

# Chat

## Video Call (UI Included)

### Product Documentation



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.



# Contents

## Video Call (UI Included)

- Overview (TUICallKit)

- Activate Service (TUICallKit)

- Integration(TUICallKit)

  - Android

  - iOS

  - Web&H5

  - Flutter

- UI Customization (TUICallKit)

  - Android

  - Web

  - Flutter

- Additional Features(TUICallKit)

  - Group Call

    - Android&iOS&Flutter

    - Web&H5

  - Setting Nickname and Avatar (Full Platform)

  - Setting Resolution and Fill Pattern

  - Floating Window

    - Android&iOS&Flutter

    - Web&H5

  - Beauty Effects (TUICallKit)

    - Flutter

  - Custom Ringtone

    - Android

    - iOS

    - Web&H5

    - Flutter

  - Monitoring Call Status

    - Android&iOS&Flutter

    - Web&H5

- API Documentation(TUICallKit)

  - Android

    - API Overview

    - TUICallKit

- TUICallEngine
- TUICallObserver
- Type Definition
- ErrorCode

#### iOS

- API Overview
- TUICallKit
- TUICallEngine
- TUICallObserver
- Type Definition
- ErrorCode

#### Web

- API Overview
- TUICallKit
- TUICallEngine
- TUICallEvent

#### Flutter

- API Overview
- TUICallKit
- TUICallEngine
- TUICallObserver
- Type Definition
- ErrorCode

#### Release Notes (TUICallKit)

- Web(Vue)
- Android & iOS
- Flutter

# Video Call (UI Included)

## Overview (TUICallKit)

Last updated : 2024-05-08 11:30:04

### Component Overview

TUICallKit is an audio and video call UI component launched by Tencent Cloud. By integrating this component, you can add audio and video call features to your application with just a few lines of code.




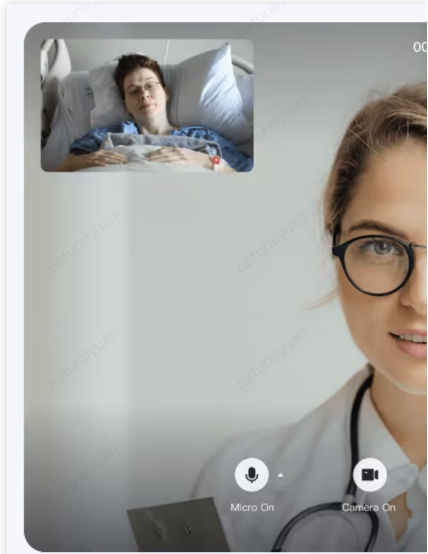
### Supported Platform

	Android	iOS	Web	uni-app mini programs	Flutter	uni-app client	WeCha Mini Progra
Supported							
Supported Languages/Frameworks	Kotlin Java	Swift Objective-C	Vue3 Vue2 React	Vue3 Vue2	Dart	Vue3 Vue2	-

Description of the Feature

Basic Feature	Advanced Feature	Advantages of the Feature
1v1 Voice/Video Call <b>Group Call</b> , Invite Others Mid-call, Join Mid-call Customize Ringtone Customize Nickname, Avatar <b>Enable/Disable Floating Window</b> Enable/Disable Ringtone	<b>Offline Push</b> <b>Virtual Background</b> On-cloud Recording AI Noise Reduction Global Interconnectivity Weak Network Jitter Optimization Call Records	<b>Comprehensive UI Interaction</b> <b>Support for Cross-platform Interconnection</b> <b>Support for Multi-device Login</b>

Use Case

Online Social Interaction	Online Consultation
	
Online Education	Online Sales



If you have any requirements or feedback, contact [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Activate Service (TUICallKit)

Last updated : 2024-07-17 11:34:00

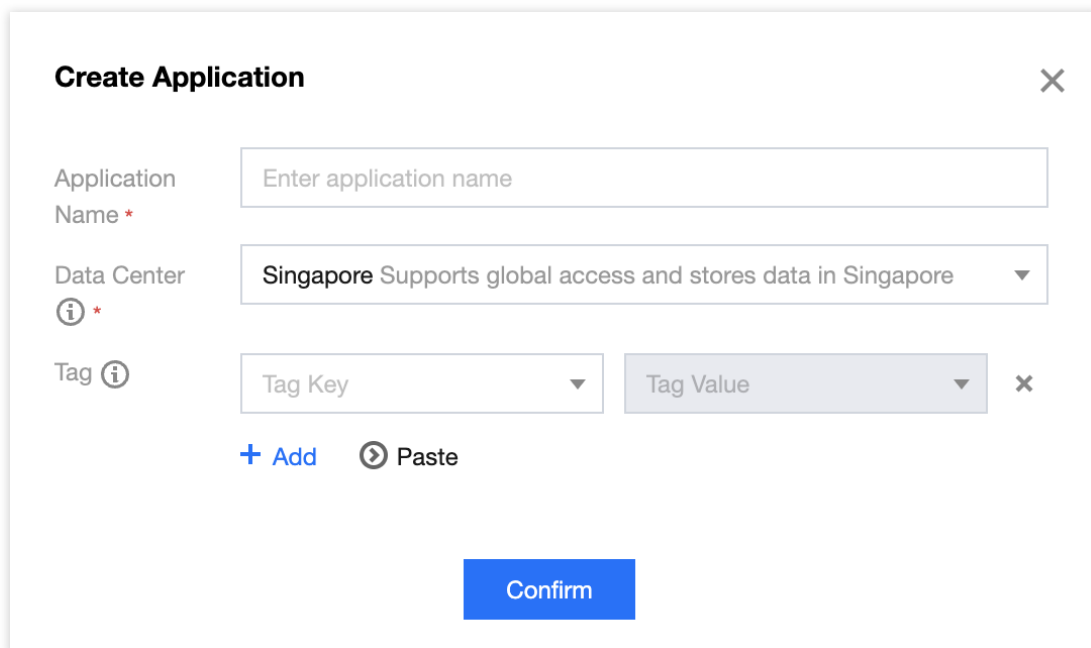
This document describes how to activate audio and video call services.

## Activate Trial Call takes effect for an individual

TUICallKit is a video and audio communication component built on Tencent Cloud's [Instant Messaging](#) and [Tencent Real-Time Communication TRTC](#), two paid PaaS services. To provide you with a better experience of the audio and video call feature, we offer a 7-day trial for each SDKAppID for free (the trial version does not grant additional call duration). Each SDKAppID can experience the trial twice, with each trial lasting 7 days; meanwhile, the total number of trials for all SDKAppIDs under one account is 10 times.

You can activate the Call Trial Version in the Instant Messaging console, with the following specific steps:

1. Visit the [IM Console](#), select a data center, and create a new application. If you already have an application, you can skip this step.



The screenshot shows a 'Create Application' dialog box with a close button (X) in the top right corner. It contains the following fields and controls:

- Application Name \***: A text input field with the placeholder 'Enter application name'.
- Data Center (i) \***: A dropdown menu showing 'Singapore Supports global access and stores data in Singapore'.
- Tag (i)**: A section containing two dropdown menus, 'Tag Key' and 'Tag Value', and a close button (X). Below these are two buttons: '+ Add' and 'Paste'.
- Confirm**: A blue button at the bottom center.

2. Click the target app card to go to the basic configuration page of the app.
3. Find the **Audio and Video Calling Capability (Call)** card, click **free trial**.

4. After confirming the content of the pop-up, click **Experience Now** to successfully activate the audio and video calling trial version.

**Try Call service for free**

The Call is a call component that comes with a UI kit. You can use it to implement features including 1-to-1/group calls and multi-platform call making/answering.

- The Call provides **7-day free trial** to let you try out all audio/video call features.
- Enabling this will activate TRTC service and create application for you by default as the feature is based on Chat and TRTC services.
- This free trial **only grants you the right to use the call features. Your calls will be billed as stated in [Billing of TRTC Basic Services](#)**, and relevant Chat services will be billed as stated in [Billing Overview](#) of Chat.
- Each Tencent Cloud account (UIN) with Call activated will get **10,000 free minutes** every month. The free minutes can deduct the usage of calls. For more information, see [here](#).

[Activate now](#)

5. If your service is going live and you need to purchase the official version in the console. Please refer to: [Buy Official Version](#).

## Feature and Billing Description

TUICallKit is supported by the underlying technology provided by Tencent Real-Time Communication TRTC and Instant Messaging. **You need to purchase a specified package to use TUICallKit.** The activation method for the free trial version can be found in [Activate TUICallKit Trial Version](#).

The table below displays the recommended packages for TUICallKit along with their features and prices. You can also freely combine TRTC's Monthly Package and the IM version to get the corresponding version of TUICallKit. The Basic version of TRTC, when combined with the Professional or Enterprise version of IM, can use the 1v1 Call Version of TUICallKit; The Deluxe and Enterprise Editions of TRTC, when combined with the Professional or Enterprise Edition of IM, can use the Group Call Version of TUICallKit.

Item		Trial	1-to-1 Call	Group Call
Price		7-day free trial	199 USD/month <a href="#">Buy Now</a>	597 USD/month <a href="#">Buy Now</a>
Free resources	<a href="#">Free minutes</a>	10,000 minutes/month	10,000 minutes/month	10,000 minutes/month
	<a href="#">Package bonus minutes</a>	-	100,000 minutes/month	300,000 minutes/month
	<a href="#">Quota of free monthly active users (MAU)</a>	100/month	5,000/month	10,000/month
	Pay-as-you-go upon exhaustion (within the validity of the package)	Services become unavailable after exhaustion.	✓	✓
Call features	Audio/Video calls	✓	✓	✓
	Complete UI	✓	✓	✓
	Call status display	✓	✓	✓
	Call notifications and offline push (If the application is not in the foreground, push notifications will be sent.)	✓	✓	✓
	Floating window (The call page can be displayed as a floating window.)	✓	✓	✓



Custom ringtones	✓	✓	✓
Make/Answer/Decline/Hang up a call	✓	✓	✓
Video call switch to Audio call	✓	✓	✓
1-to-1 call	✓	✓	✓
Group call	✓	-	✓
Virtual background	✓	-	✓
Invite to/Join ongoing calls	✓	-	✓
Call History API (Support obtaining call history data through Callback and REST API methods.)	✓	-	✓
Multi-platform call (A successful connection will automatically terminate requests from other platforms.)	✓	-	✓
Multi-device call (A user can be logged in to multiple devices of the same platform, such as several iOS devices. When a call is answered, the device that connects will automatically prevent other devices from accessing the call.)	✓ (Utilize in conjunction with the IM Premium Edition)	-	✓
AI noise suppression (Removes background noises with the help of AI.)	✓	-	✓
Less stutter under poor network conditions (Reduces stutter rate and loading time under poor	✓	-	✓

	network conditions.)			
Supported platforms		iOS、Android、Web、Flutter	iOS、Android、Web、Flutter	iOS、Android、Web、Flutter

**Note**

:

1. Free resources: The free resources can be used to deduct call durations and quota of free monthly active users (MAU). If you use the trial version, services will become unavailable for your application after you use up the package. If you use TRTC 1-to-1 Call or TRTC Group Call, after you use up the package, your additional usage will be charged at pay-as-you-go rates.
2. Free minutes: Each Tencent Cloud account (UIN) will get 10,000 free minutes per usage cycle (a usage cycle is one month) after buying a TRTC Call package. The free minutes can be used to deduct call durations of TRTC Call features and on-cloud recording and mixtranscoding durations of TRTC basic features. To learn more, see [Free Minutes](#).
3. Package bonus minutes: The bonus minutes can be used to deduct call durations of TRTC Call features. The bonus minutes are valid for one month and will expire at the end of each usage cycle.
4. Quota of free monthly active users (MAU): The number of unique users that log in to the Chat app in a given month, regardless of their number of repeated visits.
5. TRTC basic features: In addition to TRTC Call features, you can also use TRTC's basic features, which will incur additional fees. For the billing details, see [Billing of On-Cloud Recording](#) and [Billing of MixTranscoding and Relay to CDN](#).

# Integration(TUICallKit)

## Android

Last updated : 2024-05-15 17:00:53

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.

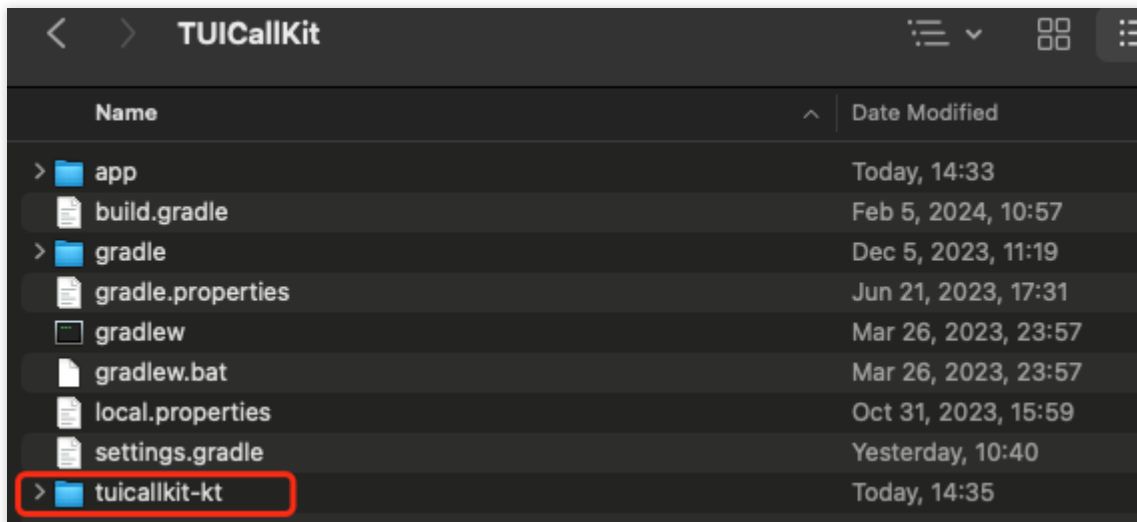
Mobile phone on Android 5.0 or later.

## Step 1. Activate the Service

Before using the audio and video services provided by Tencent Cloud, you need to go to the Console to activate the audio and video service for your application. For specific steps, please refer to [Activating the Service](#).

## 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:

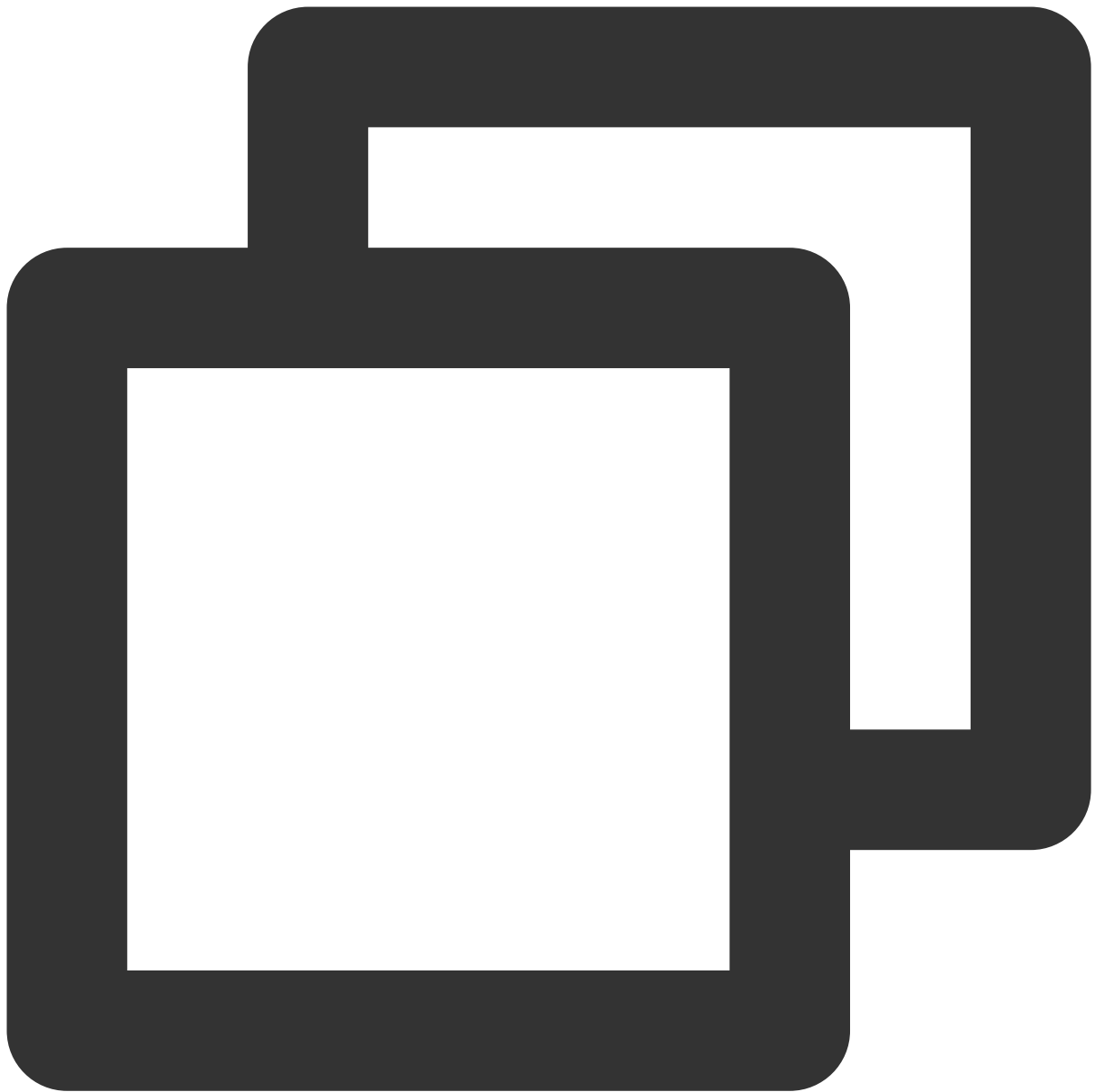


## Step 3. Configure the Project

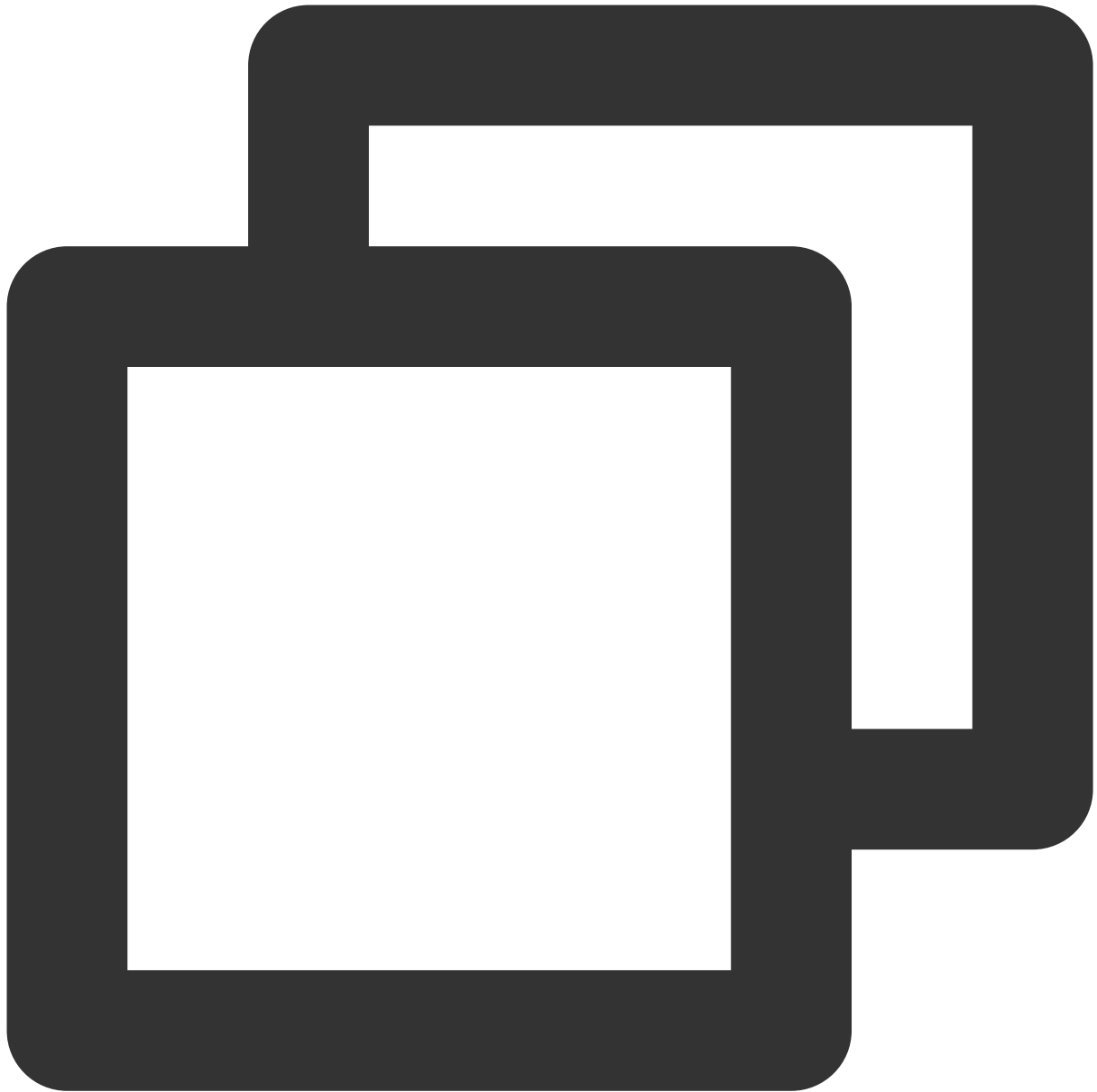
1. Find the `settings.gradle` (or `settings.gradle.kts`) file in the project root directory and add the following code to import the `TUICallKit` component downloaded in [step 2](#) to your current project:

`settings.gradle`

`settings.gradle.kts`



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

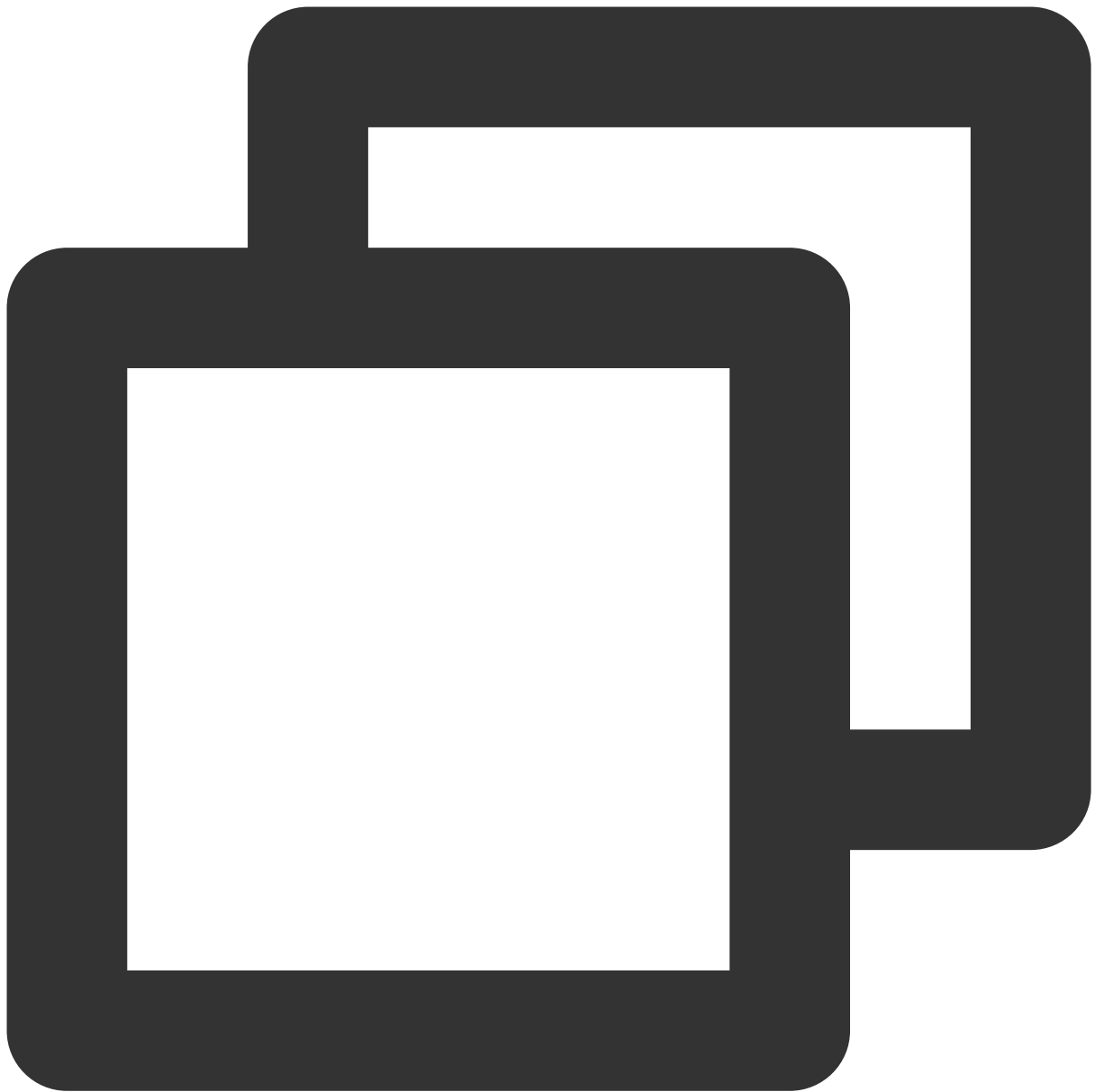


```
include(":tuicallkit-kt")
```

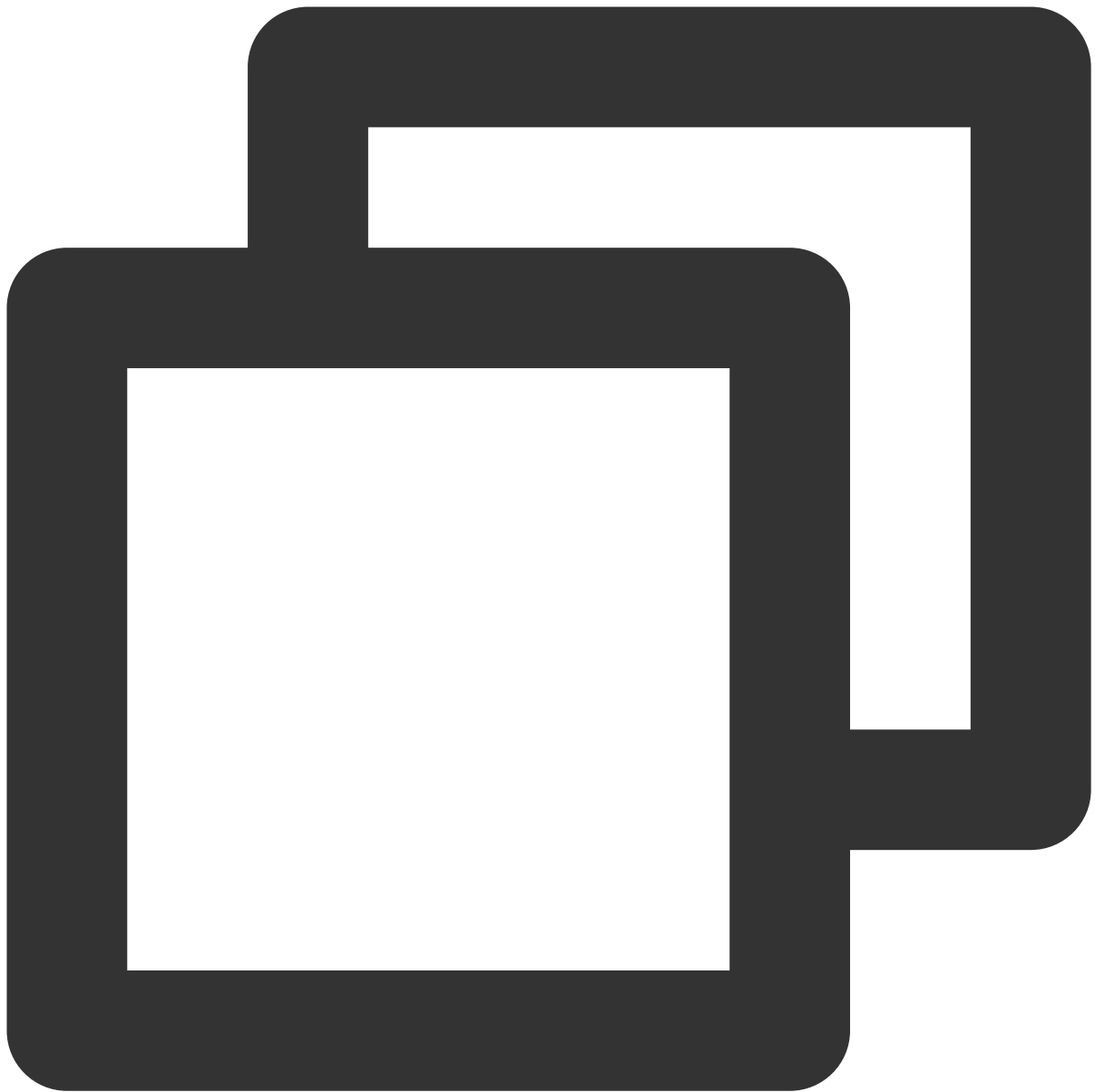
2. Find the `build.gradle` (or `build.gradle.kts`) file in the `app` directory and add the following code to declare the dependencies of the current application on the component just added:

`build.gradle`

`build.gradle.kts`



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



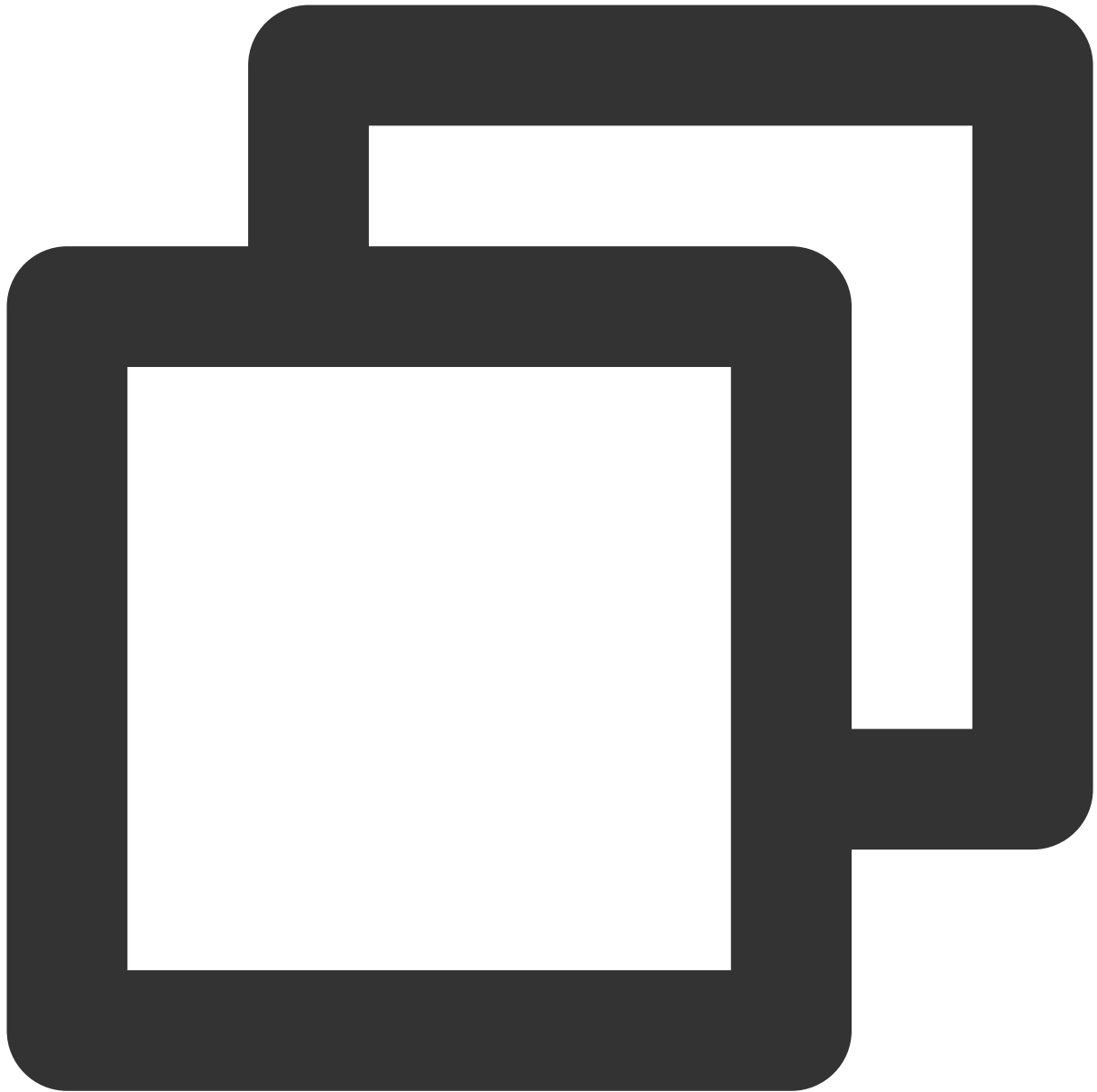
```
api(project(":tuicallkit-kt"))
```

**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:





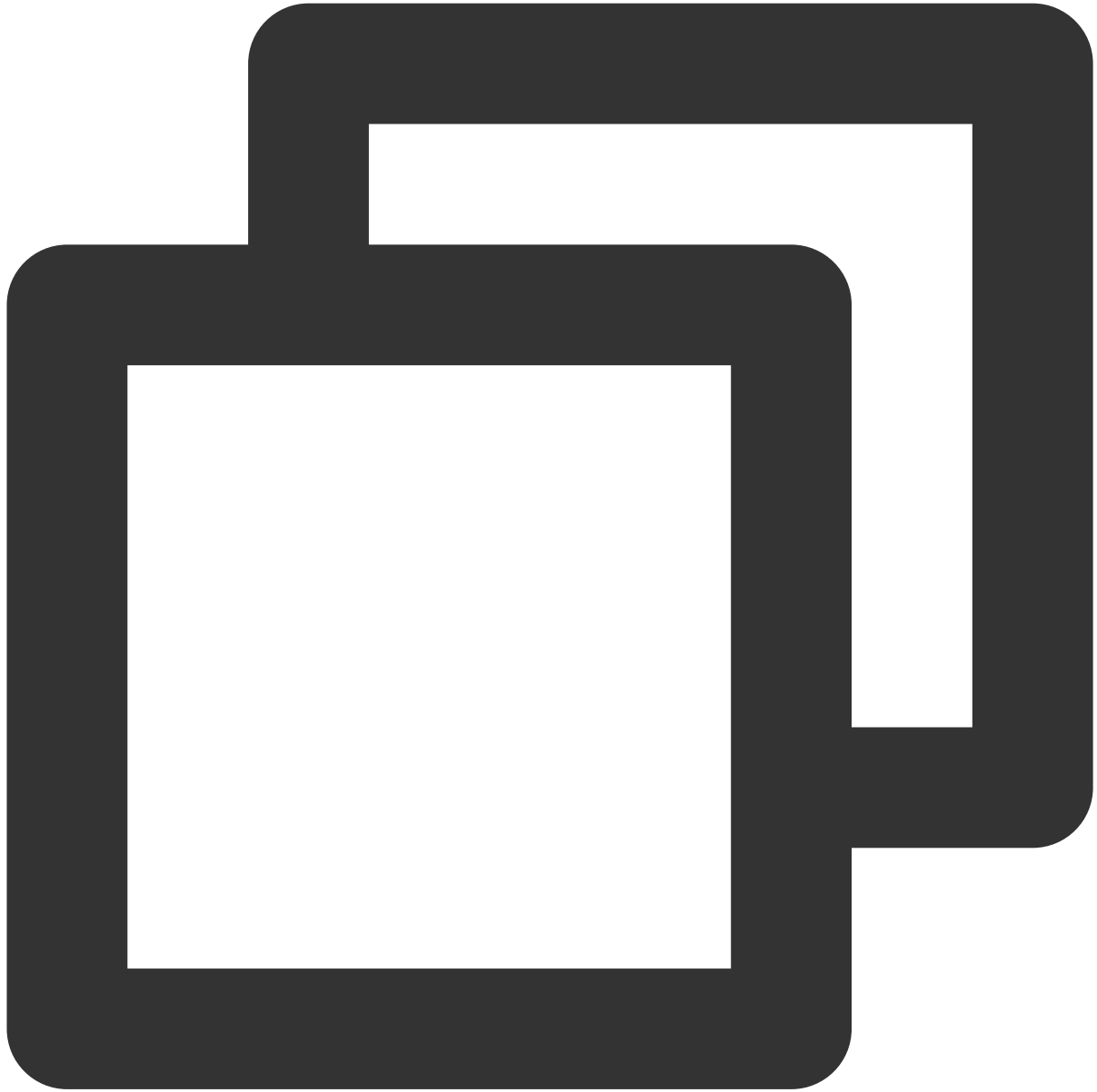
```
-keep class com.tencent.** { *;}
```

**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. Make sure 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 here
    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 no details are required 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 by mistake. 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.

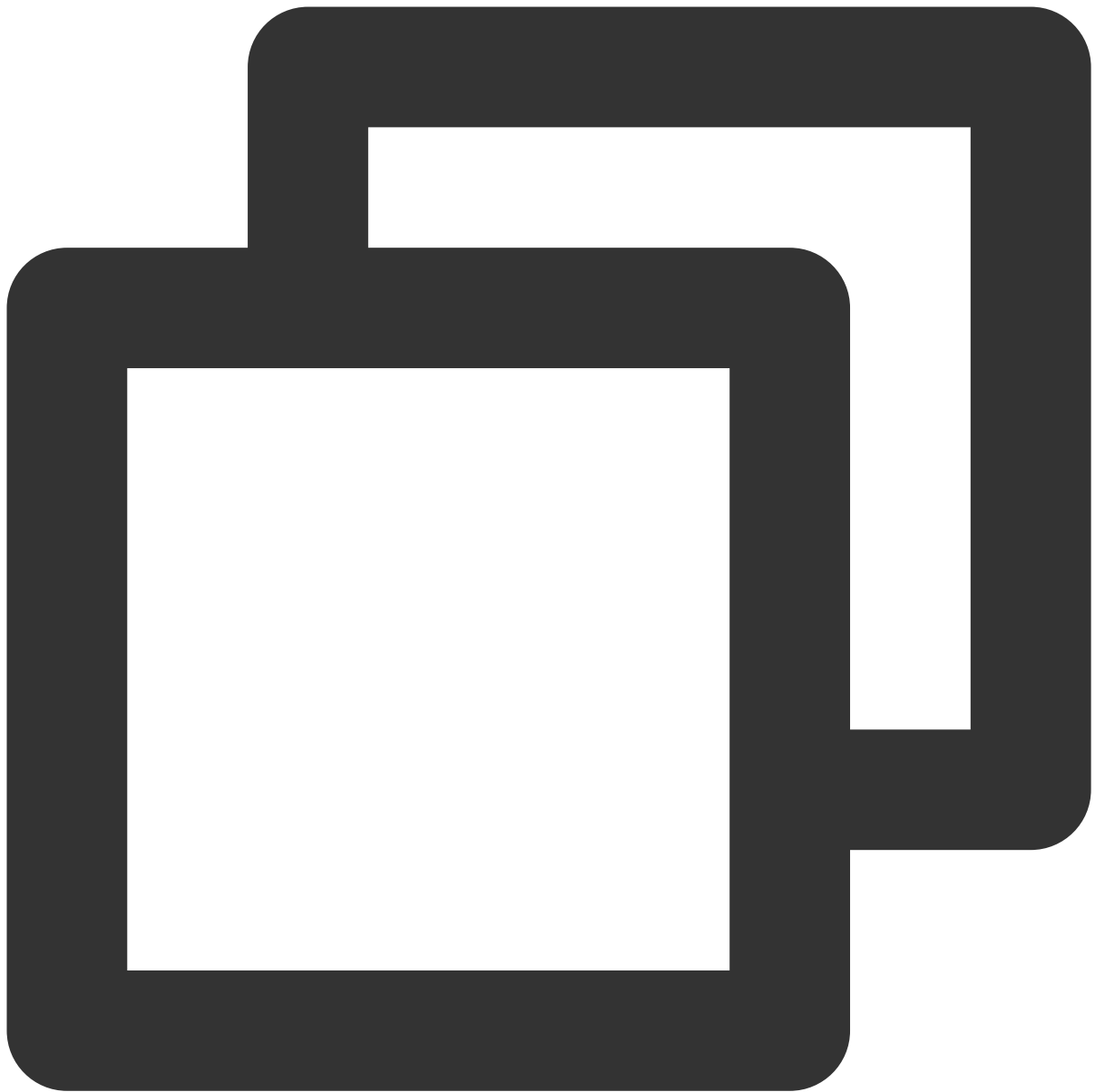
The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current access process faster. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` subsequently.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

## Step 5. Make Your First Call

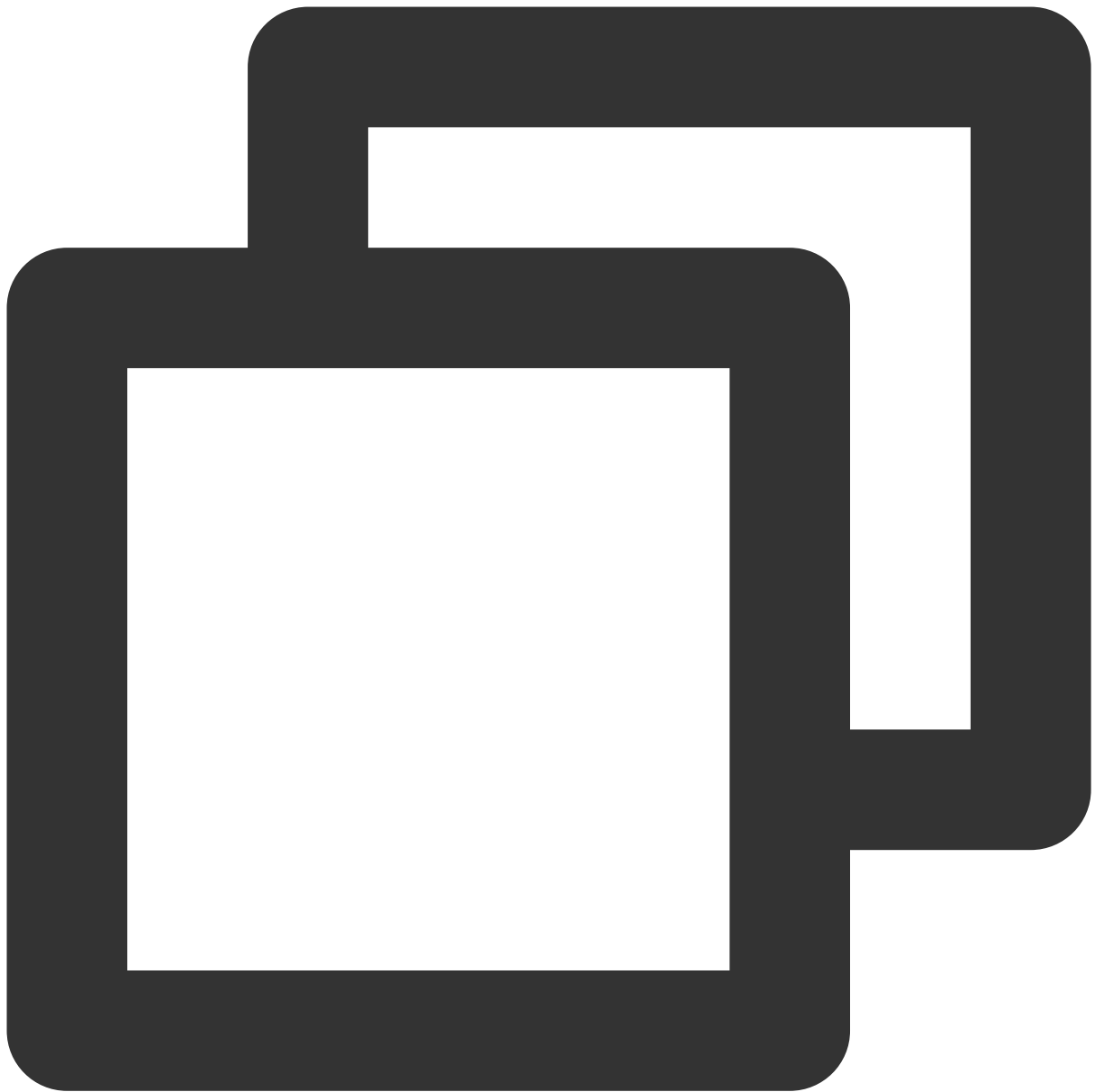
After the caller and callee have successfully signed in, the caller can initiate an audio or video call by calling the `TUICallKit`'s call method and specifying the call type and callee's userId. The callee will then receive the incoming call invitation.

Kotlin

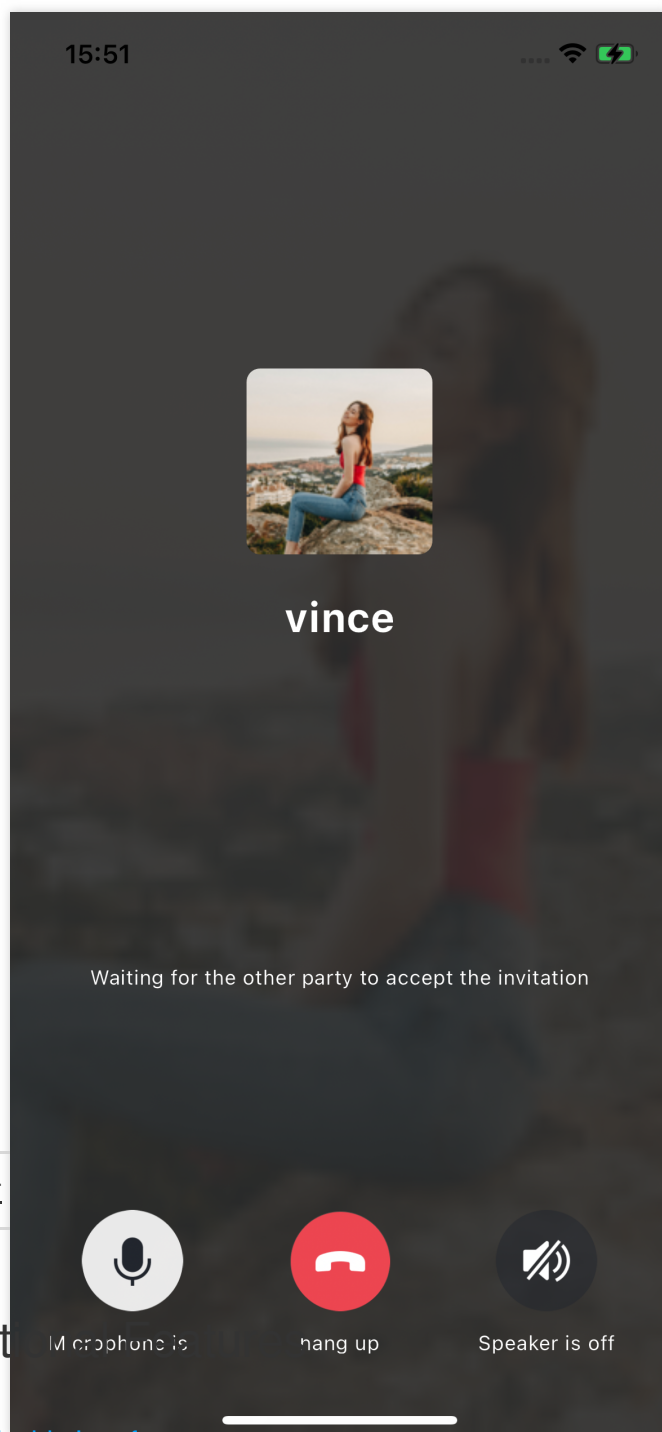
Java



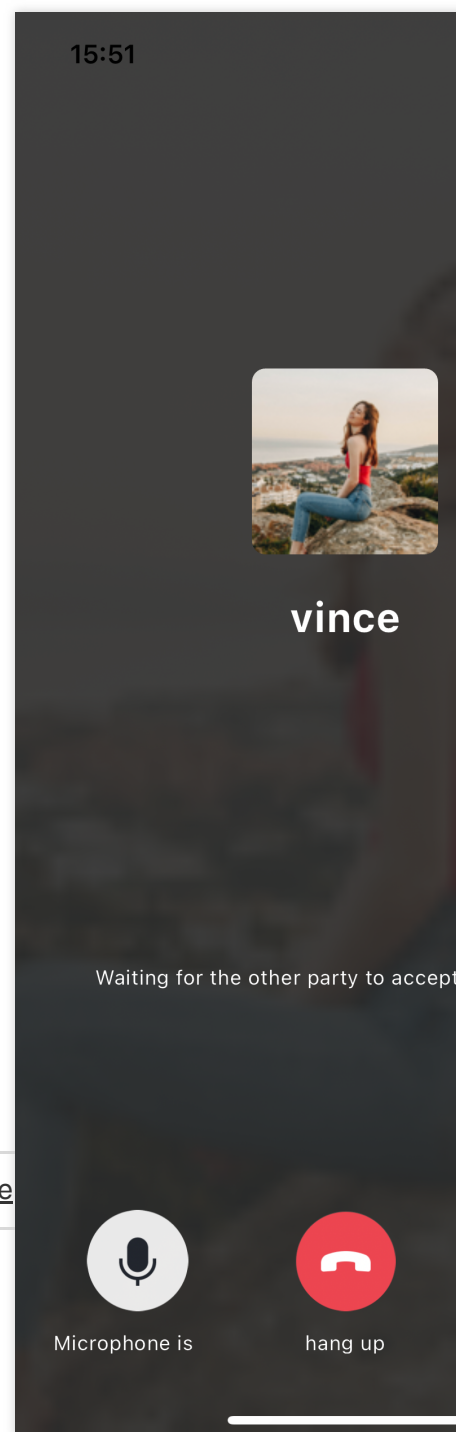
```
//Make a 1v1 video call (assuming UserID is mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video)
```



```
// Make a 1v1 video call (assuming UserID is mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video);
```



Caller



Callee

## Addit

[Customizable Interface](#)

[Group Call](#)

[Floating Window](#)

[Custom Ringtone](#)

[Monitor call status](#)

[Cloud Recording](#)

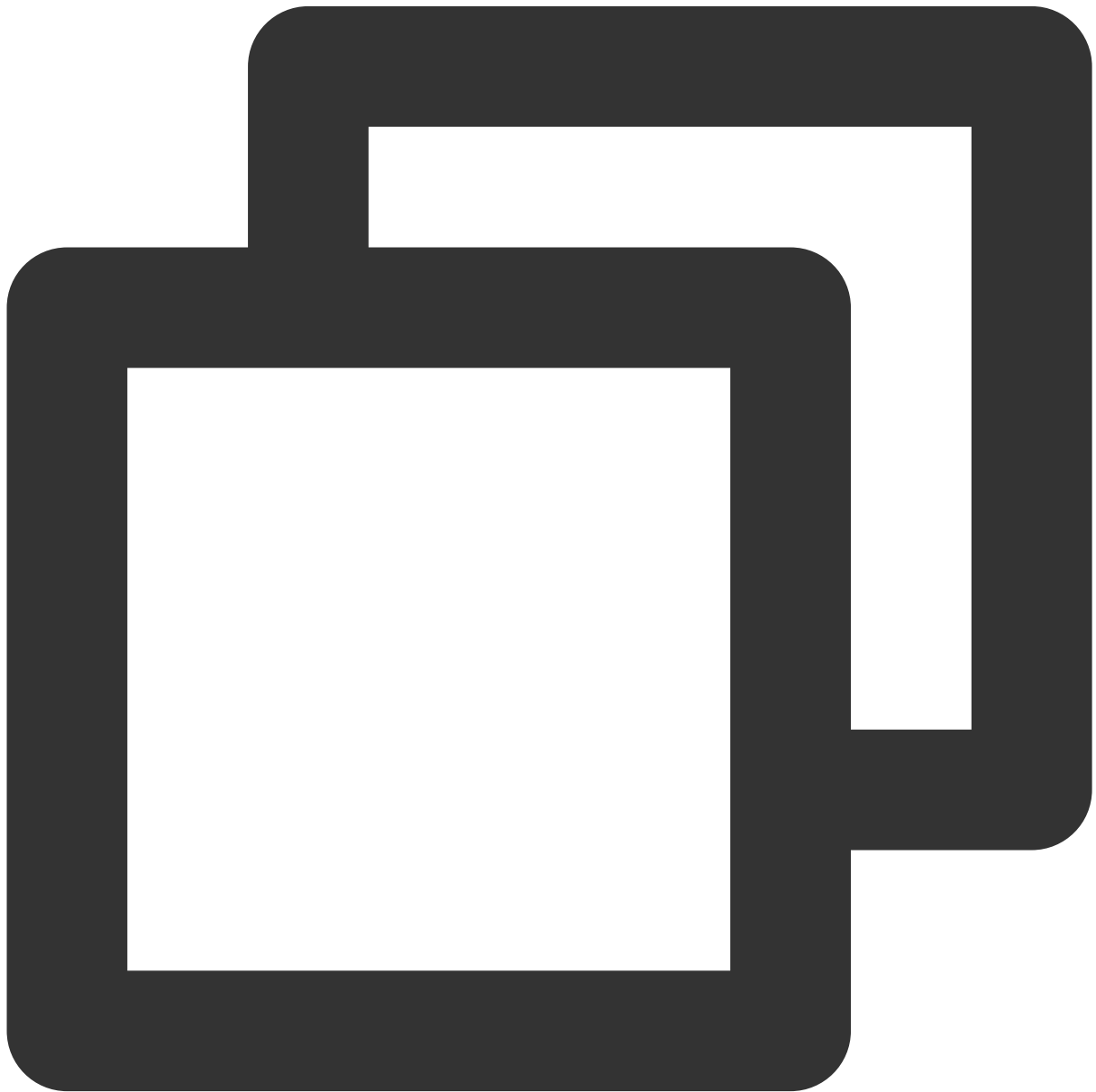
## FAQs

## 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.java:17)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.init(ServiceInitializer.java:17)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.onCreate(ServiceInitializer.java:20)
    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.java:59)
    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](mailto:info_rtc@tencent.com).



# iOS

Last updated : 2024-05-15 17:01:30

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

Before using Tencent Cloud's audio and video services, you need to go to the Console and activate the audio and video service for your application. For specific steps, please refer to [Activate the Service](#).

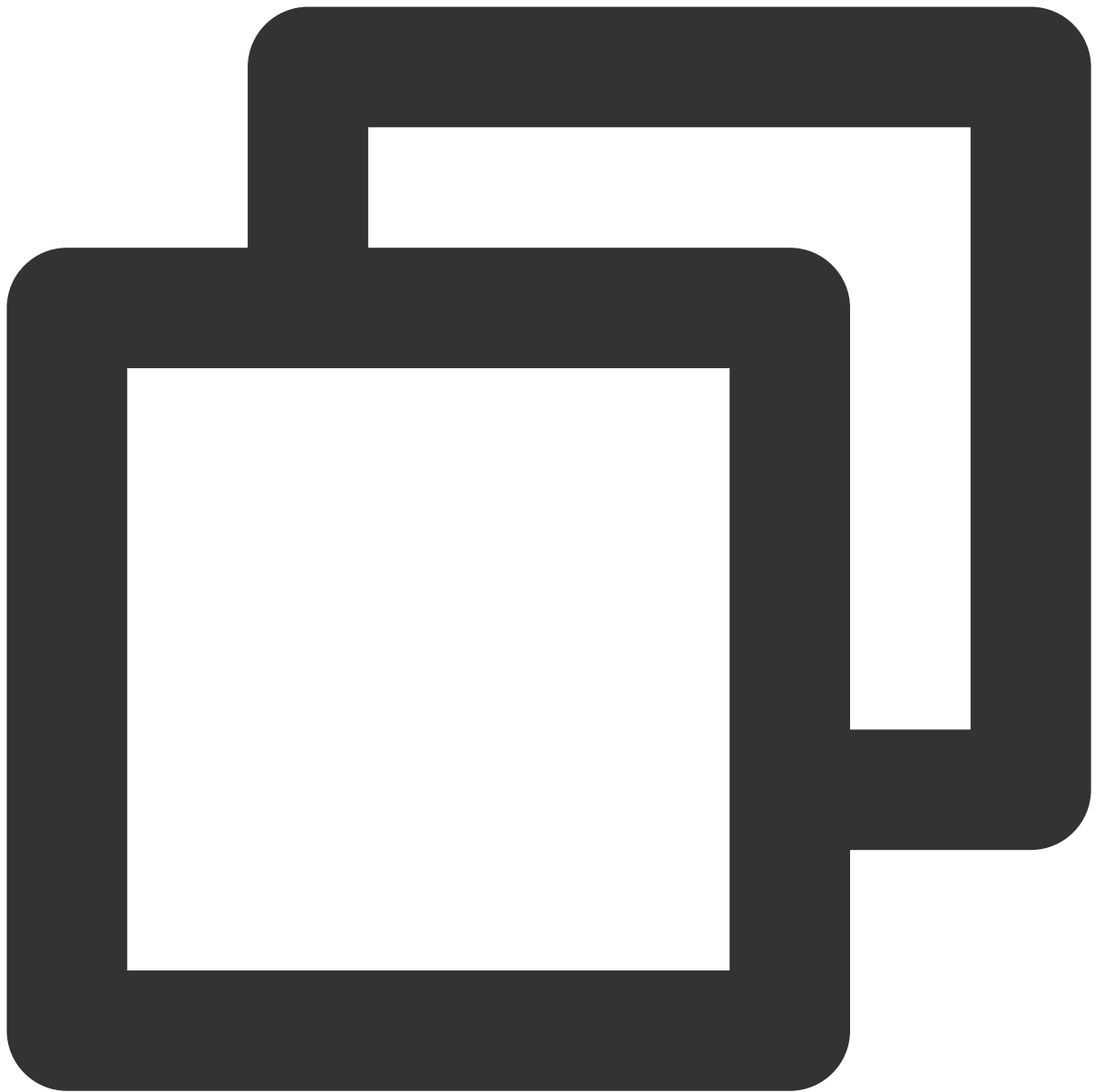
## Step 2. Import the component

Use CocoaPods to import the component as follows:

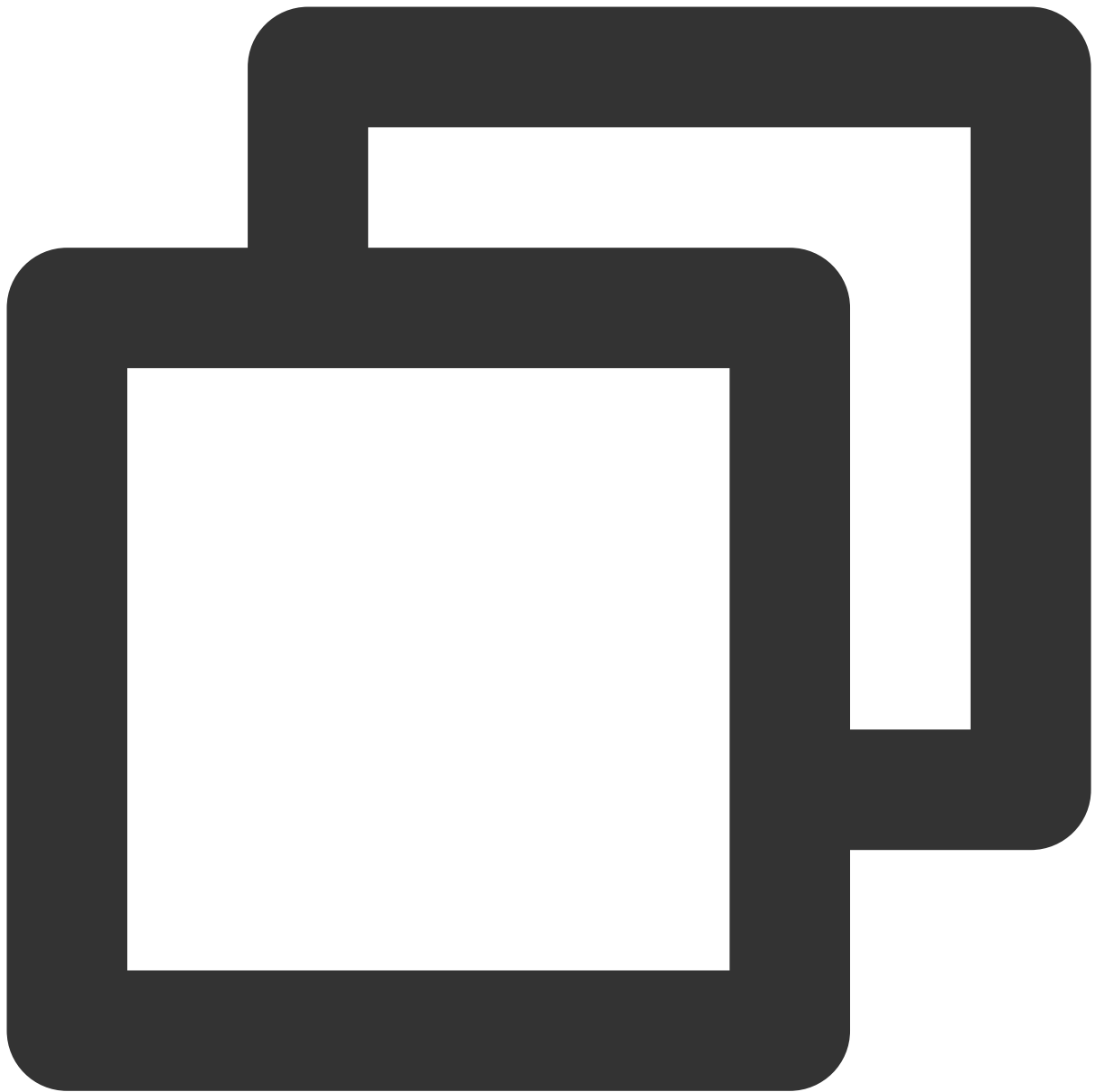
1. Add the following dependency to your `Podfile` .

Swift

Objective-C

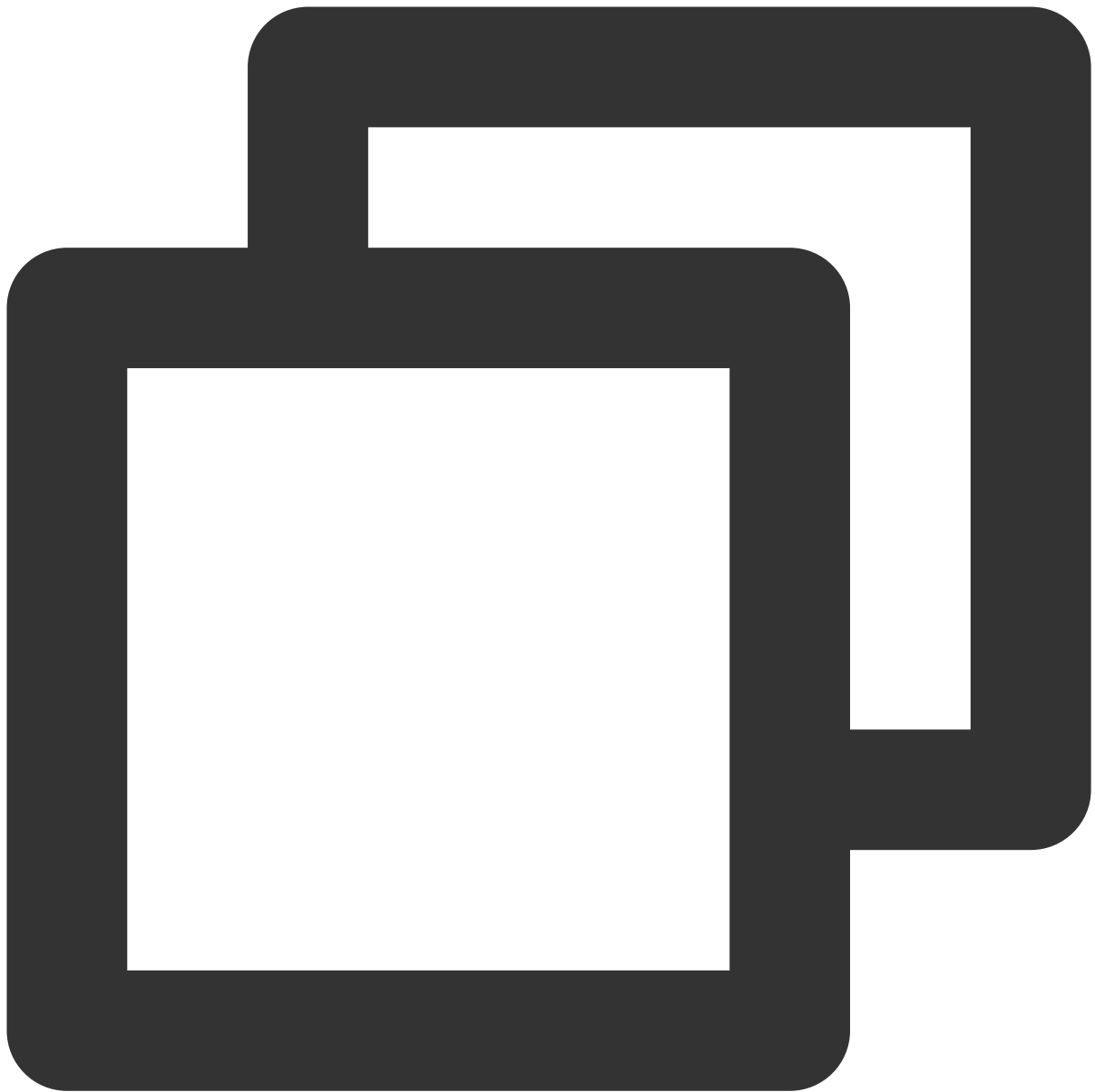


```
pod 'TUICallKit_Swift'
```



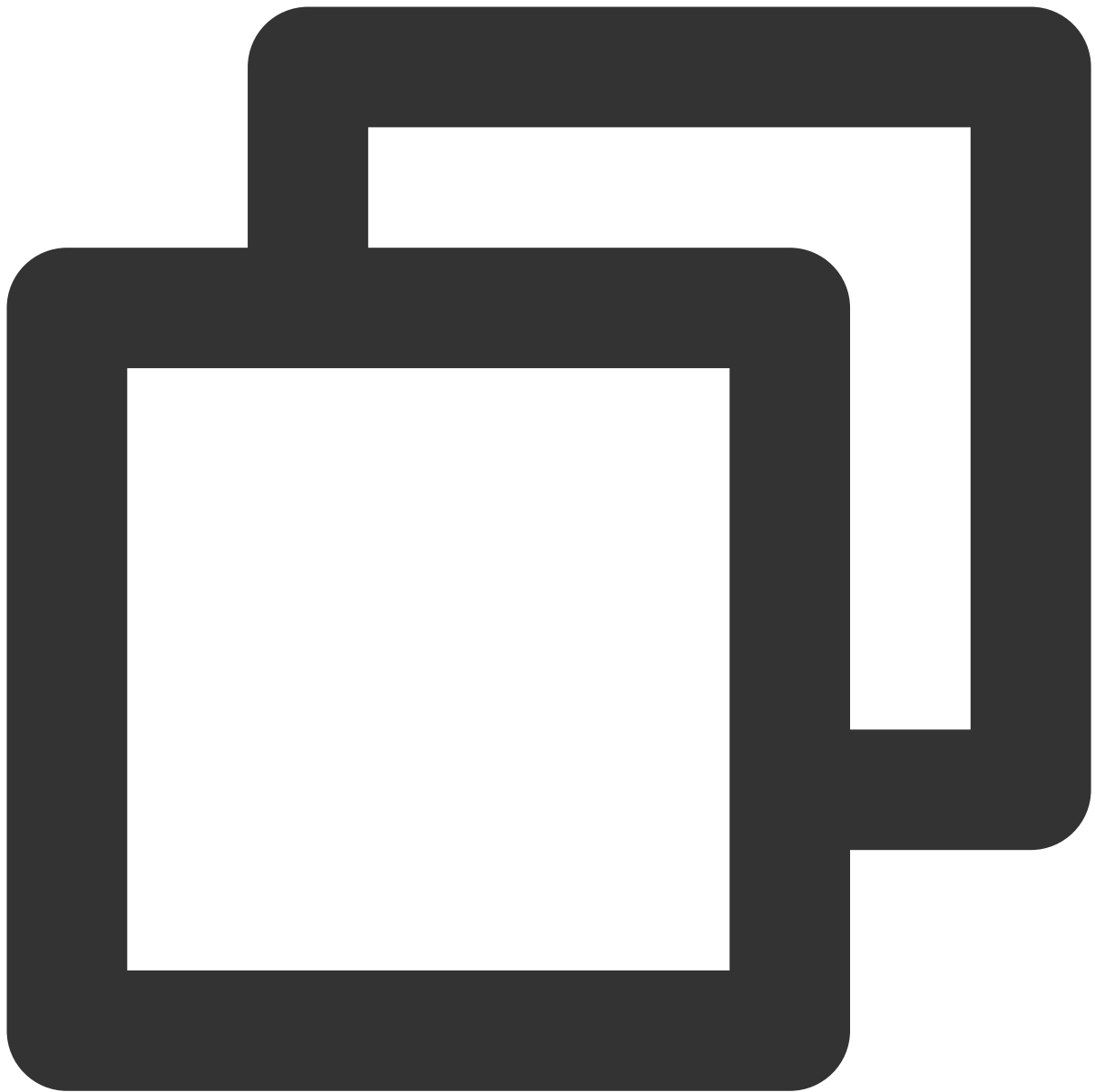
```
pod 'TUICallKit'
```

2. Run the following command to install the component:



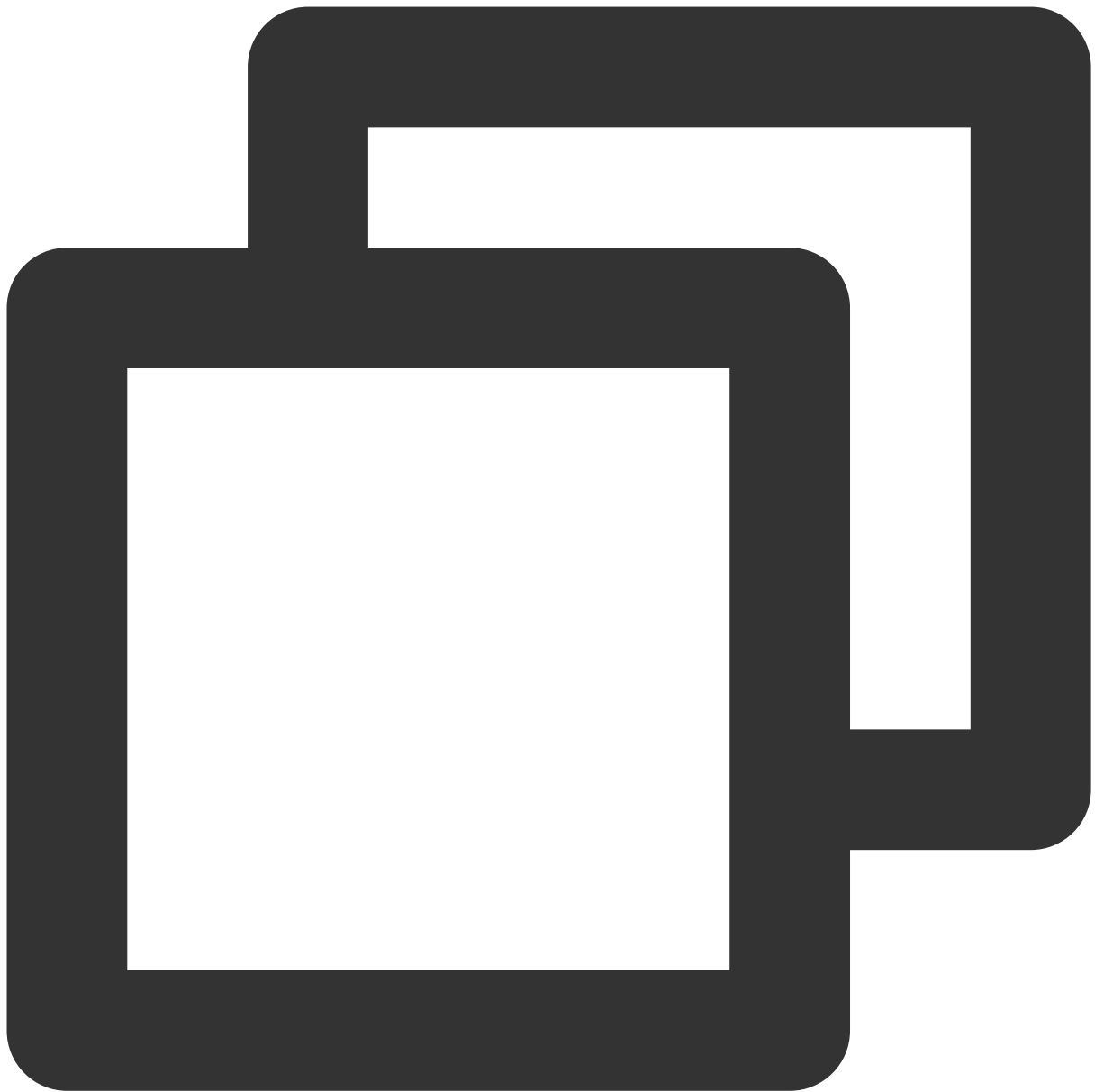
```
pod install
```

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



```
pod repo update
```

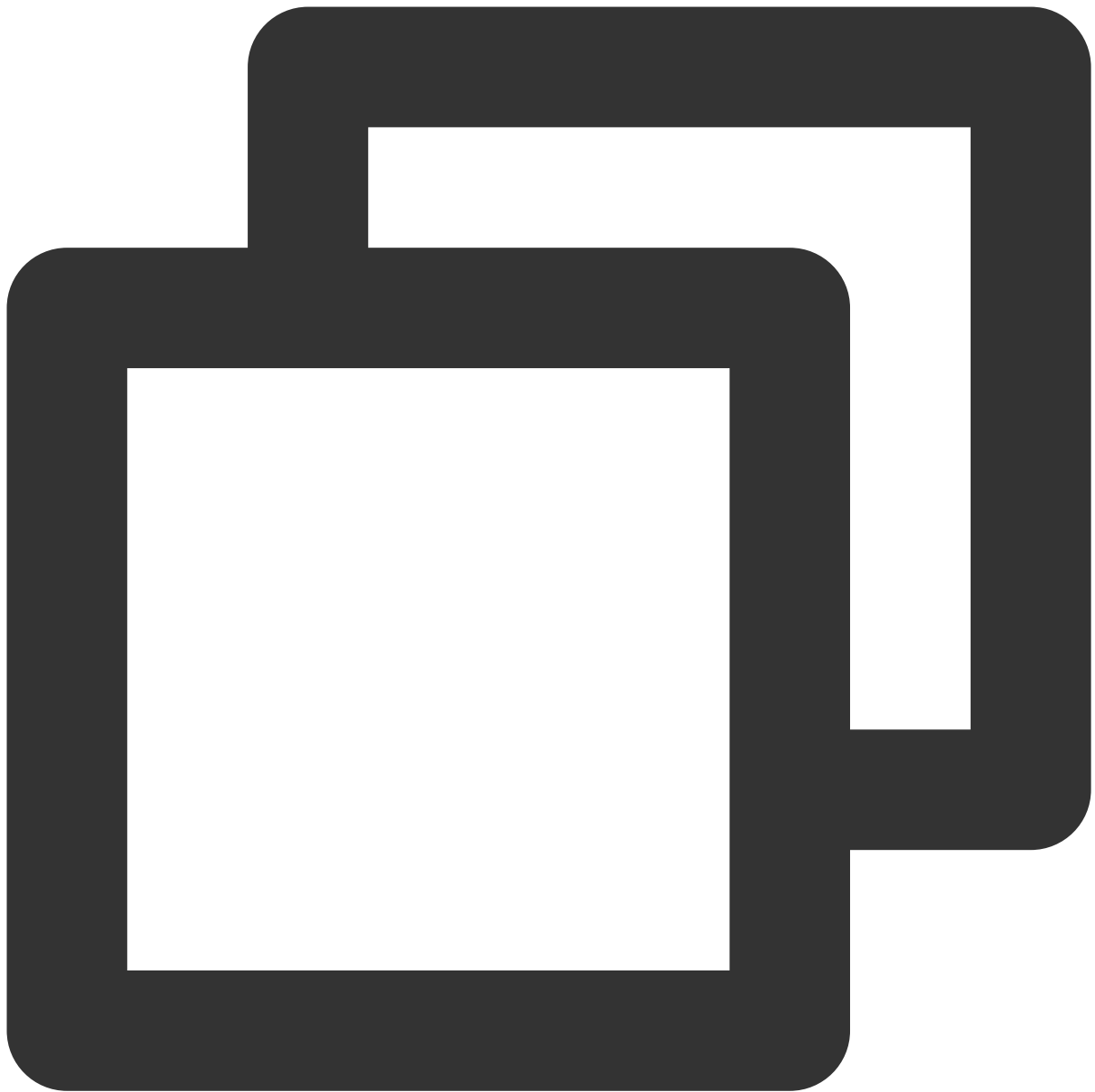
Afterwards, execute the following command to update the Pod version of the component library.



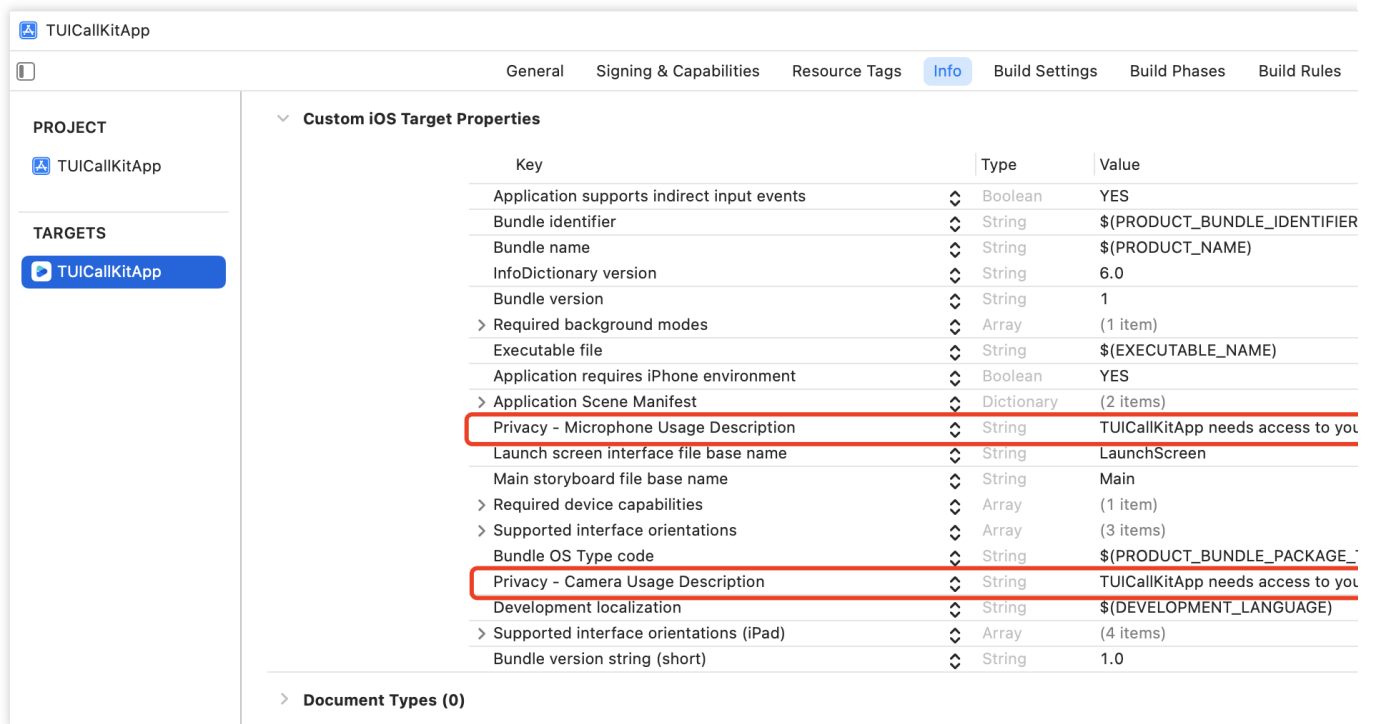
```
pod update
```

## 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>
```



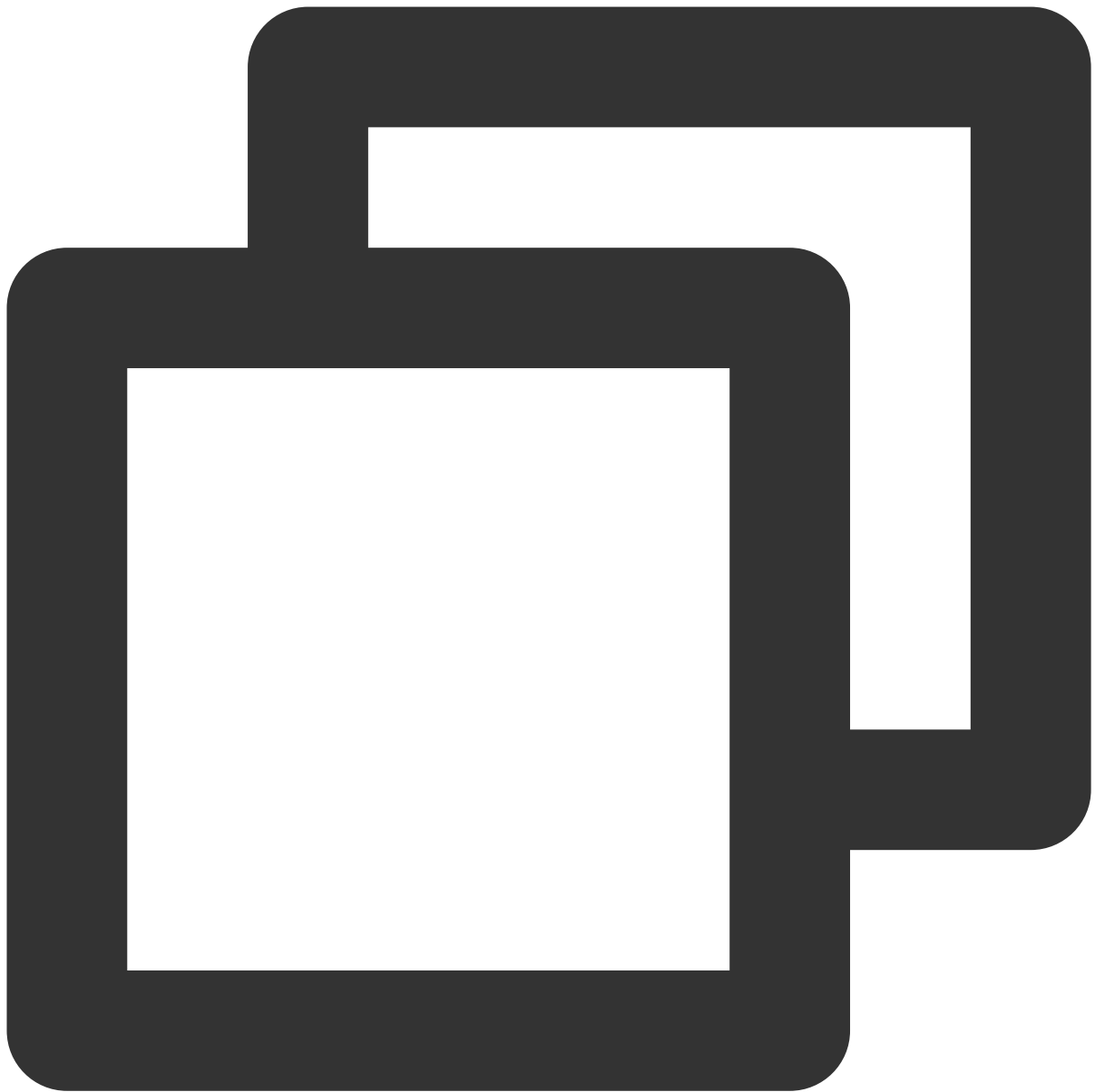
## Step 4. Log in to the `TUICallKit` component

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

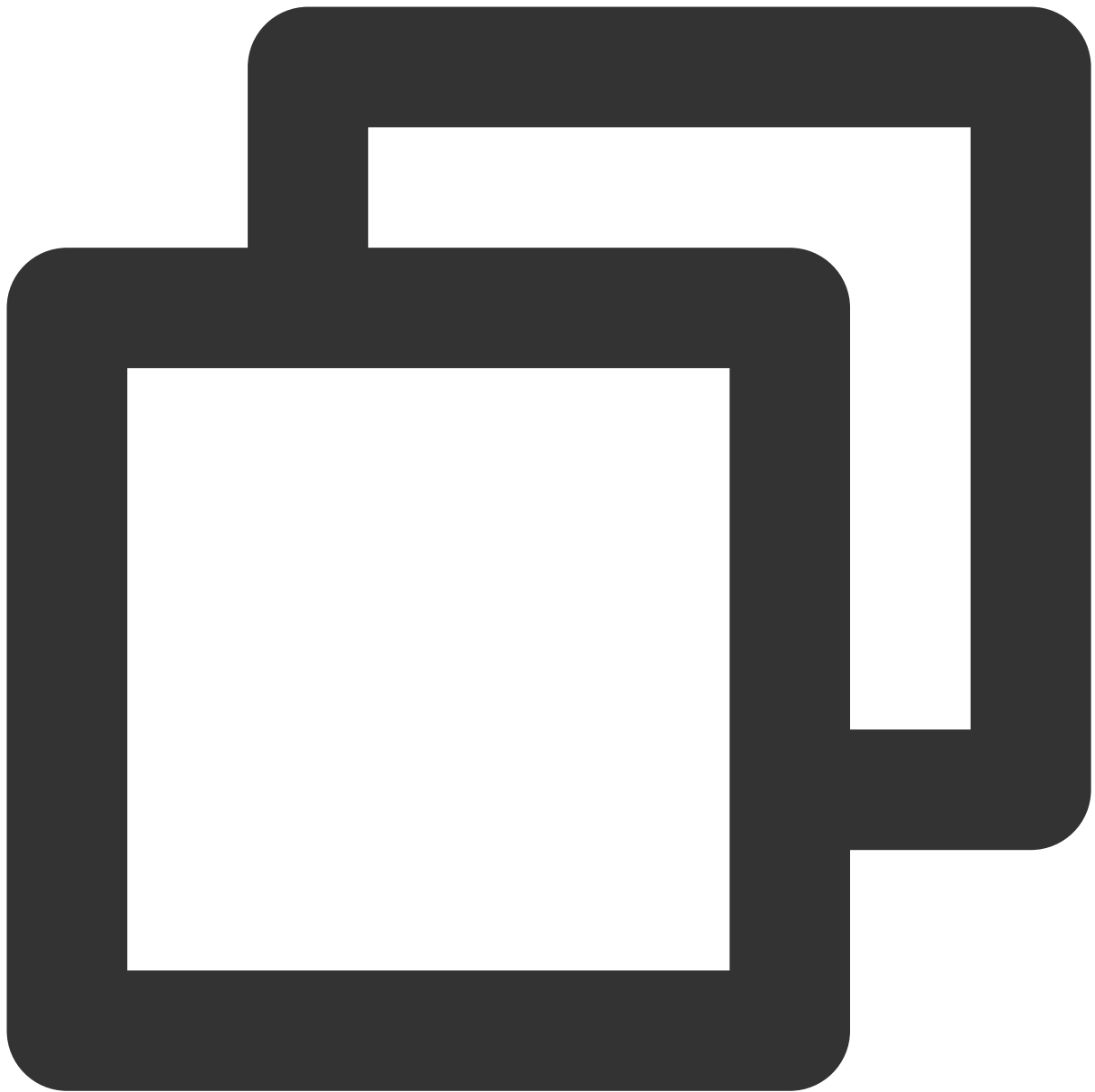
Objective-C





```
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:@"xxxxxxxxxxxx"         // 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.

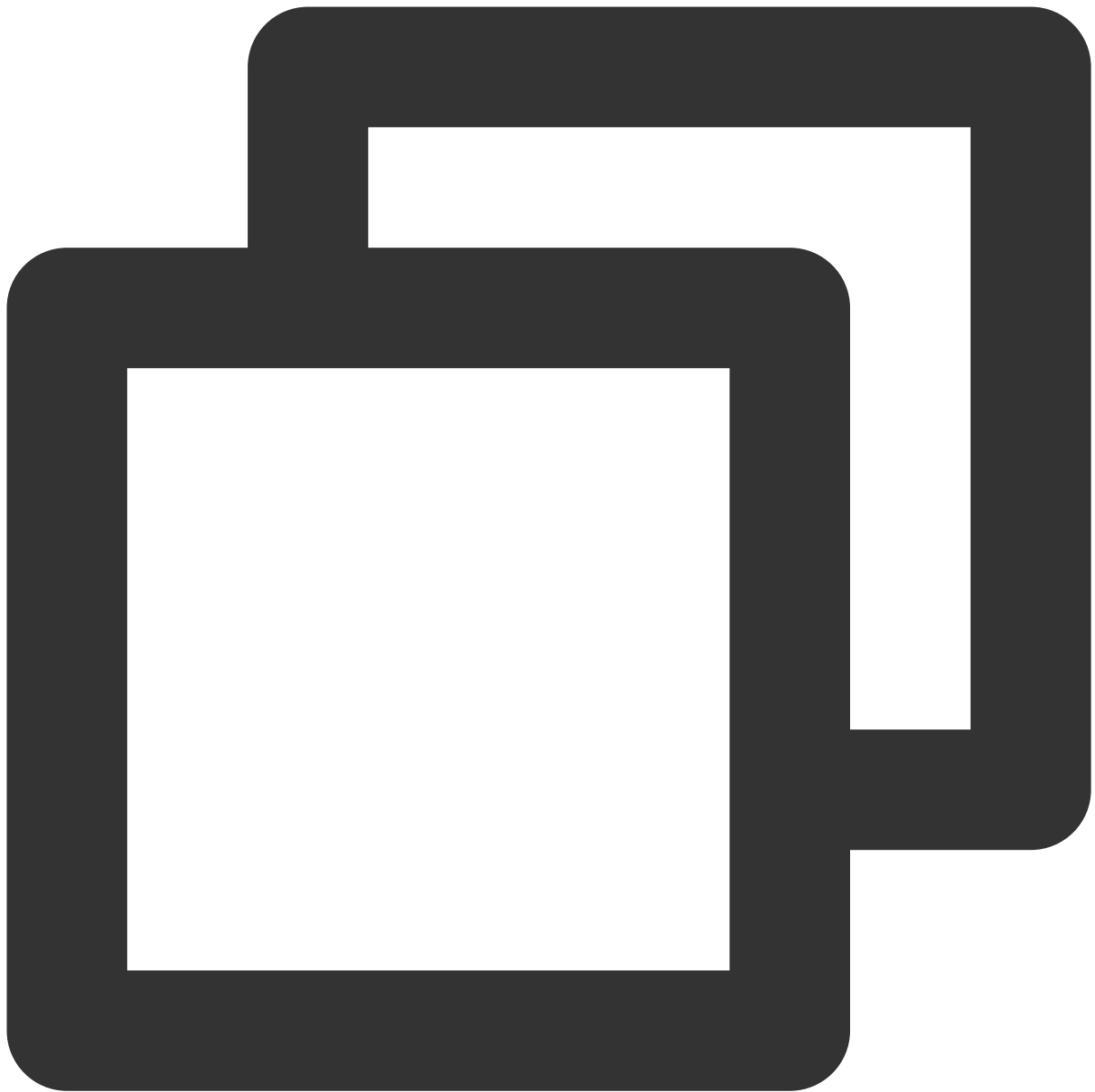
The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current integration process more quickly. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` 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: Setting Nickname and Avatar

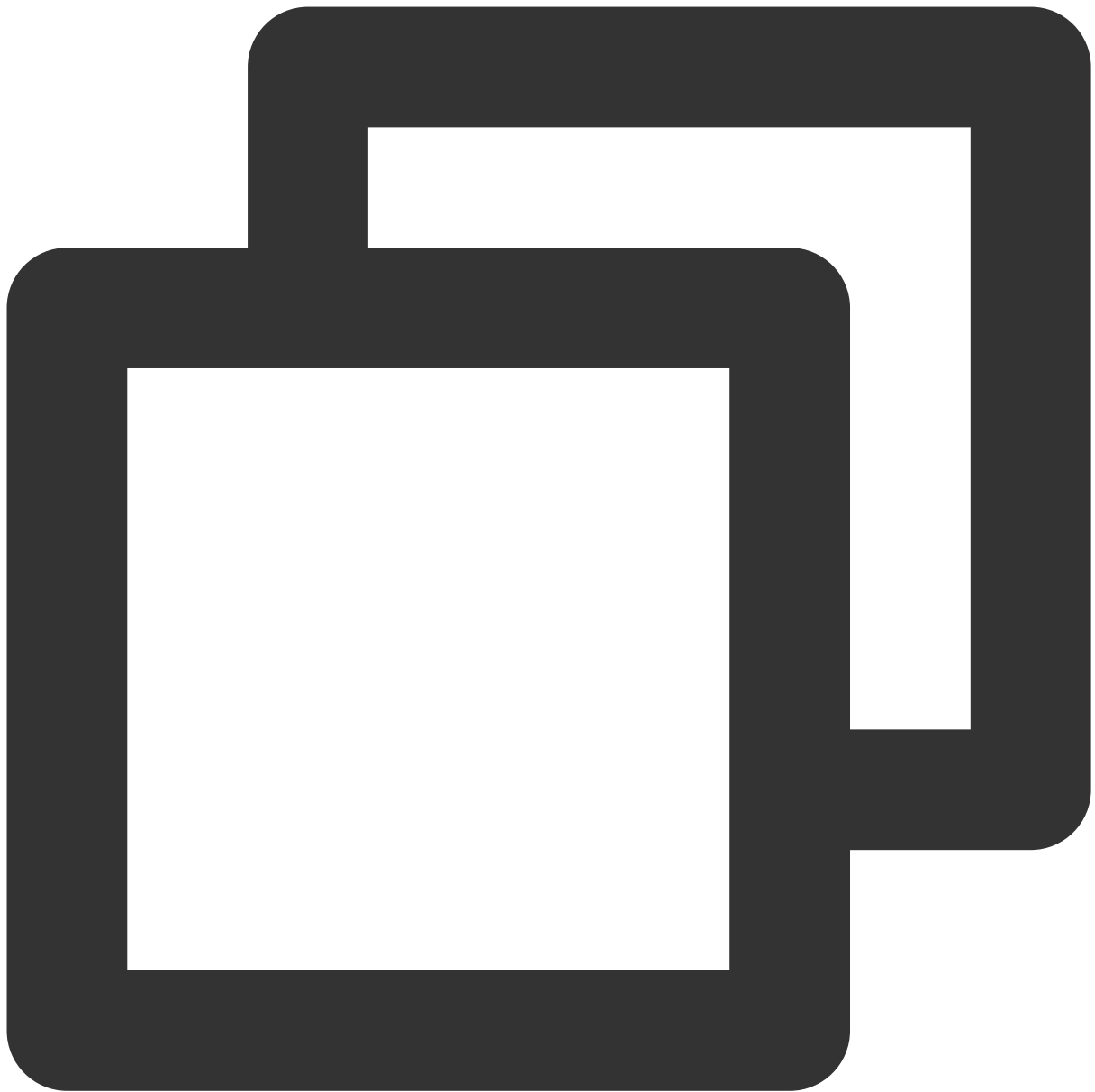
If you need to customize your nickname or avatar, you can update them using the following interface:

Swift

Objective-C



```
TUICallKit.createInstance().setSelfInfo(nickname: "nickname", avatar: "profile phot  
} fail: { code, message in  
}
```



```
[[TUICallKit sharedInstance] setSelfInfo:@"nickname" avatar:@"profile photo URL" su  
} fail:^(int code, NSString *errMsg) {  
  
}];
```

**Note:**

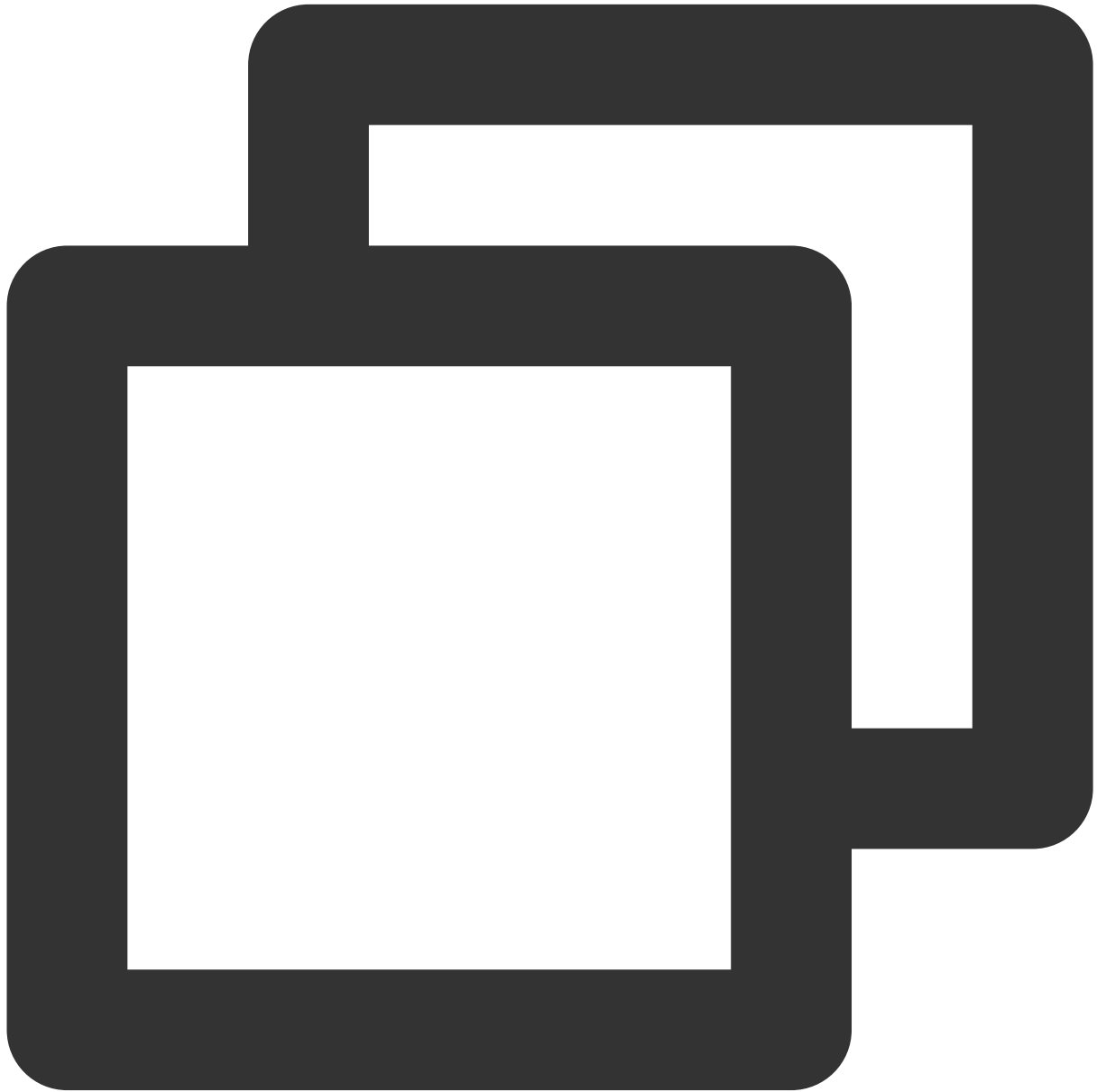
The update of the callee's nickname and profile photo may be delayed during a call between non-friend users due to the user privacy settings. After a call is made successfully, the information will also be updated properly in subsequent calls.

## Step 6: Make your First Phone Call

By calling the call function of TUICallKit and specifying the call type and the callee's userId, you can initiate a voice or video call.

Swift

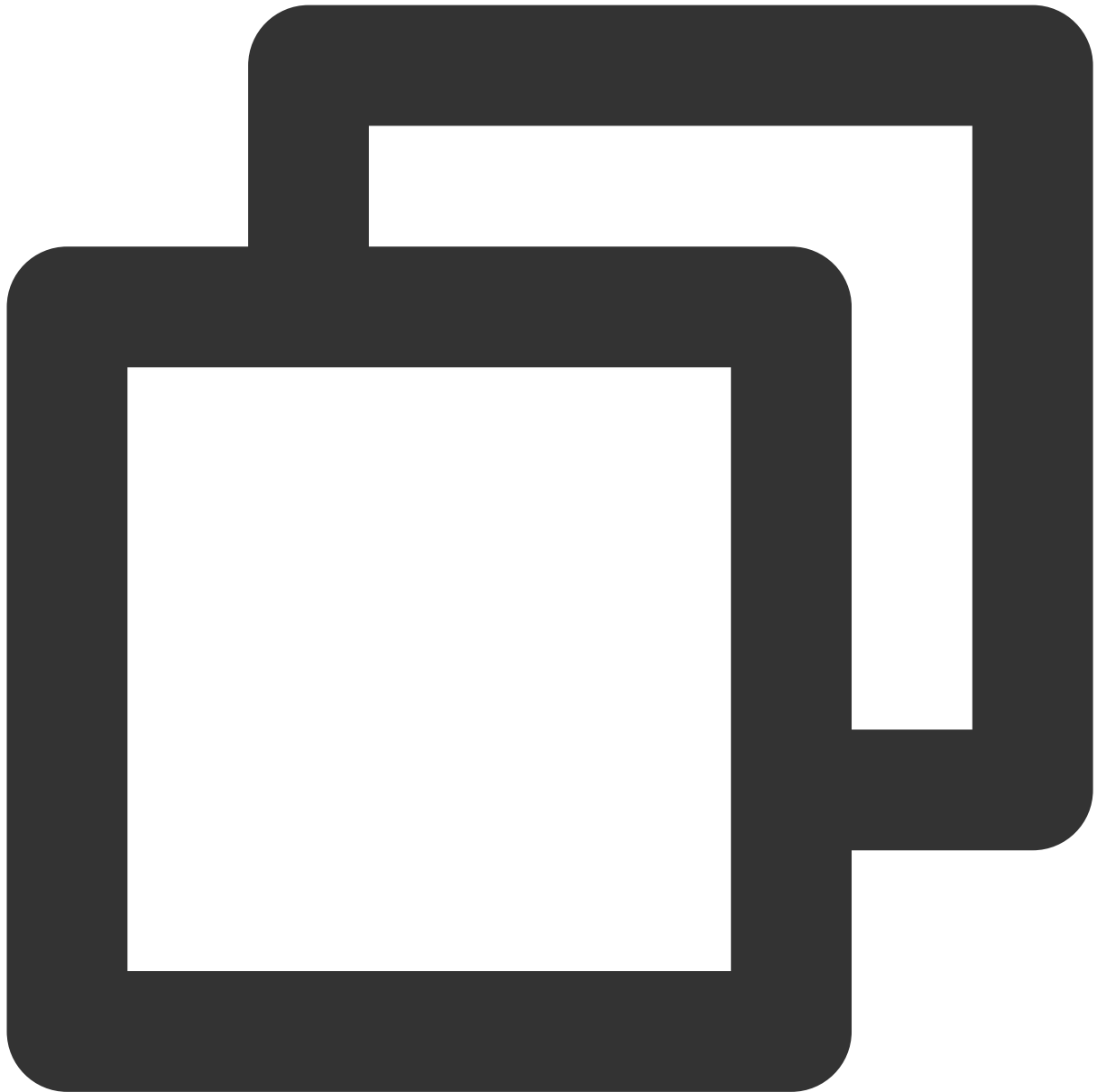
Objective-C



```
import TUICallKit

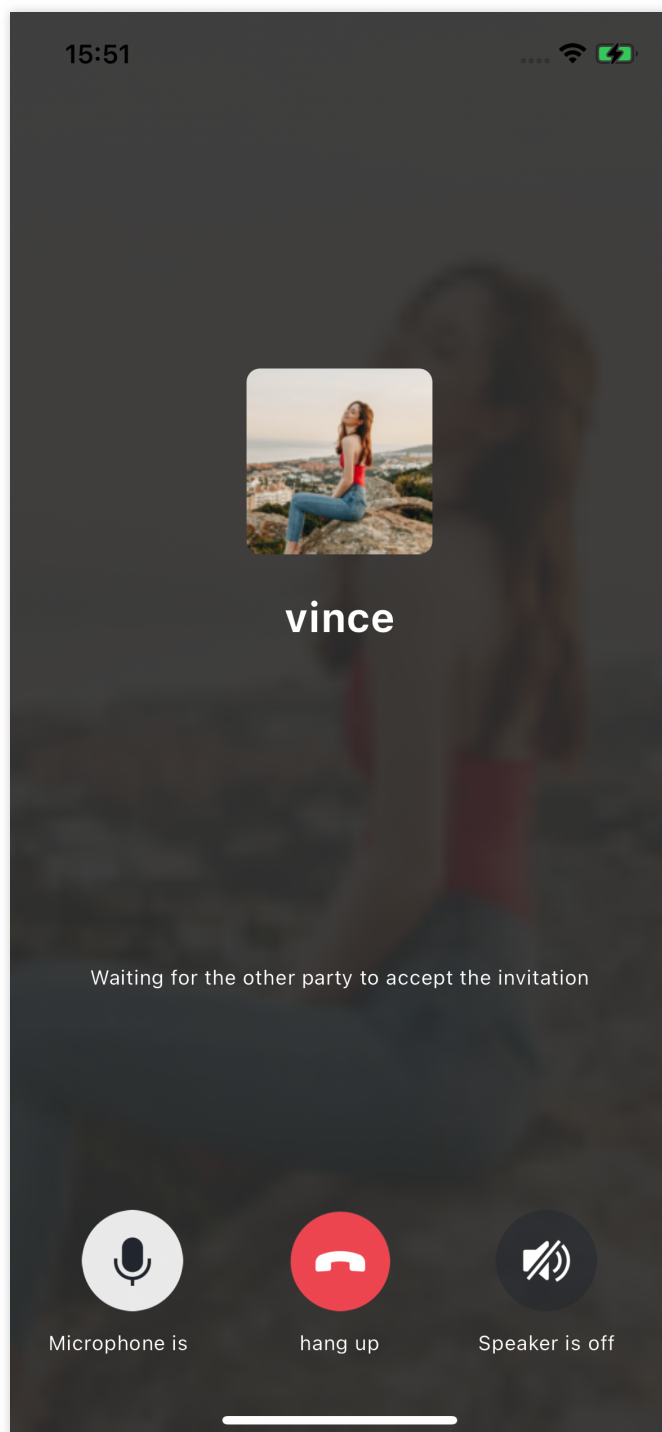
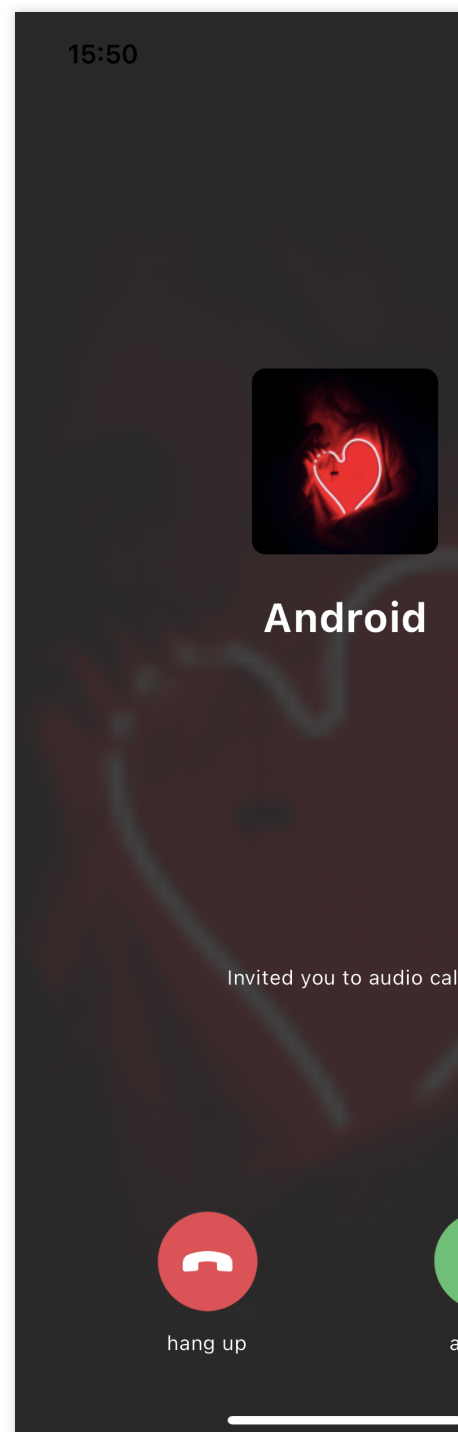
// Initiate a 1-on-1 video call (suppose the `userId` is `mike`)
```

```
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video)
```



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

// Initiate a 1-on-1 video call (suppose the `userId` is `mike`)
[[TUICallKit sharedInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```

CallerCallee

## Additional Features

[Customizable Interface](#)

[Group Call](#)



[Floating Window](#)  
[Custom Ringtone](#)  
[Monitor call status](#)  
[Cloud Recording](#)

## 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 ?

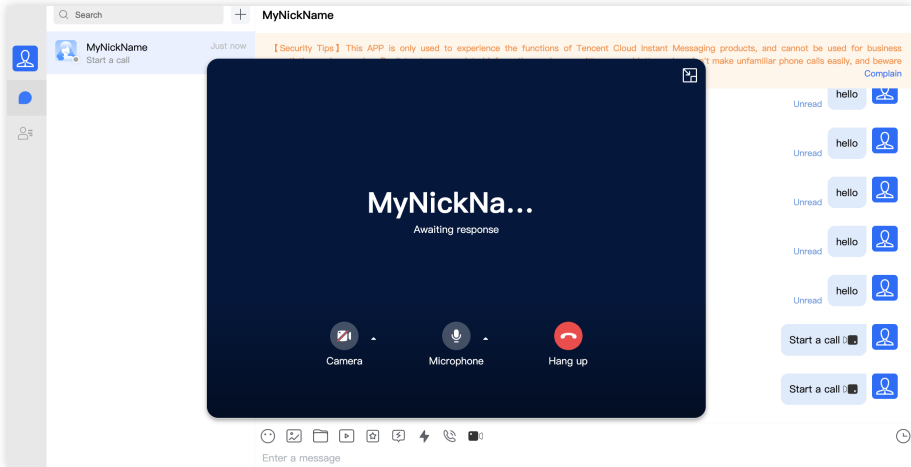
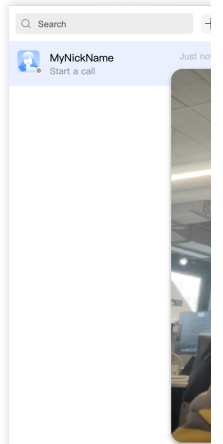
please refer to [Purchase Official Version](#).

## Suggestions and Feedback

If you have any suggestions or feedback, please contact [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Web&H5

Last updated : 2024-07-23 18:03:58

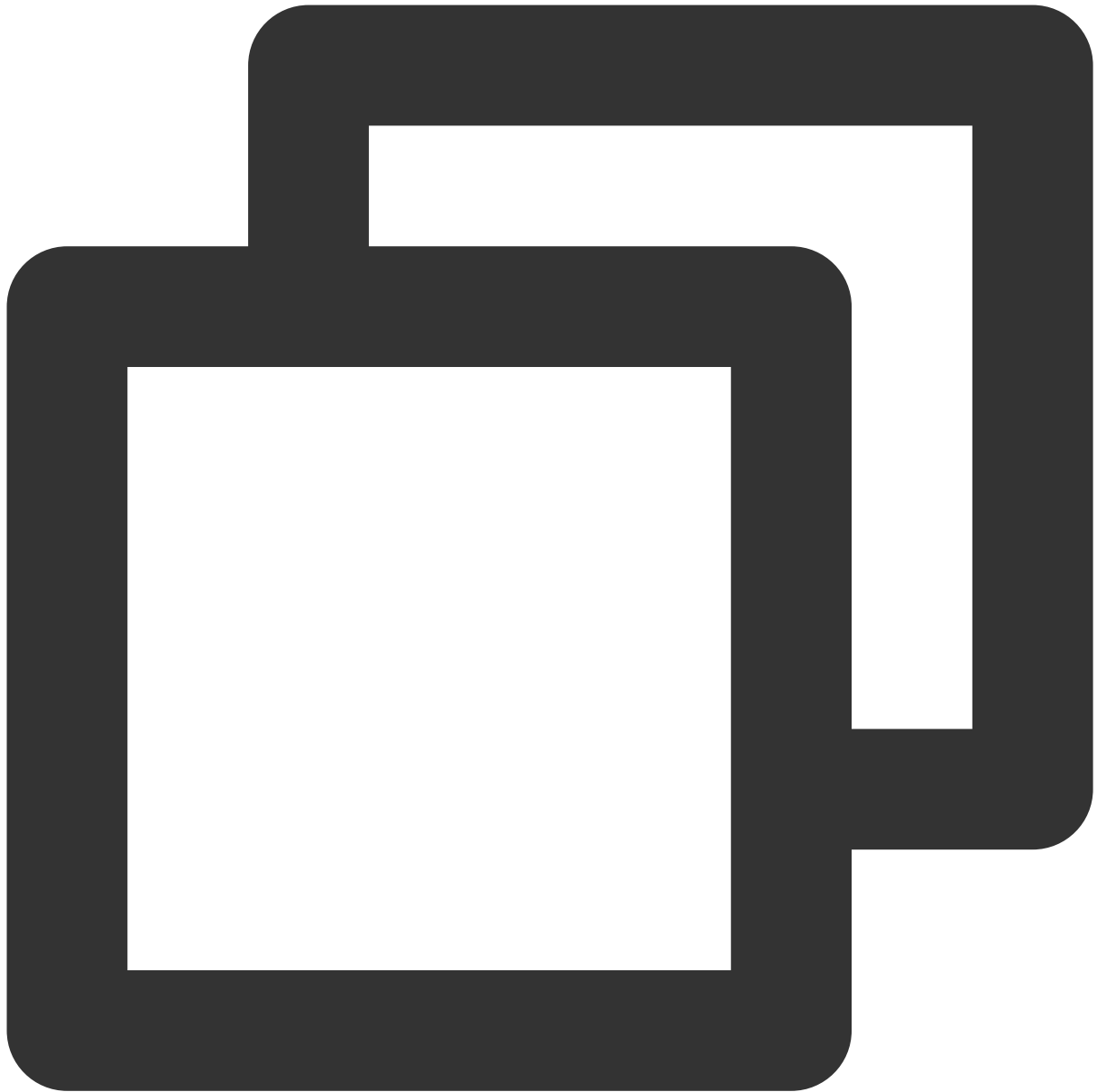
Video Call	Audio Call
	

## Step 1: Activate Audio and Video Calling Capability

Before using Tencent Cloud's audio and video services, you need to go to the Console and activate the audio and video service for your application. For specific steps, please refer to [Activate the Service](#).

## Step 2: Download the TUICallKit Component

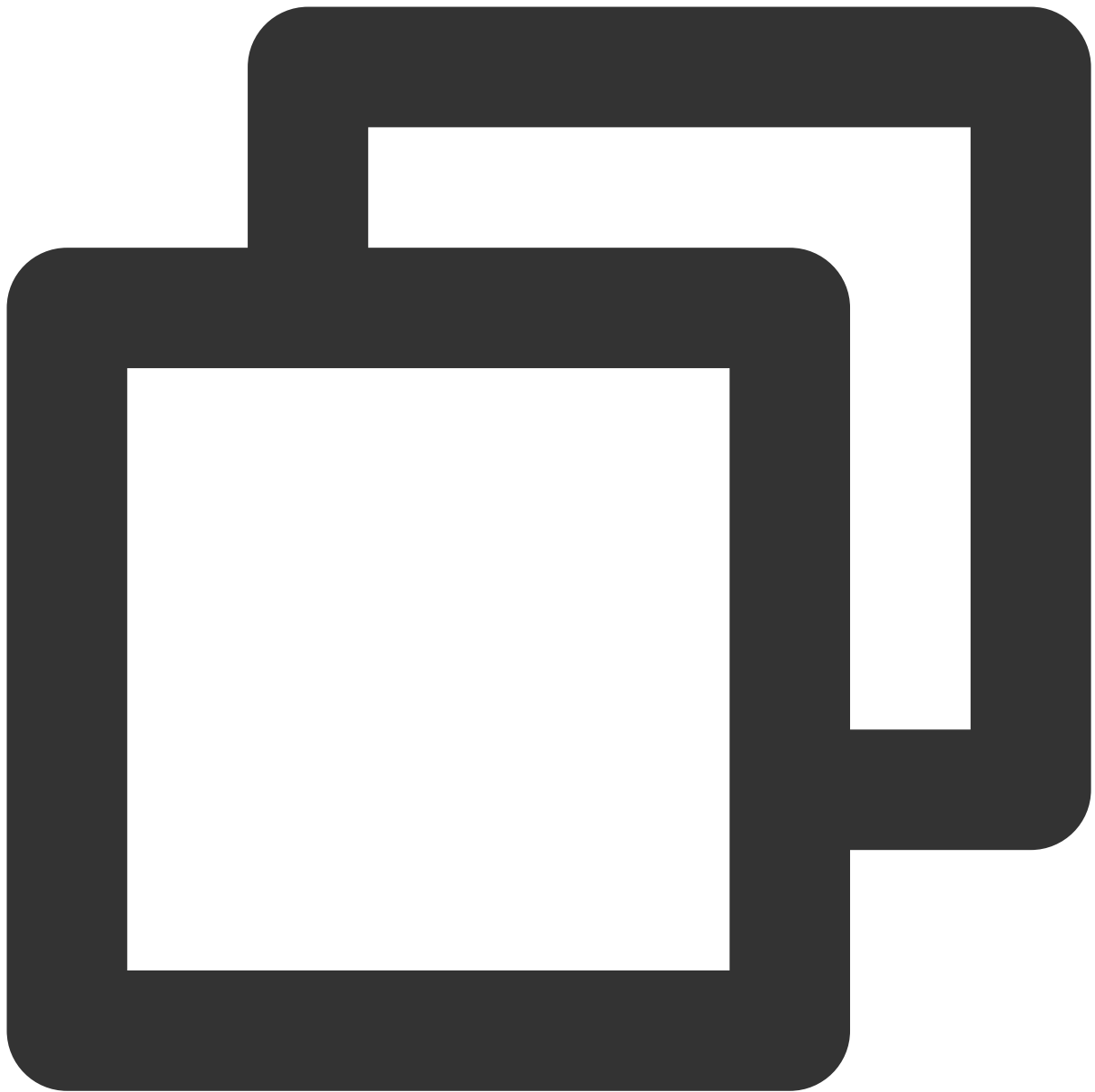
Download the TUICallKit component via [npm](#):



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

## Step 3: Import the TUICallKit Component

In `main.ts`, you only need to add two lines of code to try the call feature.  
Import `TUICallKit` and mount it to `TUIKit`.



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

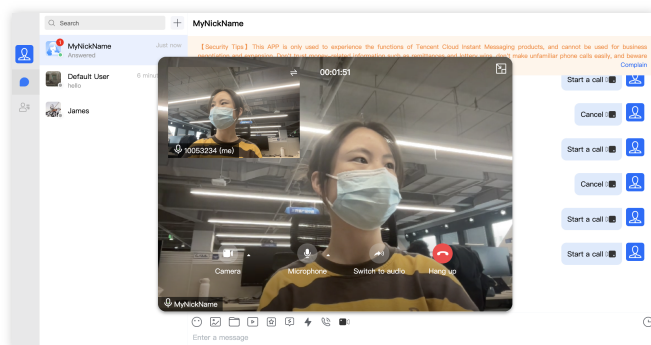
## Step 4: Invoke the TUICallKit Component

### Note:

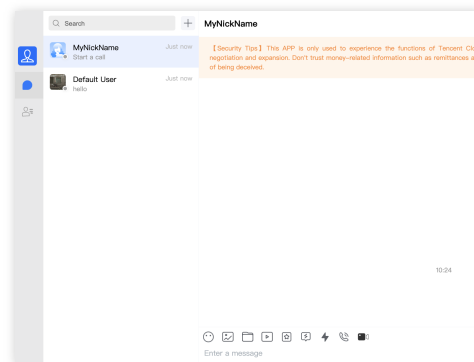
The TUICallKit component needs to be placed inside a dom node to display and control its position, width, height, and other styles.

On the page where you want to display it, simply call the TUICallKit component for use. The main components of TUICallKit include:

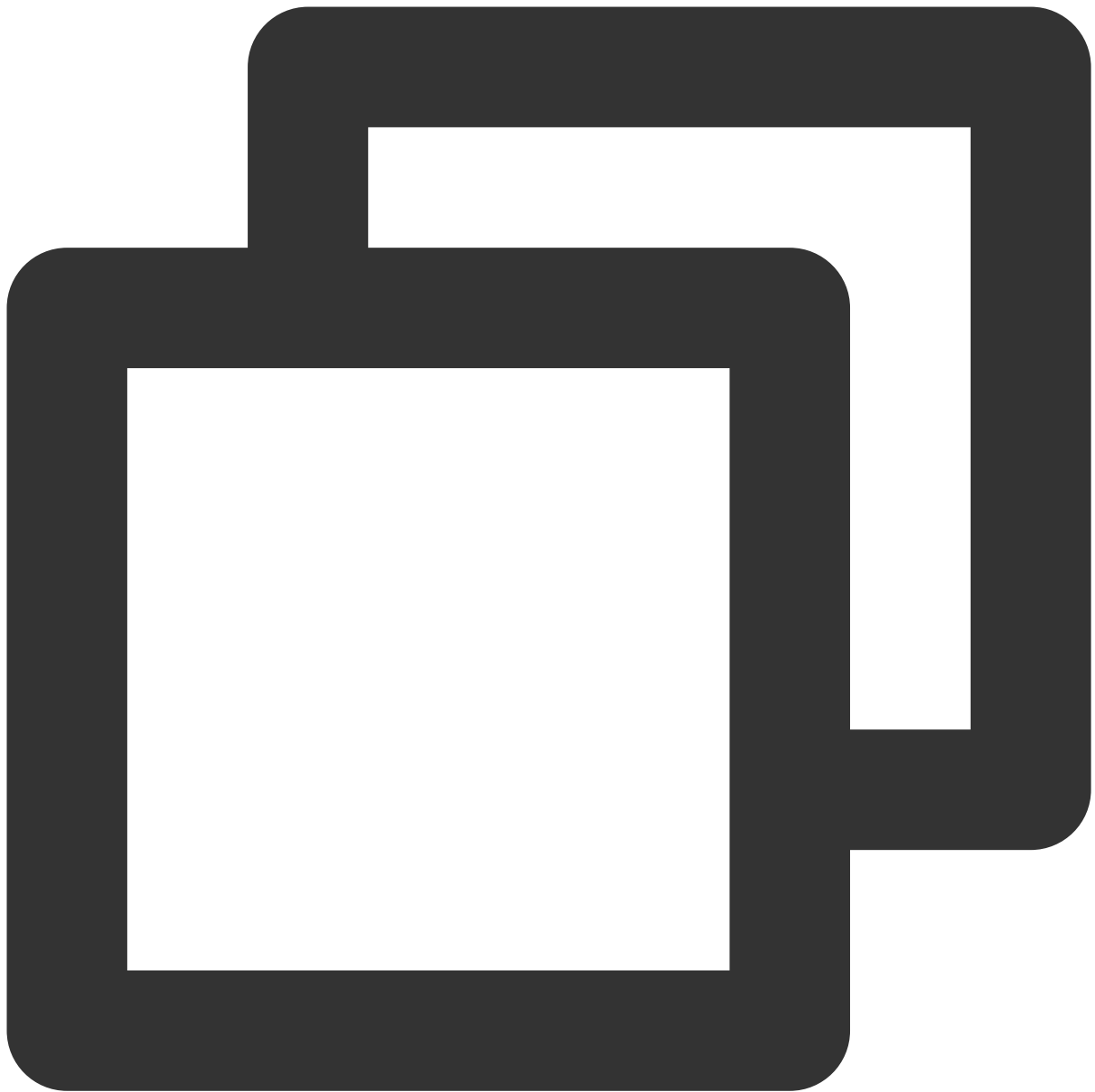
`<TUICallKit/>` : Call UI component body



`<TUICallKitMini/>` : Call UI floating view providing a minimize feature



For example: On the App.vue page, on top of the chat interface already set up, use TUICallKit to quickly build a call interface.



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

```
<h1>Welcome to Tencent Cloud Instant Messaging (IM)</h1>
</TUIChat>
</div>
<!-- TUICallKit Component: Call UI Component Body -->
<TUICallKit
  :class="!showCallMini ? 'callkit-drag-container' : 'callkit-drag-container-mi
  :allowedMinimized="true"
  :allowedFullScreen="false"
  :beforeCalling="beforeCalling"
  :afterCalling="afterCalling"
  :onMinimized="onMinimized"
  :onMessageSentByMe="onMessageSentByMe"
/>
</div>
</template>

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

export default defineComponent({
  name: "App",
  setup() {
    const data = reactive({
      env: TUIEnv(),
      currentModel: "conversation",
      showCall: false,
      showCallMini: false,
    });
    const TUIServer = (window as any)?.TUIKitTUICore?.TUIServer;
    const handleCurrentConversation = (value: string) => {
      data.currentModel = value ? "message" : "conversation";
    };
    // beforeCalling: Execute before making a call and before receiving a call invi
    const beforeCalling = (type: string, error: any) => {
      if (error) {
        handleErrorPrompts(error, type);
        return;
      }
      data.showCall = true;
    };
    // afterCalling: Execute after ending the call
    const afterCalling = () => {
      data.showCall = false;
      data.showCallMini = false;
    };
  },
});
```

```
// onMinimized: Execute when the component switches to minimized state
const onMinimized = (
  oldMinimizedStatus: boolean,
  newMinimizedStatus: boolean
) => {
  data.showCall = !newMinimizedStatus;
  data.showCallMini = newMinimizedStatus;
};

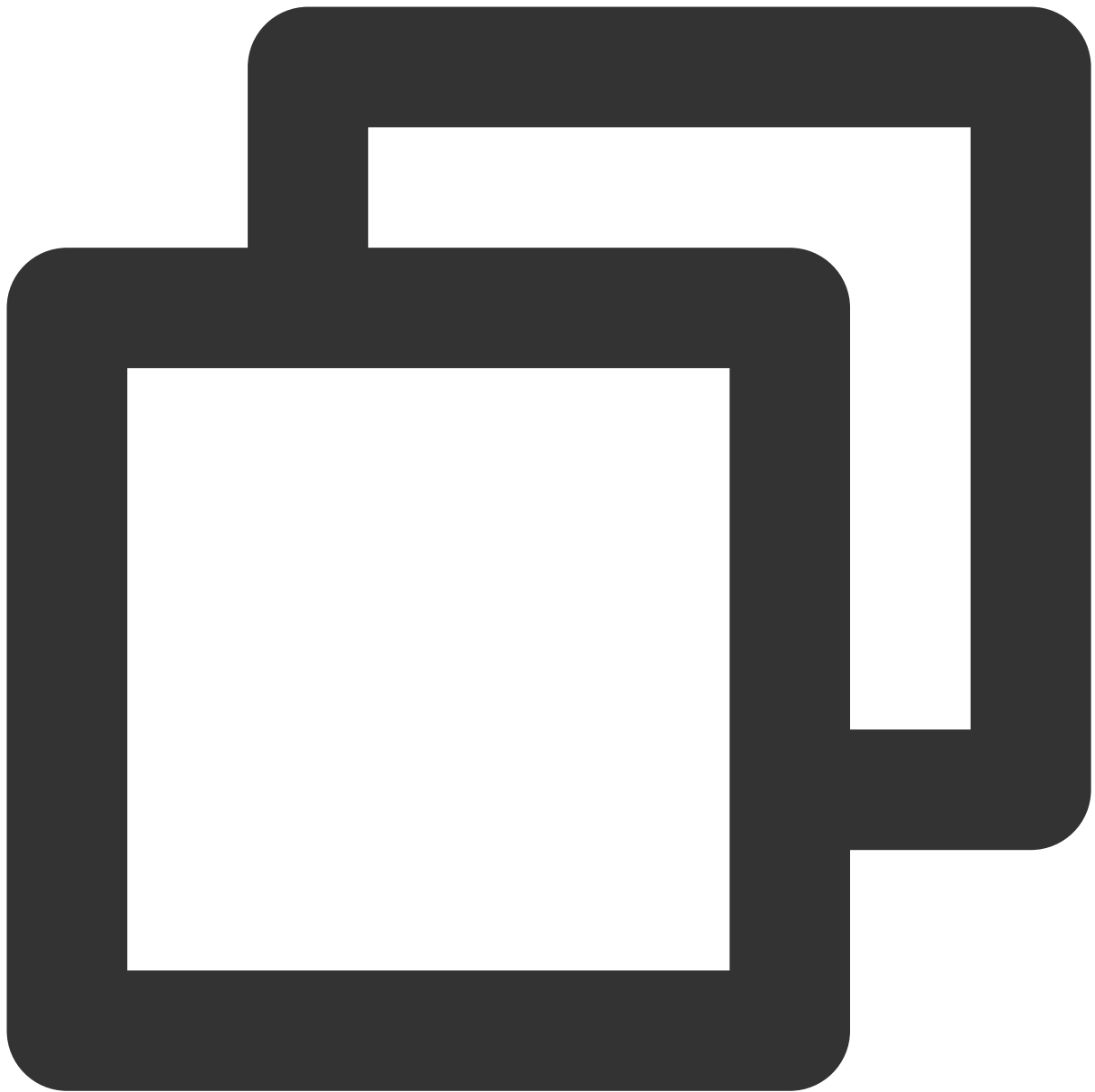
// onMessageSentByMe: Execute when sending messages during the entire call
const onMessageSentByMe = async (message: any) => {
  TUIServer?.TUIChat?.handleMessageSentByMeToView(message);
  return;
};

return {
  ...toRefs(data),
  handleCurrentConversation,
  beforeCalling,
  afterCalling,
  onMinimized,
  onMessageSentByMe,
};
},
});
</script>
<style scoped>
.home-TUIKit-main {
  display: flex;
  height: 100vh;
  overflow: hidden;
}
.search {
  padding: 12px;
}
.conversation {
  min-width: 285px;
  flex: 0 0 24%;
  border-right: 1px solid #f4f5f9;
}
.conversation-h5 {
  flex: 1;
  border-right: 1px solid #f4f5f9;
}
.chat {
  flex: 1;
  height: 100%;
  position: relative;
}
```



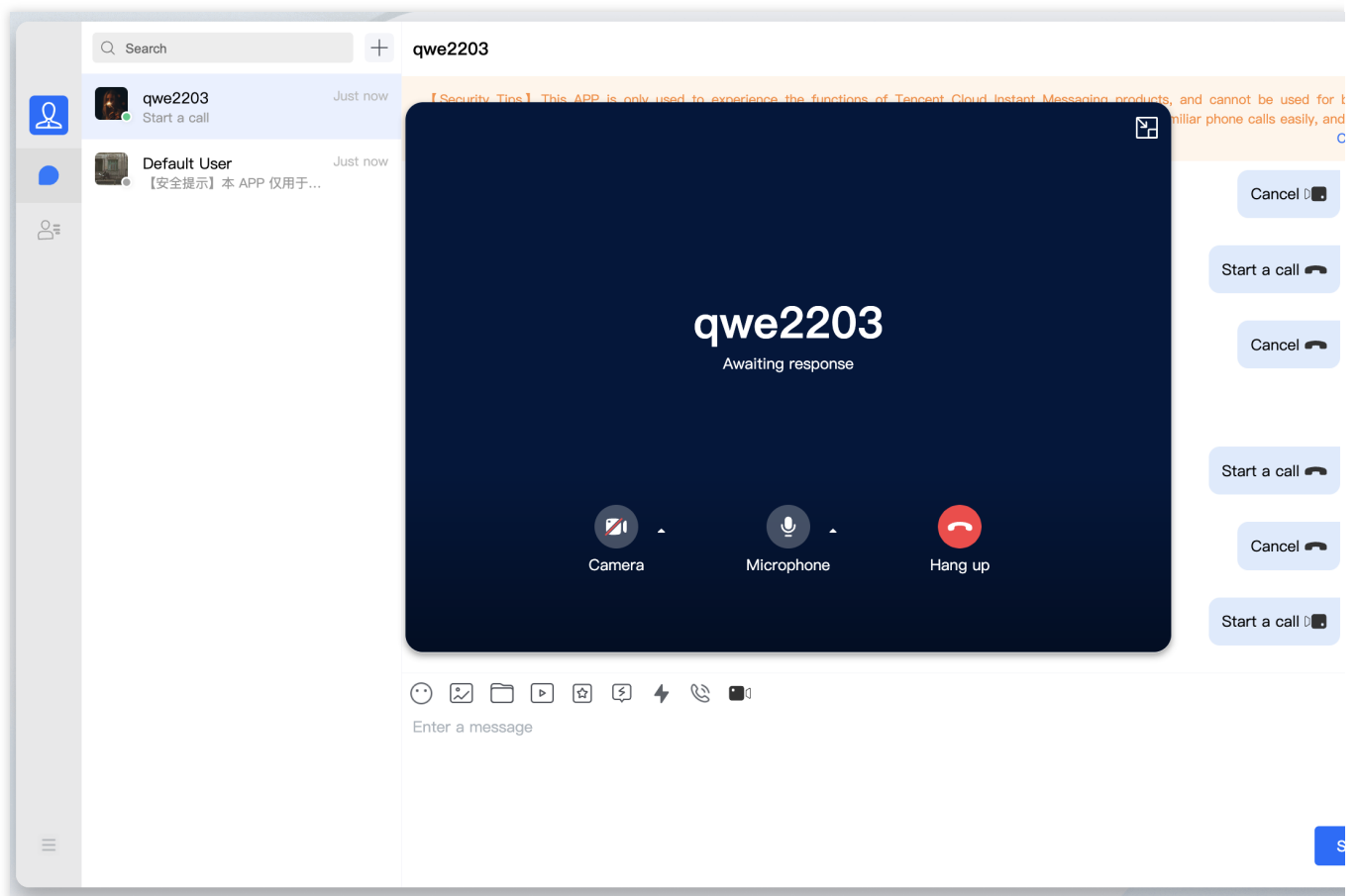
```
.callkit-drag-container {  
  position: fixed;  
  left: calc(50% - 25rem);  
  top: calc(50% - 18rem);  
  width: 50rem;  
  height: 36rem;  
  border-radius: 16px;  
  box-shadow: rgba(0, 0, 0, 0.16) 0px 3px 6px, rgba(0, 0, 0, 0.23) 0px 3px 6px;  
}  
.callkit-drag-container-mini {  
  position: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

## Step 5. Launch the Project



```
npm run serve
```

## Step 6: Make Your First Call



## Technical Support

If you have any requirements or feedback, contact [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Flutter

Last updated : 2024-05-15 17:02:11

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

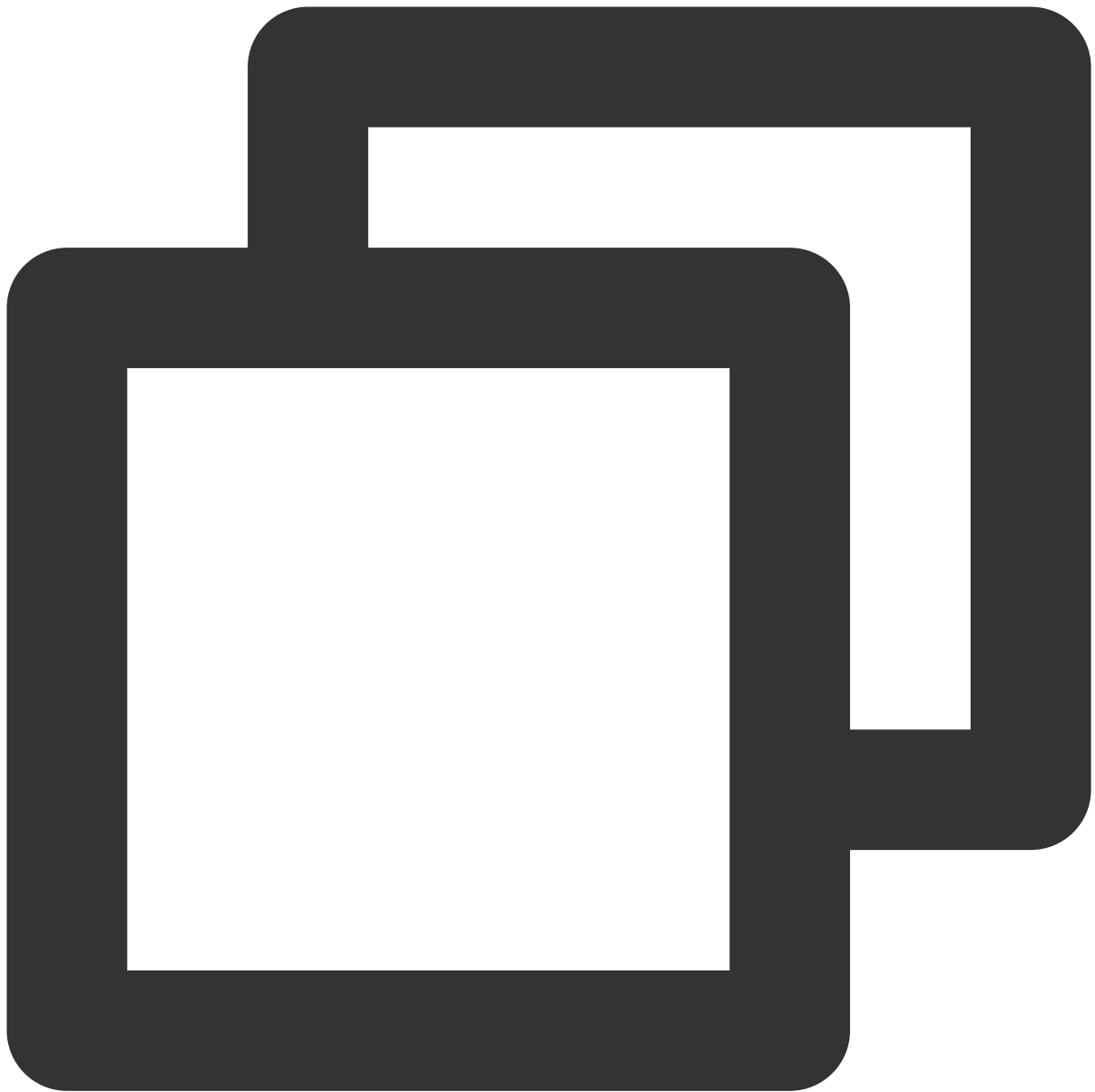
Flutter 3.0 or higher version.

### Step 1. Activate the Service

Before using Tencent Cloud's audio and video services, you need to go to the Console and activate the audio and video service for your application. For specific steps, please refer to [Activate the Service](#).

### Step 2. Import the Component

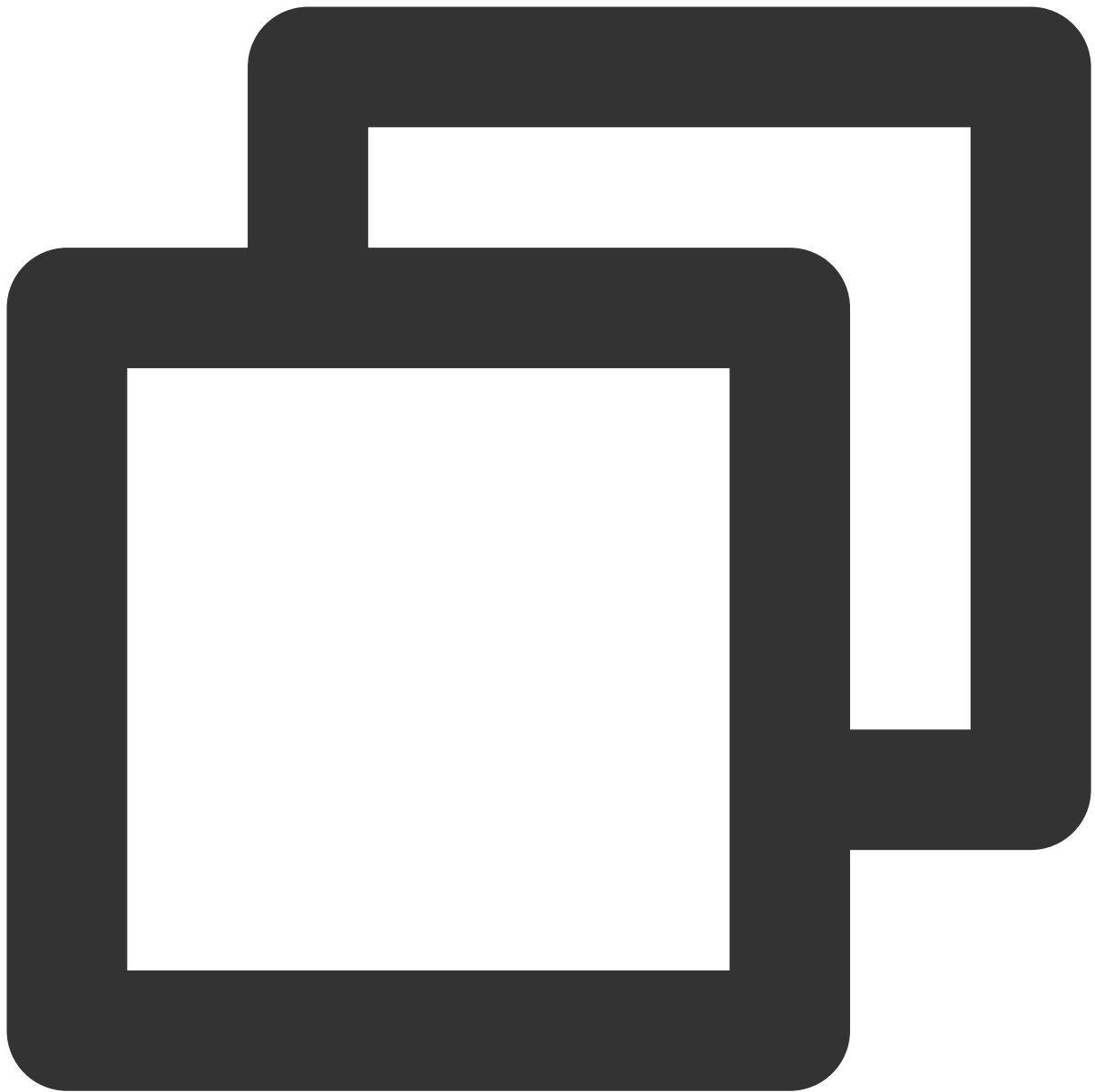
Execute the following command in the command line to install the component `tencent_calls_uikit` plugin.



```
flutter pub add tencent_calls_uikit
```

## Step 3. Configure the Project

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

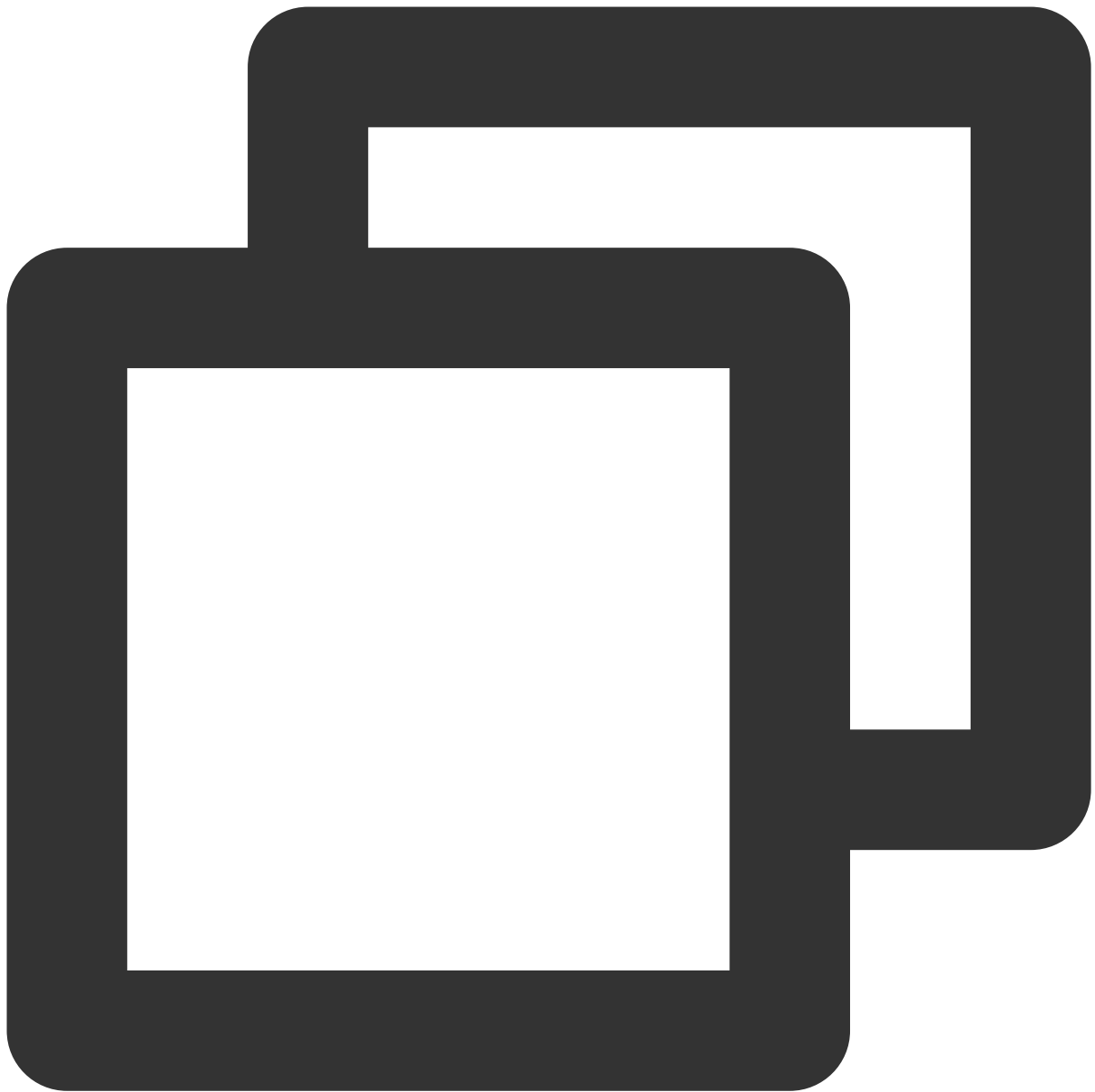


```
import 'package:tencent_calls_uikit/tuicall_kit.dart';

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

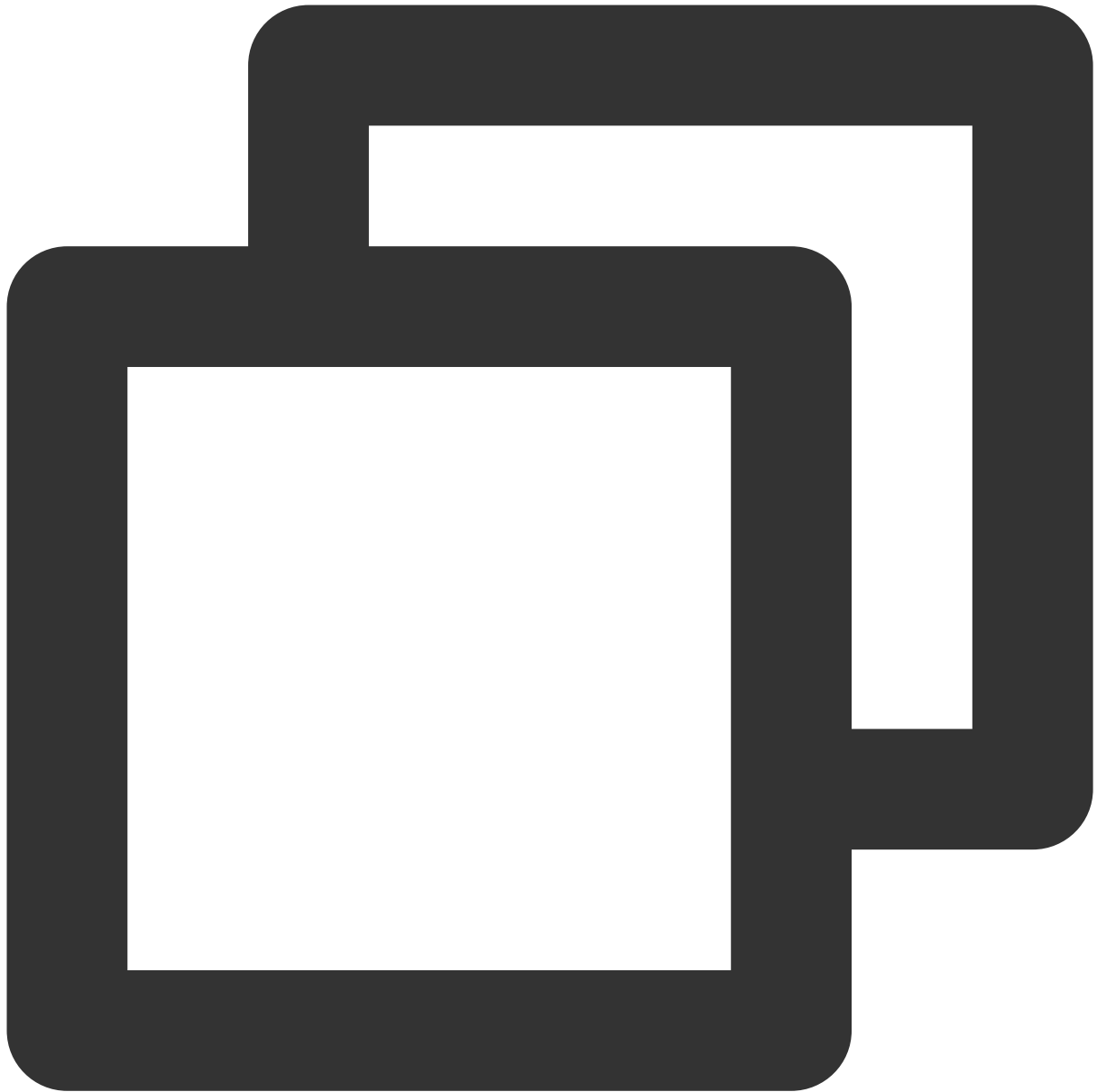
2. If you need to compile and run codes on the Android platform, since the SDK uses Java's reflection feature internally, certain classes in the SDK must be added to the non-aliasing list.

First, you need to configure in the build.gradle file under the app directory and enable aliasing rules:



```
android {  
    ...  
    buildTypes {  
        release {  
            minifyEnabled true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard  
        }  
    }  
}
```

Then add the following code in the proguard-rules.pro file:

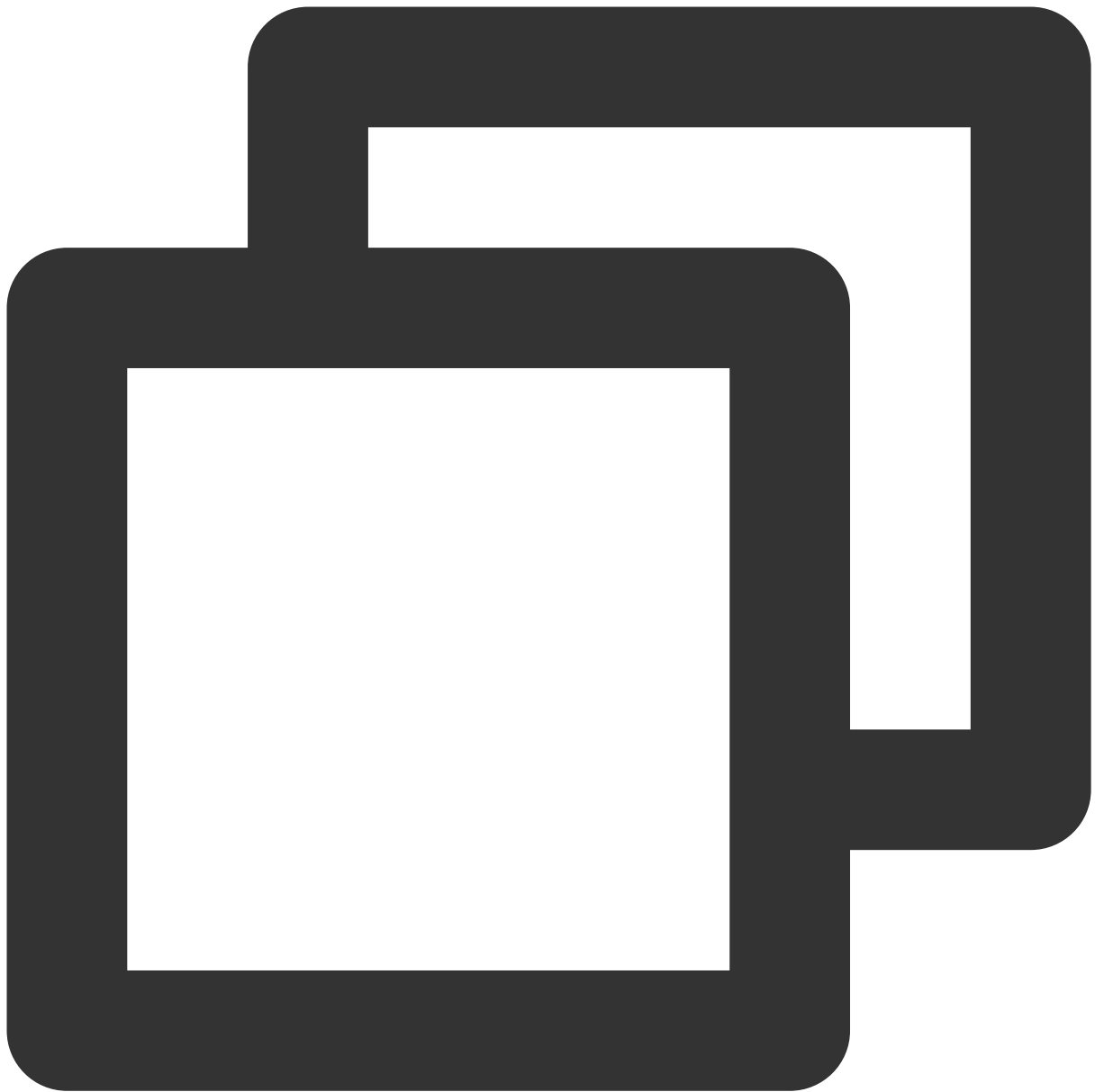


```
-keep class com.tencent.** { *; }
```

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 :
```

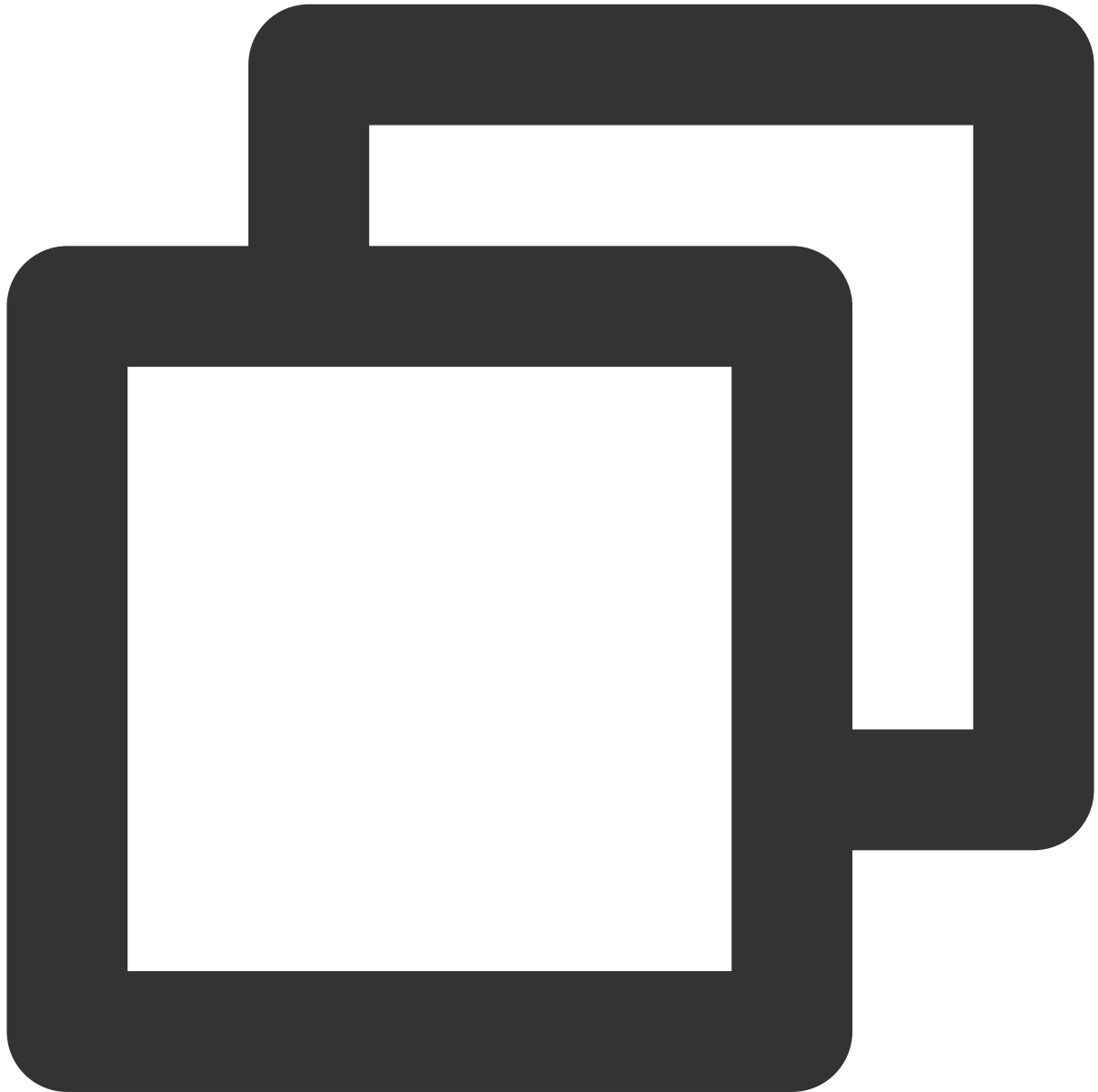




```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['VALID_ARCHS'] = 'arm64 arm64e x86_64'
      config.build_settings['VALID_ARCHS[sdk=iphonesimulator*]']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

4. If you need to use the audio and video feature on iOS, you need authorization for accessing microphone and camera.

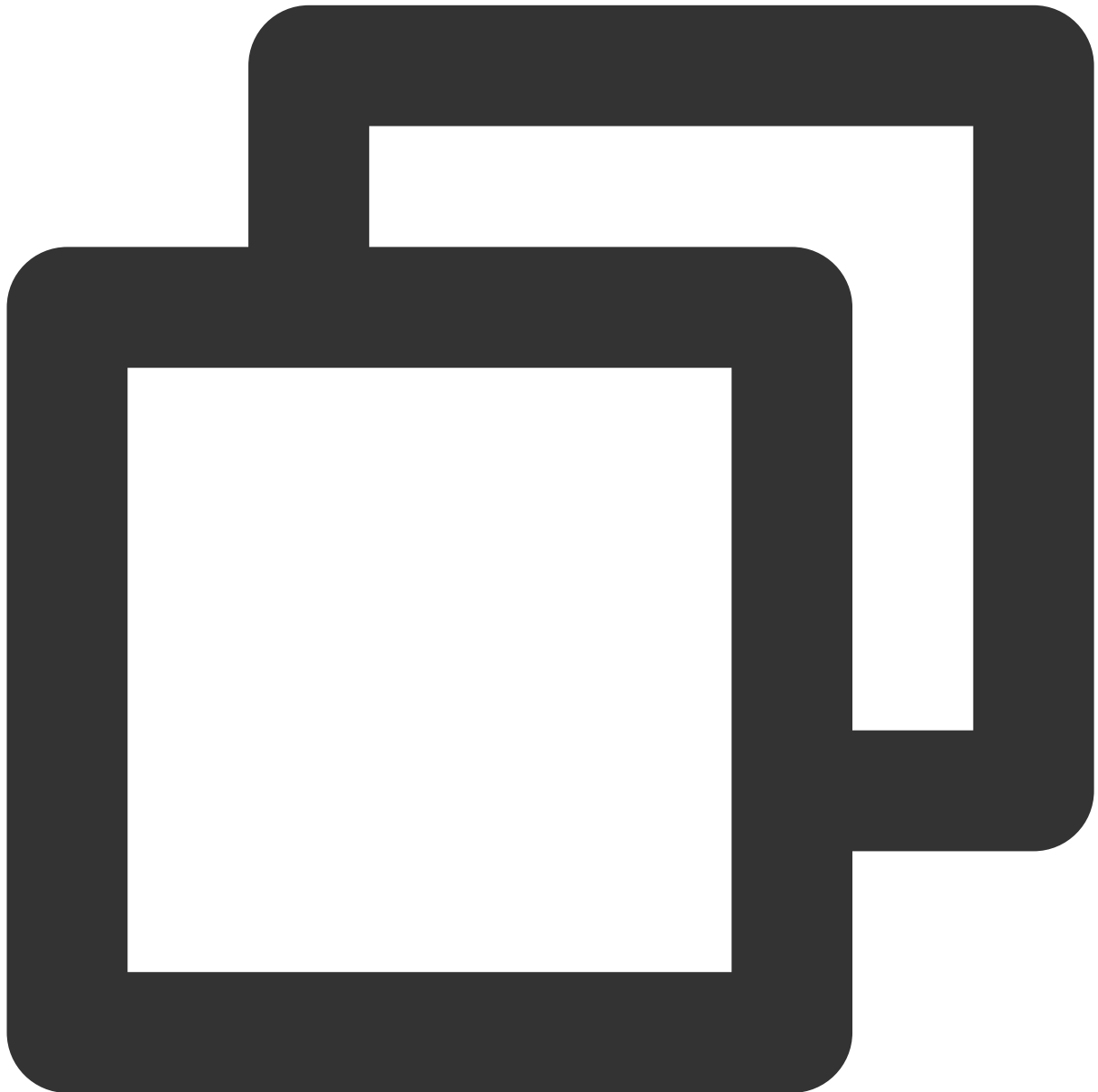
Authorization method: Add the following two items in your iOS project's Info.plist, corresponding to the prompt messages for the microphone and camera when the system pops up the authorization dialogue.



```
<key>NSCameraUsageDescription</key>  
<string>CallingApp needs to access your camera to capture video.</string>  
<key>NSMicrophoneUsageDescription</key>  
<string>CallingApp needs to access your microphone to capture voice.</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:



```
TUIResult result = TUICallKit.instance.login(SDKAppID, // Please replace it with  
                                              '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 no details are required 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.

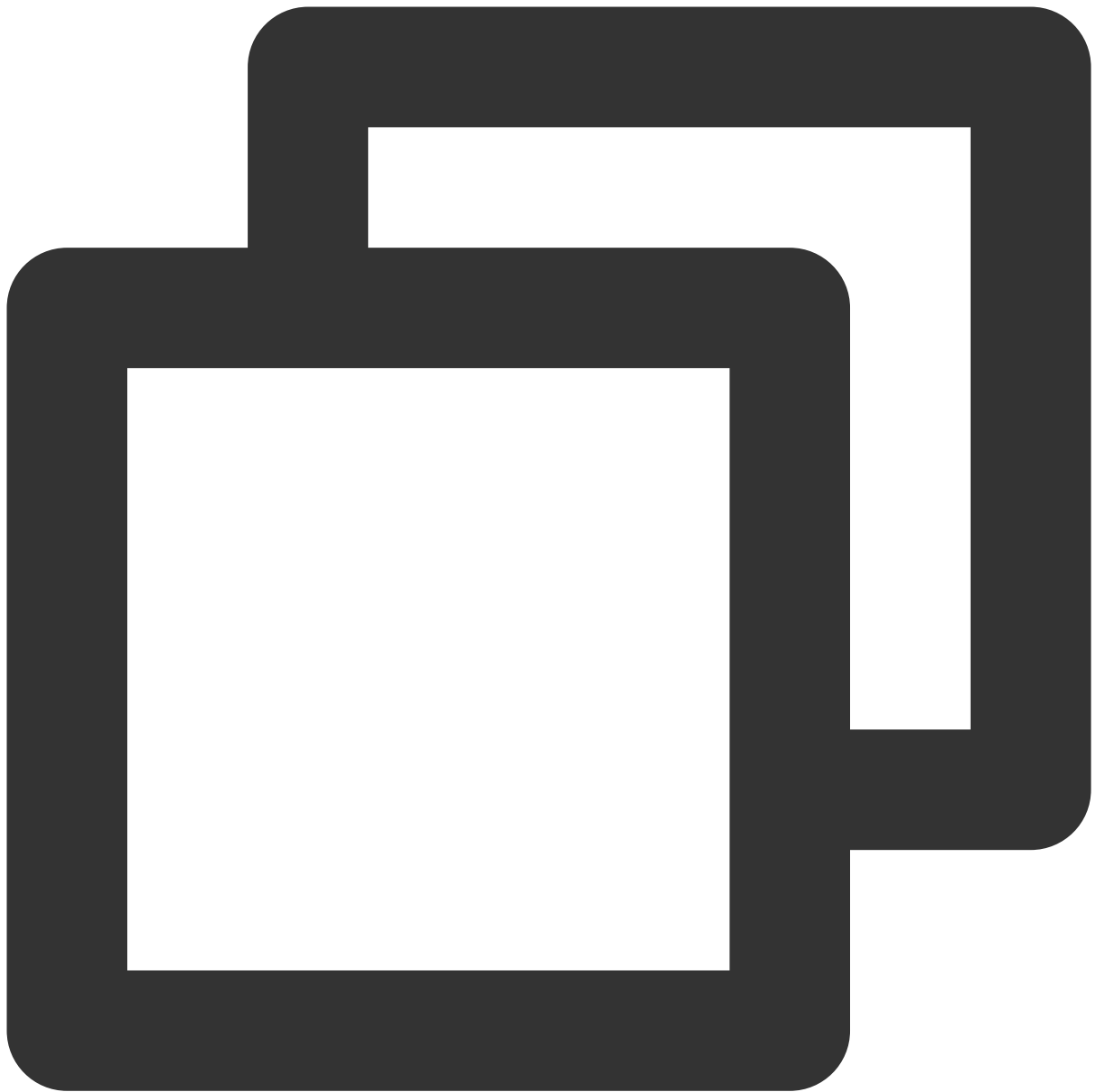
userSig is set to the value of Secretkey by mistake. The userSig is generated by using the SecretKey for the purpose of encrypting information such as sdkAppld, userId, and the expiration time. But the value of the userSig that is required cannot be directly substituted with the value of the SecretKey.

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.

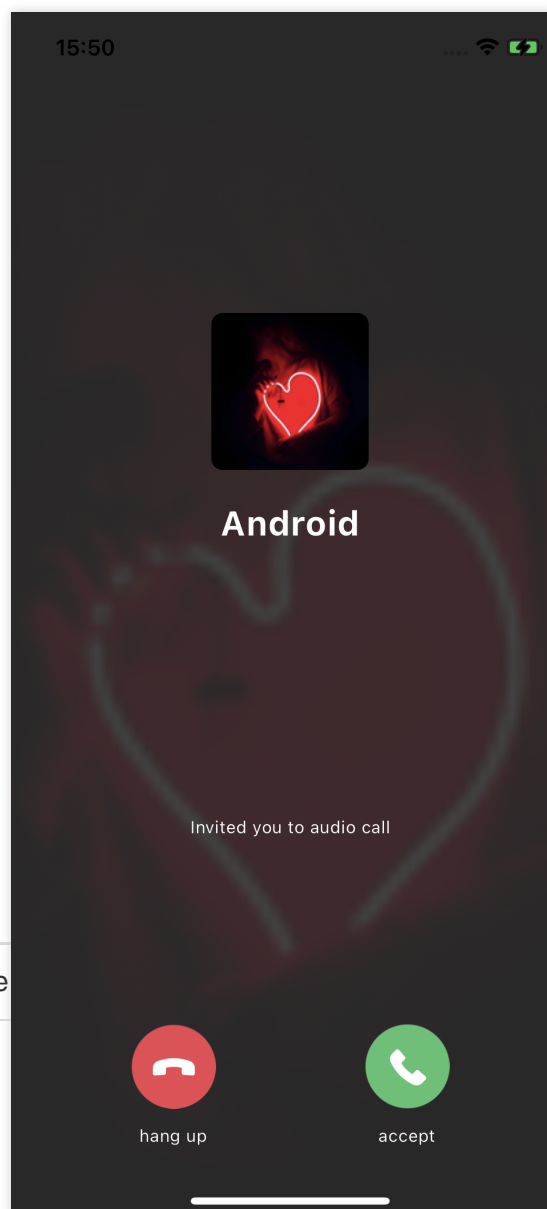
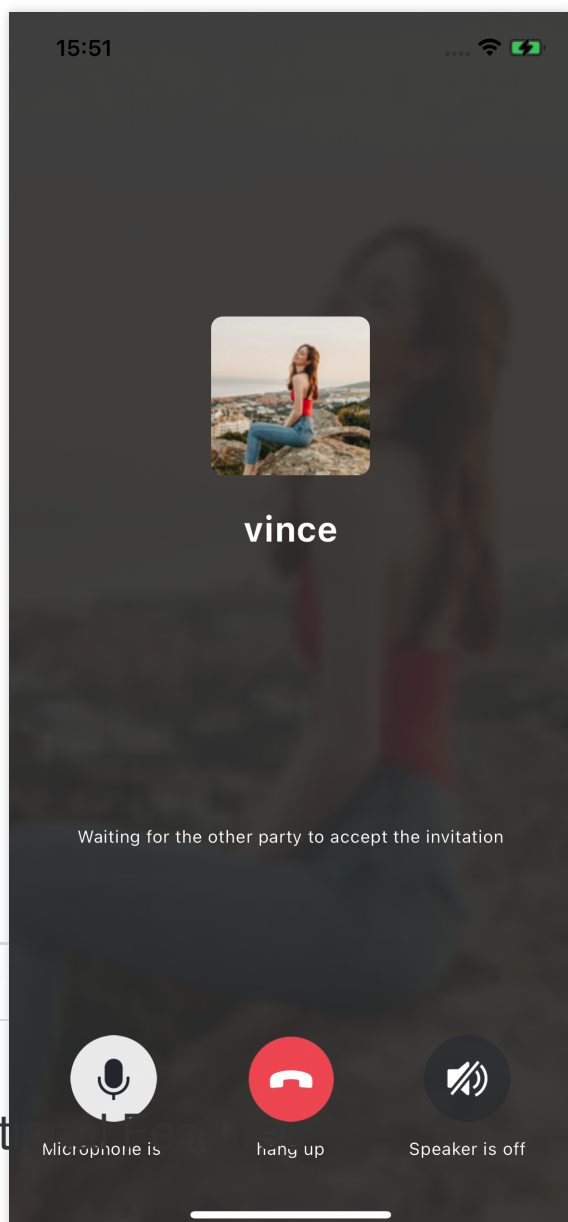
The sample code on GitHub uses the `genTestUserSig` function to calculate `UserSig` locally, so as to help you complete the current access process faster. However, this scheme exposes your `SecretKey` in the application code, which makes it difficult for you to upgrade and protect your `SecretKey` subsequently.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

## Step 5: Make your first call

After the caller and callee have successfully signed in, the caller can initiate an audio or video call by calling the `TUICallKit`'s call method and specifying the call type and callee's userId. The callee will then receive the incoming call invitation.



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



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](mailto:colleenyu@tencent.com).

# UI Customization (TUICallKit)

## Android

Last updated : 2024-05-15 16:53:56

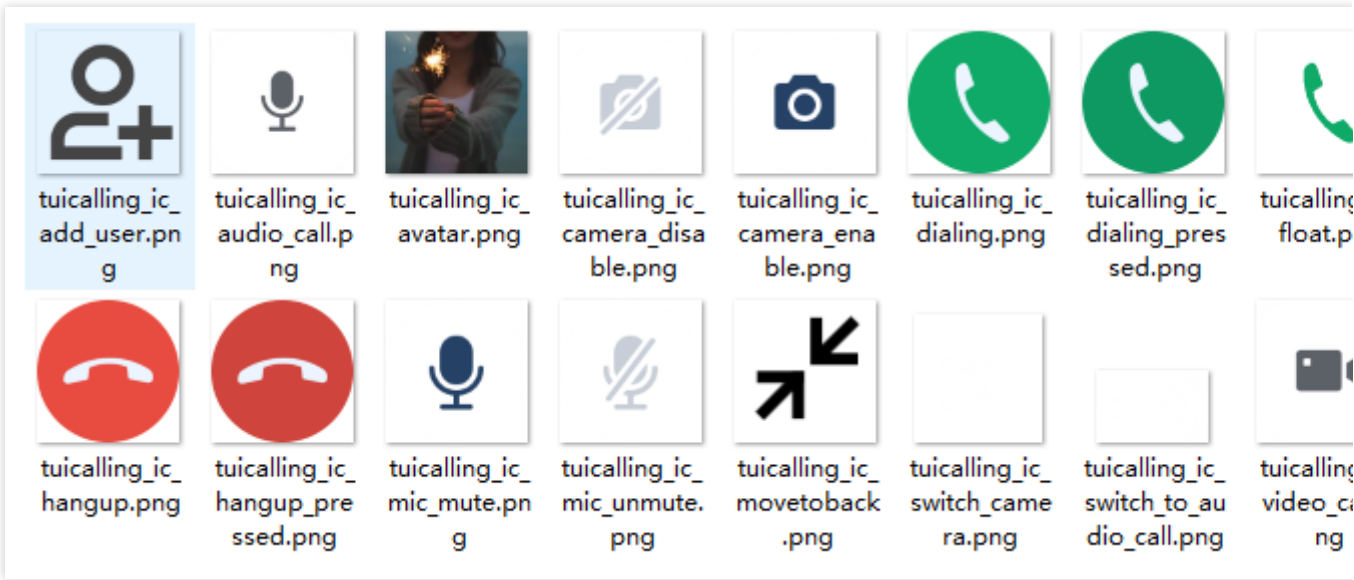
This document describes how to customize the UI of `TUICallKit` and provides two schemes for customization: **slight UI adjustment** and **custom UI implementation**.

### Scheme One. Slight UI Adjustment

You can adjust the UI of `TUICallKit` by directly modifying the UI source code in the `Android/tuicallkit` folder in [tencentyun/TUICallKit](#).

#### Replacing Icons

You can directly replace the icons in the `res\\drawable-xxhdpi` folder to customize the color tone and style of all the icons in your application. When you replace an icon, make sure the filename is the same as the original icon.



#### Replacing Ringtones

You can replace ringtones by replacing the three audio files in the `res\\raw` folder.

Filename	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 strings on the video call UI by modifying the `strings.xml` file in `values-zh` and `values-e` .

## Scheme Two. 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.

API	Description
<code>createInstance</code>	Creates a <code>TUICallEngine</code> instance (singleton pattern).
<code>destroyInstance</code>	Terminates a <code>TUICallEngine</code> instance (singleton pattern).
<code>init</code>	Completes the authentication of basic audio/video call capabilities.
<code>addObserver</code>	Adds an event callback.
<code>removeObserver</code>	Removes a callback API.
<code>call</code>	Makes a one-to-one call.
<code>groupCall</code>	Makes a group call.
<code>accept</code>	Answers a call.
<code>reject</code>	Declines a call.
<code>hangup</code>	Hangs up a call.
<code>ignore</code>	Ignores a call.
<code>inviteUser</code>	Invites a user during a group call.
<code>joinInGroupCall</code>	Joins the current group call actively.

<code>switchCallMediaType</code>	Switches the call media type, such as from video call to audio call.
<code>startRemoteView</code>	Starts subscribing to the video stream of a remote user.
<code>stopRemoteView</code>	Stops subscribing to the video stream of a remote user.
<code>openCamera</code>	Enables the camera.
<code>closeCamera</code>	Disables the camera.
<code>switchCamera</code>	Switches between the front and rear cameras.
<code>openMicrophone</code>	Enables the mic.
<code>closeMicrophone</code>	Disables the mic.
<code>selectAudioPlaybackDevice</code>	Selects the audio playback device (receiver/speaker on the device).
<code>setSelfInfo</code>	Sets the user nickname and profile photo.
<code>enableMultiDeviceAbility</code>	Enables/Disables the multi-device log-in mode of <code>TUICallEngine</code> (supported by the premium plan).

## TUICallObserver

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

API	Description
<code>onError</code>	An error occurred during the call.
<code>onCallReceived</code>	A call was received.
<code>onCallCancelled</code>	The call was canceled.
<code>onCallBegin</code>	The call was connected.
<code>onCallEnd</code>	The call ended.
<code>onCallMediaTypeChanged</code>	The call media type changed.
<code>onUserReject</code>	A user declined the call.
<code>onUserNoResponse</code>	A user didn't respond.
<code>onUserLineBusy</code>	A user was busy.
<code>onUserJoin</code>	A user joined the call.

<a href="#">onUserLeave</a>	A user left the call.
<a href="#">onUserVideoAvailable</a>	Whether a user had a video stream.
<a href="#">onUserAudioAvailable</a>	Whether a user had an audio stream.
<a href="#">onUserVoiceVolumeChanged</a>	The volume levels of all users.
<a href="#">onUserNetworkQualityChanged</a>	The network quality of all users.

## Definitions of Key Types

API	Description
TUICallDefine.MediaType	The call media 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 audio/video 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 information of the current network quality.

# Web

Last updated : 2024-08-09 17:43:52

This document describes how to customize the UI of `TUICallKit` and provides two schemes for customization: **slight UI adjustment** and **custom UI implementation**.

## Scheme 1: Slight UI Adjustment

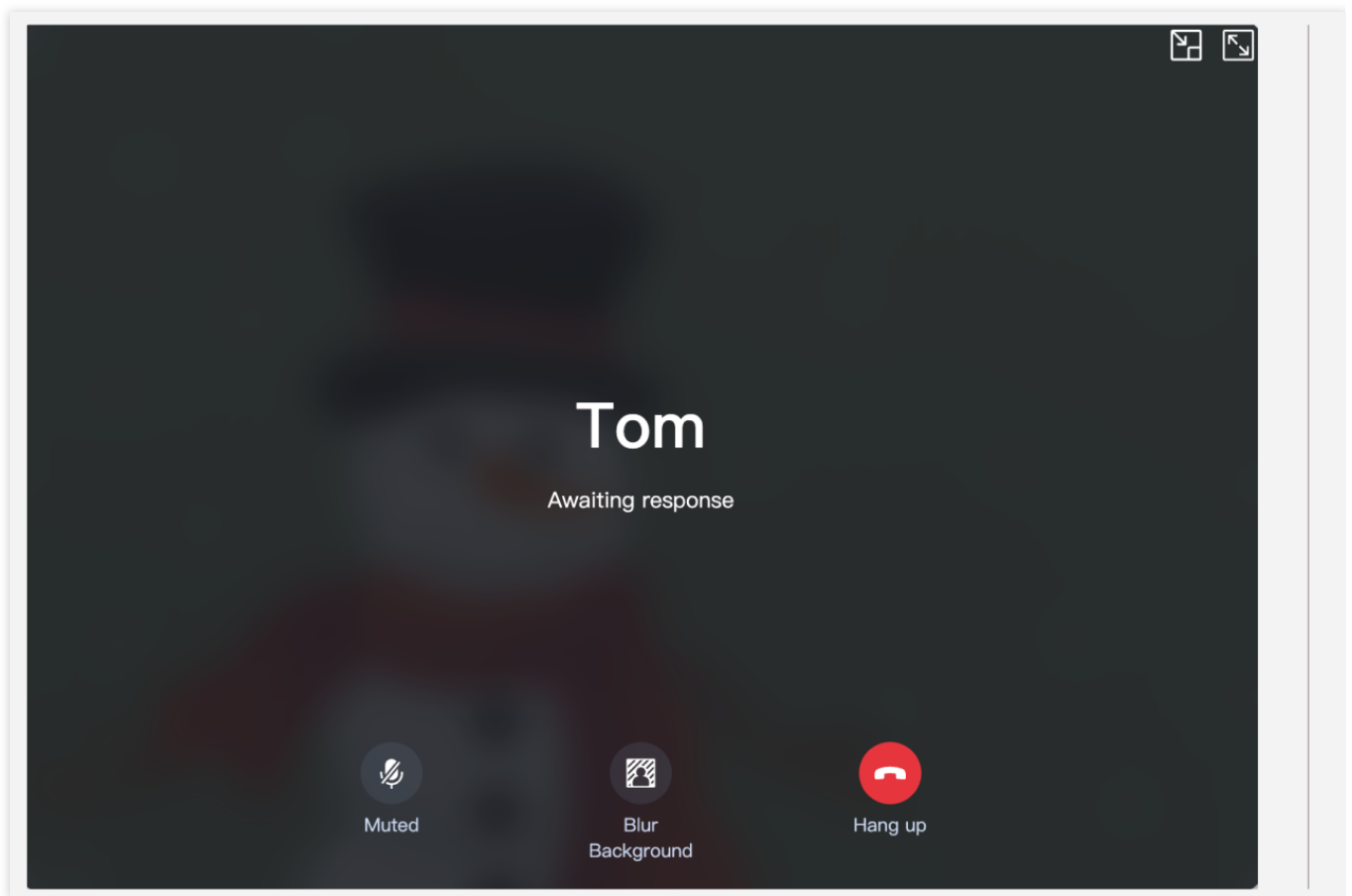
### Button Hiding

Invoke the `hideFeatureButton` interface to hide buttons, currently supporting Camera, Microphone, SwitchCamera. For details, see the enumeration type `FeatureButton`.

#### Note:

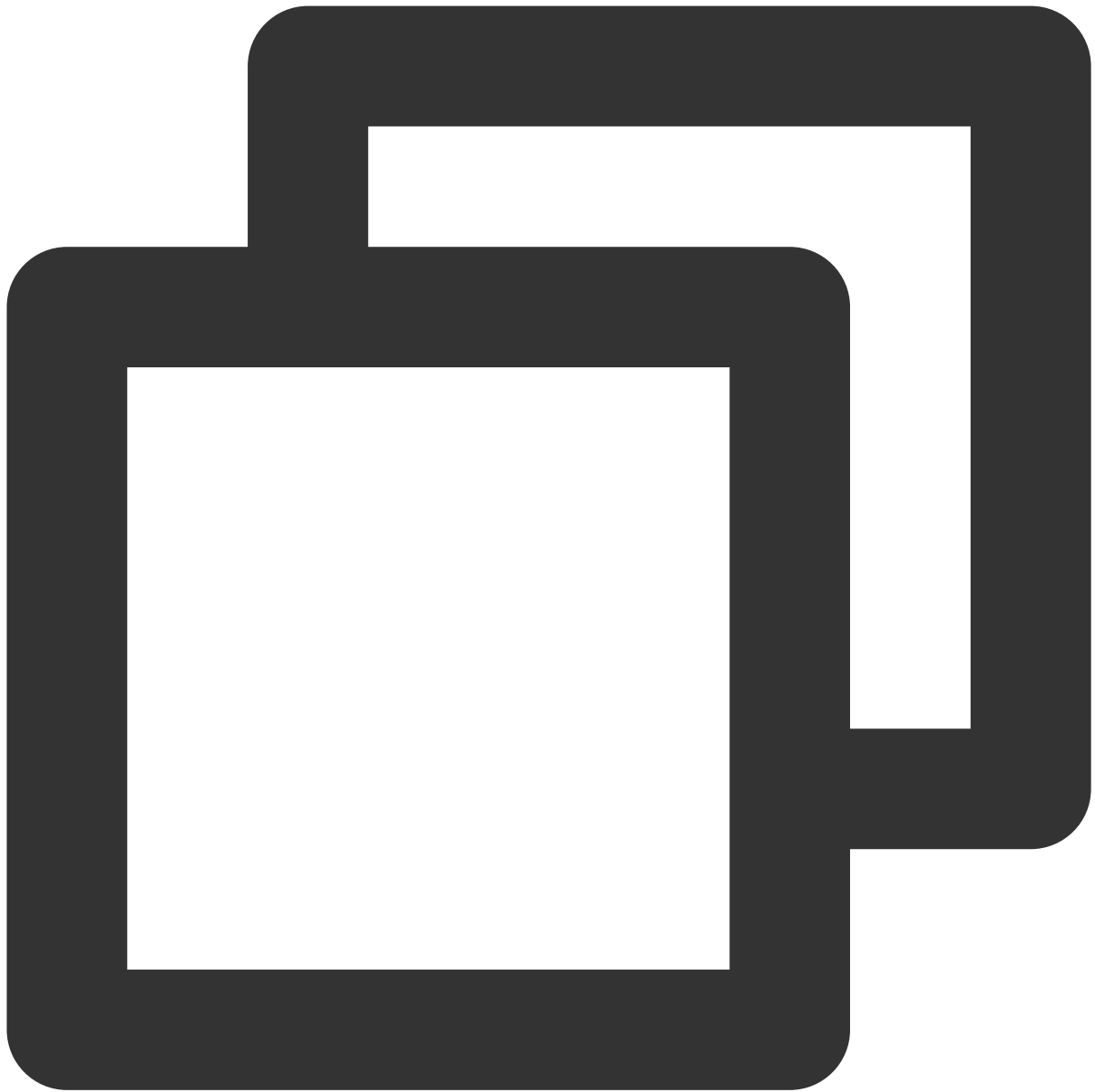
**v3.2.9+ support.**

Taking the hiding of the **Camera Button** as an example.

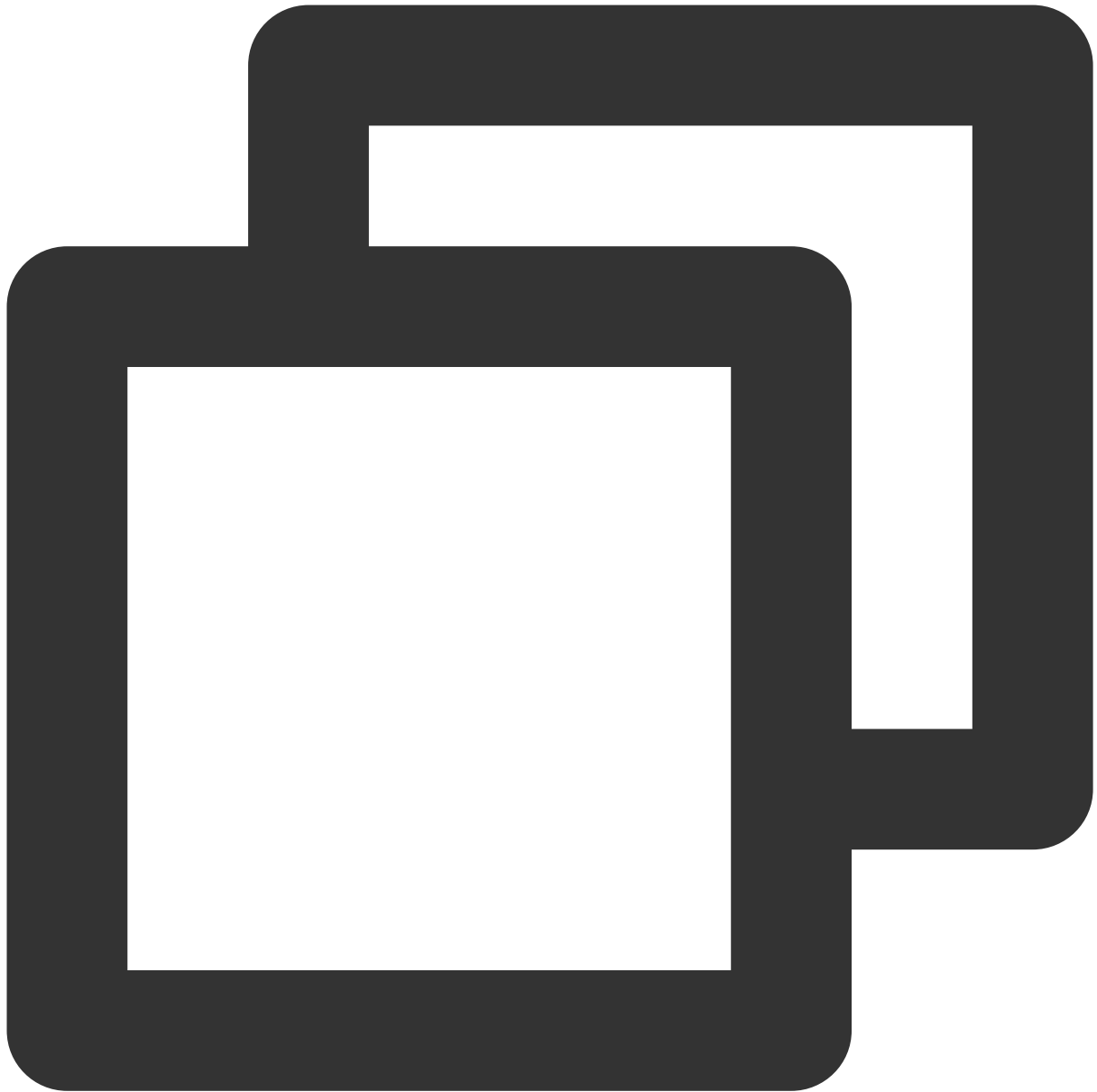


Vue3

React



```
import { TUICallKitServer, FeatureButton } from "@tencentcloud/call-uikit-vue";  
  
TUICallKitServer.hideFeatureButton(FeatureButton.Camera);
```



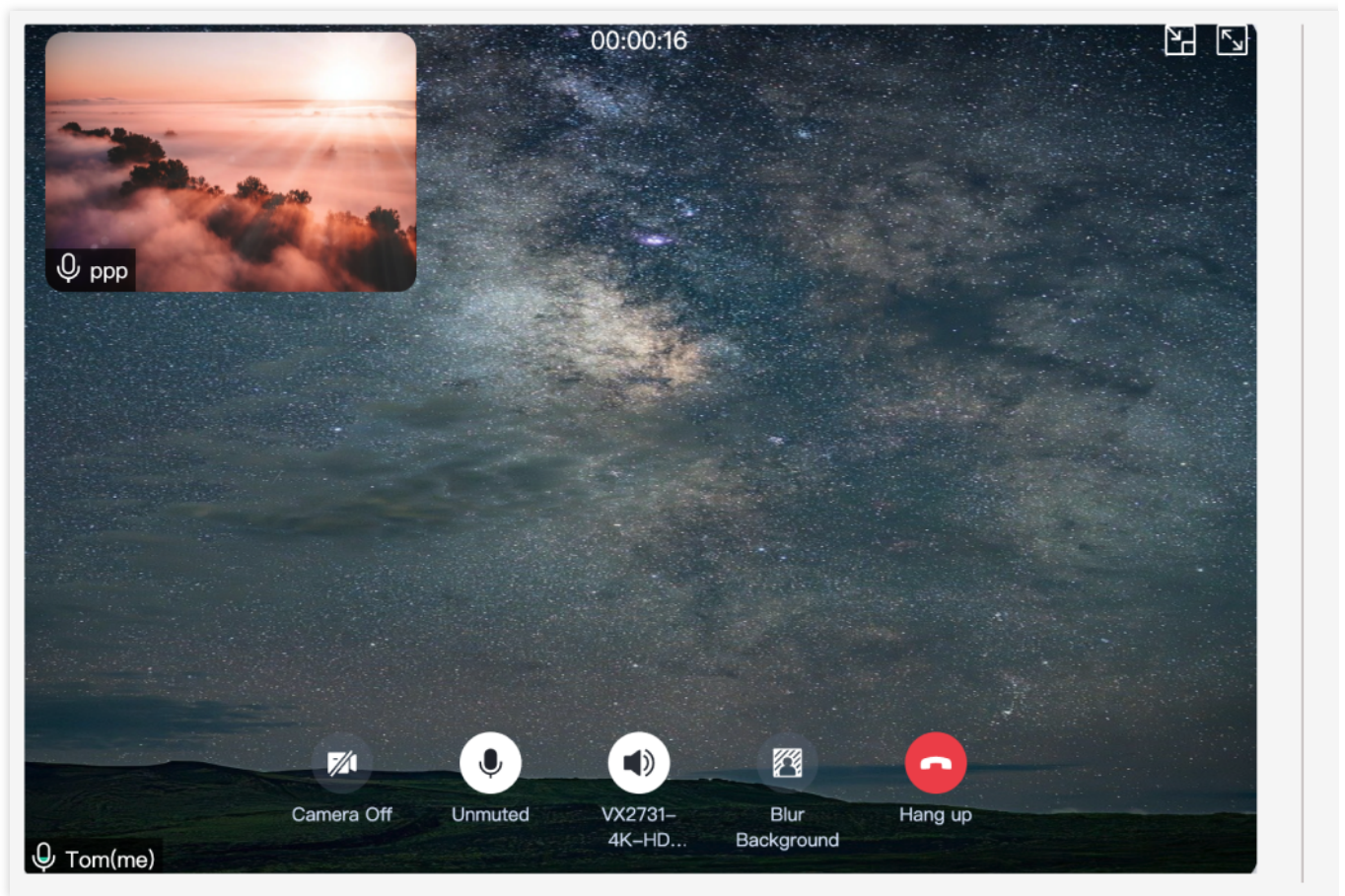
```
import { TUICallKitServer, FeatureButton } from "@tencentcloud/call-uikit-react";  
  
TUICallKitServer.hideFeatureButton(FeatureButton.Camera);
```

## Custom Call Background Image

The call background image appears when the camera is turned off during a voice or video call. Modify the local user's call interface background image by calling [setLocalViewBackgroundImage](#), and modify the remote user's call interface background image with [setRemoteViewBackgroundImage](#).

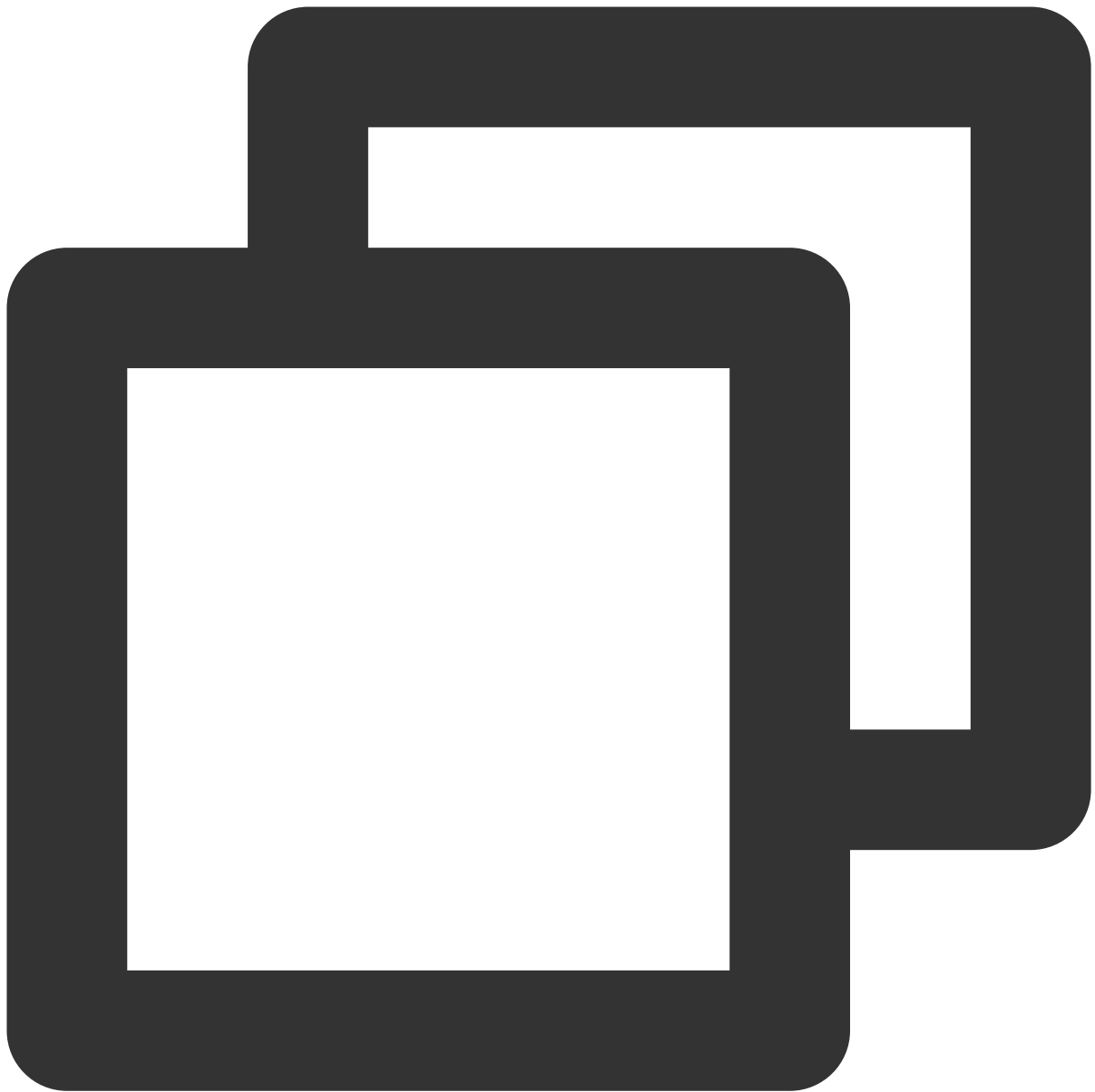
**Note:**

v3.2.9+ support.



Vue3

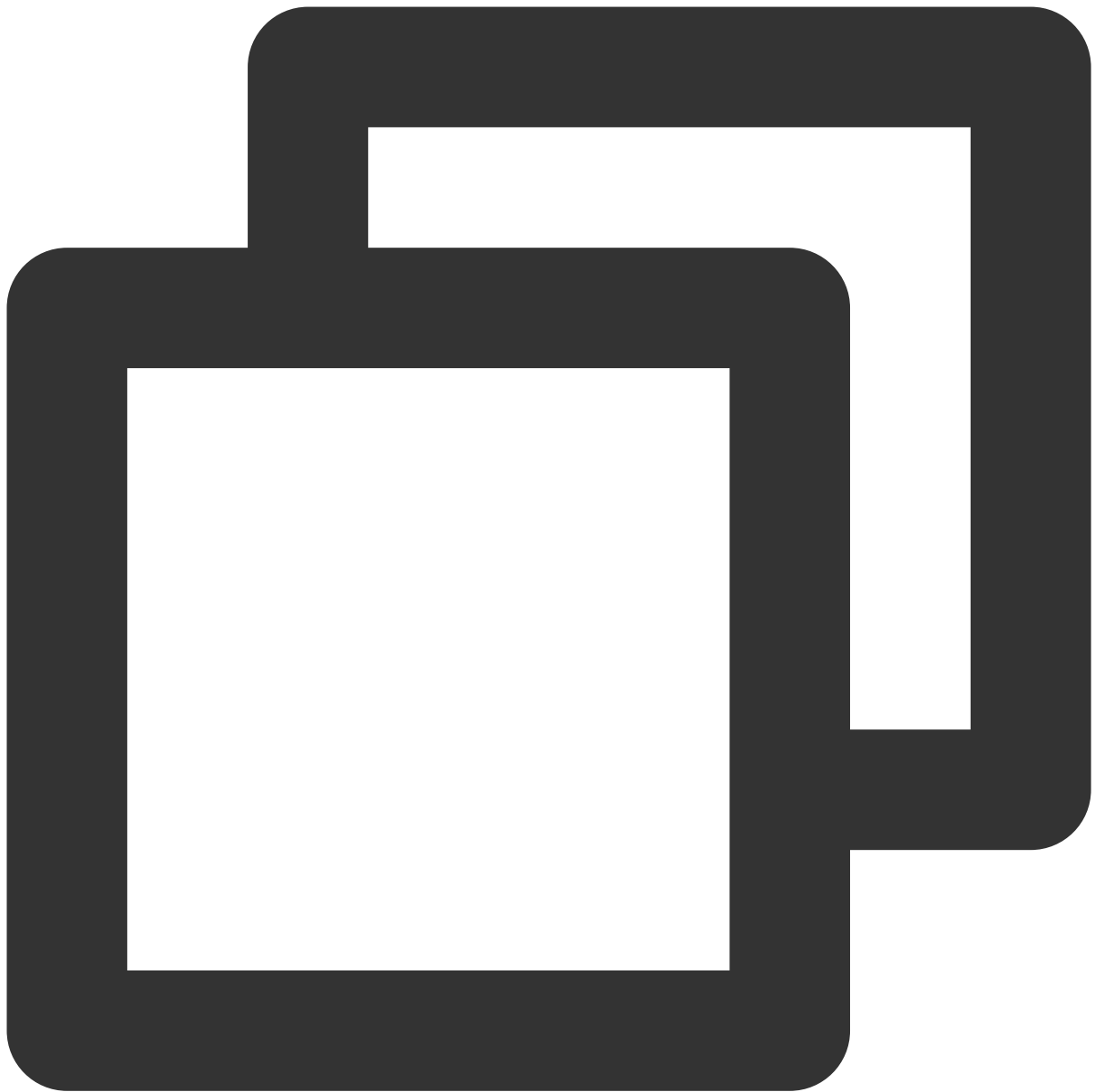
React



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

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





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

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

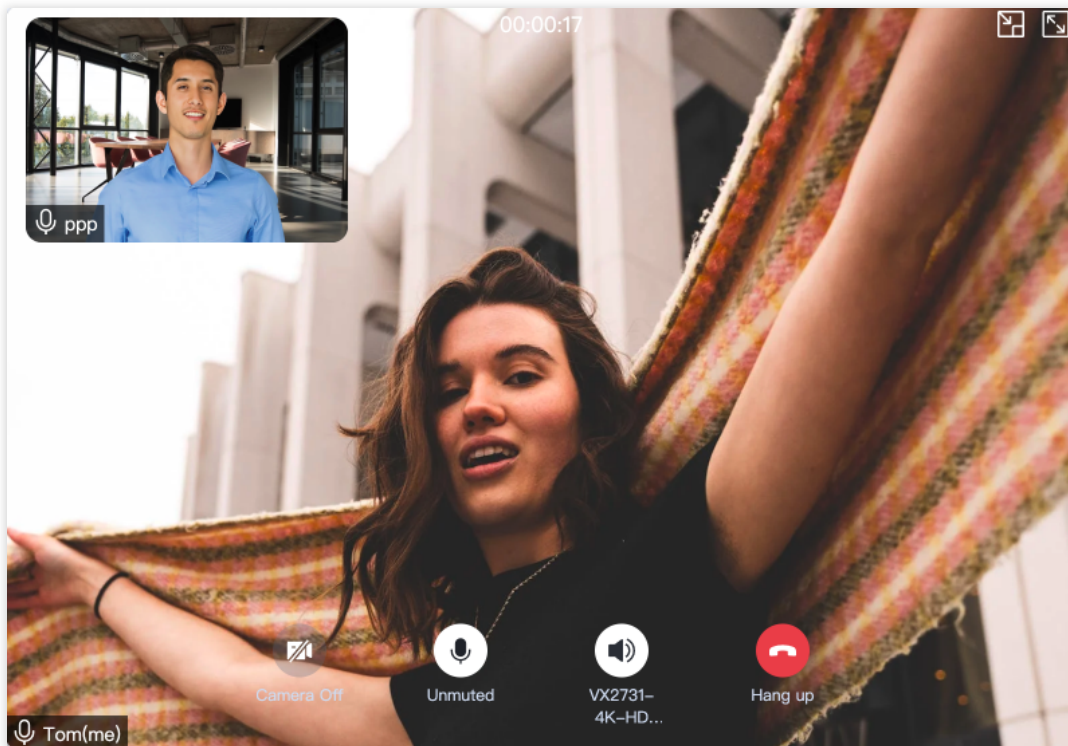
## Set Layout

### Note :

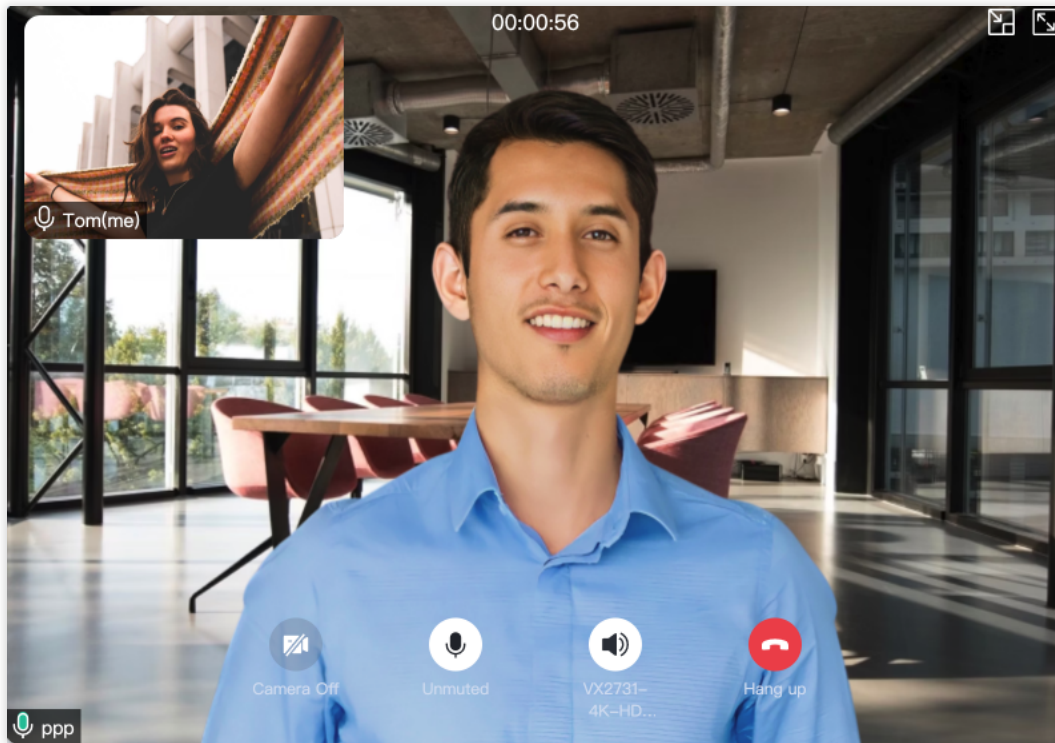
Only available for 1V1 video calls, supported from v3.3.0+.

Use [setLayoutMode](#) to set the call interface layout, currently only supports LocalInLargeView and RemoteInLargeView, see the [LayoutMode](#) enum for details.

1. LocalInLargeView layout, with the local user in the large window:

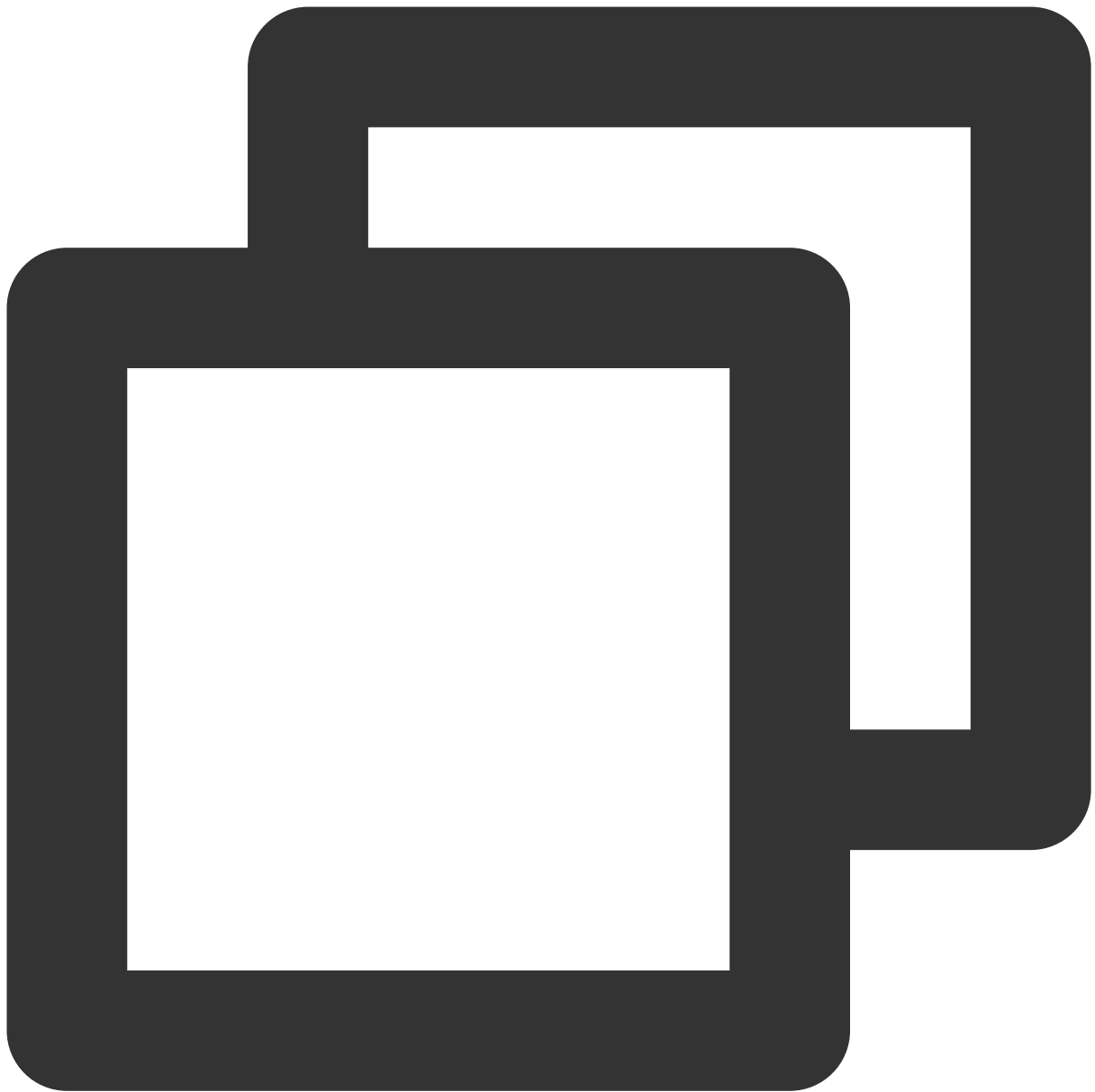


2. RemoteInLargeView layout, with the remote user in the large window:

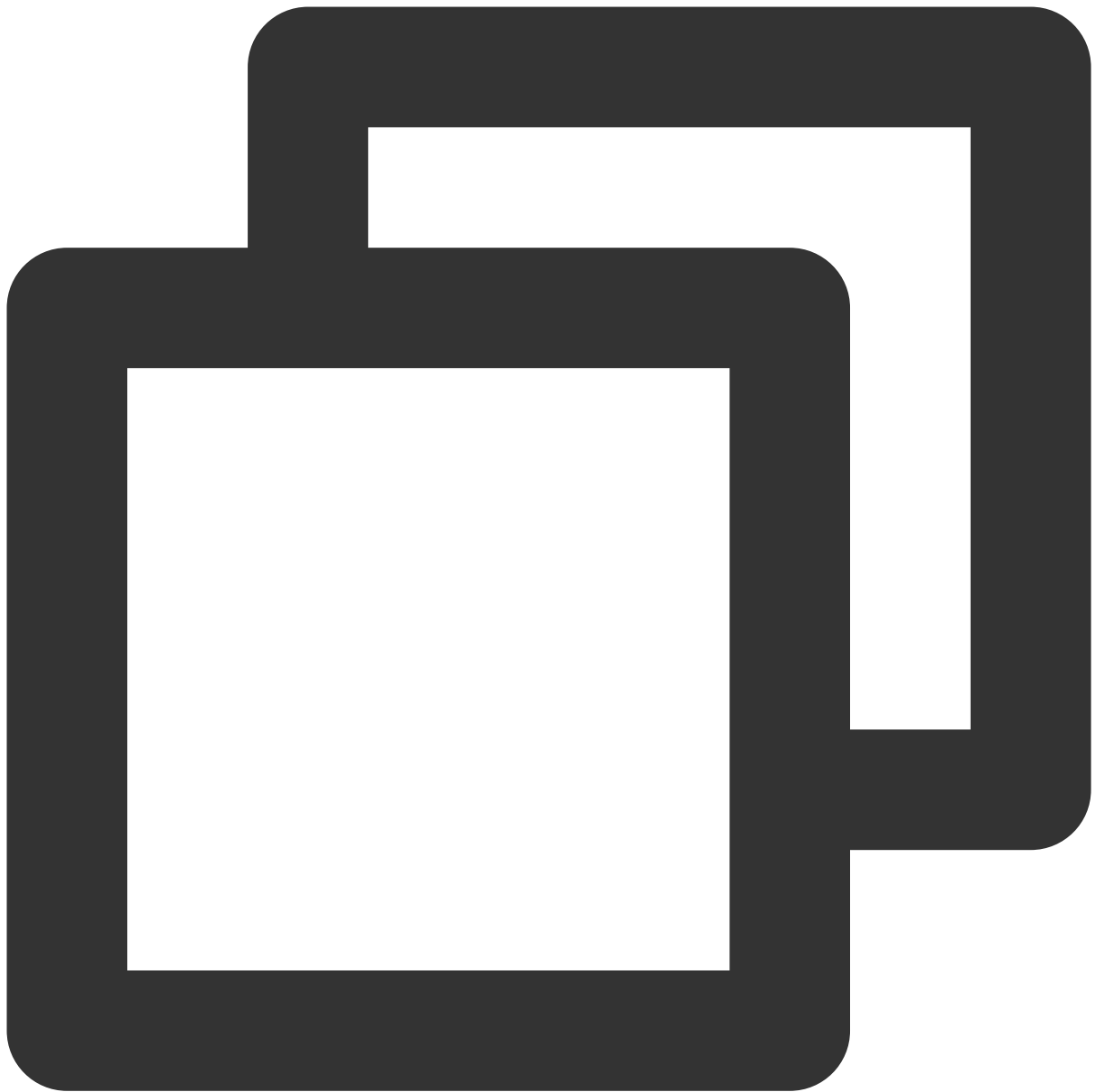


Vue3

React



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



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

## Set the initial state of the camera

### Note :

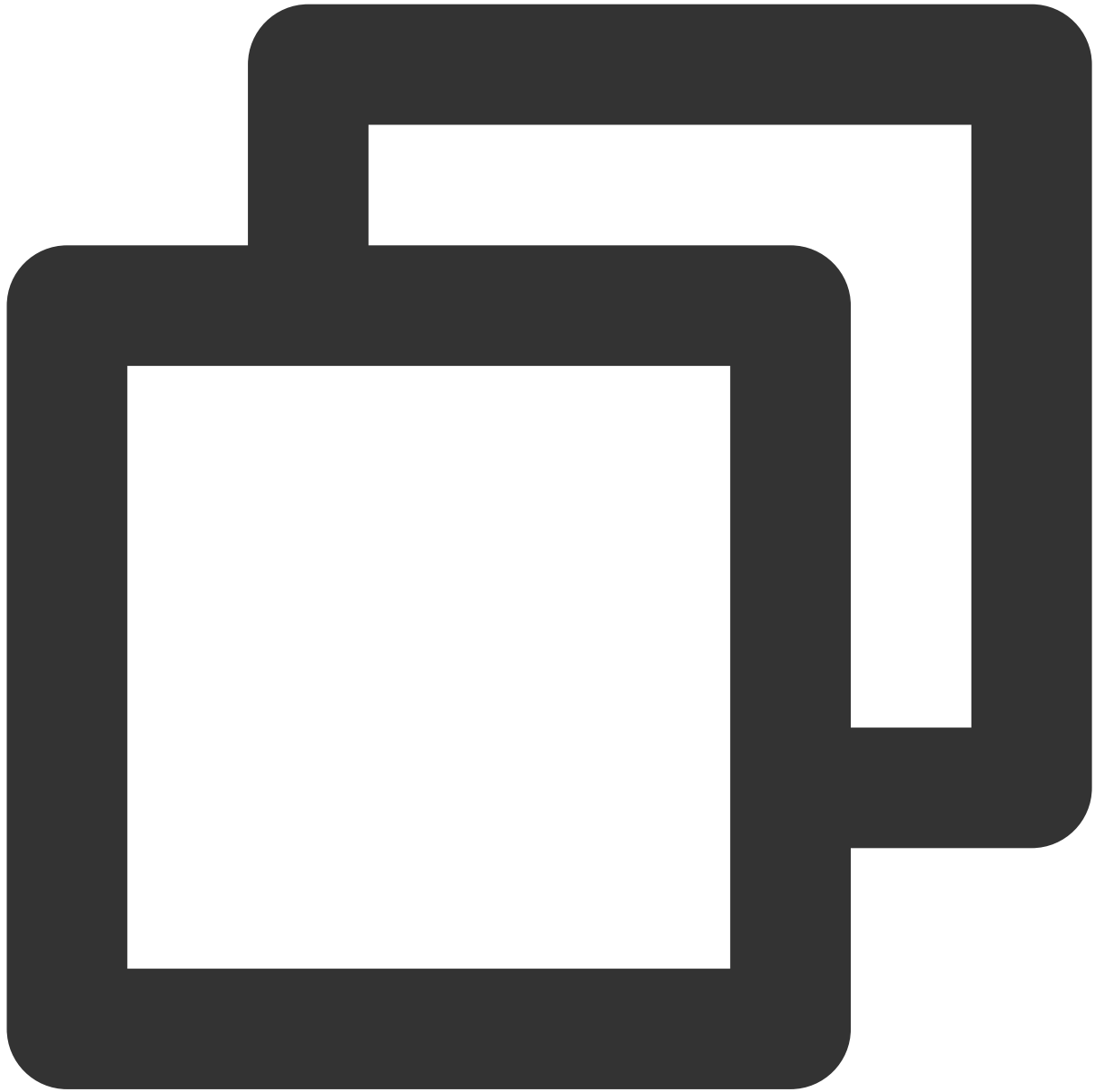
Supported from v3.3.0+.

Use [setCameraDefaultState](#) to set the initial state of the camera button, currently supports Enabled and Off.

Taking the default Off state of the camera as an example:

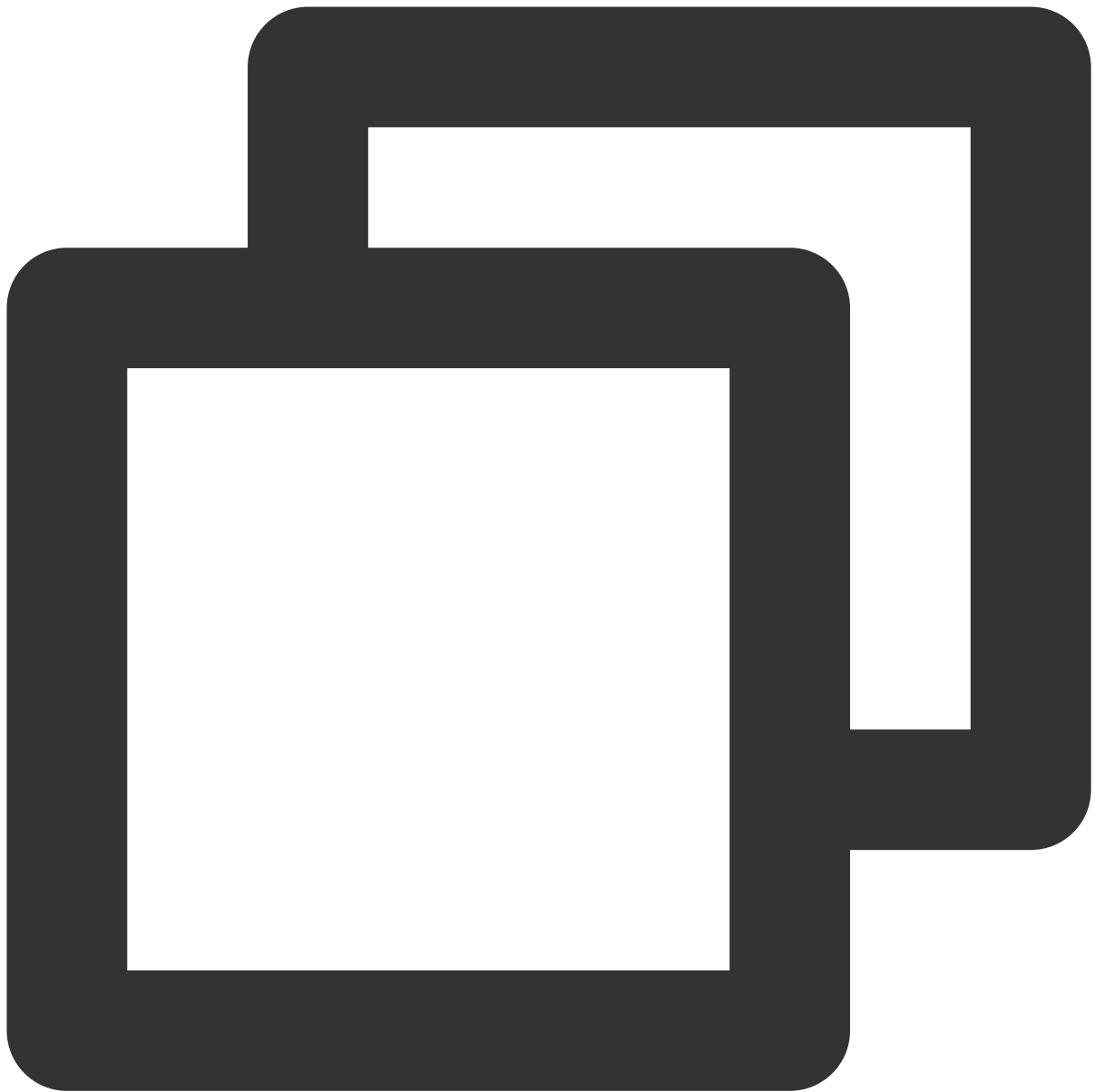
Vue3

React



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

TUICallKitServer.setCameraDefaultState(false);
```



```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";  
  
TUICallKitServer.setCameraDefaultState(false);
```

## Replacing icons

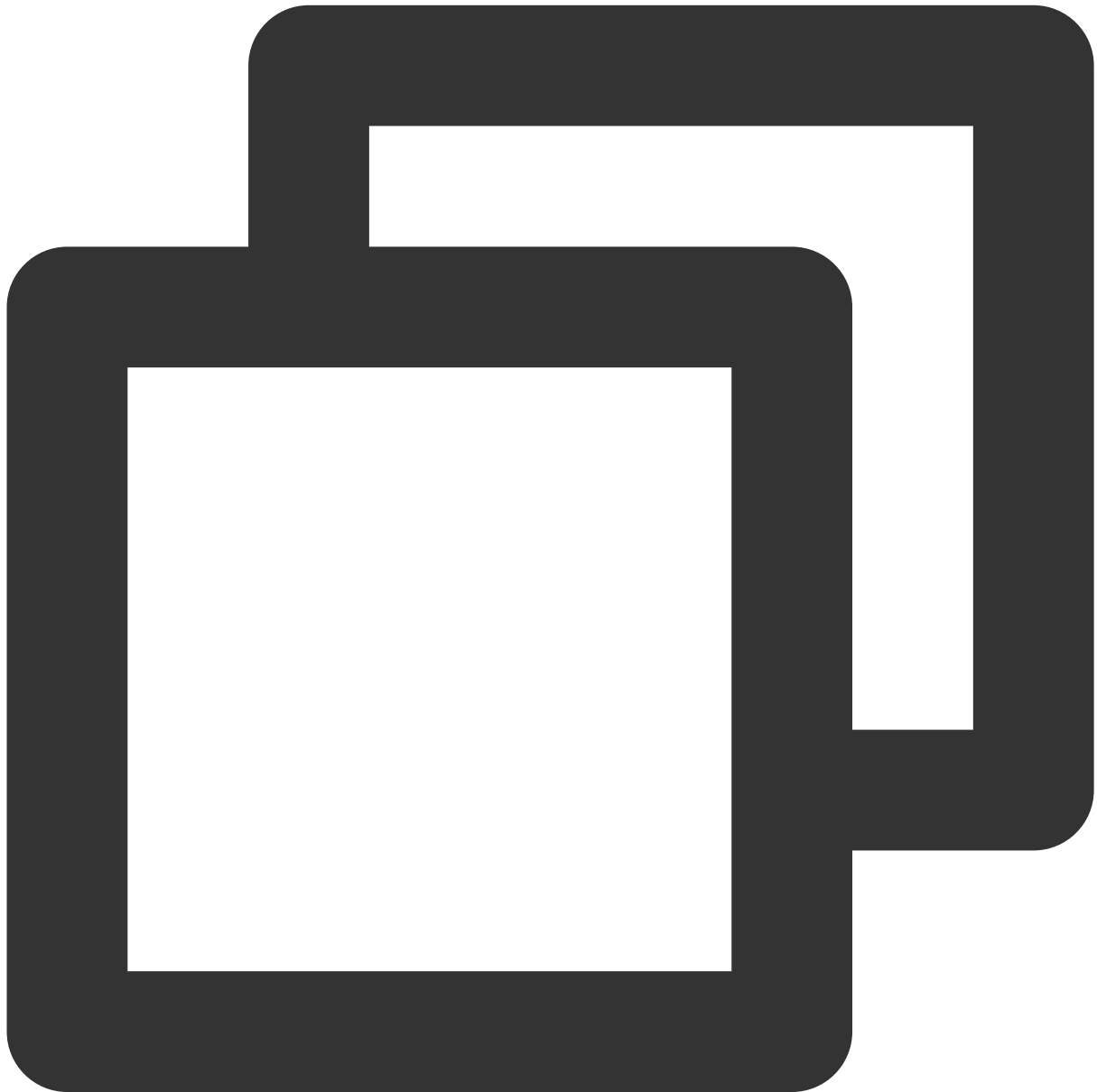
To replace an icon, source code import is required first. Copy the component to your project (the source code is in TypeScript version).

**Note :**

The Interface Replacing icons Plan is suitable for `Vue3 + TypeScript` and `@tencentcloud/call-uikit-vue` version number is 3.2.2 or later projects. If you are using other languages or technology stacks, please use the Custom UI Implementation.

### 1. Download Source Code

Vue3



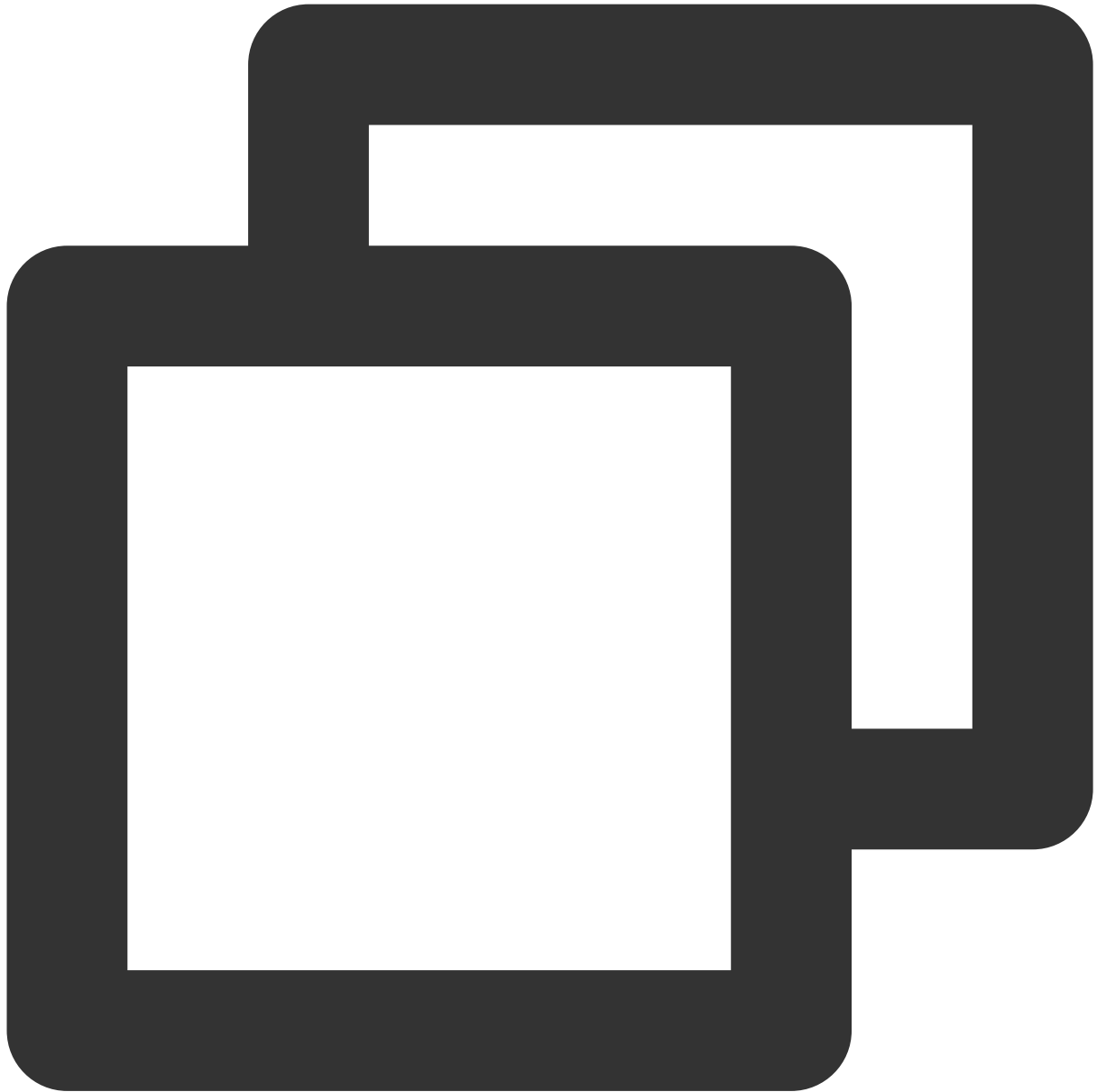
```
npm install @tencentcloud/call-uikit-vue
```

2. Copy the source code into your own project, taking copying into the `src/components/` directory as an example:

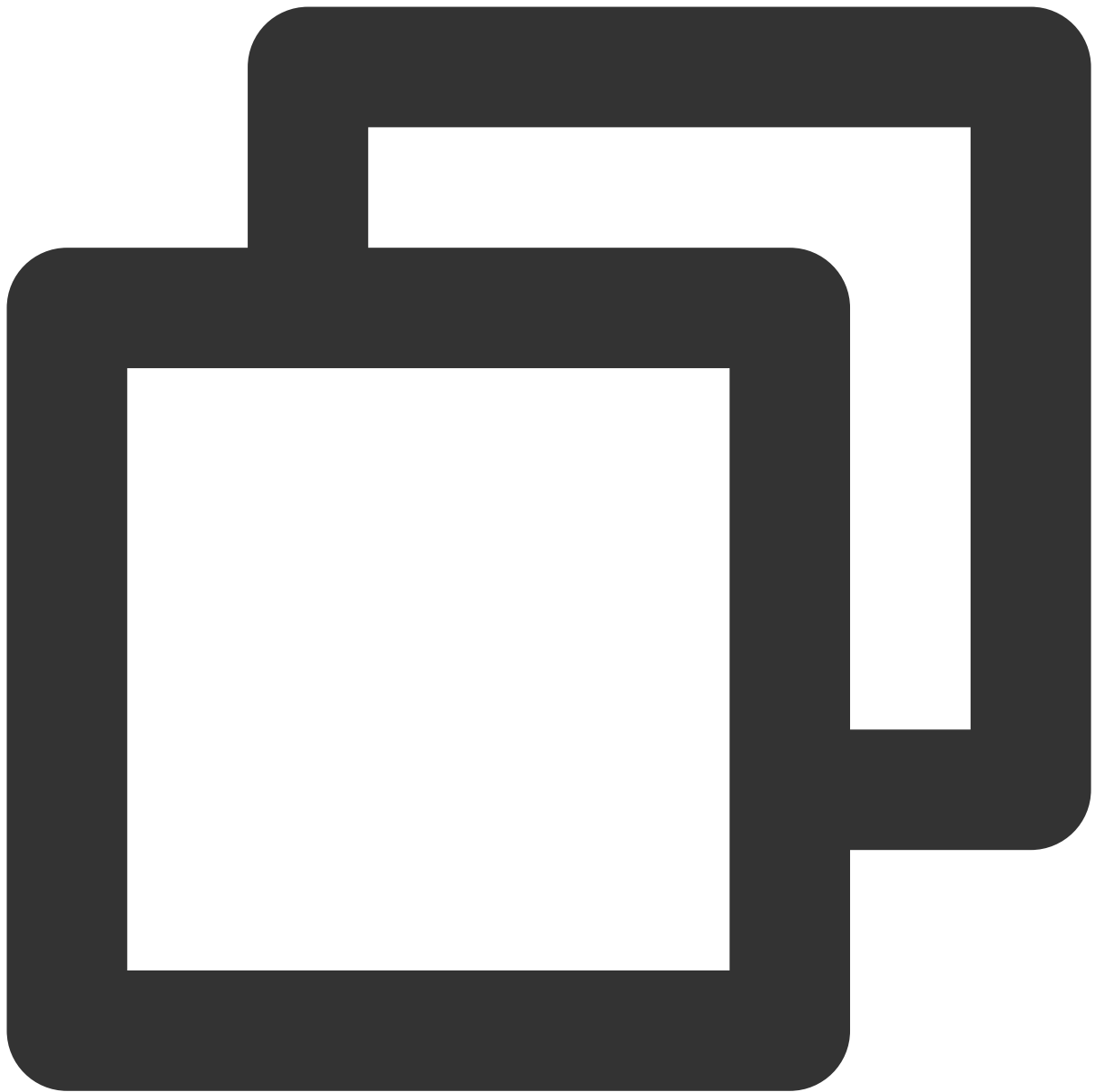


macOS + Vue3

Windows + Vue3



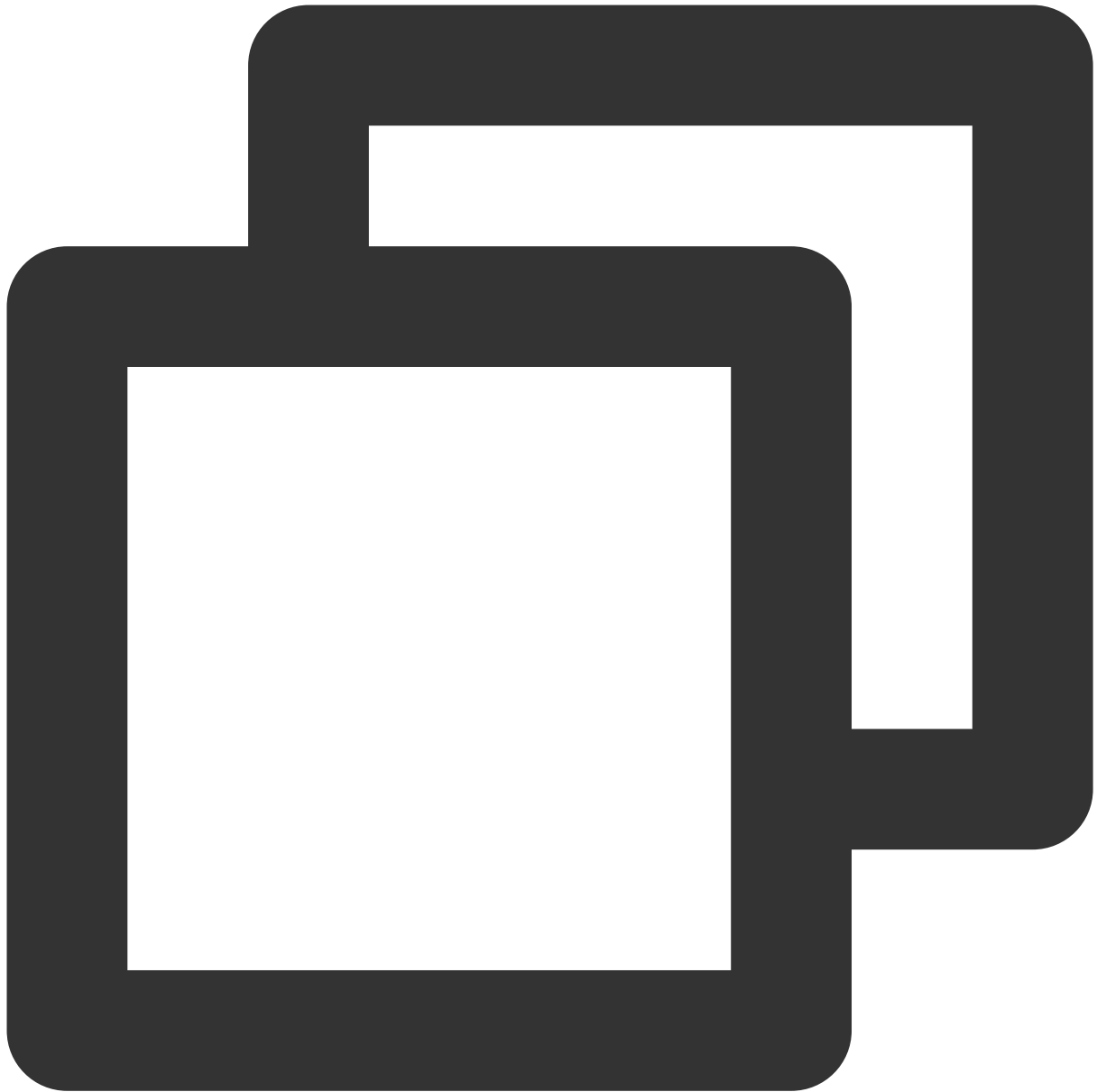
```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uik
```



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

### 3. Modify Import Path

It's necessary to change CallKit to be imported from a local file, as shown below. For other usage details, refer to [TUICallKit Quick Integration](#).



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

4.

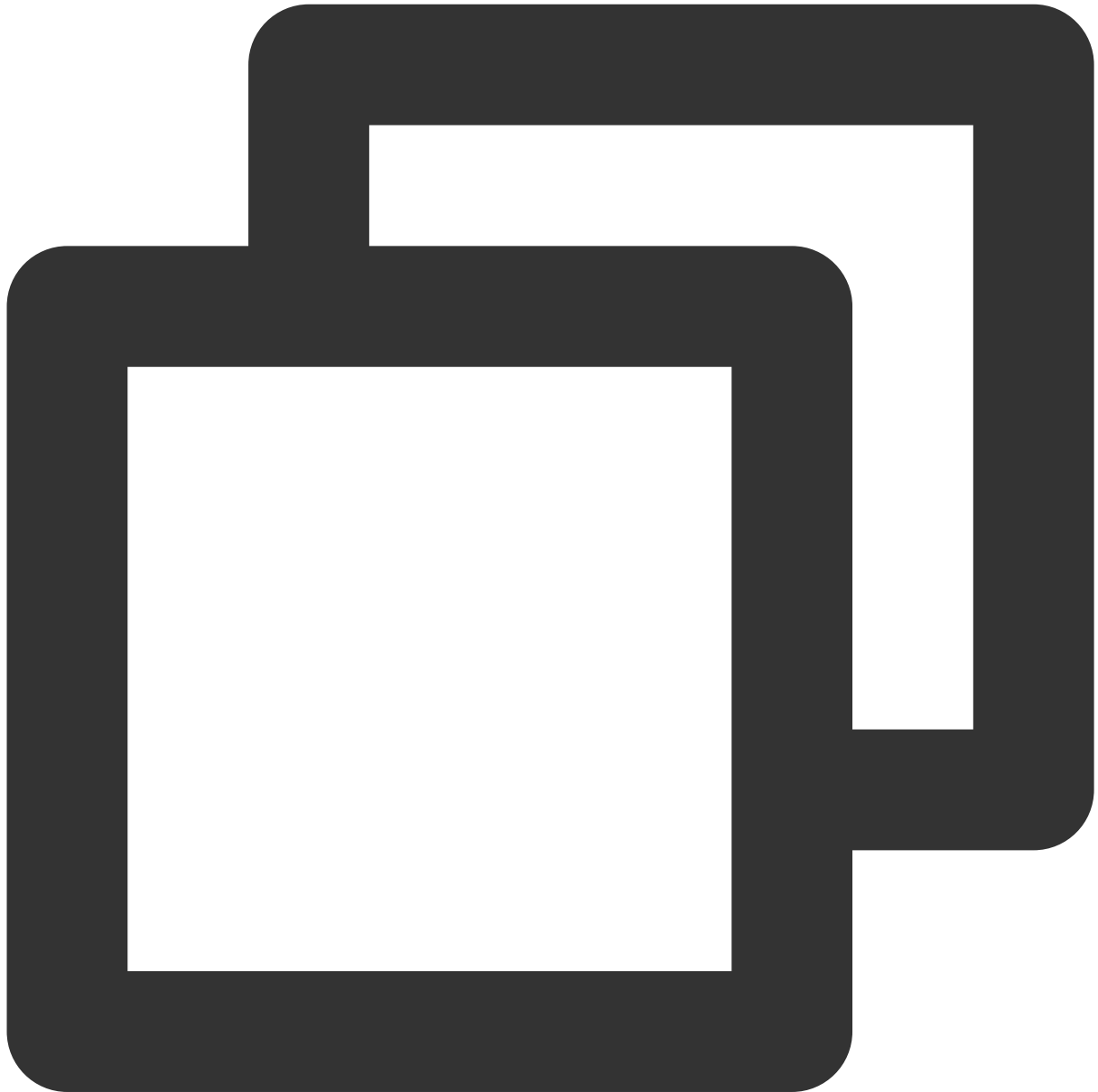
#### **Solve Errors That May Be Caused by Copying Source Code**

If you encounter an error while using the TUICallKit component, please don't worry. In most cases, this is due to inconsistencies between ESLint and TSConfig configurations. You can consult the documentation and configure correctly as required. If you need help, please feel free to contact us, and we will ensure that you can successfully use this component. Here are some common issues:

## ESLint Error

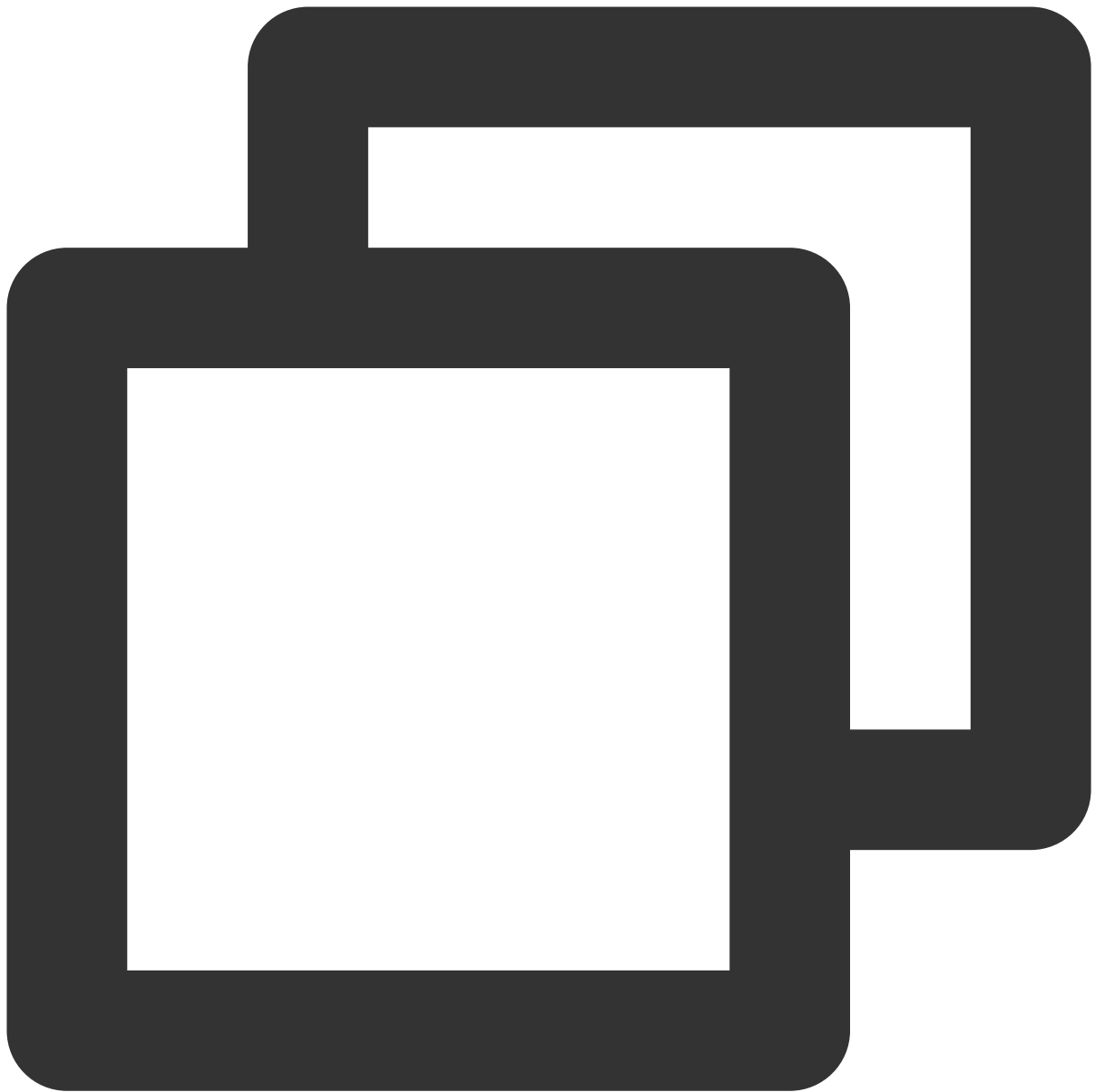
## TypeScript Error

If the TUICallKit causes an error due to inconsistency with your project's code style, you can block this component directory by adding a `.eslintignore` file in the root directory of your project, for example:



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

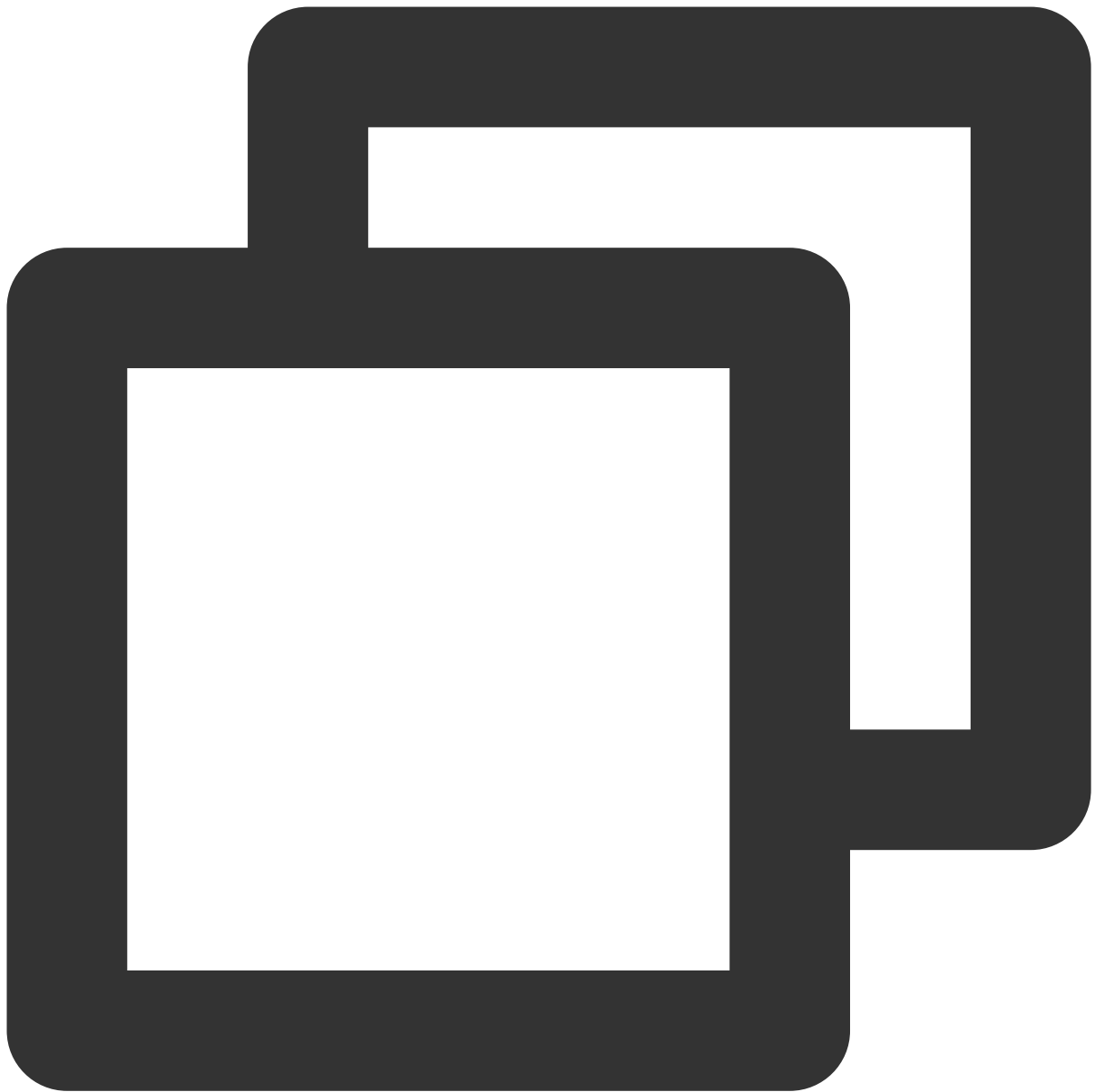
1. If you encounter the 'Cannot find module '../package.json' error, it's because TUICallKit references a JSON file. You can add the related configuration in tsconfig.json, example:



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

For other TSConfig issues, please refer to [TSConfig Reference](#).

2. If you encounter the 'Uncaught SyntaxError: Invalid or unexpected token' error, it's because TUICallKit uses decorators. You can add the related configuration in tsconfig.json, example:



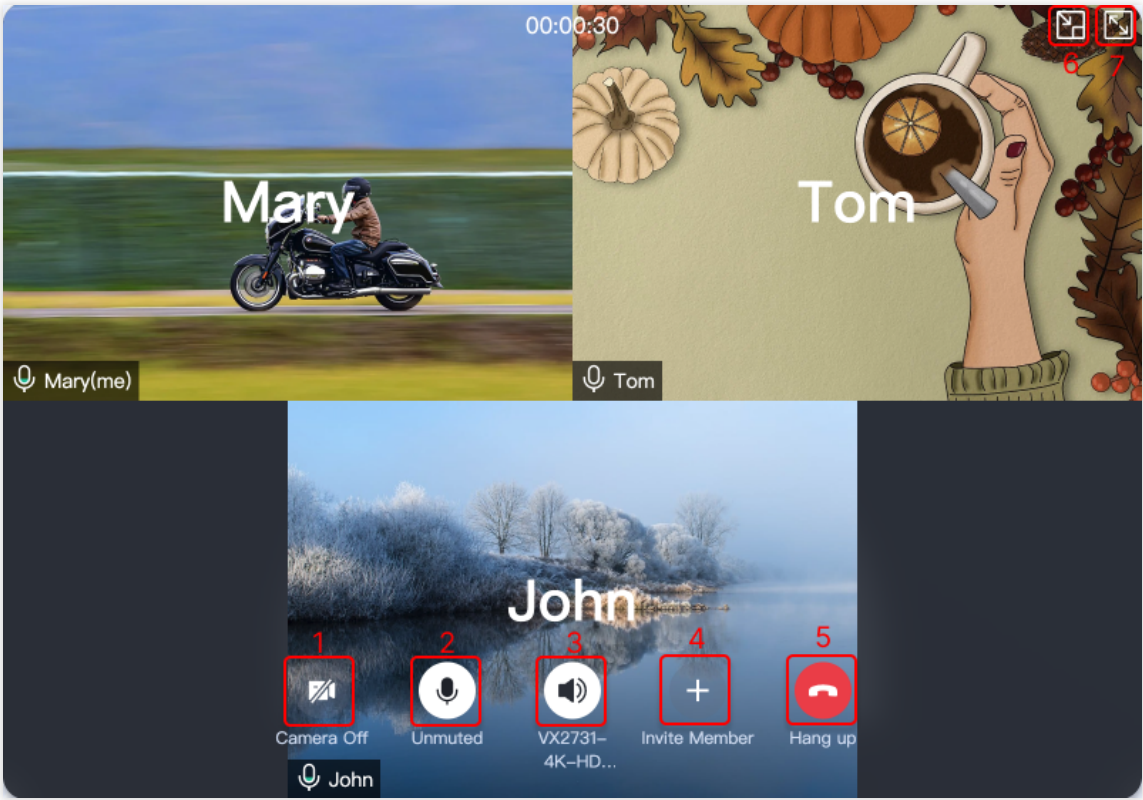
```
{
  "compilerOptions": {
    "experimentalDecorators": true
  }
}
```

##### 5. Modify the icon components in the TUICallKit/Components/assets folder

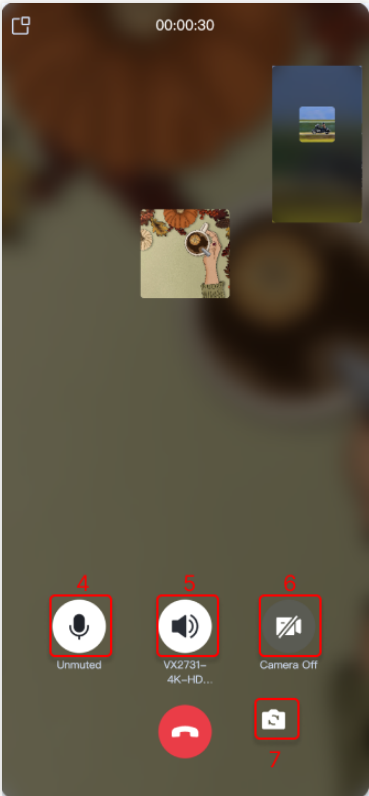
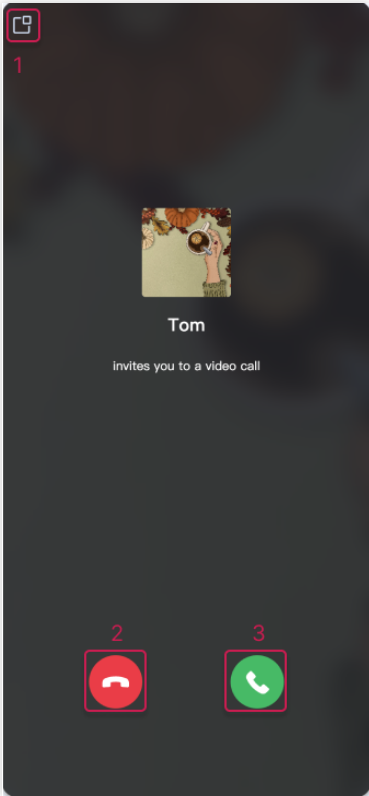
###### Note:

To ensure the icon color and style remain consistent throughout the application, please keep the icon file name unchanged when replacing.

Desktop  
Mobile



Serial number	Resource Path
1	/TUICallKit/Components/assets/button/camera-close.svg
2	/TUICallKit/Components/assets/button/microphone-open.svg
3	/TUICallKit/Components/assets/button/speaker-open.svg
4	/TUICallKit/Components/assets/button/desktop/inviteUser.svg
5	/TUICallKit/Components/assets/button/hangup.svg
6	/TUICallKit/Components/assets/button/desktop/minimize.svg
7	/TUICallKit/Components/assets/button/desktop/fullScreen.svg



Serial number	Resource Path



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

## Scheme 2: Custom UI Implementation

The features of `TUICallKit` are implemented based on the `TUICallEngine` SDK, which does not include UI elements. You can use `TUICallEngine` to implement your own UI. For detailed directions, refer to the documents below:

[TUICallEngine integration guide](#)

[TUICallEngine APIs](#)

# Flutter

Last updated : 2024-03-13 16:25:05

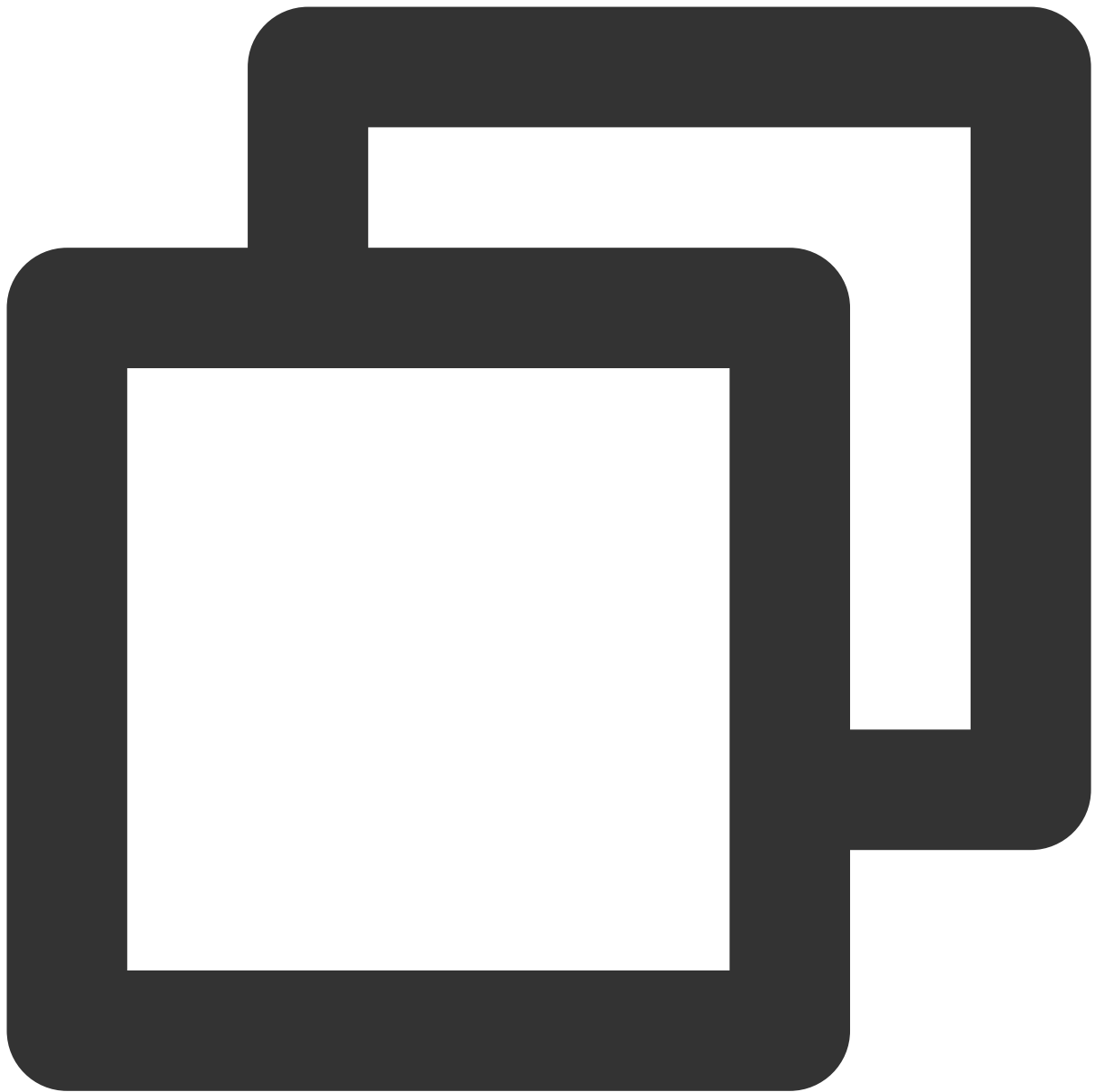
This article will introduce how to customize the user interface of TUICallKit. We provide two solutions for you to choose: **interface fine-tuning solution** and **self-implementation UI** solution.

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:

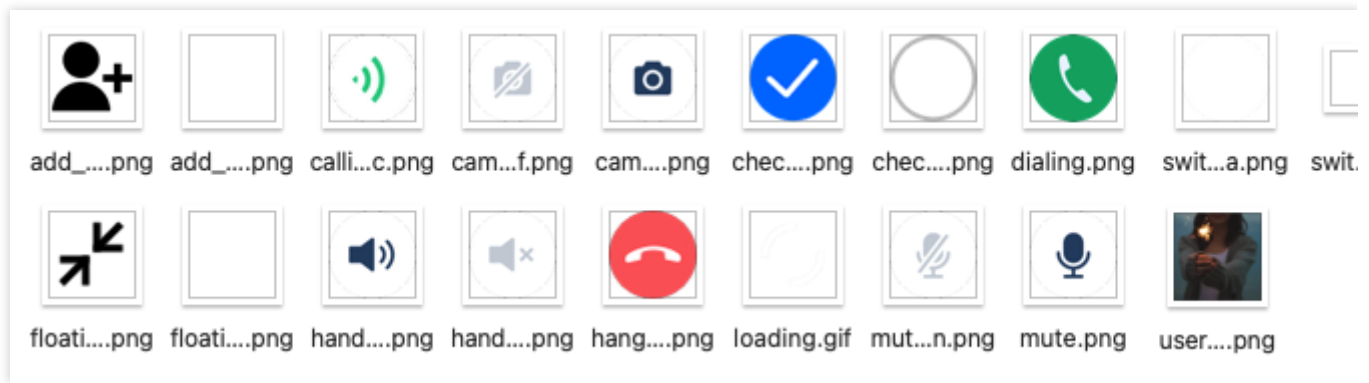
Under the **dependencies** node in the project **pubspec.yaml** file, add the **tencent\_calls\_uikit** plugin dependency, as shown below :



```
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.

# Additional Features(TUICallKit)

## Group Call

### Android&iOS&Flutter

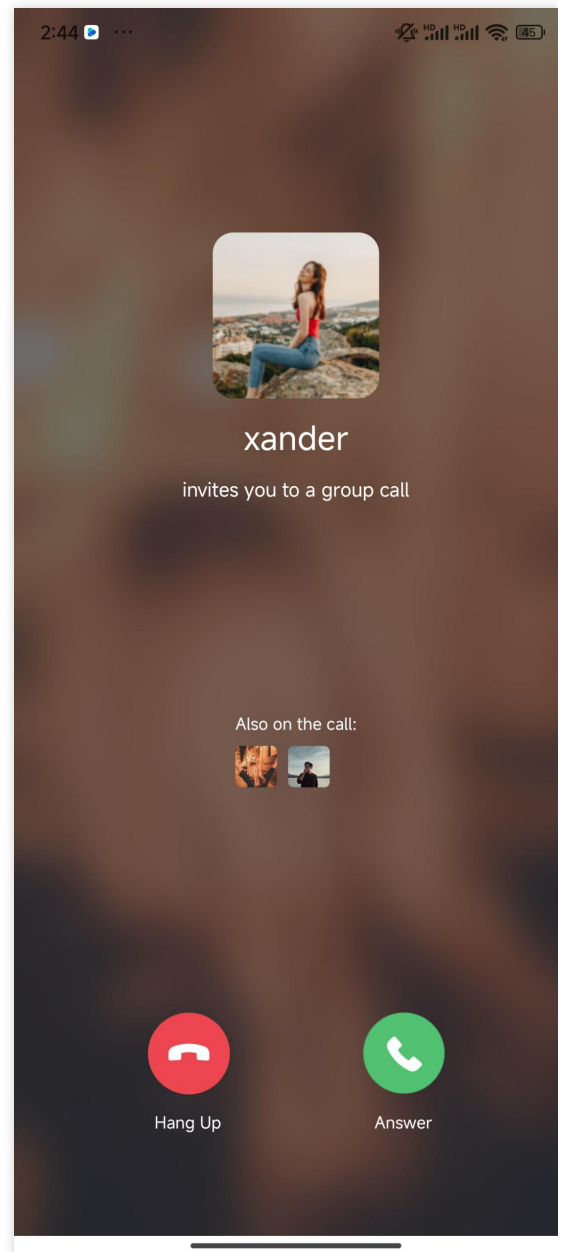
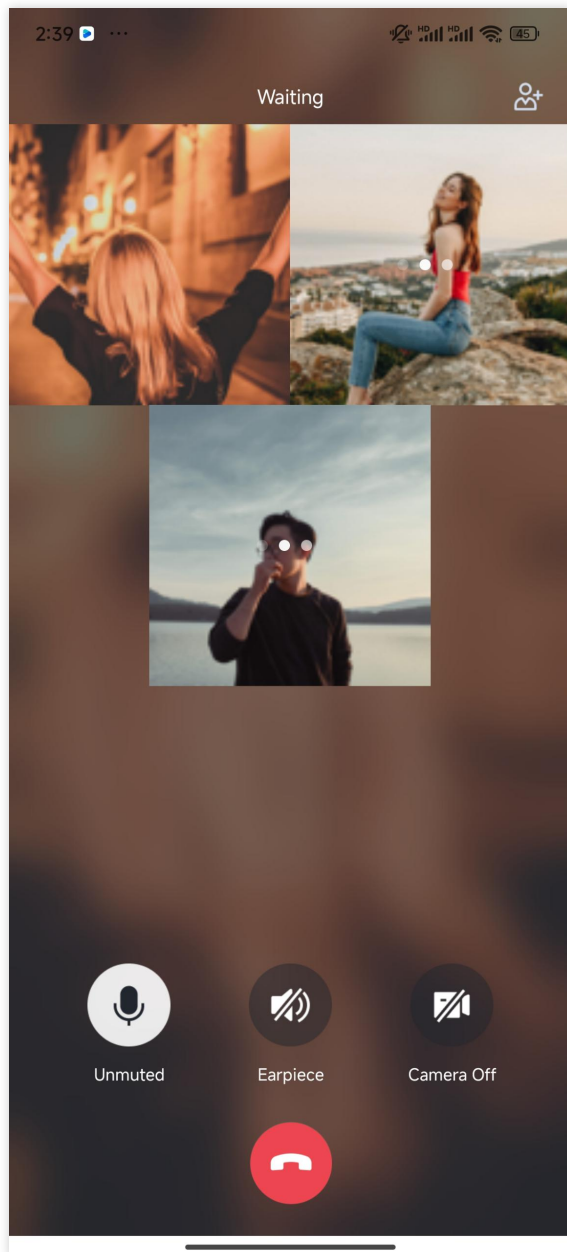
Last updated : 2024-05-08 11:37:24

This document describes how to use the group call feature, such as initiating a group call and joining a group call.

## Expected Outcome

TUICallKit supports group calls. See the expected outcome in the image below.

Initiate a Group Call	Receive a Group Call Request



## Create groupID

Before using the group call feature, you need to create a group first and initiate a group call in an existing group.

Method 1: Create a group by calling the IM API. See [IM group management](#).

Method 2: Manually create a group through the console. See [Console group management](#).

## Group Call

### Initiate a Group Call

Call the groupCall API to initiate a group call.

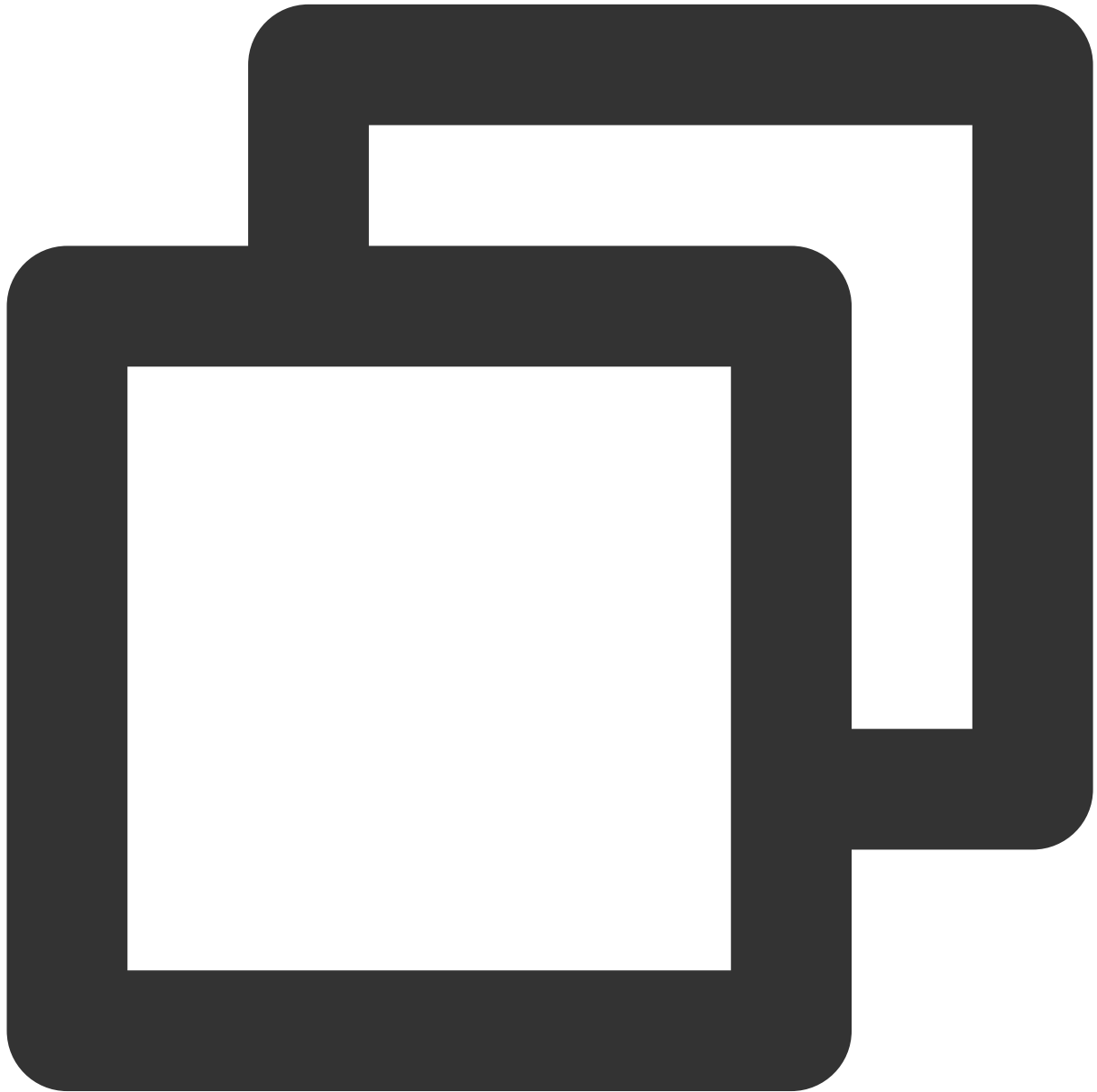
Android(Kotlin)

Android(Java)

iOS(Swift)

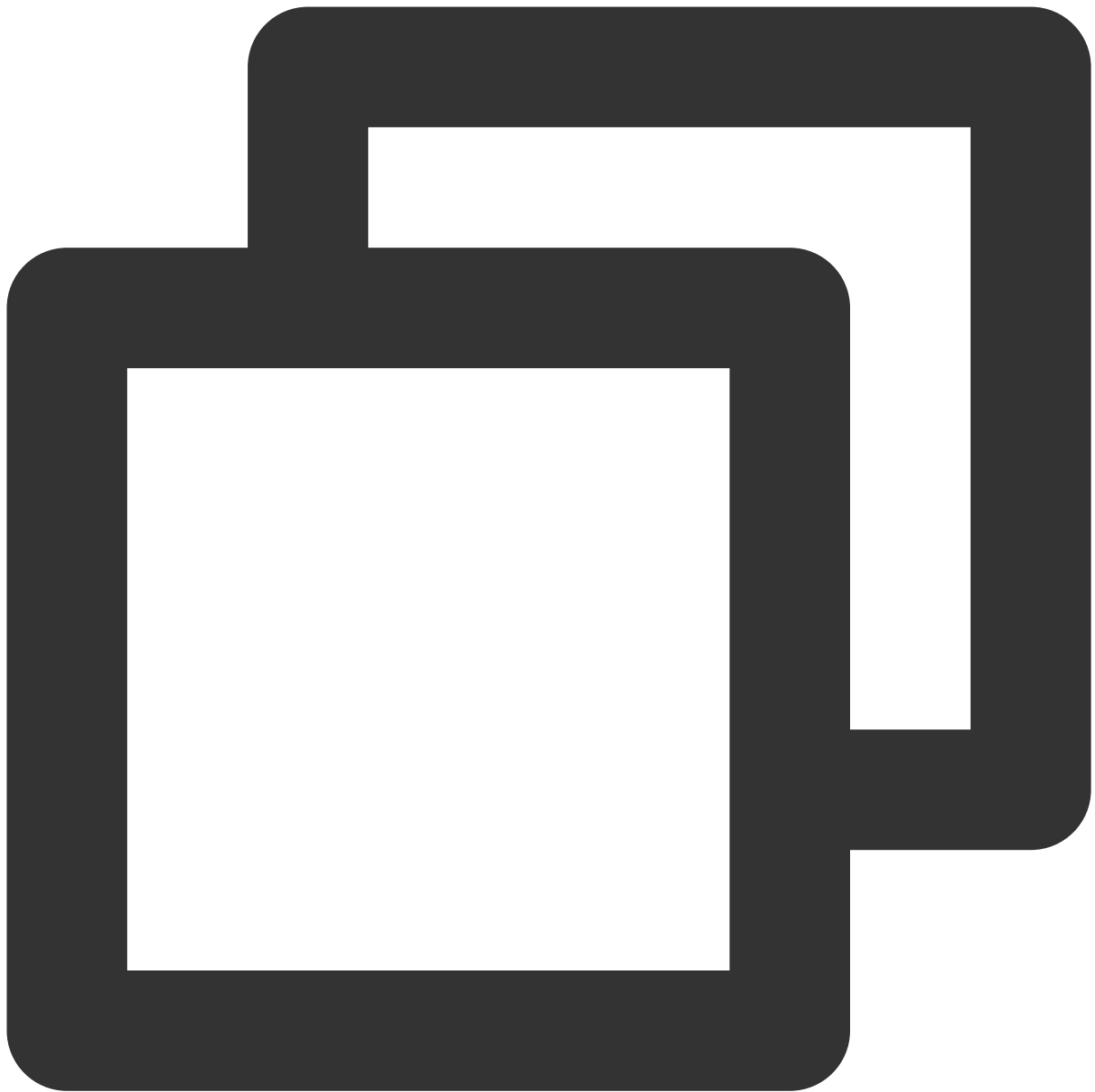
iOS(Objective-C)

Flutter(Dart)

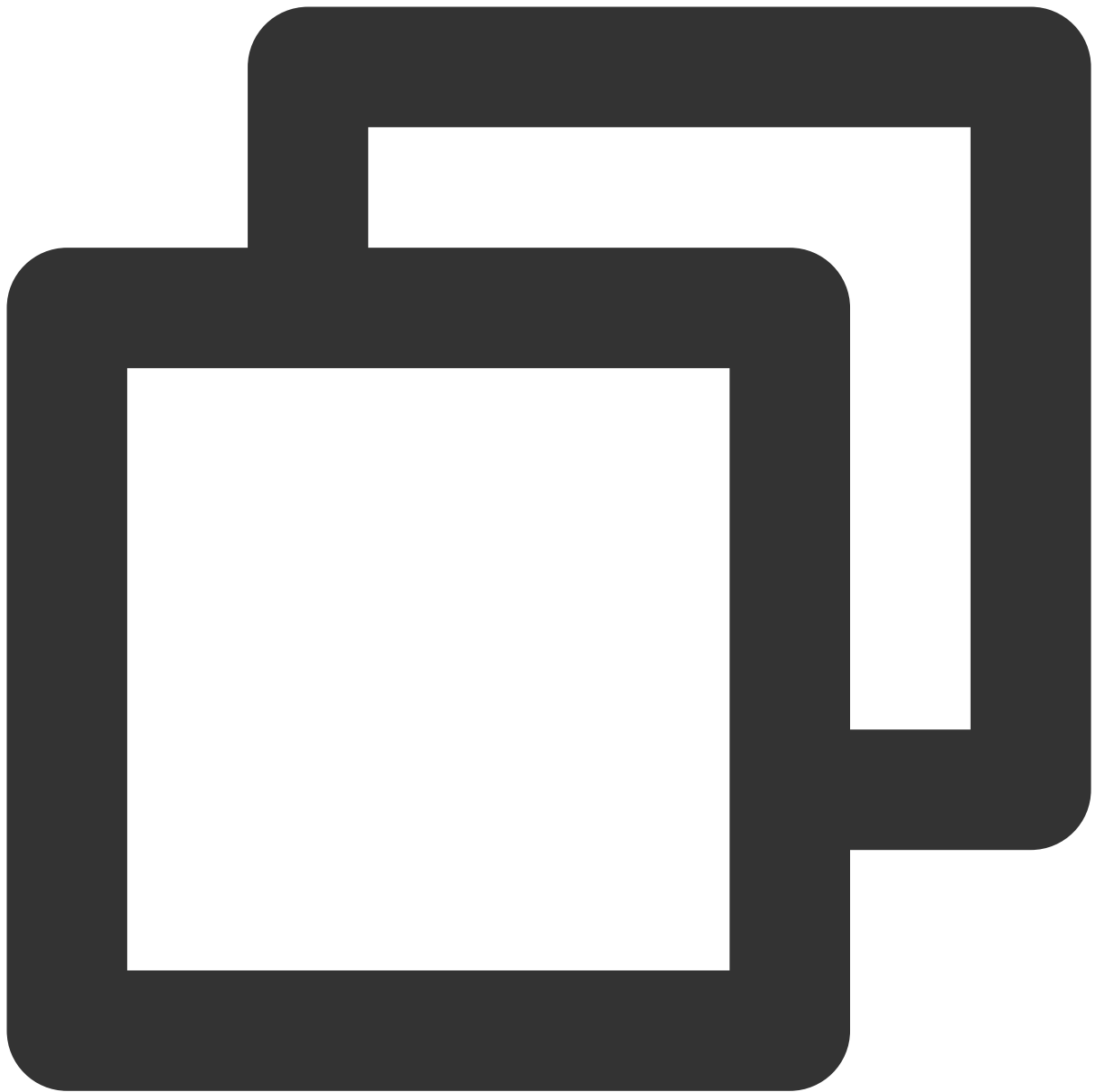


```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mik
```



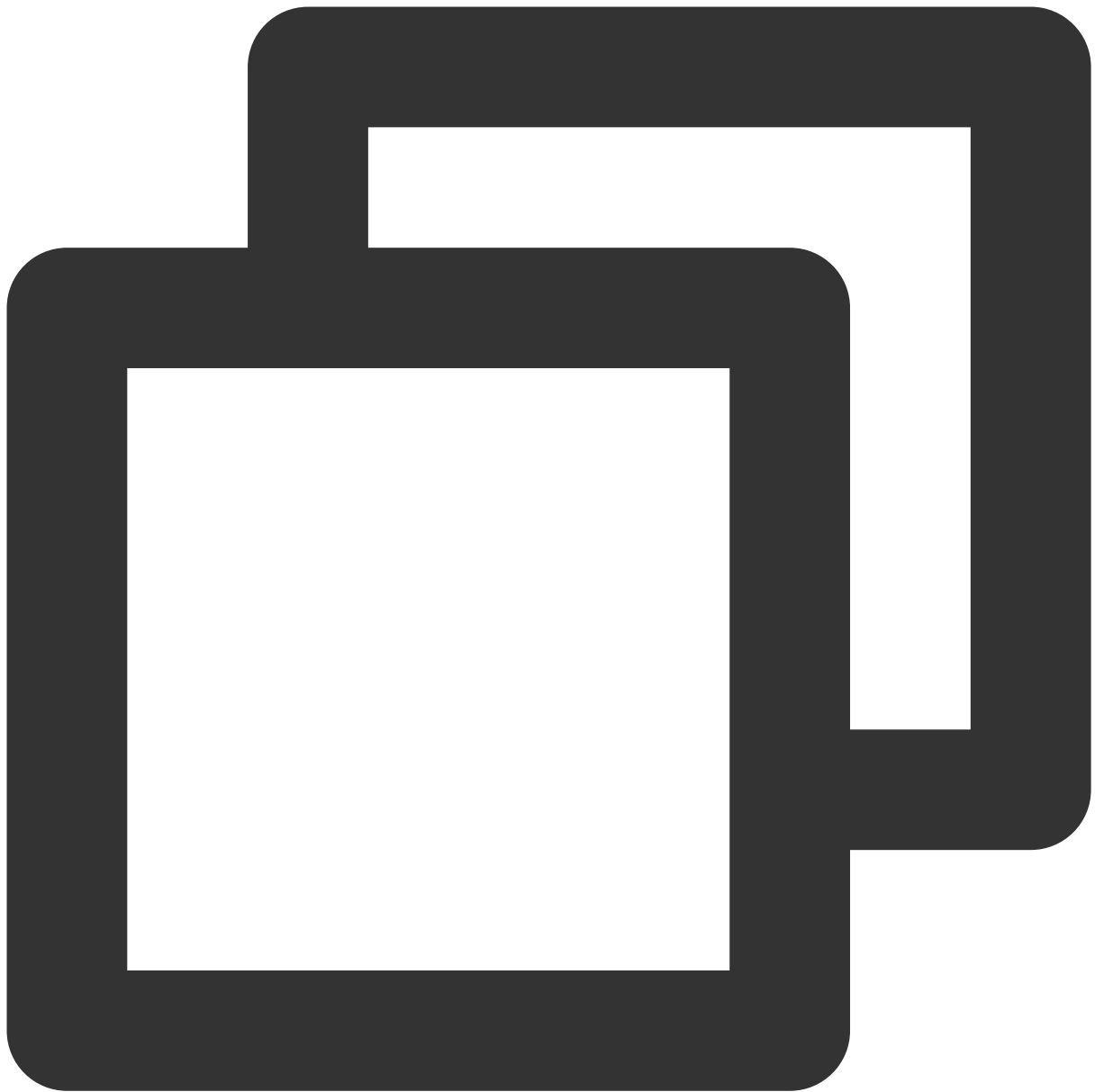


```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mik
```



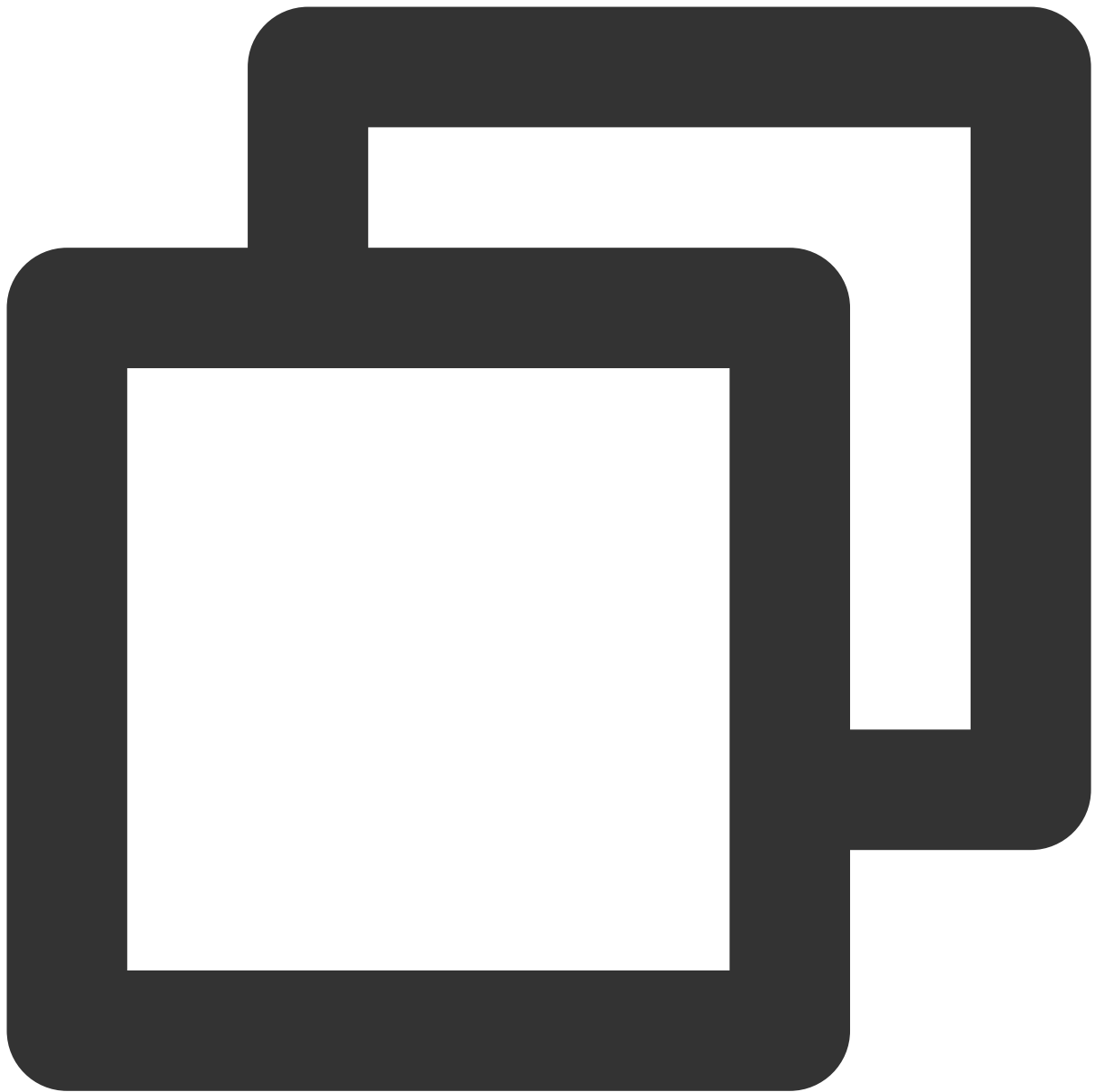
```
import TUICallKit

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



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

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



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

## Join a Group Call

Call the `joinInGroupCall` API to actively join an existing audio or video call in the group.

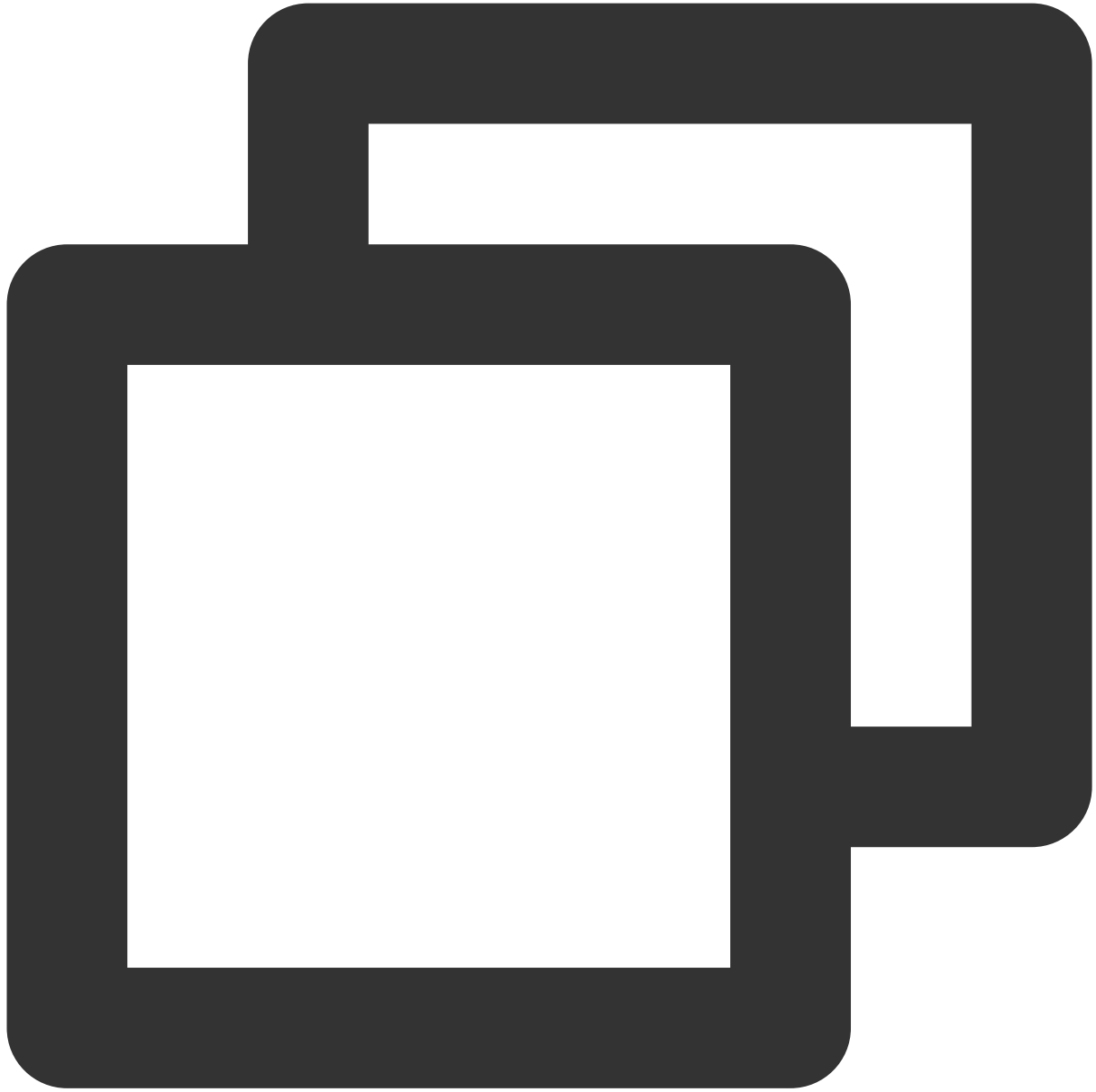
Android(Kotlin)

Android(Java)

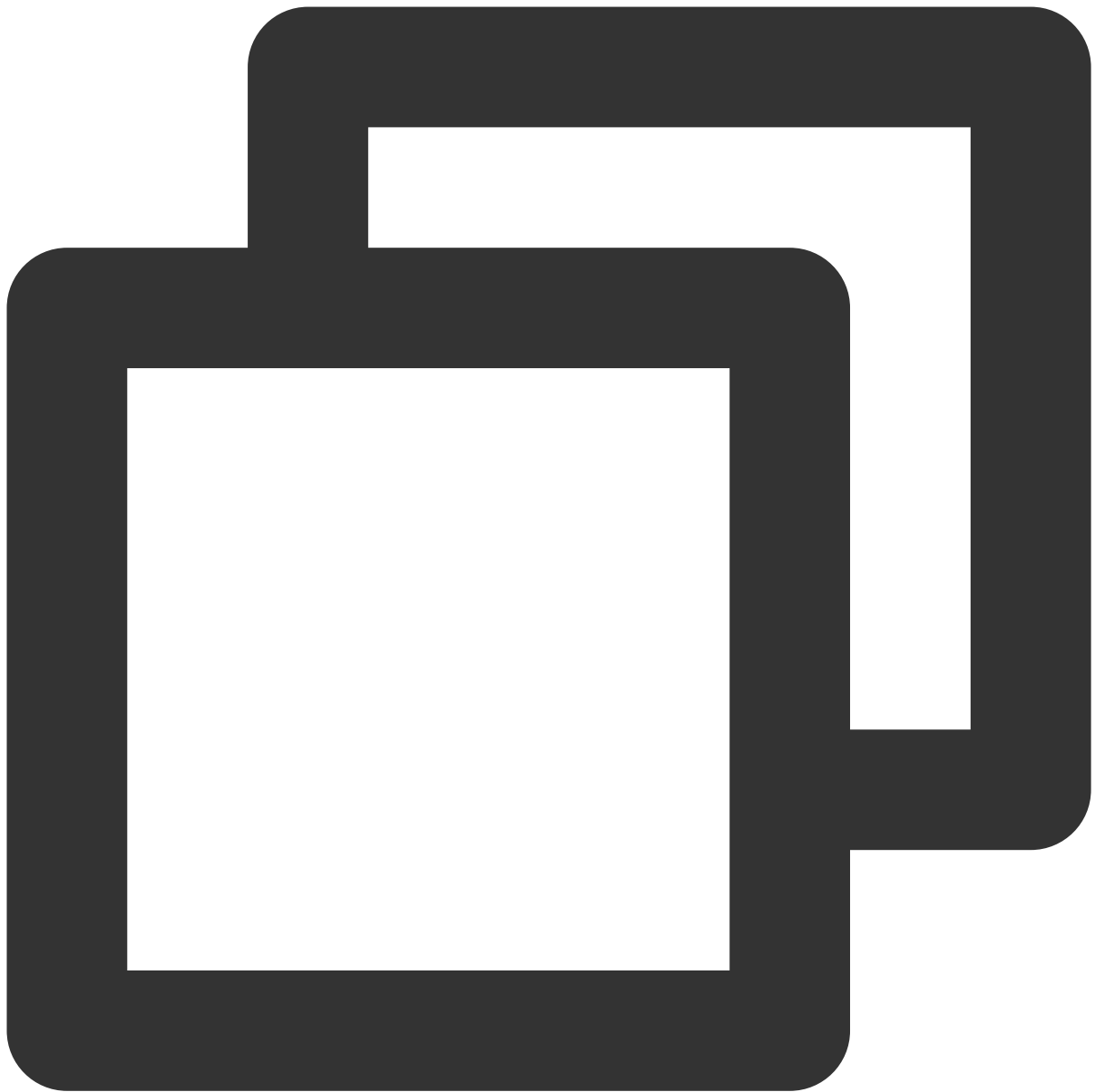
iOS(Swift)

iOS(Objective-C)

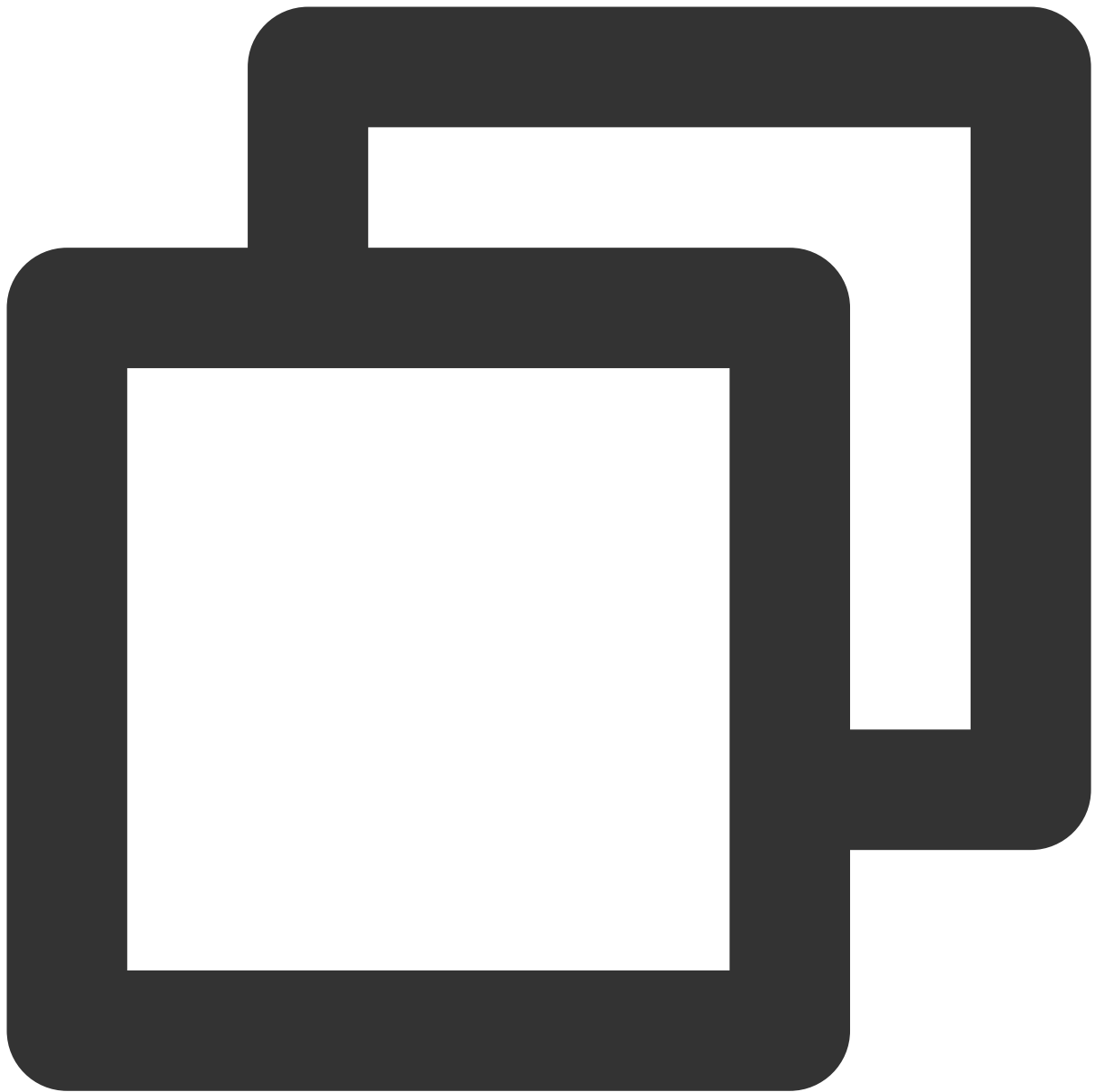
Flutter(Dart)



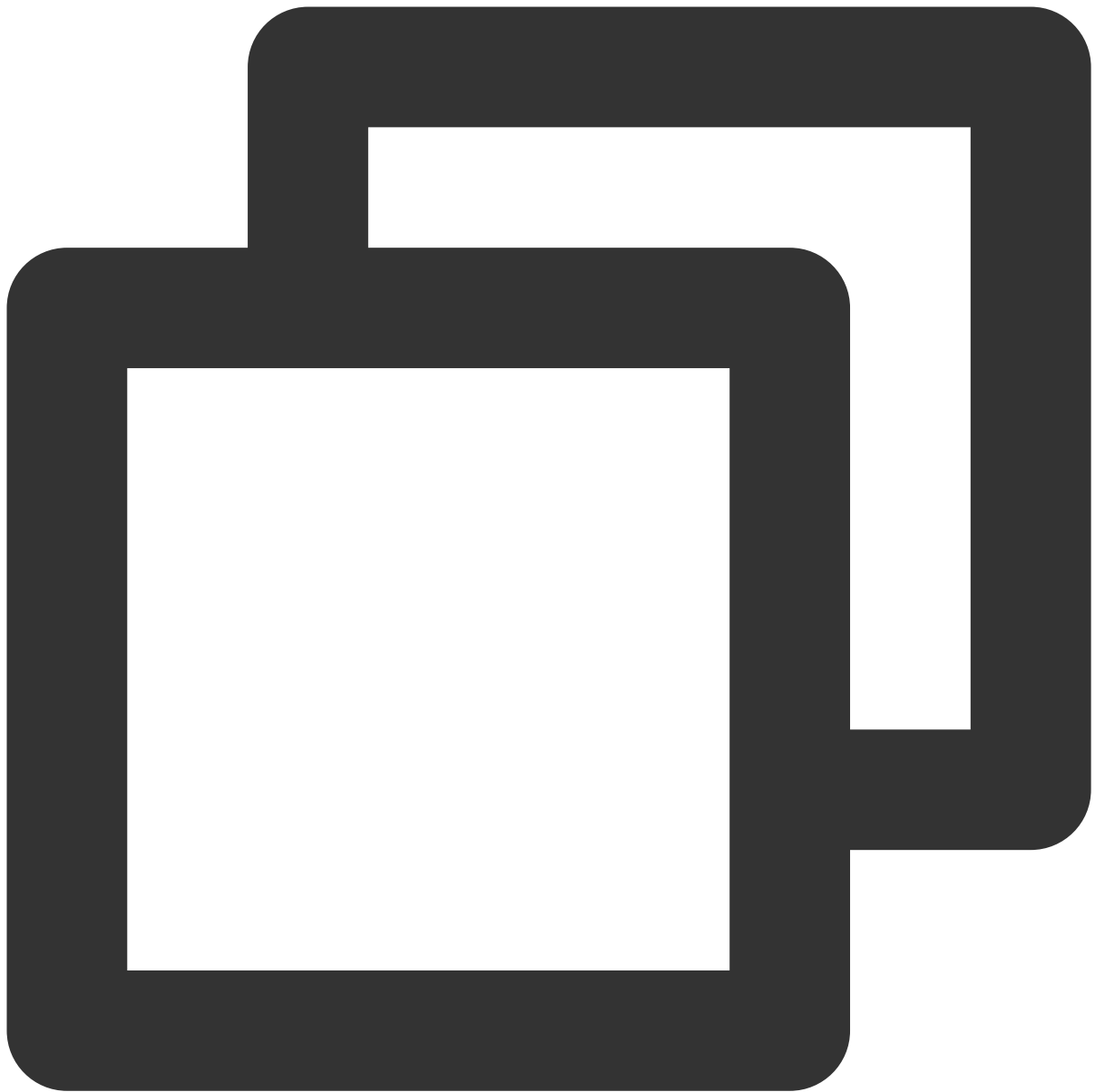
```
RoomId roomId = RoomId();  
roomId.intRoomId = 123321;  
TUICallKit.createInstance(context).joinInGroupCall(roomId, "12345678", TUICallDefin
```



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



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

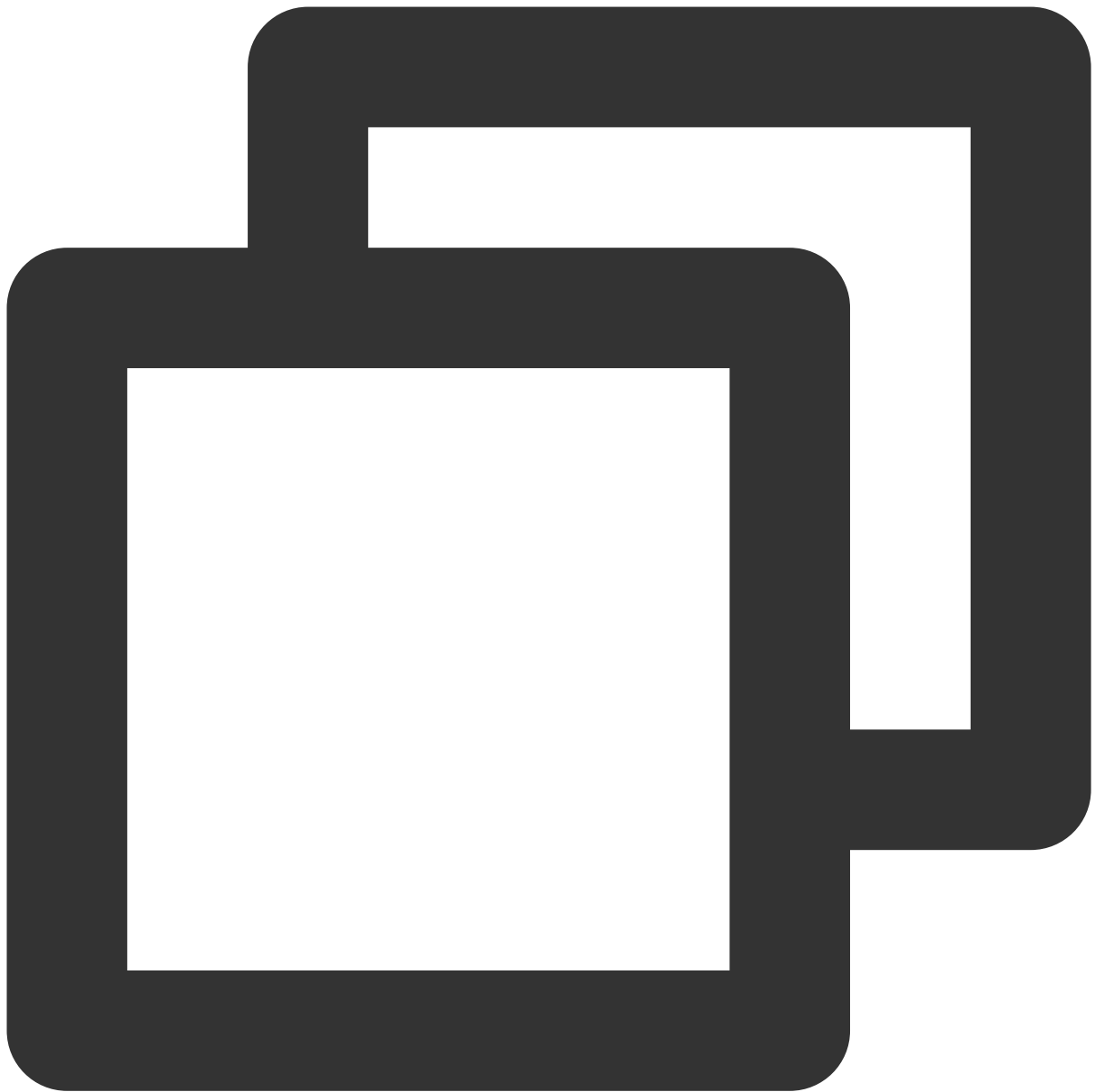


```
#import <TUICallKit/TUICallKit.h>

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

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





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

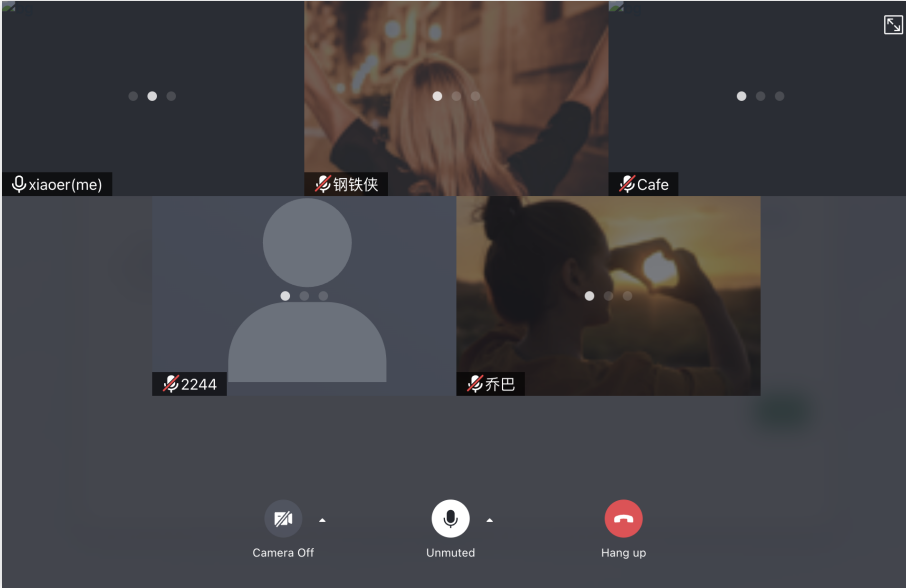
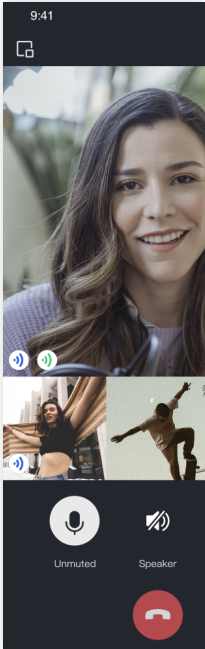
# Web&H5

Last updated : 2024-05-08 11:37:24

This document describes how to use the group call feature, such as initiating a group call and joining a group call.

## Expected outcome

TUICallKit supports group calls. See the expected outcome in the image below.

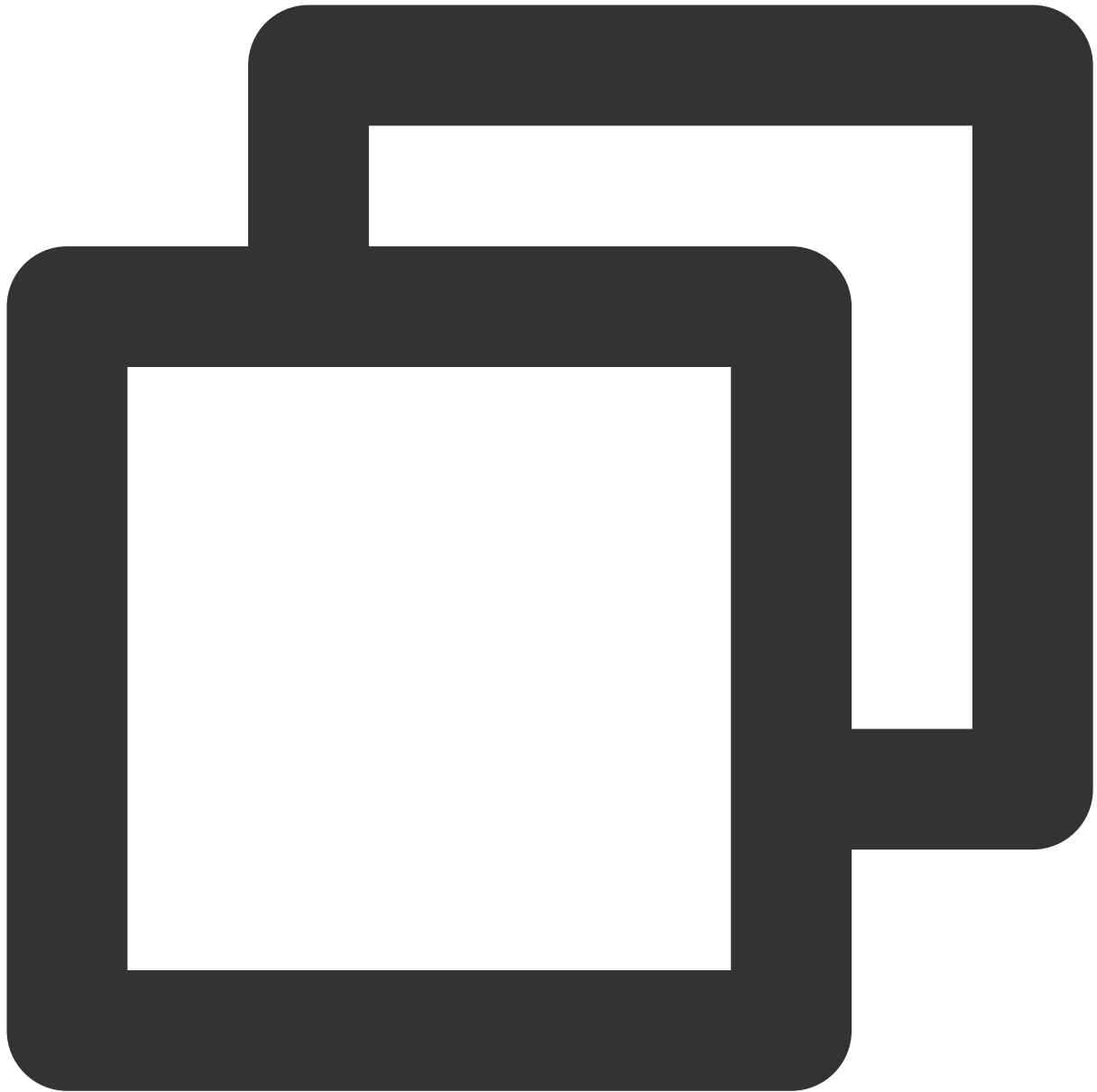
Web	Mobile Client
	

## Create groupID

Before using the group call feature, you need to create a group first and initiate a group call in an existing group.

Method 1: Create a group by calling the [IM API](#), see [IM Group Management](#) for details.

Method 2: Manually create a group through the console. See [Console group management](#).



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

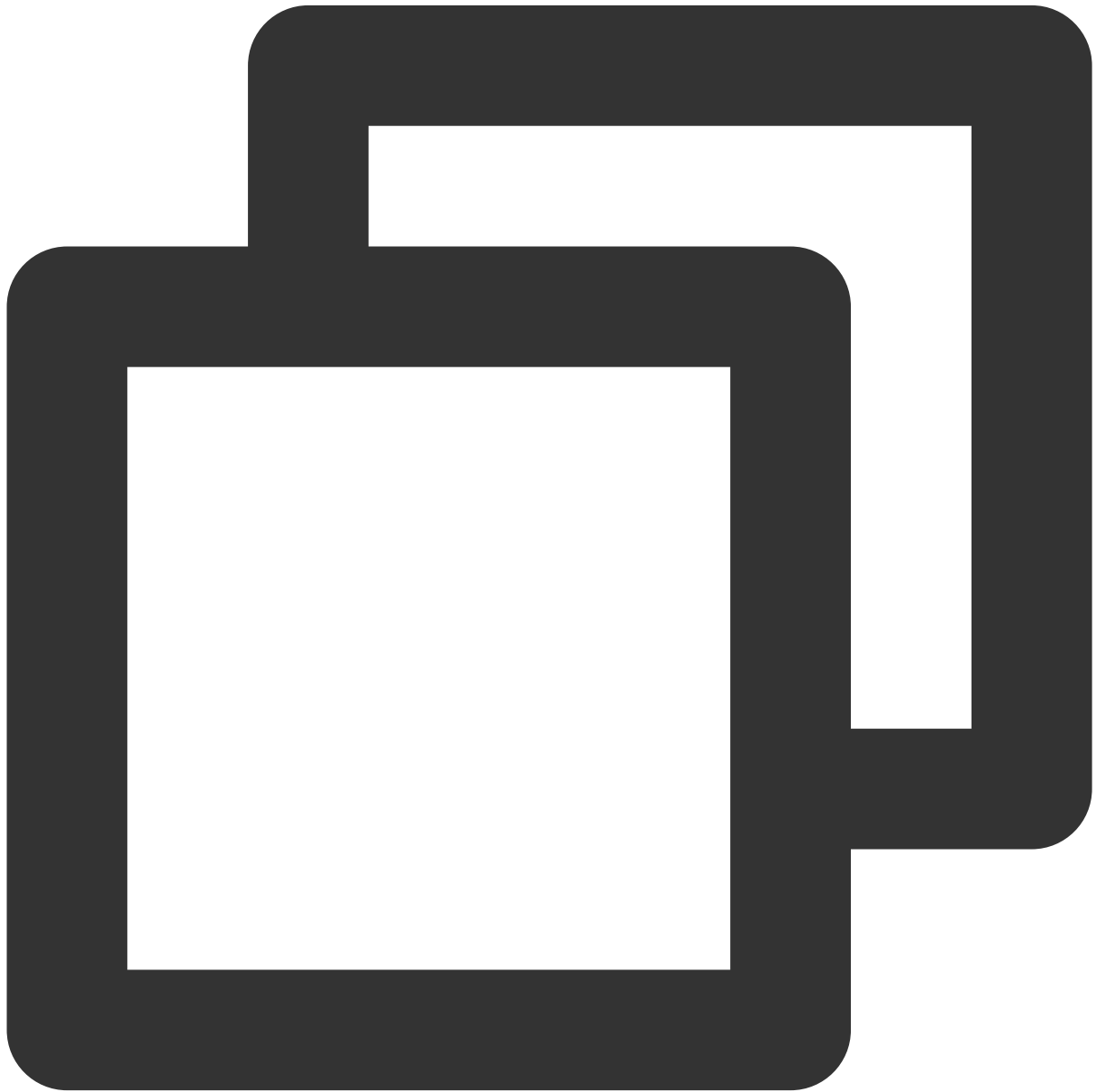
const userIDList: string[] = ['user1', 'user2'];
async function createGroupID() {
  const tim = TIM.create({ SDKAppID });
  const memberList: any[] = userIDList.map(userID => ({ userID: userID }));
  const res = await tim.createGroup({
    type: TIM.TYPES.GRP_PUBLIC, // Must be a public group
    name: 'WebSDK',
    memberList
  });
};
```

```
return res.data.group.groupID;  
}
```

## Group Call

### Initiate a Group Call

Call the `groupCall` API to initiate a group call.



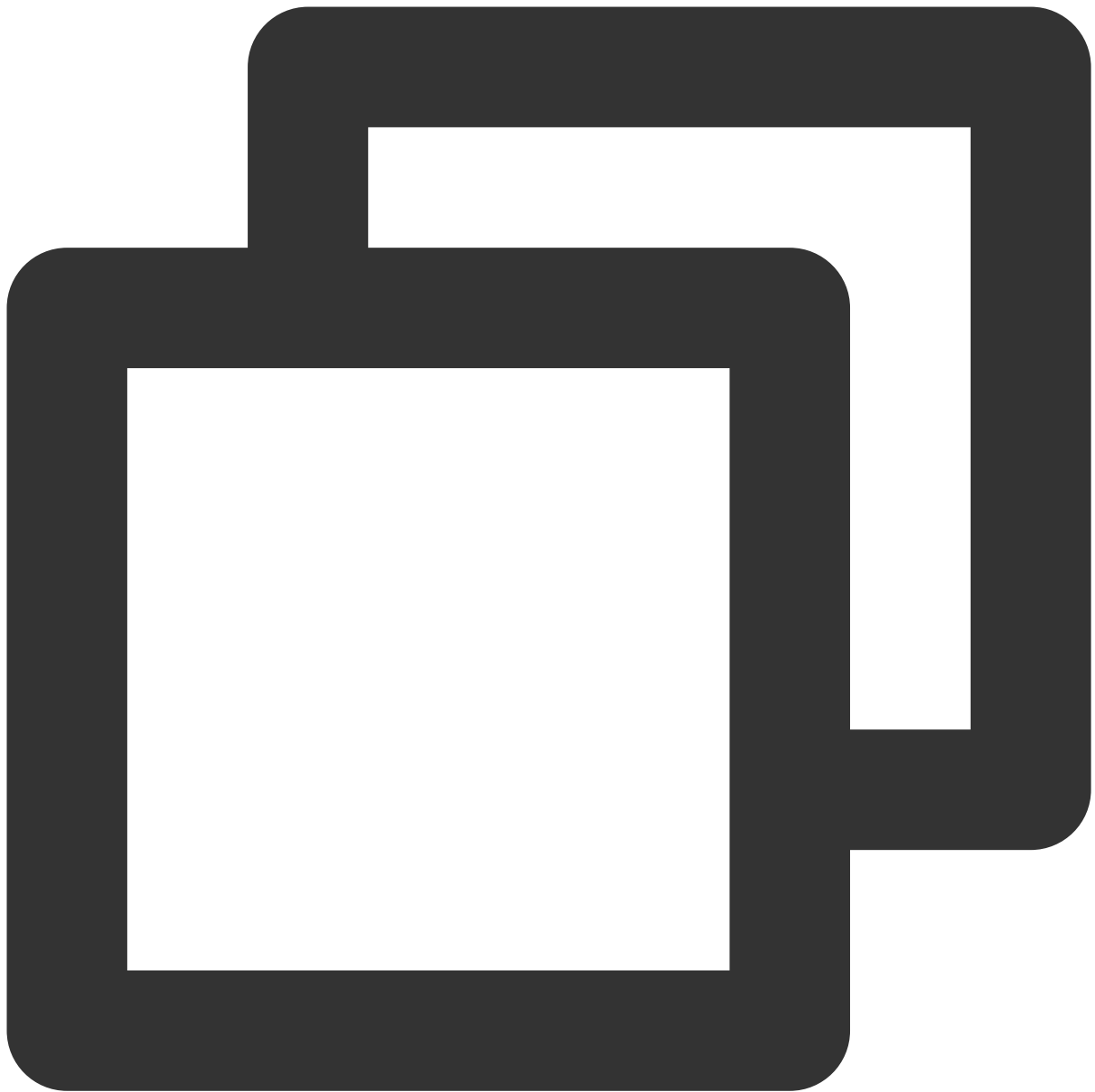
```
import { TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-react";
// Replace it with the call-uikit npm package you are currently using

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

## Join a Group Call

Call the `joinInGroupCall` API to actively join an existing audio or video call in the group.

**Note:**Vue v3.1.2 or later versions are supported



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

try {
  const params = {
    type: TUICallType.VIDEO_CALL,
    groupID: "xxx",
    roomID: 0,
  };
  await TUICallKitServer.joinInGroupCall(params);
} catch (error: any) {
```

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

# Setting Nickname and Avatar (Full Platform)

Last updated : 2024-05-09 14:25:09

This document describes how to set the user's avatar and nickname.

## Setting Avatar and Nickname

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

Android (Kotlin)

Android (Java)

iOS (Objective-C)

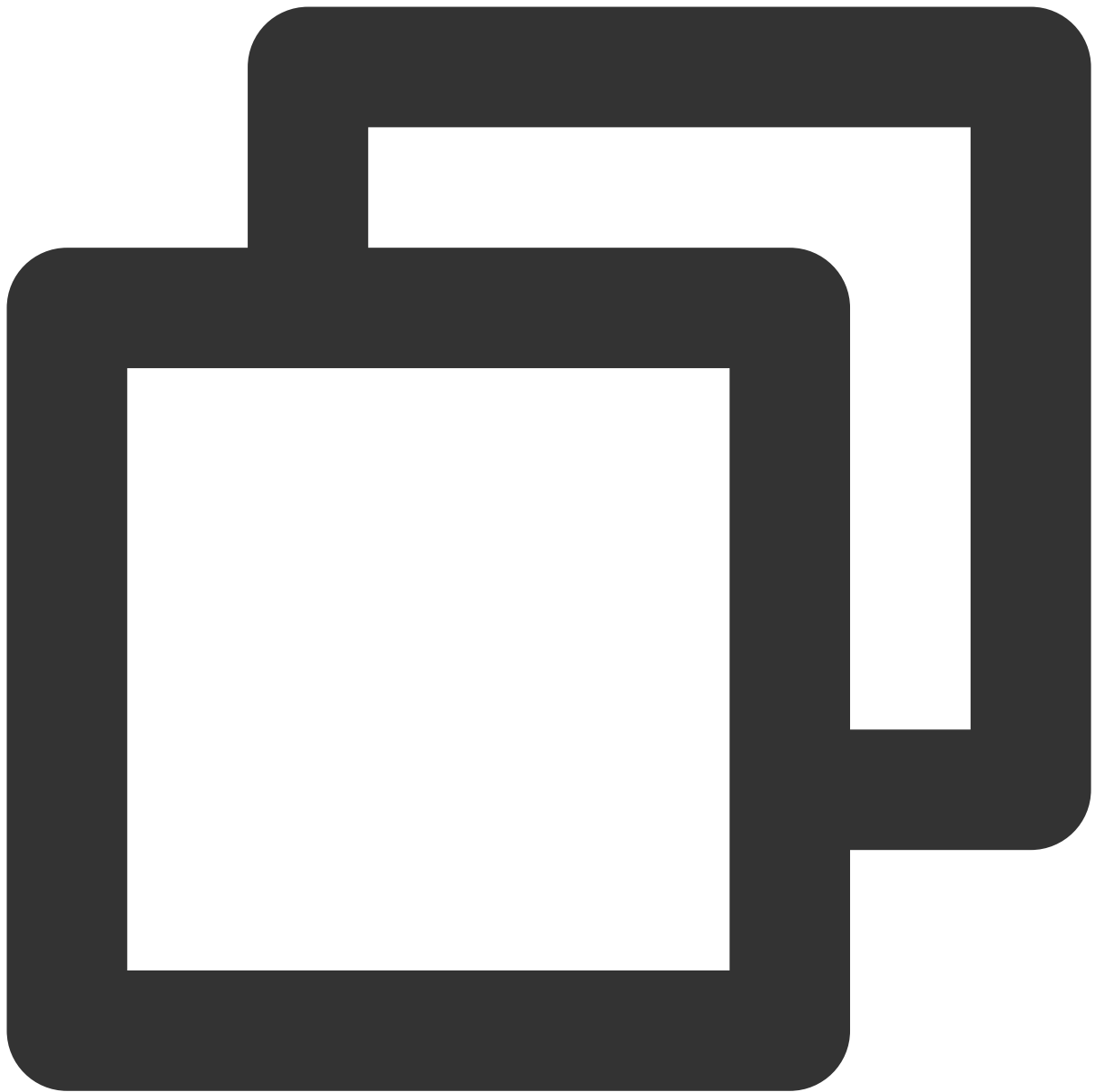
iOS (Swift)

Flutter (Dart)

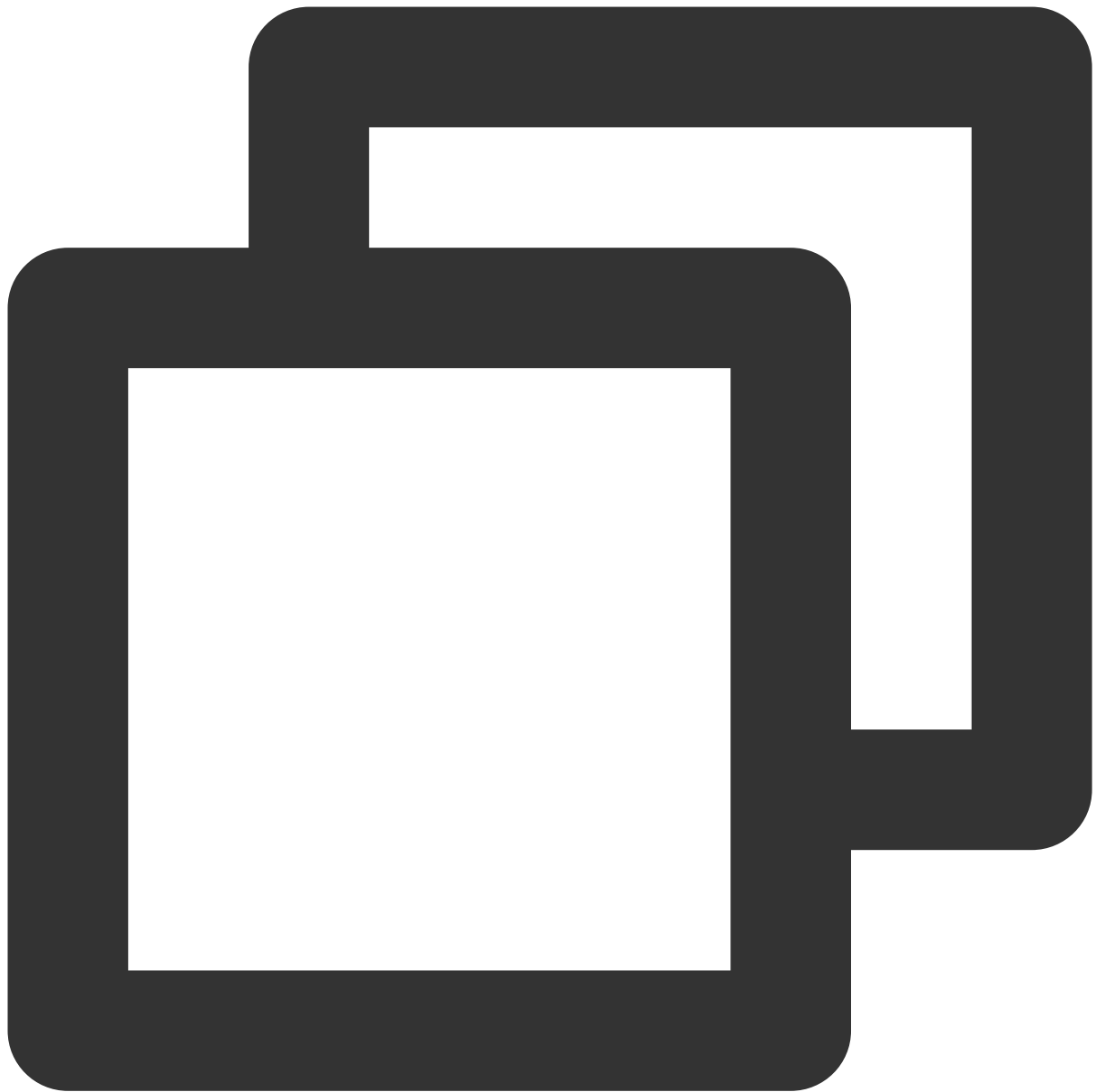
uni-app (Andorid&iOS)

Web&H5

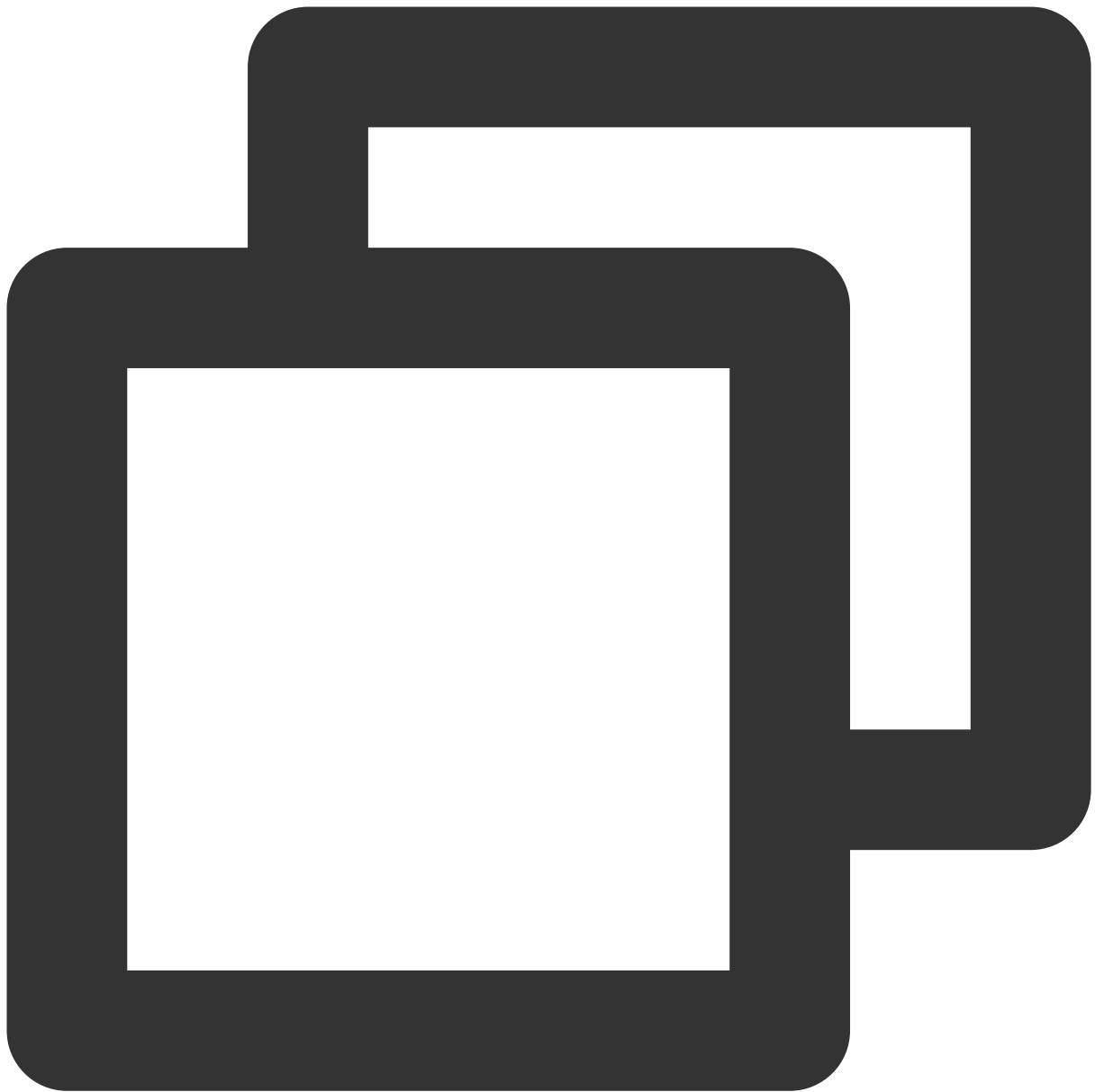




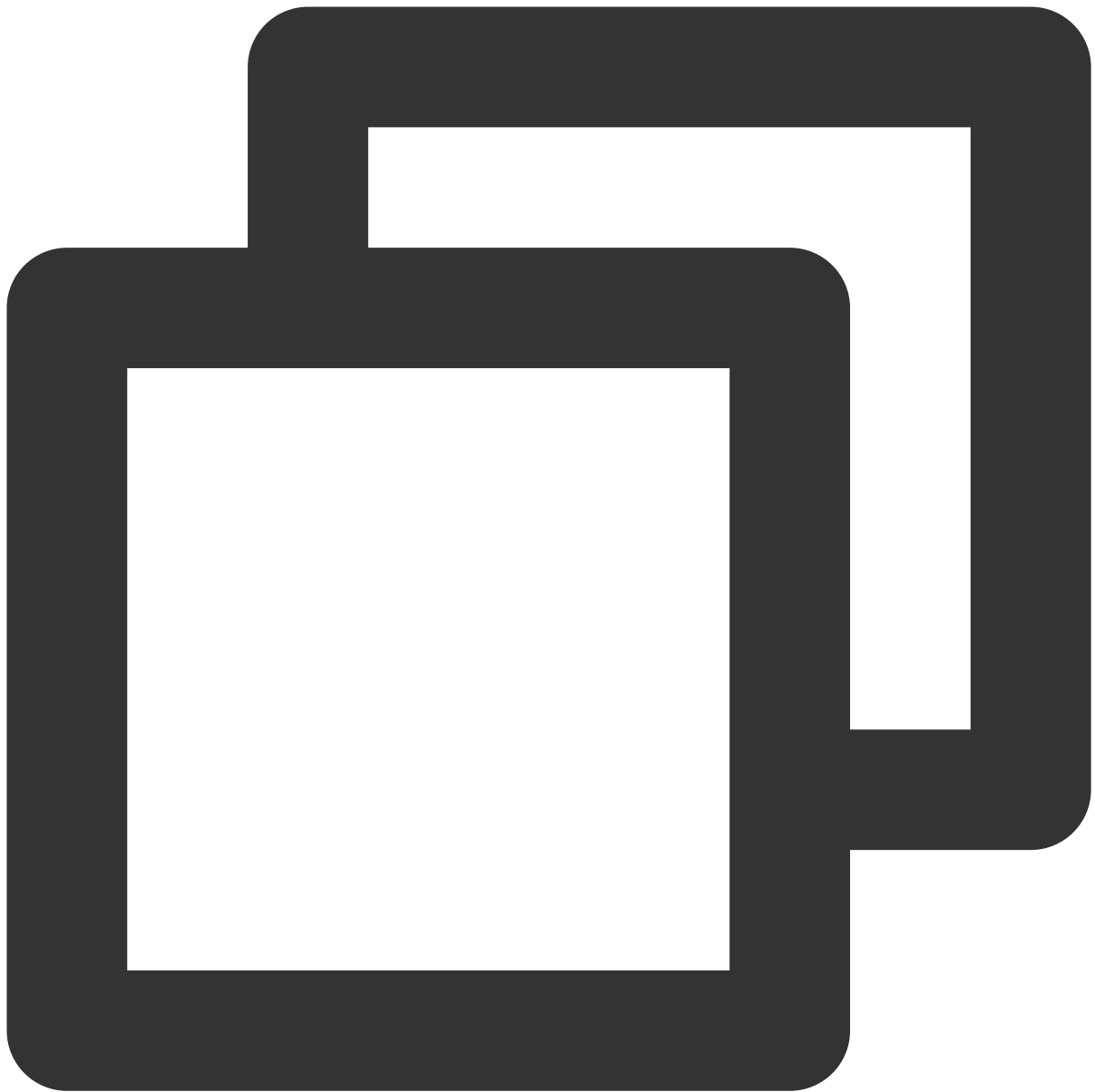
```
TUICallKit.createInstance(context).setSelfInfo("jack", "https://****/user_avatar.png")
```



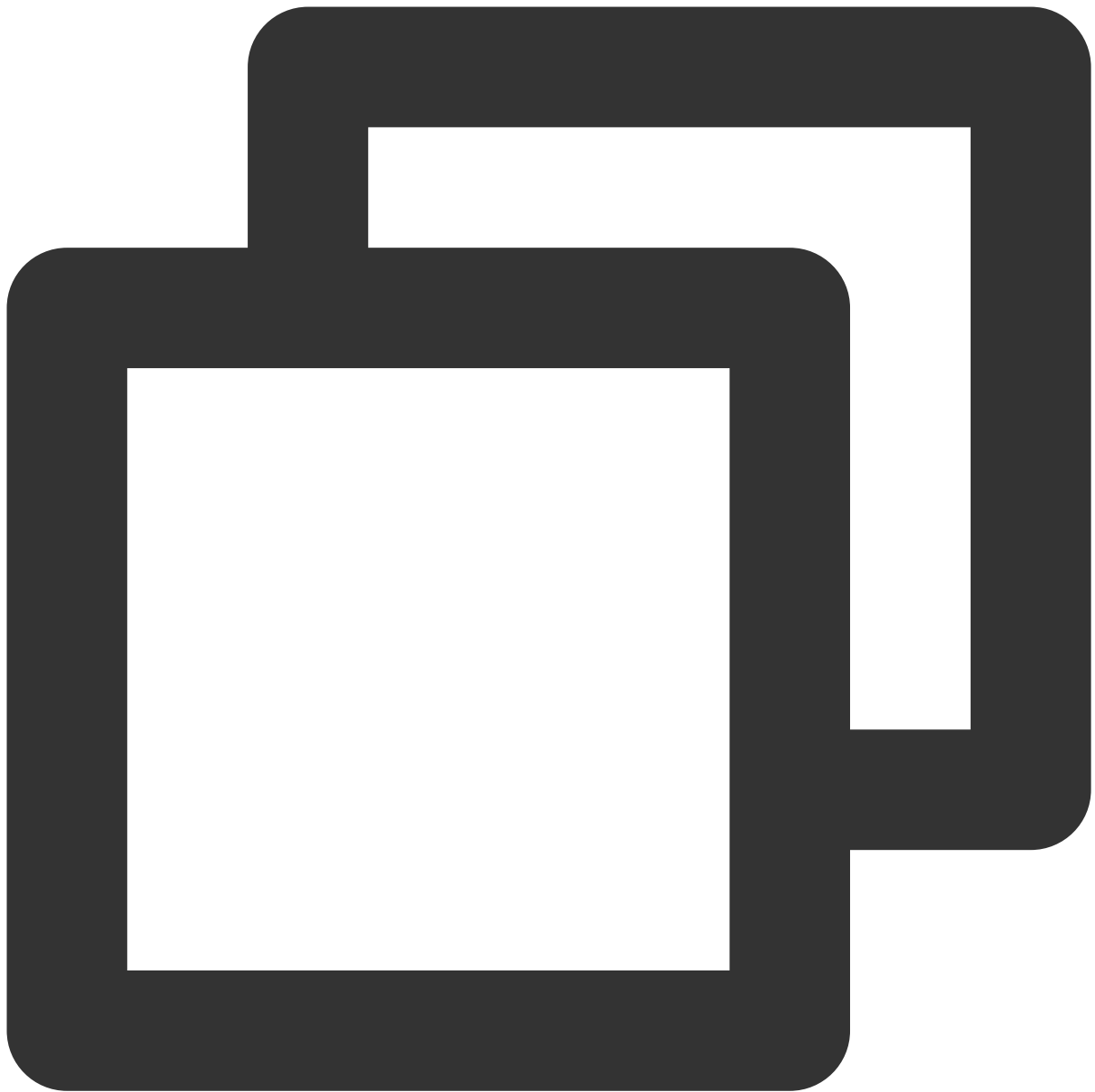
```
TUICallKit.createInstance(context).setSelfInfo("jack", "https://****/user_avatar.png")
```



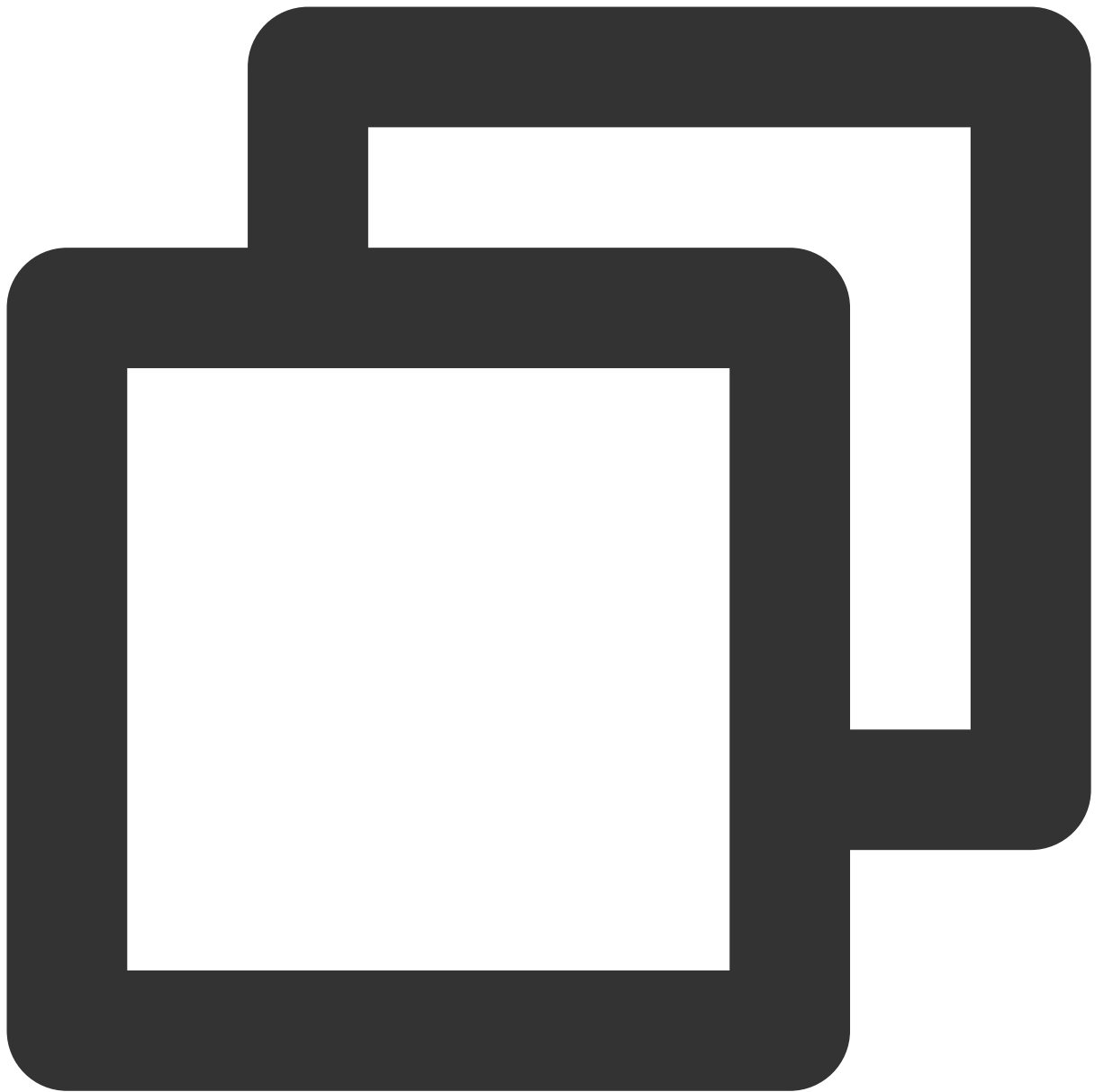
```
[[TUICallKit sharedInstance] setSelfInfo:@"" avatar:@"" succ:^(  
    } fail:^(int code, NSString *errMsg) {  
    }];
```



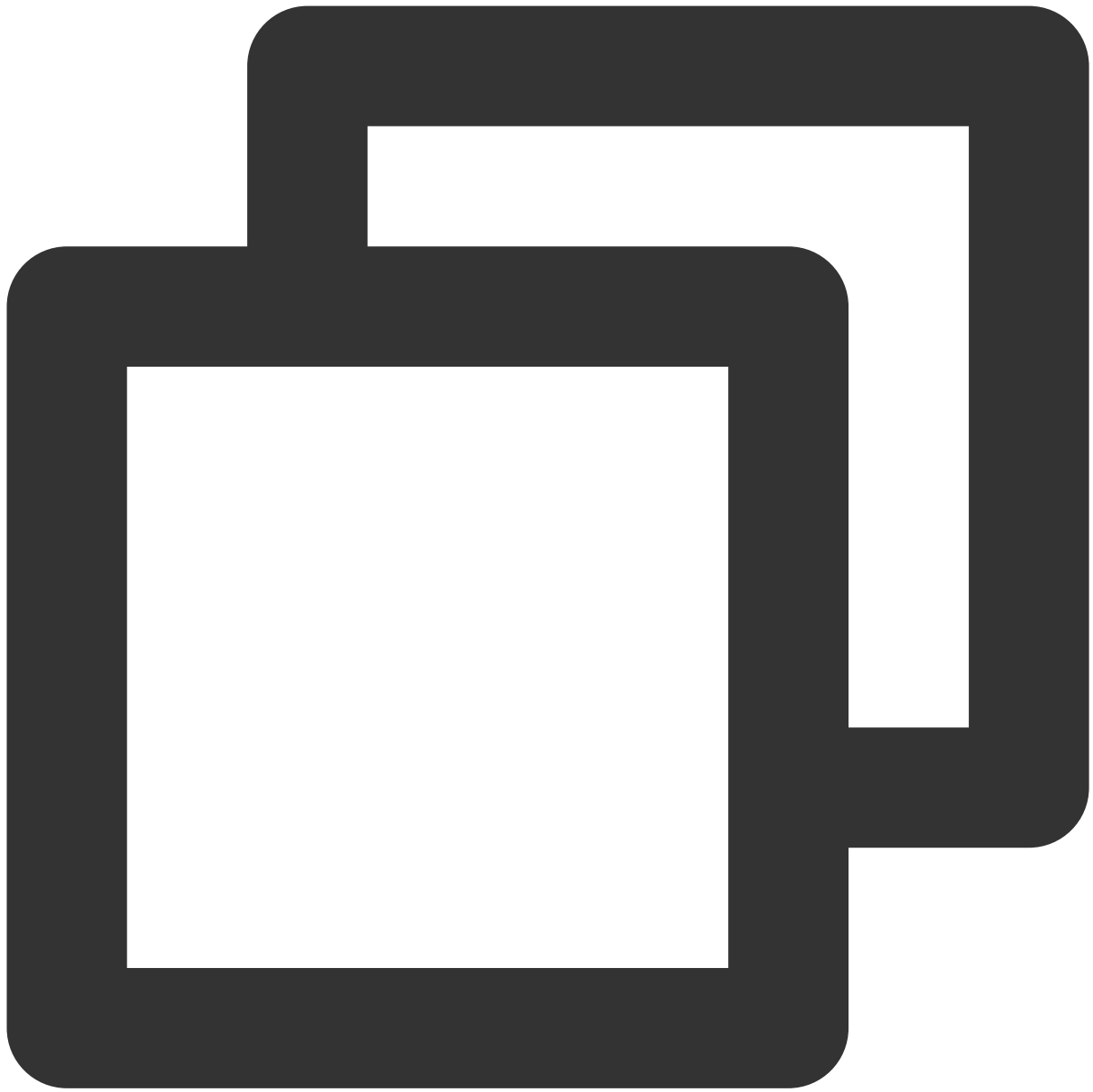
```
TUICallKit.createInstance().setSelfInfo(nickname: "", avatar: "") {  
    } fail: { code, message in  
    }  
}
```



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



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



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

**Note**

The update of the callee's nickname and avatar 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.



# Setting Resolution and Fill Pattern

Last updated : 2024-05-08 11:37:25

This document describes how to set resolution and fill mode.

## Setting Resolution and Fill Mode

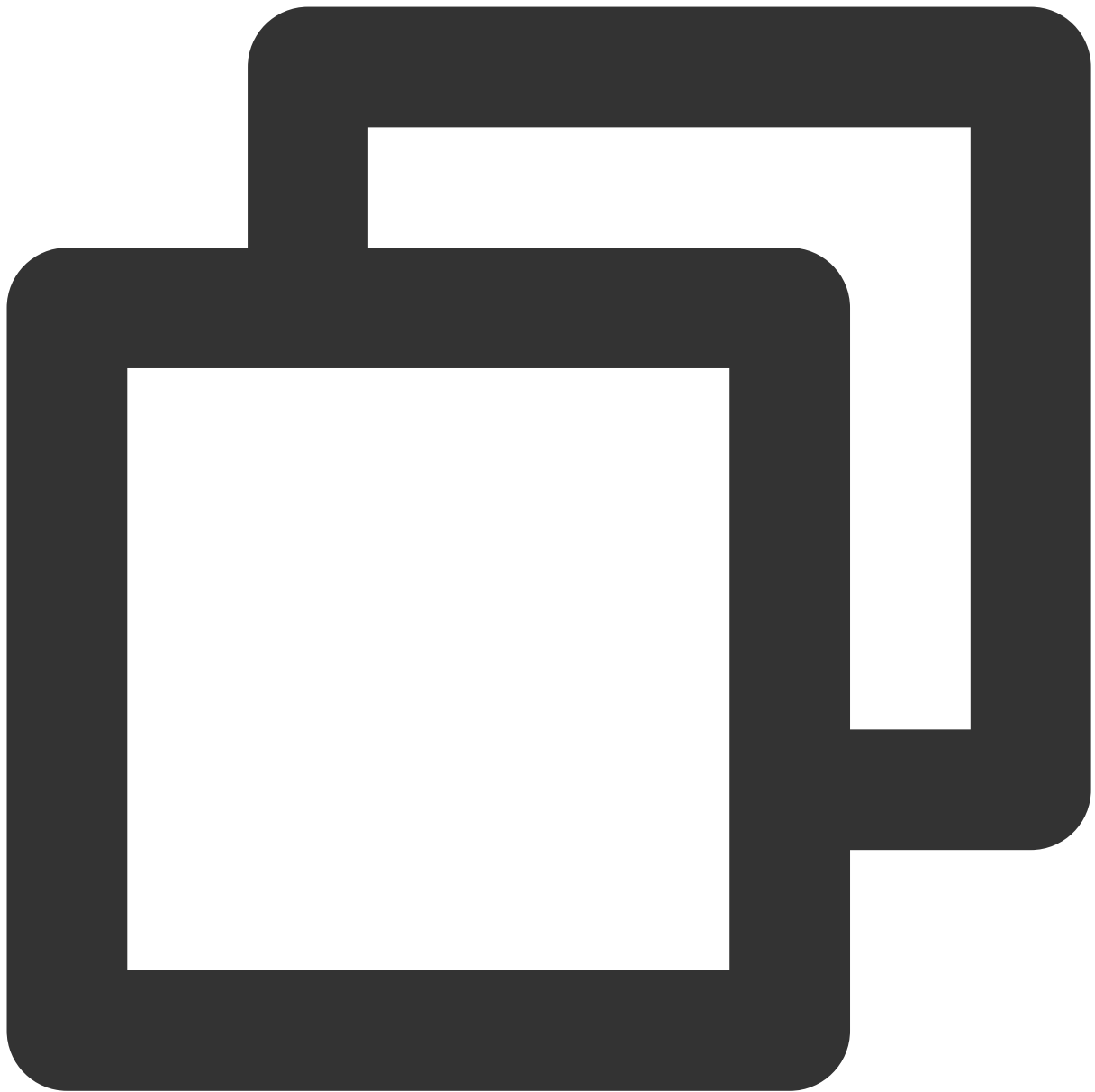
The TUICallKit component offers several feature toggles, which can be enabled or disabled as needed. For details, see [TUICallKit Property Overview](#).

videoDisplayMode: Display mode.

videoResolution: Call resolution.

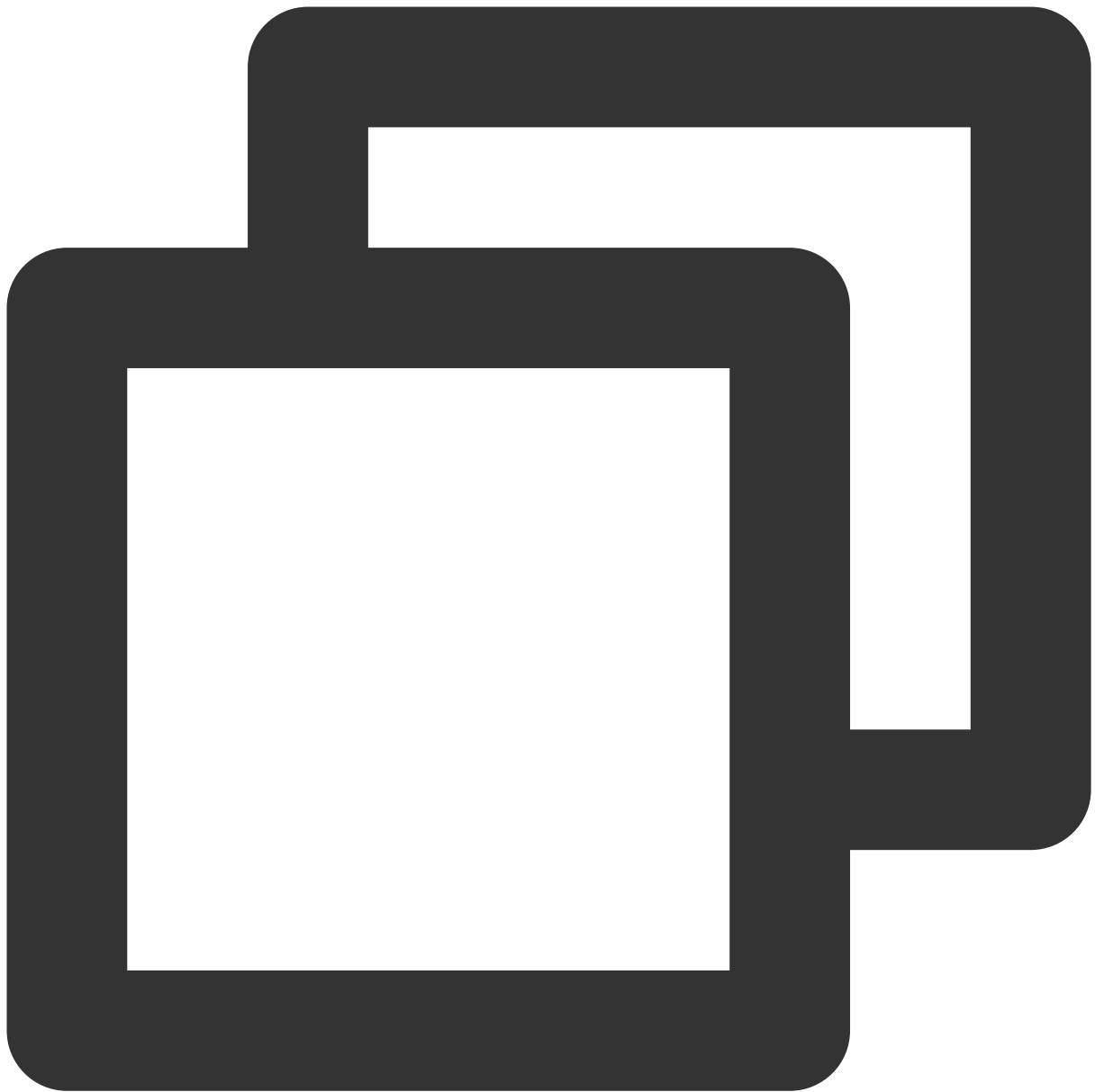
React

Vue



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

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



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

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

## videoDisplayMode

The display mode `videoDisplayMode` attribute has three values:

`VideoDisplayMode.CONTAIN`

`VideoDisplayMode.COVER`

`VideoDisplayMode.FILL`. The default value is `VideoDisplayMode.COVER`.

Attribute	Value	Description
videoDisplayMode	VideoDisplayMode.CONTAIN	Prioritize displaying the entire video content. Scale the video dimensions proportionally until one side of the video window aligns with the viewport frame. If the video dimensions do not match the display viewport size, under the precondition of maintaining aspect ratio, scale the video to fill the viewport. A ring of black bars will appear around the video after scaling.
	VideoDisplayMode.COVER	Prioritize filling the viewport. Scale the video dimensions proportionally until the entire viewport is filled by the video. If the video's aspect ratio differs from that of the display window, the video stream will fill the viewport by either cropping the periphery or stretching the image according to the viewport's ratio.
	VideoDisplayMode.FILL	Ensure the viewport is filled while displaying all video content, but the aspect ratio of the video dimensions is not guaranteed to remain unchanged. The video width and height will be stretched to match the viewport dimensions.

## videoResolution

Resolution `videoResolution` has three values:

`VideoResolution.RESOLUTION_480P`

`VideoResolution.RESOLUTION_720P`

`VideoResolution.RESOLUTION_1080P`. The default value is `VideoResolution.RESOLUTION_480P`.

### Resolution Description:

Video Profile	Resolution (W × H)	Frame Rate (fps)	Bitrate (Kbps)
480p	640 × 480	15	900
720p	1280 × 720	15	1,500
1080p	1920 × 1080	15	2,000

**FAQs:**

iOS 13 & 14 do not support encoding videos higher than 720P. It is recommended to limit the highest capture to 720P on these two system versions. See [Known issue case 12 with iOS Safari](#).

Firefox does not support custom video frame rates (30fps by default).

Due to factors such as system performance consumption, camera capture capability, and browser limitations, the actual values of video resolution, frame rate, and bitrate might not fully match the set values. In such cases, the browser will automatically adjust the profile to match the set values as closely as possible.

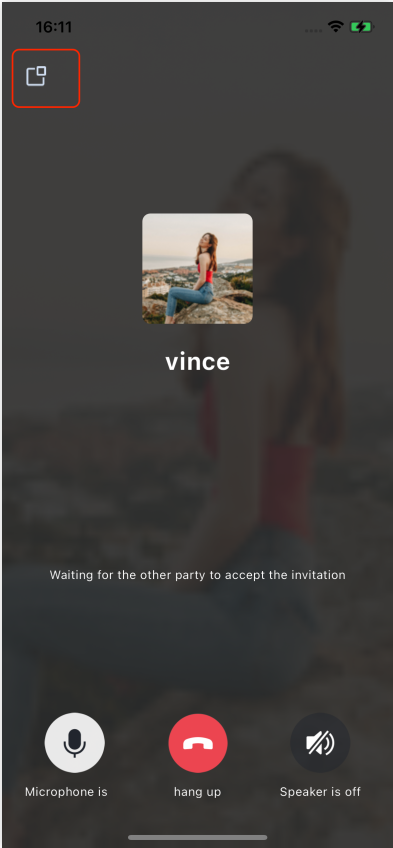
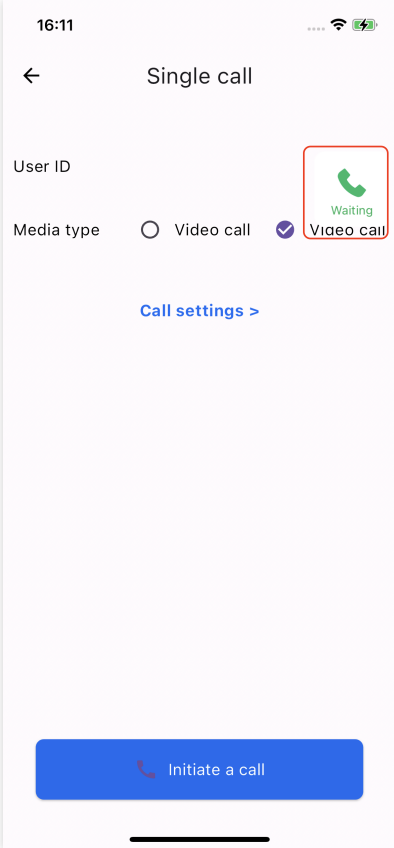

# Floating Window

## Android&iOS&Flutter

Last updated : 2024-05-08 11:37:24

This document describes how to use the Floating Window feature.

### Expected Outcome

Enable Floating Button	Voice Call Floating Window	Video Call Flo
		

### Floating Window Feature

TUICallKit allows users to minimize the call interface to a floating window using the floating window button on the upper-left corner during a call.

If your business needs to enable this feature, you can use the `enableFloatWindow` method to activate it when initializing the `TUICallKit` component:

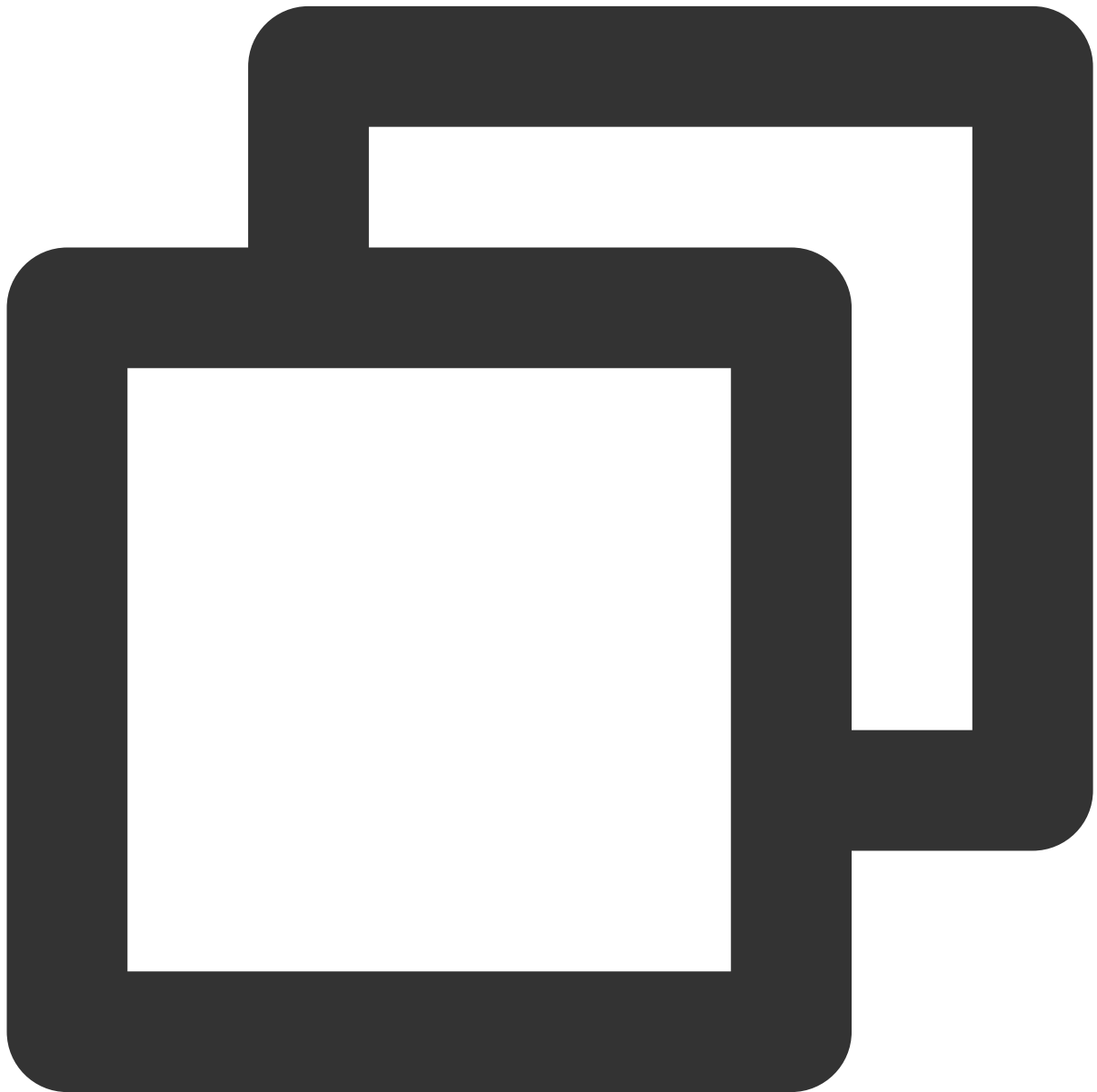
Android (Kotlin)

Android (Java)

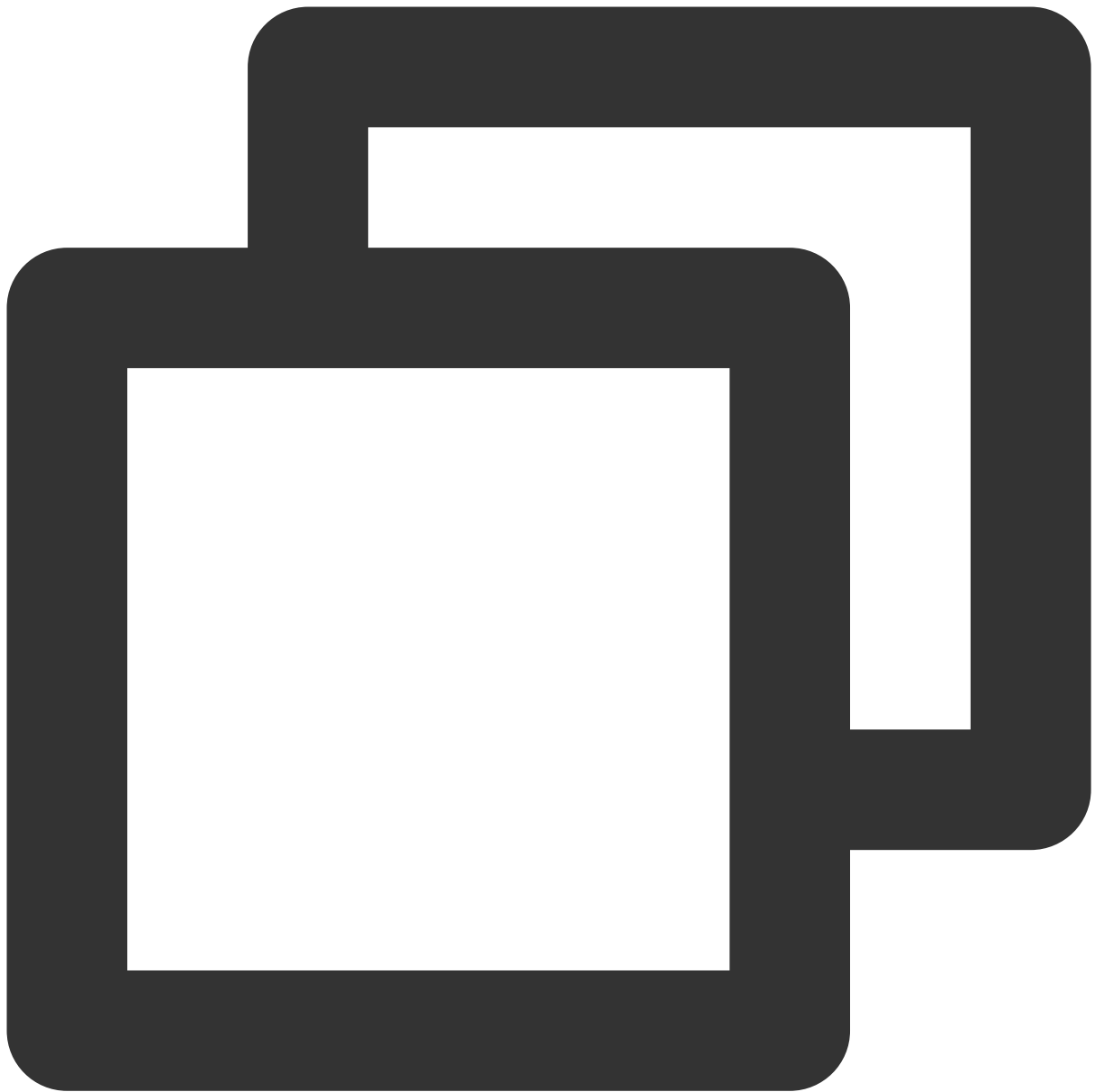
iOS (Swift)

iOS (Objective-C)

Flutter (Dart)

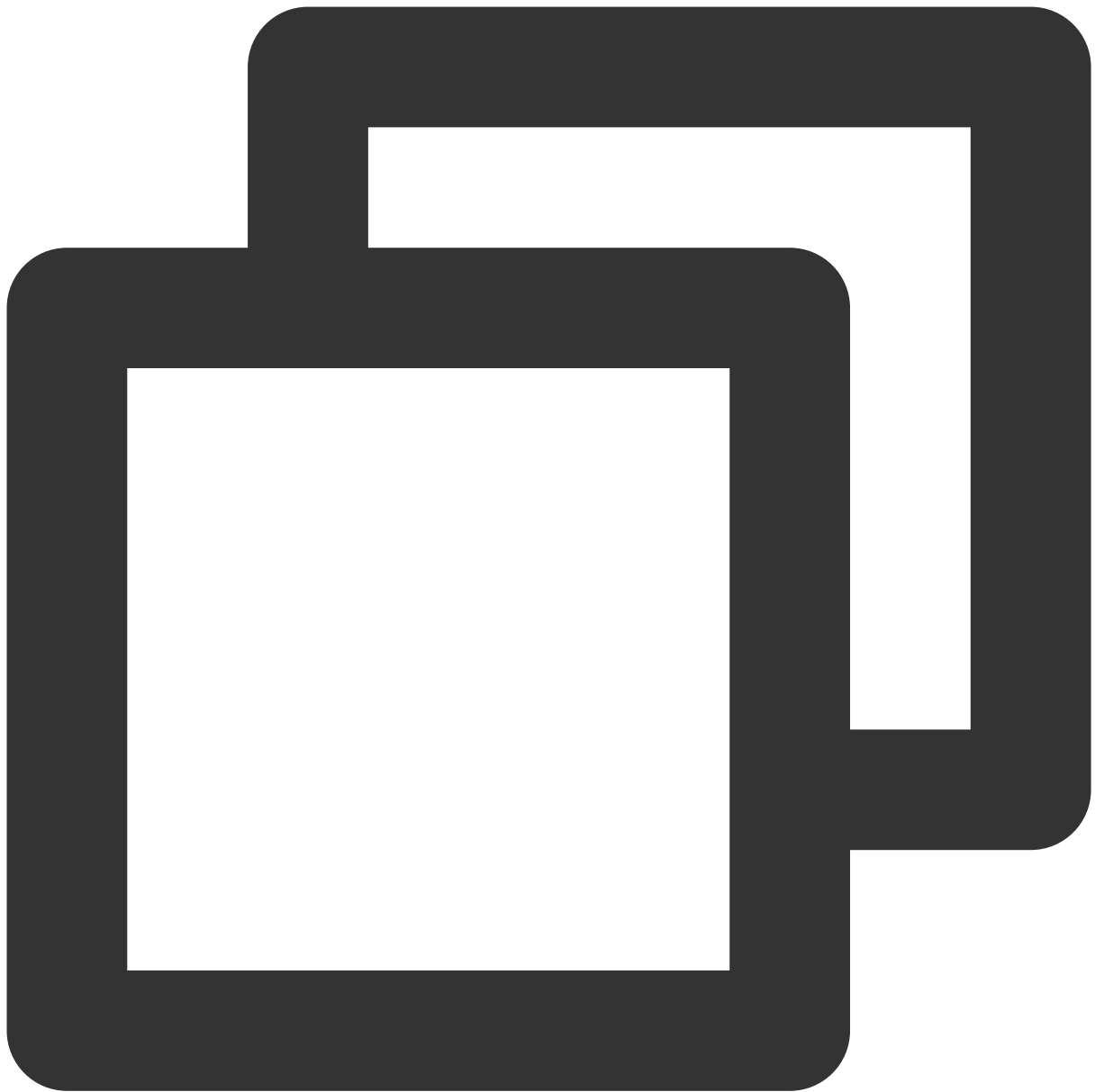


```
TUICallKit.createInstance(context).enableFloatWindow(true)
```

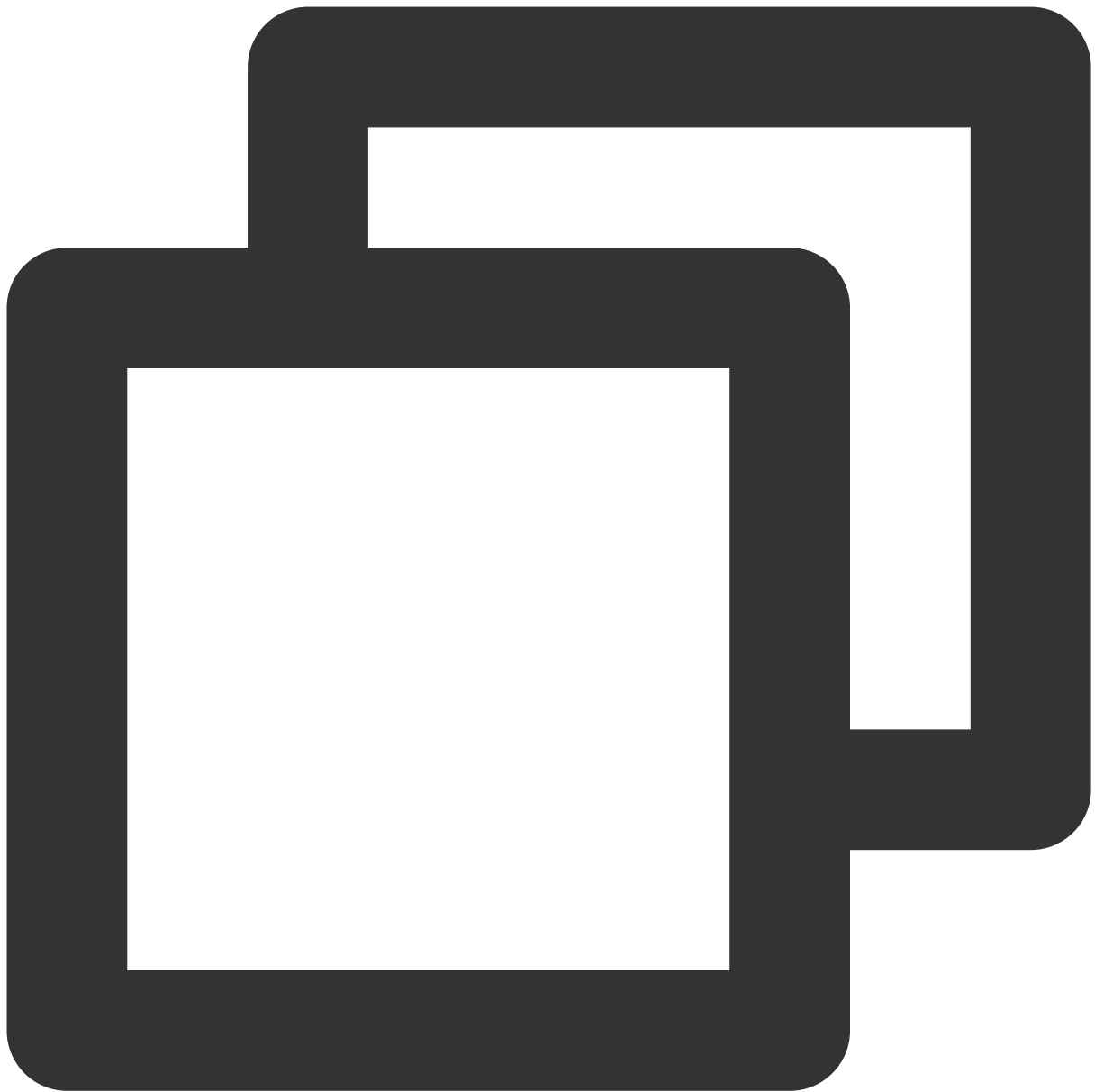


```
TUICallKit.createInstance(context).enableFloatWindow(true);
```

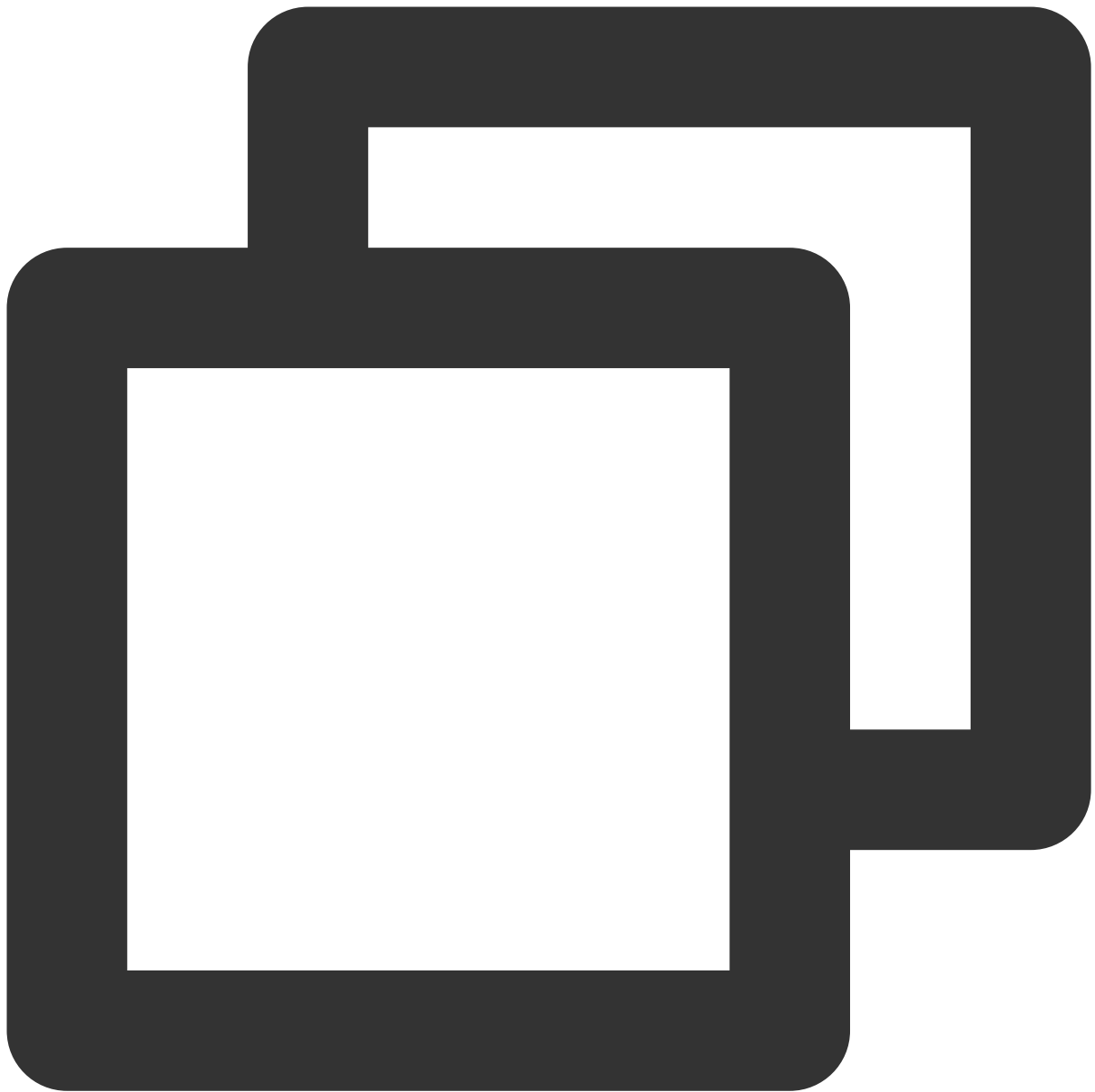




```
TUICallKit.createInstance().enableFloatWindow(true)
```



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



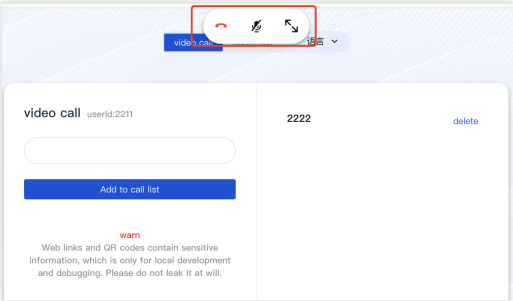
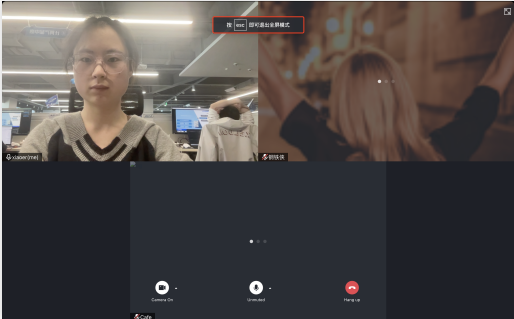
```
TUICallKit.instance.enableFloatWindow(true);
```

# Web&H5

Last updated : 2024-05-15 17:17:33

This document describes how to use the Floating Window feature.

## Expected Outcome

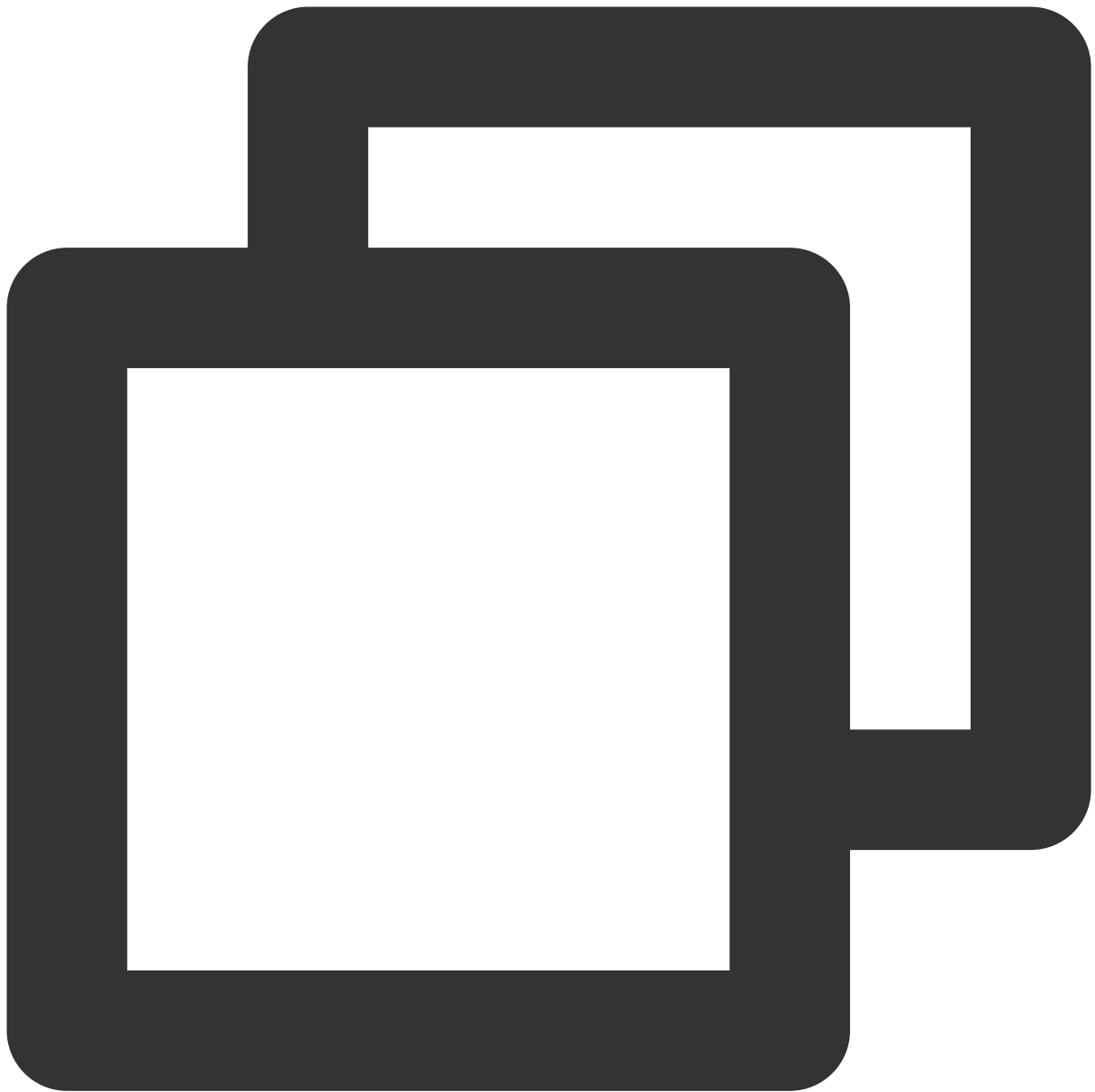
Web Floating Window	Web Full Screen
	

## Floating Window Feature

Method 1: Call the enableFloatWindow(enable: boolean) API to enable/disable the floating window.

**Note:**

Vue v3.1.0 or later versions are supported.



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

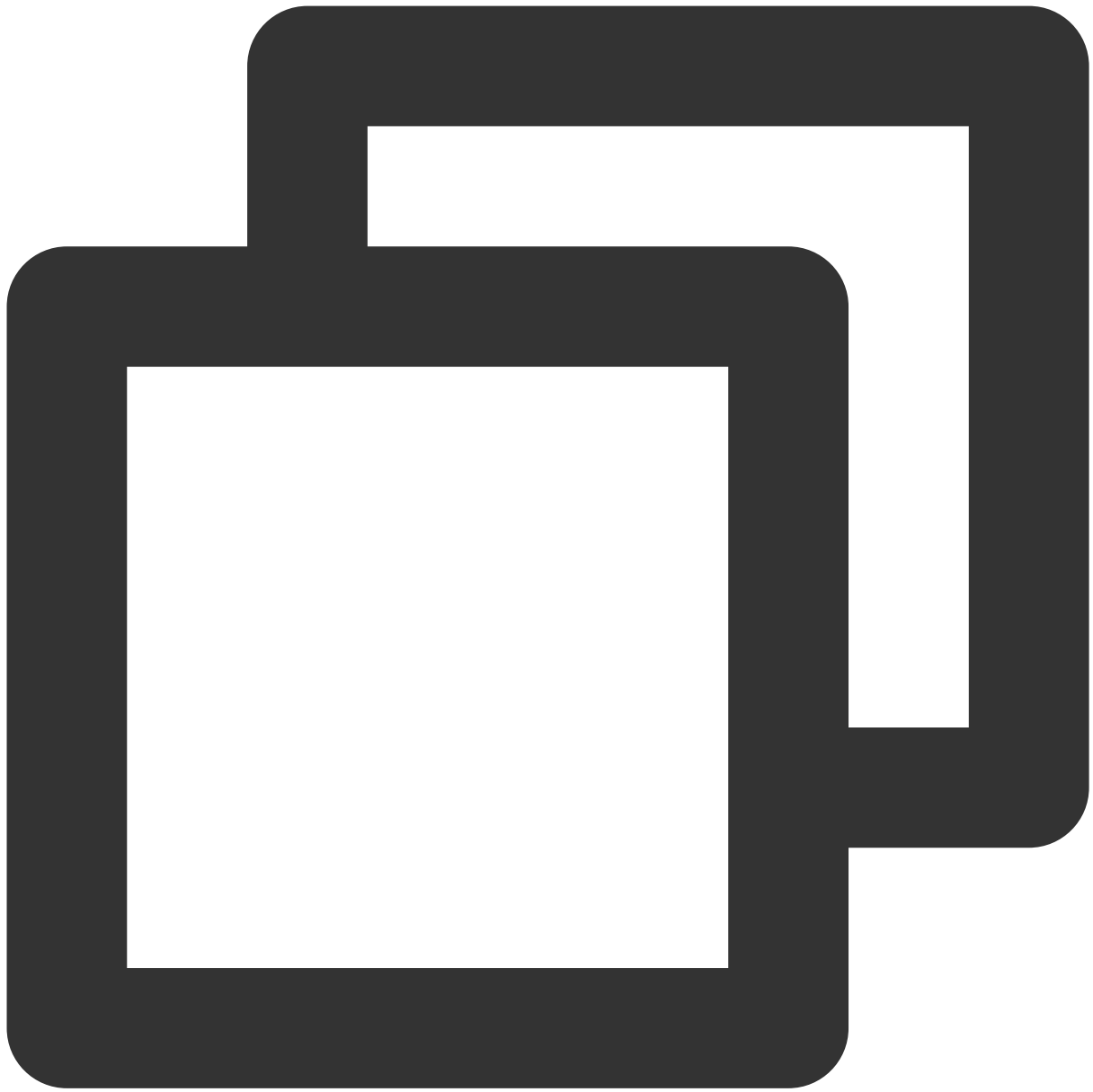
Method 2: Enable/disable the floating window and full screen via attributes.

allowedMinimized attribute controls the floating window's on/off.

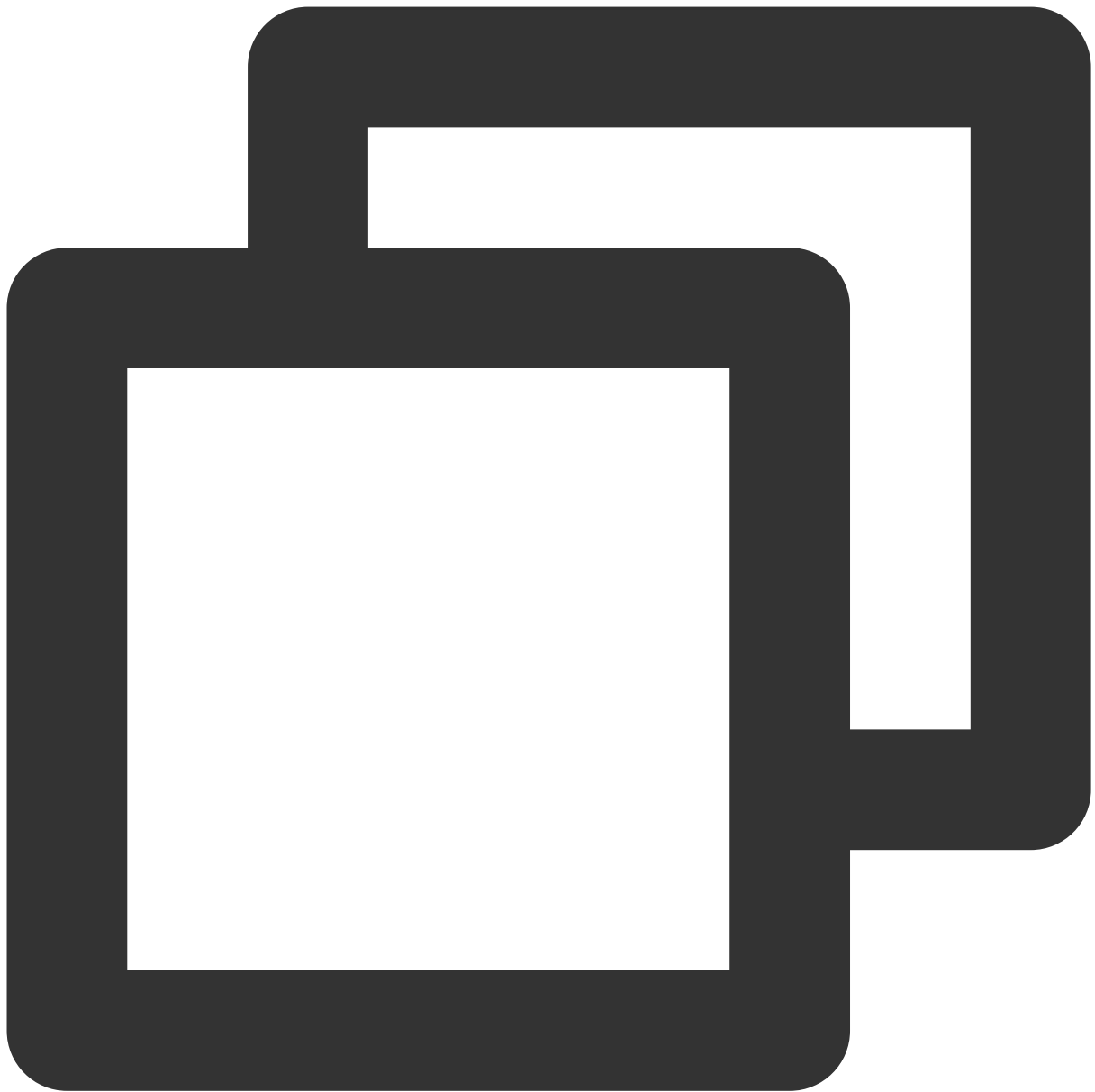
allowedMinimized attribute controls the full screen's on/off.

React

Vue



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



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

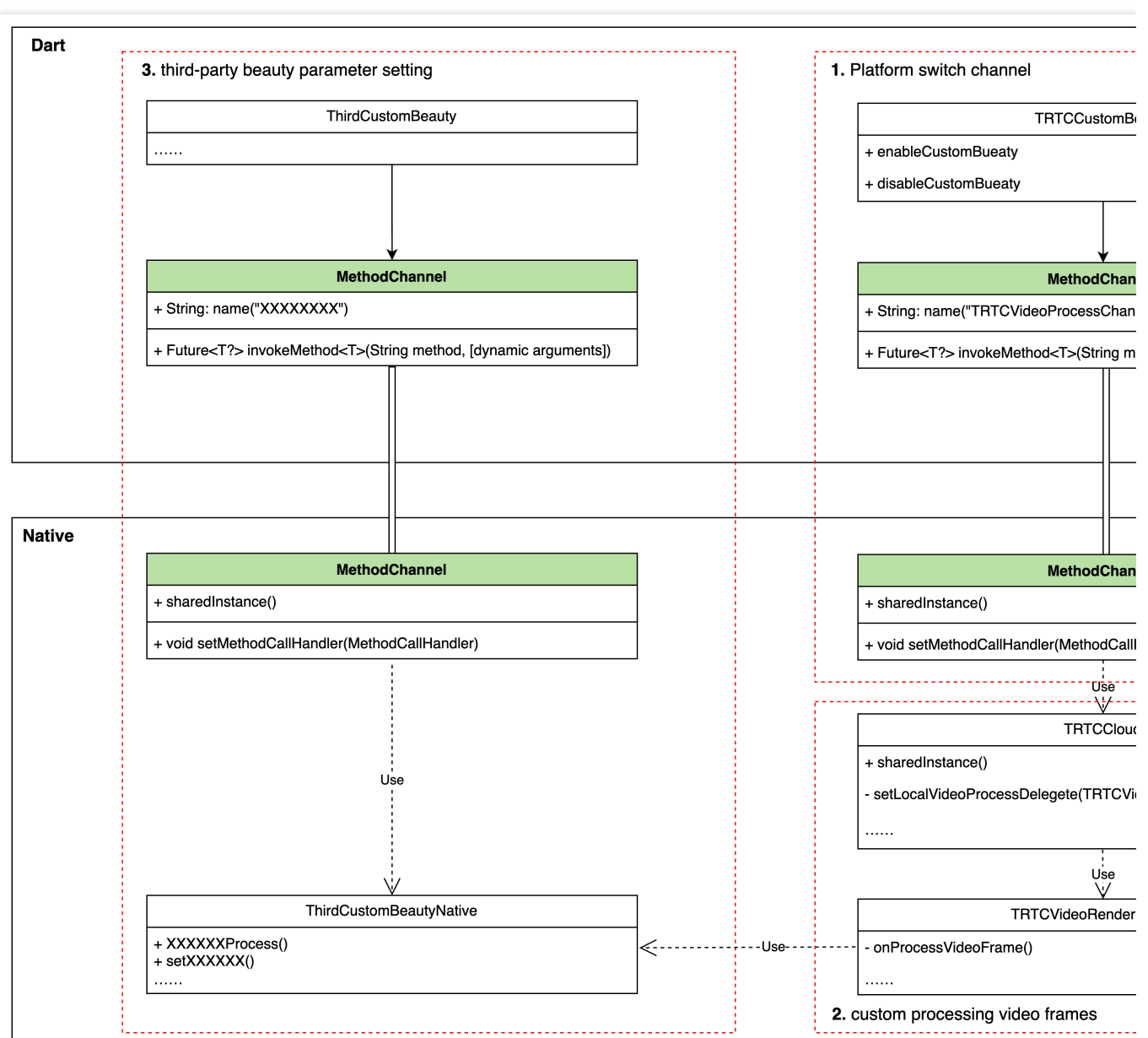
# Beauty Effects (TUICallKit)

## 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;

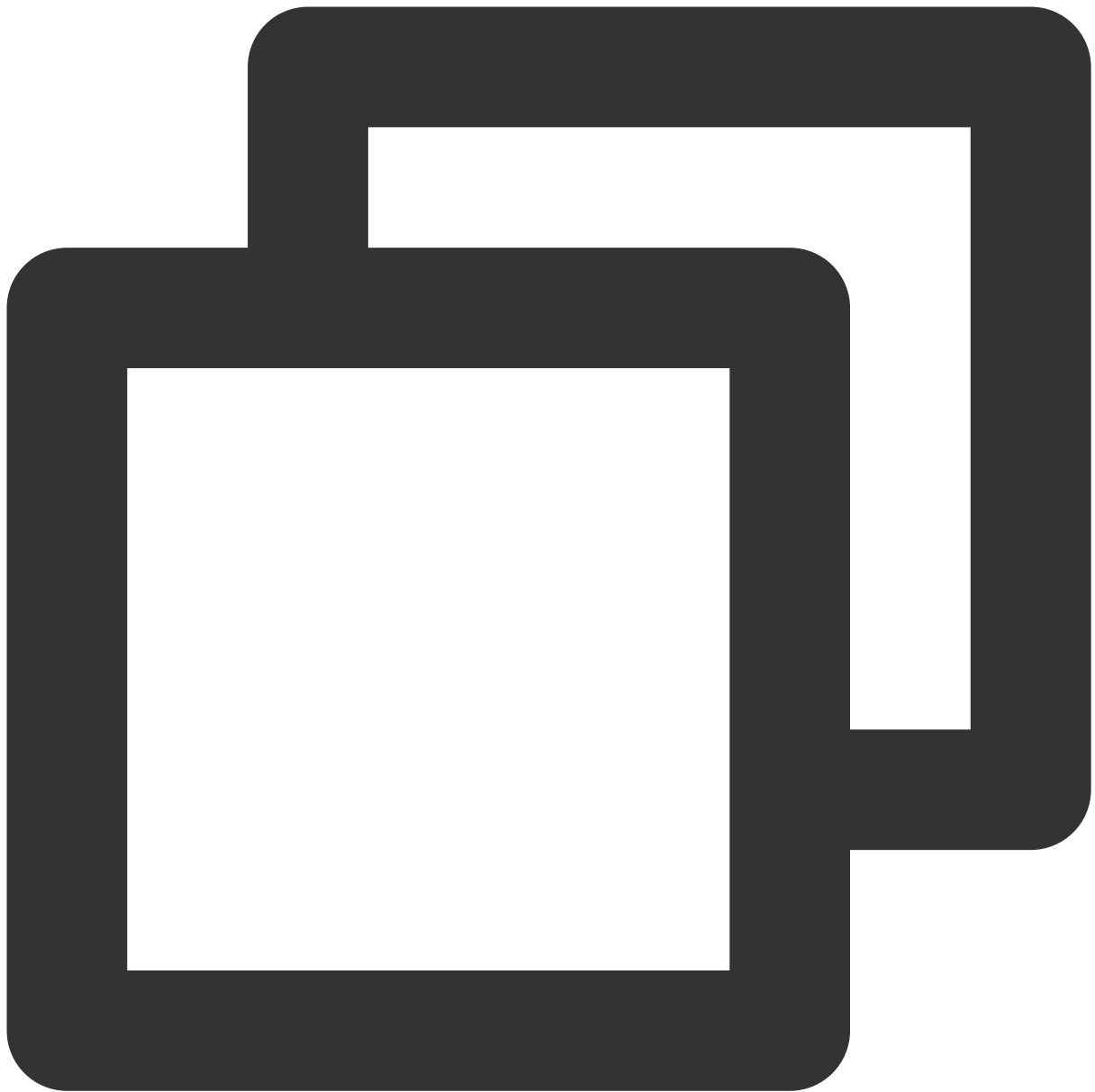
Step 2: In the TRTC custom rendering processing logic onProcessVideoFrame(), use the **beauty processing module** to process the original video frames;

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');

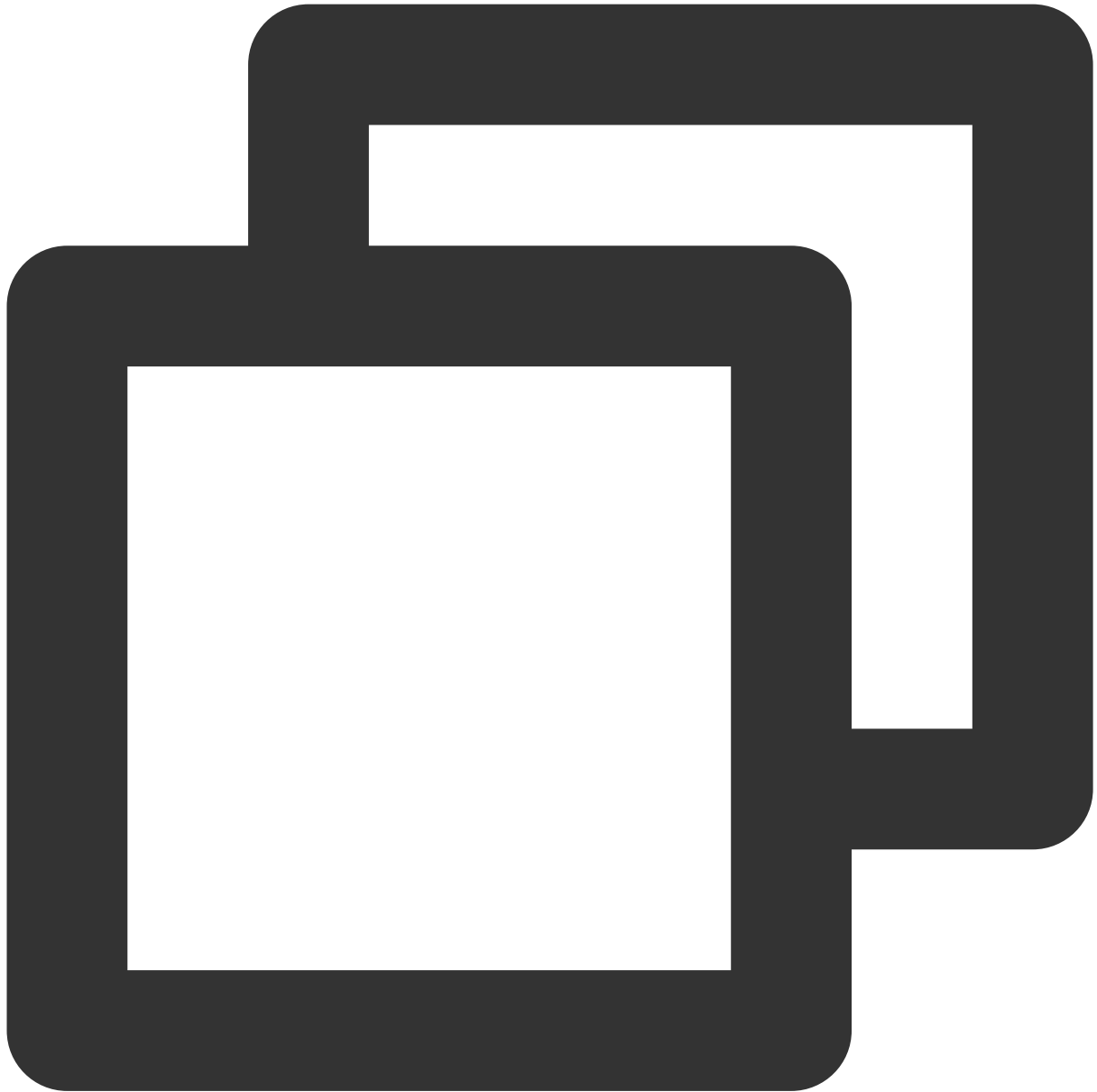
void enableTUICallKitCustomBeauty() async {
  await channel.invokeMethod('enableTUICallKitCustomBeauty');
}

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("");
})));

}

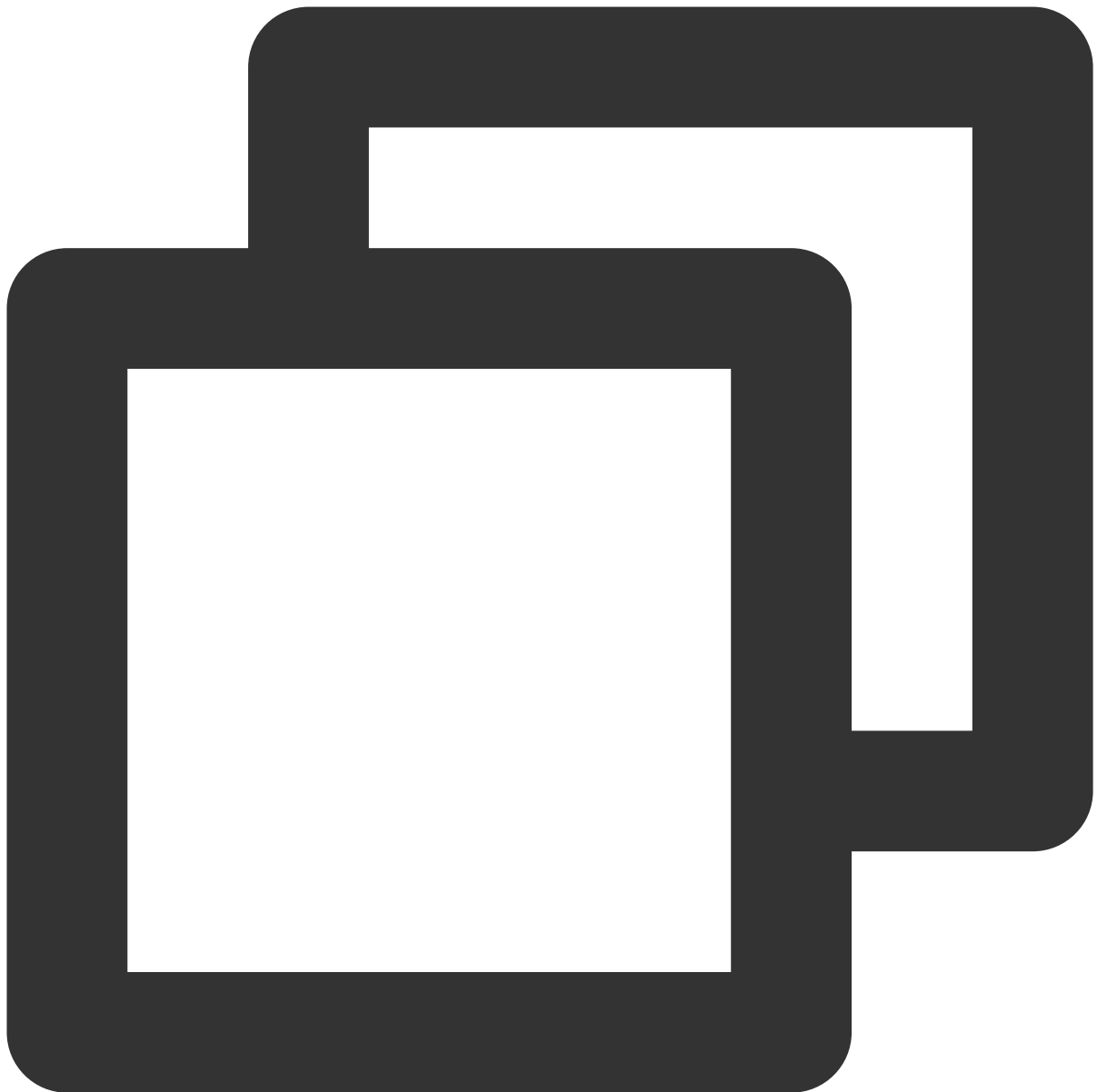
public void enableTUICallKitCustomBeauty() {

}

public void disableTUICallKitCustomBeauty() {

}

}
}J
```



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

  override func application(_ application: UIApplication,
                             didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) throws FlutterError {
    GeneratedPluginRegistrant.register(with: self)

    guard let controller = window?.rootViewController as? FlutterViewController else {
      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`:

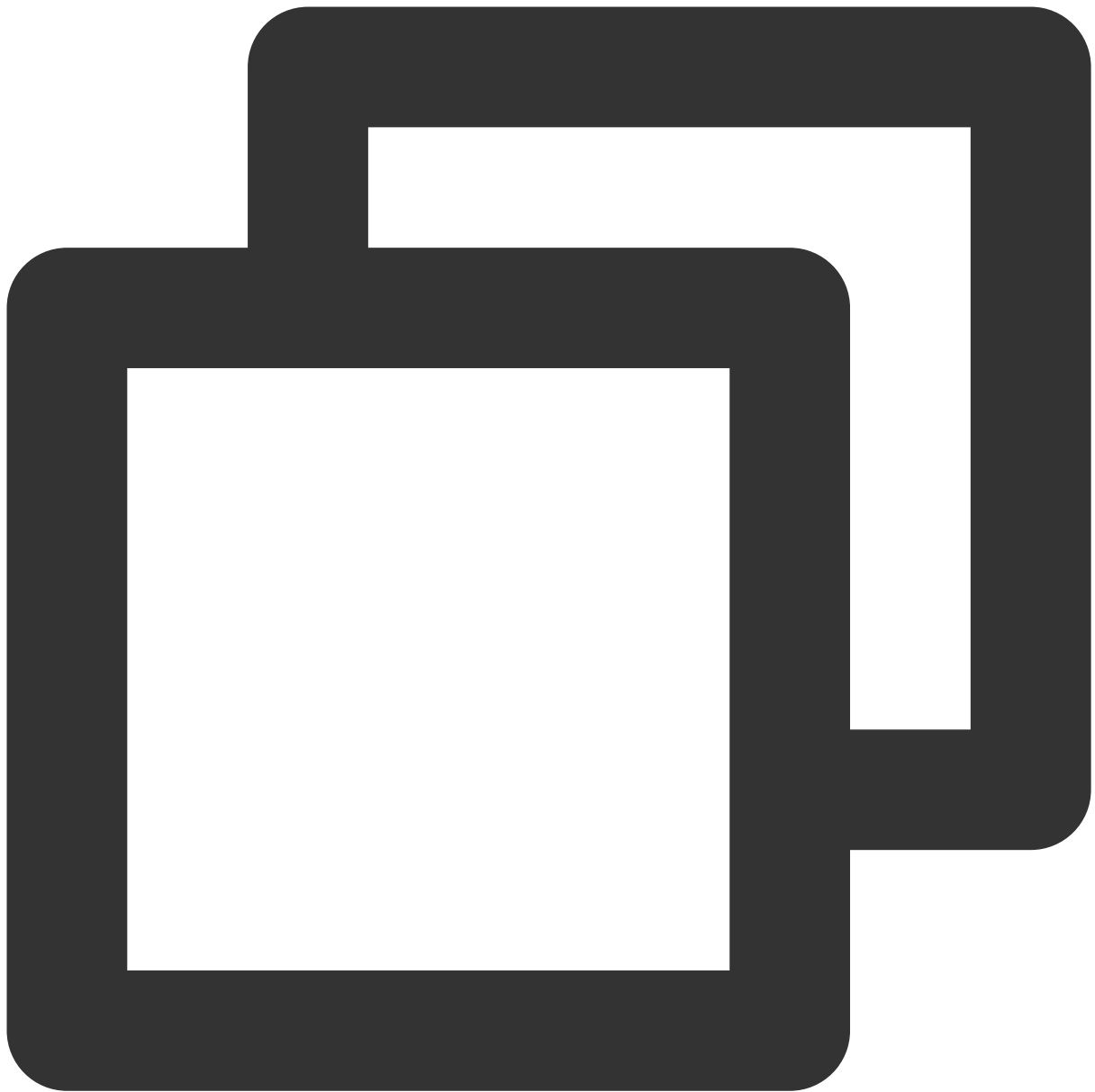
```

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();

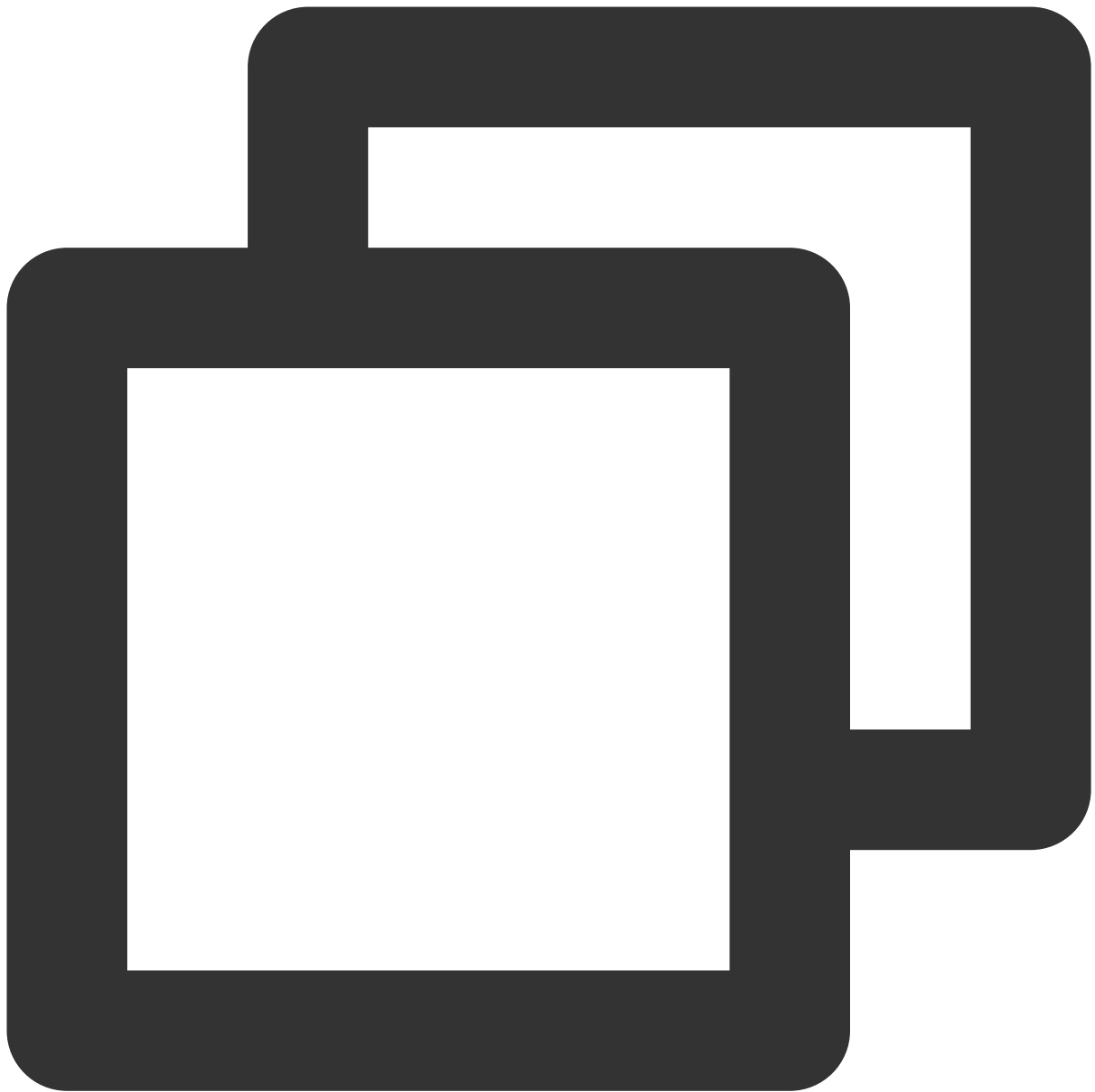
    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame trtcVideoFrame,
                                   TRTCCloudDef.TRTCVideoFrame trtcVideoFrame1) {
        //Beauty processing logic
        mBeautyModel.process(trtcVideoFrame, trtcVideoFrame1);
        ...

        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.

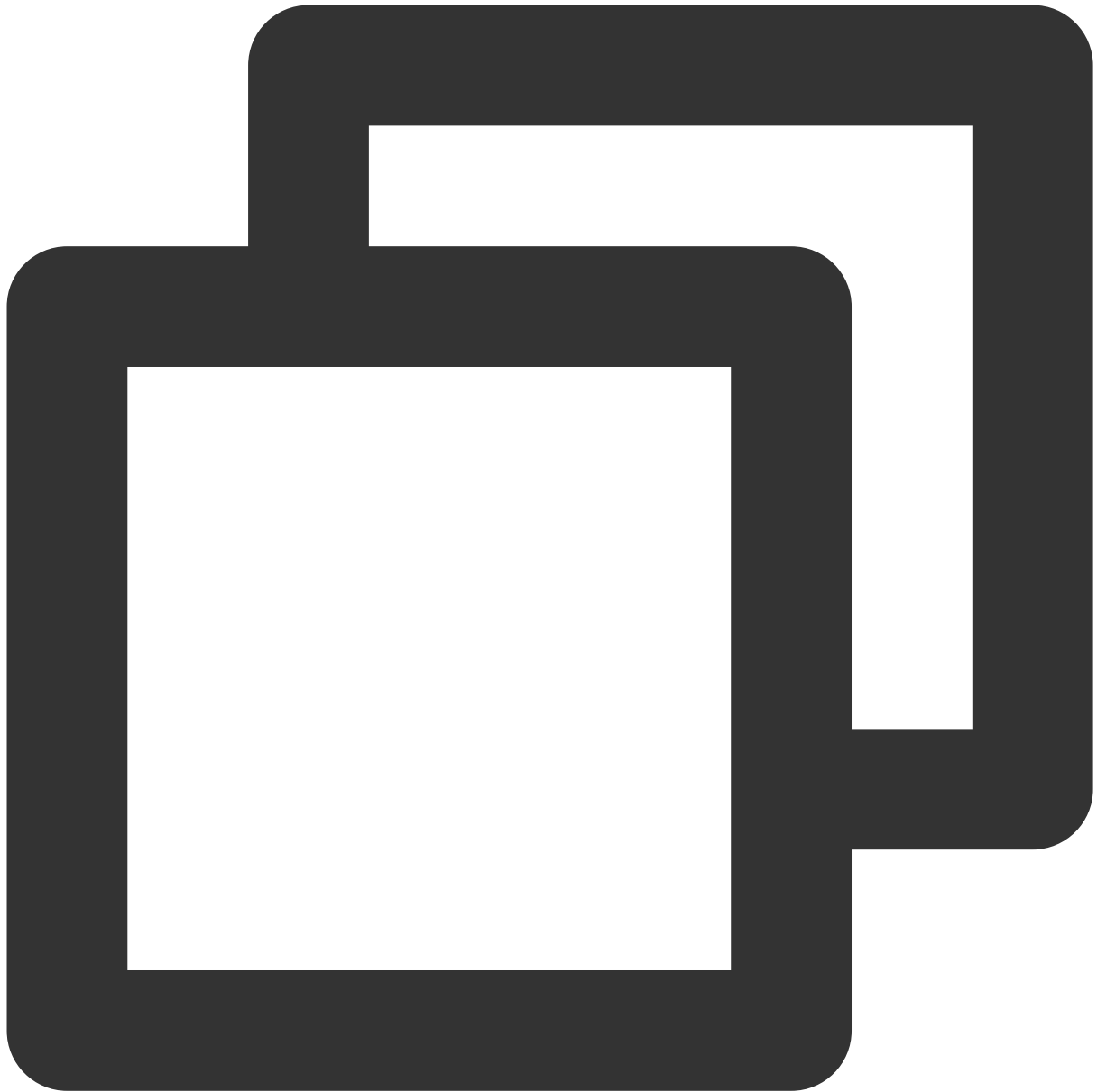
### Step 1: Download and integrate beauty effect resources.

1. Based on the package you purchased, [Download SDK](#).
2. Add the resource files to your own project:

Android

iOS

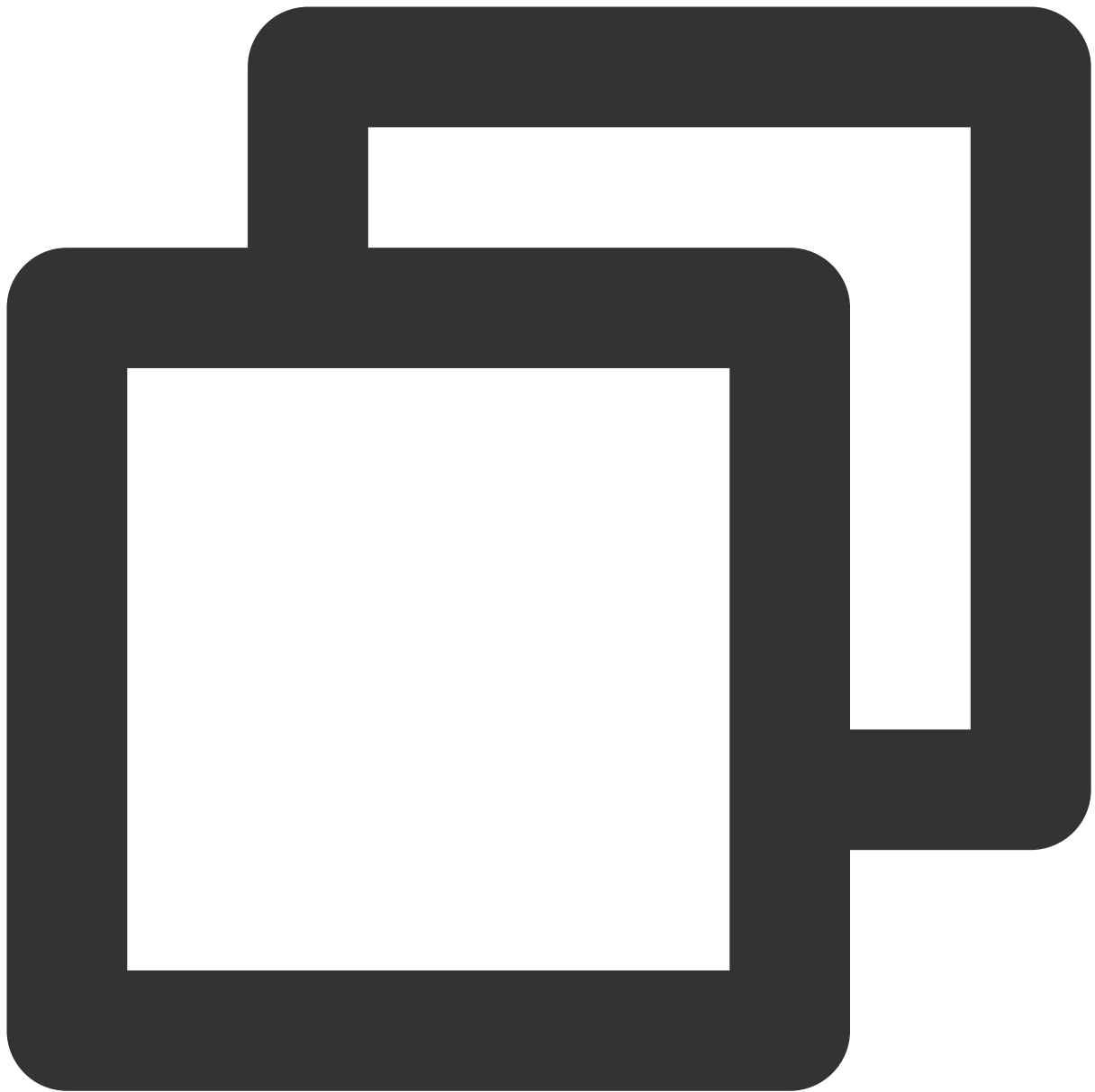
1. In the app module, find the build.gradle file and add the Maven reference address corresponding to your package. For example, if you have chosen the S1-04 package, add the following code:



```
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>)
```

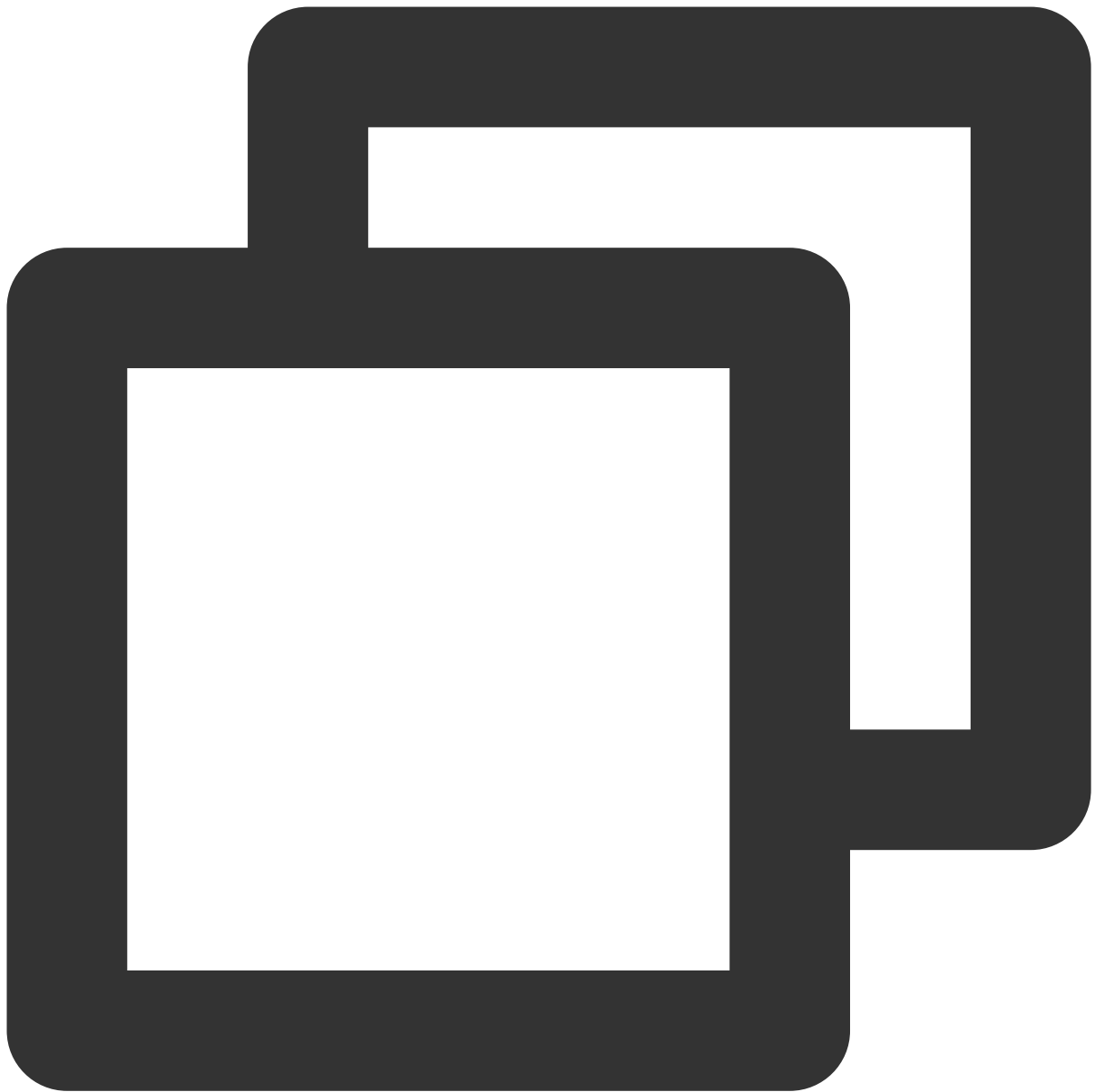
The tag is added, as shown in the following figure:

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

#### 4. Aliasing Configuration

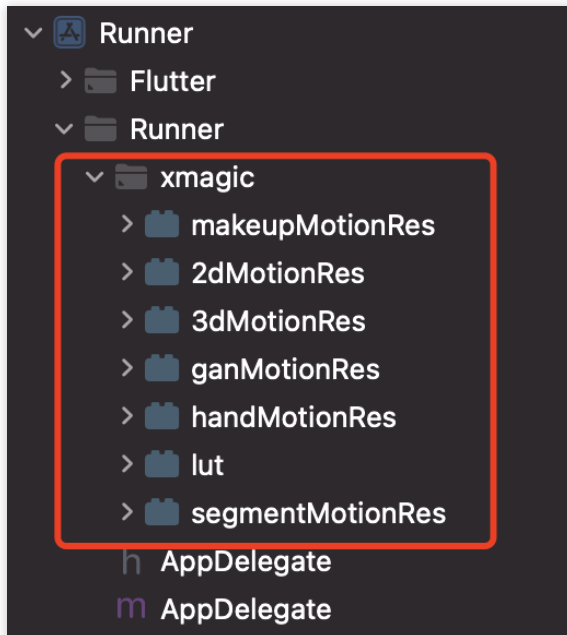
If you enable compile optimizations while building the release package (by setting `minifyEnabled` to `true`), some codes that are not called at the Java level might be trimmed. However, these codes could potentially be called at the Native level, leading to a `no xxx method` exception.

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.** { *;}
```

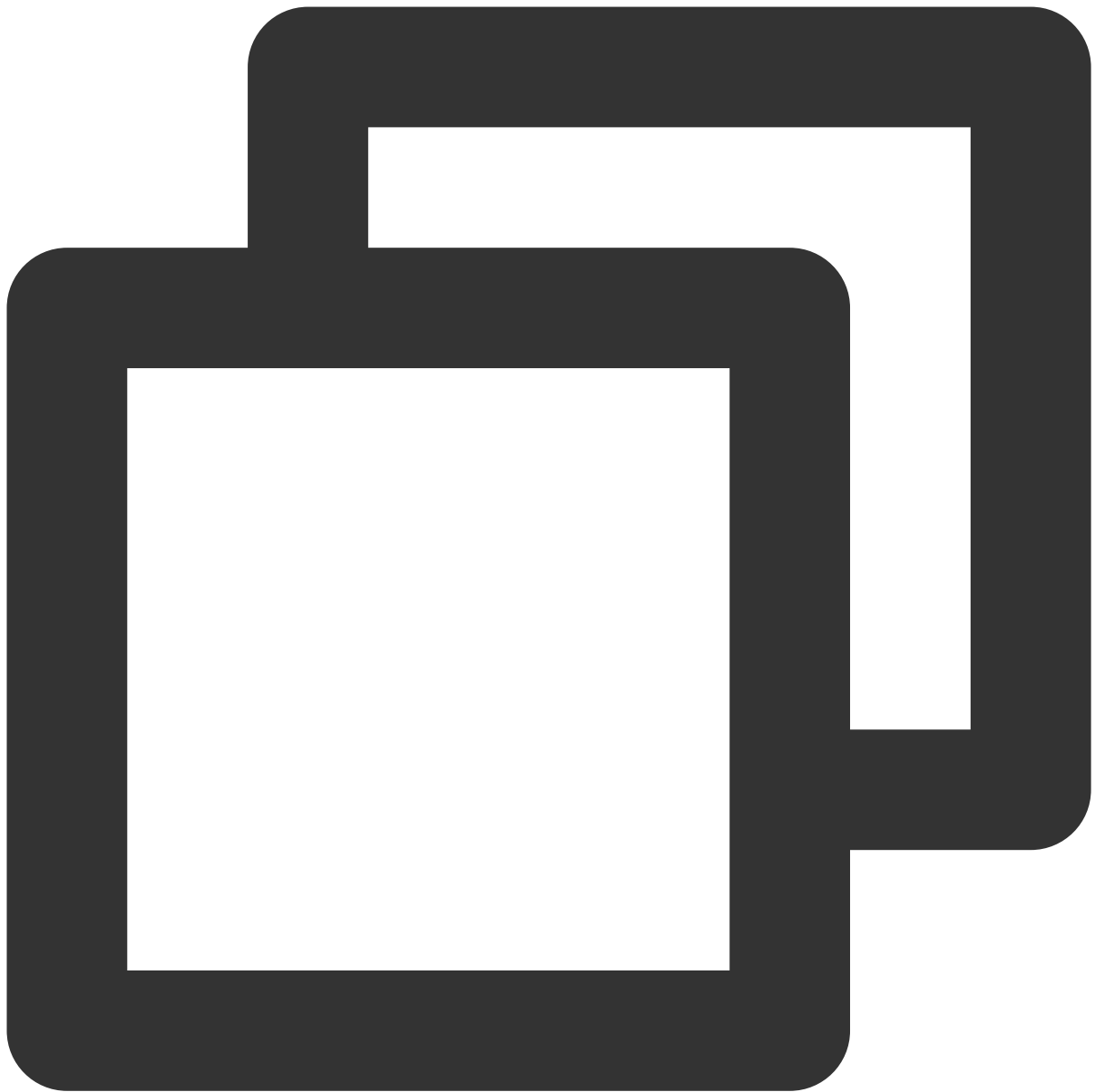
1. Add beauty effect resources to your project, as shown in the following figure (your resources may not exactly match those in the following figure):



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.

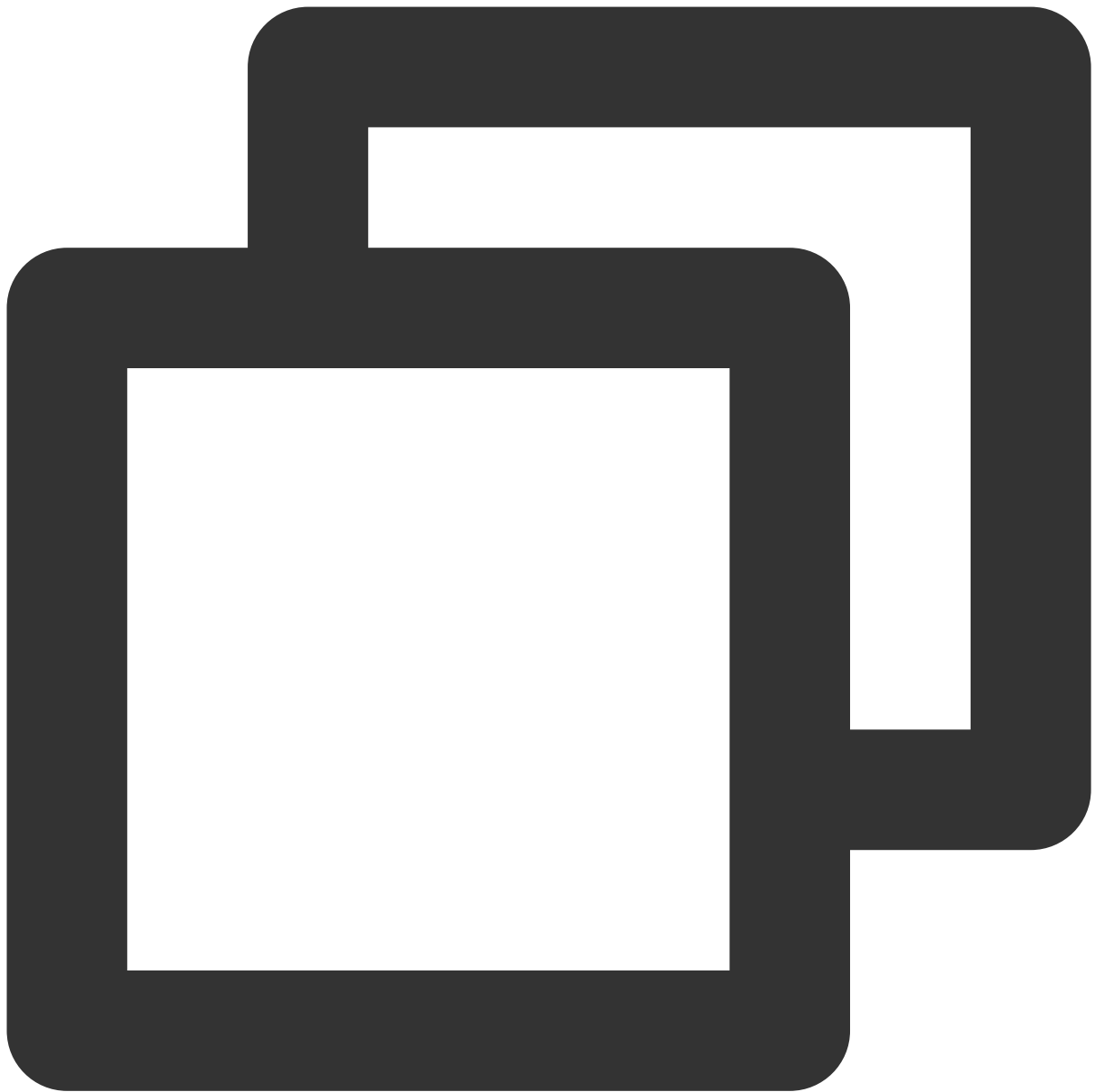
**GitHub Reference:** Add the following reference in your project's pubspec.yaml file:



```
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 it has dependency 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](#).

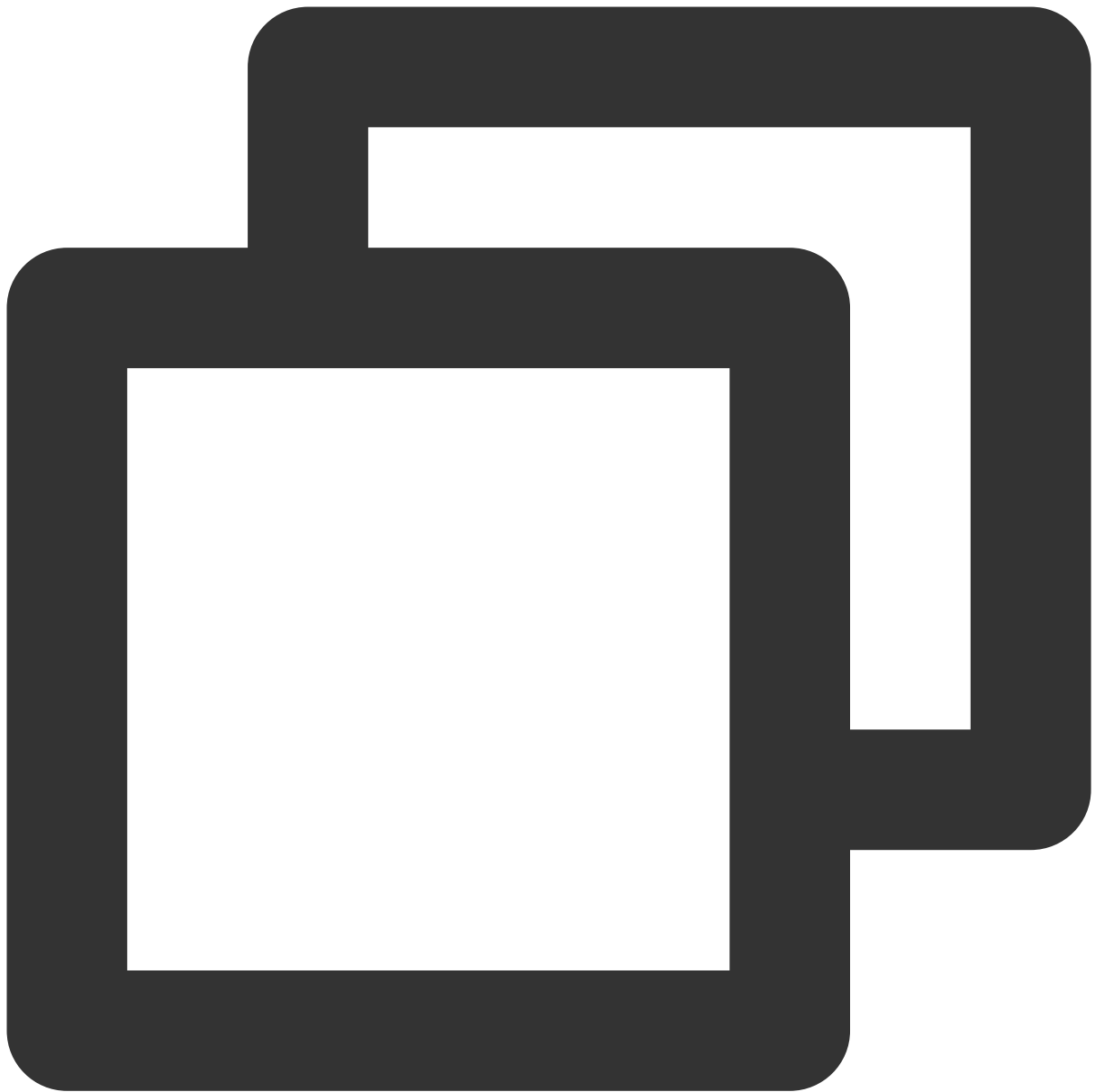
### Step 4: Complete the beauty processing in the TRTC custom rendering logic in Native.

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);
}
```

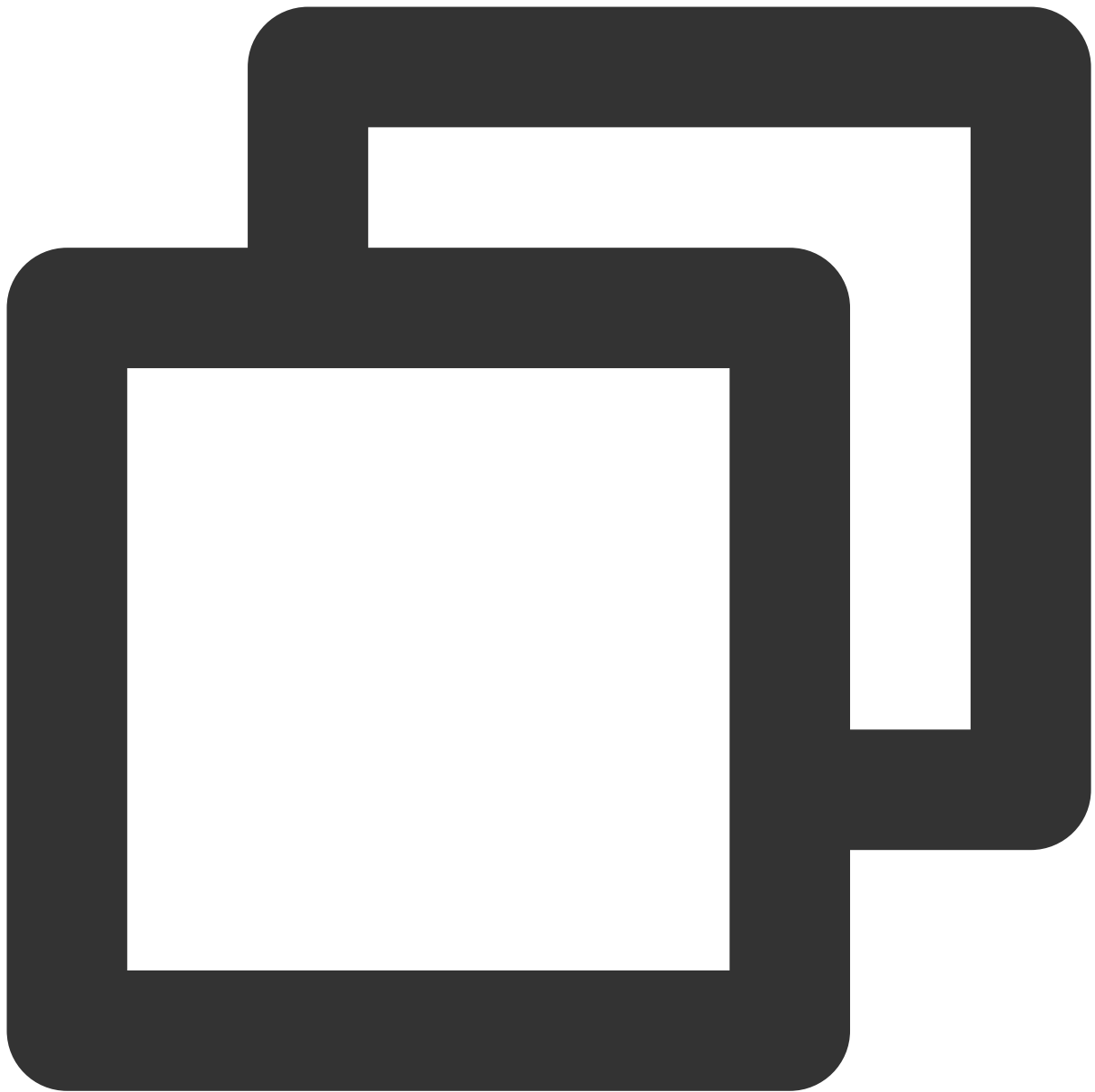
```
}

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](#).

# Custom Ringtone

## Android

Last updated : 2024-05-08 11:37:24

This document describes how to replace the incoming call ringtone of TUICallKit, which is divided into **application ringtone** and **offline push ringtone**.

## I. Setting Application Ringtone

There are two ways to set the application ringtone: replace the ringtone audio, or call the Setting Ringtone API.

### 1. Replace Audio File

If you include the TUICallKit component through source code dependency, you can achieve the goal of replacing the ringtone by changing the three audio files under the `res\raw` folder:

File Name	Purpose
phone_dialing.mp3	Ringtone when initiating a call
phone_hangup.mp3	Ringtone when the call is disconnected
phone_ringing.mp3	Ringtone when receiving a call

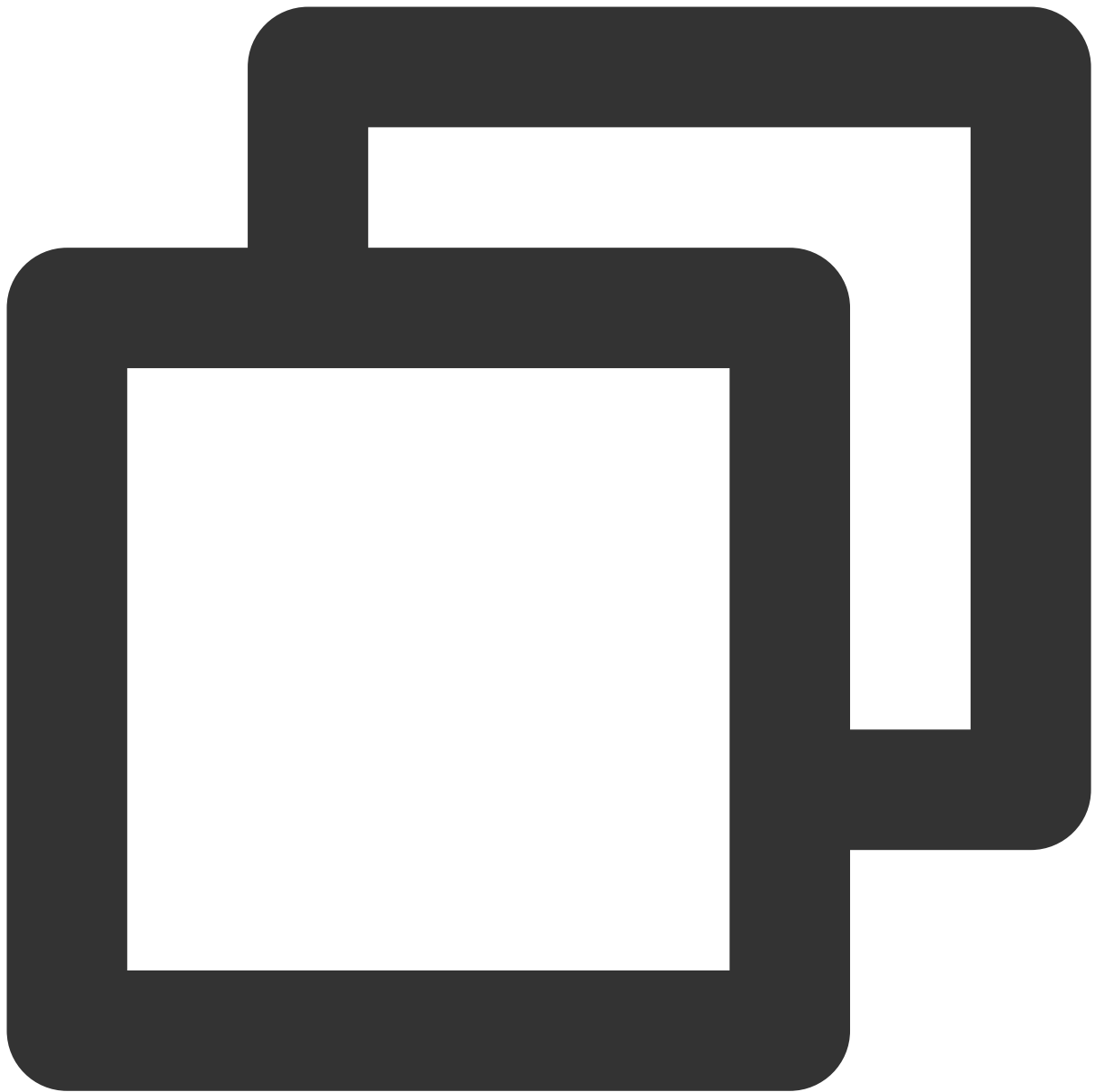
### 2. Set Ringtone API

You can also use the `setCallingBell` API to set the incoming call ringtone.

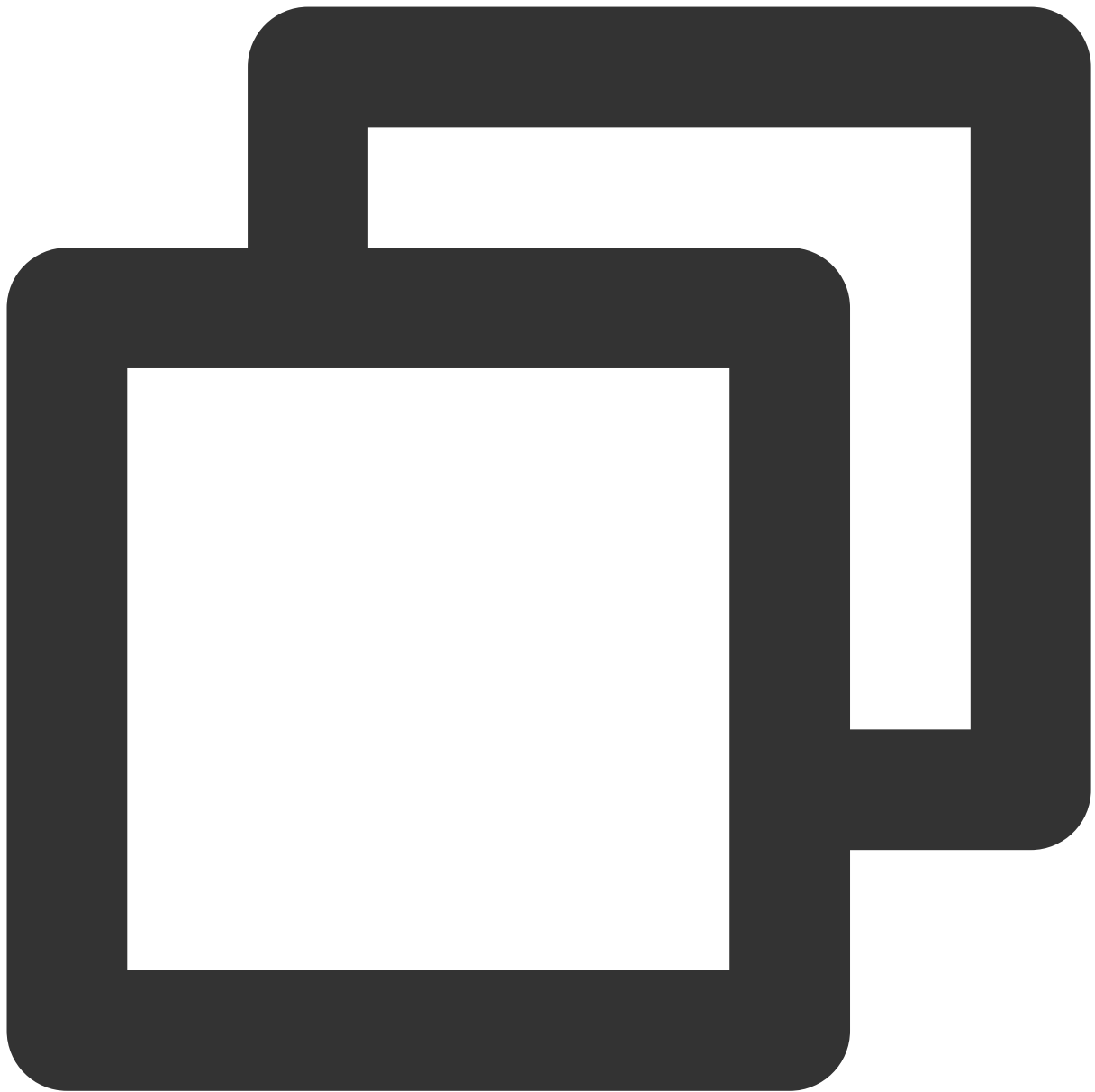
Kotlin

Java





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



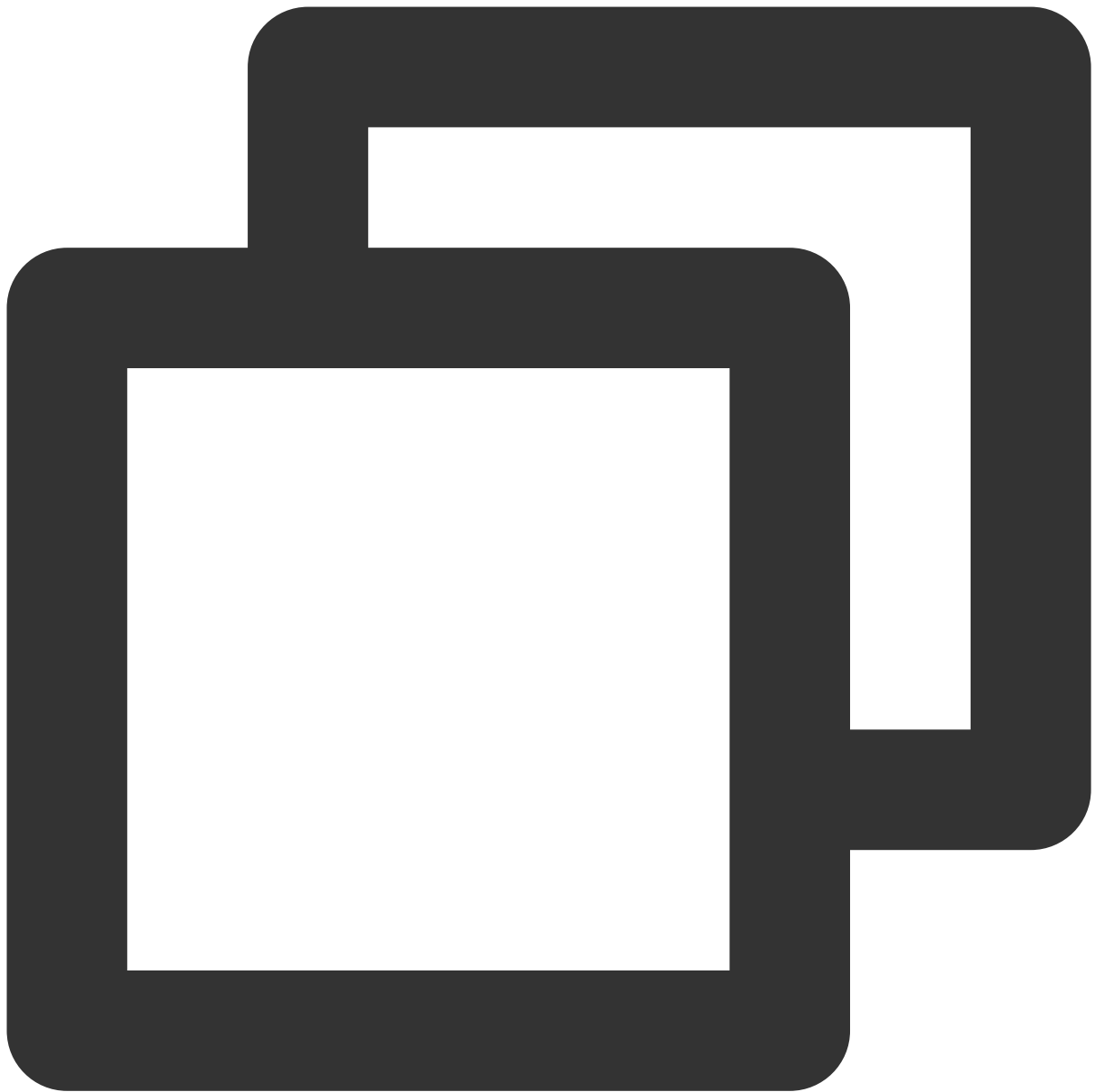
```
TUICallKit.createInstance(context).setCallingBell(filePath);
```

### 3. Setting Mute Mode

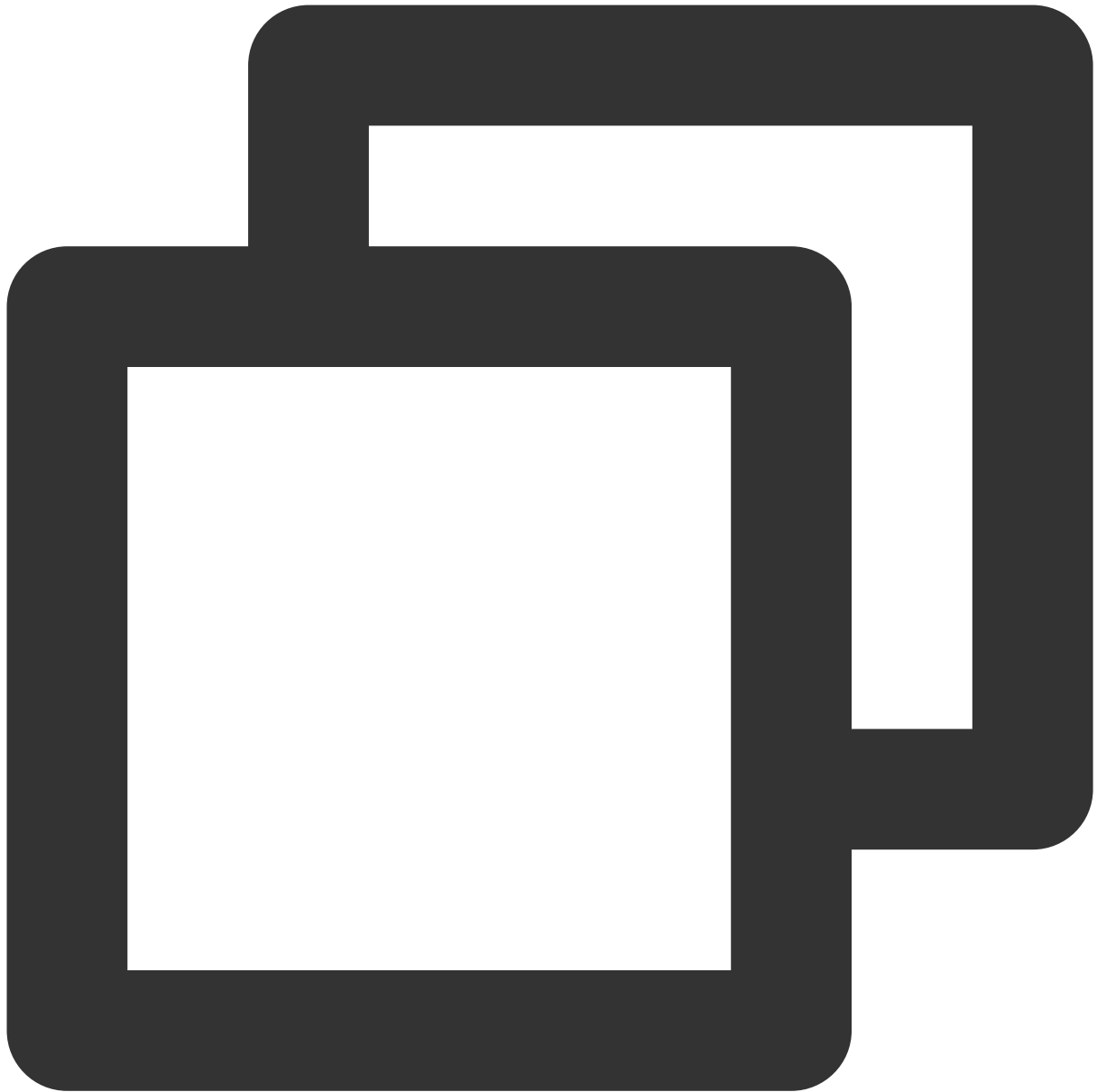
If you do not require a ringtone, you can use `enableMuteMode` to set the mute mode.

Kotlin

Java



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



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

## II. Setting Offline Push Ringtone

Offline notification ringtones only support customization from the following manufacturers: **Huawei, Xiaomi, FCM, and APNs**. Other manufacturers like OPPO, VIVO, Honor, etc., are not supported at this time.

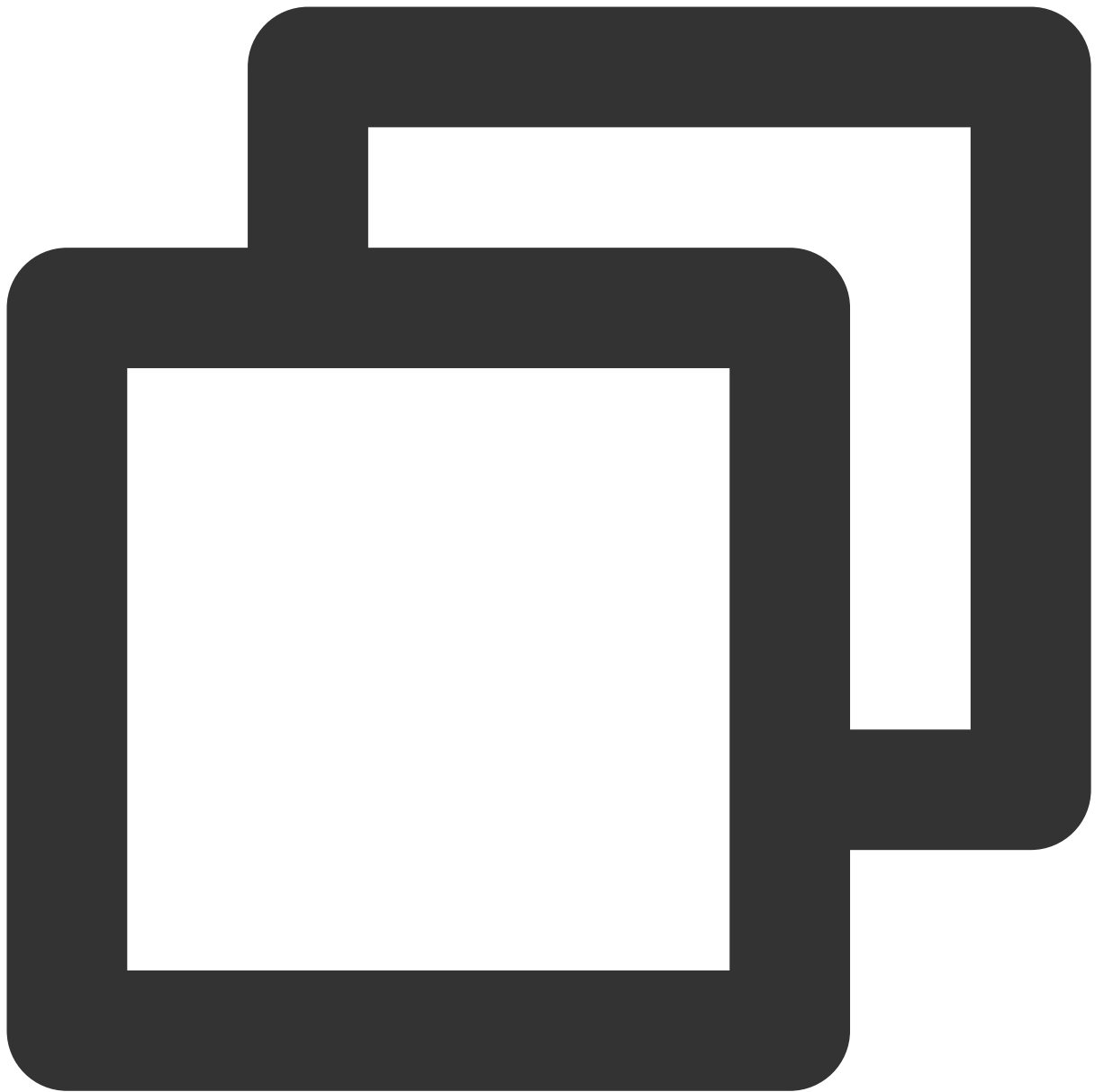
TUICallKit introduces the TUIOfflinePush component. It requires calling the following method at application launch to enable custom offline notification ringtone capabilities.

Kotlin

Java



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



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

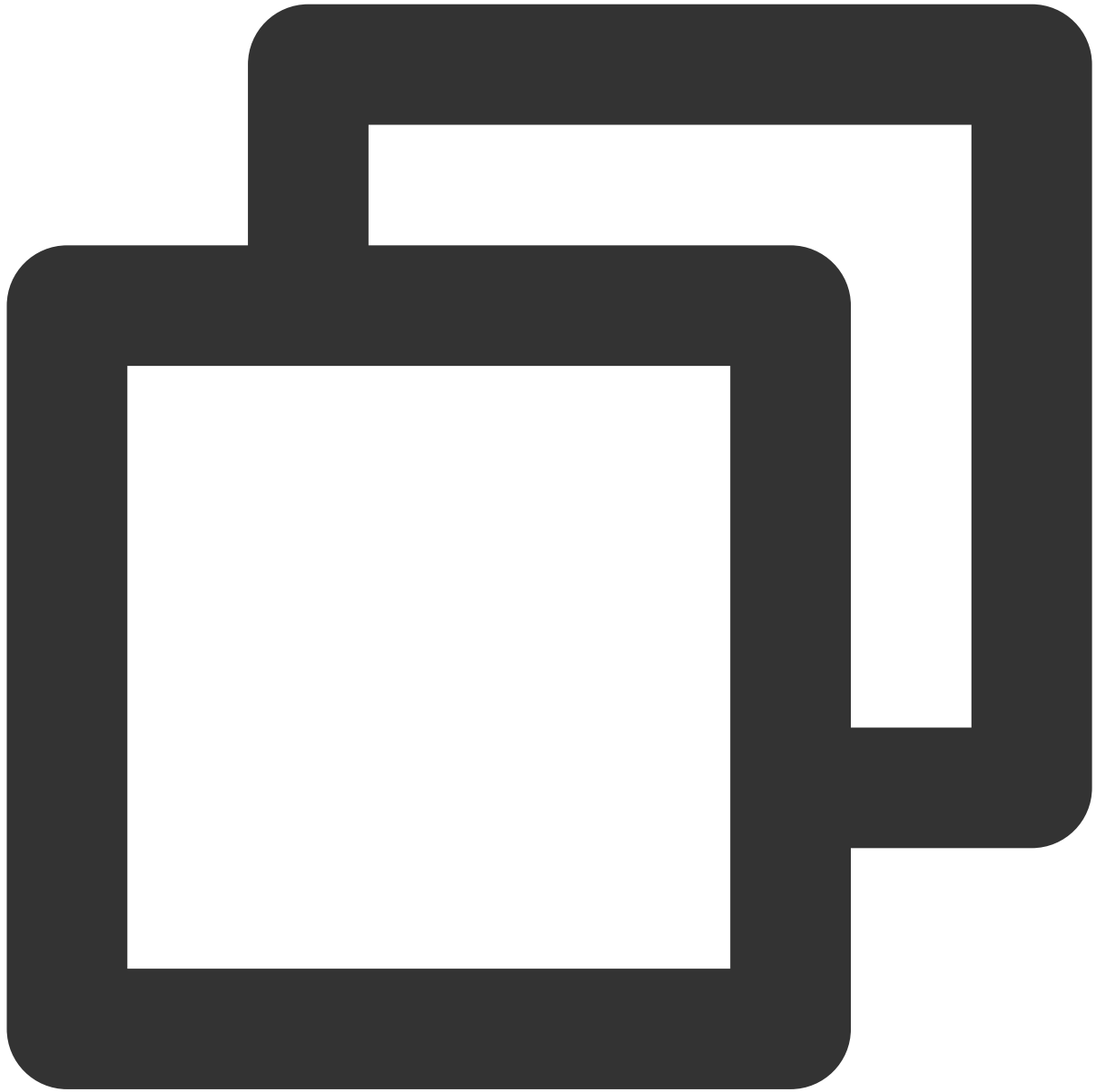
## 1, Huawei & APNs

Call the APIs including `setAndroidSound()` and `setIOSSound()`.

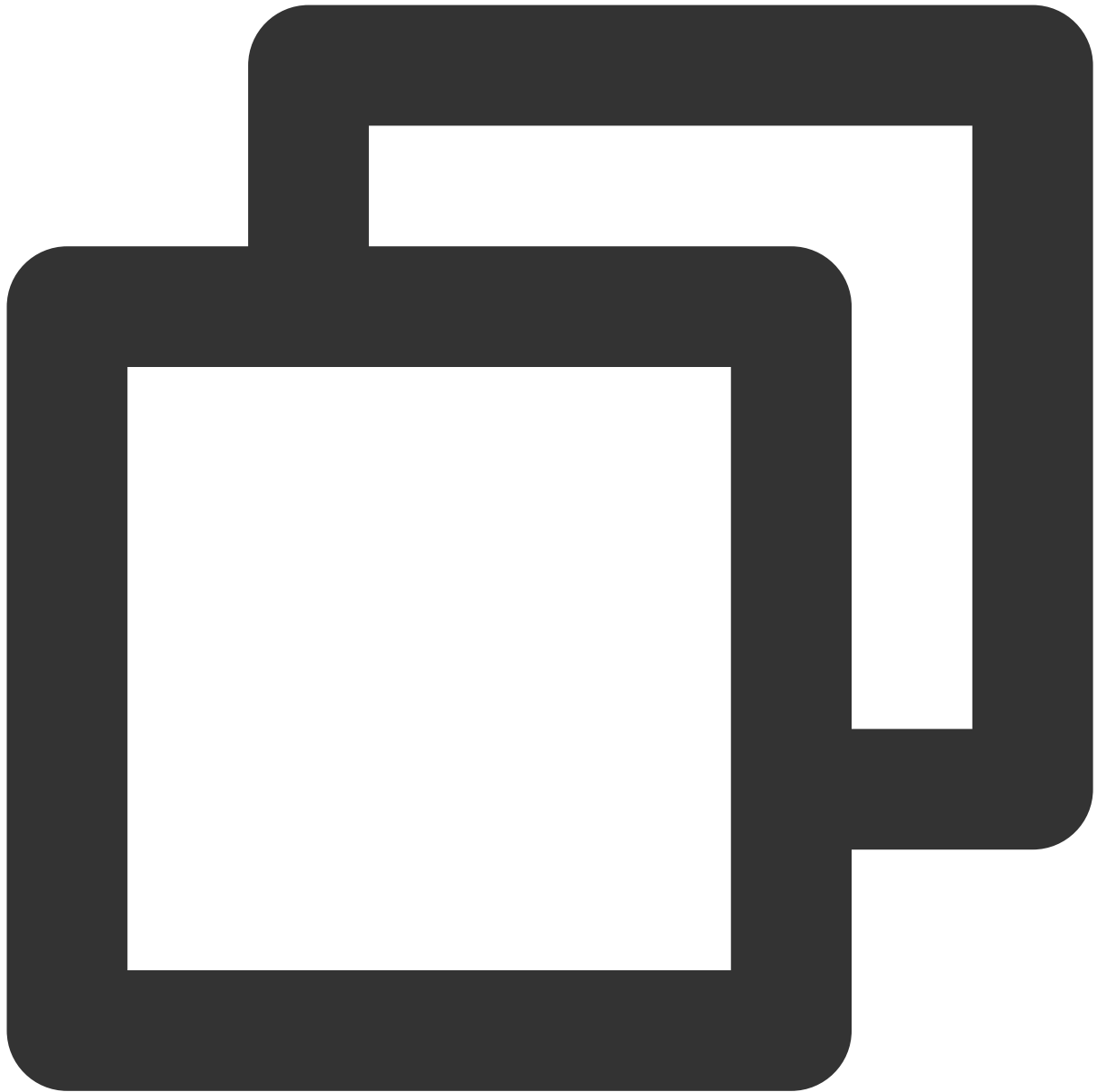
Customize Ringtone Resource File, add the ringtone file to the local Android Studio project's res/raw directory.

Kotlin

Java



```
val pushInfo = OfflinePushInfo()  
pushInfo.iosSound = "Ring Tone Name.mp3"  
pushInfo.androidSound = "Ring Tone Name"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("Ring Tone Name.mp3");
pushInfo.setAndroidSound("Ring Tone Name");
```

**Caution**

Supported in IMSDK v6.1.2155 and later versions.

The ringtone for Huawei mobile phone push messages is the one set when the notification channel is first created.

Setting a different ringtone later is ineffective. If you wish to update the custom notification ringtone, you need to uninstall and reinstall. For more details, see: [Huawei Custom Ringtone](#).



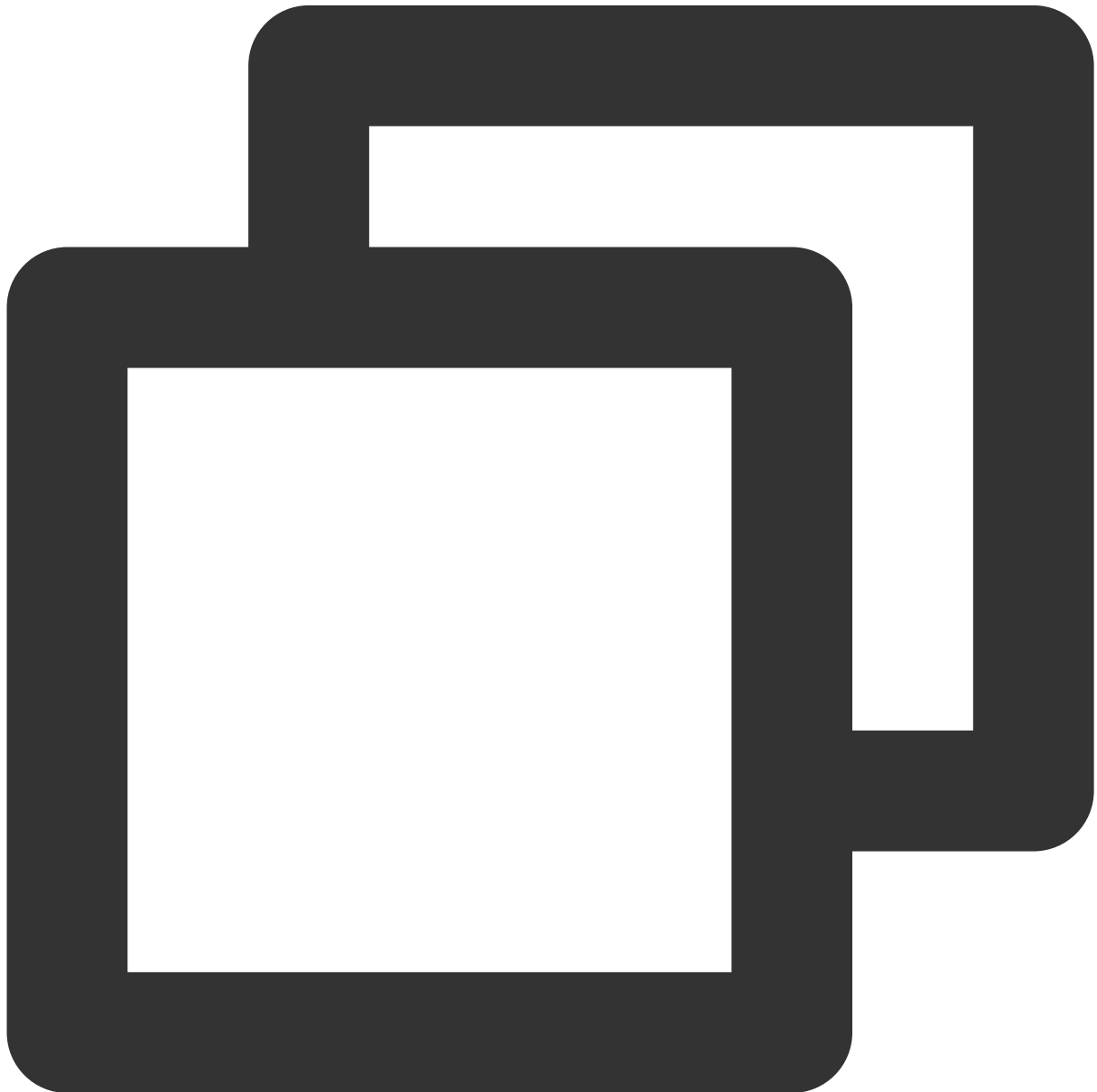
## 2, Xiaomi

(1) For versions earlier than Android 8.0, use `setAndroidSound()` and `setIOSSound()` to customize the ringtone resource file. Add the ringtone file to the project's `res/raw` directory (refer to the above Huawei invocation).

(2) For versions later than Android 8.0, you also need to sign in to the Xiaomi Vendor Console and [create a channel and configure it](#), where the ringtone file must be added to the local Android Studio project's `res/raw` directory.

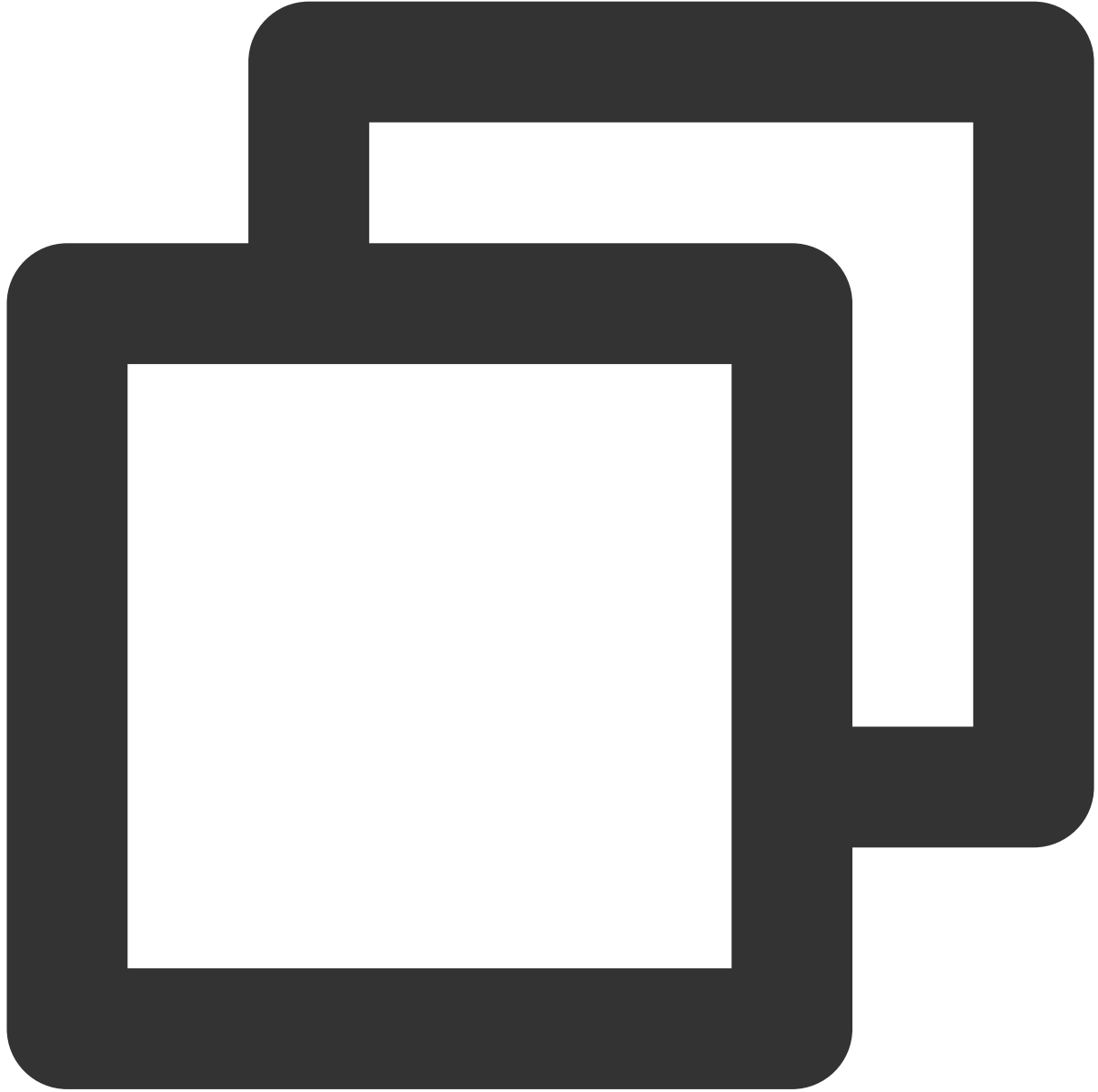
Kotlin

Java



```
val pushInfo = OfflinePushInfo()  
pushInfo.iosSound = "Ring Tone Name.mp3"
```

```
pushInfo.androidSound = "Ring Tone Name"  
pushInfo.androidXiaoMiChannelID = "Vendor Requested Channel ID"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();  
pushInfo.setIOSSound("Ring Tone Name.mp3");  
pushInfo.setAndroidSound("Ring Tone Name");  
pushInfo.setAndroidXiaoMiChannelID("Vendor Requested Channel ID");
```

**Caution**

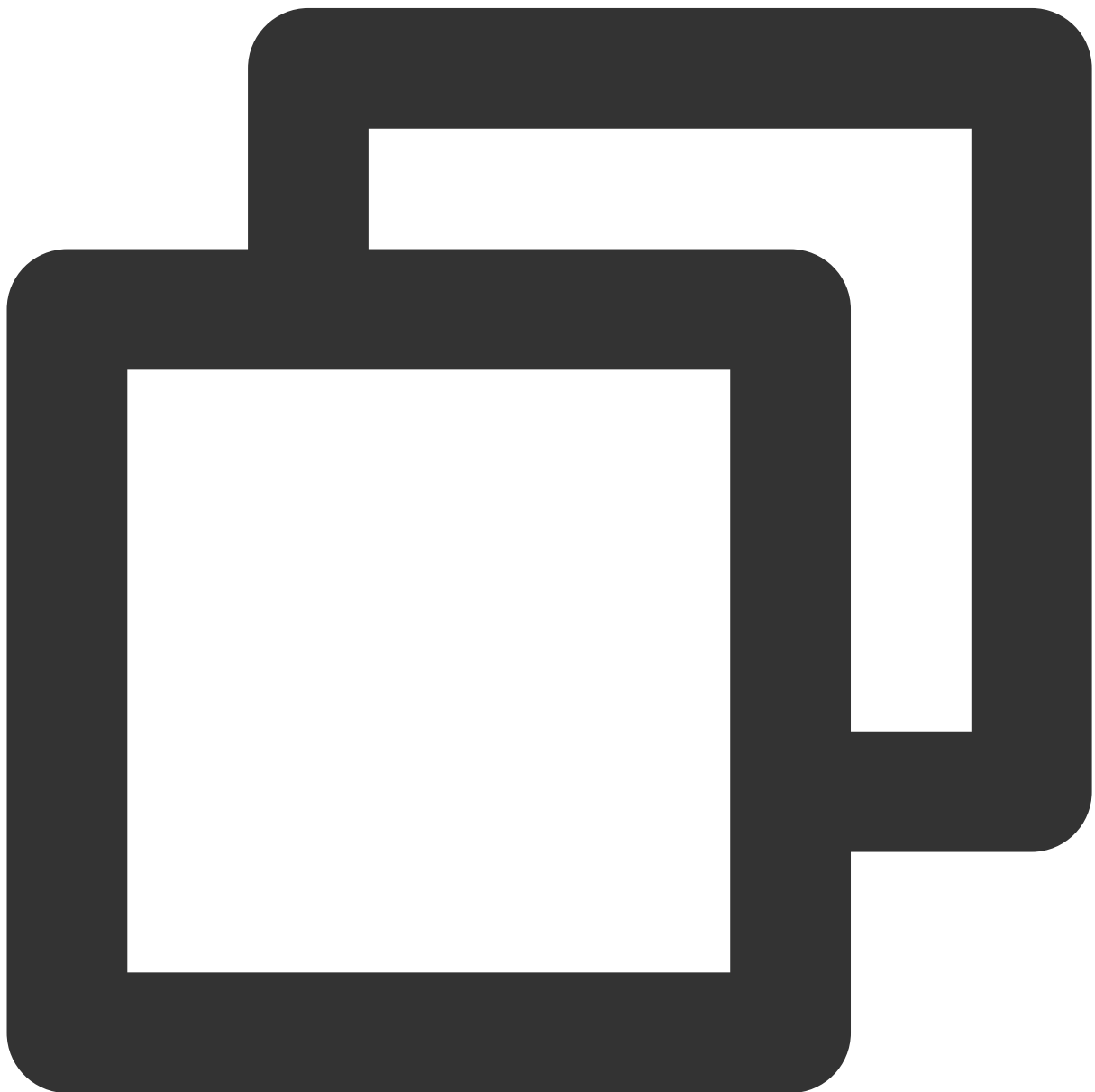
For pre-Android 8.0 methods, supported in IMSDK v6.1.2155 and later versions.

For post-Android 8.0 methods, supported in IMSDK v7.0.3754 and later versions.

### 3,FCM

(1) For versions earlier than Android 8.0, use `setAndroidSound()` and `setIOSSound()` to customize the ringtone resource file. Add the ringtone file to the project's `res/raw` directory (refer to the above Huawei invocation).

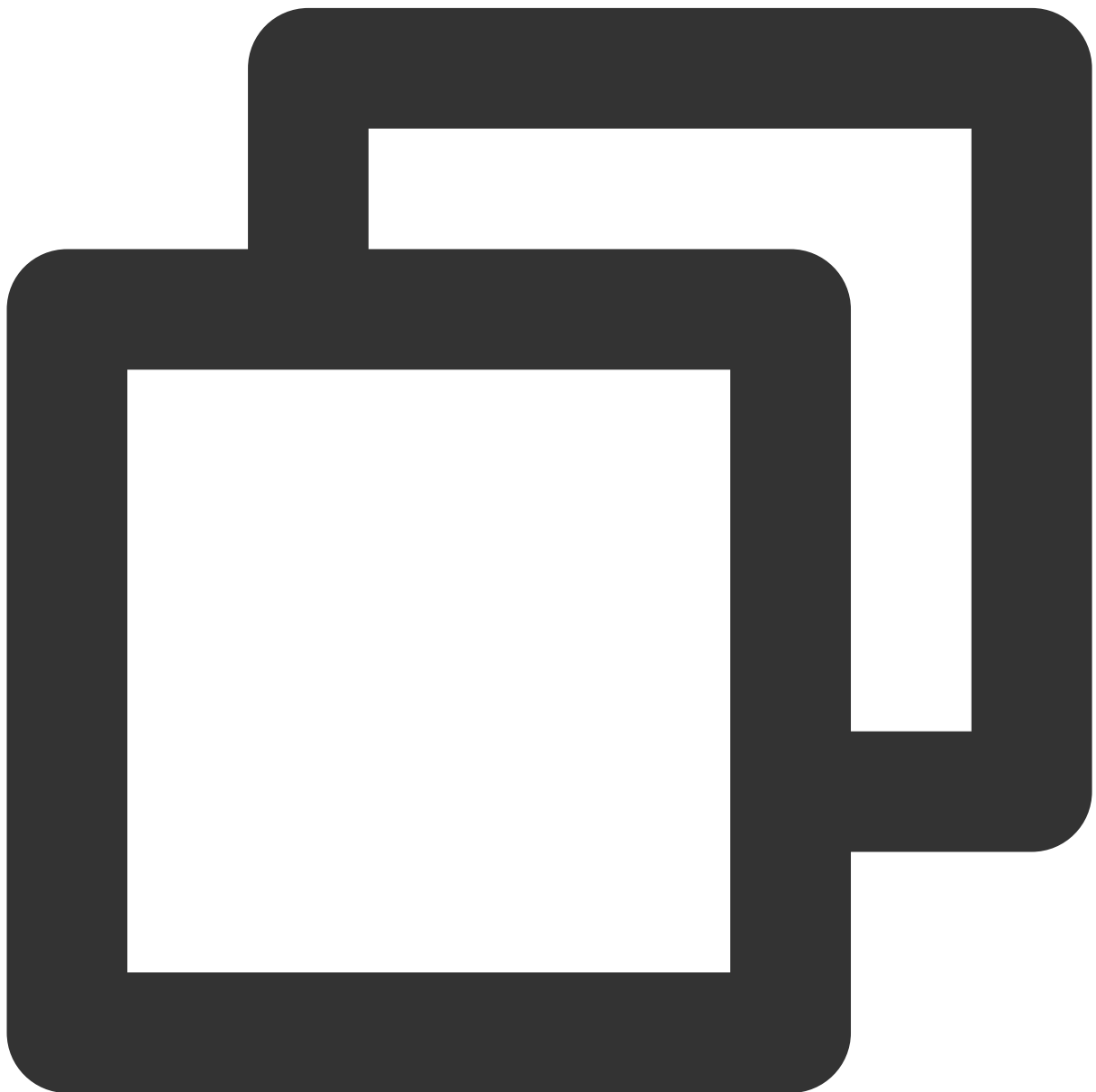
(2) For versions later than Android 8.0, FCM requires a configuration of `channelID` and ringtone resources. The `tuiofflinepush` component introduced by `TUICallKit` has already handled the playback of custom ringtones. The ringtone file needs to be added to your local Android Studio project's `res/raw` directory, specifying the ringtone name and channel ID name, see [PrivateConstants](#).



```
public class PrivateConstants {  
    // For FCM channels, specify the channel ID  
    public static String fcmPushChannelId = "FCM ChannelID";  
    // FCM requires specifying the push ringtone name for the channel, consistent w  
    public static String fcmPushChannelSoundName = "Ring Tone Name";  
}
```

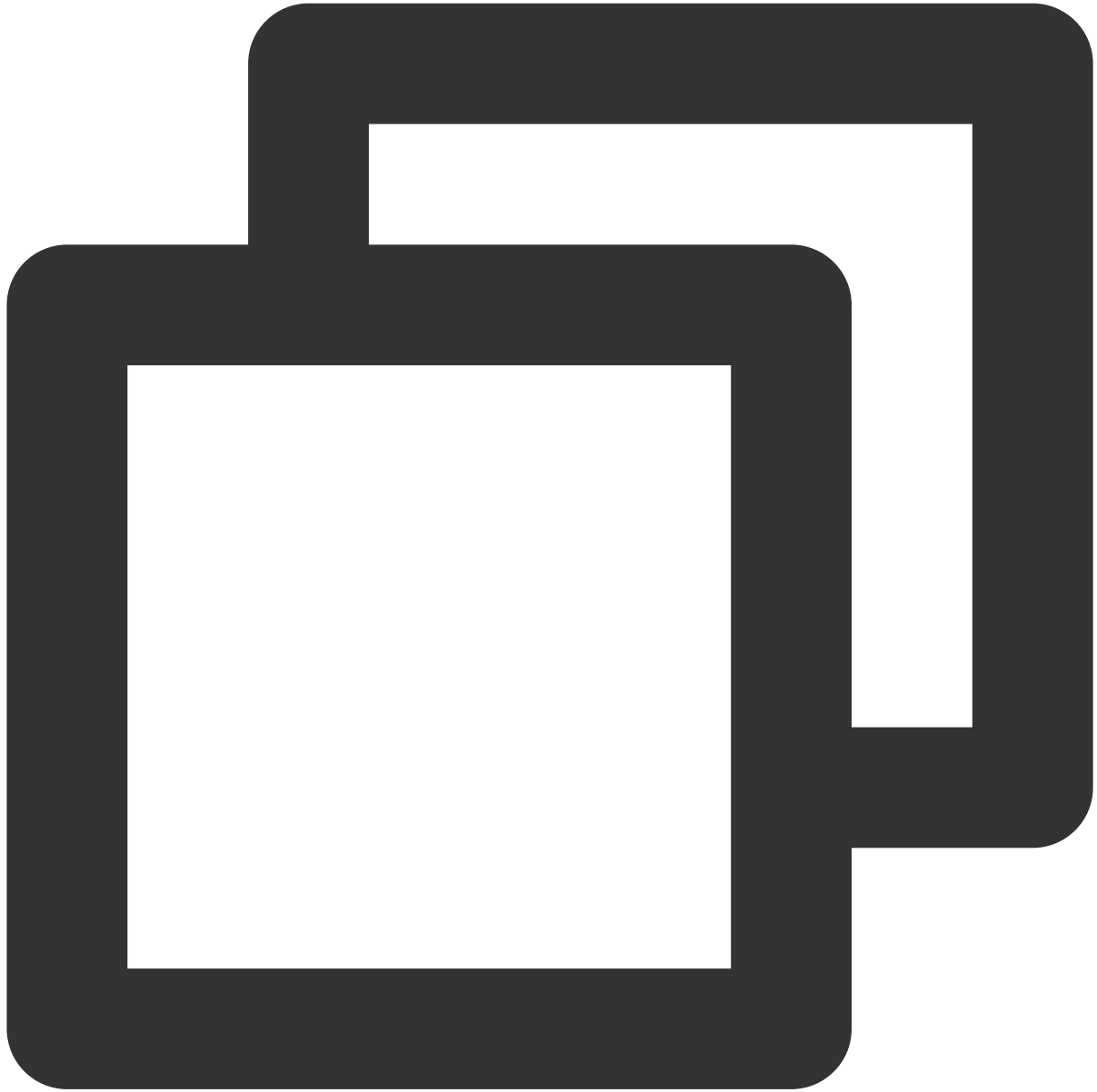
Kotlin

Java



```
val pushInfo = OfflinePushInfo()
```

```
pushInfo.iosSound = "Ring Tone Name.mp3"  
pushInfo.androidSound = "Ring Tone Name"  
pushInfo.androidFCMChannelID = "Vendor Requested Channel ID"
```



```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();  
pushInfo.setIOSSound("Ring Tone Name.mp3");  
pushInfo.setAndroidSound("Ring Tone Name");  
pushInfo.setAndroidFCMChannelID("Vendor Requested ChannelID");
```

**Caution**

For pre-Android 8.0 methods, supported in IMSDK v6.1.2155 and later versions.

---

For post-Android 8.0 methods, supported in IMSDK v7.0.3754 and later versions.  
FCM custom ringtone or Setting ChannelID is only supported in Certificate Mode.

# iOS

Last updated : 2024-05-08 11:37:24

This document describes how to replace the incoming call ringtone of TUICallKit, which is divided into **application ringtone** and **offline push ringtone**.

## 1. Setting Application Ringtone

There are two ways to set the application ringtone: replace the ringtone audio, or call the Setting Ringtone API.

### 1. Replace Audio File

If you include the TUICallKit component through source code dependency, you can replace the three audio files in the `Resources\\AudioFile` folder to achieve the purpose of changing the ringtone:

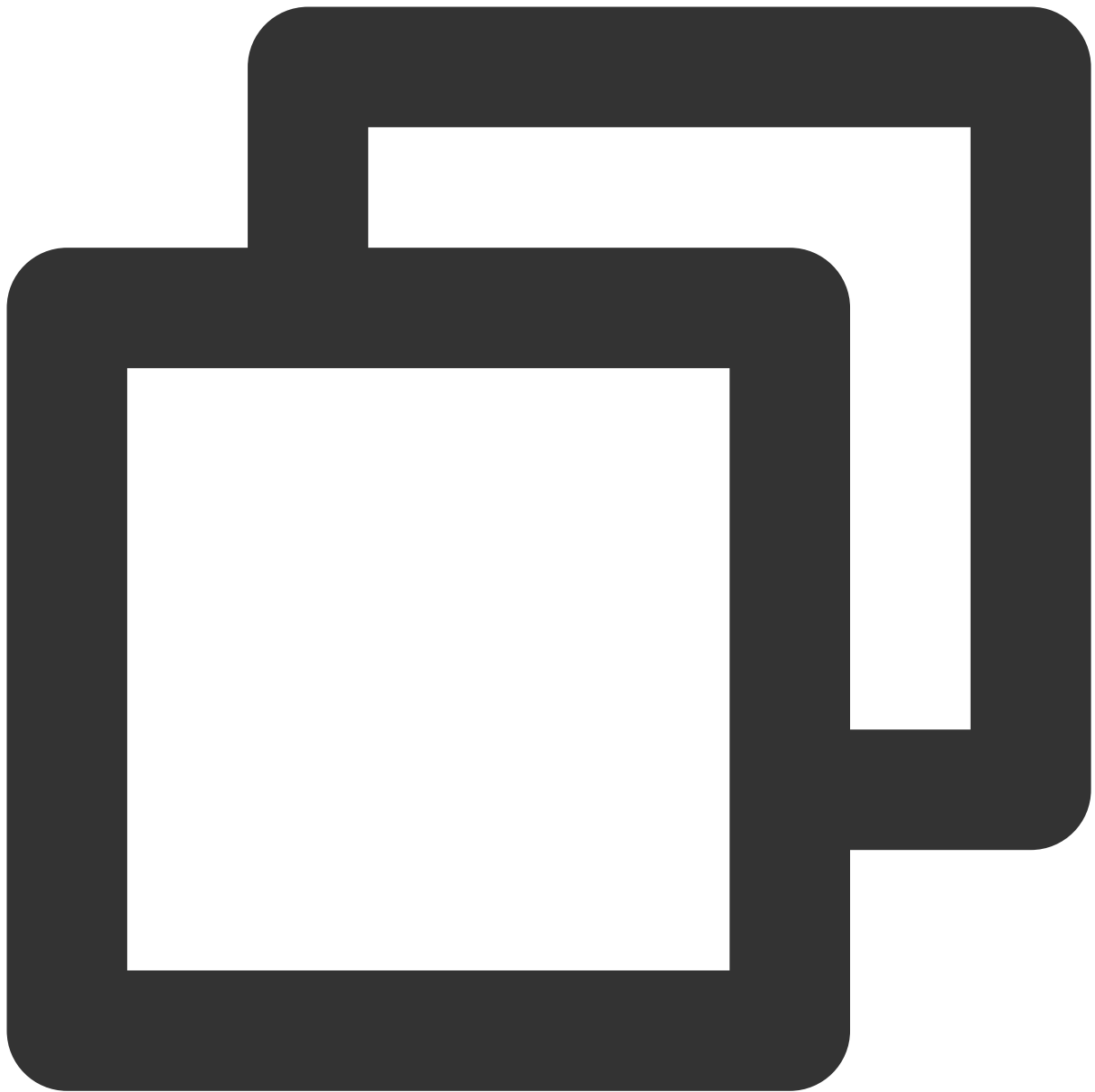
File Name	Use
phone_dialing.mp3	Ringtone when initiating a call
phone_hangup.mp3	Ringtone when the call is disconnected
phone_ringing.mp3	Ringtone when receiving a call

### 2. Set Ringtone API

You can also use the setCallingBell API to set the incoming call ringtone.

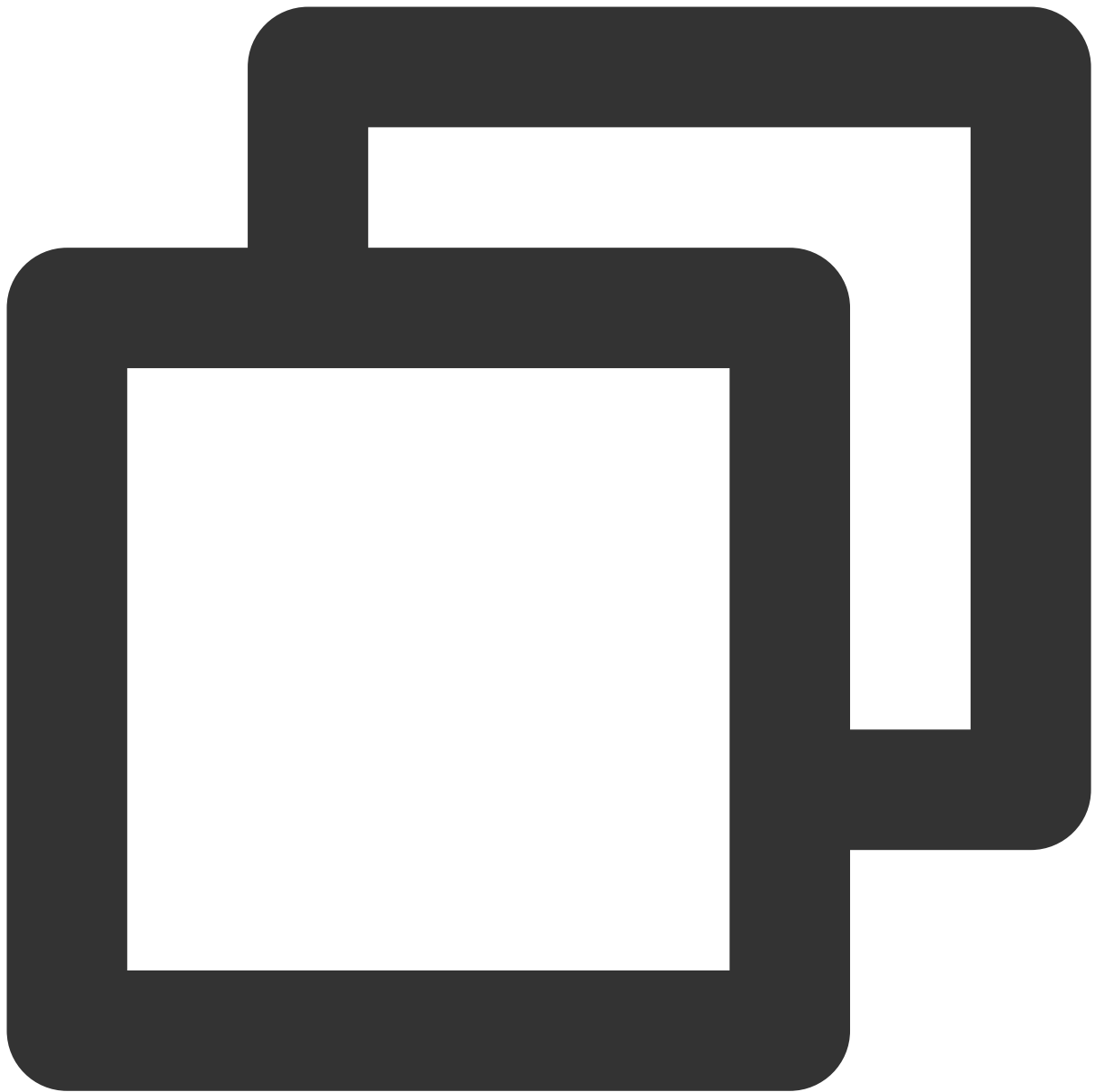
Swift

Objective-C



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





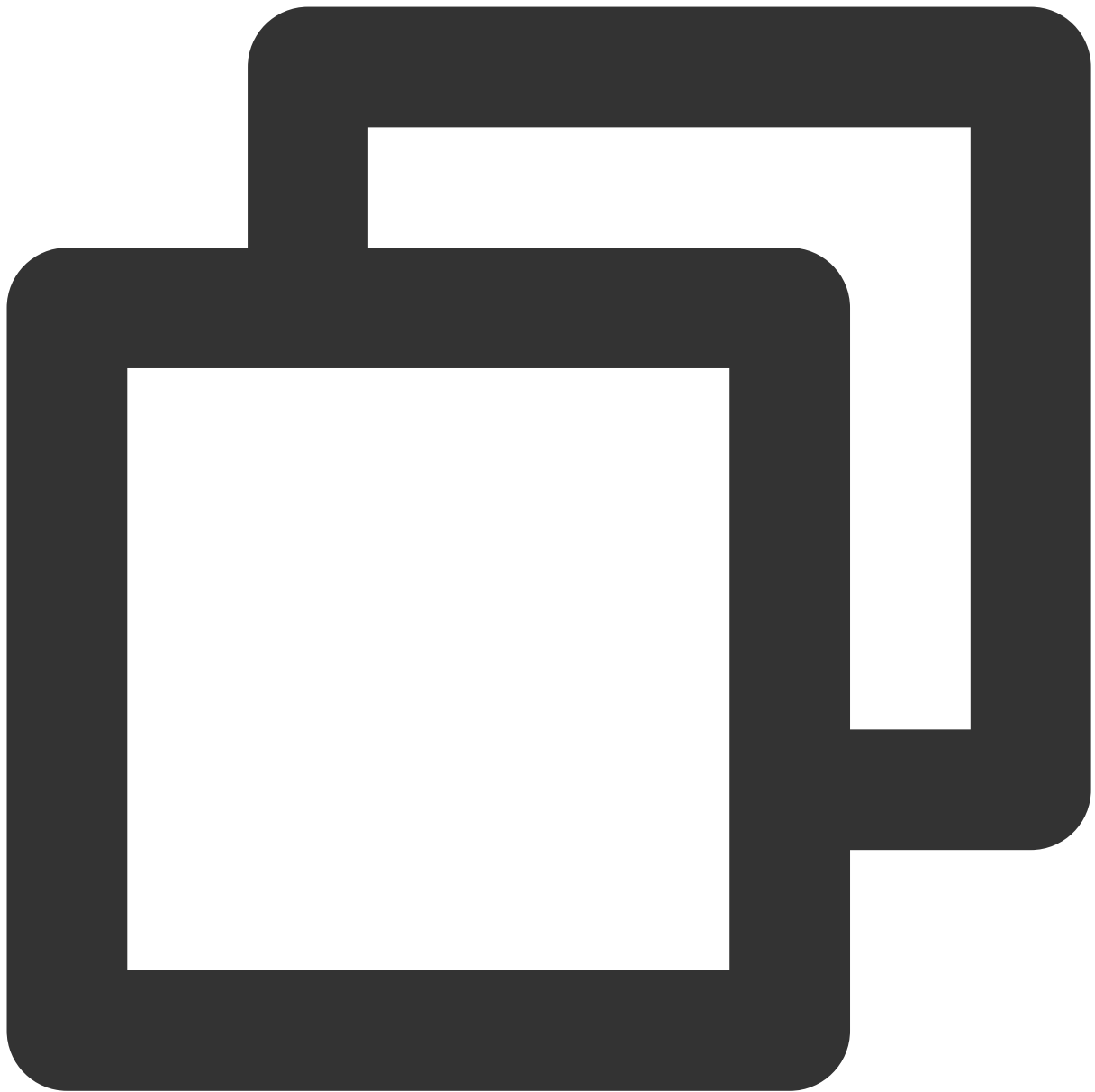
```
[[TUICallKit sharedInstance] setCallingBell:@" "];
```

### 3. Set Mute Mode

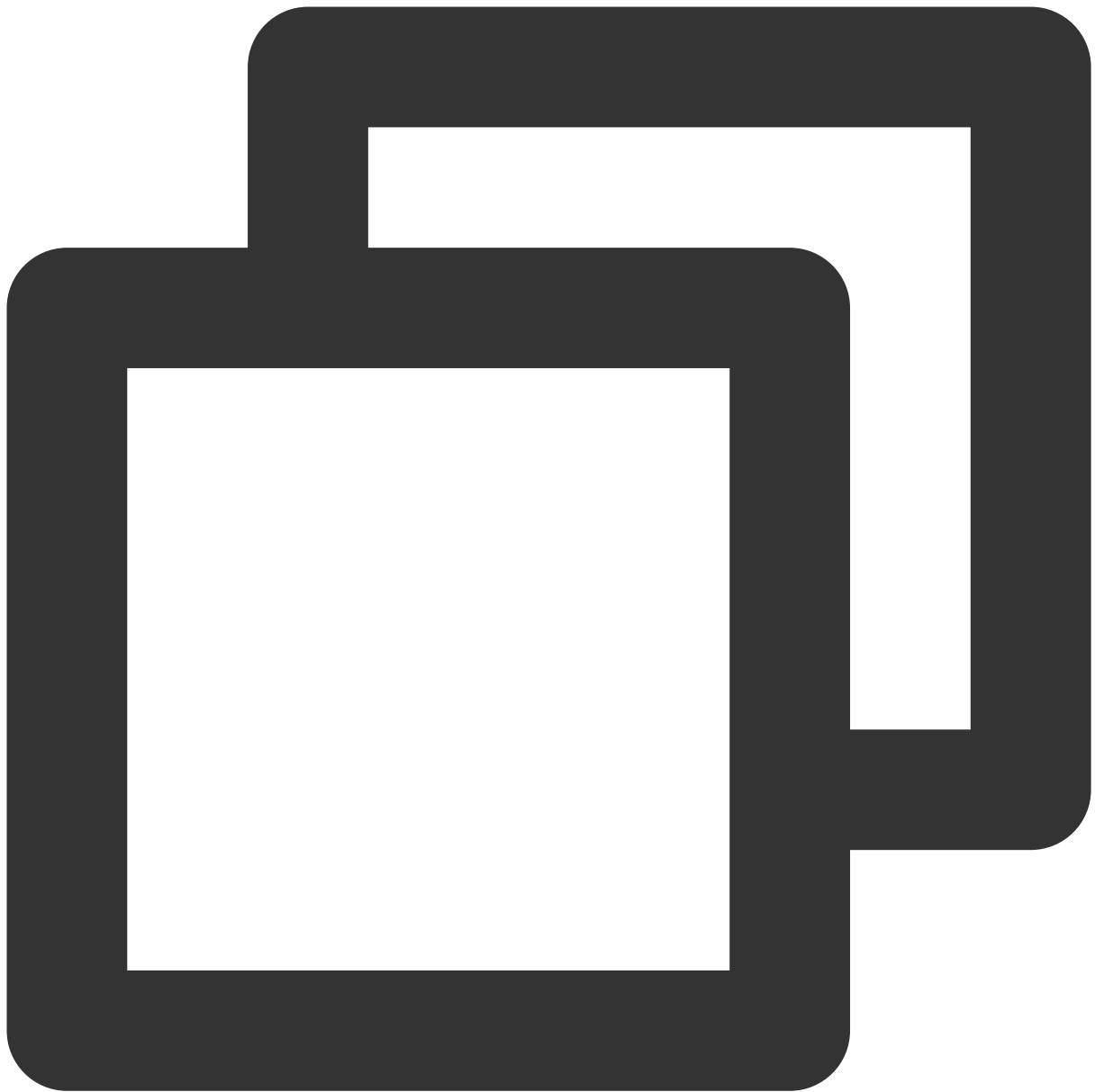
If you do not require a ringtone, you can use `enableMuteMode` to set the mute mode.

Swift

Objective-C



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



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

## II. Set Offline Push Ringtone

VoIP Push feature does not support custom push ringtones. APNs Push feature can set the `iOSSound` field in `offlinePushInfo`'s params when calling through the Call Interface, passing the audio filename as `iOSSound`.

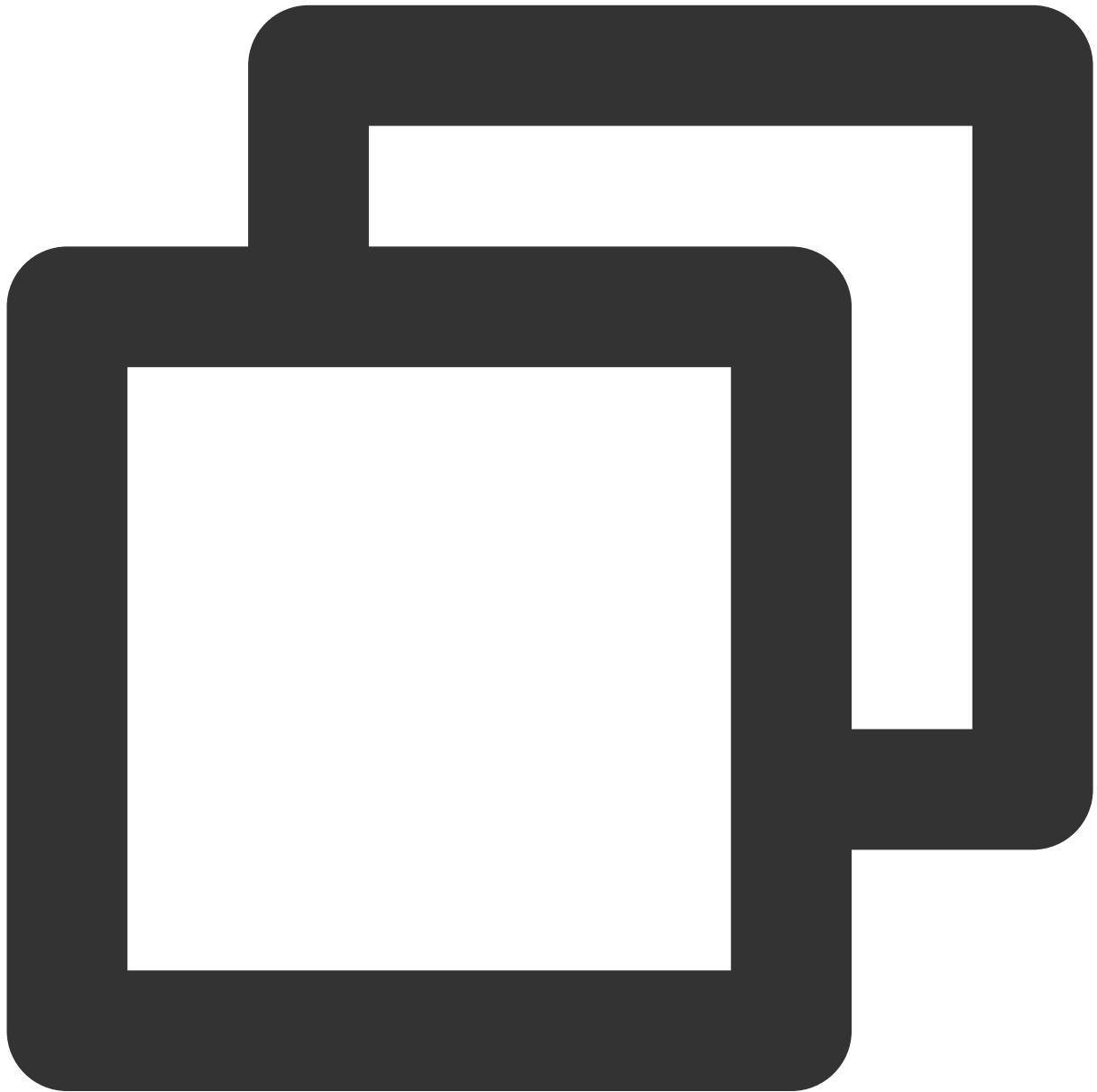
**Note:**

Offline Push Sound Setting (only effective for iOS), to customize iOSSound, you first need to link the audio file into the Xcode Project, then set the audio filename (with extension) to iOSSound.

Ringtone duration should be less than 30s.

Objective-C

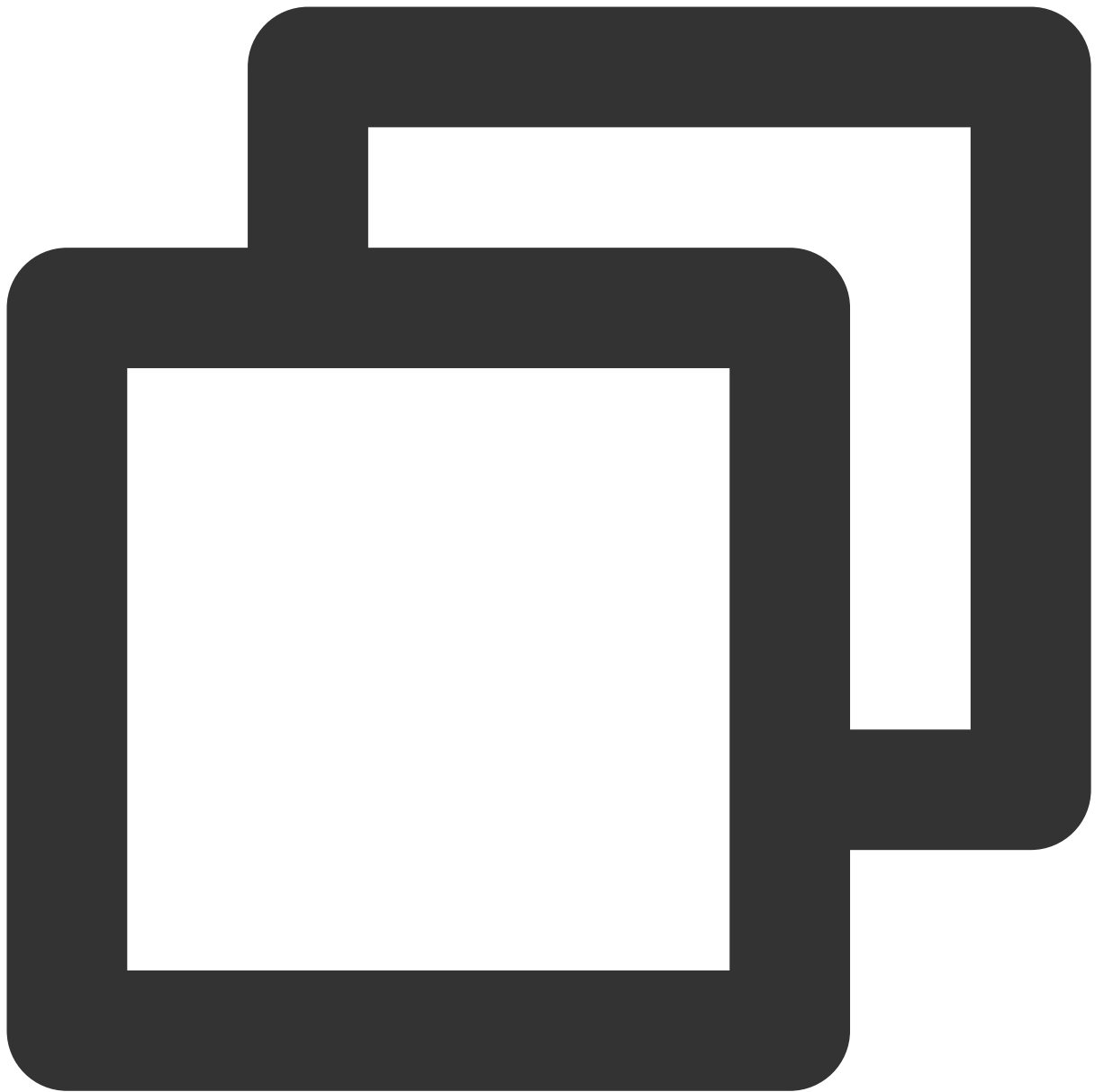
Swift



```
[[TUICallKit sharedInstance] call:@"mike's id" params:[self getCallParams] callMedia:  
- (TUICallParams *)getCallParams {
```

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

+ (TUIOfflinePushInfo *)createOfflinePushInfo {
    TUIOfflinePushInfo *pushInfo = [TUIOfflinePushInfo new];
    pushInfo.title = @"";
    pushInfo.desc = TUICallingLocalize(@"TUICallKit.have.new.invitation");
    pushInfo.iOSPushType = TUICallIOSOfflinePushTypeAPNs;
    pushInfo.ignoreIOSBadge = NO;
    pushInfo.iOSSound = @"phone_ringing.mp3";
    pushInfo.AndroidSound = @"phone_ringing";
    // For OPPO, you must set the `ChannelID` to receive push messages. The `Channe
    // OPPO must set a ChannelID to receive push messages. This channelID needs to
    pushInfo.AndroidOPPOChannelID = @"tuikit";
    // FCM channel ID, you need change PrivateConstants.java and set "fcmPushChanne
    pushInfo.AndroidFCMChannelID = @"fcm_push_channel";
    // VIVO message type: 0-push message, 1-System message(have a higher delivery r
    pushInfo.AndroidVIVOClassification = 1;
    // HuaWei message type: https://developer.huawei.com/consumer/cn/doc/developmen
    pushInfo.AndroidHuaWeiCategory = @"IM";
    return pushInfo;
}
```



```
let params = TUICallParams()
let pushInfo: TUIOfflinePushInfo = TUIOfflinePushInfo()
pushInfo.title = "TUICallKit"
pushInfo.desc = "TUICallKit.have.new.invitation"
pushInfo.iOSPushType = .apns
pushInfo.ignoreIOSBadge = false
pushInfo.iOSSound = "phone_ringing.mp3"
pushInfo.androidSound = "phone_ringing"
// For OPPO, you must set the `ChannelID` to receive push messages. The `ChannelID`
// OPPO must set a ChannelID to receive push messages. This channelId needs to be t
pushInfo.androidOPPOChannelID = "tuikit"
```

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

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

# Web&H5

Last updated : 2024-05-08 11:37:24

This document describes how to use the Definition ringtone and the silent incoming call ringtone feature.

## Setting the Incoming Call Ringtone

Only local MP3 format file addresses can be passed in, ensuring that the file is accessible.

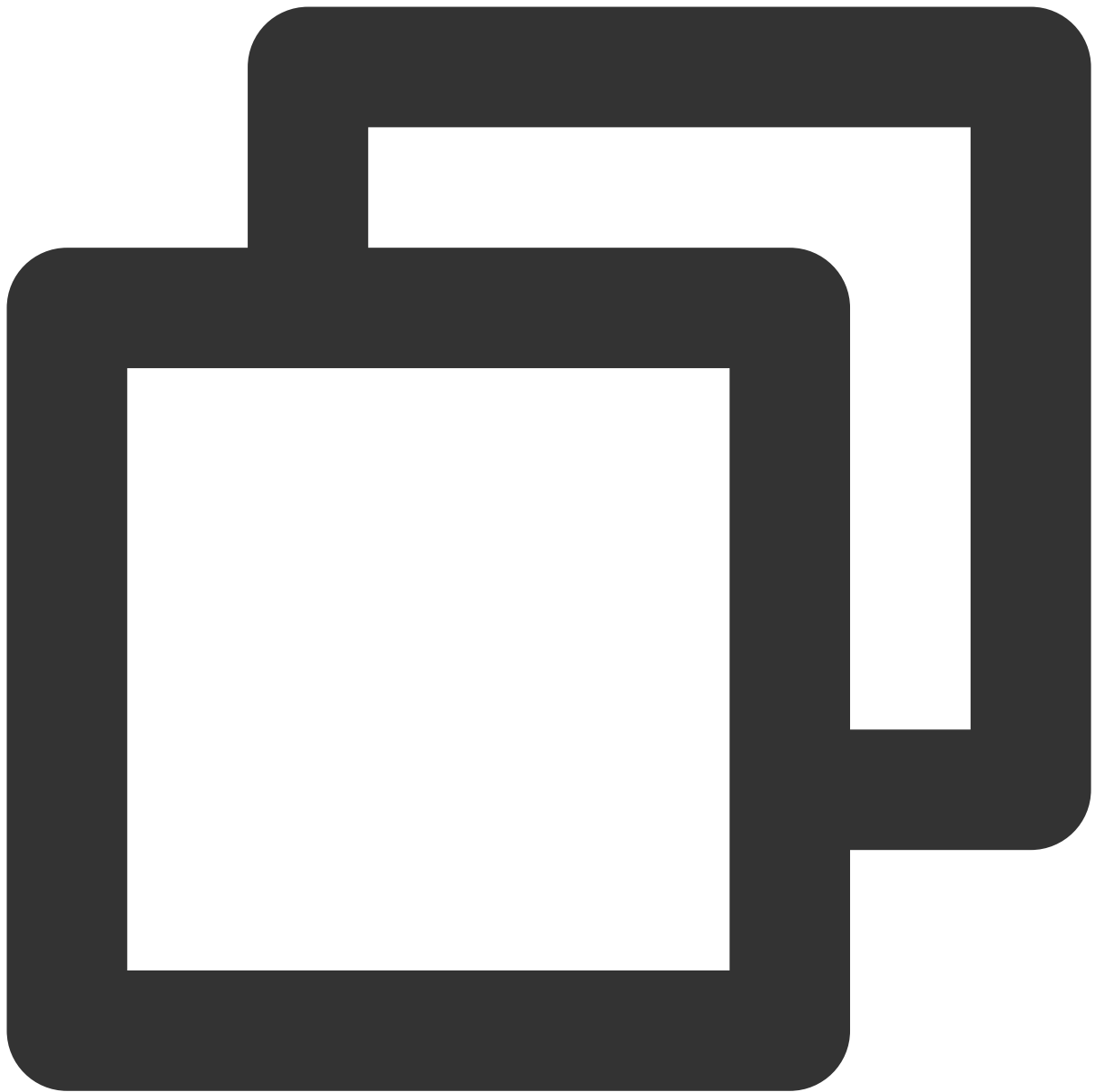
To reset to the default ringtone, pass in an empty string for filePath.

Import the ringtone file using the ES6 import method.

**Note:**

Vue v3.0.0 or later versions are supported





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

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

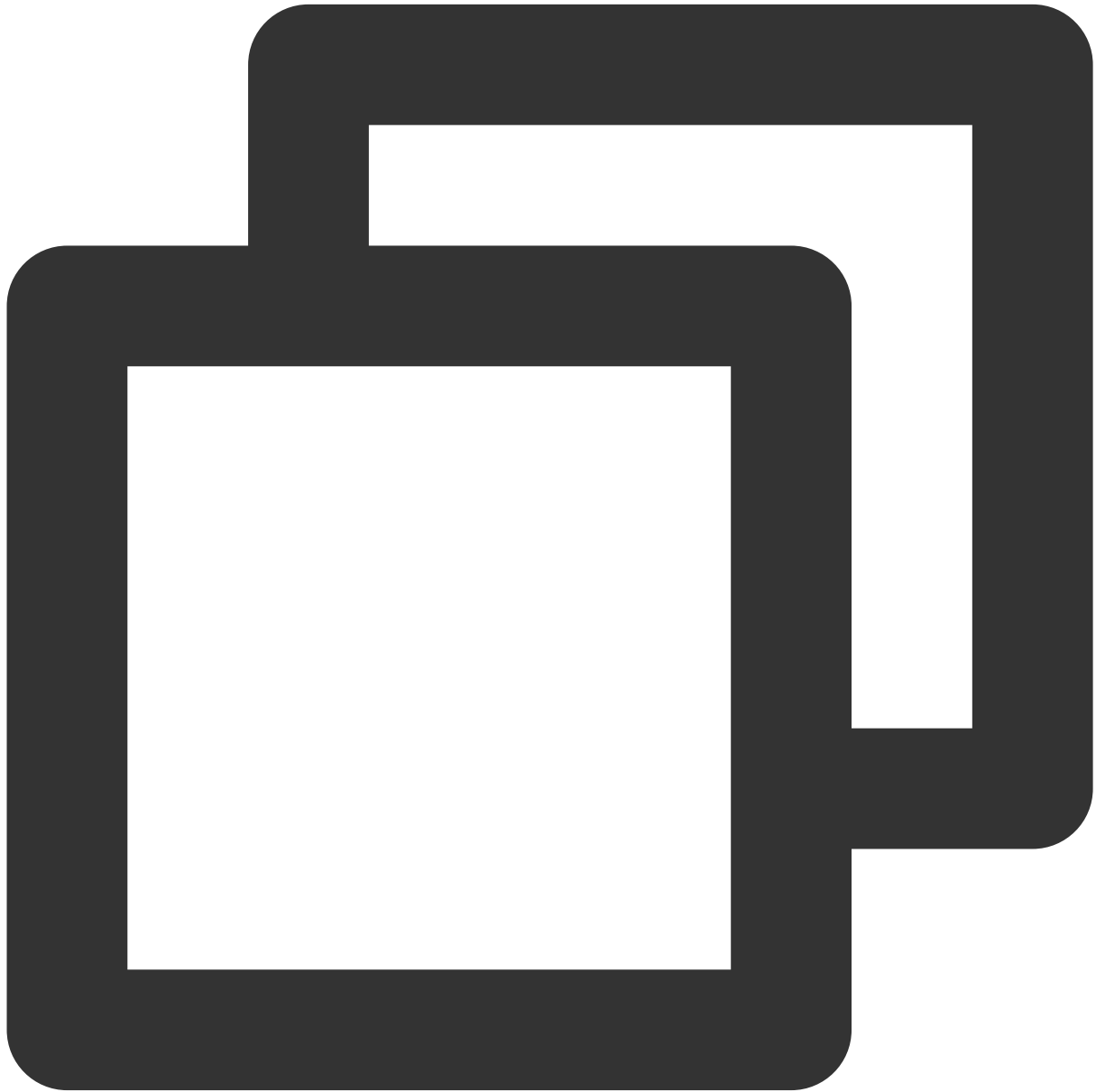
## Silent incoming call ringtone

Enable/Disable incoming call ringtone.

After enabling, the incoming call ringtone will not be played when a call request is received.

**Note:**

Vue  $\geq$  v3.1.2 is supported



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

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

# Flutter

Last updated : 2024-05-08 11:37:24

This document describes how to replace the incoming call ringtone of TUICallKit, which is divided into **Application Ringtone** and **Offline Push Ringtone**.

## First, Setting Application Ringtone

There are two ways to set the application ringtone: replace the ringtone audio, or call the Setting Ringtone API.

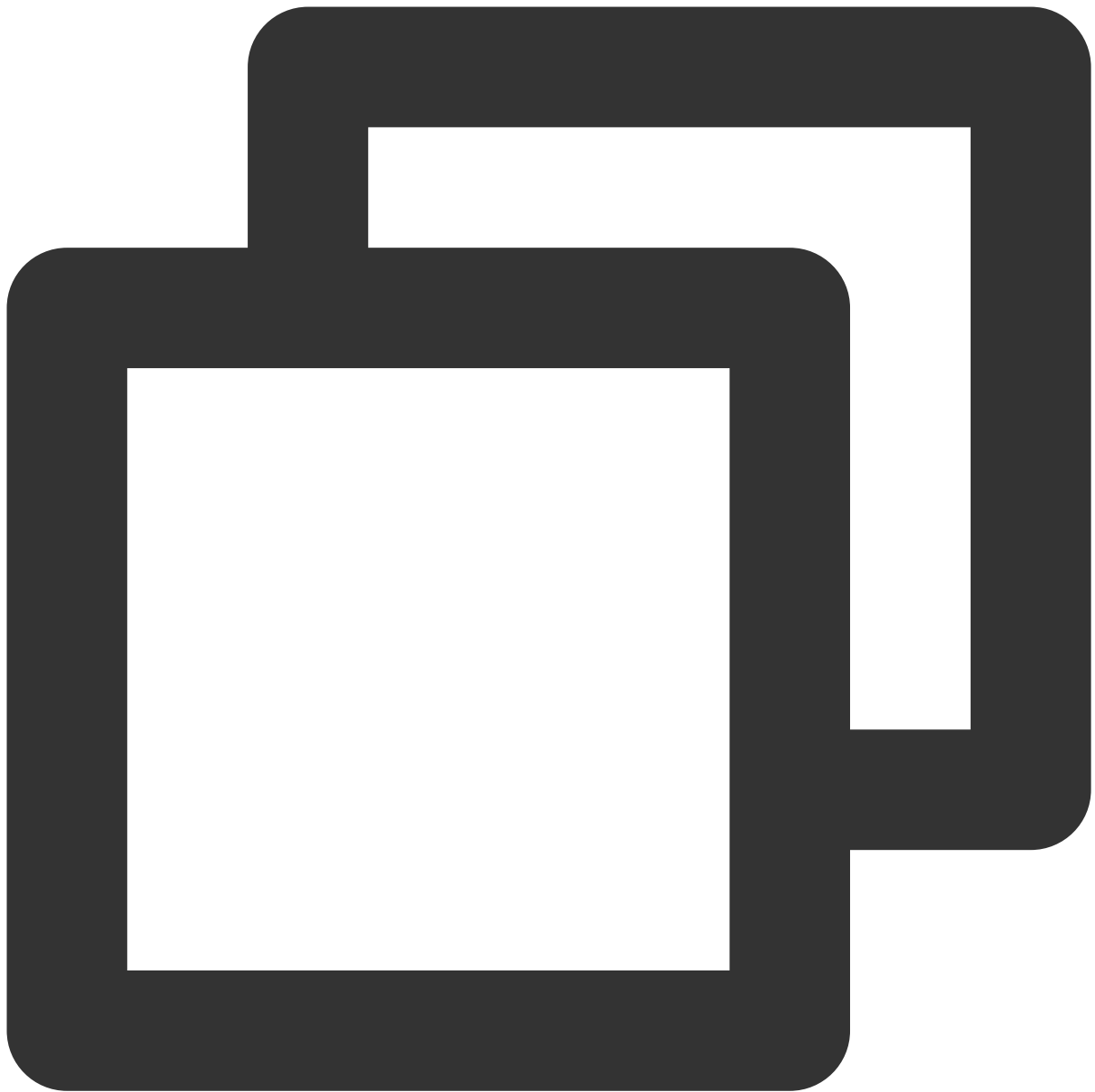
### 1, Replace Audio File

If you include the TUICallKit component via source code dependency, you can replace the three audio files under the **assets\audios** folder to achieve the purpose of changing the ringtone:

File Name	Purpose
phone_dialing.mp3	Ringtone when initiating a call
phone_hangup.mp3	Ringtone when the call is disconnected
phone_ringing.mp3	Ringtone when receiving a call

### 2, Set Ringtone API

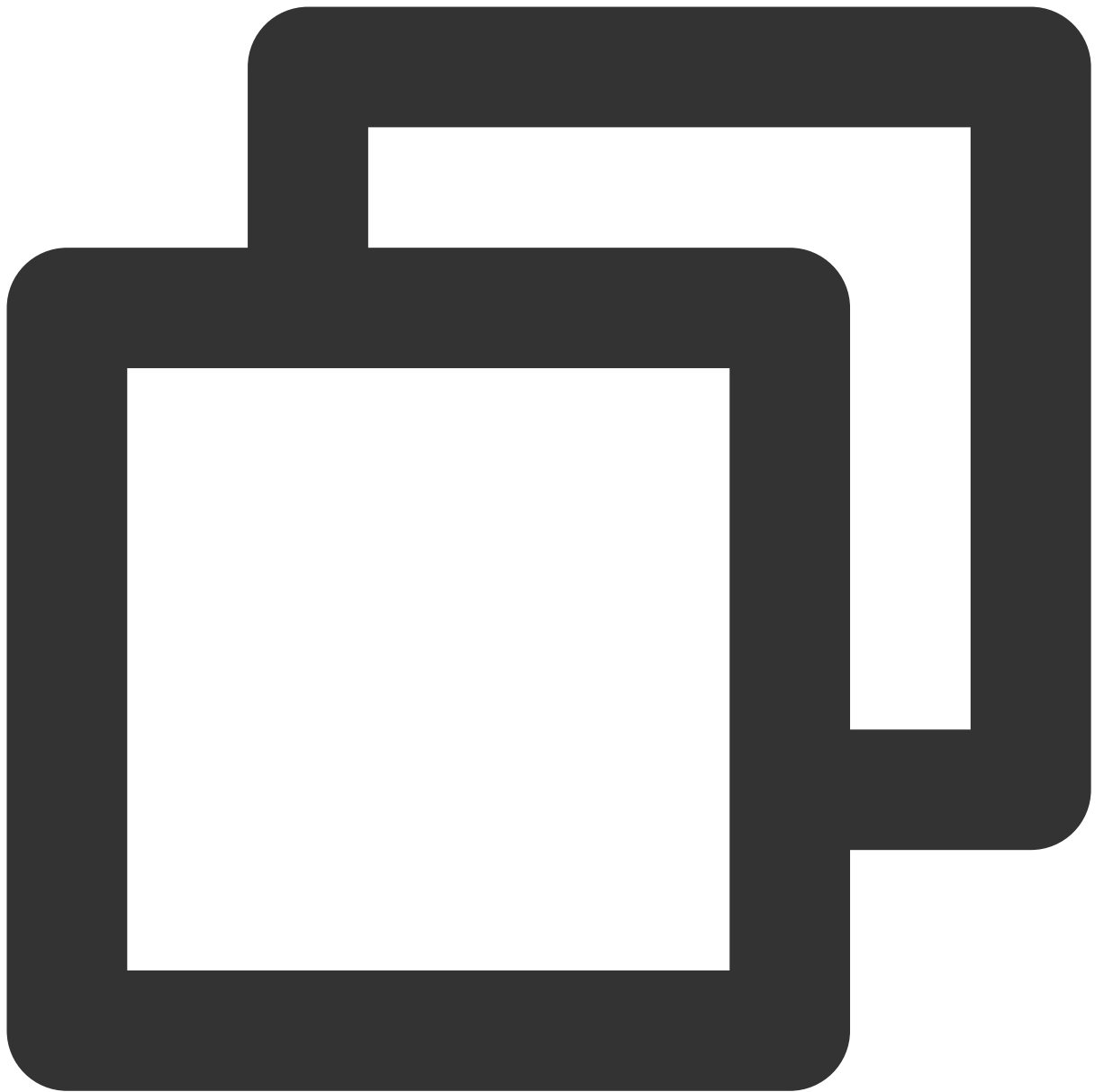
You can also use the setCallingBell interface to set the incoming call ringtone.



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

### 3, Setting the Mute Mode

If you do not require a ringtone, you can use `enableMuteMode` to set the mute mode.



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

## Second, Setting Offline Push Ringtone

### 1, iOS

VoIP push doesn't support customizing push ringtones. APNs push allows modifying the parameters in `call` and `groupcall` interfaces, under `params` including `TUIOfflinePushInfo.iOSSound` for setting offline

message ringtones on the `iOS` platform.

## 2, Android

### Note:

The API supports Huawei, Xiaomi, FCM, and APNS.

FCM's push ringtone is set as the application ringtone.

For Huawei, Xiaomi, and APNS push ringtone settings, please set the `TUIOfflinePushInfo.iOSound` 's `iOSSound` and `androidSound` fields when calling `Call` and `GroupCall`.

# Monitoring Call Status

## Android&iOS&Flutter

Last updated : 2024-05-08 11:37:24

This document describes how to use the Call Status Callback of the TUICallKit component.

## Call Status Monitoring

If your business needs to **monitor the status of calls**, such as the start and end of a call and other events during the call, you can refer to the following code:

Android(Kotlin)

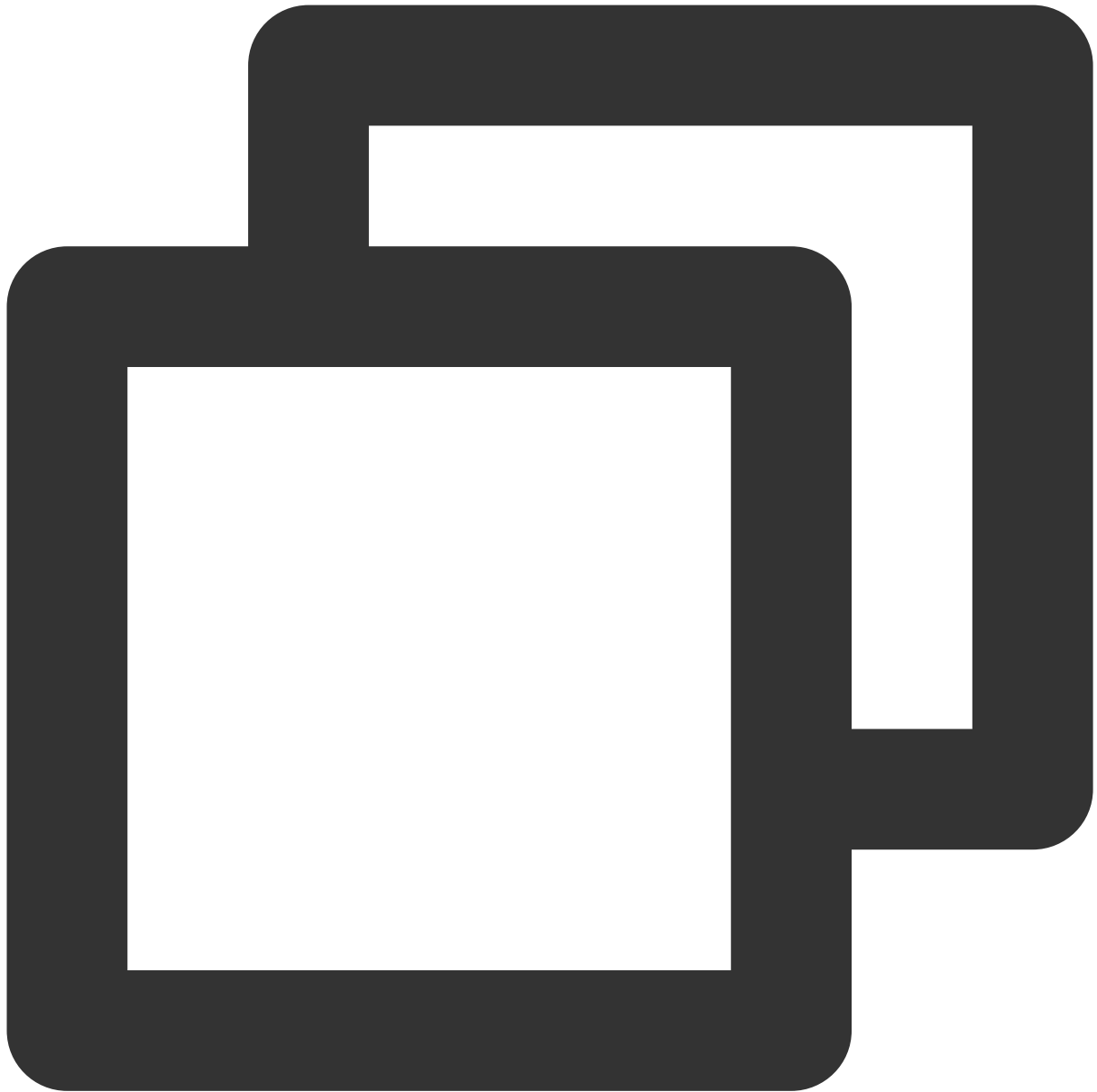
Android(Java)

iOS(Swift)

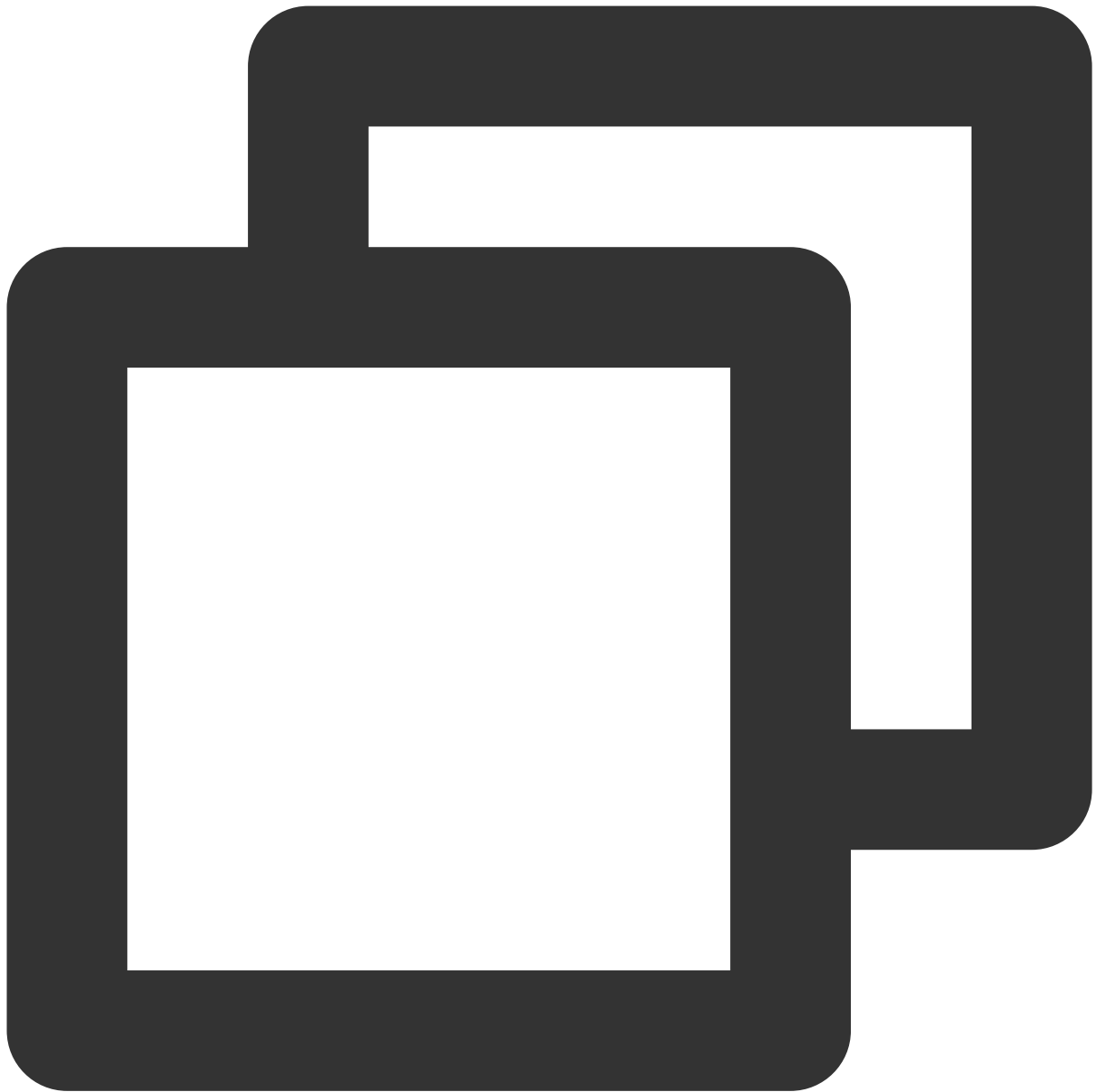
iOS(Objective-C)

Flutter(Dart)



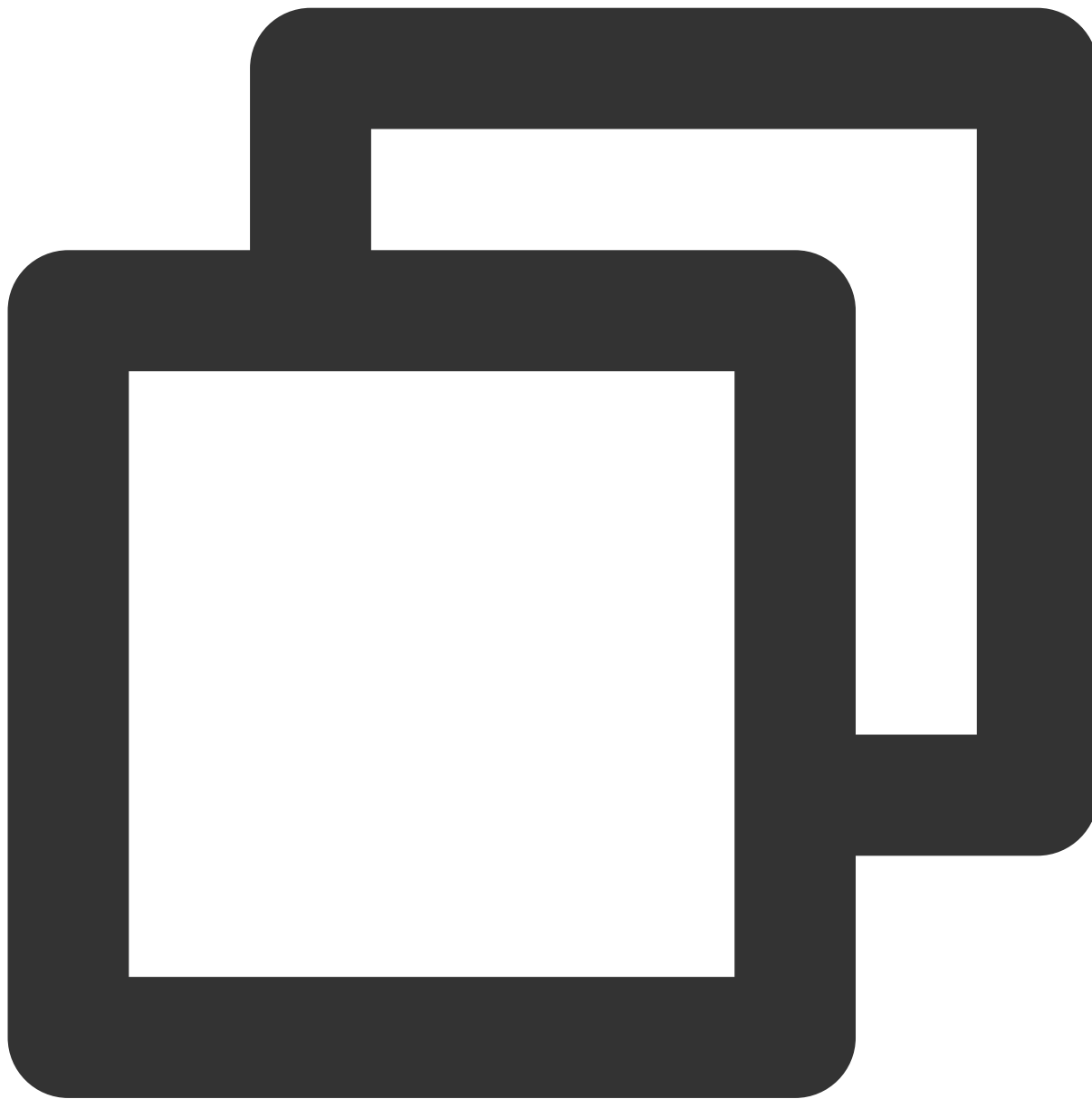


```
private val observer: TUICallObserver = object : TUICallObserver() {  
    override fun onCallBegin(roomId: TUICommonDefine.RoomId?, callMediaType: TUICal  
    }  
    override fun onCallEnd(roomId: TUICommonDefine.RoomId?, callMediaType: TUICalld  
    }  
    override fun onUserNetworkQualityChanged(networkQualityList: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



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

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



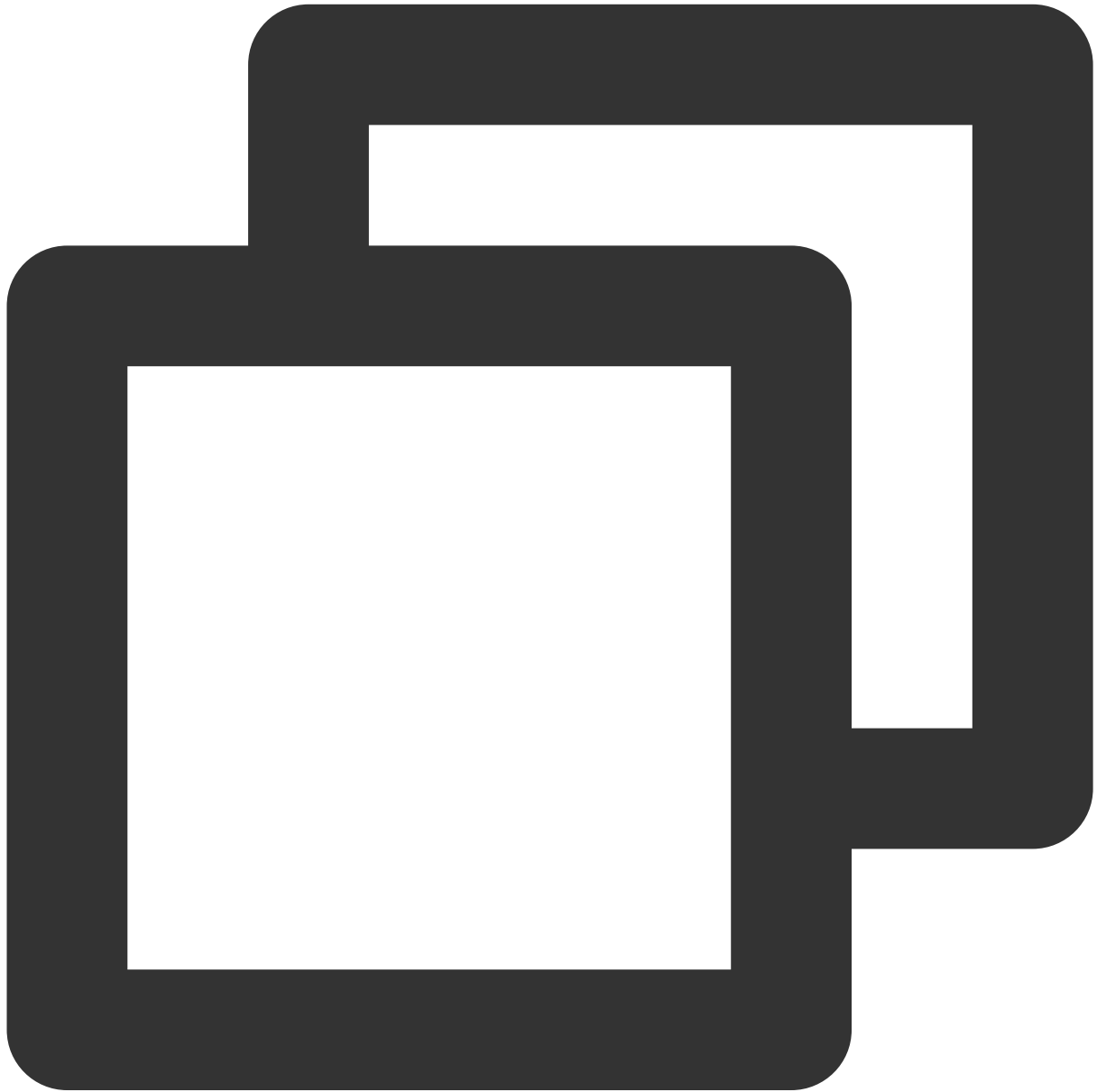
```
import TUICallEngine

TUICallEngine.createInstance().addObserver(self)

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

}
```

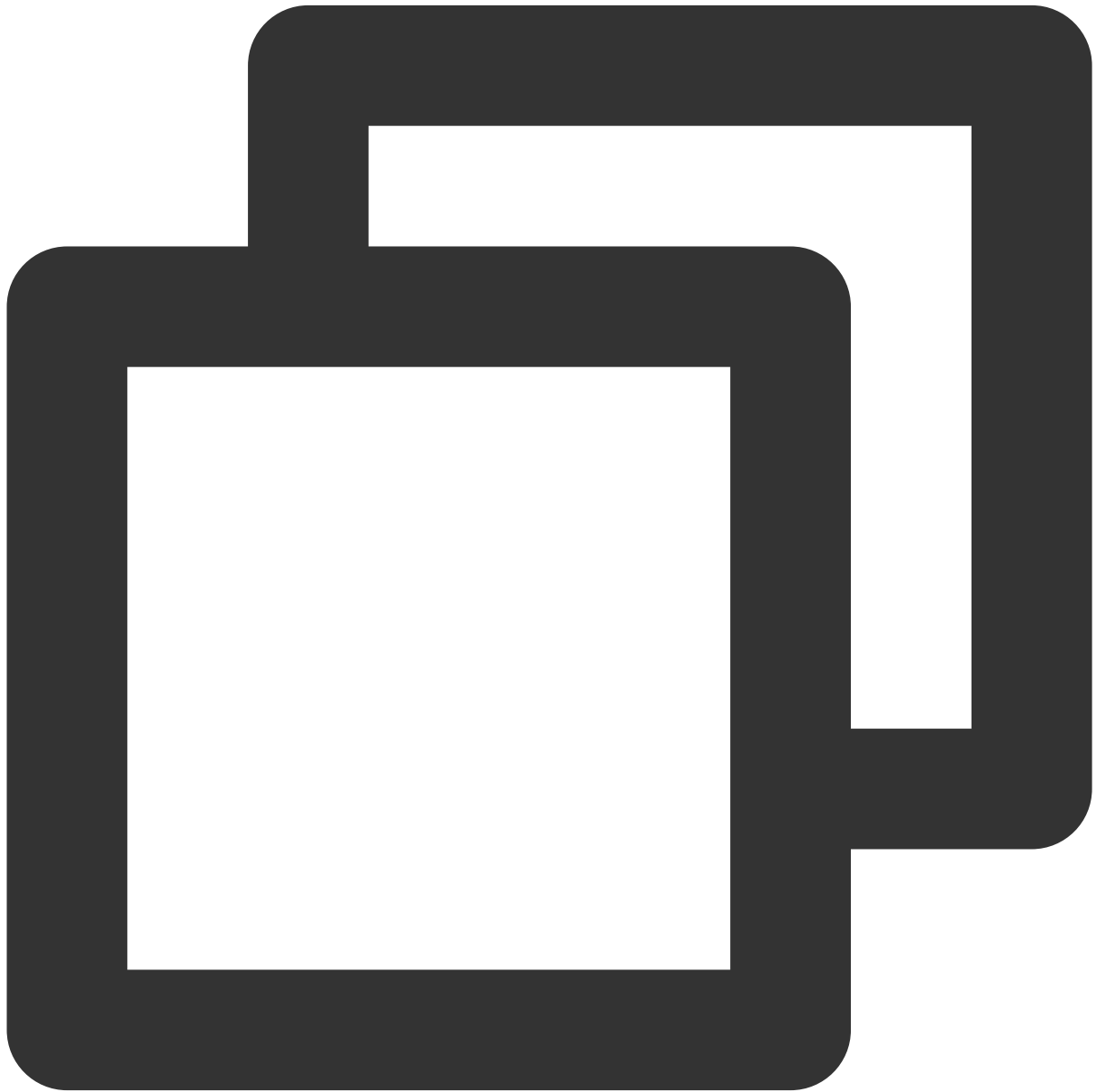
```
public func onCallEnd(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRole:
}
public func onUserNetworkQualityChanged(networkQualityList: [TUINetworkQualityInfo]
}
```



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

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

```
- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaTy  
}  
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType  
}  
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQual  
}
```



```
TUICallObserver observer = TUICallObserver(
```

```
onError: (int code, String message) {
    // Your callback handling logic
}, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
    // Your callback handling logic
}, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole ca
    // Your callback handling logic
},, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityLis
    // Your callback handling logic
}, onCallReceived: (String callerId, List<String> calleeIdList, String groupId,
    // Your callback handling logic
}
...
))

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

**Note:**

On the Android platform, when setting TUICallObserver to monitor callbacks, ensure the callback's class is not cleared. For example, it's not recommended to add monitoring in LoginActivity because the callback will be cleared when LoginActivity is terminated; it's advised to monitor in the Application class or the main interface.

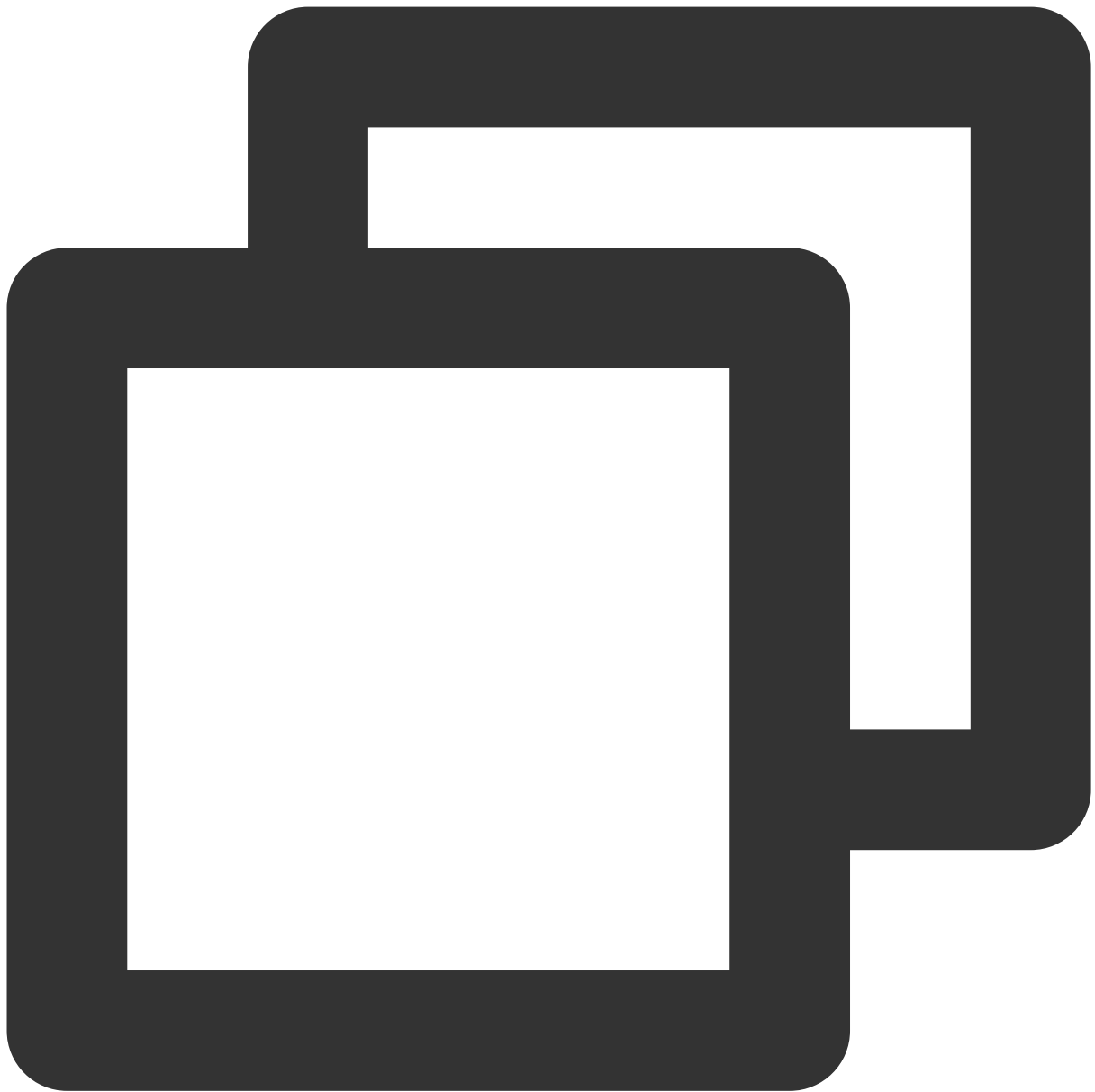
# Web&H5

Last updated : 2024-05-08 11:37:24

This document describes how to use the Call Status Callback of the TUICallKit component.

## Call Status Monitoring

If your business needs to **monitor the status of calls**, such as events during the call process including the start and end of calls (see [TUICallEvent](#) for details), refer to the following code:



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

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

TUICallKitServer.getTUICallEngineInstance().on(TUICallEvent.USER_ENTER, handleUserEnter);
TUICallKitServer.getTUICallEngineInstance().off(TUICallEvent.USER_ENTER, handleUserEnter);
```



## Component Callback Event

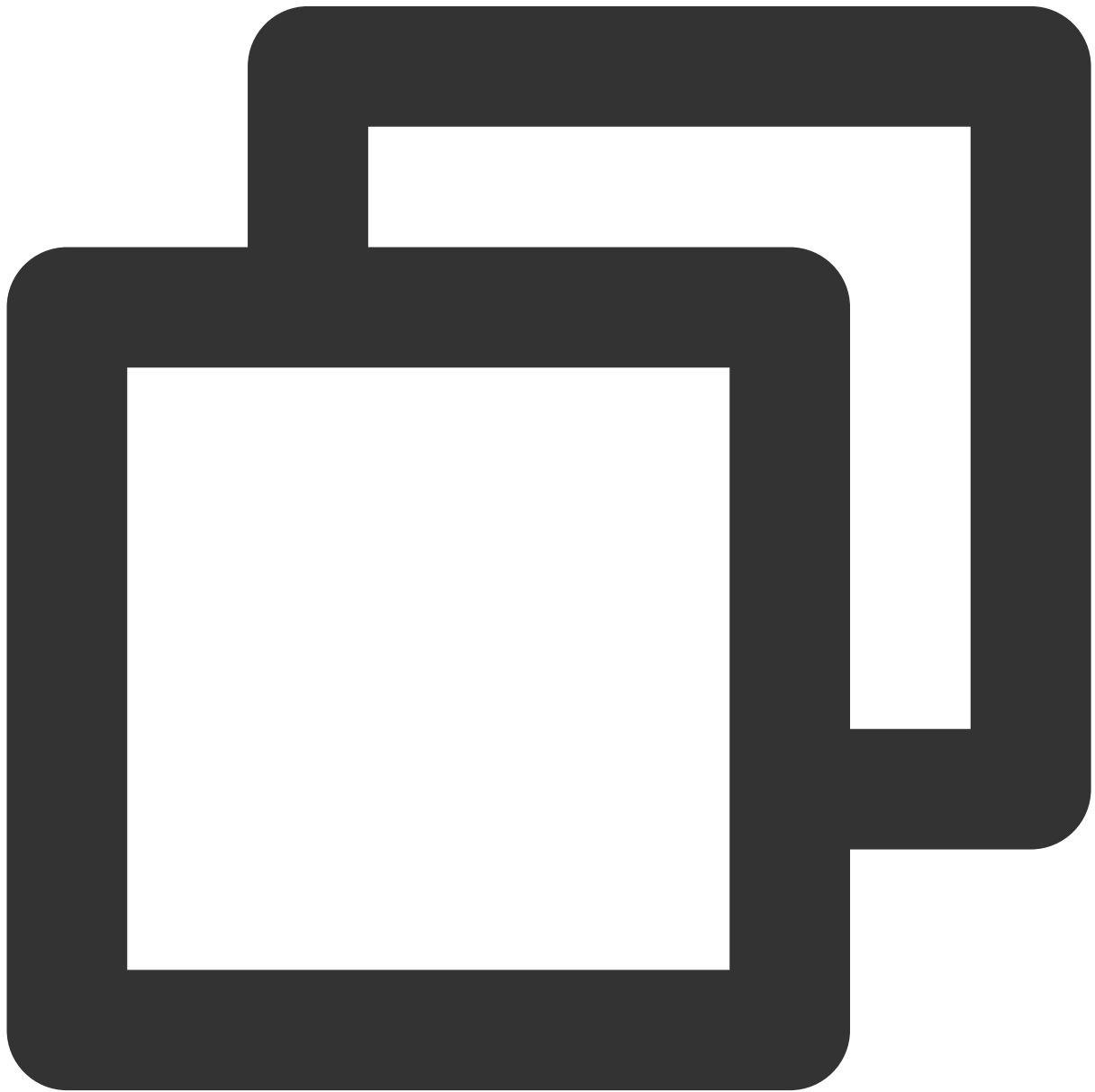
The TUICallKit component provides call status callbacks, which can be used for more interaction logic on the business side. For details, see [Introduction to TUICallKit Attributes](#).

`beforeCalling`: Executed before the call starts.

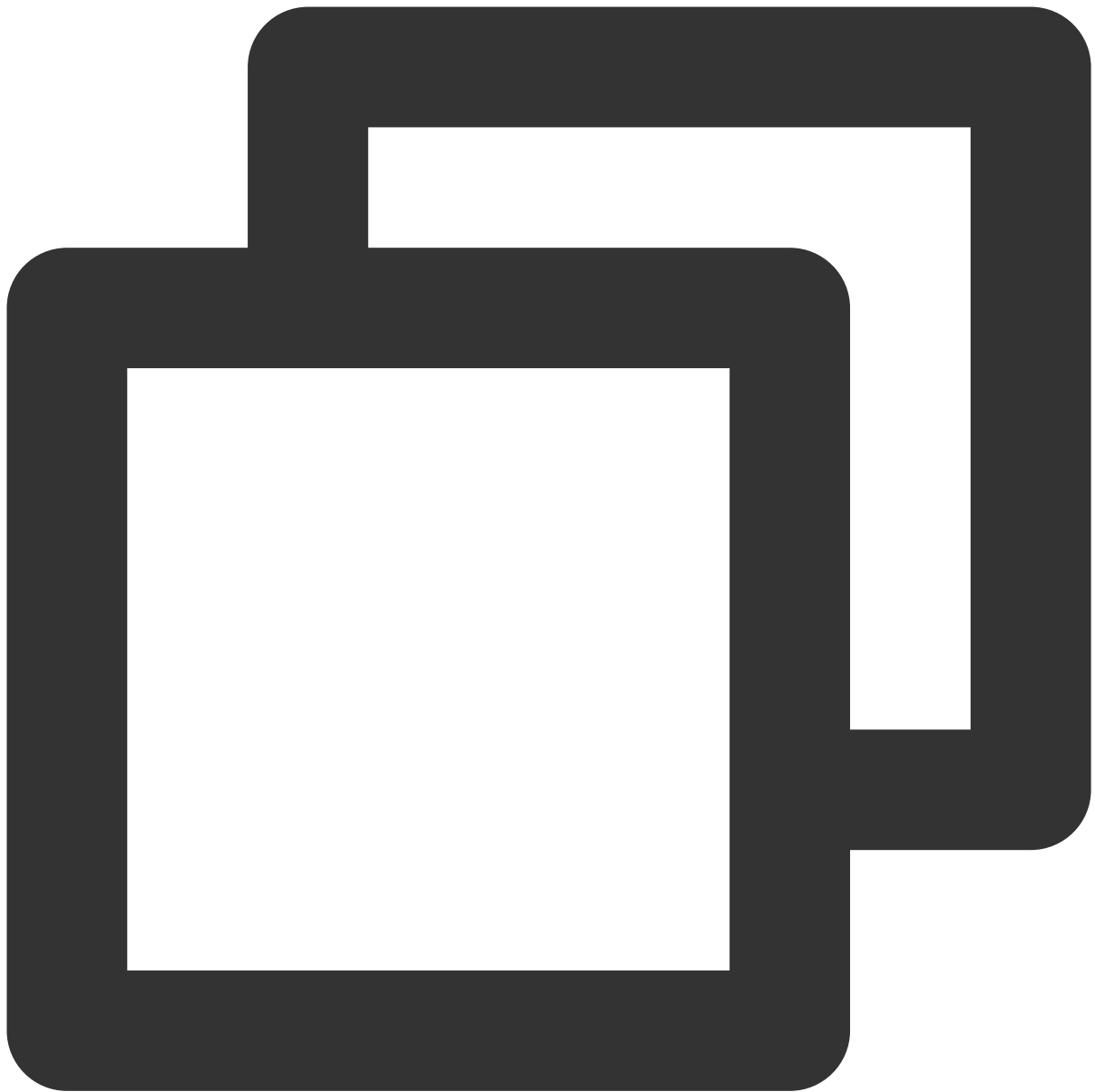
`afterCalling`: Executed after the call ends.

React

Vue



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



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

```
}  
</script>
```

# API Documentation(TUICallKit)

## Android

## API Overview

Last updated : 2024-07-30 11:19:49

### 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
<a href="#">createInstance</a>	Creates a TUICallKit instance (singleton mode).
<a href="#">setSelfInfo</a>	Sets the user nickname and profile photo.
<a href="#">call</a>	Makes a one-to-one call.
<a href="#">call</a>	Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">groupCall</a>	Makes a group call.
<a href="#">groupCall</a>	Makes a group call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">joinInGroupCall</a>	Joins a group call.
<a href="#">setCallingBell</a>	Sets the ringtone.
<a href="#">enableMuteMode</a>	Sets whether to turn on the mute mode.
<a href="#">enableFloatWindow</a>	Sets whether to enable floating windows.
<a href="#">enableIncomingBanner</a>	Sets whether to enable the incoming banner.

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

<a href="#">setVideoRenderParams</a>	Set the rendering mode of video image.
<a href="#">setVideoEncoderParams</a>	Set the encoding parameters of video encoder.
<a href="#">getTRTCCloudInstance</a>	Advanced features.
<a href="#">setBeautyLevel</a>	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
<a href="#">onError</a>	An error occurred during the call.
<a href="#">onCallReceived</a>	A call was received.
<a href="#">onCallCancelled</a>	The call was canceled.
<a href="#">onCallBegin</a>	The call was connected.
<a href="#">onCallEnd</a>	The call ended.
<a href="#">onCallMediaTypeChanged</a>	The call type changed.
<a href="#">onUserReject</a>	A user declined the call.
<a href="#">onUserNoResponse</a>	A user didn't respond.
<a href="#">onUserLineBusy</a>	A user was busy.
<a href="#">onUserJoin</a>	A user joined the call.
<a href="#">onUserLeave</a>	A user left the call.
<a href="#">onUserVideoAvailable</a>	Whether a user has a video stream.
<a href="#">onUserAudioAvailable</a>	Whether a user has an audio stream.
<a href="#">onUserVoiceVolumeChanged</a>	The volume levels of all users.
<a href="#">onUserNetworkQualityChanged</a>	The network quality of all users.
<a href="#">onKickedOffline</a>	The current user was kicked offline.
<a href="#">onUserSigExpired</a>	The user sig is expired.

## Definitions of Key Types

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



# TUICallKit

Last updated : 2024-05-27 17:41:07

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

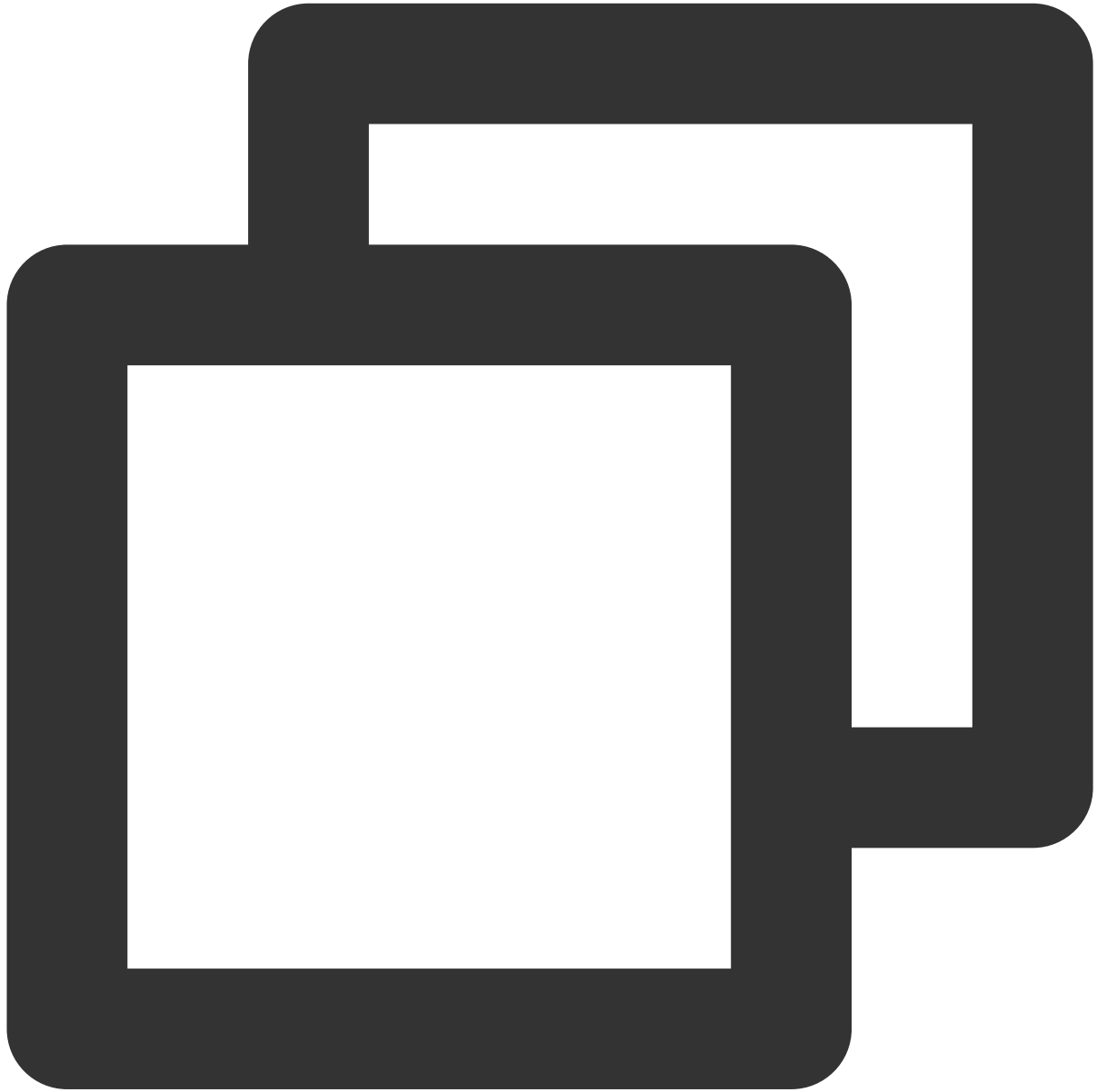
## Details

### createInstance

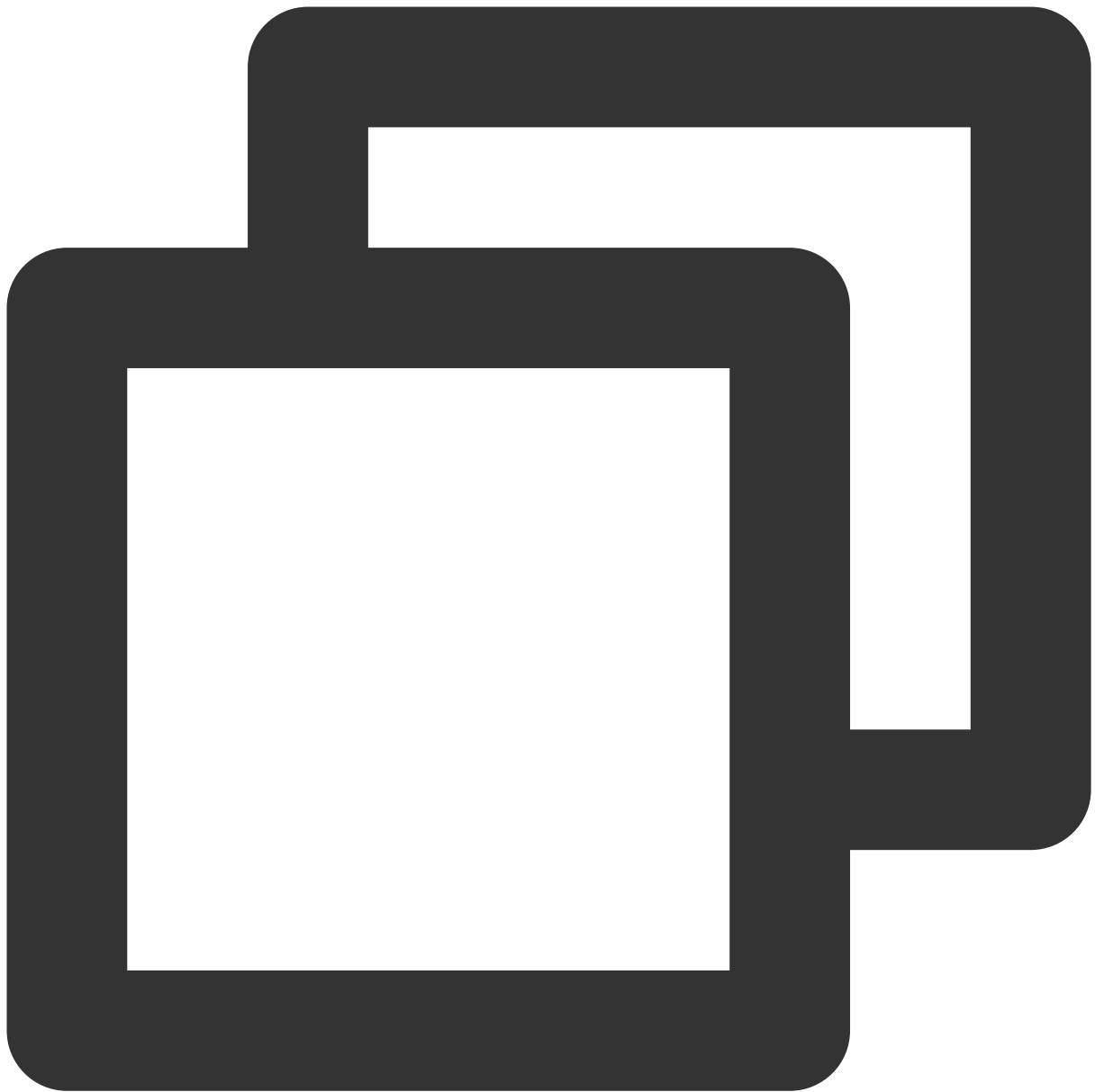
This API is used to create a `TUICallKit` singleton.

Kotlin

Java



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



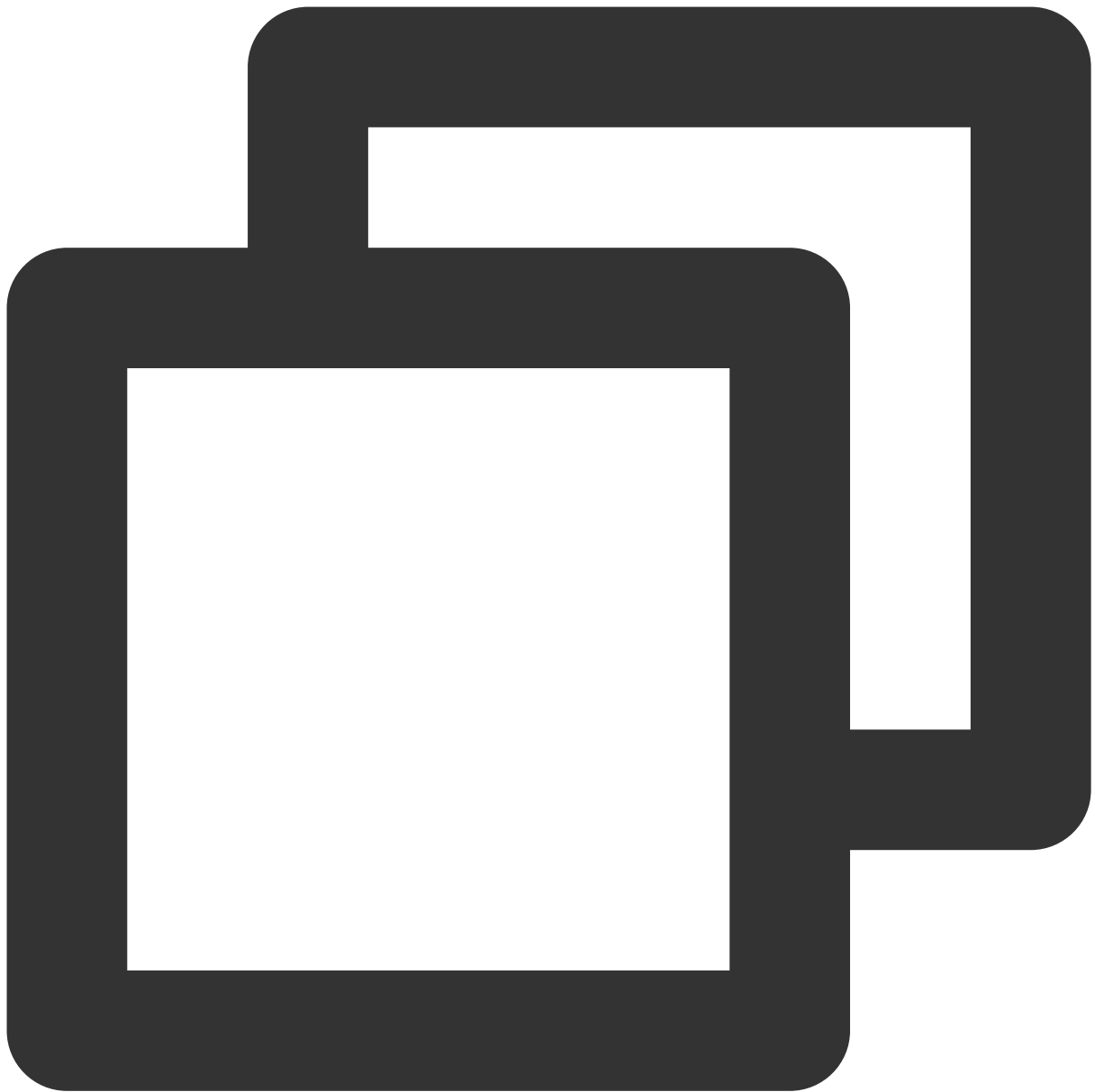
```
TUICallKit createInstance(Context context)
```

### setSelfInfo

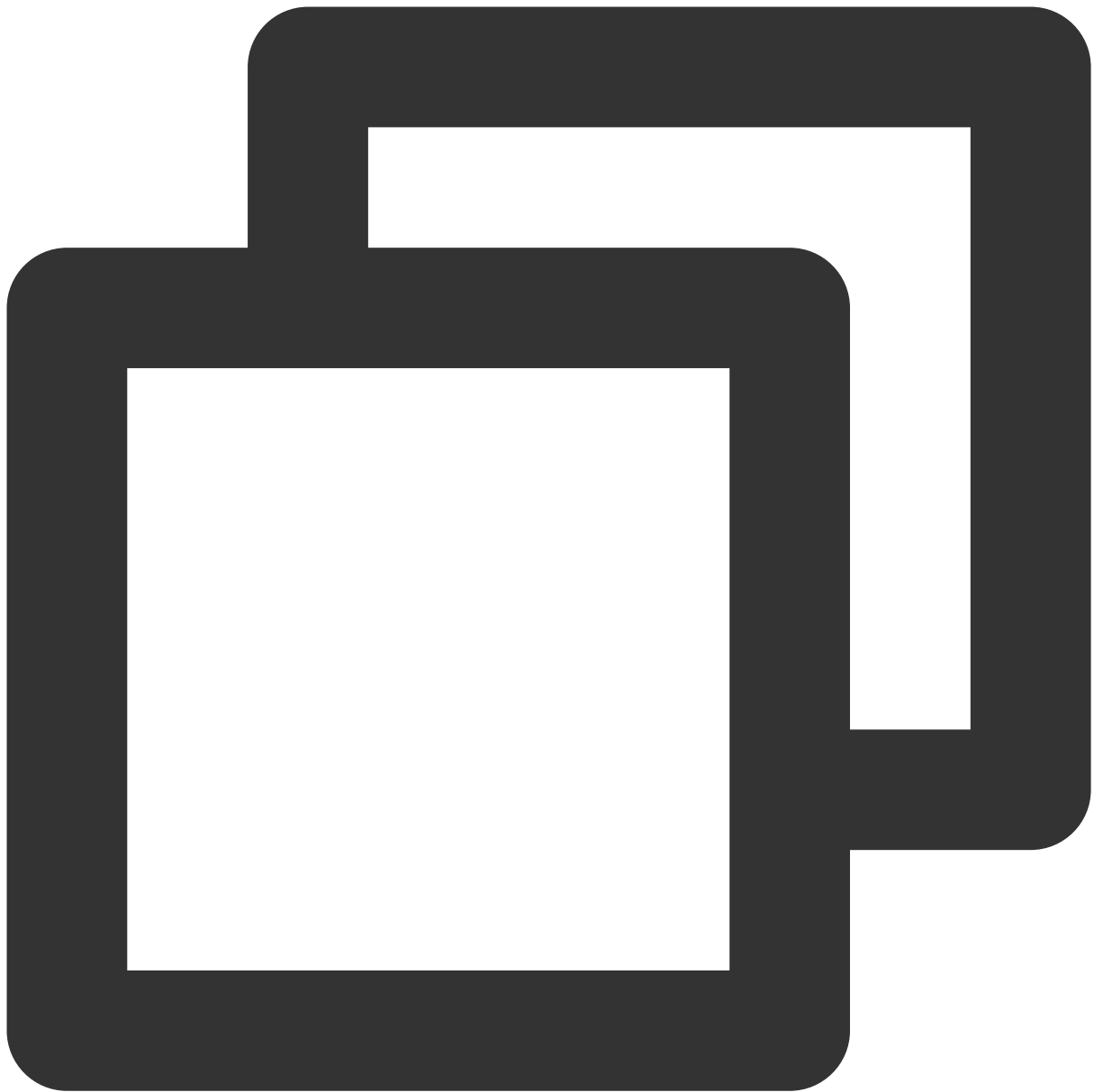
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.

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:

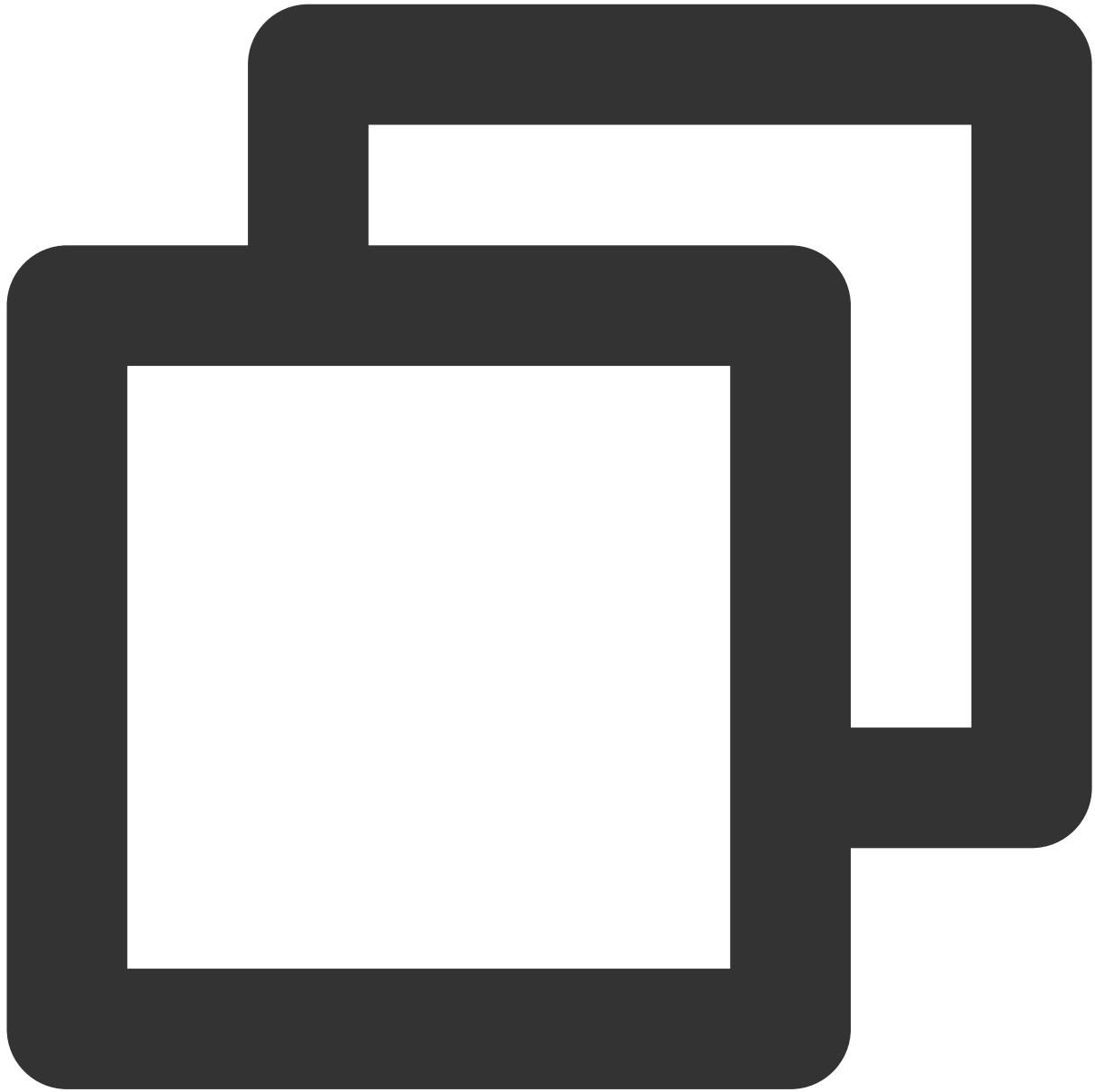
Parameter	Type	Description
nickname	String	The alias.
avatar	String	The profile photo.

**call**

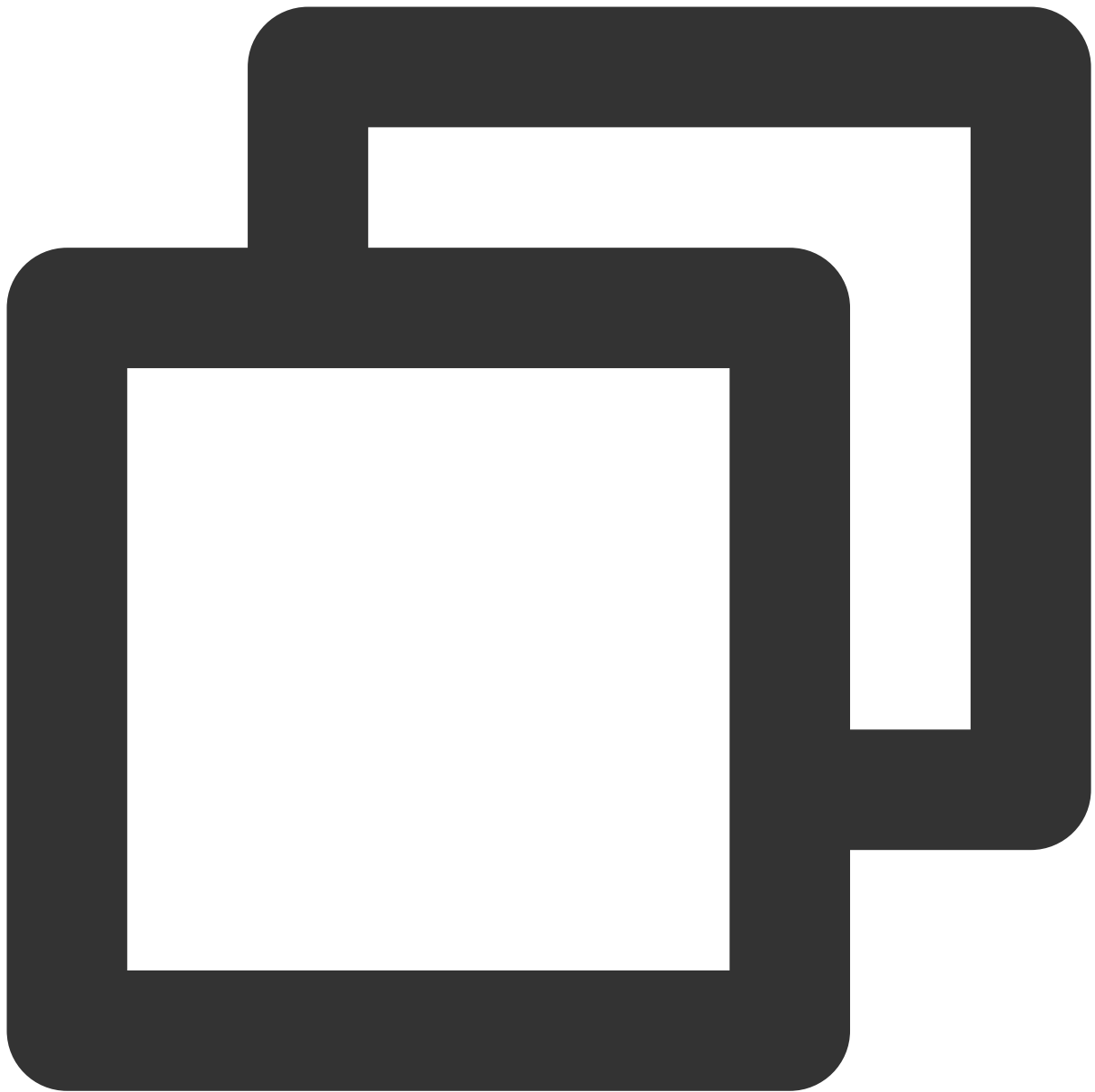
This API is used to make a (one-to-one) call.

Kotlin

Java



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



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

The parameters are described below:

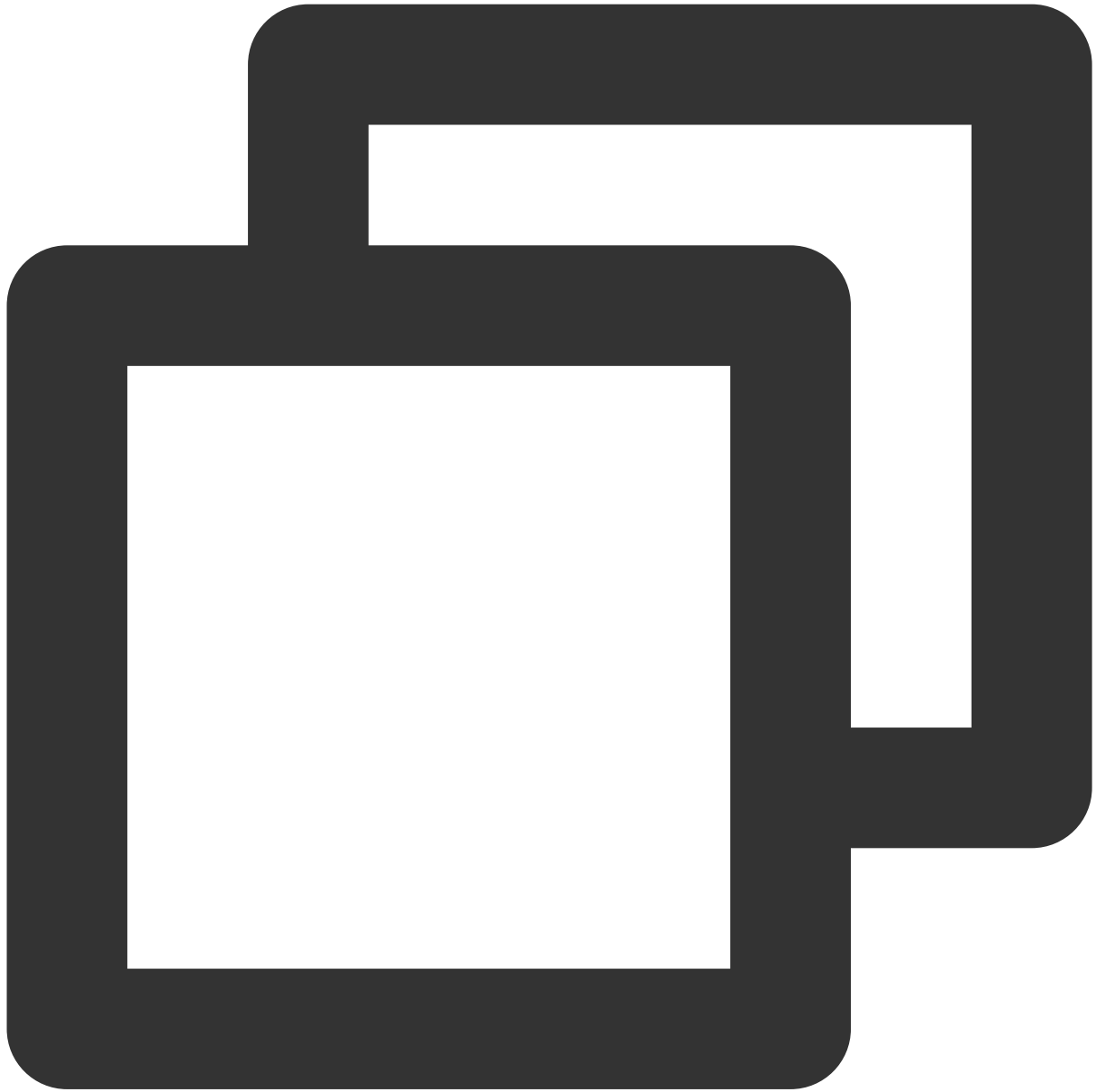
Parameter	Type	Description
userId	String	The target user ID.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.

## call

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

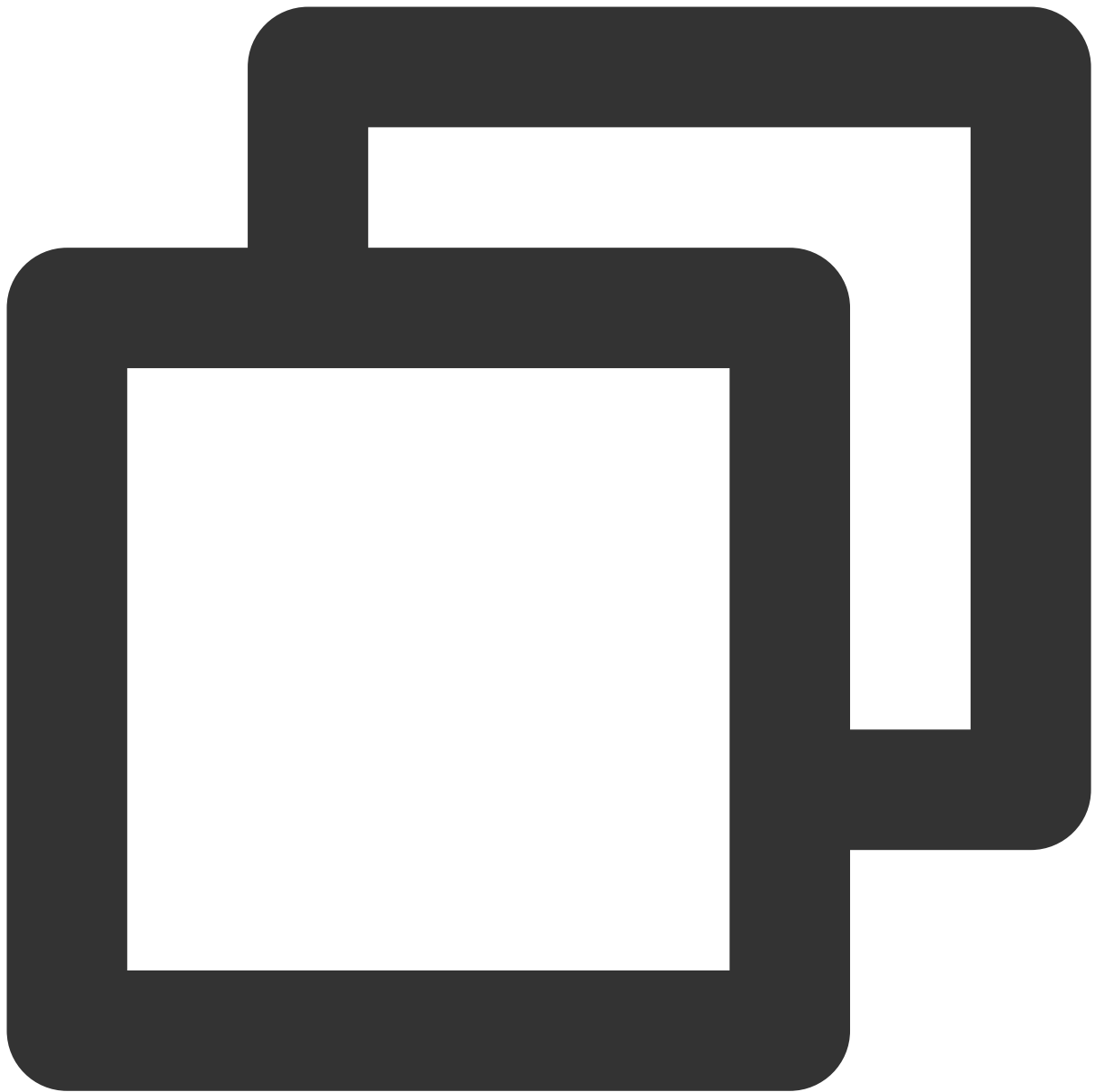
Kotlin

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	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.

params	<a href="#">TUICallDefine.CallParams</a>	Call extension parameters, such as roomID, call timeout, offline push info,etc
--------	--	--

## groupCall

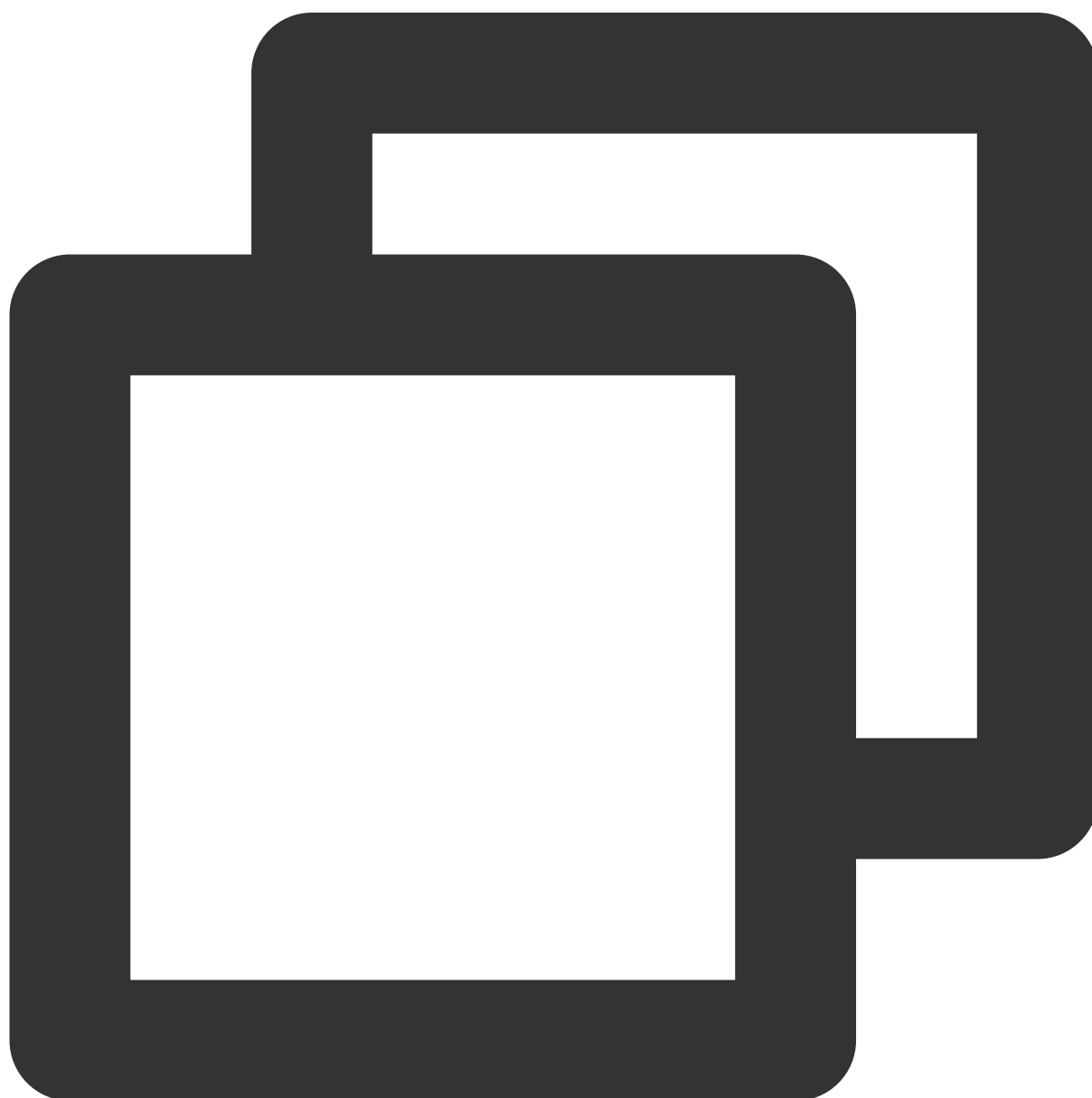
This API is used to make a group call.

### Notice:

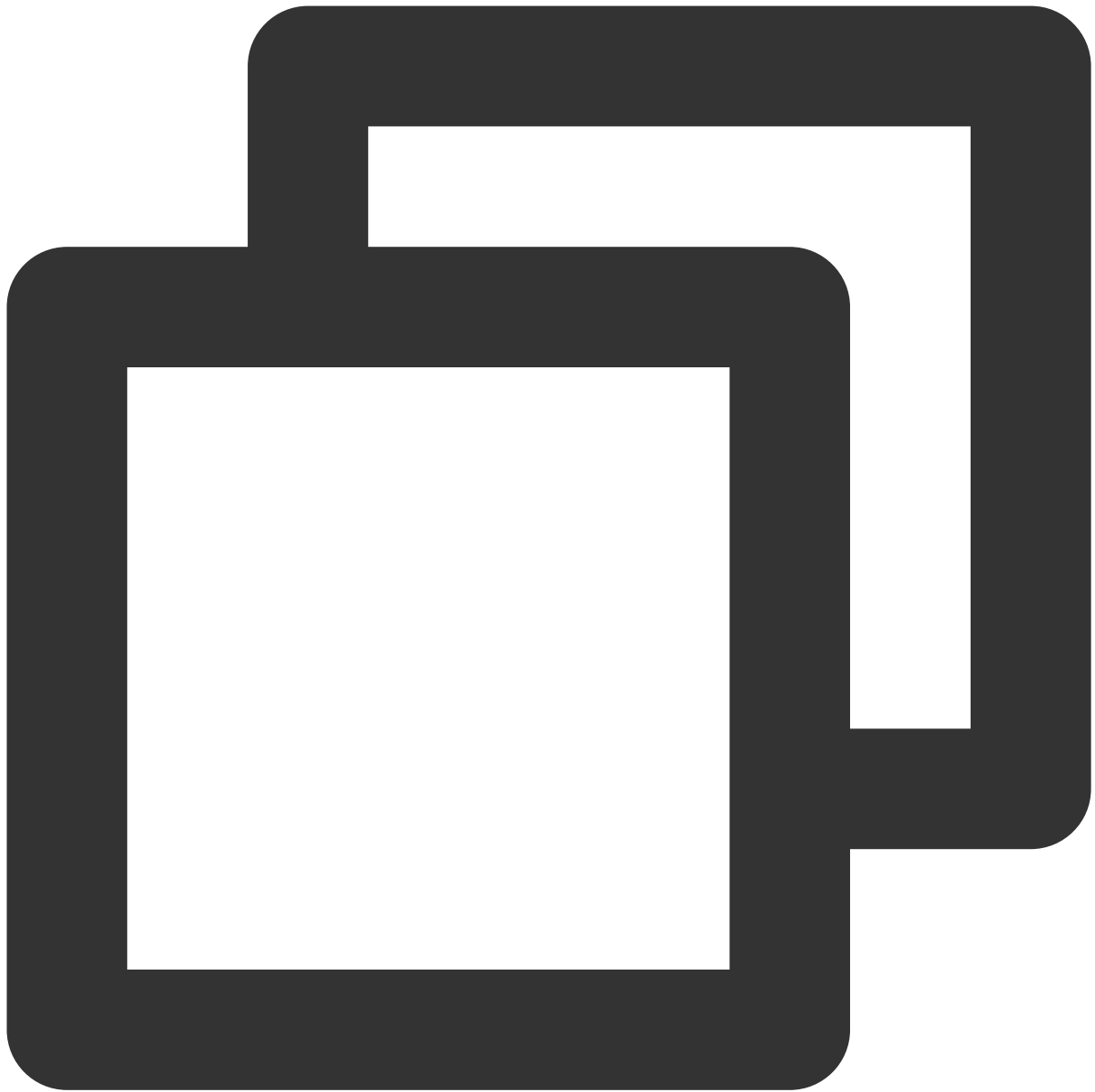
Before making a group call, you need to create an IM group first.

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

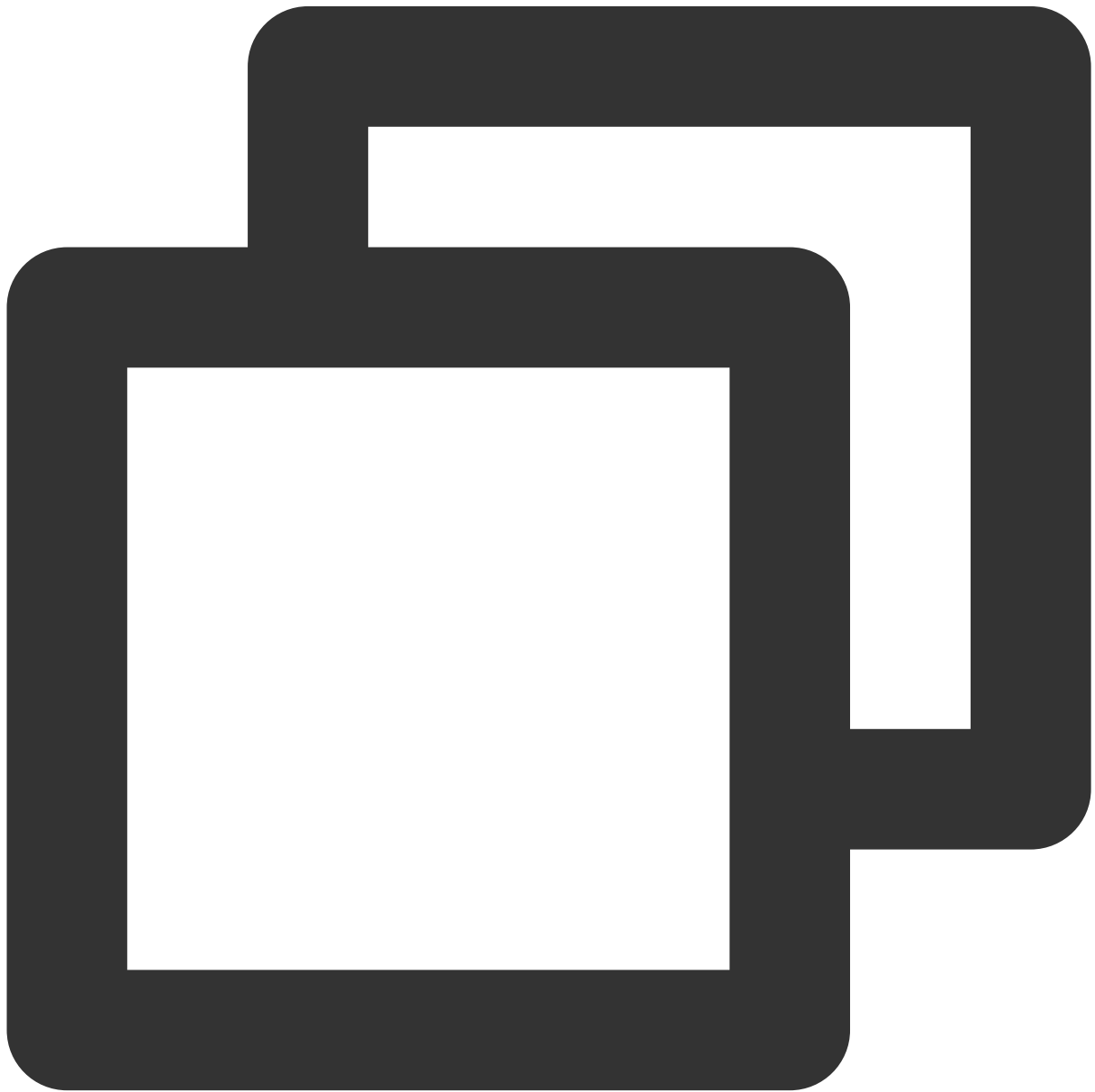
This API is used to make a group call, Support for custom room ID, call timeout, offline push content, etc.

### Notice:

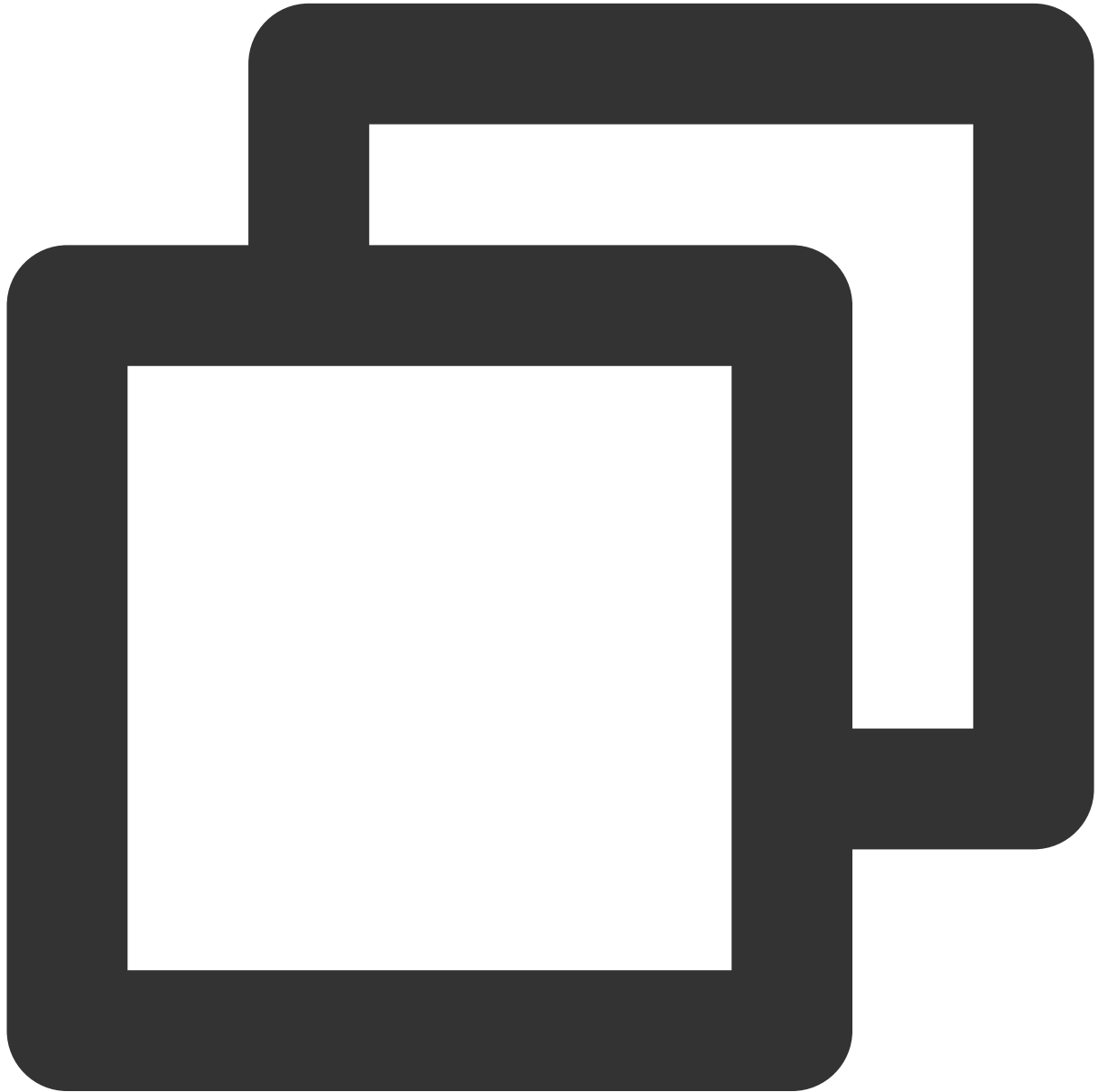
Before making a group call, you need to create an IM group first.

Kotlin

Java



```
fun groupCall(  
    groupId: String, userIdList: List<String?>?,  
    callMediaType: TUICallDefine.MediaType, params: CallParams?,  
    callback: TUICCommonDefine.Callback?  
)
```



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

The parameters are described below:

Parameter	Type	Description
groupId	String	The group ID.
userIdList	List	The target user IDs.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallDefine.CallParams</a>	Call extension parameters, such as roomId, call timeout, offline push info,etc

## joinInGroupCall

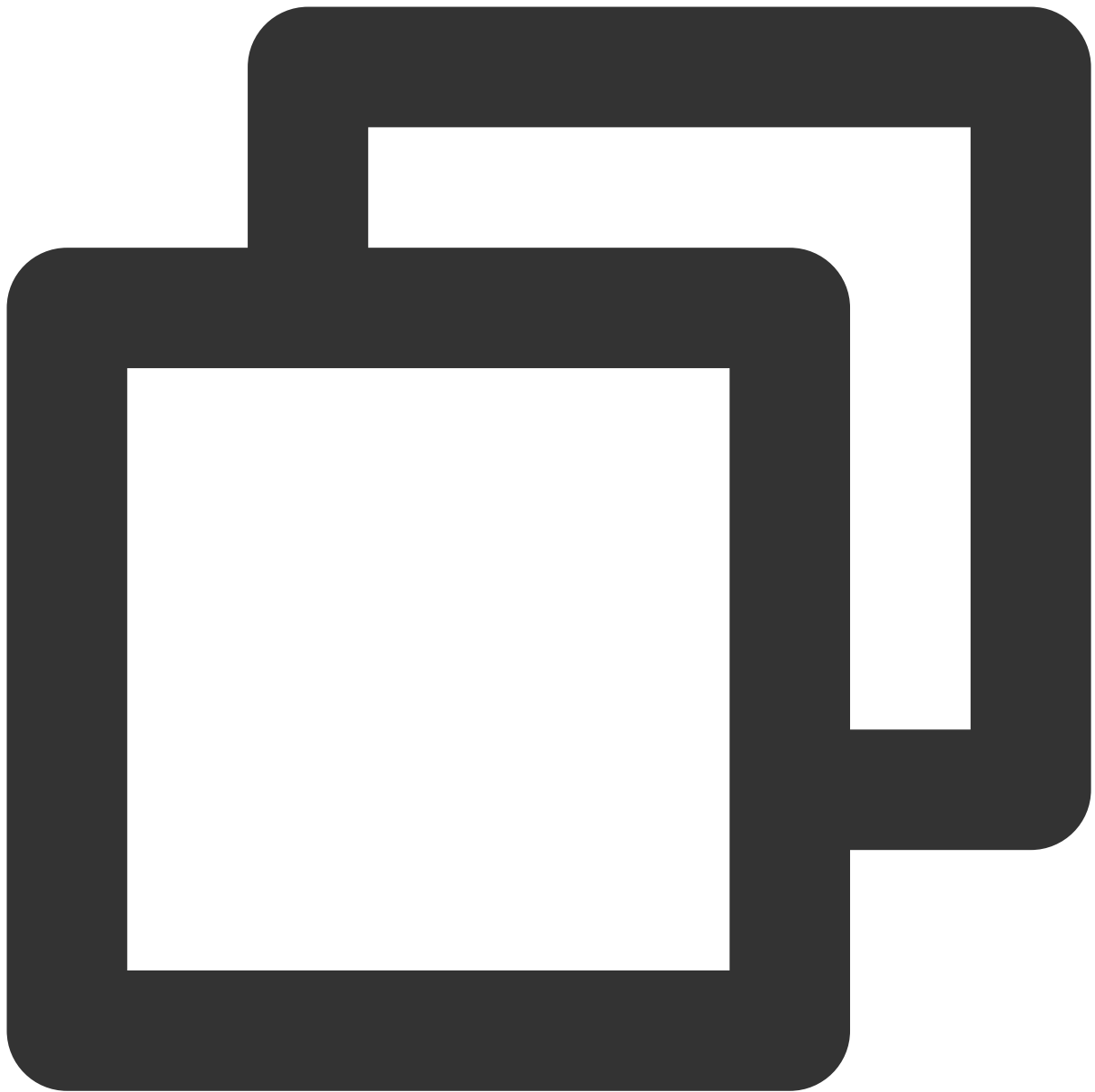
This API is used to join a group call.

### Notice:

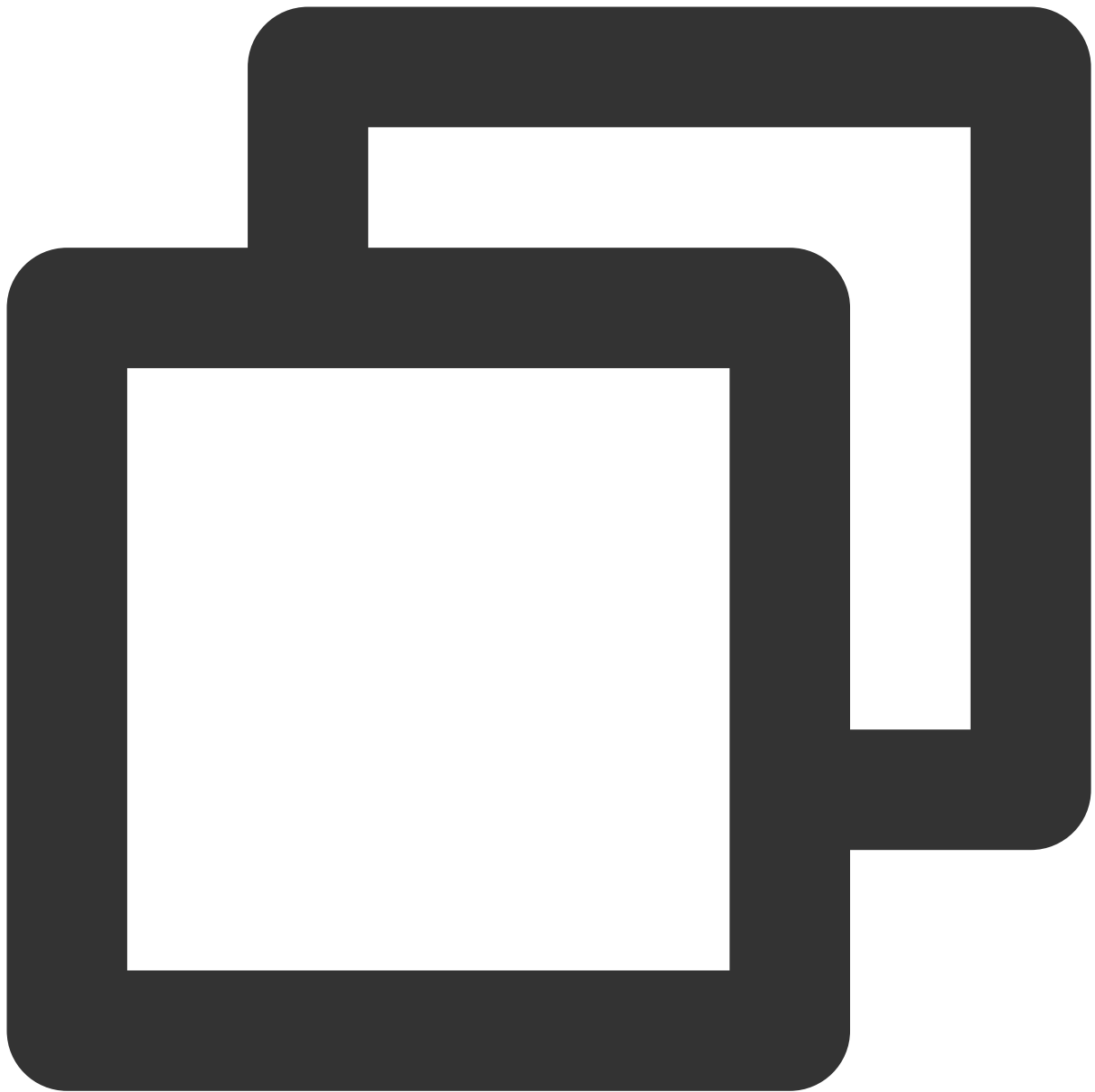
Before joining a group call, you need to create or join an IM group in advance, and there are already users in the group who are in the call.

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	<a href="#">TUICommonDefine.RoomId</a>	The room ID.
groupId	String	The group ID.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.



## setCallingBell

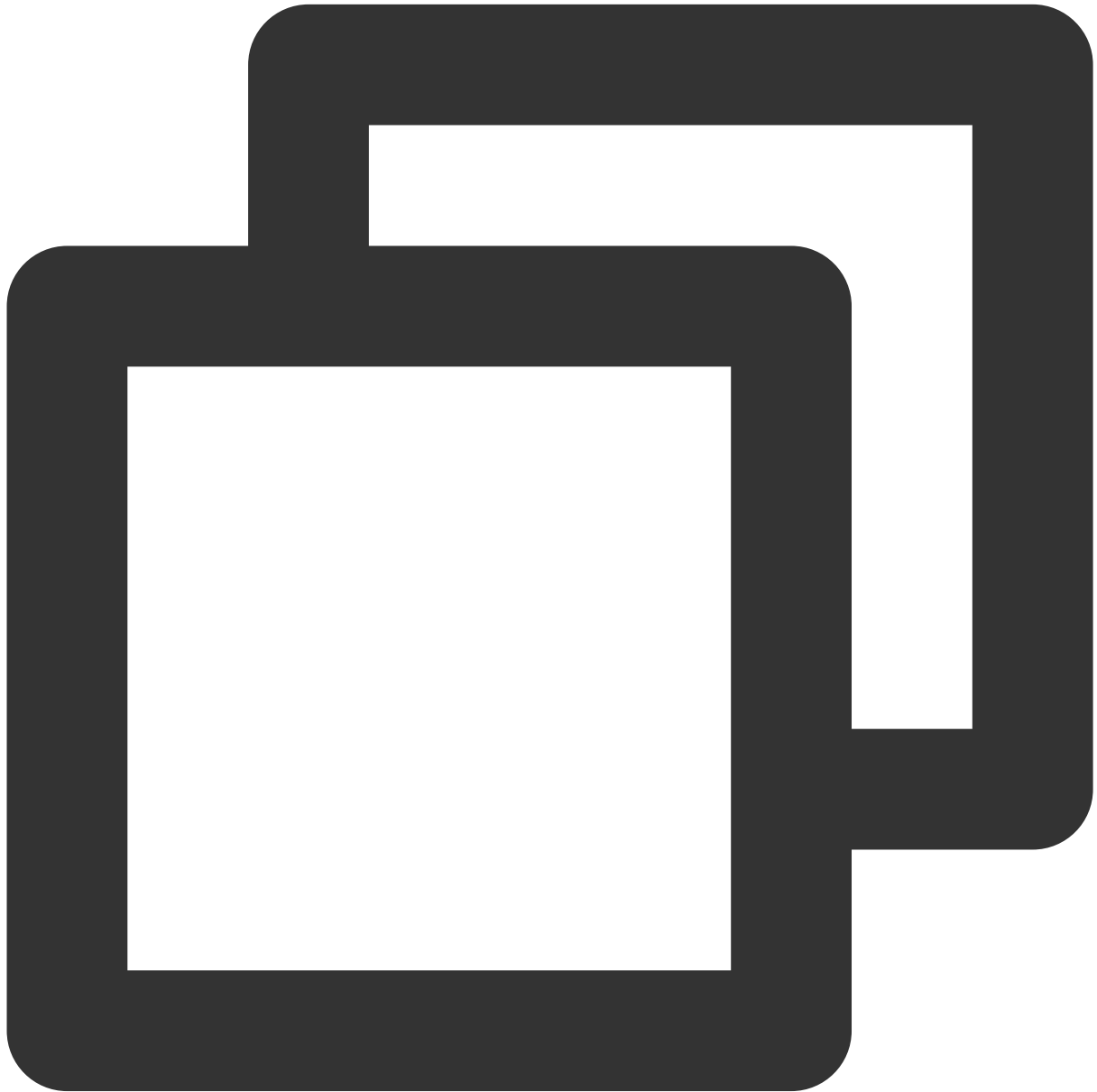
This API is used to set the ringtone. `filePath` must be an accessible local file URL.

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

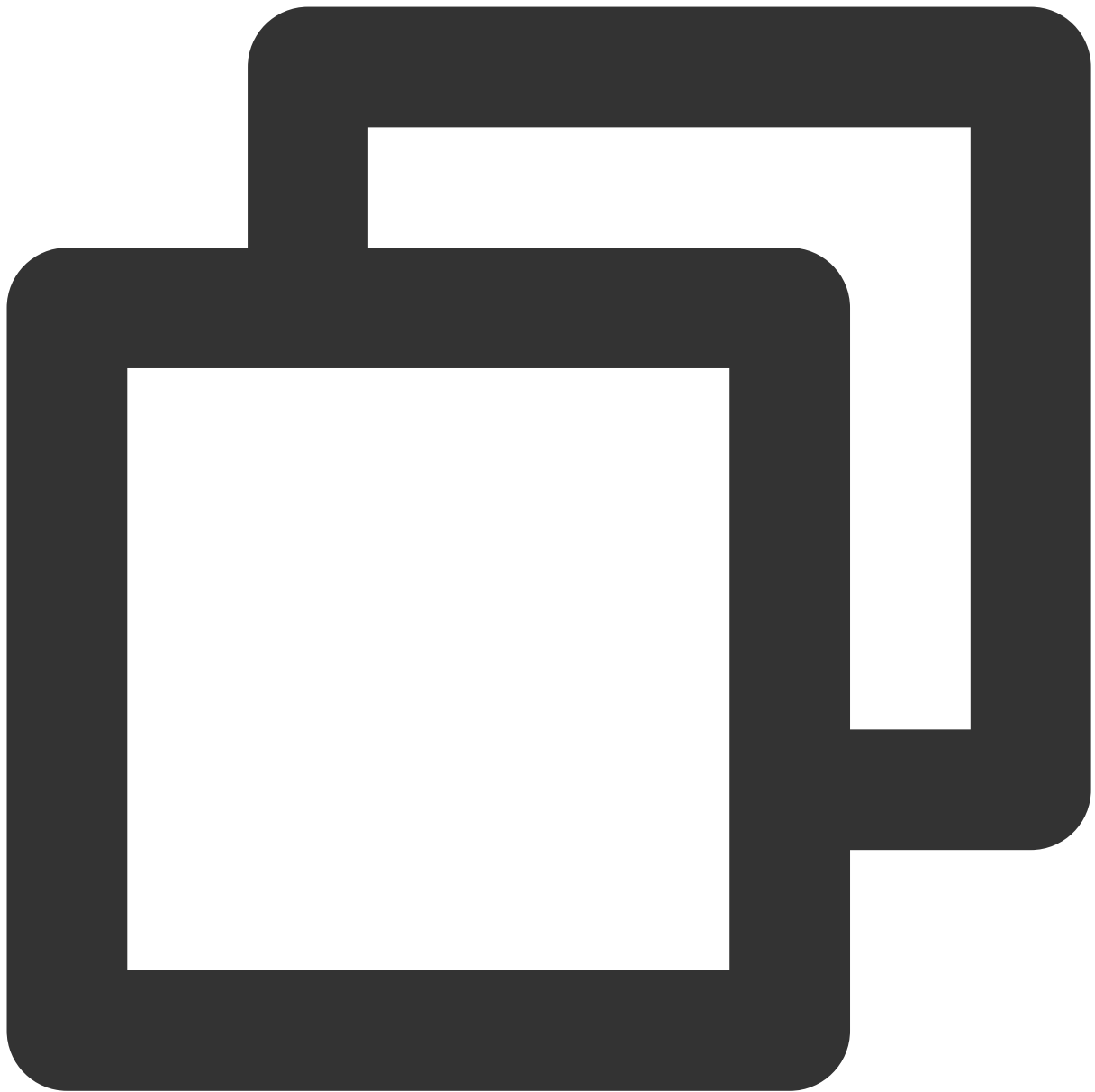
To reset the ringtone, pass in an empty string for `filePath`.

Kotlin

Java



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



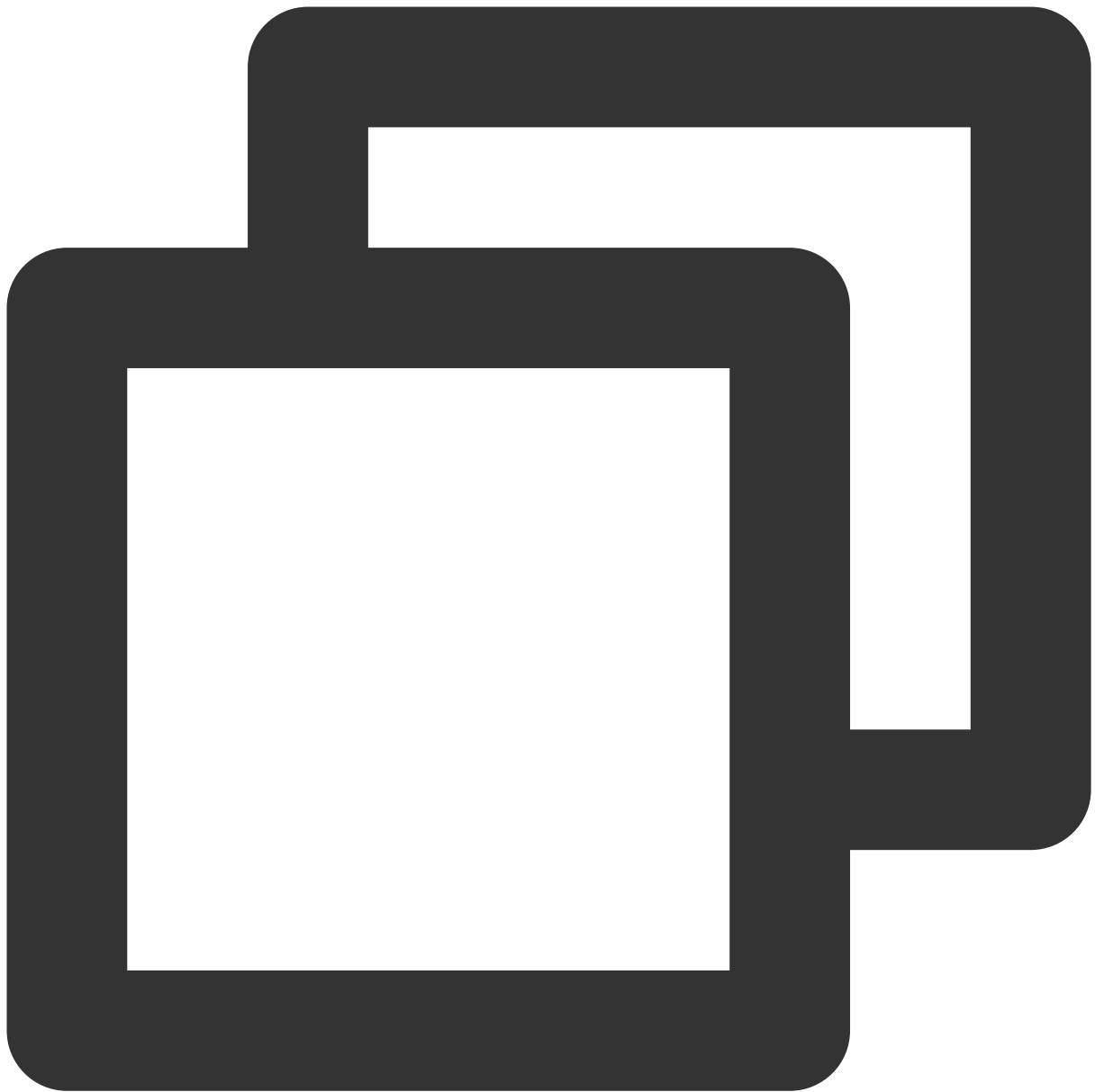
```
void setCallingBell(String filePath);
```

### **enableMuteMode**

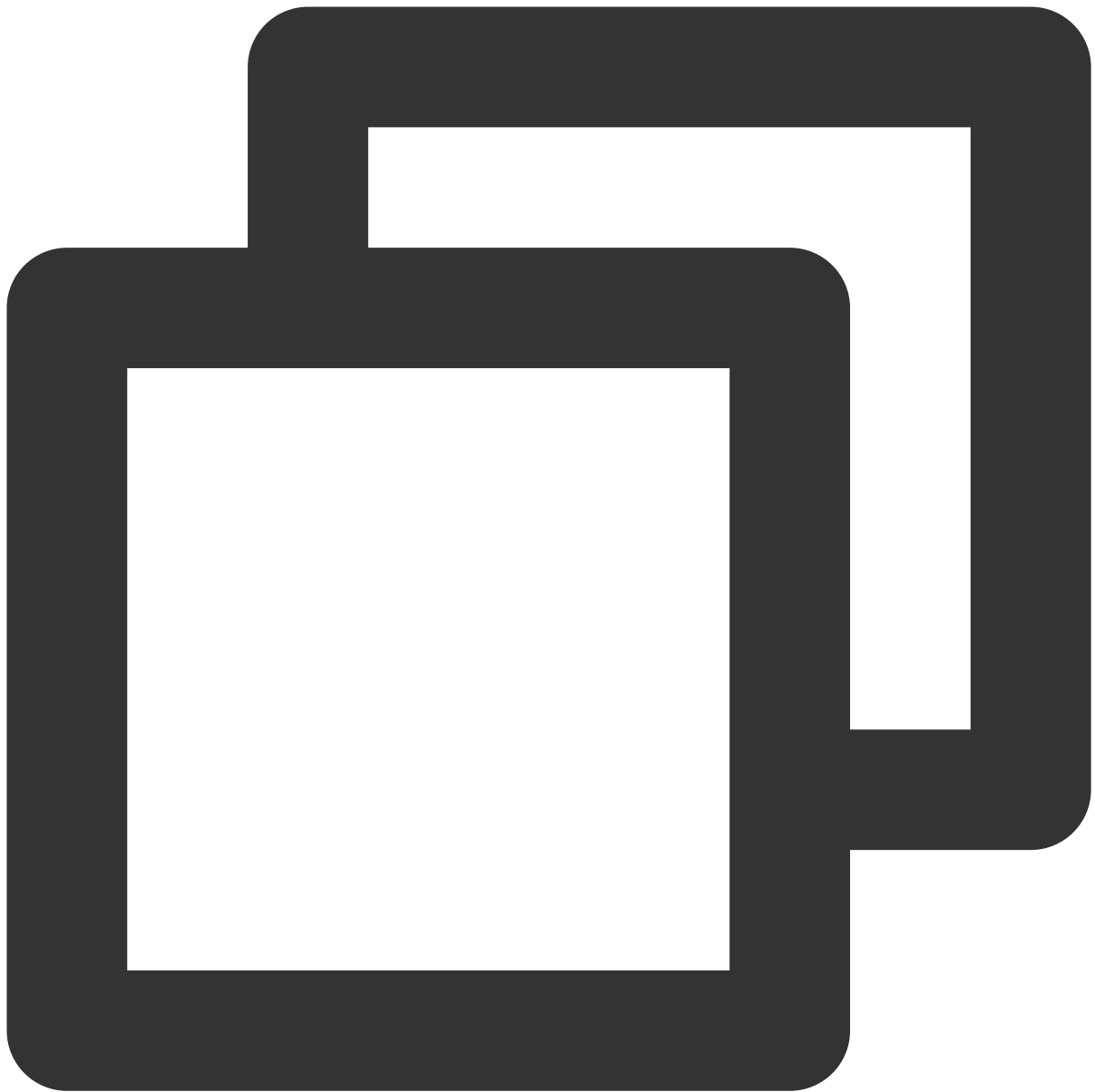
This API is used to set whether to turn on the mute mode.

Kotlin

Java



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



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

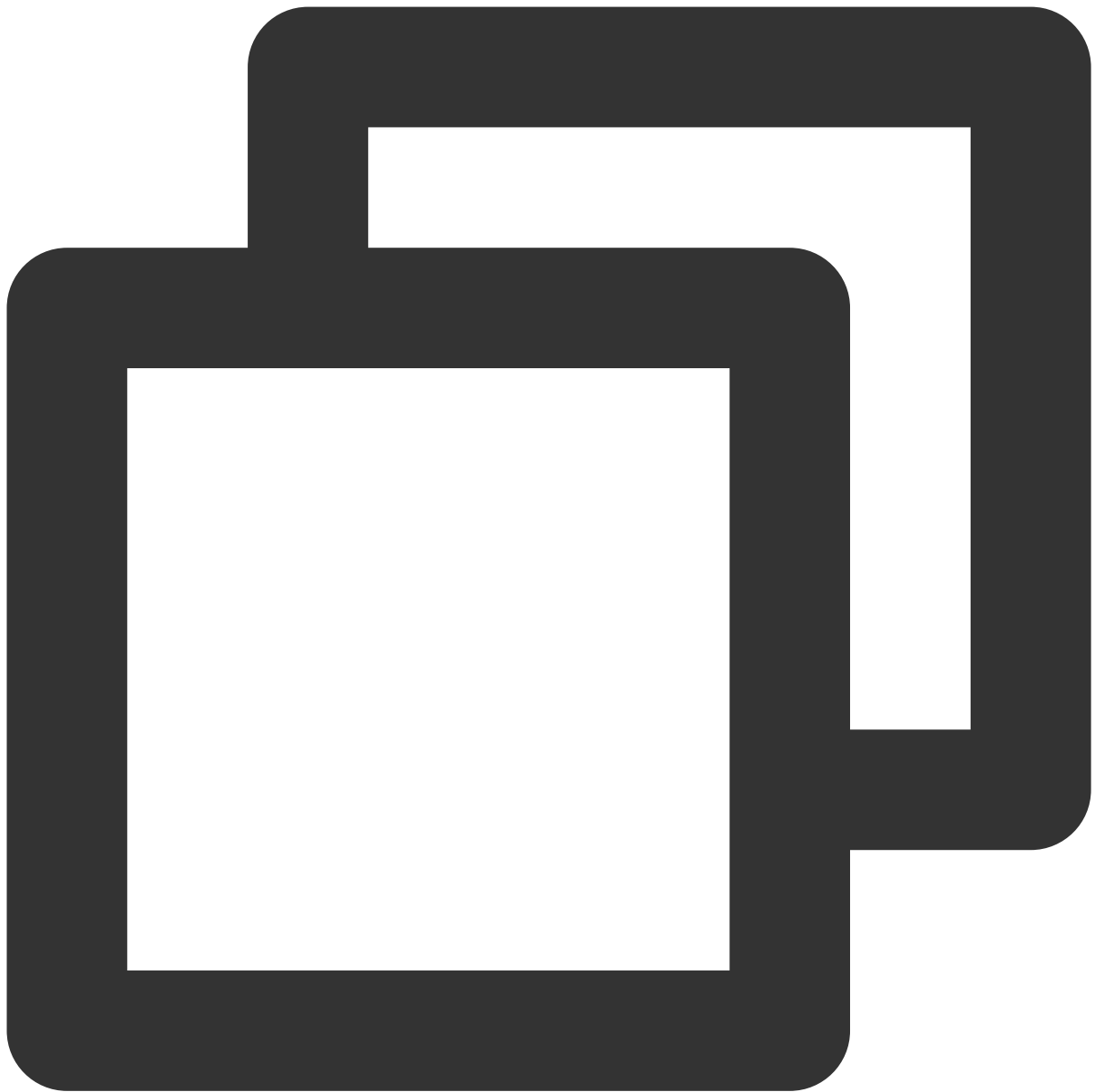
## enableFloatWindow

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

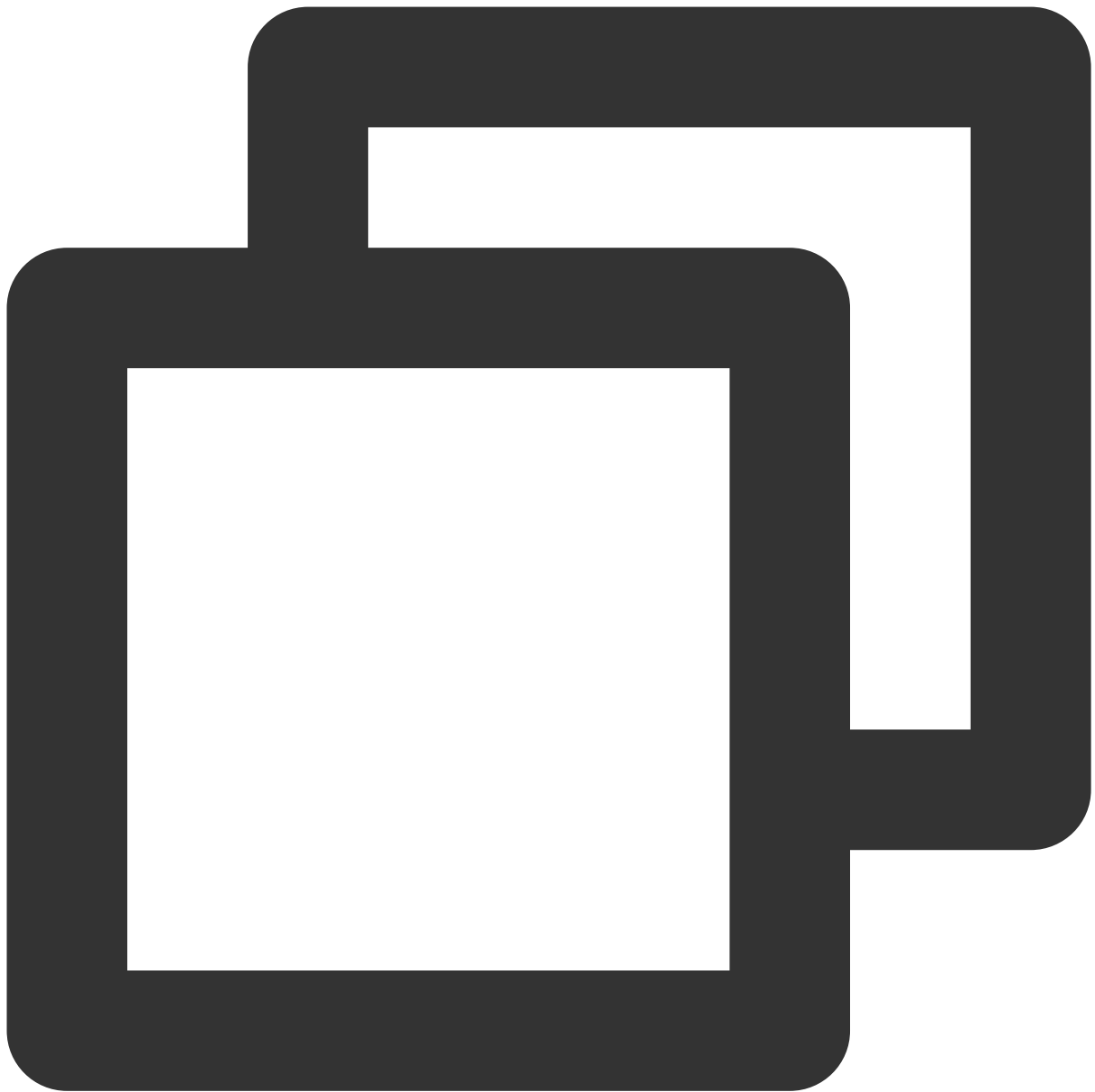
The default value is `false`, and the floating window button in the top left corner of the call view is hidden. If it is set to `true`, the button will become visible.

Kotlin

Java



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

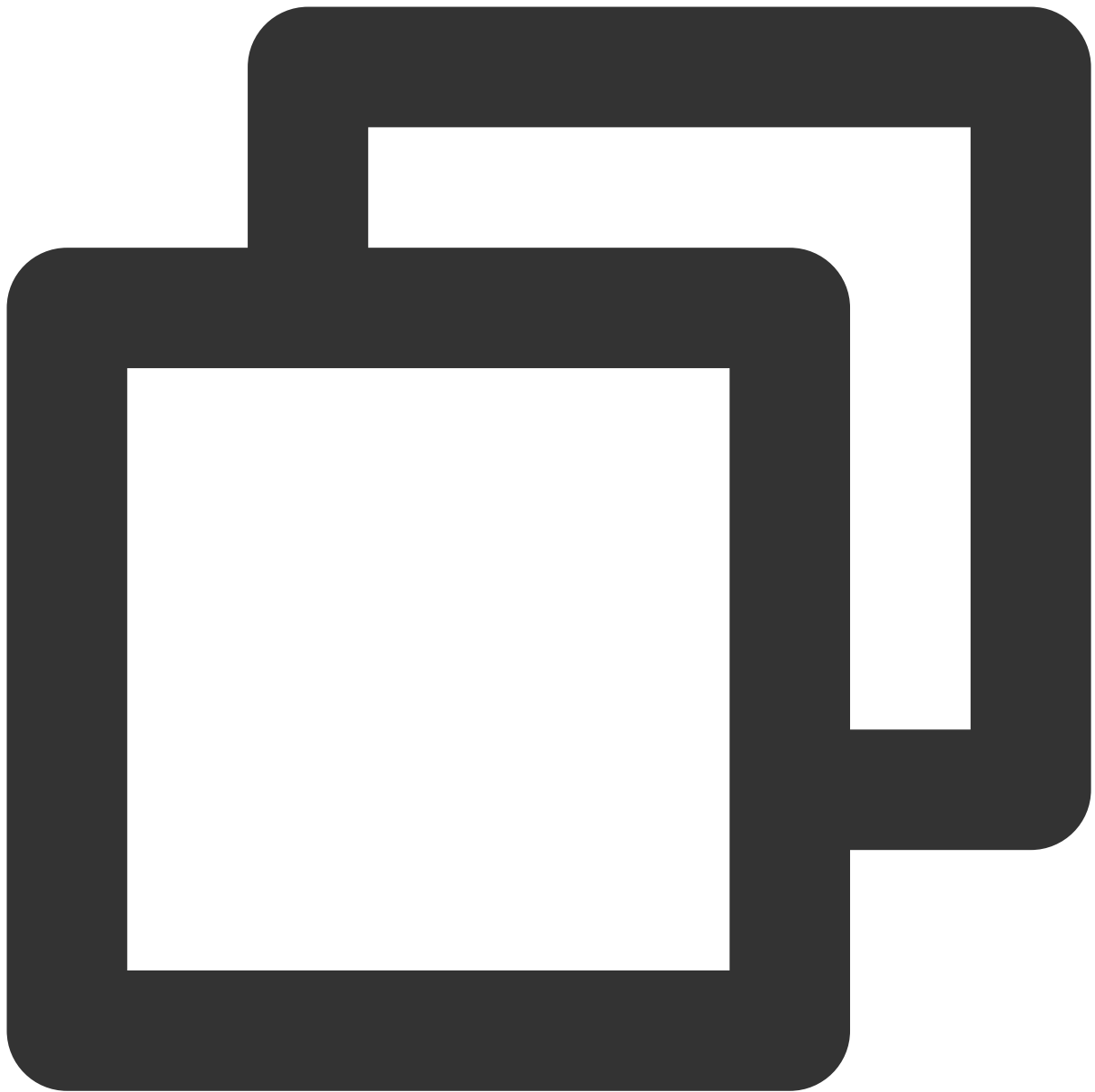


```
void enableFloatWindow(boolean enable);
```

### **enableIncomingBanner**

The API is used to set whether to enable the incoming banner.

The default value is `false`. The callee will pop up a full-screen call view by default when receiving the invitation. If it is set to `true`, the callee will display a banner first.



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

# TUICallEngine

Last updated : 2024-07-30 11:20:54

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

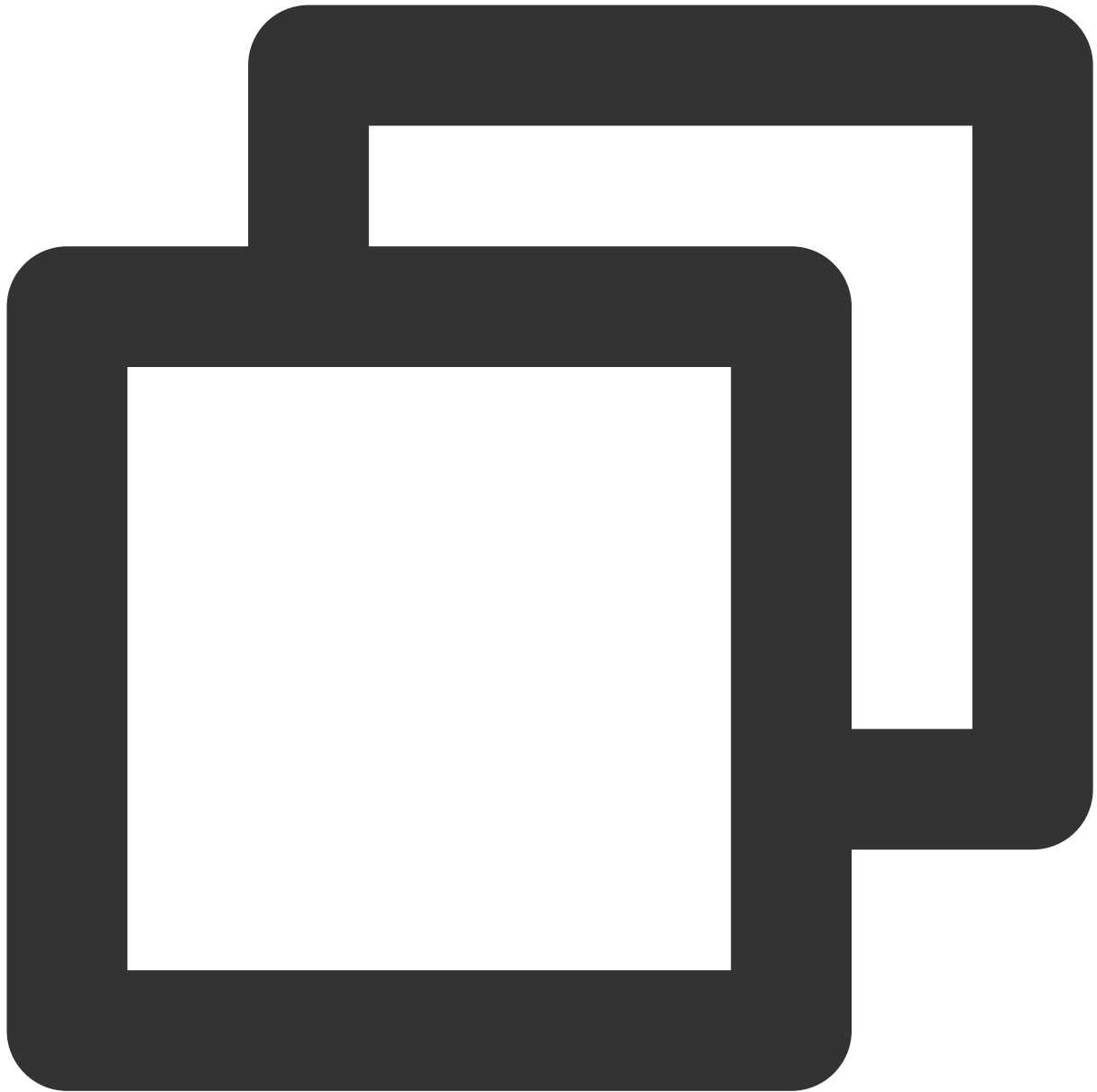


<a href="#">openCamera</a>	Turns the camera on.
<a href="#">closeCamera</a>	Turns the camera off.
<a href="#">switchCamera</a>	Switches between the front and rear cameras.
<a href="#">openMicrophone</a>	Turns the mic on.
<a href="#">closeMicrophone</a>	Turns the mic off.
<a href="#">selectAudioPlaybackDevice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

## Details

### createInstance

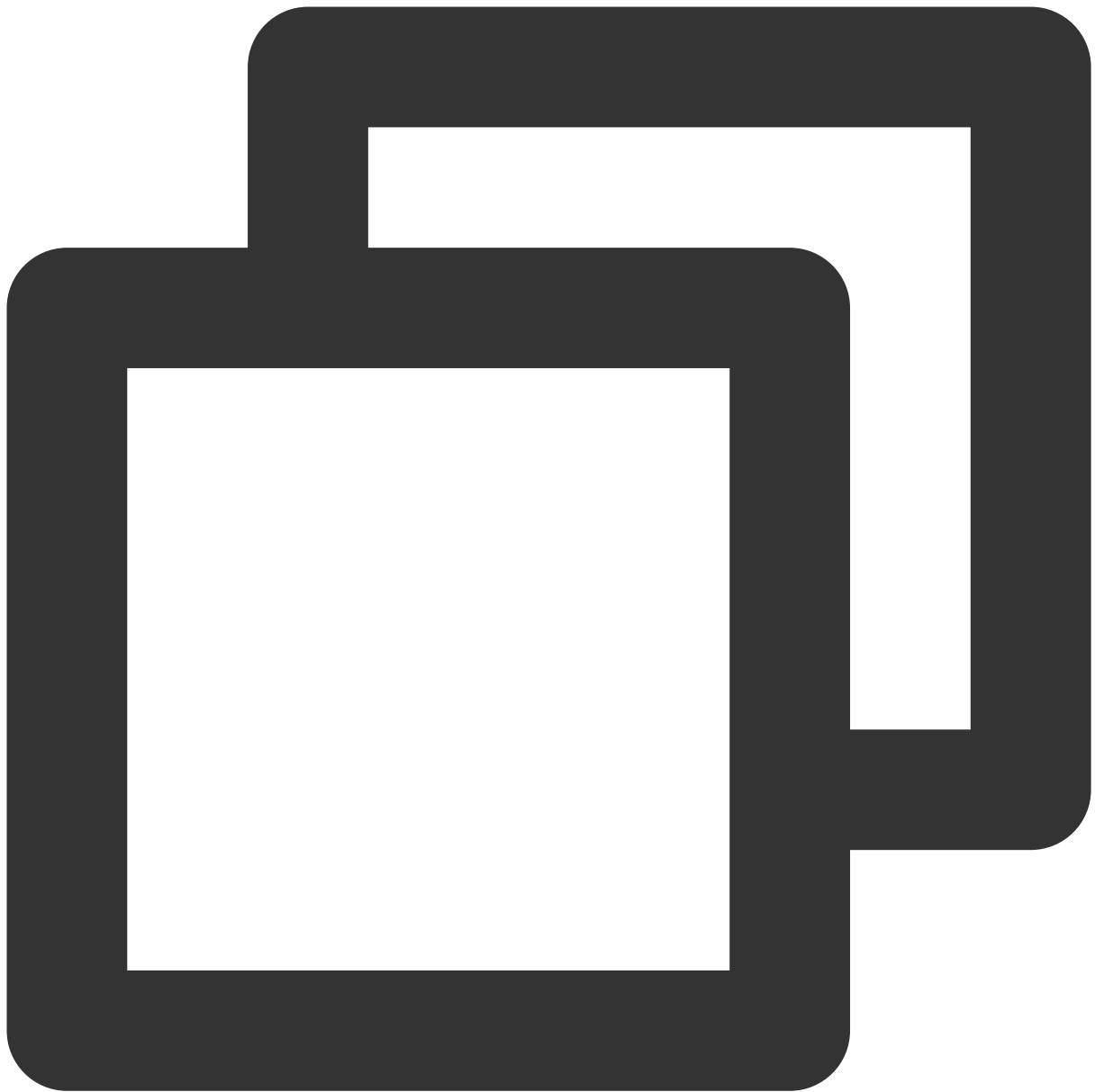
This API is used to create a `TUICallEngine` singleton.



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

### **destroyInstance**

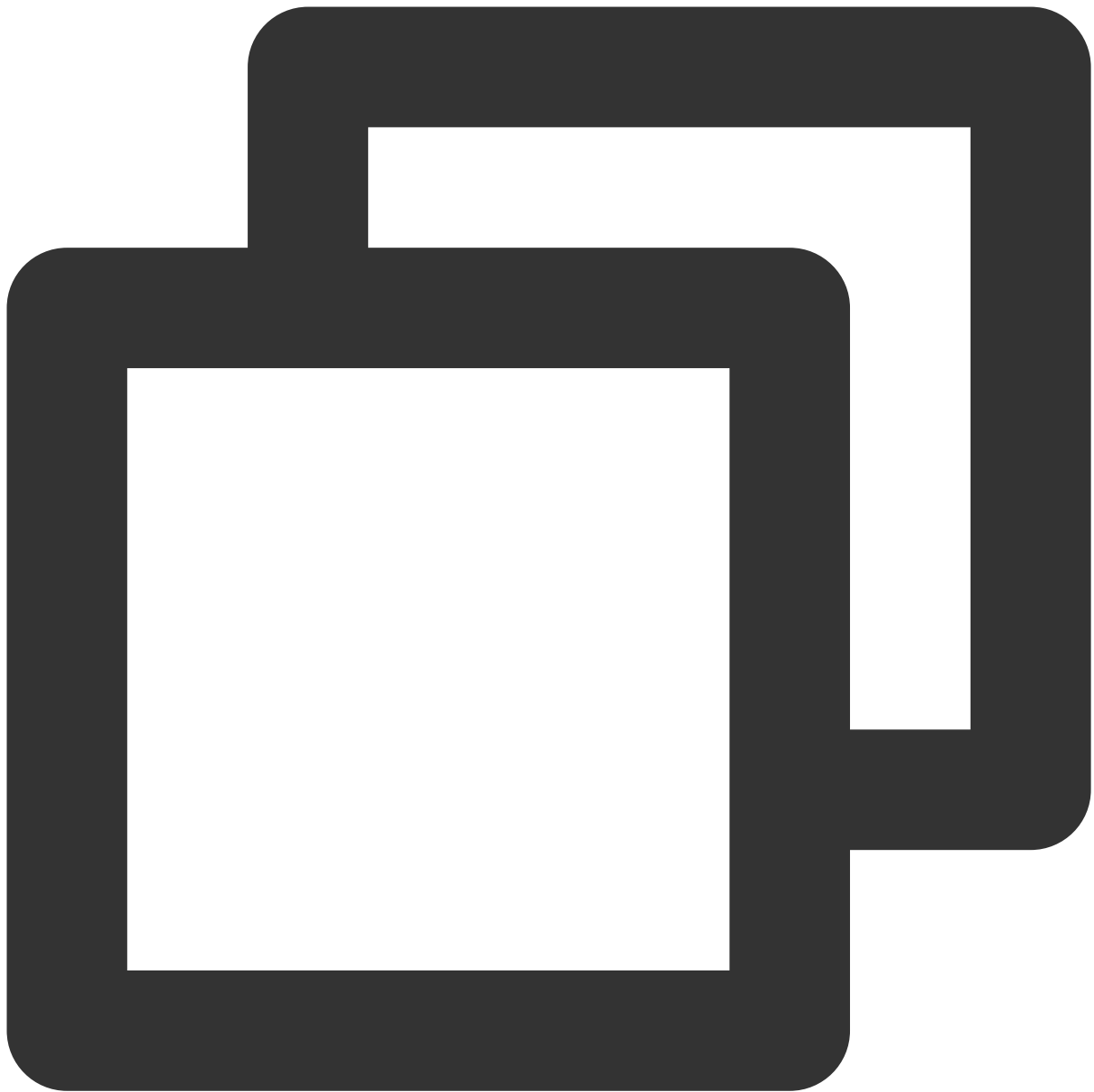
This API is used to terminate a `TUICallEngine` singleton.



```
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
```

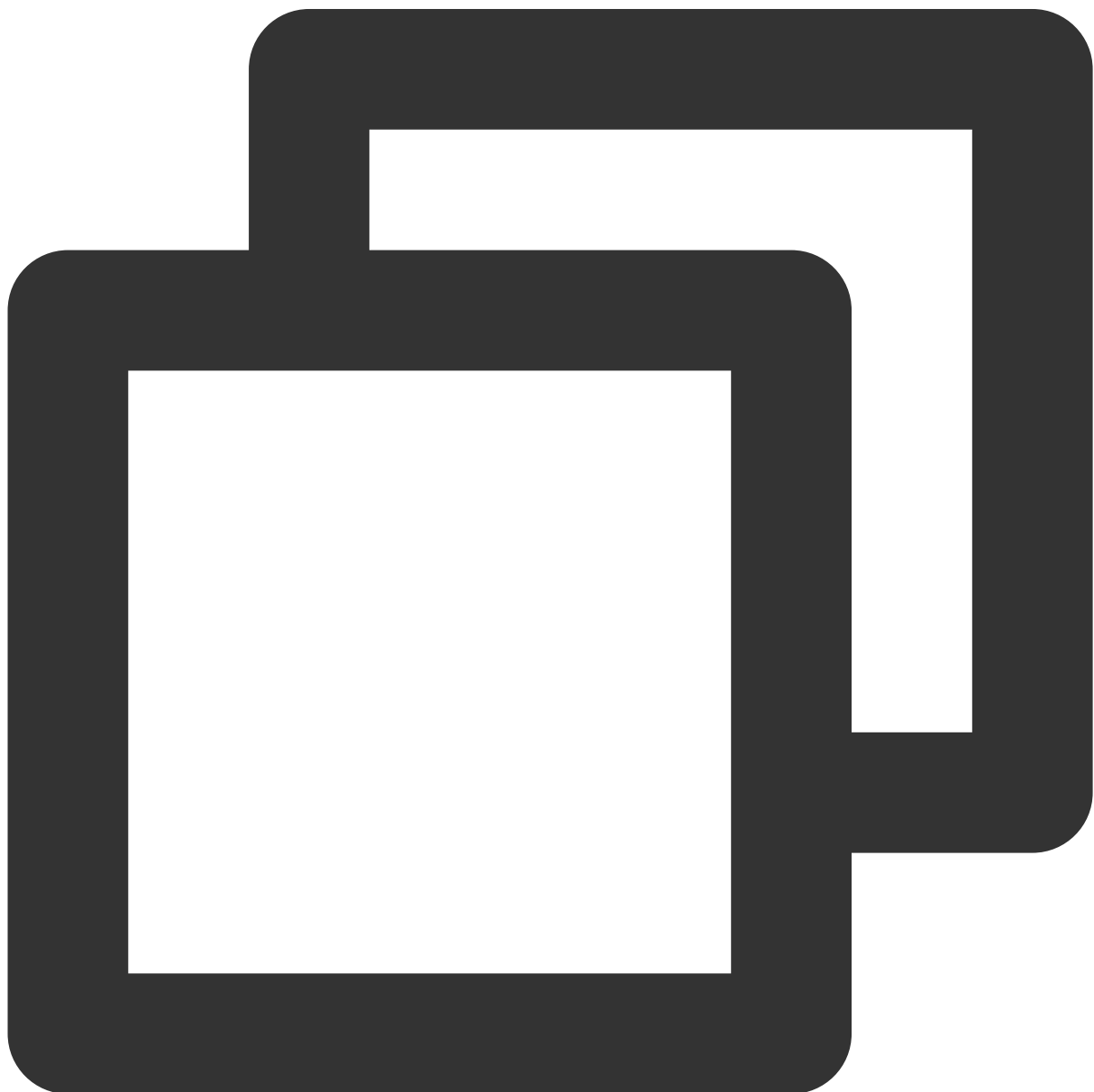
The parameters are described below:

Parameter	Type	Description
sdkAppId	int	You can view <code>SDKAppID</code> in <a href="#">Application Management</a> > <b>Application Info</b> 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 <a href="#">UserSig</a> .
callback	TUICommonDefine.Callback	The initialization callback. <code>onSuccess</code> indicates initialization is successful.

## addObserver

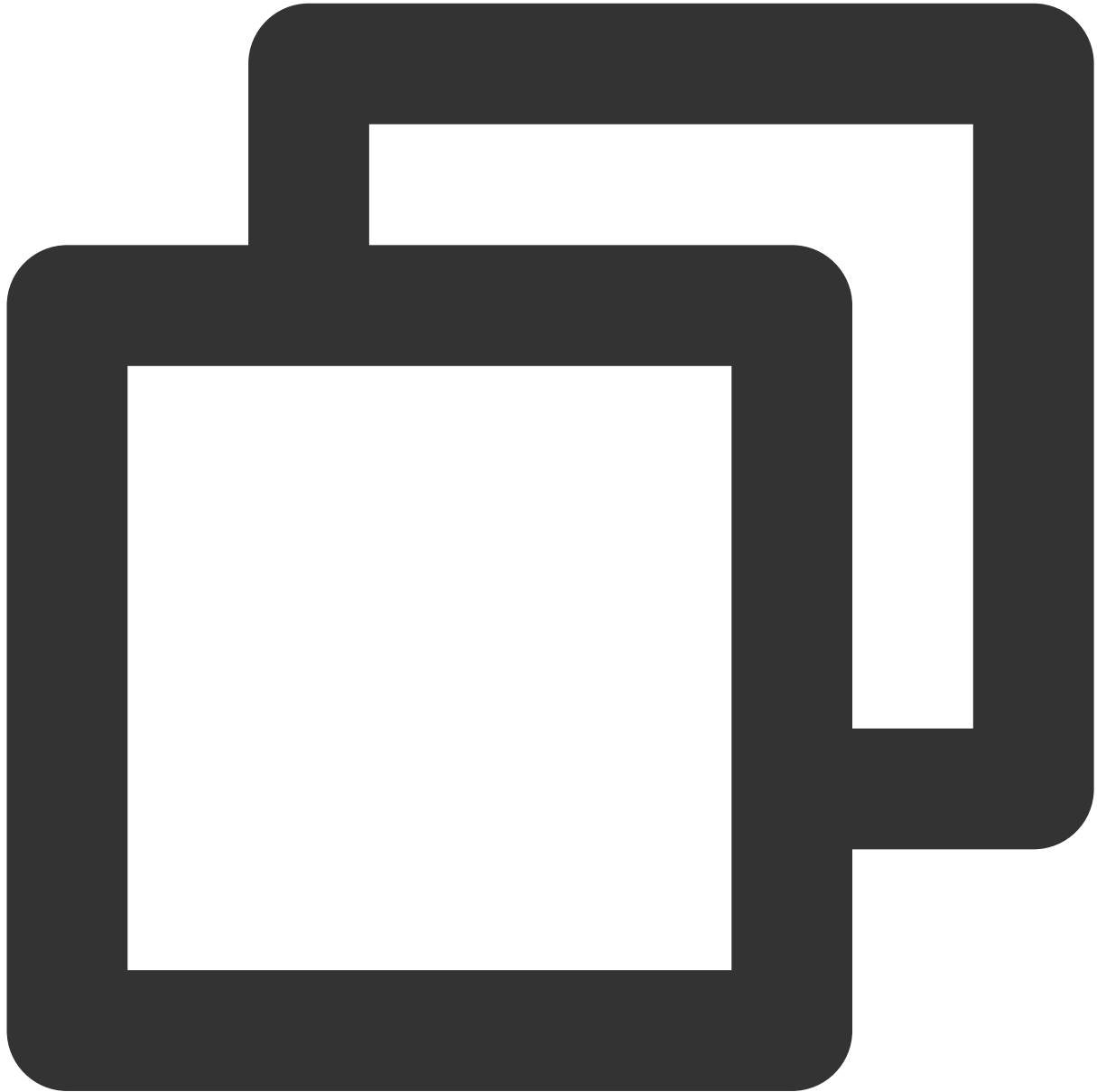
This API is used to register an event listener to listen for `TUICallObserver` events.



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

## **removeObserver**

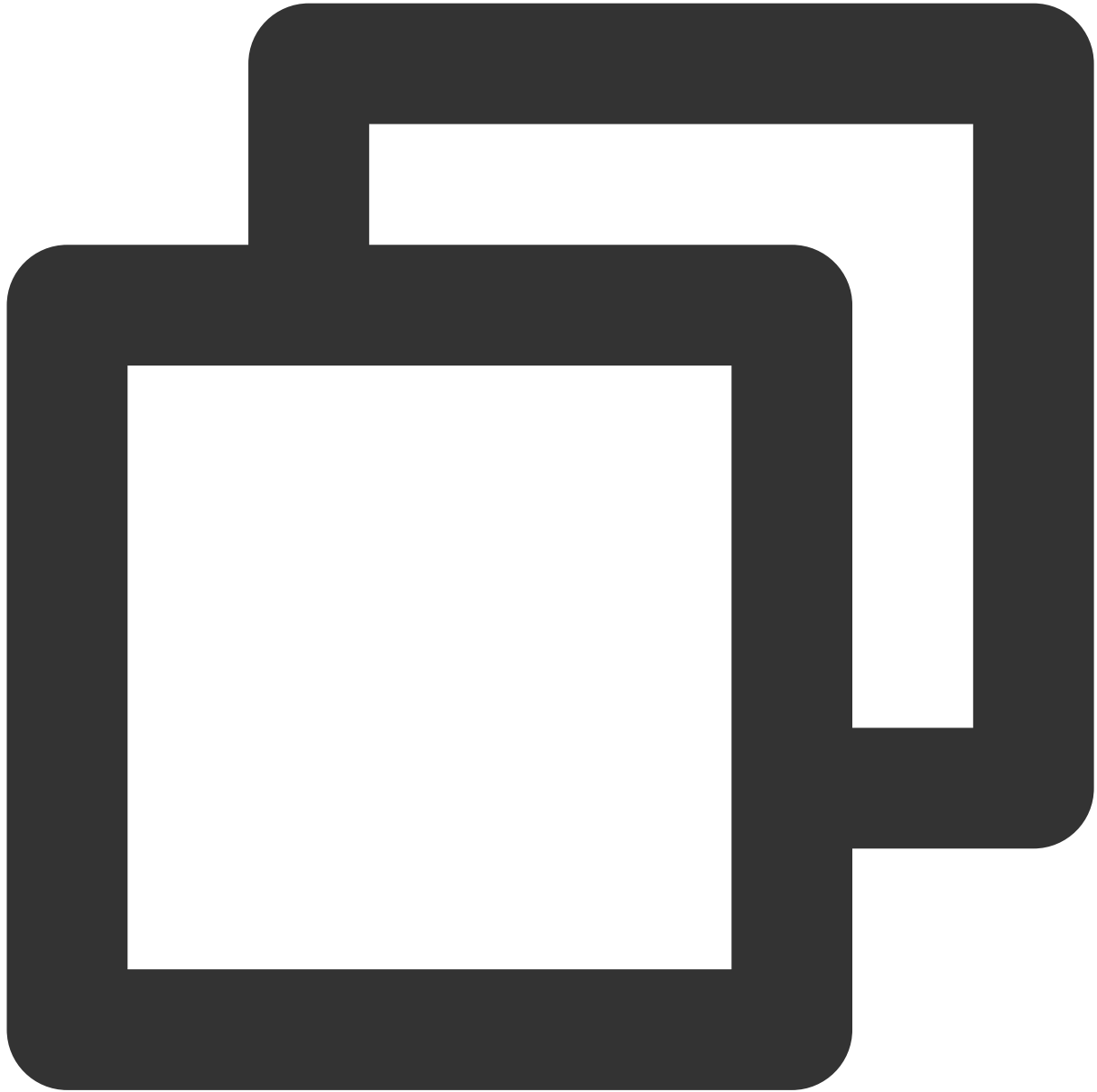
This API is used to unregister an event listener.



```
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	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.

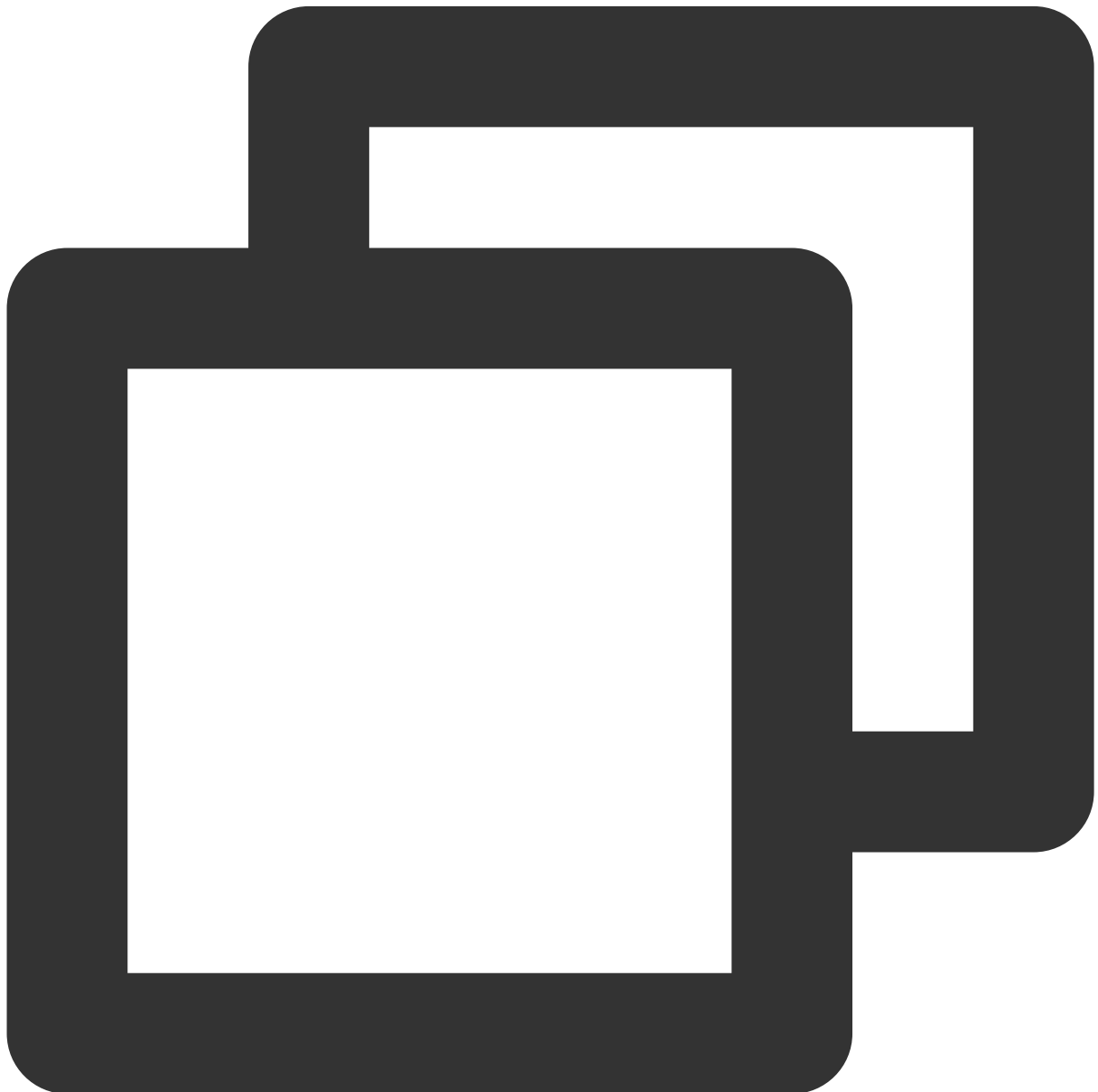
params	<a href="#">TUICallDefine.CallParams</a>	An additional parameter, such as roomId, call timeout, offline push info,etc
--------	--	--

## 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
```



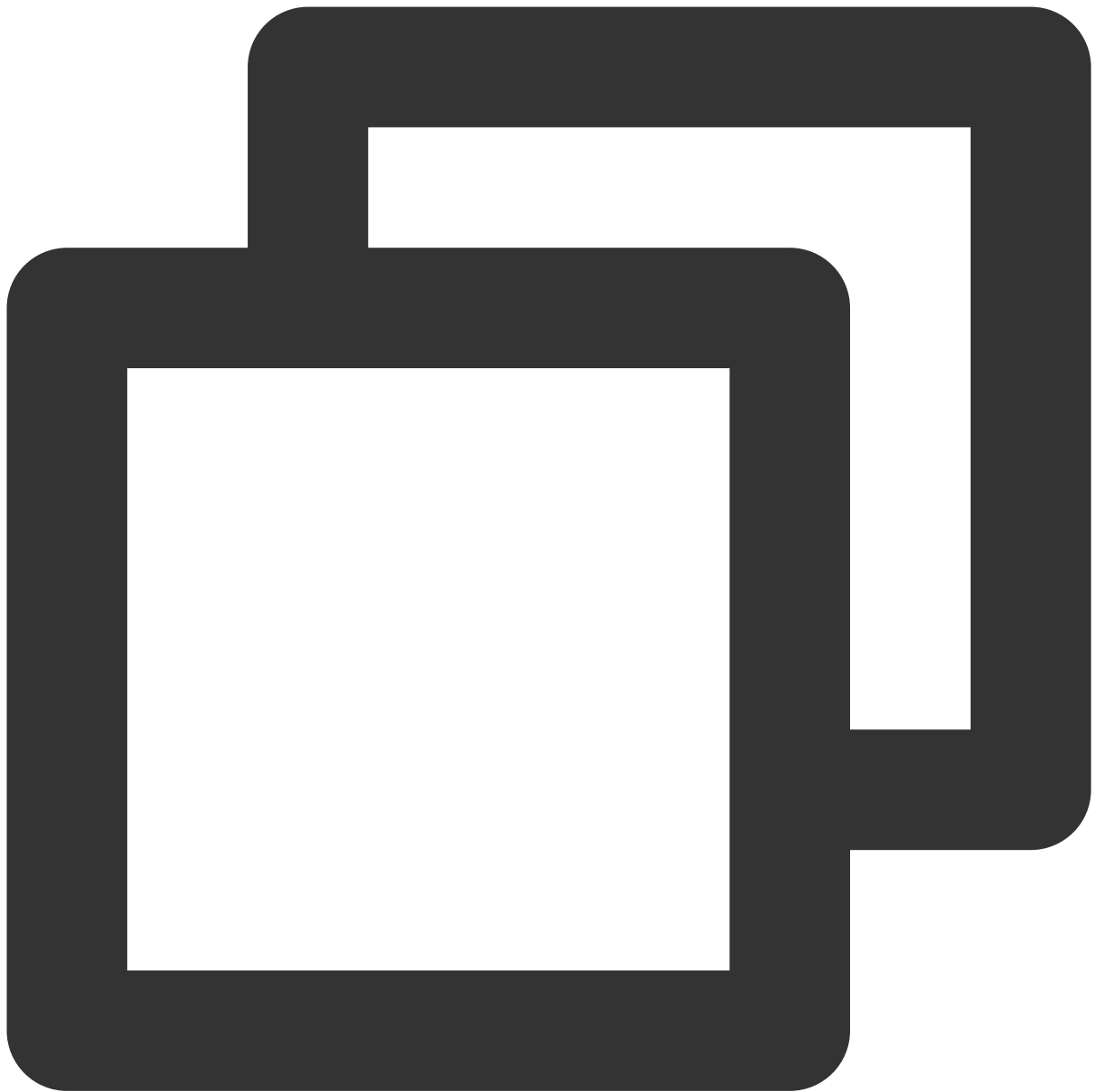
```
TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

The parameters are described below:

Parameter	Type	Description
groupId	String	The group ID.
userIdList	List	The target user IDs.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallDefine.CallParams</a>	An additional parameter. such as roomId, call timeout, offline push info,etc

## accept

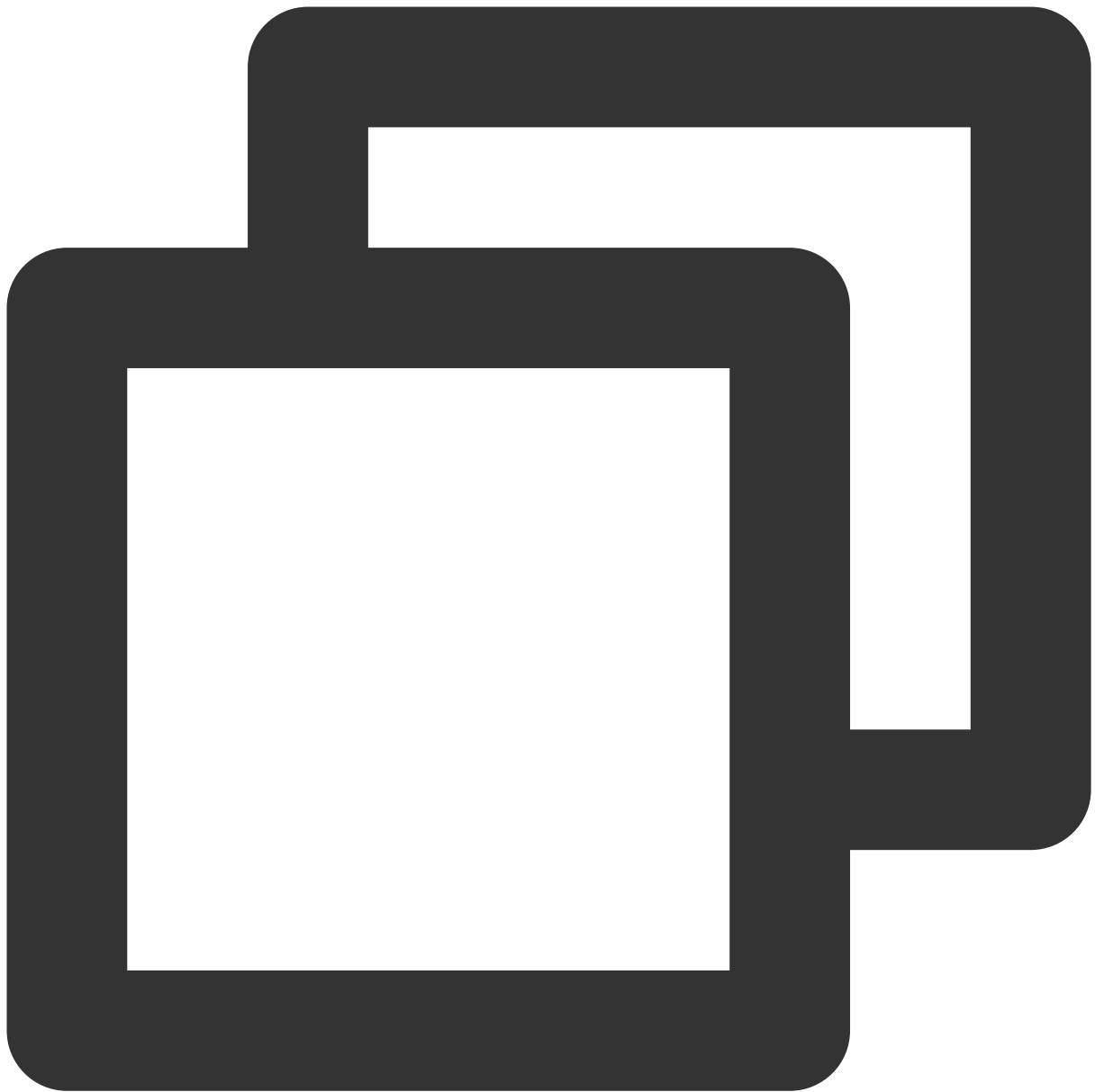
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



```
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.

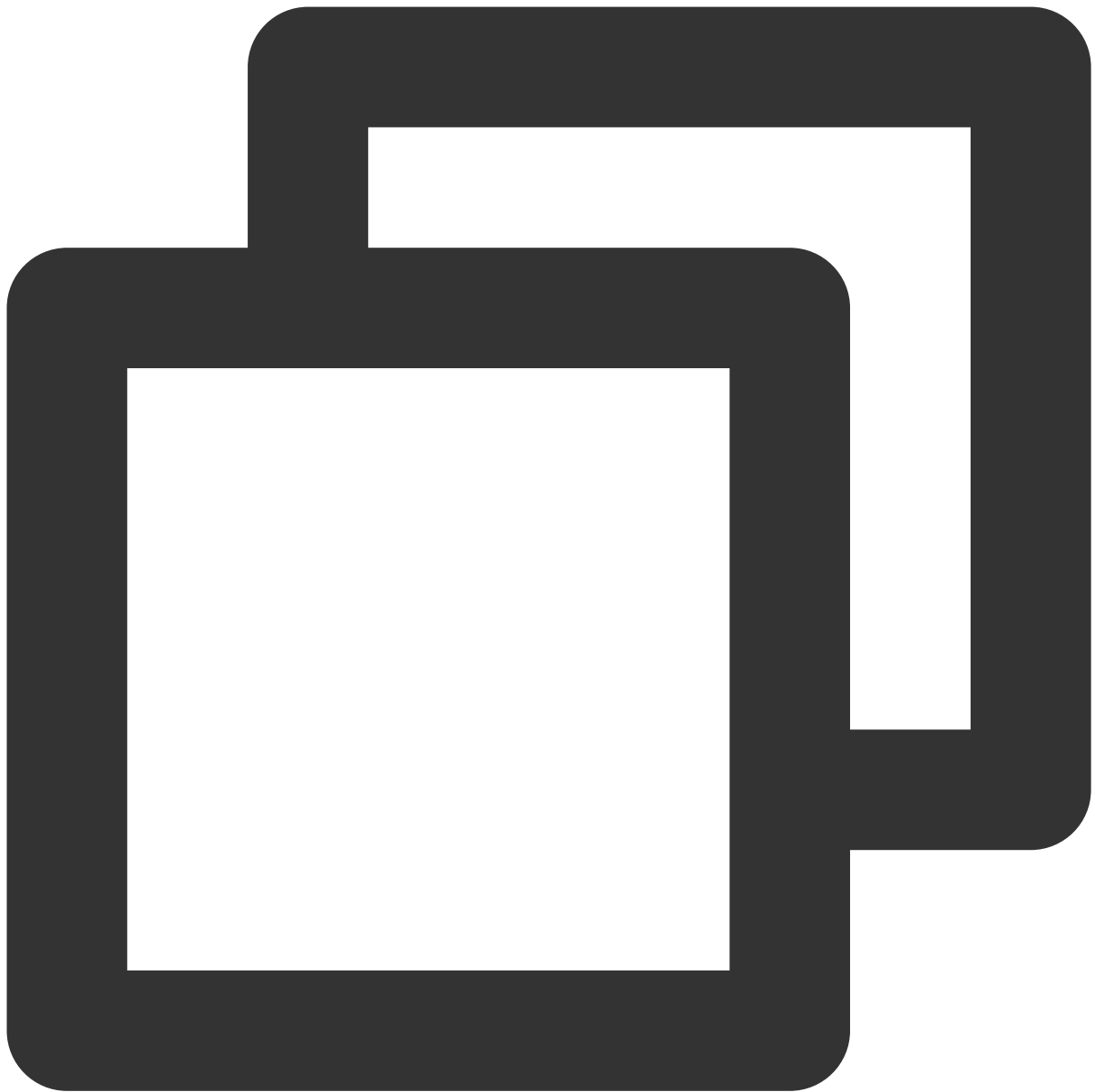


```
void reject(TUICommonDefine.Callback callback);
```

## 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.

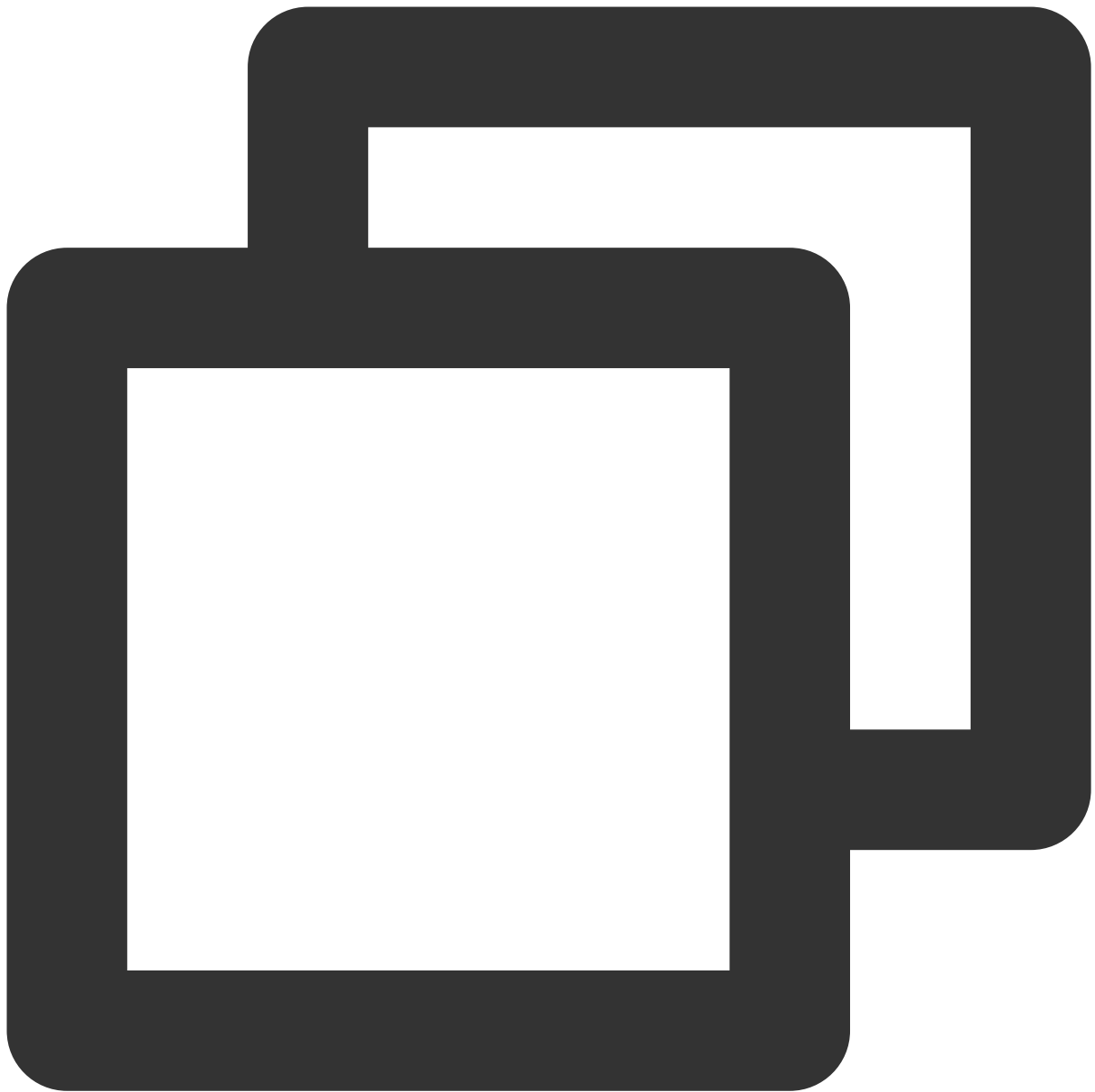
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



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

## hangup

This API is used to end a call.

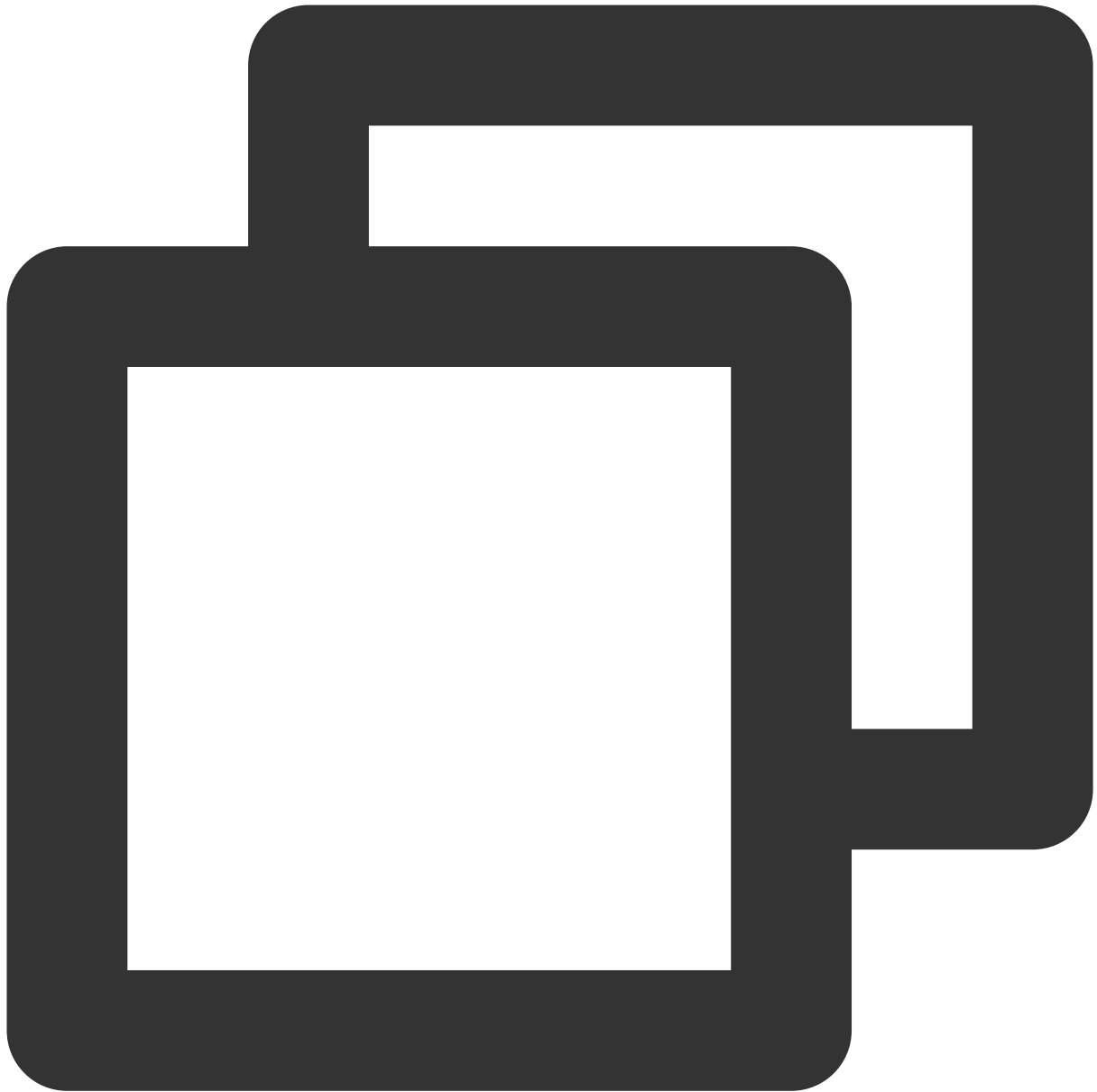


```
void hangup(TUICommonDefine.Callback callback);
```

### **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	<a href="#">TUICallDefine.CallParams</a>	An additional parameter. such as roomId, call timeout, offline push info,etc

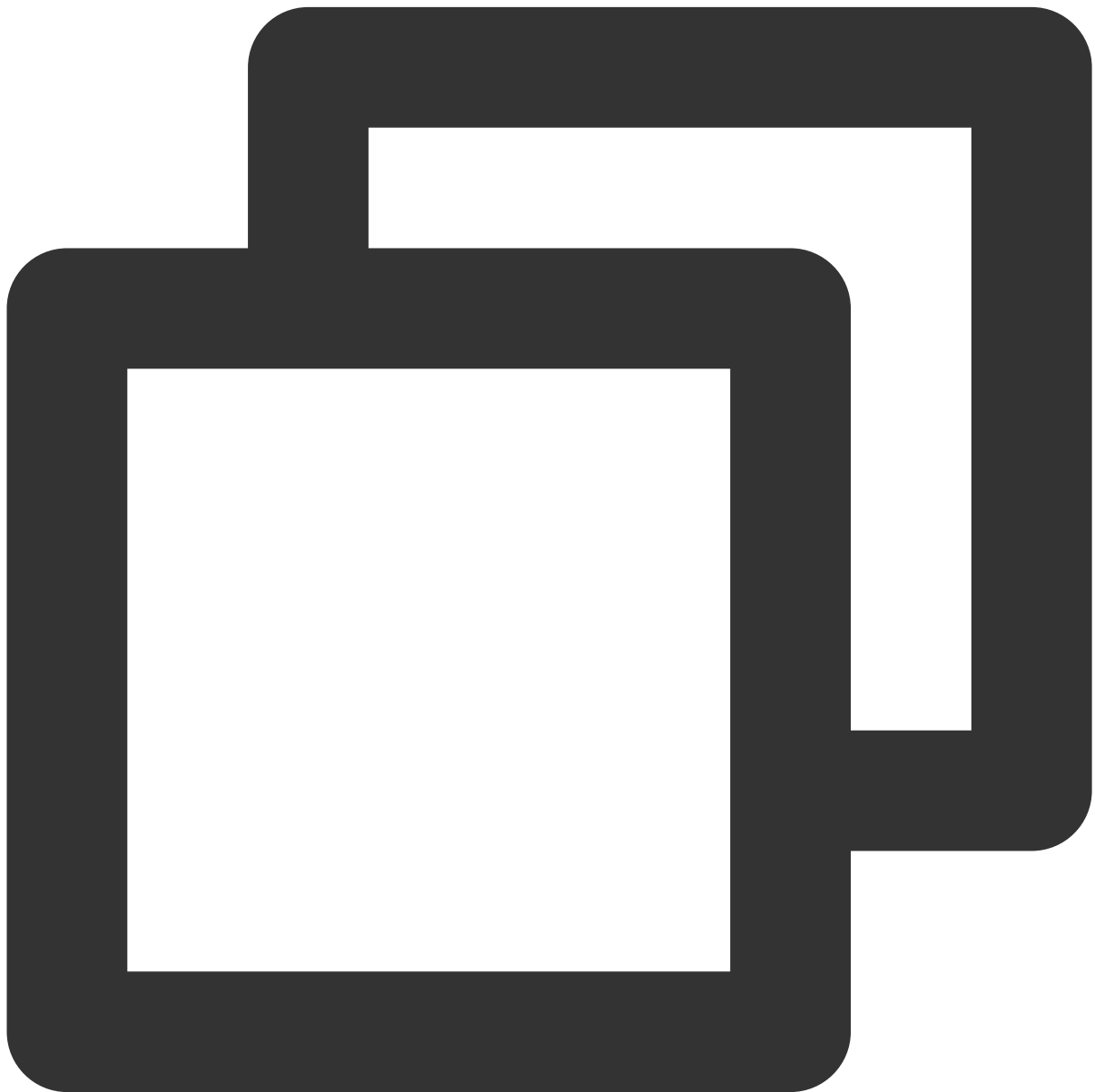
**Notice :**

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
```

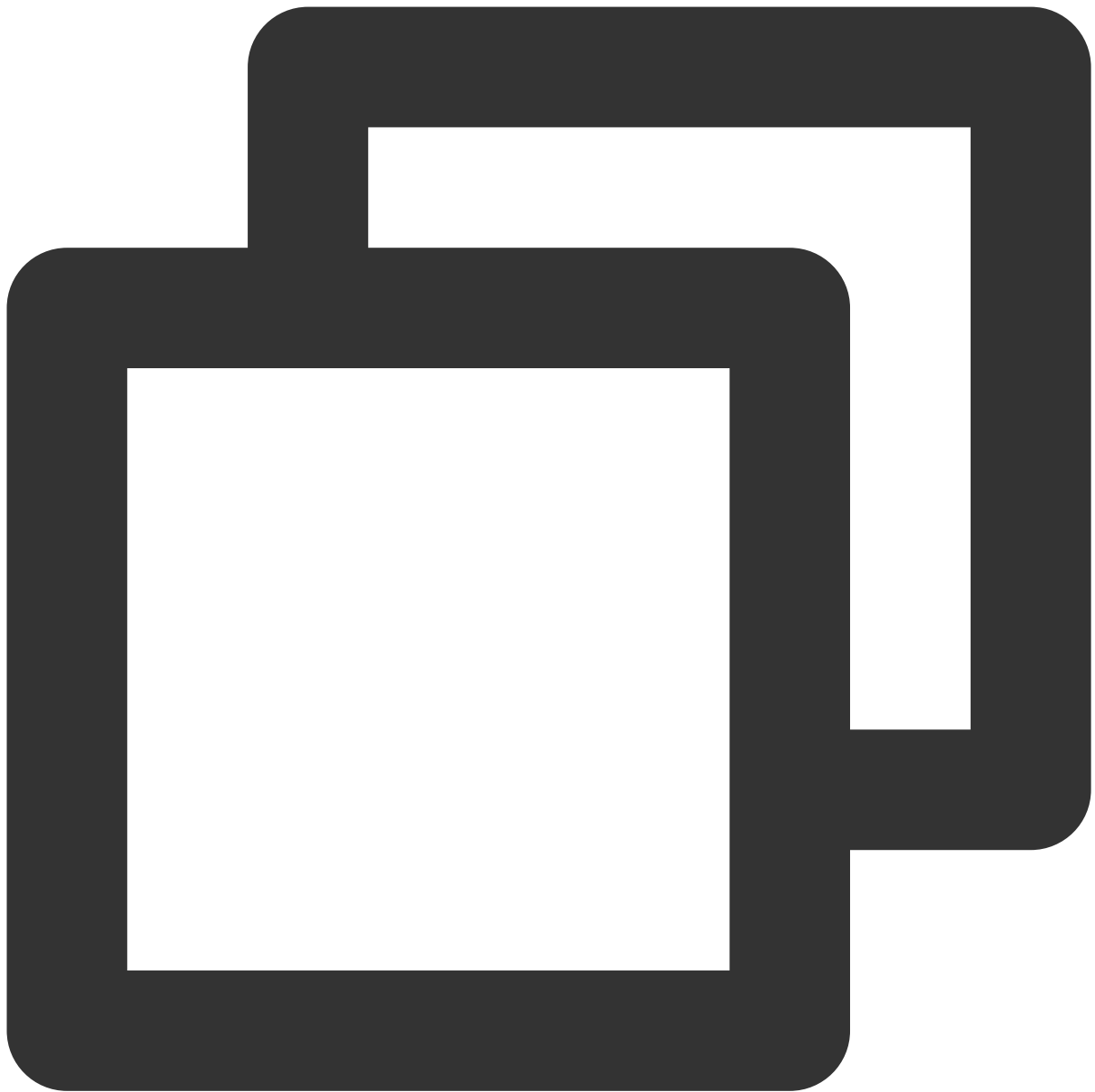
The parameters are described below:

Parameter	Type	Description
roomId	<a href="#">TUICommonDefine.RoomId</a>	The room ID.
groupId	String	The group ID.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.

### **switchCallMediaType**

This API is used to change the call type.





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

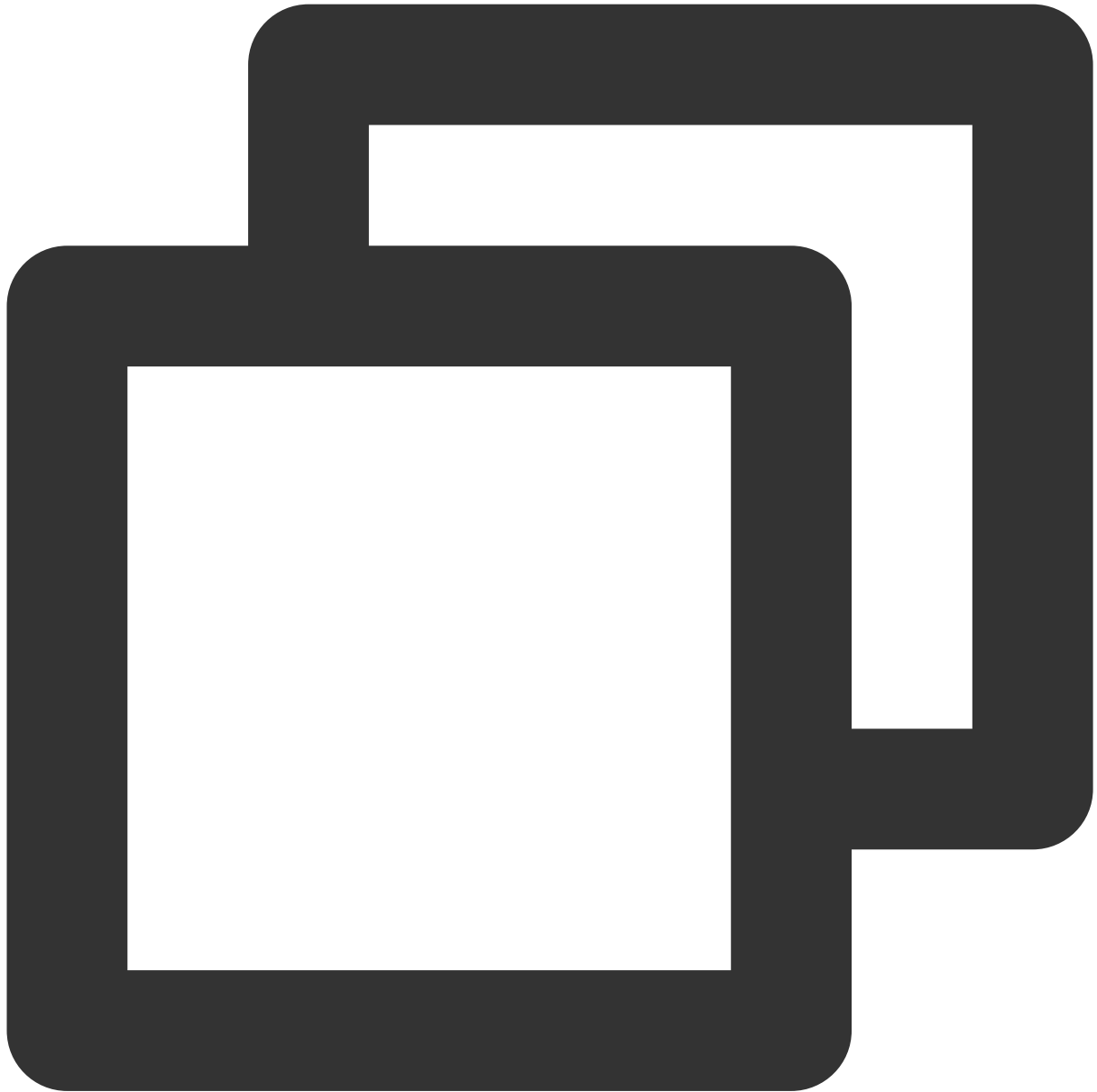
The parameters are described below:

Parameter	Type	Description
callMediaType	<a href="#">TUICallDefine.MediaType</a>	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 .
```



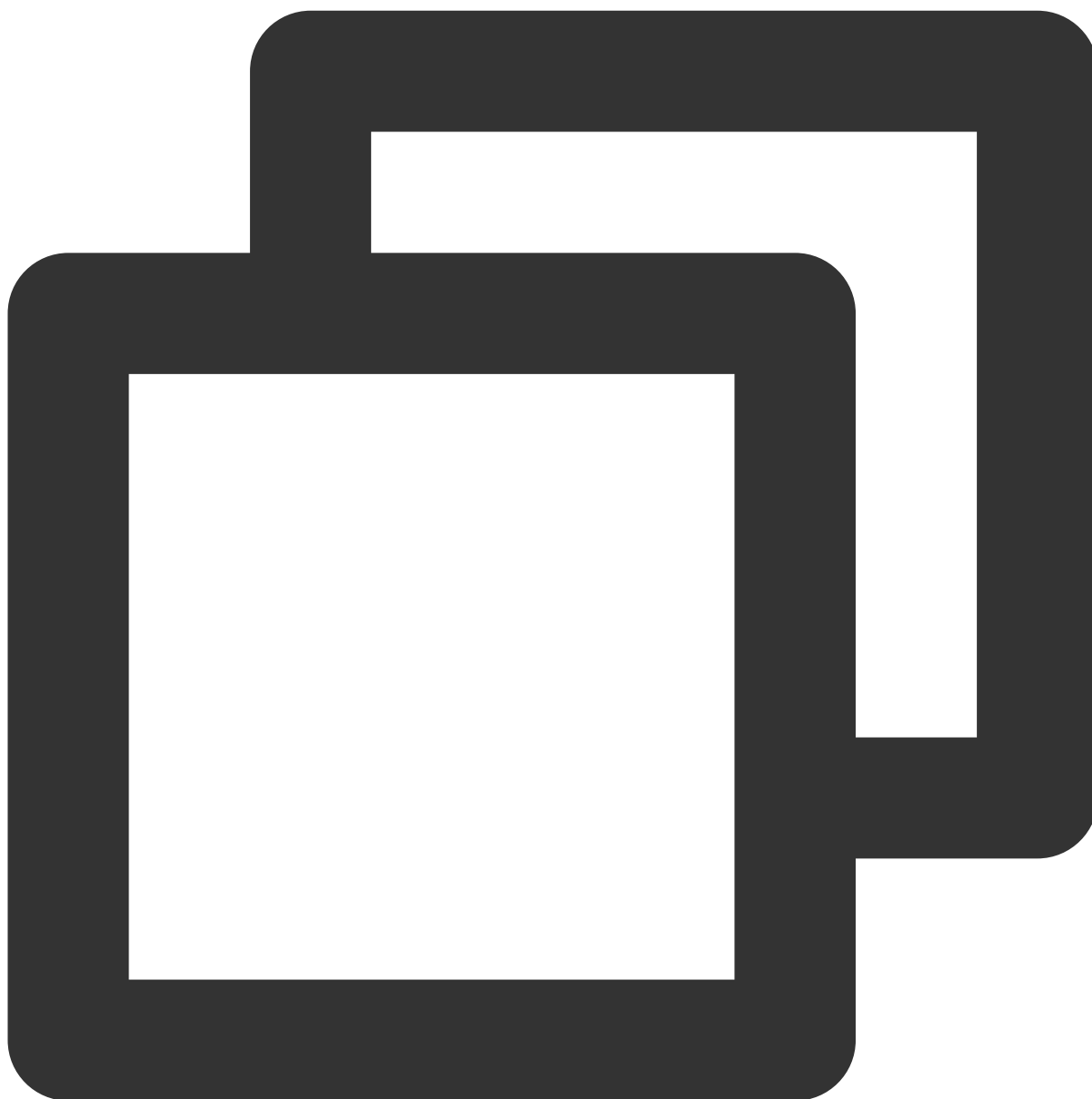
```
void startRemoteView(String userId, TUIVideoView videoView, TUICommonDefine.PlayCal
```

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.



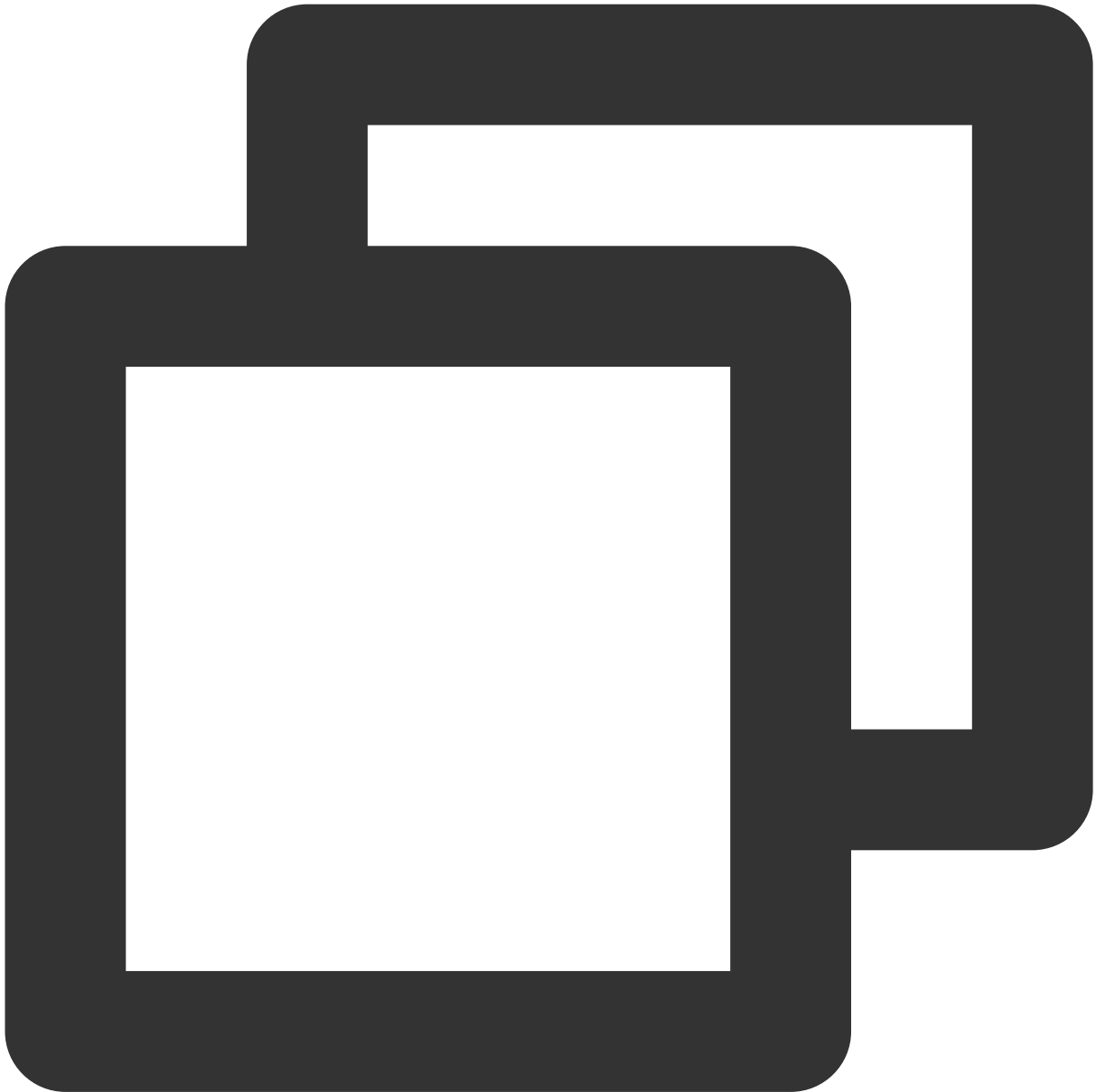
```
void stopRemoteView(String userId);
```

The parameters are described below:

Parameter	Type	Description
userId	String	The target user ID.

## openCamera

This API is used to turn the camera on.



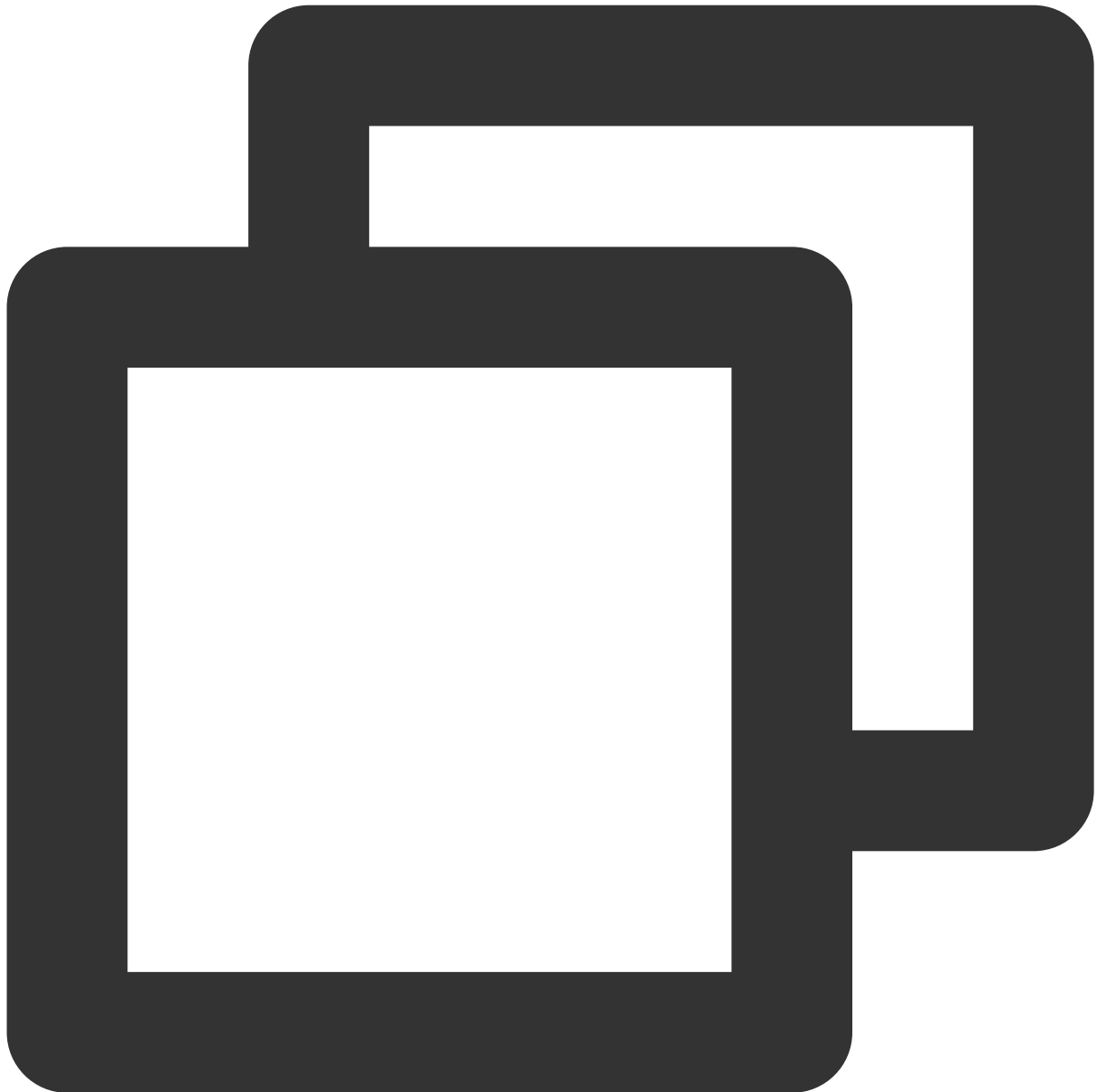
```
void openCamera(TUICommonDefine.Camera camera, TUIVideoView videoView, TUICommonDef
```

The parameters are described below:

Parameter	Type	Description
camera	<a href="#">TUICommonDefine.Camera</a>	The front or rear camera.
videoView	TUIVideoView	The view to be rendered.

## closeCamera

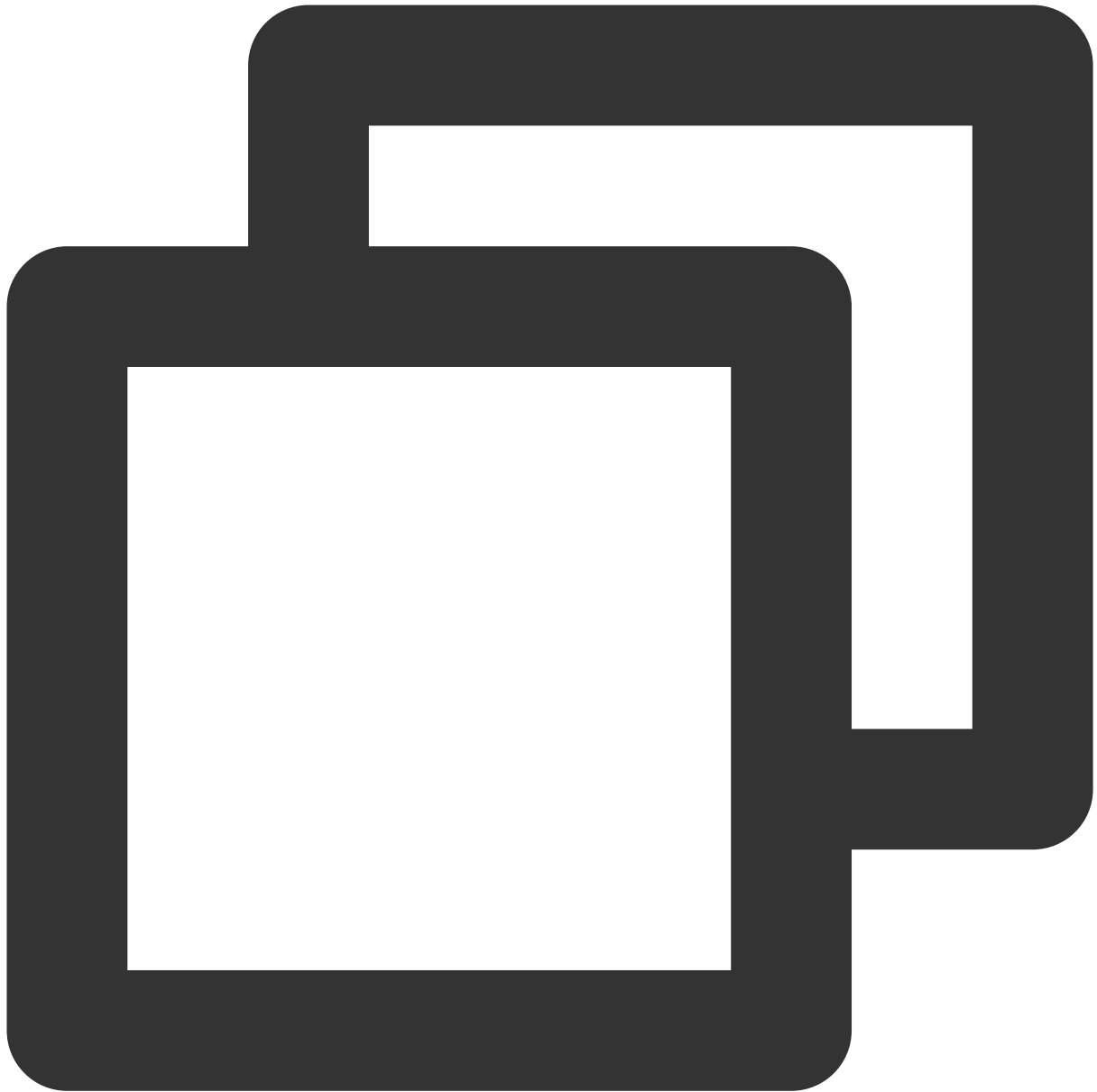
This API is used to turn the camera off.



```
void closeCamera();
```

## switchCamera

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



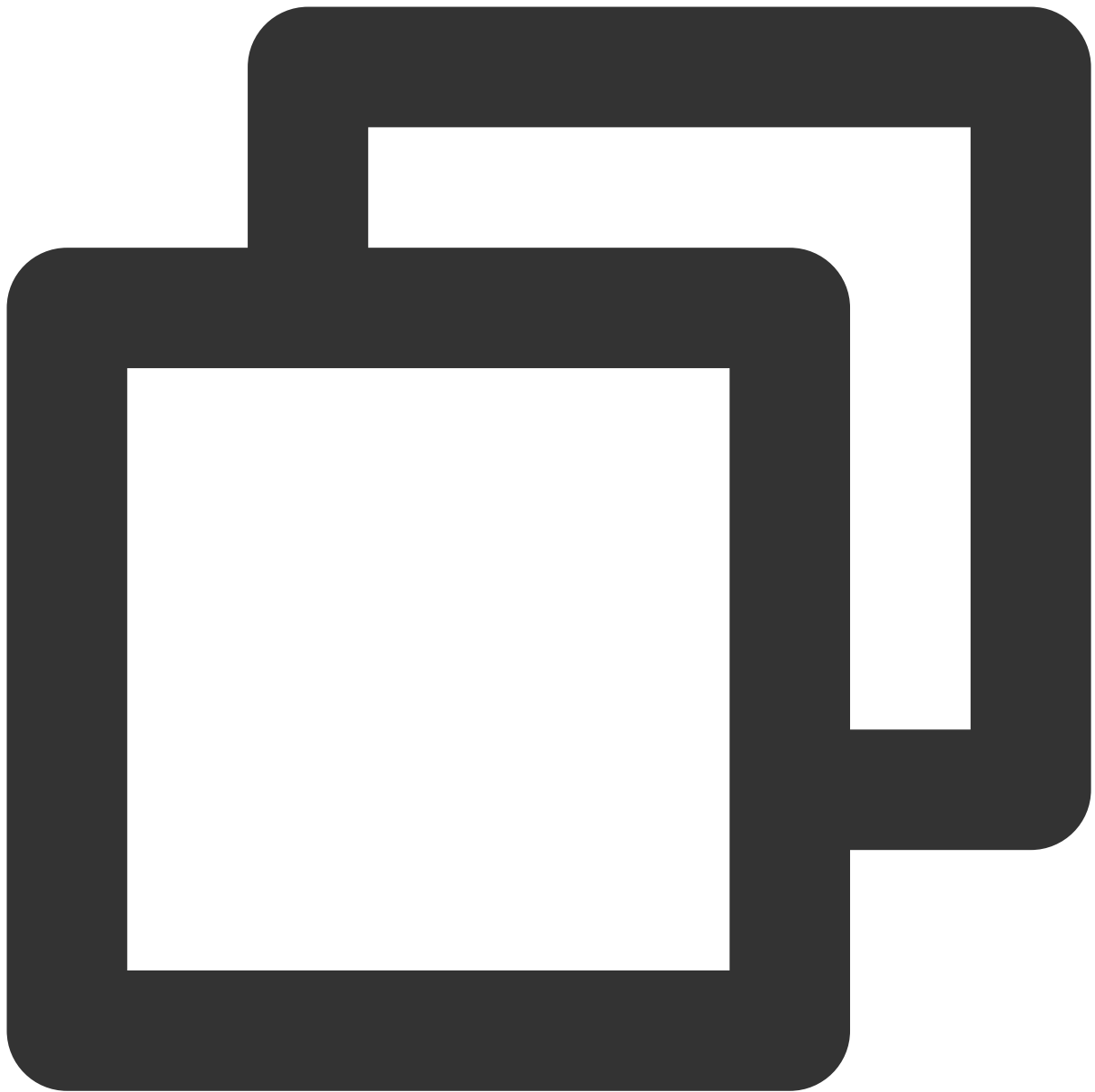
```
void switchCamera(TUICommonDefine.Camera camera);
```

The parameters are described below:

Parameter	Type	Description
camera	<a href="#">TUICommonDefine.Camera</a>	The front or rear camera.

## openMicrophone

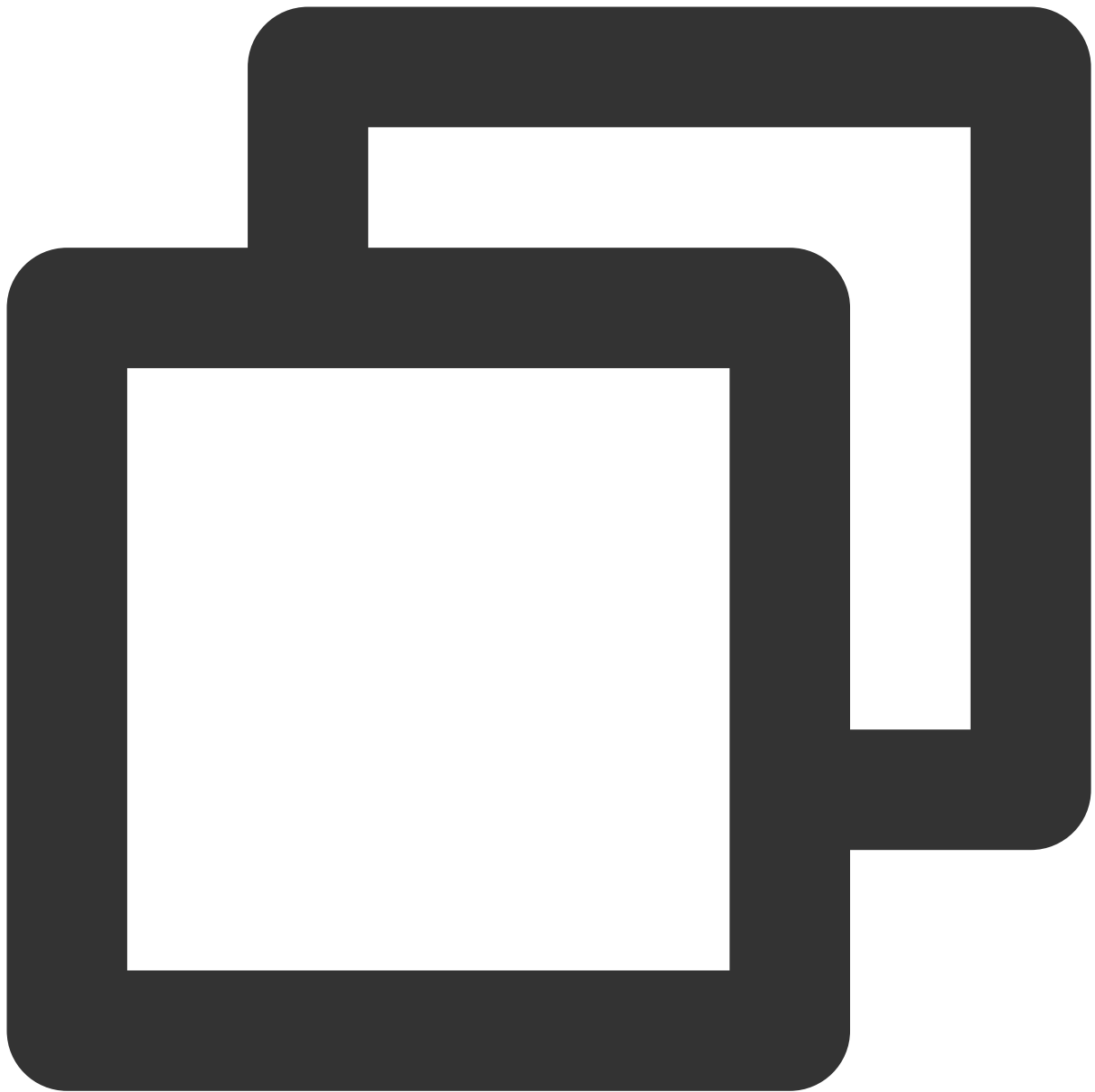
This API is used to turn the mic on.



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

### **closeMicrophone**

This API is used to turn the mic off.

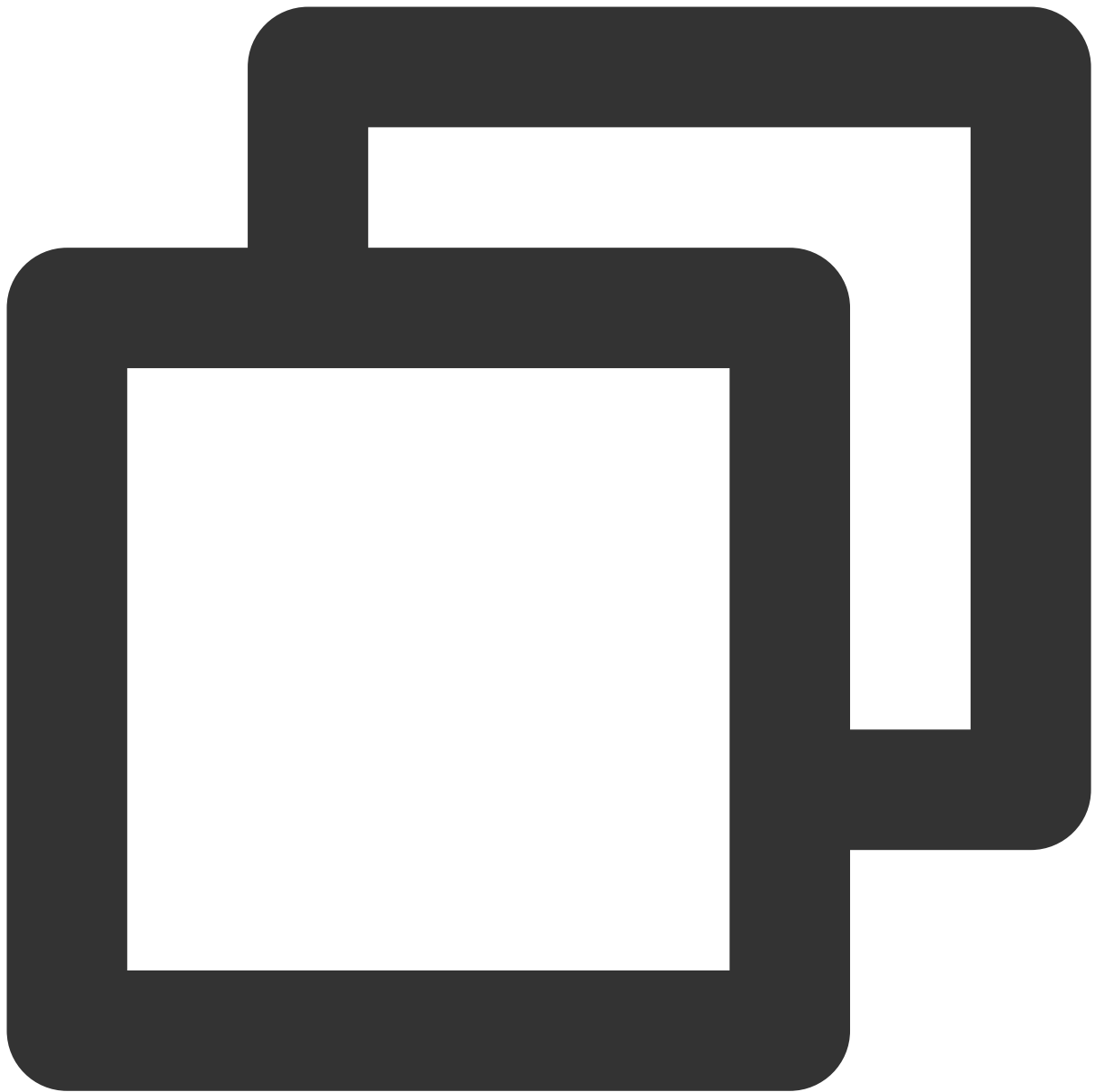


```
void closeMicrophone();
```

### **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.





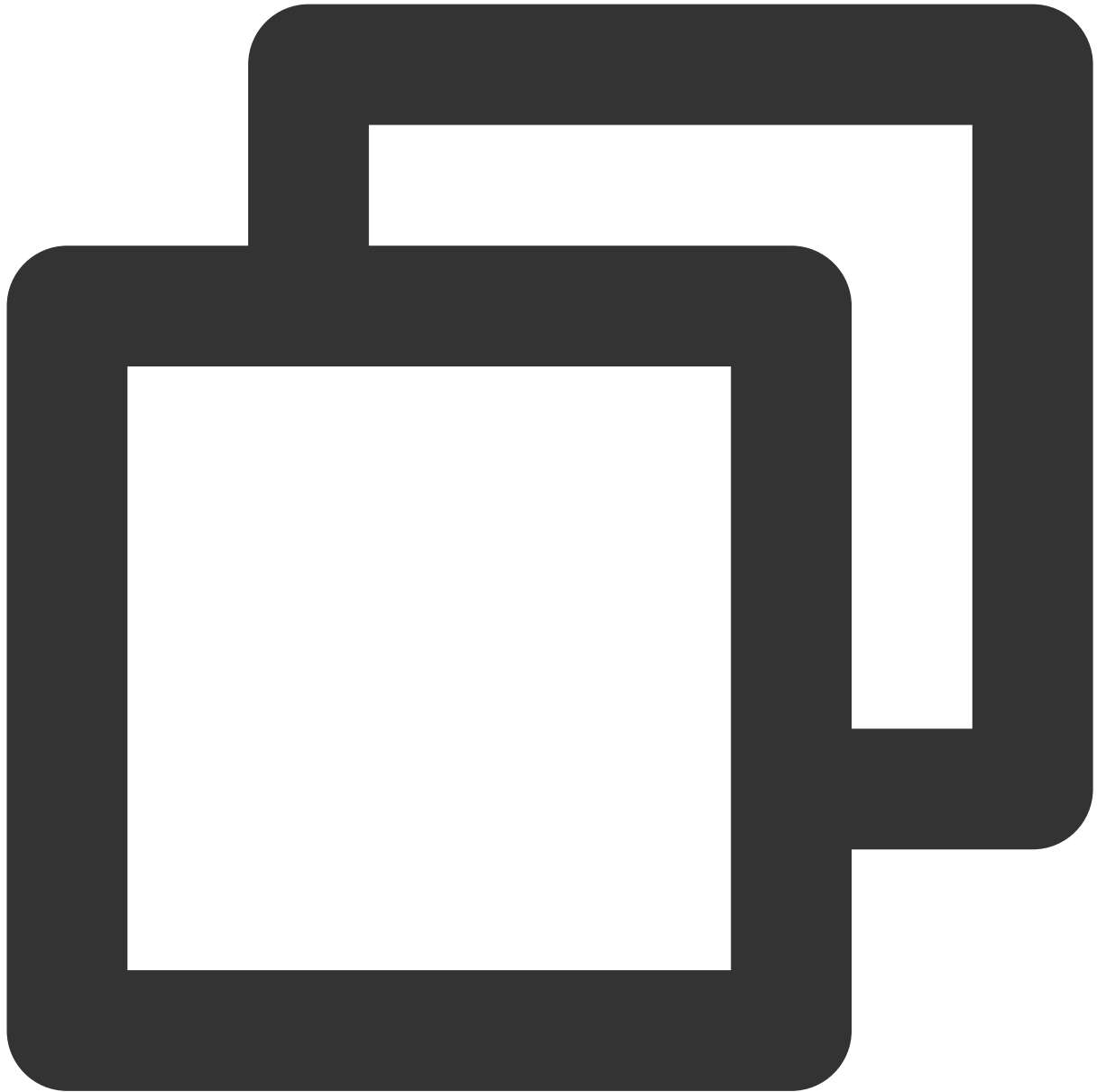
```
void selectAudioPlaybackDevice(TUICommonDefine.AudioPlaybackDevice device);
```

The parameters are described below:

Parameter	Type	Description
device	<a href="#">TUICommonDefine.AudioPlaybackDevice</a>	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.



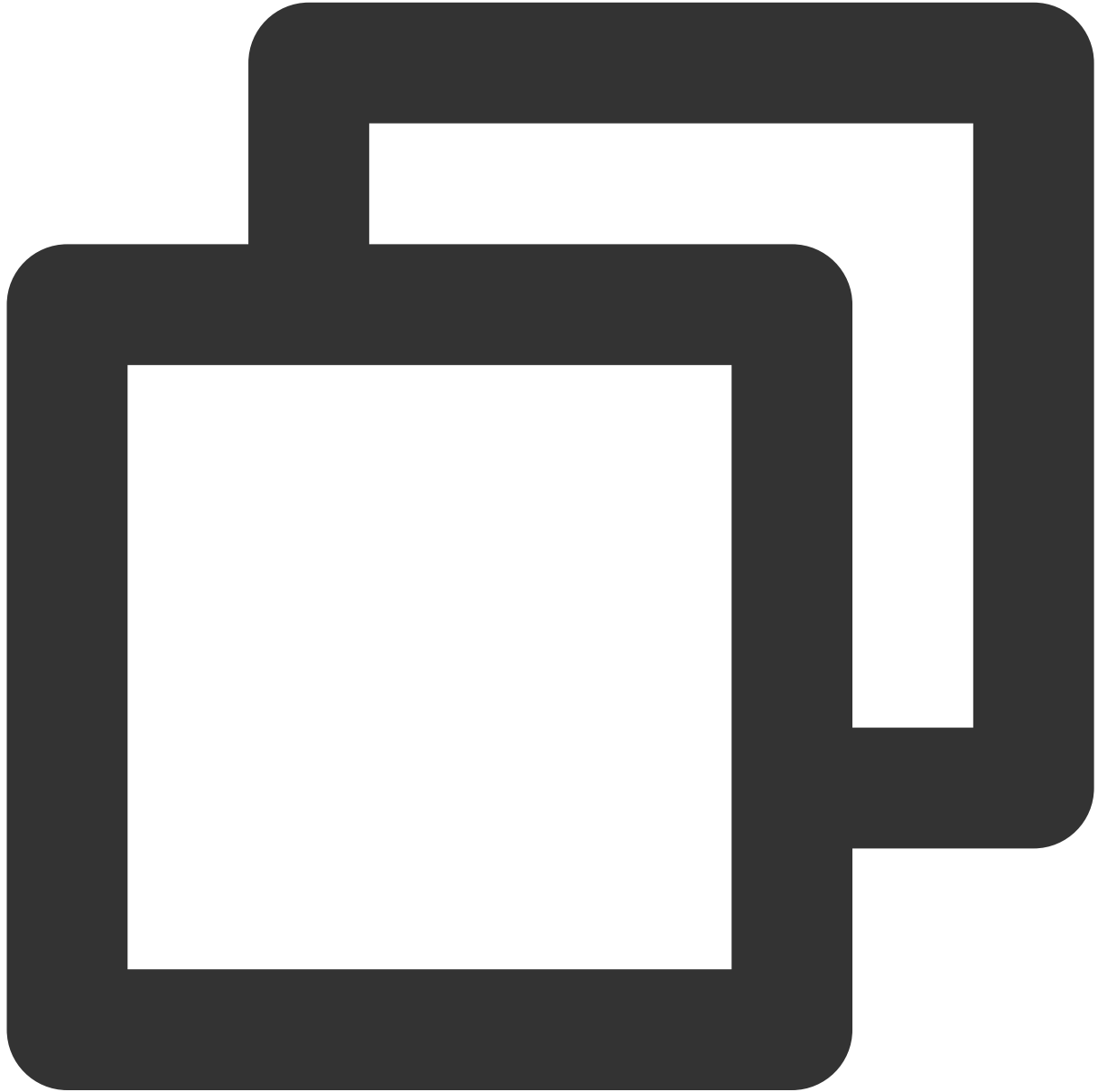
```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

The parameters are described below:

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

## enableMultiDeviceAbility

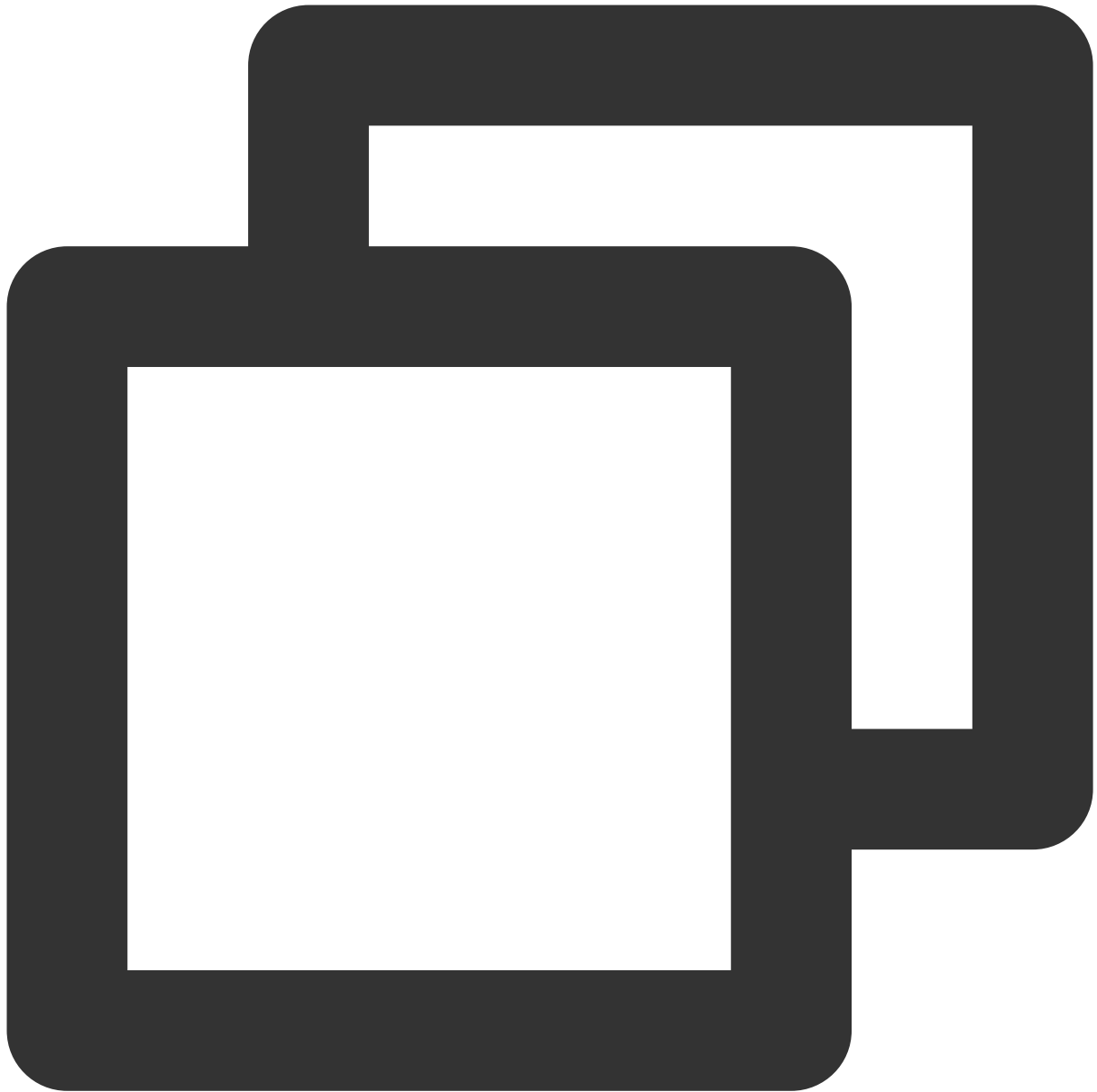
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the [Group Call package](#)).



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

## setVideoRenderParams

Set the rendering mode of video image.



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

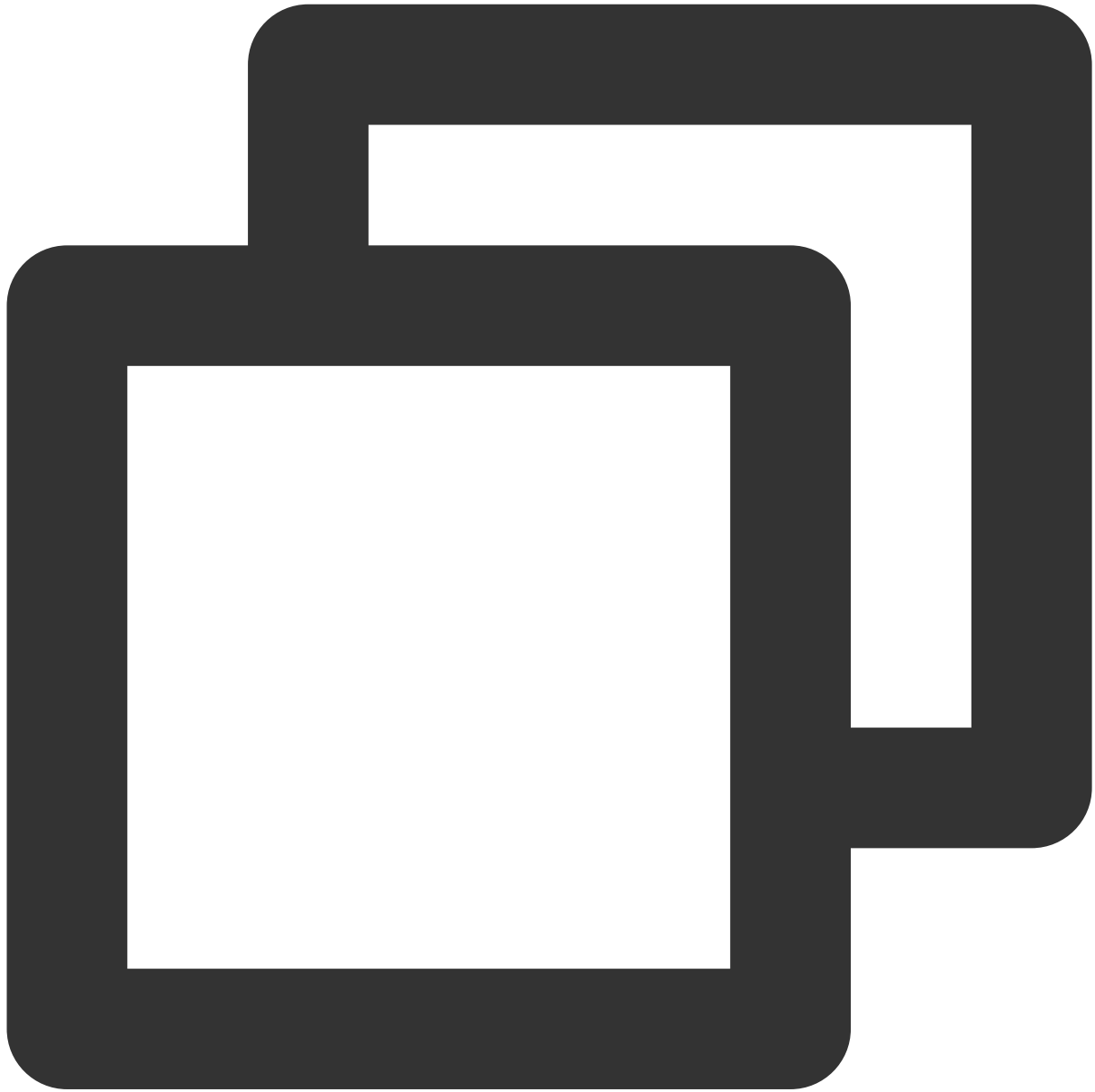
The parameters are described below:

Parameter	Type	Description
userId	String	The target user ID.
params	<a href="#">TUICommonDefine.VideoRenderParams</a>	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.



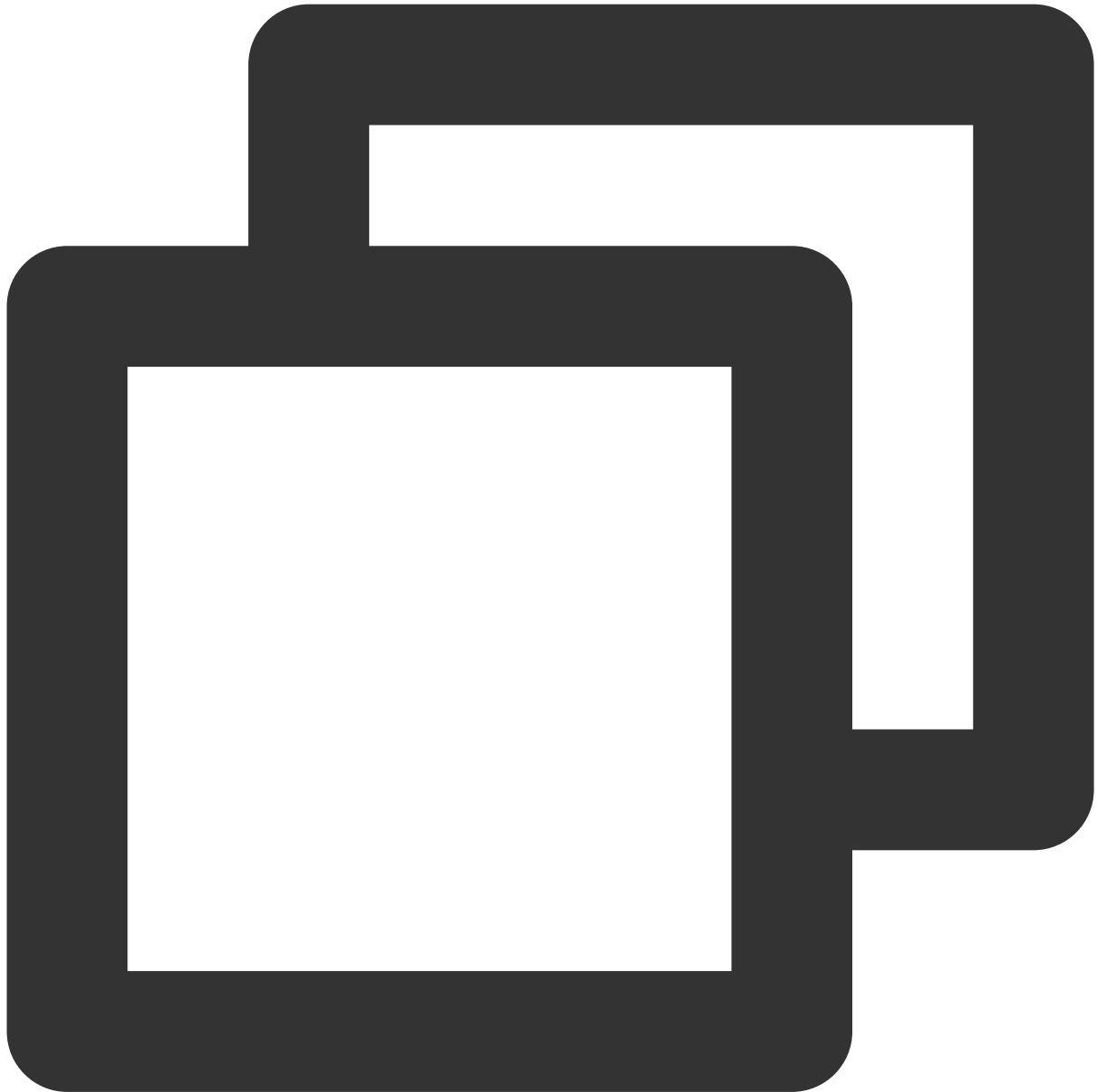
```
void setVideoEncoderParams(TUICommonDefine.VideoEncoderParams params, TUICommonDefi
```

The parameters are described below:

Parameter	Type	Description
params	<a href="#">TUICommonDefine.VideoEncoderParams</a>	Video encoding parameters

## getTRTCCloudInstance

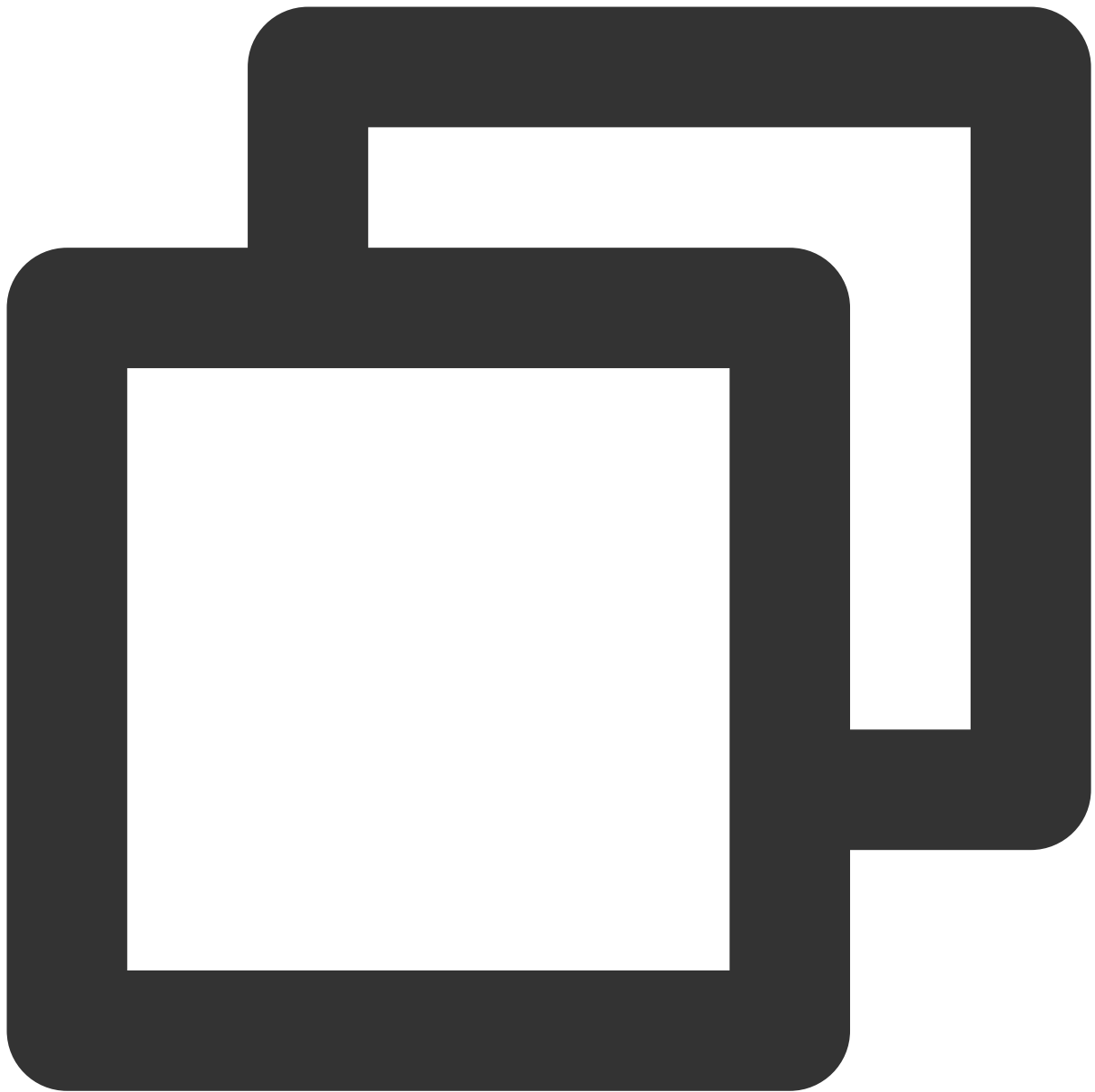
Advanced features.



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

`onUserSigExpired`

The user sig is expired.

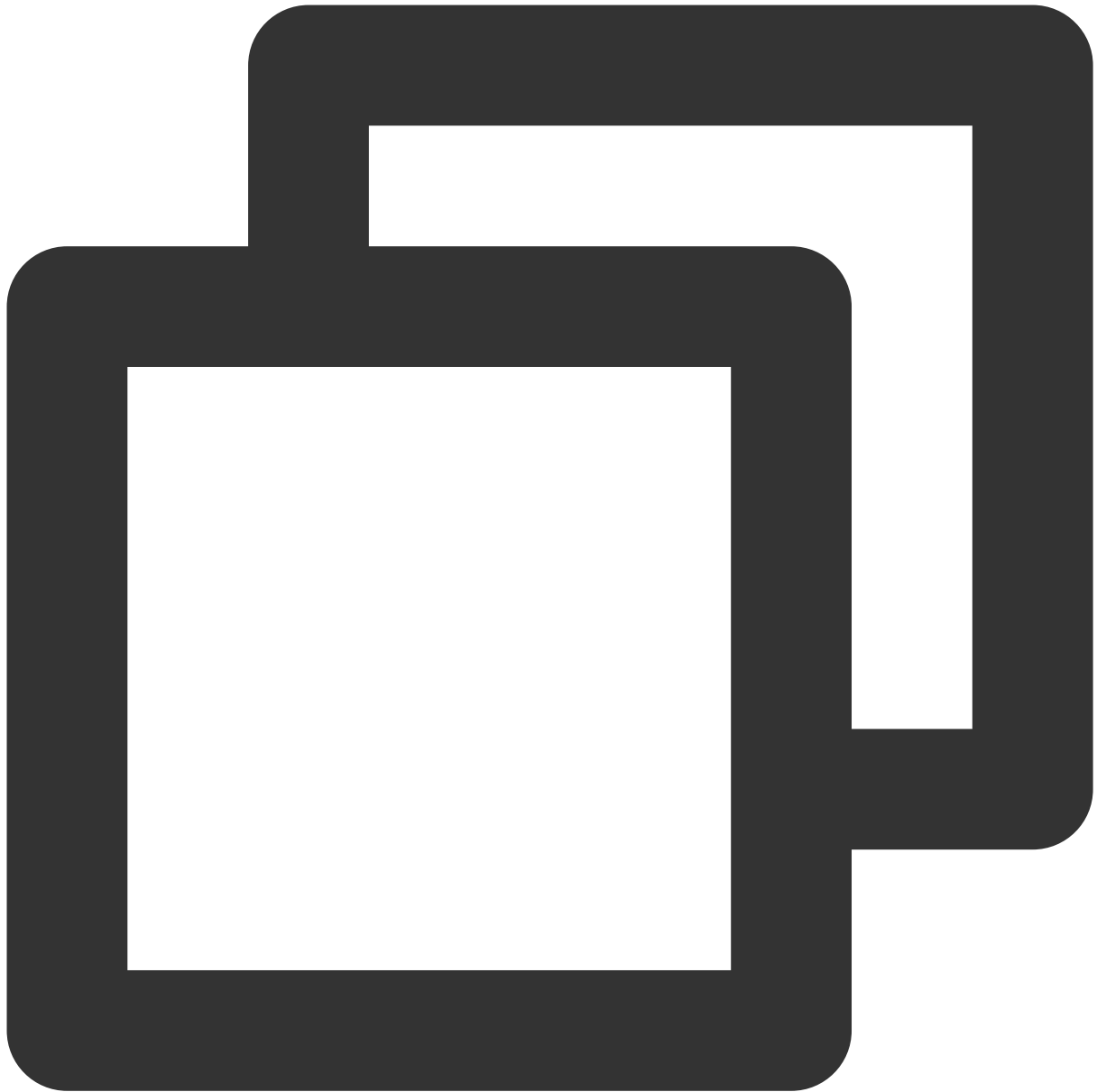
## Details

### **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.



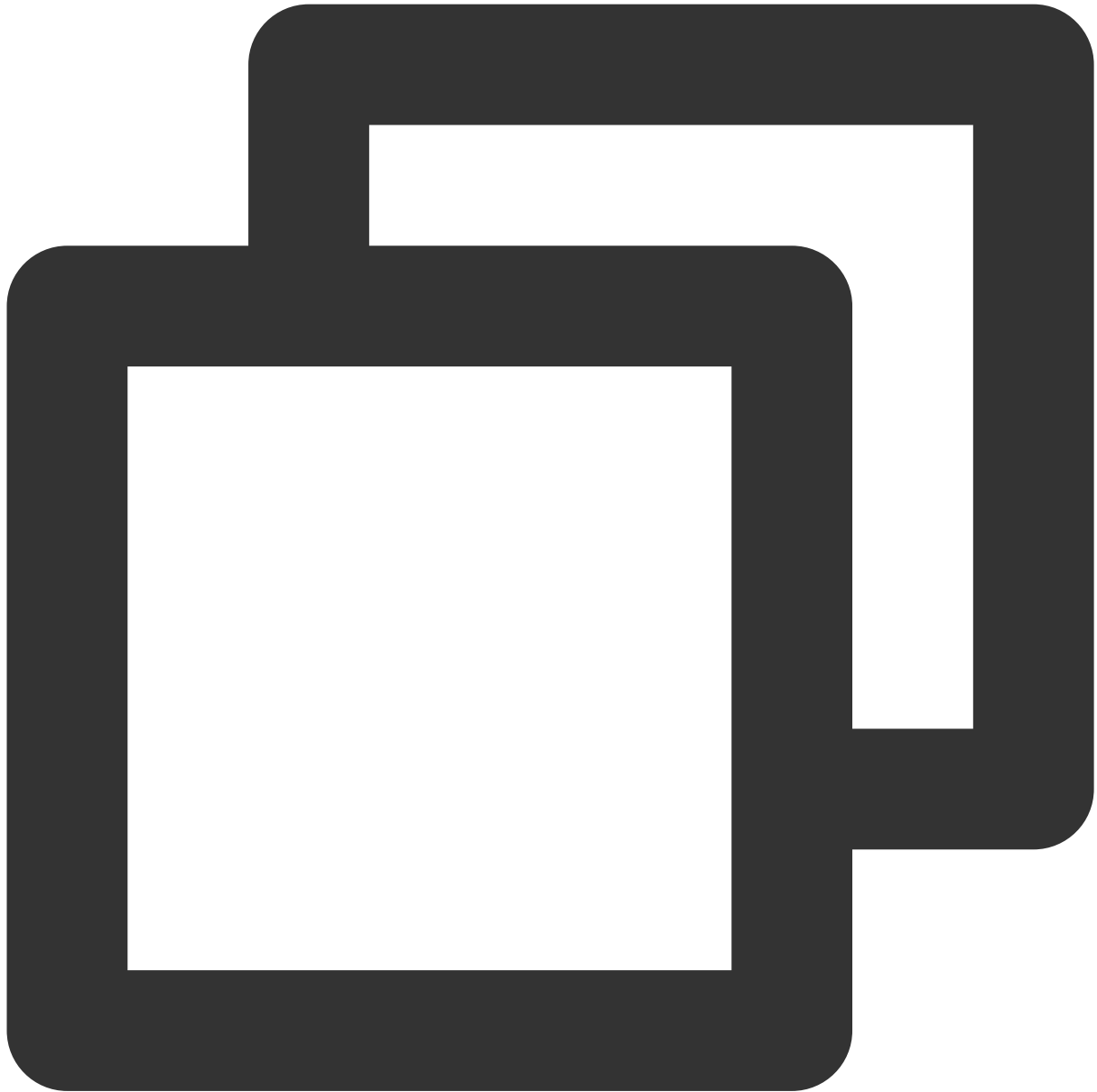
```
void onError(int code, String msg);
```

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	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.
userData	String	User-added extended fields., Please refer to: <a href="#">TUICallDefine.CallParams</a>

## 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.

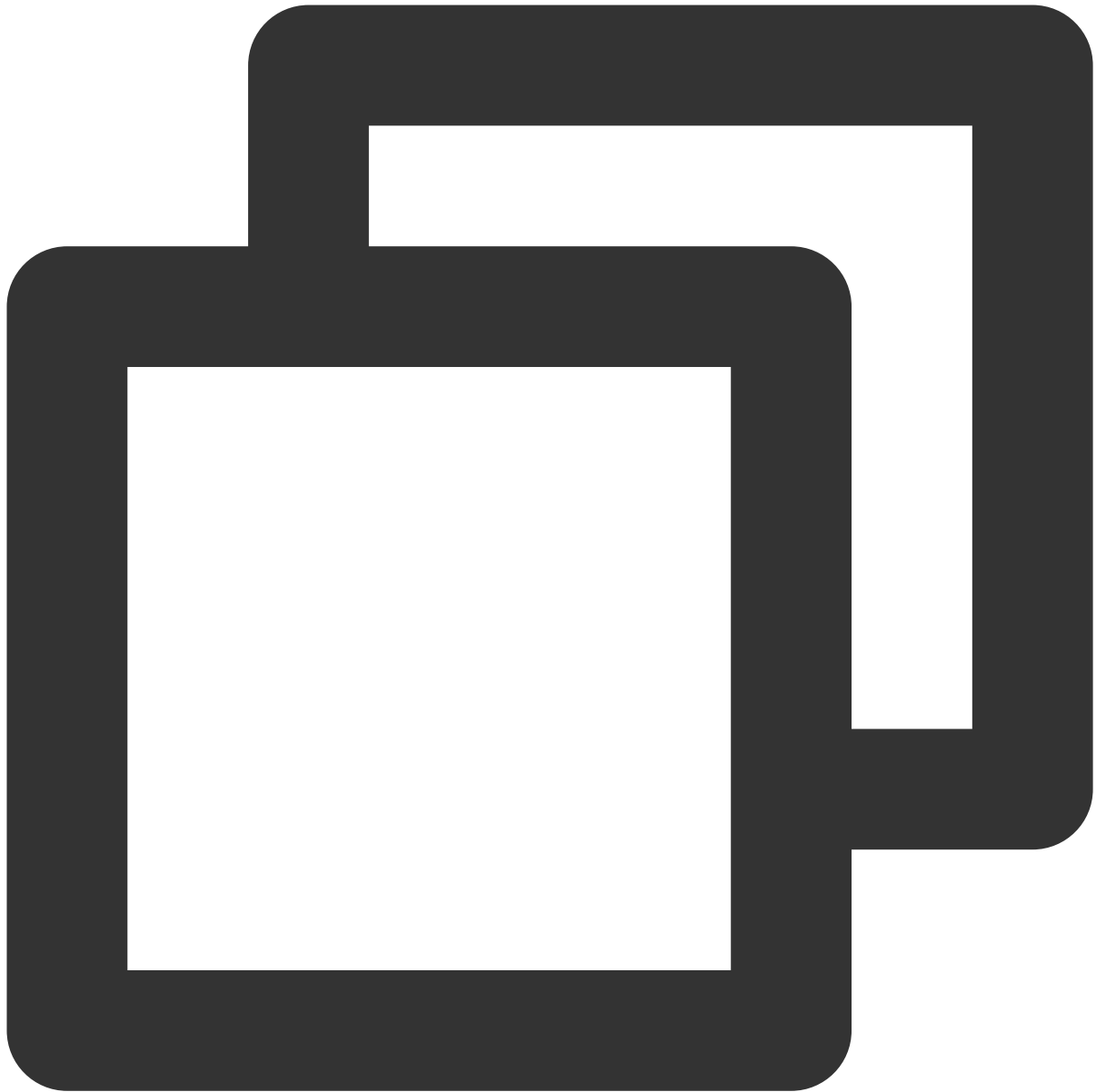
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller).

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).



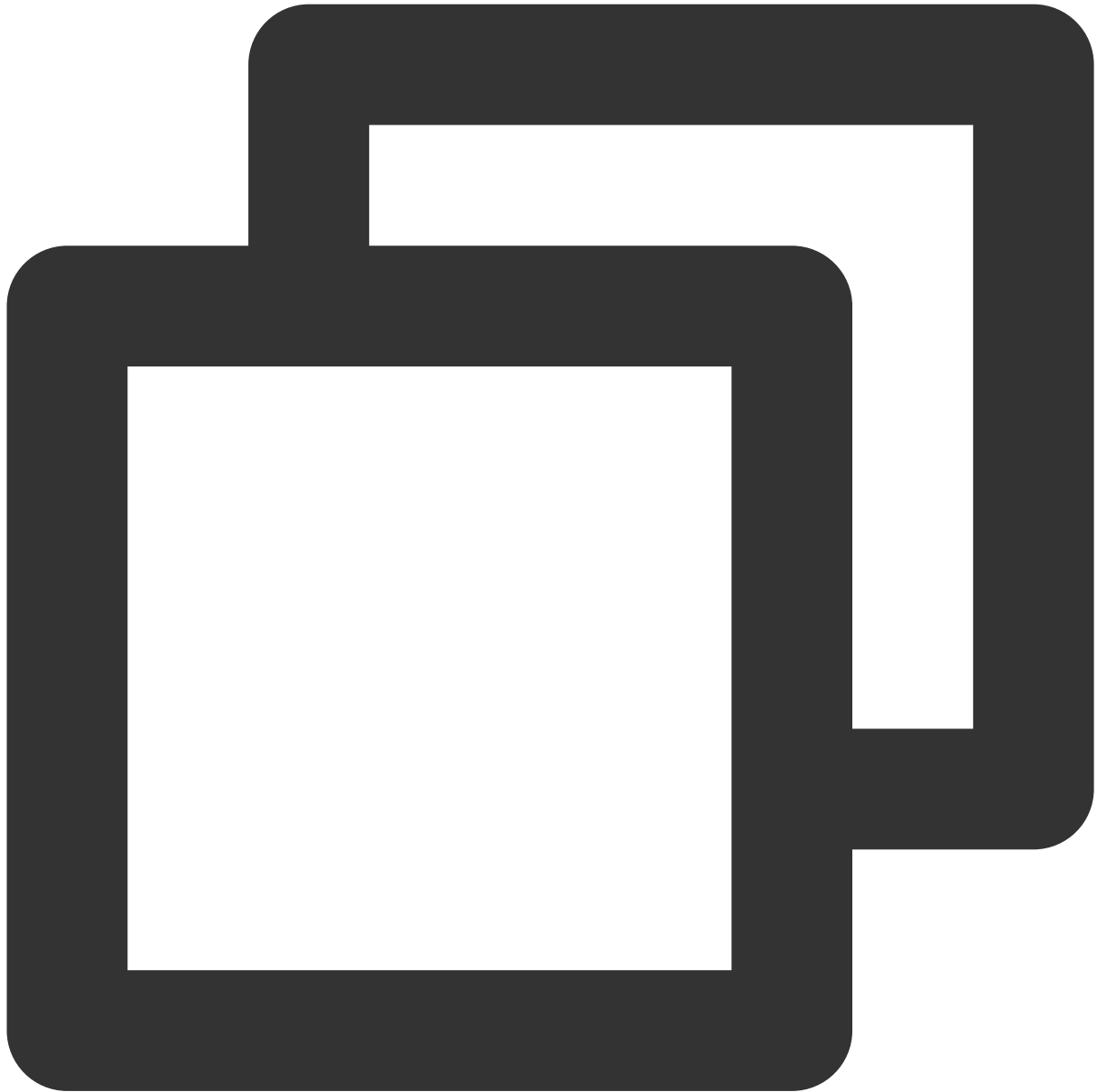
```
void onCallCancelled(String userId);
```

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
```

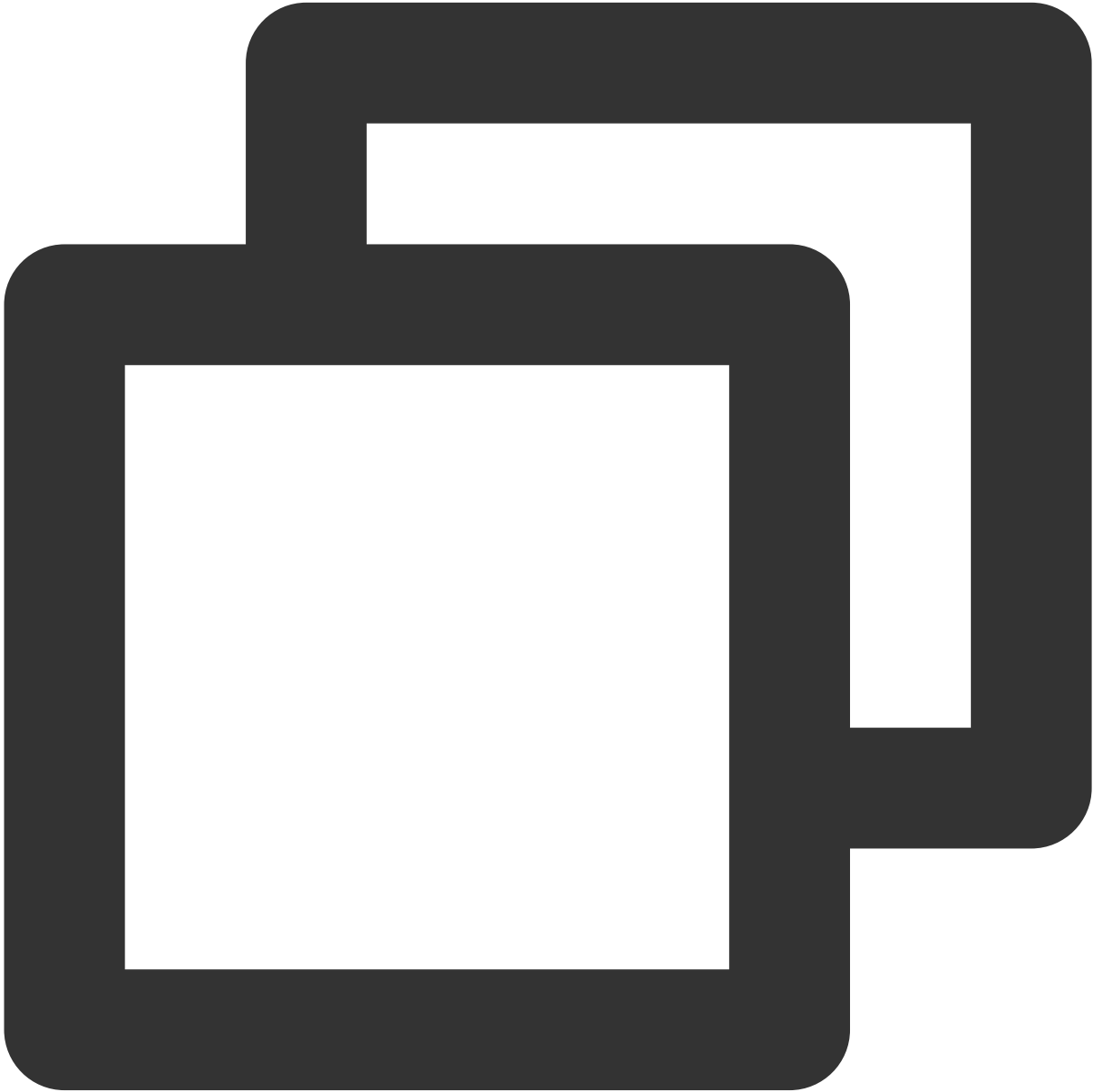
The parameters are described below:

Parameter	Type	Description
roomId	<a href="#">TUICommonDefine.RoomId</a>	The room ID.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.

callRole	<a href="#">TUICallDefine.Role</a>	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:

--	--	--



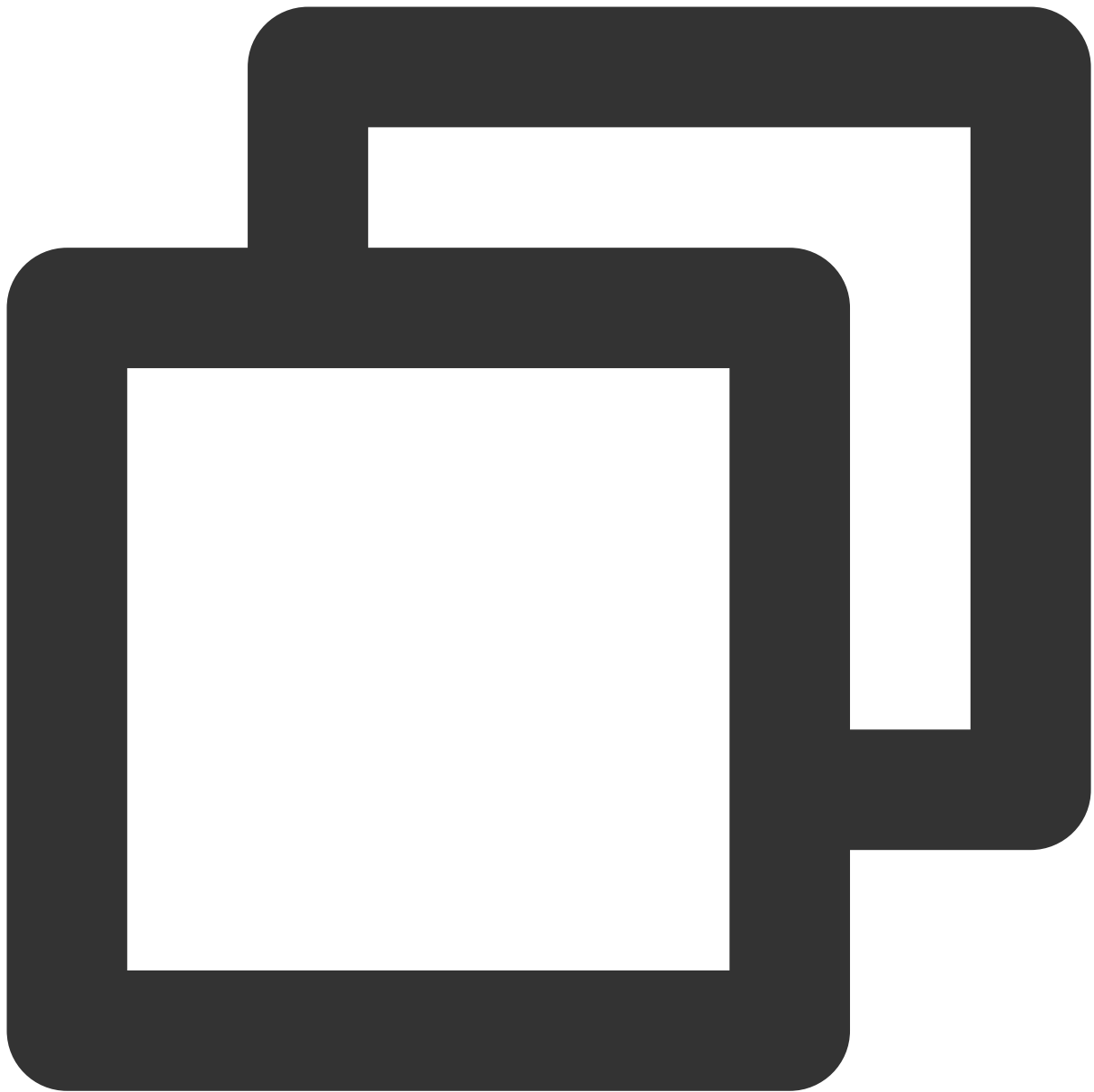
Parameter	Type	Description
roomId	<a href="#">TUICommonDefine.RoomId</a>	The room ID.
callMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type, which can be video or audio.
callRole	<a href="#">TUICallDefine.Role</a>	The role, which can be caller or callee.
totalTime	long	The call duration.

**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.



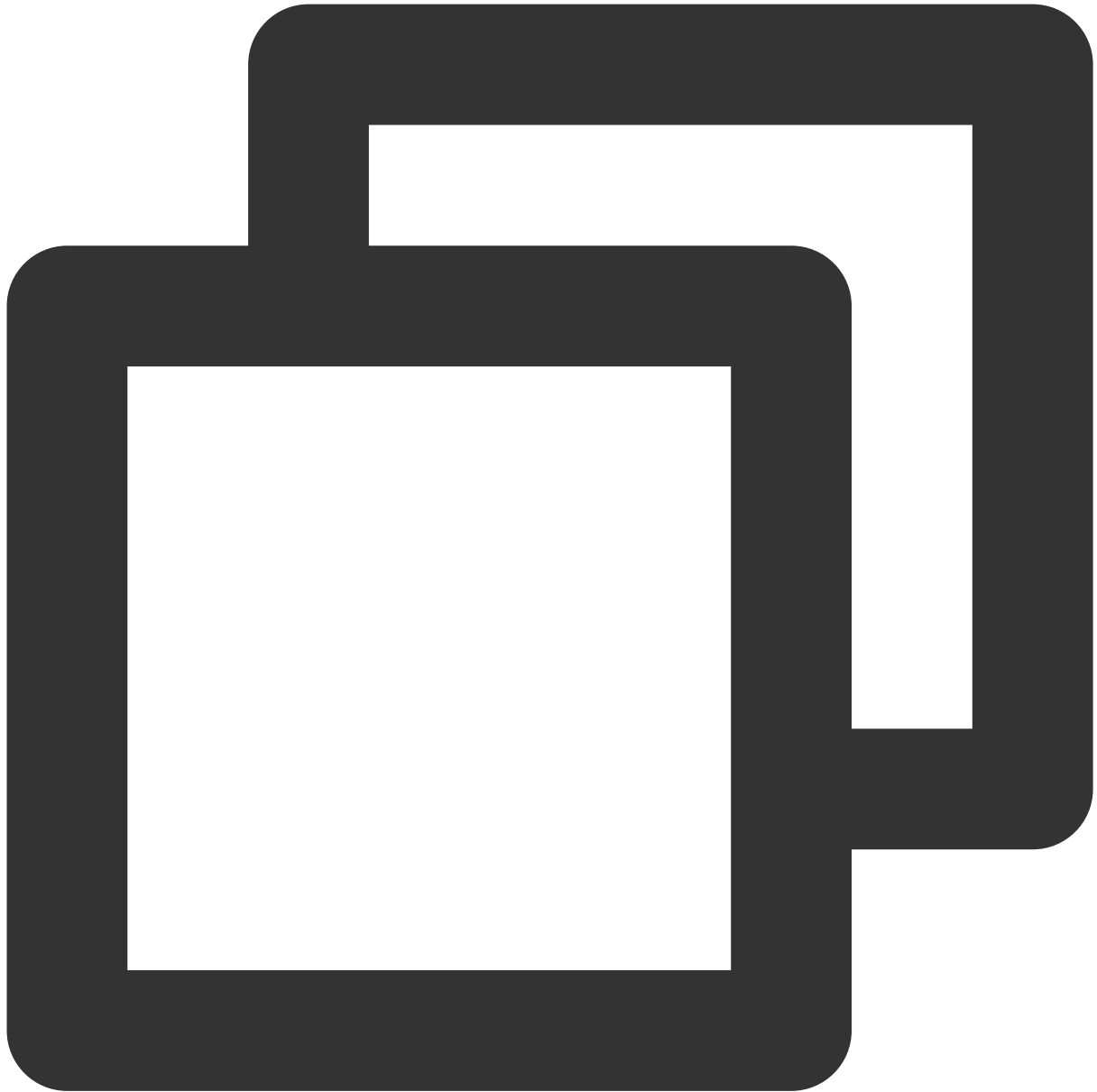
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType, TUICallDefine.
```

The parameters are described below:

Parameter	Type	Description
oldCallMediaType	<a href="#">TUICallDefine.MediaType</a>	The call type before the change.
newCallMediaType	<a href="#">TUICallDefine.MediaType</a>	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.



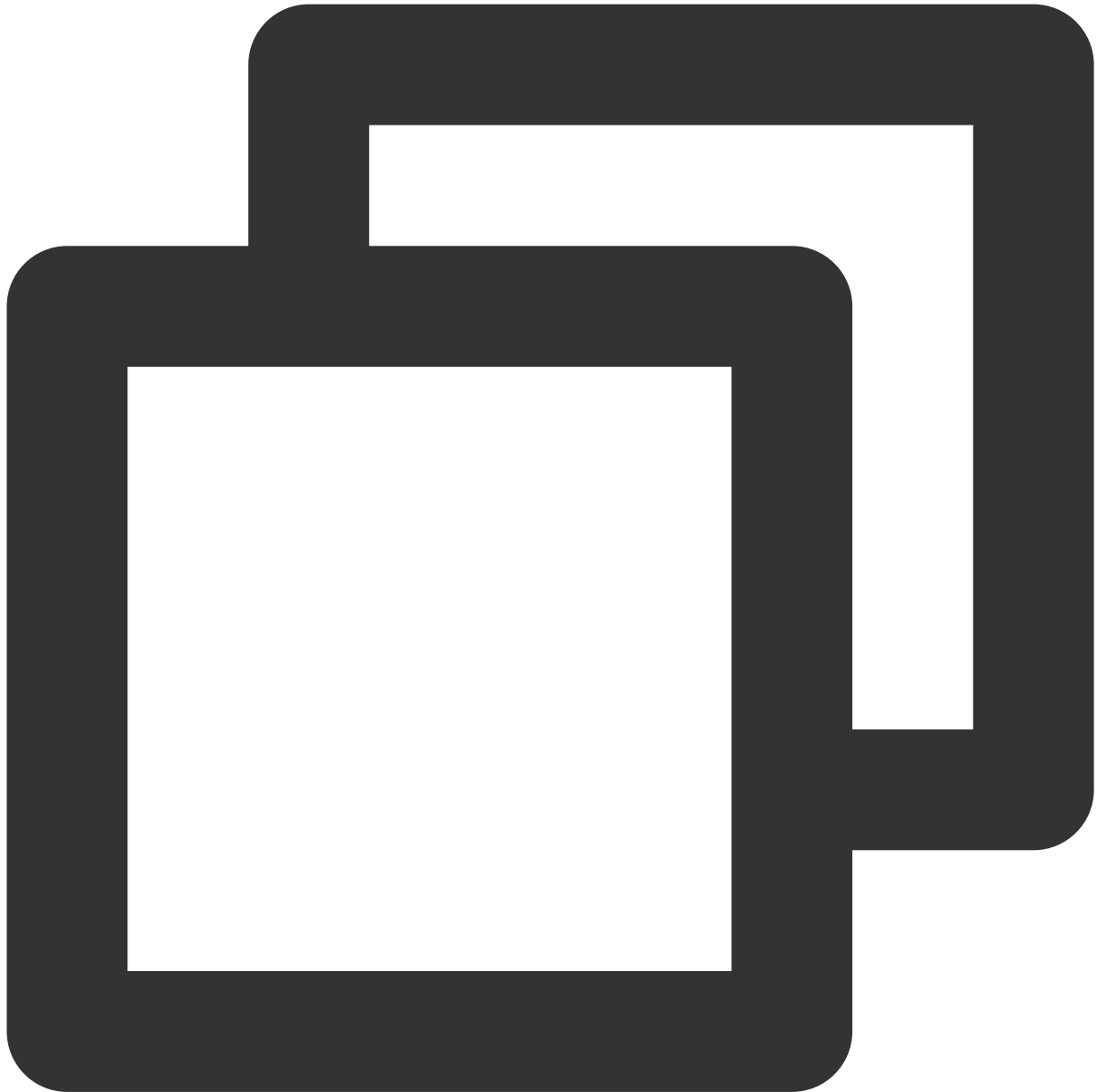
```
void onUserReject(String userId);
```

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.



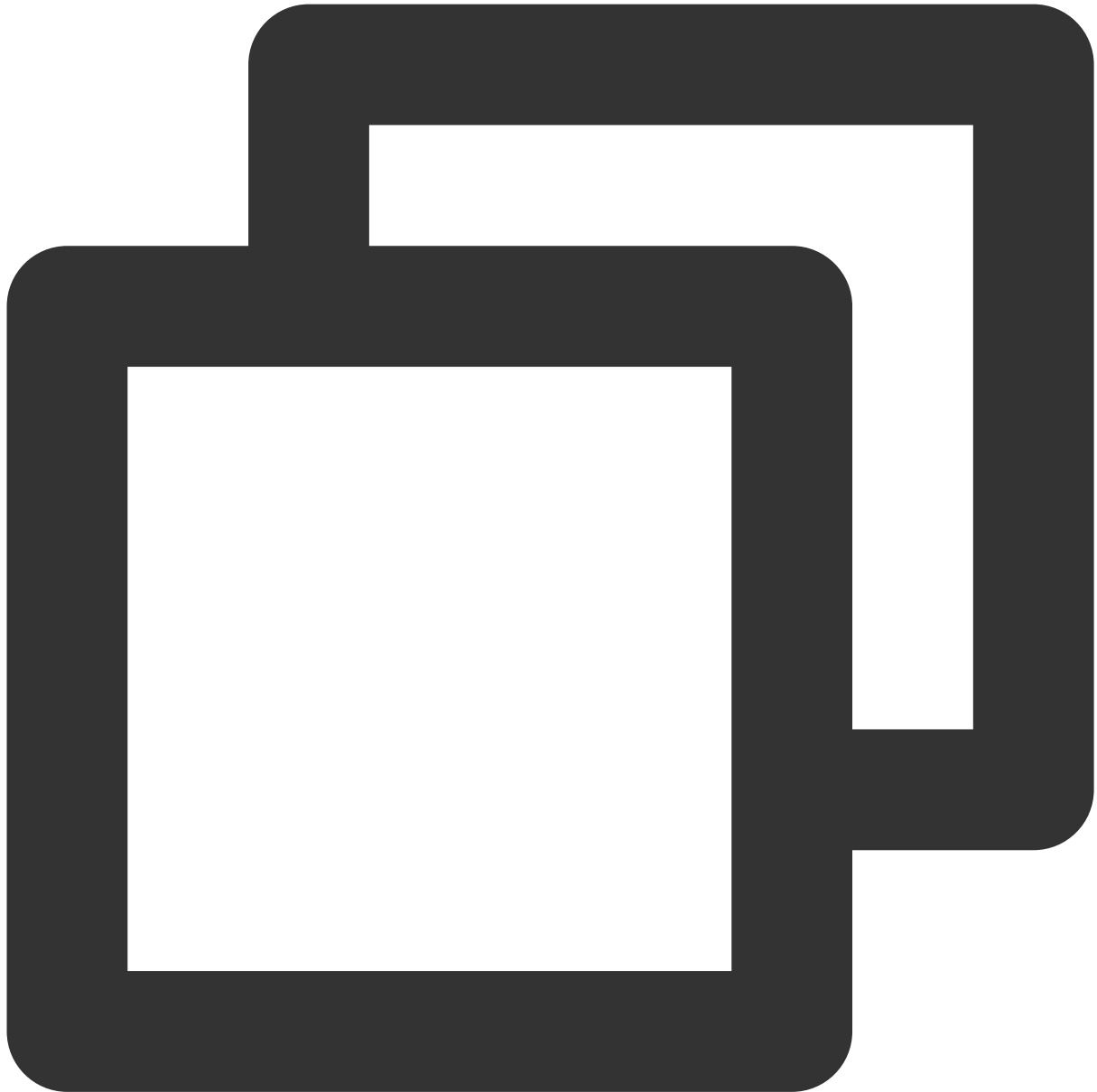
```
void onUserNoResponse(String userId);
```

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.



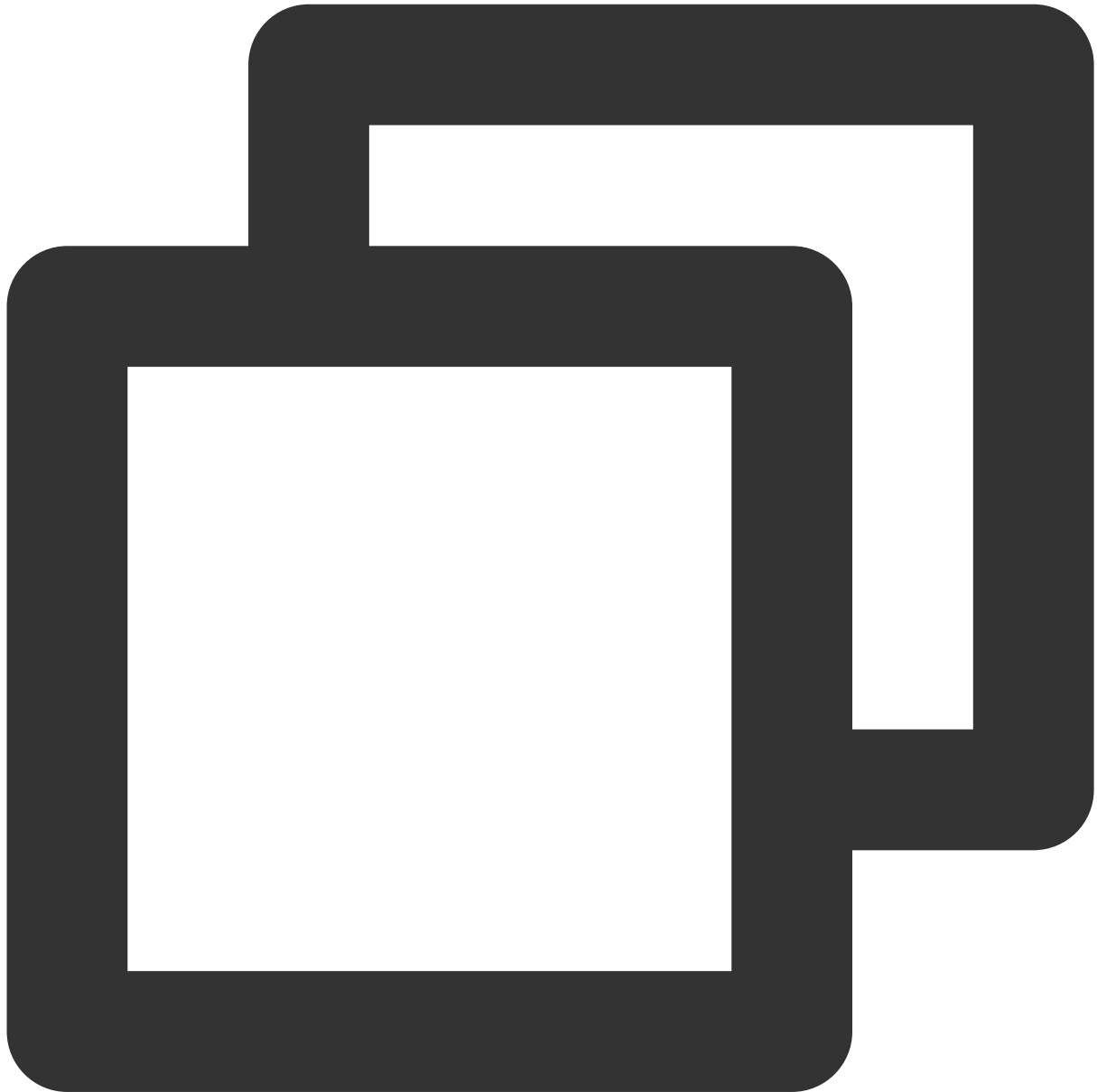
```
void onUserLineBusy(String userId);
```

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.



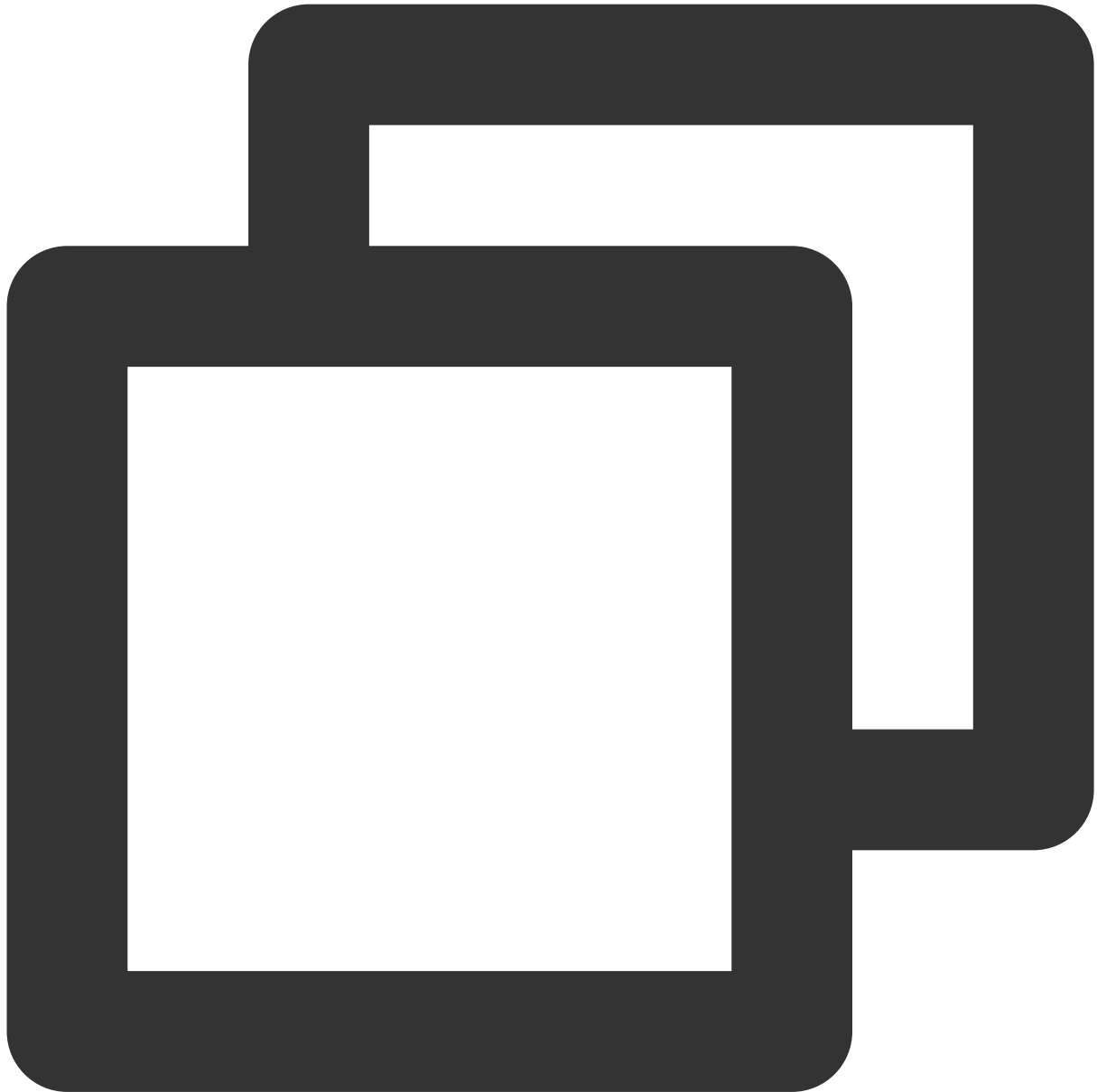
```
void onUserJoin(String userId);
```

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.



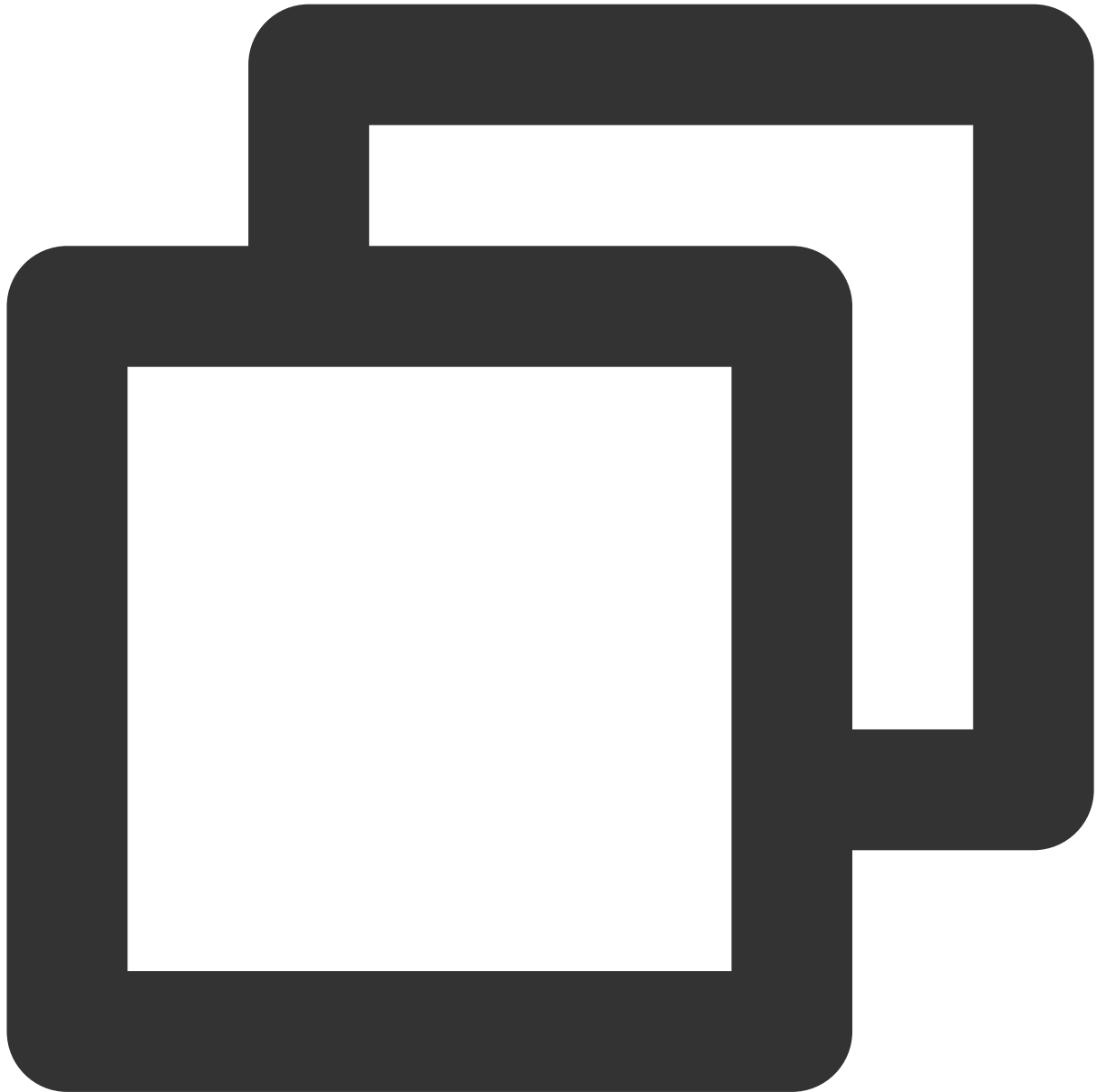
```
void onUserLeave(String userId);
```

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);
```

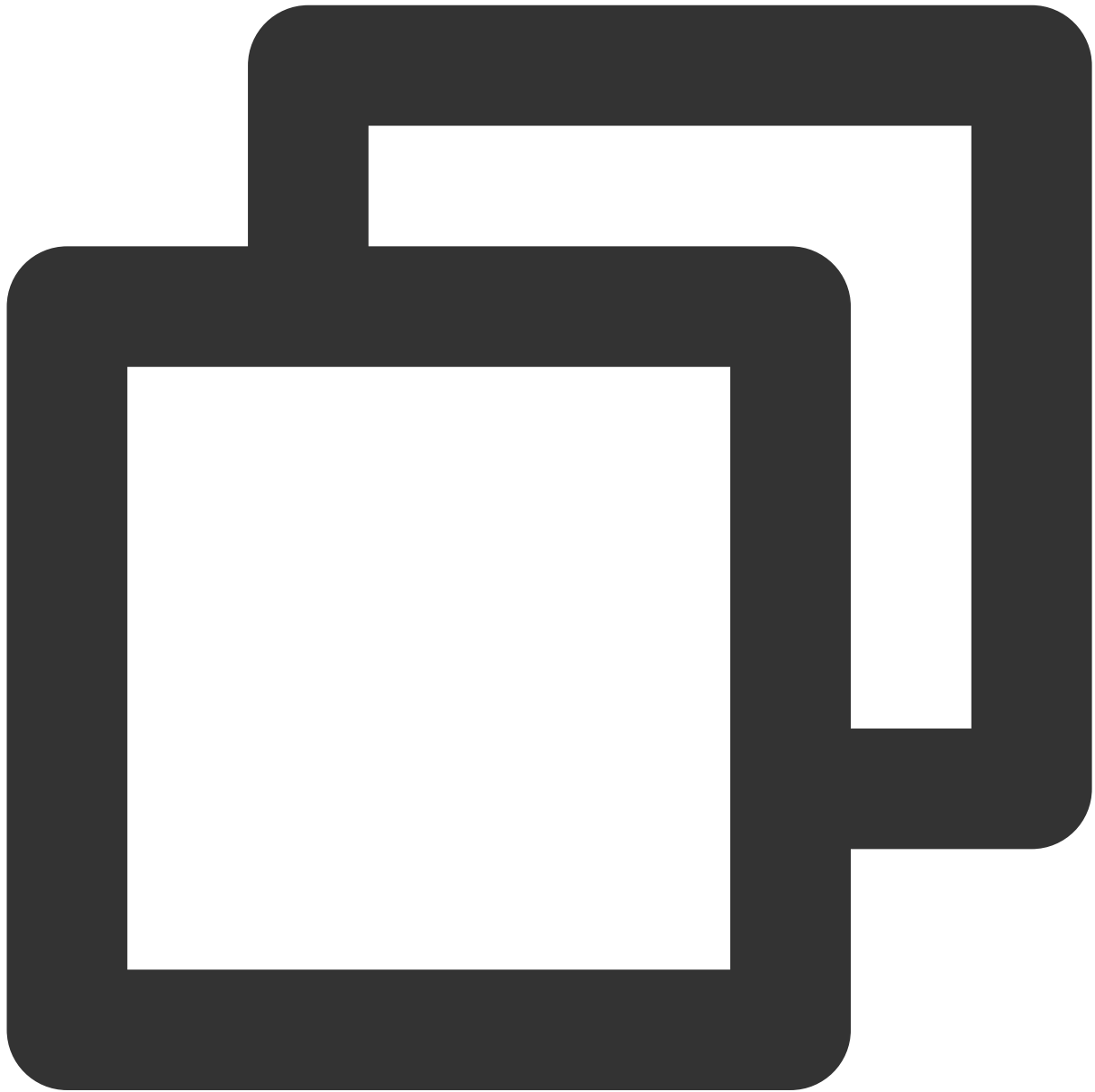
The parameters are described below:

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);
```

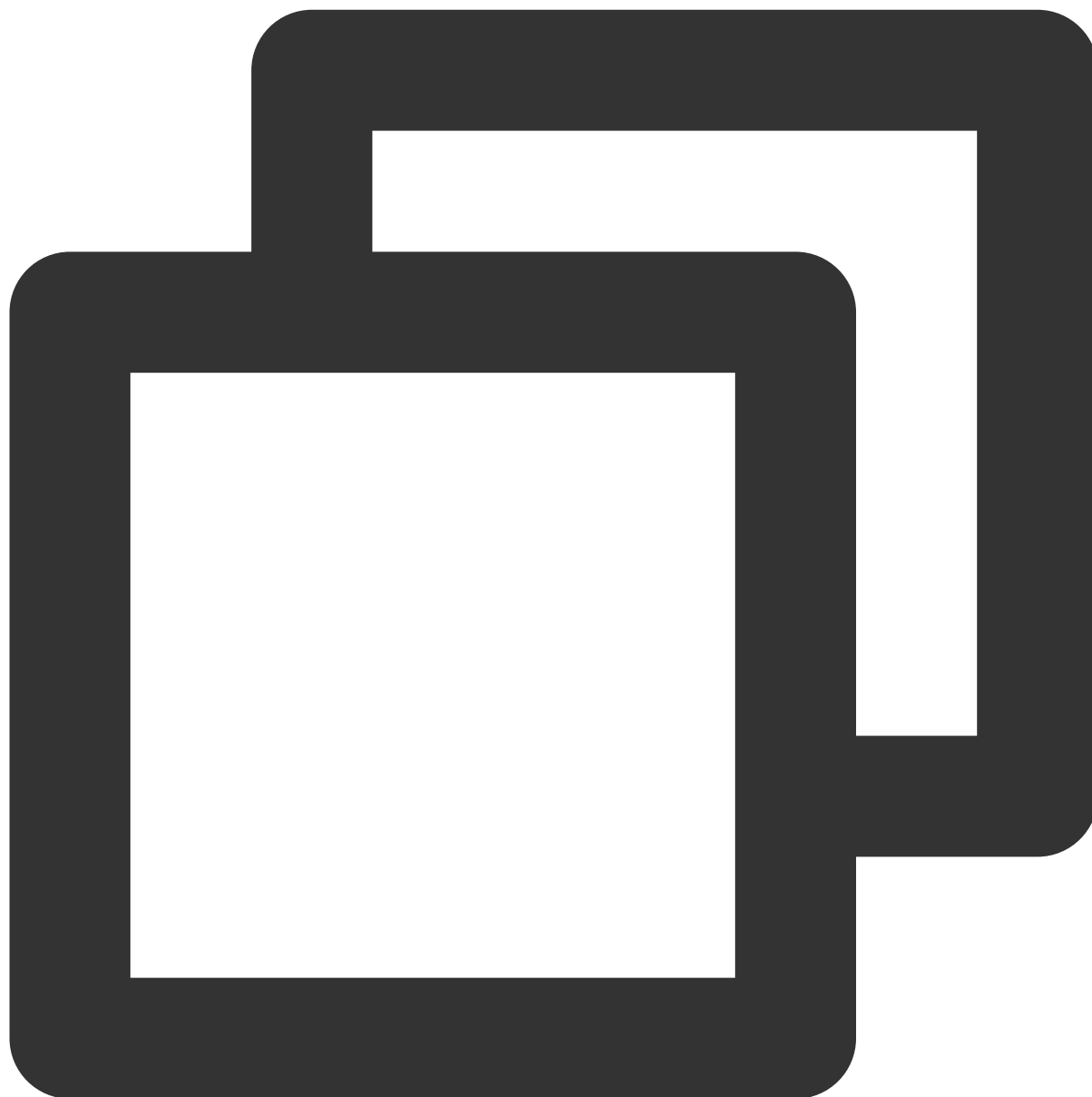
The parameters are described below:

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);
```

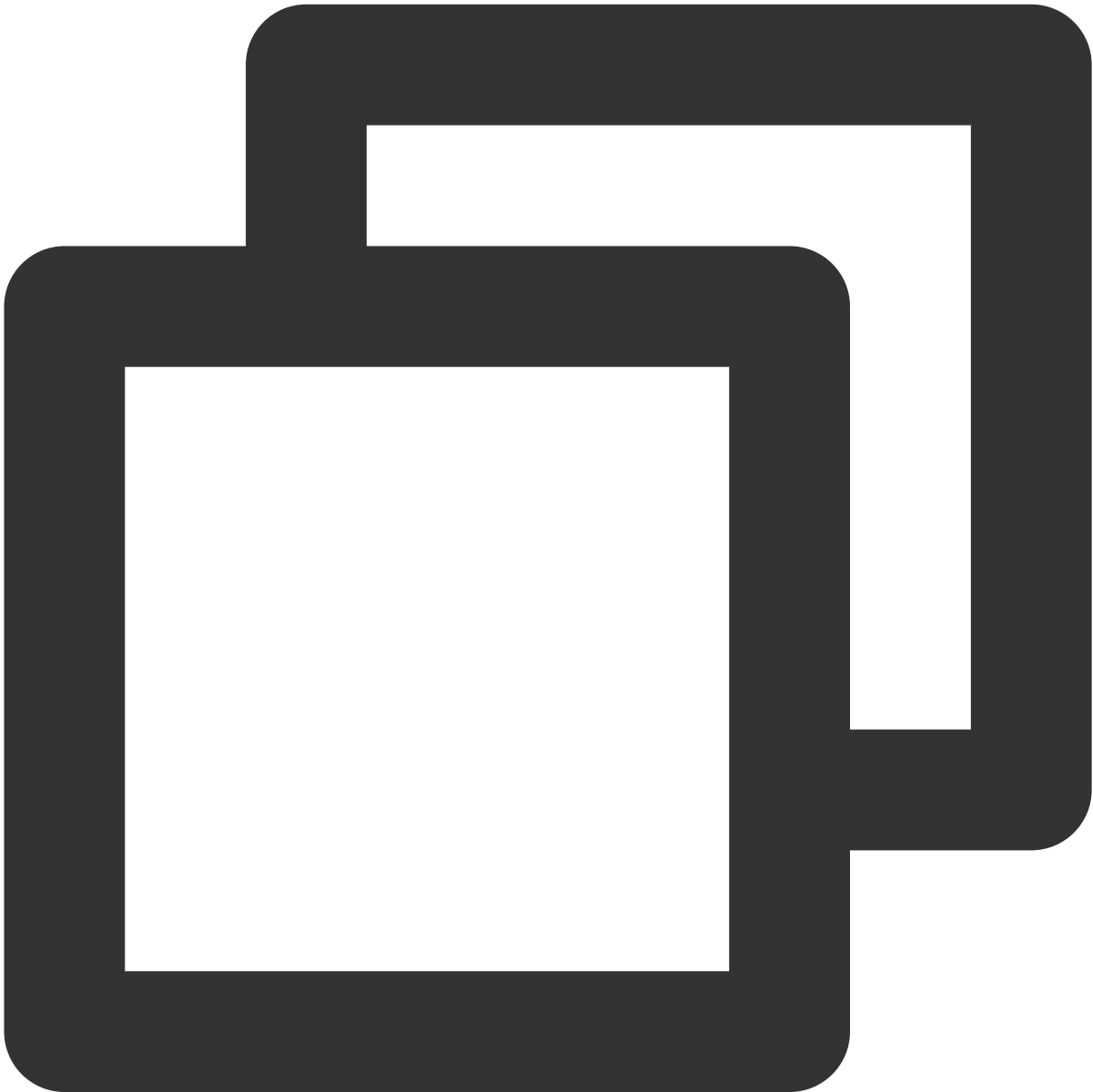
The parameters are described below:

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

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

onUserNetworkQualityChanged

The network quality of all users.



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

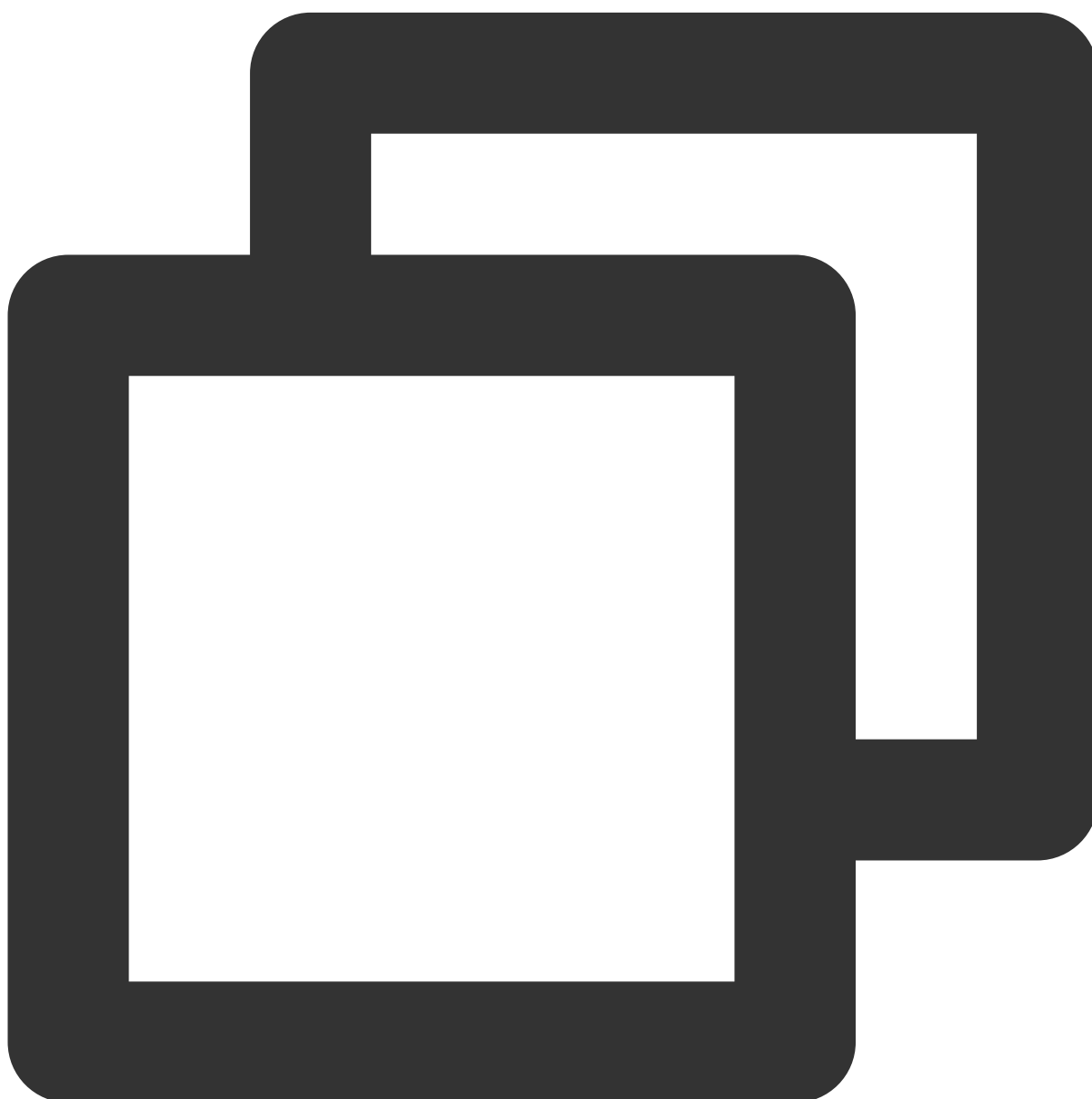
The parameters are described below:

--	--	--

Parameter	Type	Description
networkQualityList	List	The current network conditions for all users ( <code>userId</code> ).

## onKickedOffline

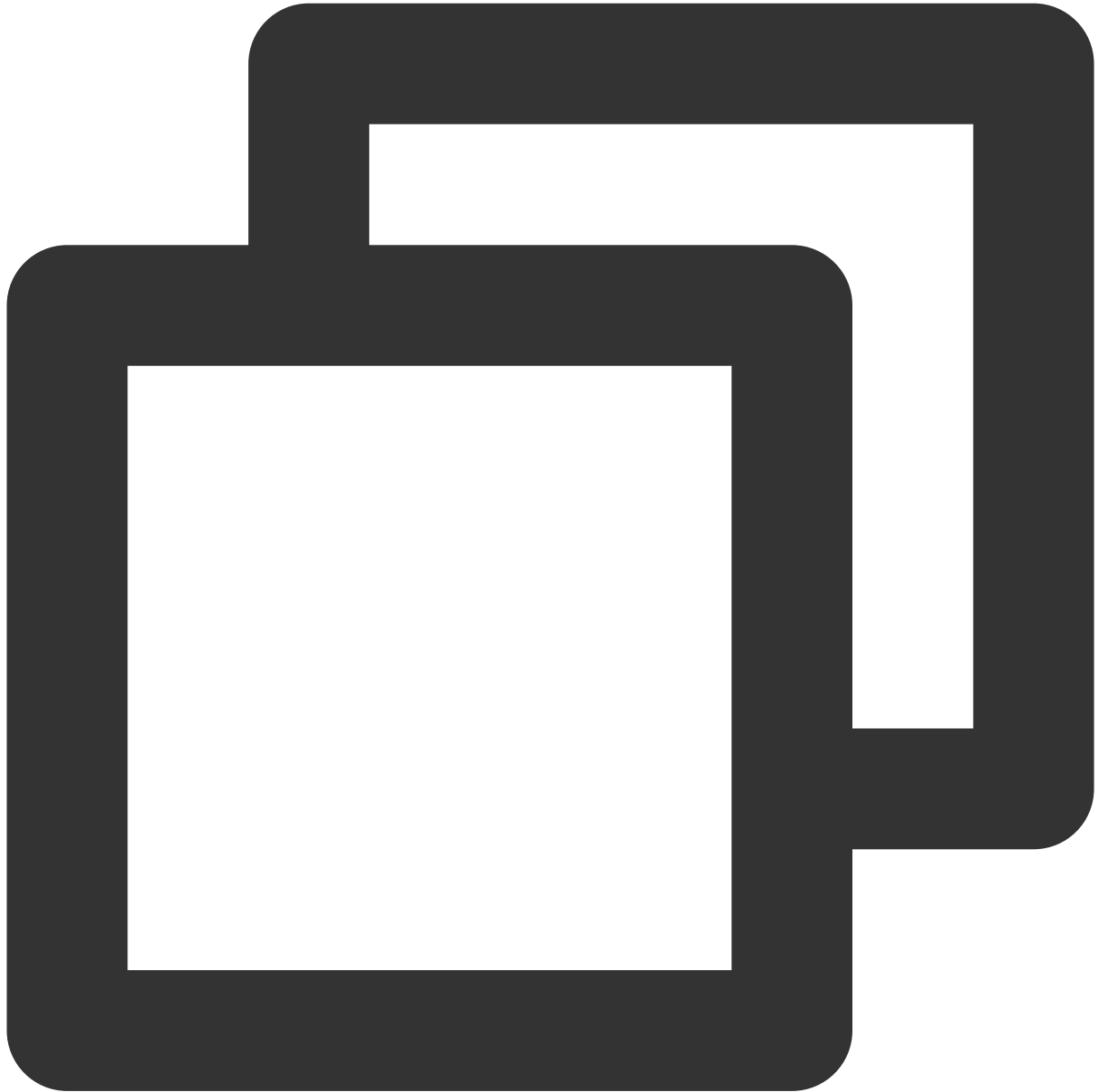
The current user was kicked offline : At this time, you can prompt the user with a UI message and then invoke `init` again.



```
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
<a href="#">CallParams</a>	An additional parameter.
<a href="#">OfflinePushInfo</a>	Offline push vendor configuration information.

### TUICommonDefine

Type	Description
<a href="#">RoomId</a>	Room ID for audio and video in a call.
<a href="#">NetworkQualityInfo</a>	Network quality information
<a href="#">VideoRenderParams</a>	Video render parameters
<a href="#">VideoEncoderParams</a>	Video encoding parameters

## Enum definition

### TUICallDefine

Type	Description
<a href="#">MediaType</a>	Media type in a call
<a href="#">Role</a>	Roles of individuals in a call.
<a href="#">Status</a>	The call status
<a href="#">Scene</a>	The call scene
<a href="#">IOSOfflinePushType</a>	iOS offline push type

### TUICommonDefine

Type	Description

<a href="#">AudioPlaybackDevice</a>	Audio route
<a href="#">Camera</a>	Camera type
<a href="#">NetworkQuality</a>	Network quality
<a href="#">FillMode</a>	Video image fill mode
<a href="#">Rotation</a>	Video image rotation direction
<a href="#">ResolutionMode</a>	Video aspect ratio mode
<a href="#">Resolution</a>	Video resolution

## CallParams

Call params

Value	Type	Description
offlinePushInfo	<a href="#">OfflinePushInfo</a>	Offline push vendor configuration information.
timeout	int	Call timeout period, default: 30s, unit: seconds.
userData	String	An additional parameter. Callback when the callee receives <a href="#">onCallReceived</a>

## 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 <code>sound = IOS_OFFLINE_PUSH_NO_SOUND</code> , there will be no sound played when the message is received. When <code>sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND</code> , 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 <code>iOSSound</code> .
<code>androidSound</code>	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: <a href="#">Xiaomi custom ringtones</a> . In addition, for Google phones, in order to set sound prompts for FCM push on Android 8.0 and above systems, you must call <code>setAndroidFCMChannelID</code> to set the <code>channelID</code> for it to take effect.
<code>androidOPPOChannelID</code>	String	Set the channel ID for OPPO phones with Android 8.0 and above systems.
<code>androidVIVOClassification</code>	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 <code>setAndroidVIVOCategory</code> to set the message category). 0: Operational messages, 1: System messages. The default value is 1.
<code>androidXiaoMiChannelID</code>	String	Set the channel ID for Xiaomi phones with Android 8.0 and above systems.
<code>androidFCMChannelID</code>	String	Set the channel ID for google phones with Android 8.0 and above systems.
<code>androidHuaWeiCategory</code>	String	Classification of Huawei push messages, please refer to: <a href="#">Huawei message classification standard</a> .
<code>isDisablePush</code>	boolean	Whether to turn off push notifications (default is on).
<code>iOSPushType</code>	<a href="#">IOSOfflinePushType</a>	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
<code>intRoomId</code>	int	Numeric room ID. <b>range</b> : 1 - 2147483647(2^31-1)
<code>strRoomId</code>	String	String room ID. <b>range</b> : Limited to 64 bytes in length. The supported character set range is as follows (a total of 89 Lowercase and uppercase English letters. (a-zA-Z) ; Number (0-9) ; 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!

## NetworkQualityInfo

User network quality information

Value	Type	Description
<code>userId</code>	String	user ID
<code>quality</code>	<a href="#">NetworkQuality</a>	network quality

## VideoRenderParams

Video render parameters

Value	Type	Description
<code>fillMode</code>	<a href="#">VideoRenderParams.FillMode</a>	Video image fill mode
<code>rotation</code>	<a href="#">VideoRenderParams.Rotation</a>	Video image rotation direction

## VideoEncoderParams

Video encoding parameters

Value	Type	Description
<code>resolution</code>	<a href="#">VideoEncoderParams.Resolution</a>	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 <a href="#">open the free experience</a> in the console or <a href="#">purchase the official package</a> .
ERROR_PACKAGE_NOT_SUPPORTED	-1002	The package you purchased does not support this ability. You can refer to console to purchase <a href="#">Upgrade package</a> .
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 <a href="#">init</a> 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 <a href="#">Generating UserSig</a> .
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 reenale 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 reenale 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 : 2024-07-30 11:21:52

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

### 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

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

<a href="#">setVideoEncoderParams</a>	Set the encoding parameters of video encoder.
<a href="#">getTRTCCloudInstance</a>	Advanced features.
<a href="#">setBeautyLevel</a>	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
<a href="#">onError</a>	An error occurred during the call.
<a href="#">onCallReceived</a>	A call was received.
<a href="#">onCallCancelled</a>	The call was canceled.
<a href="#">onCallBegin</a>	The call was connected.
<a href="#">onCallEnd</a>	The call ended.
<a href="#">onCallMediaTypeChanged</a>	The call type changed.
<a href="#">onUserReject</a>	A user declined the call.
<a href="#">onUserNoResponse</a>	A user didn't respond.
<a href="#">onUserLineBusy</a>	A user was busy.
<a href="#">onUserJoin</a>	A user joined the call.
<a href="#">onUserLeave</a>	A user left the call.
<a href="#">onUserVideoAvailable</a>	Whether a user had a video stream.
<a href="#">onUserAudioAvailable</a>	Whether a user had an audio stream.
<a href="#">onUserVoiceVolumeChanged</a>	The volume levels of all users.
<a href="#">onUserNetworkQualityChanged</a>	The network quality of all users.
<a href="#">onKickedOffline</a>	The current user was kicked offline.
<a href="#">onUserSigExpired</a>	The user sig is expired.

## Definitions of Key Types

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

# TUICallKit

Last updated : 2024-05-27 17:18:36

## 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
<a href="#">createInstance</a>	Creates a <code>TUICallKit</code> instance (singleton mode).
<a href="#">setSelfInfo</a>	Sets the alias and profile photo.
<a href="#">call</a>	Makes a one-to-one call.
<a href="#">call</a>	Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">groupCall</a>	Makes a group call.
<a href="#">groupCall</a>	Makes a group call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">joinInGroupCall</a>	Joins a group call.
<a href="#">setCallingBell</a>	Sets the ringtone.
<a href="#">enableMuteMode</a>	Sets whether to turn on the mute mode.
<a href="#">enableFloatWindow</a>	Sets whether to enable floating windows.
<a href="#">enableIncomingBanner</a>	Sets whether to enable the incoming banner.

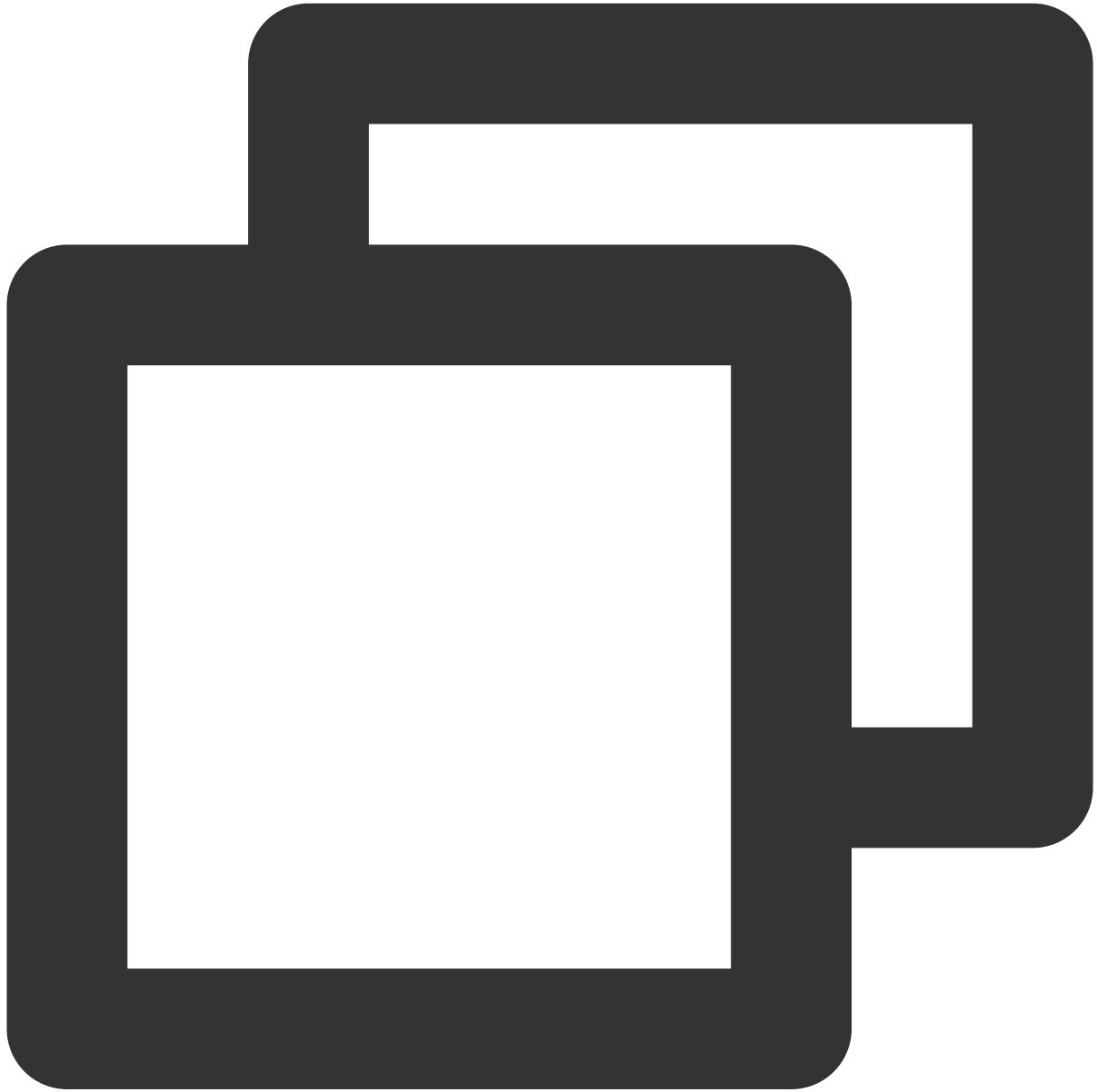
## Details

### createInstance

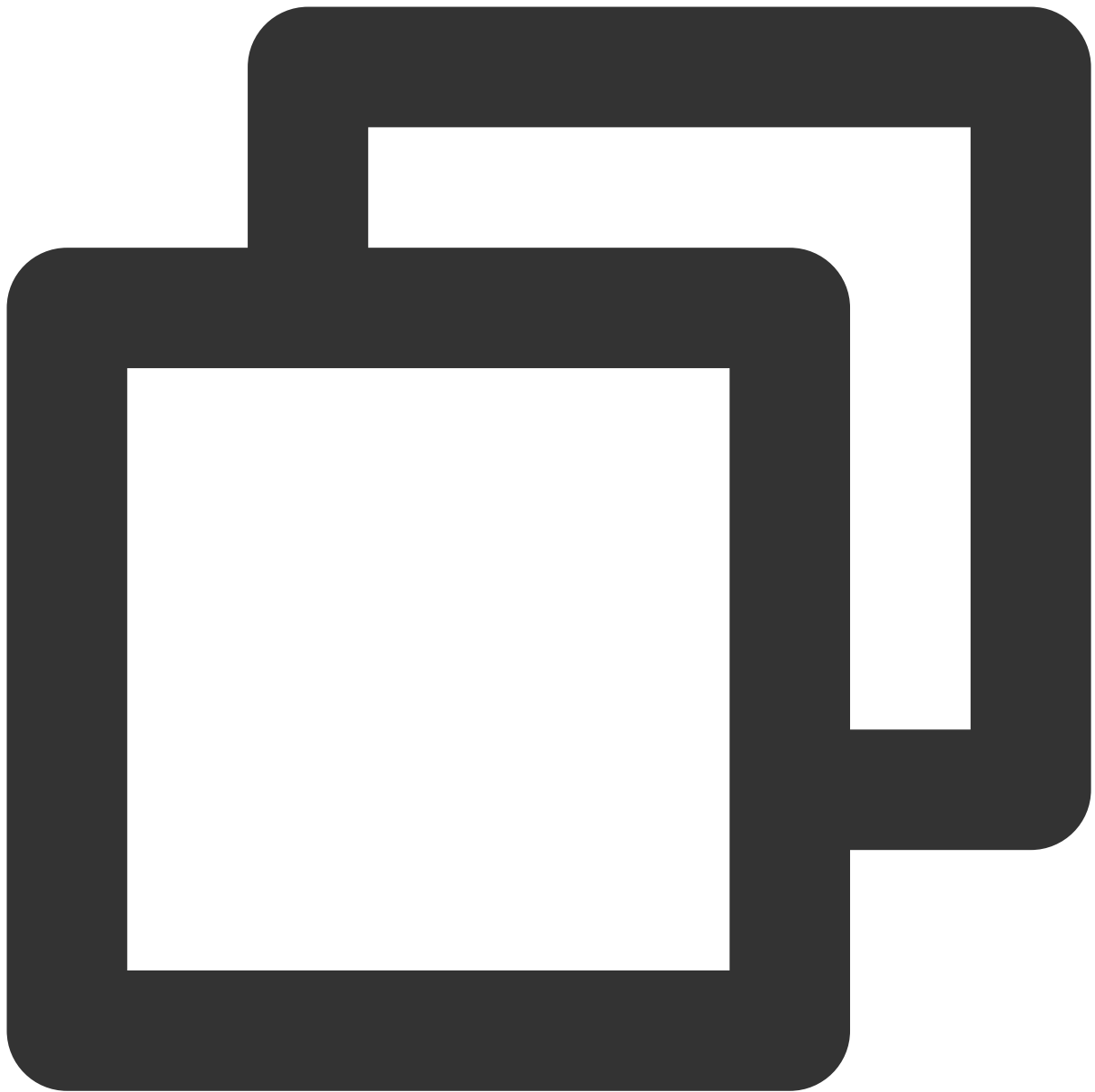
This API is used to create a `TUICallKit` singleton.

Objective-C

Swift



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



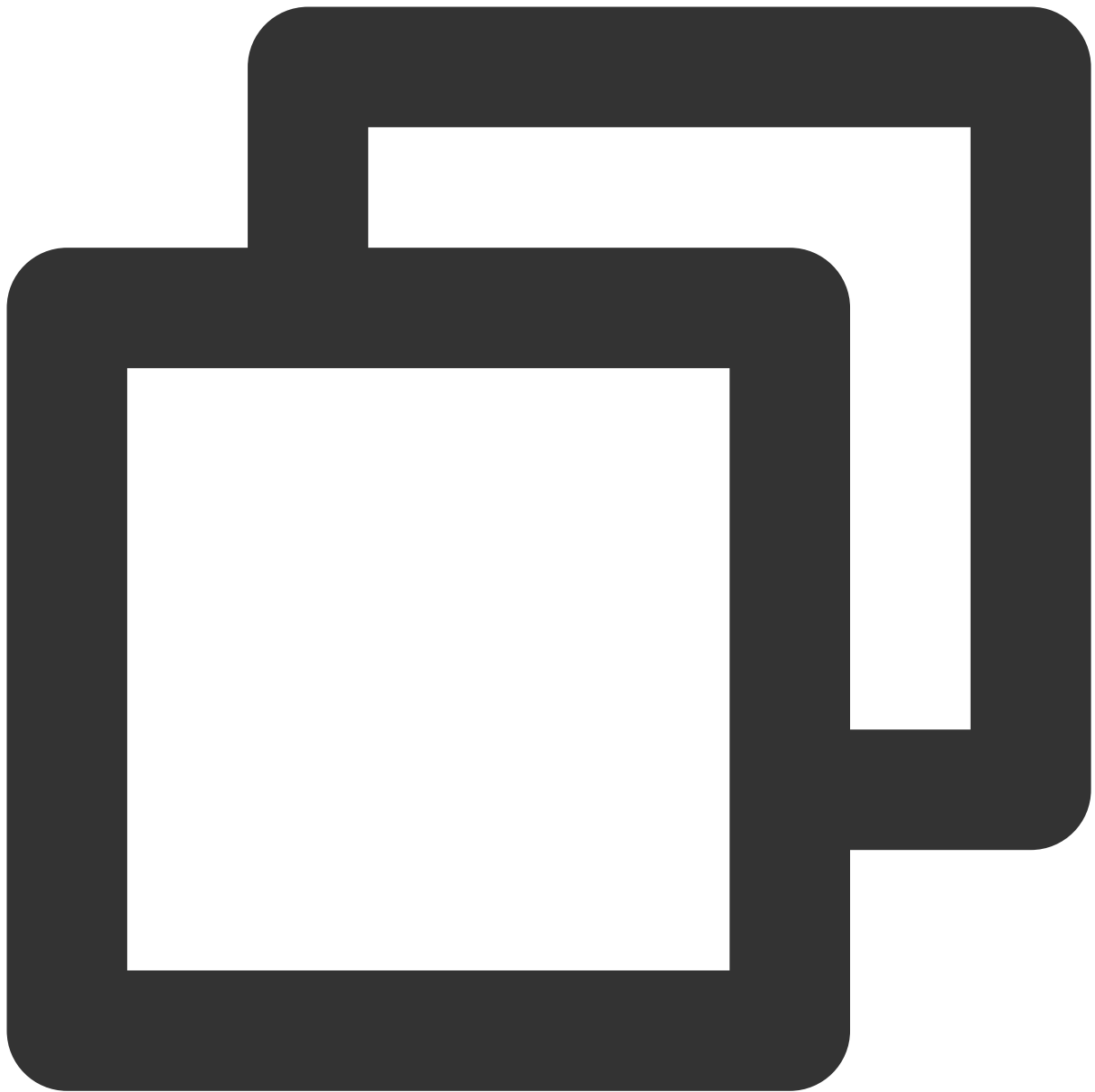
```
public static func createInstance() -> TUICallKit
```

### 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.

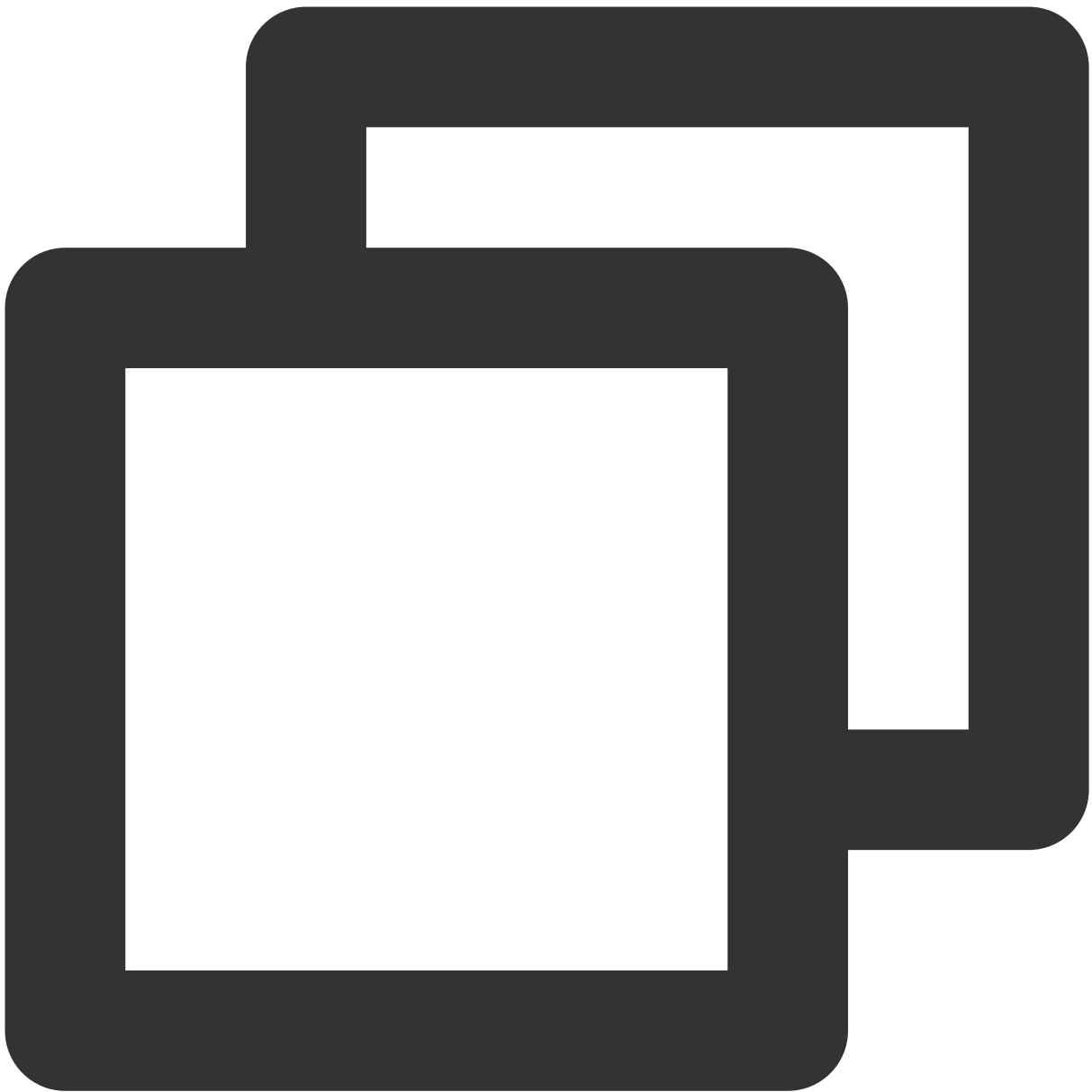
Objective-C

Swift



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





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

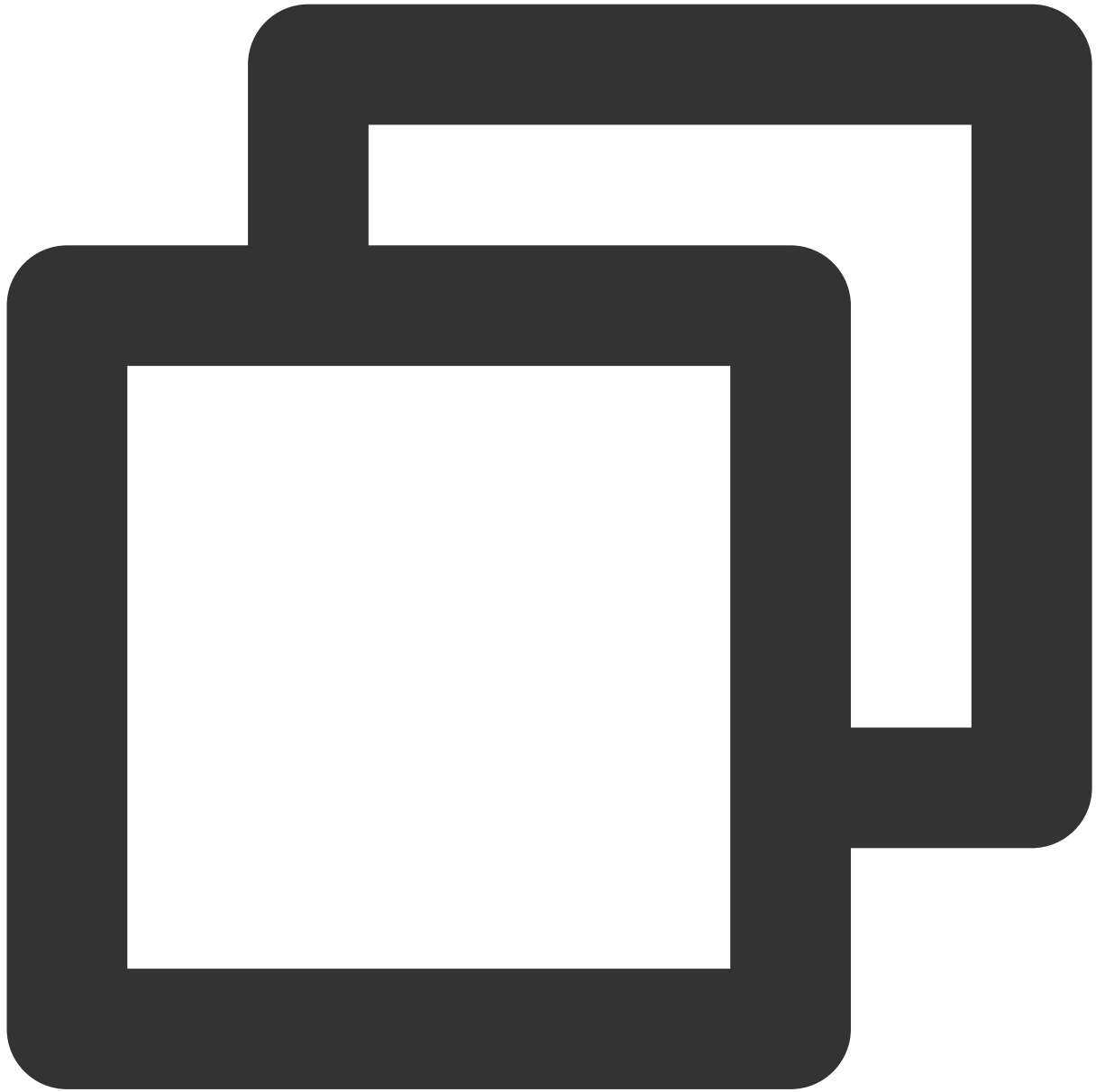
Parameter	Type	Description
nickName	NSString	The alias.
avatar	NSString	The profile photo.

call

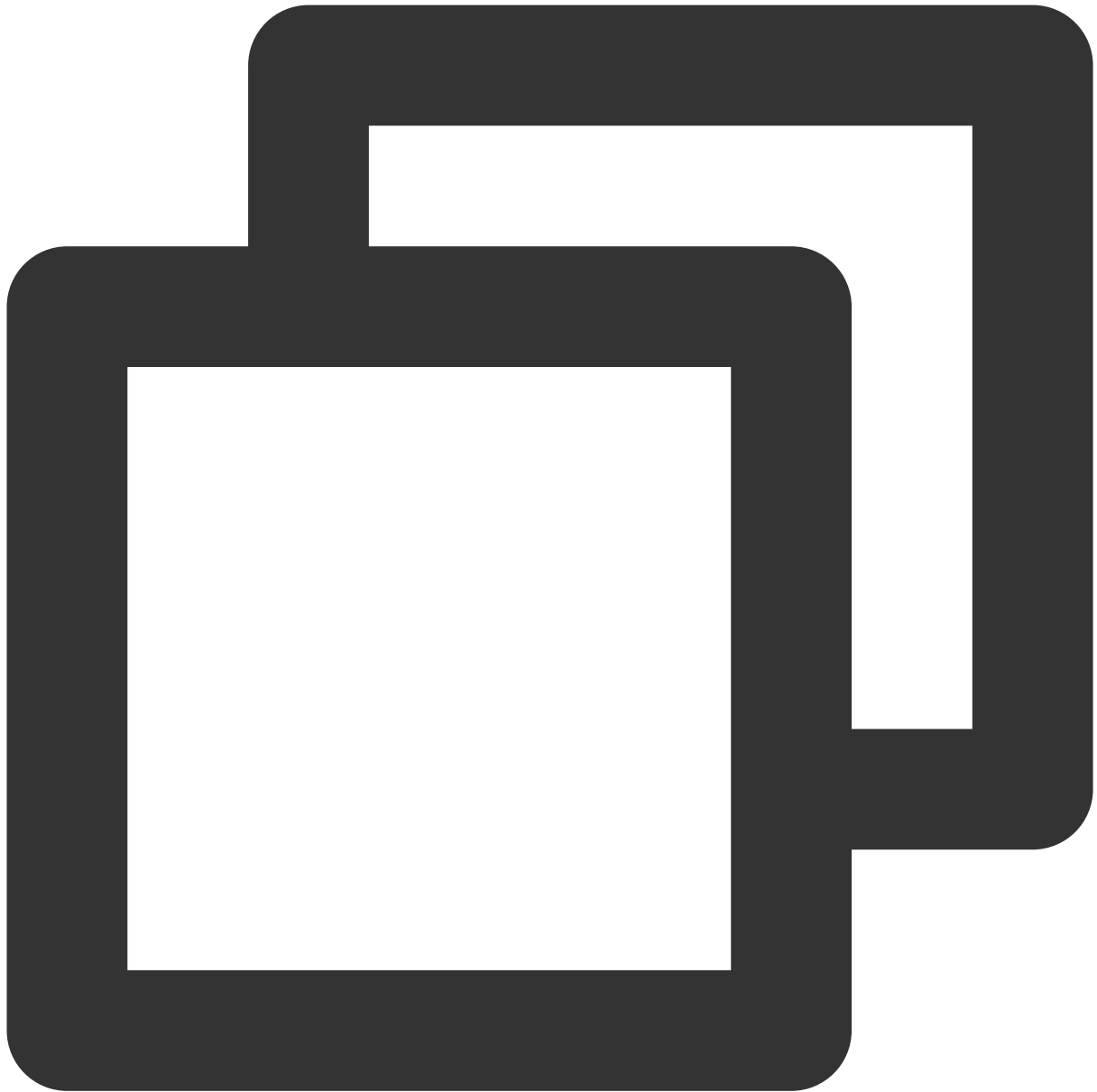
This API is used to make a (one-to-one) call.

Objective-C

Swift



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



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

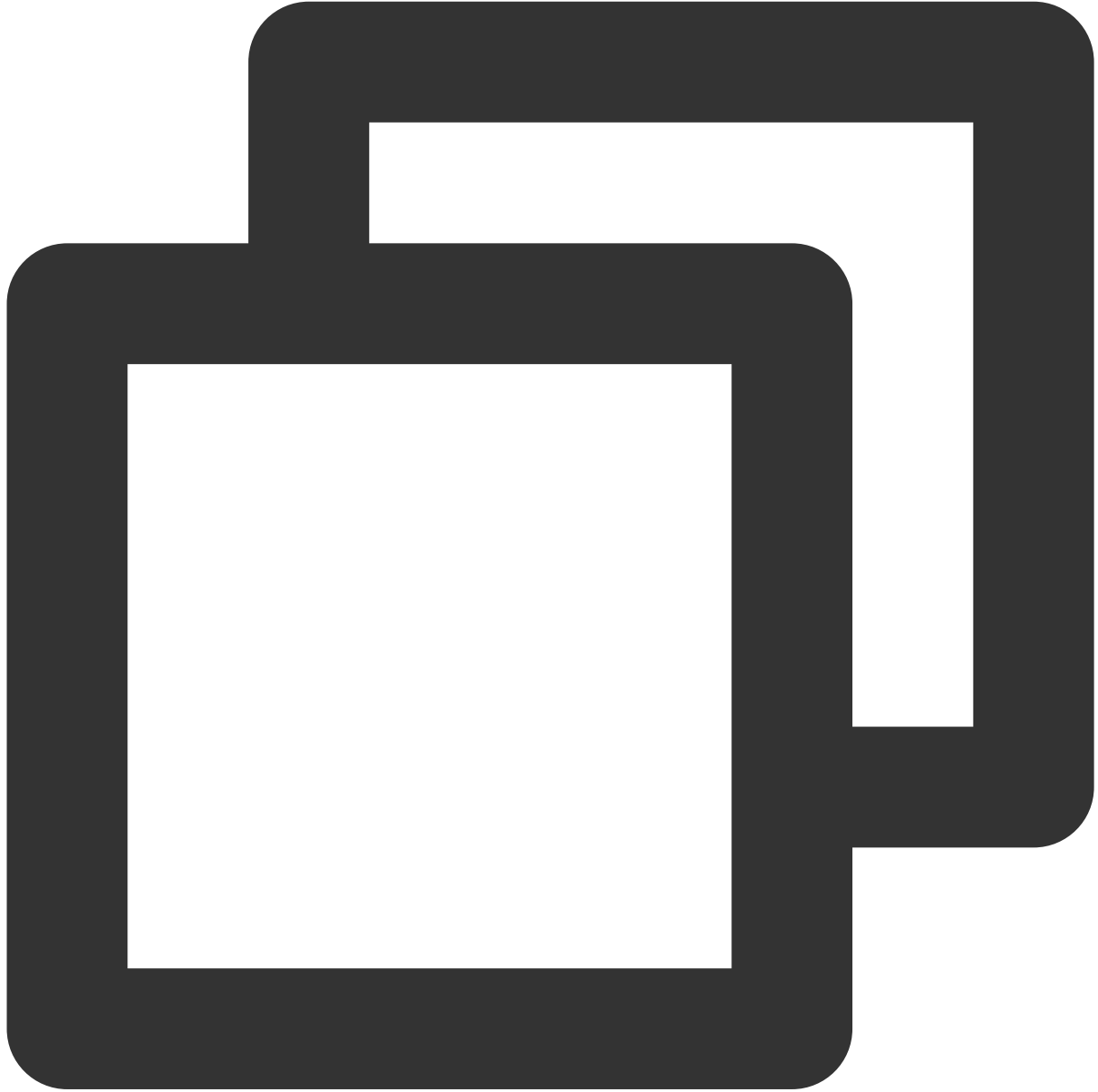
Parameter	Type	Description
userId	NSString	The target user ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

## call

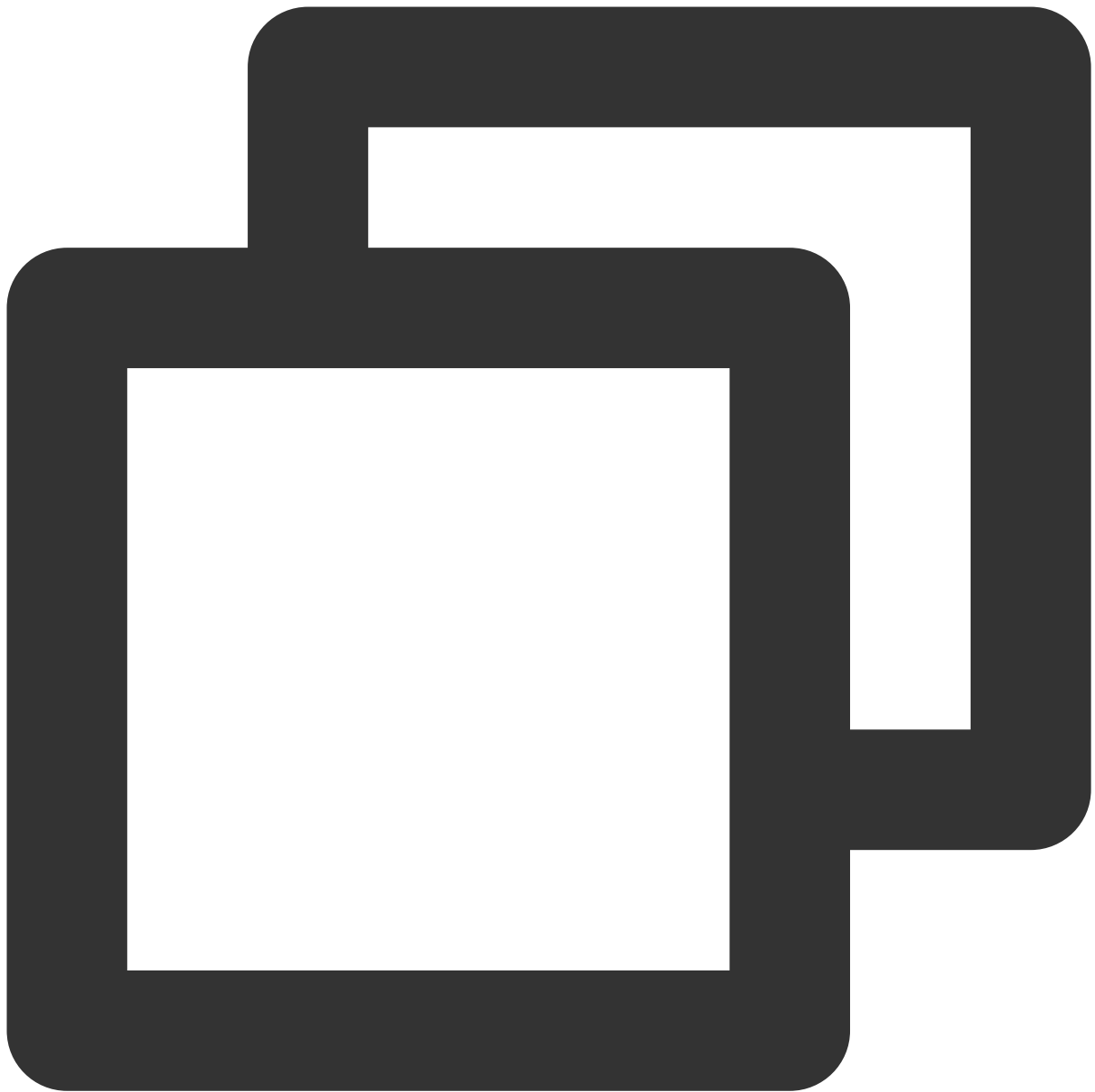
This API is used to make a (one-to-one) call, Support for custom room ID, call timeout, offline push content, etc.

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	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	Call extension parameters, such as roomId, call timeout, offline push info,etc

## groupCall

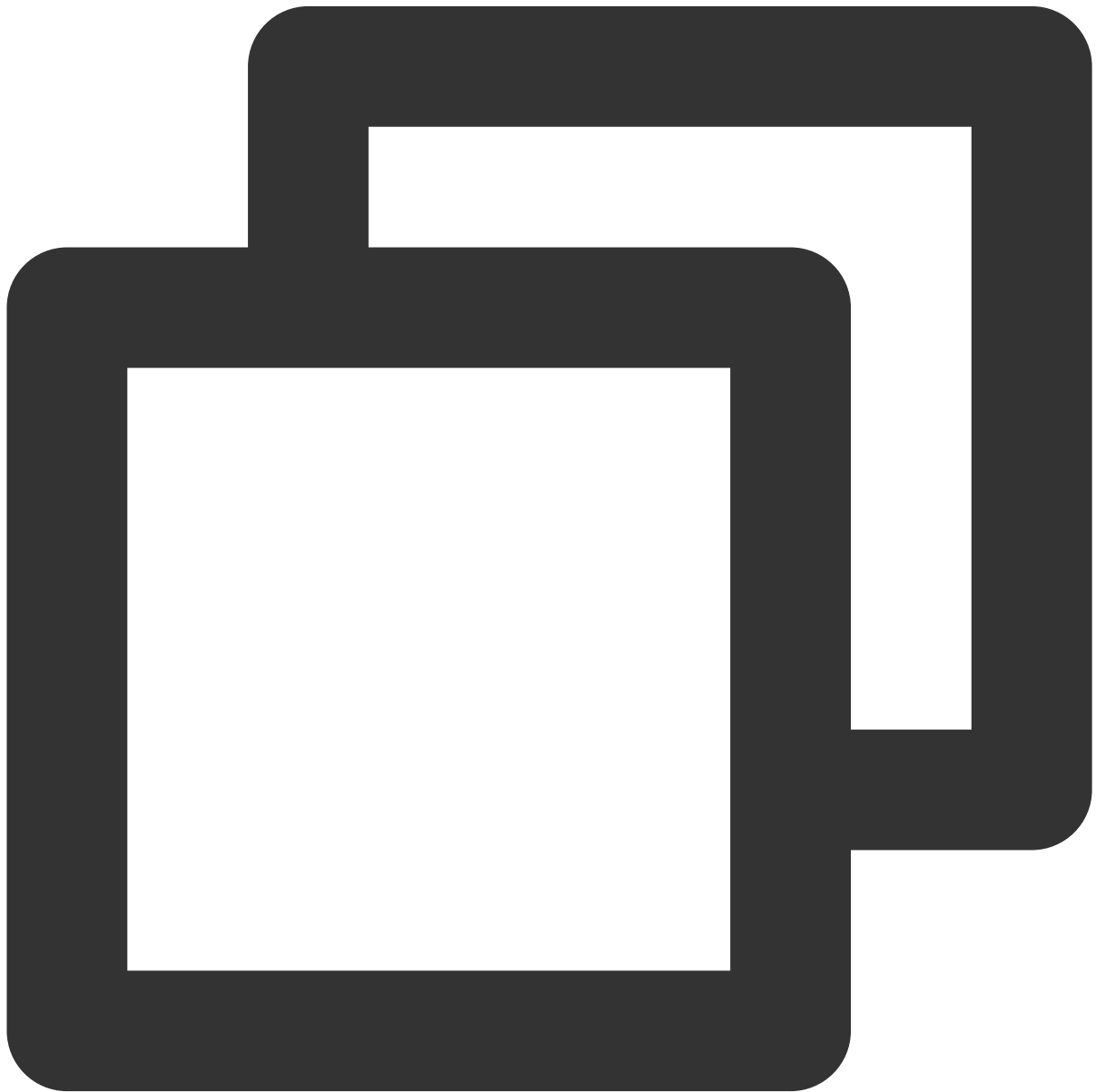
This API is used to make a group call.

### Note :

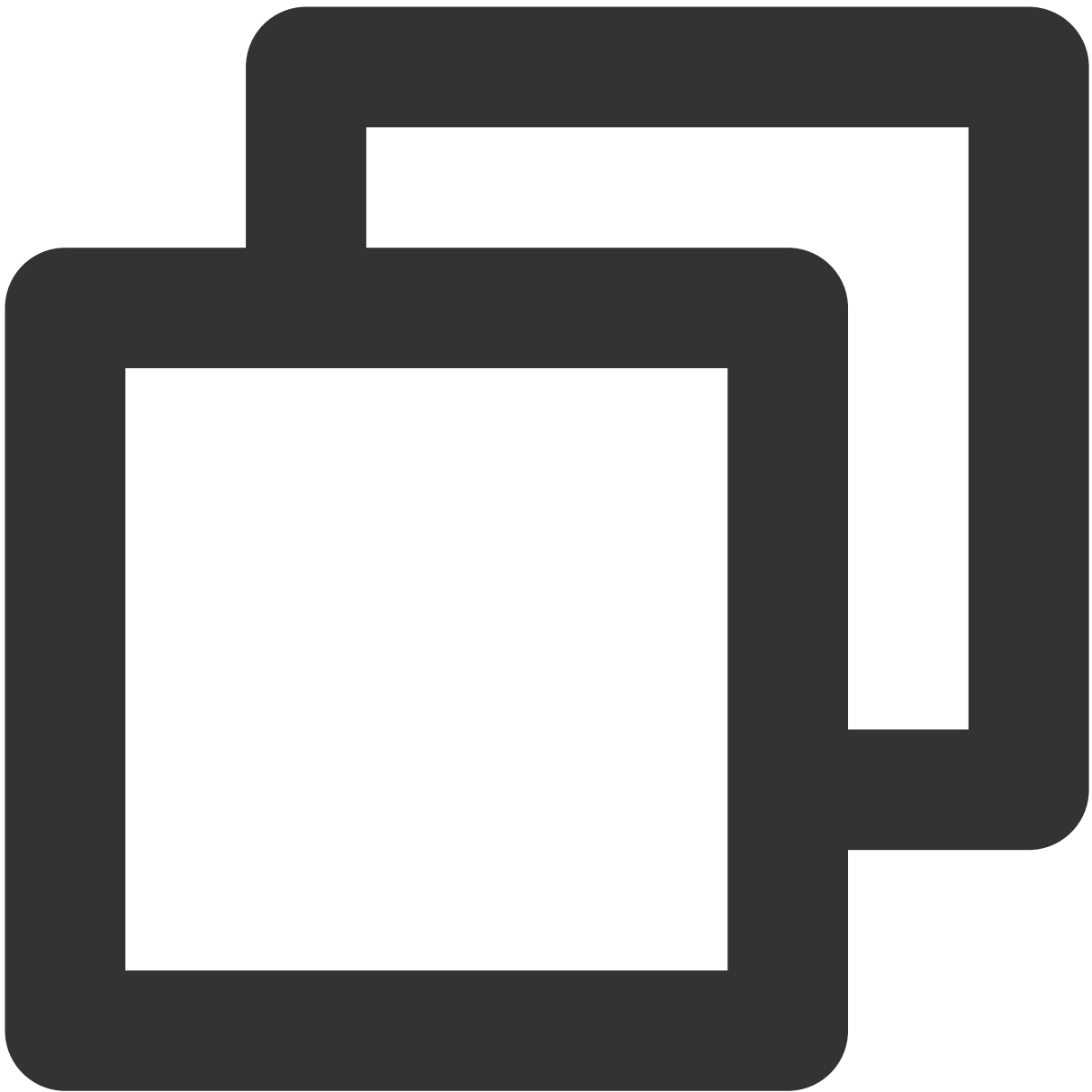
Before making a group call, you need to create an IM group first.

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	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

## groupCall

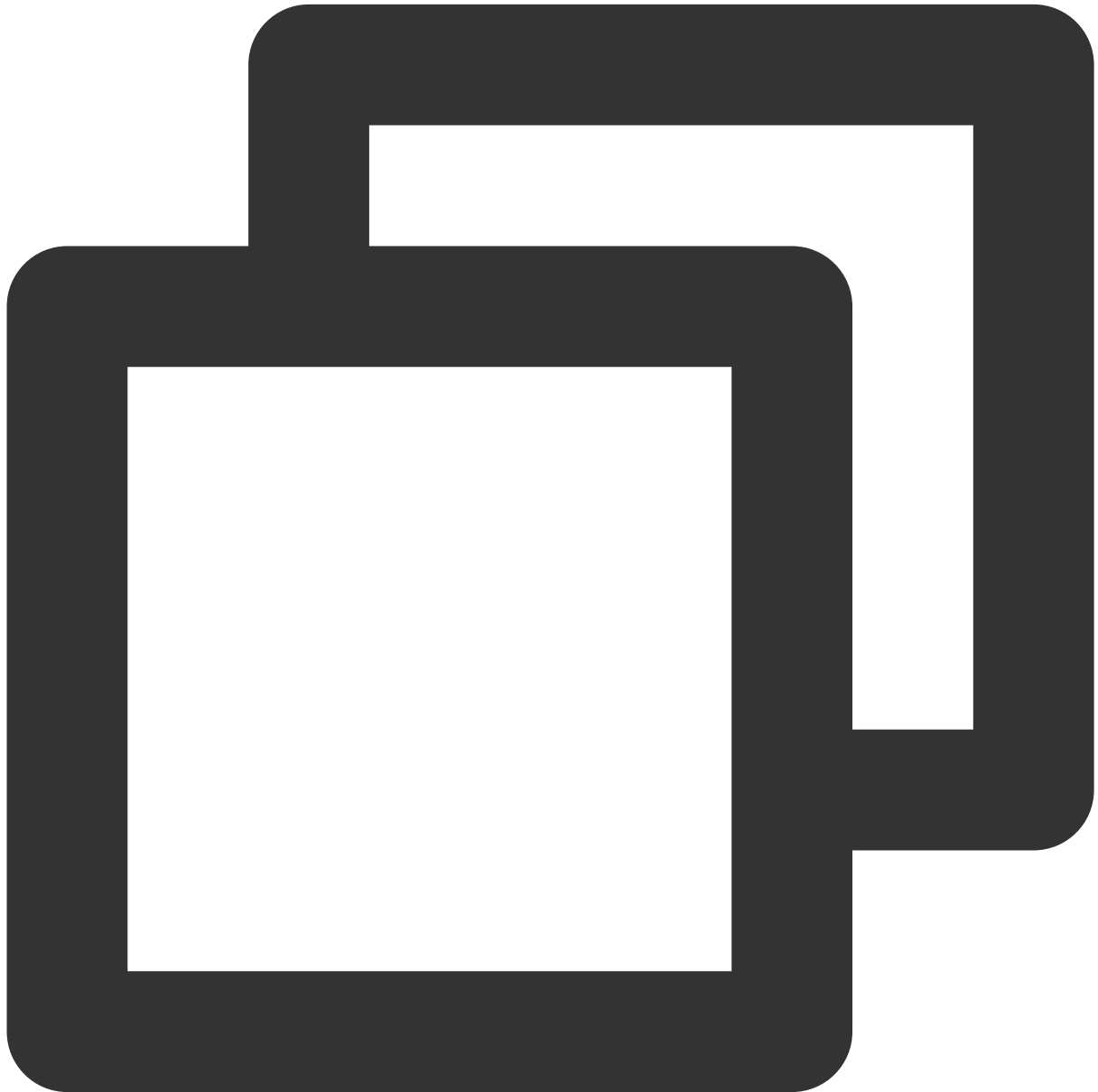
This API is used to make a group call, Support for custom room ID, call timeout, offline push content, etc.

### Note :

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

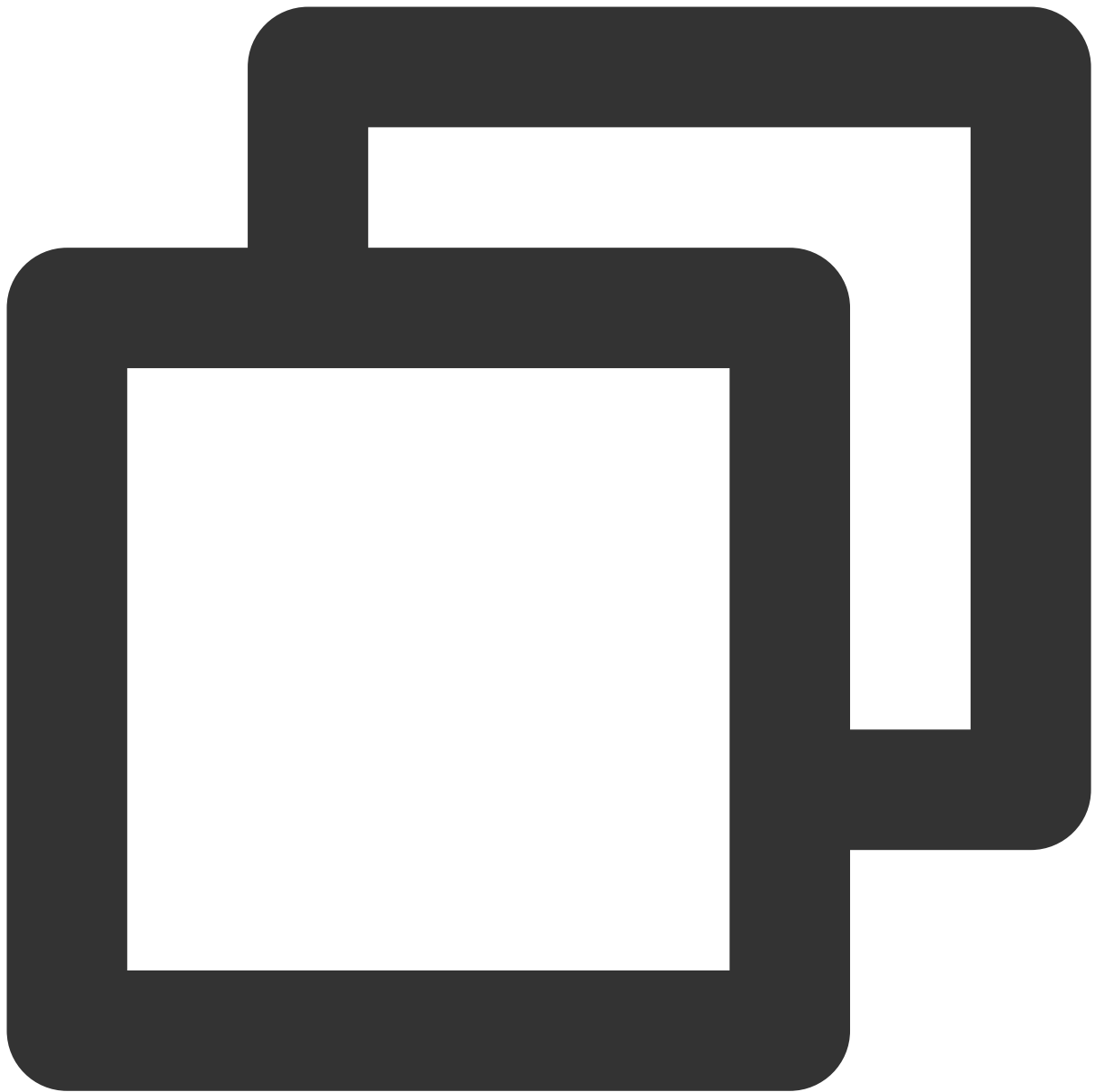
Objective-C

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	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	Call extension parameters, such as roomId, call timeout, offline

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

## joinInGroupCall

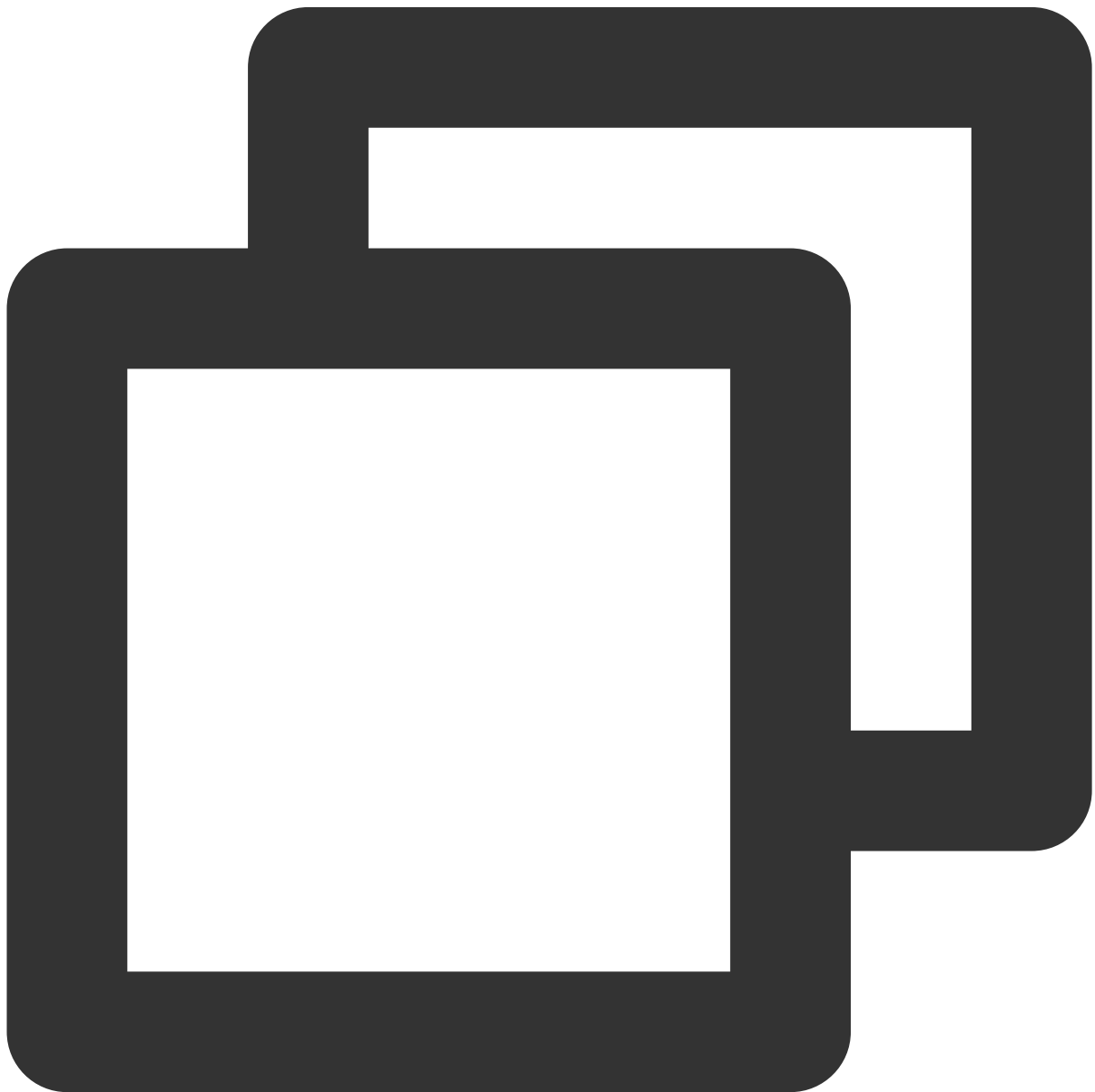
This API is used to join a group call.

### Note:

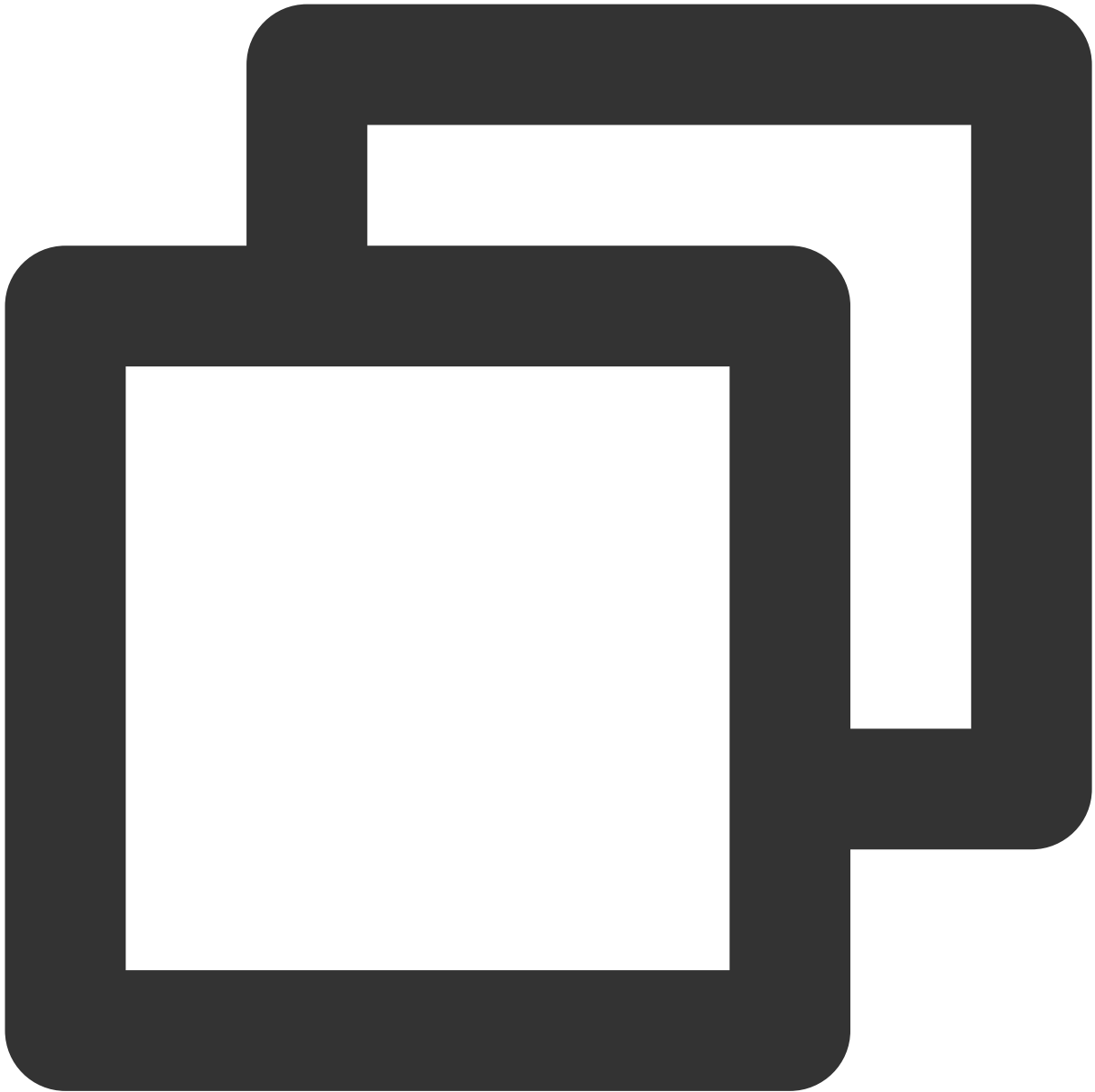
Before joining a group call, you need to create or join an IM group in advance, and there are already users in the group who are in the call.

Objective-C

Swift



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



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

Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.
groupId	NSString	The group ID.

callMediaType

TUICallMediaType

The call type, which can be video or audio.

## setCallingBell

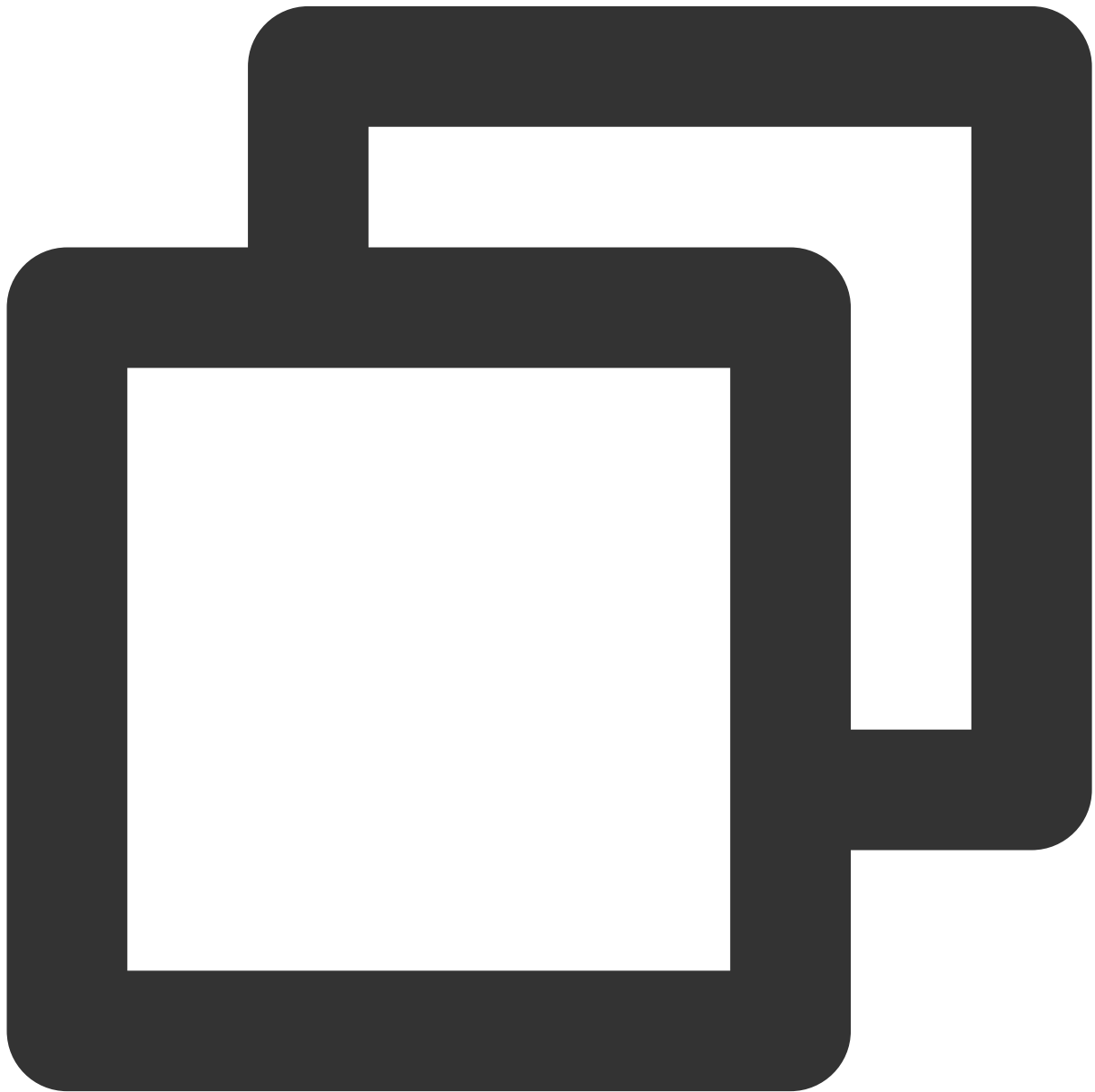
This API is used to set the ringtone. `filePath` must be an accessible local file URL.

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

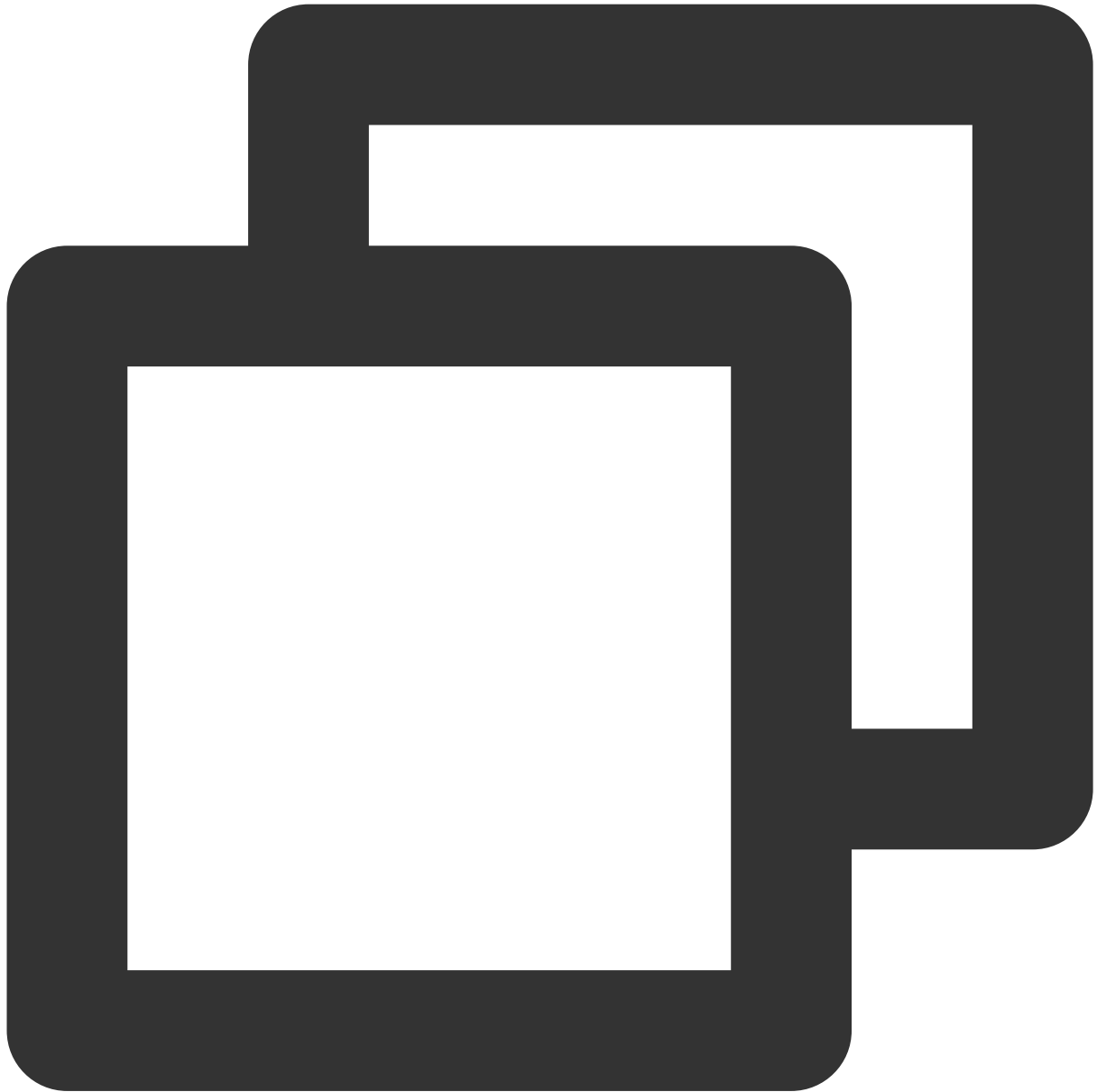
To reset the ringtone, pass in an empty string for `filePath`.

Objective-C

Swift



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



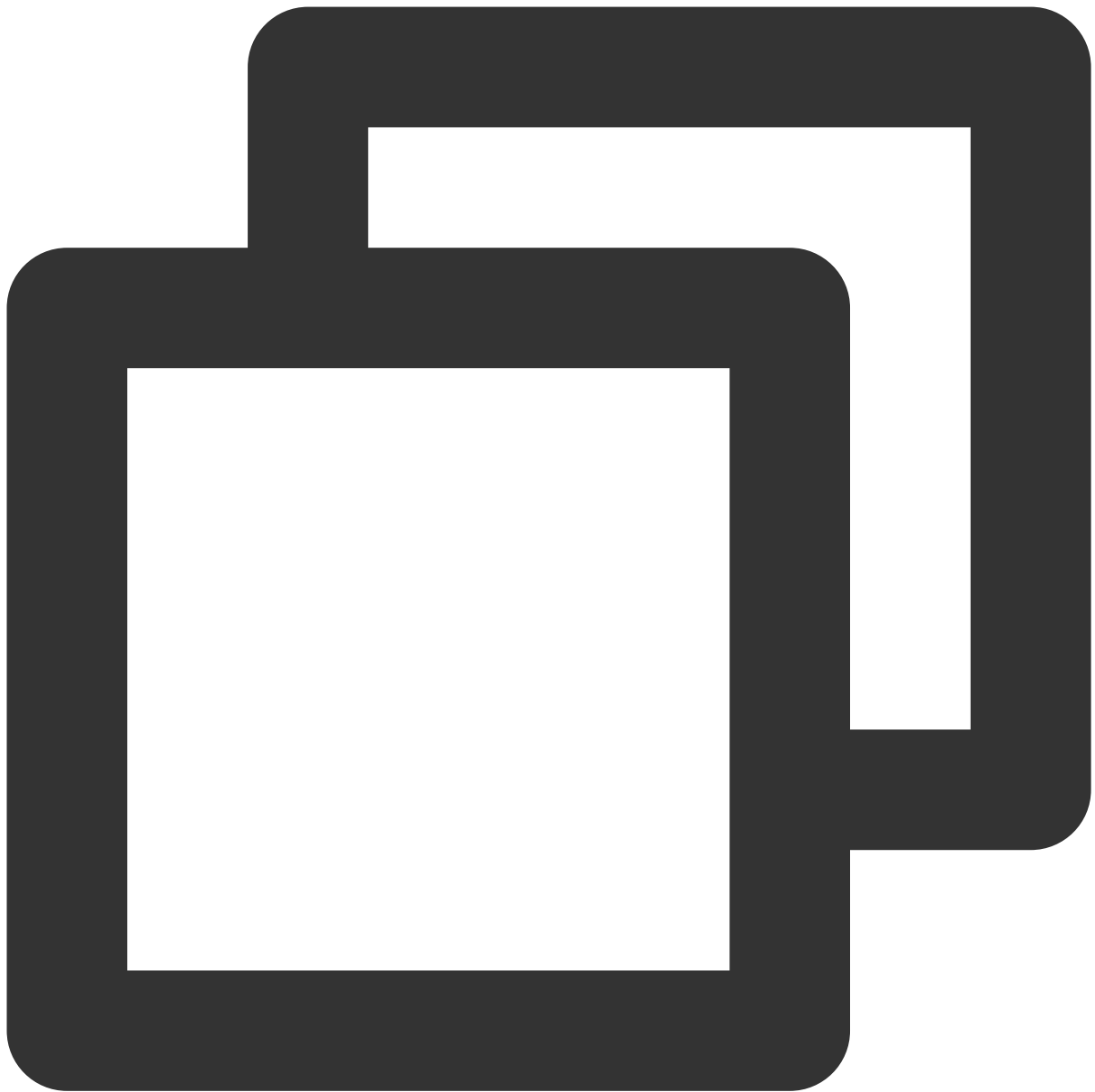
```
public func setCallingBell(filePath: String)
```

## enableMuteMode

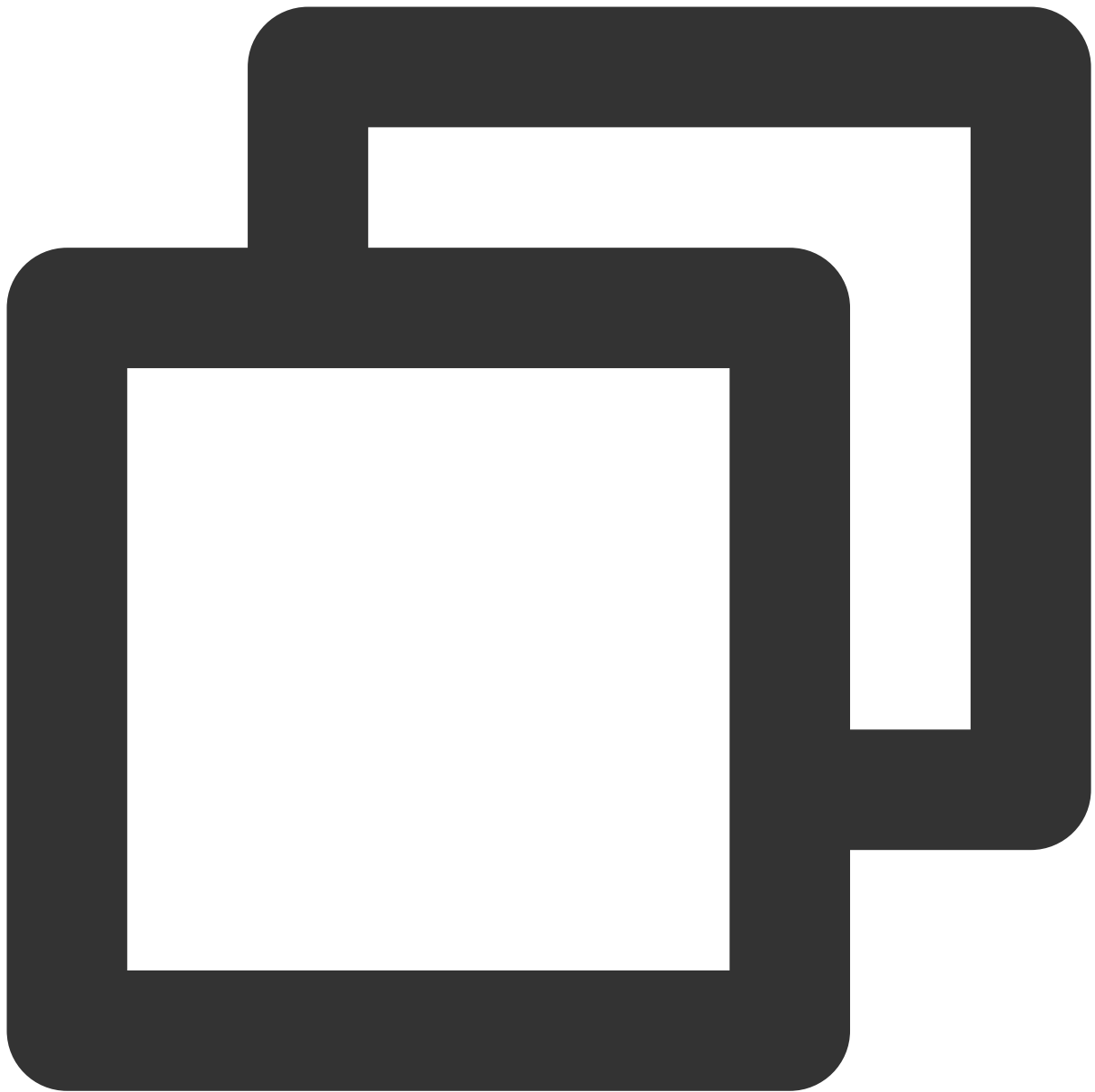
This API is used to set whether to turn on the mute mode.

Objective-C

Swift



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



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

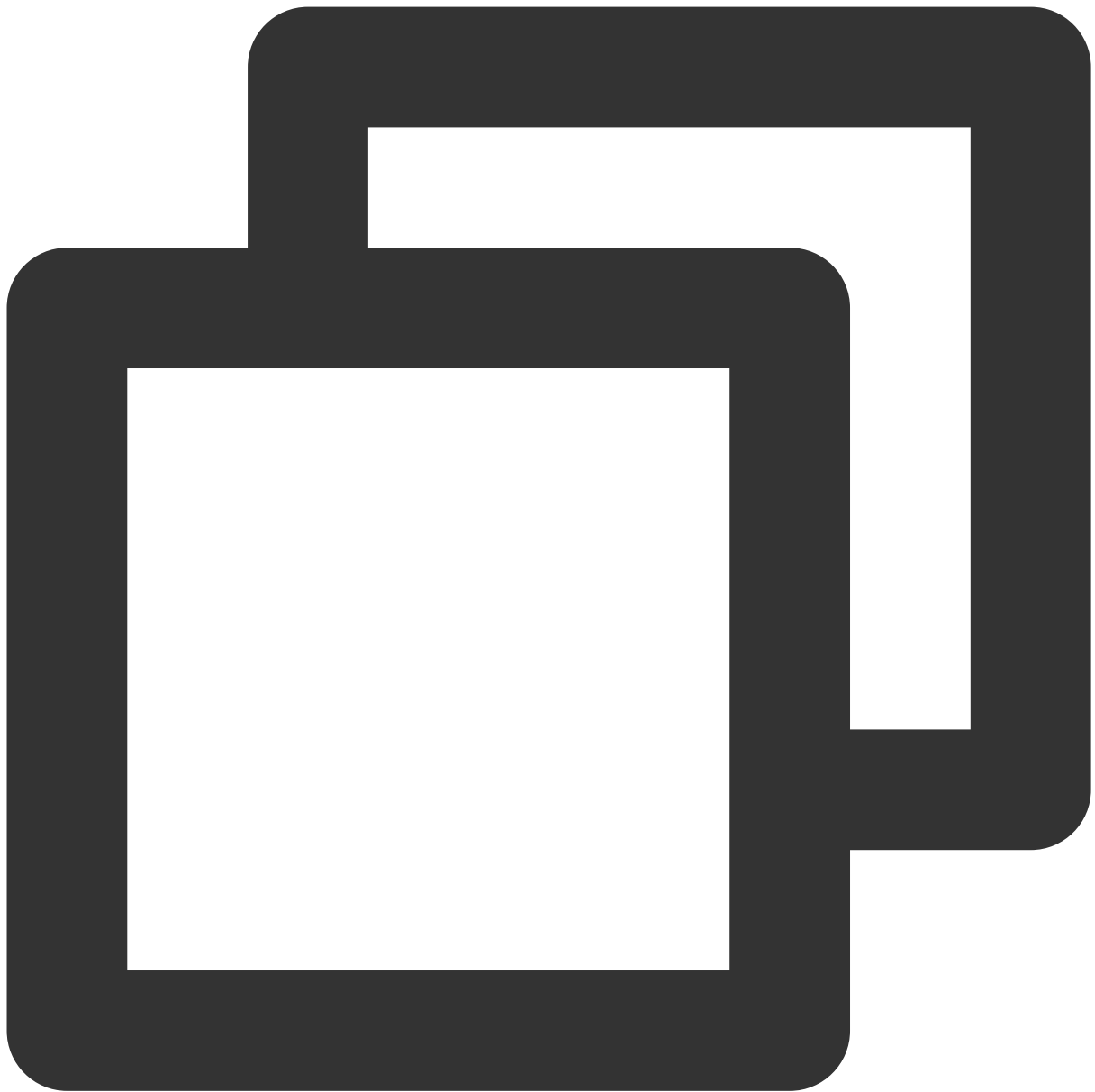
## **enableFloatWindow**

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.

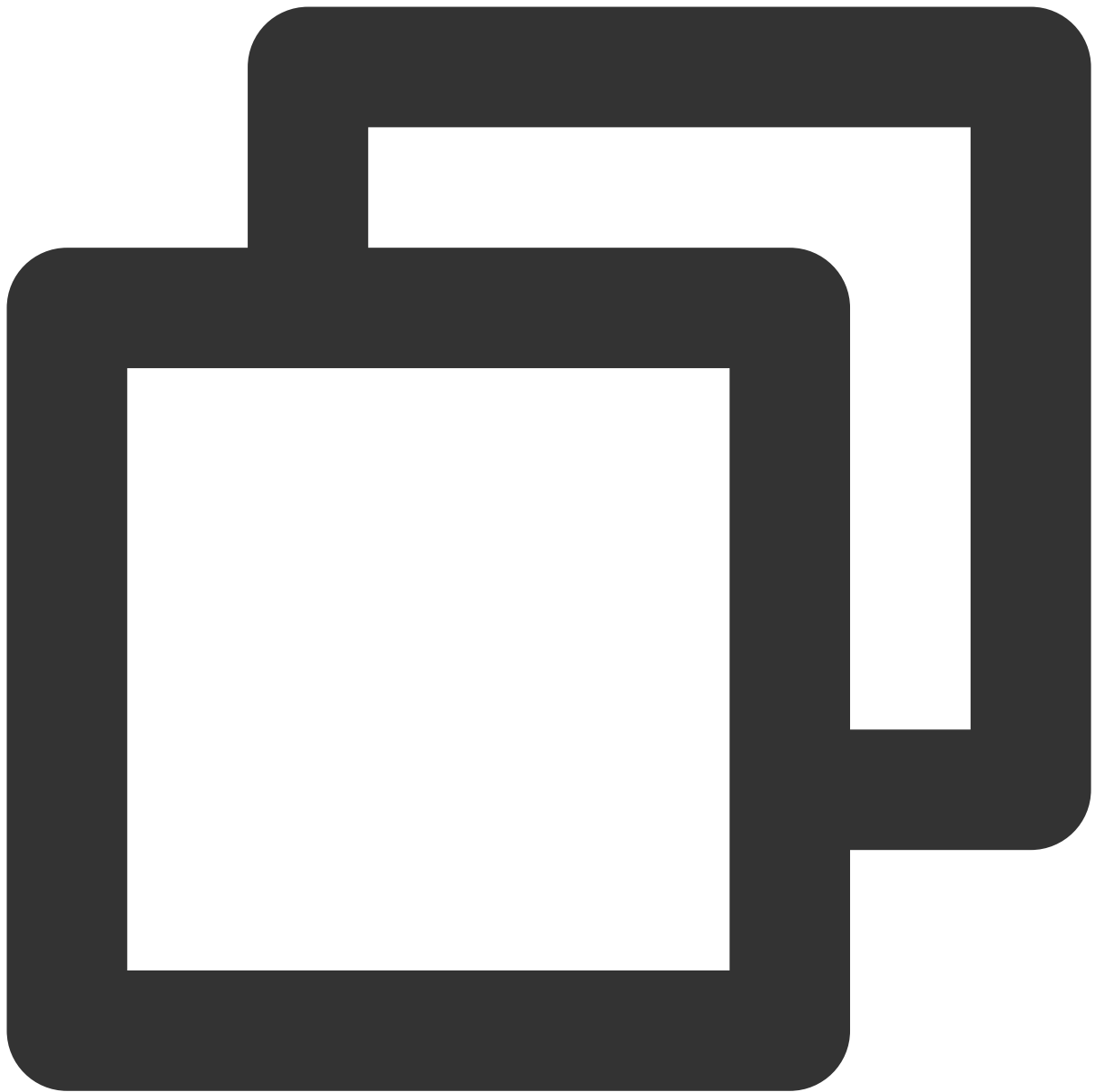
Objective-C

Swift



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



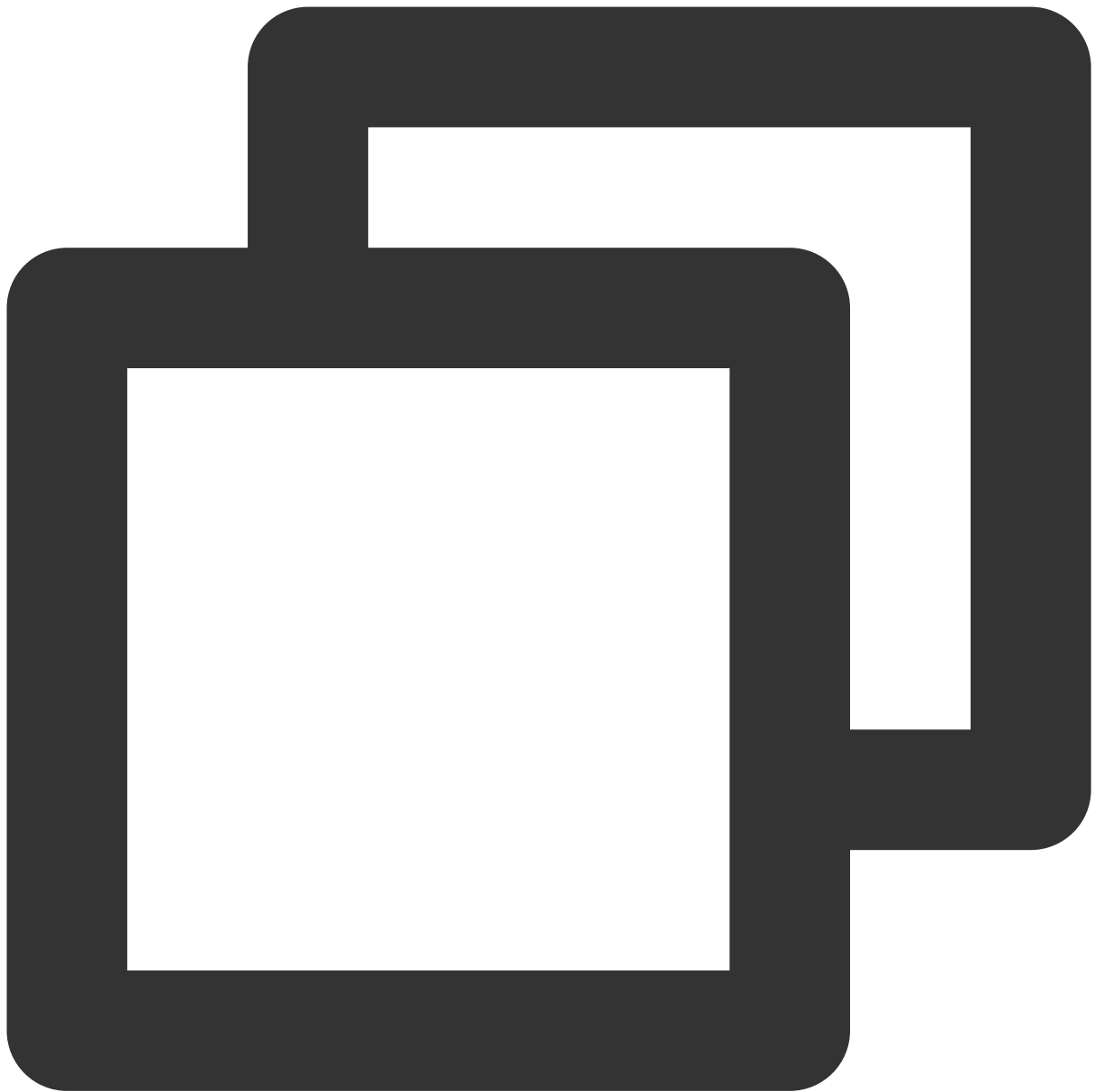


```
public func enableFloatWindow(enable: Bool)
```

### **enableIncomingBanner**

The API is used to set whether to enable the incoming banner.

The default value is `false`. The callee will pop up a full-screen call view by default when receiving the invitation. If it is set to `true`, the callee will display a banner first.



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

# TUICallEngine

Last updated : 2024-07-30 11:22:55

## 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

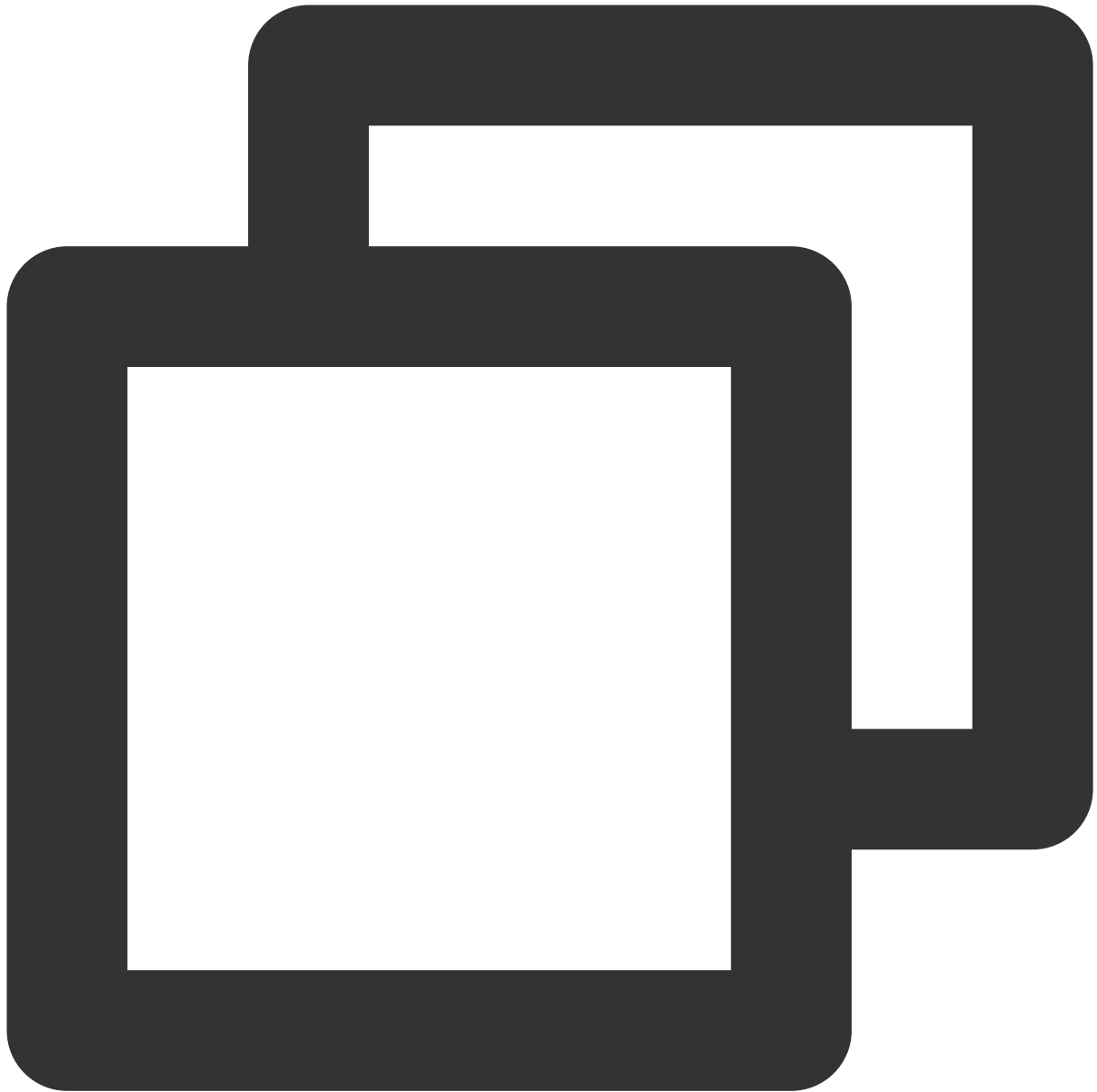
API	Description
<code>createInstance</code>	Creates a <code>TUICallEngine</code> instance (singleton mode).
<code>destroyInstance</code>	Terminates a <code>TUICallEngine</code> instance (singleton mode).
<code>init</code>	Authenticates the basic audio/video call capabilities.
<code>addObserver</code>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

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

## Details

### createInstance

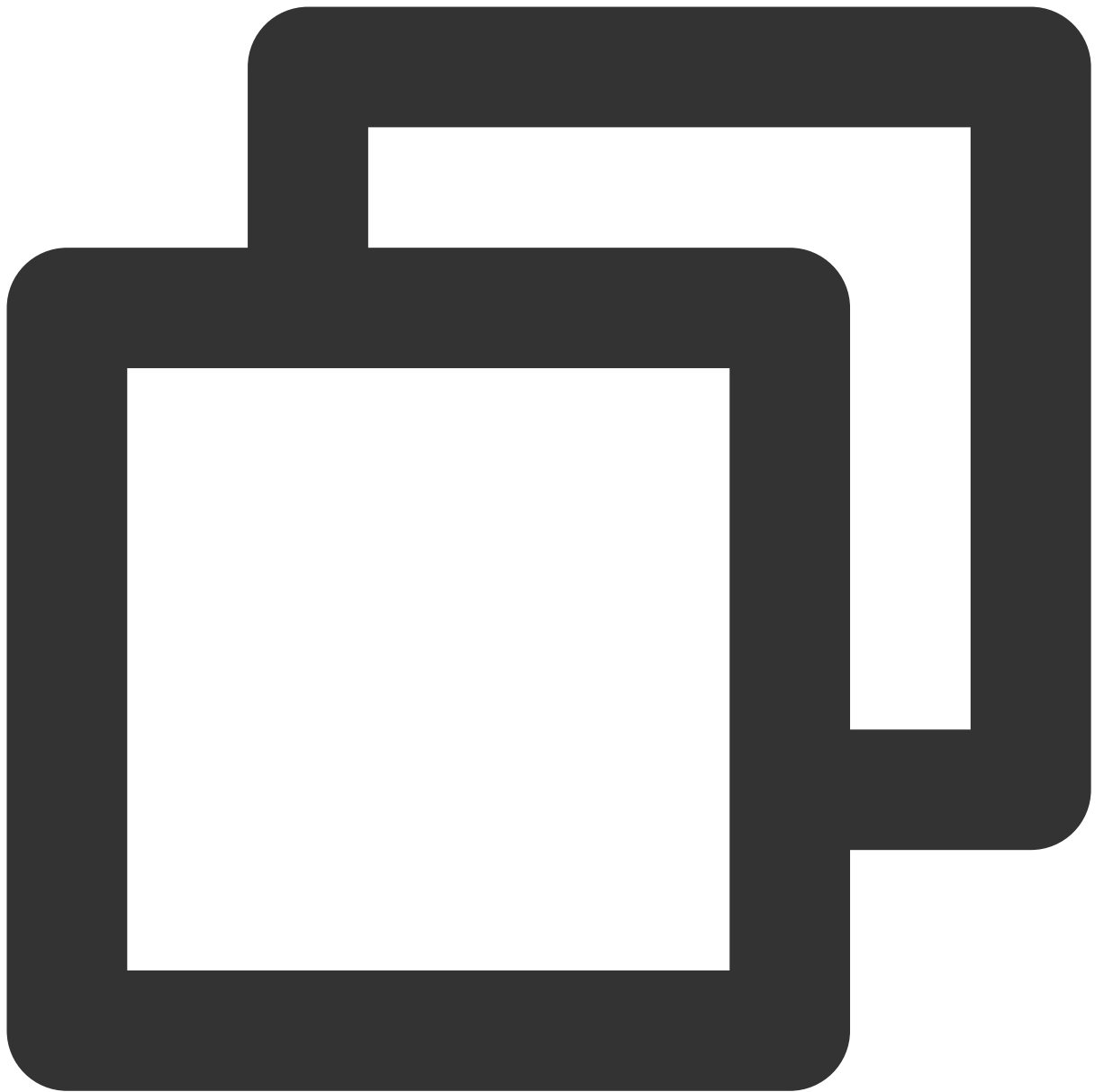
This API is used to create a `TUICallEngine` singleton.



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

### **destroyInstance**

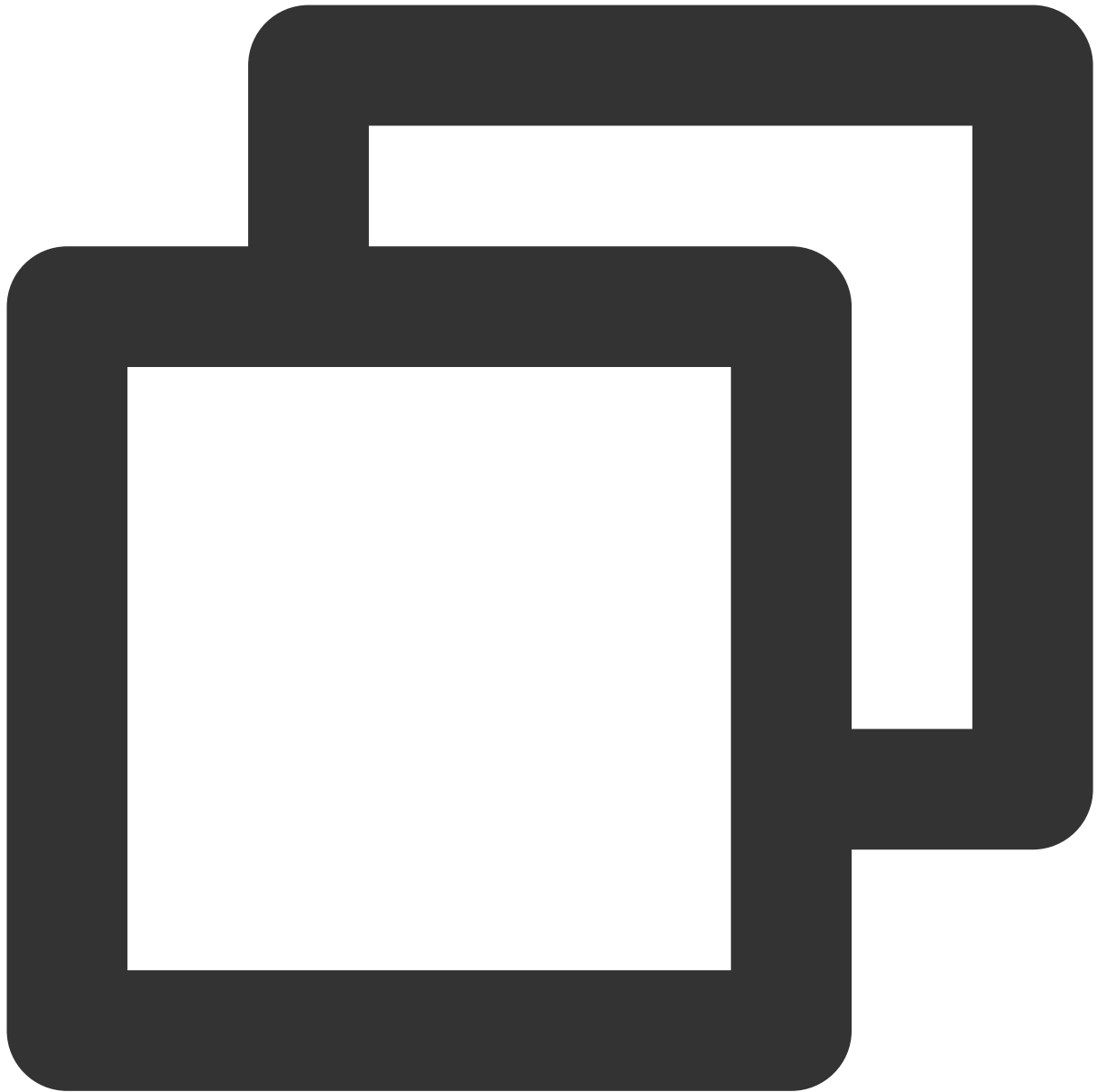
This API is used to terminate a `TUICallEngine` singleton.



```
- (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.



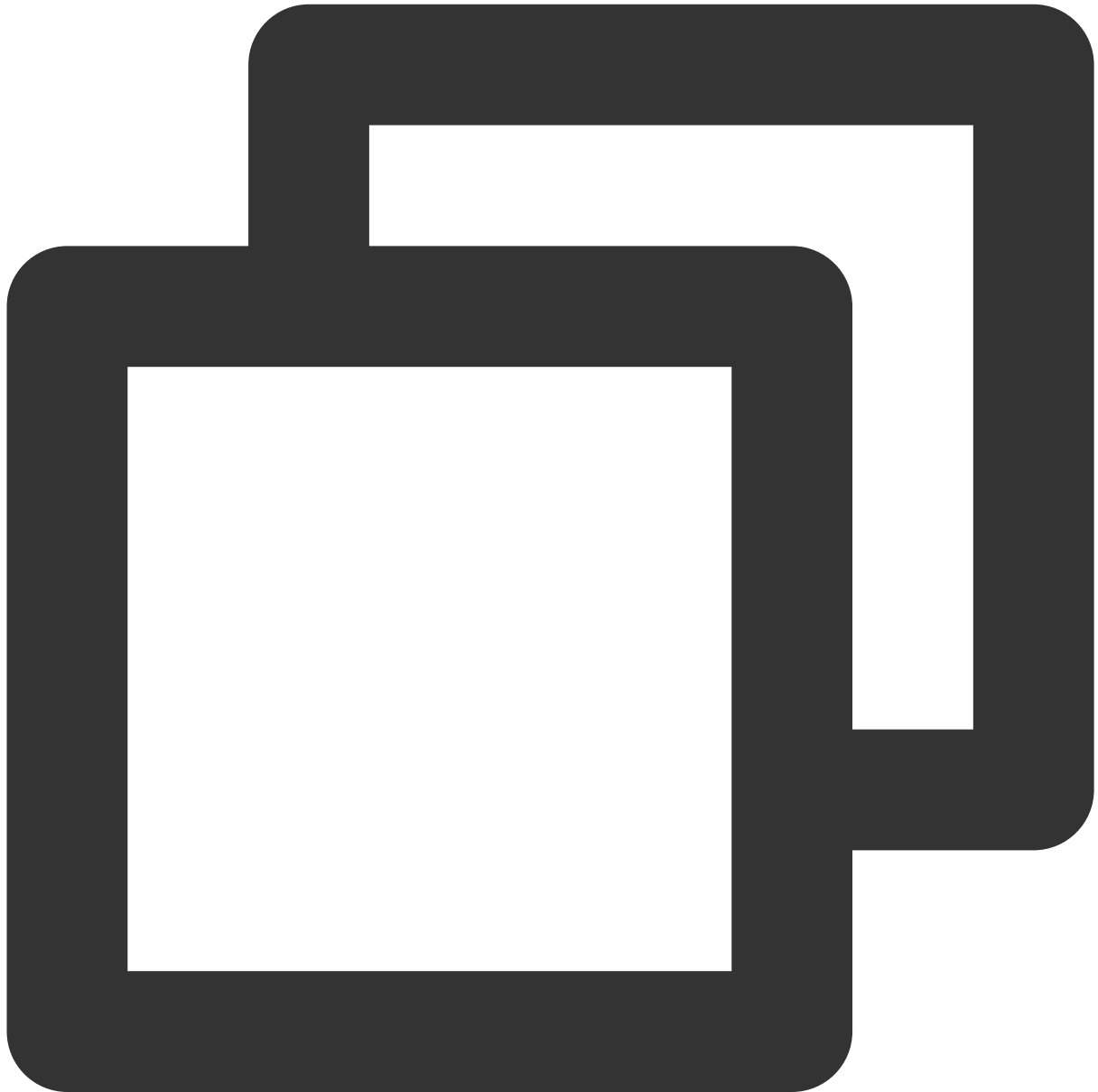
```
- (void)init:(NSString *)sdkAppID userId:(NSString *)userId userSig:(NSString *)userSig
```

Parameter	Type	Description
sdkAppID	NSString	You can view <code>SDKAppID</code> in Application Management > <b>Application Info</b> 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.

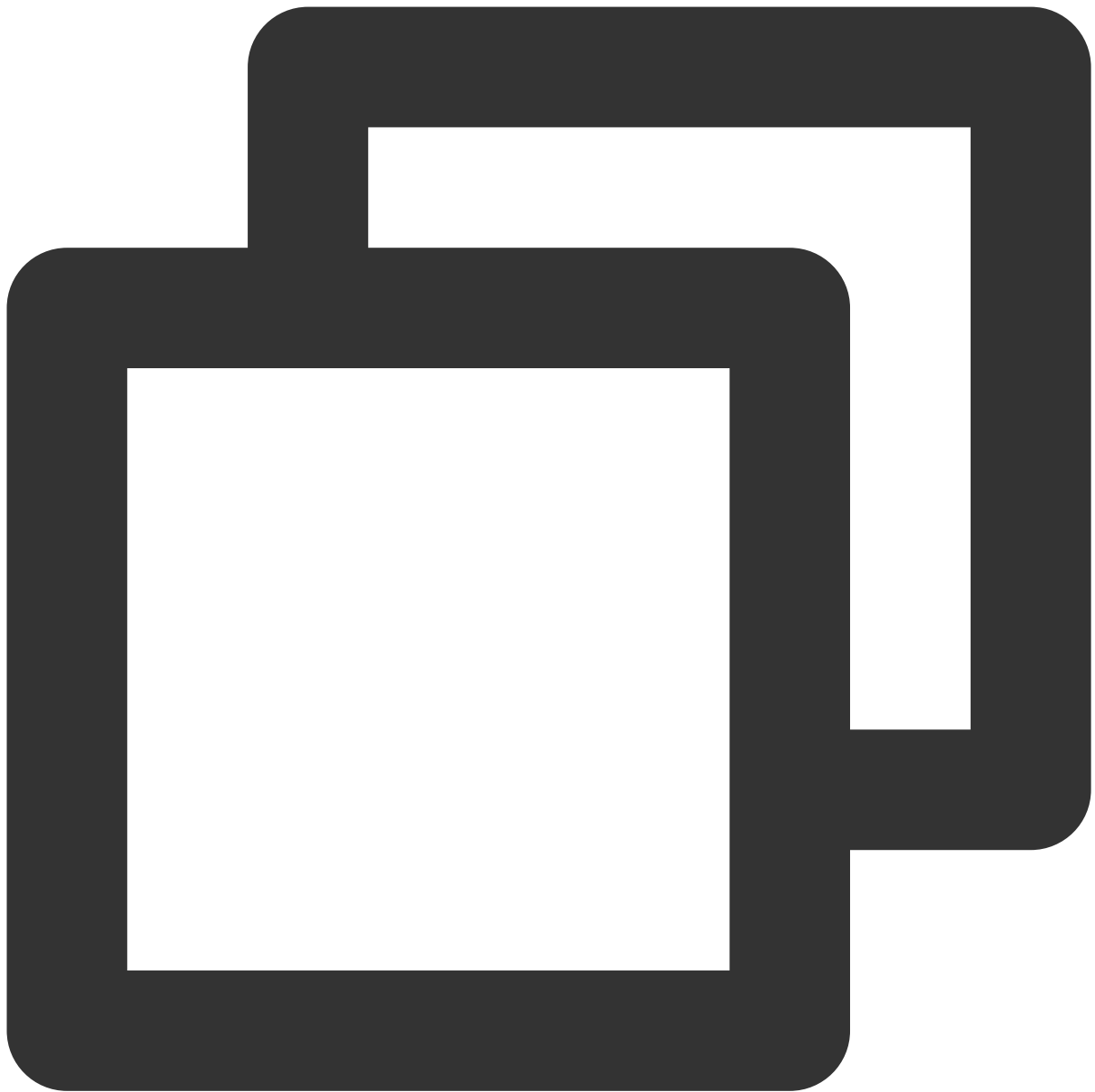


```
- (void)addObserver:(id<TUICallObserver>)observer;
```

## removeObserver

This API is used to unregister an event listener.

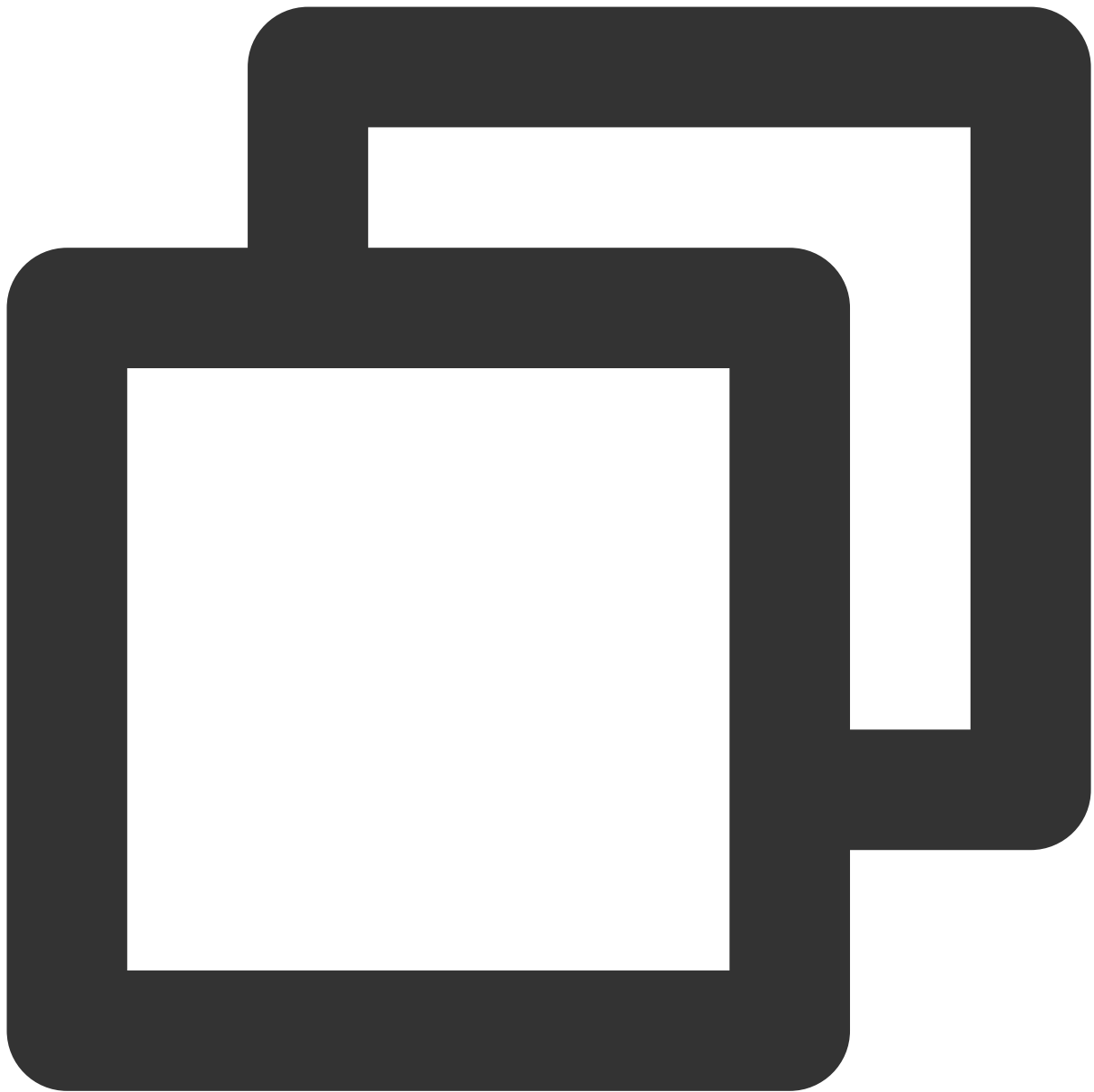




```
- (void)removeObserver:(id<TUICallObserver>)observer;
```

## call

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



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

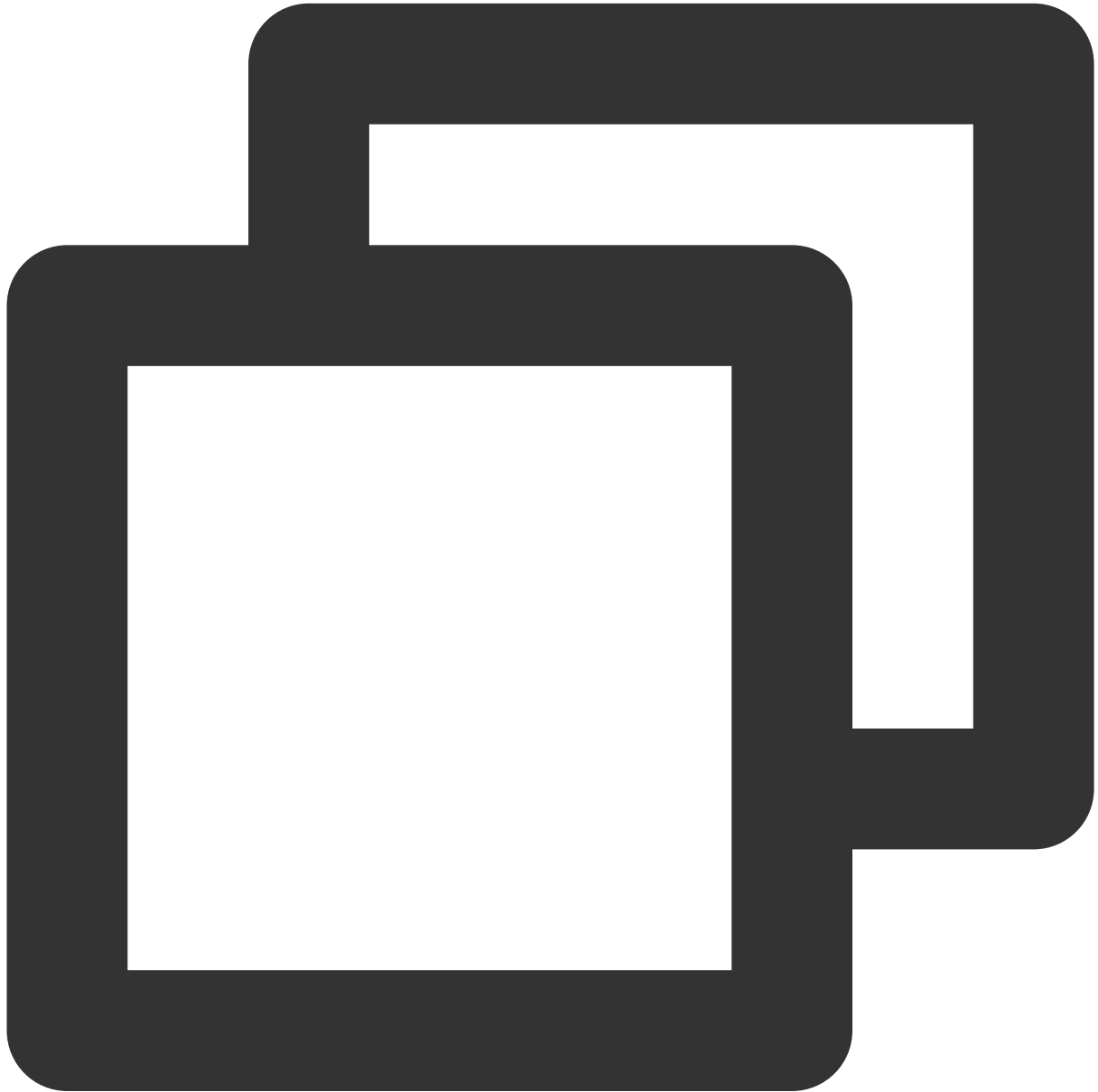
Parameter	Type	Description
userId	NSString	The target user ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	An additional parameter, such as roomId, call timeout, offline push info, etc

## groupCall

This API is used to make a group call.

### Notice :

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



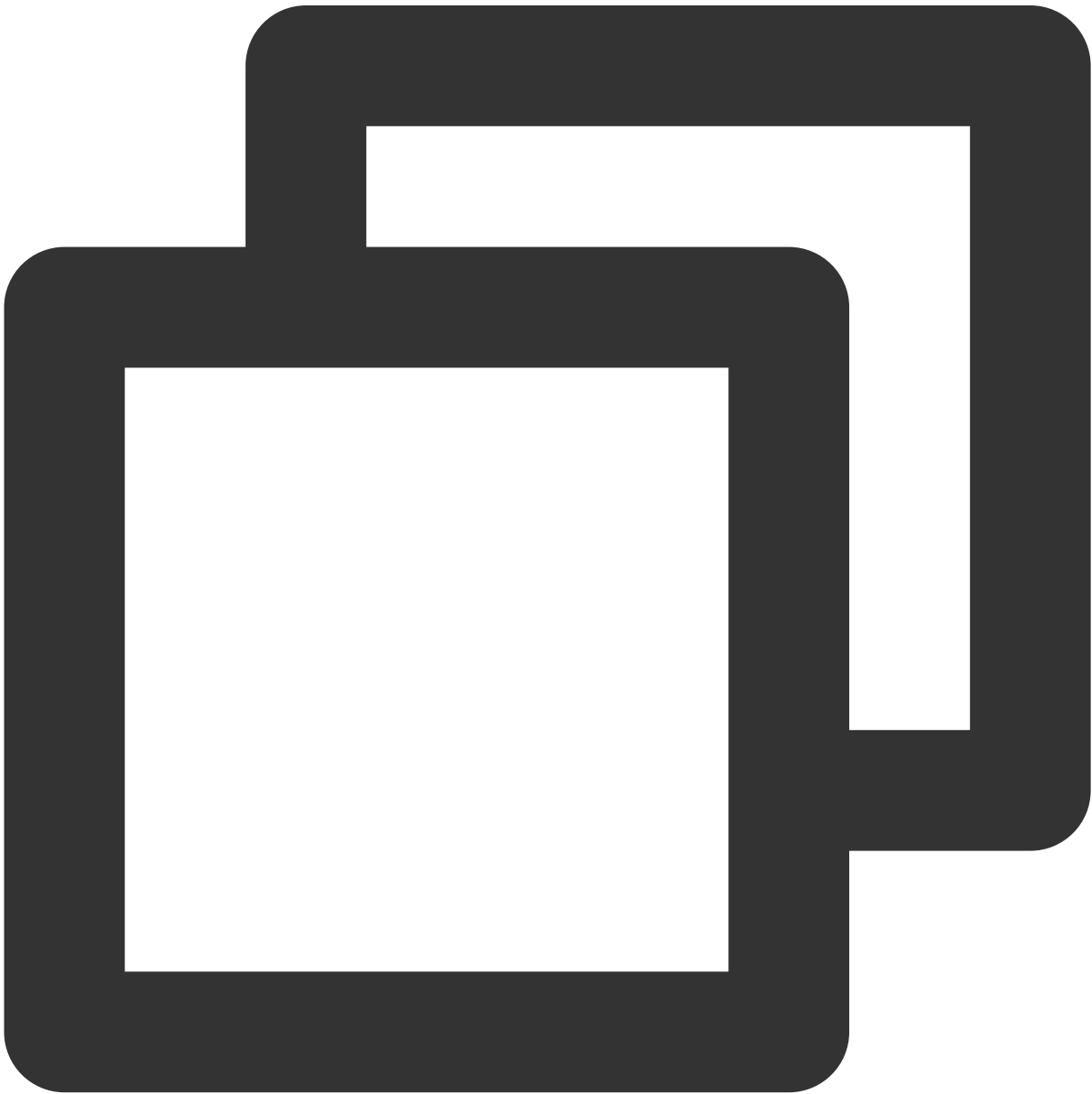
```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray <NSString *> *)userIdList
```

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

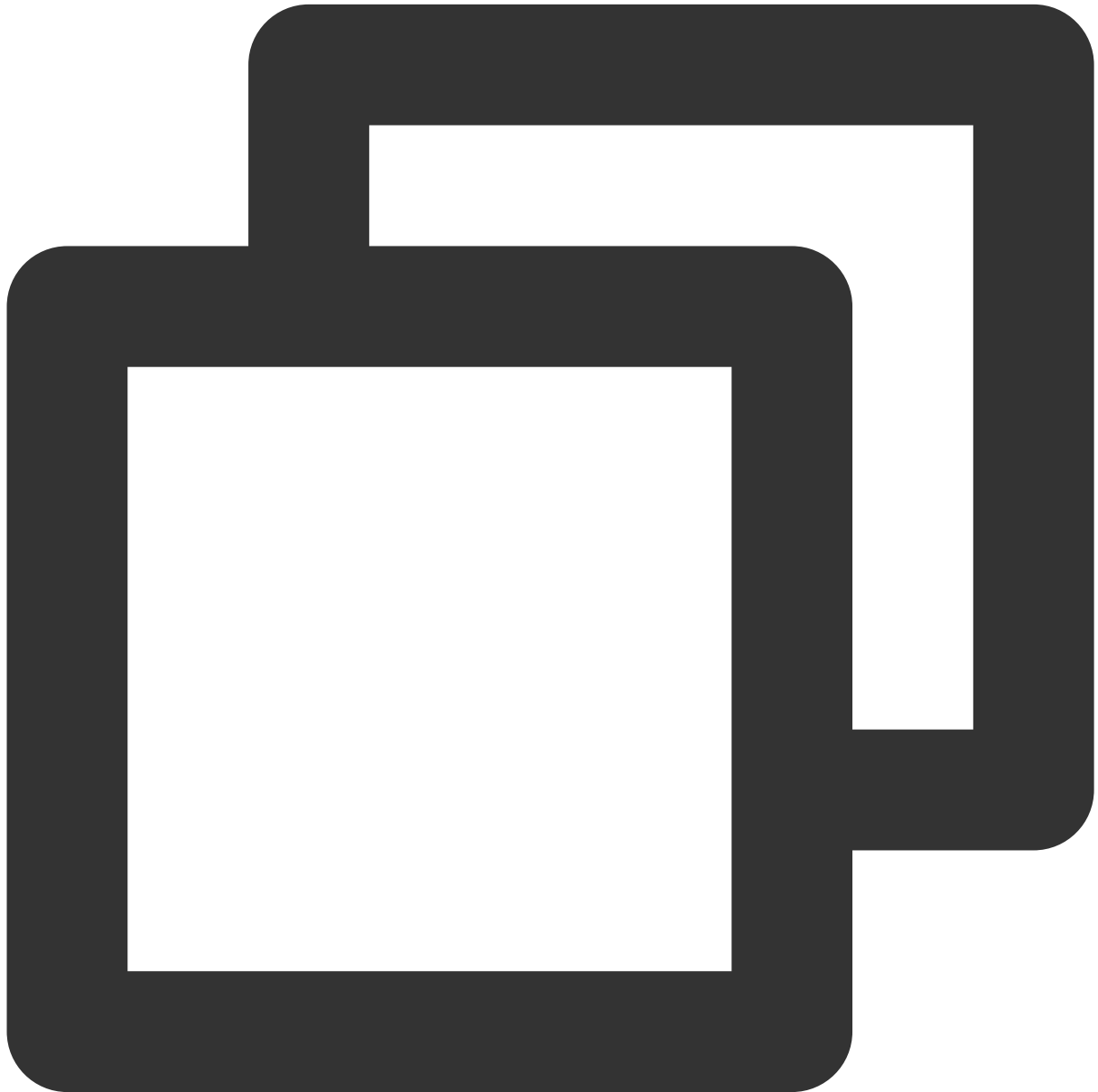
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



```
- (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.

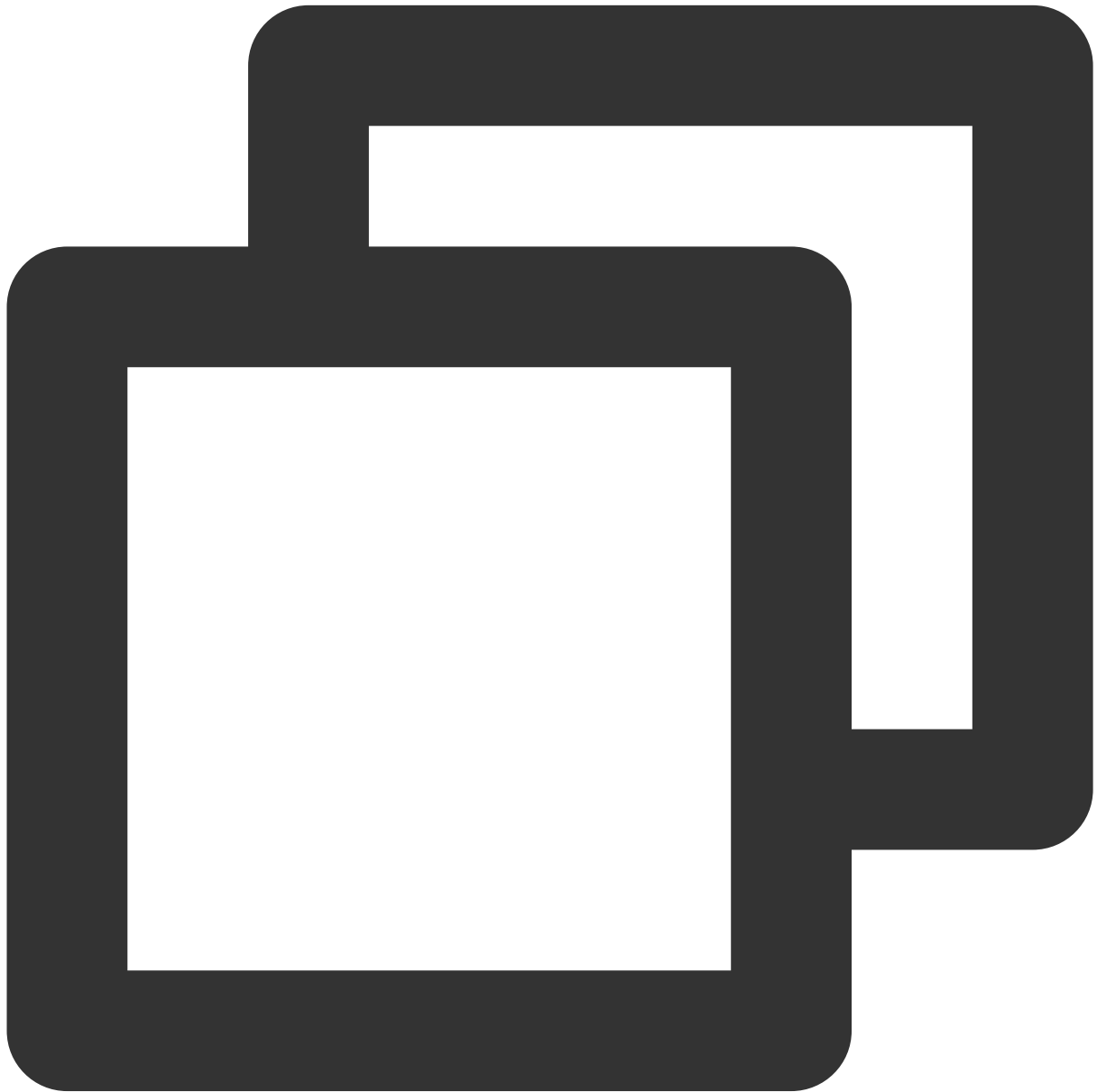


```
- (void)reject:(TUICallSucc)succ fail:(TUICallFail)fail;
```

## 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.

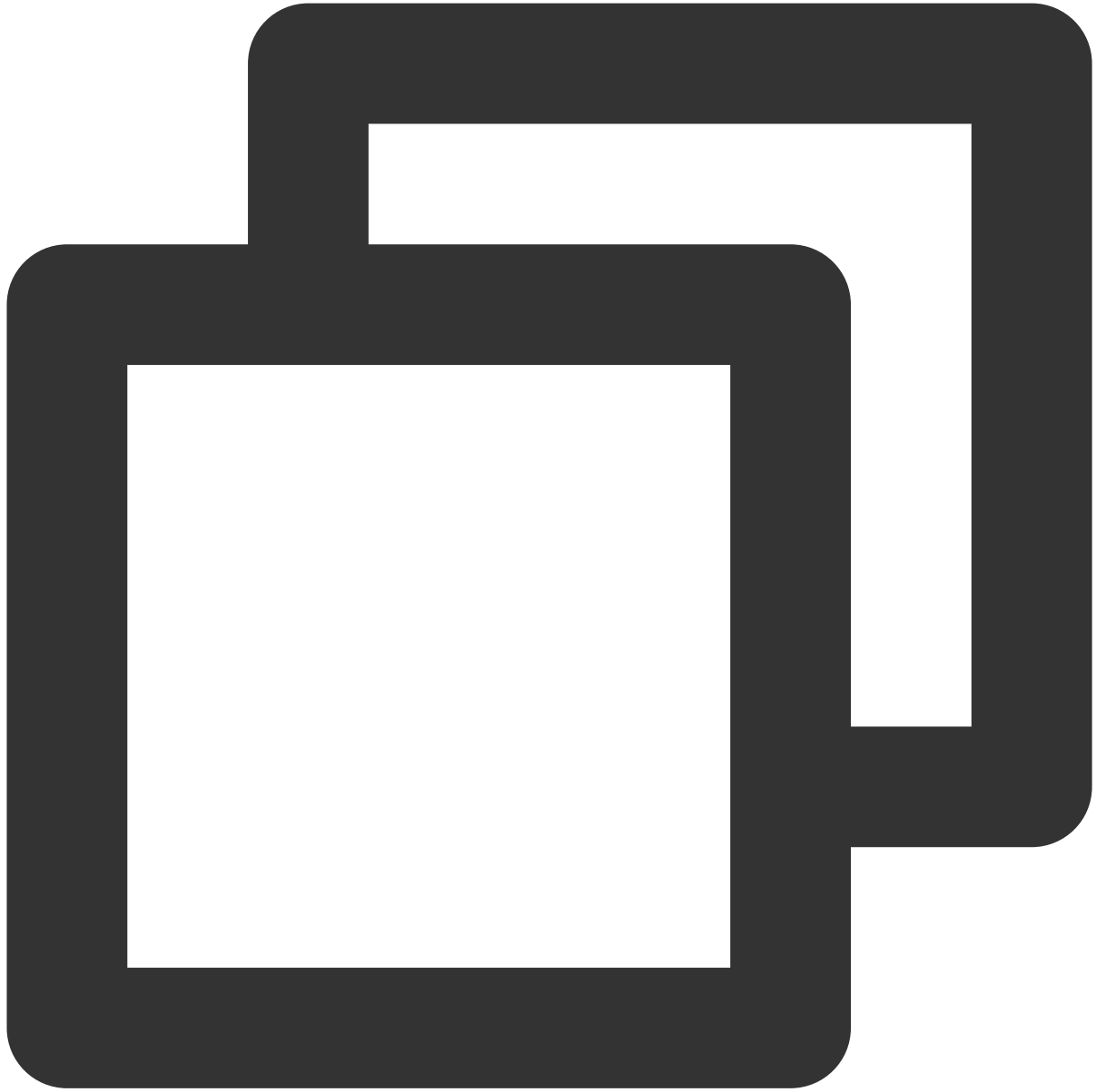
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



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

## hangup

This API is used to end a call.

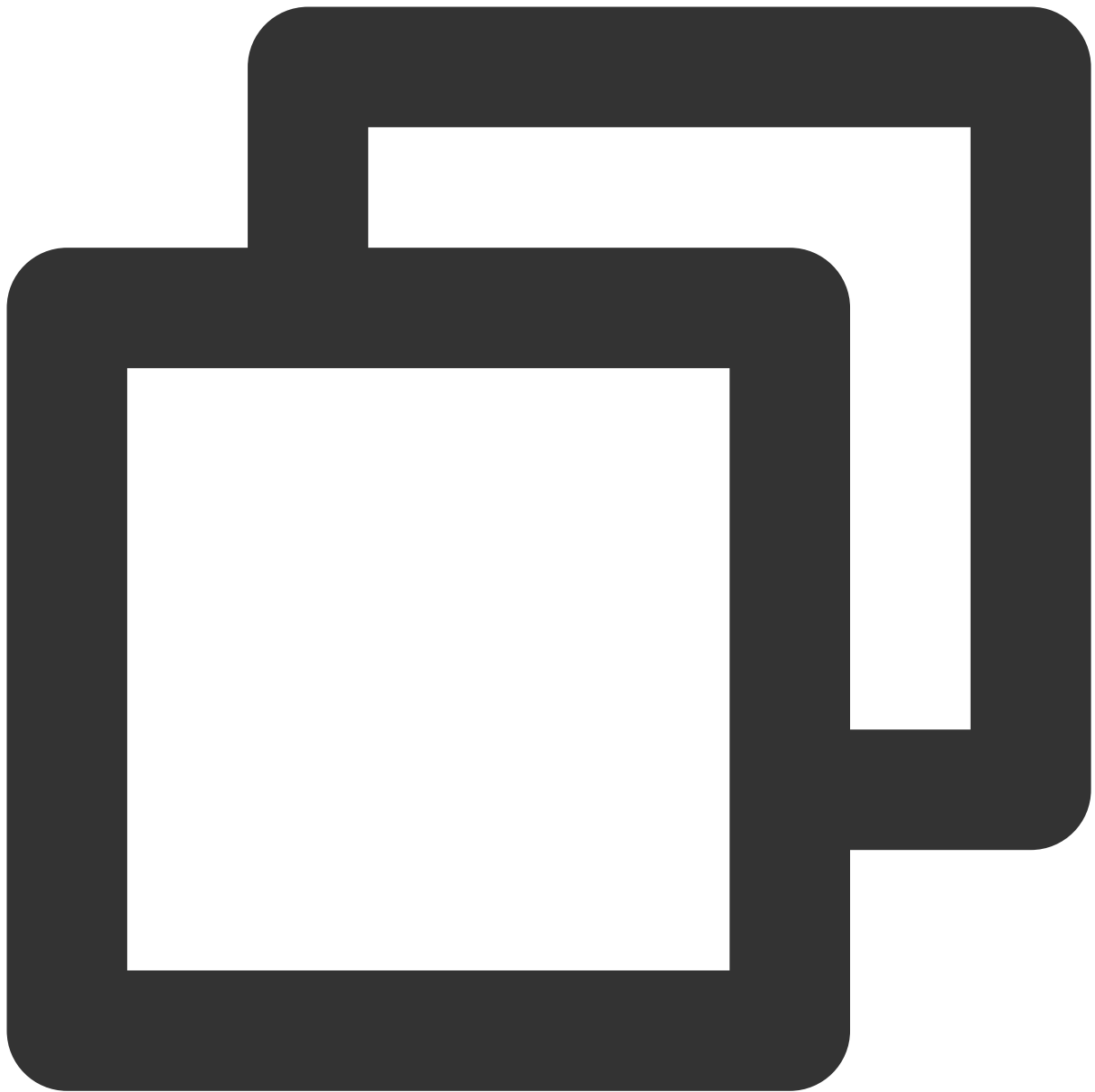


```
- (void)hangup:(TUICallSucc)succ fail:(TUICallFail)fail;
```

## 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	<a href="#">TUICallParams</a>	An additional parameter. such as roomId, call timeout, offline push info, etc

#### Notice :

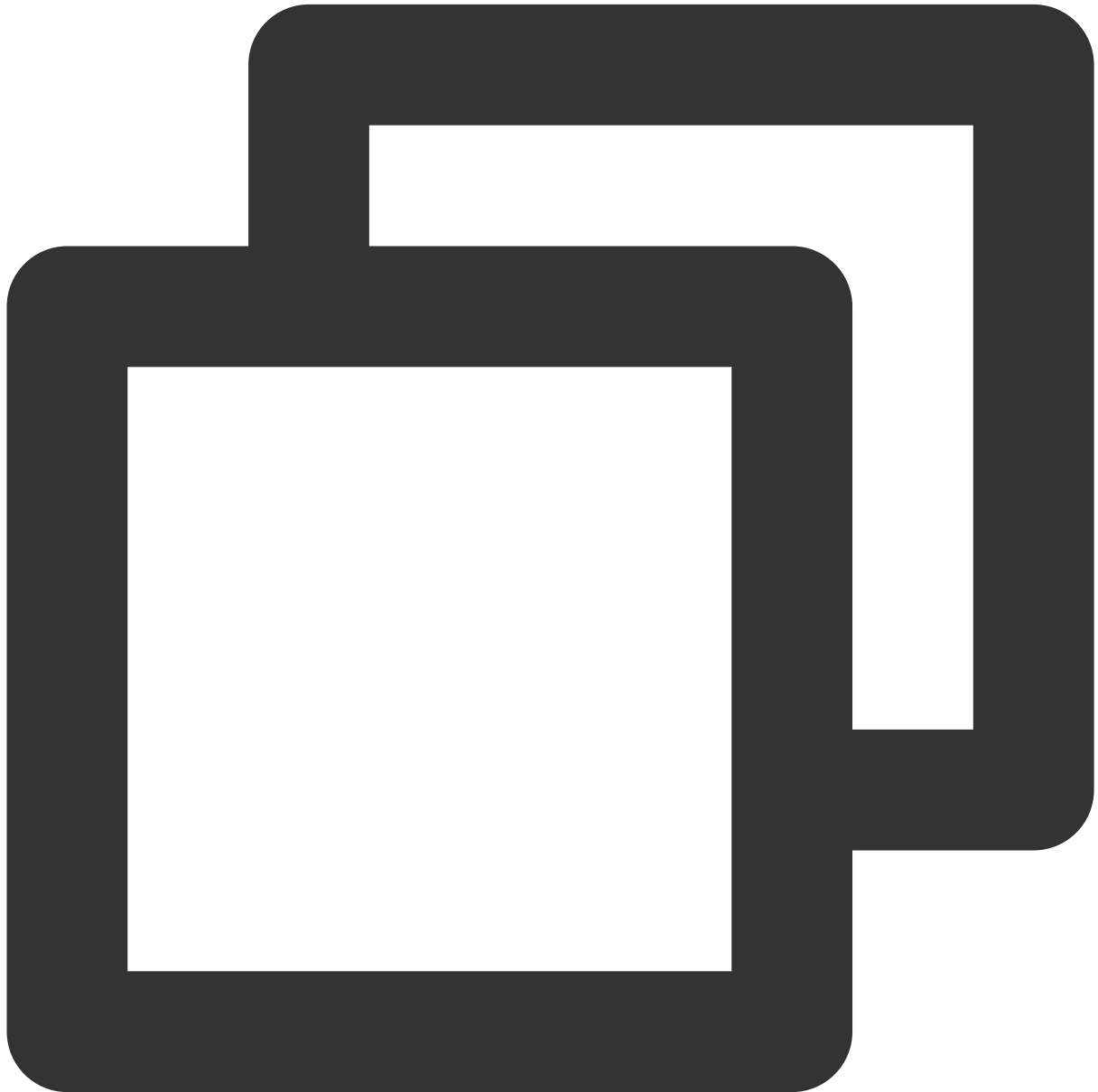


In this case, the custom RoomId is invalid. The SDK will invite others to join the room where the inviter is currently located.

## joinInGroupCall

This API is used to join a group call.

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



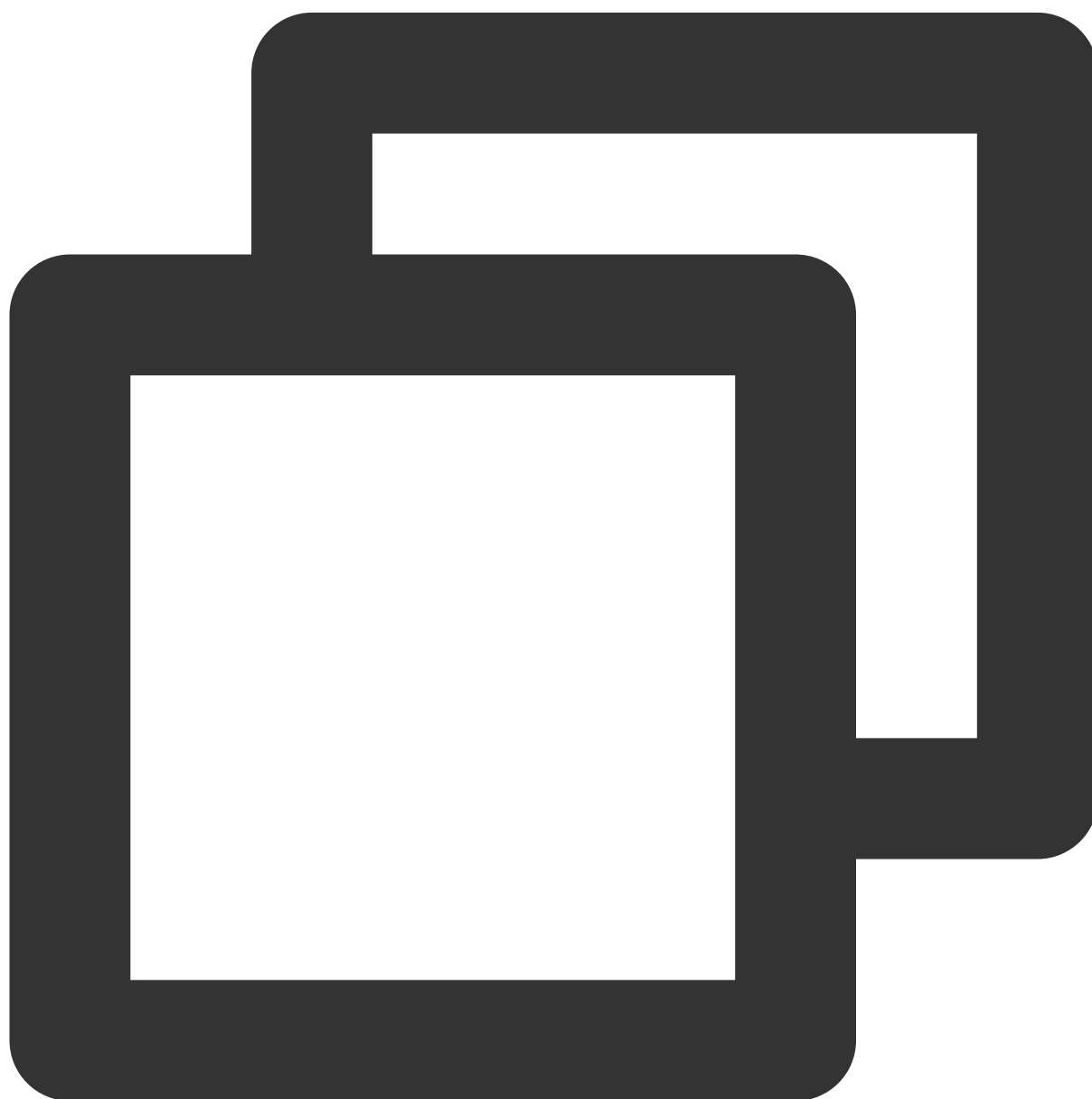
```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaTy
```

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

roomId	<a href="#">TUIRoomId</a>	The room ID.
groupId	NSString	The group ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

## switchCallMediaType

This API is used to change the call type.

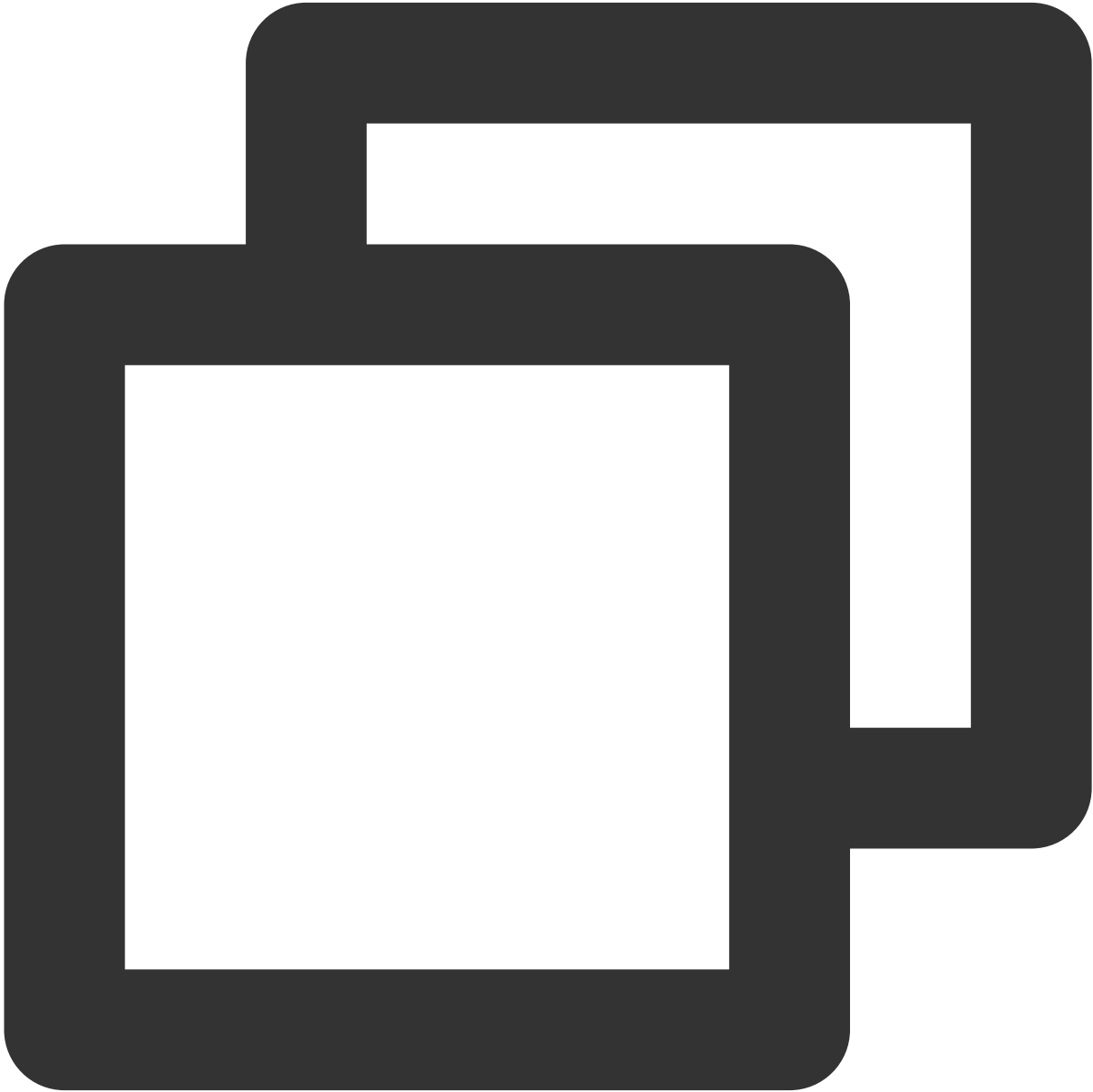


```
- (void)switchCallMediaType:(TUICallMediaType)newType;
```

Parameter	Type	Description
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

startRemoteView

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

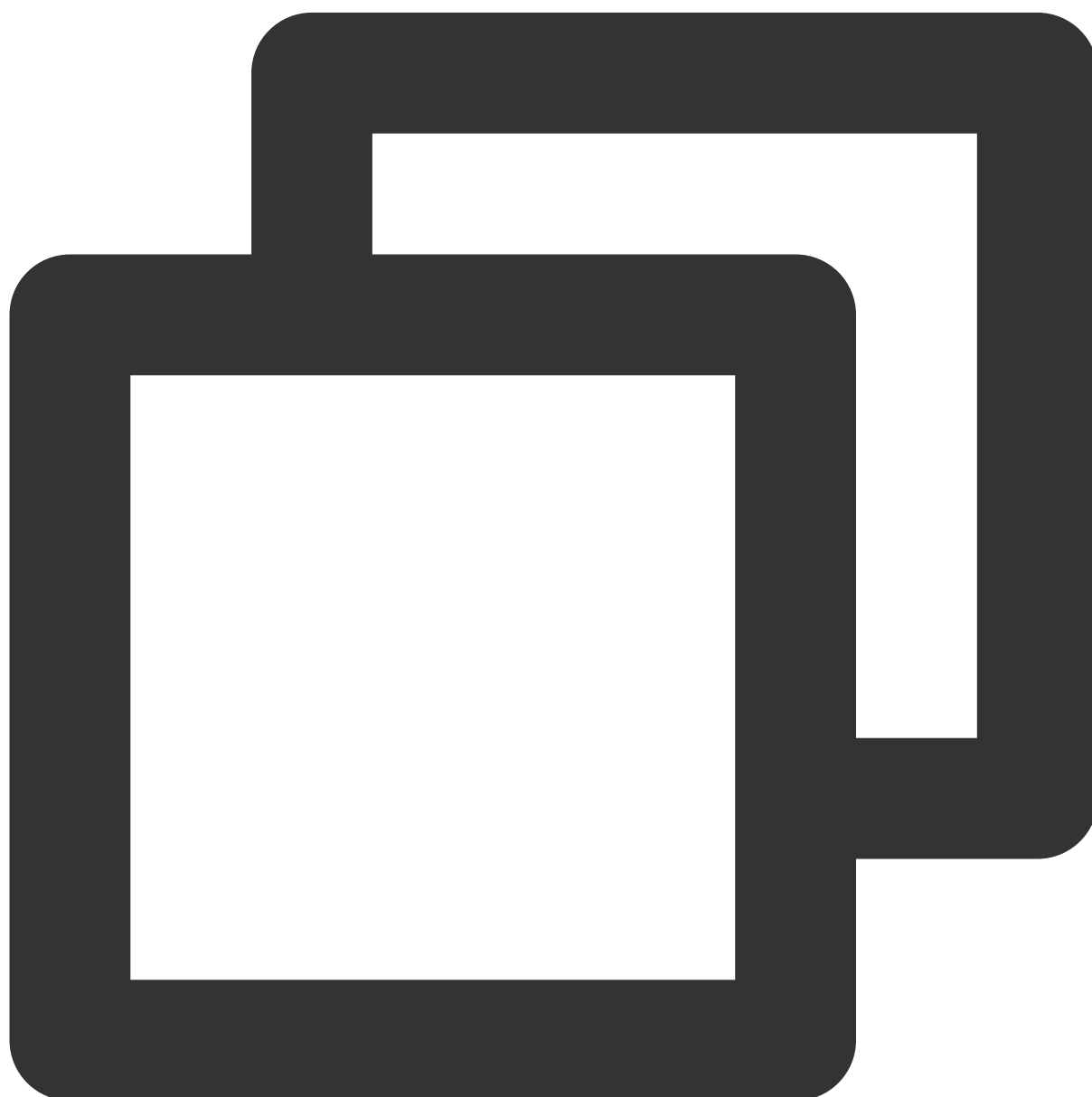


```
- (void)startRemoteView:(NSString *)userId videoView:(TUIVideoView *)videoView onPl
```

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.



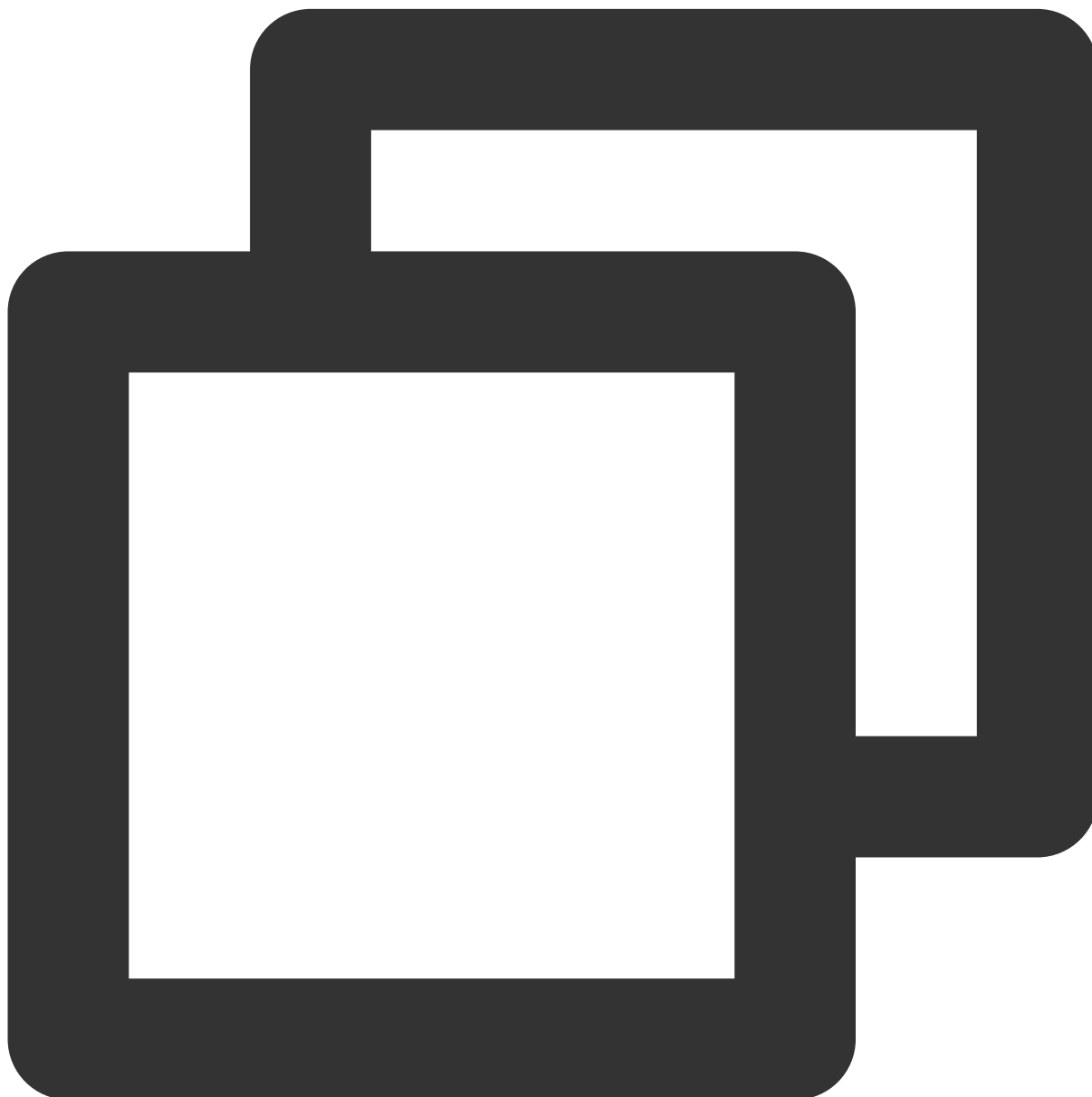
```
- (void)openCamera:(TUICamera)camera videoView:(TUIVideoView *)videoView succ:(TUIC
```

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

camera	<a href="#">TUICamera</a>	The front or rear camera.
videoView	TUIVideoView	The view to be rendered.

## closeCamera

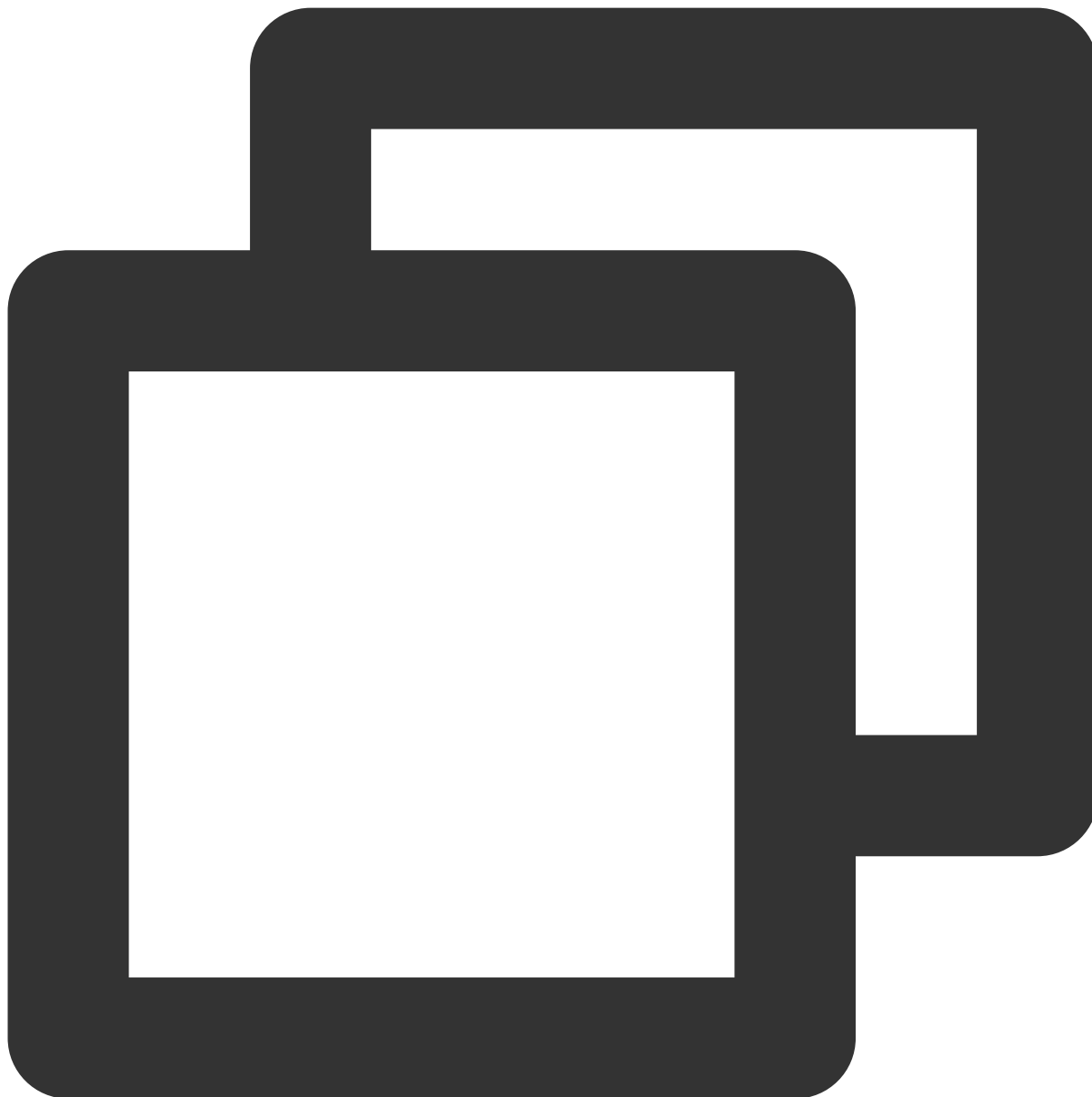
This API is used to turn the camera off.



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

## switchCamera

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

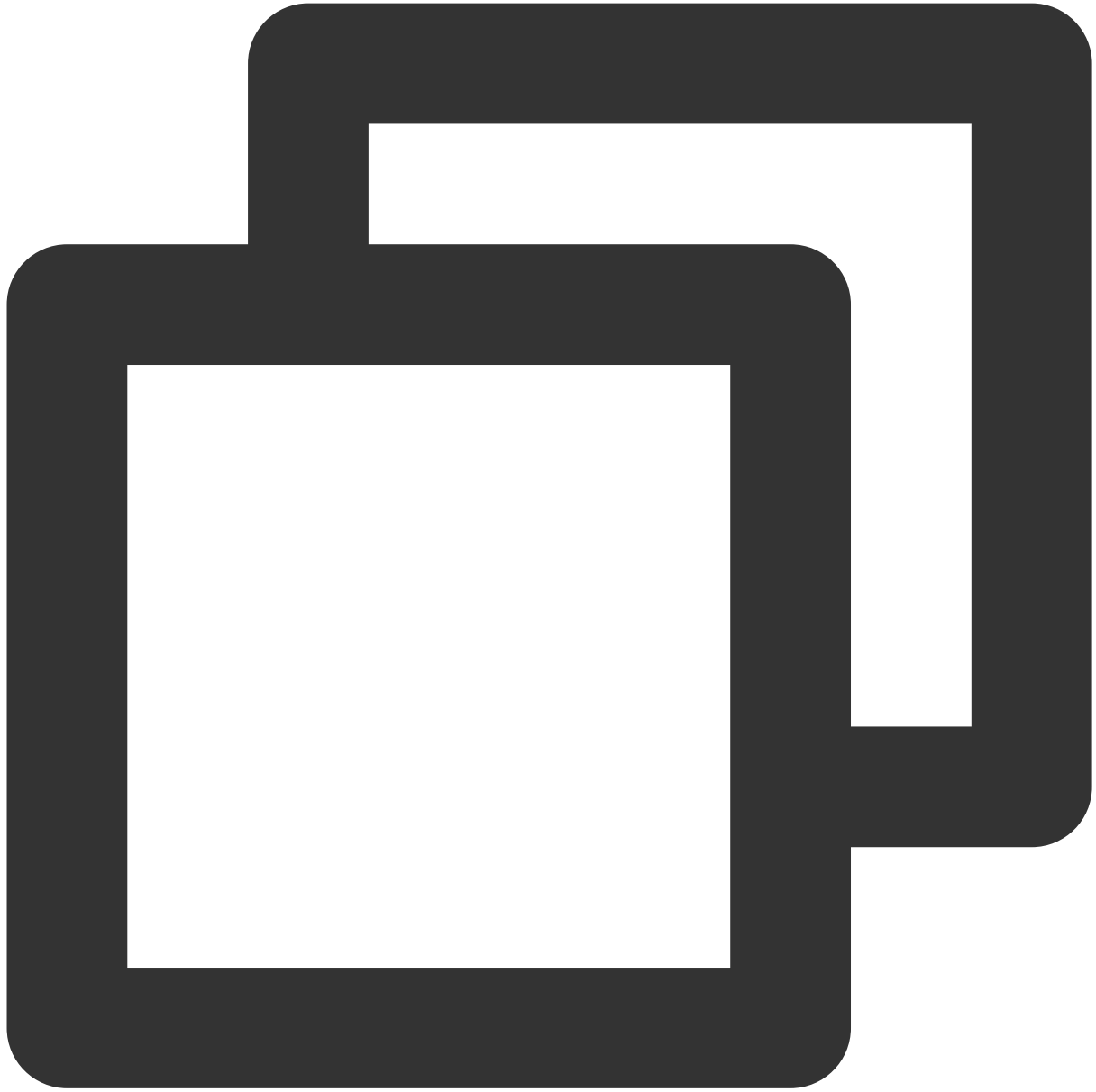


```
- (void)switchCamera:(TUICamera)camera;
```

Parameter	Type	Description
camera	<a href="#">TUICamera</a>	The front or rear camera.

## openMicrophone

This API is used to turn the mic on.

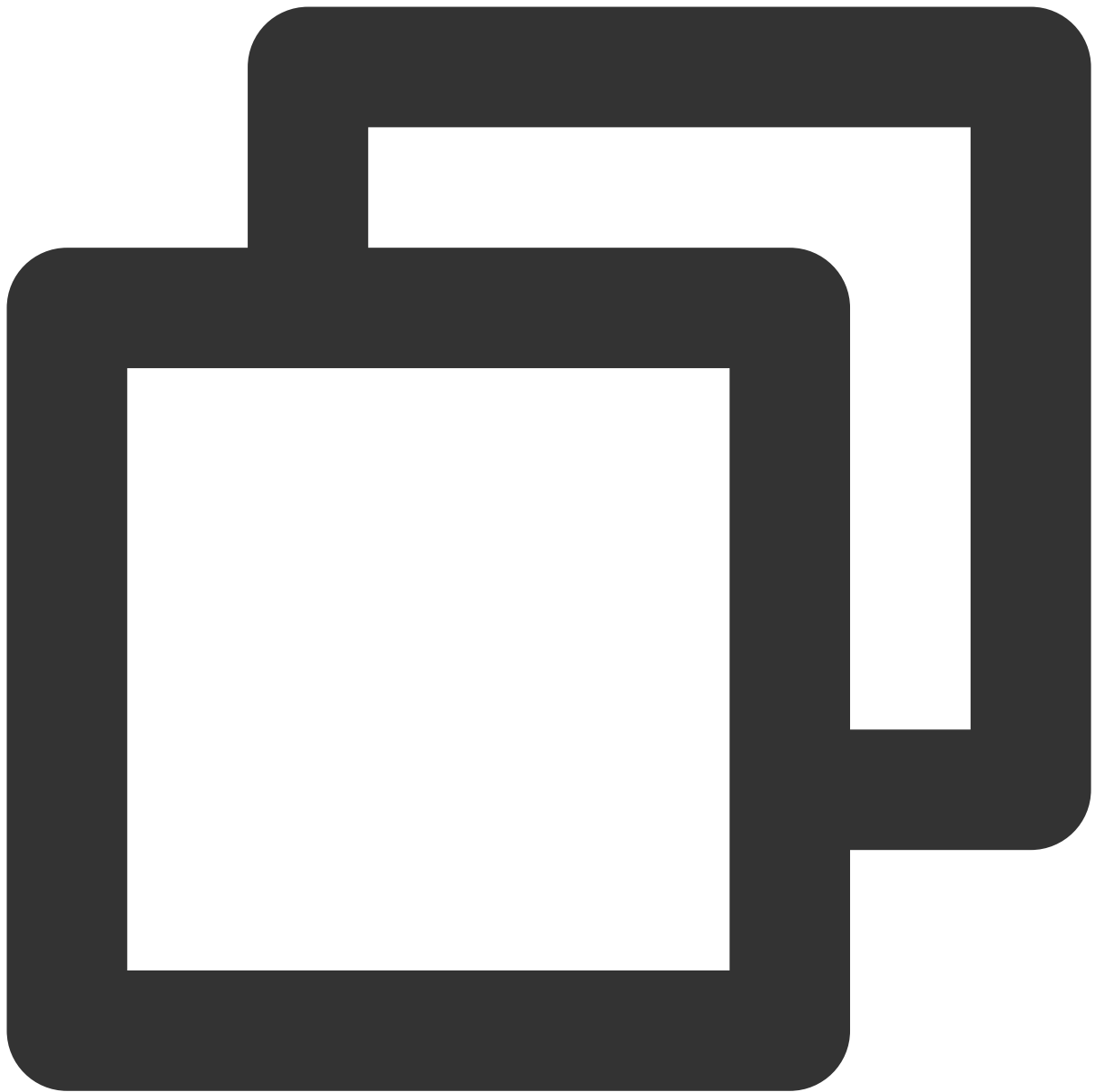


```
- (void)openMicrophone:(TUICallSucc)succ fail:(TUICallFail)fail;
```

### **closeMicrophone**

This API is used to turn the mic off.

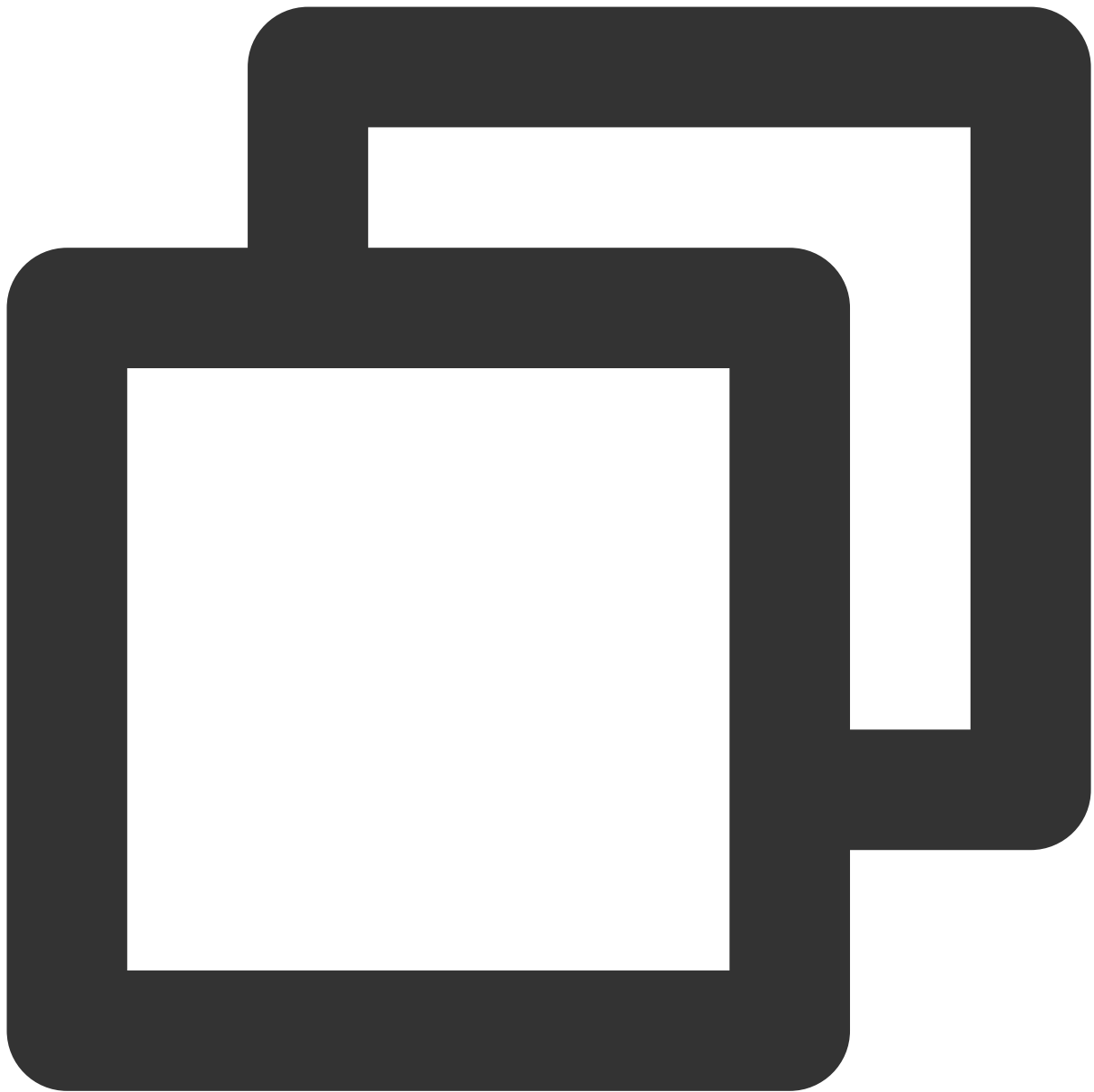




```
- (void)closeMicrophone;
```

### **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.

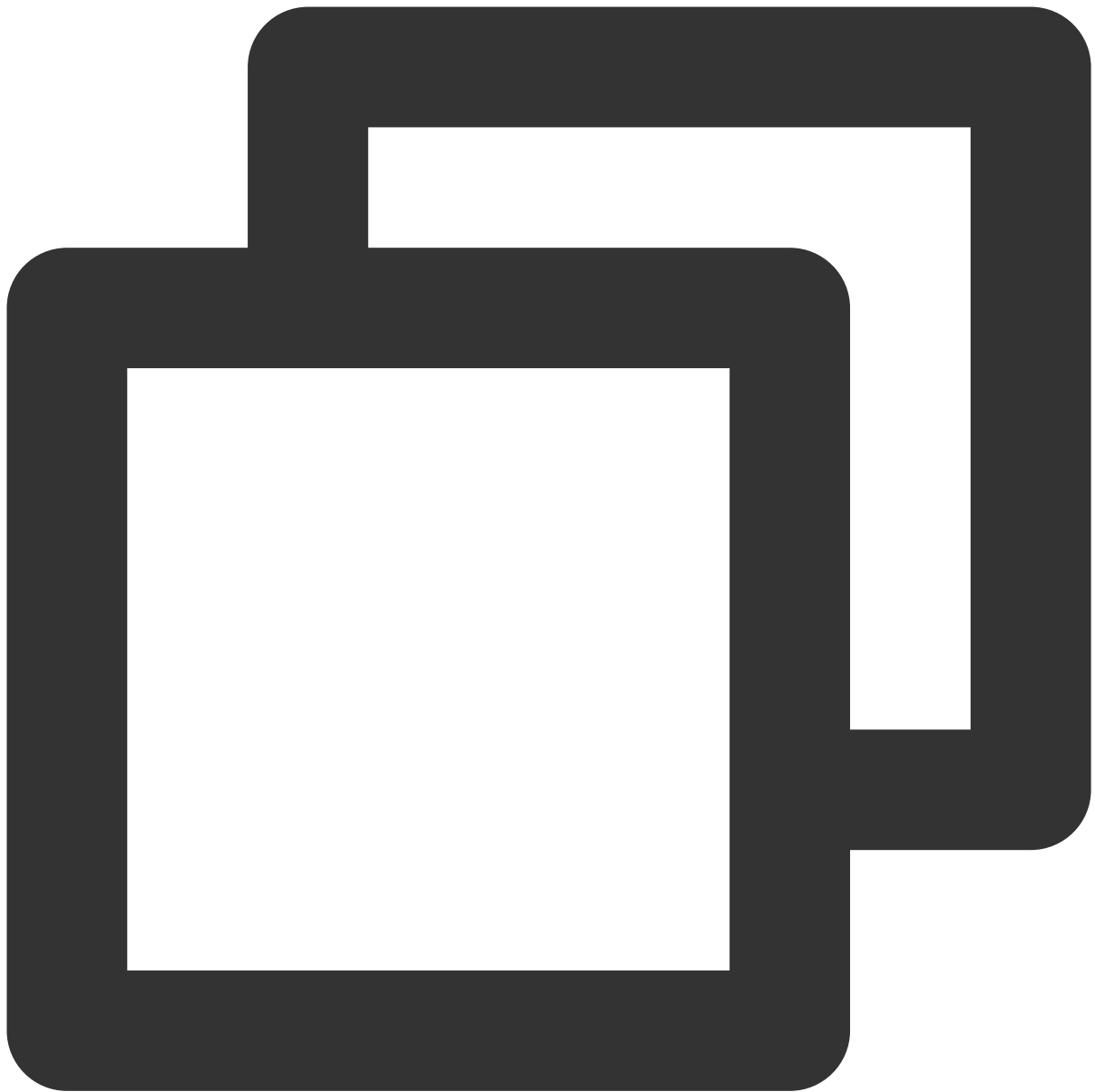


```
- (void)selectAudioPlaybackDevice:(TUIAudioPlaybackDevice) device;
```

Parameter	Type	Description
device	<a href="#">TUIAudioPlaybackDevice</a>	The speaker or receiver.

## setSelfInfo

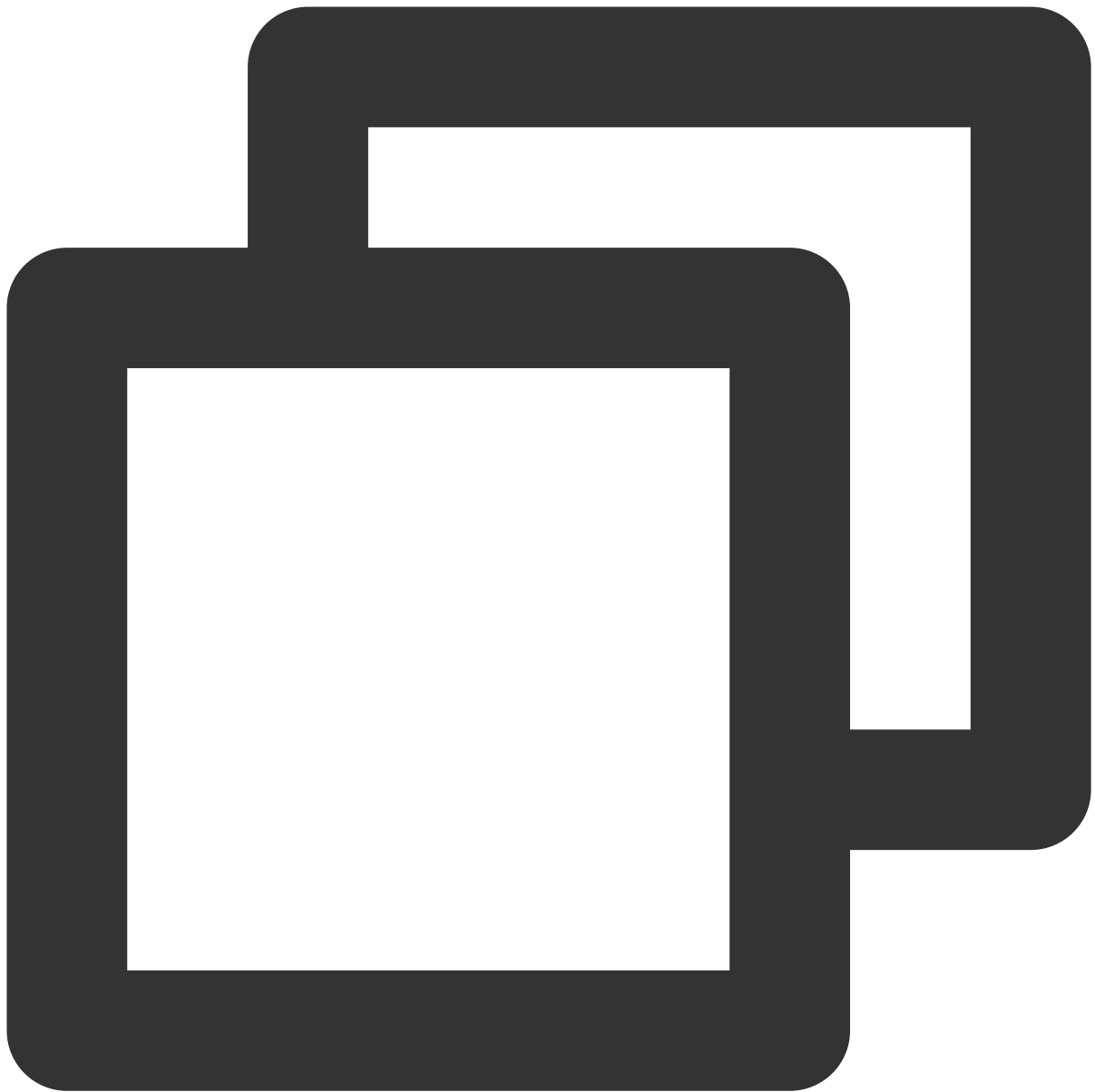
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



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

### **enableMultiDeviceAbility**

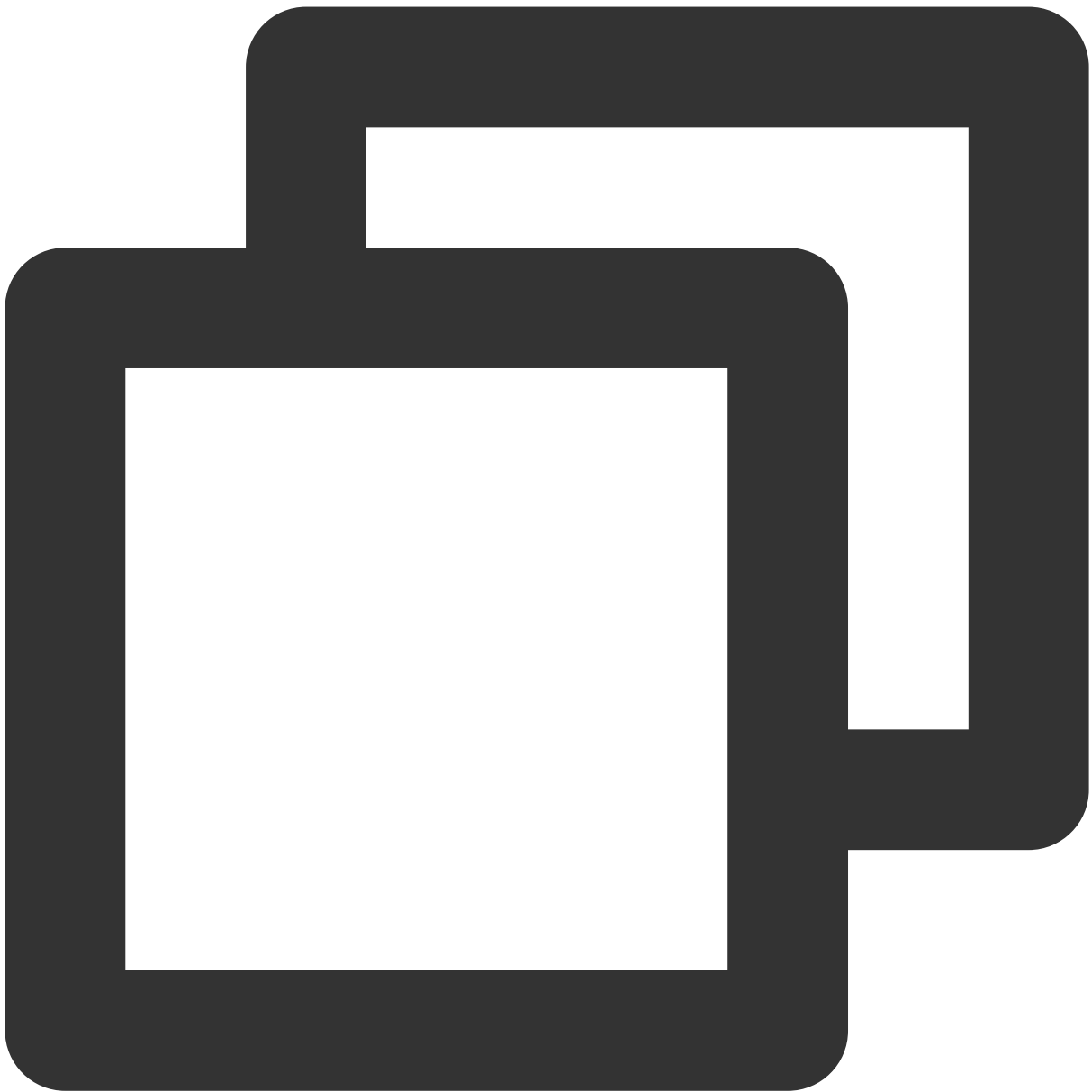
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the [Group Call package](#)).



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

## setVideoRenderParams

Set the rendering mode of video image.



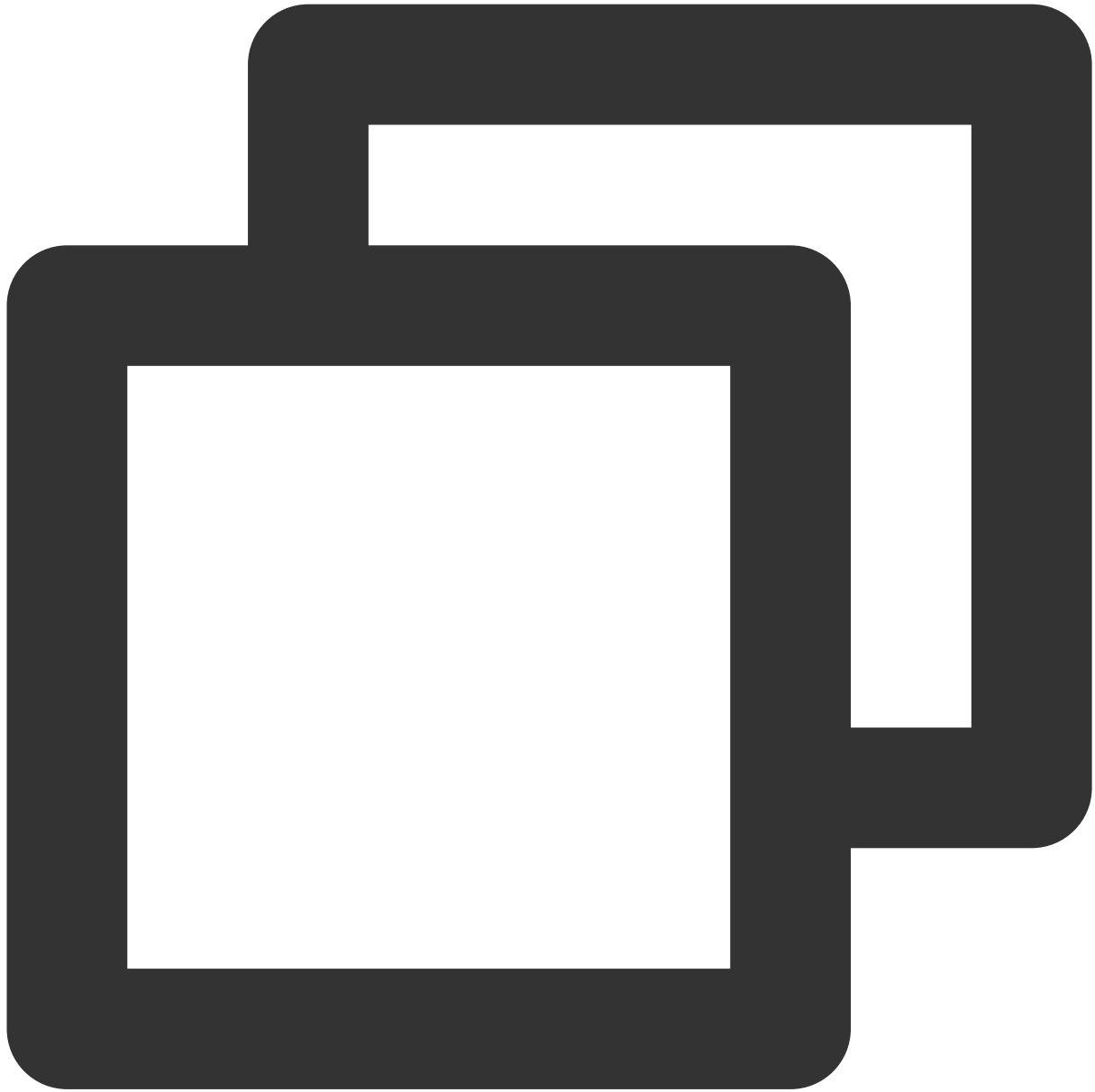
```
- (void)setVideoRenderParams:(NSString *)userId params:(TUIVideoRenderParams *)para
```

Parameter	Type	Description
userId	NSString	The target user ID.
params	<a href="#">TUIVideoRenderParams</a>	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.

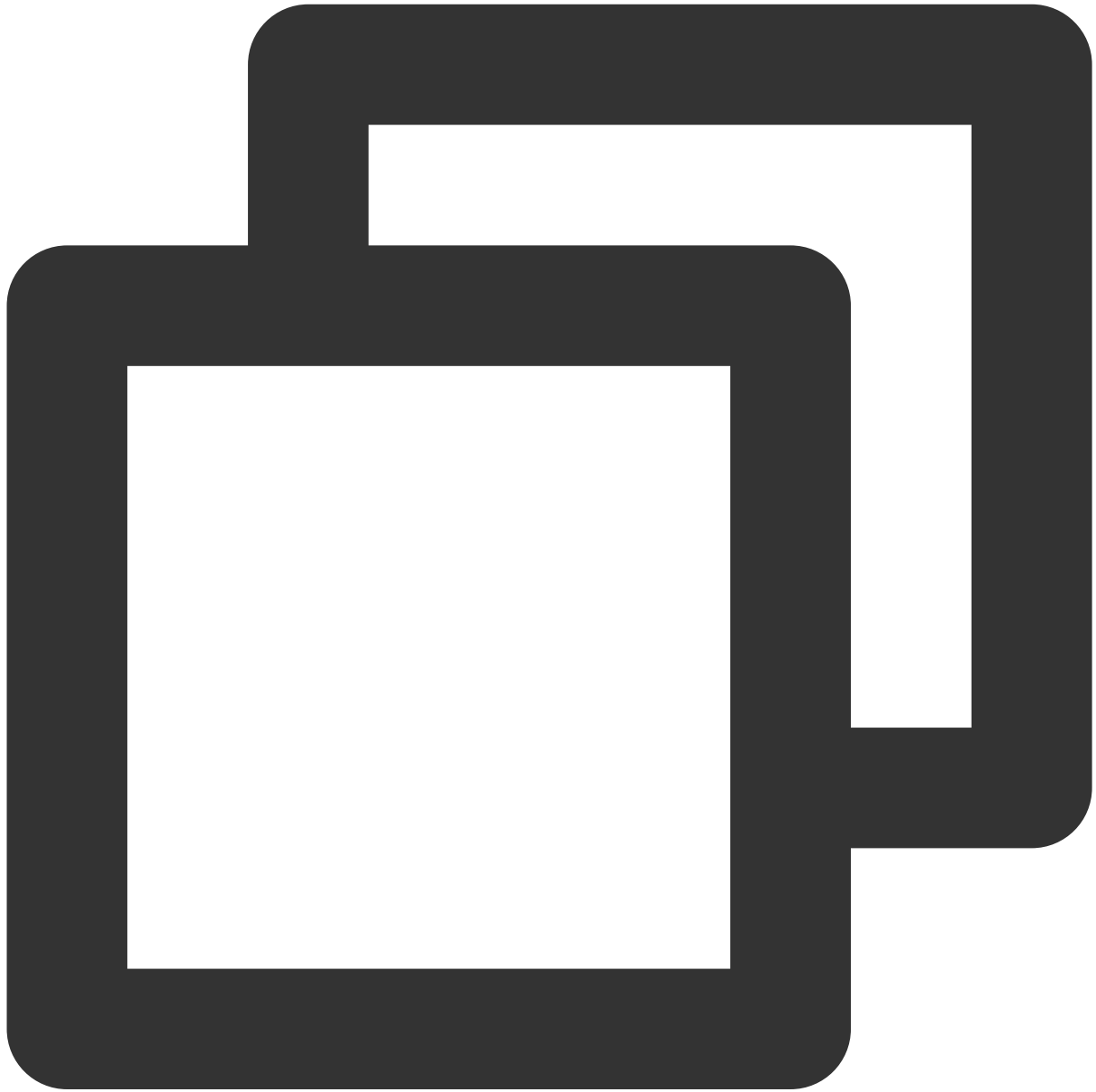


```
- (void)setVideoEncoderParams:(TUIVideoEncoderParams *)params succ:(TUICallSucc) suc
```

Parameter	Type	Description
params	<a href="#">TUIVideoEncoderParams</a>	Video encoding parameters

## getTRTCCloudInstance

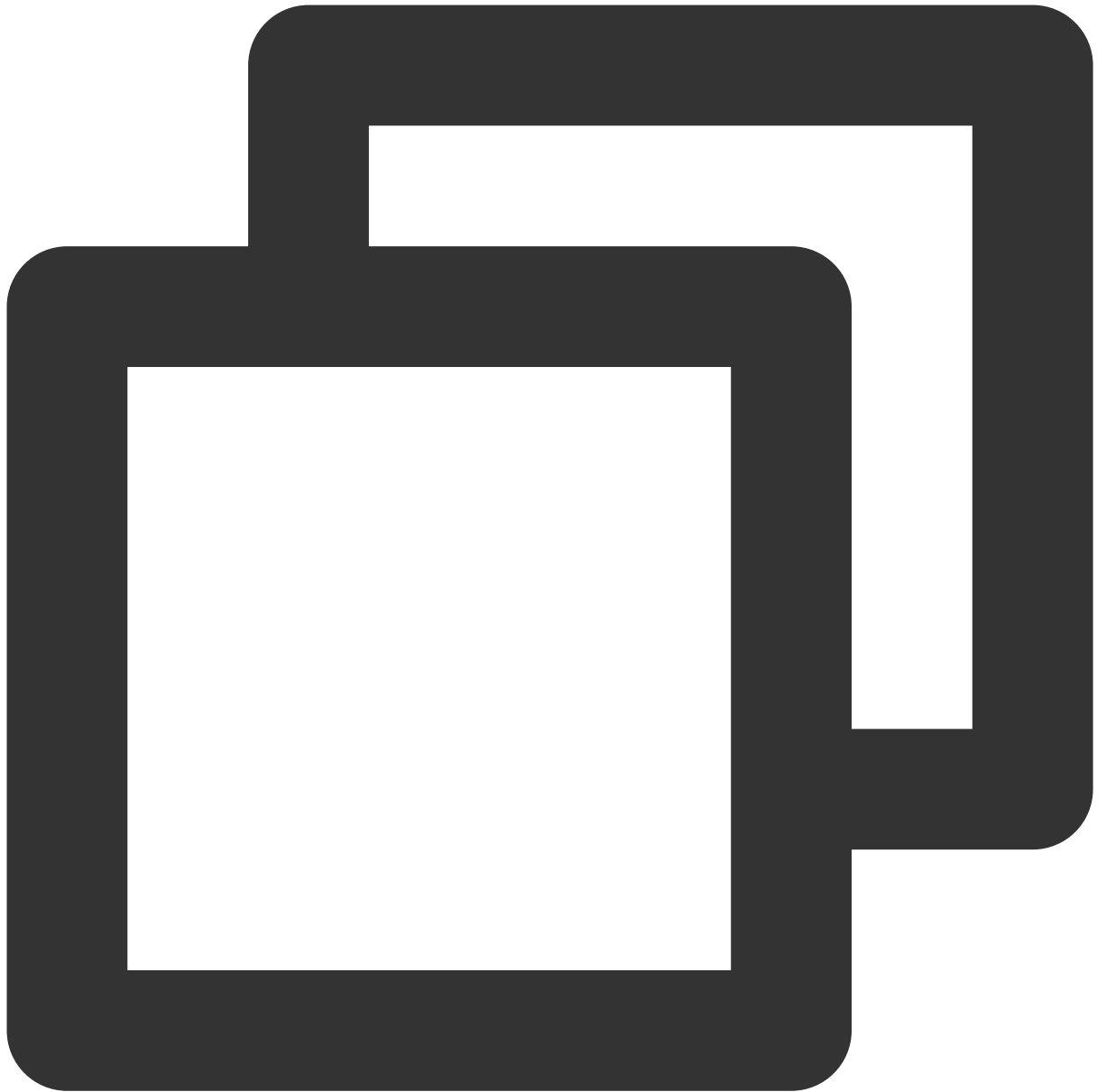
Advanced features.



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

`onUserSigExpired`

The user sig is expired.

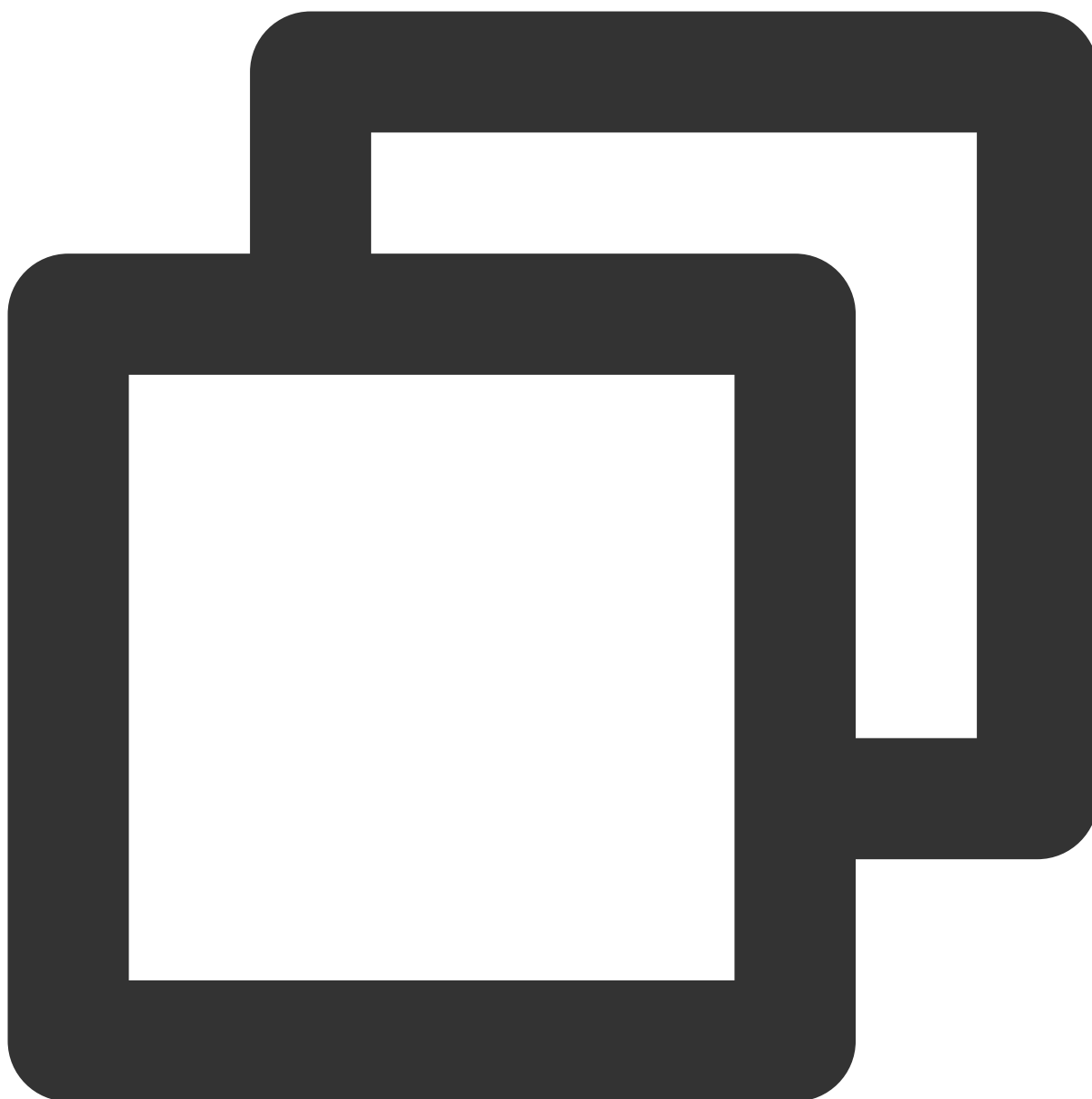
## Details

### **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.



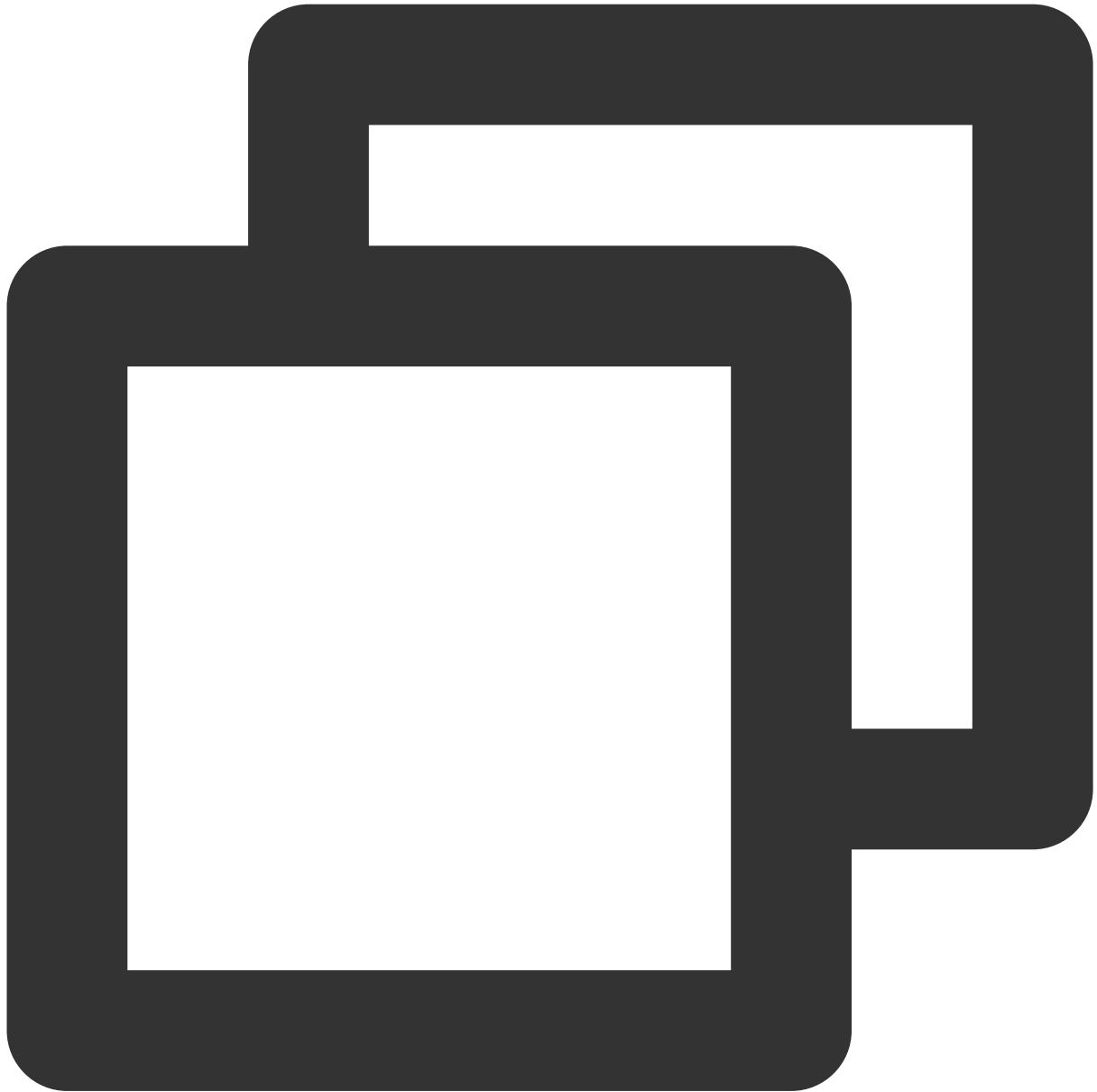
```
void onError(int code, String msg);
```

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	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
userData	NSString	User-added extended fields., Please refer to: <a href="#">TUICallParams</a>

## 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.

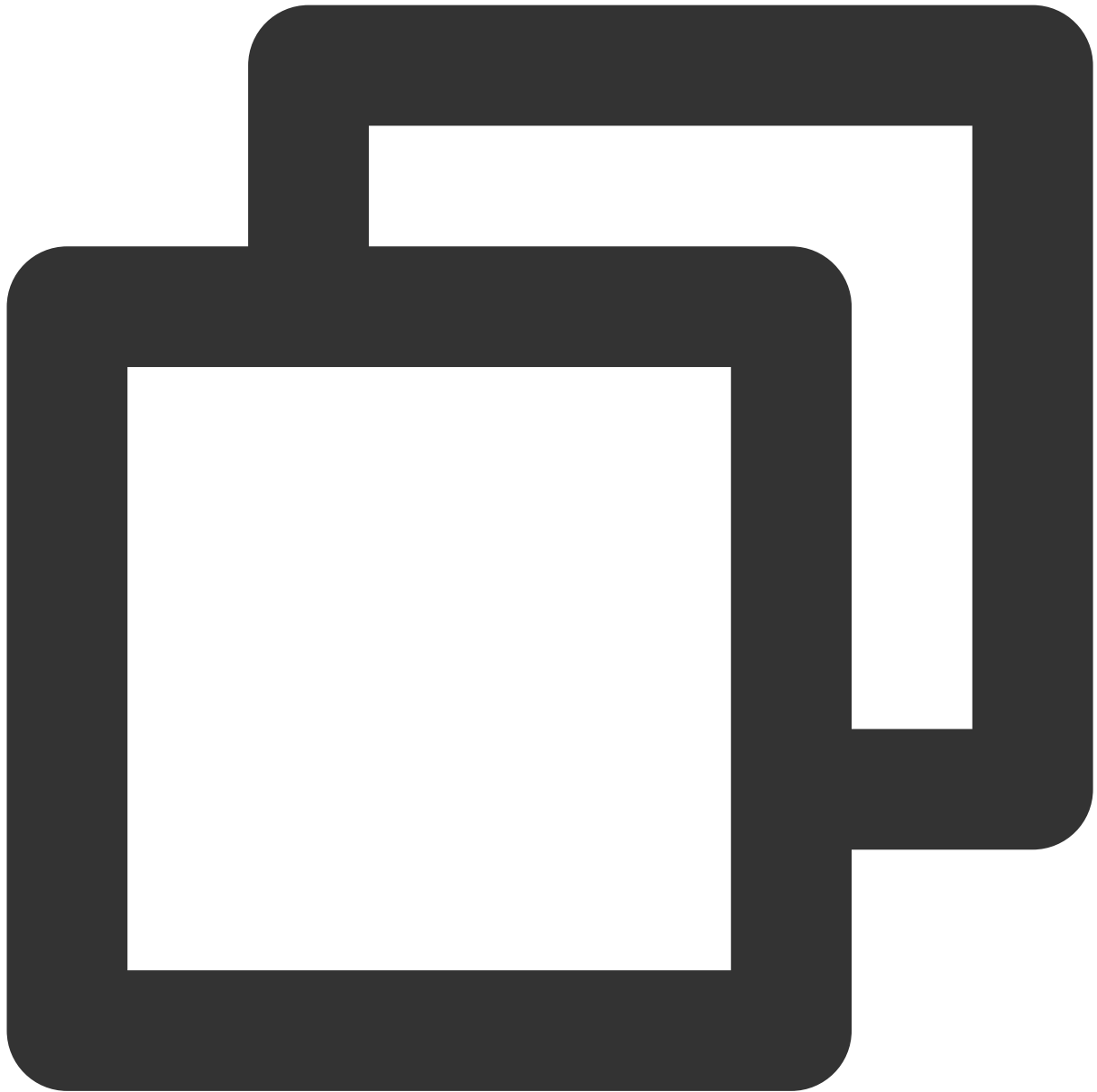
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller)

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).



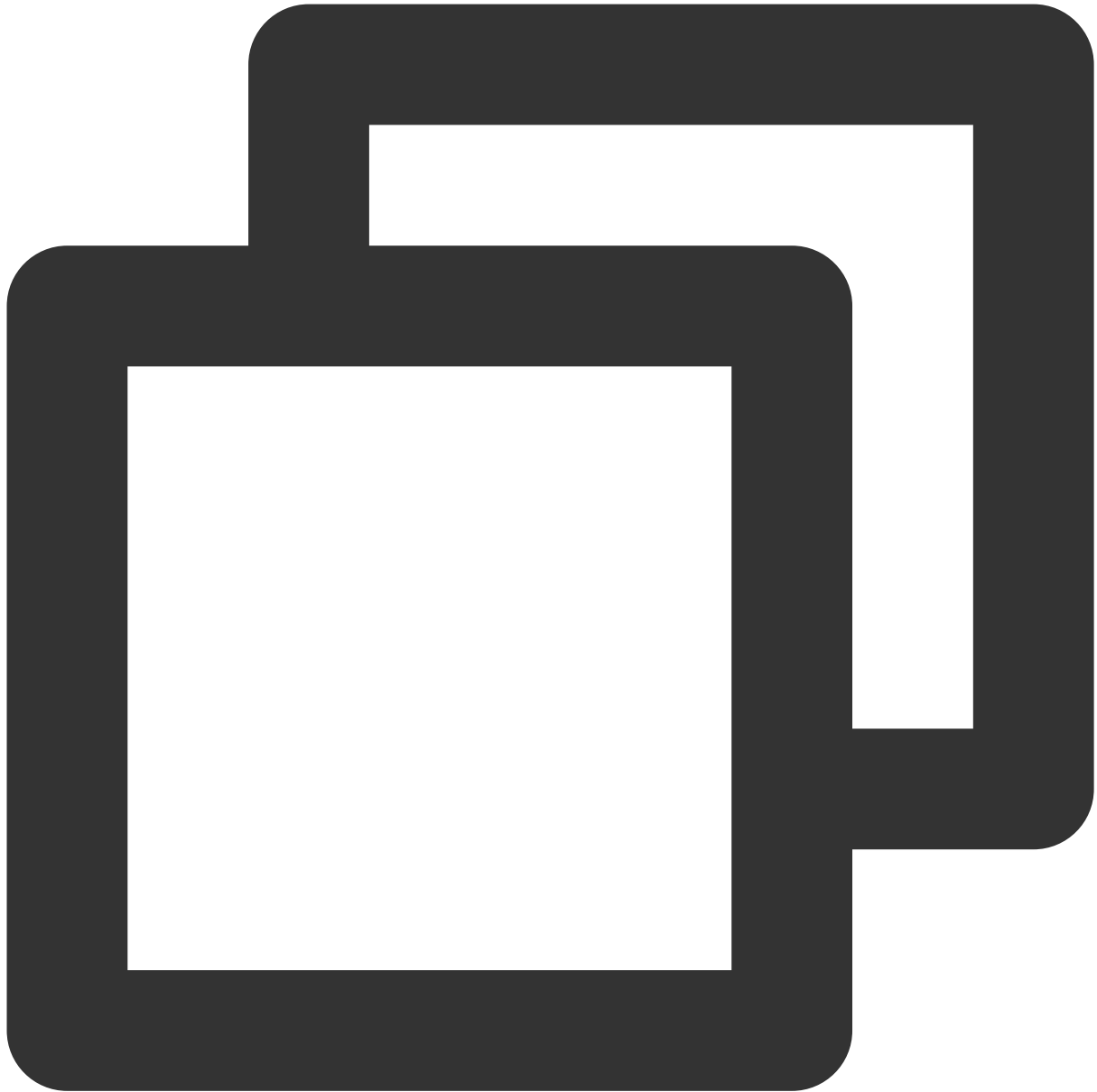
```
void onCallCancelled(String userId);
```

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
```

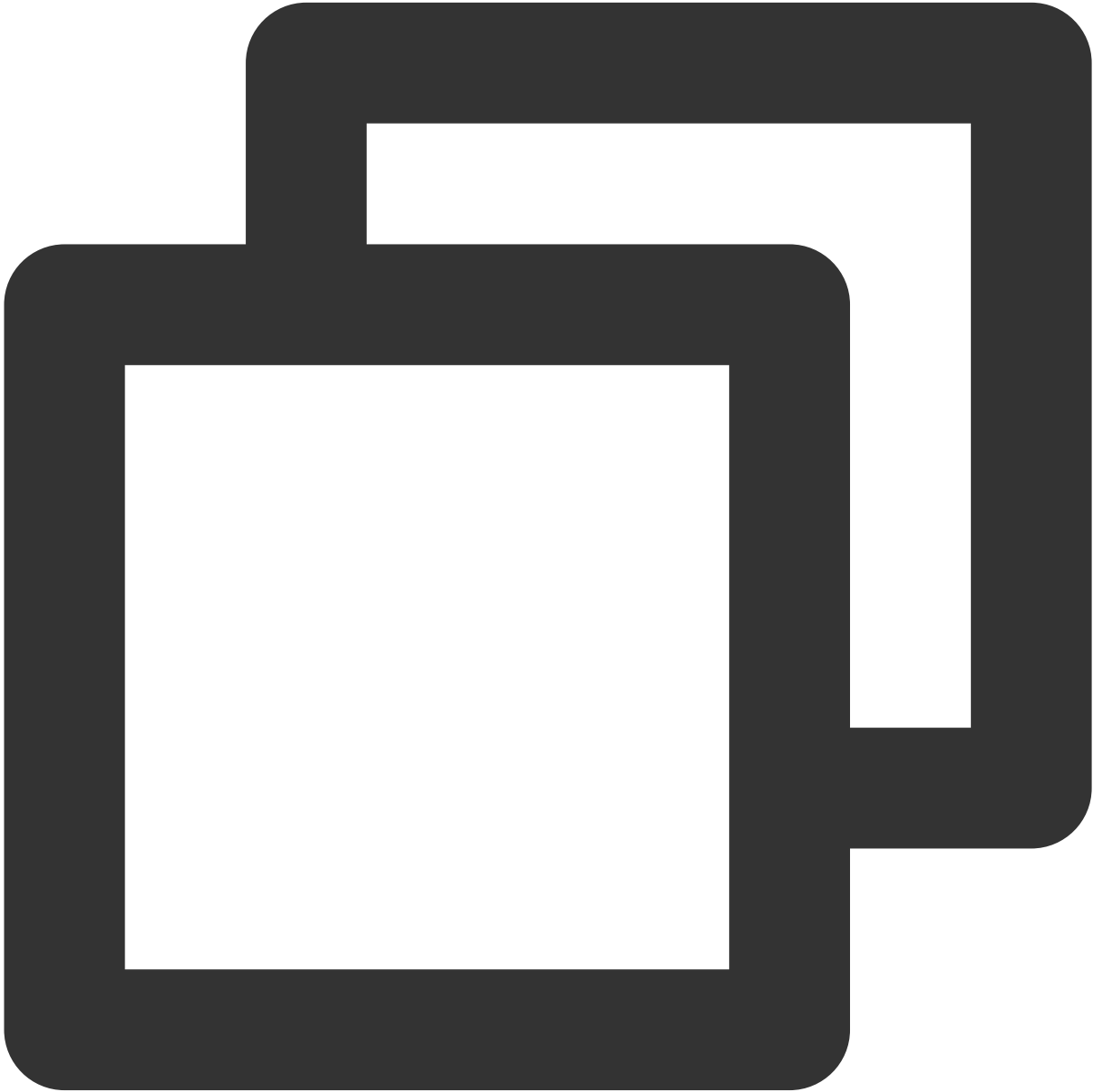
The parameters are described below:

Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

callRole	<a href="#">TUICallRole</a>	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(TUICCommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType
```

The parameters are described below:

--	--	--



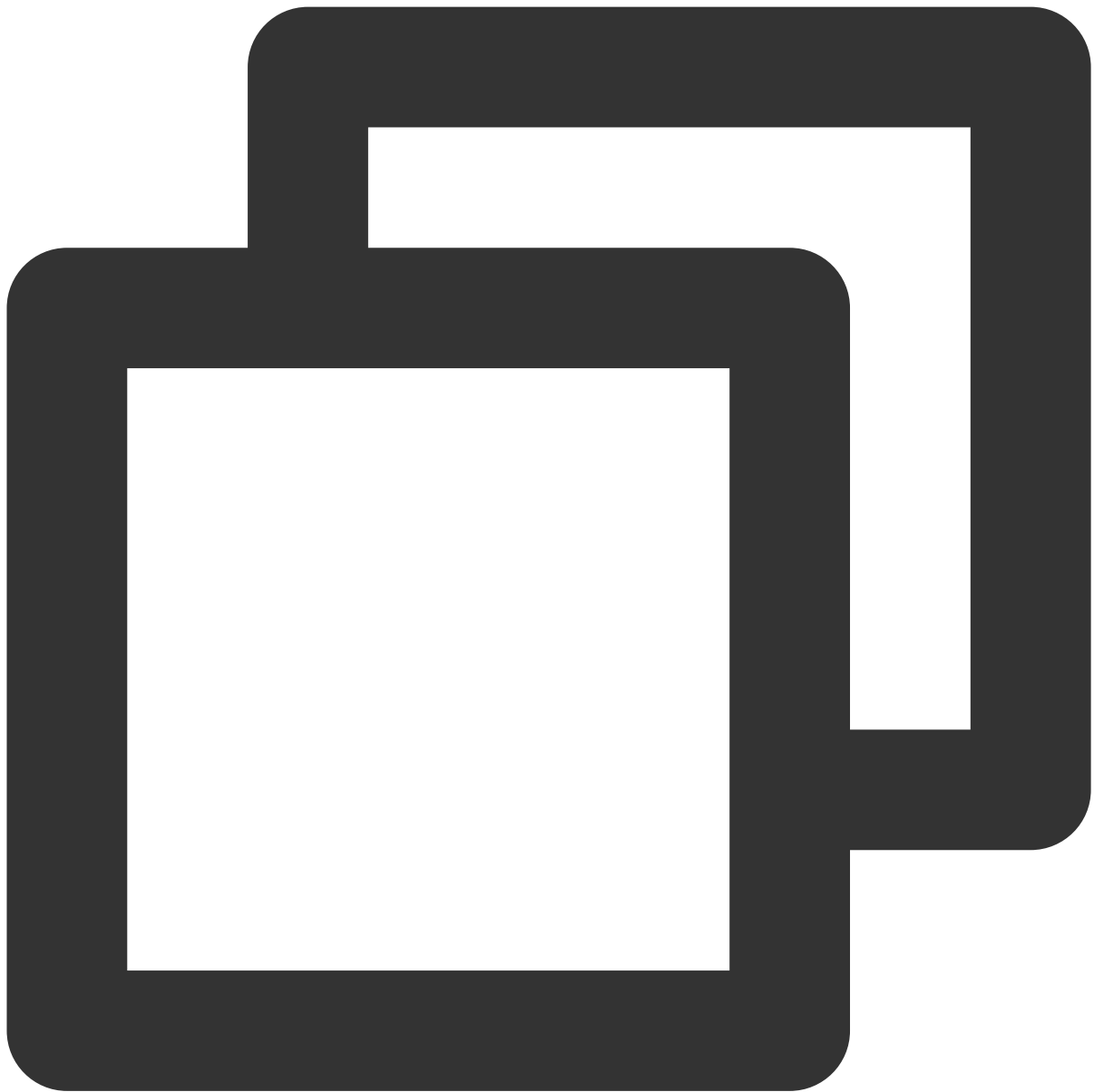
Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
callRole	<a href="#">TUICallRole</a>	The role, which can be caller or callee.
totalTime	Float	The call duration.

**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.



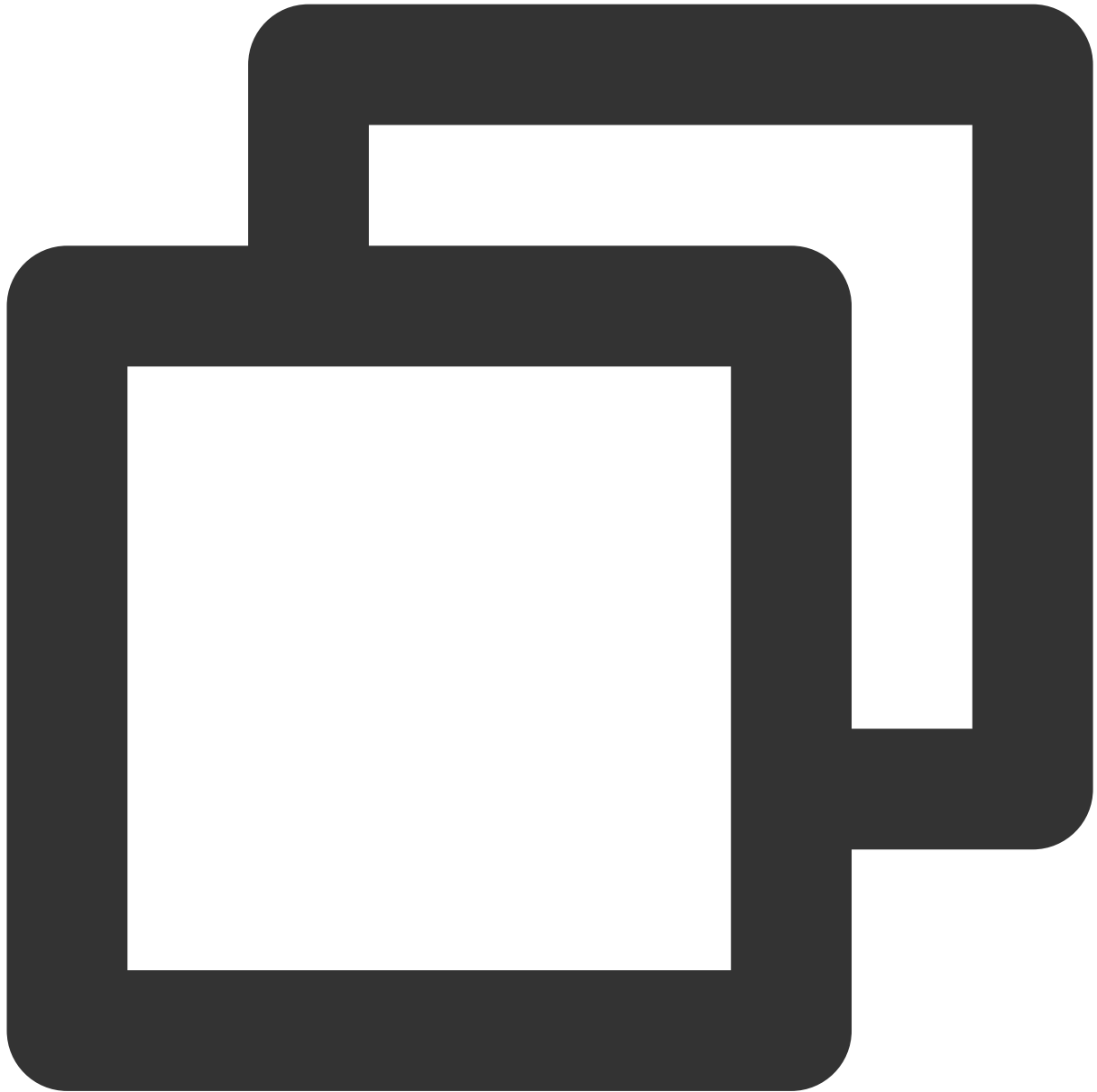
```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType, TUICallDefine.
```

The parameters are described below:

Parameter	Type	Description
oldCallMediaType	<a href="#">TUICallMediaType</a>	The call type before the change.
newCallMediaType	<a href="#">TUICallMediaType</a>	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.



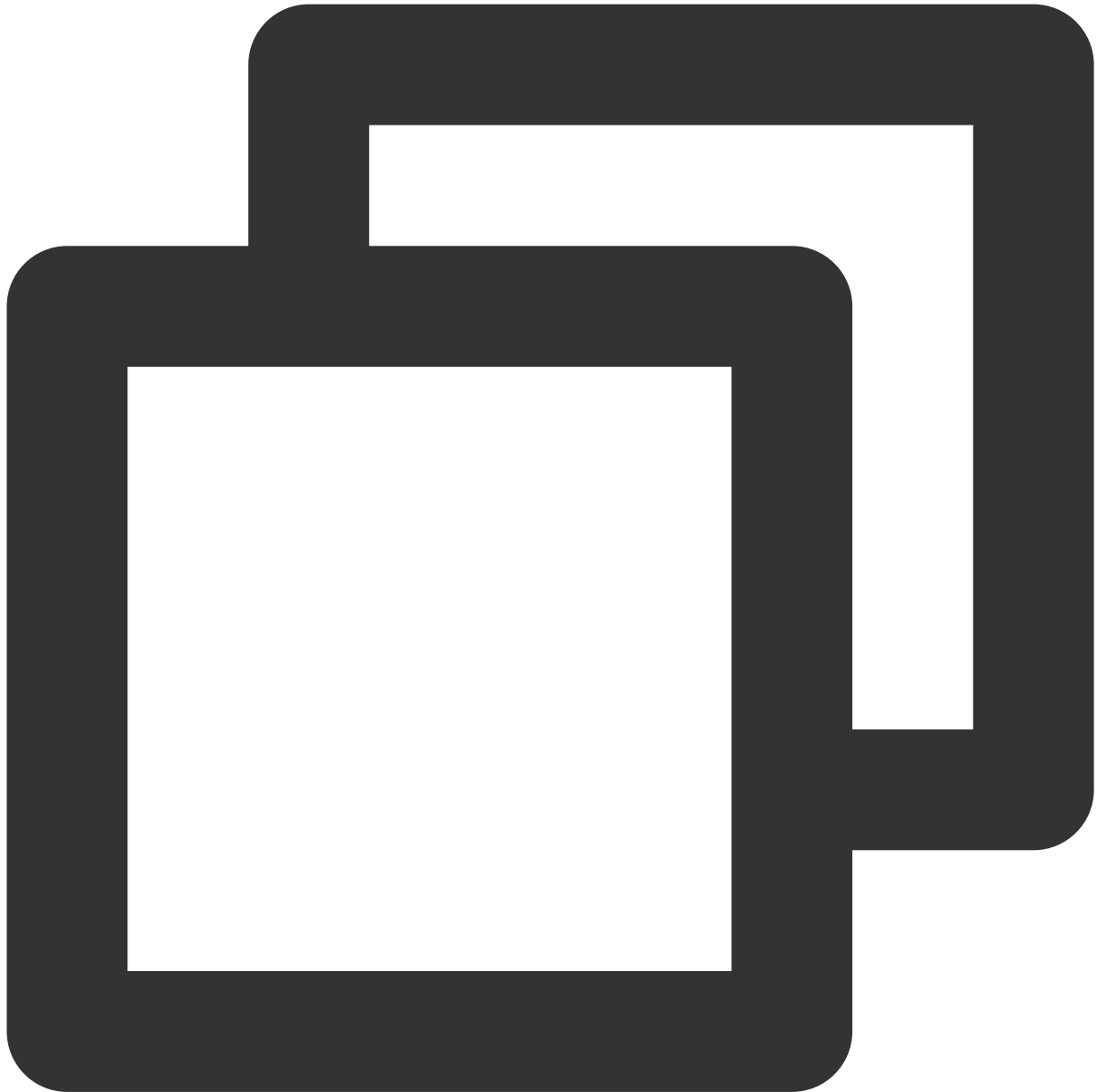
```
void onUserReject(String userId);
```

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.



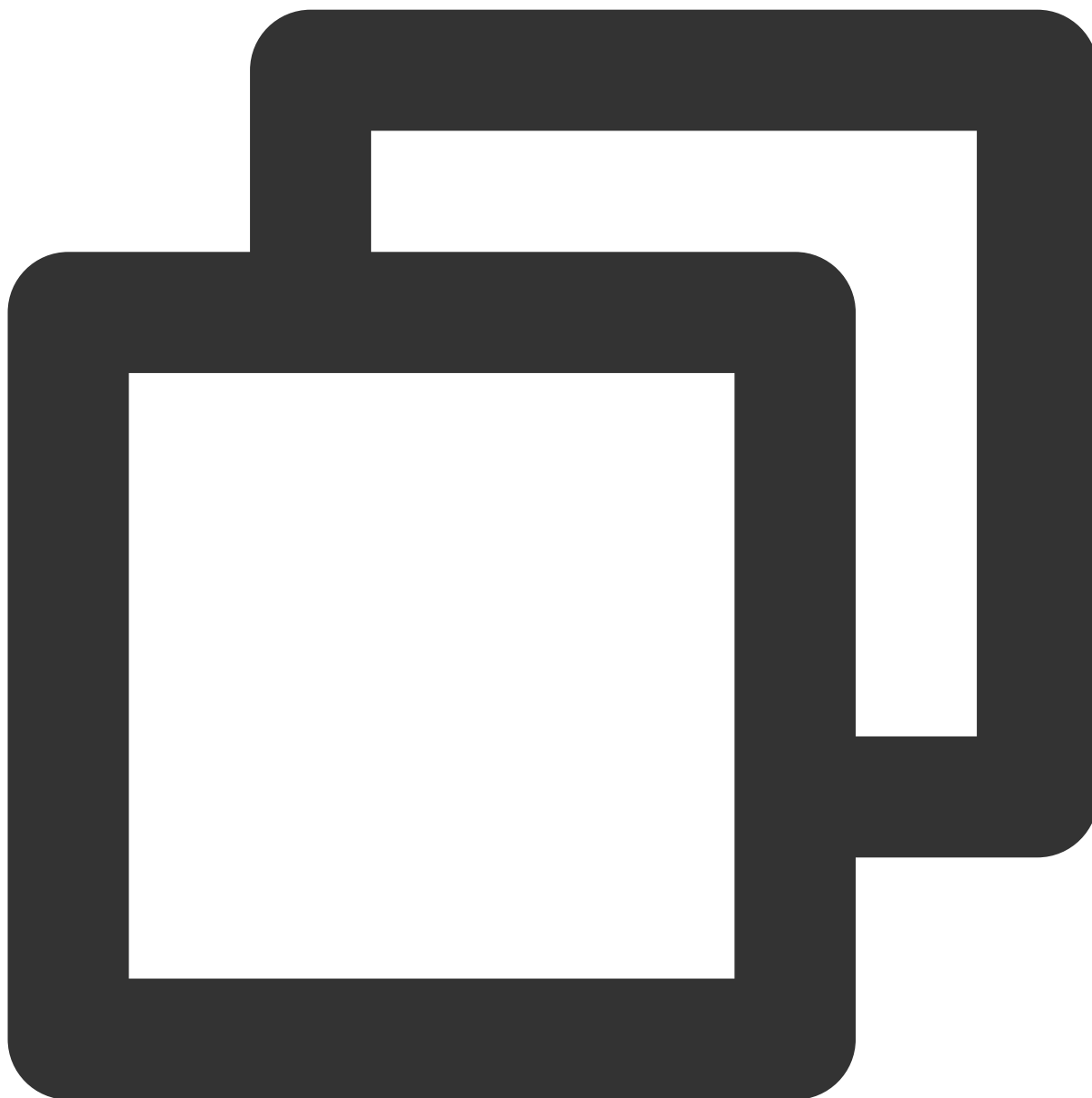
```
void onUserNoResponse (String userId);
```

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.



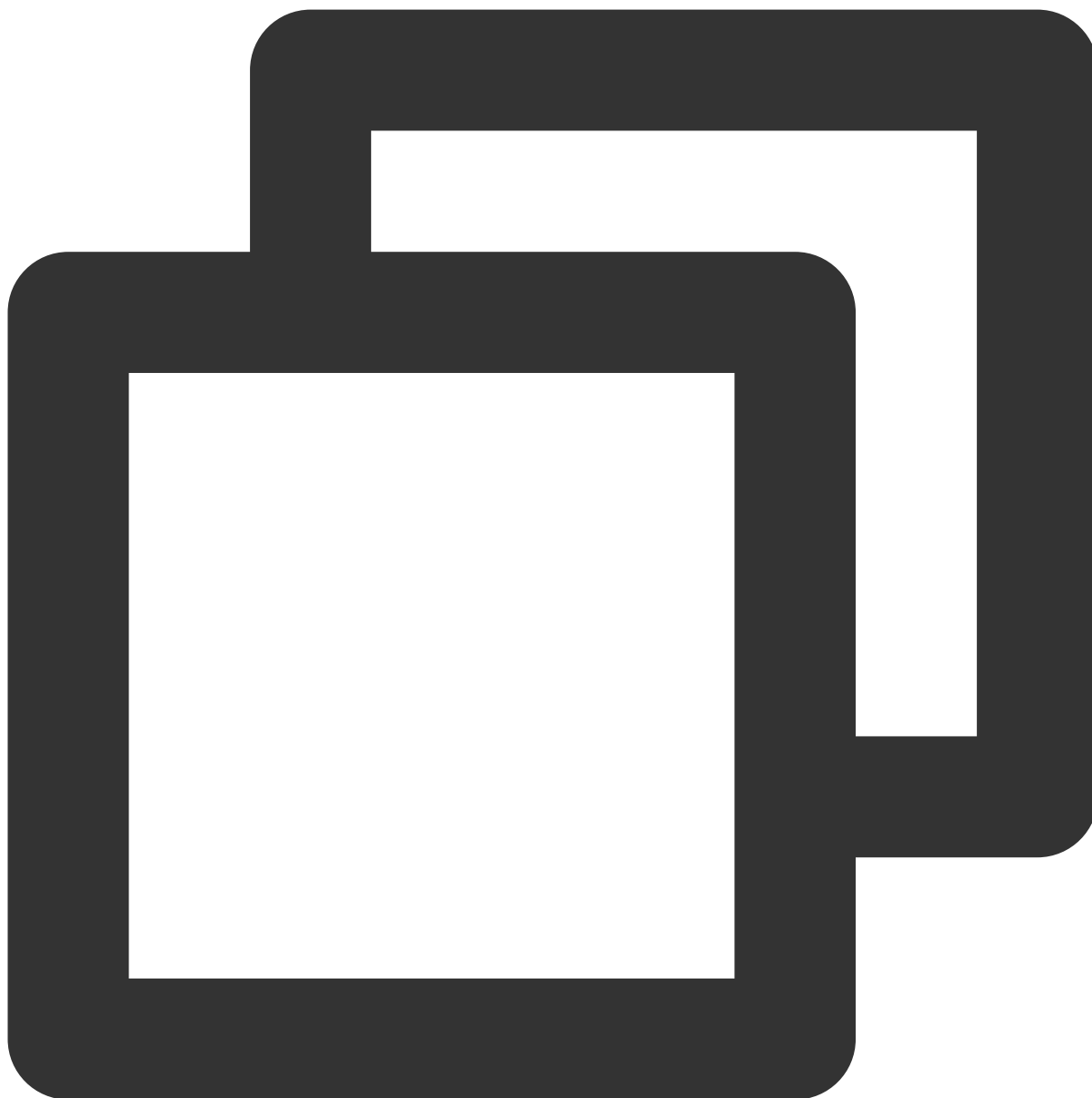
```
void onUserLineBusy(String userId);
```

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.



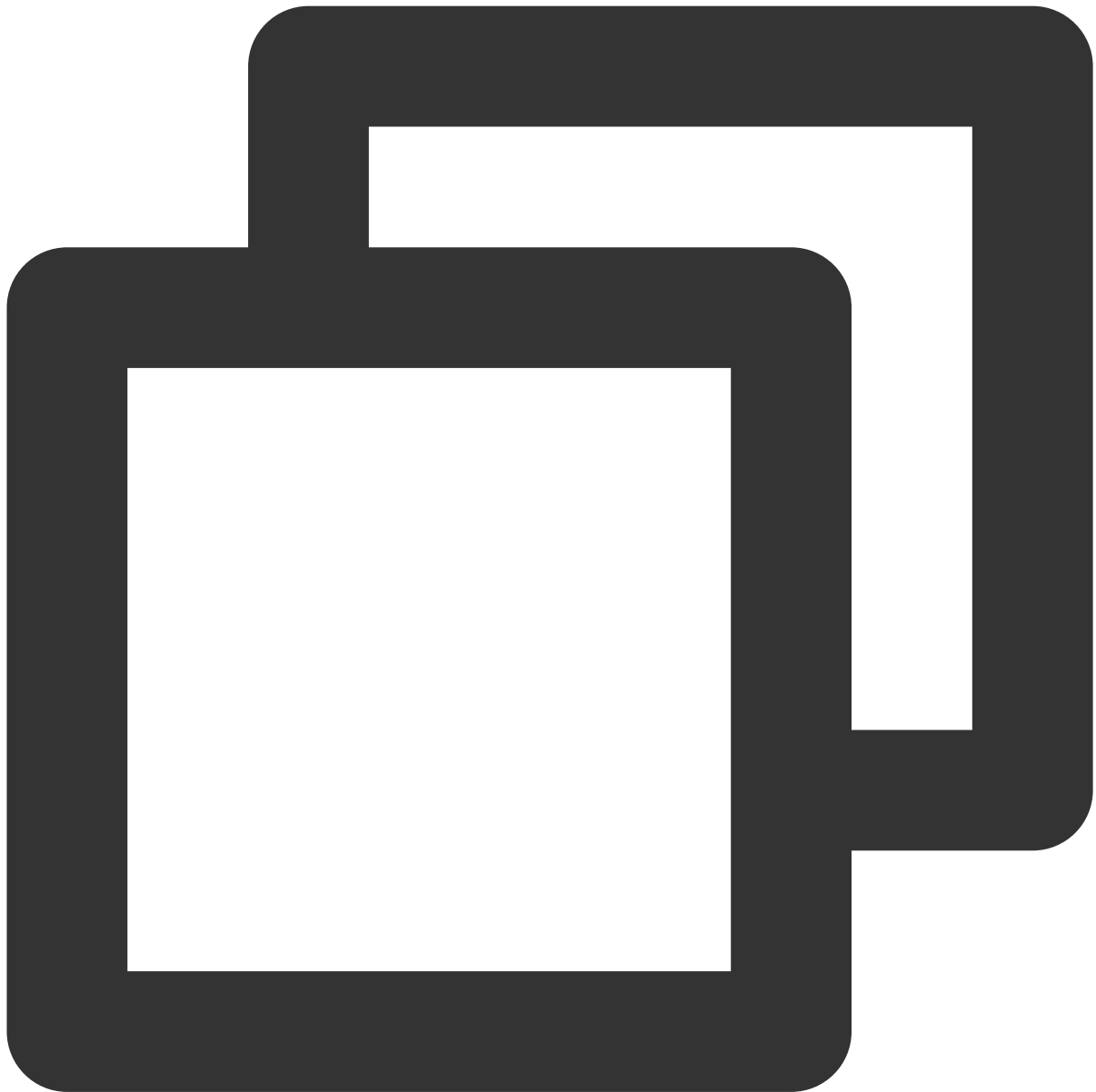
```
void onUserJoin(String userId);
```

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.



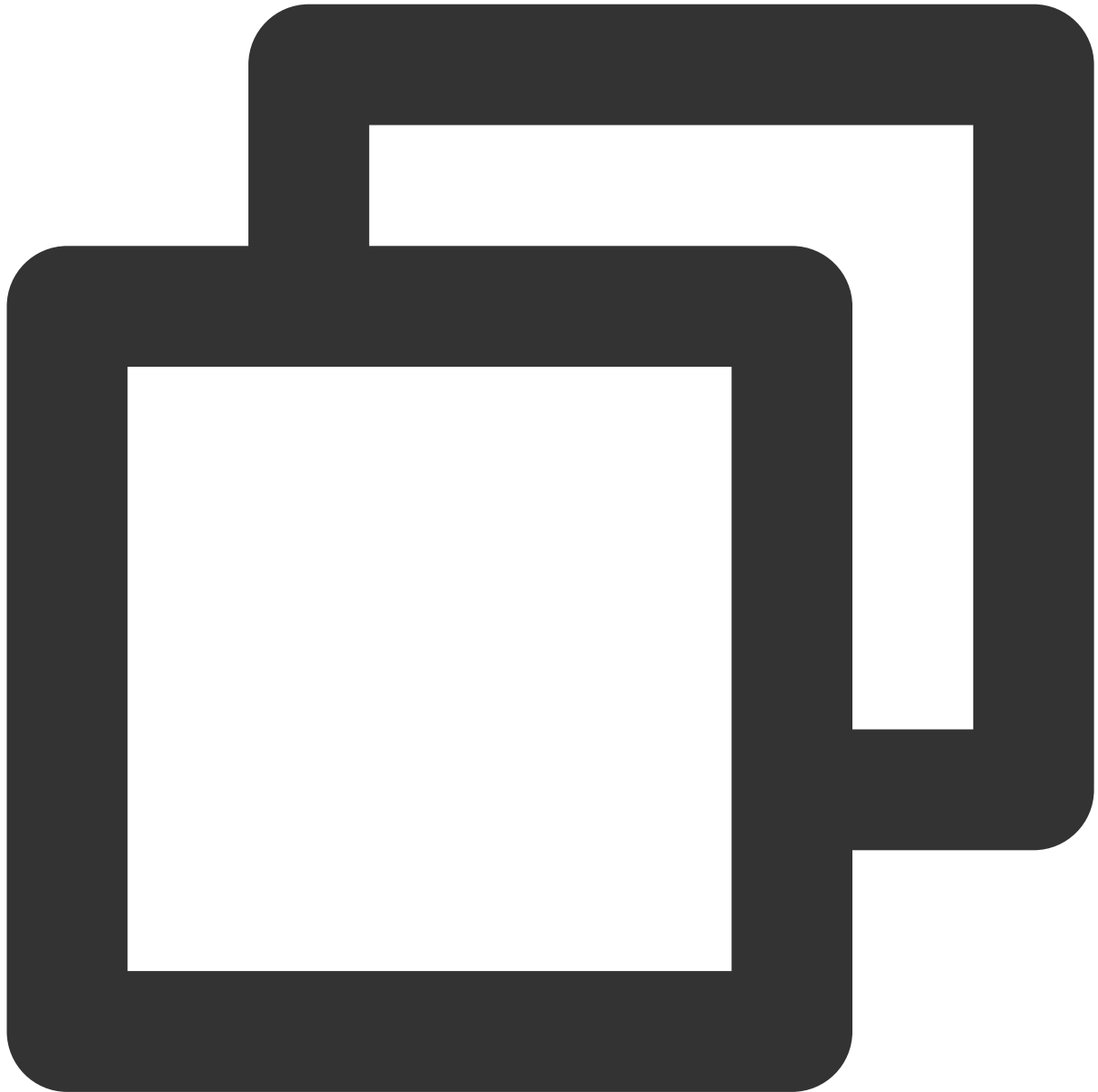
```
void onUserLeave(String userId);
```

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);
```

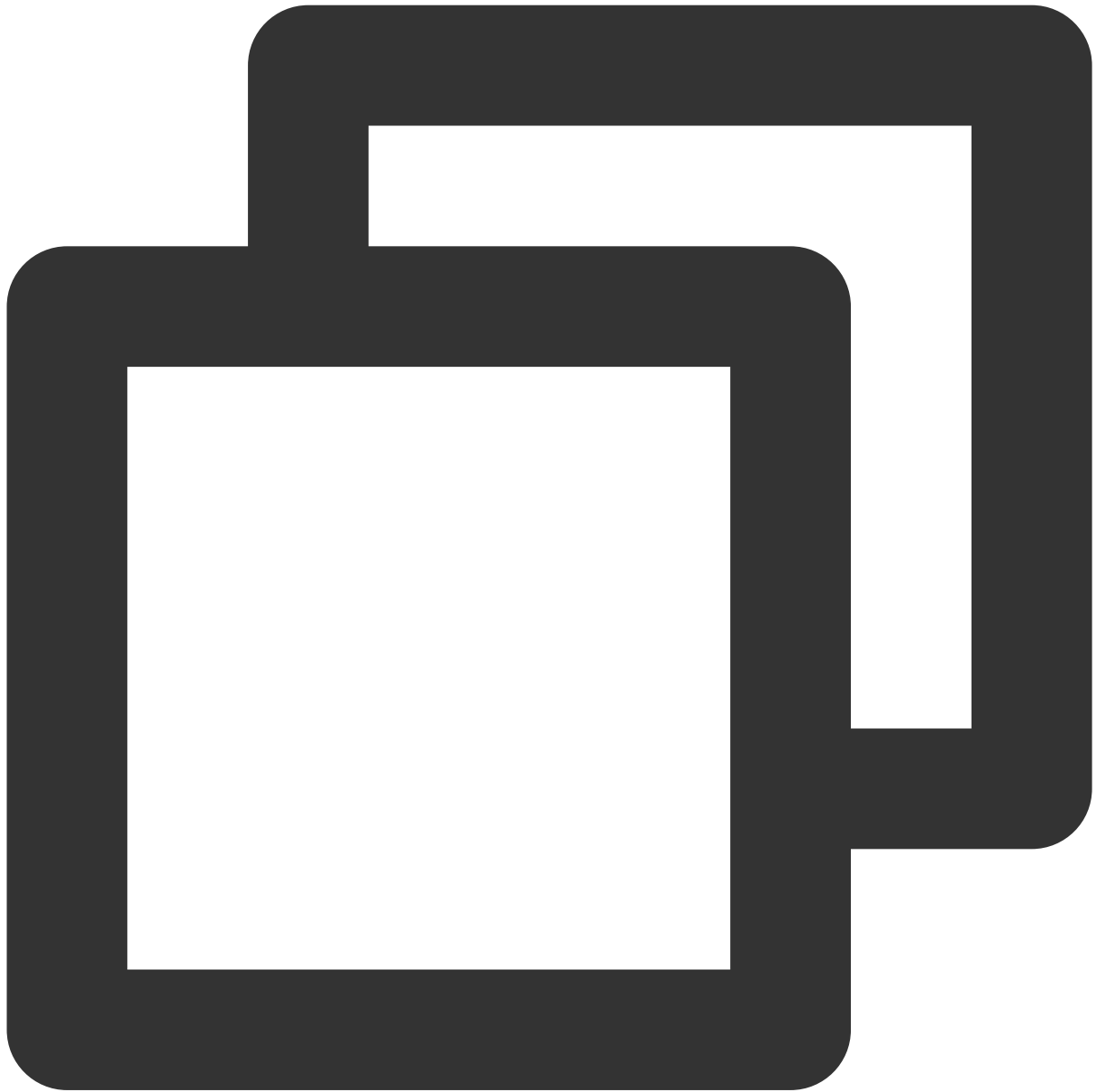
The parameters are described below:

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);
```

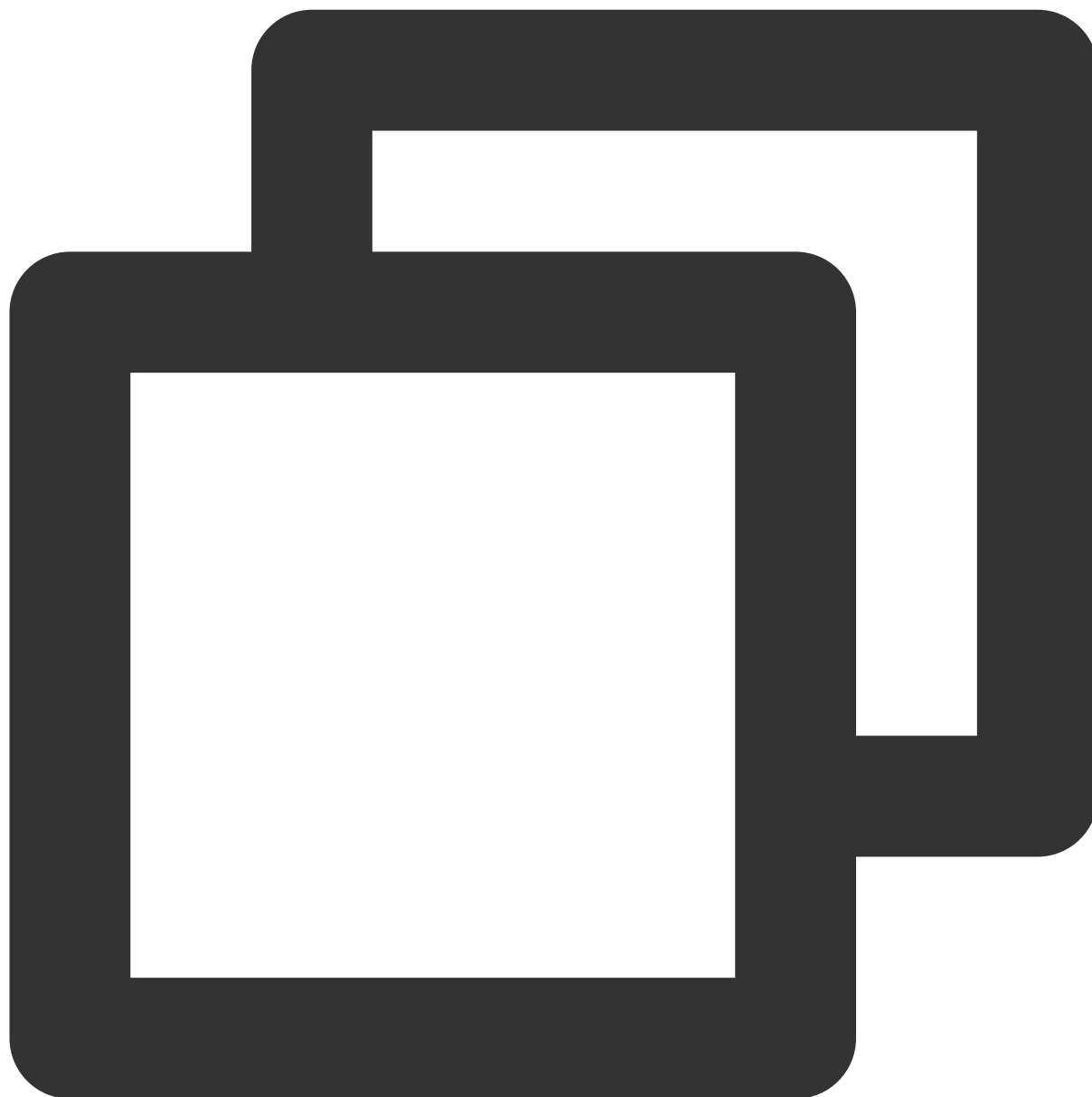
The parameters are described below:

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);
```

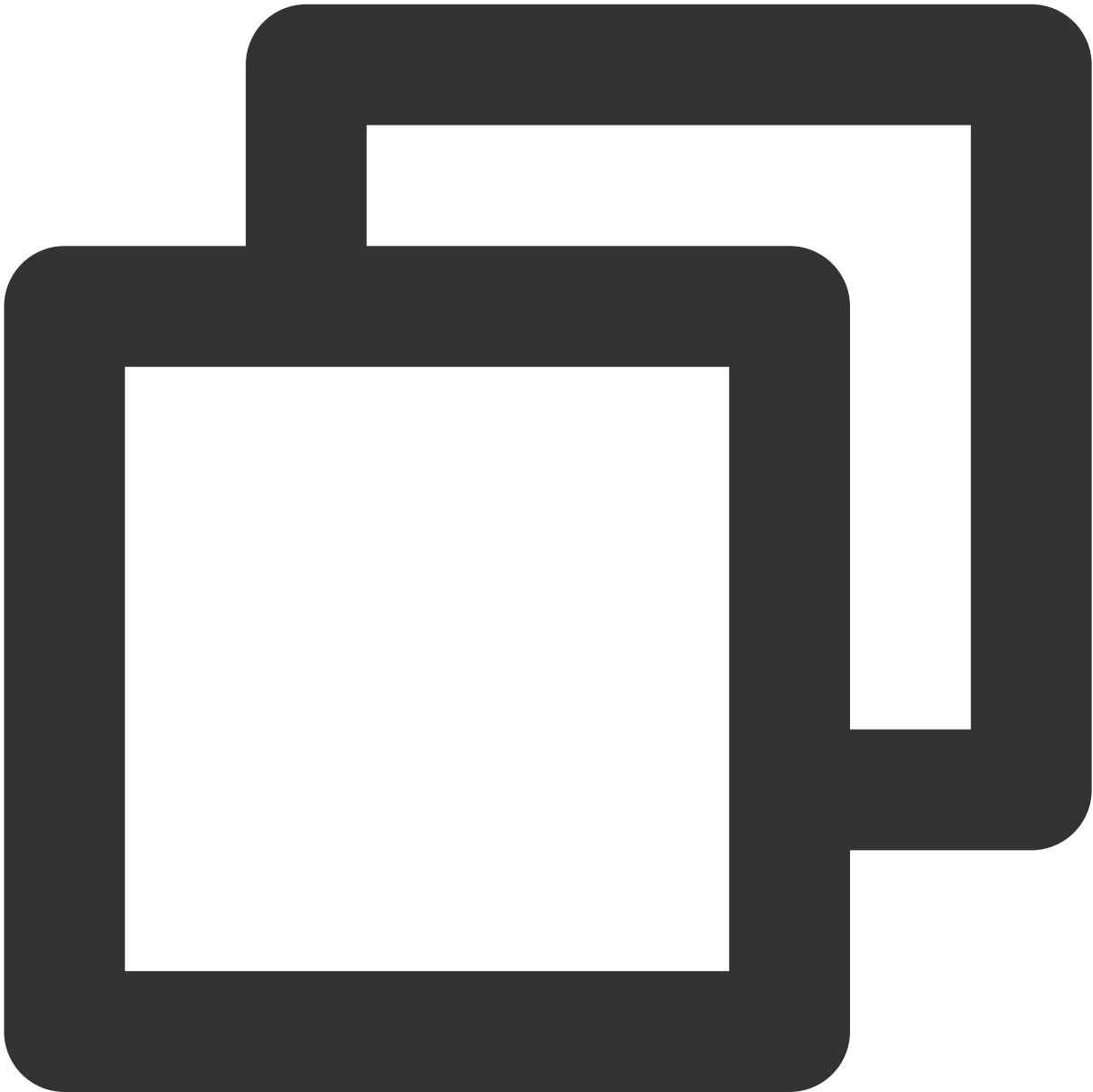
The parameters are described below:

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

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

onUserNetworkQualityChanged

The network quality of all users.



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

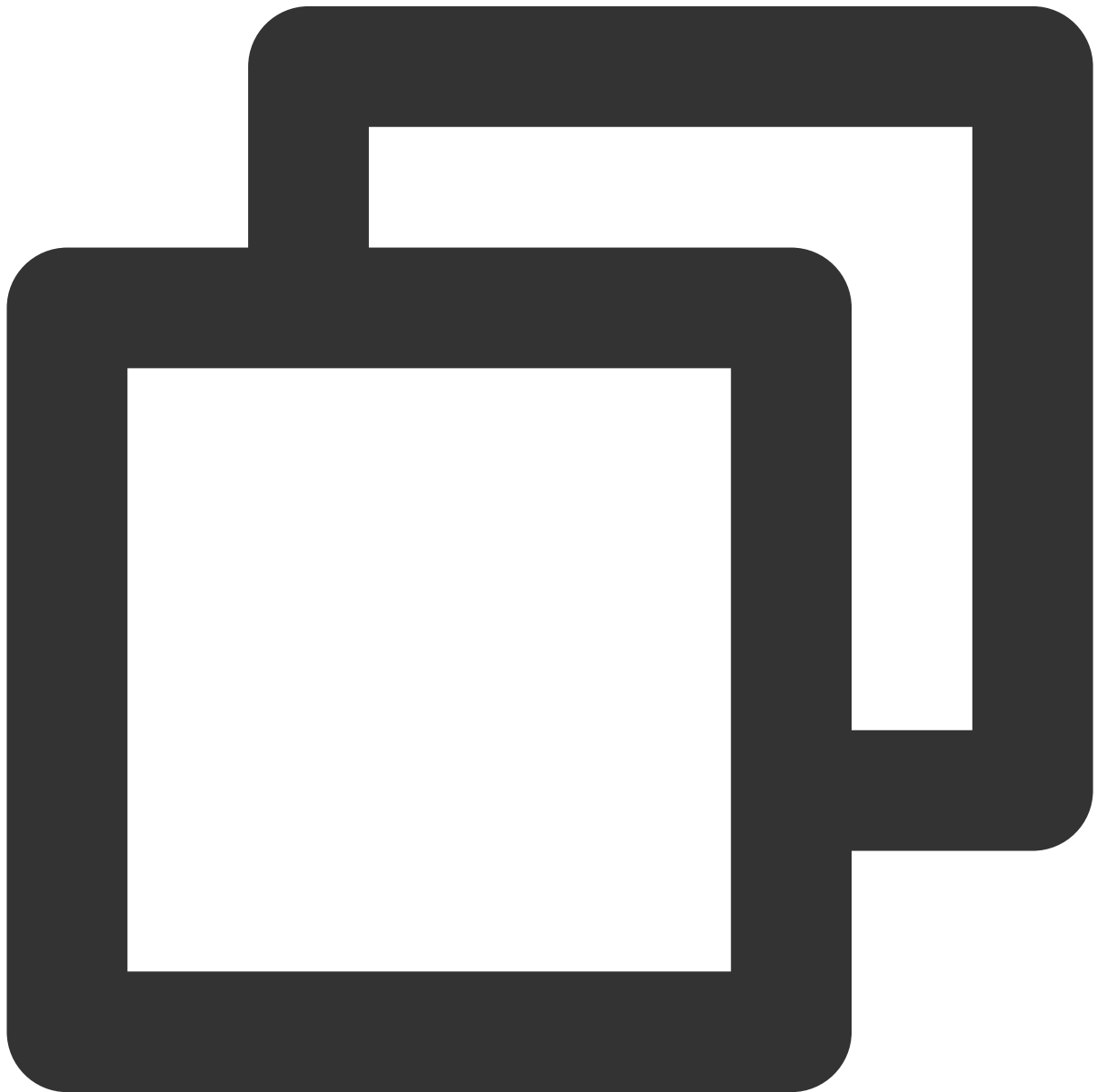
The parameters are described below:

--	--	--

Parameter	Type	Description
networkQualityList	NSArray	The current network conditions for all users ( <code>userId</code> ).

## onKickedOffline

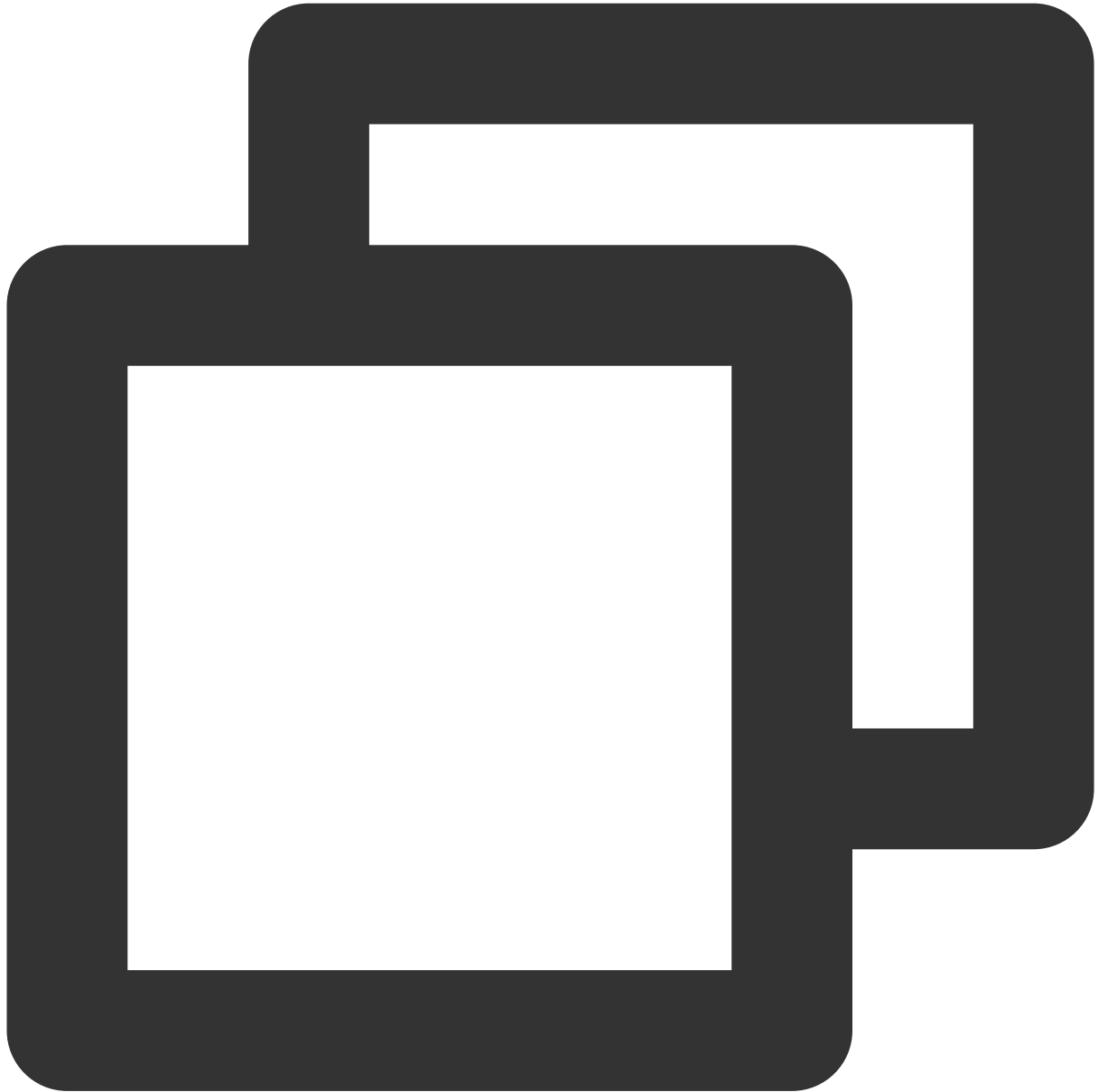
The current user was kicked offline : At this time, you can prompt the user with a UI message and then invoke `init` again.



```
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
<a href="#">TUICallParams</a>	An additional parameter.
<a href="#">TUIOfflinePushInfo</a>	Offline push vendor configuration information.

### TUICommonDefine

Type	Description
<a href="#">TUIRoomId</a>	Room ID for audio and video in a call.
<a href="#">TUINetworkQuality</a>	Network quality information
<a href="#">TUIVideoRenderParams</a>	Video render parameters
<a href="#">TUIVideoEncoderParams</a>	Video encoding parameters

## Enum definition

### TUICallDefine

Type	Description
<a href="#">TUICallMediaType</a>	Media type in a call
<a href="#">TUICallRole</a>	Roles of individuals in a call.
<a href="#">TUICallStatus</a>	The call status
<a href="#">TUICallScene</a>	The call scene
<a href="#">TUICallIOSOfflinePushType</a>	iOS offline push type

### TUICommonDefine

Type	Description

<a href="#">TUIAudioPlaybackDevice</a>	Audio route
<a href="#">TUICamera</a>	Camera type
<a href="#">TUINetworkQuality</a>	Network quality
<a href="#">TUIVideoRenderParamsFillMode</a>	Video image fill mode
<a href="#">TUIVideoRenderParamsRotation</a>	Video image rotation direction
<a href="#">TUIVideoEncoderParamsResolutionMode</a>	Video aspect ratio mode
<a href="#">TUIVideoEncoderParamsResolution</a>	Video resolution

## TUICallParams

Call params

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

## 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 = <b>IOS_OFFLINE_PUSH_NO_SOUND</b> , there will be no sound played when the message is received. When sound = <b>IOS_OFFLINE_PUSH_DEFAULT_SOUND</b> ,

		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: <a href="#">Xiaomi custom ringtones</a> . 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: <a href="#">Huawei message classification standard</a> .
isDisablePush	BOOL	Whether to turn off push notifications (default is on).
iOSPushType	<a href="#">TUICallIOSOfflinePushType</a>	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
<code>intRoomId</code>	UInt32	Numeric room ID. <b>range:</b> 1 - 2147483647( $2^{31}-1$ )
<code>strRoomId</code>	NSString	String room ID. <b>range</b> : Limited to 64 bytes in length. The supported character set range is as follows (Lowercase and uppercase English letters. (a-zA-Z) Number (0-9) Spaces、 <code>!</code> 、 <code>#</code> 、 <code>\$</code> 、 <code>%</code> 、 <code>&amp;</code> 、 <code>(</code> 、 <code>)</code> 、 <code>+</code> 、 <code>-</code> 、 <code>:</code> 、 <code>;</code> 、 <code>&lt;</code> 、

#### 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
<code>fillMode</code>	<a href="#">TUIVideoRenderParamsFillMode</a>	Video image fill mode
<code>rotation</code>	<a href="#">TUIVideoRenderParamsRotation</a>	Video image rotation direction

## TUINetworkQualityInfo

User network quality information

Value	Type	Description
<code>userId</code>	NSString	user ID
<code>quality</code>	<a href="#">NetworkQuality</a>	network quality

## TUIVideoEncoderParams

Video encoding parameters

Value	Type	Description
resolution	<a href="#">TUIVideoEncoderParamsResolution</a>	Video resolution
resolutionMode	<a href="#">TUIVideoEncoderParamsResolutionMode</a>	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 TUIVideoEncoderParamsResolutionModeLand: = 640 × 360
TUIVideoEncoderParamsResolutionModePortrait	1	Portrait resolution, such as : TUIVideoEncoderParamsResolution_640_360 TUIVideoEncoderParamsResolutionModePortr: 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 <a href="#">open the free experience</a> in the console or <a href="#">purchase the official package</a>
ERROR_PACKAGE_NOT_SUPPORTED	-1002	The package you purchased does not support this ability. You can refer to console to purchase <a href="#">Upgrade package</a>
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 <a href="#">init</a> 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 <a href="#">Generating UserSig</a> .
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 reenale 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 reenale 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 : 2024-08-09 17:04:30

### TUICallKit (Includes UI Components)

TUICallKit is an audio and video call component that **includes a UI component**. You can quickly implement a WhatsApp-like audio and video calling scenario with this component.

API	Description
<a href="#">init</a>	Initialize TUICallKit.
<a href="#">call</a>	Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">groupCall</a>	Makes a group call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">joinInGroupCall</a>	Join a group call, <b>v3.1.2+ support</b>
<a href="#">setCallingBell</a>	Customize user's ringtone, <b>v3.0.0+ support</b>
<a href="#">setSelfInfo</a>	Set your own nickname and avatar, <b>v2.2.0+ support</b>
<a href="#">enableMuteMode</a>	Turn on/off ringtone, <b>v3.1.2+ support</b>
<a href="#">enableFloatWindow</a>	Turn on/off the floating window function, <b>v3.1.0+ support</b>
<a href="#">enableVirtualBackground</a>	Turn on/off the blurred background function button, <b>v3.2.4+ support</b>
<a href="#">setLanguage</a>	Set the call language for the TUICallKit component
<a href="#">hideFeatureButton</a>	Hidden Button, <b>v3.2.9+ support</b>
<a href="#">setLocalViewBackgroundImage</a>	Set the background image for the local user's call interface, <b>v3.2.9+ support</b>
<a href="#">setRemoteViewBackgroundImage</a>	Set the background image for the remote user's call interface, <b>v3.2.9+ support</b>
<a href="#">setLayoutMode</a>	Set the call interface layout mode, <b>v3.3.0+ support</b>
<a href="#">setCameraDefaultState</a>	et whether the camera is opened by default, <b>v3.3.0+ support</b>

[destroyed](#)

Destroy TUICallKit

## TUICallEngine (No UI)

TUICallEngine API is an audio and video call component that **offers a No UI interface**. You can use this set of APIs to custom encapsulate according to your business needs.

API	Description
<a href="#">createInstance</a>	Creating a TUICallEngine Instance (Singleton Pattern)
<a href="#">destroyInstance</a>	Terminating a TUICallEngine Instance (Singleton Pattern)
<a href="#">on</a>	Listening on events
<a href="#">off</a>	Canceling Event Listening
<a href="#">login</a>	Sign in Interface
<a href="#">logout</a>	Logout Interface
<a href="#">setSelfInfo</a>	Configure the user's nickname and profile photo
<a href="#">call</a>	Initiate a one-on-one call
<a href="#">groupCall</a>	Group Chat Invitation Call
<a href="#">accept</a>	Answer Calls
<a href="#">reject</a>	Decline Call
<a href="#">hangup</a>	End Calls
<a href="#">switchCallMediaType</a>	Switch Audio and Video Calls
<a href="#">startRemoteView</a>	Initiate Remote Screen Rendering
<a href="#">stopRemoteView</a>	Stop Remote Screen Rendering
<a href="#">startLocalView</a>	Start Local Screen Rendering, <b>Note: This will be deprecated; use openCamera instead</b>
<a href="#">stopLocalView</a>	Stop Local Screen Rendering, <b>Note: This will be deprecated; use closeCamera instead</b>
<a href="#">openCamera</a>	Enable the camera

<a href="#">closeCamara</a>	Turn Off Camera
<a href="#">switchCamera</a>	Switch between front and rear cameras, note: only supported on mobile devices. <b>v3.0.0+ supported</b>
<a href="#">openMicrophone</a>	Enable Microphone
<a href="#">closeMicrophone</a>	Turn off the microphone
<a href="#">setVideoQuality</a>	Set video quality
<a href="#">getDeviceList</a>	Access device list
<a href="#">switchDevice</a>	Switch camera or microphone devices
<a href="#">enableAIVoice</a>	Enable/disable AI noise reduction
<a href="#">enableMultiDeviceAbility</a>	Turn on/off the multi-device login mode of TUICallEngine. <b>v2.1.1+ supported</b>
<a href="#">setBlurBackground</a>	Switch/set background blur, <b>v3.0.6+ supported</b>
<a href="#">setVirtualBackground</a>	Switch/set image background blur, <b>v3.0.6+ supported</b>

## Event Types

TUICallEvent is the callback event class corresponding to TUICallEngine. Through this callback, you can listen to the callback events of interest.

EVENT	Description
<a href="#">TUICallEvent.ERROR</a>	An error occurred during the call.
<a href="#">TUICallEvent.SDK_READY</a>	This event is received when the SDK enters the ready state
<a href="#">TUICallEvent.KICKED_OUT</a>	Receiving this event after a duplicate sign-in indicates that the user has been removed from the room
<a href="#">TUICallEvent.USER_ACCEPT</a>	If a user answers, this event will be received
<a href="#">TUICallEvent.USER_ENTER</a>	A user joined the call.
<a href="#">TUICallEvent.USER_LEAVE</a>	A user left the call.
<a href="#">TUICallEvent.REJECT</a>	A user declined the call.

<a href="#">TUICallEvent.NO_RESP</a>	A user didn't respond.
<a href="#">TUICallEvent.LINE_BUSY</a>	A user was busy.
<a href="#">TUICallEvent.USER_VIDEO_AVAILABLE</a>	Whether a user has a video stream.
<a href="#">TUICallEvent.USER_AUDIO_AVAILABLE</a>	Whether a user has an audio stream.
<a href="#">TUICallEvent.USER_VOICE_VOLUME</a>	The volume levels of all users.
<a href="#">TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE</a>	Group Chat Update, Invitation List this callback will be received
<a href="#">TUICallEvent.ON_CALL_BEGIN</a>	Call connected event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.INVITED</a>	A call was received. <b>It will be discarded later and it is recommended to use TUICallEvent.ON_CALL_RECEIVED</b>
<a href="#">TUICallEvent.ON_CALL_RECEIVED</a>	Call request event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.CALLING_CANCEL</a>	Call cancellation event, <b>It will be abandoned later and it is recommended to use TUICallEvent.ON_CALL_CANCELED</b>
<a href="#">TUICallEvent.ON_CALL_CANCELED</a>	Call cancellation event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.ON_CALL_BEGIN</a>	Call connected event.
<a href="#">TUICallEvent.CALLING_END</a>	The call ended.
<a href="#">TUICallEvent.DEVICED_UPDATED</a>	Device list update, this event will be received
<a href="#">TUICallEvent.CALL_TYPE_CHANGED</a>	Call type switching, this event will be received
<a href="#">TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED</a>	All user network quality events, <b>v3.0.7+ supported</b>

## Document Link

[TUICallEngine](#)

[TUICallEvent](#)

# TUICallKit

Last updated : 2024-08-09 17:38:19

## API Introduction

The TUICallKit API is the **audio and video call component that includes a UI interface**. With the TUICallKit API, you can swiftly develop audio and video call scenarios reminiscent of WeChat through simple interfaces. For further comprehensive steps to access this, please refer to: [Integrating TUICallKit](#).

## API Overview

TUICallKit is the **audio and video call component with a UI**, which enables you to swiftly create scenarios akin to WeChat for voice and video calls.

<TUICallKit/>: The core UI call component.

TUICallKitServer is the call instance, offering the following API interfaces.

API	Description
<a href="#">init</a>	Initialize TUICallKit.
<a href="#">call</a>	Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">groupCall</a>	Makes a group call, Support for custom room ID, call timeout, offline push content, etc
<a href="#">joinInGroupCall</a>	Join a group call, <b>v3.1.2+ support</b>
<a href="#">setCallingBell</a>	Customize user's ringtone, <b>v3.0.0+ support</b>
<a href="#">setSelfInfo</a>	Set your own nickname and avatar, <b>v2.2.0+ support</b>
<a href="#">enableMuteMode</a>	Turn on/off ringtone, <b>v3.1.2+ support</b>
<a href="#">enableFloatWindow</a>	Turn on/off the floating window function, <b>v3.1.0+ support</b>
<a href="#">enableVirtualBackground</a>	Turn on/off the blurred background function button. <b>v3.2.4+ support</b>
<a href="#">setLanguage</a>	Set the call language for the TUICallKit component
<a href="#">hideFeatureButton</a>	Hidden Button, <b>v3.2.9+ support</b>

<a href="#">setLocalViewBackgroundImage</a>	Set the background image for the local user's call interface, <b>v3.2.9+ support</b>
<a href="#">setRemoteViewBackgroundImage</a>	Set the background image for the remote user's call interface, <b>v3.2.9+ support</b>
<a href="#">setLayoutMode</a>	Set the call interface layout mode, <b>v3.3.0+ support</b>
<a href="#">setCameraDefaultState</a>	et whether the camera is opened by default, <b>v3.3.0+ support</b>
<a href="#">destroyed</a>	Destroy TUICallKit

## Introduction to `<TUICallKit/>` attributes

### Attribute Overview

Attribute	Description	Type	Required	Default Value
<code>allowedMinimized</code>	Is the floating window permitted?	boolean	No	false
<code>allowedFullScreen</code>	Whether to permit full screen mode for the call interface	boolean	No	true
<a href="#">videoDisplayMode</a>	Display mode for the call interface	VideoDisplayMode	No	VideoDisplayMode.COVER
<a href="#">videoResolution</a>	Call Resolution	VideoResolution	No	VideoResolution.RESOLUTION_48
<code>beforeCalling</code>	This function will be executed prior to making a call and before receiving an	function(type, error)	No	-

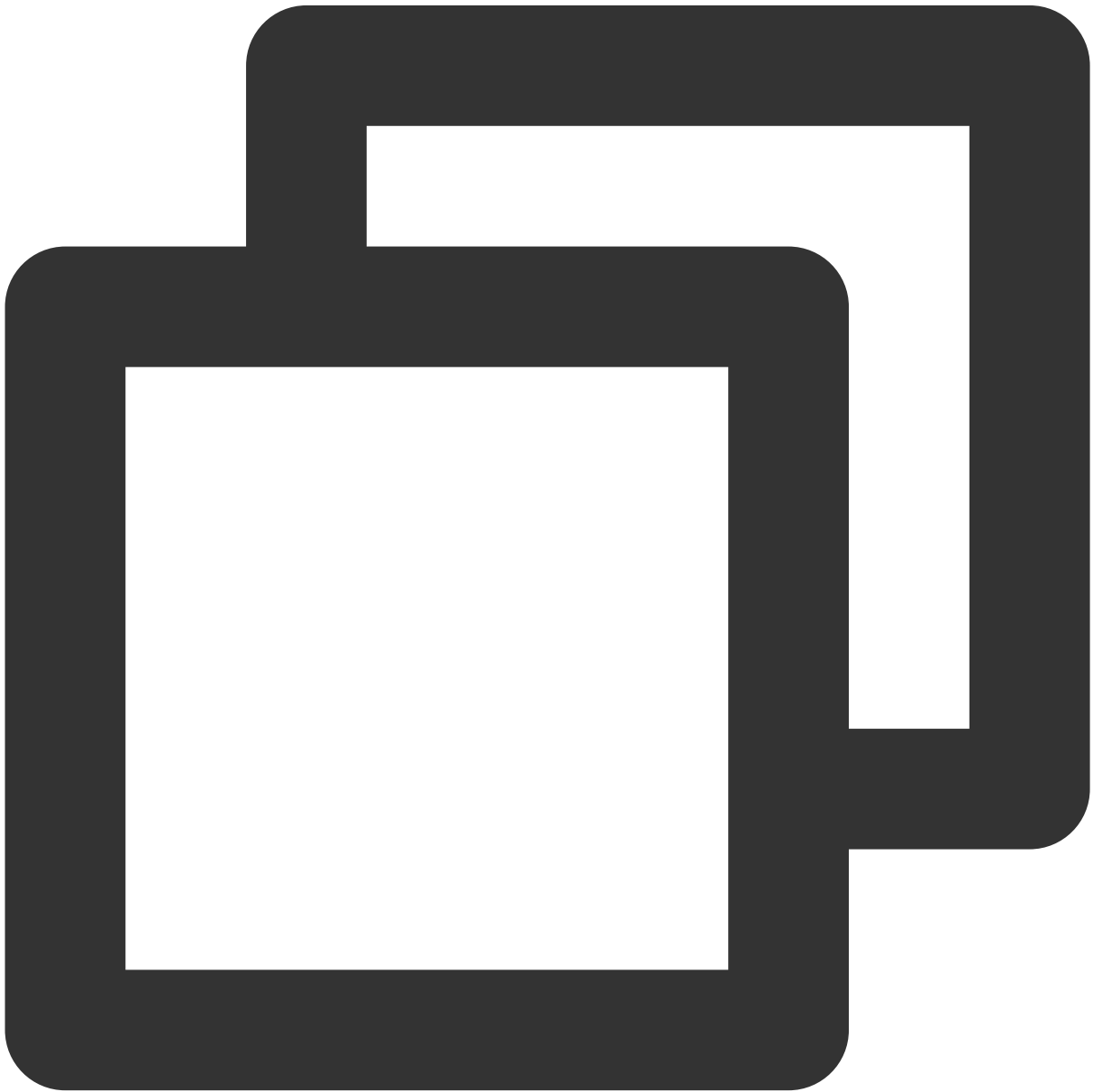
	invitation to talk			
afterCalling	This function will be executed after the termination of the call	function()	No	-
onMinimized	This function will be executed when the component switches to a minimized state. The explanation for the <a href="#">STATUS value is</a>	function(oldStatus, newStatus)	No	-
kickedOut	The events thrown by the component occur when the current logged-in user is ejected. The call will also automatically terminate	function()	No	-
statusChanged	Event thrown by the component; this event is triggered when the call status changes. For detailed types of call	function({oldStatus, newStatus})	No	-

	status, refer to <a href="#">STATUS</a> value description			
--	---	--	--	--

Sample code

React

Vue

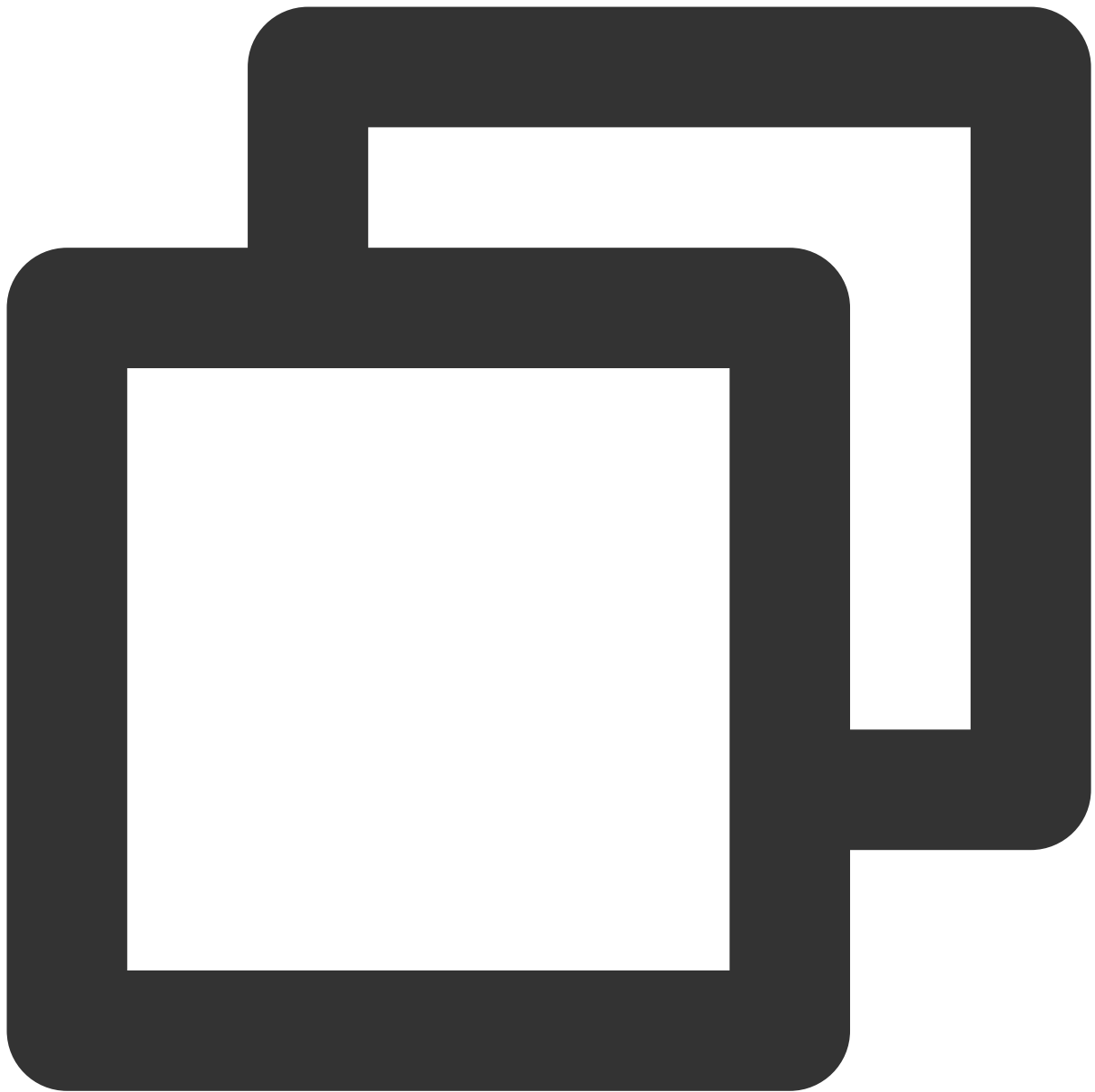


```
import { TUICallKit, VideoDisplayMode, VideoResolution } from "@tencentcloud/call-u
```



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

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

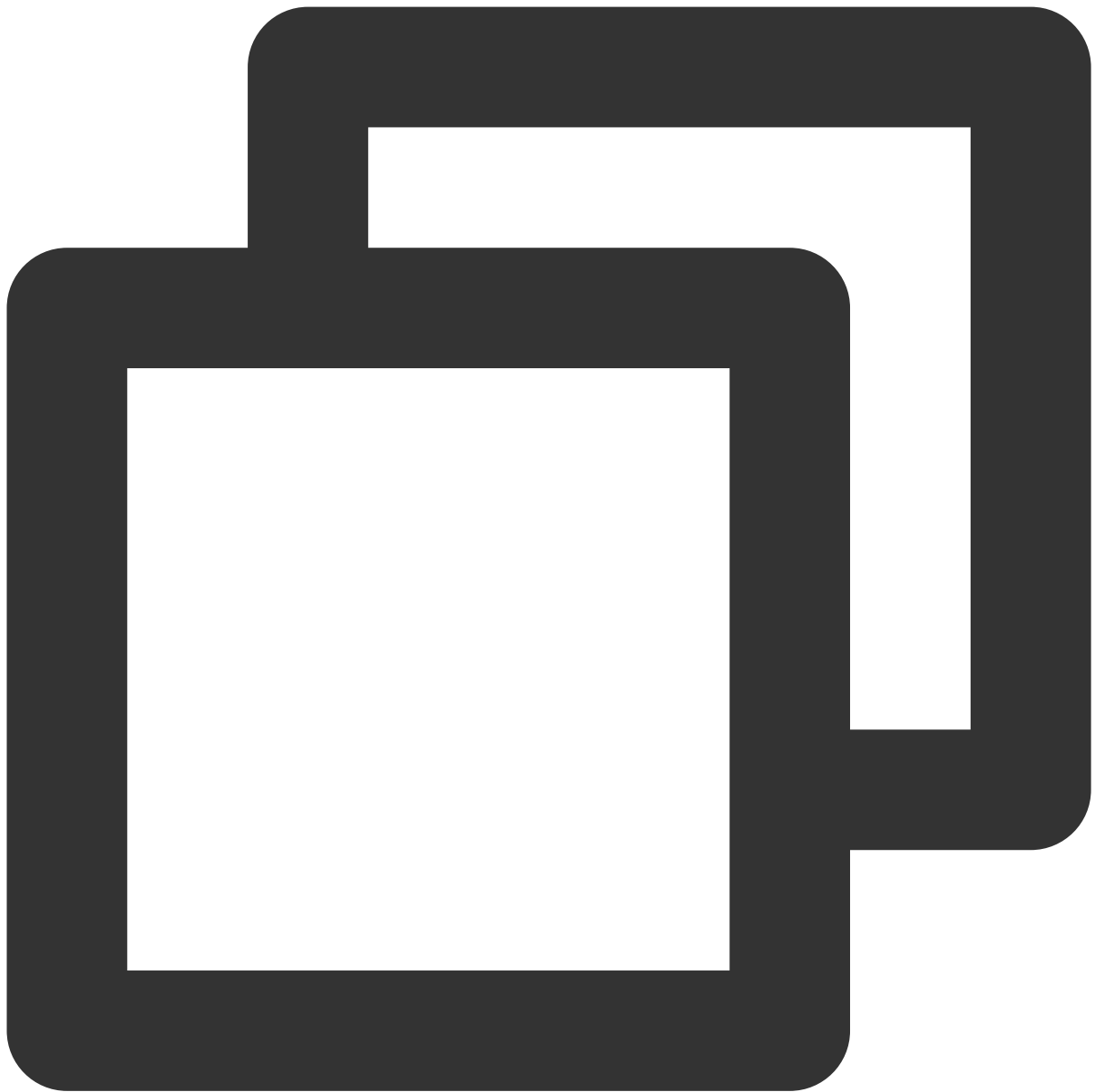


```
<template>
  <TUICallKit
    :allowedMinimized="true"
    :allowedFullScreen="true"
    :videoDisplayMode="VideoDisplayMode.CONTAIN"
    :videoResolution="VideoResolution.RESOLUTION_1080P"
    :beforeCalling="beforeCalling"
    :afterCalling="afterCalling"
    :onMinimized="onMinimized"
    :kickedOut="handleKickedOut"
    :statusChanged="handleStatusChanged"
```

```
    />
</template>
<script lang="ts" setup>
import { TUICallKit, TUICallKitServer, VideoDisplayMode, VideoResolution, STATUS }

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

## Detailed information on TUICallKitServer API



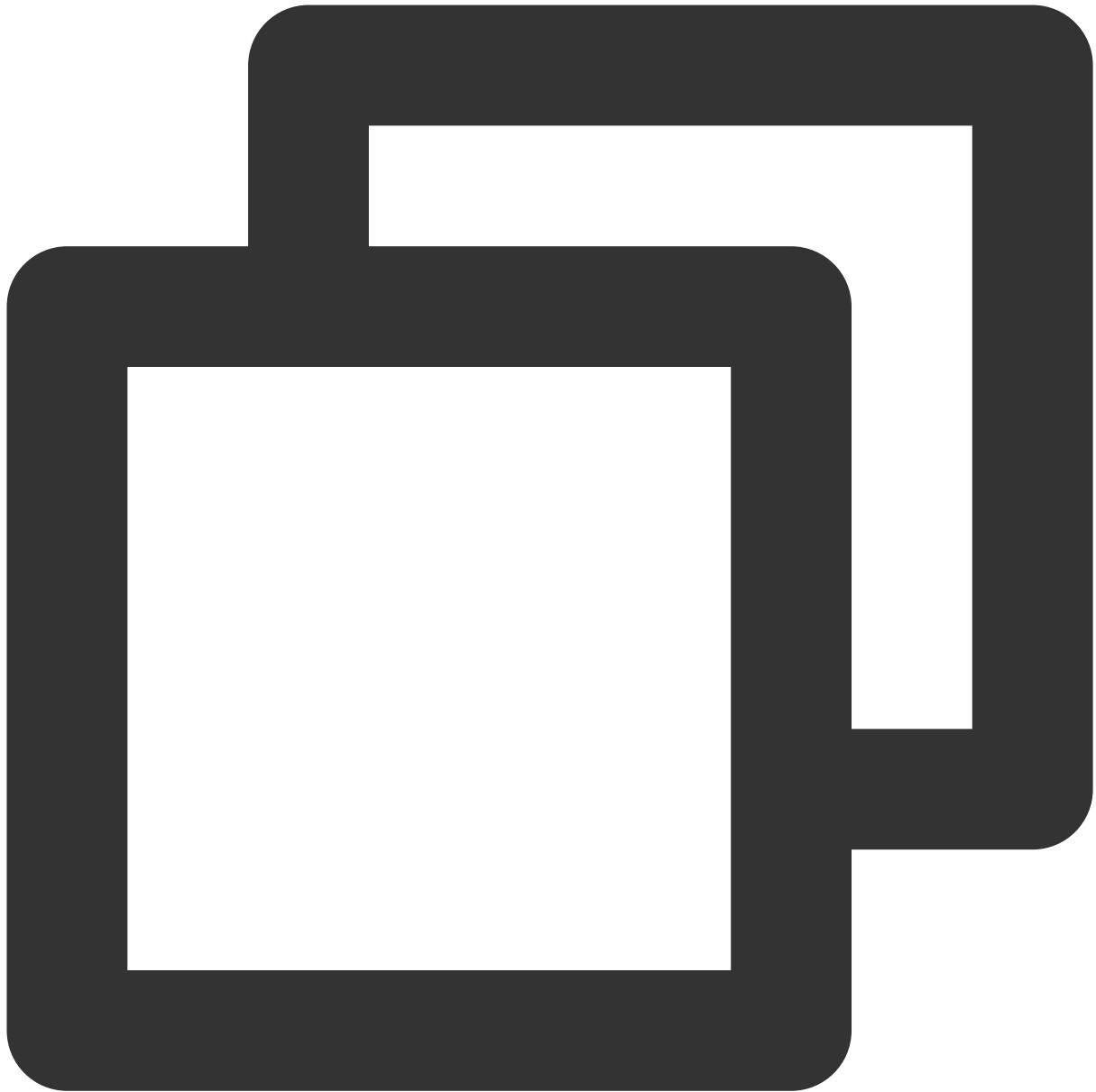
```
import { TUICallKitServer } from "@tencentcloud/call-uikit-react";  
// Replace it with the call-uikit npm package you are currently using
```

## init

Initialize TUICallKit.

### Note :

**Init initialization needs to be completed before functions such as call and groupCall can be used.**



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

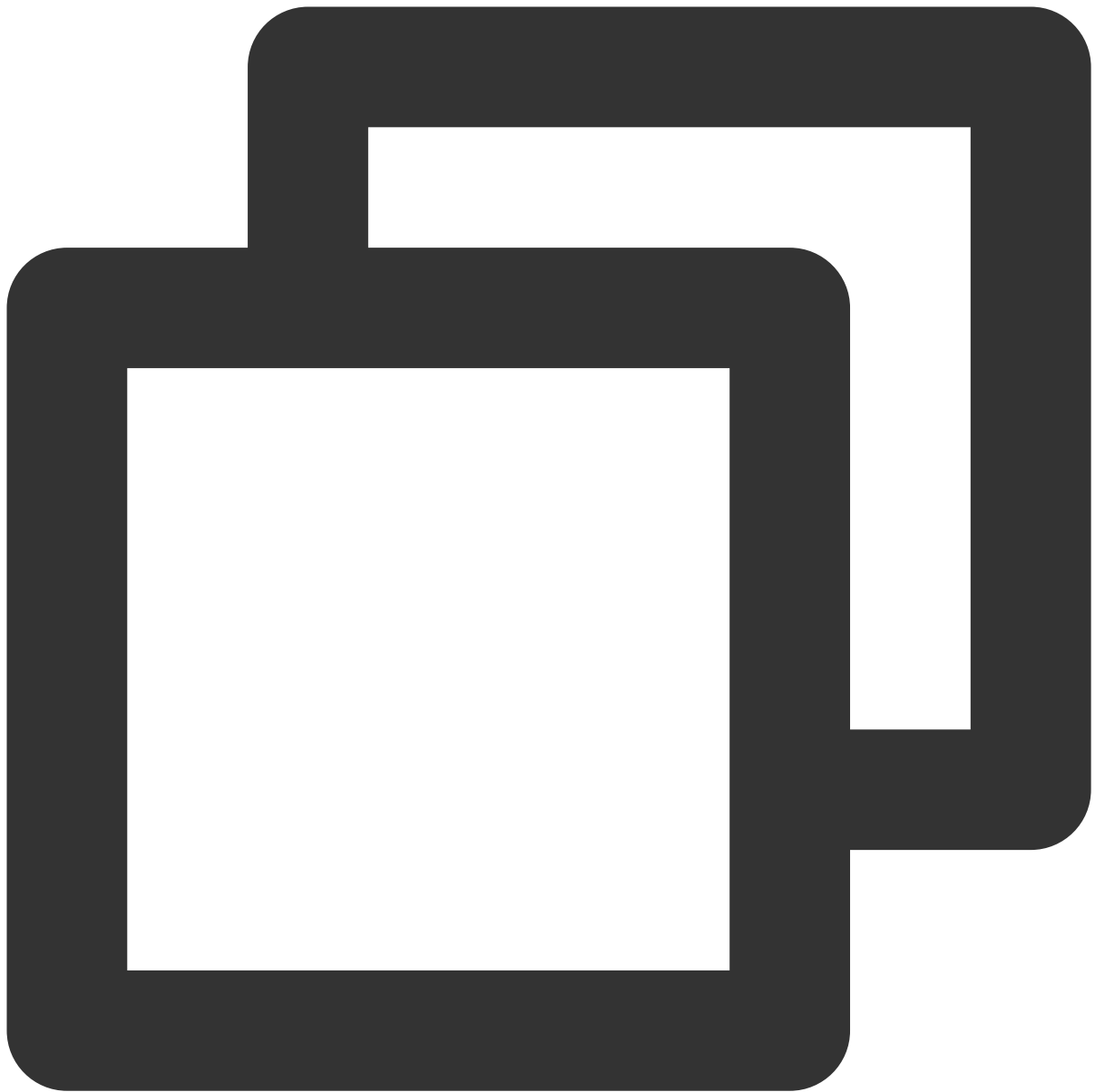
The parameters are described below:

--	--	--	--

Parameter	Type	Required	Meaning
SDKAppID	Number	Yes	The SDKAppID of your app, you can find your SDKAppID in the Live Audio and Video console. For details, see <a href="#">Activating Services</a> .
userID	String	Yes	The current user's ID is of string type, only allowing for the inclusion of English letters (a-z and A-Z), digits (0-9), hyphens (-) and underscores (_)
userSig	String	Yes	Use SDKSecretKey to encrypt SDKAppID, UserID and other information to obtain userSig. It is an authentication ticket used by Tencent Cloud to identify whether the current user can use TRTC services. For how to obtain it, please refer to <a href="#">How to Calculate UserSig</a>
tim	TencentCloudChat	No	tim is an instance of <a href="#">TencentCloudChat</a> SDK.

## call

Makes a one-to-one call, Support for custom room ID, call timeout, offline push content, etc.



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

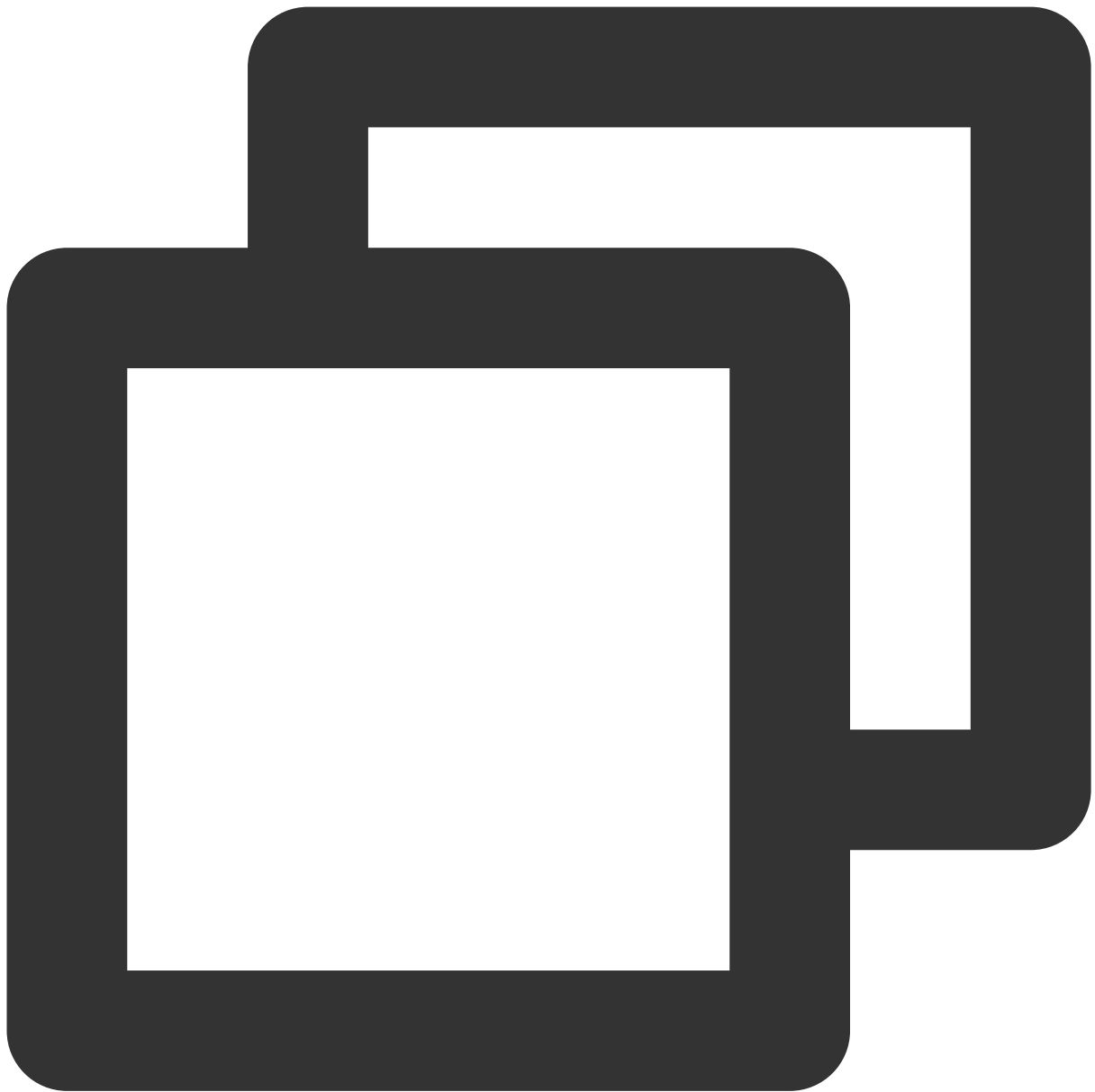
The parameters are described below:

Parameter	Type	Required	Meaning
userID	String	Yes	The username of the called user
type	<a href="#">TUICallType</a>	Yes	The media type of the call, see <a href="#">TUICallType call type</a> for param
roomID	Number	No	Numerical Room ID, range [1, 2147483647]
strRoomID	String	No	String room ID. <b>v3.3.1+ supported range :</b> Limited to 64 bytes in length. The supported character set range Lowercase and uppercase English letters. (a-zA-Z) ; Number (0-9) ; Spaces 、 ! 、 # 、 \$ 、 % 、 & 、 ( 、 ) 、 + . . 1. roomID and strRoomID are mutually exclusive, if you use strRc 2. don't mix roomID and strRoomID, because they are not interch
timeout	Number	No	Call timeout, default: 30s, unit: seconds. timeout = 0, set to no tim
userData	String	No	Customize the extended fields when initiating a call. The called u
<a href="#">offlinePushInfo</a>	Object	No	Customize offline message push parameters

## groupCall

Makes a group call, Support for custom room ID, call timeout, offline push content, etc.





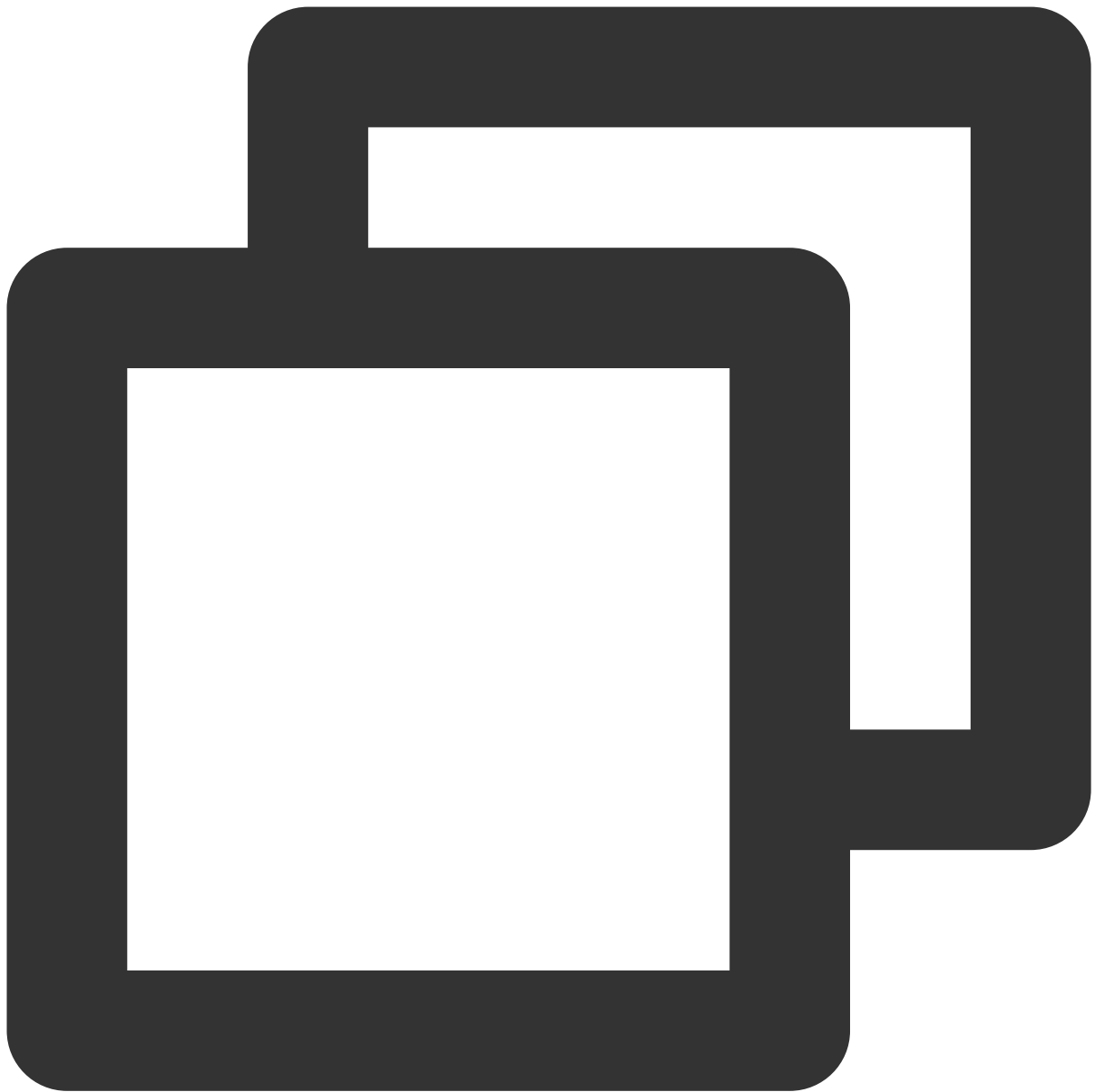
```
import { TUICallKitServer, TUICallType } from '@tencentcloud/call-uikit-react';
try {
  await TUICallKitServer.groupCall({
    userIDList: ['jack', 'tom'],
    groupID: "xxx",
    type: TUICallType.VIDEO_CALL
  });
} catch (error: any) {
  console.error(`[TUICallKit] Failed to call the groupCall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

The parameters are described below:

Parameter	Type	Required	Meaning
userIDList	Array<String>	Yes	List of called users
type	<a href="#">TUICallType</a>	Yes	The type of media for the call, you can refer to <a href="#">TUICallType</a> for i
groupID	String	Yes	Call group ID, the creation of groupID can be referred to <a href="#">chat-cr</a>
roomID	Number	No	Numerical Room ID, range [1, 2147483647]
strRoomID	String	No	String room ID. <b>v3.3.1+ supported range :</b> Limited to 64 bytes in length. The supported character set range Lowercase and uppercase English letters. (a-zA-Z) ; Number (0-9) ; Spaces 、 ! 、 # 、 \$ 、 % 、 & 、 ( 、 ) 、 + 。 1. roomID and strRoomID are mutually exclusive, if you use strF 2. don't mix roomID and strRoomID, because they are not interc
timeout	Number	No	Call timeout, default: 30s, unit: seconds. timeout = 0, set to no ti
userData	String	No	Customize the extended fields when initiating a call. The called
<a href="#">offlinePushInfo</a>	Object	No	Customize offline message push parameters

## setLanguage

Set language, currently supports: Chinese, English, Japanese.



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

The parameters are described below:

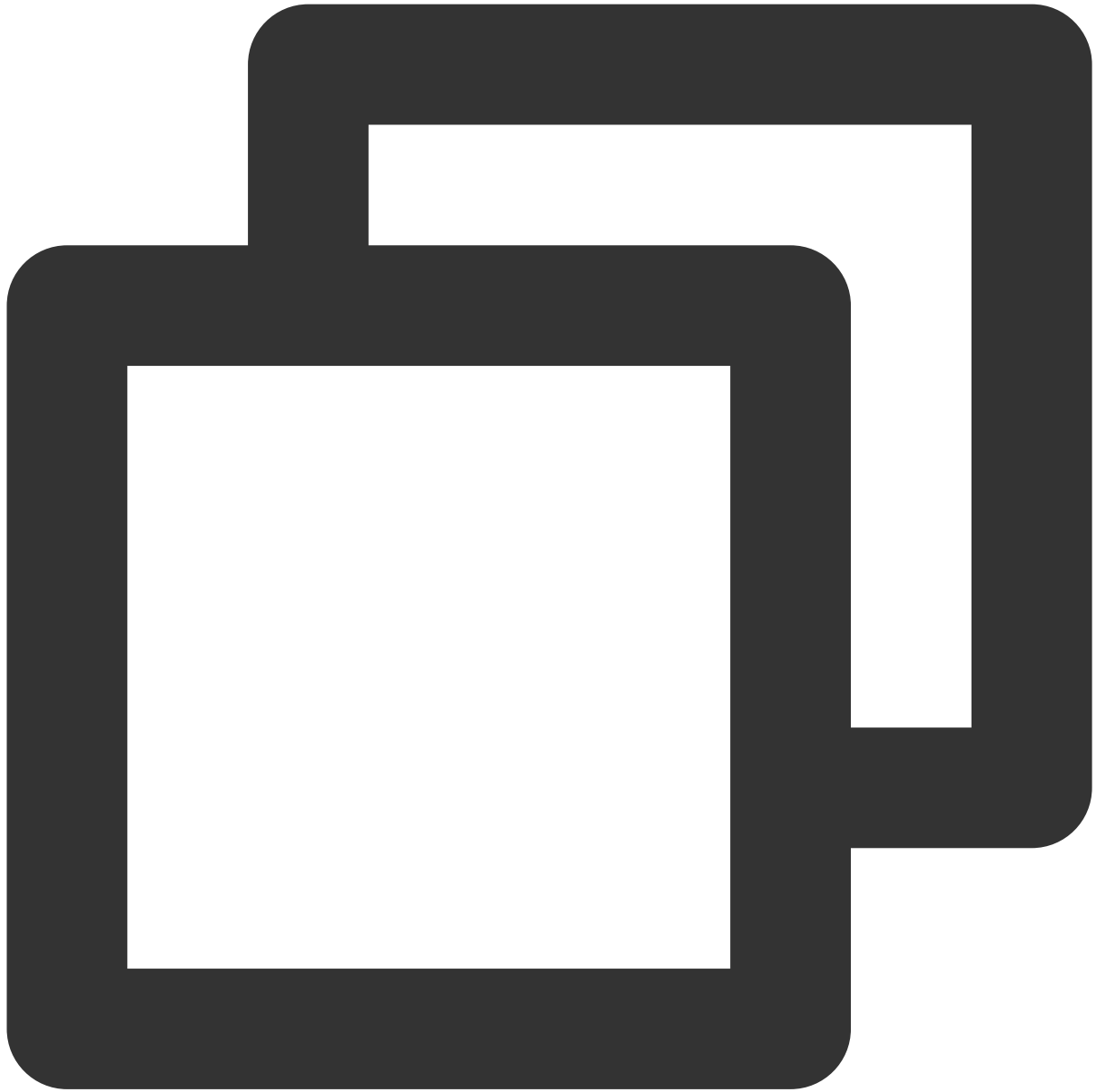
Parameter	Type	Required	Meaning
lang	String	Yes	Language type <code>en</code> , <code>zh-cn</code> and <code>ja_JP</code> .

## setSelfInfo

Set your own nickname and avatar.

**Note:**

**v2.2.0+ supported.** If you use this interface to modify user information during a call, the UI will not be updated immediately, and you will need to wait until the next call to see the changes.



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

The parameters are described below:

Parameter	Type	Required	Meaning
nickName	String	Yes	own nickname
avatar	String	Yes	own avatar address

## setCallingBell

### Note:

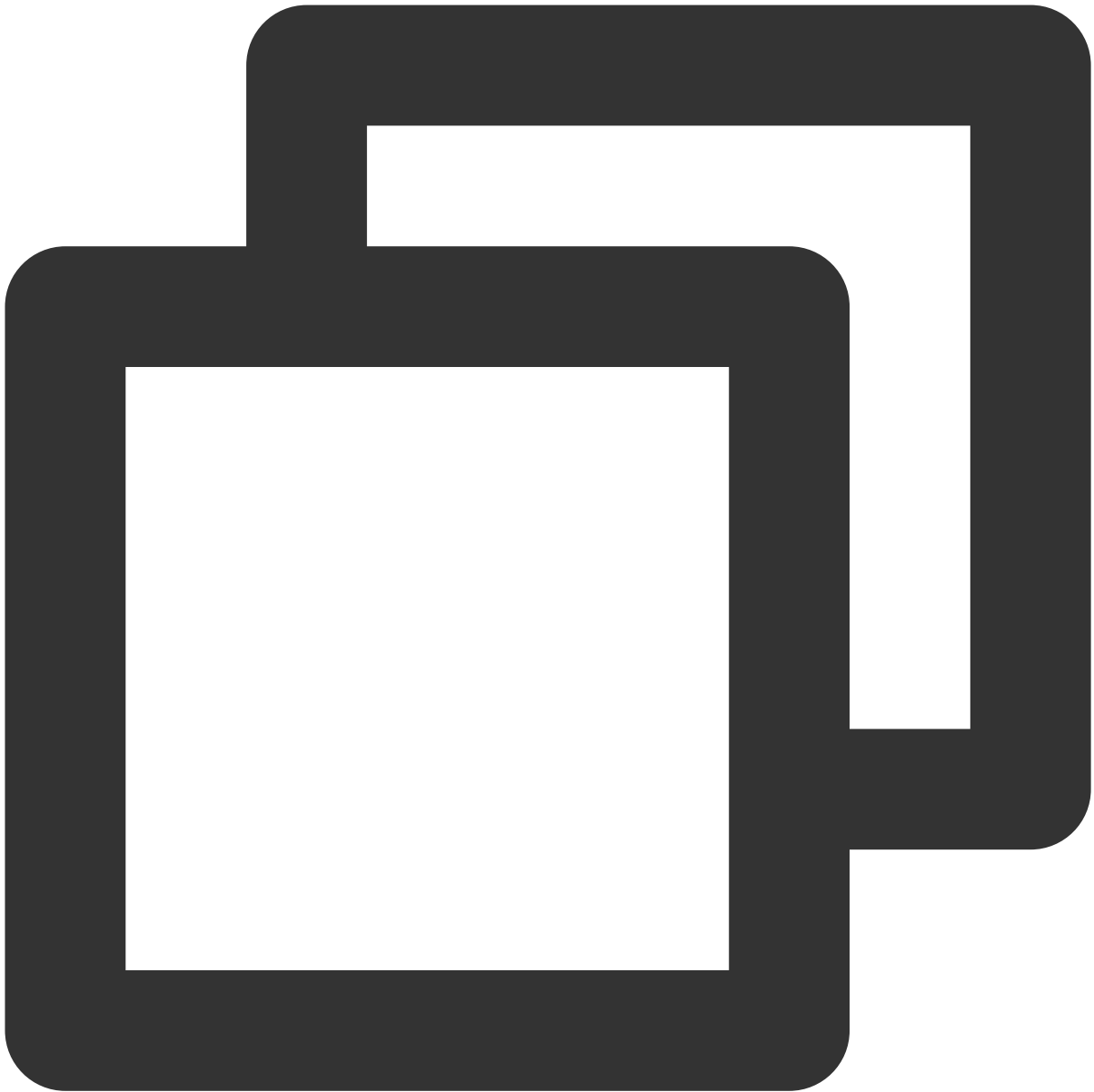
#### v3.0.0+ supported.

Customize the user's incoming call ringtone.

The input is restricted to the local MP3 format file address. It is imperative to ensure that the application has access to this file directory.

Use the import method to import the ringtone file.

If you need to restore the default ringtone, just pass empty filePath.



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

The parameters are described below:

Parameter	Type	Required	Meaning

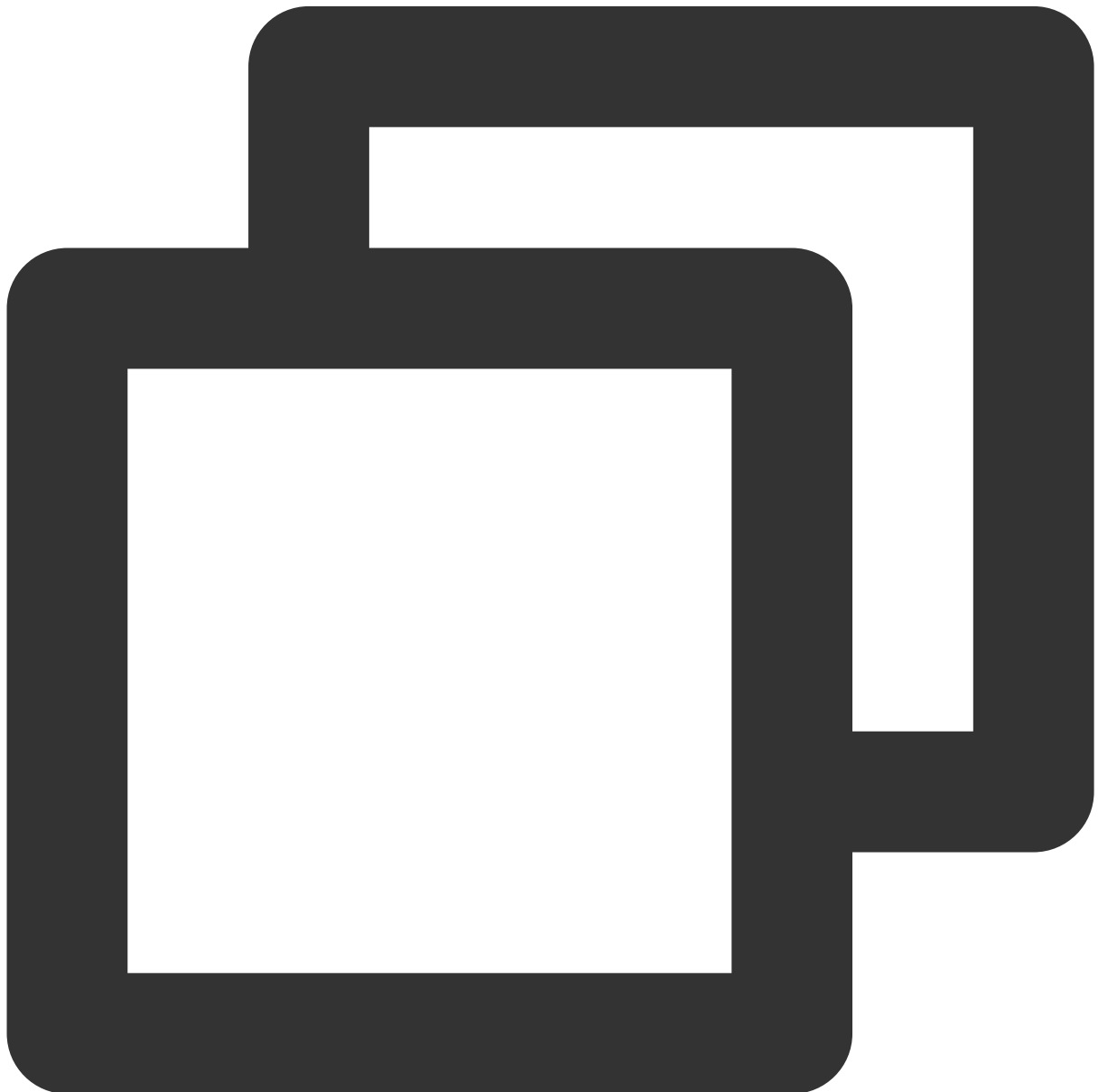
filePath	String	Yes	Ringtone file path
----------	--------	-----	--------------------

## enableFloatWindow

Turn on/off the floating window function. The default is false. The floating window button in the upper left corner of the call interface is hidden. It will be displayed after setting it to true.

### Note:

**v3.1.0+ supported.**



```
try {
```

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

The parameters are described below:

Parameter	Type	Required	Meaning
enable	Boolean	Yes	Turn on/off the floating window function. default false.

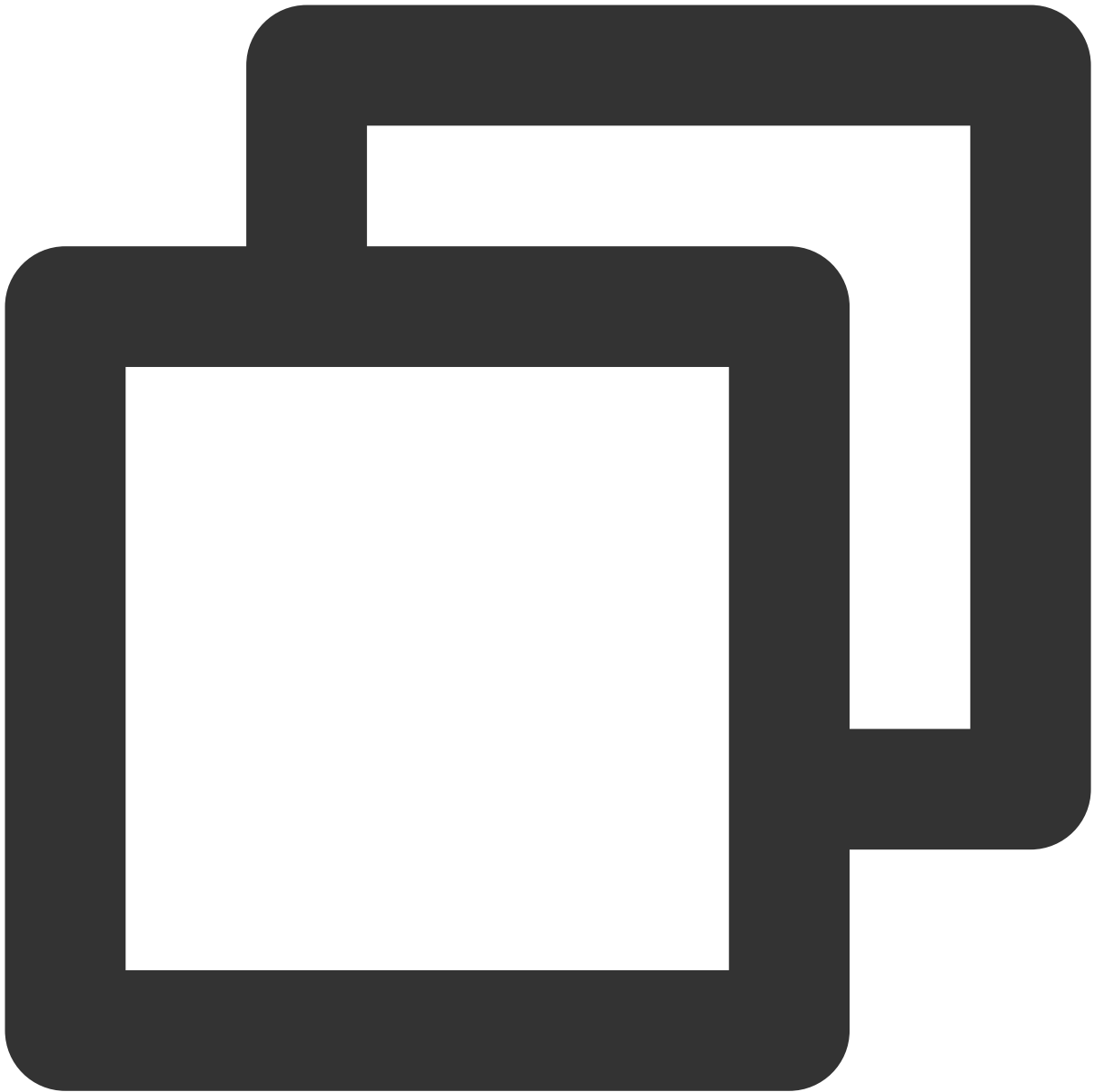
## enableMuteMode

Turn on/off the ringtone for incoming calls. When turned on, the incoming call ringtone will not be played when a call request is received.

### Note:

**v3.1.2+ supported.**





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

The parameters are described below:

Parameter	Type	Required	Meaning

enable	Boolean	Yes	Turn on/off the ringtone for incoming calls function. default false.
--------	---------	-----	--

## joinInGroupCall

Join an existing audio-video call in a group.

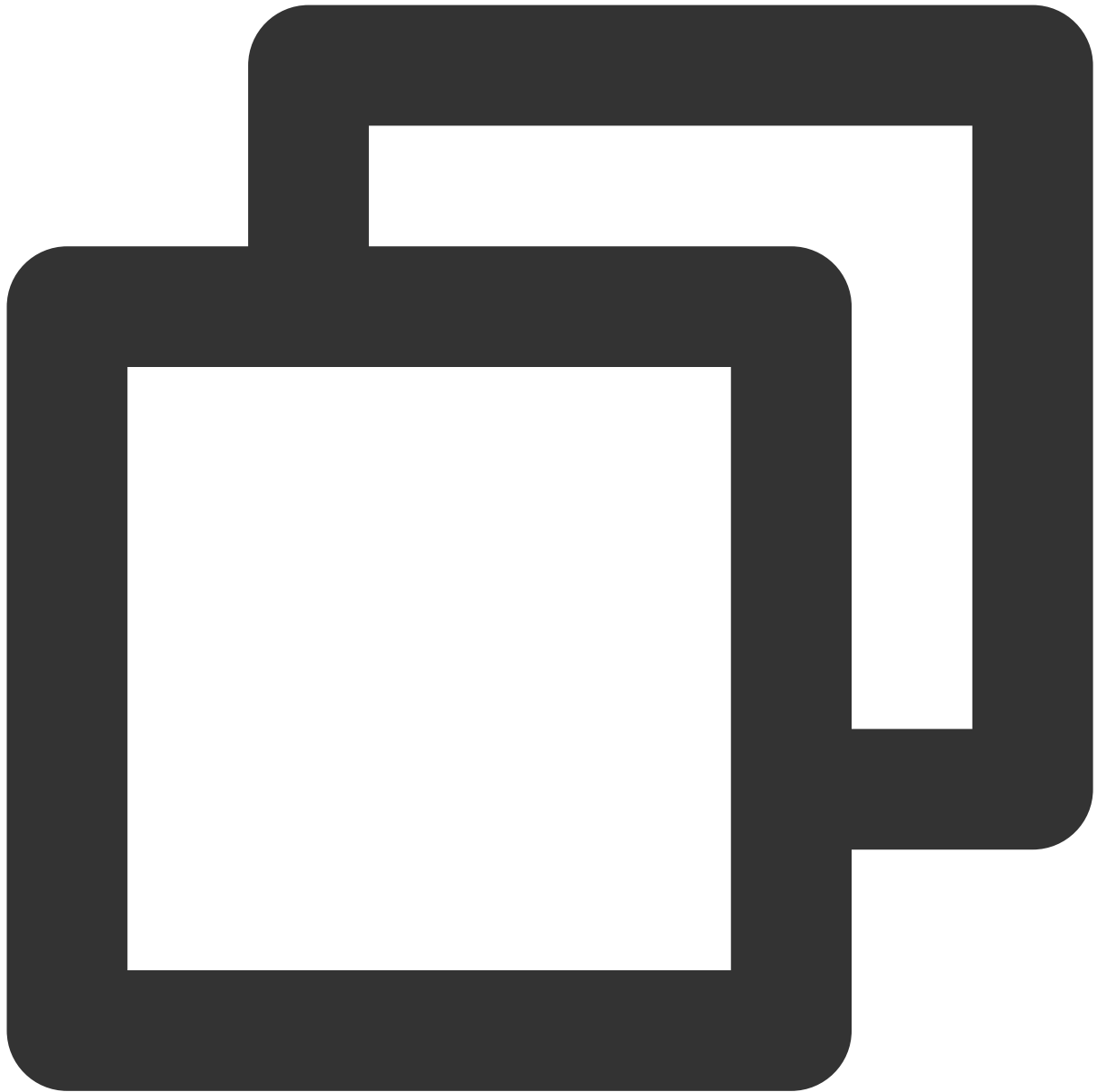
### Note:

**v3.1.2+ supported.**

### Note:

Before joining an existing audio-video call in the group, an IM group must be pre-established or joined, and users in the group must already be engaged in a call. If the group has already been formed, please ignore this requirement.

Instructions for creating a group can be found at [IM Group Management](#). Alternatively, you may directly utilize [IM TUIKit](#) for an all-in-one integration of chat, call and other scenarios.



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

The parameters are described below:

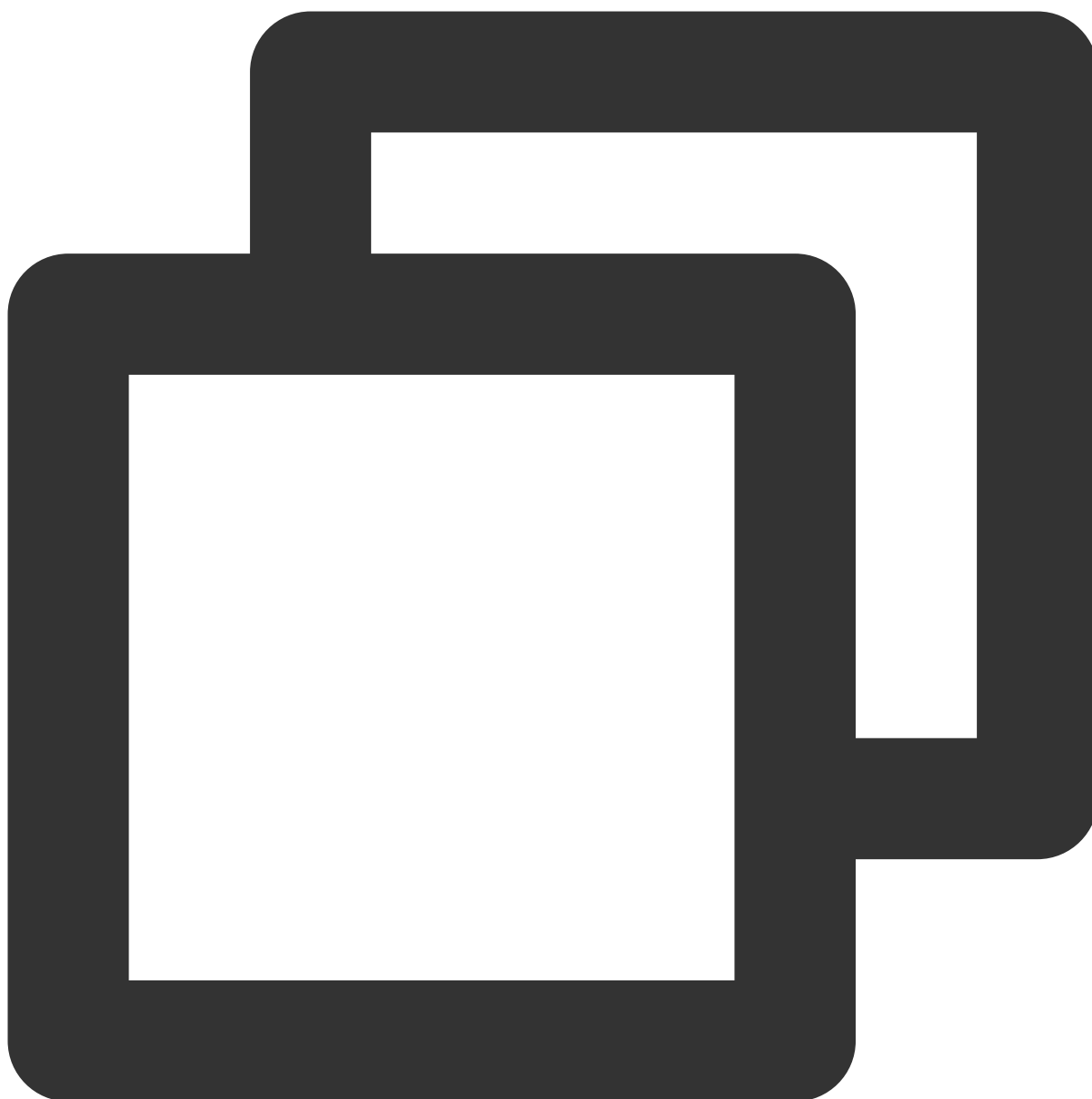
Parameter	Type	Required	Meaning
type	<a href="#">TUICallType</a>	Yes	The media type of communication, for instance, video calls, voice calls
groupID	string	Yes	The group ID for this group call
roomID	number	Yes	Audio and video room ID for this call

## enableVirtualBackground

Turn on/off the blurred background function. If you want to set the picture background to be blurry see [Web](#). By calling the interface, you can display the blurred background function button on the UI, and click the button to directly enable the blurred background function.

**Note :**

**v3.2.4+ supported.**



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

The parameters are described below:

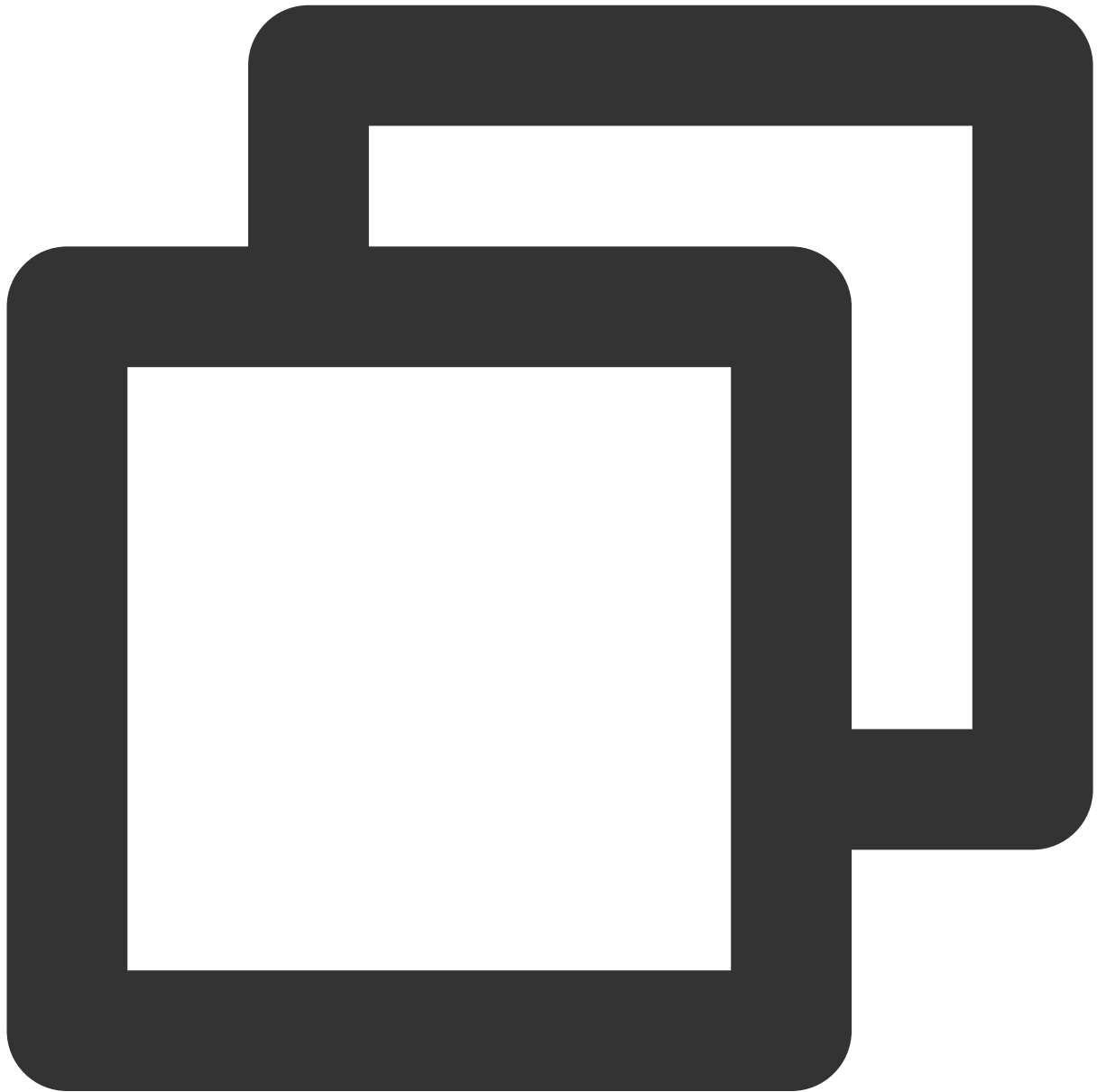
Parameter	Type	Required	Meaning
enable	boolean	Yes	enable = true, show blur background button enable = false, don't show blur background button

## hideFeatureButton

Hidden feature buttons, currently only support Camera, Microphone, and Switch Camera Button.

**Note :**

**v3.2.9+ supported.**



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

The parameters are described below:

Parameter	Type	Required	Meaning

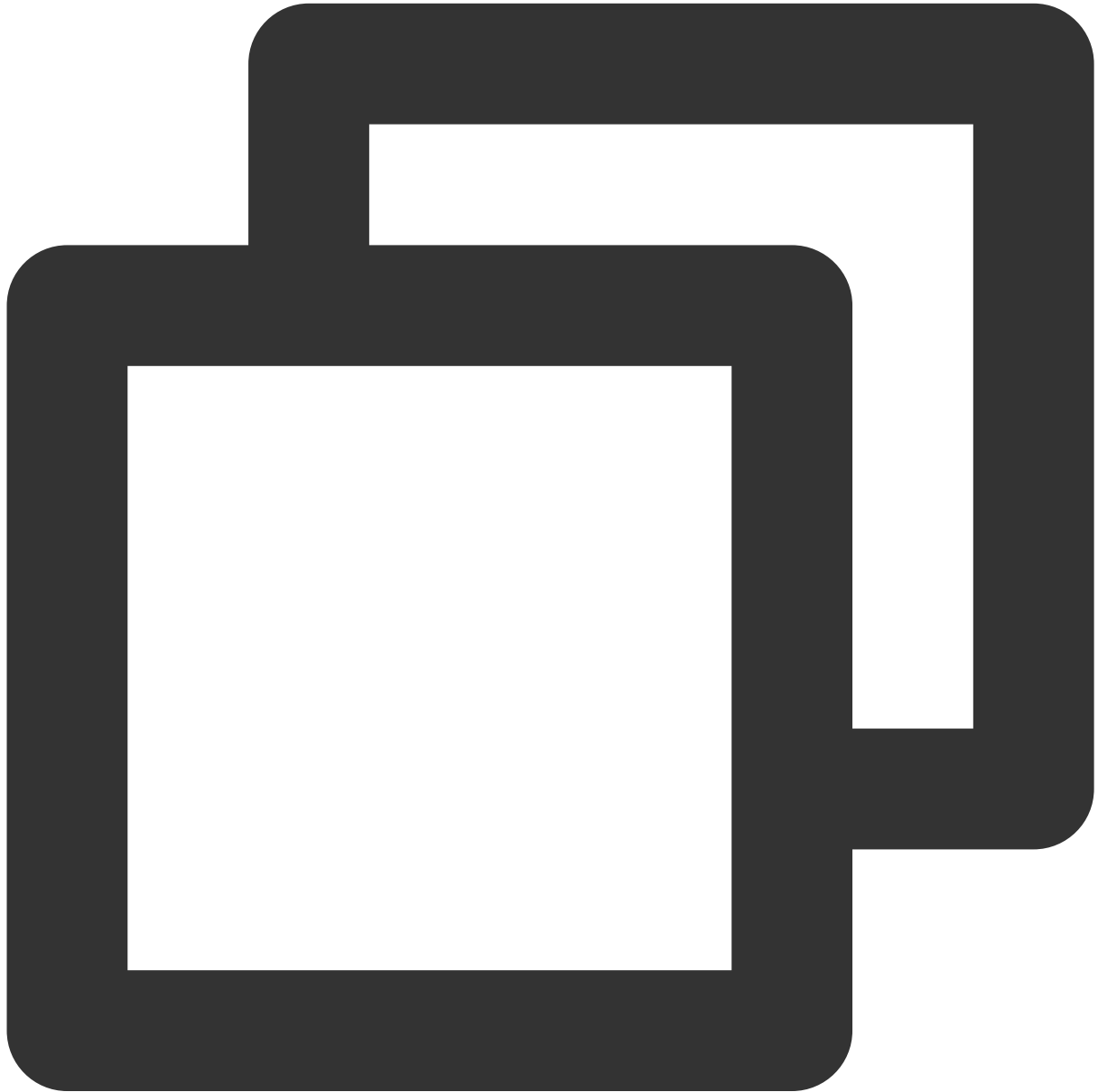
buttonName	FeatureButton	Yes	Button Name
------------	---------------	-----	-------------

## setLocalViewBackgroundImage

Set the background image for the local user's call interface.

**Note :**

**v3.2.9+ supported.**



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

The parameters are described below:

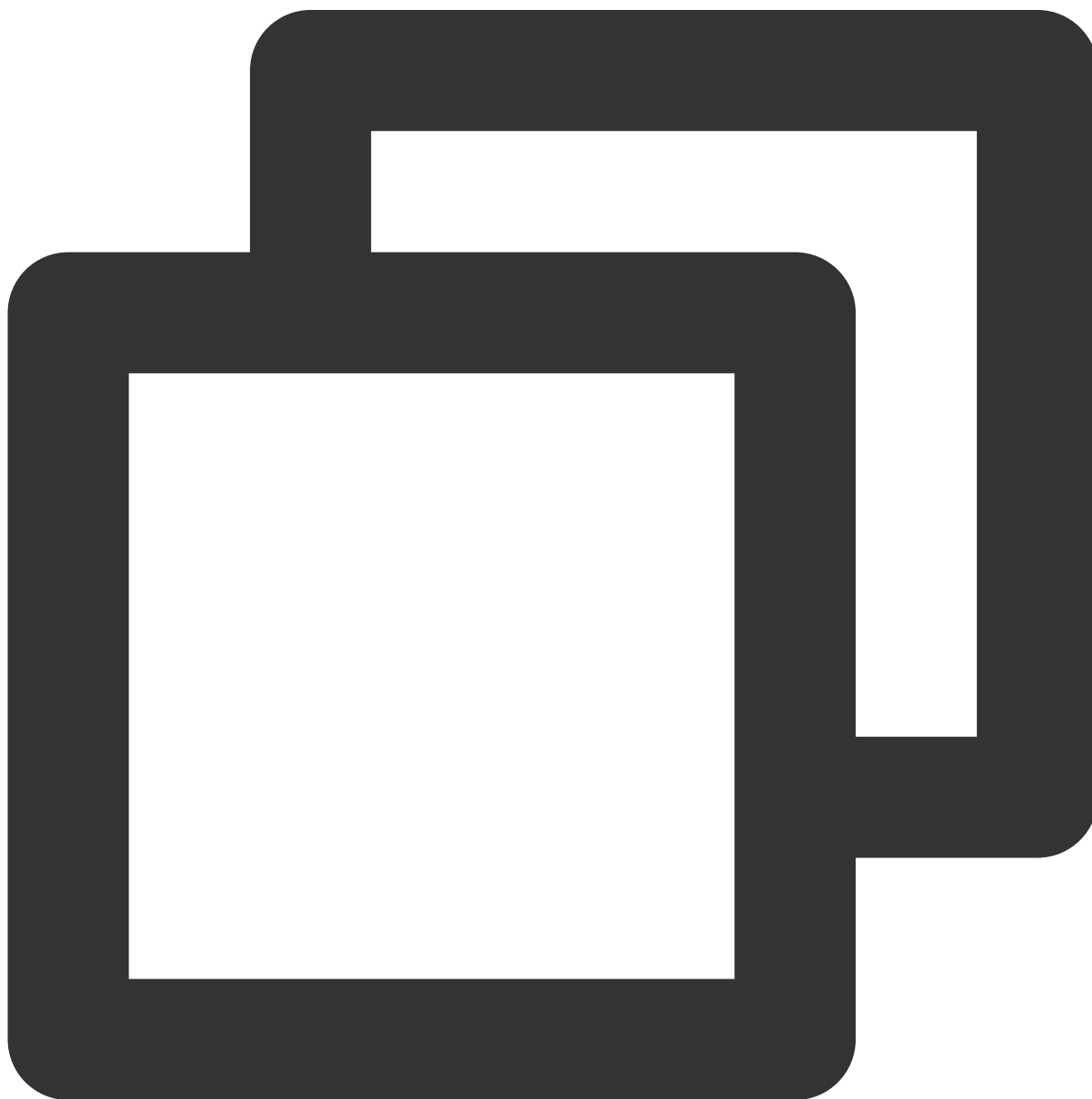
Parameter	Type	Required	Meaning
url	string	Yes	Image Address (supports Local Path and Network Address)

## setRemoteViewBackgroundImage

Set the background image for the remote user's call interface.

**Note :**

**v3.2.9+ supported.**





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

The parameters are described below:

Parameter	Type	Required	Meaning
userId	string	Yes	Remote User userId, setting to '*' means it applies to all Remote Users
url	string	Yes	Image Address (supports Local Path and Network Address)

## setLayoutMode

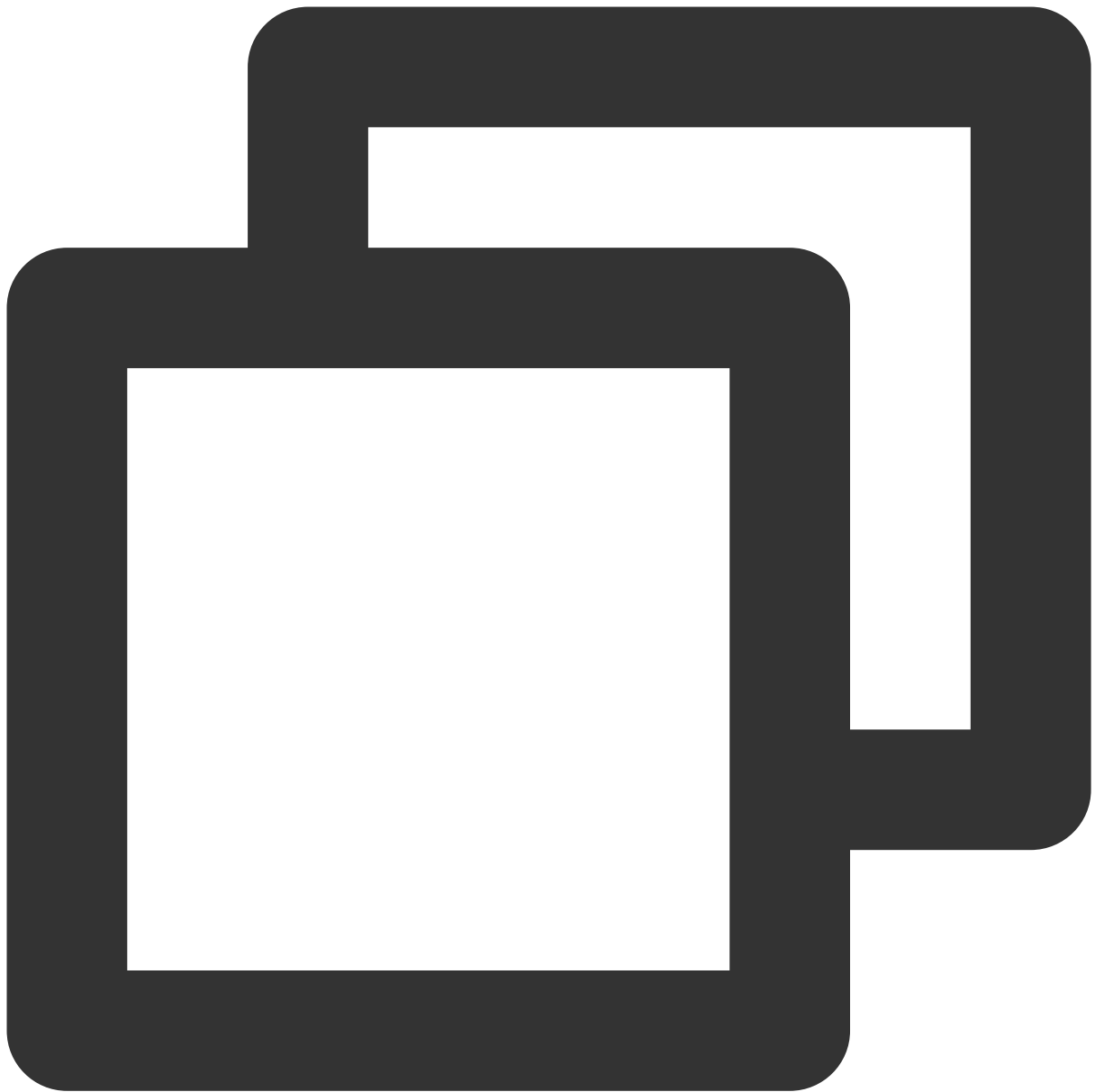
Set the call interface layout mode.

**Note :**

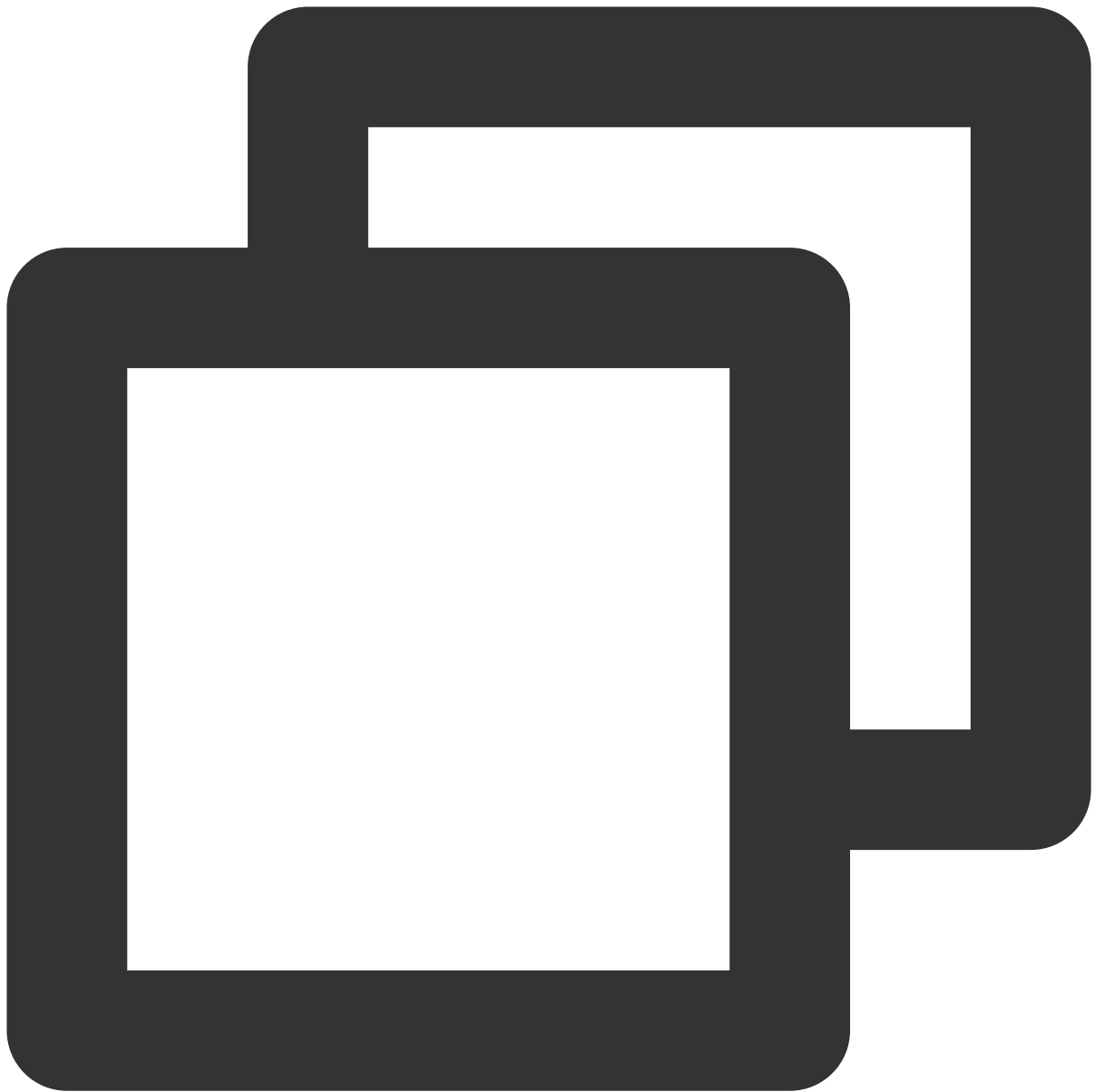
**supported from v3.3.0+.**

Vue

React



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



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

Parameter list:

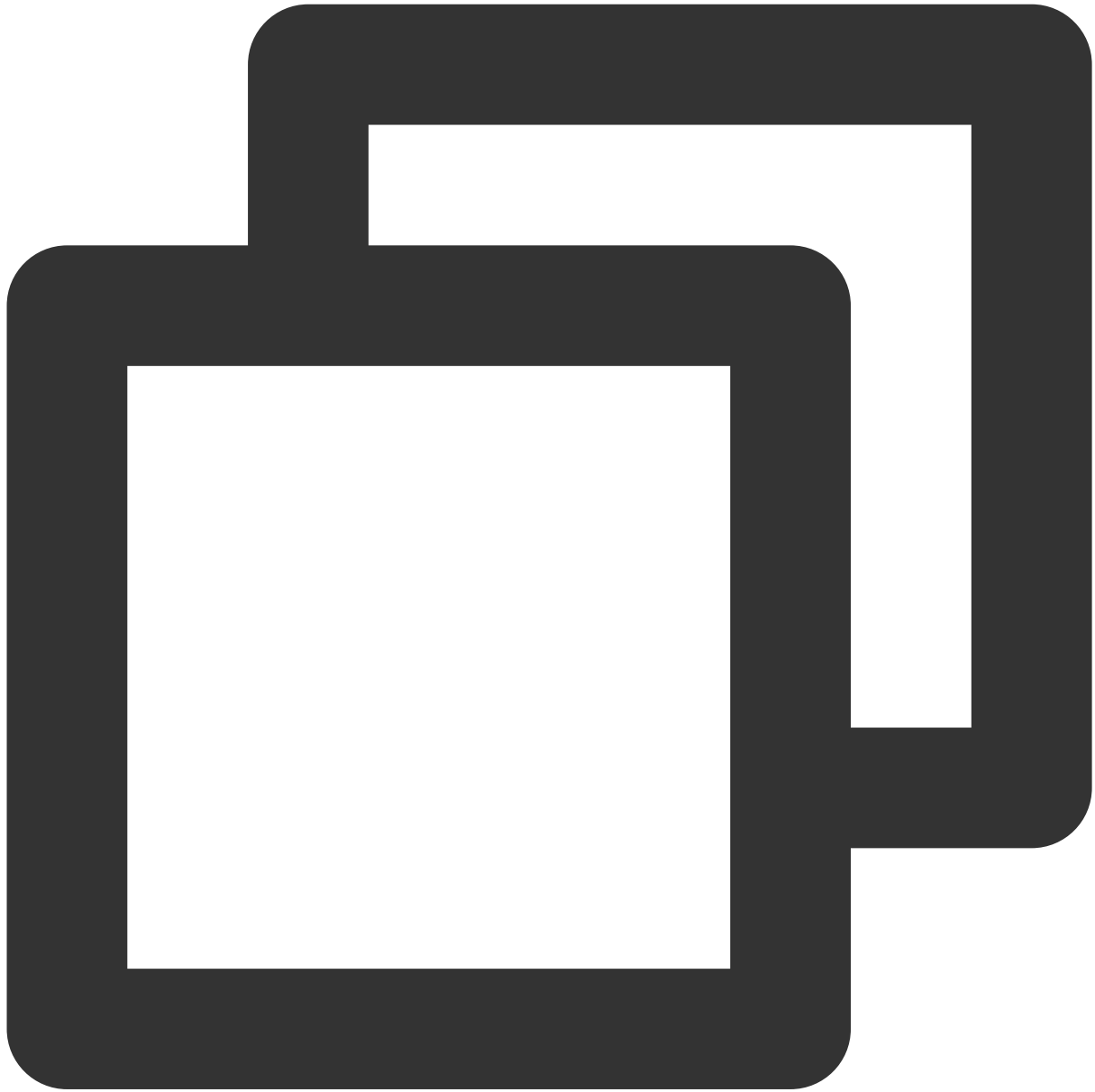
Parameter	Type	Required	Meaning
layoutMode	<a href="#">LayoutMode</a>	Yes	User flow layout mode

## setCameraDefaultState

Set whether the camera is on by default.

**Note :**

**supported from v3.3.0+.**



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

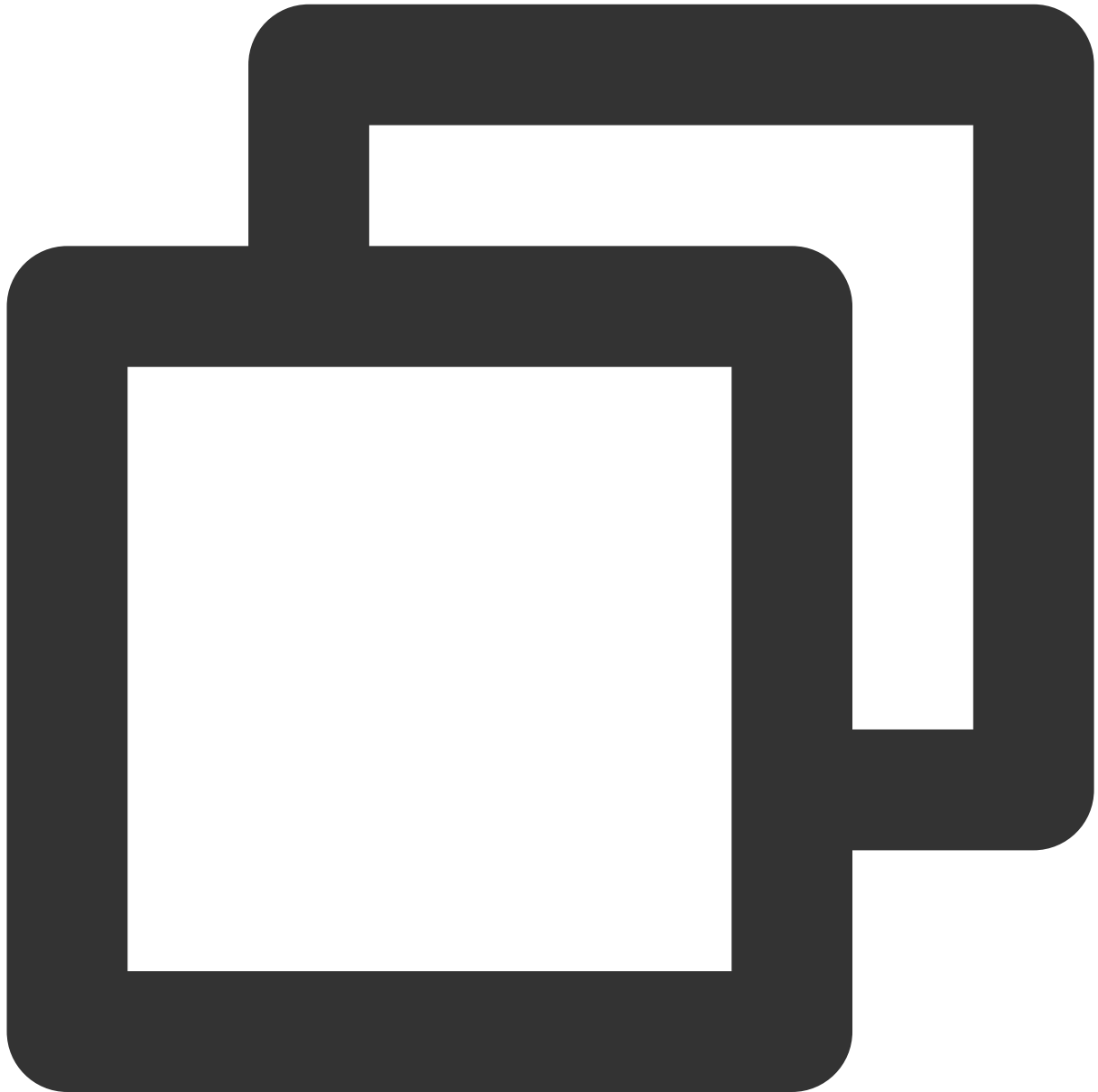
Parameter list:

Parameter	Type	Required	Meaning
isOpen	boolean	Yes	Whether to enable the camera

## destroyed

Terminate the TUICallKit instance.

This method won't automatically log out of `tim` , manual logging out is required: `tim.logout();` .



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

# TUICallKit Type Definition

## videoDisplayMode

There are three values for the `videoDisplayMode` display mode:

`VideoDisplayMode.CONTAIN`

`VideoDisplayMode.COVER`

`VideoDisplayMode.FILL` , the default value is `VideoDisplayMode.COVER` .

Attribute	Value	Description
videoDisplayMode	VideoDisplayMode.CONTAIN	Ensuring the full display of video content is our top priority. The dimensions of the video are scaled proportionally until one side aligns with the frame of the viewing window. In case of discrepancy in sizes between the video and the display window, the video is scaled - on the premise of maintaining the aspect ratio - to fill the window, resulting in a black border around the scaled video.
	VideoDisplayMode.COVER	Priority is given to ensure that the viewing window is filled. The video size is scaled proportionally until the entire window is filled. If the video's dimensions are different from those of the display window, the video stream will be cropped or stretched to match the window's ratio.
	VideoDisplayMode.FILL	Ensuring that the entire video content is displayed while filling the window does not guarantee preservation of the original video's proportion. The dimensions of the video will be stretched to match those of the window.

## videoResolution

The resolution `videoResolution` has three possible values:

`VideoResolution.RESOLUTION_480P`

`VideoResolution.RESOLUTION_720P`

`VideoResolution.RESOLUTION_1080P` , the default value is `VideoResolution.RESOLUTION_480P` .

### Resolution Explanation:

Video Profile	Resolution (W x H)	Frame Rate (fps)	Bitrate (Kbps)
---------------	--------------------	------------------	----------------

480p	640 × 480	15	900
720p	1280 × 720	15	1500
1080p	1920 × 1080	15	2000

### Frequently Asked Questions:

iOS 13&14 does not support encoding videos higher than 720P. It is suggested to limit the highest collection to 720P on these two system versions. Refer to [iOS Safari known issue case 12](#).

Firefox does not permit the customization of video frame rates (default is set to 30fps).

Due to the influence of system performance usage, camera collection capabilities, browser restrictions, and other factors, the actual values of video resolution, frame rate, and bit rate may not necessarily match the set values exactly. In such scenarios, the browser will automatically adjust the Profile to get as close to the set values as feasible.

## STATUS

STATUS attribute value	Description
STATUS.IDLE	Idle status
STATUS.BE_INVITED	Received an Audio/Video Call Invite
STATUS.DIALING_C2C	Initiating a one-to-one call
STATUS.DIALING_GROUP	Initiating a group call
STATUS.CALLING_C2C_AUDIO	Engaged in a 1v1 Audio Call
STATUS.CALLING_C2C_VIDEO	In the midst of a one-to-one video call
STATUS.CALLING_GROUP_AUDIO	Engaged in Group Audio Communication
STATUS.CALLING_GROUP_VIDEO	Engaged in group video call

## TUICallType

TUICallType Type	Description
TUICallType.AUDIO_CALL	Audio Call
TUICallType.VIDEO_CALL	Video Call

## offlinePushInfo

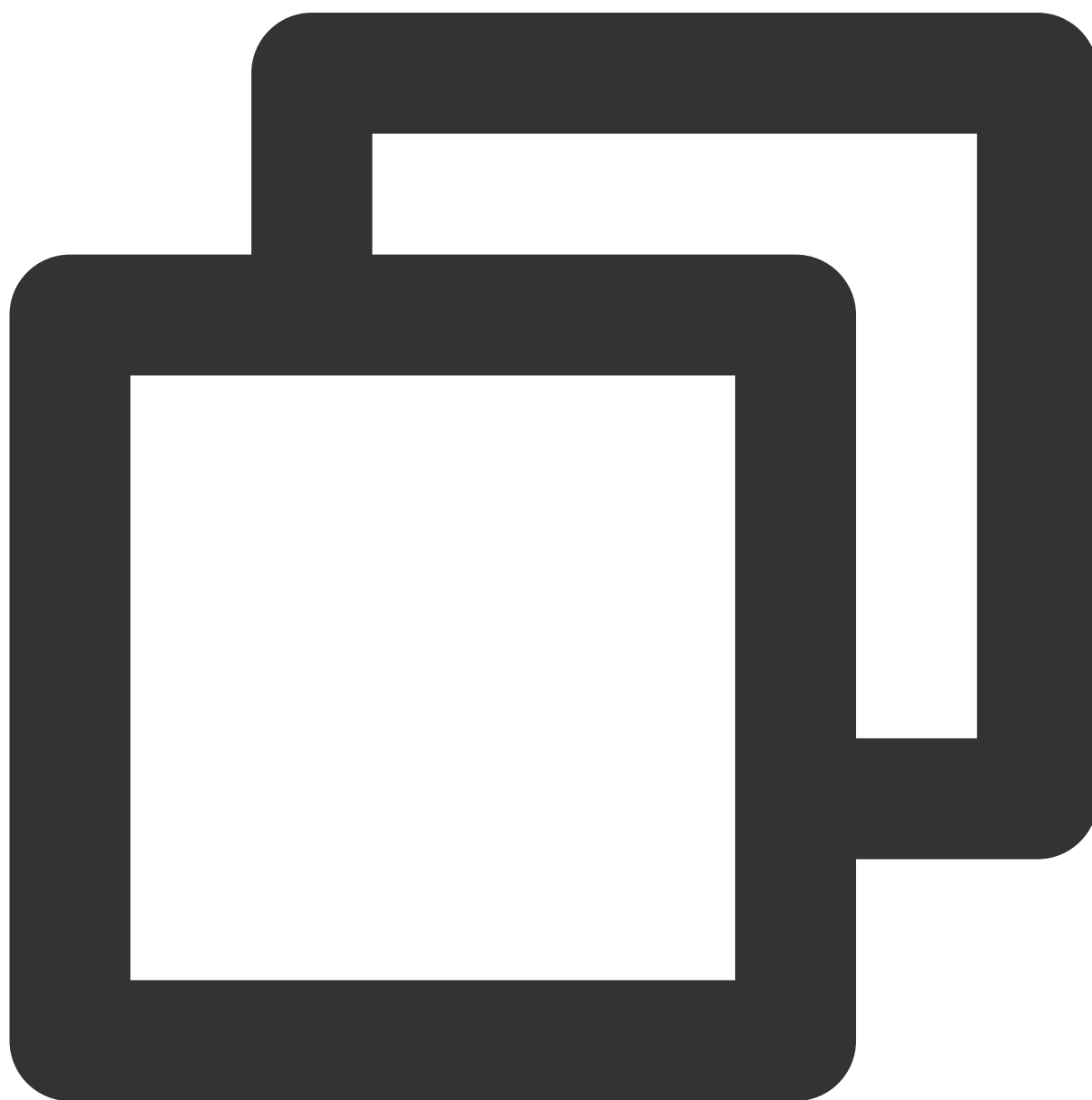
--	--	--	--

Parameter	Type	Required	Meaning
offlinePushInfo.title	String	No	Offline Push Title (Optional)
offlinePushInfo.description	String	No	Offline Push Content (Optional)
offlinePushInfo.androidOPPOChannelID	String	No	Setting the channel ID for OPPO phones with 8.0 system and above for offline pushes (Optional)
offlinePushInfo.extension	String	No	Offline push through content. Can be used to set Android <a href="#">Notification mode</a> and <a href="#">VoIP mode</a> . Default: Notification mode, it will be a notification from the system; VoIP mode is required to pass the field.
offlinePushInfo.ignoreIOSBadge	Boolean	No	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. <b>v3.3.0+ supported</b>
offlinePushInfo.iOSSound	String	No	Offline push sound setting (only for iOS). <b>v3.3.0+ supported</b>
offlinePushInfo.androidSound	String	No	Offline push sound setting. <b>v3.3.0+ supported</b>
offlinePushInfo.androidVIVOClassification	Number	No	Classification of VIVO push messages (deprecated interface, VIVO push service will optimize message classification rules on April 3, 2023. It is recommended to use <code>setAndroidVIVOCategory</code> to set the message category). 0: Operational messages, 1: System messages. The default value is 1. <b>v3.3.0+ supported</b>
offlinePushInfo.androidXiaoMiChannelID	String	No	Set the channel ID for Xiaomi phones with Android 8.0 and above systems. <b>v3.3.0+ supported</b>
offlinePushInfo.androidFCMChannelID	String	No	Set the channel ID for google phones with Android 8.0 and above systems. <b>v3.3.0+ supported</b>



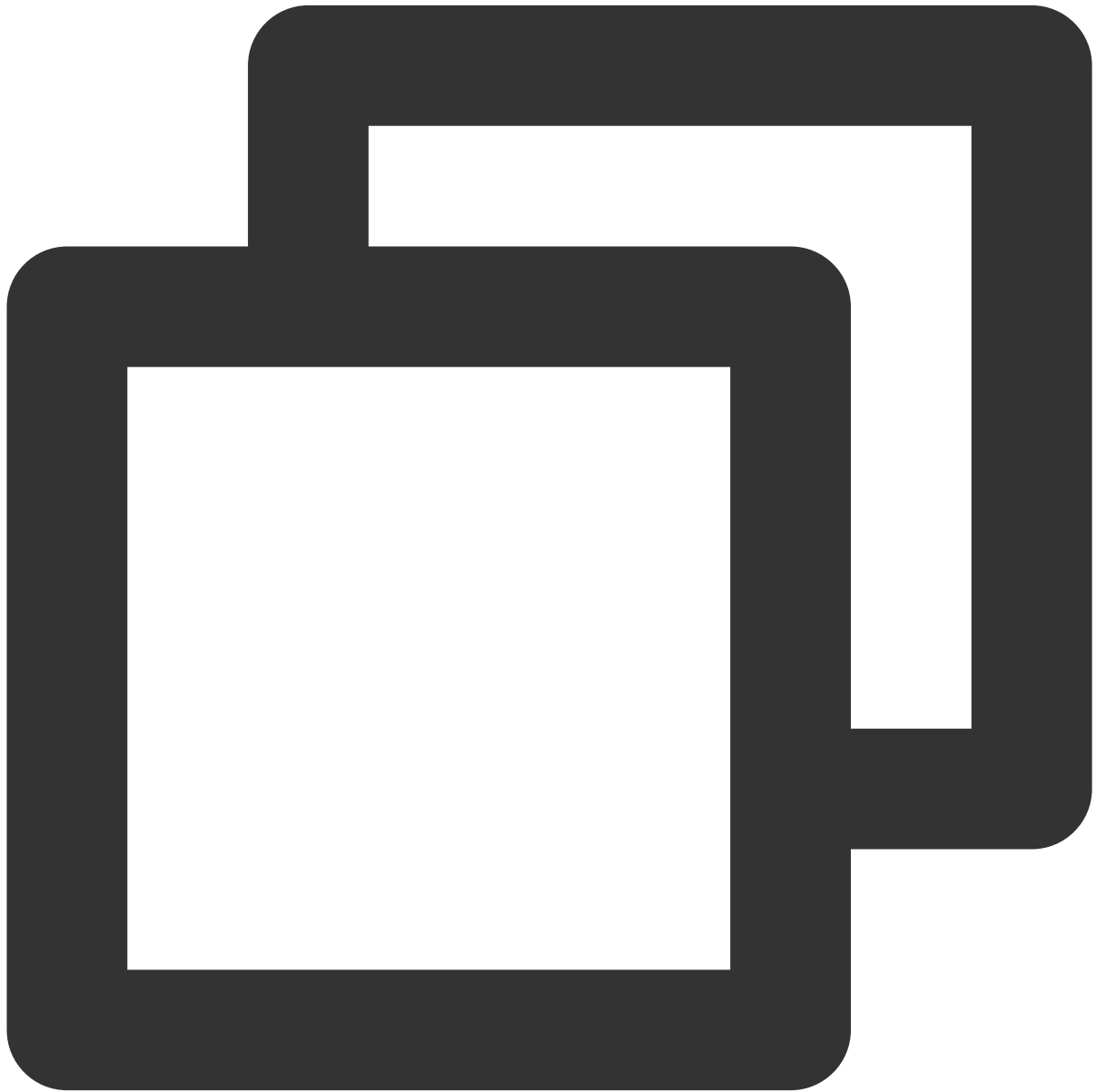
offlinePushInfo.androidHuaWeiCategory	String	No	Classification of Huawei push messages. <b>v3.3.0+ supported</b>
offlinePushInfo.isDisablePush	Boolean	No	Whether to turn off push notifications (default is on). <b>v3.3.0+ supported</b>
offlinePushInfo.iOSPushType	Number	No	iOS offline push type, default is 0. 0-APNs ; 1-VoIP. <b>v3.3.0+ supported</b>

## Android Notification Mode



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

## Android VoIP Mode



```
const extension = {
```

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

## FeatureButton

FeatureButton Type	Description
FeatureButton.Camera	Camera Button
FeatureButton.Microphone	Microphone Button
FeatureButton.SwitchCamera	Switches between the front and rear cameras.

## LayoutMode

LayoutMode type	Description
LayoutMode.LocalInLargeView	Local user in large window display
LayoutMode.RemoteInLargeView	Remote user in large window display

# TUICallEngine

Last updated : 2024-08-09 17:55:11

## TUICallEngine APIs

TUICallEngine API is the **No UI Interface** of the Audio and Video Call Components.

### API Overview

API	Description
<a href="#">createInstance</a>	Creating a TUICallEngine Instance (Singleton Pattern)
<a href="#">destroyInstance</a>	Terminating a TUICallEngine Instance (Singleton Pattern)
<a href="#">on</a>	Listening on events
<a href="#">off</a>	Canceling Event Listening
<a href="#">login</a>	Sign in Interface
<a href="#">logout</a>	Logout Interface
<a href="#">setSelfInfo</a>	Configure the user's nickname and profile photo
<a href="#">call</a>	Initiate a one-on-one call
<a href="#">groupCall</a>	Group Chat Invitation Call
<a href="#">accept</a>	Answer Calls
<a href="#">reject</a>	Decline Call
<a href="#">hangup</a>	End Calls
<a href="#">switchCallMediaType</a>	Switch Audio and Video Calls
<a href="#">startRemoteView</a>	Initiate Remote Screen Rendering
<a href="#">stopRemoteView</a>	Stop Remote Screen Rendering
<a href="#">startLocalView</a>	Start Local Screen Rendering, <b>Note: This will be deprecated; use openCamera instead</b>

<a href="#">stopLocalView</a>	Stop Local Screen Rendering, <b>Note: This will be deprecated; use <a href="#">closeCamera</a> instead</b>
<a href="#">openCamera</a>	Enable the camera
<a href="#">closeCamara</a>	Turn Off Camera
<a href="#">switchCamera</a>	Switch between front and rear cameras, note: only supported on mobile devices. <b>v3.0.0+ supported</b>
<a href="#">openMicrophone</a>	Enable Microphone
<a href="#">closeMicrophone</a>	Turn off the microphone
<a href="#">setVideoQuality</a>	Set video quality
<a href="#">getDeviceList</a>	Access device list
<a href="#">switchDevice</a>	Switch camera or microphone devices
<a href="#">enableAIVoice</a>	Enable/disable AI noise reduction
<a href="#">enableMultiDeviceAbility</a>	Turn on/off the multi-device login mode of TUICallEngine. <b>v2.1.1+ supported</b>
<a href="#">setBlurBackground</a>	Switch/set background blur, <b>v3.0.6+ supported</b>
<a href="#">setVirtualBackground</a>	Switch/set image background blur, <b>v3.0.6+ supported</b>

# TUICallEvent

Last updated : 2024-08-09 18:00:13

## TUICallEvent API Introduction

TUICallEvent API is the **Event Interface** of the Audio and Video Call Components.

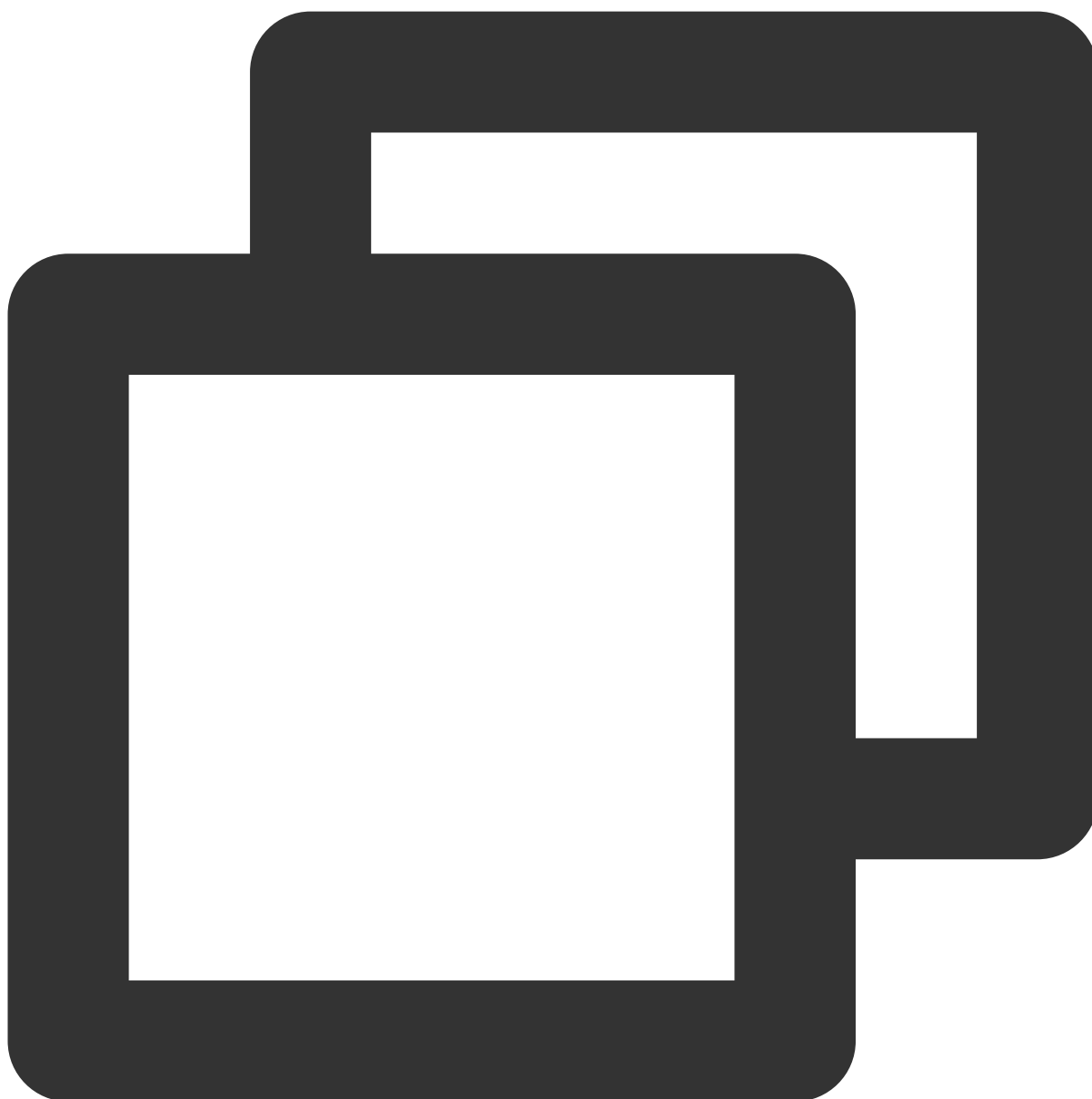
### Event List

EVENT	Description
<a href="#">TUICallEvent.ERROR</a>	An error occurred during the call.
<a href="#">TUICallEvent.SDK_READY</a>	This event is received when the SDK enters the ready state
<a href="#">TUICallEvent.KICKED_OUT</a>	Receiving this event after a duplicate sign-in indicates that the user has been removed from the room
<a href="#">TUICallEvent.USER_ACCEPT</a>	If a user answers, this event will be received
<a href="#">TUICallEvent.USER_ENTER</a>	A user joined the call.
<a href="#">TUICallEvent.USER_LEAVE</a>	A user left the call.
<a href="#">TUICallEvent.REJECT</a>	A user declined the call.
<a href="#">TUICallEvent.NO_RESP</a>	A user didn't respond.
<a href="#">TUICallEvent.LINE_BUSY</a>	A user was busy.
<a href="#">TUICallEvent.USER_VIDEO_AVAILABLE</a>	Whether a user has a video stream.
<a href="#">TUICallEvent.USER_AUDIO_AVAILABLE</a>	Whether a user has an audio stream.
<a href="#">TUICallEvent.USER_VOICE_VOLUME</a>	The volume levels of all users.
<a href="#">TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE</a>	Group Chat Update, Invitation List this callback will be received
<a href="#">TUICallEvent.ON_CALL_BEGIN</a>	Call connected event, <b>v1.4.6+ supported</b>

<a href="#">TUICallEvent.INVITED</a>	A call was received. <b>It will be discarded later and it is recommended to use TUICallEvent.ON_CALL_RECEIVED</b>
<a href="#">TUICallEvent.ON_CALL_RECEIVED</a>	Call request event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.CALLING_CANCEL</a>	Call cancellation event, <b>It will be abandoned later and it is recommended to use TUICallEvent.ON_CALL_CANCELED</b>
<a href="#">TUICallEvent.ON_CALL_CANCELED</a>	Call cancellation event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.ON_CALL_BEGIN</a>	Call connected event, <b>v1.4.6+ supported</b>
<a href="#">TUICallEvent.CALLING_END</a>	The call ended.
<a href="#">TUICallEvent.DEVICED_UPDATED</a>	Device list update, this event will be received
<a href="#">TUICallEvent.CALL_TYPE_CHANGED</a>	Call type switching, this event will be received
<a href="#">TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED</a>	All user network quality events, <b>v3.0.7+ supported</b>

## ERROR

Error event during the call. You can capture internal errors during the call by monitoring this event.



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

The parameters are described below:

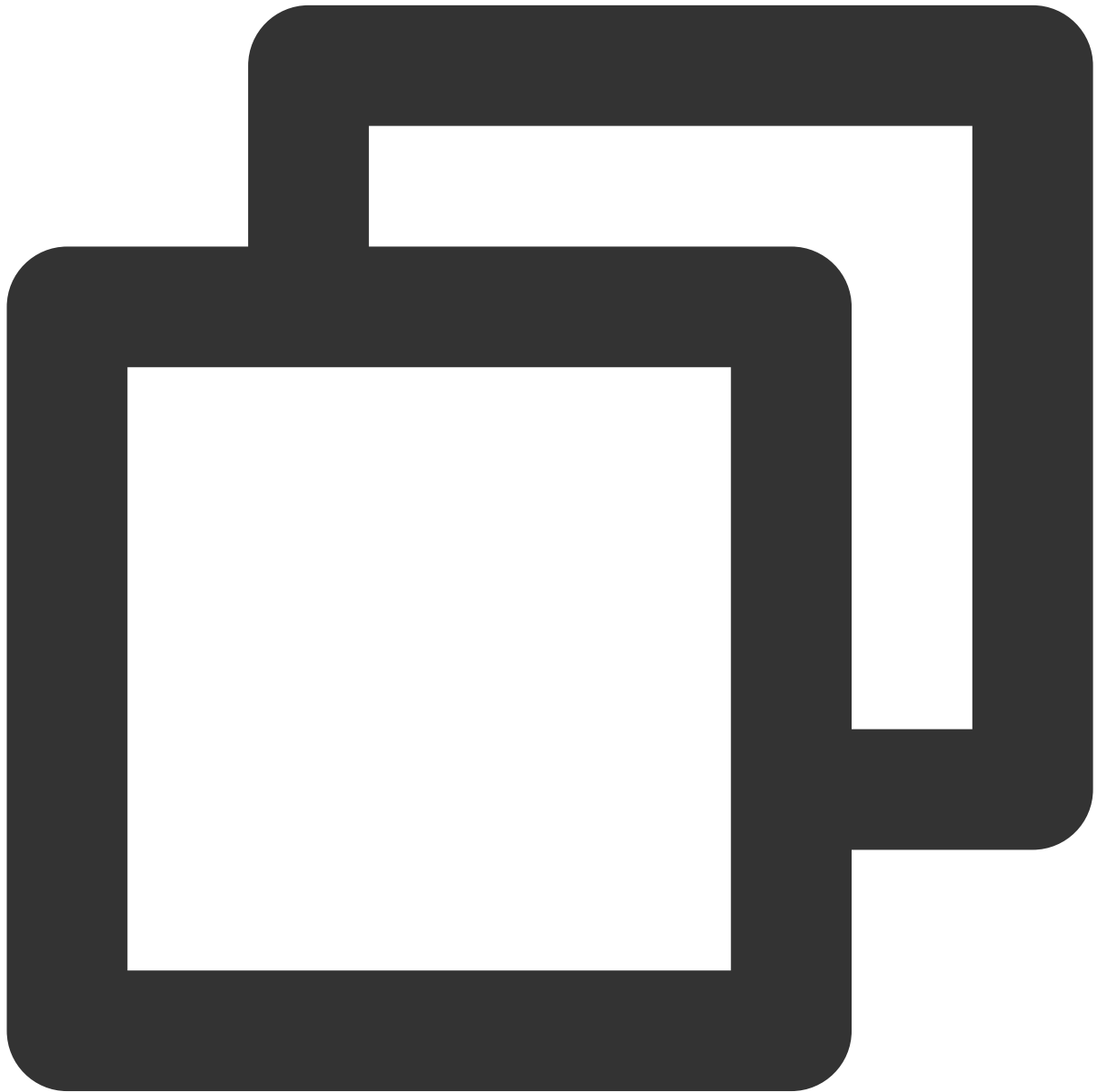
Parameter	Type	Meaning
code	Number	<a href="#">Error Code</a>



msg	String	Error message
-----	--------	---------------

## SDK\_READY

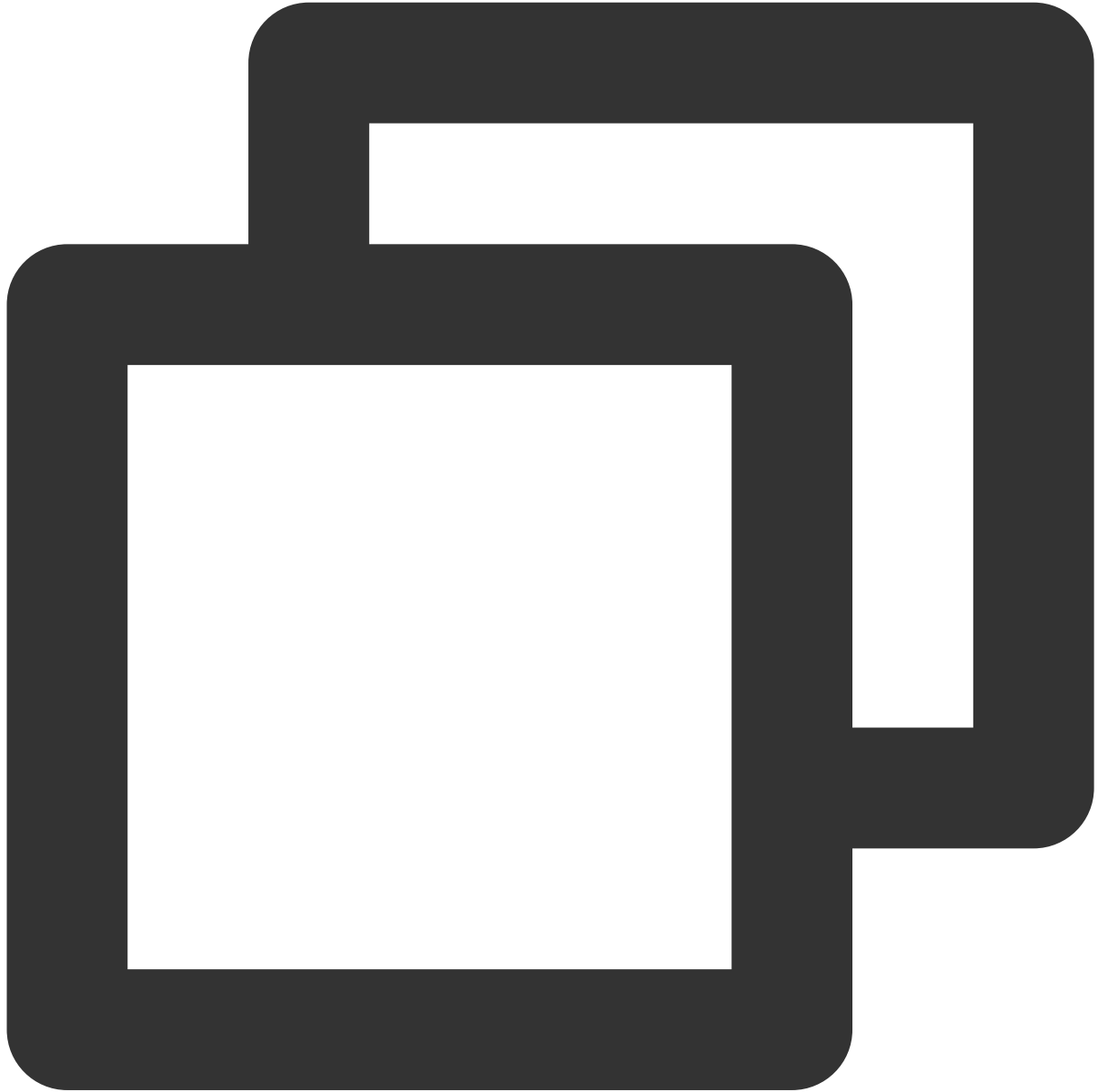
TUICallEngine relies on [@tencentcloud/chat](#) SDK. The [SDK\\_READY](#) event will be triggered only after successful login, and then you can use various functions of the SDK.



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

## KICKED\_OUT

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

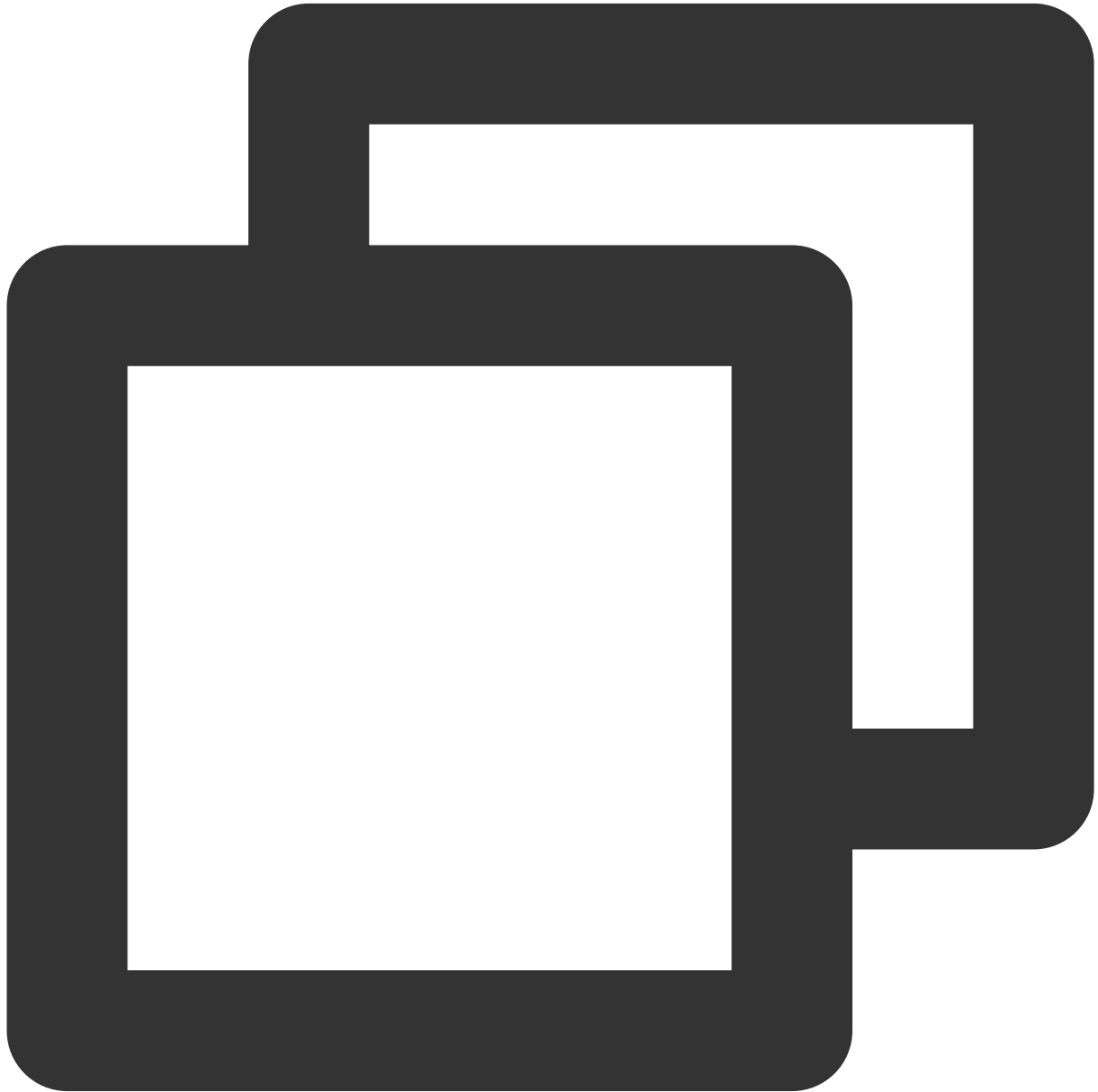


```
let handleOnKickedOut = function(event) {  
  console.log(event);  
};  
tuiCallEngine.on(TUICallEvent.KICKED_OUT, handleOnKickedOut);
```

## USER\_ACCEPT

If a user answers, all other users will receive this event, where `userID` is the user who answered.

1. In a 1v1 call: when the callee answers, the caller will throw this event.
2. In group calls: if A calls B and C, and B answers, both A and C will throw this event, with the event's `userID` being B. Similarly, if C answers, both A and B will throw this event, with the event's `userID` being C.



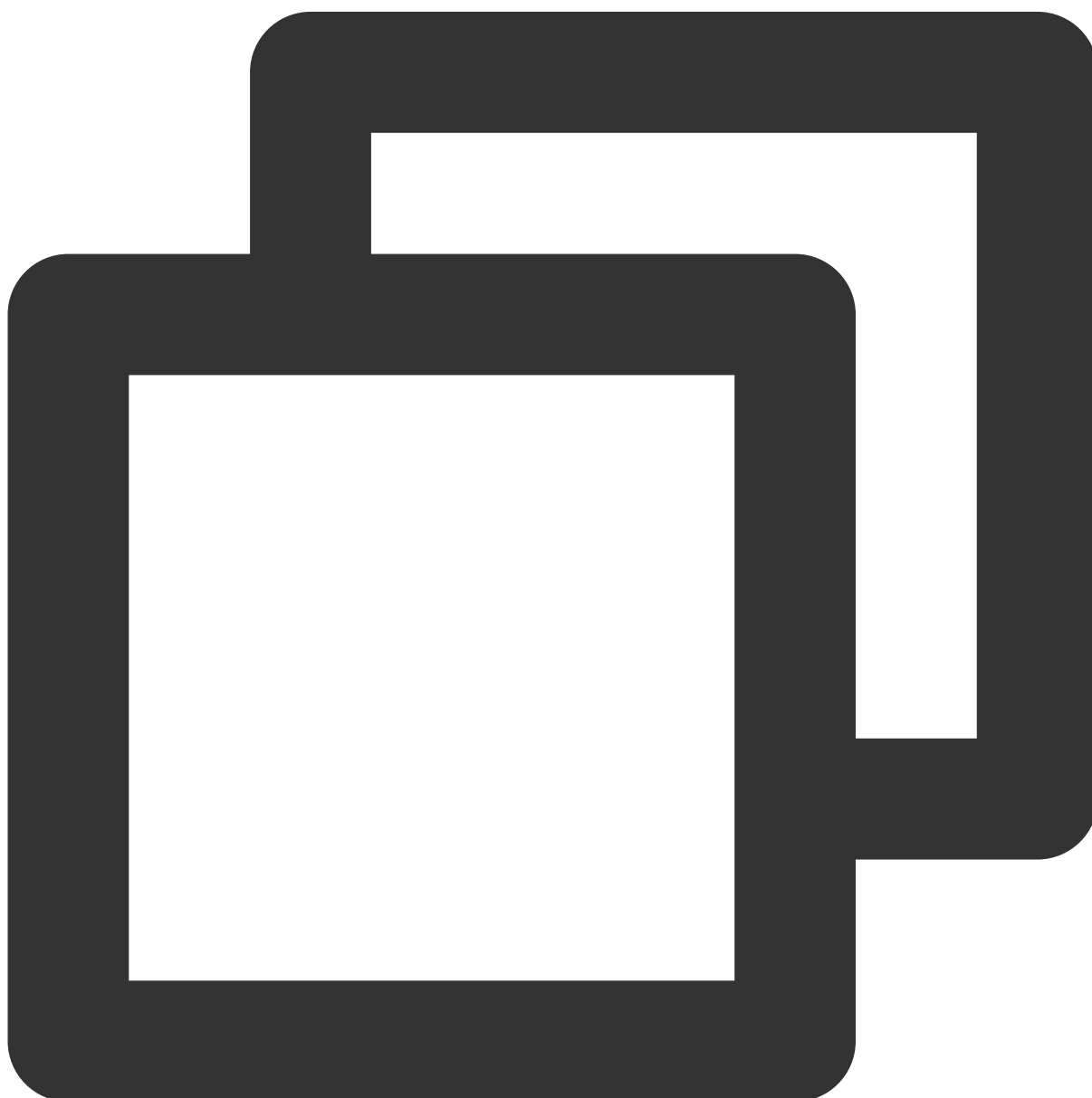
```
let handleUserAccept = function(event) {  
  console.log(event.userID);  
};  
tuiCallEngine.on(TUICallEvent.USER_ACCEPT, handleUserAccept);
```

The parameters are described below:

Parameter	Type	Meaning
userID	String	Answering User ID

## USER\_ENTER

If a user enters the call, other users will throw this event, and userID is the user name who entered the call.



```
let handleUserEnter = function(event) {
```

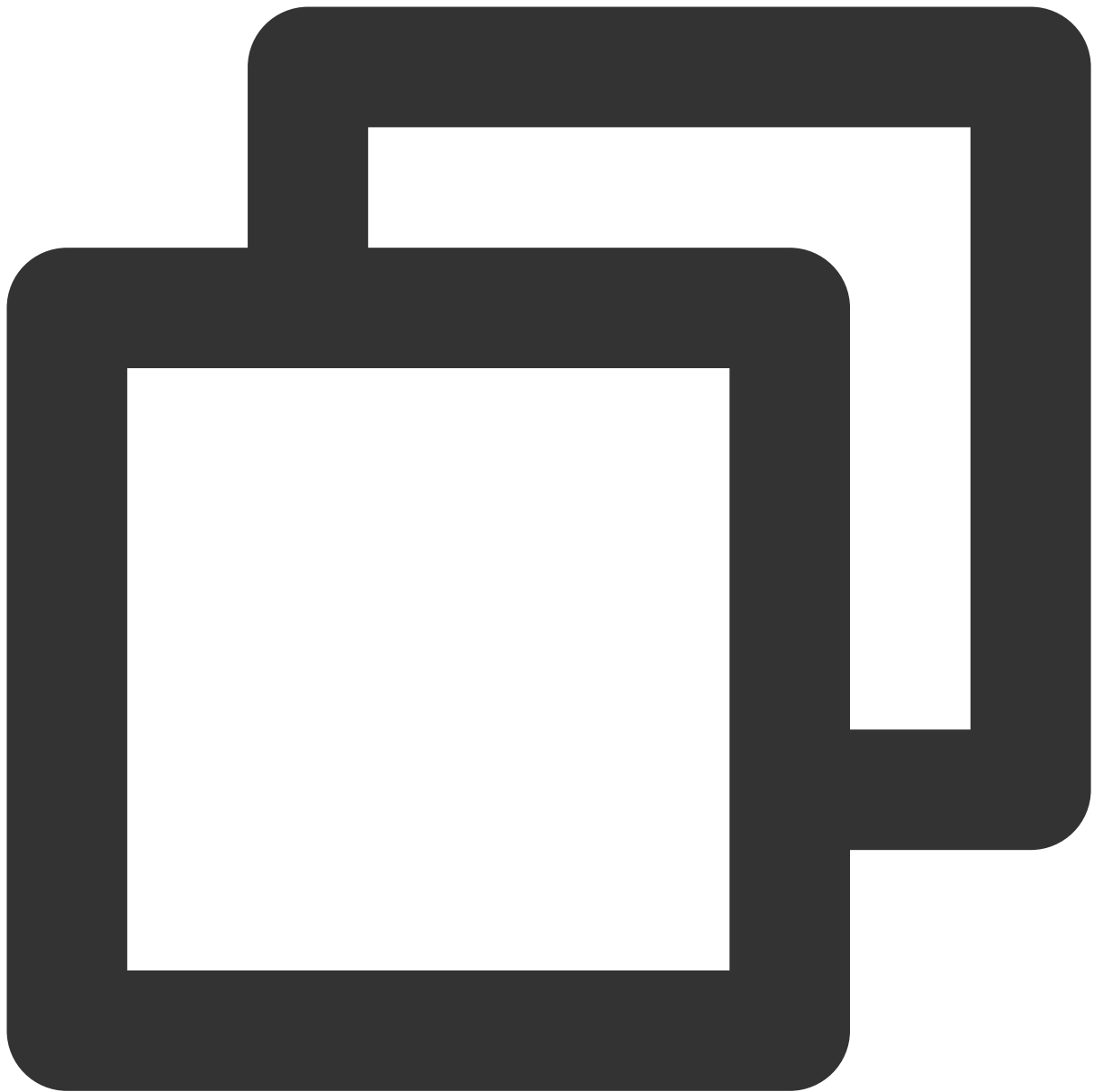
```
console.log(event.userID);  
};  
tuiCallEngine.on(TUICallEvent.USER_ENTER, handleUserEnter);
```

The parameters are described below:

Parameter	Type	Meaning
userID	String	Entering User ID

## USER\_LEAVE

When a user leaves the call, this event will be thrown by other users in the call. The userID is the name of the user who left the call.



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

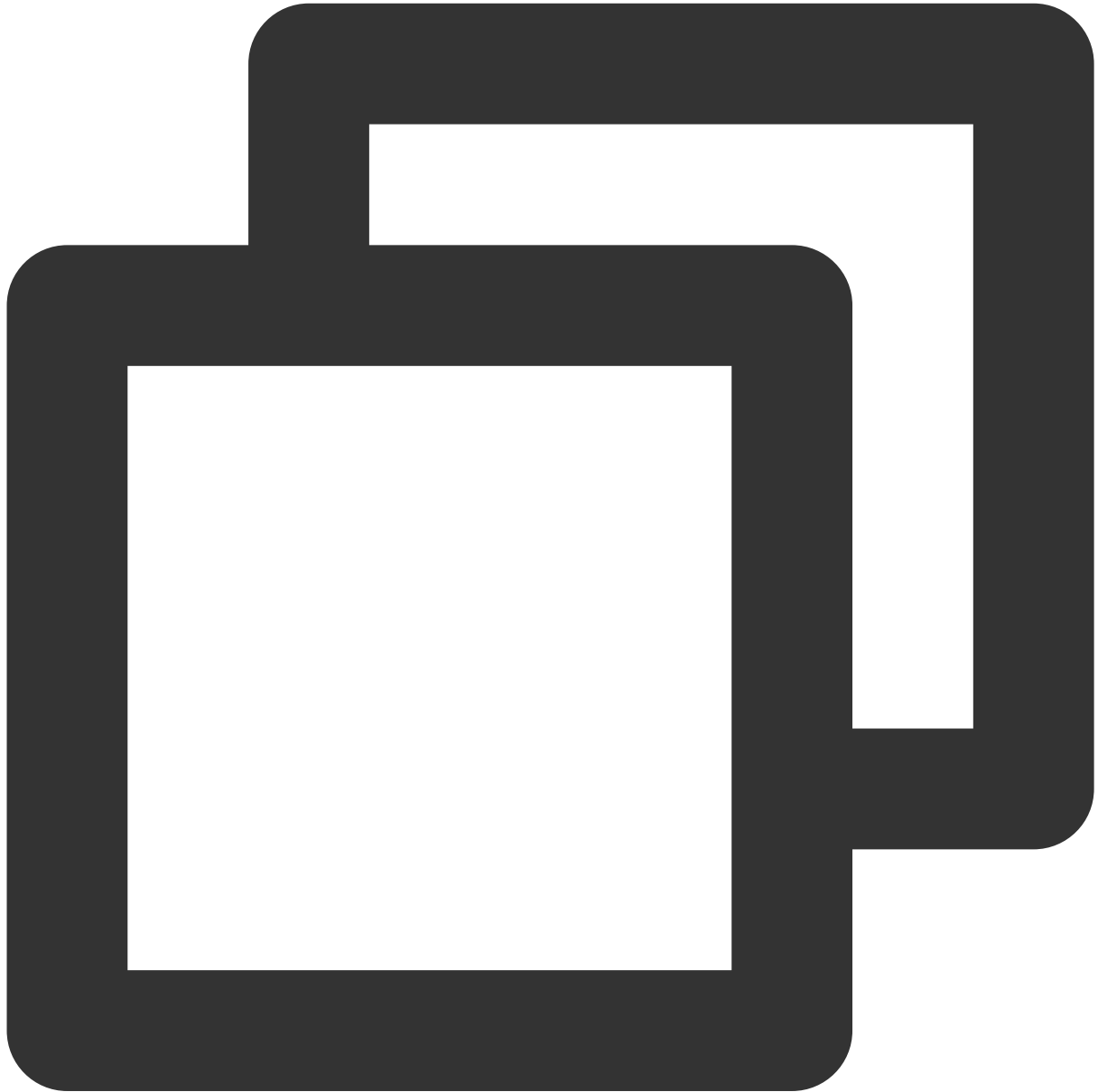
The parameters are described below:

Parameter	Type	Meaning
userID	String	Exiting User ID

## REJECT

This event is thrown when the call is rejected

1. In a 1v1 call, only the calling party will receive the rejection event, and userID is the called username.
2. In a group call, when an invitee refuses the call, this event will be thrown by other people in the group call. The userID is the name of the user who refused the call.



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

The parameters are described below:

Parameter	Type	Meaning
userID	String	Rejecting User ID

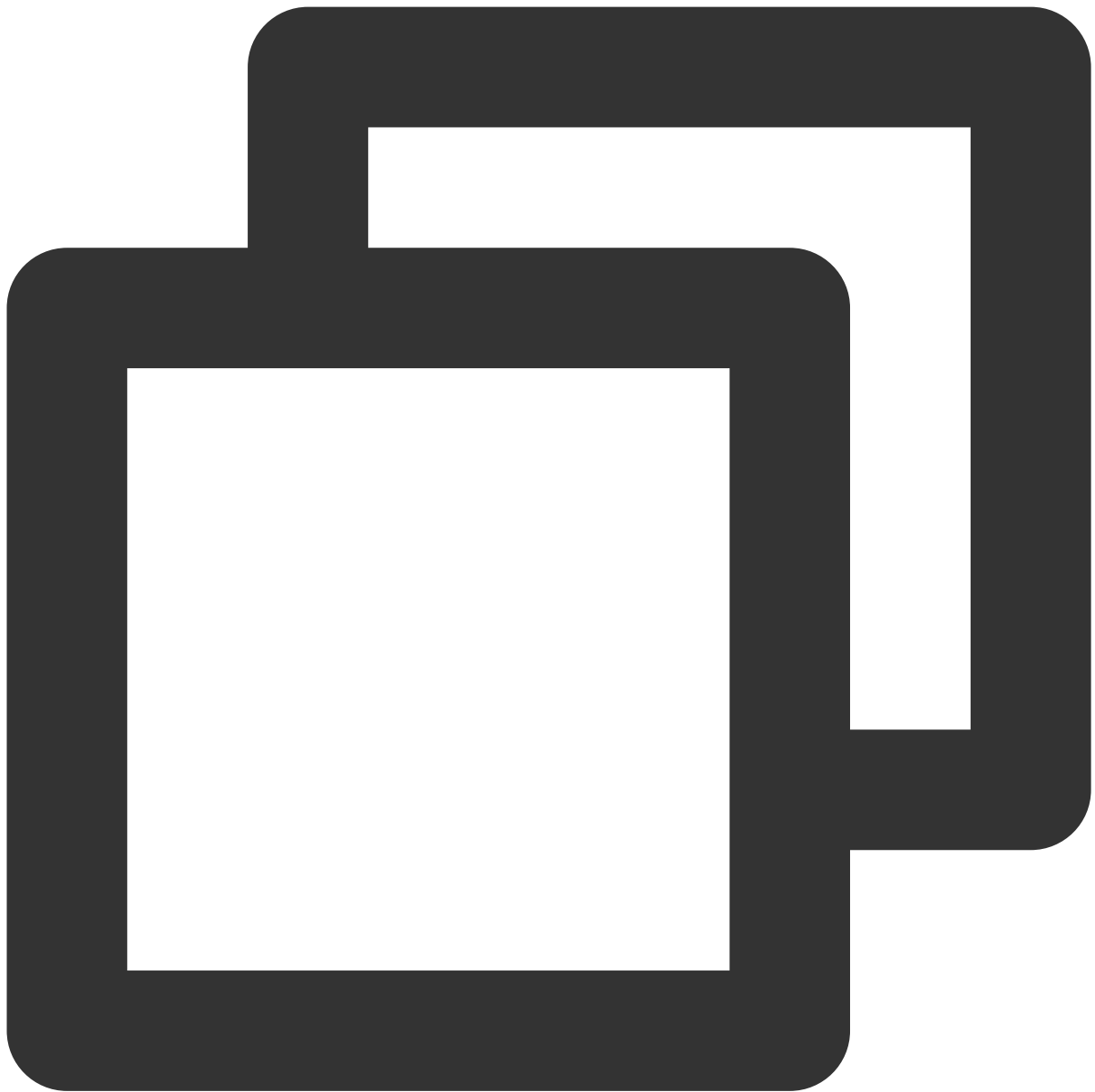
## NO\_RESP

This event will be thrown by other calling users when the callee does not respond.

In a 1v1 call, only the initiator will receive the event of no answer. For example, A invites B, B does not answer, A can receive this event.

In a group call, when an invitee does not respond, this event will be thrown by everyone else in the group call. For example, if A invites B and C to join the call, but B does not respond, both A and C will throw this event.





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

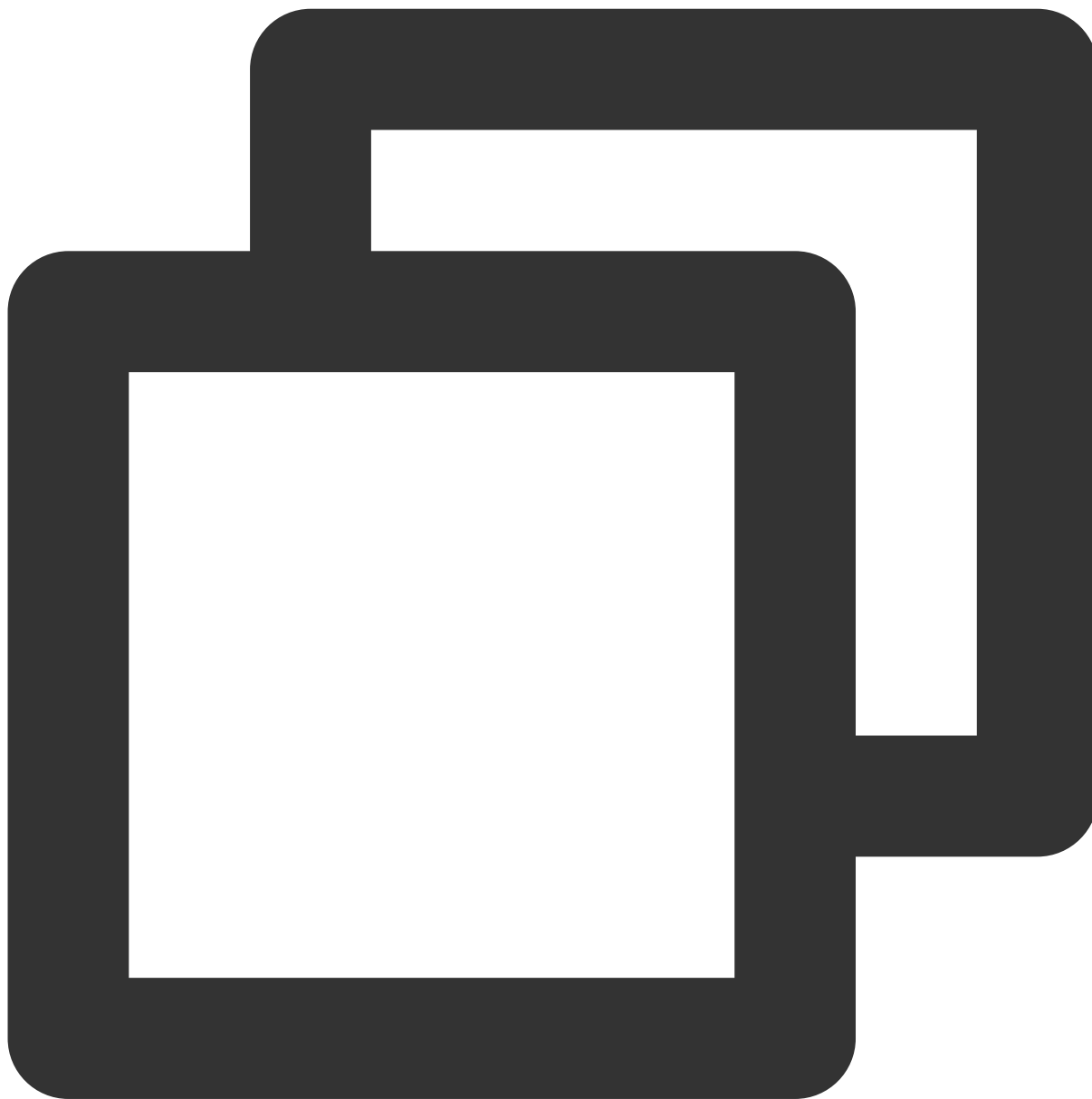
The parameters are described below:

Parameter	Type	Meaning
sponsor	String	Caller's User ID

userIDList	Array<String>	List of Users Who Triggered Timeout Due to No Response
------------	---------------	--

## LINE\_BUSY

Call busy event. For example: when B is on a call, and A calls B, A will throw an event.



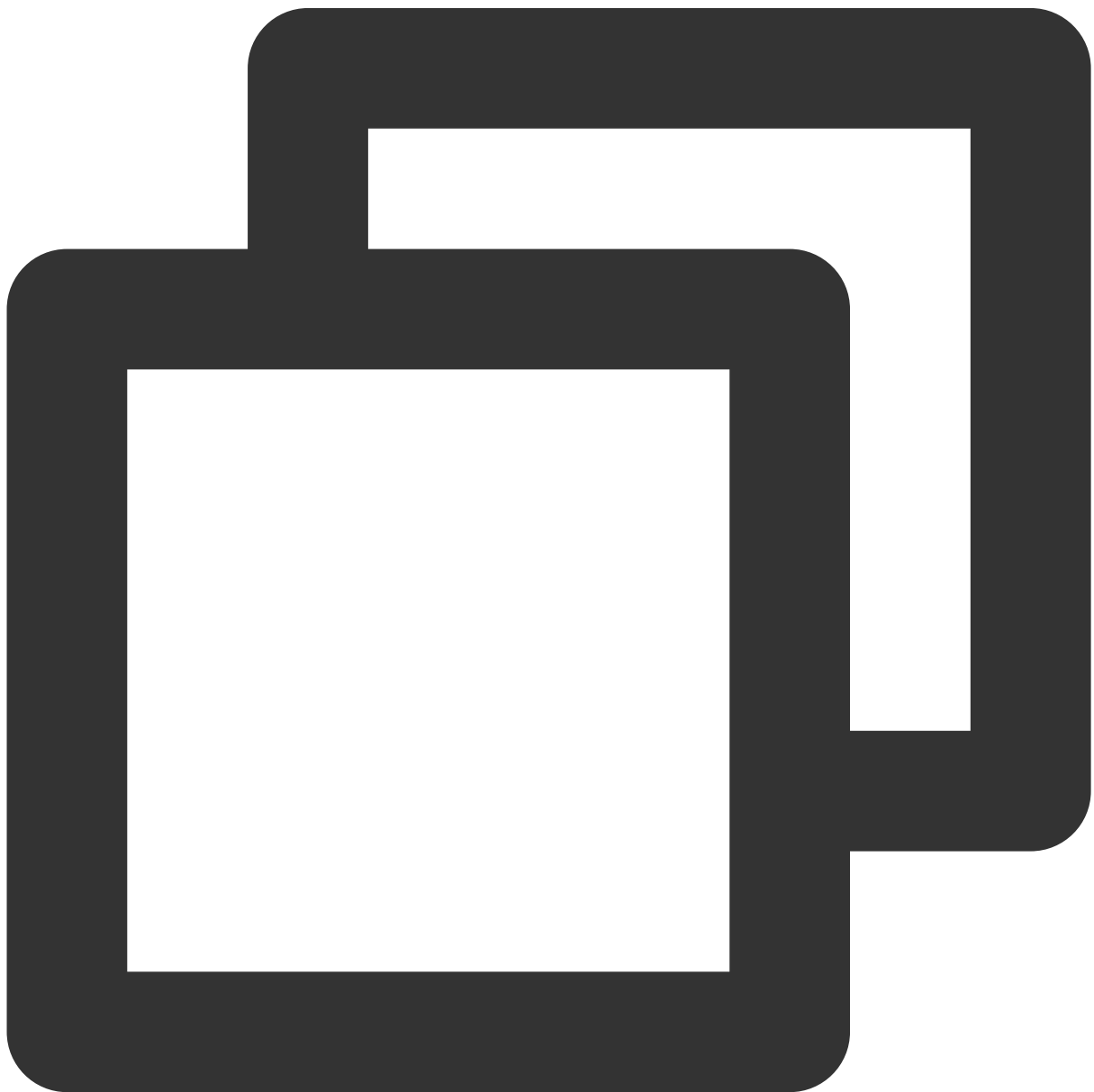
```
let handleLineBusy = function(event) {  
    console.log(event);  
};  
tuiCallEngine.on(TUICallEvent.LINE_BUSY, handleLineBusy);
```

The parameters are described below:

Parameter	Type	Meaning
userID	String	Busy User ID

## USER\_VIDEO\_AVAILABLE

If a user turns on/off the camera during a video call, this event will be thrown by other users in the call. For example: A and B are on a video call, A turns on/off the camera, and B will throw this event.



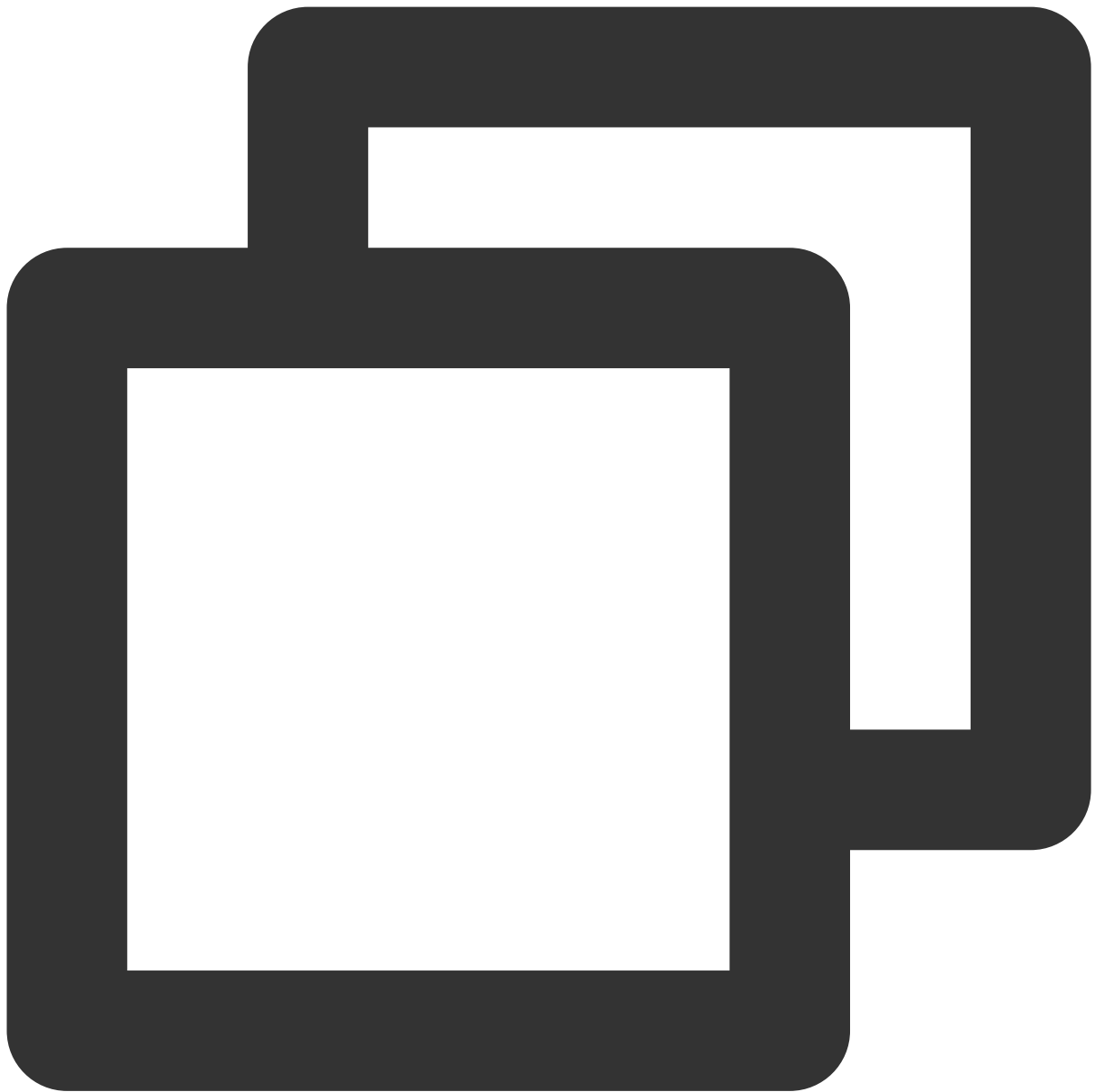
```
let handleUserVideoChange = function(event) {  
    console.log(event.userID, event.isVideoAvailable);  
};  
tuiCallEngine.on(TUICallEvent.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

The parameters are described below:

Parameter	Type	Meaning
userID	String	Remote User ID
isVideoAvailable	Boolean	true: Remote User turns Camera On; false: Remote User turns Camera Off

## USER\_AUDIO\_AVAILABLE

If a user turns on/off the microphone during an audio or video call, this event will be thrown by other users on the call. For example: A and B are having an audio and video call, and A turns on/off the microphone, and B will throw this event.



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

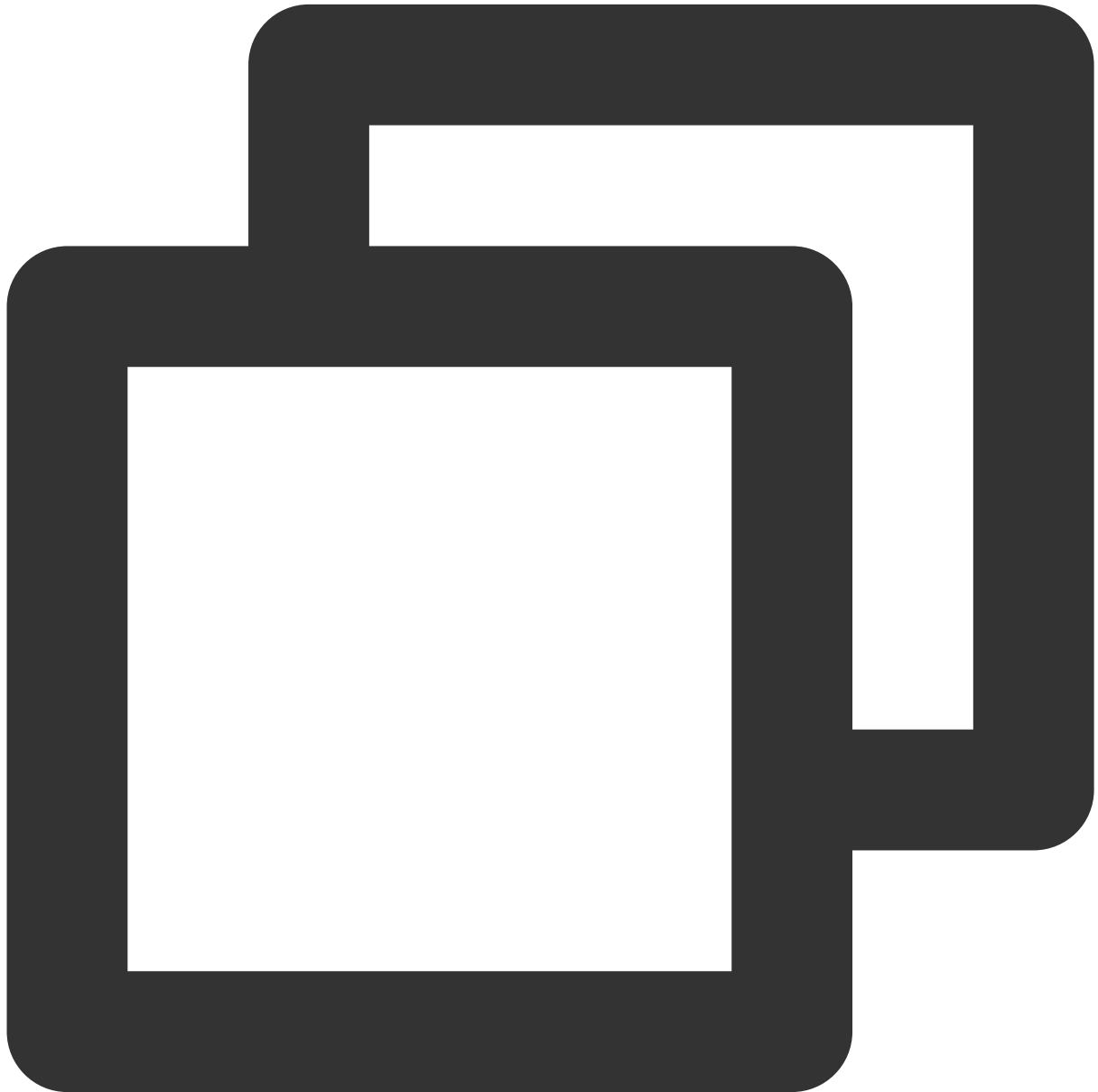
The parameters are described below:

Parameter	Type	Meaning
userID	String	User ID to turn microphone on/off

isAudioAvailable	Boolean	true the user turns on the microphone; false the user turns off the microphone
------------------	---------	--

## USER\_VOICE\_VOLUME

When the user's volume changes during an audio or video call, this event will be thrown by other users on the call. For example: A and B are having an audio and video call, and if A's volume changes, B will throw this event.



```
let handleUserVoiceVolumeChange = function(event) {  
  console.log(event.volumeMap);  
};
```

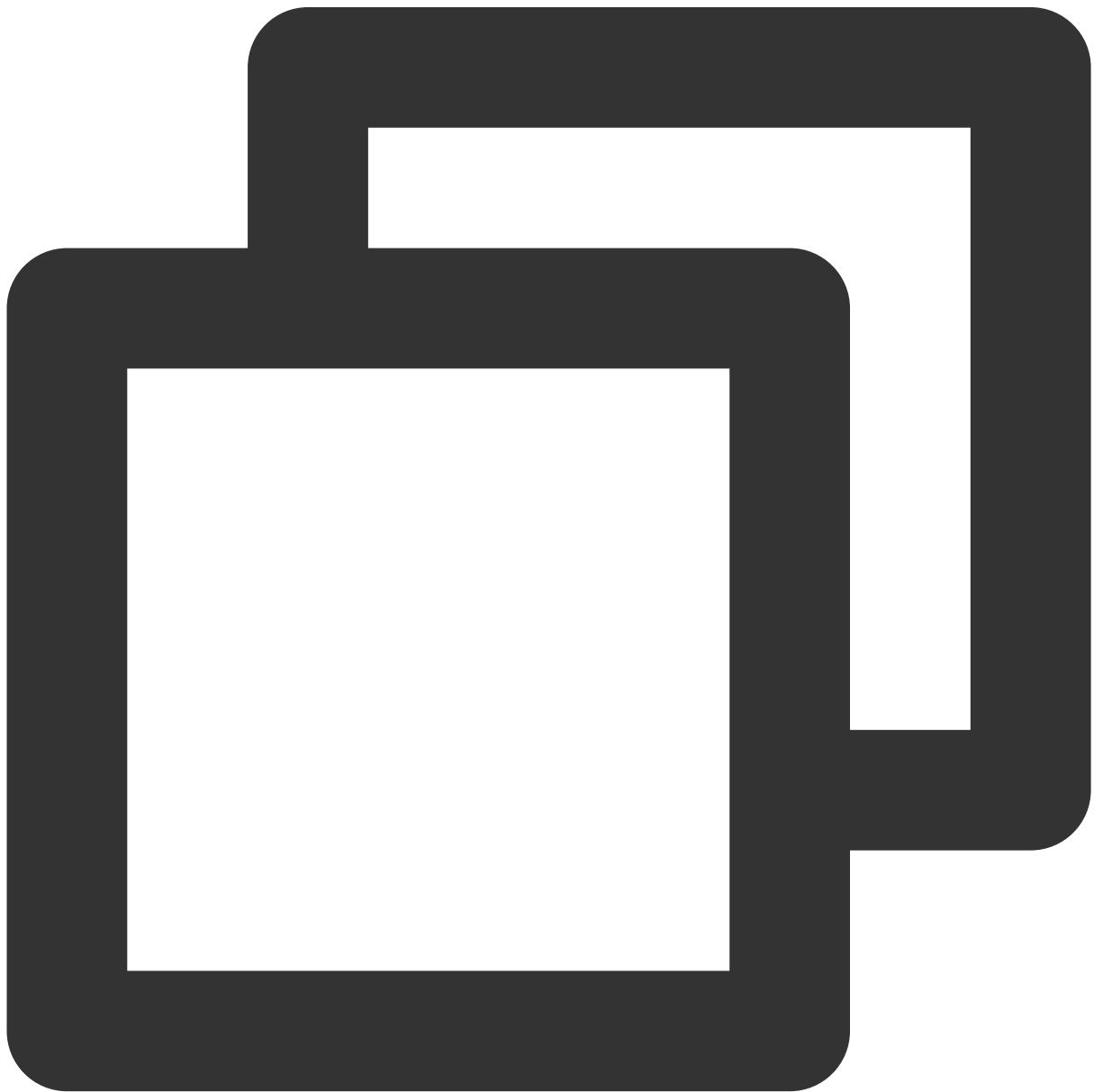
```
tuiCallEngine.on(TUICallEvent.USER_VOICE_VOLUME, handleUserVoiceVolumeChange);
```

The parameters are described below:

Parameter	Type	Meaning
volumeMap	Array<Object>	Volume meter, the corresponding volume can be obtained according to each userid, volume range: [0, 100]

## GROUP\_CALL\_INVITEE\_LIST\_UPDATE

Group chat update invitation list, this event will be received.



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

The parameters are described below:

Parameter	Type	Meaning
userIDList	Array<String>	Group update invitation list

## INVITED

Receiving a new incoming call event, the called party will be notified. By listening to this event, you can decide whether to display the call answering interface.

### Note:

**Plan to deprecate in subsequent versions. Recommended:** [ON\\_CALL\\_RECEIVED](#).

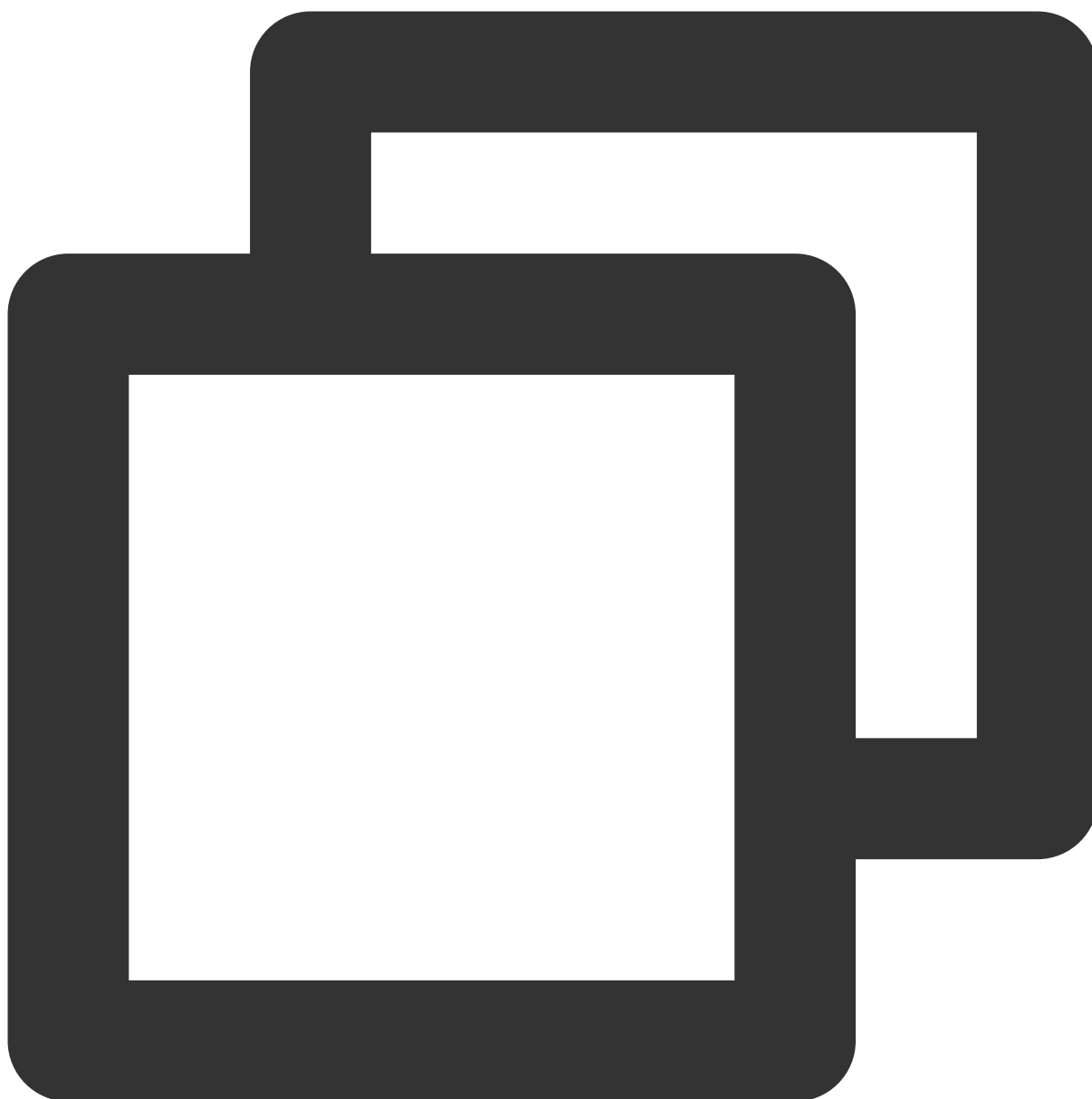
## ON\_CALL\_RECEIVED

Receiving a new incoming call event, the called party will be notified. By listening to this event, you can decide whether to display the call answering interface.

### Note:

**v1.4.6+ supported.**





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

The parameters are described below:

Parameter	Type	Meaning
sponsor	String	Inviter

userIDList	Array<String>	Also Invited Persons
isFromGroup	Boolean	Is it a Group Call
inviteData	Object	Call Data
inviteID	String	Invitation ID, identifying one invitation
userData	String	Extended field: Utilized for amplifying details in the invitation signaling
callId	String	Unique ID for this call
roomId	Number	Audio-Video Room ID for this call
callMediaType	Number	Media Type of the call, Video Call, Voice Call
callRole	String	role, Enumeration Type: Caller, Called

## CALLING\_CANCEL

**If the call is not established, this event will be thrown.** By listening to this event, you can implement display logic similar to missed calls, reset UI state, etc. Scenarios where the call is not established include:

### Note:

**Plan to deprecate in subsequent versions. Recommended:** [ON\\_CALL\\_CANCELED](#).

## ON\_CALL\_CANCELED

**If the call is not established, this event will be thrown.** By listening to this event, you can implement display logic similar to missed calls, reset UI state, etc. Scenarios where the call is not established include:

Caller Cancelled: The caller throws this event, userID is the caller; the called also throws this event, userID is the called;

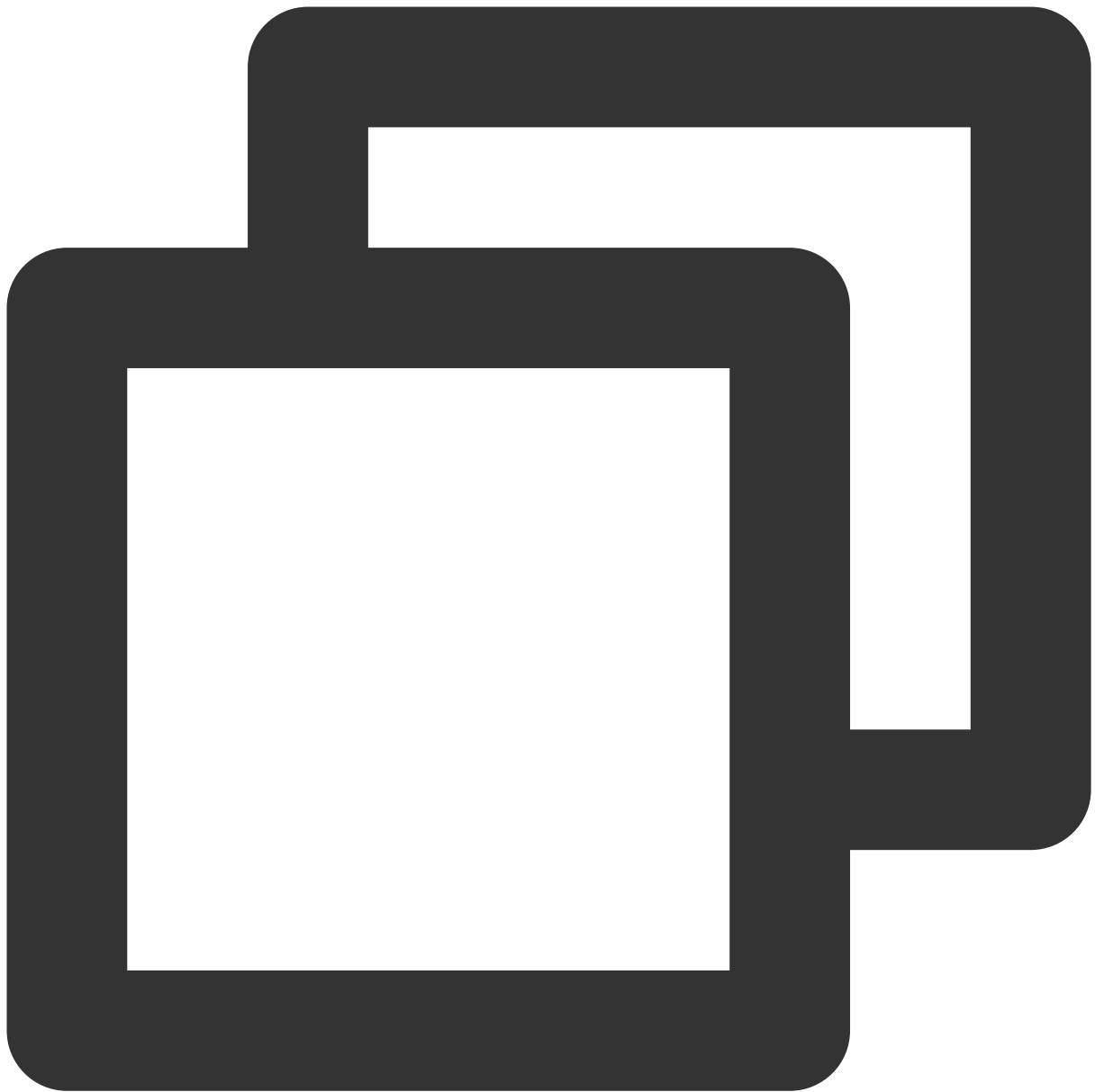
Callee Timeout: The caller will throw both [NO\\_RESP](#) and [CALLING\\_CANCEL](#) events, userID is the caller; the called throws the [CALLING\\_CANCEL](#) event, userID is the called;

Callee Rejected: The caller will throw both [REJECT](#) and [CALLING\\_CANCEL](#) events, userID is the caller; the called throws the [CALLING\\_CANCEL](#) event, userID is the called;

Callee Busy: The caller will throw both [LINE\\_BUSY](#) and [CALLING\\_CANCEL](#) events, userID is the caller; the callee throws the [CALLING\\_CANCEL](#) event, userID is the callee;

### Note:

**Supported from version v1.4.6+ .**



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

The parameters are described below:

Parameter	Type	Meaning
userID	String	Cancelled User ID

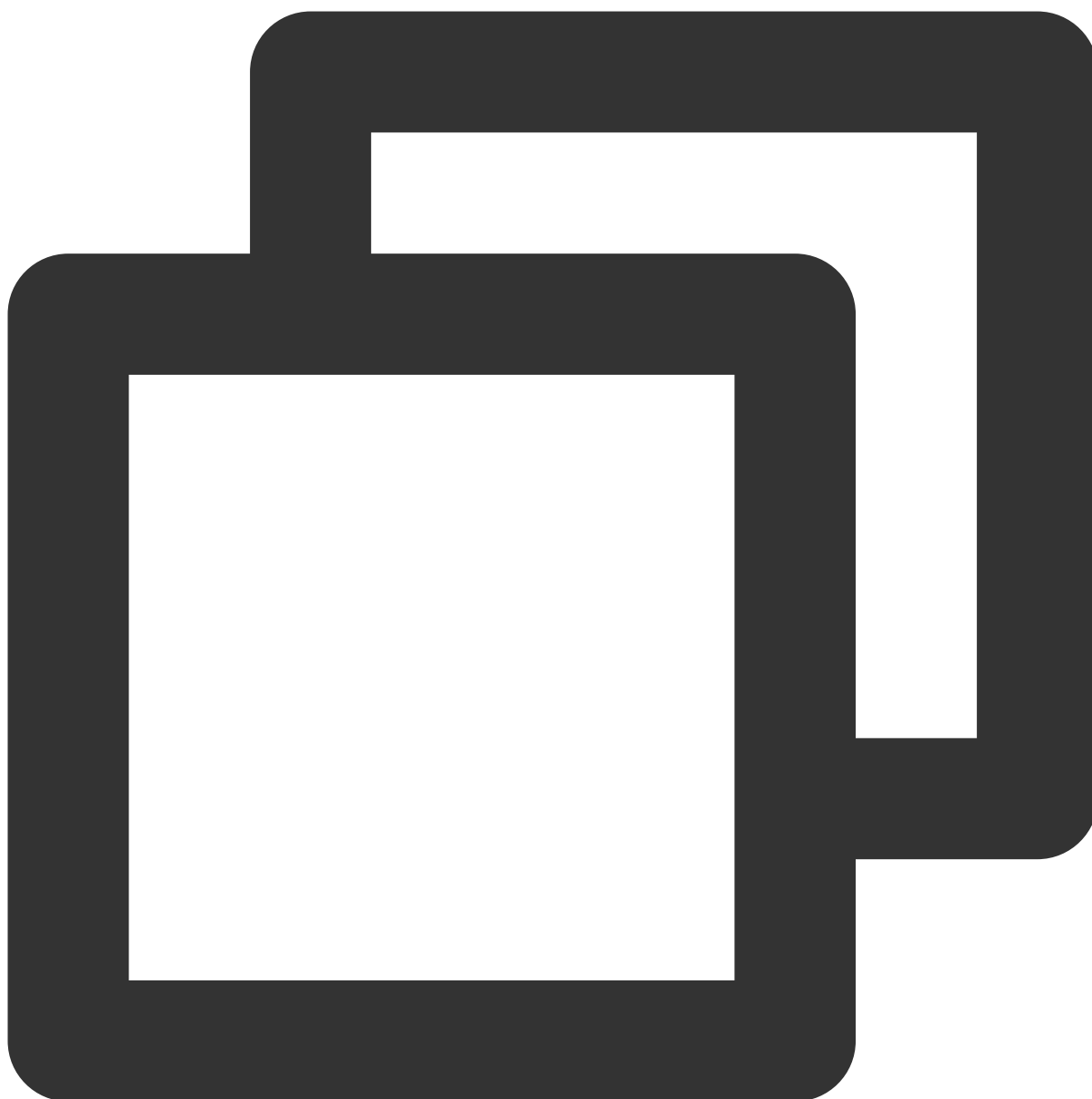
callId	String	Unique ID for this call
roomId	Number	Audio-Video Room ID for this call
callMediaType	Number	Media Type of the call, Video Call, Voice Call
callRole	String	role, Enumeration Type: Caller, Called

## ON\_CALL\_BEGIN

Indicates call connection. Both caller and called can receive it. You can start cloud recording, content review, etc., by listening to this event.

### Note:

**Supported from version v1.4.6+ .**



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

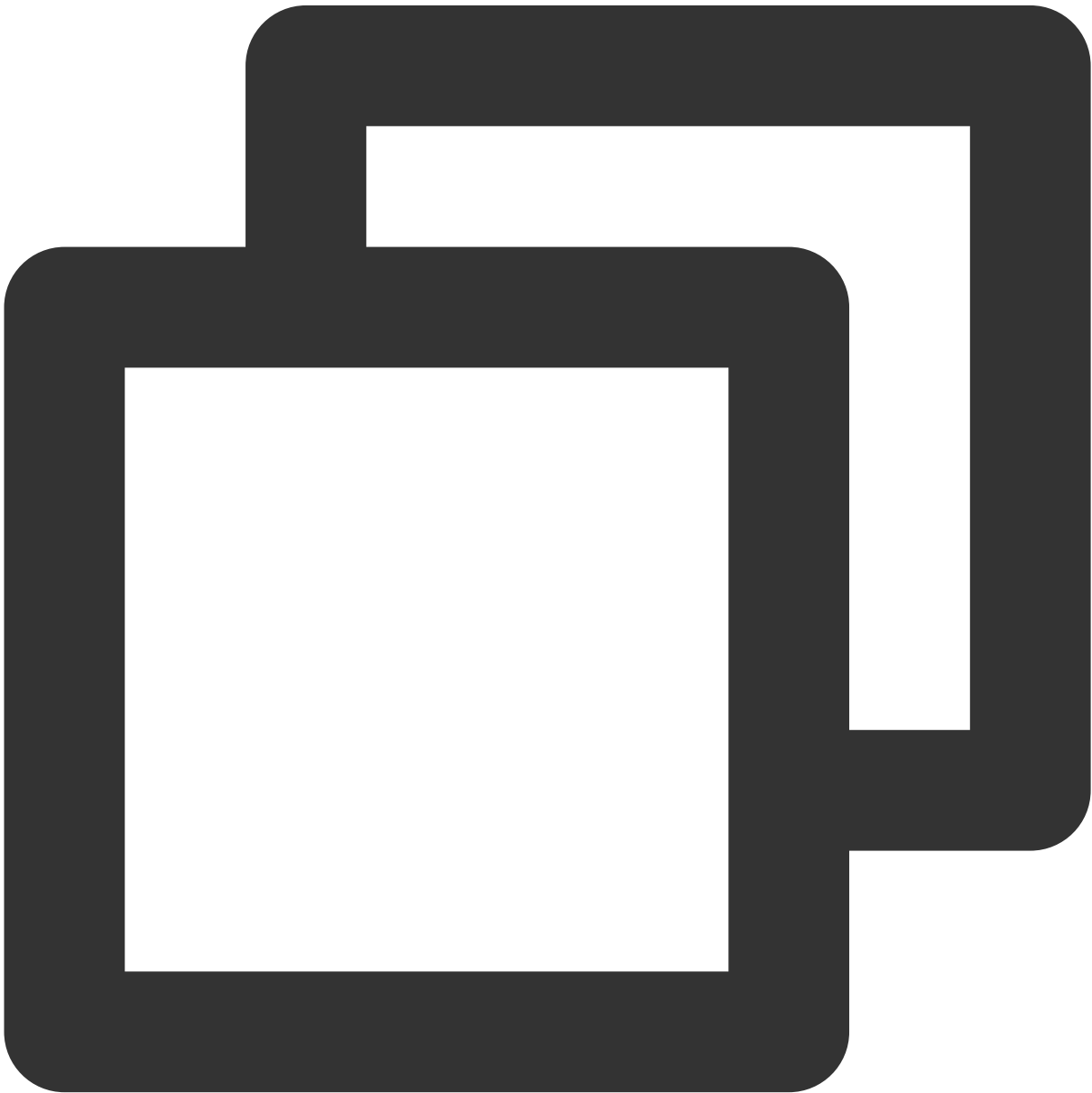
The parameters are described below:

Parameter	Type	Meaning
callId	String	Unique ID for this call

roomID	Number	Audio-Video Room ID for this call
callMediaType	Number	Media Type of the call, Video Call, Voice Call
callRole	String	role, Type: Caller, Called

### CALLING\_END

Indicates call termination. Both caller and called can trigger this event. You can display information such as call duration, call type, or stop the cloud recording process by listening to this event.



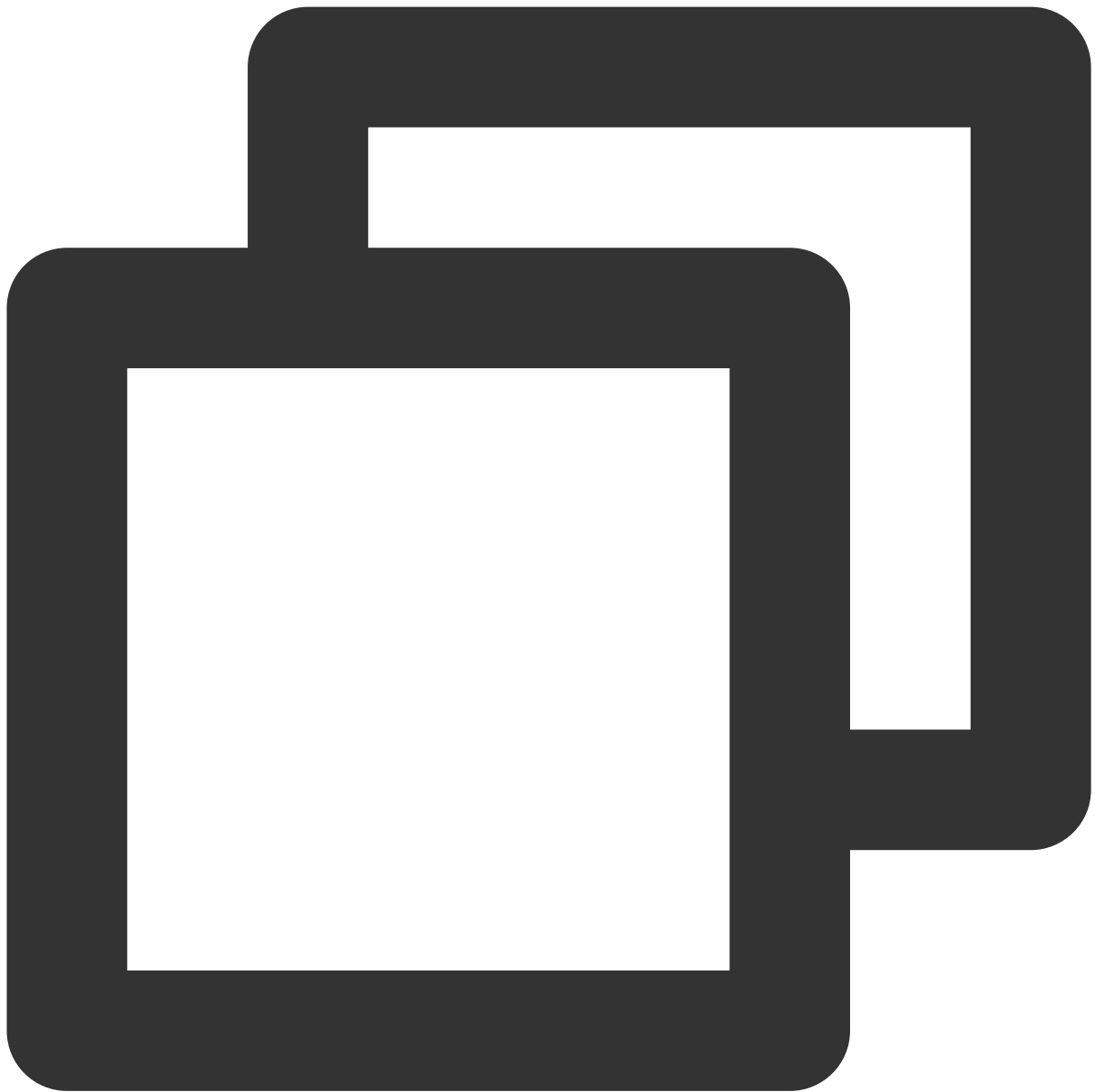
```
let handleCallingEnd = function(event) {  
    console.log(event.userID, event.);  
};  
tuiCallEngine.on(TUICallEvent.CALLING_END, handleCallingEnd);
```

The parameters are described below:

Parameter	Type	Meaning
roomID	Number	Audio-Video Room ID for this call, currently only supports numeric room number, future versions will support character string room numbers
callMediaType	Number	Media Type of the call, Video Call, Voice Call
callRole	String	role, Enumeration Type: Caller ('inviter'), Called ('invitee'), Unknown ('')
totalTime	Number	The duration of this call in seconds
userID	String	userID of the call termination.
callId	String	The unique ID for this call. <b>v1.4.6+ Supported</b>
callEnd	Number	The duration of this call ( <b>will be deprecated, Please use totalTime</b> ) in seconds

## DEVICED\_UPDATED

Device list update, this event will be received.

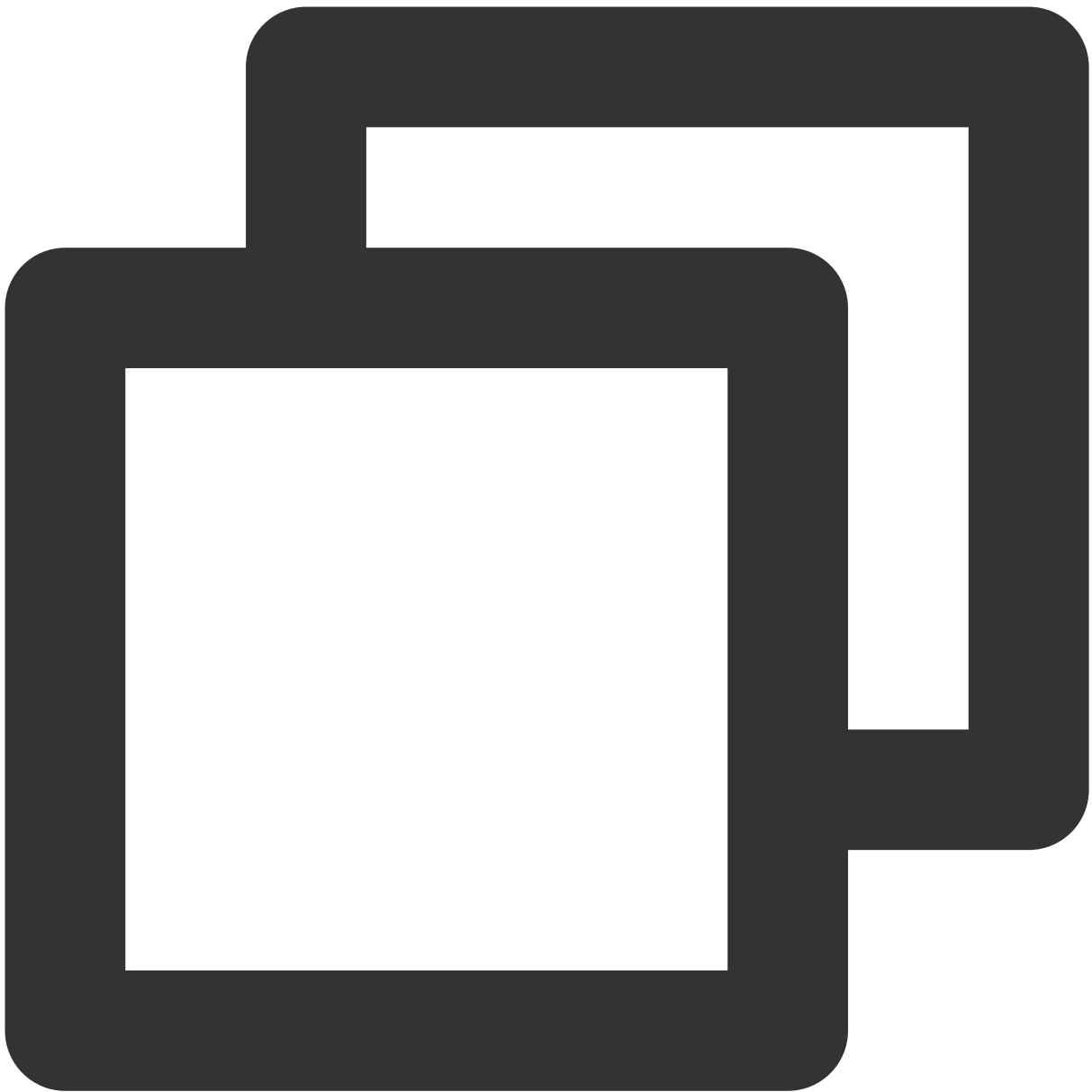


```
let handleDeviceUpdated = function({ microphoneList, cameraList, currentMicrophoneID, currentCameraID }) {  
  console.log(microphoneList, cameraList, currentMicrophoneID, currentCameraID);  
};  
tuiCallEngine.on(TUICallEvent.DEVICED_UPDATED, handleDeviceUpdated);
```

## CALL\_TYPE\_CHANGED

Call type switching, this event will be received.





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

The parameters are described below:

Parameter	Type	Meaning
oldCallType	Number	Old call type

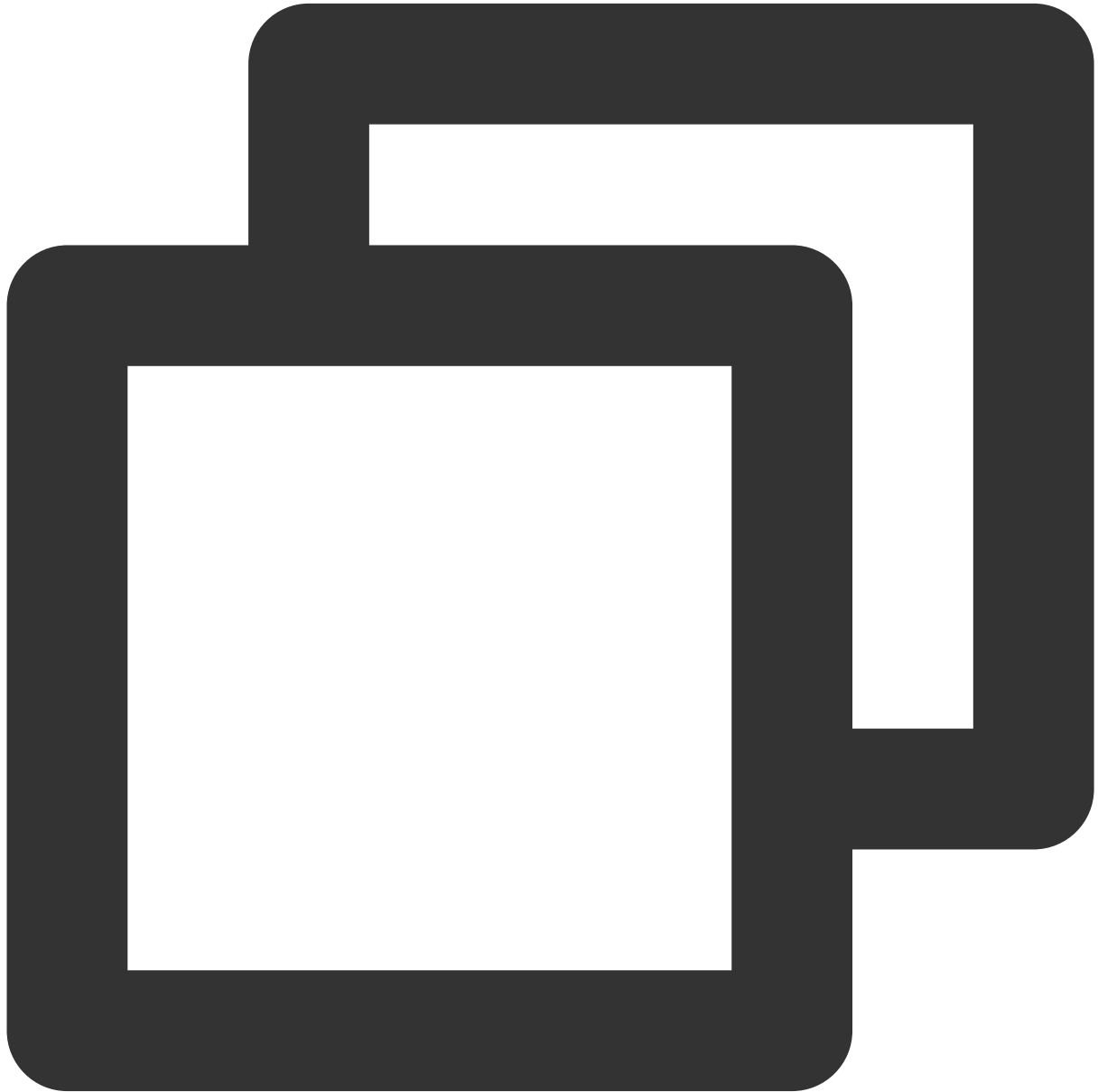
newCallType	Number	New call type
-------------	--------	---------------

## ON\_USER\_NETWORK\_QUALITY\_CHANGED

All user network quality events

**Note :**

**v3.0.7+ supported.**



```
let handleOnUserNetworkQualityChange = function(event) {  
  console.log(event.networkQualityList);  
};
```

```
tuiCallEngine.on(TUICallEvent.ON_USER_NETWORK_QUALITY_CHANGED, handleOnUserNetworkQ
```

The parameters are described below:

Parameter	Type	Meaning
networkQualityList	Array<Object>	Network status, according to userID, you can get the current network quality of the corresponding user.

# 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
<a href="#">login</a>	Login
<a href="#">logout</a>	Log out
<a href="#">setSelfInfo</a>	Sets the alias and profile photo.
<a href="#">call</a>	Makes a one-to-one call.
<a href="#">groupCall</a>	Makes a group call.
<a href="#">joinInGroupCall</a>	Joins a group call.
<a href="#">enableMuteMode</a>	Sets whether to turn on the mute mode.
<a href="#">enableFloatWindow</a>	Sets whether to enable floating windows.
<a href="#">setCallingBell</a>	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
<a href="#">init</a>	Authenticates the basic audio/video call capabilities.
<a href="#">unInit</a>	The destructor function, which releases resources used by TUICallEngine.
<a href="#">addObserver</a>	Registers an event listener.

<code>removeObserver</code>	Unregisters an event listener.
<code>call</code>	Makes a one-to-one call.
<code>groupCall</code>	Makes a group call.
<code>accept</code>	Accepts a call.
<code>reject</code>	Rejects a call.
<code>hangup</code>	Ends a call.
<code>ignore</code>	Ignores a call.
<code>inviteUser</code>	Invites users to the current group call.
<code>joinInGroupCall</code>	Joins a group call.
<code>switchCallMediaType</code>	Changes the call type, for example, from video call to audio call.
<code>startRemoteView</code>	Subscribes to the video stream of a remote user.
<code>stopRemoteView</code>	Unsubscribes from the video stream of a remote user.
<code>openCamera</code>	Turns the camera on.
<code>closeCamera</code>	Turns the camera off.
<code>switchCamera</code>	Switches between the front and rear cameras.
<code>openMicrophone</code>	Turns the mic on.
<code>closeMicrophone</code>	Turns the mic off.
<code>selectAudioPlaybackDevice</code>	Selects the audio playback device (receiver or speaker).
<code>setSelfInfo</code>	Sets the alias and profile photo.
<code>enableMultiDeviceAbility</code>	Sets whether to enable multi-device login for TUICallEngine (supported by the premium package).
<code>setVideoRenderParams</code>	Set the rendering mode of video image.
<code>setVideoEncoderParams</code>	Set the encoding parameters of video encoder.
<code>queryRecentCalls</code>	Query call record.
<code>deleteRecordCalls</code>	Delete call record.
<code>setBeautyLevel</code>	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
<code>onError</code>	An error occurred during the call.
<code>onCallReceived</code>	A call was received.
<code>onCallCancelled</code>	The call was canceled.
<code>onCallBegin</code>	The call was connected.
<code>onCallEnd</code>	The call ended.
<code>onCallMediaTypeChanged</code>	The call type changed.
<code>onUserReject</code>	A user declined the call.
<code>onUserNoResponse</code>	A user didn't respond.
<code>onUserLineBusy</code>	A user was busy.
<code>onUserJoin</code>	A user joined the call.
<code>onUserLeave</code>	A user left the call.
<code>onUserVideoAvailable</code>	Whether a user has a video stream.
<code>onUserAudioAvailable</code>	Whether a user has an audio stream.
<code>onUserVoiceVolumeChanged</code>	The volume levels of all users.
<code>onUserNetworkQualityChanged</code>	The network quality of all users.
<code>onKickedOffline</code>	The current user is kicked offline
<code>onUserSigExpired</code>	Ticket expires while online

# TUICallKit

Last updated : 2024-04-15 17:41:03

## TUICallKit API

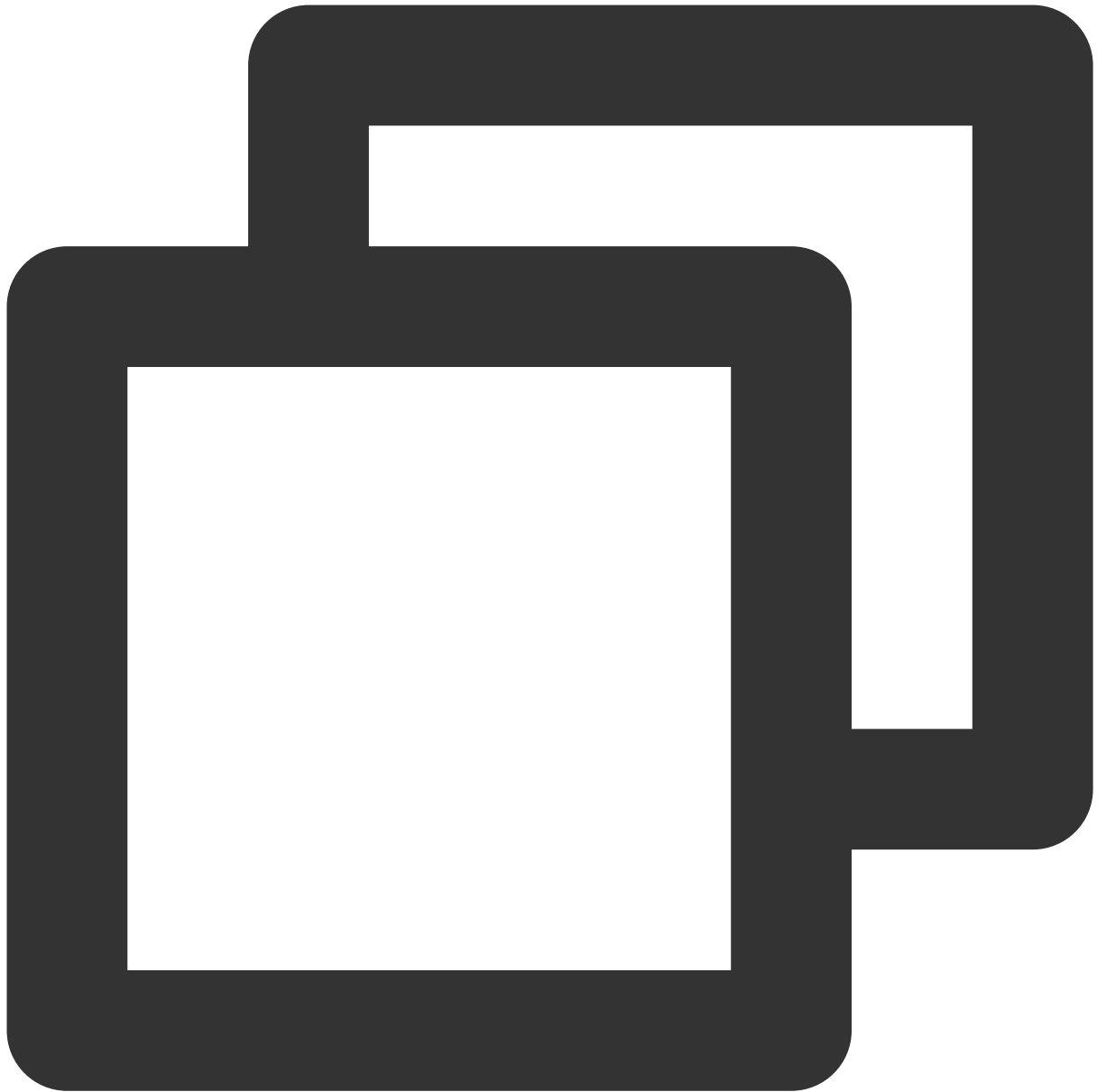
`TUICallKit` is an audio/video call component that **includes UI elements**. You can use its APIs to quickly implement an audio/video call application similar to WeChat. For directions on integration, see [Integrating TUICallKit](#).

## API Overview

API	Description
<a href="#">login</a>	Login
<a href="#">logout</a>	Log out
<a href="#">setSelfInfo</a>	Sets the alias and profile photo.
<a href="#">call</a>	Makes a one-to-one call.
<a href="#">groupCall</a>	Makes a group call.
<a href="#">joinInGroupCall</a>	Joins a group call.
<a href="#">enableMuteMode</a>	Sets whether to turn on the mute mode.
<a href="#">enableFloatWindow</a>	Sets whether to enable floating windows.
<a href="#">setCallingBell</a>	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 <a href="#">Generating UserSig</a> .)



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

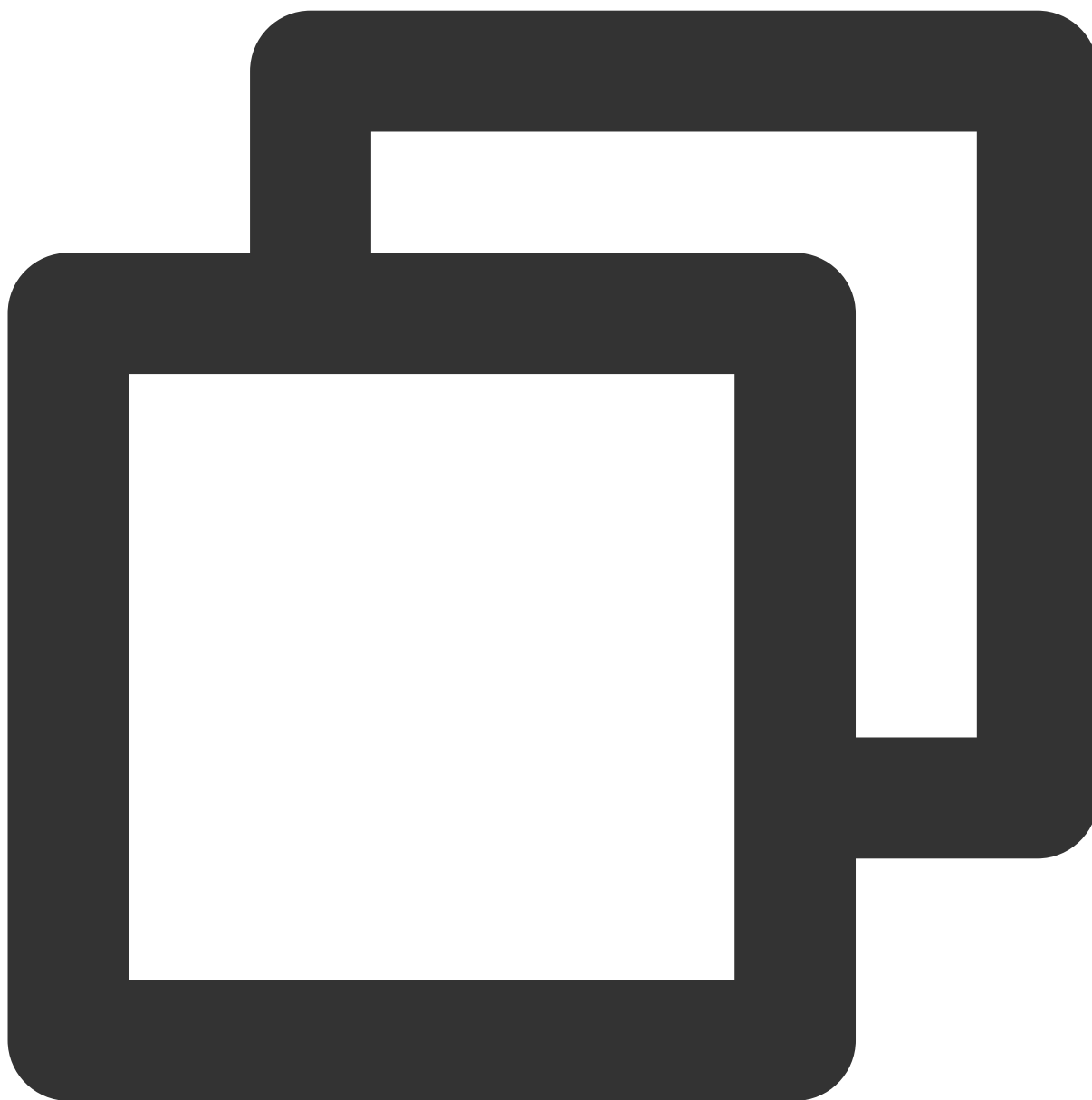
## logout



```
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.



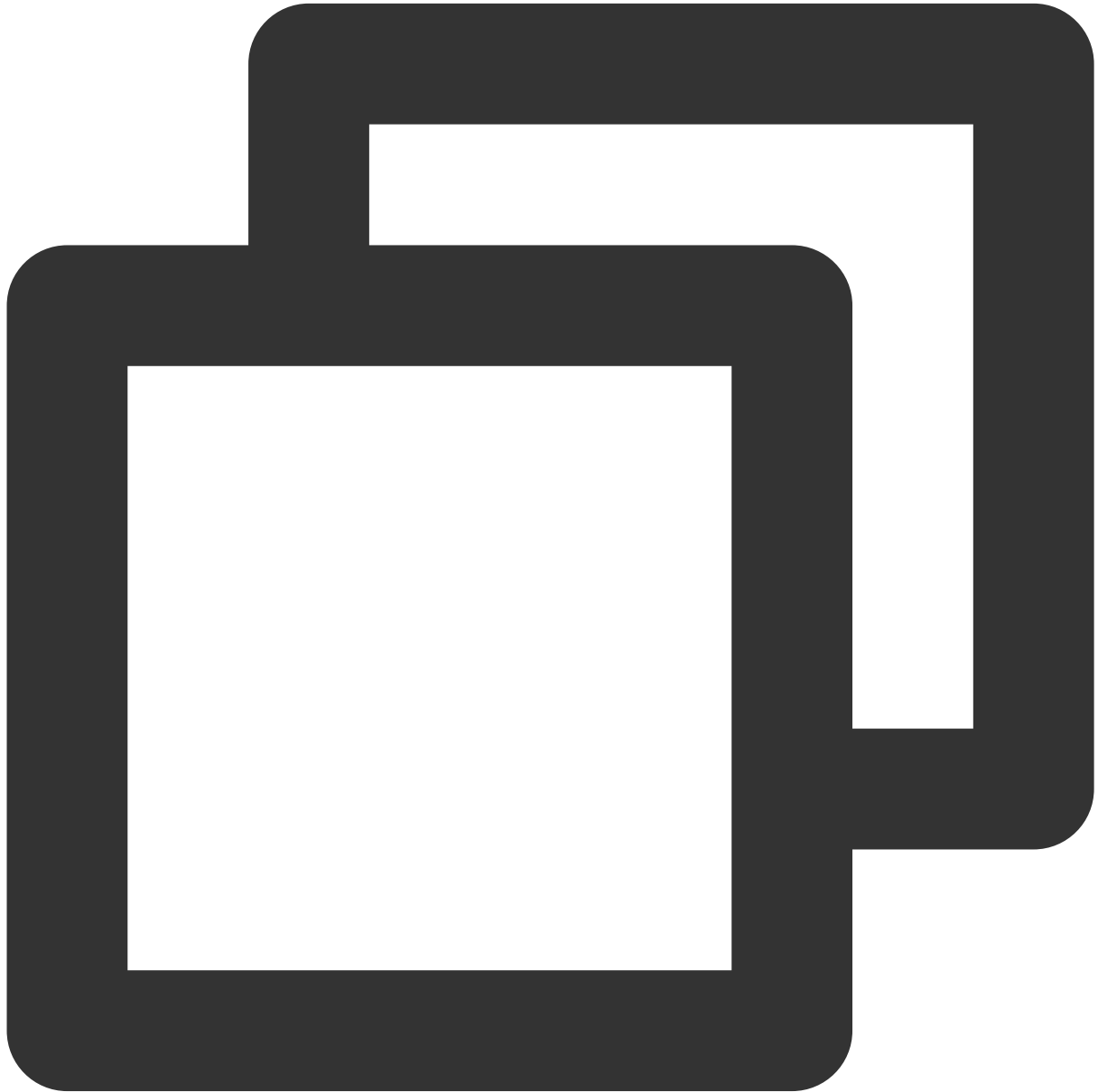
```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

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

return value	<a href="#">TUIResult</a>	Contains code and message information: If code is empty (""), the call is successful. 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:

--	--	--

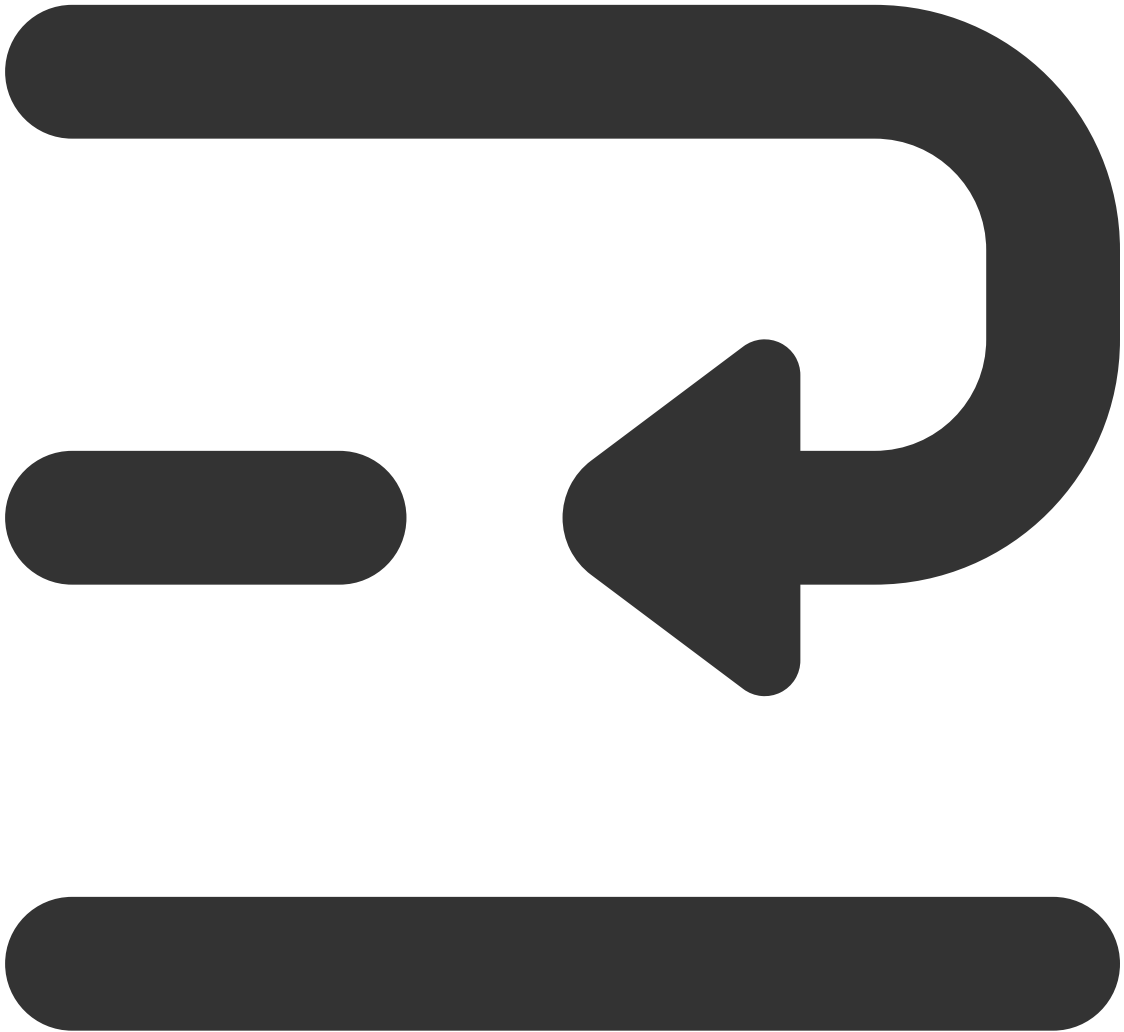
Parameter	Type	Description
userId	String	The target user ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	Call extension parameters, such as roomId, call timeout, offline push info,etc

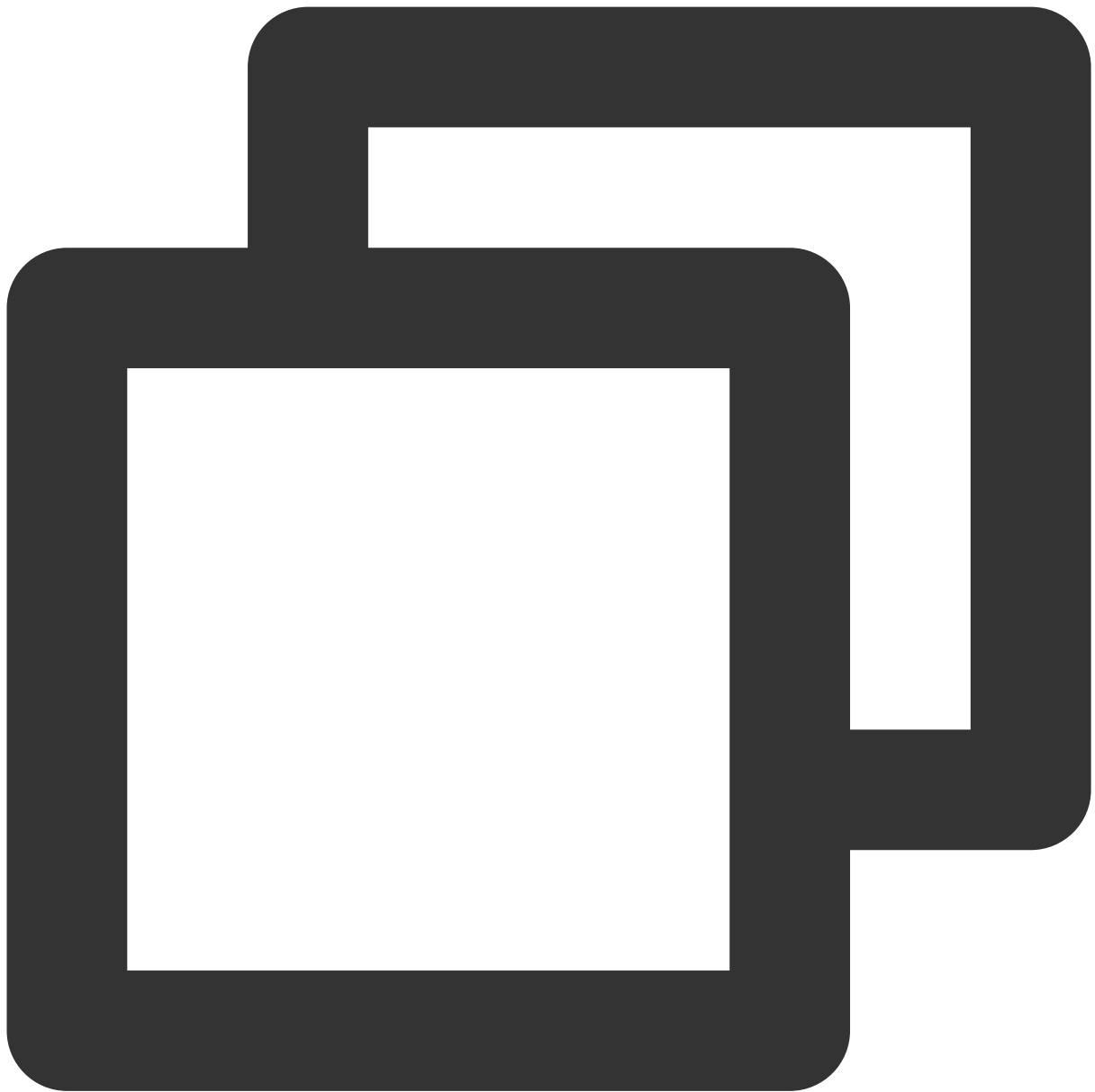
## groupCall

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	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	Call extension parameters, such as roomId, call timeout, offline push

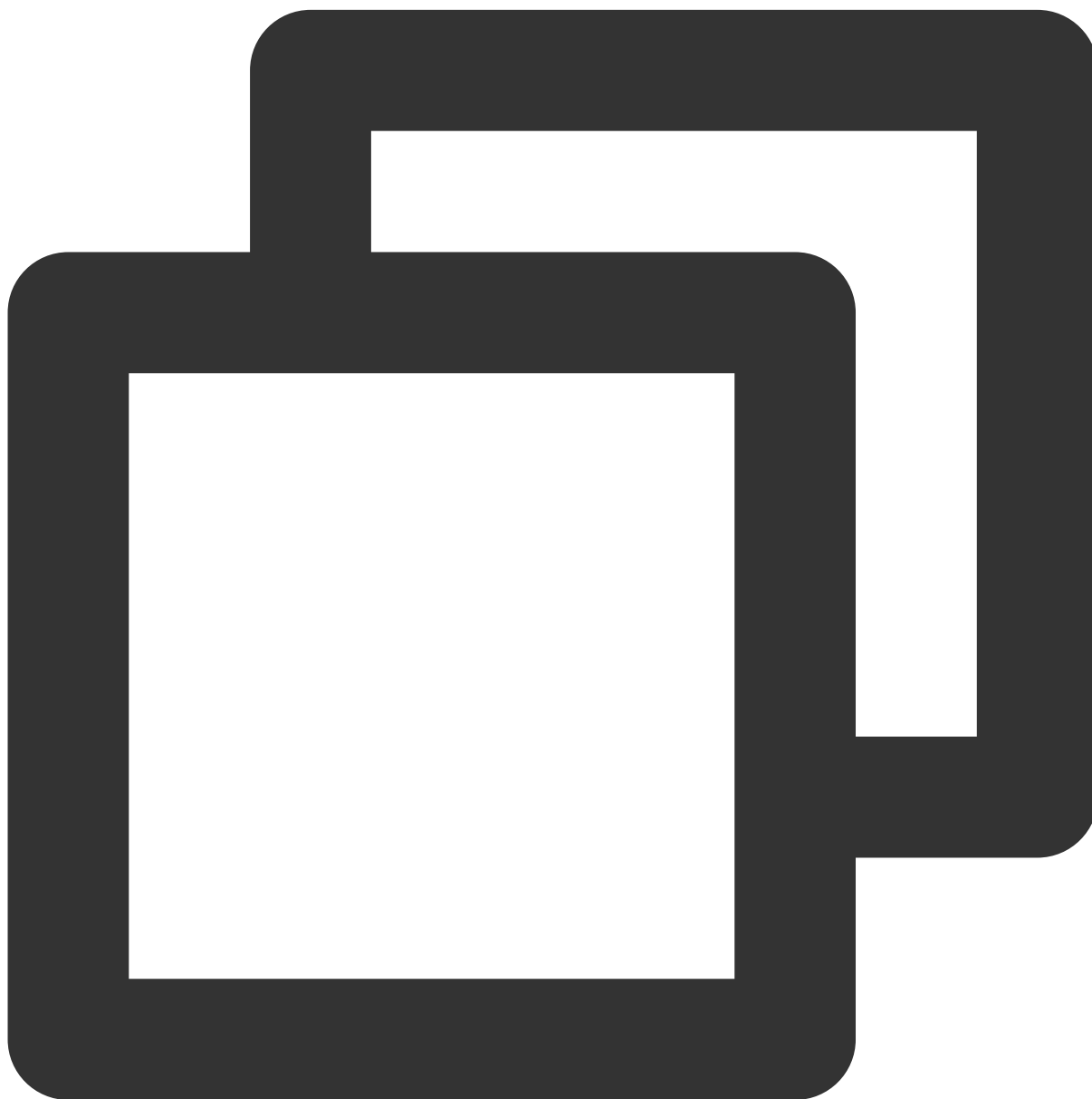
		info,etc
--	--	----------

## 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.

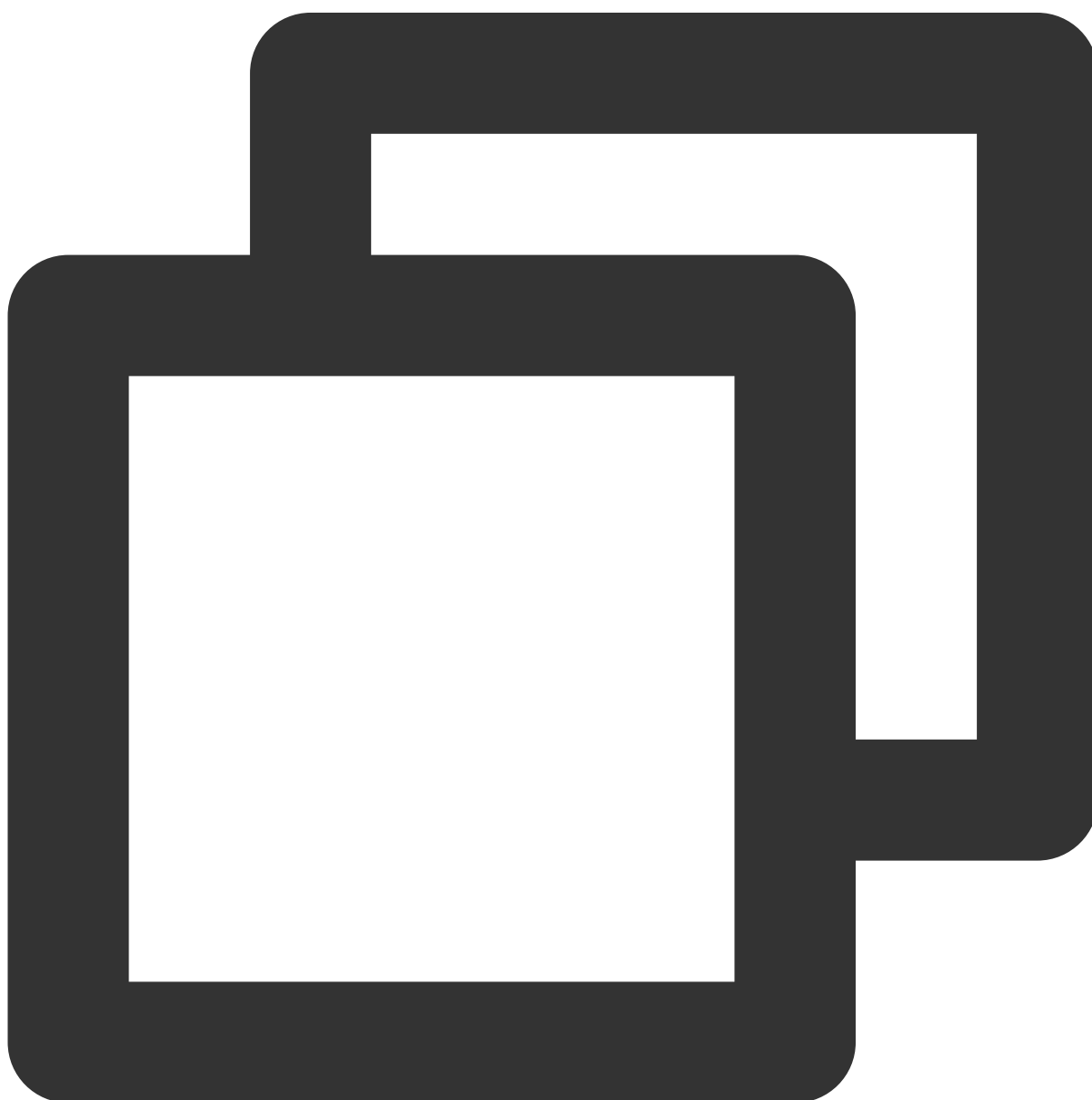


```
Future<void> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaType cal
```

Parameter	Type	Description
roomId	<a href="#">TUIRoomID</a>	The room ID.
groupId	String	The group ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

### **enableMuteMode**

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

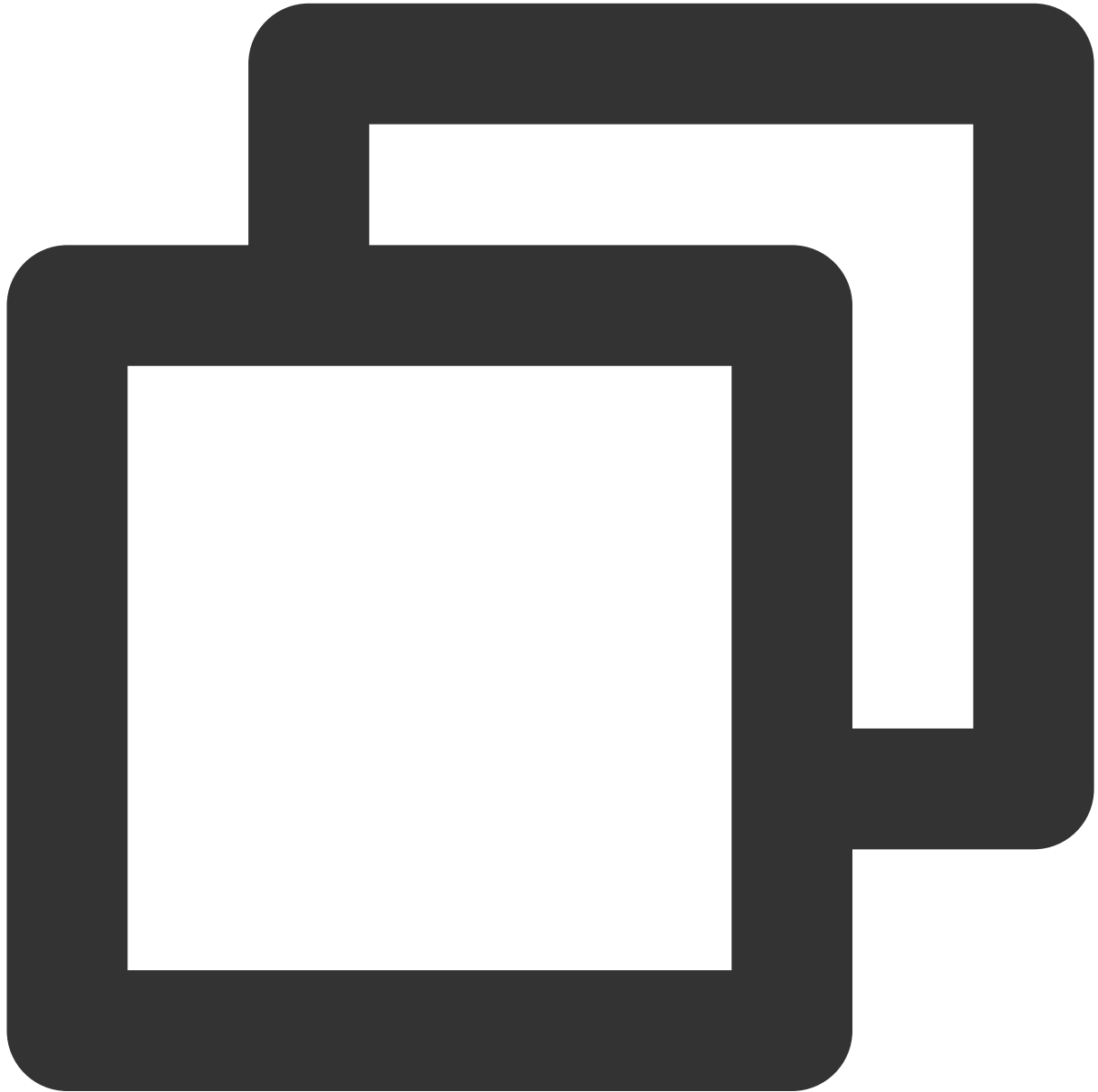




```
Future<void> enableMuteMode(bool enable)
```

## enableFloatWindow

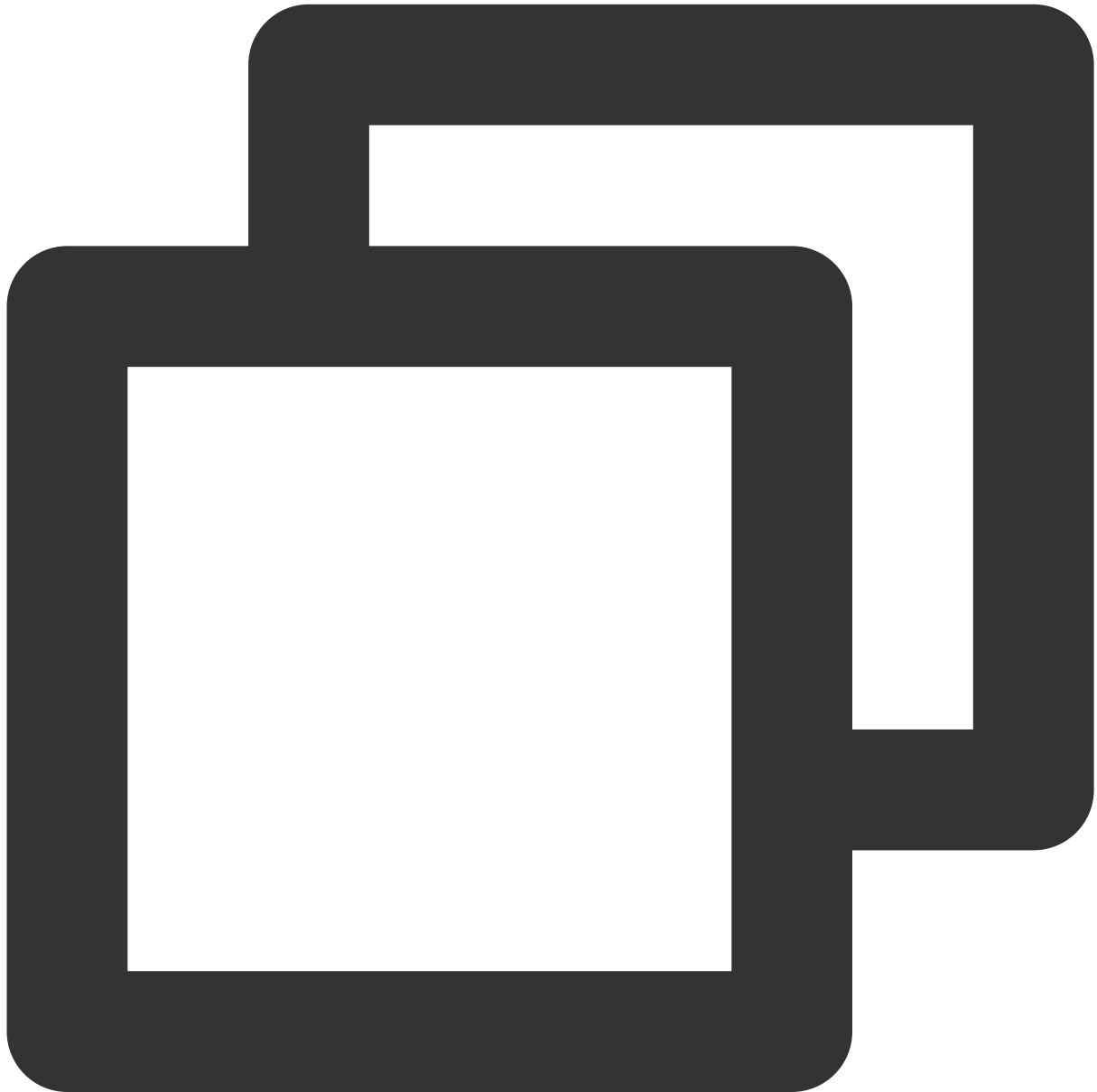
This API is used to set whether to enable floating windows. The default value is `false` , and the floating window button in the top left corner of the call view is hidden. If it is set to `true` , the button will become visible.



```
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

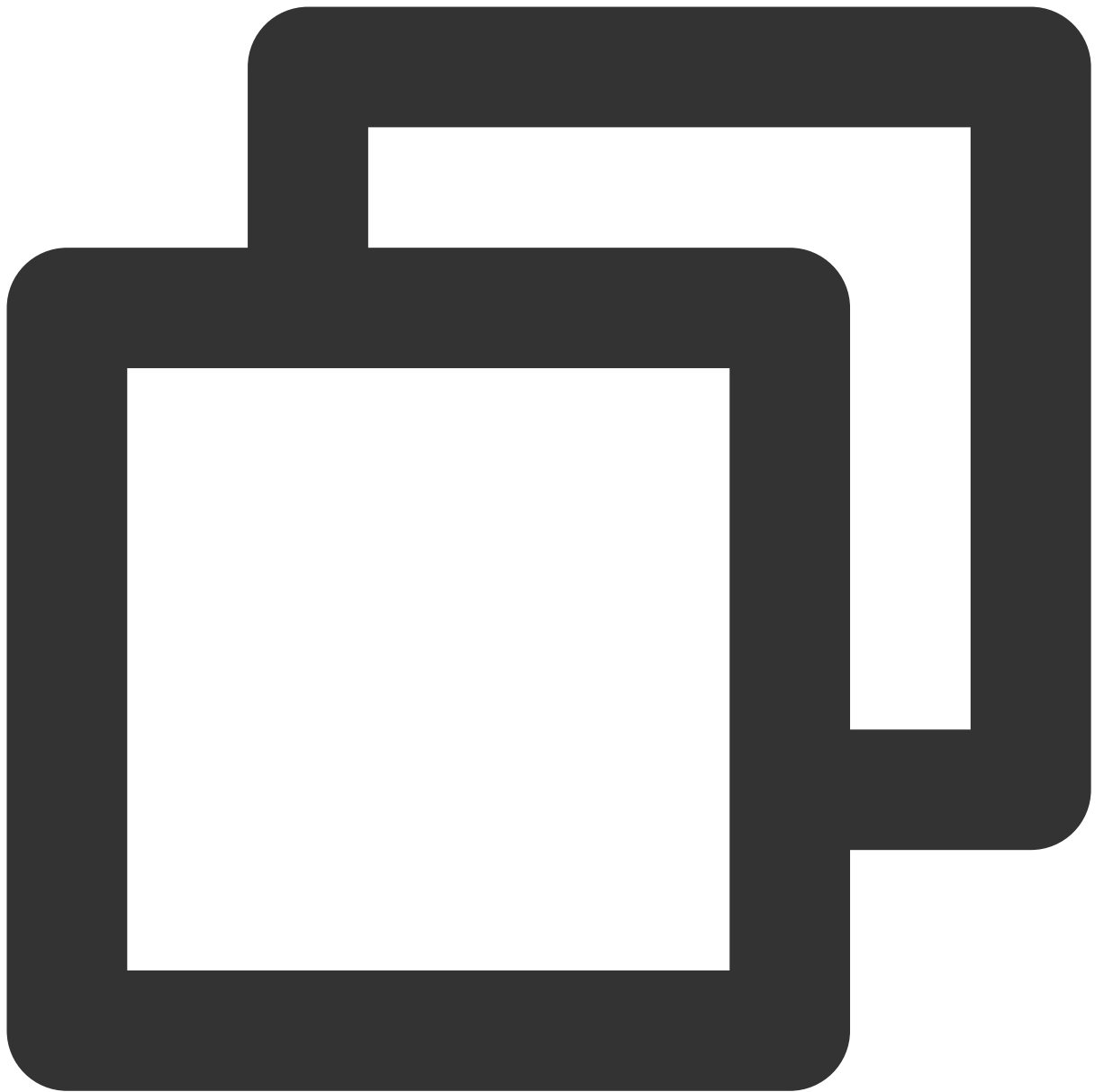
API	Description
<code>init</code>	Authenticates the basic audio/video call capabilities.
<code>unInit</code>	The destructor function, which releases resources used by TUICallEngine.
<code>addObserver</code>	Registers an event listener.
<code>removeObserver</code>	Unregisters an event listener.
<code>call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

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

## Details

### init

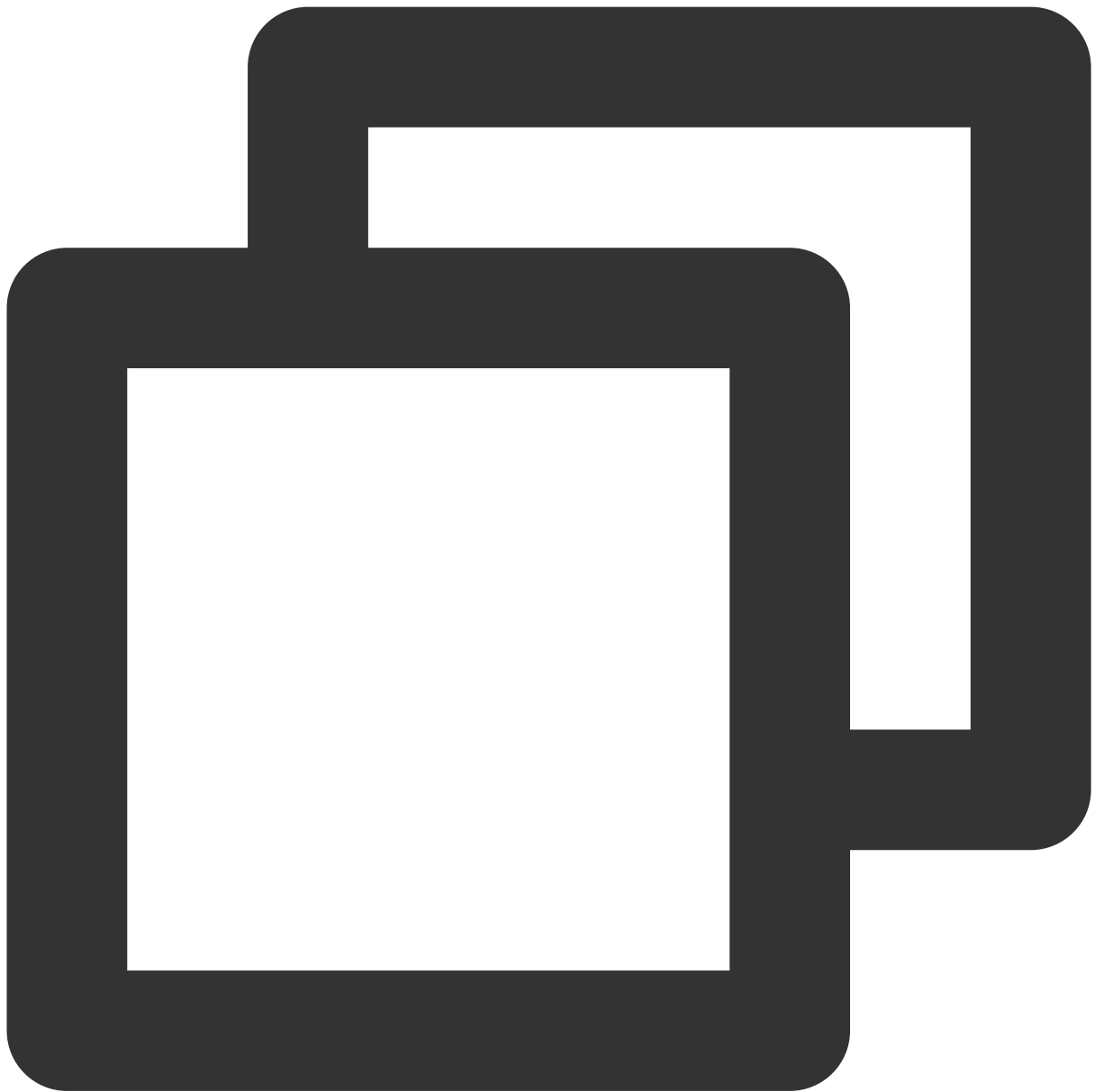
This API is used to initialize `TUICallEngine` . Call it to authenticate the call service and perform other required actions before you call other APIs.



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

## **unInit**

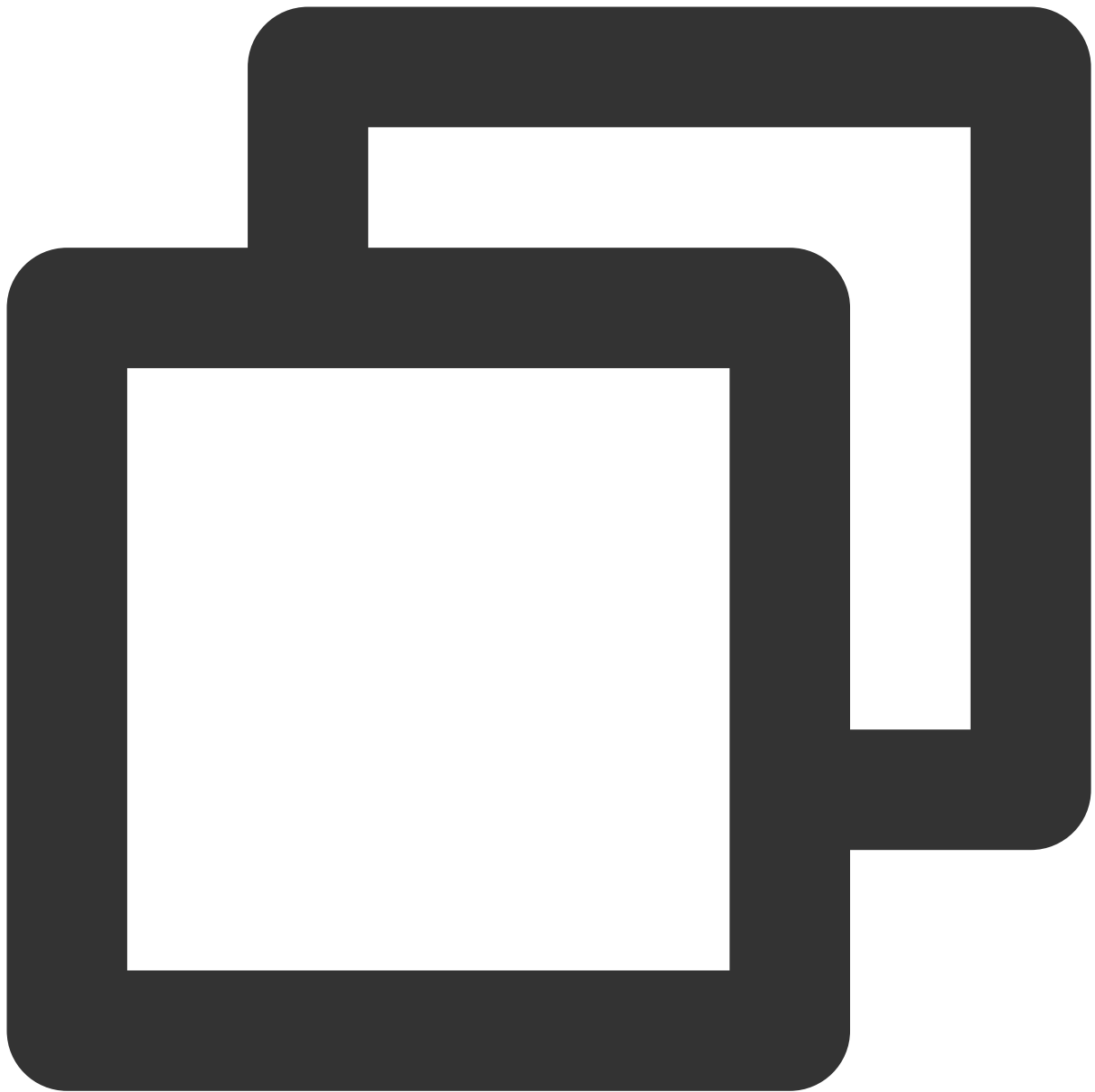
The destructor function, which releases resources used by TUICallEngine.



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

### **addObserver**

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

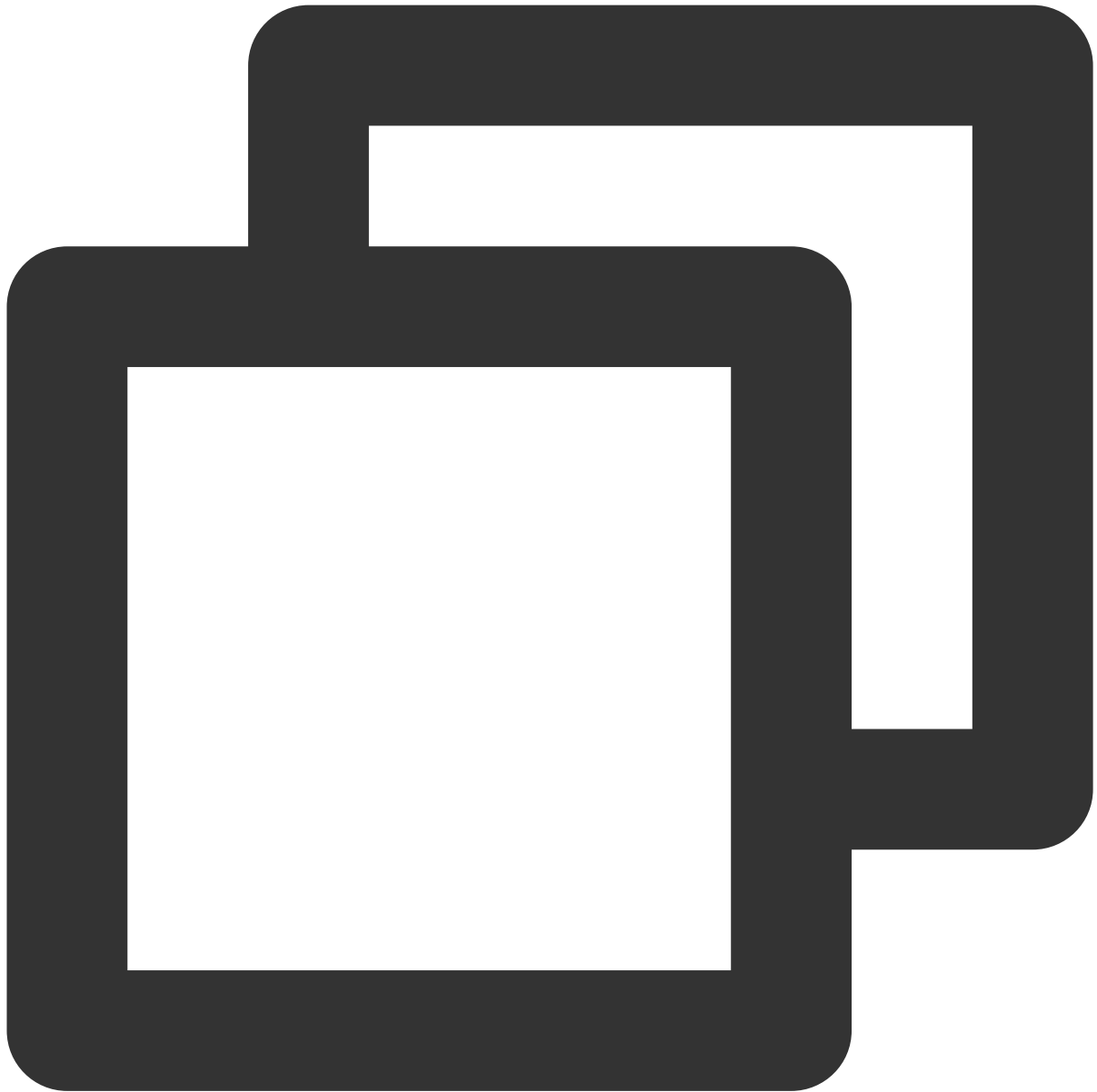


```
Future<void> addObserver(TUICallObserver observer)
```

### **removeObserver**

This API is used to unregister an event listener.

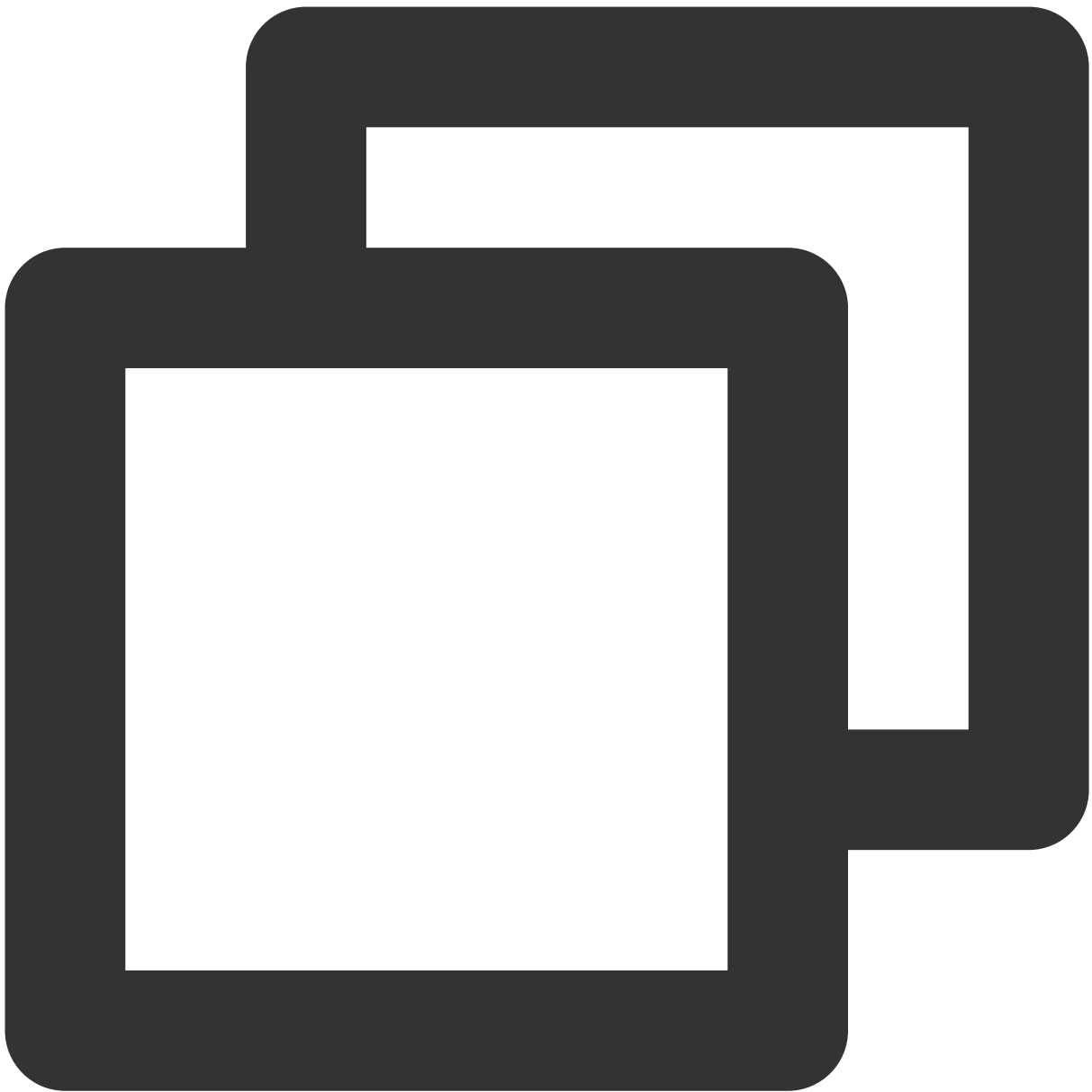




```
Future<void> removeObserver(TUICallObserver observer)
```

### call

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



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

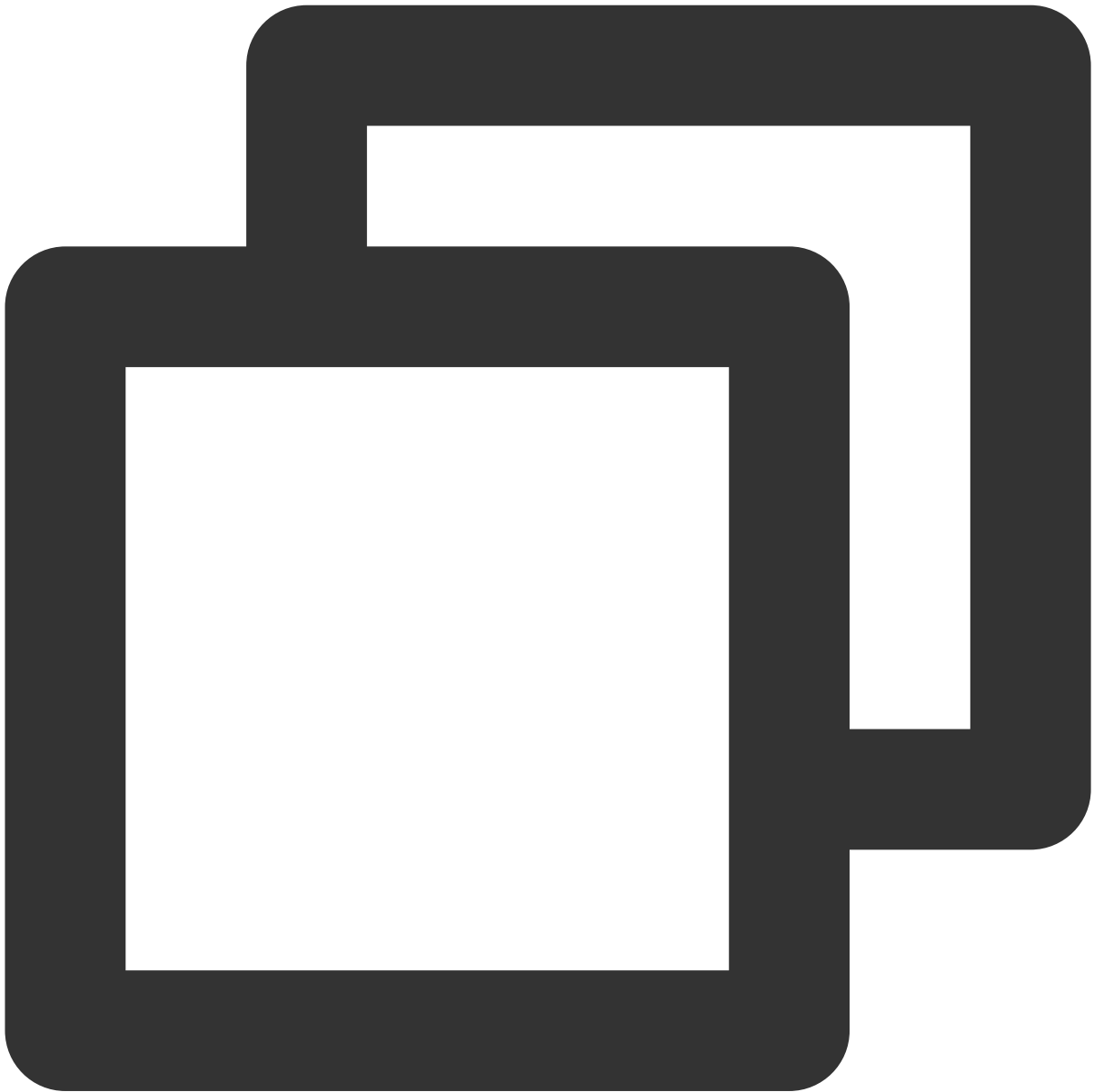
Parameter	Type	Description
userId	String	The target user ID.
mediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	An additional parameter, such as roomId, call timeout, offline push info, etc

groupCall

This API is used to make a group call.

Notice :

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



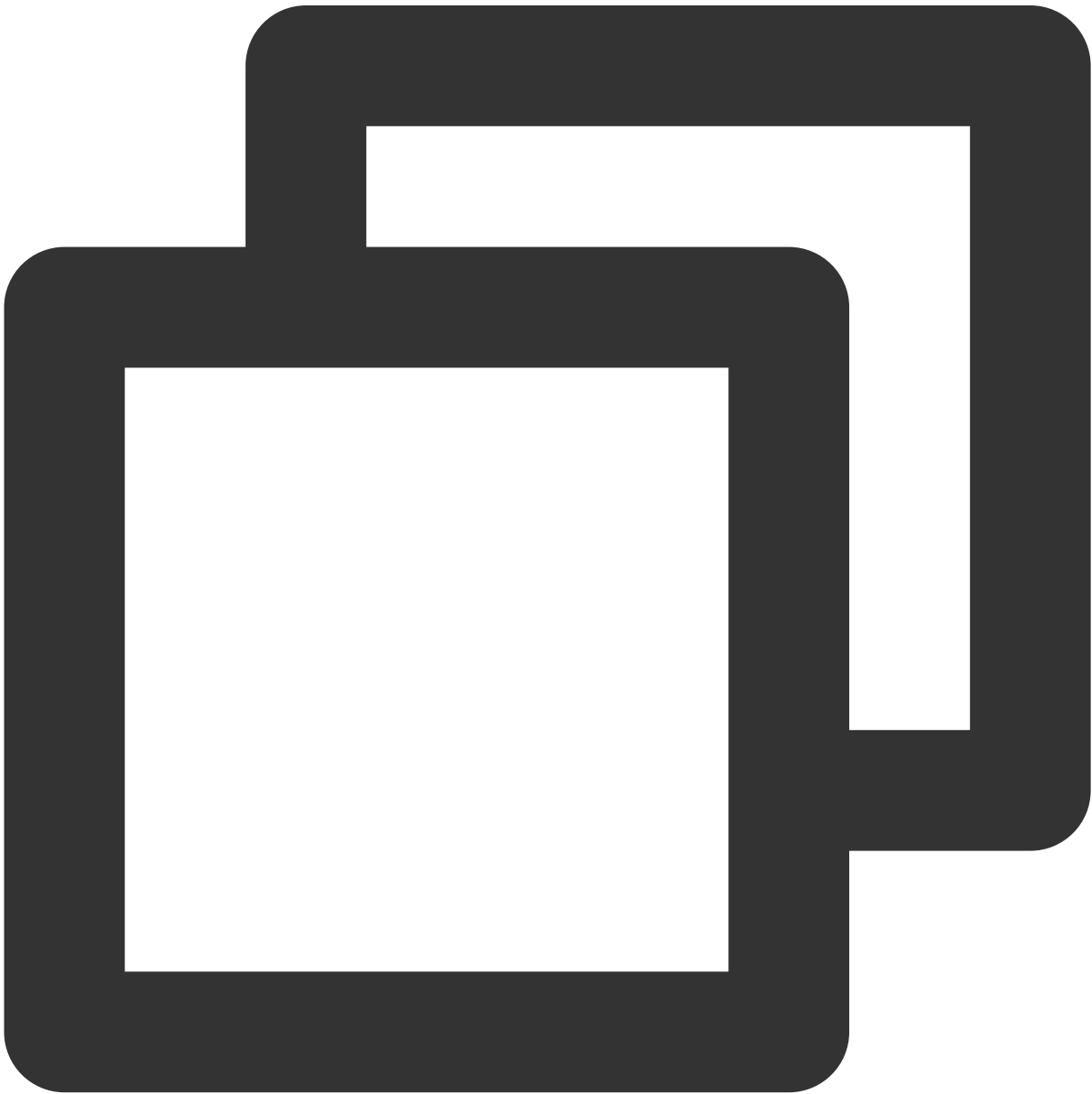
```
Future<TUIResult> groupCall(String groupId, List<String> userIdList, TUICallMediaTy
```

Parameter	Type	Description
groupId	String	The group ID.

userIdList	List<String>	The target user IDs.
mediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
params	<a href="#">TUICallParams</a>	An additional parameter. such as roomId, call timeout, offline push info, etc

accept

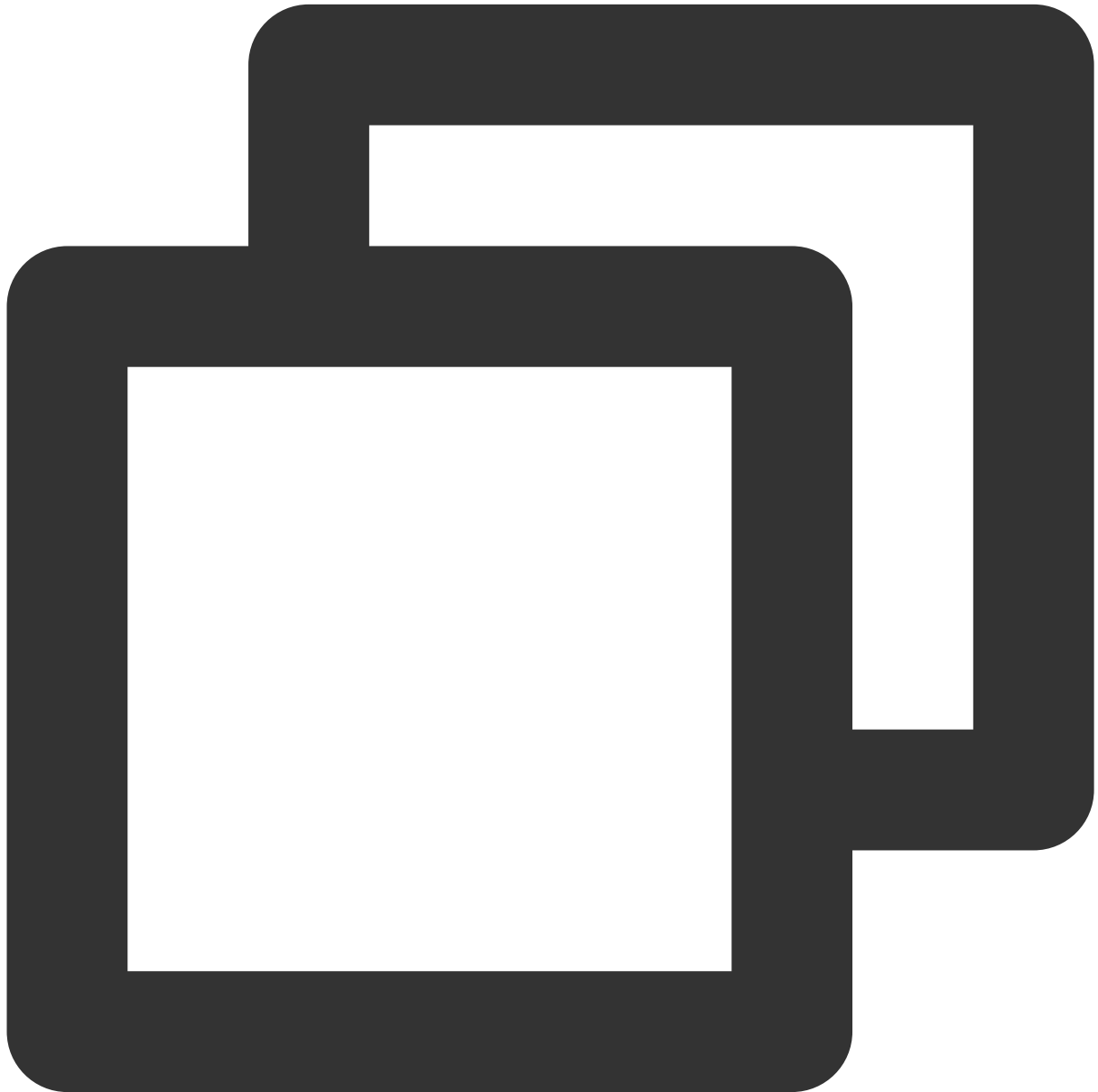
This API is used to accept a call. After receiving the `onCallReceived()` callback, you can call this API to accept the call.



```
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.

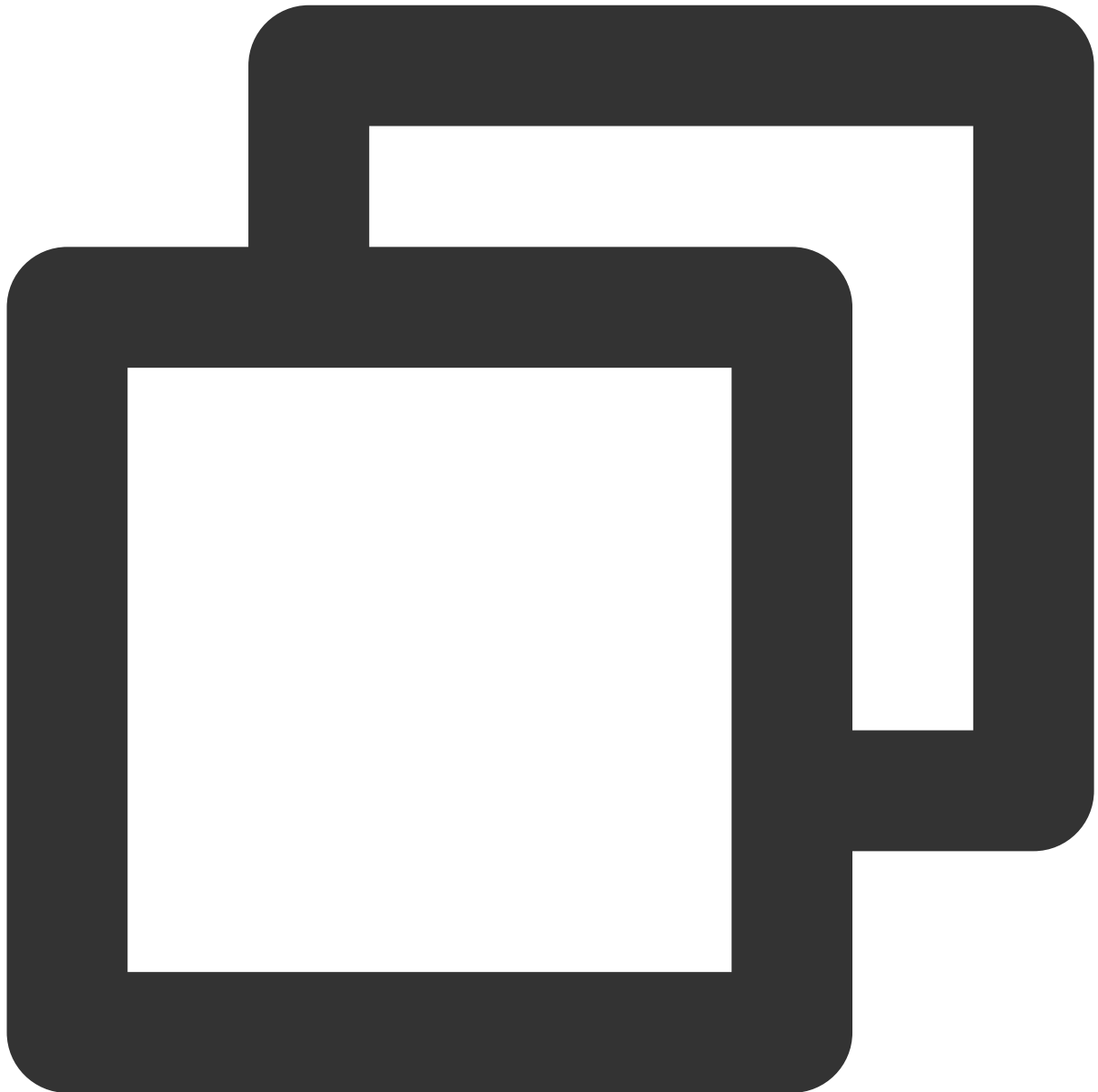


```
Future<TUIResult> reject ()
```

## 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.

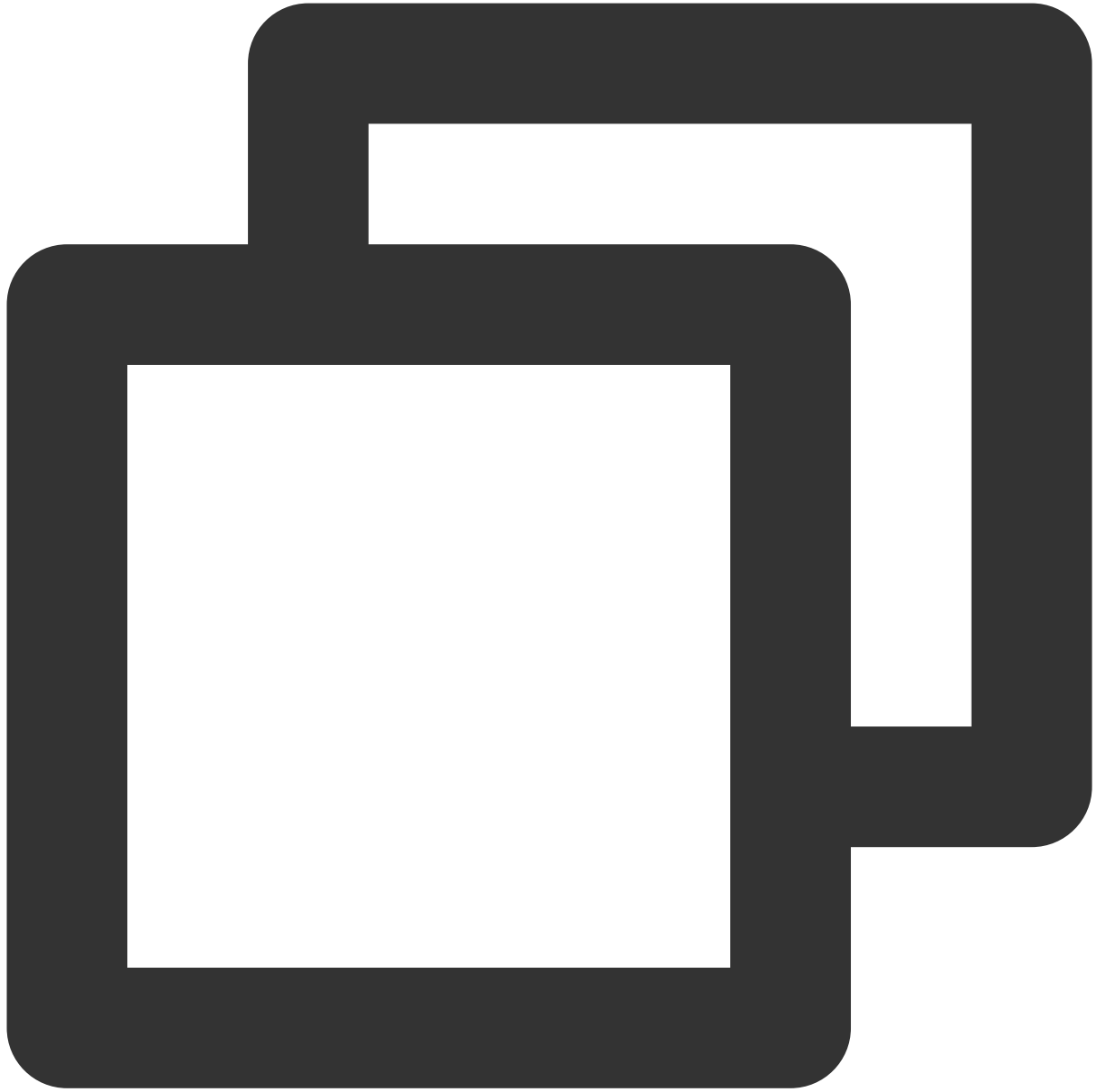
Note: If your project involves live streaming or conferencing, you can also use this API to implement the “in a meeting” or “on air” feature.



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

## hangup

This API is used to end a call.

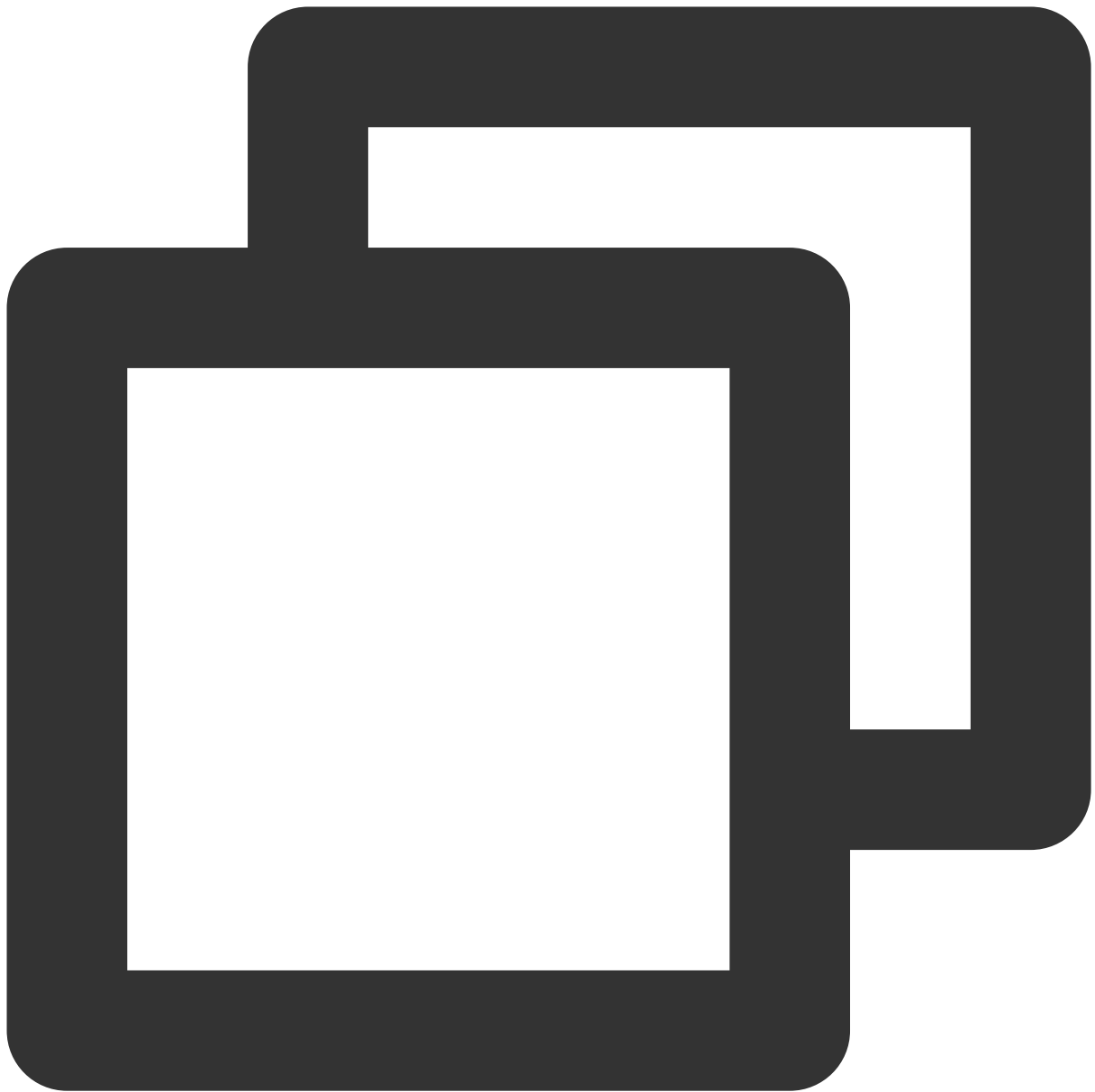


```
Future<TUIResult> hangup()
```

## 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> ininviteUser(List<String> userIdList, TUICallParams params, TUIValueCal
```

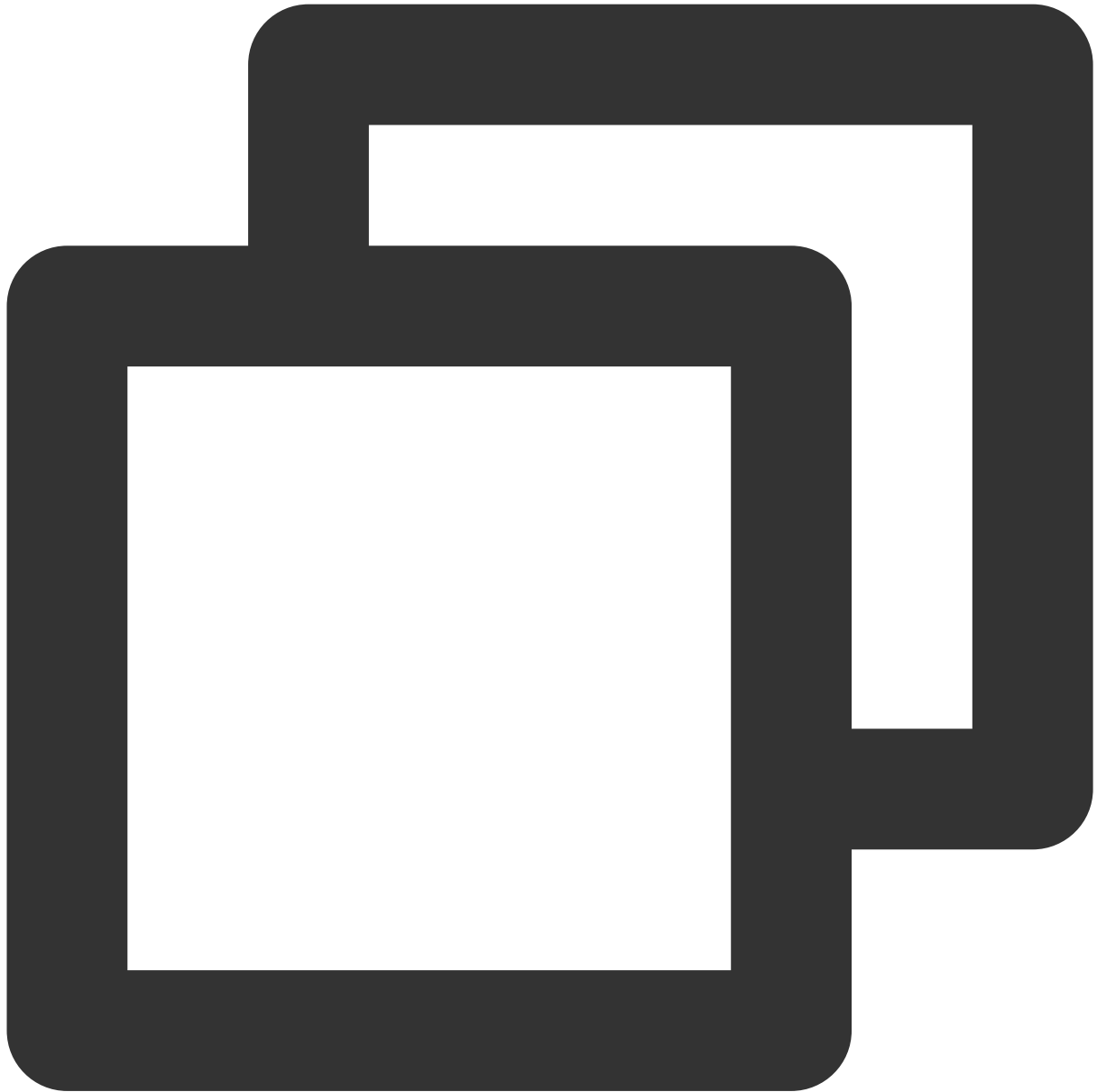
Parameter	Type	Description
userIdList	List<String>	The target user IDs.
params	<a href="#">TUICallParams</a>	An additional parameter. such as roomId, call timeout, offline push info, etc.

## joinInGroupCall



This API is used to join a group call.

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



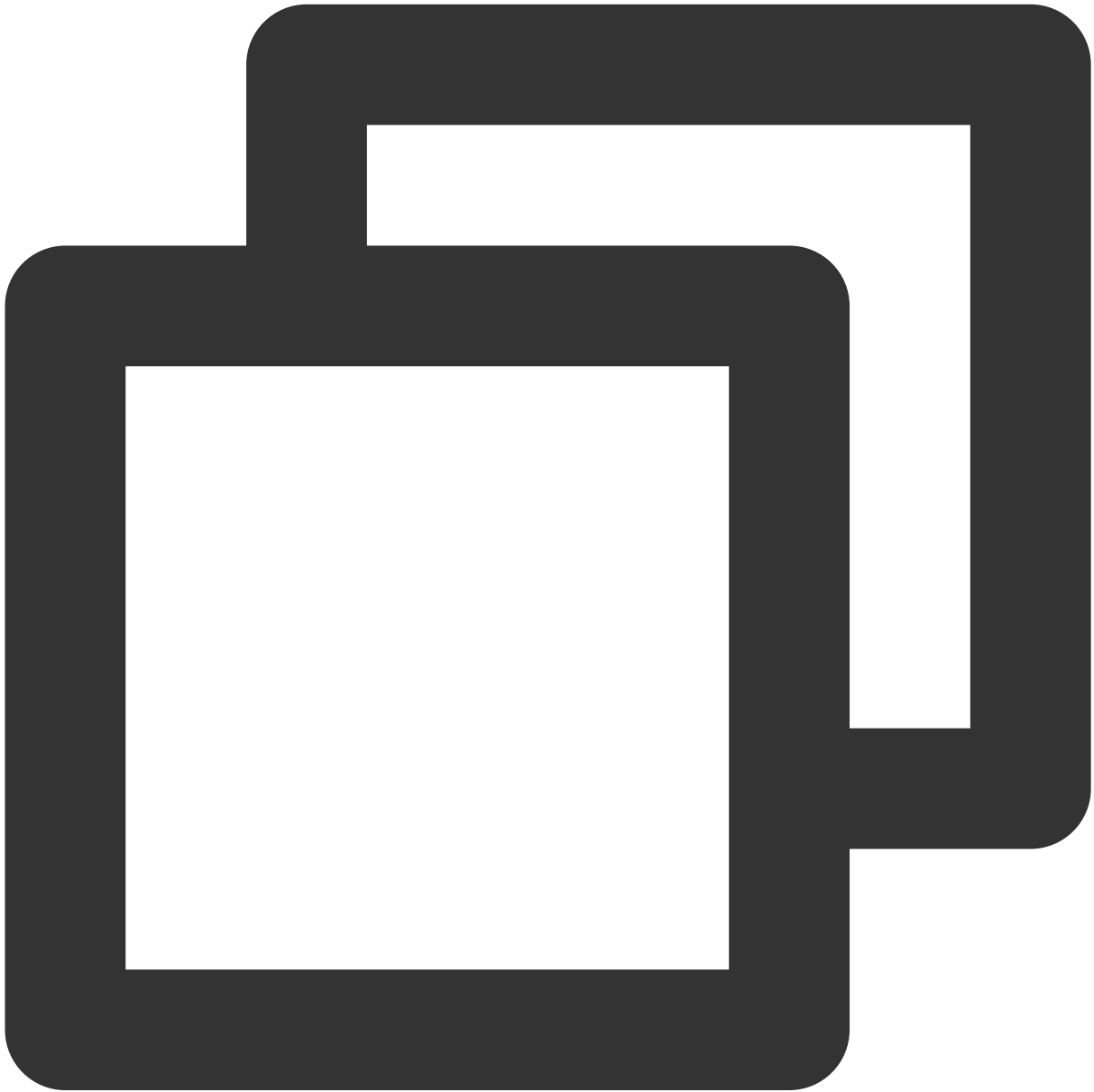
```
Future<TUIResult> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaTyp
```

Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.
groupId	String	The group ID.

mediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
-----------	----------------------------------	---

switchCallMediaType

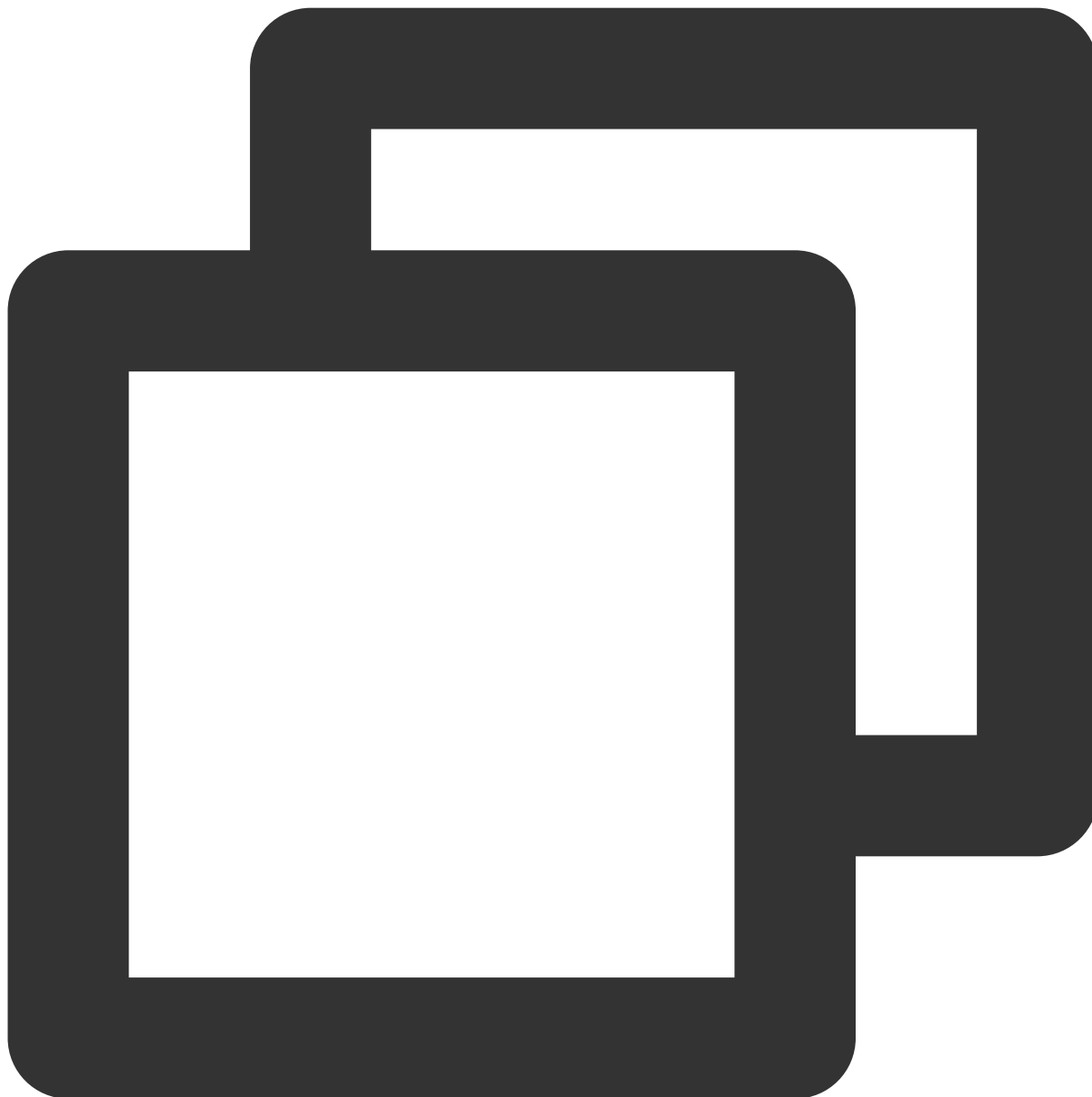
This API is used to change the call type.



Future<void> switchCallMediaType(TUICallMediaType mediaType)		
Parameter	Type	Description
mediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.

## startRemoteView

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

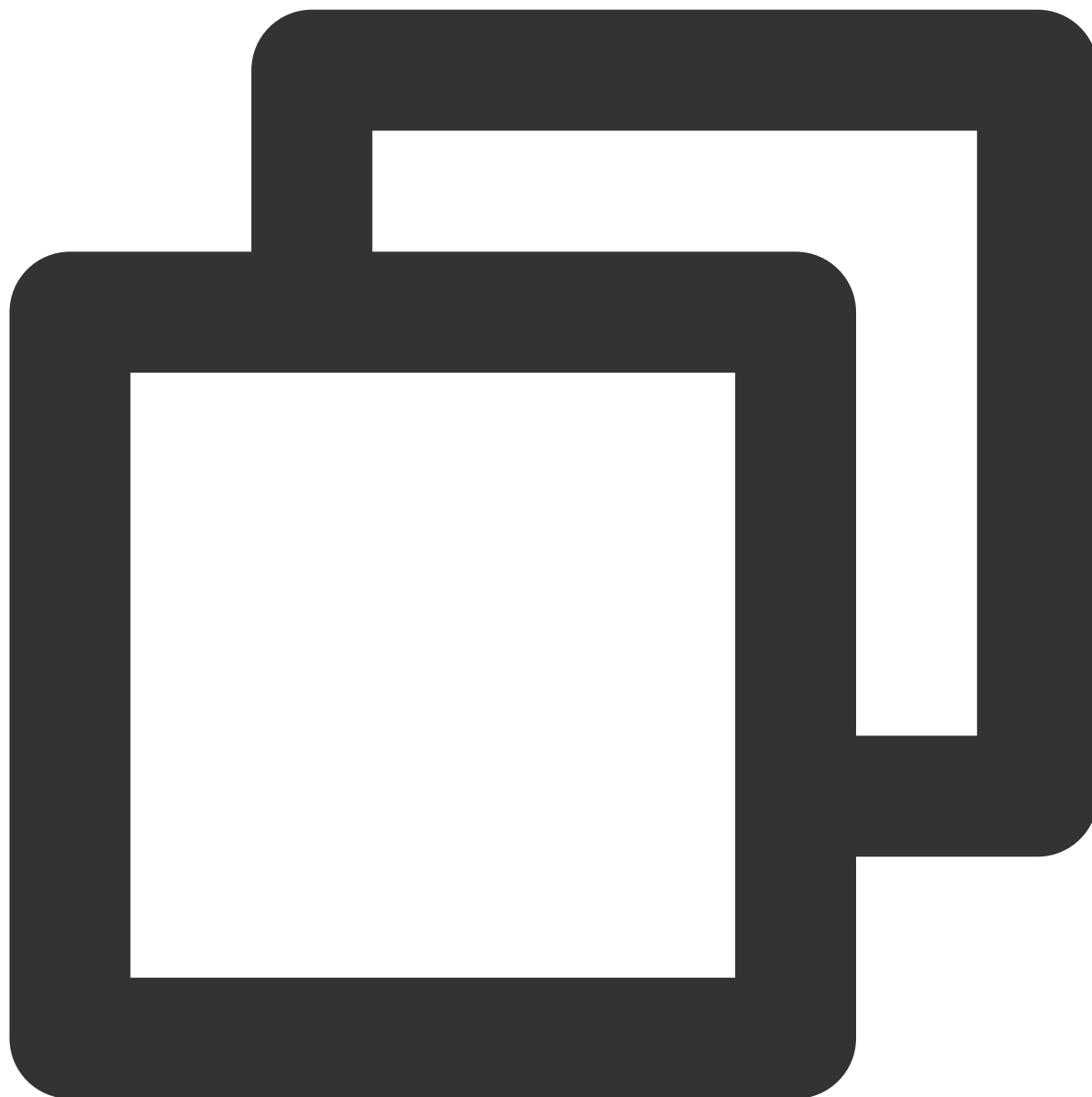


```
Future<void> startRemoteView(String userId, intviewId)
```

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.

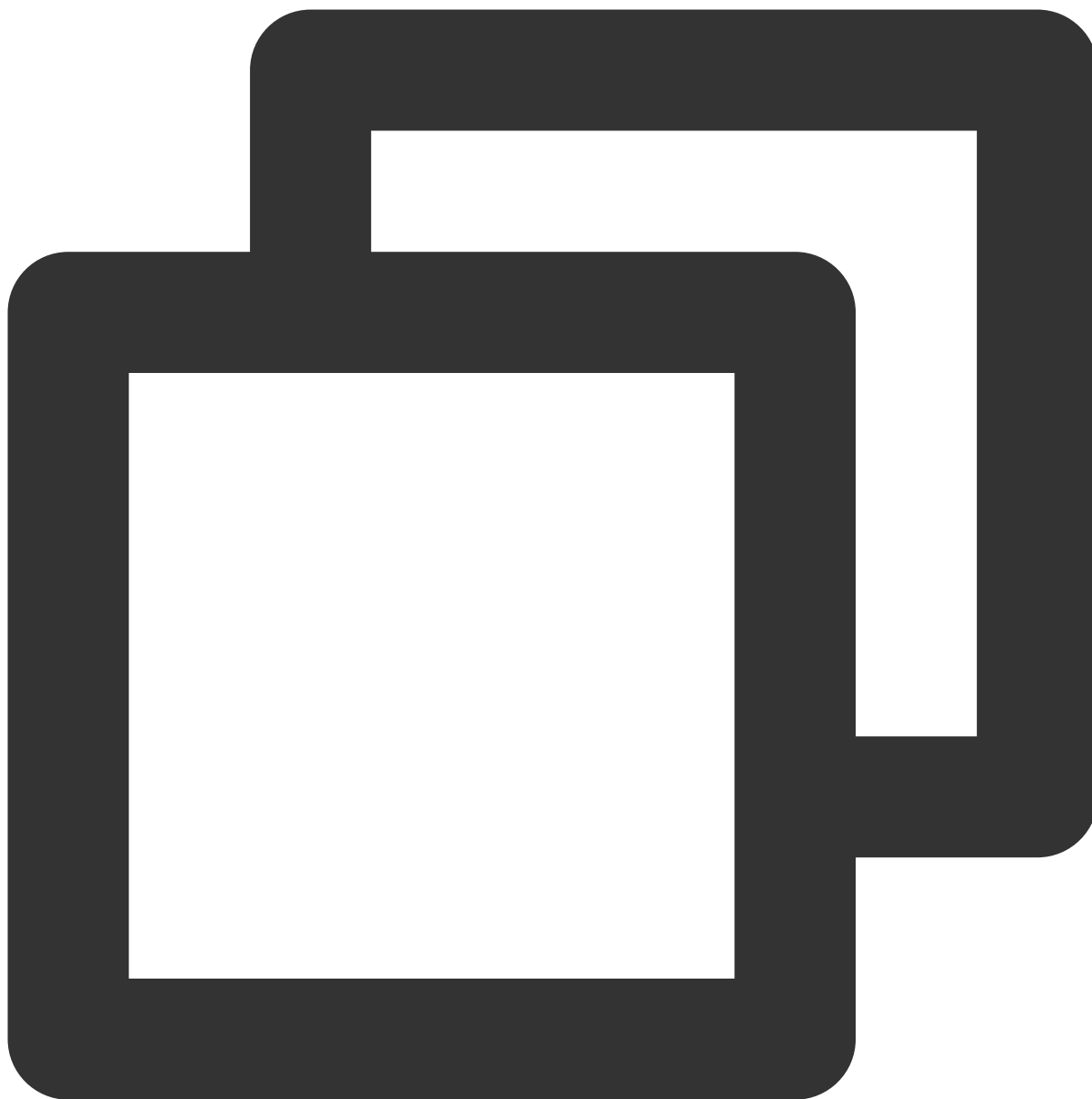


```
Future<void> stopRemoteView(String userId)
```

Parameter	Type	Description
userId	String	The target user ID.

## openCamera

This API is used to turn the camera on.

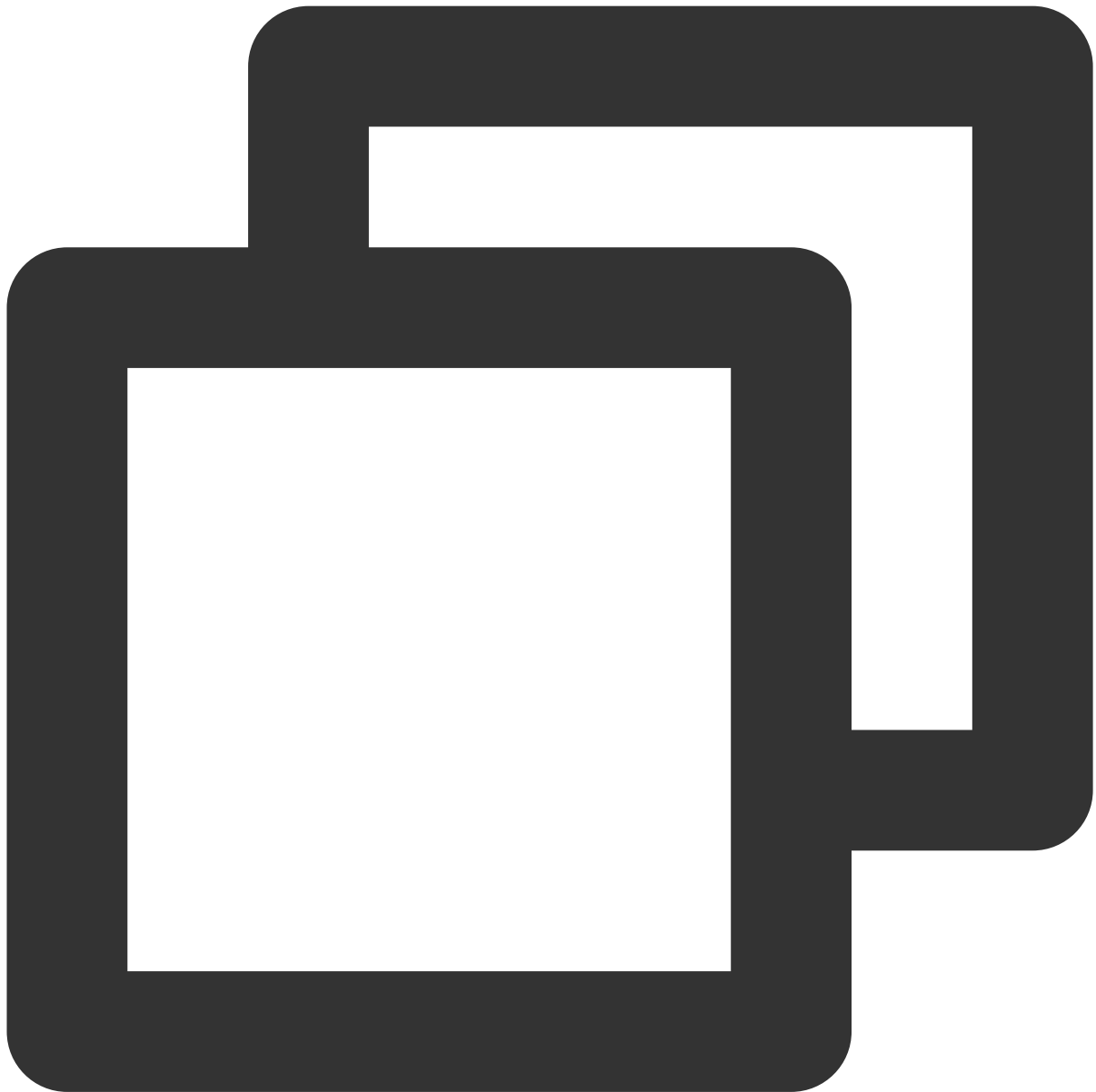


```
Future<TUIResult> openCamera(TUICamera camera, int? viewId)
```

Parameter	Type	Description
camera	<a href="#">TUICamera</a>	The front or rear camera.
viewId	int	The ID of the widget in the video rendering screen

## closeCamera

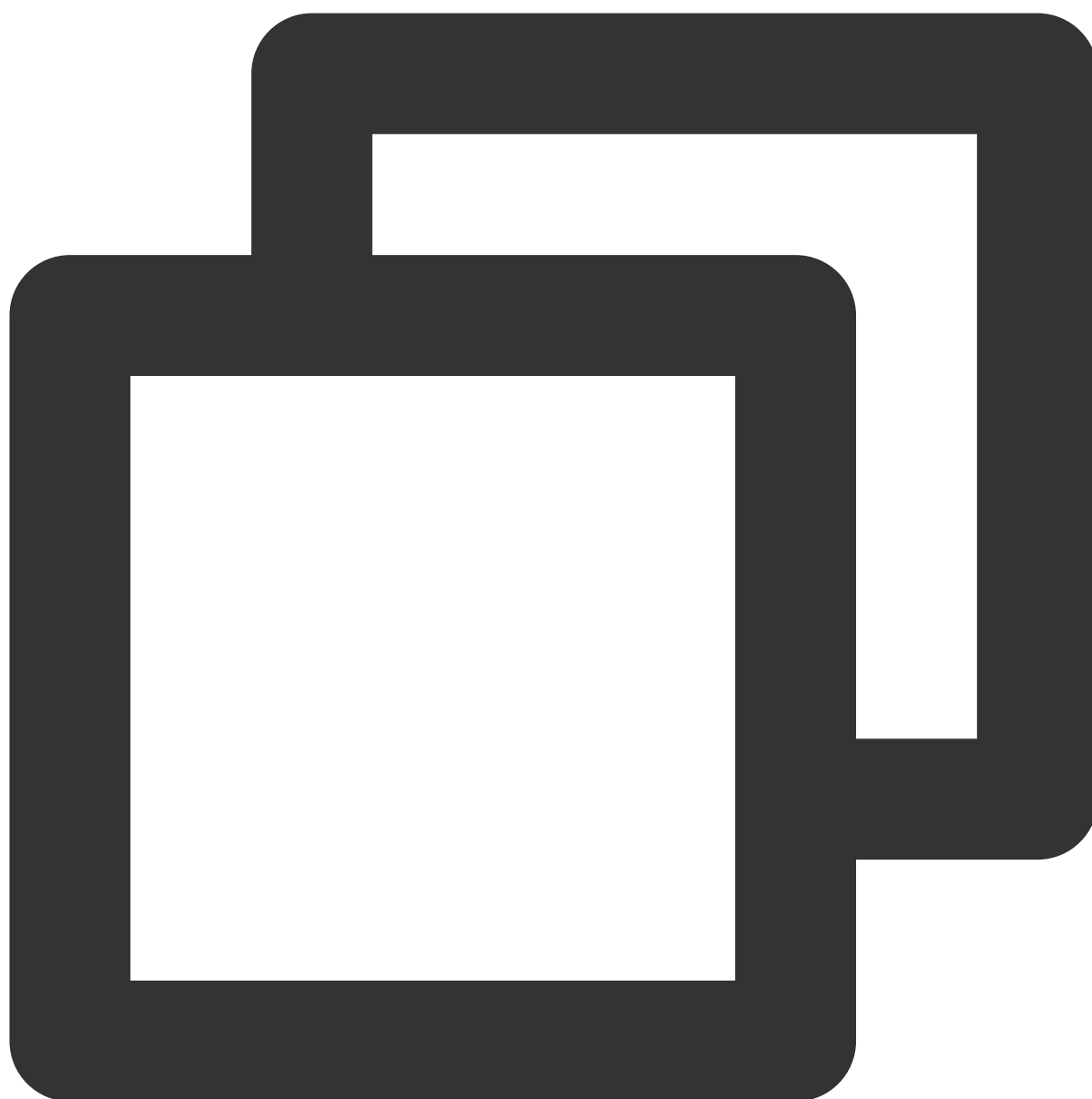
This API is used to turn the camera off.



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

## switchCamera

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

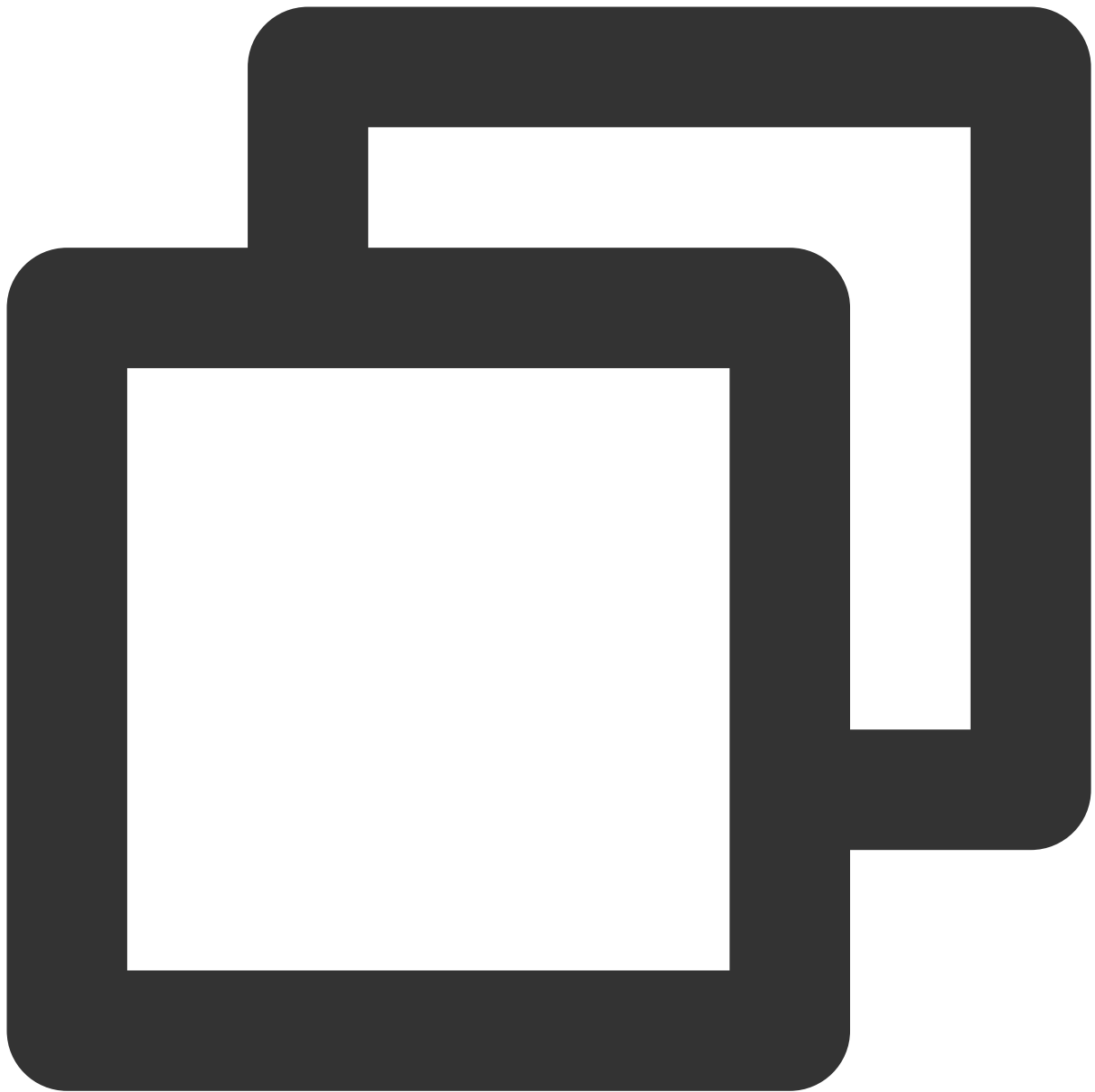


```
Future<void> switchCamera(TUICamera camera)
```

Parameter	Type	Description
camera	<a href="#">TUICamera</a>	The front or rear camera.

## openMicrophone

This API is used to turn the mic on.

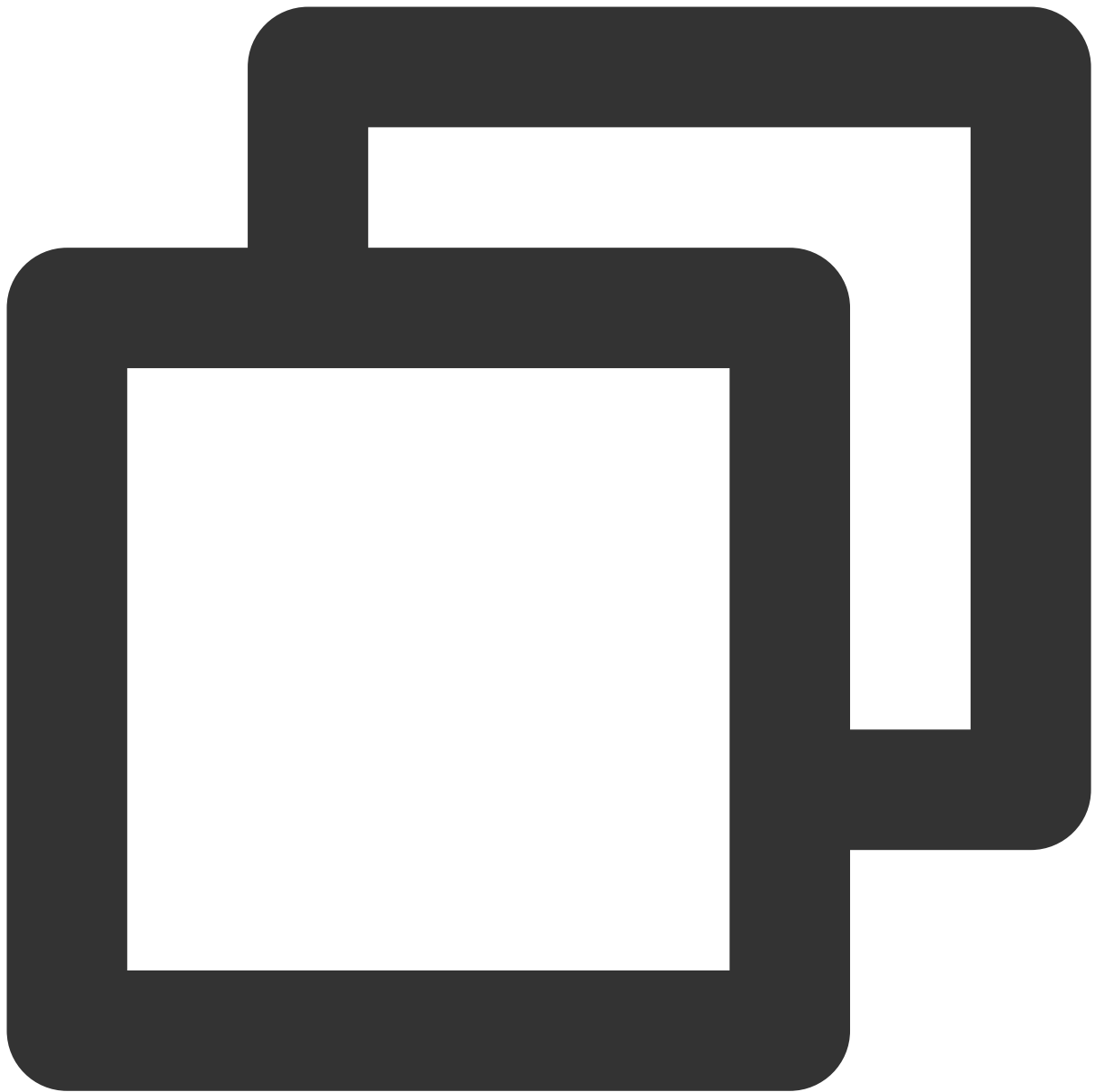


```
Future<TUIResult> openMicrophone()
```

### **closeMicrophone**

This API is used to turn the mic off.

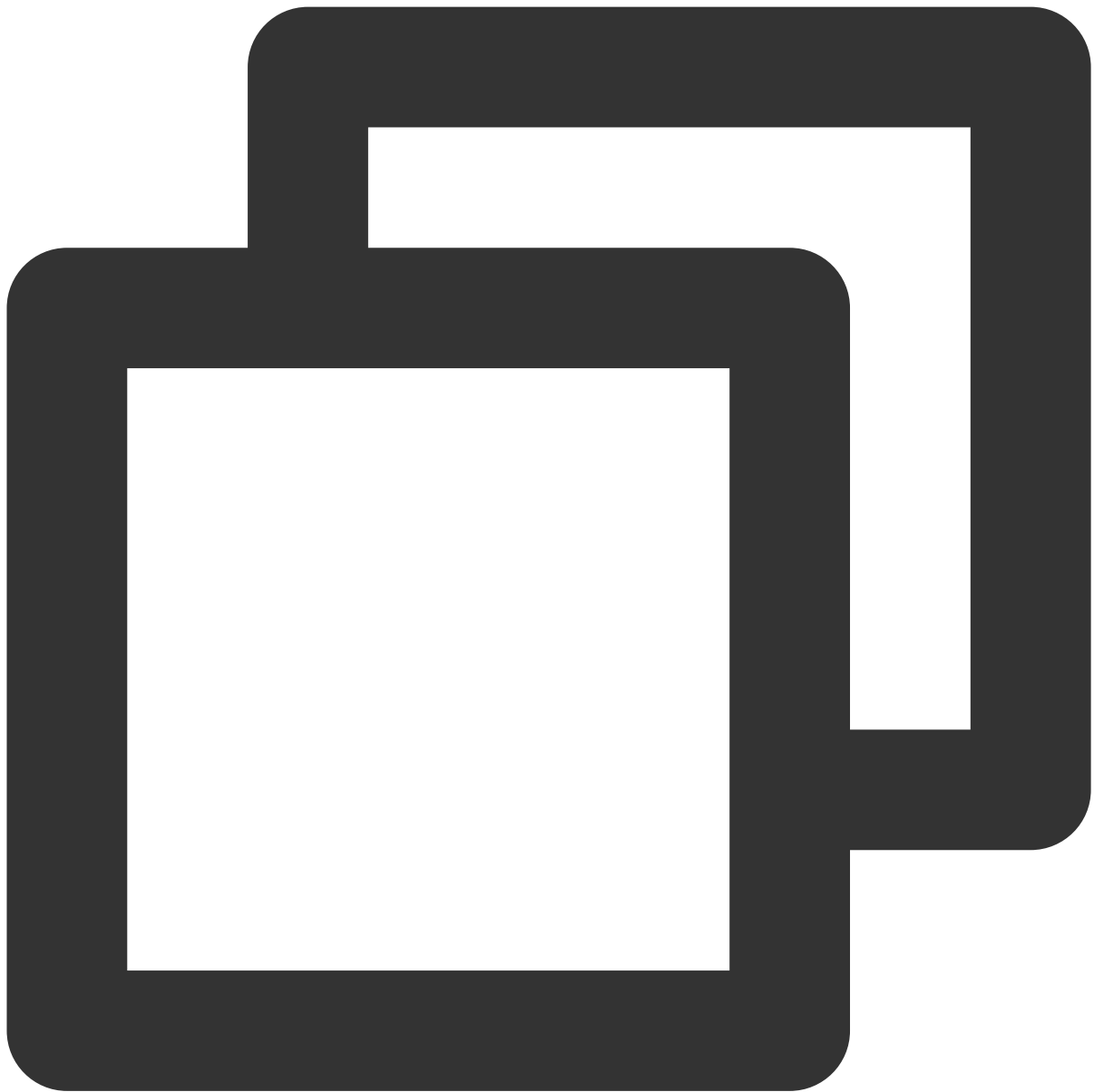




```
Future<void> closeMicrophone()
```

### **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.

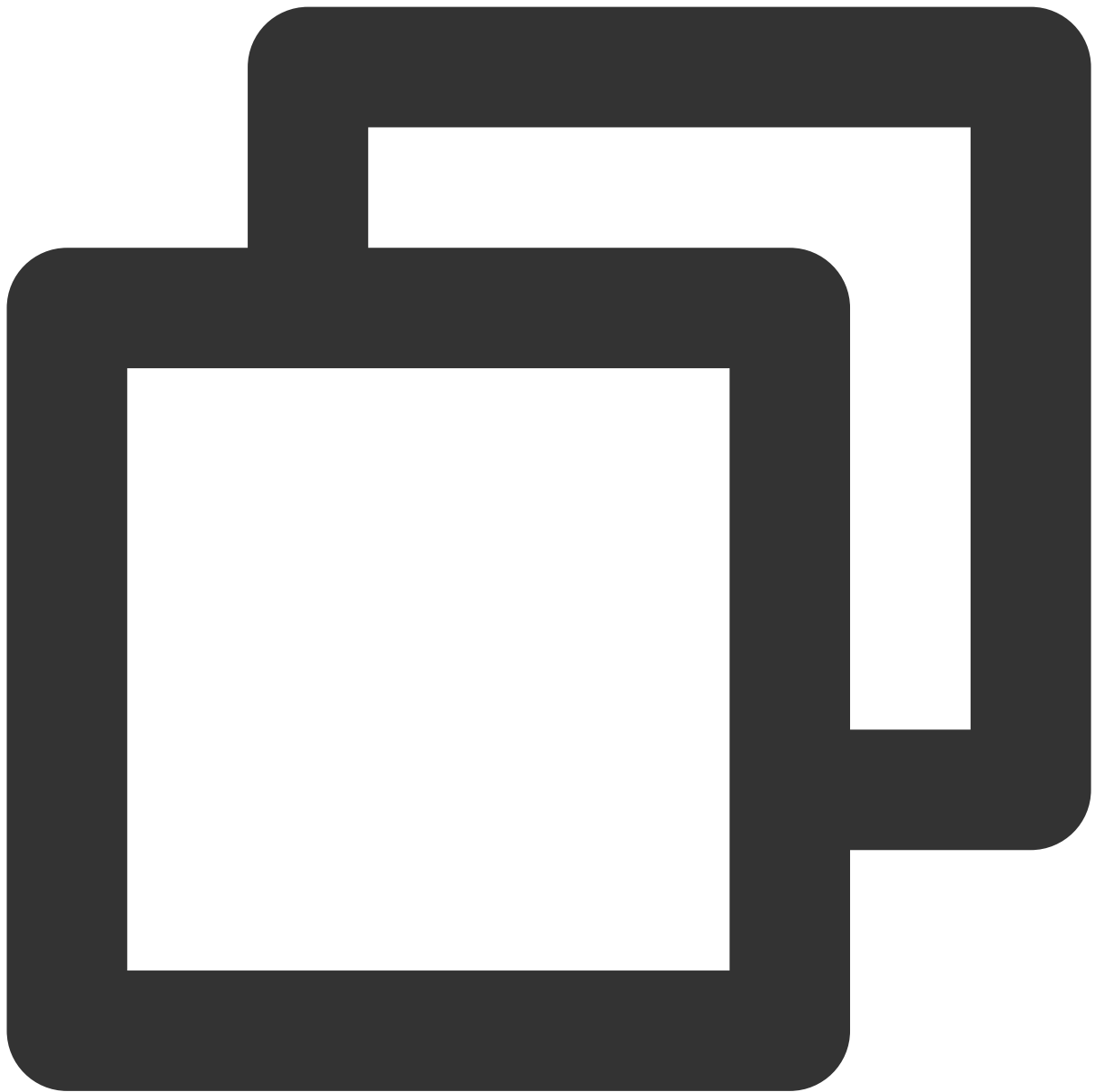


```
Future<void> selectAudioPlaybackDevice(TUIAudioPlaybackDevice device)
```

Parameter	Type	Description
device	<a href="#">TUIAudioPlaybackDevice</a>	The speaker or receiver.

## setSelfInfo

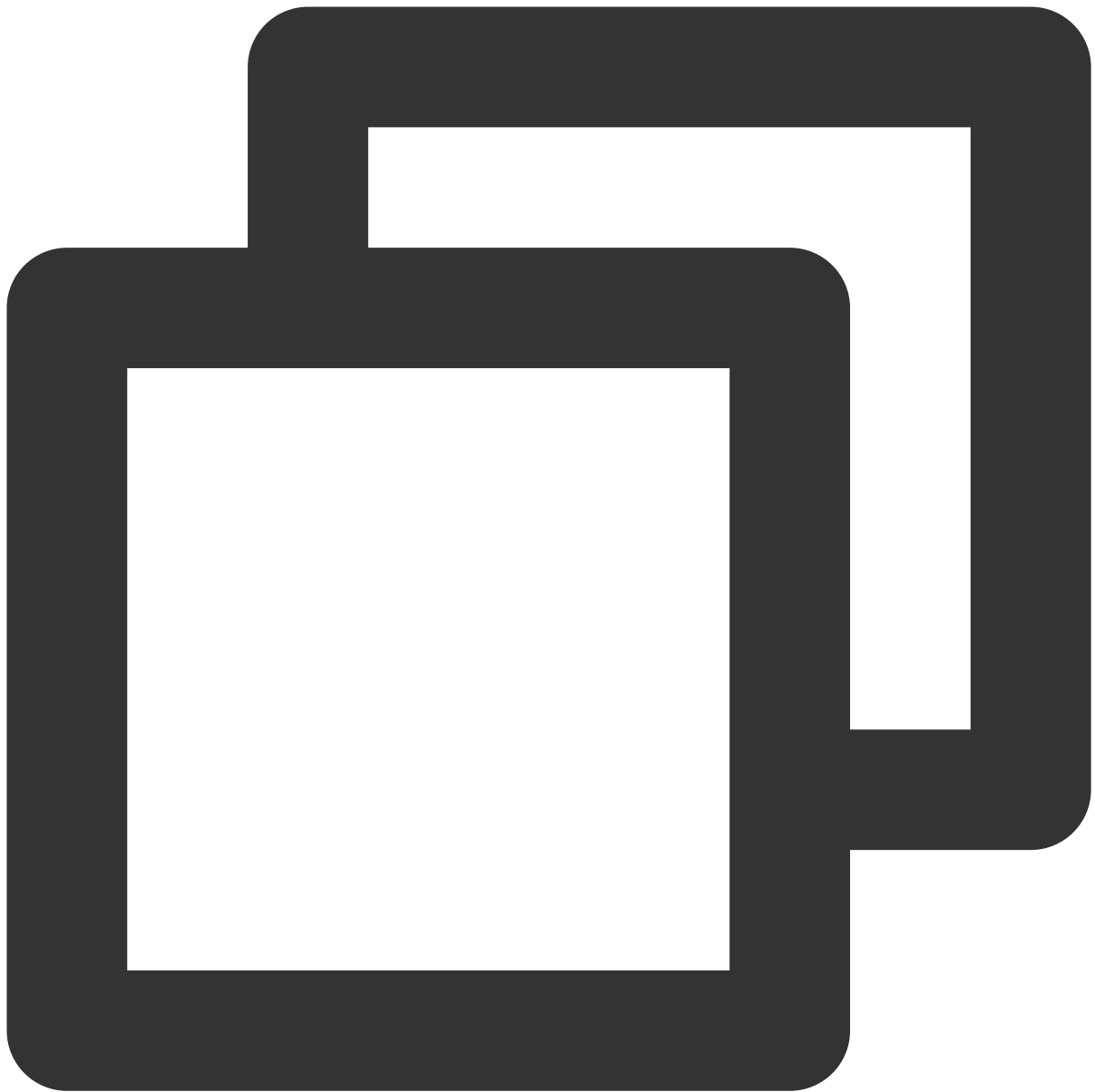
This API is used to set the alias and profile photo. The alias cannot exceed 500 bytes, and the profile photo is specified by a URL.



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

### **enableMultiDeviceAbility**

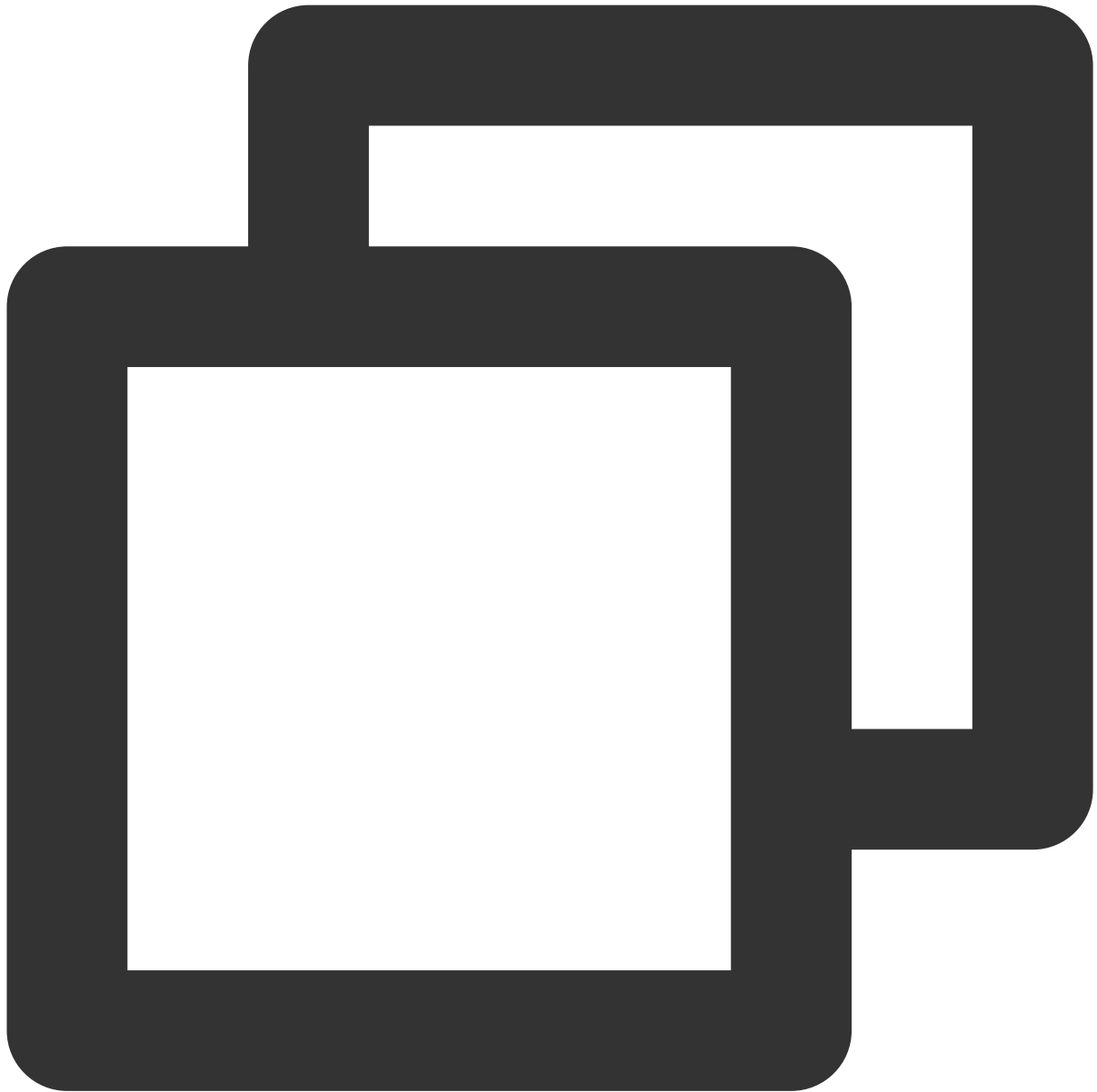
This API is used to set whether to enable multi-device login for `TUICallEngine` (supported by the premium package).



```
Future<TUIResult> enableMultiDeviceAbility(bool enable)
```

### **setVideoRenderParams**

Set the rendering mode of video image.



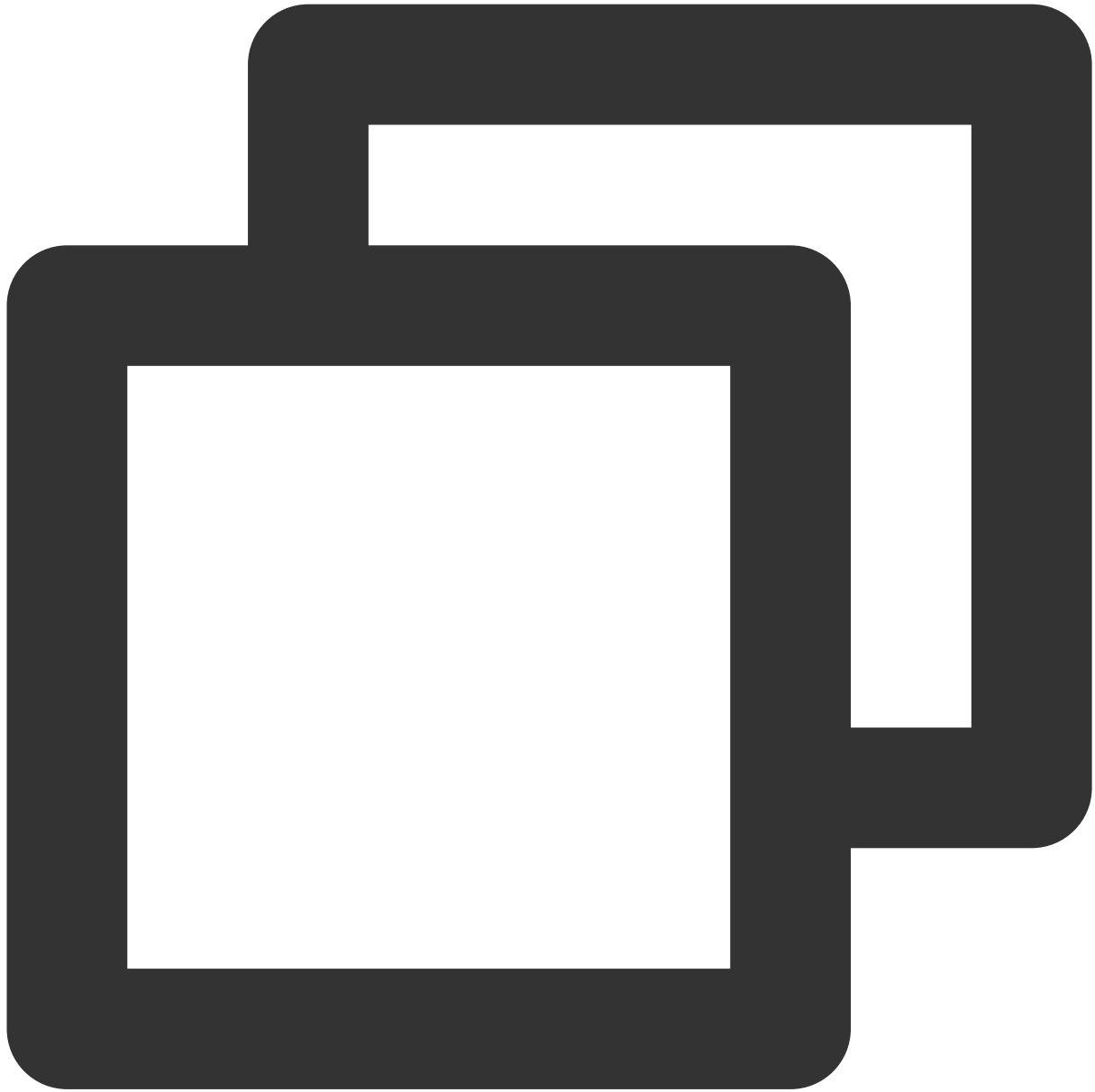
```
Future<TUIResult> setVideoRenderParams(String userId, VideoRenderParams params)
```

Parameter	Type	Description
userId	String	The target user ID.
params	<a href="#">VideoRenderParams</a>	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.

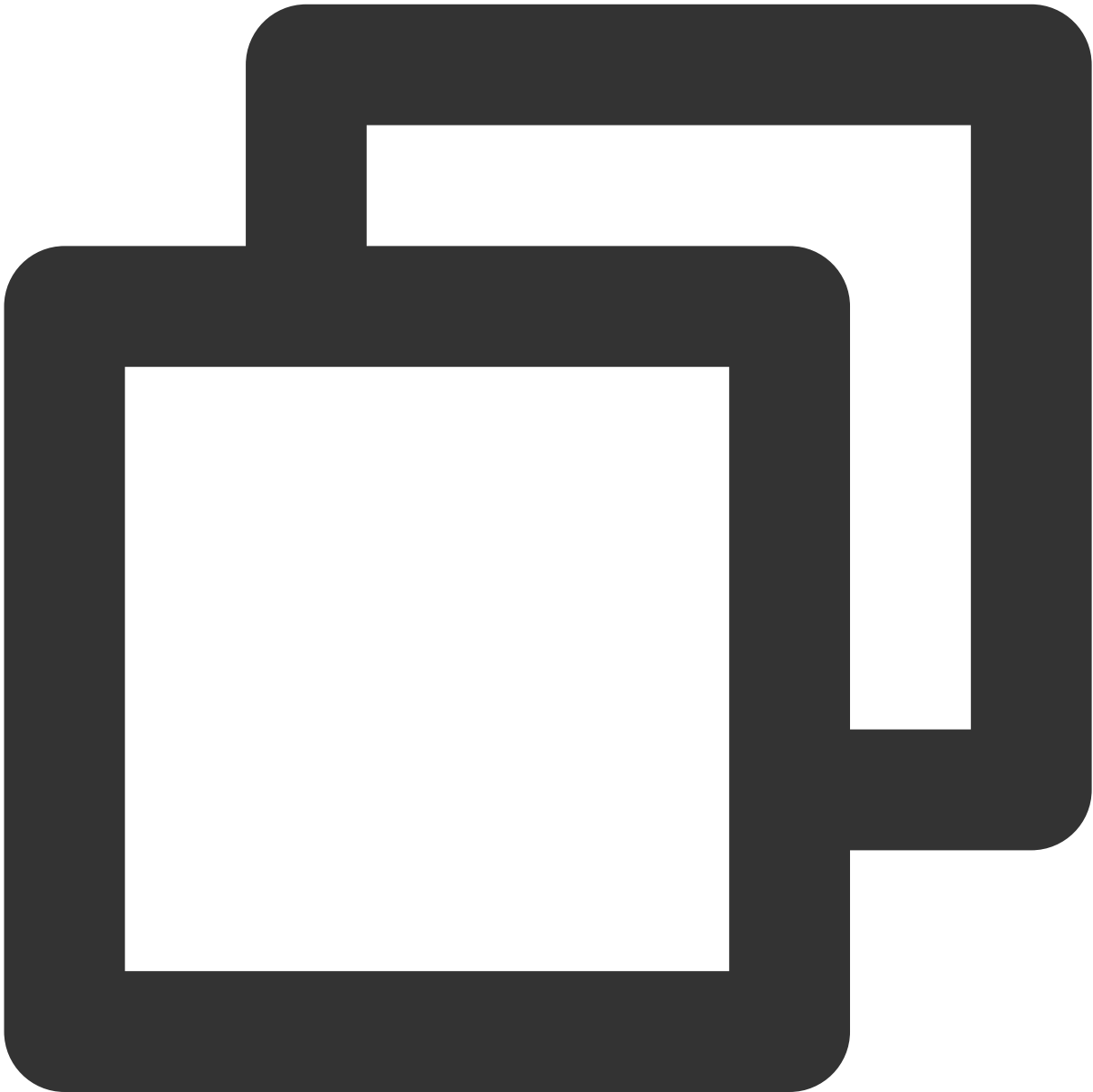


```
Future<TUIResult> setVideoEncoderParams (VideoEncoderParams params)
```

Parameter	Type	Description
params	<a href="#">VideoEncoderParams</a>	Video encoding parameters

## queryRecentCalls

Query call record.

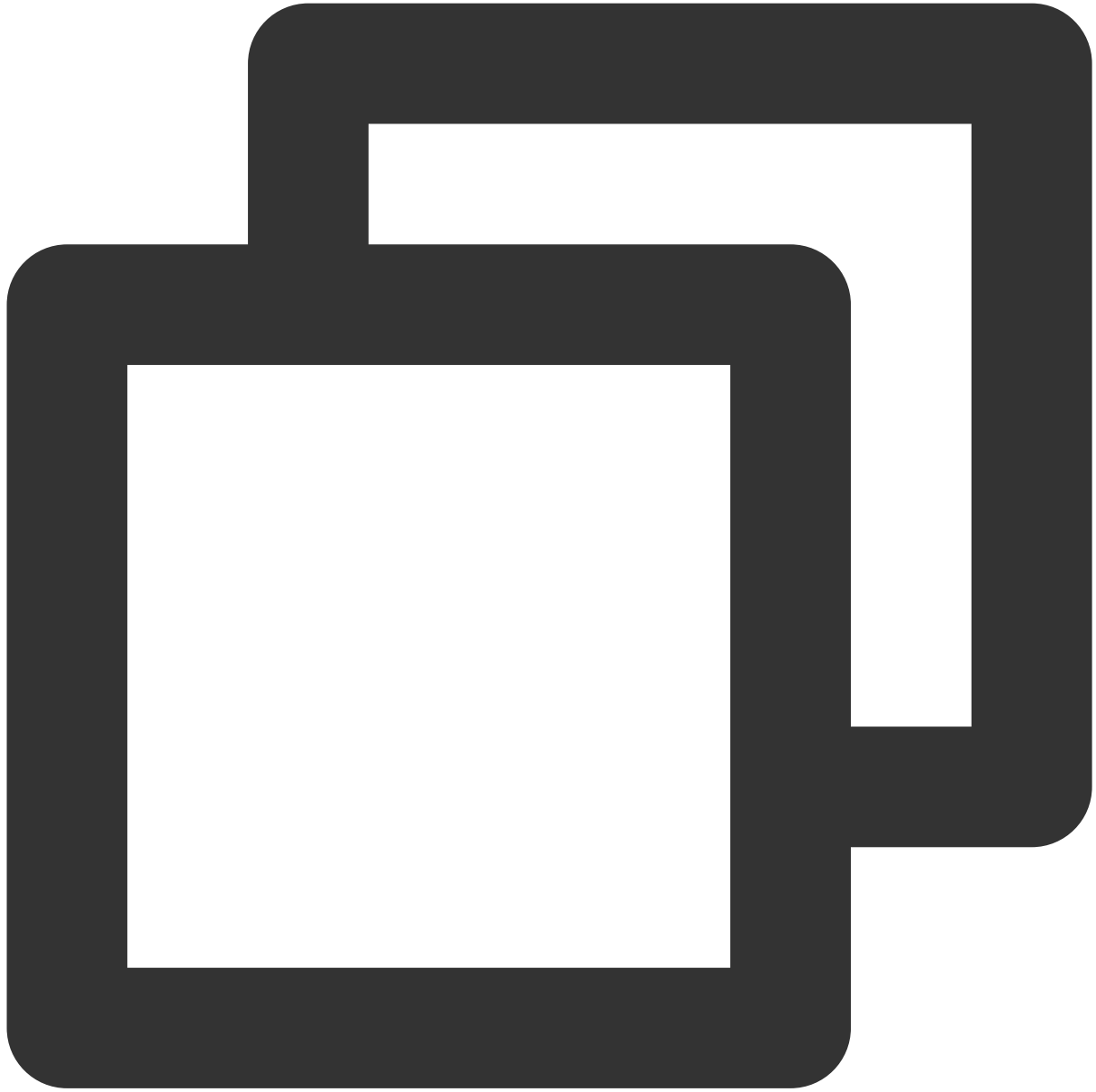


```
Future<void> queryRecentCalls(TUICallRecentCallsFilter filter, TUIValueCallback cal
```

Parameter	Type	Description
filter	<a href="#">TUICallRecentCallsFilter</a>	Filter condition

## deleteRecordCalls

Delete call record.



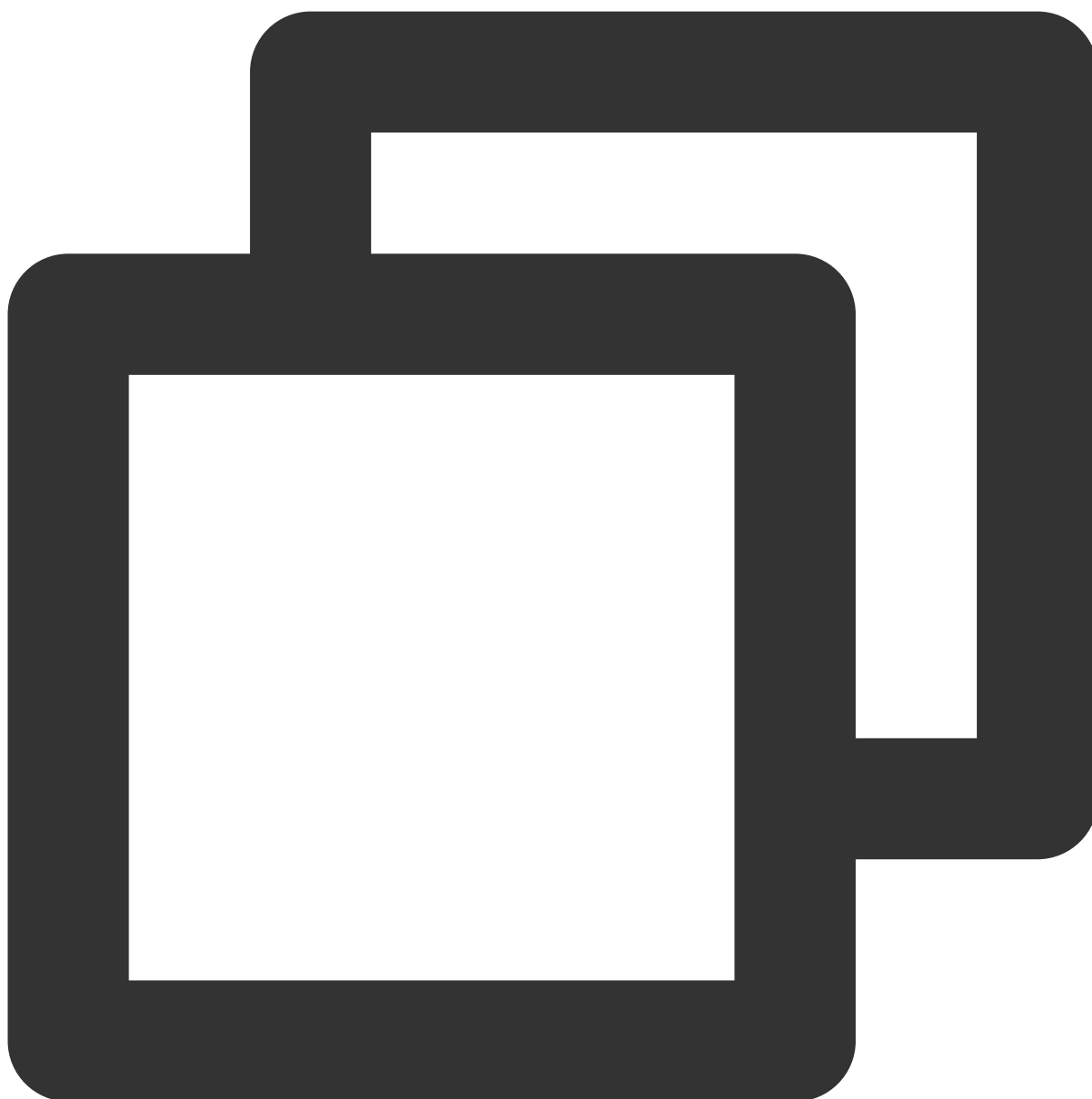
```
Future<void> deleteRecordCalls(List<String> callIdList, TUIValueCallback callback)
```

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

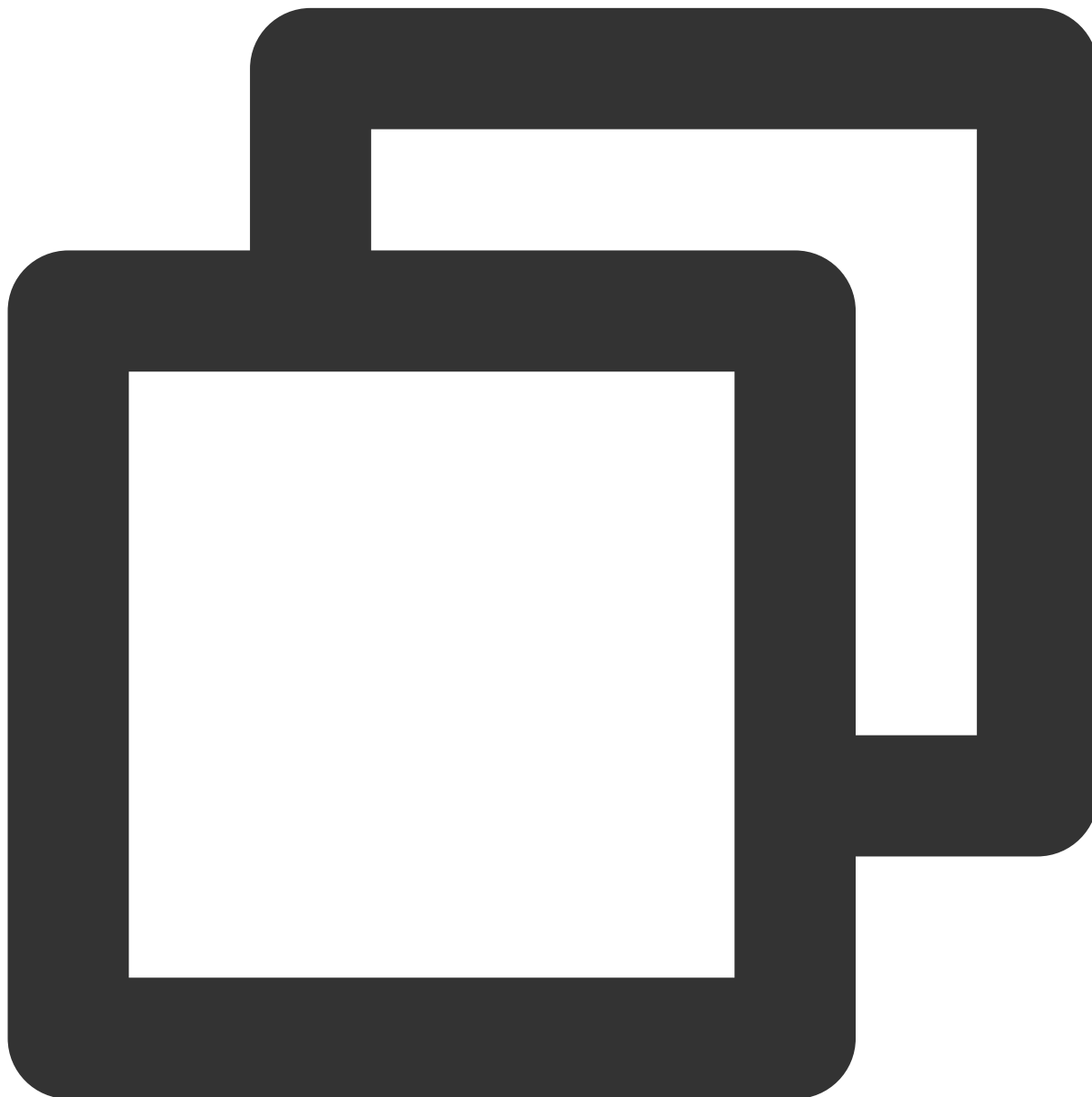
API	Description
<code>onError</code>	A call occurred during the call.
<code>onCallReceived</code>	A call invitation was received.
<code>onCallCancelled</code>	The call was canceled.
<code>onCallBegin</code>	The call was connected.
<code>onCallEnd</code>	The call ended.
<code>onCallMediaTypeChanged</code>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

    }
});

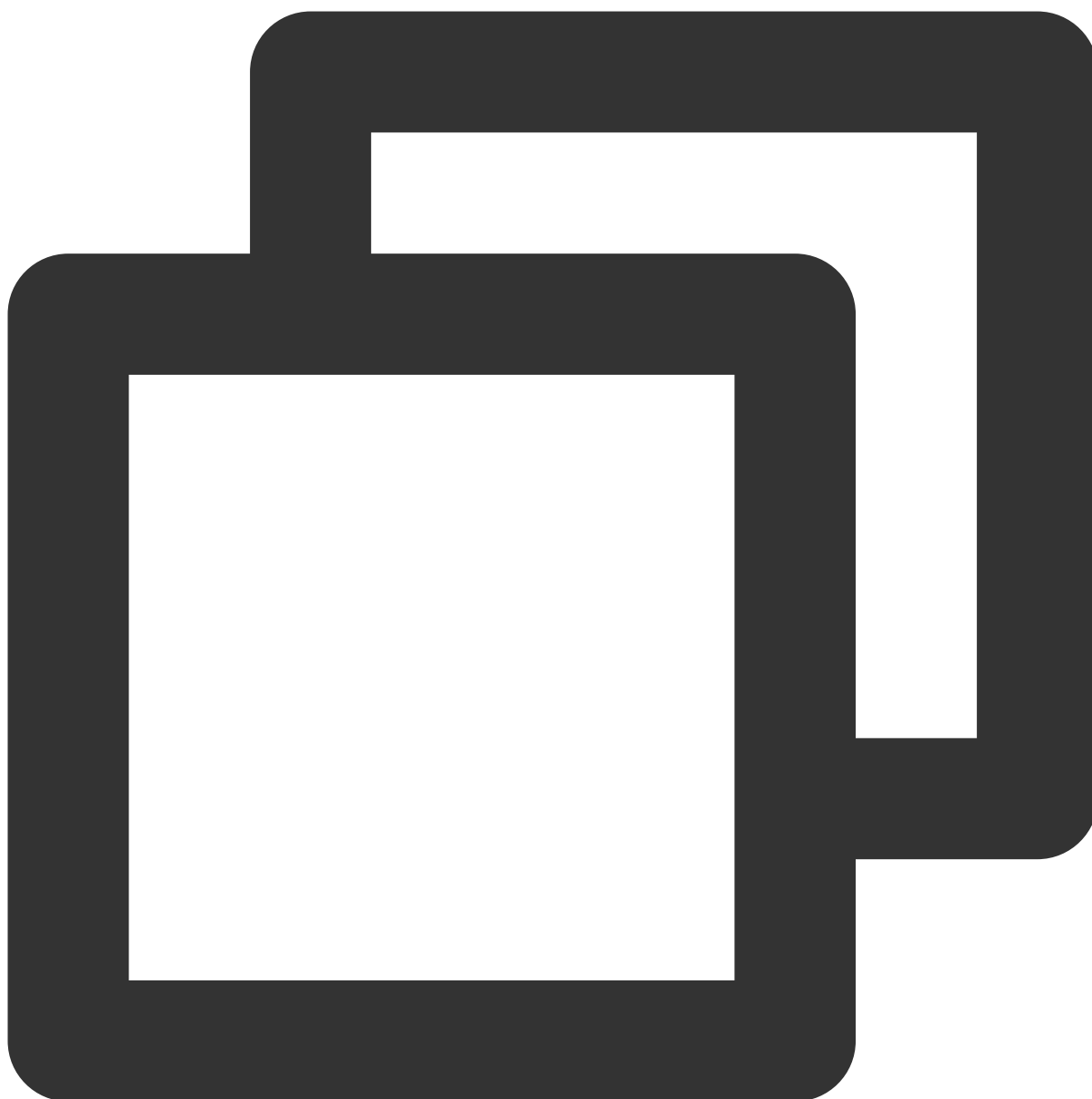
```

## 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) {  
  
    }  
));
```

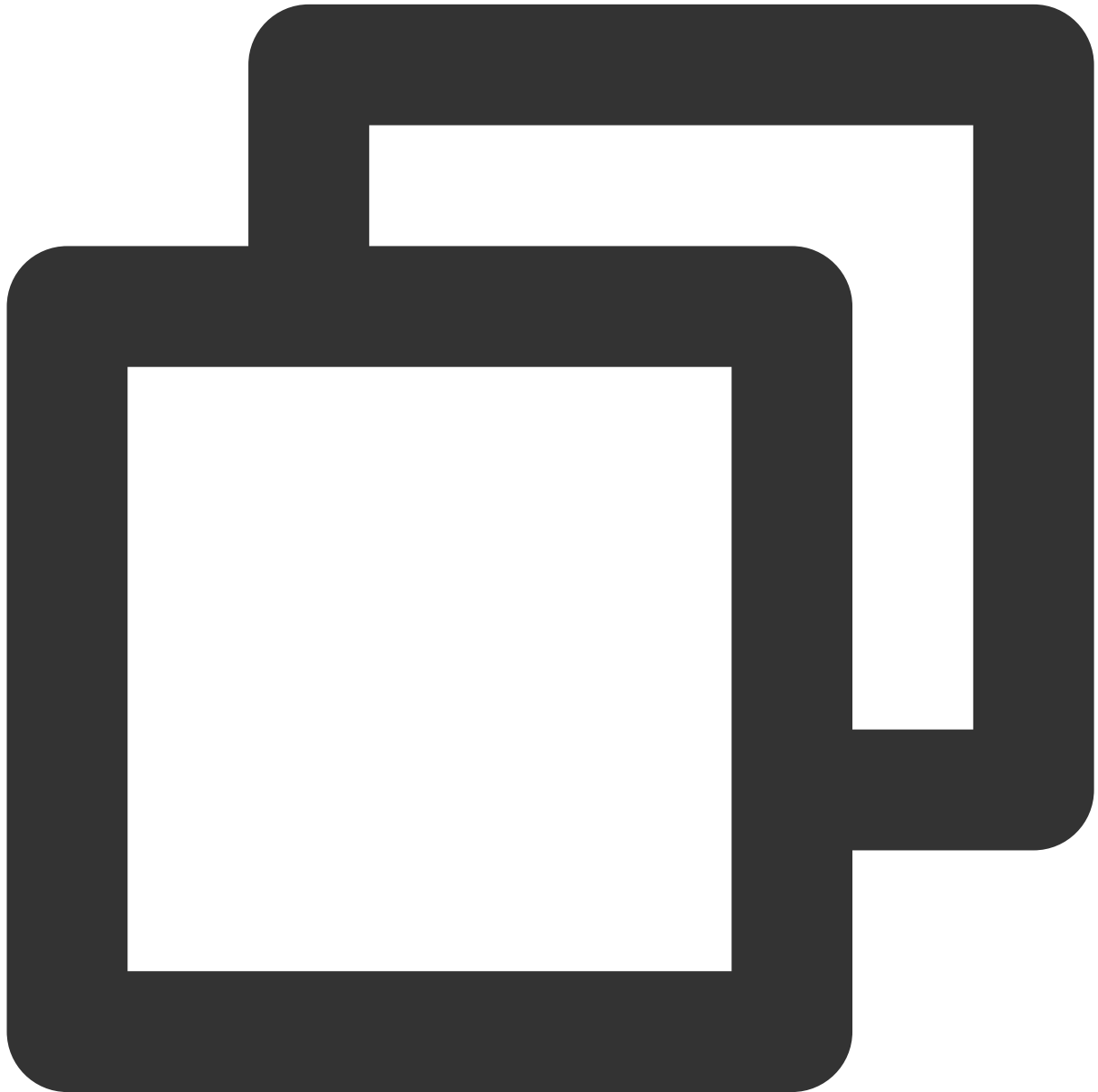
The parameters are described below:

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	<a href="#">TUICallMediaType</a>	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.

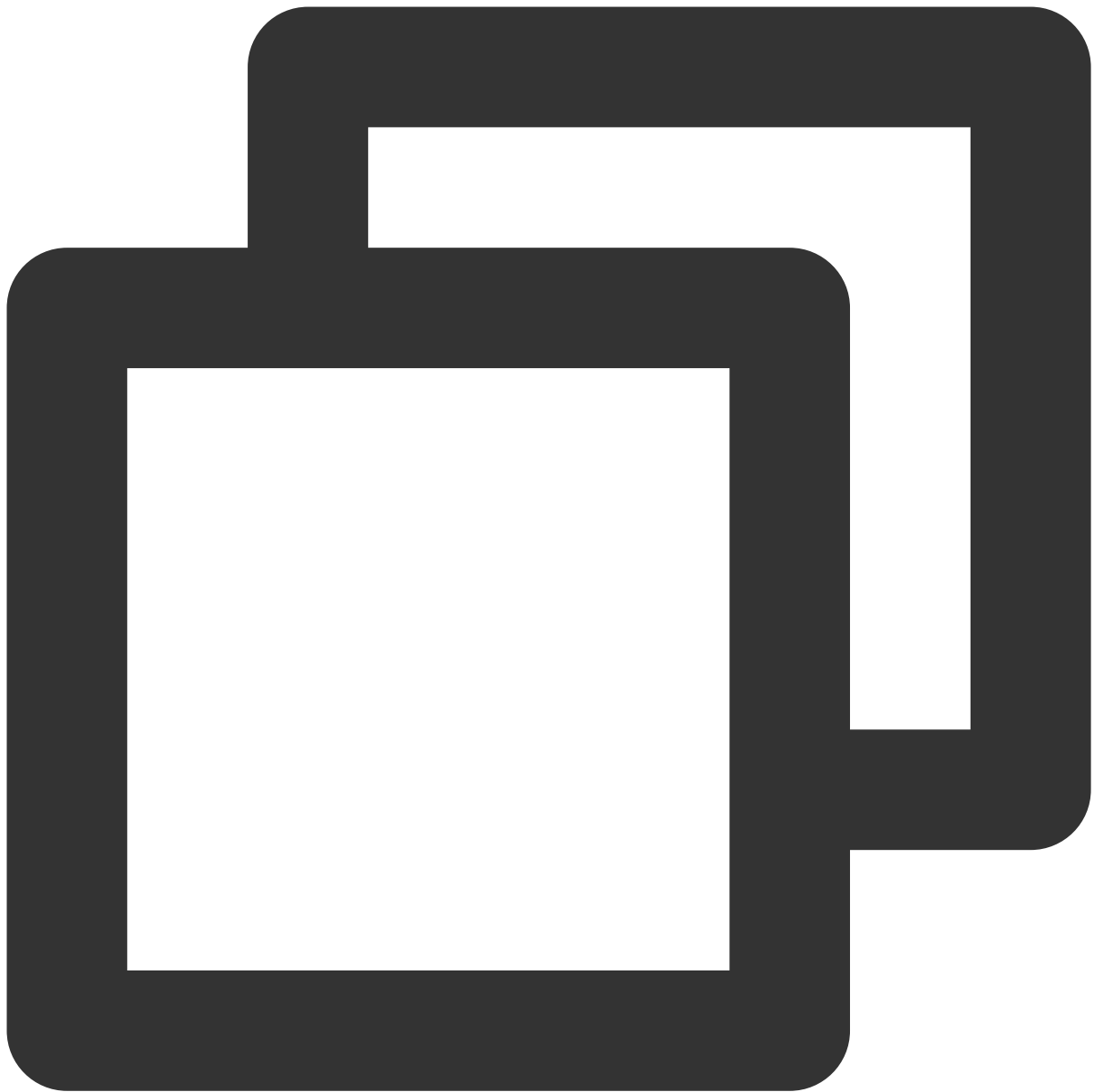
Call cancellation by the caller: The caller receives the callback (userId is himself); the callee receives the callback (userId is the ID of the caller)

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) {  
  
    }  
));
```

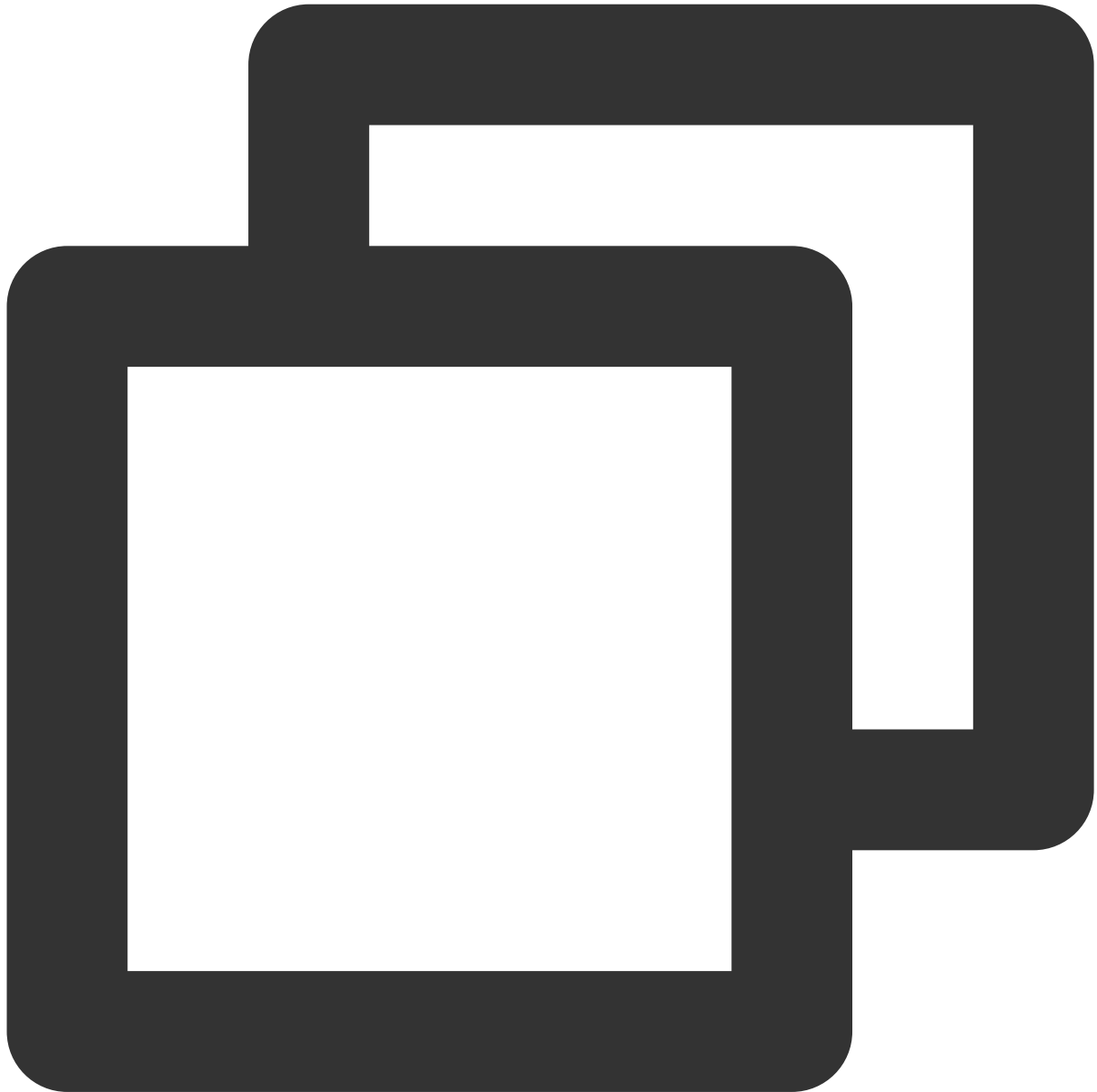
The parameters are described below:

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  
    })  
));
```

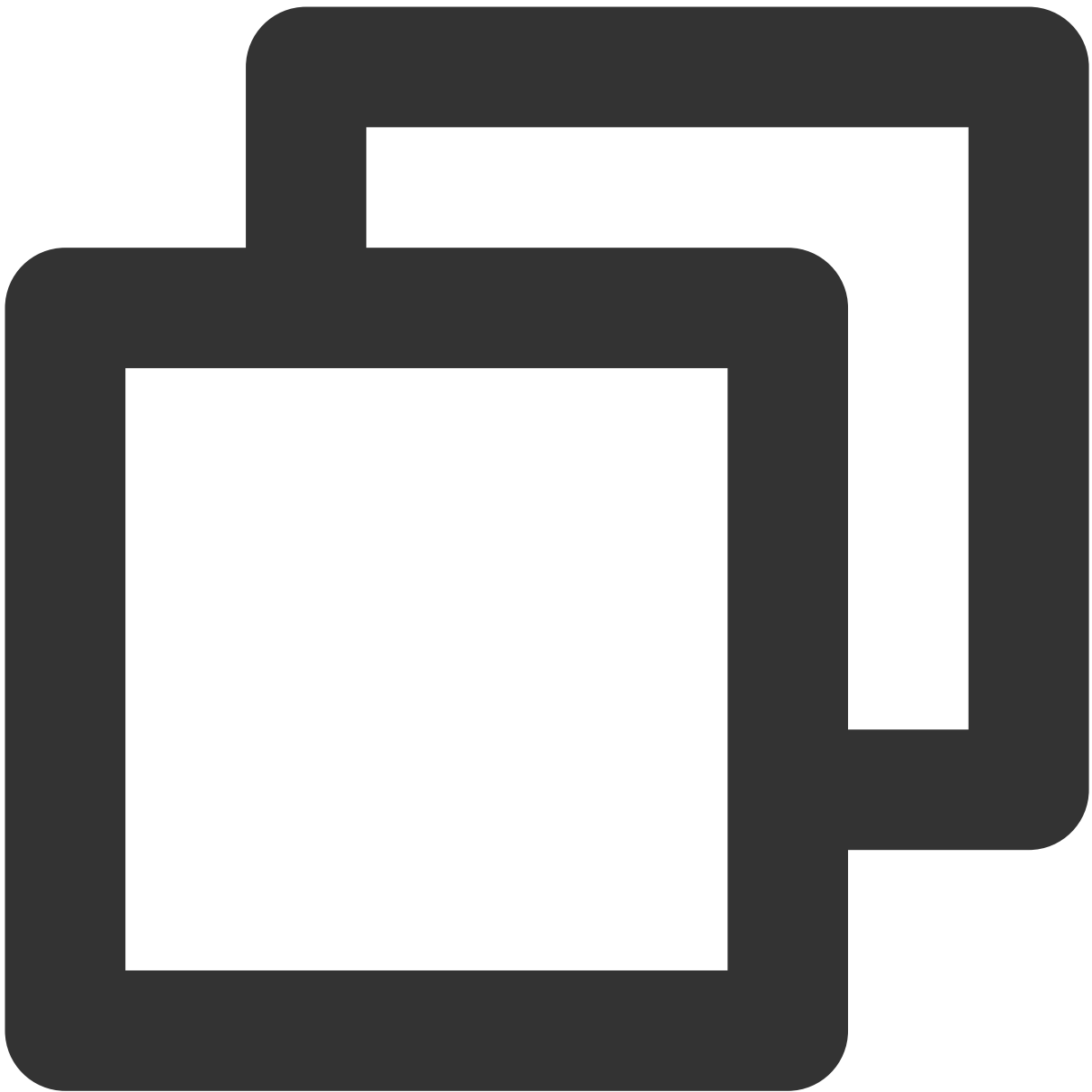
The parameters are described below:

--	--	--

Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.
callMediaType	<a href="#">TUICallMediaType</a>	The call type, which can be video or audio.
callRole	<a href="#">TUICallRole</a>	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 callRole)  
    )  
));
```

The parameters are described below:

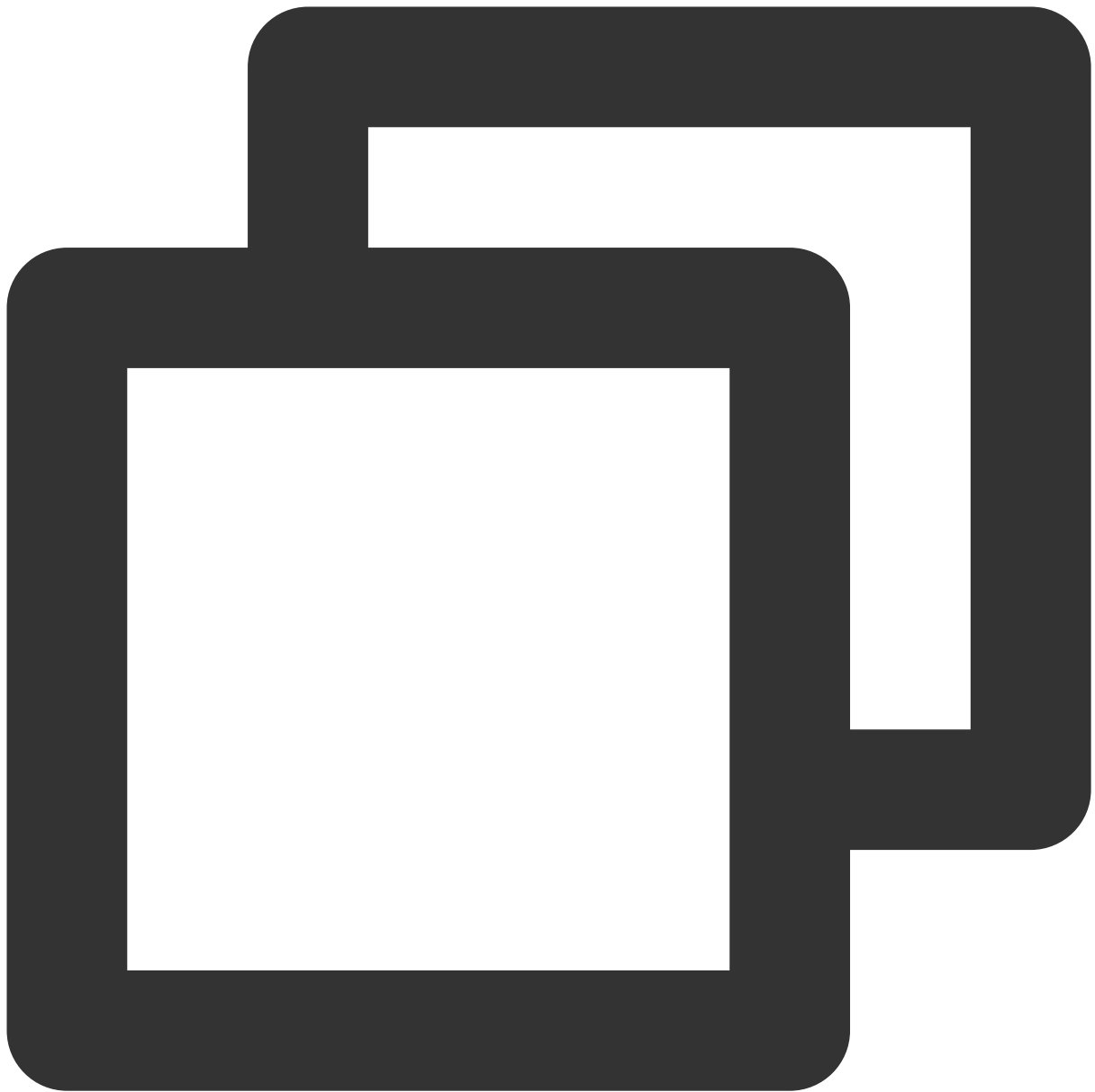
Parameter	Type	Description
roomId	<a href="#">TUIRoomId</a>	The room ID.

callMediaType	<a href="#">TUICallMediaType</a>	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	<a href="#">TUICallMediaType</a>	The call media type before the change.

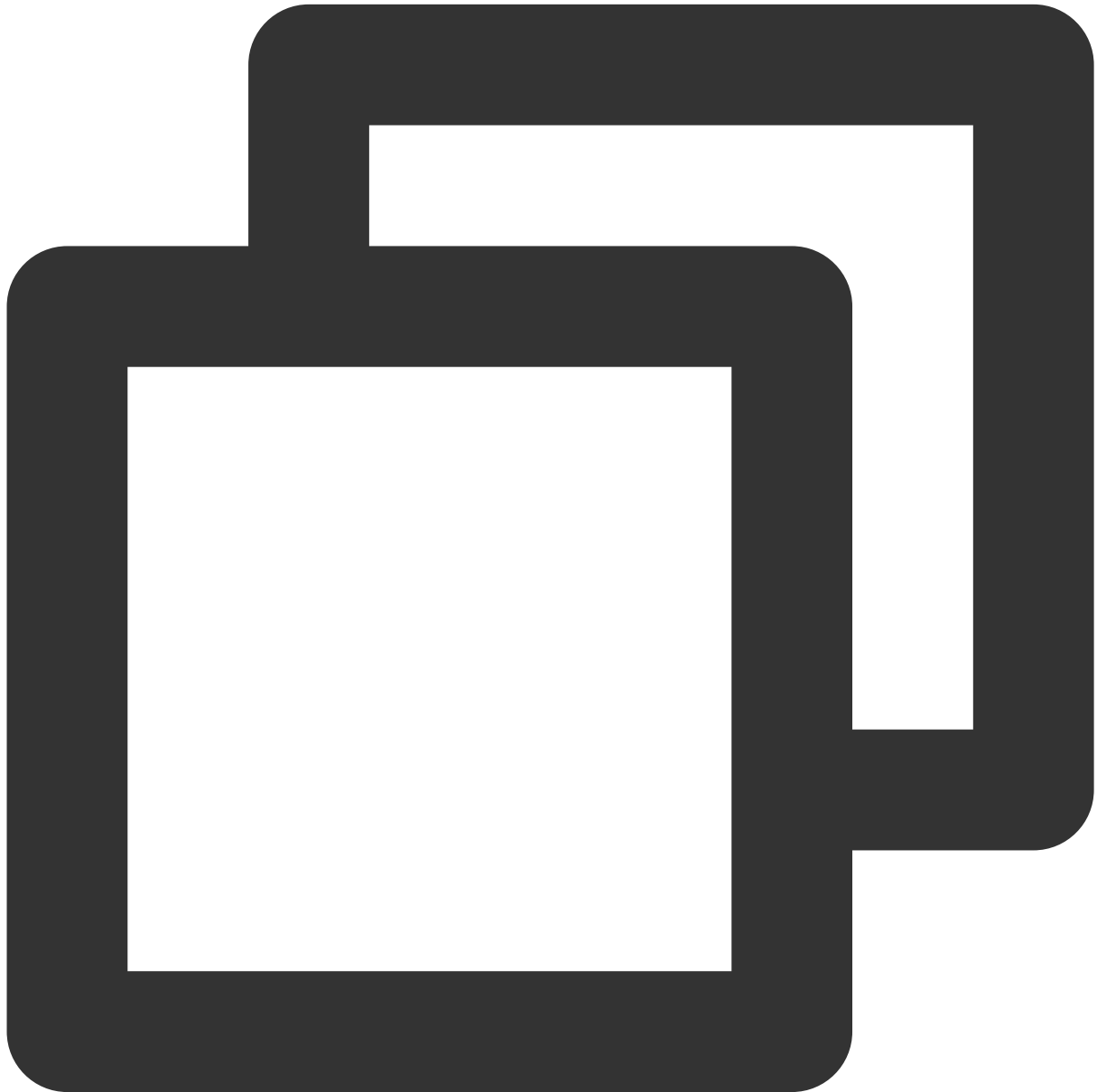
newCallMediaType

[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) {  
  
    }  
)
```

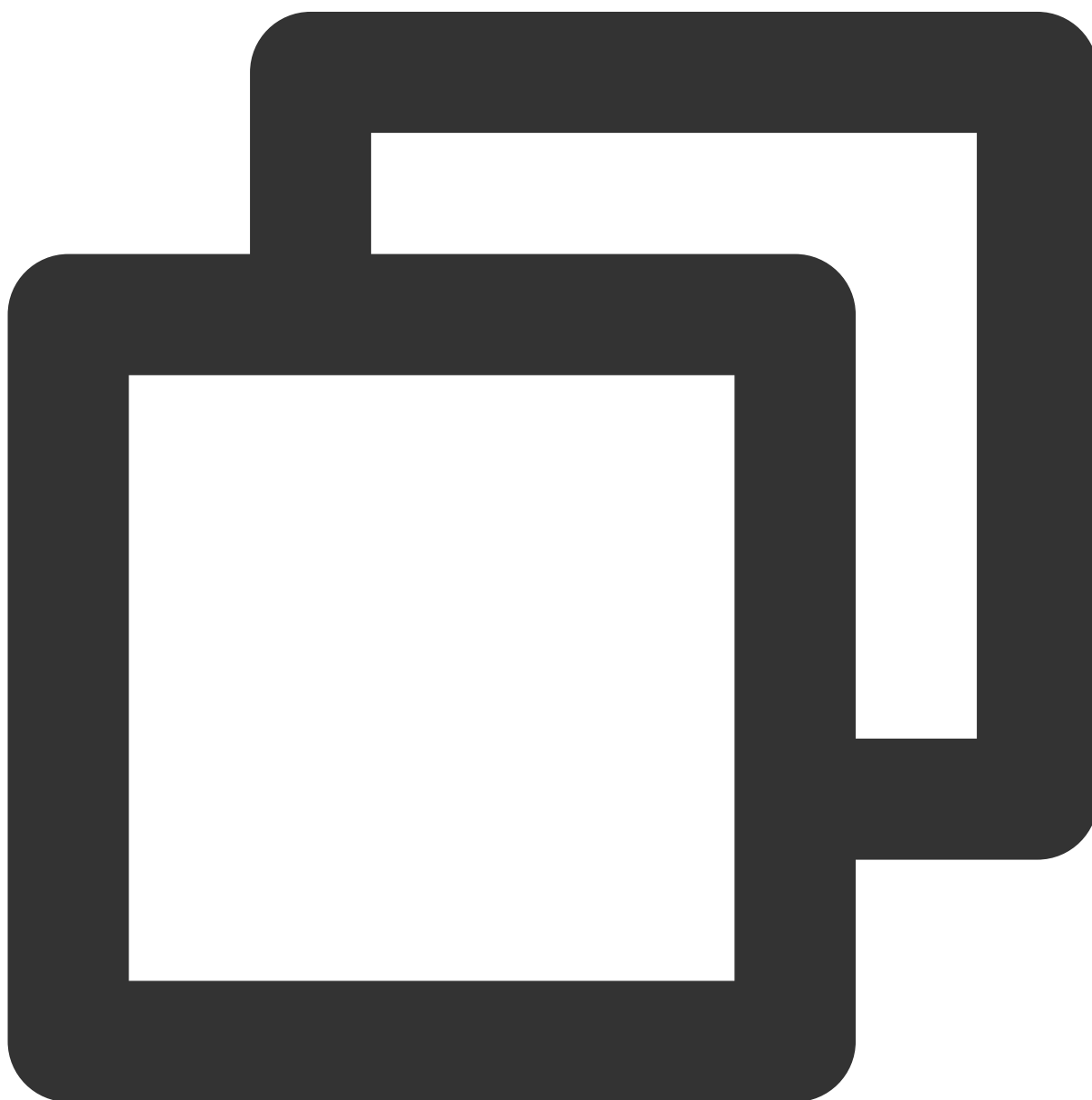
```
));
```

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) {  
  
    }  
));
```

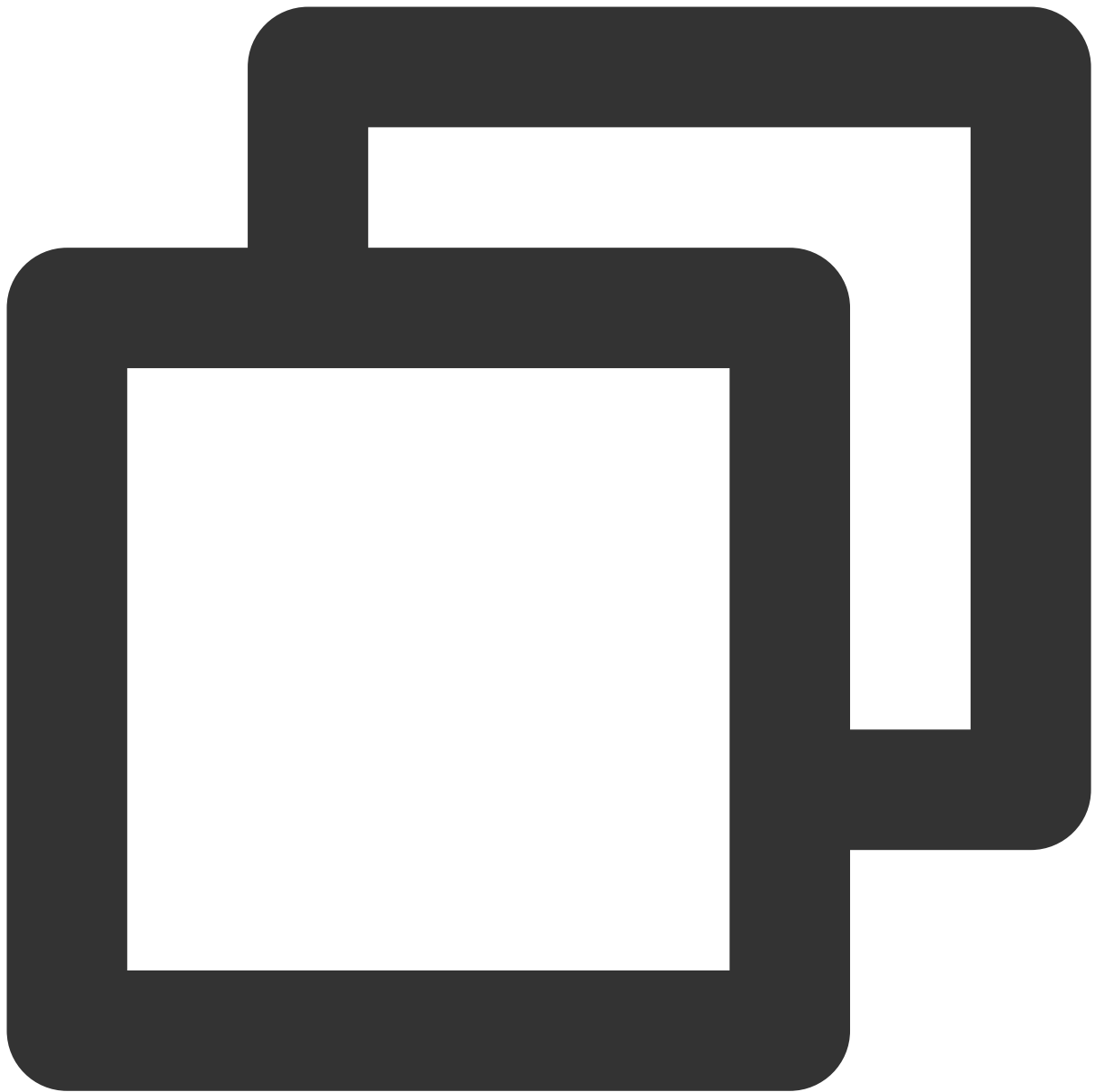
The parameters are described below:

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

### onUserLineBusy

A user is busy.





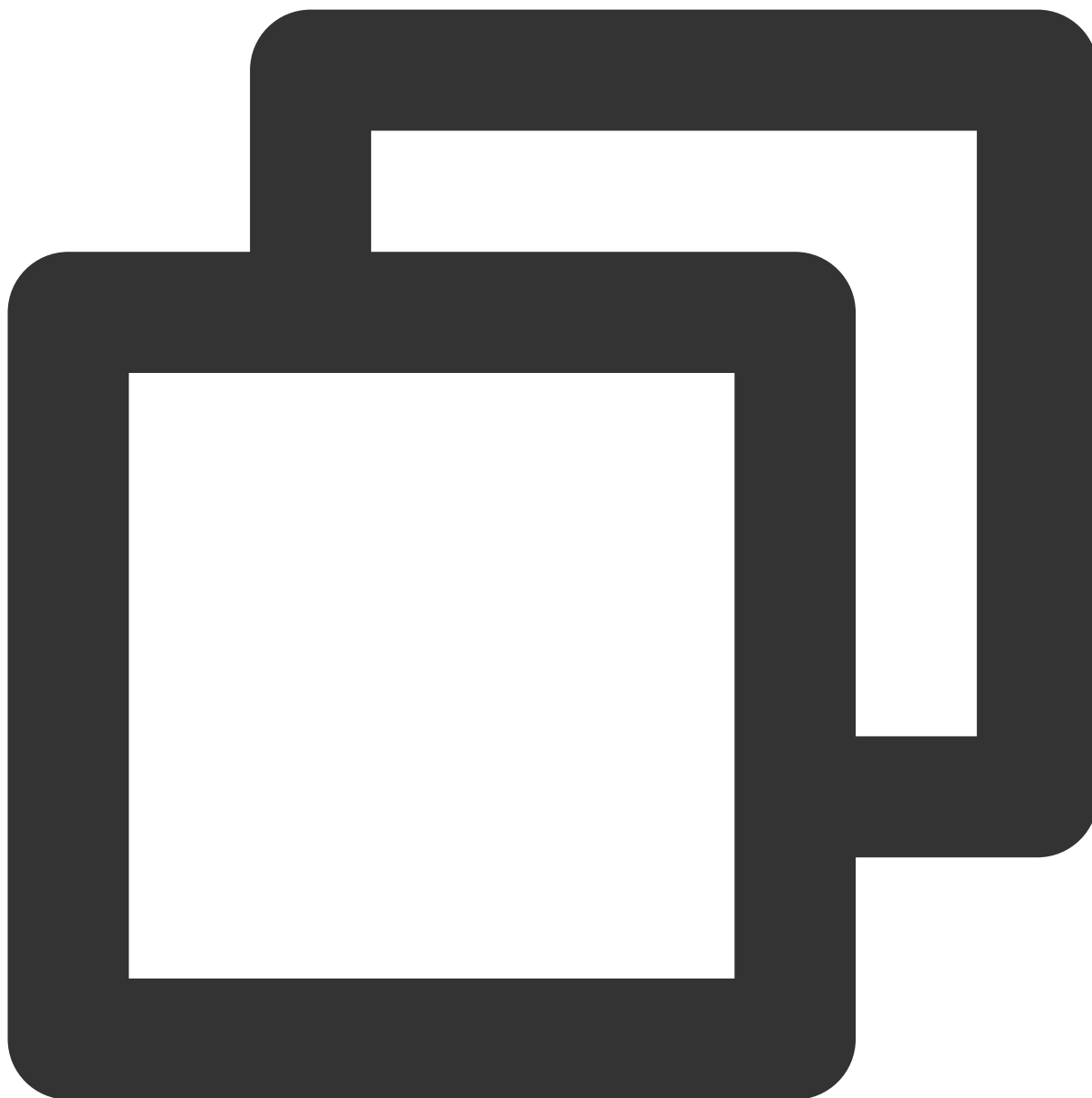
```
TUICallEngine.instance.addObserver(TUICallObserver(  
    onUserLineBusy: (String onUserLineBusy) {  
  
    },  
));
```

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) {  
  
    }  
));
```

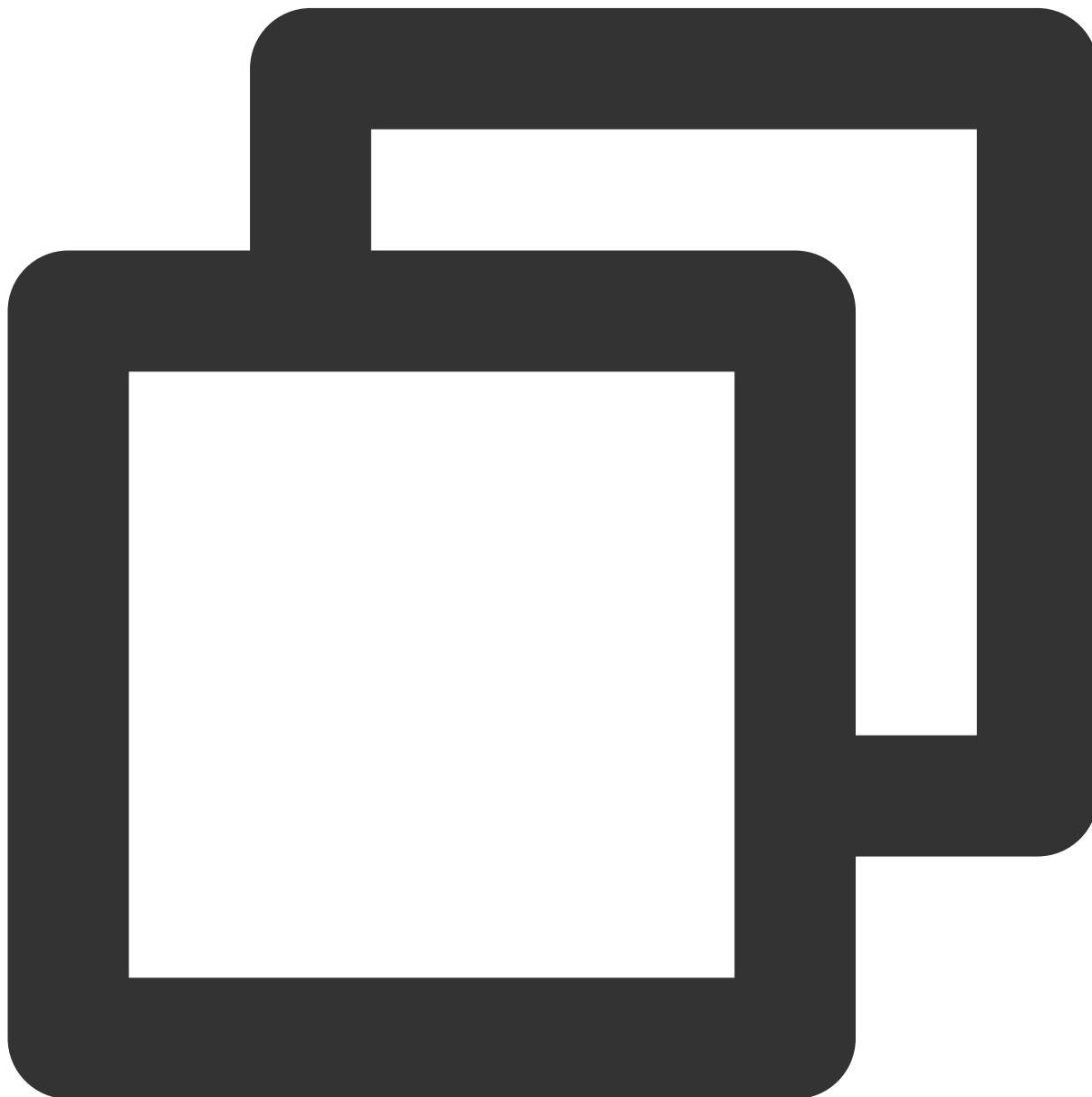
The parameters are described below:

--	--	--

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) {  
    }
```

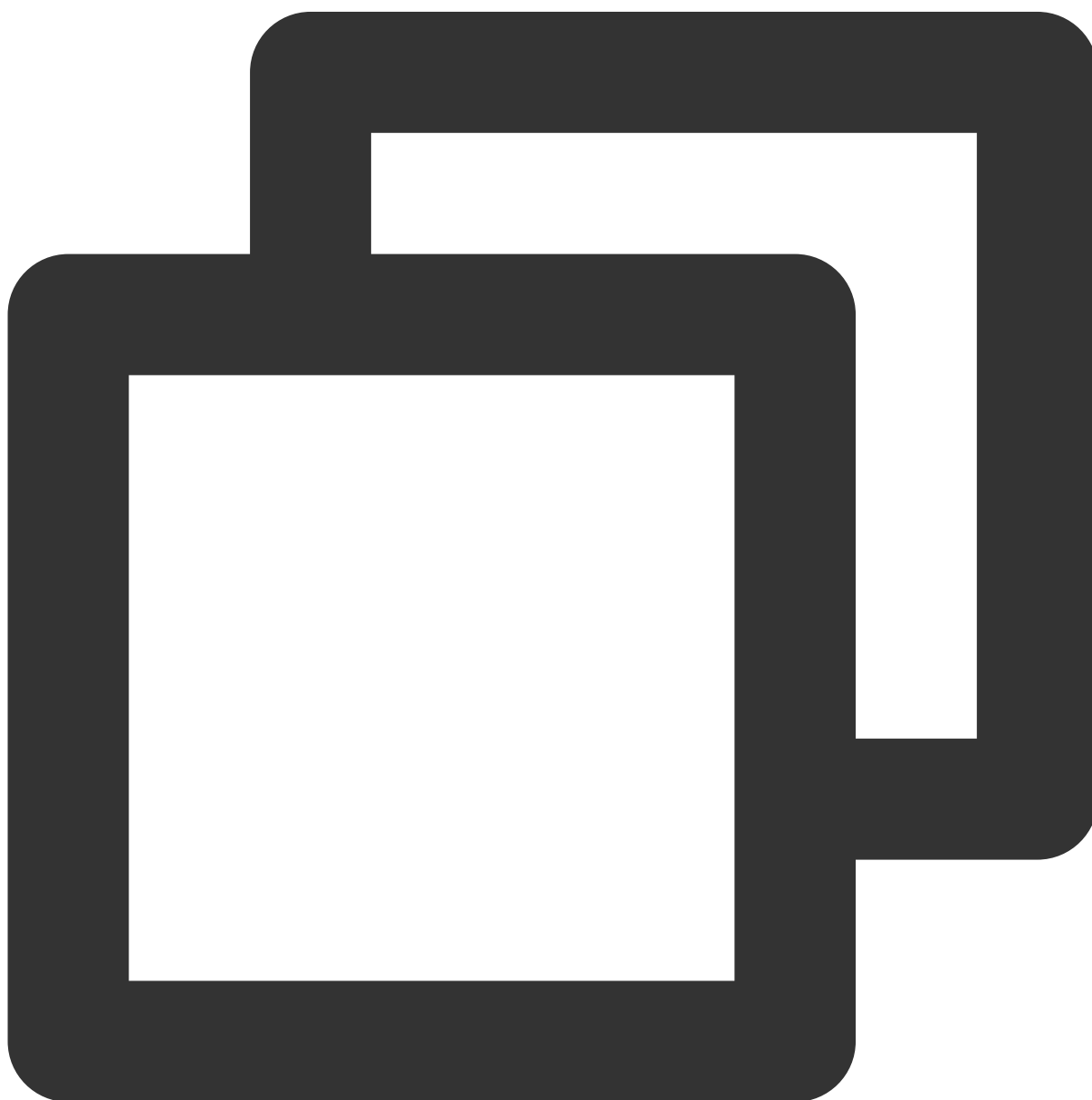
```
));
```

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.



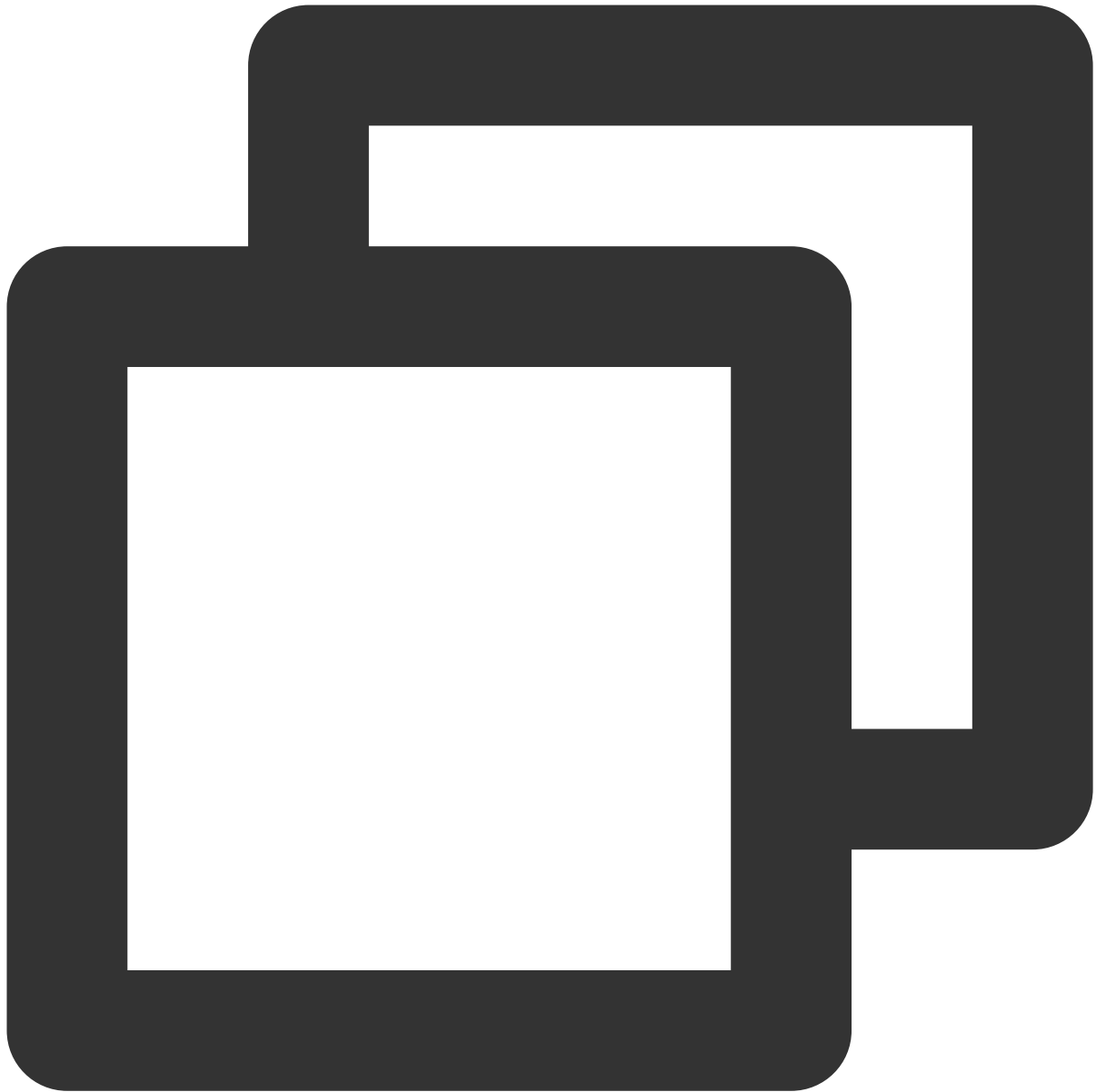
```
TUICallEngine.instance.addObserver(TUICallObserver(  
    onUserVideoAvailable: (String userId, bool isVideoAvailable) {  
    }  
));
```

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.

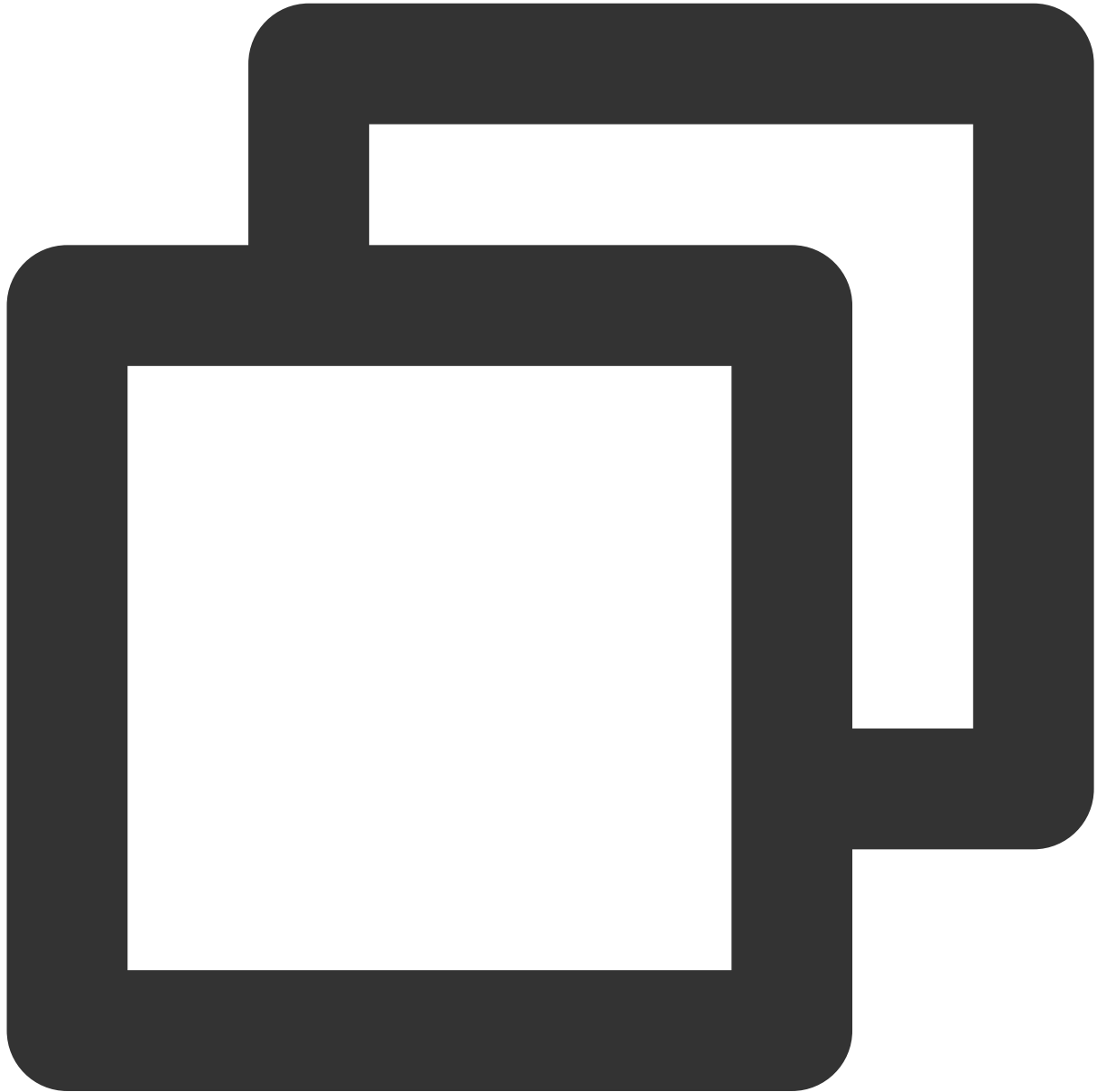
isAudioAvailable

bool

Whether the user has audio.

**onUserVoiceVolumeChanged**

The volumes of all users.



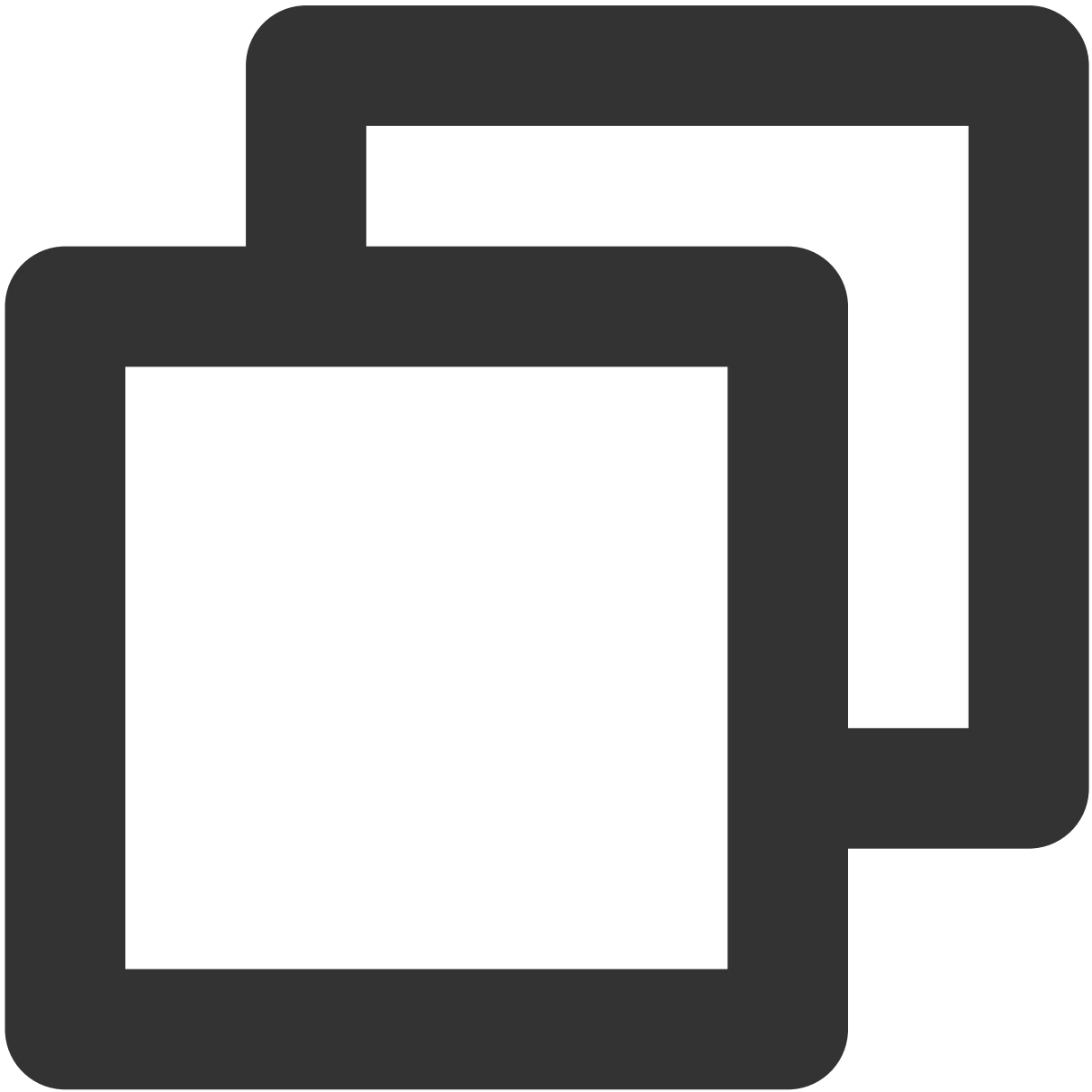
```
TUICallEngine.instance.addObserver(TUICallObserver(  
    onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {  
  
    }  
));
```

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.





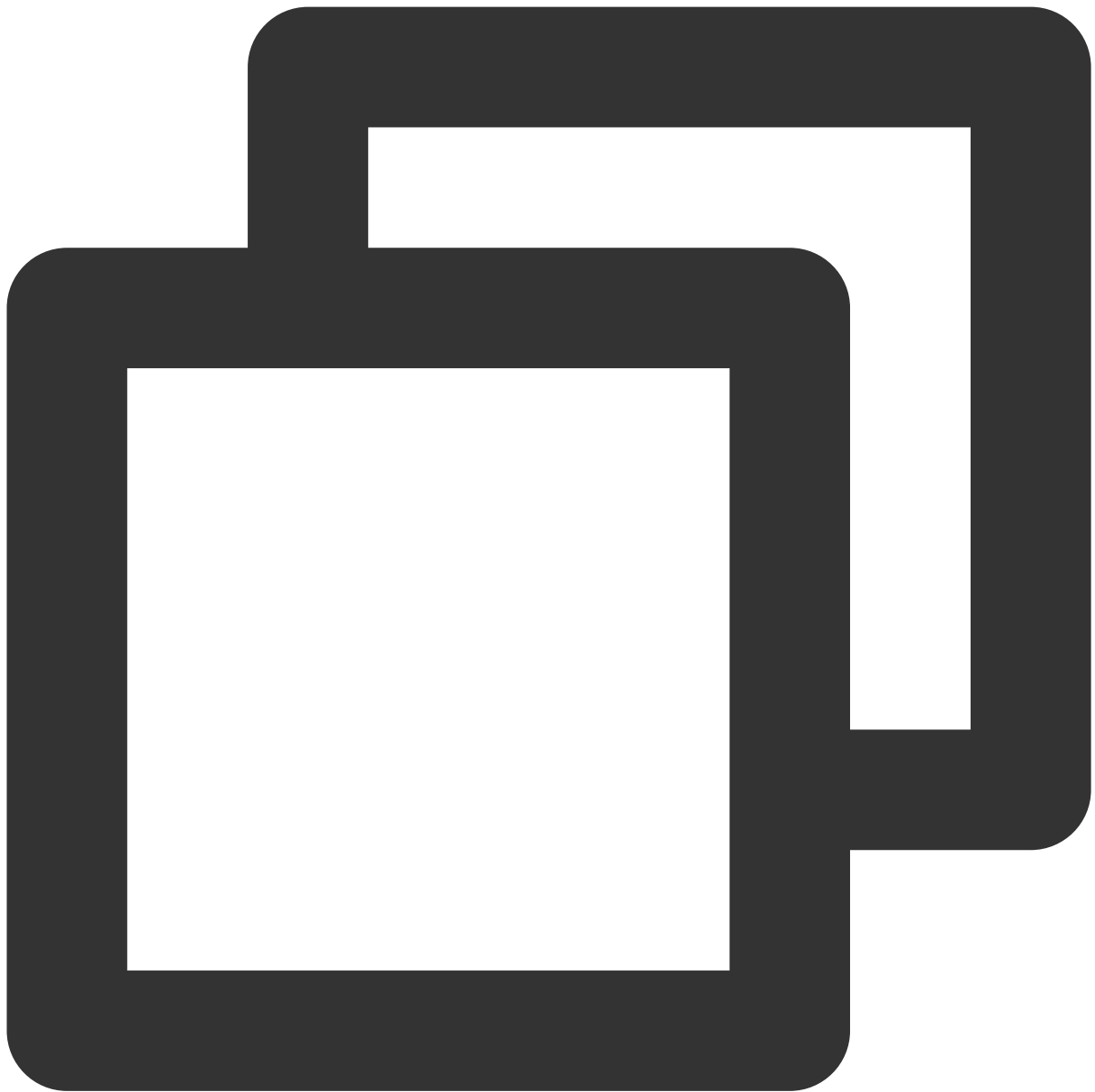
```
TUICallEngine.instance.addObserver(TUICallObserver(  
    onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {  
  
        }  
));  
  
class TUINetworkQualityInfo {  
    String userId;  
    TUINetworkQuality quality;  
    TUINetworkQualityInfo({required this.userId, required this.quality});  
}  
  
enum TUINetworkQuality {  
    unknown,  
    excellent,  
    good,  
    poor,  
    bad,  
    vBad,  
    down  
}
```

The parameters are described below:

Parameter	Type	Description
networkQualityList	List< <a href="#">TUINetworkQualityInfo</a> >	he current network conditions for all users (userId).

## onKickedOffline

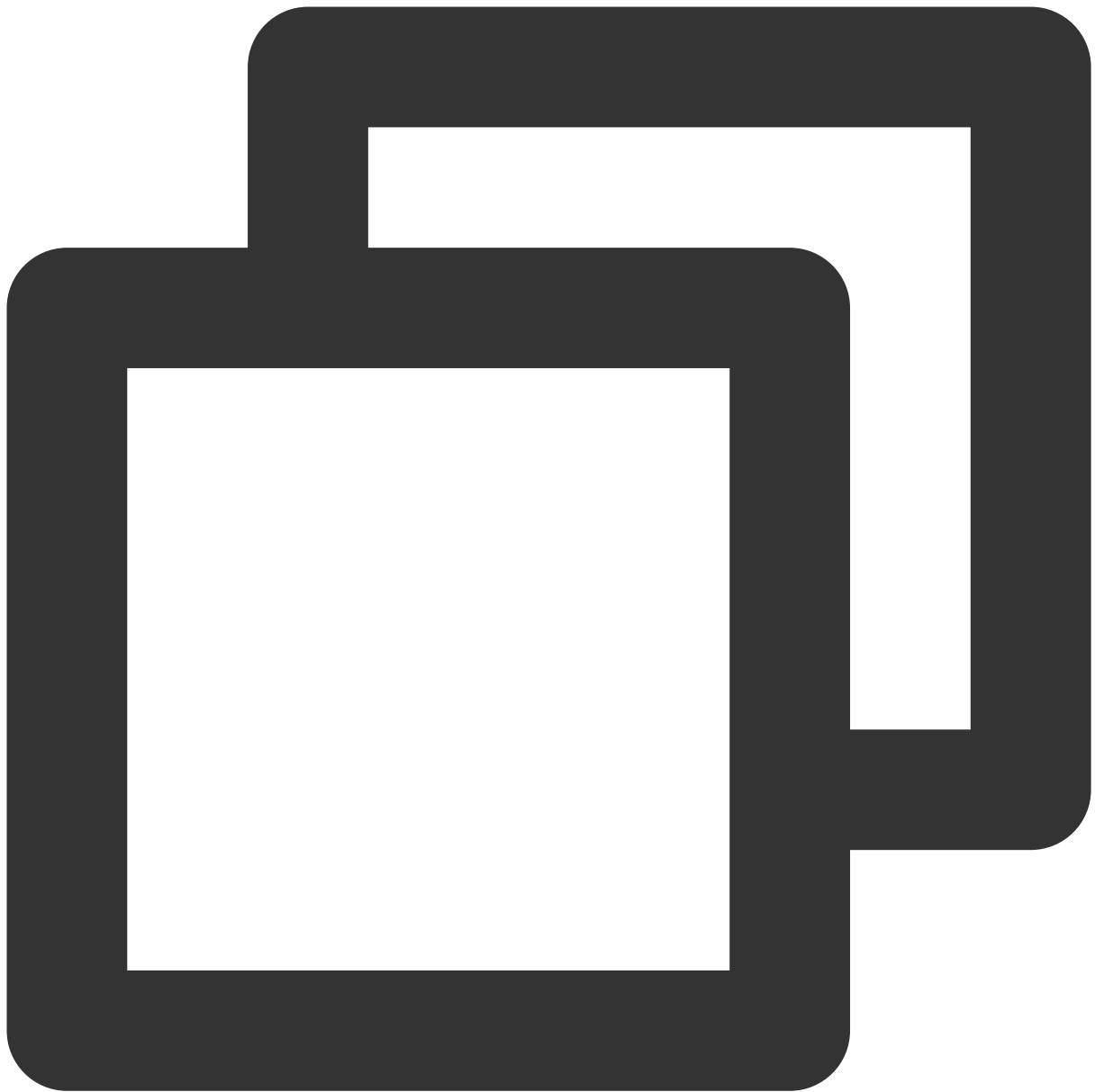
The current user is kicked offline: at this time, the UI can prompt the user and call initialization again.



```
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	<a href="#">FillMode</a>	Video image fill mode
rotation	<a href="#">Rotation</a>	Video image rotation direction

### VideoEncoderParams

--	--	--

Name	Type	Description
resolution	<a href="#">Resolution</a>	Video resolution
resolutionMode	<a href="#">ResolutionMode</a>	Video aspect ratio mode

## TUICallParams

Name	Type	Description
roomId	<a href="#">TUIRoomId</a>	Room Id.
offlinePushInfo	<a href="#">TUIOfflinePushInfo</a>	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 <code>setAndroidFCMChannelID</code> to set the <code>channelID</code> to take effect
<code>androidOPPOChannelID</code>	String	Offline push setting channel ID of OPPO mobile phone 8.0 system and above
<code>androidVIVOClassification</code>	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 <code>setAndroidVIVOCategory</code> to set the message category). 0: operation message 1: system message, the default value is 1
<code>androidXiaoMiChannelID</code>	String	Channel ID of Xiaomi mobile phone system 8.0 and above
<code>androidFCMChannelID</code>	String	Channel ID of FCM channel mobile phone system 8.0 and above
<code>androidHuaWeiCategory</code>	String	Huawei Push Message Classification
<code>isDisablePush</code>	bool	Whether to turn off push (by default, push is enabled)
<code>iOSPushType</code>	<a href="#">TUICallIOSOfflinePushType</a>	iOS offline push type, default: APNs

## TUICallRecords

Call recording information

Value	Type	Description
<code>callId</code>	String	Call recording ID.
<code>inviter</code>	String	Inviter ID.
<code>inviteList</code>	List<String>	List of invited user IDs.
<code>scene</code>	<a href="#">TUICallScene</a>	Call scenario.
<code>mediaType</code>	<a href="#">TUICallMediaType</a>	Media type.
<code>groupId</code>	String	Group ID.
<code>role</code>	<a href="#">TUICallRole</a>	Role.

result	<a href="#">TUICallResultType</a>	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	<a href="#">TUICallResultType</a>	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 <a href="#">open the free experience</a> in the console or <a href="#">purchase the official package</a> .
ERROR_PACKAGE_NOT_SUPPORTED	-1002	The package you purchased does not support this ability. You can refer to console to purchase <a href="#">Upgrade package</a> .
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 <a href="#">init</a> 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 <a href="#">Generating UserSig</a> .
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 reenale 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 reenale 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(Vue)

Last updated : 2024-06-14 16:27:48

## Note:

TUICallKit Vue 3 Github version: [Github TUICallKit Web](#).

## Version 3.2.9 @2024.05.29

### Optimize

Added log reporting to the UI customization API.

## Version 3.2.8 @2024.05.27

### Fix

Fixed the SDK import ref path error issue.

## Version 3.2.7 @2024.05.17

### Add

Added UI components for mid-call .

Added custom UI API to support setting call background and hiding buttons.

Adjusted the parameter validation when initiating a call to support string room numbers.

## Version 3.2.4 @2024.05.06

### Add

Added support for background blur in video calls.

### Fix

Fixed the issue that image loading icon failed when packaging web project in uni-app.

Fixed the issue with the camera switch button during group calls.

Optimized the abnormal issue caused by clicking the button again after the application is stuck.

## Version 3.2.3 @2024.04.19

### Add

Added group call feature supporting front and rear camera switching.

### Optimize

Optimized [TUICallKit SDK](#) Readme.

## Version 3.2.2 @2024.03.25

### Add

Added a brand new UI visual effect, with clearer features and better experience.

### Optimize

Optimized data reporting when using TUICallKit.

### Add

Added language log reporting.

## Version 3.2.1 @2024.03.08

### Add

Added language log report.

## Version 3.2.0 @2024.02.23

### Add

Added default offline push parameters.

### Fix

Fixed the issue of no nickname in group calls.

## Version 3.1.9 @2024.01.30

### Fix

Fixed the issue of not displaying user information in group calls.

Fixed the issue that the OK button remains active in the selection component when there are no members to select.

Fixed the issue that the microphone is turned off during a call, resulting in no uplink audio streaming in subsequent calls (upgrade [trtc-cloud-js-sdk](#) to v2.2.7+).

## Version 3.1.8 @2024.01.19

### Fix

Fixed the issue of the selection component style affecting the page.

## Version 3.1.7 @2024.01.12

### Fix

Added retry mechanism for APIs to fix playback failure caused by not finding the dom node.

Fixed the issue of device list selection style on PC.

## Version 3.1.6 @2023.12.29

### Optimize

Optimized prompt messages during group calls.

Optimized display issues of over-long nicknames.

### Fix

Fixed the issue of requesting camera permission for voice calls.

Fixed the Destroyed problem.

Fixed hangup issues in floating windows across different calling scenarios.

Fixed the issue of displaying remote end in caller state.

Fixed the styling issues with incomplete filling on PC.

## Version 3.1.5 @2023.12.15



**Add**

Optimized the timing for obtaining device permissions. Do not obtain device permissions during initialization, and obtain only when using call.

**Fix**

Fixed the issue of [@tencentcloud/call-uikit-vue2](#) and [@tencentcloud/call-uikit-vue2.6](#) components not having a claim file.

## Version 3.1.4 @2023.12.01

**Add**

Integrated into Chat and added isFromChat reporting.

**Fix**

Fixed the issue that the button is active while loading.

## Version 3.1.3 @2023.11.17

**Add**

Added parameter check in API.

## Version 3.1.2 @2023.11.03

**Add**

Added the Invite User feature.

Added the feature of adding attendees by [joinInGroupCall](#).

Added the API [enableMuteMode](#) for muting incoming call ringtones.

**Fix**

Fixed the issue of incorrect display of remote stream microphone status.

## Version 3.1.0 @2023.10.20

**Add**

Added the floating window feature.

Added the [enableFloatWindow](#) API for enabling/disabling the floating window feature.

Desktop terminal supports camera and microphone device switching.

Added prompt message for failing to call users on the blocklist.

Added support for Japanese.

## Optimize

In video calls, the large screen displays the remote user by default.

## Version 3.0.8 @2023.10.10

### Add

Added version number, framework, and other information for reporting.

## Version 3.0.7 @2023.10.08

### Add

Added call duration display for desktop terminal video calls.

## Optimize

Optimized the issue with rounded corners and black borders in video stream previews on the desktop terminal.

Optimized the display priority of remote stream user information: remarks > nickname > user ID.

Optimized the TUICallKit component package size (removed unused images and code).

## Version 3.0.6 @2023.09.19

### Fix

Fixed the issue with message display for integration into TUIKit.

## Version 3.0.5 @2023.09.15

## Optimize

Optimized the mutual references between TUICallKit components to avoid stack overflow issues when mini-programs are packaged in uniapp.

## Add

Added a prompt for desktop devices without permissions, guiding customers on how to authorize the device.

## Fix

Fixed [setCallingBell](#). The called ringtone was overwritten by the calling ringtone, causing a ringtone repetition issue.

Fixed styling issues on mobile devices.

# Version 3.0.4 @2023.09.01

## Fix

Fixed the object of [setCallingBell](#) as the incoming call ringtone (called ringtone).

Fixed the [destroyed](#) error reporting issue.

Fixed the issue of no Chinese and English prompts in error pop-up window.

Fixed the issue that the user failed to switch between large and small screens after turning off the camera during a 1v1 call.

# Version 3.0.3 @2023.8.25

## Add

Added [@tencentcloud/call-uikit-vue2.6](#) adaptation for Vue 2.6.

## Optimize

Optimized component default language to match the system's default language.

Optimized printed log information.

Optimized [tuicall-engine-webrtc](#) error message prompts.

Optimized resource cleanup after component termination.

## Fix

Fixed the issue where [videoDisplayMode](#) and [videoResolution](#) did not take effect when a call was made again after it was hung up.

Fixed issues that during calls [statusChanged](#) errors were not thrown.

Fixed the issue where [init](#) was called multiple times.

Fixed the issue of failing to switch between large and small screens after the camera was turned off during a call.

# Version 3.0.2 @2023.8.14

**Fix**

Fixed styling issues with the called component on HTML5 platforms.

Fixed styling issues that occur after switching to a small window during a subsequent call.

## Version 3.0.1 @2023.8.8

**Fix**

Fixed the issue that the caller in a group call could not initiate local preview, and modified the component layer's default reading mode from the data layer.

## Version 3.0.0 @2023.8.4

**Breaking Change**

Upgraded the underlying dependency [tuicall-engine-webrtc](#) to ^2.0.0. It no longer supports [tim-js-sdk](#) to create tim instances. For creating tim instances, use [@tencentcloud/chat](#).

**Add**

Added custom ringtone feature [setCallingBell](#).

## Version 2.4.2 @2023.11.03

**Add**

Added the Invite User feature.

Added the feature of adding attendees by [joinInGroupCall](#).

Added the API [enableMuteMode](#) for muting incoming call ringtones++++.

**Fix**

Fixed the issue of incorrect display of remote stream microphone status.

## Version 2.4.0 @2023.10.20

**Add**

Added the floating window feature.

Added the [enableFloatWindow](#) API for enabling/disabling the floating window feature.

Desktop terminal supports camera and microphone device switching.

Added prompt message for failing to call users on the blocklist.

Added support for Japanese.

## Optimize

In video calls, the large screen displays the remote user by default.

## Version 2.3.9 @2023.10.10

### Add

Added version number, framework, and other information for reporting.

## Version 2.3.8 @2023.10.08

### Add

Added call duration display for desktop terminal video calls.

## Optimize

Optimized the issue with rounded corners and black borders in video stream previews on the desktop terminal.

Optimized the display priority of remote stream user information: remarks > nickname > user ID.

Optimized the TUICallKit component package size (removed unused images and code).

## Version 2.3.6 @2023.09.15

## Optimize

Optimized the mutual references between TUICallKit components to avoid stack overflow issues when mini-programs are packaged in uniapp.

### Add

Added a prompt for desktop devices without permissions, guiding customers on how to authorize the device.

### Fix

Fixed [setCallingBell](#). The called ringtone was overwritten by the calling ringtone, causing a ringtone repetition issue.

Fixed styling issues on mobile devices.

## Version 2.3.5 @2023.9.5

### Fix

Fixed the issue that the camera and microphone were on by default before a user entered the room.

## Version 2.3.4 @2023.9.1

### Add

Added the feature of disabling or enabling the camera before a video call is answered.

### Fix

Fixed the issue of failing to switch between large and small screen after turning off the camera during a 1v1 call.

Fixed the issue that `statusChanged` errors were not thrown when a video call was switched to a voice call.

## Version 2.3.3 @2023.8.22

### Fix

Fixed the issue where `videoDisplayMode` and `videoResolution` did not take effect when a call was made again after it was hung up.

Fixed issues that during calls `statusChanged` errors were not thrown.

## Version 2.3.2 @2023.7.26

### Breaking Change

Removed the floating window `TUICallKitMini` component and merged it into the `TUICallKit` component.

The `@kicked-out` event was adjusted to an callback binding method `:kickedOut` .

The `@status-changed` event was adjusted to an callback binding method `:statusChanged` .

### Add

Added new animation effect when the call page appears.

Added new group call layout on the H5 page.

### Optimize

Optimized problem prompt information and prompt method during calls.

Optimized support and interaction on the H5 page.

Optimized the call API invocation time.

Optimized [@tencentcloud/call-uikit-vue](#) of the package directory structure.

### Fix

Fixed call issues such as immediate disconnection after connection under certain edge operations.

Fixed styling issues on H5 pages in some models and browsers.

Fixed call issues caused by repeated clicks.

## Version 2.2.1 @ 2023.7.7

### Add

[@tencentcloud/call-uikit-vue2](#) Added the detection and prompt for the Vue version.

### Fix

Fixed the issue that clicking the Answer button multiple times on the incoming call page caused the call answering failure.

## Version 2.2.0 @6.30

### Add

[call](#) and [groupCall](#) support the definition of numeric roomID parameters.

[call](#) and [groupCall](#) support the definition of the userData parameter (used to add extended information in the invite signal).

Added the [setSelfInfo](#) API, allowing users to set their nickname and avatar.

## Version 2.1.0 @2023.4.14

### New features

During voice call in H5 mode, the caller's nickname is displayed when calling.

When the call initiation fails, "Call initiation failed" is displayed on the call screen.

When an incoming call is not connected, "Failed to answer" is displayed on the incoming call screen.

Support monitoring whether the current user is kicked out (e.g. being logged out unexpectedly).

Support monitoring TUICallKit call status.

Support business-side code to control the answer, cancel and hangup of calls.

In Vue 2 version added TypeScript type declaration file, which can compile type normally in TypeScript project.

#### Fix

Fixed the warning of updating profile interface appearing in the console when initializing the component.

Fixed the issue that the background image of the callee answer button appears misaligned in H5 mode.

#### API changes

`TUICallKitServer.destroy()` Add call restriction, only support call in uncalled state.

## Version 2.0.1 @2023.03.31

#### Add

Optimized the rendering logic of 1v1 call and multiplayer call video to improve performance and stability.

Optimized UI display, supporting displaying corresponding UI during `TUICallKitServer.call()` execution, i.e. `<TUICallKit/>` UI component can be displayed immediately by clicking the call button.

#### Fix

Fix the issue that nicknames are displayed incorrectly in multiplayer calls.

Fix the issue that CSS does not limit the effective range and pollutes the global style.

## Version 2.0.0 @2023.03.21

#### 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 Vue 2: [call-uikit-vue2](#).

#### Fix

Fixed the issue of call initiation failure due to missing camera device or permission.

## Version 1.4.2 @2023.03.03

#### New features

Support for setting call resolution. See API Document for more details.

Support for modifying the screen display mode. See API Document for more details.

Optimized access steps.

Optimized error throwing.



## Version 1.4.1 @2023.02.13

### New features

Optimized the preview logic of local camera.

Optimized the rendering logic of remote video streams.

## Version 1.4.0 @2023.01.06

### New features

Support Vue 2.7+ project introduction.

Show nickname by default in call API.

## Version 1.3.3 @2022.12.27

### 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, for example, it will not pop 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.

### Fix

Fixed Basic Demo errors caused by outdated vue-CLI dependency.

## Version 1.3.2 @2022.12.07

### New features

Support language switching. See [setLanguage](#) for API details.

Optimized the logic of Basic Demo device detection, it will not pop up again after manually skipping.

### Fix

Fix a warning caused by introducing defineProps.

## Version 1.3.1 @2022.11.29

### 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 monitoring the change of call type by the caller if the call is not answered.

Added support for device testing in the Basic Demo.

### Fix

Fixed an internal logic error that occurs when the user hangs up.

## Version 1.3.0 @2022.11.14

### 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.

### Fix

Fixed the issue that the TIM instance is not entirely terminated after `TUICallKitServer.destroy()` .

Fixed the issue that 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 @2022.11.03

### New features

Adapted to the new version of the TUICallEngine SDK.

## Version 1.1.0 @2022.10.21

### New features

Added support for full screen during a call.

Added support for call window minimizing using `<TUICallKitMini/>` .

**Fix**

Fixed known issues and improved stability.

## Version 1.0.3 @2022.10.14

**New features**

Added support for quick user ID copying and one-click window opening.

## Version 1.0.2 @2022.09.30

**New features**

Added demonstrations and more detailed directions to the integration guide.

**Fix**

Fixed the issue that 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 @2022.09.26

**New features**

Added support for hiding the mic icon of the callee when making a call.

**Fix**

Changed the SDKAppID input restriction in the Basic Demo to numeric.

## Version 1.0.0 @2022.09.23

TUICallKit Basic Demo

[Integration \(TUICallKit\)](#)

[API Documentation \(TUICallKit\)](#)

[UI Customization \(TUICallKit\)](#)

[FAQs \(Web\)](#)

# Android & iOS

Last updated : 2024-08-14 10:37:53

## The new version (1.1.0.103) involves changes to the TUICallEngine interface :

If you encounter Android build errors due to the Maven latest update mechanism during development, you can solve it in two ways :

Upgrade TUICallKit to the latest version.

Modify the tuicallengine dependency of tuicallkit/build.gradle to the fixed version of 1.0:

```
com.tencent.liteav.tuikit:tuicallengine:1.0.0.53 .
```

If you encounter iOS build errors due to executing pod update during development, you can solve it in two ways: :

Upgrade TUICallKit to the latest version.

Add `pod 'TUICallEngine', '1.0.0.53'` to the Podfile.

## Version 2.5.0.1025 @ 2024.8.7

### Feature optimization

Android: Optimize the logic of joinInGroupCall to fix memory leaks.

### Bugs Fixed

Android & iOS: Fix the issue where web users do not receive group call invitations sent from Android and iOS.

Android: Fix the issue where during a group call, A voice calls B, and when B clicks the push notification to open the interface, it shows Speaker instead of Earpiece.

## Version 2.4.0.970 Released June 15, 2024

### Feature Optimization

Android & iOS: Show tips in weak network conditions.

Android: Optimize the incoming call strategy when the callee's screen is locked.

iOS: Fix the issue of abnormal memory growth in group calls.

### Bugs Fixed

Android & iOS: Fix the display issue when the joinInGroupCall interface is invoked.

## Version 2.3.0.915 Released on April 15, 2024

## Feature Optimization

Android and iOS: Added support for displaying the call status at the top of the TUIChat group, and allowing group members to join the chat actively.

Android and iOS: Optimized the incoming call pop-up logic. The answer box is displayed in landscape mode by default.

Android and iOS: Added support for blurring the video call background.

## Bugs Fixed

Android: Fixed the issue of no response when the Delete button on the call record editing interface is clicked.

iOS: Fixed the issue of ghosting during the switching process when you click the member view in a group call.

iOS: Fixed the issue of not displaying the audio and video interface in specific scenarios.

Android and OS: Fixed the issue of missing prompts after the call ends when calling a user on a busy call.

# Version 2.2.0.860 Released on February 1, 2024

## Feature Optimization

Android and iOS: Optimized the UI visual effects, feature clarity, and user experience.

## Bugs Fixed

Android and iOS: Fixed the issue of microphone and camera device occupation after answering an incoming call in the process of a conference or live broadcast.

# Version 2.1.0.810 Released on December 19, 2023

## Feature Optimization

Android: Optimized the prompt for exceptions when calling the TUICallKit API without logging in.

Android: Optimized compatibility for the Android 14 platform (API 34). For details, refer to [Android 14 behavior changes](#).

Android and iOS: Optimized the display of user nicknames, displayed in the following order: user remarks > user nickname > user ID. User ID is displayed by default.

## Bugs Fixed

iOS: Fixed the issue of overlapping group call avatars.

iOS: Fixed the issued that the keyboard cannot be retracted when you return to the call interface after enabling the floating window to send messages during a video call.

iOS: Fixed the issue that the camera cannot be switched and moved when you switch the camera, turn off the camera, and then switch back to the original camera and reopen it during a video call.

Android: Fixed the issue that the small window of a single video call cannot be moved under the right-to-left layout mode (RTL mode) in languages such as Arabic.

## Version 2.0.0.750 Released on November 3, 2023

### Feature Optimization

Android and iOS: Added support for Japanese by UIKit.

Android and iOS: Optimized the display of call nicknames.

Android and iOS: Adjusted the default ringtone volume from 60% to 100%.

### Bugs Fixed

iOS: Fixed the issue of slow image loading in the Swift version.

iOS: Fixed the issue that the call invitation is automatically canceled after the caller initiating the call moves the application to the background.

## Version 1.9.0.680 Released on September 27, 2023

### Feature Optimization

Android and iOS: Added support for the Arabic language.

Android and iOS: Optimized the package purchase prompts. You can click the link to jump to the corresponding package purchase page.

Android and iOS: Optimized the default bitrate at different resolutions to ensure clearer images at higher resolutions.

Android and iOS: Adjusted the default bitrate for video calls to 600 kbps, with a beauty level of grade 4.

### Bugs Fixed

Android and iOS: Resolved inconsistencies between the rejection prompt when initiating a call to a user on the blocklist and the rejection prompt when sending a private chat message.

iOS: Rectified an anomaly in the video placement for the initiator, which occurred on a 4-user group video call interface when one member declined the call.

iOS: Fixed the issue that the resolution is reset if the beauty feature is enabled immediately after successful login.

## Version 1.8.0.620 Released on August 14, 2023

## Feature Optimization

Android and iOS: Added the feature that call messages are excluded from the unread count by default.

Android: Optimized the redirection page for floating window permissions on Xiaomi smartphones.

## Bugs Fixed

iOS: Fixed the issue that the onKickOffline callback API becomes ineffective after being kicked offline.

iOS: Fixed the issue that, the list is empty after you clear the calls on the missed call interface and then return to the all calls interface.

## Version 1.7.2.570 Released July 20, 2023

### Functionality Enhancement

Android: Gravity sensor is turned off by default, optimizing the call experience on large-screen and customized devices.

### Defect Rectification

Android & iOS: Rectified an issue where, after User A (online) calls User B (offline) and cancels the call, User A calls back User B who logs in thereafter, leading to abnormal cloud call records for user B.

Android: Resolved the crash issue of TUICallKit after upgrading the TRTC SDK version to 11.3.

## Version 1.7.0.460 Released June 25, 2023

### Functionality Enhancement

Android & iOS: Includes UI integration solutions, optimizes sample projects, and improves call setting items.

Android: Reduced the status preservation level during a call to only show standby prompts in the status bar; removed notifications and vibrations.

## 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 : 2024-04-30 16:11:13

## Version 2.3.2 @2024.04.24

### New Features :

Android & iOS: Optimized popup logic for incoming calls, displaying the banner answering box by default.

Android & iOS: Supported background blur for video calls..

## Version 2.3.1 @2024.04.22

### Bug Fixes :

Fixed the issue of the Method Channel error reporting on non-iOS & Android platforms.

## Version 2.3.0 @2024.04.18

### New Features :

Android & iOS: Supported displaying call status at the top of the group in `tencent_cloud_chat_uikit` and allowing group members to join a call.

### Feature Optimization :

Android & iOS: Optimized the login method when used concurrently with `tencent_cloud_chat_uikit`.

### Dependency Description :

Upgraded `tencent_cloud_uikit_core` to version 1.6.0.

Upgraded the dependency client `TUICallEngine SDK` to version 2.3.0.915.

## Version 2.2.3 @2024.03.08

### Dependency Description :

Upgraded `tencent_cloud_uikit_core` to version 1.5.2.

## Version 2.2.2 @2024.03.07

### New Features :

Android&iOS: TUICallObserver's onCallReceived callback adds user-defined parameter userData.

## Version 2.2.1 @2024.02.06

### Bug Fixes :

Android: After killing the process, the application received the FCM push but did not answer it, and then entered the application page with an exception.

## Version 2.2.0 @2024.02.03

### New Features :

Android: Supports FCM push and needs to be used together with tencent\_cloud\_chat\_push

## Version 2.1.1 @2024.01.06

### Bug Fixes :

Android&iOS : Modify some UI 3.0 UI related bugs

## Version 2.1.0 @2024.01.04

### New Features :

Android&iOS : Use the new UI3.0.

## Version 2.0.6 @2023.12.15

### Bug Fixes :

iOS : The problem of missing some fields in iOS offline push information

## Version 2.0.5 @2023.12.10

### Bug Fixes :

iOS : Fixed the issue of abnormal ringtone when entering the call page during VoIP push and abnormal pulling of remote stream.

Android : Fixed an issue where the incoming call page would be abnormal when in the background when the floating window permission was not obtained.

## Version 2.0.4 @2023.12.04

### Bug Fixes :

Android&iOS : Fixed the intermittent issue of incoming call page not displaying sometimes after clicking on an incoming call notification.

## Version 2.0.2 @2023.11.27

### Bug Fixes :

Android : Fix the issue of call failure under multiple Flutter Engine conditions.

## Version 2.0.1 @2023.11.15

### Bug Fixes :

Android & iOS : Fixed an incompatibility issue caused by Tencent RTC Observer when using Tencent RTC components with other components that also use Tencent RTC.

Android : Fixed an incompatibility issue when using TUIRoom in conjunction with other devices.

## Version 2.0.0 @2023.11.06

### Dependency Description :

Upgrade the dependent client 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: Support Voip.

### Bug Fixes :

iOS: Fixed an issue where the call page would be abnormal when receiving a call in the background.

## Version 1.9.0 @2023.10.09

### New Features :

Android&iOS: Add an interface for setting ringtones.

### Feature Optimization :

Android & iOS: Optimize package purchasing prompts.

Android & iOS: Optimize default bitrates for different resolutions, [see details](#).

### Bug Fixes :

iOS: Fixed the issue where the same Observer object can be registered twice.

### Dependency Description :

Upgrade the dependent client 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 problem of no call message display when using tencent\_cloud\_chat\_uikit.

Android&iOS: Fixed the problem of occasionally pulling up the group call page during a single-person call.

Android&iOS: Fixed the problem of occasionally pulling up the call page twice during a call.

Android&iOS: Fixed the problem of abnormal display of call duration.

Feature 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 problem of some compilers failing to compile 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 of the other party being displayed during a group call 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

### Feature 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



**Feature Optimization:**

Android: Supports development & debugging using the emulator.

**Dependency Description :**

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.

**Feature 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 Description :**

Upgrade the dependent client 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.

**Feature 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 dependent client 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.

**Feature 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

**Feature 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

**Feature 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.