push message. This is valid only if your `tsignaling` version is 0.8.0 or later. |

offlinePushInfo

| Parameter | Type | Required | Description |
| --- | --- | --- | --- |
| offlinePushInfo.title | String | No | The message title. |
| offlinePushInfo.description | String | No | The message content. |
| offlinePushInfo.androidOPPOChannelID | String | No | The channel ID for the offline push message on OPPO phones in v8.0 or later. |
| offlinePushInfo.extension | String | No | The pass-through content. This is valid only if your `tsignaling` version is 0.9.0 or later. |

### groupCall

This API is used to make a group call.



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.groupCall({
  userIDList: ['jack', 'tom'],
  groupID: 'xxx',
  type: 1,
});
```

The parameters are described below:

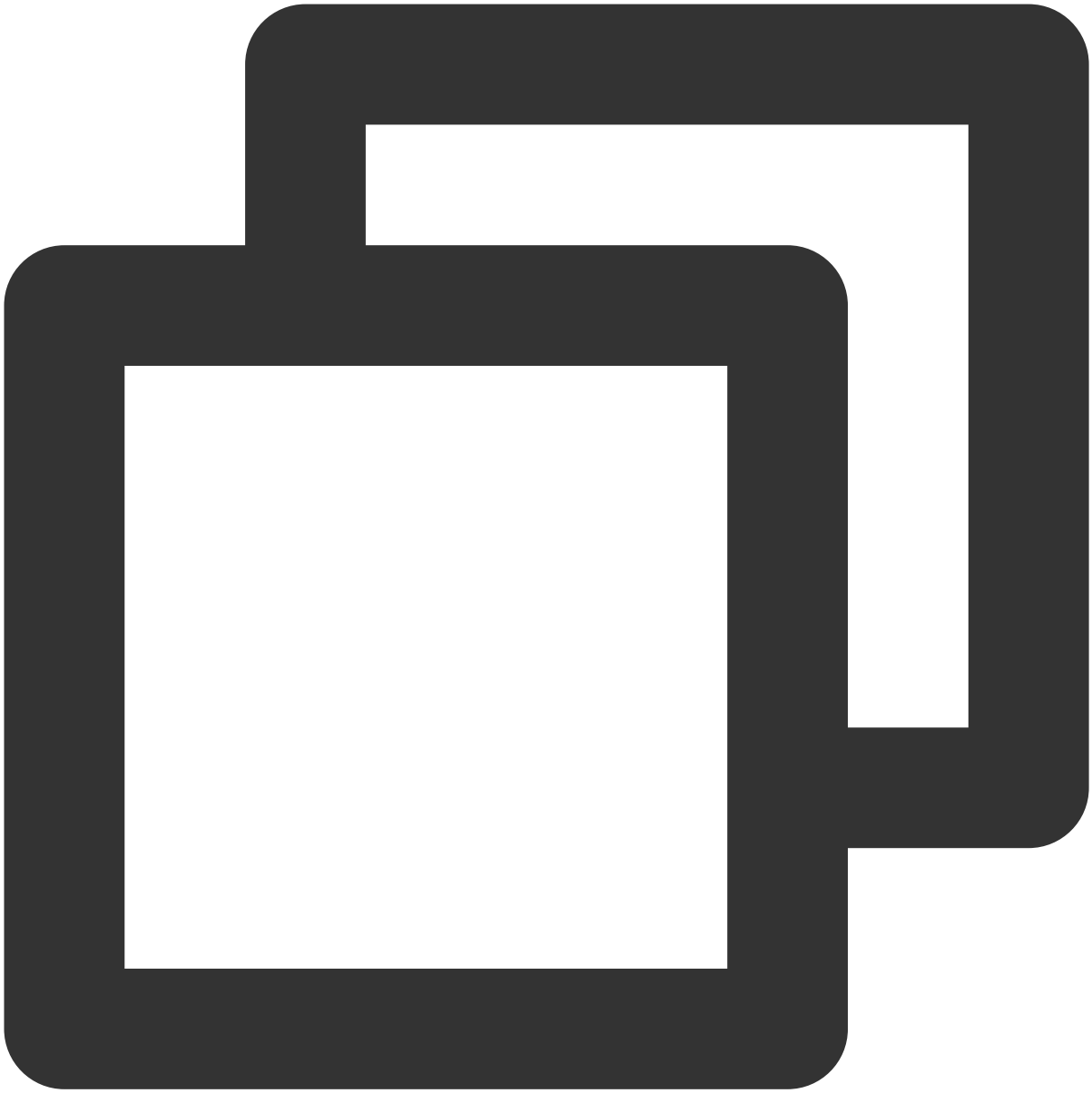
| Parameter | Type | Required | Description |
| --- | --- | --- | --- |
| | | | |

| | | | |
|---|---|---|---|
| userIDList | Array | Yes | The IDs of the users to call. |
| type | Number | Yes | The call type. 1: Audio call; 2: Video call. |
| groupID | String | Yes | The group ID. |
| timeout | Number | No | The timeout period (seconds). `0` indicates the call will not time out. The default value is `30` . |
| offlinePushInfo | Object | No | The custom offline push message. This is valid only if your `tsignaling` version is 0.8.0 or later. |

For the parameters of `offlinePushInfo` , see above.

### setLanguage

Set language。



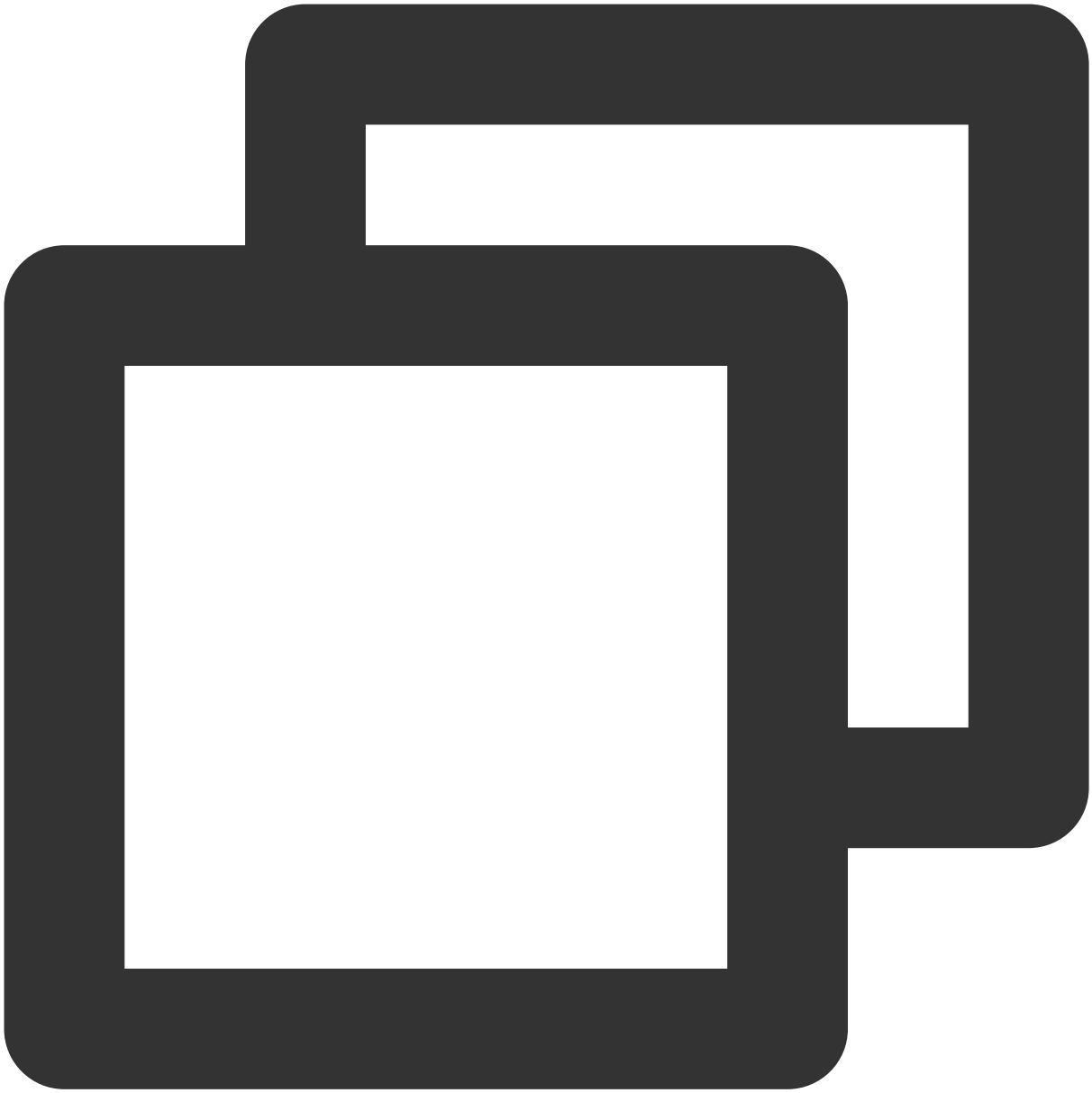
```
TUICallKitServer.setLanguage("zh-cn"); // "en" | "zh-cn"
```

The parameters are described below:

| Parameter | Type | Required | Description |
| --- | --- | --- | --- |
| lang | String | Yes | language: `en` 或 `zh-cn` |

### destroyed

This API is used to terminate `TUICallKit` .



```
import { TUICallKitServer } from "./components/TUICallKit/Web";
TUICallKitServer.destroyed();
```

# TUICallEngine

Last updated：2023-06-30 16:17:39

## TUICallEngine APIs

`TUICallEngine` is an audio/video call component that **does not include UI elements**.

## API Overview

| API | Description |
| --- | --- |
| createInstance | Creates a `TUICallEngine` instance (singleton mode). |
| destroyInstance | Terminates a `TUICallEngine` instance (singleton mode). |
| on | Listens for an event. |
| off | Stops listening for an event. |
| login | Logs in. |
| logout | Logs out. |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins the current group call. |
| switchCallMediaType | Changes the call type. |
| startRemoteView | Starts rendering a remote video. |
| | |

| | |
|---|---|
| stopRemoteView | Stops rendering a remote video. |
| startLocalView | Starts rendering the local video. |
| stopLocalView | Stops rendering the local video. |
| openCamera | Turns the camera on. |
| closeCamara | Turns the camera off. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| setVideoQuality | Sets the video quality. |
| getDeviceList | Gets the device list. |
| switchDevice | Changes to a different camera/mic. |
| enableAIVoice | Enables/Disables AI noise canceling. |

# TUICallEvent

Last updated：2023-06-30 16:20:03

## TUICallEvent APIs

`TUICallEvent` APIs are the **callback APIs** of the audio/video call component.

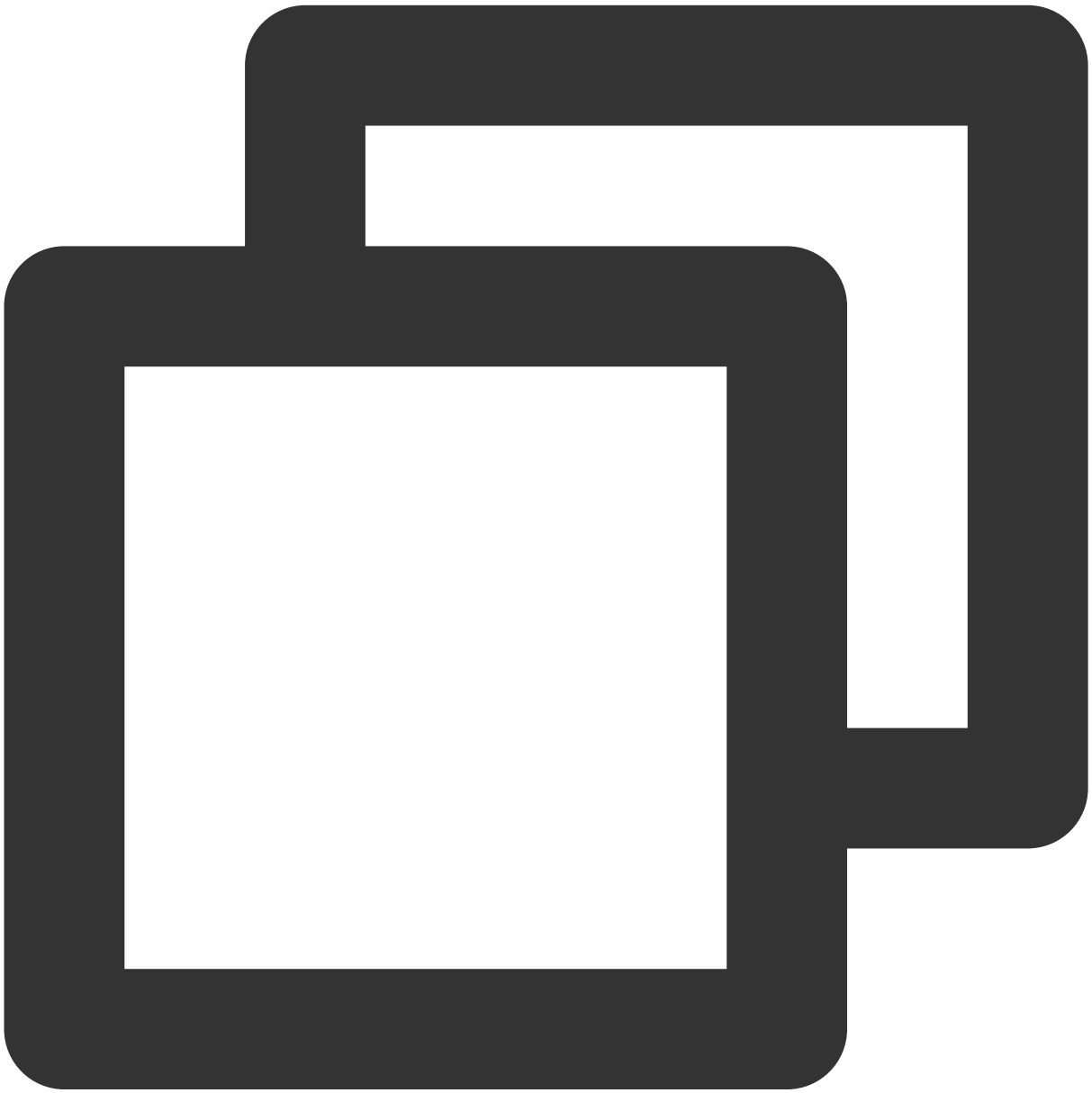
## Event List

| Event | Description |
| --- | --- |
| TUICallEvent.ERROR | An internal error occurred. |
| TUICallEvent.SDK_READY | The SDK is ready. |
| TUICallEvent.KICKED_OUT | The current user was removed from the room due to repeated login. |
| TUICallEvent.USER_ACCEPT | A user accepted the call. |
| TUICallEvent.USER_ENTER | A user joined the call. |
| TUICallEvent.USER_LEAVE | A user left the call. |
| TUICallEvent.REJECT | A user rejected the call. |
| TUICallEvent.NO_RESP | The invitee user did not answer. |
| TUICallEvent.LINE_BUSY | The line is busy. |
| TUICallEvent.USER_VIDEO_AVAILABLE | A remote user turned on/off their camera. |
| TUICallEvent.USER_AUDIO_AVAILABLE | A remote user turned on/off their mic. |
| TUICallEvent.USER_VOICE_VOLUME | A remote user adjusted their call volume. |
| TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE | The invitation list for a group call was updated. |
| TUICallEvent.INVITED | You were invited to a call. |
| TUICallEvent.CALLING_CANCEL | The call was canceled (received by an invitee). |
| TUICallEvent.CALLING_END | The call ended. |

| | |
|---|---|
| TUICallEvent.DEVICED_UPDATED | The device list was updated. |
| TUICallEvent.CALL_TYPE_CHANGED | The call type changed. |

### ERROR

An error occurred inside the SDK.

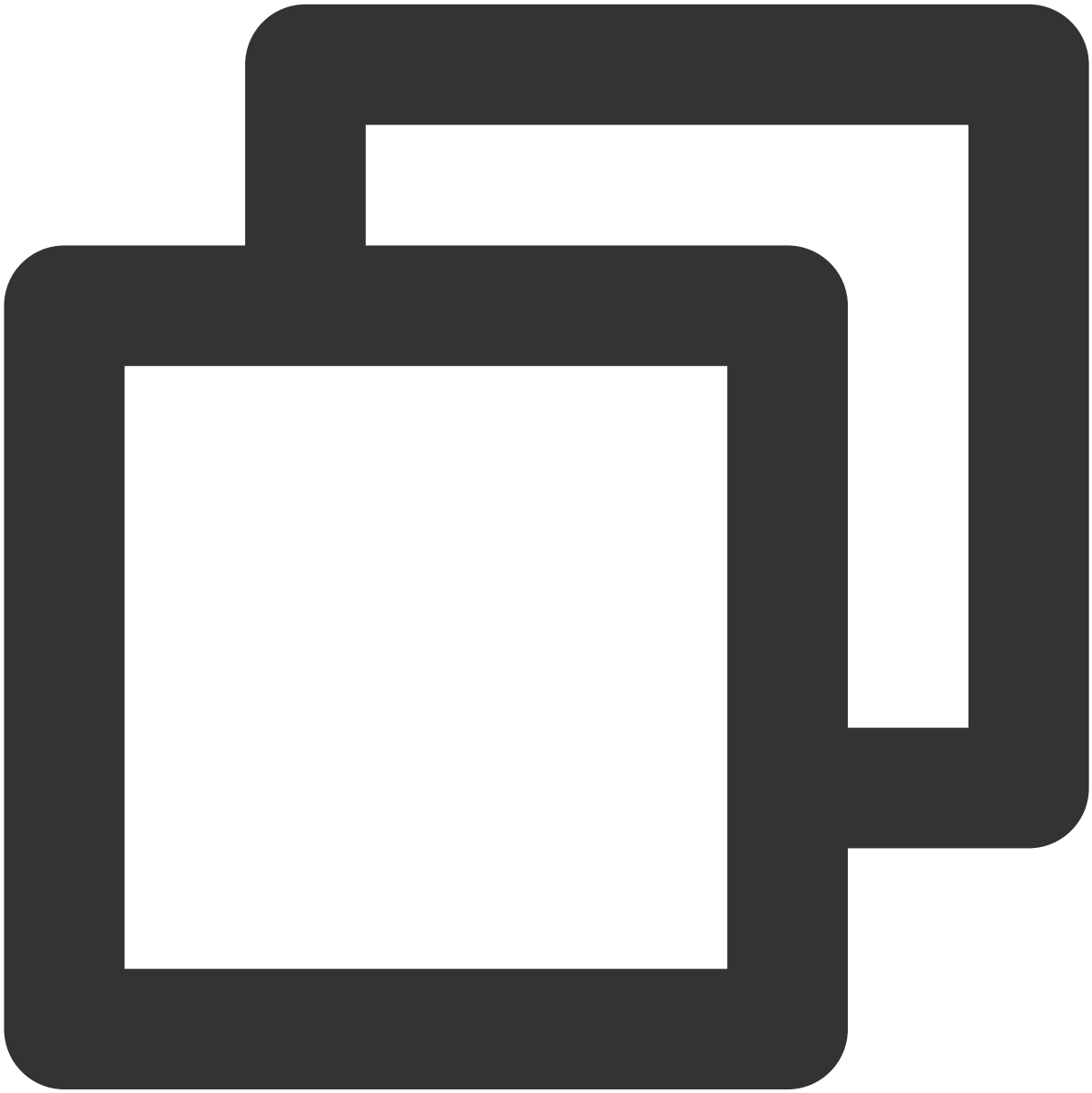


```
let onError = function(error) {
  console.log(error);
};
```

```
tuiCallEngine.on(TUICallEvent.ERROR, onError);
```

### SDK_READY

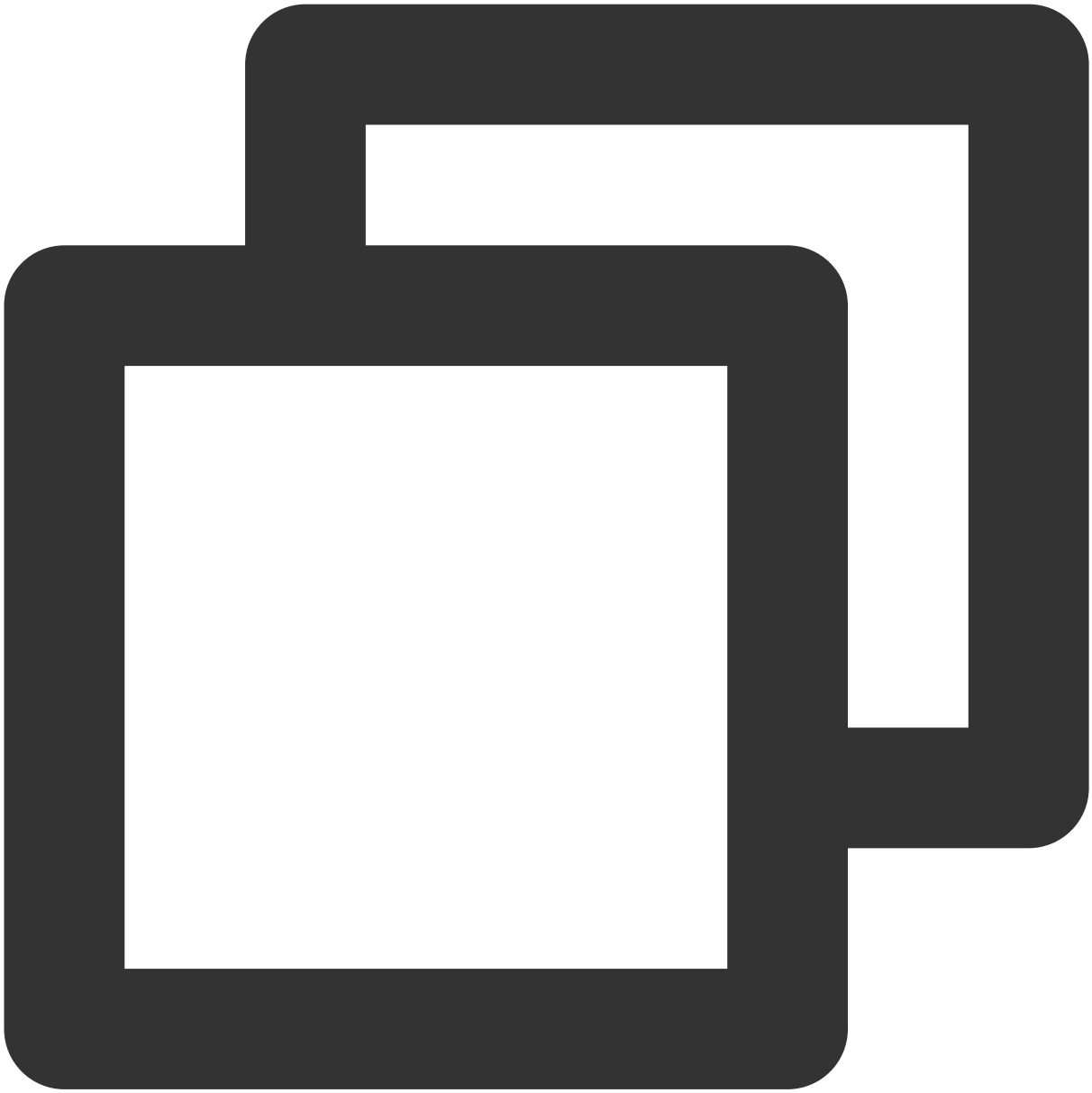
The SDK is ready.



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

### KICKED_OUT

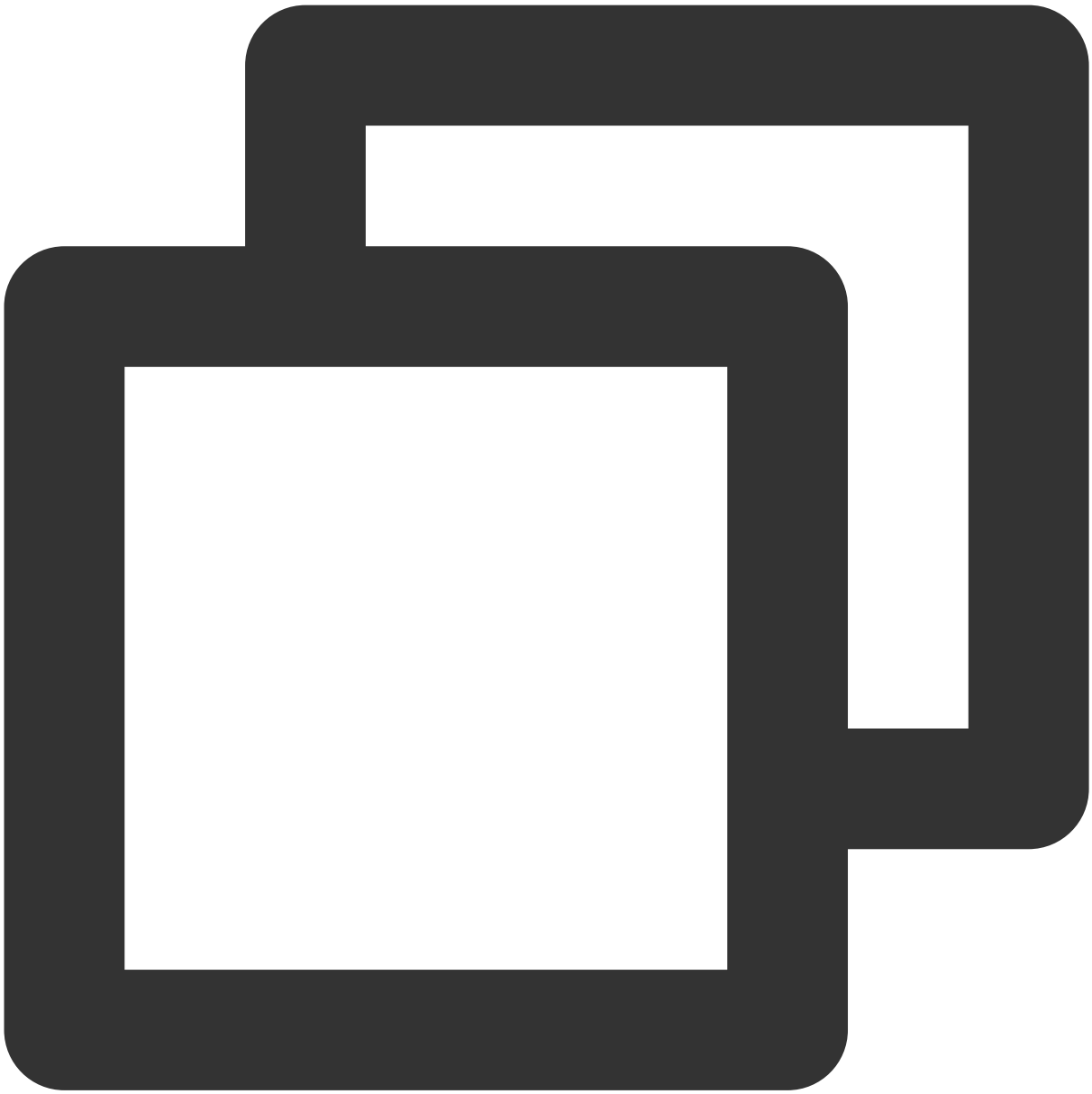
You were removed from the room due to repeated login.



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

### USER_ACCEPT

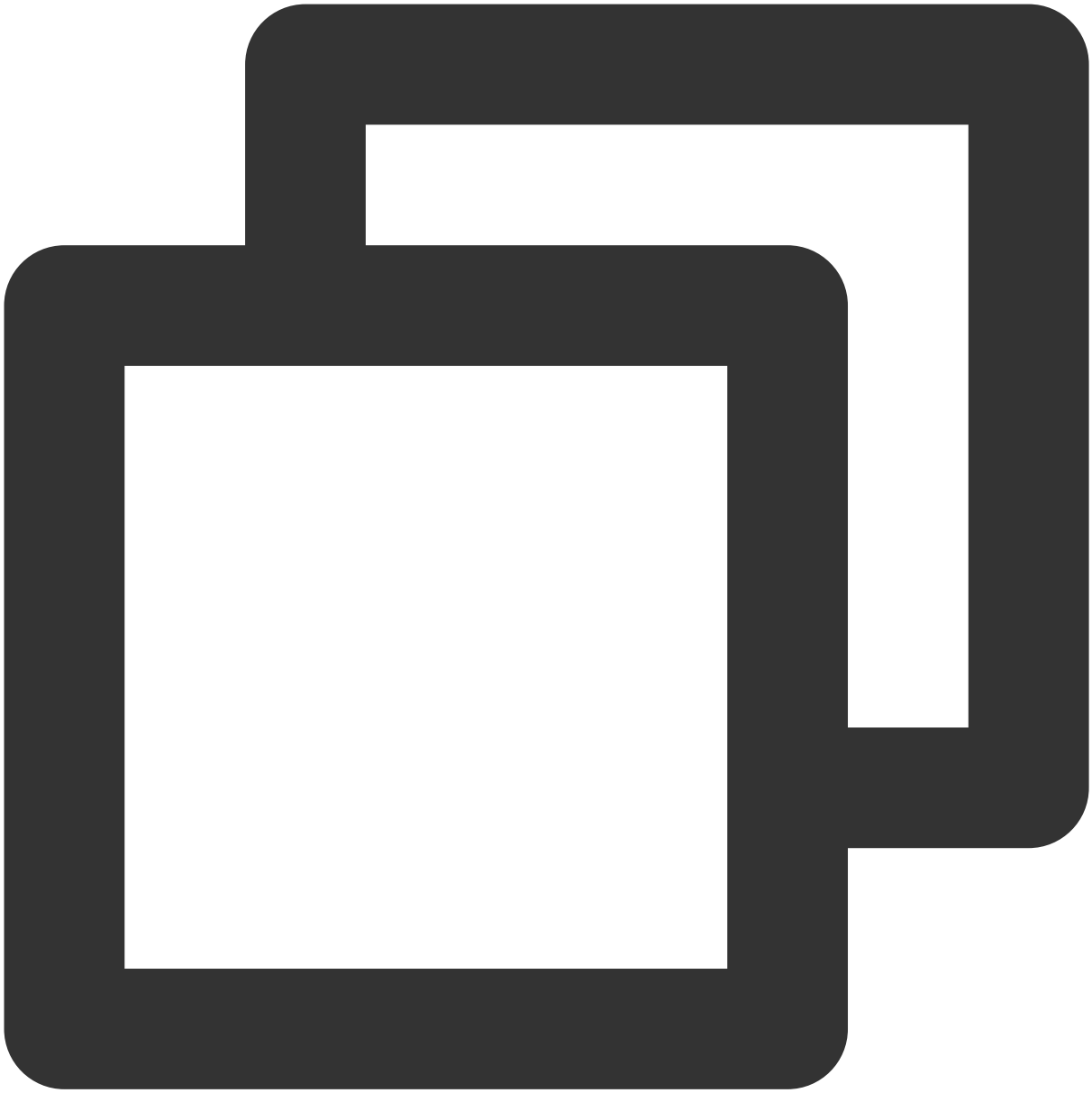
A user answered the call.



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

### USER_ENTER

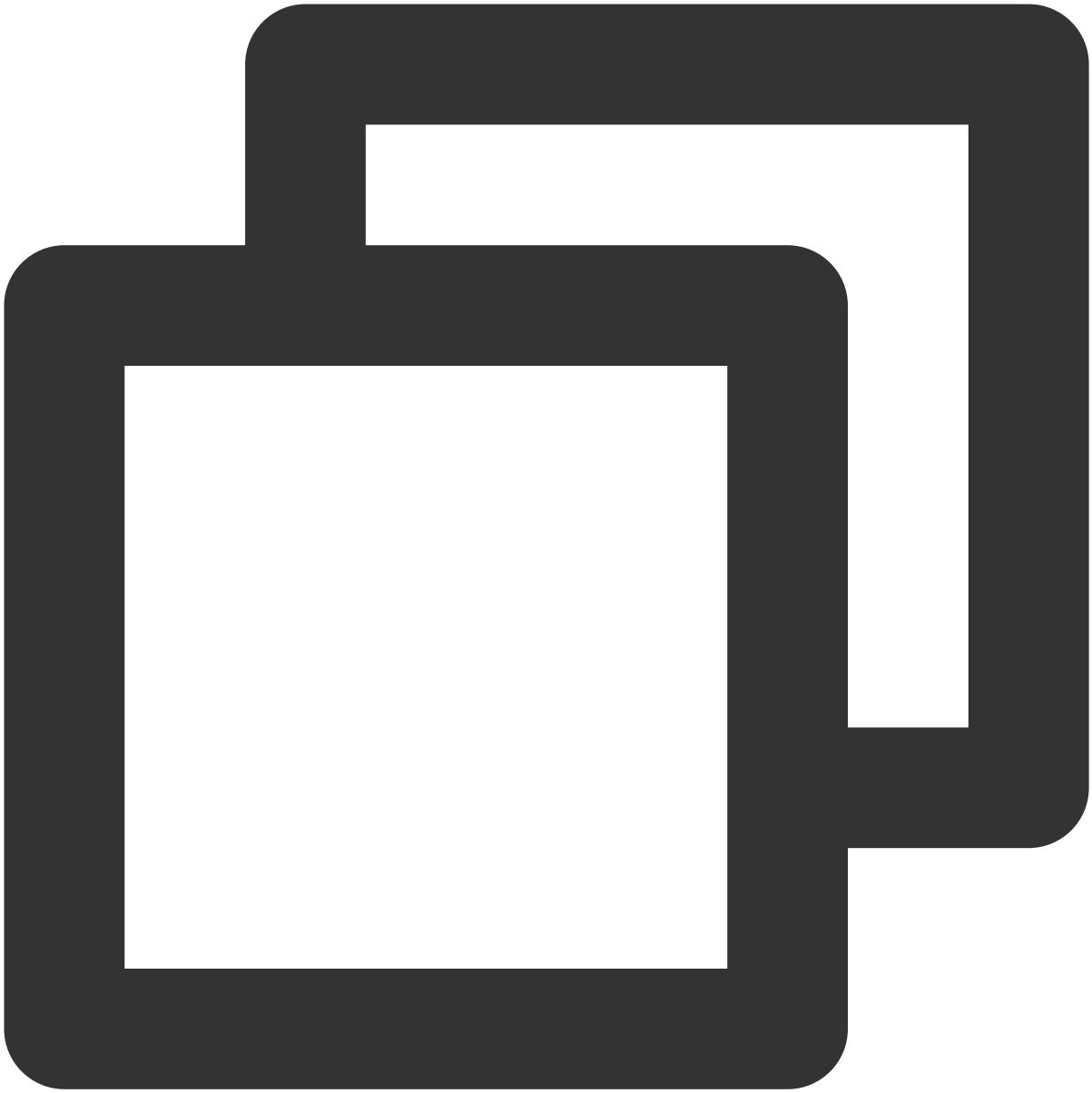
A user agreed to join the call.



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

### USER_LEAVE

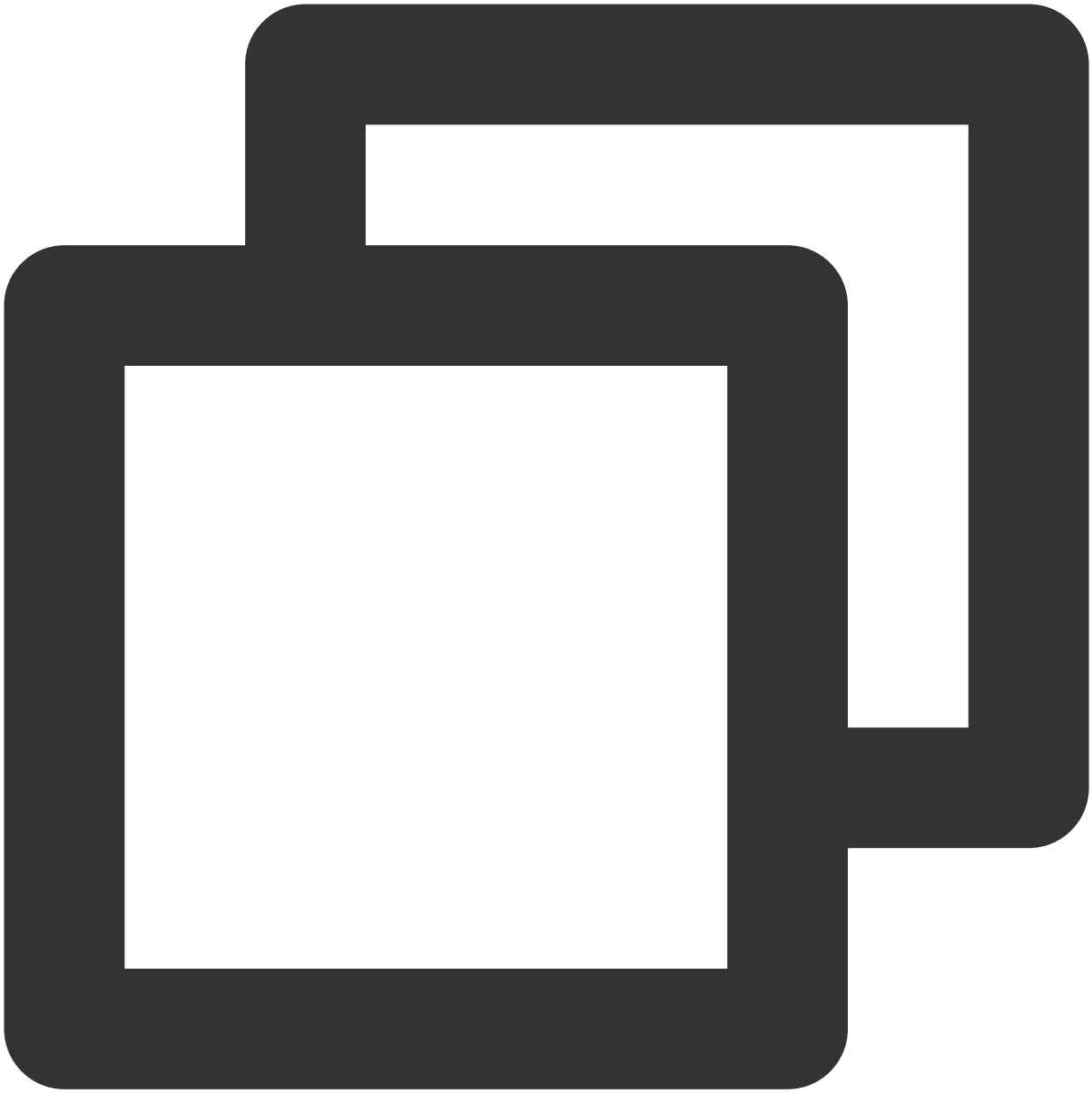
A user agreed to leave the call.



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

### REJECT

The user rejected the call.



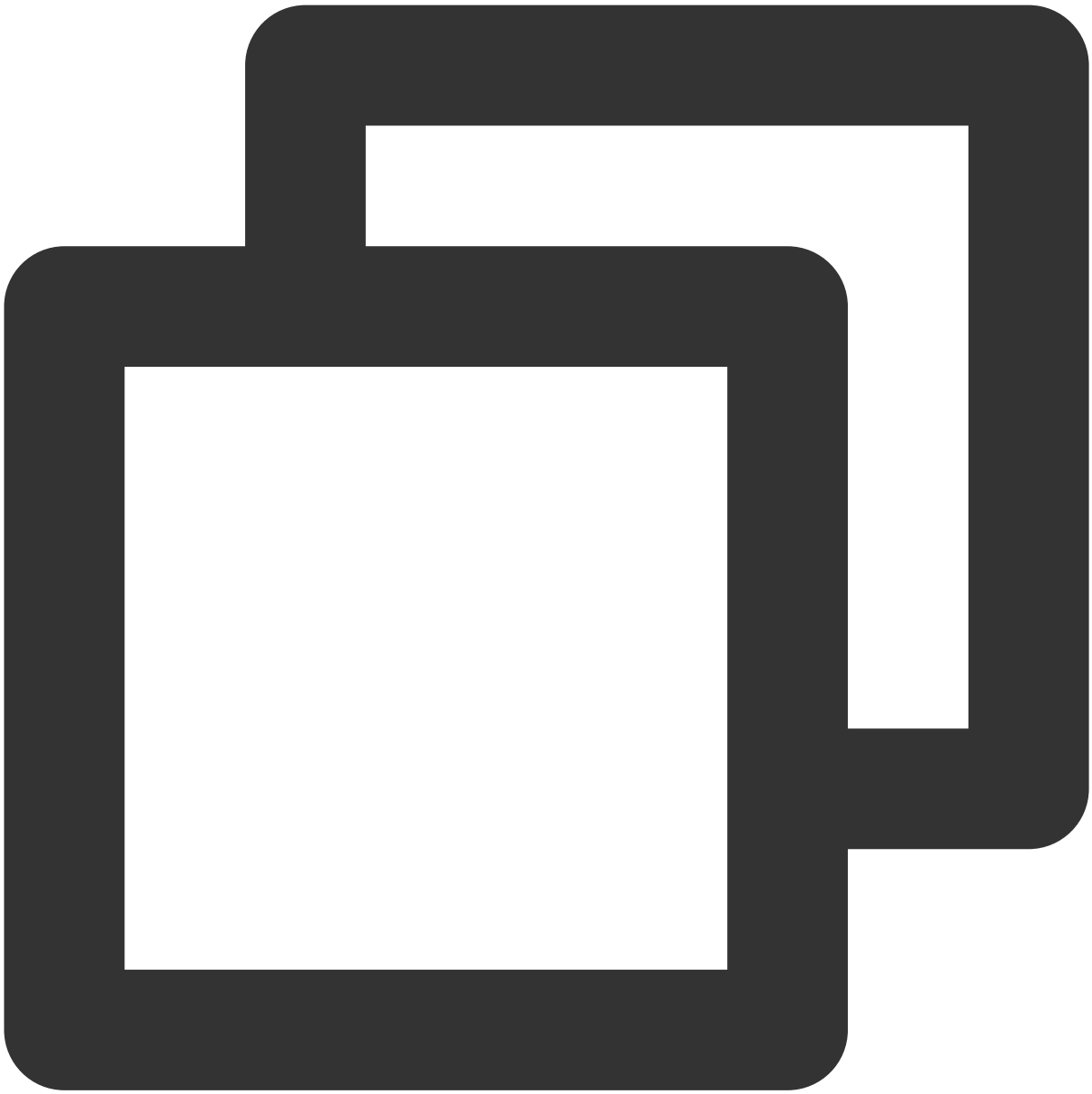
```
let handleInviteeReject = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.REJECT, handleInviteeReject);
```

### NO_RESP

The invitee did not answer.

In a one-to-one call, if the invitee does not answer, the inviter will receive this callback.

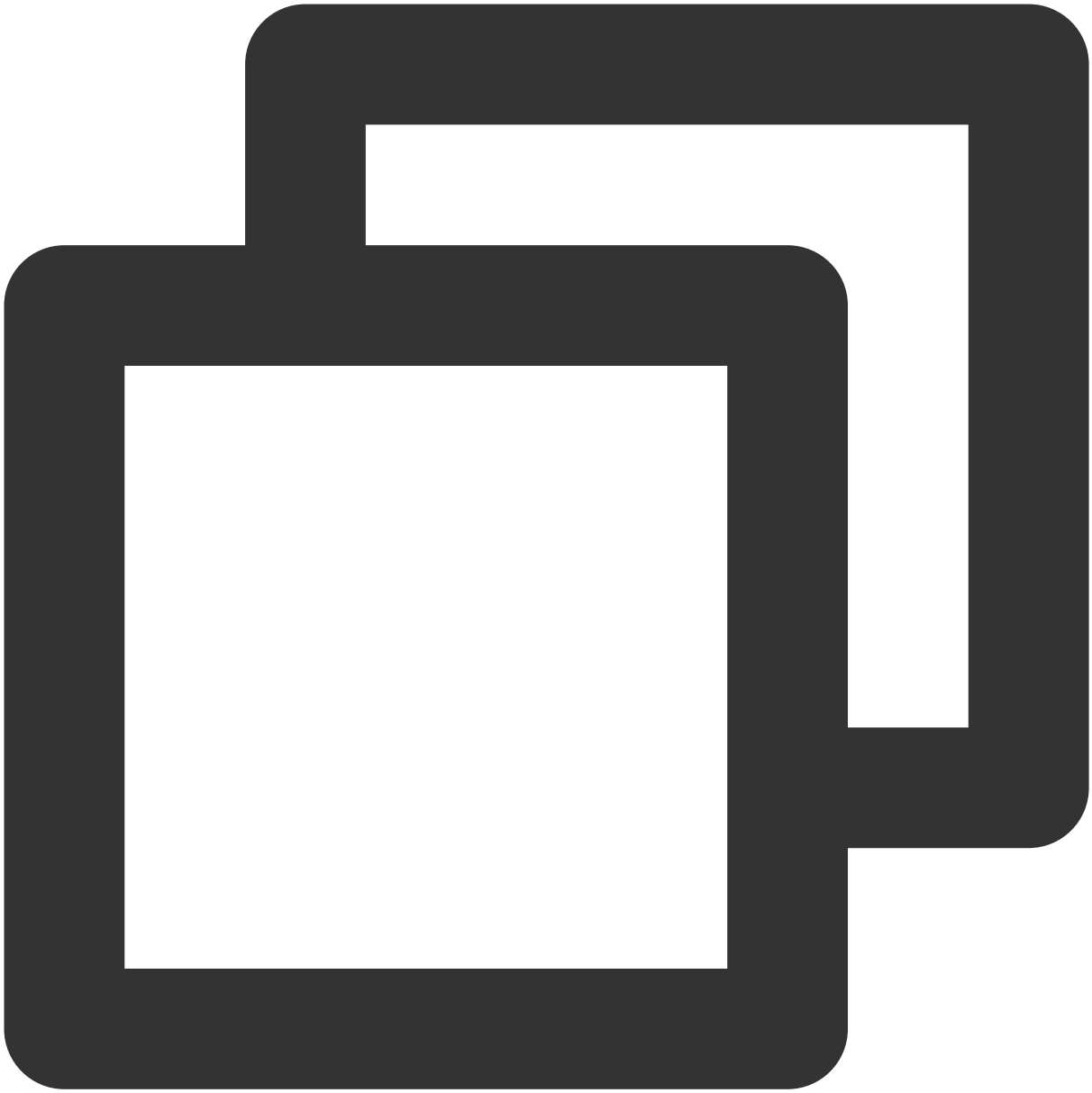
In a group call, all invitees can receive this callback. For example, if user A invited user B and user C to a group call, and B did not answer, both A and C would receive this callback.



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

### LINE_BUSY

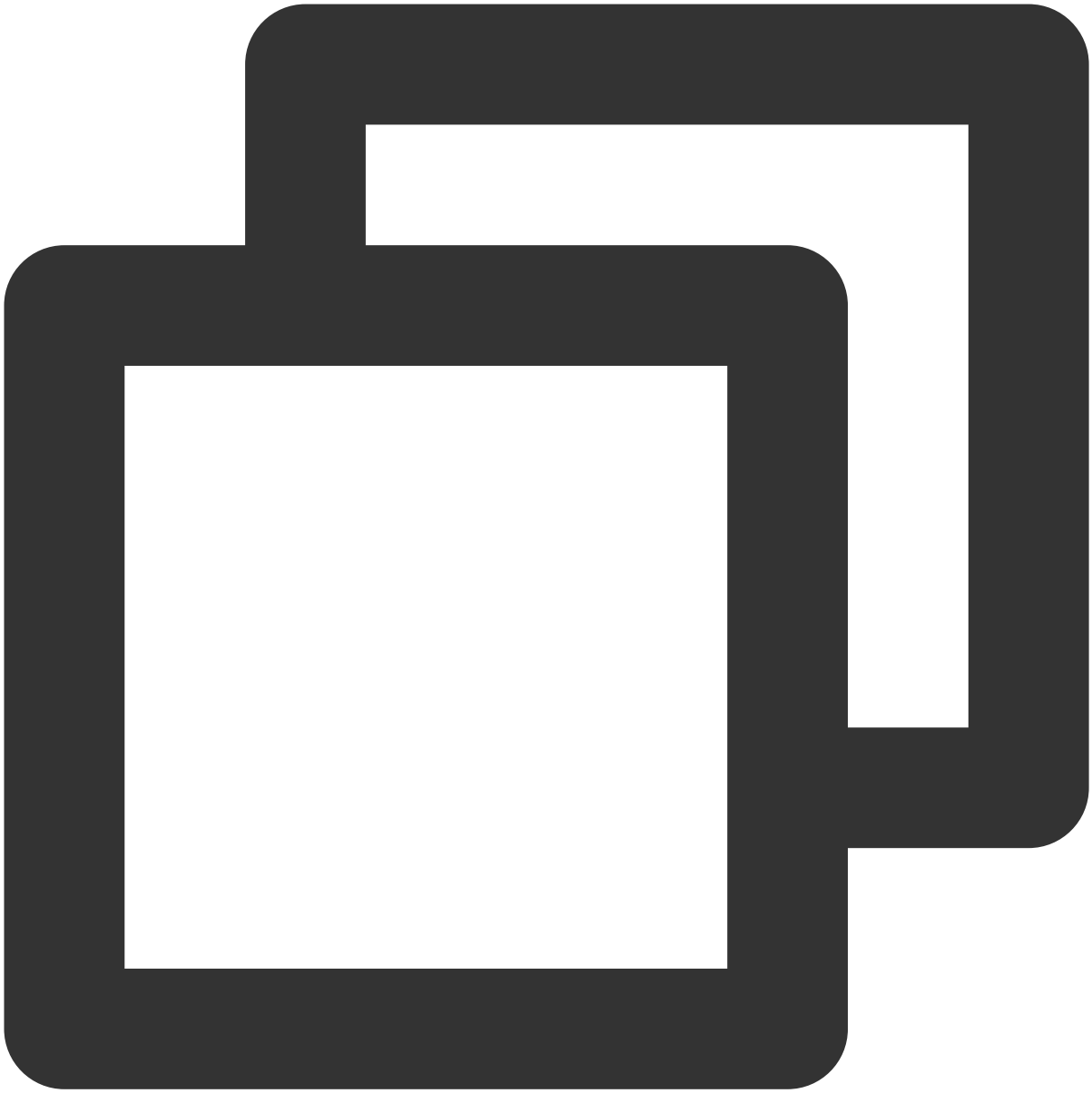
The invitee is busy.



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

### USER_VIDEO_AVAILABLE

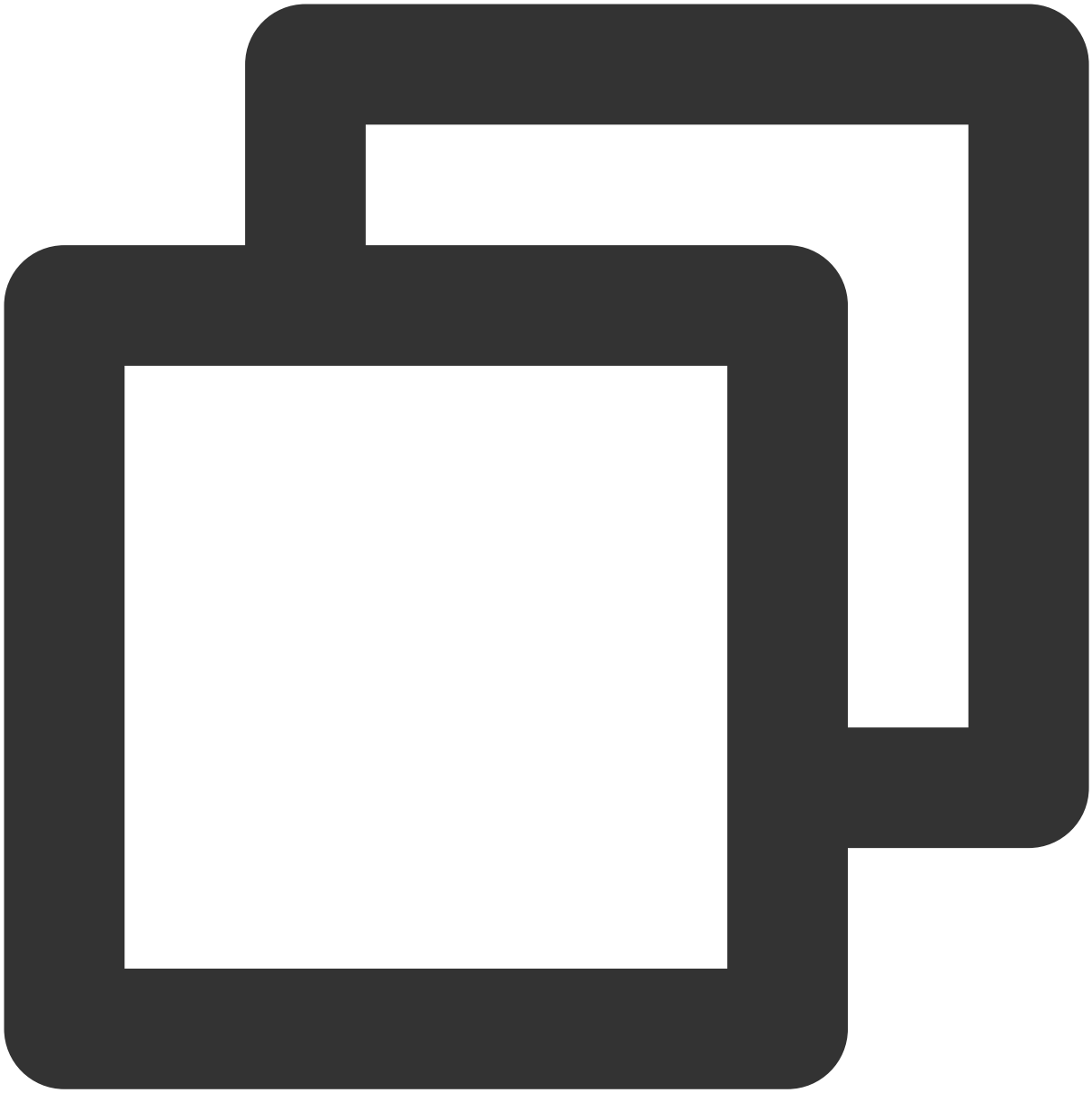
A remote user turned on/off their camera.



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

### USER_AUDIO_AVAILABLE

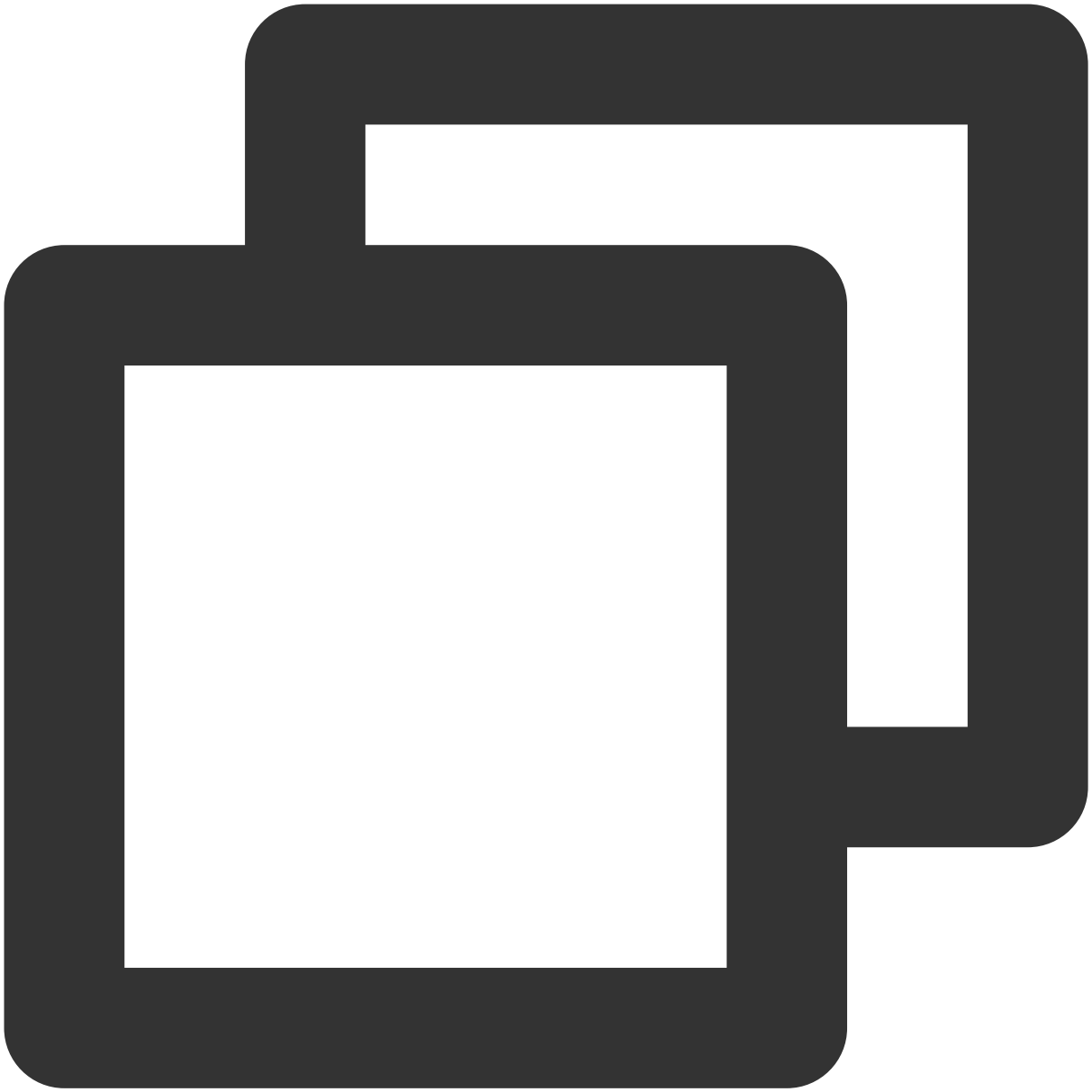
A remote user turned on/off their mic.



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

### USER_VOICE_VOLUME

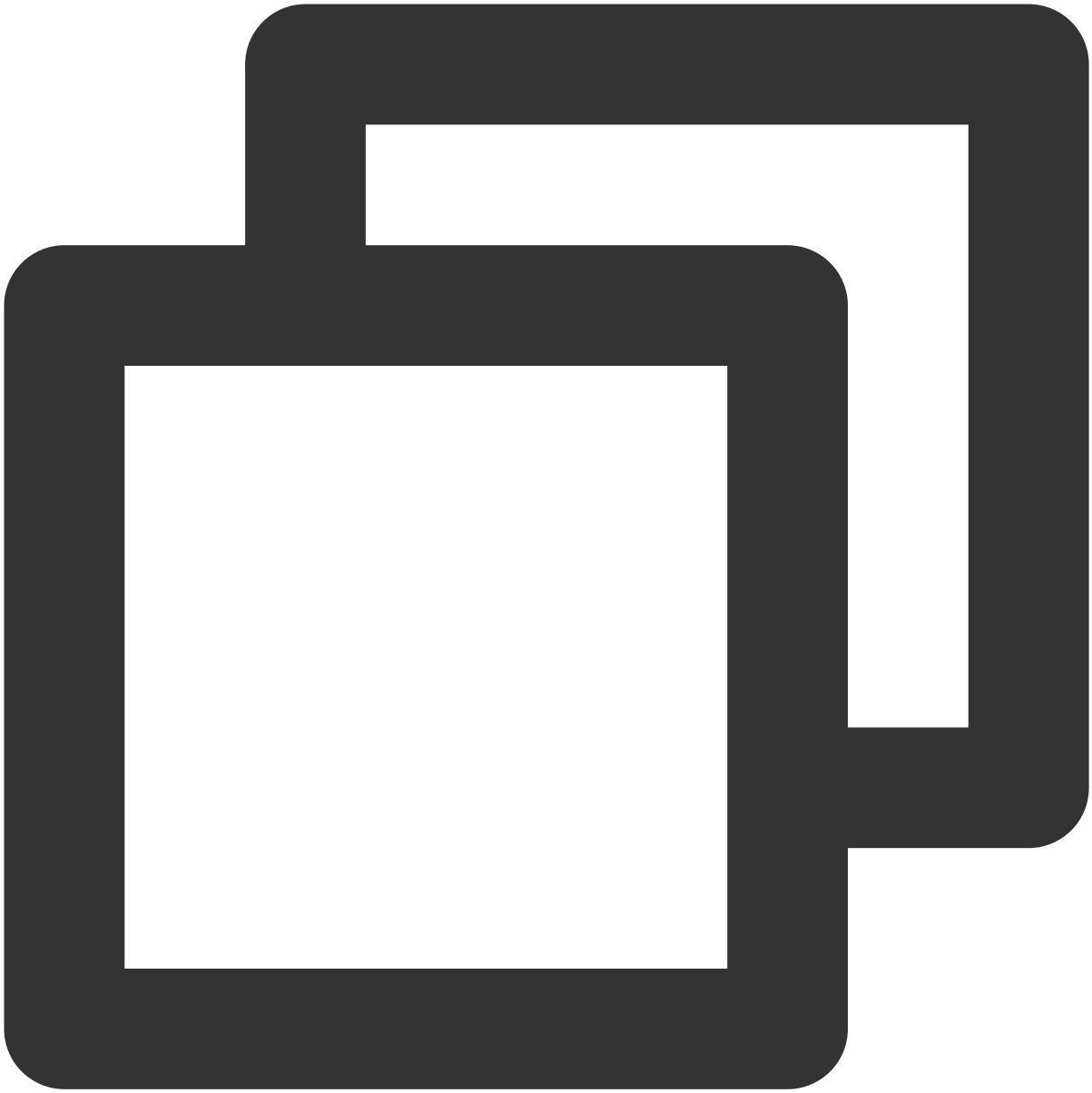
A remote user adjusted their call volume.



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

### GROUP_CALL_INVITEE_LIST_UPDATE

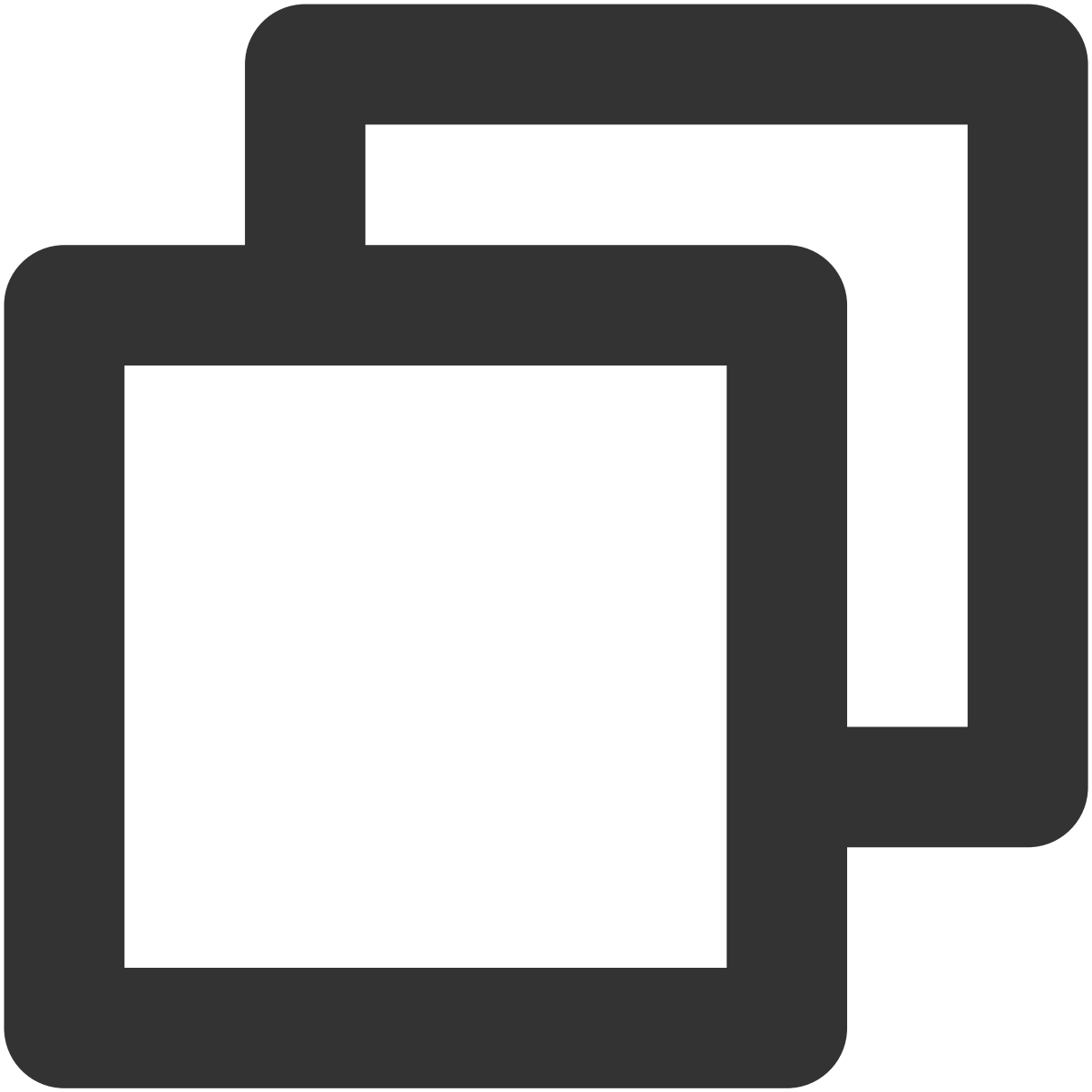
The invitee list for a group call was updated.



```
let handleGroupInviteeListUpdate = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeLis
```

### INVITED

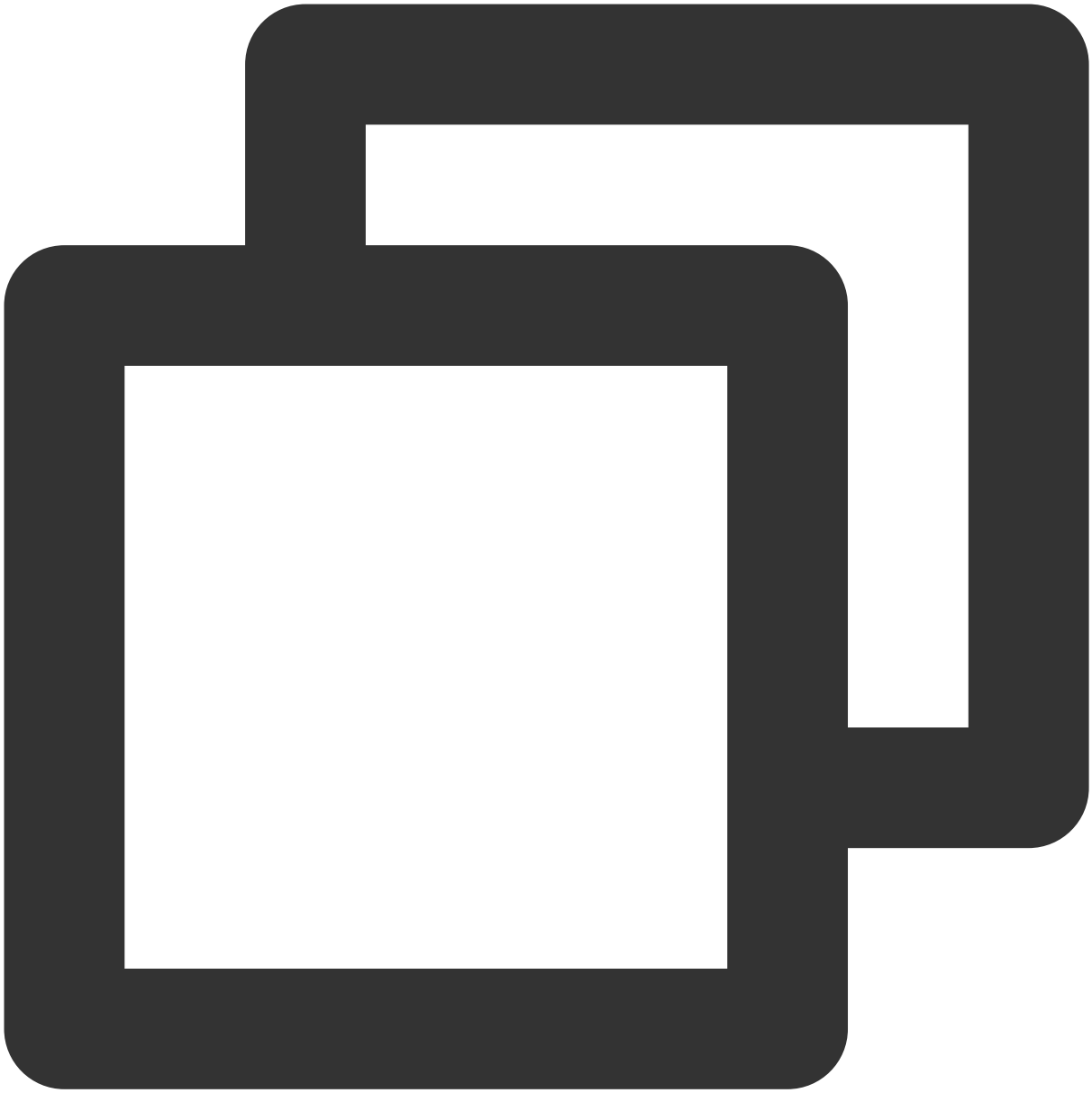
You were invited to a call.



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

### CALLING_CANCEL

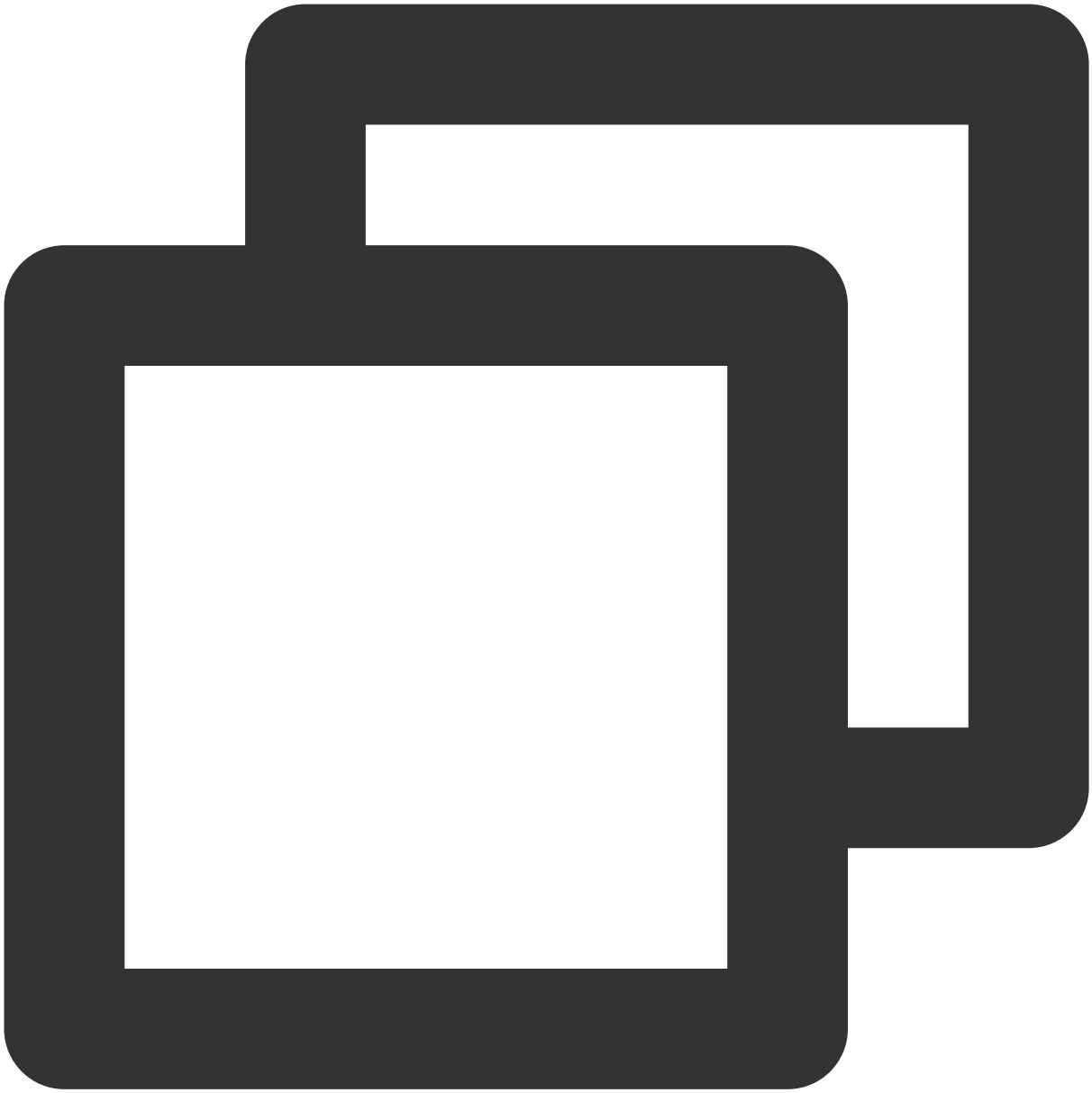
The call was canceled. This callback is received by an invitee.



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

### CALLING_END

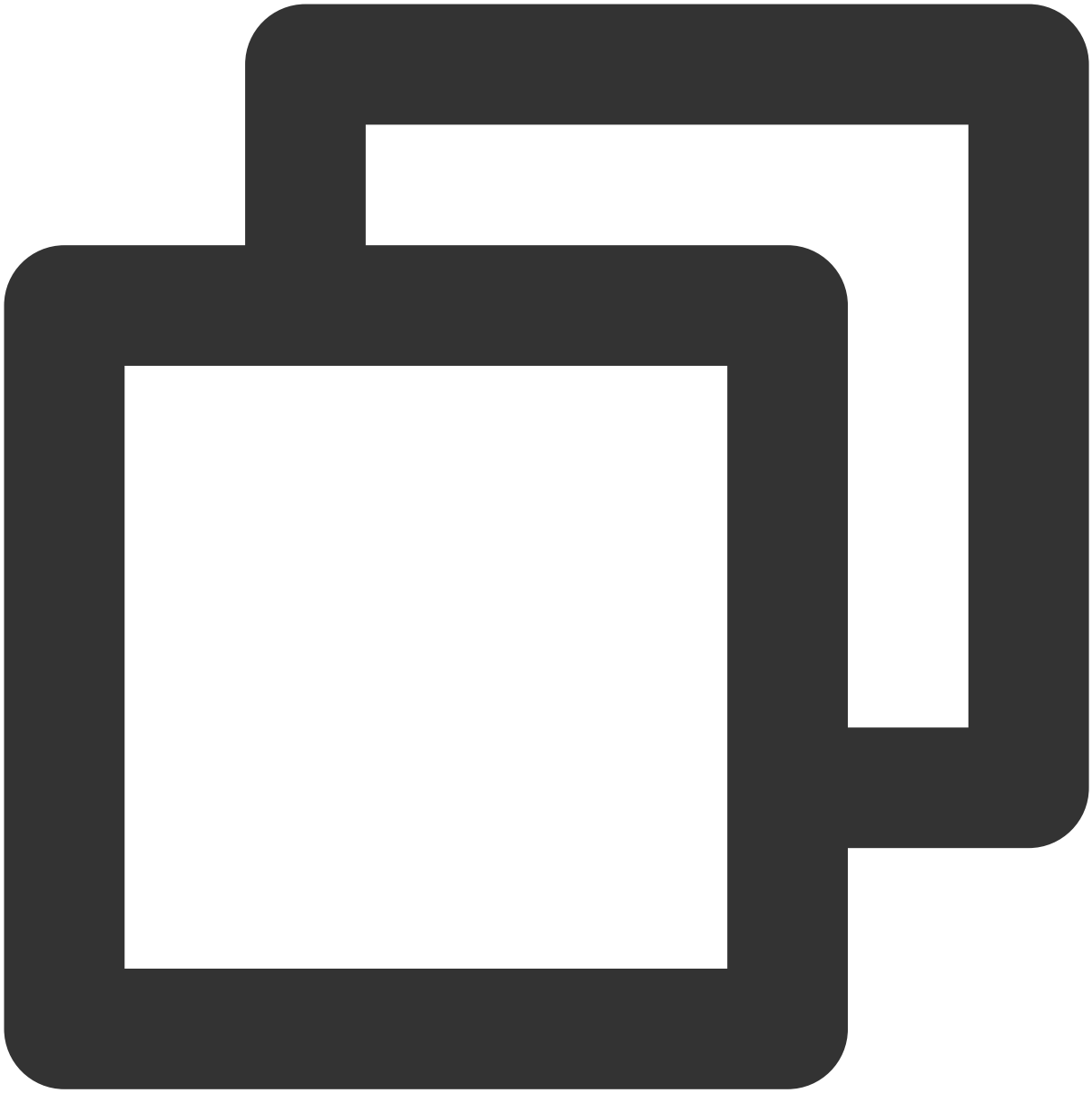
The call ended.



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

### DEVICED_UPDATED

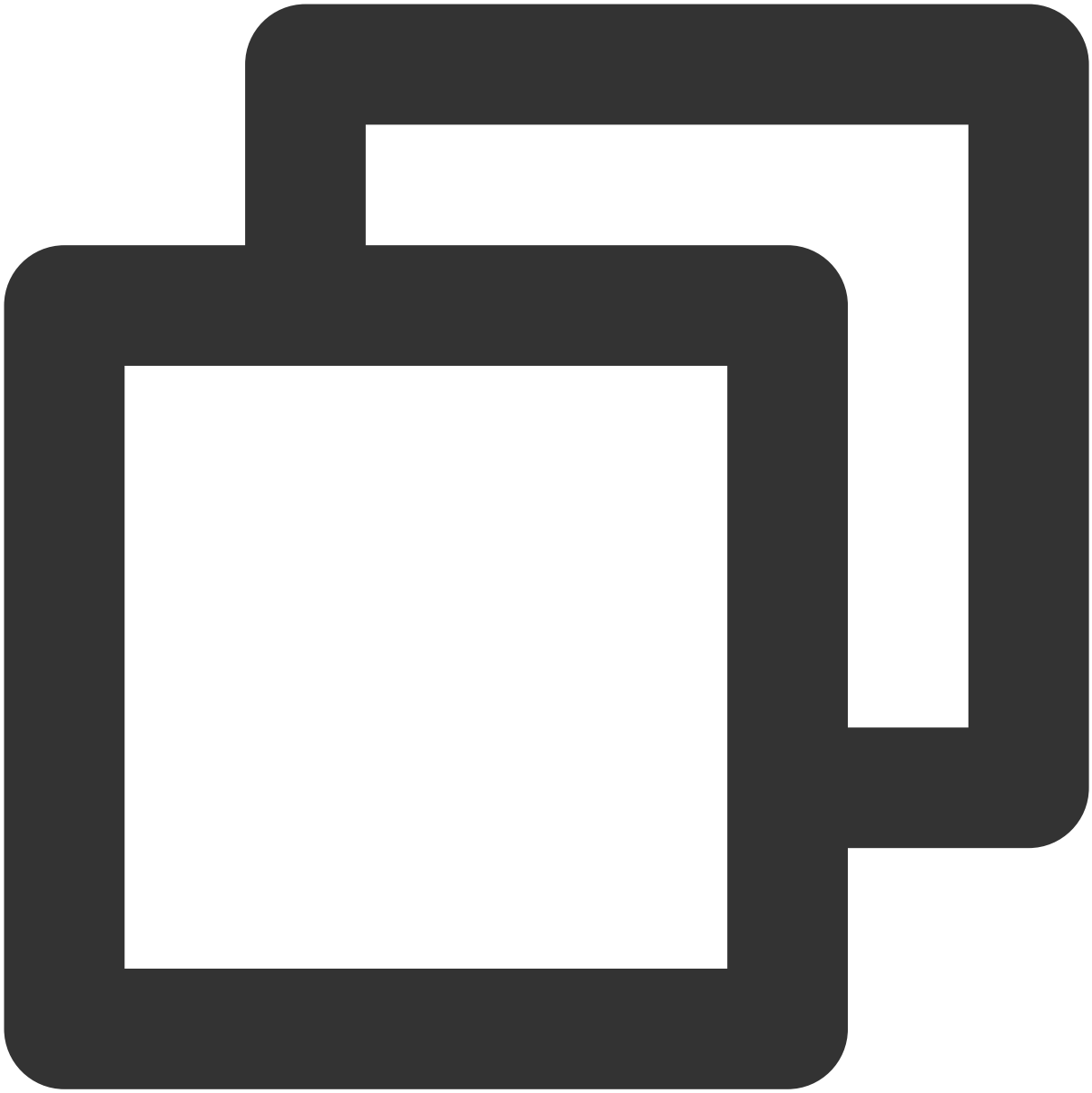
The device list was updated.



```
let handleDeviceUpdated = function({ microphoneList, cameraList, currentMicrophoneI
  console.log(microphoneList, cameraList, currentMicrophoneID, currentCameraID)
};
tuiCallEngine.on(TUICallEvent.DEVICED_UPDATED, handleDeviceUpdated);
```

### CALL_TYPE_CHANGED

The call type changed.



```
let handleCallTypeChanged = function({ oldCallType, newCallType }) {
  console.log(oldCallType, newCallType)
};
tuiCallEngine.on(TUICallEvent.CALL_TYPE_CHANGED, handleDeviceUpdated);
```

# Flutter
# API Overview

Last updated：2024-03-20 17:08:47

## TUICallKit (UI Included)

TUICallKit is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat.

| API | Description |
| --- | --- |
| login | Login |
| logout | Log out |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| joinInGroupCall | Joins a group call. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |
| setCallingBell | Custom ringtone. |

## TUICallEngine (No UI)

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

| API | Description |
| --- | --- |
| init | Authenticates the basic audio/video call capabilities. |
| unInit | The destructor function, which releases resources used by TUICallEngine. |
| addObserver | Registers an event listener. |

| | |
|---|---|
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Changes the call type, for example, from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |
| openCamera | Turns the camera on. |
| closeCamera | Turns the camera off. |
| switchCamera | Switches between the front and rear cameras. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver or speaker). |
| setSelfInfo | Sets the alias and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for TUICallEngine (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |
| setVideoEncoderParams | Set the encoding parameters of video encoder. |
| queryRecentCalls | Query call record. |
| deleteRecordCalls | Delete call record. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

# TUICallObserver

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

| API | Description |
| --- | --- |
| onError | An error occurred during the call. |
| onCallReceived | A call was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user has a video stream. |
| onUserAudioAvailable | Whether a user has an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | The current user is kicked offline |
| onUserSigExpired | Ticket expires while online |

# TUICallKit

Last updated：2024-04-15 17:41:03

## TUICallKit API

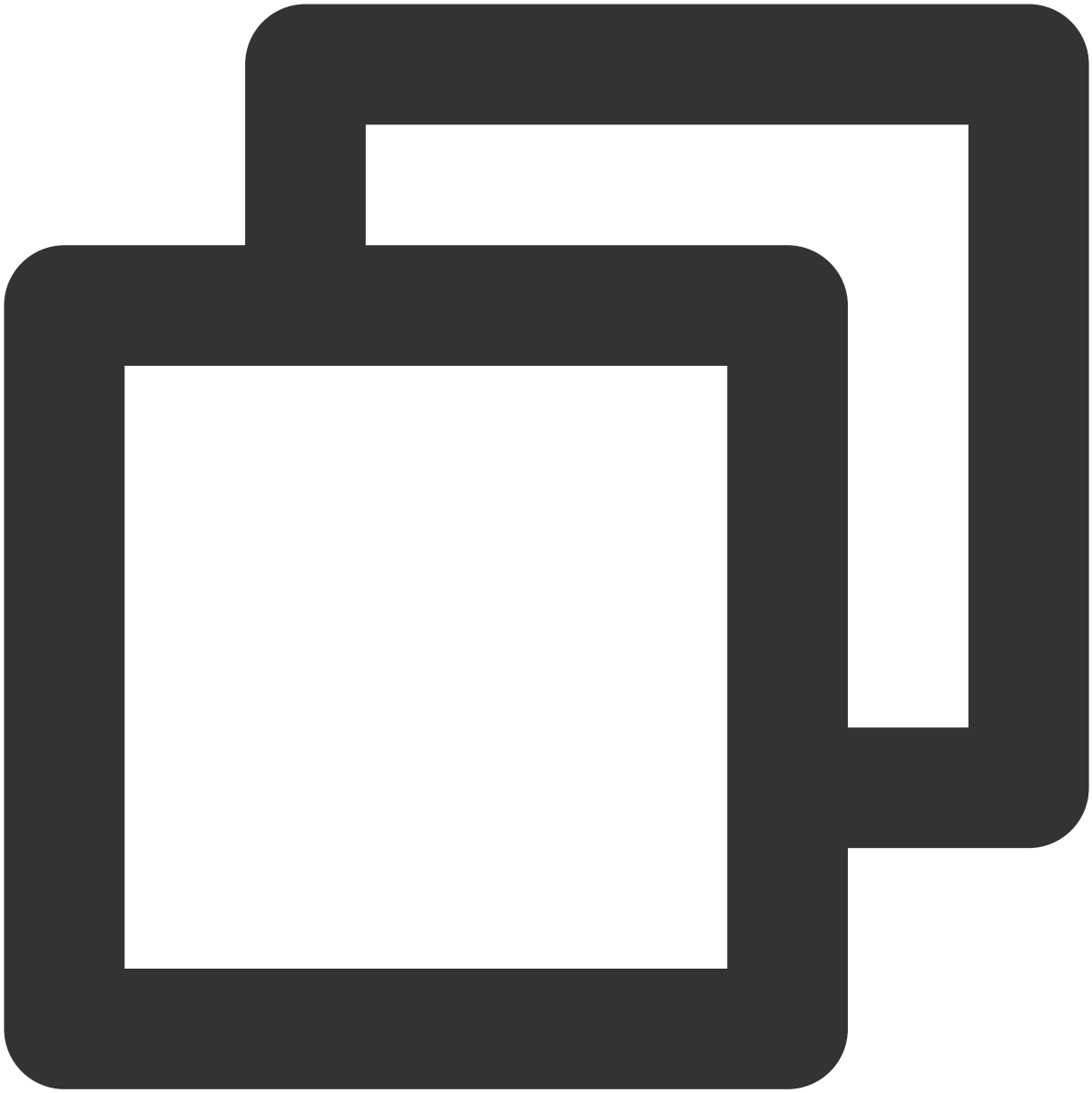
`TUICallKit` is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat. For directions on integration, see Integrating TUICallKit.

## API Overview

| API | Description |
| --- | --- |
| login | Login |
| logout | Log out |
| setSelfInfo | Sets the alias and profile photo. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| joinInGroupCall | Joins a group call. |
| enableMuteMode | Sets whether to turn on the mute mode. |
| enableFloatWindow | Sets whether to enable floating windows. |
| setCallingBell | Custom ringtone. |

## API Details

### login

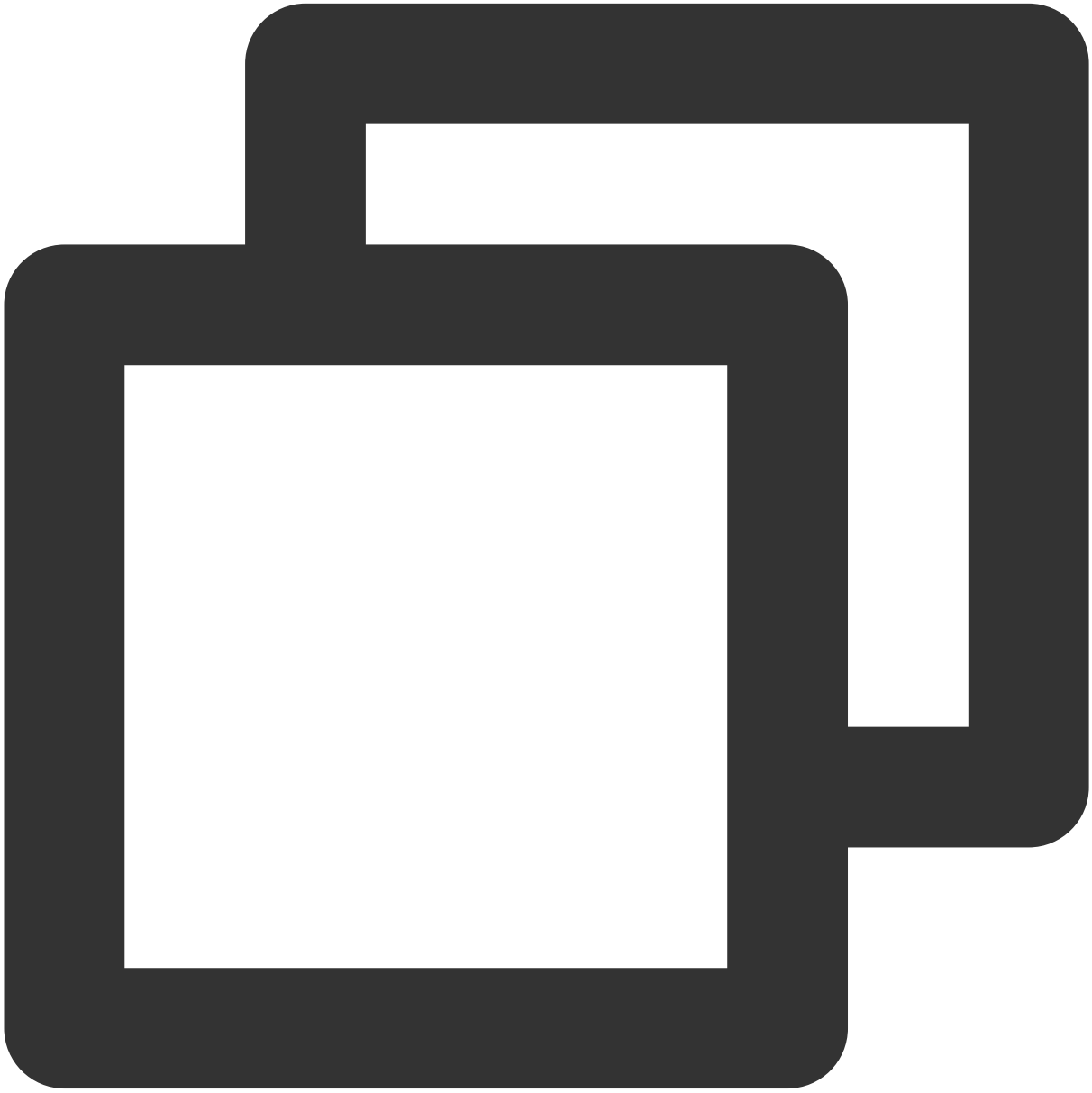


```
Future<TUIResult> login(int sdkAppId, String userId, String userSig);
```

| Parameter | Type | Description |
| --- | --- | --- |
| sdkAppId | int | You can view SDKAppID in Application Management of the IM console. |
| userId | String | The ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_). |
| userSig | String | User signature. (For details on the calculation method, see Generating UserSig.) |

| | | |
|---|---|---|
| return value | TUIResult | Contains code and message information:<br>If code is empty (""), the call is successful.<br>If code is not empty ("") , the call fails. See message for the failure reason. |

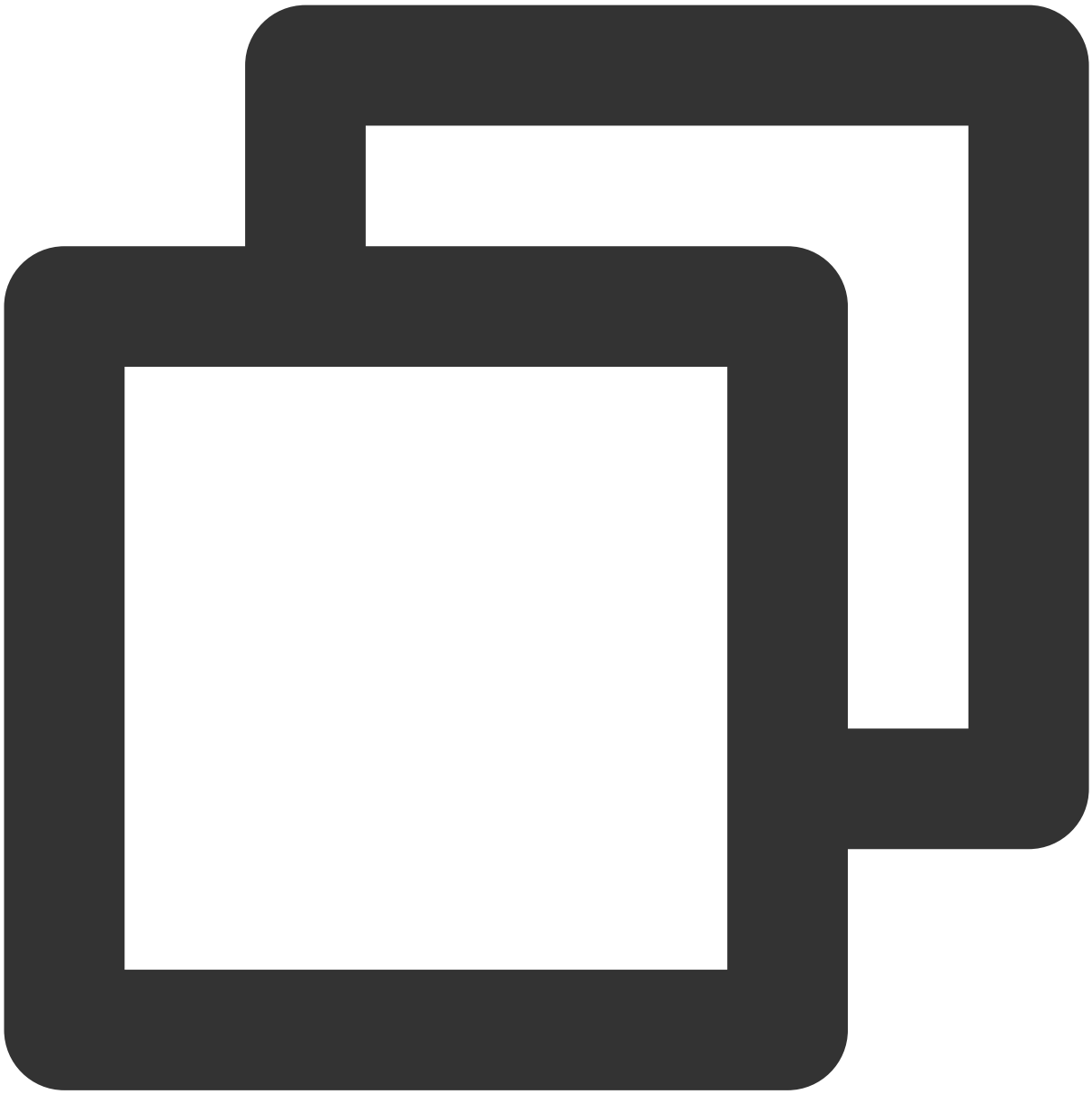
### logout



```
Future<void> logout()
```

### setSelfInfo

This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



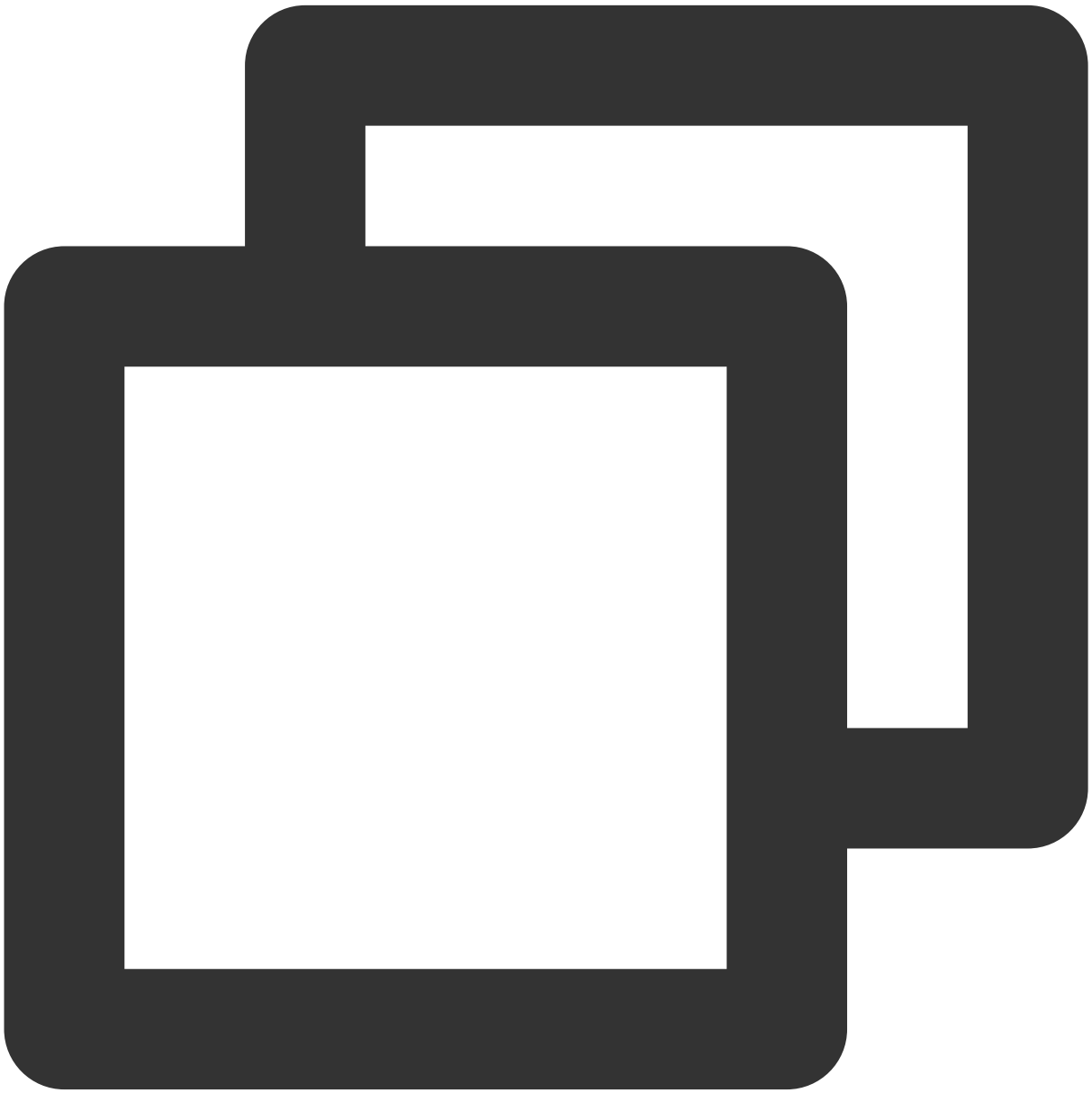
```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

| Parameter | Type | Description |
| --- | --- | --- |
| nickname | String | The alias. |
| avatar | String | The profile photo. |
| | | |

| | | |
|---|---|---|
| return value | TUIResult | Contains code and message information:<br>If code is empty (""), the call is successful.<br>If code is not empty ("") , the call fails. See message for the failure reason. |

### call

This API is used to make a (one-to-one) call.



```
Future<void> call(String userId, TUICallMediaType callMediaType, [TUICallParams? pa
```

The parameters are described below:

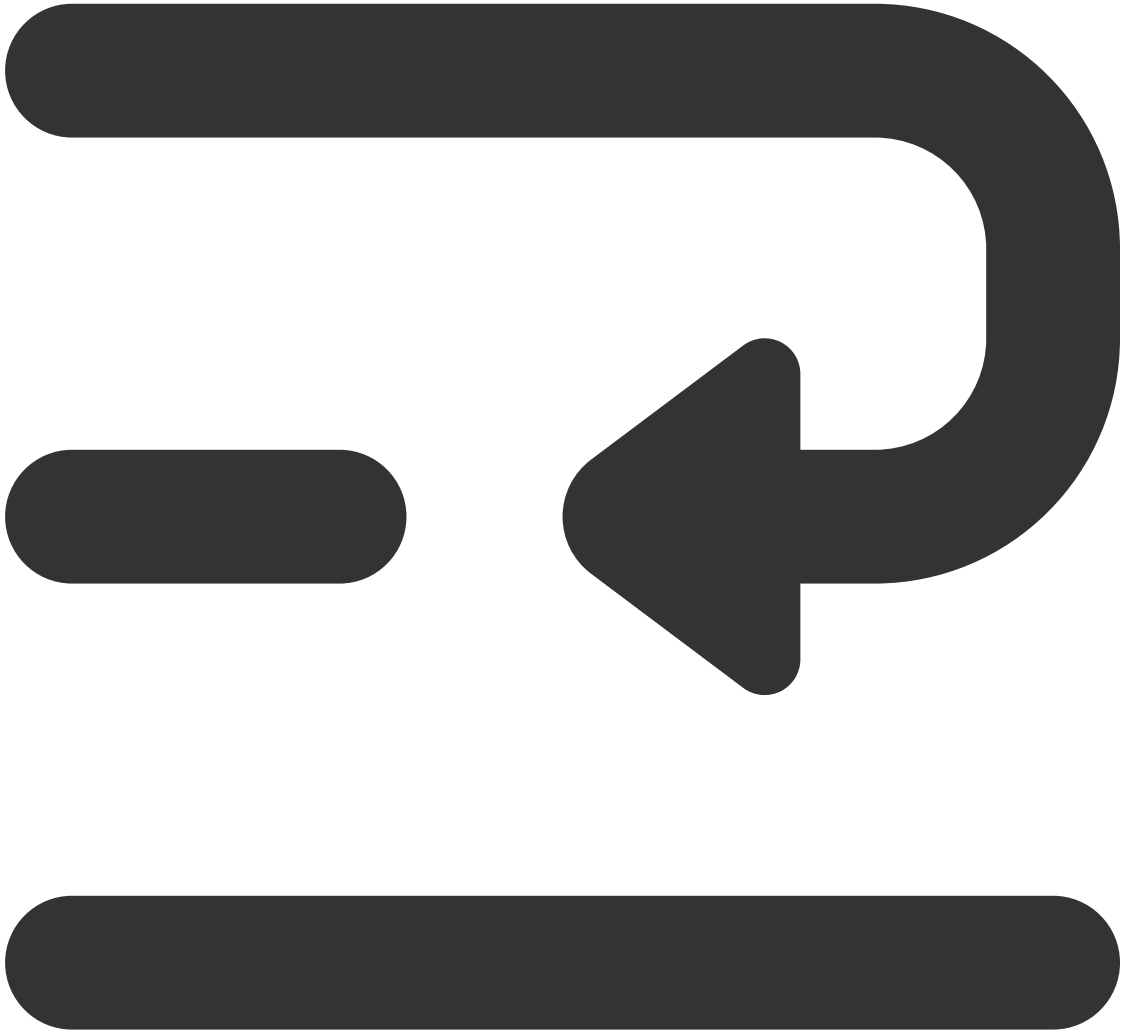
| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | Call extension parameters, such as roomID, call timeout, offline push info,etc |

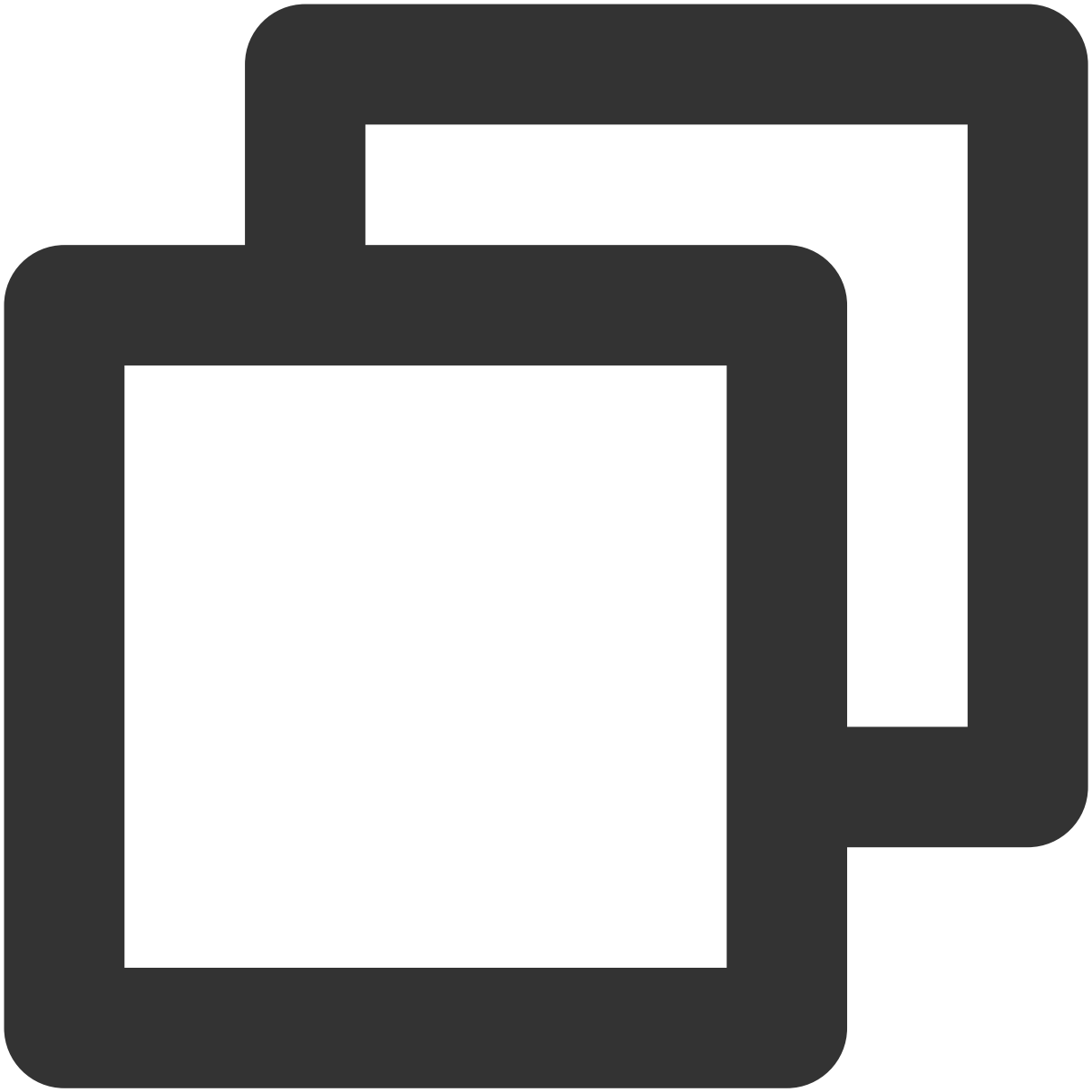
### groupCall

This API is used to make a group call.

**Notice：**

you need to create an IM group before using the group call. If you have already created it, please ignore it.





```
Future<void> groupCall(String groupId, List<String> userIdList, TUICallMediaType ca
```

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID. |
| userIdList | List<String> | The target user IDs. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | Call extension parameters, such as roomID, call timeout, offline push |

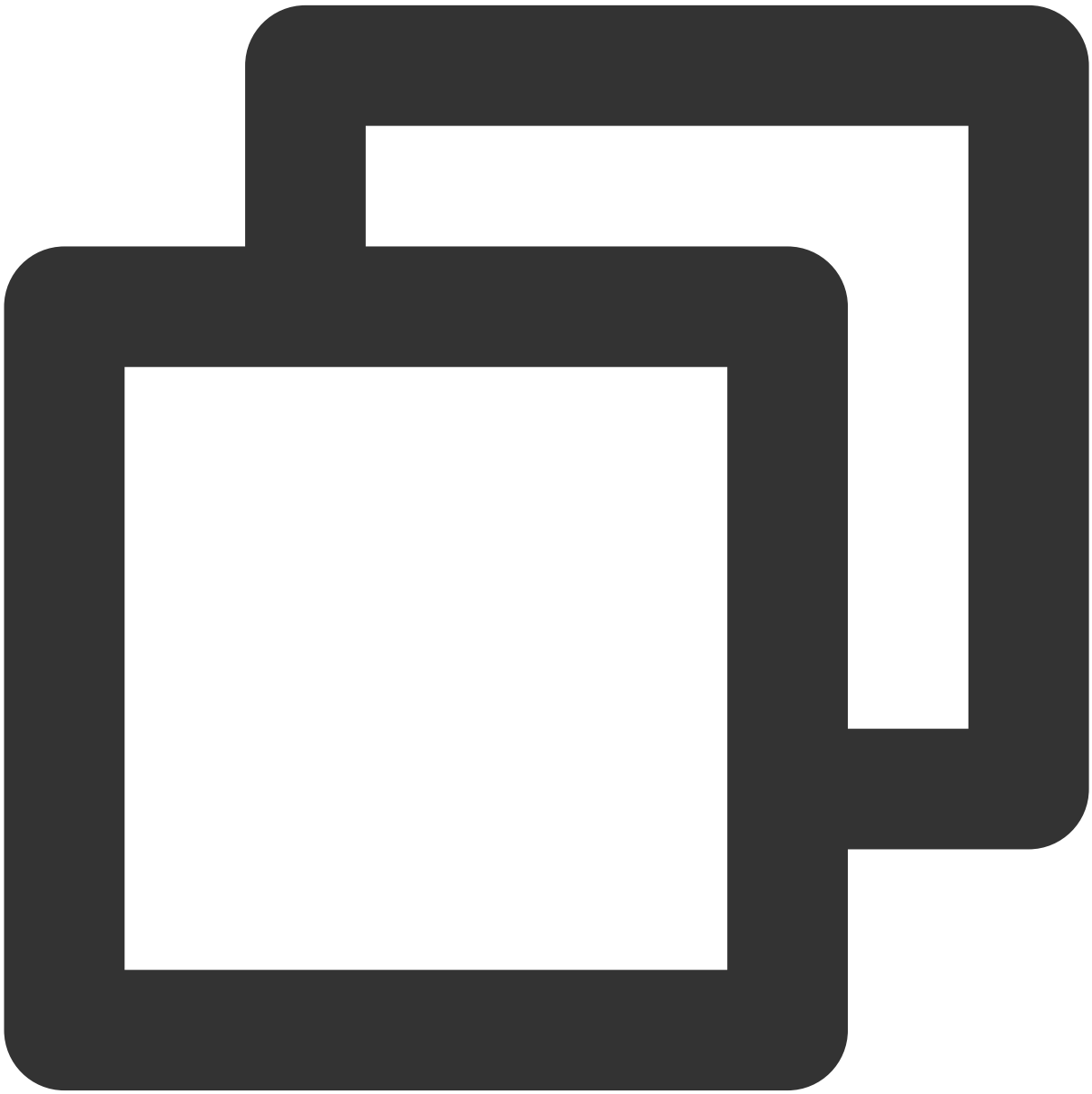
| | | |
|---|---|---|
| | | info,etc |

### joinInGroupCall

This API is used to join a group call.

**Notice：**

you need to create an IM group before using the group call. If you have already created it, please ignore it.



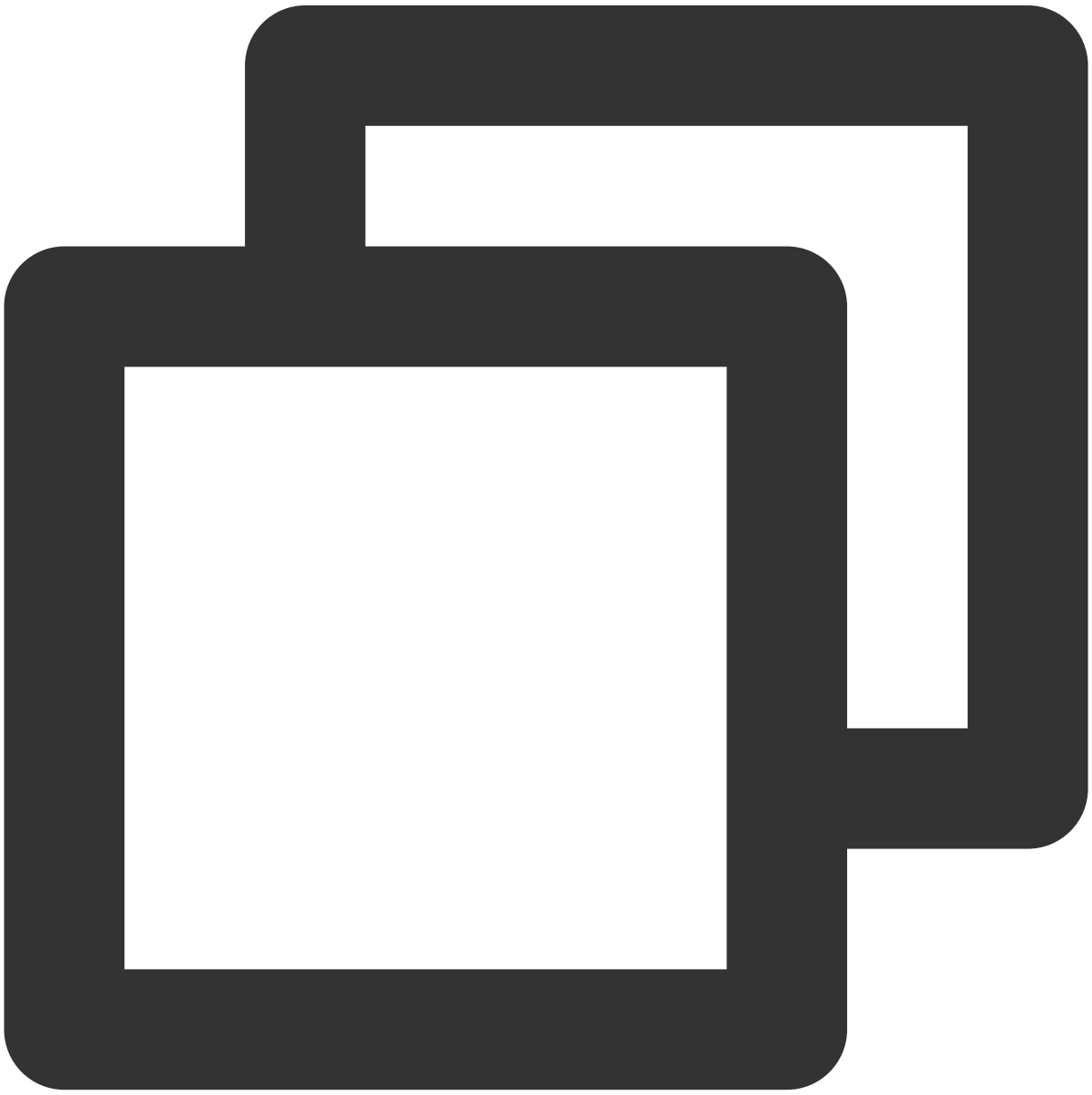
```
Future<void> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaType cal
```

| | | |
|---|---|---|
| | | |

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomID | The room ID. |
| groupId | String | The group ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### enableMuteMode

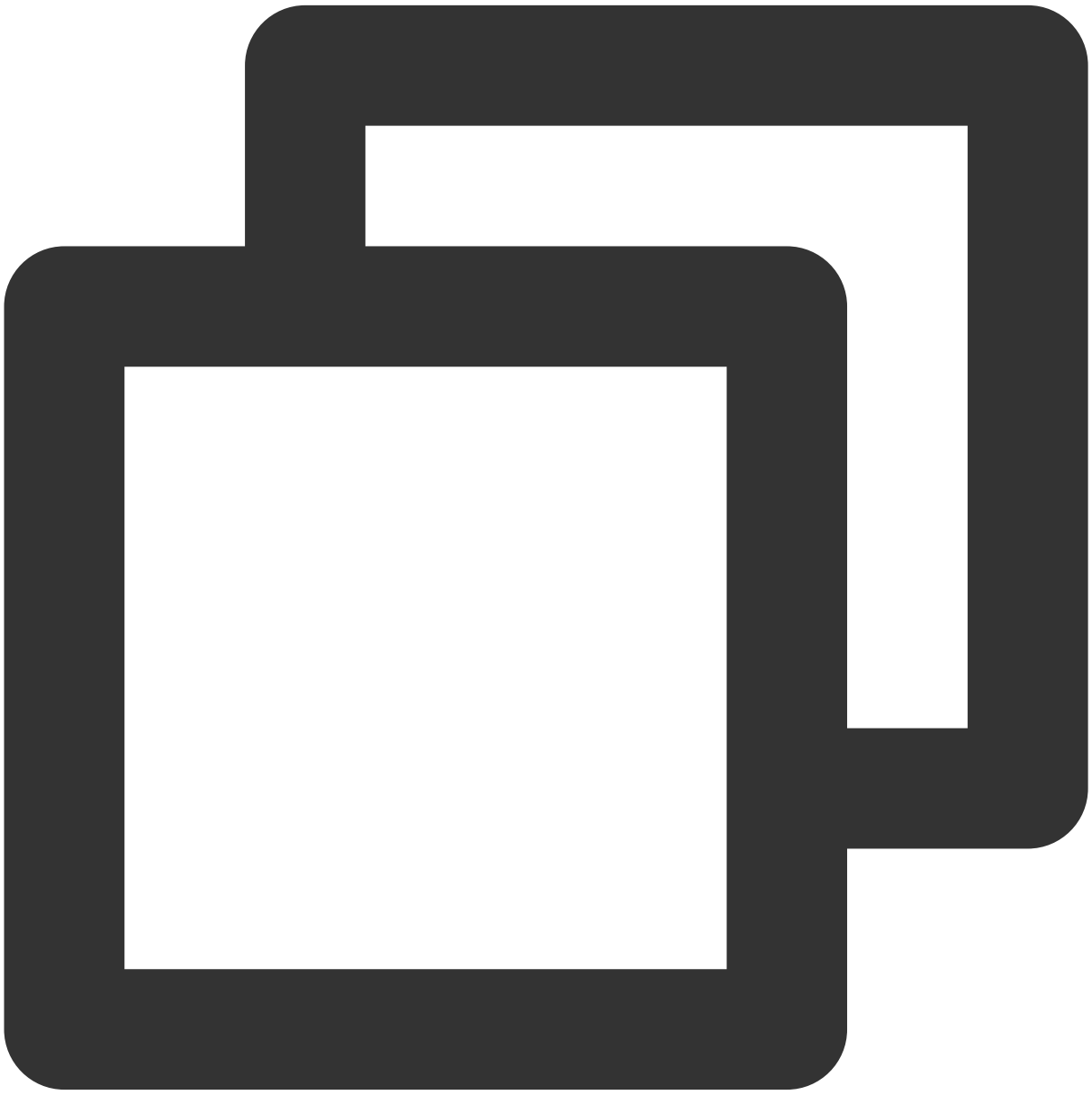
This API is used to set whether to turn on the mute mode.



```
Future<void> enableMuteMode(bool enable)
```

### enableFloatWindow

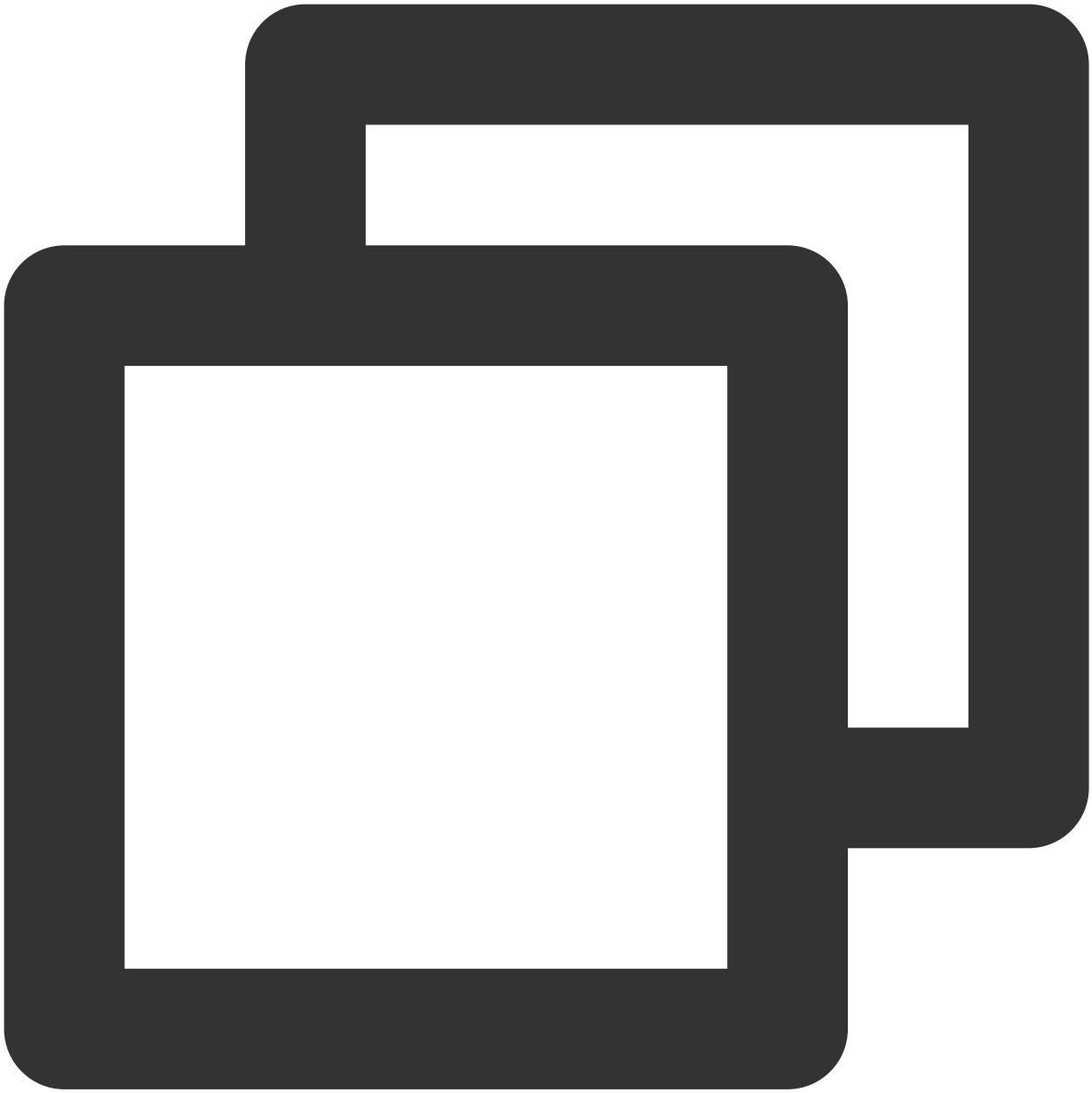
This API is used to set whether to enable floating windows. The default value is `false`, and the floating window button in the top left corner of the call view is hidden. If it is set to `true`, the button will become visible.



```
Future<void> enableFloatWindow(bool enable)
```

### setCallingBell

Custom ringtone。



```
Future<void> setCallingBell(String assetName)
```

| Parameter | Type | Description |
| --- | --- | --- |
| assetName | String | The path of the ringtone. The ringtone file needs to be added to the assets resource of the main project. |

# TUICallEngine

Last updated：2023-08-22 10:27:55

## TUICallEngine APIs

`TUICallEngine` is an audio/video call component that **does not include UI elements**. If `TUICallKit` does not meet your requirements, you can use the APIs of `TUICallEngine` to customize your project.

## Overview

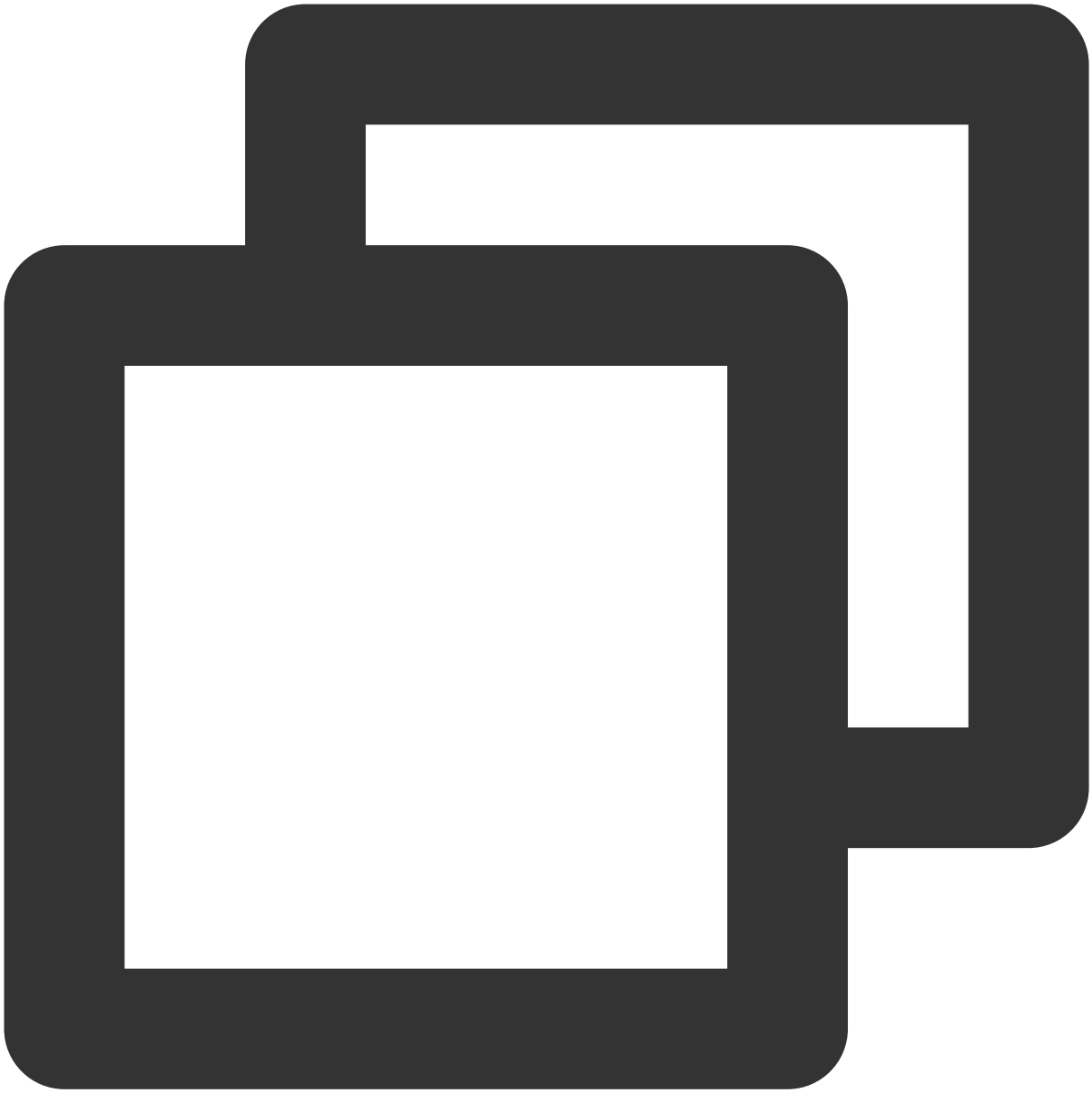
| API | Description |
| --- | --- |
| init | Authenticates the basic audio/video call capabilities. |
| unInit | The destructor function, which releases resources used by TUICallEngine. |
| addObserver | Registers an event listener. |
| removeObserver | Unregisters an event listener. |
| call | Makes a one-to-one call. |
| groupCall | Makes a group call. |
| accept | Accepts a call. |
| reject | Rejects a call. |
| hangup | Ends a call. |
| ignore | Ignores a call. |
| inviteUser | Invites users to the current group call. |
| joinInGroupCall | Joins a group call. |
| switchCallMediaType | Changes the call type, for example, from video call to audio call. |
| startRemoteView | Subscribes to the video stream of a remote user. |
| stopRemoteView | Unsubscribes from the video stream of a remote user. |
| openCamera | Turns the camera on. |

| | |
|---|---|
| closeCamera | Turns the camera off. |
| switchCamera | Switches between the front and rear cameras. |
| openMicrophone | Turns the mic on. |
| closeMicrophone | Turns the mic off. |
| selectAudioPlaybackDevice | Selects the audio playback device (receiver or speaker). |
| setSelfInfo | Sets the alias and profile photo. |
| enableMultiDeviceAbility | Sets whether to enable multi-device login for TUICallEngine (supported by the premium package). |
| setVideoRenderParams | Set the rendering mode of video image. |
| setVideoEncoderParams | Set the encoding parameters of video encoder. |
| queryRecentCalls | Query call record. |
| deleteRecordCalls | Delete call record. |
| setBeautyLevel | Set beauty level, support turning off default beauty. |

# Details

### init

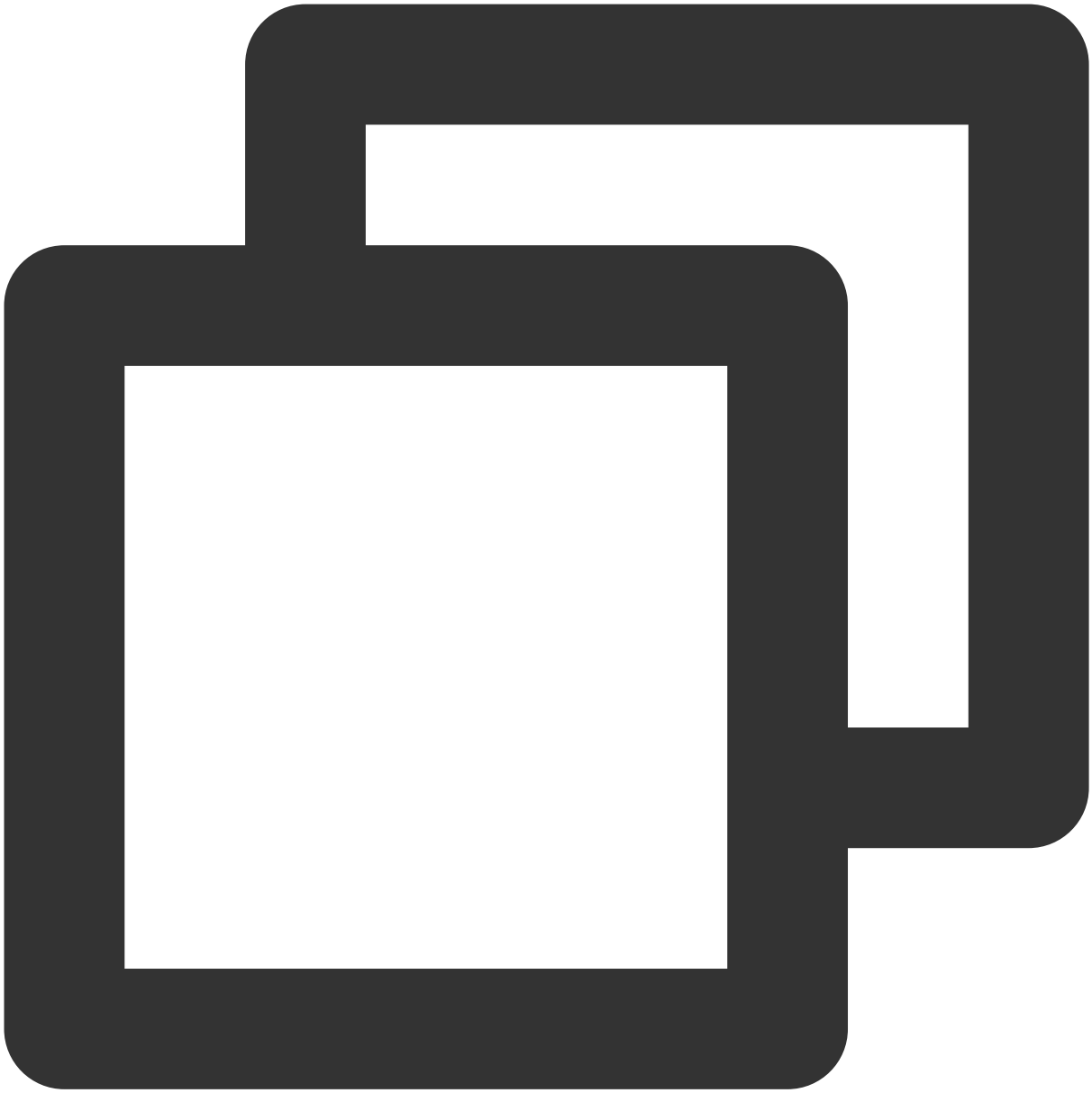
This API is used to initialize `TUICallEngine` . Call it to authenticate the call service and perform other required actions before you call other APIs.



```
Future<TUIResult> init(int sdkAppID, String userId, String userSig)
```

### unInit

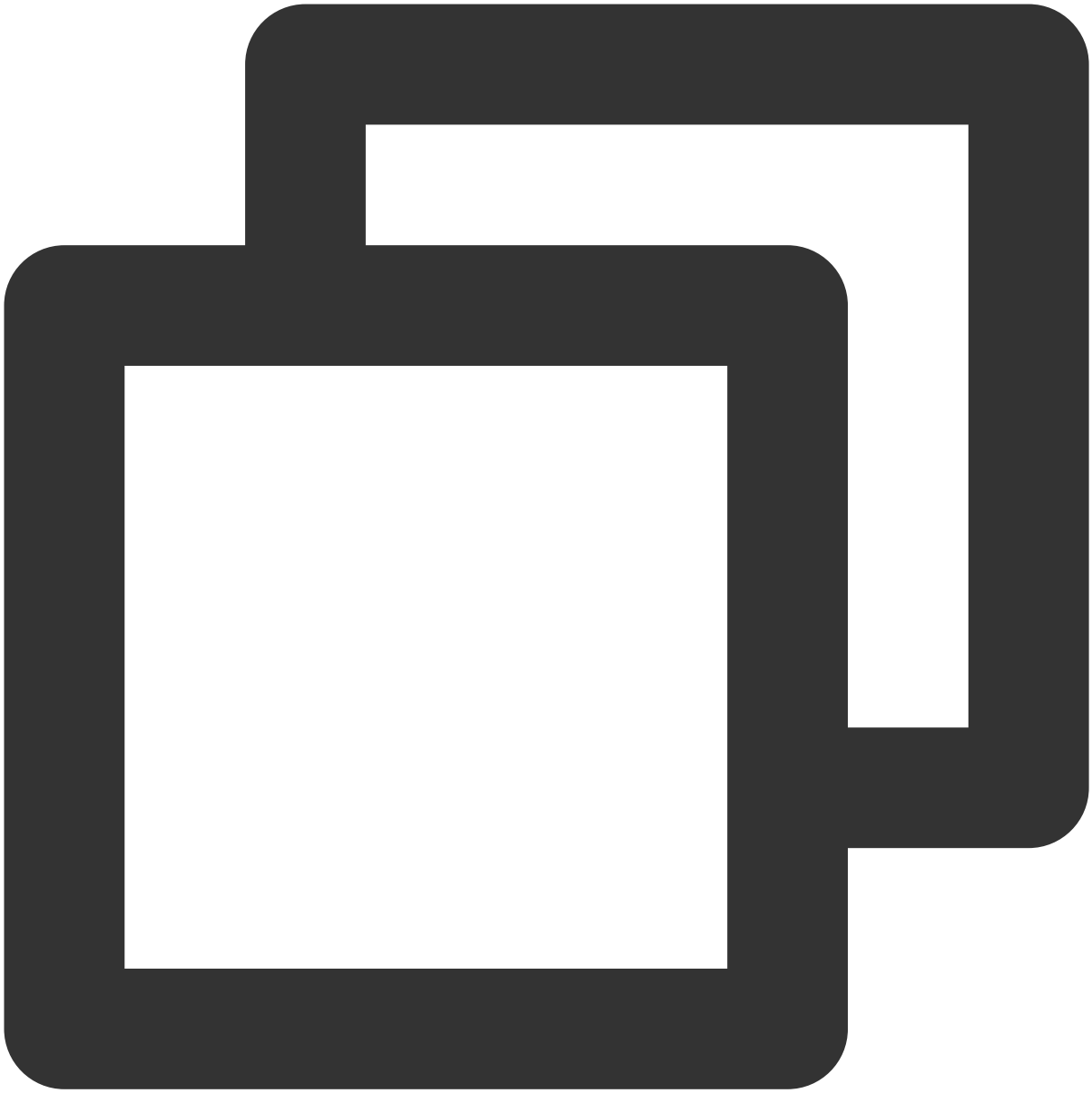
The destructor function, which releases resources used by TUICallEngine.



```
Future<TUIResult> unInit()
```

### addObserver

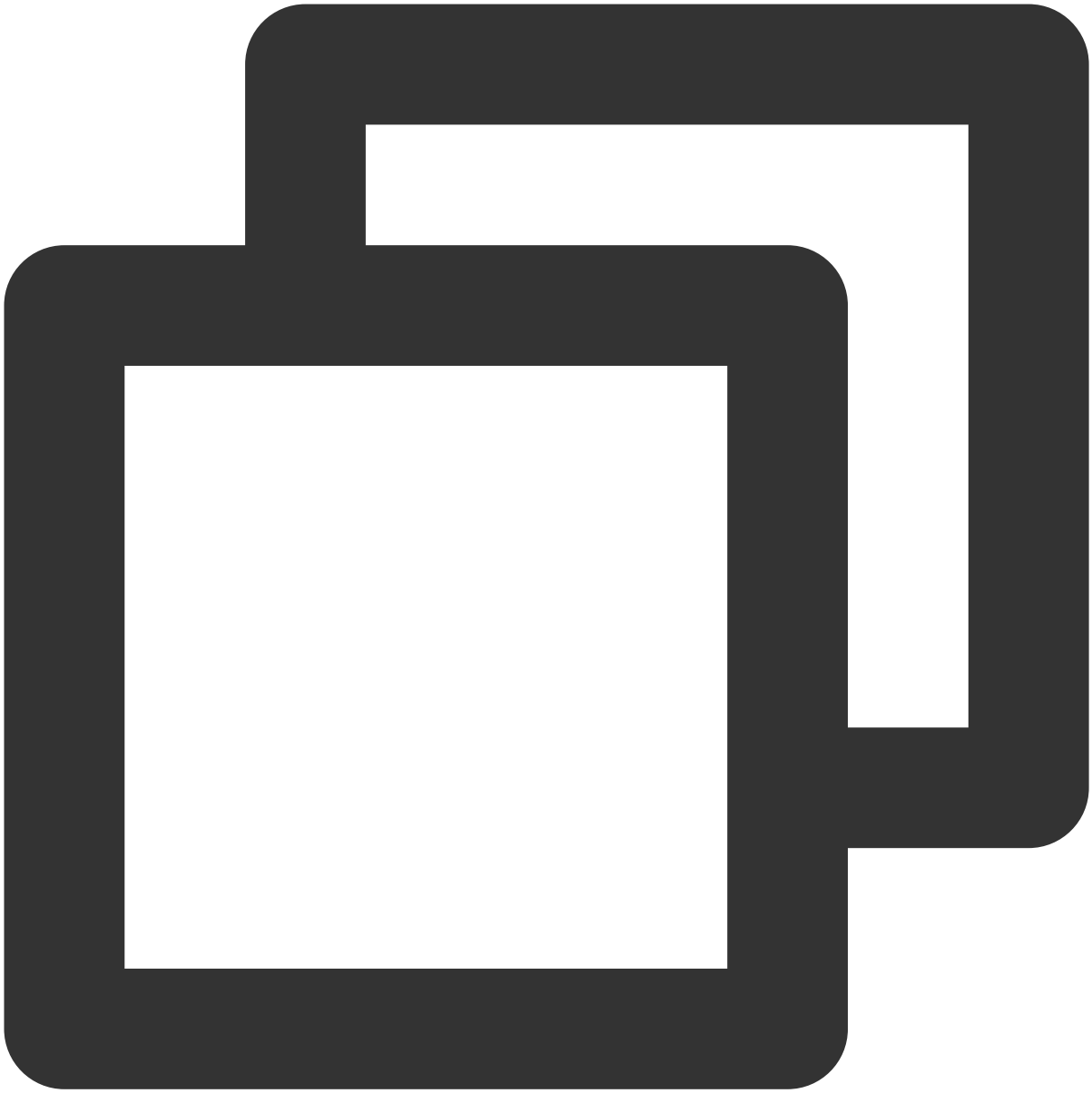
This API is used to register an event listener to listen for `TUICallObserver` events.



```
Future<void> addObserver(TUICallObserver observer)
```

### removeObserver

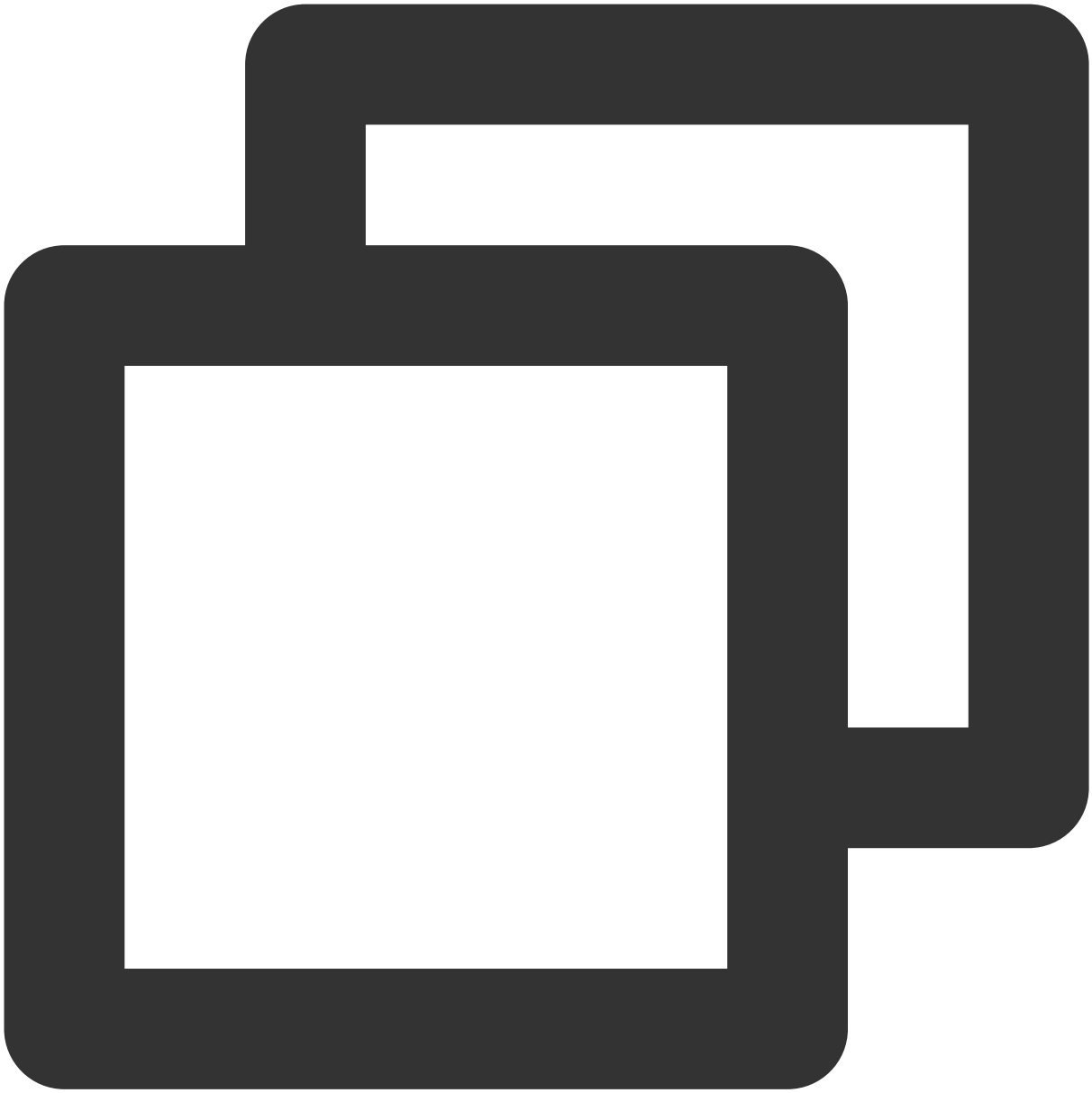
This API is used to unregister an event listener.



```
Future<void> removeObserver(TUICallObserver observer)
```

### call

This API is used to make a (one-to-one) call.



```
Future<TUIResult> call(String userId, TUICallMediaType mediaType, TUICallParams par
```

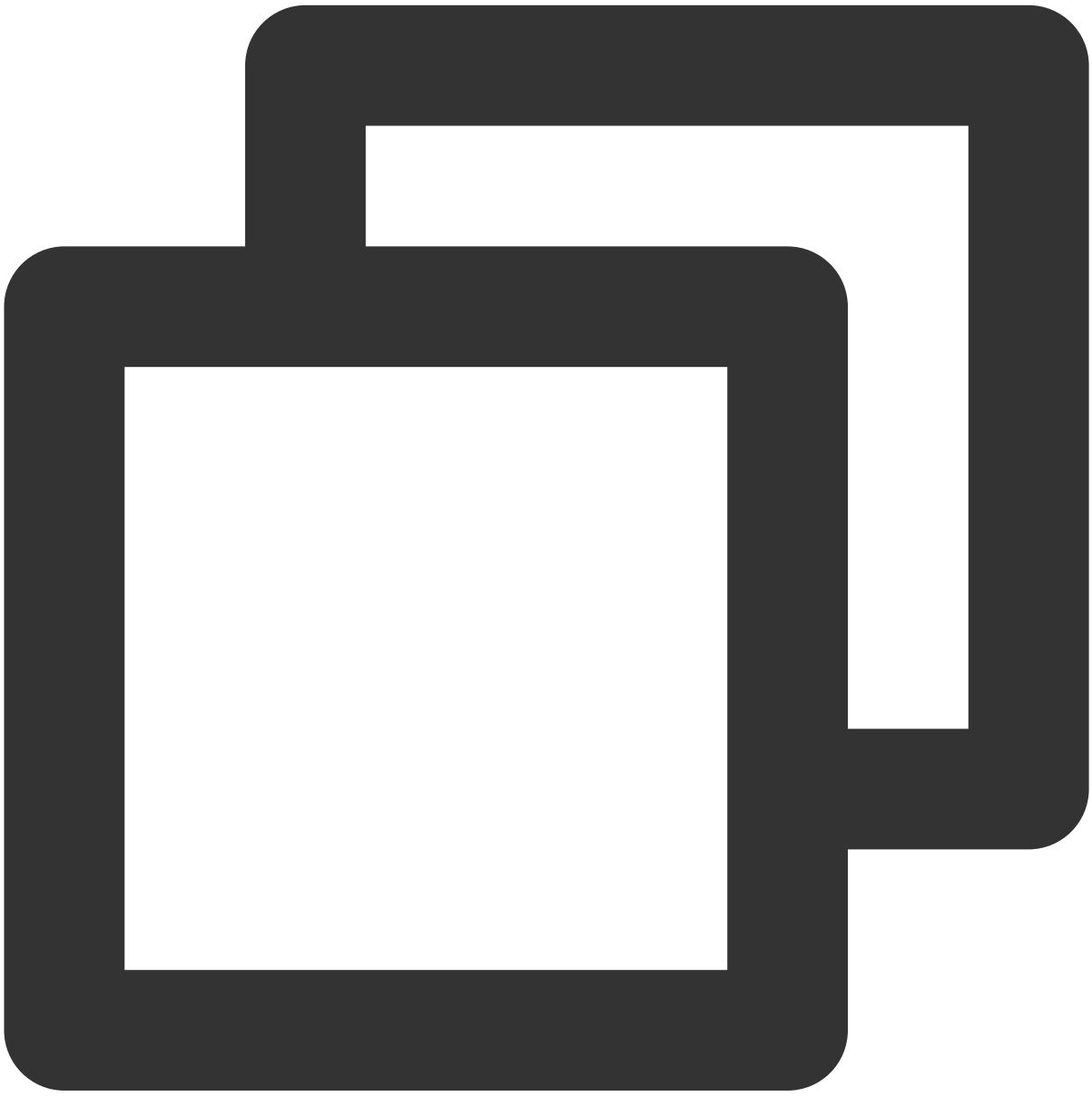
| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| mediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | An additional parameter, such as roomID, call timeout, offline push info, etc |

### groupCall

This API is used to make a group call.

**Notice：**

Before making a group call, you need to create an IM group first.



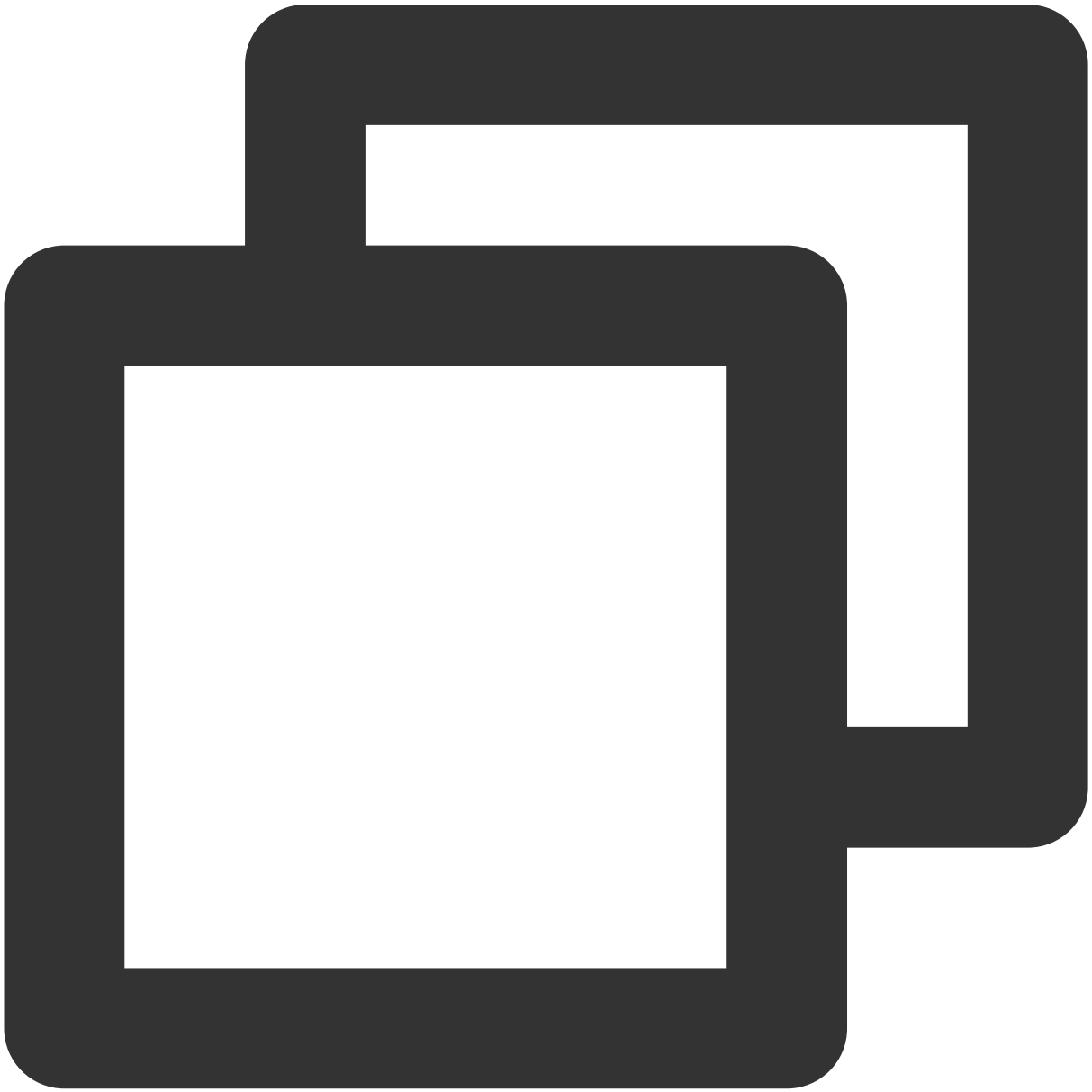
```
Future<TUIResult> groupCall(String groupId, List<String> userIdList, TUICallMediaTy
```

| Parameter | Type | Description |
| --- | --- | --- |
| groupId | String | The group ID. |

| | | |
|---|---|---|
| userIdList | List<String> | The target user IDs. |
| mediaType | TUICallMediaType | The call type, which can be video or audio. |
| params | TUICallParams | An additional parameter. such as roomID, call timeout, offline push info, etc |

### accept

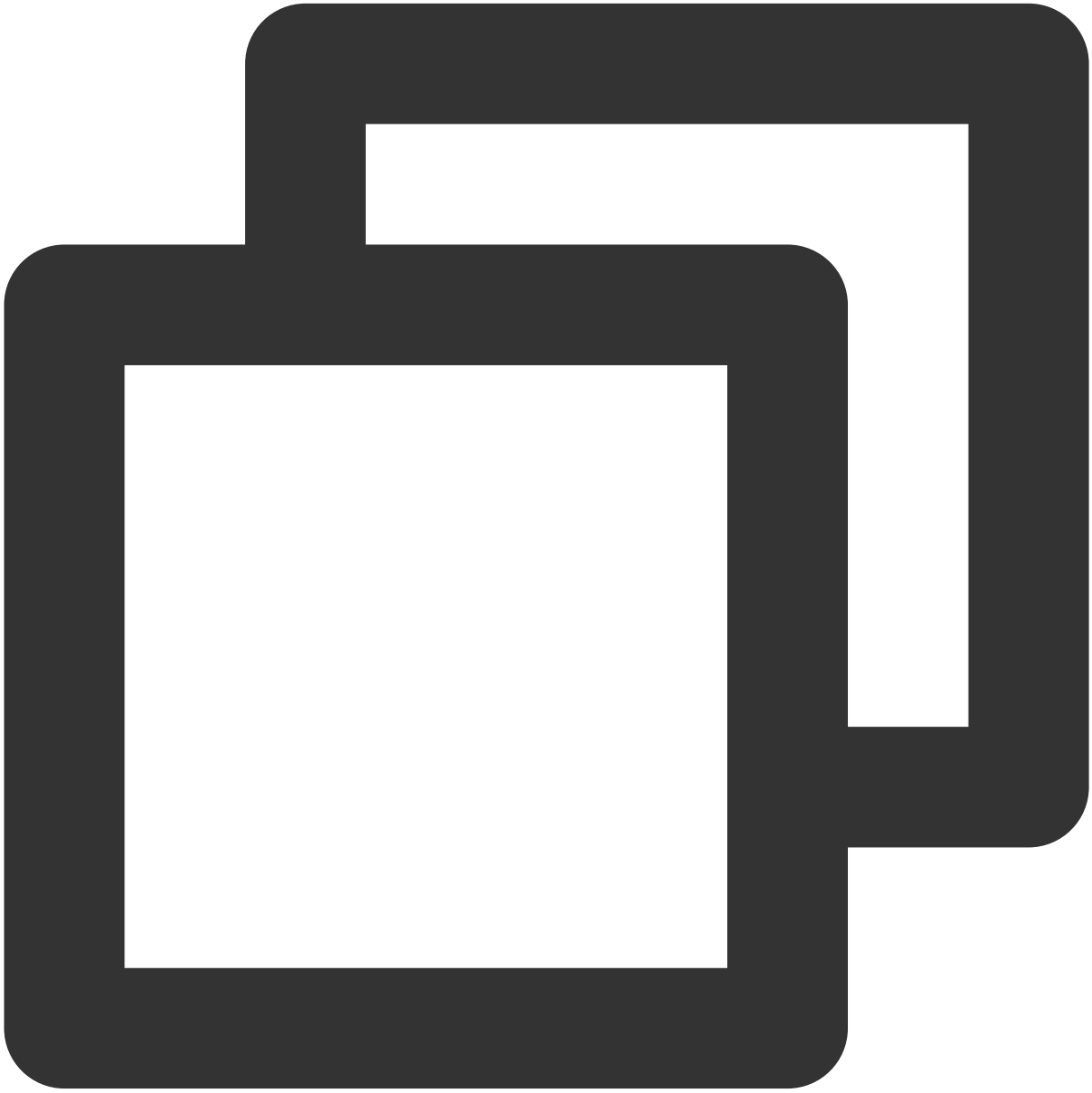
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



```
Future<TUIResult> accept()
```

### reject

This API is used to reject a call. After receiving the `onCallReceived()` callback, you can call this API to reject the call.

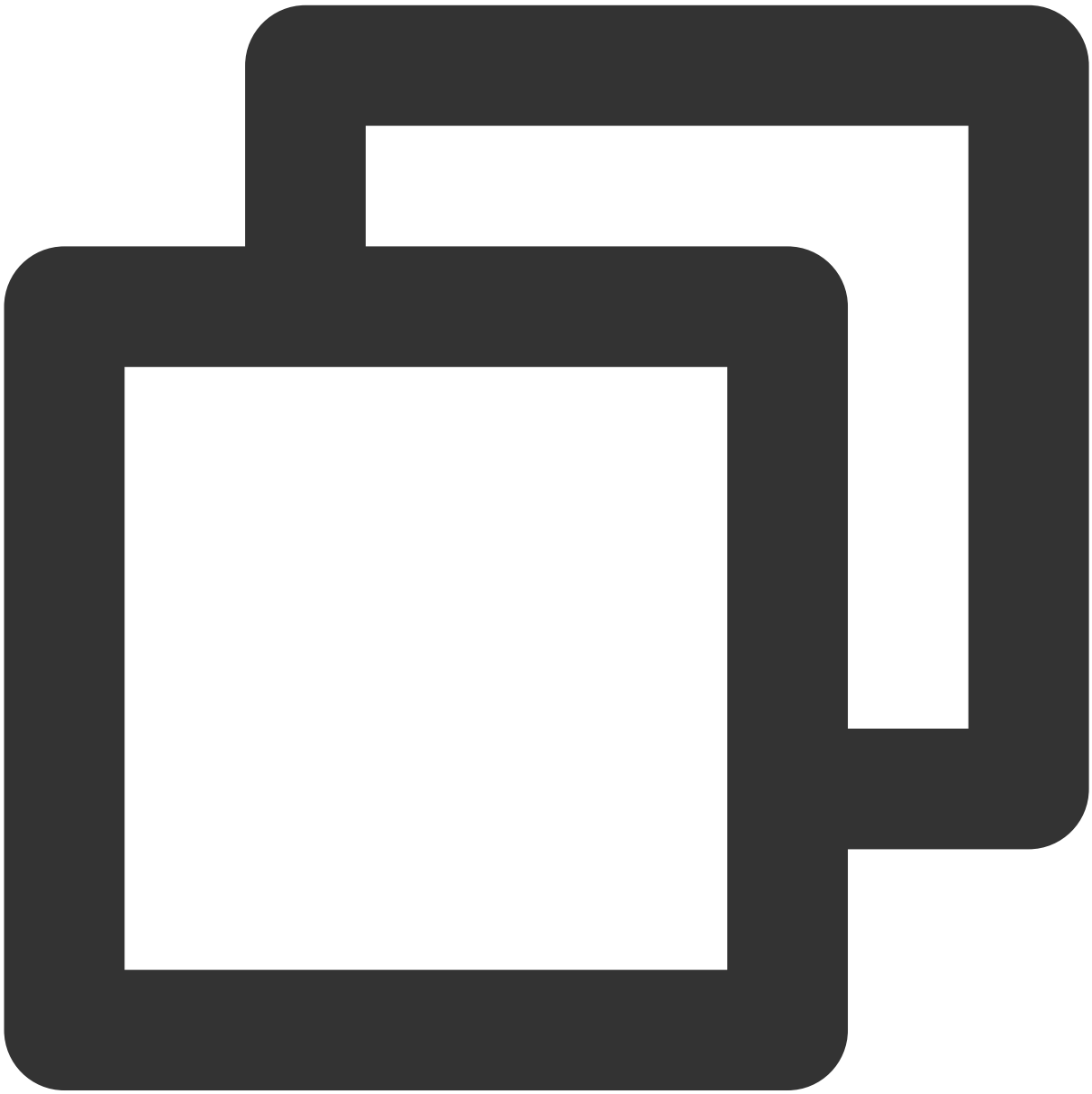


```
Future<TUIResult> reject()
```

### ignore

This API is used to ignore a call. After receiving the `onCallReceived()` , you can call this API to ignore the call. The caller will receive the `onUserLineBusy` callback.

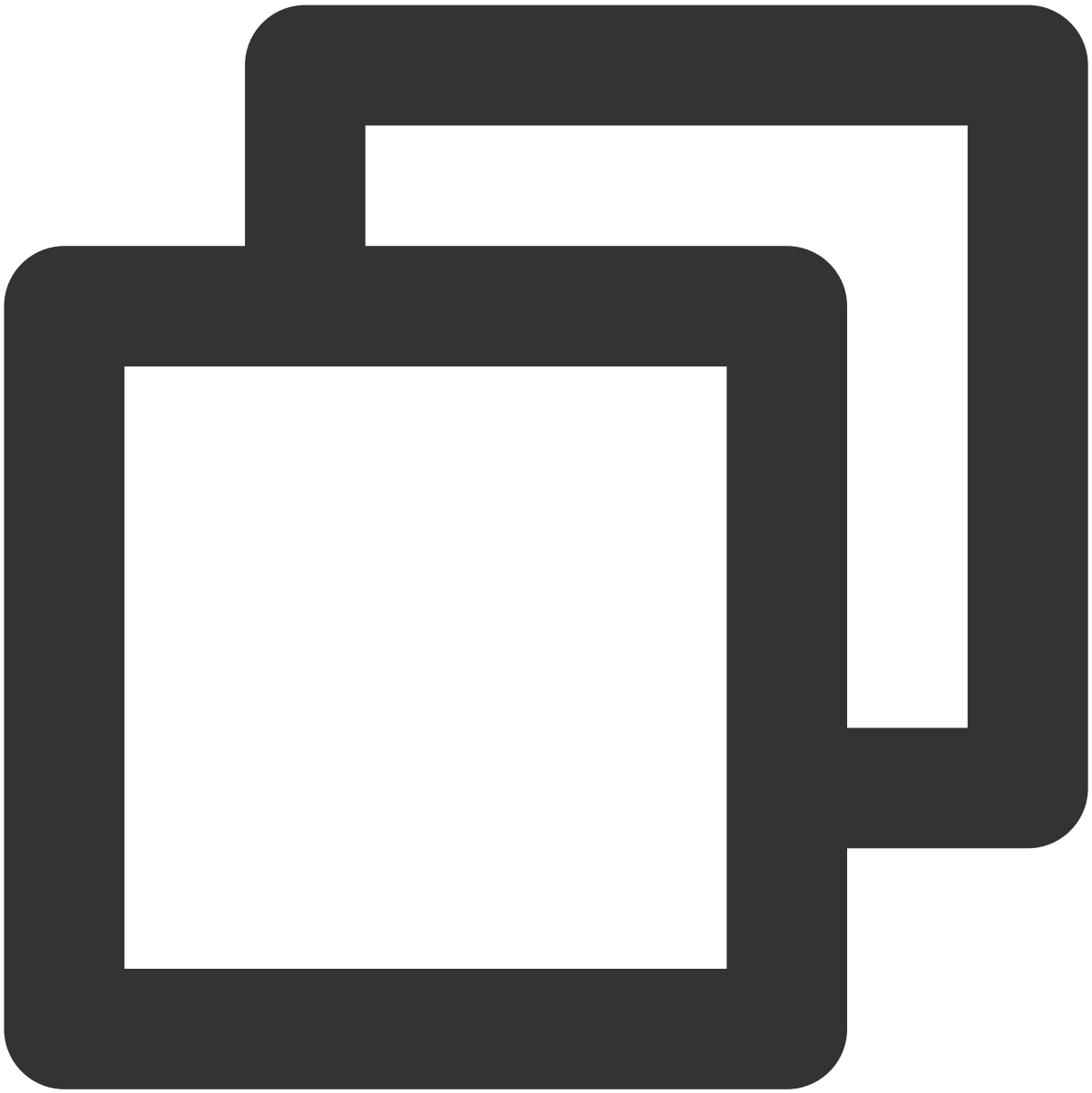
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



```
Future<TUIResult> ignore()
```

### hangup

This API is used to end a call.

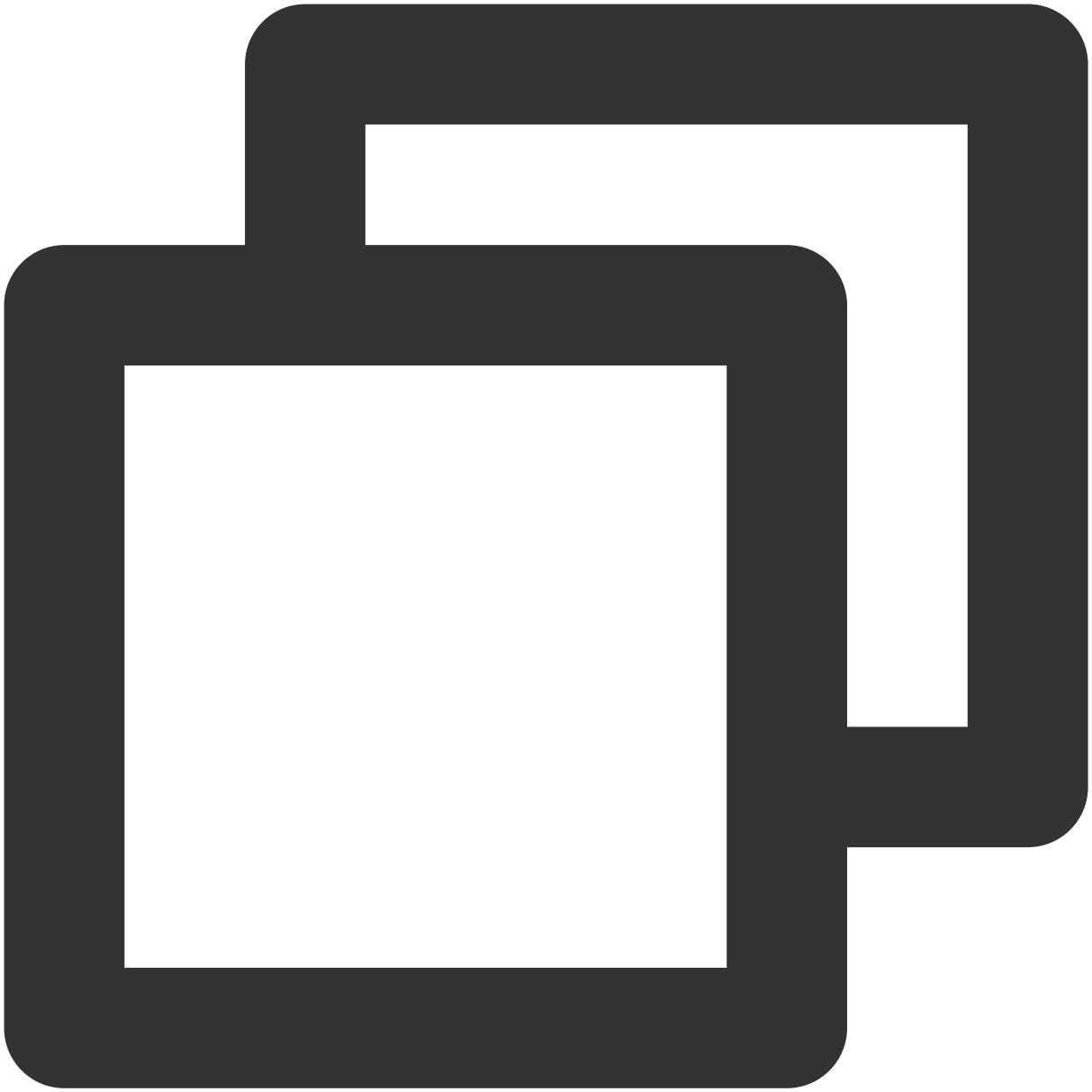


```
Future<TUIResult> hangup()
```

### inviteUser

This API is used to invite users to the current group call.

This API is called by a participant of a group call to invite new users.



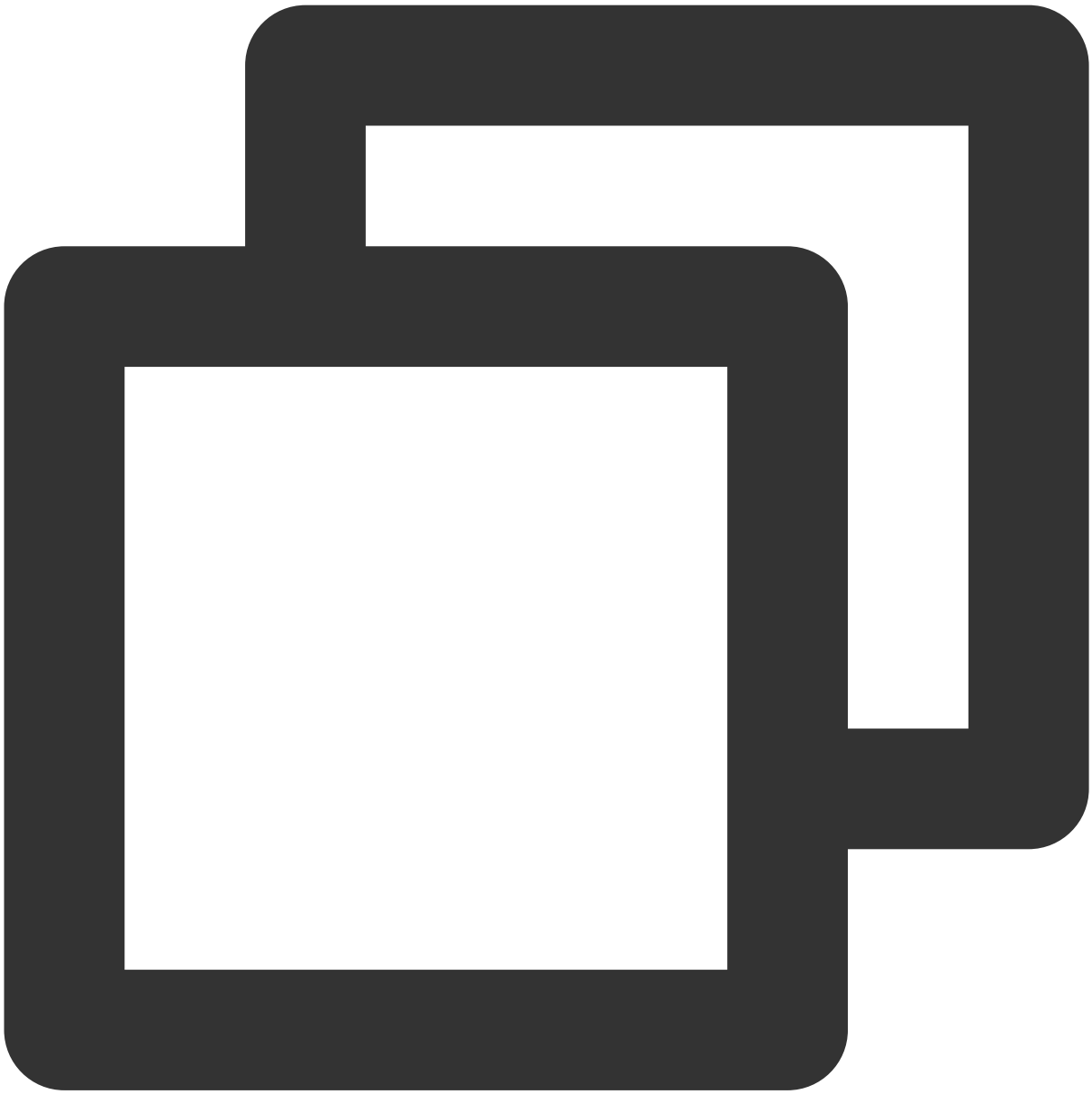
```
Future<void> iniviteUser(List<String> userIdList, TUICallParams params, TUIValueCal
```

| Parameter | Type | Description |
| --- | --- | --- |
| userIdList | List<String> | The target user IDs. |
| params | TUICallParams | An additional parameter. such as roomID, call timeout, offline push info, etc. |

### joinInGroupCall

This API is used to join a group call.

This API is called by a group member to join the group's call.



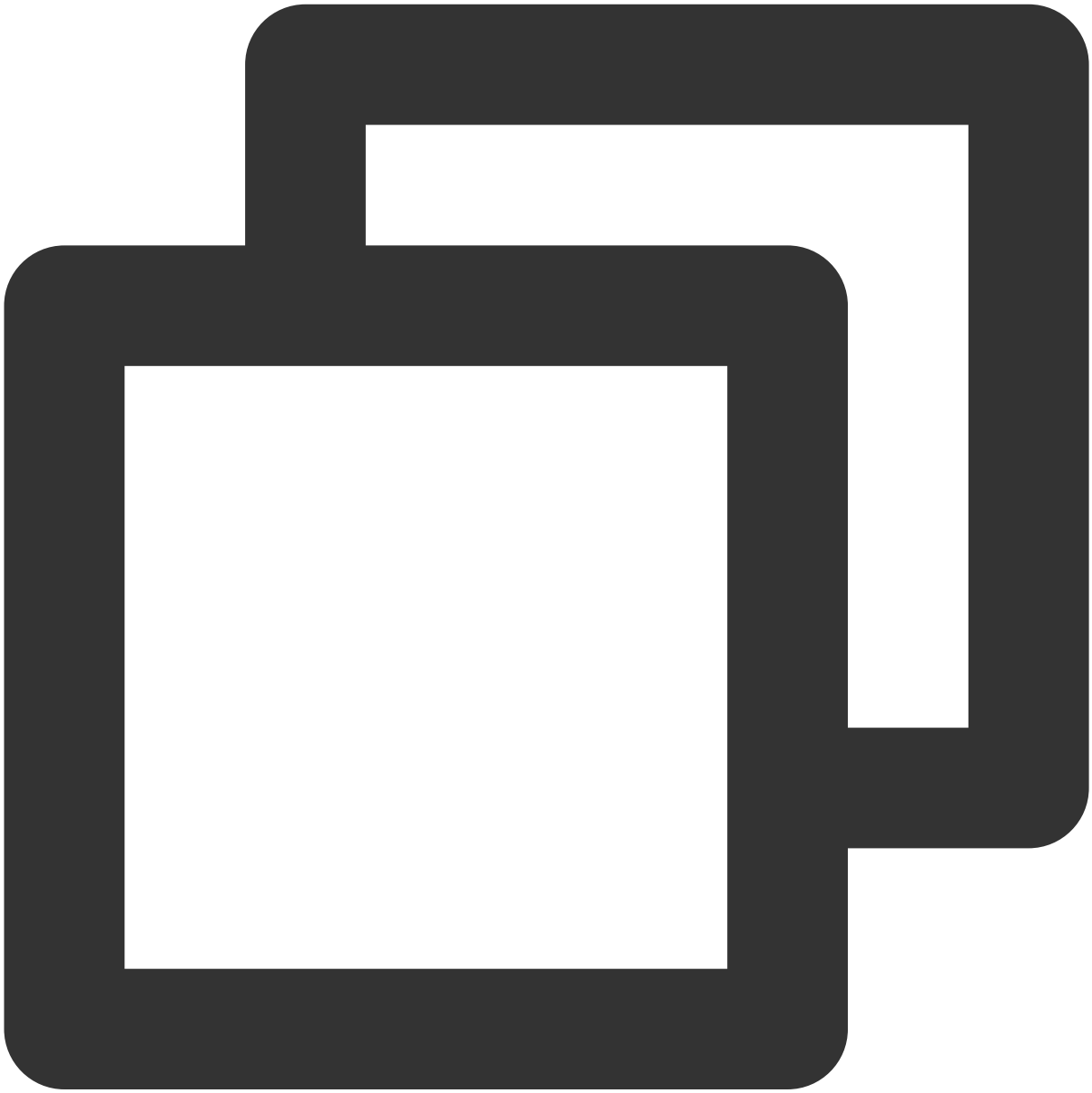
```
Future<TUIResult> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaTyp
```

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |
| groupId | String | The group ID. |
| | | |

| | | |
|---|---|---|
| mediaType | TUICallMediaType | The call type, which can be video or audio. |

### switchCallMediaType

This API is used to change the call type.

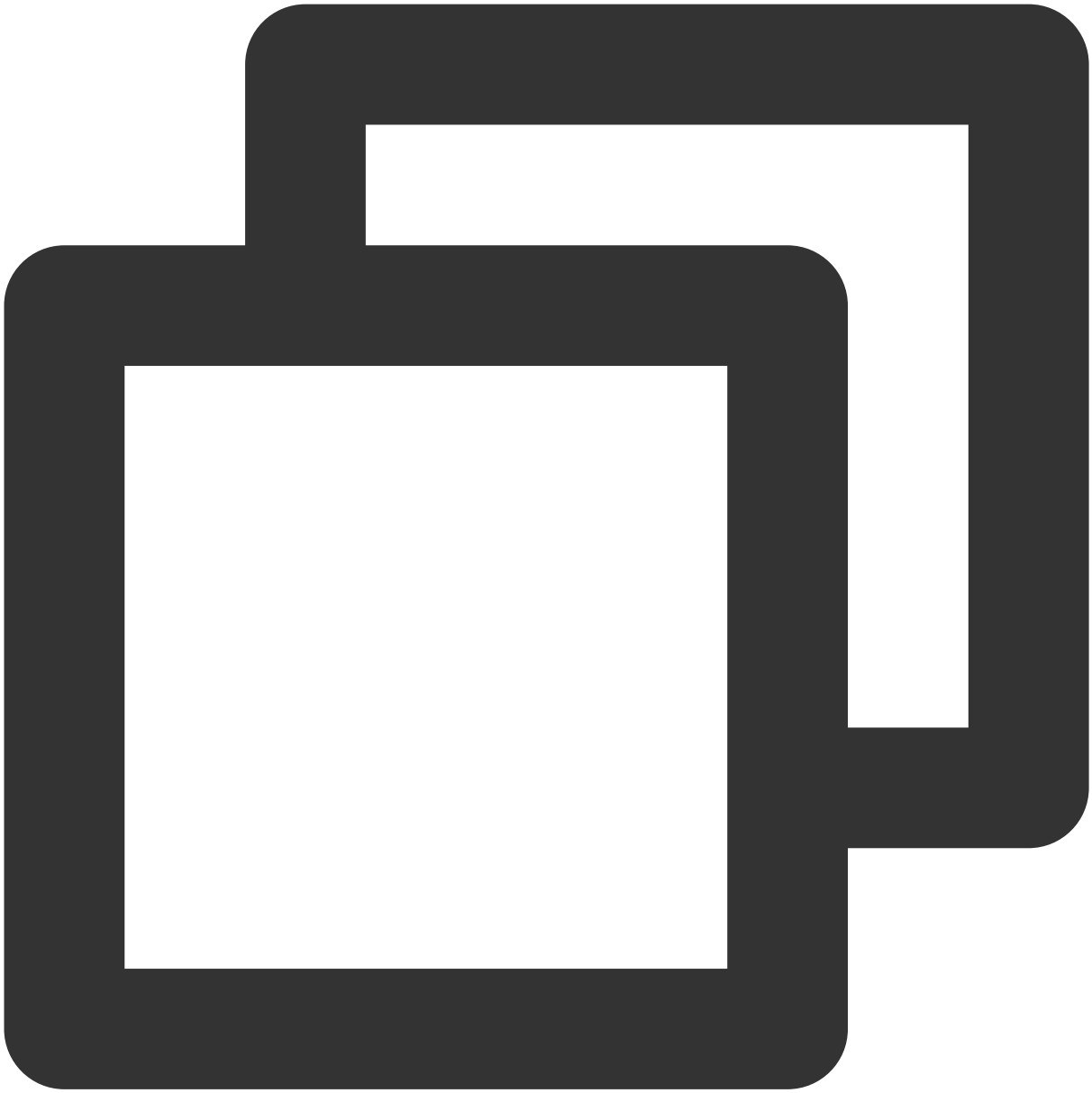


```
Future<void> switchCallMediaType(TUICallMediaType mediaType)
```

| Parameter | Type | Description |
|---|---|---|
| mediaType | TUICallMediaType | The call type, which can be video or audio. |

### startRemoteView

This API is used to set the view object to display a remote video.

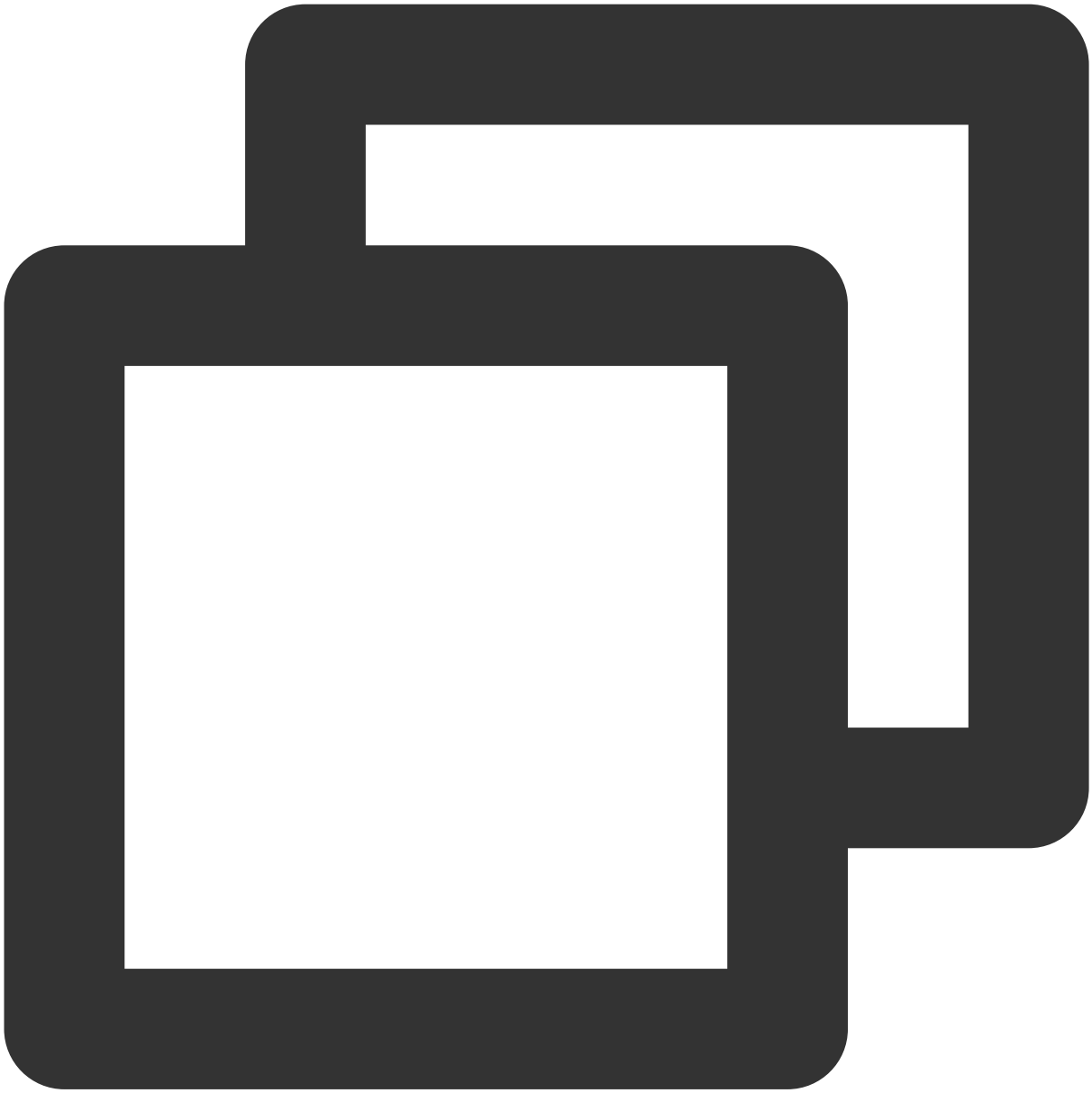


```
Future<void> startRemoteView(String userId, intviewId)
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| intviewId | int | The ID of the widget in the video rendering screen |

### stopRemoteview

This API is used to unsubscribe from the video stream of a remote user.

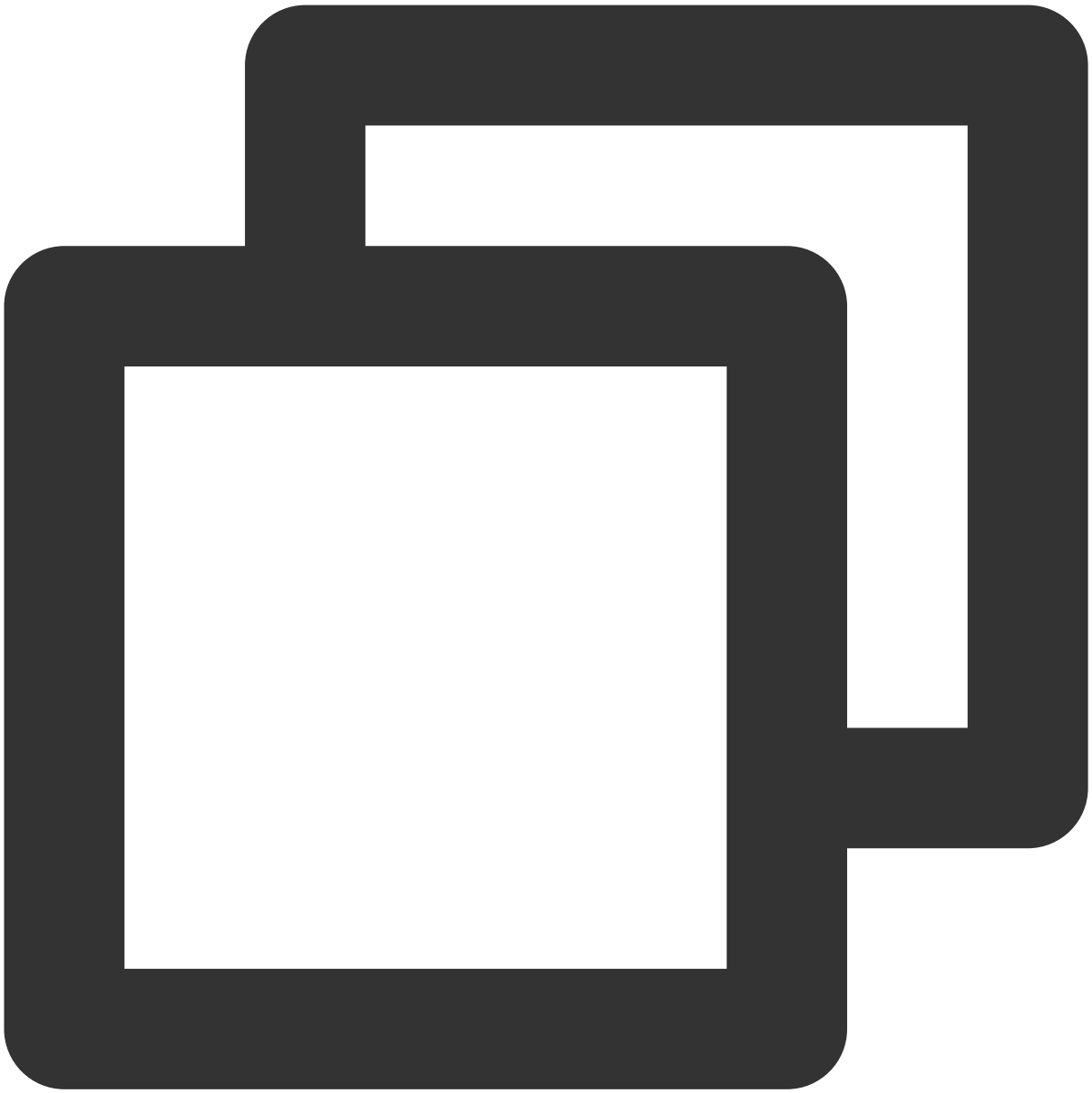


```
Future<void> stopRemoteView(String userId)
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |

### openCamera

This API is used to turn the camera on.

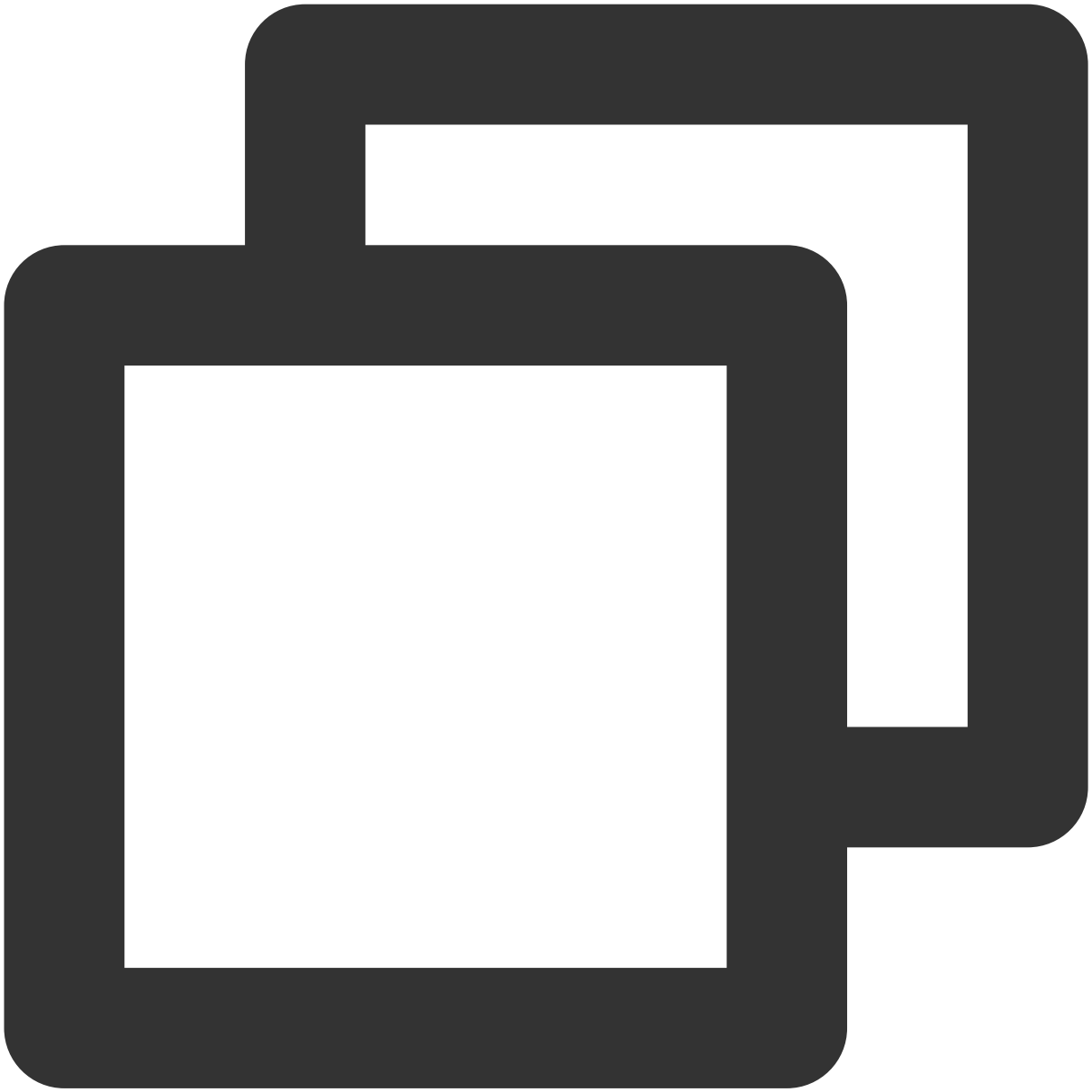


```
Future<TUIResult> openCamera(TUICamera camera, int? viewId)
```

| Parameter | Type | Description |
| --- | --- | --- |
| camera | TUICamera | The front or rear camera. |
| viewId | int | The ID of the widget in the video rendering screen |

### closeCamera

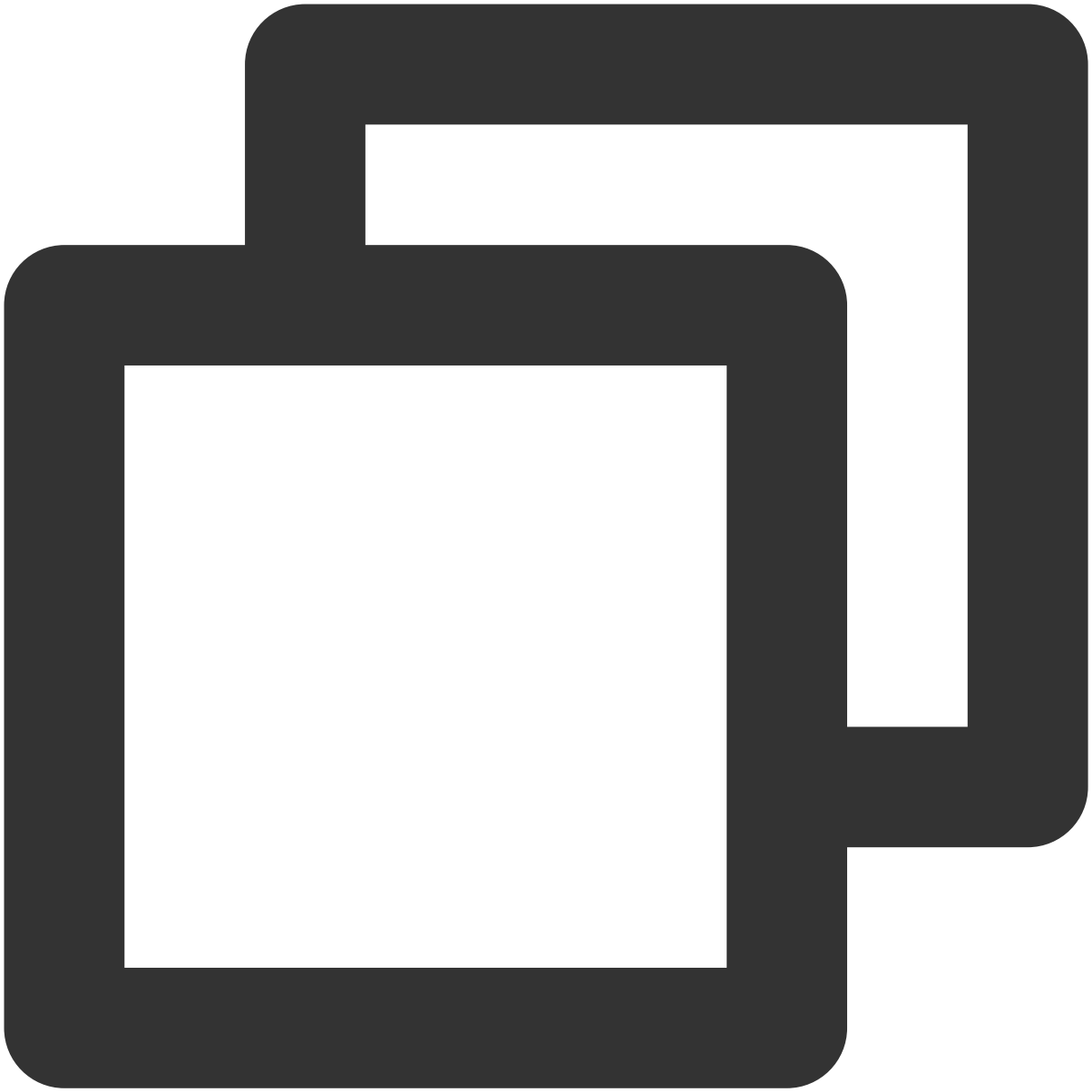
This API is used to turn the camera off.



```
Future<void> closeCamera()
```

### switchCamera

This API is used to switch between the front and rear cameras.

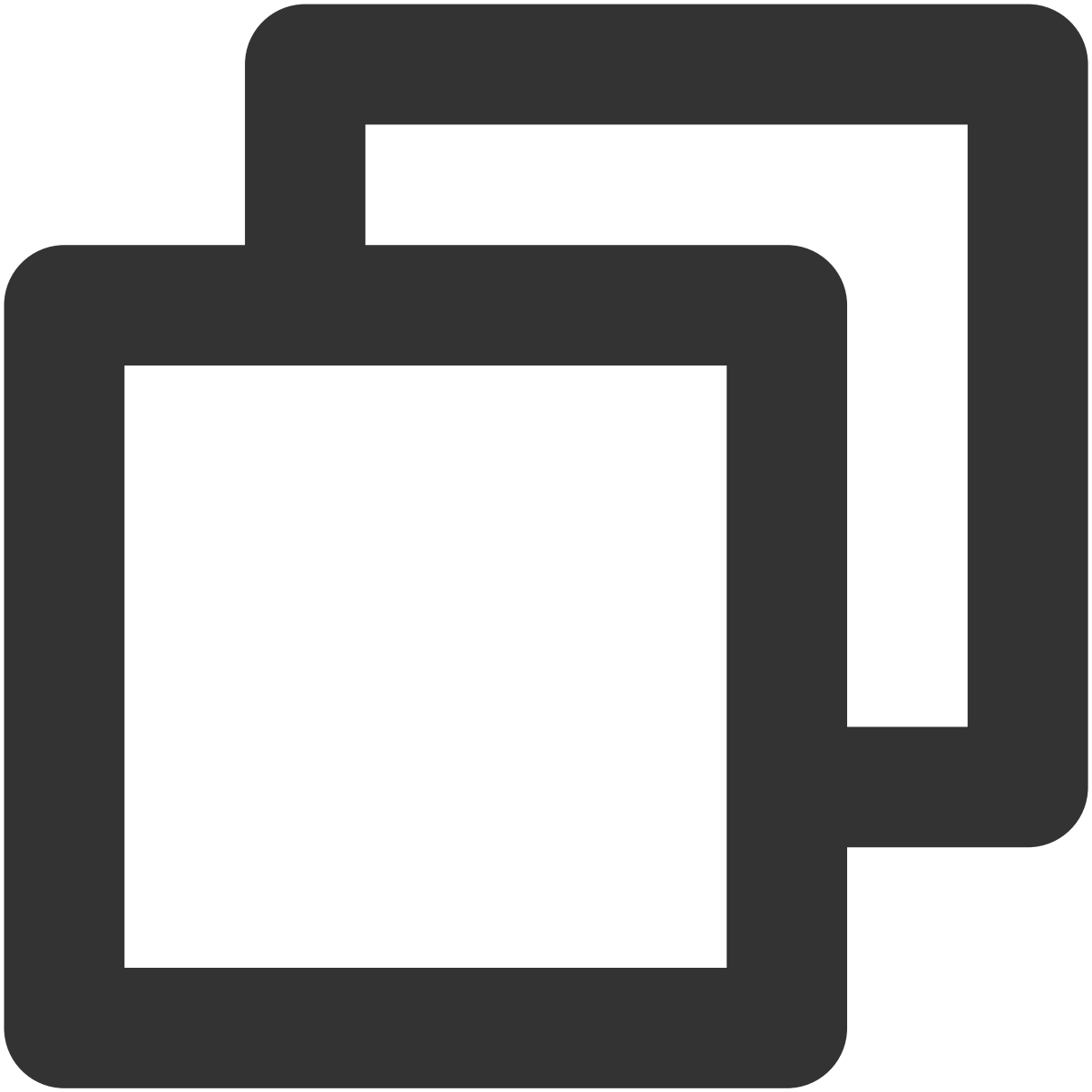


```
Future<void> switchCamera(TUICamera camera)
```

| Parameter | Type | Description |
| --- | --- | --- |
| camera | TUICamera | The front or rear camera. |

### openMicrophone

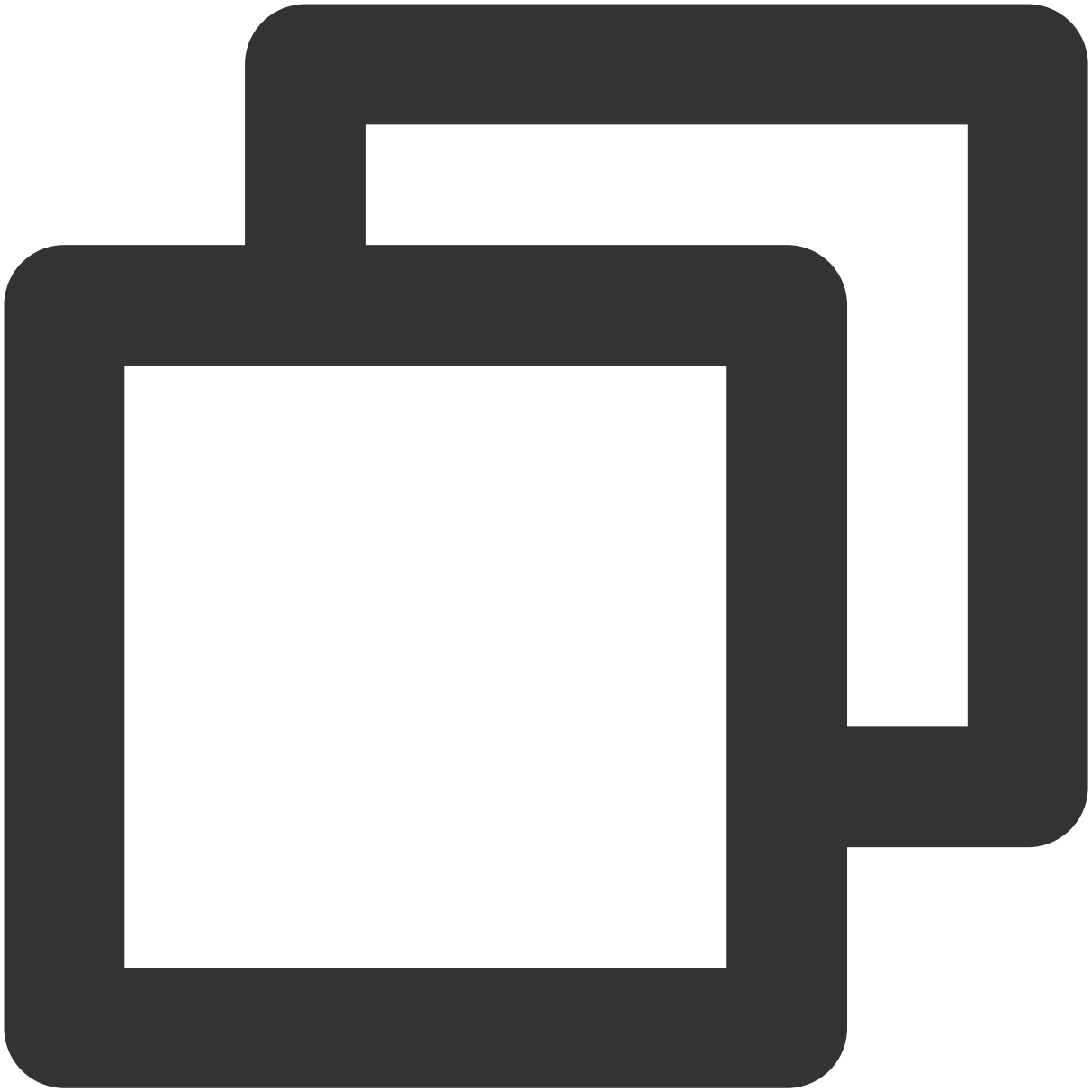
This API is used to turn the mic on.



```
Future<TUIResult> openMicrophone()
```

### closeMicrophone

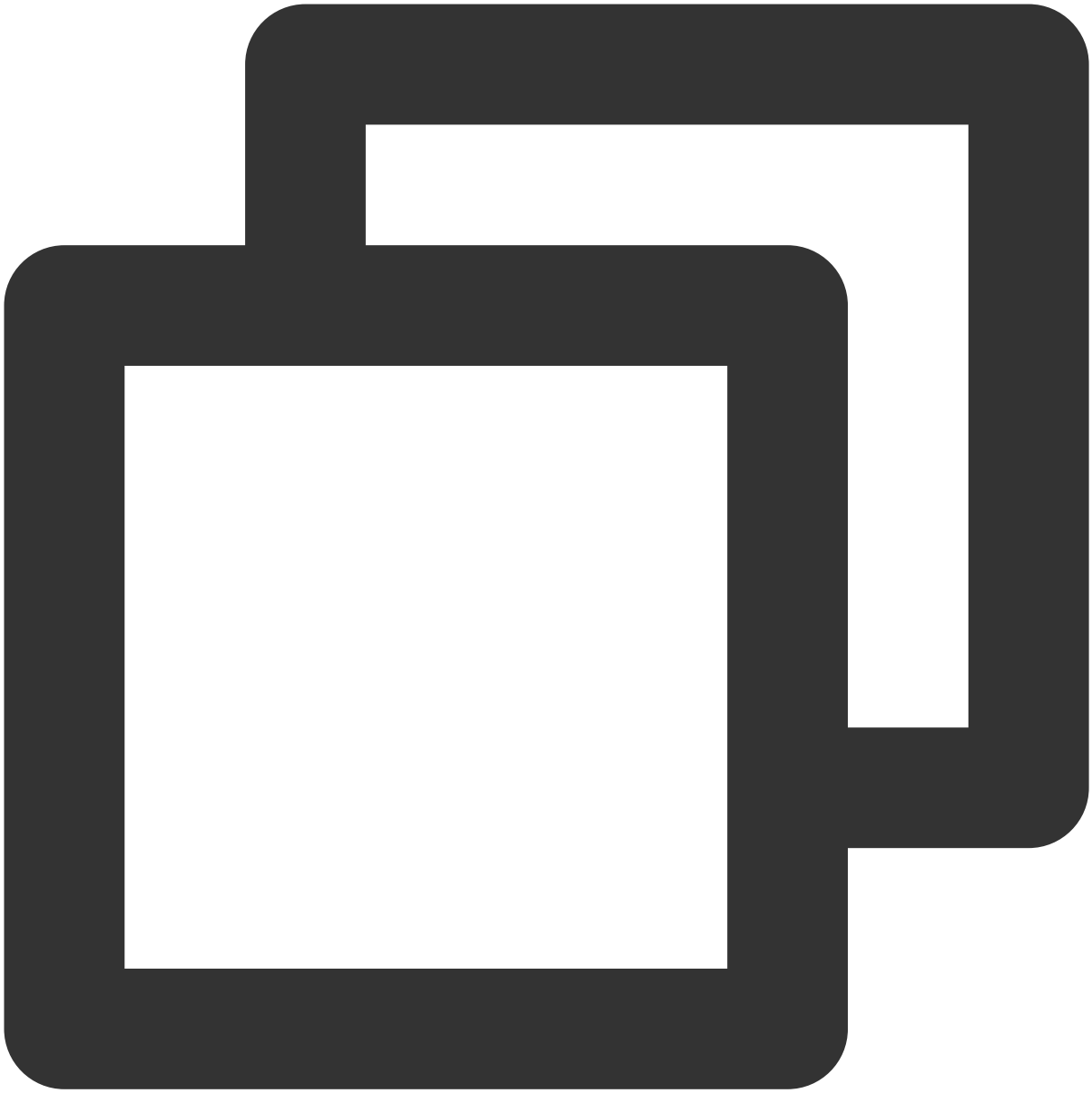
This API is used to turn the mic off.



```
Future<void> closeMicrophone()
```

### selectAudioPlaybackDevice

This API is used to select the audio playback device (receiver or speaker). In call scenarios, you can use this API to turn on/off hands-free mode.

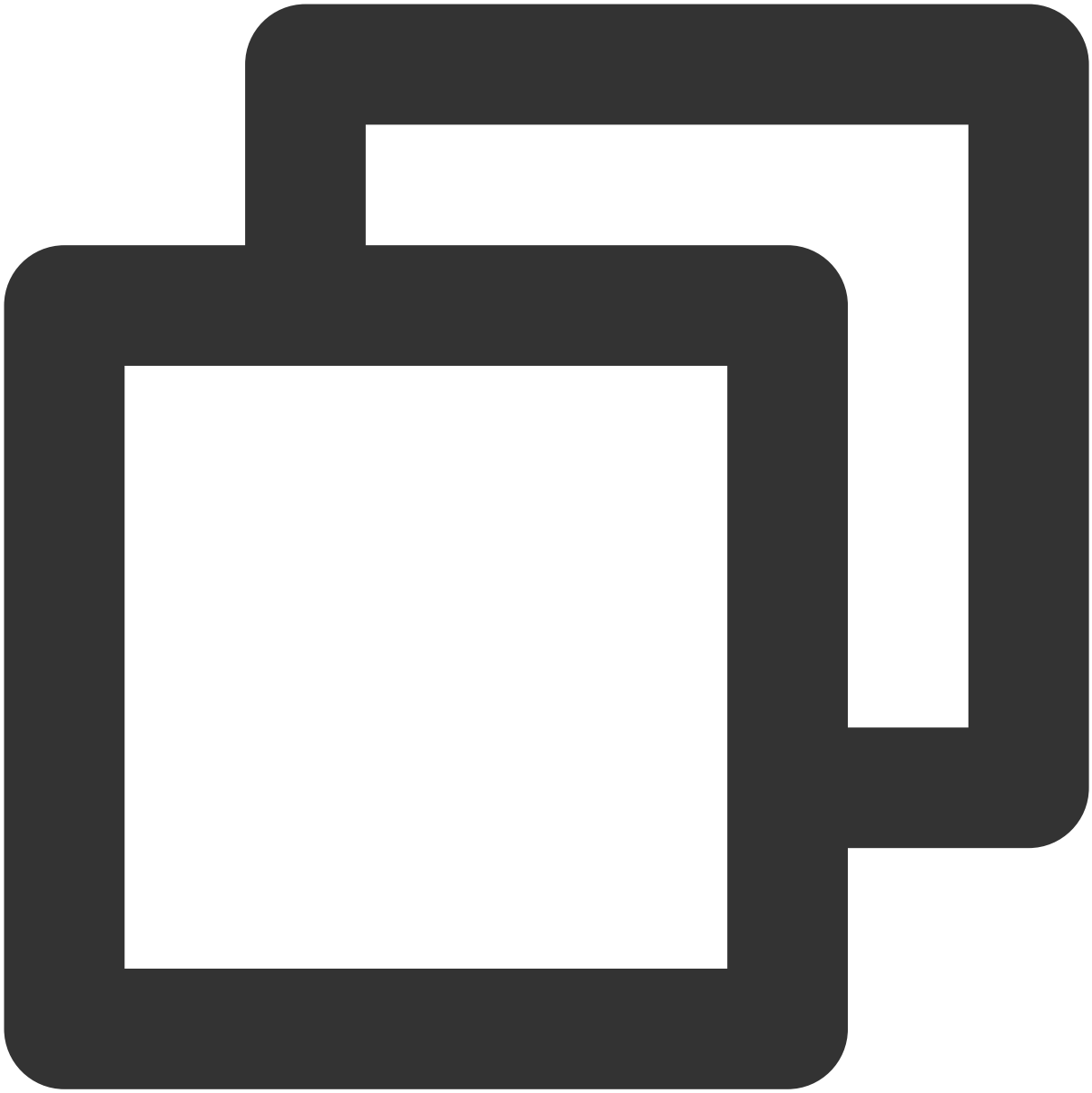


```
Future<void> selectAudioPlaybackDevice(TUIAudioPlaybackDevice device)
```

| Parameter | Type | Description |
| --- | --- | --- |
| device | TUIAudioPlaybackDevice | The speaker or receiver. |

### setSelfInfo

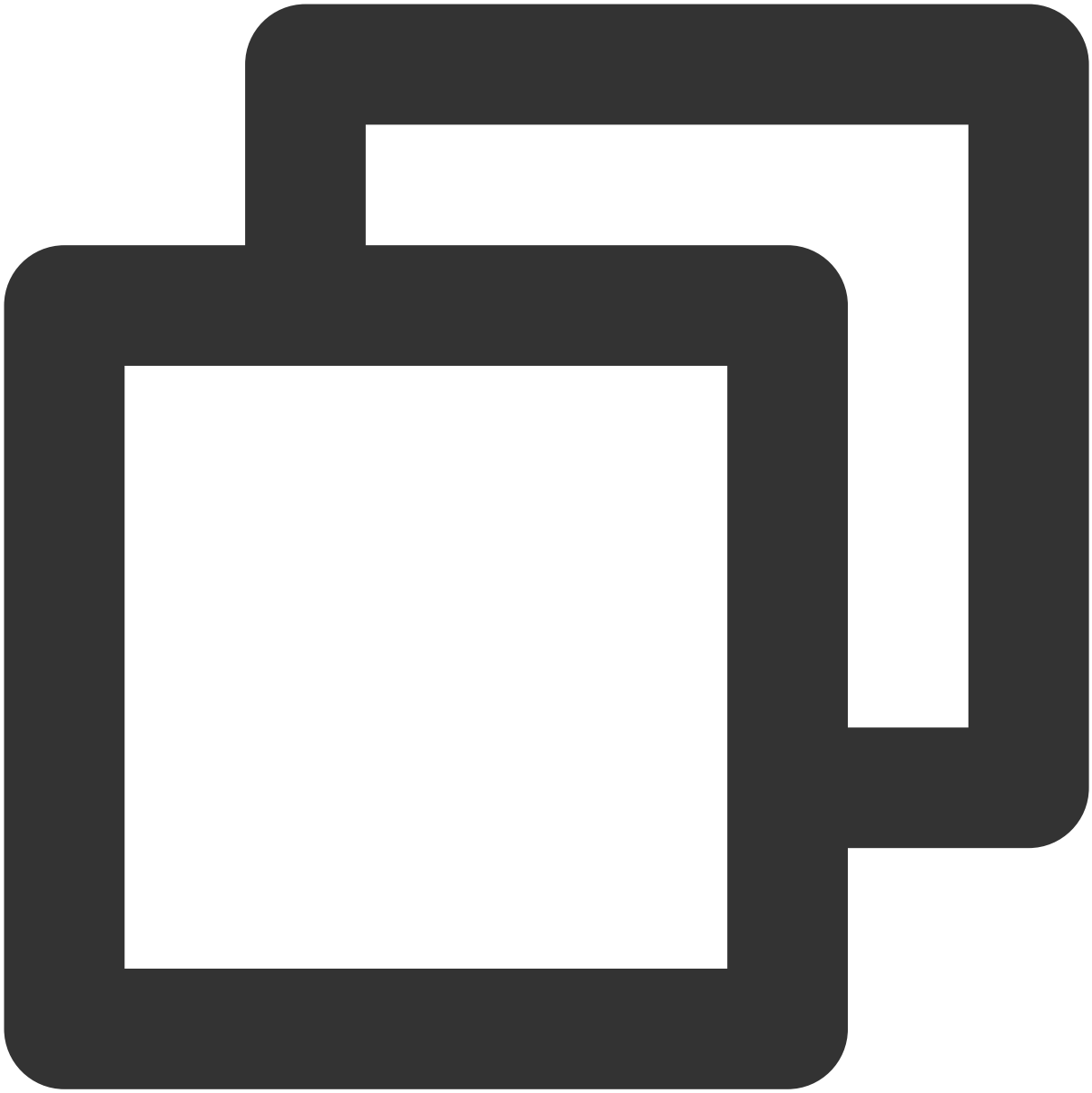
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

### enableMultiDeviceAbility

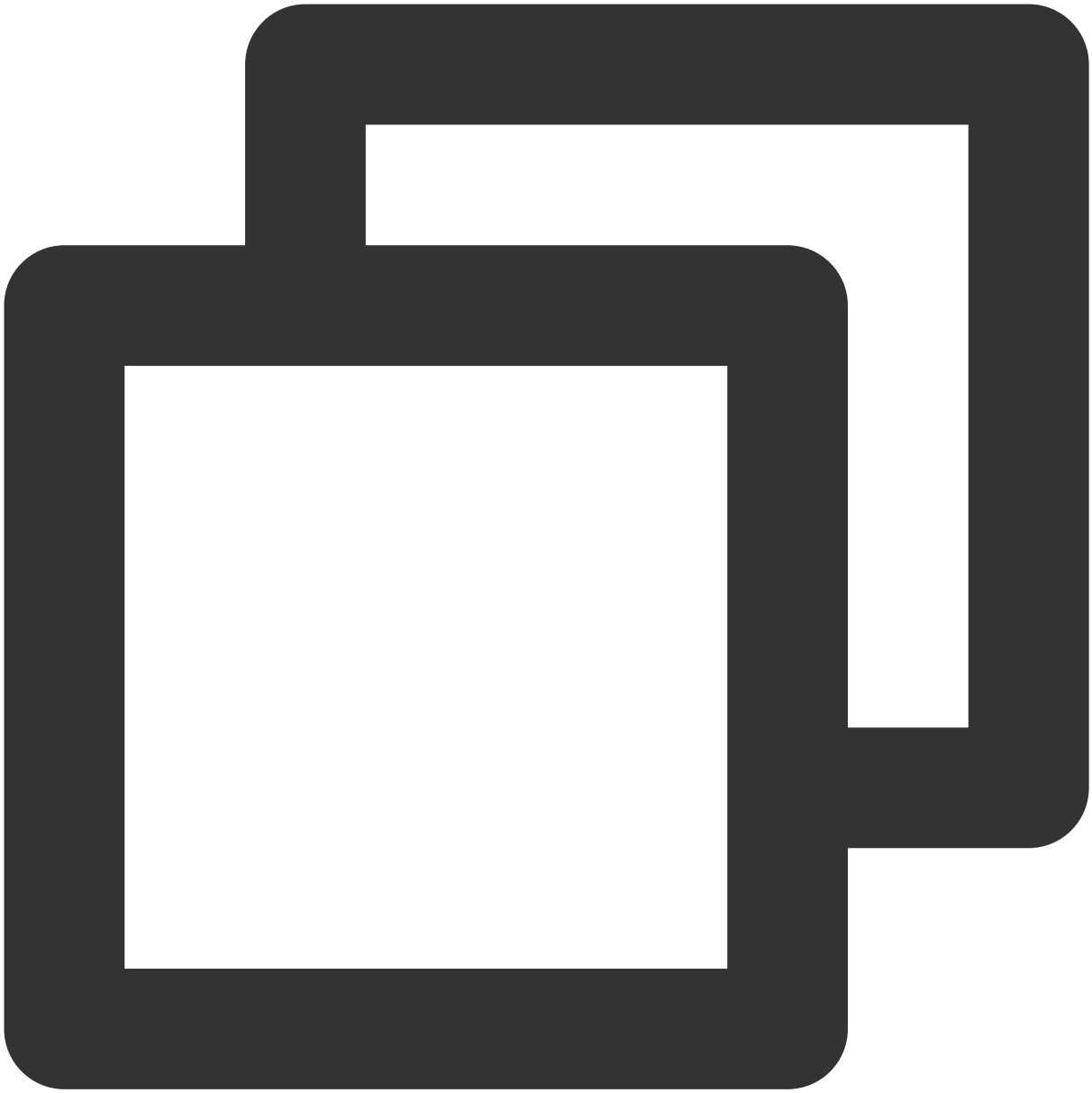
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the premium package).



```
Future<TUIResult> enableMultiDeviceAbility(bool enable)
```

### setVideoRenderParams

Set the rendering mode of video image.



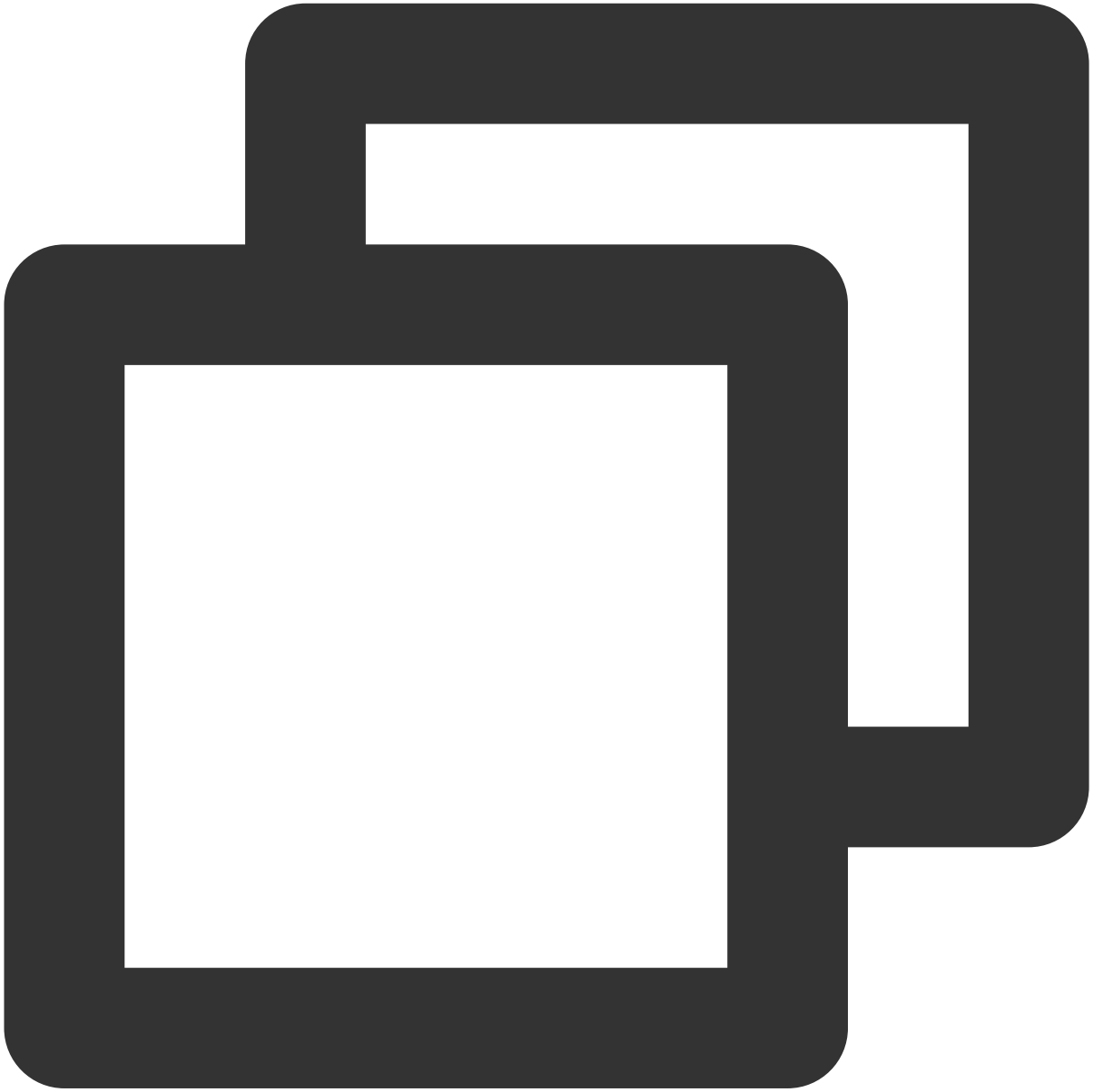
```
Future<TUIResult> setVideoRenderParams(String userId, VideoRenderParams params)
```

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The target user ID. |
| params | VideoRenderParams | Video render parameters. |

### setVideoEncoderParams

Set the encoding parameters of video encoder.

This setting can determine the quality of image viewed by remote users, which is also the image quality of on-cloud recording files.

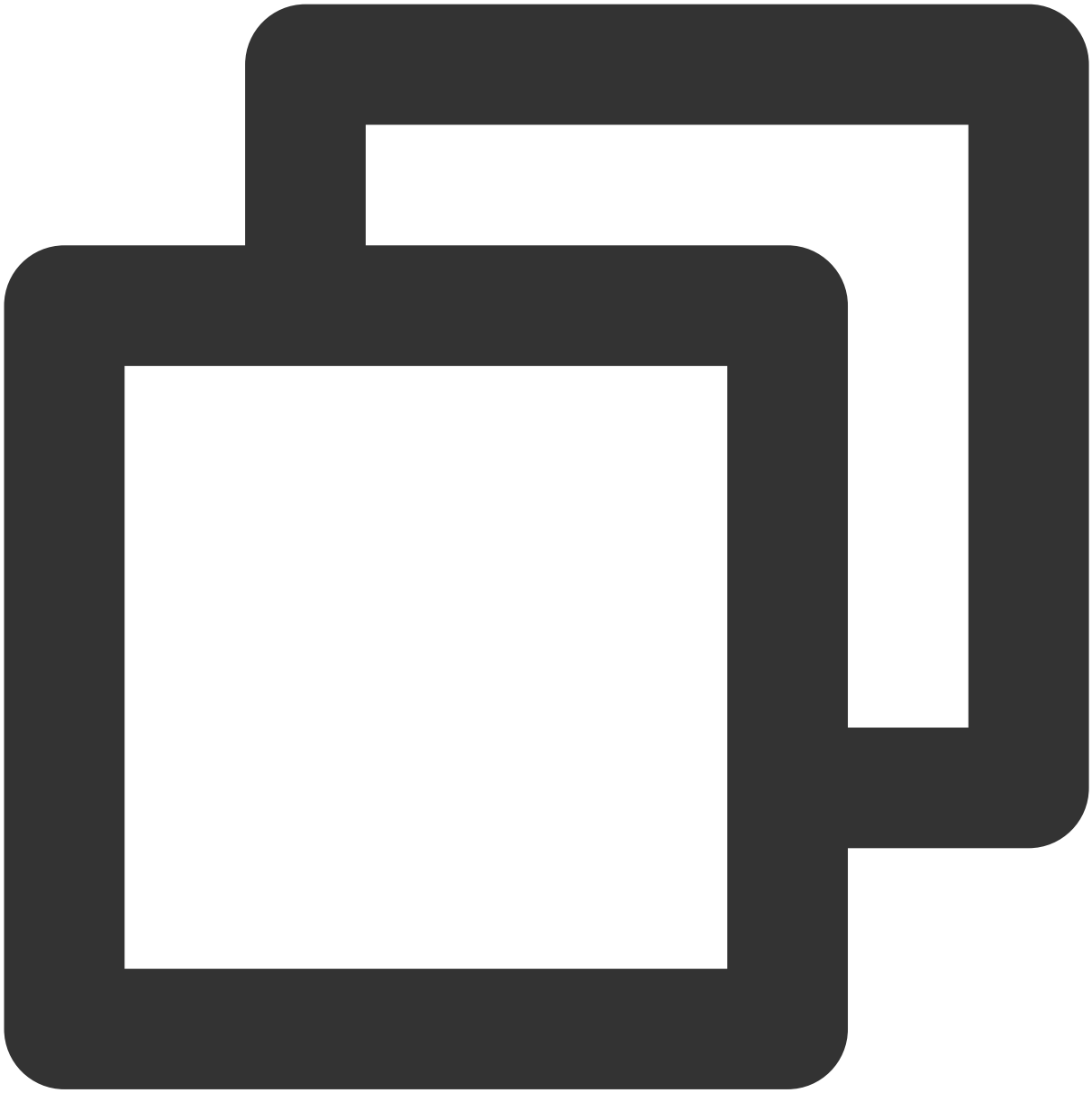


```
Future<TUIResult> setVideoEncoderParams(VideoEncoderParams params)
```

| Parameter | Type | Description |
| --- | --- | --- |
| params | VideoEncoderParams | Video encoding parameters |

### queryRecentCalls

Query call record.

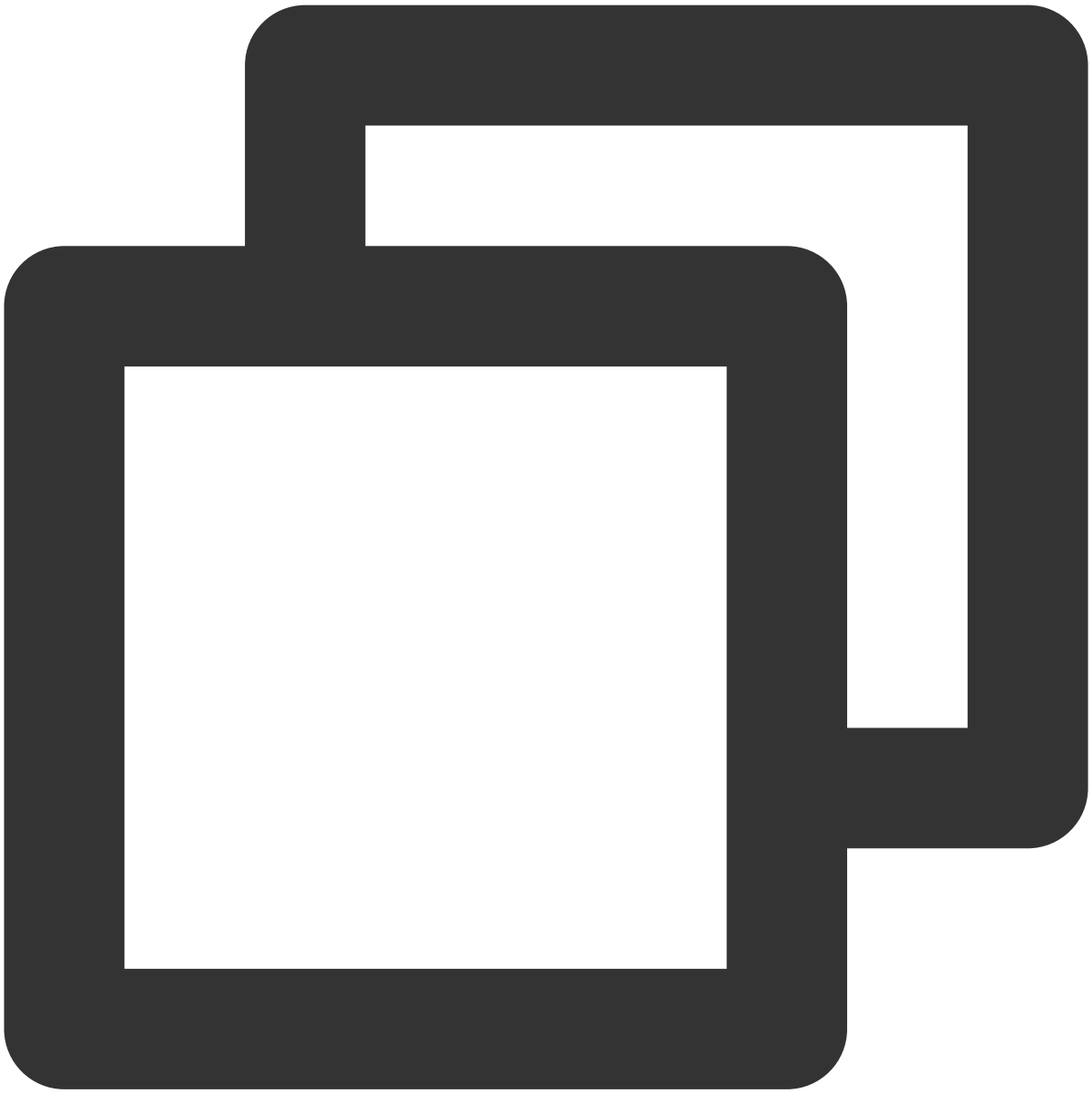


```
Future<void> queryRecentCalls(TUICallRecentCallsFilter filter, TUIValueCallback cal
```

| Parameter | Type | Description |
| --- | --- | --- |
| filter | TUICallRecentCallsFilter | Filter condition |

### deleteRecordCalls

Delete call record.

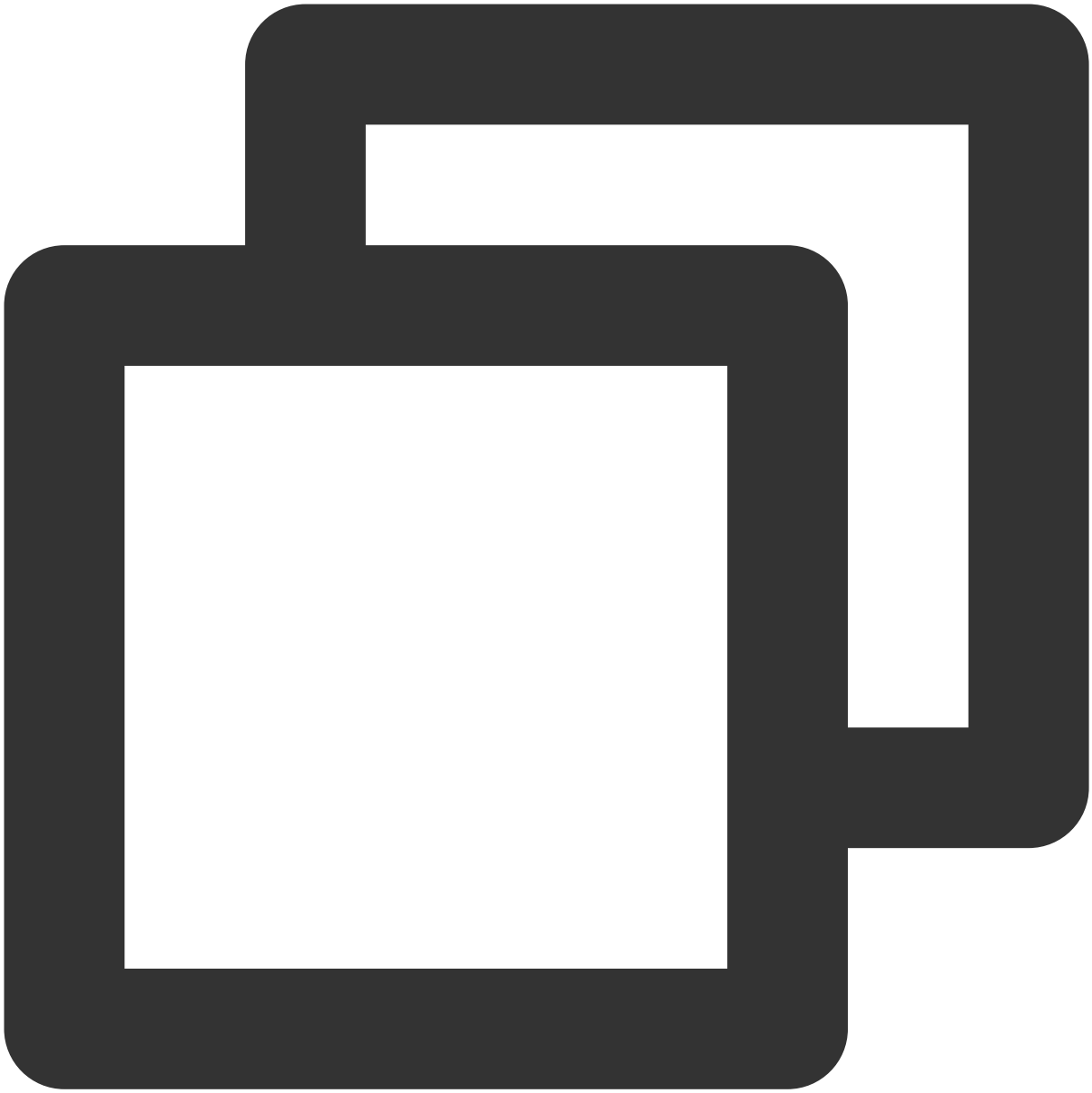


```
Future<void> deleteRecordCalls(List<String> callIdList, TUIValueCallback callback)
```

| Parameter | Type | Description |
| --- | --- | --- |
| callIdList | List<String> | List of IDs of records to be deleted. |

### setBeautyLevel

Set beauty level, support turning off default beauty.



```
Future<TUIResult> setBeautyLevel(double level)
```

| Parameter | Type | Description |
| --- | --- | --- |
| level | double | Beauty level, range 0.0 to 9.0. |

# TUICallObserver

Last updated：2024-01-25 16:18:11

## TUICallObserver APIs

`TUICallObserver` is the callback class of `TUICallEngine`. You can use it to listen for events.

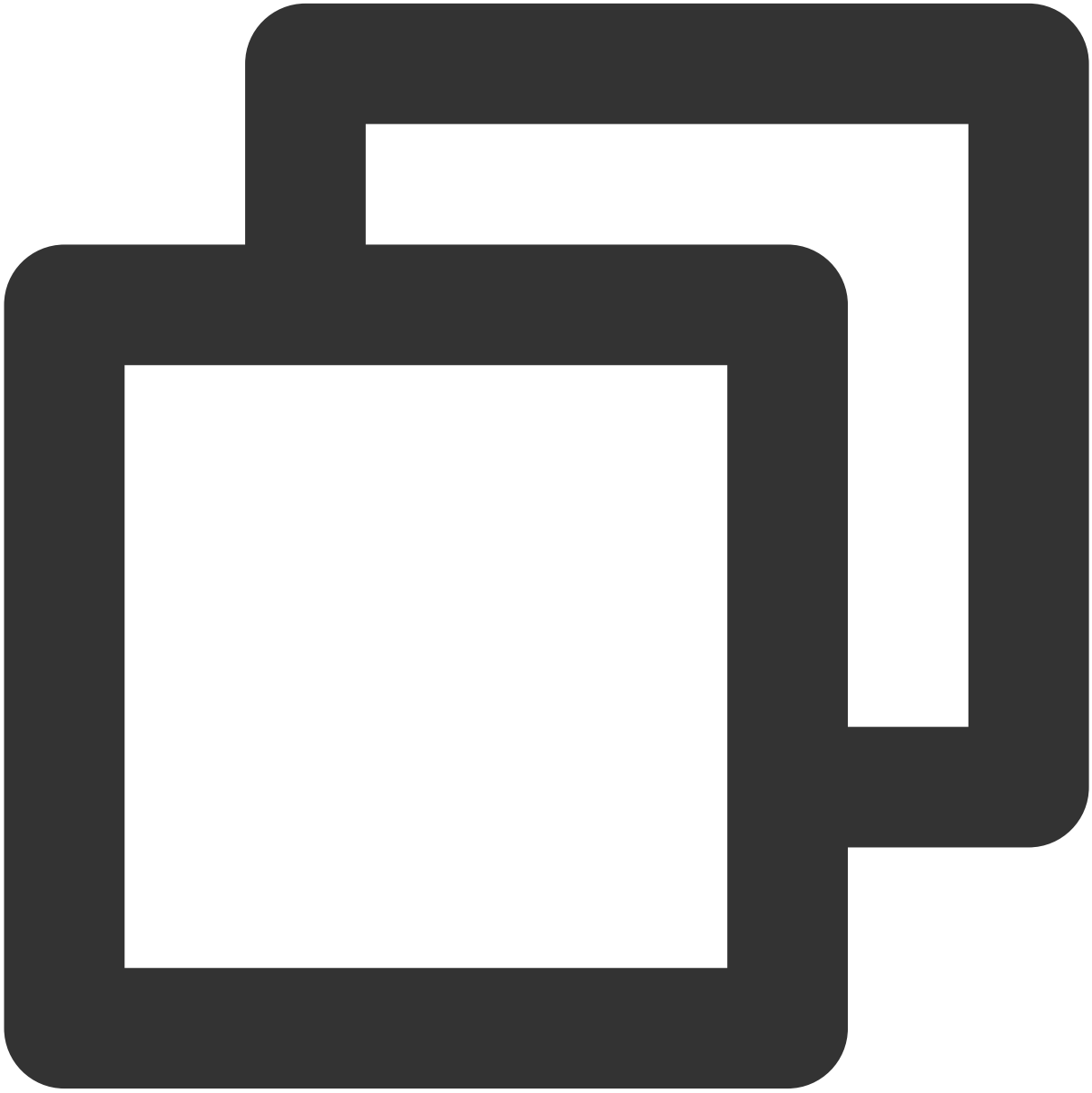
## Overview

| API | Description |
| --- | --- |
| onError | A call occurred during the call. |
| onCallReceived | A call invitation was received. |
| onCallCancelled | The call was canceled. |
| onCallBegin | The call was connected. |
| onCallEnd | The call ended. |
| onCallMediaTypeChanged | The call media type changed. |
| onUserReject | A user declined the call. |
| onUserNoResponse | A user didn't respond. |
| onUserLineBusy | A user was busy. |
| onUserJoin | A user joined the call. |
| onUserLeave | A user left the call. |
| onUserVideoAvailable | Whether a user had a video stream. |
| onUserAudioAvailable | Whether a user had an audio stream. |
| onUserVoiceVolumeChanged | The volume levels of all users. |
| onUserNetworkQualityChanged | The network quality of all users. |
| onKickedOffline | current user is logged out |
| | |

| | |
|---|---|
| onUserSigExpired | Token Expiration |

## Details

Listen to the events thrown by `addObserver` .



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {
```

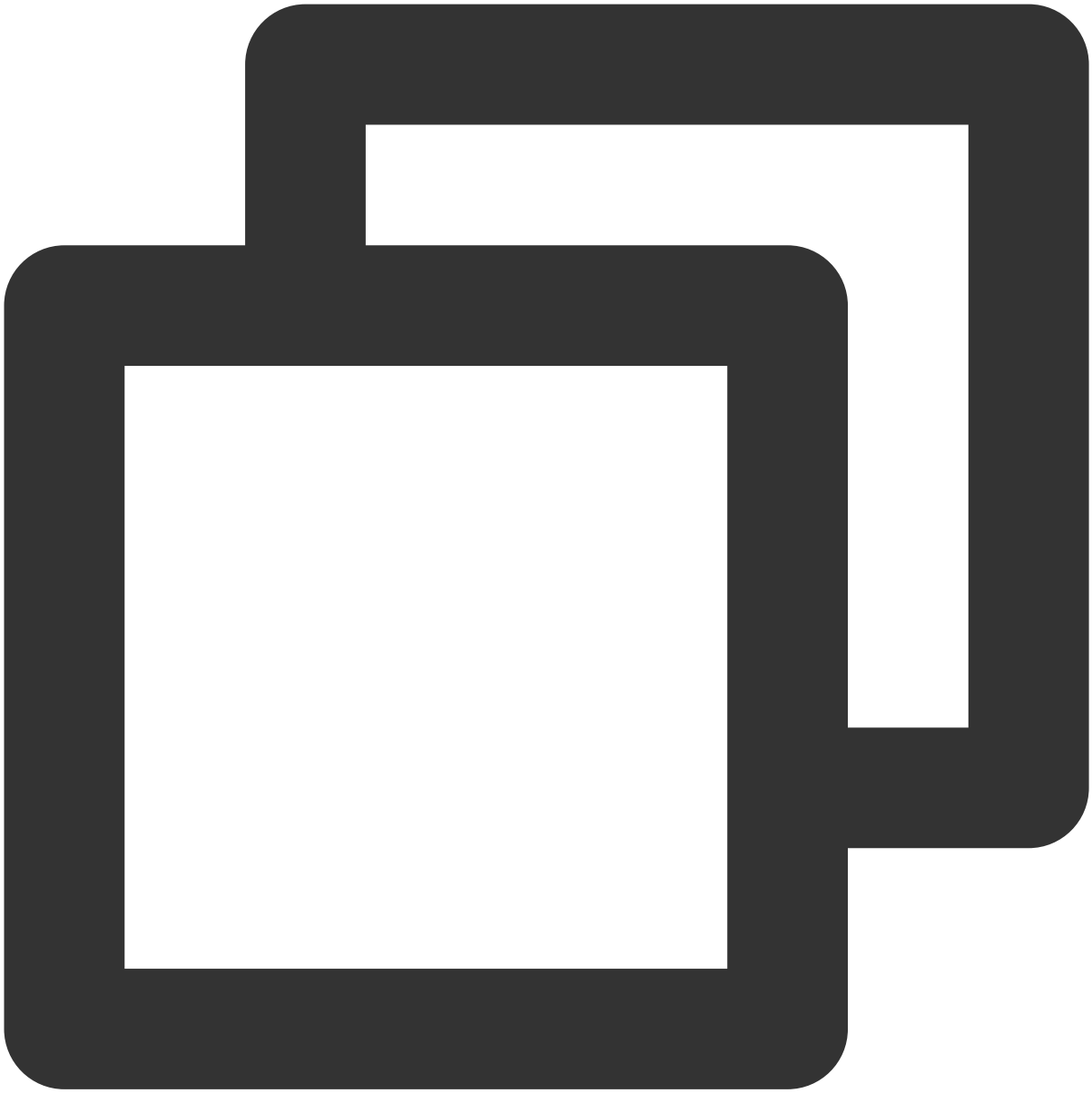
```
    },onCallCancelled: (String callerId) {

    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole

    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca

    }, onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType

    }, onUserReject: (String userId) {

    }, onUserNoResponse: (String userId) {

    }, onUserLineBusy: (String onUserLineBusy) {

    }, onUserJoin: (String userId) {

    }, onUserLeave: (String userId) {

    }, onUserVideoAvailable: (String userId, bool isVideoAvailable) {

    }, onUserAudioAvailable: (String userId, bool isAudioAvailable) {

    }, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList

    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,

    }, onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {

    }, onKickedOffline: () {

    }, onUserSigExpired: () {

    }
));
```

### onError

An error occurred.

**Note**

This callback indicates that the SDK encountered an unrecoverable error. Such errors must be listened for, and UI reminders should be sent to users if necessary.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {

    }
));
```

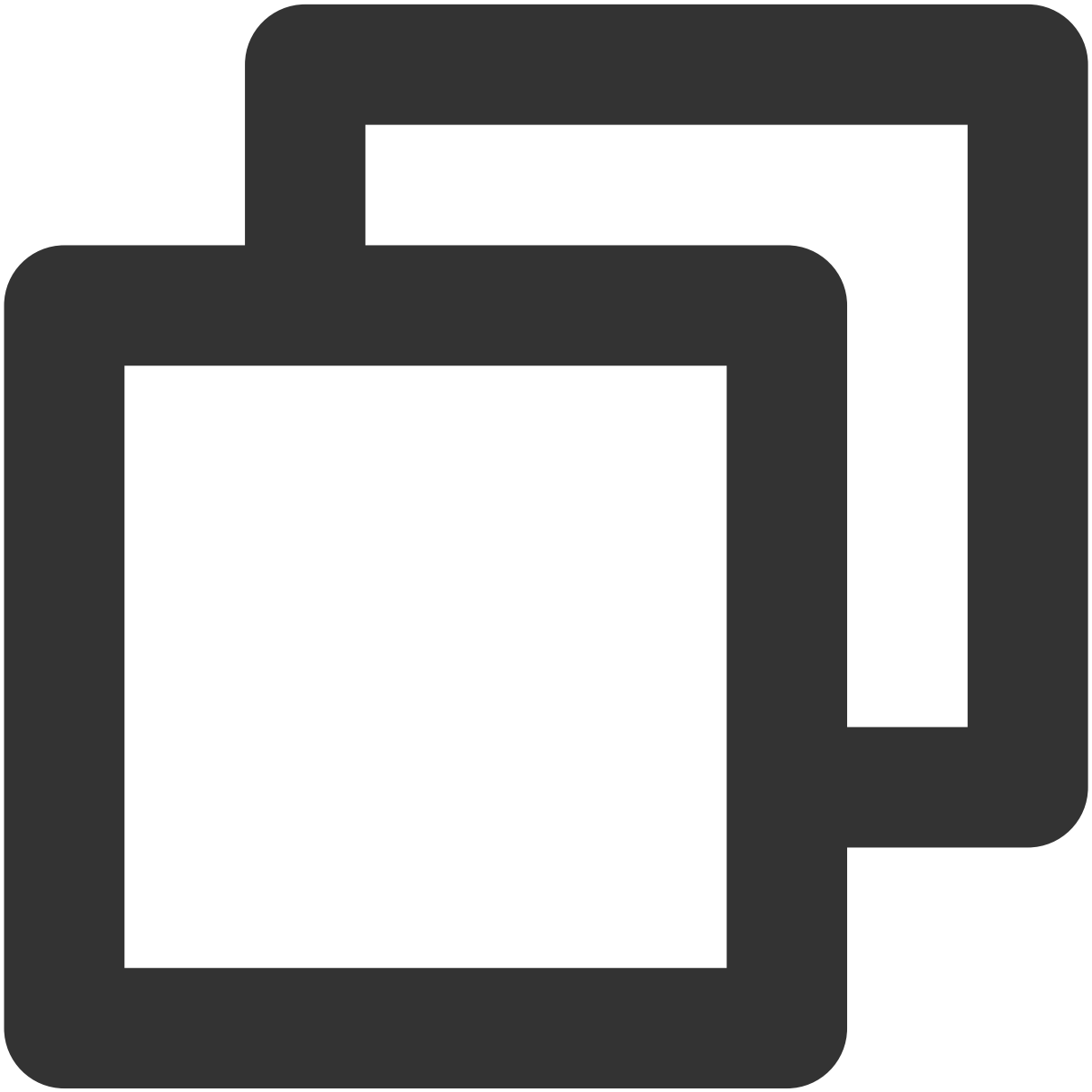
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| code | int | The error code. |

| | | |
|---|---|---|
| message | String | The error message. |

### onCallReceived

A call invitation was received. This callback is received by an invitee. You can listen for this event to determine whether to display the incoming call view.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallReceived: (String callerId, List<String> calleeIdList, String groupId, TU

    }
```

```
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| callerId | String | The user ID of the inviter. |
| calleeIdList | List<String> | The invitee list. |
| groupId | String | The group ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |

### onCallCancelled

The call was canceled by the inviter or timed out. This callback is received by an invitee. You can listen for this event to determine whether to show a missed call message.

This indicates that the call was canceled by the caller, timed out by the callee, rejected by the callee, or the callee was busy. There are multiple scenarios involved. You can listen to this event to achieve UI logic such as missed calls and resetting UI status.

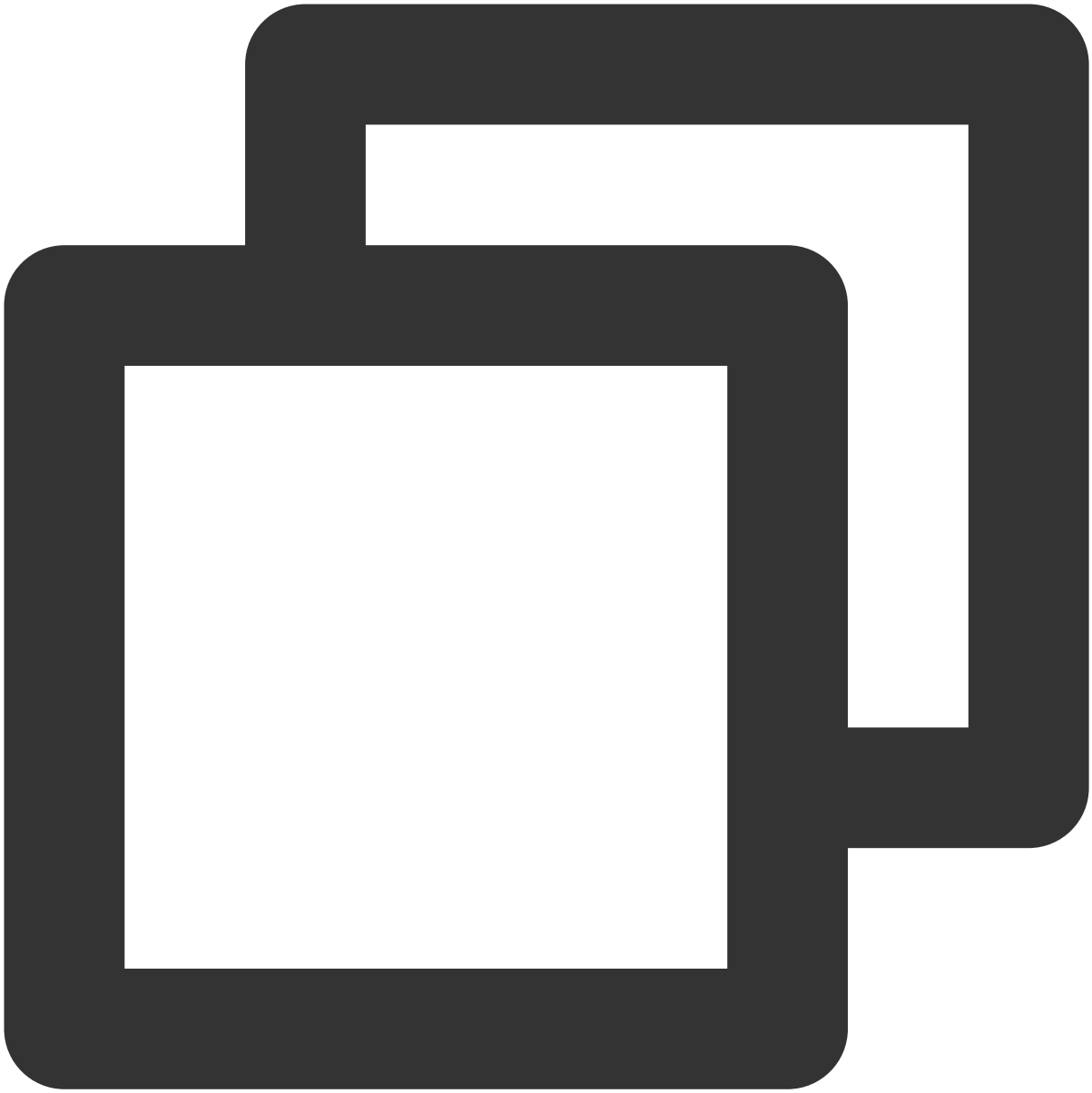
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller)

Callee timeout: the caller will simultaneously receive the onUserNoResponse and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID).

Callee rejection: The caller will simultaneously receive the onUserReject and onCallCancelled callbacks (userId is his own ID); the callee receives the onCallCancelled callback (userId is his own ID).

Callee busy: The caller will simultaneously receive the onUserLineBusy and onCallCancelled callbacks (userId is his own ID).

Abnormal interruption: The callee failed to receive the call， he receives this callback (userId is his own ID).



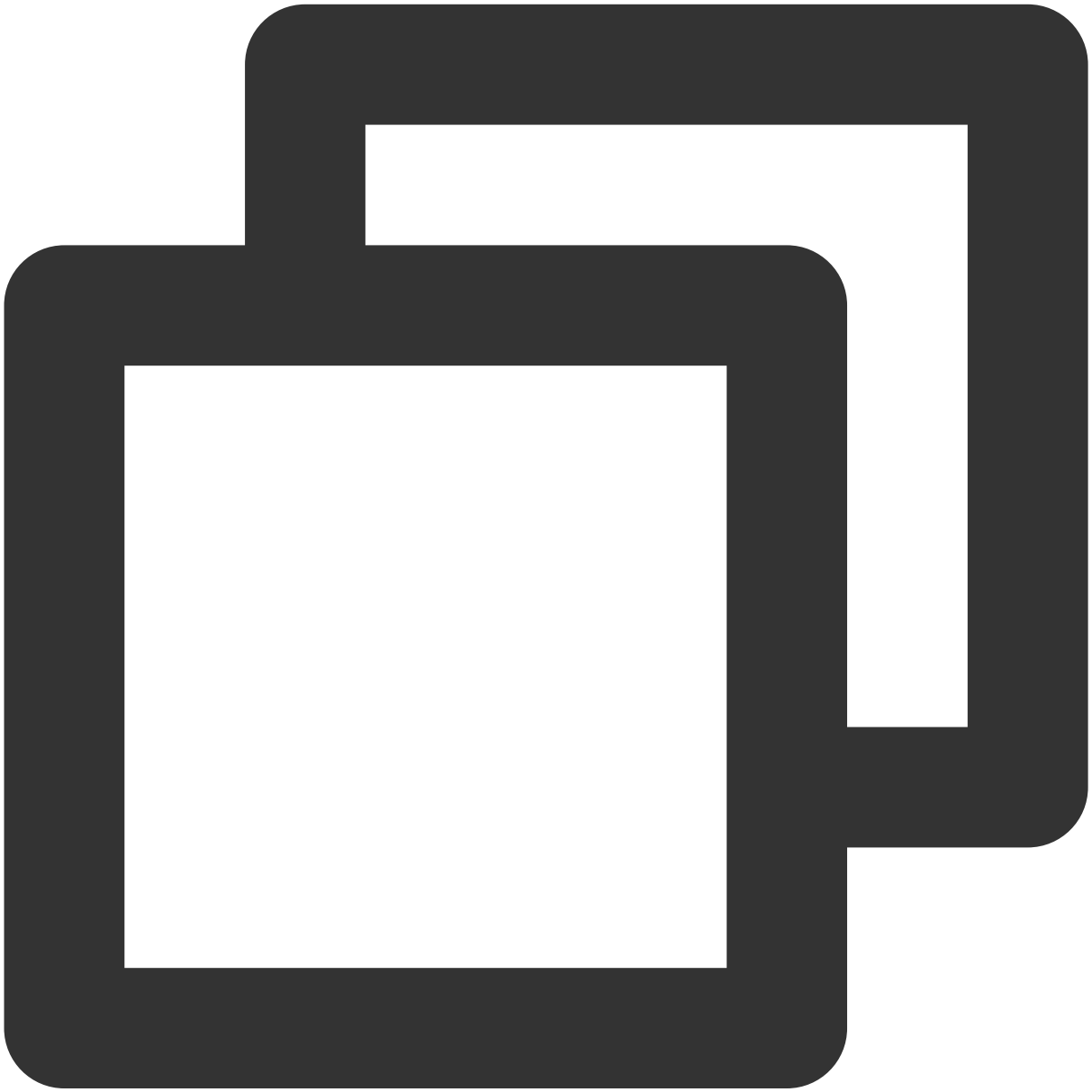
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallCancelled: (String callerId) {

    }
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| callerId | String | The user ID of the inviter. |

### onCallBegin

The call was connected. This callback is received by both the inviter and invitees. You can listen for this event to determine whether to start on-cloud recording, content moderation, or other tasks.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole cal
    }
));
```

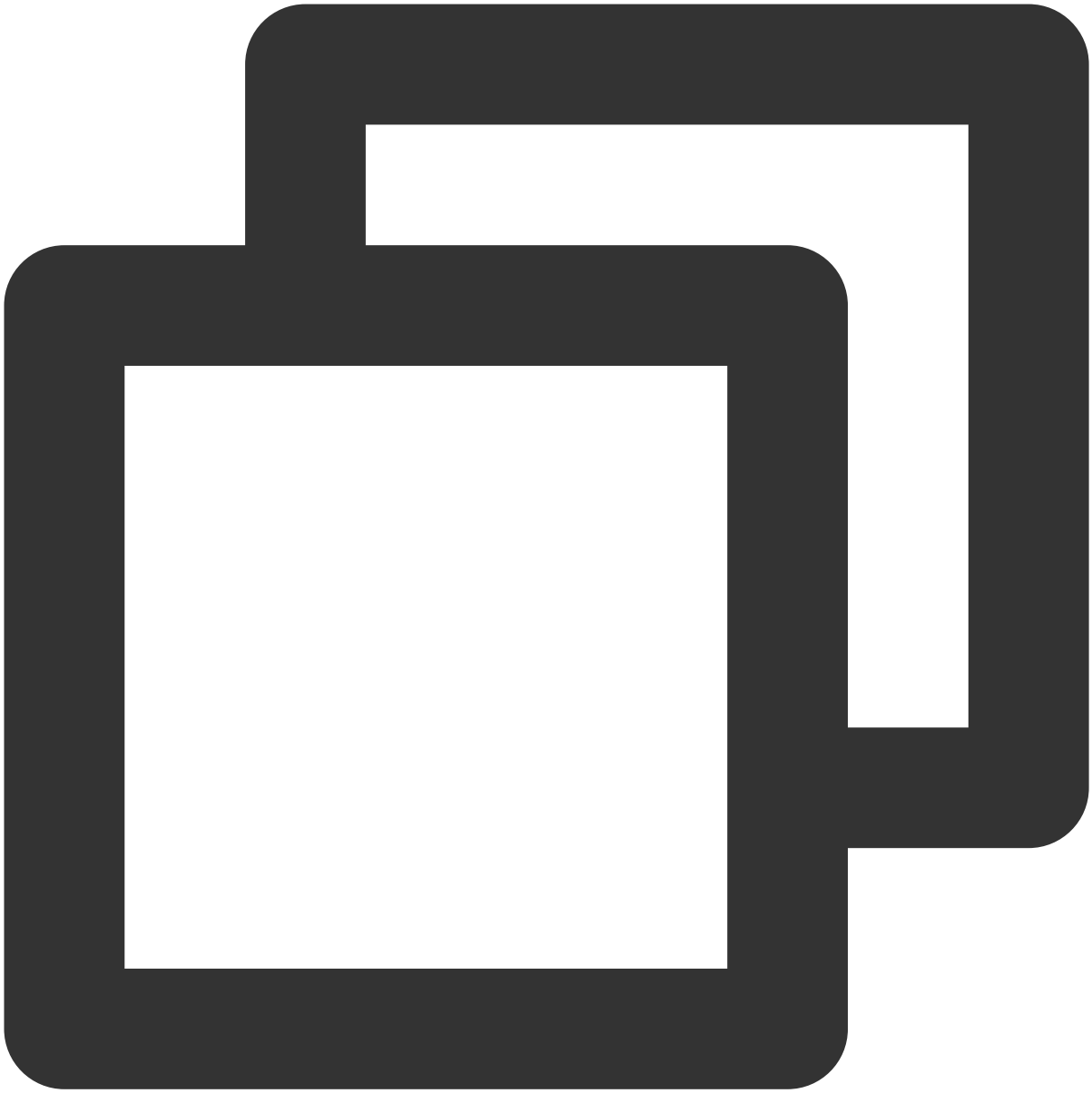
The parameters are described below:

| | | |
|---|---|---|

| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| callRole | TUICallRole | The role, which can be caller or callee. |

### onCallEnd

The call ended. This callback is received by both the inviter and invitees. You can listen for this event to determine when to display call information such as call duration and call type, or stop on-cloud recording.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callR

    }
));
```

The parameters are described below:

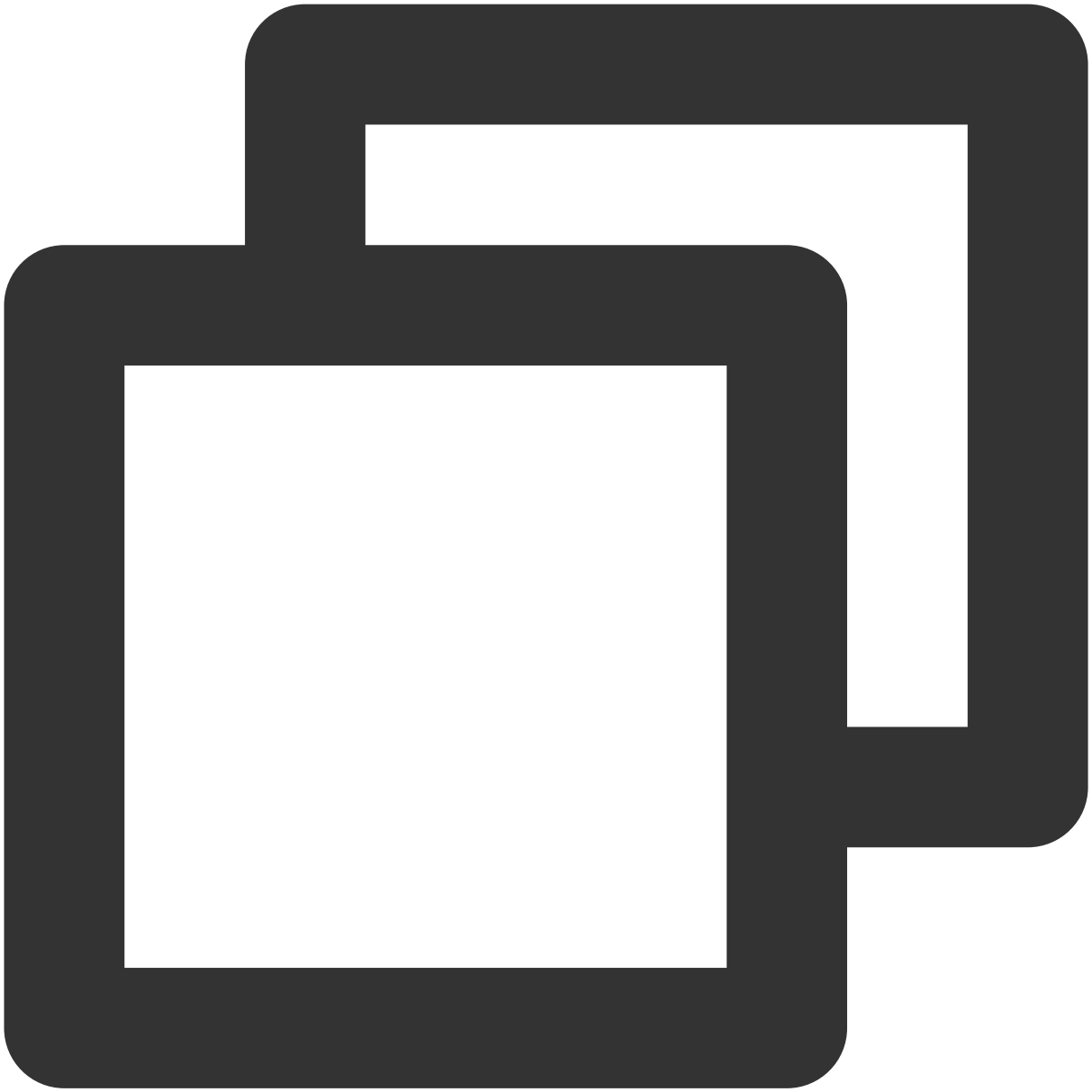
| Parameter | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | The room ID. |

| | | |
|---|---|---|
| callMediaType | TUICallMediaType | The call type, which can be video or audio. |
| callRole | Number | The role, which can be caller or callee. |
| totalTime | double | The call duration: ms |

**Note**

Client-side callbacks are often lost when errors occur, for example, when the process is closed. If you need to measure the duration of a call for billing or other purposes, we recommend you use the RESTful API.

### onCallMediaTypeChanged



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType ne

    }
));
```

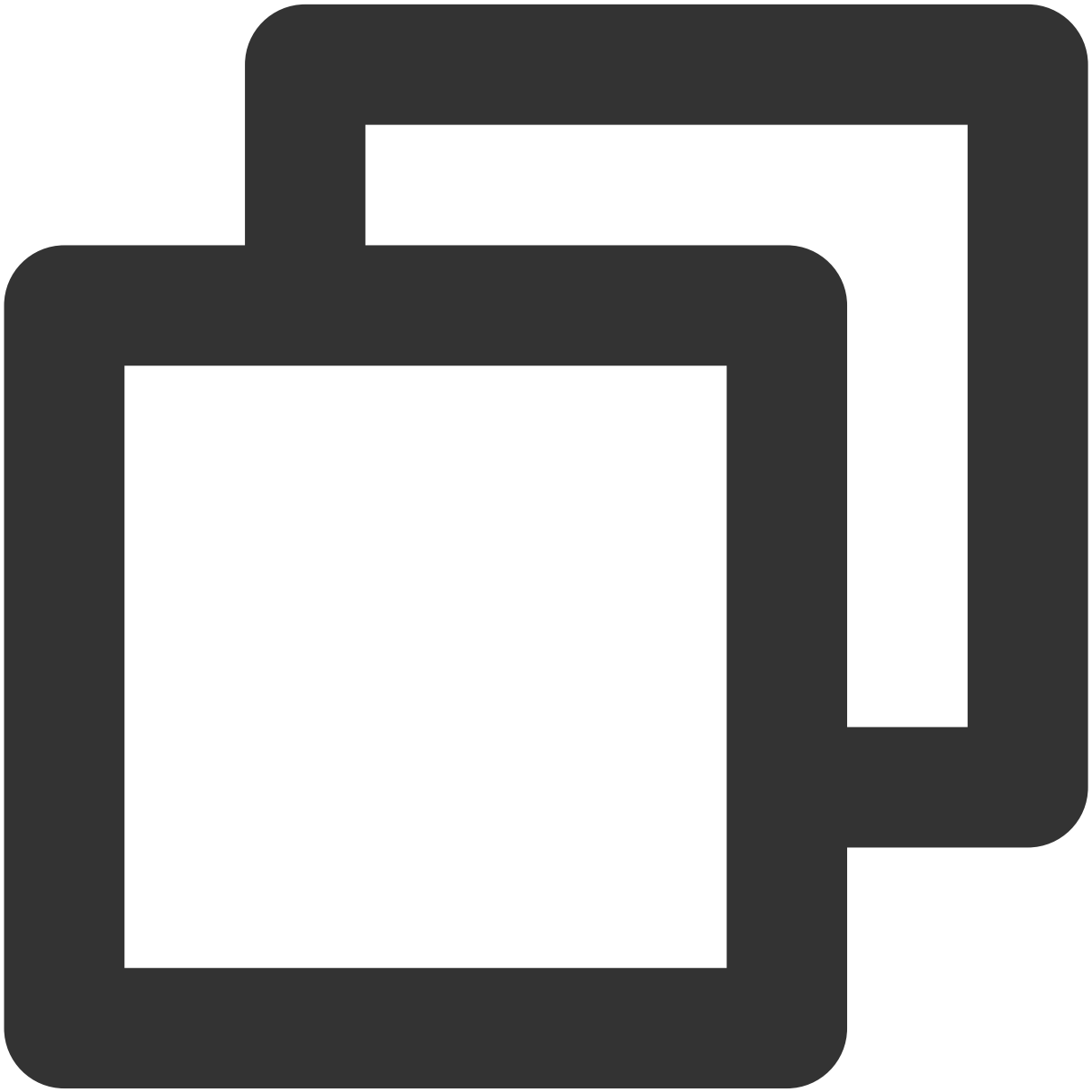
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| oldCallMediaType | TUICallMediaType | The call media type before the change. |

| | | |
|---|---|---|
| newCallMediaType | TUICallMediaType | The call media type after the change. |

### onUserReject

The call was rejected. In a one-to-one call, only the inviter will receive this callback. In a group call, all invitees will receive this callback.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserReject: (String userId) {

    }
```

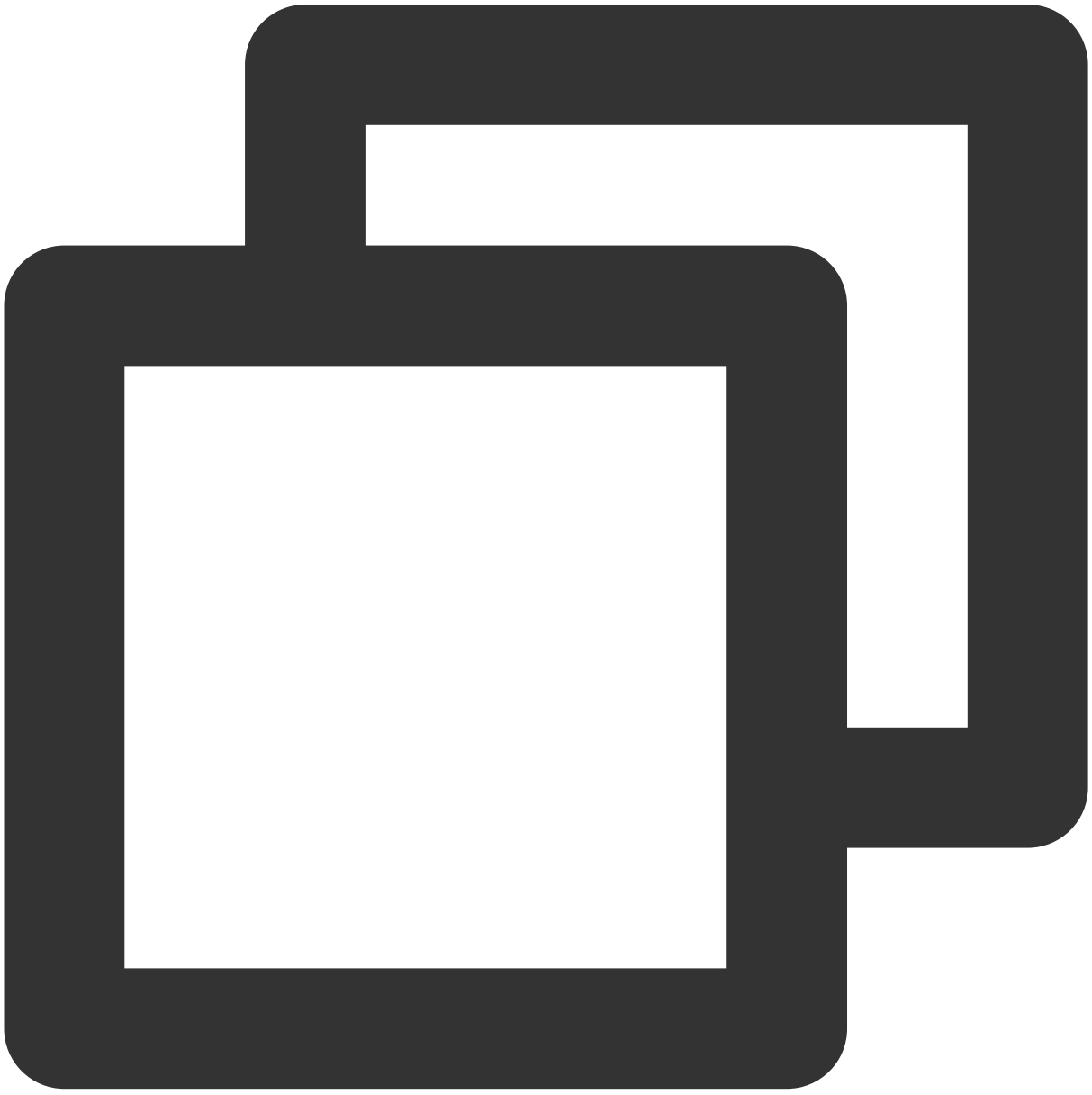
```
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| res.userId | String | The user ID of the invitee who rejected the call. |

### onUserNoResponse

A user did not respond.



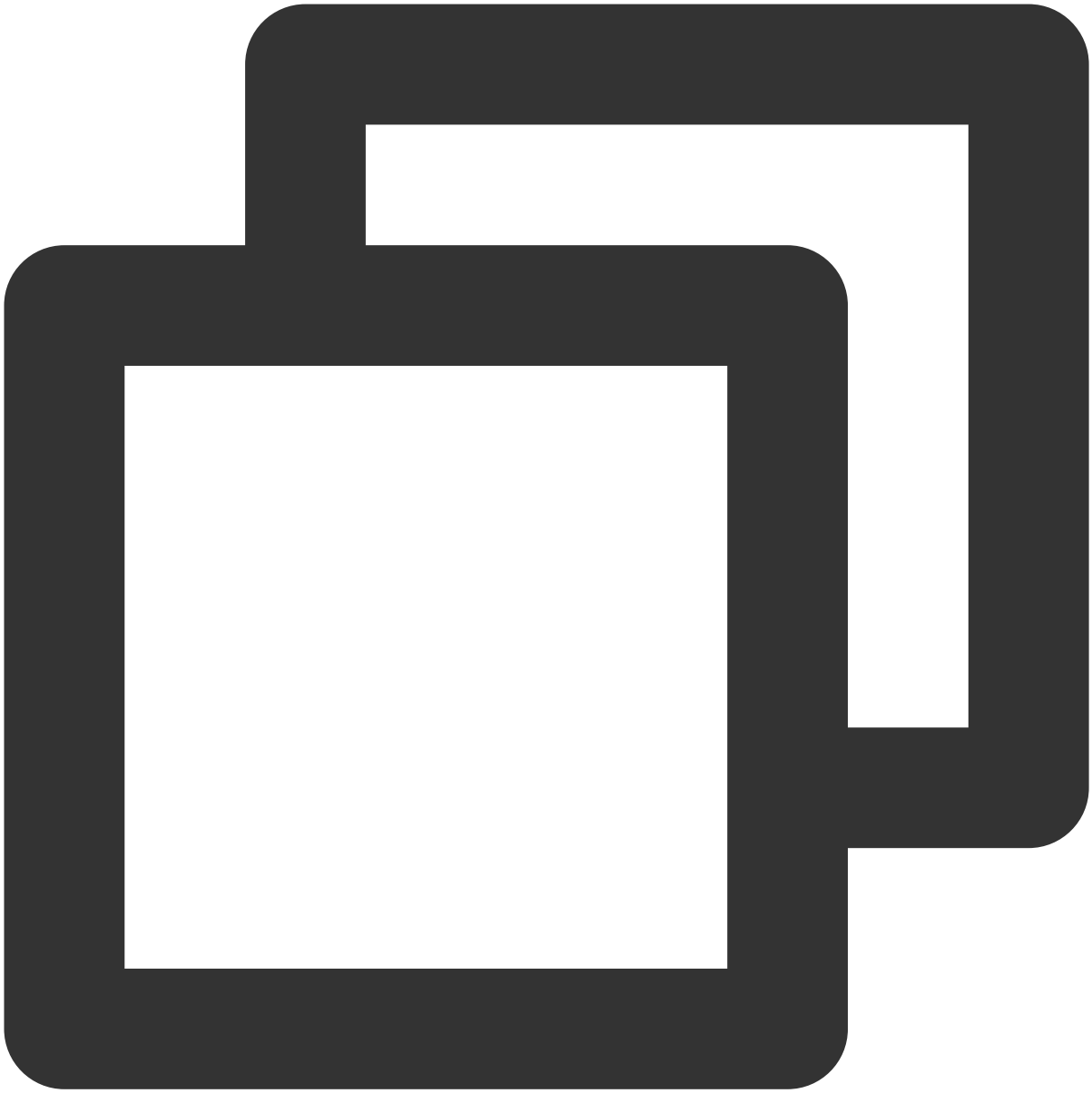
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNoResponse: (String userId) {

    }
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the invitee who did not answer. |

### onUserLineBusy

A user is busy.



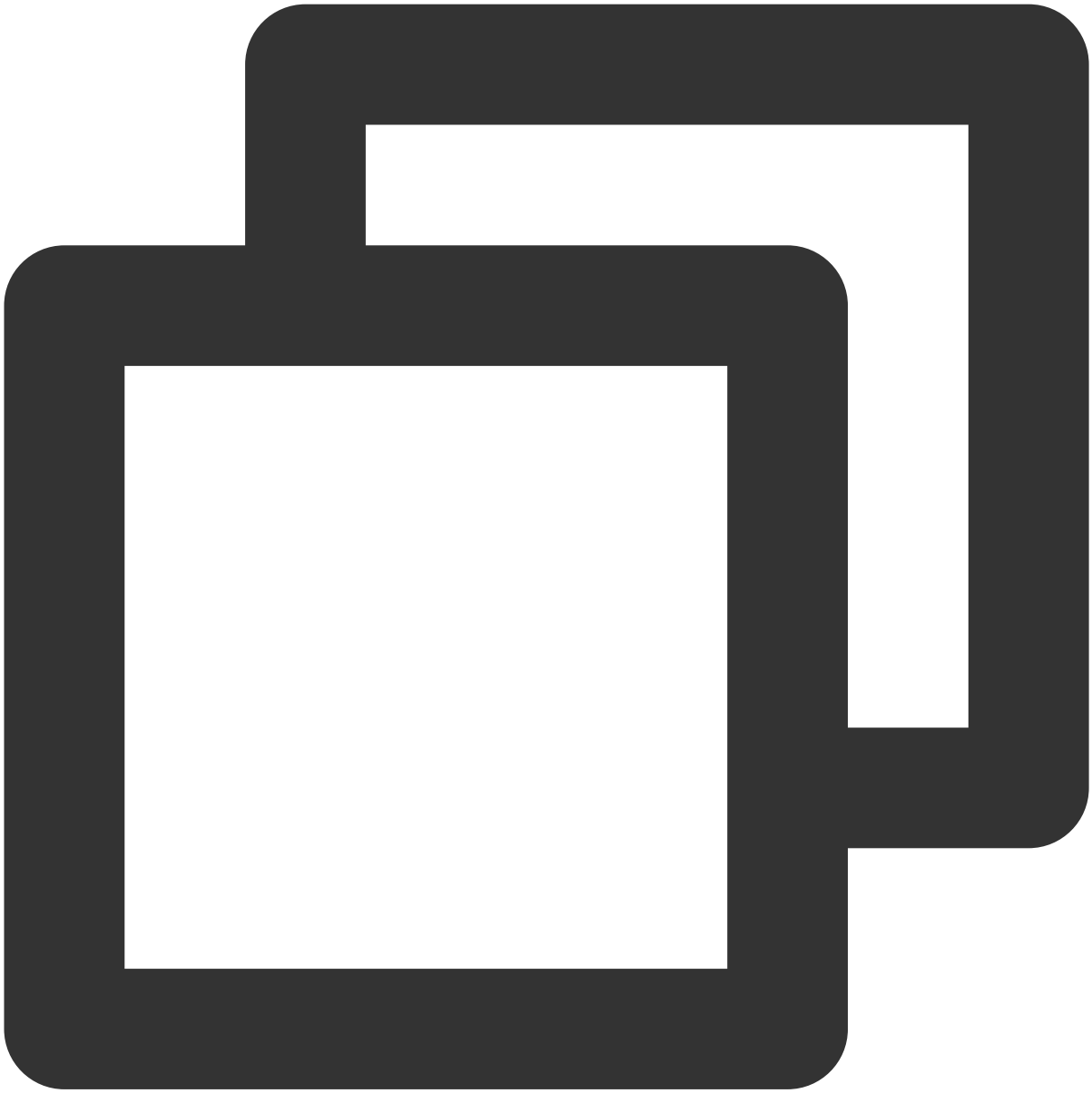
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLineBusy: (String onUserLineBusy) {

    },
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID of the invitee who is busy. |

### onUserJoin

A user joined the call.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserJoin: (String userId) {

    }
));
```

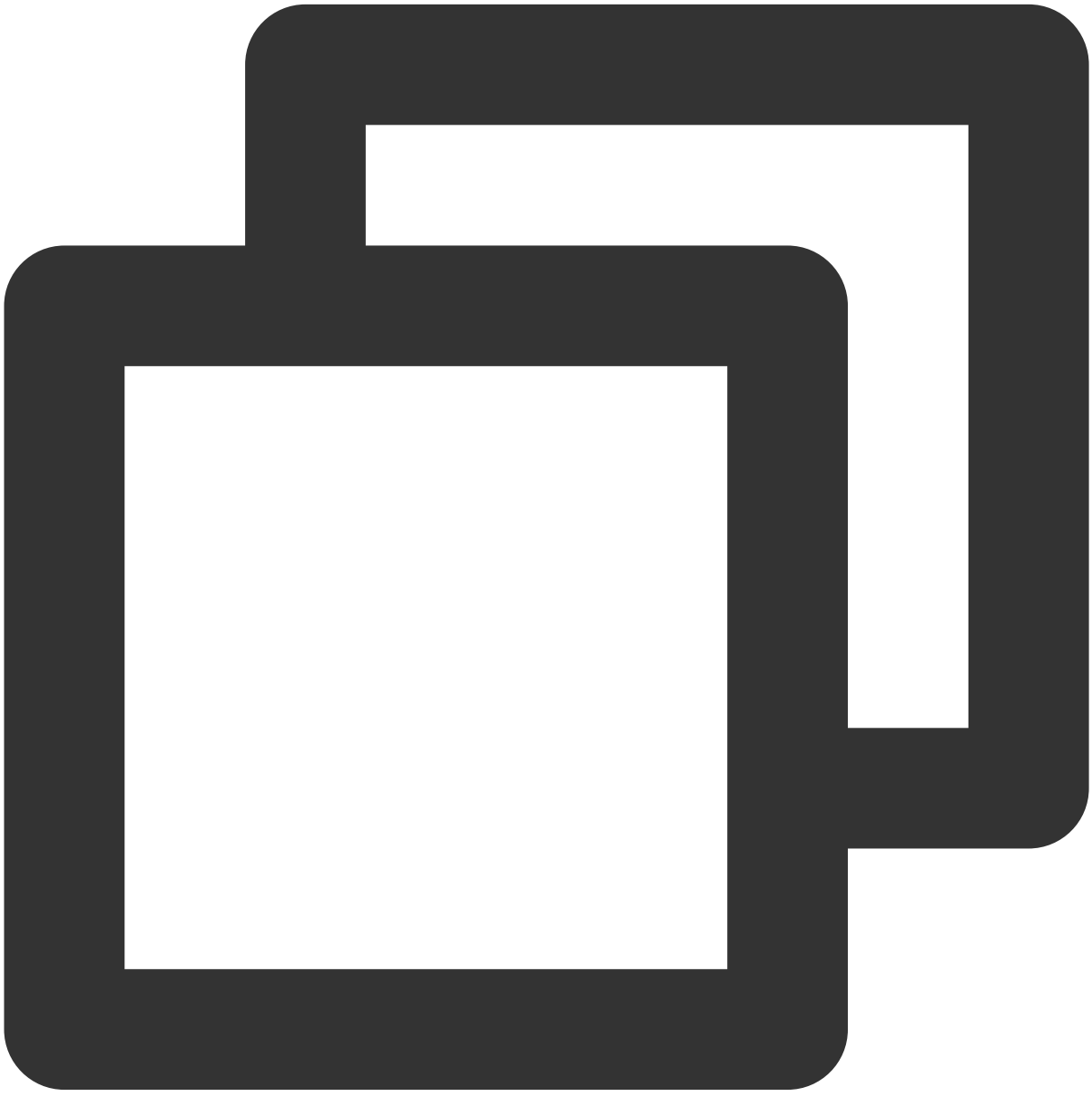
The parameters are described below:

| | | |
|---|---|---|

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the user who joined the call. |

### onUserLeave

A user left the call.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLeave: (String userId) {
    }
```

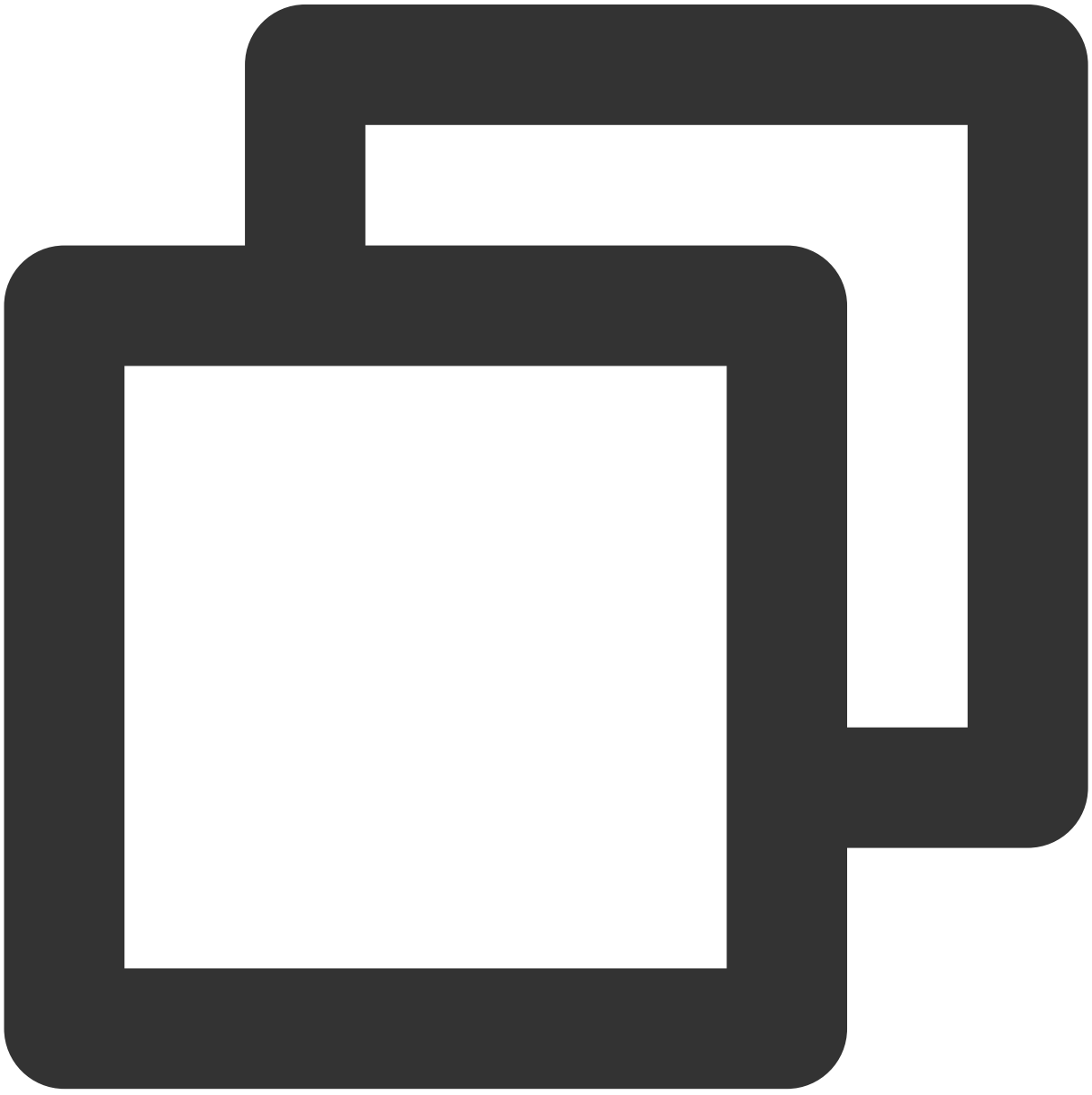
```
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The ID of the user who left the call. |

### onUserVideoAvailable

Whether a user is sending video.



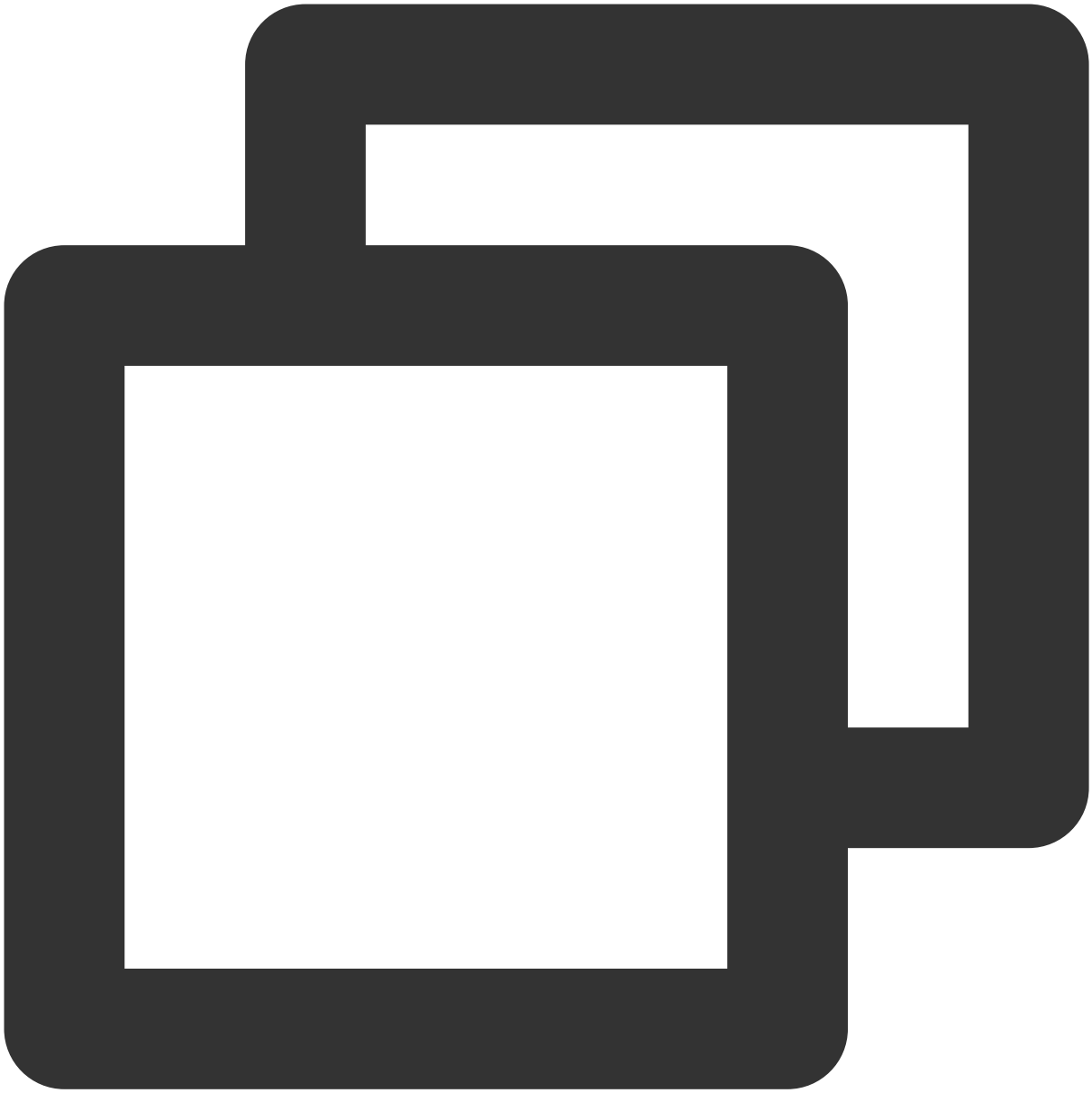
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVideoAvailable: (String userId, bool isVideoAvailable) {
    }
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID. |
| isVideoAvailable | bool | User video available |

### onUserAudioAvailable

Whether a user is sending audio.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserAudioAvailable: (String userId, bool isAudioAvailable) {

    }
));
```

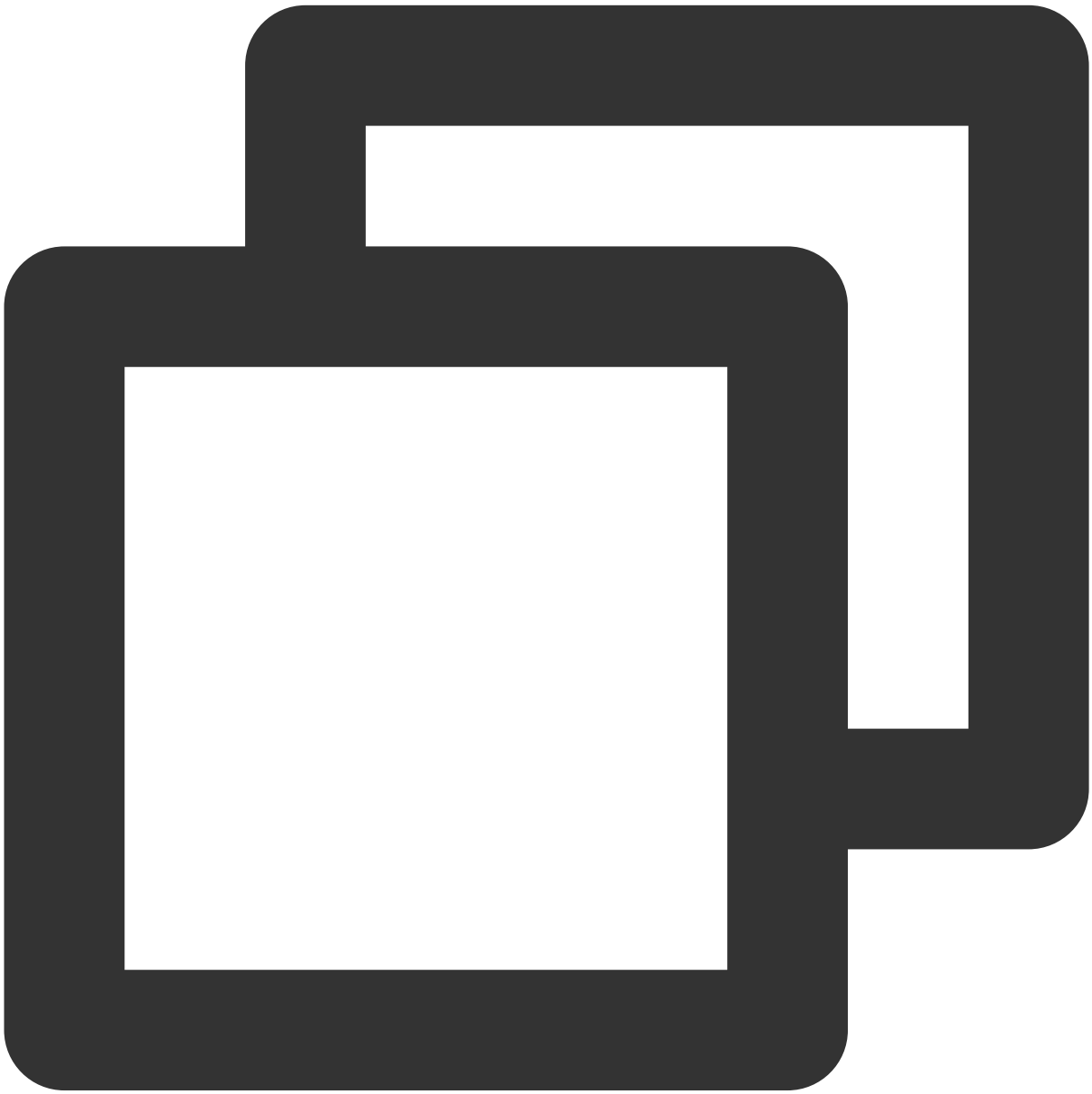
The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| userId | String | The user ID. |

| | | |
|---|---|---|
| isAudioAvailable | bool | Whether the user has audio. |

### onUserVoiceVolumeChanged

The volumes of all users.



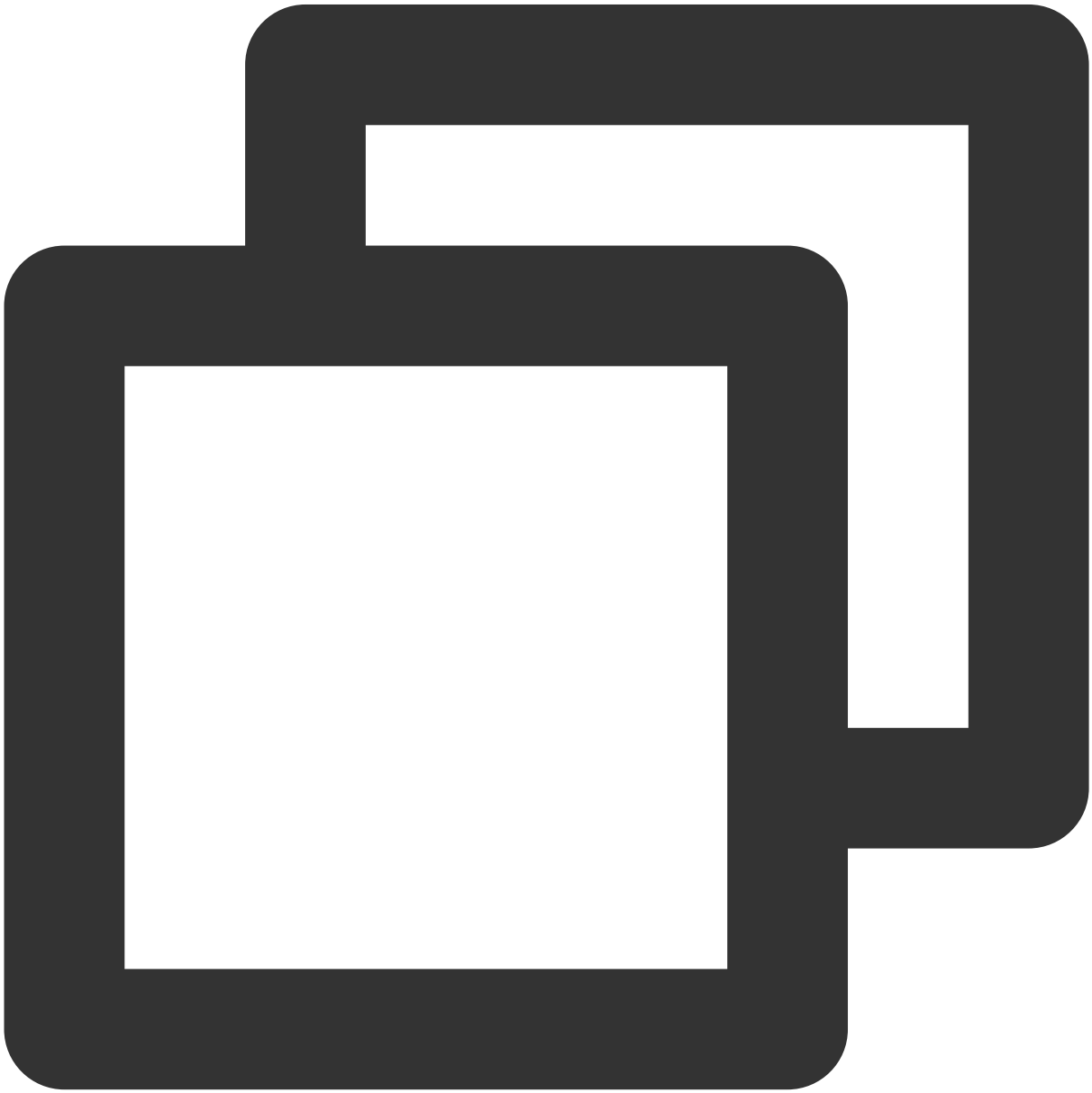
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {

    }
));
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| volumeMap | Map<String, int> | The volume table, which includes the volume of each user (userId). Value range: 0-100. |

### onUserNetworkQualityChanged

The network quality of all users.



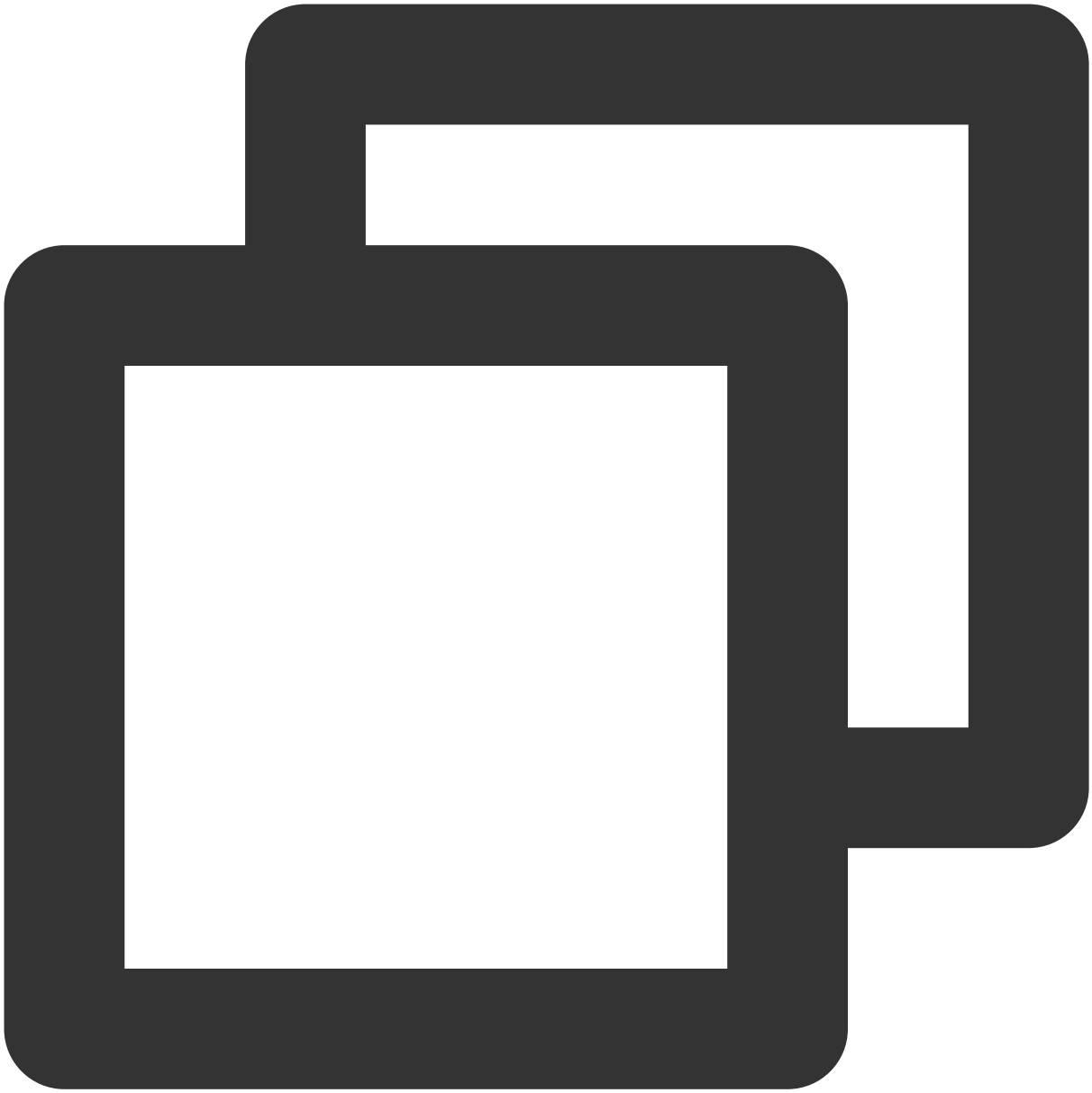
```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {

    }
));

class TUINetworkQualityInfo {
    String userId;
    TUINetworkQuality quality;
    TUINetworkQualityInfo({required this.userId, required this.quality});
}

enum TUINetworkQuality {
  unknown,
  excellent,
  good,
  poor,
  bad,
  vBad,
  down
}
```

The parameters are described below:

| Parameter | Type | Description |
| --- | --- | --- |
| networkQualityList | List<TUINetworkQualityInfo> | he current network conditions for all users (userId). |

### onKickedOffline

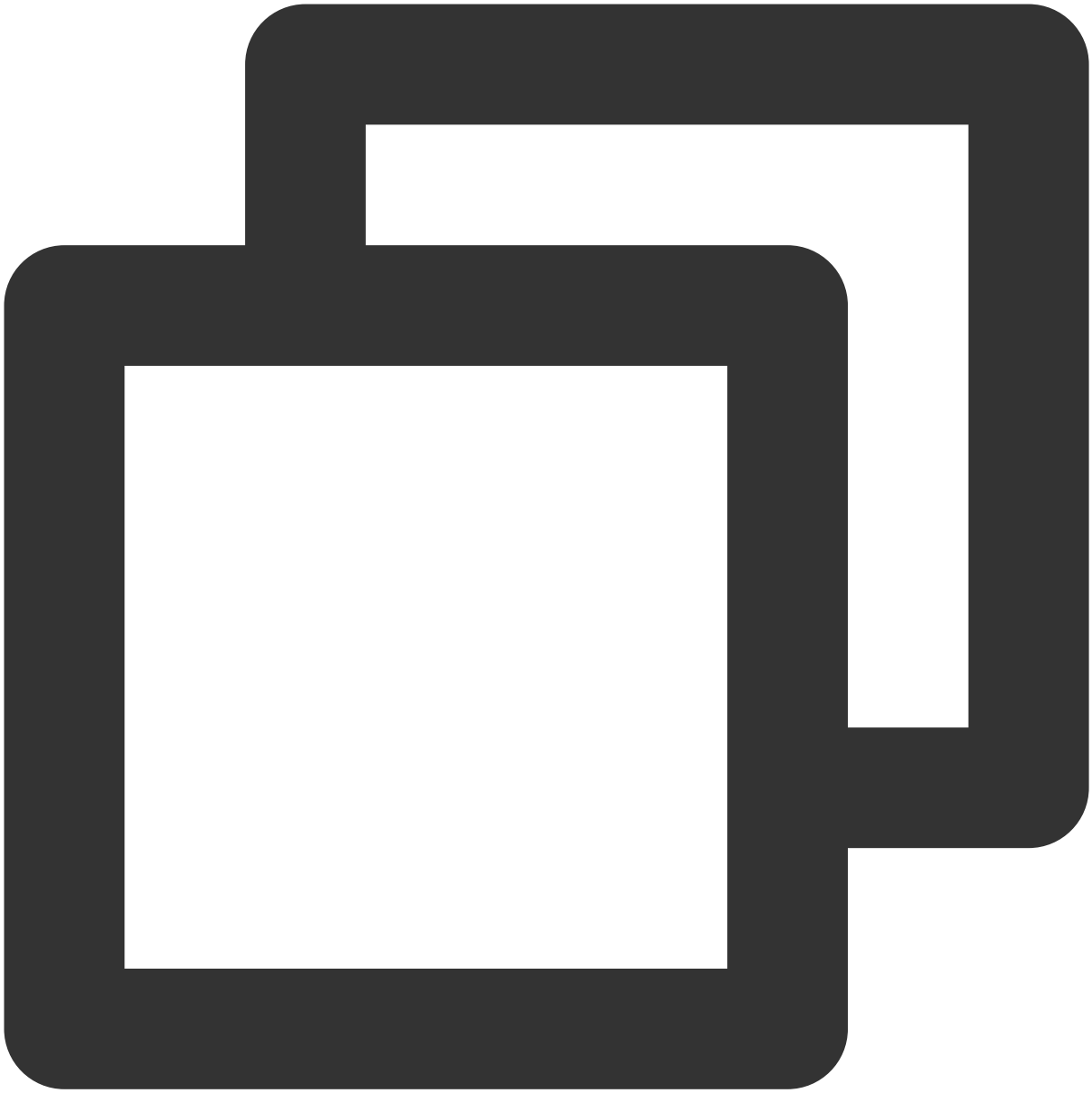
The current user is kicked offline: at this time, the UI can prompt the user and call initialization again.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onKickedOffline: () {

    }
));
```

### onUserSigExpired

Token expired: you need to generate a new userSig and login again.



```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserSigExpired: () {

    }
));
```

# Type Definition

Last updated：2023-08-22 10:13:05

## Common

### TUIResult

| Name | Type | Description |
| --- | --- | --- |
| code | String | If the code is empty "", it means the call succeeded, if the code is not empty "", it means the call failed. |
| message | String | Error message |

### TUIRoomId

Room ID for audio and video in a call.

**Note：**

(1) `intRoomId` and `strRoomId` are mutually exclusive. If you choose to use `strRoomId`, `intRoomId` needs to be filled in as 0. If both are filled in, the SDK will prioritize `intRoomId`.

(2) Do not mix `intRoomId` and `strRoomId` because they are not interchangeable. For example, the number 123 and the string "123" represent two completely different rooms.

| Value | Type | Description |
| --- | --- | --- |
| intRoomId | int | Numeric room ID. |
| strRoomId | String | String room number. |

### VideoRenderParams

| Value | Type | Description |
| --- | --- | --- |
| fillMode | FillMode | Video image fill mode |
| rotation | Rotation | Video image rotation direction |

### VideoEncoderParams

| | | |
| --- | --- | --- |

| Name | Type | Description |
| --- | --- | --- |
| resolution | Resolution | Video resolution |
| resolutionMode | ResolutionMode | Video aspect ratio mode |

### TUICallParams

| Name | Type | Description |
| --- | --- | --- |
| roomId | TUIRoomId | Room Id. |
| offlinePushInfo | TUIOfflinePushInfo | Offline push info configuration information |
| timeout | String | Call timeout period, default: 30s, unit: seconds. |
| userData | String | An additional parameter. |

### TUIOfflinePushInfo

| Name | Type | Description |
| --- | --- | --- |
| title | String | title |
| desc | String | description |
| ignoreIOSBadge | bool | Offline push ignores the badge count (only valid for iOS). If set to true, this message will not increase the unread count of the app icon on the iOS receiving end. |
| iOSSound | String | Offline push sound settings (only valid for iOS). When sound = IOS_OFFLINE_PUSH_NO_SOUND, it means no sound will be played when receiving. When sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND, it means playing system sound when receiving. If you want to customize iOSSound, you need to link the voice file into the Xcode project first, and then set the voice file name (with suffix) to iOSSound |
| androidSound | String | Push sound settings offline (only valid for Android, supported by IMSDK 6.1 and above). |

| | | |
|---|---|---|
| | | FCM push to set the sound prompt on Android 8.0 and above, you must call setAndroidFCMChannelID to set the channelID to take effect |
| androidOPPOChannelID | String | Offline push setting channel ID of OPPO mobile phone 8.0 system and above |
| androidVIVOClassification | int | VIVO push message classification (to be discarded interface, VIVO push service will optimize the message classification rules on April 3, 2023, it is recommended to use setAndroidVIVOCategory to set the message category). 0: operation message 1: system message, the default value is 1 |
| androidXiaoMiChannelID | String | Channel ID of Xiaomi mobile phone system 8.0 and above |
| androidFCMChannelID | String | Channel ID of FCM channel mobile phone system 8.0 and above |
| androidHuaWeiCategory | String | Huawei Push Message Classification |
| isDisablePush | bool | Whether to turn off push (by default, push is enabled) |
| iOSPushType | TUICallIOSOfflinePushType | iOS offline push type, default: APNs |

### TUICallRecords

Call recording information

| Value | Type | Description |
|---|---|---|
| callId | String | Call recording ID. |
| inviter | String | Inviter ID. |
| inviteList | List<String> | List of invited user IDs. |
| scene | TUICallScene | Call scenario. |
| mediaType | TUICallMediaType | Media type. |
| groupId | String | Group ID. |
| role | TUICallRole | Role. |

| | | |
|---|---|---|
| result | TUICallResultType | Call result type. |
| beginTime | int | Start time. |
| totalTime | int | Total time. |

### TUICallRecentCallsFilter

Call recording filtering conditions.

| Value | Type | Description |
|---|---|---|
| begin | double | Start time. |
| end | double | End time. |
| resultType | TUICallResultType | Call result type. |

## Enum definition

### TUICallMediaType

| Name | Value | Description |
|---|---|---|
| none | 0 | None |
| audio | 1 | Audio Calls |
| video | 2 | Video Calls |

### TUICallRole

| Name | Value | Description |
|---|---|---|
| none | 0 | None |
| caller | 1 | Caller |
| called | 2 | Called |

### TUICallStatus

| Name | Value | Description |
|---|---|---|
| none | 0 | None |

| | | |
|---|---|---|
| waiting | 1 | waiting calls |
| accept | 2 | calls answered |

### TUICallScene

| Name | Value | Description |
|---|---|---|
| groupCall | 0 | Group Call |
| multiCall | 1 | Multi calls (not supported yet, stay tuned) |
| singleCall | 2 | one-to-one call |

### TUINetworkQuality

| Name | Value | Description |
|---|---|---|
| unknown | 0 | Unkown |
| excellent | 1 | The current network is very good |
| good | 2 | The current network is good |
| poor | 3 | The current network is poor |
| bad | 4 | The urrent network is bad |
| vBad | 5 | The current network is very poor |
| down | 6 | The current network is unavailable |

### FillMode

| Name | Value | Description |
|---|---|---|
| fill | 0 | Filling mode: the content of the screen will be centered and scaled proportionally to fill the entire display area, and the part beyond the display area will be cropped, and the screen may be incomplete in this mode. |
| fit | 1 | Fit mode: scale according to the long side of the screen to fit the display area, and the short side will be filled with black. In this mode, the image is complete but there may be black borders. |

### Rotation

| Name | Value | Description |
| --- | --- | --- |
| rotation_0 | 0 | Rotate 0 degrees clockwise |
| rotation_90 | 1 | Rotate 90 degrees clockwise |
| rotation_180 | 2 | Rotate 180 degrees clockwise |
| rotation_270 | 3 | Rotate 270 degrees clockwise |

### ResolutionMode

| Name | Value | Description |
| --- | --- | --- |
| landscape | 0 | Landscape resolution，for example : eResolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360。 |
| portrait | 1 | Portrait resolution，for example : Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640。 |

### Resolution

| Name | Value | Description |
| --- | --- | --- |
| resolution_640_360 | 0 | aspect ratio 16:9；resolution 640x360；recommended bitrate（VideoCall）500kbps |
| resolution_640_480 | 1 | aspect ratio 4:3；resolution 640x480；recommended bitrate（VideoCall）600kbps |
| resolution_960_540 | 2 | aspect ratio 16:9；resolution 960x540；recommended bitrate（VideoCall）850kbps |
| resolution_960_720 | 3 | aspect ratio 4:3；resolution 960x720；recommended bitrate（VideoCall）1000kbps |
| resolution_1280_720 | 4 | aspect ratio 16:9；resolution 1280x720；recommended bitrate（VideoCall）1200kbps |
| resolution_1920_1080 | 5 | aspect ratio 16:9；resolution 1920x1080；recommended bitrate（VideoCall）2000kbps |

### TUICallIOSOfflinePushType

| Name | Value | Description |
| --- | --- | --- |
| APNs | 0 | APNS Push |
| VoIP | 1 | VoIP Push |

### TUICamera

Camera type.

| Type | Value | Description |
| --- | --- | --- |
| front | 0 | Front camera. |
| back | 1 | Rear camera. |

### TUIAudioPlaybackDevice

Audio playback device.

| Type | Value | Description |
| --- | --- | --- |
| speakerphone | 0 | Speaker |
| earpiece | 1 | Earpiece |

### TUICallResultType

Call recording type.

| Type | Value | Description |
| --- | --- | --- |
| unknown | 0 | Unknown |
| missed | 1 | Missed |
| incoming | 2 | Incoming call |
| outgoing | 3 | Outgoing Call |

# ErrorCode

Last updated：2023-06-30 16:29:17

Notify users of warnings and errors that occur during audio and video calls.

## TUICallDefine Error Code

| Definition | Value | Description |
| --- | --- | --- |
| ERROR_PACKAGE_NOT_PURCHASED | -1001 | You do not have TUICallKit package, please open the free experience in the console or purchase the official package. |
| ERROR_PACKAGE_NOT_SUPPORTED | -1002 | The package you purchased does not support this ability. You can refer to console to purchase Upgrade package. |
| ERROR_TIM_VERSION_OUTDATED | -1003 | The IM SDK version is too low, please upgrade the IM SDK version to >= 6.6; Find and modify in the build.gradle file.："com.tencent.imsdk:imsdk-plus:7.1.3925" |
| ERROR_PERMISSION_DENIED | -1101 | Failed to obtain permission. The audio/video permission is not authorized. Check if the device permission is enabled. |
| ERROR_GET_DEVICE_LIST_FAIL | -1102 | Failed to get the device list (only supported on web platform). |
| ERROR_INIT_FAIL | -1201 | The init method has not been called for initialization. The TUICallEngine API should be used after initialization. |
| ERROR_PARAM_INVALID | -1202 | param is invalid. |
| ERROR_REQUEST_REFUSED | -1203 | The current status can't use this function. |
| ERROR_REQUEST_REPEATED | -1204 | The current status is waiting/accept, please do not call it repeatedly. |
| ERROR_SCENE_NOT_SUPPORTED | -1205 | The current calling scene does not support this feature. |
| ERROR_SIGNALING_SEND_FAIL | -1406 | Failed to send signaling. You can check the specific |

| | | |
|---|---|---|
| | | error message in the callback of the method. |

## IM Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for communication, such as the core logic of call signaling and busy signaling. Common error codes are as follows:

| Error Code | Description |
|---|---|
| 6014 | You have not logged in to the Chat SDK or have been forcibly logged out. Log in to the Chat SDK first and try again after a successful callback. To check whether you are online, use TIMManager getLoginUser. |
| 6017 | Invalid parameter. Check the error information to locate the invalid parameter. |
| 6206 | UserSig has expired. Get a new valid UserSig and log in again. For more information about how to get a UserSig, see Generating UserSig. |
| 7013 | The current package does not support this API. Please upgrade to the Flagship Edition package. |
| 8010 | The signaling request ID is invalid or has been processed. |

Explanation：
More IM SDK error codes are available at：IM Error Code

## TRTC Error Code

Video and audio calls use Tencent Cloud's IM SDK as the basic service for calling, such as the core logic of switching camera and microphone on or off. Common error codes are as follows:

| Enum | Value | Description |
|---|---|---|
| ERR_CAMERA_START_FAIL | -1301 | Failed to turn the camera on. This may occur when there is a problem with the camera configuration program (driver) on Windows or macOS. Disable and reenable the camera, restart the camera, or update the configuration program. |
| ERR_CAMERA_NOT_AUTHORIZED | -1314 | No permission to access to the camera. This usually occurs on mobile devices and may be because the user denied access. |
| | | |

| | | |
|---|---|---|
| ERR_CAMERA_SET_PARAM_FAIL | -1315 | Incorrect camera parameter settings (unsupported values or others). |
| ERR_CAMERA_OCCUPY | -1316 | The camera is being used. Try another camera. |
| ERR_MIC_START_FAIL | -1302 | Failed to turn the mic on. This may occur when there is a problem with the mic configuration program (driver) on Windows or macOS. Disable and reenable the mic, restart the mic, or update the configuration program. |
| ERR_MIC_NOT_AUTHORIZED | -1317 | No permission to access to the mic. This usually occurs on mobile devices and may be because the user denied access. |
| ERR_MIC_SET_PARAM_FAIL | -1318 | Failed to set mic parameters. |
| ERR_MIC_OCCUPY | -1319 | The mic is being used. The mic cannot be turned on when, for example, the user is having a call on the mobile device. |
| ERR_TRTC_ENTER_ROOM_FAILED | -3301 | Failed to enter the room. For the reason, refer to the error message for -3301. |

Explanation：

More TRTC SDK error codes are available at：TRTC Error Code

# Release Notes (TUICallKit)
# Web

Last updated：2023-07-04 10:16:56

**Description：**

TUICallKit Vue3 Github demo：Github TUICallKit Web。

## Version 2.1.0 Released on April 14, 2023

**New features**

In H5 mode voice calls, the caller's nickname is displayed when calling.

When the call initiation fails, "Call initiation failed" is displayed on the call screen.

When a call is received, "Failed to answer" is displayed on the incoming call screen.

Support listening to whether the current user is kicked out (e.g. being squeezed out of login).

Support listening to TUICallKit call status.

Support business-side code to control the answer, cancel and hangup of calls.

Vue2 version adds TypeScript type declaration file, which can compile type normally in TypeScript project.

**Bug fixing**

Fix the warning of updating profile interface appearing in the console when initializing the component.

Fix the problem that the background image of the called answer button appears misaligned in H5 mode.

**API changes**

`TUICallKitServer.destroy()` Add call restriction, only support call in uncalled state

## Version 2.0.1 Released on March 31, 2023

**New features**

Optimize the rendering logic of 1v1 call and multiplayer call video to improve performance and stability.

Optimize UI display, support displaying corresponding UI during `TUICallKitServer.call()` execution, i.e. <TUICallKit/> UI component can be displayed immediately by clicking the call button.

**Bug fixing**

Fix the problem that nicknames are displayed incorrectly in multiplayer calls.

Fix the problem that CSS does not limit the effective range and pollutes the global style.

## Version 2.0.0 Released on March 21, 2023

**New features**

Support for importing packaged CallKit files from npm.

Support for JavaScript versions of Vue projects.

Support for all versions of Vue projects, npm package for Vue2: call-uikit-vue2.

**Bug fixing**

Fix the problem of abnormal call initiation due to no camera device or authority.

## Version 1.4.2 Released on March 03, 2023

**New features**

Support for setting call resolution.

Support for modifying the screen display mode.

Optimized access steps.

Optimized error throwing.

## Version 1.4.1 Released on February 12, 2023

**New features**

Optimized the preview logic of local camera.

Optimized the rendering logic of remote video streams.

## Version 1.4.0 Released on January 06, 2023

**New features**

Support Vue2.7+ project introduction.

Show nickname by default in call interface.

## Version 1.3.3 Released on December 27, 2022

**New features**

Added empty value detection for call list when making a call in Basic demo.

Added loading icon in Basic demo when making a call.

Optimized the logic of device detection in Basic demo, it no longer pops up actively after manually skipping.

Optimized the reference method of component icons.

Changed the default package management tool to npm.

Optimized the rendering method of video, and reduced the number of repeated rendering.

**Bug fixing**

Fixed Basic Demo error caused by outdated vue-CLI dependency.

## Version 1.3.2 Released on December 07, 2022

**New features**

Support language switching.

Optimize the logic of Basic Demo device detection, it will not pop up again after manually skipping.

**Bug fixing**

Fix a warning caused by introducing defineProps.

## Version 1.3.1 Released on November 29, 2022

**Note**

The new version relies on `tuicall-engine-webrtc@1.2.1` . To use this version, please update `tuicall-engine-webrtc` first.

**New features**

Optimized the style.

Added support for listening for the change of call type by the caller before a call is answered.

Added support for device testing in the basic demo.

**Bug fixing**

Fixed an internal logic error that occurs when the user hangs up.

## Version 1.3.0 Released on November 14, 2022

**Note**

Before you update to this version, please read the update guide.

**New features**

The call view can now automatically adapt to portrait mode on mobile webpages.

Added support for local camera preview when making a call.

Added support for device testing before a call in the basic demo.

**Bug fixing**

Fixed the issue where the TIM instance is not entirely terminated after `TUICallKitServer.destroy()` .

Fixed the issue where the caller receives a no response notification when the callee is busy.

Fixed failure to package TypeScript types in the context of Vite.

**API changes**

If an error occurs after `TUICallKitServer.call()` or `TUICallKitServer.groupCall()` is called, the `beforeCalling` callback is no longer returned. You can use “try…catch” to catch the error.

## Version 1.2.0 Released on November 03, 2022

**New features**

Adapted to the new version of the TUICallEngine SDK.

## Version 1.1.0 Released on October 21, 2022

**New features**

Added support for full screen during a call.

Added support for call window minimizing using `<TUICallKitMini/>` .

**Bug fixing**

Fixed known issues and improved stability.

## Version 1.0.3 Released on October 14, 2022

**New features**

Added support for quick user ID copying and one-click window opening.

## Version 1.0.2 Released on September 30, 2022

**New features**

Added demonstrations and more detailed directions to the integration guide.

**Bug fixing**

Fixed the issue where device status is not shown when the user enters a room for the first time.

Fixed occasional failure to load icons when Webpack is used for packaging.

Fixed several known style issues.

## Version 1.0.1 Released on September 26, 2022

**New features**

Added support for hiding the mic icon of the callee when making a call.

**Bug fixing**

Changed the SDKAppID input restriction in the basic demo to numeric.

## Version 1.0.0 Released on September 23, 2022

TUICallKit basic demo

Integration (TUICallKit)

API Documentation (TUICallKit)

UI Customization (TUICallKit)

FAQs (Web)

# Android & iOS

Last updated：2023-07-04 10:12:27

## Version 1.6.1.410 Released on May 22, 2023

**New features:**

Android & iOS: The UI interface call() and groupCall() now support custom room ID.
Android & iOS: When initiating a call, a string-type room ID can be passed in, see CallParams for details.

**Bug fixes:**

Android: Fixed issue where an error would occur on the groupCall when generating list parameters using Arrays.asList.
Android: Fixed issue where the video call display was abnormal.
iOS: Fixed issue where it conflicted with TUIRoom component.
iOS: Fixed issue where initiating a call immediately after successful login would cause a crash.
iOS: Fixed issue where the invite page would not appear intermittently when clicking on a notification message to enter the app.

## Version 1.6.0.360 Released on April 27, 2023

**New features:**

Android: TUICallKit added the Kotlin language version;
iOS: TUICallKit added the Swift language version;
Android & iOS: Added a page to display local call records.

**Functional optimization:**

Android: Optimized the display of video call avatars.
Android & iOS: In group calls, other group members can be invited to join the call by default.

**Bug fixes:**

Android: Fixed issues where devices running Android 12 or higher would have no sound after being connected to Bluetooth;
Android: Fixed intermittent issues where the muting setting on the callee side was not effective;
iOS: Fixed intermittent issues where devices could not receive incoming call invitations after relogging in;
iOS: Fixed the issue where the enableCustomViewRoute interface of TUICallKit was not valid;
iOS: Fixed the issue where the nickname was displayed incorrectly on the VoIP push page.

## Version 1.5.1.310 Released on April 17, 2023

**New features:**

Android & iOS: Added VoIP message push function to provide a better call answering experience.
Android & iOS: Support custom extended fields when initiating a call, see TUICallDefine.CallParams parameter in the call() method for details.

**Functional optimization:**

Android & iOS: Optimized offline push capabilities for Huawei, Xiaomi, FCM, and other manufacturers, added manufacturer message categories, and channel ID settings.

**Bug fixes:**

Android & iOS: Fixed issue where the IM custom property was overwritten after initiating a call.
Android: Fixed issue where the totalTime unit in the onCallEnd callback was incorrect.

## Version 1.5.0.305 Released on March 09, 2023

**Functional optimization:**

Android & iOS: Optimized chat-message display.
Android: The ear-to-screen messaging function is turned off by default now.
Android: Upgraded gradle plugin and version.
Android: Optimized mediaPlayer class, supporting loop playback of ringtones.

**Bug fixes:**

Android & iOS: Fixed issue where the callee would not receive the onCallCancel callback when answering a call fails.
Android: Fixed issue where the caller would receive an exception when the callee fails to answer the call.
Android: Fixed issue where the caller cancels the call during the permission check of the first call and the callee pulls up the interface again.
Android: Fixed the issue where userId was empty when returning network quality to the upper callback.

## Version 1.4.0.255 Released on January 06, 2023

**New features:**

Android & iOS: Support custom call timeout time, see TUICallDefine.CallParams parameter in the call() method for details.

**Bug fixes:**

Android & iOS: Fixed issue where joining a room actively (joinInGroupCall) would result in abnormal termination of the call.
Android: Fixed issue where there were abnormalities with call status when you exited the audio and video call answer interface and came back to the foreground again.

### Version 1.3.0.205 Released on November 30, 2022

**New features:**

Android & iOS: Added beauty setting interface setBeautyLevel(), supporting turning off default beauty.

**Functional optimization:**

iOS: Optimized the framework size of TUICallKit.

### Bug fixes:

Android & iOS: Fixed issue where the calling interface did not disappear when the server dissolves a room or kicks out a user.
Android: Fixed issue where if A called offline user B and then cancelled, then A called B again and B came online, the calling interface did not appear.

## Version 1.2.0.153 Released on November 14, 2022

### New features:

Android & iOS: Support for custom video encoding resolutions.
Android & iOS: Support setting rendering parameters for video: rendering direction and filling mode.
Android & iOS: Support integration of third-party beauty features.
Android & iOS: TUICallKit has added overloaded interfaces call() and groupCall(), supporting custom offline messages (see API documentation for details).

### Functional optimization:

Android & iOS: Optimized some TUICallObserver callback exception issues.
Android & iOS: Optimized the video-to-audio switching function, supporting switching in offline state.
Android & iOS: Improved error codes and error prompts for TUICallKit.

iOS: Standardized TUICallEngine and TUICallKit Swift API names.

### Bug fixes:

Android & iOS: Fixed issue where you would still receive previously rejected incoming calls after logging back in to your account.
Android: Fixed issue where you would encounter abnormal hang-ups in invites from group chats in one-to-one chats.
Android: Fixed issue where there were abnormalities with multiple-scene exits from live rooms, preventing the initiation of calls.
Android: Fixed issue where contacting person A while B and C were calling each other at the same time would cause C to enter A's room at random.

## Version 1.1.0.103 Released on September 30, 2022

Android & iOS: Optimized the feature of inviting new members to the current group call.
Android & iOS: Optimized the call process to avoid the charging of recording, moderation, and other fees before a call is answered.
Android & iOS: Added support for custom offline notifications.
Android & iOS: Changed the parameters of some **TUICallEngine** APIs. For details, see call(), groupCall(), inviteUser(), and onCallReceived().
Android & iOS: Fixed occasional callback errors during a group call.
Android & iOS: Fixed the status abnormal issue caused by repeated login or expired `UserSig`.
iOS: Fixed the issue where, when a mixed-language `TUICallKit` project is built with Objective-C and Swift, an error occurs when `init` is called.
Android: Fixed the issue where an error occurs when the floating window feature is integrated for a Kotlin project.

## Version 1.0.0.53 Released on August 15, 2022

### First release:

Android & iOS: Supports one-to-one and group audio/video calls.
Android & iOS: Supports offline call push for mainstream devices on the market.

Android & iOS: Supports custom profile photos and aliases.

Android & iOS: Supports floating call windows.

Android & iOS: Supports custom ringtones.

Android & iOS: Supports receiving calls when the user is logged in on multiple platforms.

# Flutter

Last updated：2024-03-20 16:32:31

## Version 2.2.2 @2024.03.07

**New Features:**

Android & iOS: Added the user-defined parameter userData for TUICallObserver's onCallReceived callback.

## Version 2.2.1 @2024.02.06

**Bug Fixes:**

Android: Fixed the issue of abnormal app page caused by the app receiving the FCM push after the app process is killed.

## Version 2.2.0 @2024.02.03

**New Features:**

Android: Provided support for FCM push (tencent_cloud_chat_push is needed).

## Version 2.1.1 @2024.01.06

**Bug Fixes:**

Android & iOS: Fixed some UI related bugs in UI 3.0.

## Version 2.1.0 @2024.01.04

**New Features:**

Android & iOS: Upgraded to UI3.0.

## Version 2.0.6 @2023.12.15

### Bug Fixes:

iOS: Fixed the issue of missing fields in iOS offline push information.

## Version 2.0.5 @2023.12.10

### Bug Fixes:

iOS: Fixed the issue of abnormal ringtone when the user accesses the call page during VoIP push and abnormal pulling of remote stream.
Android: Fixed the issue of abnormal incoming call page display when this app is running in the background without the floating window permission.

## Version 2.0.4 @2023.12.04

### Bug Fixes:

Android & iOS: Fixed the intermittent issue of incoming call page not displayed after the user clicks on an incoming call notification.

## Version 2.0.2 @2023.11.27

### Bug Fixes

Android : Fixed the issue of call failure of Method Channel under multiple Flutter Engine conditions.

## Version 2.0.1 @2023.11.15

### Bug Fixes

Android & iOS : Fixed an incompatibility issue of Tencent RTC Observer caused by the use of other Tencent RTC components.
Android : Fixed an incompatibility issue caused by the use of the combination of TUICallKit and TUIRoom.

## Version 2.0.0 @2023.11.06

### Dependency Description

Upgrade the dependency native SDK version: Android&iOS TUICore:7.6.5011, Android&iOS TUICallEngine:2.0.0.750.

# Version 1.9.1 @2023.10.21

### New Features：

iOS: Supports Voip.

### Bug Fixes

iOS: Fixed the abnormality of the call page when a call is received in the background.

# Version 1.9.0 @2023.10.09

### New Features

Android$iOS: Added an interface for setting ringtones.

### Function Optimization:

Android & iOS: Optimized the purchase package prompts.
Android & iOS: Optimized default bitrates for different resolutions, [see details](https://trtc.io/document/46660/51002/54904/54909).

### Bug Fixes

iOS: Fixed that the same Observer object can be registered twice.

### Dependency Description

1. Upgrade the dependency native SDK version: Android&iOS TUICore:7.5.4852, Android&iOS TUICallEngine:1.9.0.680.

# Version 1.8.3 @2023.08.25

### Bug Fixes

Android&iOS: Fixed the issue of no call message display when tencent_cloud_chat_uikit is used.
Android&iOS: Fixed the issue of occasionally pulling up the group call page during a single-person call.
Android&iOS: Fixed the issue of occasionally pulling up the call page twice during a call.
Android&iOS: Fixed the issue of abnormal display of call duration.

## Function Optimization:

Android: Optimized the problem of failing to pull up the interface in the background when receiving a call.

**Dependency Description：**

Upgrade tencent_cloud_uikit_core to version 1.1.1.

# Version 1.8.2 @2023.08.19

# Bug Fixes

iOS: Fixed the compilation failure of some compilers due to the use of deprecated Swift interfaces.

# Version 1.8.1 @2023.08.18

**Bug Fixes：**

Android&iOS: Fixed the problem of the video stream display during a group voice call.

# Version 1.8.0 @ 2023.08.17

**New Features:**

Android&iOS：Build a new TUICallkit based on the Dart language, which makes it easier to customize your own UI style.
Android&iOS：TUICallEngine adds multiple business interfaces such as hangup, accept, and reject.

# Version 1.7.4 @ 2023.07.20

**Function Optimization:**

Android：By default, the gravity sensor is turned off to optimize the call experience on large screens and customized devices.

# Bug Fixes:

Android&iOS：A calls B (offline) and then cancels, A calls B again, B logs in and goes online, and B's cloud call record is abnormal.

## Version 1.7.3 @ 2023.07.19

### Function Optimization:

Android: Supports development & debugging using the emulator.

### Dependency Notes:

The client SDK version that the upgrade depends on: Android LiteAVSDK_Professional: 11.3.0.13176.

## Version 1.7.2 @ 2023.07.09

### Bug Fixes:

iOS: Upgrade the client SDK version to fix the problem of AppStore listing failure caused by Non-public API usage.

## Version 1.7.1 @ 2023.07.03

### New Features:

Android&iOS：Added cloud call records, you can activate the service on the console for experience query.

### Function Optimization:

Android：Reduce the level of system keep-alive during calls, only display the keep-alive reminder in the status bar, and remove notifications and vibrations.

## Version 1.6.3 @ 2023.06.03

### Bug Fixes:

iOS: Fix the issue of an empty page when adding people halfway after calling joinInGroupCall.
iOS: Fix the issue of user screen overlap after calling joinInGroupCall.

## Version 1.6.2 @ 2023.05.30

### Bug Fixes:

Android: Fix the crash issue caused by calling the joinInGroupCall API.

## Version 1.6.1 @ 2023.05.15

### Bug Fixes:

Android: Fix the occasional crash when applying for floating window permissions on specific Vivo models.

### Dependency Notes:

Upgrade the dependency native SDK version: Android LiteAVSDK_Professional: 11.1.0.13111, iOS TXLiteAVSDK_Professional: 11.1.14143.

## Version 1.6.0 @ 2023.05.03

### New Features:

Android&iOS: Add hangup interface.
Android&iOS: Add user-defined fields and user-defined call timeout duration.
Android&iOS: Add midway join page for group calls.

### Function Optimization:

Android: Optimize single-user video call avatar display.
Android&iOS: By default, support inviting other group members to join the call in group calls.

### Bug Fixes:

Android: Fix the issue of no sound on Android 12 and above devices after connecting to Bluetooth.
Android: Fix the occasional issue of mute settings not taking effect on the called party's side.
iOS: Fix the occasional issue of the device not receiving incoming call invitations after re-login.
iOS: Fix the issue of incorrect nickname display on VoIP push page.

### Dependency Notes:

Upgrade the dependency native SDK version: Android LiteAVSDK_Professional: 11.1.0.13111, iOS TXLiteAVSDK_Professional: 11.1.14143.

## Flutter Version 1.5.4 @ 2023.04.14

## New Features:

Android&iOS: Add offline push parameters for Xiaomi, Huawei, and VIVO.
iOS: Supports VoIP message push function.
Android&iOS: Add resolution setting function.

### Function Optimization:

Android: Optimize the unit of the totaltime parameter in the onCallEnd callback to milliseconds.

### Bug Fixes:

Android&iOS: Fix onCallReceived callback exception issue.
iOS: Fix the issue of incomplete call page display when the screen is rotated.

## Version 1.5.3 @ 2023.03.17

### Bug Fixes:

Android: Fix the packaging failure issue.
Android&iOS: Fix the issue of throwing exceptions when callback methods are not implemented.

## Version 1.5.2 @ 2023.03.13

### Bug Fixes:

Android: Fix the compilation error caused by the API change of TUICallDefine.OfflinePushInfo.

## Version 1.5.1 @ 2023.03.13

### Bug Fixes:

Android&iOS: Fix the issue of incorrect version dependency of tencent_calls_engine.

## Version 1.5.0 @ 2023.03.13

### Function Optimization:

Android: Proximity wake-up function is turned off by default.
Android: Upgrade gradle plugin and version.

Android: Optimize the ringtone playback class, supporting loop playback.

### Bug Fixes:

Android&iOS: Fix the issue that the onCallCancel callback is missing when the called party fails to answer the call.
Android: Fix the abnormal problem of the caller when the called party fails to answer the call.
Android: Fix the issue that the interface is pulled up again by the called party when checking permissions for the first call and the calling party cancels the call.
Android: Fix the issue of userId being empty when calling back the network quality to the upper layer.
iOS: Fix the issue of incorrect Observer registration timing in Example.

## Version 1.4.2 @ 2023.02.24

### Bug Fixes:

Android&iOS: Fix the issue of abnormal calls caused by the wrong Observer registration timing in Example.
iOS: Fix the occasional invalid setting issue of removeObserver API.

## Version 1.4.1 @ 2023.02.20

### Bug Fixes:

Android: Fix the issue of the OnCallEnd event being lost after the call ends.

## Version 1.4.0 @ 2023.01.16

### Bug Fixes:

Android&iOS: Fix the issue of abnormal call termination when actively joining a room (joinInGroupCall).
Android: Fix the issue of abnormal call status when entering the background during audio and video call answering and returning to the foreground.
Android: Fix the issue of call initiation failure caused by login status when integrating the tencent_cloud_chat_uikit plugin.
Android: Fix the issue of call initiation failure due to parameter check issues when initiating a group call.

## Version 1.3.1 @ 2022.12.27

### New Features:

Android&iOS: Support custom offline push message during a call.

### Bug Fixes:

Android: Fix the issue of sdkappid is invalid prompt when integrating the tencent_cloud_chat_uikit plugin.

## Version 1.3.0 @ 2022.12.02

### Function Optimization:

iOS: Optimize the size of the TUICallKit Framework.

### Bug Fixes:

Android&iOS: Fix the issue of the call interface not disappearing when the server-side dissolves the room or kicks out users.
Android: Fix the issue that the call interface does not display when user A calls offline user B, then cancels the call; A calls B again, and B's call interface does not display after going online.

## Version 1.2.2 @ 2022.11.17

### Bug Fixes:

iOS: Fix the compilation issue caused by static library linking.

## Version 1.2.0 @ 2022.11.17

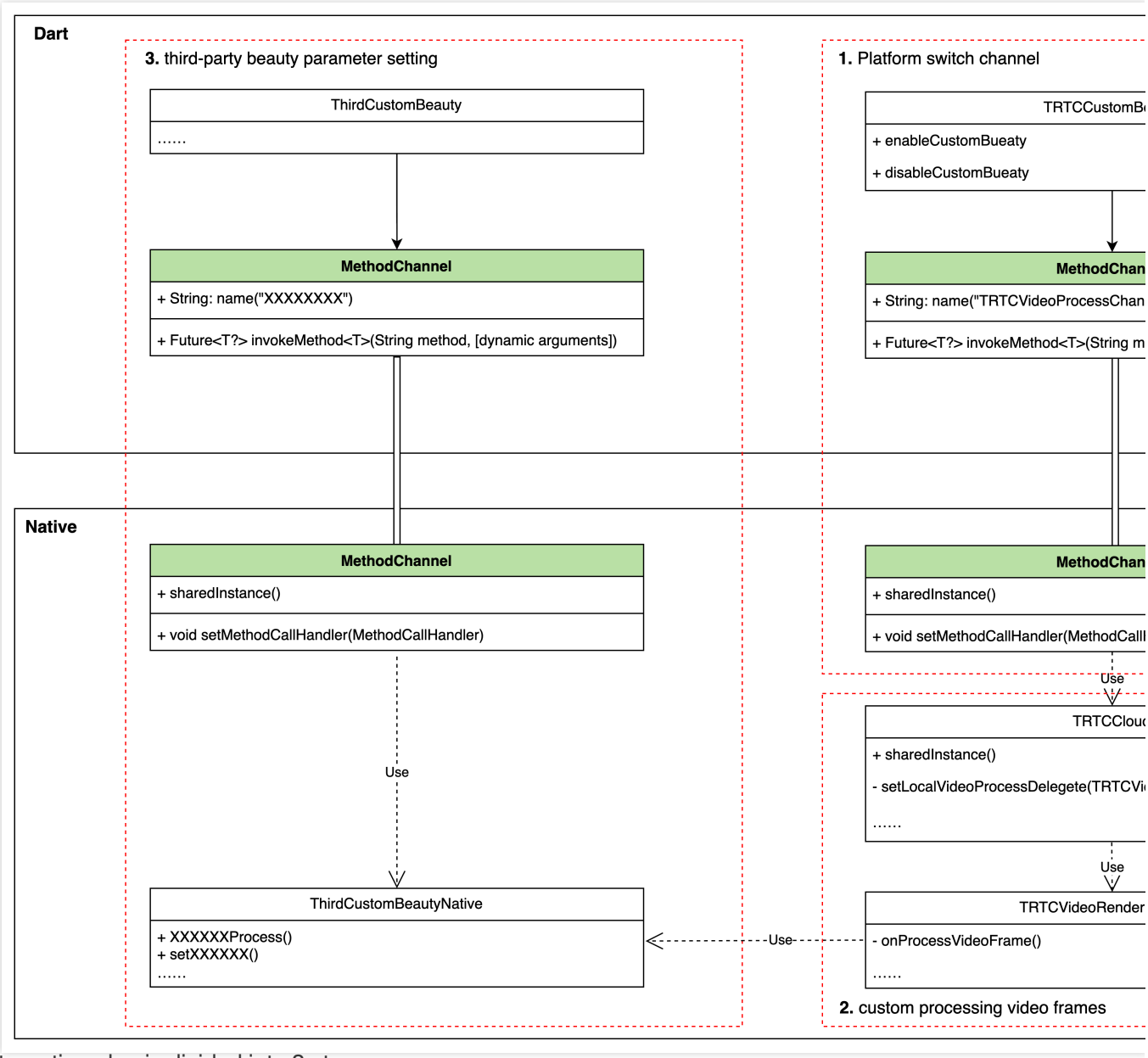
### New Features:

Support 1v1 audio and video calls, group audio and video calls.
Support custom avatar and custom nickname.
Support setting custom ringtones.
Support enabling floating window during the call.
Support incoming call service for multi-platform login status.

# Beauty Effects (TUICallKit)
# Flutter

Last updated：2024-03-20 16:26:08

This document mainly introduces how to integrate beauty effects in TUICallKit.
To perform custom beauty processing in Flutter, it is necessary to use TRTC's custom video rendering. Due to Flutter's lack of proficiency in handling real-time transmission of large data volumes, work involving TRTC's custom video rendering needs to be completed in the Native section. The specific plan is as follows:



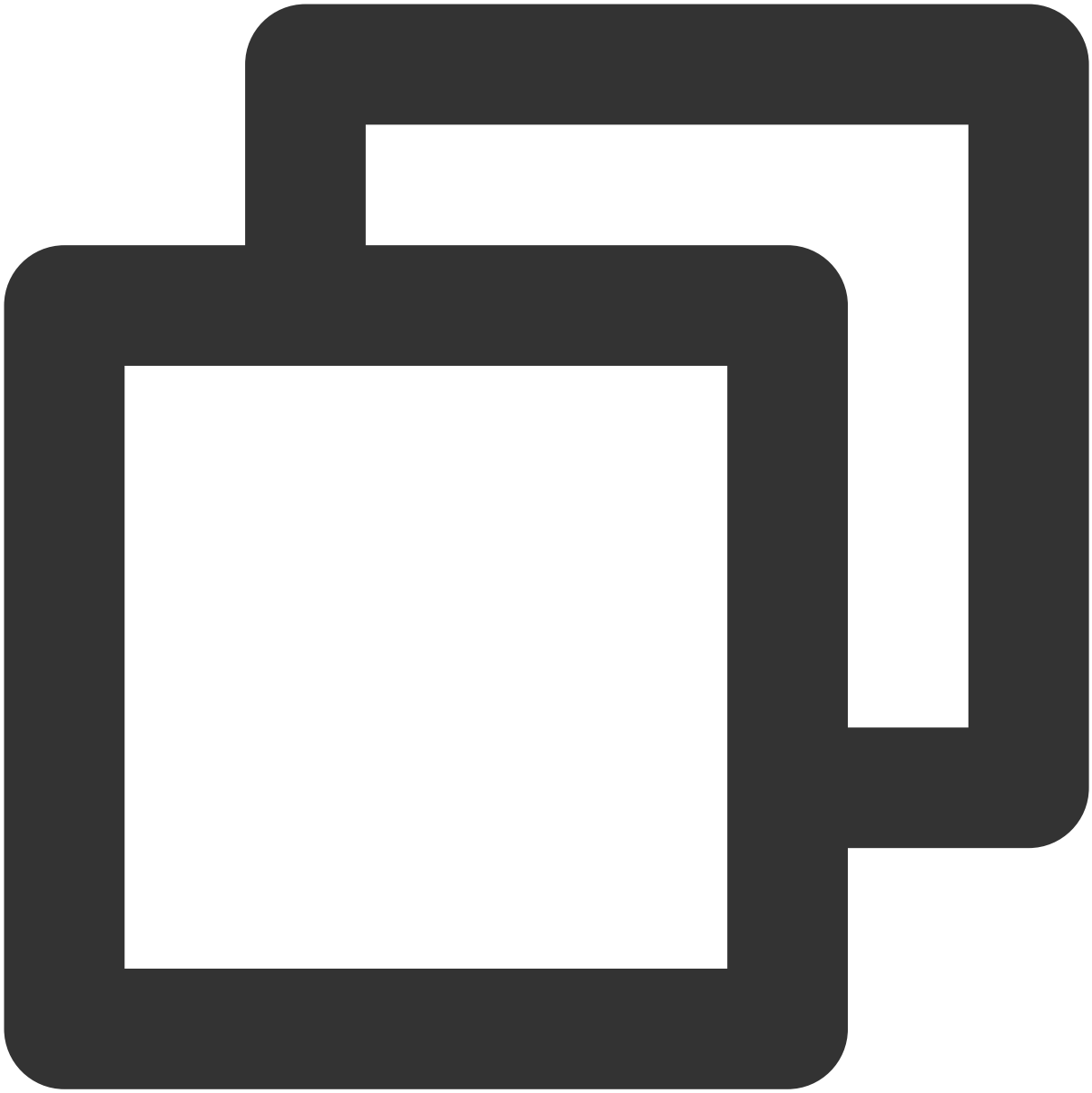

The integration plan is divided into 3 steps:

Step 1: Enable/Disable the TRTC custom rendering logic through MethodChannel;
Step 2: In the TRTC custom rendering processing logic onProcessVideoFrame(), use the **beauty processing module** to process the original video frames;
Step 3: On the user's beauty processing module, the beauty parameters also need to be set through the Dart interface. Users can set beauty parameters through MethodChannel based on their needs and the beauty effects they use.

# Integrating Third-Party Beauty Effects

### Step 1: Implement the control interface of enabling/disabling beauty effects from Dart to Native.

Implement the interface in Dart:

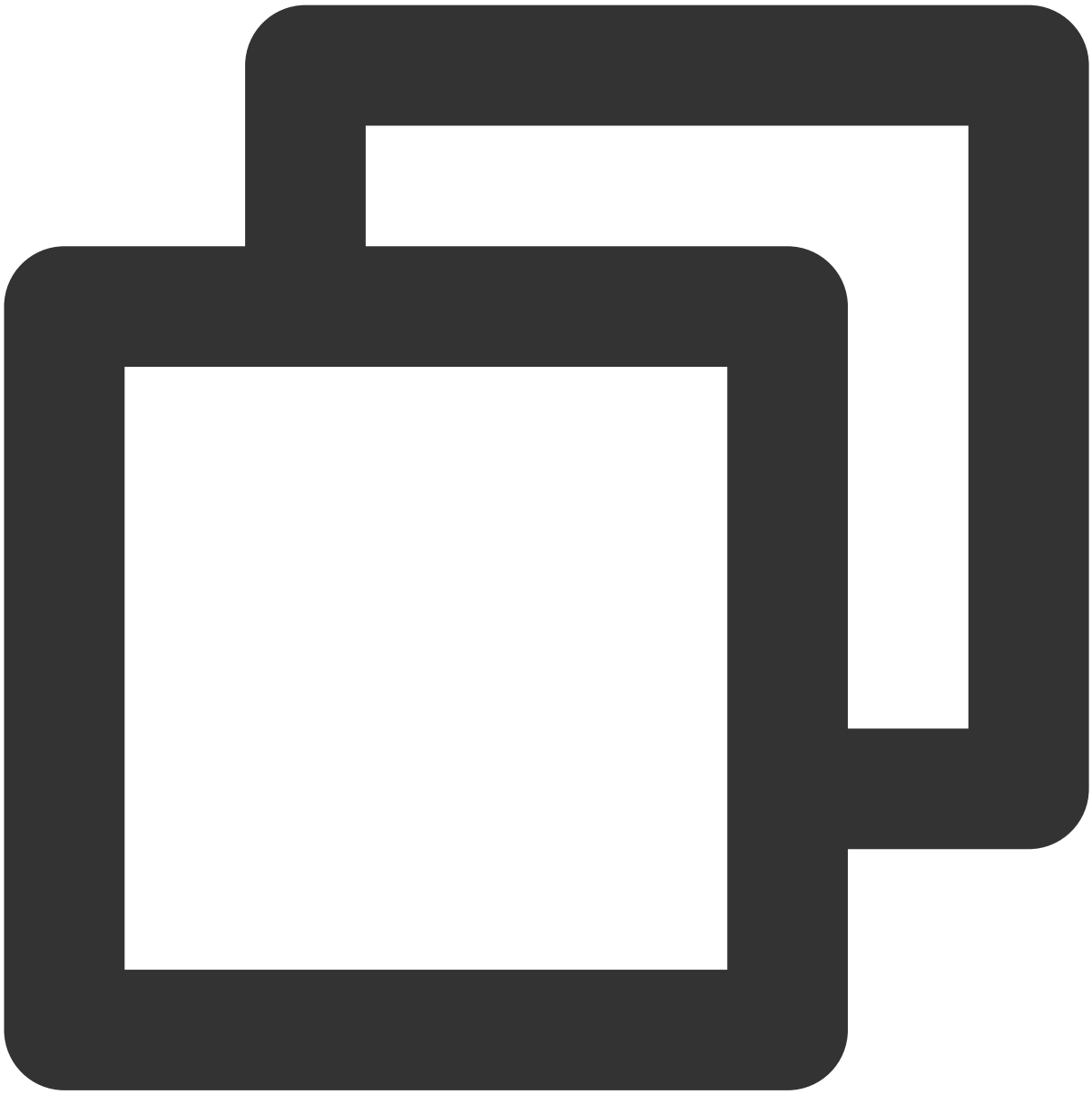


```
final channel = MethodChannel('TUICallKitCustomBeauty');

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

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

Implement the corresponding interface in Native:

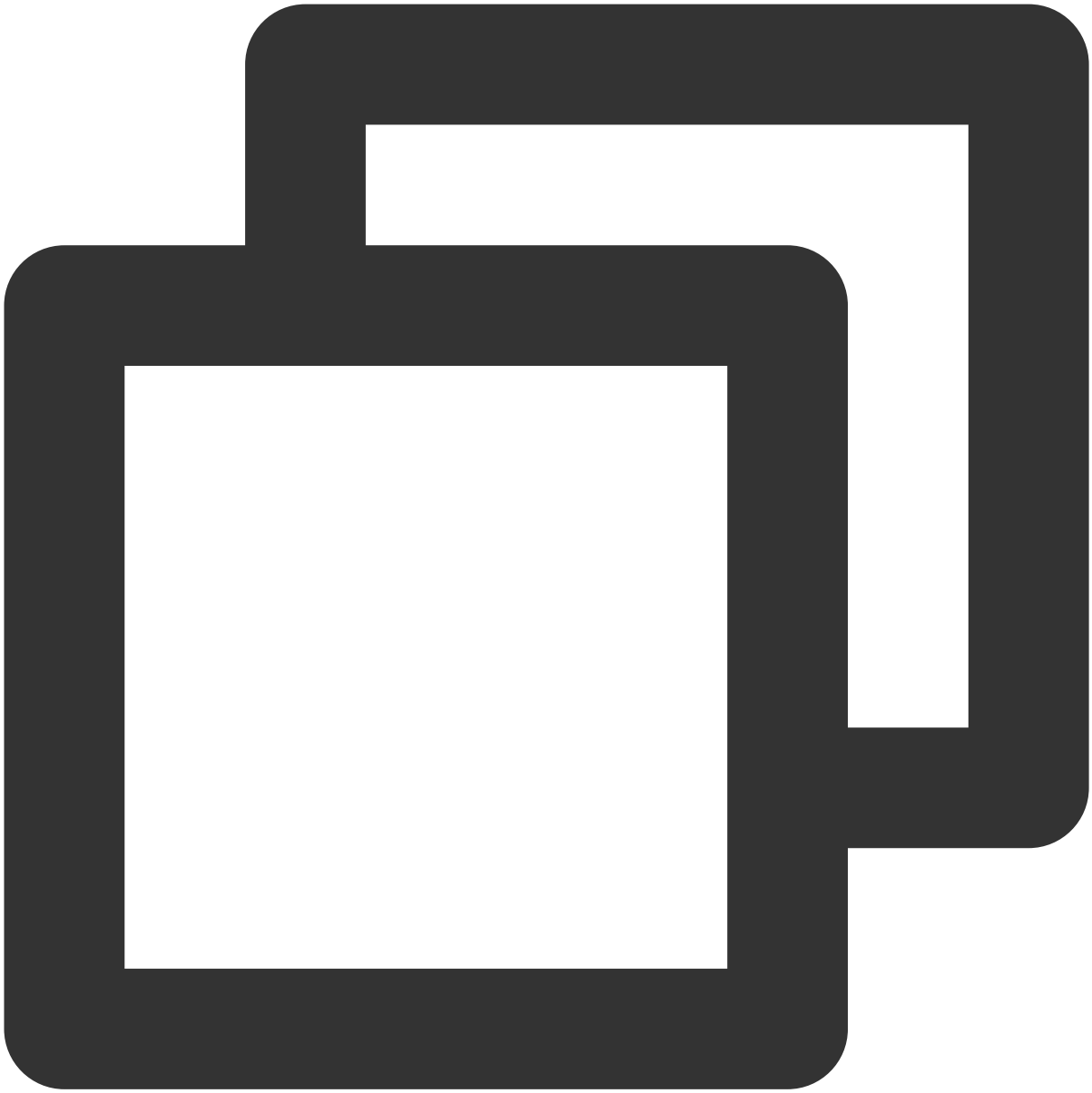
Java

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
```

### Step 2: Complete the beauty processing in the TRTC custom rendering logic in Native.

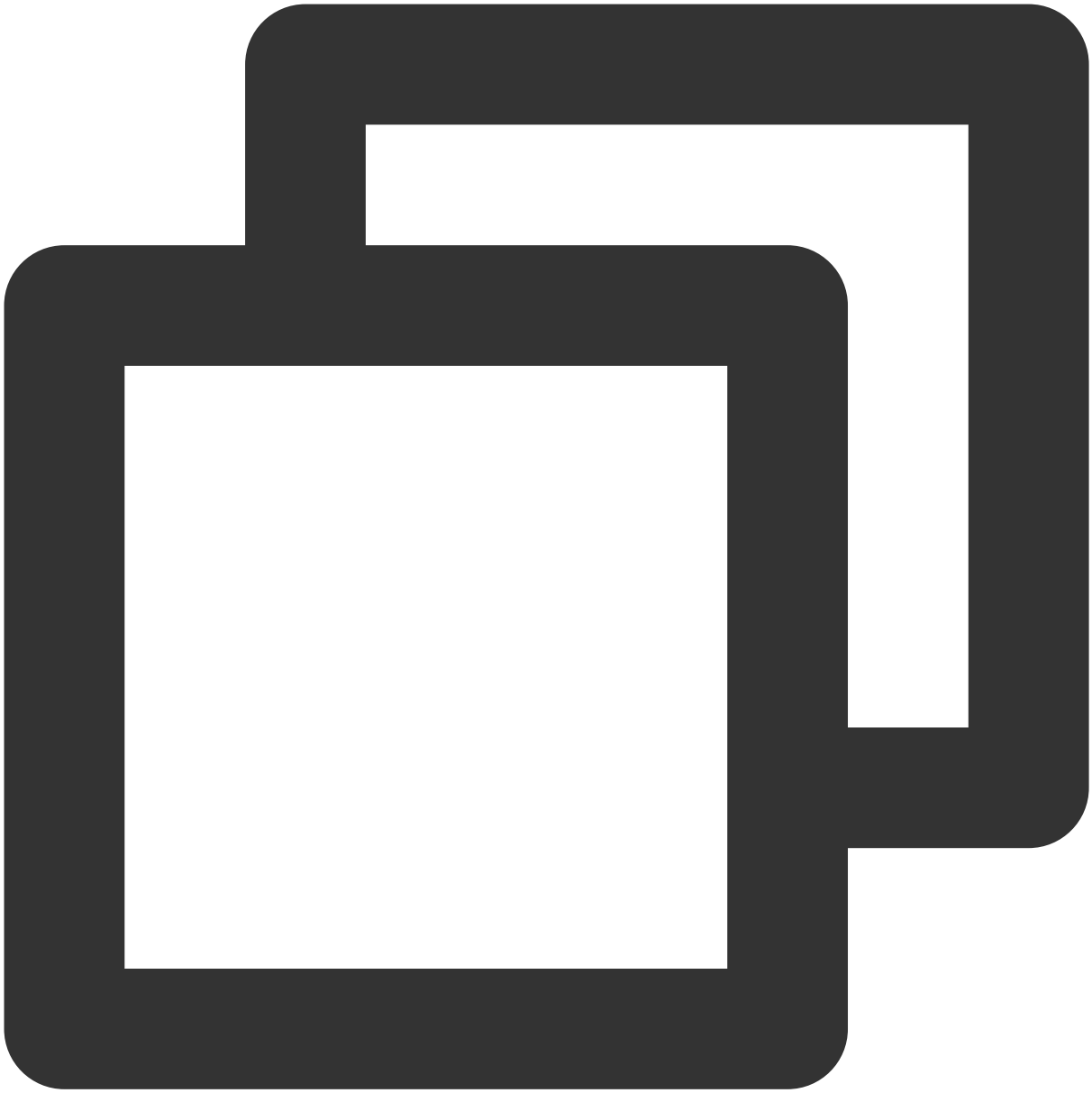
**Note:**

Android requires a dependency on **LiteAVSDK_Professional during the beauty integration process. Add the following dependency in the Android project's app/build.gradle:**

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

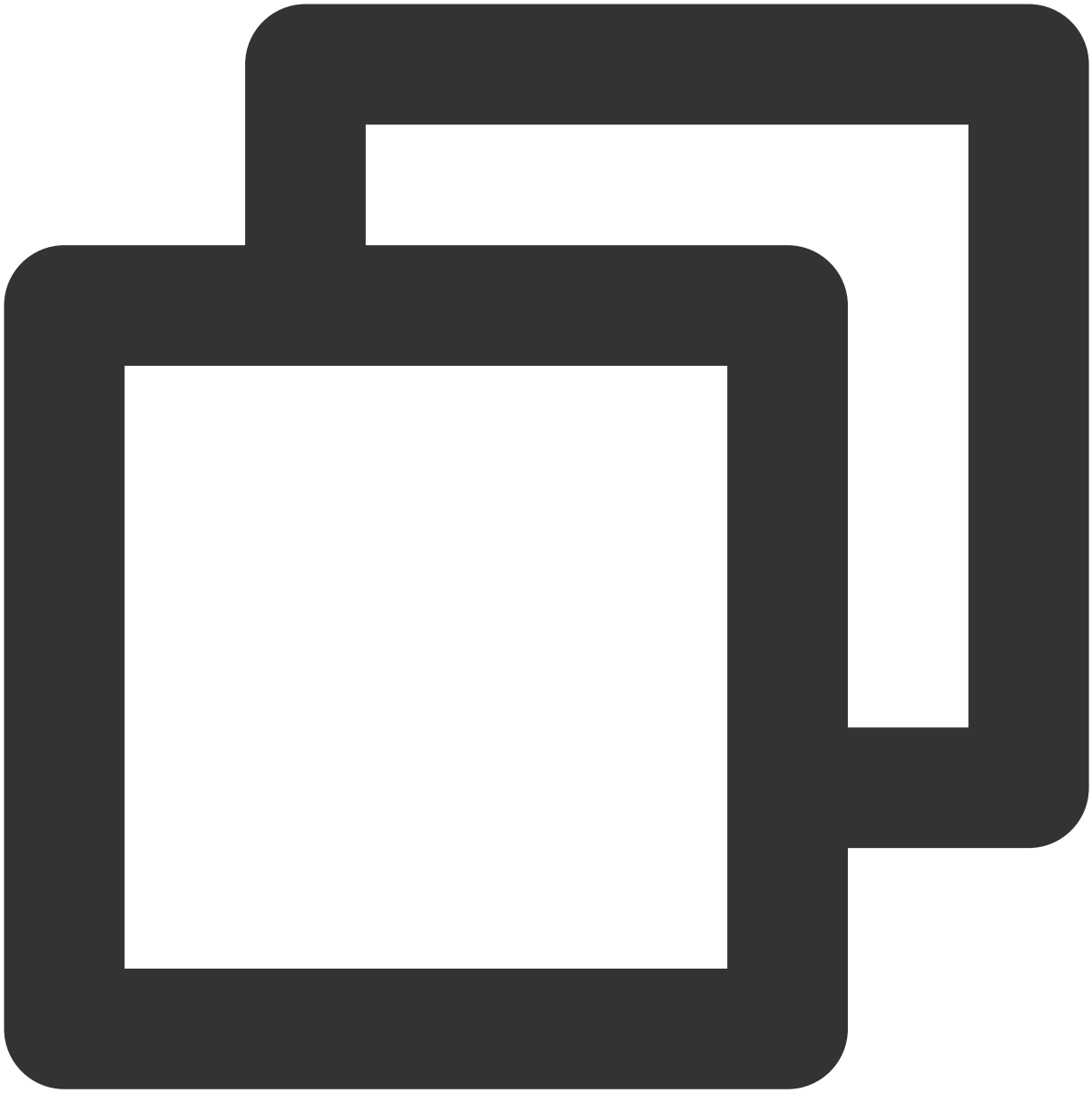
Java

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) {
        //Beauty processing logic
        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)
        //Beauty processing logic
        bueutyModel.onProcessVideoFrame(srcFrame, dstFrame)
        …

        return 0
    }
}S
```

### Step 3: Customize third-party beauty parameter control logic.

In this step, users can use a specific beauty module as needed. See Step 1 for the implementation of beauty parameter settings. The specific implementation depends on the actual use case.

## Integrating Tencent Beauty Effects

The method of integrating Tencent beauty effects also follows the above-mentioned steps. The following description takes Tencent beauty effects as an example to illustrate the method of integration.

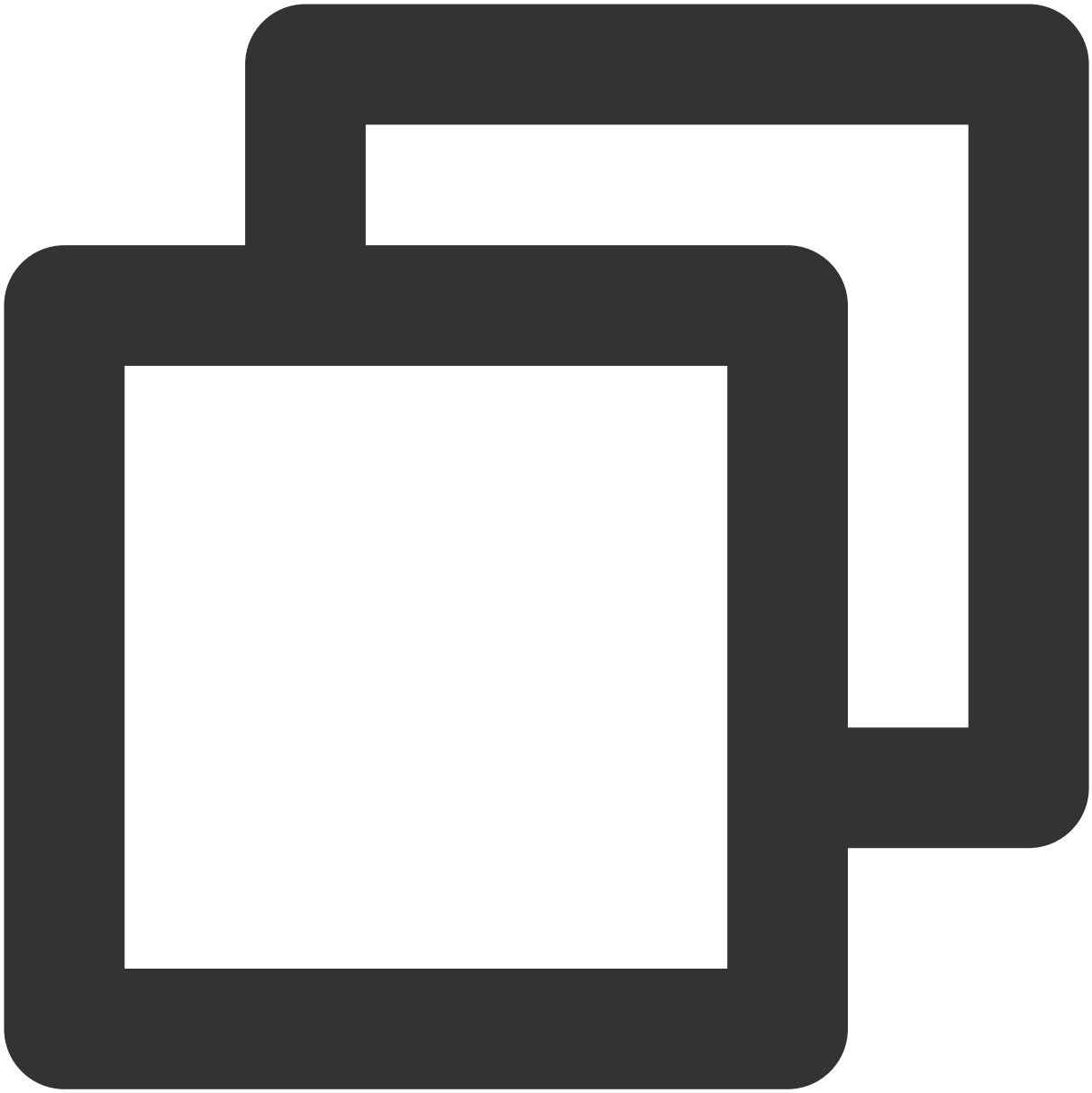
### Step 1: Download and integrate beauty effect resources.

1. Based on the package you purchased, Download SDK.
2. Add the resource files to your own project:

Android

iOS

1. In the app module, find the build.gradle file and add the Maven reference address corresponding to your package. For example, if you have chosen the S1-04 package, add the following code:

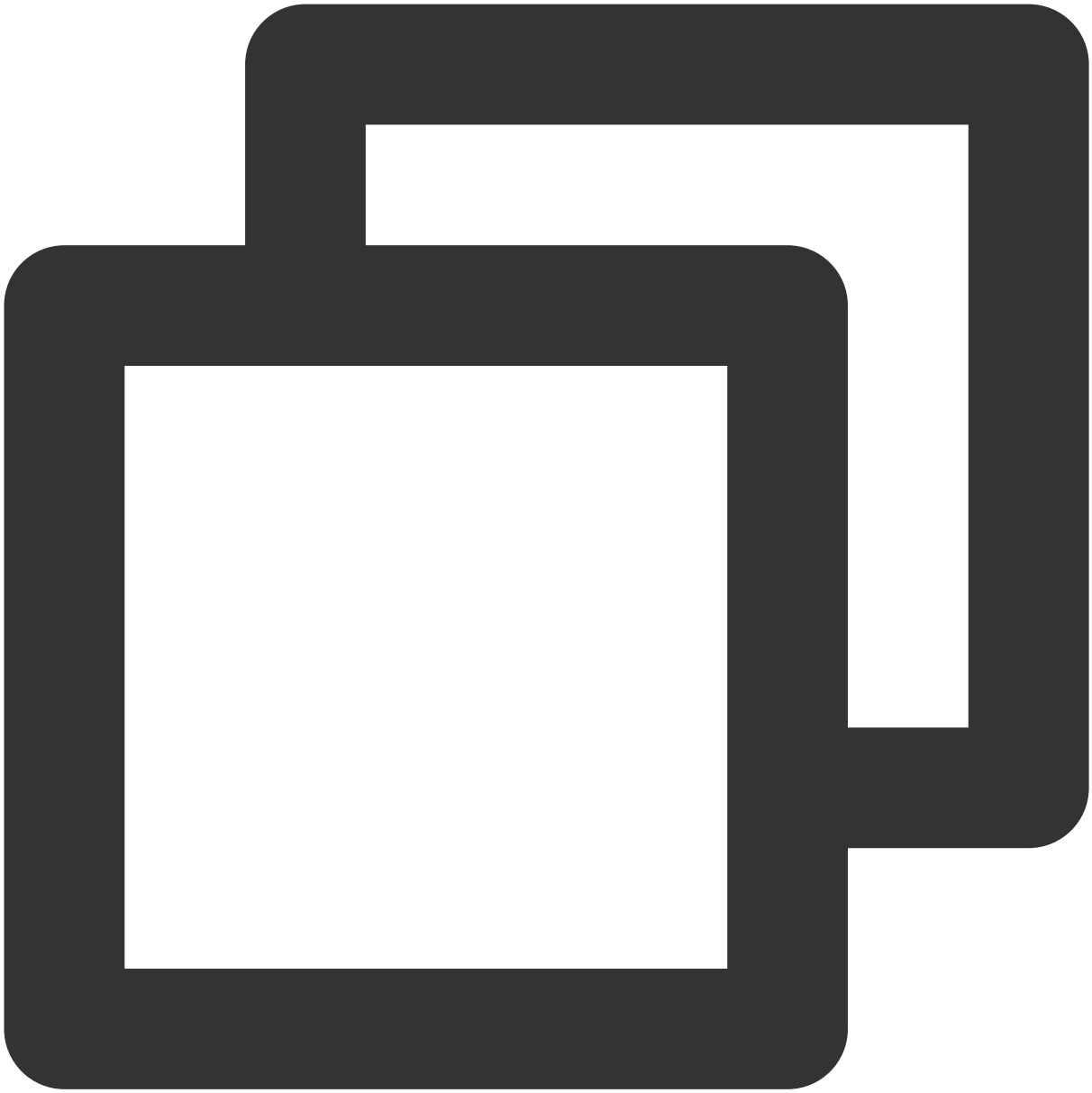


```
dependencies {
    implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
}
```

**For the Maven address corresponding to each package, refer to Documentation.**

2. In the app module, find the src/main/assets folder, or create one if it does not exist. Check if the downloaded SDK package includes the MotionRes folder, and if so, copy this folder to the `../src/main/assets` directory.

3. Find the AndroidManifest.xml file in the app module, and add the following tag in the application element.



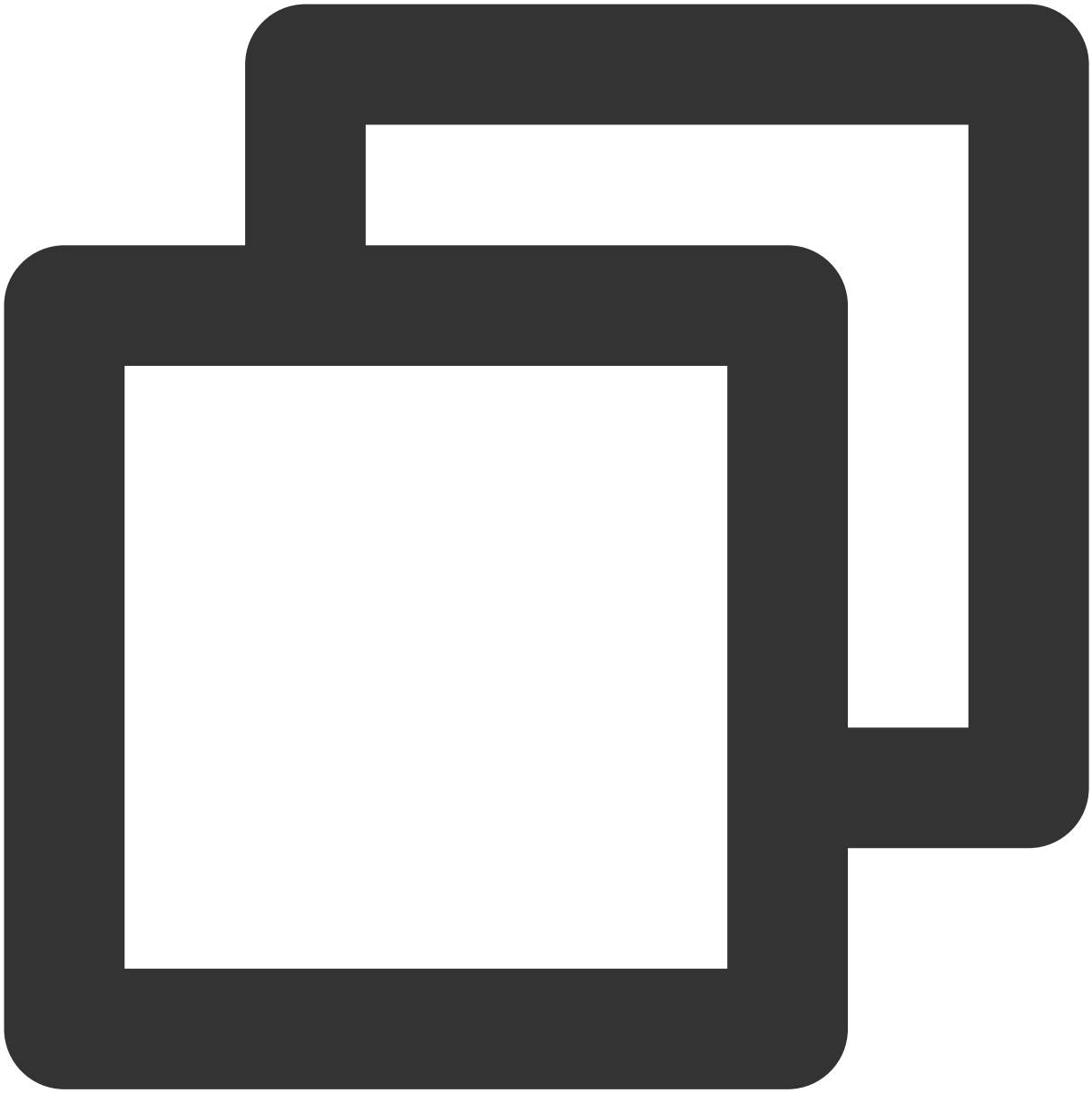
```
<uses-native-library
       android:name="libOpenCL.so"
       android:required="true" />
       //The "true" here indicates that if the library is unavailable, the applica
       //The "false" indicates that the application can utilize the library (if av
       //Official Android website introduction: %!s(<nil>)
```

The tag is added, as shown in the following figure:


4. Aliasing Configuration

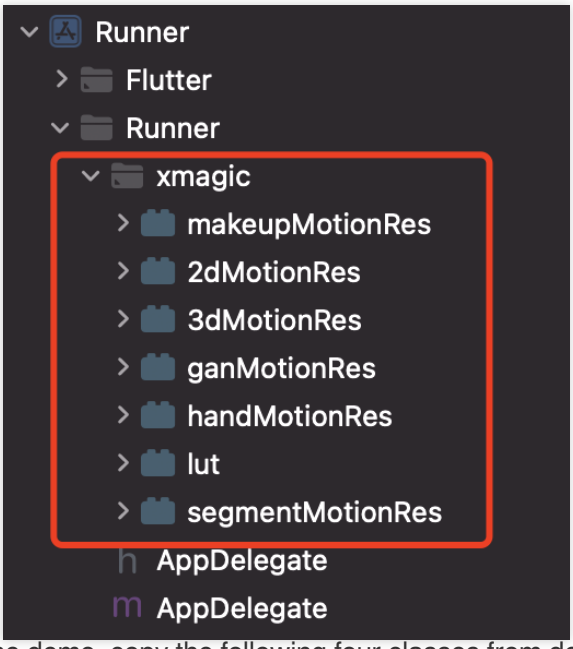
If you enable compile optimizations while building the release package (by setting minifyEnabled to true), some codes that are not called at the Java level might be trimmed. However, these codes could potentially be called at the Native level, leading to a `no xxx method` exception.

If you enable such compile optimizations, you must add these "keep" rules to prevent the codes of xmagic from being trimmed:



```
-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.** { *;}
```

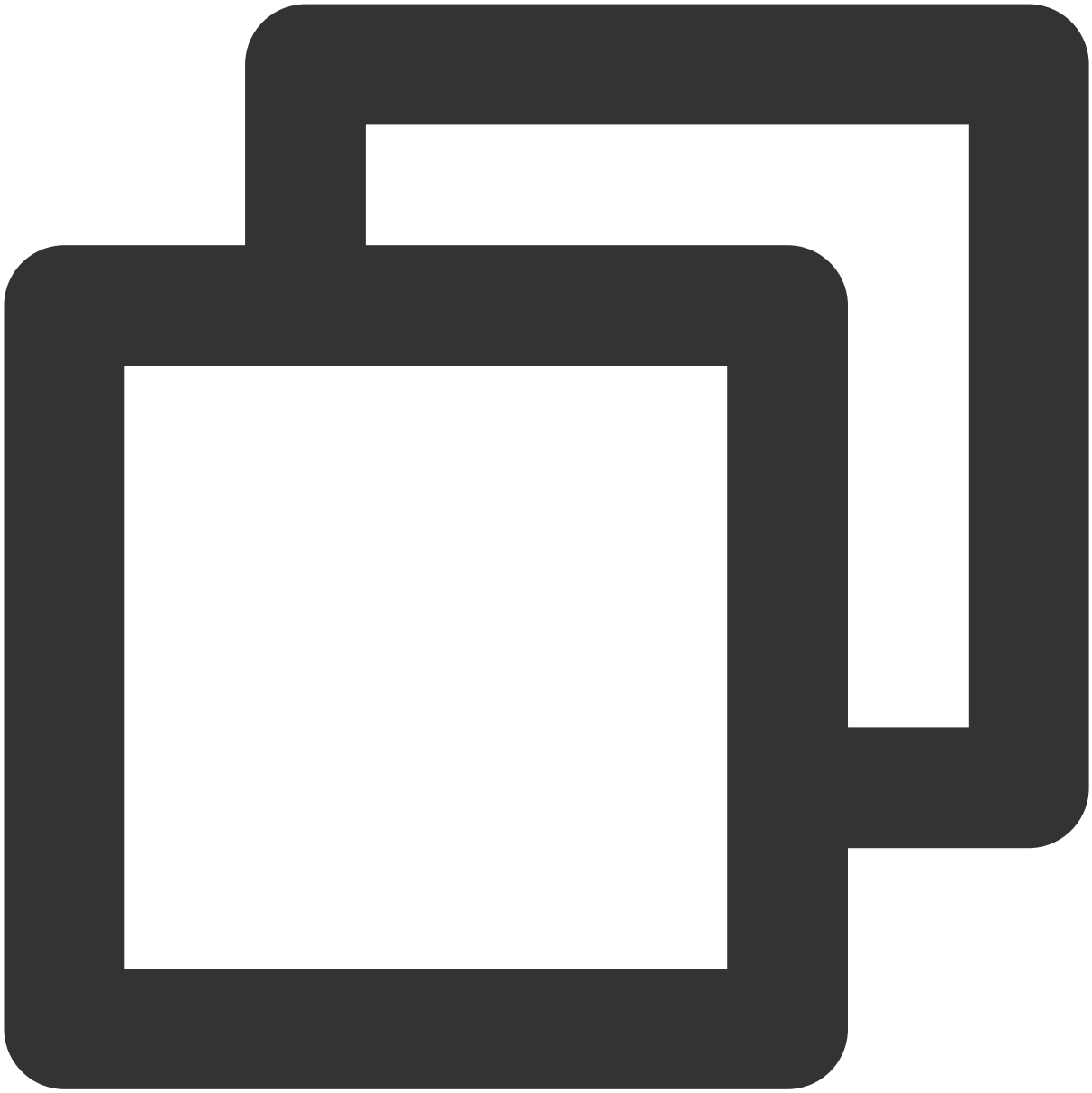
1. Add beauty effect resources to your project, as shown in the following figure (your resources may not exactly match those in the following figure):




2. In the demo, copy the following four classes from demo/lib/producer to your own Flutter project: BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid, and BeautyPropertyProducerIOS. These four classes are used to configure beauty effect resources and display the beauty types on the beauty panel.

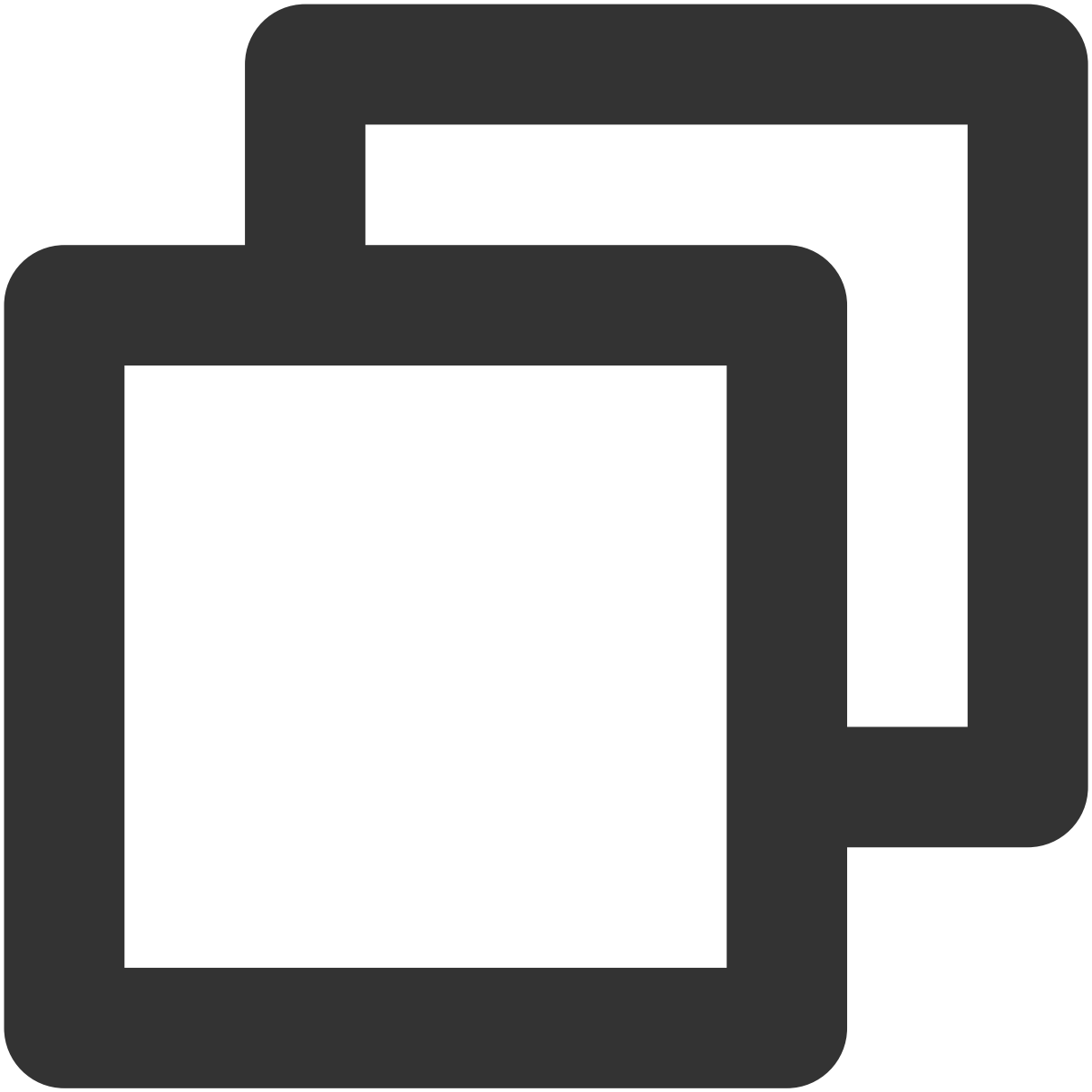
### Step 2: Reference the Flutter SDK.

**GitHub Reference**: Add the following reference in your project's pubspec.yaml file:



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

**Local Reference**: Download the latest version of tencent_effect_flutter, then add the folders `android` , `ios` , `lib` , and the files `pubspec.yaml` , `tencent_effect_flutter.iml` to the project directory. Next, add the following reference in your project's pubspec.yaml file (you can refer to the demo) :



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

tencent_effect_flutter merely provides a bridge, and it is the XMagic that is has dependancy on, which is set to the latest version by default, that actually implements beauty effects.

To use the latest version of the beauty SDK, you can upgrade the SDK by following these steps:

Android

iOS

Execute the `flutter pub upgrade` command in the project directory or click **Pub upgrade** in the top right corner of the `subspec.yaml` page.
Execute the flutter pub upgrade command in the project directory, and then execute the `pod update` command in the iOS directory.

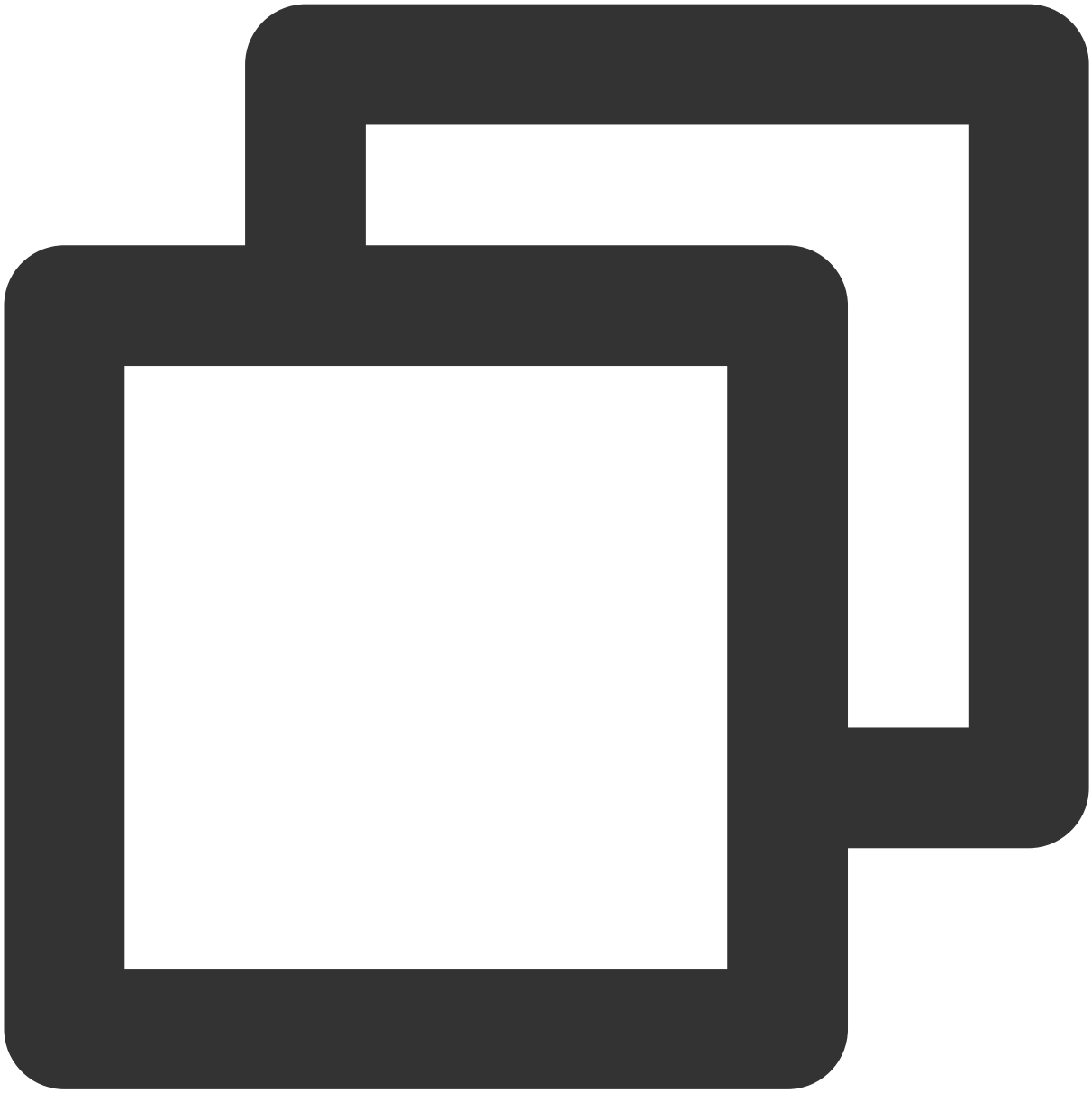
### Step 3: Implement the control interface of enabling/disabling beauty effects from Dart to Native.

For this part, refer to Integrating Third-Party Beauty Effects - Step 1.

### Step 4: Complete the beauty processing in the TRTC custom rendering logic in Native.

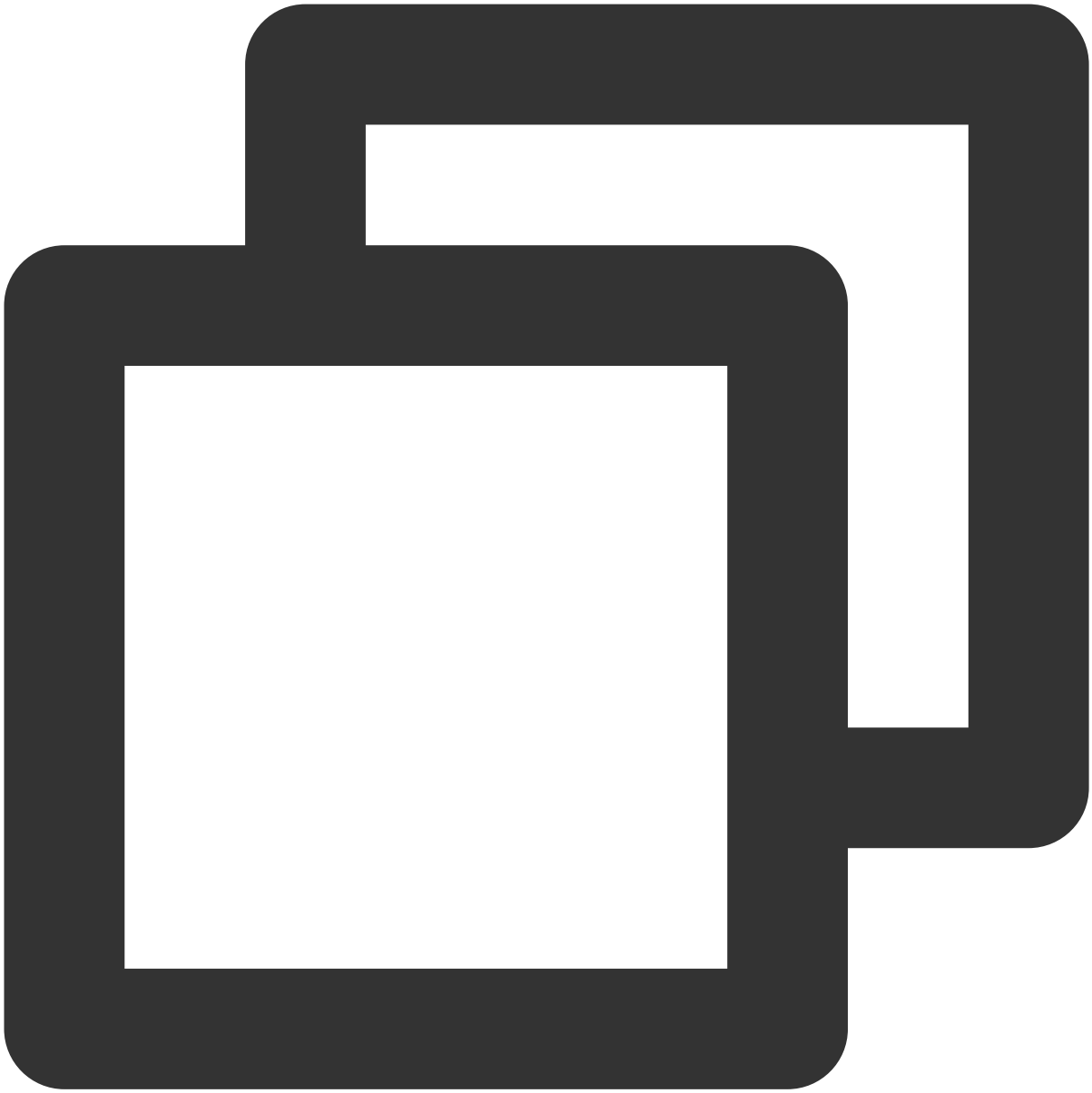
Android
iOS
Android requires a dependency on **LiteAVSDK_Professional during the beauty integration process. Add the following dependency in the Android project's app/build.gradle:**

```
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
    }
}
```

### Step 5: Enable beauty effects and set beauty parameters.

After completing the configurations above, you can enable/disable beauty effects by using `enableTUICallKitCustomBeauty()/disableTUICallKitCustomBeauty()` . You can set beauty parameters through the Tencent Effect beauty Flutter interface.