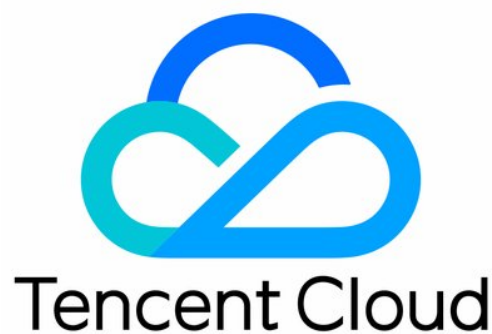


# **Tencent Real-Time Communication Multi-Participant Conference (with UI) Product Documentation**



## 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

## Multi-Participant Conference (with UI)

- Overview (TUIRoomKit)

- Activate the Service (TUIRoomKit)

- Run Demo (TUIRoomKit)

  - Web

  - iOS

  - Android

  - Electron

  - Flutter

- Integration (TUIRoomKit)

  - iOS

  - Android

  - Electron

  - Web

  - Flutter

- Schedule a meeting (TUIRoomKit)

  - Android&iOS&Flutter

  - Web&Electron

- UI Customization (TUIRoomKit)

  - Electron

  - Android

  - iOS

  - Web

  - Flutter

- Virtual Background (TUIRoomKit)

  - Web

- Conference Control (TUIRoomKit)

  - Android&iOS&Flutter

  - Web&Electron

  - H5

- Cloud Recording (TUIRoomKit)

- AI Noise Reduction (TUIRoomKit)

- In-Conference Chat (TUIRoomKit)

  - Web&Electron

  - Android&iOS

Flutter

Robot Streaming (TUIRoomKit)

Enhanced Features (TUIRoomKit)

Floating Window

Text Watermark

Setting Nickname, Avatar (All Platform)

Monitor Conference Status

Client APIs (TUIRoomKit)

iOS&Mac

RoomKit API

RoomEngine API

API Overview

TUIRoomEngine

TUIRoomObserver

Type Definition

Android

RoomKit API

RoomEngine API

API Overview

TUIRoomEngine

TUIRoomObserver

Type Definition

Web

RoomKit API

RoomEngine API

API Overview

TUIRoomEngine

TUIRoomEvents

TUIRoomEngine Defines

Electron

RoomKit API

RoomEngine API

API Overview

TUIRoomEngine

TUIRoomEvent

TUIRoomEngine Defines

Flutter

RoomKit API

## RoomEngine API

- API Overview

- TUIRoomObserver

- TUIRoomEngine

- Type Definition

## Server APIs (TUIRoomKit)

### REST API

- RESTful API Overview

- RESTful API List

### Room Management

- Create a Room

- Destroy a Room

- Update the Room Information

- Get the Room Information

### User Management

- Get the Room Member List

- Update the Room Member Information

- Change the Room Ownership

- Mark Room Members

- Ban Room Members

- Unban Room Members

- Get the Banned Room Member List

- Remove Room Member

### Seat Management

- Get the Seat List

- Pick User on the Seat

- Kick User off the Seat

- Lock the Seat

## Third-Party Callback

- Callback Overview

- Callback Command List

### Callback Configuration

- Query Callback Configuration

- Create Callback Configuration

- Update Callback Configuration

- Delete Callback Configuration

## Room Related

- After a Room Is Created

After a Room Is Destroyed

After the Room Information Is Updated

User Related

After a Room Is Entered

After a Room Is Left

Seat Connection Related

After the Seat List Is Changed

FAQs (TUIRoomKit)

Web

iOS

Android

Electron

Flutter

Error Code (TUIRoomKit)

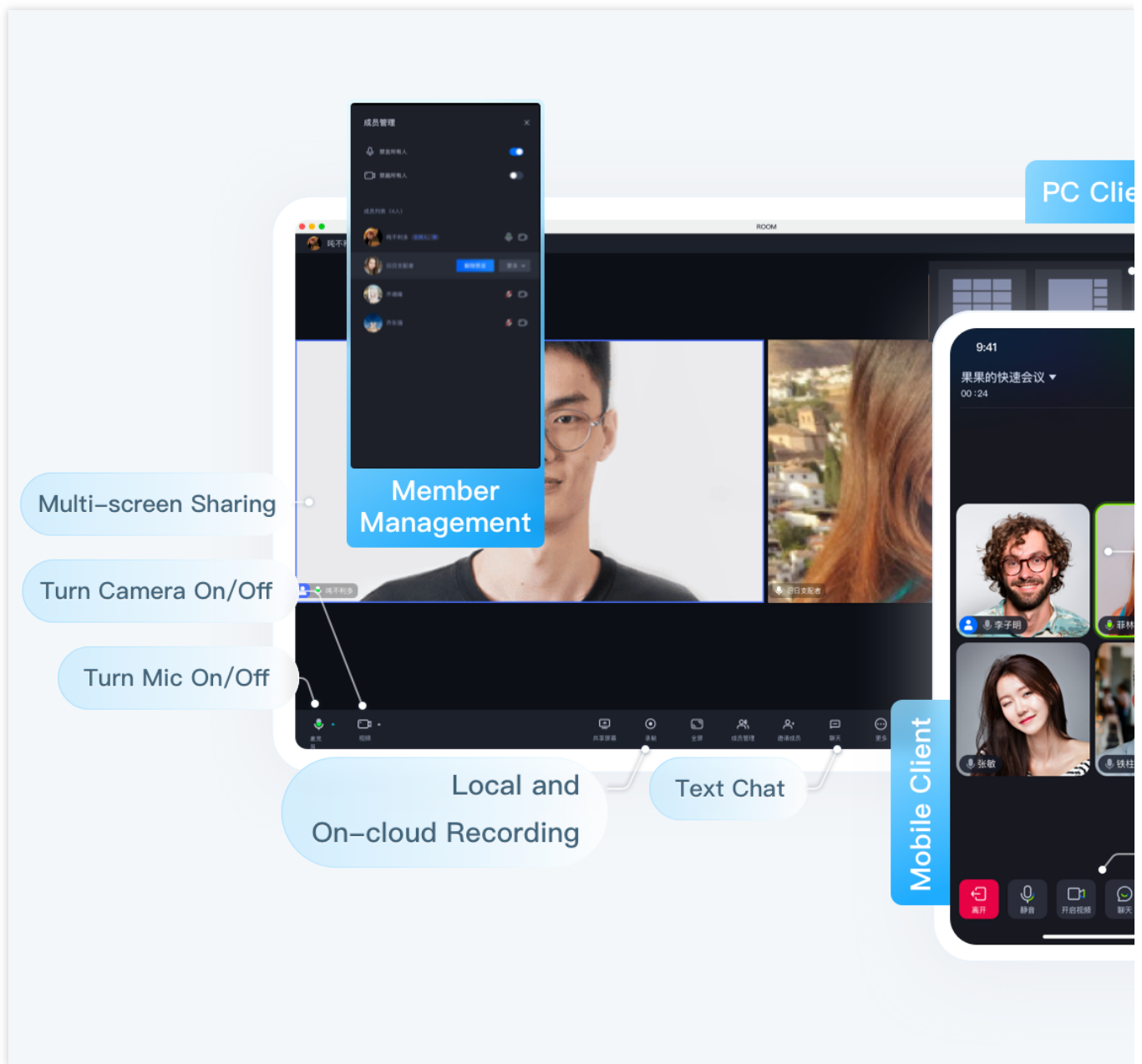
SDK Update Log (TUIRoomKit)

# Multi-Participant Conference (with UI) Overview (TUIRoomKit)

Last updated : 2024-04-23 17:47:02

## Overview

Conference (TUIRoomKit) is a product suitable for multi-person audio and video conversation scenarios such as business meetings, webinars, and online education. By integrating this product, you can add room management, member management, screen sharing, and other functions to your app in just three steps within a day, allowing you to quickly launch your business. The basic features of the component are shown below:

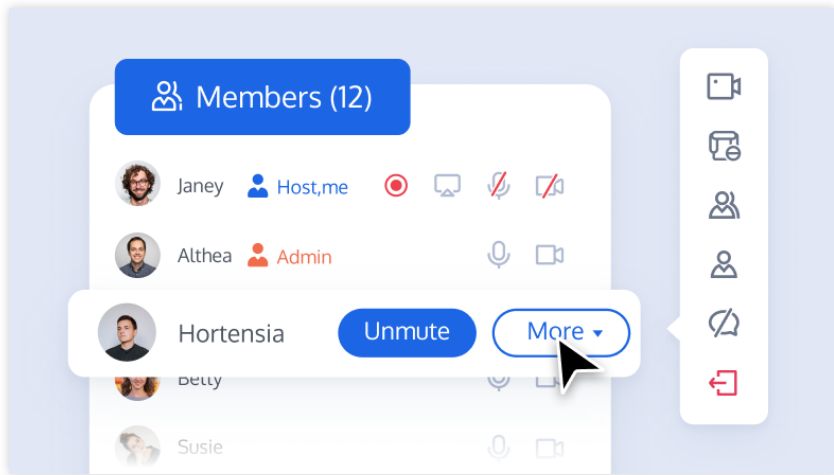
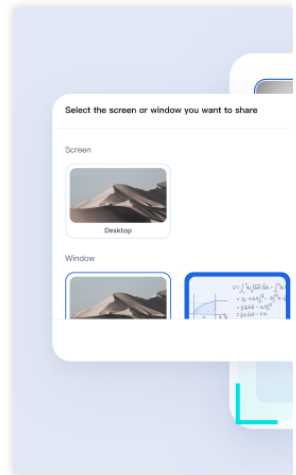


## Supported Platform

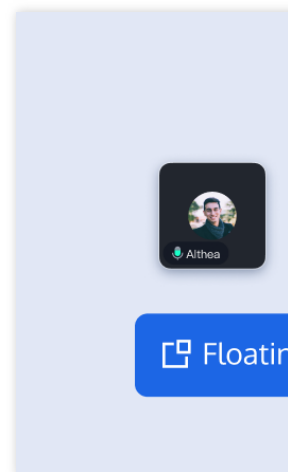
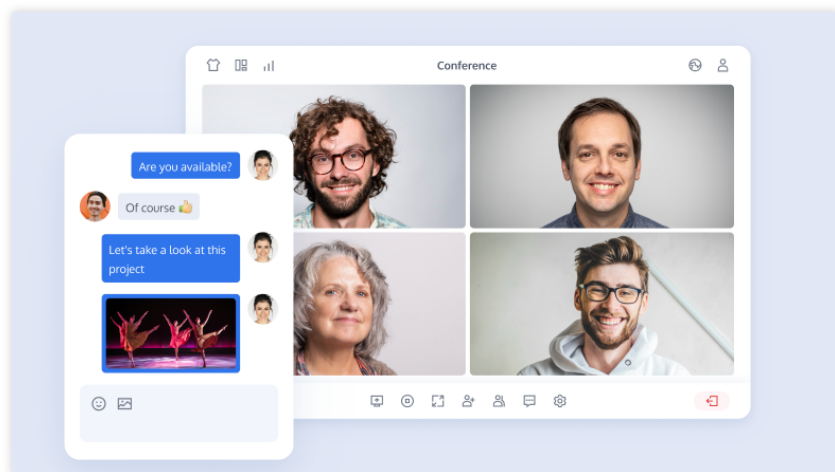
Platform	Android	iOS	Web	Flutter	Electron
Supported					
Supported Languages/Frameworks	Java	Swift	Vue3 Vue2	Dart	Vue3 Vue2

## Description of the Feature

Basic Feature	Advanced Feature	Feature Advantages
Multi-person video conversation Rich meeting control functions Support for on-stage speaking mode Enable/Disable Floating Window Audio and video settings Screen sharing	In-meeting chat On-cloud recording AI Noise Reduction	<b>Quick integration</b> <b>Comprehensive UI Interaction</b> <b>Cross-platform Interconnection Support</b> <b>Multi-device Login Support</b>

Member Management	Screen Sharing
	

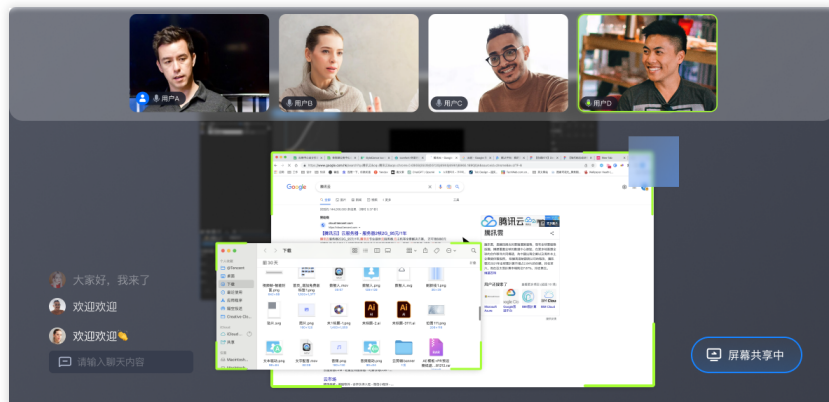
In-meeting Chat	Floating Window



## Applicable Scenario

Multi-person audio and video conversation interaction, covering business meetings, webinars, online education, online recruitment, and audio chat rooms.

### Audio/Video Conference



### Online Classroom



### Webinar

### Corporate Training





## Trying It Online

Platform	Web	Android	iOS	Others
Online Experience				/
Demo Integration	<a href="#">Github: web</a>	<a href="#">Github: Andorid</a>	<a href="#">Github: iOS</a>	<a href="#">Github</a>

## Exchange and Feedback

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

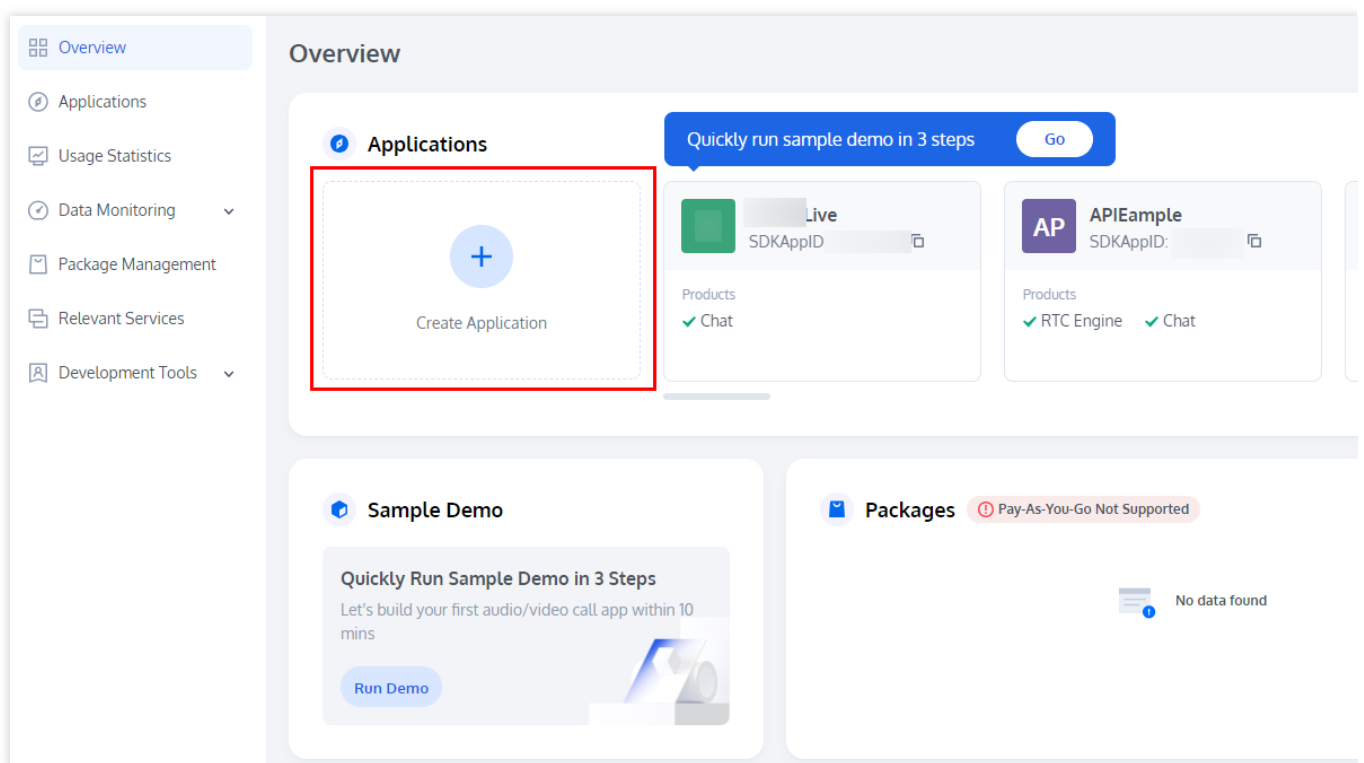
# Activate the Service (TUIRoomKit)

Last updated : 2024-07-03 17:01:13

## Free trial

You can follow the steps below to activate the TRTC Conference product service and receive a free 14-day trial version.

1. Log in to the [TRTC Console](#), click **Create Application**.



2. In the Create Pop-up Window, select **Conference** and enter the application name, select the Data Storage **Region**, and click **Create**.

### Note:

The default Data Storage for real-time audio and video service data is in Singapore, and the storage for instant messaging service data is in your selected data center.

### Create application

Application name  
The application name can contain only digits, letters, and underscores

Select product


☐ Call

☒ **Conference**

☐ Live

☐ RTC Engine

☐ Chat



Version

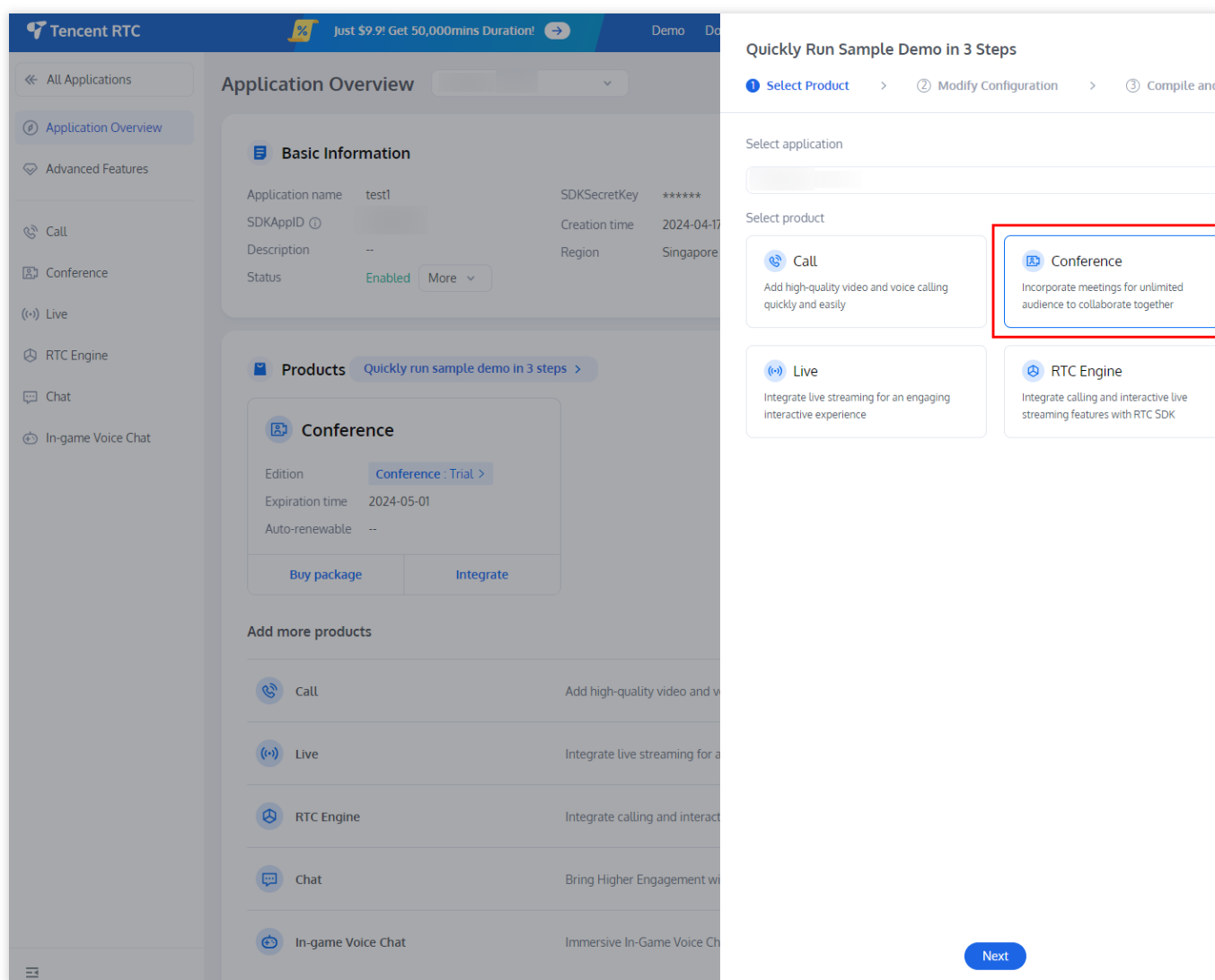
Free Trial **14 Days** Free for 10,000 minutes every month [Version Details](#)

Region

Create

3. After completing the application creation, you will automatically enter the application details page, at which point you have quickly created an application and successfully received a free multi-person audio and video (TUIRoomKit) trial version. You can follow the three steps in the pop-up window on the right to quickly get the sample Demo running. The specific steps are as follows:

3.1 Select the product you need to quickly get running, **Conference**, and then click **Next**.



The screenshot displays the Tencent RTC console interface. On the left, a sidebar lists navigation options: All Applications, Application Overview, Advanced Features, Call, Conference, Live, RTC Engine, Chat, and In-game Voice Chat. The main area is titled 'Application Overview' and shows basic information for an application named 'test1', including SDKAppID, SDKSecretKey, Creation time (2024-04-17), Region (Singapore), and Status (Enabled). Below this, a 'Products' section highlights the 'Conference' product with a 'Quickly run sample demo in 3 steps' button. The 'Conference' product details show an edition of 'Conference : Trial >', an expiration time of '2024-05-01', and an 'Auto-renewable' status. A 'Buy package' button is visible. On the right, a 'Quickly Run Sample Demo in 3 Steps' wizard is shown, with the first step 'Select Product' highlighted. The 'Select product' section lists four options: Call, Conference (highlighted with a red box), Live, and RTC Engine. Each option has a brief description. A 'Next' button is located at the bottom right of the wizard.

**Quickly Run Sample Demo in 3 Steps**

1 Select Product > 2 Modify Configuration > 3 Compile and

Select application

Select product

- Call**  
Add high-quality video and voice calling quickly and easily
- Conference**  
Incorporate meetings for unlimited audience to collaborate together
- Live**  
Integrate live streaming for an engaging interactive experience
- RTC Engine**  
Integrate calling and interactive live streaming features with RTC SDK

**Next**

**3.2 Integration Guide:** Select the corresponding platform according to the application, modify the configuration following the steps. After completing the changes, click **Next**.

### Quickly Run Sample Demo in 3 Steps

✓ Select Product

>

2 Modify Configuration

>

3 Compile and Run

#### Basic Information

Application name

test1

SDKAppID

SDKSecretKey

Modify configuration

[Integration Document](#)

Web

Electron

Flutter

Android

iOS

Windows

Step1

Download sample code

[Download Zip](#)

Step2

Unzip the source code package reported in step 1, find and open the Web/vue2/src/config/basic-info-config.js file.

Step3

Paste the SDKAppID and SDKSecretKey above to the location indicated in the following picture.

```
/*
 * @Description: Basic information configuration for TCBRealKit applications
 */
import LibGenerateTestUserSig from './Lib-generate-test-user-sig.js'

/*
 * Tencent Cloud SDKAppID, which should be replaced with user's SDKAppID.
 * Enter Tencent Cloud TRTC (Console) (https://console.cloud.tencent.com/trtc) to create an application,
 * and you will see the SDKAppID.
 * It is a unique identifier used by Tencent Cloud to identify users.
 */
export const SDKAPPID = '0'; // Paste SDKAppID here

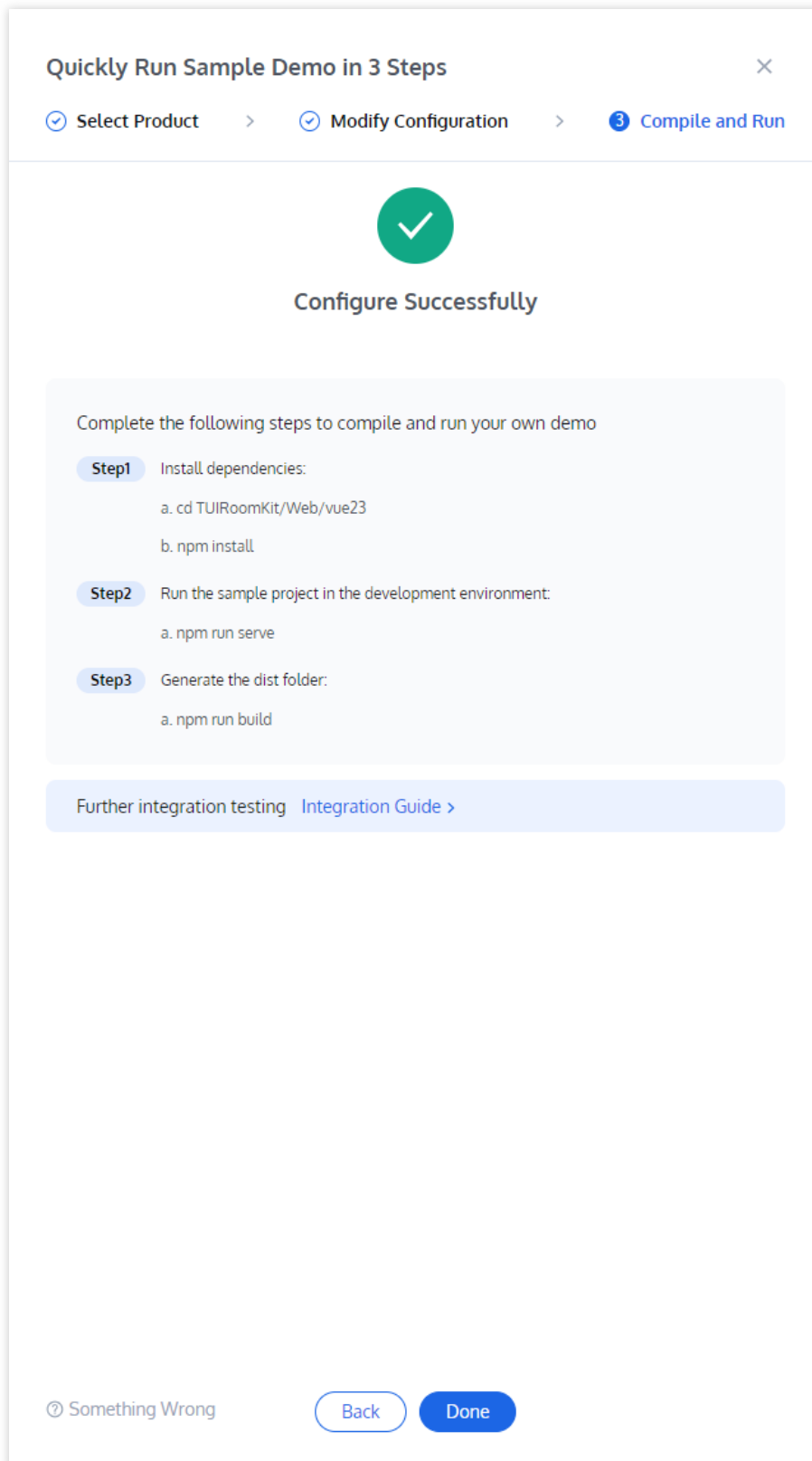
/*
 * Encryption key for calculating signature, which can be obtained in the following steps:
 *
 * 1. Step1: Enter Tencent Cloud TRTC (Console) (https://console.cloud.tencent.com/trtc)
 * and create an application (if you don't have one).
 * 2. Step2: Click your application to find "Quick Start".
 * 3. Step3: Click "View Secret Key" to see the encryption key for calculating UserSig,
 * and copy it to the following variable.
 *
 * Notes: This method is only applicable for debugging items. Before official launch,
 * please migrate the UserSig calculation code and key to your backend server to avoid
 * unauthorized traffic use caused by the leakage of encryption key.
 * Document: https://intl.cloud.tencent.com/document/product/347/33186#server
 */
export const SECRETKEY = '0'; // Paste SECRETKEY here

/*
 * Signature expiration time, which should not be too short
 * Time unit: second
 * Default time: 7 * 24 * 60 * 60 = 604800 = 7days
 */
export const EXPIRETIME = 604800;
```

Back

Next

3.3 At this point, you have successfully configured it. You can follow the prompt steps to start compiling and running your own sample Demo. Click **Done** to complete the configuration. For further testing, click on **Integration Guide** to go to the Conference configuration details page.



4. After activation, you can peek at the information on the current page and refer to the [Integration Guide](#) for integration. The `SDKAppID` and `SDKSecretKey` here will be used in subsequent steps.

### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

[More Features](#)

On-cloud recording	Disabled
Relay to CDN	Disabled
Callbacks	Disabled
Advanced permission control	Disabled

#### Products

[Quickly run sample demo in 3 steps >](#)

##### Conference

Edition	<a href="#">Conference : Trial &gt;</a>
Expiration time	2024-05-01
Auto-renewable	--

[Buy package](#)[Integrate](#)

## Purchase the official editions

The utilization of TUIRoomKit necessitates the purchase of a Conference monthly subscription package, the pricing and feature comparison of which can be found in the [Conference Monthly Package Billing Explanation](#).

1. Visit the [TRTC Conference Purchase page](#), select the Application (SDKAppID) and Conference Package you want to purchase. At the same time, **we recommend you to enable Auto-renewal to avoid affecting business use**. After Confirming purchase information and agreeing to the relevant agreement, check the agreement content and click **Subscribe now**.

## Conference Monthly Packages

Application (SDKAppID) ⓘ



[+ Create Application](#)

ⓘ Please select the correct SDKAppID, as it cannot be modified after purchase.

Package editions [Detail](#)

### Lite

- 20 participants per room
- 20 Active Rooms
- Minimize features
- 100,000 FREE Minutes Included

**\$299/mo.**

### Standard

- 50 participants per room
- 100 Active Rooms
- Advanced Features
- 300,000 FREE Minutes Included

**\$599/mo.**

**Most Popular**

### Pro

- 200 participants per room
- 200 Active Rooms
- All features included
- 450,000 FREE Minutes Included

**\$899/mo.**



**Automatic renewal**

Automatically renew monthly after expiration. You can cancel at any time, please feel free to check it.

☐ Data Monitoring [Detail](#)

Available for all application (SDKAppID). Providing comprehensive quality troubleshooting and real-time Quality Monitoring services, assisting you in quickly understanding the status of your application.

☒ I have read and agree to [TRTC Service Level Agreement](#)

2. Go to the order confirmation page to confirm product information.



## Please confirm the following product information [Go Back to Modify Configuration](#)

### Product List

sp\_rav\_conference

899.00 USD

Monthly package: TRTC Conference Pro

Unit Price: 899.00USD/month

Quantity: 1

Payment Mode: Prepaid

Term: 1month

### Discounts and Vouchers

☐ Use promo voucher

No available Promo voucher

Che

sp\_rav\_

Upfront

Tax:

Total

3. Go to the payment page to complete the payment. After the purchase is completed, you can go to the [TRTC Console](#) to view application edition information, and refer to the [Integration guide](#) for integration.

## Renew the official edition

Refer to the [Purchasing the official edition](#) and purchase the same edition of the Conference monthly package again to complete the renewal. It is recommended that you renew Conference by enabling auto-renewal. **When the account balance is sufficient, it will automatically renew monthly after expiration.**

## Conference Monthly Packages

Application (SDKAppID) ⓘ



[+ Create Application](#)

ⓘ Please select the correct SDKAppID, as it cannot be modified after purchase.

Package editions [Detail](#)

### Lite

- 20 participants per room
- 20 Active Rooms
- Minimize features
- 100,000 FREE Minutes Included

**\$299/mo.**

### Standard

- 50 participants per room
- 50 Active Rooms
- Advanced Features
- 300,000 FREE Minutes Included

**\$599/mo.**

**Most Popular**

### Pro

- 200 participants per room
- 200 Active Rooms
- All Features Included
- 400,000 FREE Minutes Included

**\$899/mo.**



**Automatic renewal**

Automatically renew monthly after expiration. You can cancel at any time, please feel free to check it.

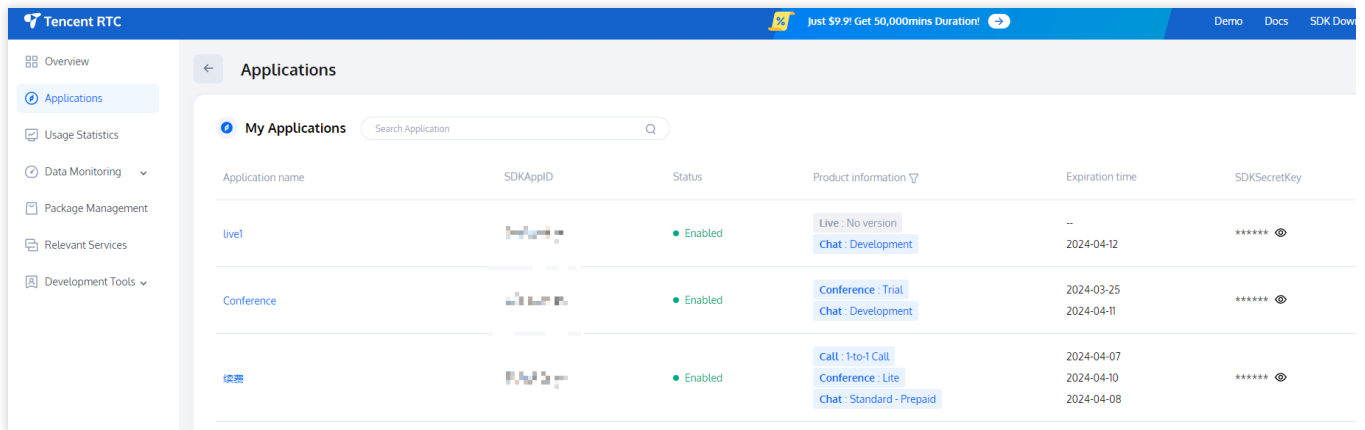
☐ Data Monitoring [Detail](#)

Available for all application (SDKAppID). Providing comprehensive quality troubleshooting and real-time Quality Monitoring services, assisting you in quickly understanding the quality of your application.

☒ I have read and agree to [TRTC Service Level Agreement](#)

The specific steps to enable auto-renewal in the console are as follows:

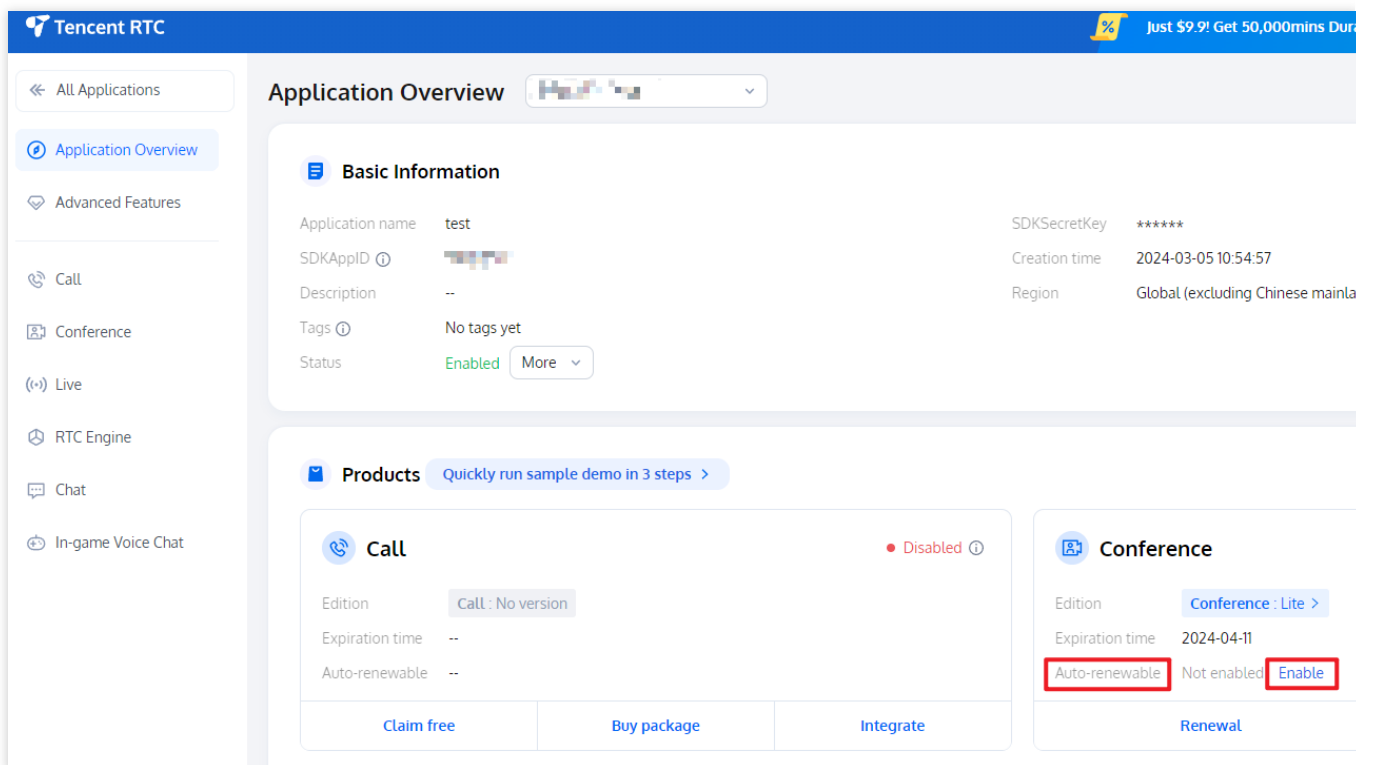
1. Access the [TRTC Console > Applications](#), find the application and click **manage** to enter the application details page.



The screenshot shows the 'Applications' page in the Tencent RTC console. It features a sidebar with navigation options like Overview, Applications, Usage Statistics, Data Monitoring, Package Management, Relevant Services, and Development Tools. The main content area is titled 'My Applications' and contains a table with columns for Application name, SDKAppID, Status, Product information, Expiration time, and SDKSecretKey. Three applications are listed: 'live1', 'Conference', and '语音' (Voice). The 'Conference' application is highlighted, showing its product information as 'Conference : Trial' and 'Chat : Development'.

Application name	SDKAppID	Status	Product information	Expiration time	SDKSecretKey
live1	[redacted]	Enabled	Live : No version Chat : Development	2024-04-12	*****
Conference	[redacted]	Enabled	Conference : Trial Chat : Development	2024-03-25 2024-04-11	*****
语音	[redacted]	Enabled	Call : 1-to-1 Call Conference : Lite Chat : Standard - Prepaid	2024-04-07 2024-04-10 2024-04-08	*****

2. In the Conference product information, click the **Enable auto-renewable** button, a confirmation pop-up will appear, click **Enable**.



The screenshot shows the 'Application Overview' page for an application named 'test'. The page is divided into two main sections: 'Basic Information' and 'Products'. The 'Basic Information' section displays details such as Application name, SDKAppID, Description, Tags, Status, SDKSecretKey, Creation time, and Region. The 'Products' section shows two product cards: 'Call' and 'Conference'. The 'Call' product is currently 'Disabled'. The 'Conference' product is 'Not enabled' and has an 'Auto-renewable' button highlighted with a red box, which is labeled 'Enable'.

**Application Overview**

**Basic Information**

Application name	test	SDKSecretKey	*****
SDKAppID	[redacted]	Creation time	2024-03-05 10:54:57
Description	--	Region	Global (excluding Chinese mainla
Tags	No tags yet		
Status	Enabled		

**Products** Quickly run sample demo in 3 steps >

**Call** Disabled

Edition: Call : No version

Expiration time: --

Auto-renewable: --

[Claim free](#) [Buy package](#) [Integrate](#)

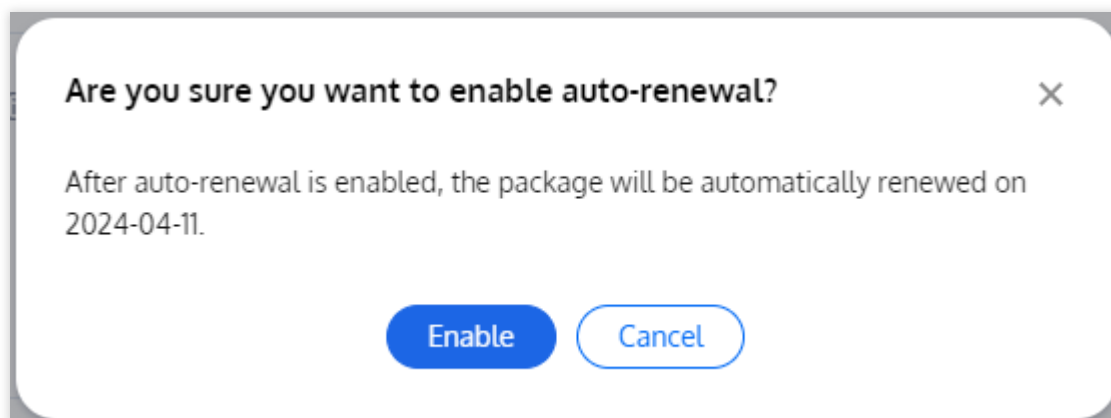
**Conference**

Edition: Conference : Lite >

Expiration time: 2024-04-11

Auto-renewable: Not enabled **Enable**

[Renewal](#)

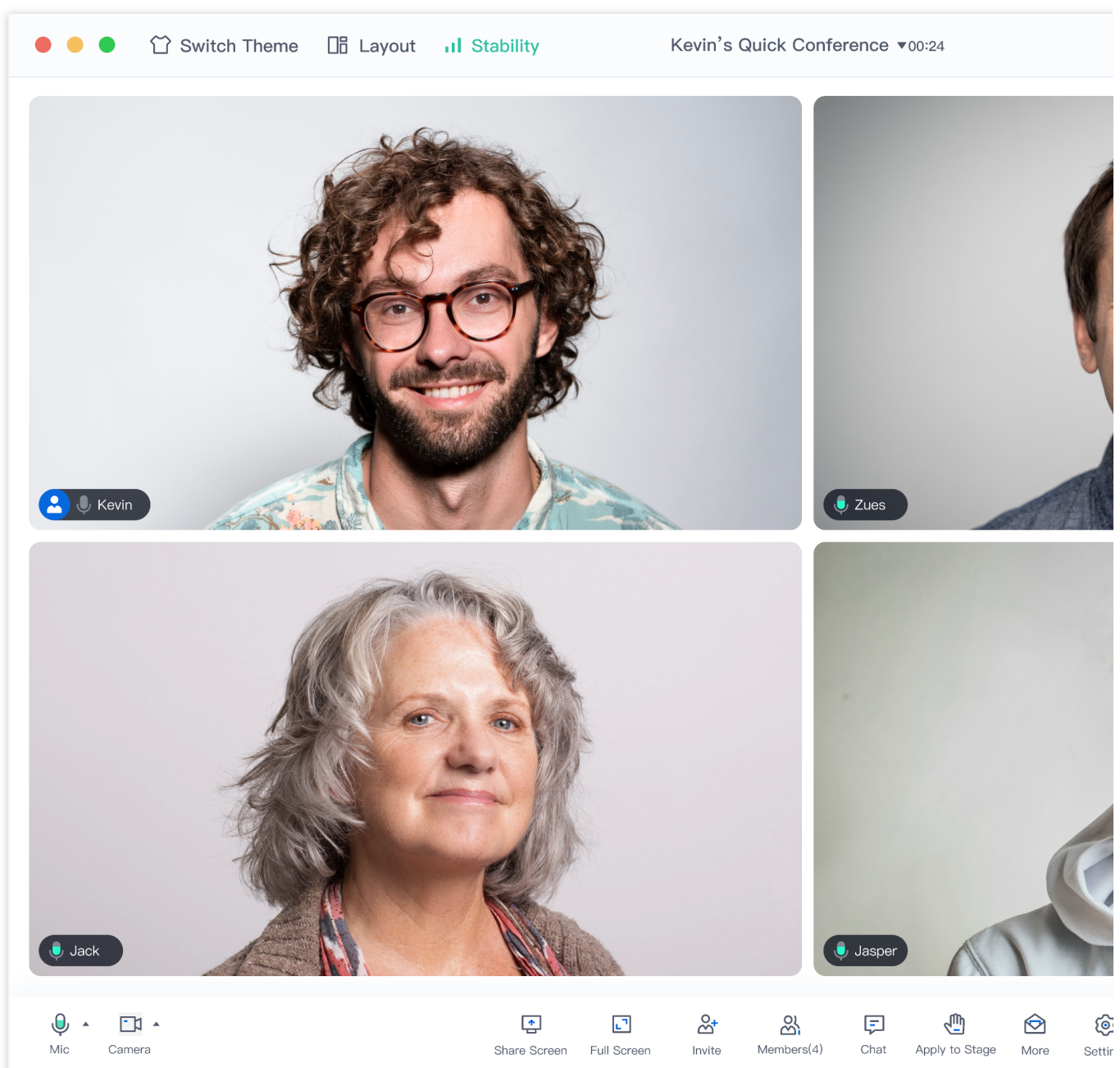


# Run Demo (TUIRoomKit)

## Web

Last updated : 2024-08-15 17:06:16

This document mainly introduces how to quickly run through the **Conference (TUIRoomKit) sample project** and experience a high-quality multi-person video conference. By following this document, you can run through the demo within 10 minutes and ultimately experience a multi-person video conference feature with a complete UI interface.



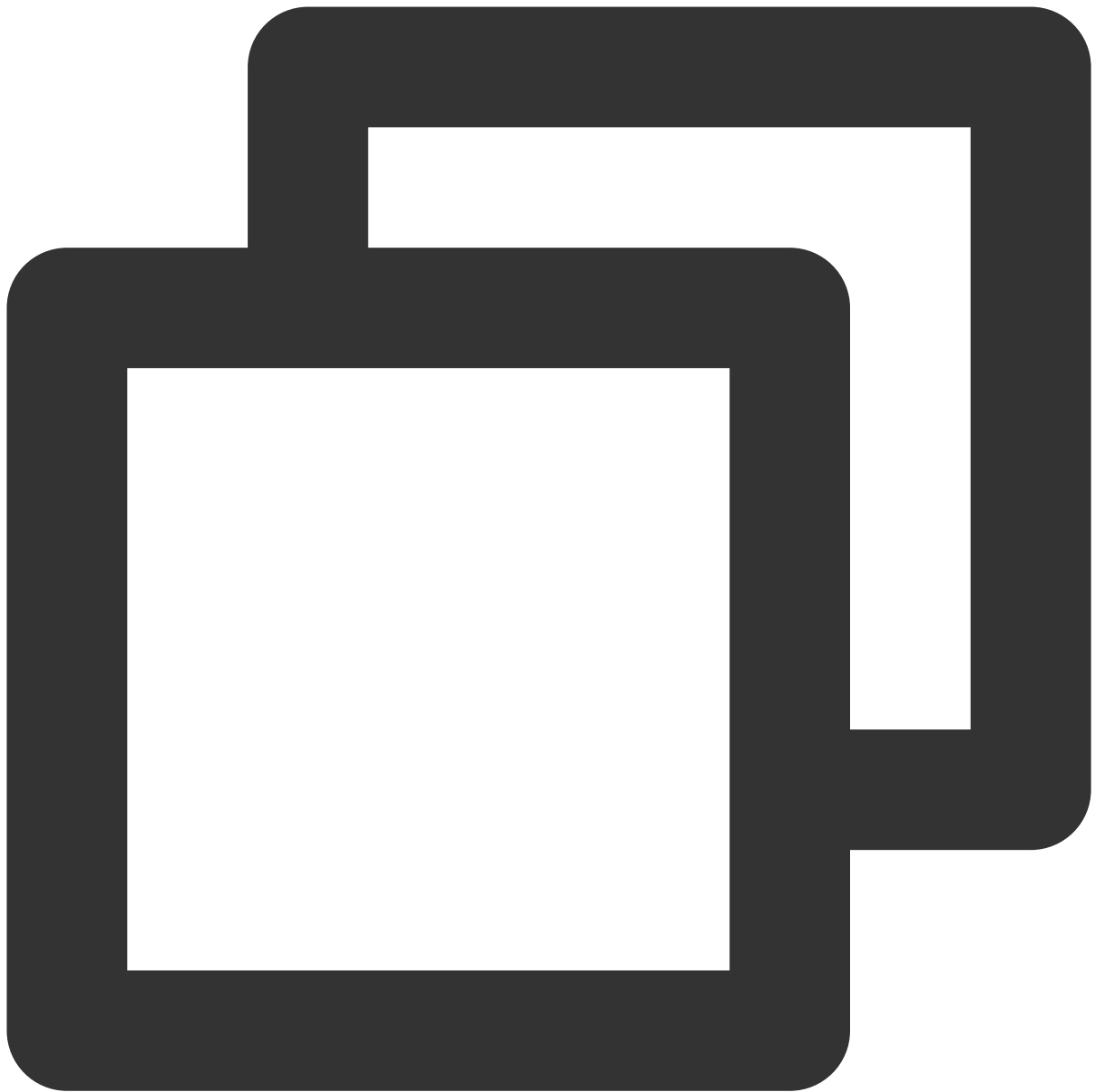
## Prerequisites

Node.js version: Node.js  $\geq 16.19.1$  (we recommend using the official LTS version, please match the npm version with the node version).

Modern browser, [supporting WebRTC APIs](#).

## Download Demo

1. Open the Terminal, copy and paste the sample command to clone the repository.

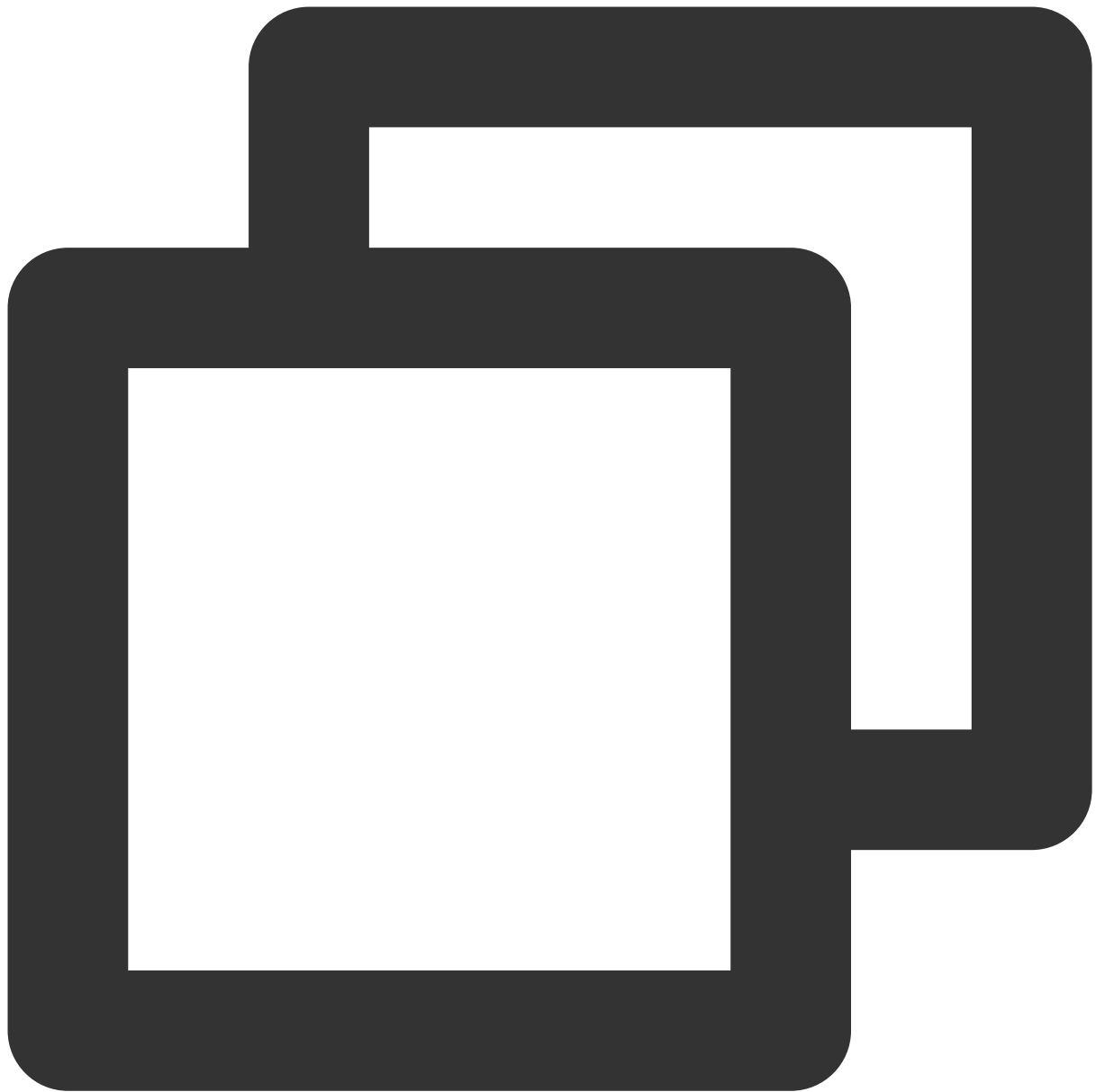


```
git clone https://github.com/Tencent-RTC/TUIRoomKit.git
```

2. Install dependencies.

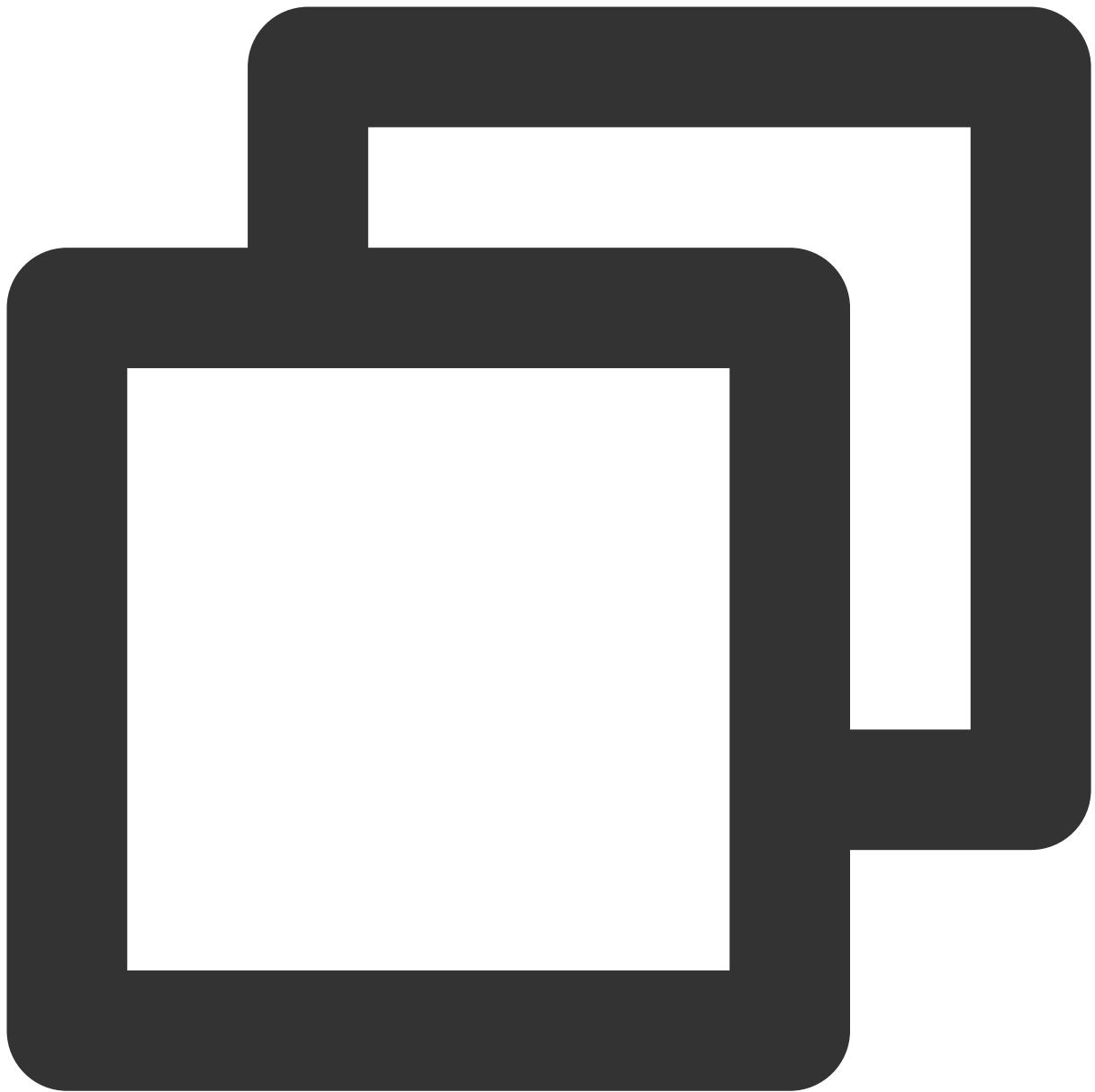
Vue3

Vue2

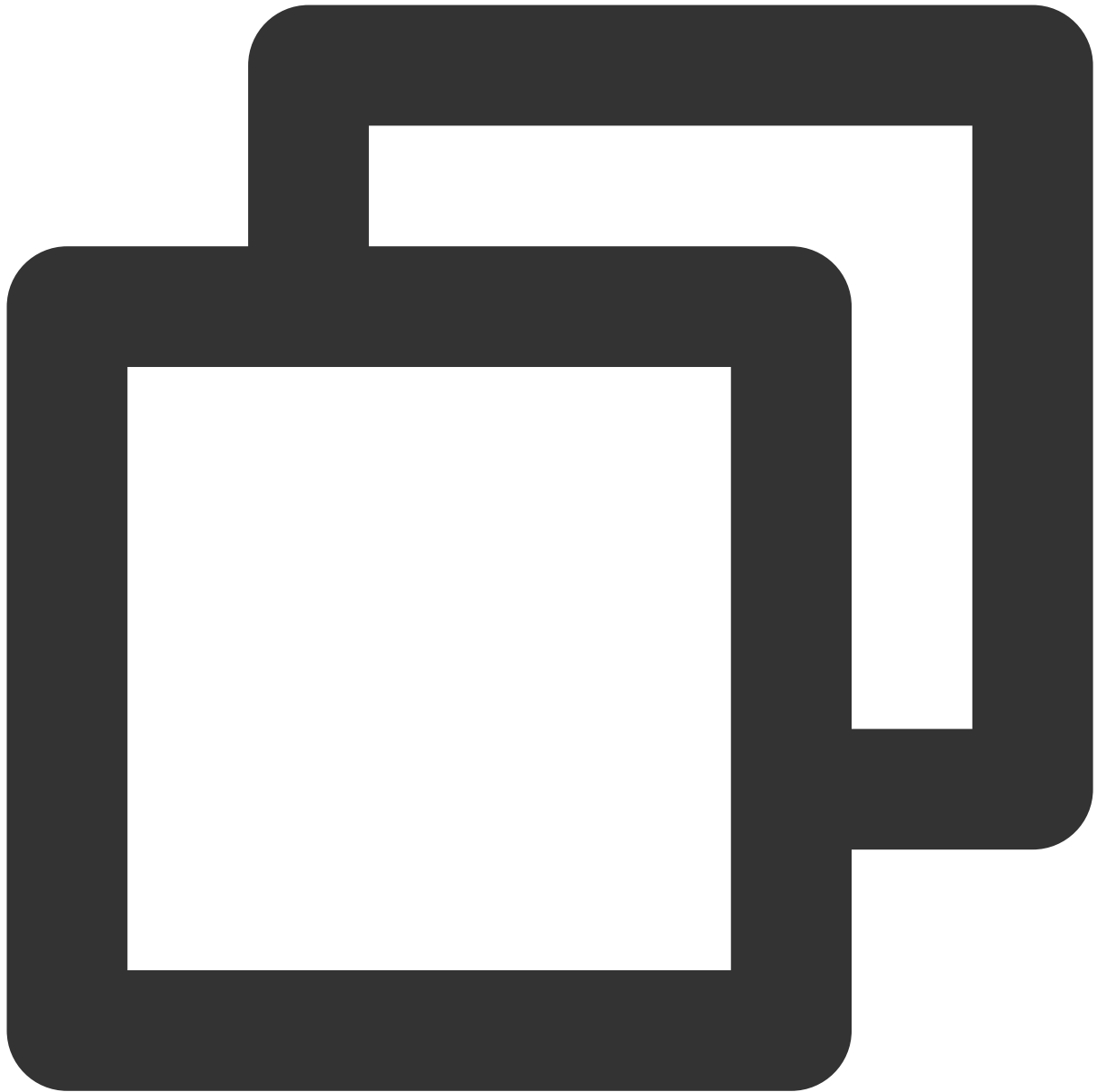


```
cd ../TUIRoomKit/Web/example/vite-vue3-ts
```





```
cd ./TUIRoomKit/Web/example/webpack-vue2.7-ts
```



```
npm install
```

## Configure Demo

1. [Activate the TUIRoomKit service](#), get the **SDKAppID** and **SDKSecretKey**.

### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

On-cloud recording	Dis:
Relay to CDN	Dis:
Callbacks	Dis:
Advanced permission control	Dis:

#### Products

[Quickly run sample demo in 3 steps >](#)

##### Conference

Edition	<a href="#">Conference : Trial &gt;</a>
Expiration time	2024-05-01
Auto-renewable	--

[Buy package](#)[Integrate](#)

2. Open the `TUIRoomKit/Web/example/vite-vue3-ts/src/config/basic-info-config.js` file and enter the **SDKAppID** and **SDKSecretKey** you got when you activated the service:

```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

import LibGenerateTestUserSig from './lib-generate-test-usersig-es.min';

/**
 * Tencent Cloud SDKAppId, which should be replaced with user's SDKAppId.
 * Enter Tencent Cloud TRTC [Console] (https://console.cloud.tencent.com/trtc) to create an application,
 * and you will see the SDKAppId.
 * It is a unique identifier used by Tencent Cloud to identify users.
 */

export const SDKAPPID = 0;

/**
 * Encryption key for calculating signature, which can be obtained in the following steps:
 *
 * Step1. Enter Tencent Cloud TRTC [Console](https://console.cloud.tencent.com/rav),
 * and create an application if you don't have one.
 * Step2. Click your application to find "Quick Start".
 * Step3. Click "View Secret Key" to see the encryption key for calculating UserSig,
 * and copy it to the following variable.
 *
 * Notes: this method is only applicable for debugging Demo. Before official launch,
 * please migrate the UserSig calculation code and key to your backend server to avoid
 * unauthorized traffic use caused by the leakage of encryption key.
 * Document: https://intl.cloud.tencent.com/document/product/647/35166#Server
 */

export const SDKSECRETKEY = '';
```

### Note :

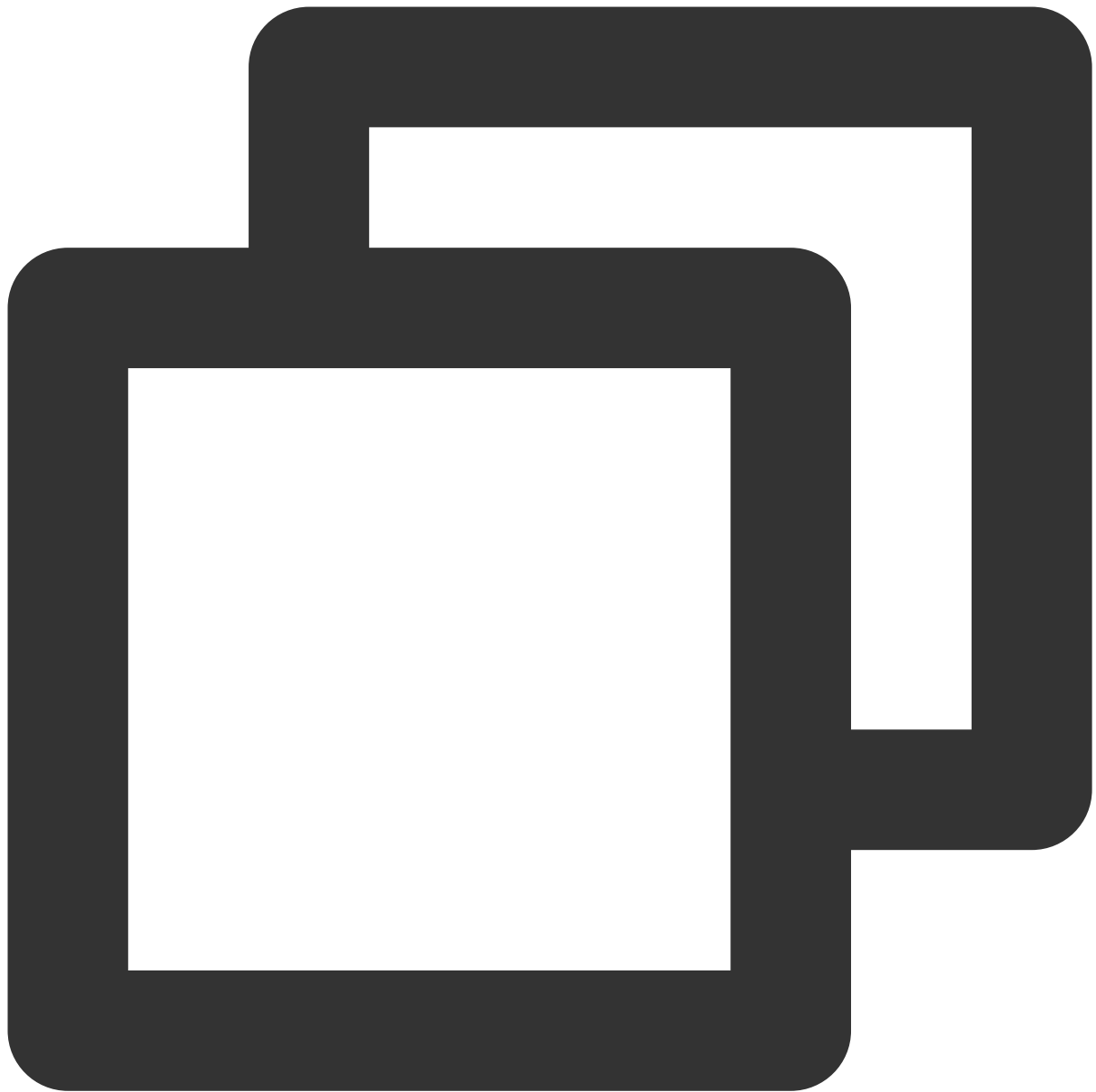
For Vue2 projects, open the **TUIRoomKit/Web/example/webpack-vue2.7-ts/src/config/basic-info-config.js** file and enter the **SDKAppID** and **SDKSecretKey** you got when you activated the service.

## Run Demo

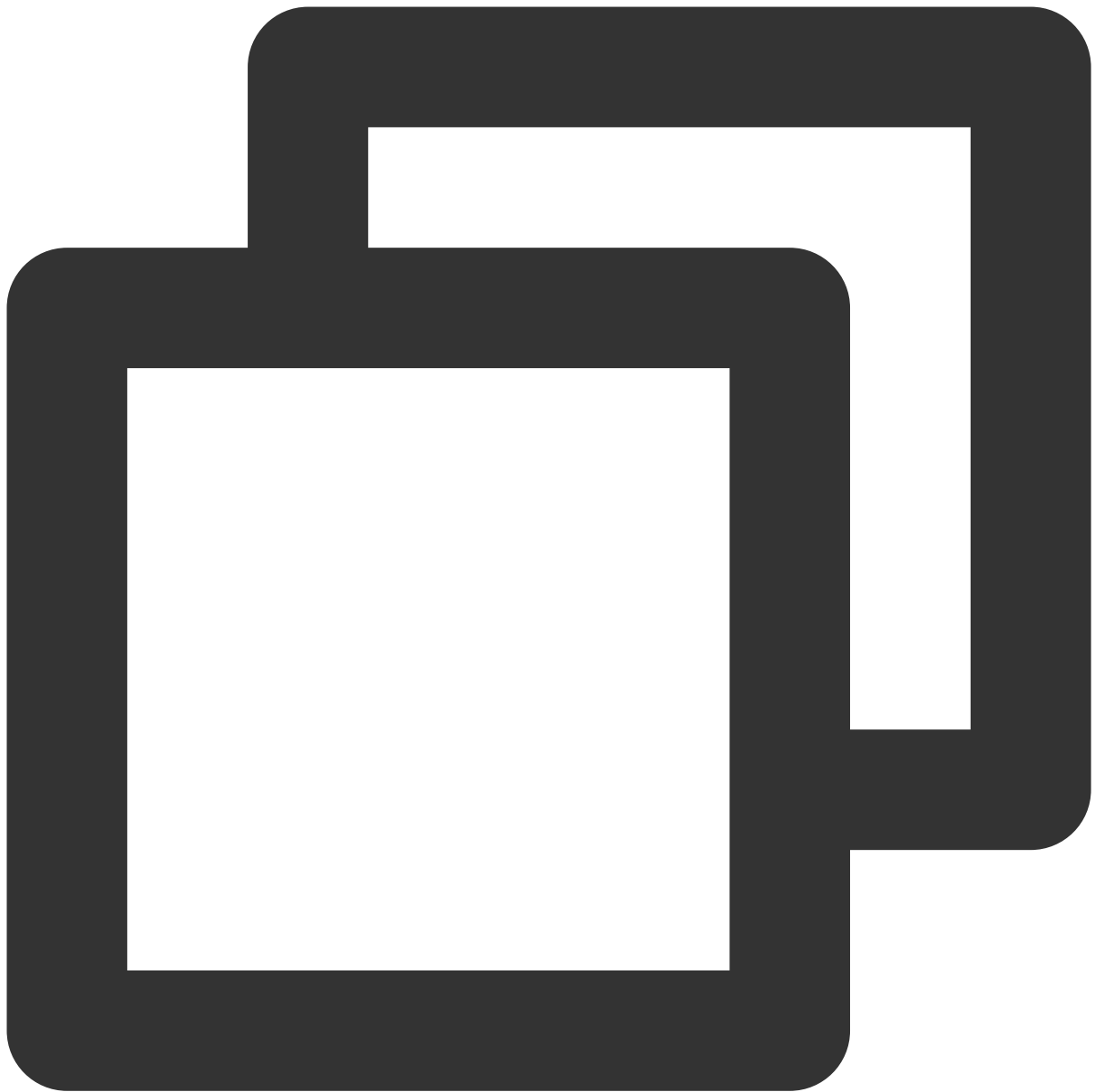
1. Run Demo by typing the command in the terminal.

Vue3

Vue2



```
# cd TUIRoomKit/Web/example/vite-vue3-ts  
npm run dev
```

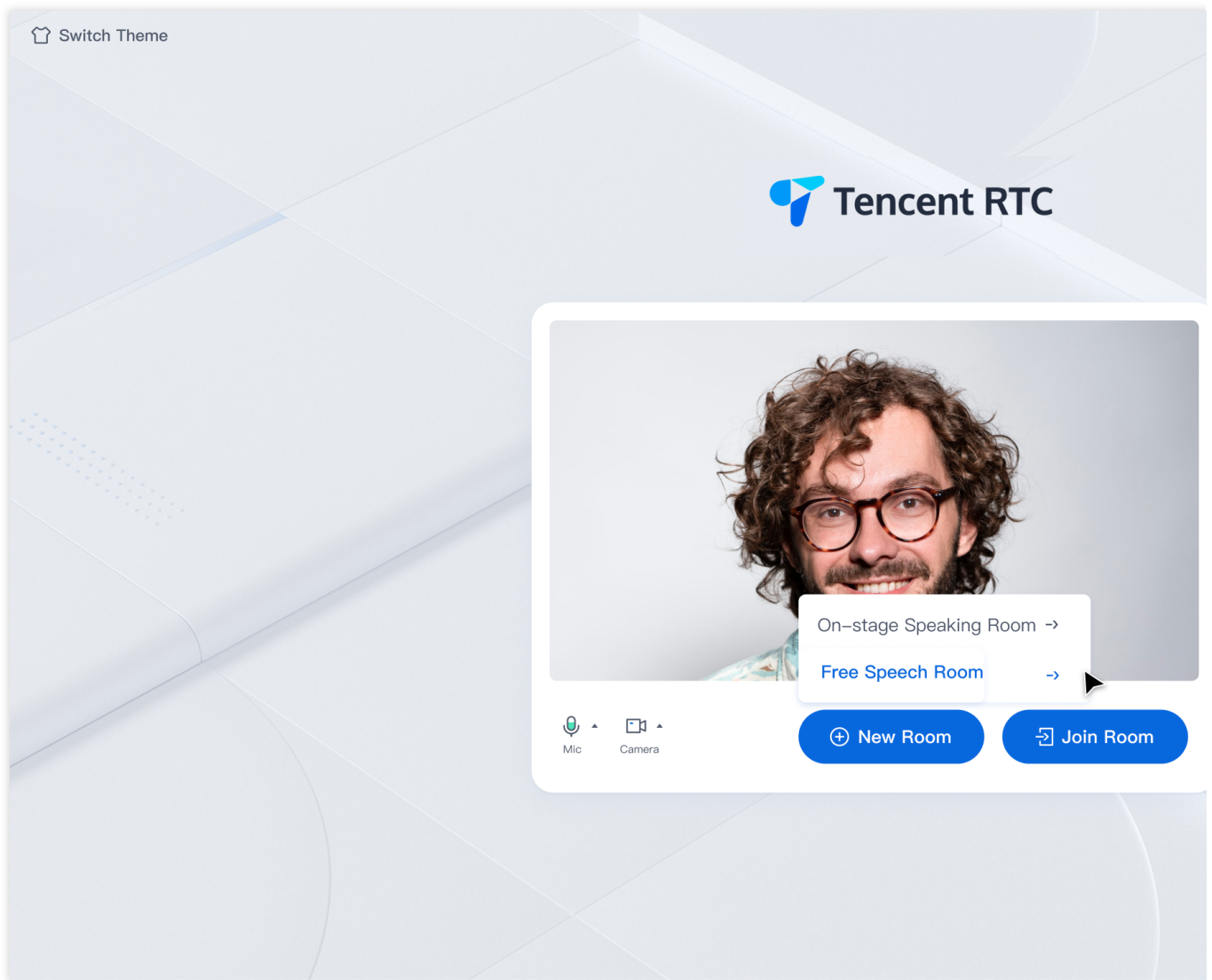


```
# cd TUIRoomKit/Web/example/webpack-vue2.7-ts  
npm run serve
```

**Note :**

For local environment, please access under localhost protocol, please refer to [the description of network access protocol](#).

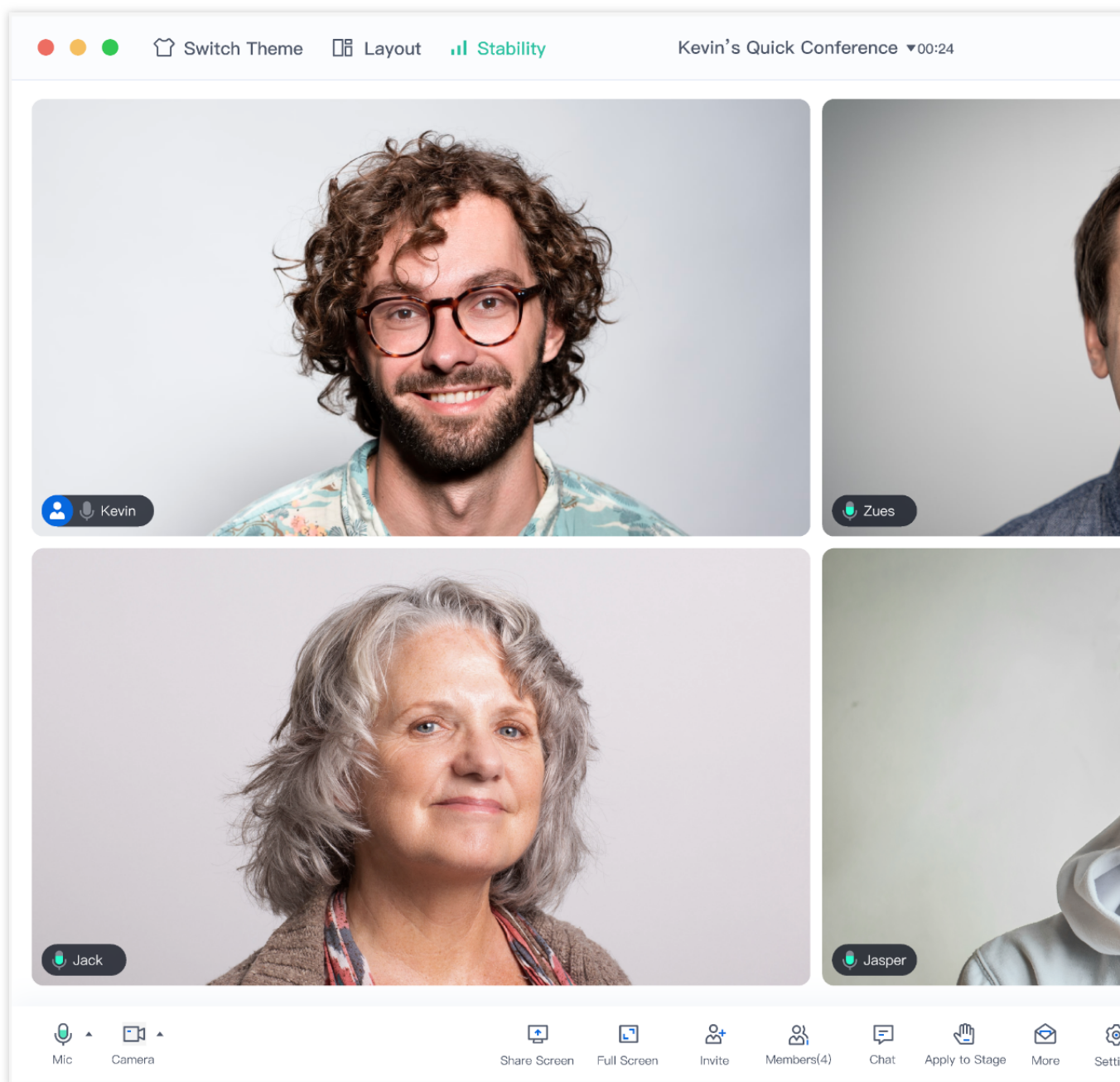
2. Open a browser page and enter the corresponding URL.



## Create your first conference

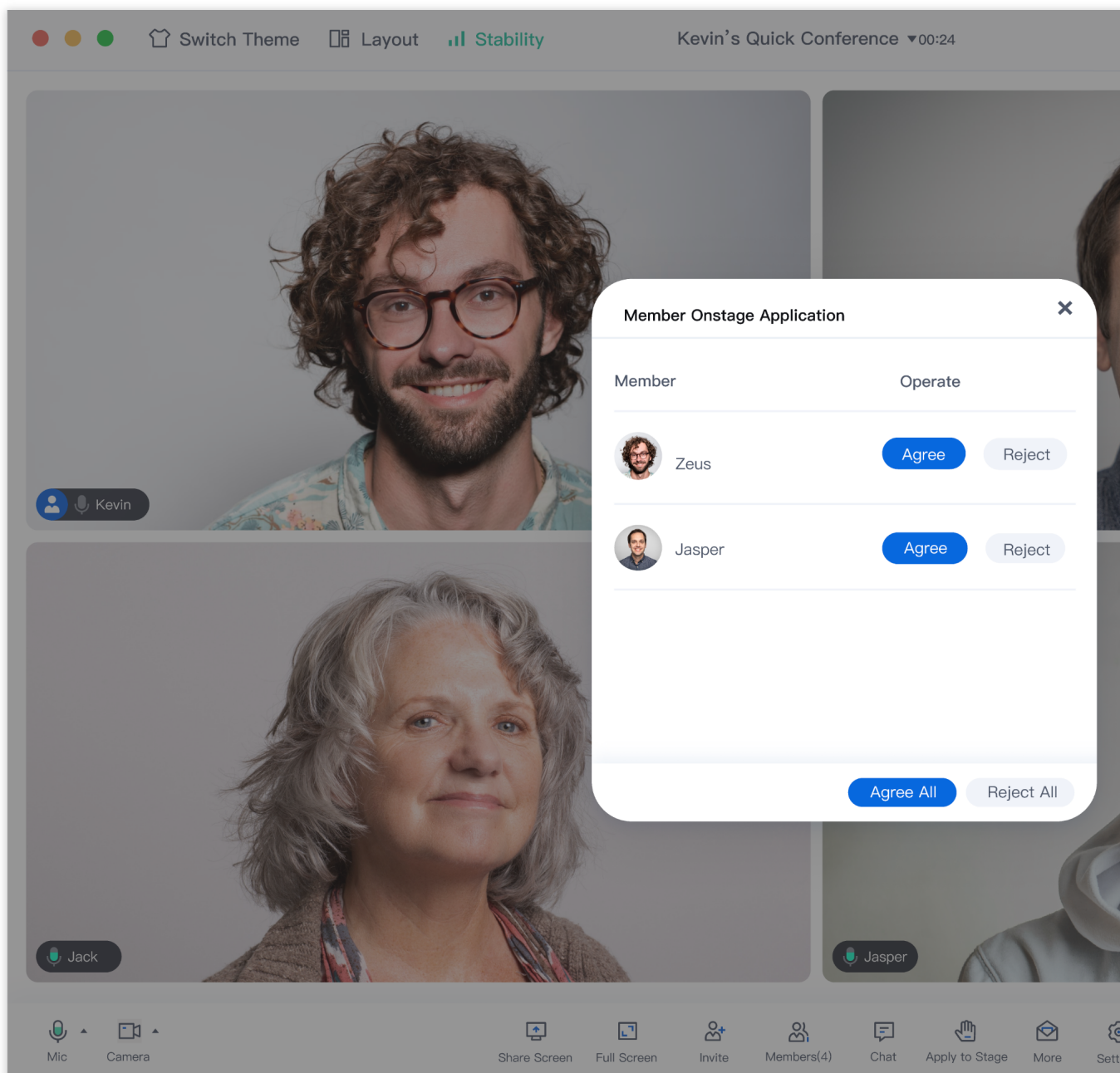
Click on the **New Room** button to create your first meeting room. The room types are **On-stage Speaking Room** or **Free Speech Room**.

### 1. Free speech room



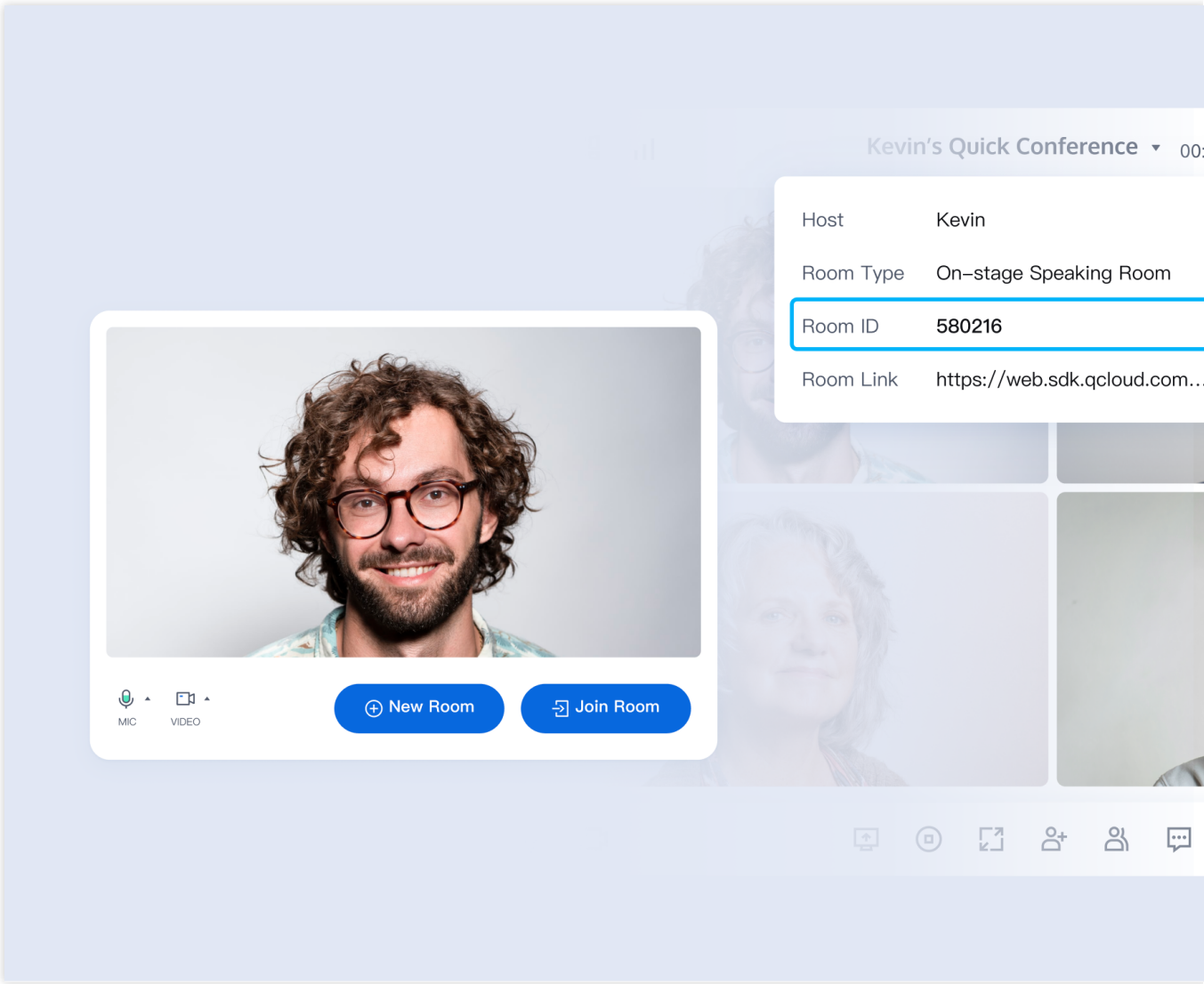
## 2. On-stage Speaking Room





## Join conference

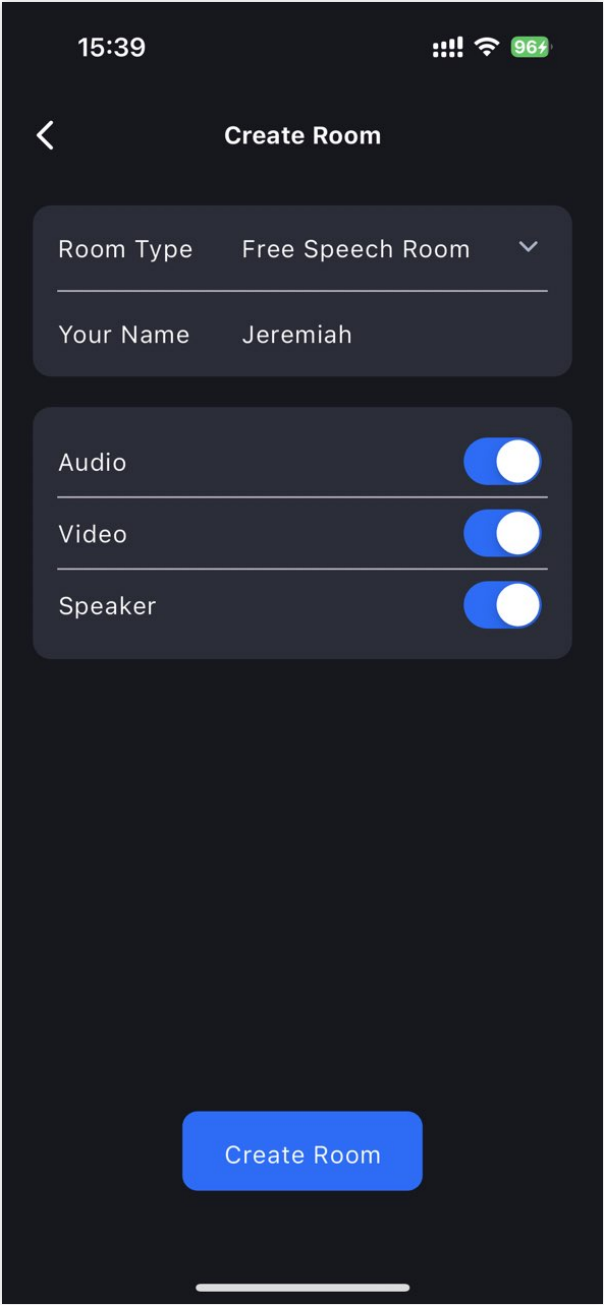
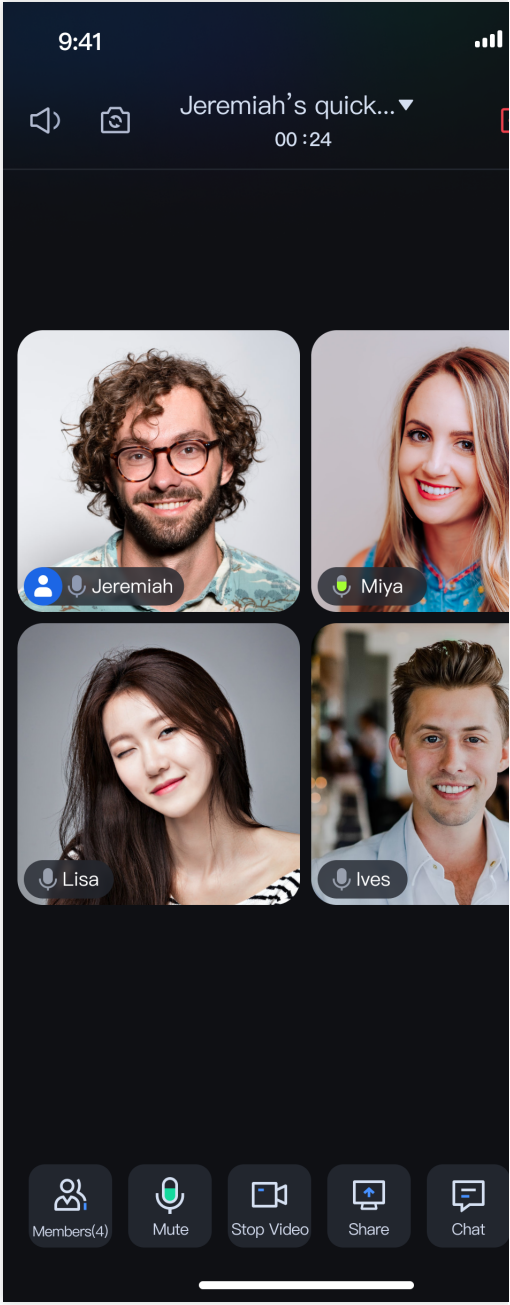
Participants can join a meeting created by the meeting host by filling in the corresponding `RoomId` .



# iOS

Last updated : 2024-08-15 17:06:16

This document mainly introduces how to quickly run through the **Conference (TUIRoomKit) sample project** and experience a high-quality multi-person video conference. By following this document, you can run through the demo within 10 minutes and ultimately experience a multi-person video conference feature with a complete UI interface.

Conference creation page	Conference main page
	

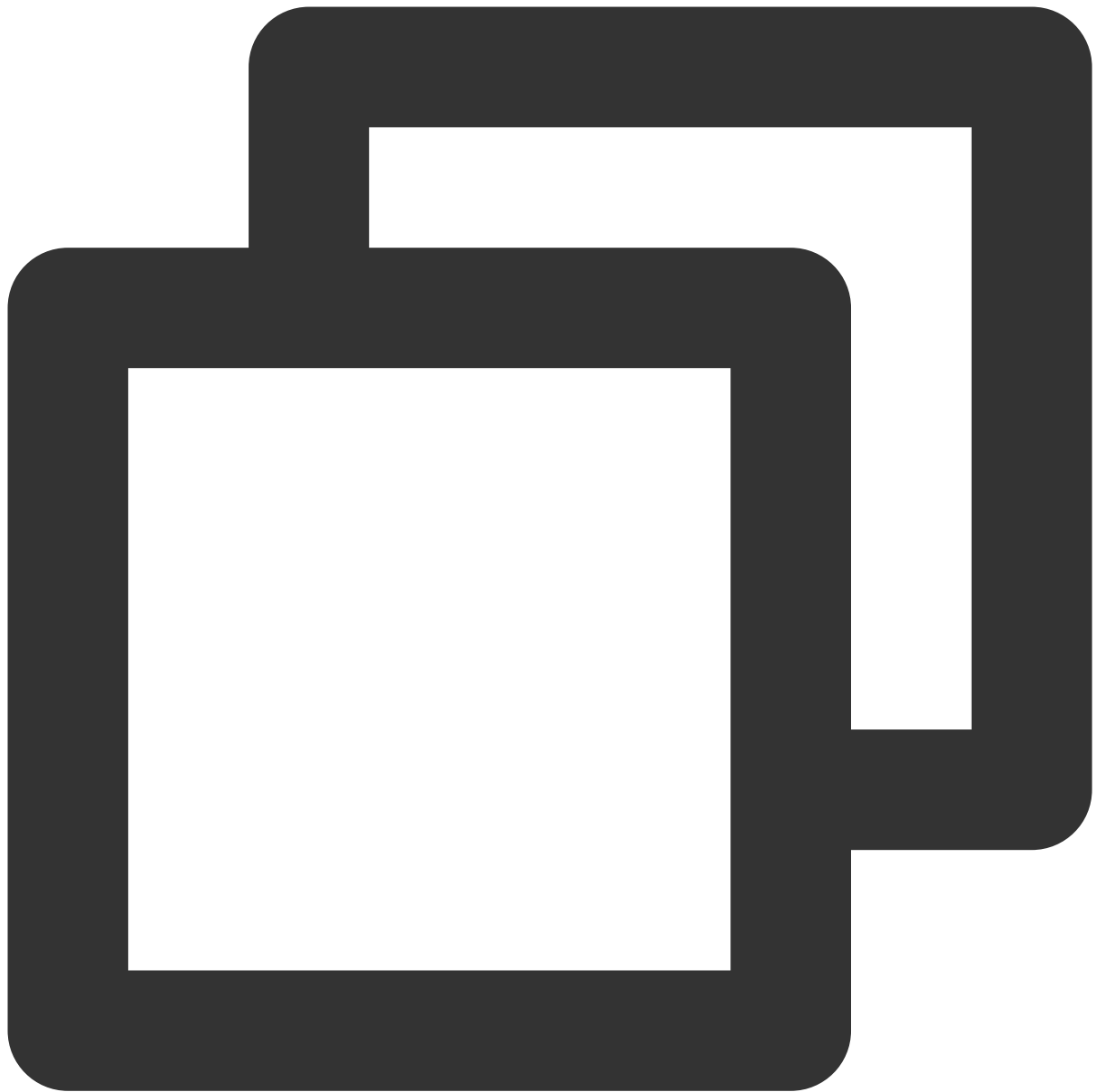
## Prerequisites

iOS 13.0 or later.

Xcode 15.0 or later.

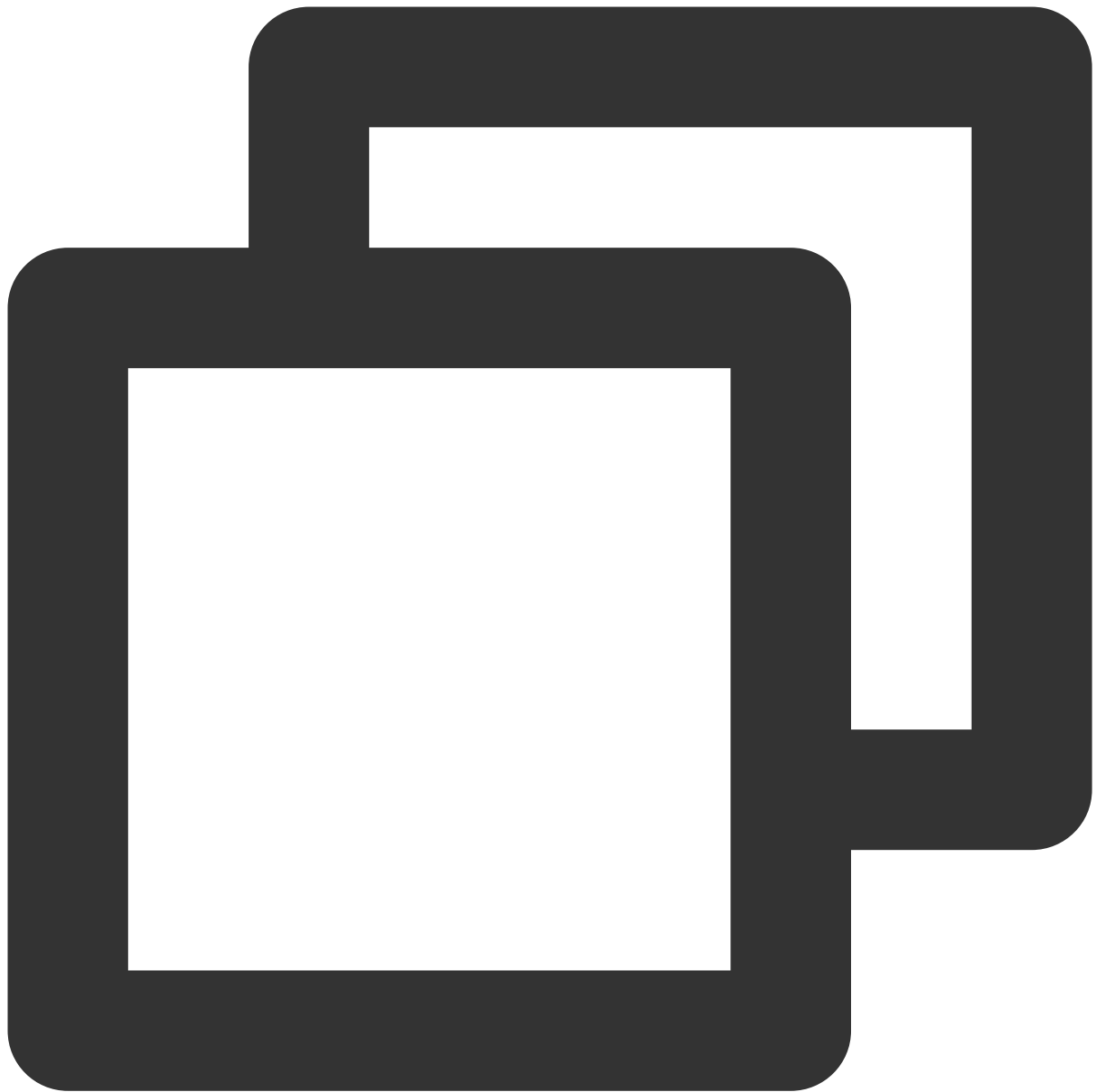
## Download the Demo

1. Download the [TUIRoomKit Demo](#) source code from GitHub, or directly execute the following command in the command line:



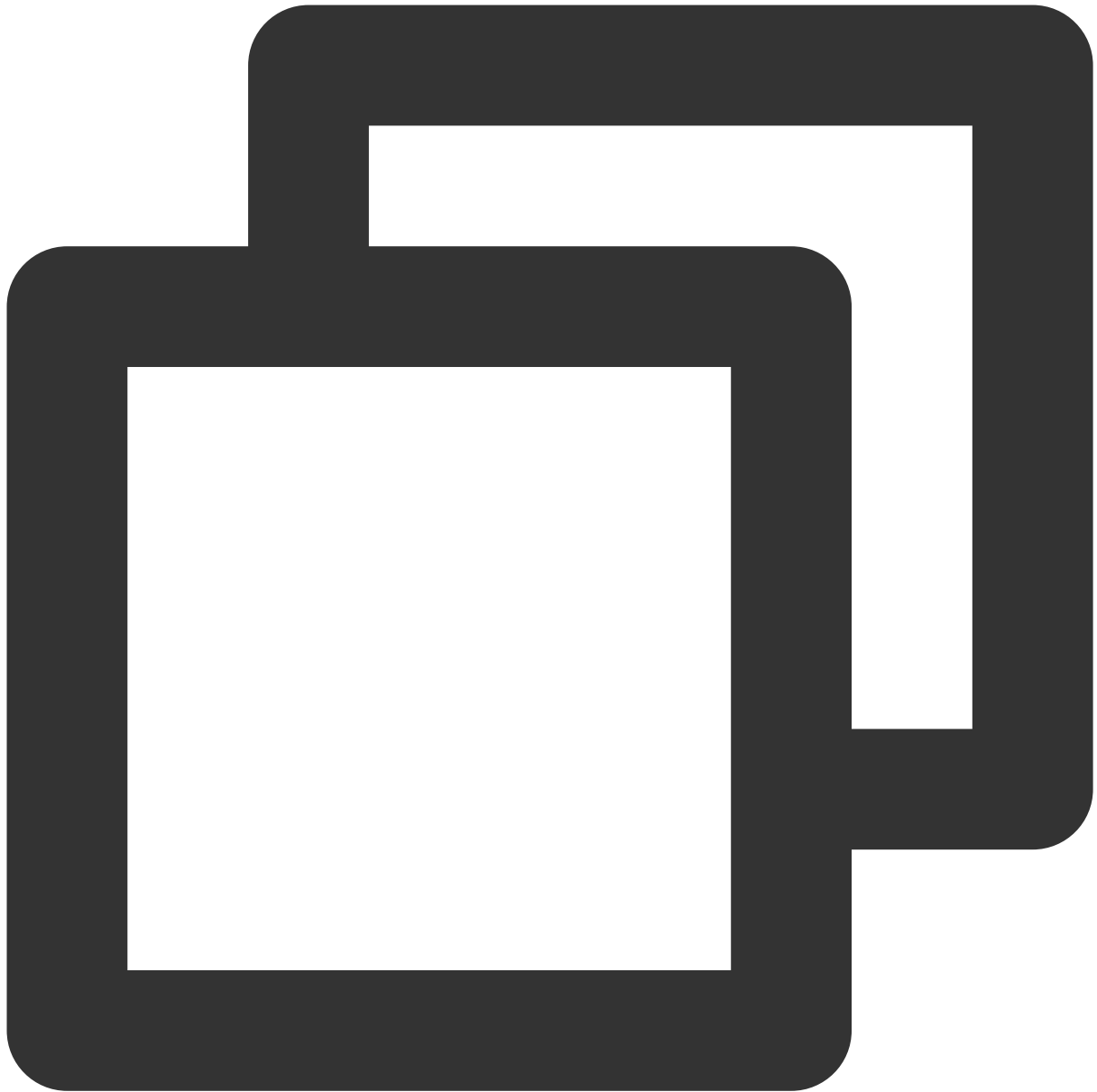
```
git clone https://github.com/Tencent-RTC/TUIRoomKit.git
```

2. Enter the iOS project directory in the command line:



```
cd TUIRoomKit/iOS/Example
```

3. Load the dependent libraries:



```
pod install
```

**Note:**

If you haven't installed CocoaPods, you can refer to [this](#) for instructions on how to install.

## Configure the Demo

1. [Activate the Conference services](#), to obtain the **SDKAppID** and **SDKSecretKey**.

### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

On-cloud recording	Dis:
Relay to CDN	Dis:
Callbacks	Dis:
Advanced permission control	Dis:

#### Products

Quickly run sample demo in 3 steps >

##### Conference

Edition	<a href="#">Conference : Trial &gt;</a>
Expiration time	2024-05-01
Auto-renewable	--

[Buy package](#)[Integrate](#)

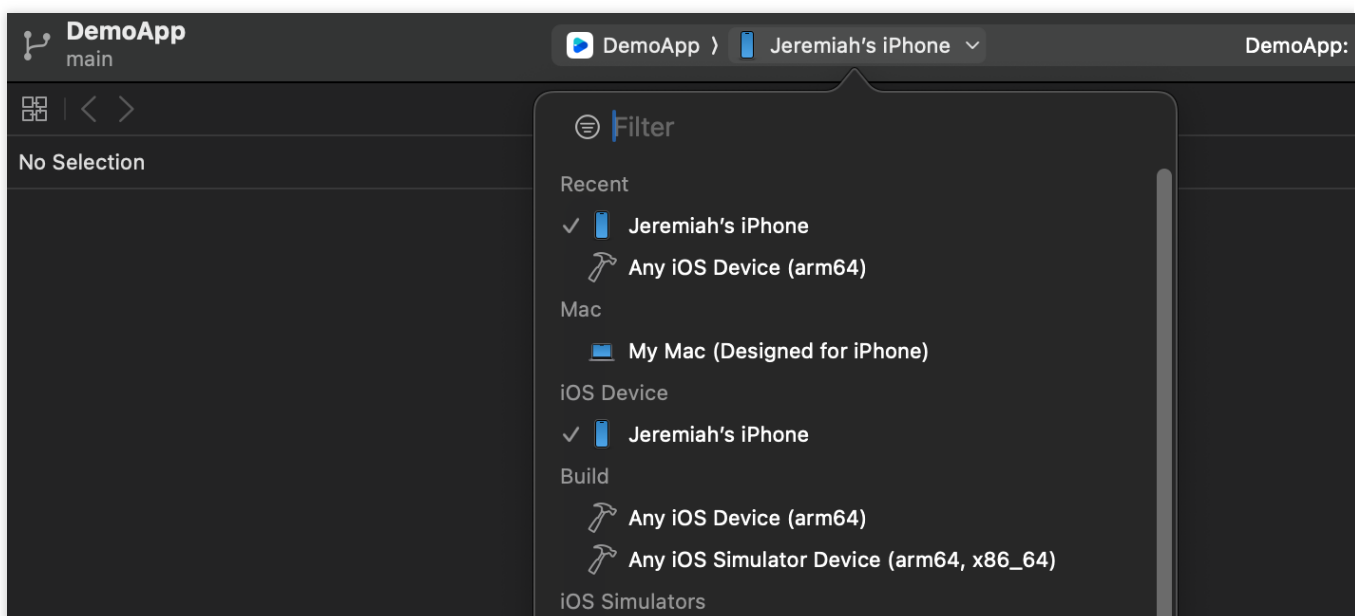
2. Open the project, and within it, find the `iOS/Example/Debug/GenerateTestUserSig.swift` file. Enter the corresponding **SDKAppID** and **SDKSecretKey** obtained from Back:



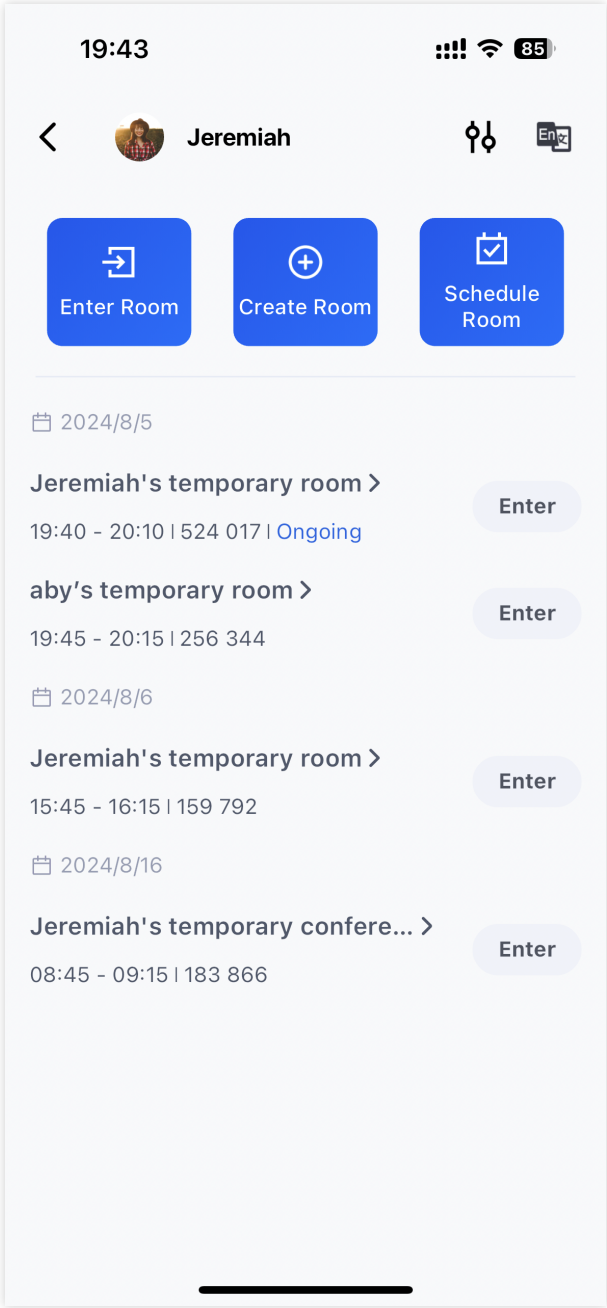
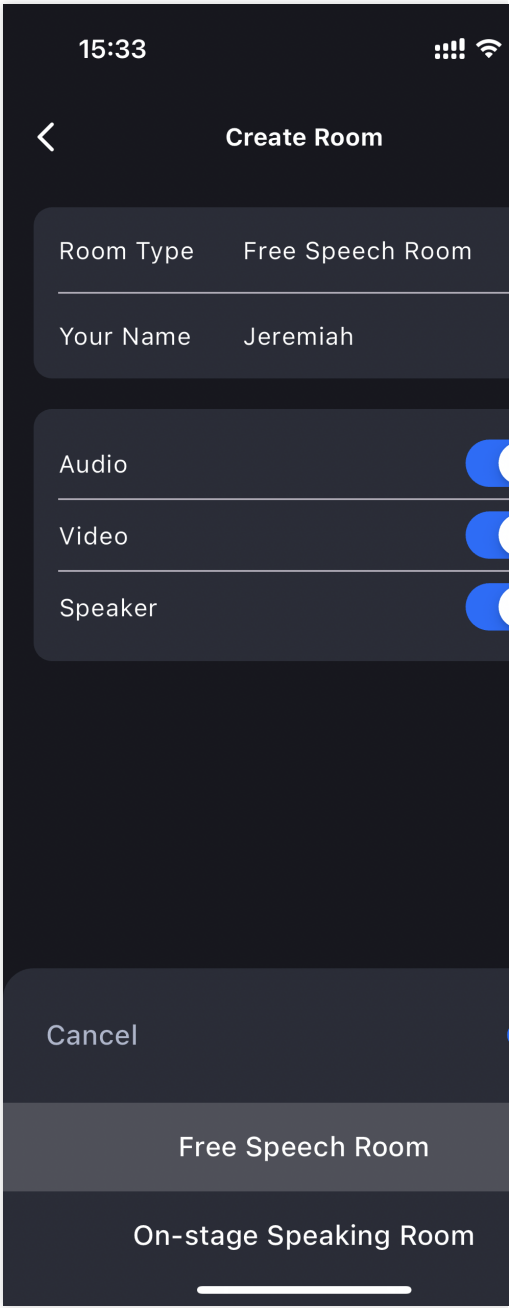
```
12 /**
13  * Tencent Cloud SDKAppID. Set it to the SDKAppID of your account.
14  * You can view your `SDKAppID` after creating an application in the [TRTC console] (https://console.trtc.io/).
15  * SDKAppID uniquely identifies a Tencent Cloud account.
16  */
17 let SDKAppID: Int = 0
18
19 /**
20  * Signature validity period, which should not be set too short
21  * Time unit: Second
22  * Default value: 604800 (7 days)
23  */
24 let EXPIRETIME: Int = 604_800
25
26 /**
27  * Follow the steps below to obtain the key required for UserSig calculation.
28  * Step 1. Log in to the [TRTC console] (https://console.trtc.io/). If you don't have an application yet, create one
29  * Step 2. Click your application and find "Basic Information".
30  * Step 3. Click "Display Key" to view the key used for UserSig calculation. Copy and paste the key to the variable below.
31  * Note: This method is for testing only. Before commercial launch,
32  * please migrate the UserSig calculation code and key to your backend server to prevent key disclosure and traffic stealing.
33  * Documentation: https://trtc.io/document/35166
34  */
35 let SDKSecretKey = ""
36
```

## Running the Demo

1. In Xcode, select the device you want to run the Demo on as shown below:



2. After selecting, click to run and our TUIRoomKit iOS Demo will run on the target device.

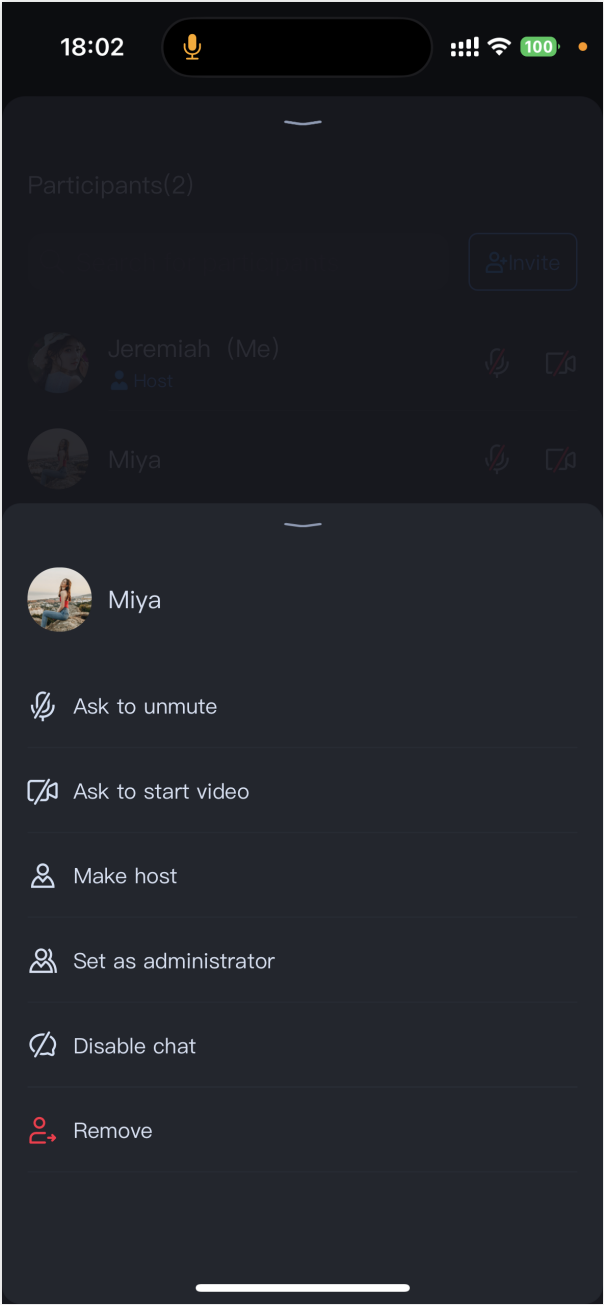
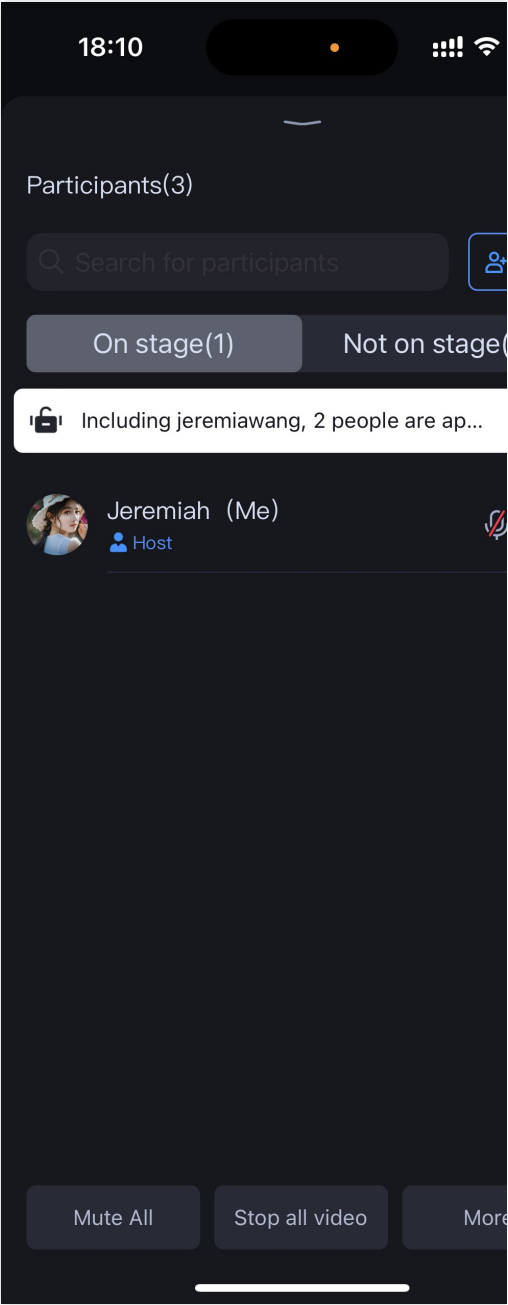
APP main page	Conference creation page
	

## Create your first conference

Click on the **Create Room** button to create your first meeting room. The room types are **On-stage Speaking Room** and **Free Speech Room**.

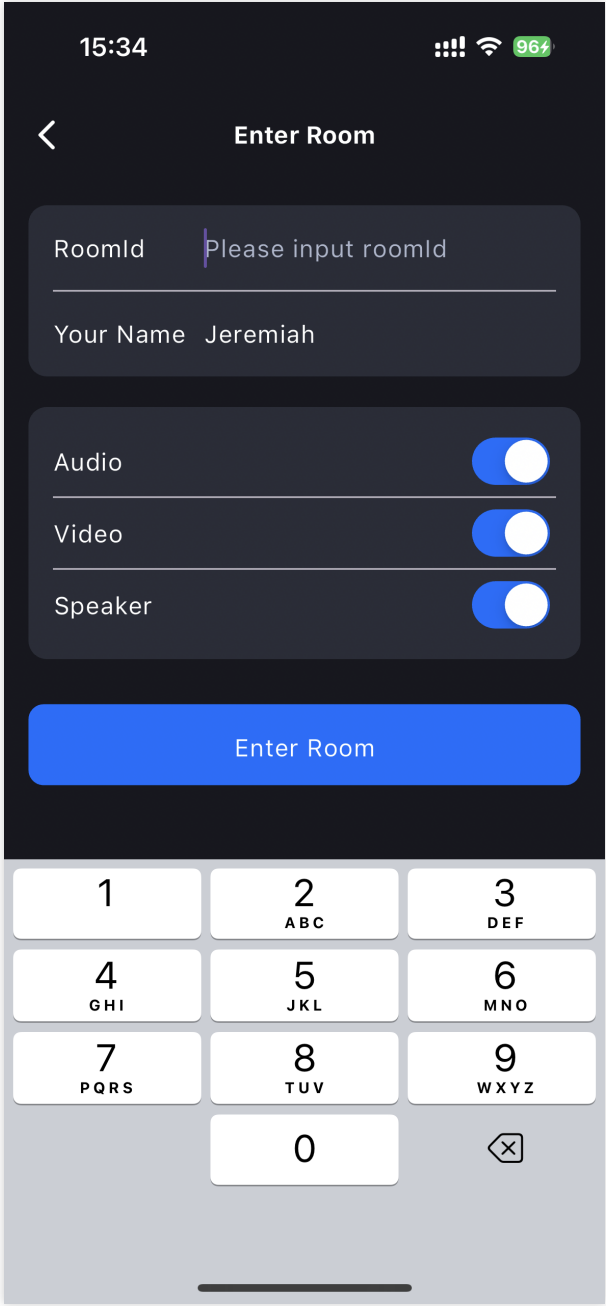
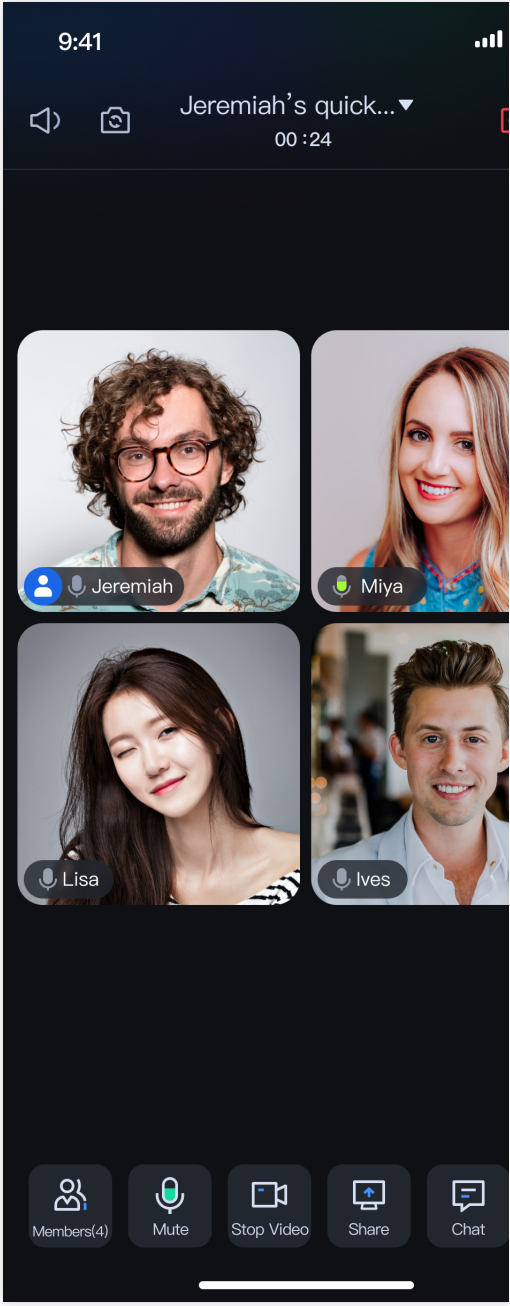
**Free Speech Room:** Regular users can freely speak and have the liberty to turn their microphones and cameras on or off.

**On-stage Speaking Room:** Only users on stage can freely turn their microphones and cameras on or off. Regular audience members can apply to become stage users by raising their hand.

Free speech room member management panel	On-stage speaking room member list panel
 A screenshot of a mobile app interface for a 'Free speech room'. At the top, the time is 18:02. Below the status bar, there's a header 'Participants(2)'. A list shows two participants: 'Jeremiah (Me)' with a 'Host' tag and 'Miya'. To the right of the list is an 'Invite' button. Below the list, a detailed view for 'Miya' is shown, including a profile picture and a list of actions: 'Ask to unmute', 'Ask to start video', 'Make host', 'Set as administrator', 'Disable chat', and 'Remove'.	 A screenshot of a mobile app interface for an 'On-stage speaking room'. At the top, the time is 18:10. Below the status bar, there's a header 'Participants(3)'. A search bar 'Search for participants' is present. Below it are two tabs: 'On stage(1)' and 'Not on stage(0)'. A notification banner says 'Including jeremiawang, 2 people are ap...'. Below this, a participant 'Jeremiah (Me)' is shown with a 'Host' tag. At the bottom, there are buttons for 'Mute All', 'Stop all video', and 'More'.

## Join conference

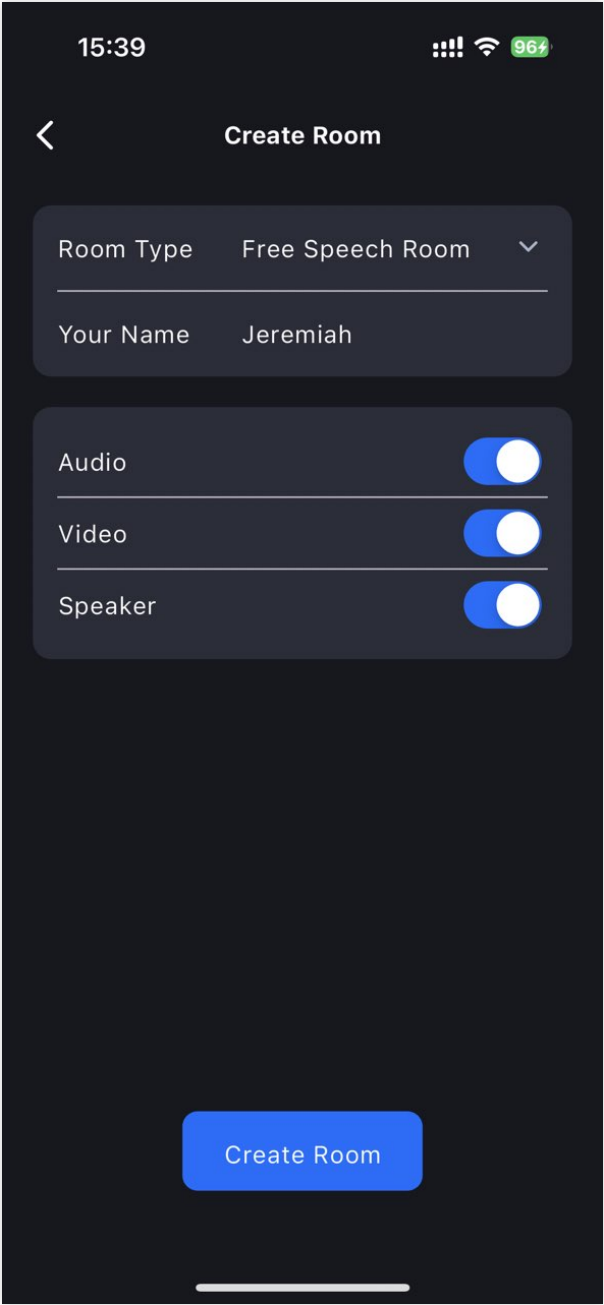
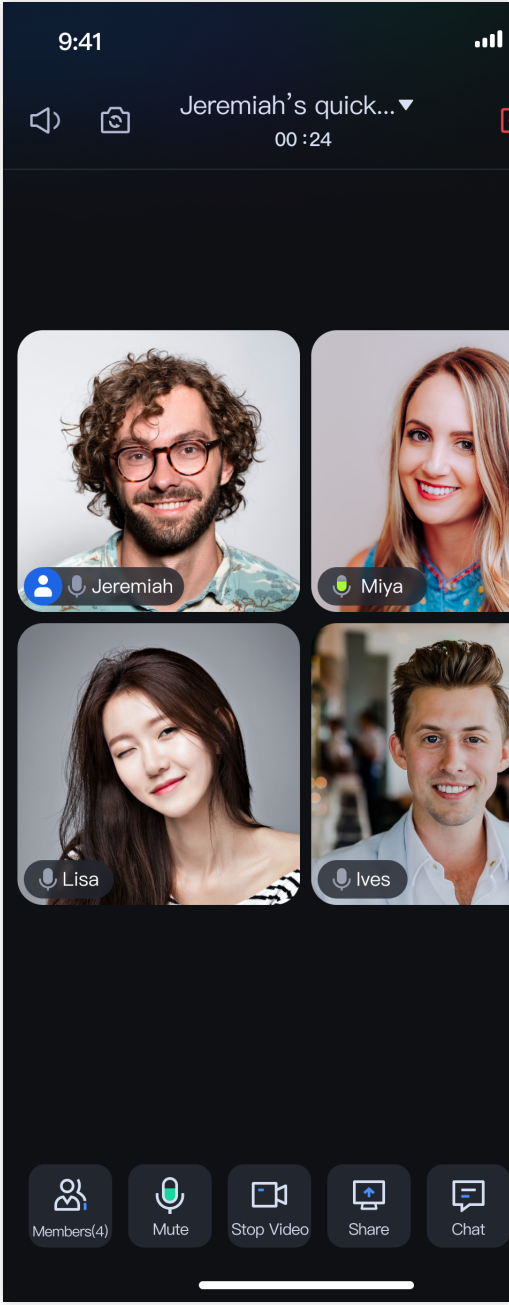
After clicking **Join Room**, participants can join the meeting created by the host by filling in the corresponding RoomId .

Join conference page	Conference main page
	

# Android

Last updated : 2024-08-15 17:06:16

This document mainly introduces how to quickly run through the **Conference (TUIRoomKit) sample project** and experience a high-quality multi-person video conference. By following this document, you can run through the demo within 10 minutes and ultimately experience a multi-person video conference feature with a complete UI interface.

Conference creation page	Conference main page
	

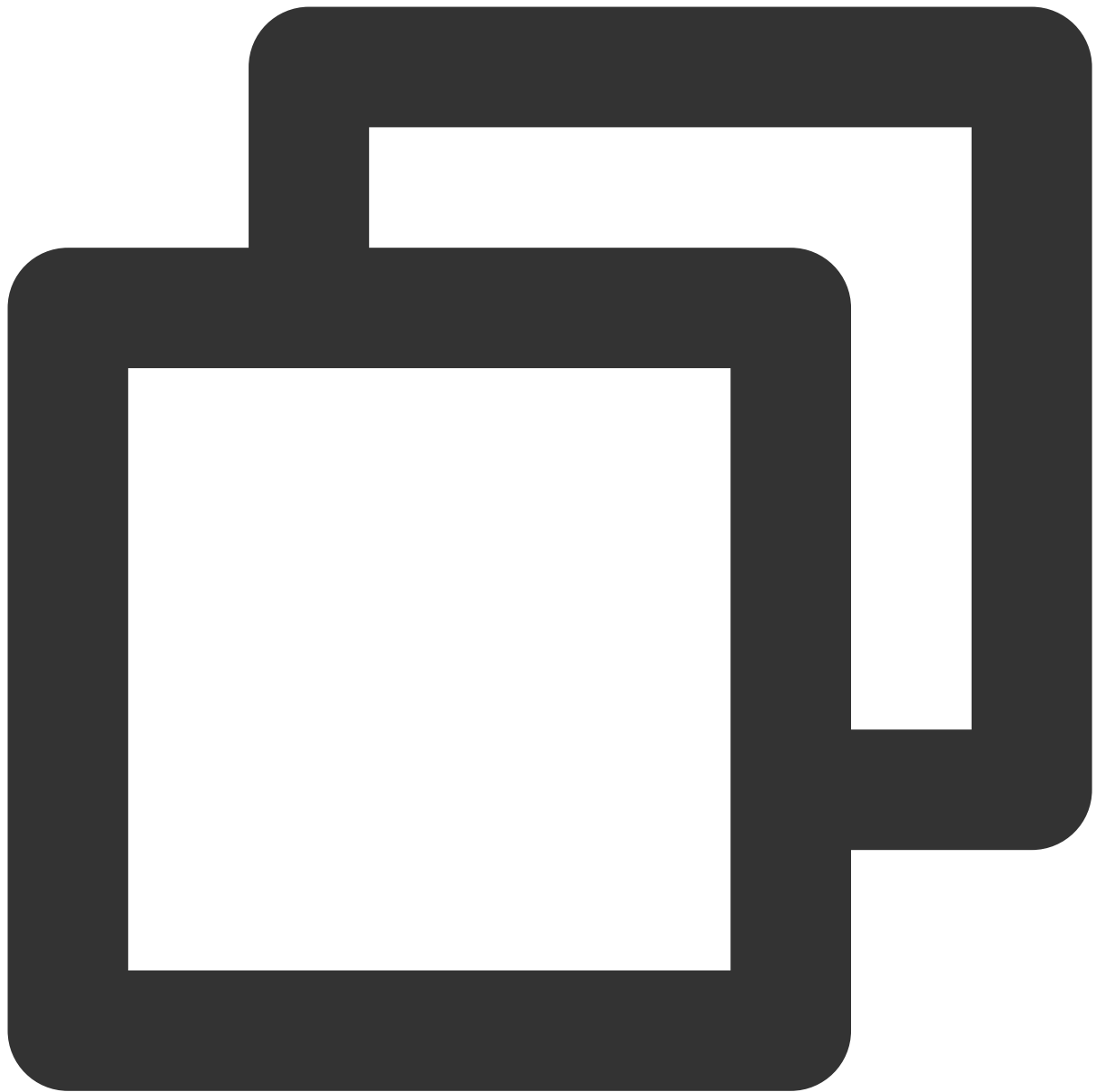
## Prerequisites

Minimum compatibility with Android 4.4 (SDK API Level 19), with Android 5.0 (SDK API Level 21) or higher recommended.

Android Studio 3.5 or above.

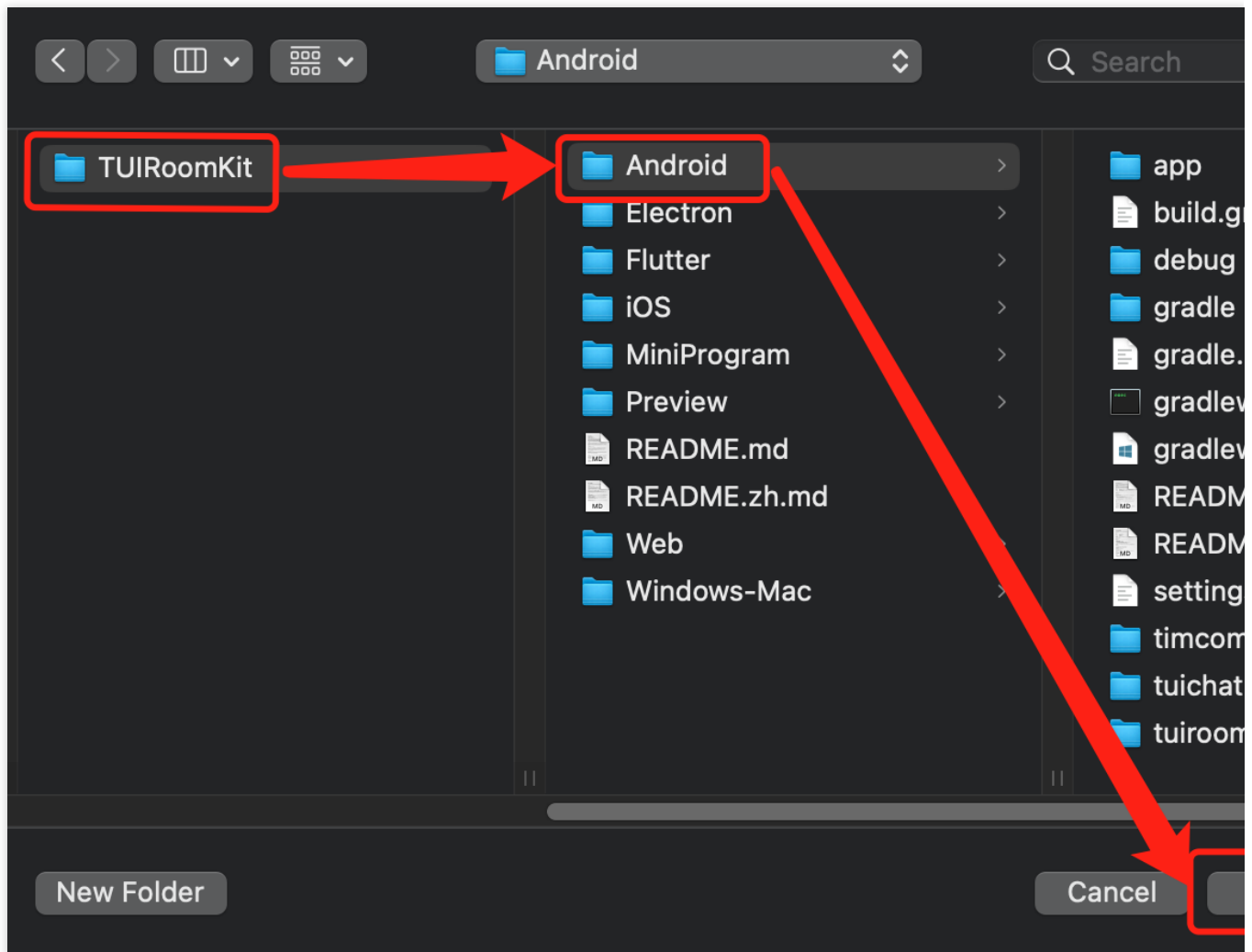
## Download the Demo

1. Download the [TUIRoomKit Demo](#) source code from GitHub, or directly execute the following command in the command line:



```
git clone https://github.com/Tencent-RTC/TUIRoomKit.git
```

2. Open the TUIRoomKit Android project via Android Studio:



## Configure the Demo

1. [Activate the Conference services](#), to obtain the **SDKAppID** and **SDKSecretKey**.



### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

On-cloud recording	Dis:
Relay to CDN	Dis:
Callbacks	Dis:
Advanced permission control	Dis:

#### Products

[Quickly run sample demo in 3 steps >](#)

##### Conference

Edition	<a href="#">Conference : Trial &gt;</a>
Expiration time	2024-05-01
Auto-renewable	--

[Buy package](#)[Integrate](#)

2. Open the project and find the

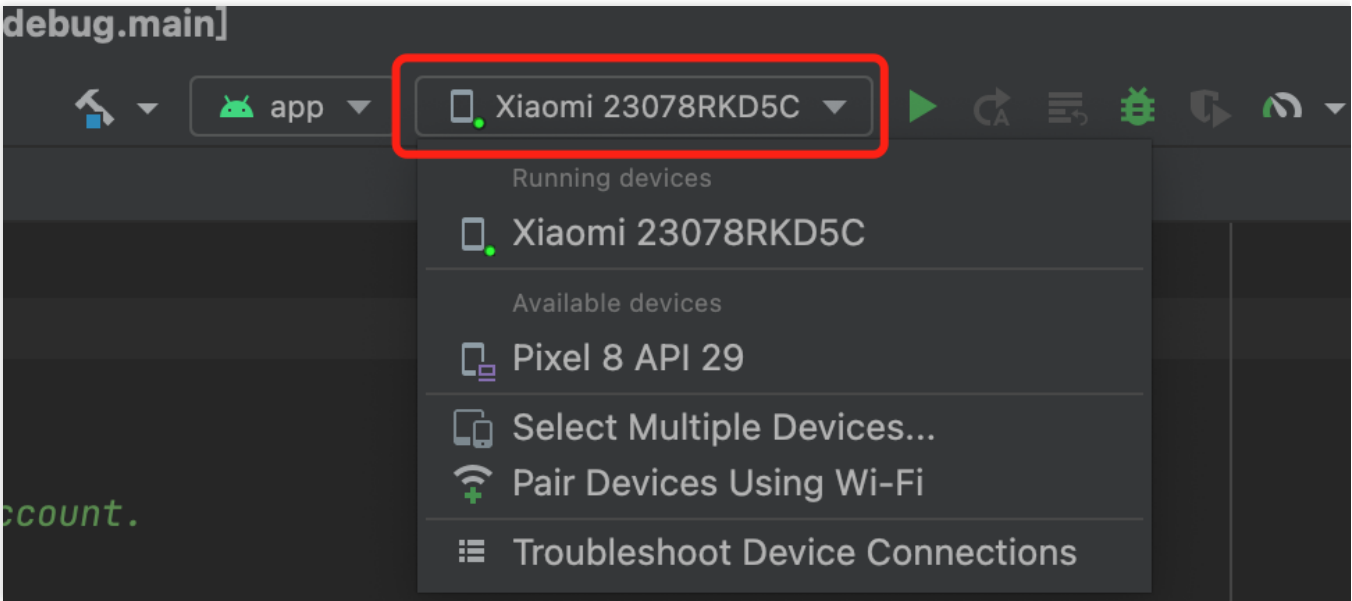
`Android/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java` file.

Enter the corresponding **SDKAppID** and **SDKSecretKey** obtained from Back:

```
public class GenerateTestUserSig {  
  
    /**  
     * Tencent Cloud SDKAppID. Set it to the SDKAppID of your account.  
     * <p>  
     * You can view your `SDKAppId` after creating an application in the [TRTC console](https://console.trtc.io/).  
     * SDKAppID uniquely identifies a Tencent Cloud account.  
     */  
    3 usages  
    public static final int SDKAppID = PLACEHOLDER;  
  
    /**  
     * Signature validity period, which should not be set too short  
     * <p>  
     * Unit: Second  
     * Default value: 604800 (seven days)  
     */  
    1 usage  
    private static final int EXPIRE_TIME = 604800;  
  
    /**  
     * Follow the steps below to obtain the key required for UserSig calculation.  
     * <p>  
     * Step 1. Log in to the [TRTC console](https://console.trtc.io/). If you don't have an application yet, create one.  
     * Step 2. Click your application and find "Basic Information".  
     * Step 3. Click "Display Key" to view the key used for UserSig calculation.  
     * Copy and paste the key to the variable below.  
     * <p>  
     * Note: This method is for testing only. Before commercial launch,  
     * please migrate the UserSig calculation code and key to your backend server  
     * to prevent key disclosure and traffic stealing.  
     * Documentation: https://trtc.io/document/35166  
     */  
    1 usage  
    private static final String SDKSecretKey = "PLACEHOLDER";  
}
```

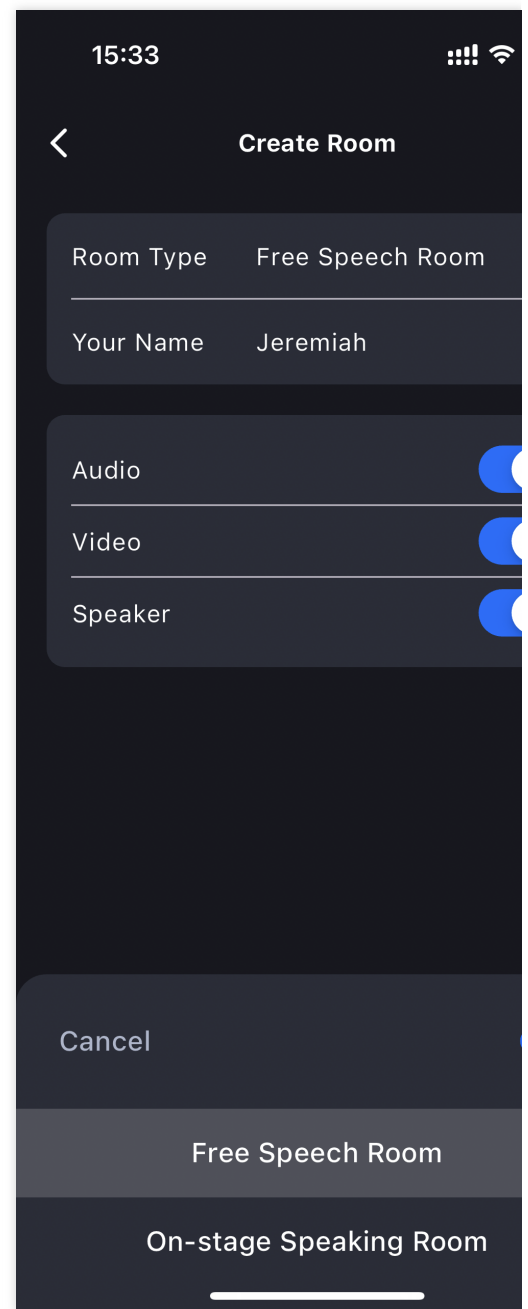
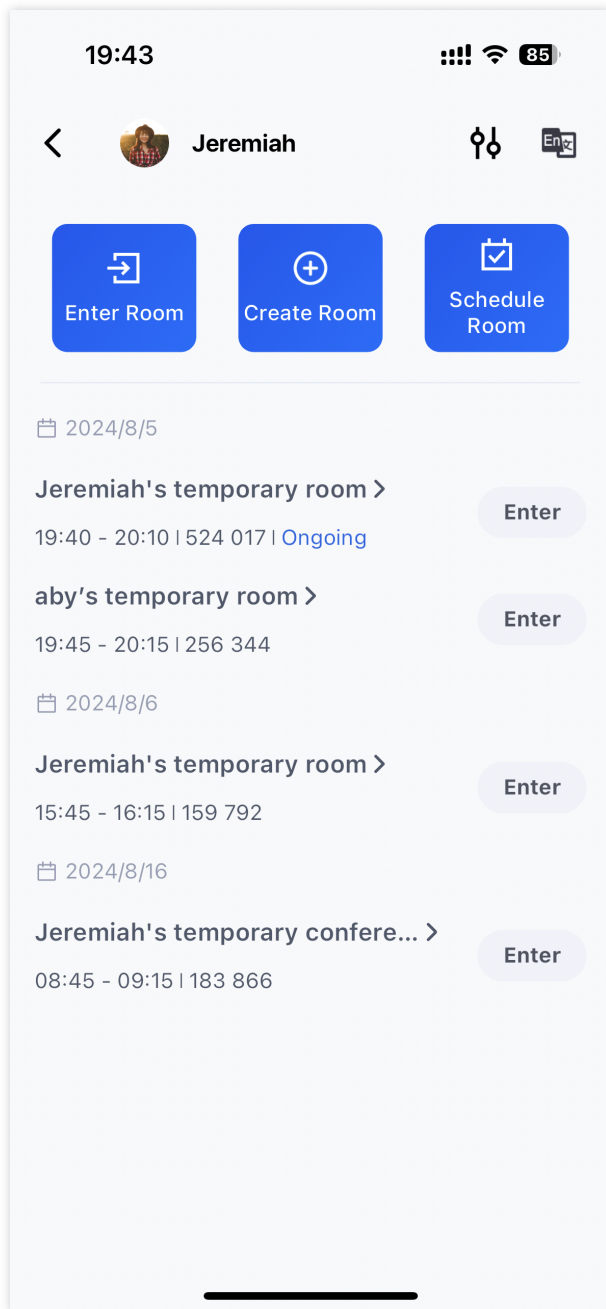
## Running the Demo

1. In the top right corner of Android Studio, select the device you want to run the Demo on as shown below:



2. After selection, click to run, deploying the TUIRoomKit Android Demo to the target device.

APP main page	Conference creation page

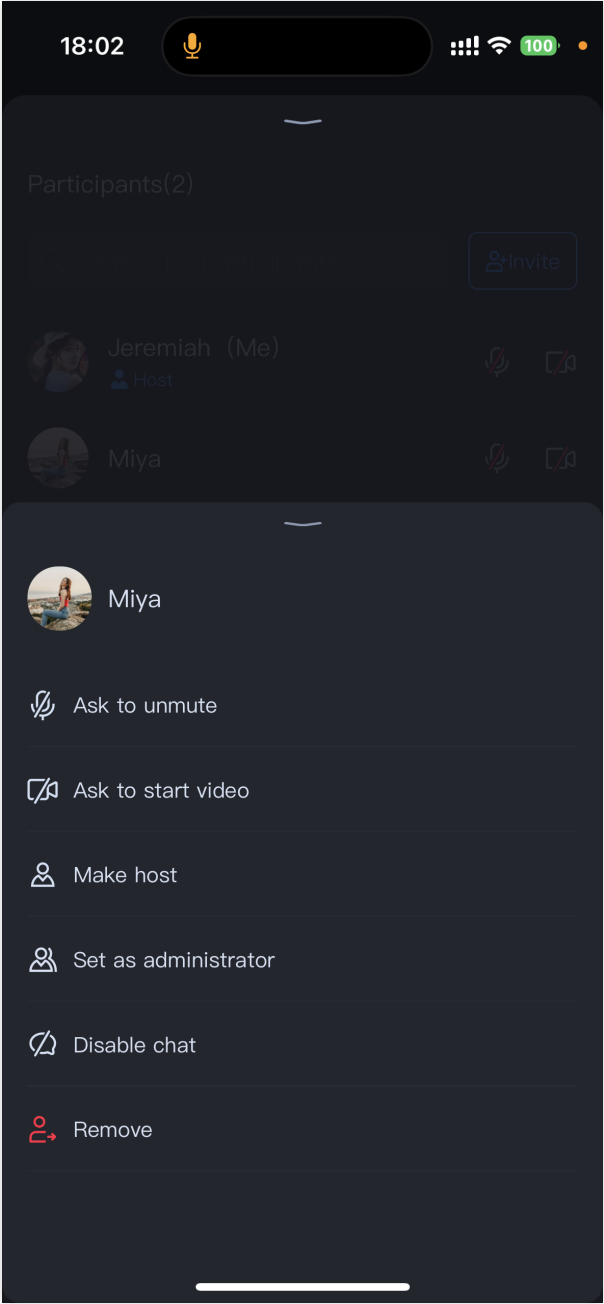
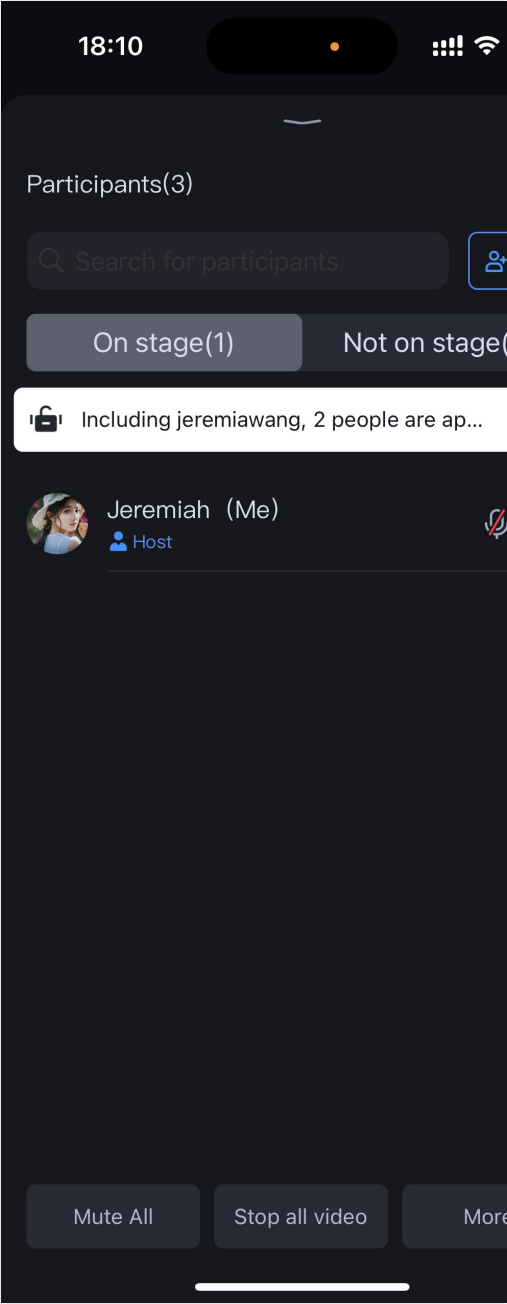


## Create your first conference

Click on the **Create Room** button to create your first meeting room. The room types are **On-stage Speaking Room** and **Free Speech Room**.

**Free Speech Room:** Regular users can freely speak and have the liberty to turn their microphones and cameras on or off.

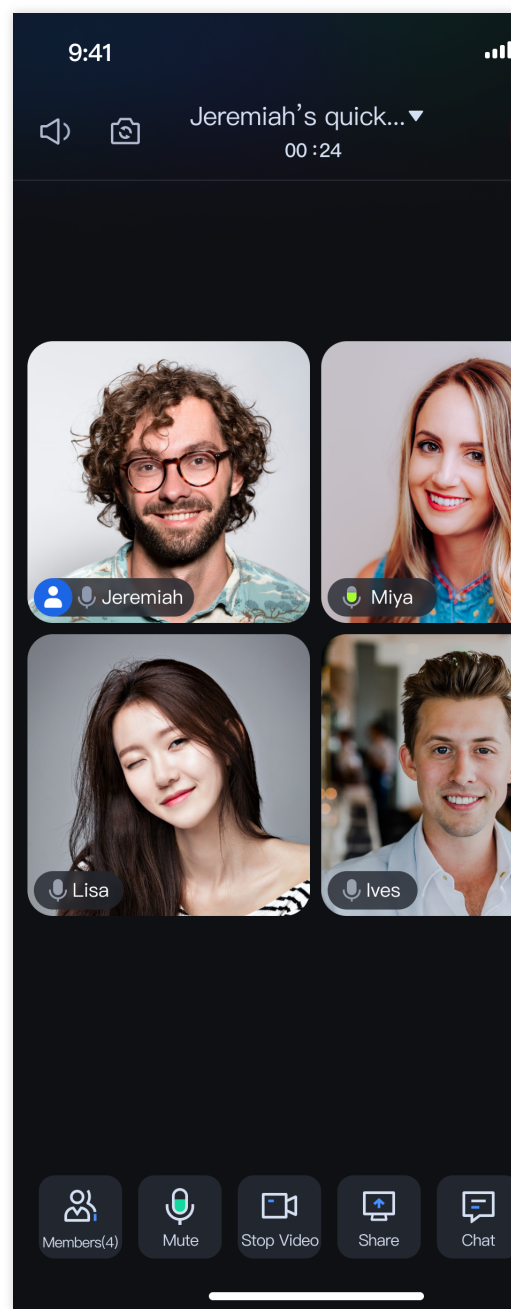
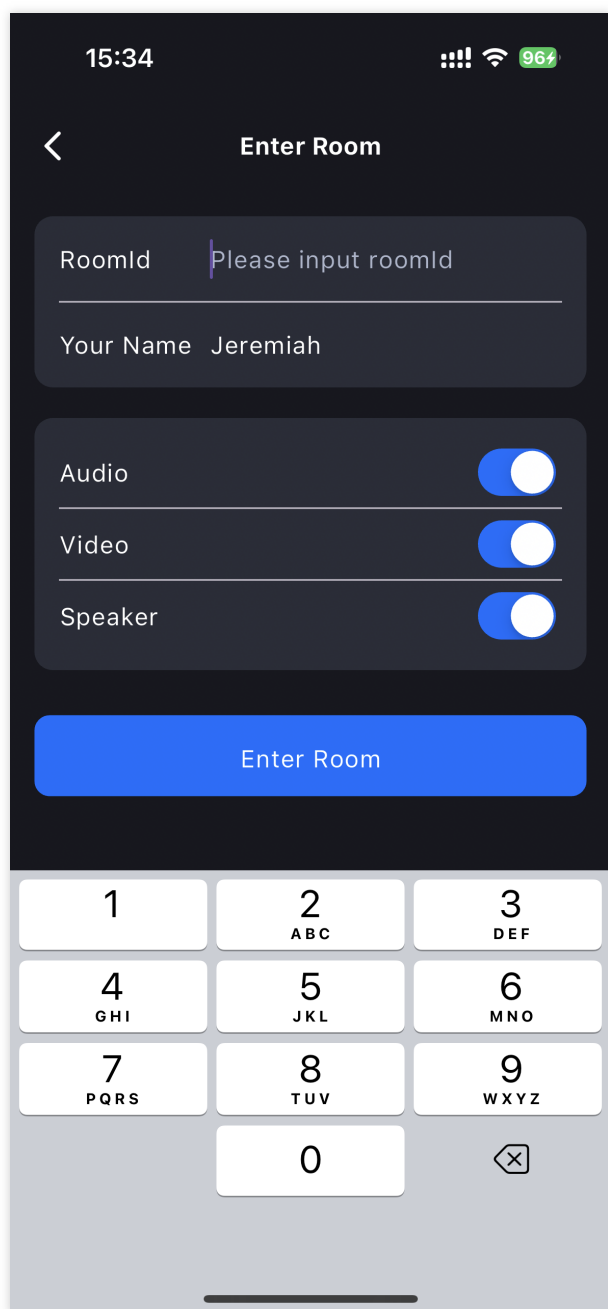
**On-stage Speaking Room:** Only users on stage can freely turn their microphones and cameras on or off. Regular audience members can apply to become stage users by raising their hand.

Free speech room member management panel	On-stage speaking room member list panel
	

## Join conference

After clicking **Join Room**, participants can join the meeting created by the host by filling in the corresponding `RoomId` .

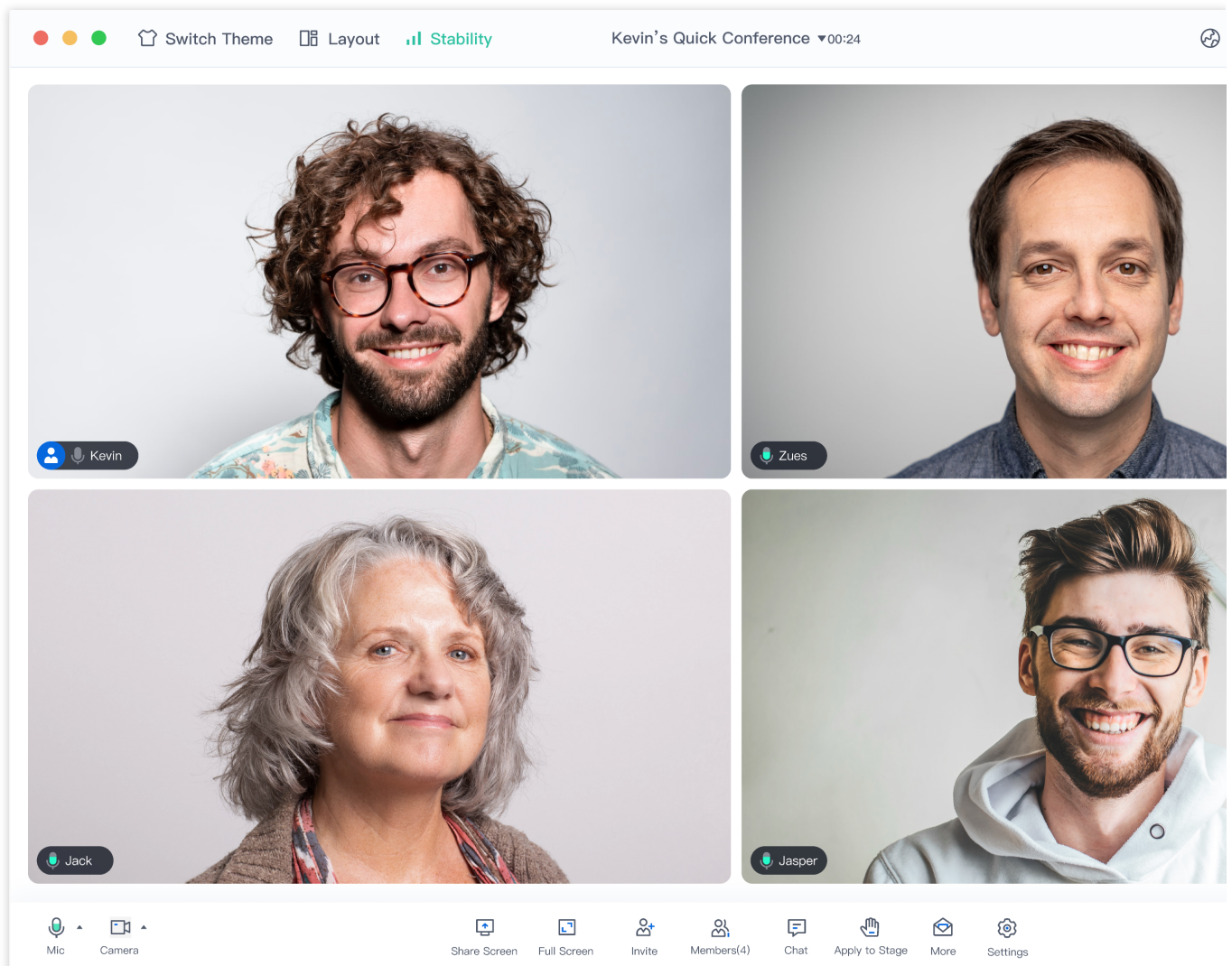
Join conference page	Conference main page



# Electron

Last updated : 2024-08-15 17:06:16

This document mainly introduces how to quickly run through the **Conference (TUIRoomKit) sample project** and experience a high-quality multi-person video conference. By following this document, you can run through the demo within 10 minutes and ultimately experience a multi-person video conference feature with a complete UI interface



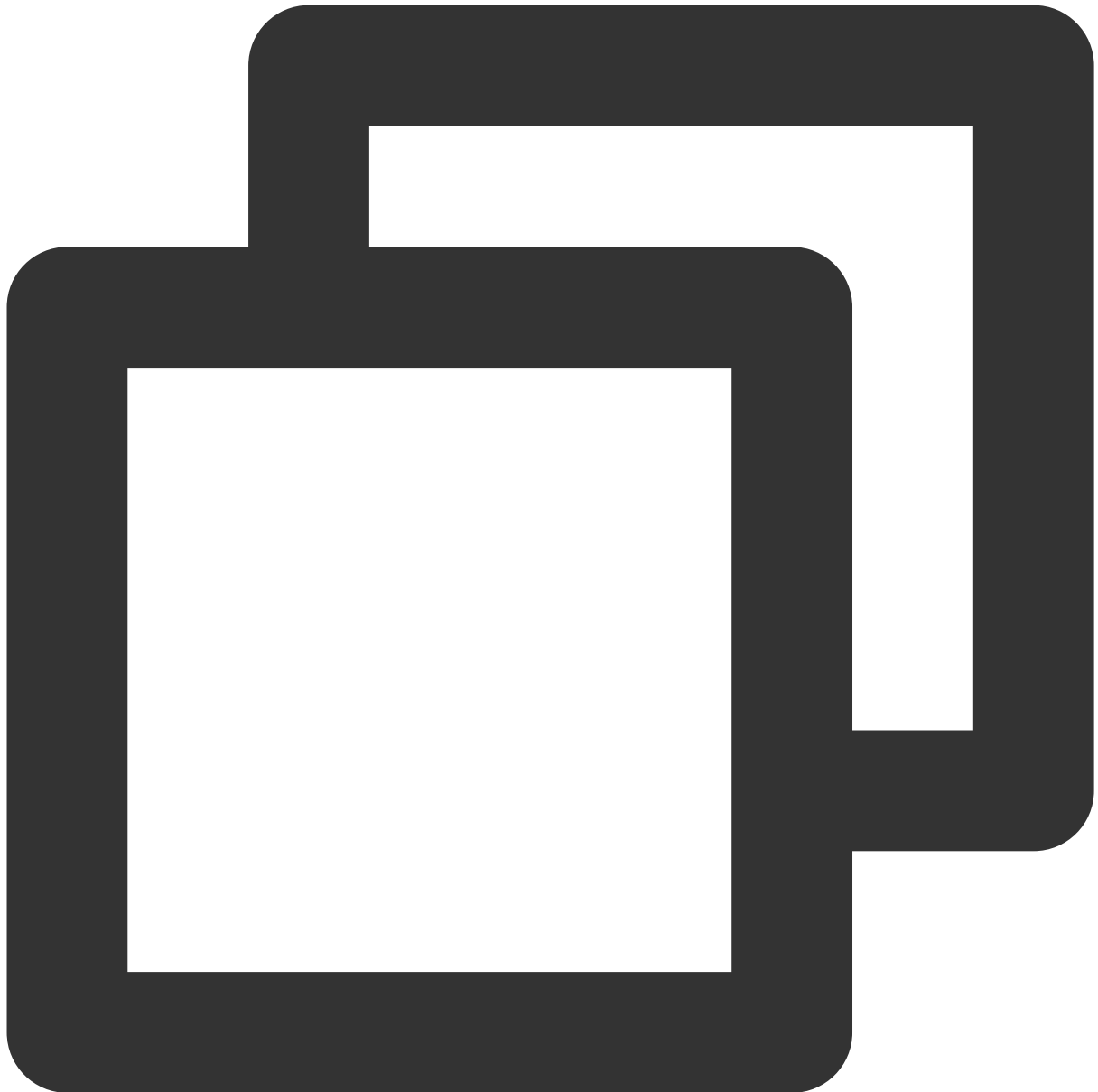
## Prerequisites

Node.js version: Node.js  $\geq 16.19.1$  (we recommend using the official LTS version, please match the npm version with the node version).

Modern browser, [supporting WebRTC APIs](#).

## Download Demo

1. Open the Terminal, copy and paste the sample command to clone the repository.



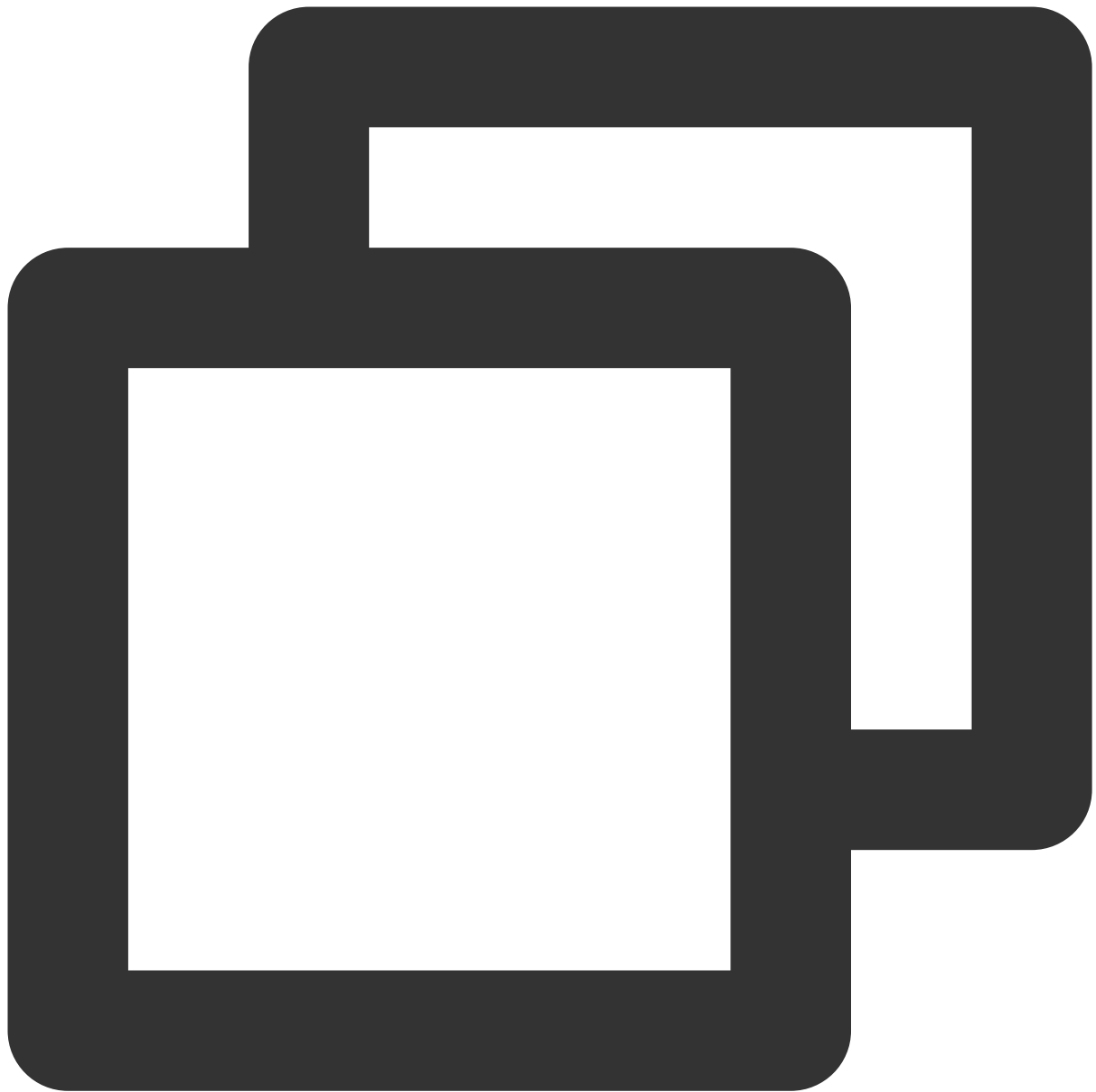
```
git clone https://github.com/Tencent-RTC/TUIRoomKit.git
```

2. Install dependencies.

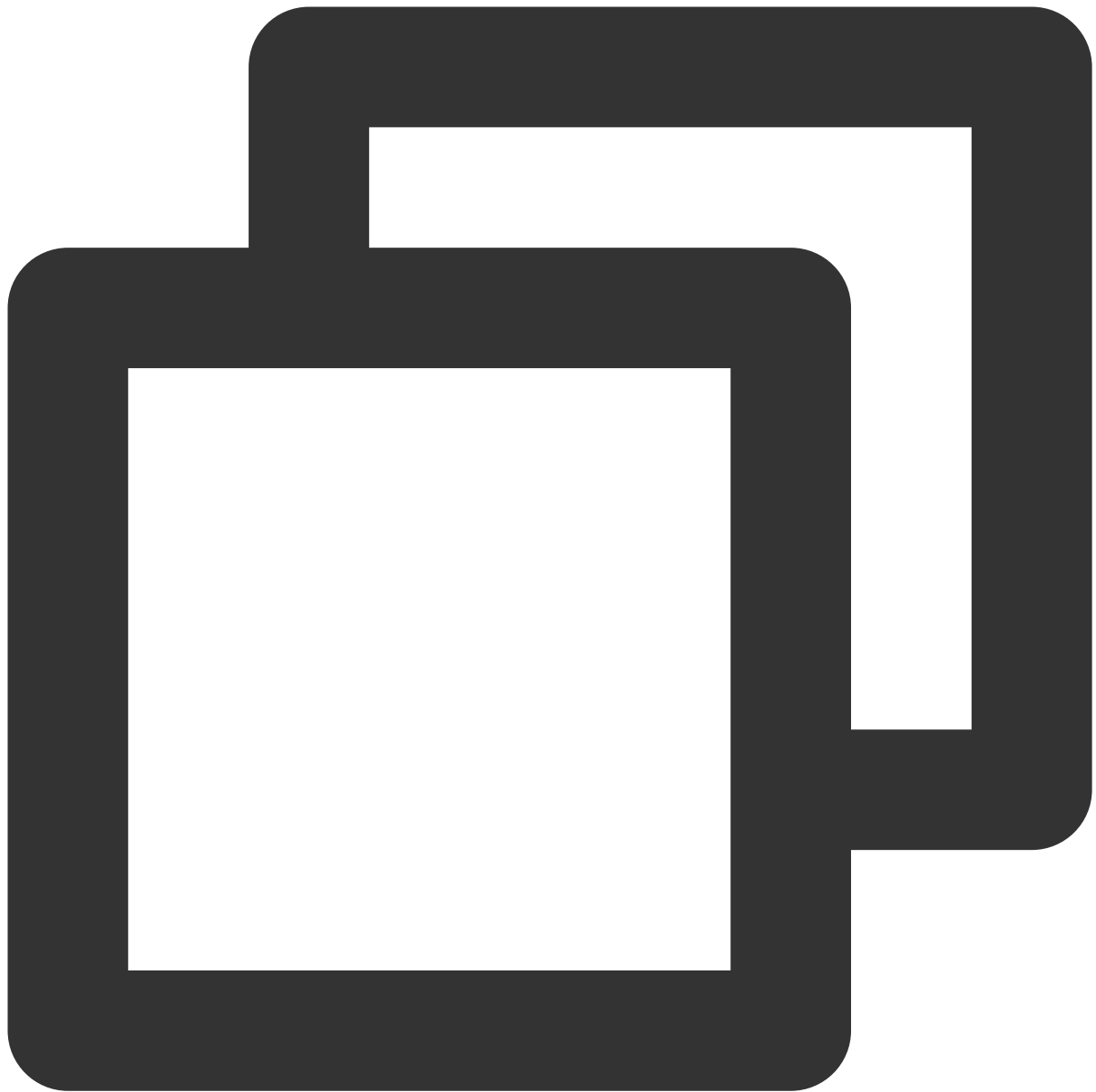
Vue3

Vue2

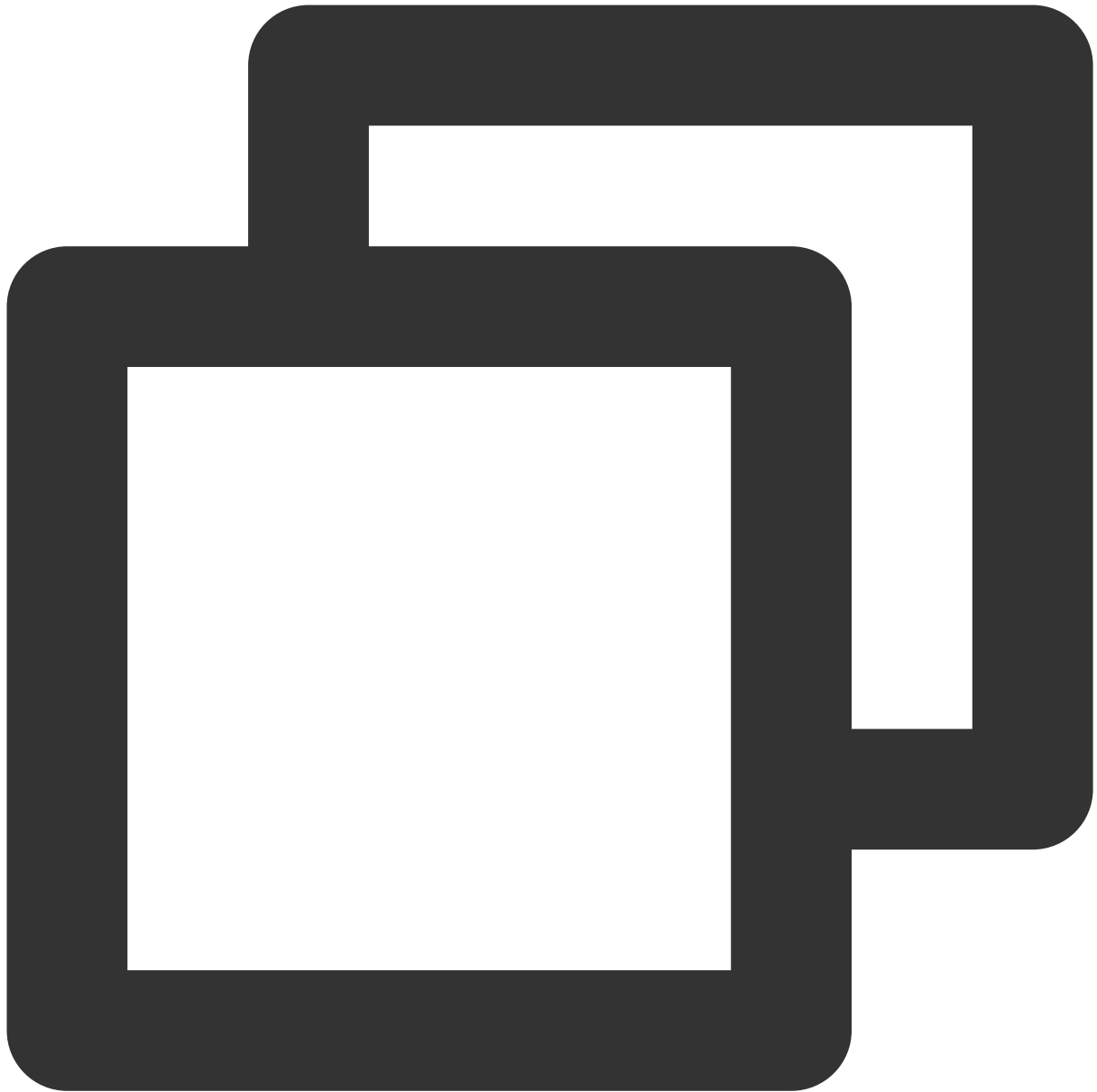




```
cd ../TUIRoomKit/Electron/example/vue3
```



```
cd ../TUIRoomKit/Electron/example/vue2
```



```
npm install
```

## Configure Demo

1. [Activate the TUIRoomKit service](#), get the **SDKAppID** and **SDKSecretKey**.

### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

On-cloud recording	Disabled
Relay to CDN	Disabled
Callbacks	Disabled
Advanced permission control	Disabled

#### Products

[Quickly run sample demo in 3 steps](#)

##### Conference

Edition	<a href="#">Conference : Trial</a>
Expiration time	2024-05-01
Auto-renewable	--

[Buy package](#)[Integrate](#)

2. Open the **TUIRoomKit/Electron/example/vue3/packages/renderer/src/config/basic-info-config.js** file and enter the **SDKAppID** and **SDKSecretKey** you got when you activated the service:

```
/*
 * @Description: Basic information configuration for TUIRoomKit applications
 */

import LibGenerateTestUserSig from './lib-generate-test-usersig-es.min';

/**
 * Tencent Cloud SDKAppId, which should be replaced with user's SDKAppId.
 * Enter Tencent Cloud TRTC [Console] (https://console.cloud.tencent.com/trtc ) to create an application,
 * and you will see the SDKAppId.
 * It is a unique identifier used by Tencent Cloud to identify users.
 */

export const SDKAPPID = 0;

/**
 * Encryption key for calculating signature, which can be obtained in the following steps:
 *
 * Step1. Enter Tencent Cloud TRTC [Console](https://console.cloud.tencent.com/rav ),
 * and create an application if you don't have one.
 * Step2. Click your application to find "Quick Start".
 * Step3. Click "View Secret Key" to see the encryption key for calculating UserSig,
 * and copy it to the following variable.
 *
 * Notes: this method is only applicable for debugging Demo. Before official launch,
 * please migrate the UserSig calculation code and key to your backend server to avoid
 * unauthorized traffic use caused by the leakage of encryption key.
 * Document: https://intl.cloud.tencent.com/document/product/647/35166#Server
 */

export const SDKSECRETKEY = '';
```

### Note :

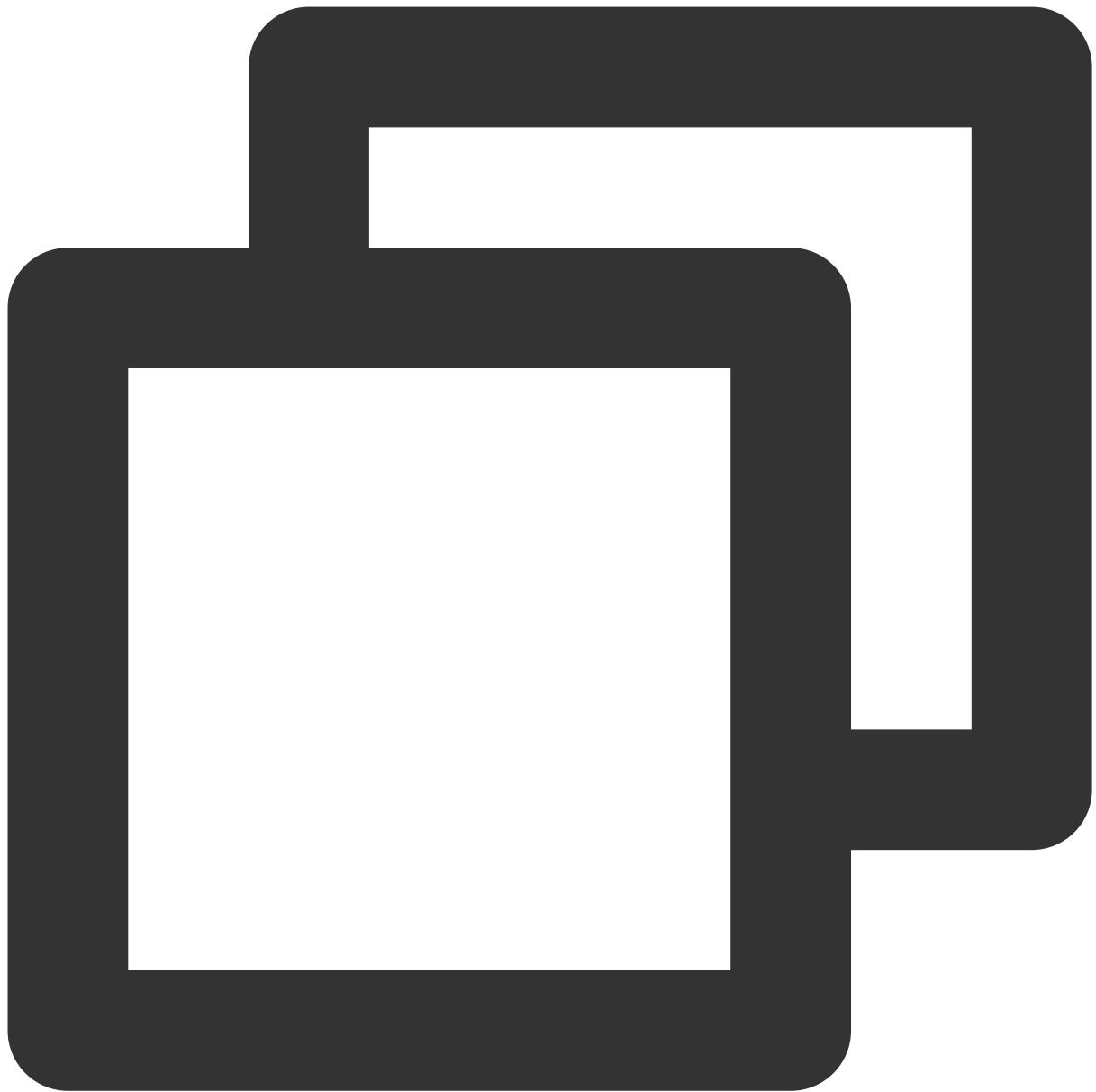
For Vue2 projects, open the **TUIRoomKit/Electron/example/vue2/src/config/basic-info-config.js** file and enter the **SDKAppID** and **SDKSecretKey** you got when you activated the service.

## Run Demo

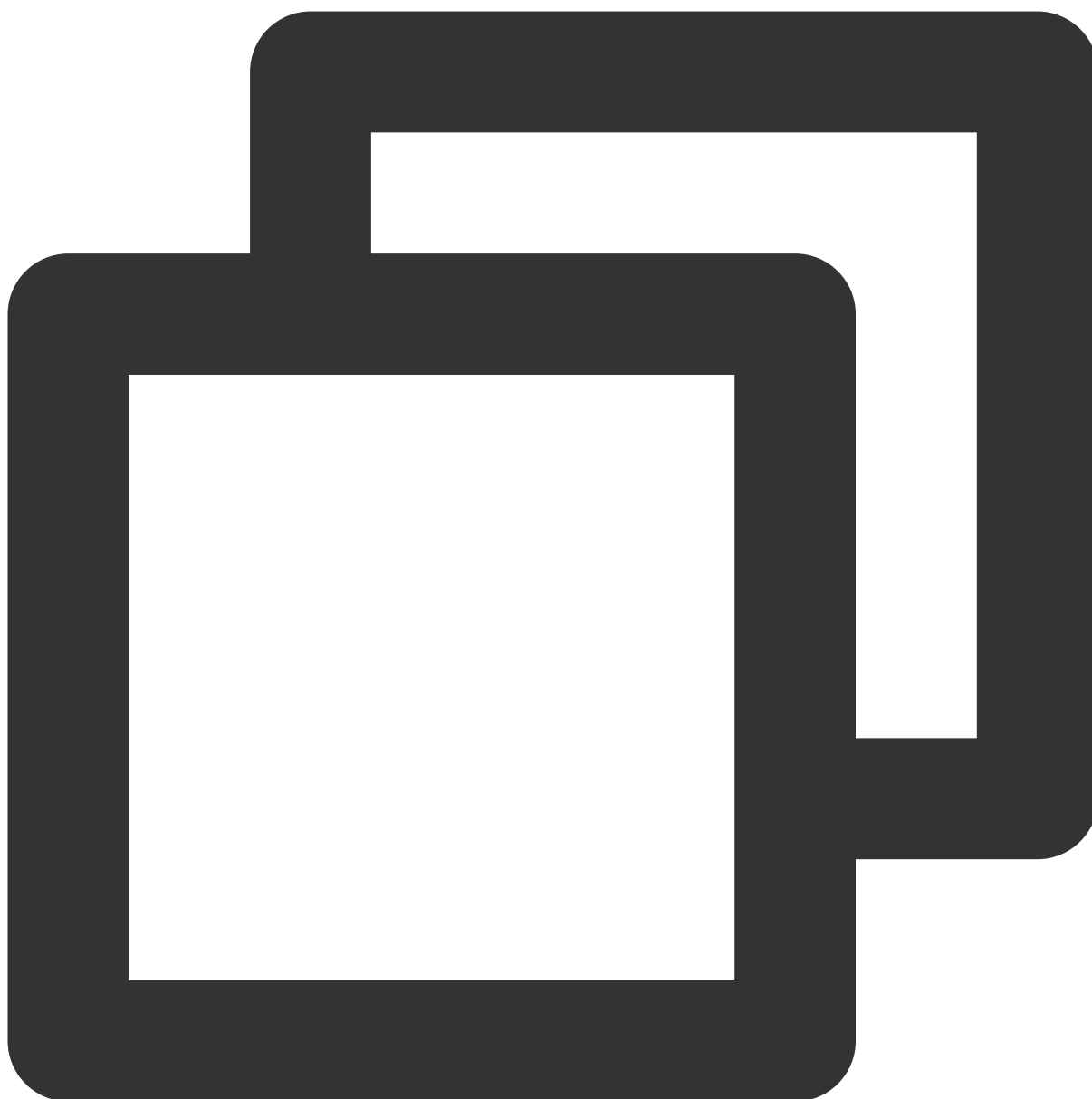
1. Run Demo by typing the command in the terminal.

Vue3

Vue2



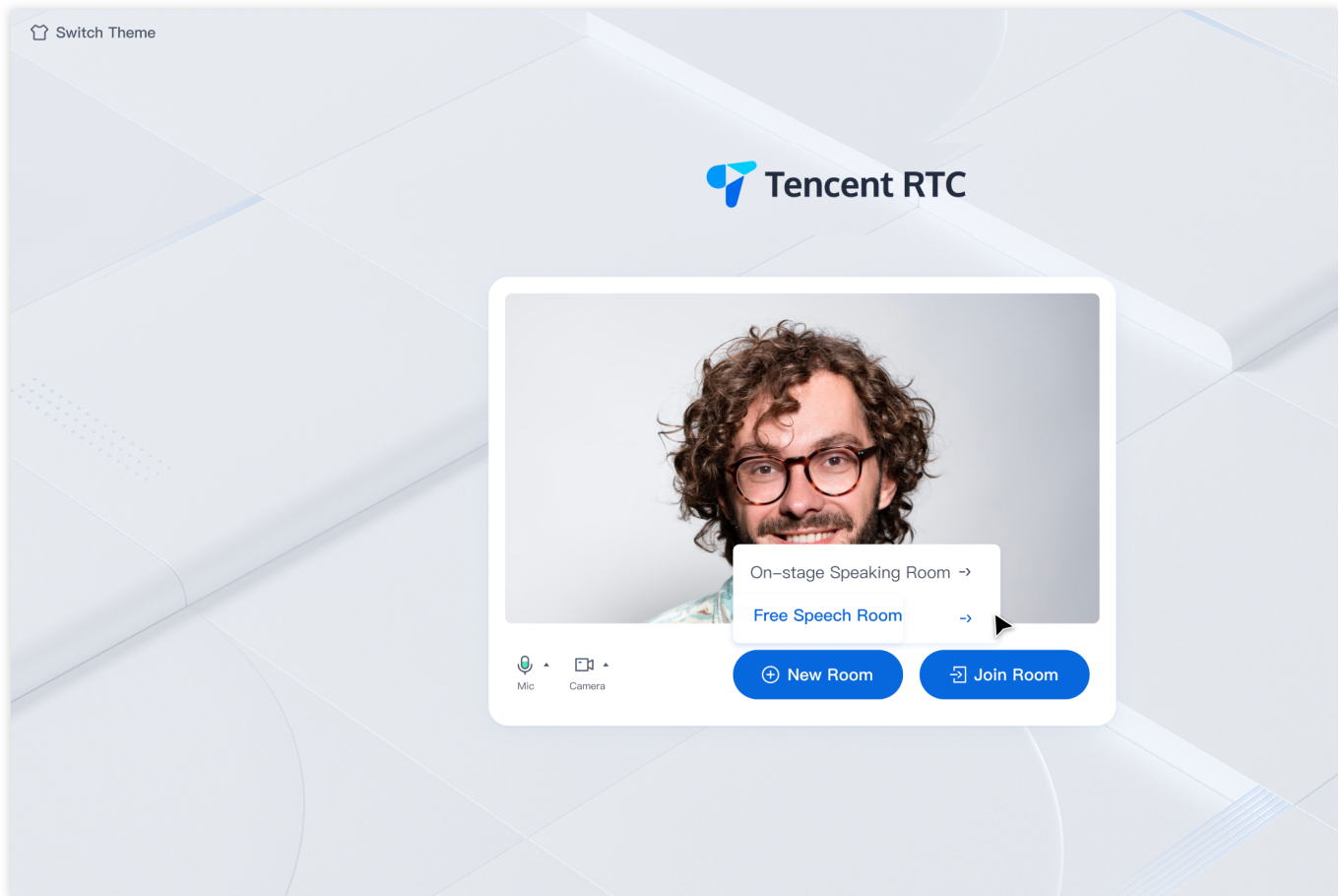
```
# cd TUIRoomKit/Electron/example/vue3  
npm run dev
```



```
# cd TUIRoomKit/Electron/example/vue2  
npm run start
```

**Note :**

For local environment, please access under localhost protocol, please refer to [the description of network access protocol](#).

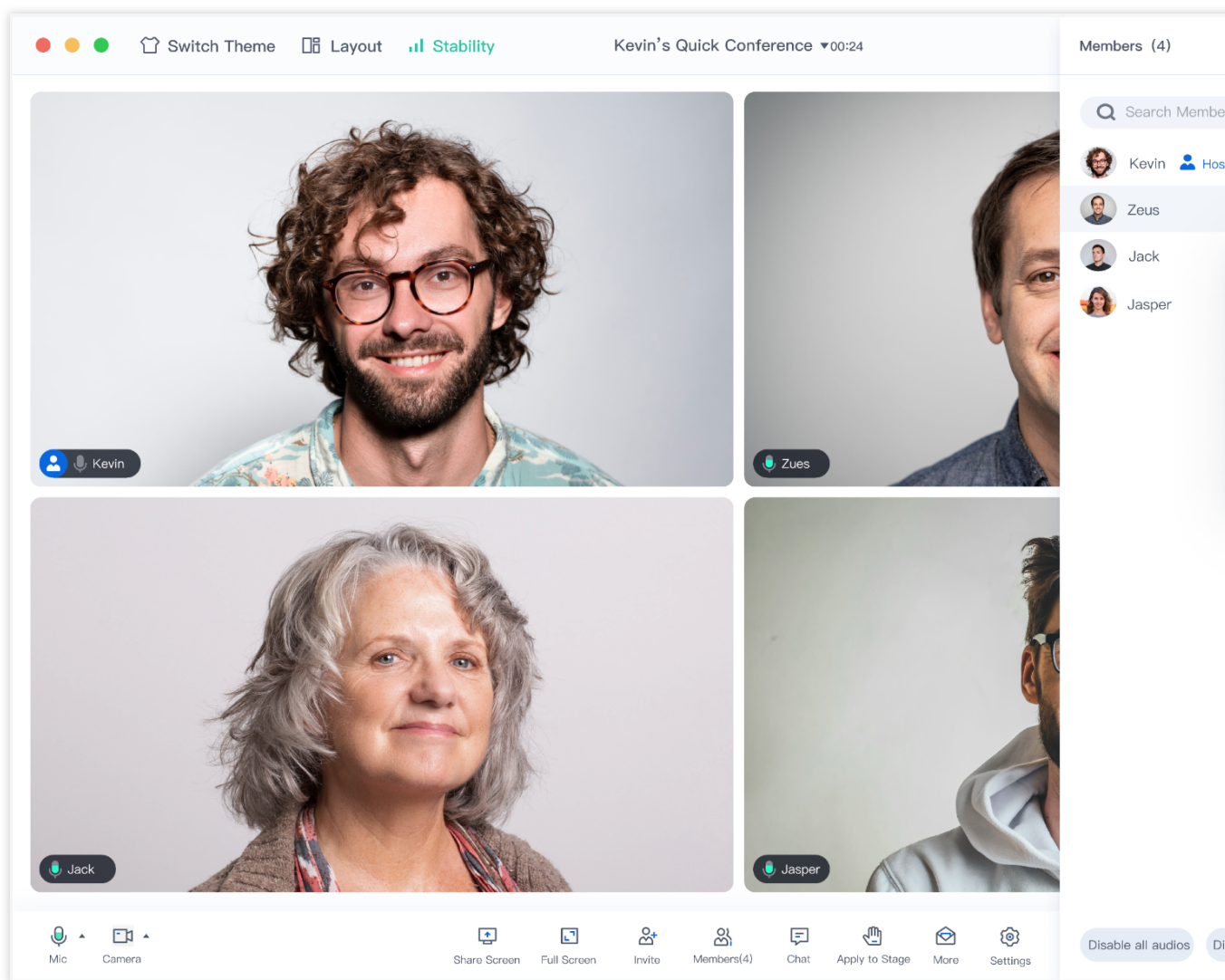


## Create your first conference

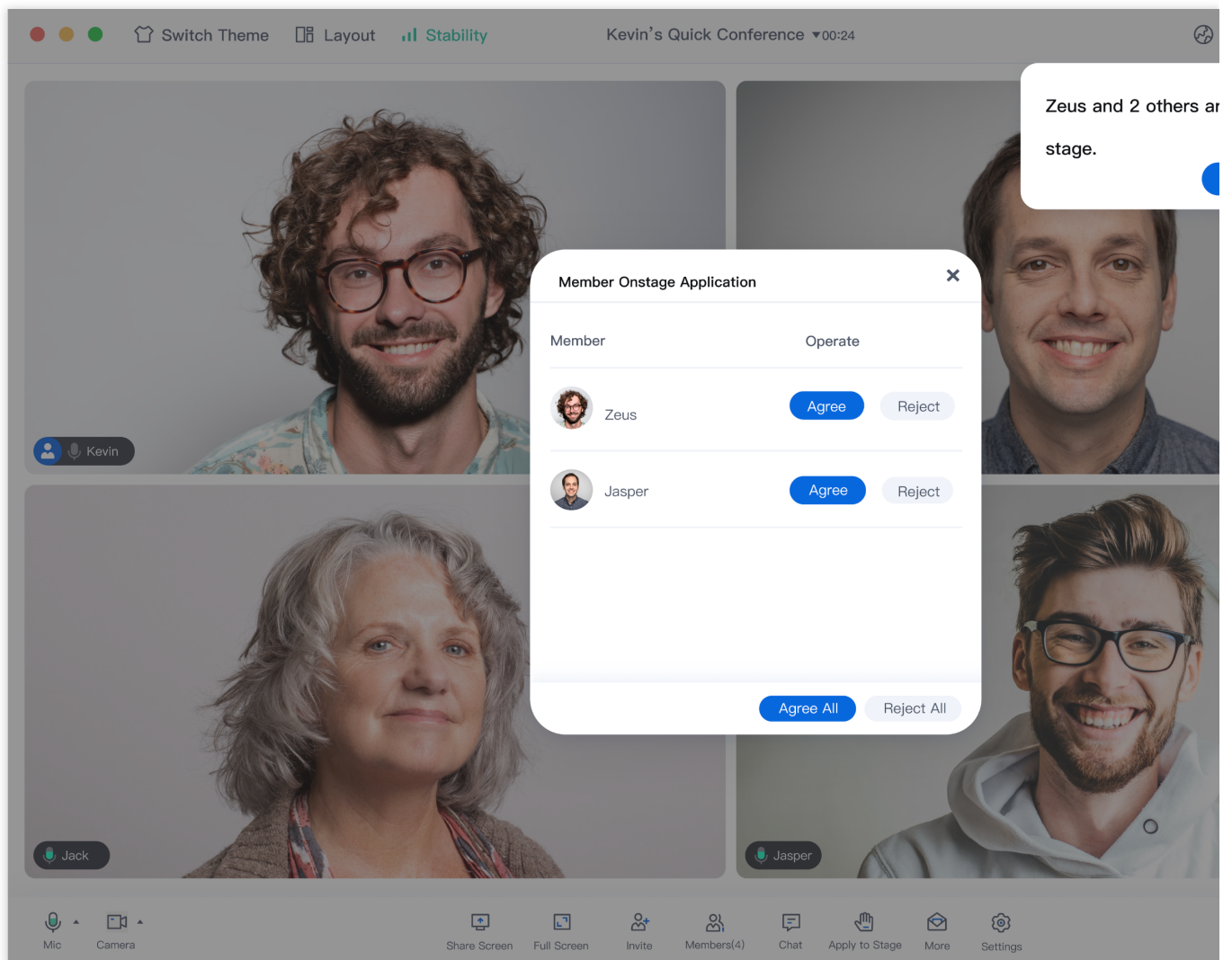
Click on the **New Room** button to create your first meeting room. The room types are **On-stage Speaking Room** or **Free Speech Room**.

1. Free speech room



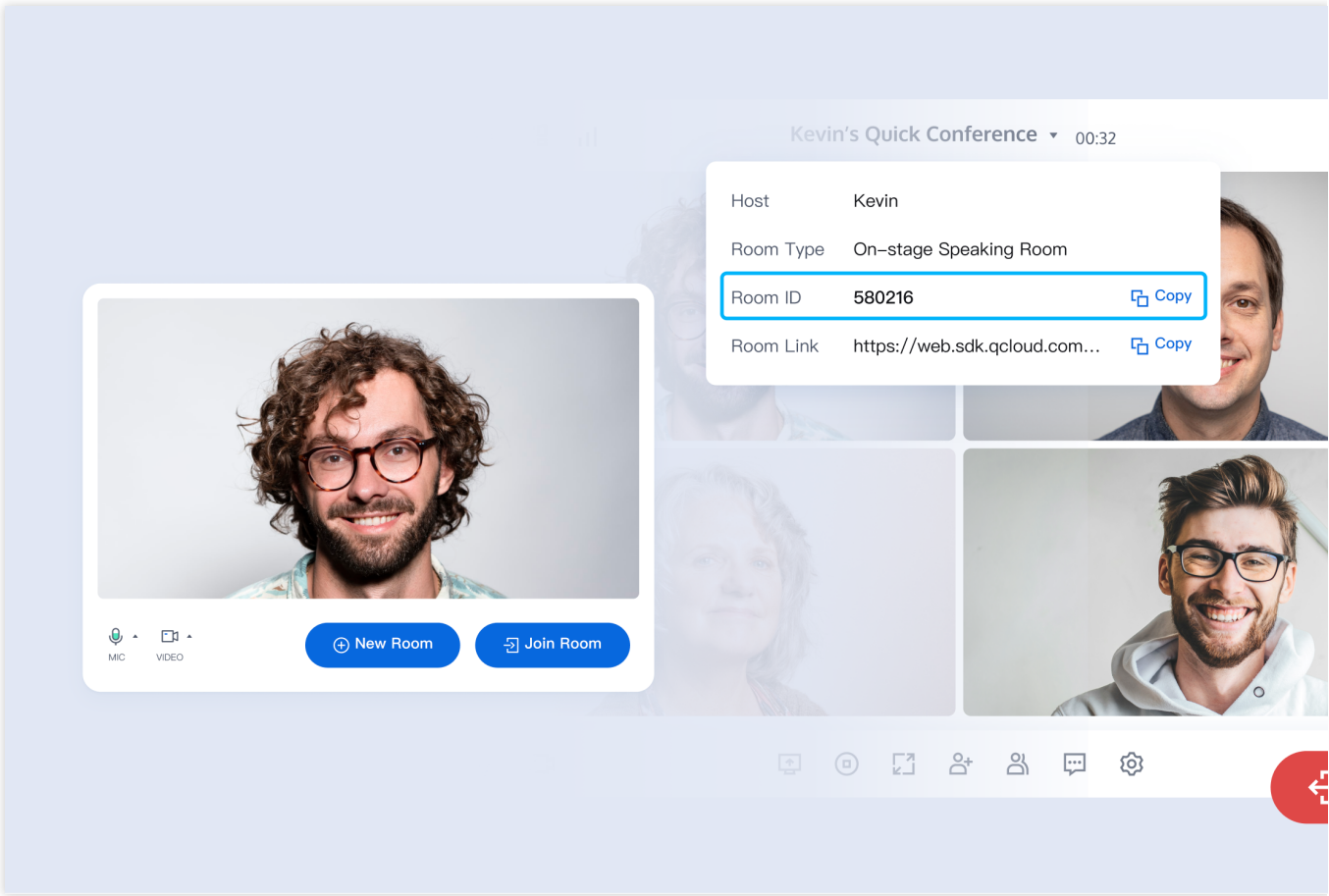


## 2. On-stage Speaking Room



## Join conference

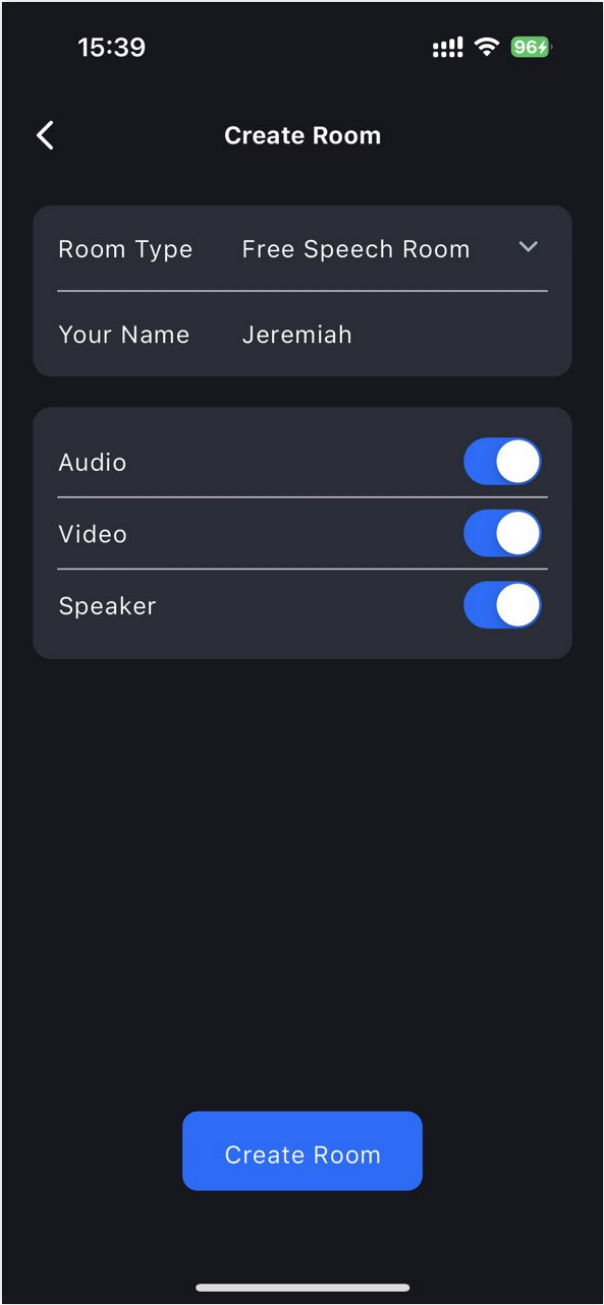
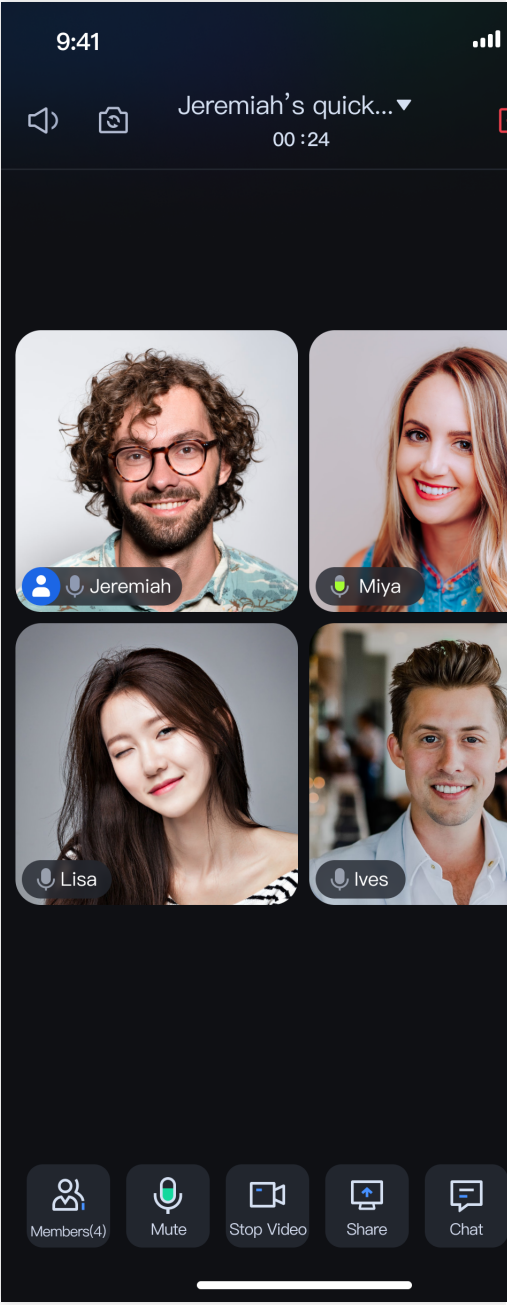
Participants can join a meeting created by the meeting host by filling in the corresponding `RoomId` .



# Flutter

Last updated : 2024-08-15 17:06:16

This document mainly introduces how to quickly run through the **Conference (TUIRoomKit) sample project** and experience a high-quality multi-person video conference. By following this document, you can run through the demo within 10 minutes and ultimately experience a multi-person video conference feature with a complete UI interface.

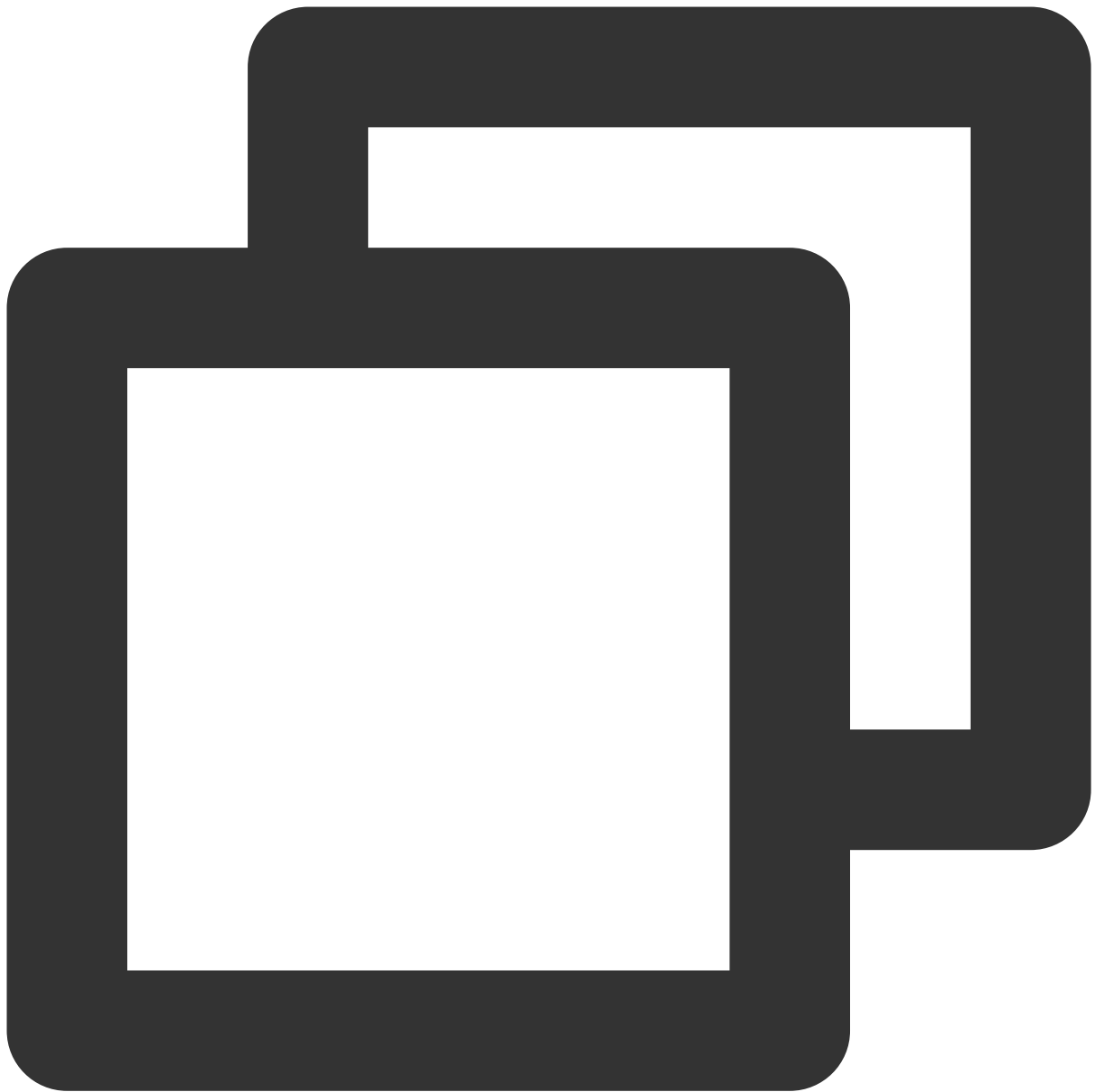
Conference creation page	Conference main page
	

## Prerequisites

Platform	Version
Flutter	3.0.0 or later.
Android	Android 4.1 (SDK API level 16) or later (Android 5.0 (SDK API level 21) or later is recommended). Android Studio 3.5 or later (Gradle 3.5.4 or later). Android 4.1 or later mobile devices.
iOS	iOS 12.0 or later.

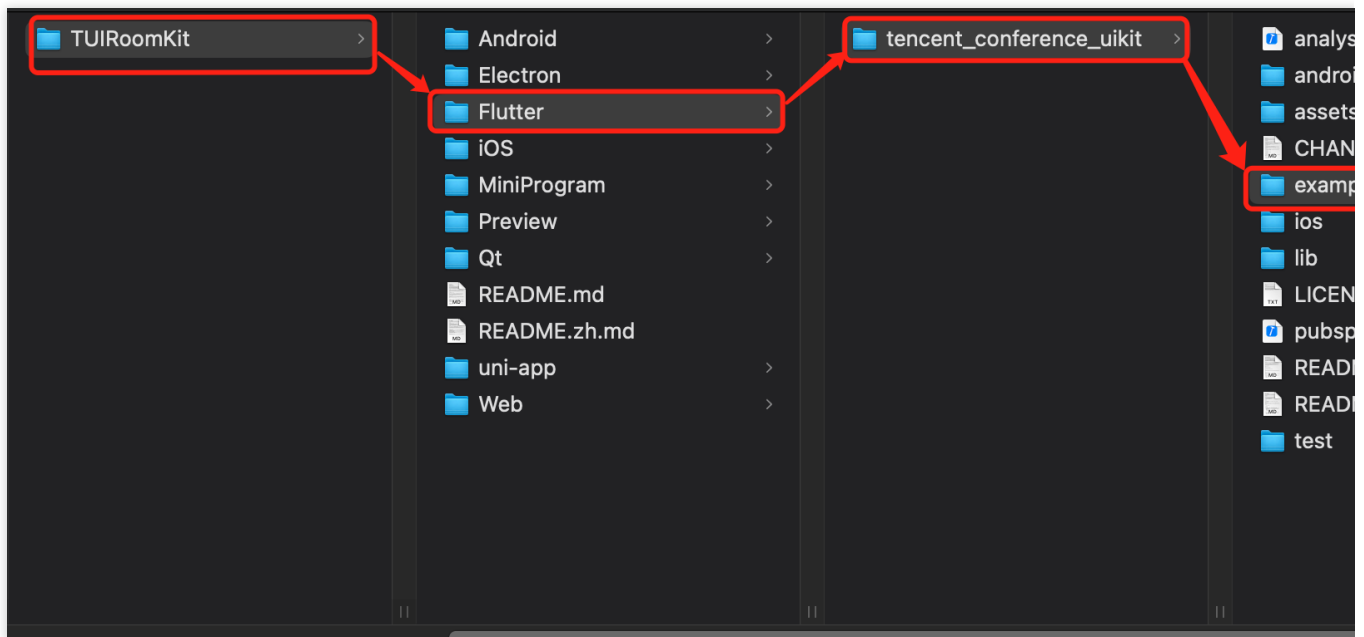
## Download the Demo

1. Download the [TUIRoomKit Demo](#) source code from GitHub, or directly execute the following command in the command line:



```
git clone https://github.com/Tencent-RTC/TUIRoomKit.git
```

2. Open the TUIRoomKit Flutter's example project with Android Studio or VSCode. The following process will take VSCode as an example:



## Configure the Demo

1. [Activate the Conference services](#), to obtain the **SDKAppID** and **SDKSecretKey**.

### Application Overview

#### Basic Information

Application name	test1	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-17 16:20:52
Description	--	Region	Singapore
Status	Enabled	<a href="#">More</a>	

#### Advanced Features

- On-cloud recording
- Relay to CDN
- Callbacks
- Advanced permission control

#### Products

[Quickly run sample demo in 3 steps >](#)

##### Conference

Edition: [Conference : Trial >](#)

Expiration time: 2024-05-01

Auto-renewable: --

[Buy package](#)[Integrate](#)

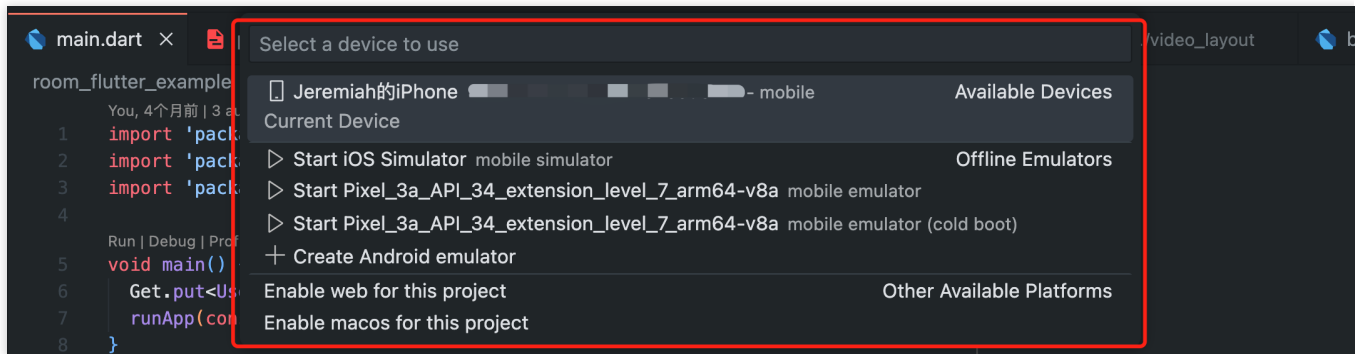
2. Open the project and find the `example/lib/debug/generate_test_user_sig.dart` file within the project. Enter the **SDKAppID** and **SDKSecretKey** obtained from Back into it:

```
class GenerateTestUserSig {  
  /// Tencent Cloud `SDKAppID`. Set it to the `SDKAppID` of your account.  
  ///  
  /// You can view your `SDKAppID` after creating an application in the [TRTC console](https://console.cloud.tencent.com/trtc).  
  /// `SDKAppID` uniquely identifies a Tencent Cloud account.  
  static int sdkAppId = 0;  
  
  /// Signature validity period, which should not be set too short  
  /// <p>  
  /// Unit: second  
  /// Default value: 604800 (7 days)  
  static int expireTime = 604800;  
  
  /// Follow the steps below to obtain the key required for UserSig calculation.  
  ///  
  /// Step 1. Log in to the [TRTC console](https://console.cloud.tencent.com/trtc), and create an application if you don't.  
  /// Step 2. Find your application, click "Application Info", and click the "Quick Start" tab.  
  /// Step 3. Copy and paste the key to the code, as shown below.  
  ///  
  /// Note: this method is for testing only. Before commercial launch, please migrate the UserSig calculation code and key to the production environment.  
  /// Reference: https://cloud.tencent.com/document/product/647/17275#Server-Side-Signature-Verification  
  static String secretKey = '';  
  static genTestSig(String userId) {  
    // ...  
  }  
}
```

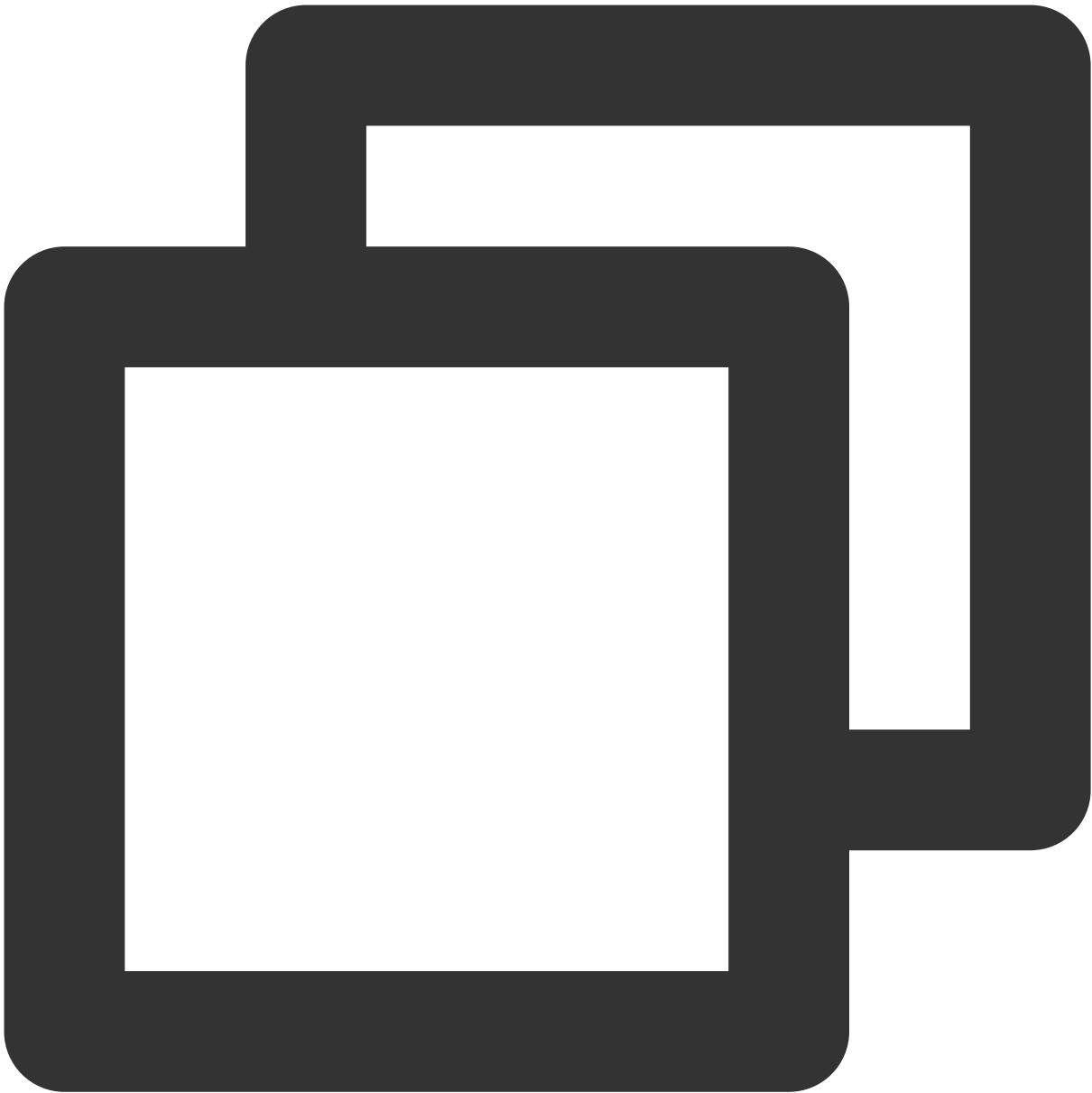


## Running the Demo

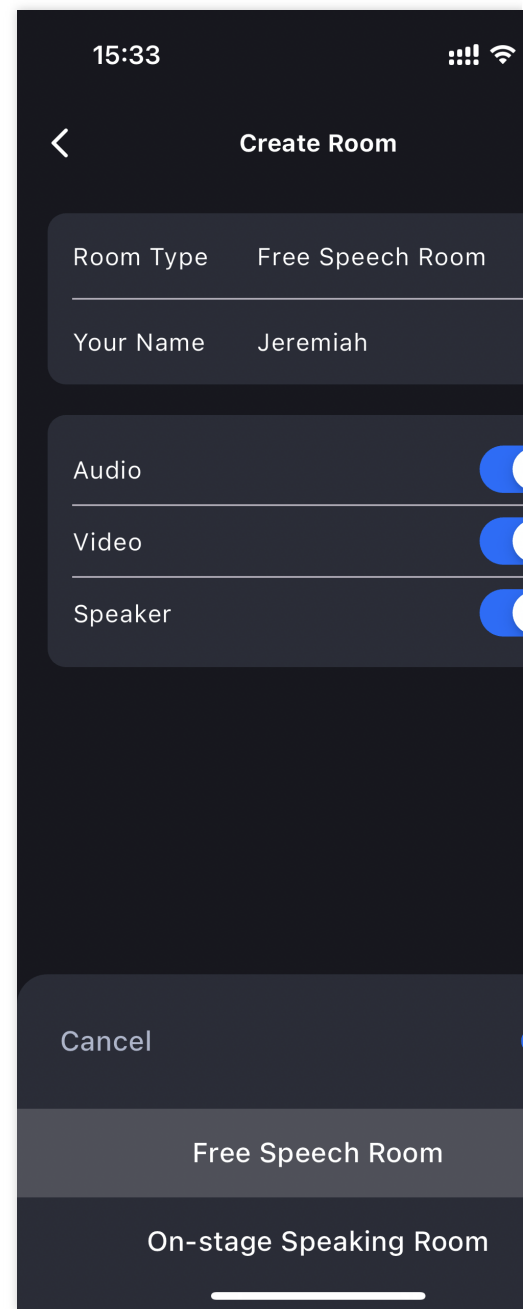
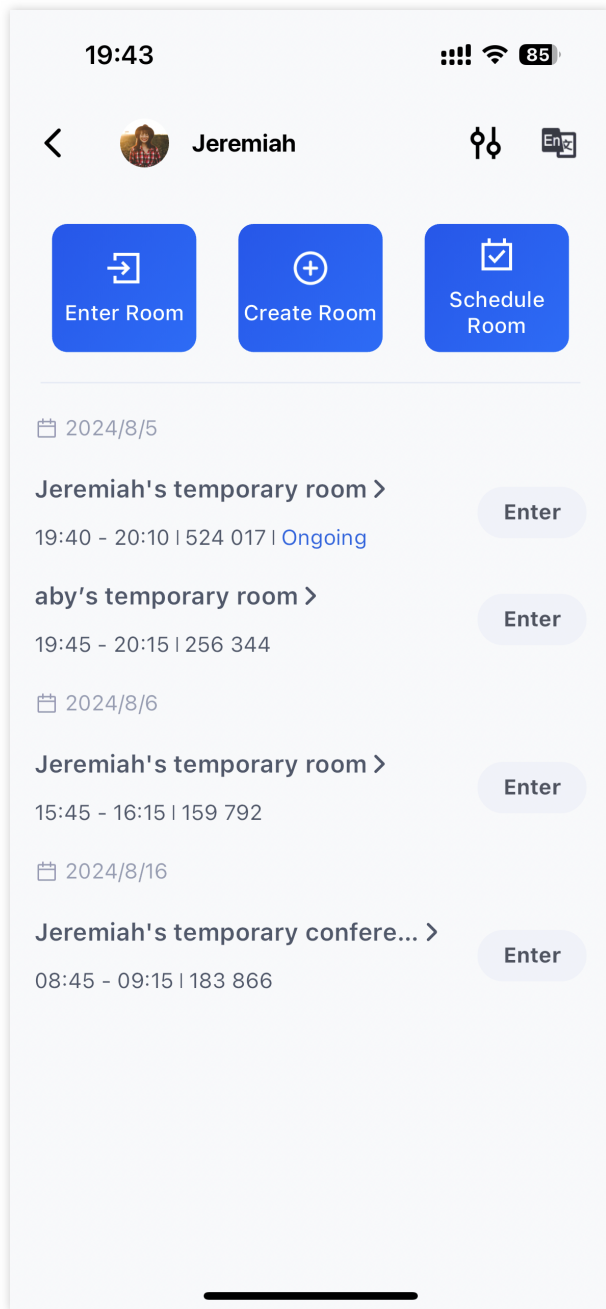
1. Open `example/lib/main.dart` with VSCode, and click the device connection button at the bottom right. Choose the device you wish to run from the pop-up box at the top. After selecting, click the run button at the top right to run the TUIRoomKit Flutter Demo on the device.



2. You can also run the following command in the example directory to run it on your device.



flutter run	
APP main page	Conference creation page

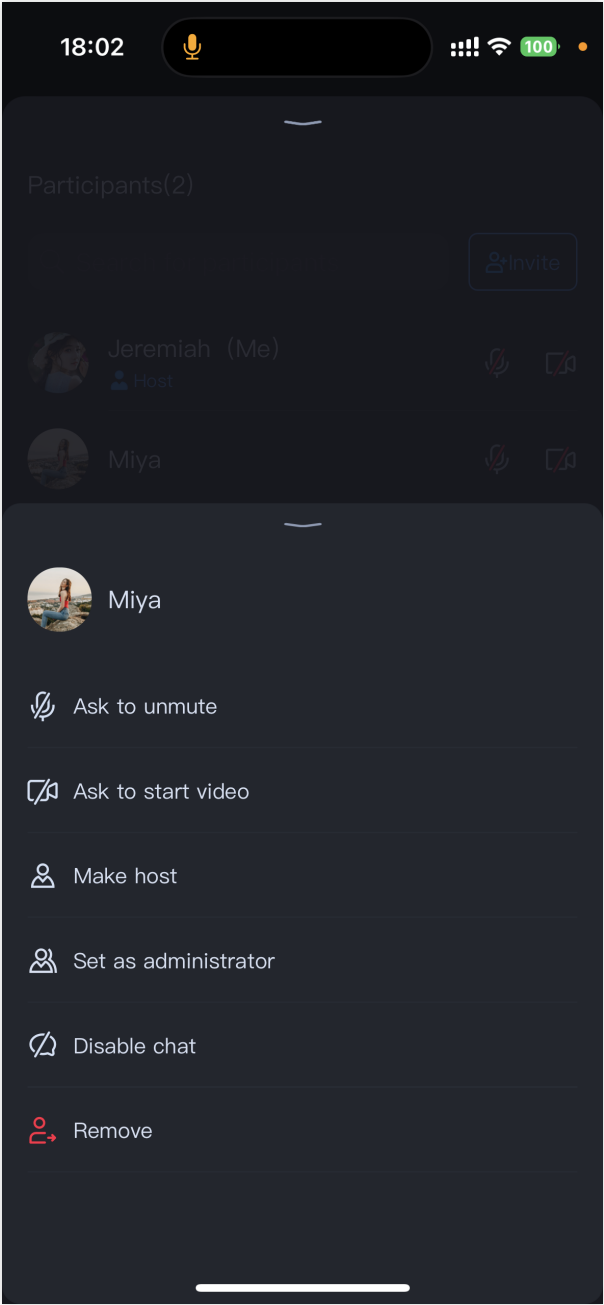
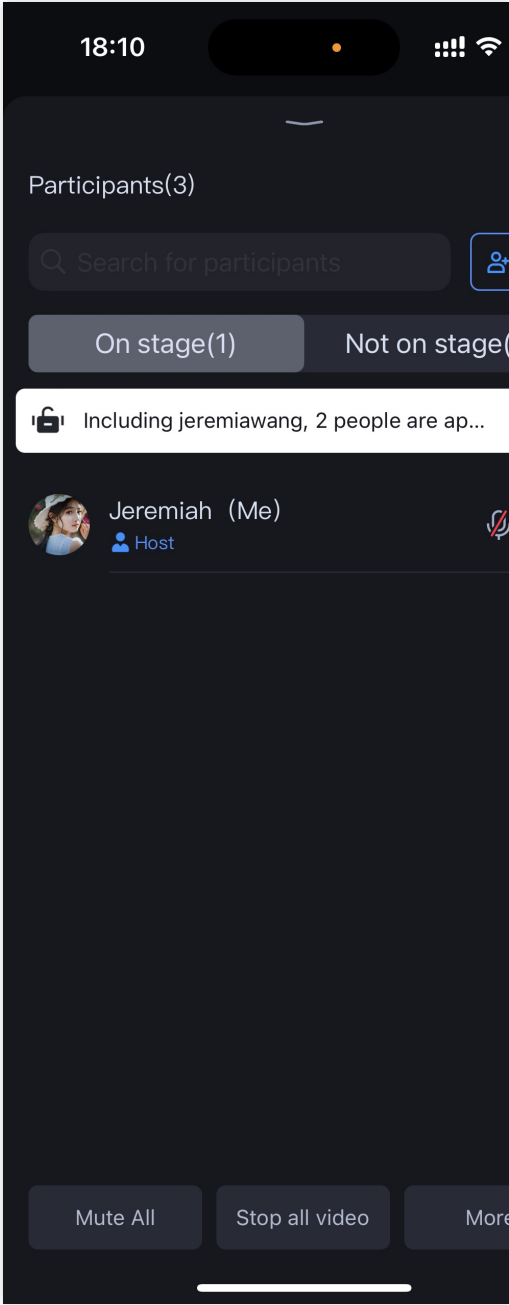


## Create your first conference

Click on the **Create Room** button to create your first meeting room. The room types are **On-stage Speaking Room** and **Free Speech Room**.

**Free Speech Room:** Regular users can freely speak and have the liberty to turn their microphones and cameras on or off.

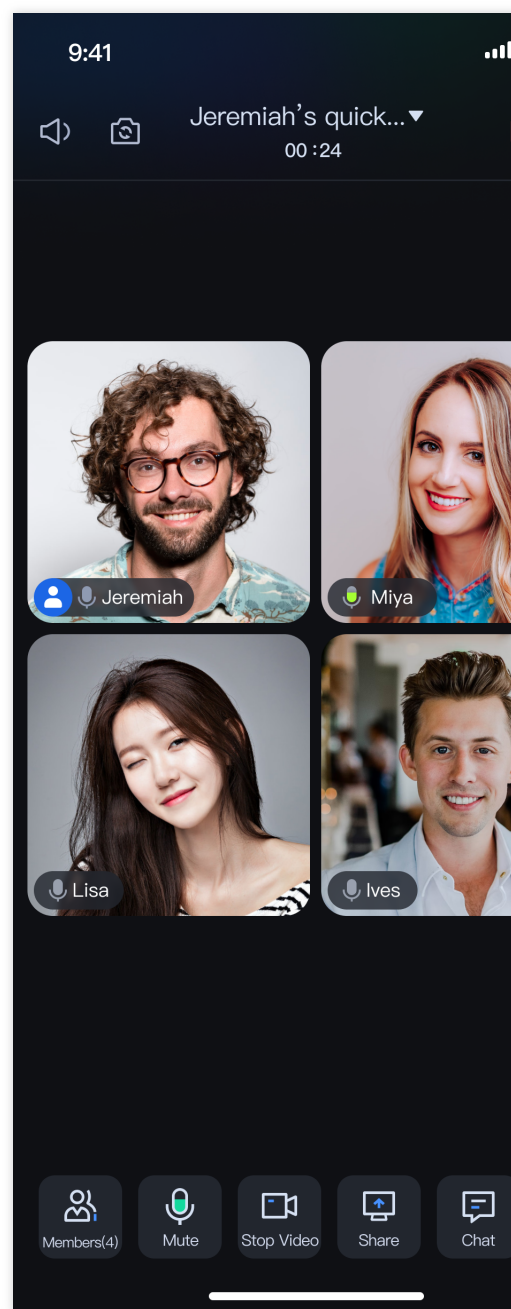
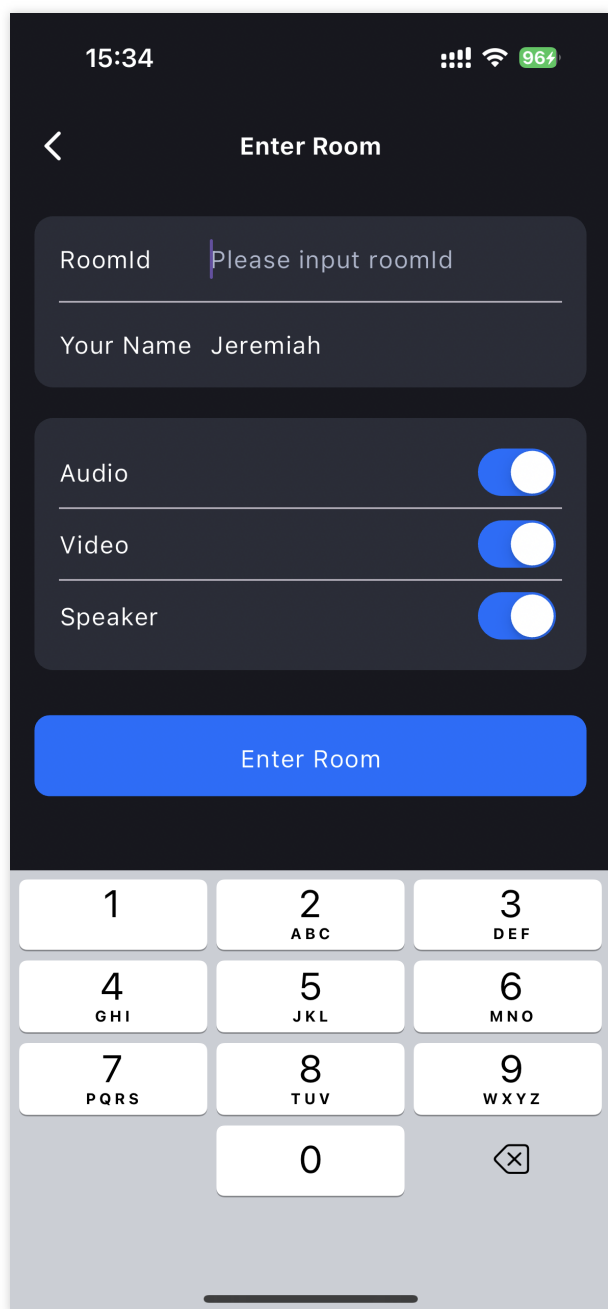
**On-stage Speaking Room:** Only users on stage can freely turn their microphones and cameras on or off. Regular audience members can apply to become stage users by raising their hand.

Free speech room member management panel	On-stage speaking room member list panel
	

## Join conference

After clicking **Join Room**, participants can join the meeting created by the host by filling in the corresponding `RoomId` .

Join conference page	Conference main page



# Integration (TUIRoomKit)

## iOS

Last updated : 2024-05-13 10:43:56

This article will introduce how to complete the integration of the `TUIRoomKit` Component in the shortest time. By following this document, you will complete the following key steps within an hour and ultimately obtain an audio/video conference function with a complete UI interface.

## Environment preparation

iOS 13.0 and higher.

Xcode 12.0 and higher.

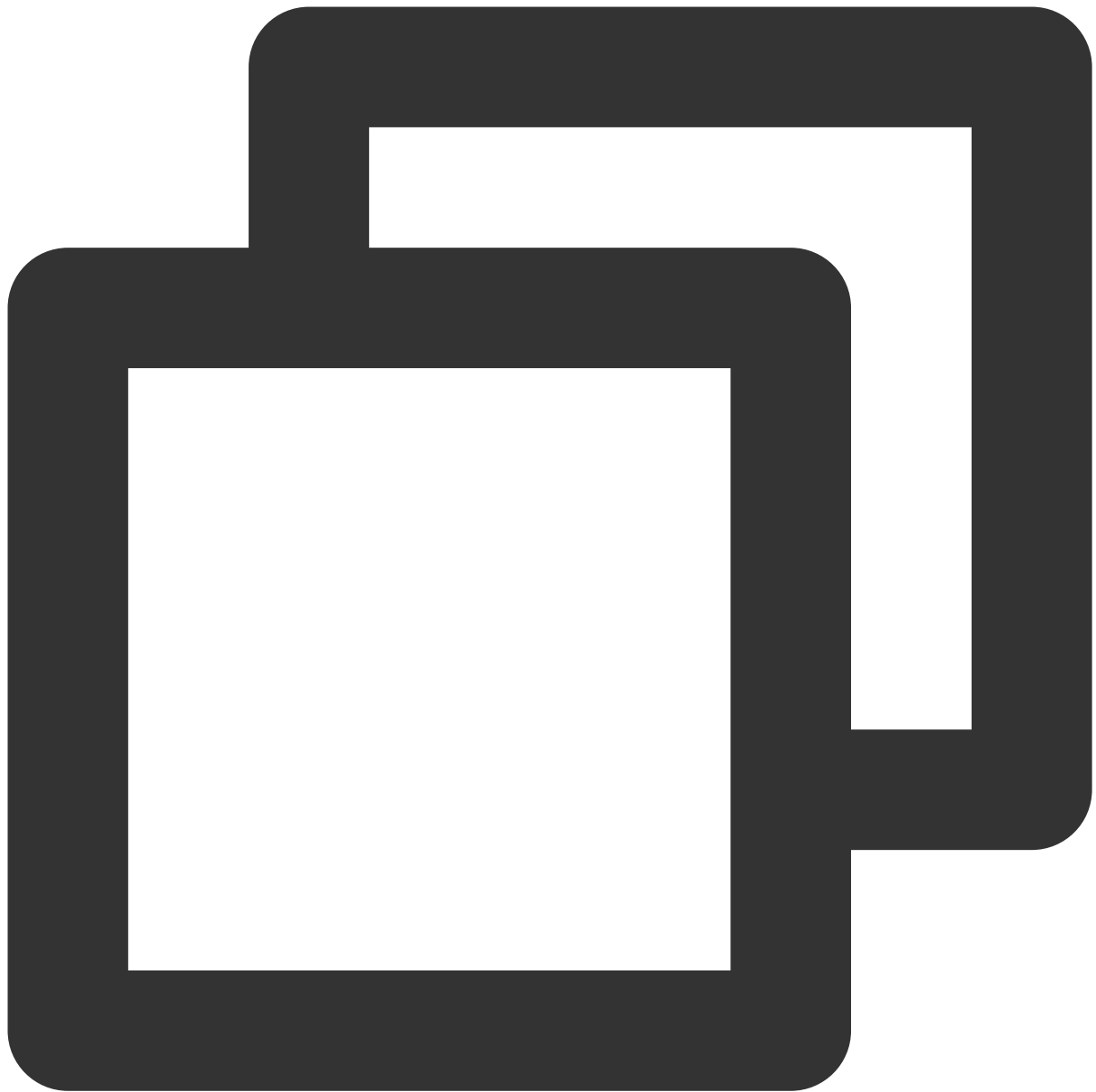
Swift 4.2 and higher.

## Step 1: Activate the service

Before initiating a meeting with TUIRoomKit, you need to activate the exclusive multi-person audio and video interaction service for TUIRoomKit on the console. For specific steps, please refer to [Activate Service](#).

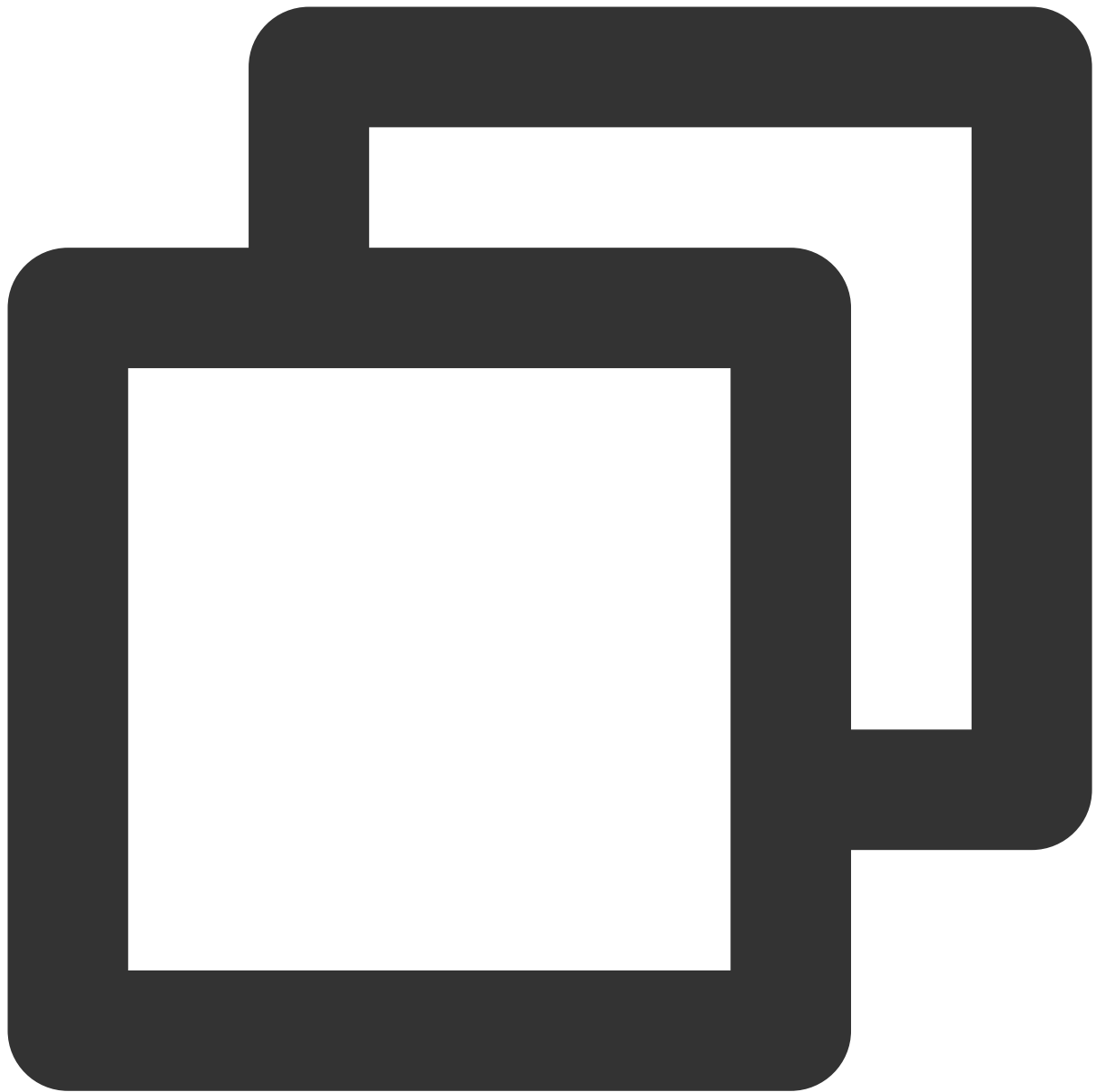
## Step 2: Integrate the TUIRoomKit Component

1. Add the following dependencies to your Podfile file.



```
pod 'TUIRoomKit'
```

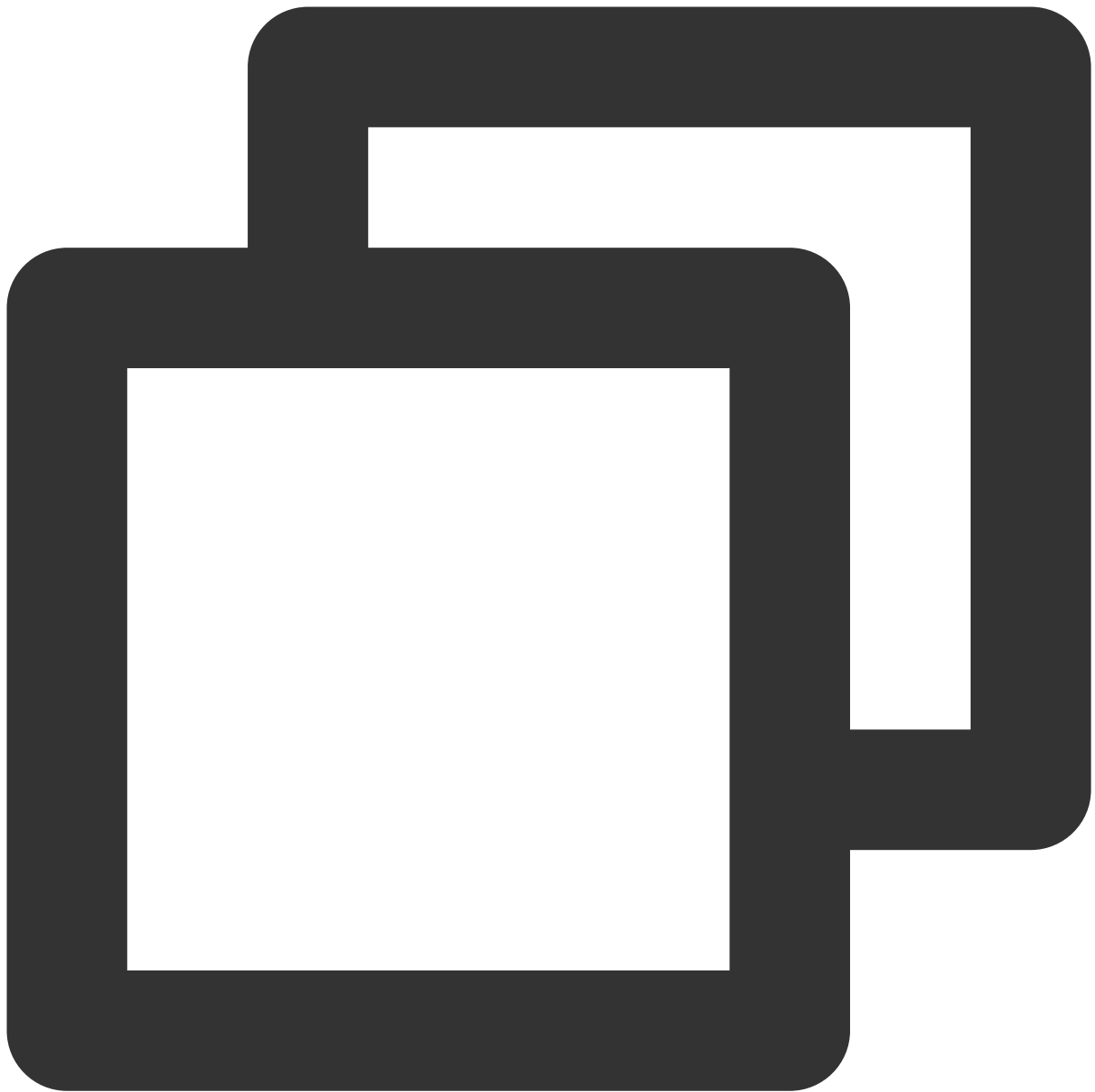
2. Execute the following command to install the Component.



```
pod install
```

**Note:** If you cannot install the latest version of TUIRoomKit, execute the following command to update the local CocoaPods repository list.

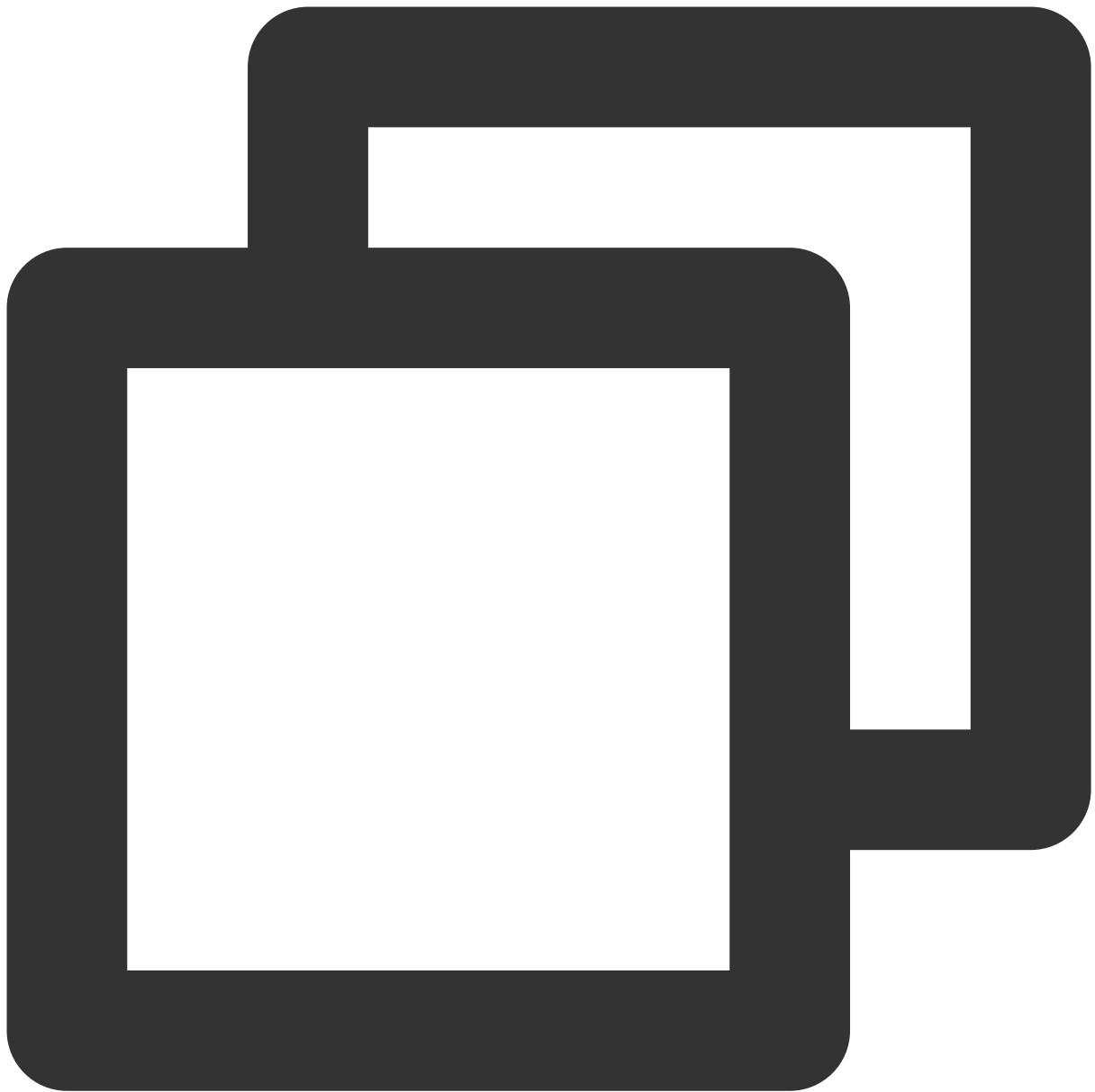




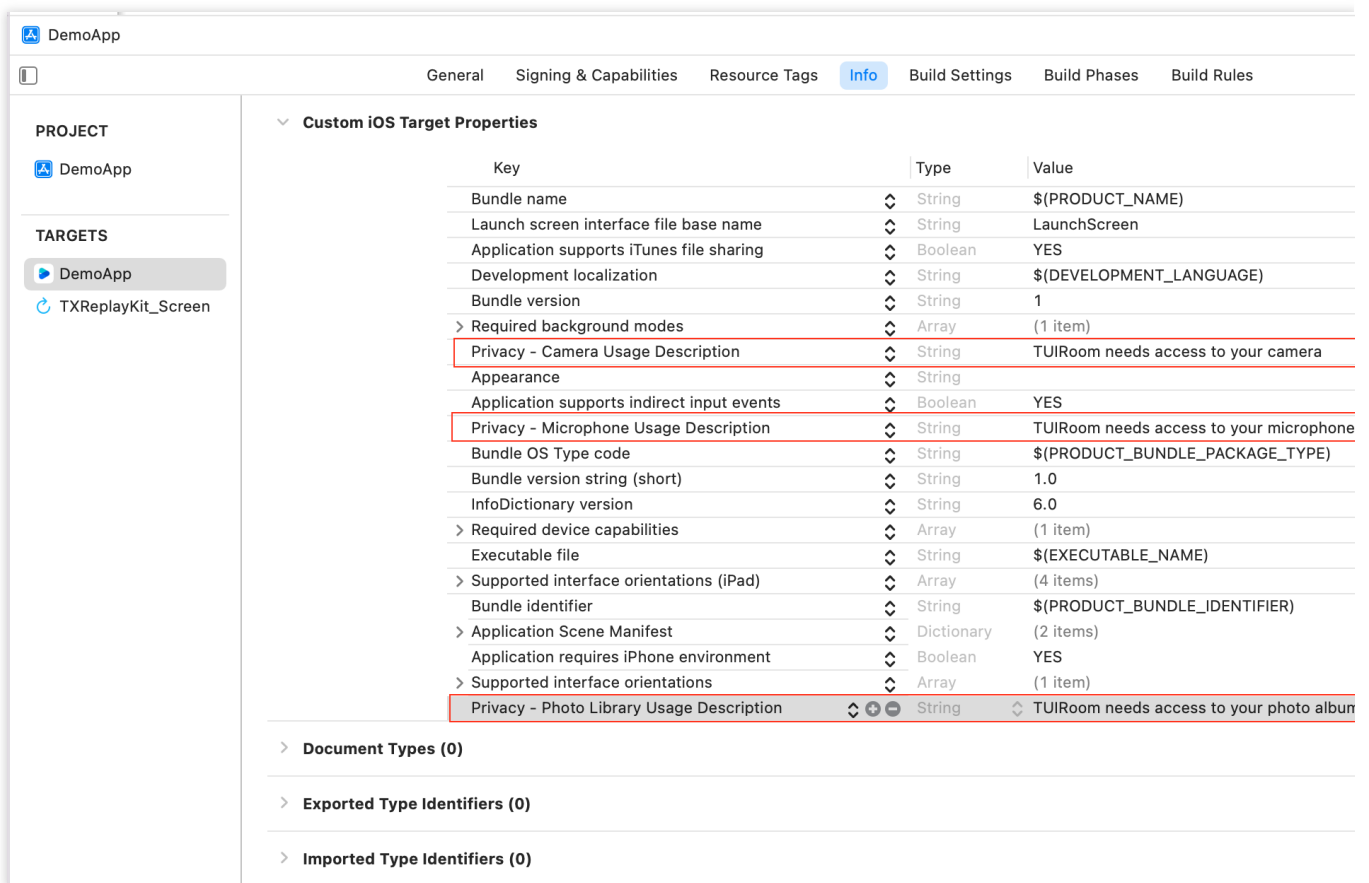
```
pod repo update
```

## Step 3: Project Configuration

To use the audio and video functions, you need to authorize the use of microphone, camera and photo album. Add the following items to the App's Info.plist, corresponding to the prompt messages for the microphone, camera, and photo album when the system pops up the authorization dialog box.



```
<key>NSCameraUsageDescription</key>  
<string>TUIRoom needs access to your Camera permission</string>  
<key>NSMicrophoneUsageDescription</key>  
<string>TUIRoom needs access to your Mic permission</string>  
<key>NSPhotoLibraryUsageDescription</key>  
<string>TUIRoom needs access to your Photo Library</string>
```



## Step 4: Log in

Add the following code to your project. Its function is to complete the initialization of TUI components by calling the relevant interfaces in TUICore. This step is very critical, because all functions of TUIRoomKit can only be used normally after successful login, so please be patient and check whether the relevant parameters are configured correctly:

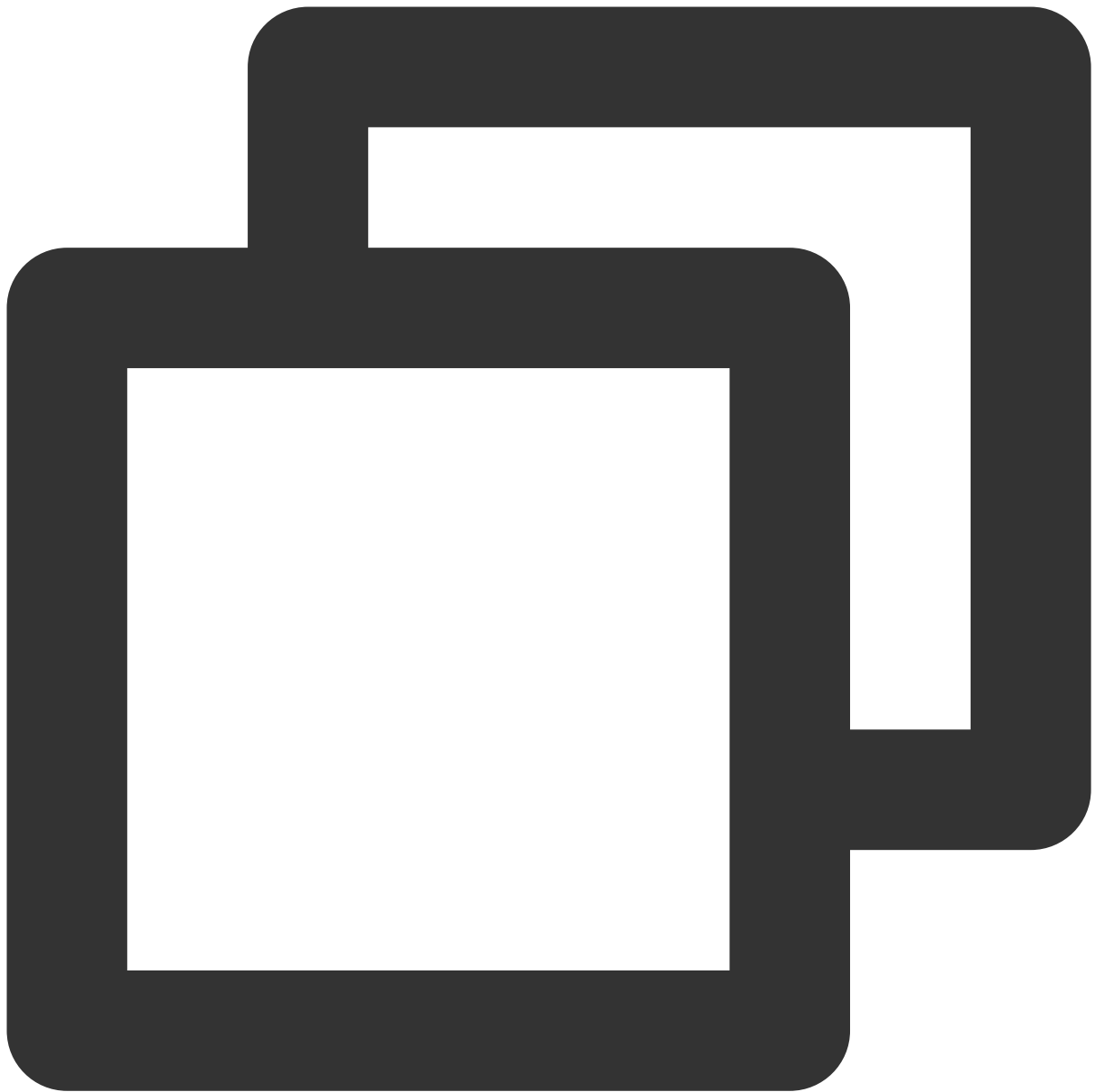
Swift

OC



```
import TUICore

TUILogin.login(1400000001,                                // Please replace with the SDKApp
               userID: "998",                               // Please replace with your UserI
               userSig: "xxxxxxxxxx") {                     // You can calculate a UserSig in
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \(code), error: \(message ?? "nil")")
}
```



```
#import "TUICore/TUILogin.h"

[TUILogin login:1400000001 // Please replace with the SDKAppID
             userID:@"998" // Please replace with your UserID
             userSig:@"xxxxxxxxxx" succ:^( // You can calculate a UserSig in t
                 NSLog(@"login,success");
             } fail:^(int code, NSString * _Nullable msg) {
                 NSLog(@"login,failed,code:%d,msg:%@",code,msg);
             }
];
```

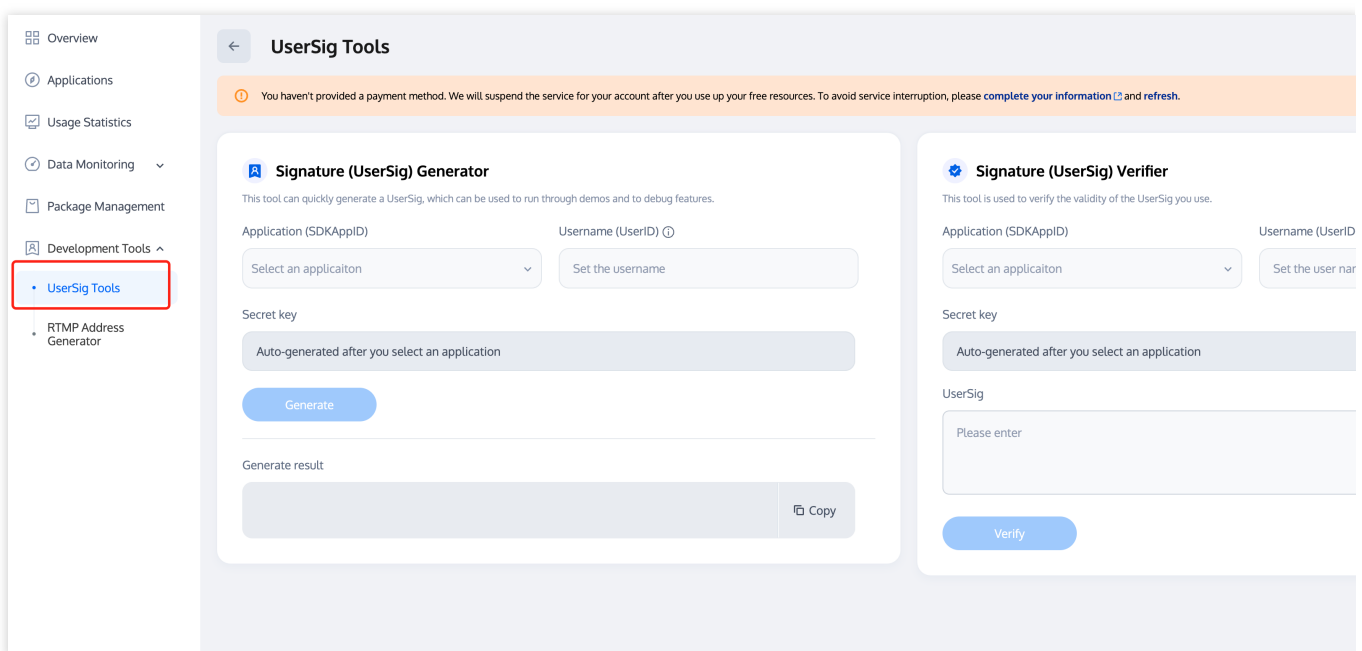
## Parameter Description

Here is a detailed introduction to the key parameters used in the login function:

**SDKAppID** : You have already obtained it in [Activate the service](#), so it will not be repeated here.

**UserID** : The ID of the current user, string type, only allows to contain English letters (a-z and A-Z), numbers (0-9), hyphens (-), and underscores (\_).

**UserSig** : Encrypt the SDKAppID, UserID, etc. with the SDKSecretKey obtained in [Activate the service](#) to get the UserSig, which is a ticket for authorization and is used for Tencent Cloud to recognize whether the current user can use the TRTC service. You can create a temporarily available UserSig through the [UserSig Tools](#) through the project sidebar in the console.



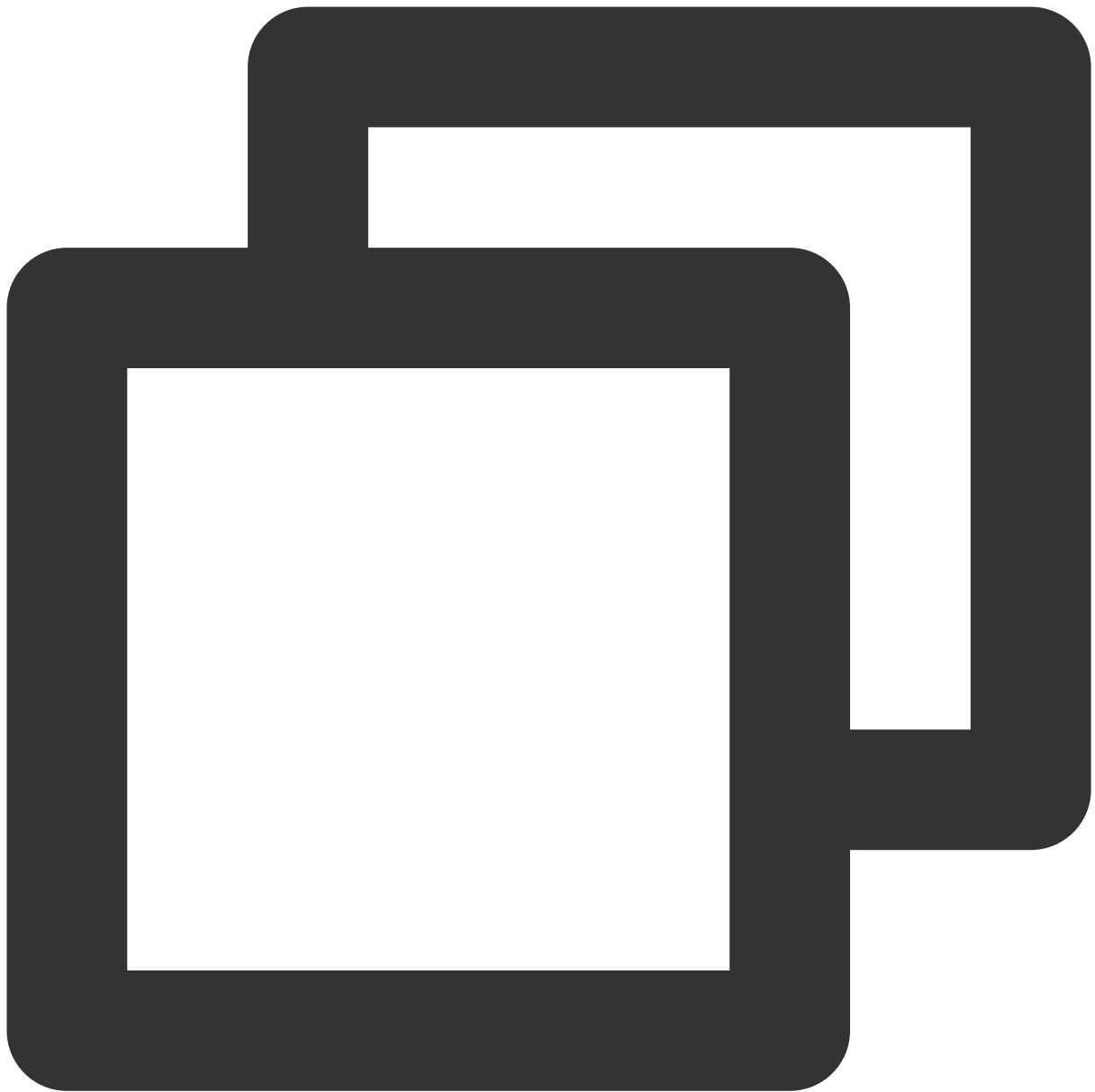
For more information, please refer to the [UserSig related](#).

## Step 5: The host start a quick conference

The main conference page is `ConferenceMainViewController`. Just call `quickStartConference` to initiate a quick conference and jump to `ConferenceMainViewController` in the `onConferenceStarted` callback.

Swift

OC



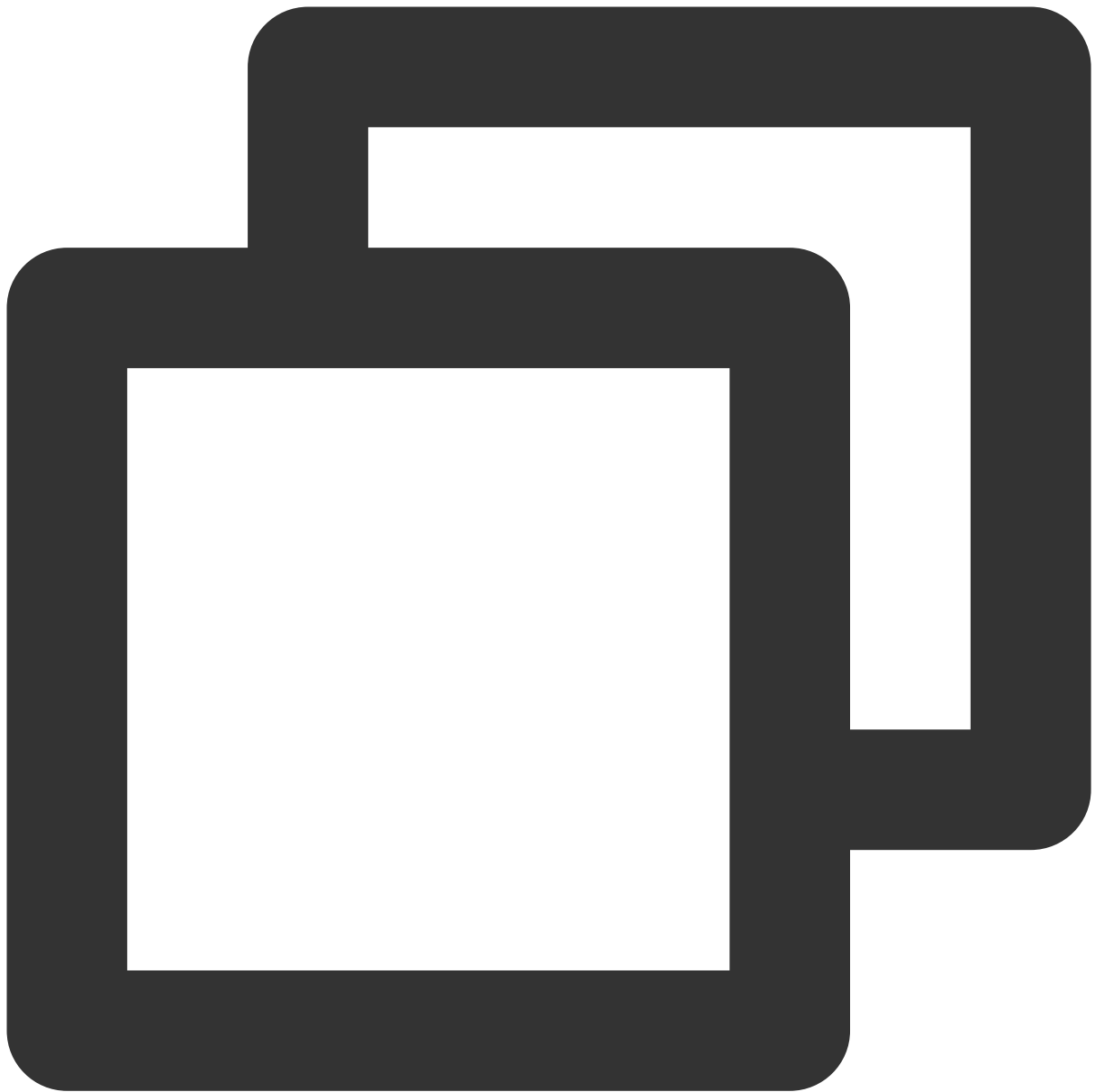
```
import TUIRoomKit

// CreateConferenceViewController is your own ViewController.
class CreateConferenceViewController: UIViewController {
    private var conferenceViewController: ConferenceMainViewController?
    func quickStartConferenceAction() {
        conferenceViewController = ConferenceMainViewController()
        let params = ConferenceParams() // The setting of params is optional, i
        params.isMuteMicrophone = false
        params.isOpenCamera = false
        conferenceViewController?.setConferenceParams(params: params)
```

```
        conferenceViewController?.setConferenceObserver(observer: self)
        conferenceViewController?.quickStartConference(conferenceId: "123456")
    }
}

extension CreateConferenceViewController: ConferenceObserver {
    func onConferenceStarted(conferenceId: String, error: ConferenceError) {
        if error == .success, let vc = conferenceViewController {
            navigationController?.pushViewController(vc, animated: true)
        }
        conferenceViewController = nil
    }
}
```





```
// CreateConferenceViewController is your own ViewController.
@interface CreateConferenceViewController ()<ConferenceObserver>
@property(strong, nonatomic) ConferenceMainViewController * conferenceController;
@end

@implementation CreateConferenceViewController

- (void)quickStartConferenceAction {
    _conferenceController = [[ConferenceMainViewController alloc] init];
    ConferenceParams * params = [[ConferenceParams alloc] init]; // The setting of
    params.isMuteMicrophone = false;
```

```
params.isOpenCamera = false;
[_conferenceController setConferenceParamsWithParams:params];
[_conferenceController setConferenceObserverWithObserver:self];
[_conferenceController quickStartConferenceWithConferenceId:@"123456"];
}

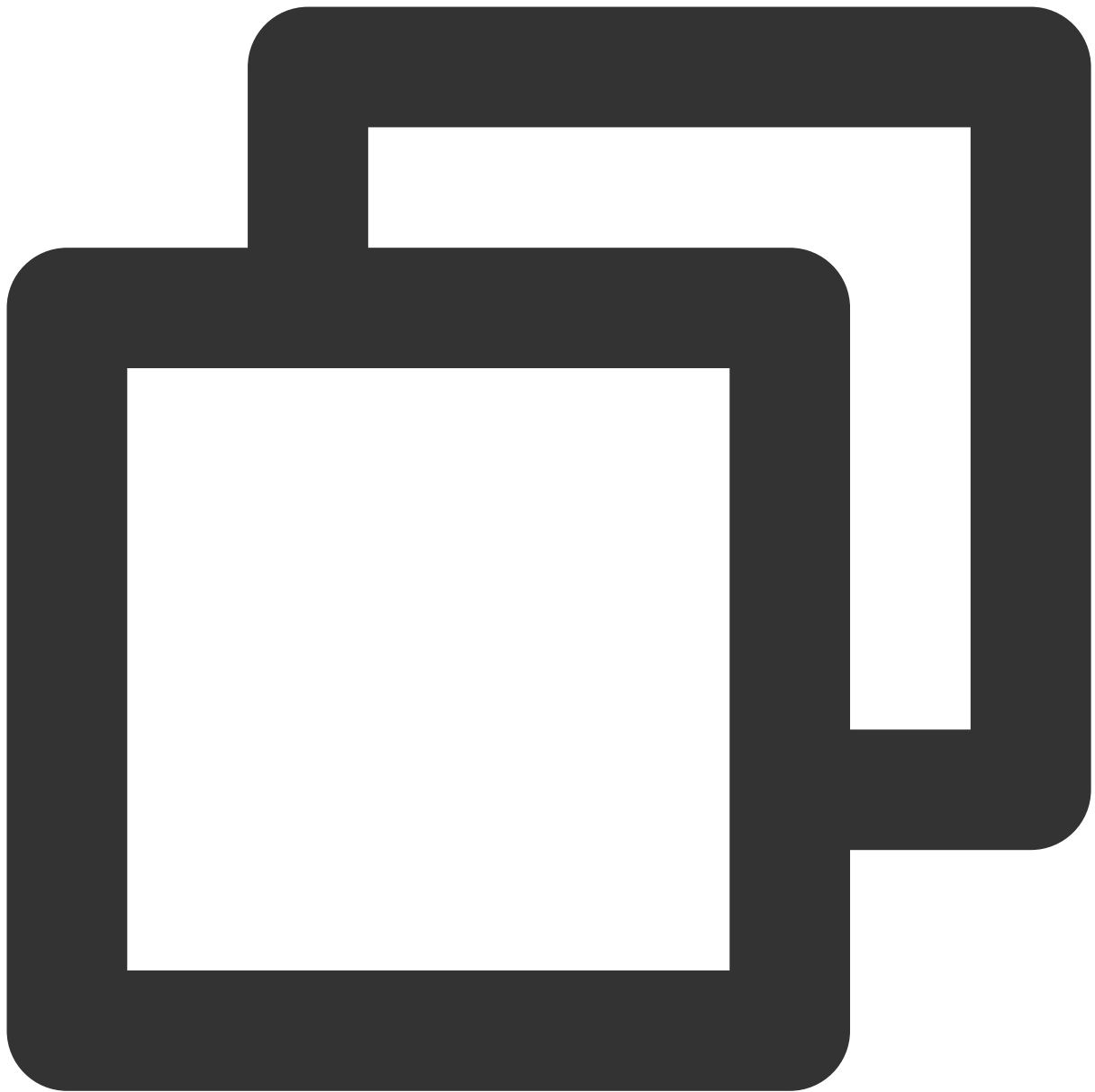
- (void)onConferenceStartedWithConferenceId:(NSString *)conferenceId error:(enum Co
    if (error == ConferenceErrorSuccess) {
        [self.navigationController pushViewController:_conferenceController animate
    }
    _conferenceController = nil;
}
@end
```

## Step 6: Members join the conference

The main conference page is `ConferenceMainViewController`. Just call `joinConference` to join the conference and jump to `ConferenceMainViewController` in the `onConferenceJoined` callback.

Swift

OC

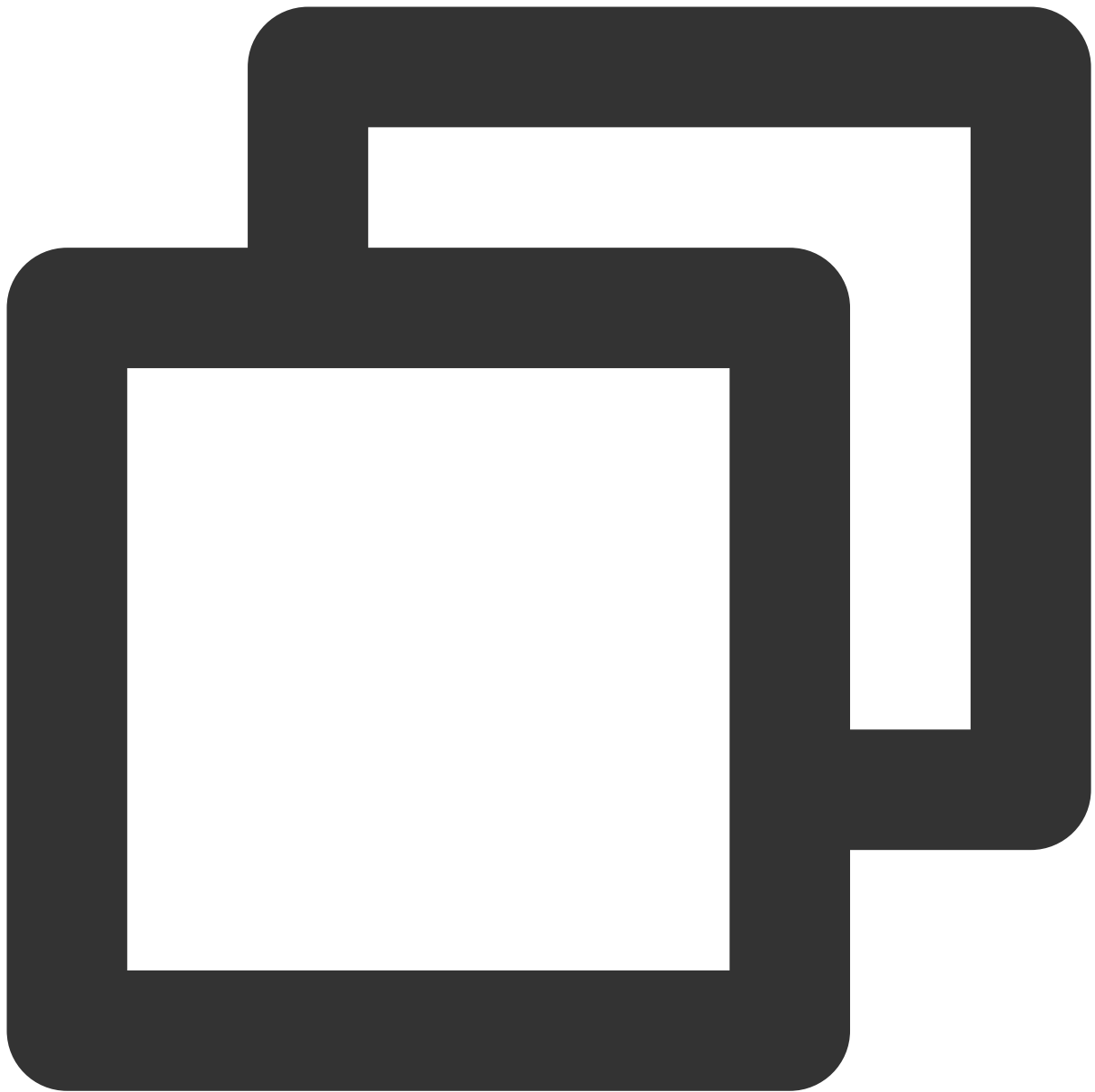


```
import TUIRoomKit

// EnterConferenceViewController is your own ViewController.
class EnterConferenceViewController: UIViewController {
    private var conferenceViewController: ConferenceMainViewController?
    private func joinConferenceAction() {
        conferenceViewController = ConferenceMainViewController()
        let params = ConferenceParams() // The setting of params is optional, i
        params.isMuteMicrophone = false
        params.isOpenCamera = false
        conferenceViewController?.setConferenceParams(params: params)
    }
}
```

```
        conferenceViewController?.setConferenceObserver(observer: self)
        conferenceViewController?.joinConference(conferenceId: "123456")
    }
}

extension EnterConferenceViewController: ConferenceObserver {
    func onConferenceJoined(conferenceId: String, error: ConferenceError) {
        if error == .success, let vc = conferenceViewController {
            navigationController?.pushViewController(vc, animated: true)
        }
        conferenceViewController = nil
    }
}
```



```
// EnterConferenceViewController is your own ViewController.
@interface EnterConferenceViewController ()<ConferenceObserver>
@property(strong, nonatomic) ConferenceMainViewController * conferenceController;
@end

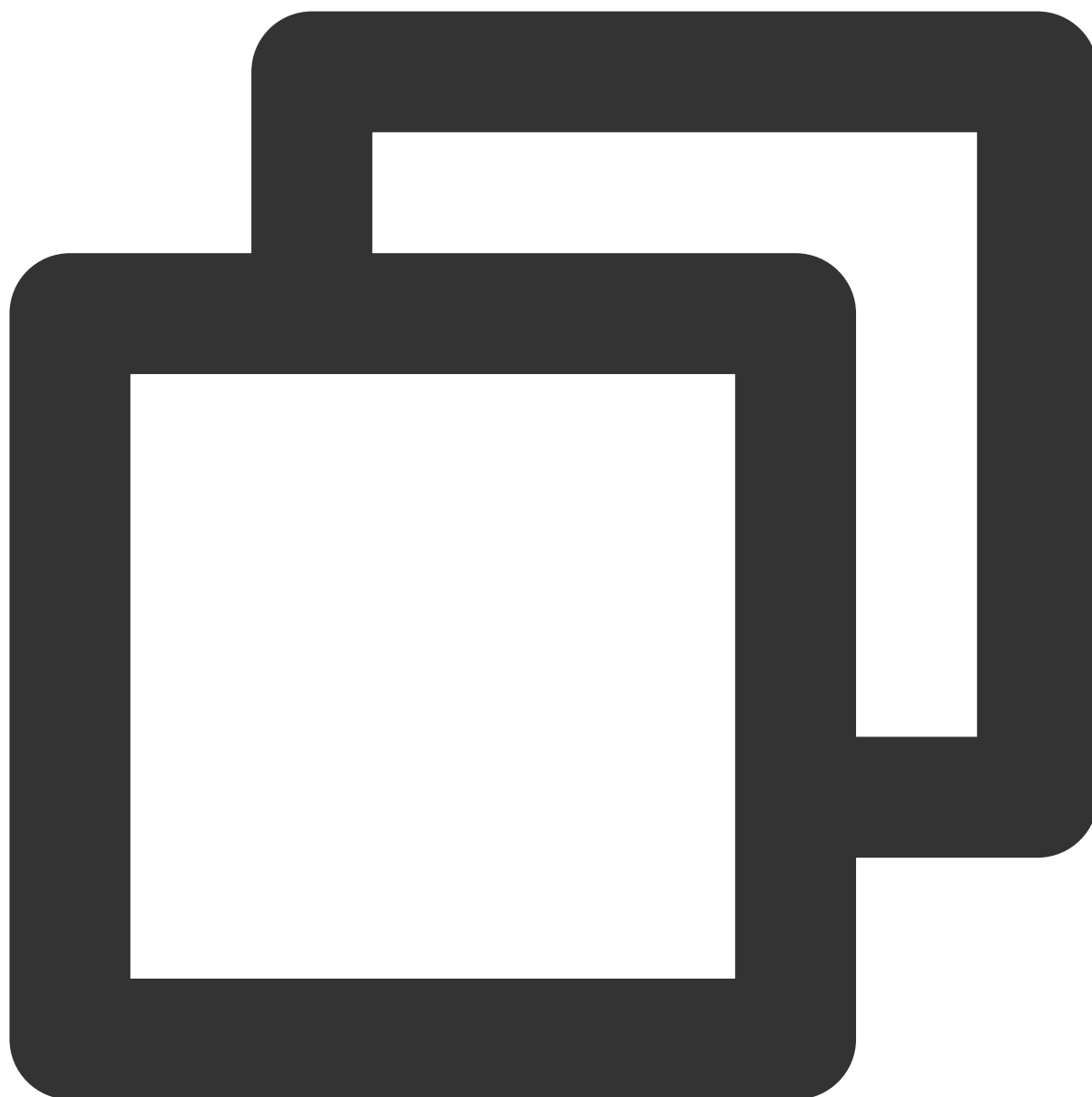
@implementation EnterConferenceViewController

- (void)joinConferenceAction {
    _conferenceController = [[ConferenceMainViewController alloc] init];
    ConferenceParams * params = [[ConferenceParams alloc] init]; // The setting of
    params.isMuteMicrophone = false;
```

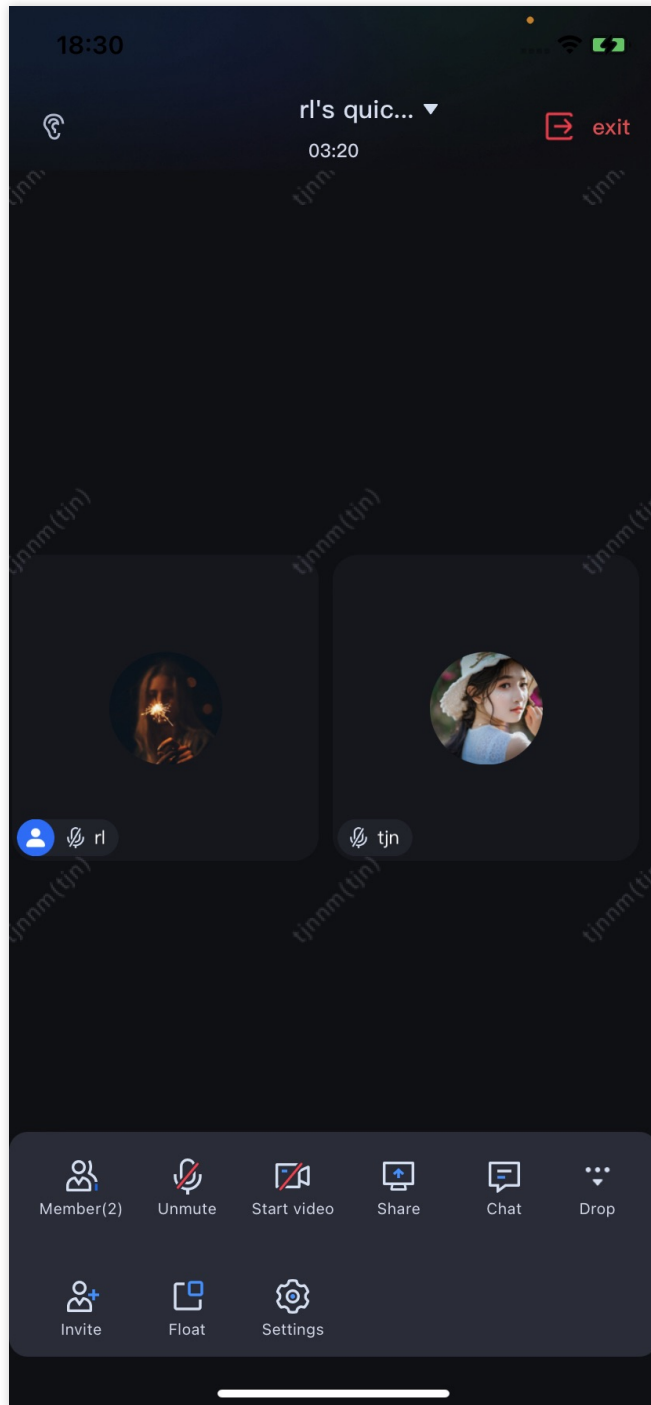
```
params.isOpenCamera = false;
[_conferenceController setConferenceParamsWithParams:params];
[_conferenceController setConferenceObserverWithObserver:self];
[_conferenceController joinConferenceWithConferenceId:@"123456"];
}

- (void) onConferenceJoinedWithConferenceId:(NSString *)conferenceId error:(enum Co
    if (error == ConferenceErrorSuccess) {
        [self.navigationController pushViewController:_conferenceController animate
    }
    _conferenceController = nil;
}
```

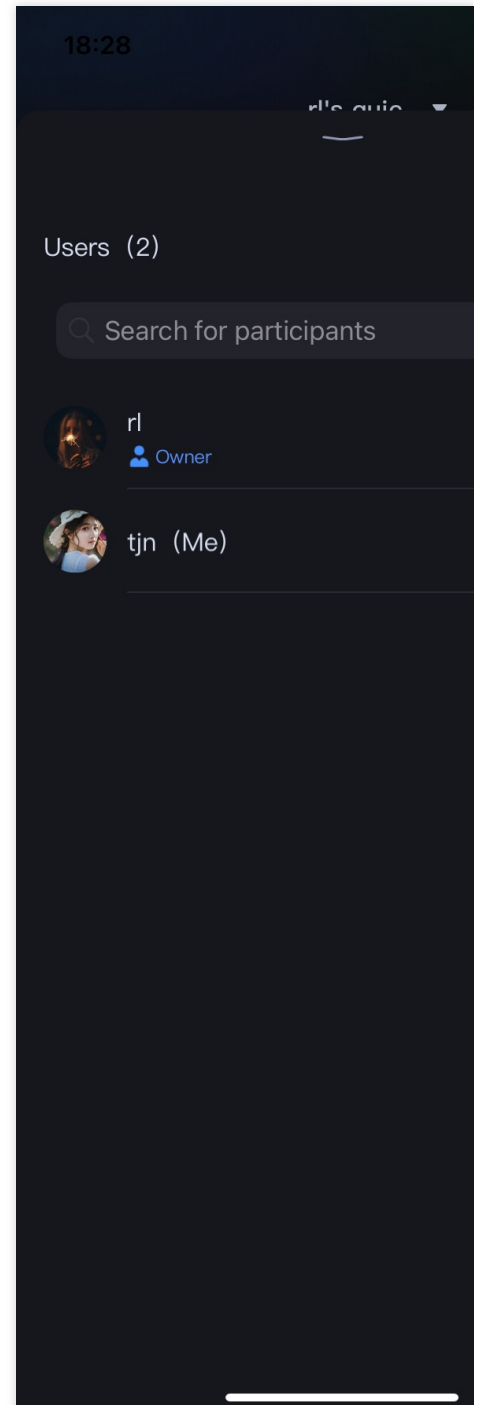
**Note:** When using OC for access, TUIRoomKit needs to be imported into the bridge file.



```
#import <TUIRoomKit/TUIRoomKit-Swift.h>
```



Conference main interface



User list

## Common Problems

If you encounter problems when using TUIRoomKit, please check the [FAQ document](#) first.

## Suggestions and Feedback



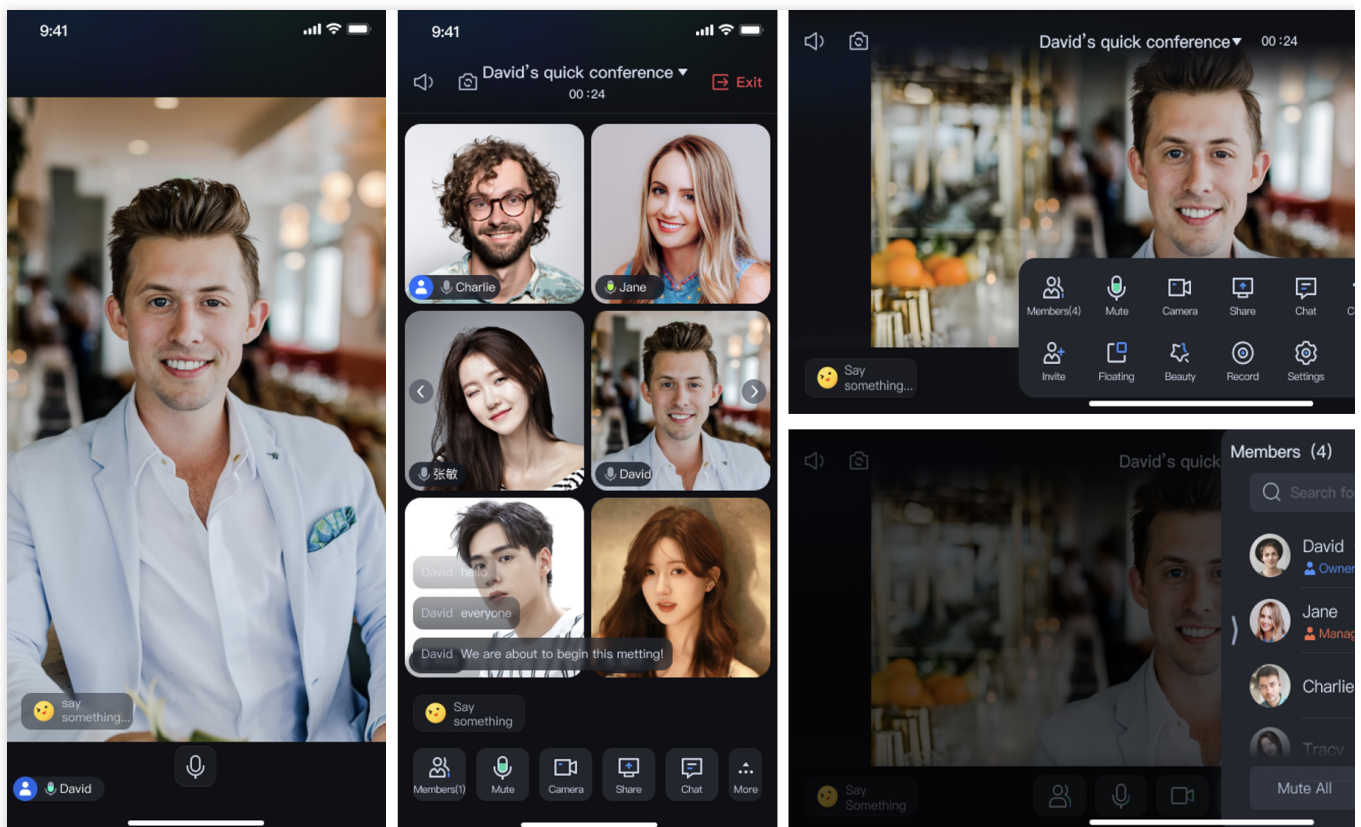
If you have any suggestions or feedback, please contact [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Android

Last updated : 2024-08-09 18:21:47

This article will introduce how to complete the integration of the `TUIRoomKit` Component in the shortest time. By following this document, you will complete the following key steps within ten minutes and ultimately obtain an audio/video conference function with a complete UI interface.

Conference interface and partial function display



## Environment preparation

Minimum compatibility with Android 4.4 (SDK API Level 19), recommended to use Android 5.0 (SDK API Level 21) and above.

Android Studio 3.5 and above (Gradle 3.5.4 and above).

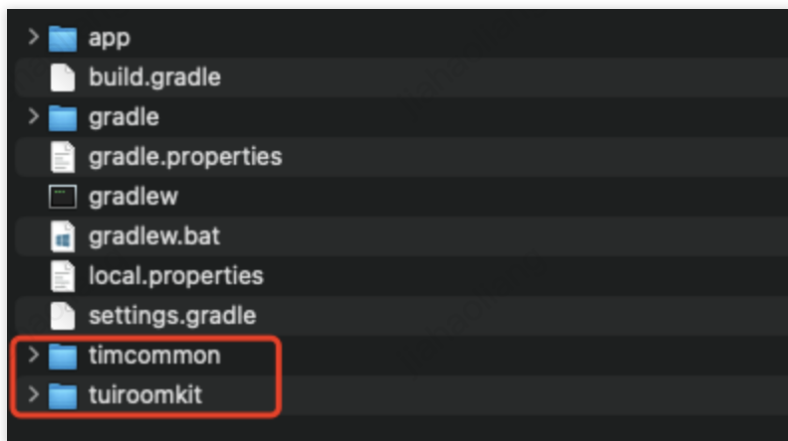
Mobile devices with Android 4.4 and above.

## Step 1: Activate the service

Before initiating a meeting with TUIRoomKit, you need to activate the exclusive multi-person audio and video interaction service for TUIRoomKit on the console. For specific steps, please refer to [Activate Service](#).

## Step 2: Download the TUIRoomKit component

1. Clone/download the code in [Github](#), and then copy the timcommon and tuiroomkit subdirectories in the Android directory to the same level directory as the app in your current project.

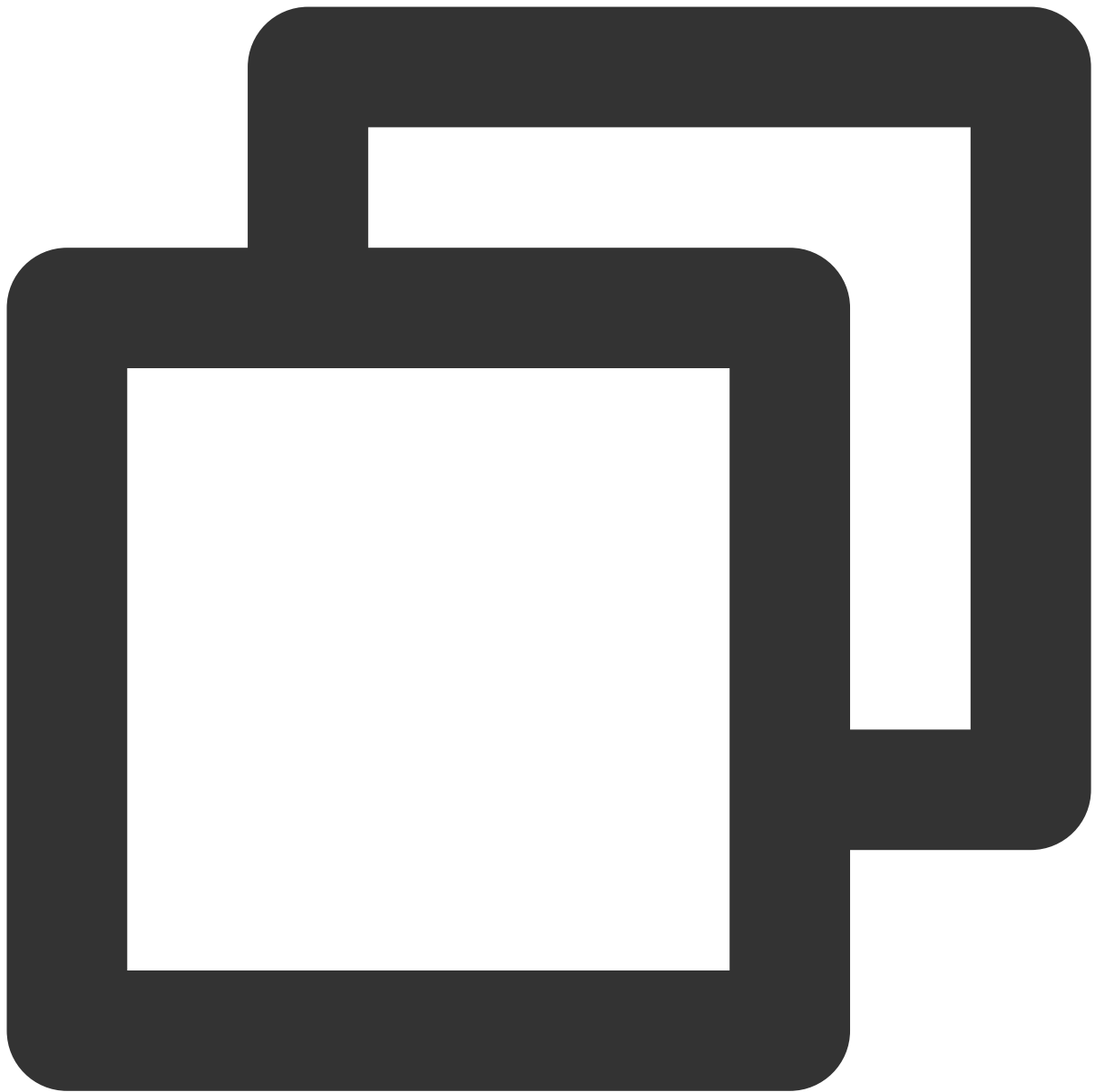


## Step 3: Project configuration

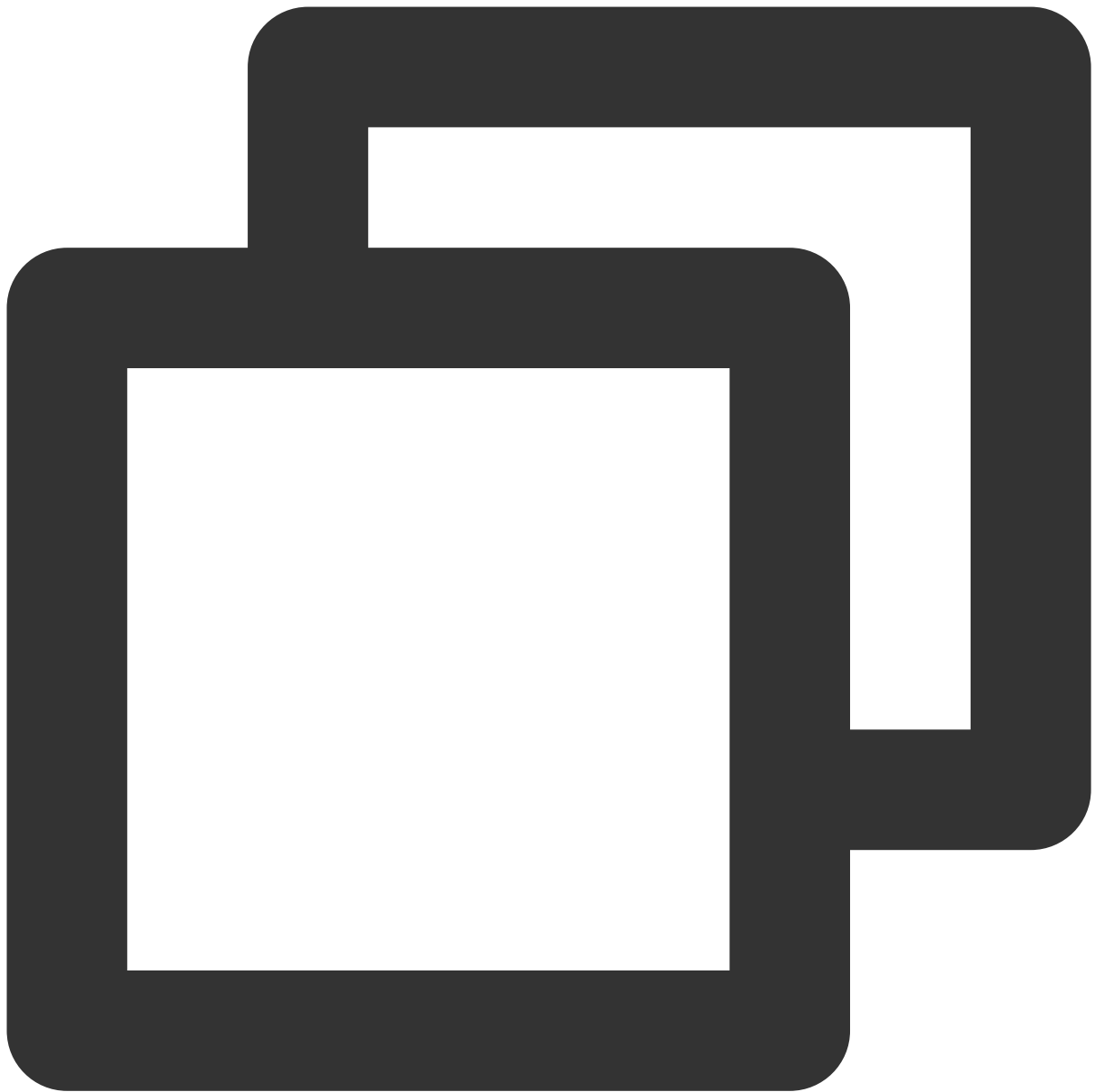
1. Find the `setting.gradle` (or `settings.gradle.kts`) file in the project root directory and add the following code to it. Its function is to import the tuiroomkit component into your current project.

`setting.gradle`

`settings.gradle.kts`



```
include ':timcommon'  
include ':tuiroomkit'
```

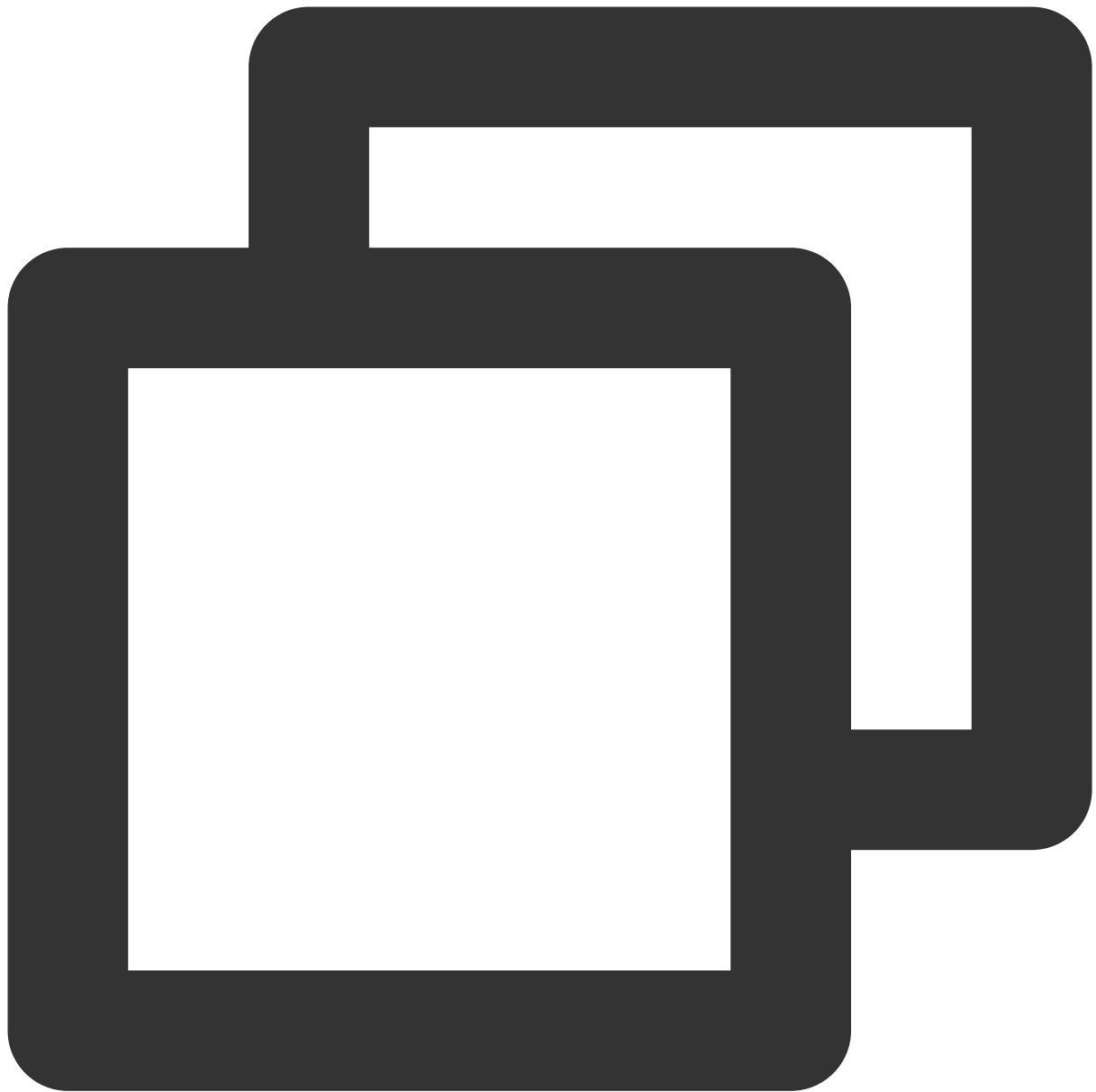


```
include (":timcommon")
include (":tuiroomkit")
```

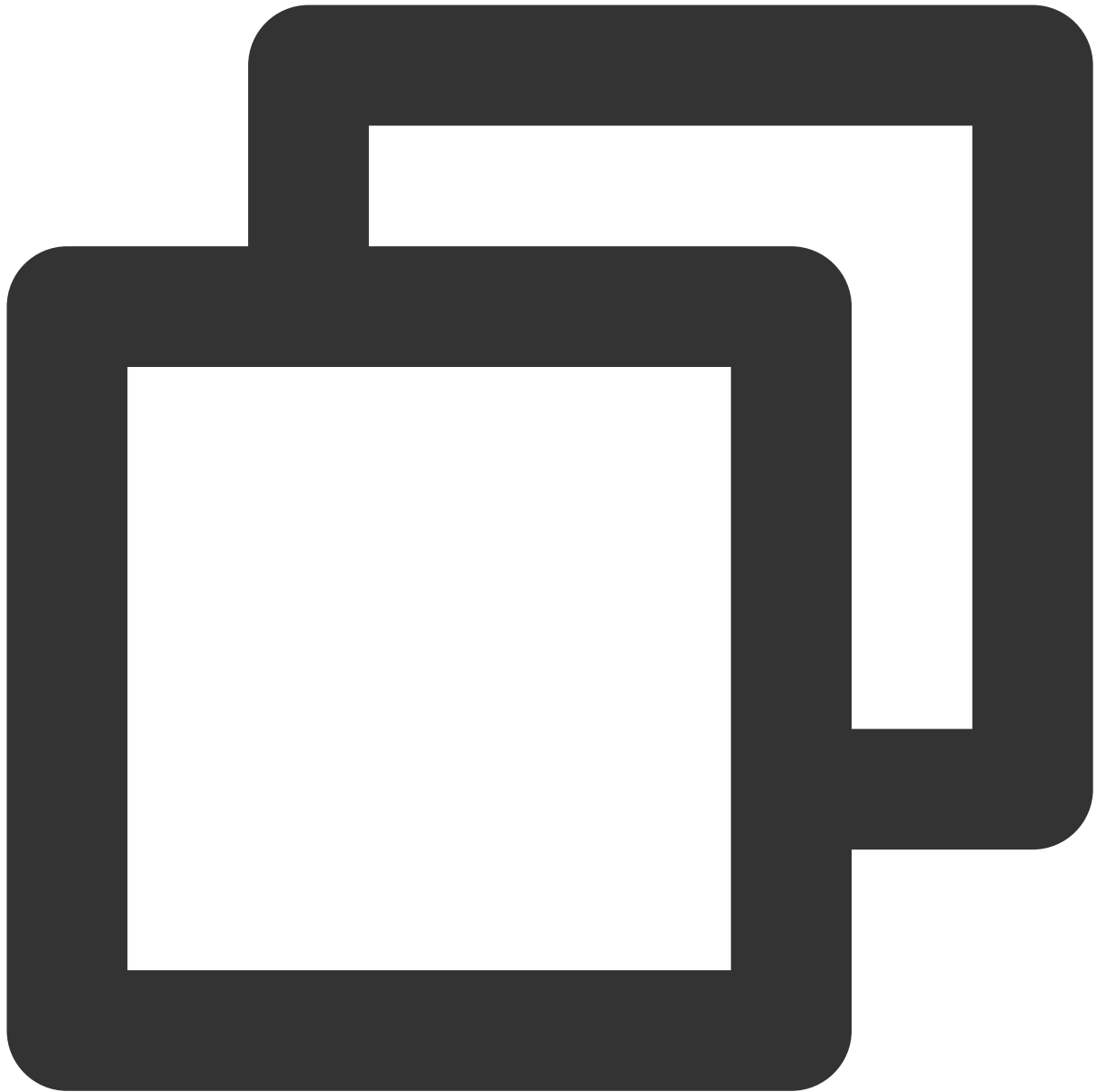
2. Find the `build.gradle` (or `build.gradle.kts`) file in the app directory and add the following code to it. Its function is to declare the current app's dependence on the newly added `tuiroomkit` component.

`build.gradle`

`build.gradle.kts`

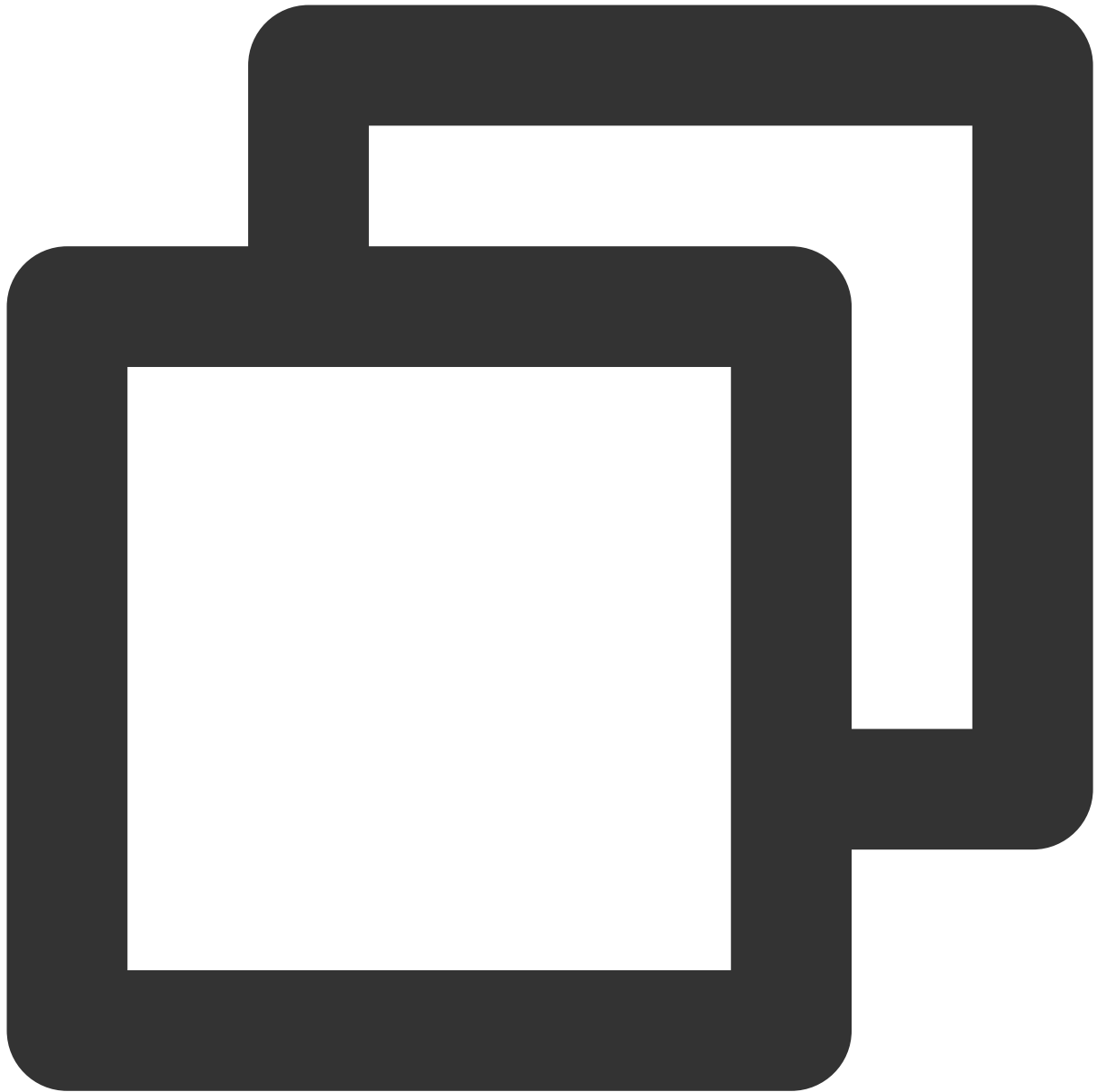


```
api project(':tuiroomkit')
```



```
api(project(":tuiroomkit"))
```

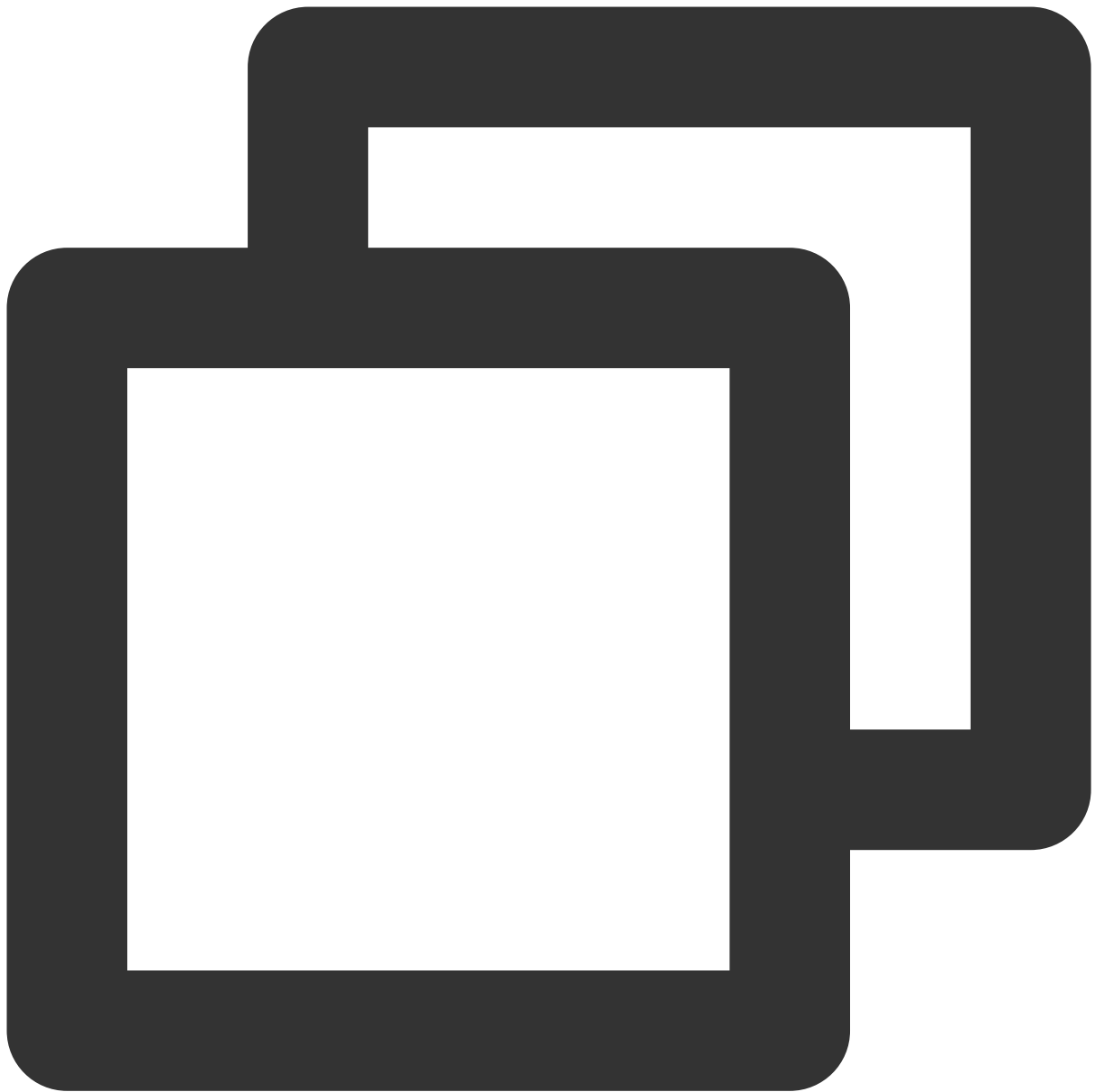
3. Since we use the reflection feature of Java inside the SDK, we need to add some classes in the SDK to the unobfuscated list, so you need to add the following code to the **proguard-rules.pro** file:



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

4. Find the AndroidManifest.xml file in the app directory, and add `tools:replace="android:allowBackup"` to the application node to override the settings within the component and use your own settings.





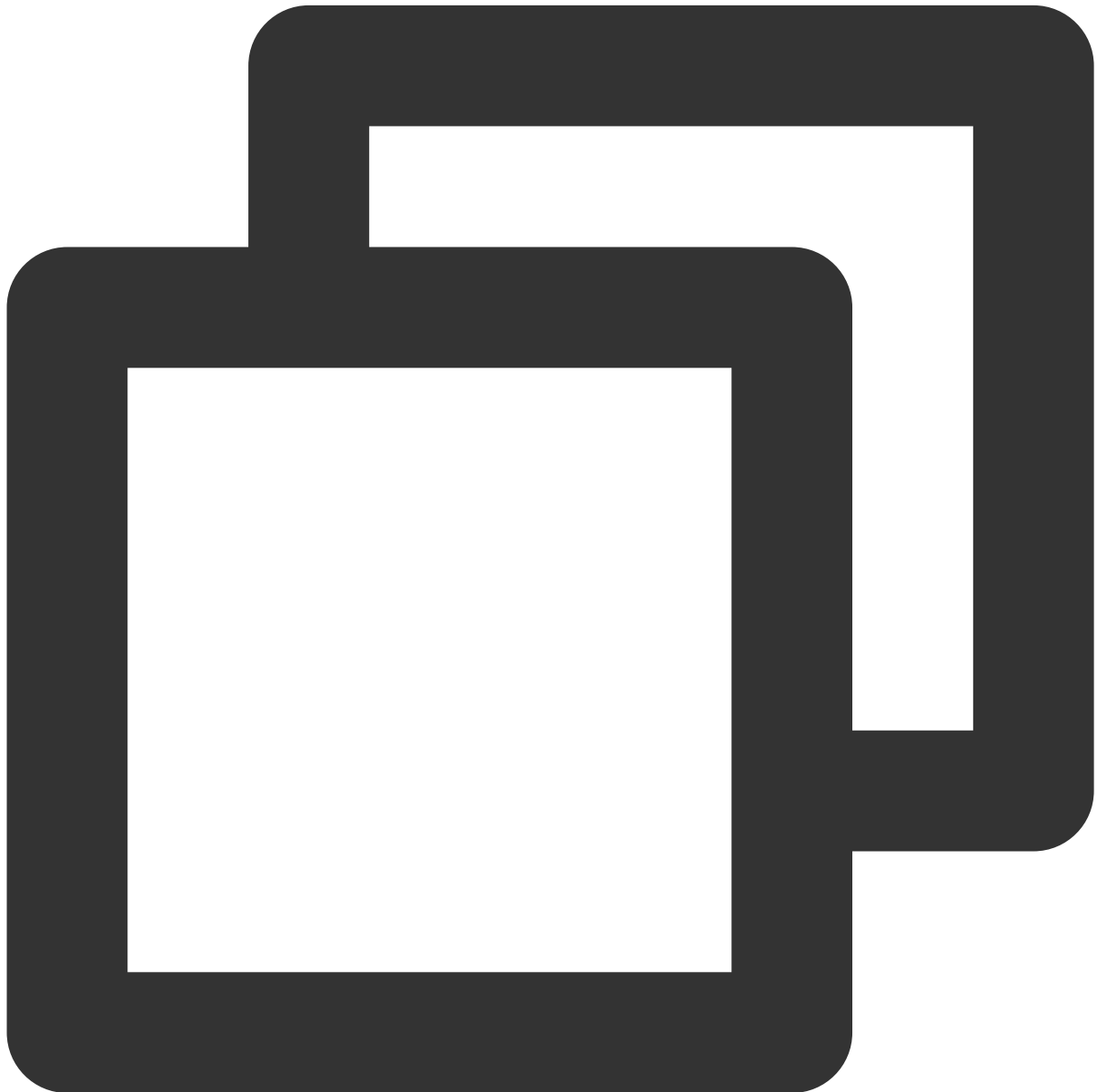
```
// app/src/main/AndroidManifest.xml
<application
    android:name=".DemoApplication"
    android:allowBackup="false"
    android:icon="@drawable/app_ic_launcher"
    android:label="@string/app_name"
    android:largeHeap="true"
    android:theme="@style/AppTheme"
    tools:replace="android:allowBackup">
```

## Step 4: Login

Add the following code to your project. Its function is to complete the component login by calling the relevant interface in TUILogin. **This step is extremely critical**, because you can only use the various functions of TUIRoomKit normally after logging in, so please be patient and check whether the relevant parameters are configured correctly:

Java

Kotlin



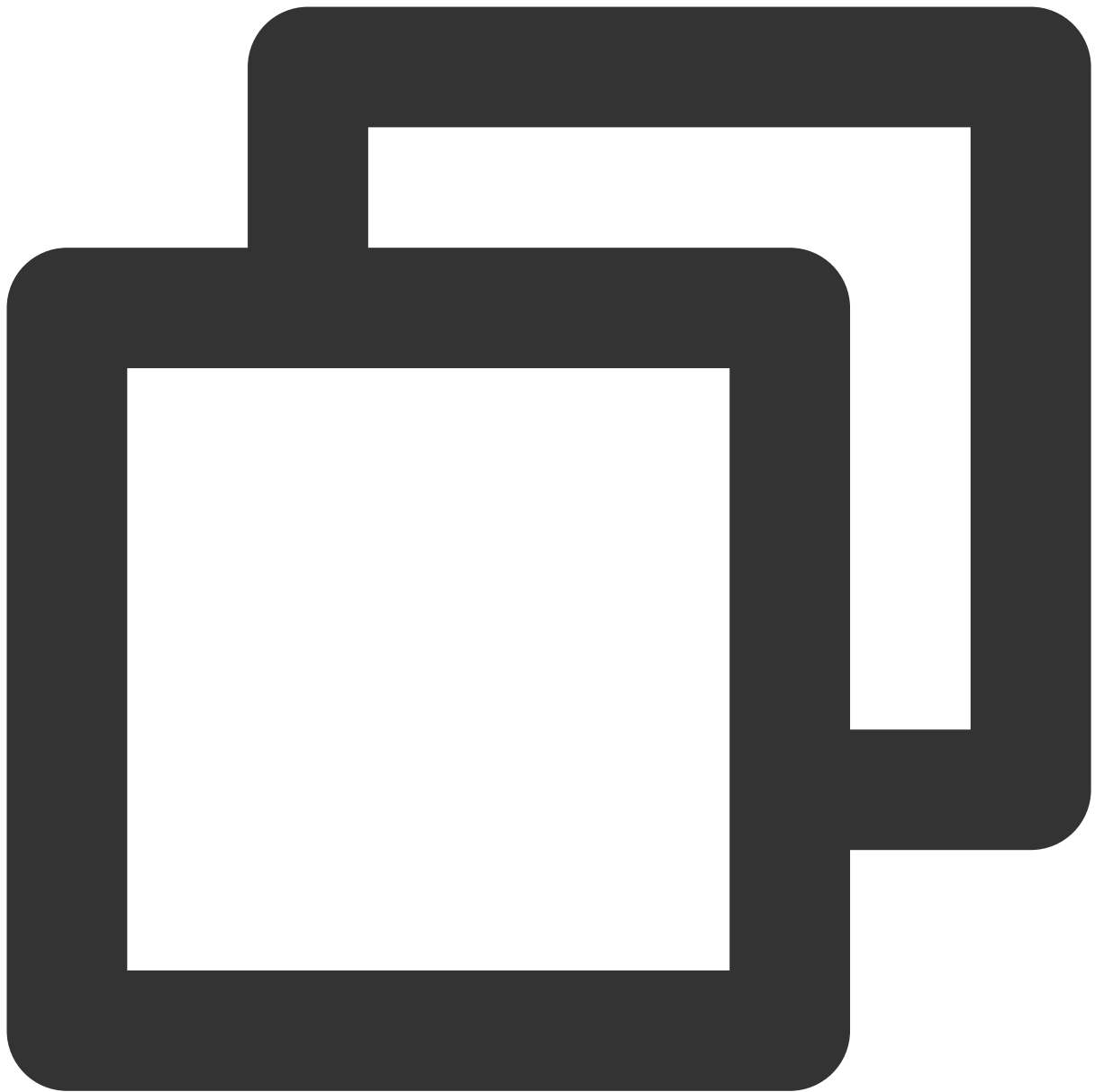
```
import com.tencent.qcloud.tuicore.TUILogin;
import com.tencent.qcloud.tuicore.interfaces.TUICallback;
```

```
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig;

String userId = "denny" // Please replace with your UserID
int sdkAppId = 1400000001 // Please replace with the sdkAppId obtained in step one
String sdkSecretKey = "xxxx" // Please replace with your sdkSecretKey
String userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId, sdkSecretKey)

TUILogin.login(context,
    sdkAppId,
    userId,
    userSig,
    new TUICallback() {
        @Override
        public void onSuccess() {
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
        }
    });
```



```
import com.tencent.qcloud.tuicore.TUILogin
import com.tencent.qcloud.tuicore.interfaces.TUICallback
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig

val userId = "denny" // Please replace with your UserID
val sdkAppId = 1400000001 // Please replace with the sdkAppId obtained in step one
val sdkSecretKey = "xxxx" // Please replace with your sdkSecretKey
val userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId, sdkSecretKey)

TUILogin.login(this,
    sdkAppId,
```

```
        userId,
        userSig,
        object : TUICallback() {
            override fun onSuccess() {

            }

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

            }
        })
    }
```

#### TUILogin.login Parameter Description :

Here is a detailed introduction to several key parameters needed in the login function:

**SDKAppID** : Get it in the last step of [activating the service](#).

**UserID** : The ID of the current user, a string type, only allowed to contain English letters (a-z and A-Z), numbers (0-9), hyphens (-) and underscores (\_).

**UserSig** : Use the SDKSecretKey obtained in step 4 of [activating the service](#) to encrypt the SDKAppID, UserID and other information to obtain the UserSig, which is an authentication ticket used by Tencent Cloud to identify whether the current user can use TRTC services. You can generate a temporarily available UserSig through the auxiliary tools in the [console](#).

**Note :**

**Development environment** : If you are in the local development and debugging stage, you can use the local GenerateTestUserSig.genTestUserSig() function to generate a userSig. The SDKSecretKey in this method can be easily decompiled and reverse-engineered. Once your key is leaked, attackers can steal your Tencent Cloud traffic.

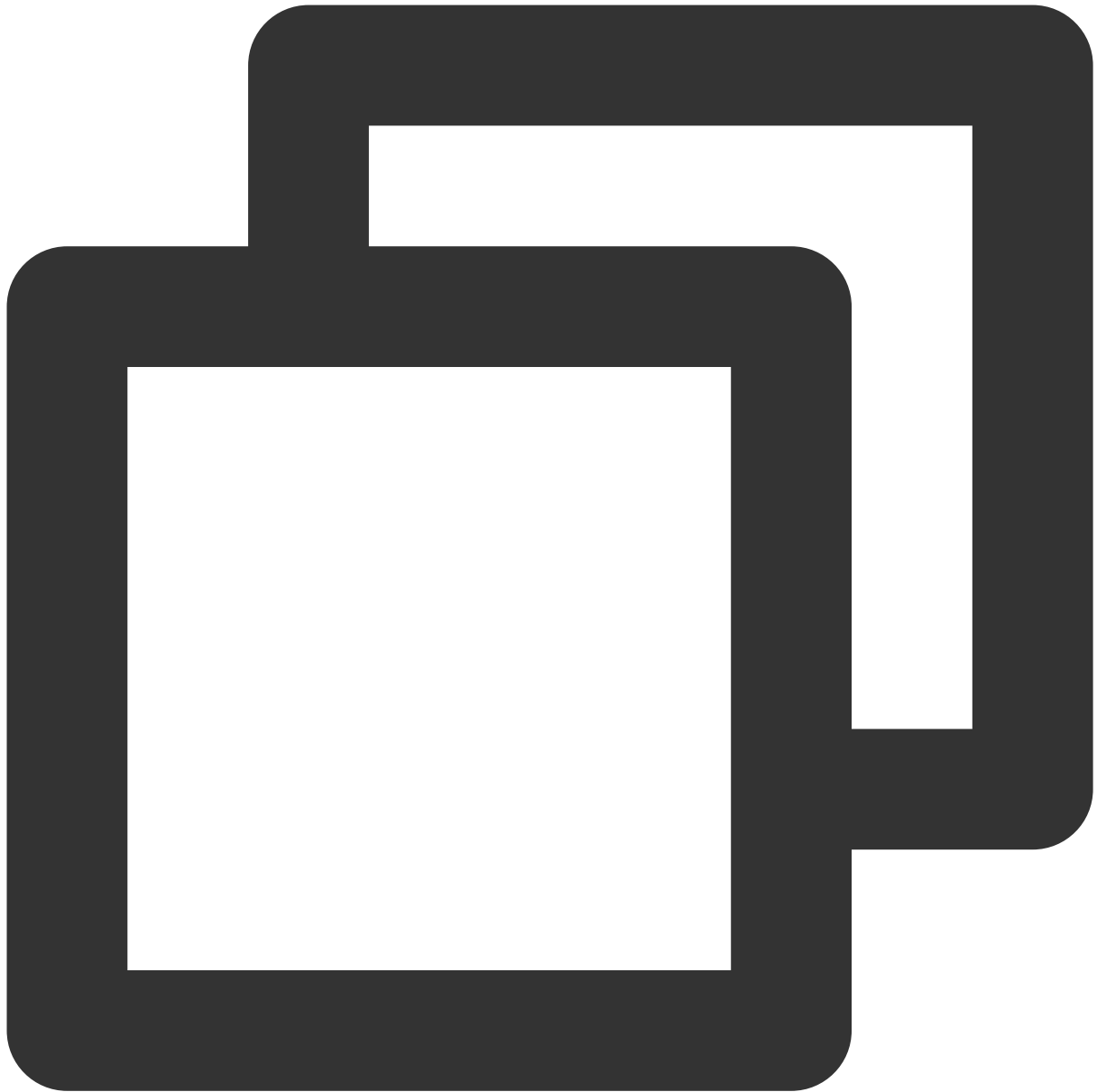
**production environment** : If your project is to be launched, please use the method of [server-side generation of UserSig](#).

## Step 5: Start your first conference

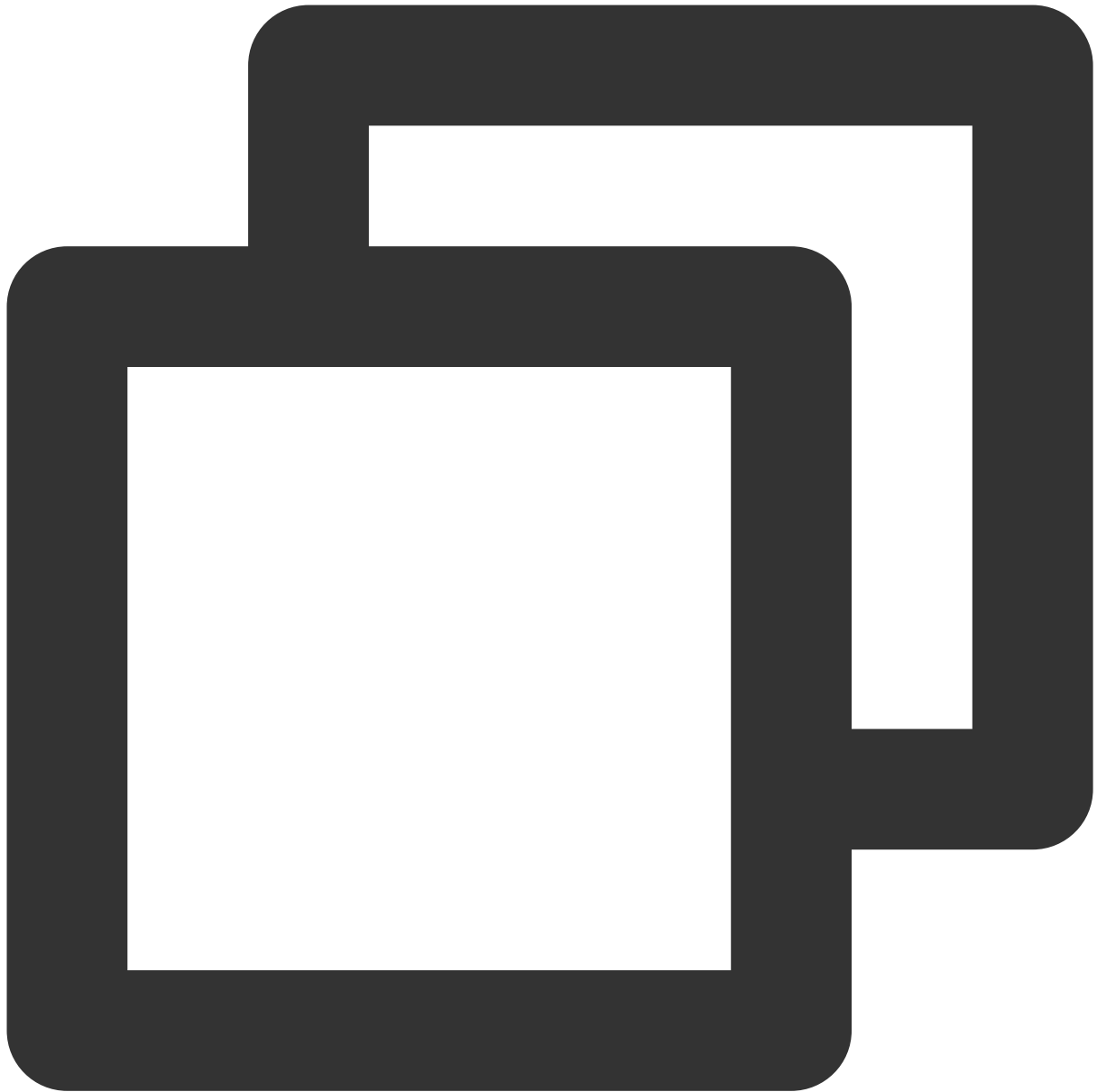
After [TUILogin.login](#) succeeds, refer to the following code to initiate a conference.

Java

Kotlin



```
// Please replace "123456" with your customized conference number
ConferenceDefine.StartConferenceParams params = new ConferenceDefine.StartConferenceParams("123456");
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent);
```



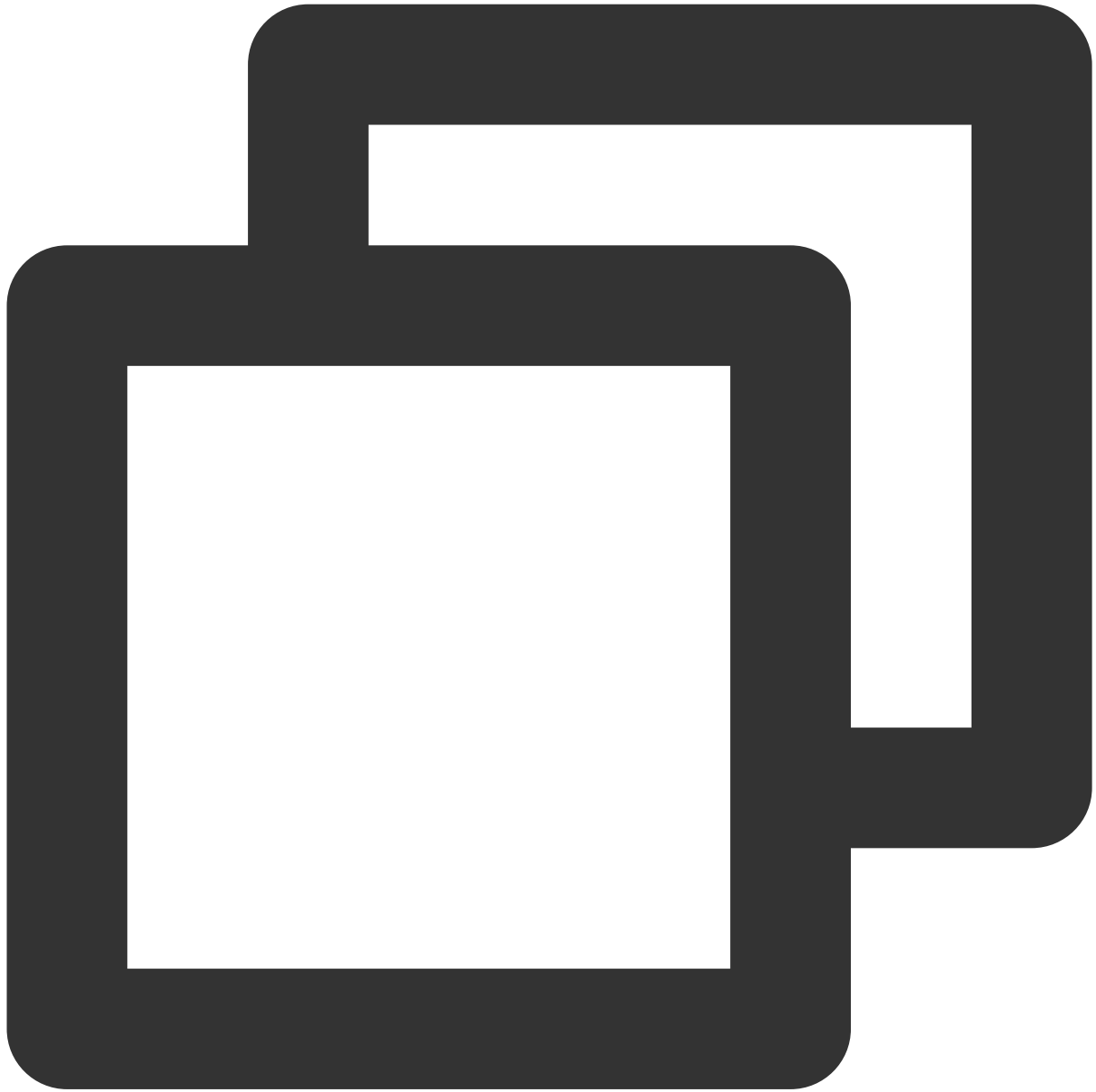
```
// Please replace "123456" with your customized conference number
val params = ConferenceDefine.StartConferenceParams("123456")
val intent = Intent(this, ConferenceMainActivity::class.java)
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent)
```

## Step 6: Join conference

After [TUILogin.login](#) succeeds, refer to the following code to join the conference.

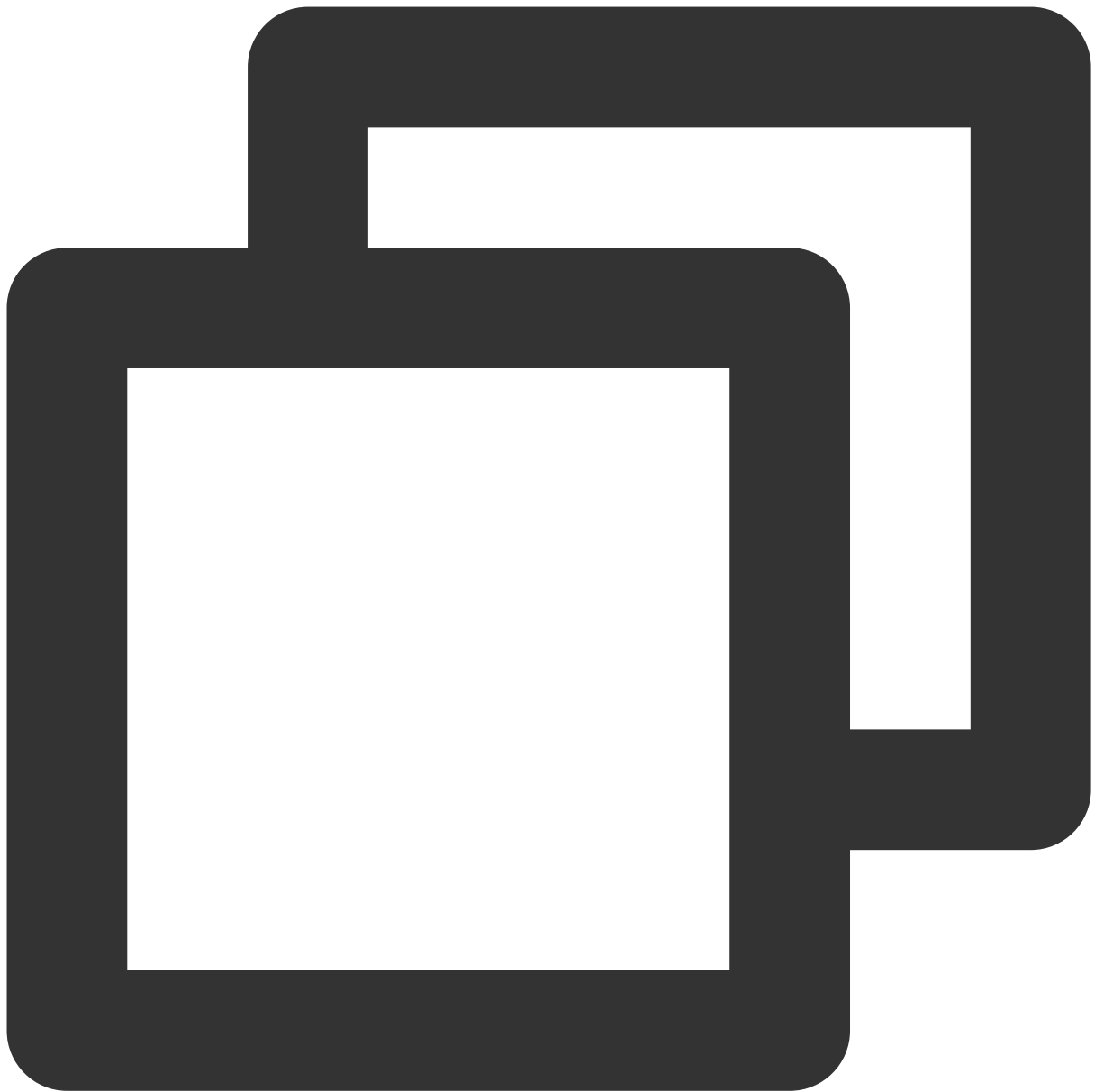
Java

Kotlin



```
// Please replace "123456" with your customized conference number
ConferenceDefine.JoinConferenceParams params = new ConferenceDefine.JoinConferenceP
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent);
```

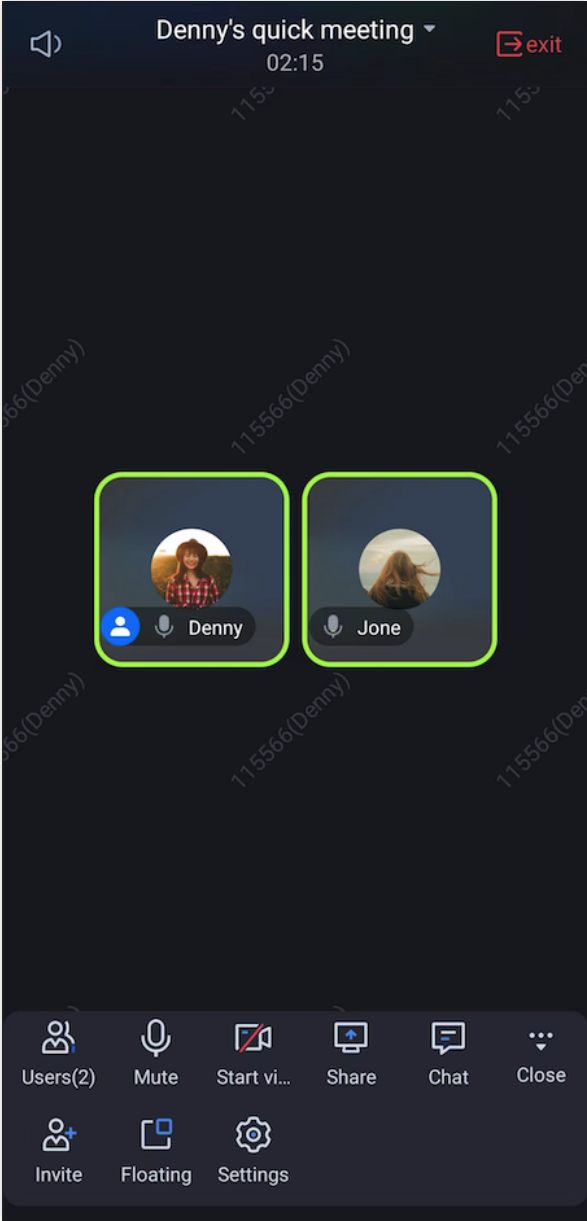
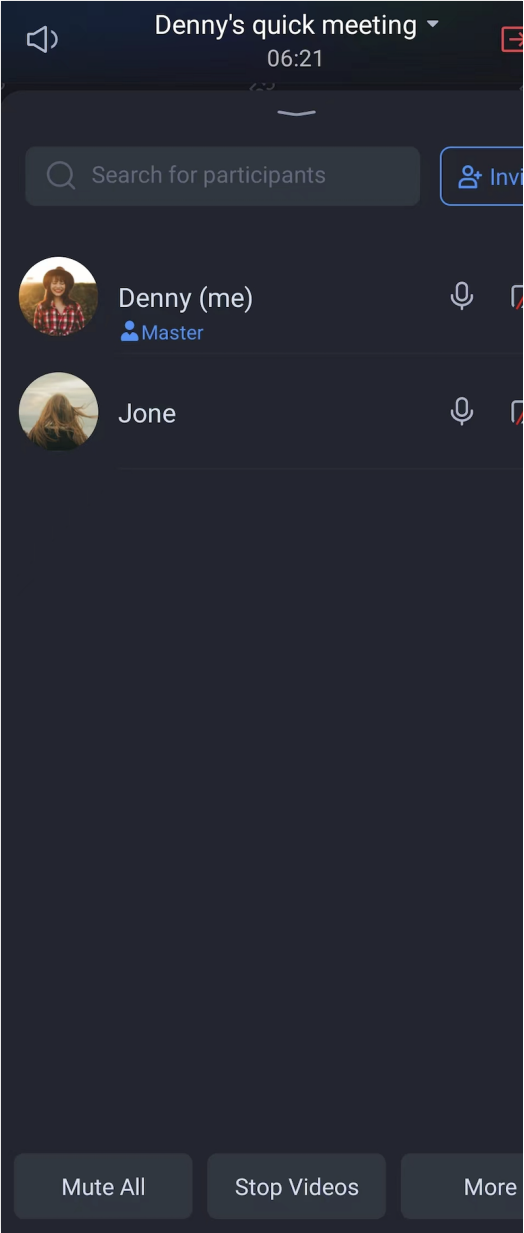




```
// Please replace "123456" with your customized conference number
val params = ConferenceDefine.JoinConferenceParams("123456")
val intent = Intent(this, ConferenceMainActivity::class.java)
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent)
```

## Interface display

When you successfully complete steps 1 - 6, the interface will appear as follows:

Conference main interface	User list
	

## Common problem

If you encounter problems with access and use, please see the [FAQ](#).

## Suggestions and Feedback

---

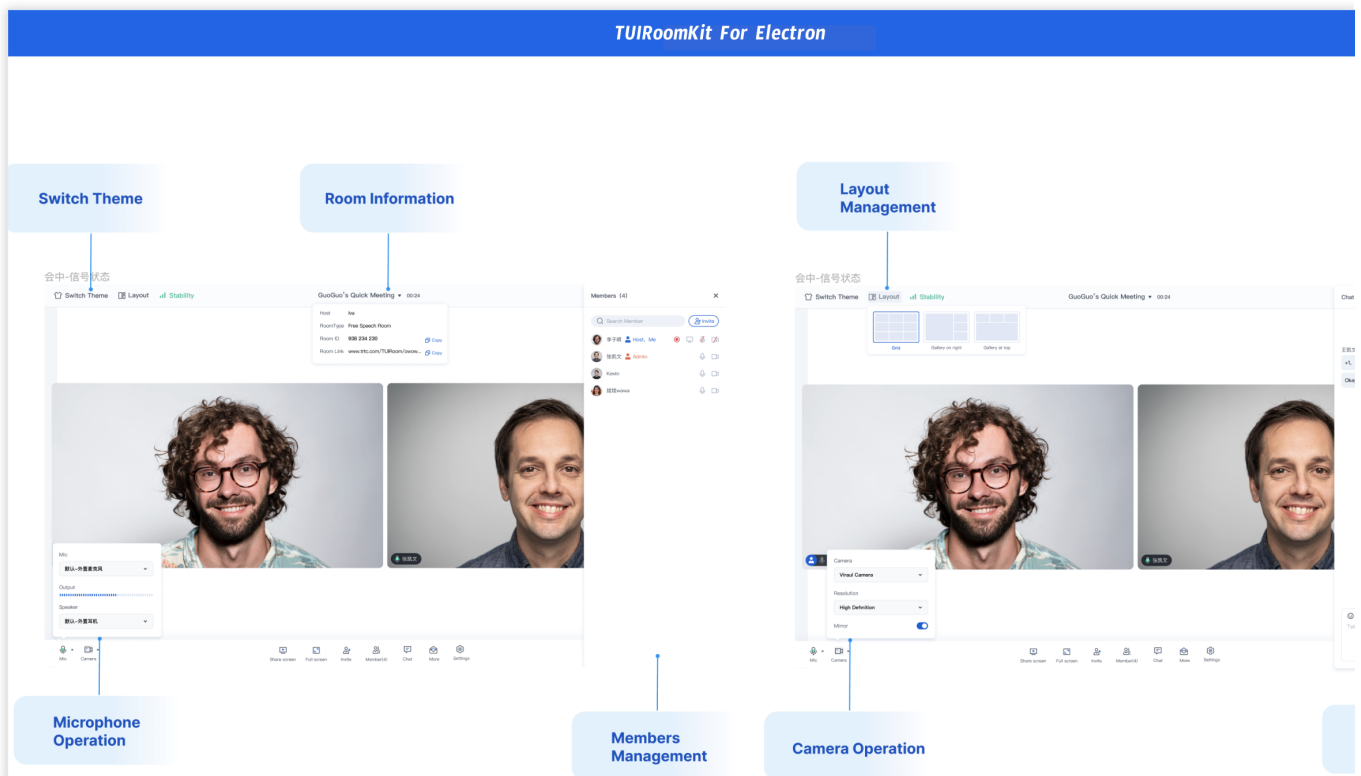
If you have any suggestions or feedback, please contact [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Electron

Last updated : 2024-06-17 15:12:28

TUIRoomKit is a multiperson audio/video UI component library launched by Tencent Cloud. It offers a rich array of features such as room management, audio and video control, screen sharing, member management, microphone position management, instant messaging, and custom layout switching, among others. It also supports switching between Chinese and English, and one-click skin changing capabilities.

This article introduces the access guidance for TUIRoomKit (Electron) to help you quickly launch business scenarios such as corporate meetings, online education, medical consultations, online patrols, and remote loss assessment.



## TUIRoomKit Demo Experience

You can click [Mac OS Version](#) and [Windows Version](#) to download and experience more features of TUIRoomKit Electron.

You can click [GitHub](#) to download the TUIRoomKit code, and refer to the README.md document in the code repository to run through the TUIRoomKit Electron sample project.

## Environment preparations

Node.js version: Node.js  $\geq 16.19.1$  (Using the official LTS version is recommended; please ensure the npm version matches the node version).

### NPM Package Integration

Vue3 development environment, integrate [@tencentcloud/room-electron-vue3](#) NPM package.

Vue2.7 development environment: Integrate [@tencentcloud/roomkit-electron-vue2.7](#) NPM package.

### Source Code Integration

Vue3 + TypeScript development environment: Copy source code from [@tencentcloud/room-electron-vue3](#) NPM package.

Vue2.7 + TypeScript development environment: Copy source code from [@tencentcloud/roomkit-electron-vue2.7](#) NPM package.

## Integrating TUIRoomKit Component

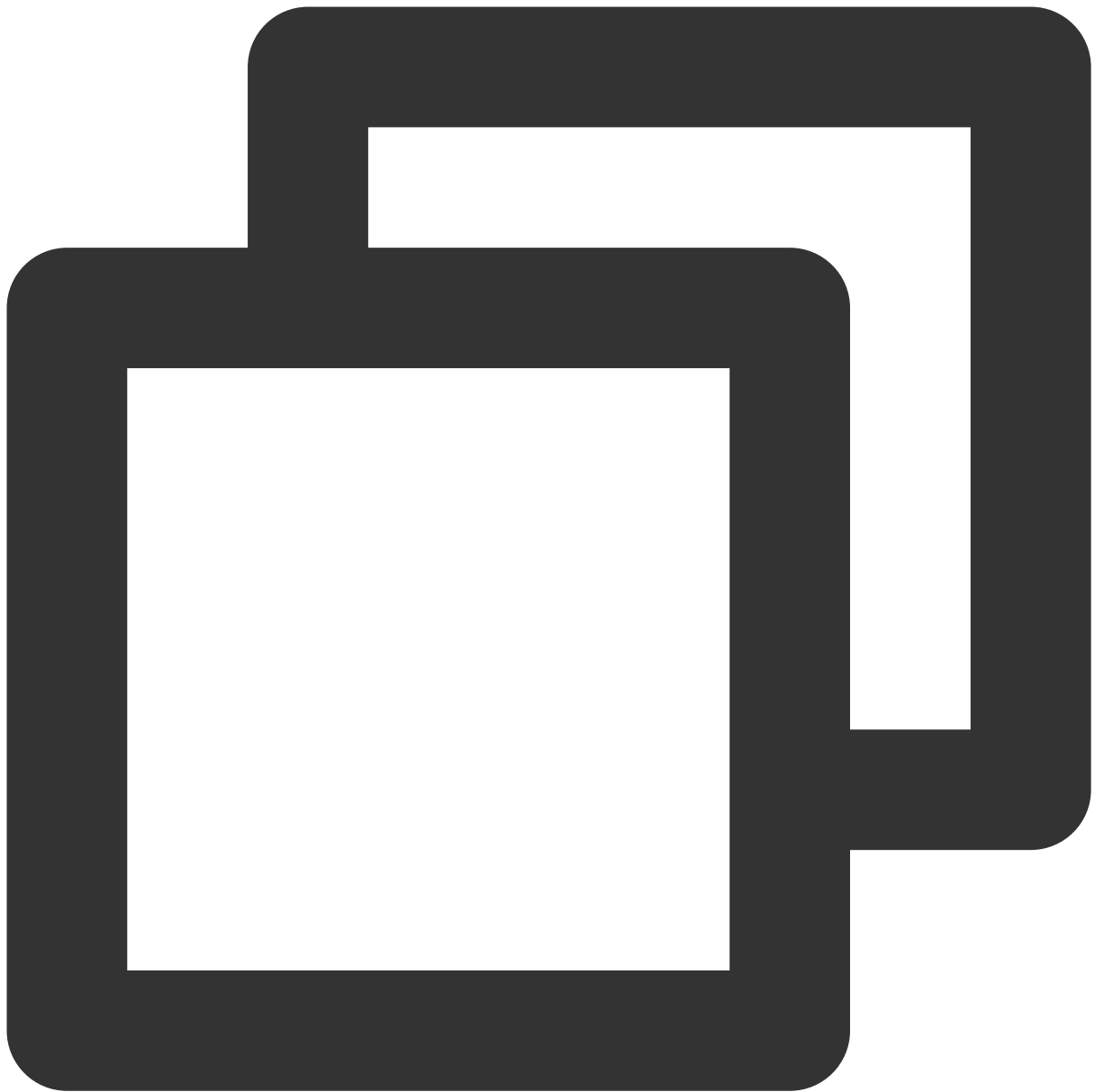
If **you don't have a Vue project**, you can go to [GitHub](#) to download the TUIRoomKit code, and refer to the code repository's README.md to run the TUIRoomKit Electron sample project.

If you need to integrate it into an existing project, please follow the steps below.

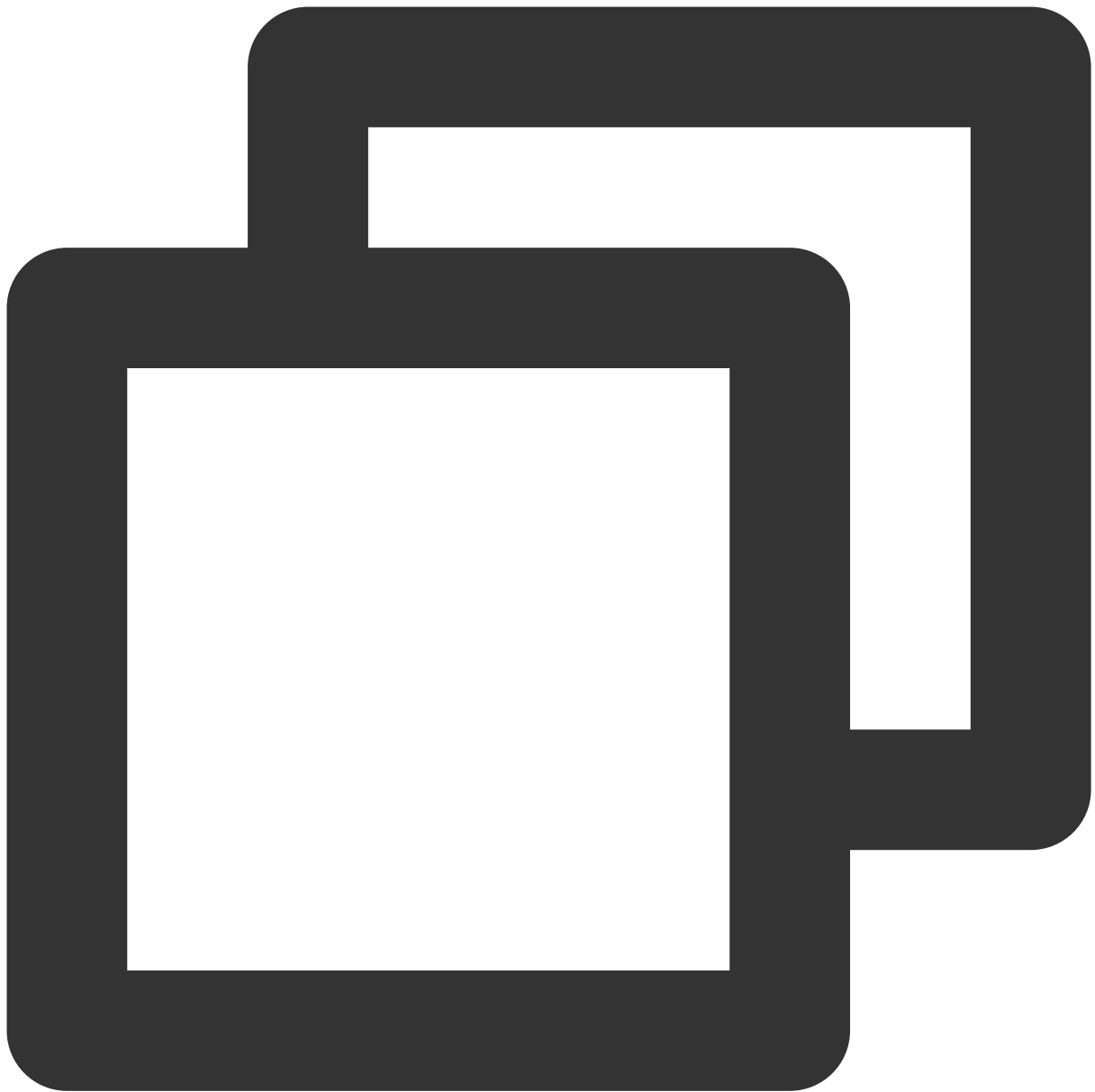
### Step 1: Install dependencies

Vue3

Vue2



```
npm install @tencentcloud/roomkit-electron-vue3 pinia --save
```



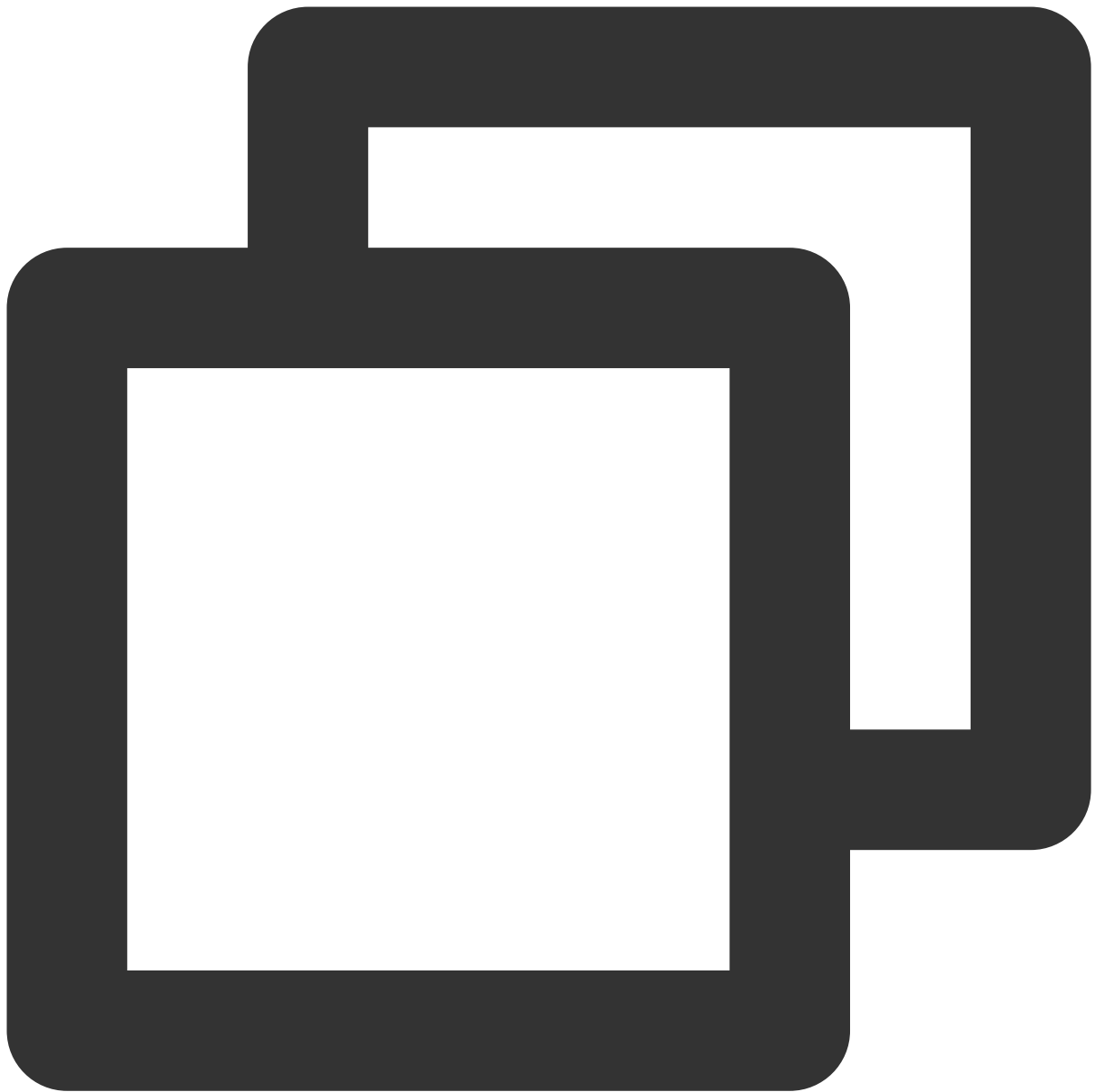
```
# Note that Vue version >= 2.7.16 is required here. If installation fails, please c  
npm install @tencentcloud/roomkit-electron-vue2.7 pinia
```

## Step 2: Configure the project

**Register Pinia:** TUIRoom uses Pinia for room data management, so you need to register Pinia in the entry file of the project. The entry file is the `src/main.ts` file.

Vue3

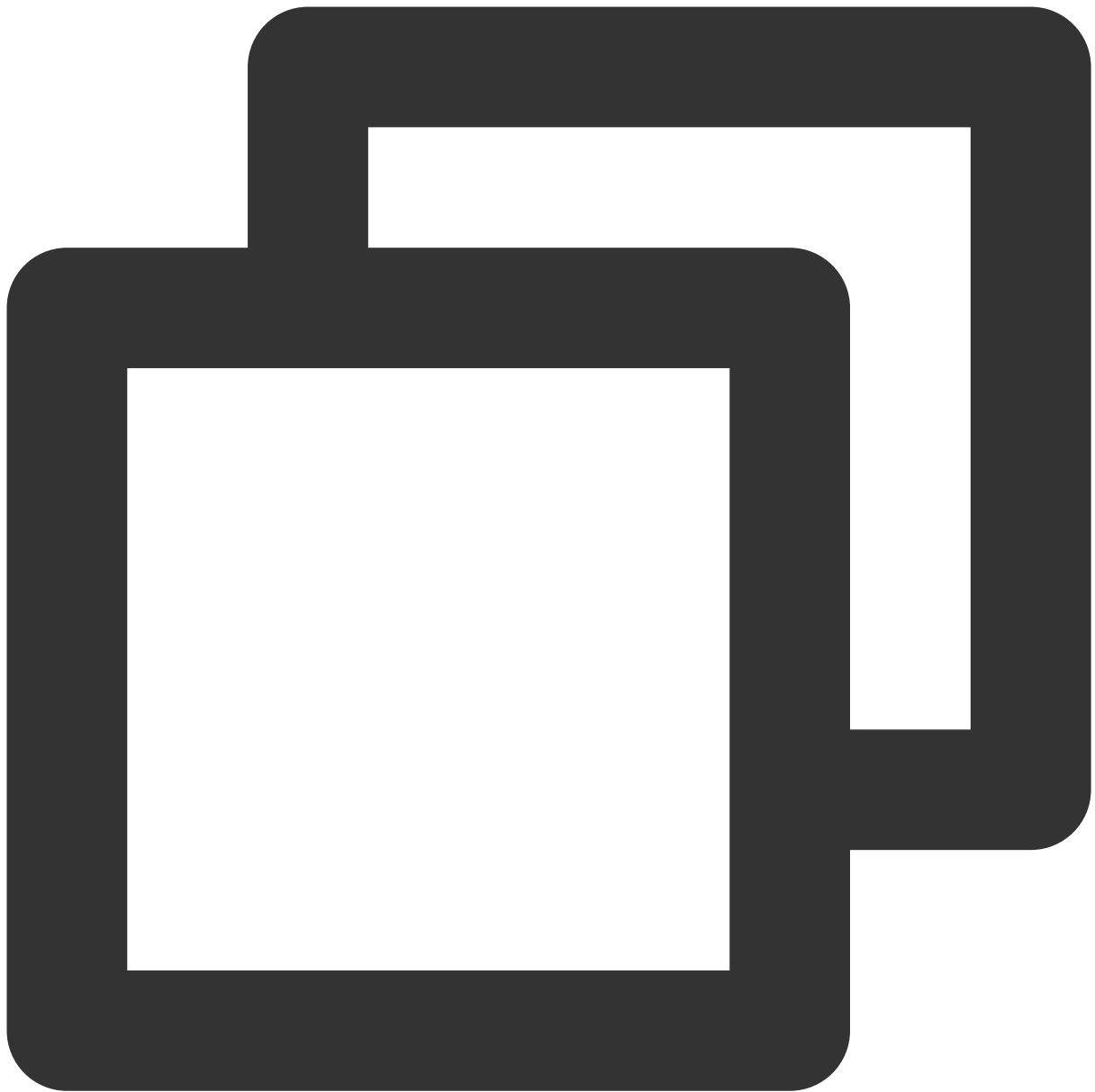
Vue2



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

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



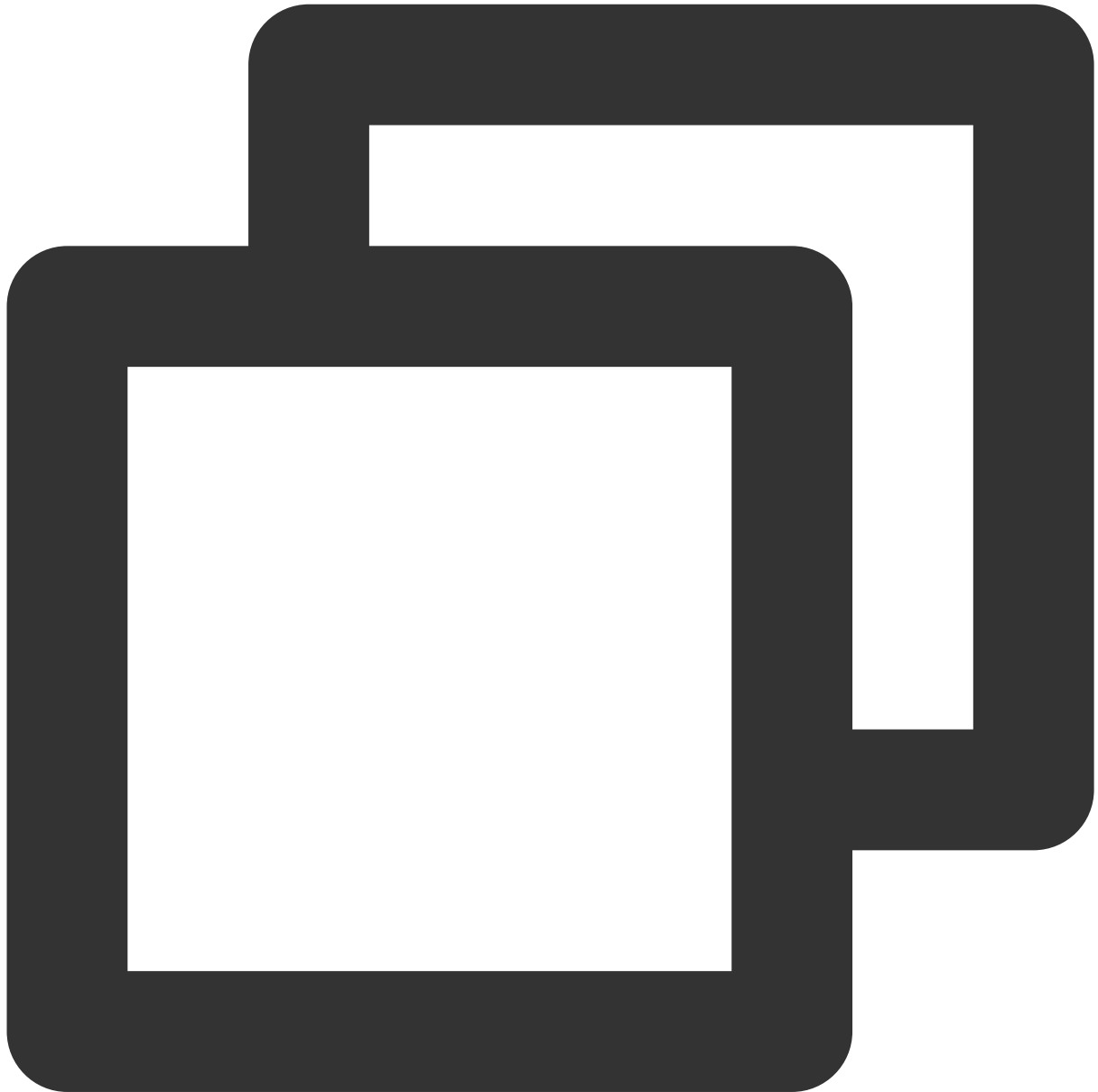


```
// src/main.ts file
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
}).$mount('#app');
```

**Configure vite.config.ts:** To unify code style and import trtc-electron-sdk through import in the UI layer (otherwise, it must be imported via require), you need to configure in `packages/renderer/vite.config.ts`. Please replace the content in resolve with the following configuration items, refer to the file [packages/renderer/vite.config.ts](#) for details.



```
// vite.config.ts
export default defineConfig({
  // ...
  plugins: [
    resolve({
      'trtc-electron-sdk': `
```

```
const TRTCCloud = require("trtc-electron-sdk");
const TRTCParams = TRTCCloud.TRTCParams;
const TRTCAppScene = TRTCCloud.TRTCAppScene;
const TRTCVideoStreamType = TRTCCloud.TRTCVideoStreamType;
const TRTCScreenCaptureSourceType = TRTCCloud.TRTCScreenCaptureSourceType;
const TRTCVideoEncParam = TRTCCloud.TRTCVideoEncParam;
const Rect = TRTCCloud.Rect;
const TRTCAudioQuality = TRTCCloud.TRTCAudioQuality;
const TRTCScreenCaptureSourceInfo = TRTCCloud.TRTCScreenCaptureSourceInfo;
const TRTCDeviceInfo = TRTCCloud.TRTCDeviceInfo;
const TRTCVideoQosPreference = TRTCCloud.TRTCVideoQosPreference;
const TRTCQualityInfo = TRTCCloud.TRTCQualityInfo;
const TRTCQuality = TRTCCloud.TRTCQuality;
const TRTCStatistics = TRTCCloud.TRTCStatistics;
const TRTCVolumeInfo = TRTCCloud.TRTCVolumeInfo;
const TRTCDeviceType = TRTCCloud.TRTCDeviceType;
const TRTCDeviceState = TRTCCloud.TRTCDeviceState;
const TRTCBeautyStyle = TRTCCloud.TRTCBeautyStyle;
const TRTCVideoResolution = TRTCCloud.TRTCVideoResolution;
const TRTCVideoResolutionMode = TRTCCloud.TRTCVideoResolutionMode;
const TRTCVideoMirrorType = TRTCCloud.TRTCVideoMirrorType;
const TRTCVideoRotation = TRTCCloud.TRTCVideoRotation;
const TRTCVideoFillMode = TRTCCloud.TRTCVideoFillMode;
const TRTCRoleType = TRTCCloud.TRTCRoleType;
const TRTCScreenCaptureProperty = TRTCCloud.TRTCScreenCaptureProperty;
export {
  TRTCParams,
  TRTCAppScene,
  TRTCVideoStreamType,
  TRTCScreenCaptureSourceType,
  TRTCVideoEncParam,
  Rect,
  TRTCAudioQuality,
  TRTCScreenCaptureSourceInfo,
  TRTCDeviceInfo,
  TRTCVideoQosPreference,
  TRTCQualityInfo,
  TRTCStatistics,
  TRTCVolumeInfo,
  TRTCDeviceType,
  TRTCDeviceState,
  TRTCBeautyStyle,
  TRTCVideoResolution,
  TRTCVideoResolutionMode,
  TRTCVideoMirrorType,
  TRTCVideoRotation,
  TRTCVideoFillMode,
```

```
        TRTCRoleType,  
        TRTCQuality,  
        TRTCScreenCaptureProperty,  
    };  
    export default TRTCCloud.default;  
},  
],  
});
```

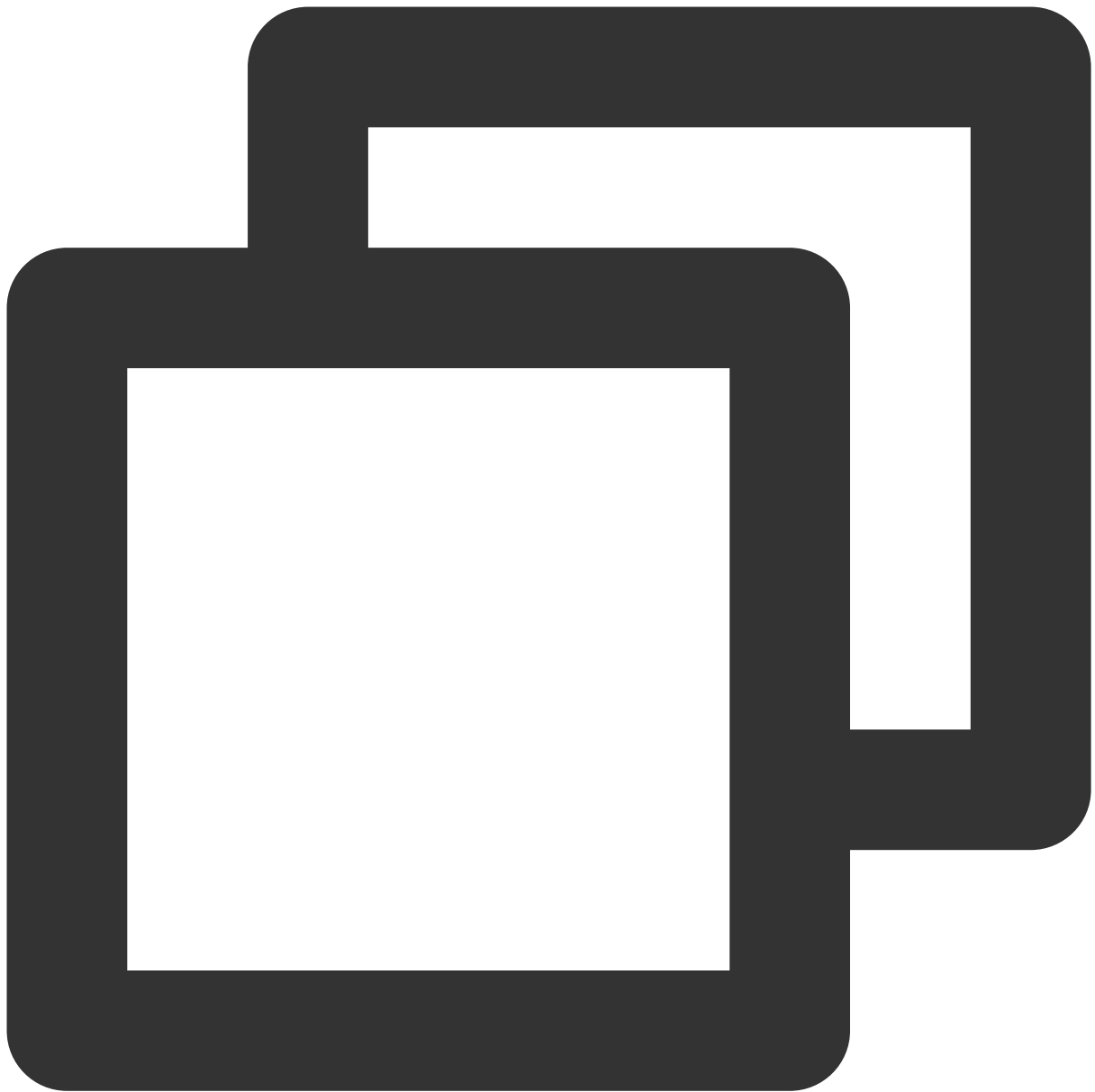
### Step 3: Reference TUIRoomKit Component

#### Note:

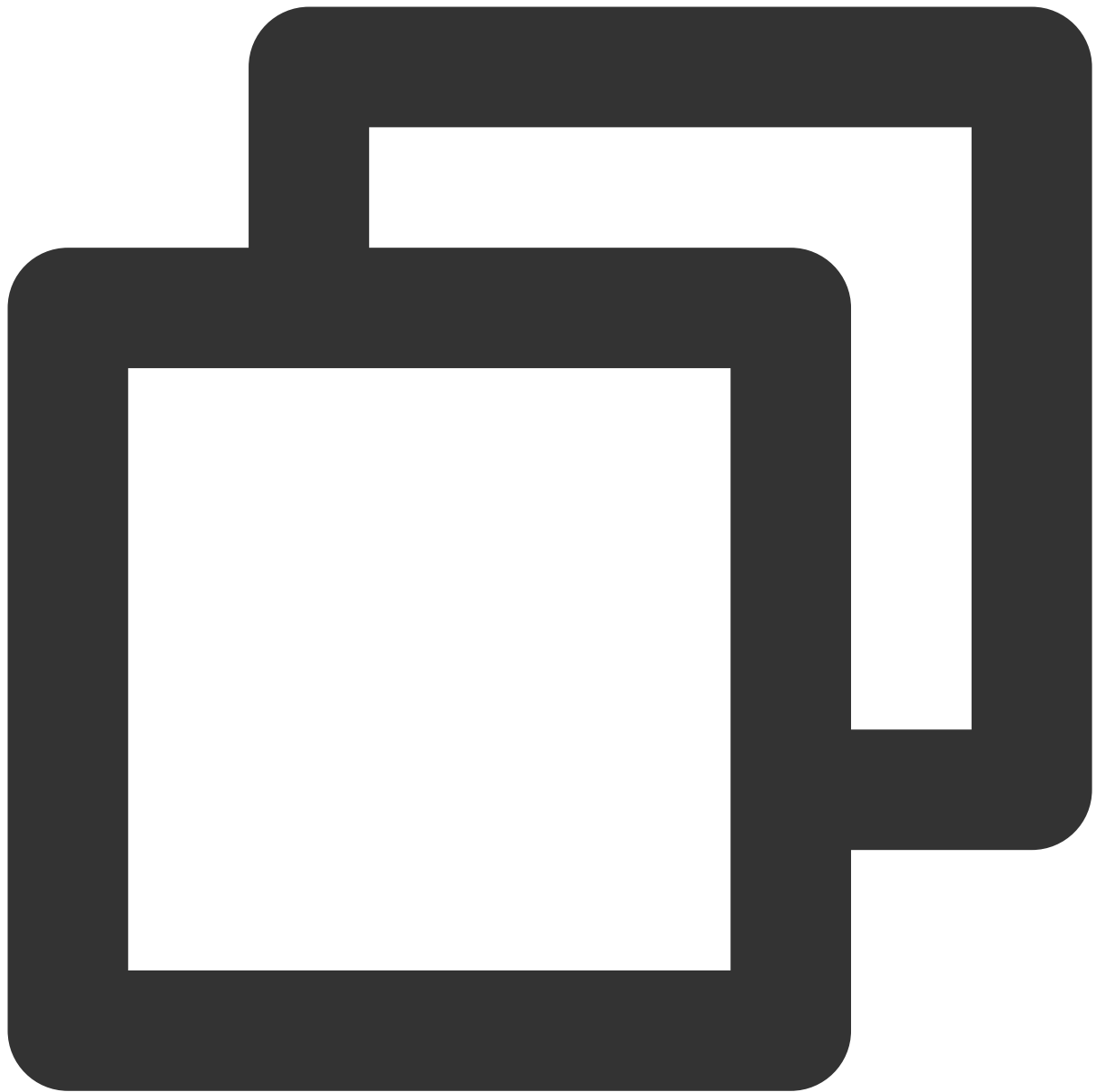
Introduce the ConferenceMainView component, which is defaulted to [Permanent Mode](#) (the component is always displayed, its visibility is not controlled internally. Without business side control, the component will remain visible).

Vue3

Vue2



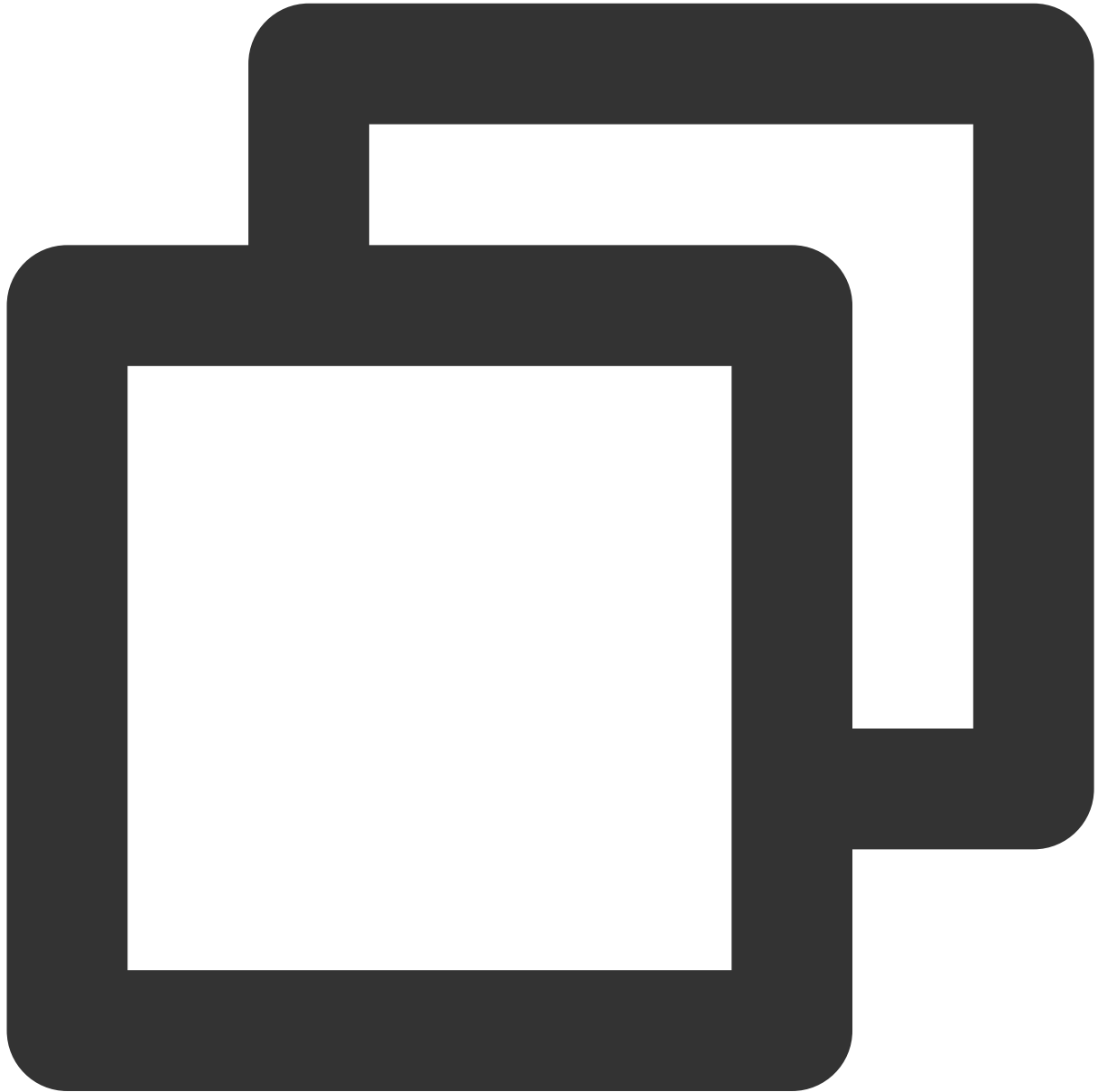
```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue3';
</script>
```



```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script>
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
};
</script>
```

#### Step 4: Log in to TUIRoomKit Component

Before starting a meeting, you need to call the [login](#) interface to log in and get sdkAppId, userId, userSig as described in [Activate Service](#).



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.login({
  sdkAppId: 0, // Replace with your sdkAppId
  userId: '', // Replace with your userId
```

```
userSig: '', // Replace with your userSig
});
```

### Parameter Description

Here is a detailed introduction to several key parameters required in the `login` function:

**sdkAppId**: Obtained in the last step of [Activating Service](#).

**userId**: The ID of the current user, a string type, can only contain letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (\_).

**userSig**: By encrypting the information such as `SDKSecretKey` obtained in [Activating Service](#), `SDKAppID`, `UserID`, etc., one can generate `UserSig`, which is an authentication ticket used by Tencent Cloud to recognize if the current user is eligible to use `TRTC` services. A temporary `UserSig` can be generated through the [Auxiliary Tools](#) in the console.

For more information, see [How to Calculate and Use UserSig](#).

#### Note:

**This step has received the most feedback from developers, with common issues as follows:**

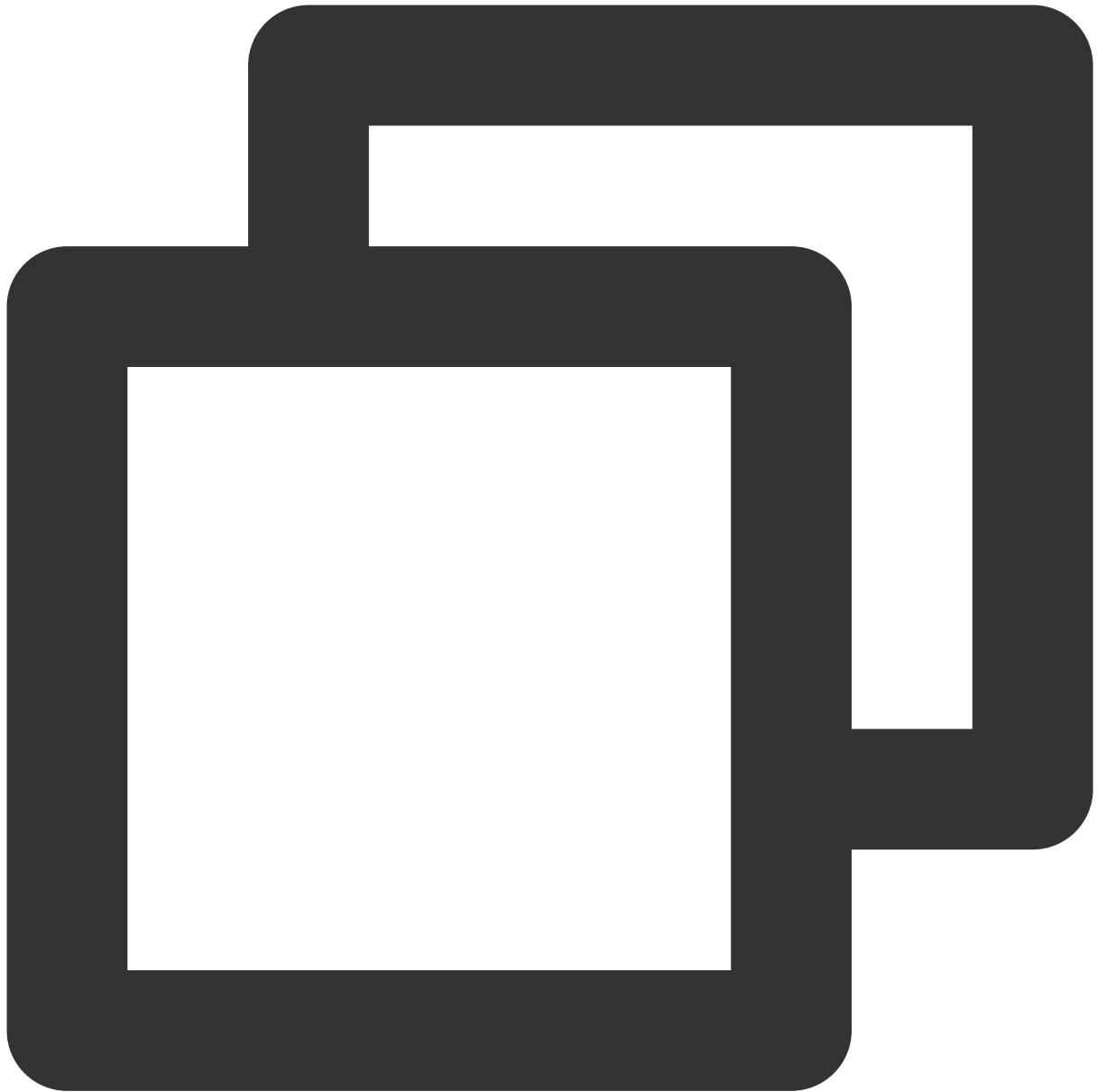
**sdkAppId** is set incorrectly. For domestic sites, the `SDKAppID` usually starts with 140 and is a 10-digit integer. `UserSig` is mistakenly configured as the encryption key (`SecretKey`). `UserSig` is derived from encrypting information like `SDKAppID`, `UserID`, and expiration time with the `SecretKey`, rather than directly setting the `SecretKey` as `UserSig`. **userId** is set to simple strings like "1", "123", "111". Since **TRTC does not support multiple logins with the same UserID**, in a collaborative development scenario, `UserIDs` such as "1", "123", "111" can easily be taken by your colleagues, leading to login failures. Therefore, we recommend using more unique `UserIDs` during testing.

[GitHub](#) sample code uses the `genTestUserSig` function to compute `UserSig` locally to expedite the integration process. However, this approach exposes your `SecretKey` in the code, which is not secure for future upgrades and protecting your `SecretKey`. Therefore, we strongly suggest moving the `UserSig` computation logic to the server-side and requesting a realtime computed `UserSig` from your server each time you use the `TUIRoomKit` component.

### Step Five: Launch a New Meeting

The meeting host can initiate a new meeting by invoking the [start](#) interface. Other participants can refer to the description in [Step Six](#) and join the meeting by calling the [join](#) interface.



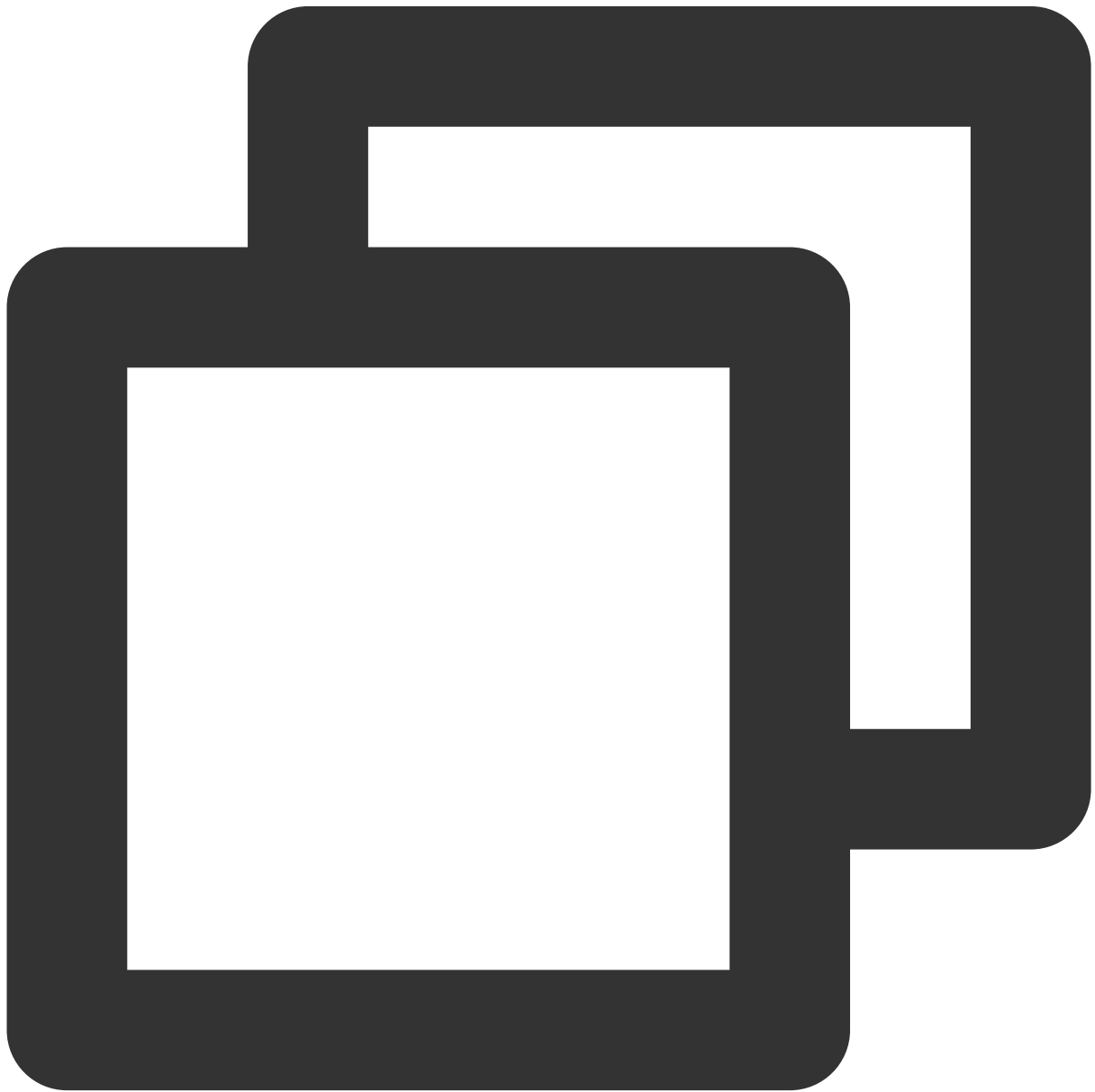


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const startConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.start('123456', {
    roomName: 'TestRoom',
    isSeatEnabled: false,
```

```
        isOpenCamera: false,  
        isOpenMicrophone: false,  
    });  
}  
startConference()
```

## Step Six: Enter an Existing Meeting

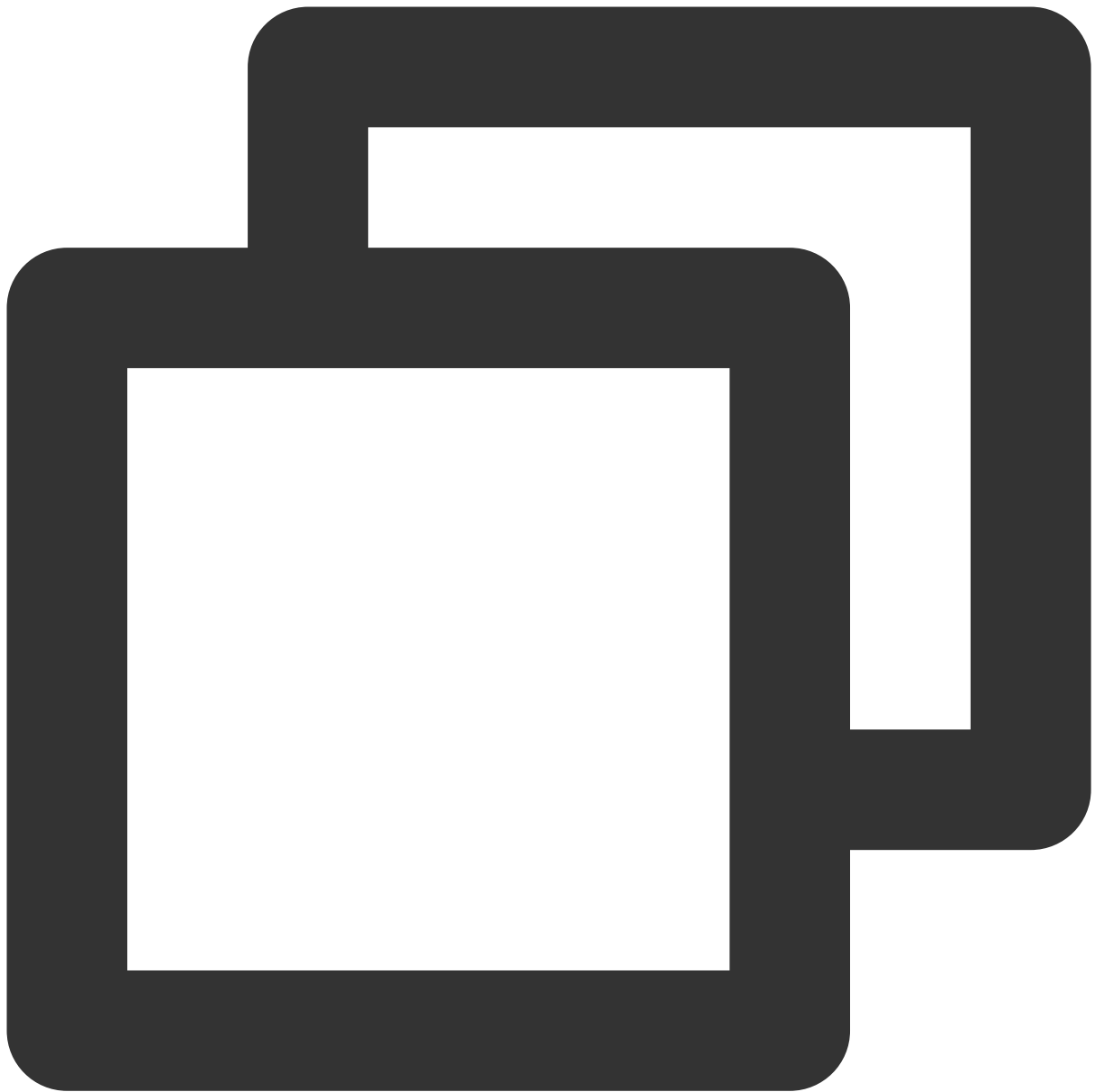
Participants can join a meeting initiated by the host in [Step Five](#) by invoking the [join](#) interface and filling in the corresponding roomId parameter.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const joinConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.join('123456', {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
joinConference()
```

## Development Environment Operation

1. Execute the development environment command. (As an example, this uses a vue3 + vite default project, however, the dev command may differ for different projects. Please adjust according to your own project)



```
npm run dev
```

**Note:**

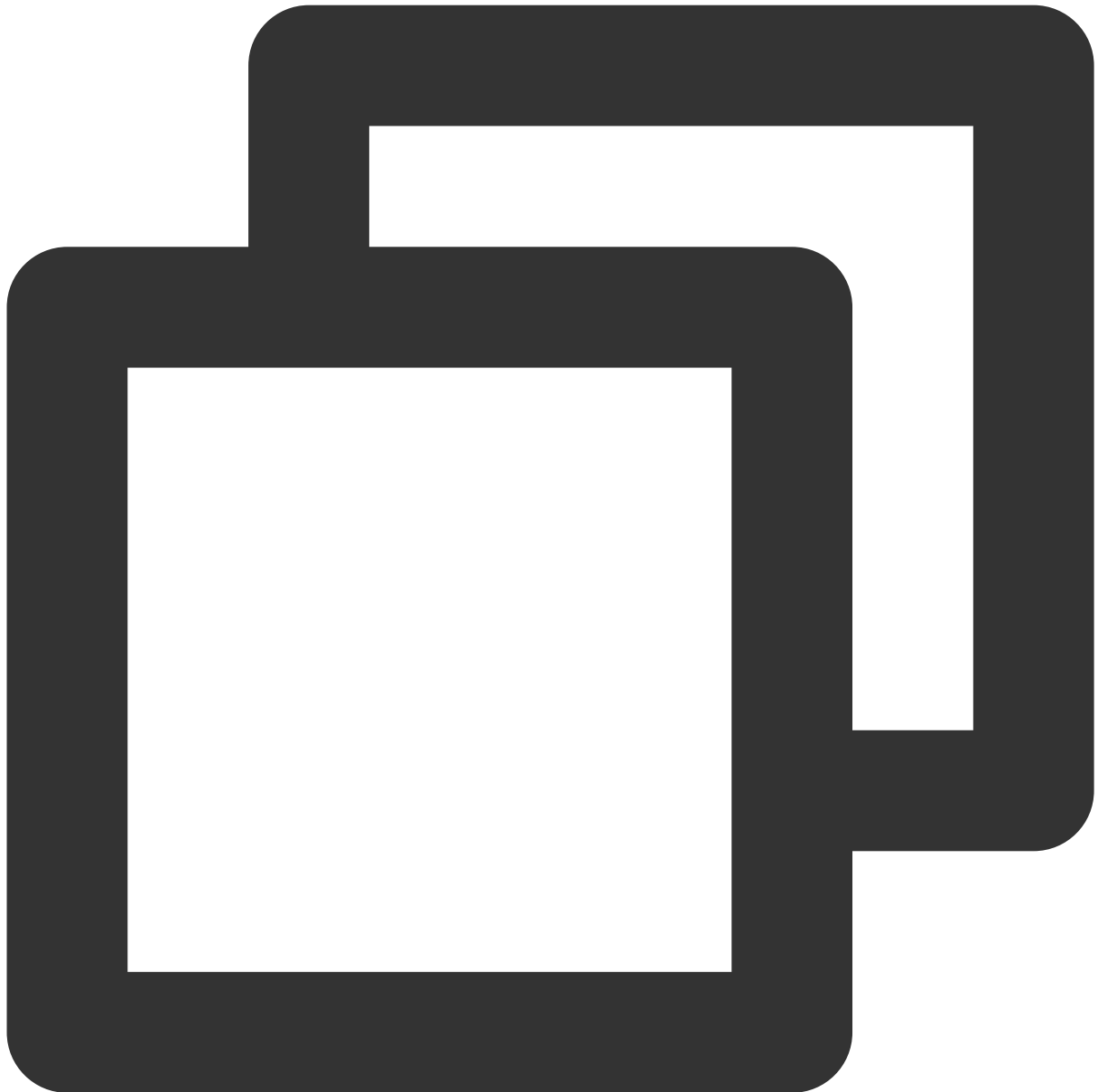
If there are eslint errors in the `src/TUIRoom` directory during execution, you can add the `/src/TUIRoom/` path to the `.eslintignore` file to ignore the eslint checks.

2. Follow the console prompts to open the page in a browser, e.g., `http://localhost:3000/`.

3. Experience the TUIRoomKit component features.

## Production environment deployment

1. Pack the dist file.



```
npm run build
```

2. Deploy the dist file to the server.

**Note:**

The production environment requires the use of an HTTPS domain name.

[TUIRoomKit](#)

[TUIRoom Demo Quick Run](#)[UI Customization \(TUIRoomKit\)](#)[Frequently Asked Questions](#)

## Communication and feedback

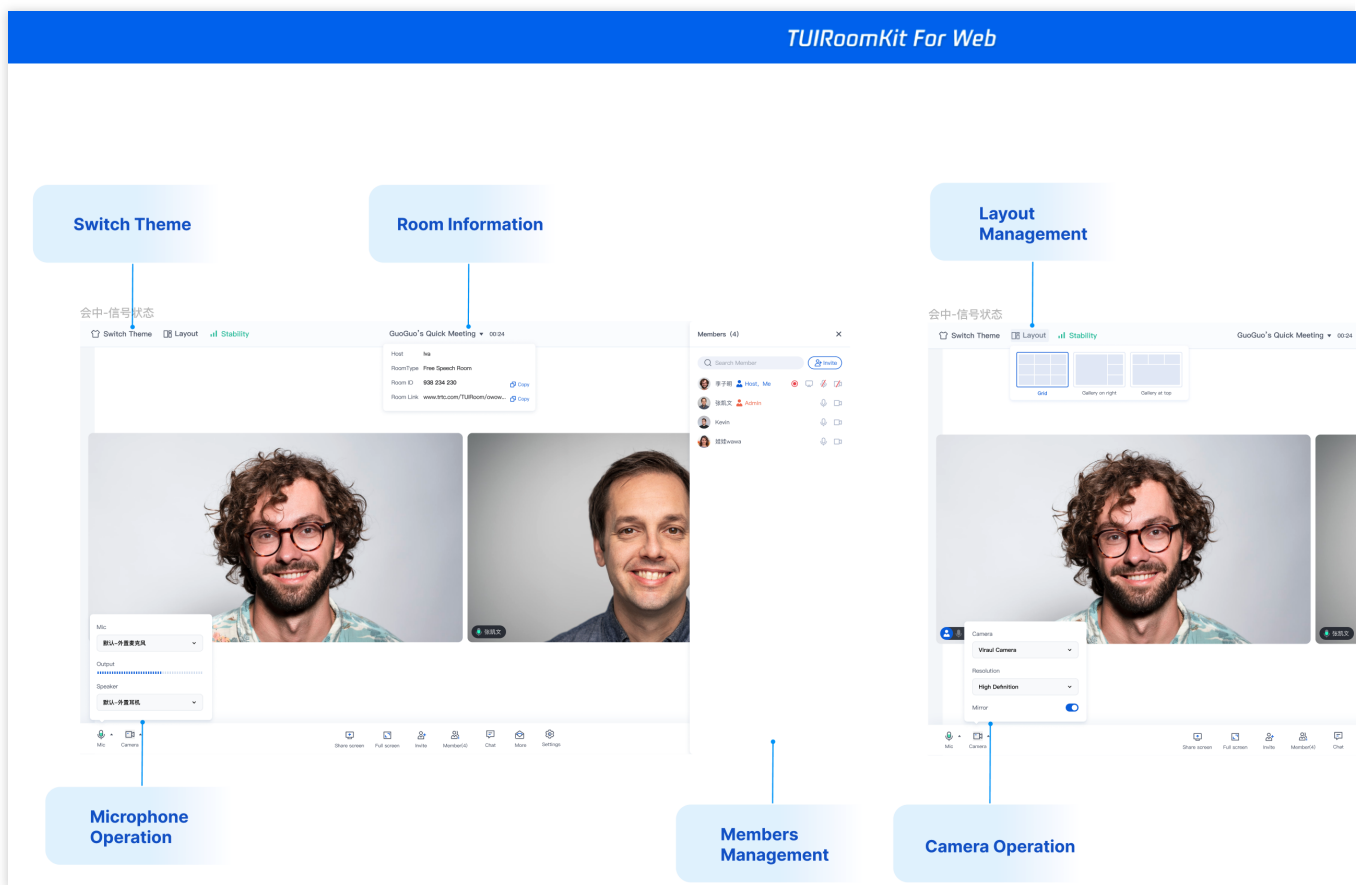
If you have any needs or feedback, you can contact: [info\\_rtc@tencent.com](mailto:info_rtc@tencent.com).

# Web

Last updated : 2024-08-09 11:03:36

TUIRoomKit is a group audio and video room UI component that provides rich features such as room management, audio and video control, screen sharing, member management, microphone management, instant messaging, and custom layout switching. It also supports one-click switching between Chinese and English UI languages and between different app themes.

This document describes the process of integrating TUIRoomKit (Web), helping you quickly implement features for different business scenarios, such as enterprise meetings, online education, medical consultations, online inspections, and remote loss assessments.



## TUIRoomKit Demo Experience

You can visit our online [TUIRoomKit demo](#) to experience more features of TUIRoomKit.

You can also visit [Github](#) to download the TUIRoomKit code and refer to the README.md file to run the TUIRoomKit Web sample project.

## Environment Preparation

Node.js version: Node.js  $\geq 16.19.1$  (It is recommended to use the official LTS version, and the npm version should match the node version).

Modern browser, [supporting WebRTC APIs](#).

## Integrating TUIRoomKit Component

### Note :

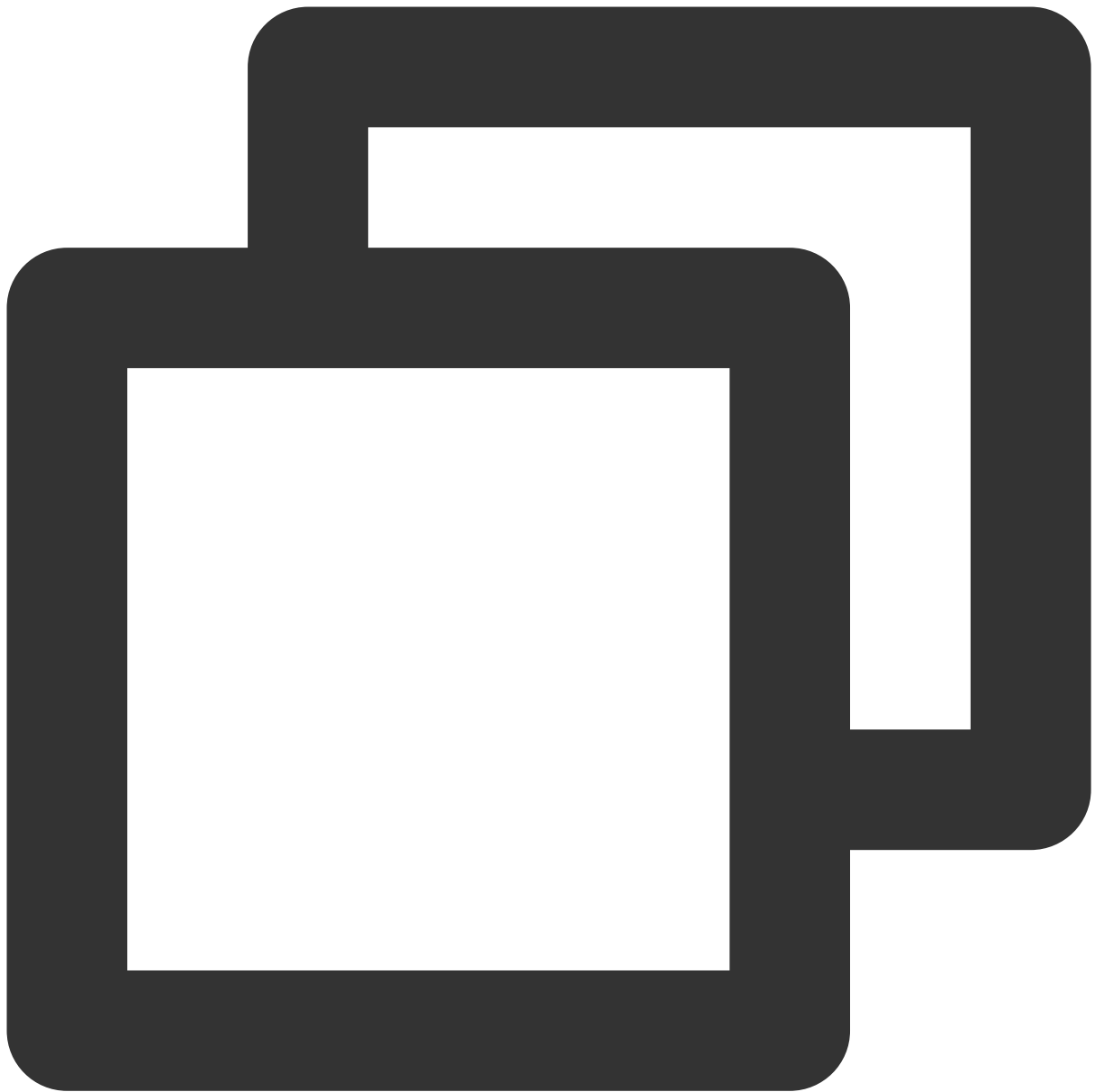
If you **don't have a vue project**, you can directly refer to the [Run Sample Demo](#) run through [Github](#) sample project.  
If you need to integrate it into an existing project, please follow the steps below for integration.

### Step 1: Install Dependencies

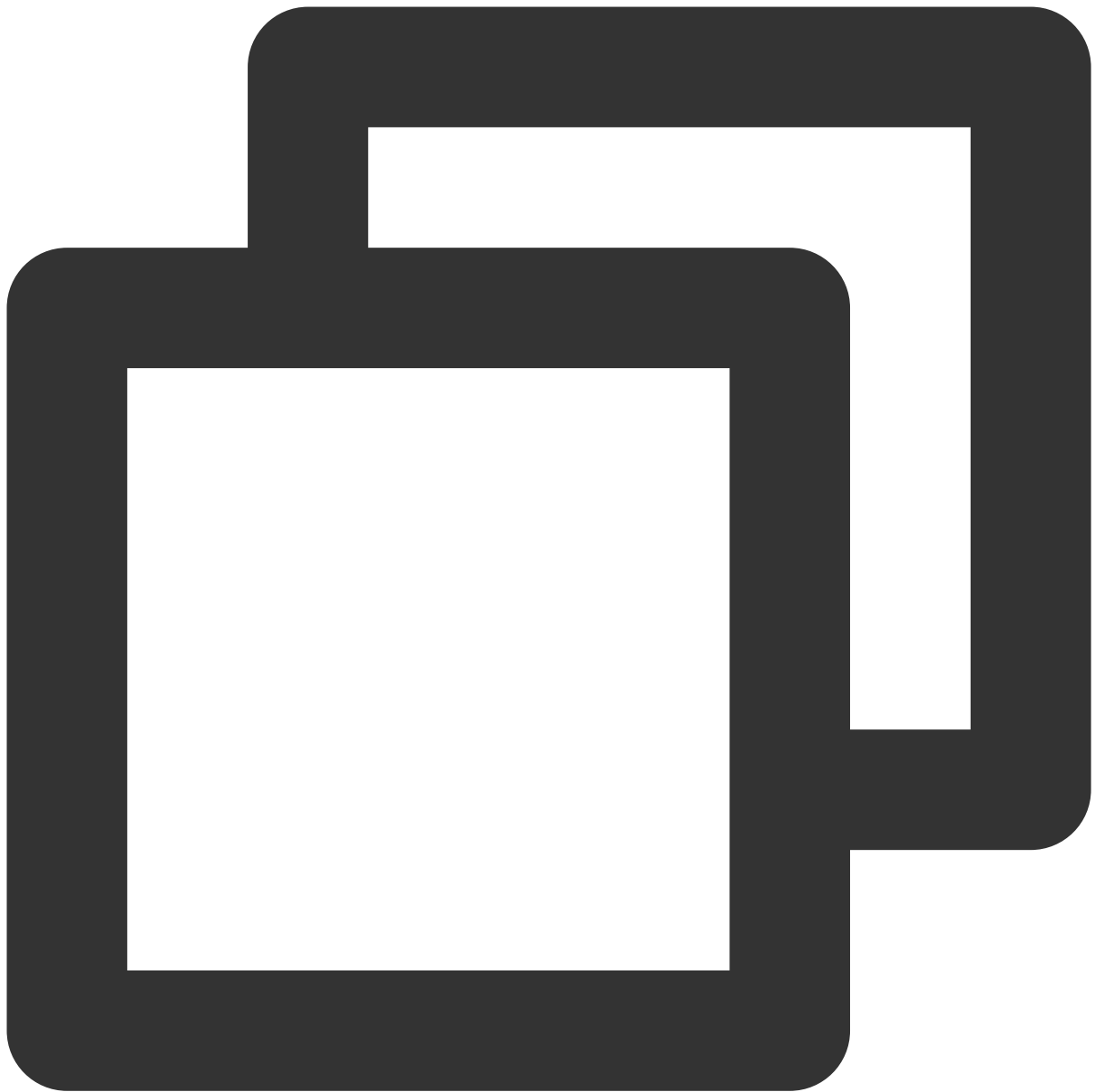
Vue3

Vue2





```
npm install @tencentcloud/roomkit-web-vue3 pinia --save
```



```
# Please note that the required Vue version is >= 2.7.16. If the installation fails  
# please check if your Vue version is supported.  
npm install @tencentcloud/roomkit-web-vue2.7 pinia
```

**Note :**

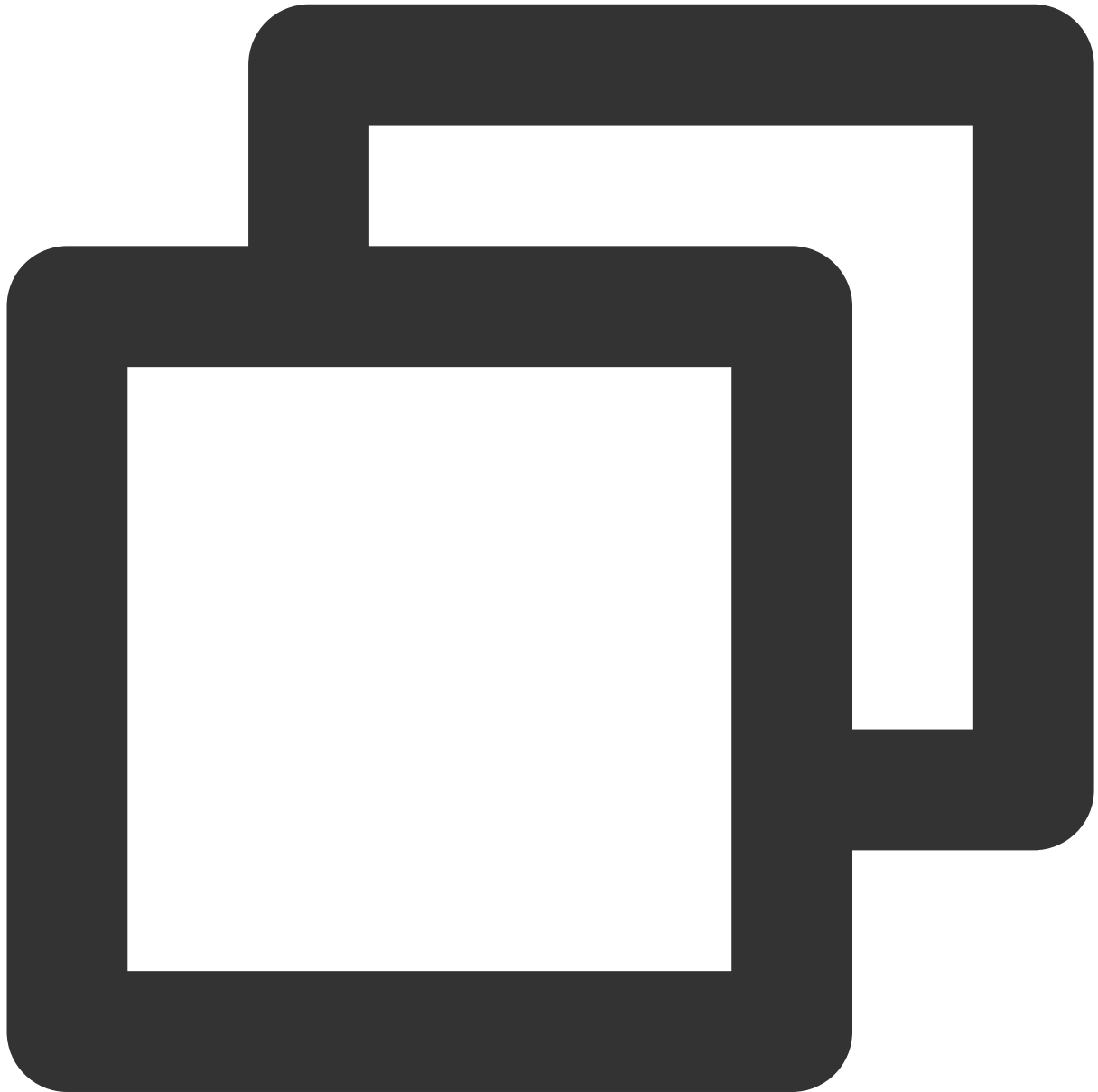
TUIRoomKit package provides a pre-meeting preview component, an in-meeting component, and methods for starting meetings, joining meetings, and fine-tuning the interface. For more, see [RoomKit API](#). If these APIs don't meet your business needs, you can refer to [UIKit source code export](#) for accessing the TUIRoomKit source code.

**Step 2: Project Engineering Configuration**

**Register Pinia:** TUIRoom uses Pinia for room data management, and you need to register Pinia in the project entry file. The project entry file is `src/main.ts`.

Vue3

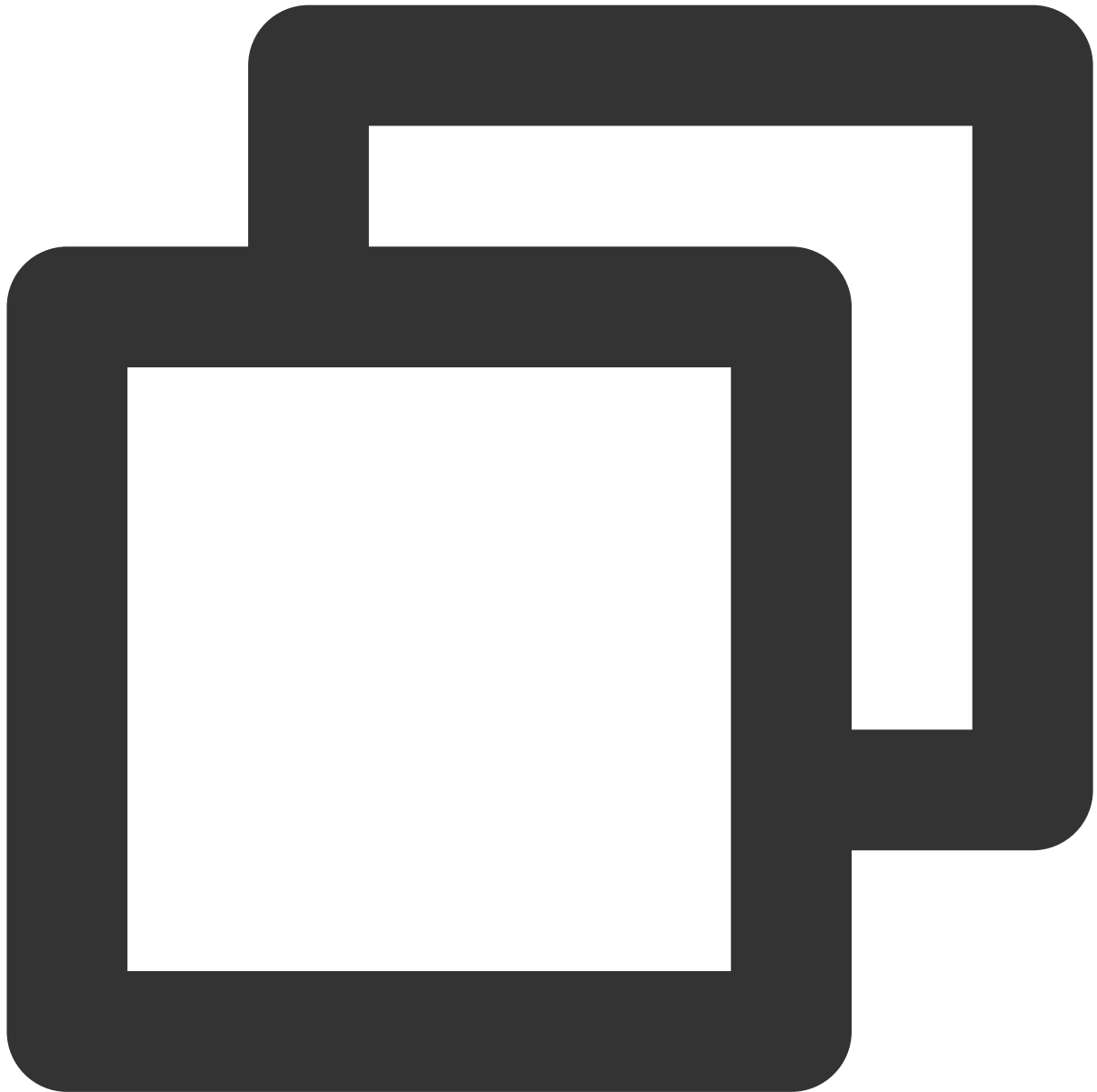
Vue2



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

const app = createApp(App);
// register pinia
app.use(createPinia());
```

```
app.mount('#app')
```



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

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
```

```
}).$mount('#app');
```

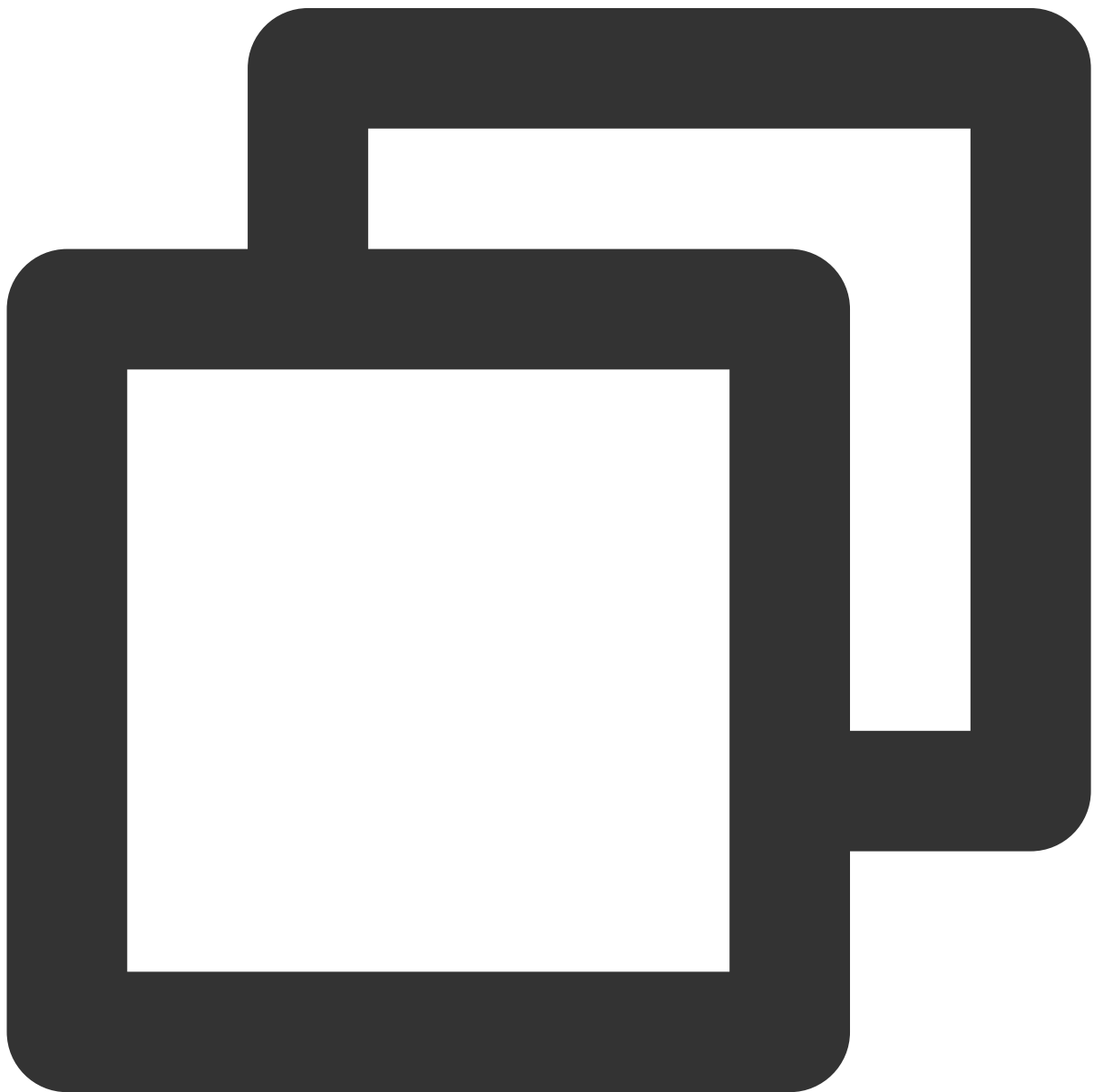
### Step 3: Import the TUIRoom component

**Note:**

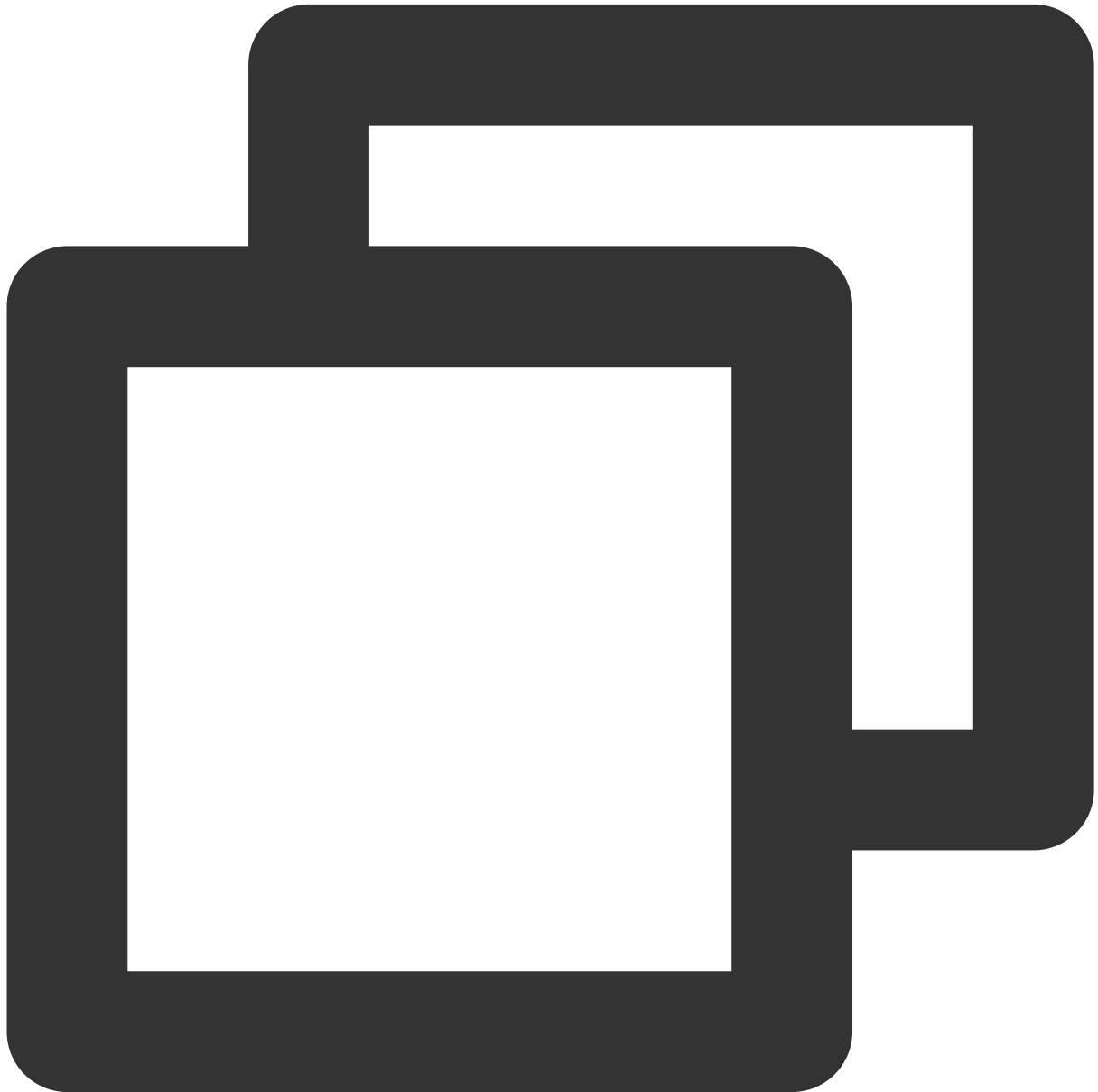
Import the ConferenceMainView component, which is by default in the [permanent mode](#) (the component is always displayed, and its display and hiding are not controlled internally. If the business end does not control it, the component will always remain displayed).

Vue3

Vue2



```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue3';
</script>
```

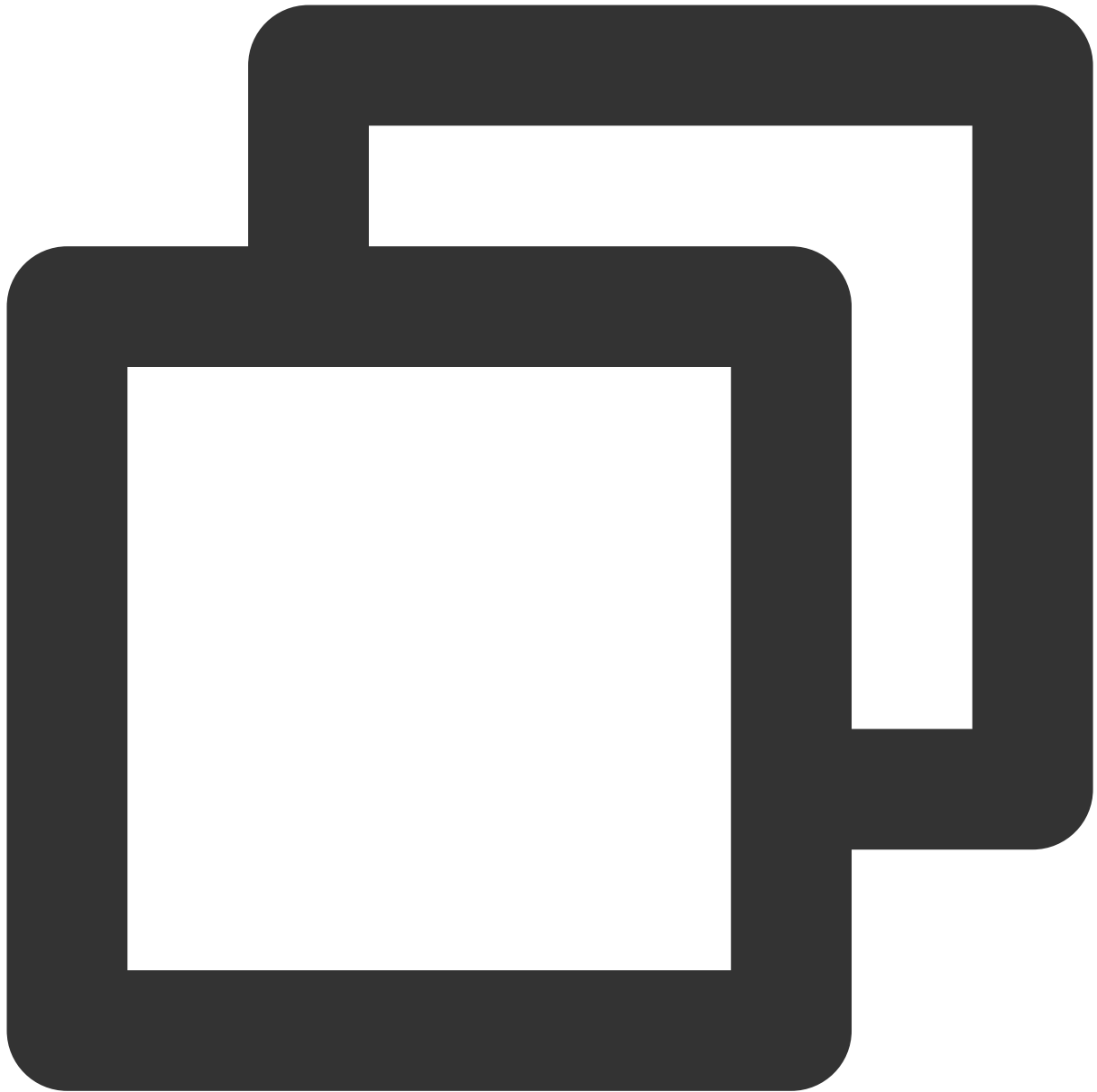


```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
```

```
<script>
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
};
</script>
```

#### Step 4: Logging into the TUIRoomKit Component

Before initiating a meeting, it is necessary to invoke the [login](#) interface for authentication. To obtain the sdkAppId, userId, and userSig, refer to the [service activation](#).



```
import { conference } from '@tencentcloud/roomkit-web-vue3';
await conference.login({
  sdkAppId: 0, // Replace with your sdkAppId
  userId: '', // Replace with your userId
  userSig: '', // Replace with your userSig
});
```

Parameter Explanation:

**sdkAppId:** Acquired in the final step of [service activation](#).



**userId:** The current user's ID, a string type, allowing only English letters (a-z, A-Z), numbers (0-9), hyphens (-), and underscores (\_).

**userSig:** Obtained by encrypting information such as `SDKAppID` and `UserID` with the `SDKSecretKey` retrieved during [service activation](#). `UserSig` is an authentication ticket used by Tencent Cloud to verify if the current user can access TRTC services. You can generate a temporarily available `UserSig` through the [Auxiliary Tool](#) in the console.

For more information, refer to the guide on '[How to Calculate and Utilize UserSig](#).'

#### Note:

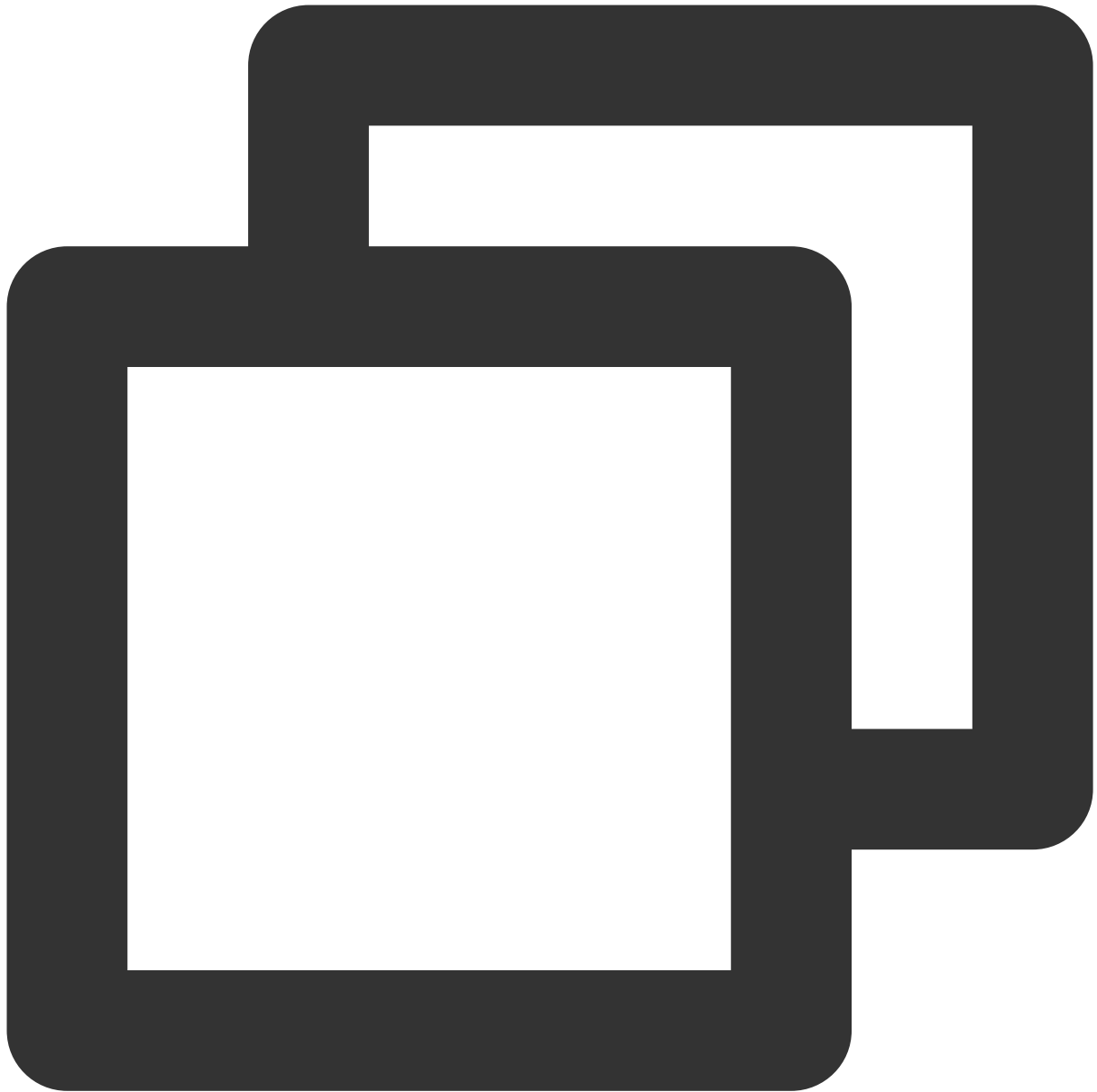
**This step is also the step that we have received the most feedback from developers so far. Frequently asked questions are as follows:**

The `sdkAppId` setting is wrong. The `SDKAppID` of domestic sites is generally a 10-digit integer starting with 140. `UserSig` was mismatched to the encryption key (`SecretKey`). `UserSig` is obtained by using `SecretKey` to encrypt information such as `SDKAppID`, `UserID`, and expiration time, instead of directly configuring `SecretKey` to `UserSig`. `userId` is set to simple strings such as "1", "123", "111", etc. Since TRTC does not support multi-terminal login with the same `UserID`, when multiple people collaborate in development, the form is like "1", "123", "111". Such a `UserID` can easily be occupied by your colleagues, causing login failure. Therefore, we recommend that you set some highly identifiable `UserIDs` during debugging.

The sample code in [Github](#) uses the `genTestUserSig` function to calculate `UserSig` locally to allow you to run through the current access process faster, but this solution will expose your `SecretKey` in the code, which is not conducive to your subsequent upgrades and protection. Your `SecretKey`, so we strongly recommend that you put the calculation logic of `UserSig` on the server side, and request the real-time calculated `UserSig` from your server every time you use the `TUIRoomKit` component.

## Step 5: Initiate a new meeting

The conference host can initiate a new conference by calling the [start](#) interface. Other participants can refer to the description in [step 6](#) and call the [join](#) interface to [join](#) the conference.

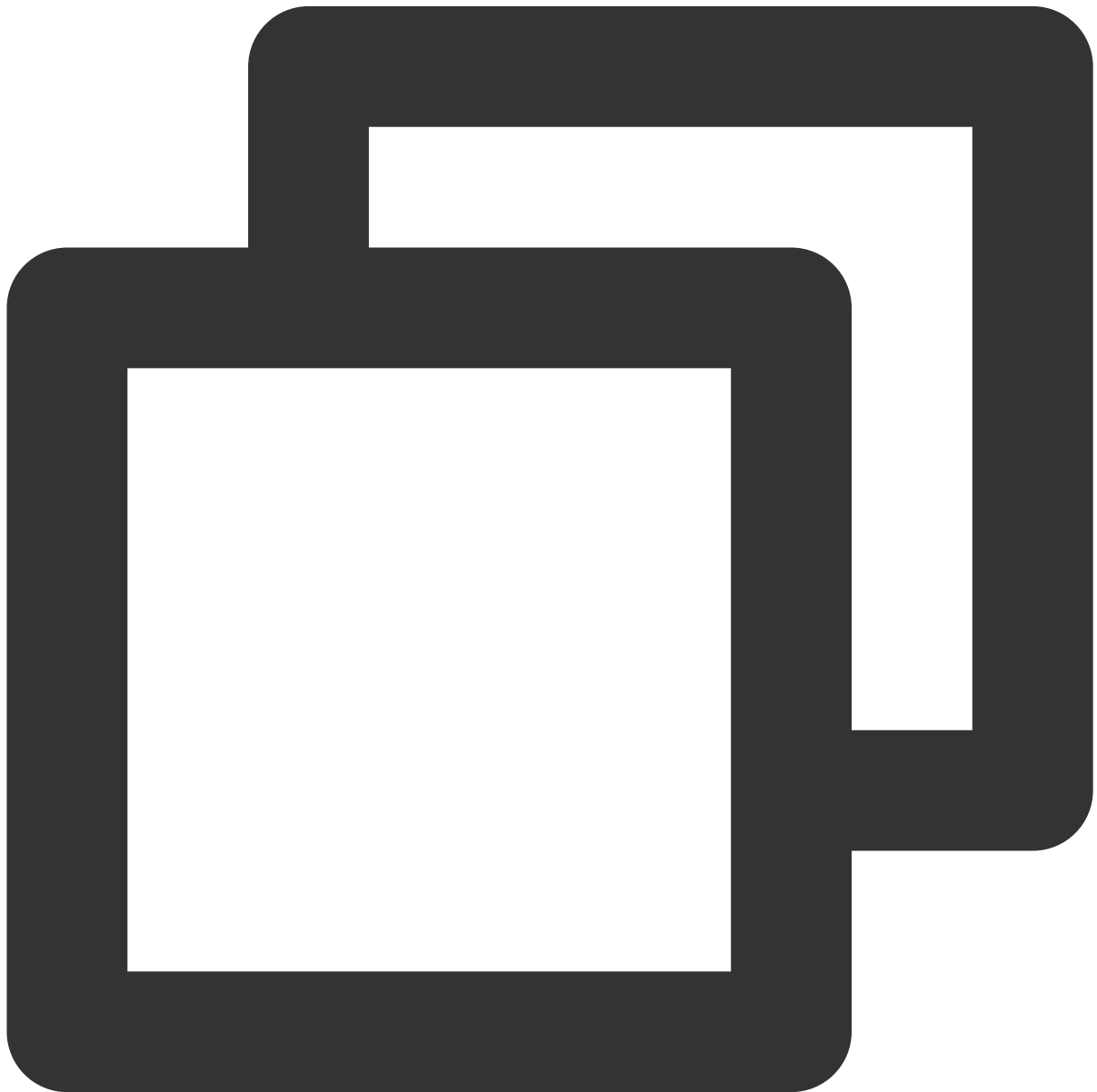


```
import { conference } from '@tencentcloud/roomkit-web-vue3';
const startConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.start('123456', {
    roomName: 'TestRoom',
    isSeatEnabled: false,
    isOpenCamera: false,
```

```
    isOpenMicrophone: false,  
  });  
}  
startConference()
```

## Step 6: Enter an existing meeting

Participants can [join](#) the conference initiated by the conference host in [step 5](#) by calling the join interface and filling in the corresponding roomId parameter.

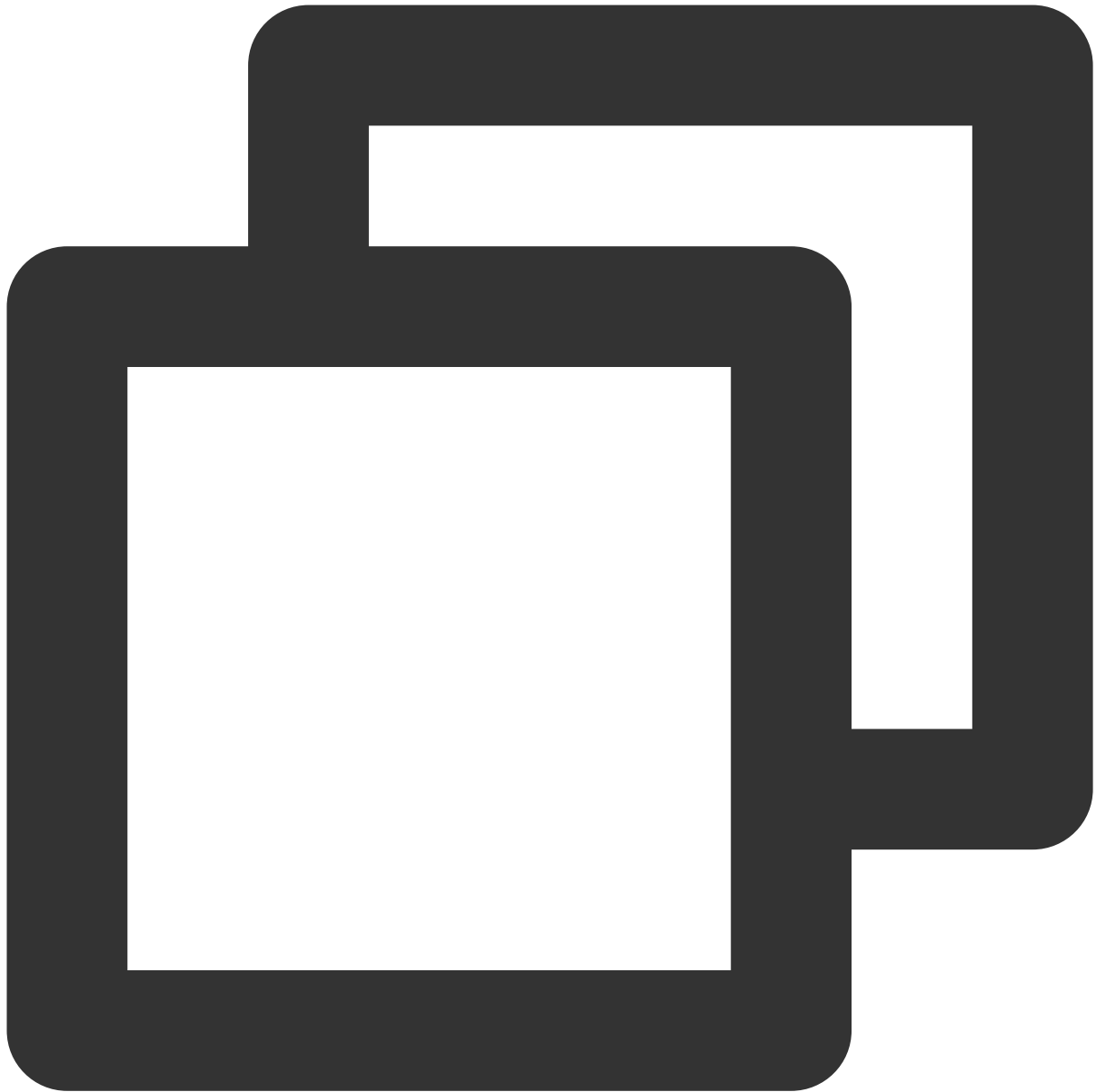


```
import { conference } from '@tencentcloud/roomkit-web-vue3';
```

```
const joinConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.join('123456', {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
joinConference()
```

## Development environment running

1. Execute the development environment command. (Here we take the vue3 + vite default project as an example. The dev instructions of different projects may be different. Please adjust according to your own project)

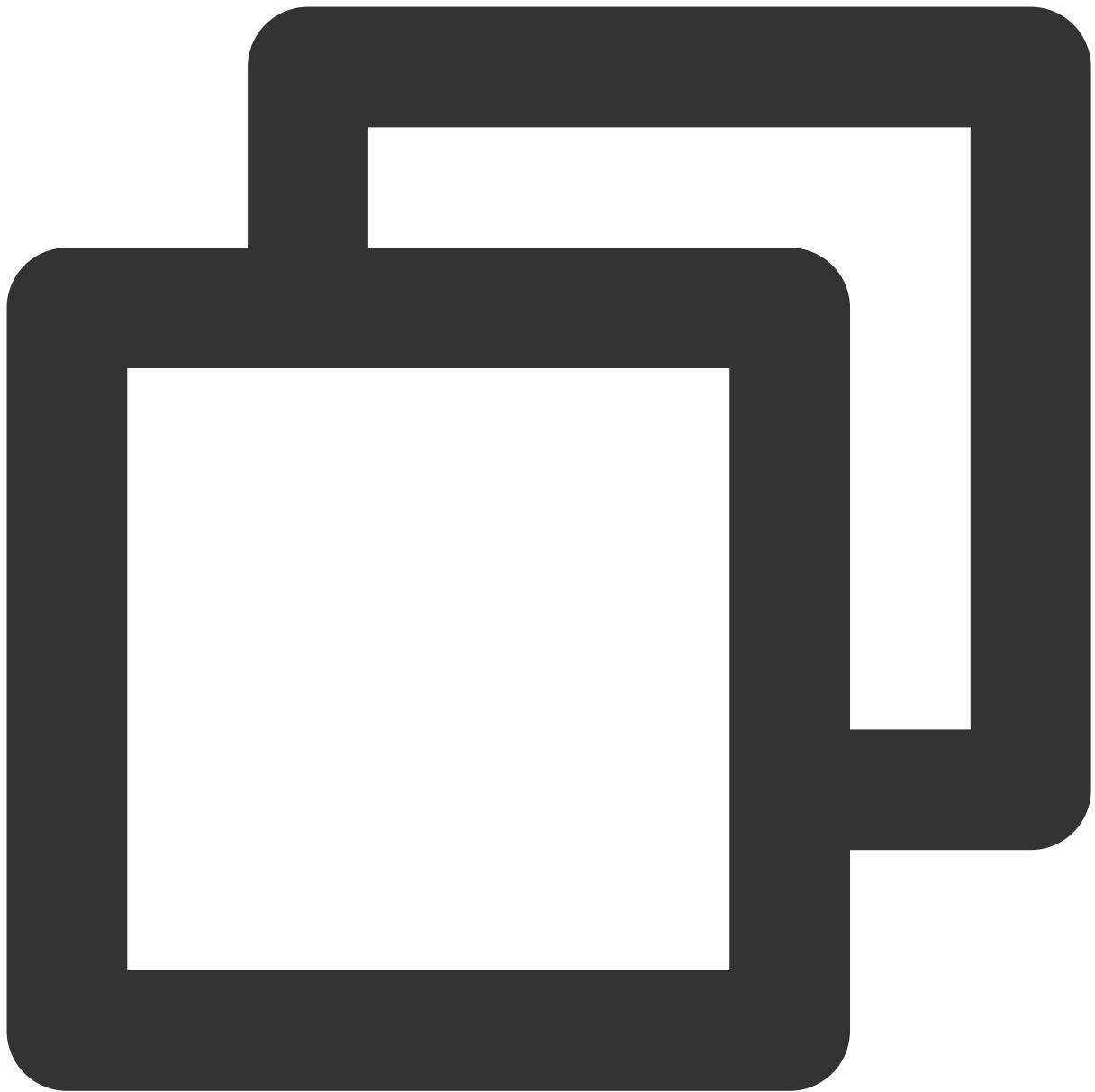


```
npm run dev
```

2. According to the console prompts, open the page in the browser, such as: <http://localhost:3000/>.
3. Experience the TUIRoomKit component functions.

## Production environment deployment

1. Package the dist file.



```
npm run build
```

2. Deploy the dist file to the server.

**Note:**

The production environment requires the use of an HTTPS domain name.

## Other documents

[TUIRoomKit](#)[TUIRoom Demo Quick Start](#)[interface customization \(TUIRoomKit\)](#)[common problem](#)

## Communication and feedback

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

# Flutter

Last updated : 2024-07-05 19:09:37

This document describes how to quickly integrate the Conference (TUIRoomKit) component. The following steps generally take about one hour to complete, after which you can implement audio and video room features with ready-made UI.

## Environment Preparation

Platform	Version
Flutter	3.0.0 and above versions.
Android	Android 4.1 (SDK API level 16) or later (Android 5.0 (SDK API level 21) or later is recommended). Android Studio 3.5 or later (Gradle 3.5.4 or later). Mobile phone on Android 4.1 or later.
iOS	iOS 12.0 and higher.

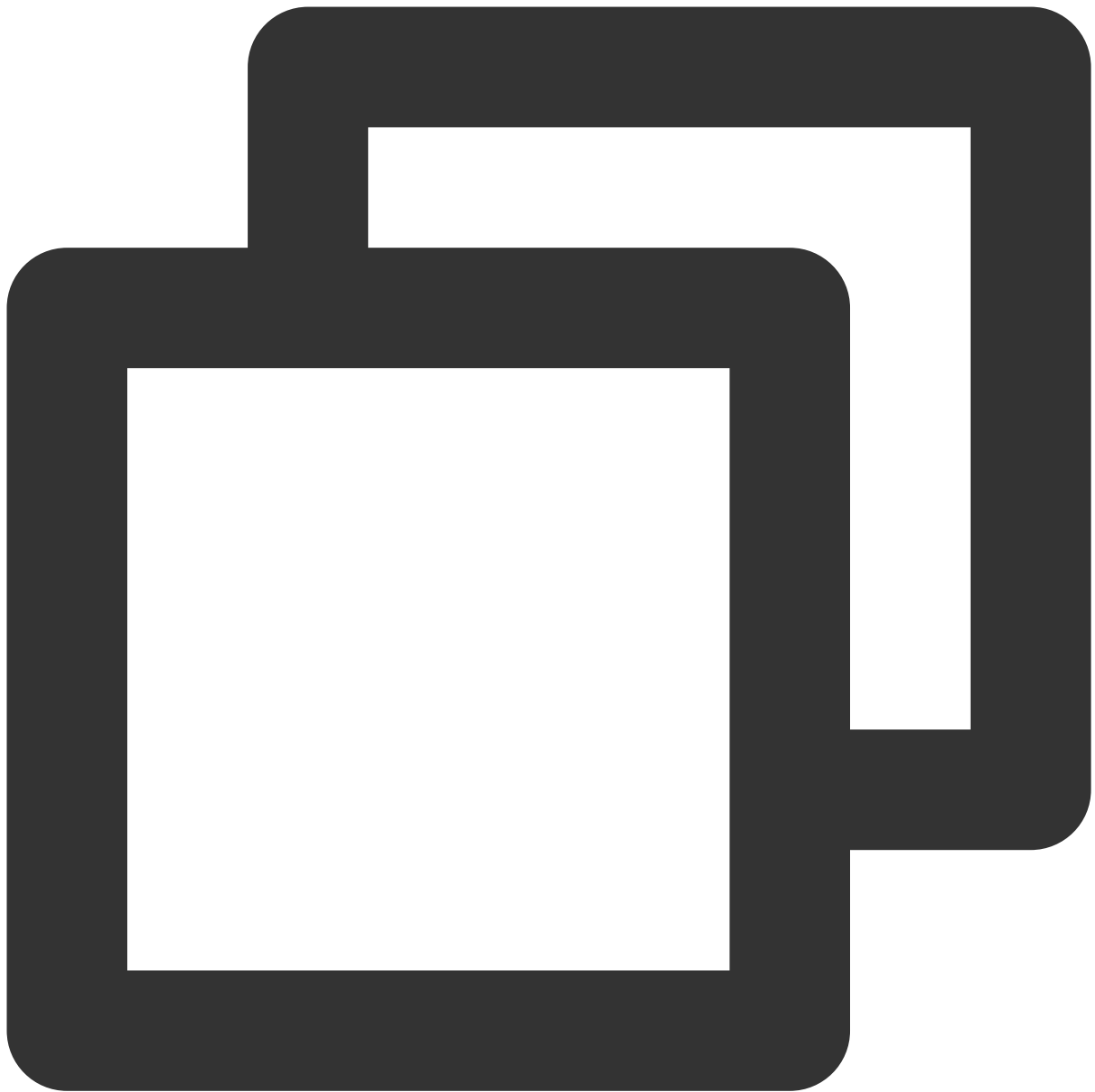
## Step 1: Activate the Service

Before initiating a meeting with TUIRoomKit, you need to activate the exclusive multi-person audio and video interaction service for TUIRoomKit on the console. For specific steps, please refer to [Activate Service](#).

## Step 2: Integrate the TUIRoomKit Component

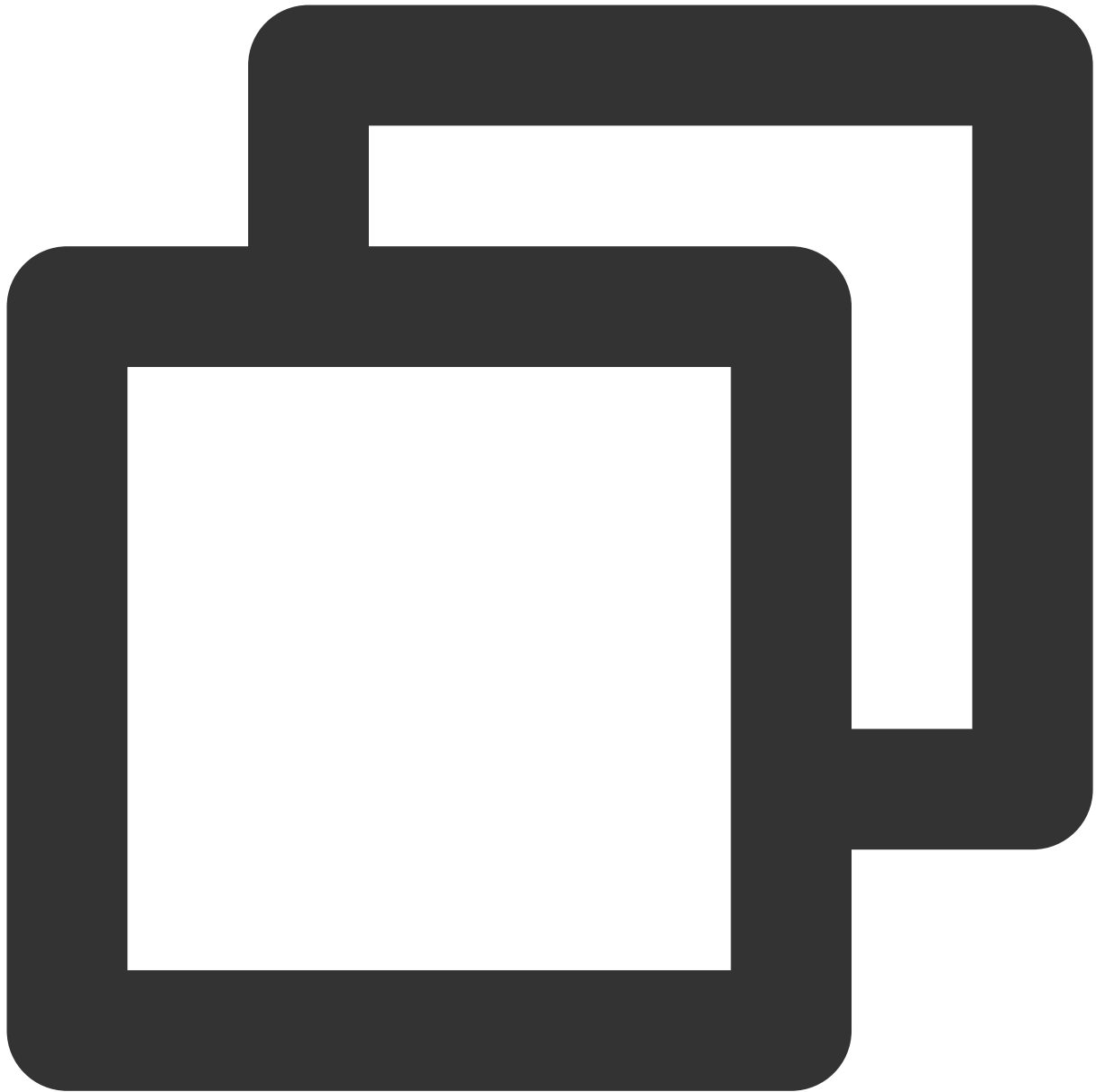
Add the [tencent\\_conference\\_uikit](#) plugin dependency in pubspec.yaml file in your project.





```
dependencies:  
  tencent_conference_uikit: The latest version
```

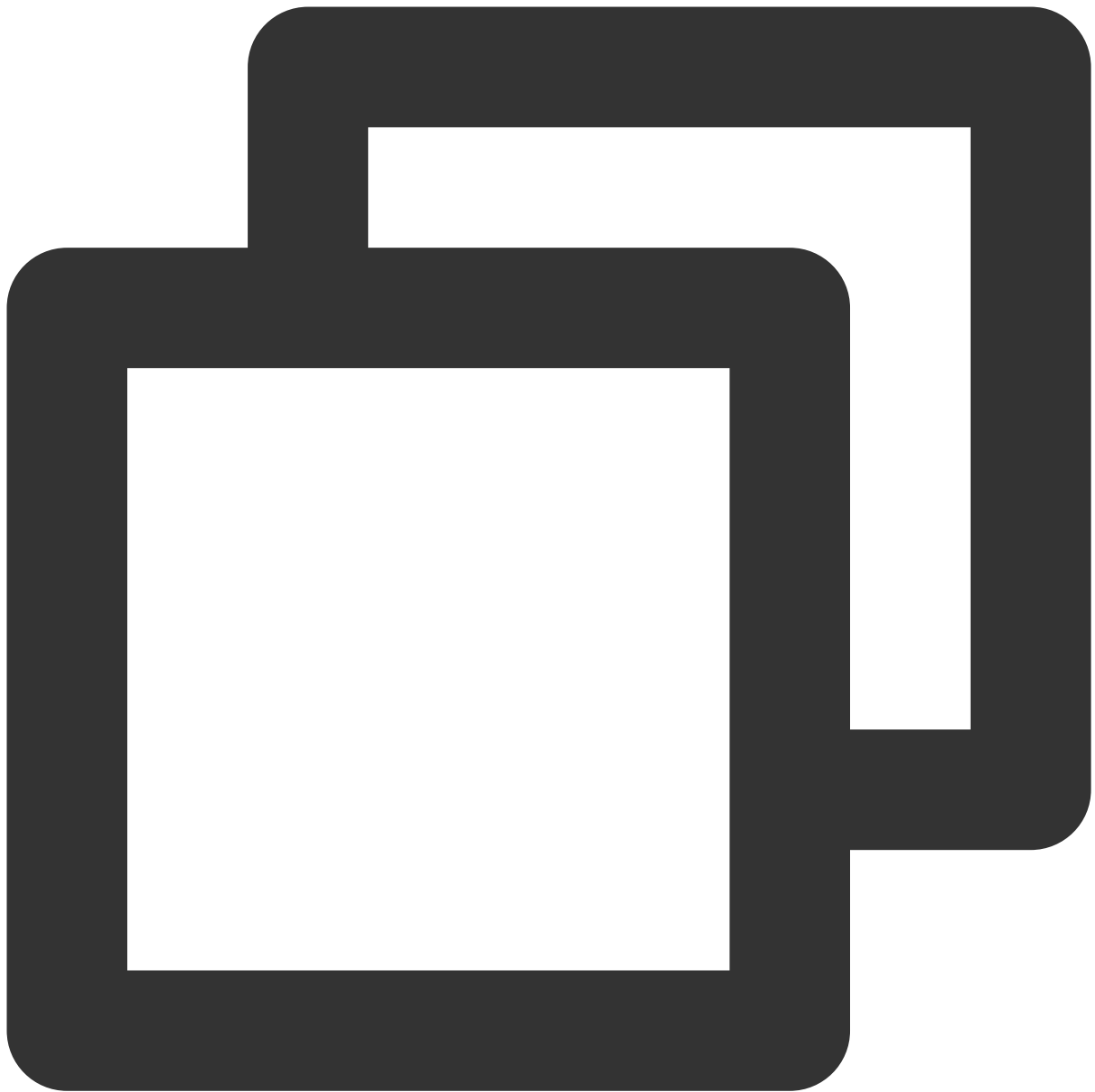
Execute the following command to install the plugin.



```
flutter pub get
```

## Step 3: Complete Project Configuration

Since the **tencent\_conference\_uikit** has utilized the relevant features of the `GetX` state management library, you need to replace `MaterialApp` with `GetMaterialApp` in your application. Alternatively, you can set the `navigatorKey` attribute in your `MaterialApp` to `Get.key` for the same effect.



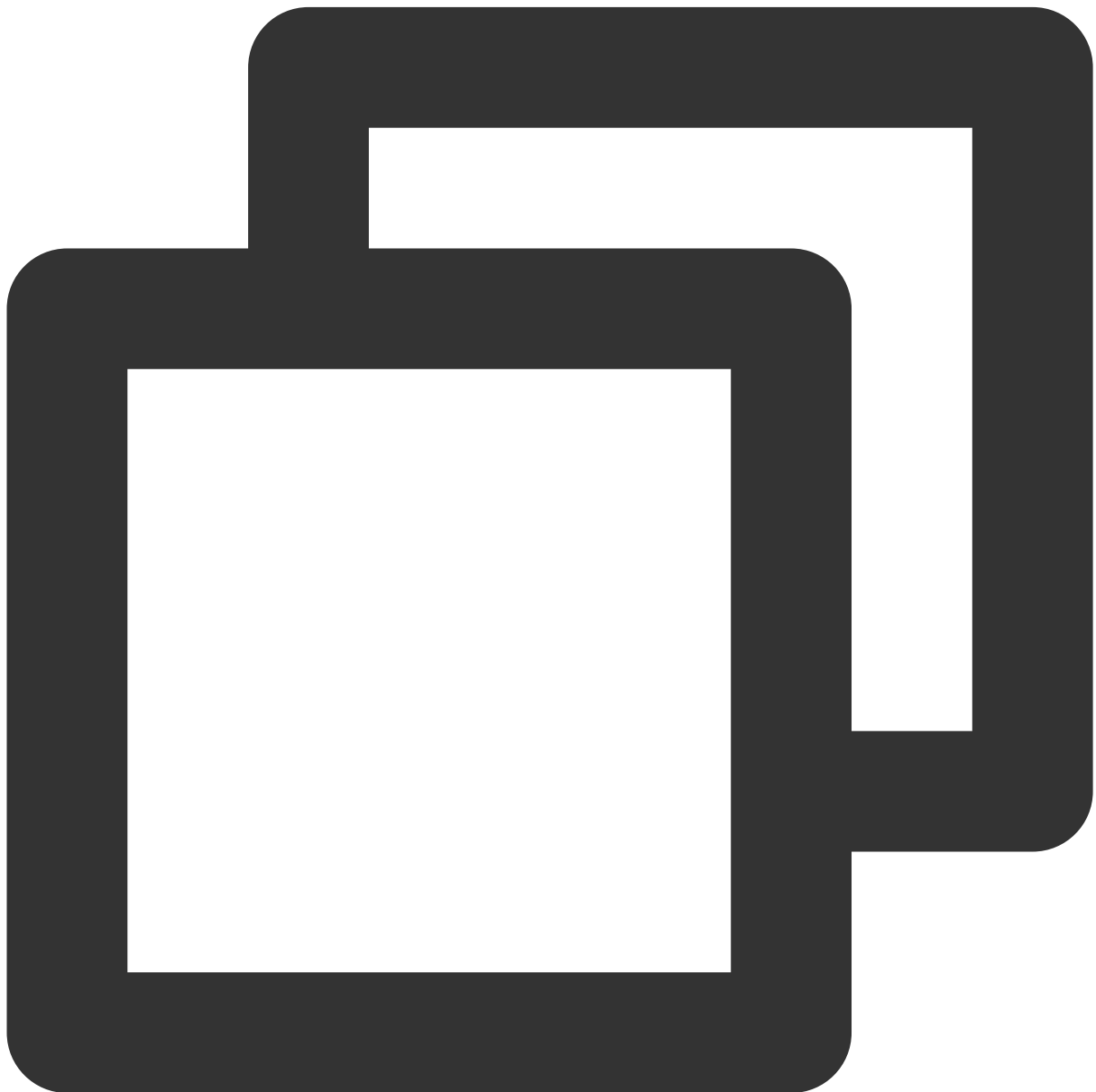
```
// This step requires importing the get package before proceeding.  
// Since the tencent_conference_uikit already has a dependency on get, you don't ne  
import 'package:get/get.dart';  
  
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return GetMaterialApp( // Use GetMaterialApp to replace MaterialApp  
      // Your original MaterialApp content  
    );  
  }  
}
```

```
}
```

Open your project with **Xcode**, choose **Targets > Building Settings > Deployment**, and set **Strip Style** to **Non-Global Symbols** to maintain the necessary global symbol information.

If you need to use audio and video features on **iOS**, authorize the microphone and camera usage rights (**Android** has declared the relevant permissions in the SDK, and no manual configuration is necessary).

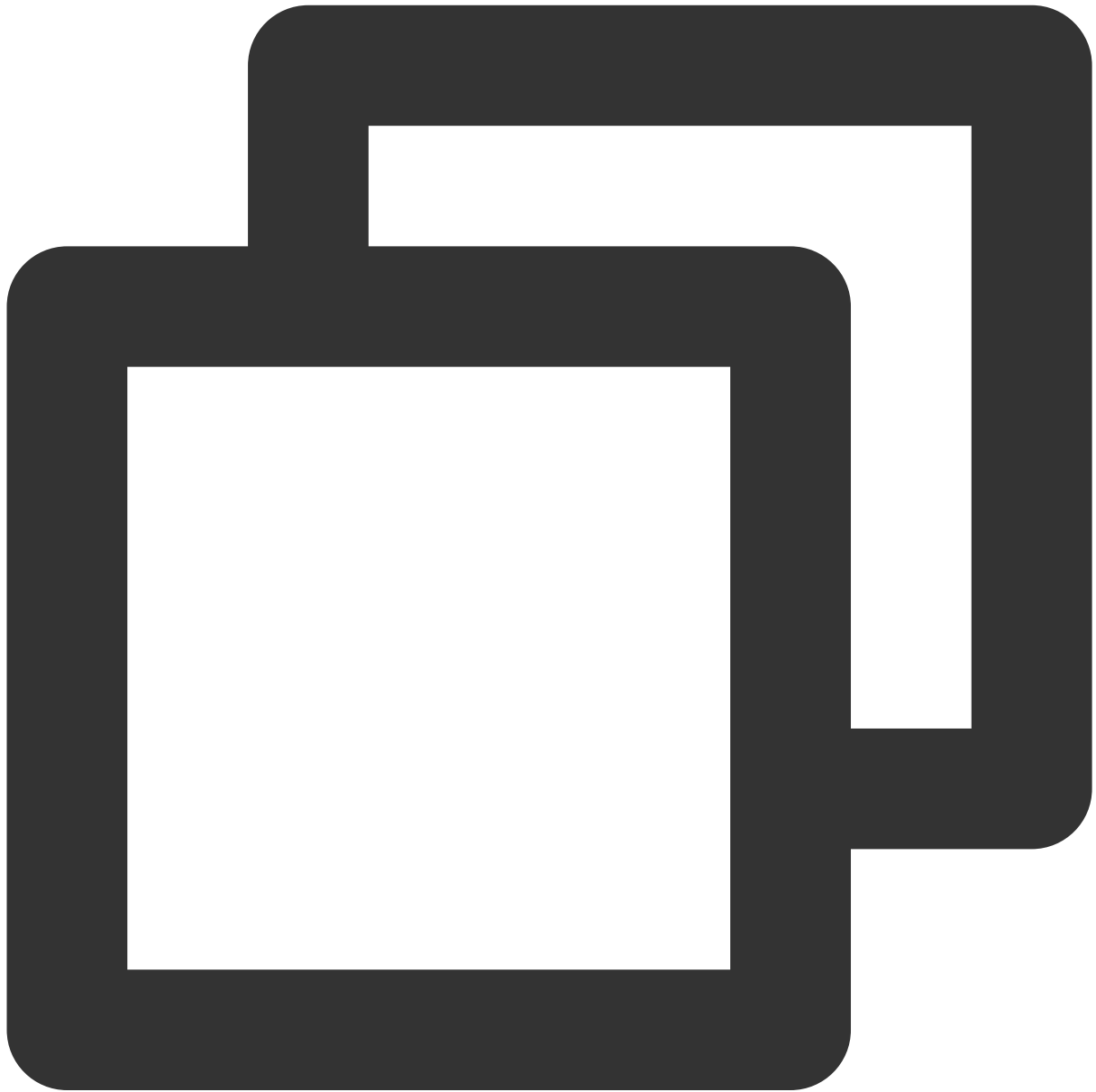
Add the following two items to your application's **Info.plist**. They correspond to the messages in the popup licensing dialog box while microphone and camera permissions are requested.



```
<key>NSCameraUsageDescription</key>
```

```
<string>TUIRoom requires access to your camera.</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUIRoom requires access to your microphone.</string>
```

After the above are added, add the following preprocessor definitions in your **ios/Podfile** to enable camera and microphone permissions.

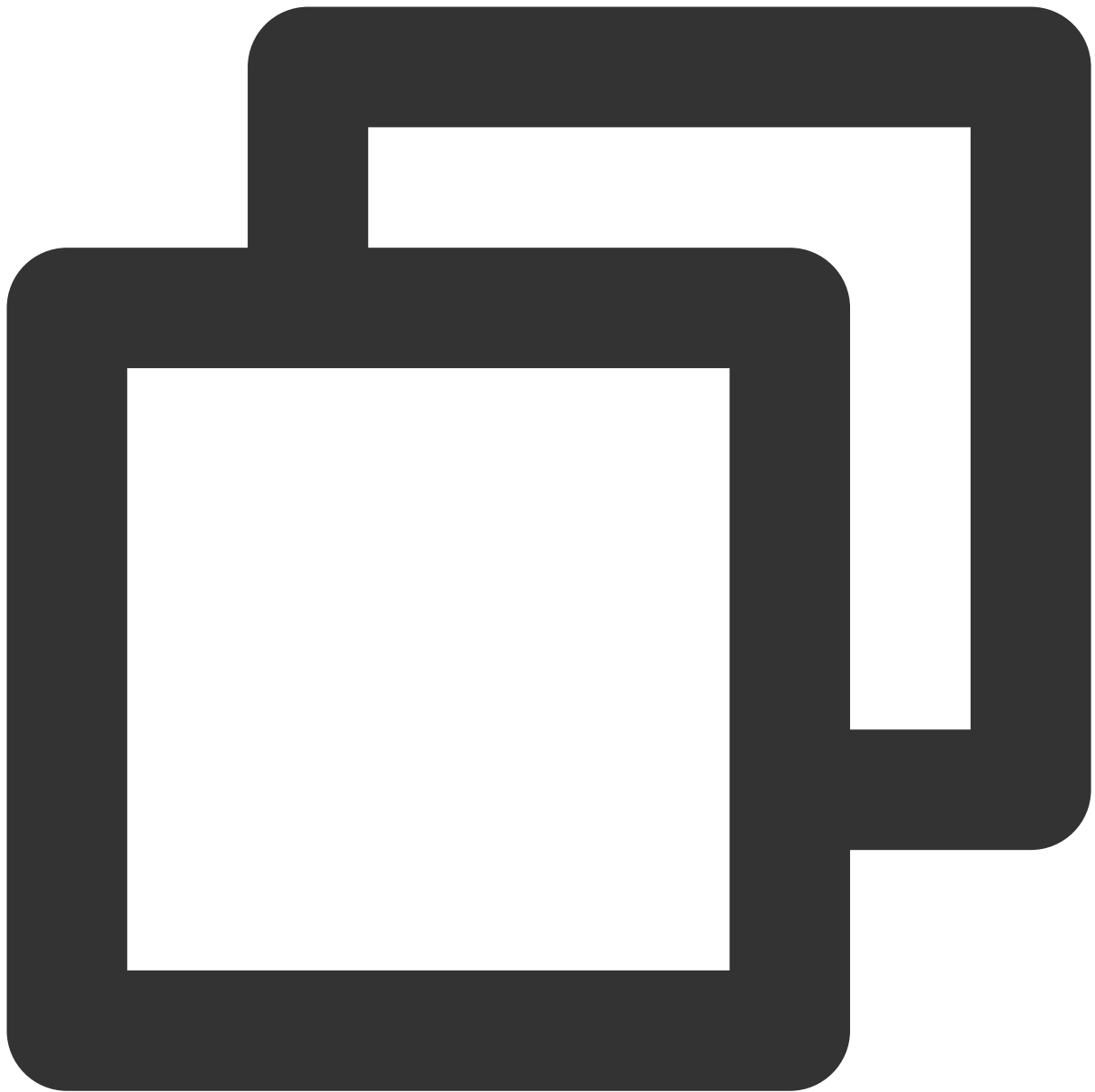


```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
```

```
        config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] += [  
            '$(inherited) ',  
            'PERMISSION_MICROPHONE=1 ',  
            'PERMISSION_CAMERA=1 ',  
        ]  
    end  
end  
end
```

## Step 4: Log in to TUIRoomKit Component

Add the following code to your project, which serves to log in to the component by calling the relevant APIs in TUIRoomKit. This step is extremely critical, as only after logging in can you use the various functions of TUIRoomKit, so please be patient and check if the relevant parameters are configured correctly:



```
import 'package:rtc_room_engine/rtc_room_engine.dart';

var result = await TUIRoomEngine.login(
  SDKAPPID,    // Please replace with your SDKAPPID
  'userId',    // Please replace with your user ID
  'userSig',   // Please replace with your userSig
);

if (result.code == TUIError.success) {
  // login success
} else {
```

```
// login error
}
```

## Parameter Description

Here is a detailed introduction to the key parameters used in the login function:

**SDKAppID** : Obtained in the last part of [Step 1](#).

**UserID** : The ID of the current user, which is a string that can only contain English letters (a-z and A-Z), numbers (0-9), hyphens (-), and 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 information such as SDKAppID and UserID. You can generate a temporary UserSig on the [UserSig Tools](#) page in the TRTC console.

For more information, please refer to the [UserSig](#).

### Note:

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

The `SDKAppID` is incorrect.

`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` cannot be directly substituted with the value of the `SecretKey`.

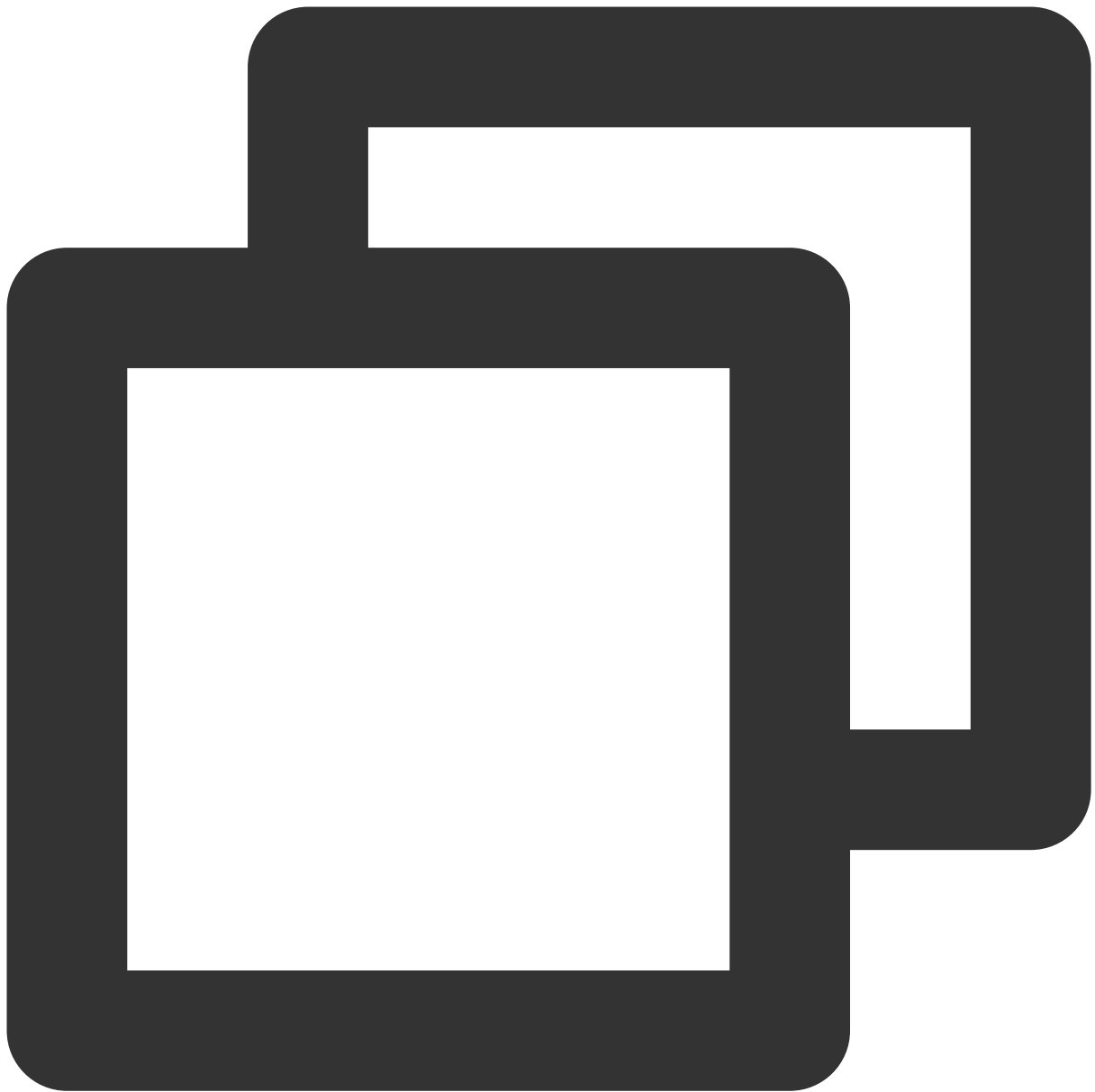
The `UserID` is set to a simple string such as 1, 123, or 111, and your colleague may be using the same UserID while working on a project simultaneously. **In this case, login will fail as TRTC doesn't support login on multiple terminals with the same `UserID`.** Therefore, we recommend you use some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

### Log in to Floating Chat (optional)

Flutter **TUIRoomKit** (**tencent\_conference\_uikit**) introduced the **floating chat feature** starting from version **2.4.1**. If you need to use the floating chat feature, you need to complete the following initialization and login (If you also need to use the [In-Conference Chat](#) page, you can **ignore** this step and complete the [initialization and login for In-Conference Chat](#)):



```
import 'package:tencent_cloud_chat_sdk/enum/V2TimSDKListener.dart';
import 'package:tencent_cloud_chat_sdk/enum/log_level_enum.dart';
import 'package:tencent_cloud_chat_sdk/models/v2_tim_callback.dart';
import 'package:tencent_cloud_chat_sdk/tencent_im_sdk_plugin.dart';

// Initialize
var initResult = await TencentImSDKPlugin.v2TIMManager.initSDK(
  sdkAppID: SDKAPPID, // Replace with your SDKAPPID
  loglevel: LogLevelEnum.V2TIM_LOG_INFO, // Log registration level
  listener: V2TimSDKListener(), // Event listener. When using the float
);
```

```
if (initResult.code == 0) { // Initialized successfully
    // Login
    V2TimCallback imLoginResult = await TencentImSDKPlugin.v2TIMManager.login(
        userID: 'userId',    // Replace with your userID
        userSig: 'userSig', // Replace with your userSig
    );
}
```

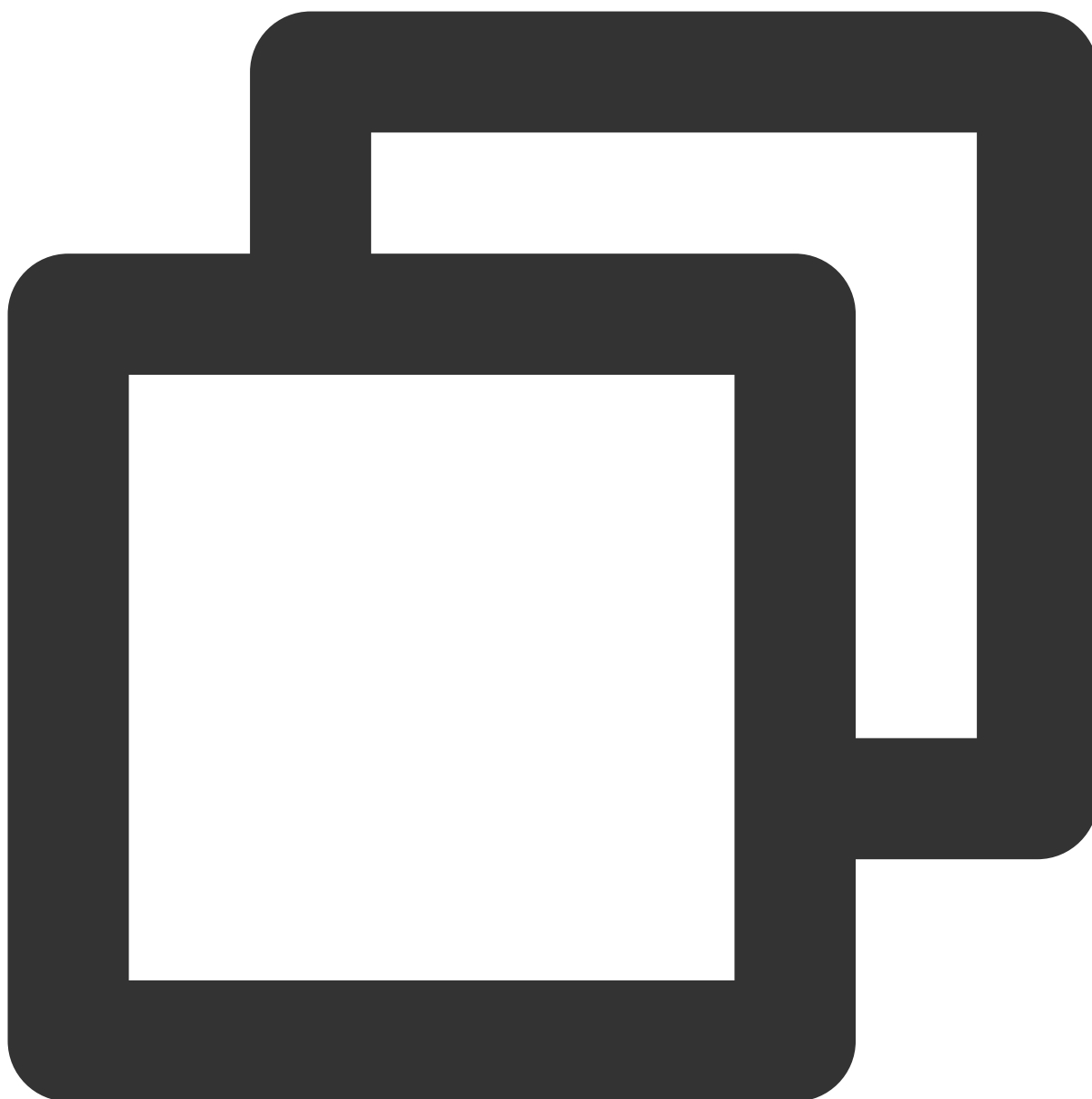
**Note:**

The floating chat feature is enabled by default in TUIRoomKit. If you do not need the floating chat feature, you do not need to perform the above initialization and login steps. You can disable the floating chat feature through the **Bottom toolbar > Setting > Open Floating Chat** option.

## Step 5: Use TUIRoomKit

### Set User Information (optional)

You can set the username and profile photo by calling TUIRoomEngine's `setSelfInfo` .



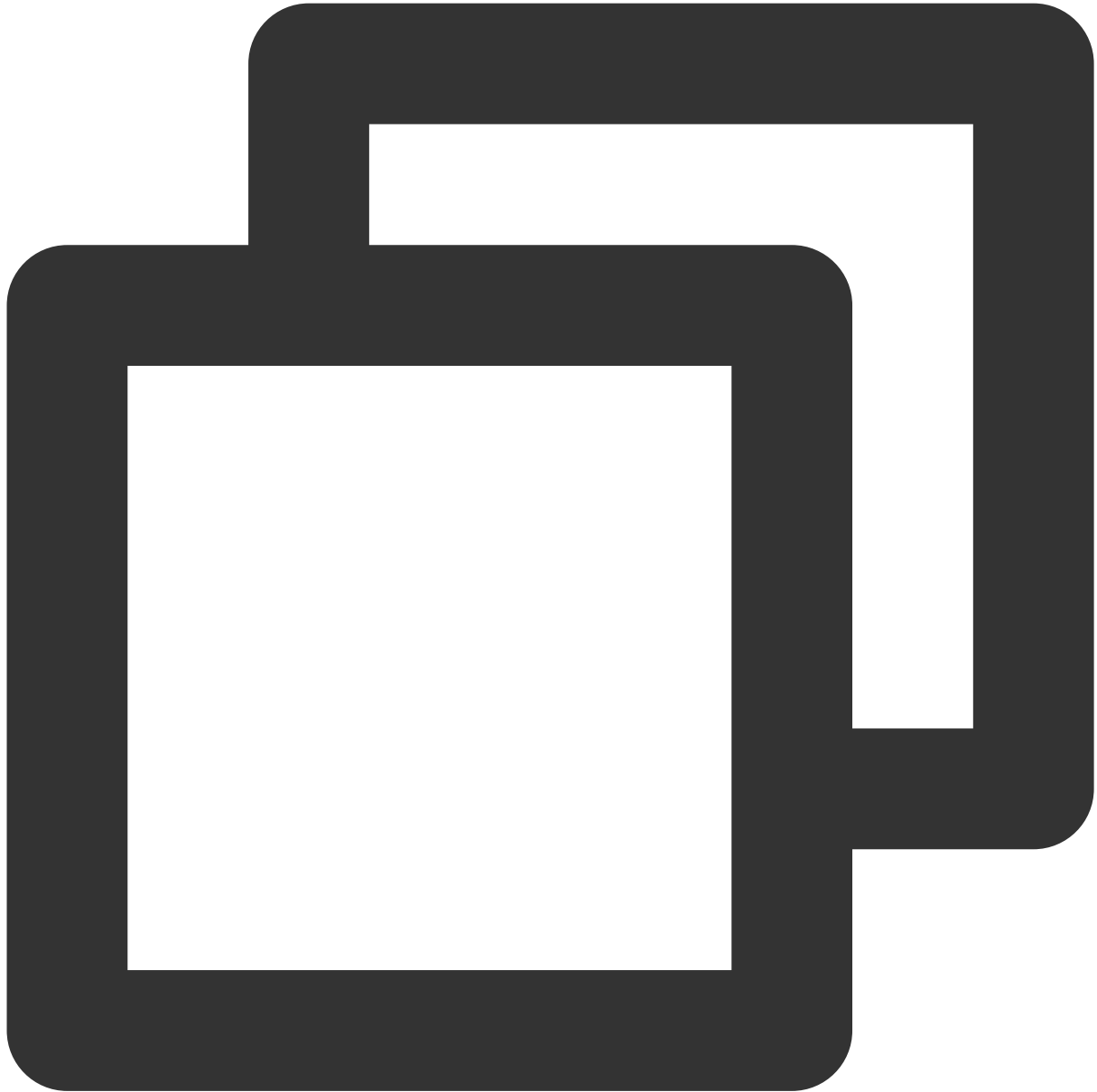
```
import 'package:rtc_room_engine/rtc_room_engine.dart';
```

```
TUIRoomEngine.setSelfInfo(userName, avatarURL);
```

Parameter	Type	Description
userName	String	User name
avatarURL	String	User profile photo URL

## Start a quick conference

By calling the `quickStart` method of `ConferenceSession`, you can start a quick conference.



```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

ConferenceSession.newInstance('your roomId')
  ..onActionSuccess = _quickStartSuccess
  ..onActionError = _quickStartError
  ..quickStart();

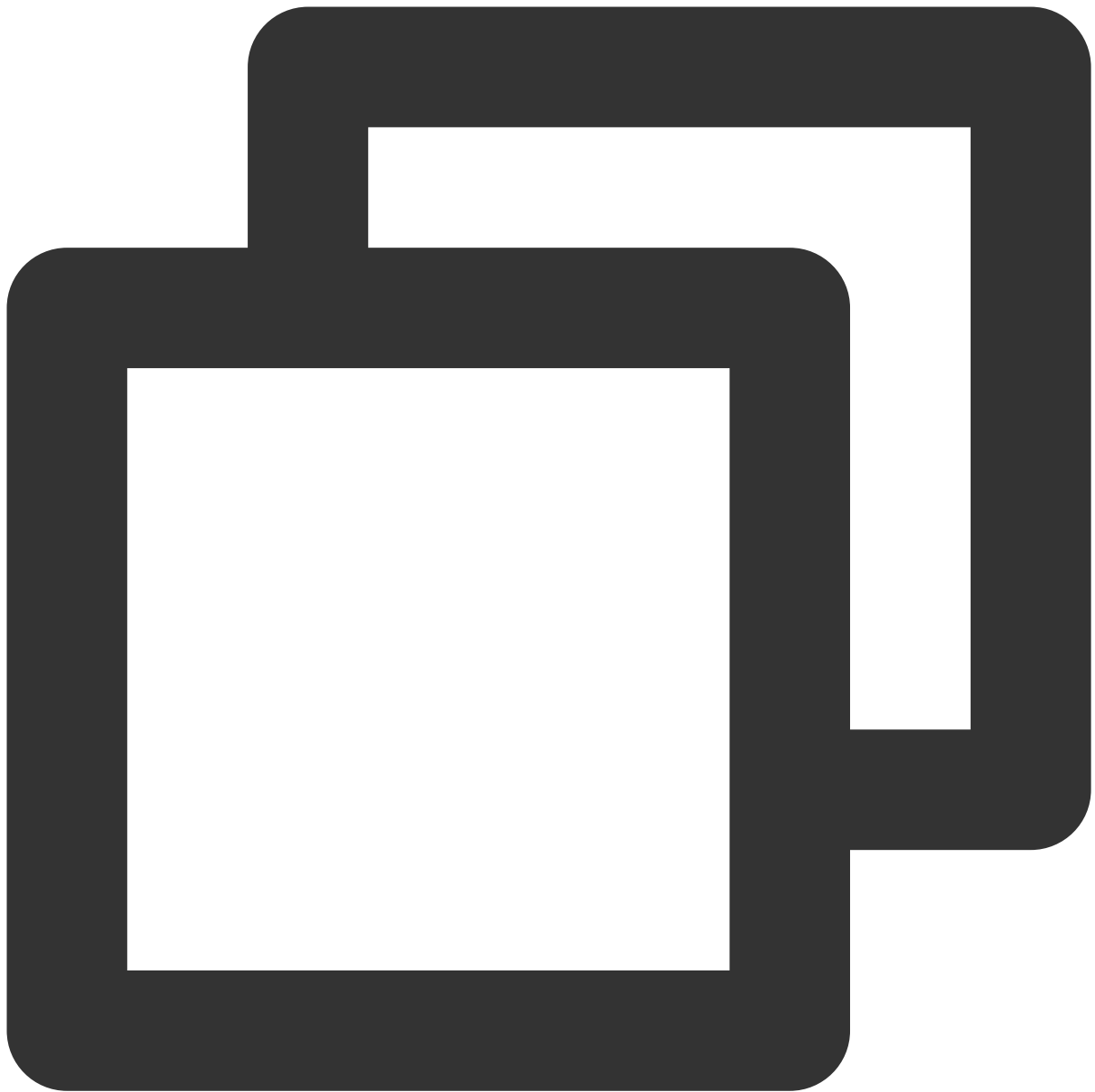
void _quickStartSuccess() {
```

```
// You can navigate to the conference page on your own in this success callback o
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ConferenceMainPage(),
  ),
);
}

void _quickStartError(ConferenceError error, String message) {
  debugPrint("code: $error message: $message");
}
```

## Join a conference

By calling the `join` method of `ConferenceSession`, you can join the specified conference.



```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

ConferenceSession.newInstance('your roomId')
  ..onActionSuccess = _joinSuccess
  ..onActionError = _joinError
  ..join();

void _joinSuccess() {
  //You can navigate to the conference page on your own in this success callback of
  Navigator.push(
    context,
```

```
MaterialPageRoute(  
    builder: (context) => ConferenceMainPage(),  
),  
);  
}  
  
void _joinError(ConferenceError error, String message) {  
    debugPrint("code: $error message: $message");  
}
```

**Note:**

In the above sample code, only the simplest way to create/join a conference is shown. If you need to complete more settings before entering the conference, please refer to [Pre-conference Control](#).

## More Features

TUIRoomEngine SDK provides more rich features for audio and video rooms. For more information, see, [Specific Content Please Refer API Overview](#).

## Suggestions and Feedback

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



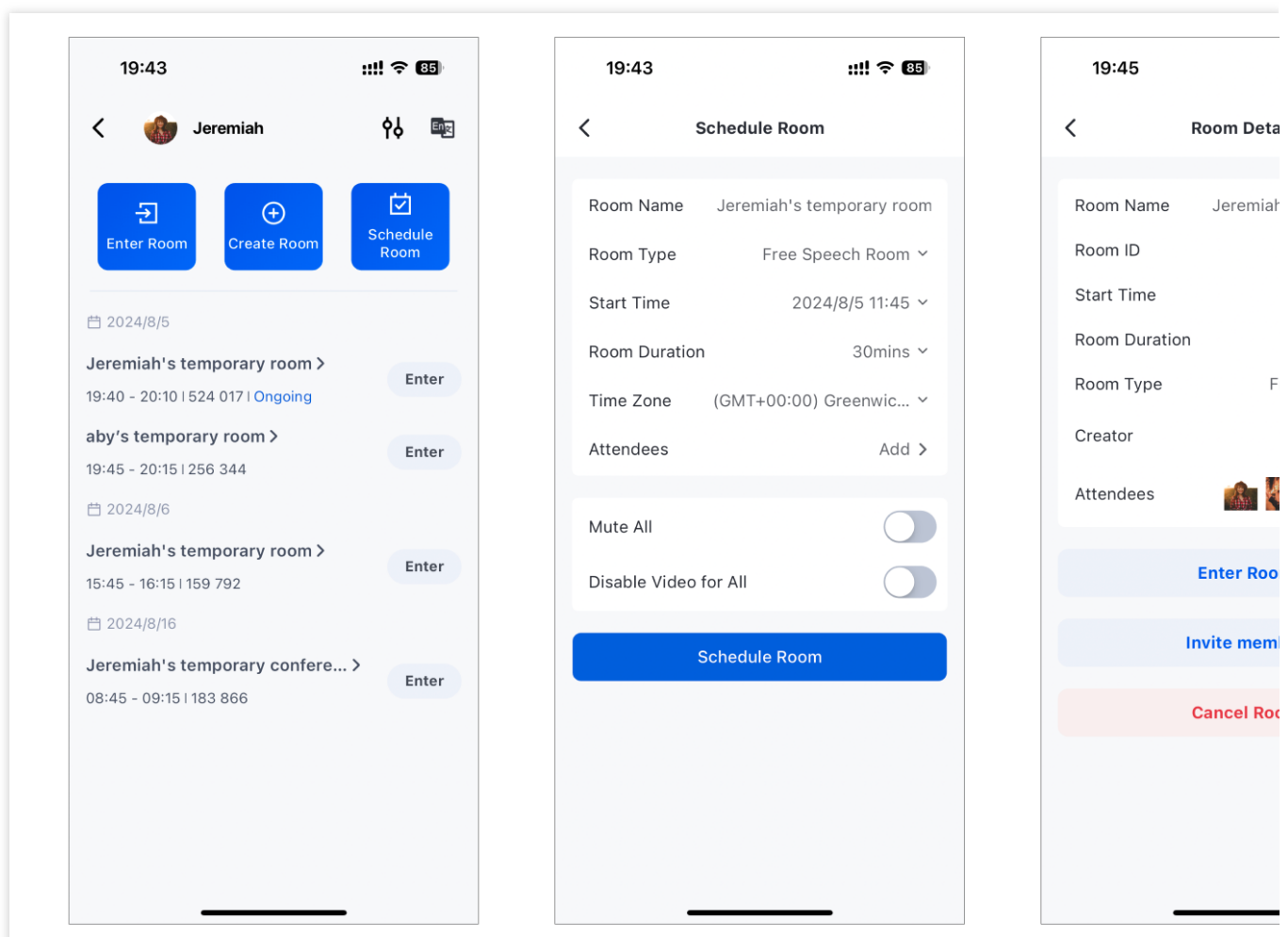
# Schedule a meeting (TUIRoomKit)

## Android&iOS&Flutter

Last updated : 2024-08-09 15:08:11

## Description of the Feature

Conference(TUIRoomKit) now supports a new feature for conference reservations. Users can reserve a room and schedule conferences. When the conference time arrives, users can start the conference with one click. This article will detail the related features and explain how to use this functionality in the TUIRoomKit component.



# Preparation Requirements

Before using the scheduled conference feature provided by Conference(TUIRoomKit), you need to complete the related configuration for integrating Conference and **log in**. For more details, refer to [Integration](#).

**Note:**

**The scheduled conference feature requires Conference v2.5.0 and above.**

# How to Schedule a Conference

To utilize the scheduled conference feature, you need to access the conference scheduling page provided by TUIRoomKit:

Android

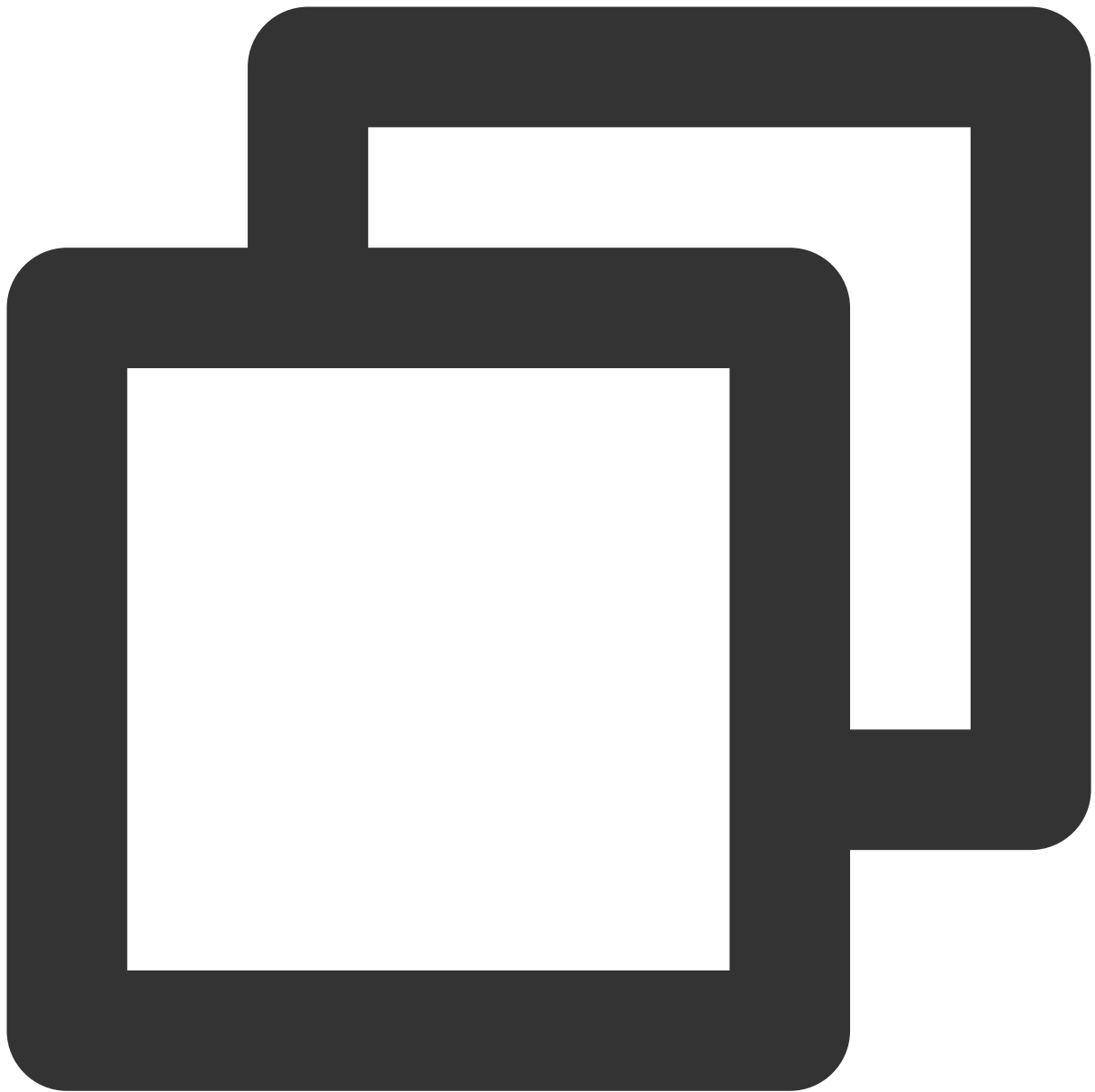
iOS

Flutter

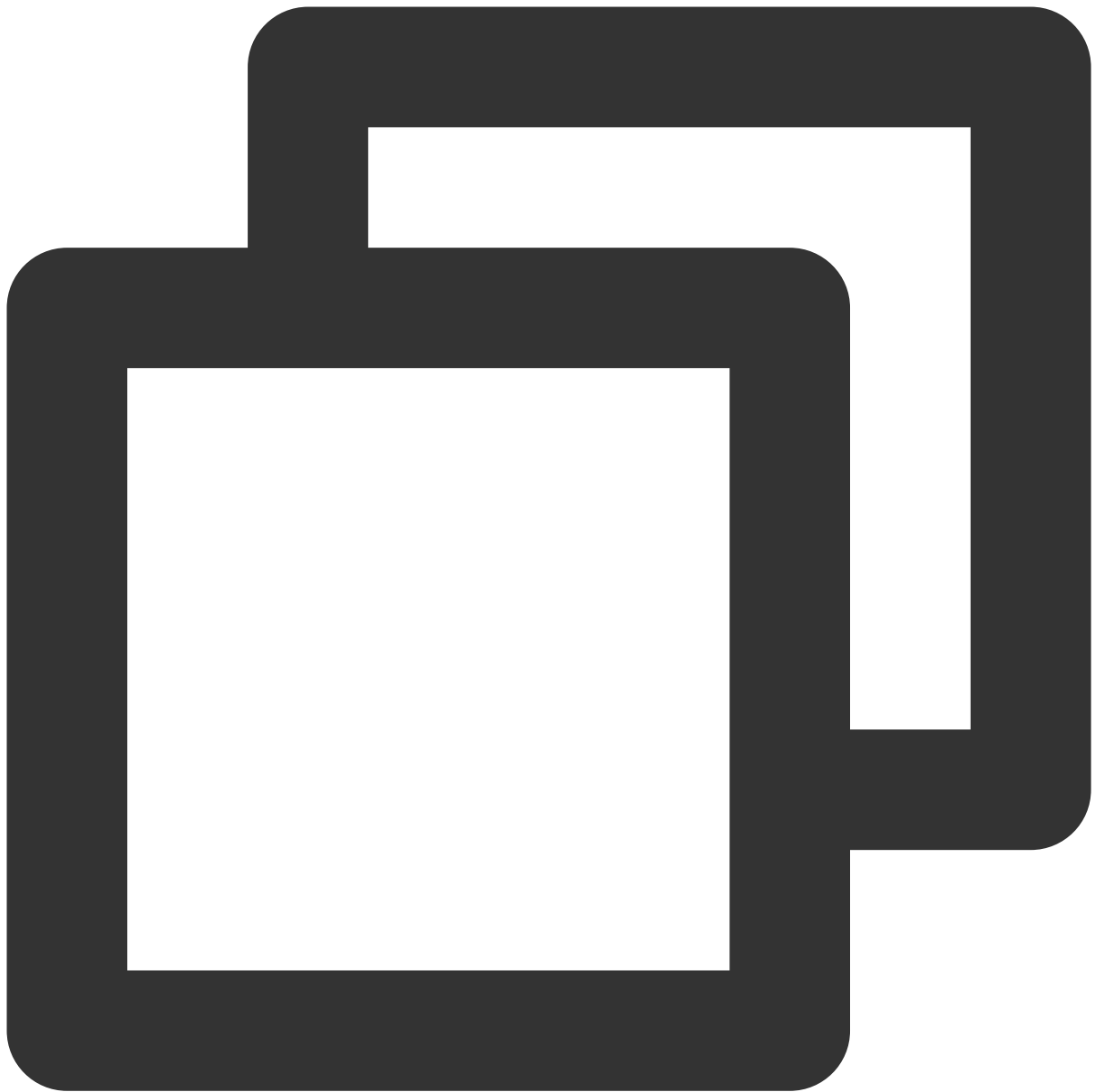
According to your business, simply call the following code in the corresponding Activity to schedule a conference.

java

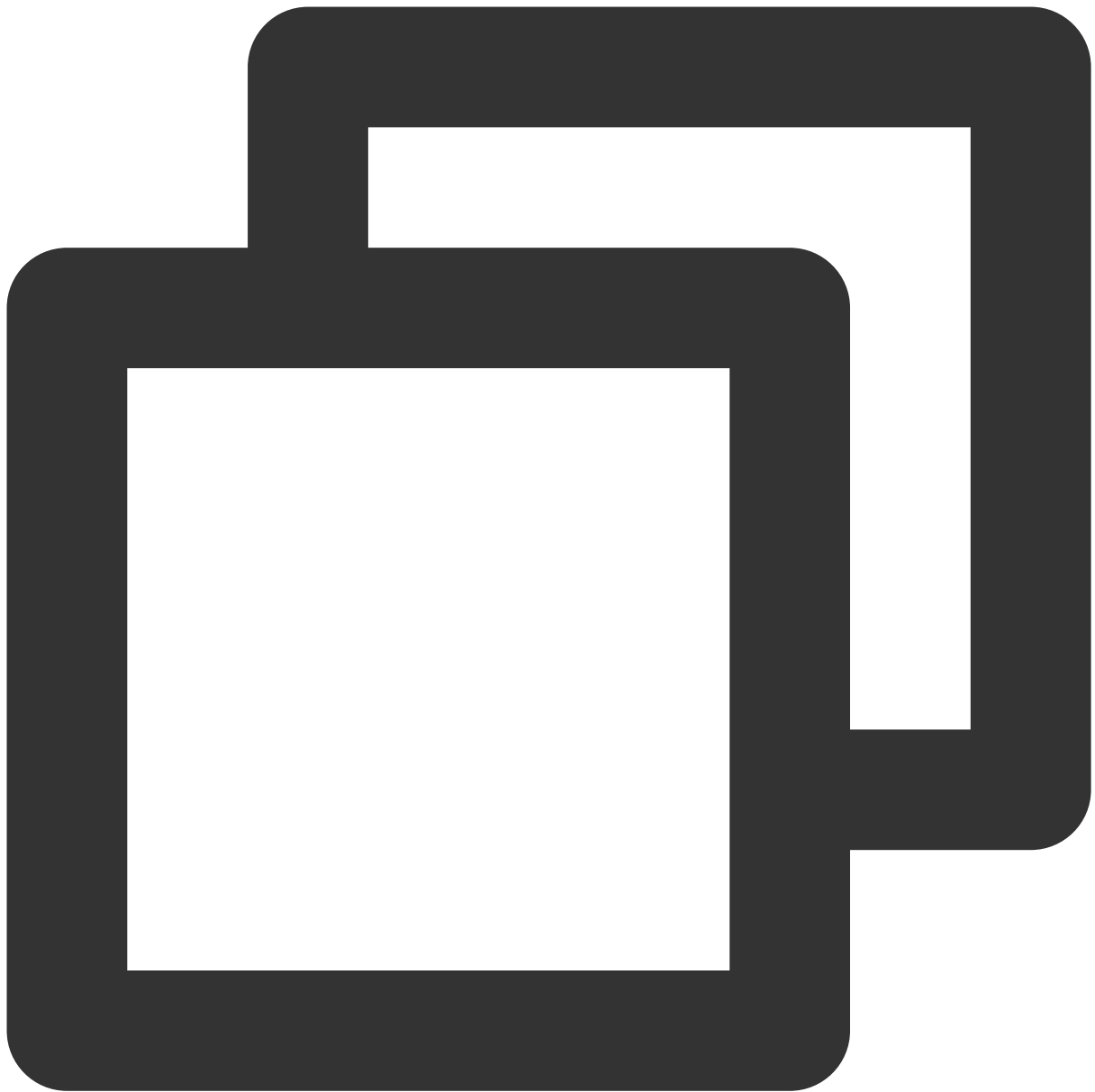
kotlin



```
Intent intent = new Intent(this, ScheduleConferenceActivity.class);
startActivity(intent);
```

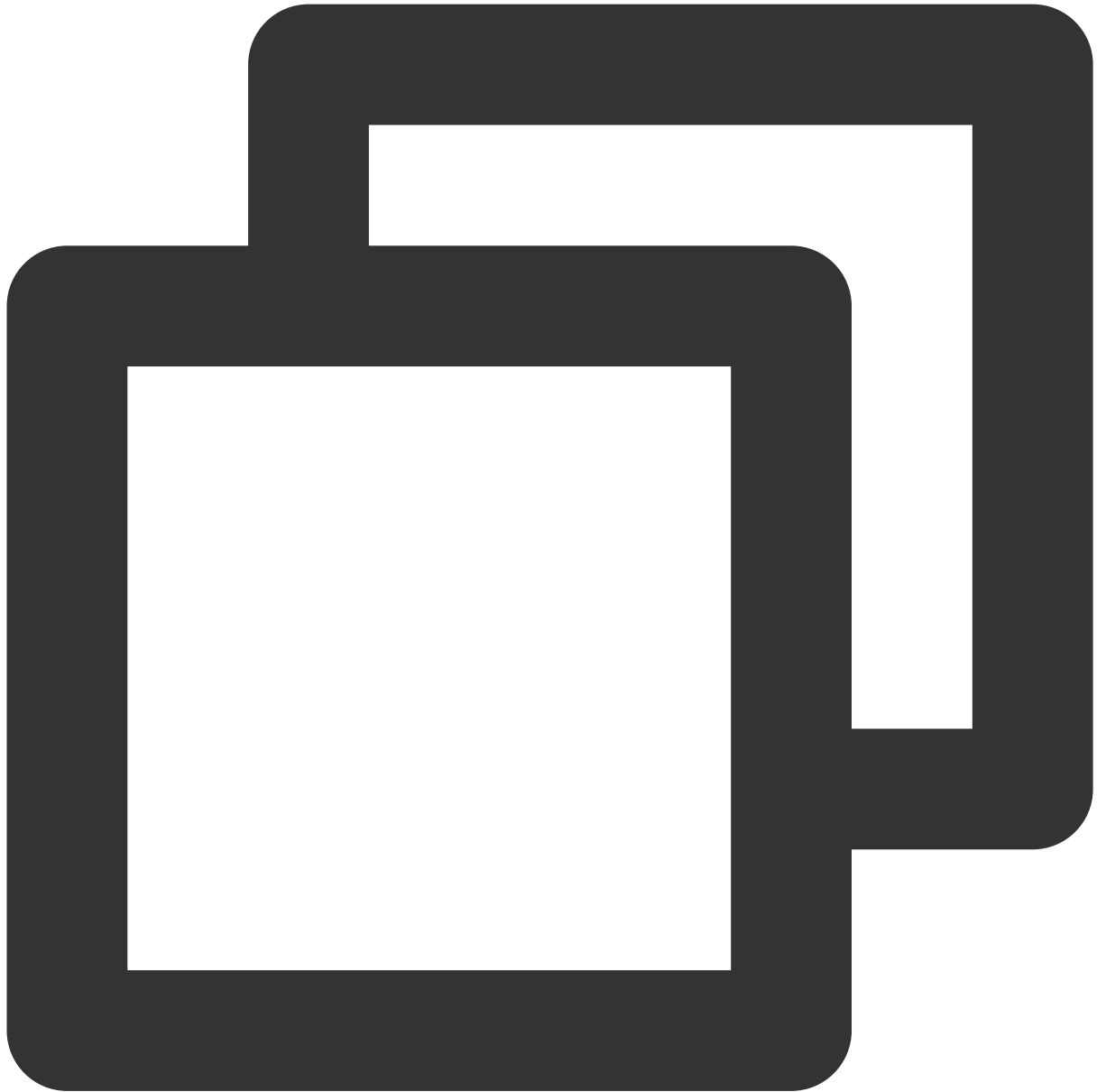


```
val intent = Intent(this, ScheduleConferenceActivity::class.java)
startActivity(intent)
```



```
class YourViewController: UIViewController {  
    // The memberSelectFactory parameter is used to create your User Selection Page.  
    // For detailed usage, please refer to "Invite to Enter Room".  
    func jumpToScheduleViewController {  
        let scheduleViewController =  
            ScheduleConferenceViewController(memberSelectFactory: ni  
            navigationController?.pushViewController(scheduleViewController, animated:  
    }  
}
```

According to your business, navigate to `ScheduleRoomPage` when needed to access the conference scheduling page.



```
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart'

Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ScheduleRoomPage(),
  ),
);
```

---

On the conference scheduling page, after filling in and selecting the Relevant Information for the scheduled conference, click the **Schedule Room** button to complete the schedule.

## View scheduled conferences

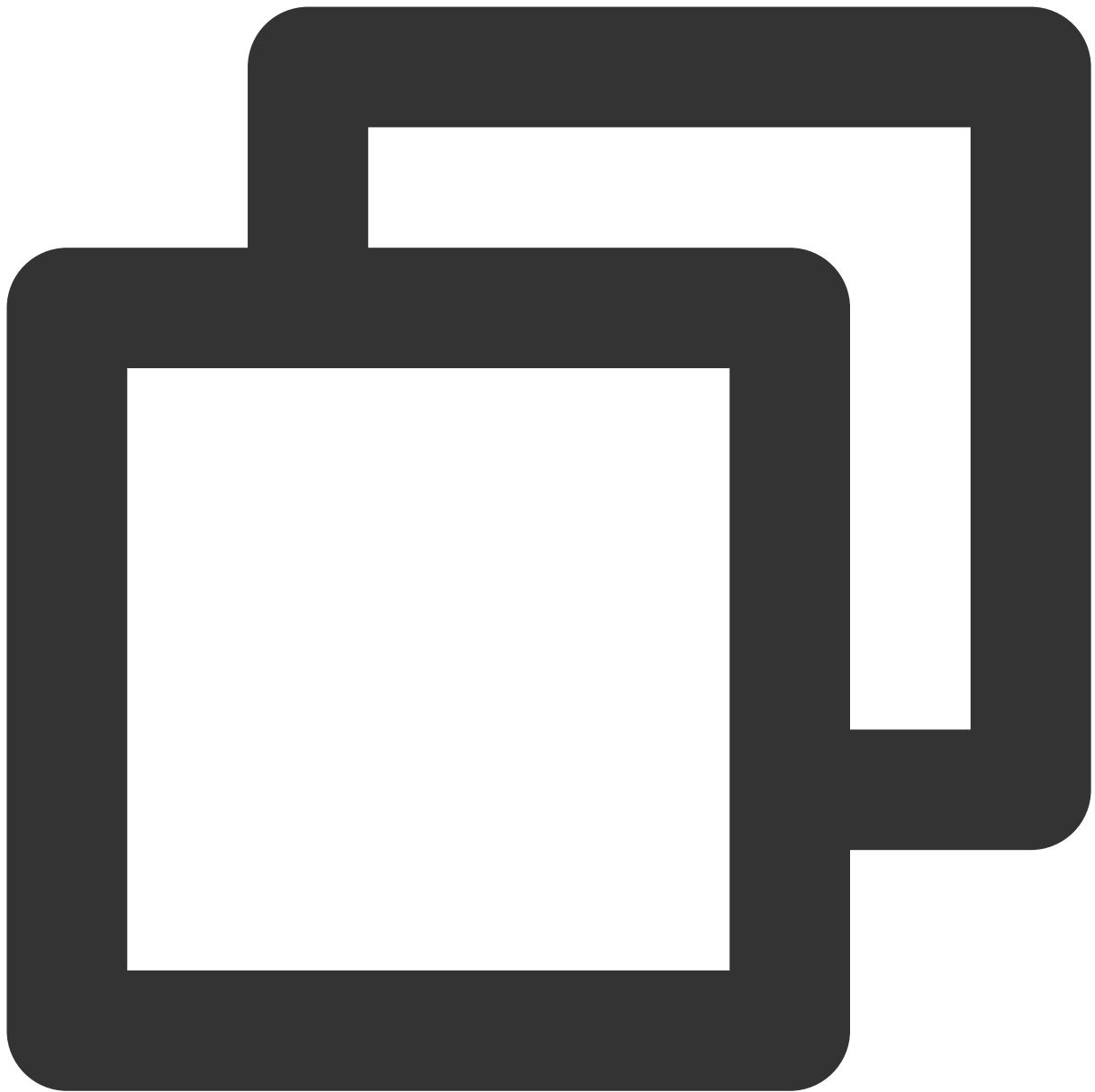
TUIRoomKit provides UI components for the conference list. By arranging the conference list component on your page, you can view and manage all conferences by the current user:

Android

iOS

Flutter

1. Based on your business, add the meeting list layout to your layout:



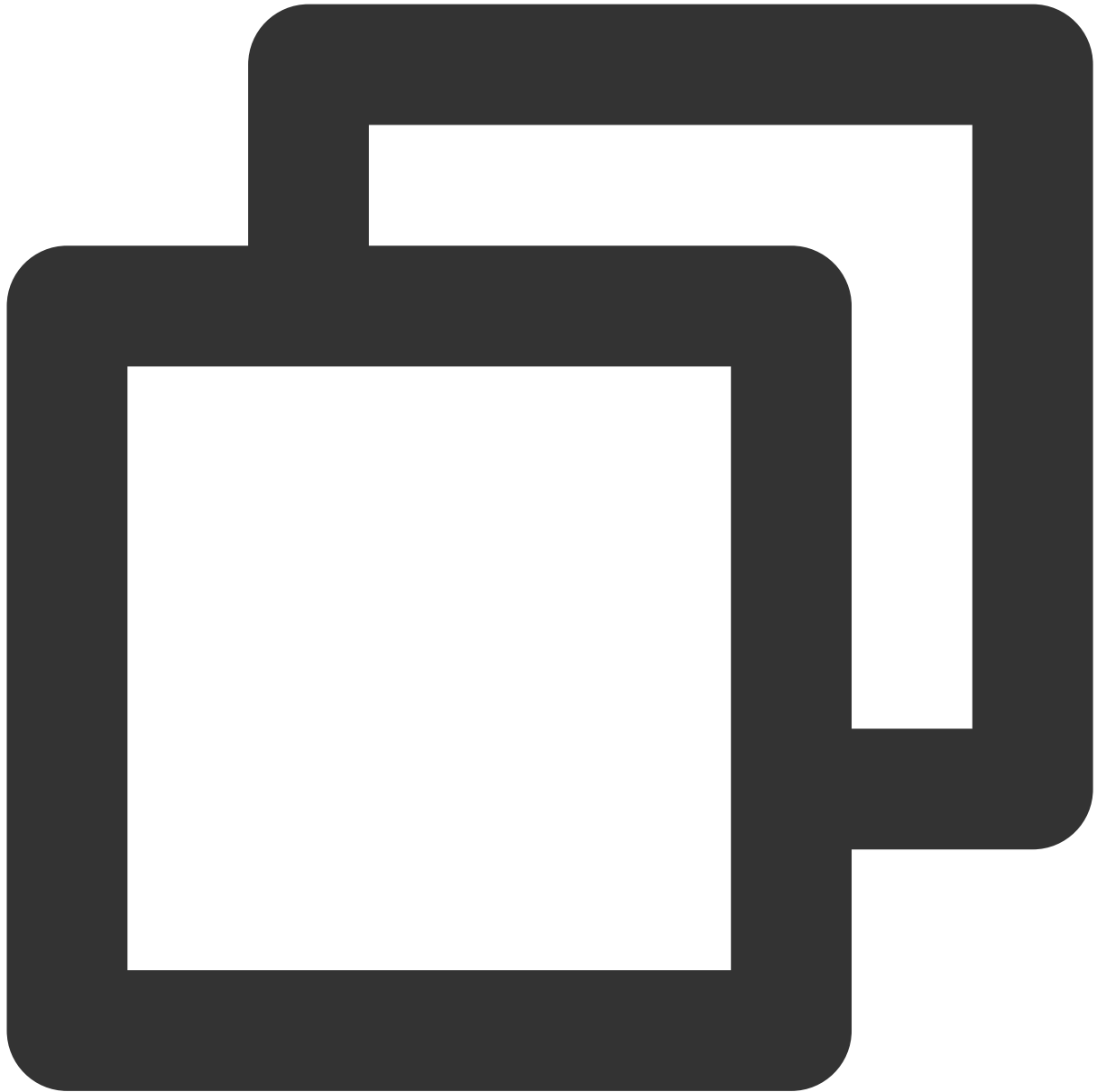
```
<!-- For example, if your parent layout is ConstraintLayout, add the following code
<FrameLayout
    android:id="@+id/tuiroomkit_fl_conference_list_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

2. Add the following code in the corresponding Activity to pull up the meeting list:

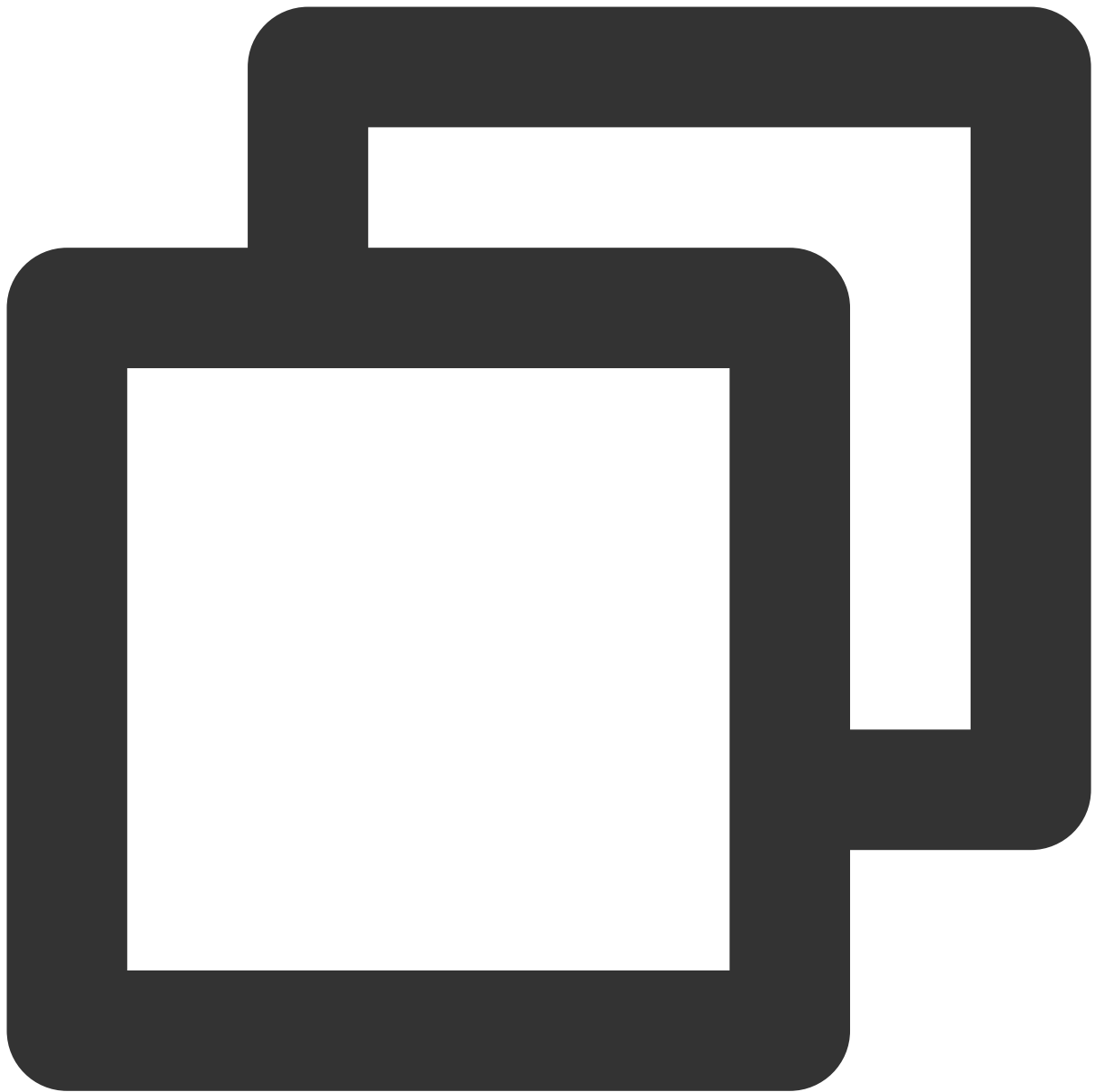


java

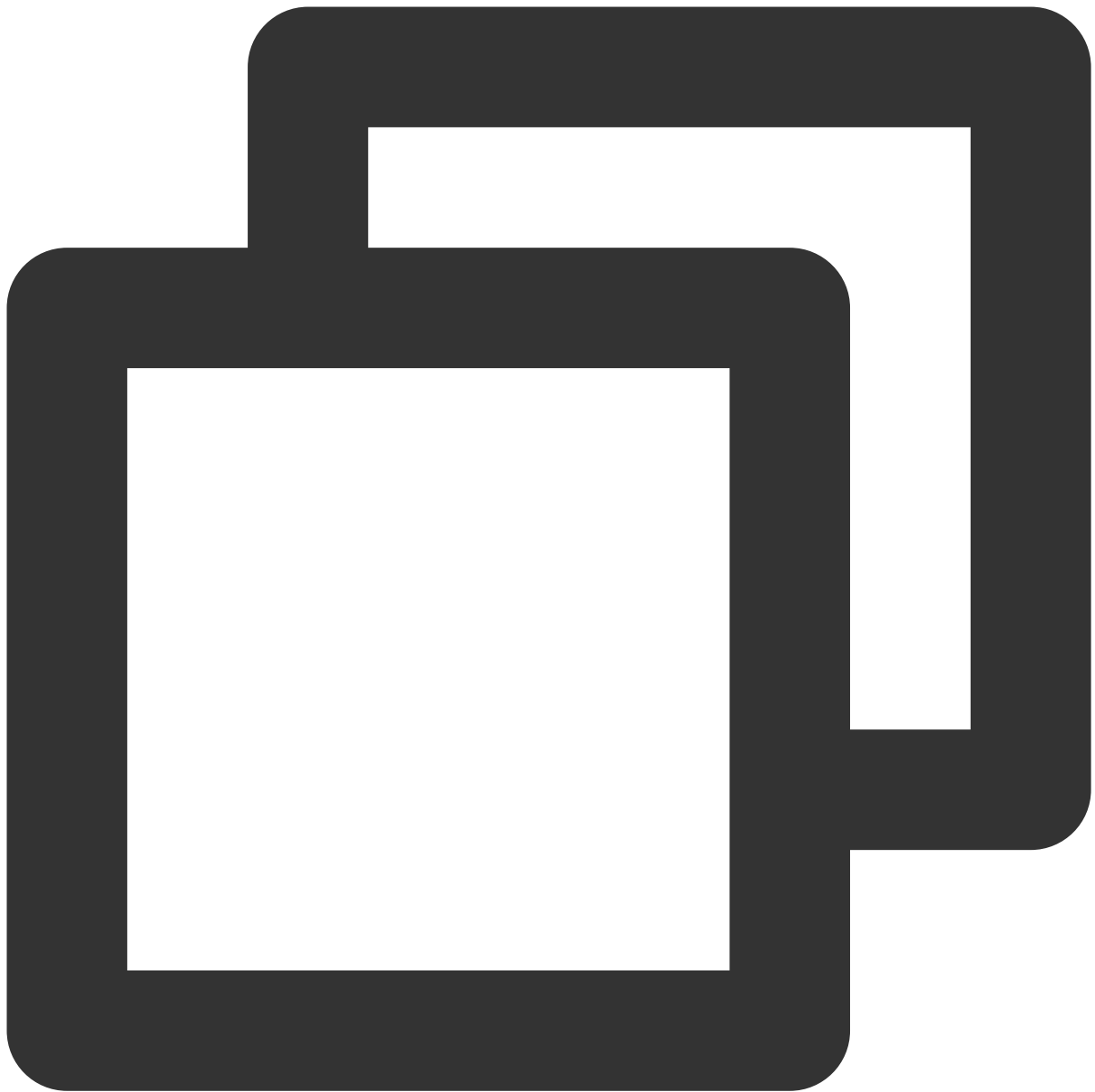
kotlin



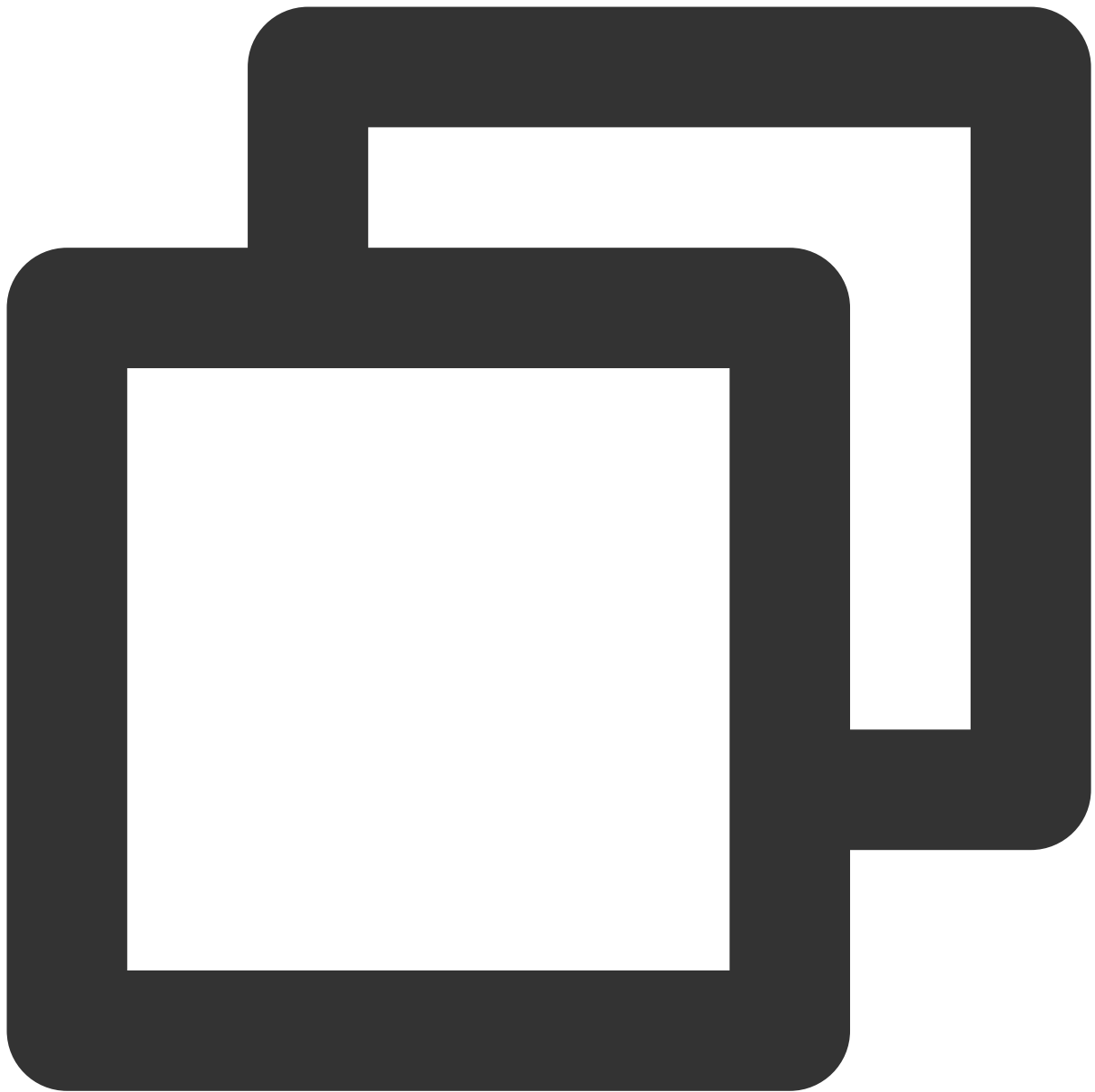
```
ConferenceListFragment fragment = new ConferenceListFragment();  
FragmentTransaction transaction = this.getSupportFragmentManager().beginTransaction  
transaction.add(R.id.tuiroomkit_fl_conference_list_container, fragment);  
transaction.commitAllowingStateLoss();
```



```
val fragment = ConferenceListFragment()
val transaction = supportFragmentManager.beginTransaction()
transaction.add(R.id.tuiroomkit_fl_conference_list_container, fragment)
transaction.commitAllowingStateLoss()
```



```
class YourViewController: UIView {  
    // ConferenceListView initialization requires two parameters:  
    // @param viewController, the viewController to which the meeting list page  
    // @param memberSelectFactory, creates your User Selection Page. For detail  
    func showConferenceList {  
        let listView = ConferenceListView(viewController: self, memberSelectFa  
        view.addSubview(listView)  
    }  
}
```



```
import 'package:flutter/material.dart';
import 'package:tencent_conference_uikit/tencent_conference_uikit.dart';

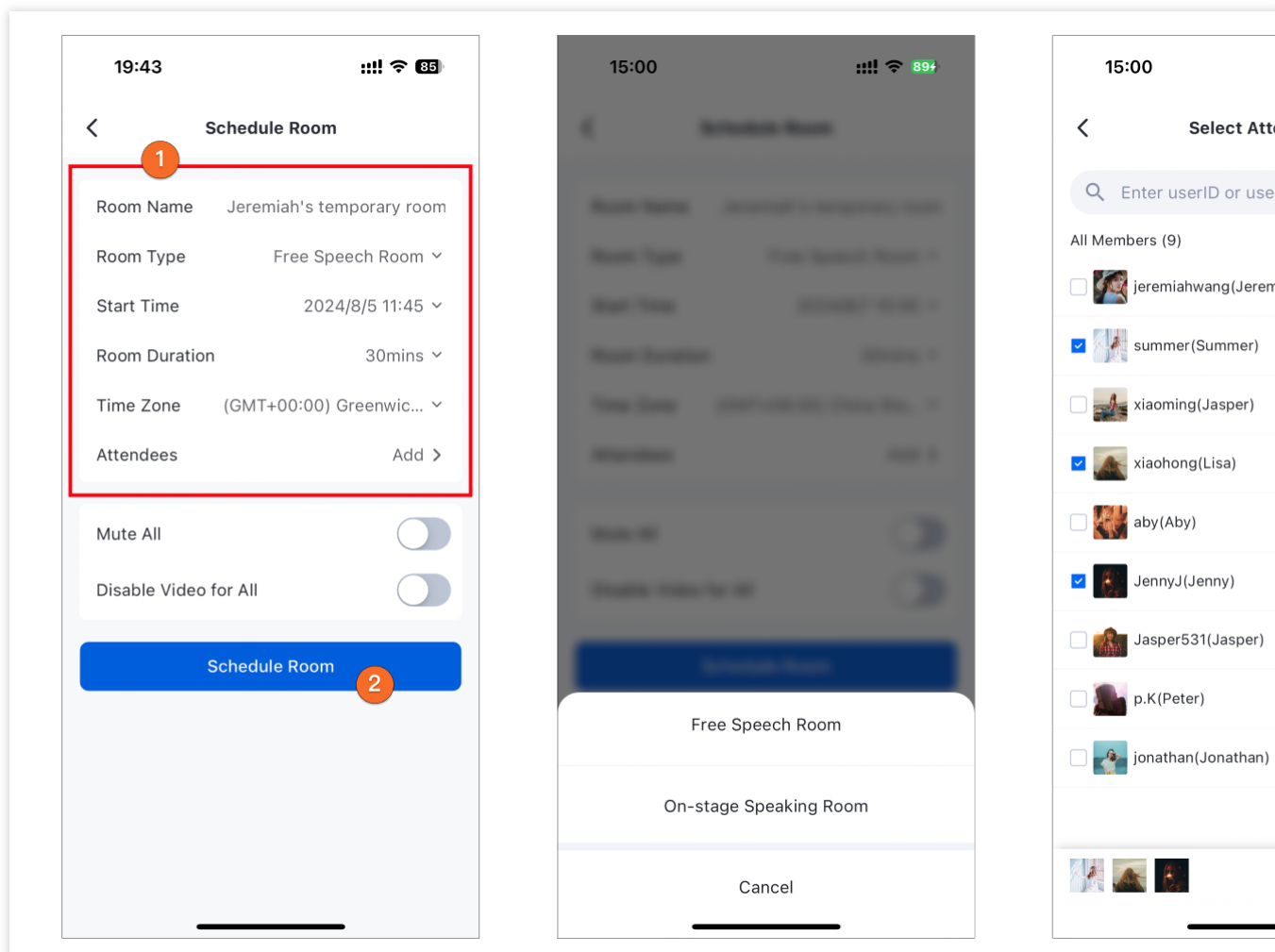
// In your own page, add ConferenceListWidget
class YourPage extends StatelessWidget {
  const YourPage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Your Page")),
      body: Column(
```

```
        children: [
          ...YourWidgets(),          // Your other widgets, this is just an example
          const ConferenceListWidget(), // Conference list component
        ],
      ),
    );
  }
}
```

# Conference Scheduling Control

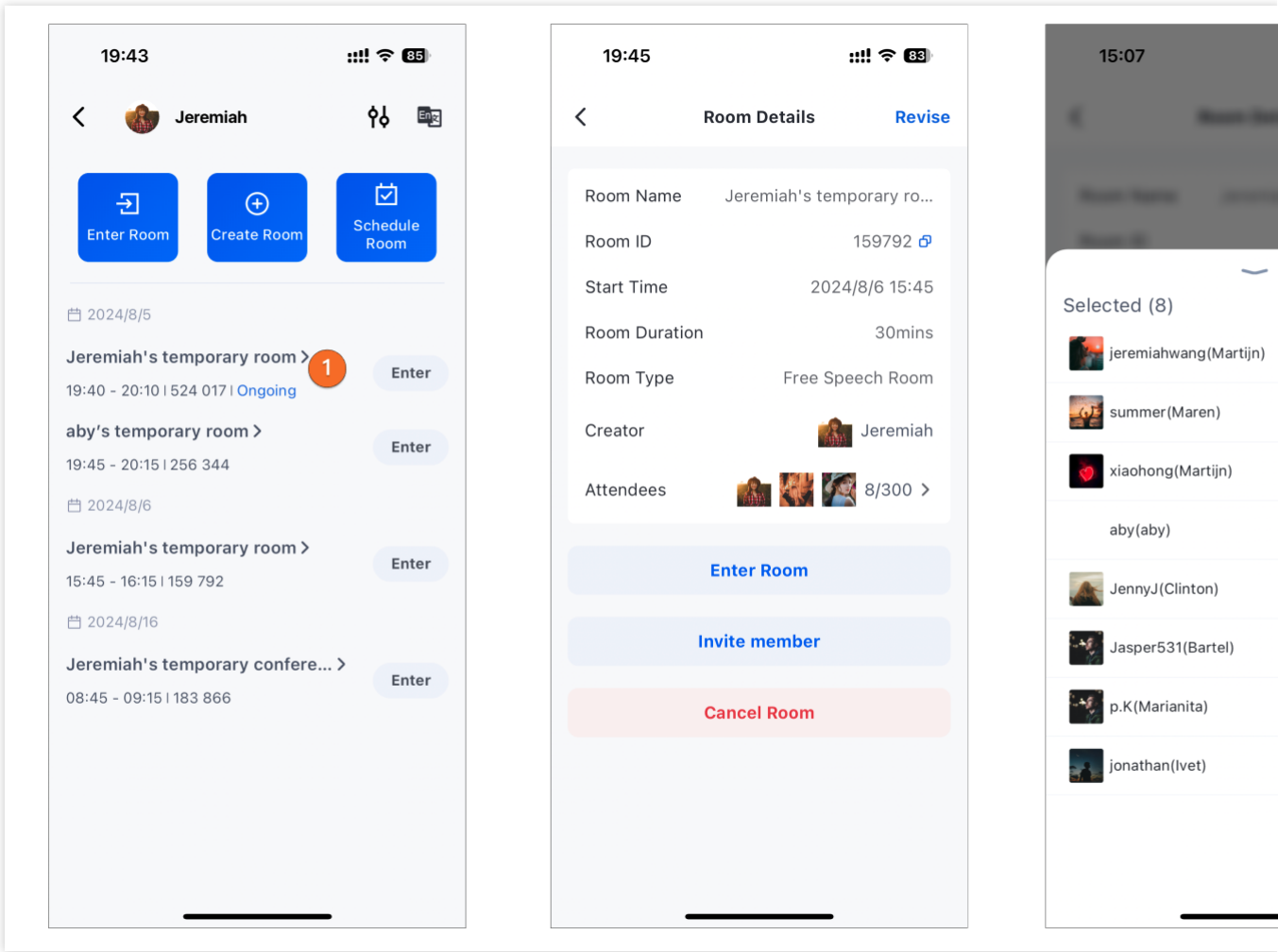
## Schedule Conference

On the conference scheduling page, users can set conference information according to their needs. The configurable options include: **Room Name**, **Room Type**, **Start Time**, **Room Duration**, **Timezone**, **Participating Members**, **Member Management (Mute All, Mute All Video)**, etc. For detailed introduction of the feature of inviting members to participate, refer to [Invite to Join Conference](#).



## View details

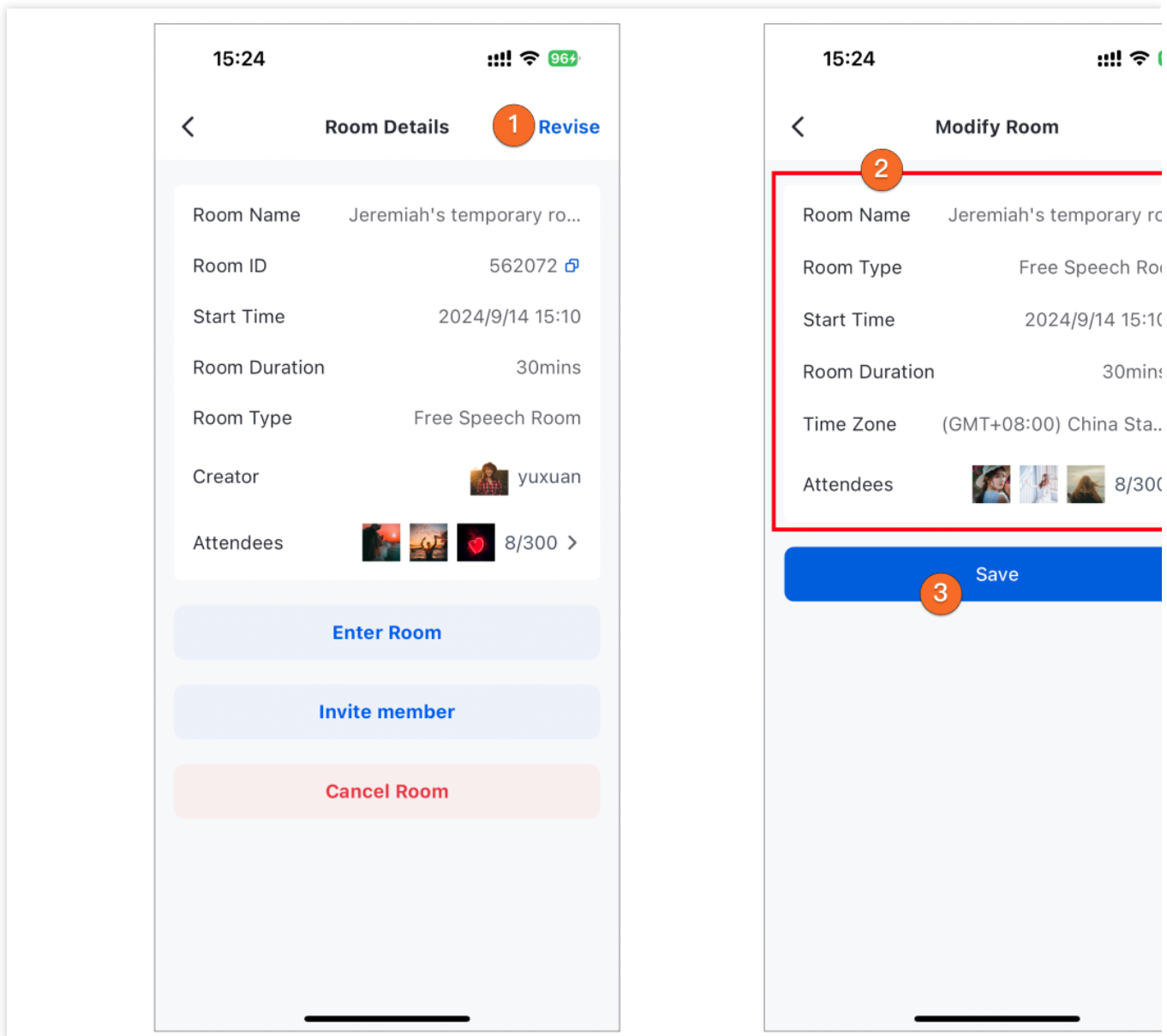
In the Conference List, users can click on a conference to view the detailed information of the corresponding scheduled conference.



## Edit Conference Information

The conference owner can modify the information of the scheduled conference. After completing the modifications, click **Save** to update the scheduled conference information with the new details.

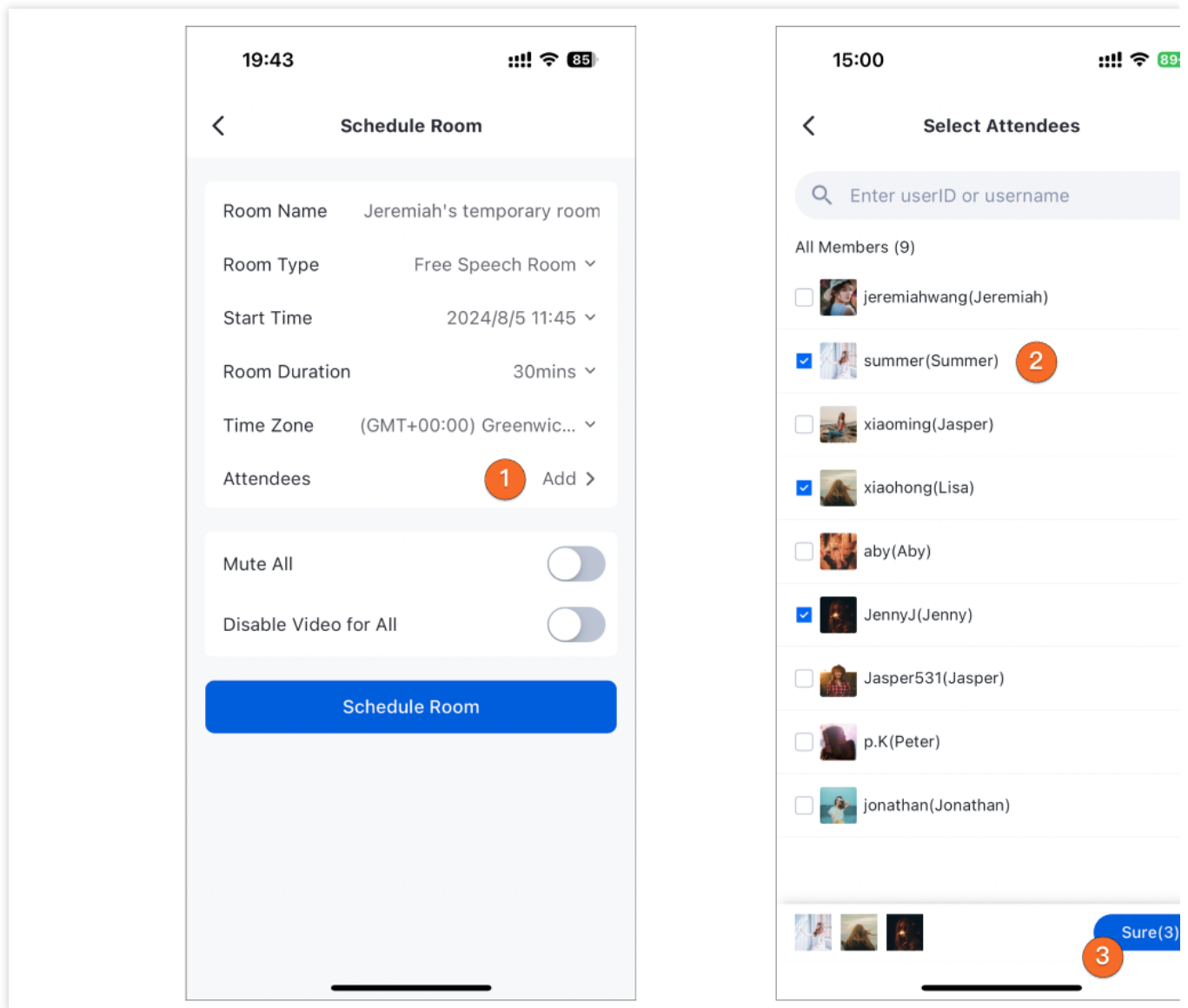
For a detailed introduction of the feature to invite participating members, refer to [Invite to Join Conference](#).



## Invite to Join Conference

Users can pop up the participant selection screen by clicking the **Participating Members** button on the scheduling or modifying conference interface. Here, you can invite or remove members. After inviting members and completing the scheduling or modification of the conference, the conference you reserved will **appear in the participants' conference list**.





By default, the Invite Member User List is empty. If you need to use this feature, you must import the members to be invited according to your business needs through the following methods:

Android

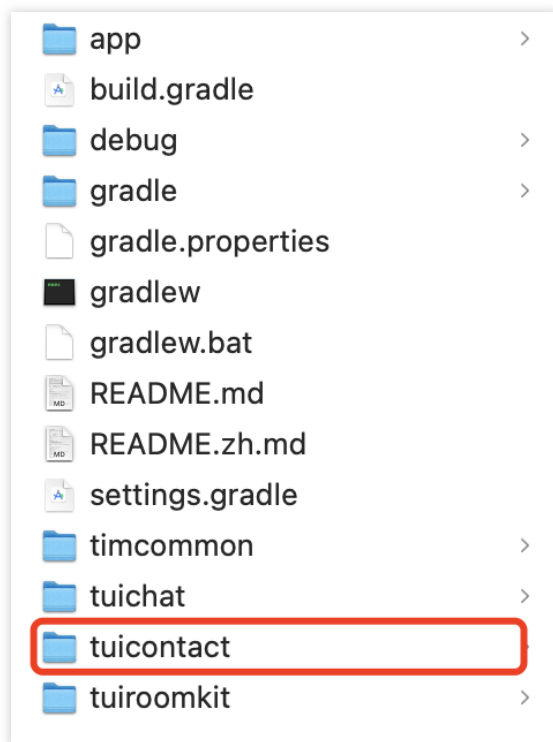
iOS

Flutter

The list of members invited by the Android TUIRoomKit comes from the **Chat friends relationship**, it is necessary to use **Chat to add friends**, and integrate the tuicontact members list component.

#### 1. Complete the following steps for quick integration with TUIContact :

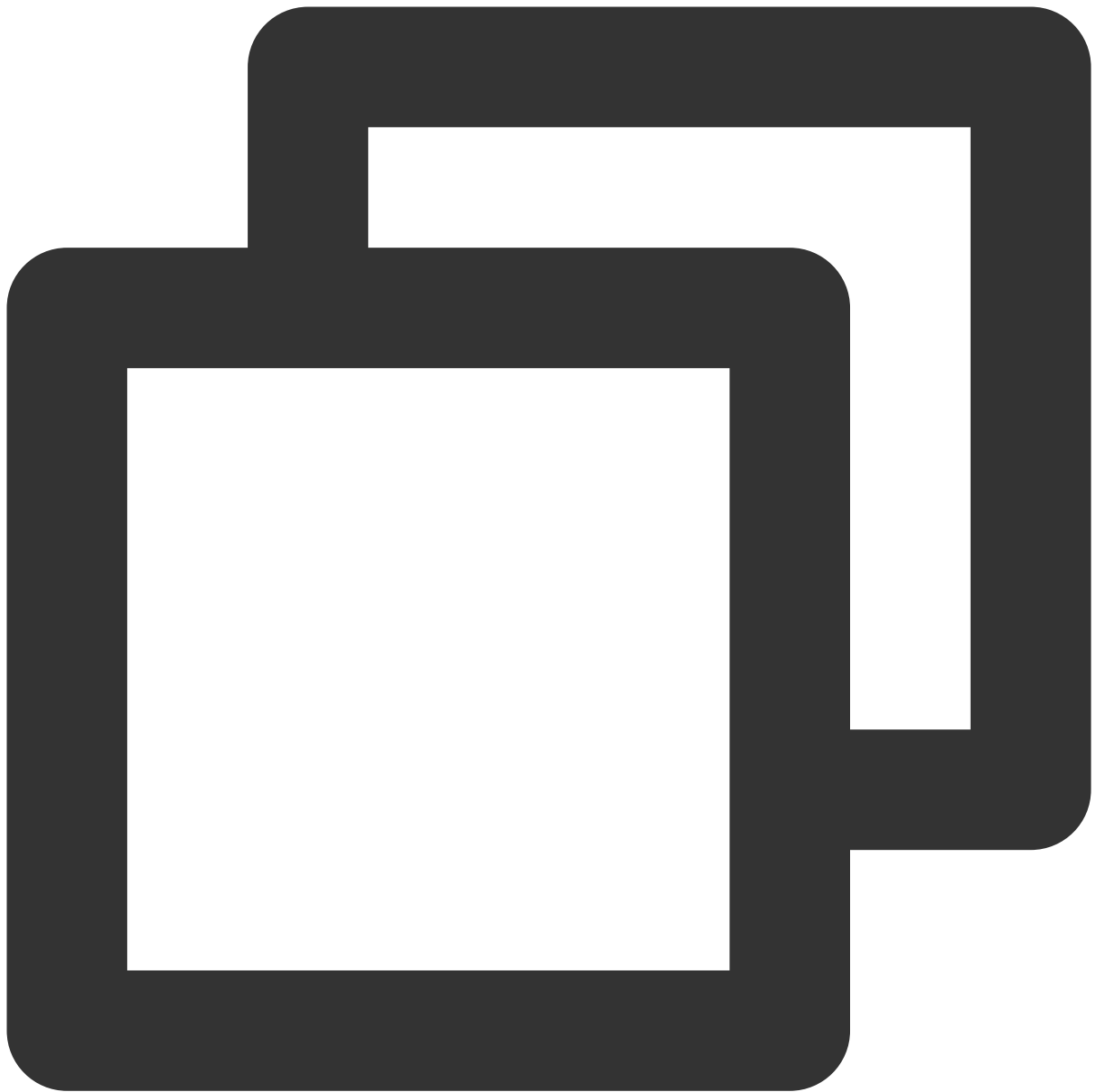
Clone or download the code from GitHub, then copy the tuicontact subdirectory from the Android directory to the same level as your current project's app directory, as shown below:



Find the `setting.gradle` (or `settings.gradle.kts`) file in the engineering root directory, and add the following code to import the `tuicontact` component into your current project.

`settings.gradle`

`setting.gradle.kts`



```
include ':tuicontact'
```

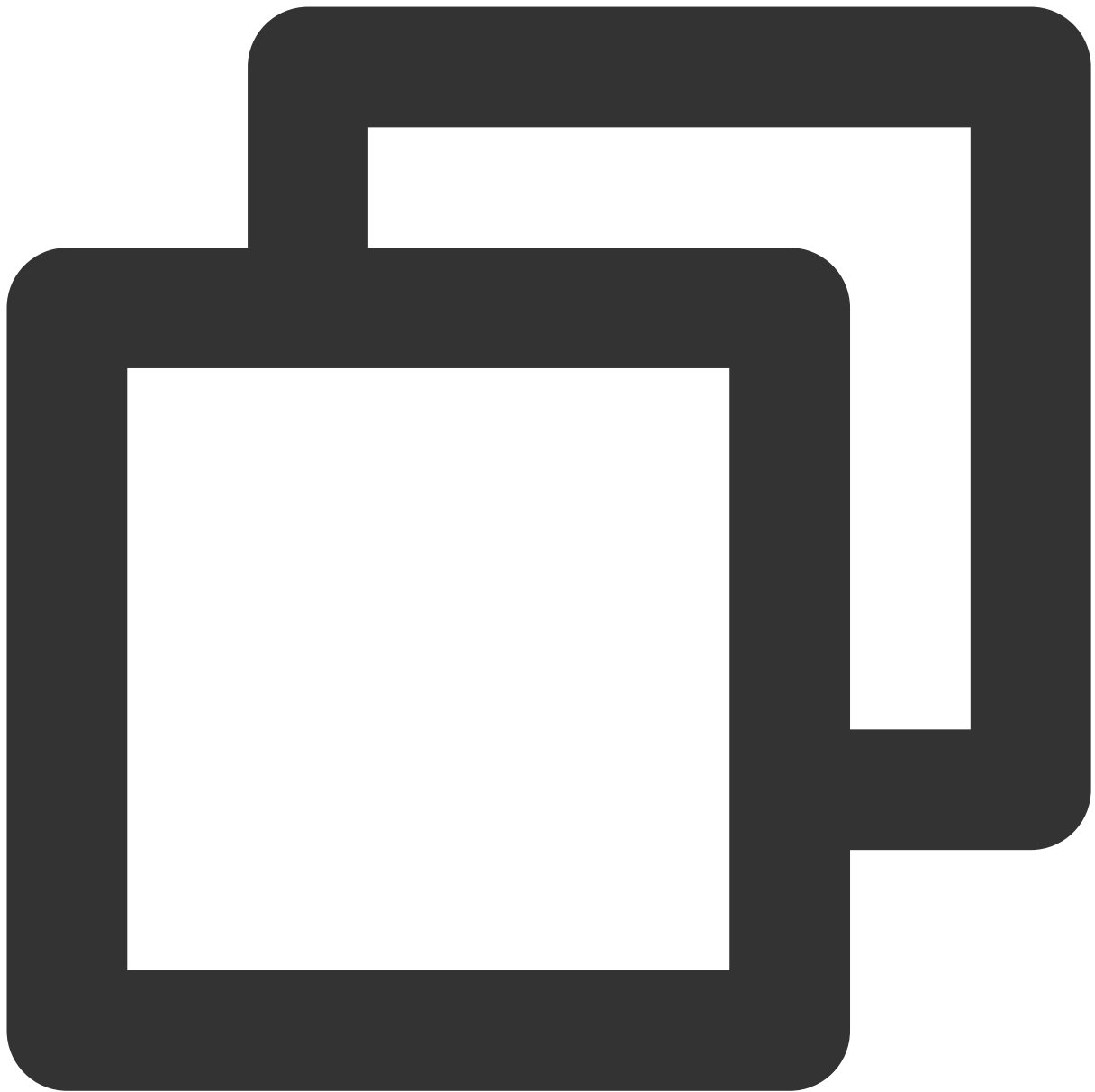


```
include (":tuicontact")
```

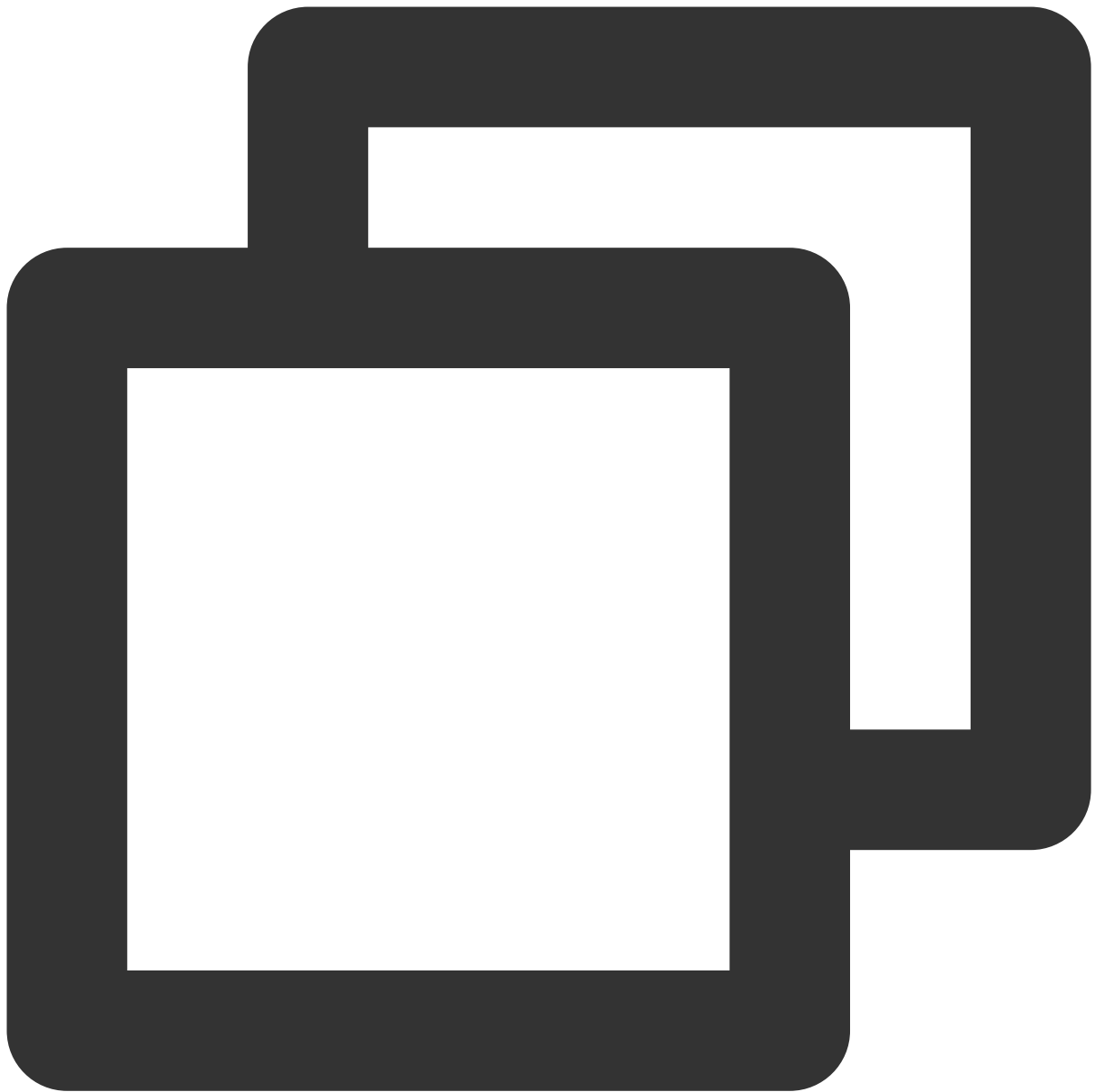
Find the `build.gradle` (or `build.gradle.kts`) file in the app directory, and add the following code to declare the current app's dependency on the newly added `tuicontact` component.

`build.gradle`

`build.gradle.kts`



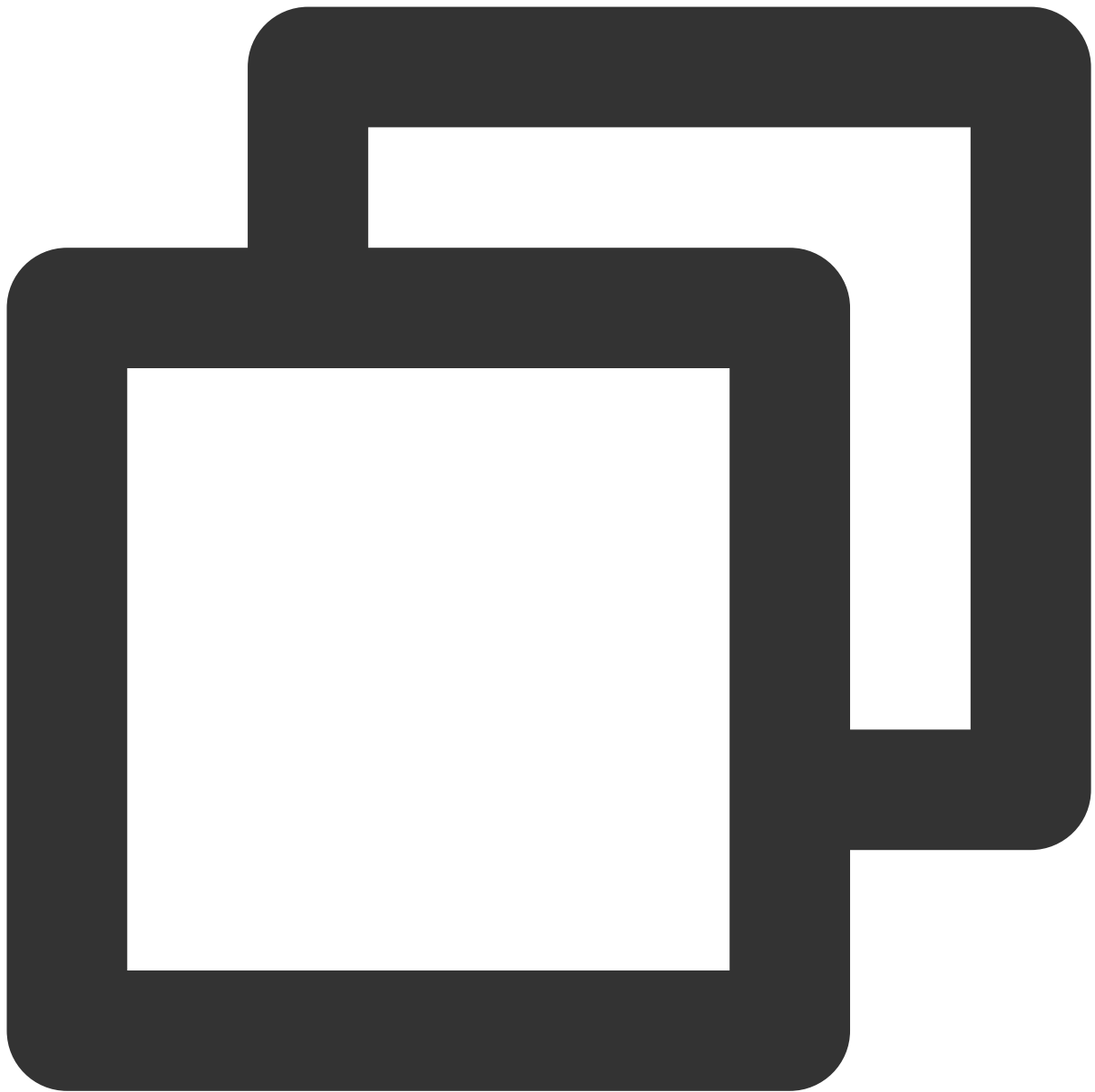
```
implementation project(':tuicontact')
```



```
implementation(project(":tuicontact"))
```

## 2. Add Friend

You can add a friend using the code below after logging in. The friend you add will appear in the Invite Members screen for you to select. If you want to officially go live and use the Chat Friend Relationship, please refer to [Chat Friend Management Documentation](#) or [Adding Friends RestAPI](#).



```
// Add a friend
V2TIMFriendAddApplication application = new V2TIMFriendAddApplication("userId");
application.setAddType(V2TIMFriendInfo.V2TIM_FRIEND_TYPE_BOTH);
V2TIMManager.getFriendshipManager().addFriend(application, new V2TIMValueCallback<V
    @Override
    public void onSuccess(V2TIMFriendOperationResult v2TIMFriendOperationResult) {
        // Friend request sent successfully
    }

    @Override
    public void onError(int code, String desc) {
```

```
// Failed to add the friend
}
});
```

## How to experience the invite member feature

First, refer to [Run Sample Code](#), and complete the demo. In the `members.json` file of the demo project, some test user information has been pre-configured. You can choose two accounts, log in with the configured `userId` on two different phones, and then select the other user when scheduling or editing a meeting. This way, the scheduled meeting will appear in the other user's schedule list.

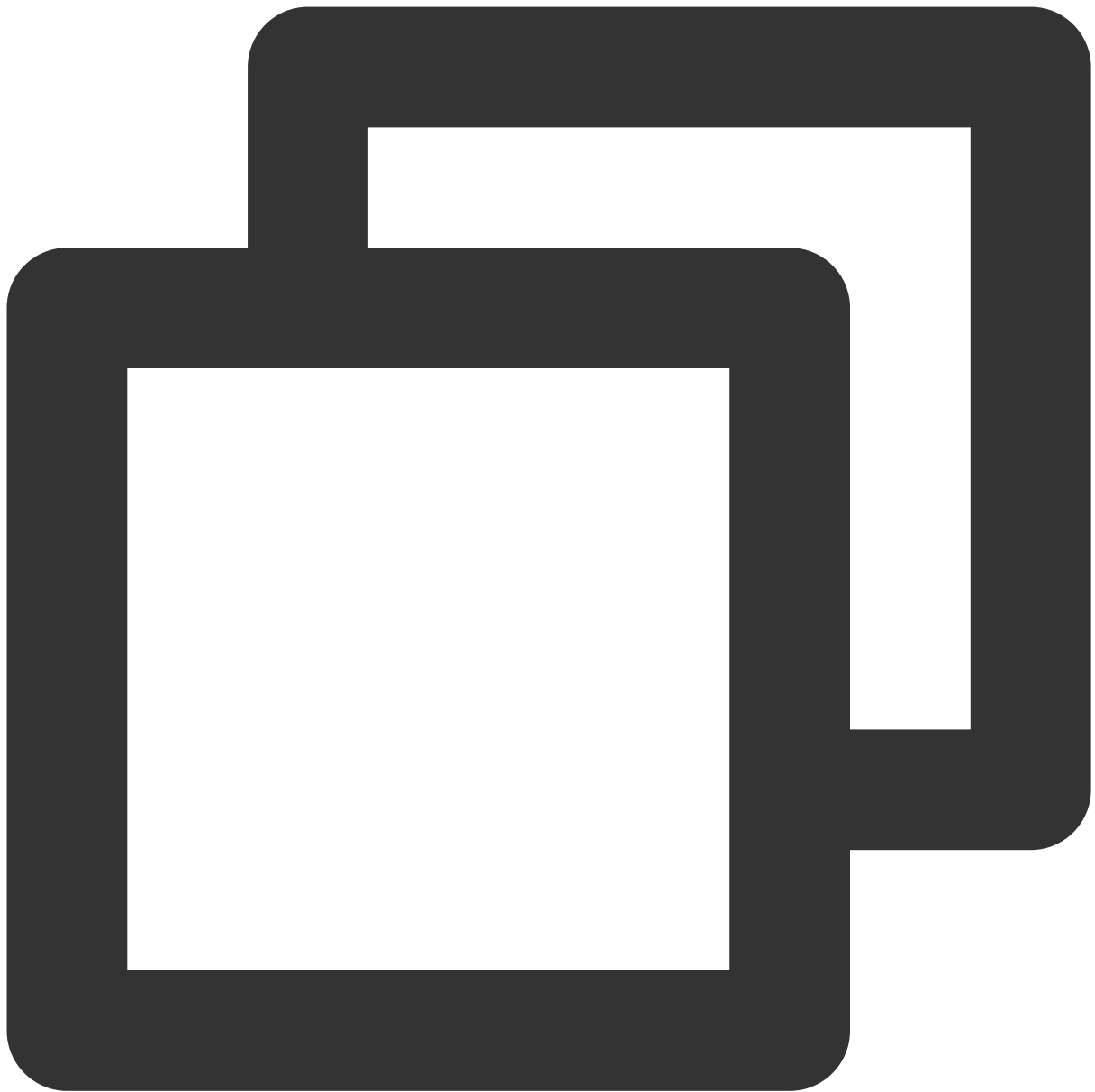


## Code Implementation for Member List Page

Considering the complexity of the user list data on the invite members page, we have designed a solution that allows you to customize the Member Selection Interface. Next, we will guide you on how to integrate your own member selection page (of course, you can also directly use the UI provided in the demo, which will be introduced later).

1. Prepare your Friend Selection Page viewController, implementing the `SelectMemberControllerProtocol` protocol.

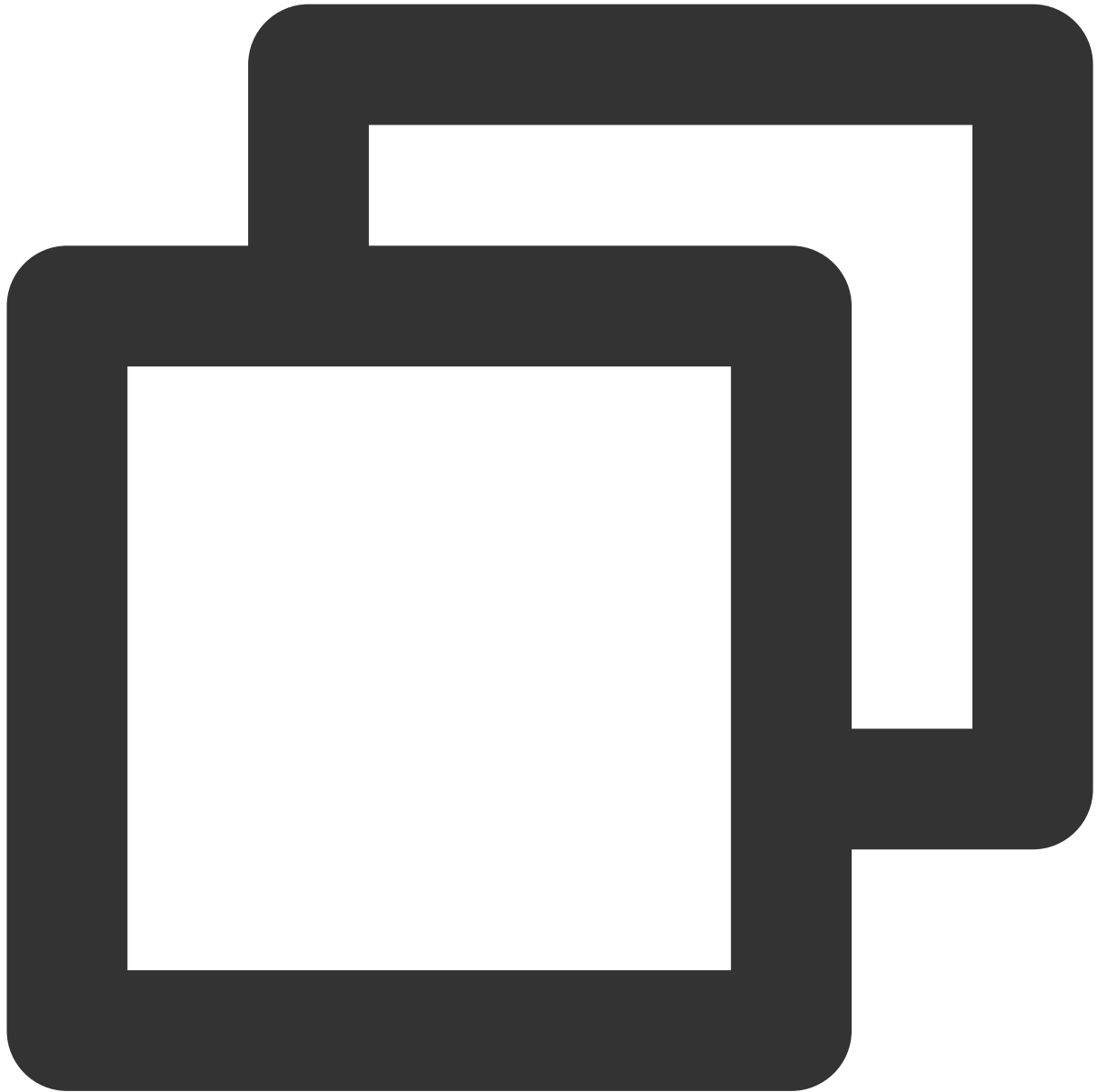




```
// Sample Code
class SelectMemberViewController: UIViewController, SelectMemberControllerProtocol {
    weak var delegate: MemberSelectionDelegate?
    var selectedUsers: [User]

    func didSelectFinished() {
        // Through the delegate, call back the selected members to Conference in the me
        delegate?.onMemberSelected(self, invitees: selectedMembers)
    }
}
```

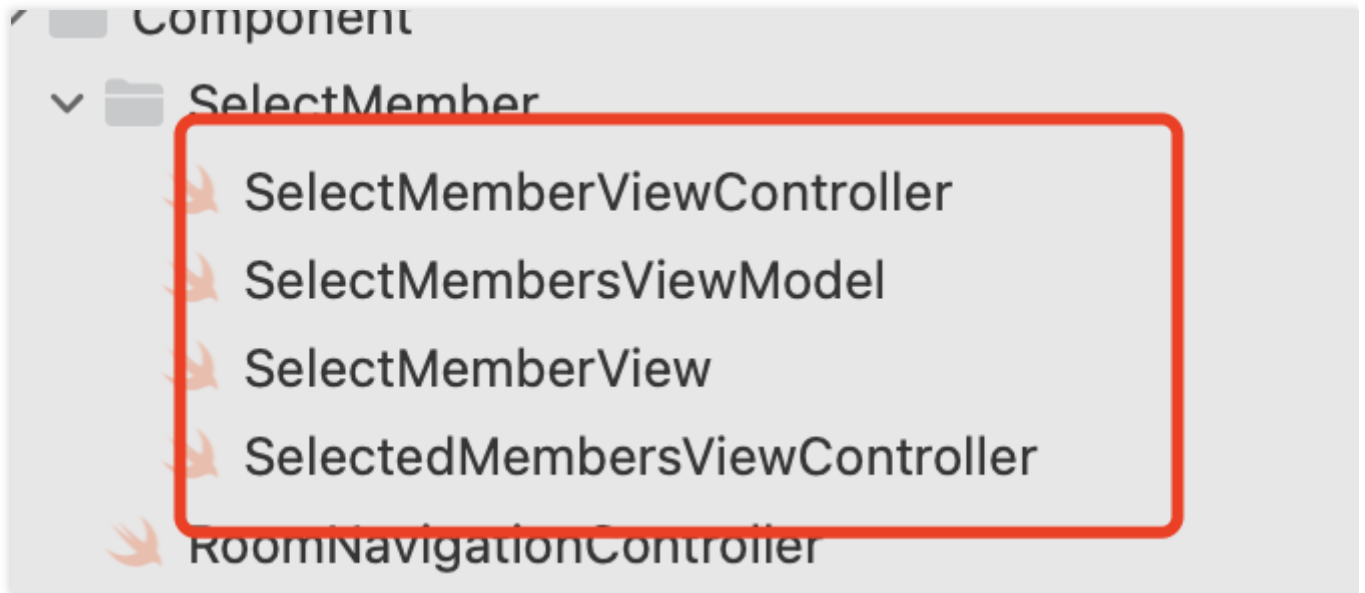
2. When initializing `ScheduleConferenceViewController` and `ConferenceListView`, pass in a closure. In the closure, return the `SelectMemberViewController` you created in `Back`.



```
// Sample Code
class YourViewController: UIViewController {
    func jumpToScheduleViewController {
        let scheduleViewController = ScheduleConferenceViewController { selectedList in
            return SelectMemberViewController(selectedUsers: selectedList)
        }
        navigationController?.pushViewController(scheduleViewController, animated: true)
    }
}
```

```
}
```

3. With the above two steps, you can display your own member selection page. We also provide the page code in the demo as shown in the image above. You can directly copy the following files into your project to get our example page.



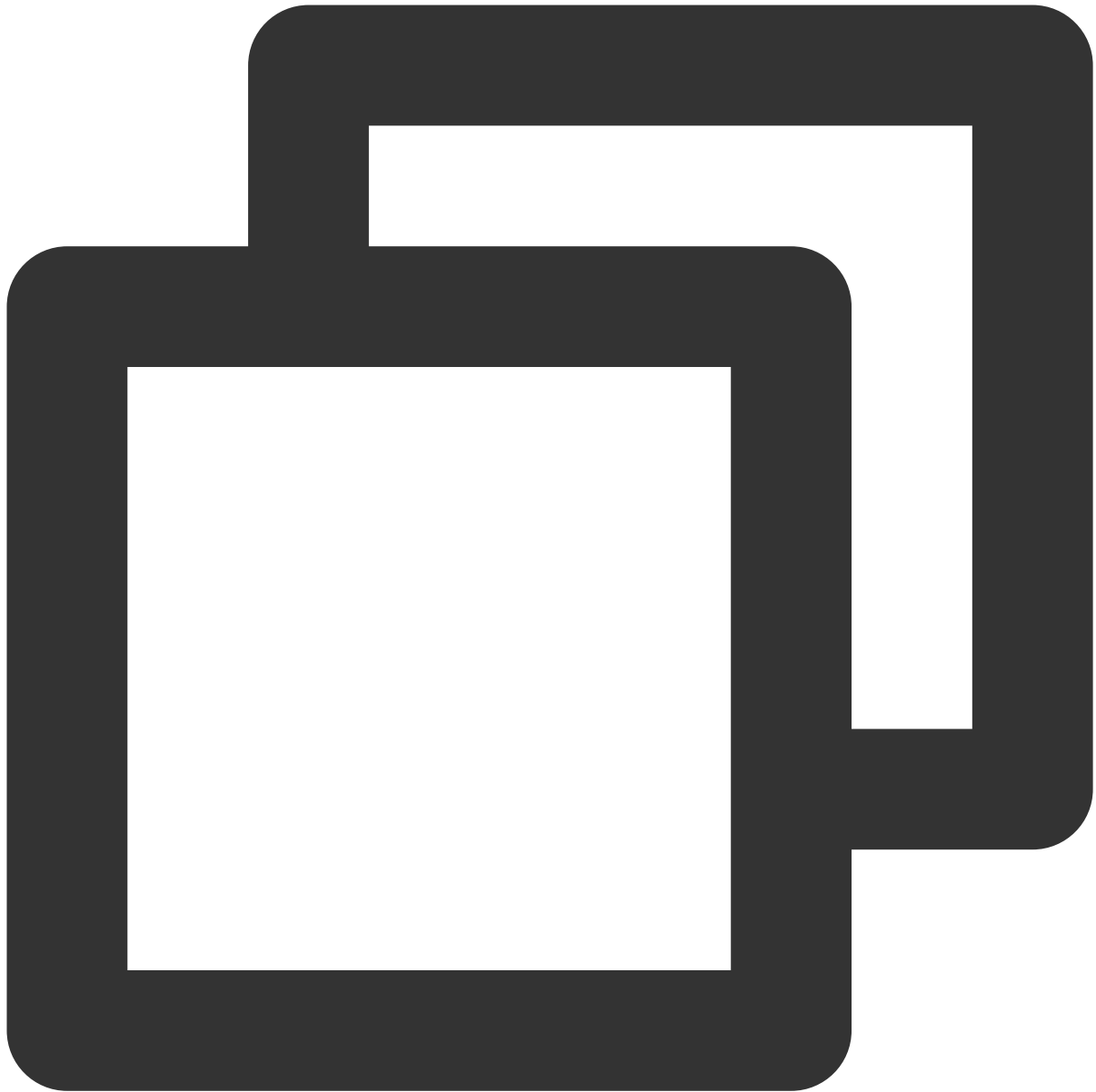
In the `loadMembers` method of `SelectMembersViewModel`, you can load your own member list data (you can also directly obtain Chat Relationship Chain Data).

When you need to use the member invitation feature to invite members into the conference, we provide the following two solutions for you to add the members you need to invite:

### Solution 1: Using json to configure user information

You can refer to our [Example Project](#). In the example/assets directory, `members.json` stores the user information required for invitations. Follow these steps:

1. In the `assets` directory of your project, create a new `members.json` file listing all the user information you need. The file format should be the same as the `members.json` mentioned above.
2. In the `pubspec.yaml` file of your project, complete the following configuration:



```
assets:  
  - assets/members.json
```

After completing the above configuration, you can select the members listed in `members.json` in the member invitation interface.

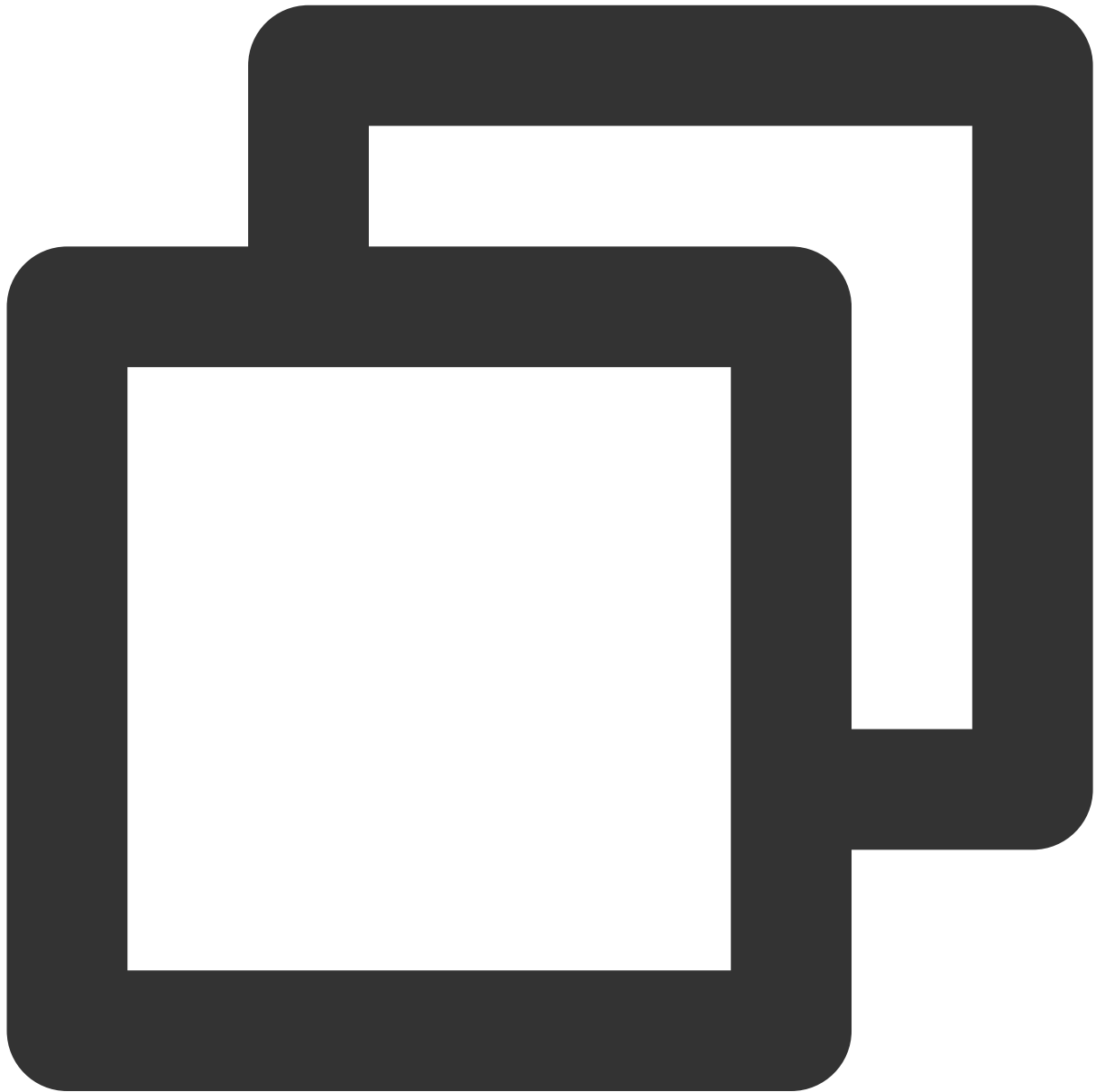
## Solution 2: Using Chat Buddy Information

If you have not configured the `members.json` mentioned above, the invite friends interface will default to pulling your Chat Buddy Information. When you need to invite other members to the conference, you can add the members

you want to invite as friends according to your business needs.

Prerequisites: log in to Chat SDK, the login process is the same as the [log in to Floating Chat](#) process. If you have already completed the login process or are using [In-Conference Chat](#), you can skip this step.

The code for adding friends is as follows:



```
import 'package:tencent_cloud_chat_sdk/manager/v2_tim_manager.dart';

// In Flutter Conference, there is already a dependency on tencent_cloud_chat_sdk,
// Replace the userID in the code with the UserID of the user you want to add to co
```

```
V2TIMManager().getFriendshipManager().addFriend(  
    userID: 'userID', addType: FriendTypeEnum.V2TIM_FRIEND_TYPE_BOTH);
```

After adding friends, you can choose to invite the added users **each time** you make a reservation.

If you need more related friend operations, please refer to: [Friend Management](#). If you need to use Rest API to batch add friends, please refer to: [Adding Friends RestAPI](#).

## Notes

After successfully scheduling a conference, you can get the conference id and reservation information.

To batch schedule conferences at different dates/times, select the times and submit in batches.

The start time for scheduling a conference cannot be earlier than the current time, but there is no limit on the number of days in advance.

After the scheduled conference reaches the end time, if the conference has not been dissolved and there are no users in the conference, the conference will be retained for 6 hours from the scheduled end time. During this period, you can still enter the conference at any time.

## Exchange and Feedback

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

# Web&Electron

Last updated : 2024-08-15 17:58:27

## Function Introduction

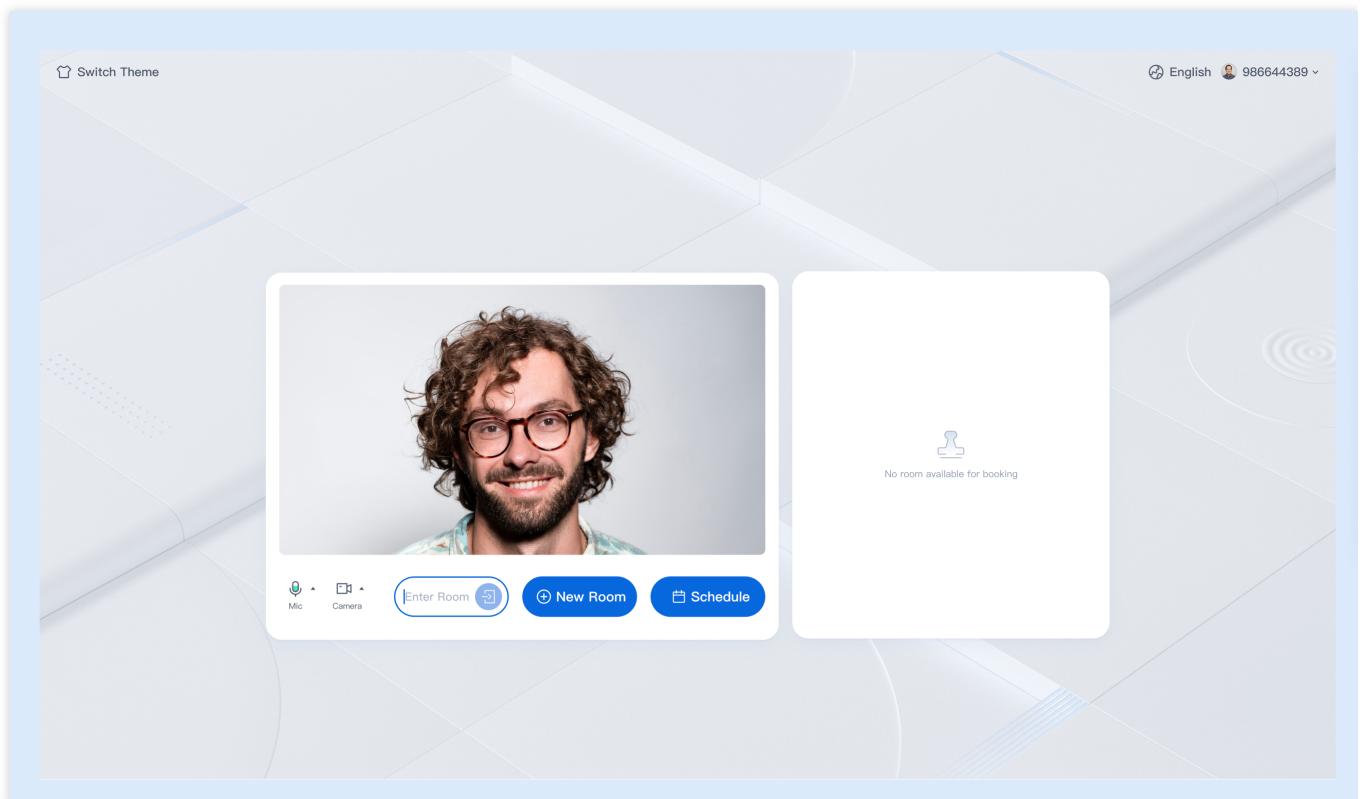
TUIRoomKit introduces a new feature for schedule rooms, which allows users to reserve a room and schedule a meeting on their schedule.

### Note :

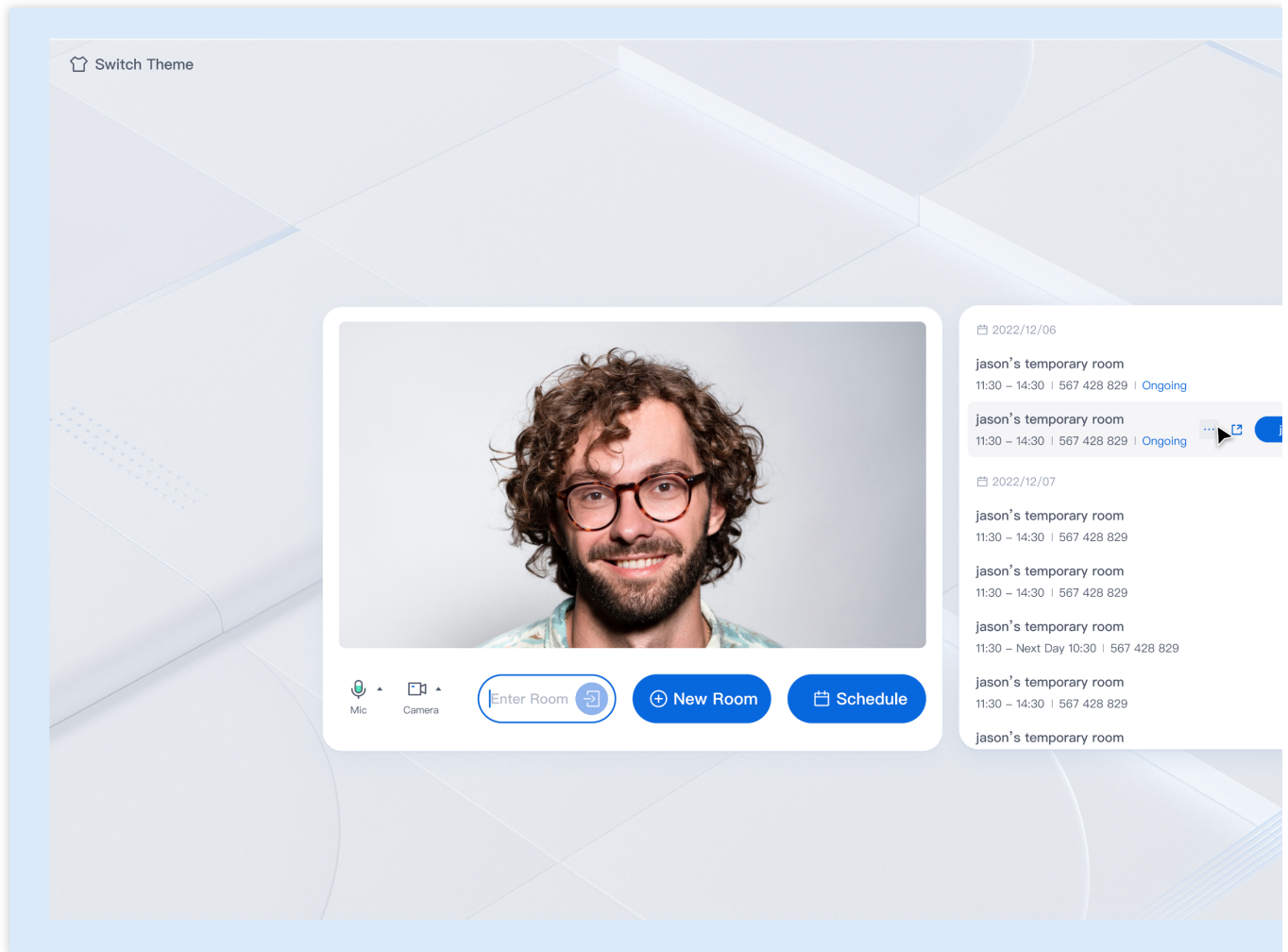
Since v2.5.0, TUIRoomKit supports the ability to schedule rooms, view room list, modify room information, etc. Integrate with the latest version of TUIRoomKit to experience the room schedule process.

## How to schedule a room

1. On the TUIRoomKit Preview screen, click **Schedule** > Fill in the room reservation information and set up related privileges > Complete the settings and click **Schedule**.



2. The schedule result will be synchronized with the schedule result in the list of **schedule rooms** on the right side of the preview page, and it supports operations such as **viewing details**, **modifying rooms**, **canceling rooms**, **copying invitation information**, and so on. In addition, you can click **join** to enter the schedule room.



## Terms of preparation

Before you can use the room schedule feature provided by Tencent Cloud, you need to go to the console and enable the multiplayer conferencing [service for the application](#). See Turning on the service for more information on how to do this. Next, you need to bring in the TUIRoomKit component, as described in [Quick Run-Through](#).

### Schedule Room example run-through

#### Note :

You need to introduce **PreConferenceView** (the pre-conference preview component) and **ConferenceMainView** (the main body of the TUIRoomkit UI component). In the example, the **v-if** and **v-else**



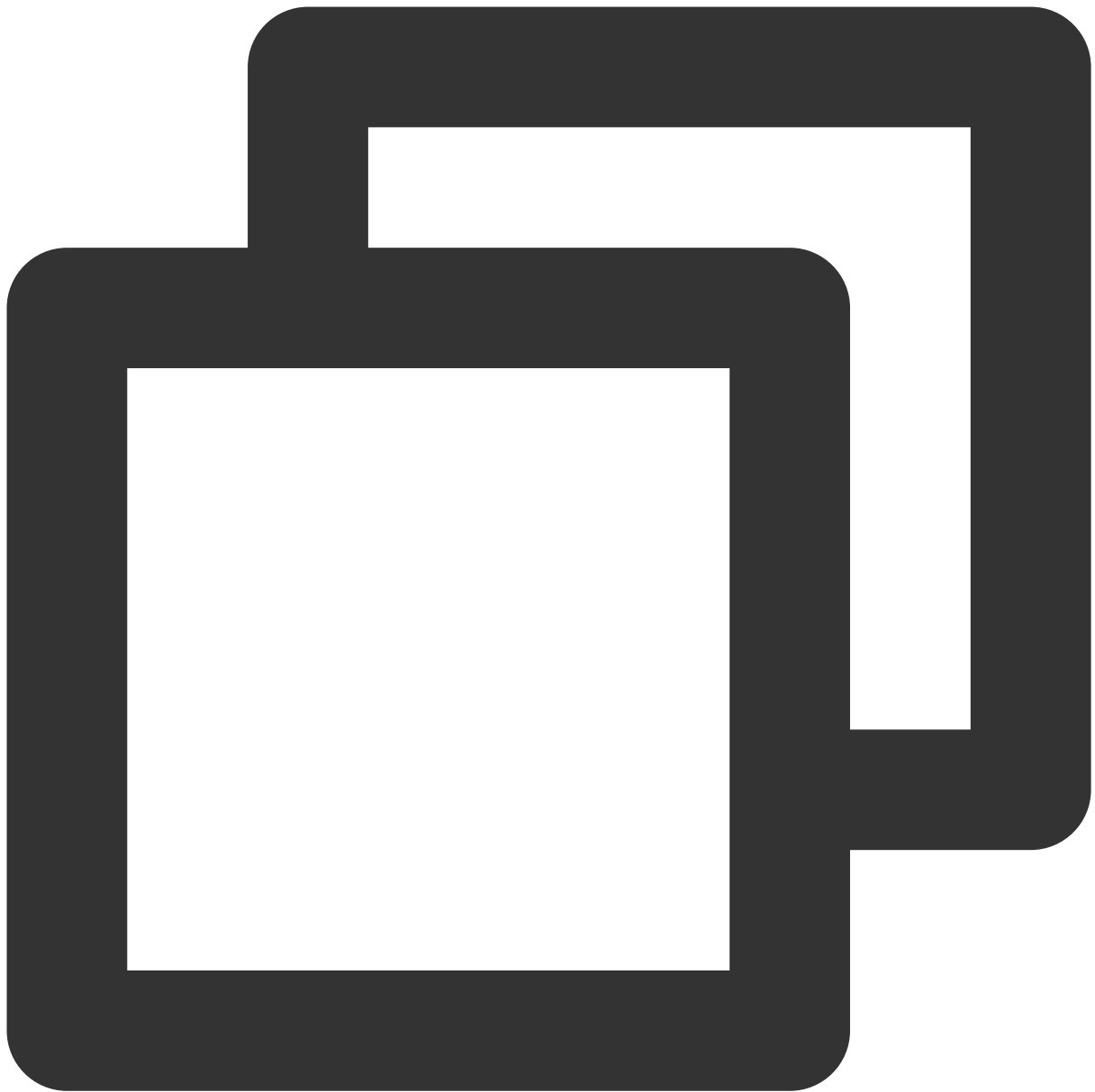
directives are used to control the showing and hiding of the two components, and you can also use route jumping to switch between the components.

In the **PreConferenceView** component, you control whether the scheduled room features are shown or hidden by setting the value of the **enable-scheduled-conference** property. In addition, the component listens to the on-enter-room event, so when the user clicks into the room, the [join](#) interface is invoked by triggering the **handleEnterRoom** method.

In the **ConferenceMainView** component, the **on-destroy-room** event is listened to and the **onDestroyRoom** method is triggered when the room is destroyed.

Web

Electron



```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // Setting whether to enable the schedule r
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
```

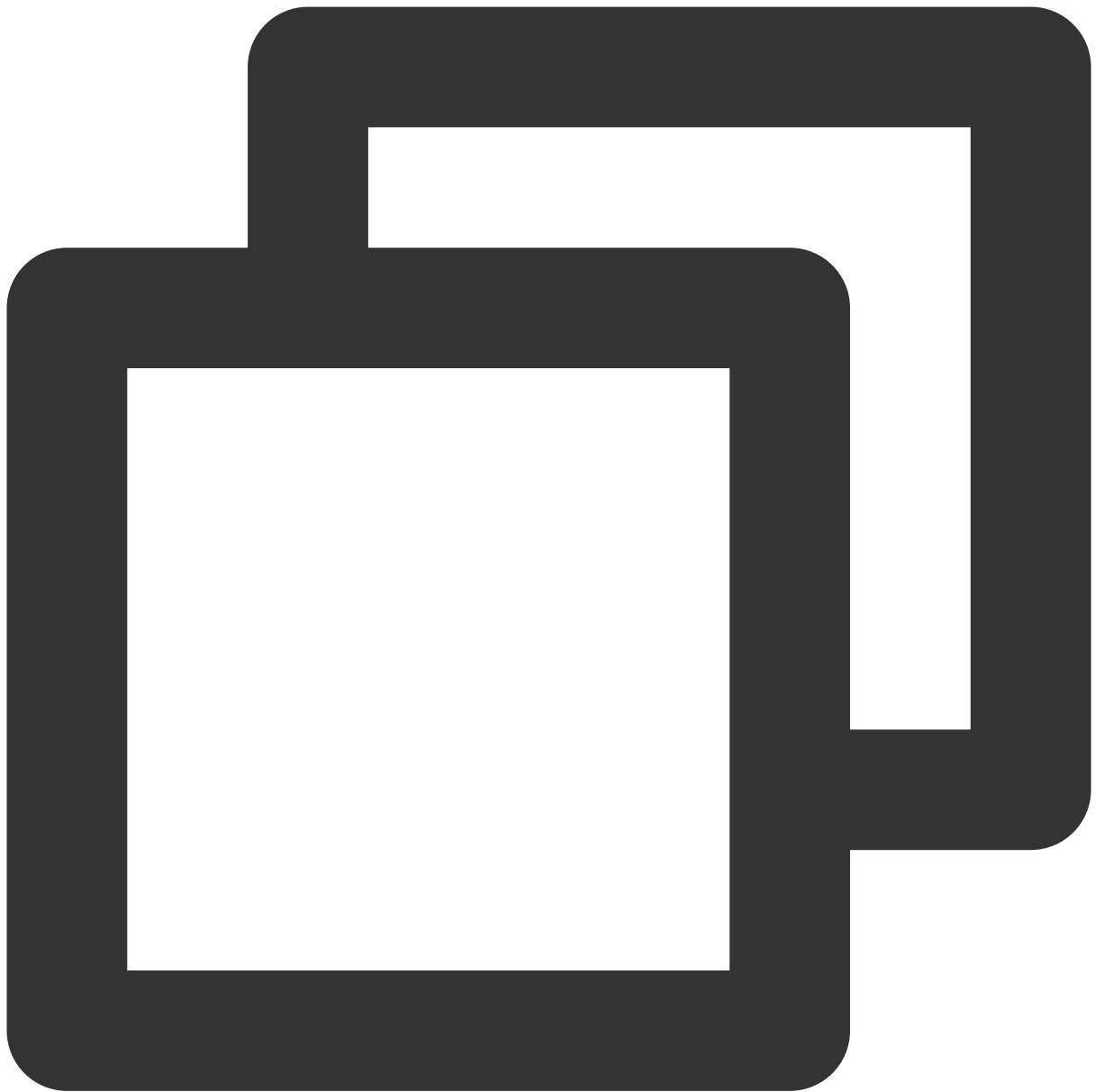
```
</template>

<script setup lang="ts">
import { ref } from 'vue';
// Note the package name, if you are using the vue2 version change the package name
import { PreConferenceView, conference, ConferenceMainView } from '@tencentcloud/rtc-sdk-vue';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
}
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```



```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // Setting whether to enable the schedule r
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
```

```
</template>

<script setup lang="ts">
import { ref } from 'vue';
// Note the package name, if you are using the vue2 version change the package name
import { PreConferenceView, conference, ConferenceMainView } from '@tencentcloud/rtc-sdk-vue';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
}
init();

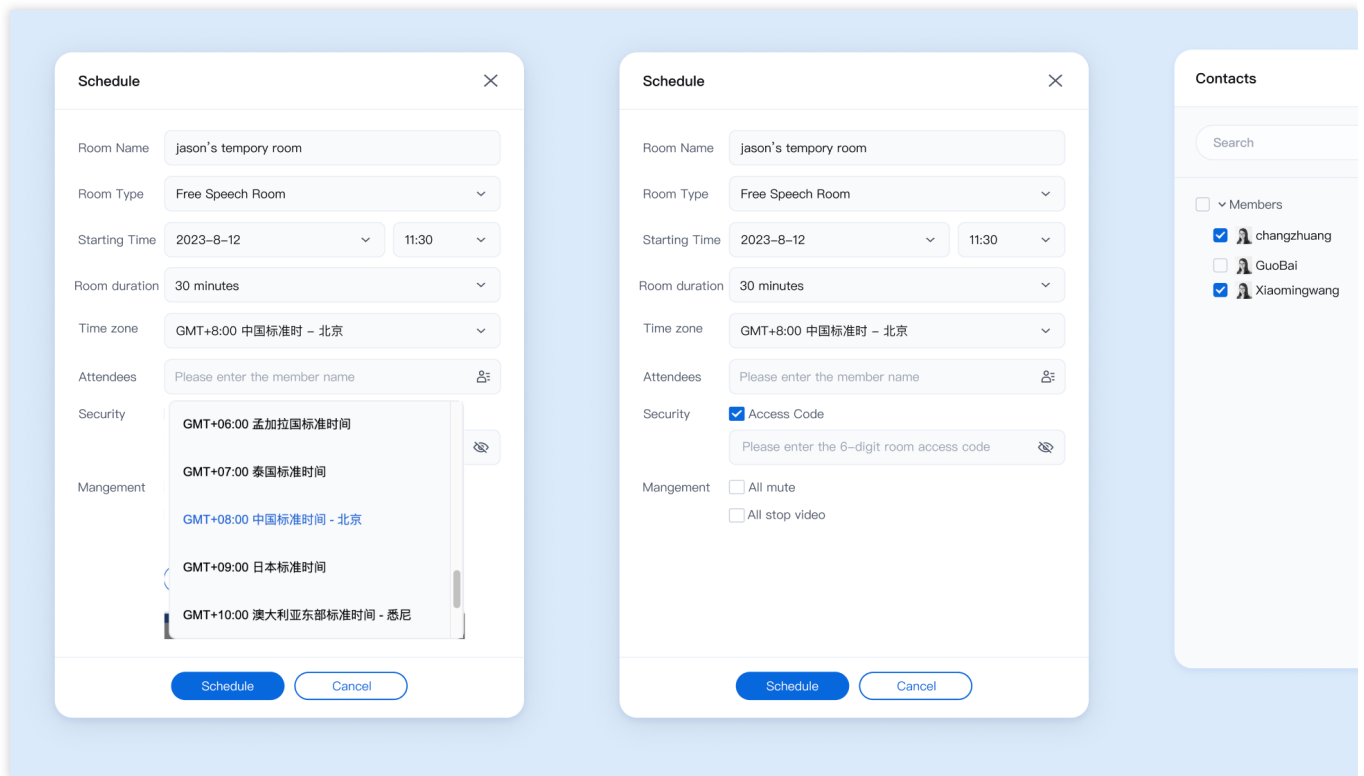
async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

## Schedule Room Control

### Schedule Room

After clicking **Schedule** on the Preview page, the schedule Room Setting box pops up, and users can set the room information according to their needs, which can be set to include: **room name, room type, start time, room duration, time zone, attendees, security (access code), management (all mute, all stop video)**, and so on.

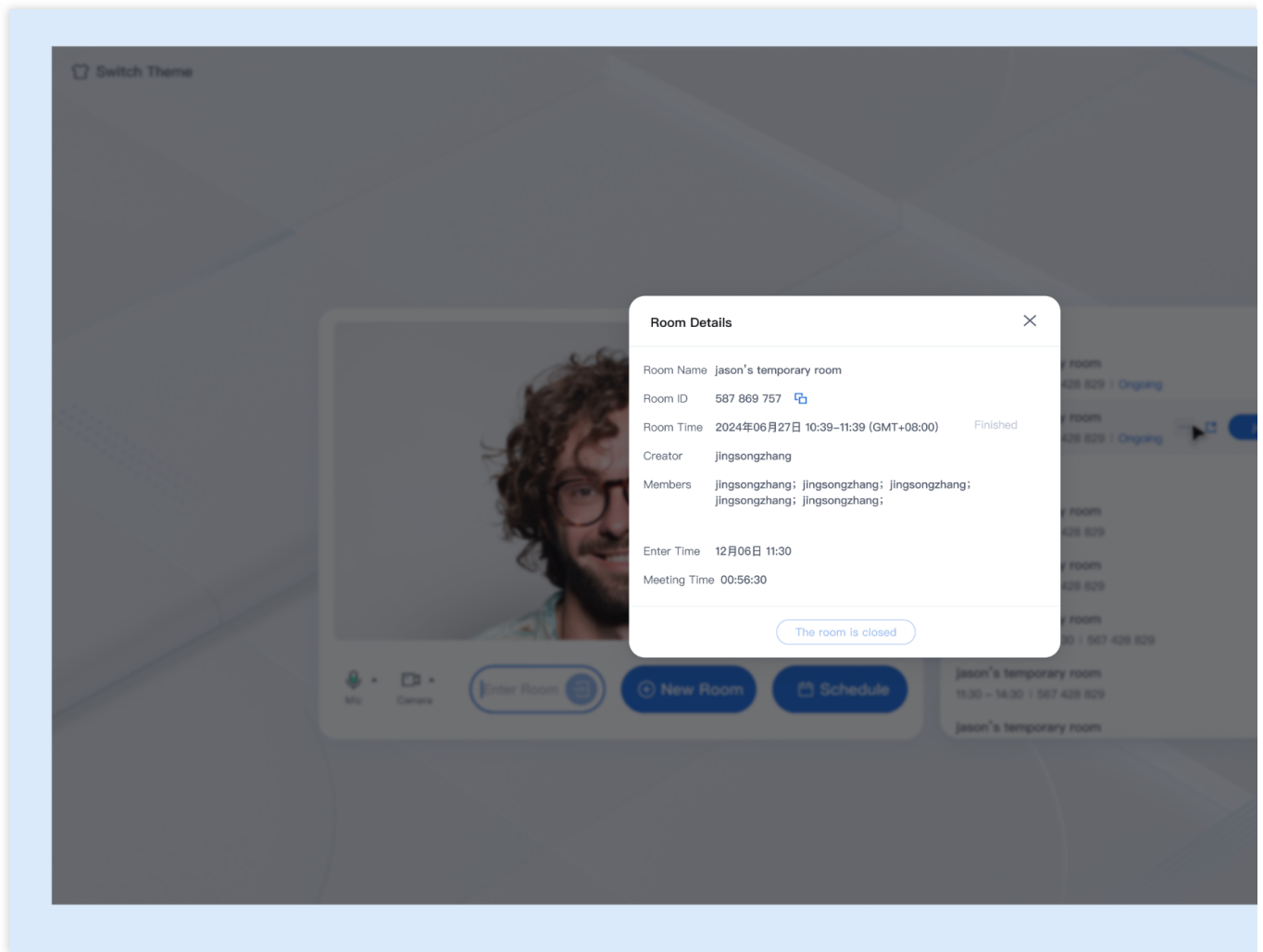


### Note :

**TUIRoomKit's** invite members list is derived from **IM friends**, so you need to use **IM to add friends**. You can replace the user data by **adding IM friends**. In this case, you need to use the **IM REST API** to get the **IM** friend chain data by adding a friend relationship. If you are importing data from an attendee's address book or adding users, see [Adding Friends](#). If you want to remove an attendee, you can delete the buddy relationship in the IM relationship chain, see [Deleting Buddies](#).

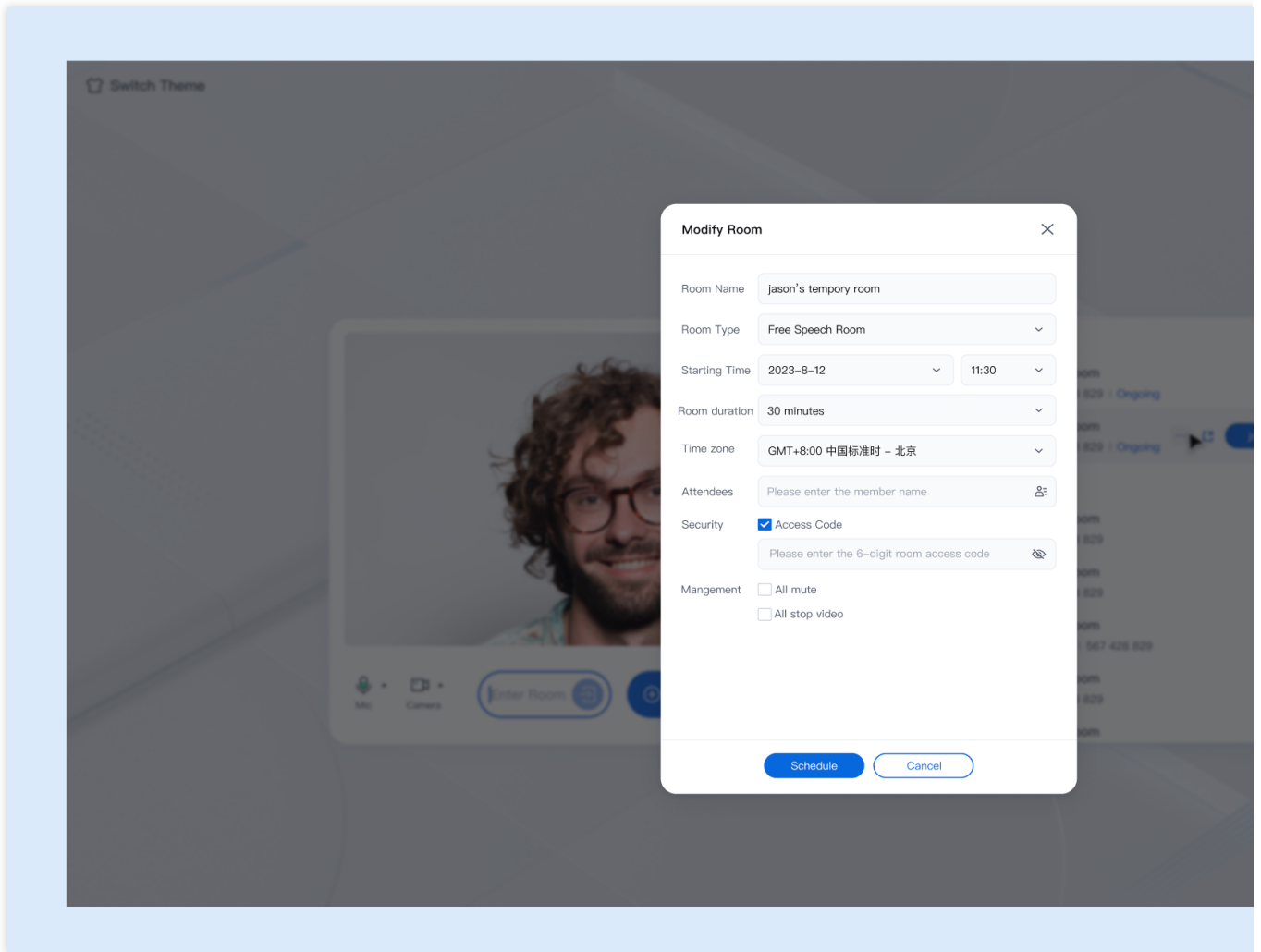
### View Details

Users can click **View Details** to view the details of the corresponding schedule room.



## Modify room information

The room owner can modify the information of the schedule room, after modification, click **Schedule** to adjust the information of the currently schedule room to the modified room information.



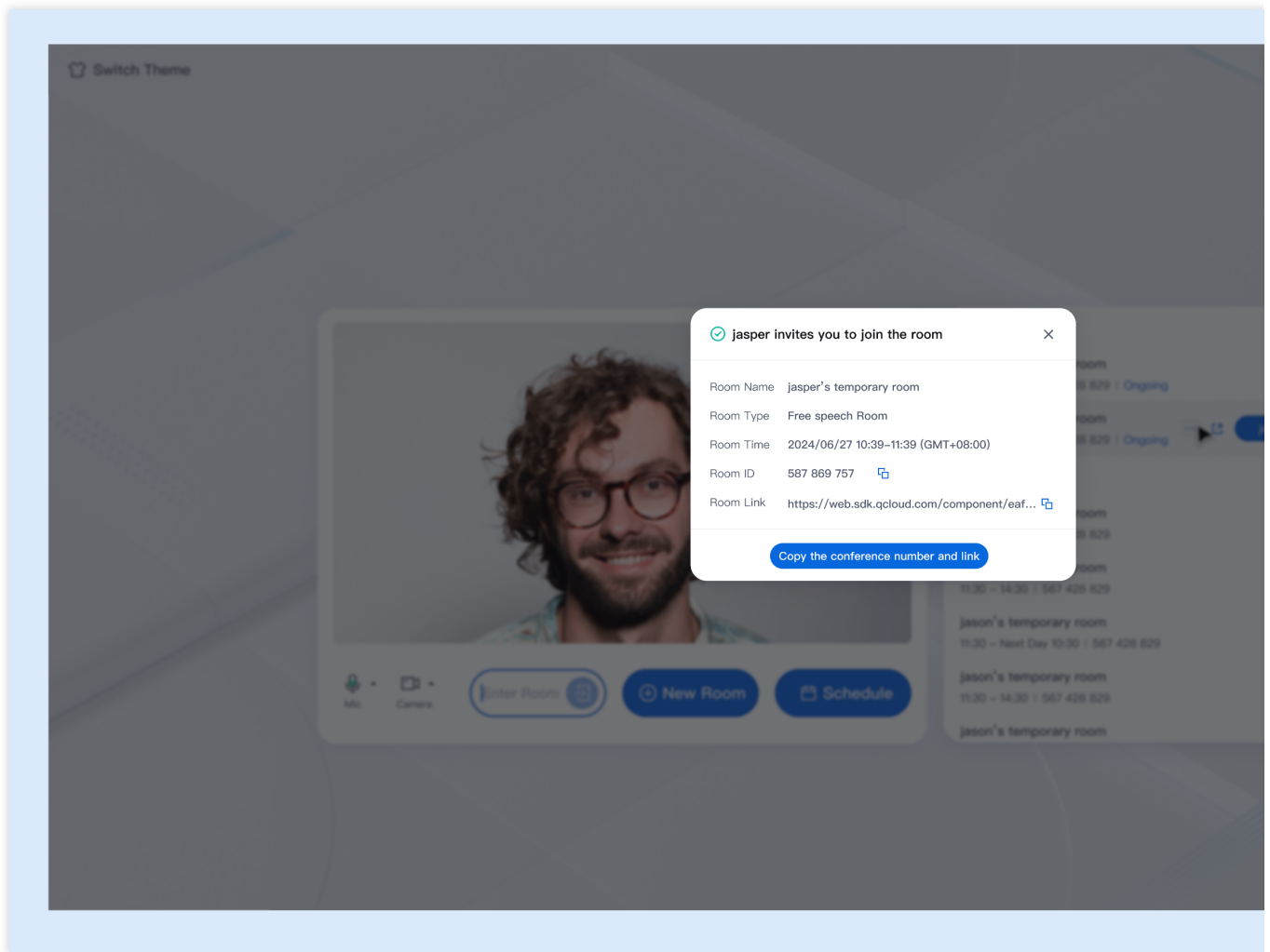
## Invitation to the room

Users can click



Invite to bring up the meeting information invitation box, and **Copy the conference number and link** to the pasteboard to share with other users by clicking Copy Meeting Number & Link.





## Caveat

Room reservations cannot start earlier than the current time, but there is no limit to the number of days in advance.

If you want to schedule rooms for different dates/times at the same time, just select the times and submit them at the same time.

Once a room is schedule, the room number will be reserved for 6 hours from the start time of the schedule, if the room is not occupied, during which time you can return to the room at any time.

The room number and reservation information will be available once the room is successfully schedule.

## Communication and feedback

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

# UI Customization (TUIRoomKit)

## Electron

Last updated : 2024-06-14 11:49:41

This article will detail how to customize the TUIRoomkit user interface. TUIRoomkit offers two customization methods: one is through the simple use of Custom UI API, and the other is by replacing existing UI components. We will introduce these methods in detail below.

### Scheme 1: Interface Fine-tuning

TUIRoomkit provides a series of [API](#), allowing for easy customization of the UI. The table below lists some of the main APIs and their functions:

API	Description
<a href="#">setLanguage</a>	Set the interface language.
<a href="#">setTheme</a>	Set the interface topic.
<a href="#">enableWatermark</a>	Enable the text messaging feature in the application. For details, see: <a href="#">Text Watermark</a> .
<a href="#">disableTextMessaging</a>	Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.
<a href="#">disableScreenSharing</a>	Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.
<a href="#">hideFeatureButton</a>	Hide specific feature buttons in the application. By invoking this function and passing in the appropriate <a href="#">FeatureButton</a> enumerated values, the corresponding buttons will be hidden from the user interface.

### Scheme 2: UIKit Source Code Export and Modification

You can directly modify the UI source code we provide to adjust the TUIRoomKit user interface according to your needs. Execute the following node script to automatically copy the TUIRoomKit interface source code into your project (default path `./src/components/TUIRoom` ).

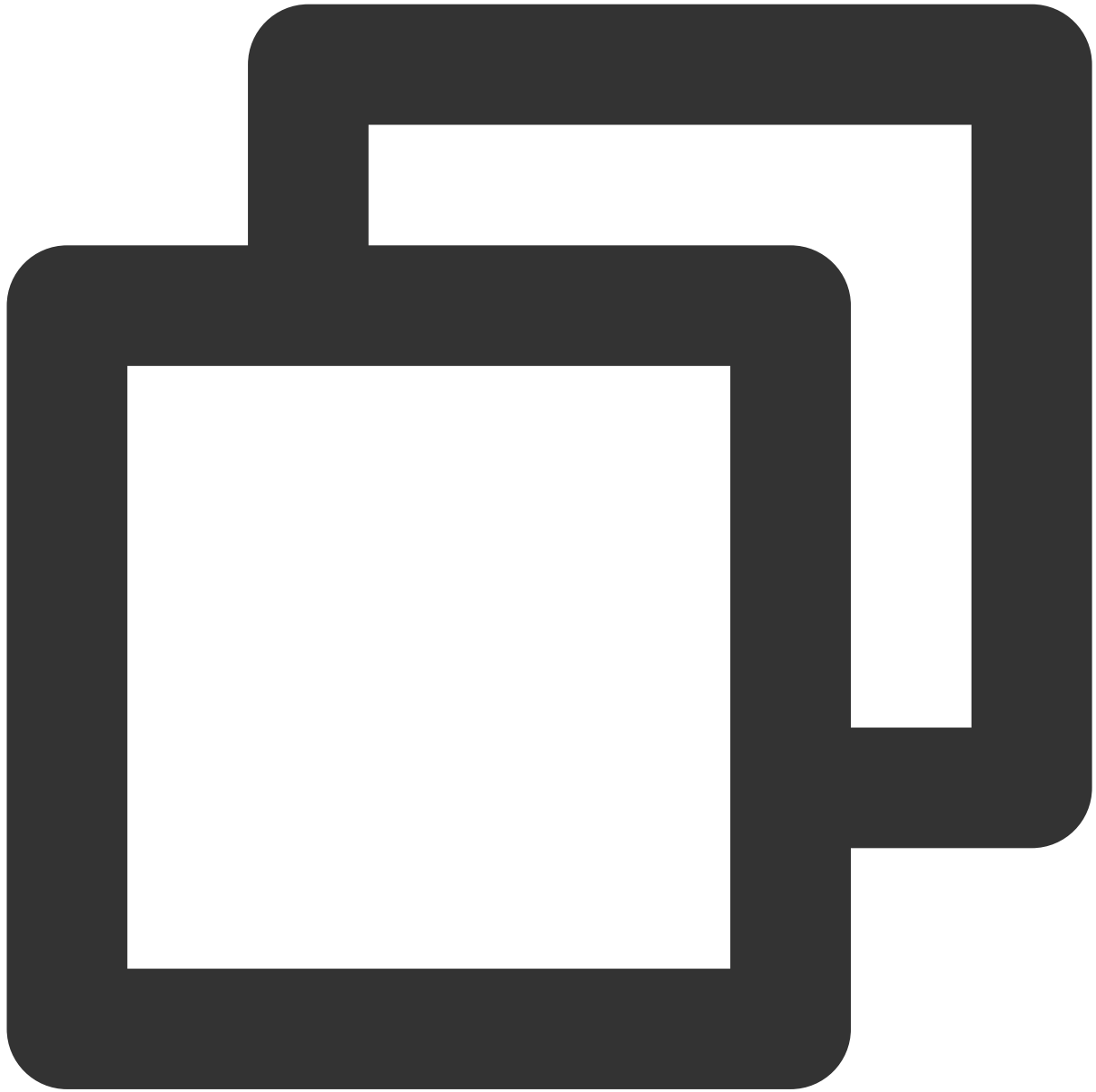
**Note:**

After exporting the source code, you need to manually change the reference of the TUIRoom component from the npm

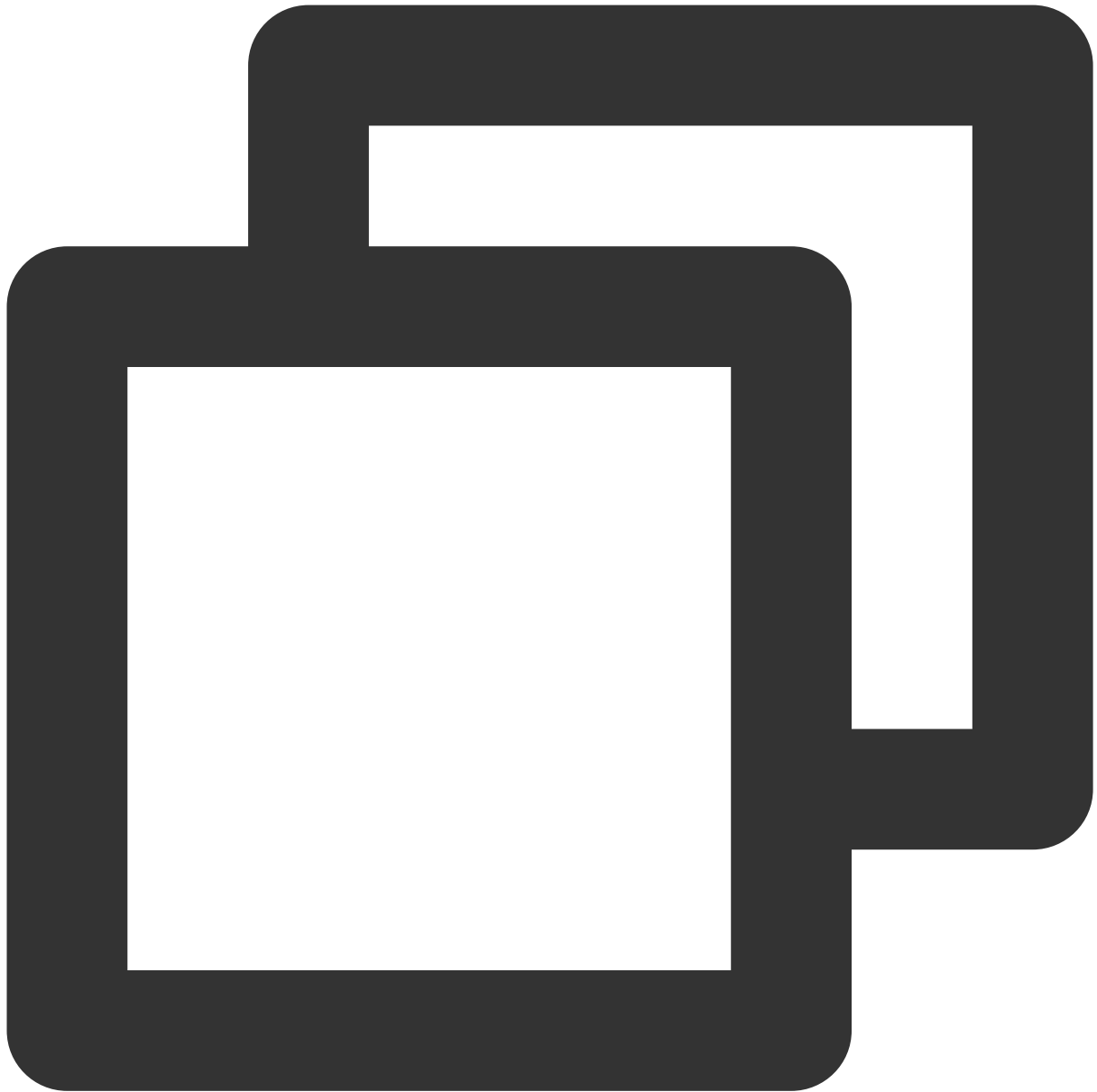
package address to the relative path address of the TUIRoom source code.

Vue3

Vue2



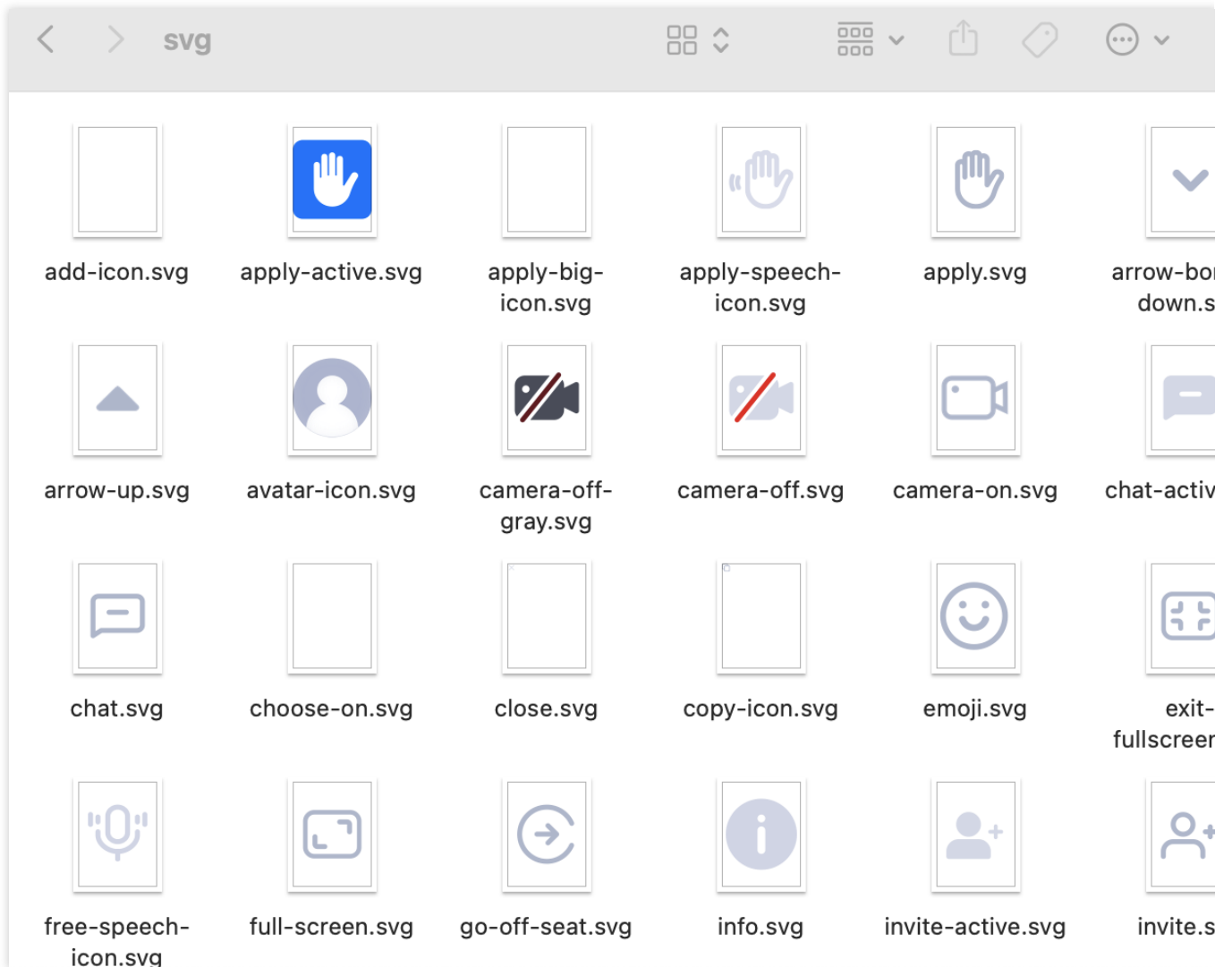
```
node ./node_modules/@tencentcloud/roomkit-electron-vue3/scripts/eject.js
```



```
node ./node_modules/@tencentcloud/roomkit-electron-vue2.7/scripts/eject.js
```

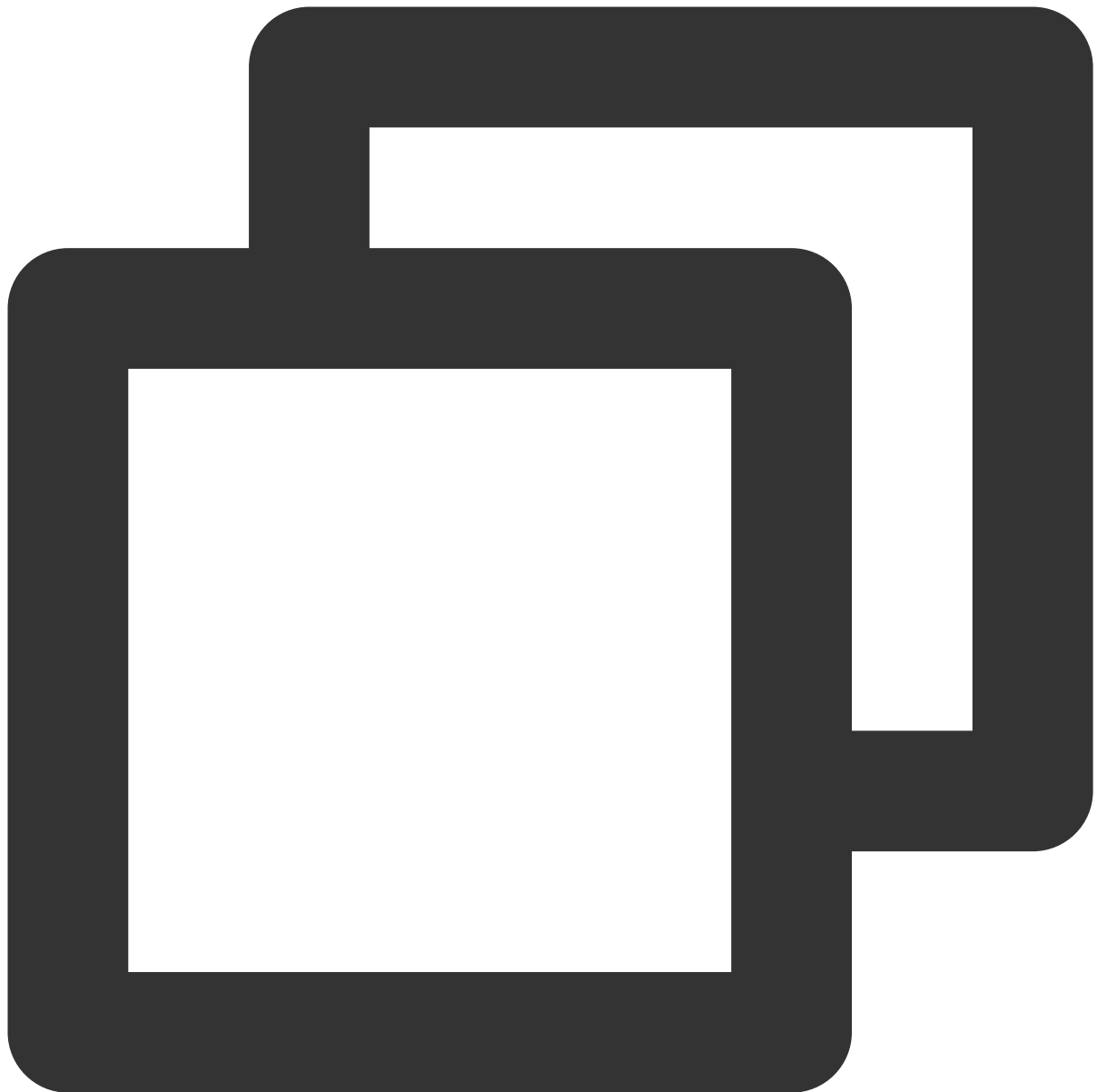
## 1. Replace Icons

You can directly modify the icon components in the `/TUIRoom/assets/icons/svg` folder, to ensure the icon color and style are consistent throughout the app. Please keep the icon file names unchanged while replacing them.



## 2. Adjust UI Layout

You can adjust the UI layout of the multi-person video conference interface by modifying different components in the `/TUIRoom/components/` folder



```
- components/  
  - Chat                Chat  
  - common              Public Icon Components  
  - ManageMember        Member Management  
  - RoomContent          Room Video  
  - RoomFooter           Room Page Footer  
  - RoomHeader           Room Page Header  
  - RoomHome             Home Page  
  - RoomInvite           Invite Members  
  - RoomLogin            Login Page  
  - RoomMore             More
```

- RoomSetting Settings
- RoomSidebar Drawer Components

## Solution 3: Implement Your Own UI

The overall feature of TUIRoomKit is based on the TUIRoomEngine, a UI-less SDK. You can fully implement your own UI interface based on TUIRoomEngine. For more details, see:

[TUIRoomEngine Integration Guide](#)

[TUIRoomEngine API Address](#)

# Android

Last updated : 2023-09-25 10:49:44

This article will introduce how to customize the User Interface of TUIRoomKit. We provide two solutions for you to choose from: **Fine-tuning Solution** and **Self-implemented UI Solution**.

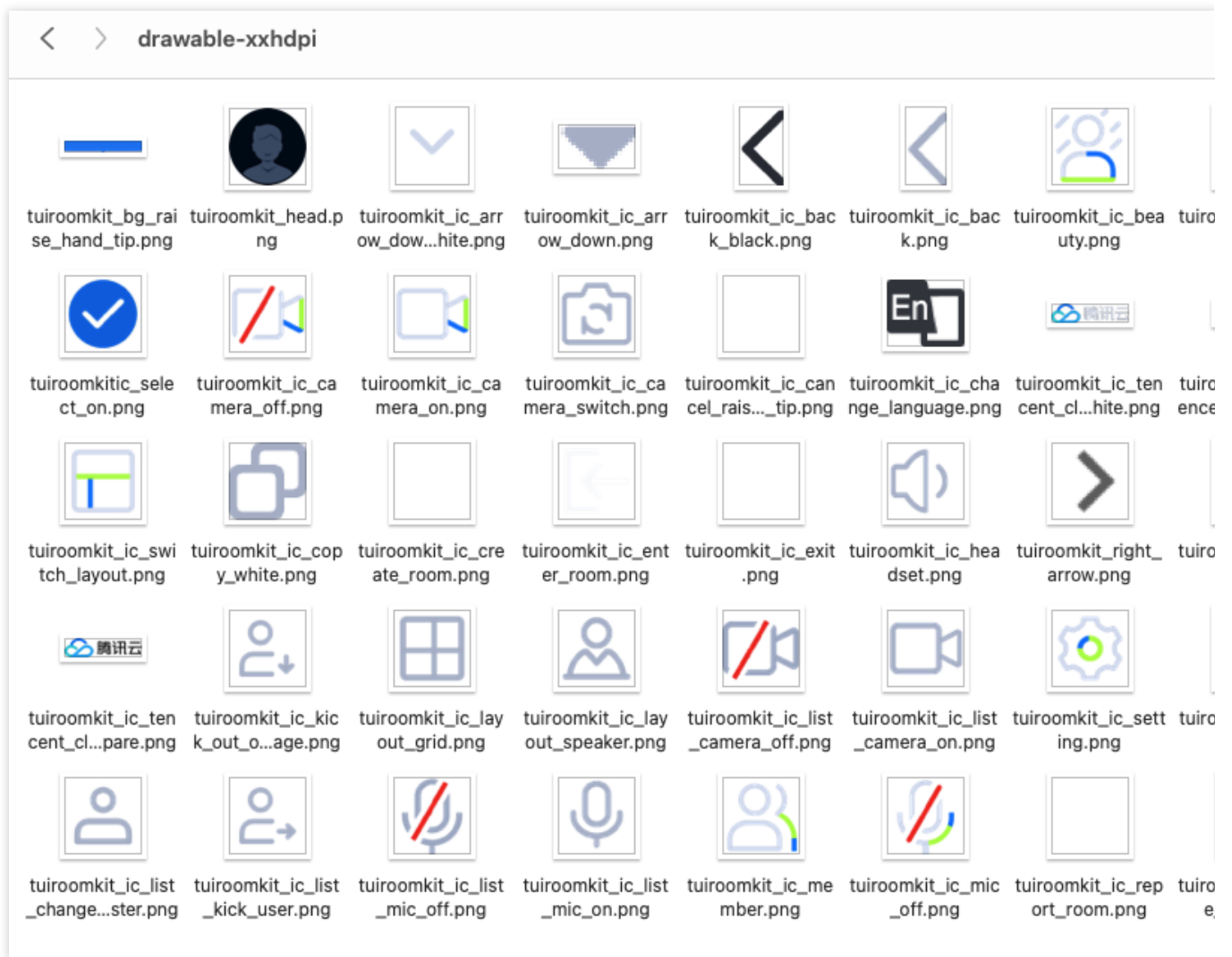
## Solution 1: Fine-tuning Solution

By directly modifying the UI source code we provide, you can adjust the User Interface of TUIRoomKit. The source code of TUIRoomKit's interface is located in the `Android/tuiroomkit` folder on Github:

### Replace Icon

You can directly replace the icons in the `src/res/drawable-xxhdpi` folder to ensure that the color tone and style of the icons in the entire App remain consistent. Please keep the name of the icon file unchanged when replacing.





## Replace Copywriting

You can modify the string content of the video conference interface by modifying the `strings.xml` files in `values-zh` and `values-en`.

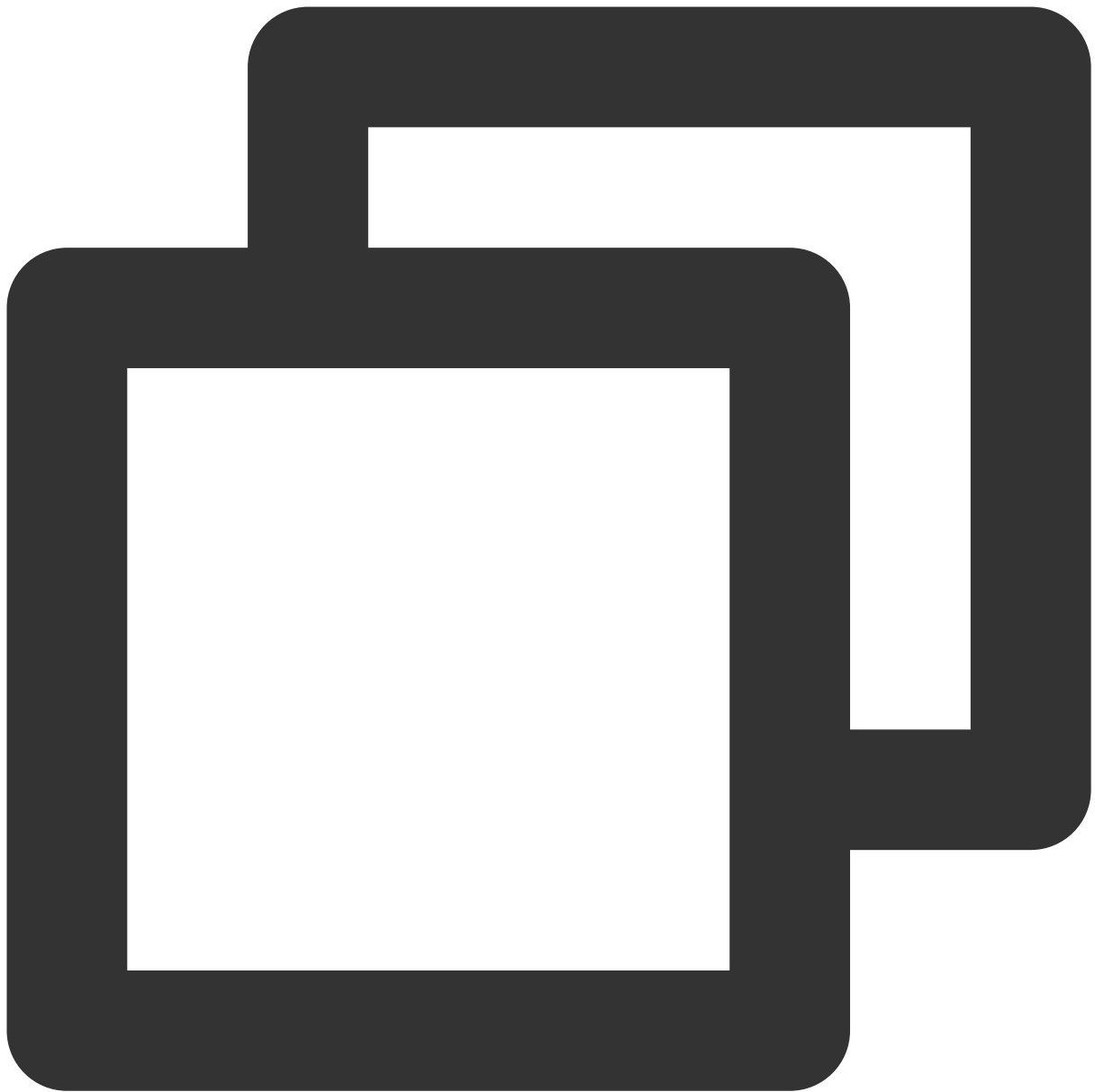
## Solution 2: Custom Partial UI Solution

The UI code of `TUIRoomKit` is located in the

`src/main/java/com/tencent/cloud/tuikit/roomkit/view` directory, and the screen view is in the

`TUIVideoSeat` Component.

The key files of `TUIRoomKit`'s View are as follows. You can change the corresponding view according to your needs and adjust your UI.

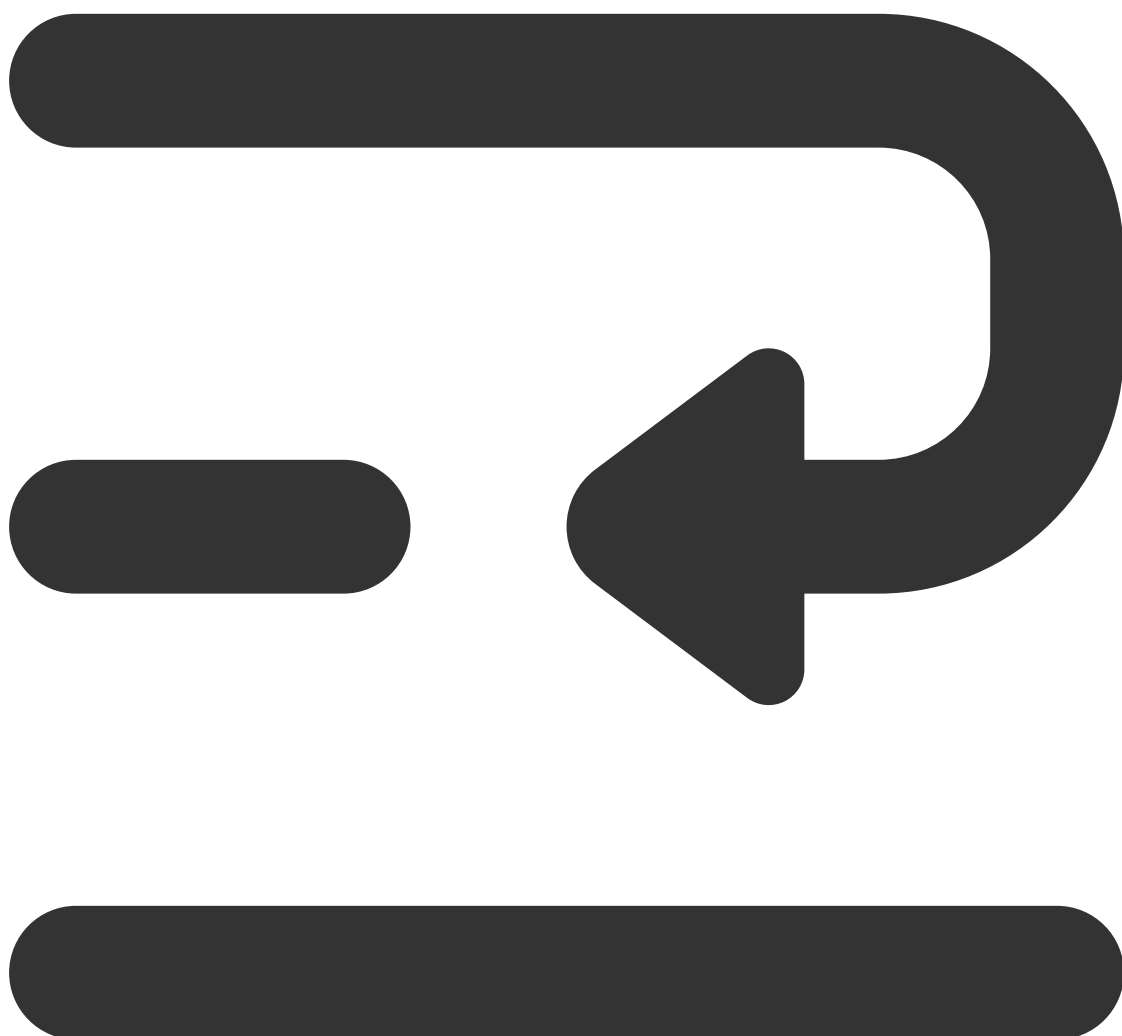


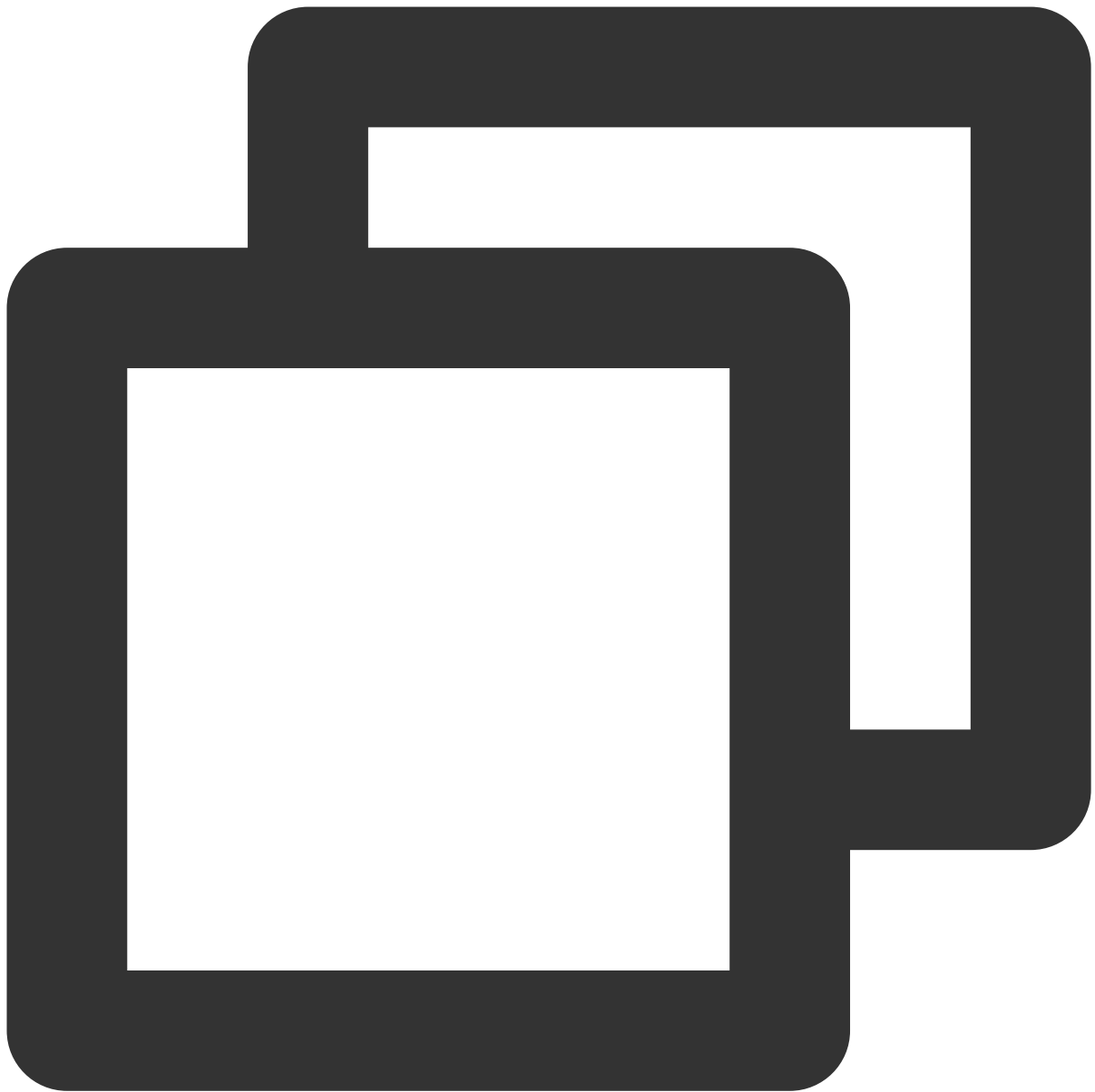
```
        |— PrepareActivity    // Preparation page Activity
        |
        |— CreateRoomActivity // Create room Activity
        |
        |— EnterRoomActivity  // Enter room Activity
        |
activity —|— RoomMainActivity // Room main page Activity
        |
view —|
        |
component—|— PrepareView      // Preparation page View
```

```
|
|— CreateRoomView      // Create room View
|
|— EnterRoomView       // Enter room View
|
|— RoomMainView        // Audio/video conference main page View
|
|— TopView             // Top button view, including: Speaker/Receiv
|
|— BottomView          // Bottom button view, including: Camera, Mic
|
|— UseListView         // User List view
|
|— RaiseHandApplicationListView // Raise Hand to Speak mode, R
|
|— TransferMasterView // Room Owner Transfer page
|
|— MoreFunctionView    // "More" function view, including Chat, Beau
|
|— MeetingActivity     // Audio/video conference main activity
```

## Modification of BottomView's Bottom Button

To make it easier for you to customize the bottom function buttons, our BottomView is automatically constructed by reading the list. Taking the video switch button as an example, the code is as follows.





```
private BottomItemData createCameraItem() {
    BottomItemData cameraItemData = new BottomItemData();
    //Set button type to differentiate different buttons
    cameraItemData.setType(BottomItemData.Type.VIDEO);
    //Set whether the button is clickable
    if (isOwner()) {
        cameraItemData.setEnable(true);
    } else if (mRoomStore.roomInfo.enableSeatControl) {
        cameraItemData.setEnable(false);
    } else {
        cameraItemData.setEnable(mRoomStore.roomInfo.enableVideo);
    }
}
```

```

}
//Set the default icon of the button
cameraItemData.setIconId(R.drawable.tuiroomkit_ic_camera_off);
//Set the background image of the button
cameraItemData.setBackground(R.drawable.tuiroomkit_bg_bottom_item_black);
//Set the icon of the button when it is not clickable
cameraItemData.setDisableIconId(R.drawable.tuiroomkit_ic_camera_off);
//Set the default icon of the button
cameraItemData.setName(mContext.getString(R.string.tuiroomkit_item_open_camera))

//Button click effect, if your button needs to switch images/names, etc. when cl
BottomSelectItemData camaraSelectItemData = new BottomSelectItemData();
//Set the name of the button when selected
camaraSelectItemData.setSelectedName(mContext.getString(R.string.tuiroomkit_item
//Set the name of the button when not selected
camaraSelectItemData.setUnSelectedName(mContext.getString(R.string.tuiroomkit_it
//Set whether the button is selected
camaraSelectItemData.setSelected(false);
//Set the icon of the button when selected
camaraSelectItemData.setSelectedIconId(R.drawable.tuiroomkit_ic_camera_on);
//Set the icon of the button when not selected
camaraSelectItemData.setUnSelectedIconId(R.drawable.tuiroomkit_ic_camera_off);
//Set the click event of the button when selected/unselected
camaraSelectItemData.setOnItemSelectedListener(new BottomSelectItemData.OnItemSele
    @Override
    public void onItemSelected(boolean isSelected) {
        enableCamera(isSelected);
    }
});
cameraItemData.setSelectItemData(camaraSelectItemData);
return cameraItemData;
}

```

## Solution 3: Custom All UI Solution

The overall function of TUIRoomKit is based on the TUIRoomEngine, a UI-less SDK. You can completely implement your own UI interface based on TUIRoomEngine. For details, please refer to the TUIRoomEngine API interface address.

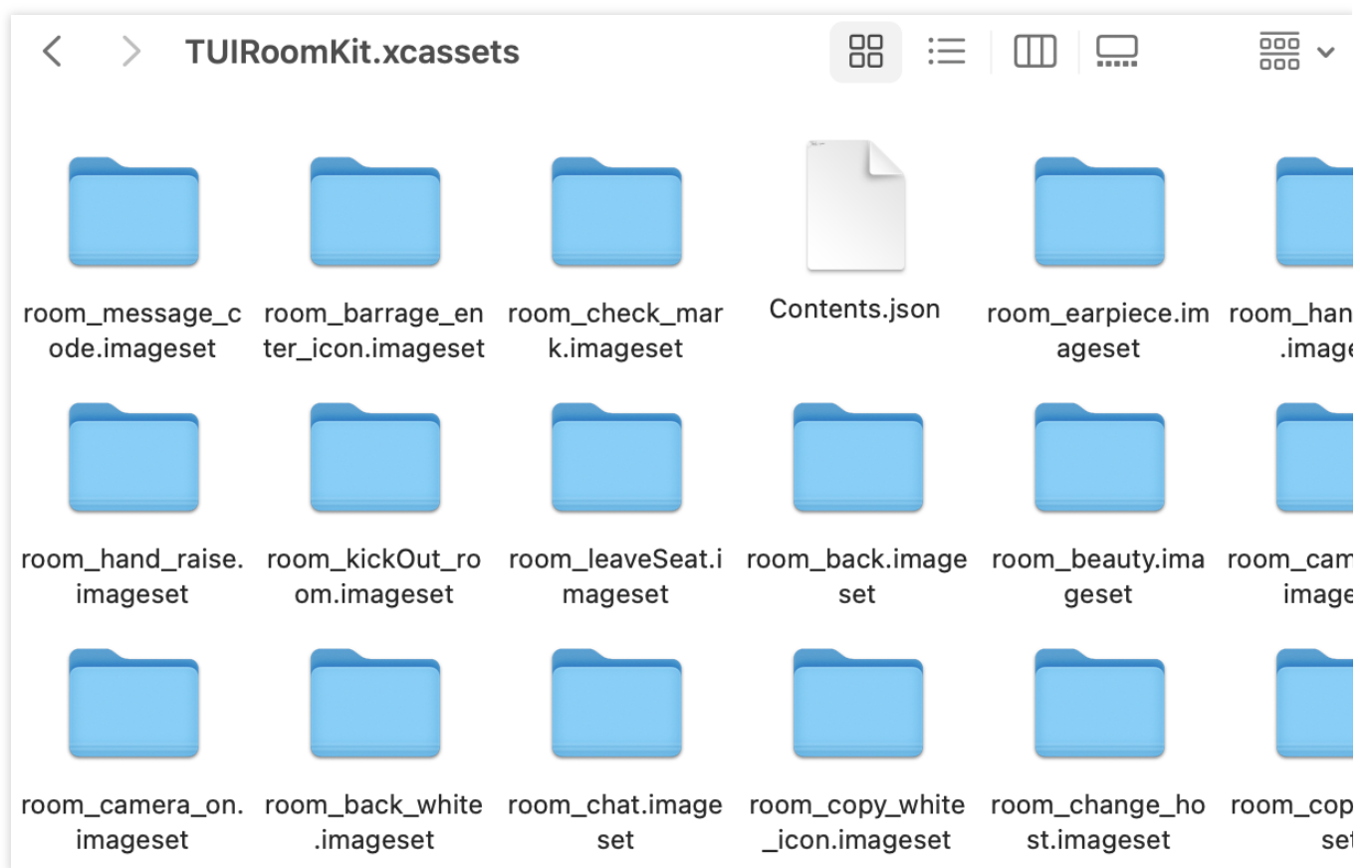
# iOS

Last updated : 2024-05-30 13:08:27

This article will introduce how to customize the user interface of TUIRoomKit. By reading this article, you will understand various schemes for UI customization in TUIRoomKit to meet your specific application needs. Through these schemes, you can flexibly adjust and optimize UI elements to better fit your requirements.

## Replace Icon

You can directly modify the Icon Components in the **TUIRoomKit/Resources/TUIRoomKit.xcassets** folder to ensure a consistent color scheme for icons throughout the App. Please keep the icon file names unchanged when replacing.



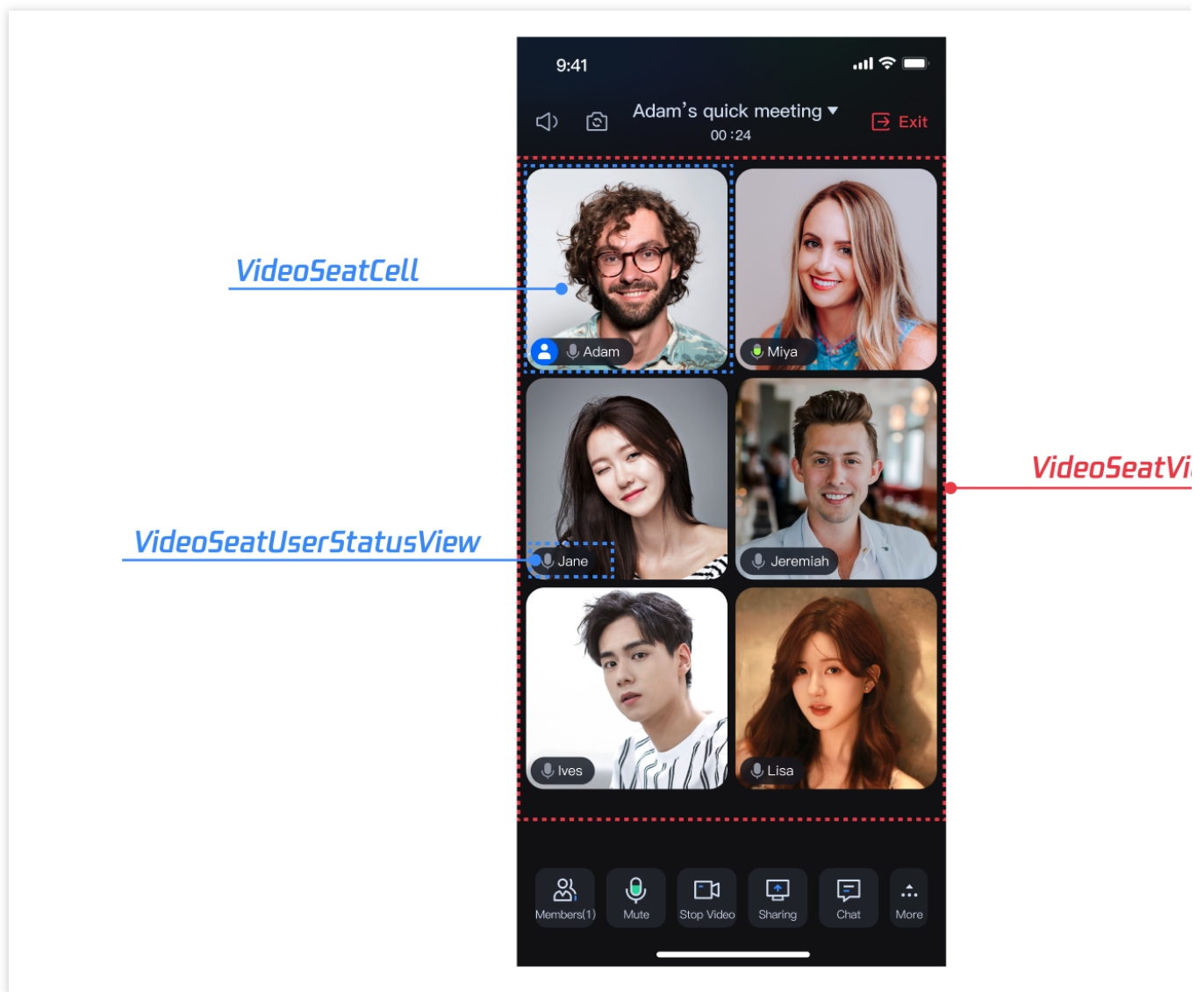
## Replace Copywriting

You can modify the string content of the video conferencing interface by modifying the

`TUIRoomKitLocalized.xcstrings` file in the `TUIRoomKit/Resources/Localized` folder.

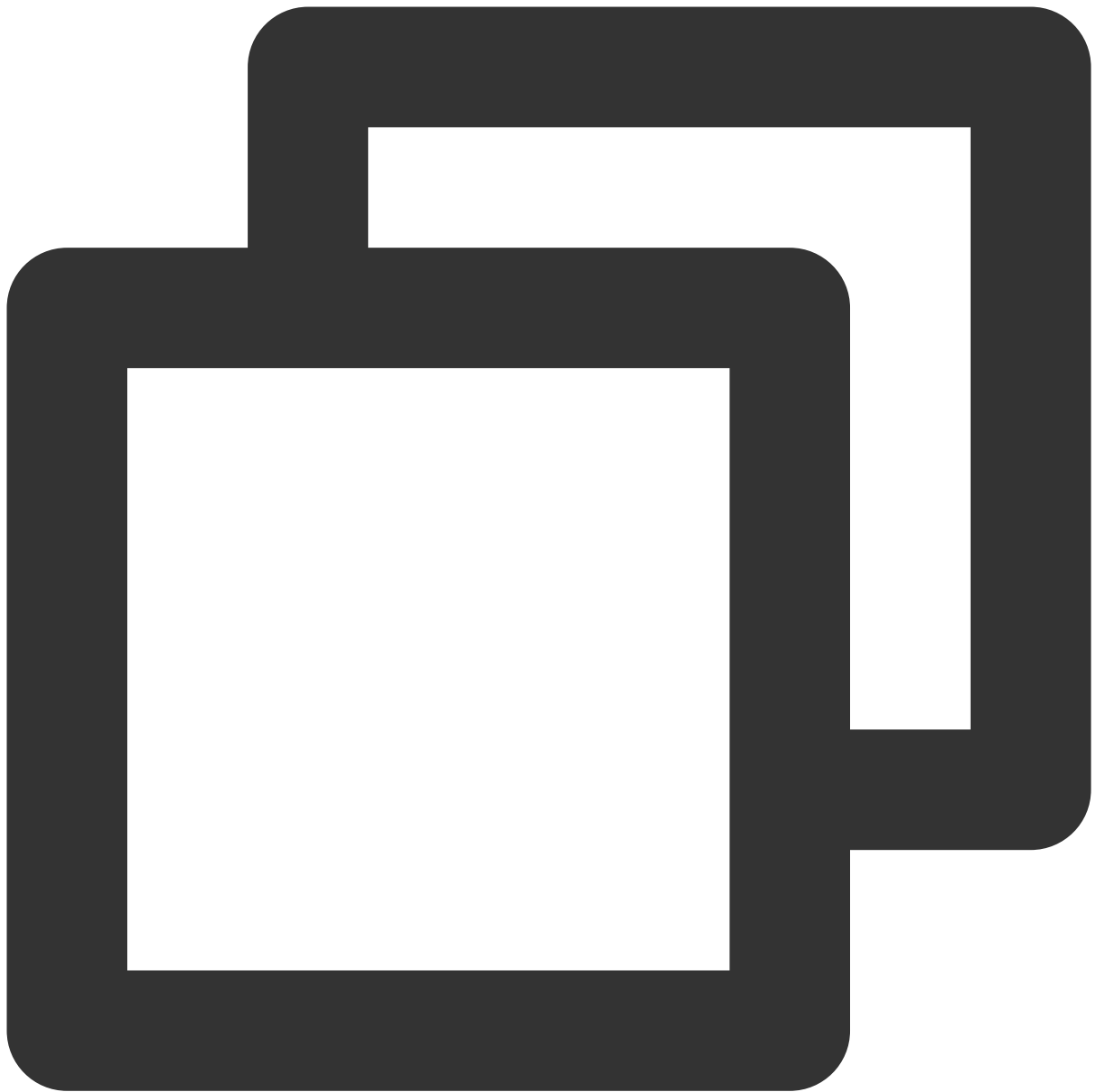
## Adjust Video Window Layout

In the Meeting Main Interface, the classes related to video footage are as shown:



The file directory structure for classes related to video footage is as follows. You can adjust the video footage by modifying the contents in the `TUIRoomKit/Source/View` file.

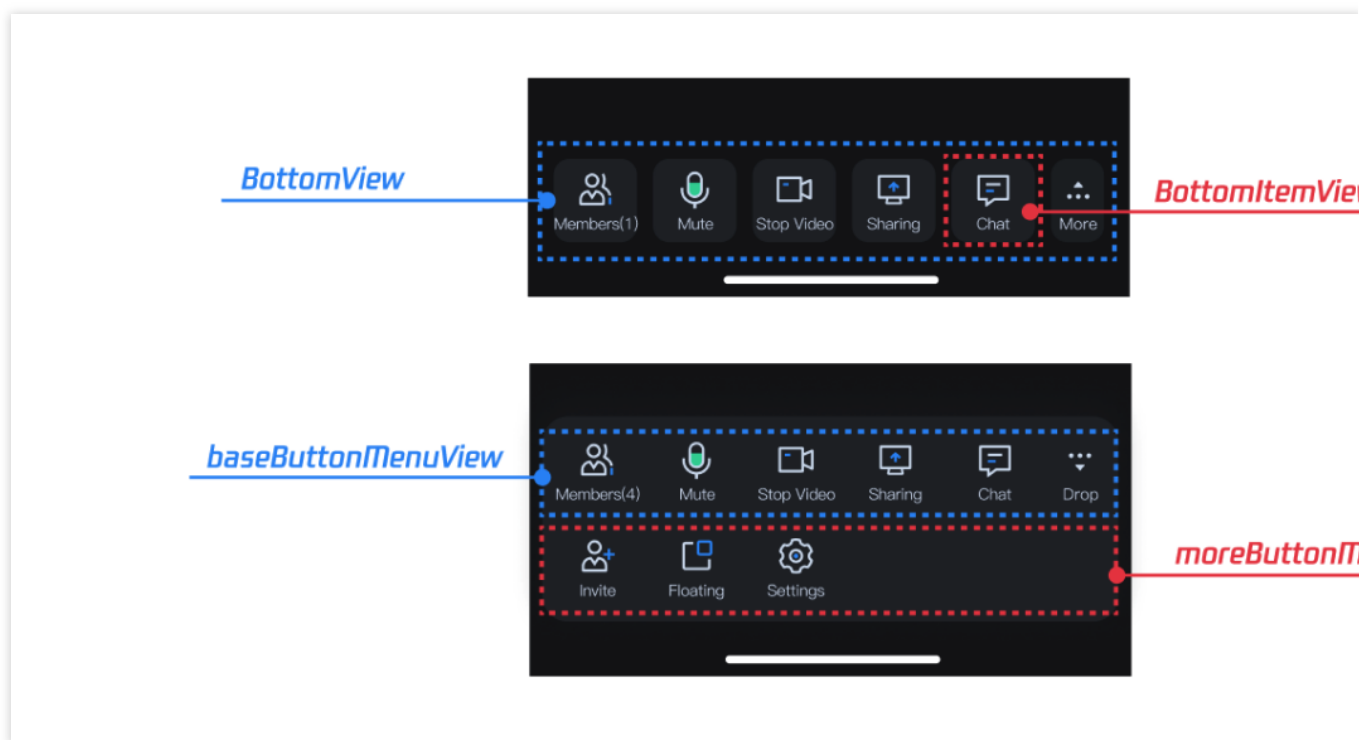




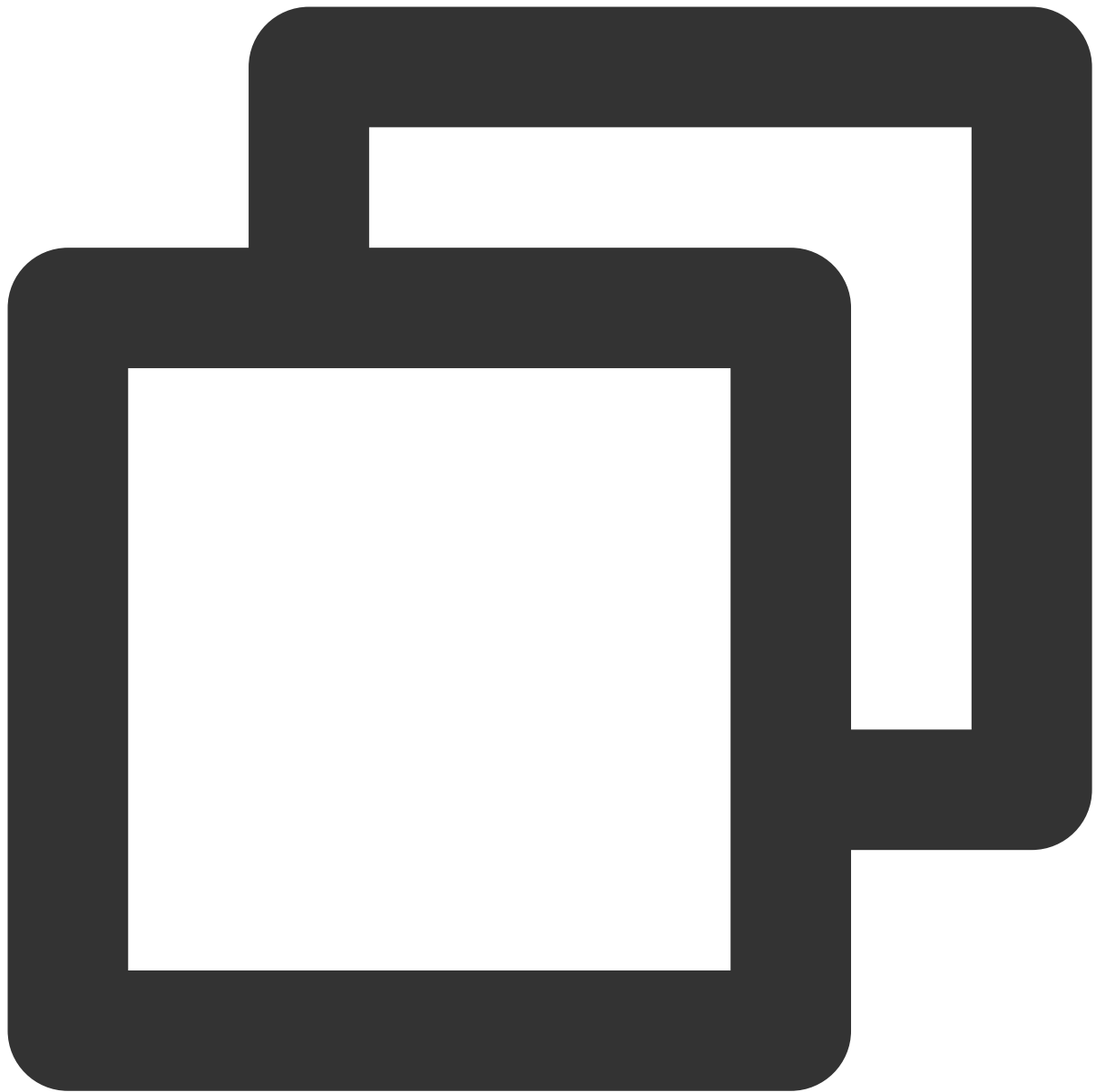
```
View
├── Page
│   ├── ConferenceMainView.swift                // Meeting Main View
│   └── Widget
│       ├── VideoSeat
│       │   ├── ScreenCaptureMaskView.swift    // Panel displayed duri
│       │   ├── VideoSeatCell.swift            // Video Image Cell
│       │   ├── VideoSeatLayout.swift          // Video Screen Layout
│       │   ├── VideoSeatUserStatusView.swift  // User Information Bel
│       └── VideoSeatView.swift                // Overall Video Screen
```

## Adjust Bottom Toolbar

In the bottom toolbar of the meeting interface, there are various buttons related to the meeting. Classes or objects related to the bottom bar and its UI are as shown:



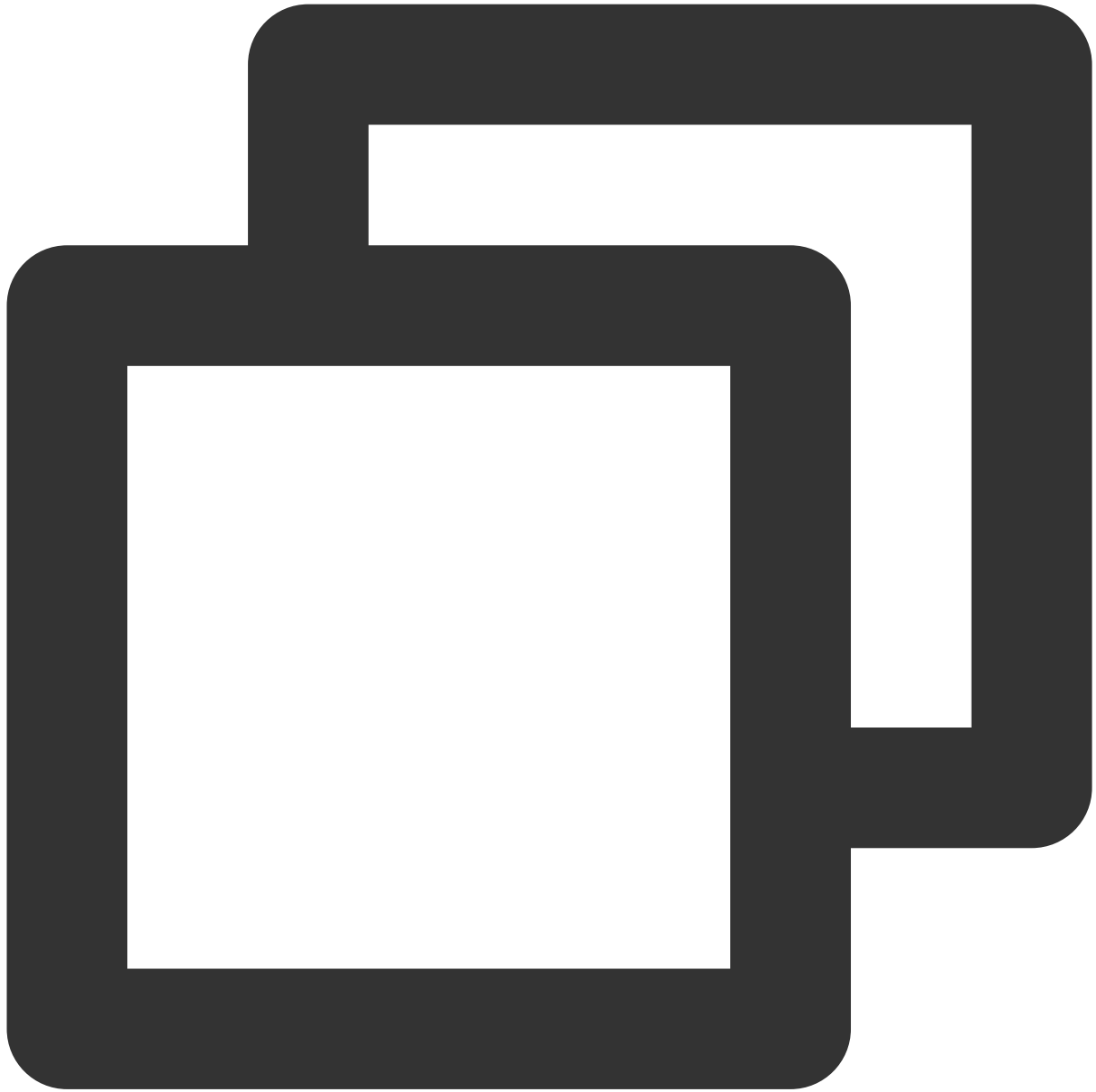
The file directory structure of classes related to the bottom toolbar is as follows. You can adjust the bottom bar by modifying the content in the `TUIRoomKit/Source/View` file.



```
View
├── Page
│   └── Widget
│       └── BottomNavigationBar
│           ├── BottomItemView.swift           // Bottom Bar Universal
│           └── BottomView.swift               // Bottom Toolbar
```

## Modification of Bottom Toolbar Button

To facilitate your custom Definition of bottom feature buttons, our BottomView automatically constructs by reading a list. For example, to switch the video button, the code is as follows. You can modify the content of the button to achieve your desired requirements.

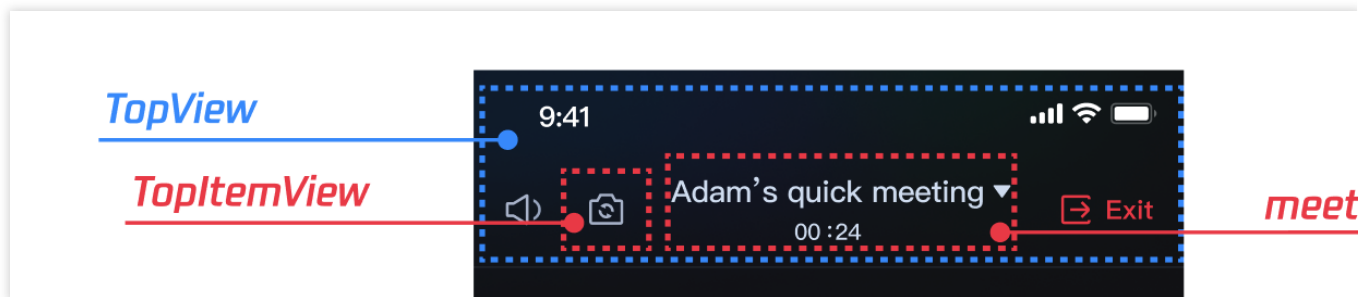


```
func createBottomData() {  
    let muteVideoItem = ButtonItemData()  
    // Set the default button title  
    muteVideoItem.normalTitle = .unMuteVideoText  
    // Set the button title after clicking  
    muteVideoItem.selectedTitle = .muteVideoText  
    // Set the default button icon
```

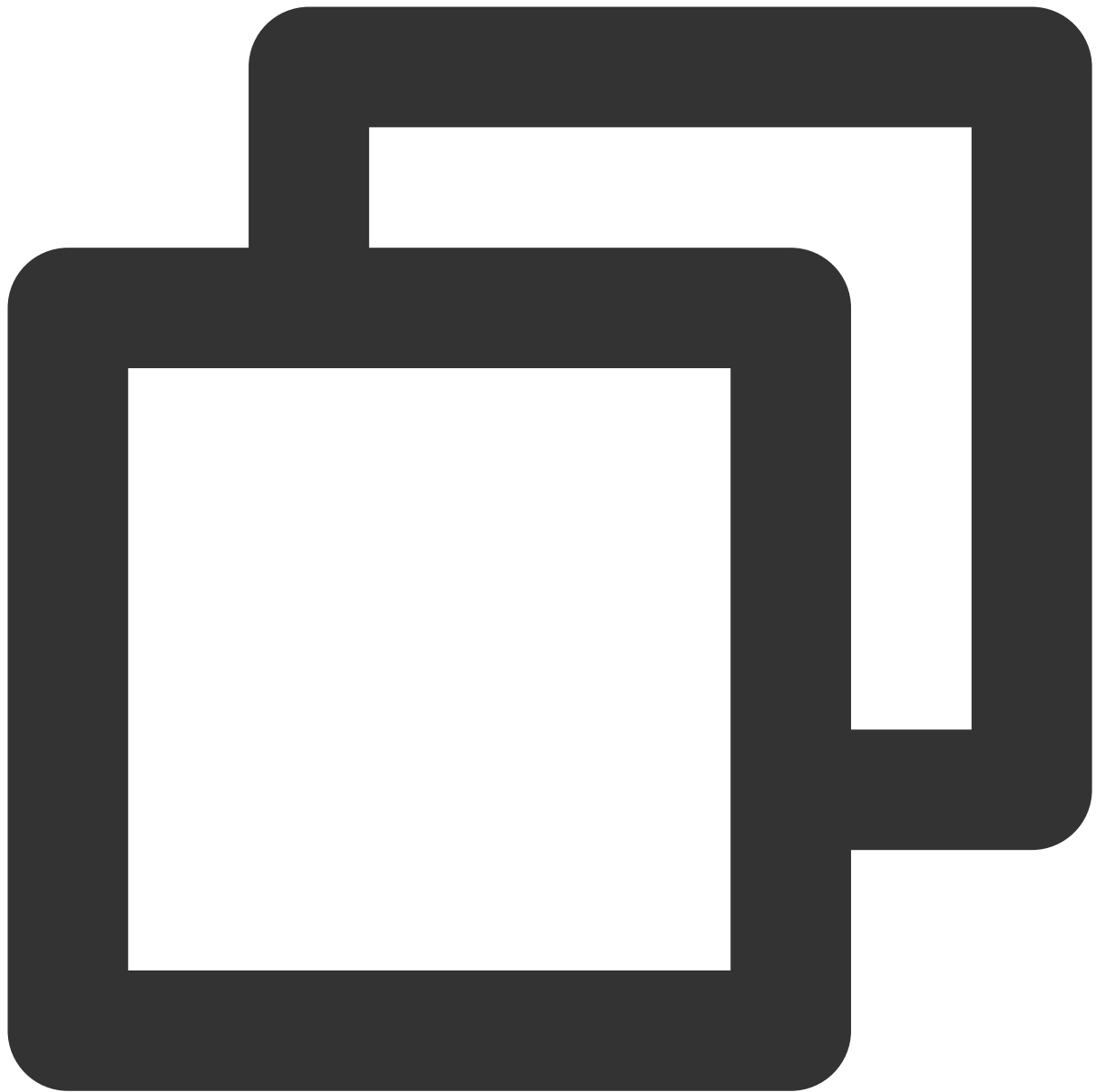
```
muteVideoItem.normalIcon = "room_camera_on"
// Set the button icon after clicking
muteVideoItem.selectedIcon = "room_camera_off"
// Set the button background color
muteVideoItem.backgroundColor = UIColor(0xA3AEC7)
// Set Button Image Resource Acquisition Location
muteVideoItem.resourceBundle = tuiRoomKitBundle()
// Set Whether the Button is Clickable
muteVideoItem.isSelect = !(roomInfo.isOpenCamera)
// Set Button Type to Distinguish Different Buttons
muteVideoItem.buttonType = .muteVideoItemType
// Set Button Click Event
muteVideoItem.action = { [weak self] sender in
    guard let self = self, let button = sender as? UIButton else { return }
    self.muteVideoAction(sender: button)
}
}
```

## Adjust Top Toolbar

In the meeting main interface, classes or objects related to the top toolbar are as shown:



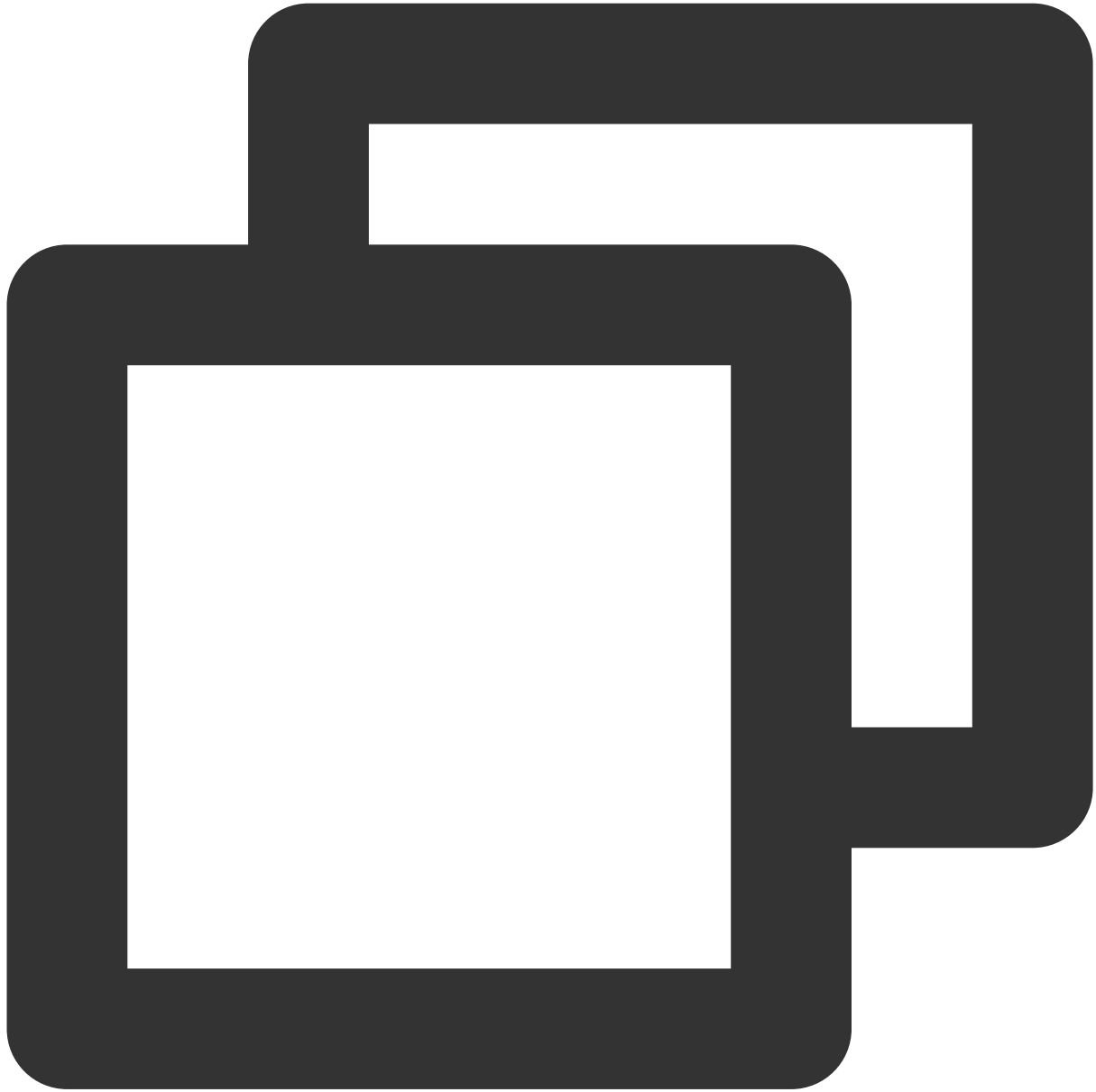
The file directory structure of classes related to the top toolbar is as follows. You can adjust the top bar by modifying the content in the `TUIRoomKit/Source/View` file.



```
View
├── Page
│   └── Widget
│       ├── TopNavigationBar
│       │   ├── TopItemView.swift // Top Bar Universal Bu
│       │   └── TopView.swift      // Top Toolbar
```

## Adjust Other UI

When you need to adjust other UI elements, you can refer to the directory structure of other UIs under the `TUIRoomKit/Source/View` file. In the directory structure below, each file's corresponding UI has been marked. You can modify parts of the UI you wish to change according to your needs.



```
View
├── Component
├── Page
│   ├── ConferenceMainView.swift // Meeting Main Page
│   └── Widget
│       ├── Dialog
│       └── ExitRoomView.swift // Exit Room Popup
```

```

|   |─ MemberInviteView.swift           // Invite Member Popup
|   |─ RaiseHandNoticeView.swift        // Raise Hand Notificat
|   └─ RoomInfoView.swift              // Room Information Pop
└─ FloatWindow
|   |─ RoomUserStatusView.swift         // Floating Window User
|   └─ RoomVideoFloatView.swift        // Floating Window
└─ LocalAudioIndicator
|   └─ LocalAudioView.swift            // Bottom Microphone Bu
└─ MediaSettings
|   |─ MediaSettingView.swift           // Settings Interface
|   |─ QualityInfoPanel.swift           // Quality Inspection P
|   └─ VideoChoicePanel.swift           // Video Settings Panel
└─ PopUpControlPanel
|   └─ PopUpView.swift                 // General Bottom Popup
└─ RaiseHandControlPanel
|   |─ RaiseHandApplicationCell.swift   // Stage Application Li
|   |─ RaiseHandApplicationListView.swift // Stage Application Li
|   └─ RaiseHandApplicationNotificationView.swift // Stage Application No
└─ TransferOwnerControlPanel
|   └─ TransferMasterView.swift         // Transfer Master Pane
└─ UserControlPanel
|   |─ UserListCell.swift               // User List Member Cel
|   |─ UserListManagerView.swift        // Manage User Panel
|   └─ UserListView.swift               // User List Panel
└─ WaterMark
|   |─ FeatureSwitch.swift              // Watermark Toggle
|   |─ WaterMarkLayer.swift             // Watermark View
|   └─ WaterMarkLineStyle.swift         // Watermark Text Style

```

## Custom Definition UI Scheme

The overall feature of TUIRoomKit is based on the UI-less SDK, TUIRoomEngine. You can fully implement your own UI interface based on TUIRoomEngine. For more details, see [TUIRoomEngine API](#).



# Web

Last updated : 2024-07-17 16:33:10

This article will provide a detailed introduction on how to customize the user interface of TUIRoomkit. TUIRoomkit offers two customization methods: one is through a simple custom UI API, and the other is by replacing existing UI components. We will introduce each method in detail below.

## Method 1: Fine-tuning the interface

TUIRoomkit provides a series of [APIs](#) that make it easy to customize the UI. The table below lists some of the main APIs and their functions:

API	Description
<a href="#">setLanguage</a>	Set the interface language.
<a href="#">setTheme</a>	Set the interface theme.
<a href="#">enableWatermark</a>	Enable the text messaging feature in the application. For details, see: <a href="#">Text Watermark</a> .
<a href="#">enableVirtualBackground</a>	Enable the virtual background feature in the application. After calling this function, a virtual background feature button will be displayed on the UI. For details, see: <a href="#">Virtual Background</a> .
<a href="#">disableTextMessaging</a>	Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.
<a href="#">disableScreenSharing</a>	Disable the screen sharing feature in the application. After calling this function, users will not be able to share their screen with others.
<a href="#">hideFeatureButton</a>	Hide specific feature buttons in the application. After calling this function and passing in the appropriate <a href="#">FeatureButton</a> enumeration value, the corresponding button will be hidden from the user interface.

## Method 2: Modify the UIKit source code

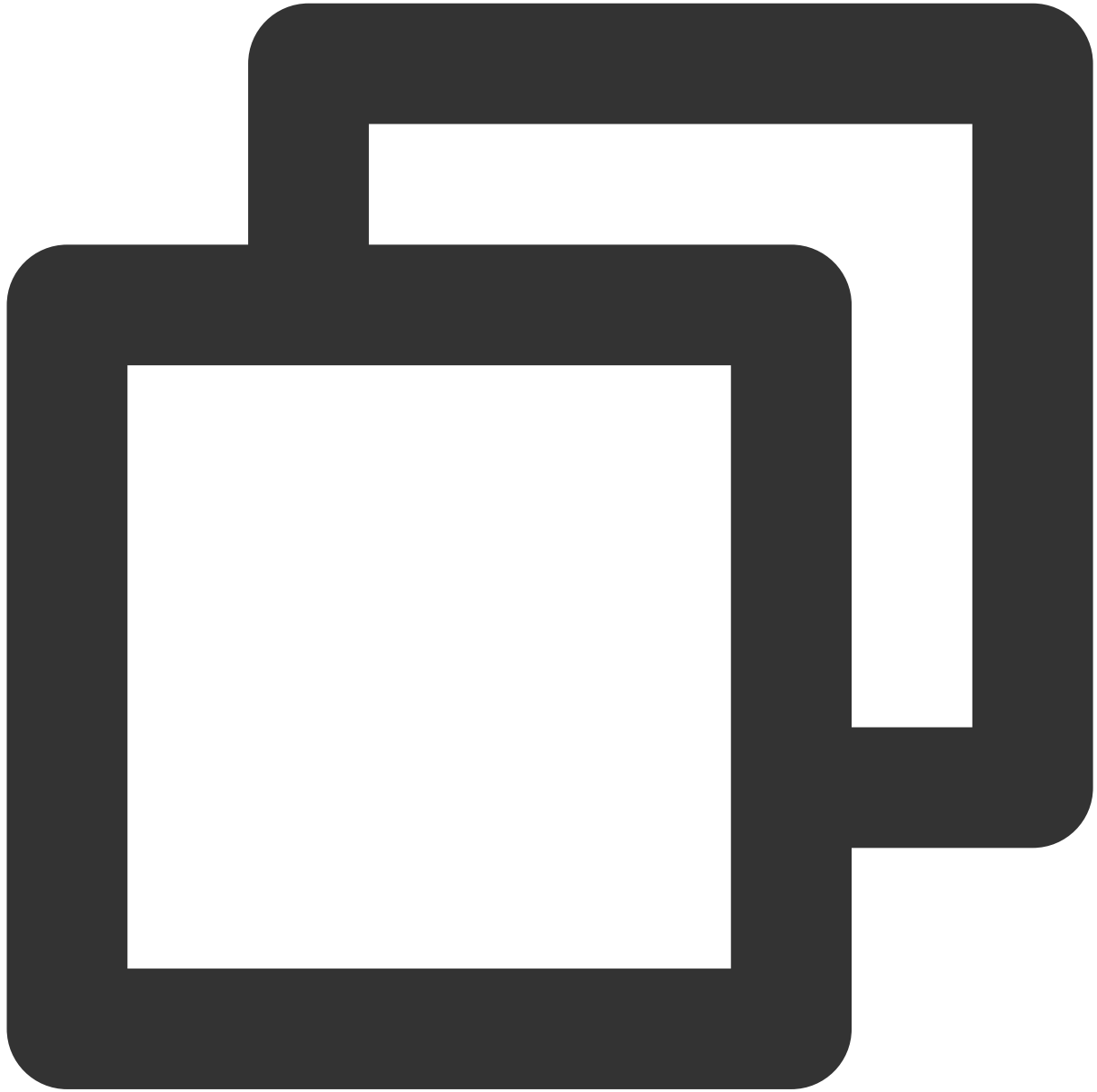
You can directly modify the provided UI source code to adjust the user interface of TUIRoomKit according to your requirements.

### Step 1: Exporting UIKit Source Code

Vue3

Vue2

1. Execute the source code export script, the default copy path is `./src/components/TUIRoom`

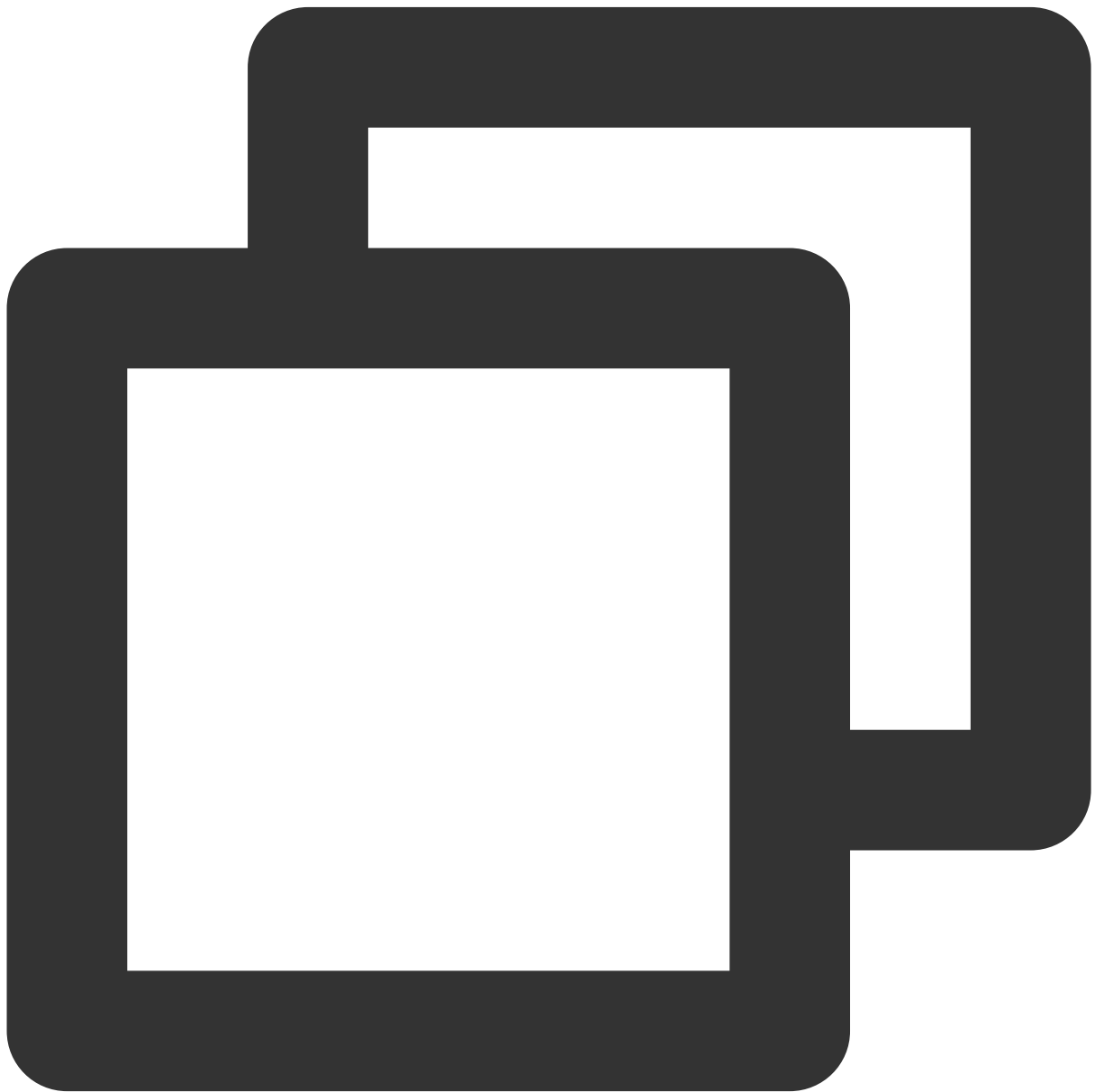


```
node ./node_modules/@tencentcloud/roomkit-web-vue3/scripts/eject.js
```

2. Follow the script prompts to confirm that you want to copy the TUIRoomKit source code to the `./src/components/TUIRoom` directory. Enter 'y' if you want to customize the copy directory, otherwise enter 'n'.

```
● → tuiroomkit-vue3-webpack-ts git:(main) x node ./node_modules/@tencentcloud/roomkit-web-vue3/s
This script copies the TUIRoomKit source code to the ./src/components/TUIRoom path, do you want
Please enter your copy path: ./src/TUIRoomKit
[SUCCESS] The TUIRoomKit source code has been successfully copied to the ./src/TUIRoomKit pat
○ → tuiroomkit-vue3-webpack-ts git:(main) x █
```

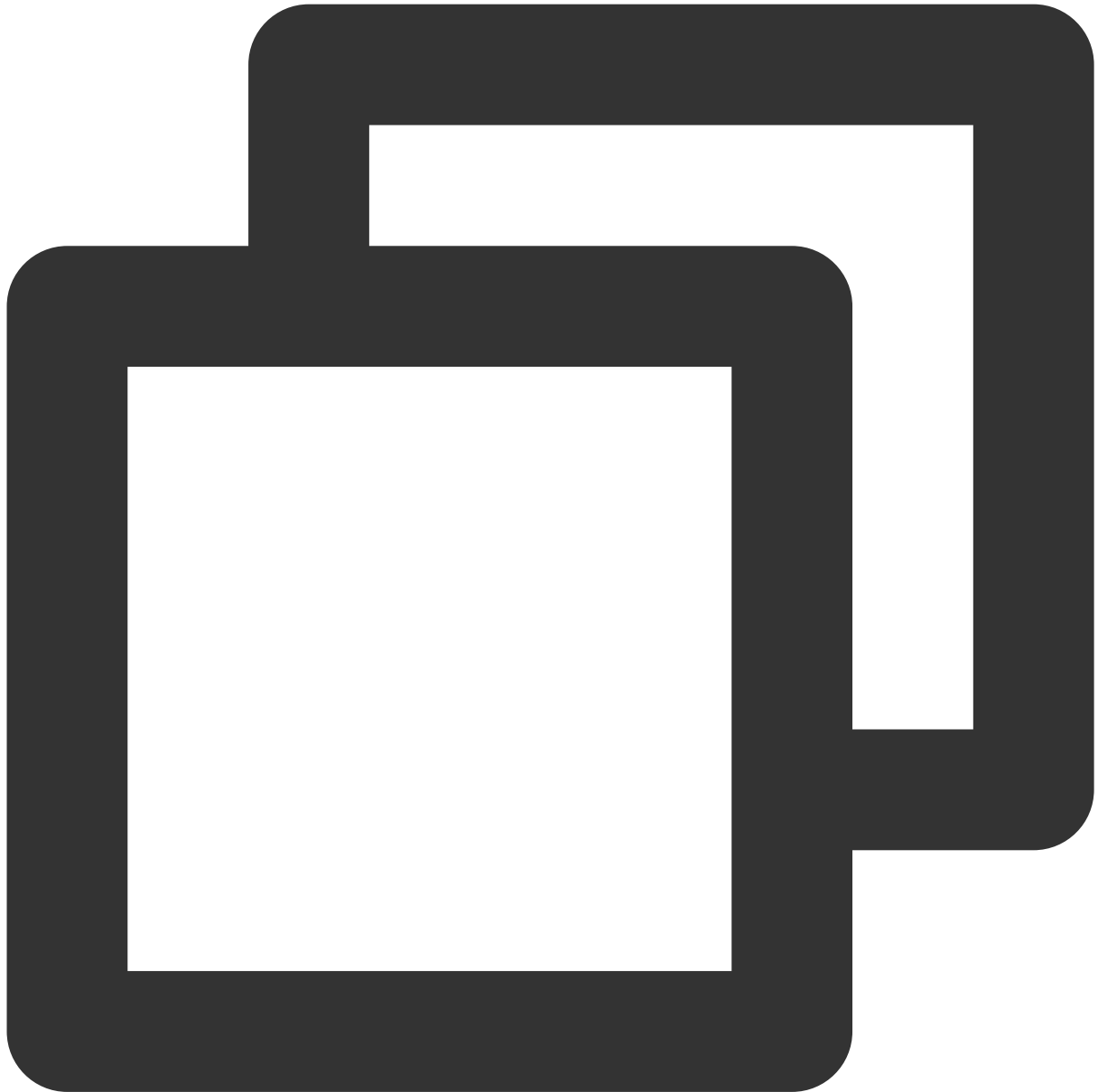
3. After exporting the source code, the TUIRoomKit source code will be added to the project path you specified. At this point, you need to manually change the reference to the ConferenceMainView component, conference object from the npm package address to the relative path address of the TUIRoom source code.



```
- import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue3';
```

```
// Replace the reference path with the real path of the TUIRoomKit source code.  
+ import { ConferenceMainView, conference } from './src/components/TUIRoom/index.ts'
```

1. Execute the source code export script, the default copy path is `./src/components/TUIRoom`

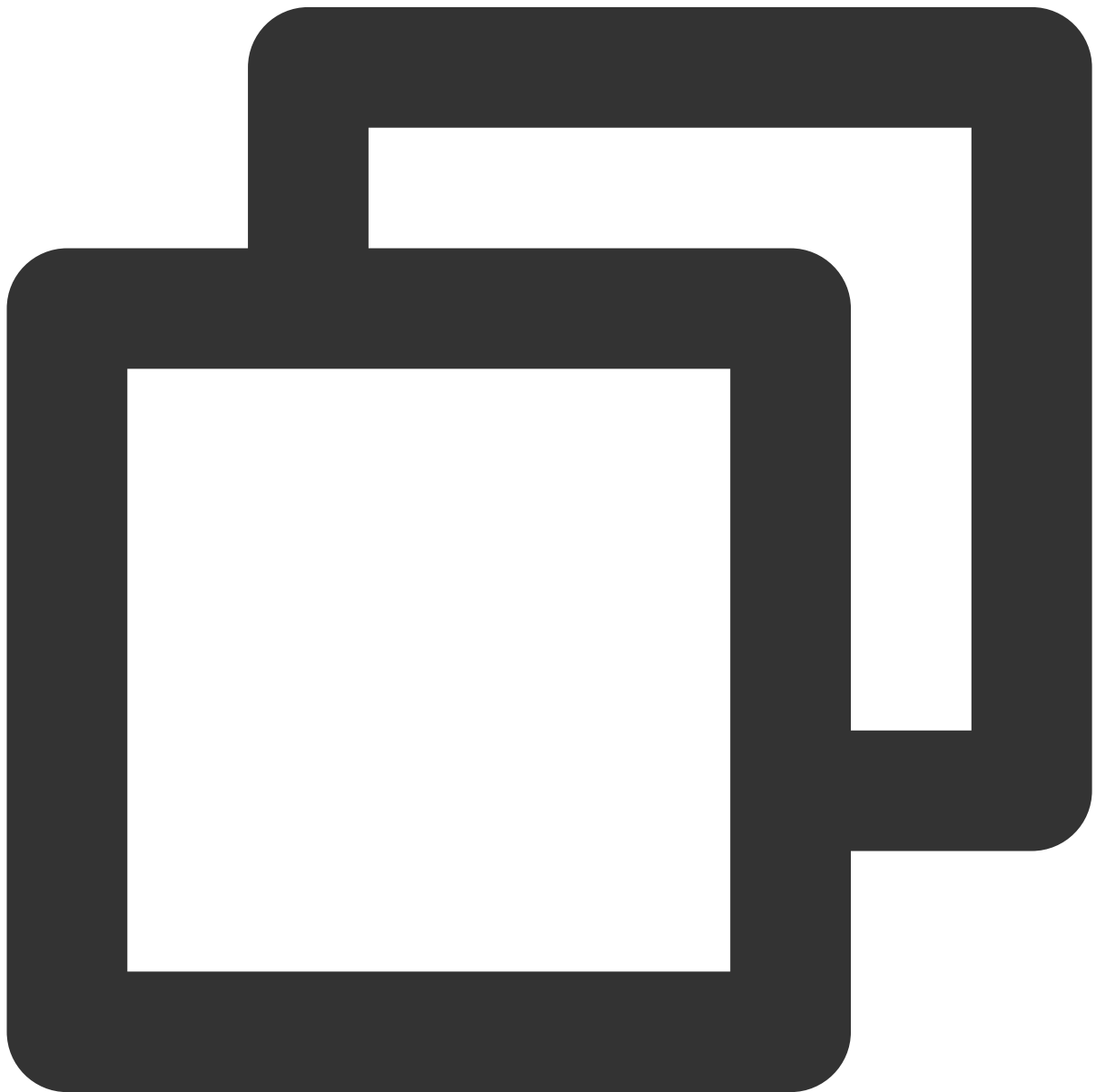


```
node ./node_modules/@tencentcloud/roomkit-web-vue2.7/scripts/eject.js
```

2. Follow the script prompts to confirm that you want to copy the TUIRoomKit source code to the `./src/components/TUIRoom` directory. Enter 'y' if you want to customize the copy directory, otherwise enter 'n'.

```
• → tuiroomkit-vue2.7-js git:(main) x node ./node_modules/@tencentcloud/roomkit-web-vue2.7/
This script copies the TUIRoomKit source code to the ./src/components/TUIRoom path, do you
Please enter your copy path: ./src/TUIRoomKit
[SUCCESS] The TUIRoomKit source code has been successfully copied to the ./src/TUIRoomK
○ → tuiroomkit-vue2.7-js git:(main) x
```

3. After exporting the source code, the TUIRoomKit source code will be added to the project path you specified. At this point, you need to manually change the reference to the ConferenceMainView component, conference object from the npm package address to the relative path address of the TUIRoom source code.



```
- import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue2.7'  
// Replace the reference path with the real path of the TUIRoomKit source code.  
+ import { ConferenceMainView, conference } from './src/components/TUIRoom/index.ts'
```

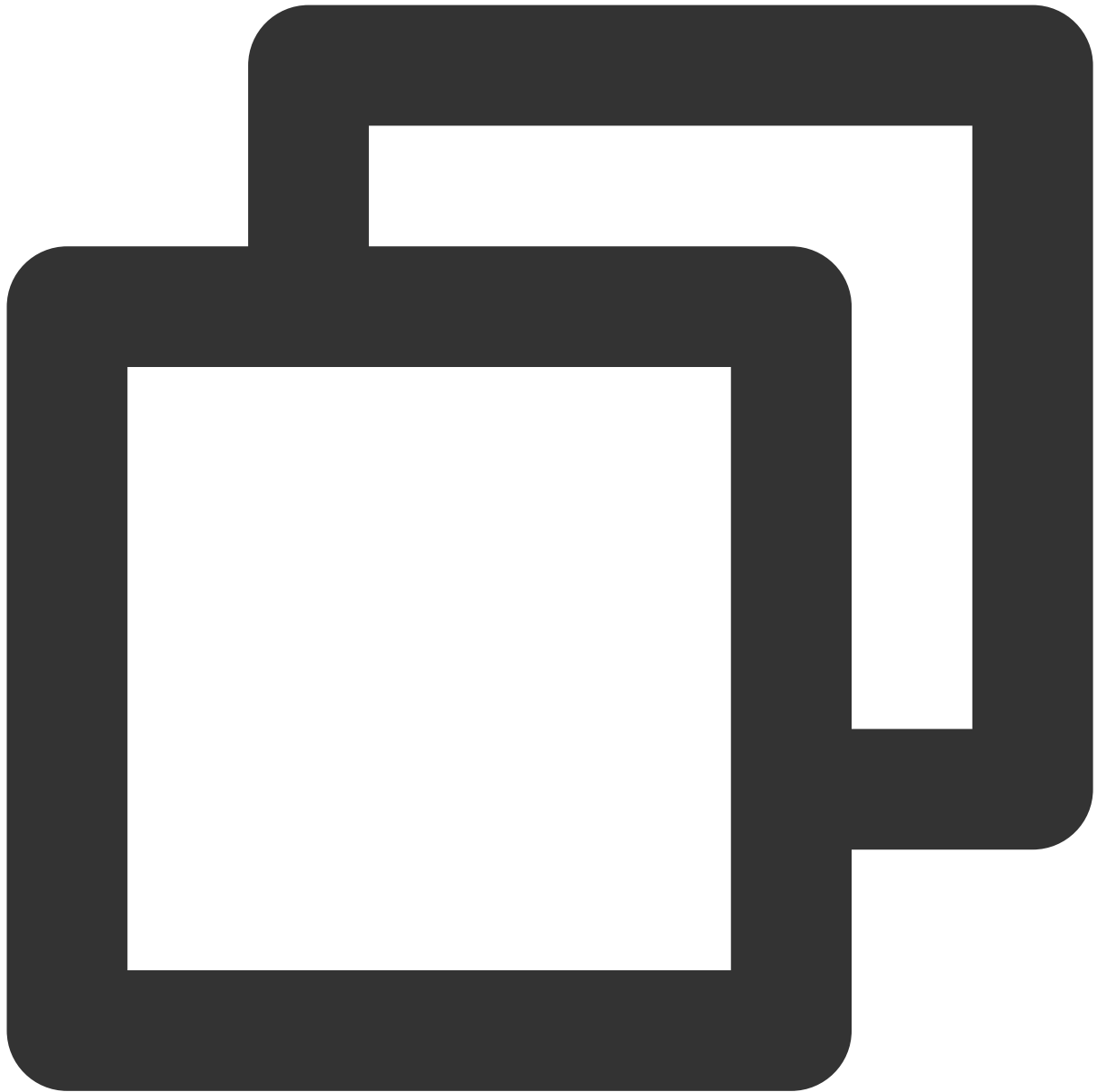
## Step 2 : Configuring the UIKIT source code development environment

Config for Vue3 + Vite + TS

Config for Vue3 + Webpack + TS

Config for Vue2 + Webpack + TS

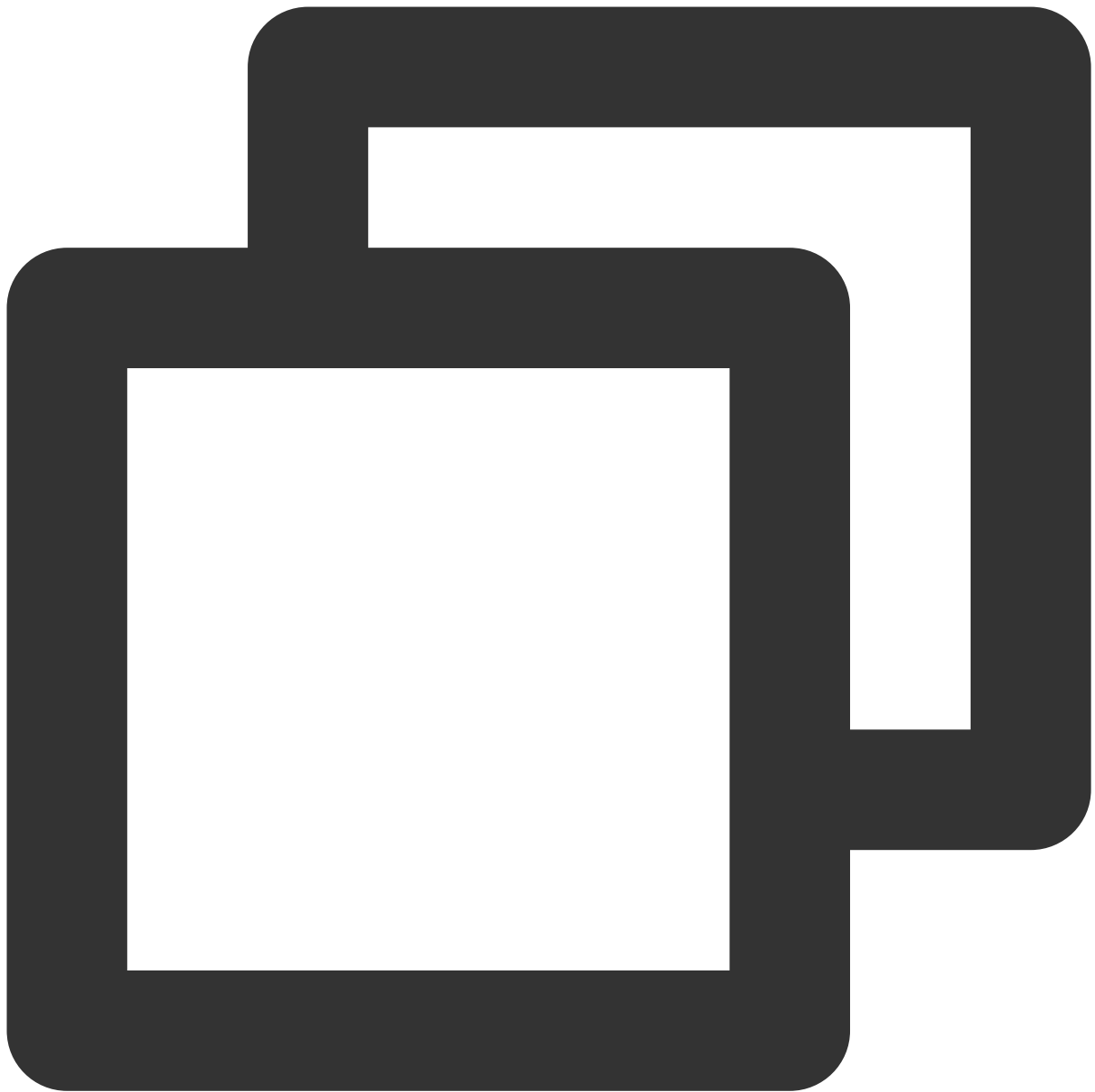
### 1. Install development environment dependencies



```
npm install typescript -S -D
```

## 2. Register for Pinia

TUIRoom uses Pinia for room data management, you need to register Pinia in the project entry file, which is the `src/main.ts` file.



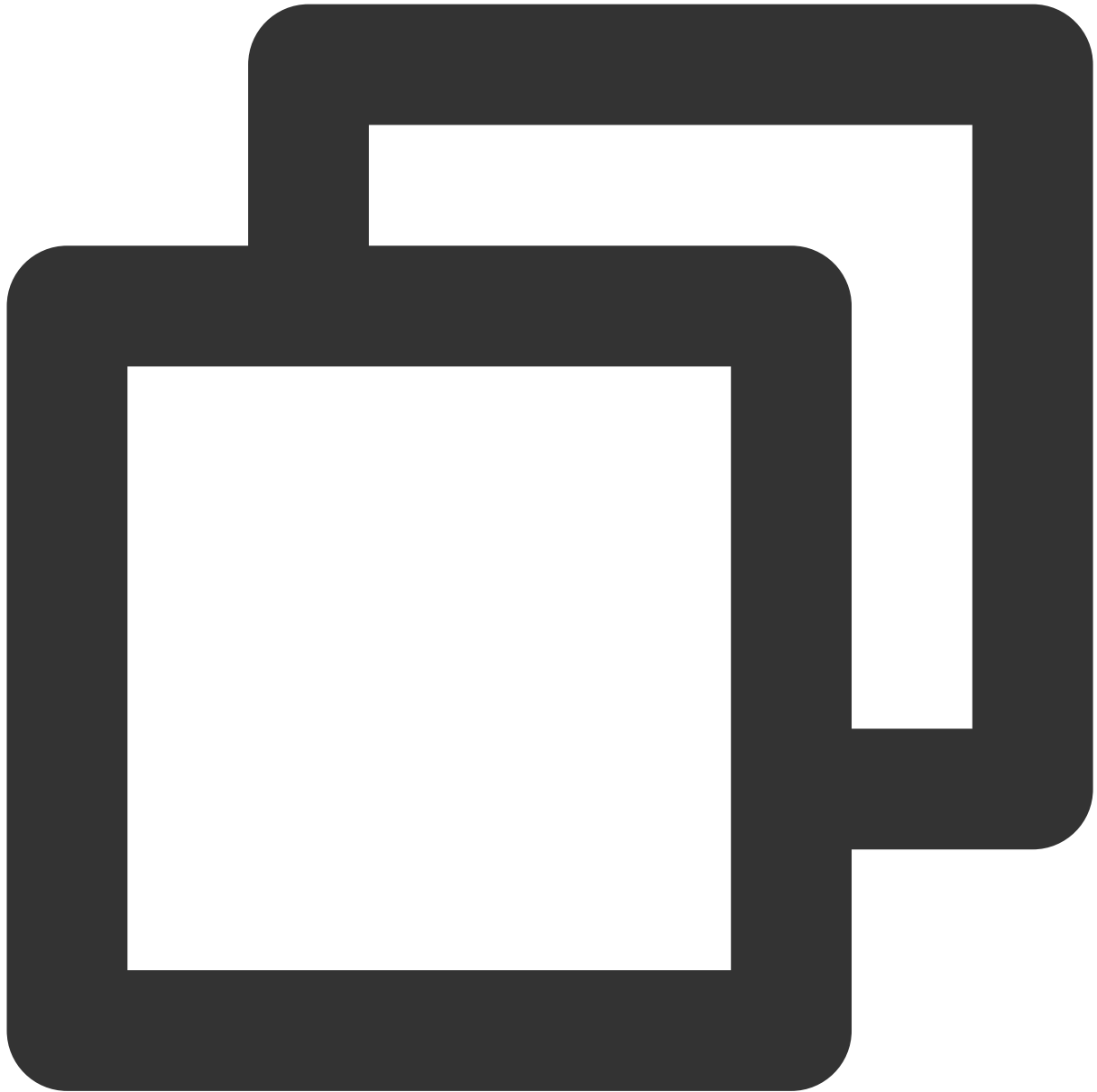
```
// src/main.ts
import { createPinia } from 'pinia';

const app = createApp(App);
// register for Pinia
app.use(createPinia());
app.mount('#app');
```

### 3. Configuring esLint Checksums

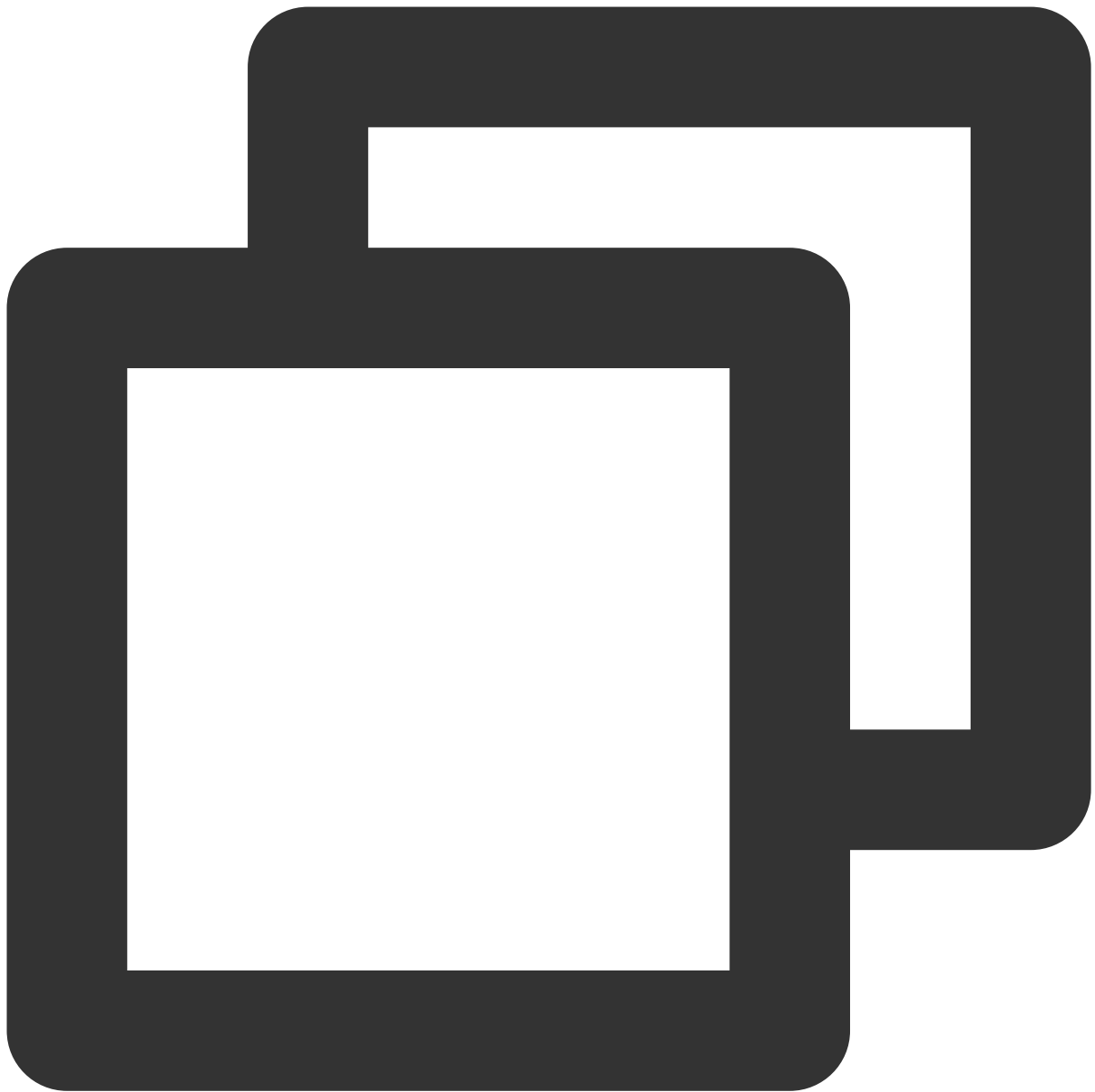


If you don't want the TUIRoomKit component's esLint rules to conflict with your local rules and cause runtime errors, you can add the Ignore TUIRoom folder to `.eslintignore`.



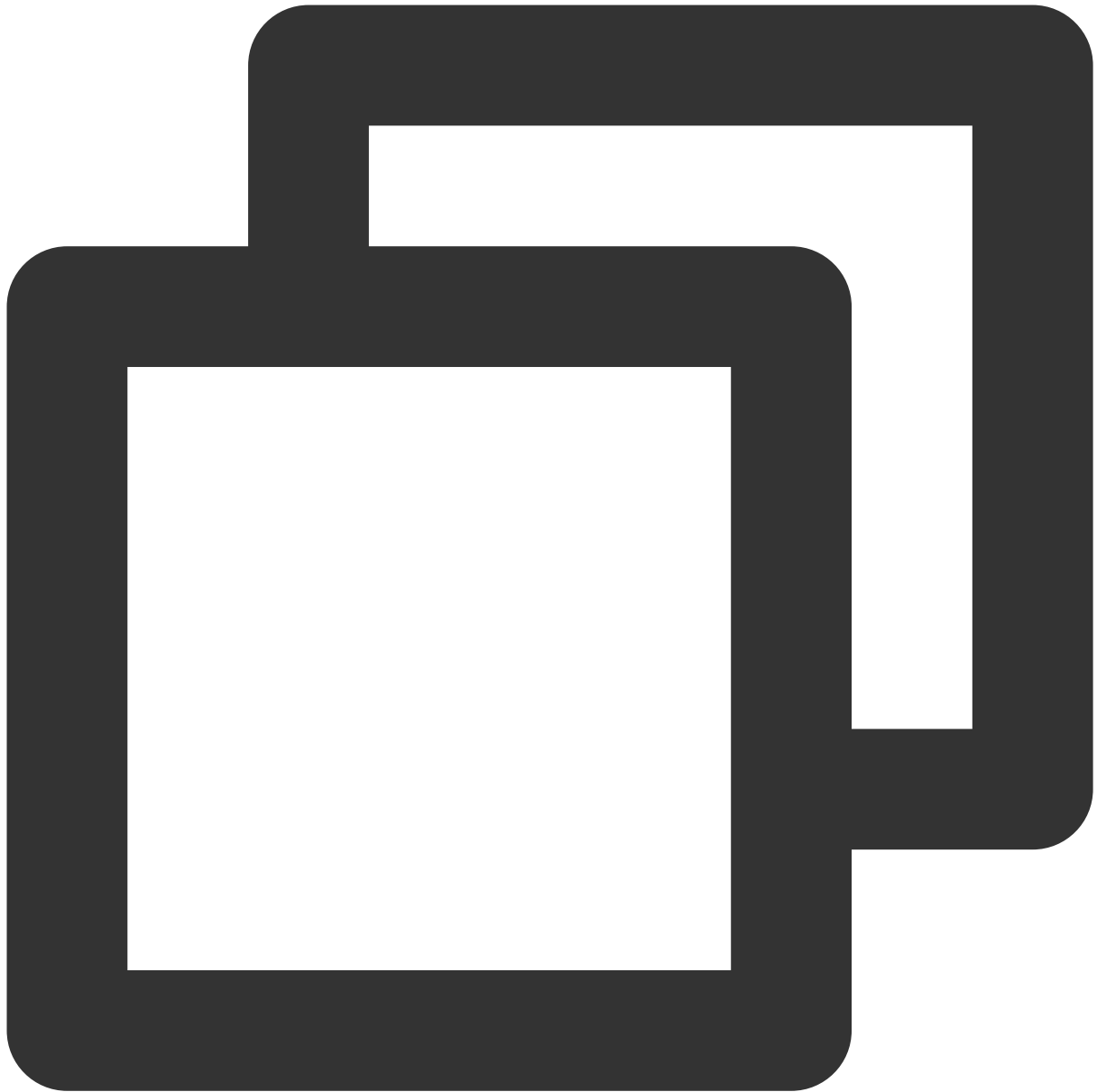
```
// Please add the real path to the TUIRoom source code  
src/components/TUIRoom
```

4. At this point you can run the project to see the effect of source code importation.



```
npm run dev
```

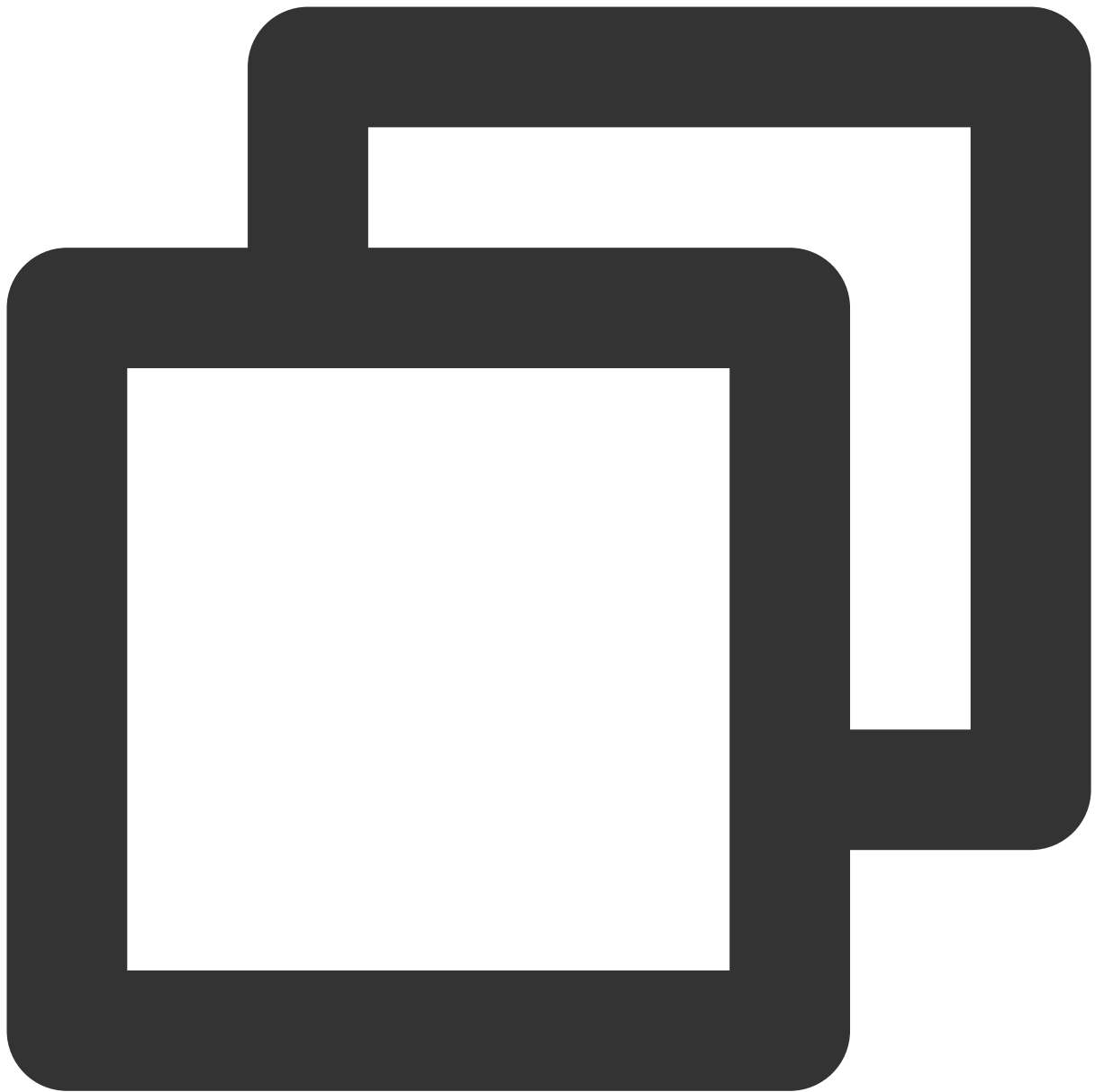
### 1. Install development environment dependencies



```
npm install typescript -S -D
```

## 2. Register for Pinia

TUIRoom uses Pinia for room data management, you need to register Pinia in the project entry file, which is the `src/main.ts` file.

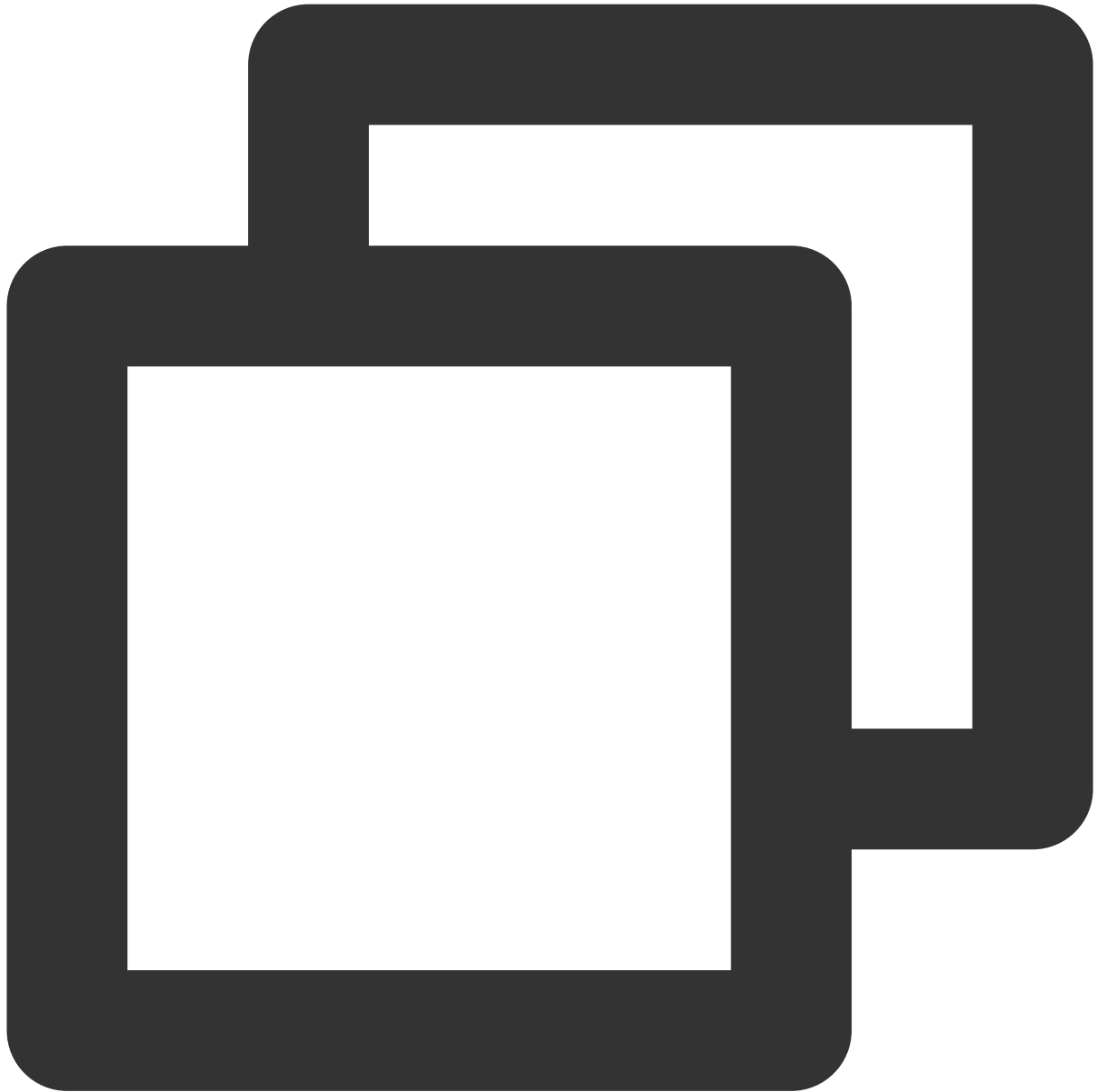


```
// src/main.ts
import { createPinia } from 'pinia';

const app = createApp(App);
// register for Pinia
app.use(createPinia());
app.mount('#app');
```

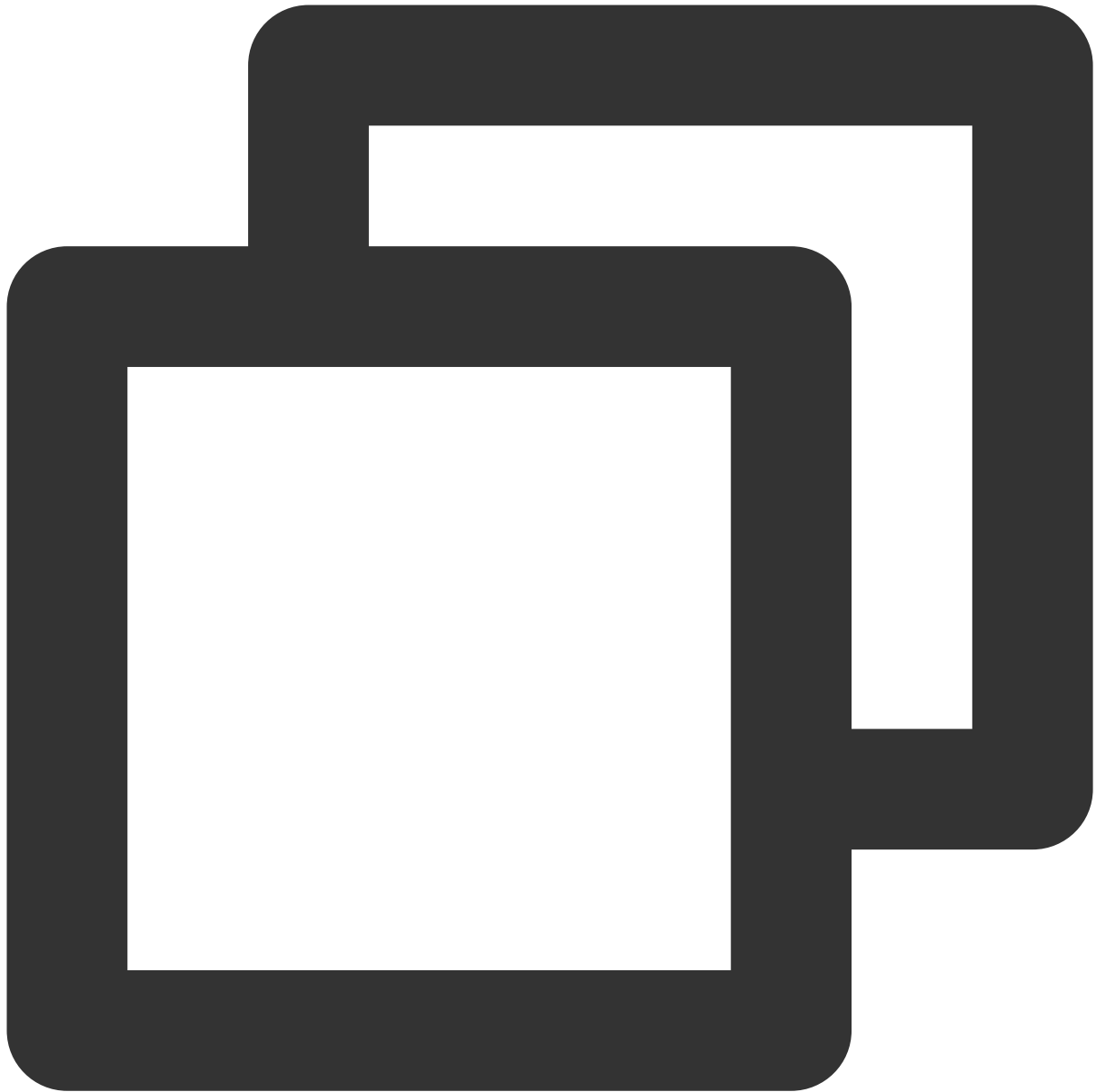
### 3. Configuring esLint Checksums

If you don't want the TUIRoomKit component's esLint rules to conflict with your local rules and cause runtime errors, you can add the Ignore TUIRoom folder to `.eslintignore`.



```
// Please add the real path to the TUIRoom source code  
src/components/TUIRoom
```

4. At this point you can run the project to see the effect of source code importation.



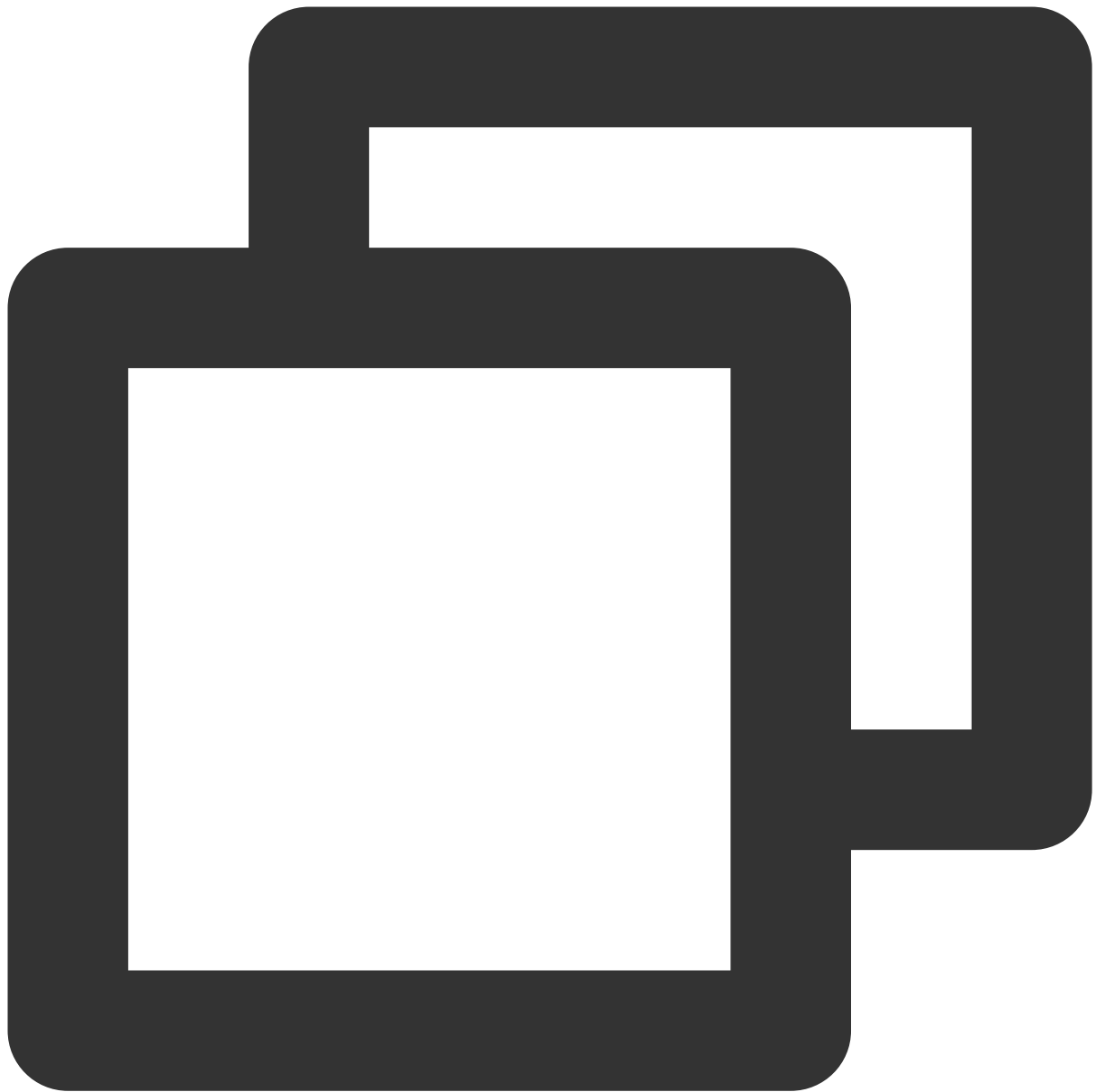
```
npm run serve
```

**Note :**

TUIRoomKit component requires vue2 project to have vue2.7 installed and supports typescript environment. If your current project is in vue2.6 + js environment, follow the steps below to upgrade to vue2.7 + ts environment.

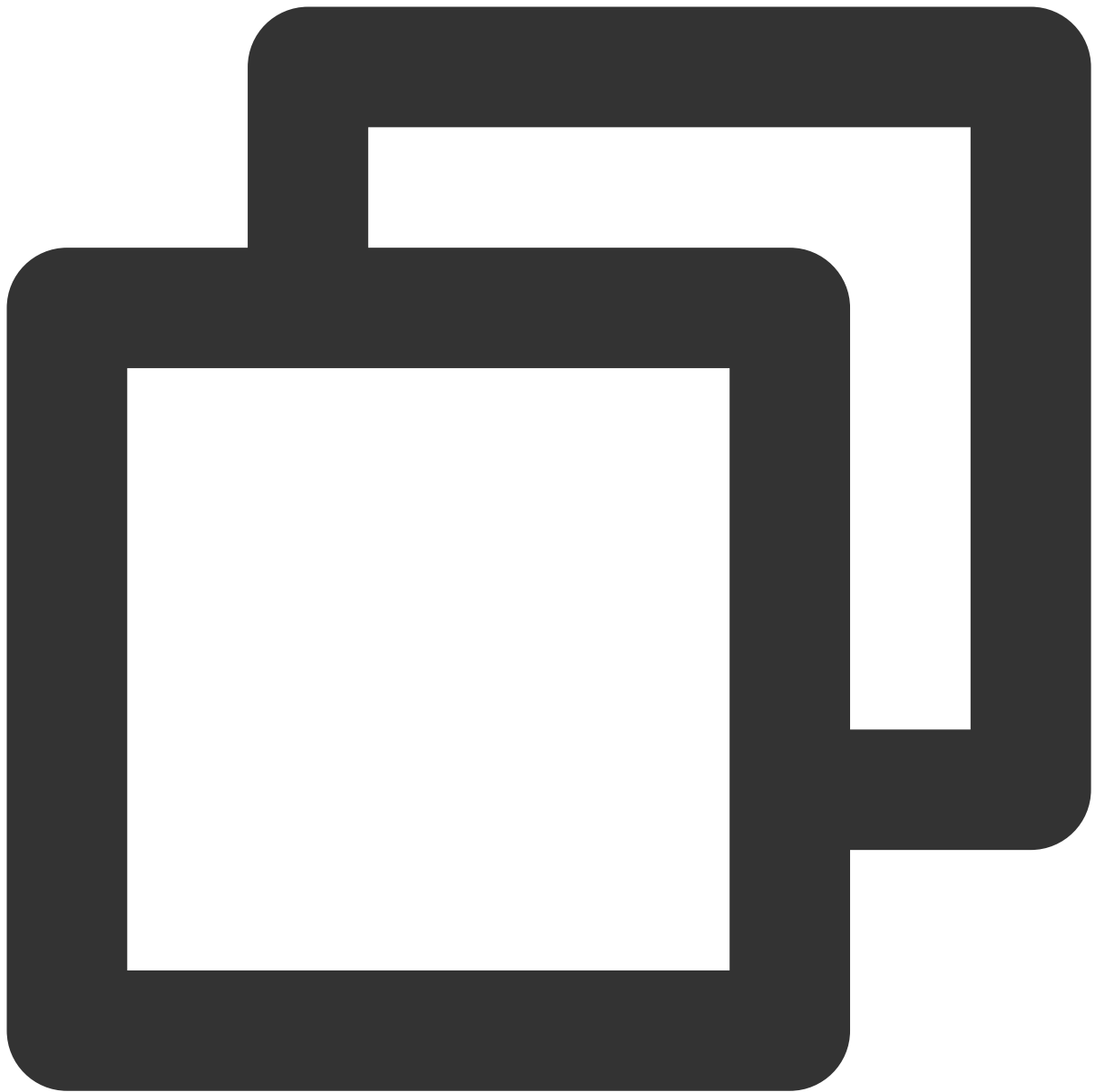
Upgrading from vue2.6 to vue2.7 is a smooth upgrade and has no effect on existing code. After configuring the ts environment, there is no impact on the existing js code. Please feel free to upgrade.

**1. Upgrade vue2 to v2.7+, if your project's vue version is already v2.7+, ignore this step.**



```
npm install vue@2 -S
```

## 2. Configure typescript to support TUIRoom component loading.



```
vue add typescript
```

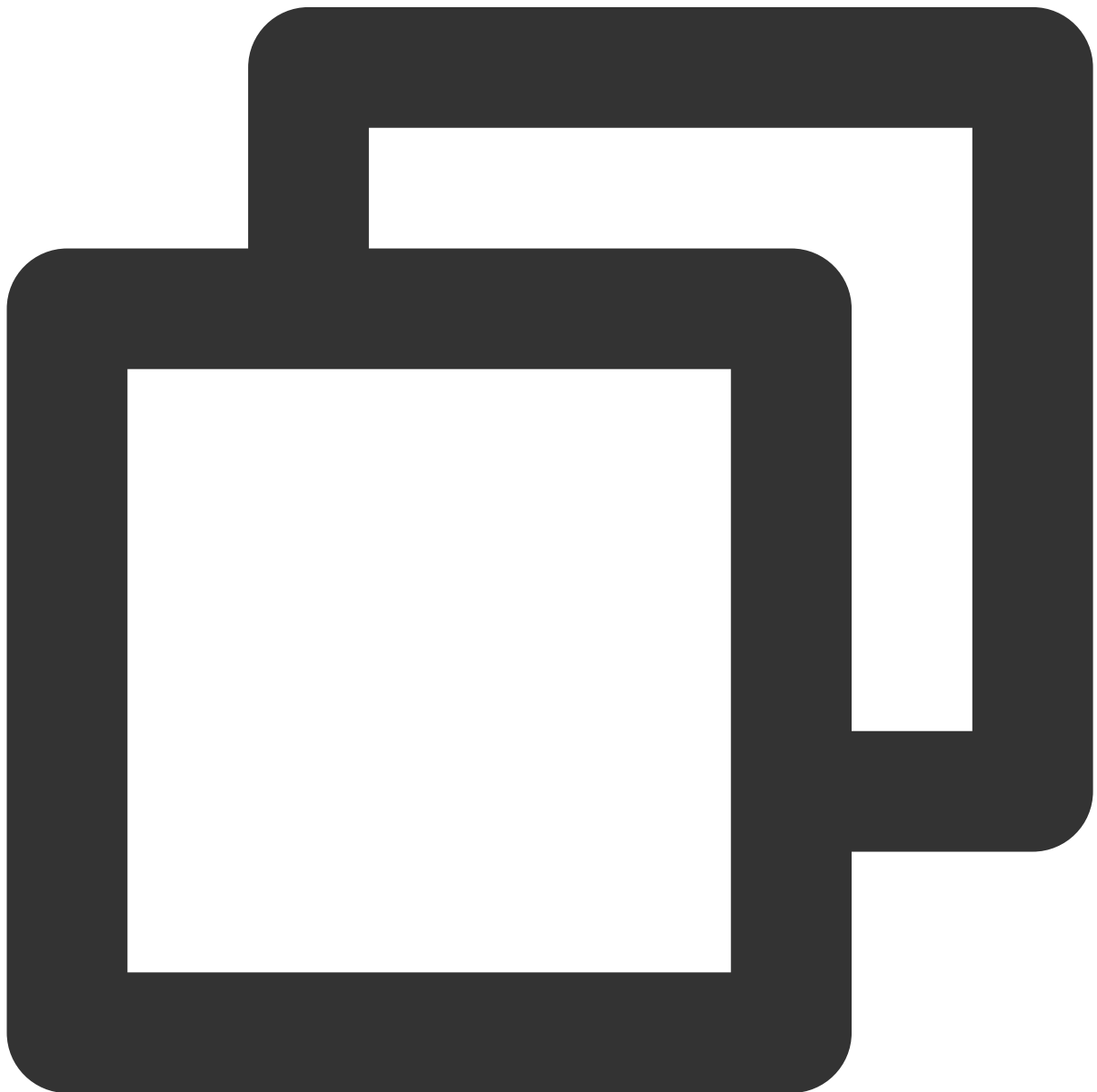
The options for configuring the TS development environment can be found in the image :



```
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? Yes
? Convert all .js files to .ts? No
? Allow .js files to be compiled? Yes
? Skip type checking of all declaration files (recommended for apps)? No
```

### 3. Register for Pinia

TUIRoom uses Pinia for room data management, you need to register Pinia in the project entry file, which is the `src/main.ts` file.



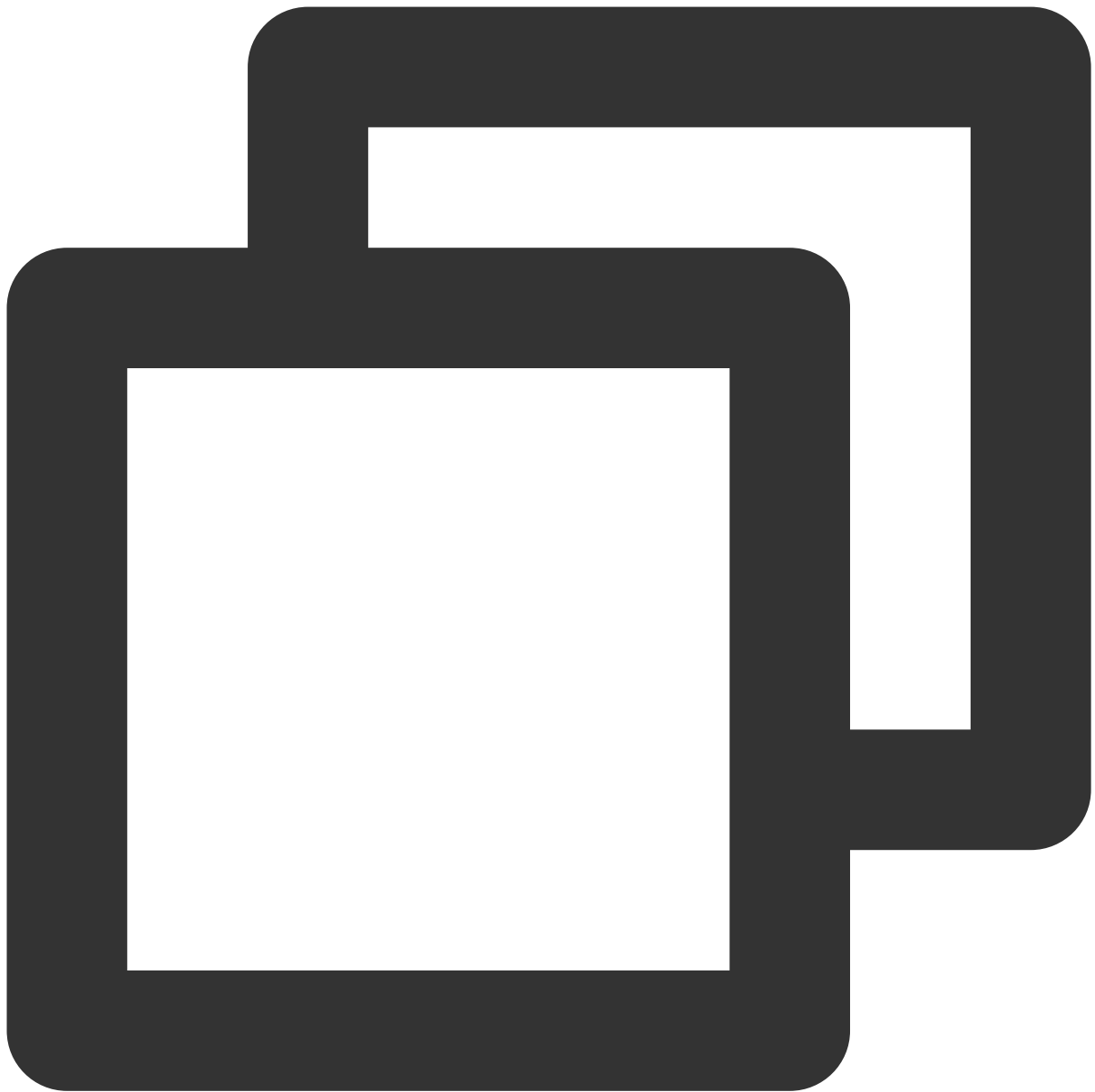
```
import { createPinia, PiniaVuePlugin } from 'pinia';
```

```
Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
}).$mount('#app');
```

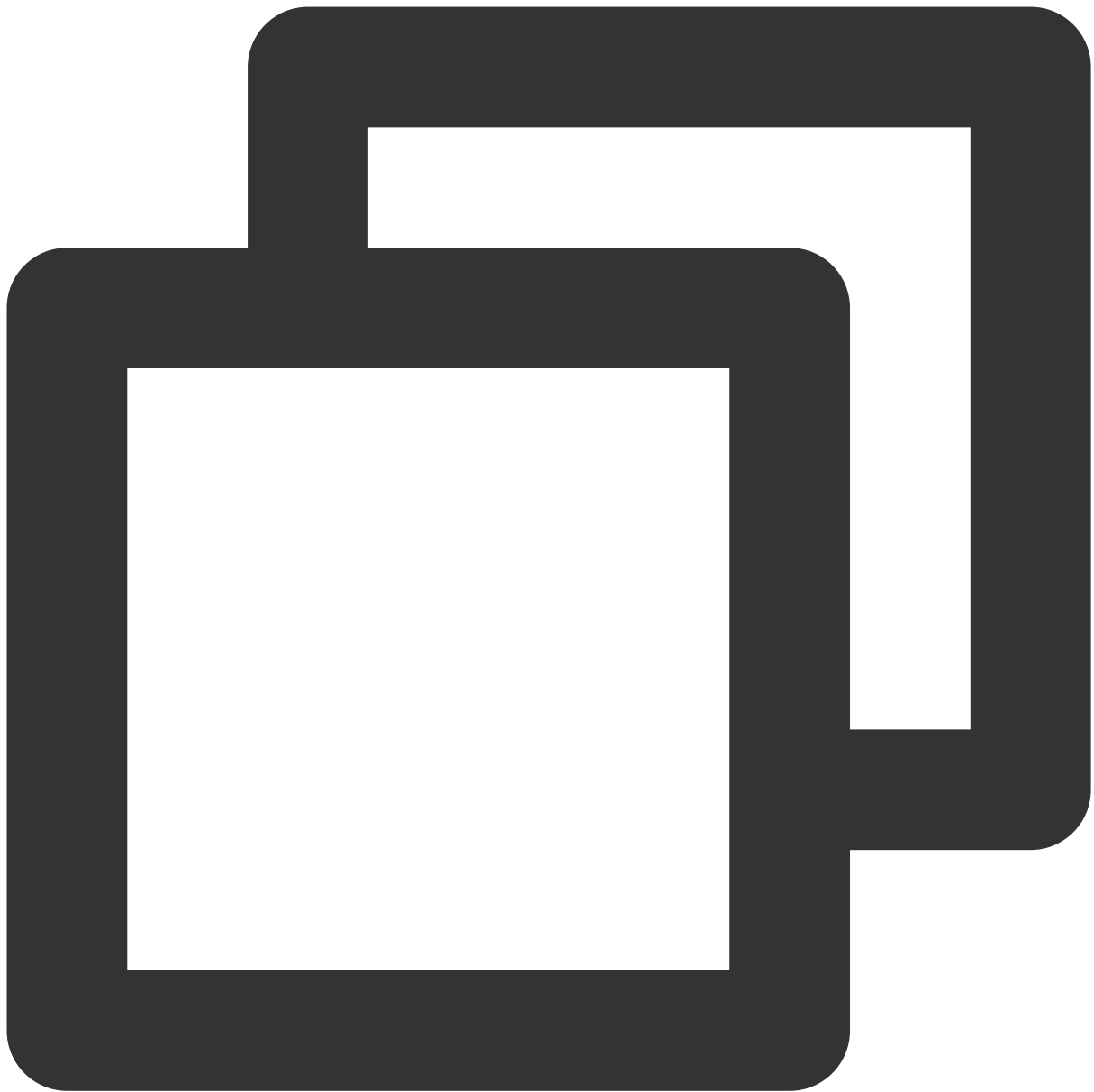
#### 4. Configuring esLint Checksums

If you don't want the TUIRoomKit component's esLint rules to conflict with your local rules and cause runtime errors, you can add the Ignore TUIRoom folder to `.eslintignore`.



```
// Please add the real path to the TUIRoom source code  
src/components/TUIRoom
```

5. At this point you can run the project to see the effect of source code importation.



```
npm run serve
```

### Step 3 : Modify the source code according to the requirements

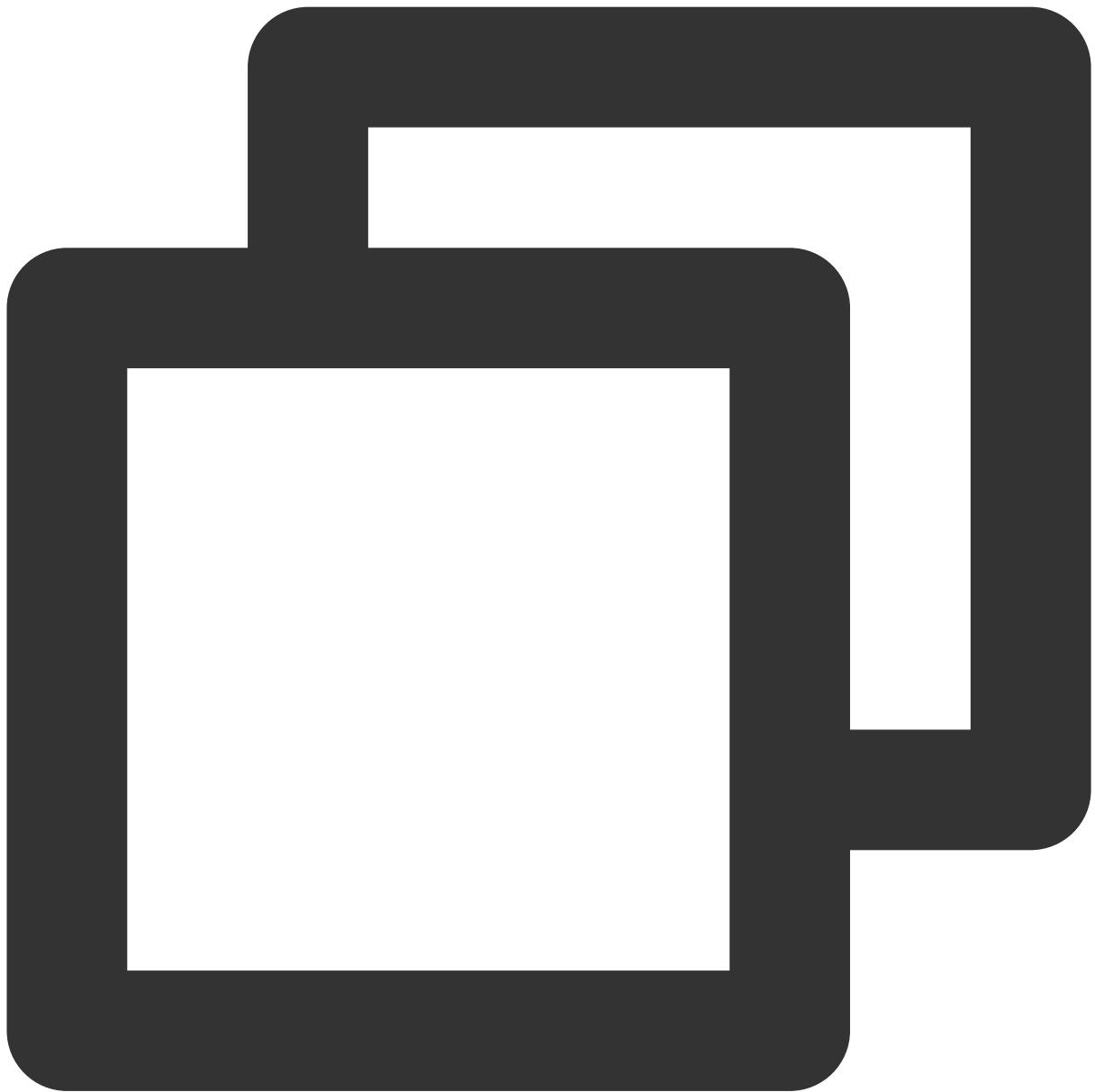
#### 1. Replace icons

You can directly modify the icon components in the `/TUIRoom/components/common/icons` folder to ensure that the icon color and style are consistent throughout the app. Please keep the icon file names unchanged when replacing.

## 2. Adjust UI layout

You can adjust the multi-person video conference interface UI layout by modifying different components in the

`/TUIRoom/components/` folder:



- components/
  - Chat Chat
  - common Common icon components
  - ManageMember Member management
  - RoomContent Room video
  - RoomFooter Room page Footer section
  - RoomHeader Room page Header section

- RoomHome	Home page
- RoomInvite	Invite members
- RoomLogin	Login page
- RoomMore	More
- RoomSetting	Settings
- RoomSidebar	Drawer component

## Method 3: Implement your own UI

The overall functionality of TUIRoomKit is based on the TUIRoomEngine, a UI-less SDK. You can fully implement your own UI interface based on TUIRoomEngine. For more information, please see:

[TUIRoomEngine Integration Guide](#)

[TUIRoomEngine API Interface Address](#)

# Flutter

Last updated : 2024-04-12 11:22:08

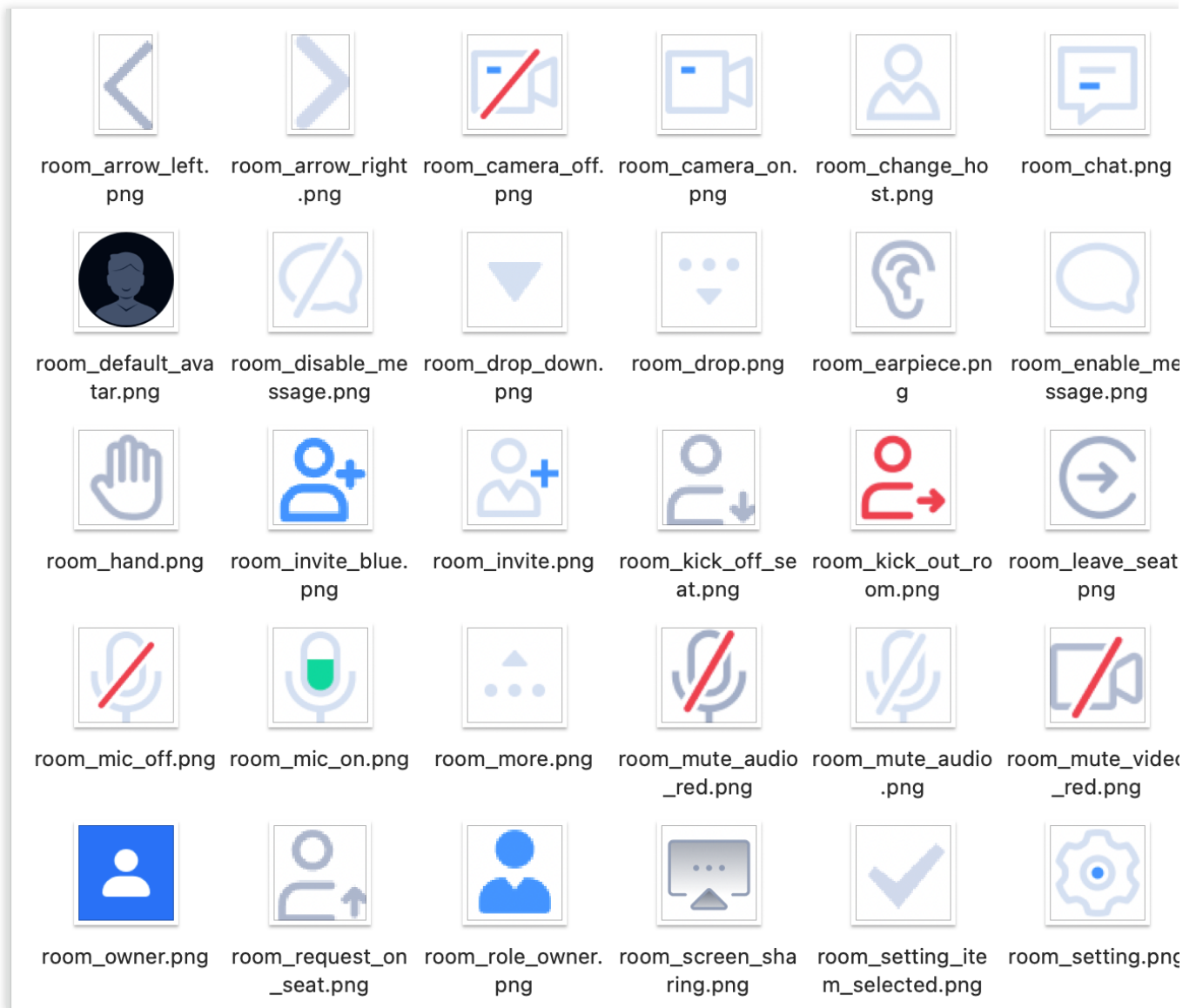
This article will introduce how to customize the user interface of `TUIRoomKit` . We provide two solutions for you to choose from: **fine-tuning solution** and **custom UI solution**.

## Solution 1: Fine-tuning solution

By directly modifying the UI source code we provide, you can adjust the user interface of `TUIRoomKit` .

### Replace icons

You can directly modify the icon components in the `rtc_conference_tui_kit/assets/images` folder to ensure that the icon color tone style is consistent throughout the app. Please keep the icon file name unchanged when replacing.



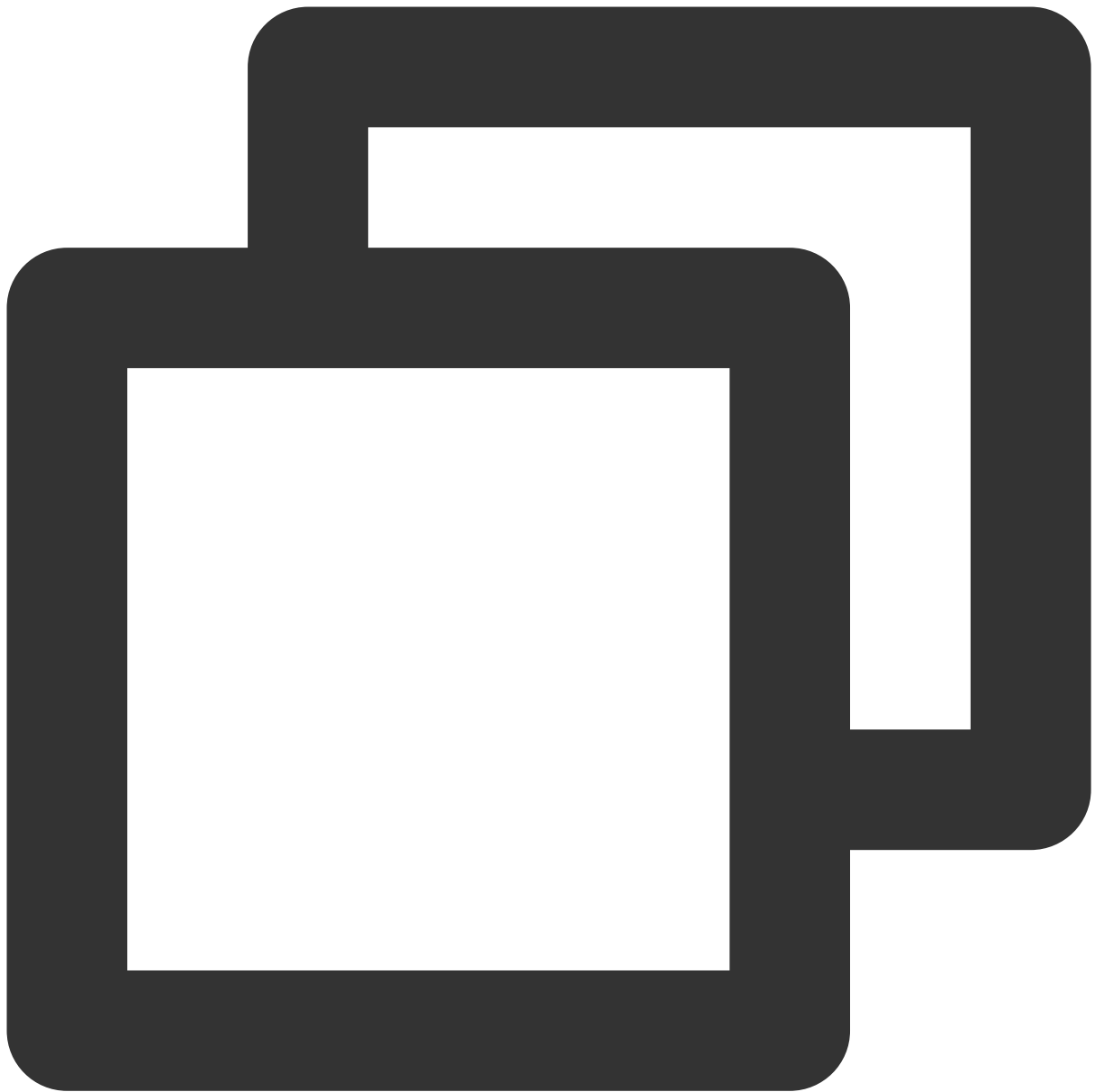
## Replace copywriting

You can modify the `en_us.dart` and `zh_cn.dart` files in the `rtc_conference_tui_kit/lib/common/languages` folder to change the string content of the video conference interface.

## Solution 2: Custom part of UI solution

The UI component folders of `TUIRoomKit` are as follows. You can modify the `view.dart` in the corresponding folder to customize the UI according to your actual business needs.





```
├─ lib
├─ common
│   └─ widgets
│       ├── bottom_sheet.dart (1 KB)           // Universal bottom sheet
│       ├── copy_text_button.dart (1 KB)       // Universal copy button
│       ├── dialog.dart (4 KB)                 // Universal dialog
│       ├── drop_down_button.dart (<1 KB)      // Universal dropdown button
│       ├── info_list.dart (1 KB)              // Universal information dialog
│       ├── rounded_container.dart (<1 KB)     // Universal rounded corner
│       ├── search_bar.dart (1 KB)             // Universal search bar
│       └── single_select_list.dart (1 KB)     // Universal single select
```

```

|       |─ slider.dart (1 KB) // Universal slider
|       |─ switch.dart (<1 KB) // Universal switcher
|       |─ toast.dart (<1 KB) // Universal toast
|       |─ user_info.dart (3 KB) // Universal user info item
|       |─ volume_bar.dart (2 KB) // Universal dynamic microp
|─ pages
|   |─ conference_main
|   |   |─ view.dart (3 KB)
|   |   |─ widgets
|   |       |─ bottom_view
|   |       |   |─ view.dart (1 KB) // Function button of bott
|   |       |   |─ widgets
|   |       |       |─ base_button.dart (10 KB) // Unexpanded basic bottom
|   |       |       |─ bottom_button_item.dart (2 KB) // Encapsulated universal
|   |       |       |─ mic_button.dart (1 KB) // Microphone button that
|   |       |       |─ more_button.dart (3 KB) // All bottom function but
|   |       |─ exit
|   |       |   |─ view.dart (2 KB) // Bottom sheet of exit ro
|   |       |─ invite_sheet
|   |       |   |─ invite_sheet.dart (1 KB) // Invite popup
|   |       |─ local_screen_sharing
|   |       |   |─ local_screen_sharing.dart (1 KB) // Prompt component during
|   |       |─ raise_hand_list
|   |       |   |─ view.dart (3 KB) // Raise hand application
|   |       |   |─ widgets
|   |       |       |─ title.dart (<1 KB) // List title bar
|   |       |       |─ user_item.dart (1 KB) // A single item of the en
|   |       |       |─ user_table.dart (1 KB) // List component
|   |       |─ setting
|   |       |   |─ view.dart (1 KB) // Settings panel
|   |       |   |─ widgets
|   |       |       |─ audio_setting.dart (1 KB) // Audio setting component
|   |       |       |─ setting_info_select.dart (1 KB) // Radio value setting com
|   |       |       |─ setting_item.dart (<1 KB) // Settings panel single s
|   |       |       |─ video_setting.dart (4 KB) // Video setting component
|   |       |─ top_view
|   |       |   |─ view.dart (3 KB) // Function button of top
|   |       |   |─ widgets
|   |       |       |─ meeting_title.dart (2 KB) // Top meeting information
|   |       |       |─ room_info_sheet.dart (2 KB) // Detailed meeting inform
|   |       |       |─ top_button_item.dart (1 KB) // Top function single uni
|   |       |─ transfer_host
|   |       |   |─ view.dart (1 KB) // Transfer homeowners pan
|   |       |   |─ widgets
|   |       |       |─ title.dart (<1 KB) // Title bar of transfer h
|   |       |       |─ user_item.dart (1 KB) // Transferable homeowner
|   |       |       |─ user_table.dart (1 KB) // Transferable homeowner

```

```

|      | └─ user_list
|      |   └─ view.dart (3 KB)                // User list Page
|      |     └─ widgets
|      |         └─ button_item.dart (1 KB)    // Button item at the bott
|      |         └─ user_control.dart (7 KB)    // The member management p
|      |         └─ user_control_item.dart (1 KB) // The item of member mana
|      |         └─ user_item.dart (3 KB)       // User list item
|      |         └─ user_table.dart (3 KB)      // User list
|      | └─ video_seat
|      |   └─ video_layout
|      |     └─ view.dart (3 KB)                // The layout of video ite
|      |       └─ widgets
|      |           └─ video_item                // Single video frame item
|      |               └─ view.dart (4 KB)      // Single video frame item
|      |                   └─ widgets
|      |                       └─ video_user_info.dart (2 KB)    // User inform
|      |                       └─ volume_bar.dart (<1 KB)        // Microphone
|      |                       └─ with_draggable_window_widget.dart (1 KB) // Two-person
|      | └─ video_page_turning
|      |   └─ view.dart (2 KB)                  // Video scree
|      |     └─ widgets
|      |         └─ page_indicator.dart (1 KB)  // Page indica

```

## Solution 3: Custom all UI solution

The overall function of `TUIRoomKit` is based on the `TUIRoomEngine`, a UI-less SDK. You can completely implement your own UI interface based on `TUIRoomEngine`. For details, please see [TUIRoomEngine API interface address](#).

# Virtual Background (TUIRoomKit)

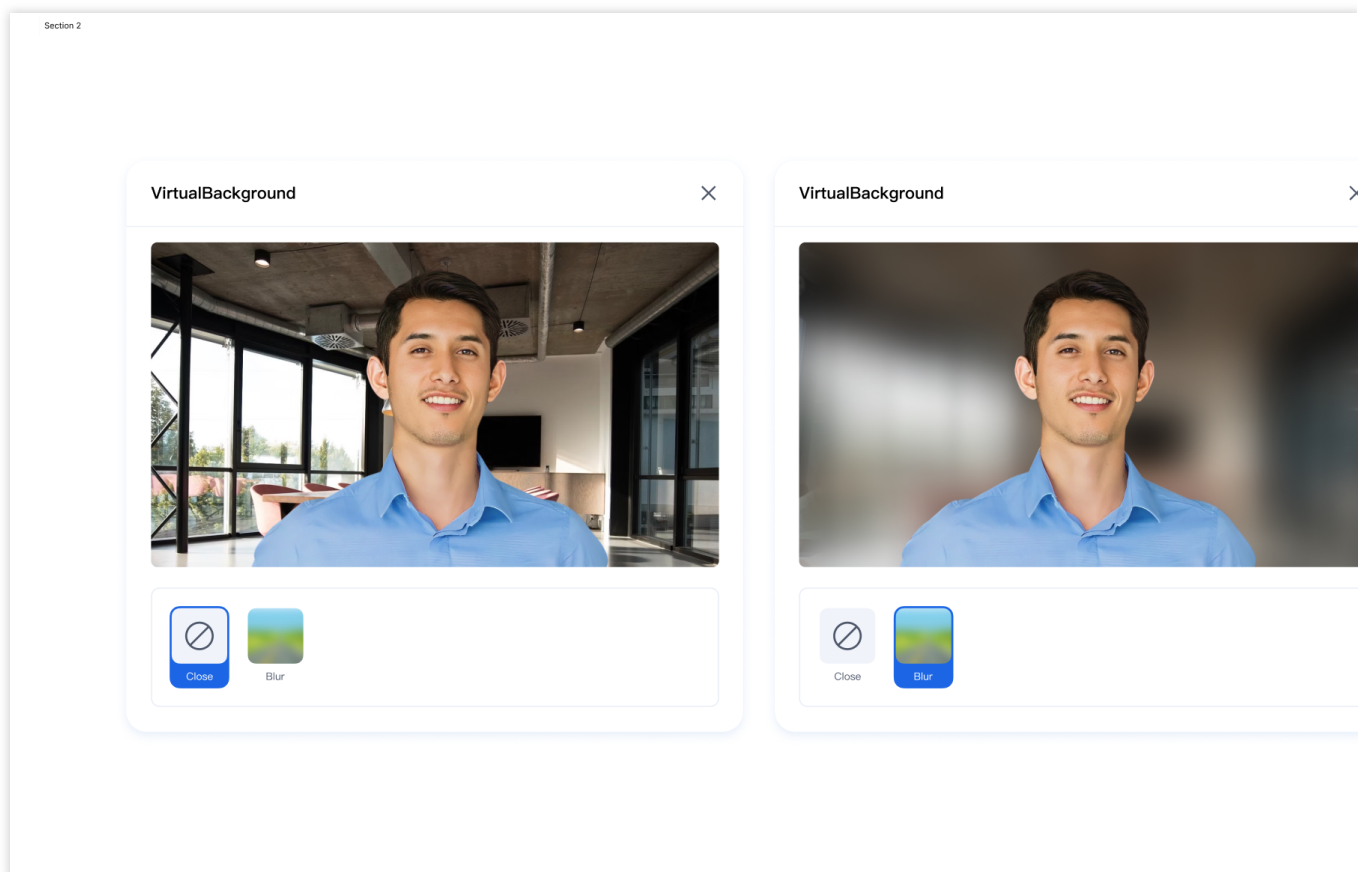
## Web

Last updated : 2024-05-30 13:07:39

TUIRoomKit has launched a Virtual Background feature, allowing users to set a blurry background during group meetings, hide the actual meeting environment, protect privacy, and add fun to the meeting. This article will introduce in detail how to use this feature in the TUIRoomKit component.

## Integration effect

After integrating the Virtual Background feature in the TUIRoomKit component, the display effect is as follows:



## Preparation Requirements

Before using Tencent Cloud's Virtual Background feature, you need to go to the Console to activate the group meeting service for your application, and purchase the **Advanced Interactive Version/Large Room Interactive Version** package. For specific steps, please refer to [Activate Service](#).

## Enable Blurry Background

### Note:

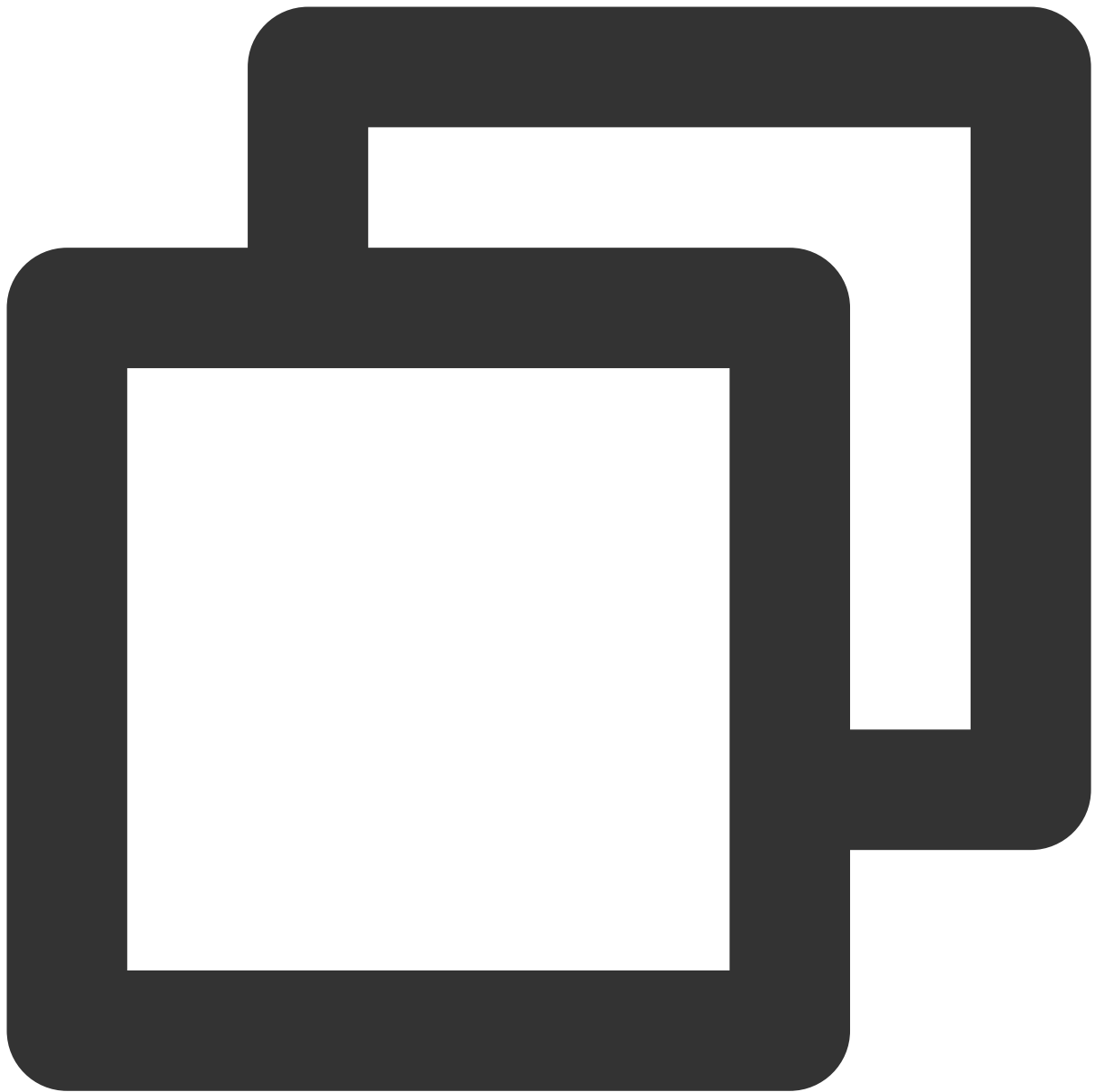
**H5 is not supported, only applicable to Web PC.**

**You need TUIRoomKit v2.3.3 or later.**

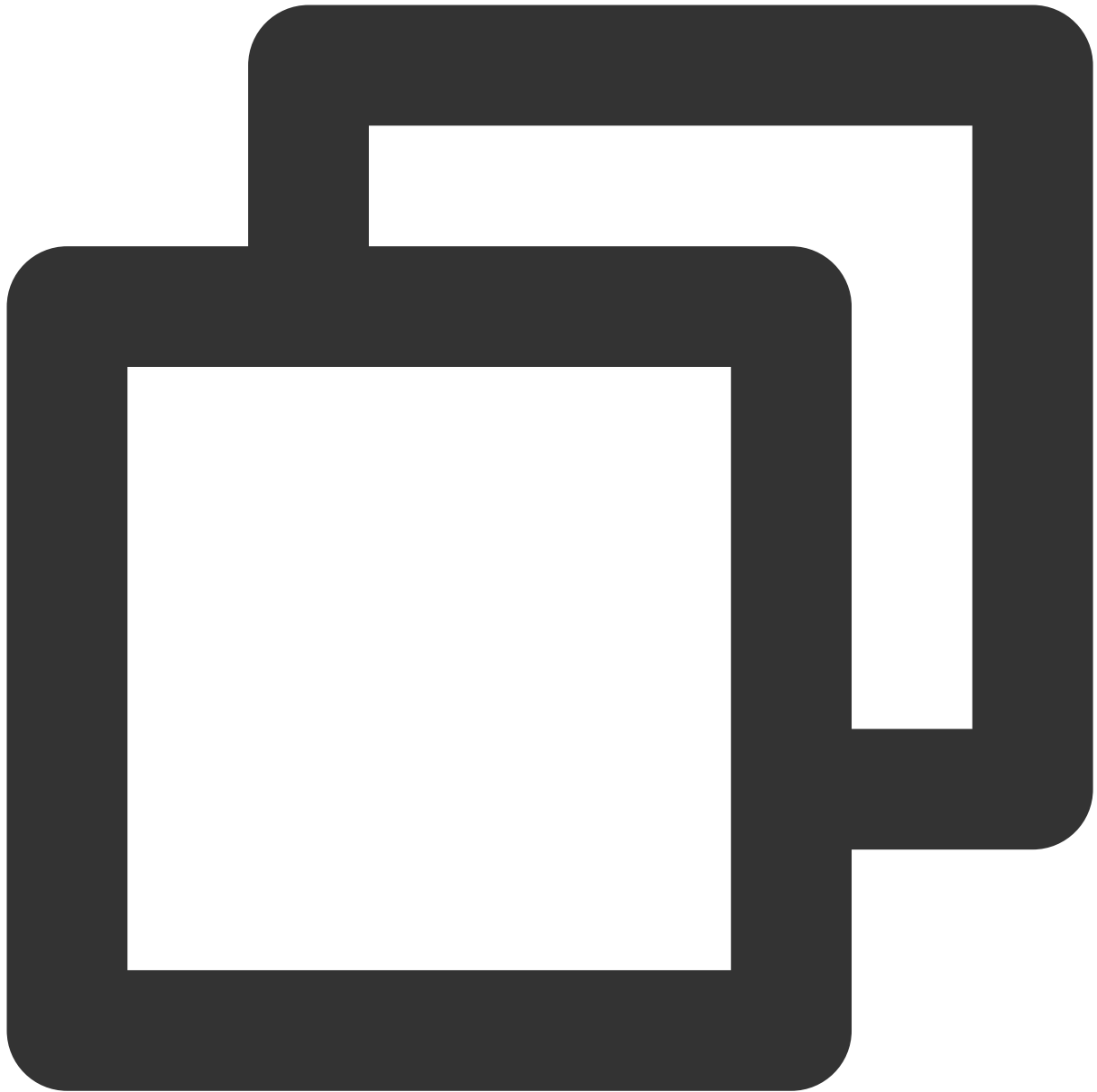
TUIRoomKit's UI solution supports setting a virtual background. You can display the Virtual Background feature button on the UI by calling the [enableVirtualBackground](#) interface. Clicking this button will enable the Virtual Background feature.

Vue3

Vue2



```
import { conference } from '@tencentcloud/roomkit-web-vue3';  
conference.enableVirtualBackground();
```



```
import { conference } from '@tencentcloud/roomkit-web-vue2.7';  
conference.enableVirtualBackground();
```

## FAQs

### No response or delay when activating virtual background?

Make sure you have purchased the group meeting **Advanced Interactive Version/Large Room Interactive Version** package, see [Activate Service](#) for details.

When the network connection is poor, the virtual background model file may not have been completely downloaded, leading to the failure of activating the virtual background.

### **Can I activate the virtual background with the camera off?**

No.



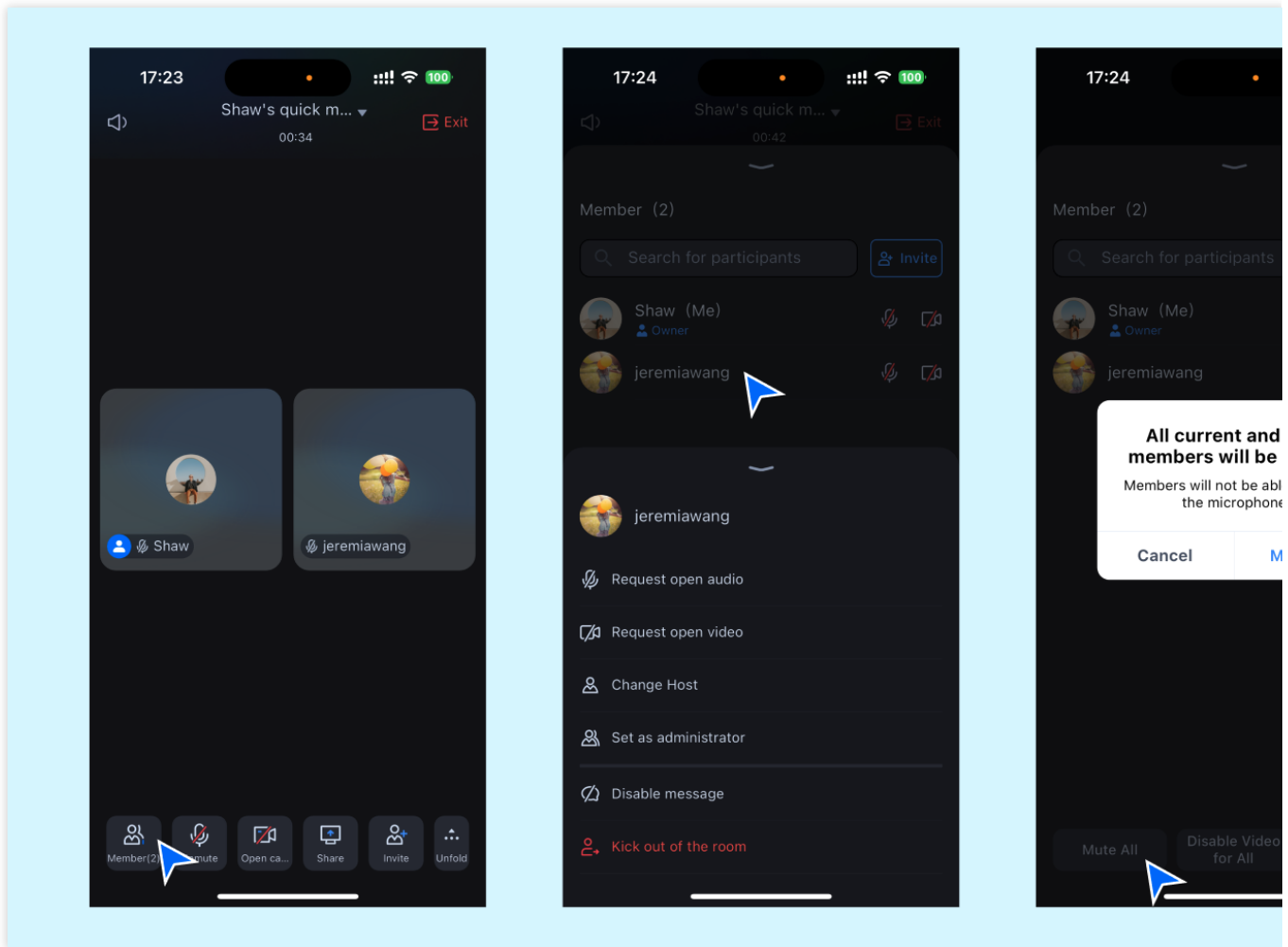
# Conference Control (TUIRoomKit)

## Android&iOS&Flutter

Last updated : 2024-05-29 15:07:56

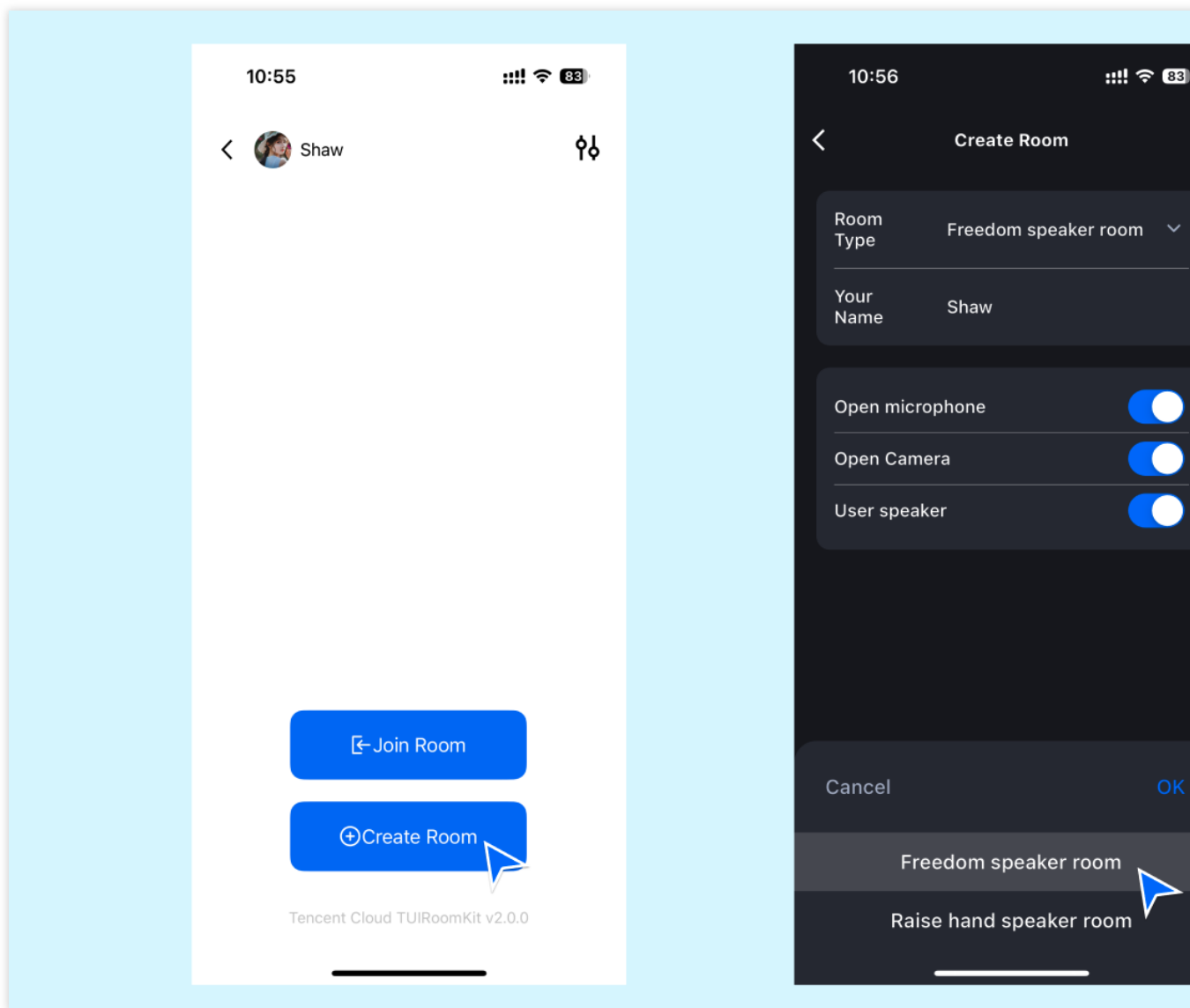
This document will provide a detailed introduction to the pre-conference control, in-conference control, and other aspects of `TUIRoomKit` to help you better master the conference control features of `TUIRoomKit`. Through this document, you will be able to fully utilize the functions of `TUIRoomKit` to achieve high-quality audio and video conferences.

Upon creating and entering a room via `Android&iOS&Flutter` platforms, the room host or an administrator can access the participant management options by clicking the member button on the bottom toolbar. This action will prompt a member list to appear at the bottom of the screen. Within this list, the host or administrator can select any regular member to enforce actions such as **muting a user's messages** or **setting them as an administrator**, among other conference control operations. In addition, the host or administrator has the ability to perform conference-wide control actions, such as **muting audio for all participants** in the room.



## Pre-conference Control

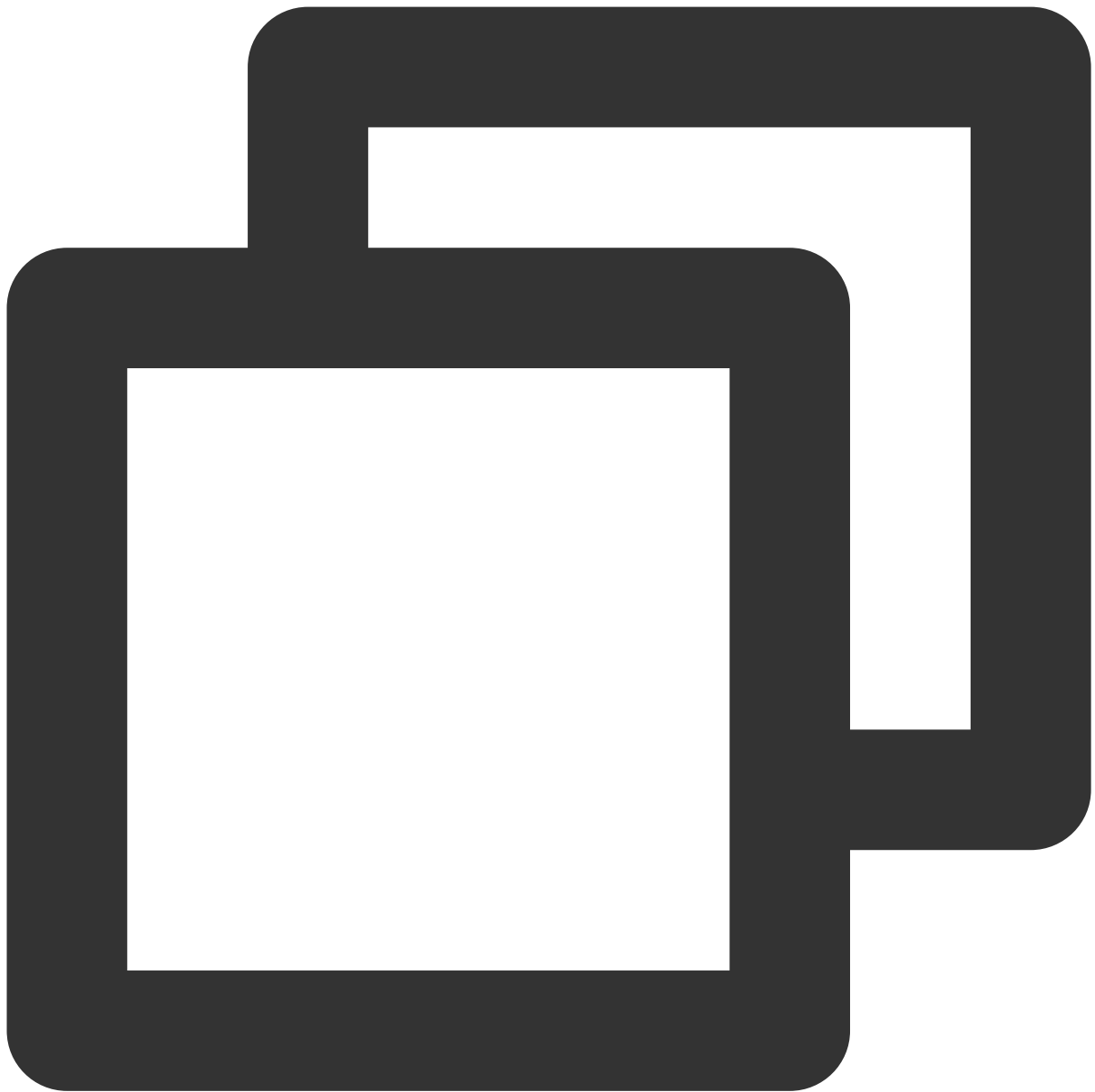
Before entering the conference, you can use the pre-conference control features of `TUIRoomKit` to set the relevant parameters of the conference in advance, ensuring the smooth progress of the conference.



iOS (Swift)

Android (Java)

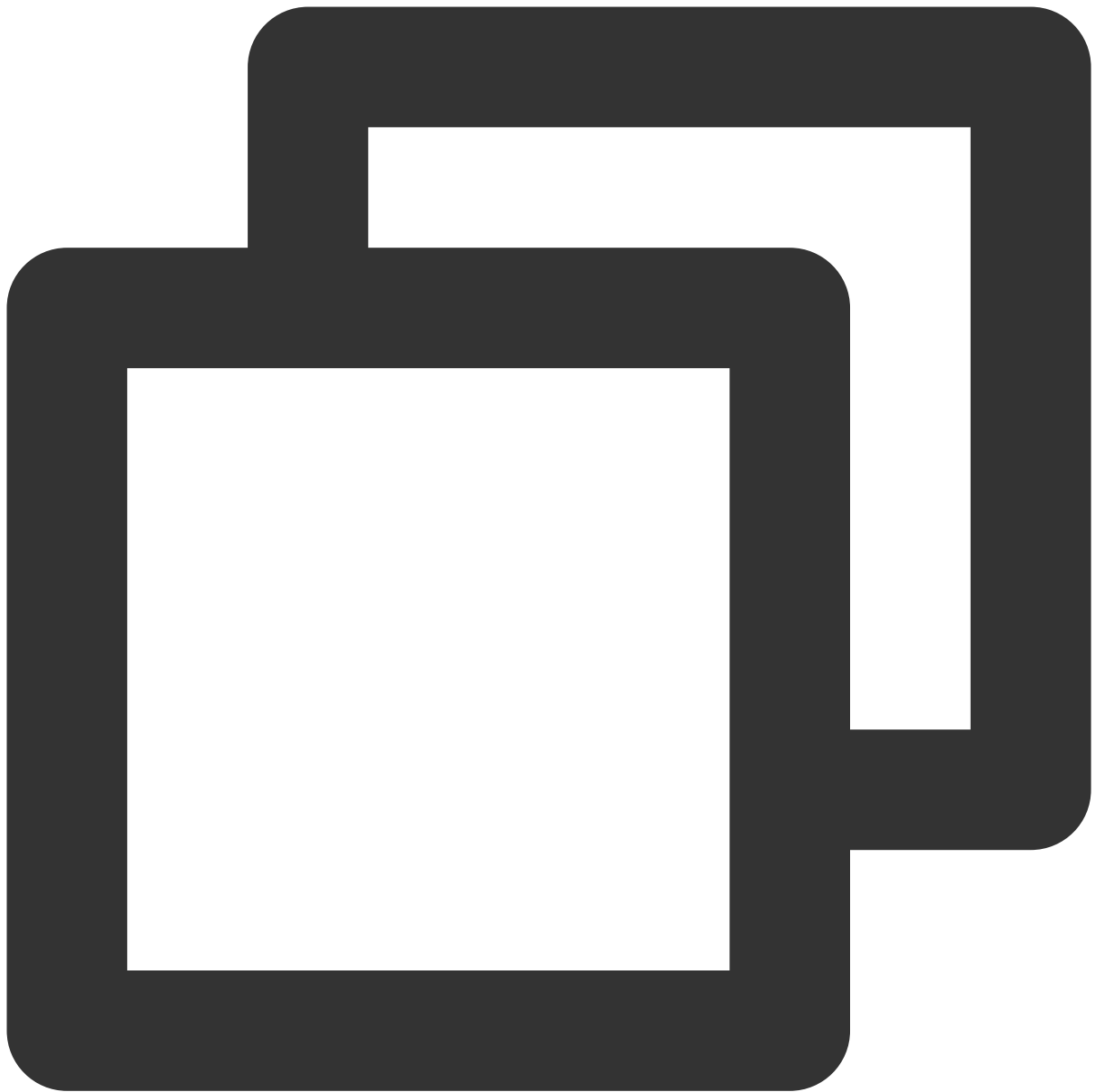
Flutter (Dart)



```
// CreateRoomViewController for your own ViewController
class CreateConferenceViewController: UIViewController {
    private var conferenceViewController: ConferenceMainViewController?
    func quickStartConferenceAction() {
        conferenceViewController = ConferenceMainViewController()
        // Implement the pre-conference control features by setting the parameters in
        let params = ConferenceParams()
        params.isMuteMicrophone = false
        params.isOpenCamera = false
        params.isSoundOnSpeaker = true
        params.name = "your conference name"
```

```
        params.enableMicrophoneForAllUser = true
        params.enableCameraForAllUser = true
        params.enableMessageForAllUser = true
        params.enableSeatControl = false
        conferenceViewController?.setConferenceParams(params: params)
        conferenceViewController?.setConferenceObserver(observer: self)
        // After completing the settings, call the interface to start or join a conference
        conferenceViewController?.quickStartConference(conferenceId: "your conference id")
    }
}

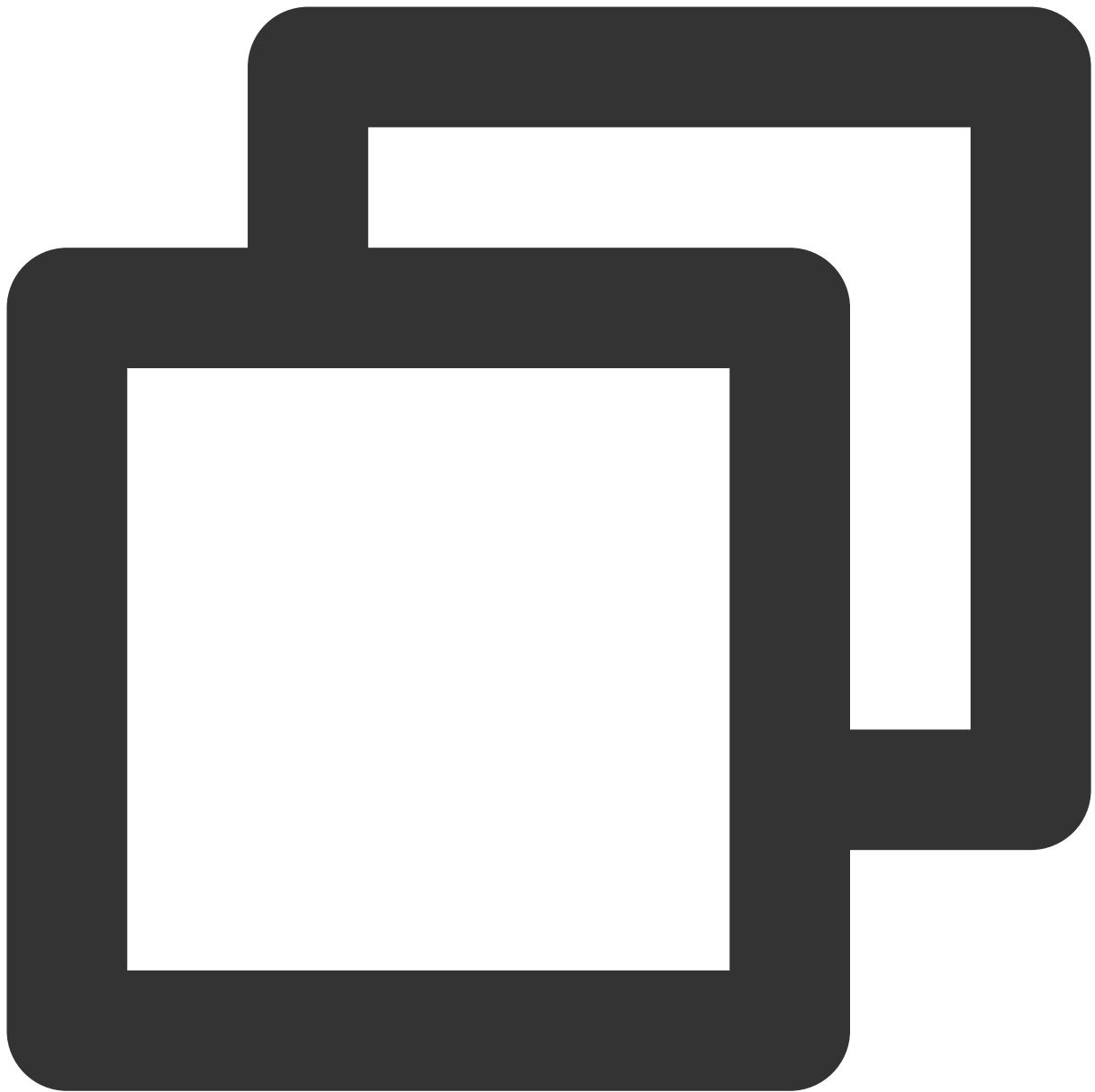
extension CreateConferenceViewController: ConferenceObserver {
    func onConferenceStarted(conferenceId: String, error: ConferenceError) {
        if error == .success, let vc = conferenceViewController {
            navigationController?.pushViewController(vc, animated: true)
        }
        conferenceViewController = nil
    }
}
```



```
public class ConferenceOwnerActivity extends AppCompatActivity {  
    private static final String TAG = "ConferenceOwnerActivity";  
  
    private ConferenceObserver mConferenceObserver;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.app_activity_conference_main);  
        // Implement the pre-conference control features by setting the parameters  
        ConferenceParams params = new ConferenceParams();  
        params.setMuteMicrophone(false);  
    }  
}
```

```
        params.setOpenCamera(false);
        params.setSoundOnSpeaker(true);
        params.setName("your conference name");
        params.setEnableMicrophoneForAllUser(true);
        params.setEnableCameraForAllUser(true);
        params.setEnableMessageForAllUser(true);
        params.setEnableSeatControl(false);
        ConferenceMainFragment fragment = new ConferenceMainFragment();
        fragment.setConferenceParams(params);
        setConferenceObserver(fragment);
        fragment.quickStartConference("your conferenceId"); // After completing th
    }

    private void setConferenceObserver(ConferenceMainFragment fragment) {
        mConferenceObserver = new ConferenceObserver() {
            @Override
            public void onConferenceStarted(String conferenceId, ConferenceError er
                super.onConferenceStarted(conferenceId, error);
                if (error != ConferenceError.SUCCESS) {
                    Log.e(TAG, "Error : " + error);
                    return;
                }
                FragmentManager manager = getSupportFragmentManager();
                FragmentTransaction transaction = manager.beginTransaction();
                transaction.add(R.id.conference_owner_container, fragment);
                transaction.commitAllowingStateLoss();
            }
        };
        fragment.setConferenceObserver(mConferenceObserver);
    }
}
```



```
var conferenceSession = ConferenceSession.newInstance("your conferenceId")
    ..isMuteMicrophone = false
    ..isOpenCamera = false
    ..isSoundOnSpeaker = true
    ..name = "your conference name"
    ..enableMicrophoneForAllUser = true
    ..enableCameraForAllUser = true
    ..enableMessageForAllUser = true
    ..enableSeatControl = false
    ..onActionSuccess = () { // Successful operation callback, you can navigate
        Navigator.push(
```



```
        context,
        MaterialPageRoute(
            builder: (context) => ConferenceMainPage(),
        ),
    );
}

..onActionError = (ConferenceError error, String message) {} // Failure oper
..quickStart(); // After completing the settings, call the interface
```

Here is a detailed introduction to the parameters in the above code.

Parameter	Type	Meaning
isMuteMicrophone	bool	Whether to mute the microphone (default is false)
isOpenCamera	bool	Whether to open the camera (default is false)
isSoundOnSpeaker	bool	Whether to turn on the speakers (default is true)
name	String	conference name (default is your conferenceld)
enableMicrophoneForAllUser	bool	Whether to enable microphone permission for all members (default is true)
enableCameraForAllUser	bool	Whether to enable camera permission for all members (default is true)
enableMessageForAllUser	bool	Whether to enable message permission for all members (default is true)
enableSeatControl	bool	Whether to enable on-stage speaking mode (default is false)

#### Note:

The above is an introduction to the parameters for creating and joining a conference in the aforementioned code. You can create either a free-speech room or a stage-speech room based on the different values passed to the `isSeatEnable` parameter. The control features available in these two types of rooms will also vary:

**Free-Speech Room:** Regular users can freely speak and have the liberty to turn their microphones and cameras on or off.

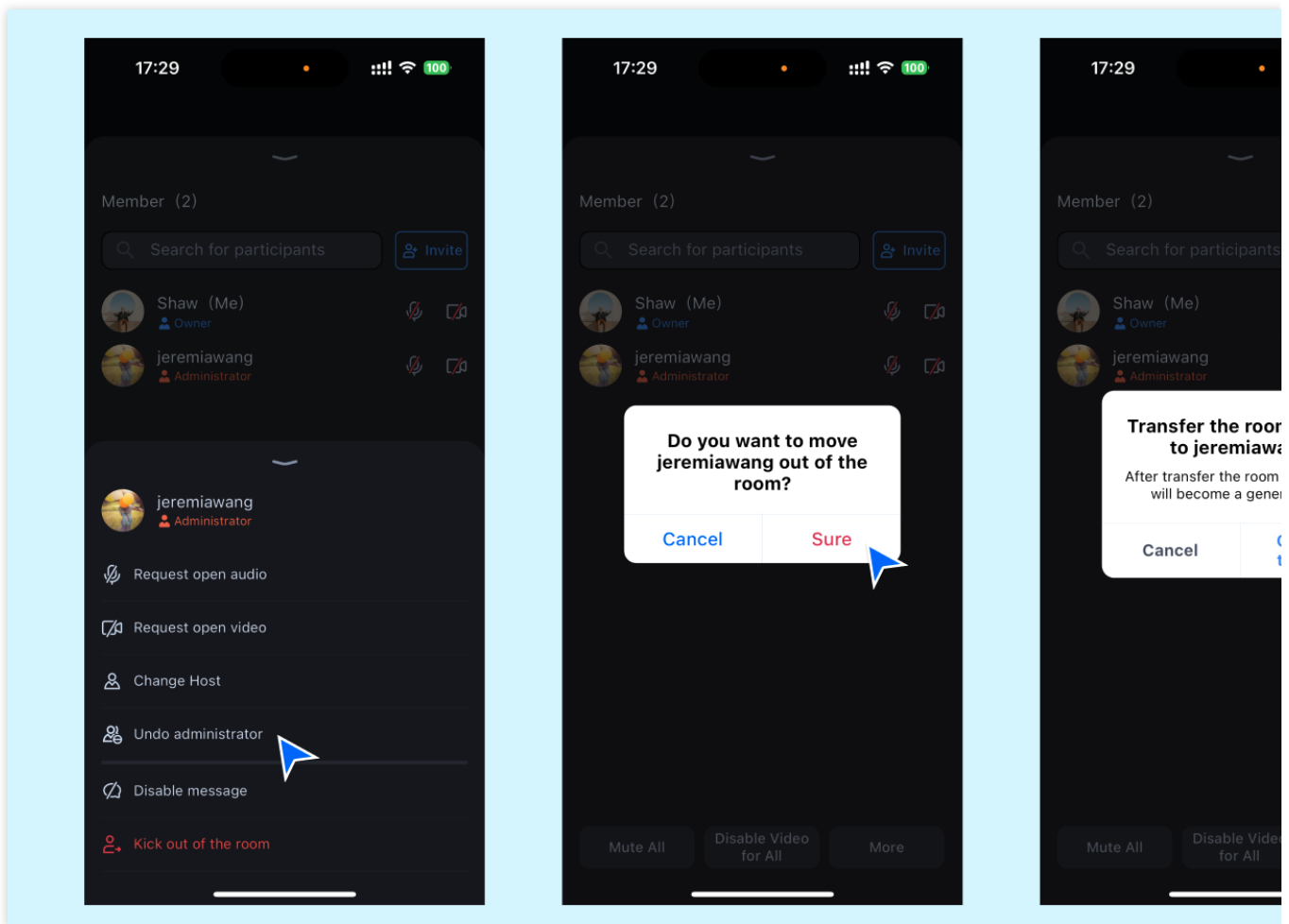
**Stage-Speech Room:** Only users on stage can freely turn their microphones and cameras on or off. Regular audience members can apply to become stage users by raising their hand.

## In-conference Control

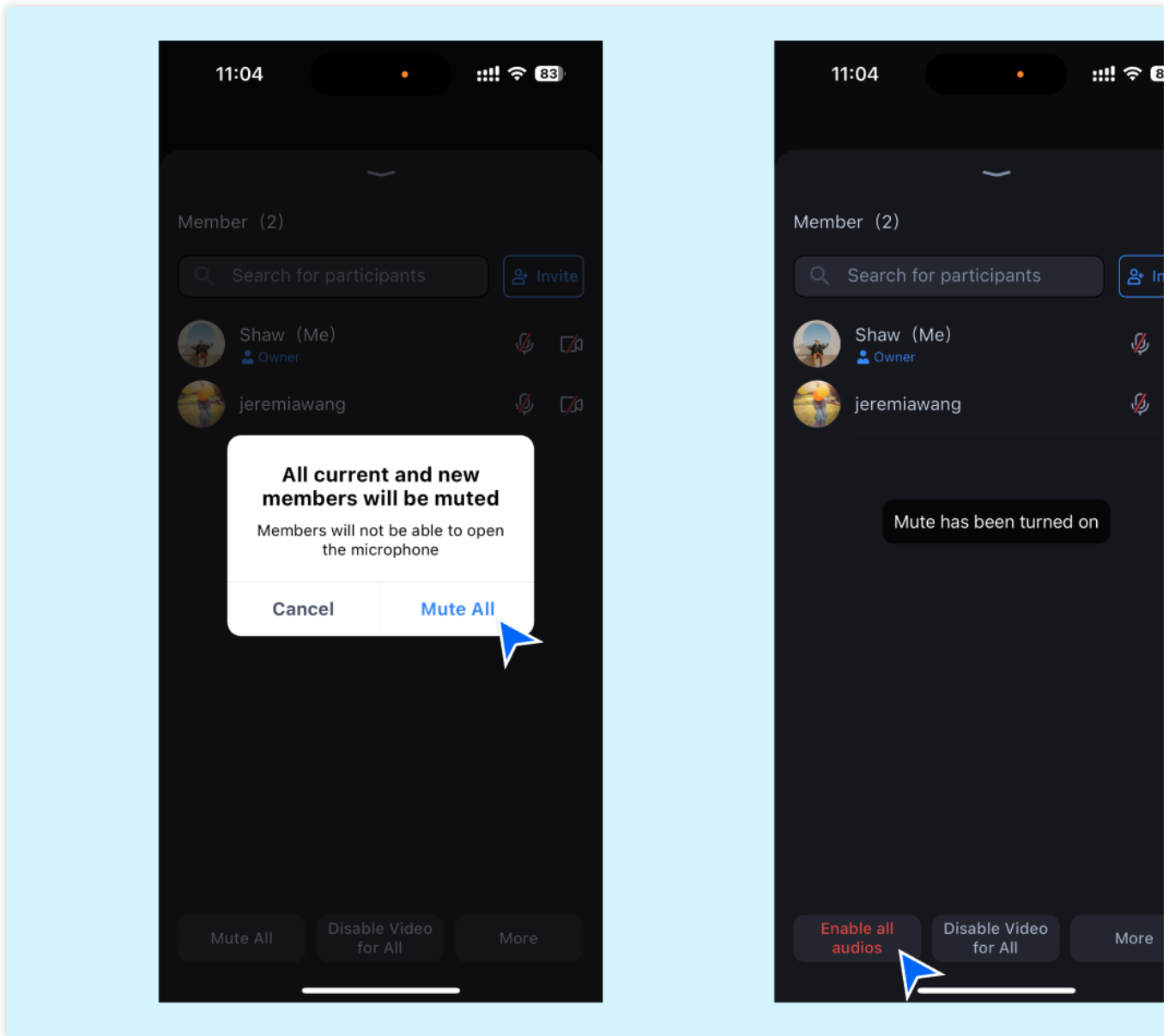
## Managing a Free-Speech Room

When the room type is a free-speech room, the host or administrator can manage all members in the meeting through **Members** > Member List.

The host or administrator can select any member for individual control: **unmute/mute**, **turn video on/off**, **set as an administrator**, **transfer host**, **disable/enable messaging**, or **kick out from the room**.



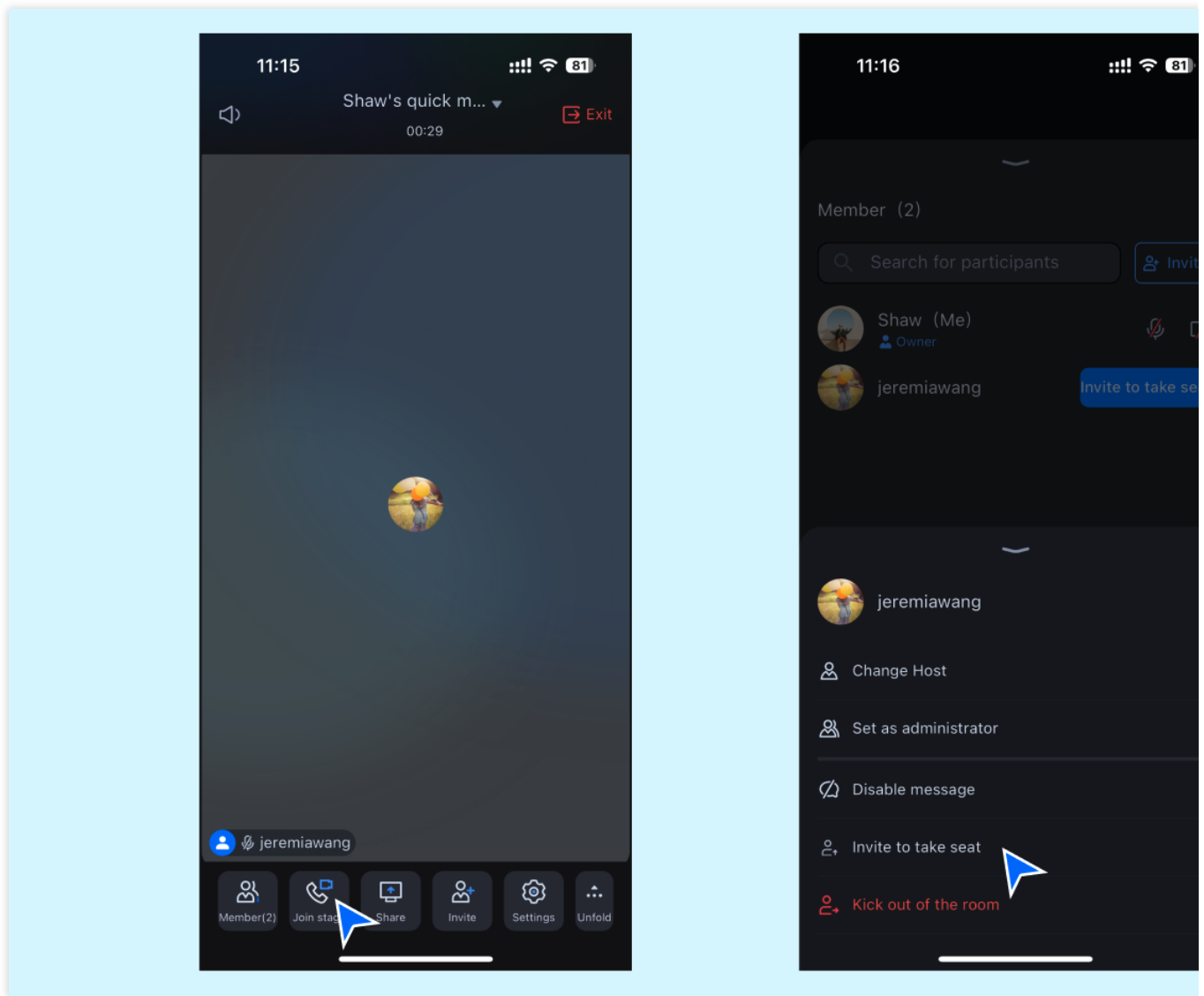
The host or administrator can perform group control actions for all members within the room: **mute/unmute all** or **disable/enable video for all**.



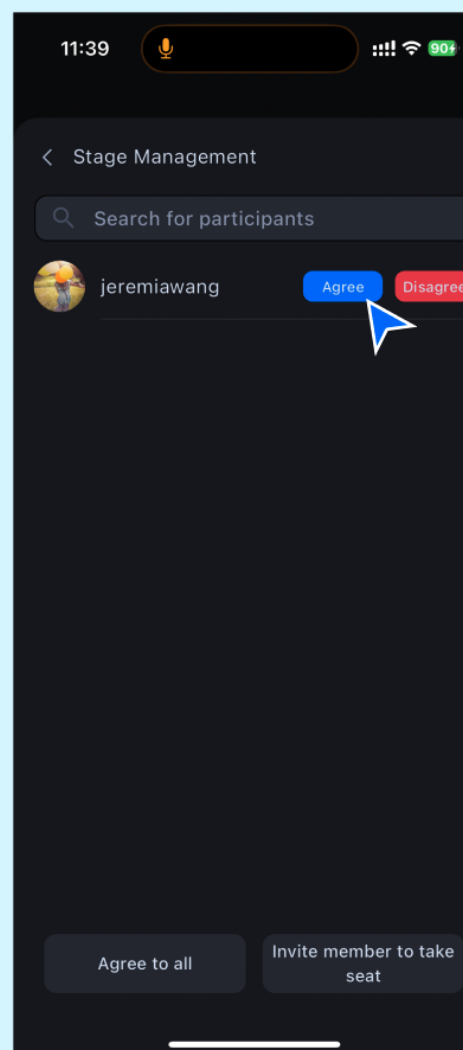
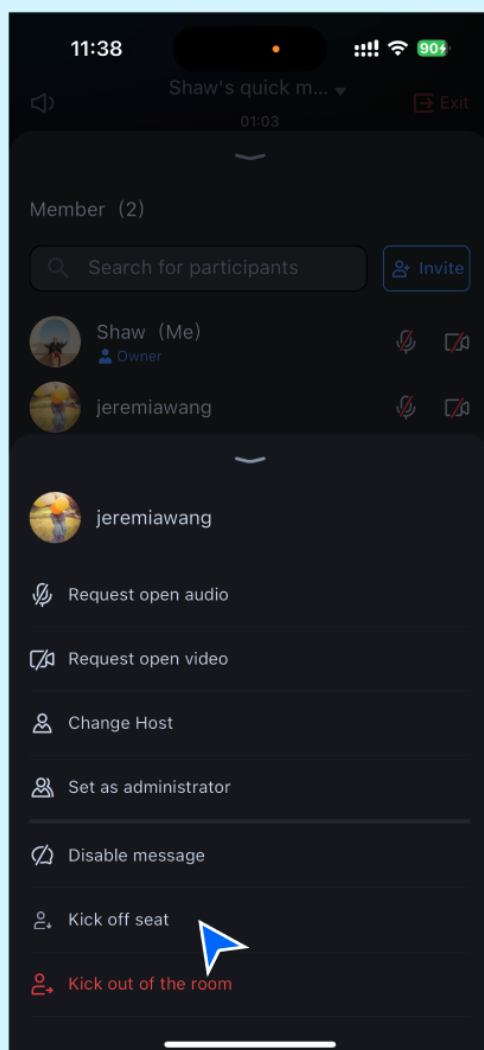
## Managing a Stage-Speech Room

When the room type is a stage-speech room, the host or administrator can manage the members in the meeting through **Members** > Member List, as well as manage the on-stage status of selected members in the Stage Management section.

The host or administrator can select any regular member for individual control: in addition to the actions available in the free-speech room, such as **unmute/mute**, **turn video on/off**, **disable/enable messaging**, **set as an administrator**, **transfer host**, and **kick out from the room**, stage-speech rooms also offer unique actions such as **inviting on stage** and **asking to leave the stage**.



The host or administrator can manage the status of members who have applied to be on stage within the room: in the **Stage Management** section, they can **approve or reject** selected members, or **approve all members** who have applied to be on stage.

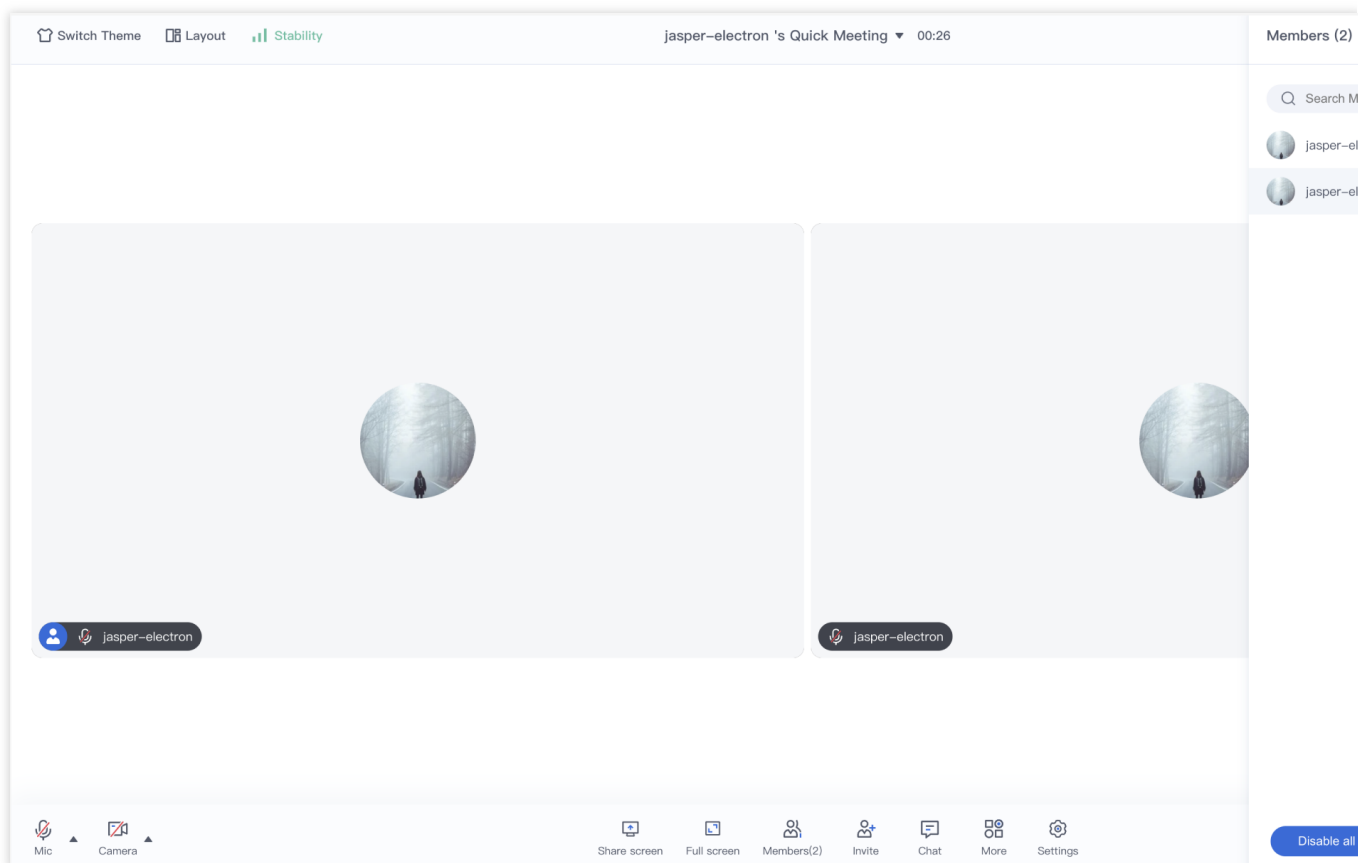


# Web&Electron

Last updated : 2024-05-29 15:06:57

This document will introduce `TUIRoomKit` conference control in detail, helping you to better master the operation of `TUIRoomKit` functions before and during the meeting. Through this document, you can make full use of `TUIRoomKit`'s features to achieve high-quality audio and video conferences.

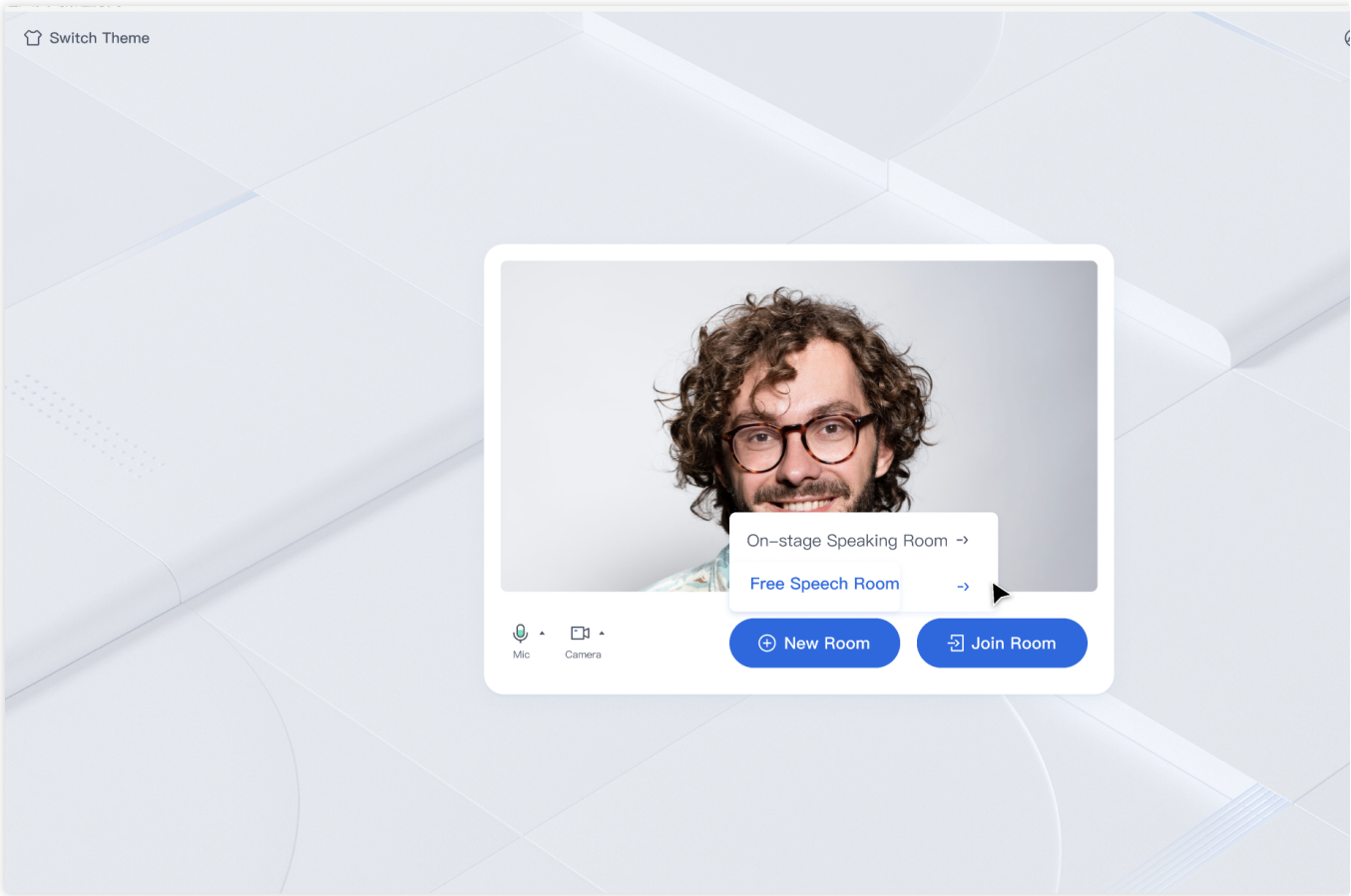
After a user creates and enters a room on `Web` and `Electron`, the owner or administrator can select any ordinary member in the member list popped up on the right side by clicking the **member** button on the toolbar at the bottom to perform the meeting control operations such as **banning**, **setting as administrator**, **etc.**, and can also perform the meeting control operations such as **mute all the members** in the room.

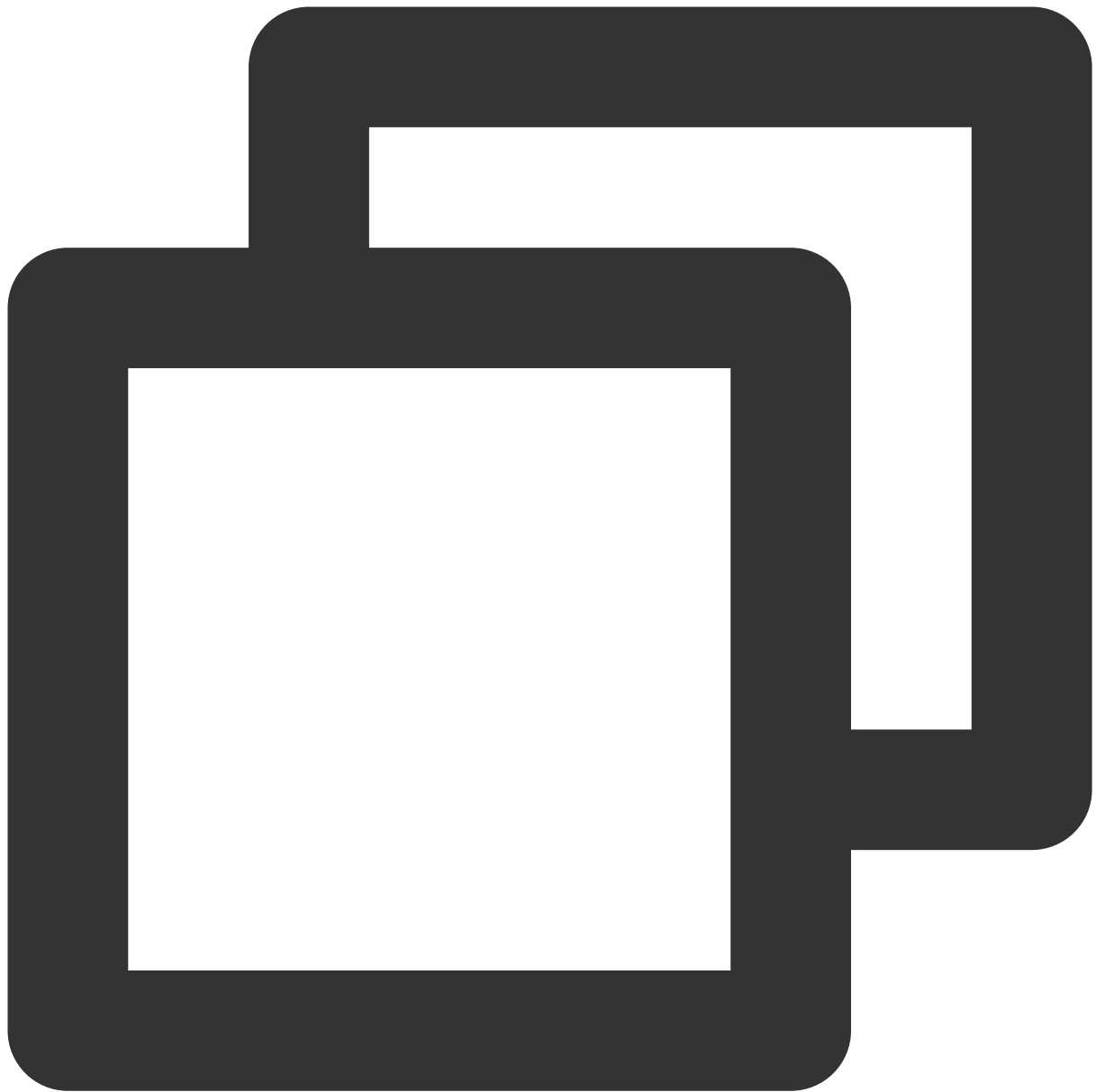


## Pre-meeting control

When creating and joining a room, you need to create and enter the room with pre-set parameters related to the meeting to ensure that the meeting runs smoothly through the relevant features of `TUIRoomKit` pre-meeting

control.





```
// Note the package name:  
// if you are using the vue2 version change the package name to @tencentcloud/roomkit-web-vue2  
import { conference } from '@tencentcloud/roomkit-web-vue3';  
conference.start('123456', {  
  roomName: 'TestRoom',  
  isSeatEnabled: false,  
  isOpenCamera: false,  
  isOpenMicrophone: false,  
});
```



Field	Type	Meaning
isSeatEnable	bool	Whether to open the room for on-stage speaking (default false)
isOpenCamera	bool	Whether to enable the camera (default true)
isOpenMicrophone	bool	Whether to turn on the microphone (default true)

**Note :**

The above is an introduction to creating rooms and adding room parameters in the above code. For example, depending on the value of the parameter `isSeatEnable`, you can create a free speech room and a stage speech room, in which the two types of rooms will have different features of the control and operable functions:

**Free Speech Room:** Ordinary users can speak freely and switch on and off the microphone and camera.

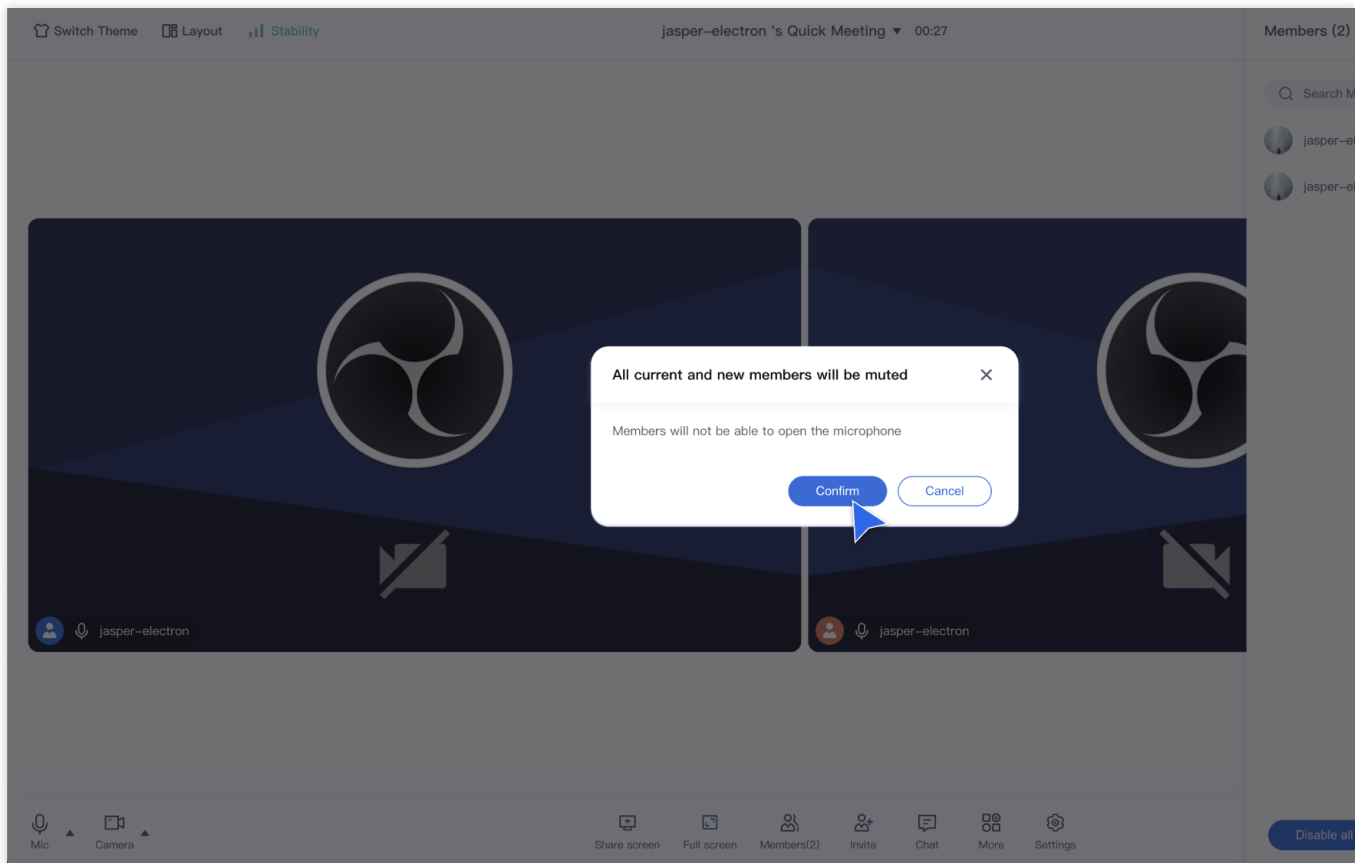
**Room for on-stage speakers:** Only on-stage users are free to switch the microphone and camera on or off, and ordinary audience members can apply to become on-stage users by raising their hands.

## in-session control

### Management of the Free Speech Room

When the room type is Free Speech Room, the room owner or administrator can manage all members of the club in **Members** > In Member List.

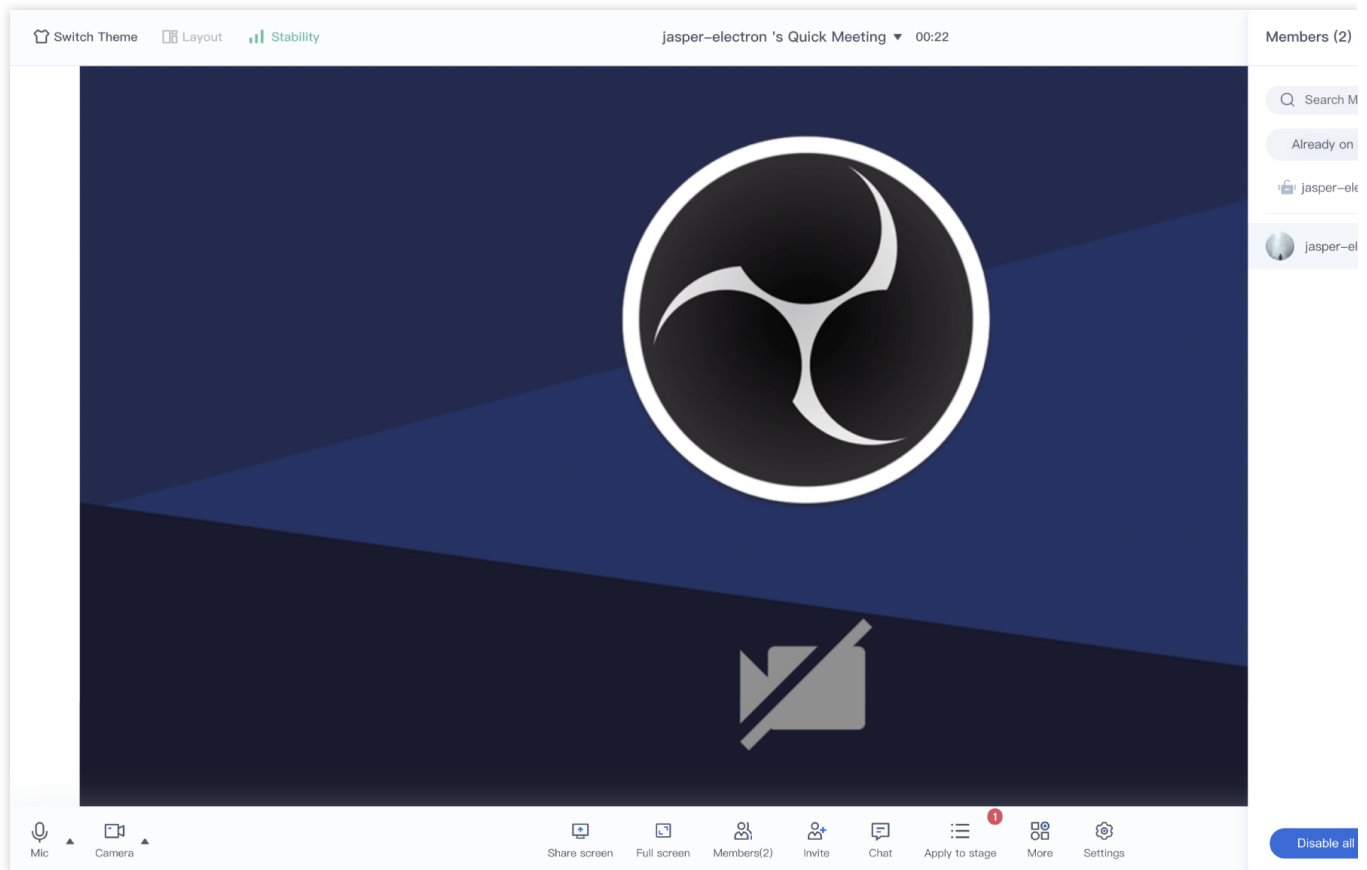
The host or administrator can select any member for individual control: **unmute/mute** them, as well as click more to **request video on/off**, **ban/unban**, **set as administrator**, **forward to the host**, **kick out of the room**, and more. The host or administrator can control all members of the room: **mute/unmute all** or **ban/unban all**.



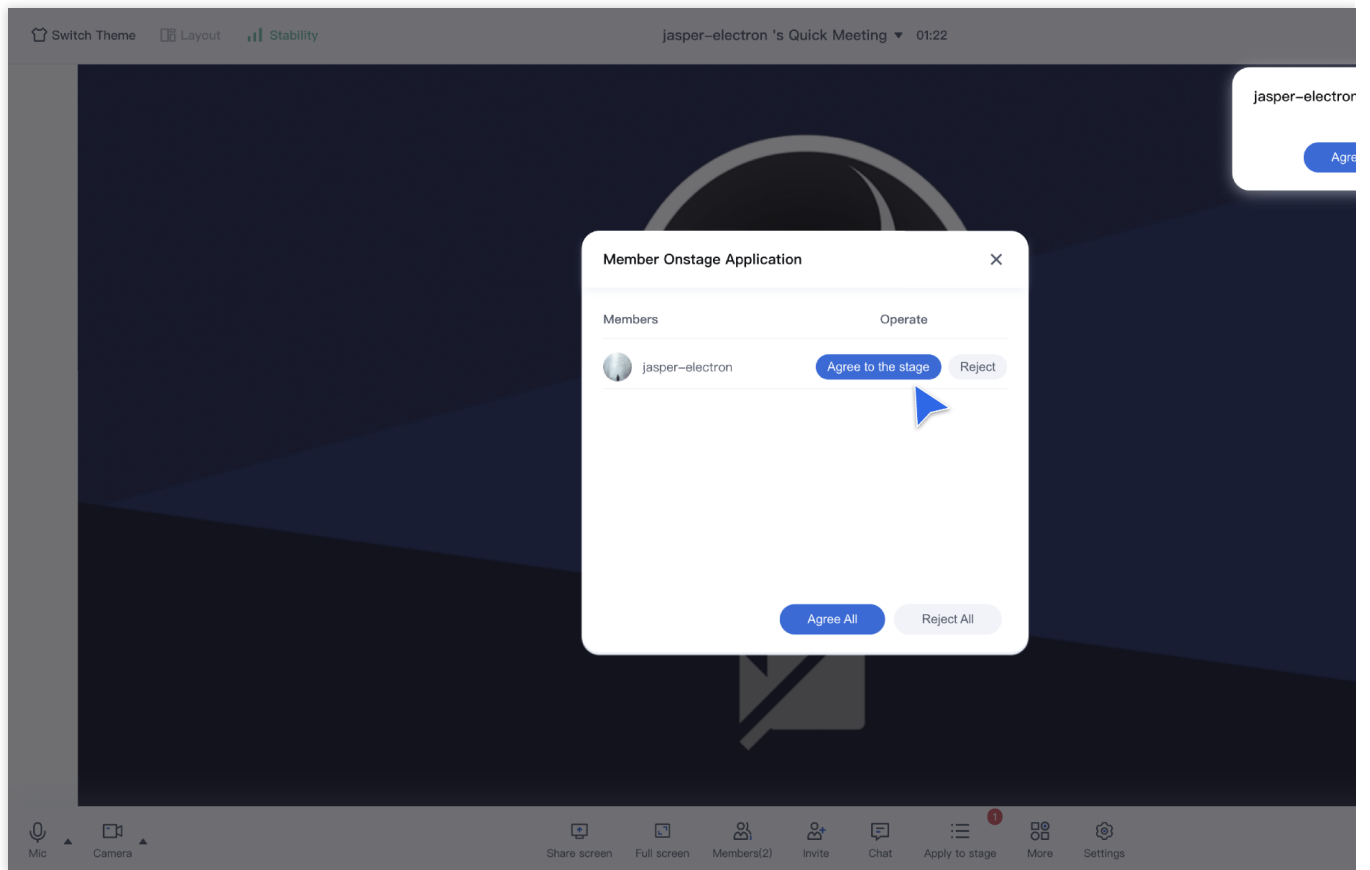
## Management podium room

When the room type is Speaker Room, the room owner or administrator can manage the on-stage status of the selected members in the On-stage request, in addition to managing the members of the meeting in the **Members** > Members list.

Moderators or administrators can select any ordinary member for individual control: in addition to **unmute/mute**, **turn on video/turn off video**, **ban/unban**, **set as administrator**, **transfer to moderator**, and **kick out of the room** contained in the free speech room, the invitation to go on stage, agree to go on stage/refuse to go on stage, and get off the stage, etc., which are unique to the room of the speaker on stage, are also available.



The host or administrator can manage the status of the members in the room who have **applied for on-stage application**: they can **agree** or **reject** the selected members in the on-stage application, or agree or reject all the members who have applied for on-stage application.

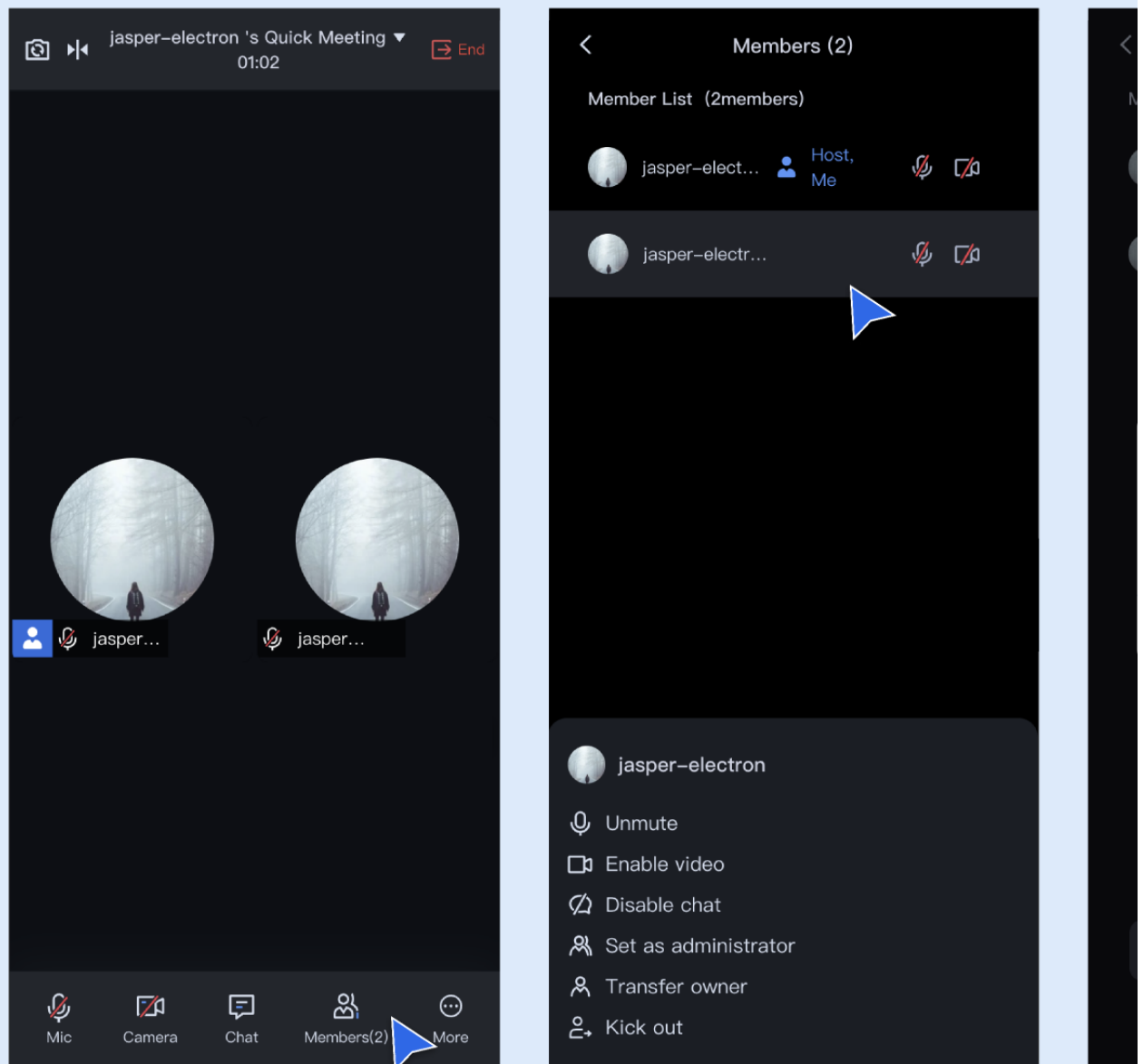


# H5

Last updated : 2024-05-29 15:06:32

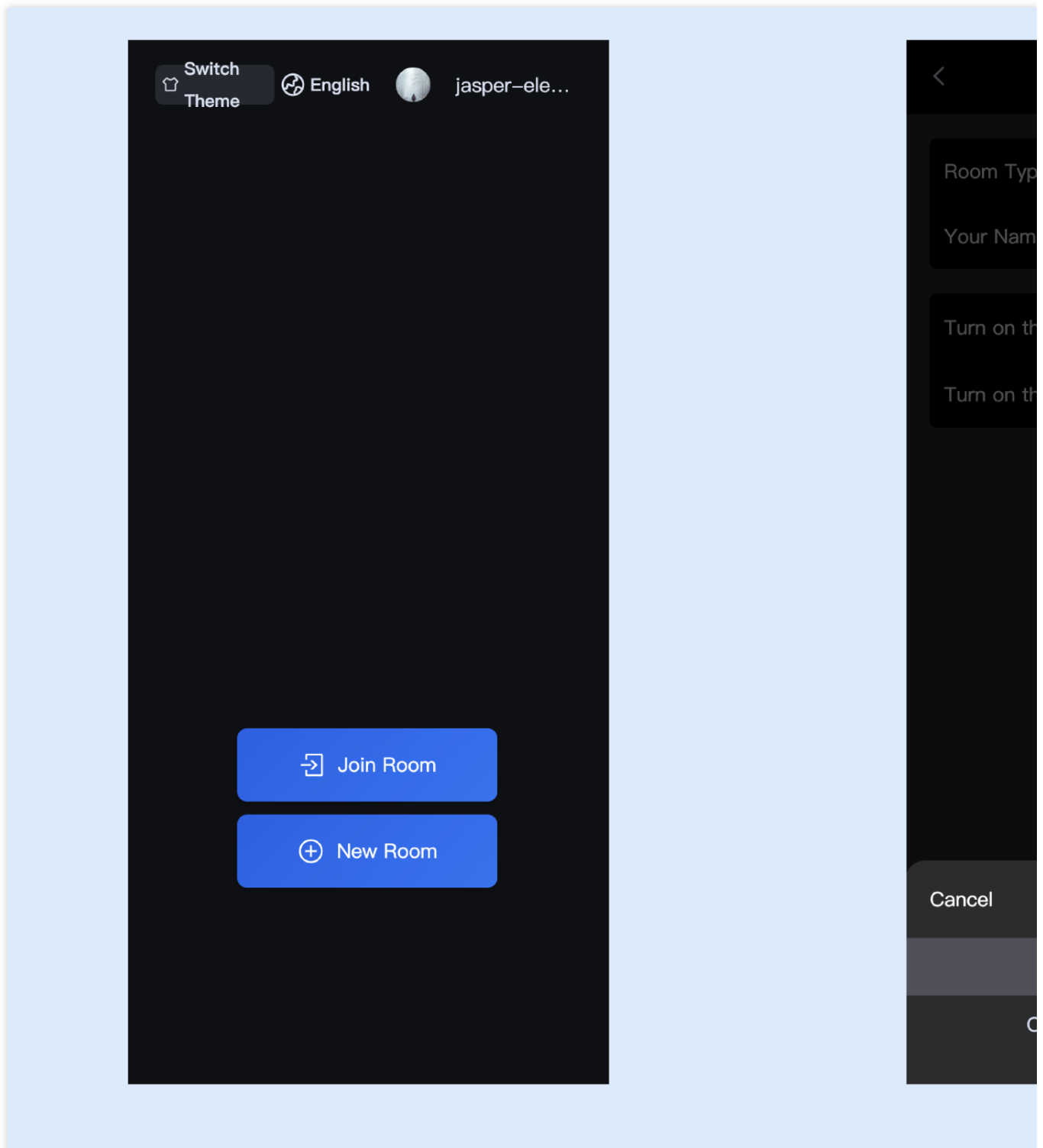
This document will introduce `TUIRoomKit` conference control in detail, helping you to better master the operation of `TUIRoomKit` functions before and during the meeting. Through this document, you can make full use of `TUIRoomKit`'s features to achieve high-quality audio and video conferences.

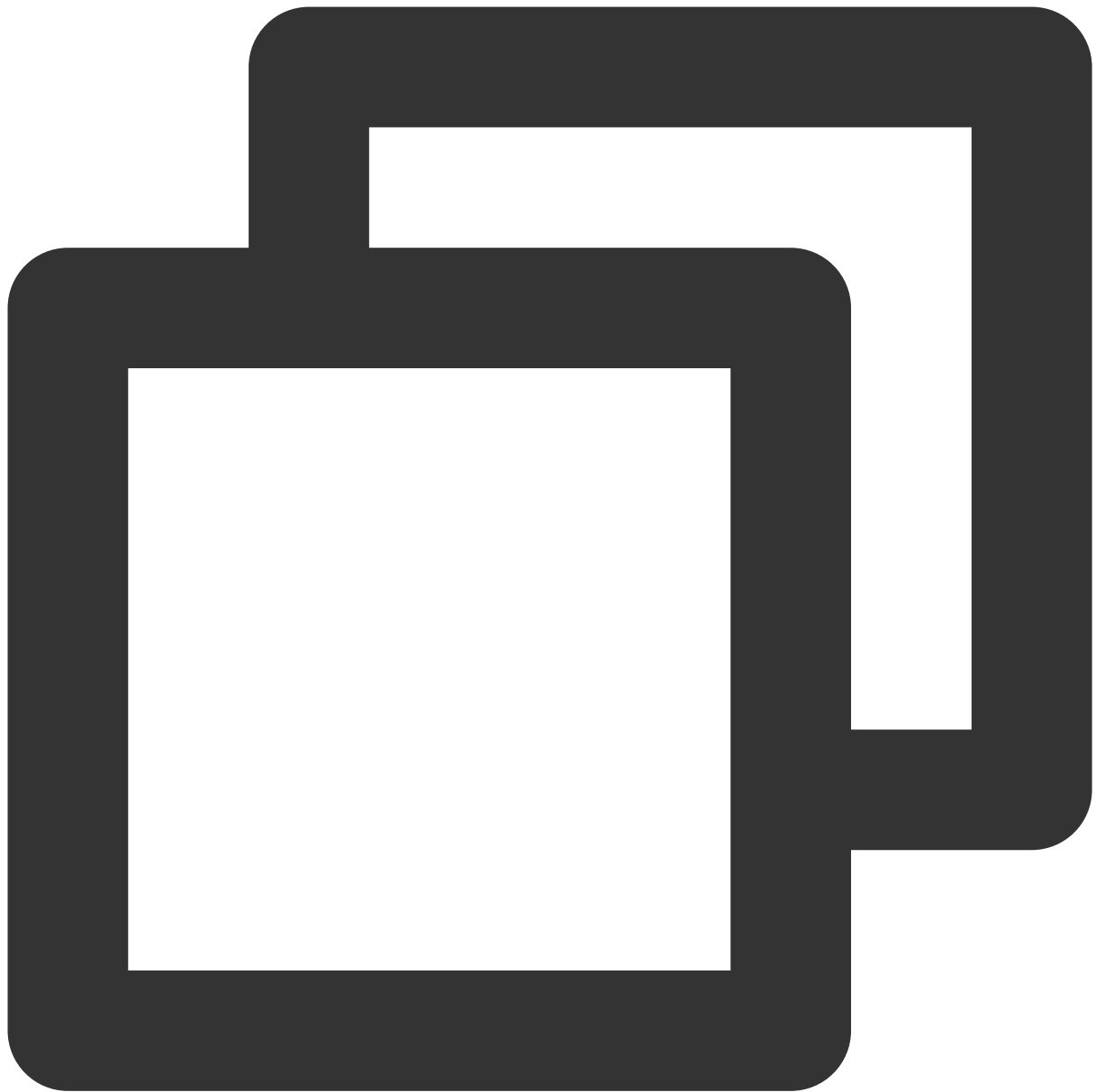
After a user creates and enters a room on `H5`, the owner or administrator can select any ordinary member in the member list popped up on the right side by clicking the **member** button on the toolbar at the bottom to perform the meeting control operations such as **banning, setting as administrator, etc.**, and can also perform the meeting control operations such as **mute all the members** in the room.



## Pre-meeting control

When creating and joining a room, you need to create and enter the room with pre-set parameters related to the meeting to ensure that the meeting runs smoothly through the relevant features of `TUIRoomKit` pre-meeting control.





```
// Note the package name:
// if you are using the vue2 version change the package name to @tencentcloud/roomkit-web-vue2
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```



Field	Type	Meaning
isSeatEnable	bool	Whether to open the room for on-stage speaking (default false)
isOpenCamera	bool	Whether to enable the camera (default true)
isOpenMicrophone	bool	Whether to turn on the microphone (default true)

**Note :**

The above is an introduction to creating rooms and adding room parameters in the above code. For example, depending on the value of the parameter `isSeatEnable`, you can create a free speech room and a stage speech room, in which the two types of rooms will have different features of the control and operable functions:

**Free Speech Room:** Ordinary users can speak freely and switch on and off the microphone and camera.

**Room for on-stage speakers:** Only on-stage users are free to switch the microphone and camera on or off, and ordinary audience members can apply to become on-stage users by raising their hands.

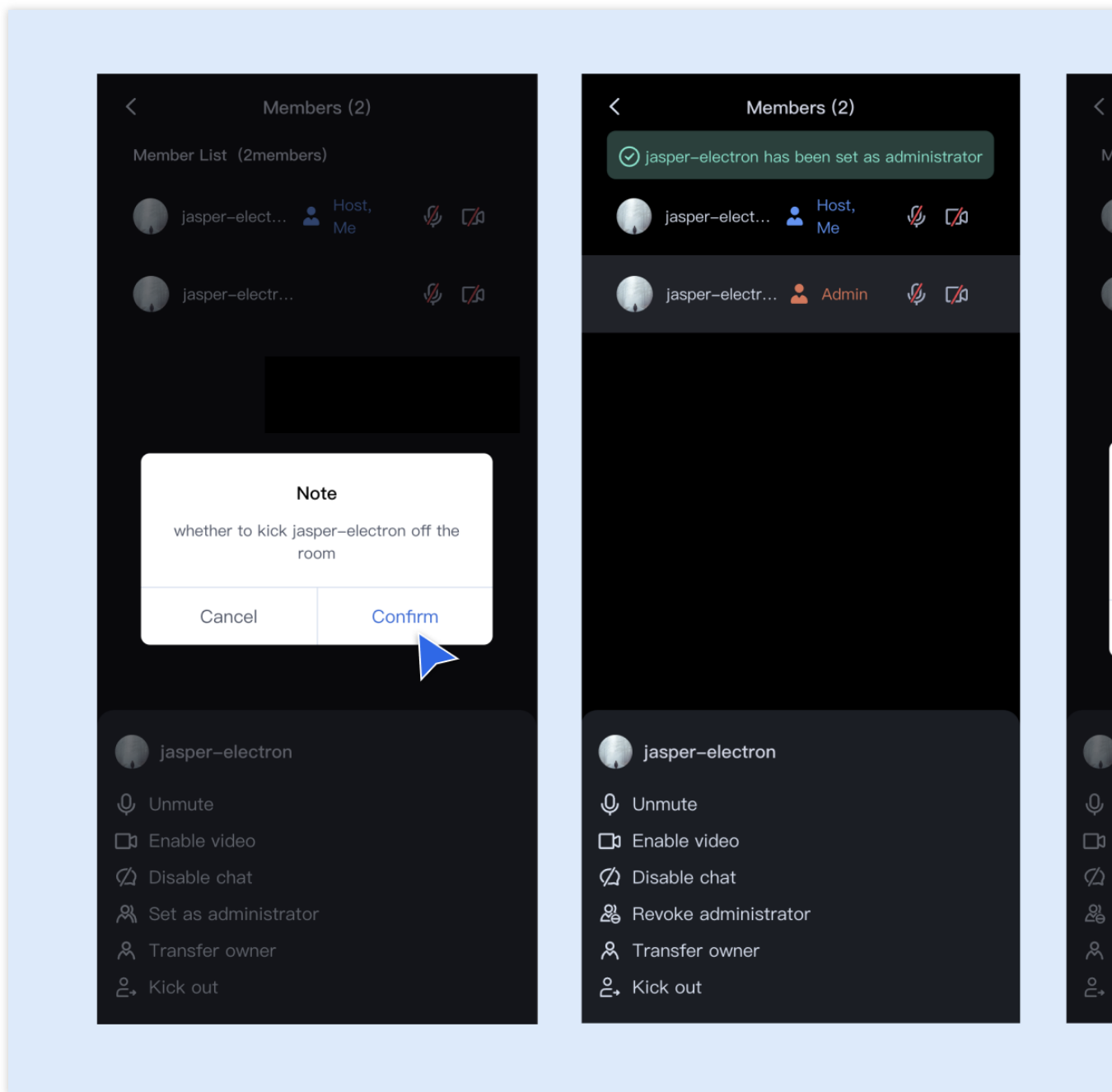
## in-session control

### Management of the Free Speech Room

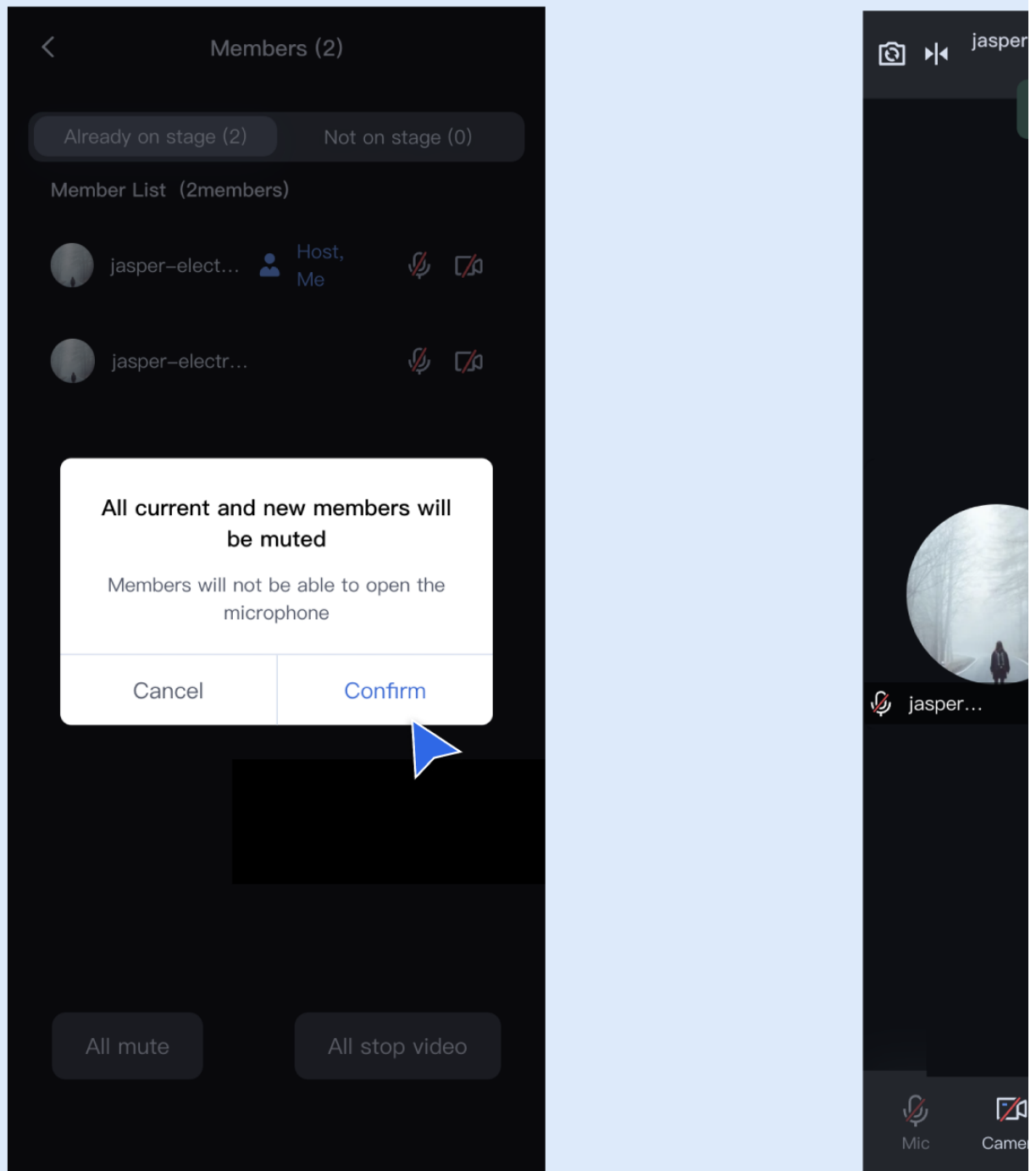
When the room type is Free Speech Room, the room owner or administrator can manage all members of the club in

**Members** > In Member List.

The host or administrator can select any member for individual control: **unmute/mute** them, as well as click more to **request video on/off**, **ban/unban**, **set as administrator**, **forward to the host**, **kick out of the room**, and more.



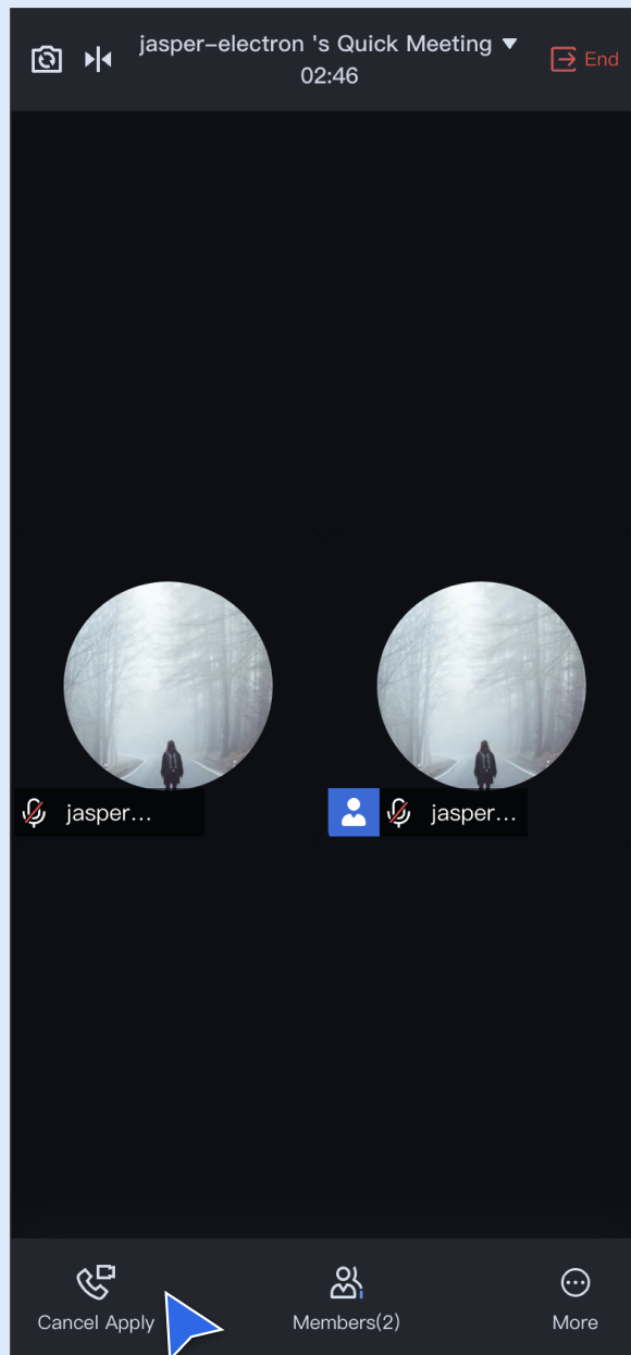
The host or administrator can control all members of the room: **mute/unmute all** or **ban/unban all**.



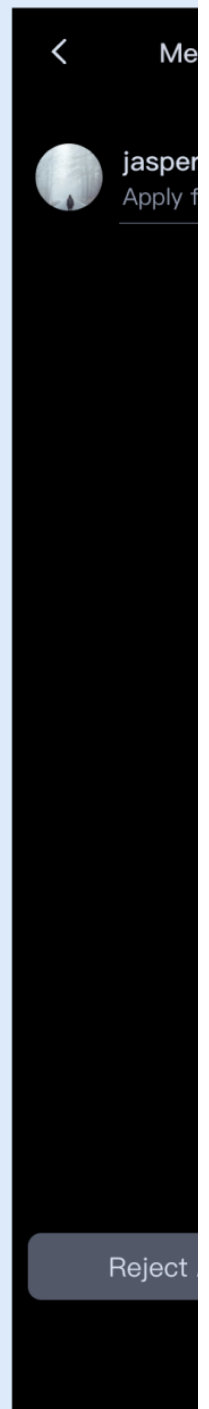
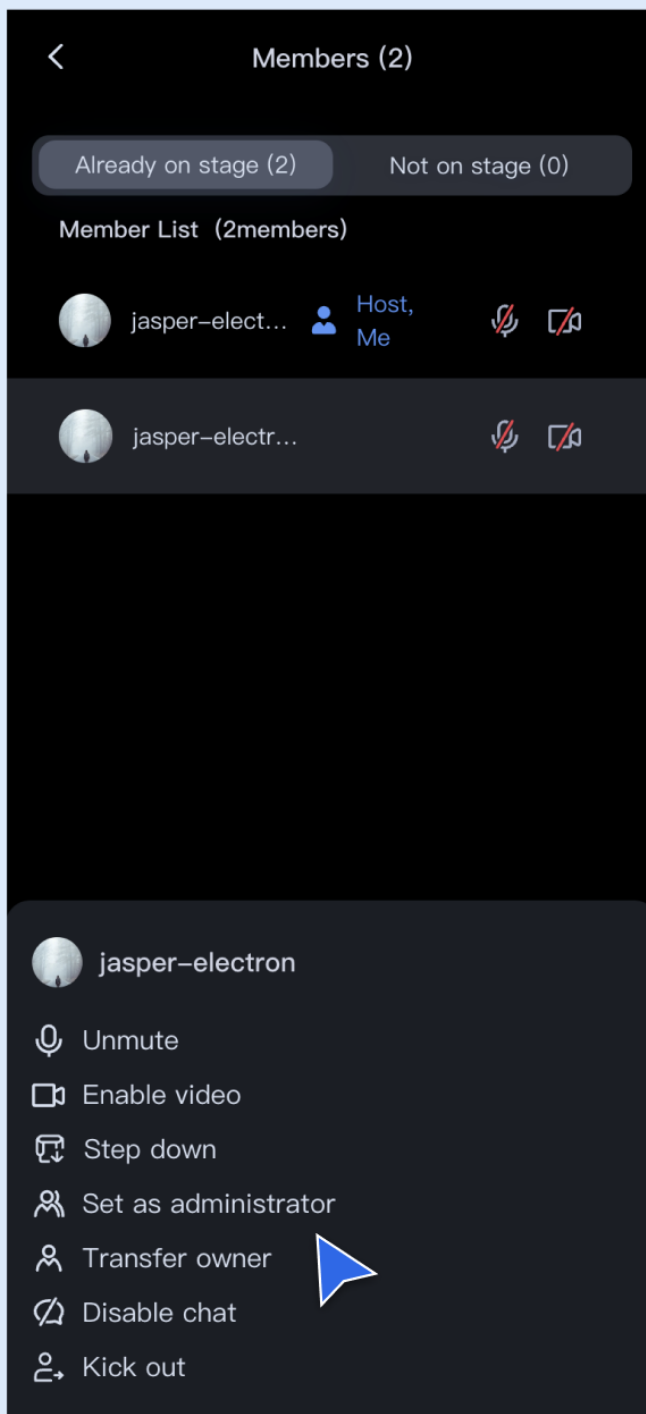
## Management podium room

When the room type is Speaker Room, the room owner or administrator can manage the on-stage status of the selected members in the On-stage request, in addition to managing the members of the meeting in the **Members** > Members list.

Moderators or administrators can select any ordinary member for individual control: in addition to **unmute/mute**, **turn on video/turn off video**, **ban/unban**, **set as administrator**, **transfer to moderator**, and **kick out of the room** contained in the free speech room, the invitation to go on stage, agree to go on stage/refuse to go on stage, and get off the stage, etc., which are unique to the room of the speaker on stage, are also available.



The host or administrator can manage the status of the members in the room who have **applied for on-stage application**: they can **agree** or **reject** the selected members in the on-stage application, or agree or reject all the members who have applied for on-stage application.





# Cloud Recording (TUIRoomKit)

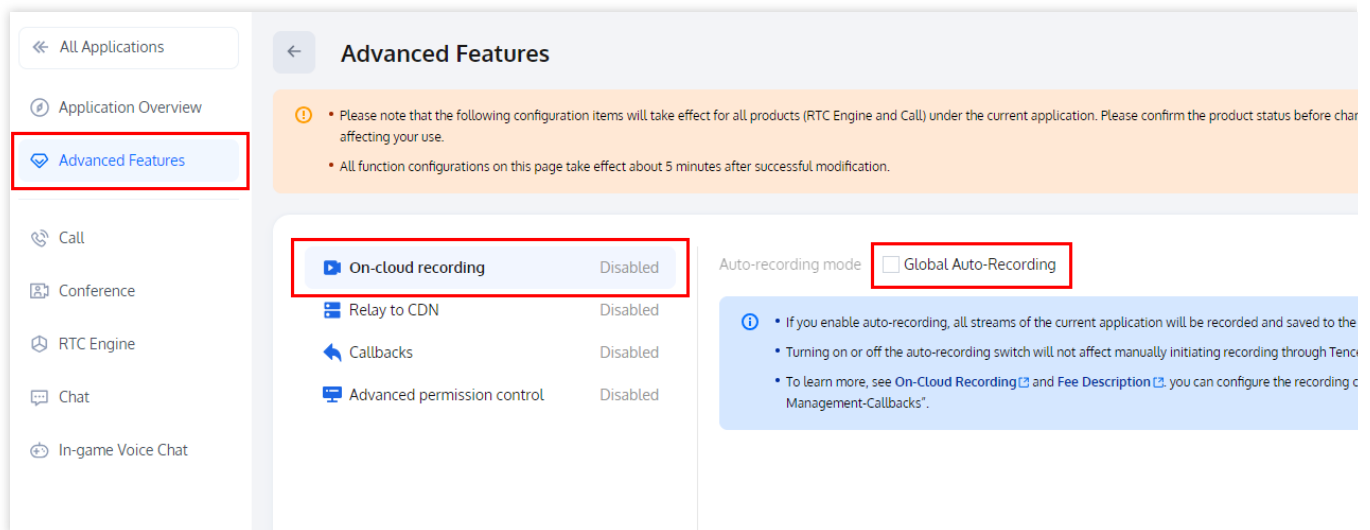
Last updated : 2024-04-17 18:02:37

This article introduces how to quickly use TUIRoomKit's cloud recording feature to help developers achieve diversified needs such as archiving and reviewing important content in video conferencing, online education, live interactive, and other scenarios. We provide two solutions for you to choose from: [Automatic Recording](#) and [RESTful API-Based Recording](#).

## Scheme 1. Automatic Recording (Recommended)

We recommend using the automatic recording solution, which does not require the business side to start and stop the recording. The Tencent Cloud Real-Time Audio and Video backend manages the recording tasks, and it automatically records when there is audio and video stream uploading during the call. The access process is fast and simple. You can complete it in the following steps:

1. Find the application of the target `SDKAppId` in [Application Management](#) in the TRTC console and enter the **Advanced Feature** page.
2. In the **Advanced Function** configuration page, you can see the option for **On-cloud recording** configuration. Click on **Global Auto-Recording** to enter the configuration popup window.



3. In the configuration popup window, You can customize the recording template as needed. ◦

## Global auto recording

**i** After you enable global auto recording, TRTC will automatically record the streams in all newly created rooms according to the settings. Recording fees are based on the duration of the recording file generated.

### Parameter \*

Recording type ☒ Video ☐ Audio

File format ☒ MP4 ☐ HLS

Max file duration  min

Files exceeding 2GB in size will be split.

Timeout period for resumption **i**  s

The longer the waiting time for continuous recording, the longer the recorded file will be.

Remove audio **i** ☐

### Storage \*

### Callback

Protocols for callback URLs: HTTP and HTTPS up to 2083 characters

Submit

Cancel

### Note :

Global automatic recording only supports [single-stream recording \(i.e., each host records a single file\)](#). Once enabled, it only applies to newly created rooms and does not take effect for rooms created before the automatic recording feature is enabled. If you need to record the screen after mixing multiple streams, please use the [RESTful API-Based recording](#).

Global automatic recording supports recording up to 25 hosts in a room. If there are more than 25 hosts, they will be sorted by the time they enter the room, and the first 25 hosts will be recorded (if you need to record more than 25 hosts in a single stream, please refer to the [RESTful API-Based recording](#)).

After enabling the global automatic recording feature, the automatic start recording task will be triggered after the conference starts and there is audio and video uploading. After the conference ends, the recording will automatically stop. If you leave the room abnormally due to network or other situations, our recording backend will automatically

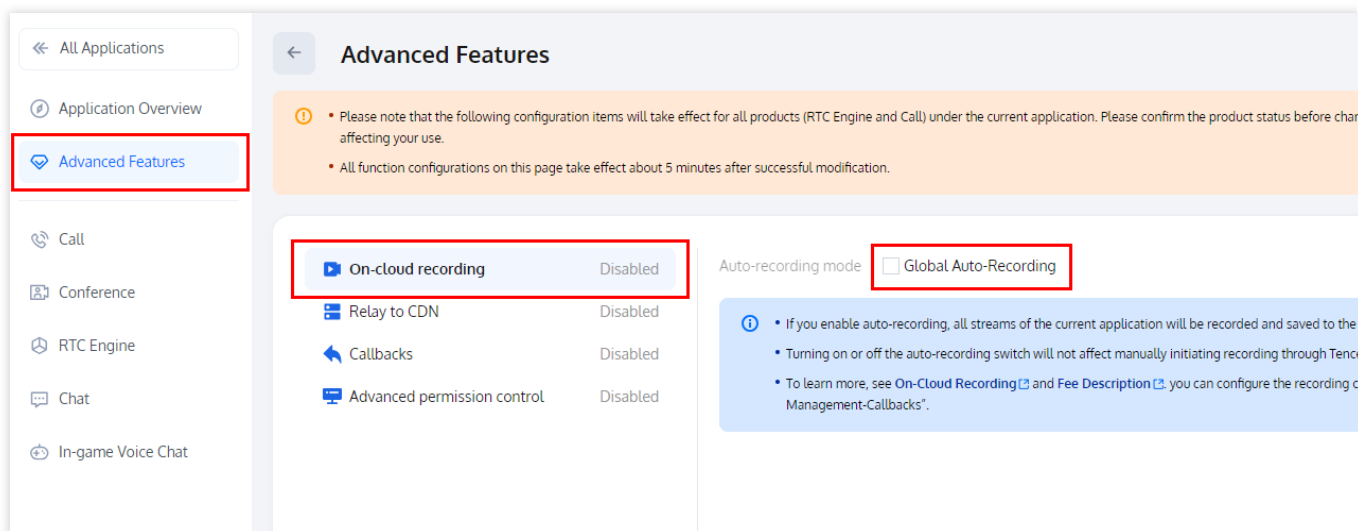


stop the recording task according to the `MaxIdleTime` value you set (idle waiting time, default 5s) to avoid causing additional billing losses.

## Scheme 2. RESTful API-Based Recording

If the automatic recording scheme doesn't meet your needs, you can also use the more flexible RESTful API-based recording scheme. In this scheme, you can record and subscribe to a specified anchor in the room, customize the layout of mixed streams, and update the layout and subscription during recording. However, **using its features requires using it together with the business backend service and performing complex integration operations:**

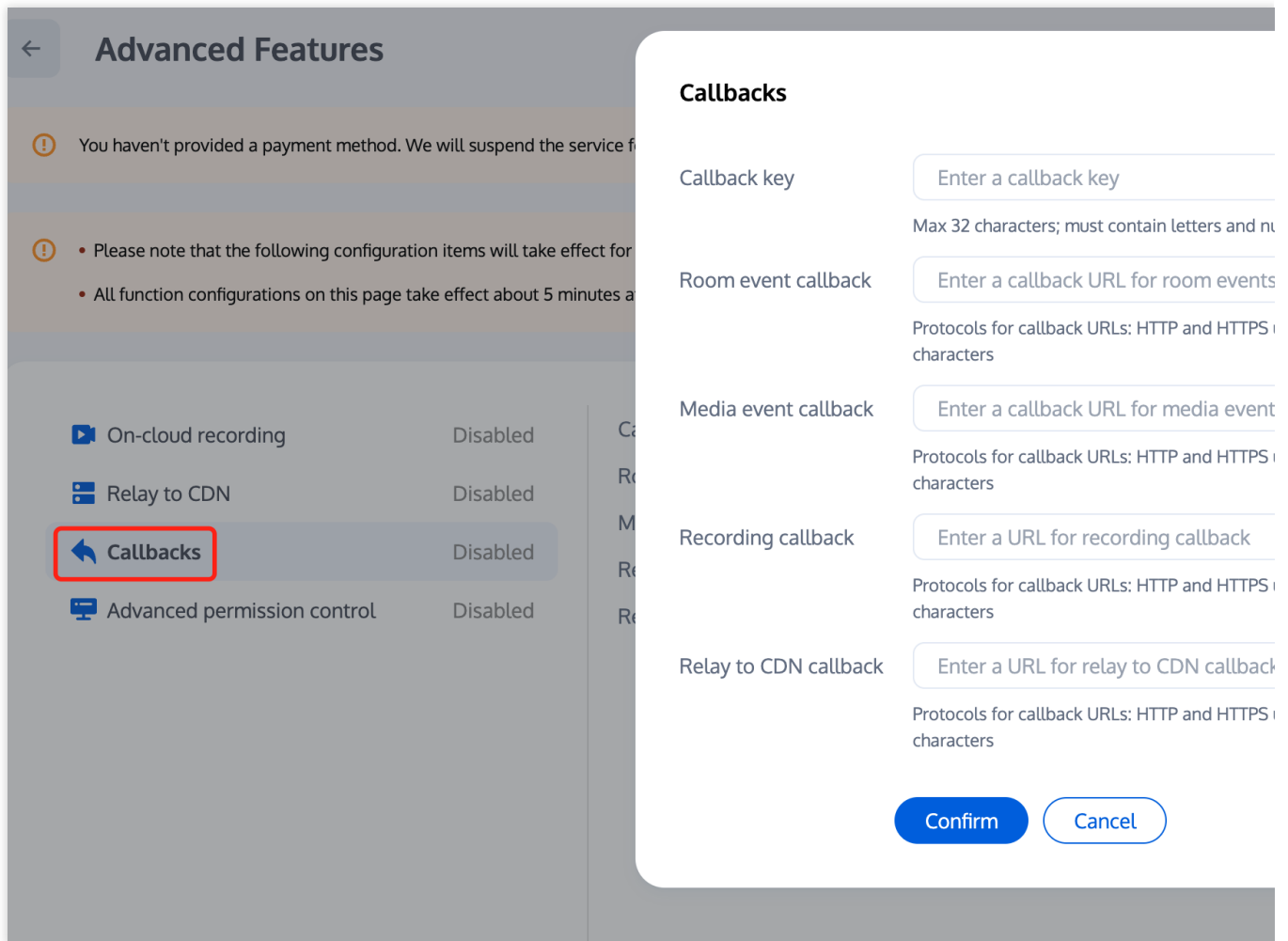
1. Find the application of the target `SDKAppId` in [Application Management](#) in the TRTC console and enter the **Advanced Feature** page.
2. On the Advanced Features configuration page, you can see the options for cloud recording configuration. Enable the cloud recording feature. Here, it is recommended to turn off Global Automatic Recording, which defaults to Manual Custom Recording, that is, RESTful API-based recording mode.



3. Afterward, you can call the REST API ([CreateCloudRecording](#)) to start the cloud recording. Here, it is recommended that you can listen to the notification events of [TUIRoomObserver](#) and start recording when the meeting starts.

### Note:

**Under manual recording, you can go to the console to configure the callback address to receive recording callback events**, please see [Recording Callback Description](#).



## FAQs

### 1. Can the two recording solutions be used simultaneously?

The [Automatic Recording Solution](#) and [REST API Recording Solution](#) do not conflict and can be used simultaneously. However, two recording files and [fees](#) will be generated.

### 2. How do I view recorded files?

Log in to the [VOD console](#), select **Video/Audio Management** on the left sidebar, click **Search by prefix** above the list, select **Search by prefix**, and enter the keyword in the search box. The recording filenames are in the following formats:

The filename format of an MP4 single-stream recording file:

```
<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Index>.mp4
```

The filename format of an MP4 mixed-stream recording file: `<SdkAppId>_<RoomId>_<Index>.mp4`

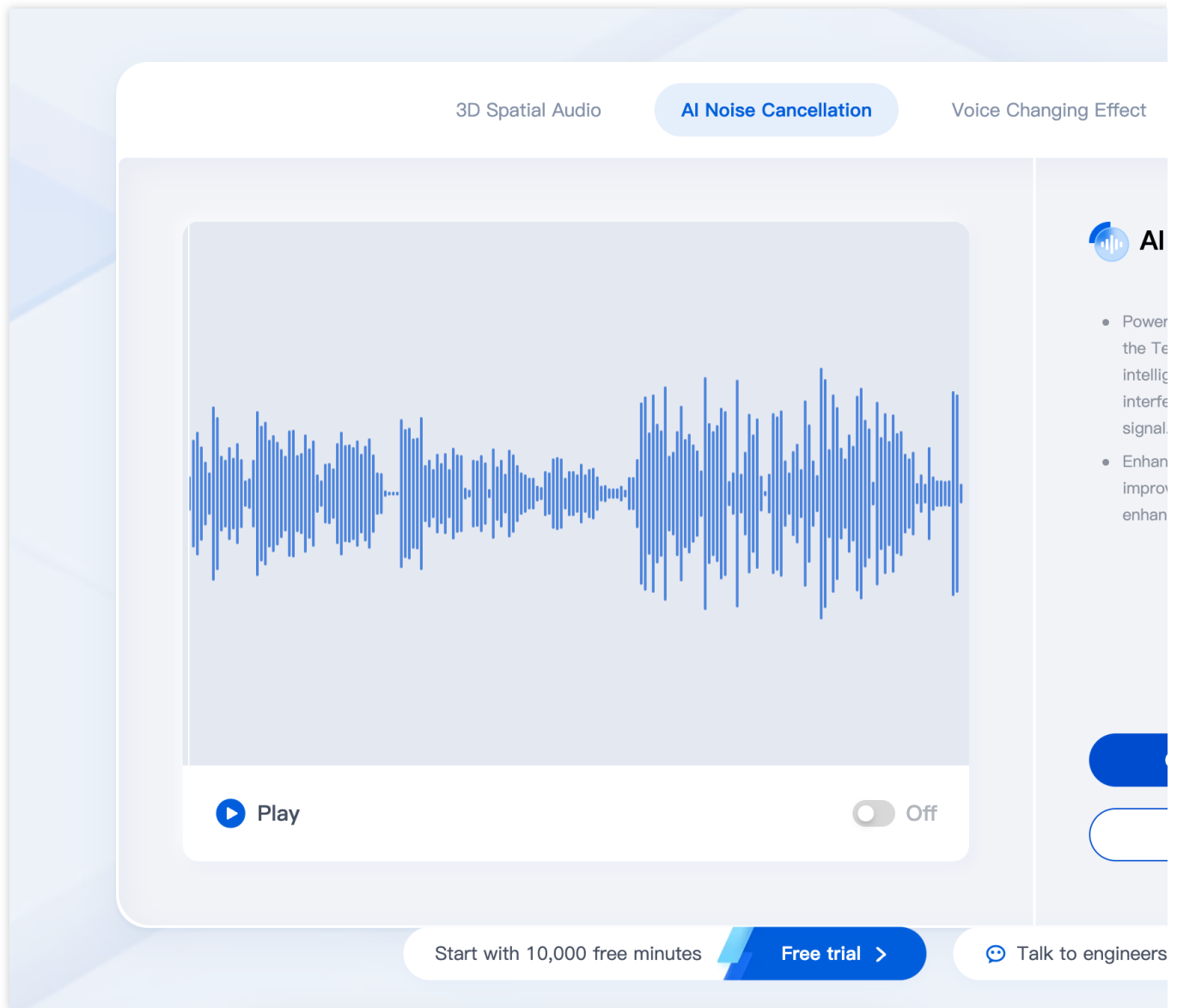
# AI Noise Reduction (TUIRoomKit)

Last updated : 2024-06-25 16:55:44

AI Noise Suppression, derived from Tencent's Tianlai Labs AI algorithms, intelligently detects and eliminates noise interference mixed in the transmission signal. Thus, it significantly improves the quality of voice, enhances sound clarity, and improves user listening experience. The application of AI Noise Suppression features in TUIRoomKit allows users to obtain a clear and stable sound experience in various environments such as offices, internet cafes, malls, and outdoors.

## Trying It Online

You can also enter our [TRTC Experience Pavilion](#) to try the excellent sound effects brought by AI Noise Suppression online.



## Activate AI Noise Suppression

TUIRoomKit now comes with AI Noise Suppression functionality enabled by default. Users can enjoy high-quality noise suppression within their applications without the need for additional configurations or actions.

# In-Conference Chat (TUIRoomKit) Web&Electron

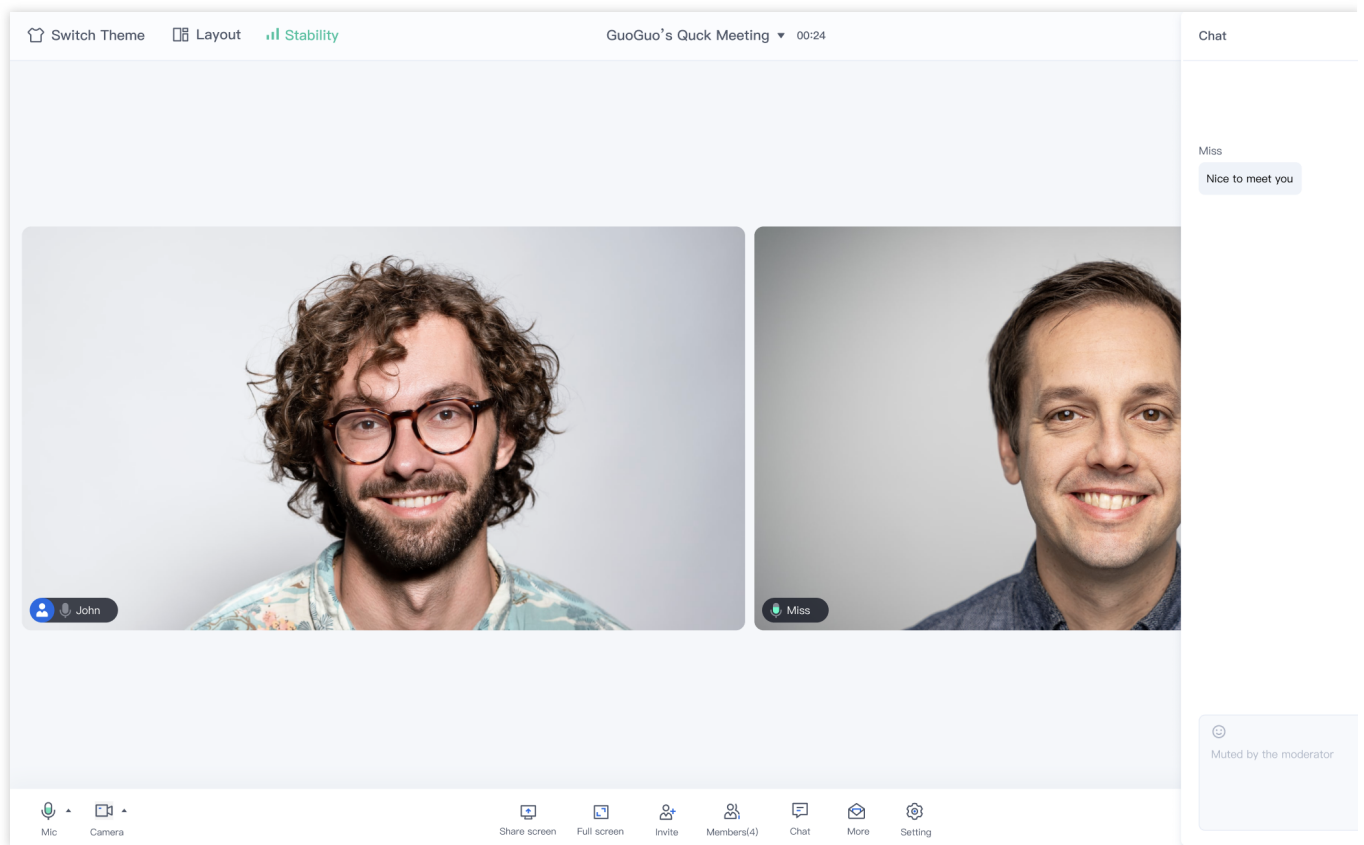
Last updated : 2024-07-03 17:14:46

In video conferences, participants can send messages in real-time in the chat area, share opinions and ideas, and create a relaxed and pleasant communication environment by exchanging emoticons and animations. To maintain the order of the meeting, the host or administrator can set to prohibit participants from sending messages in the chat, ensuring the focus and efficiency of the conference content. By flexibly using these features, video conferences can provide efficient and convenient communication experiences for various scenarios.

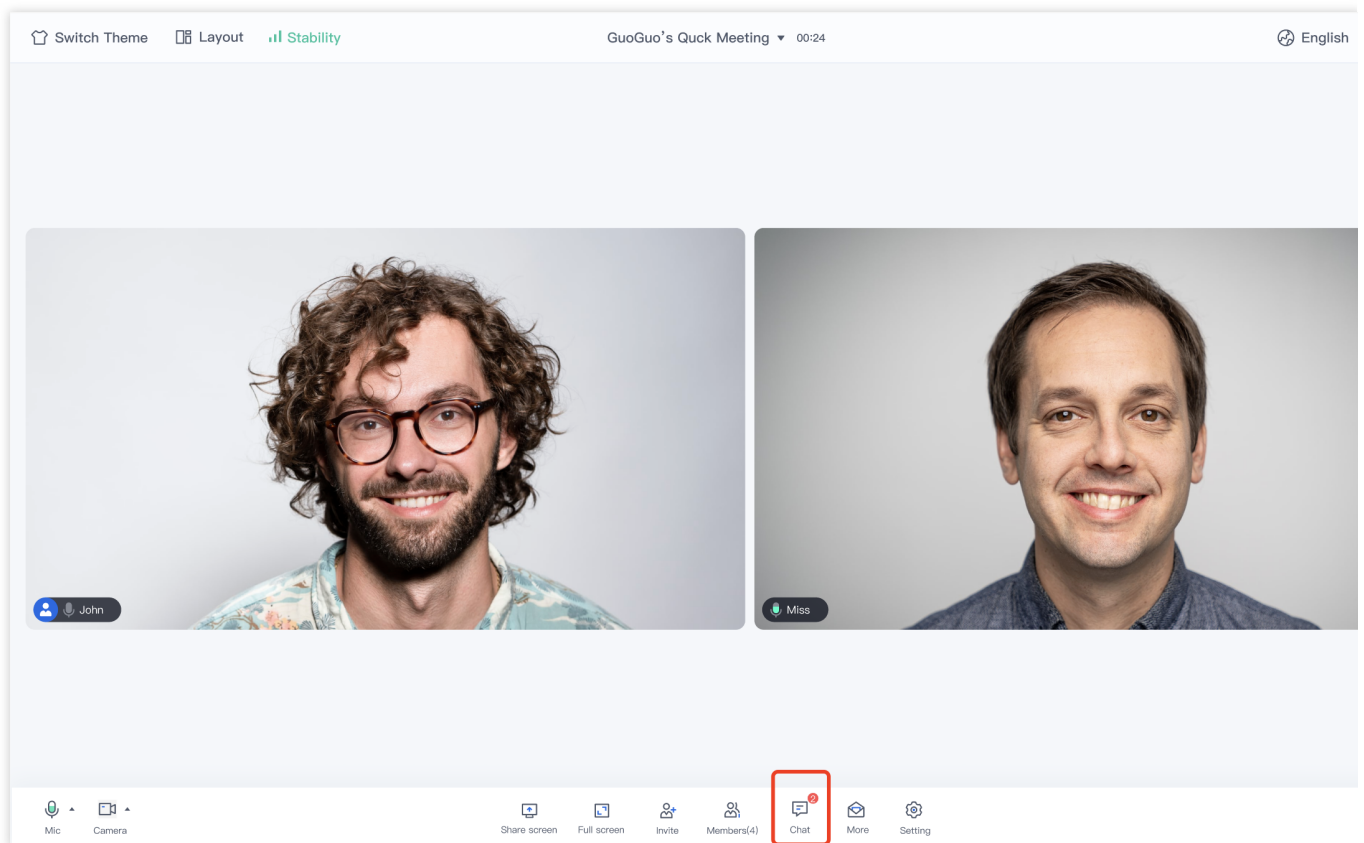
## Feature Introduction

### Text Interaction

At the bottom menu bar of the conference interface, you can find an option called **Chat**. Click on it to expand the chat box located on the right side of the screen. In the chat box, participants can communicate with other participants by sending custom text messages. This design facilitates real-time communication between participants without affecting the speaker.

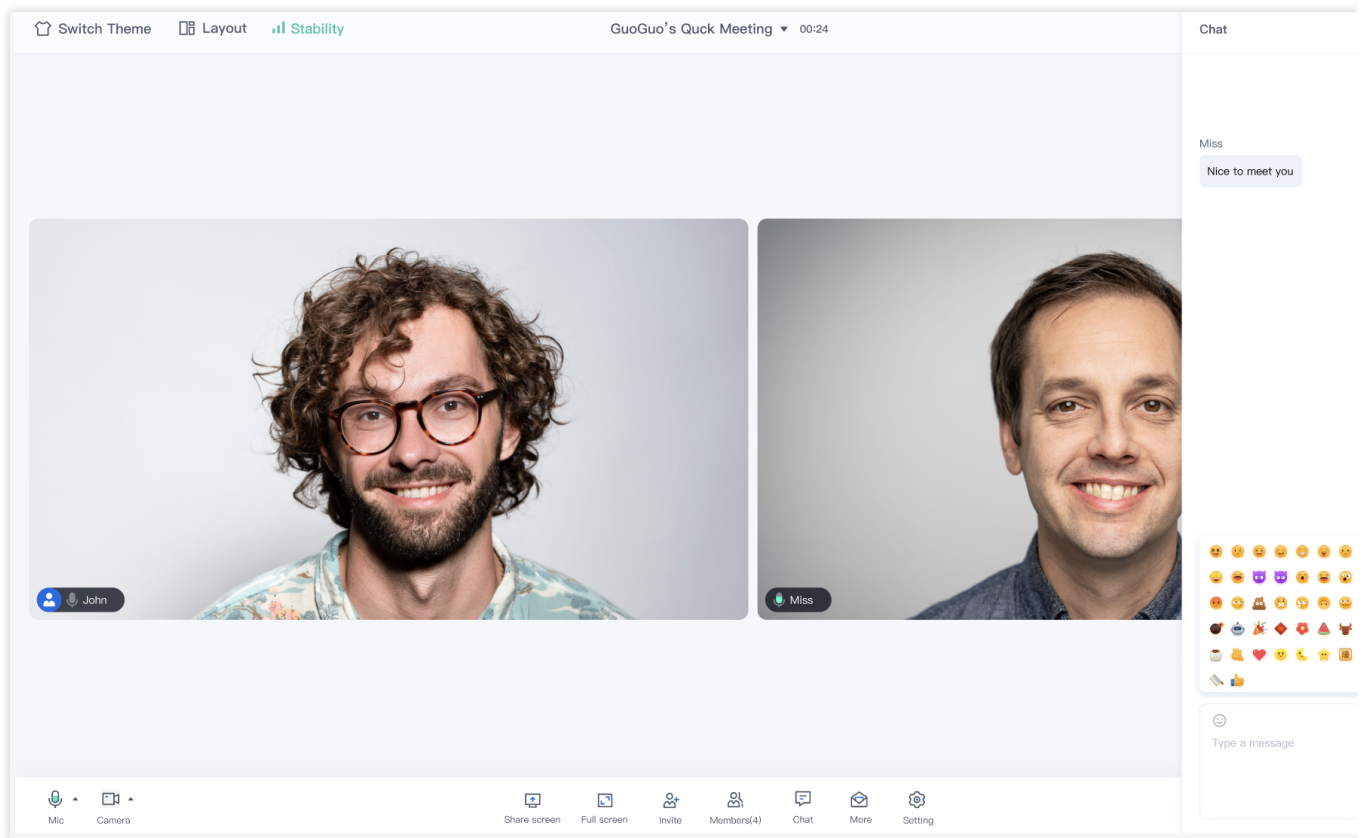


When a new chat message is sent, a red bubble prompt will appear in the upper right corner of the **Chat** on the bottom toolbar, reminding you of unread messages. This way, you can always keep track of real-time dynamics in the conference and ensure that you don't miss any important information.



## Emoticon Interaction

Click the **emoticon icon** in the chat interface's message editing bar to display the emoticon list. Click the corresponding emoticon to send it and give a thumbs-up to the speaker's brilliant speech.

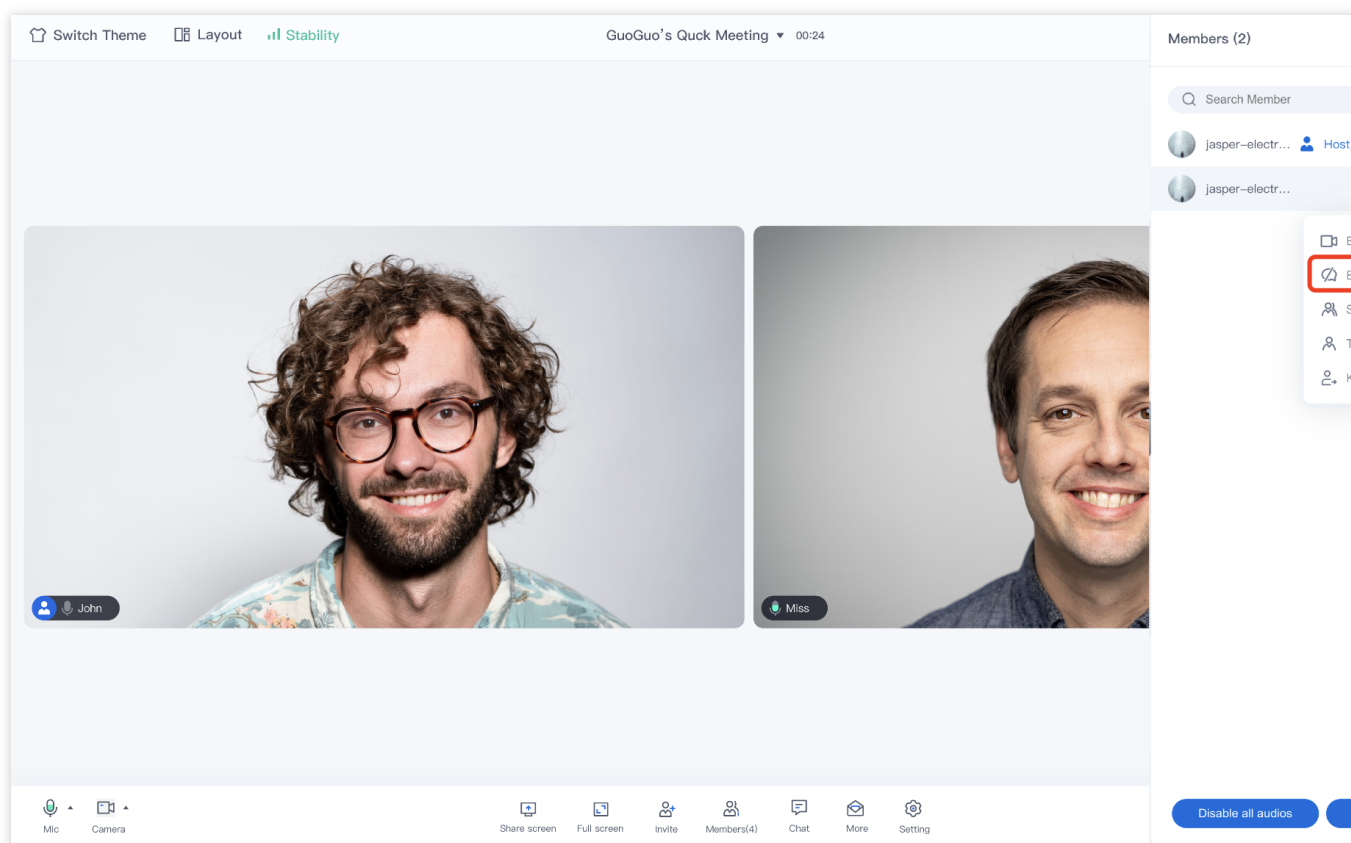
**Note :**

To respect the emoji design copyright, the Conference project does not include large emoji element slices. Before the official commercial launch, please replace them with your own designed emoji or other emoji packs that you own the copyright to. The default **Little Yellow Face Emoji Pack** copyright belongs to **Tencent Cloud** and can be licensed for a fee. If you wish to obtain a license, you can [Submit a ticket](#) to contact us.

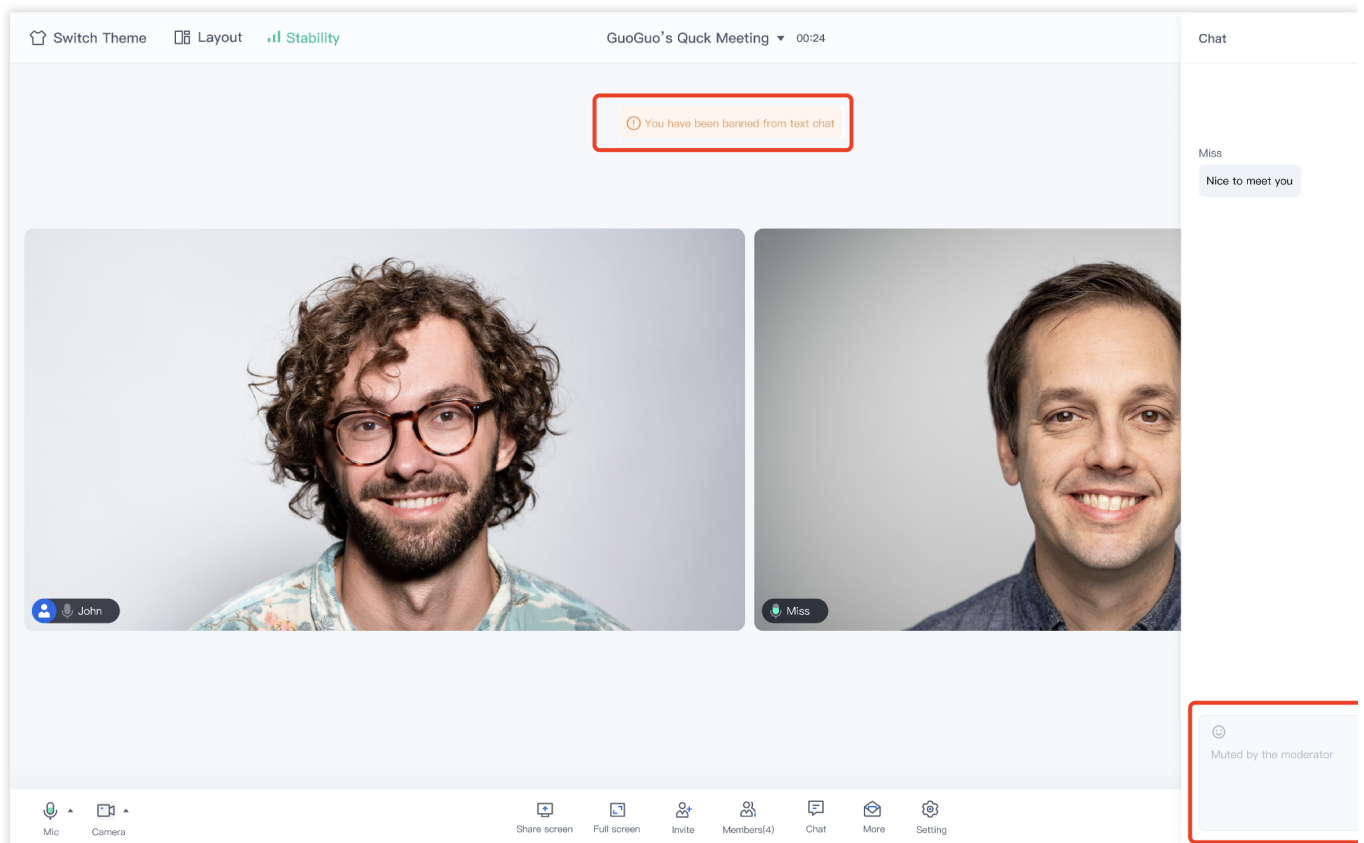
**Manage In-conference Chat Permissions**

The host/administrator can set the chat permissions of a member in member management.





Ordinary members who are prohibited from speaking will not be able to edit or send text or emoticons.



## Feature Access

Currently, the TUIRoomKit for Web and Electron has integrated chat capabilities during the meeting. You can click [Github](#) to download the TUIRoomKit code, and refer to the README.md documentation in the code repository to run the TUIRoomKit Web sample project.

### Note:

TUIRoomKit Web&Electron currently only supports text message chat. If you need to send image message chat and other extended content, you can implement it by getting the tim instance. For the web-side tim capabilities, please check: [IM API Documentation](#).

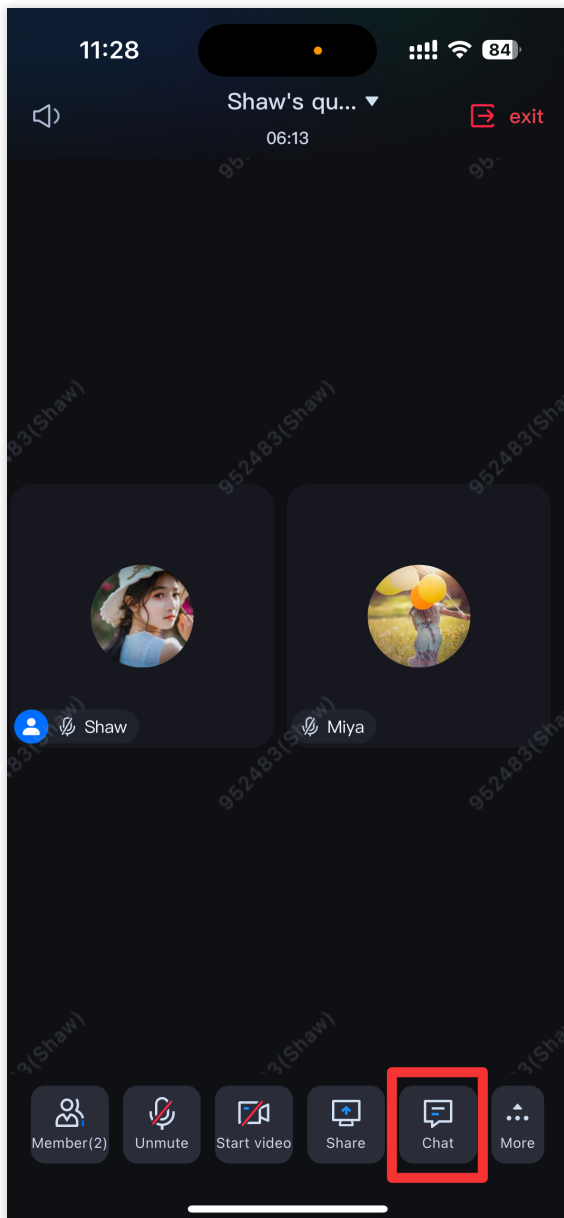
# Android&iOS

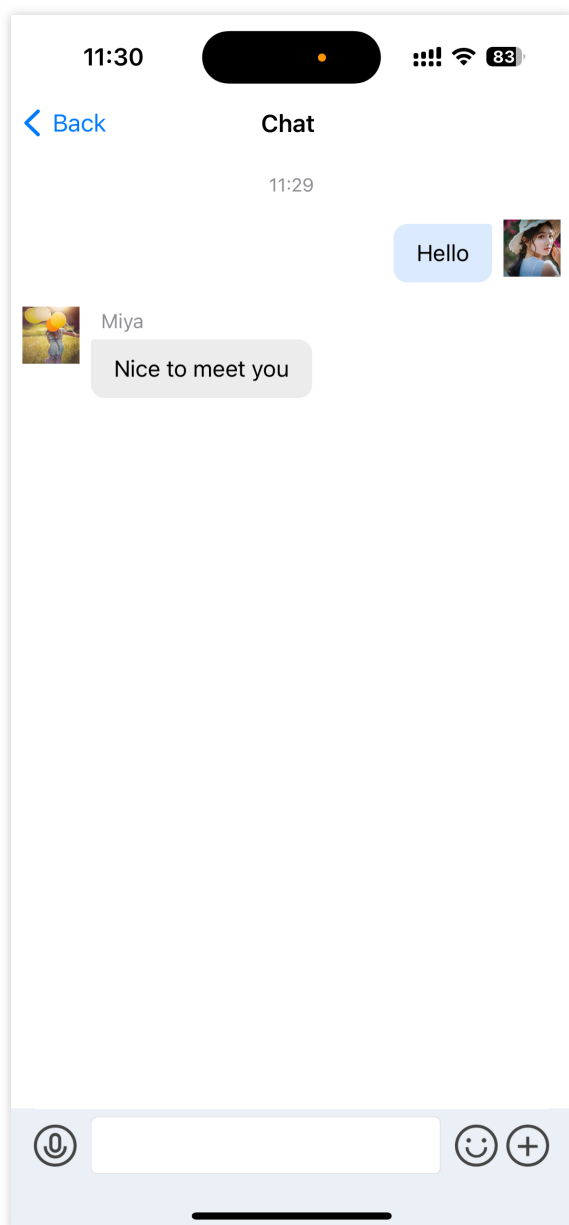
Last updated : 2024-07-03 17:14:46

In video conferences, participants can send messages in real-time in the chat area, share opinions and ideas, and create a relaxed and pleasant communication environment by exchanging emoticons and animations. To maintain the order of the meeting, the host or administrator can set to prohibit participants from sending messages in the chat, ensuring the focus and efficiency of the conference content. By flexibly using these features, video conferences can provide efficient and convenient communication experiences for various scenarios.

## Text Interaction

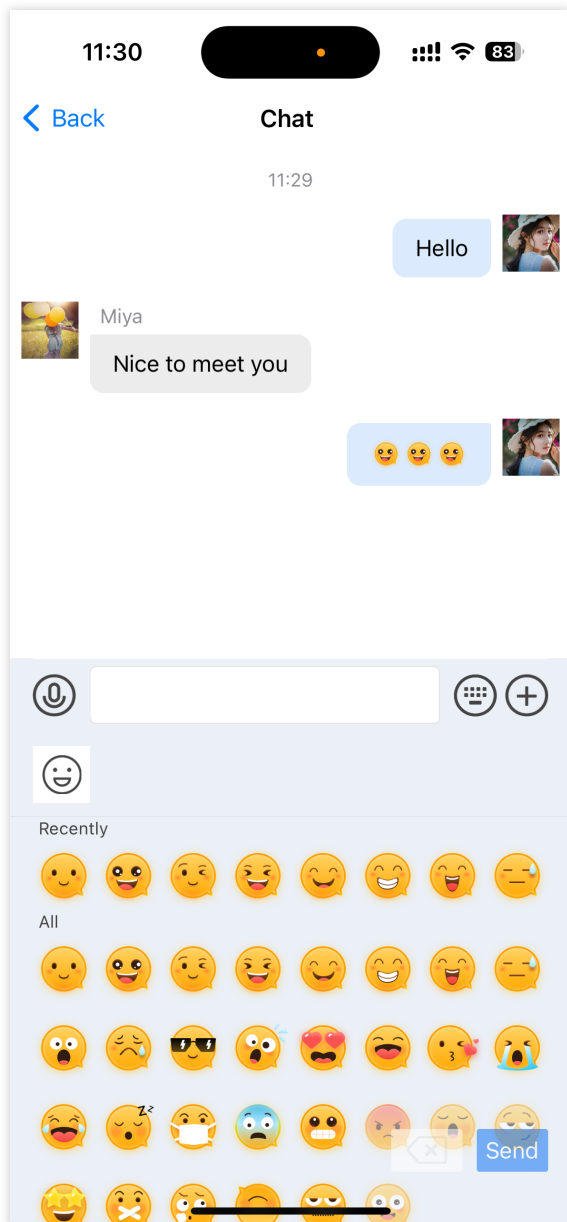
At the bottom menu bar of the conference interface, you can find an option called **Chat**. Click on it, and you will be redirected to the chat page. On the chat page, participants can communicate with other participants by sending custom text messages. This design facilitates real-time communication between participants without affecting the person who is speaking.





## Emoticon Interaction

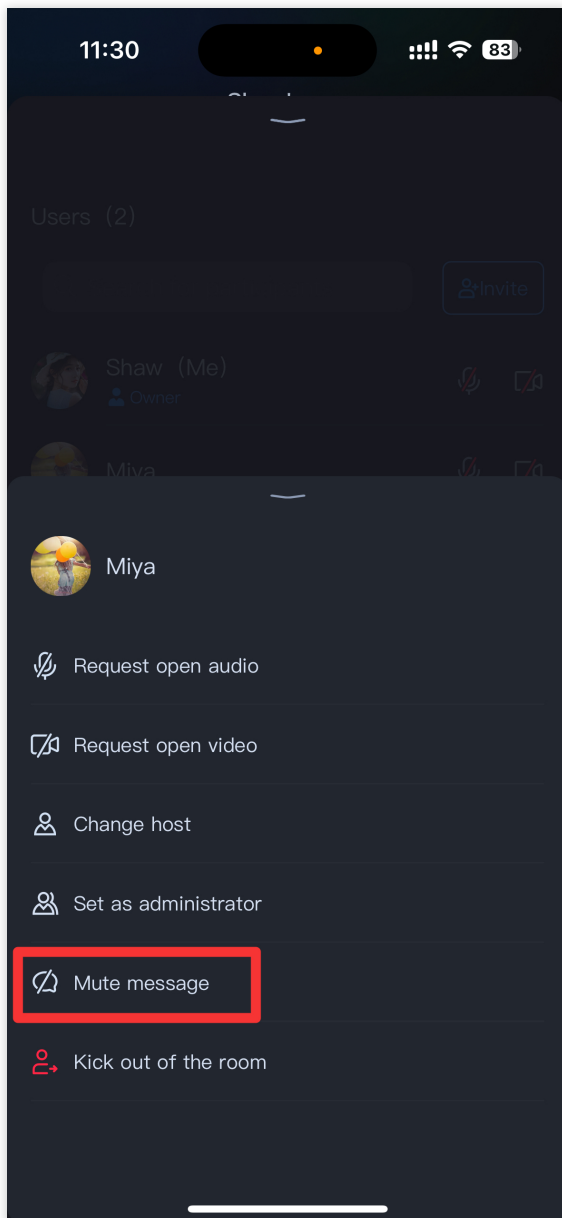
Click the **emoticon icon** in the chat interface's message editing bar to display the emoticon list. Click the corresponding emoticon to send it and give a thumbs-up to the speaker's brilliant speech.

**Note :**

To respect the emoji design copyright, the Conference project does not include large emoji element slices. Before the official commercial launch, please replace them with your own designed emoji or other emoji packs that you own the copyright to. The default **Little Yellow Face Emoji Pack copyright belongs to Tencent Cloud** and can be licensed for a fee. If you wish to obtain a license, you can [Submit a ticket](#) to contact us.

## Manage In-conference Chat Permissions

The host/administrator can set the chat permissions of a member in member management. Ordinary members who are prohibited from speaking will not be able to edit or send text or emoticons.



## Access Management

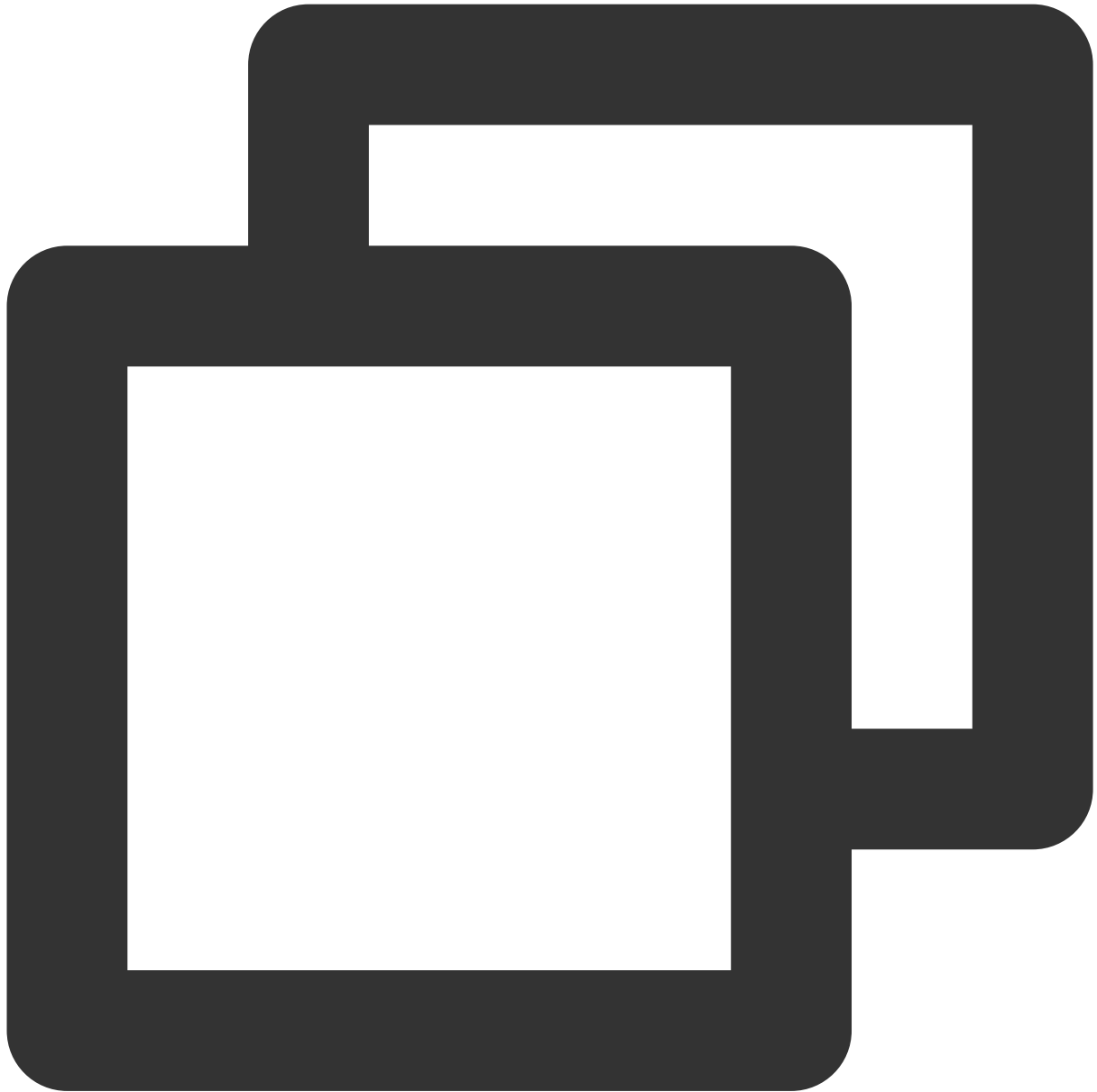
Android

iOS

## Module Source Code Integration

1. Clone/download the code from [Github](#), and then copy the `tuichat` subdirectory from the Android directory to the same level as your current project's app directory.

2. Find the `setting.gradle` file in the root directory of the project and add the following code to it. This code imports the `tuichat` as a local module into your current project.



```
include ':tuichat'
```

3. Find the `build.gradle` file in the app directory and add the following code to it. This code declares the current app's dependency on the newly added `tuichat` component.





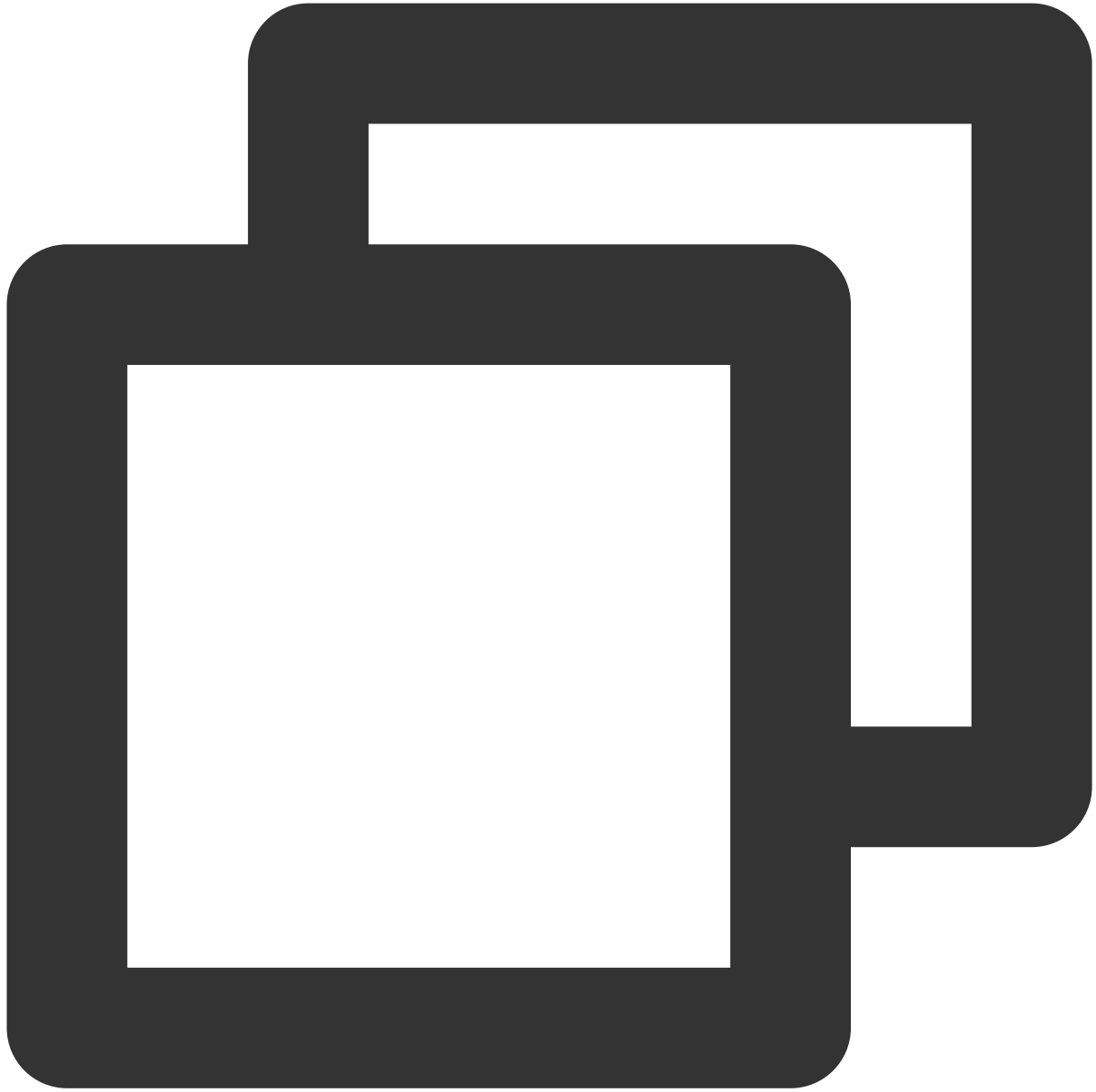
```
api project(':tuichat')
```

4. Add the maven repository and Kotlin support in the `build.gradle` file of the root project (at the same level as `settings.gradle`):



```
buildscript {  
    repositories {  
        mavenCentral()  
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:7.0.0'  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"  
    }  
}
```

If you are using Gradle 8.x, you need to add the following code.



```
buildscript {  
    repositories {  
        mavenCentral()  
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:8.0.2'  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"  
    }  
}
```

```
}
```

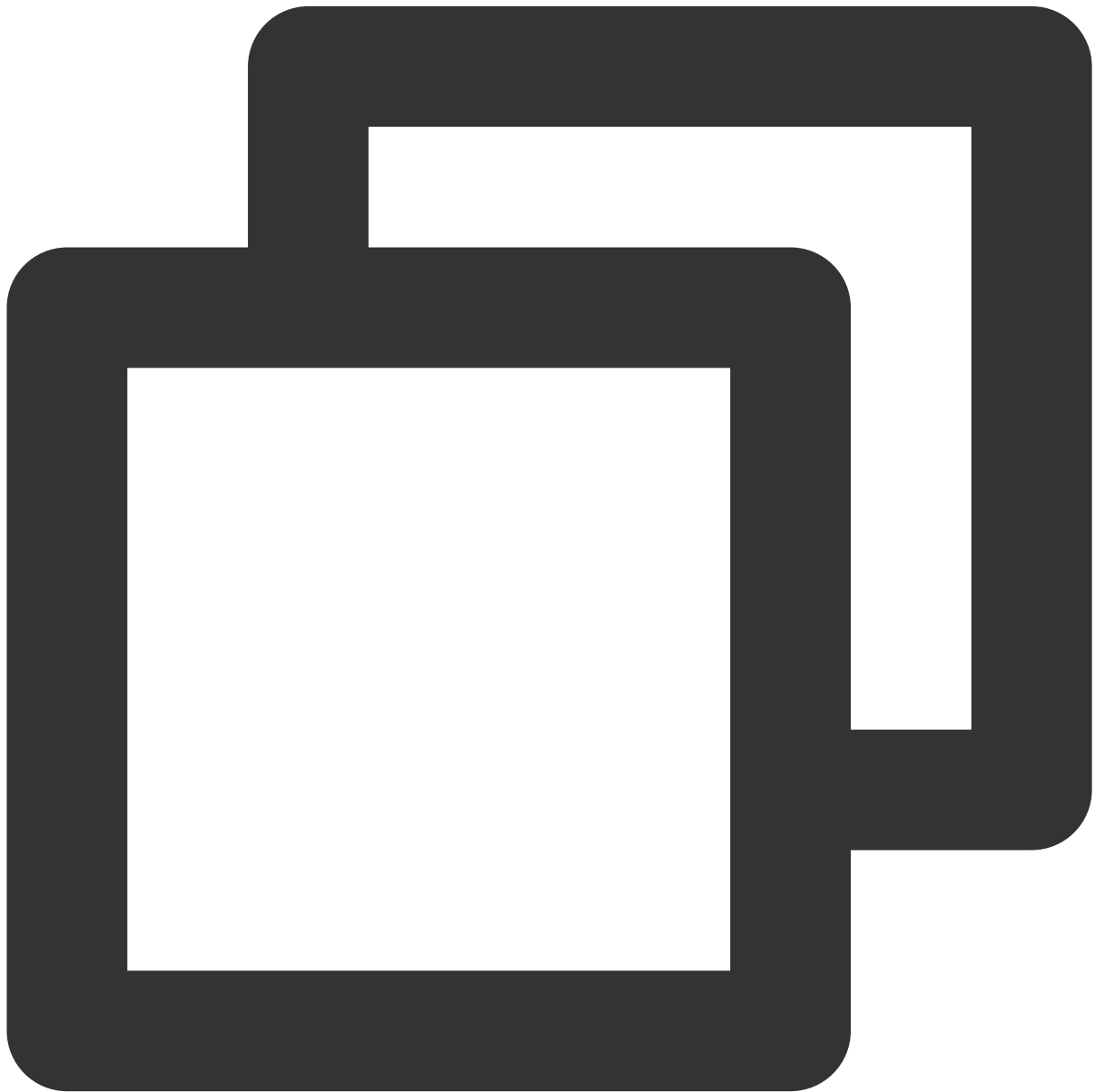
**Note:**

The corresponding relationship between Kotlin, Gradle, and AGP versions can be [viewed here](#).

## Access Chat Widget

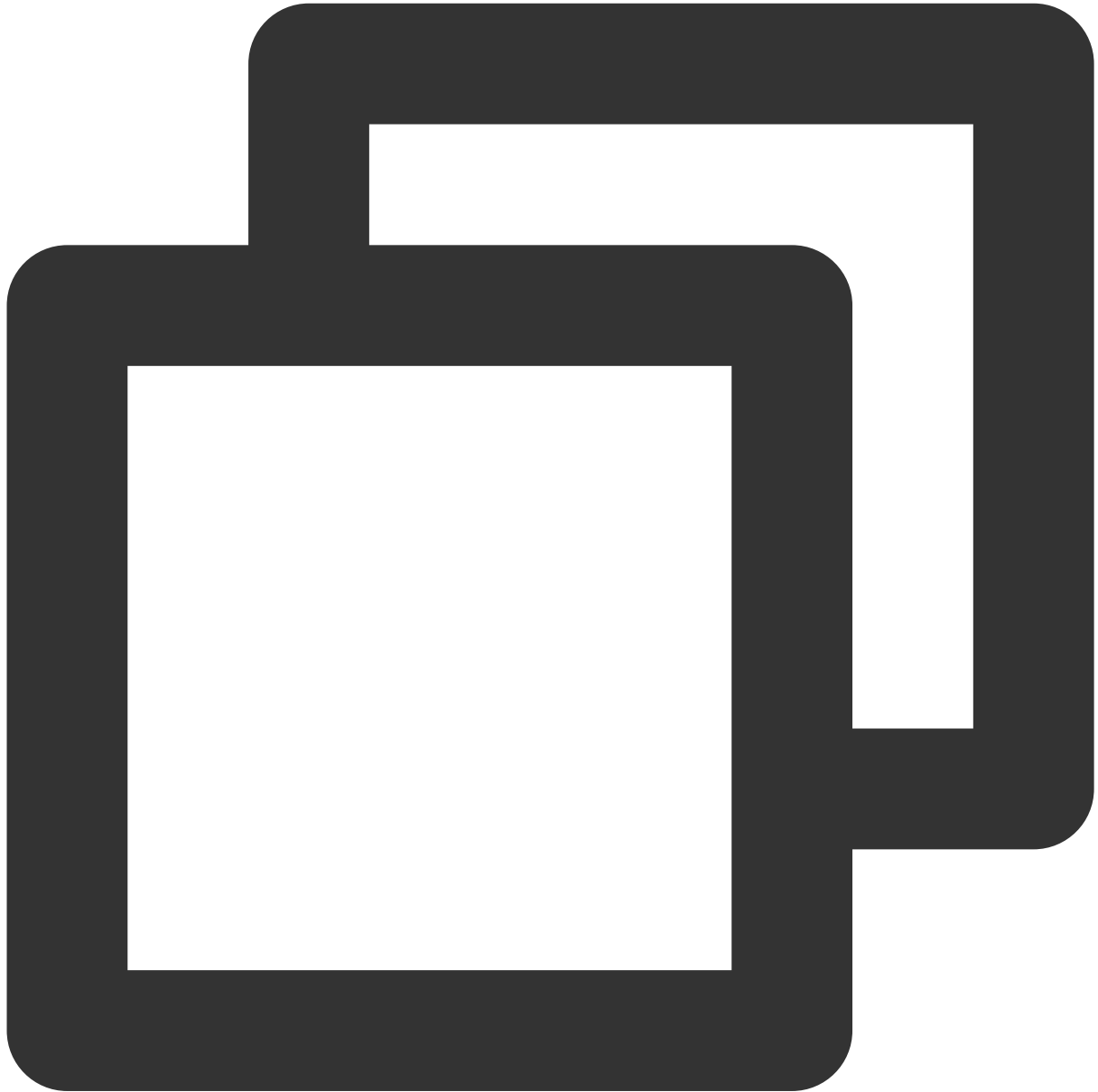
To import the chat widget using CocoaPods, follow these steps:

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



```
pod 'TUIChat'          # [Optional] Chat widget
```

2. Execute the following command to install the components.



```
pod install
```

# Flutter

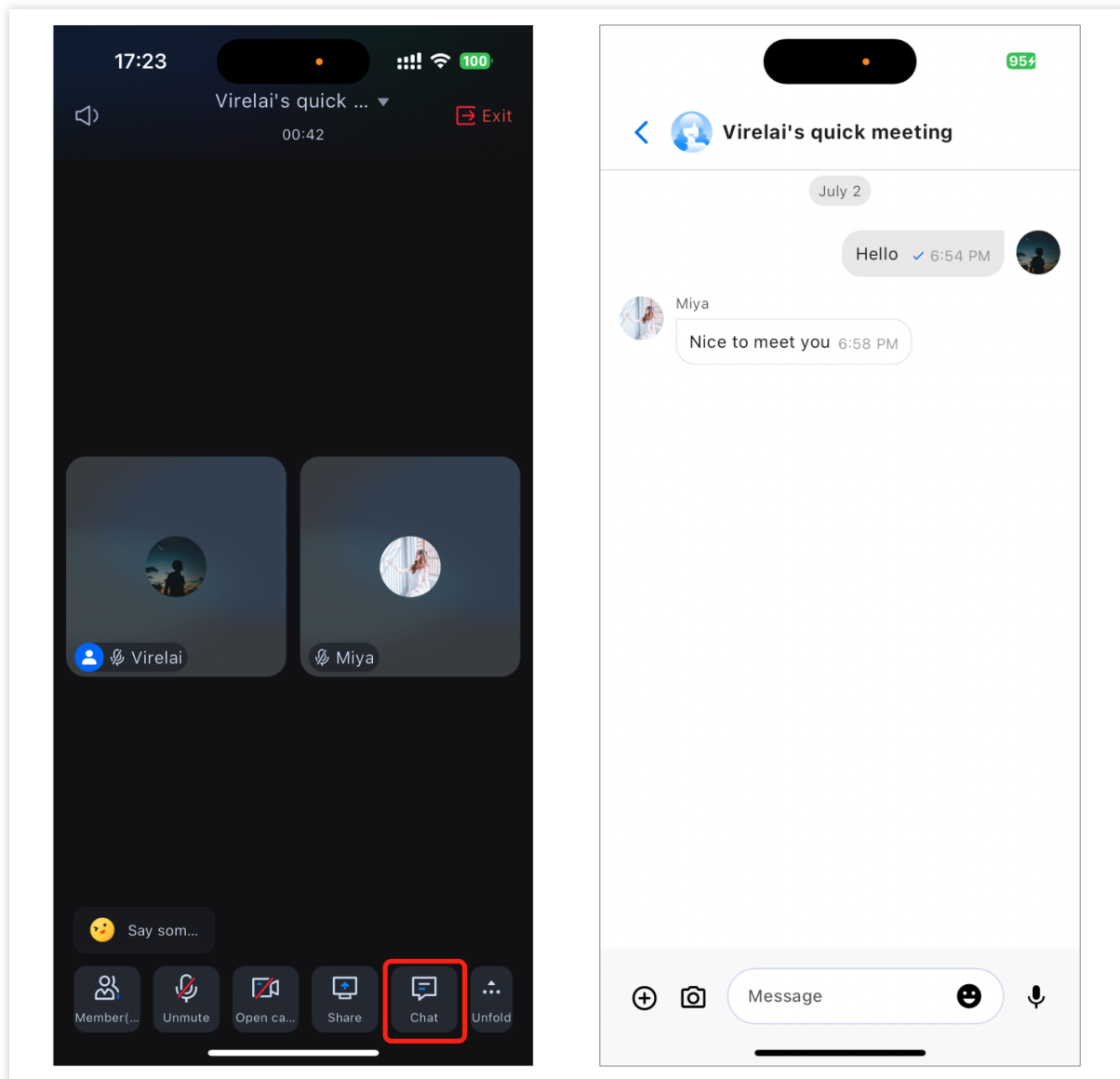
Last updated : 2024-07-05 19:07:47

In video conferences, participants can send messages in real-time in the chat area, share opinions and ideas, and create a relaxed and pleasant communication environment by exchanging emoticons and animations. To maintain the order of the meeting, the host or administrator can set to prohibit participants from sending messages in the chat, ensuring the focus and efficiency of the conference content. By flexibly using these features, video conferences can provide efficient and convenient communication experiences for various scenarios.

## Feature Introduction

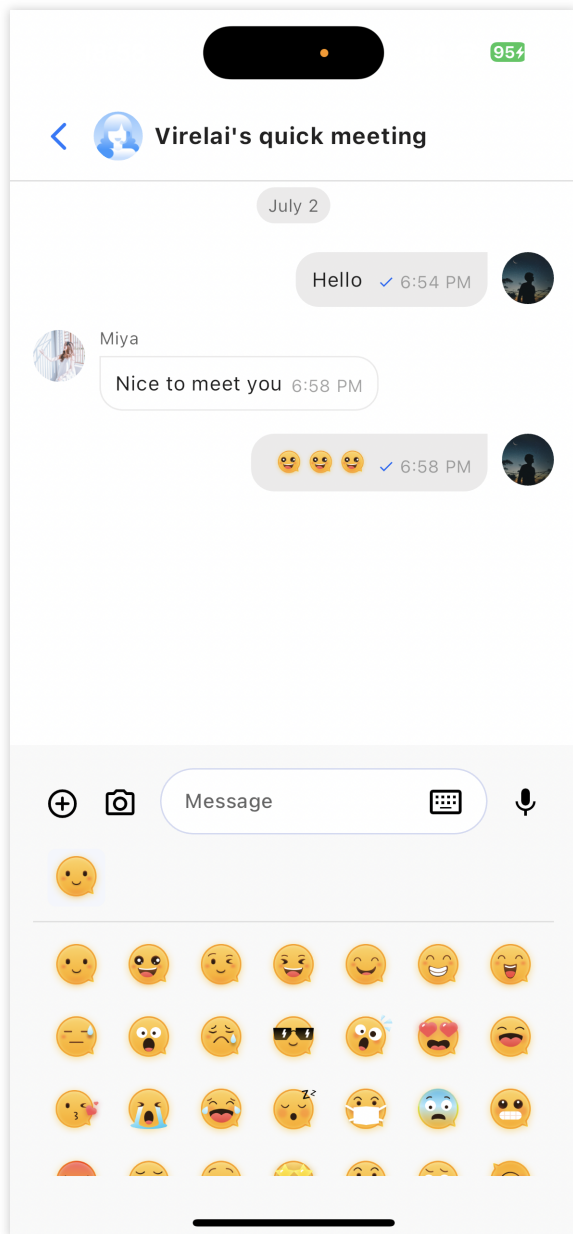
### Text, Multimedia Information Interaction

Click the **Chat** option at the bottom of the conference interface to access the chat interface. Participants can freely send **text**, **pictures**, **videos**, and **voice** messages, enabling real-time communication without disrupting the conference flow.



## Emoji Interaction

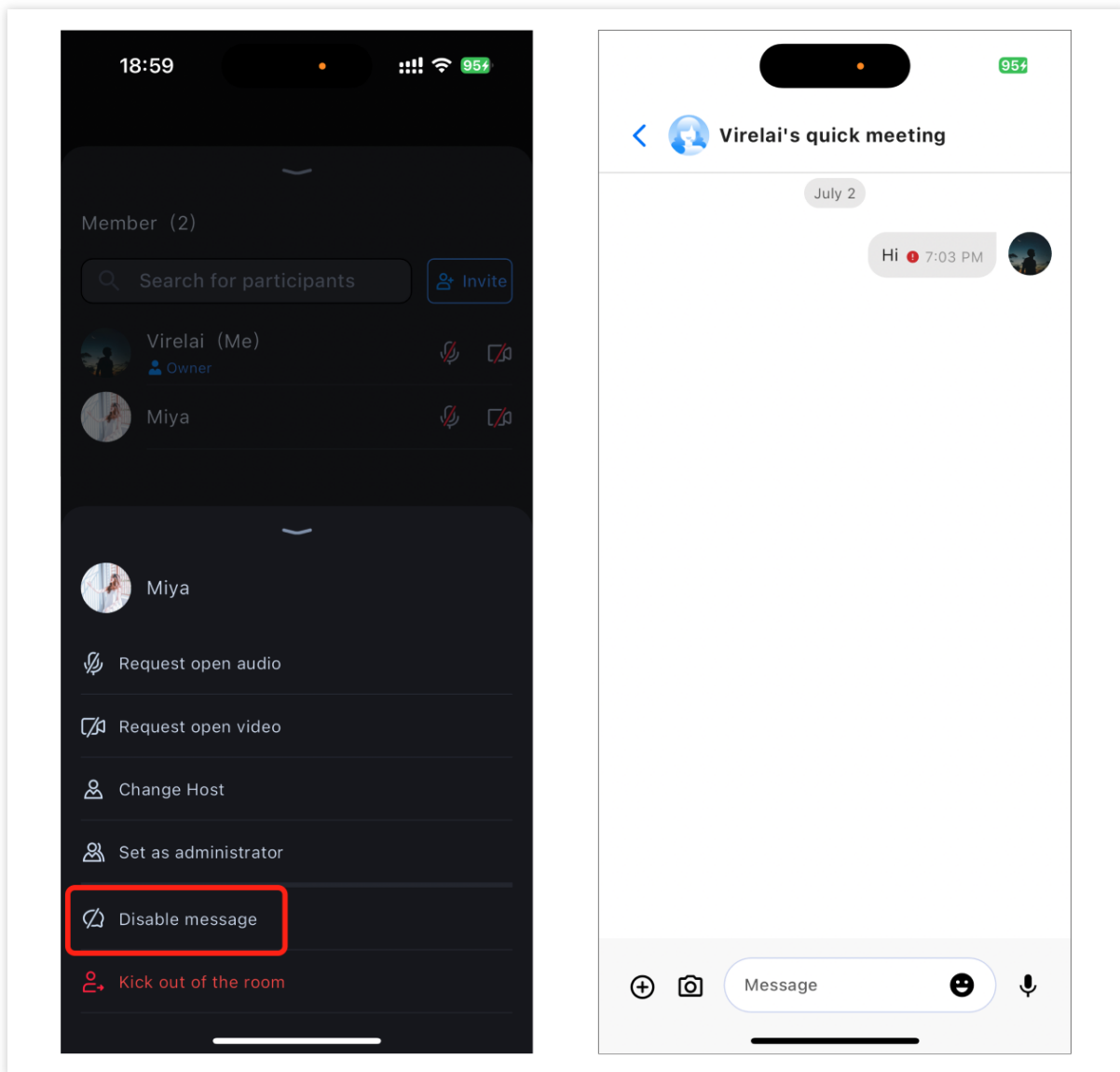
Click the **Emoji Icon** in the message board editor in the chat interface to bring up the emoji list. Click the corresponding emoji to display it in the message board for sending.



## Chat permissions in the control panel

The host/administrator can set a member's chat permissions in Member Management. If muted, regular members will not be able to send messages.





## feature Integration

### Integrate chat component

Add the `tencent_cloud_chat_message` plugin dependency to your project's `pubspec.yaml` file.



```
dependencies:  
  tencent_cloud_chat_message: latest version
```

## International Language Configuration

This step is required. First, import the localization tools into the entry file of the application.



```
import 'package:tencent_cloud_chat_intl/localizations/tencent_cloud_chat_localizati
```

Next, add the localization configuration to entries provided by third-party packages such as `MaterialApp` or `GetMaterialApp`. Here, `GetMaterialApp` is used as an example:



```
GetMaterialApp(  
  localizationsDelegates: const [  
    /// Your original configuration  
    GlobalMaterialLocalizations.delegate,  
  
    /// Add this line  
    ...TencentCloudChatLocalizations.localizationsDelegates,  
  ],  
  supportedLocales: [  
    /// Your original configuration  
    ...S.delegate.supportedLocales,
```

```
    /// Add this line
    ...TencentCloudChatLocalizations.supportedLocales,
  ],
  /// Other settings
)
```

## Initialization and Login

Add the following code to your project. Its function is to complete the initialization and log in to by calling the relevant interfaces in the chat component. This step is crucial, as the chat can only be used normally after initialization. Therefore, please be patient and check whether the relevant parameters are configured correctly. Among the parameters, **sdkAppId**, **userId**, and **userSig**, you have already used them when [logging in to TUIRoomKit](#).



```
import 'package:tencent_cloud_chat/components/component_config/tencent_cloud_chat_m
import 'package:tencent_cloud_chat/components/component_config/tencent_cloud_chat_m
import 'package:tencent_cloud_chat/models/tencent_cloud_chat_models.dart';
import 'package:tencent_cloud_chat/tencent_cloud_chat.dart';
import 'package:tencent_cloud_chat_message/tencent_cloud_chat_message.dart';

await TencentCloudChat.controller.initUIKit(
  options: TencentCloudChatInitOptions(
    sdkAppID: 'SDKAPPID', // Your SDKAPPID
    userID: 'userID',      // Your userID
    userSig: 'userSig',    // Your userSig
```

```
),
components: TencentCloudChatInitComponentsRelated(
  usedComponentsRegister: [TencentCloudChatMessageManager.register], // Register
  componentConfigs: TencentCloudChatComponentConfigs(
    messageConfig: TencentCloudChatMessageConfig(
      // The following configuration is recommended.
      showMessageSenderName: ({groupID, topicID, userID}) => true,
      showSelfAvatar: ({groupID, topicID, userID}) => true,
      defaultMessageMenuConfig: ({groupID, topicID, userID}) =>
        TencentCloudChatMessageDefaultMessageMenuConfig(
          enableMessageForward: false,
          enableMessageSelect: false,
        ),
    ),
  ),
),
plugins: [],
);
```

## Using Emojis (Optional)

If you need to send and receive emojis, you need to configure as follows:

### Add Dependencies

In your project's `pubspec.yaml` file, add the `tencent_cloud_chat_sticker` plugin dependency.

### Complete configuration

In the `plugins` of [Initialization and Login](#) in the previous step, add the following code:



```
plugins: [  
  TencentCloudChatPluginItem(  
    name: "sticker",  
    initData: TencentCloudChatStickerInitData(  
      useDefaultSticker: true,           // Default stickers, only this sticker pa  
      useDefaultCustomFace_4350: false, // If you do not need to use TUIRoomKit f  
      useDefaultCustomFace_4351: false,  
      useDefaultCustomFace_4352: false,  
      userID: 'userId',                 // Your userId  
    ).toJson(),  
    pluginInstance: TencentCloudChatStickerPlugin(  

```



```
        context: context,  
    ),  
),  
],
```

**Note:**

To respect the copyright of emoji designs, the TUIRoomKit example project does not include large emoji cut elements. Before official commercial use, please replace them with your own designs or other emoji packs you have copyright to. The default **Little Yellow Face emoji pack is copyrighted by Tencent Cloud**, and can be licensed for a fee. If you wish to obtain a license, you can [Submit a ticket](#) to contact us.

**Using Chat**

When you [create or join a conference successfully](#), you need to pass the `chatWidget` into the conference page, `ConferenceMainPage`. After passing it in, the chat button will **display** in the bottom toolbar. Clicking the chat button will automatically navigate to `chatWidget`.



```
Navigator.push(  
  context,  
  MaterialPageRoute(  
    builder: (context) => ConferenceMainPage( // Conference main page  
      chatWidget: TencentCloudChatMessage(  
        options: TencentCloudChatMessageOptions(groupId: 'yourConferenceId'), //  
      ),  
    ),  
  ),  
);
```

After completing the above configuration, you can click the chat button to chat during the conference.

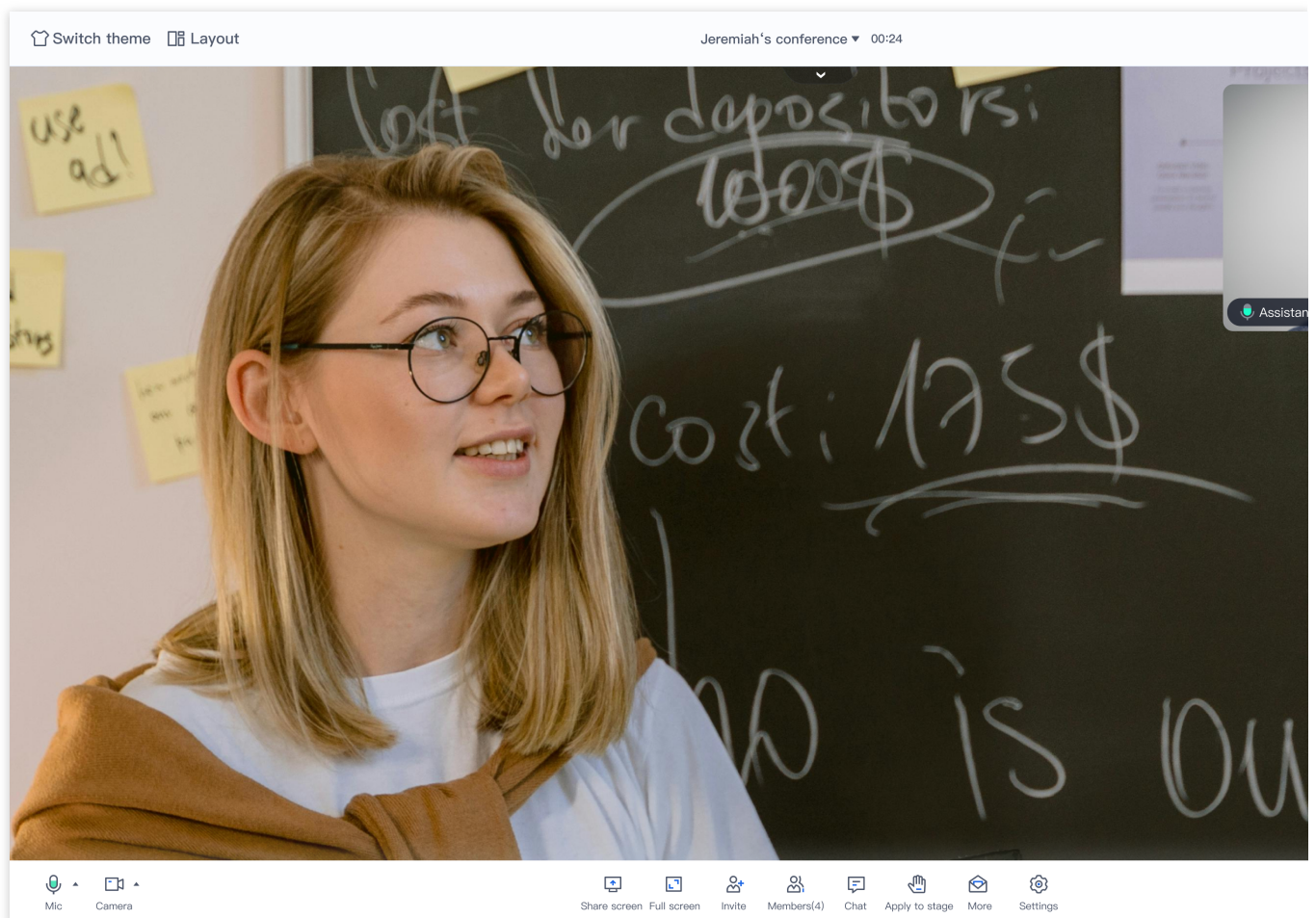
# Robot Streaming (TUIRoomKit)

Last updated : 2024-07-02 17:06:35

## Description of the Feature

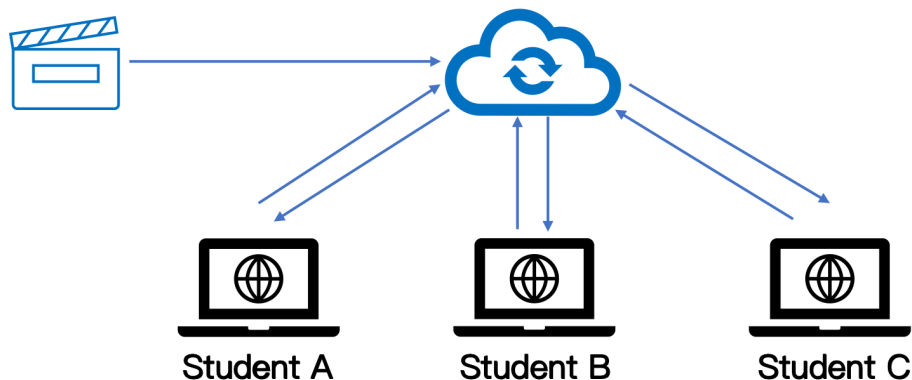
TUIRoomKit supports the use of robot push streaming, allowing users to push local or online media files into the room. In some scenarios, meetings may require the push of pre-recorded instructional videos or online video streams to online classrooms for users to watch and learn. TUIRoomKit can meet these needs, achieving automated and efficient video playback.

To use this feature, you need to subscribe to the [TUIRoomKit Monthly Package](#). An robot stream will be considered a user in a room, which will incur call duration fees. For more information, please refer to the [Billing of Audio and Video Duration](#). Transcoding operations will be performed during the process of connecting robot streams, thereby incurring transcoding fees. For more information, please refer to the [Billing of On-Cloud MixTranscoding](#).



## Application Scenario

Watching together, listening together, playing together, and learning together – experiences that previously required offline, face-to-face interaction are increasingly being moved online. The ability to watch movies, listen to music, and then discuss and share opinions with friends thousands of miles away is a magical real-time interactive experience that is loved by young people today, becoming a focus and mainstream direction of current audio and video products. TUIRoomKit input media stream feature allows sharing of external media streams within a room. Users can use this feature to share music, movies, lectures, courses, and other media content with other users in the room, engaging in real-time interaction with them. Platforms can leverage this new feature of TUIRoomKit to quickly implement a **watch together** scenario. In addition to watching movies and listening to music together, the input media stream capability of TUIRoomKit can also bring more innovative possibilities to the platform in scenarios such as interactive classrooms, sports competitions, and web conferences.



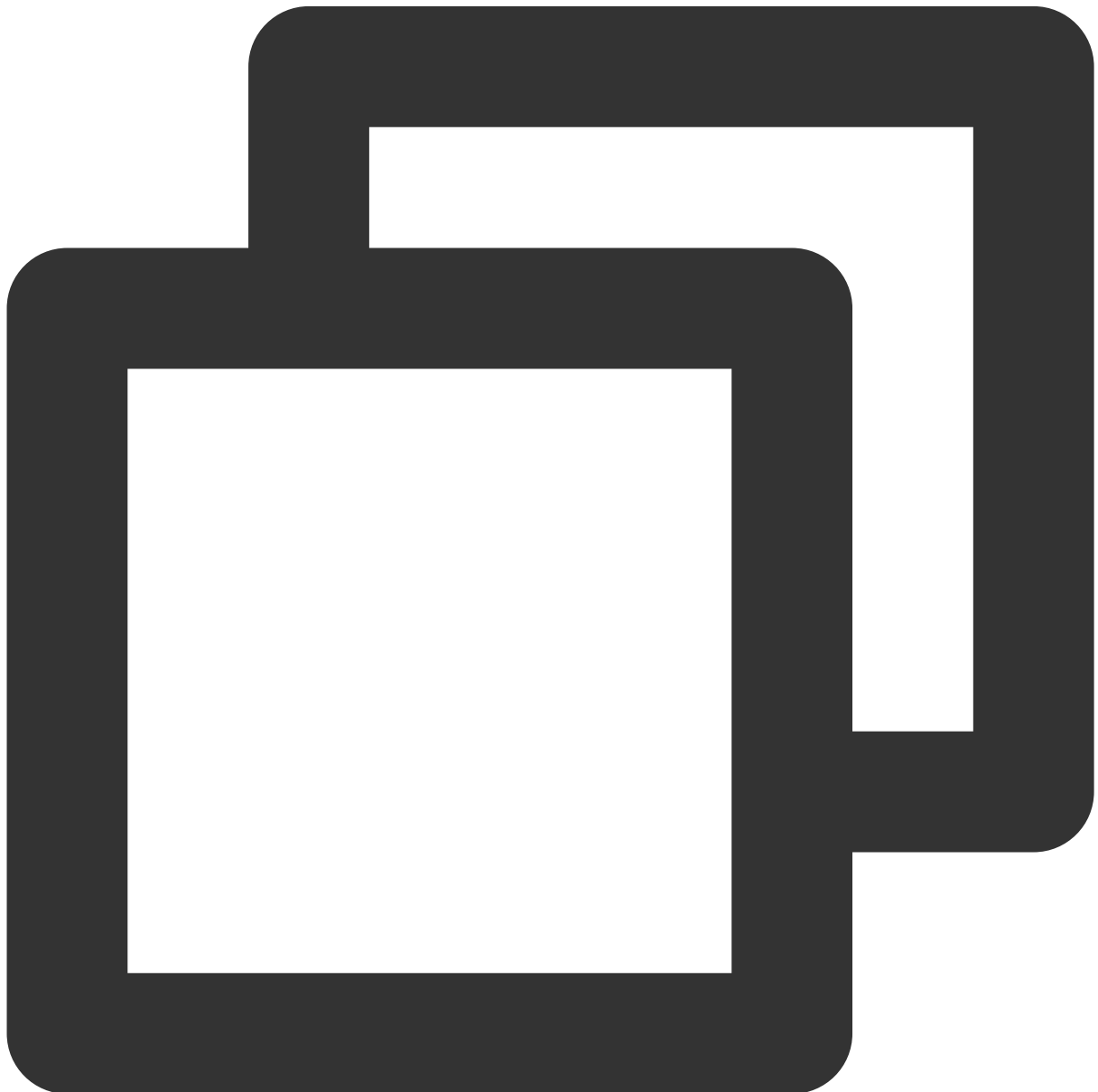
## Usage: Using RTMP for streaming

TUIRoomKit supports RTMP standard protocol for streaming. Depending on the situation, you can choose to install [OBS](#), FFmpeg, or other RTMP libraries for streaming.

## Using FFmpeg to publish streams

FFmpeg requires specific command configuration parameters for different scenarios, so you should have some experience with FFmpeg. Below are commonly used FFmpeg Command Line options; for more FFmpeg Options, please refer to the [FFmpeg official website](#).

### FFmpeg Command Line



```
ffmpeg [global_options] {[input_file_options] -i input_url} ... {[output_file_options]
```

## Common FFmpeg Options

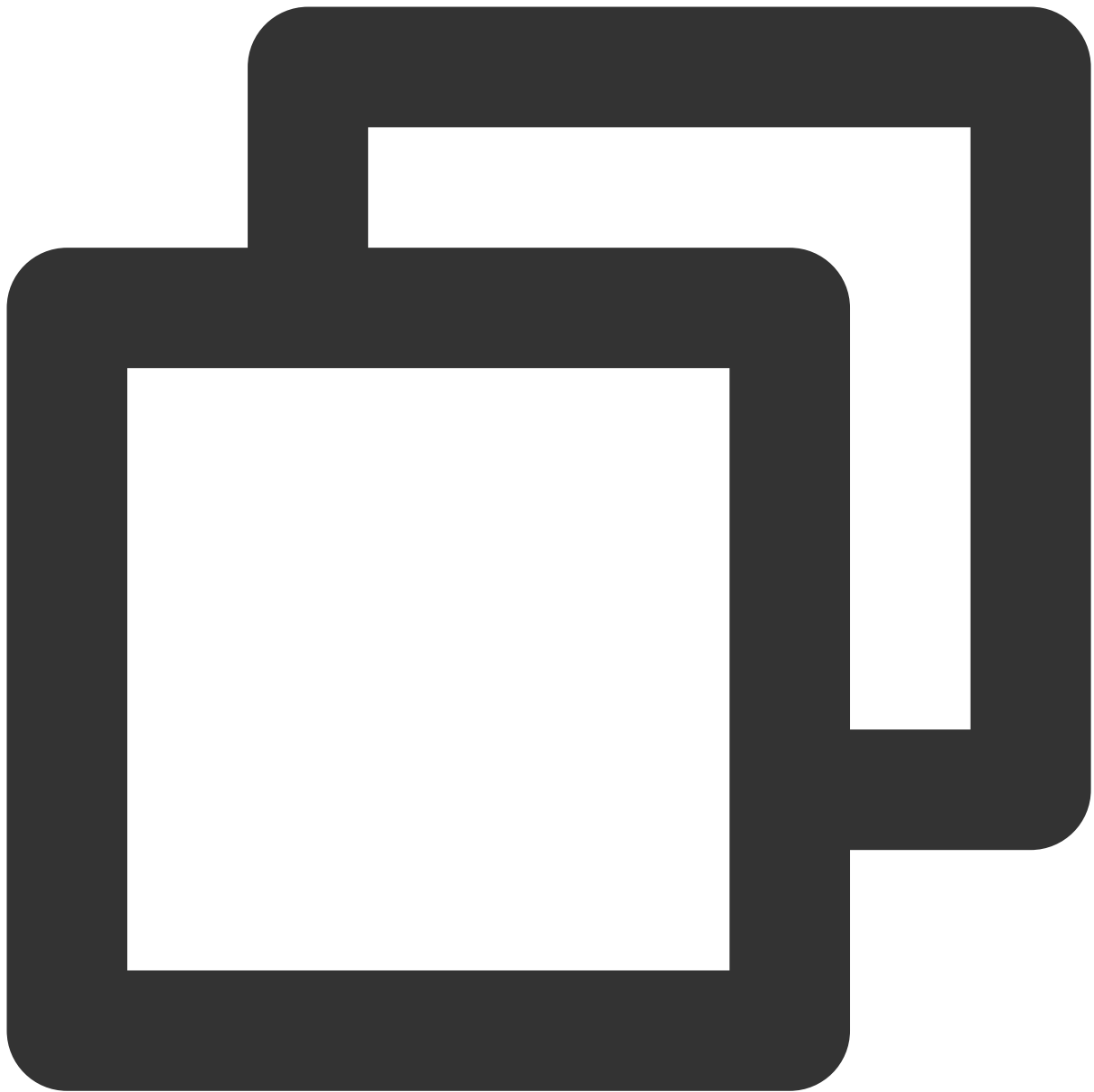
Option	Description
-re	Reads input at native frame rate, generally only used for reading local files

Among them, the configurable options for **output\_file\_options** include:

Option	Description
-c:v	Video Encoding, recommended to use libx264
-b:v	Video bitrate, for example, 1500k means 1500kbps
-r	Video Frame Rate
-profile:v	Video profile, specifying baseline will not encode B frames, TUIRoomKit backend does not support B frames
-g	GOP frame interval
-c:a	Audio Encoding, recommended to use libfdk_aac
-ac	Number of channels, fill in 2 or 1
-b:a	Audio Bitrate
-f	Specify format, fixed fill in <code>flv</code> , send to TUIRoomKit using FLV container

## Usage

The following example uses FFmpeg command to read files and push to TUIRoomKit, note the quotes around the URL.



```
ffmpeg -loglevel debug -re -i sample.flv -c:v libx264 -preset ultrafast -profile:v
```

The explanations for the parameters in the above command are as follows:

Parameter	Meaning
-i sample.flv	The media file that needs to be streamed to TUIRoomKit. You can replace sample.flv with the local or online media file you need to stream.
yourRoomId	The Room ID you need to stream to. You need to replace yourRoomId with your actual RoomId.



userId	The UserID you need to stream with. You need to change the value after "=" to the actual userId.
sdkappid	Your sdkappid, which you have previously obtained in <a href="#">Activate Service</a> . You need to change the value after "=" to the actual sdkAppID.
usersig	Your usersig, which you have obtained when logging into TUIRoomKit. You need to change the value after "=" to the actual userSig.

When you need to implement the above feature in code, you can use the relevant FFmpeg library on your platform or invoke FFmpeg through the command line to achieve the above command.

## Using OBS to publish streams

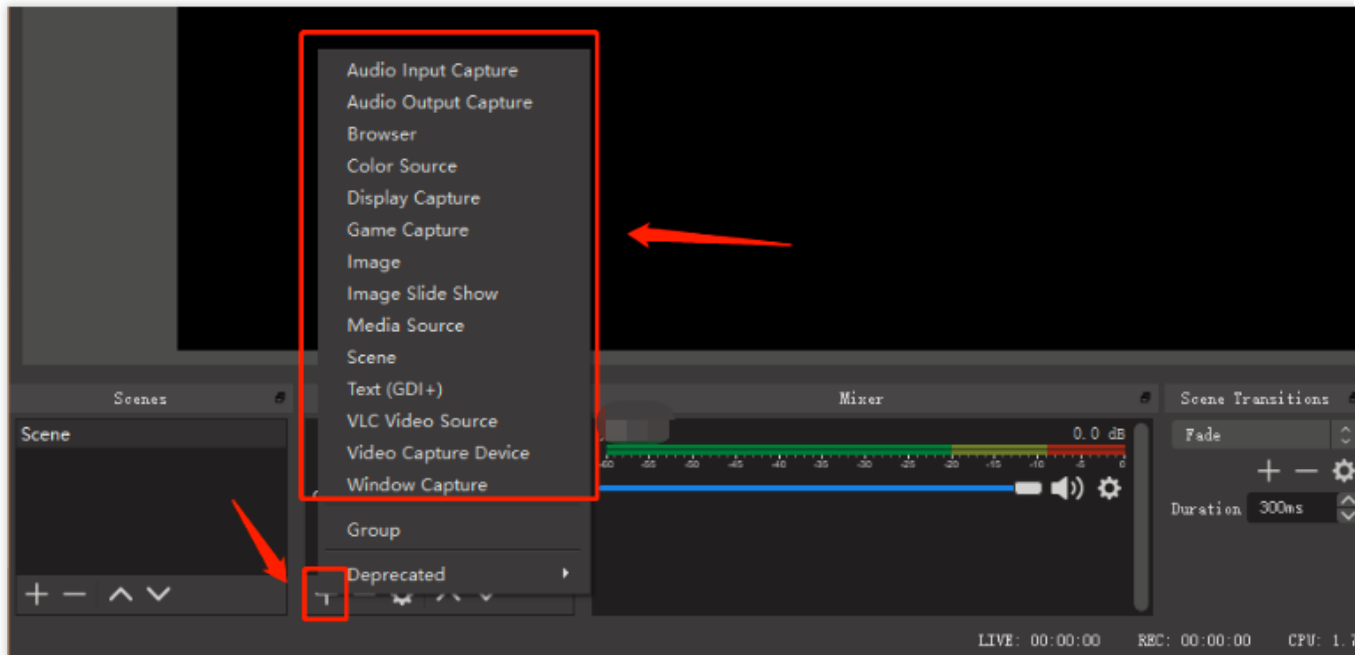
### Prerequisites

You have installed [OBS](#).

### Step 1. Select a source

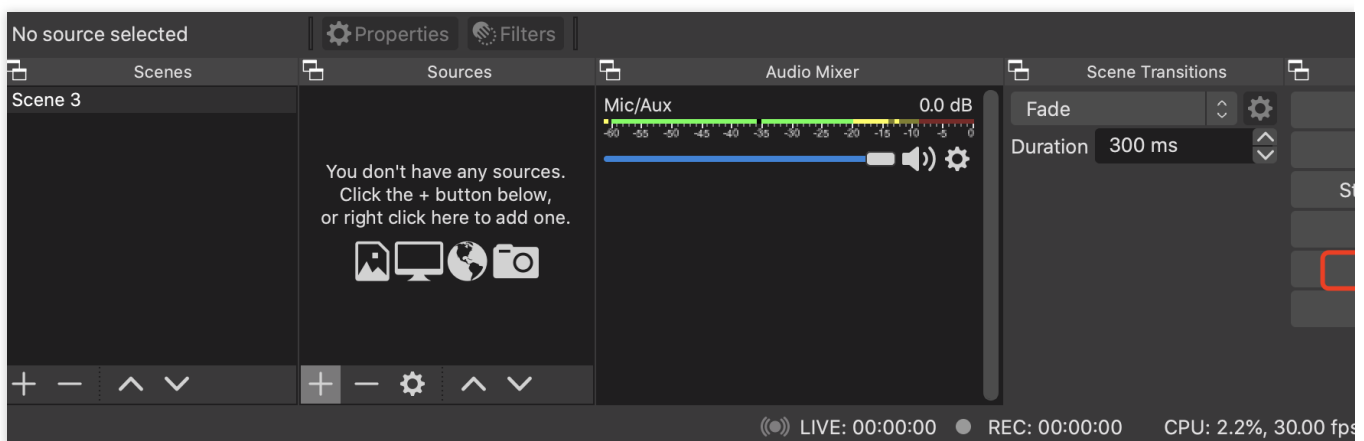
In the **Sources** panel at the bottom, click **++**, and select a source based on your needs. Common sources include the following:

Source	Note
Image	Publishes a single image
Image Slide Show	Publishes multiple images (you can determine the order of playback and whether to loop the playback)
Scene	Inserts an entire scene to enable various streaming effects
Media Source	Uploads a local file and publishes it as a live stream
Text	Adds real-time text to your stream
Window Capture	Captures and publishes the window you select in real time
Video Capture Device	Captures and publishes the images captured by a camera in real time
Audio Input Capture	Audio live streaming (audio input device)
Audio Output Capture	Audio live streaming (audio output device)

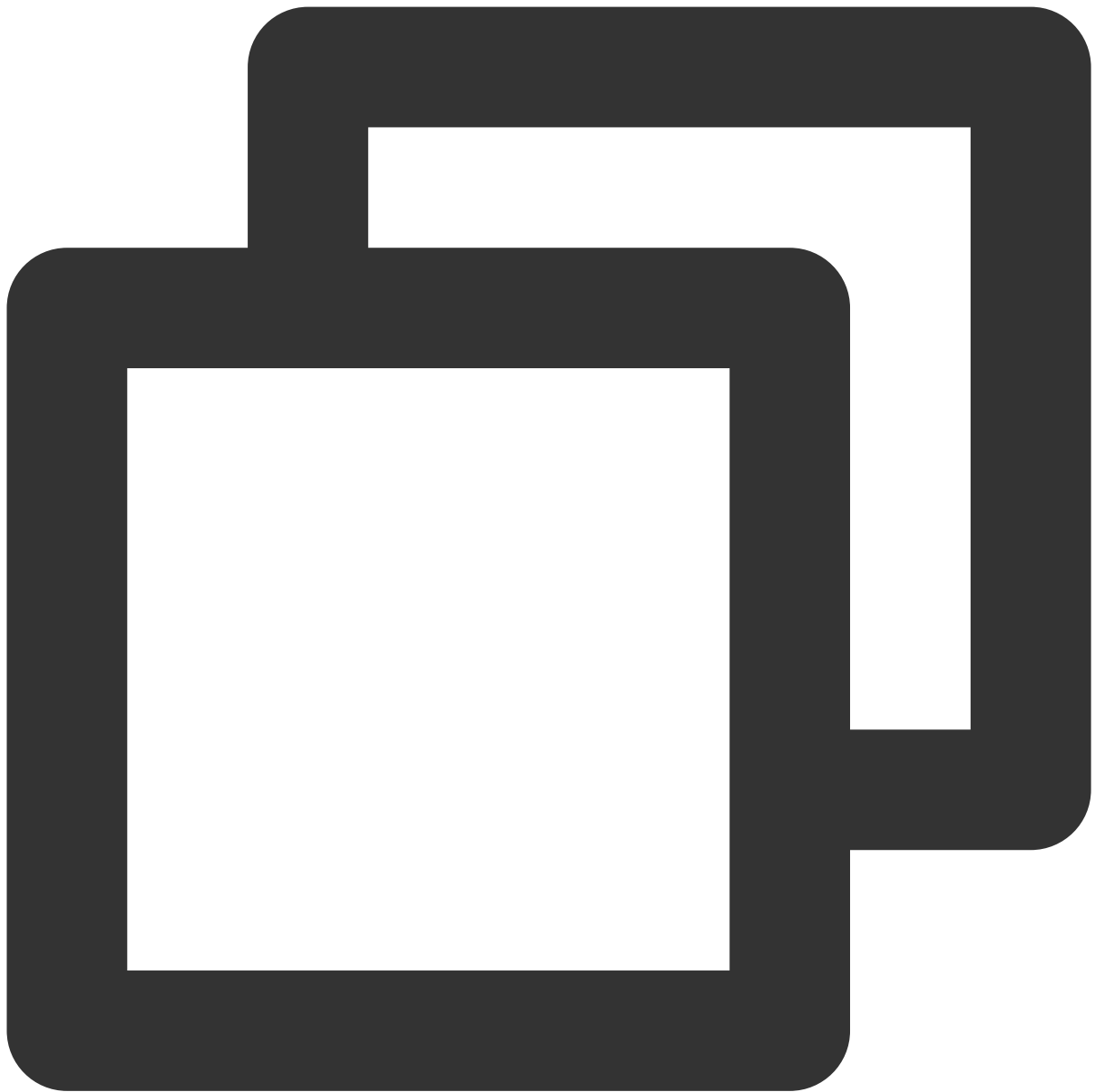


## Step 2. Set publishing parameters

1. In the **Controls** panel at the bottom, click **Settings**.

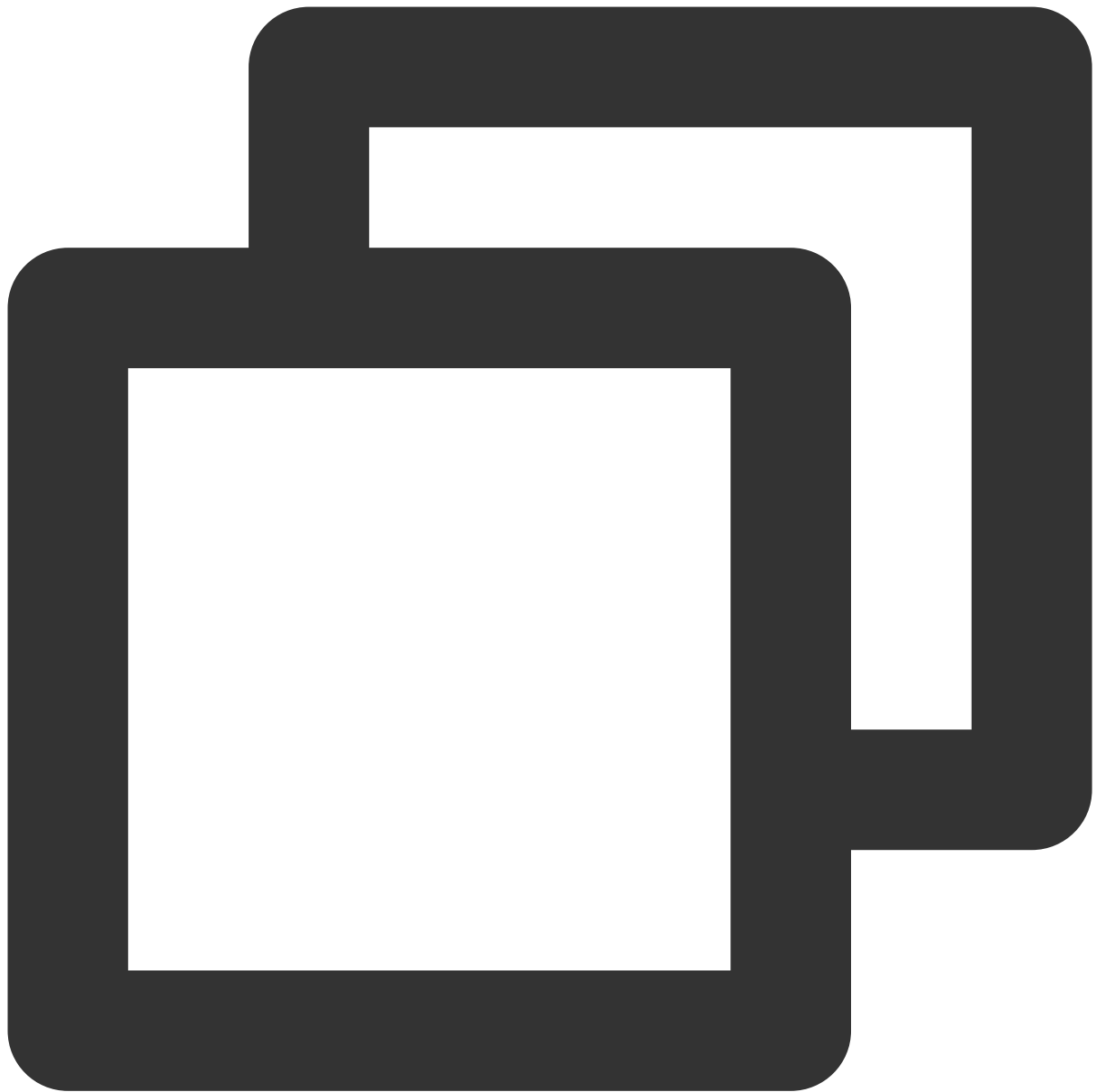


2. Click **Stream** and select **Custom** for **Service**.
3. Enter `rtmp://intl-rtmp.rtc.qq.com/push/` for **Server**.
4. Enter a stream key in the following format:

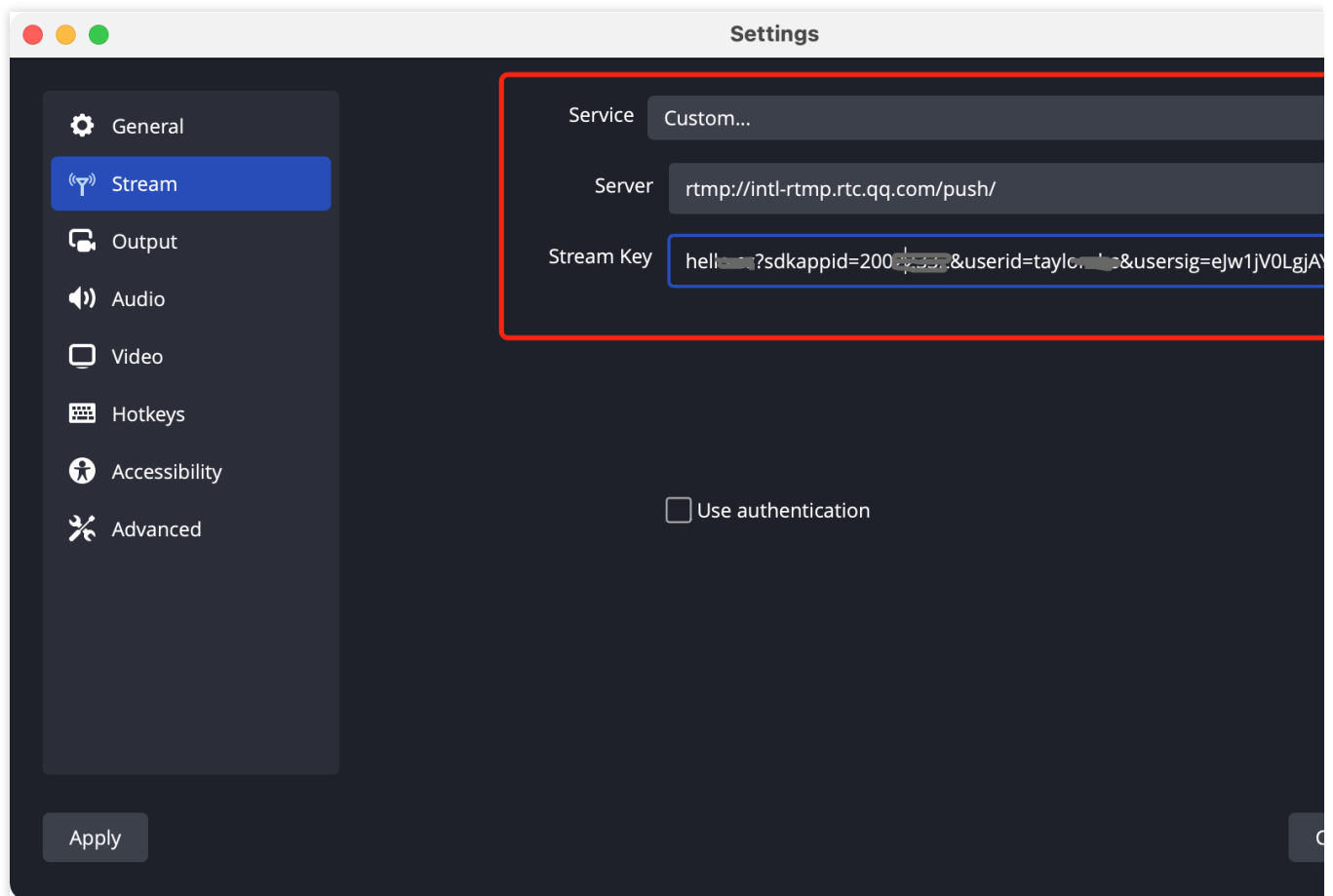


```
Room ID?sdkappid=Application&userid=User ID&usersig=Signature
```

Replace "Room ID", "Application ID", "User ID", and "Signature" with the actual values, for example:



`yourRoomId?sdkappid=140*****66&userid=*****rtmp2&usersig=eJw1jdE*****ZLg`



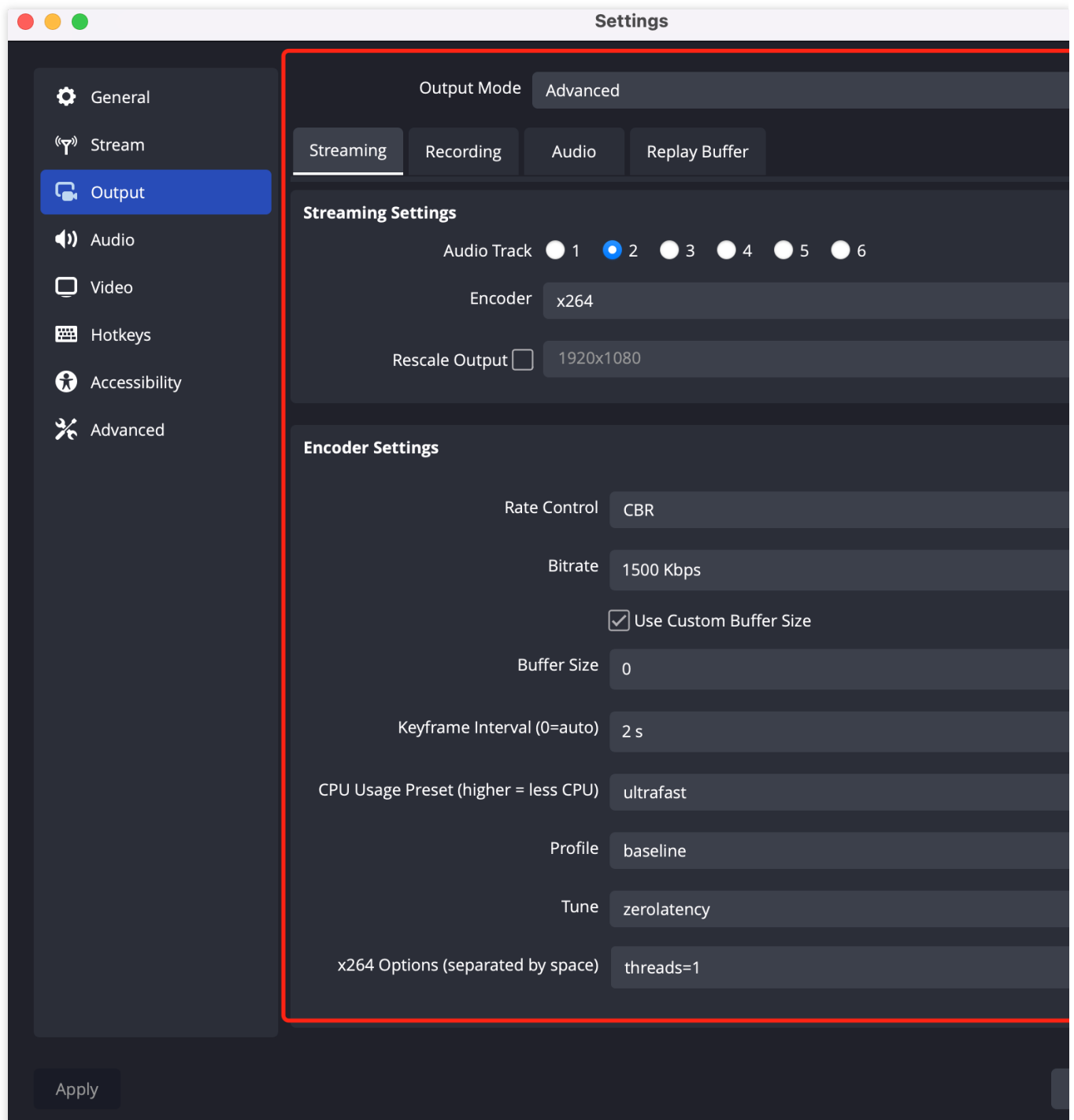
### Step 3. Configure the output

Because RTMP does not support B-frames, set the video encoding parameters as follows to remove B-frames.

1. Go to **Controls > Settings > Output**.
2. Select **Advanced** for **Output Mode**. The recommended **Keyframe Interval** is **1** or **2**. For **CPU Usage Preset**, select **ultrafast**. For **Profile**, select **baseline**. For **Tune**, select **zerolatency**. And for **x264 Options**, enter `threads=1`, and then click **OK**.

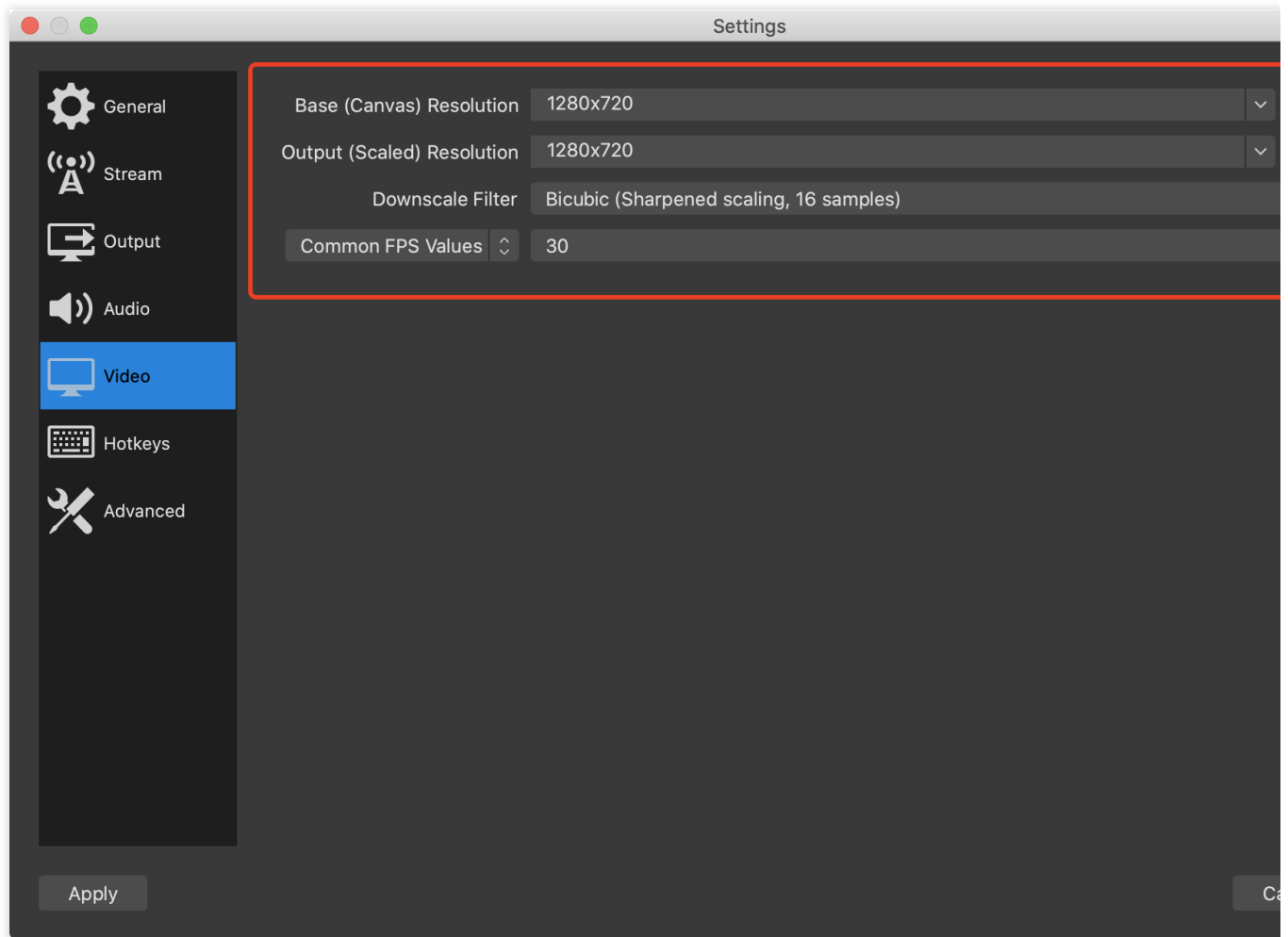
#### Warning:

You need to remove the B-frames in RTMP streams, otherwise the connection will be disconnected after pushing. To achieve this, select baseline for Profile.



#### Step 4. Set video parameters

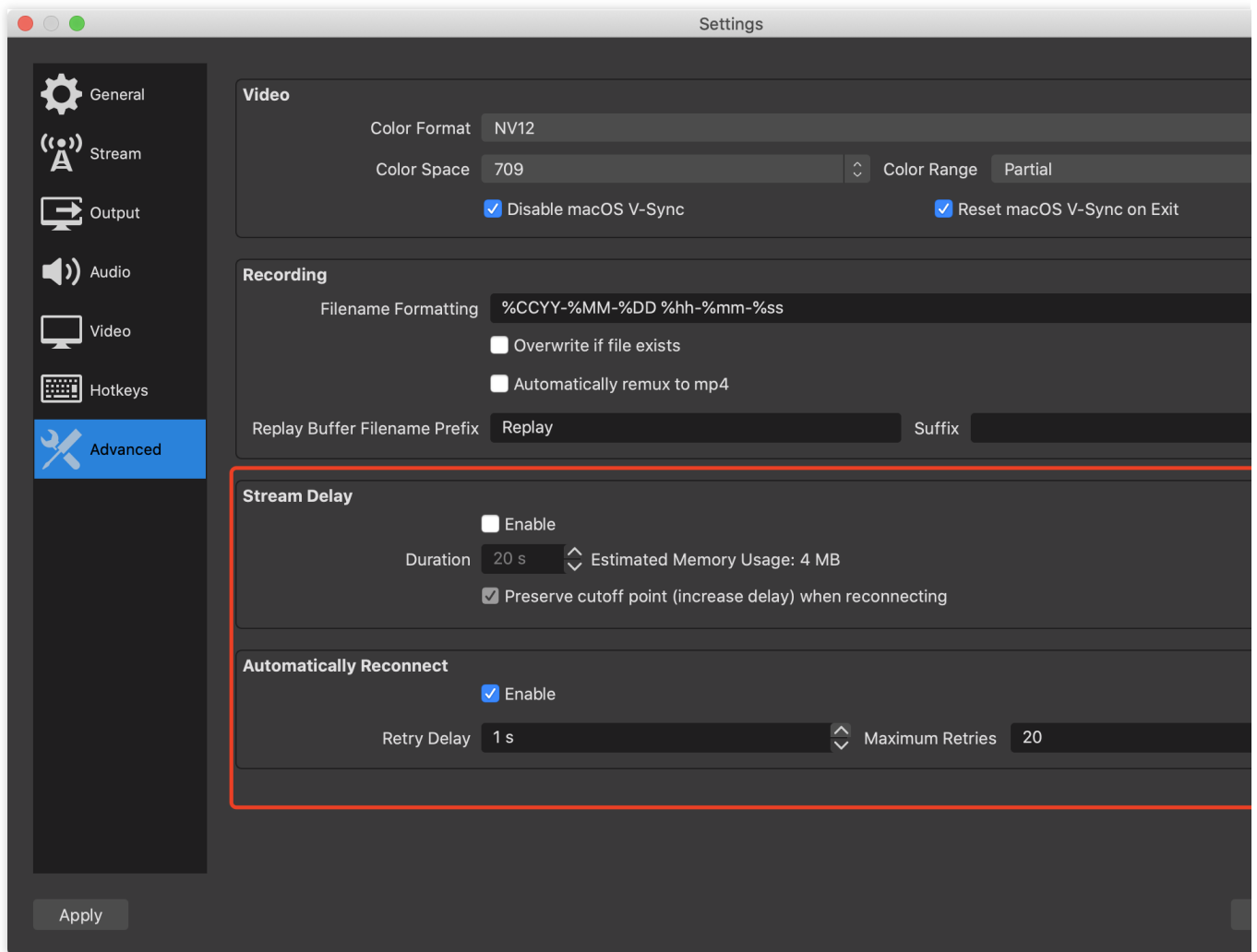
You can set video resolution and frame rate under the **Video** section of **Settings**. Resolution determines the clarity of video shown to audience members. The higher the resolution, the clearer the video. Frame rate (frames per second) determines playback smoothness. Typical frame rate falls in the range of 24 fps to 30 fps. Playback may stutter if frame rate is lower than 16 fps. Video games require higher frame rate and tend to stutter at a frame rate lower than 30 fps.



## Step 5. Configure advanced settings

To reduce end-to-end delay, we recommend you do not enable **Stream Delay**.

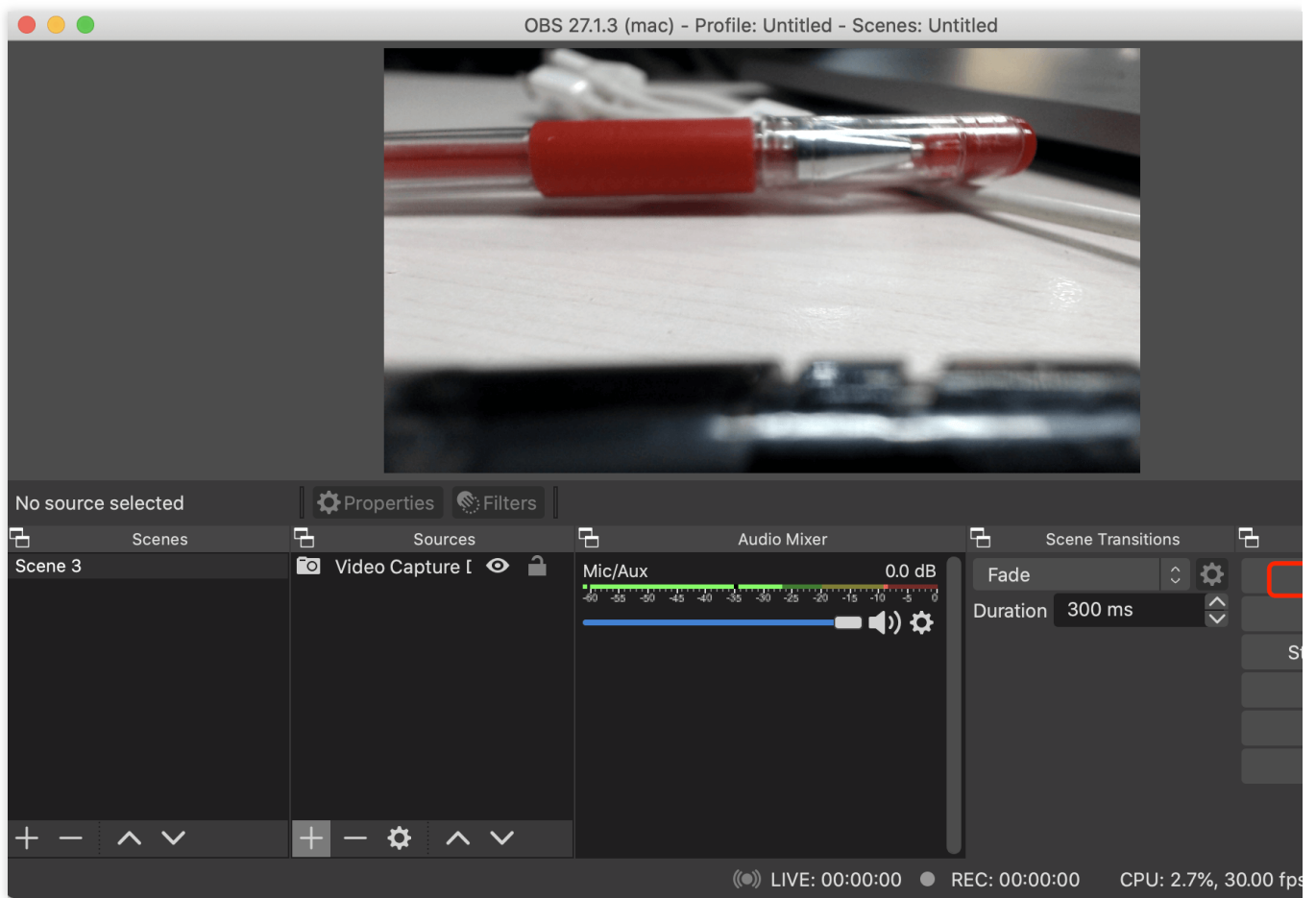
Keep **Automatically Reconnect** enabled and make **Retry Delay** as short as possible so that the publisher can be reconnected quickly after a disconnection occurs due to network jitter.



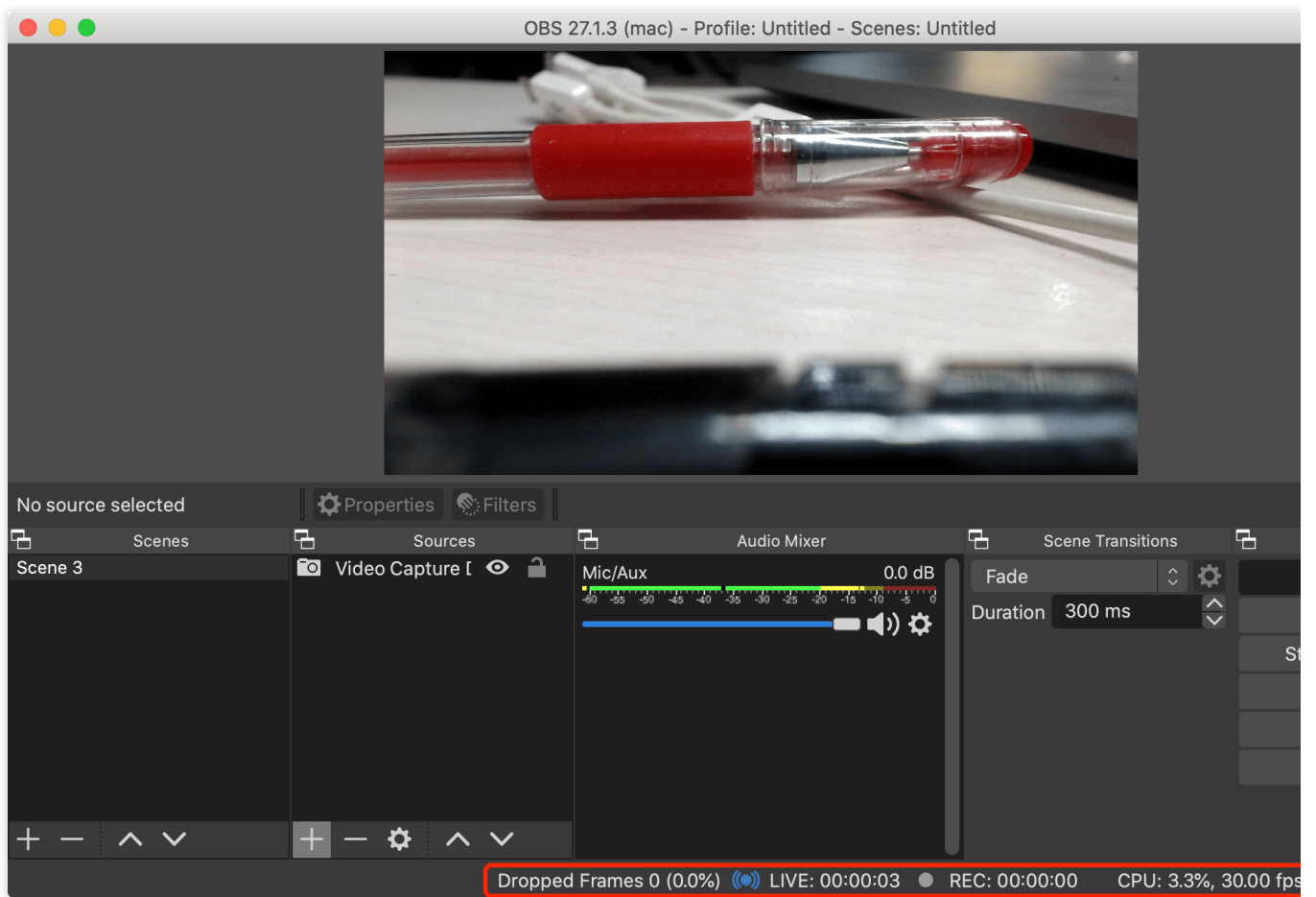
## Step 6. Publish the stream

1. In the **Controls** panel at the bottom, click **Start Streaming**.





2. If streaming is successful, the bottom bar will show streaming statistics, and the entry of a user will be recorded by the [monitoring dashboard](#).



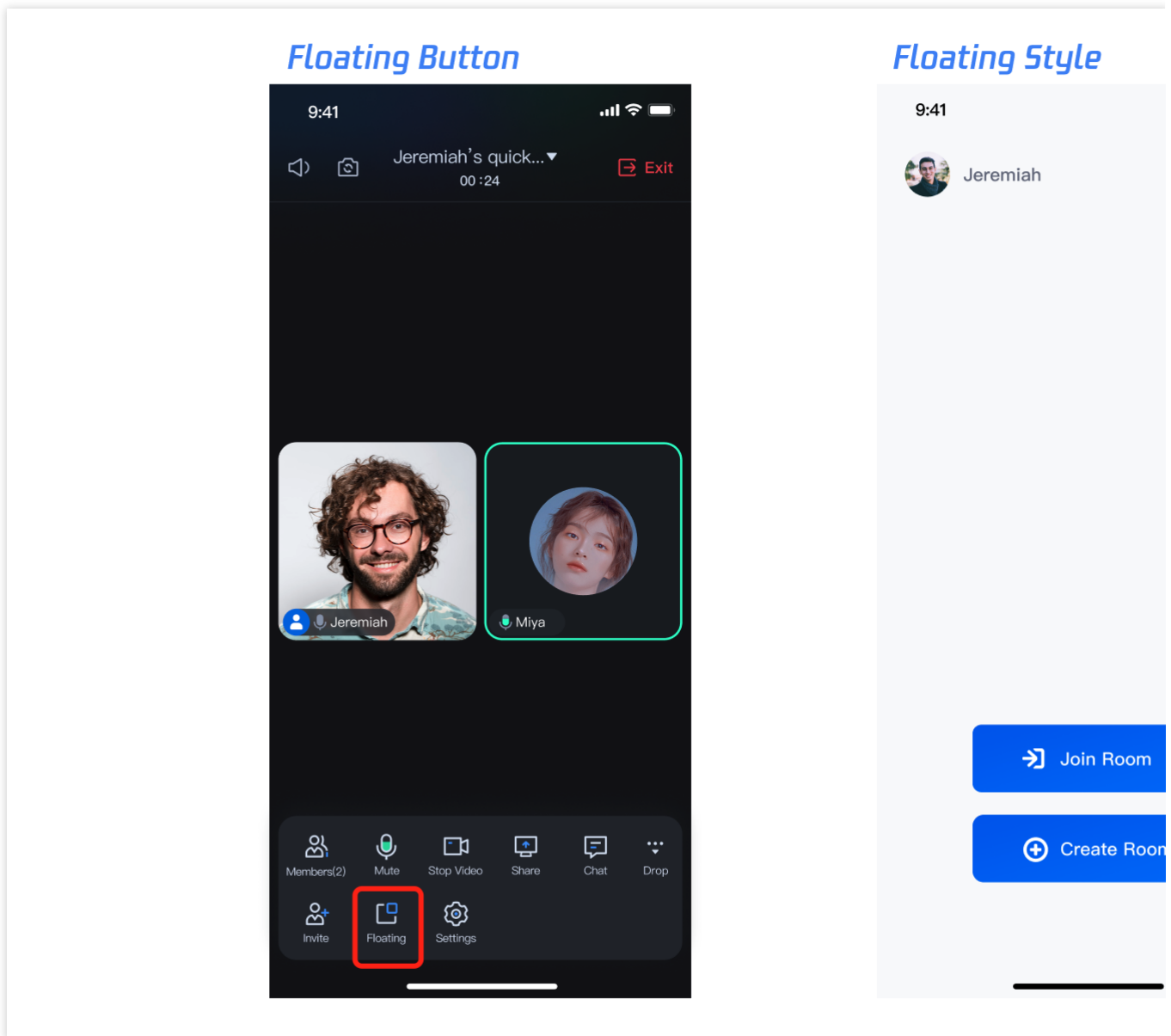
# Enhanced Features (TUIRoomKit)

## Floating Window

Last updated : 2024-05-24 16:28:14

## Feature Introduction

TUIRoomKit supports the floating window feature, allowing users to create a freely draggable floating window for displaying and managing the TUIRoomKit video conferencing interface. This makes it easier for users to handle other tasks while participating in a video conference.



## Supported Platforms

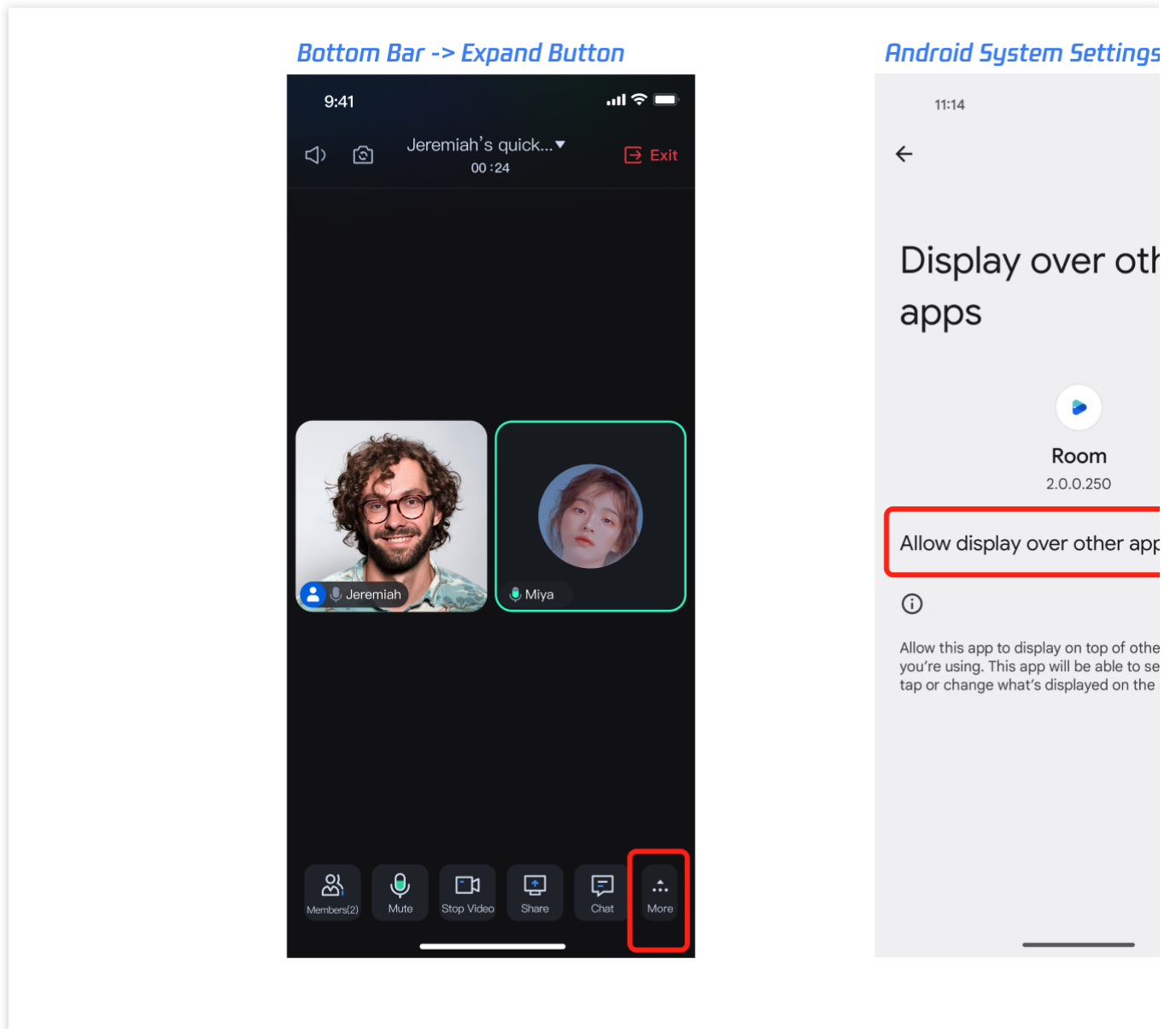
Android.

iOS.

## Operation Process

1. During the meeting, click on the **More > Floating** button in the bottom toolbar to enable the floating window.

2. When enabling the "Floating Window" feature for the first time, Android devices will be redirected to the relevant system settings page. You need to enable the appropriate permissions for the app, such as **Floating Window**, **Background Pop-up Interface**, **Allow display over other Apps** etc. The names of the relevant system settings may vary slightly between different device models.



3. After enabling the relevant system permissions, Android devices support both in-app and out-of-app floating windows, while iOS devices only support in-app floating windows.

4. In the floating window state, click on the floating window to return to the conference.

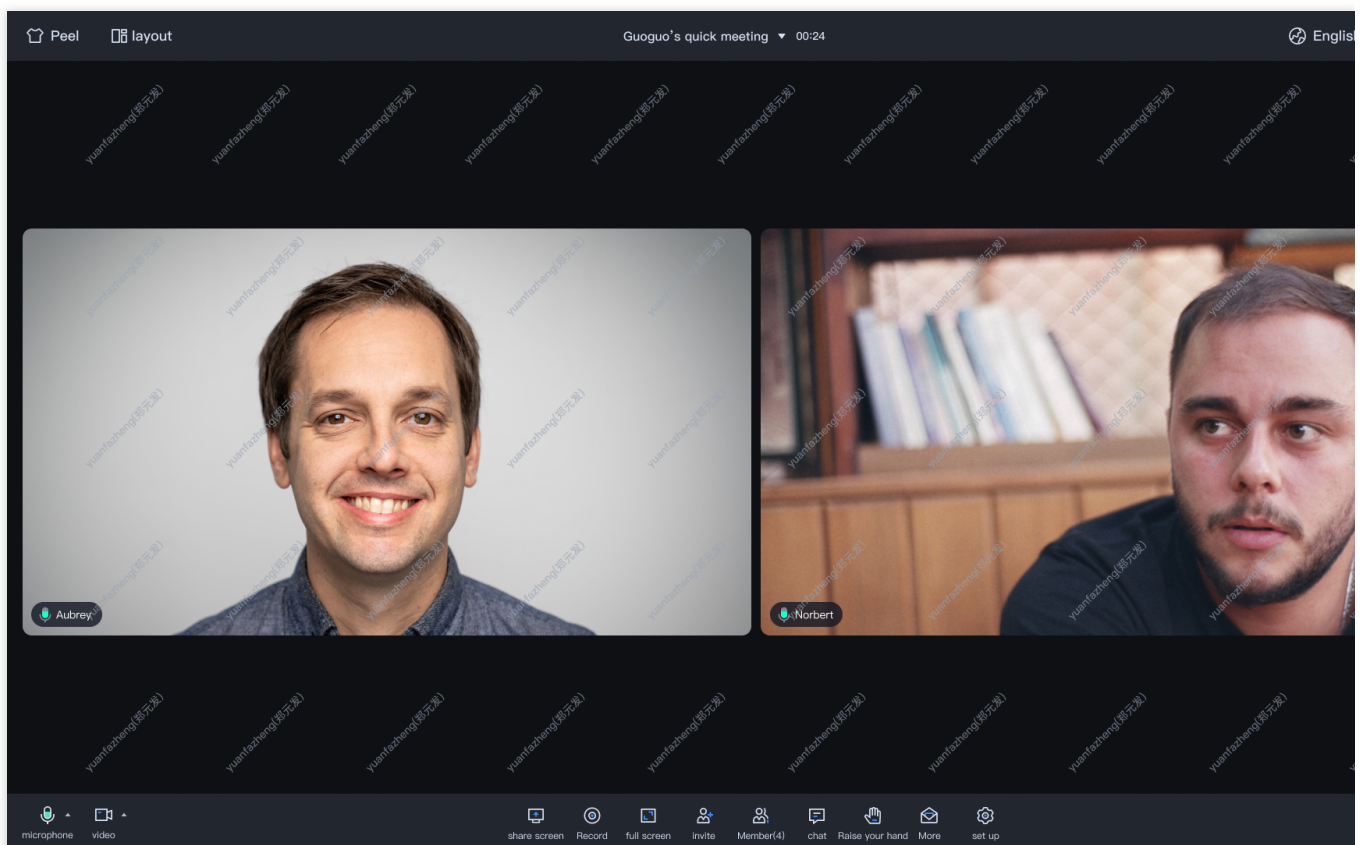
# Text Watermark

Last updated : 2024-05-30 13:03:08

TUIRoomKit has launched the Text Watermark feature, allowing users to set text watermarks during group meetings. This article will detail how to use this feature within the TUIRoomKit component.

## Integration effect

After integrating the Text Watermark feature in the TUIRoomKit component, the display effect is as follows:



## Preparation Requirements

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

## Enable Text Watermark

### Note:

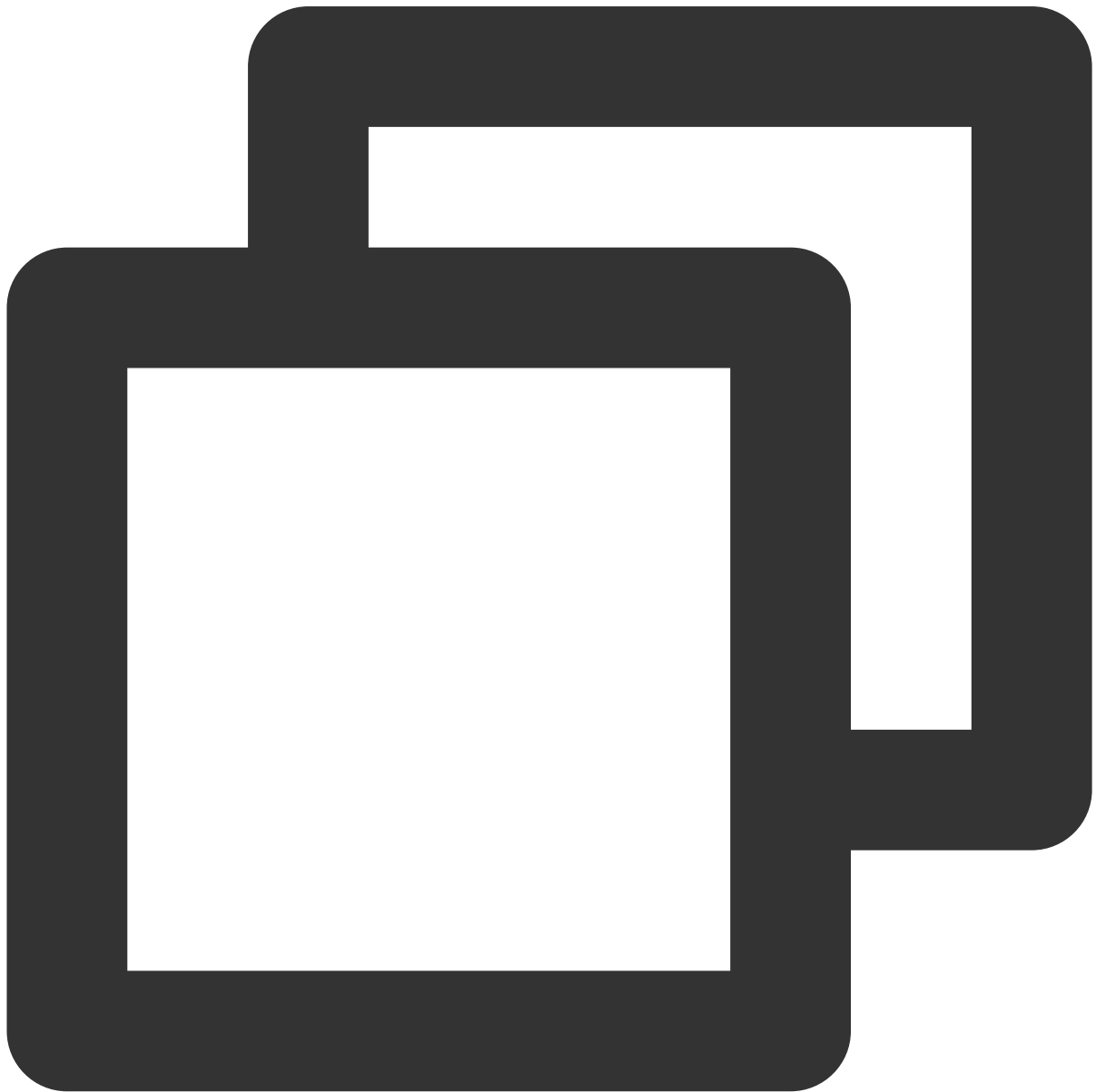
1. The Mini Program Platform does not support this feature yet, but both the Electron and Web versions are supported.

2. You need TUIRoomKit v2.3.3 or later.

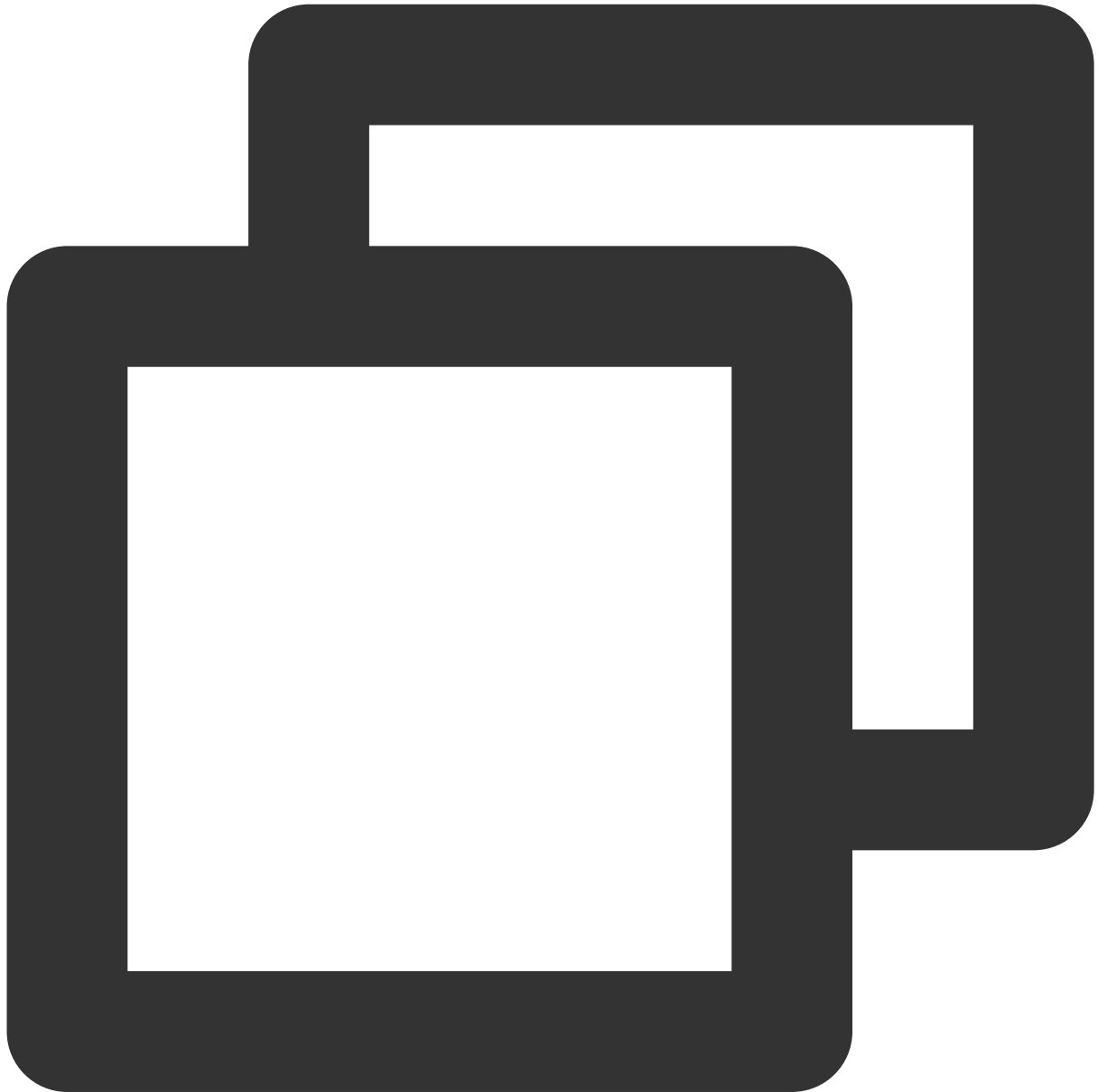
TUIRoomKit offers the Text Watermark feature. You can enable this feature by calling the [enableWatermark](#) interface.

Web

Electron



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableWatermark();
```



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.enableWatermark();
```



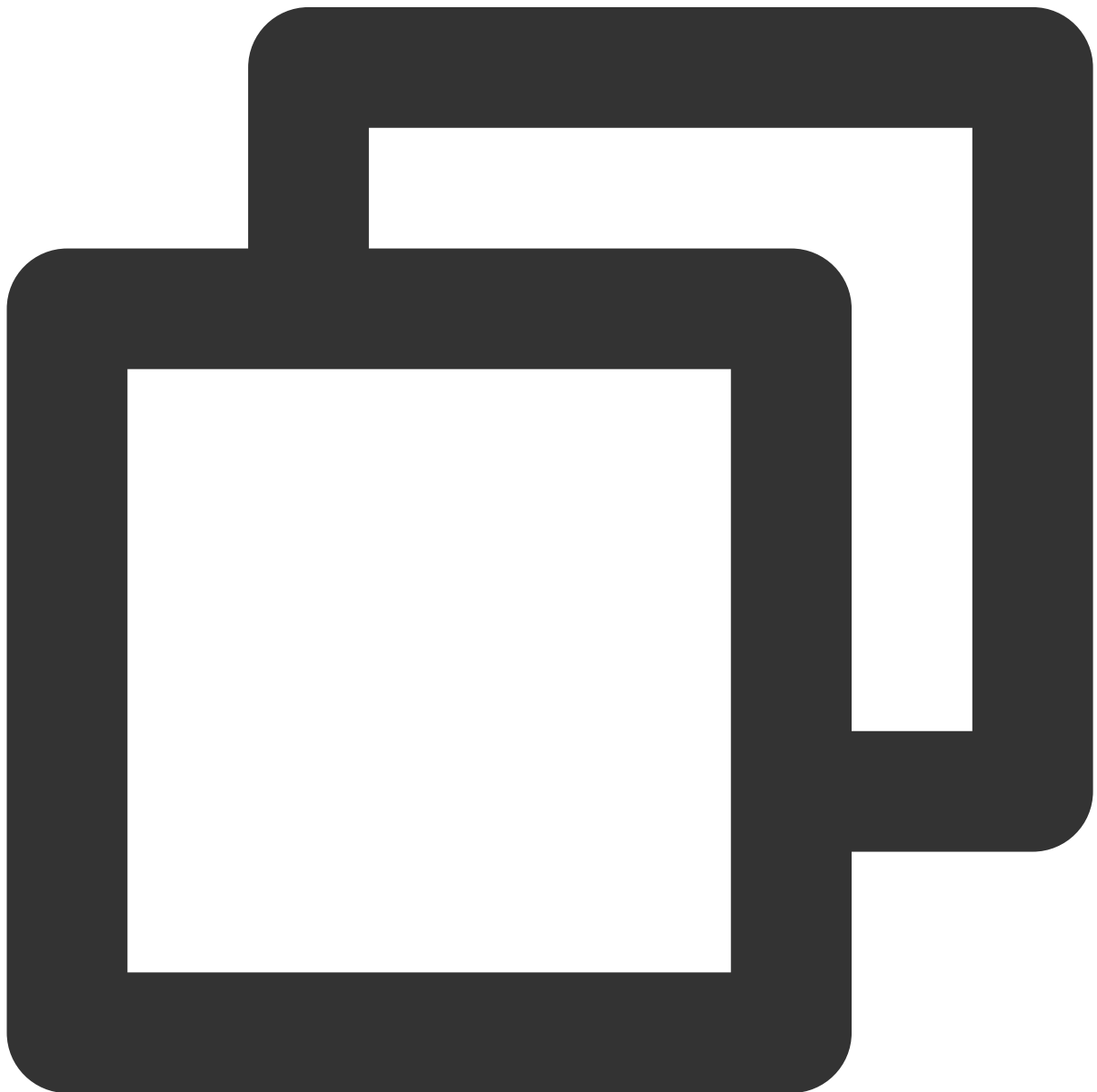
## FAQs

### 1. Why does the page style appear abnormal after enabling Text Watermark?

Please check the parent element with **id 'roomContainer'**, and confirm its `style` is set to: `width: 100%; height: 100%; overflow: hidden; .`

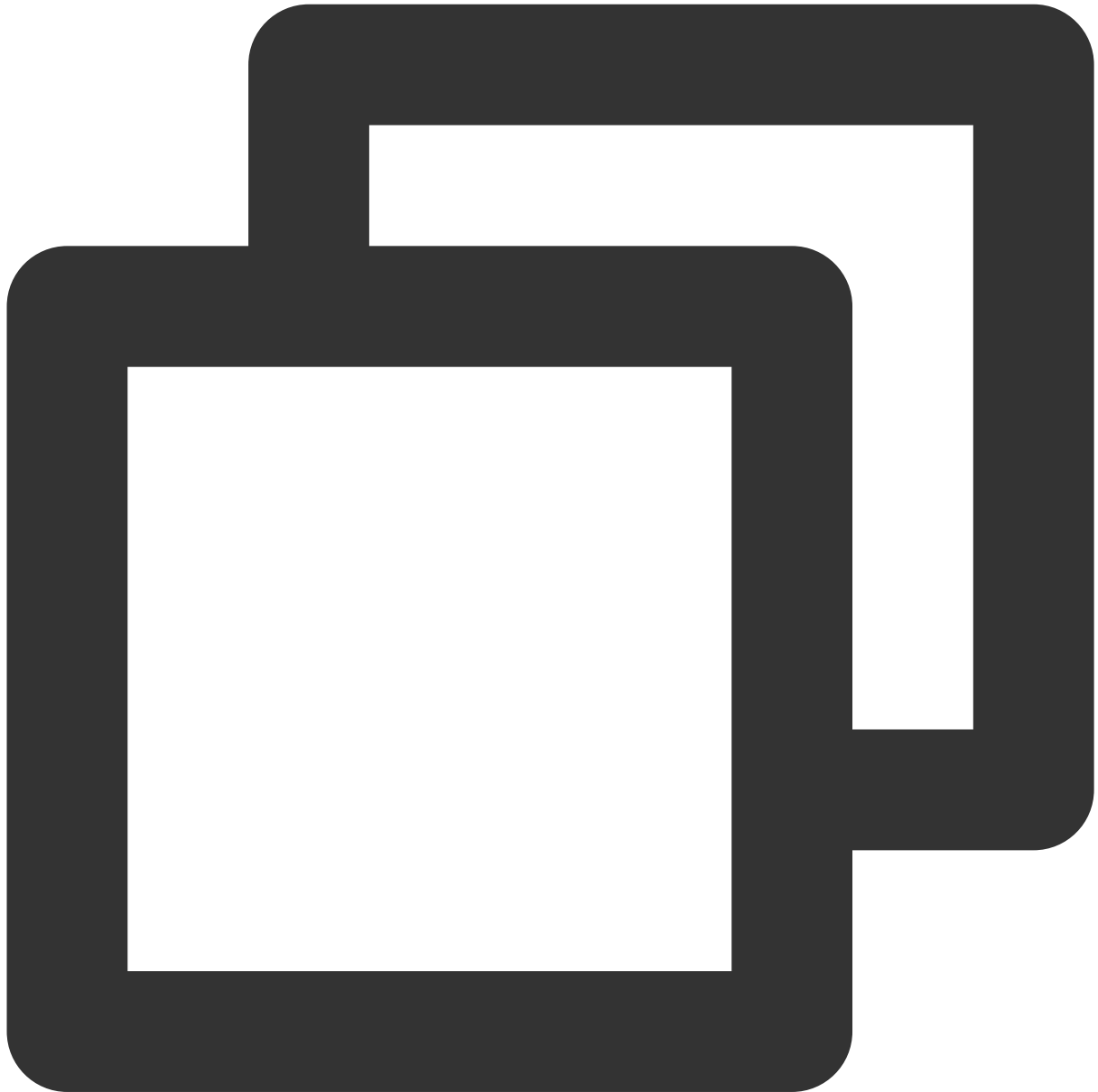
Web

Electron



```
// Note the package name. If you are using the vue2 version, please change the pack
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';  
conference.enableWatermark();
```



```
// Note the package name. If you are using the vue2 version, please change the pack  
import { conference } from '@tencentcloud/roomkit-electron-vue3';  
conference.enableWatermark();
```

# Setting Nickname, Avatar (All Platform)

Last updated : 2024-08-02 17:35:28

This document introduces how to set the user's avatar and nickname in TUIRoomKit.

## Setting Avatar, Nickname

If you need to customize your nickname or avatar, you can use the following API to update:

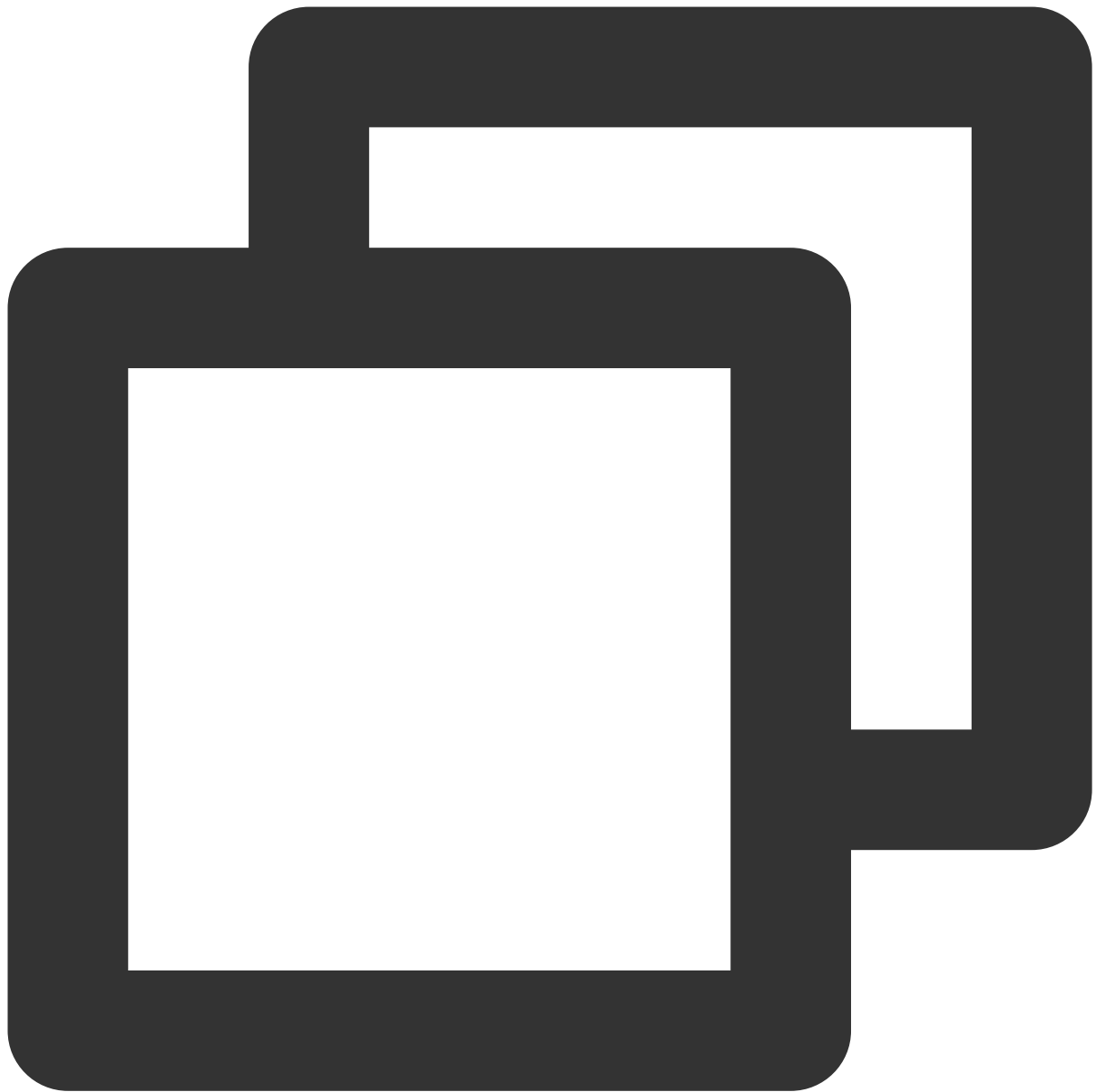
Web&H5

Android

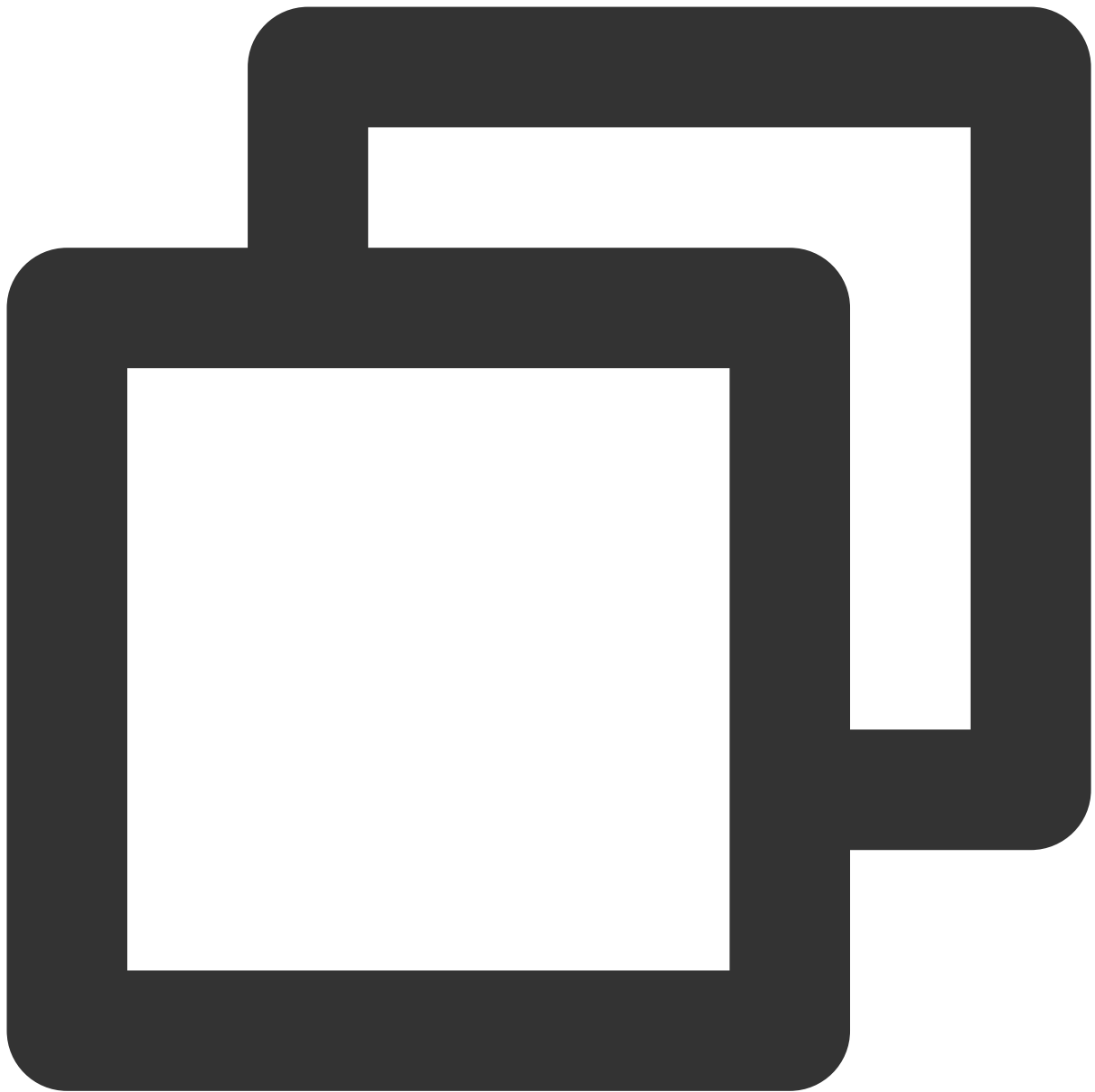
iOS

Flutter

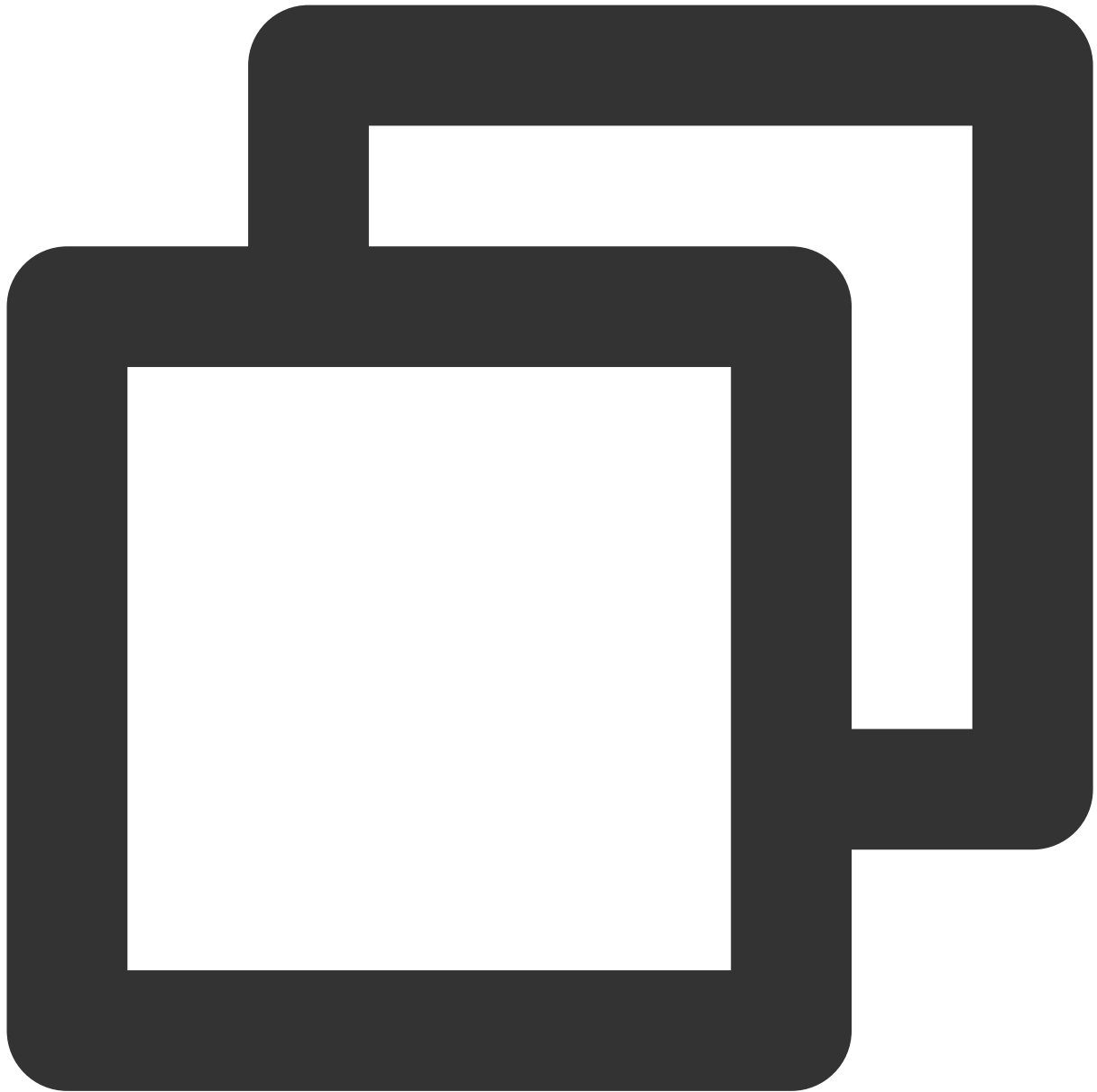
Electron



```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl: 'http://xxx' });
```

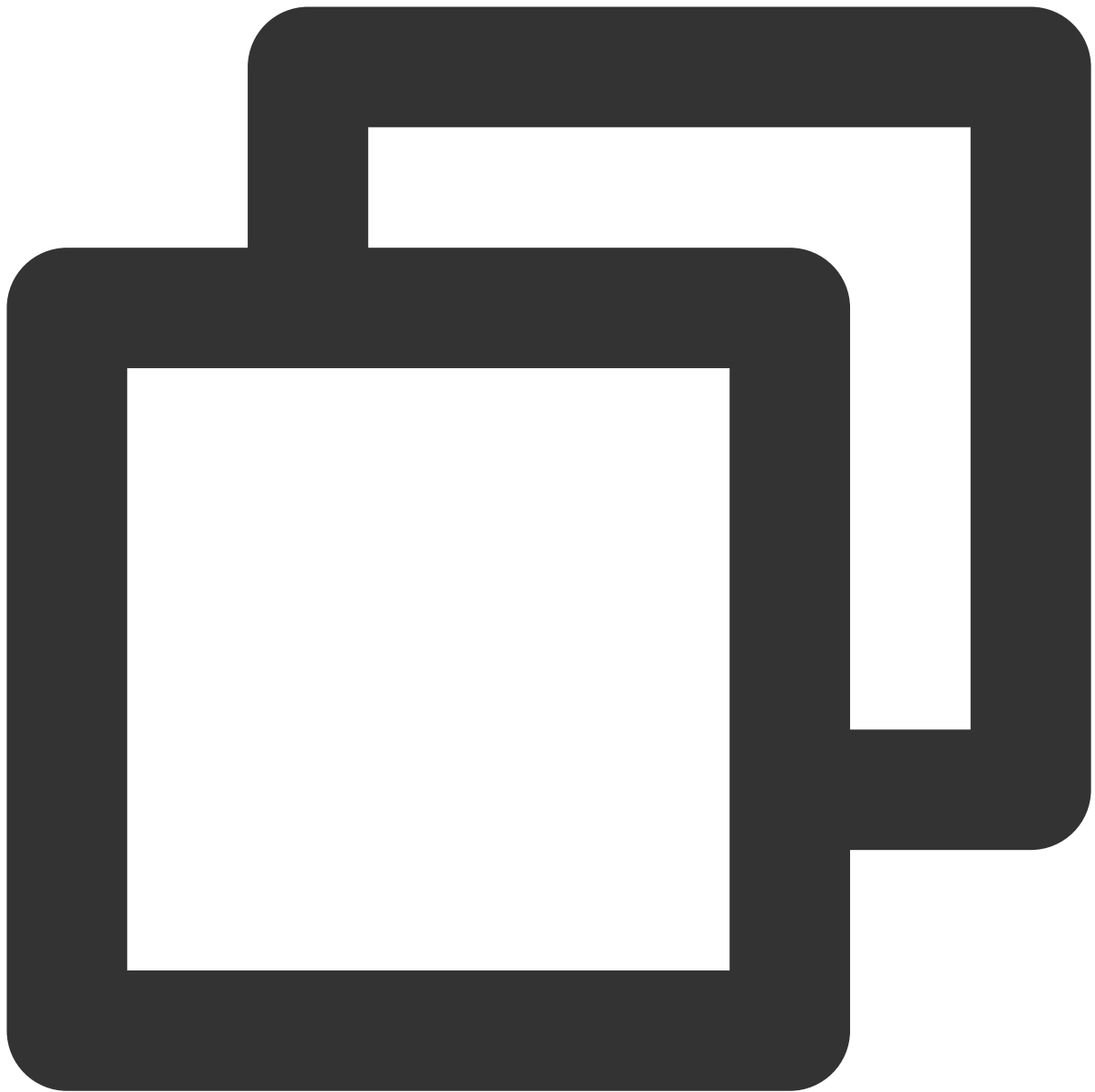


```
TUIRoomEngine.setSelfInfo("userName", "avatarUrl", null);
```

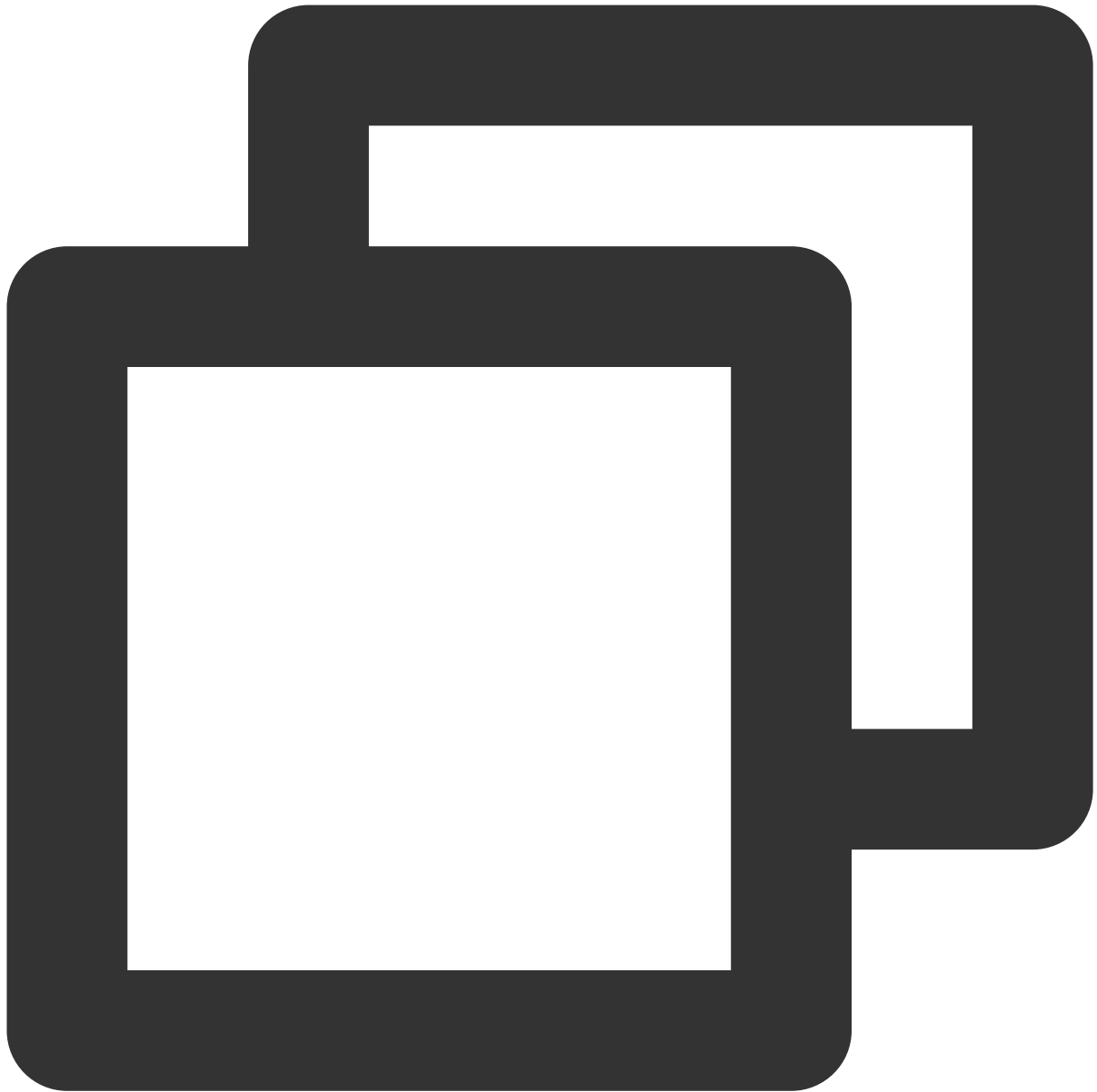


```
import TUIRoomEngine

TUIRoomEngine.setSelfInfo(userName: "xxx", avatarUrl: "xxx") {
    print("setSelfInfo success")
} onError: { code, message in
    print("setSelfInfo failed, code:\\(code),message:\\(message)")
}
```



```
import 'package:rtc_room_engine/rtc_room_engine.dart';  
  
TUIRoomEngine.setSelfInfo("userName", "avatarURL");
```



```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl: 'http://xxx' });
```

**Note:**

Due to user privacy restrictions, updates to the nickname and profile picture may be delayed. If you require higher real-time updates, you can use the [Rename during the conference](#) feature.

## Rename during the conference



In a meeting, participants can change their nicknames in real time to suit different scenarios. The updated nickname takes effect immediately, but is **limited to the current conference only**.

**Note :**

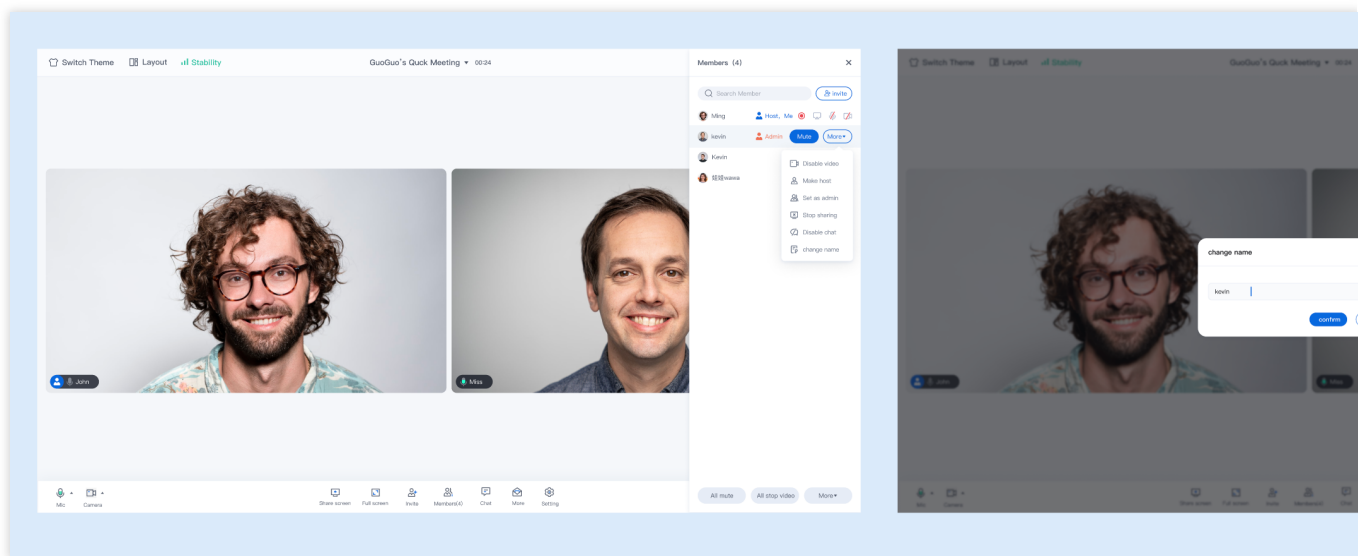
The Rename During Meeting feature requires **TUIRoomKit v2.5.0** or higher. Currently, this feature **supports** Web, Electron and H5 platforms **only**.

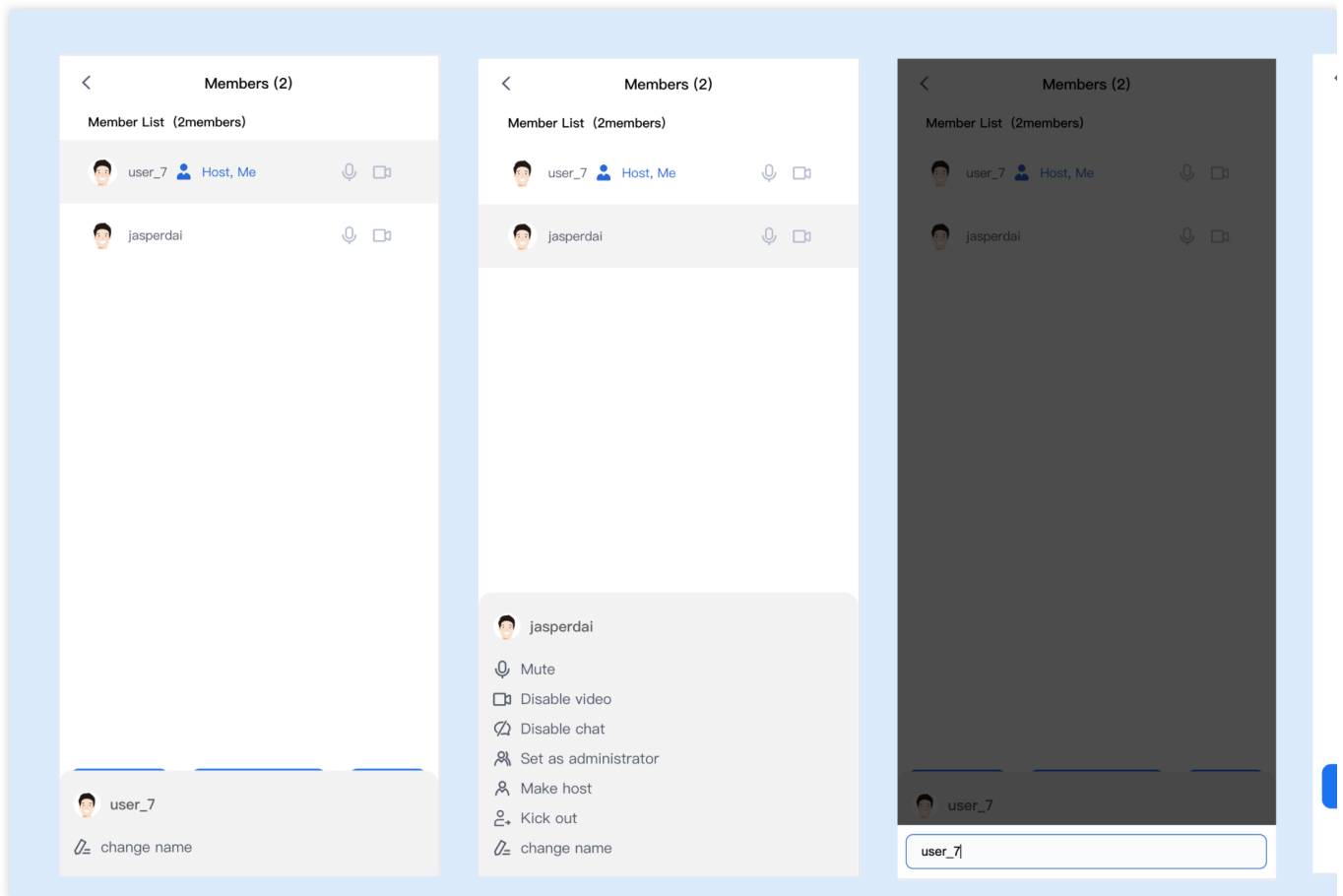
## Flowchart

1. Within TUIRoomKit, during the meeting, click the bottom toolbar **Member Management** > select yourself or the user you want to rename > **More** > **change name**;
2. In the pop-up window, enter the new name and click OK to apply the changes immediately.

Web&Electron

H5





## Permission to Operations

Ordinary users can only change their own nickname.

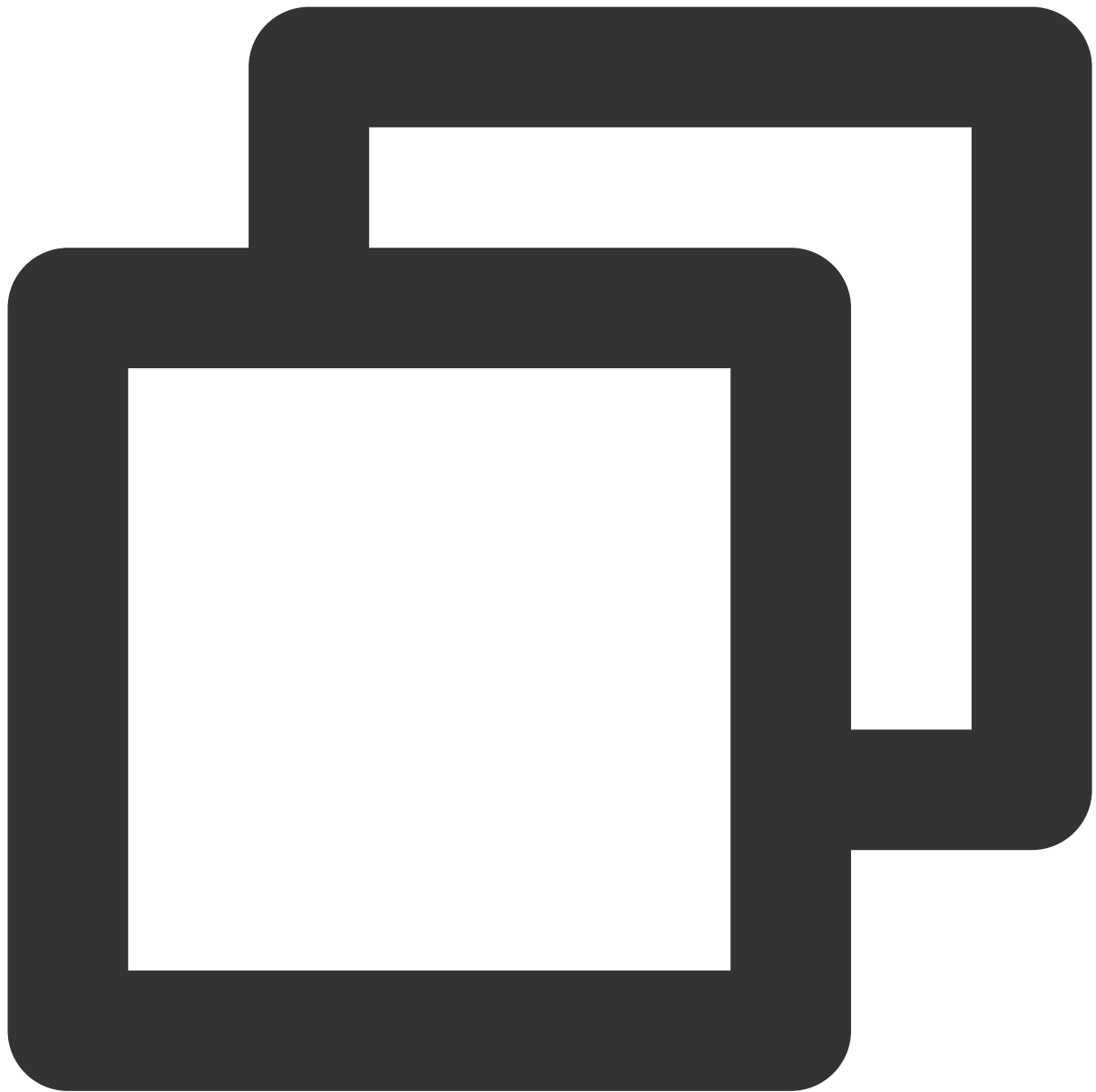
The host or admin can change their own or other users' nicknames.

## Sample code

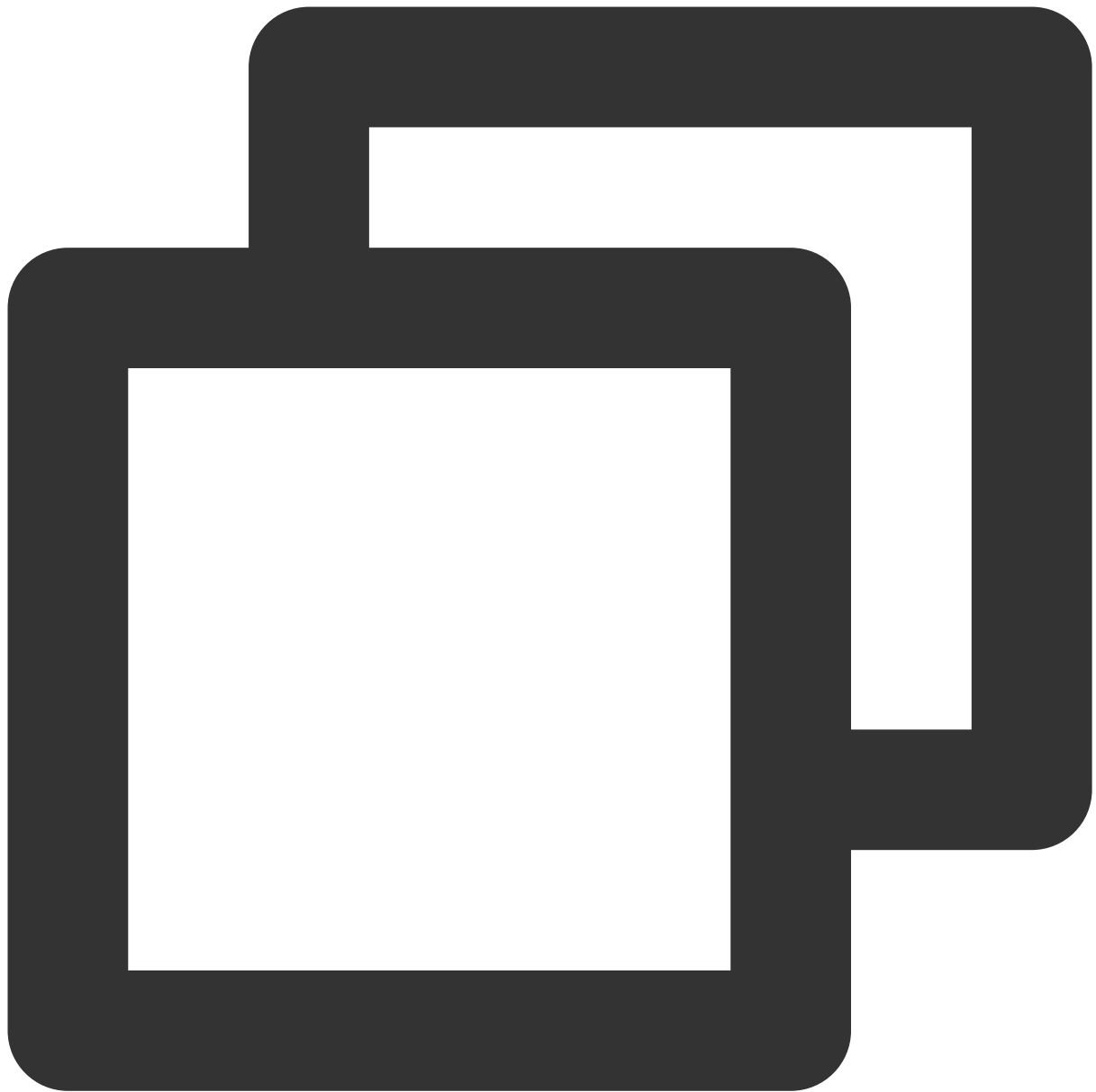
If you need to modify it yourself in your project to support the feature of changing nicknames during meetings, you can use the following **TUIRoomEngine** interface:

Web&H5

Electron



```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```



```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```

# Monitor Conference Status

Last updated : 2024-08-15 16:34:27

This article introduces the use of TUIRoomKit component meeting status callback.

## Conference status monitoring

If your business needs to **monitor the status of a conference**, such as the start and end of a conference, you can refer to the following code:

Android (Java)

Android (Kotlin)



```
ConferenceDefine.ConferenceObserver observer = new ConferenceDefine.ConferenceObser
@Override
public void onConferenceStarted(TUIRoomDefine.RoomInfo roomInfo, TUICommonDefin
}

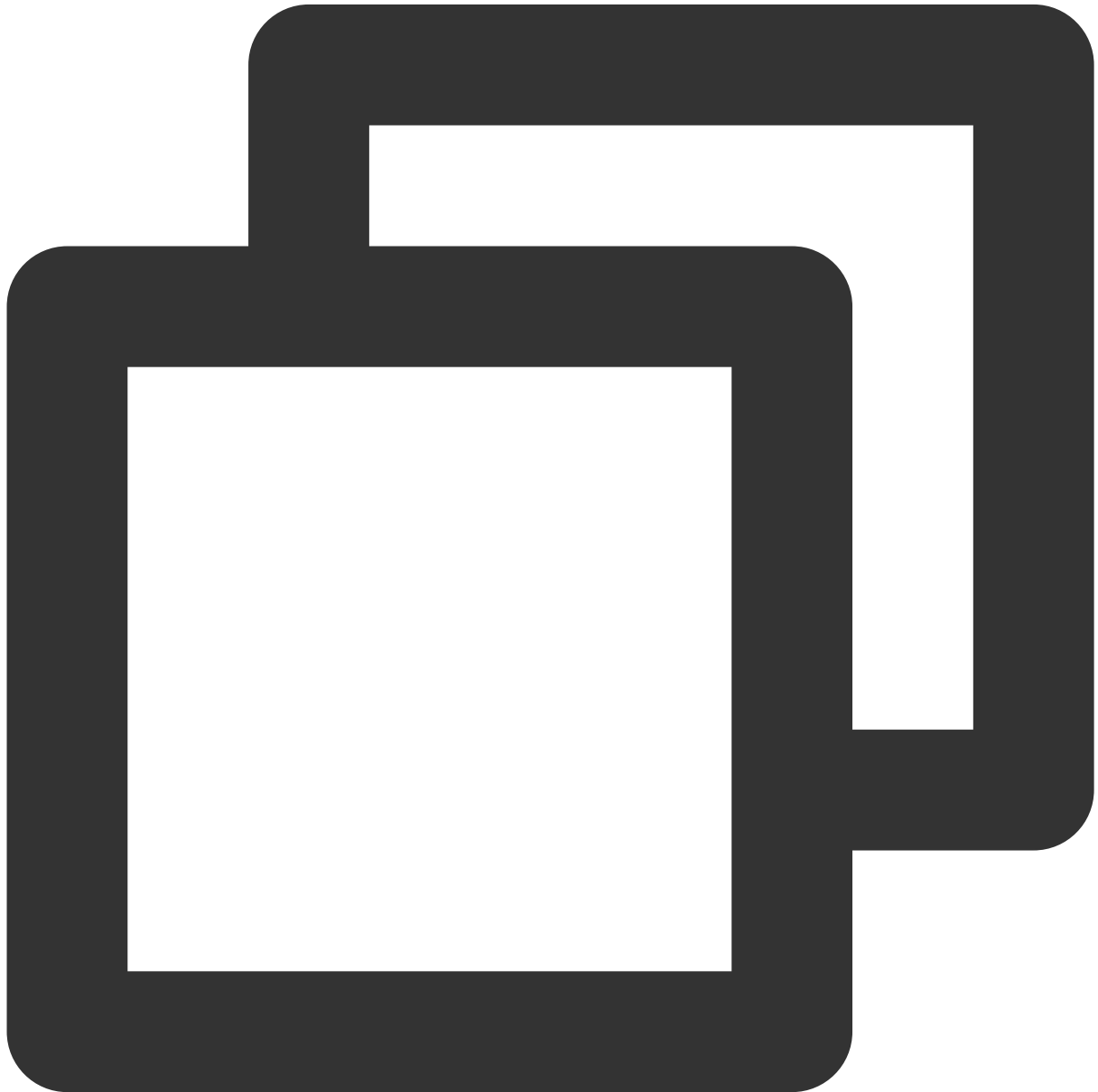
@Override
public void onConferenceJoined(TUIRoomDefine.RoomInfo roomInfo, TUICommonDefine
}

@Override
public void onConferenceExisted(String roomId) {
```

```
    }

    @Override
    public void onConferenceFinished(String roomId) {
    }

};
ConferenceSession.sharedInstance().addObserver(observer);
```



```
val observer: ConferenceObserver = object : ConferenceObserver() {
    override fun onConferenceStarted(roomInfo: TUIRoomDefine.RoomInfo?, error: TUIC
}
```

```
        override fun onConferenceJoined(roomInfo: TUIRoomDefine.RoomInfo?, error: TUICo
    }

    override fun onConferenceExisted(roomId: String?) {
    }

    override fun onConferenceFinished(roomId: String?) {
    }
}
ConferenceSession.sharedInstance().addObserver(observer)
```



# Client APIs (TUIRoomKit)

## iOS&Mac

## RoomKit API

Last updated : 2024-03-15 16:07:57

### Introduction

TUIRoomKit is an open source UI layer suite for conference SDK, currently only supports Swift language on the iOS platform. The conference UI can be called up through simple API calls.

### TUIRoomKit Interface

API	Description
<a href="#">createInstance</a>	Initialize the TUIRoomKit singleton object
<a href="#">destroyInstance</a>	Destroy the TUIRoomKit singleton object
<a href="#">setSelfInfo</a>	Set user information (avatar, nickname) (optional)
<a href="#">createRoom</a>	Create a room
<a href="#">enterRoom</a>	Enter the room

#### **createInstance**

Initialize the TUIRoomKit singleton object.



```
public class func createInstance() -> TUIRoomKit
```

### **destroyInstance**

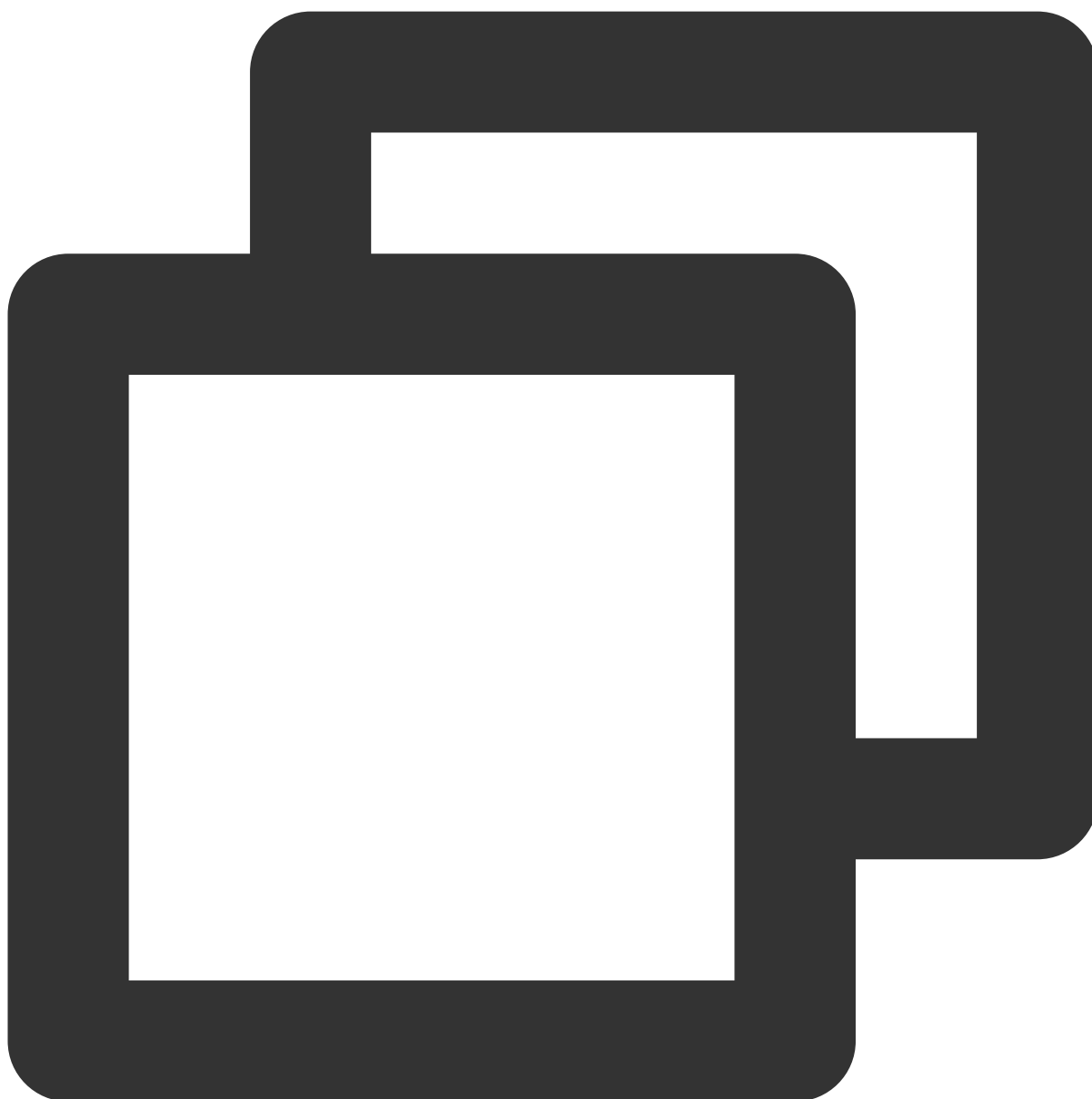
Destroy the TUIRoomKit singleton object.



```
public class func destroyInstance() -> Void
```

### **setSelfInfo(optional)**

Set user information (avatar, nickname).



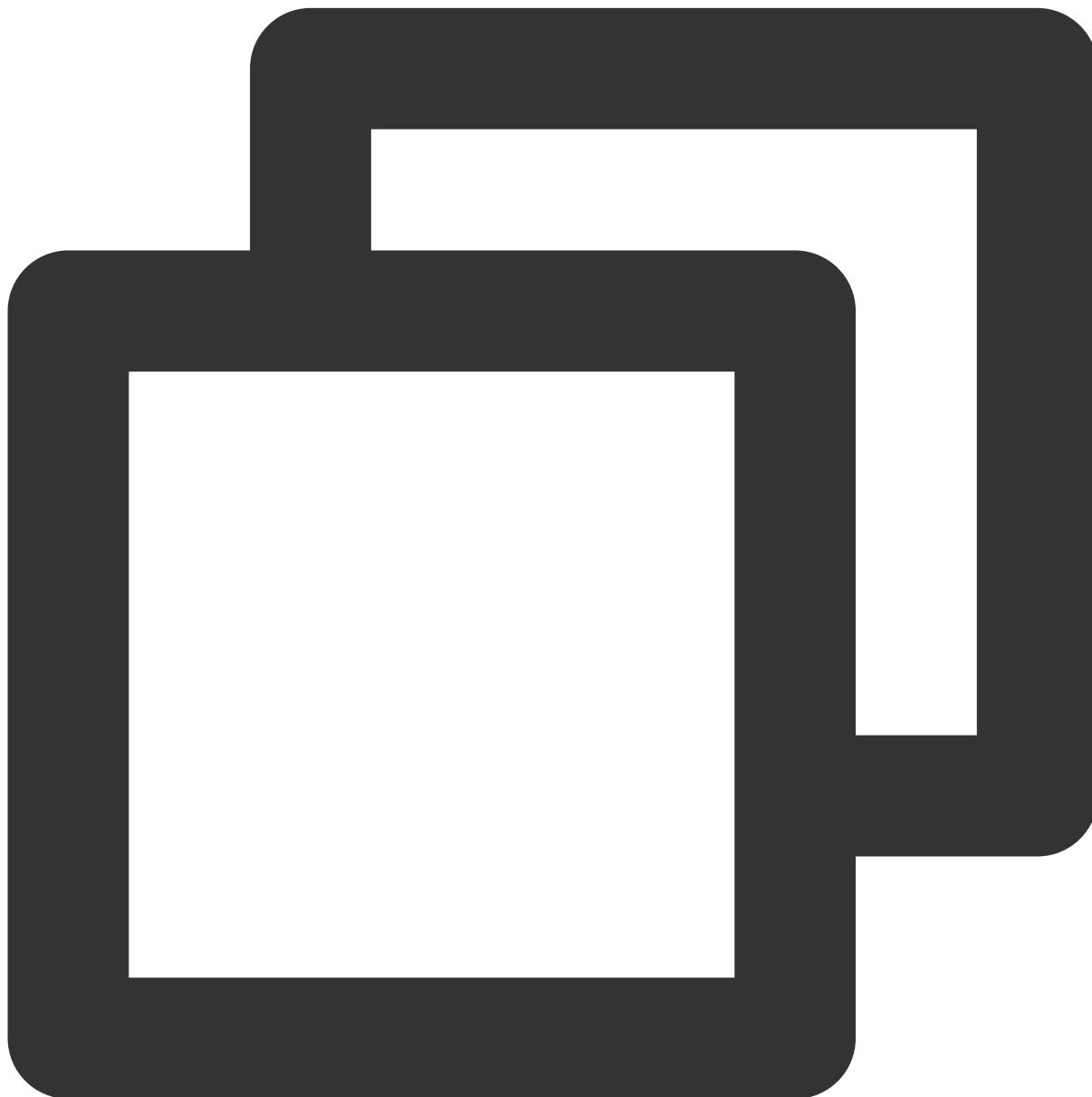
```
public func setSelfInfo(userName: String,  
                        avatarURL: String,  
                        onSuccess: @escaping TUISuccessBlock,  
                        onError: @escaping TUIErrorBlock) -> Void
```

Parameter	Type	Meaning
userName	String	Username
avatarURL	String	User avatar URL

onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failure callback

## createRoom

Create a room.



```
public func createRoom(roomInfo: TUIRoomInfo,  
                        onSuccess: @escaping TUISuccessBlock,  
                        onError: @escaping TUIErrorBlock) -> Void
```

The parameters are as follows:

Parameter	Type	Meaning
roomInfo	<a href="#">TUIRoomInfo</a>	Room data
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failure callback

## enterRoom

Enter the room.



```
public func enterRoom(roomId: String,  
    enableAudio: Bool,  
    enableVideo: Bool,  
    isSoundOnSpeaker: Bool,  
    onSuccess: @escaping TUISuccessBlock,  
    onError: @escaping TUIErrorBlock) -> Void
```

The parameters are as follows:

Parameter	Type	Meaning

roomId	String	Room ID string
enableAudio	Bool	Whether to turn on the audio when entering the room
enableVideo	Bool	Whether to turn on the video when entering the room
isSoundOnSpeaker	Bool	Whether to turn on the speakers when entering the room
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failure callback



# RoomEngine API

## API Overview

Last updated : 2023-12-18 18:01:32

### TUIRoomEngine API List

TUIRoomEngine API is the UI-free interface of the Conference Component, which allows you to customize the encapsulation according to your business needs.

TUIRoomEngine

#### TUIRoomEngine Core Methods

API	Description
<a href="#">init</a>	Create Instance of TUIRoomEngine.
<a href="#">login</a>	Login Interface, you need to initialize user information before entering the room and perform a series of operations.
<a href="#">logout</a>	Logout Interface, there will be active room leaving operation and resources destruction.
<a href="#">setSelfInfo</a>	Set local user name and avatar.
<a href="#">getSelfInfo</a>	Get the basic information of the local user login.
<a href="#">addObserver</a>	Set Event Callback.
<a href="#">removeObserver</a>	Remove Event Callback.

#### Active Interface related to the room

API	Description
<a href="#">createRoom</a>	Create a room.
<a href="#">destroyRoom</a>	Close the room.
<a href="#">enterRoom</a>	Entered room.
<a href="#">exitRoom</a>	Leave the room.

<a href="#">connectOtherRoom</a>	Connect to another room.
<a href="#">disconnectOtherRoom</a>	Disconnect from another room.
<a href="#">fetchRoomInfo</a>	Get Room data.
<a href="#">updateRoomNameByAdmin</a>	Update Room ID (only administrator or group owner can call).
<a href="#">updateRoomSpeechModeByAdmin</a>	Set Mic Control Mode for the room (only administrator or group owner can call).

## Local user view rendering and video management

API	Description
<a href="#">setLocalVideoView</a>	Set the View Control for local user video rendering.
<a href="#">openLocalCamera</a>	Open local Camera.
<a href="#">closeLocalCamera</a>	Close local Camera.
<a href="#">updateVideoQuality</a>	Update Encoding Quality settings for local video.
<a href="#">startPushLocalVideo</a>	Start pushing local video.
<a href="#">stopPushLocalVideo</a>	Stop pushing local video.
<a href="#">startScreenCapture</a>	Start Screen Sharing (this interface is only supported on mobile devices).
<a href="#">startScreenCapture</a>	Start Screen Sharing (this interface is only supported on Mac OS desktop systems).
<a href="#">stopScreenCapture</a>	End Screen Sharing.
<a href="#">getScreenCaptureSources</a>	Enumerate shareable screens and windows (this interface is only supported on Mac OS systems).
<a href="#">selectScreenCaptureTarget</a>	Select the screen or window to share (this interface is only supported on Mac OS systems).

## Local user audio management

API	Description
<a href="#">openLocalMicrophone</a>	Open local mic.
<a href="#">closeLocalMicrophone</a>	Close local mic.

<a href="#">updateAudioQuality</a>	Update local Audio Encoding Quality settings.
<a href="#">startPushLocalAudio</a>	Start pushing local audio.
<a href="#">stopPushLocalAudio</a>	Stop pushing local audio.

## Remote user view rendering and video management

API	Description
<a href="#">setRemoteVideoView</a>	Set the View Control for remote user video rendering.
<a href="#">startPlayRemoteVideo</a>	Start Playback of remote user video.
<a href="#">stopPlayRemoteVideo</a>	Stop Playback of remote user video.
<a href="#">muteRemoteAudioStream</a>	Mute remote user.

## Room user information

API	Description
<a href="#">getUserList</a>	Get the member list in the room.
<a href="#">getUserInfo</a>	Get member information.

## Room user management

API	Description
<a href="#">changeUserRole</a>	Modify user role (only administrator or group owner can call).
<a href="#">kickRemoteUserOutOfRoom</a>	Kick remote user out of the room (only administrator or group owner can call).

## Room user speech management

API	Description
<a href="#">disableDeviceForAllUserByAdmin</a>	Control the permission status of all users in the current room to open audio streams, video streams, and capture devices, such as: all users are prohibited from opening mics, all users are prohibited from opening cameras, all users are prohibited from opening screen sharing (currently only available in conference scenarios, and only administrators or group owners can call).

<a href="#">openRemoteDeviceByAdmin</a>	Request remote user to open media device (only administrator or group owner can call).
<a href="#">closeRemoteDeviceByAdmin</a>	Close remote user media device (only administrator or group owner can call).
<a href="#">applyToAdminToOpenLocalDevice</a>	Request to open local media device (available for ordinary users).

## Room mic seat management

API	Description
<a href="#">setMaxSeatCount</a>	Set the maximum number of mic seats (only supported when entering the room and creating the room).
<a href="#">getSeatList</a>	Get the list of mic seats.
<a href="#">lockSeatByAdmin</a>	Lock the mic seat (only administrator or group owner can call, including position lock, audio status lock, and video status lock).
<a href="#">takeSeat</a>	Apply to Go Live (no need to apply in free speech mode).
<a href="#">leaveSeat</a>	Apply to leave the mic (no need to apply in free speech mode).
<a href="#">takeUserOnSeatByAdmin</a>	Host/Administrator invites user to Go Live.
<a href="#">kickUserOffSeatByAdmin</a>	Host/Administrator kicks user off the mic.

## Signaling management

API	Description
<a href="#">cancelRequest</a>	Cancel Request.
<a href="#">responseRemoteRequest</a>	Reply Request.

## Send message

API	Description
<a href="#">sendTextMessage</a>	Send Text Message.

<a href="#">sendCustomMessage</a>	Send Custom Message.
<a href="#">disableSendingMessageByAdmin</a>	Disable remote user's ability to send text messages (only administrator or group owner can call).
<a href="#">disableSendingMessageForAllUser</a>	Disable all users' ability to send text messages (only administrator or group owner can call).

### Advanced features: Get TRTC instance

API	Description
<a href="#">getDeviceManager</a>	Get native TRTC Device Management class.
<a href="#">getAudioEffectManager</a>	Get native TRTC Sound Effect Class.
<a href="#">getBeautyManager</a>	Get native TRTC Beauty Class.
<a href="#">getTRTCCloud</a>	Get native TRTC Instance Class.

## TUIRoomObserver Callback Events

TUIRoomObserver is the callback event class corresponding to TUIRoomEngine. You can listen to the callback events you need through this callback.

TUIRoomObserver

## TUIRoomObserver

### Error callback

API	Description
<a href="#">onError</a>	Error Event Callback.

### Login status event callback

API	Description
<a href="#">onKickedOffLine</a>	Other terminal login is kicked offline.

[onUserSigExpired](#)

User Credential Timeout Event.

**Room event callback**

API	Description
<a href="#">onRoomNameChanged</a>	Room ID change event.
<a href="#">onAllUserMicrophoneDisableChanged</a>	All users in the room have their mics disabled event.
<a href="#">onAllUserCameraDisableChanged</a>	All users in the room have their cameras disabled event.
<a href="#">onSendMessageForAllUserDisableChanged</a>	All users in the room have their text message sending disabled event.
<a href="#">onKickedOutOfRoom</a>	Kicked out of the room event.
<a href="#">onRoomDismissed</a>	Room closed event.
<a href="#">onRoomSpeechModeChanged</a>	Room Mic Control Mode changed.

**Room user event callback**

API	Description
<a href="#">onRemoteUserEnterRoom</a>	Remote user entered room event.
<a href="#">onRemoteUserLeaveRoom</a>	Remote user left the room event.
<a href="#">onUserRoleChanged</a>	User role changed event.
<a href="#">onUserVideoStateChanged</a>	User video status changed event.
<a href="#">onUserAudioStateChanged</a>	User audio status changed event.
<a href="#">onUserScreenCaptureStopped</a>	User screen capture stopped event.
<a href="#">onUserVoiceVolumeChanged</a>	User volume change event.
<a href="#">onSendMessageForUserDisableChanged</a>	User text message sending ability changed event.
<a href="#">onUserNetworkQualityChanged</a>	User network status change event.

**Room mic seat event callback**

API	Description

<a href="#">onRoomMaxSeatCountChanged</a>	Maximum mic seat changed event in the room (only effective in conference type rooms).
<a href="#">onSeatListChanged</a>	Mic seat list changed event.
<a href="#">onKickedOffSeat</a>	Received user kicked off mic event.

### Request signaling event callback

API	Description
<a href="#">onRequestReceived</a>	Received request message event.
<a href="#">onRequestCancelled</a>	Received request cancelled event.

### Room message event callback

API	Description
<a href="#">onReceiveTextMessage</a>	Received normal text message event.
<a href="#">onReceiveCustomMessage</a>	Received custom message event.

# TUIRoomEngine

Last updated : 2023-12-18 17:57:07

## TUIRoomEngine API Introduction

TUIRoomEngine API is a No UI Interface for Conference Component, you can use this API to custom encapsulate according to your business needs.

### **init**

Create TUIRoomEngine instance



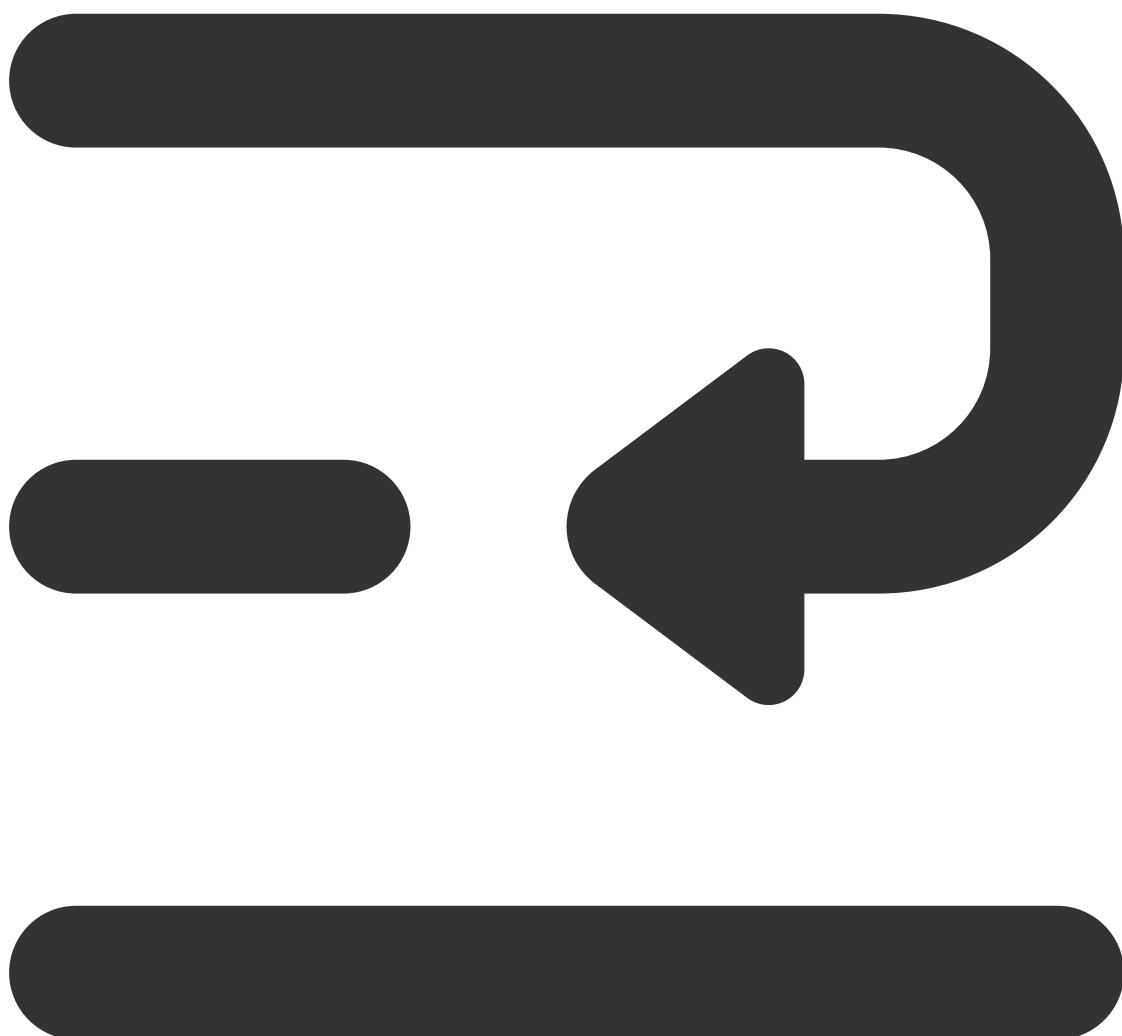


```
TUIRoomEngine *roomEngine = [[TUIRoomEngine alloc] init];
```

## login

Login interface, you need to initialize user information before entering the room and perform a series of operations

**Note:** In v1.0.0, this interface is named setup, and in v1.0.1 and above, please use TUIRoomEngine.login to log in to TUIRoomEngine.





```
+ (void)loginWithSDKAppId:(NSInteger) sdkAppId
    userId:(NSString *)userId
    userSig:(NSString *)userSig
    onSuccess:(TUISuccessBlock) onSuccess
    onError:(TUIErrorBlock) onError;
```

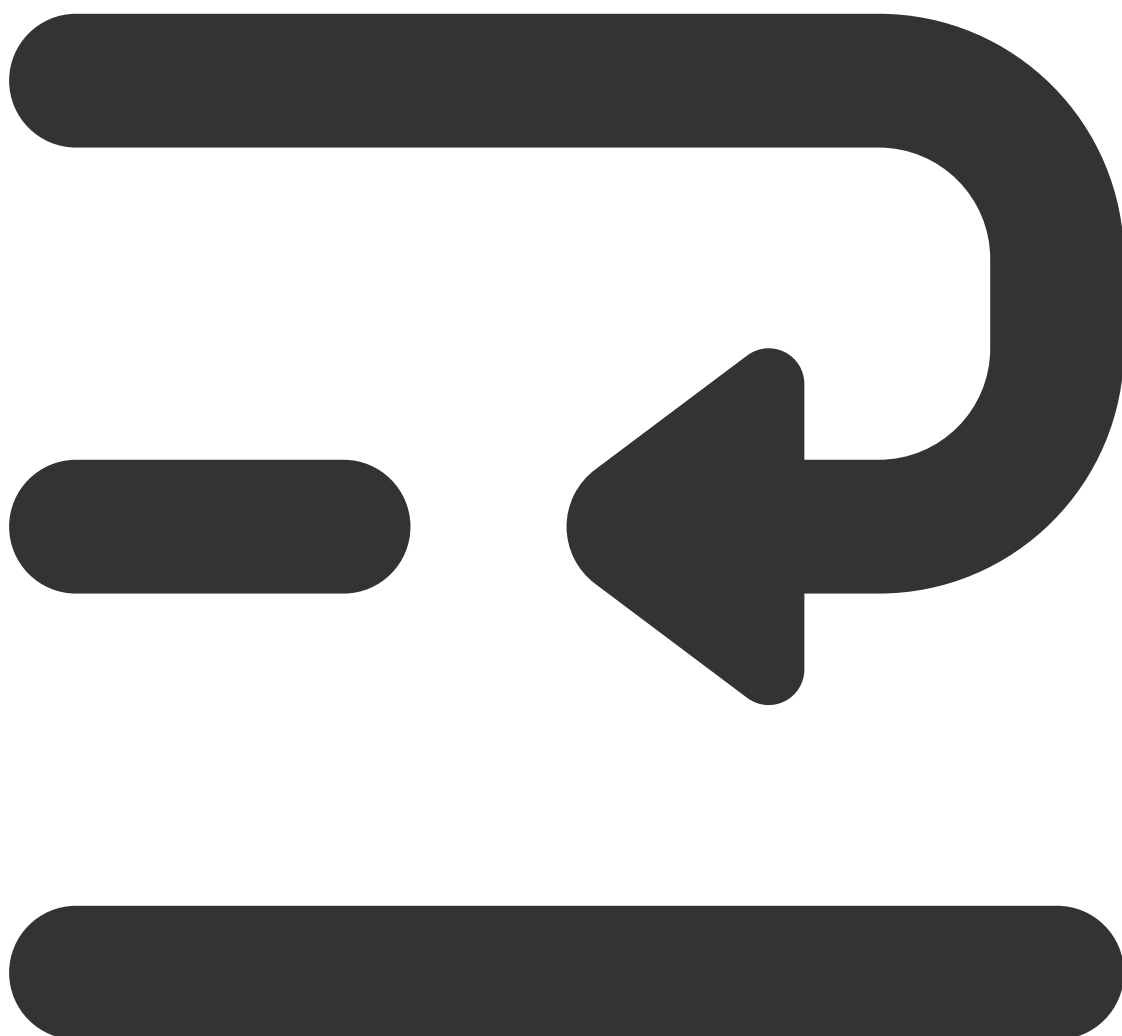
**The parameters are as follows:**

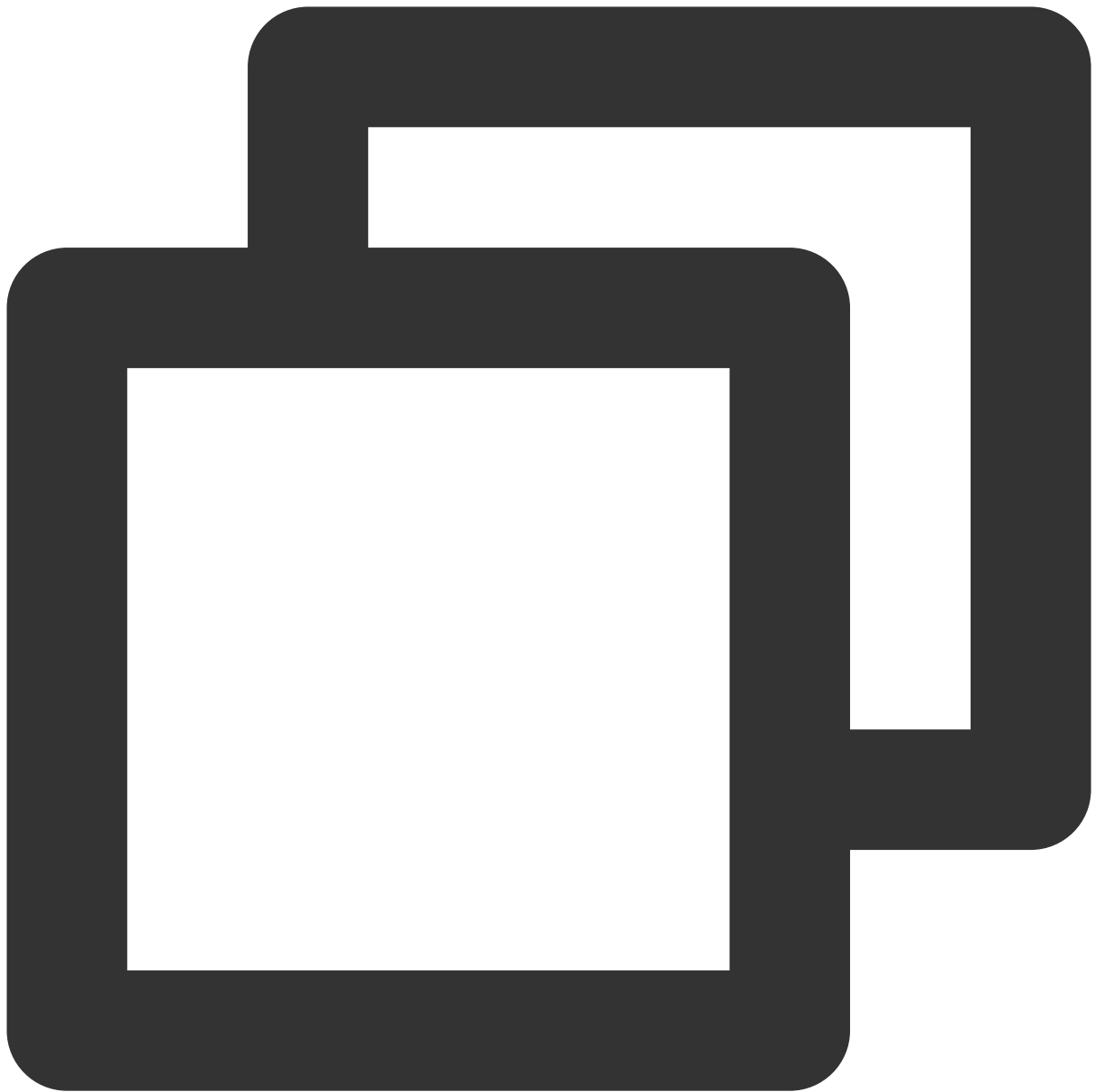
Parameter	Type	Meaning
sdAppId	NSInteger	SDKAppID of Tencent Cloud communication

		application
userId	NSString *	User ID for Differentiate different users
userSig	NSString *	UserSig for Tencent Cloud flow authentication
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## logout

Logout interface, there will be actively leave the room operation, destroy resources





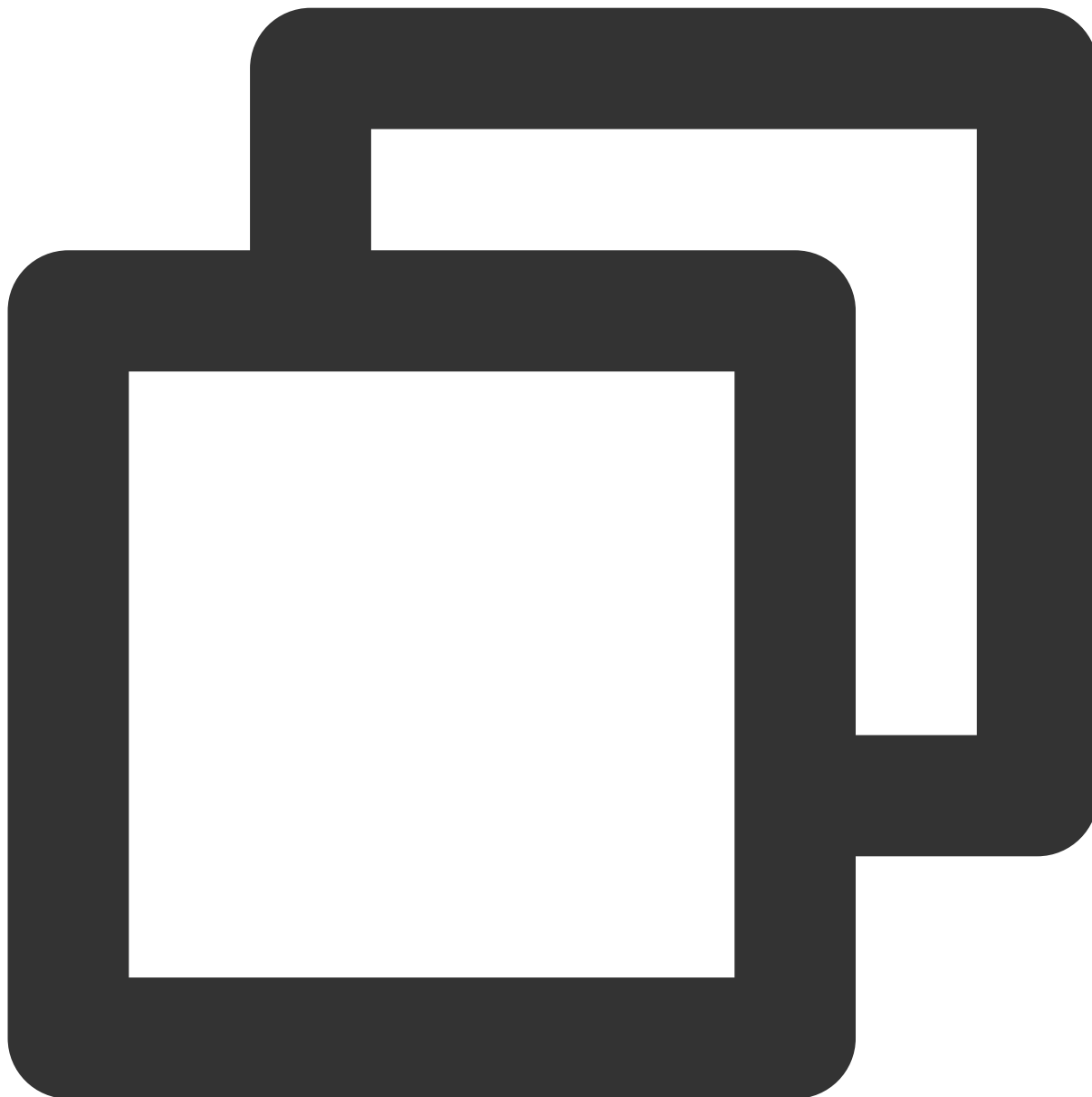
```
+ (void)logout:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

**The parameters are as follows:**

Parameter	Type	Meaning
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## setSelfInfo

Set local user name and avatar.



```
+ (void)setSelfInfoWithUserName:(NSString *)userName
      avatarUrl:(NSString *)avatarUrl
  onSuccess:(TUISuccessBlock) onSuccess
  onError:(TUIErrorBlock) onError;
```

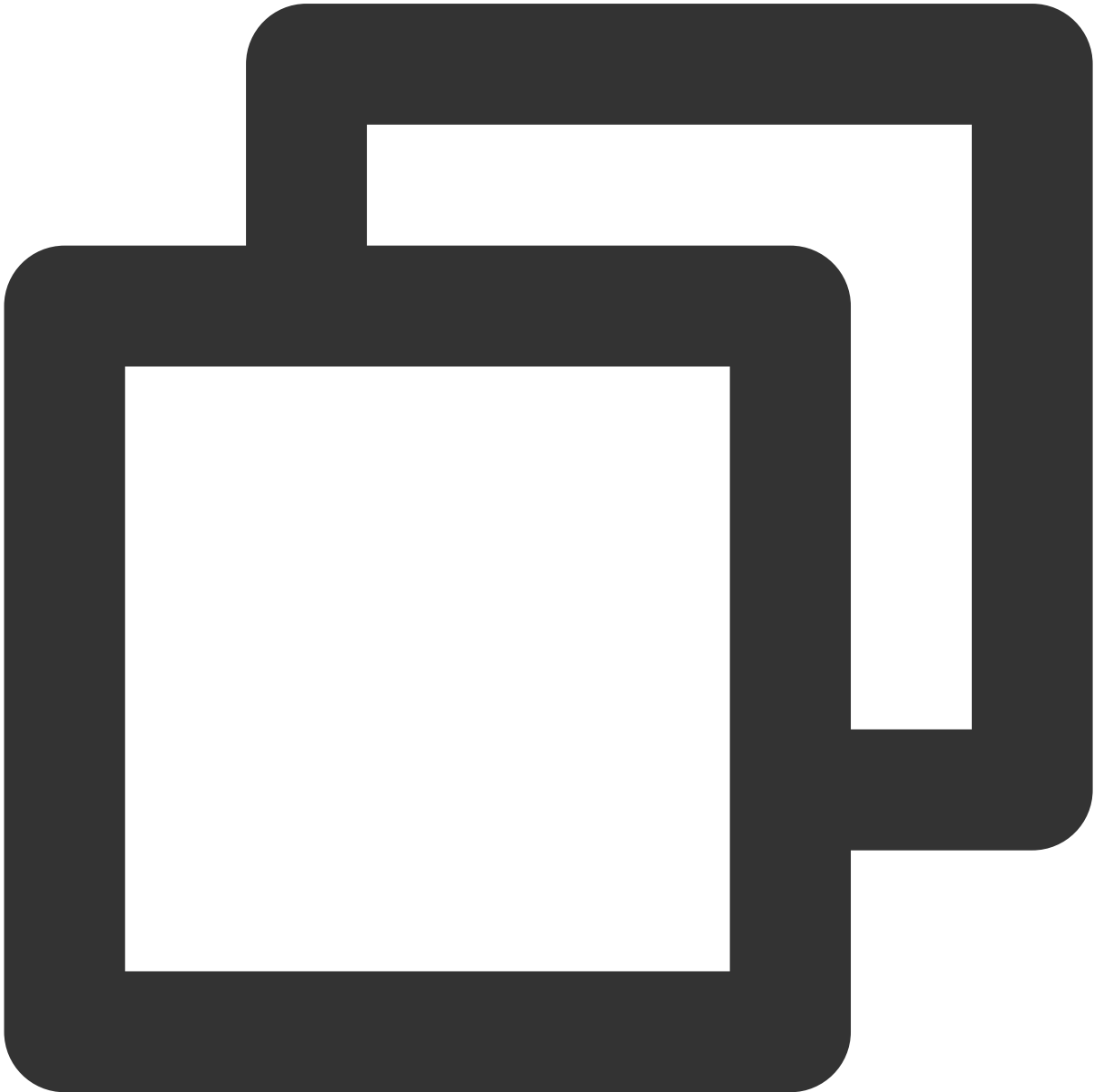
The parameters are as follows:

Parameter	Type	Meaning

userName	NSString *	User name
avatarUrl	NSString *	User avatar URL address
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

### getSelfInfo

Get the basic information of the local user login.





```
+ (TUILoginUserInfo *)getSelfInfo;
```

The parameters are as follows:

Parameter	Type	Meaning
selfInfo	<a href="#">TUILoginUserInfo</a> *	The basic information of the local user login, the detailed definition can be referred to <code>TUIRoomDefine.h</code> in <a href="#">TUILoginUserInfo</a> .

## addObserver

Set event callback.



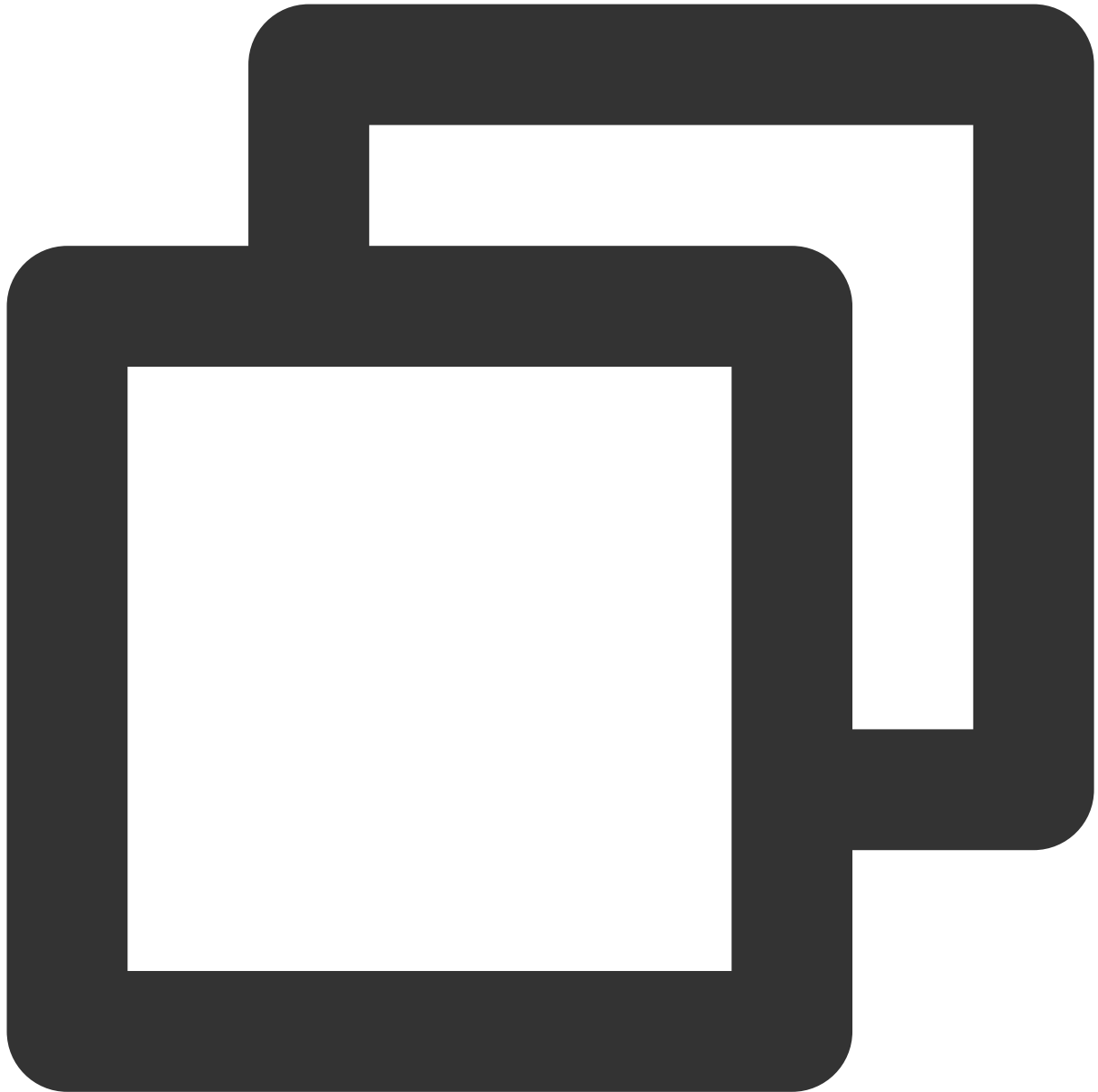
```
- (void)addObserver:(id<TUIRoomObserver>) observer;
```

**The parameters are as follows:**

Parameter	Type	Meaning
observer	TUIRoomObserver *	The pointer of the callback instance, you can get various event notifications (such as: error code, remote user entered room, audio and video status parameters, etc.) through TUIRoomObserver

## removeObserver

Remove event callback.



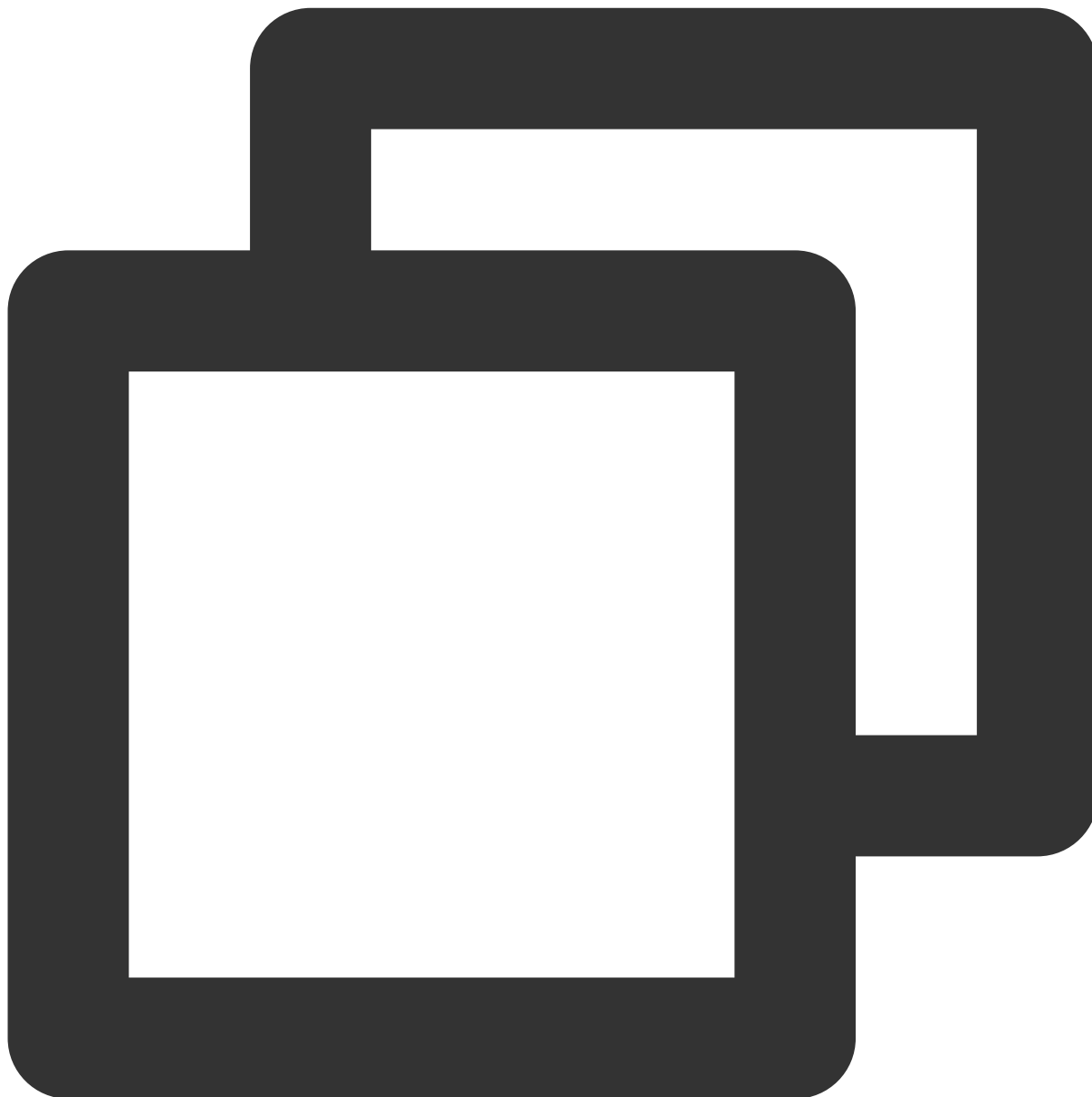
```
- (void)removeObserver:(id<TUIRoomObserver>)observer;
```

The parameters are as follows:

Parameter	Type	Meaning
observer	TUIRoomObserver *	The pointer of the callback instance

## createRoom

Create room.



```
- (void)createRoom:(TUIRoomInfo *)roomInfo
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

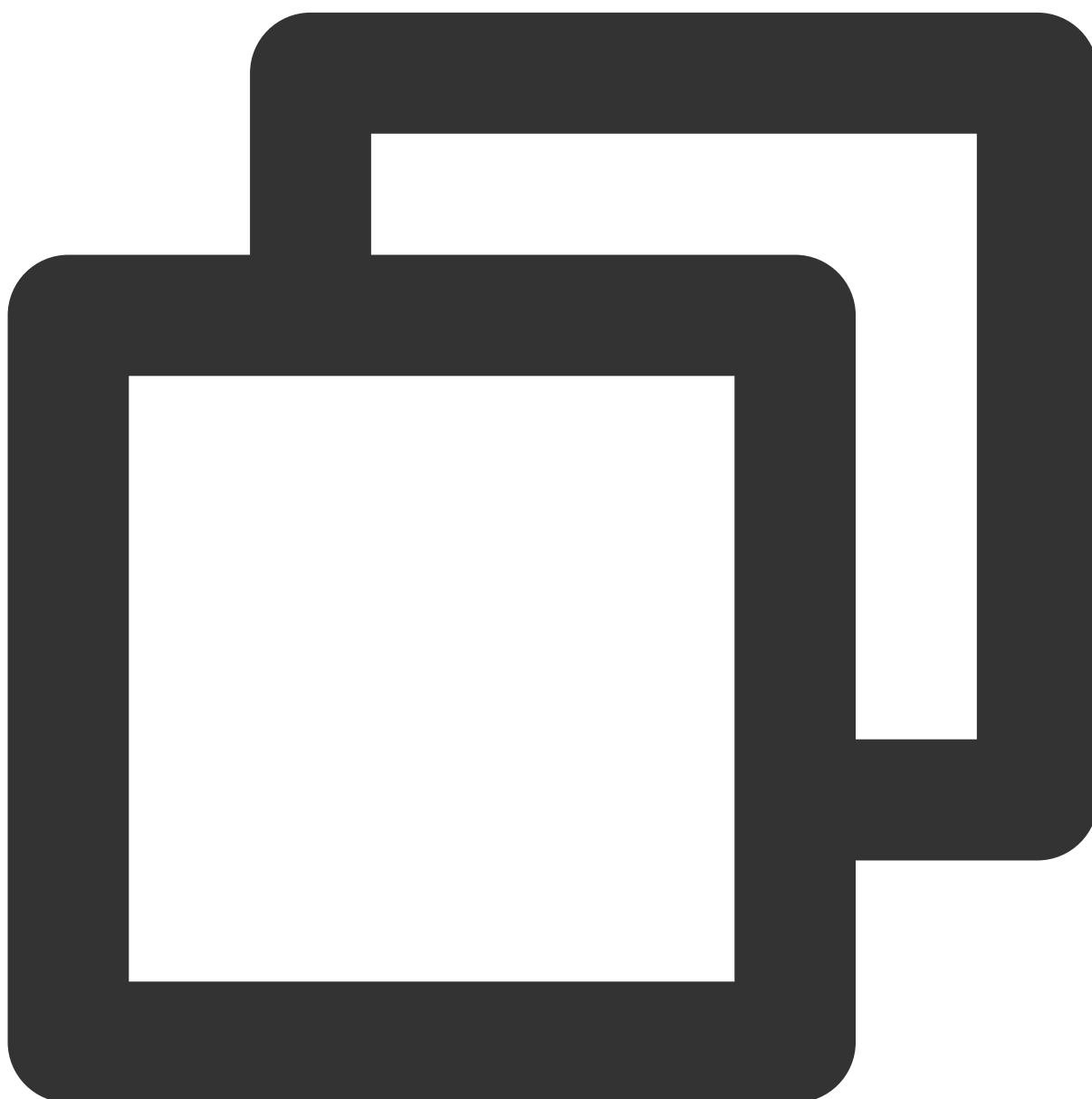
The parameters are as follows:

Parameter	Type	Meaning

roomInfo	<a href="#">TUIRoomInfo</a> *	Room data, you can initialize some room settings
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## destroyRoom

close the room.



```
- (void)destroyRoom:(TUISuccessBlock)onSuccess onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## enterRoom

Entered room.



```
- (void)enterRoom:(NSString *)roomId
    onSuccess:(TUIRoomInfoBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID
onSuccess	TUIRoomInfoBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## exitRoom

Leave room.



```
- (void)exitRoom:(BOOL)syncWaiting  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

**The parameters are as follows:**

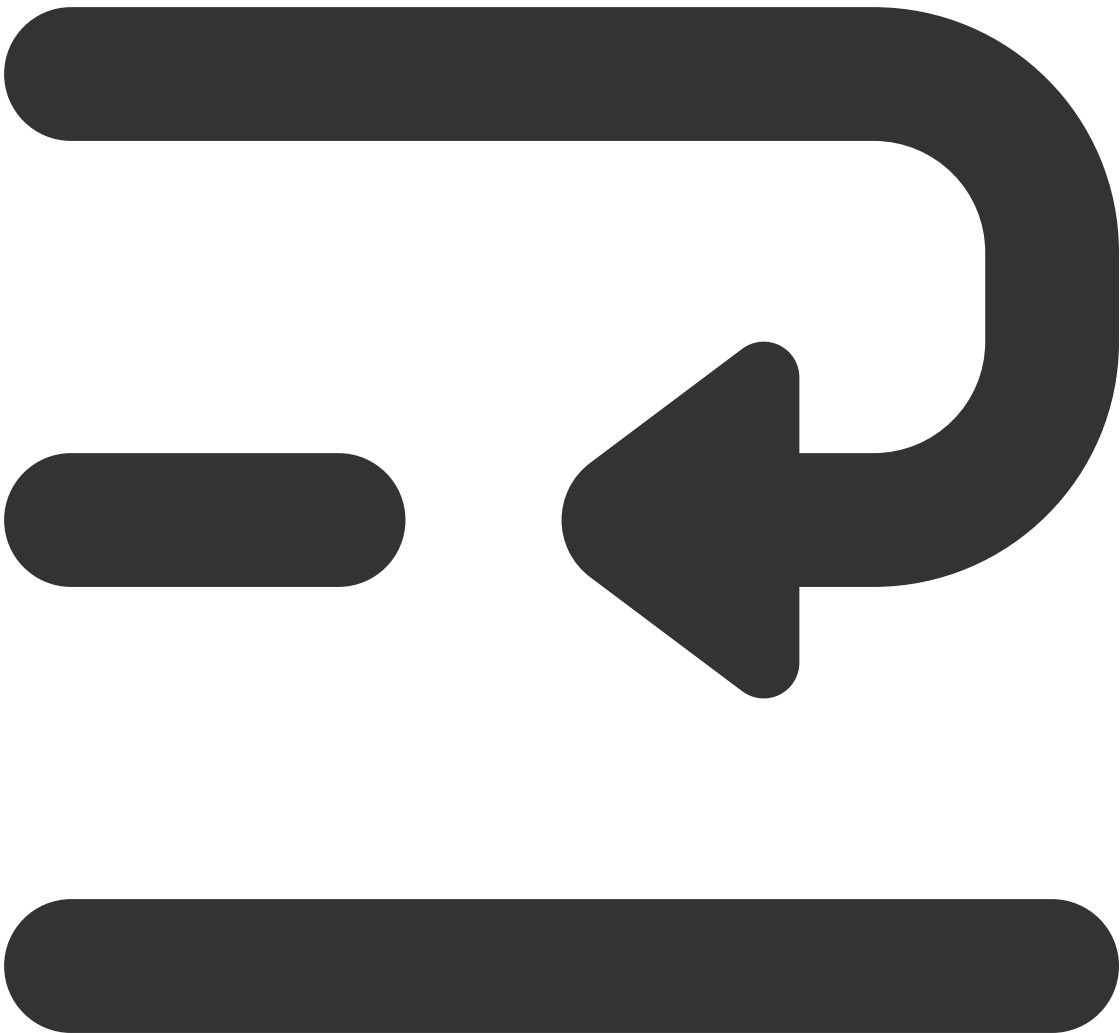
Parameter	Type	Meaning
syncWaiting	BOOL	Whether to synchronize and wait for the interface to return



onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

**connectOtherRoom**

Connect to other rooms (used for streaming scenario to apply for cross-room streaming).





```
- (TUIRequest *)connectOtherRoom:(NSString *)roomId
    userId:(NSString *)userId
    timeout:(NSTimeInterval)timeout
    onAccepted:(TUIRequestAcceptedBlock)onAccepted
    onRejected:(TUIRequestRejectedBlock)onRejected
    onCancelled:(TUIRequestCancelledBlock)onCancelled
    onTimeout:(TUIRequestTimeoutBlock)onTimeout
    onError:(TUIRequestErrorBlock)onError;
```

The parameters are as follows:

--	--	--

Parameter	Type	Meaning
roomId	NSString *	Room ID
userId	NSString *	User ID
timeout	NSTimeInterval	Timeout period (Unit seconds, if set to 0, SDK will not perform timeout detection and will not trigger timeout Callback)
onAccepted	TUIRequestAcceptedBlock	Invitation accepted Callback
onRejected	TUIRequestRejectedBlock	Invitation rejected Callback
onCancelled	TUIRequestCancelledBlock	Invitation canceled Callback
onTimeout	TUIRequestTimeoutBlock	Invitation timeout unprocessed Callback
onError	TUIRequestErrorBlock	Invitation error occurred Callback

Return value	Type	Meaning
request	TUIRequest	Request body

## disconnectOtherRoom

Disconnect from other rooms (used for disconnecting cross-room streaming in live streaming scenarios).



```
- (void)disconnectOtherRoom:(TUISuccessBlock)onSuccess onError:(TUIErrorBlock)onErr
```

The parameters are as follows:

Parameter	Type	Meaning
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## fetchRoomInfo

Get Room data.



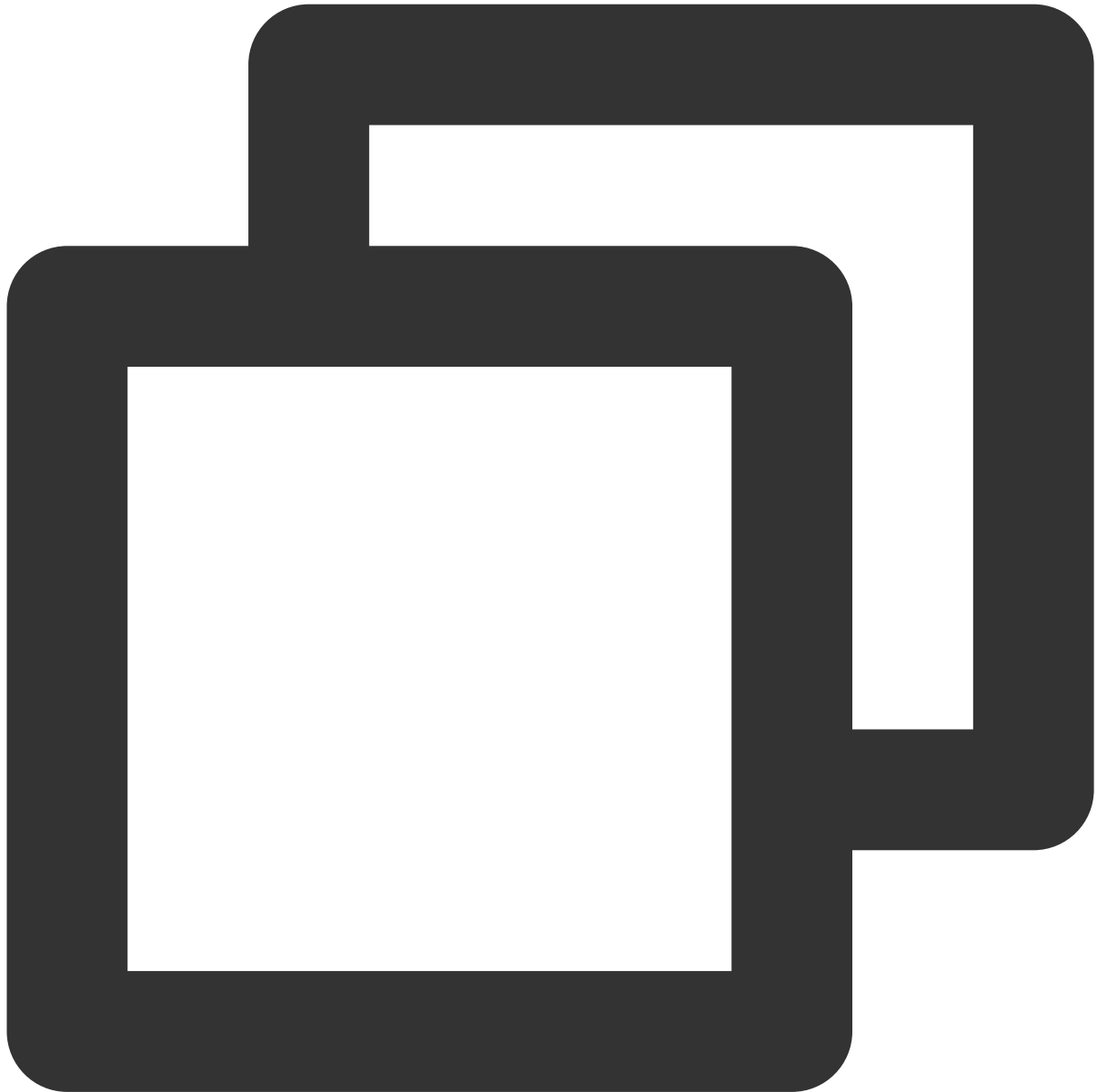
```
- (void)fetchRoomInfo:(TUIRoomInfoBlock)onSuccess onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
onSuccess	TUIRoomInfoBlock	Success Callback with <a href="#">TUIRoomInfo</a> Room data
onError	TUIErrorBlock	Failed callback

## updateRoomNameByAdmin

Update Room ID (Only Administrator or Group owner can call).



```
- (void)updateRoomNameByAdmin:(NSString *)roomName
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

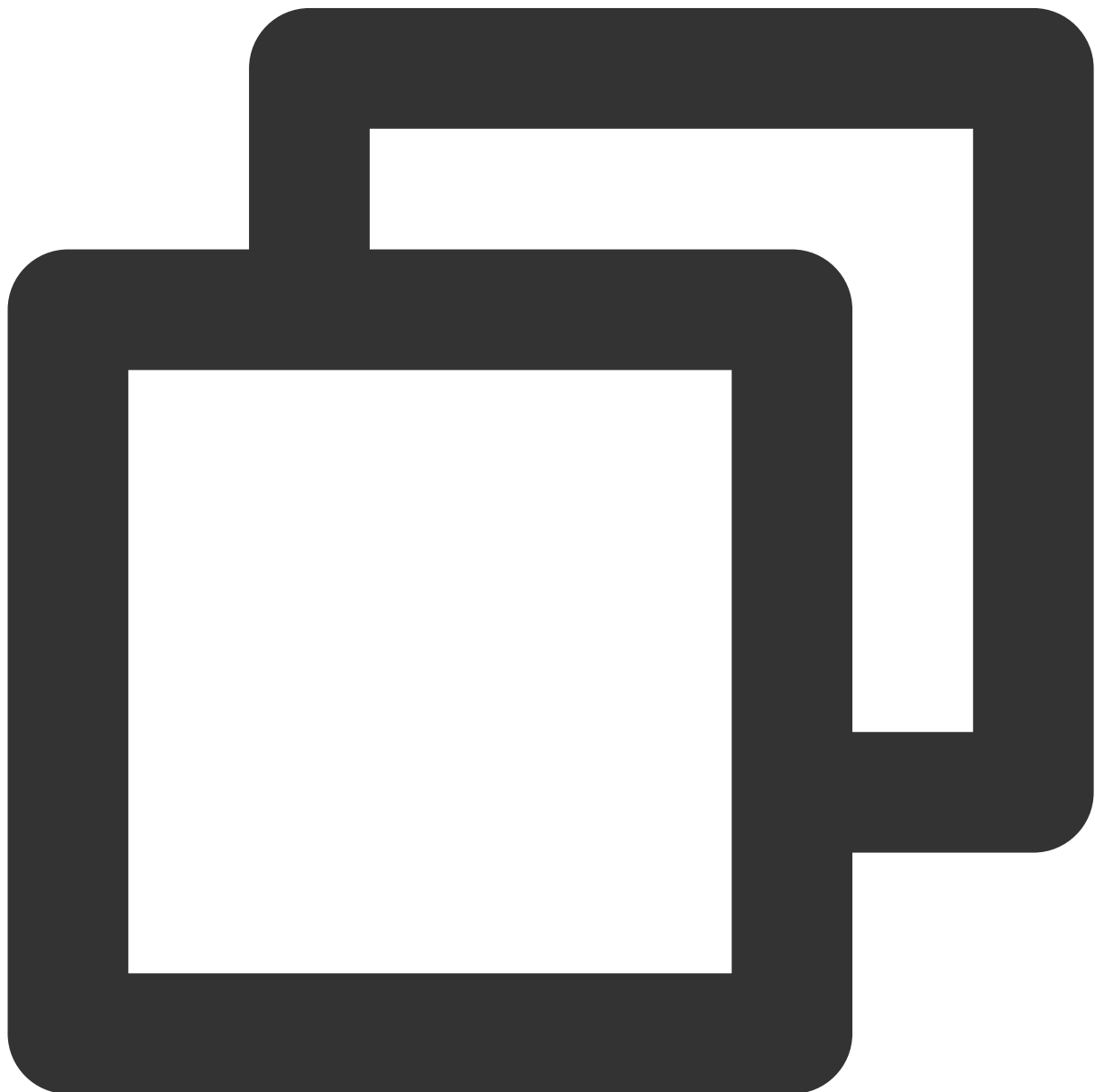
The parameters are as follows:

Parameter	Type	Meaning

roomName	NSString *	Room Name
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## updateRoomSpeechModeByAdmin

Set up the room's mic control mode (only administrators or group owners can call).



```
- (void)updateRoomSpeechModeByAdmin:(TUISpeechMode)mode
```

```
onSuccess: (TUISuccessBlock) onSuccess  
onError: (TUIErrorBlock) onError;
```

The parameters are as follows:

Parameter	Type	Meaning
mode	<a href="#">TUISpeechMode</a>	Speech Mode, detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUISpeechMode</a> .
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## setLocalVideoView

Set Local User Video Rendering Control.





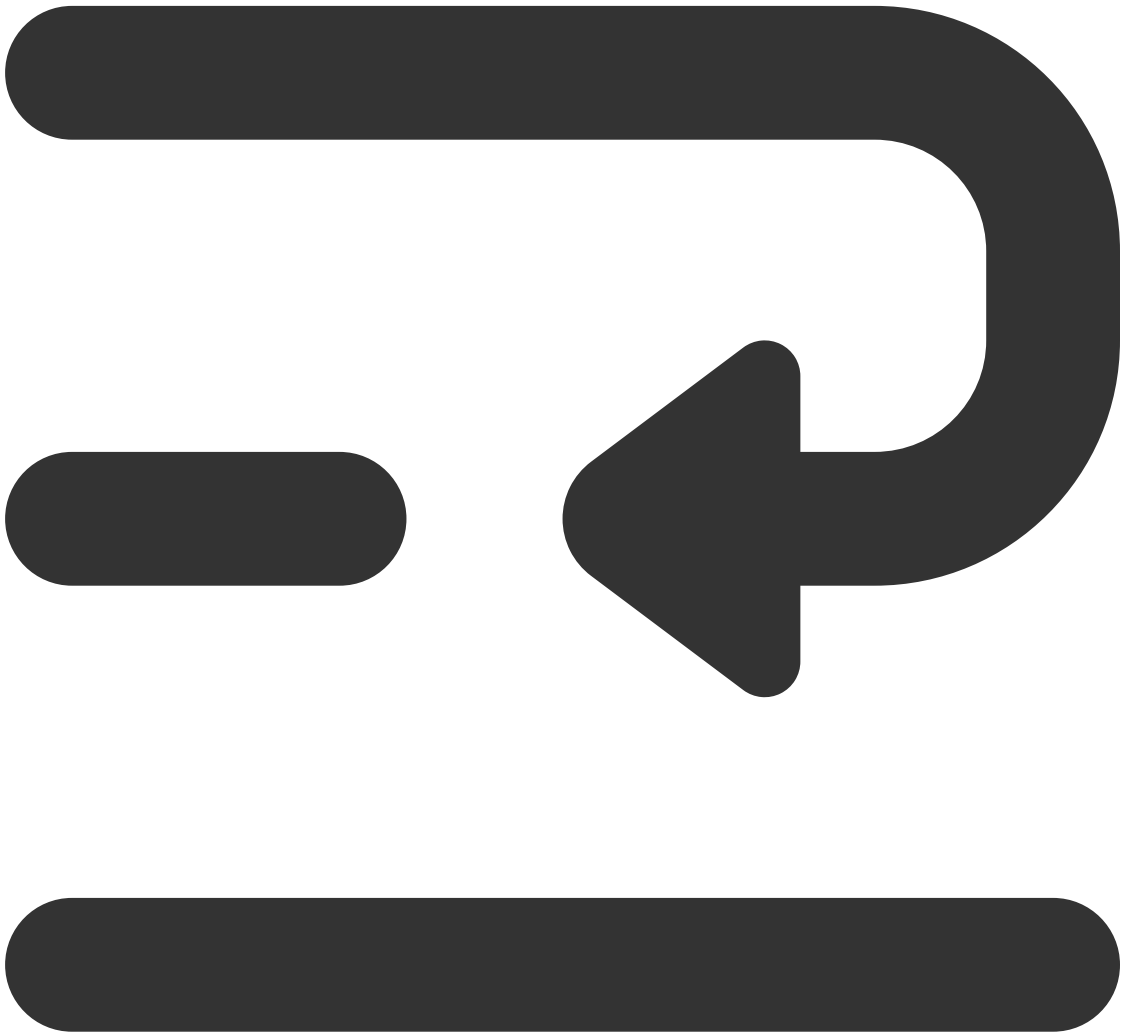
```
- (void)setLocalVideoView:(TUIVideoStreamType)streamType view:(TUIVideoView *)view;
```

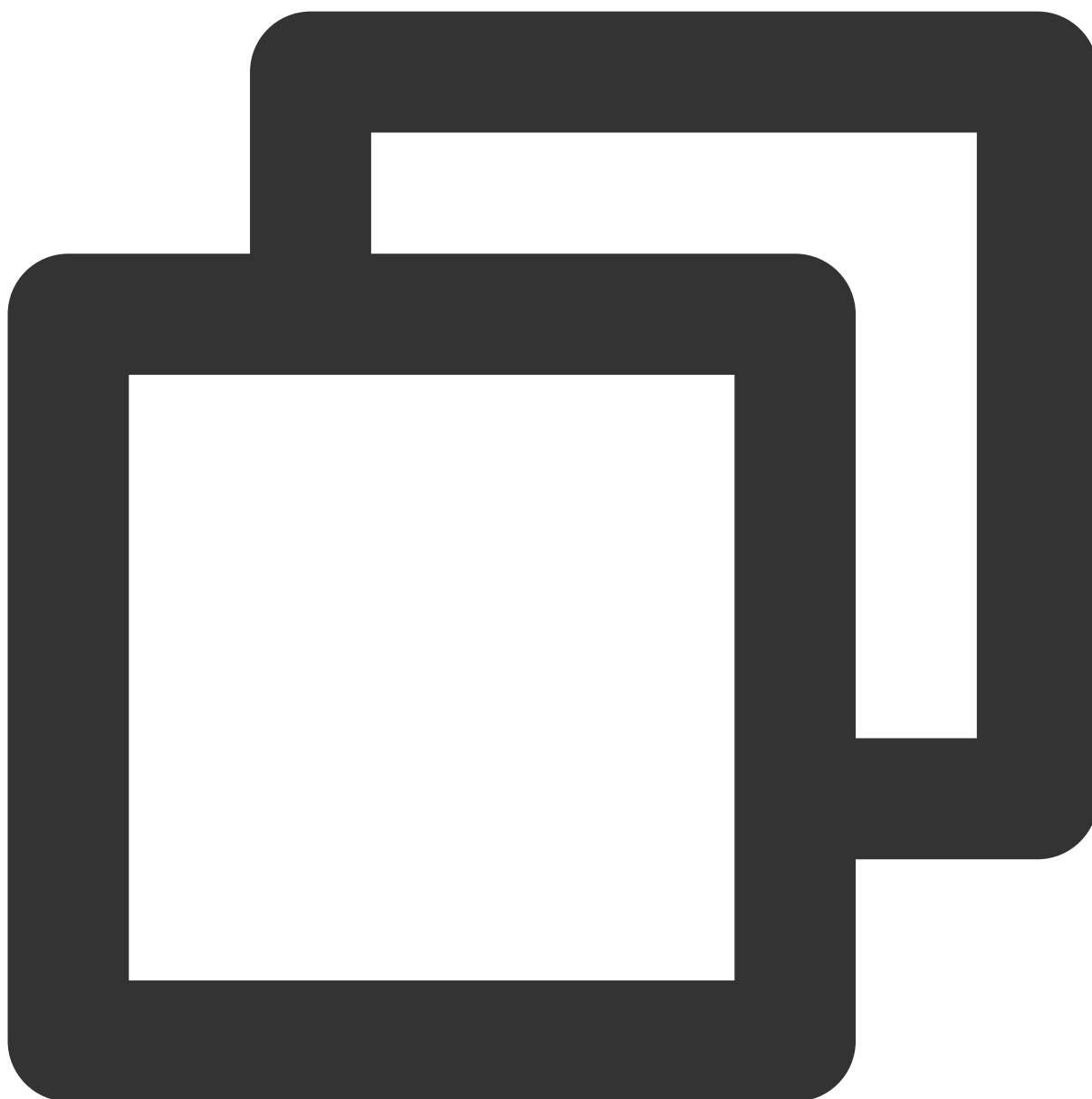
The parameters are as follows:

Parameter	Type	Meaning
streamType	<a href="#">TUIVideoStreamType</a>	Video streams Type, detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUIVideoStreamType</a> .
view	TUIVideoView *	Video Rendering View

## setRemoteVideoView

Set Remote User Video Rendering Control.





```
- (void)setRemoteVideoView:(NSString *)userId
    streamType:(TUIVideoStreamType) streamType
    view:(TUIVideoView * __nullable) view;
```

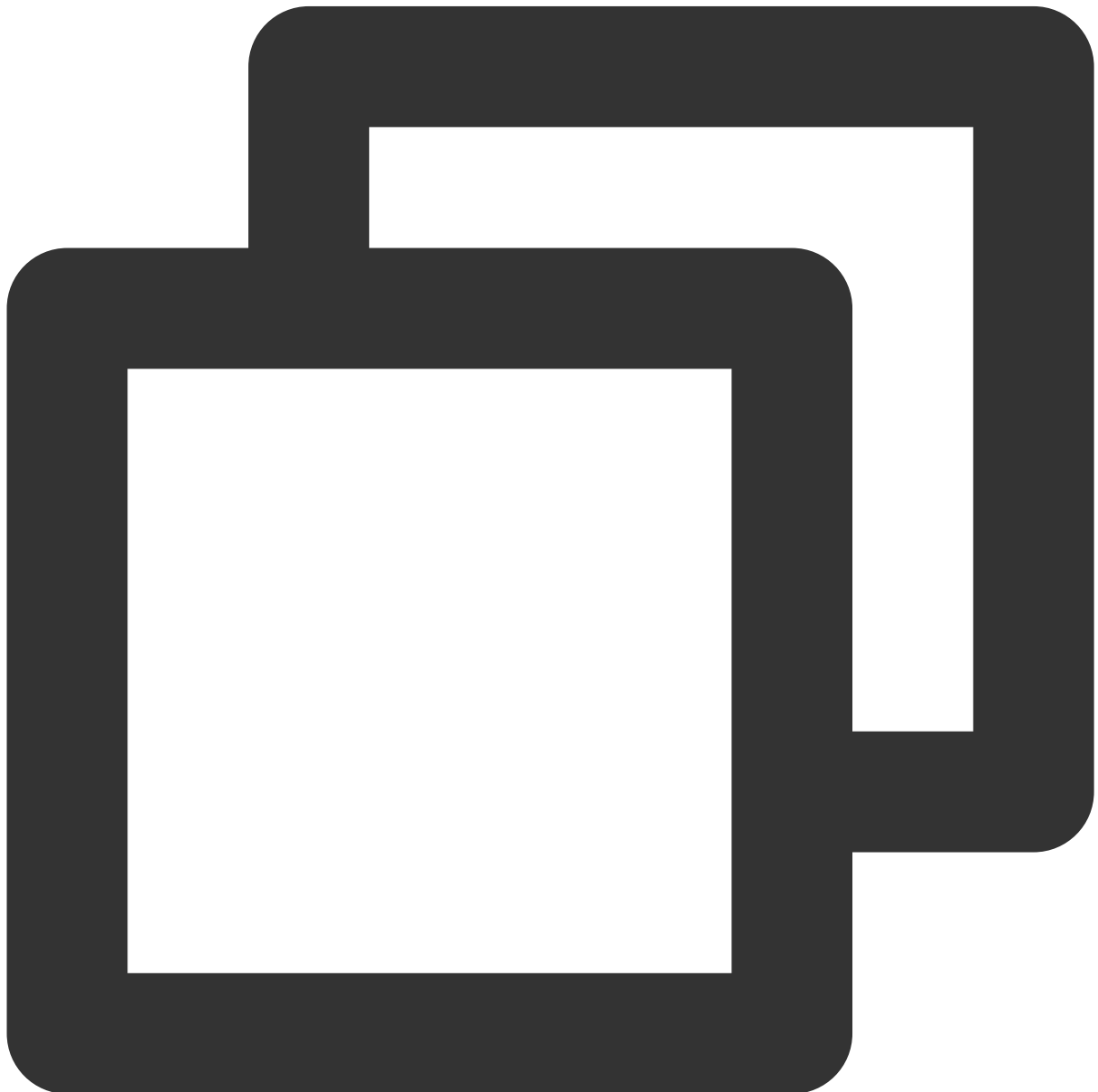
The parameters are as follows:

Parameter	Type	Meaning
userId	NSString	Remote User ID
streamType	<a href="#">TUIVideoStreamType</a>	Video streams Type, detailed Definition can be found in

		TUIRoomDefine.h file's TUIVideoStreamType.
view	TUIVideoView *	Video Rendering View

## openLocalCamera

Open Local Camera.



```
- (void)openLocalCamera:(BOOL)isFront  
    quality:(TUIVideoQuality)quality
```

```
onSuccess:(TUISuccessBlock) onSuccess  
onError:(TUIErrorBlock) onError;
```

Parameter	Type	Meaning
isFront	BOOL	Front or not
quality	<a href="#">TUIVideoQuality</a>	Video Quality, detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUIVideoQuality</a> .
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## closeLocalCamera

Close Local Camera.



```
- (void)closeLocalCamera;
```

### **updateVideoQuality**

Update Local Video Codec Quality Setting.

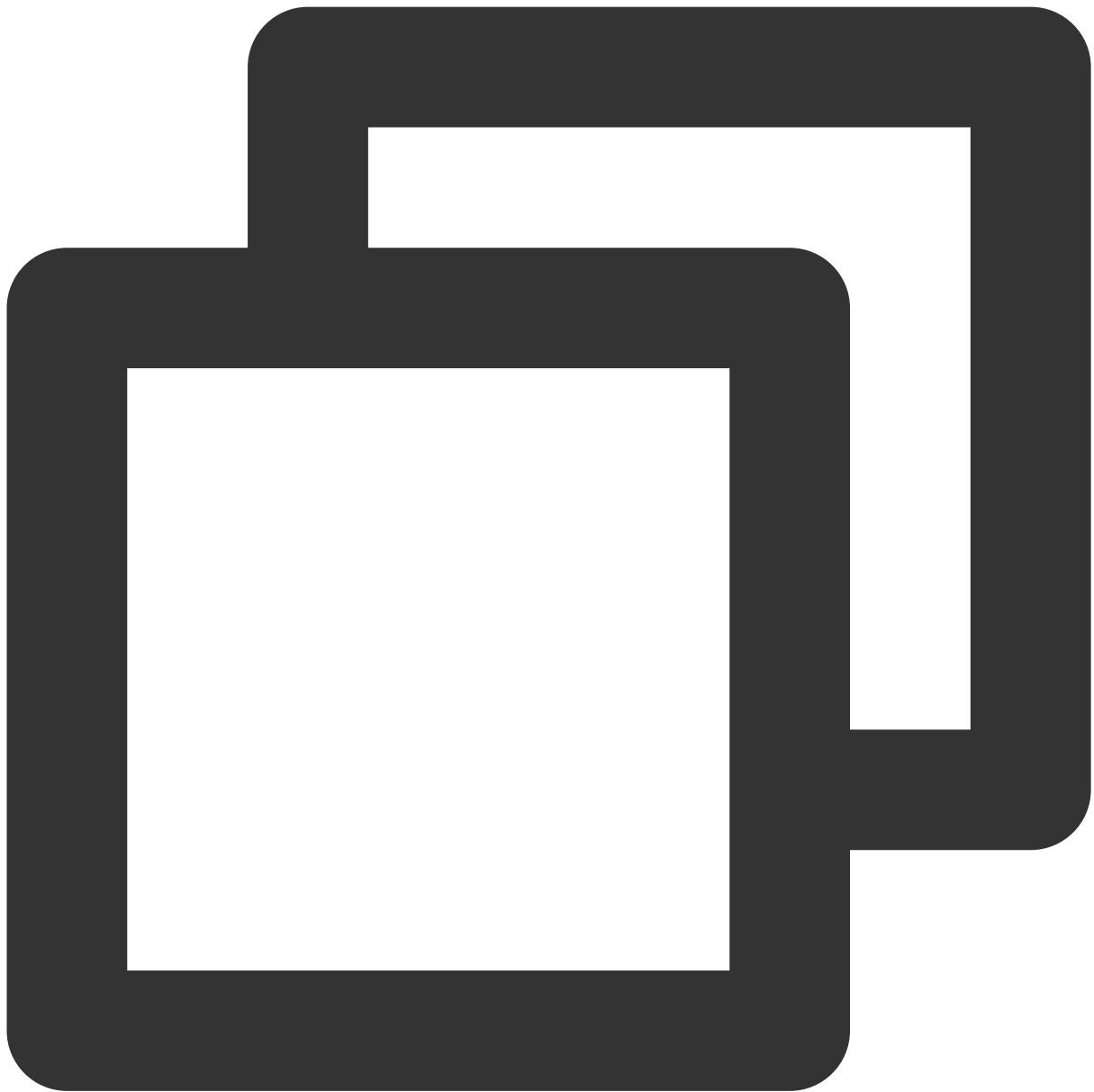


```
- (void)updateVideoQuality:(TUIVideoQuality)quality;
```

Parameter	Type	Meaning
quality	<a href="#">TUIVideoQuality</a>	Video Quality

## openLocalMicrophone

Open Local mic.



```
- (void)openLocalMicrophone:(TUIAudioQuality) quality  
    onSuccess:(TUISuccessBlock) onSuccess  
    onError:(TUIErrorBlock) onError;
```

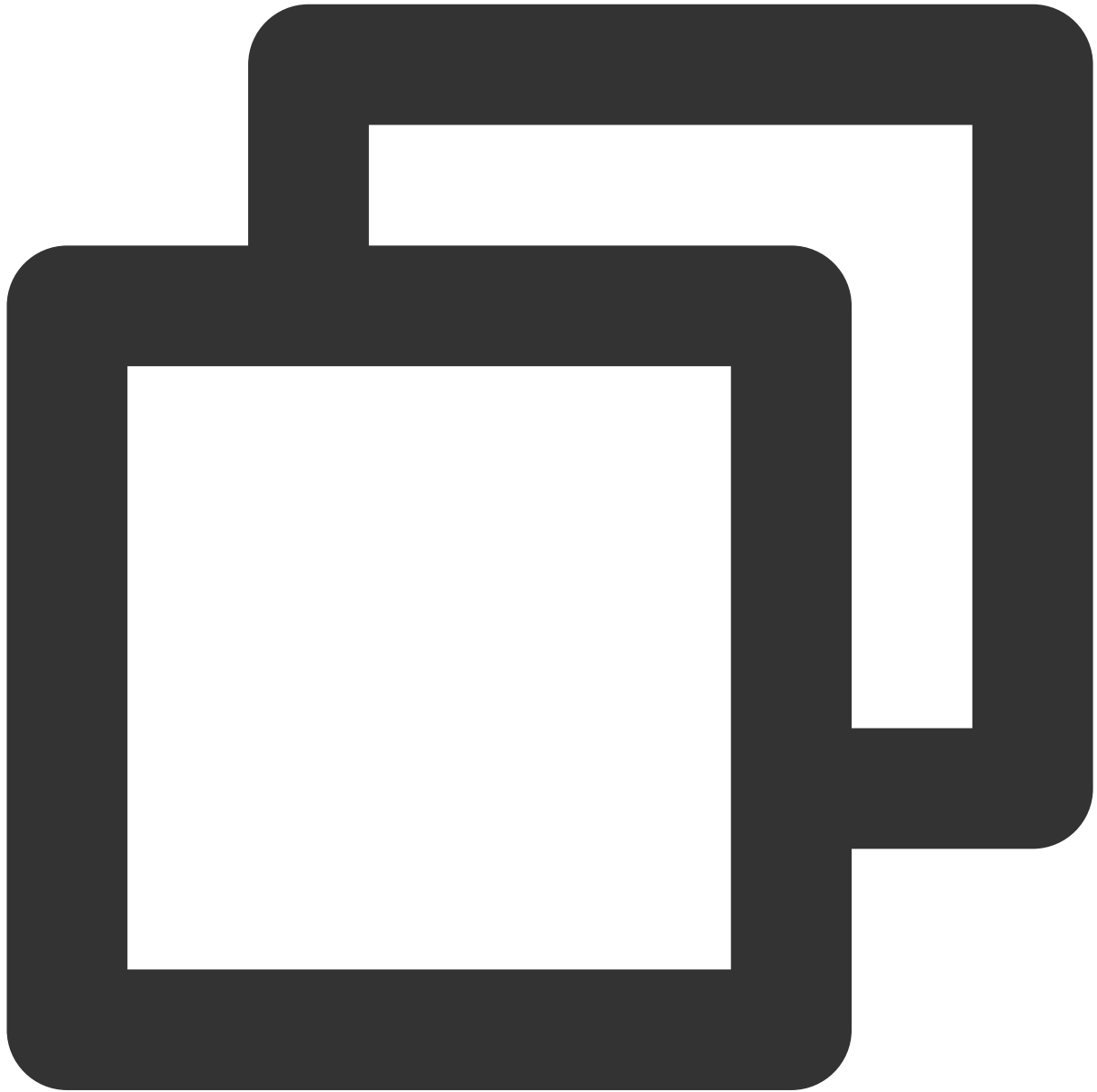
Parameter	Type	Meaning
quality	<a href="#">TUIAudioQuality</a>	Audio Quality, detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUIAudioQuality</a>
onSuccess	TUISuccessBlock	Successful callback



onError	TUIErrorBlock	Failed callback
---------	---------------	-----------------

## closeLocalMicrophone

Close Local mic.



```
- (void)closeLocalMicrophone;
```

## updateAudioQuality

Update Local Audio Codec Quality Setting.



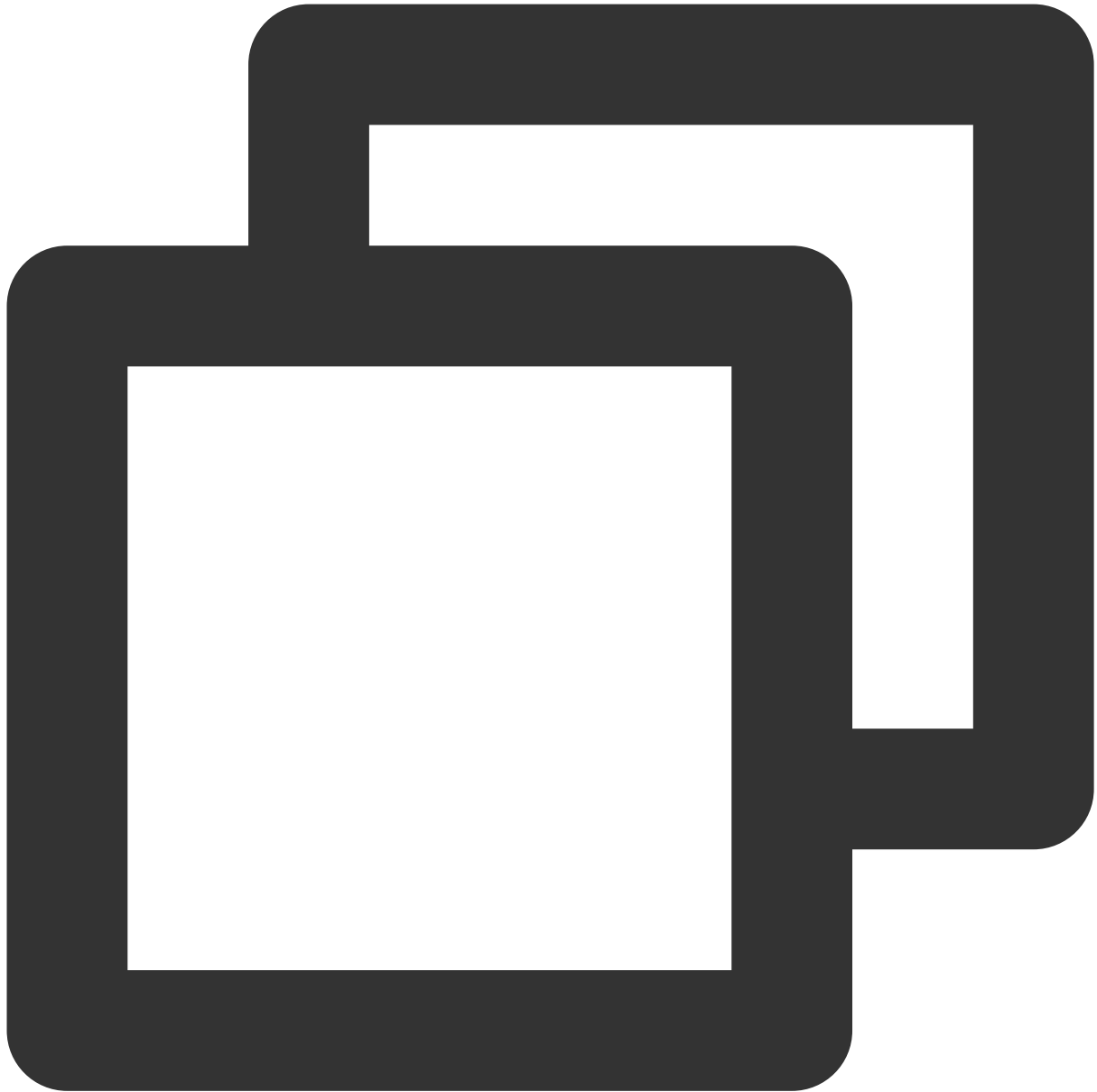
```
- (void)updateAudioQuality:(TUIAudioQuality)quality;
```

Parameter	Type	Meaning
quality	<a href="#">TUIAudioQuality</a>	Audio Quality

## startScreenCapture

Start Screen Sharing (Only supports mobile iOS 11.0 and above systems)

This interface supports capturing the entire iOS system screen, similar to Tencent Meeting's full system-level Screen Sharing.



```
- (void)startScreenCaptureByReplaykit:(NSString *)appGroup API_AVAILABLE(ios(11.0))
```

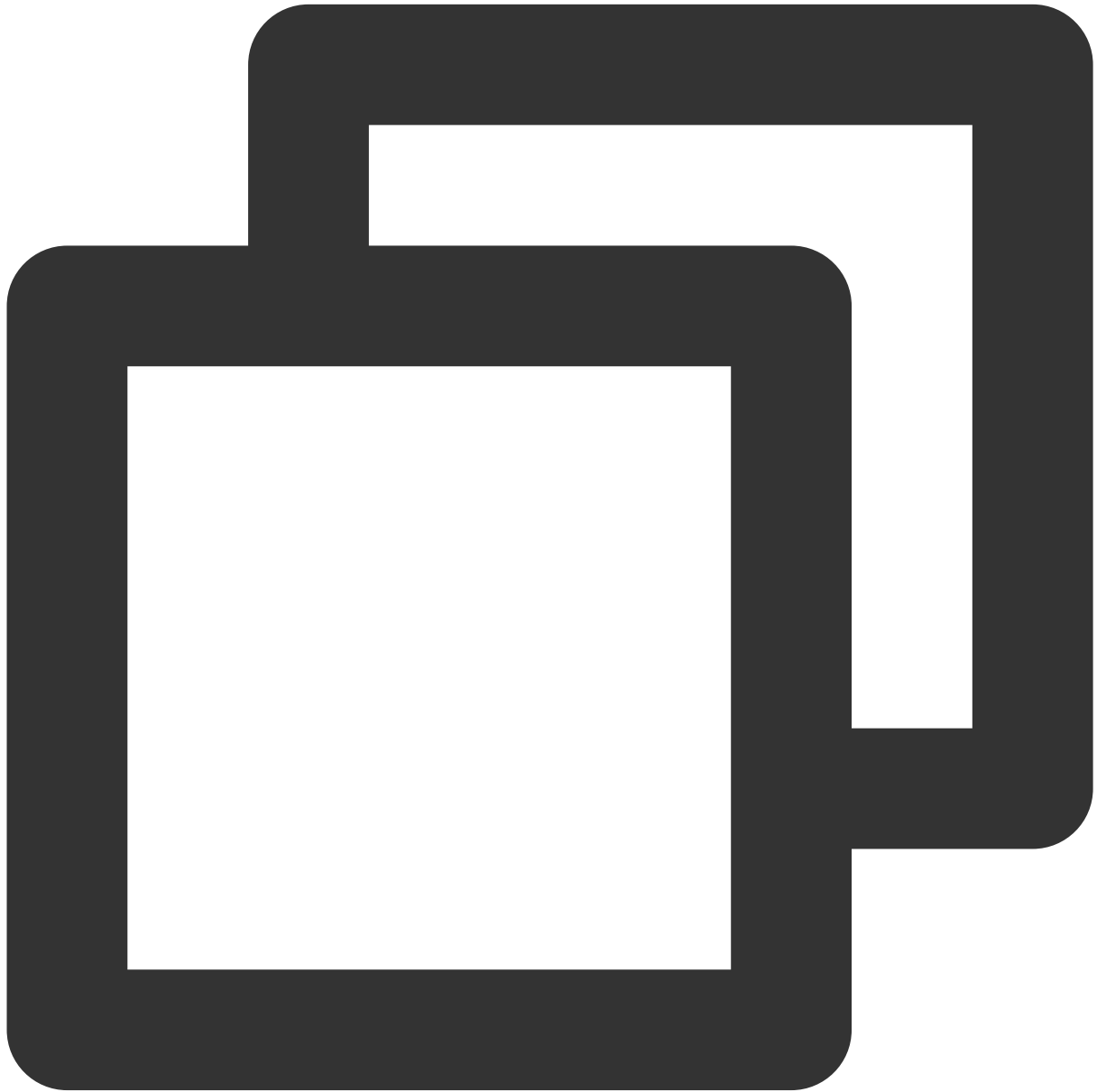
**The parameters are as follows:**

Parameter	Type	Meaning
appGroup	NSString *	Specify the Application Group Identifier shared by your app and the Screen recording Process, you can set this parameter to nil, but it is recommended to set it according to the documentation for better Reliability.

## startScreenCapture

Start Screen Sharing (This interface only supports desktop Mac OS systems)

This interface can capture the entire Mac OS system screen or the window content of a specified app and share it with other users in the same room.



```
- (void)startScreenCapture:(TUIVideoView *)view  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
view	TUIVideoView *	Parent Control of the Rendering Control, can be set to null, indicating no preview effect of Screen Sharing.
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## stopScreenCapture

End Screen Sharing.



```
- (void)stopScreenCapture;
```

## getScreenCaptureSources

Enumerate available screens and windows for sharing (This interface only supports Mac OS systems)

When you integrate the desktop system's Screen Sharing function, you generally need to display a Target selection Interface so that users can use this Interface to choose whether to share the entire screen or a specific window.

Through this interface, you can query the ID, name, and thumbnail of the windows available for sharing in the current system. We provide a default Interface implementation in the Demo for your reference.



```
- (NSArray<TUIShareTarget *> *)getScreenCaptureSources;
```

Return Value	Type	Meaning
screenCaptureSources	NSArray< <a href="#">TUIShareTarget</a> *> *	Window List including screens

### **selectScreenCaptureTarget**

Select Screen Sharing Target.



```
- (void)selectScreenCaptureTarget:(NSString *)targetId;
```

The parameters are as follows:

Parameter	Type	Meaning
targetId	NSString *	Designated Sharing Source

## startPushLocalVideo

Start Pushing Local Video.





```
- (void)startPushLocalVideo;
```

### **stopPushLocalVideo**

Stop Pushing Local Video.



```
- (void)stopPushLocalVideo;
```

### **startPushLocalAudio**

Start Pushing Local Audio.



```
- (void)startPushLocalAudio;
```

### **stopPushLocalAudio**

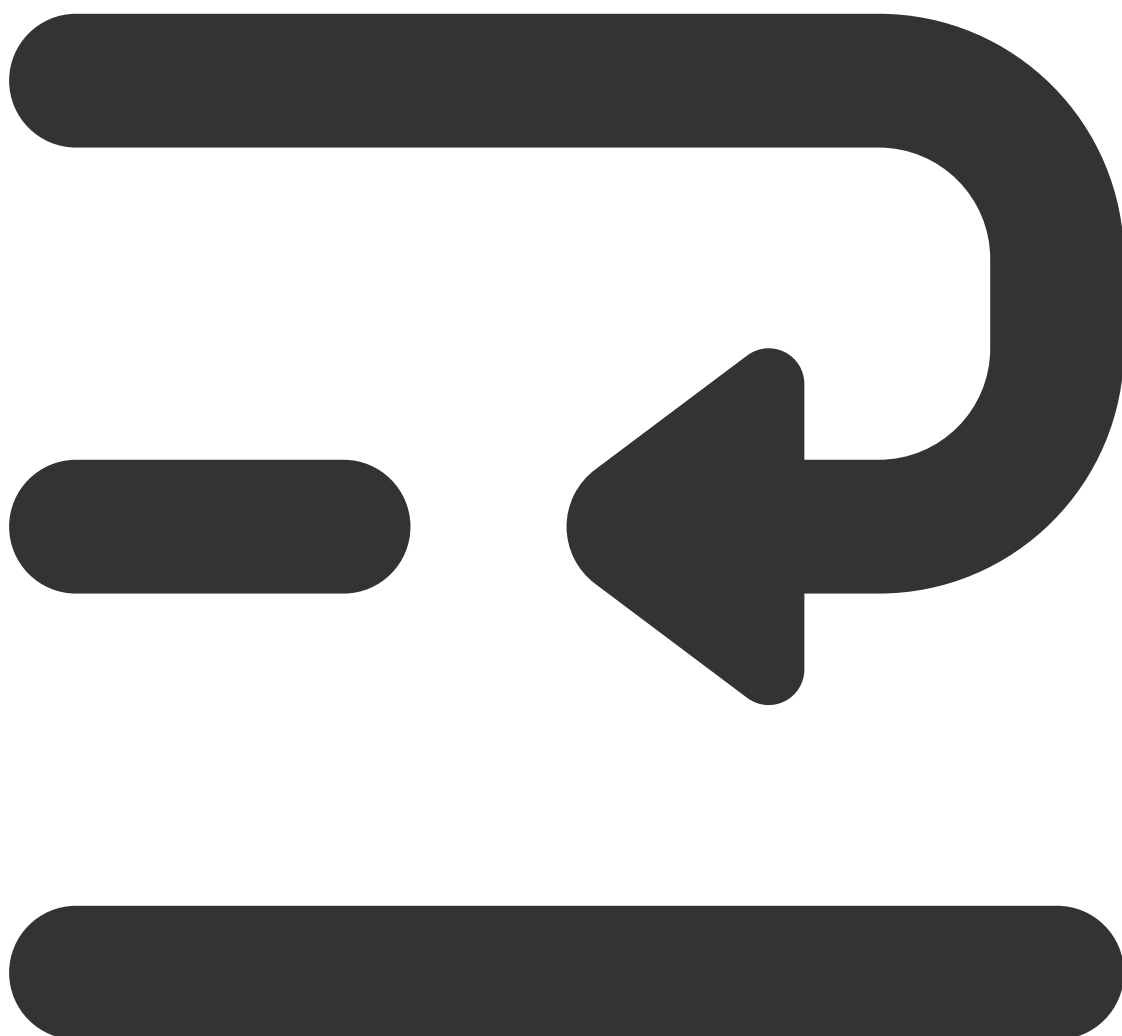
Stop Pushing Local Audio.

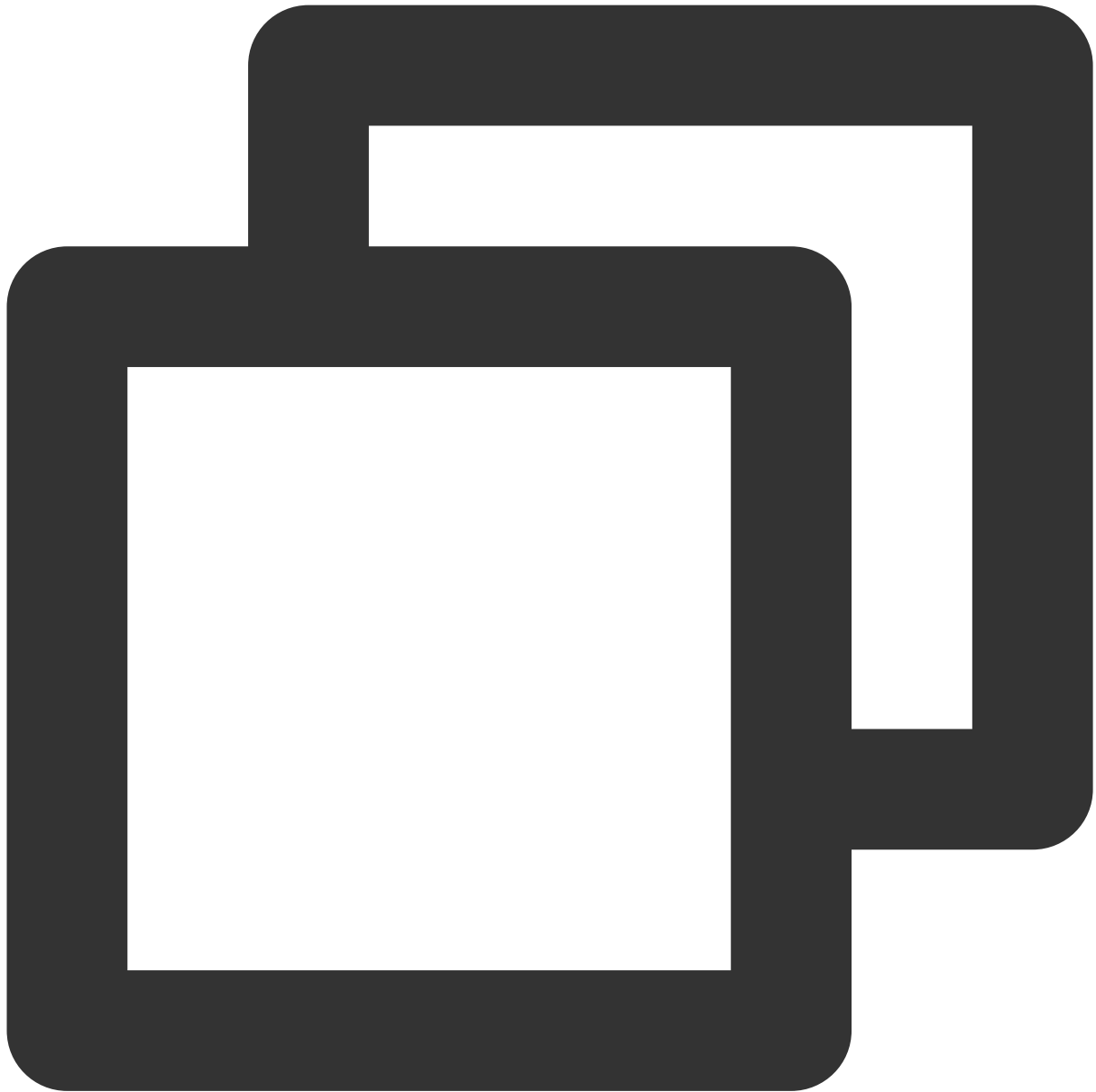


```
- (void)stopPushLocalAudio;
```

### **startPlayRemoteVideo**

Start Playback Remote User Video.





```
- (void)startPlayRemoteVideo:(NSString *)userId
    streamType:(TUIVideoStreamType)streamType
    onPlaying:(TUIPlayOnPlayingBlock)onPlaying
    onLoading:(TUIPlayOnLoadingBlock)onLoading
    onError:(TUIPlayOnErrorBlock)onError;
```

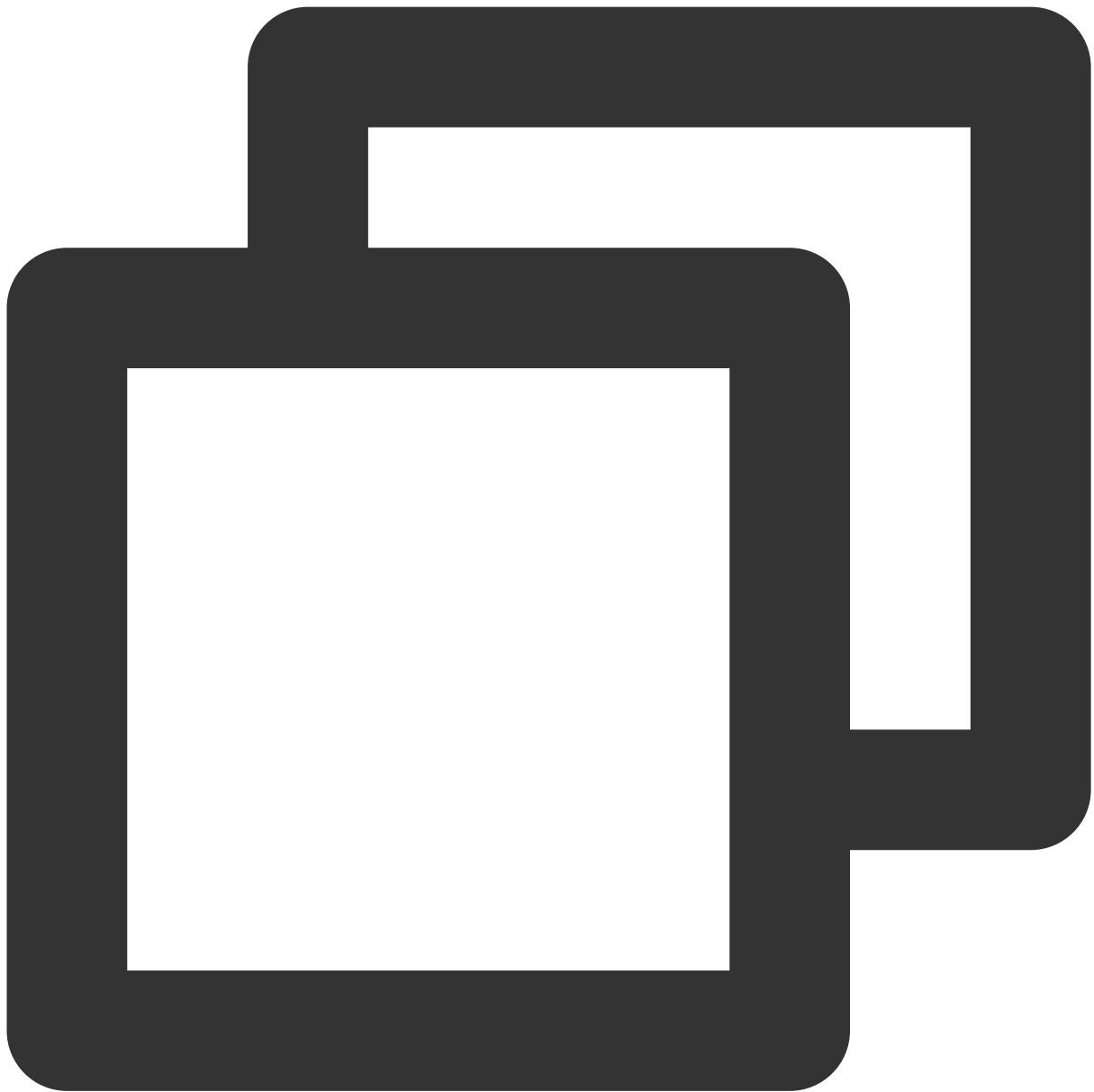
The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID

streamType	<a href="#">TUIVideoStreamType</a>	Type of Video streams. Detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUIVideoStreamType</a> .
onPlaying	TUIPlayOnPlayingBlock	Playback Callback
onLoading	TUIPlayOnLoadingBlock	Load Callback
onError	TUIPlayOnErrorBlock	Error Callback

## stopPlayRemoteVideo

Stop Playback Remote User Video.



```
- (void)stopPlayRemoteVideo:(NSString *)userId streamType:(TUIVideoStreamType)streamType
```

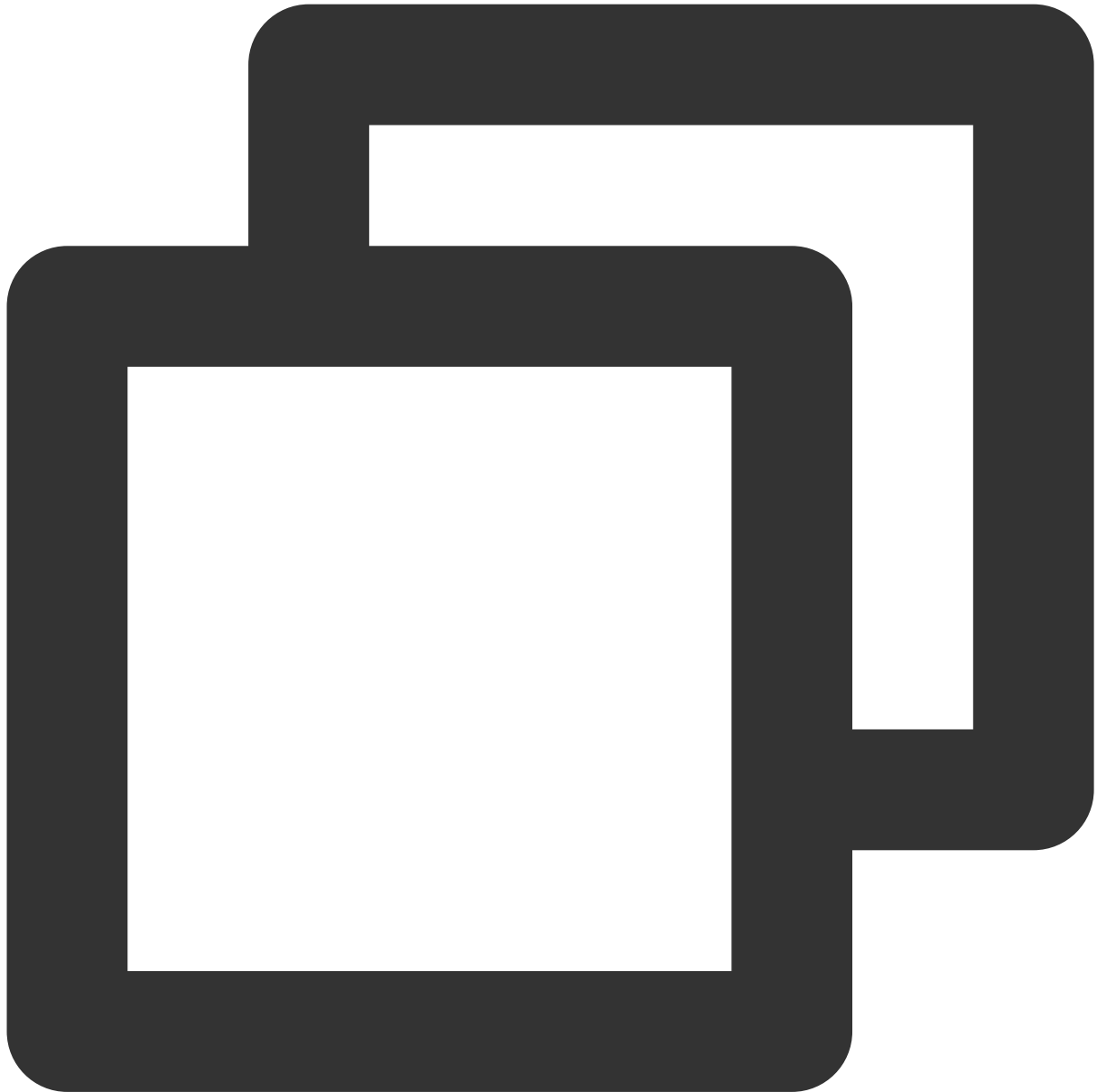
The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Type of Video streams. Detailed Definition can be found in TUIRoomDefine.h file's <a href="#">TUIVideoStreamType</a> .



## **muteRemoteAudioStream**

Mute Remote User

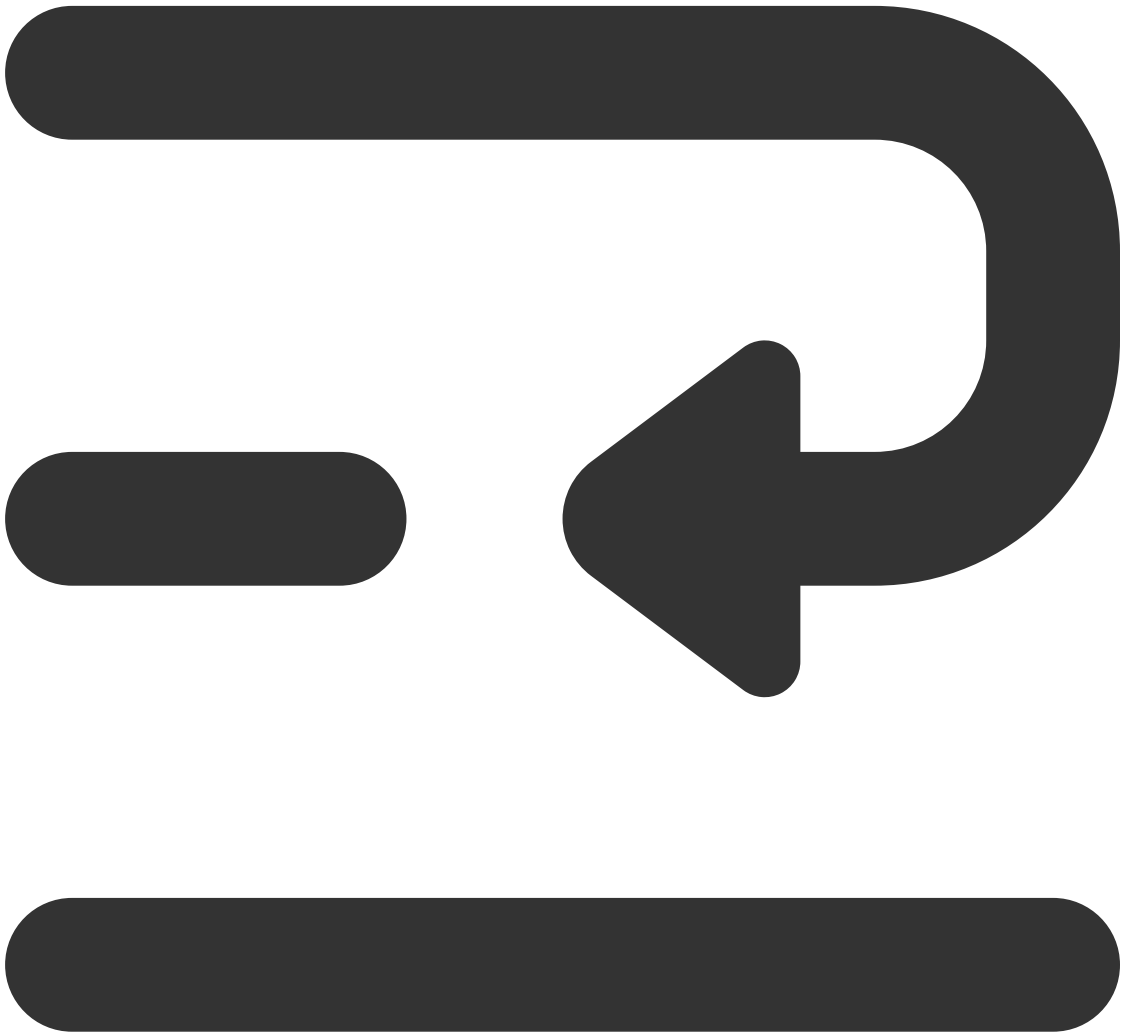


```
- (void)muteRemoteAudioStream:(NSString *)userId isMute:(BOOL)isMute;
```

Parameter	Type	Meaning
userId	NSString *	User ID
isMute	BOOL	Mute or not

## getUserList

Get Member List in the Room.





```
- (void)getUserList:(NSInteger)nextSequence
    onSuccess:(TUIUserListResponseBlock) onSuccess
    onError:(TUIErrorBlock) onError;
```

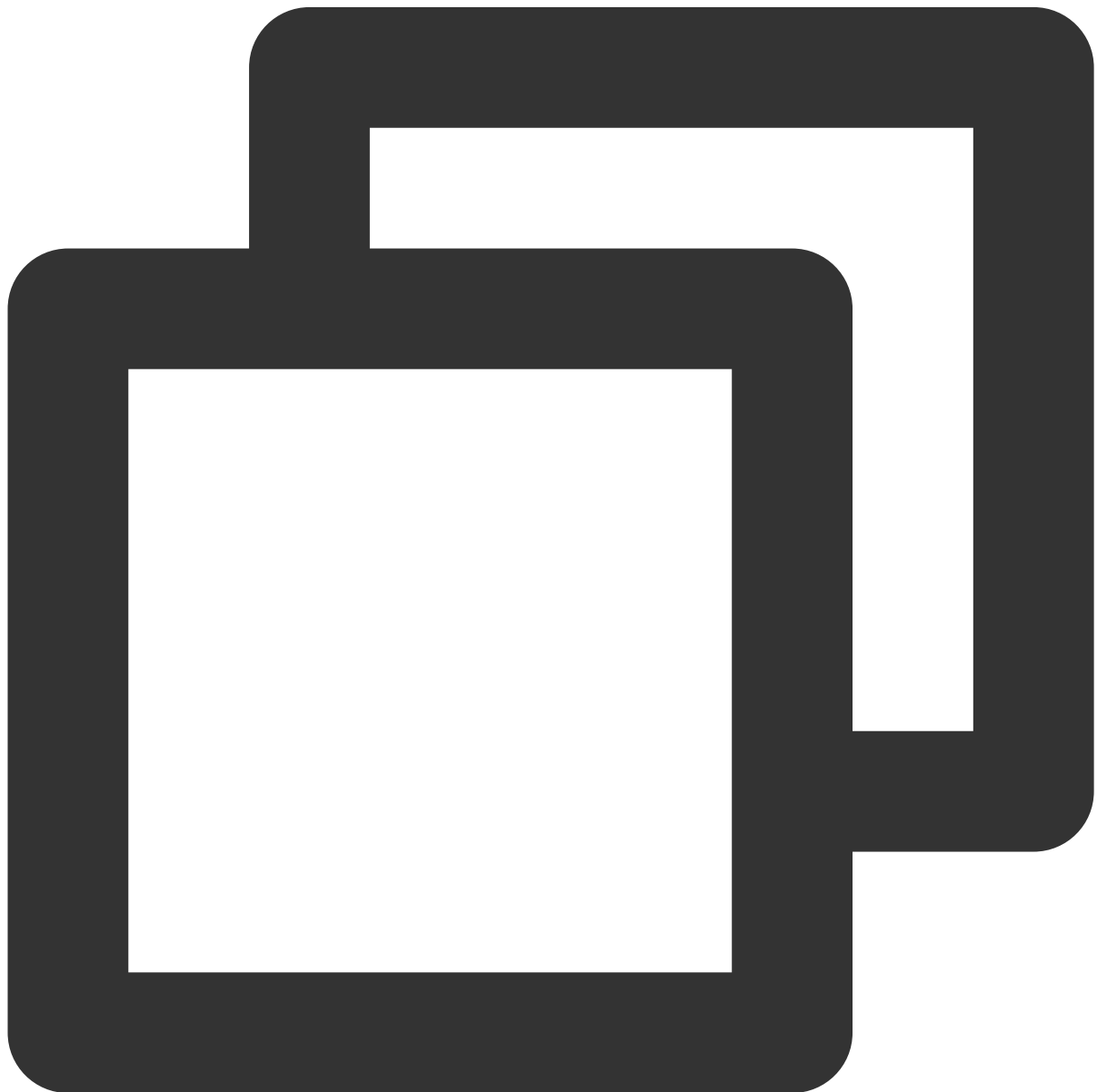
**The parameters are as follows:**

Parameter	Type	Meaning
nextSequence	NSInteger	Pagination Fetch flag, fill in 0 for the first Fetch, if the Callback returns TUIUserListResult with next_sequence not equal to zero, Pagination is required, pass in again to Fetch until it is 0

onSuccess	TUIUserListResponseBlock	Success Callback. Detailed Definition can be found in TUIRoomDefine.h file's TUIUserListResponseBlock
onError	TUIErrorBlock	Failed callback

## getUserInfo

Get Member data.



```
- (void)getUserInfo:(NSString *)userId  
    onSuccess:(TUIUserInfoBlock)onSuccess
```

```
onError: (TUIErrorBlock) onError;
```

The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID
onSuccess	TUIUserInfoBlock	Success Callback. Detailed Definition can be found in TUIRoomDefine.h file's TUIUserInfoBlock
onError	TUIErrorBlock	Error callback

## changeUserRole

Modify User Role (Only Administrator or Group owner can call).



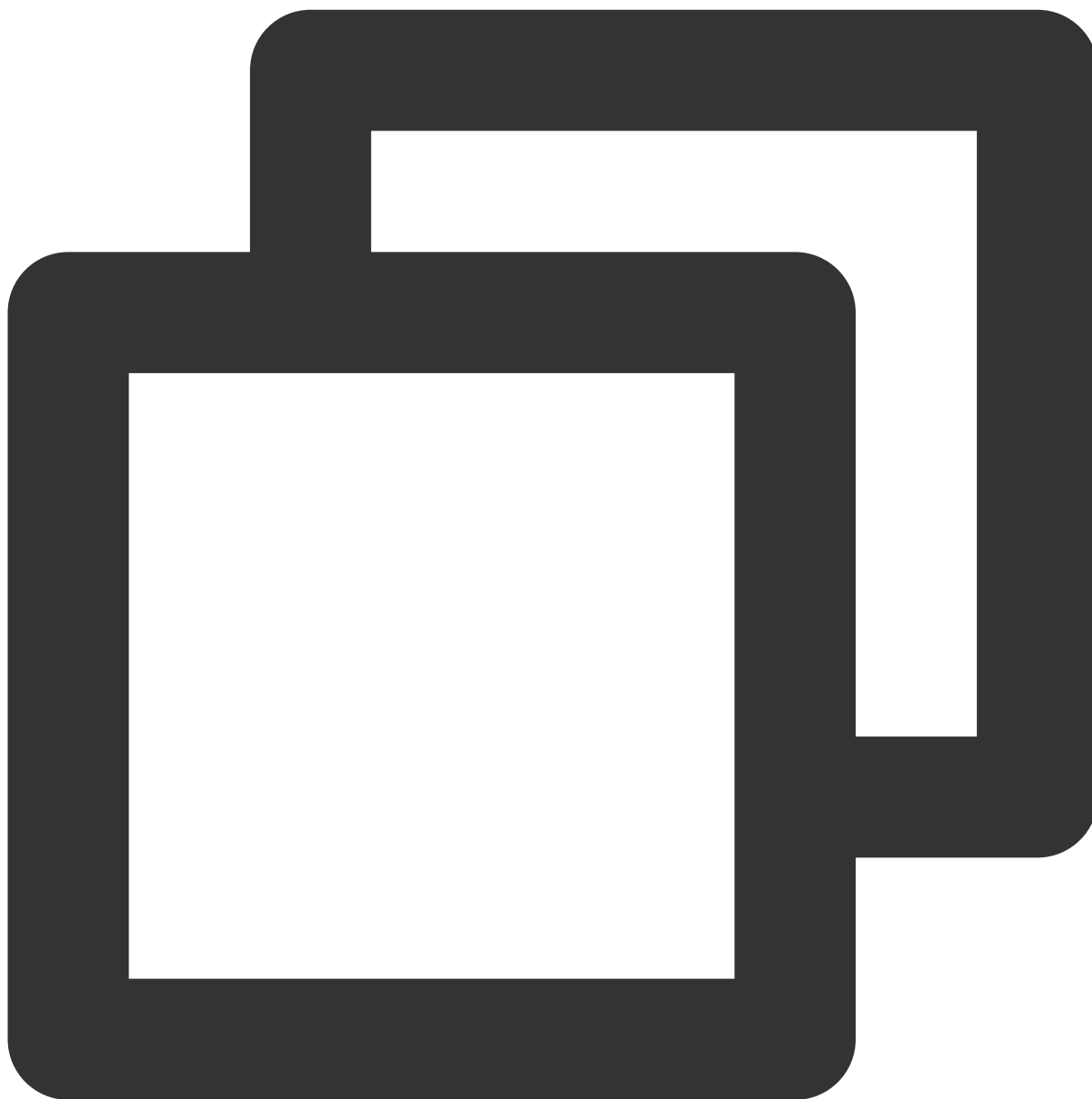
```
- (void)changeUserRoleWithUserId:(NSString *)userId
    role:(TUIRole)role
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID

role	<a href="#">TUIRole</a>	Role to switch to
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

## kickRemoteUserOutOfRoom



```
- (void)kickRemoteUserOutOfRoom:(NSString *)userId
    onSuccess:(TUISuccessBlock) onSuccess
    onError:(TUIErrorBlock) onError;
```

Kick Remote User out of the Room (Only Administrator or Group Owner can call).

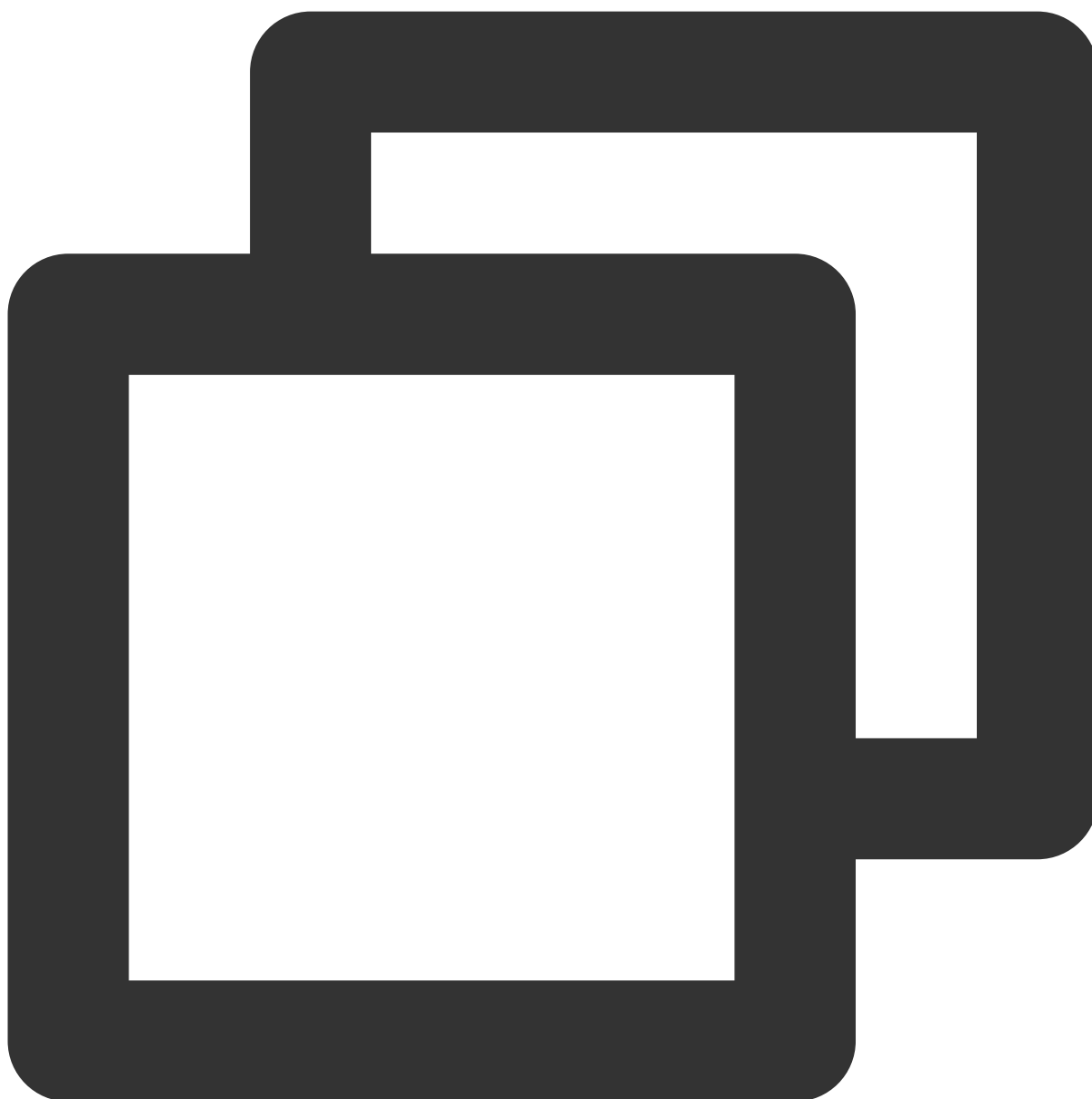
Parameter	Type	Meaning
userId	NSString *	User ID
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Failed callback

### **disableDeviceForAllUserByAdmin**

Control the permission status of all users in the current room to open Audio and Video Capturing devices, such as: Disable all users from opening mic, Disable all users from opening Camera, Disable all users from opening Screen Sharing

(Currently only available in conference scenes, and only Administrator or Group Owner can call).





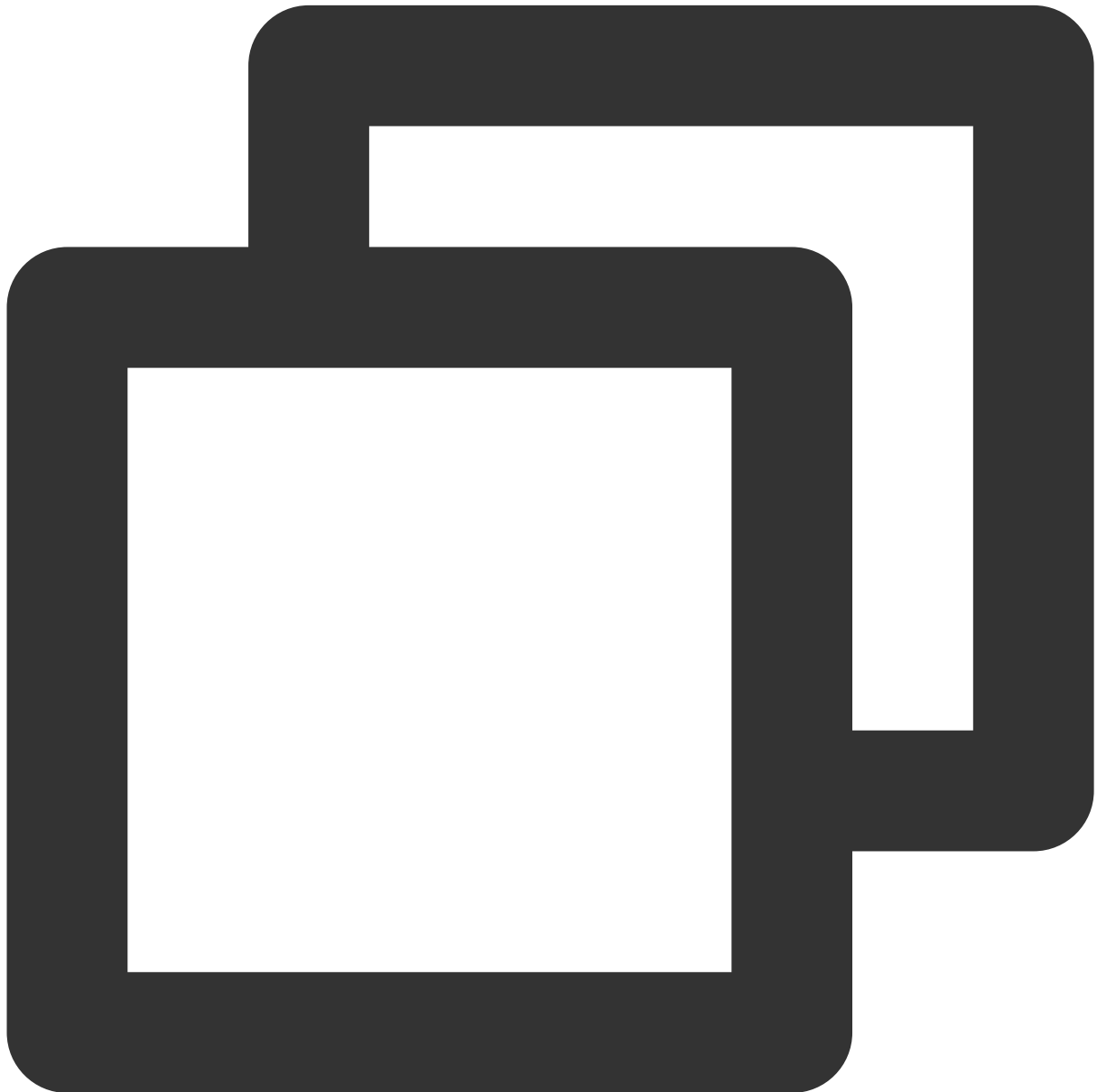
```
- (void)disableDeviceForAllUserByAdmin:(TUIMediaDevice)device
    isDisable:(BOOL)isDisable
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

Parameter	Type	Meaning
device	<a href="#">TUIMediaDevice</a>	Device Type
isDisable	BOOL	Disable or Not

onSuccess	TUISuccessBlock	Operation Success Callback
onError	TUIErrorBlock	Operation Failure Callback

## openRemoteDeviceByAdmin

Request Remote User to Open Media Device (Only Administrator or Group Owner can call).



```
- (TUIRequest *)openRemoteDeviceByAdmin:(NSString *)userId
```

```

        device: (TUIMediaDevice) device
        timeout: (NSTimeInterval) timeout
        onAccepted: (nullable TUIRequestAcceptedBlock) onAccepted
        onRejected: (nullable TUIRequestRejectedBlock) onRejected
        onCancelled: (nullable TUIRequestCancelledBlock) onCancelled
        onTimeout: (nullable TUIRequestTimeoutBlock) onTimeout
        onError: (nullable TUIRequestErrorBlock) onError;

```

Parameter	Type	Meaning
userId	NSString *	User ID
device	<a href="#">TUIMediaDevice</a>	Media Device
timeout	NSTimeInterval	Timeout Duration, Unit Second, if set to 0, SDK will not perform timeout detection and will not trigger timeout callback
onAccepted	nullable TUIRequestAcceptedBlock	Invitation Accepted Callback
onRejected	nullable TUIRequestRejectedBlock	Invitation Rejected Callback
onCancelled	nullable TUIRequestCancelledBlock	Invitation Canceled Callback
onTimeout	nullable TUIRequestTimeoutBlock	Invitation Timeout Unhandled Callback
onError	nullable TUIRequestErrorBlock	Invitation Error Callback

Return Value	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Body

## closeRemoteDeviceByAdmin

Close Remote User's Media Device (Only Administrator or Group Owner can call).



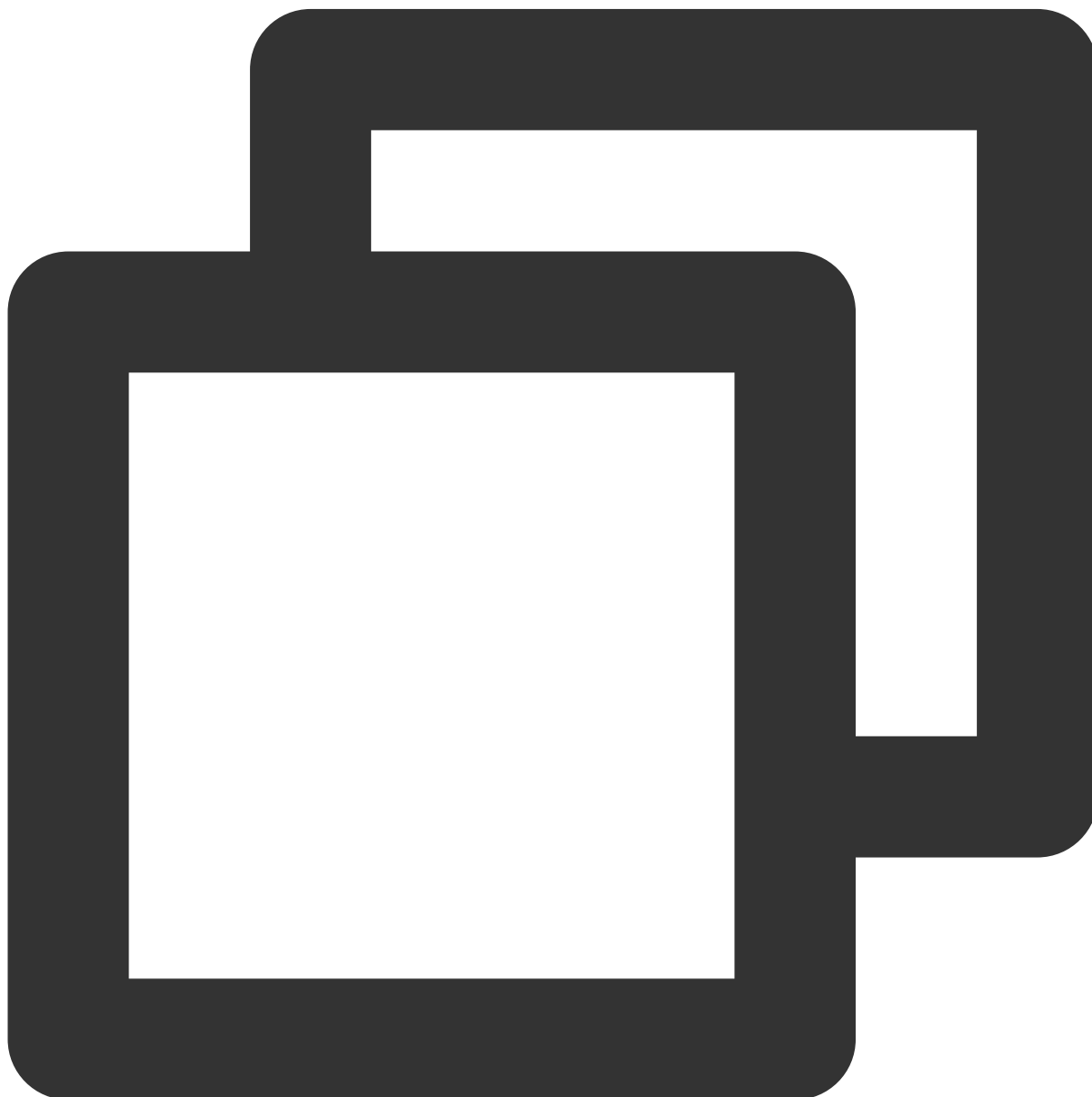
```
- (void)closeRemoteDeviceByAdmin:(NSString *)userId
    device:(TUIMediaDevice) device
    onSuccess:(TUISuccessBlock) onSuccess
    onError:(TUIErrorBlock) onError;
```

Parameter	Type	Meaning
userId	NSString *	User ID
device	<a href="#">TUIMediaDevice</a>	Media Device

onSuccess	TUISuccessBlock	Call Success Callback
onError	TUIErrorBlock	Call Failure Callback

## **applyToAdminToOpenLocalDevice**

Request to Open Local Media Device (Available for ordinary users).



```
- (TUIRequest *)applyToAdminToOpenLocalDevice:(TUIMediaDevice) device
```

```

        timeout:(NSTimeInterval)timeout
        onAccepted:(nullable TUIRequestAcceptedBlock)onA
        onRejected:(nullable TUIRequestRejectedBlock)onR
        onCancelled:(nullable TUIRequestCancelledBlock)on
        onTimeout:(nullable TUIRequestTimeoutBlock)onTi
        onError:(nullable TUIRequestErrorBlock)onErro

```

Parameter	Type	Meaning
device	<a href="#">TUIMediaDevice</a>	Media Device
timeout	NSTimeInterval	Timeout Duration, Unit Second, if set to 0, SDK will not perform timeout detection and will not trigger timeout callback
onAccepted	nullable TUIRequestAcceptedBlock	Invitation Accepted Callback
onRejected	nullable TUIRequestRejectedBlock	Invitation Rejected Callback
onCancelled	nullable TUIRequestCancelledBlock	Invitation Canceled Callback
onTimeout	nullable TUIRequestTimeoutBlock	Invitation Timeout Unhandled Callback
onError	nullable TUIRequestErrorBlock	Invitation Error Callback

Return Value	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Body

## setMaxSeatCount

Set Maximum Seat Count (Only supported when entering the room and creating the room).

When roomType is TUIRoomTypeConference (Educational and Conference Scenes), maxSeatCount value is not limited;

When roomType is TUIRoomTypeLivingRoom (Live Streaming Scene), maxSeatCount is limited to 16;



```
- (void)setMaxSeatCount:(NSUInteger)maxSeatCount  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

Parameter	Type	Meaning
maxSeatCount	NSUInteger	Maximum microphone positions
onSuccess	TUISuccessBlock	Successful callback

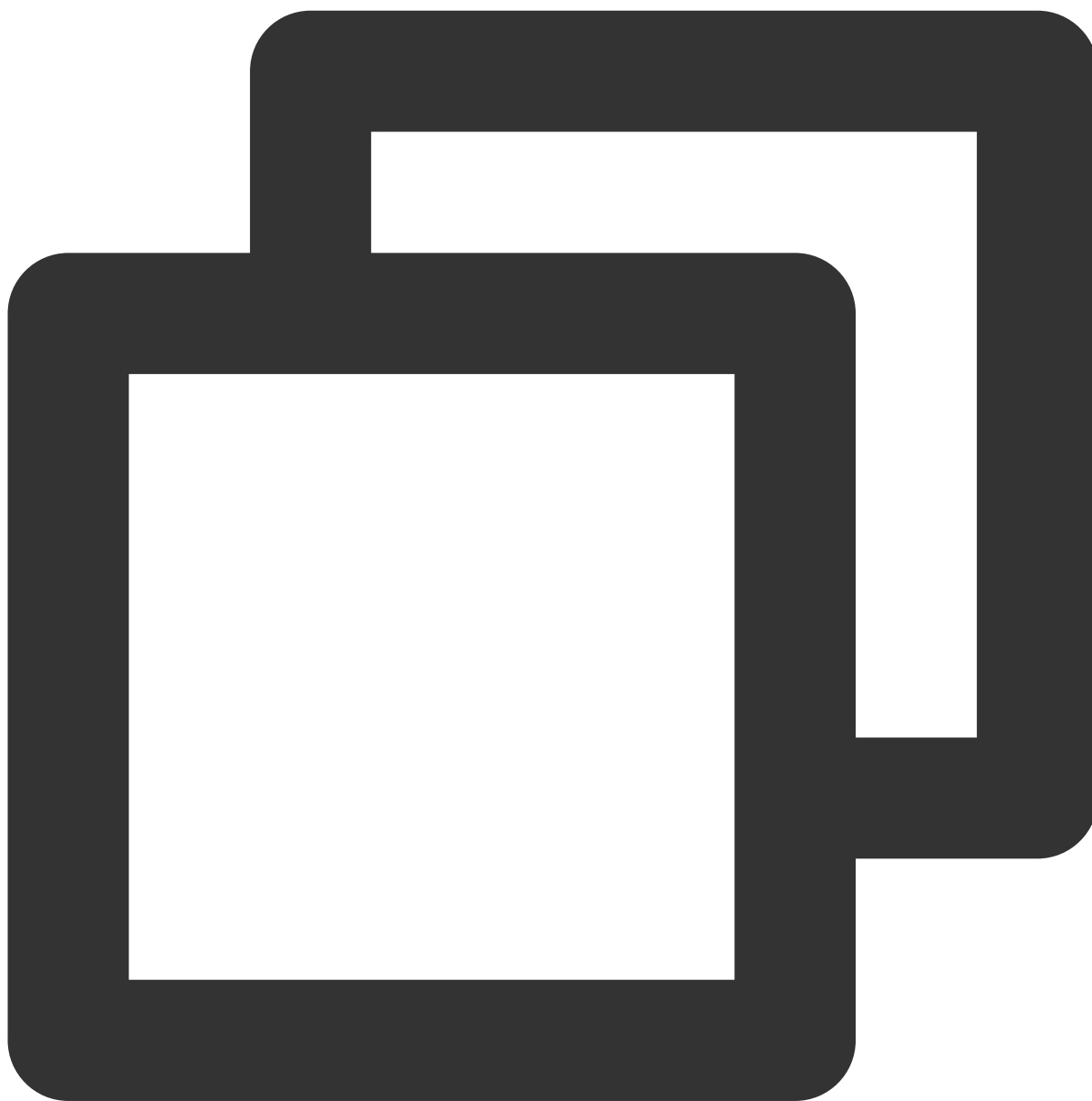
onError

TUIErrorBlock

Error callback

## getSeatList

Get Seat List.



```
- (void) getSeatList: (TUISeatListResponseBlock) onSuccess  
    onError: (TUIErrorBlock) onError;
```

The parameters are as follows:

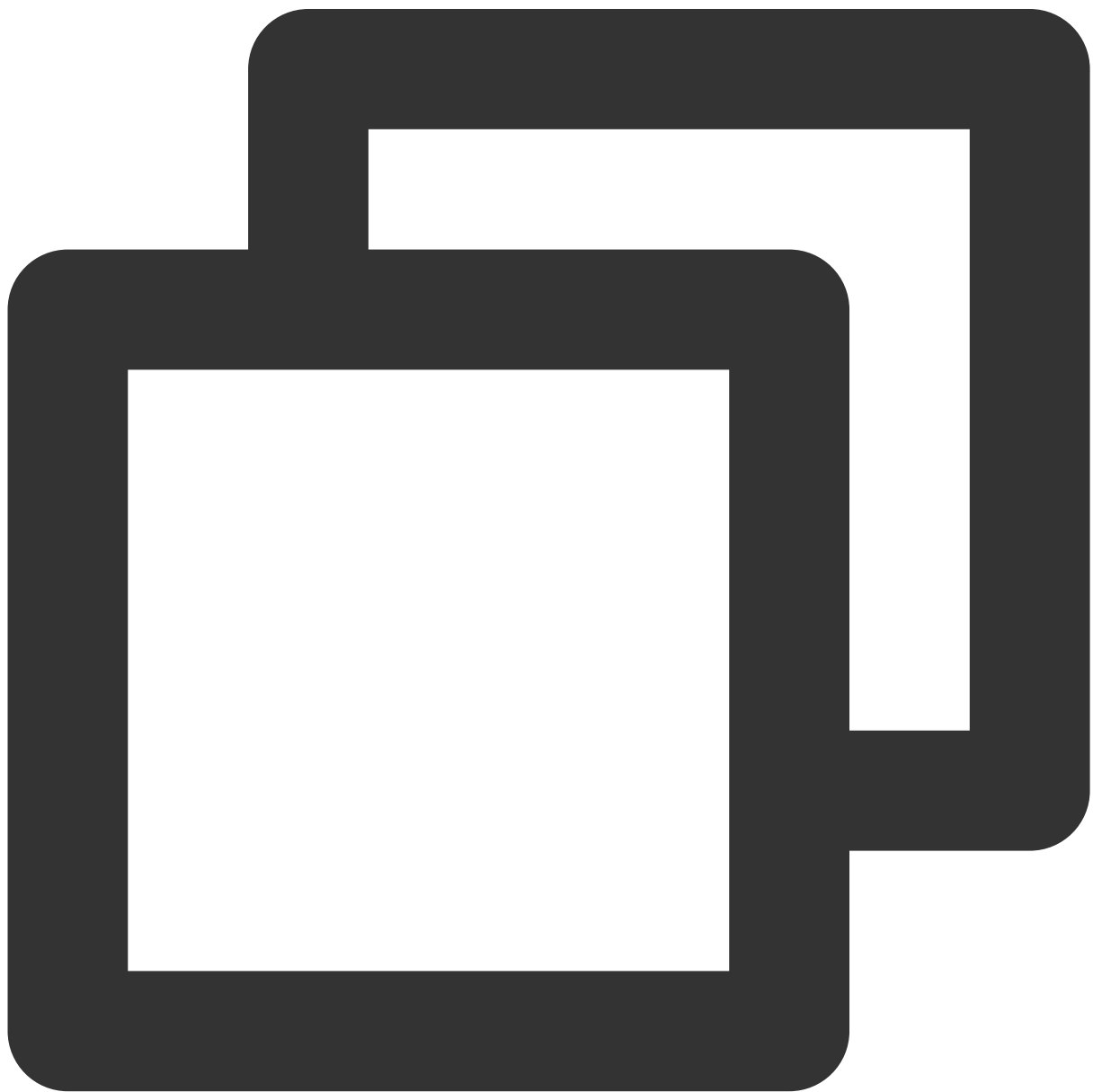
Parameter	Type	Meaning
-----------	------	---------



onSuccess	TUISeatListResponseBlock	Success Callback. For detailed definition, please refer to TUIRoomDefine.h for TUISeatListResponseBlock and <a href="#">TUISeatInfo</a> definition
onError	TUIErrorBlock	Error callback

## lockSeatByAdmin

Lock Seat (Only Administrator or Group Owner can call, including position lock, Audio status lock and Video status lock).



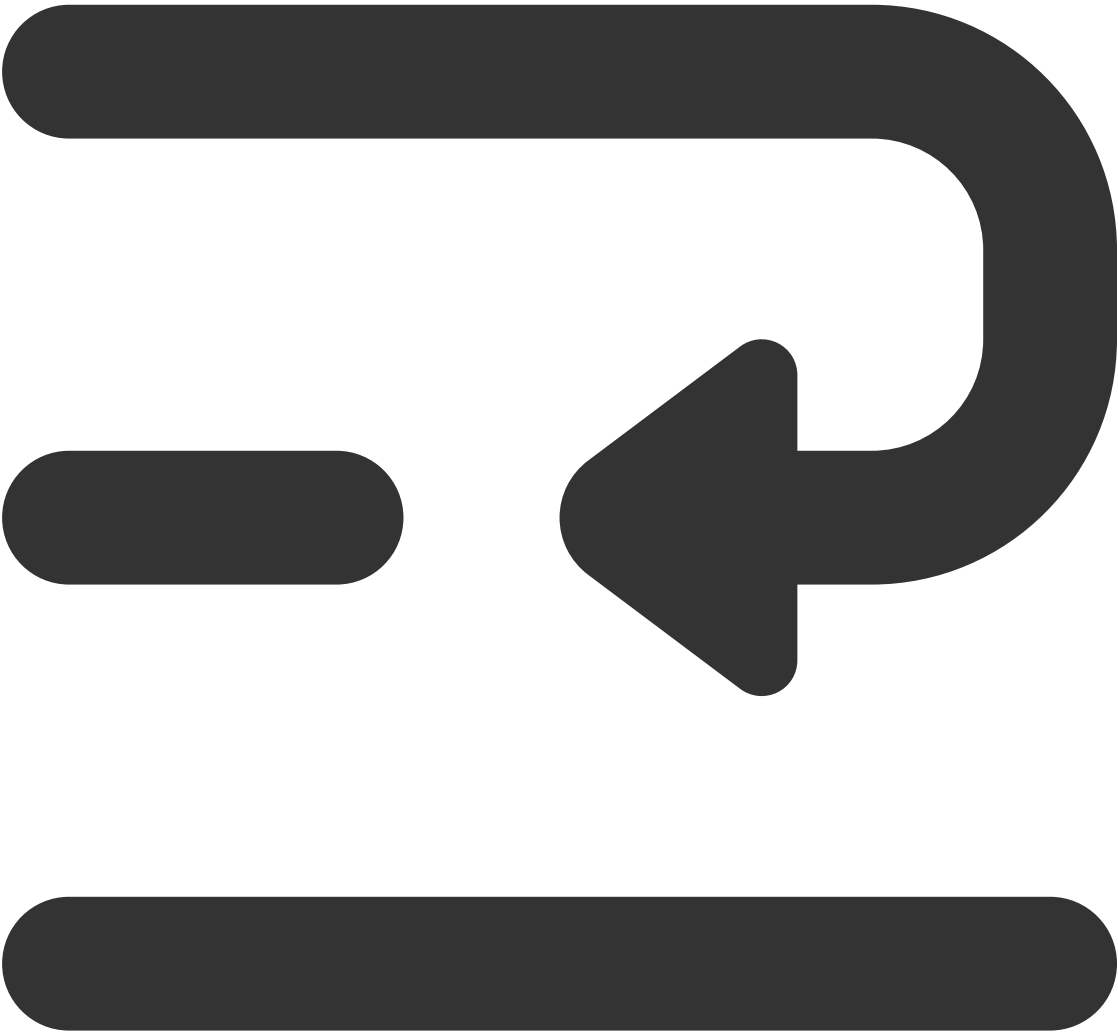
```
- (void)lockSeatByAdmin:(NSInteger)seatIndex
    lockMode:(TUISeatLockParams *)lockParams
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

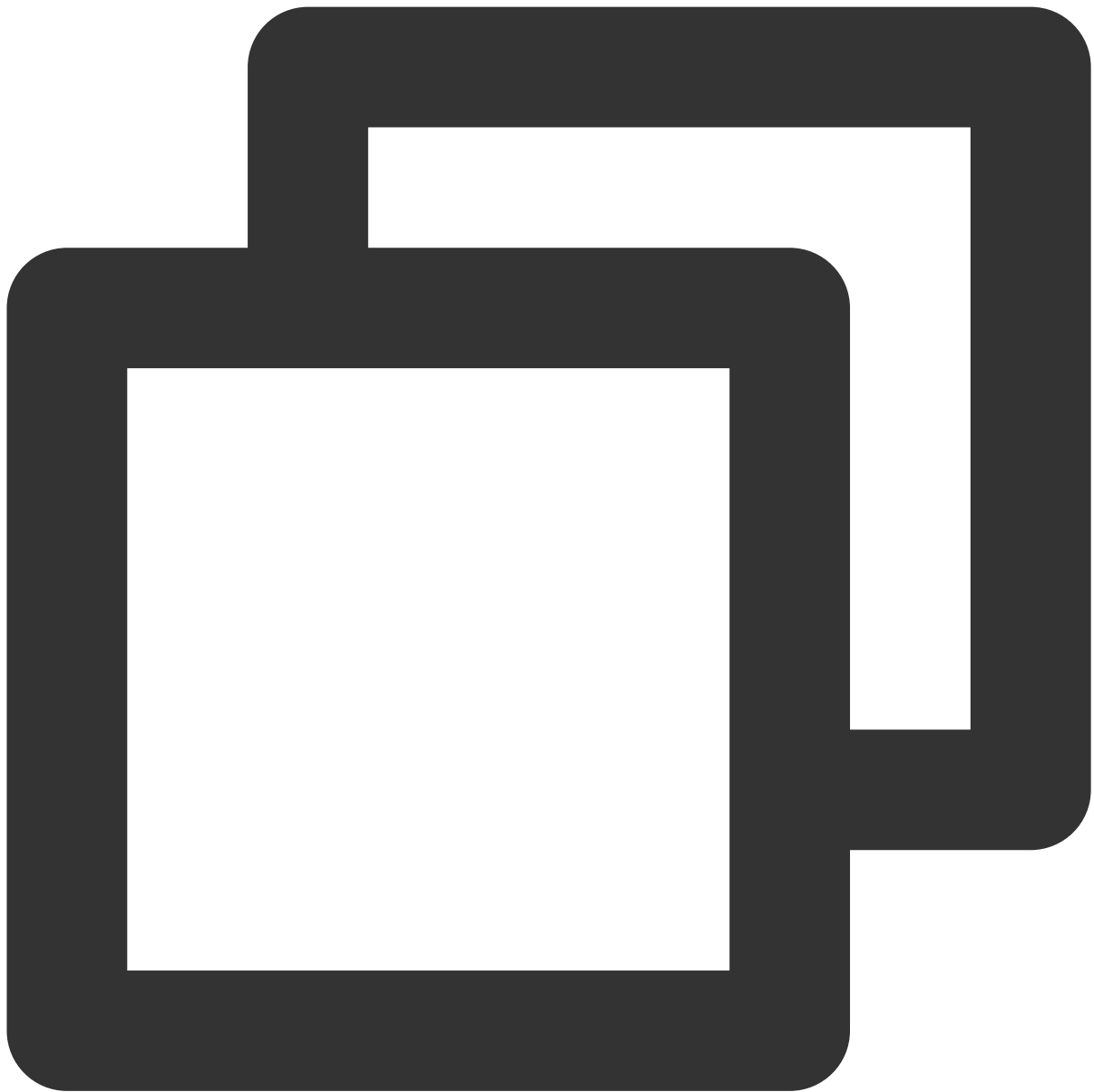
The parameters are as follows:

Parameter	Type	Meaning
seatIndex	NSInteger	Seat Number
lockParams	<a href="#">TUISeatLockParams</a> *	Lock Seat Parameters, for detailed definition, please refer to TUIRoomDefine.h for <a href="#">TUISeatLockParams</a> definition
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## takeSeat

Local User Go Live (In Free Speech Mode, no application is required, in Mic Control Mode, application to the host or administrator is required to allow Go Live, in Live Streaming Scene, users can Go Live freely, after Go Live, users can speak, in Conference Scene, no need to call this interface to speak).





```
- (TUIRequest *)takeSeat:(NSInteger)seatIndex
    timeout:(NSTimeInterval)timeout
    onAccepted:(TUIRequestAcceptedBlock)onAccepted
    onRejected:(TUIRequestRejectedBlock)onRejected
    onCancelled:(TUIRequestCancelledBlock)onCancelled
    onTimeout:(TUIRequestTimeoutBlock)onTimeout
    onError:(TUIRequestErrorBlock)onError;
```

The parameters are as follows:

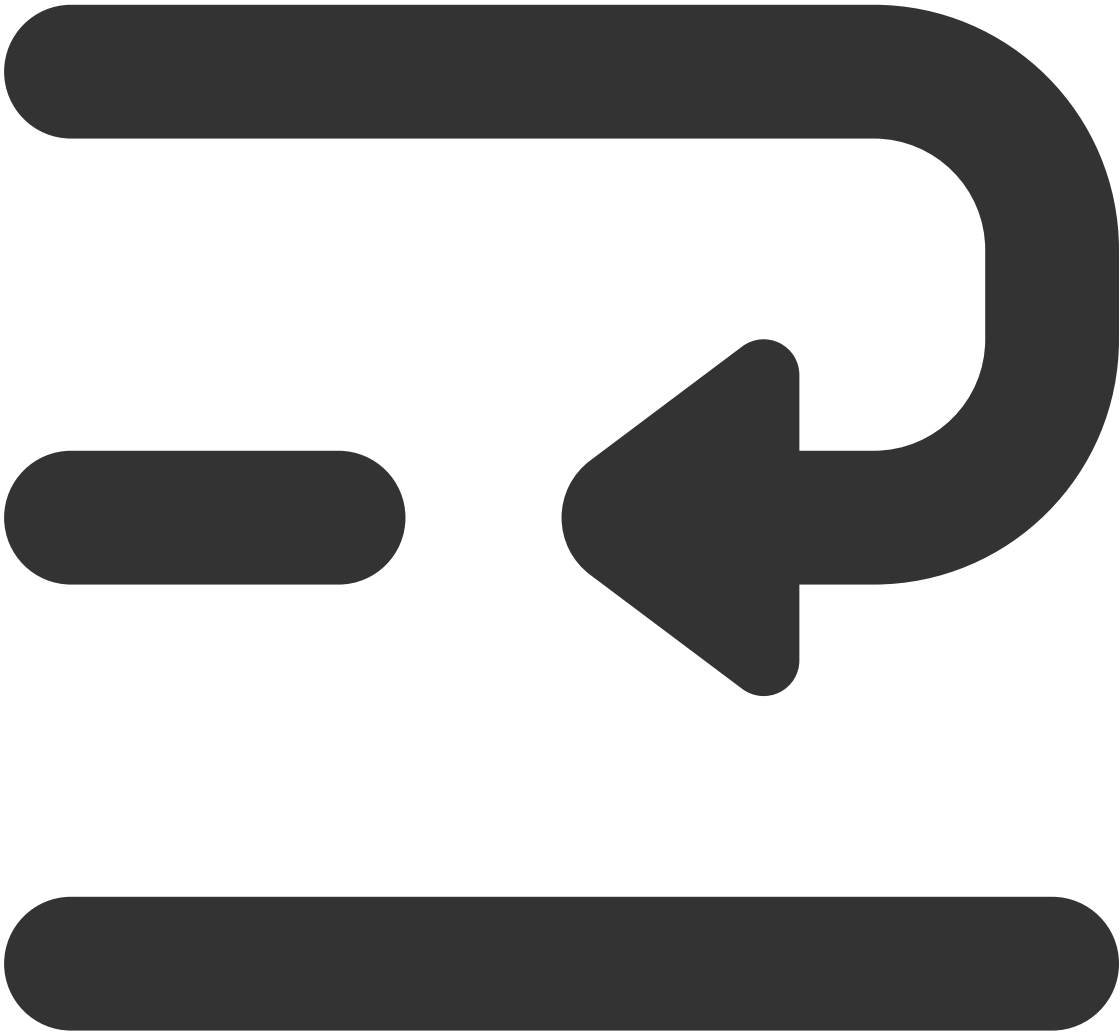
Parameter	Type	Meaning
-----------	------	---------

seatIndex	NSInteger	Seat Number
timeout	NSTimeInterval	Timeout Duration
onAccepted	TUIRequestAcceptedBlock	Signaling Accepted Callback
onRejected	TUIRequestRejectedBlock	Signaling Rejected Callback
onCancelled	TUIRequestCancelledBlock	Signaling Canceled Callback
onTimeout	TUIRequestTimeoutBlock	Signaling Timeout Callback
onError	TUIRequestErrorBlock	Error Callback

Return Value	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Body

## leaveSeat

Local User Get Off the Mic (In Free Speech Mode, no application is required).





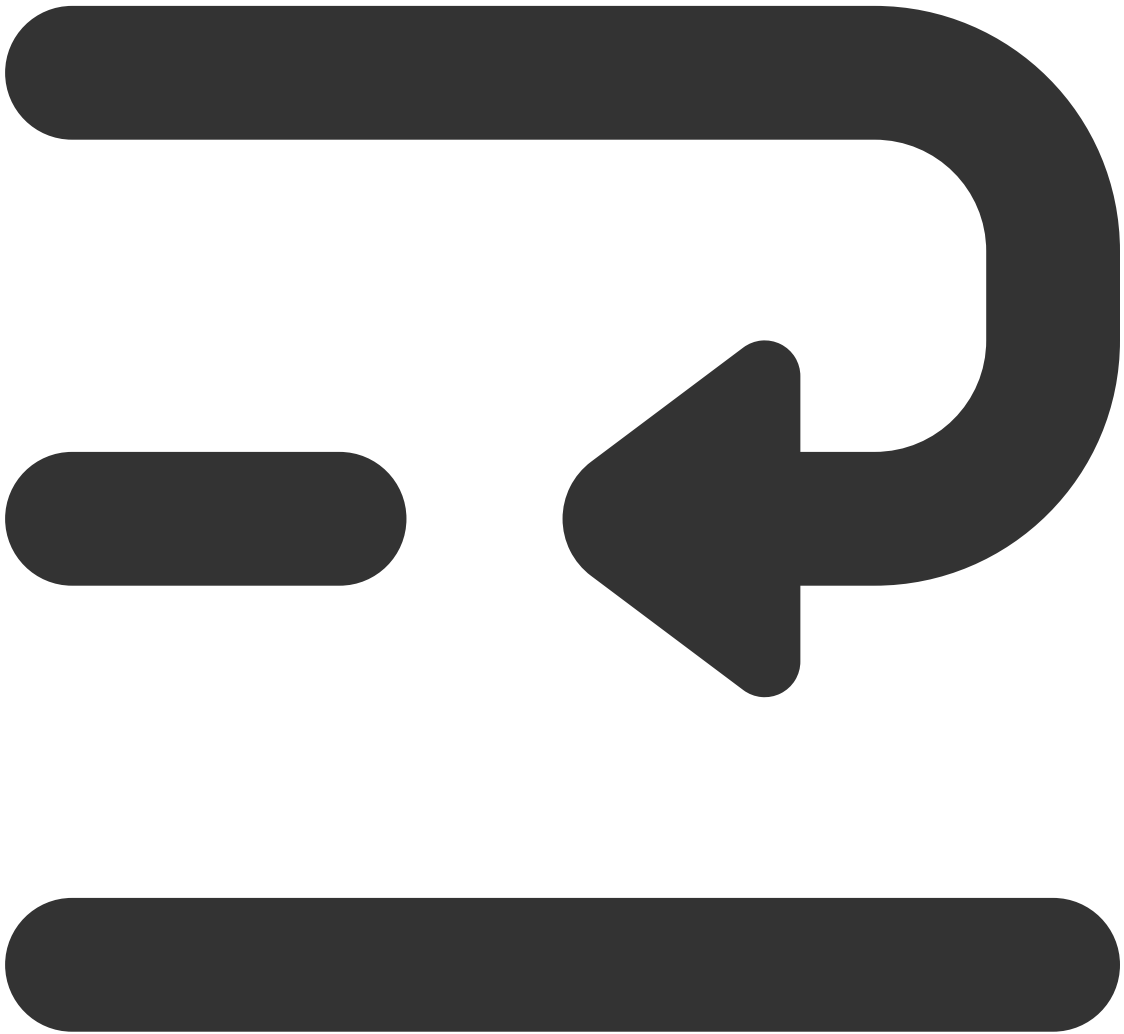
```
- (void)leaveSeat:(TUISuccessBlock) onSuccess  
    onError:(TUIErrorBlock) onError;
```

**The parameters are as follows:**

Parameter	Type	Meaning
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## takeUserOnSeatByAdmin

Host/Administrator Invite User to Go Live.







```
- (TUIRequest *)takeUserOnSeatByAdmin:(NSInteger)seatIndex
    userId:(NSString *)userId
    timeout:(NSTimeInterval)timeout
    onAccepted:(TUIRequestAcceptedBlock)onAccepted
    onRejected:(TUIRequestRejectedBlock)onRejected
    onCancelled:(TUIRequestCancelledBlock)onCancelled
    onTimeout:(TUIRequestTimeoutBlock)onTimeout
    onError:(TUIRequestErrorBlock)onError;
```

The parameters are as follows:

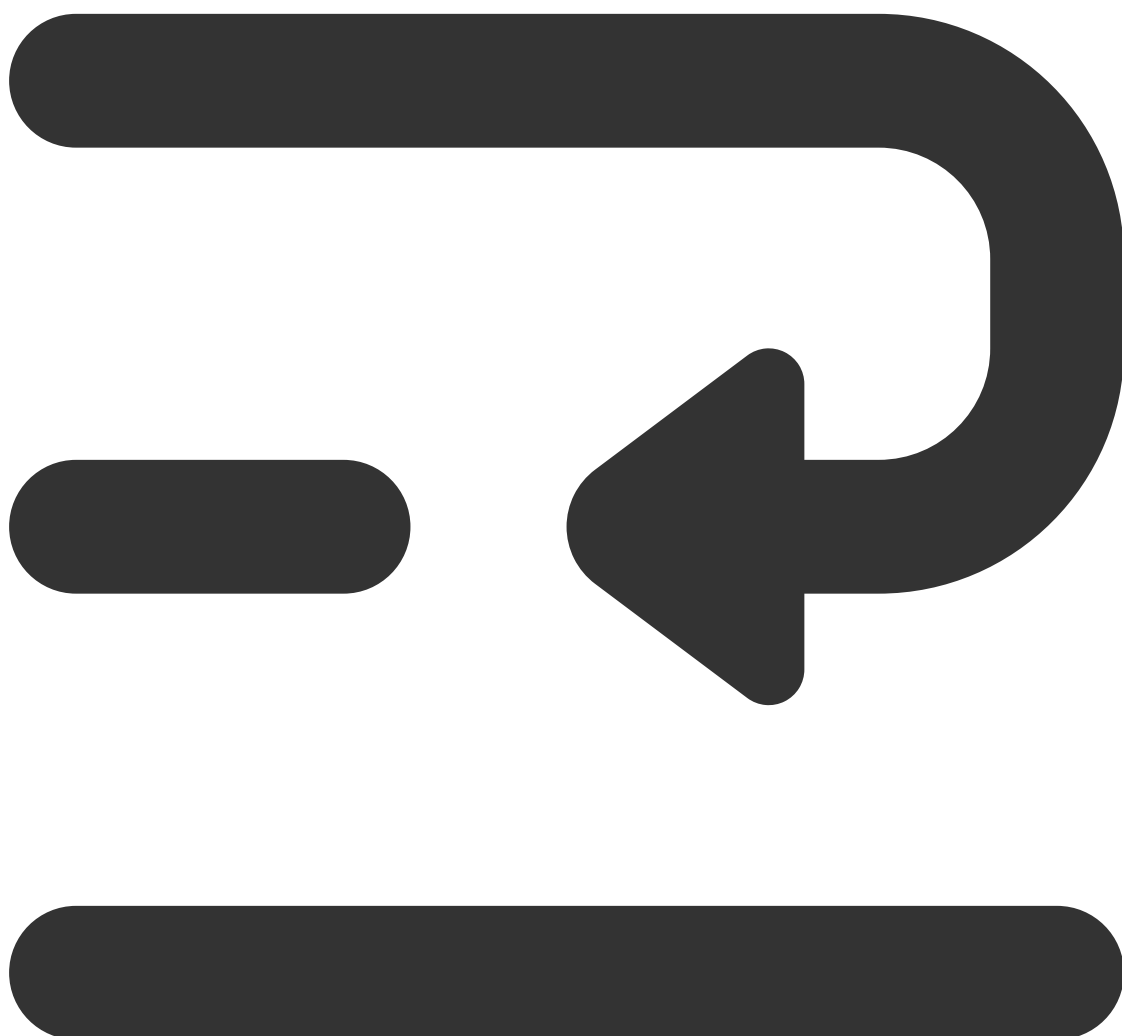
--	--	--

Parameter	Type	Meaning
seatIndex	NSInteger	Seat Number
userId	NSString *	User ID
timeout	NSTimeInterval	Timeout Duration (Unit Second, if set to 0, SDK will not perform timeout detection and will not trigger timeout callback)
onAccepted	TUIRequestAcceptedBlock	Invitation Accepted Callback
onRejected	TUIRequestRejectedBlock	Invitation Rejected Callback
onCancelled	TUIRequestCancelledBlock	Invitation Canceled Callback
onTimeout	TUIRequestTimeoutBlock	Invitation Timeout Unhandled Callback
onError	TUIRequestErrorBlock	Invitation Error Callback

Return Value	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Body

## kickUserOffSeatByAdmin

Host/Administrator Kick User Off the Mic.





```
- (void)kickUserOffSeatByAdmin:(NSInteger)seatIndex
    userId:(NSString *)userId
    onSuccess:(TUISuccessBlock)onSuccess
    onError:(TUIErrorBlock)onError;
```

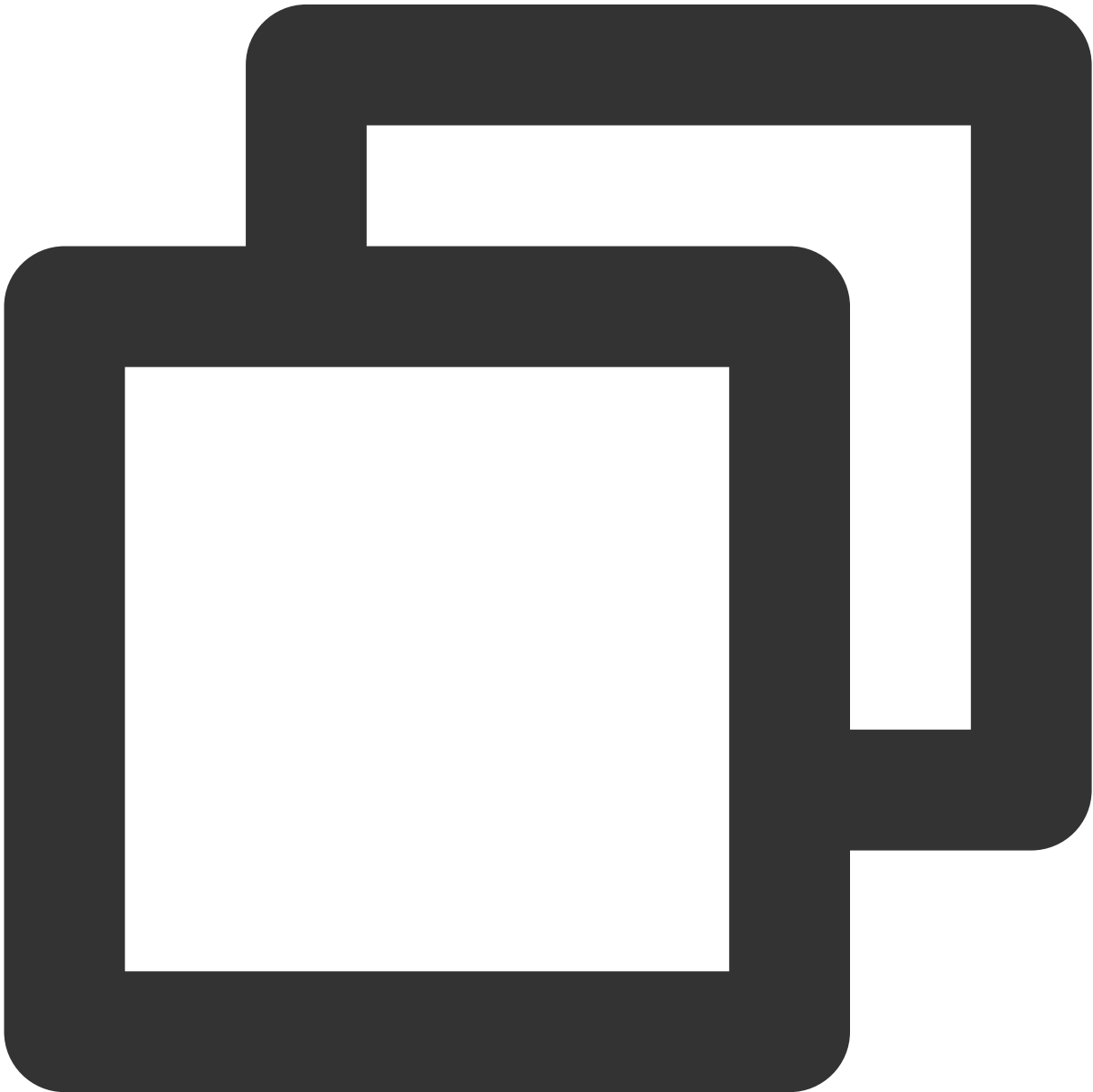
The parameters are as follows:

Parameter	Type	Meaning
seatIndex	NSInteger	Seat Number

userId	NSString *	User ID
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

### cancelRequest

Cancel Request.



```
- (void)cancelRequest:(NSInteger)requestId
```

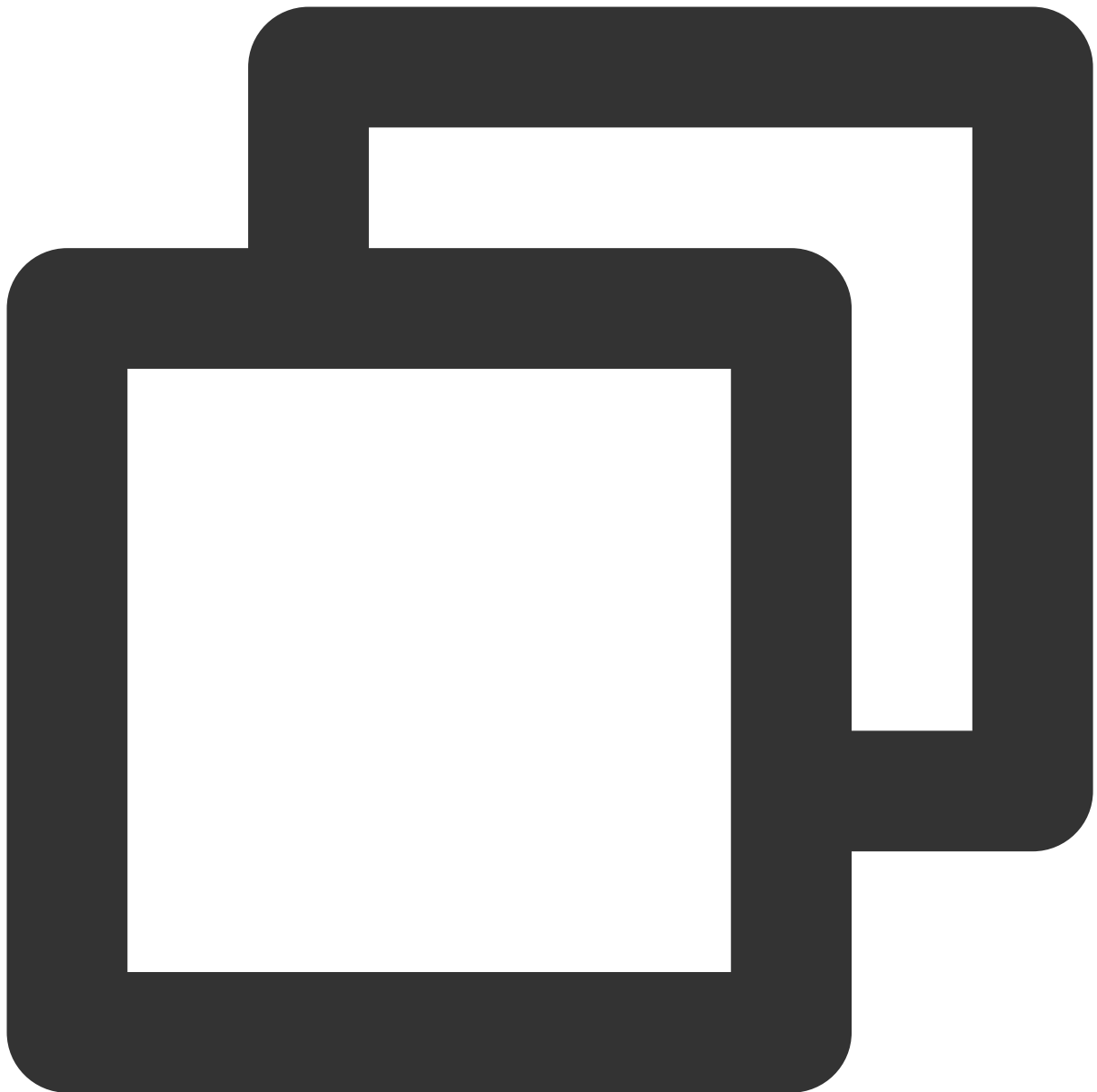
```
onSuccess:(TUISuccessBlock)onSuccess  
onError:(TUIErrorBlock)onError;
```

The parameters are as follows:

Parameter	Type	Meaning
requestId	NSString *	Request ID. Returned by the sending request interface
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## responseRemoteRequest

Reply Request.



```
- (void)responseRemoteRequest:(NSString *)requestId  
    agree:(BOOL)agree  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

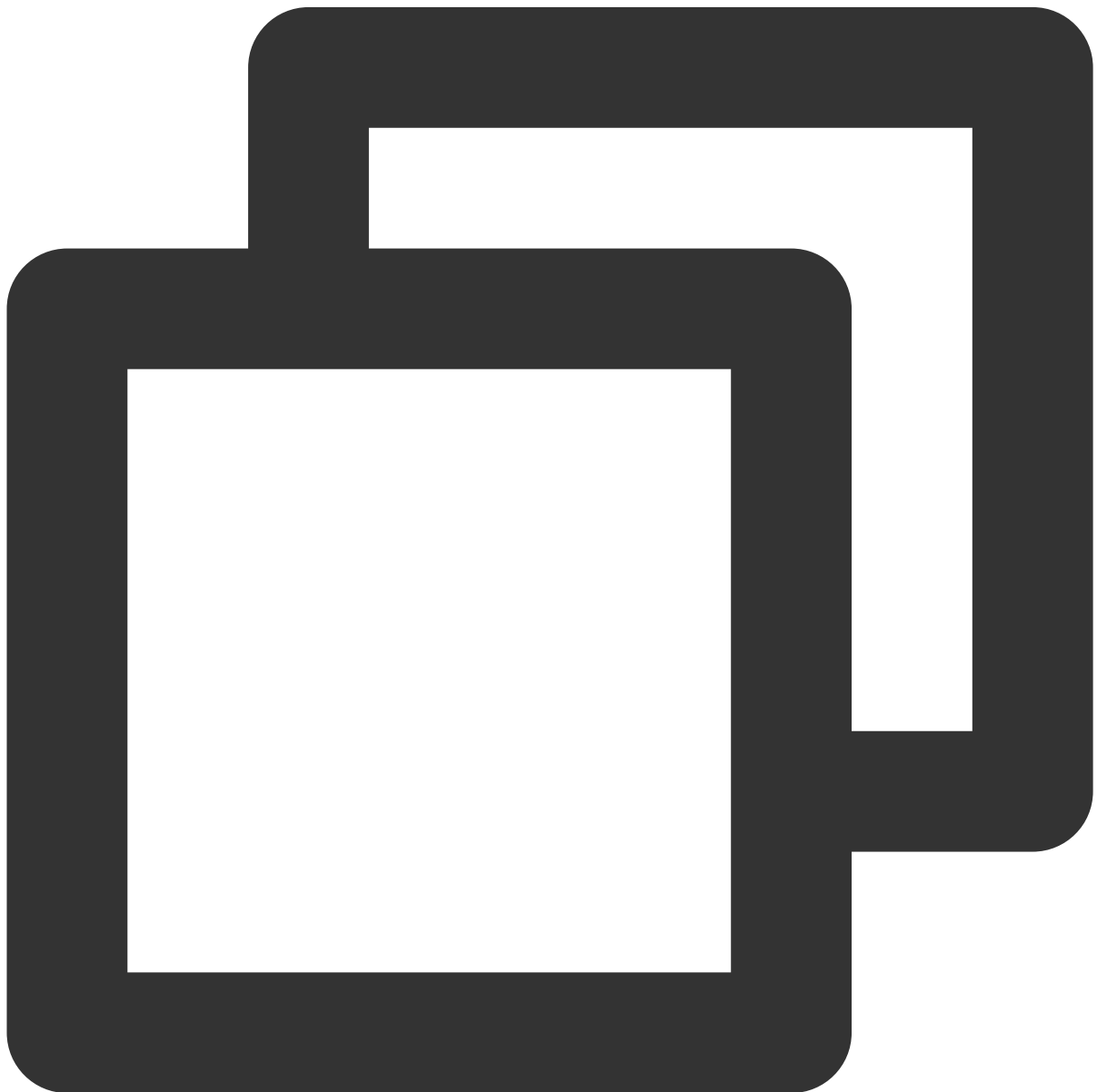
The parameters are as follows:

Parameter	Type	Meaning
requestId	NSString *	Request ID. Returned by the sending request interface or OnRequestReceived event notification

agree	BOOL	Agree or not. YES: Agree, NO: Reject
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## sendMessage

Send Text Message.



```
- (void)sendMessage:(NSString *)message
```



```
onSuccess:(TUISuccessBlock) onSuccess  
onError:(TUIErrorBlock) onError;
```

The parameters are as follows:

Parameter	Type	Meaning
message	NSString *	Message Content
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## sendCustomMessage

Send Custom Message.



```
- (void)sendCustomMessage:(NSString *)message  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

**The parameters are as follows:**

Parameter	Type	Meaning
message	NSString *	Message Content
onSuccess	TUISuccessBlock	Successful callback

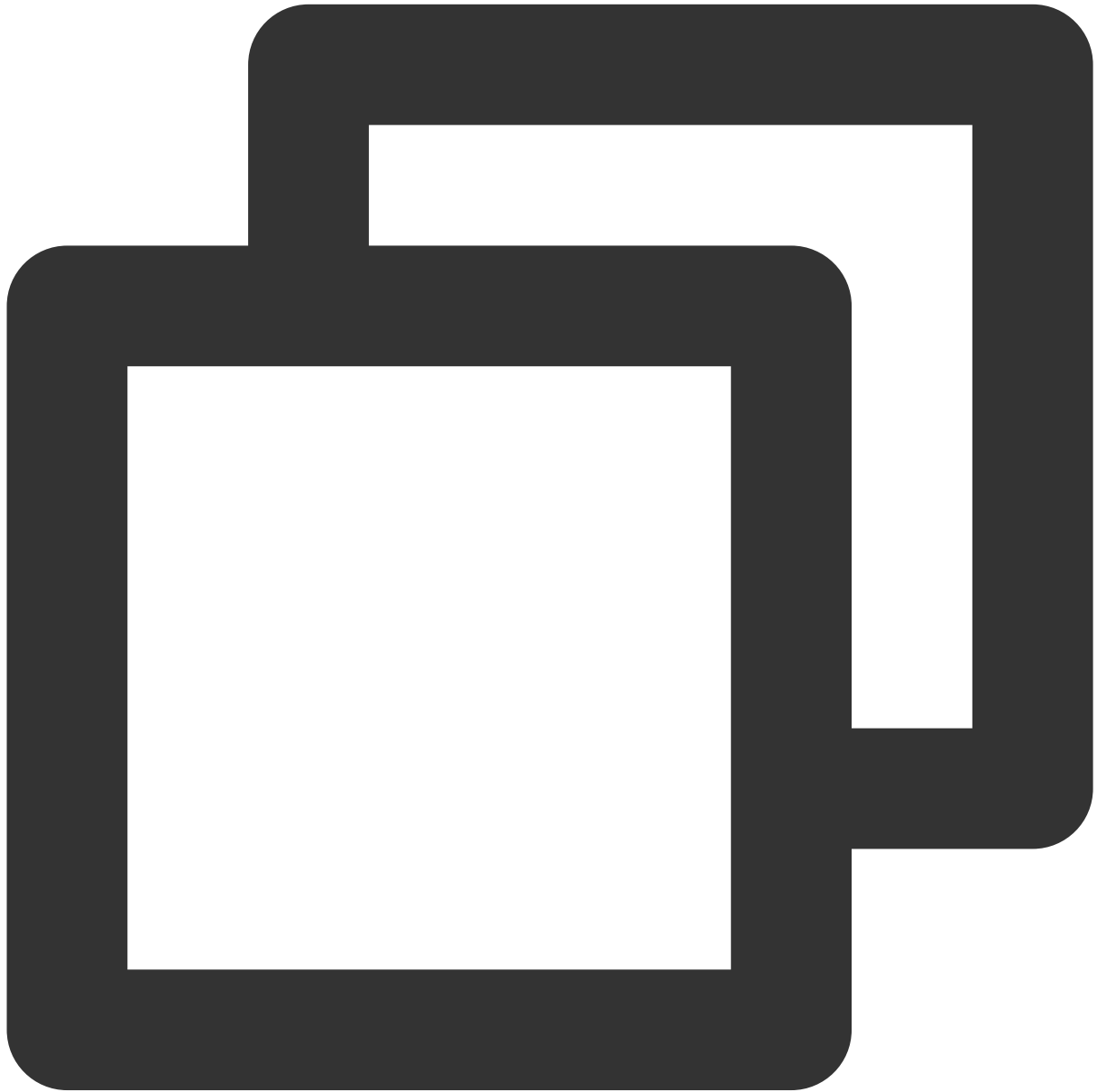
onError

TUIErrorBlock

Error callback

**disableSendingMessageByAdmin**

Disable Remote User's Text Message Sending Ability (Only Administrator or Group Owner can call).

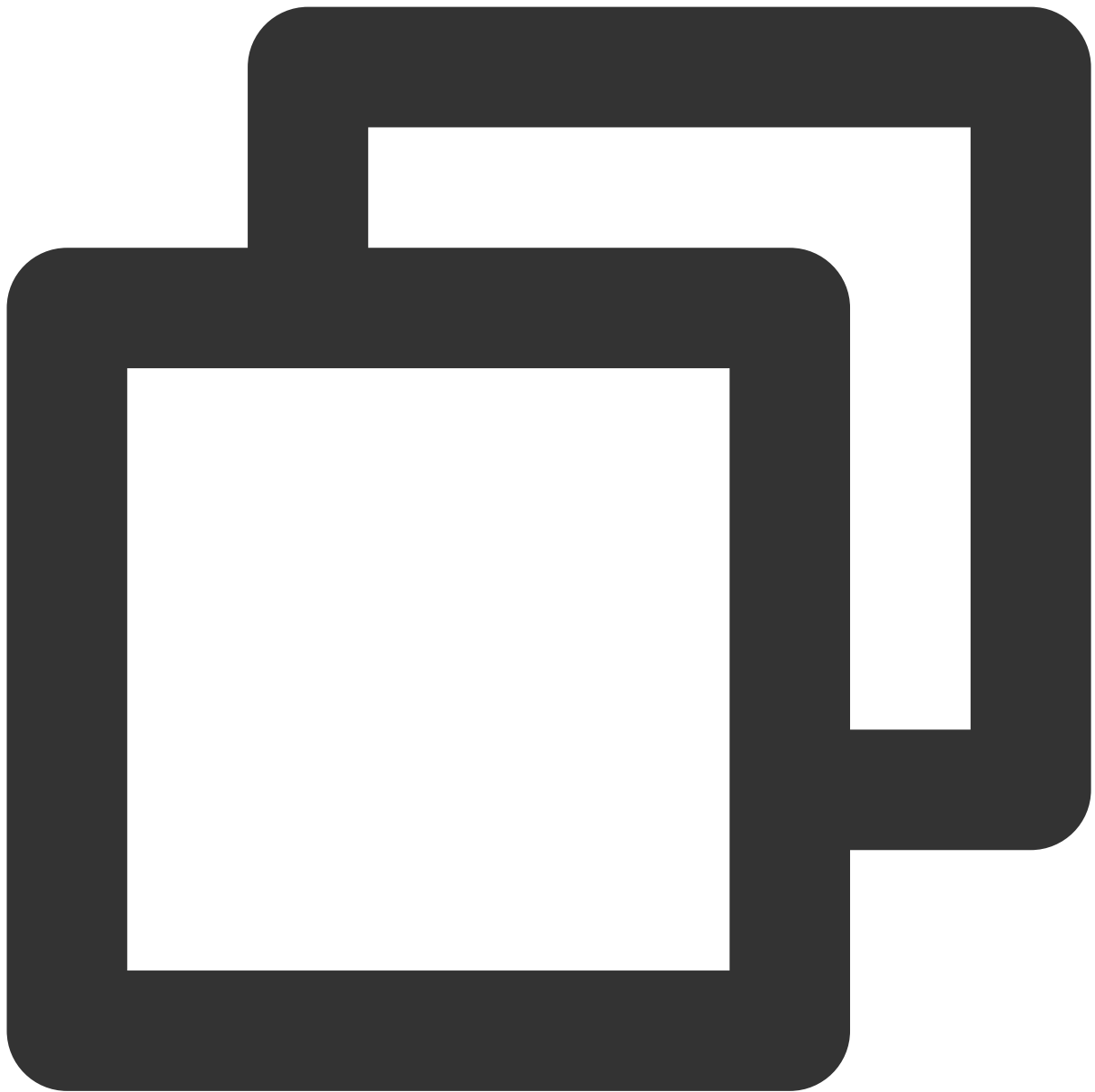


```
- (void)disableSendingMessageByAdmin:(NSString *)userId
    isDisable:(BOOL)isDisable
  onSuccess:(TUISuccessBlock)onSuccess
  onError:(TUIErrorBlock)onError;
```

Parameter	Type	Meaning
userId	NSString *	User ID
isDisable	BOOL	Disable or Not
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

### **disableSendingMessageForAllUser**

Disable All Users' Text Message Sending Ability (Only Administrator or Group Owner can call).

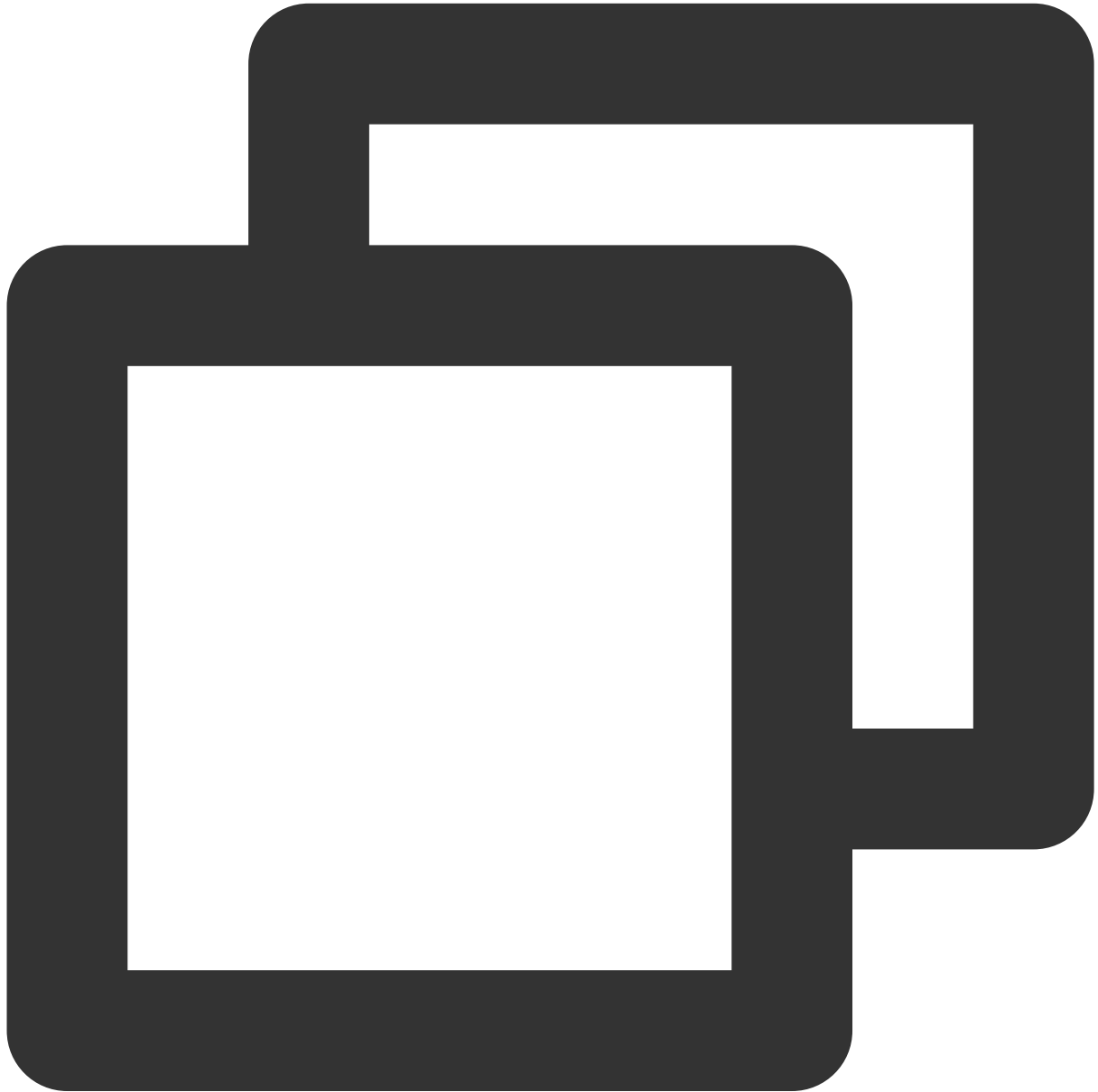


```
- (void)disableSendingMessageForAllUser:(BOOL)isDisable  
    onSuccess:(TUISuccessBlock)onSuccess  
    onError:(TUIErrorBlock)onError;
```

Parameter	Type	Meaning
isDisable	BOOL	Disable or Not
onSuccess	TUISuccessBlock	Successful callback
onError	TUIErrorBlock	Error callback

## getDeviceManager

Get Device Manager Object.

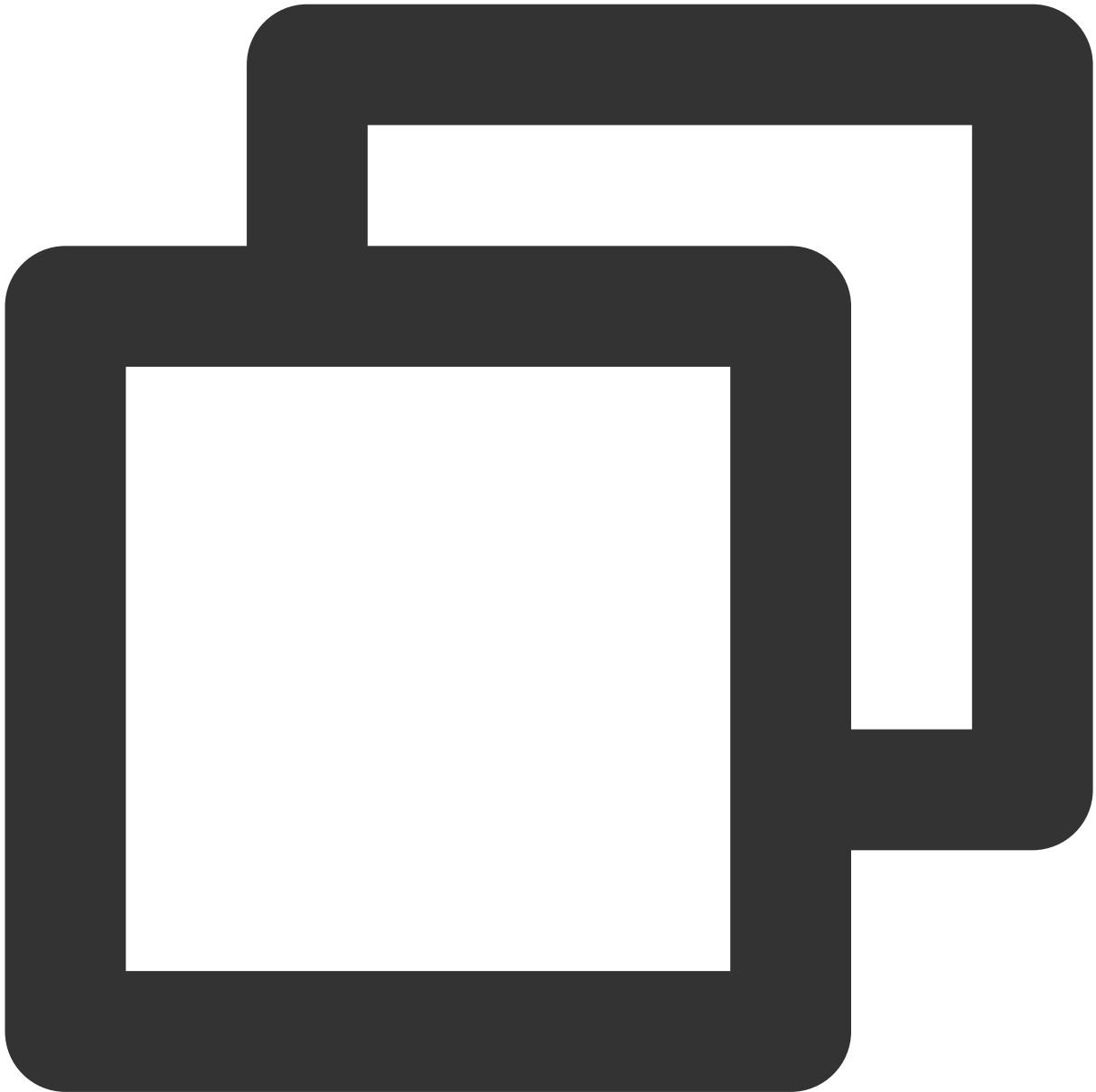


```
- (TXDeviceManager *)getDeviceManager;
```

Return Value	Type	Meaning
manager	TXDeviceManager *	TXDeviceManager Object

getBeautyManager

Get Sound Effect Manager Object.

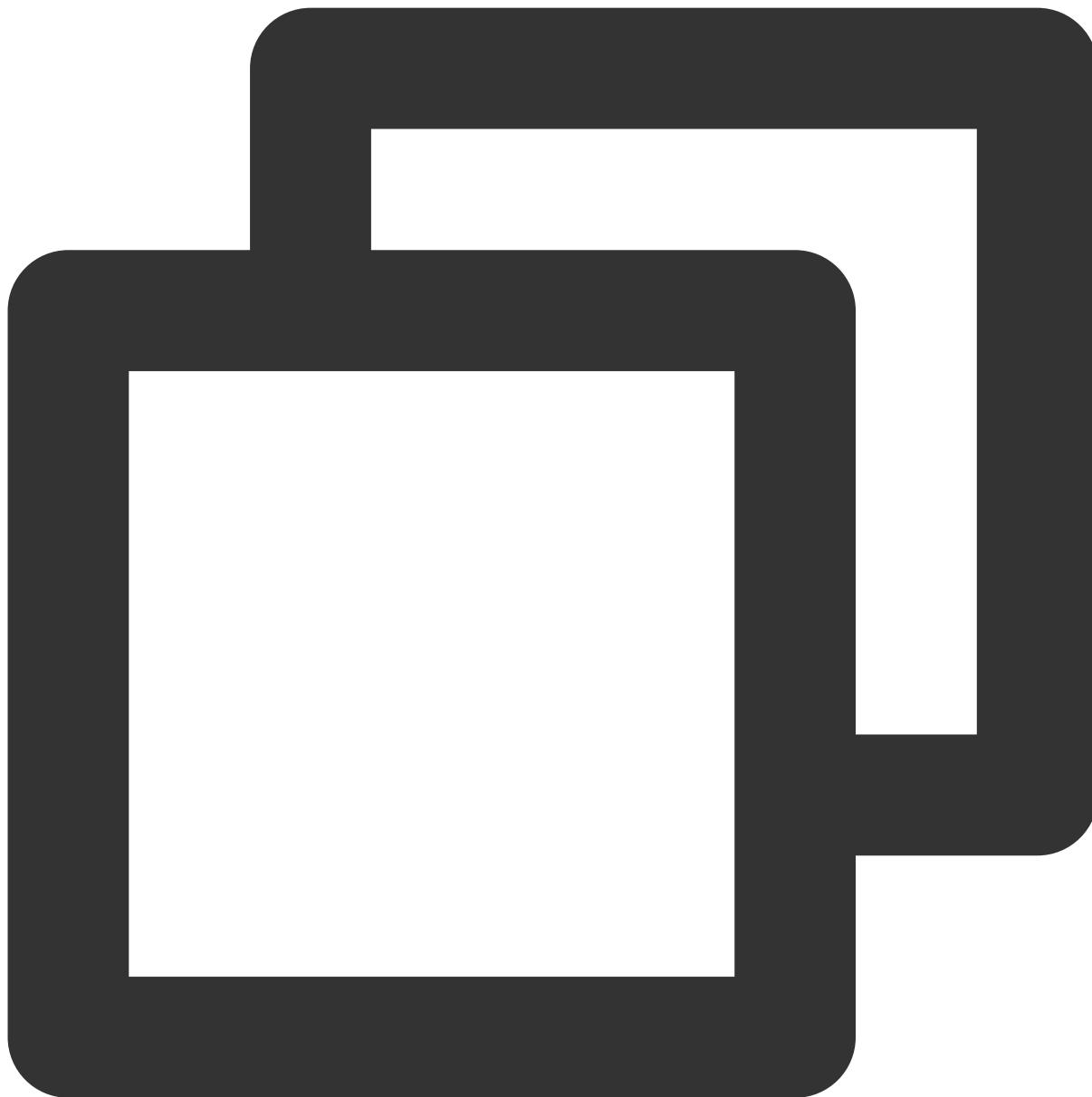


- (TXBeautyManager \*)getBeautyManager;

Return Value	Type	Meaning
manager	TXBeautyManager *	TXBeautyManager Object

getAudioEffectManager

Get Sound Effect Manager Object.



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

Return Value	Type	Meaning
manager	TXAudioEffectManager *	TXAudioEffectManager Object

## getTRTCCloud

Get TRTC Instance Object.





```
- (TRTCCloud *)getTRTCCloud;
```

Return Value	Type	Meaning
manager	TRTCCloud *	TRTCCloud Object

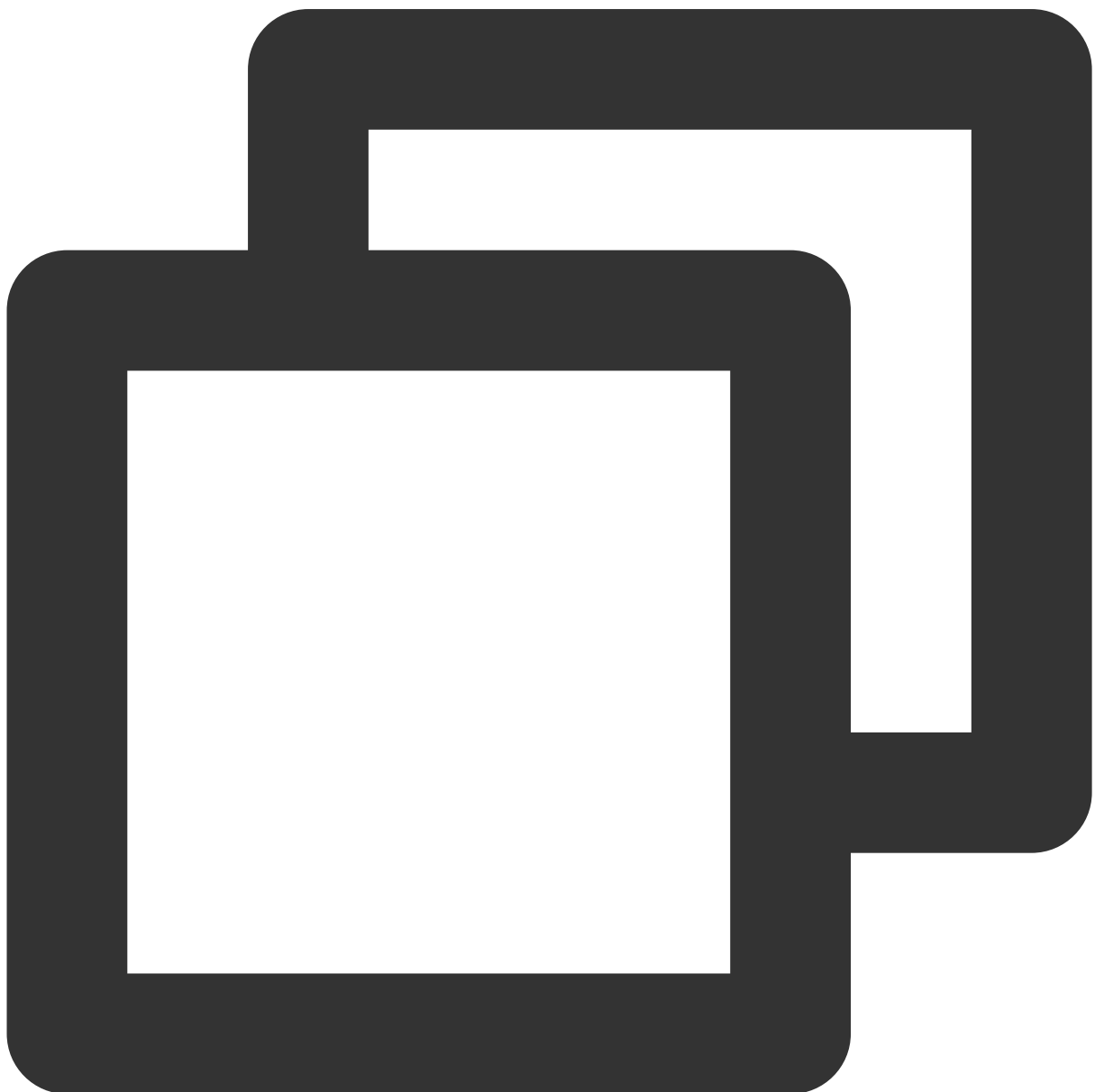
# TUIRoomObserver

Last updated : 2023-12-18 17:52:47

The TUIRoomObserver class is the Callback Event class corresponding to the TUIRoomEngine. You can listen to the Callback Events you are interested in through this interface.

## **onError**

Error Event Callback.



```
- (void)onError:(TUIError)error message:(NSString *)message;
```

The parameters are as follows:

Parameter	Type	Meaning
error	<a href="#">TUIError</a>	Error Code
message	NSString *	Error Message

## onKickedOffline

Other terminals login and get kicked off event.



```
- (void)onKickedOffLine:(NSString *)message;
```

The parameters are as follows:

Parameter	Type	Meaning
message	NSString *	Kicked out description

### onUserSigExpired

User credential timeout event.



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

### **onRoomNameChanged**

Room name change event.

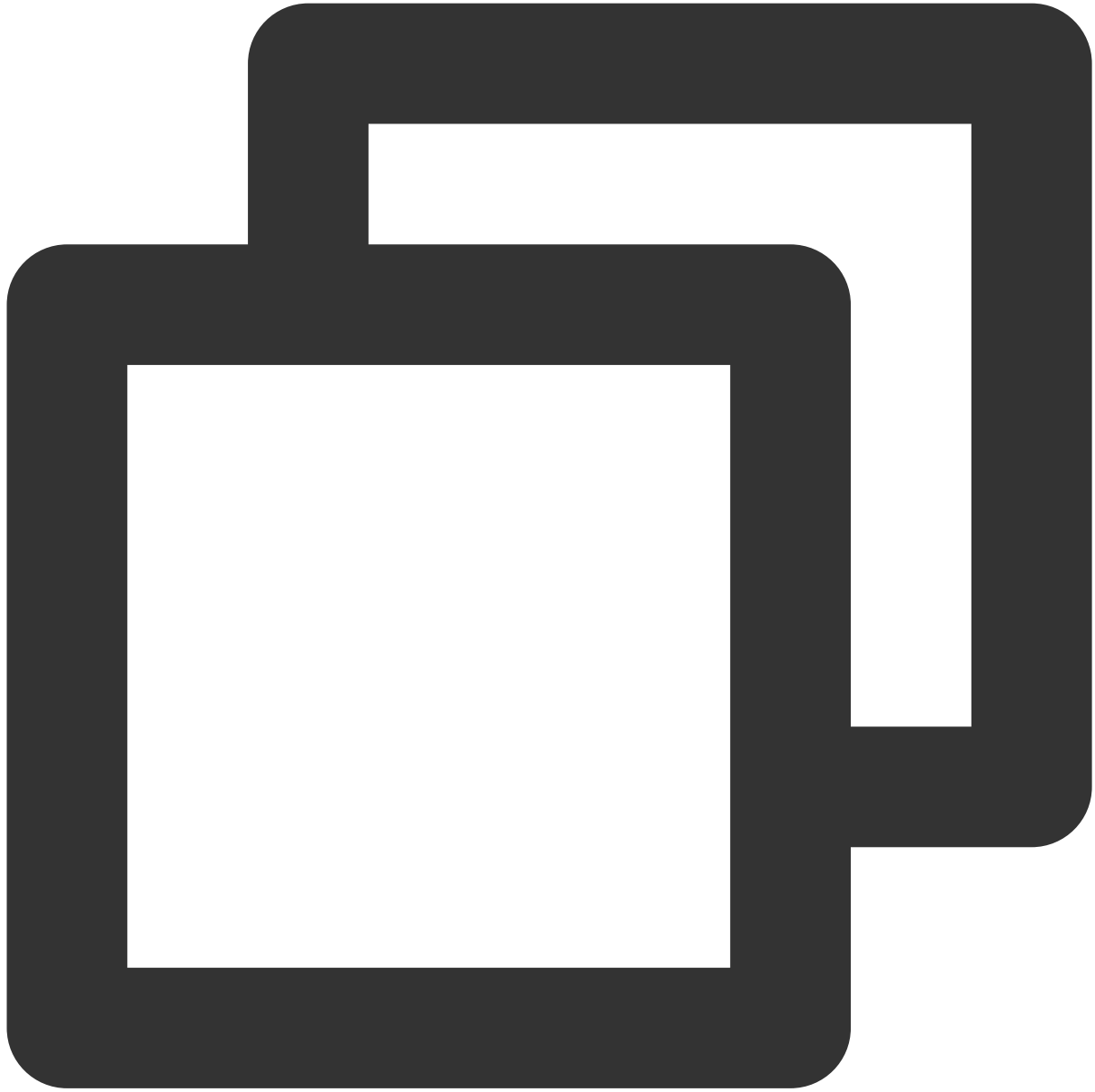


```
- (void)onRoomNameChanged:(NSString *)roomId roomName:(NSString *)roomName;
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
roomName	NSString *	Room Name

### onAllUserMicrophoneDisableChanged

Inside the room, all users' mic is disabled event.

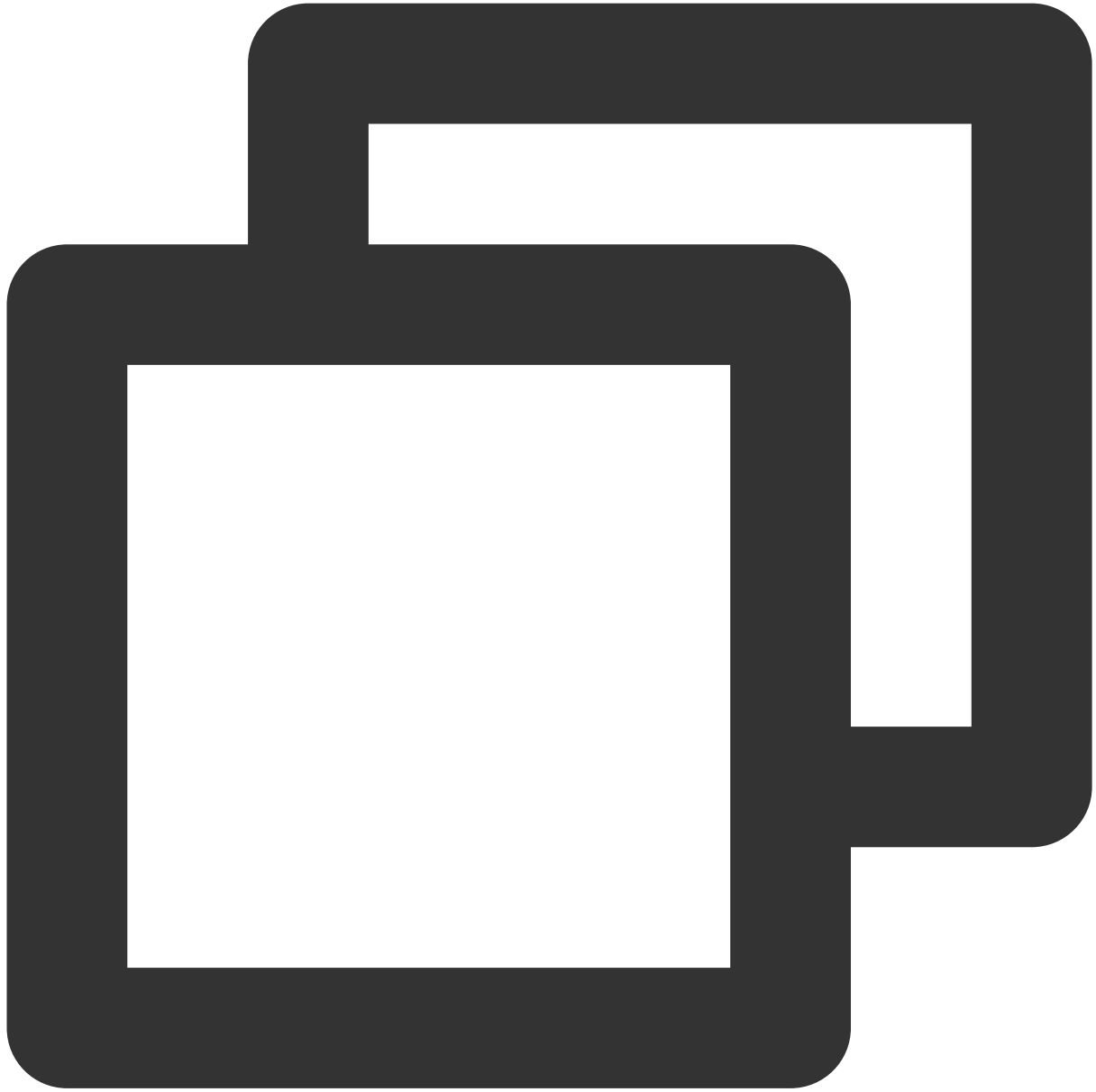


```
- (void)onAllUserMicrophoneDisableChanged:(NSString *)roomId isDisable:(BOOL)isDisa
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
isDisable	BOOL	Whether it is disabled

### onAllUserCameraDisableChanged

All users' Camera in the Room are disabled event.



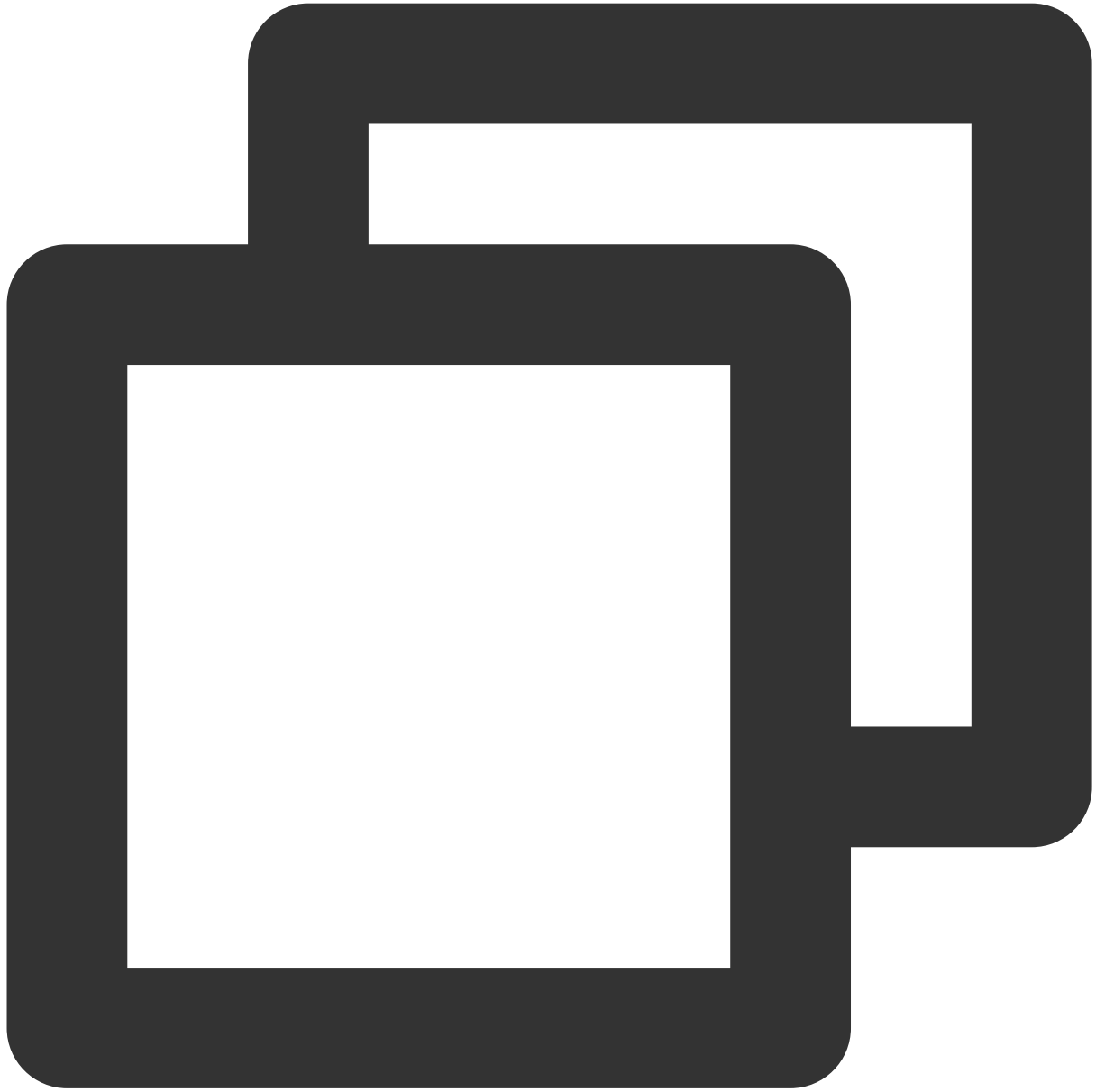
```
- (void)onAllUserCameraDisableChanged:(NSString *)roomId isDisable:(BOOL)isDisable;
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
isDisable	BOOL	Whether it is disabled

### onSendMessageForAllUserDisableChanged



Inside the room, all users' text message sending is disabled event.



```
- (void)onSendMessageForAllUserDisableChanged:(NSString *)roomId isDisable:(BOOL)is
```

### **onKickedOutOfRoom**

Kicked out of the room event.



```
- (void)onKickedOutOfRoom:(NSString *)roomId message:(NSString *)message;
```

The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID
message	NSString *	Description of being kicked out

## onRoomDismissed

Room dissolution event.



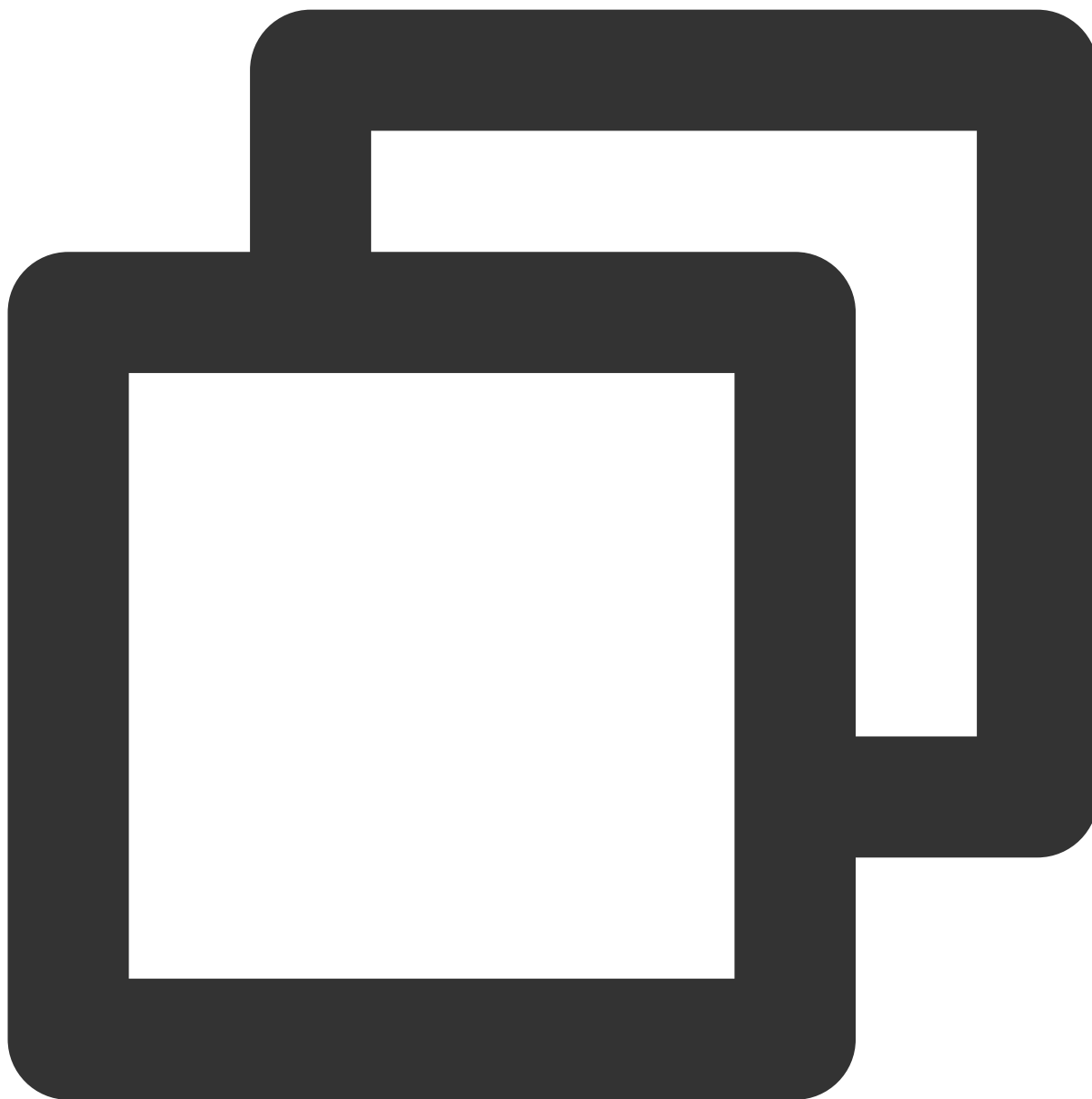
```
- (void)onRoomDismissed:(NSString *)roomId;
```

The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID

**onRoomSpeechModeChanged**

Mic control mode changes in the room.

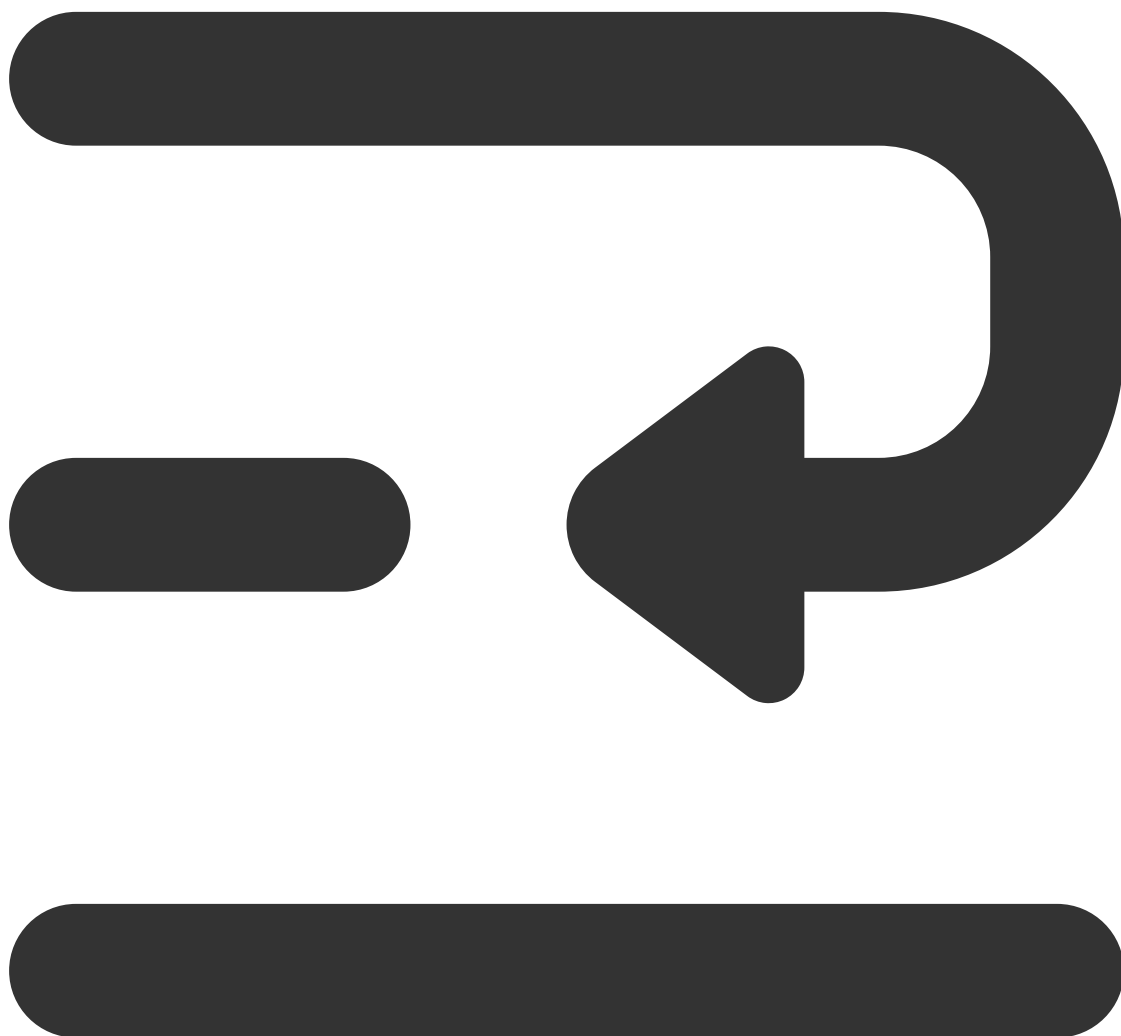


```
- (void)onRoomSpeechModeChanged:(NSString *)roomId speechMode:(TUISpeechMode)mode;
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
mode	<a href="#">TUISpeechMode</a>	Mic control mode

## onRemoteUserEnterRoom

Remote user enters the room event.





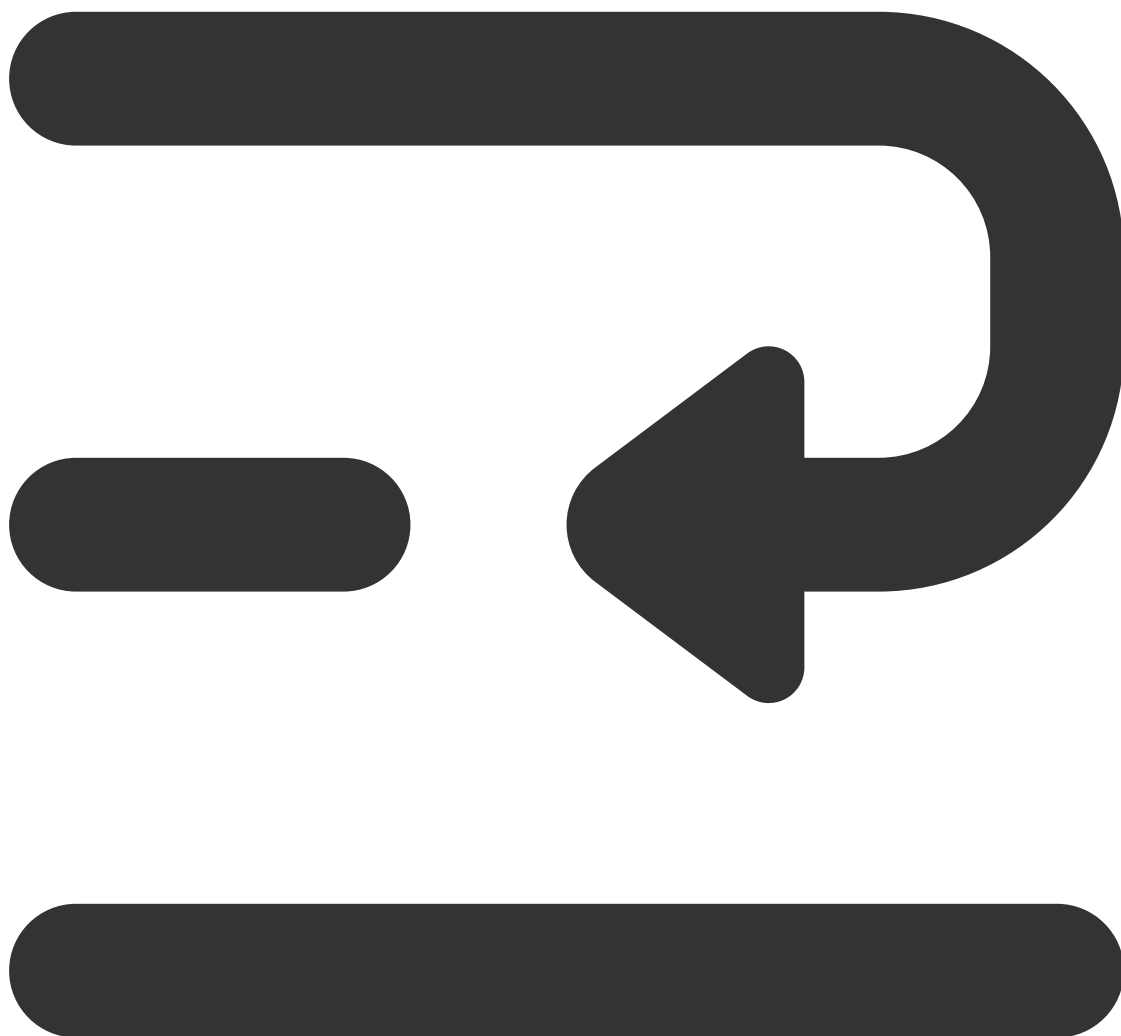
```
- (void)onRemoteUserEnterRoom:(NSString *)roomId userInfo:(TUIUserInfo *)userInfo;
```

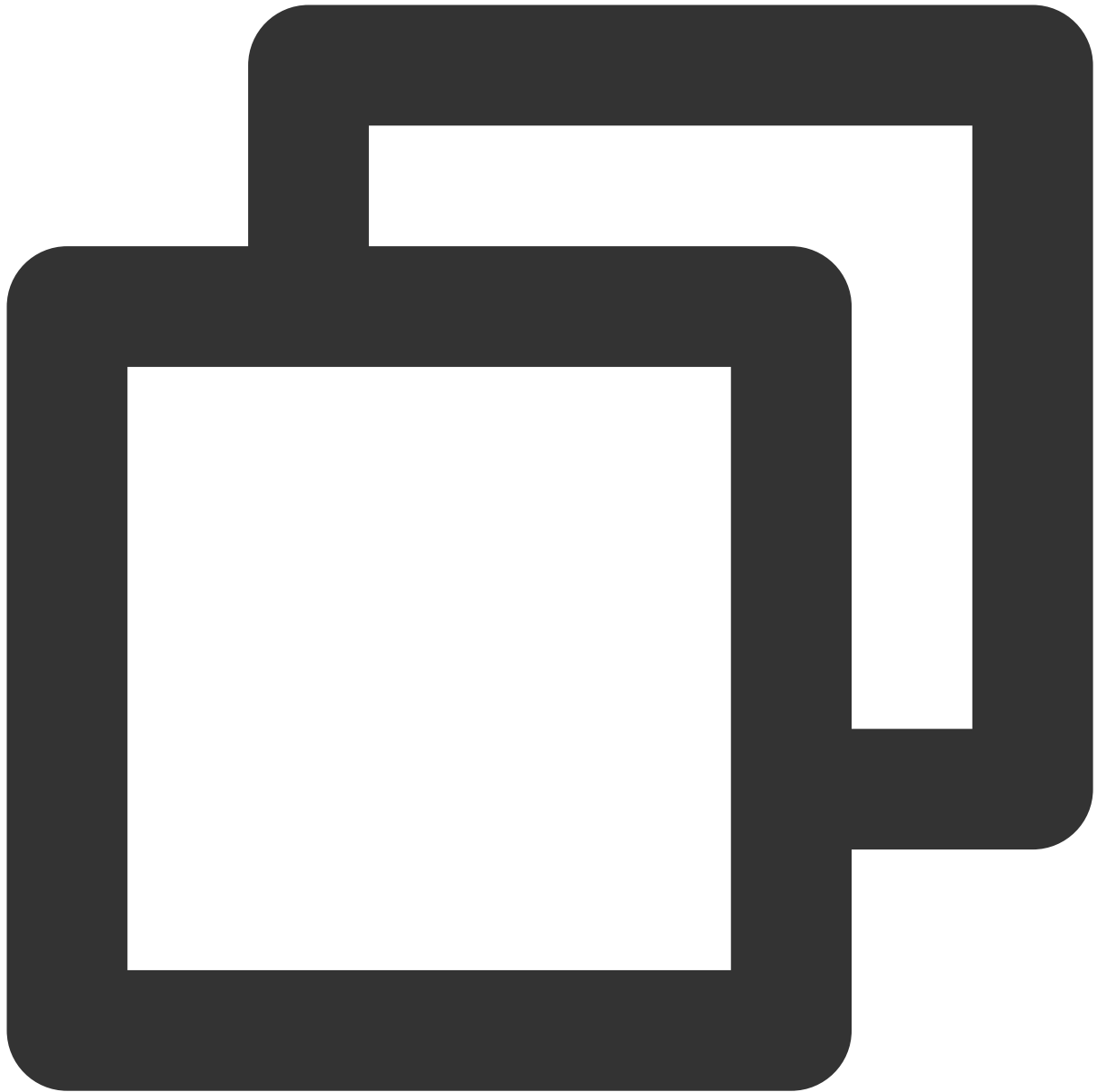
The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID
userInfo	<a href="#">TUIUserInfo</a> *	User information

## onRemoteUserLeaveRoom

Remote user leaves the room event.





```
- (void)onRemoteUserLeaveRoom:(NSString *)roomId userInfo:(TUIUserInfo *)userInfo;
```

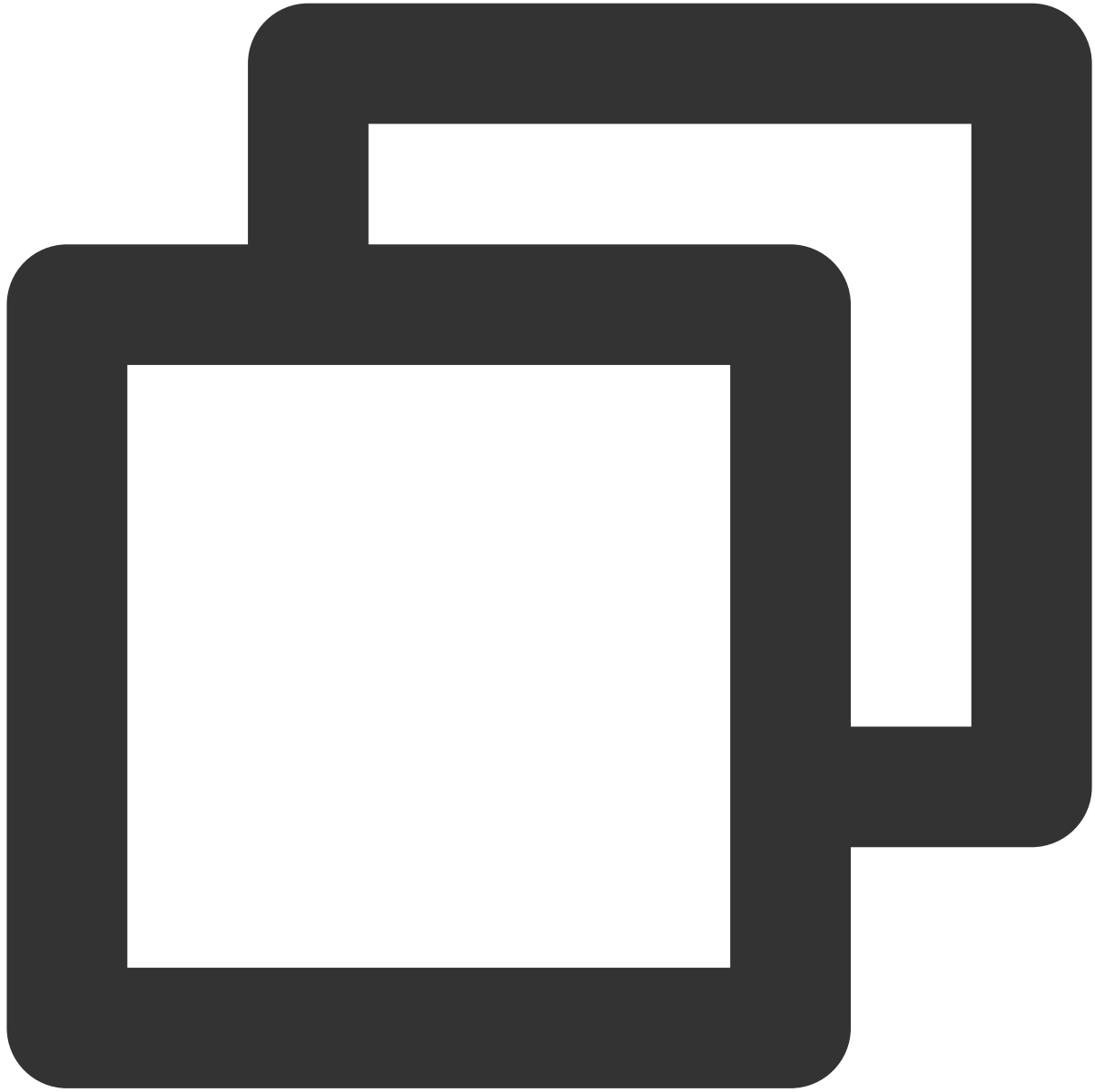
The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID
userInfo	<a href="#">TUIUserInfo</a> *	User information

## onUserRoleChanged



User role changes event.



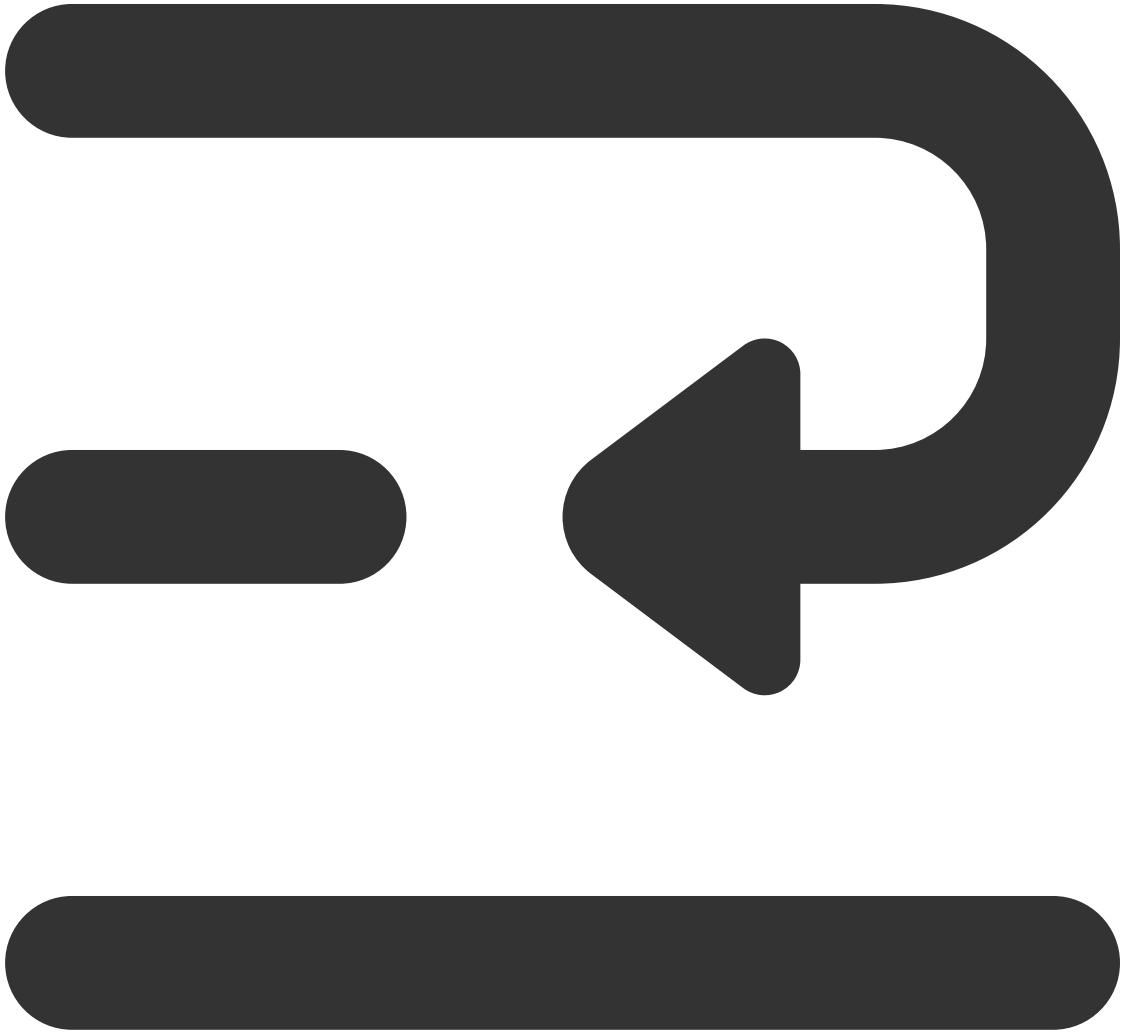
```
- (void)onUserRoleChanged:(NSString *)userId userRole:(TUIRole)userRole;
```

The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID
userRole	<a href="#">TUIRole</a>	User Role

## onUserVideoStateChanged

User Video status changes event.





```
- (void)onUserVideoStateChanged:(NSString *)userId
    streamType:(TUIVideoStreamType)streamType
    hasVideo:(BOOL)hasVideo
    reason:(TUIChangeReason)reason;
```

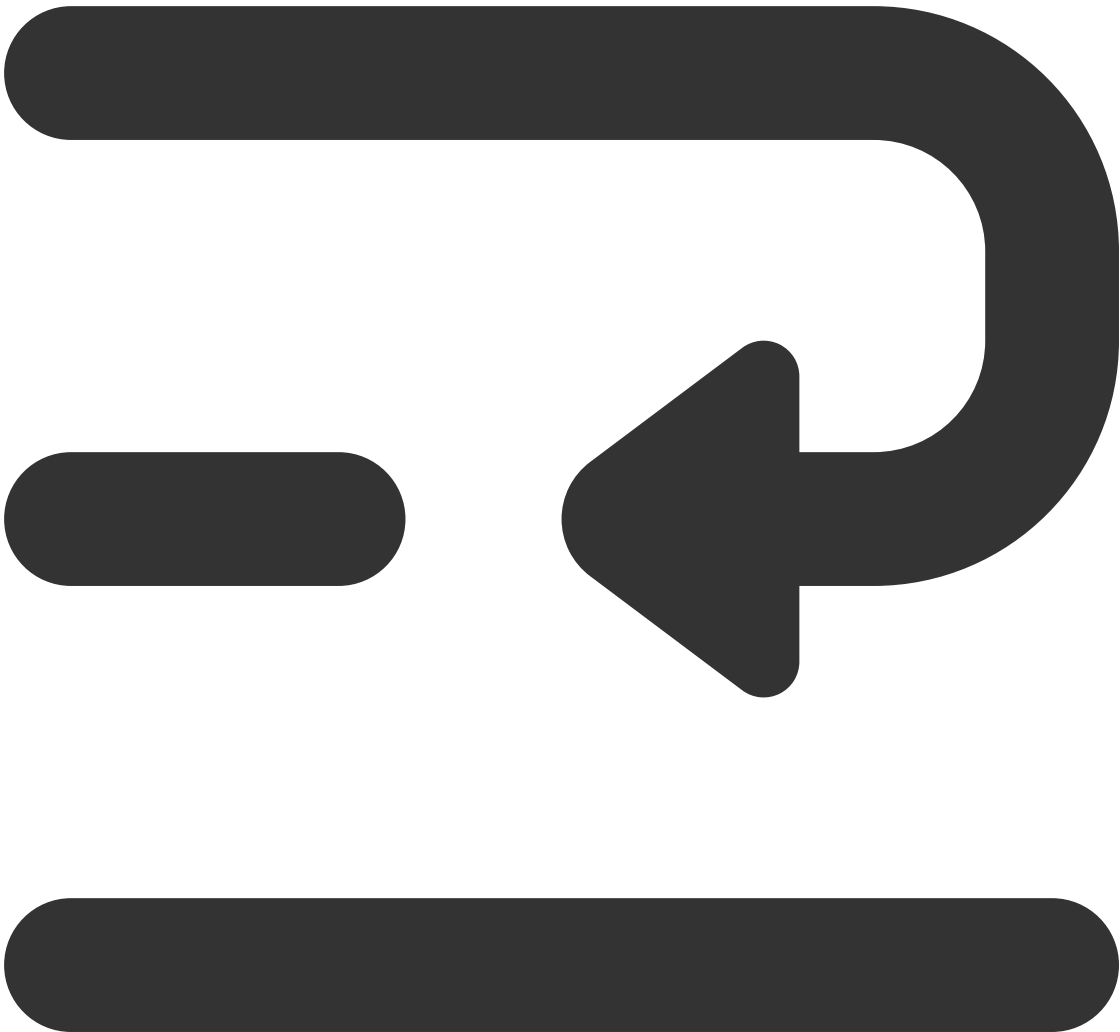
The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID

streamType	<a href="#">TUIVideoStreamType</a>	Streams type
hasVideo	BOOL	Whether there are streams
reason	<a href="#">TUIChangeReason</a>	Reason for streams change

### onUserAudioStateChanged

User Audio status changes event.





```
- (void)onUserAudioStateChanged:(NSString *)userId
    hasAudio:(BOOL)hasAudio
    reason:(TUIChangeReason) reason;
```

The parameters are as follows:

Parameter	Type	Meaning
userId	NSString *	User ID
hasAudio	BOOL	Whether there are Audio streams

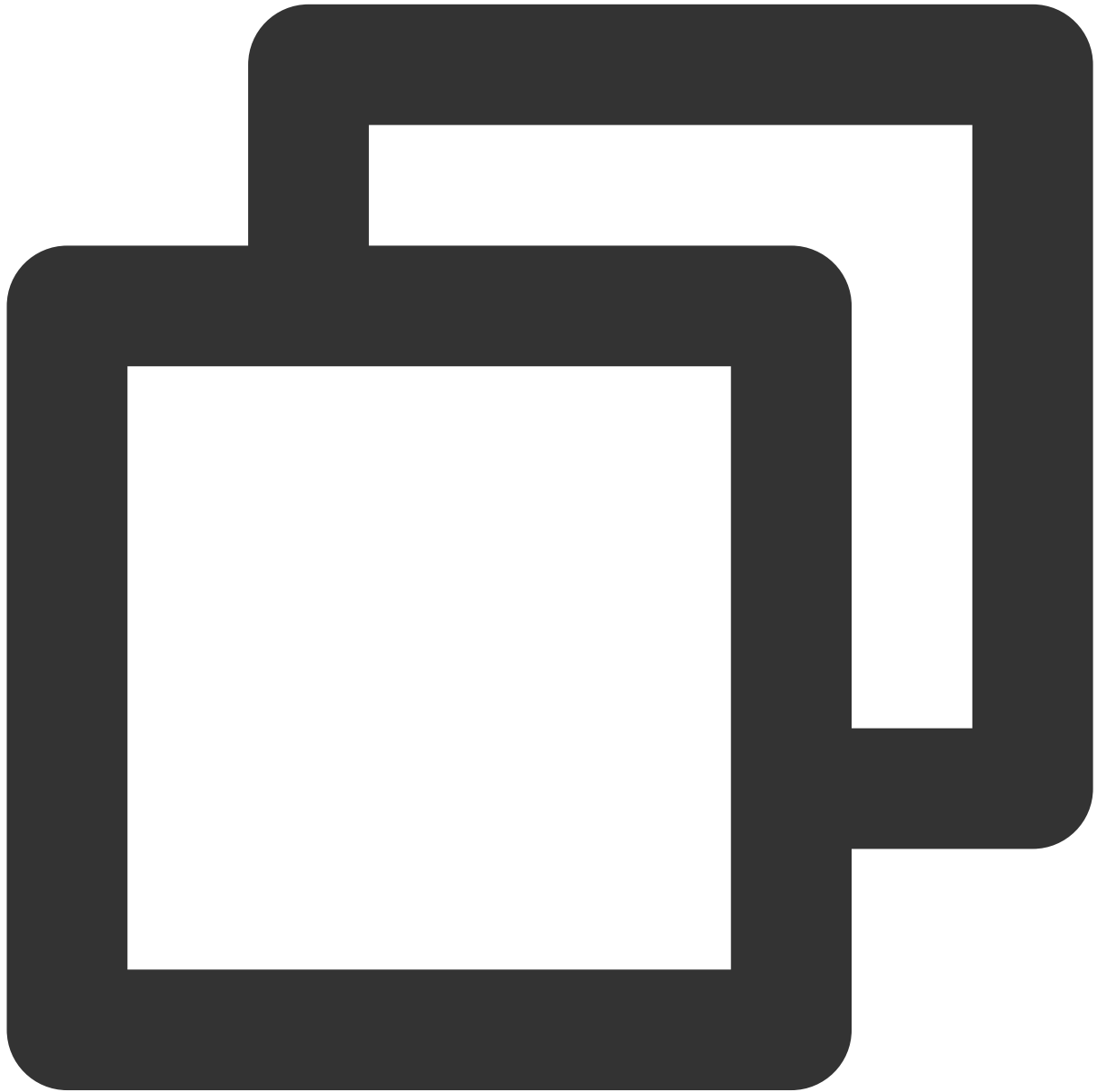
reason

TUIChangeReason

Reason for Audio streams change

## onUserScreenCaptureStopped

Screen Sharing stopped Callback event.



```
- (void)onUserScreenCaptureStopped: (NSInteger) reason;
```

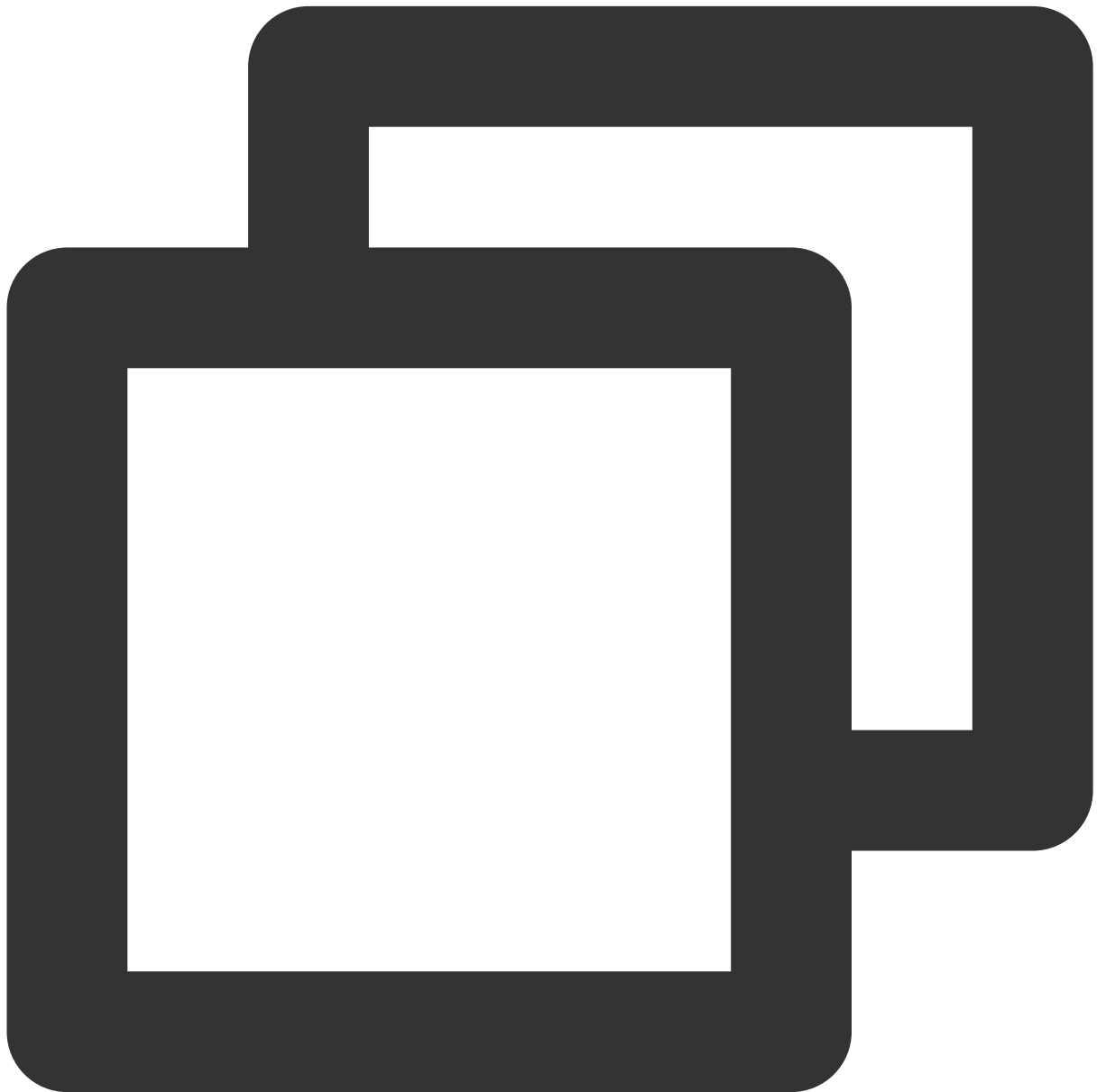
The parameters are as follows:

Parameter	Type	Meaning

reason	NSInteger	Stop reason: 0: User actively stops 1: Screen window closing causes the stop 2: Screen Sharing display screen status change (such as interface being unplugged, Projection mode change, etc.)
--------	-----------	--

## onRoomMaxSeatCountChanged

Maximum number of mic slots changes event in the room (only effective in meeting type rooms).



```
- (void)onRoomMaxSeatCountChanged:(NSString *)roomId maxSeatNumber:(NSInteger)maxSe
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
maxSeatNumber	NSInteger	Maximum number of mic slots in the room

### onUserVoiceVolumeChanged

User volume change event.





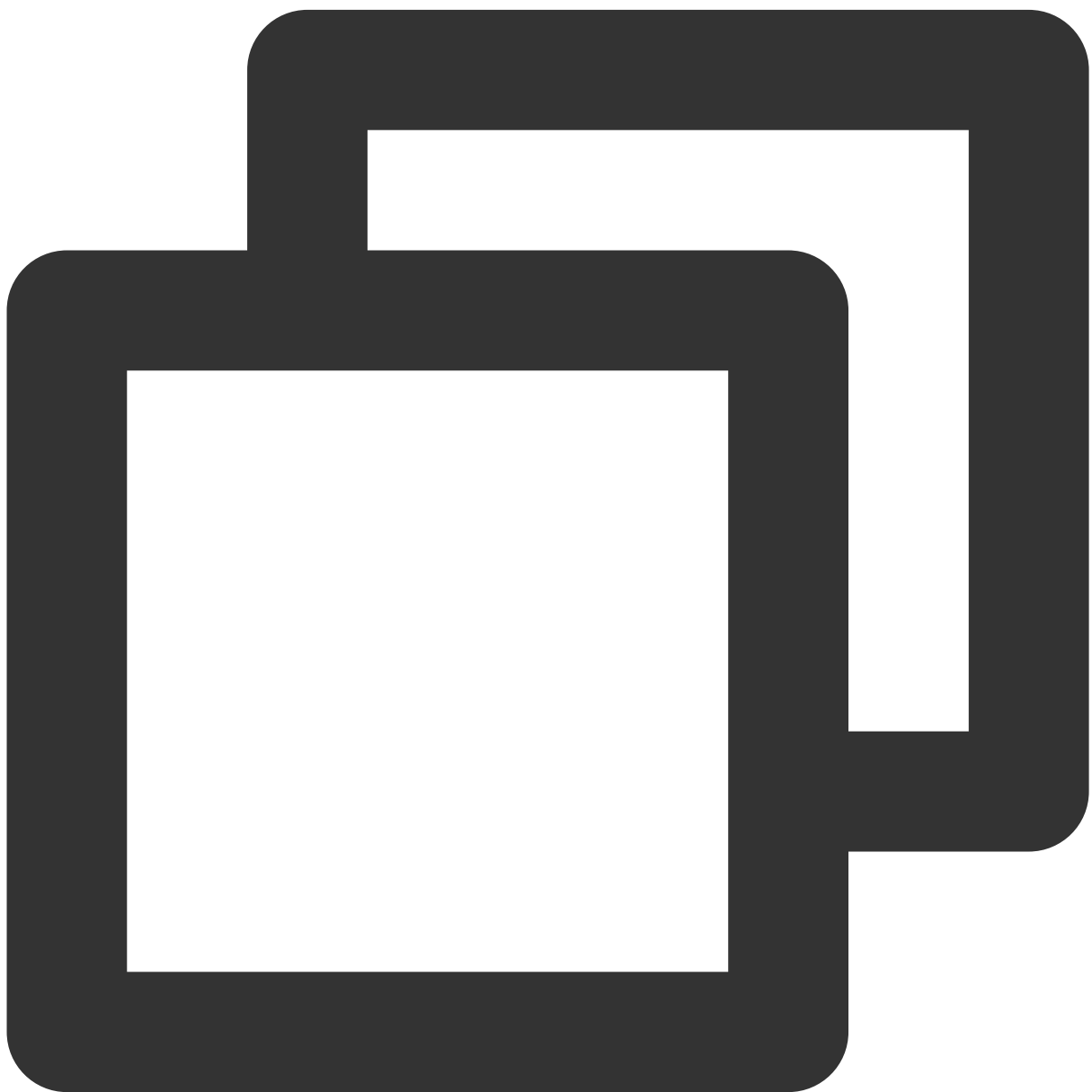
```
- (void)onUserVoiceVolumeChanged:(NSDictionary<NSString *, NSNumber *> *)volumeMap;
```

The parameters are as follows:

Parameter	Type	Meaning
volumeMap	NSDictionary<NSString *, NSNumber *> *	Volume map

## onSendMessageForUserDisableChanged

User text message sending ability changes event.

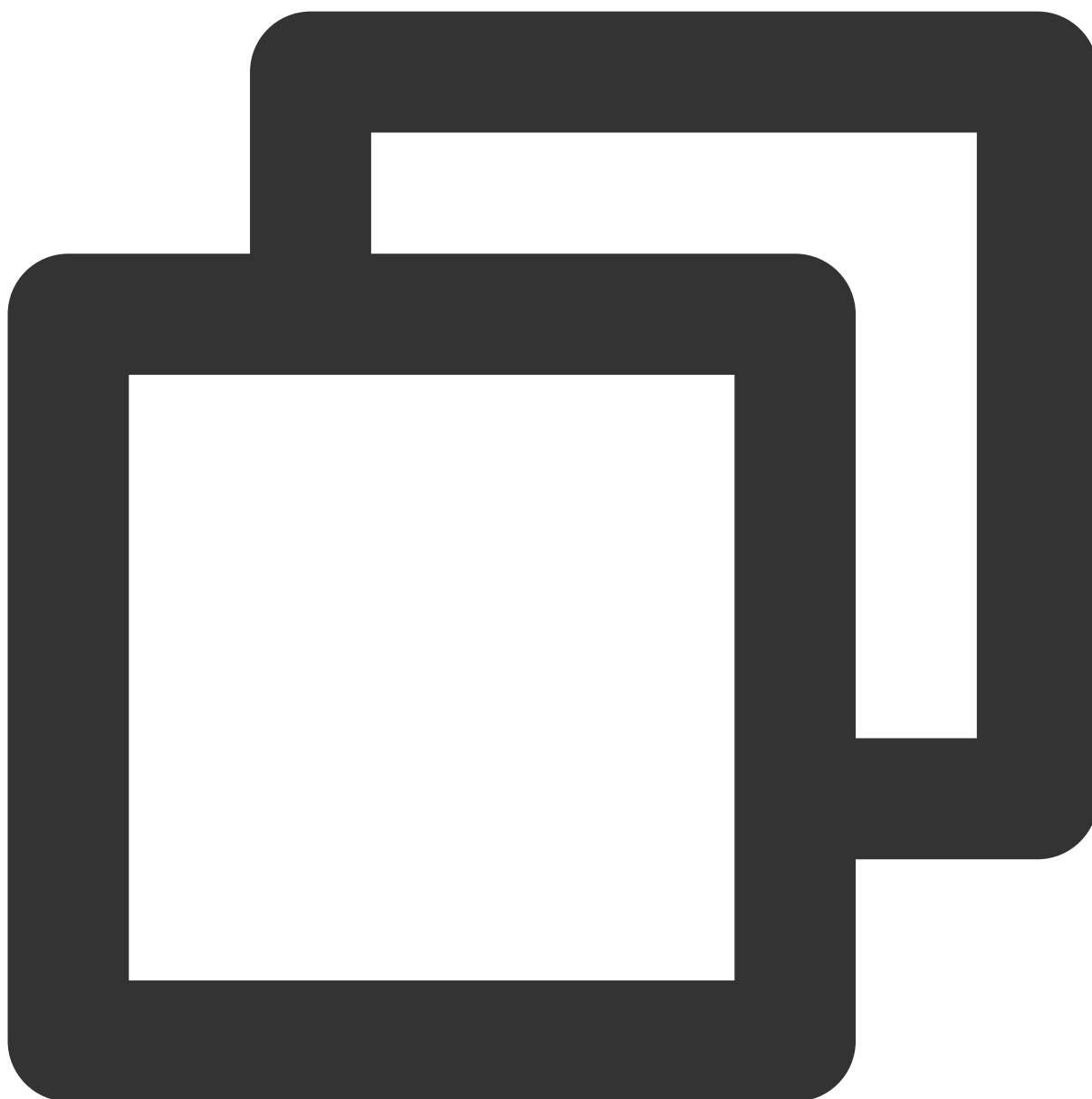


```
- (void)onSendMessageForUserDisableChanged:(NSString *)roomId
    userId:(NSString *)userId
    isDisable:(BOOL)muted;
```

Parameter	Type	Meaning
roomId	NSString *	Room ID
userId	NSString *	User ID
muted	BOOL	Whether it is prohibited to send text messages.

## onUserNetworkQualityChanged

User network status change event.



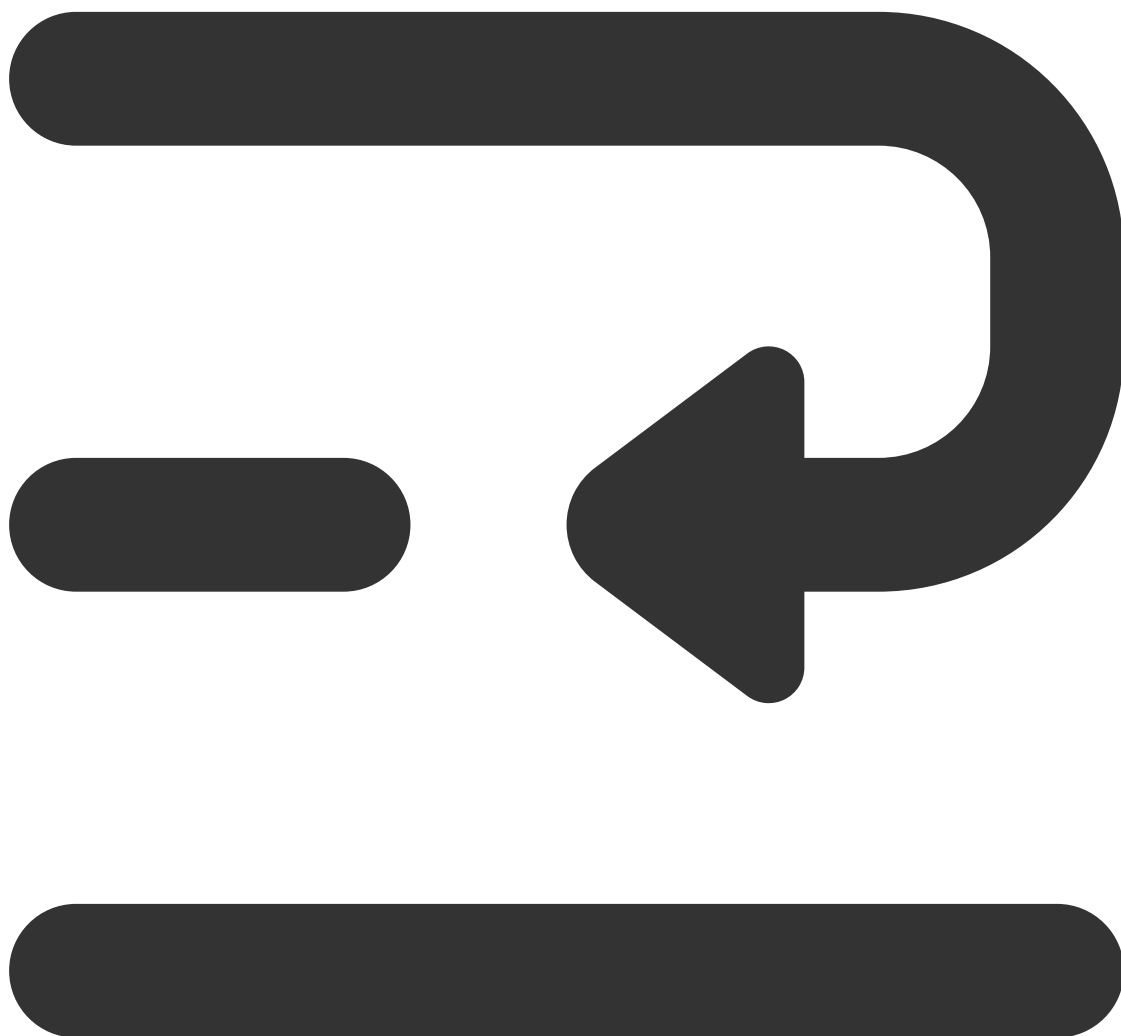
```
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkInfo *> *)networkList;
```

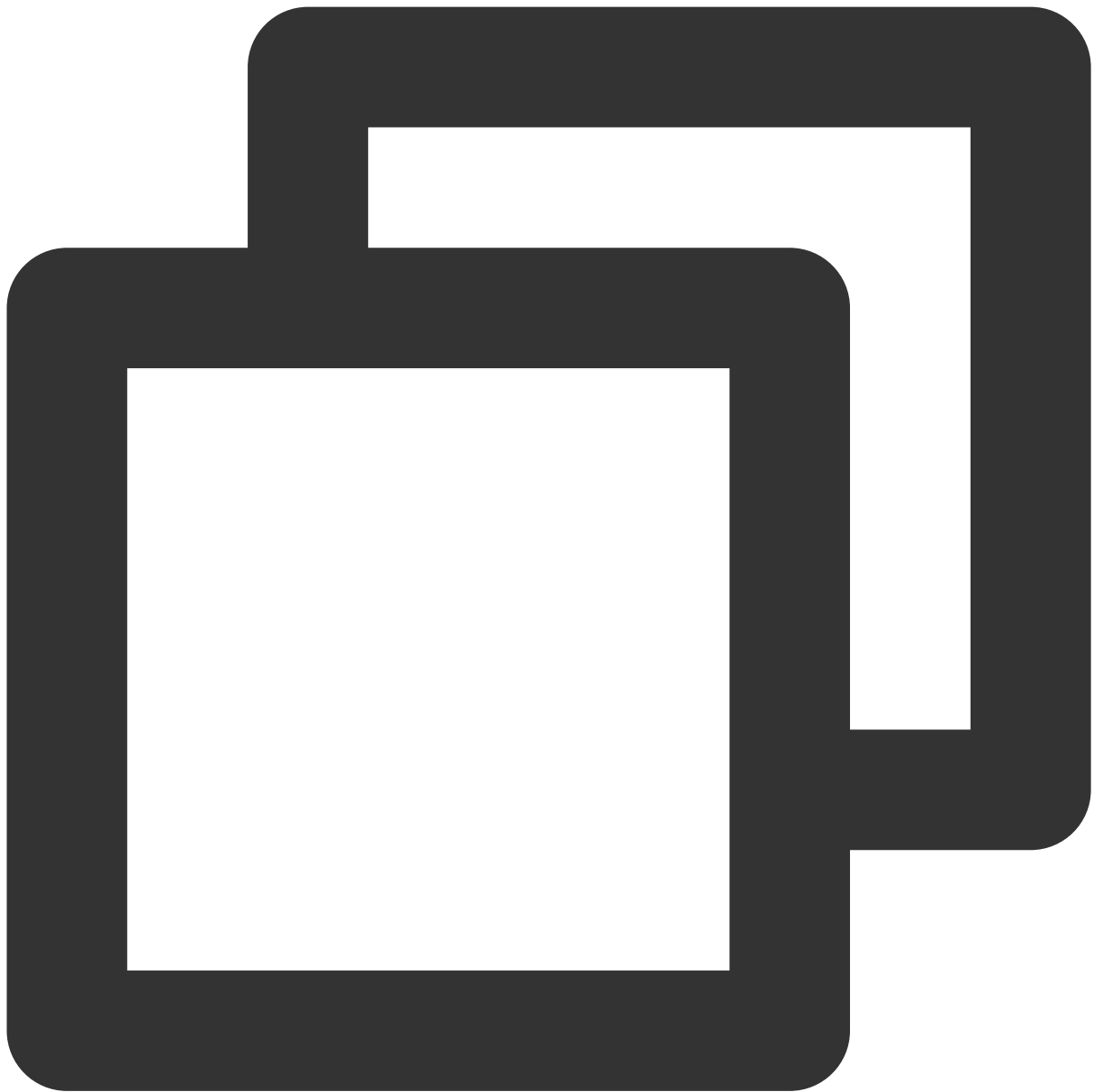
The parameters are as follows:

Parameter	Type	Meaning
networkList	NSArray<TUINetworkInfo>*	Network status array, you can refer to <code>TUINetworkInfo</code> object

## onSeatListChanged

Mic slot list changes event.





```
- (void)onSeatListChanged:(NSArray<TUISeatInfo *> *)seatList  
    seated:(NSArray<TUISeatInfo *> *)seatedList  
    left:(NSArray<TUISeatInfo *> *)leftList;
```

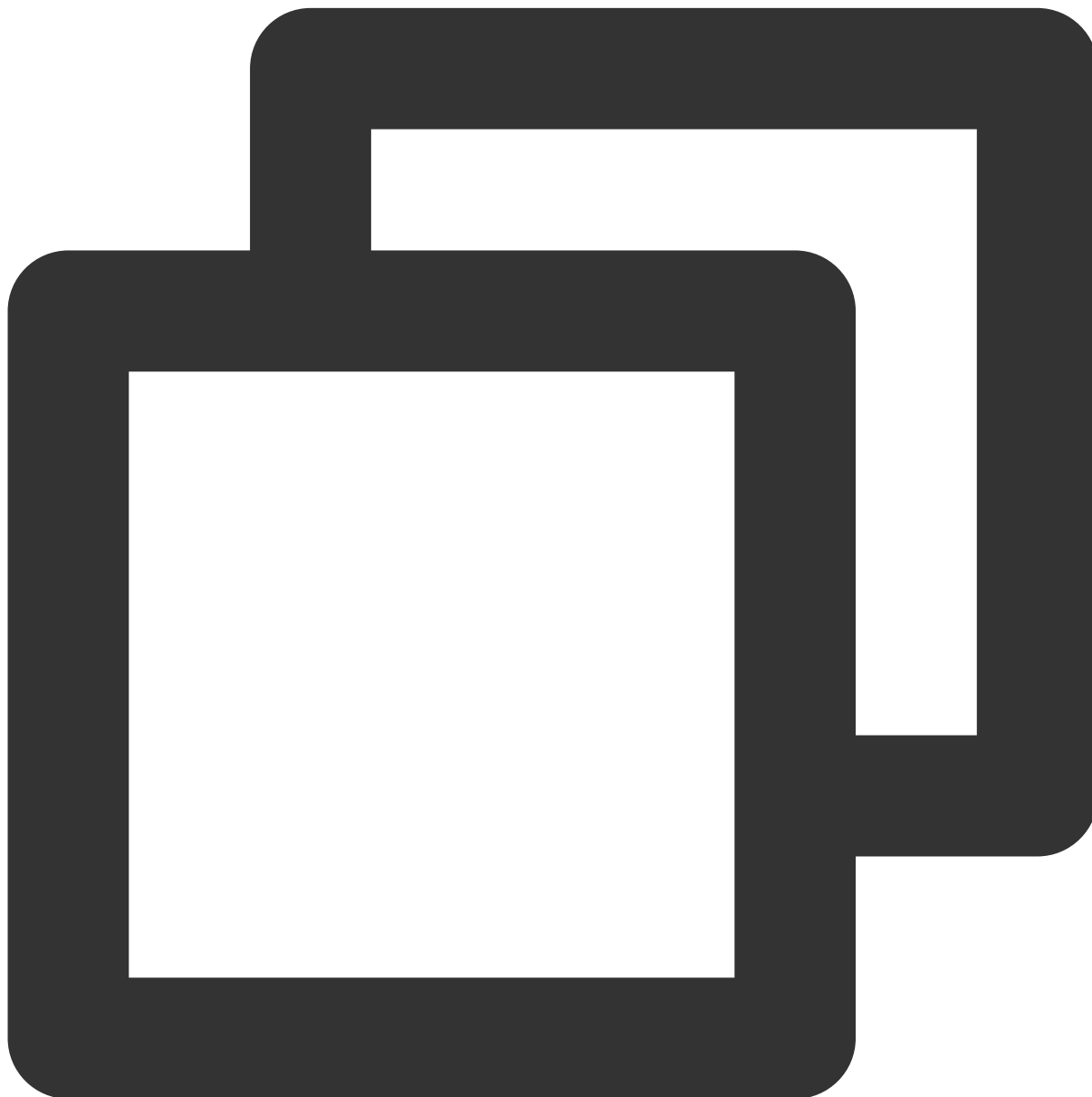
The parameters are as follows:

Parameter	Type	Meaning
seatList	NSArray<TUISeatInfo *> *	The latest user list on the mic, including newly on mic users

seatedList	NSArray<TUISeatInfo *> *	Newly on mic user list
leftList	NSArray<TUISeatInfo *> *	Newly off mic user list

## onKickedOffSeat

Received the event of user being kicked off mic.



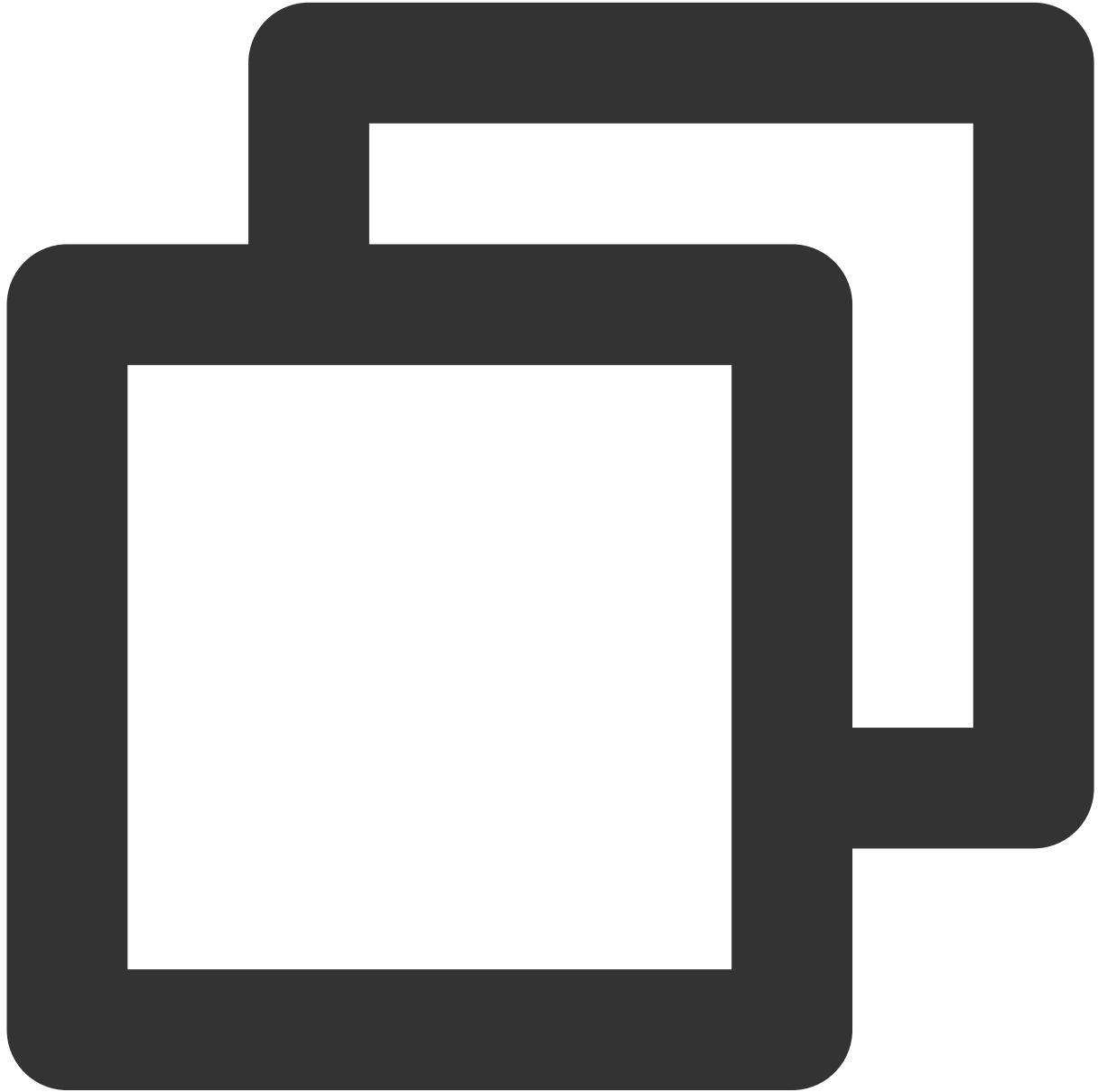
```
- (void)onKickedOffSeat:(NSString *)userId;
```

Parameter	Type	Meaning
-----------	------	---------

userId	NSString *	User ID
--------	------------	---------

## onRequestReceived

Received request message event.



```
- (void)onRequestReceived:(TUIRequest *)request;
```

The parameters are as follows:

Parameter	Type	Meaning
-----------	------	---------

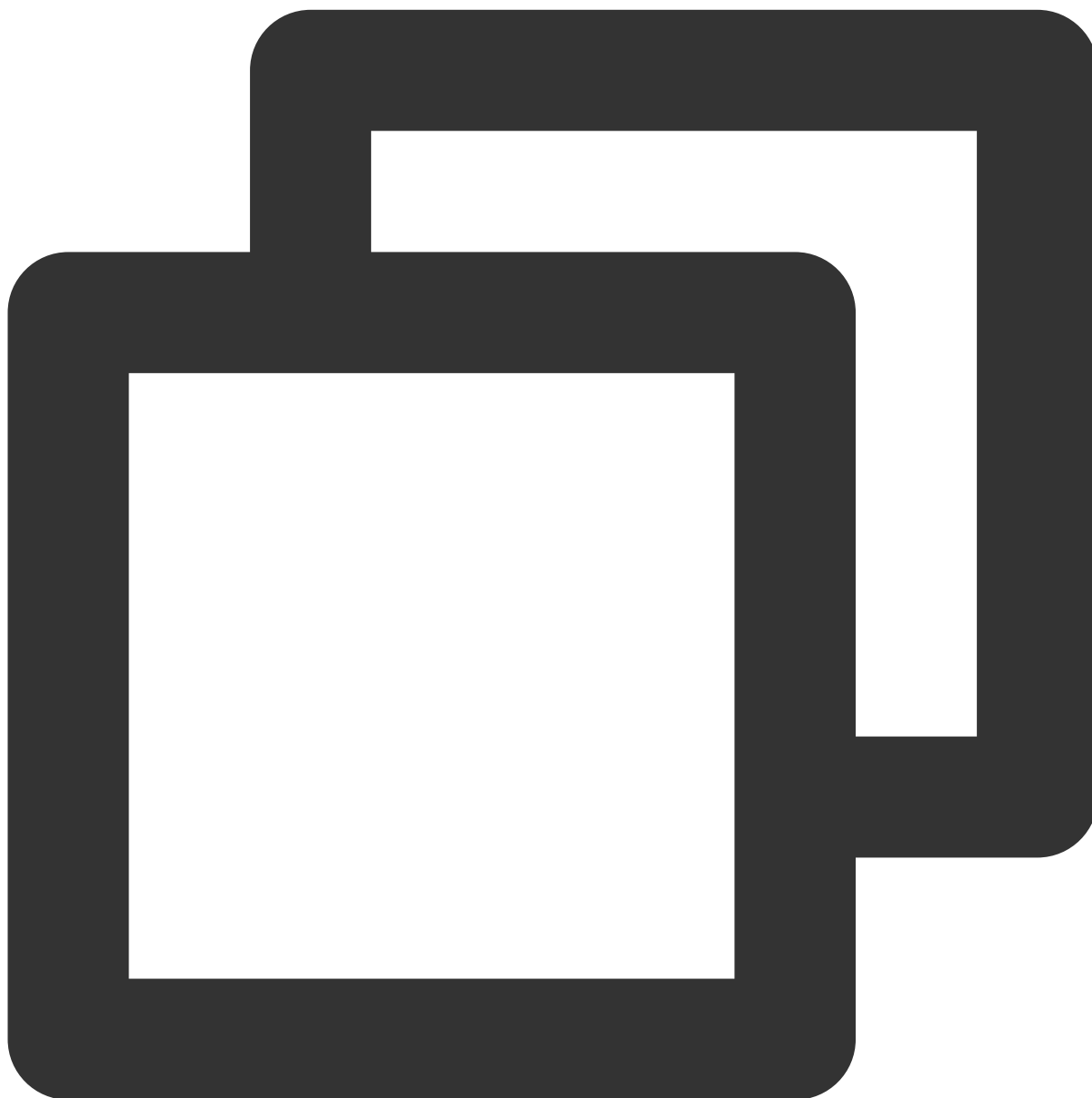
request

TUIRequest \*

Request content

## onRequestCancelled

Received request cancellation event.



```
- (void)onRequestCancelled:(NSString *)requestId;
```

The parameters are as follows:

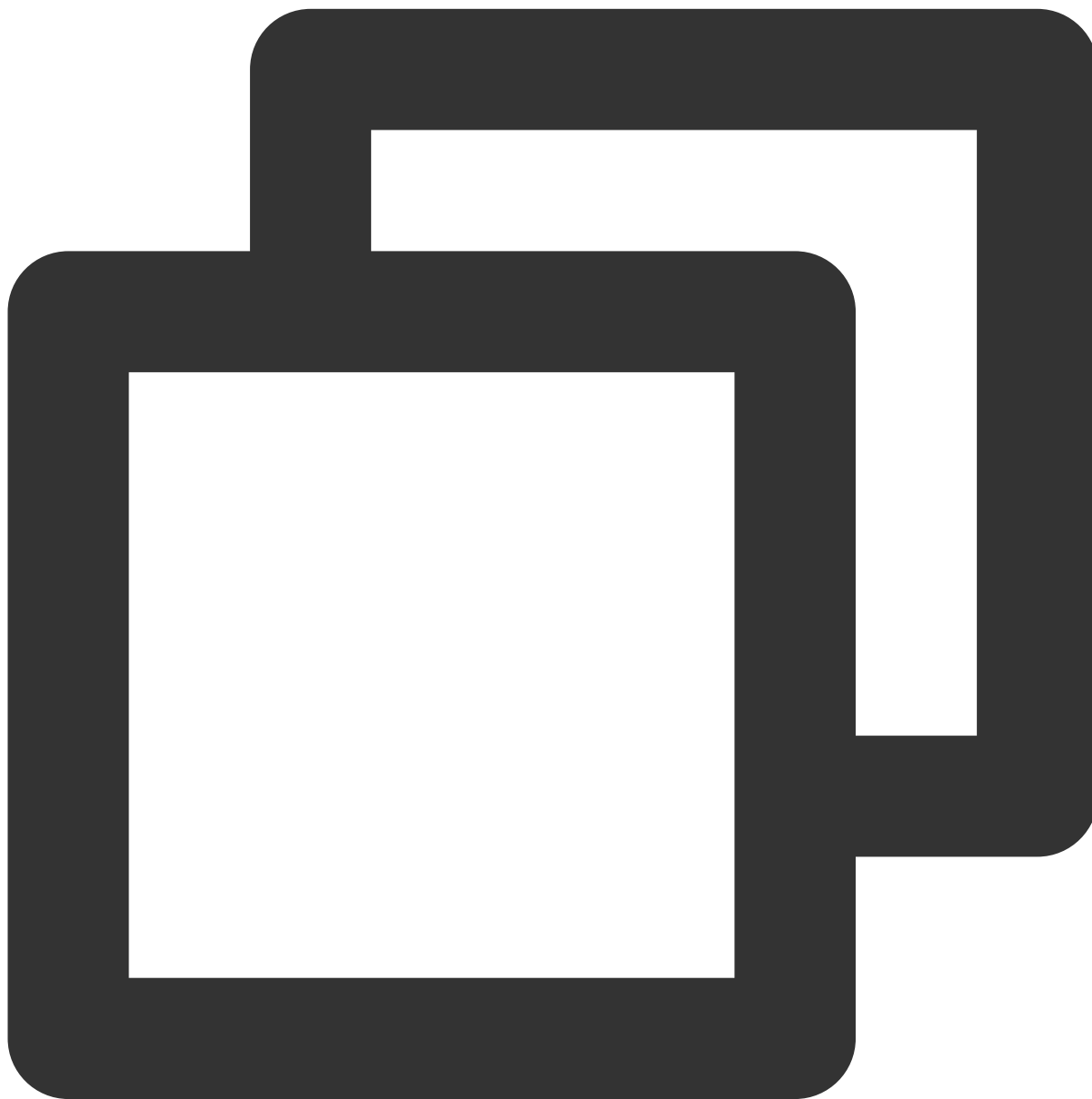
Parameter	Type	Meaning



requestId	NSString *	Request ID
-----------	------------	------------

## onReceiveTextMessage

Received ordinary text message event.



```
- (void)onReceiveTextMessage:(NSString *)roomId  
    message:(TUIMessage *)message;
```

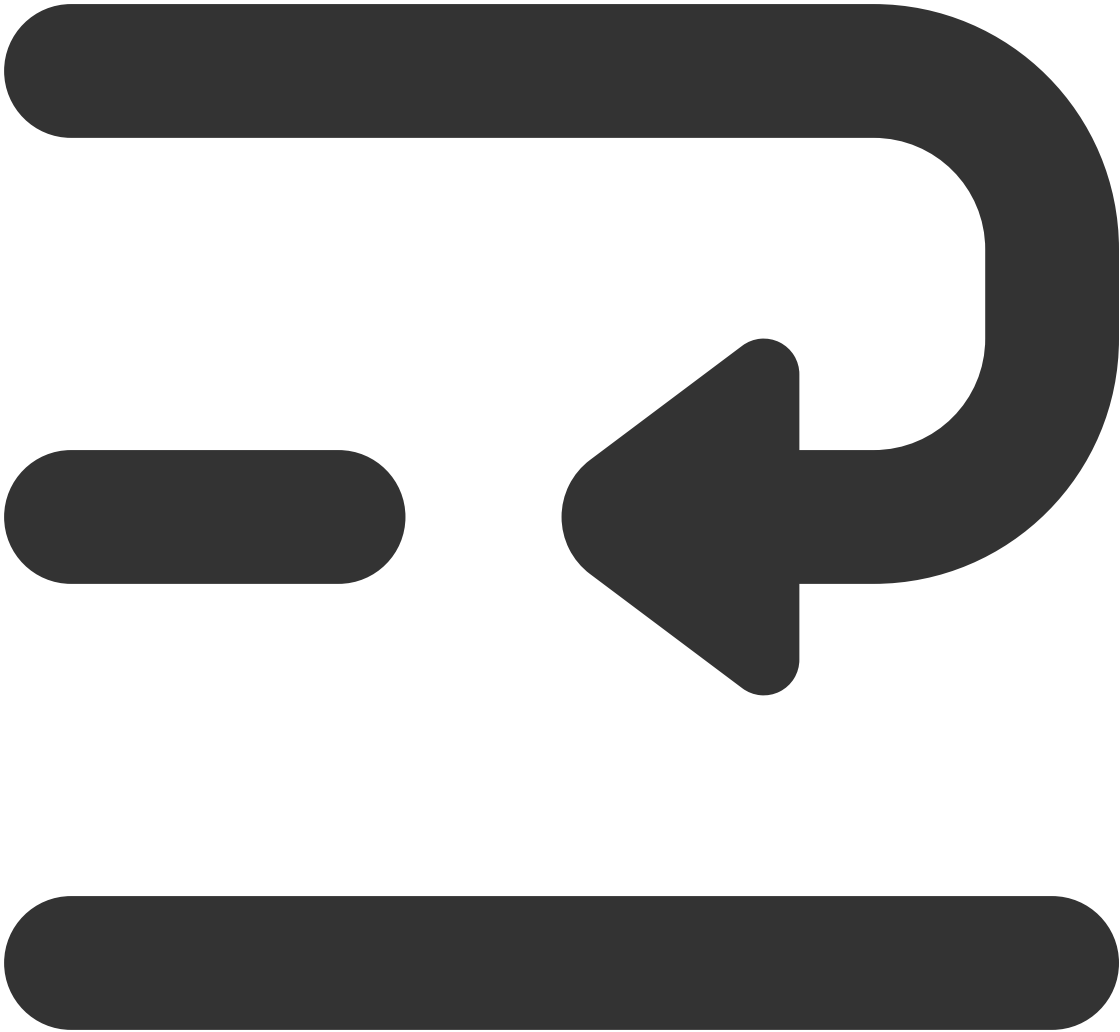
The parameters are as follows:

Parameter	Type	Meaning
-----------	------	---------

roomId	NSString *	Room ID
message	TUIMessage	Message content, detailed definition can refer to <code>TUIRoomDefine.h</code> file in TUIMessage definition

### onReceiveCustomMessage

Received custom message event.





```
- (void)onReceiveCustomMessage:(NSString *)roomId  
    message:(TUIMessage *)message;
```

The parameters are as follows:

Parameter	Type	Meaning
roomId	NSString *	Room ID
message	<a href="#">TUIMessage</a>	Message content



# Type Definition

Last updated : 2023-12-18 18:05:05

## Enumeration Definition

### TUIRoomDefine

Type	Description
<a href="#">TUIRoomType</a>	Room Type
<a href="#">TUISpeechMode</a>	Mic Control Mode
<a href="#">TUIMediaDevice</a>	Room Media Device Type
<a href="#">TUIRole</a>	Room Role Type
<a href="#">TUIVideoQuality</a>	Video Quality
<a href="#">TUIAudioQuality</a>	Audio Quality
<a href="#">TUIVideoStreamType</a>	Video Stream Type
<a href="#">TUIChangeReason</a>	Change Reason (User audio and video status change operation reason: self-modification or modified by room owner/administrator)
<a href="#">TUICaptureSourceType</a>	Screen Sharing Capture Source Type
<a href="#">TUIRequestAction</a>	Request Type

### TUICommonDefine

Type	Description
<a href="#">TUINetworkQuality</a>	Network Quality

## Common Structure

### TUIRoomDefine

Type	Description
<a href="#">TUIRoomInfo</a>	Room data
<a href="#">TUILoginUserInfo</a>	User Login Information

<a href="#">TUIUserInfo</a>	Room User Information
<a href="#">TUISeatInfo</a>	Room Seat Information
<a href="#">TUISeatLockParams</a>	Lock Seat Operation Parameters
<a href="#">TUIUserVoiceVolume</a>	Room User Volume
<a href="#">TUIRequest</a>	Signaling Request
<a href="#">TUIShareTarget</a>	Screen Sharing Capture Source Information

### TUICommonDefine

Type	Description
<a href="#">TUINetworkInfo</a>	Network Quality Information
<a href="#">TUIMessage</a>	Message

### TUIRoomType

Room Type

Enumeration	Value	Description
<a href="#">TUIRoomTypeConference</a>	1	Conference Type Room, suitable for conference and education scenarios, this room can enable free speech, apply for speech, go live and other modes.
<a href="#">TUIRoomTypeLivingRoom</a>	2	Live Type Room, suitable for live broadcast scenarios, this room can enable free speech, mic control mode, and the seats in this room are numbered.

### TUISpeechMode

Mic Control Mode

Enumeration	Value	Description
<a href="#">TUISpeechModeFreeToSpeak</a>	1	Free speech mode.
<a href="#">TUISpeechModeApplyToSpeak</a>	2	Apply to speak mode. (Only effective in conference type room)
<a href="#">TUISpeechModeApplySpeakAfterTakingSeat</a>	3	Go Live mode.

## TUIMediaDevice

### Room Media Device Type

Enumeration	Value	Description
TUIMediaDeviceMicrophone	1	Mic
TUIMediaDeviceCamera	2	Camera
TUIMediaDeviceApplyScreenSharing	3	Screen Sharing

## TUIRole

### Room Role Types

Enumeration	Value	Description
TUIRoleRoomOwner	0	Room Owner, generally refers to the creator of the room, the highest authority holder in the room
TUIRoleAdministrator	1	Room Administrator
TUIRoleGeneralUser	2	General Member in the room

## TUIVideoQuality

### Video Quality

Enumeration	Value	Description
TUIVideoQuality360P	1	Low-quality 360P
TUIVideoQuality540P	2	Standard Definition 540P
TUIVideoQuality720P	3	High Definition 720P
TUIVideoQuality1080P	4	Ultra-clear 1080P

## TUIAudioQuality

### Audio Quality

Enumeration	Value	Description
TUIAudioQualitySpeech	0	Speech Mode

TUIAudioQualityDefault	1	Default Mode
TUIAudioQualityMusic	2	Music Mode

## TUIVideoStreamType

Video Stream Type

Enumeration	Value	Description
TUIVideoStreamTypeCameraStream	0	High-quality Camera Video Stream
TUIVideoStreamTypeScreenStream	1	Screen Sharing Video Stream
TUIVideoStreamTypeCameraStreamLow	2	Low-quality Camera Video Stream

## TUIChangeReason

Change Reason (User audio and video status change operation reason: self-modification or modification by room owner/administrator)

Enumeration	Value	Description
TUIChangeReasonBySelf	0	Self-operation
TUIChangeReasonByAdmin	1	Room Owner or Administrator Operation

## TUICaptureSourceType

Screen Sharing Capture Source Type

Enumeration	Value	Description
TUICaptureSourceTypeUnknown	-1	Undefined
TUICaptureSourceTypeWindow	0	Window
TUICaptureSourceTypeScreen	1	Screen

## TUIRequestAction

Request Type

Enumeration	Value	Description
TUIRequestActionInvalidAction	0	Invalid Request
TUIRequestActionOpenRemoteCamera	1	Request Remote User to



		Open Camera
TUIRequestActionOpenRemoteMicrophone	2	Request Remote User to Open Microphone
TUIRequestActionConnectOtherRoom	3	Request to Connect to Other Room
TUIRequestActionTakeSeat	4	Request to Go Live
TUIRequestActionRemoteUserOnSeat	5	Request Remote User to Go Live
TUIRequestActionApplyToAdminToOpenLocalCamera	6	Request to Admin to Open Local Camera
TUIRequestActionApplyToAdminToOpenLocalMicrophone	7	Request to Admin to Open Local Microphone

## TUINetworkQuality

### Network Quality

Enumeration	Value	Description
TUINetworkQualityUnknown	0	Undefined
TUINetworkQualityExcellent	1	Current Network is Excellent
TUINetworkQualityGood	2	Current Network is Good
TUINetworkQualityPoor	3	Current Network is Average
TUINetworkQualityBad	4	Current Network is Poor
TUINetworkQualityVeryBad	5	Current Network is Very Poor
TUINetworkQualityDown	6	Current Network Does Not Meet TRTC's Minimum Requirements

## TUIRoomInfo

### Room Information

Field	Type	Description
roomId	NSString *	Room ID
roomType	<a href="#">TUIRoomType</a>	Room Type

ownerId	NSString *	Host ID, default is the room creator (read-only)
name	NSString *	Room Name, default is the room ID
speechMode	TUISpeechMode	Mic Control Mode
createTime	NSUInteger	Room Creation Time (read-only)
memberCount	NSInteger	Number of Members in the Room (read-only)
maxSeatCount	NSUInteger	Maximum Number of Mic Seats (only supported when entering the room and creating the room)
isCameraDisableForAllUser	BOOL	Whether to Disable Opening Camera (optional when creating a room), default value: false
isMicrophoneDisableForAllUser	BOOL	Whether to Disable Opening Microphone (optional when creating a room), default value: false
isMessageDisableForAllUser	BOOL	Whether to Disable Sending Messages (optional when creating a room), default value: false
enableCDNStreaming	BOOL	Whether to Enable CDN Live Streaming (optional when creating a room, for live streaming rooms), default value: false
cdnStreamDomain	NSString*	Live Streaming Push Domain (optional when creating a room, for live streaming rooms), default value: empty

## TUILoginUserInfo

User Login Information

Field	Type	Meaning
userId	NSString *	User ID
userName	NSString *	User Name
avatarUrl	NSString *	User Avatar URL

## TUIUserInfo

User Information in the Room

Field	Type	Description
userId	NSString *	User ID
userName	NSString *	User Name
avatarUrl	NSString *	User Avatar URL
userRole	TUIRole	User Role Type
hasAudioStream	BOOL	Whether There is Audio Stream, default value: false
hasVideoStream	BOOL	Whether There is Video Stream, default value: false
hasScreenStream	BOOL	Whether There is Screen Sharing Stream, default value: false

## TUISeatInfo

Seat Information in the Room

Field	Type	Description
index	NSInteger	Mic Seat Number
userId	NSString *	User ID
isLocked	BOOL	Whether the Mic Seat is Locked, default false
isVideoLocked	BOOL	Whether the Mic Seat is Prohibited from Opening Camera, default false
isAudioLocked	BOOL	Whether the Mic Seat is Prohibited from Opening Microphone, default false

## TUISeatLockParams

Lock Seat Operation Parameters

Field	Type	Meaning

lockSeat	BOOL	Lock Mic Seat, default false
lockVideo	BOOL	Lock Mic Seat Camera, default false
lockAudio	BOOL	Lock Mic Seat Microphone, default false

## TUIUserVoiceVolume

User Voice Volume in the Room

Field	Type	Description
userId	NSString *	User ID
volume	NSUInteger	Volume Size, Value range 0 - 100

## TUIRequest

Signaling Request

Field	Type	Description
requestId	NSString *	Request ID
requestAction	<a href="#">TUIRequestAction</a>	Request Type
userId	NSString *	User ID
content	NSString *	Signaling Content
timestamp	NSUInteger	Timestamp

## TUIShareTarget

Screen Sharing Capture Source Information

Field	Type	Description
targetId	NSString *	Capturing Source ID, for windows, this field represents the window ID; for screens, this field represents the display ID
sourceType	<a href="#">TUICaptureSourceType</a>	Capturing Source Type
sourceName	NSString *	Capturing Source Name
thumbnailImage	TUIImage *	Thumbnail

iconImage	TUIImage *	Icon
extInfo	NSDictionary *	Window's Extended Information

## TUINetworkInfo

Network Quality Information

Field	Type	Description
userId	NSString *	User ID
quality	<a href="#">TUINetworkQuality</a>	Network Quality
upLoss	uint32_t	Upstream Packet Loss Rate
downLoss	uint32_t	Downstream Packet Loss Rate
delay	uint32_t	Network Delay

## TUIMessage

Message

Field	Type	Description
messageId	NSString *	Message ID
message	NSString *	Message Text
timestamp	uint64_t	Message Time
userId	NSString *	Message Sender
userName	NSString *	Message Sender Nickname
avatarUrl	NSString *	Message Sender Avatar

# Android

## RoomKit API

Last updated : 2024-03-29 17:53:52

Module: TUIRoomKit

Function: Multi-person audio and video main function interface

Version: 2.1.0

### TUIRoomKit

## TUIRoomKit

function list	describe
<a href="#">createInstance</a>	Create a TUIRoomKit instance (singleton mode).
<a href="#">destroyInstance</a>	Destroy the TUIRoomKit instance.
<a href="#">setSelfInfo</a>	Set up personal information, including username and avatar.
<a href="#">createRoom</a>	Create a room.
<a href="#">enterRoom</a>	Enter the room.

## createInstance

Initialize the TUIRoomKit singleton object.



```
public static TUIRoomKit createInstance();
```

### **destroyInstance**

Destroy the TUIRoomKit instance.



```
public static void destroyInstance();
```

## setSelfInfo

Set up personal information, including username and avatar.



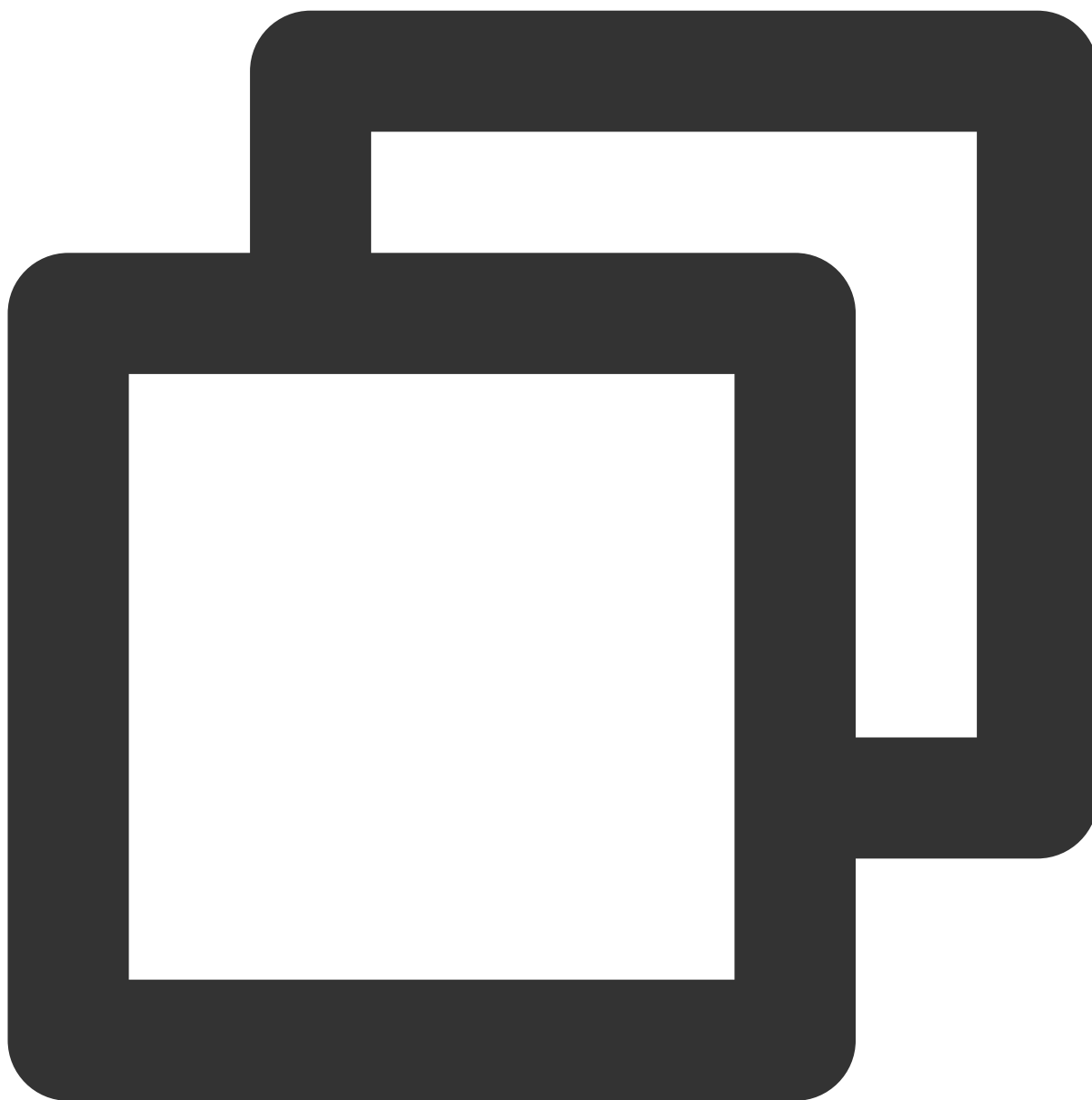


```
public abstract void setSelfInfo(String userName, String avatarURL,  
                                TUIRoomDefine.ActionCallback callback);
```

parameter	describe
userName	The individual's username.
avatarURL	Personal avatar link.
callback	Callback for success in setting personal information.

## createRoom

Create a room.



```
public abstract void createRoom(TUIRoomDefine.RoomInfo roomInfo, TUIRoomDefine.Acti
```

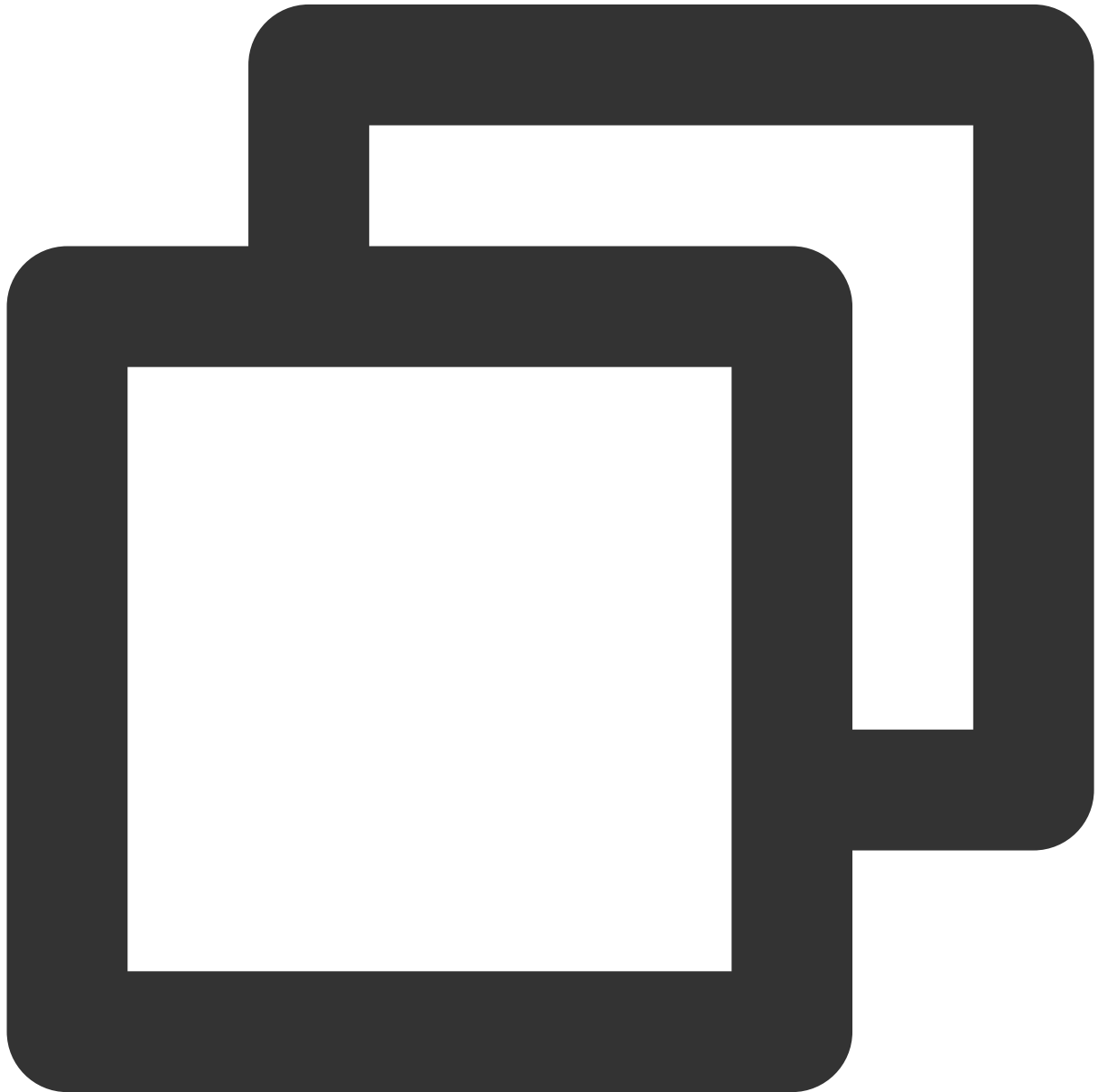
parameter	describe
roomInfo	Parameters for creating a room, including room number, room name, etc., where roomId is required and the rest can be default values.

callback

Callback to determine whether the room is successfully created.

## enterRoom

Enter the room.



```
public abstract void enterRoom(String roomId,  
                                boolean enableAudio,  
                                boolean enableVideo,
```

```
boolean isSoundOnSpeaker,  
TUIRoomDefine.GetRoomInfoCallback callback);
```

parameter	describe
roomId	Room number to enter the room.
enableAudio	true: When entering the room, turn on the microphone and push local audio data to the remote end. Other members can hear the local sound normally; false: When entering the room, only the microphone is turned on and local audio data is not pushed to the remote end. Other members cannot hear the local sound.
enableVideo	true: Enter the room, turn on the camera, and push the local video data to the remote end. Other members can see the local picture normally; false: When entering the room, the camera will not be turned on and local video data will not be pushed to the remote end. Other members will not be able to see the local video.
isSoundOnSpeaker	Whether to use the speaker to play sound, true to use the speaker, false to use the earpiece.
callback	Callback whether the room entry is successful.

# RoomEngine API

## API Overview

Last updated : 2023-10-24 17:16:07

### TUIRoomEngine (No UI Interface)

TUIRoomEngine API is the Audio/Video call Component's No UI Interface, you can use this set of API to customize packaging according to your business needs.

TUIRoomEngine

#### TUIRoomEngine Core Methods

API	Description
<a href="#">createInstance</a>	Create TUIRoomEngine Instance
<a href="#">destroyInstance</a>	Destroy TUIRoomEngine Instance
<a href="#">login</a>	Login interface, you need to initialize user information before entering the room and perform a series of operations.
<a href="#">logout</a>	Logout interface, there will be actively leave room operation, destroy resources
<a href="#">setSelfInfo</a>	Set local user name and avatar
<a href="#">getSelfInfo</a>	Get local user basic information
<a href="#">addObserver</a>	Set event callback
<a href="#">removeObserver</a>	Remove event callback

#### Room Related Active Interface

API	Description
<a href="#">createRoom</a>	Create room
<a href="#">destroyRoom</a>	Close the room
<a href="#">enterRoom</a>	Entered room
<a href="#">exitRoom</a>	Leave room

<a href="#">connectOtherRoom</a>	Connect to other room
<a href="#">disconnectOtherRoom</a>	Disconnect from other room
<a href="#">fetchRoomInfo</a>	Get room data
<a href="#">updateRoomNameByAdmin</a>	Update room name
<a href="#">updateRoomSpeechModeByAdmin</a>	Set room management mode (only administrator or group owner can call)

## Local User View Rendering, Video Management

API	Description
<a href="#">setLocalVideoView</a>	Set the view control for local user video rendering
<a href="#">openLocalCamera</a>	Open local camera
<a href="#">closeLocalCamera</a>	Close local camera
<a href="#">updateVideoQuality</a>	Update local video codec quality settings
<a href="#">startScreenSharing</a>	Start screen sharing
<a href="#">stopScreenSharing</a>	End screen sharing
<a href="#">startPushLocalVideo</a>	Start pushing local video
<a href="#">stopPushLocalVideo</a>	Stop pushing local video

## Local User Audio Management

API	Description
<a href="#">openLocalMicrophone</a>	Open local microphone
<a href="#">closeLocalMicrophone</a>	Close local microphone
<a href="#">updateAudioQuality</a>	Update local audio codec quality settings
<a href="#">startPushLocalAudio</a>	Start pushing local audio
<a href="#">stopPushLocalAudio</a>	Stop pushing local audio

## Remote User View Rendering, Video Management

API	Description
-----	-------------

<a href="#">setRemoteVideoView</a>	Set the view control for remote user video rendering
<a href="#">startPlayRemoteVideo</a>	Start playing remote user video
<a href="#">stopPlayRemoteVideo</a>	Stop playing remote user video
<a href="#">muteRemoteAudioStream</a>	Mute remote user

## Room User Information

API	Description
<a href="#">getUserList</a>	Get the member list in the room
<a href="#">getUserInfo</a>	Get member information

## Room User Management

API	Description
<a href="#">changeUserRole</a>	Modify user role (only administrator or group owner can call)
<a href="#">kickRemoteUserOutOfRoom</a>	Kick Remote User out of the Room (Only Administrator or Group Owner can call)

## Speech Management in Room

API	Description
<a href="#">disableDeviceForAllUserByAdmin</a>	Media Device Management for All Users (Only Administrator or Group Owner can call)
<a href="#">openRemoteDeviceByAdmin</a>	Request Remote User to Open Media Device (Only Administrator or Group Owner can call)
<a href="#">closeRemoteDeviceByAdmin</a>	Close Remote User's Media Device (Only Administrator or Group Owner can call)
<a href="#">applyToAdminToOpenLocalDevice</a>	Request to Open Local Media Device (Available for Ordinary Users)

## Microphone Seat Management in Room

API	Description
<a href="#">setMaxSeatCount</a>	Set Maximum Number of Microphone Seats (Only supported when entering the

	room and creating the room)
<a href="#">getSeatList</a>	Get Microphone Seat List
<a href="#">lockSeatByAdmin</a>	Lock Microphone Seat (Including Position Lock, Audio State Lock, Video State Lock)
<a href="#">takeSeat</a>	Go Live Locally Conference Scene: <a href="#">SPEAK_AFTER_TAKING_SEAT</a> mode requires application to the host or administrator to allow going live, other modes do not support going live. Live Broadcast Scene: <a href="#">FREE_TO_SPEAK</a> mode allows free going live, and speak after going live; <a href="#">SPEAK_AFTER_TAKING_SEAT</a> mode requires application to the host or administrator to allow going live; other modes do not support going live.
<a href="#">leaveSeat</a>	Leave Microphone Seat Locally
<a href="#">takeUserOnSeatByAdmin</a>	Host/Administrator invites user to go live
<a href="#">kickUserOffSeatByAdmin</a>	Host/Administrator kicks user off the microphone seat

## Signaling Management

API	Description
<a href="#">cancelRequest</a>	Cancel Request
<a href="#">responseRemoteRequest</a>	Reply to Request

## Send Message

API	Description
<a href="#">sendTextMessage</a>	Send Text Message
<a href="#">sendCustomMessage</a>	Send Custom Message
<a href="#">disableSendingMessageByAdmin</a>	Disable Remote User's Text Message Sending Ability (Only Administrator or Group Owner can call)
<a href="#">disableSendingMessageForAllUser</a>	Disable All Users' Text Message Sending Ability (Only Administrator or Group Owner can call) Advanced Feature: Get TRTC Instance

## Advanced Feature: Get TRTC Instance

API	Description
-----	-------------



<a href="#">getTRTCCloud</a>	Get TRTC Instance Object
<a href="#">getDeviceManager</a>	Get Device Management Object
<a href="#">getAudioEffectManager</a>	Get Audio Effect Management Object
<a href="#">getBeautyManager</a>	Get Beauty Management Object

## Event Type Definition

TUIRoomObserver is the Callback Event class corresponding to TUIRoomEngine. You can listen to the callback events you need through this callback.

TUIRoomObserver

### Error Callback

Event	Description
<a href="#">onError</a>	Error Callback Event

### Login Status Event Callback

API	Description
<a href="#">onKickedOffLine</a>	User Kicked Offline Event
<a href="#">onUserSigExpired</a>	User Credential Timeout Event

### Room Event Callback

API	Description
<a href="#">onRoomNameChanged</a>	Room Name Change Event
<a href="#">onAllUserMicrophoneDisableChanged</a>	All Users' Microphones Disabled in Room Event
<a href="#">onAllUserCameraDisableChanged</a>	All Users' Cameras Disabled in Room Event
<a href="#">onSendMessageForAllUserDisableChanged</a>	All Users' Text Message Sending Disabled in Room Event
<a href="#">onRoomDismissed</a>	Room Dismissed Event

<a href="#">onKickedOutOfRoom</a>	Kicked Out of Room Event
<a href="#">onRoomSpeechModeChanged</a>	Room Microphone Control Mode Change

### Room User Event Callback

API	Description
<a href="#">onRemoteUserEnterRoom</a>	Remote User Entering Room Event
<a href="#">onRemoteUserLeaveRoom</a>	Remote User Leaving Room Event
<a href="#">onUserRoleChanged</a>	User Role Change Event
<a href="#">onUserVideoStateChanged</a>	User Video State Change Event
<a href="#">onUserAudioStateChanged</a>	User Audio State Change Event
<a href="#">onUserVoiceVolumeChanged</a>	User Volume Change Event
<a href="#">onSendMessageForUserDisableChanged</a>	User Text Message Sending Ability Change Event
<a href="#">onUserNetworkQualityChanged</a>	User Network Status Change Event
<a href="#">onUserScreenCaptureStopped</a>	Screen Sharing End Event

### Room Microphone Seat Event Callback

API	Description
<a href="#">onRoomMaxSeatCountChanged</a>	Room Maximum Microphone Seat Number Change Event (Only effective in conference type rooms)
<a href="#">onSeatListChanged</a>	Microphone Seat List Change Event
<a href="#">onKickedOffSeat</a>	Received User Kicked Off Microphone Event

### Request Signaling Event Callback

API	Description
<a href="#">onRequestReceived</a>	Received Request Message Event
<a href="#">onRequestCancelled</a>	Received Request Cancellation Event

### Room Message Event Callback

API	Description
<a href="#">onReceiveTextMessage</a>	Received Normal Text Message Event
<a href="#">onReceiveCustomMessage</a>	Received Custom Message Event

# TUIRoomEngine

Last updated : 2023-10-23 11:32:46

## TUIRoomEngine (No UI Interface)

TUIRoomEngine API is a No UI Interface for Conference Component, you can use this API to custom encapsulate according to your business needs.

### **createInstance**

Create TUIRoomEngine Instance



```
static TUIRoomEngine createInstance()
```

**return:**TUIRoomEngine Instance

### **destroyInstance**

Destroy TUIRoomEngine Instance



```
void destroyInstance()
```

## login

Login interface, you need to initialize user information before entering the room and perform a series of operations

**Note :** In v1.0.0, this interface is named `setup`, and in v1.0.1 and above, please use `TUIRoomEngine.login` to log in to `TUIRoomEngine`.



```
void static login(Context context,  
    int sdkAppId,  
    String userId,  
    String userSig,  
    TUIRoomDefine.ActionCallback callback)
```

**Parameters:**

Parameter	Type	Meaning
context	Context	Android Context

sdkAppld	int	Get sdkAppld information from Application Info
userId	String	User ID
userSig	String	UserSig
callback	TUIRoomDefine.ActionCallback	API Callback for notifying the success or failure of the interface call

## logout

Logout interface, there will be actively leave the room operation, destroy resources





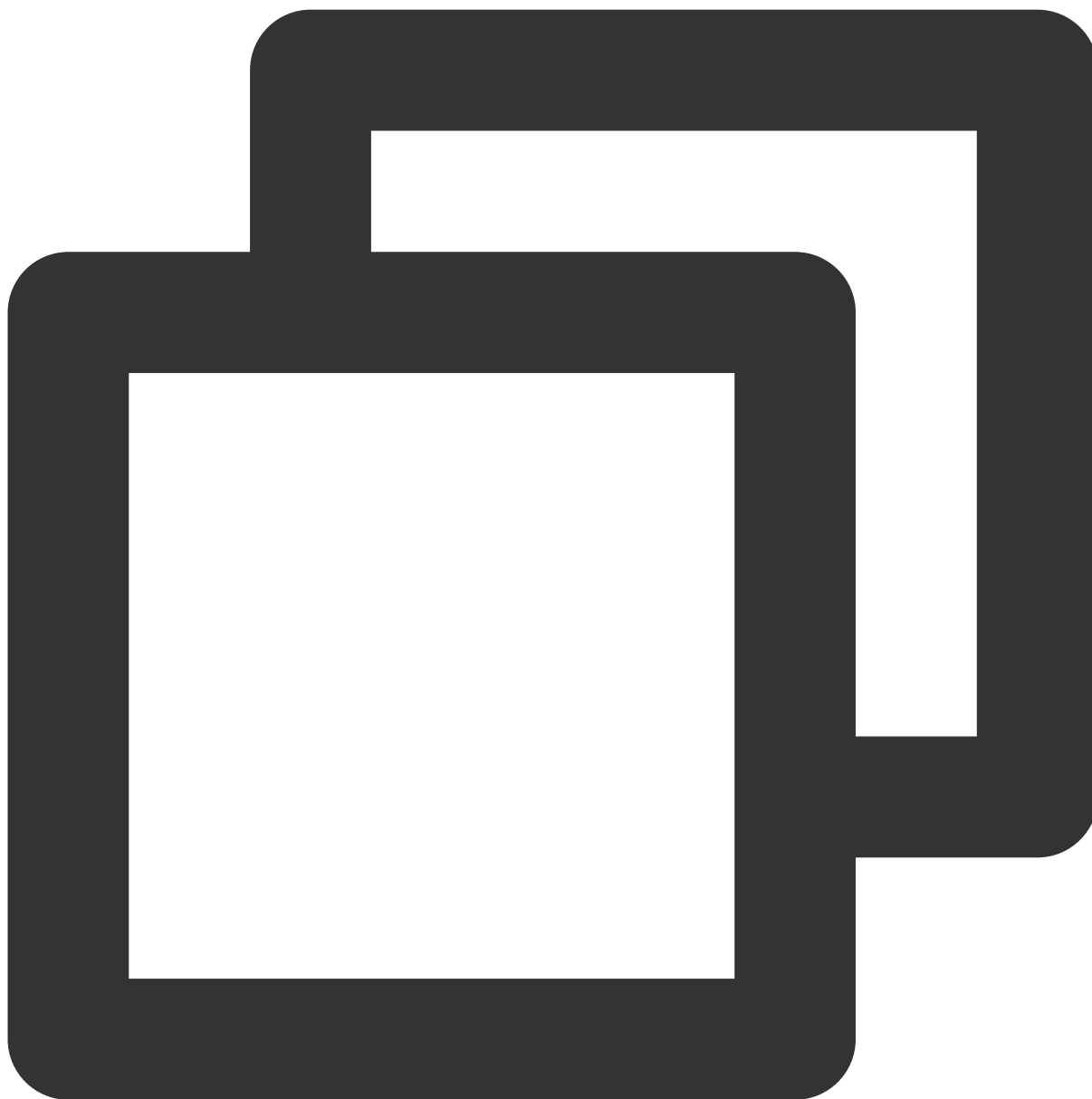
```
static void logout(TUIRoomDefine.ActionCallback callback)
```

**Parameters:**

Parameter	Type	Meaning
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

**setSelfInfo**

Set local user name and avatar



```
static void setSelfInfo(String userName, String avatarURL, TUIRoomObserver.ActionCa
```

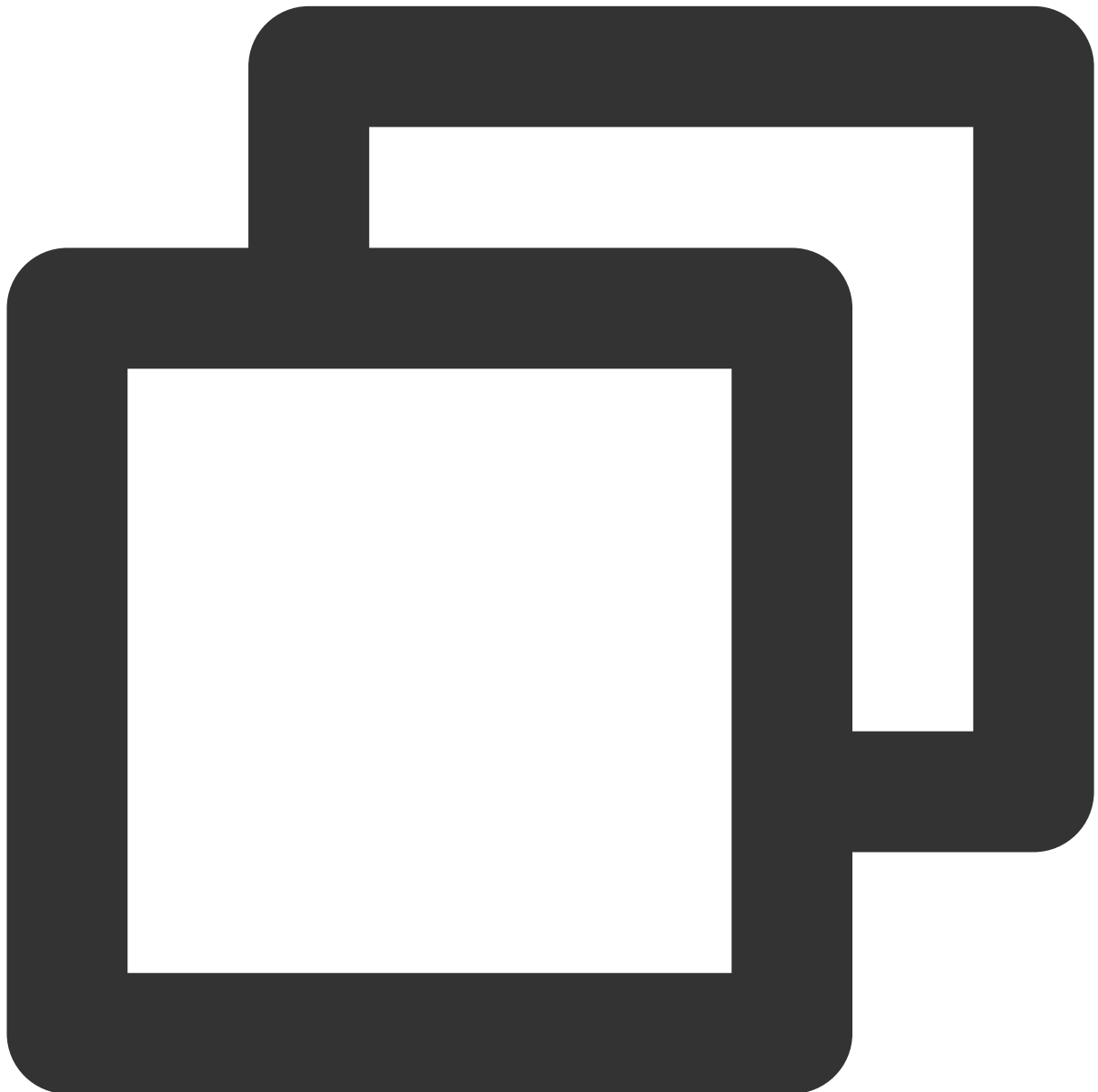
**Parameters:**

Parameter	Type	Meaning
userName	int	User Name
avatarUrl	String	User avatar URL address

callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call
----------	------------------------------	--

## getSelfInfo

Get the basic information of the local user login

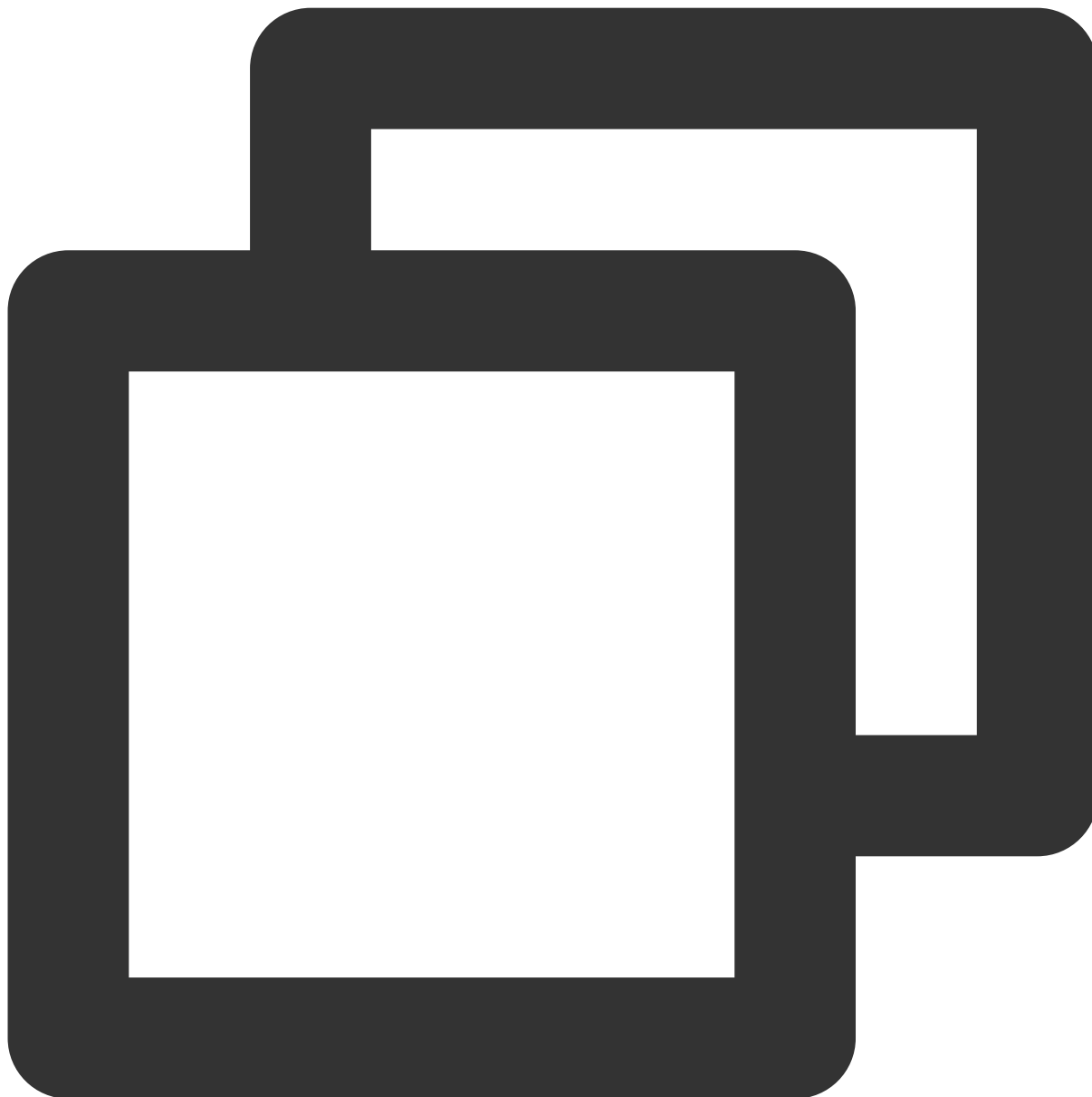


```
static TUIRoomDefine.LoginUserInfo getSelfInfo()
```

**return:**the basic information of the local user login

## addObserver

Add TUIRoomEngine Event Callback



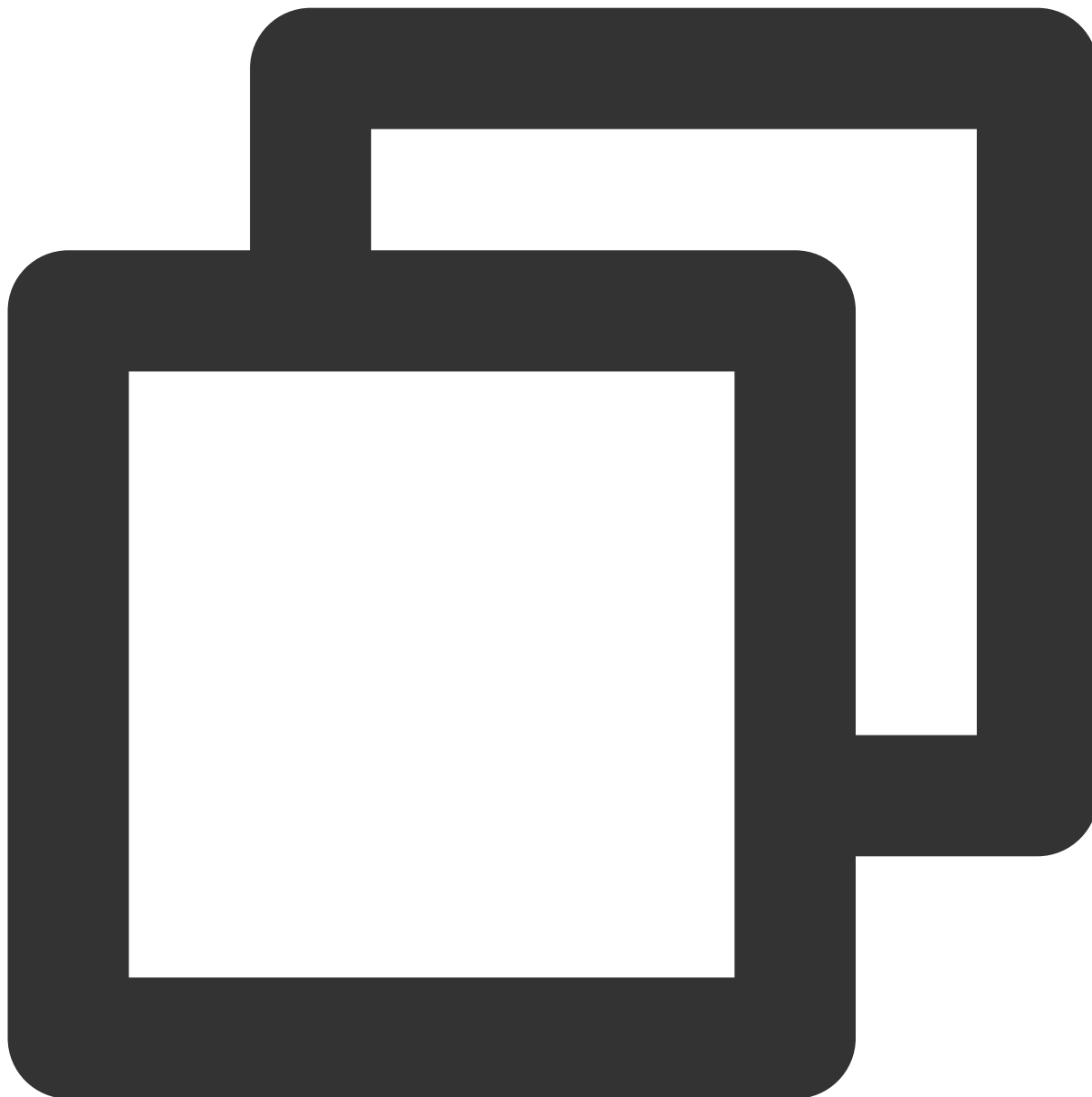
```
void addObserver(TUIRoomObserver observer)
```

### Parameters:

Parameter	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine Event Callback

## removeObserver

Remove TUIRoomEngine Event Callback

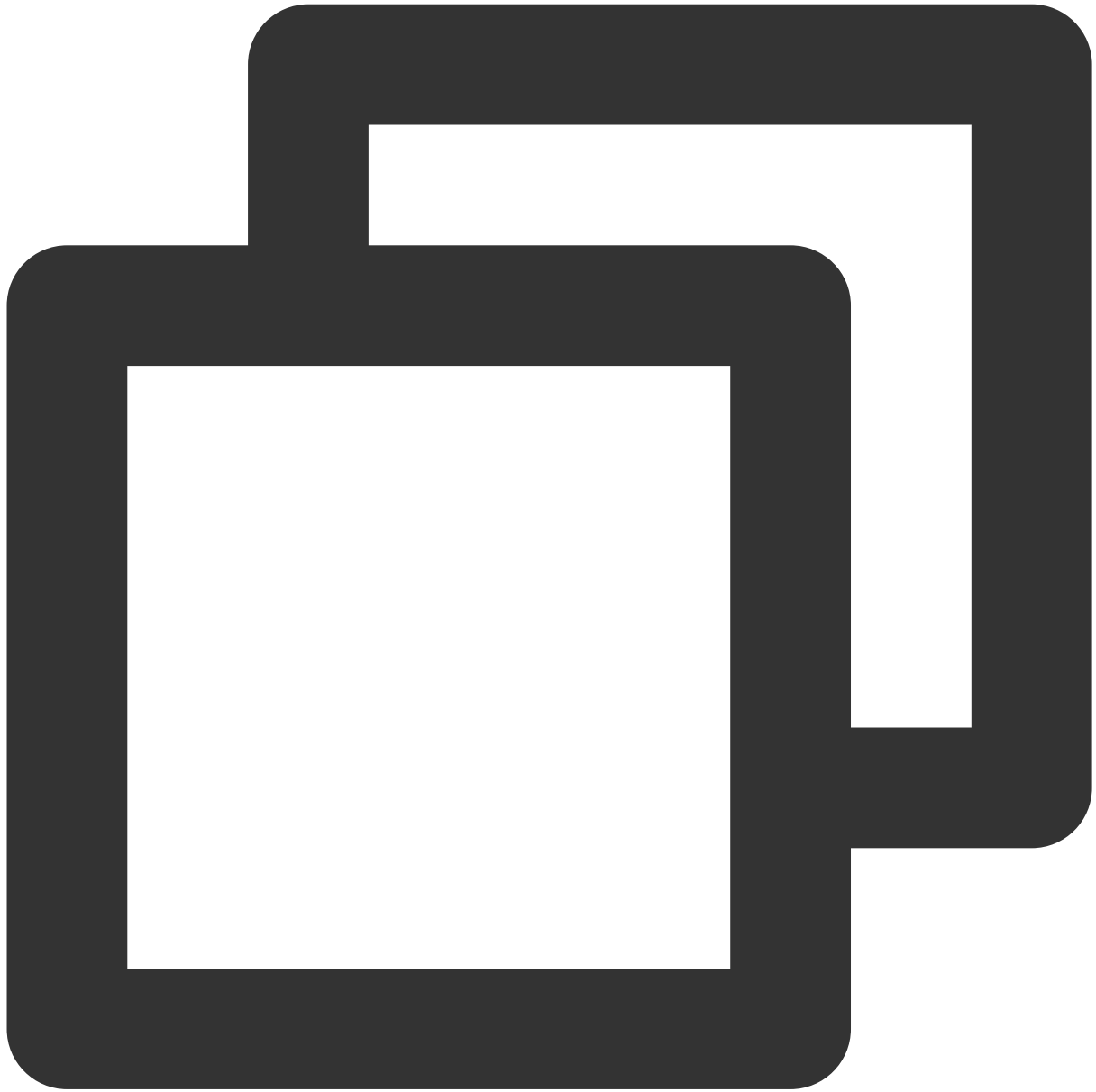


```
void removeObserver(TUIRoomObserver observer)
```

Parameter	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine Event Callback

## createRoom

The host creates a room, and the user who calls `createRoom` is the owner of the room. When creating a room, you can set the Room ID, room name, and whether the room allows users to join, enable video and audio, send messages, and other functions.



```
void createRoom(TUIRoomDefine.RoomInfo roomInfo, TUIRoomDefine.ActionCallback callb
```

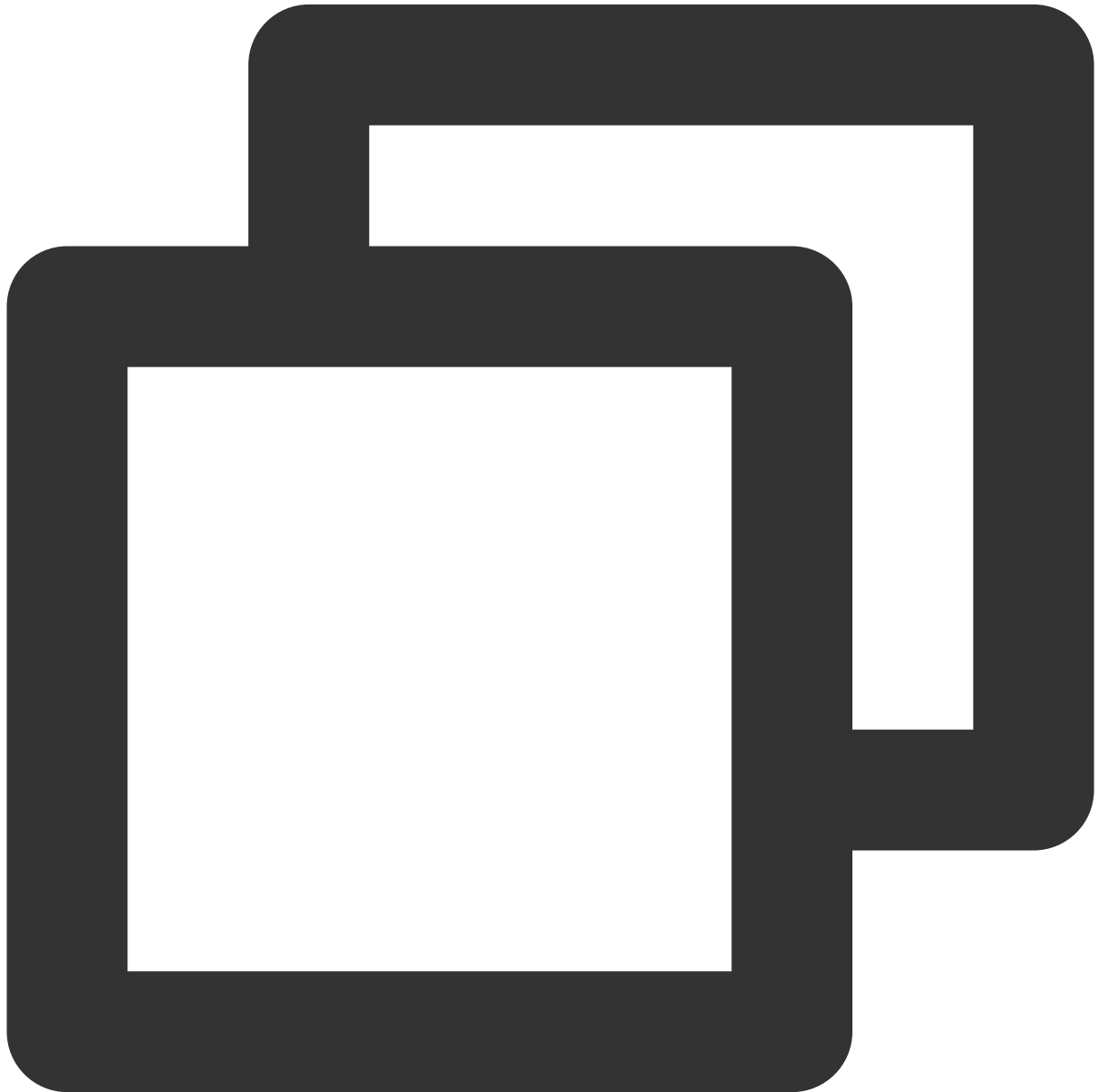
**Parameters:**

Parameter	Type	Meaning
roomInfo	<a href="#">TUIRoomDefine.RoomInfo</a>	Room data

callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call
----------	------------------------------	--

## destroyRoom

Destroy Room Interface, the room owner must initiate the destruction of the room. After the room is destroyed, it is unavailable for entry.



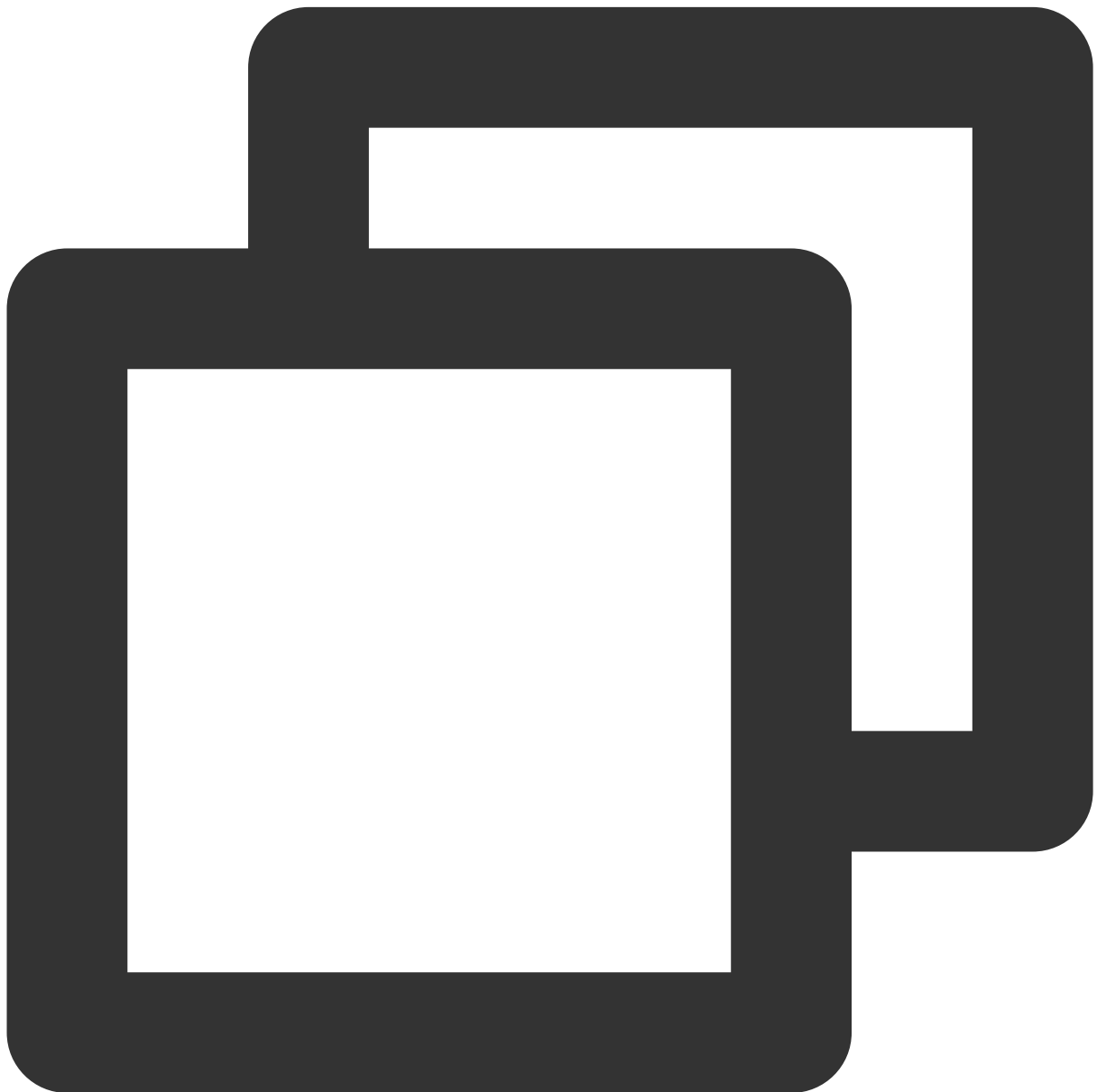
```
void destroyRoom(TUIRoomDefine.ActionCallback callback)
```

### Parameters:

Parameter	Type	Meaning
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

## enterRoom

Entered room.



```
void enterRoom(String roomId, TUIRoomDefine.GetRoomInfoCallback callback)
```

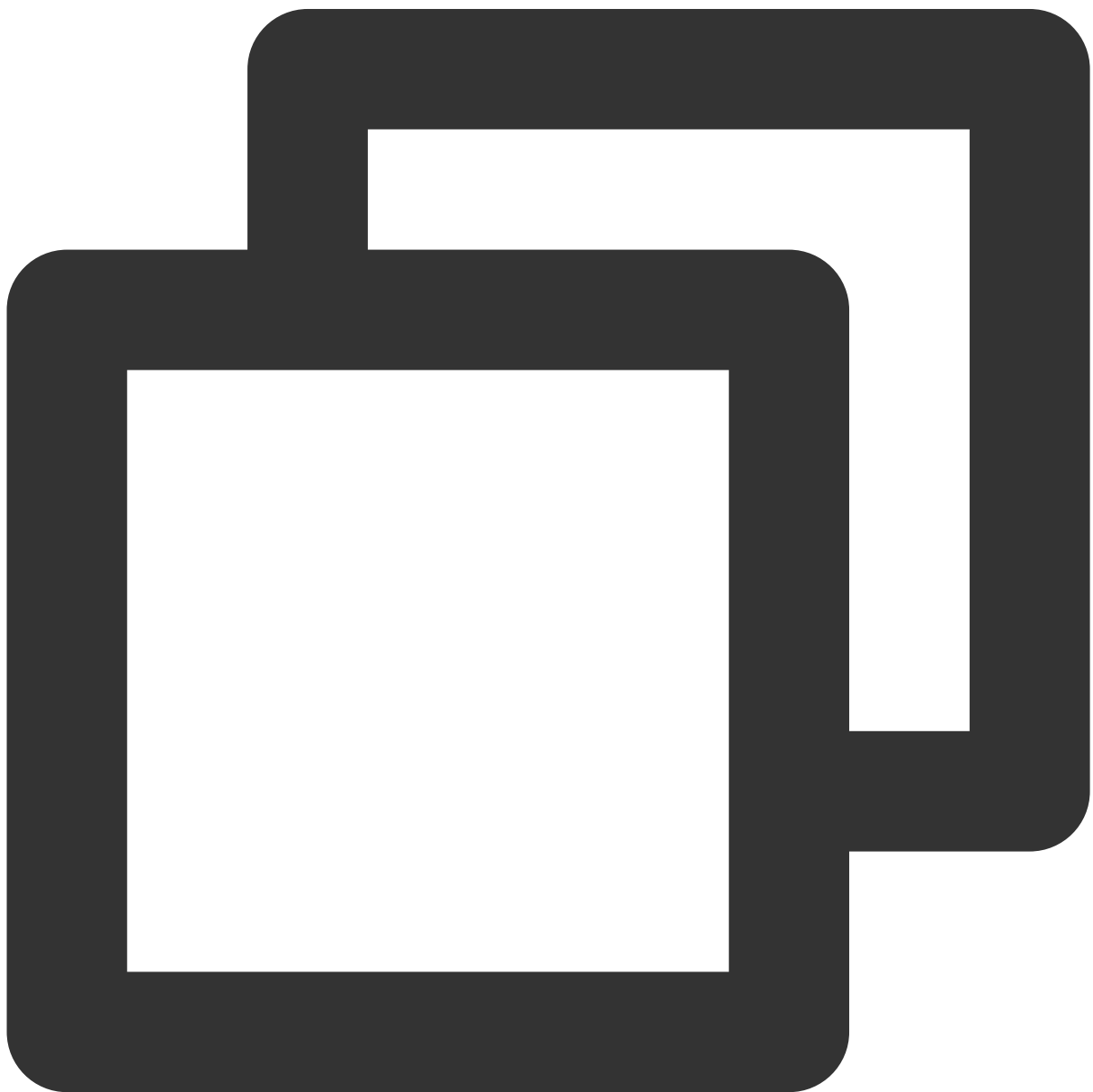


**Parameters:**

Parameter	Type	Meaning
roomId	String	Room ID
callback	TUIRoomDefine.GetRoomInfoCallback	Get the entered Room data Callback

**exitRoom**

Exit Room Interface, after the user executes Enter Room, they can leave the room through Leave Room.



```
void exitRoom(boolean isSyncWaiting, TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

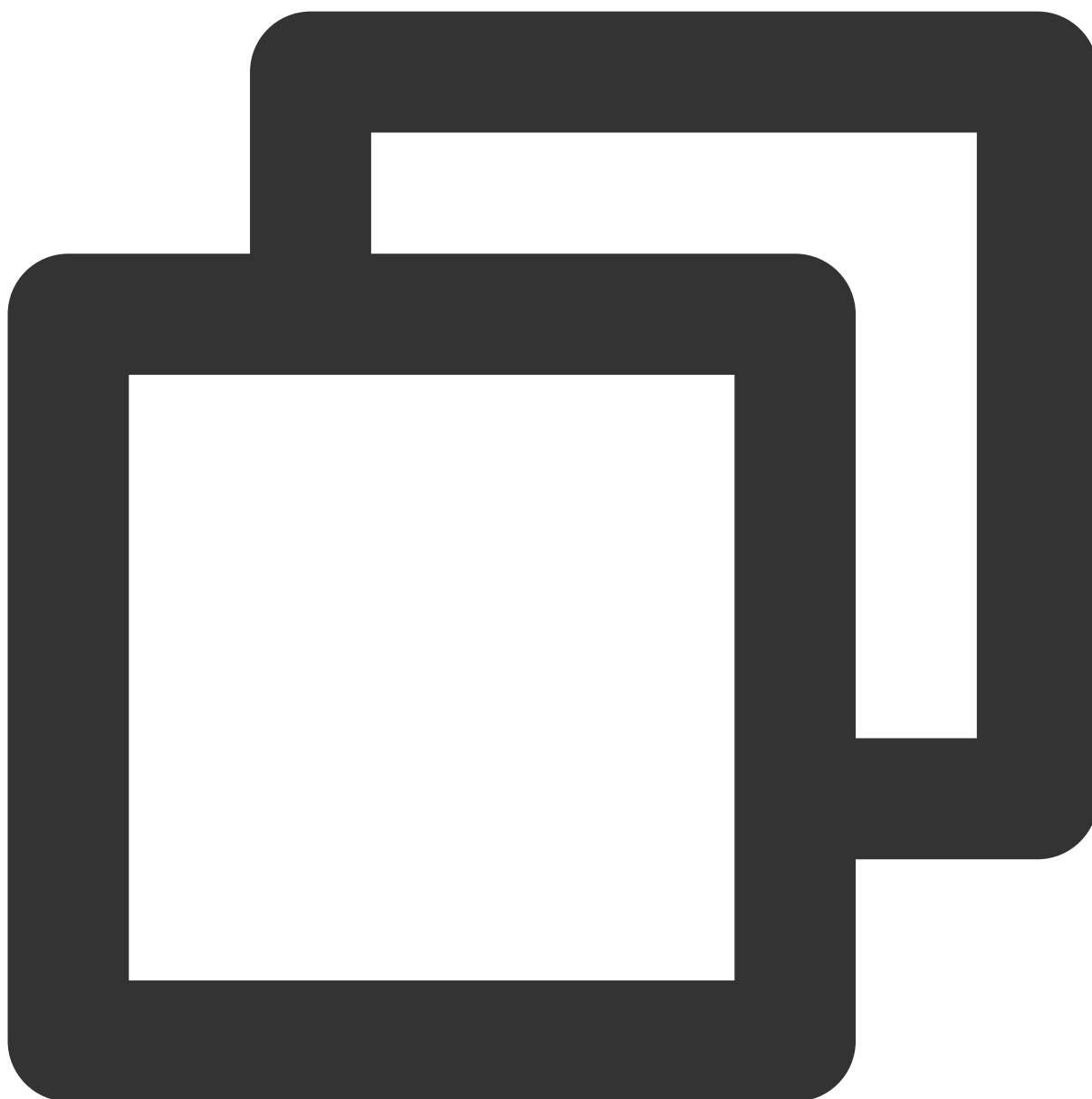
Parameter	Type	Meaning
isSyncWaiting	boolean	Whether to synchronize leaving the room
callback	TUIRoomDefine.ActionCallback	Leave Room Result Callback

**connectOtherRoom**

Connect to other rooms

**Description:**

Used for live streaming scenario to apply for cross-room streaming



```
TUIRoomDefine.Request connectOtherRoom(String roomId,  
                                         String userId,  
                                         int timeout,  
                                         TUIRoomDefine.RequestCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
roomId	String	Room ID

userId	String	User ID
timeout	int	Time
callback	TUIRoomDefine.RequestCallback	Connect to other rooms Callback

**Return :** Request body

### **disconnectOtherRoom**

Disconnect from other rooms

**Description:**

Used for live streaming scenario to disconnect cross-room streaming



```
void disconnectOtherRoom(TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
callback	TUIRoomDefine.ActionCallback	Disconnect with other rooms Connection result Callback

## fetchRoomInfo

Get Room data



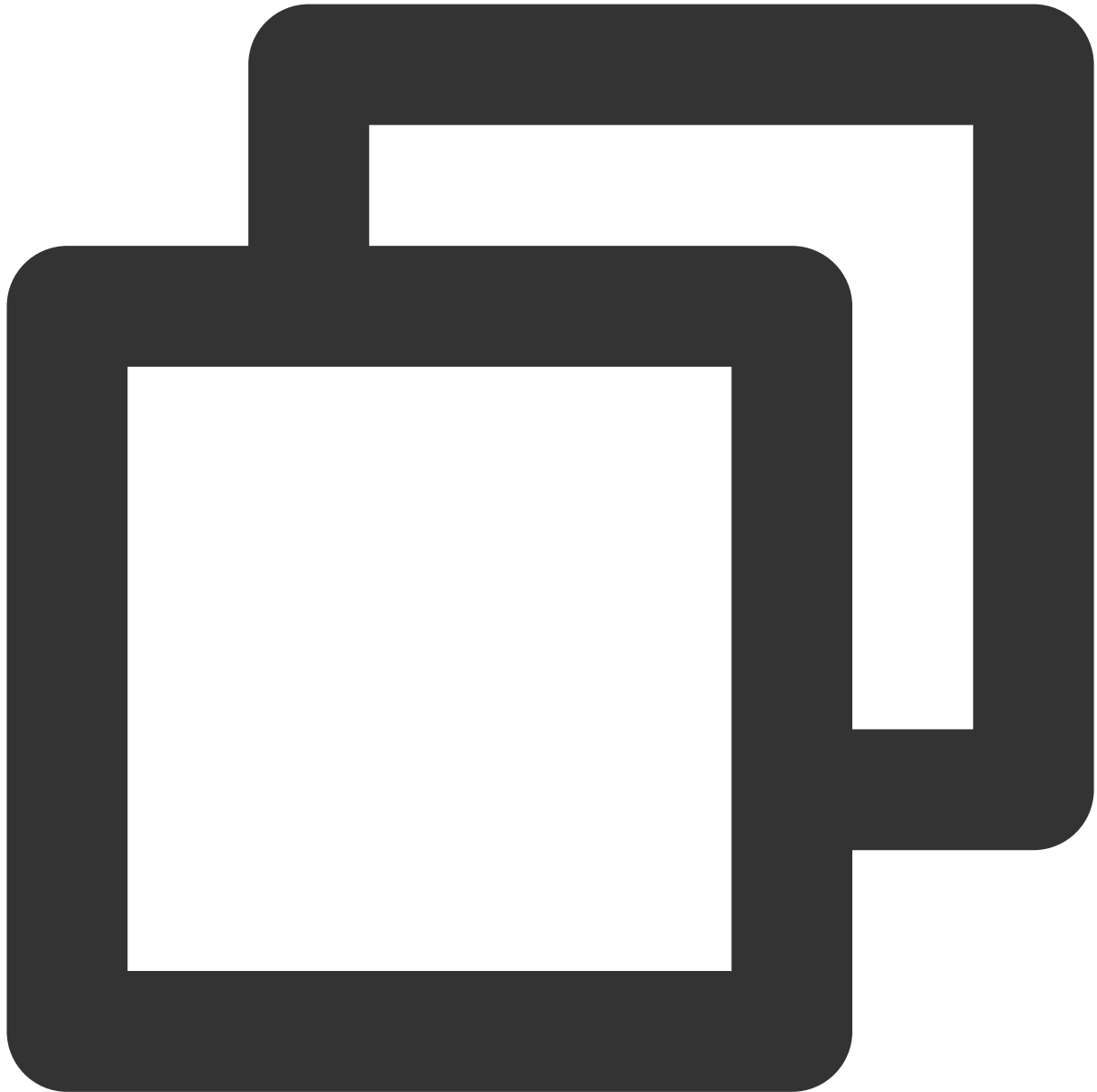
```
void fetchRoomInfo(TUIRoomDefine.GetRoomInfoCallback callback);
```

### Parameters:

Parameter	Type	Meaning
callback	TUIRoomDefine.GetRoomInfoCallback	Get Room data Callback

## updateRoomNameByAdmin

Update Room ID



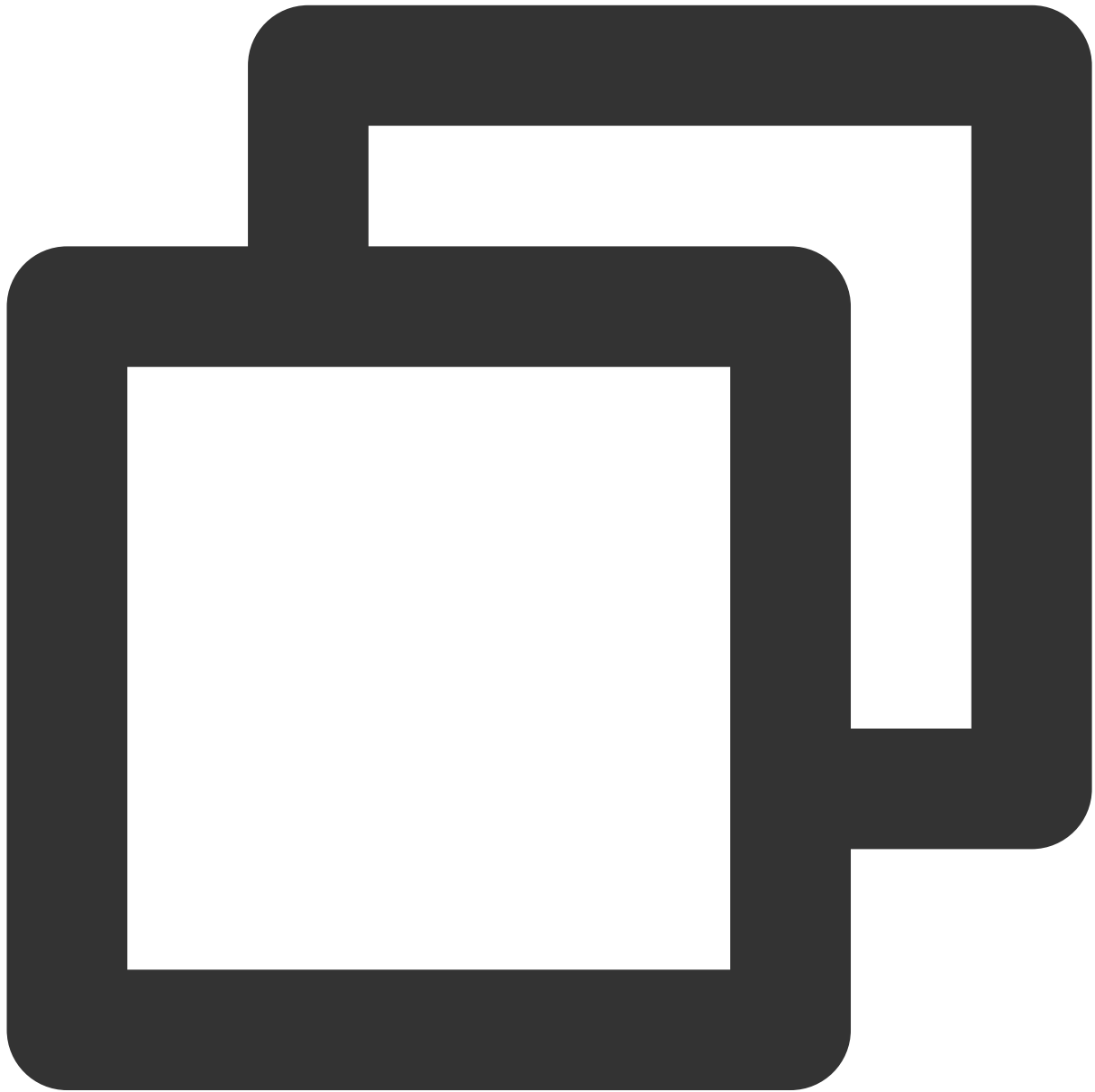
```
void updateRoomNameByAdmin(String roomName, TUIRoomDefine.ActionCallback callback);
```

### Parameters:

Parameter	Type	Meaning
roomName	String	Room ID
callback	TUIRoomDefine.ActionCallback	Update Operation result Callback

## updateRoomSpeechModeByAdmin

Set Management mode (only Administrator or Group owner can call)



```
void updateRoomSpeechModeByAdmin(TUIRoomDefine.SpeechMode mode, TUIRoomDefine.Actio
```

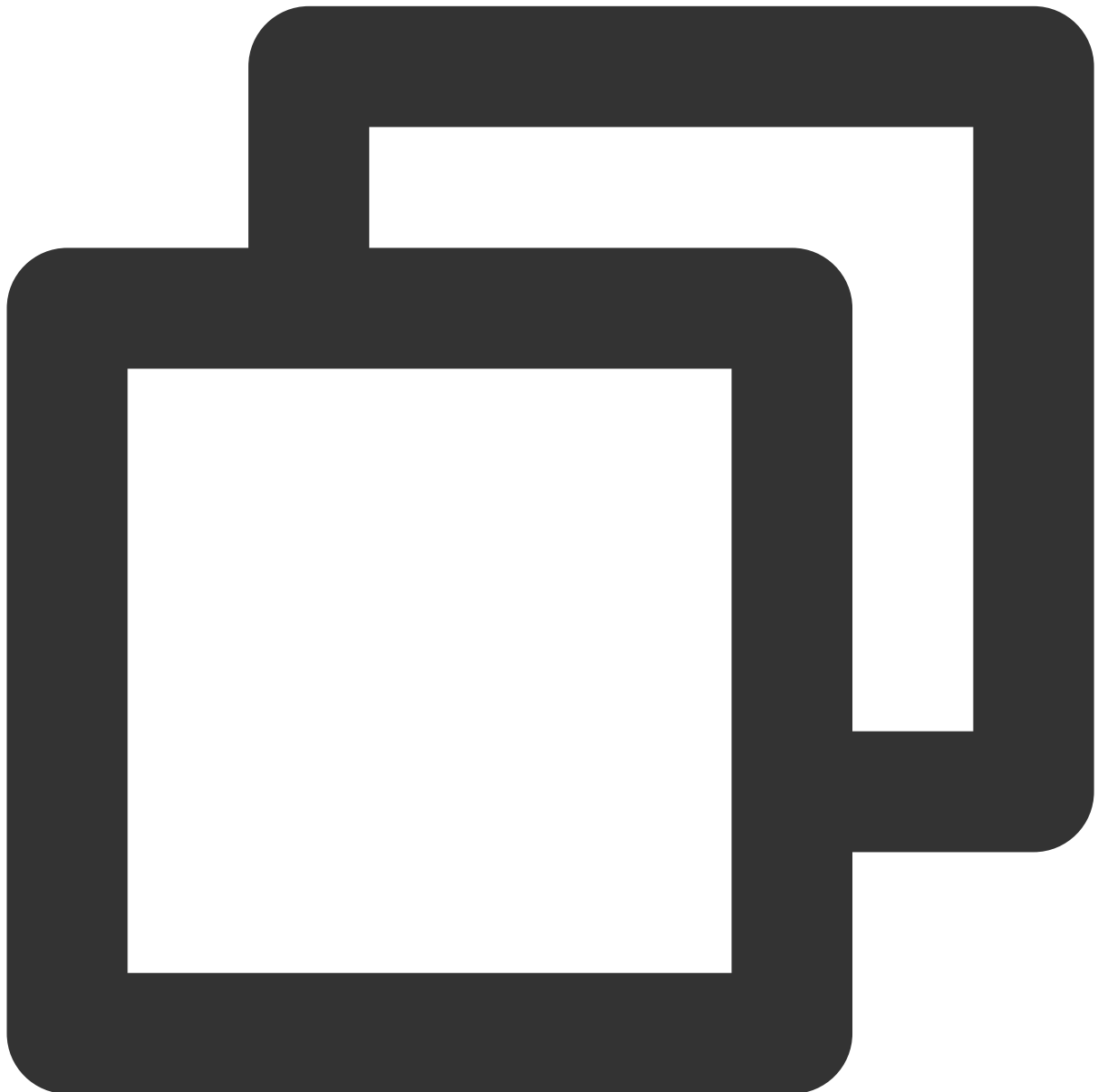
Parameter	Type	Meaning
mode	<a href="#">TUIRoomDefine.SpeechMode</a>	Management mode



callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call
----------	------------------------------	--

## setLocalVideoView

Set Local user Video Rendering View control



```
void setLocalVideoView(TUIRoomDefine.VideoStreamType streamType,  
                        TUIVideoView view);
```

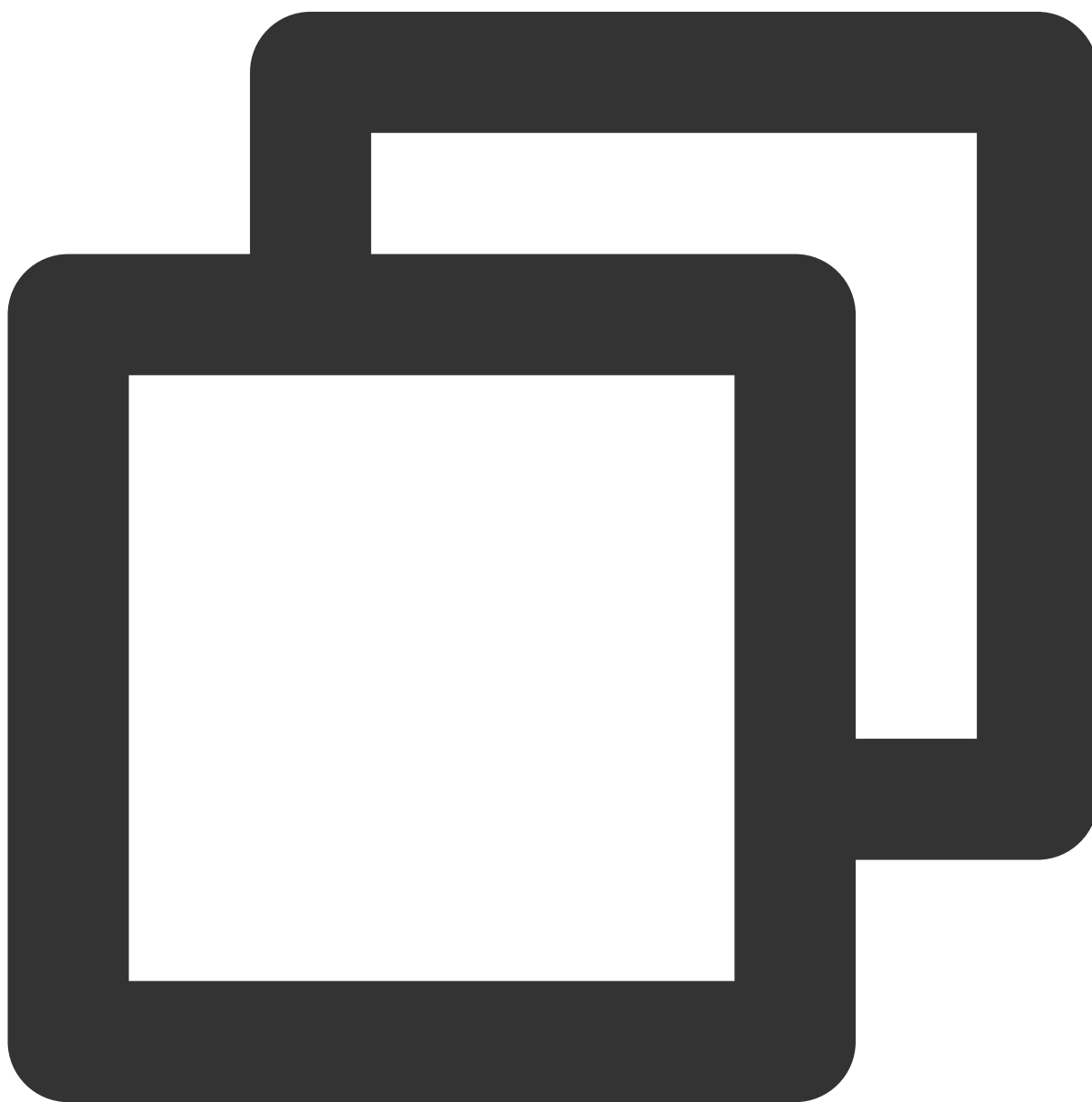
### Parameters:

--	--	--

Parameter	Type	Meaning
streamType	<a href="#">TUIRoomDefine.VideoStreamType</a>	Local streams type
view	TUIVideoView	To be rendered view, Video Rendering on this view

## openLocalCamera

Open Local Camera, Start Video Capturing



```
void openLocalCamera(boolean isFront,  
                    TUIRoomDefine.VideoQuality quality,  
                    TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
isFront	boolean	Whether to use Front Camera
quality	<a href="#">TUIRoomDefine.VideoQuality</a>	Video Quality
callback	TUIRoomDefine.ActionCallback	Open Camera Operation result Callback

**closeLocalCamera**

Close Local Camera



```
void closeLocalCamera();
```

## updateVideoQuality

Set Local Video Parameter



```
void updateVideoQuality(TUIRoomDefine.VideoQuality quality);
```

**Parameters:**

Parameter	Type	Meaning
quality	<a href="#">TUIRoomDefine.VideoQuality</a>	Video Quality

**startScreenSharing**

Start Screen sharing



```
void startScreenSharing();
```

## stopScreenSharing

Stop Screen sharing



```
void stopScreenSharing();
```

### **startPushLocalVideo**

Start pushing Local Video streams to Remote



```
void startPushLocalVideo();
```

### **stopPushLocalVideo**

Stop pushing Local Video streams to Remote





```
void stopPushLocalVideo();
```

## openLocalMicrophone

Open Local mic

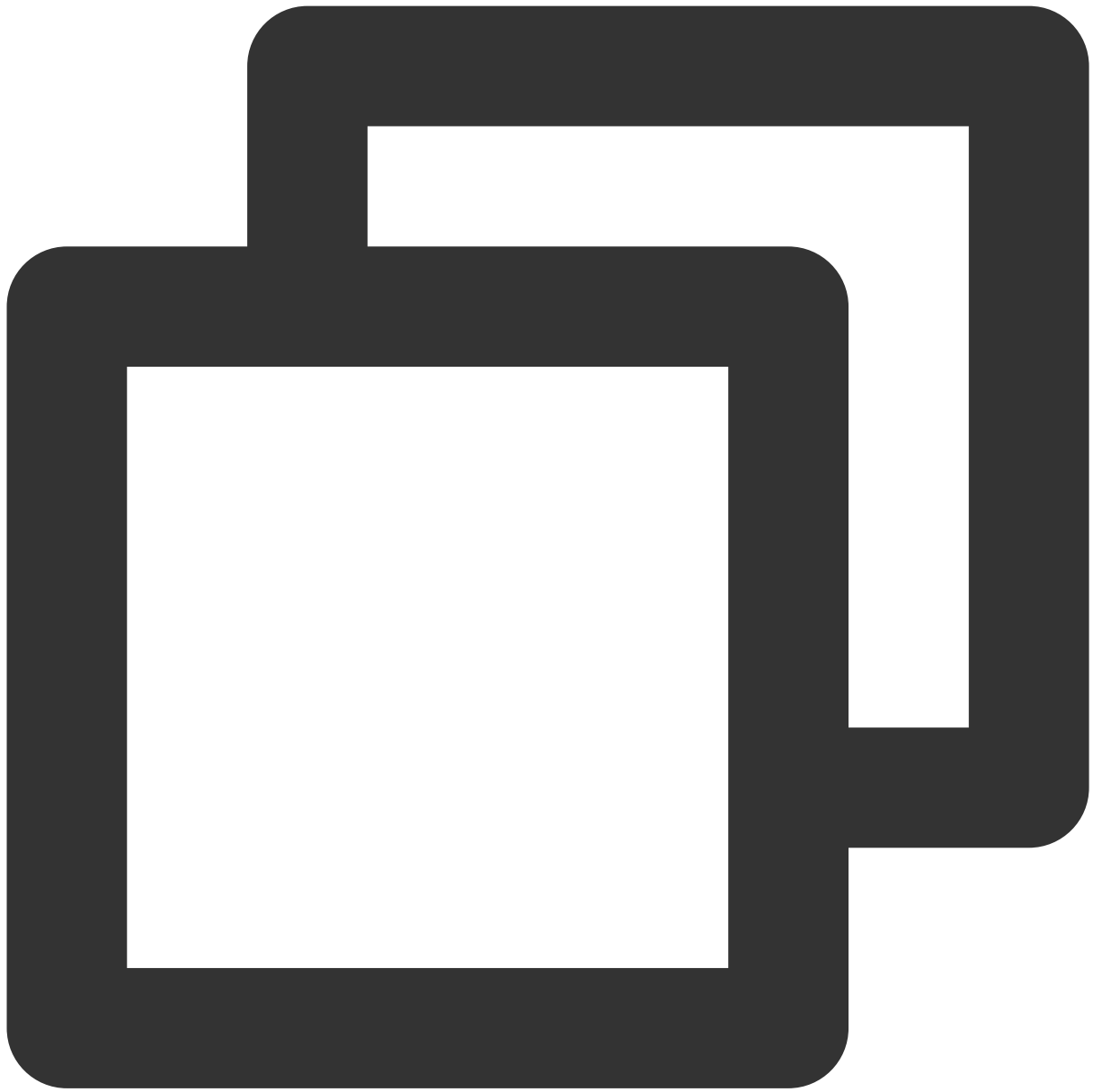


```
void openLocalMicrophone(TUIRoomDefine.AudioQuality quality, TUIRoomDefine.ActionCa
```

Parameter	Type	Meaning
quality	<a href="#">TUIRoomDefine.AudioQuality</a>	Audio Quality
callback	TUIRoomDefine.ActionCallback	Open mic Operation result Callback

### closeLocalMicrophone

Close Local mic



```
void closeLocalMicrophone();
```

### **updateAudioQuality**

Update Local Audio Codec Quality setting



```
void updateAudioQuality(TUIRoomDefine.AudioQuality quality);
```

**Parameters:**

Parameter	Type	Meaning
quality	<a href="#">TUIRoomDefine.AudioQuality</a>	Audio Quality

**startPushLocalAudio**

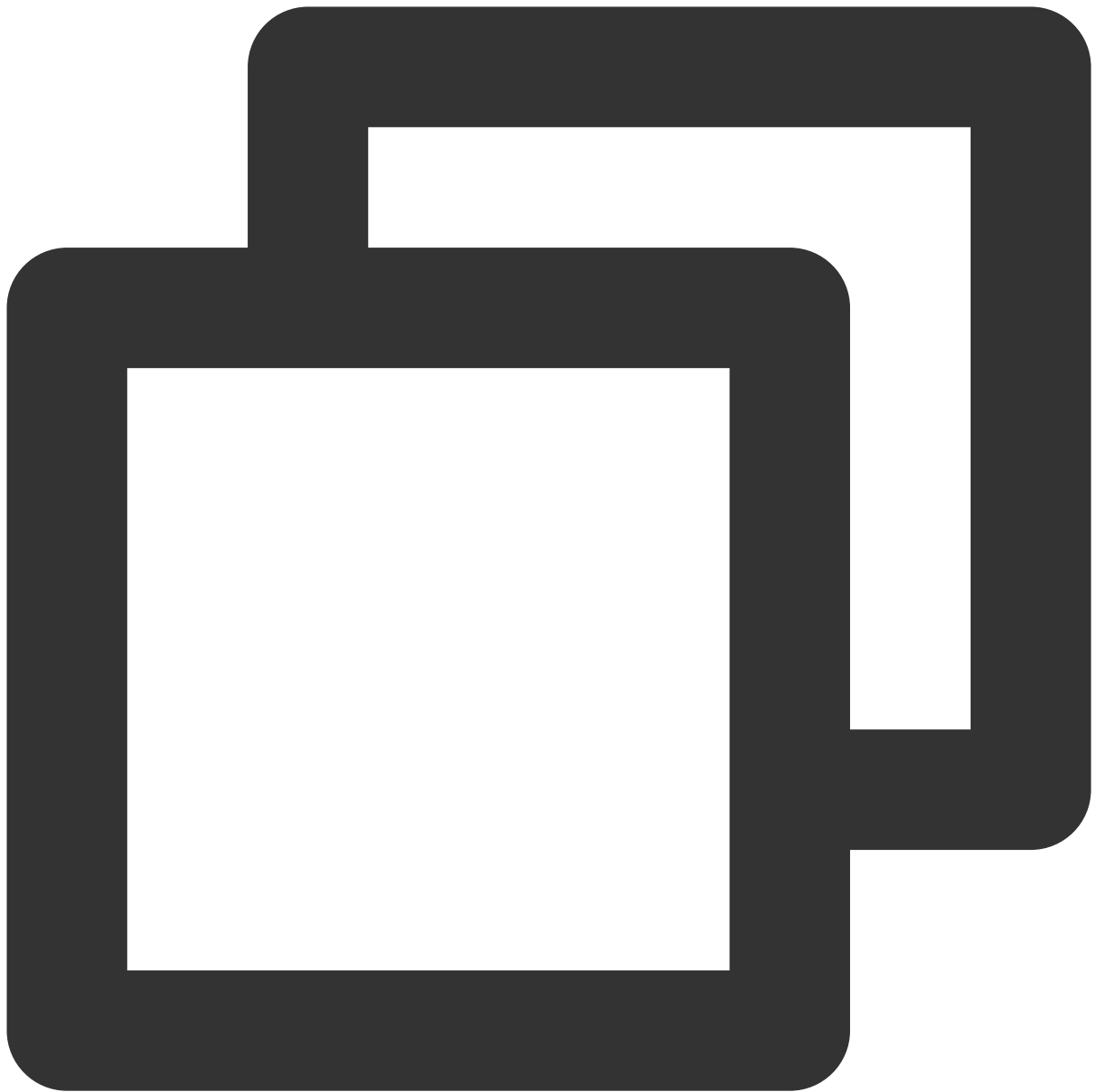
Start pushing Local Audio streams to Remote



```
void startPushLocalAudio();
```

### **stopPushLocalAudio**

Stop pushing Local Audio streams to Remote



```
void stopPushLocalAudio();
```

### **setRemoteVideoView**

Set Remote user Video Rendering View control



```
void setRemoteVideoView(String userId,
                        TUIRoomDefine.VideoStreamType streamType,
                        TUIVideoView view);
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIRoomDefine.VideoStreamType</a>	User streams type

view

TUIVideoView

Playback Remote user streams view

## startPlayRemoteVideo

Start Playback Remote user Video streams



```
void startPlayRemoteVideo(String userId,  
                           TUIRoomDefine.VideoStreamType streamType,  
                           TUIRoomDefine.PlayCallback callback);
```

### Parameters:



Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIRoomDefine.VideoStreamType</a>	User streams type
callback	TUIRoomDefine.PlayCallback	Playback Operation result Callback

## stopPlayRemoteVideo

Stop Playback Remote user Video streams



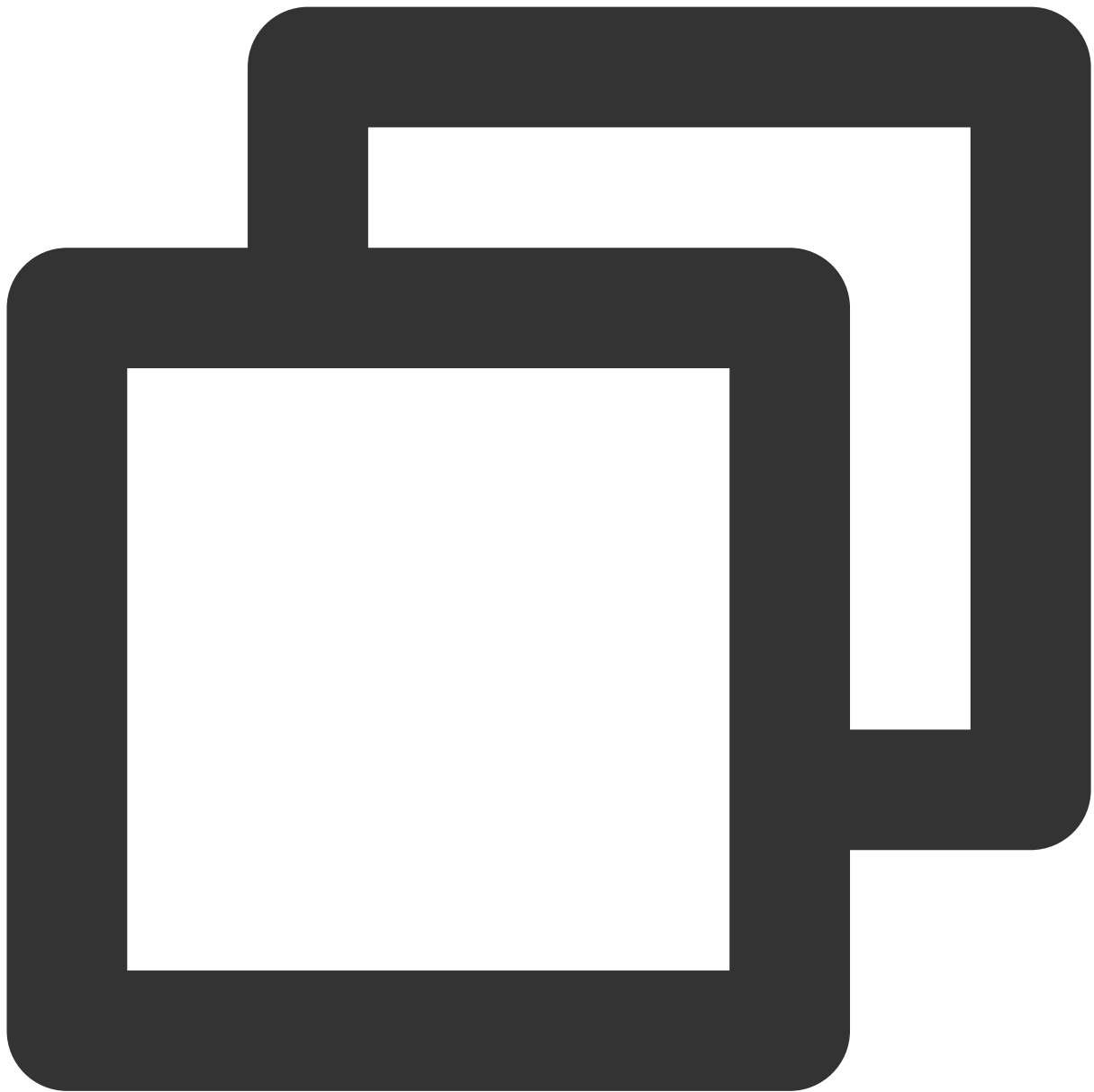
```
void stopPlayRemoteVideo(String userId, TUIRoomDefine.VideoStreamType streamType);
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIRoomDefine.VideoStreamType</a>	User streams type

**muteRemoteAudioStream**

Mute Remote user



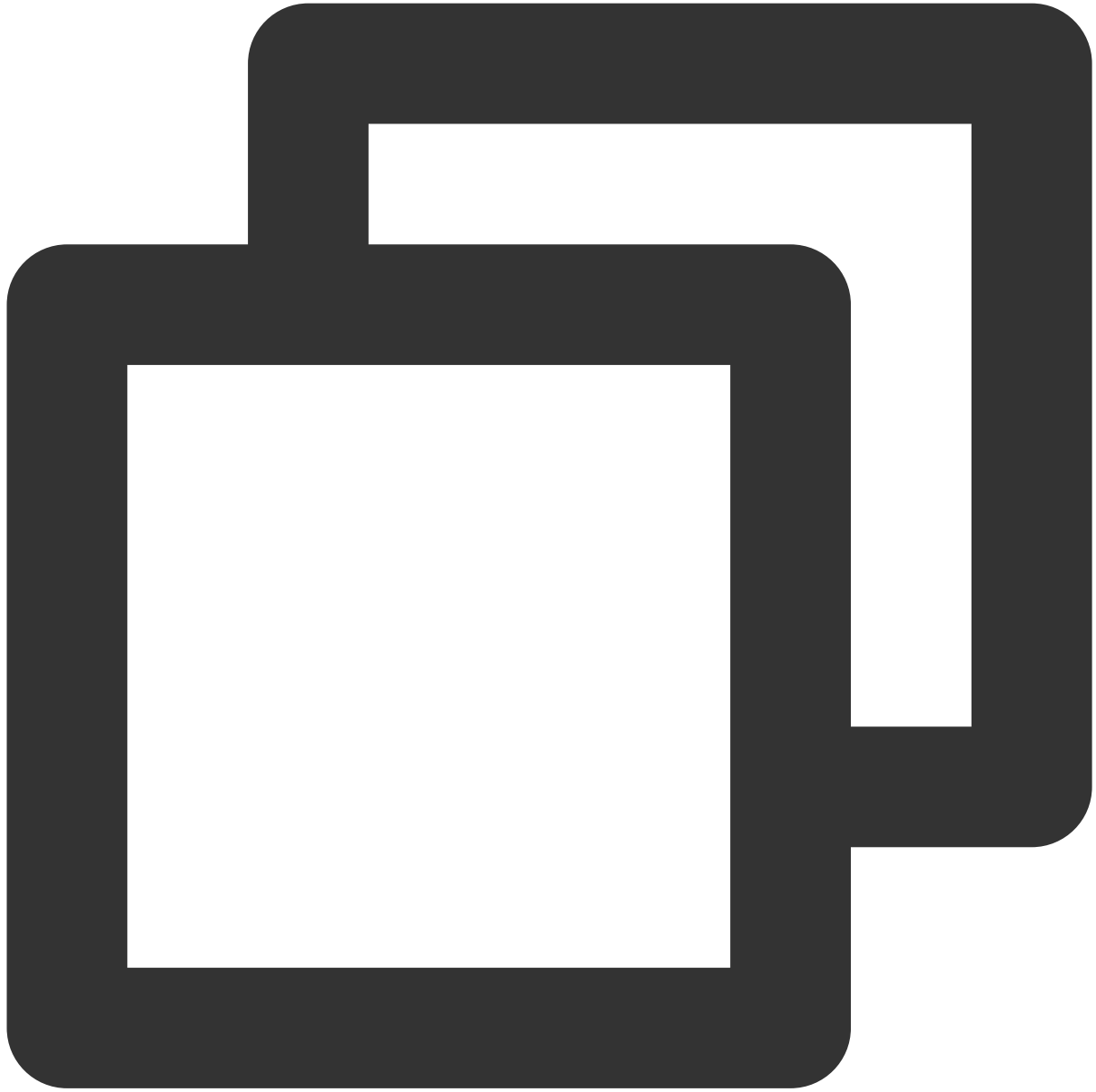
```
void muteRemoteAudioStream(String userId, boolean isMute);
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
isMute	boolean	Whether to mute

**getUserList**

Get current User list in the room, Note that the maximum number of User list fetched by this Interface is 100



```
void getUserList(long nextSequence, TUIRoomDefine.GetUserListCallback callback);
```

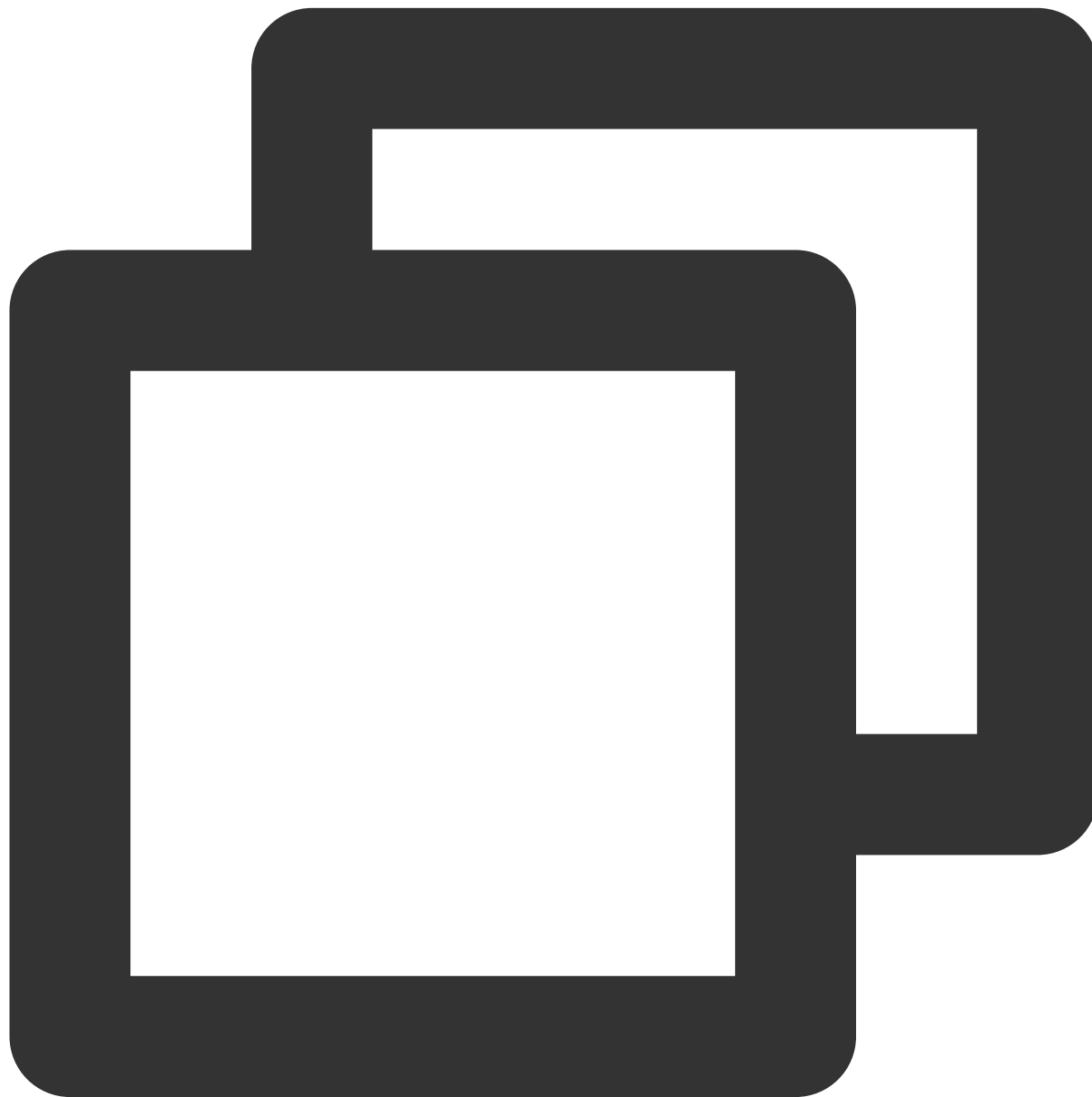
**Parameters:**

Parameter	Type	Meaning
nextSequence	long	Pagination Fetch Flag, fill in 0 for the first Fetch, if the nextSeq in the Callback is not 0, you need to do Pagination, pass in the nextSeq to Fetch again until the nextSeq in the Callback is 0

callback	TUIRoomDefine.GetUserListCallback	Get User list Callback
----------	-----------------------------------	------------------------

## getUserInfo

Get User [Learn more](#)



```
void getUserInfo(String userId, TUIRoomEngineDef.GetUserInfoCallback callback);
```

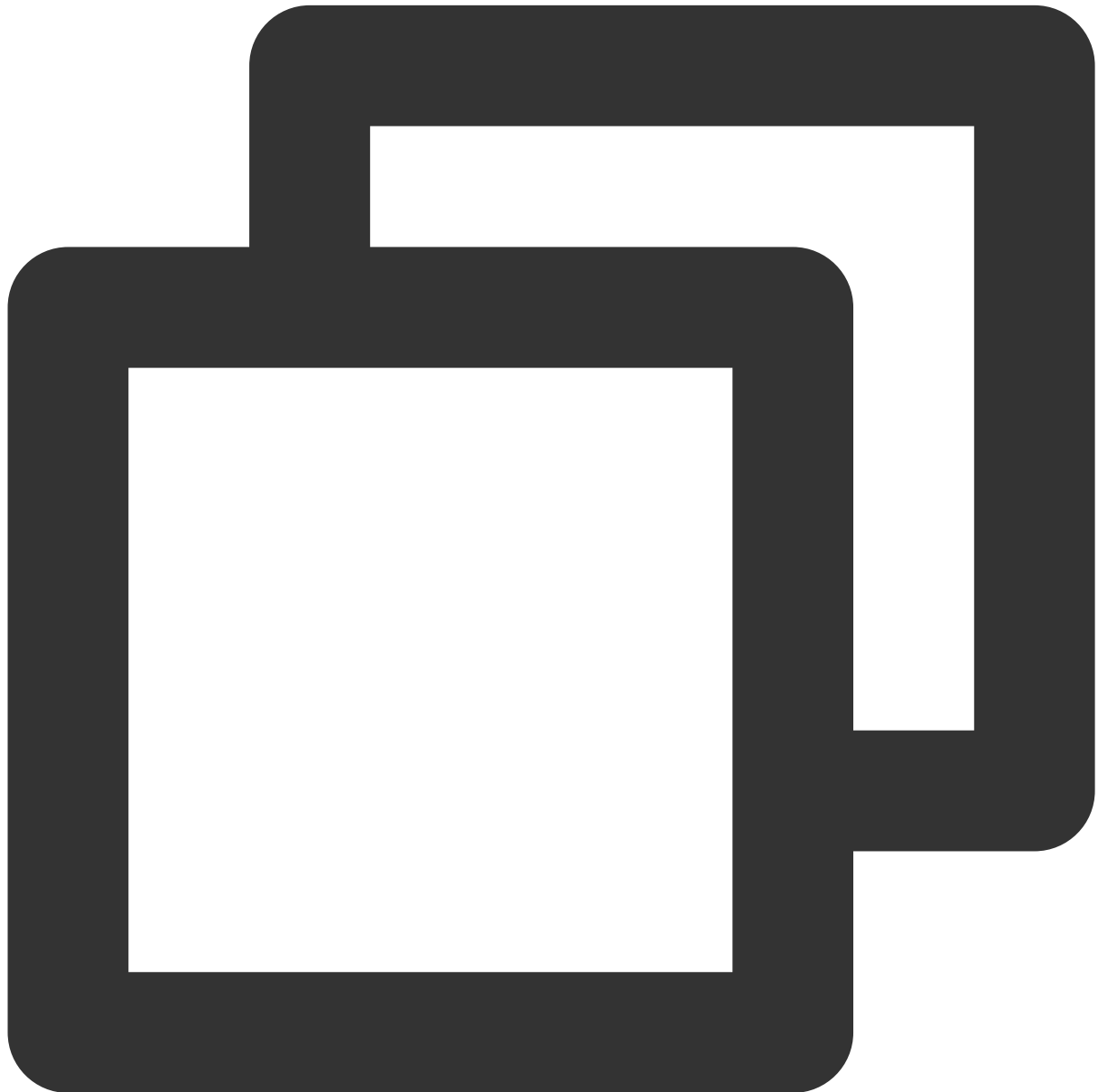
### Parameters:

Parameter	Type	Meaning
-----------	------	---------

userId	String	Get Learn more of the user by userId
callback	TUIRoomDefine.GetUserInfoCallback	Get User Learn more Callback

## changeUserRole

Change user Role, only Administrator or Group owner can call



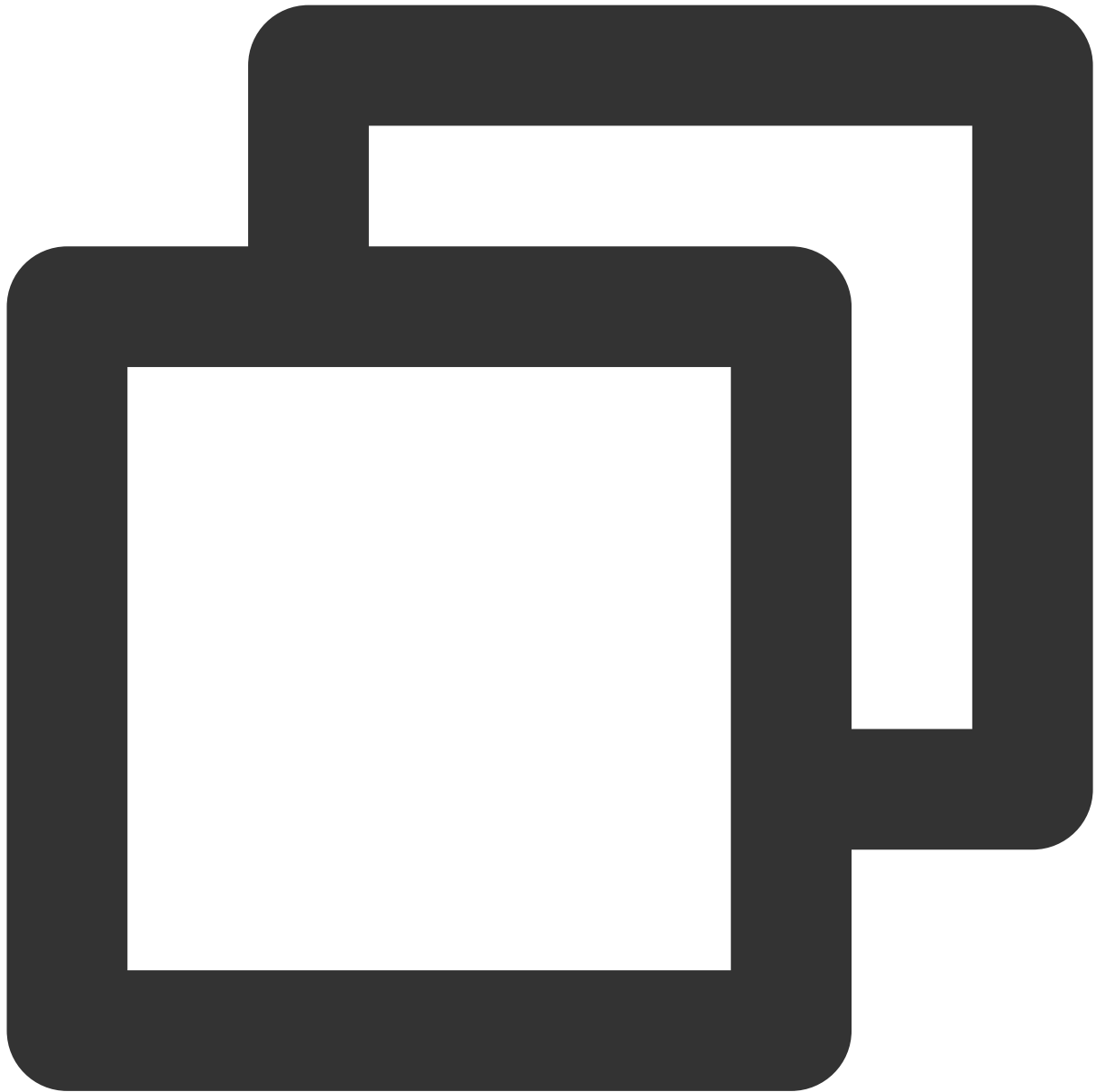
```
void changeUserRole(String userId,  
                    TUIRoomDefine.Role role,  
                    TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
role	<a href="#">TUIRoomDefine.Role</a>	User Role
callback	TUIRoomDefine.ActionCallback	Change Role Operation result Callback

**kickRemoteUserOutOfRoom**

Kick user out of the room, only Administrator or Group owner can call



```
void kickRemoteUserOutOfRoom(String userId, TUIRoomDefine.ActionCallback callback);
```

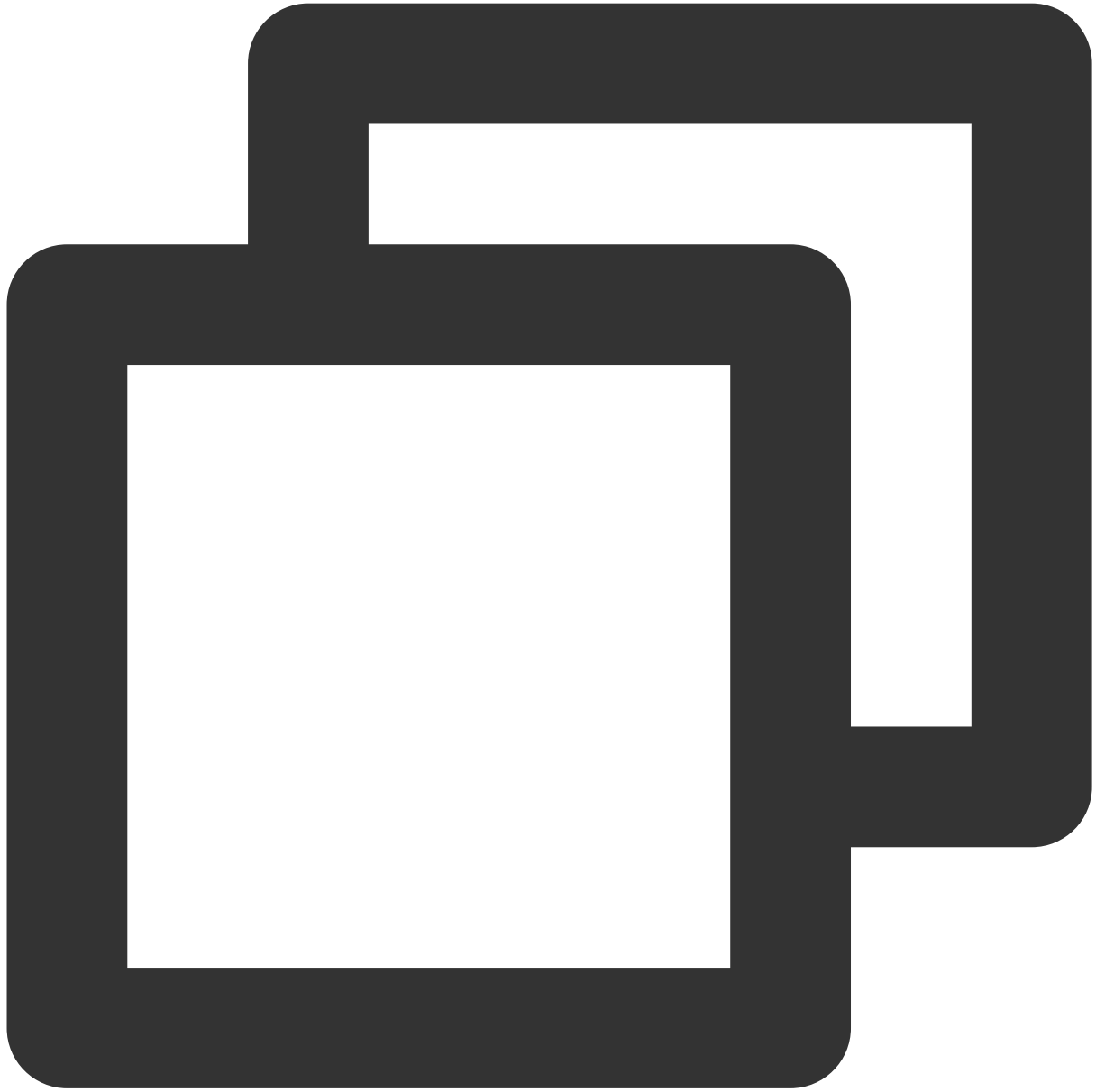
**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
callback	TUIRoomDefine.ActionCallback	Operation result Callback

**disableDeviceForAllUserByAdmin**



Manage Media device for all users, only Administrator or Group owner can call



```
void disableDeviceForAllUserByAdmin(TUIRoomDefine.MediaDevice device,  
                                     boolean isDisable,  
                                     TUIRoomDefine.ActionCallback callback);
```

Parameter	Type	Meaning
device	<a href="#">TUIRoomDefine.MediaDevice</a>	Device
isDisable	boolean	Whether to Disable

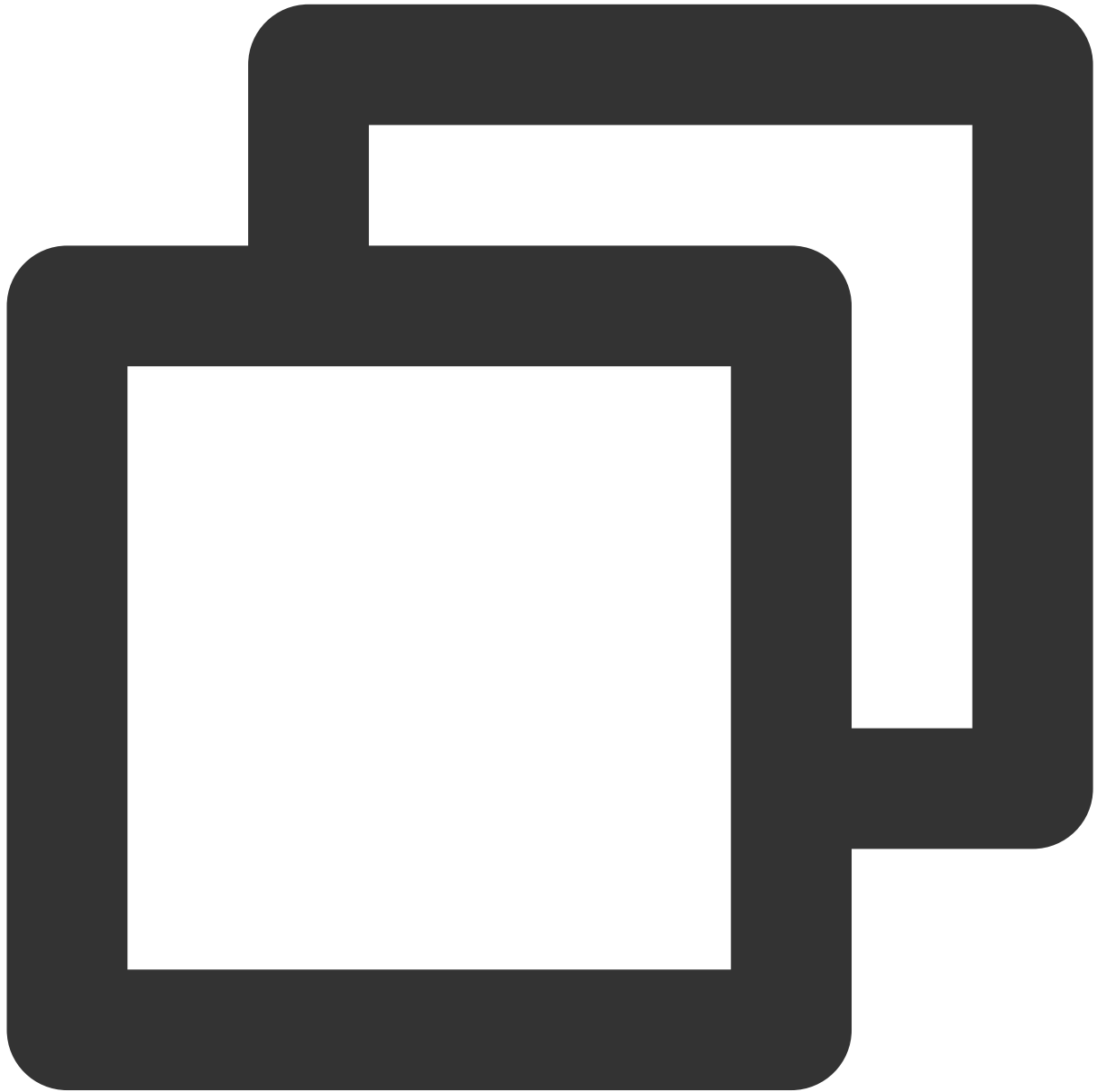
callback

TUIRoomDefine.ActionCallback

Operation result Callback

## openRemoteDeviceByAdmin

Request Remote user to open Media device, only Administrator or Group owner can call



```
TUIRoomDefine.Request openRemoteDeviceByAdmin(String userId,  
                                                TUIRoomDefine.MediaDevice device,  
                                                int timeout,  
                                                TUIRoomDefine.RequestCallback callbac
```

Parameter	Type	Meaning
userId	String	User ID
device	<a href="#">TUIRoomDefine.MediaDevice</a>	Device
timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
callback	TUIRoomDefine.ActionCallback	Operation result Callback

### **closeRemoteDeviceByAdmin**

Close Remote user Media device, only Administrator or Group owner can call

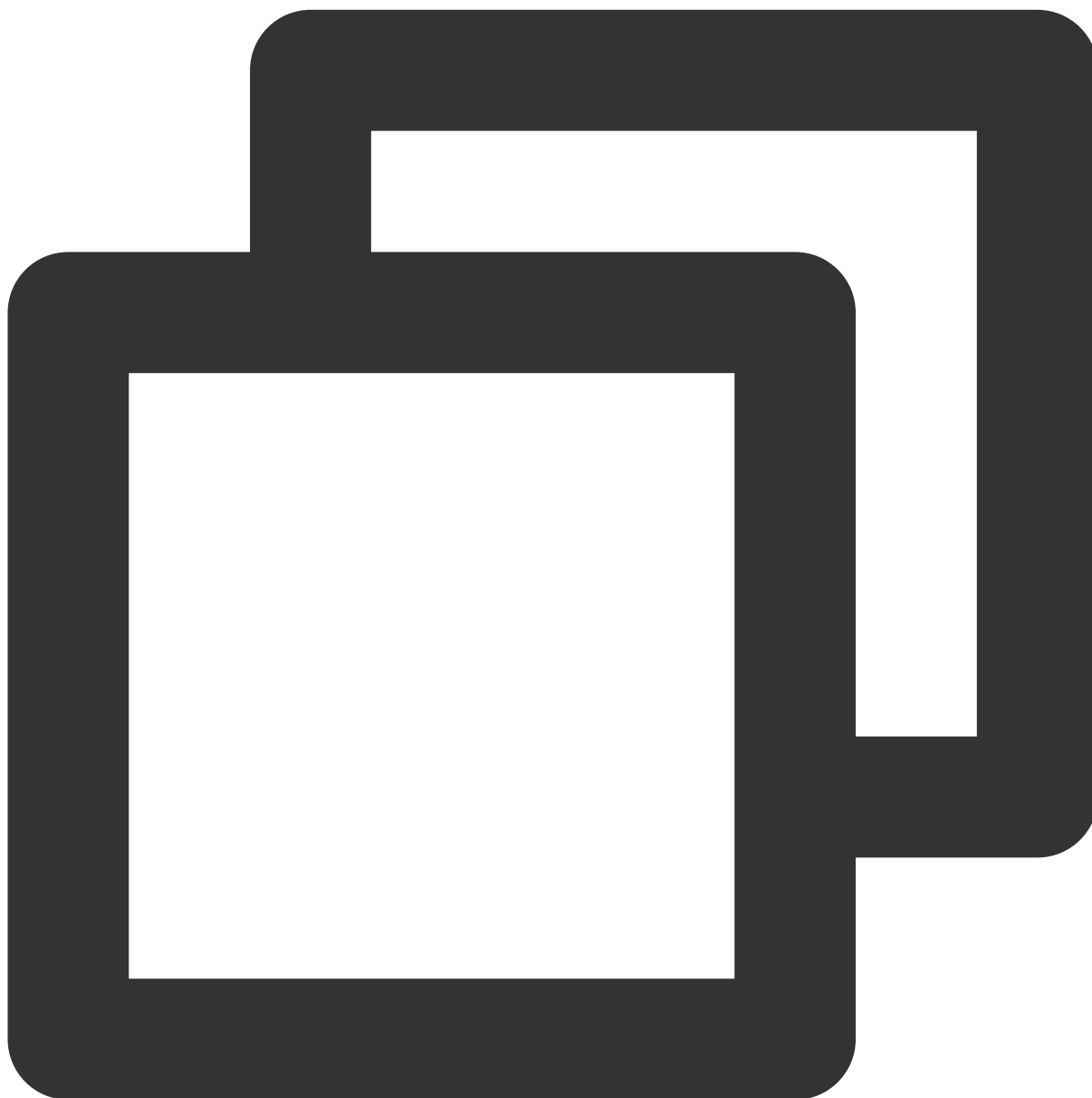


```
void closeRemoteDeviceByAdmin(String userId,  
                               TUIRoomDefine.MediaDevice device,  
                               TUIRoomDefine.ActionCallback callback);
```

Parameter	Type	Meaning
device	<a href="#">TUIRoomDefine.MediaDevice</a>	Device
isDisable	boolean	Whether to Disable
callback	TUIRoomDefine.ActionCallback	Operation result Callback

## applyToAdminToOpenLocalDevice

Lock all users' Media device management



```
TUIRoomDefine.Request applyToAdminToOpenLocalDevice (TUIRoomDefine.MediaDevice device,
                                                         int timeout,
                                                         TUIRoomDefine.RequestCallback callback)
```

Parameter	Type	Meaning

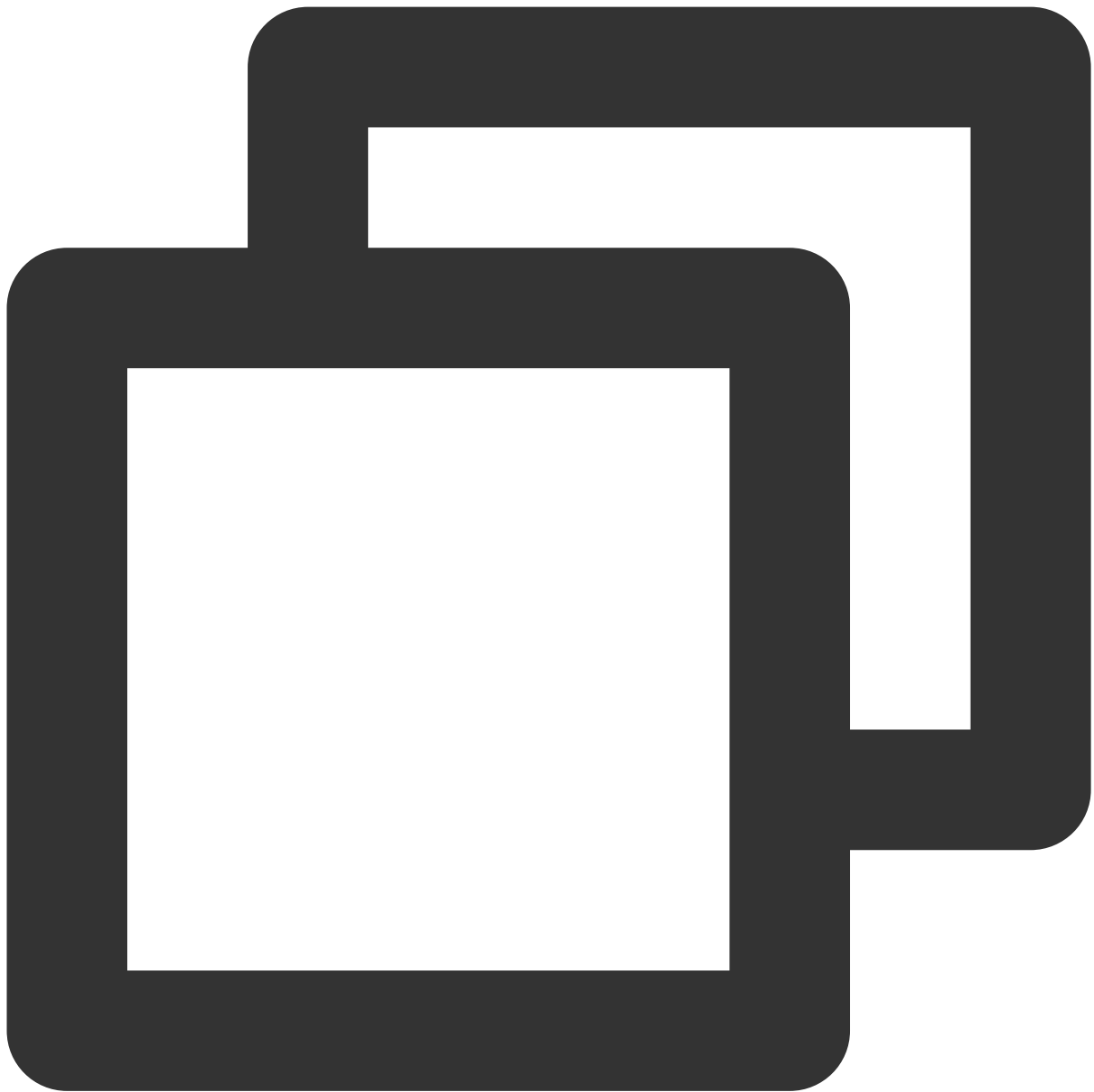
device	<a href="#">TUIRoomDefine.MediaDevice</a>	Device
timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
callback	TUIRoomDefine.ActionCallback	Operation result Callback

## setMaxSeatCount

Set Maximum number of seats, only supported when entering the room and creating the room

When roomType is RoomType.CONFERENCE (Education and Conference scene), maxSeatCount value is not limited;

When roomType is RoomType.LIVE\_ROOM (Live broadcast scene), maxSeatCount is limited to 16;

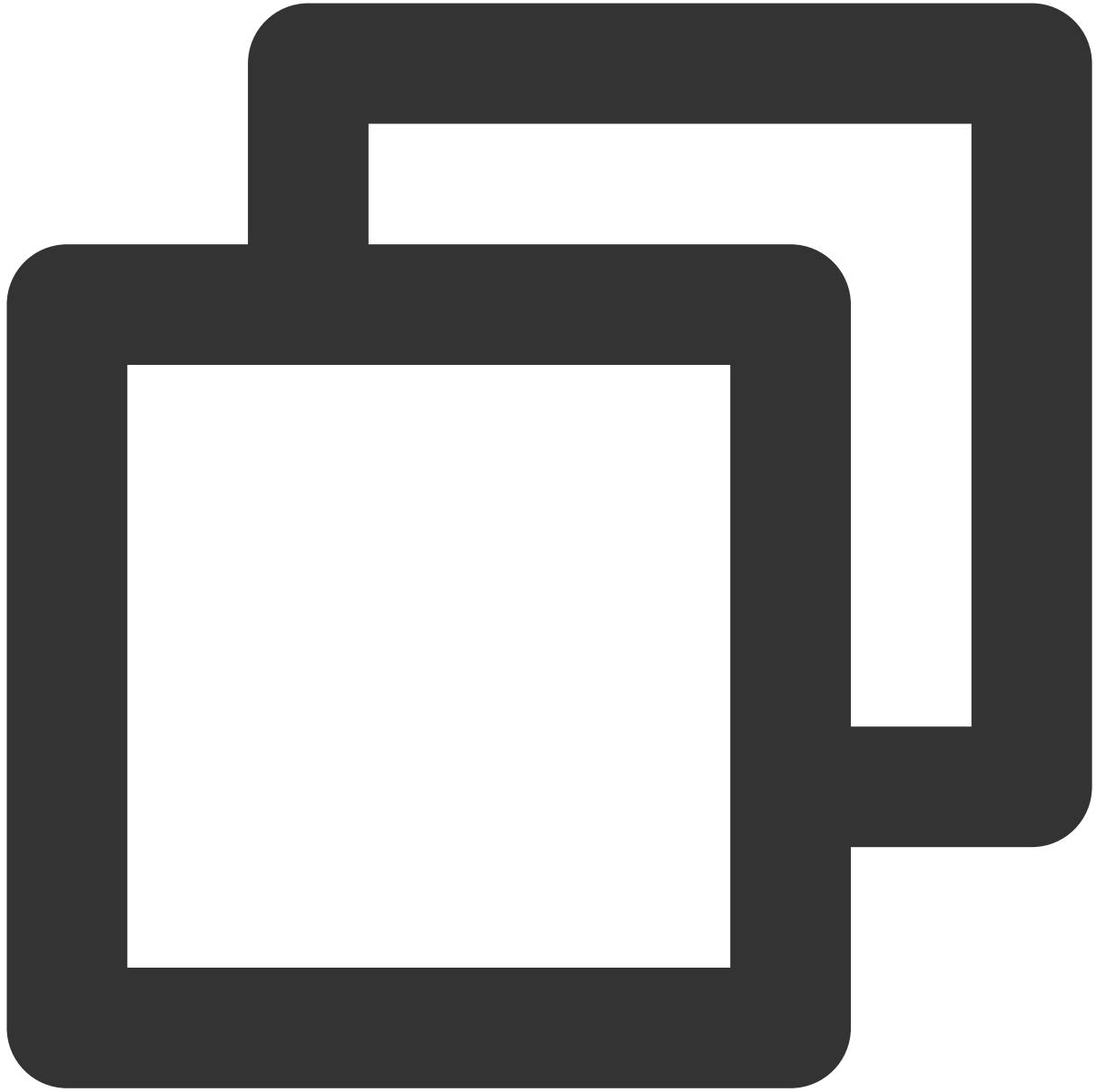


```
void setMaxSeatCount(int maxSeatCount, TUIRoomDefine.ActionCallback callback);
```

Parameter	Type	Meaning
maxSeatCount	int	Maximum number of seats
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

### lockSeatByAdmin

Lock seat (including position lock, audio status lock, video status lock)



```
void lockSeatByAdmin(int seatIndex,  
                    TUIRoomDefine.SeatLockParams lockParams,  
                    TUIRoomDefine.ActionCallback callback);
```

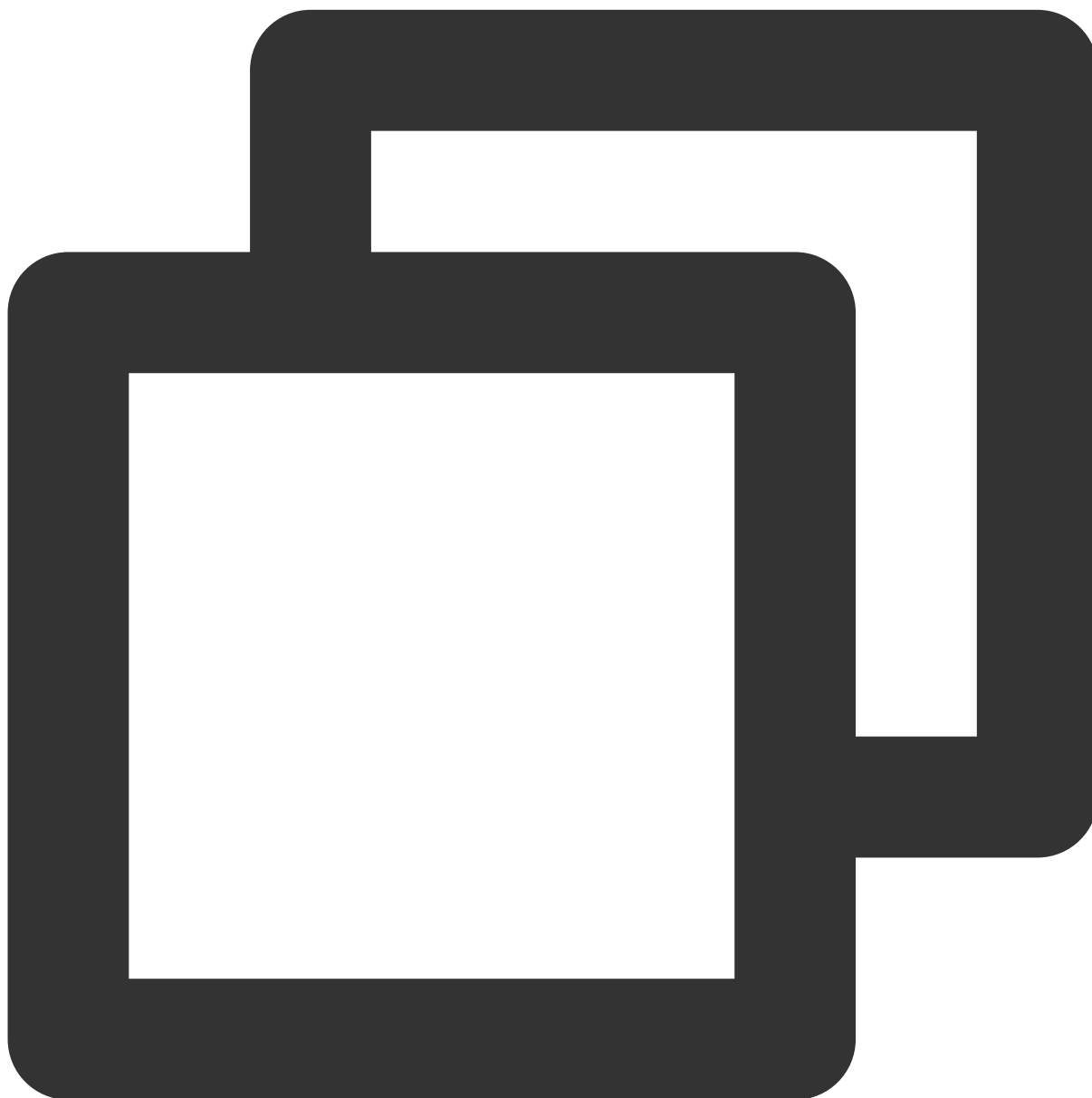
Parameter	Type	Meaning
seatIndex	int	Seat number
lockParams	<a href="#">TUIRoomDefine.SeatLockParams</a>	Lock microphone parameter



callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call
----------	------------------------------	--

## getSeatList

Get seat list



```
void getSeatList(TUIRoomDefine.GetSeatListCallback callback);
```

Parameter	Type	Meaning

callback	TUIRoomDefine.GetSeatListCallback	Get seat list Callback
----------	-----------------------------------	------------------------

## takeSeat

Local on microphone

### Explanation:

Conference scene: [SPEAK\\_AFTER\\_TAKING\\_SEAT](#) mode requires an application to the host or administrator to allow on microphone, other modes do not support on microphone.

Live broadcast scene: [FREE\\_TO\\_SPEAK](#) mode can freely on microphone, after on microphone, open microphone to speak; [SPEAK\\_AFTER\\_TAKING\\_SEAT](#) mode requires an application to the host or administrator to allow on microphone; other modes do not support on microphone.



```
TUIRoomDefine.Request takeSeat(int seatIndex,  
                                int timeout,  
                                TUIRoomDefine.RequestCallback callback);
```

**Parameters:**

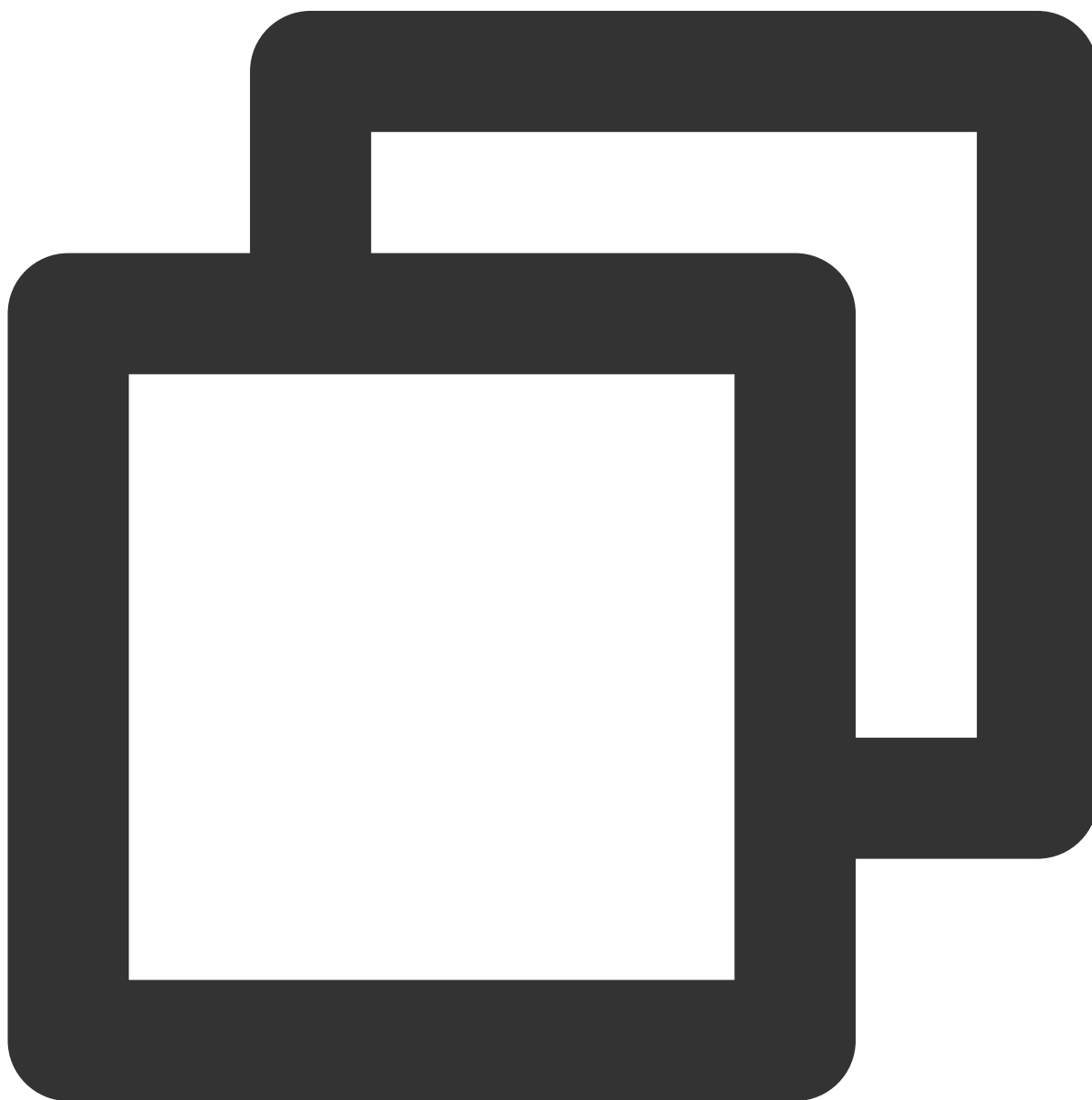
Parameter	Type	Meaning
seatIndex	int	Seat number
timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout

		detection and will not trigger timeout Callback
callback	TUIRoomDefine.RequestCallback	Call interface Callback, used to notify the request Callback status

**Return:** Request body

## leaveSeat

Local off microphone



```
void leaveSeat(TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

**takeUserOnSeatByAdmin**

Host/Administrator Invite User on microphone



```
TUIRoomDefine.Request takeUserOnSeatByAdmin(int seatIndex,  
                                              String userId,  
                                              int timeout,  
                                              TUIRoomDefine.RequestCallback callback)
```

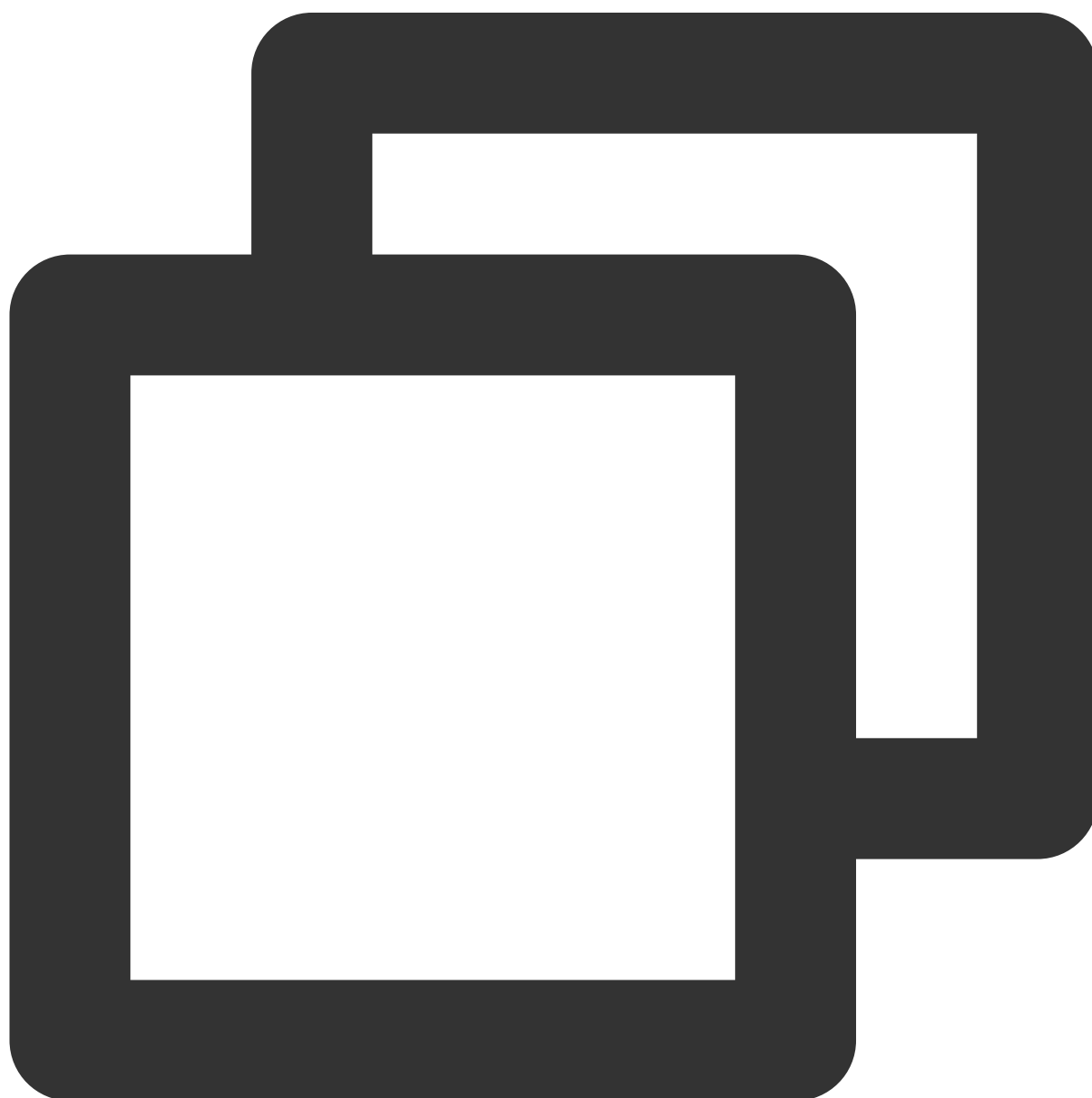
Parameter	Type	Meaning
seatIndex	int	Seat number
userId	String	User ID

timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
callback	TUIRoomDefine.RequestCallback	Call interface Callback, used to notify the request Callback status

**Return:** Request body

### kickUserOffSeatByAdmin

Host/Administrator Take User off microphone



```
void kickUserOffSeatByAdmin(int seatIndex, String userId, TUIRoomDefine.ActionCallb
```

Parameter	Type	Meaning
seatIndex	int	Seat number
userId	String	User ID
callback	TUIRoomDefine.RequestCallback	Call interface Callback, used to notify the request Callback status

## sendTextMessage

Send Text message





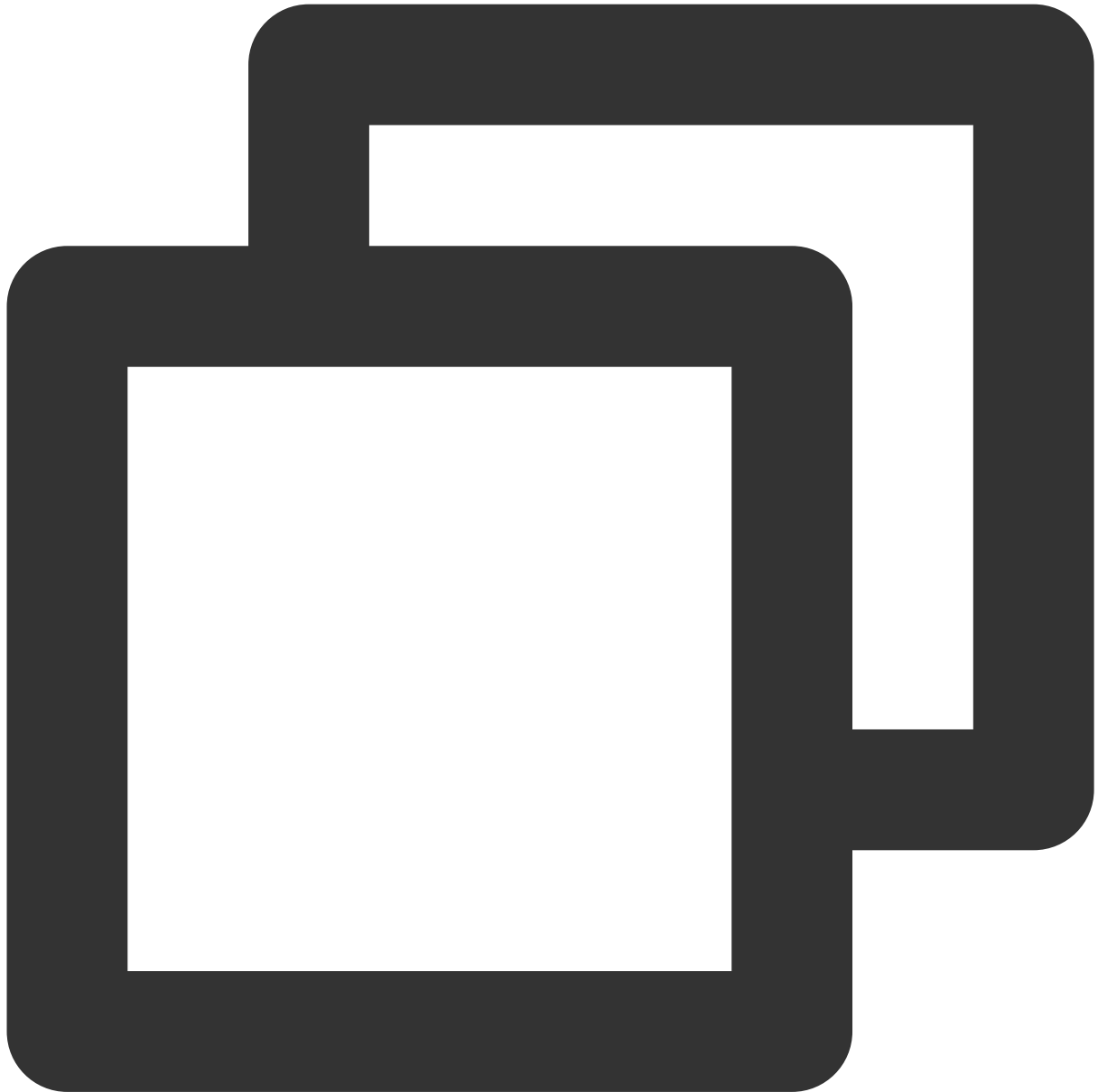
```
void sendTextMessage(String message,  
                     TUIRoomDefine.ActionCallback callback);
```

**Parameters:**

Parameter	Type	Meaning
message	String	Text message Content
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

## sendCustomMessage

Send Custom message



```
void sendCustomMessage(String message,  
                        TUIRoomDefine.ActionCallback callback);
```

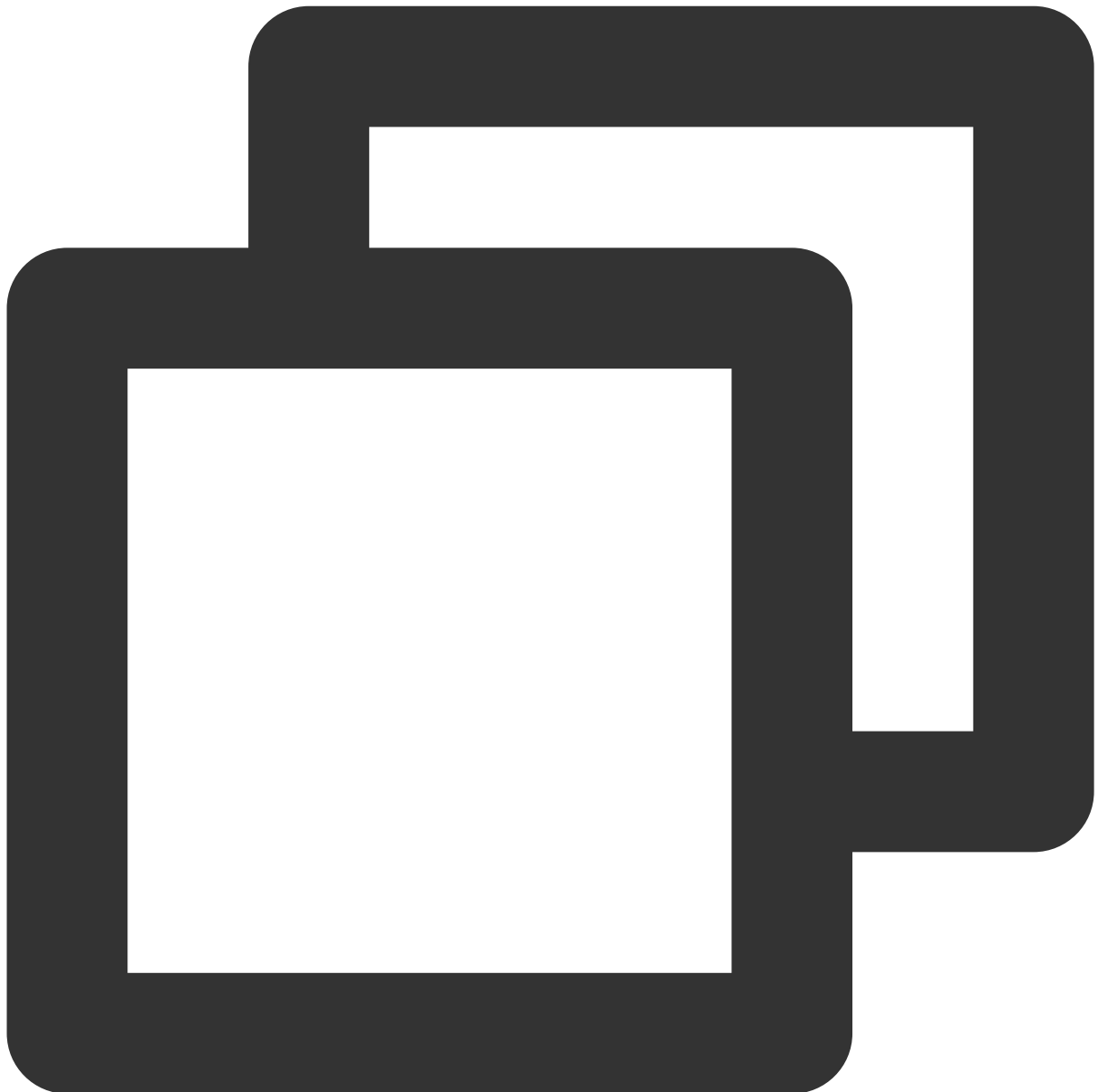
### Parameters:

Parameter	Type	Meaning
message	String	Custom message Content

callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call
----------	------------------------------	--

### **disableSendingMessageByAdmin**

Disable Remote user's Text message sending Ability (only Administrator or Group owner can call)

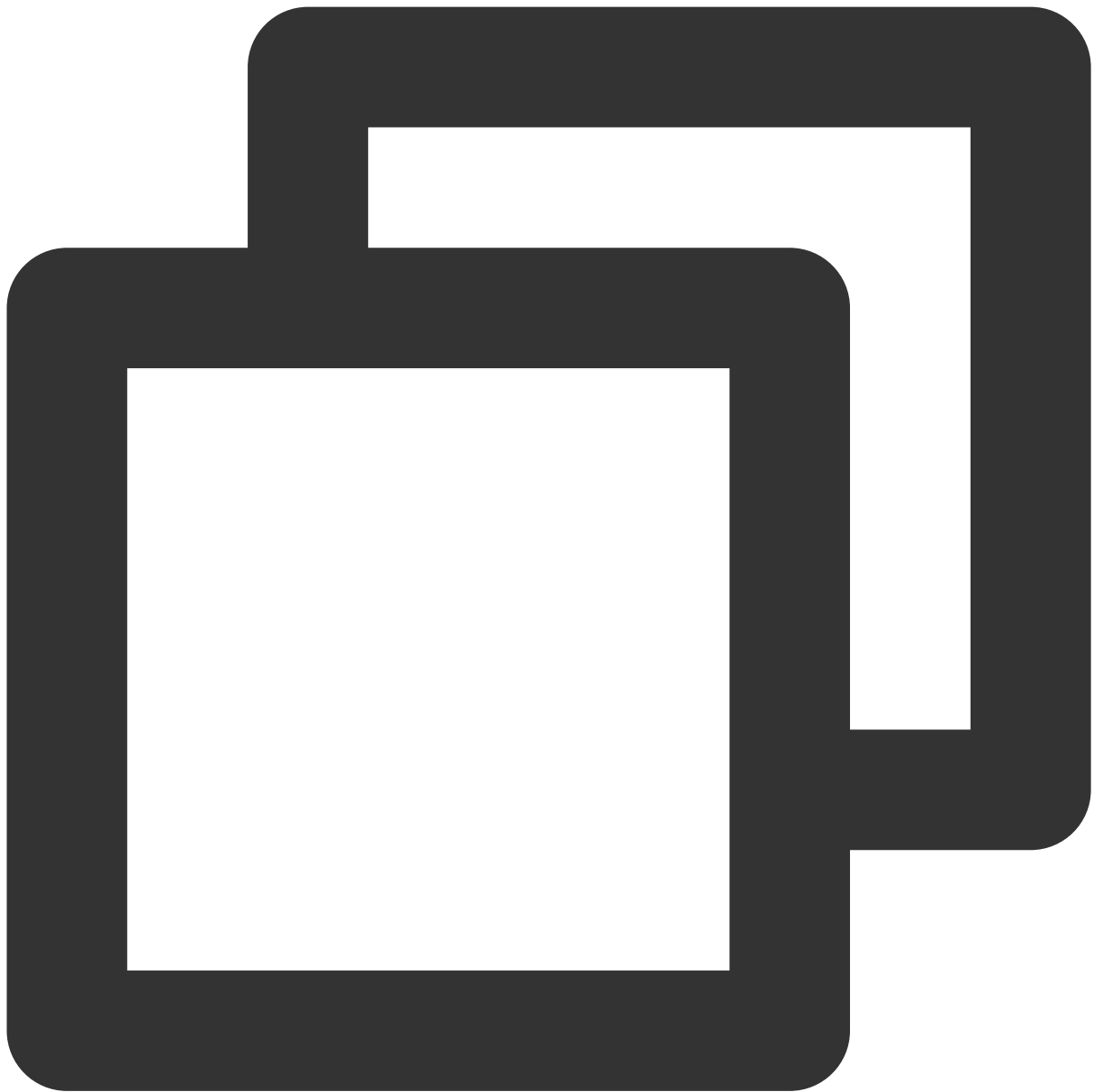


```
void disableSendingMessageByAdmin(String userId,  
                                   boolean isDisable,  
                                   TUIRoomDefine.ActionCallback callback);
```

Parameter	Type	Meaning
userId	String	User ID
isDisable	boolean	Whether to Disable
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

### **disableSendingMessageForAllUser**

Disable all users' Text message sending Ability (only Administrator or Group owner can call)

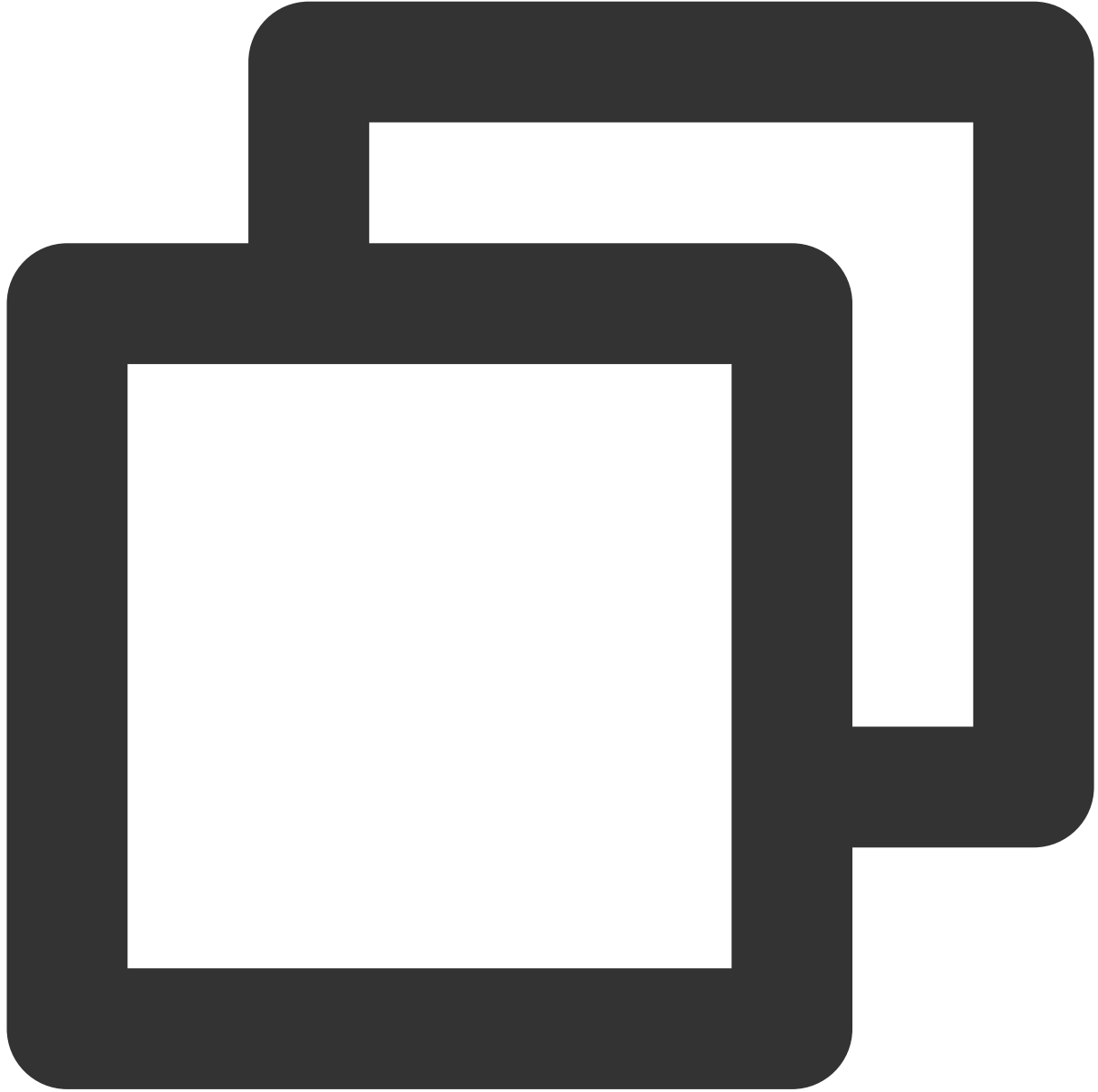


```
void disableSendingMessageForAllUser(boolean isDisable,  
                                     TUIRoomDefine.ActionCallback callback);
```

Parameter	Type	Meaning
isDisable	boolean	Whether to Disable
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

## cancelRequest

Cancel sent Request



```
void cancelRequest(String requestId, TUIRoomDefine.ActionCallback callback);
```

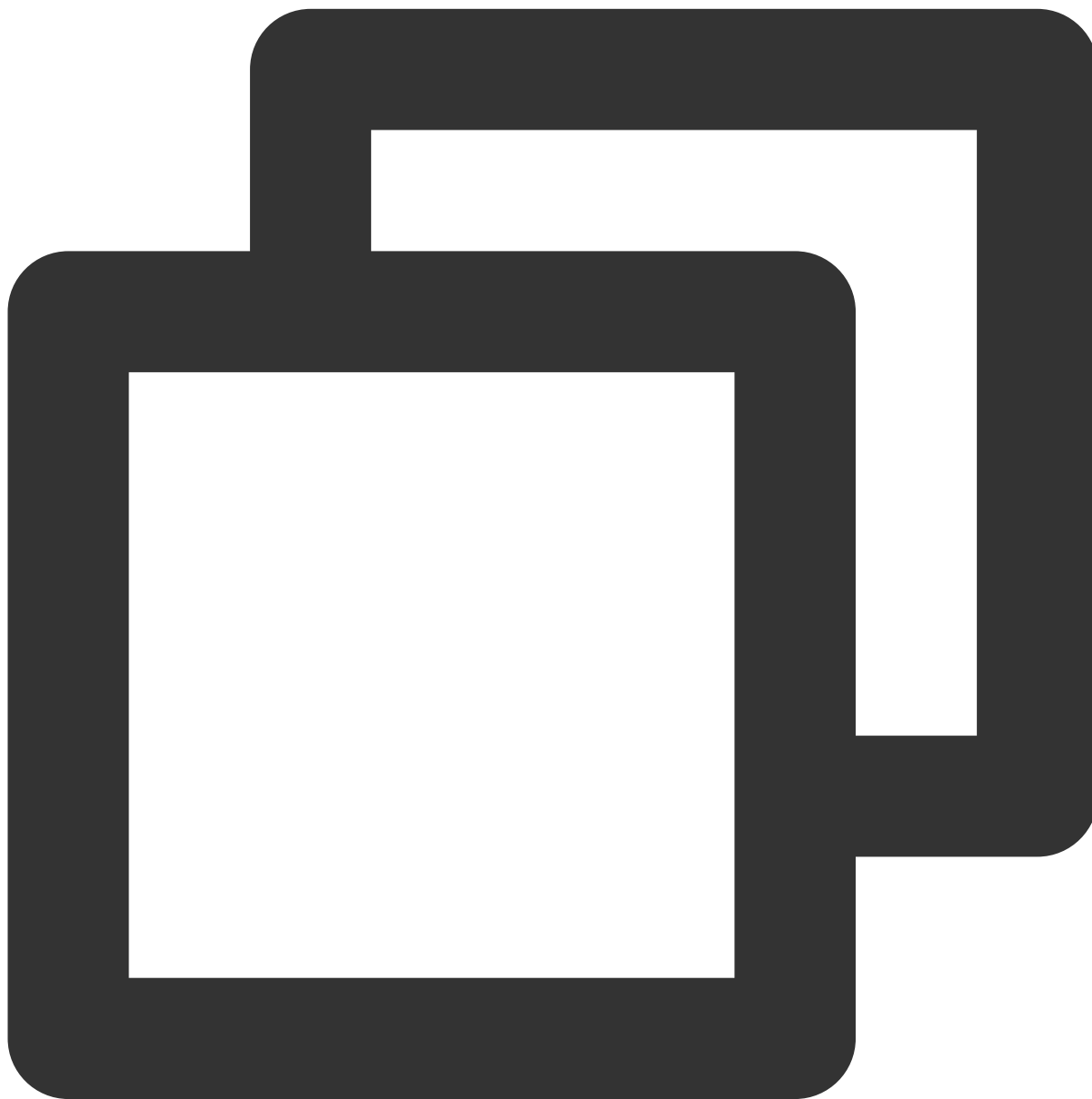
### Parameters:

Parameter	Type	Meaning
requestId	String	Request ID
callback	TUIRoomDefine.ActionCallback	Callback of API, used to Notify the Success

		or Failure of the API call
--	--	----------------------------

## responseRemoteRequest

Reply to Remote user's Request



```
void responseRemoteRequest(String requestId,  
                           boolean agree,  
                           TUIRoomDefine.ActionCallback callback);
```

### Parameters:

--	--	--

Parameter	Type	Meaning
requestId	String	Request ID
agree	boolean	Whether to agree
callback	TUIRoomEngineDef.ActionCallback	Callback of API, used to Notify the Success or Failure of the API call

## getTRTCCloud

Get TRTCCloud Instance





```
TRTCCloud getTRTCCloud();
```

## getDeviceManager

Get deviceManager, you can use deviceManager's method to get Device list, even Device, switch Device and other functions.



```
TXDeviceManager getDeviceManager();
```

## **getAudioEffectManager**

Get Audio management



```
TXAudioEffectManager getAudioEffectManager();
```

## **getBeautyManager**

Get Beauty management object



```
TXBeautyManager getBeautyManager();
```

# TUIRoomObserver

Last updated : 2023-10-24 17:07:24

## TUIRoomEngine Event Callback

### **onError**

Error Event Callback

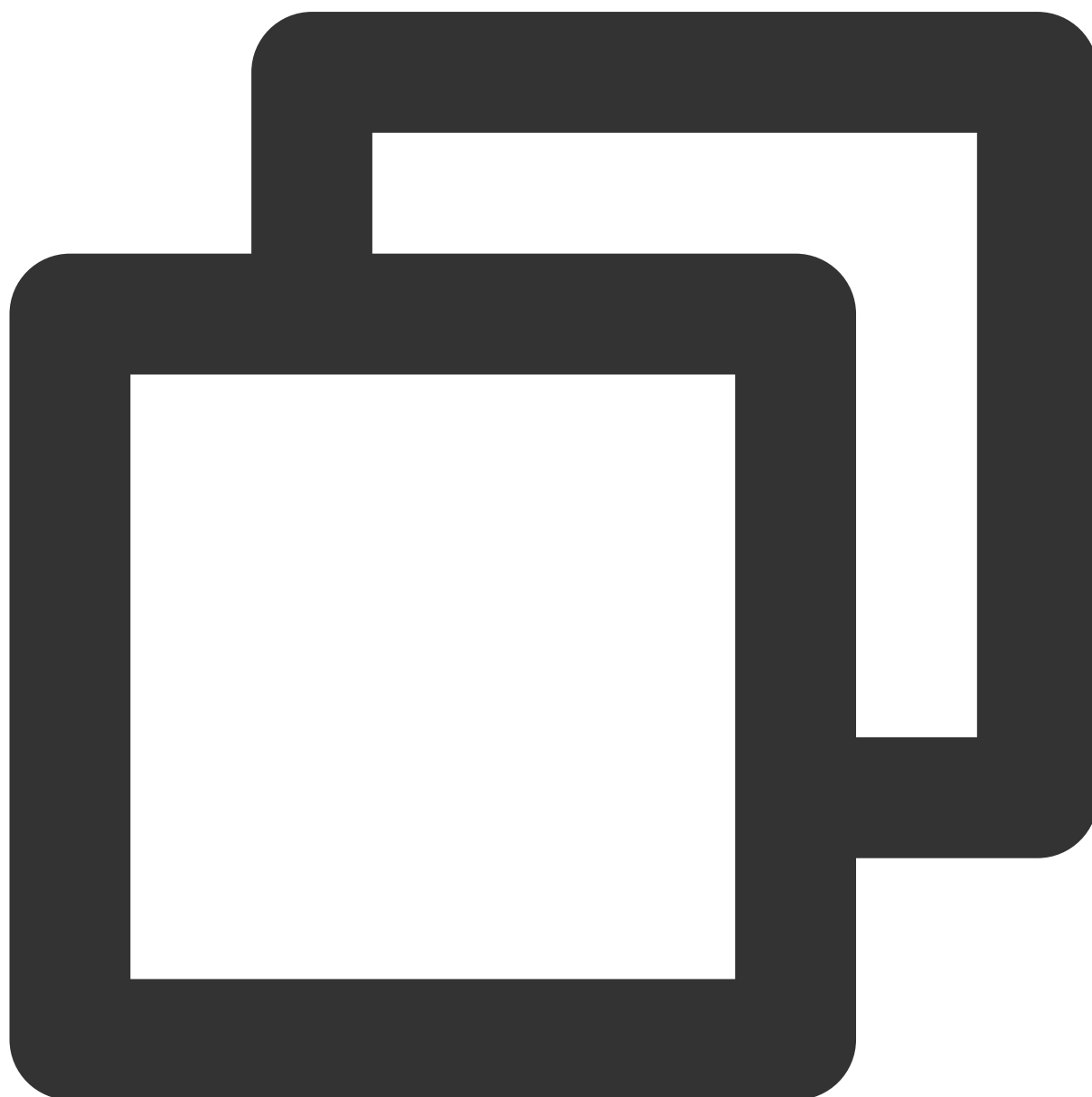


```
void onError(TUICommonDefine.Error errorCode, String message)
```

Parameter	Type	Description
errorCode	TUICommonDefine.Error	Error Code
message	String	Error Message

## onKickedOffLine

Other terminals login and get kicked off event.

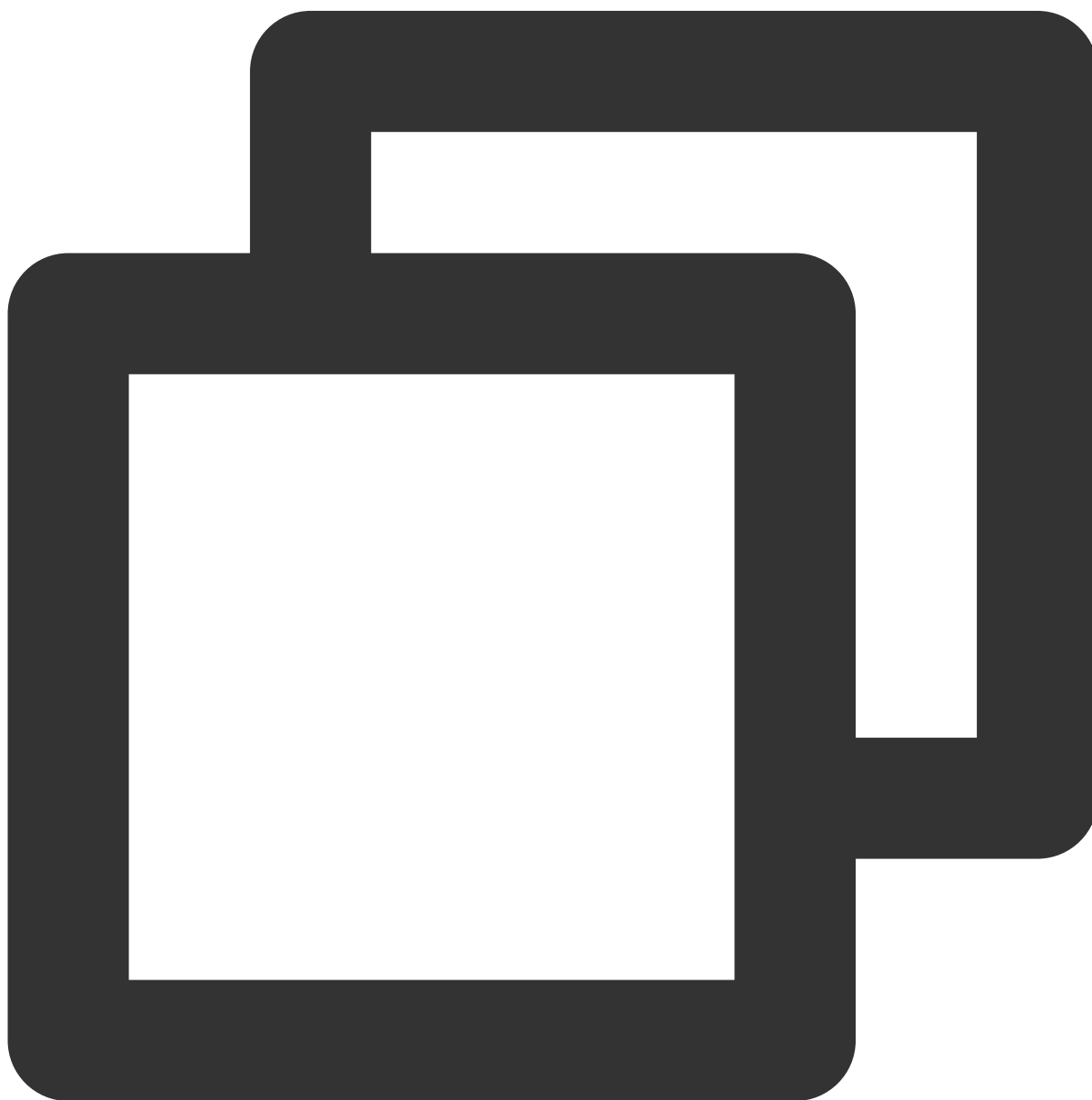


```
void onKickedOffLine(String message)
```

Parameter	Type	Description
message	String	Kicked out description

## onUserSigExpired

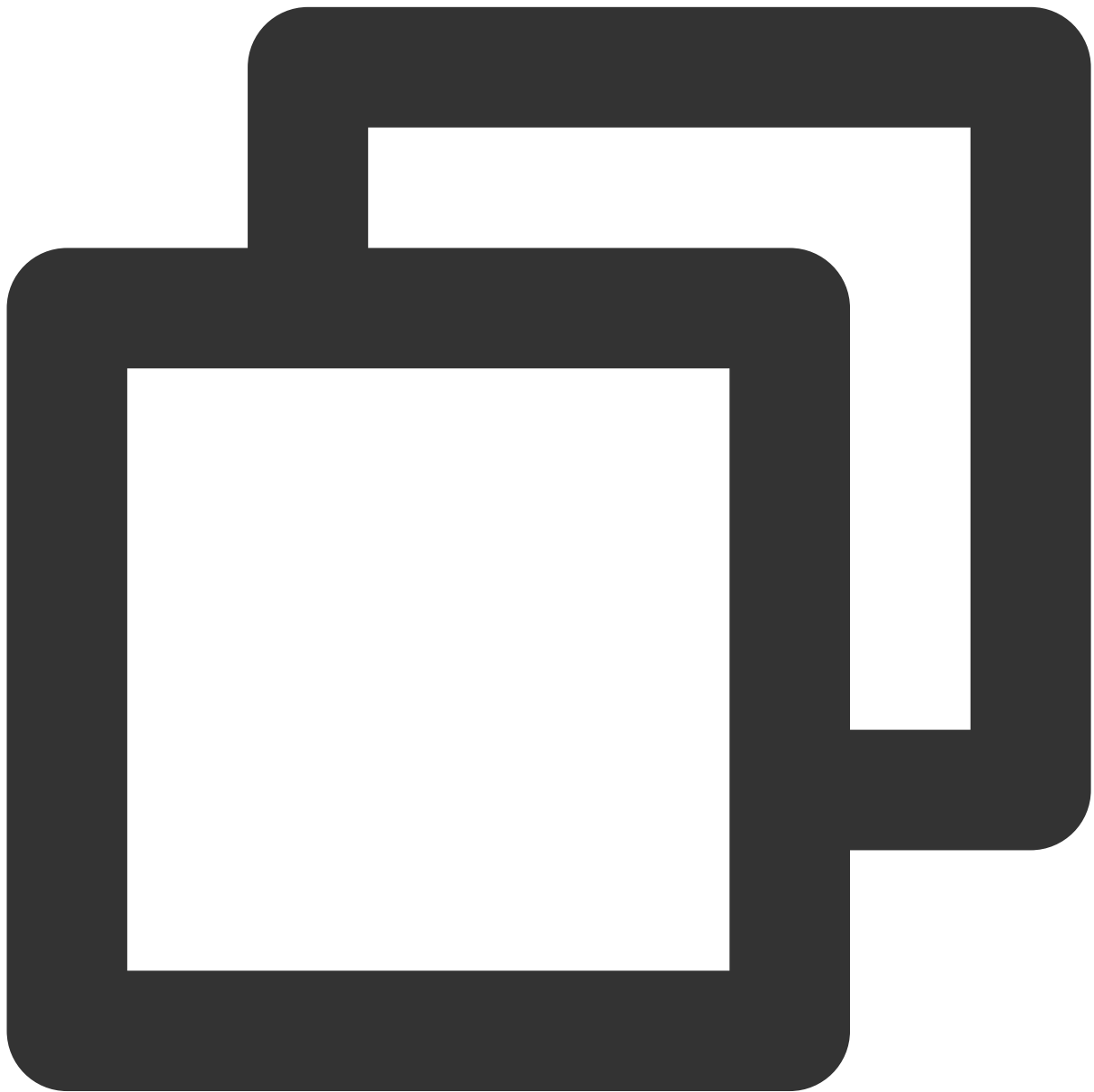
User credential timeout event.



```
void onUserSigExpired()
```

## onRoomNameChanged

Room name change event.



```
void onRoomNameChanged(String roomId, String roomName)
```

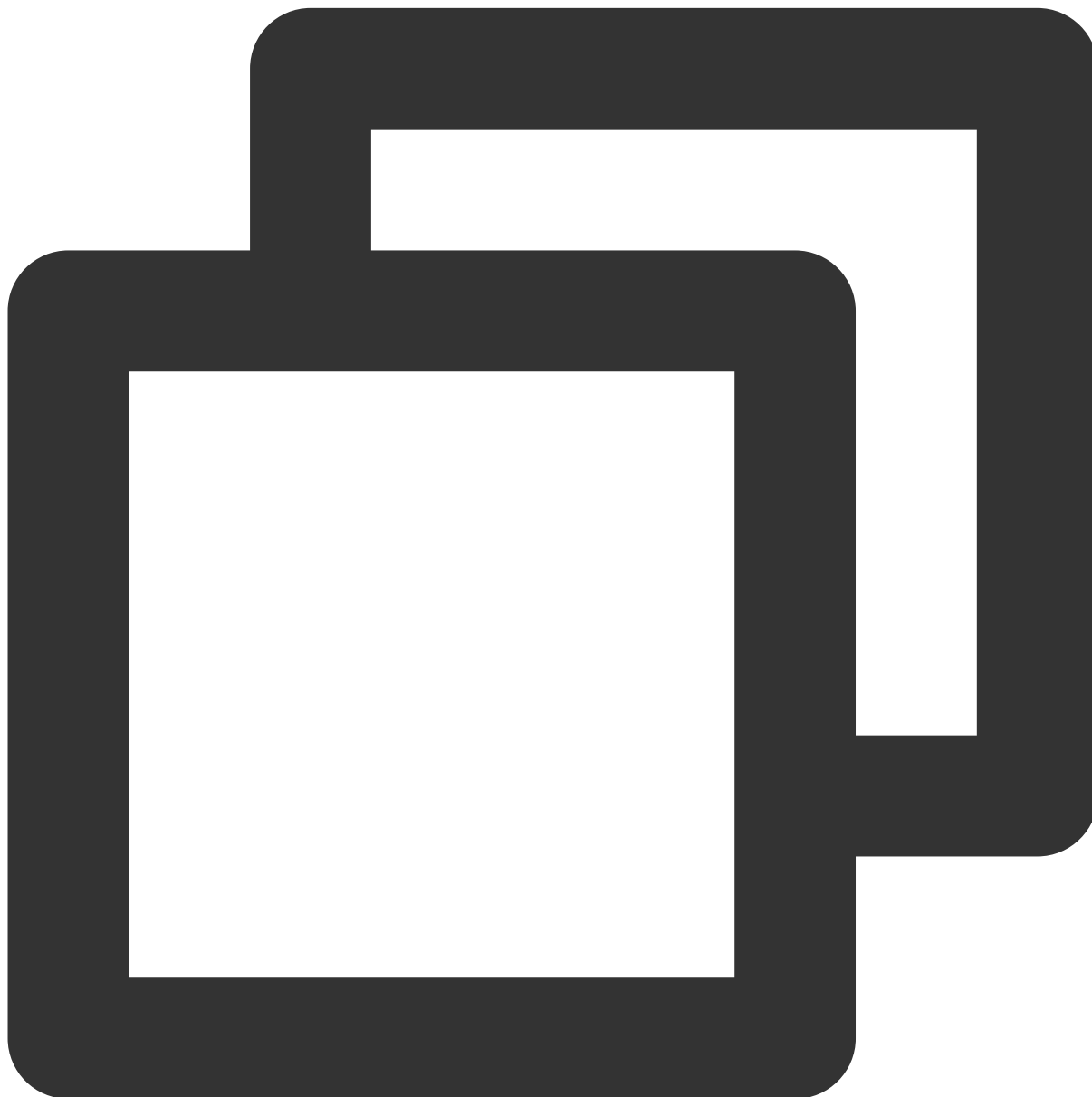
Parameter	Type	Description



roomId	String	Room ID
roomName	String	Room Name

### onAllUserMicrophoneDisableChanged

Inside the room, all users' mic is disabled event.



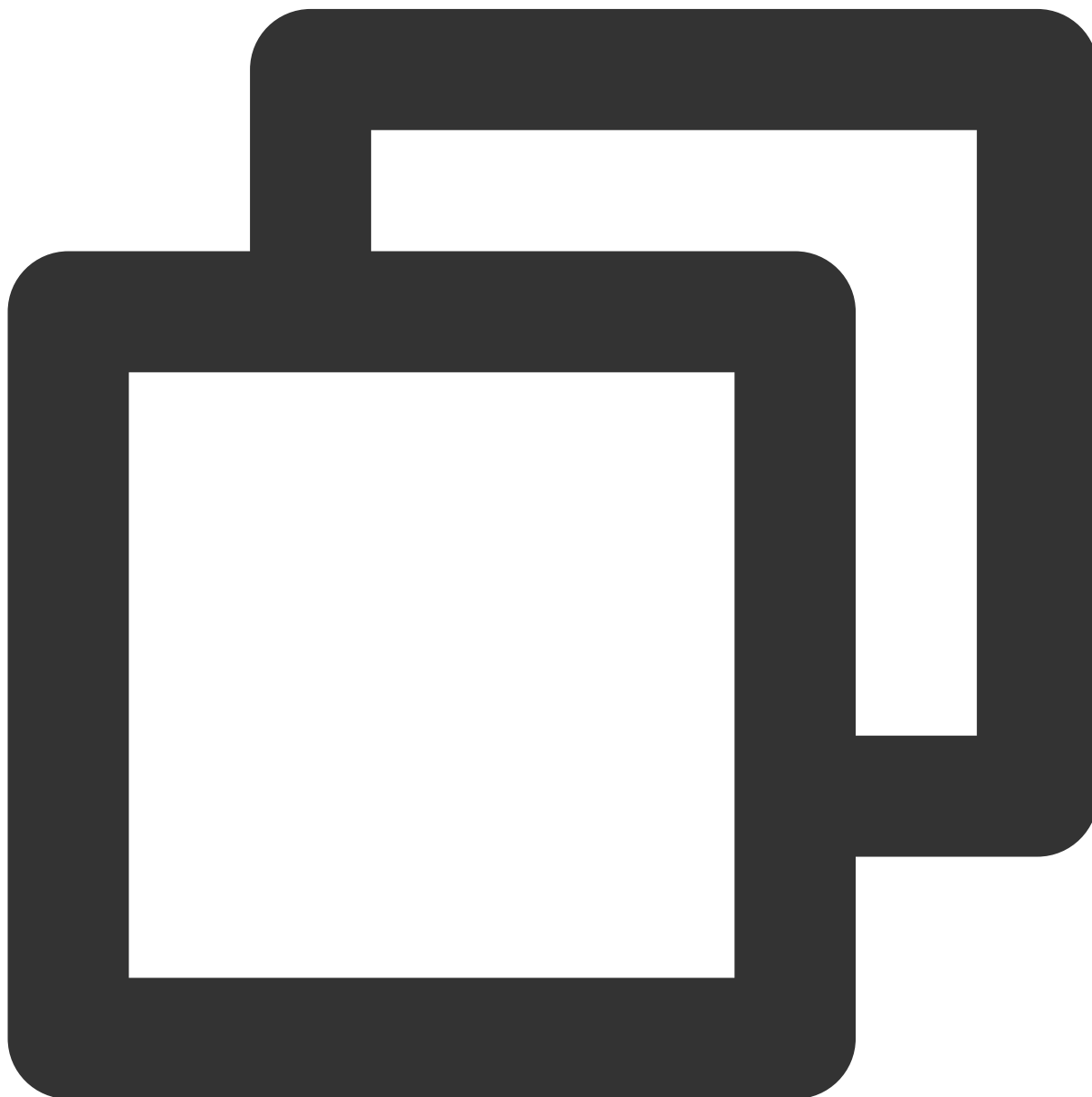
```
void onAllUserMicrophoneDisableChanged(String roomId, boolean isDisable)
```

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

roomId	String	Room ID
isDisable	boolean	Whether it is disabled

### onAllUserCameraDisableChanged

All users' Camera in the Room are disabled event.



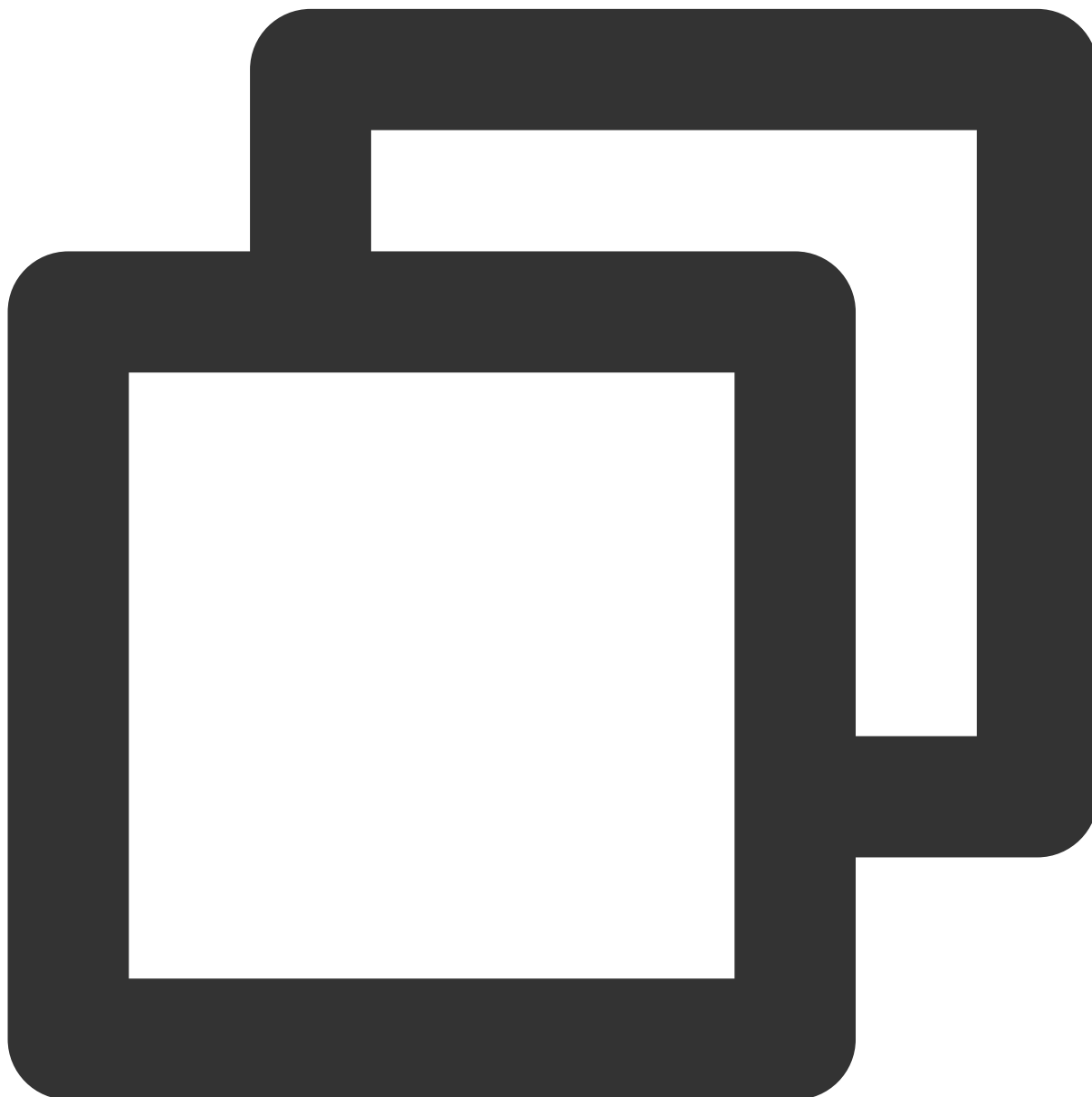
```
void onAllUserCameraDisableChanged(String roomId, boolean isDisable)
```

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

roomId	String	Room ID
isDisable	boolean	Whether it is disabled

### onSendMessageForAllUserDisableChanged

Inside the room, all users' text message sending is disabled event.

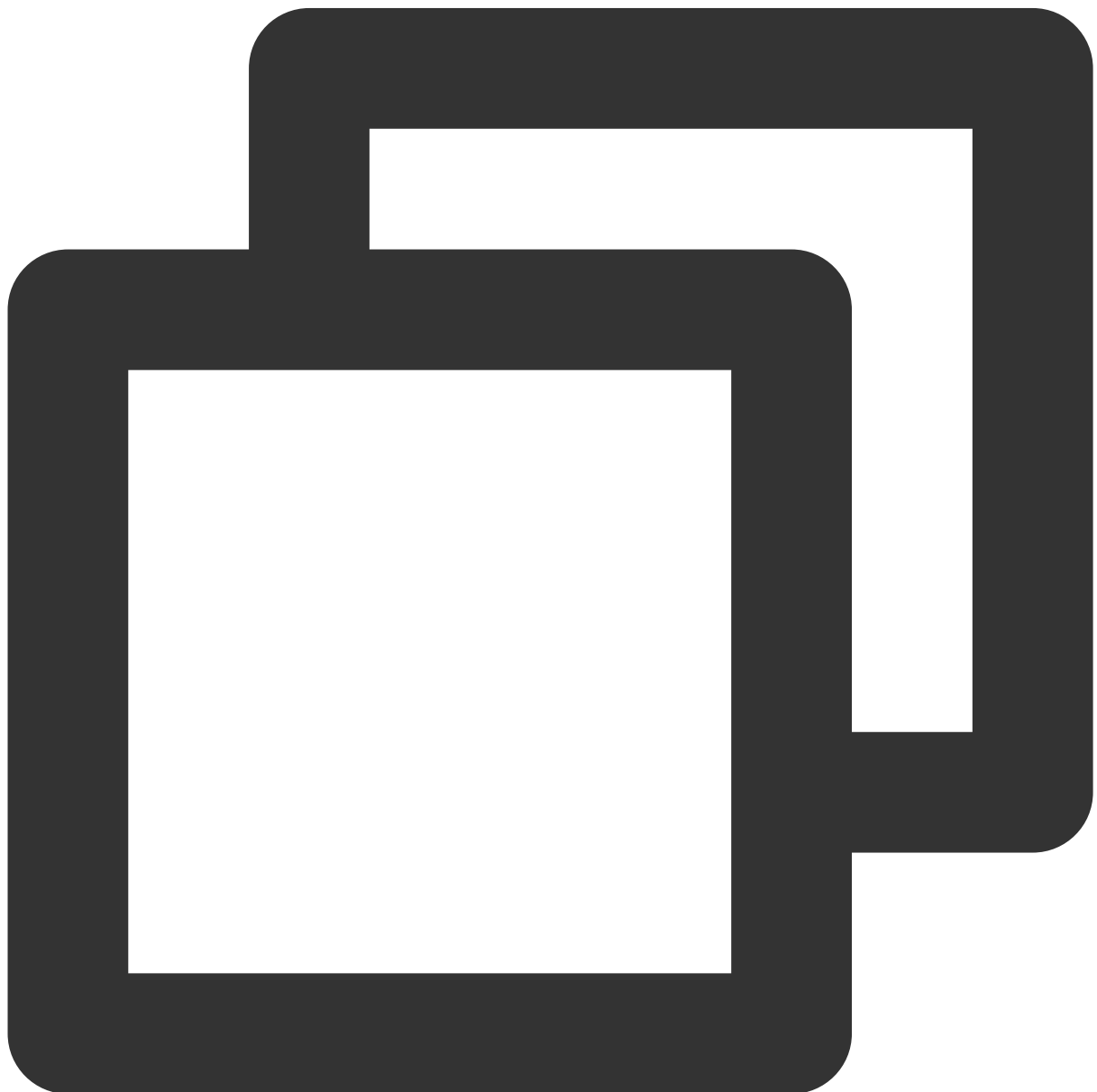


```
void onSendMessageForAllUserDisableChanged(String roomId, boolean isDisable)
```

Parameter	Type	Description
roomId	String	Room ID
isDisable	boolean	Whether it is disabled

## onRoomDismissed

Room dissolution event.

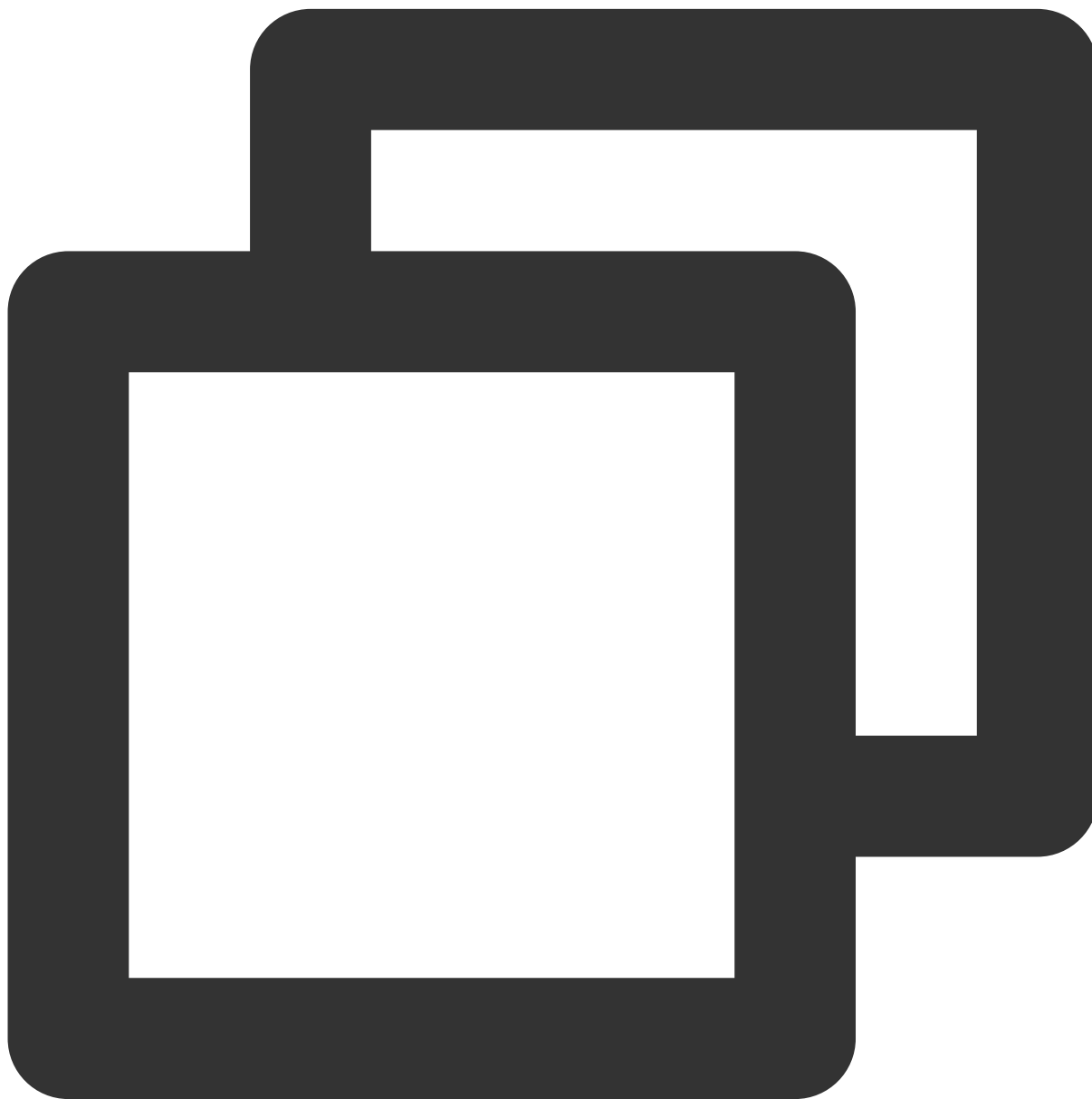


```
void onRoomDismissed(String roomId)
```

Parameter	Type	Description
roomId	String	Room ID

## onKickedOutOfRoom

Kick out of the room event



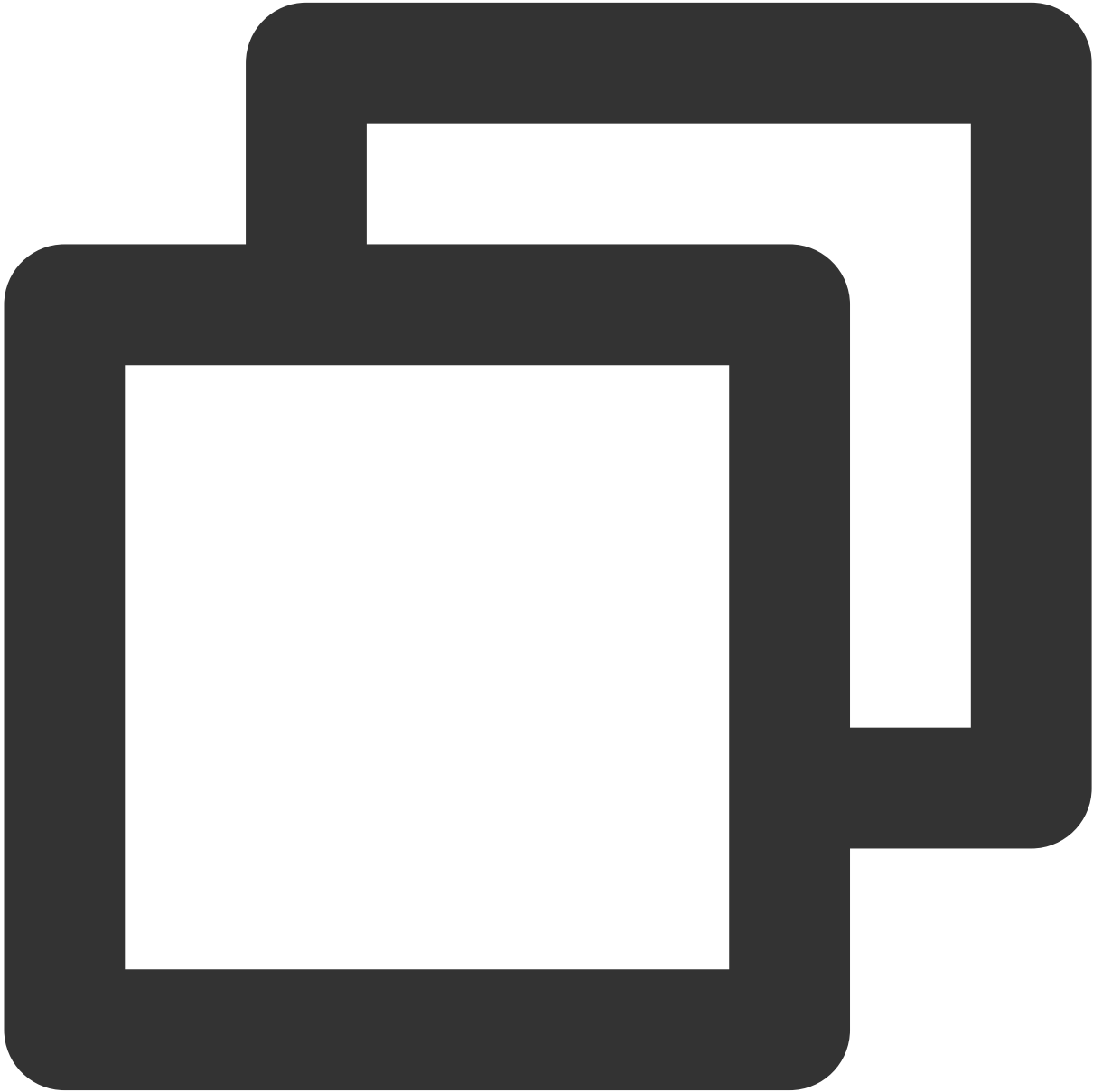
```
void onKickedOutOfRoom(String roomId, String message)
```

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

roomId	String	Room ID
message	String	Description of being kicked out

**onRoomSpeechModeChanged**

Mic control mode changes in the room.

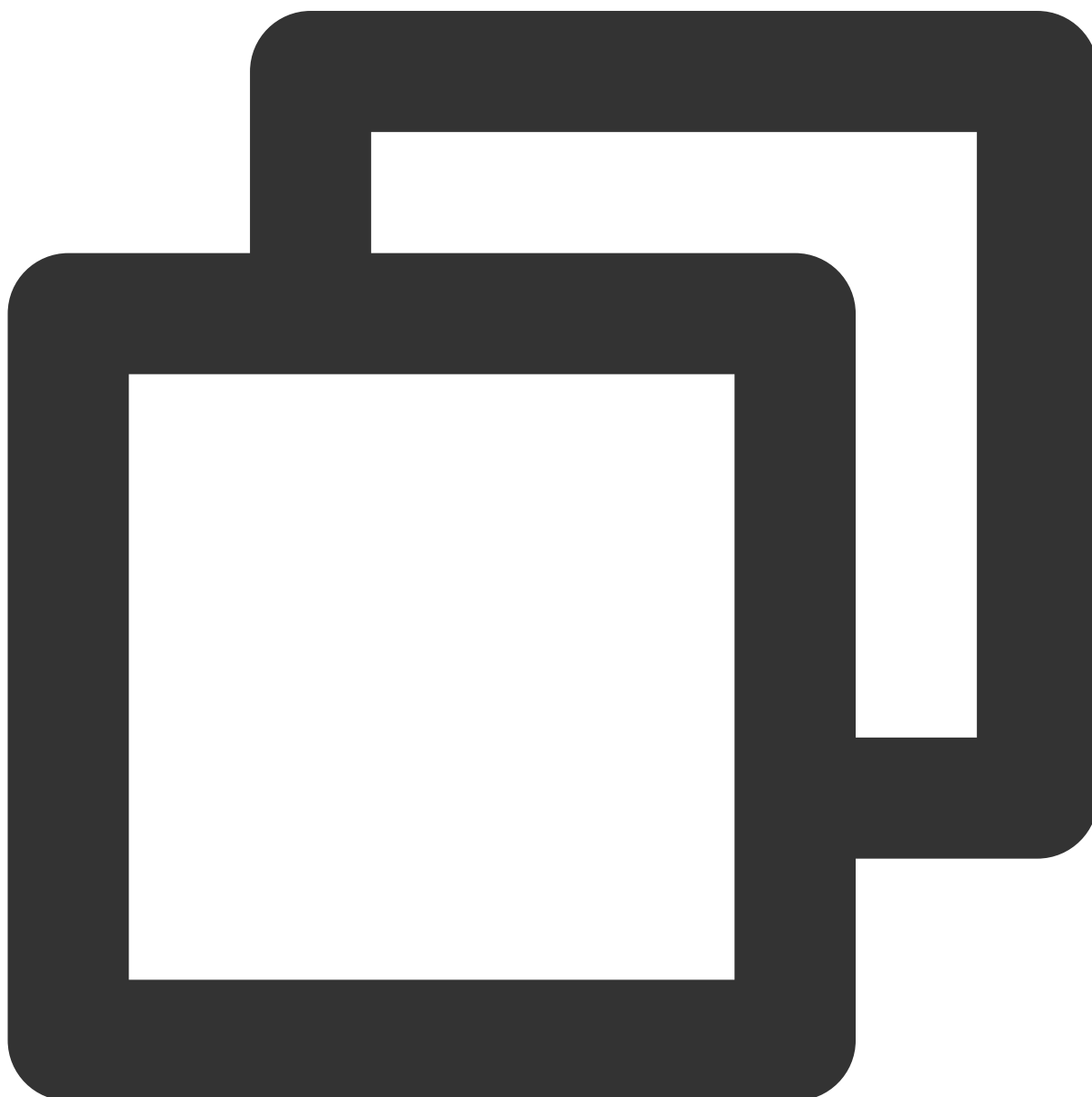


<pre>void onRoomSpeechModeChanged(String roomId, TUIRoomDefine.SpeechMode speechMode)</pre>		

Parameter	Type	Description
roomId	String	Room ID
speechMode	<a href="#">TUIRoomDefine.SpeechMode</a>	Mic control mode

## onRemoteUserEnterRoom

Remote user enters the room event.

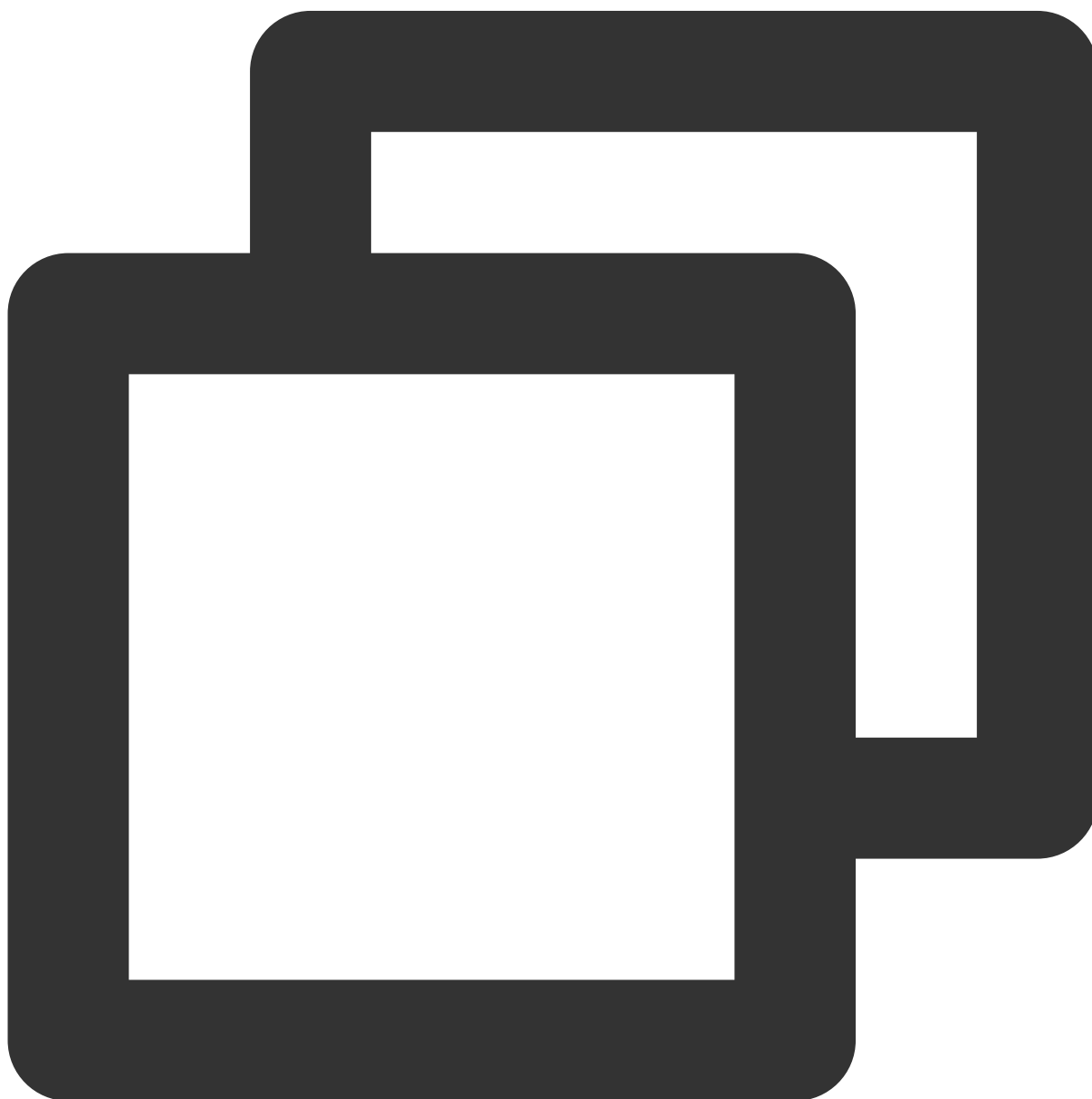


```
void onRemoteUserEnterRoom(String roomId, TUIRoomDefine.UserInfo userInfo)
```

Parameter	Type	Description
roomId	String	Room ID
userInfo	<a href="#">TUIRoomDefine.UserInfo</a>	User information

## onRemoteUserLeaveRoom

Remote user leaves the room event.



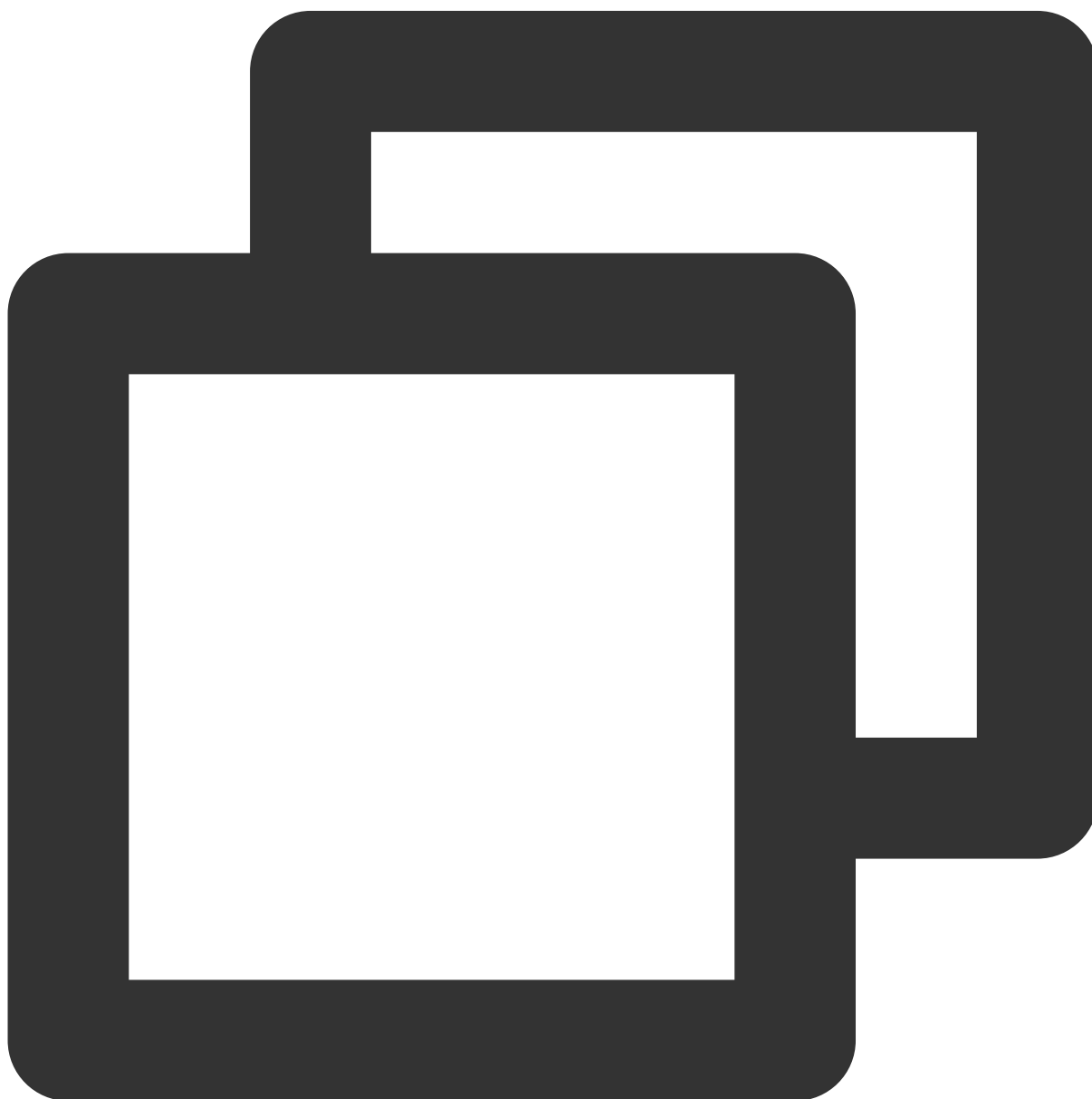
```
void onRemoteUserLeaveRoom(String roomId, TUIRoomDefine.UserInfo userInfo)
```



Parameter	Type	Description
roomId	String	Room ID
userInfo	<a href="#">TUIRoomDefine.UserInfo</a>	User information

## onUserRoleChanged

User role changes event.

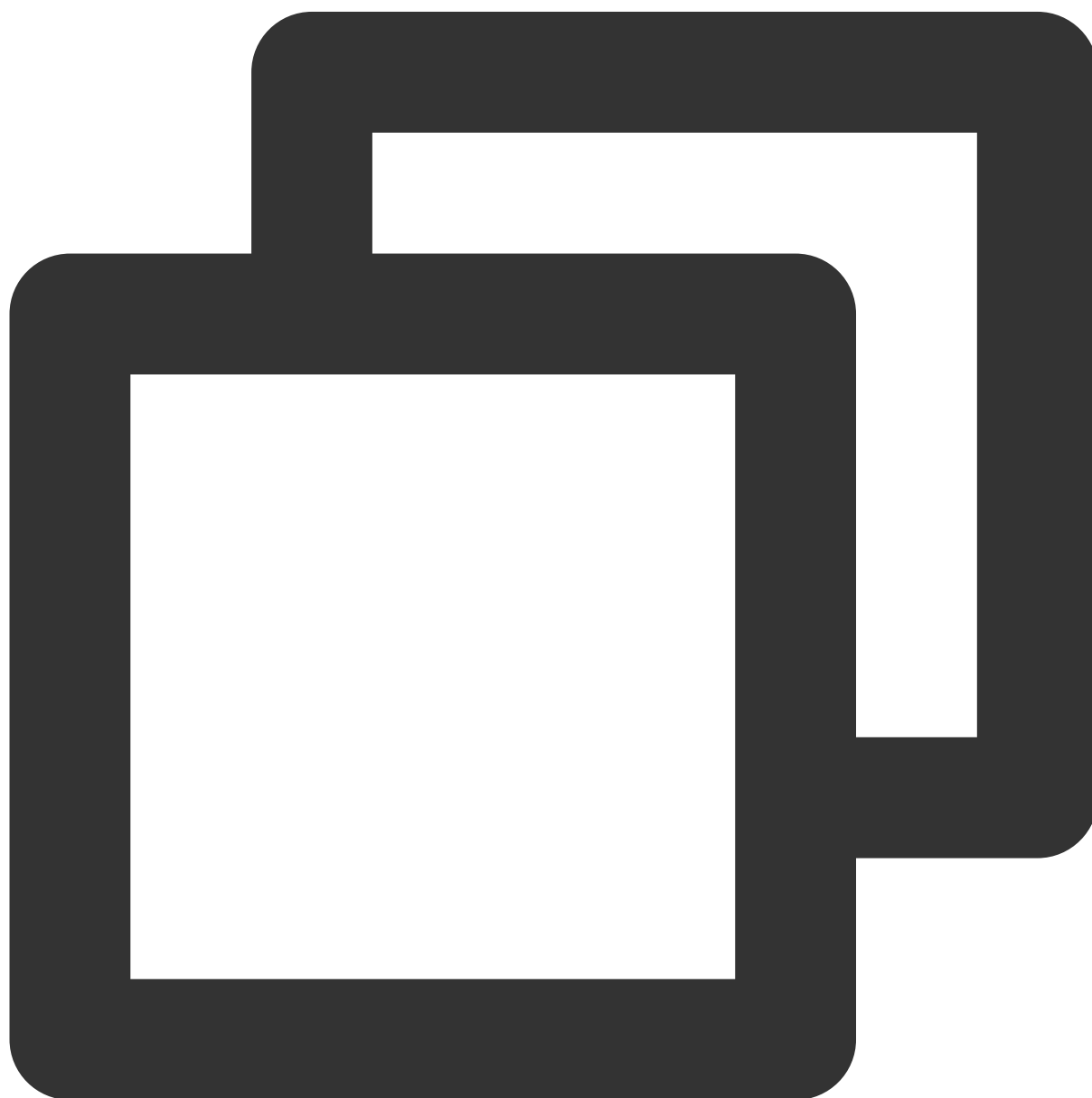


```
void onUserRoleChanged(String userId, TUIRoomDefine.Role role)
```

Parameter	Type	Description
userId	String	User ID
role	<a href="#">TUIRoomDefine.Role</a>	User Role

## onUserVideoStateChanged

User Video status changes event.

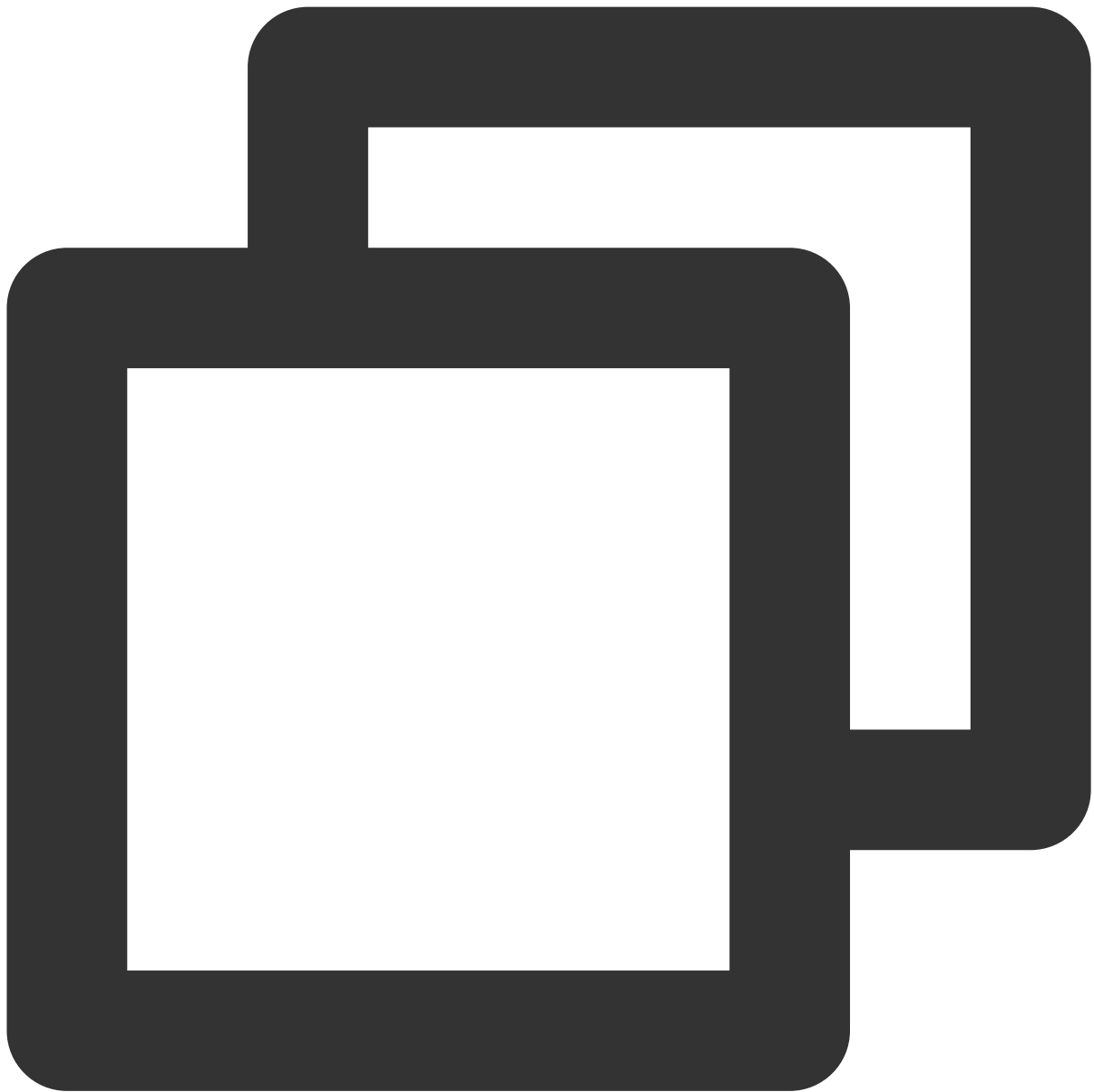


```
void onUserVideoStateChanged(String userId,
                             TUIRoomDefine.VideoStreamType streamType,
                             boolean hasVideo,
                             TUIRoomDefine.ChangeReason reason)
```

Parameter	Type	Description
userId	String	User ID
streamType	<a href="#">TUIRoomDefine.VideoStreamType</a>	Streams type
hasVideo	boolean	Whether there are streams
reason	<a href="#">TUIRoomDefine.ChangeReason</a>	Reason for streams change

## onUserAudioStateChanged

User Audio status changes event.

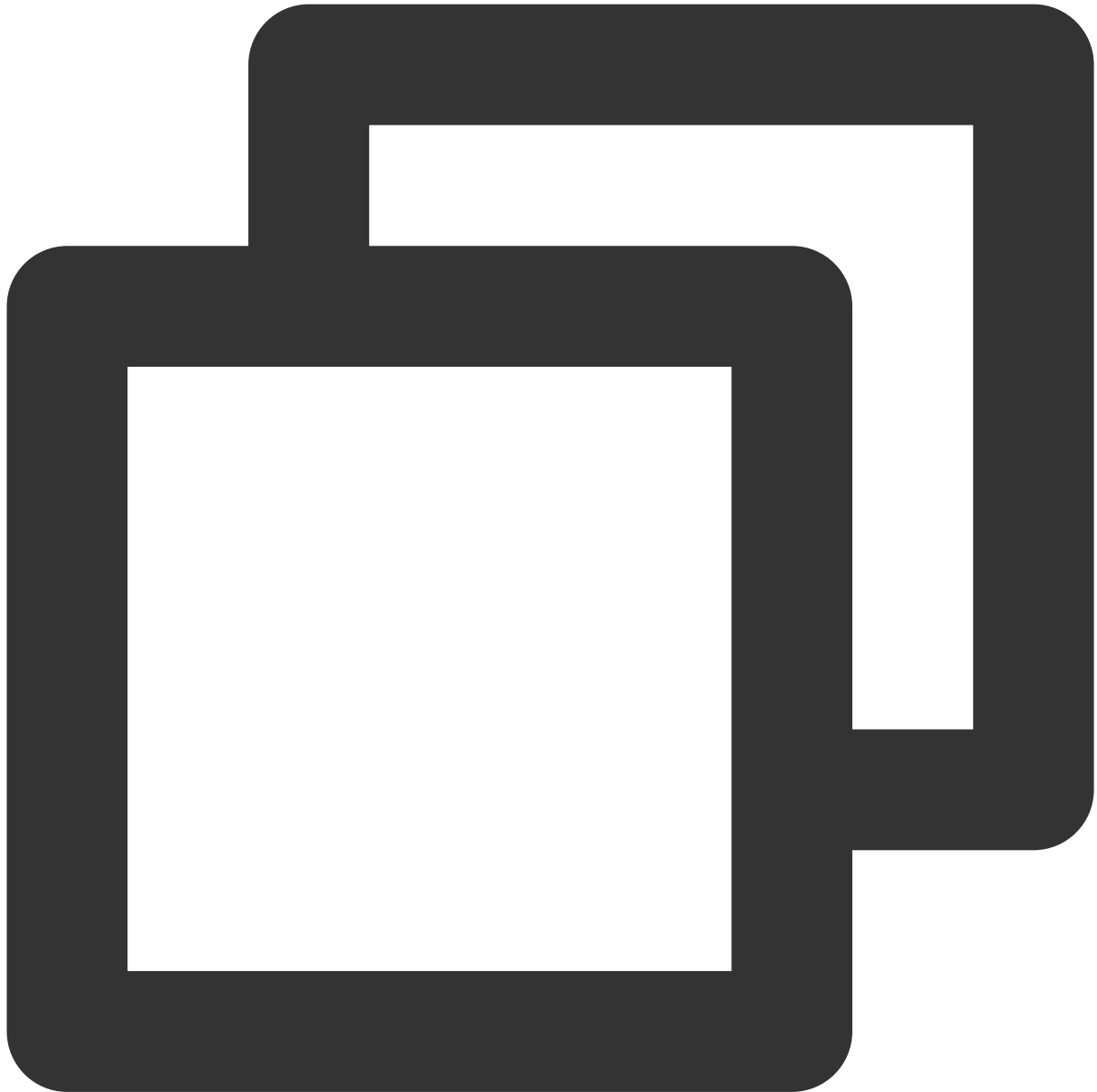


```
void onUserAudioStateChanged(String userId, boolean hasAudio, TUIRoomDefine.ChangeReason reason)
```

Parameter	Type	Description
userId	String	User ID
hasAudio	boolean	Whether there are Audio streams
reason	<a href="#">TUIRoomDefine.ChangeReason</a>	Reason for Audio streams change

## onUserVoiceVolumeChanged

User volume change event.

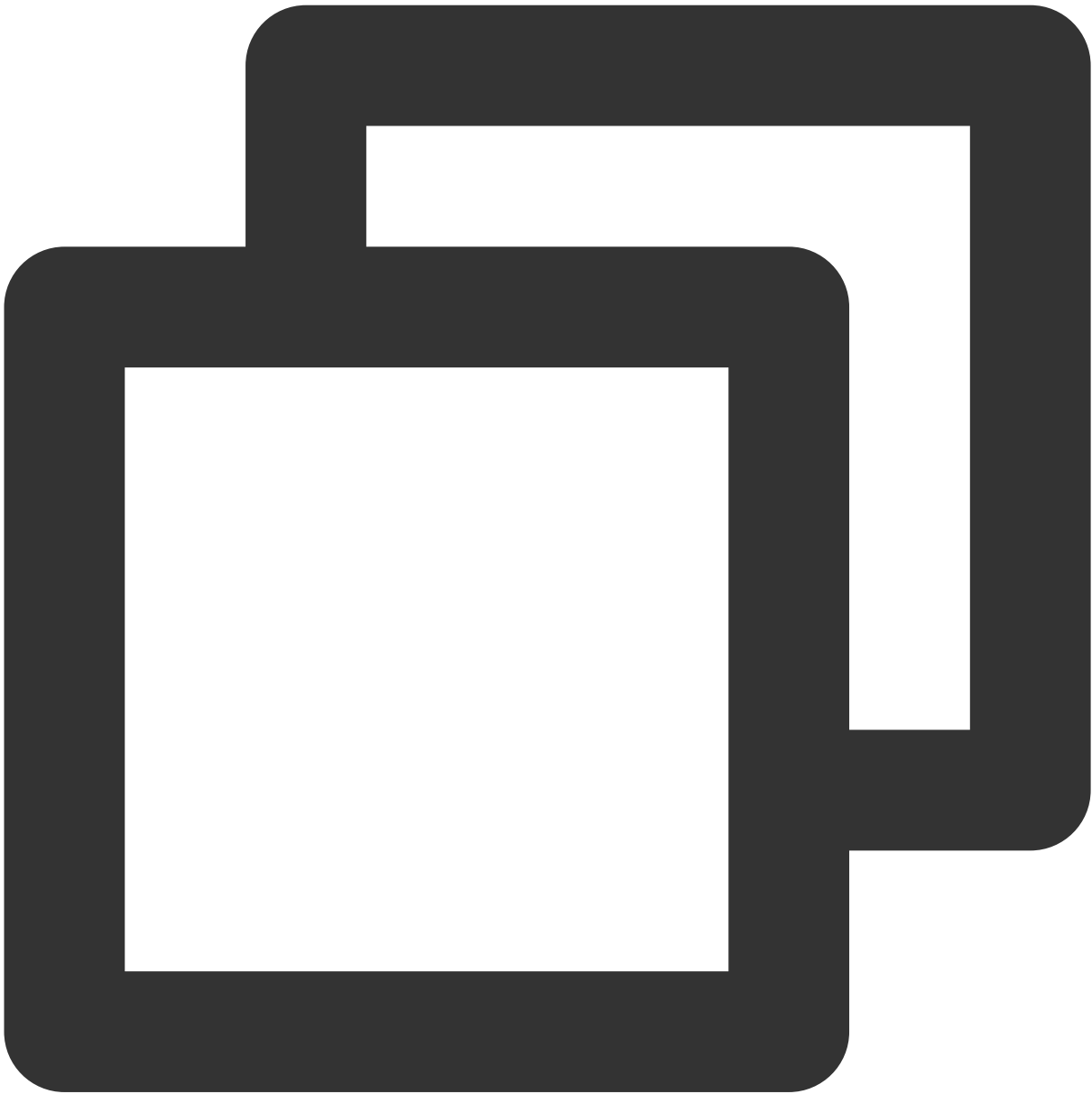


```
void onUserVoiceVolumeChanged(Map<String, Integer> volumeMap)
```

Parameter	Type	Description
volumeMap	Map	User Volume Map key: userId value: Used for carrying the volume size of all speaking users, Value range 0 - 100

onSendMessageForUserDisableChanged

User text message sending ability changes event.



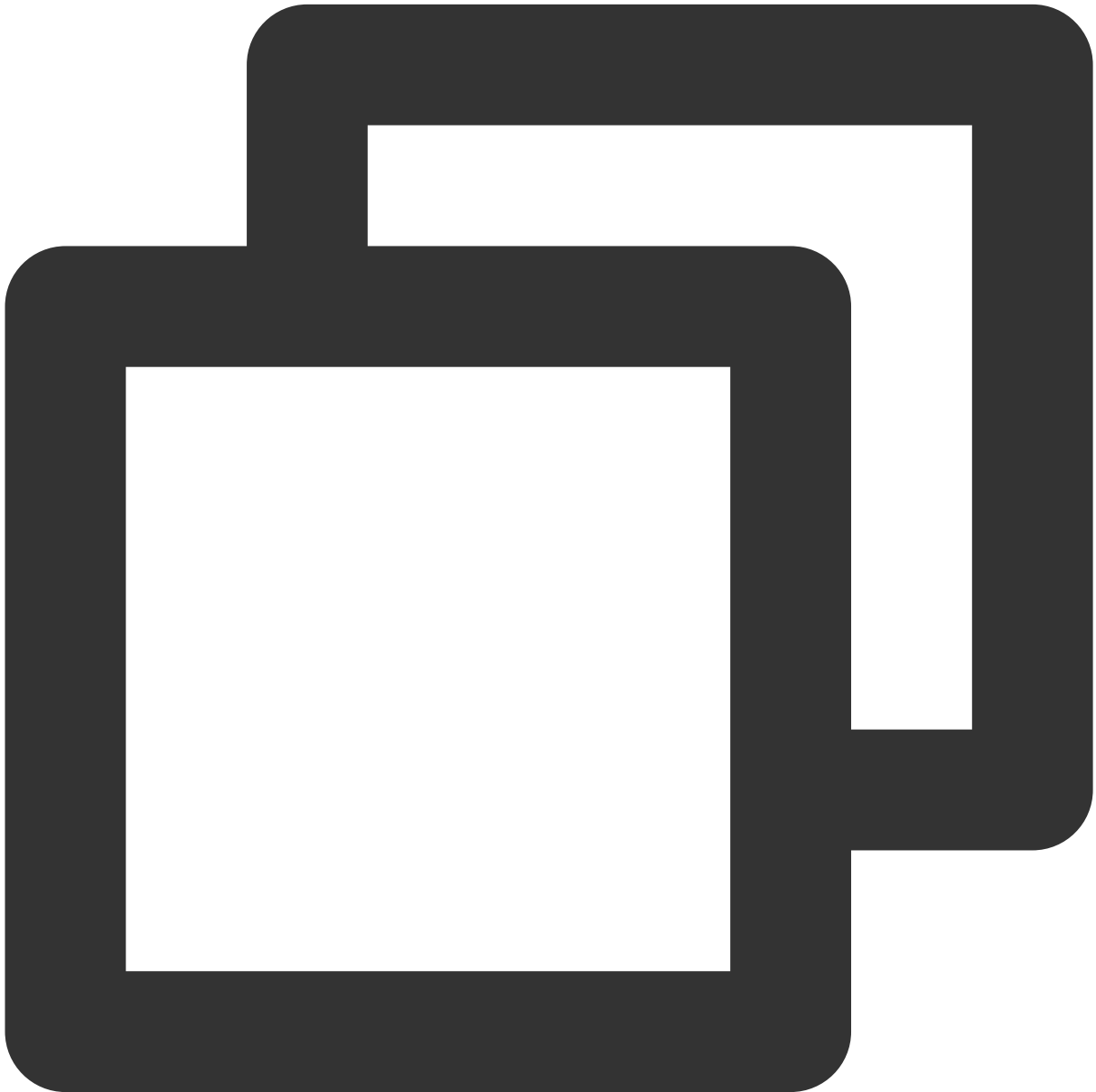
```
void onSendMessageForUserDisableChanged(String roomId, String userId, boolean isDis
```

Parameter	Type	Description
roomId	String	Room ID

userId	String	User ID
isDisable	boolean	Whether it is prohibited to send text messages.

onUserNetworkQualityChanged

User network status change event.



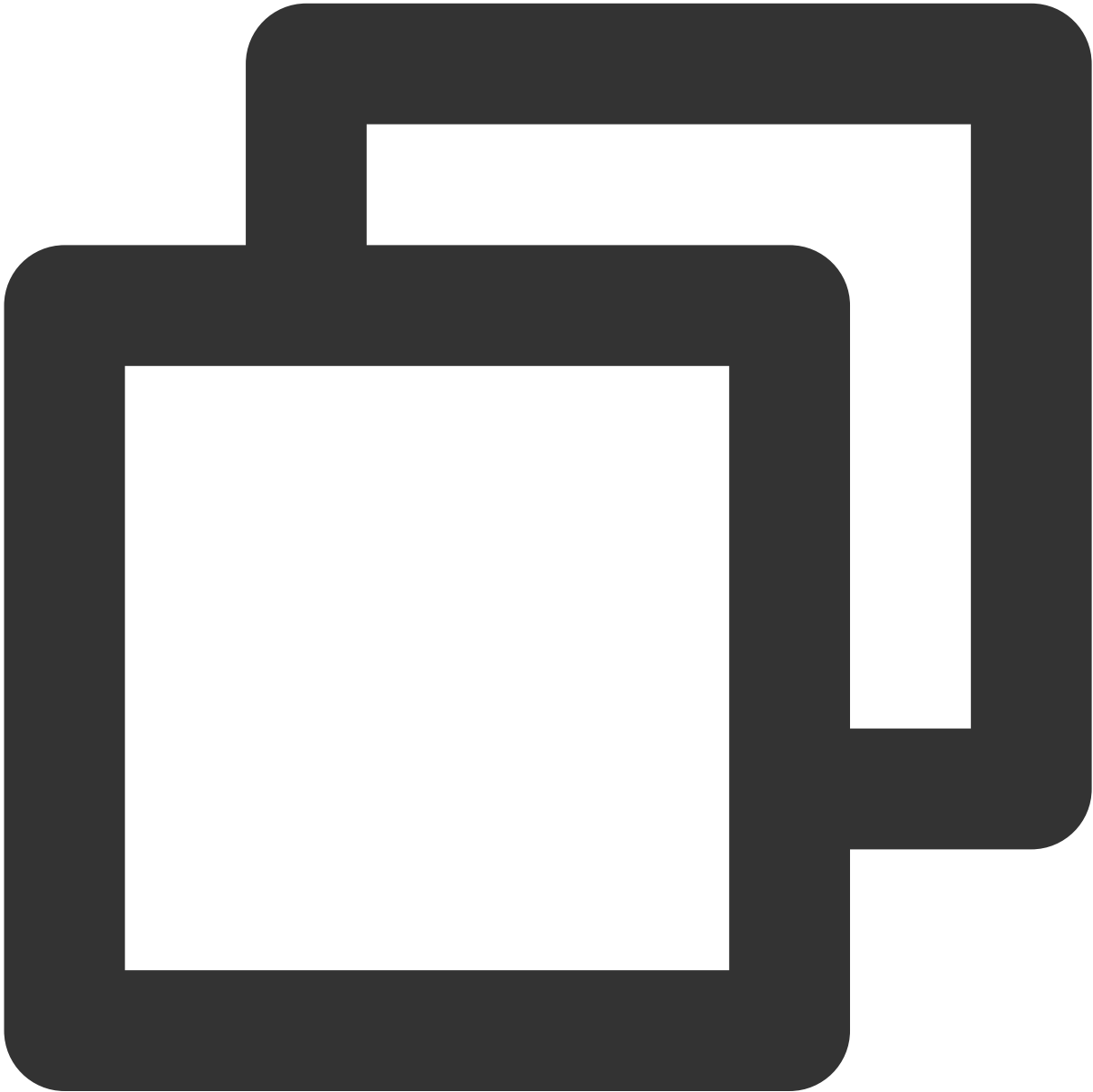
```
void onUserNetworkQualityChanged (Map<String, TUICommonDefine.NetworkInfo> networkMa
```

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

networkMap	Map	User Network Status Map key: userId value: Network Condition
------------	-----	--

**onUserScreenCaptureStopped**

Screen Sharing stopped Callback event.



```
void onUserScreenCaptureStopped(int reason)
```

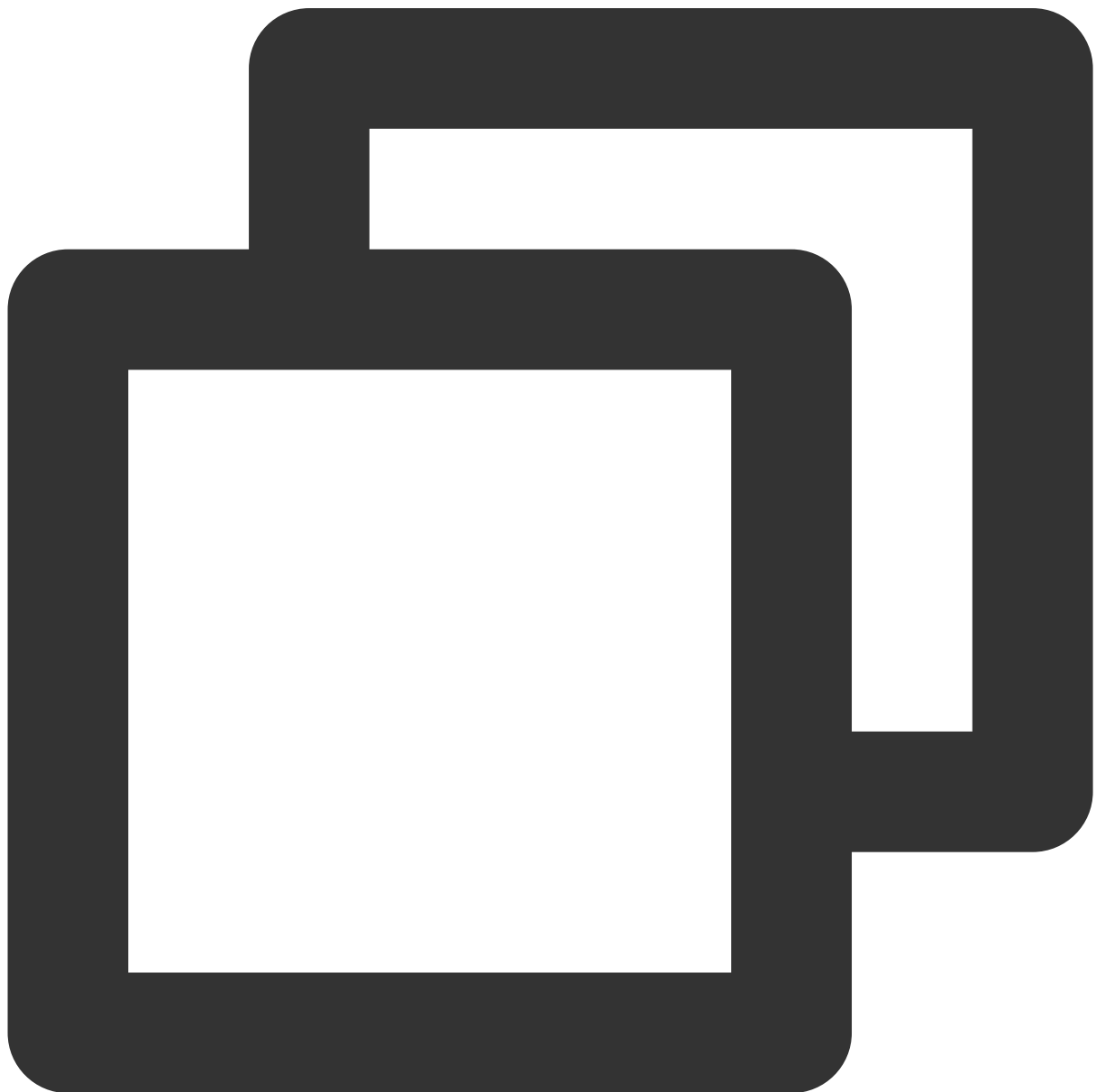
--	--	--



Parameter	Type	Description
reason	int	Stop reason: 0: User actively stops 1: Screen window closing causes the stop 2: Screen Sharing display screen status change (such as interface being unplugged, Projection mode change, etc.)

### **onRoomMaxSeatCountChanged**

Maximum number of mic slots changes event in the room (only effective in meeting type rooms).

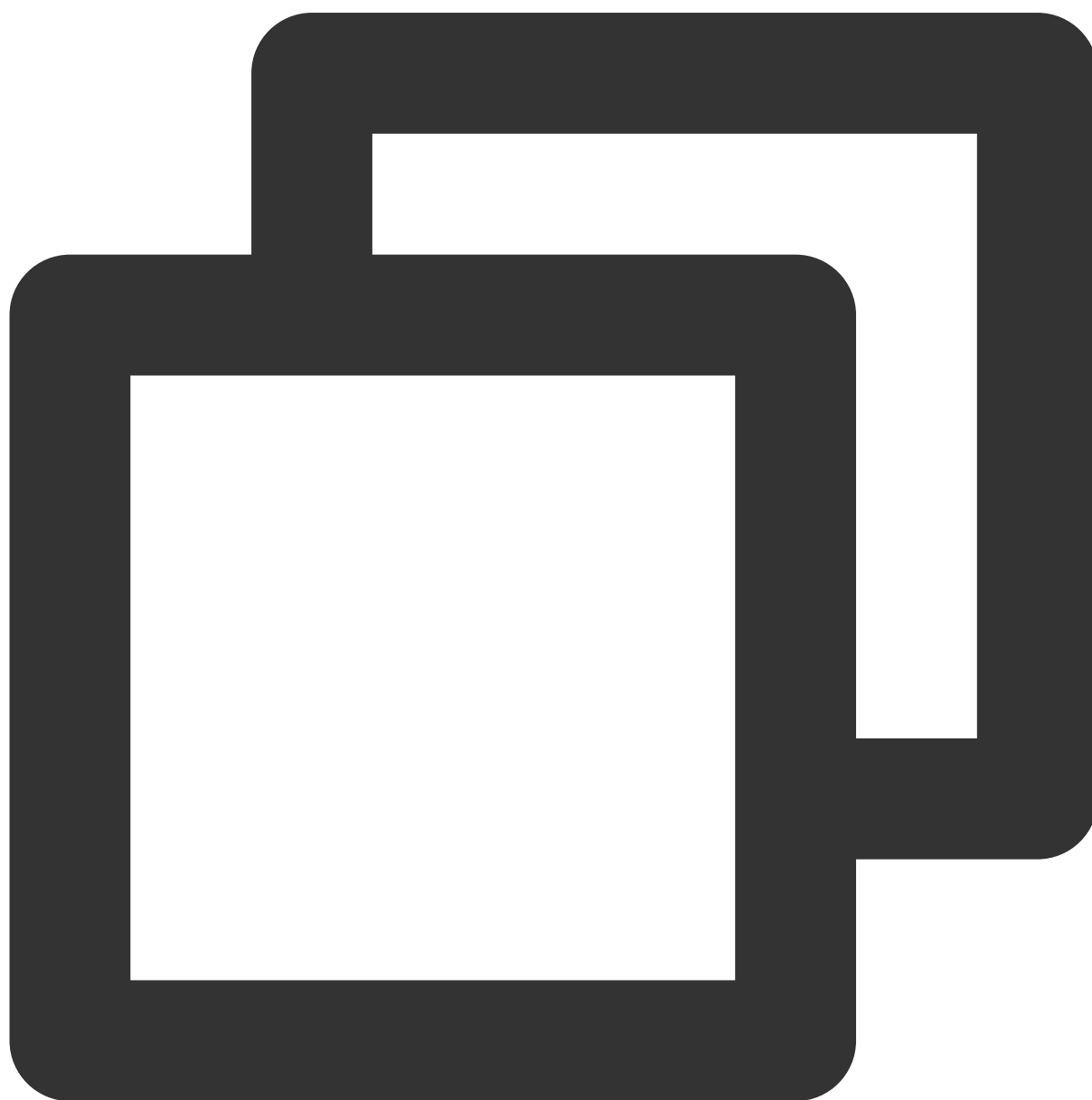


```
void onRoomMaxSeatCountChanged(String roomId, int maxSeatCount)
```

Parameter	Type	Description
roomId	String	Room ID
maxSeatCount	int	Maximum number of mic slots in the room

## onSeatListChanged

Mic slot list changes event.



```
void onSeatListChanged(List<TUIRoomDefine.SeatInfo> seatList,  
                      List<TUIRoomDefine.SeatInfo> seatedList,  
                      List<TUIRoomDefine.SeatInfo> leftList)
```

Parameter	Type	Description
seatList	List< <a href="#">TUIRoomDefine.SeatInfo</a> >	The latest user list on the mic, including newly on mic users
seatedList	List< <a href="#">TUIRoomDefine.SeatInfo</a> >	Newly on mic user list
leftList	List< <a href="#">TUIRoomDefine.SeatInfo</a> >	Newly off mic user list

### onKickedOffSeat

Received the event of user being kicked off mic.



```
void onKickedOffSeat (String userId)
```

Parameter	Type	Description
userId	String	Operate Kick-out of the (Host/Administrator) User ID

## onRequestReceived

Received request message event.



```
void onRequestReceived(TUIRoomDefine.Request request)
```

Parameter	Type	Description
request	<a href="#">TUIRoomDefine.Request</a>	Request content

### onRequestCancelled

Received request cancellation event.

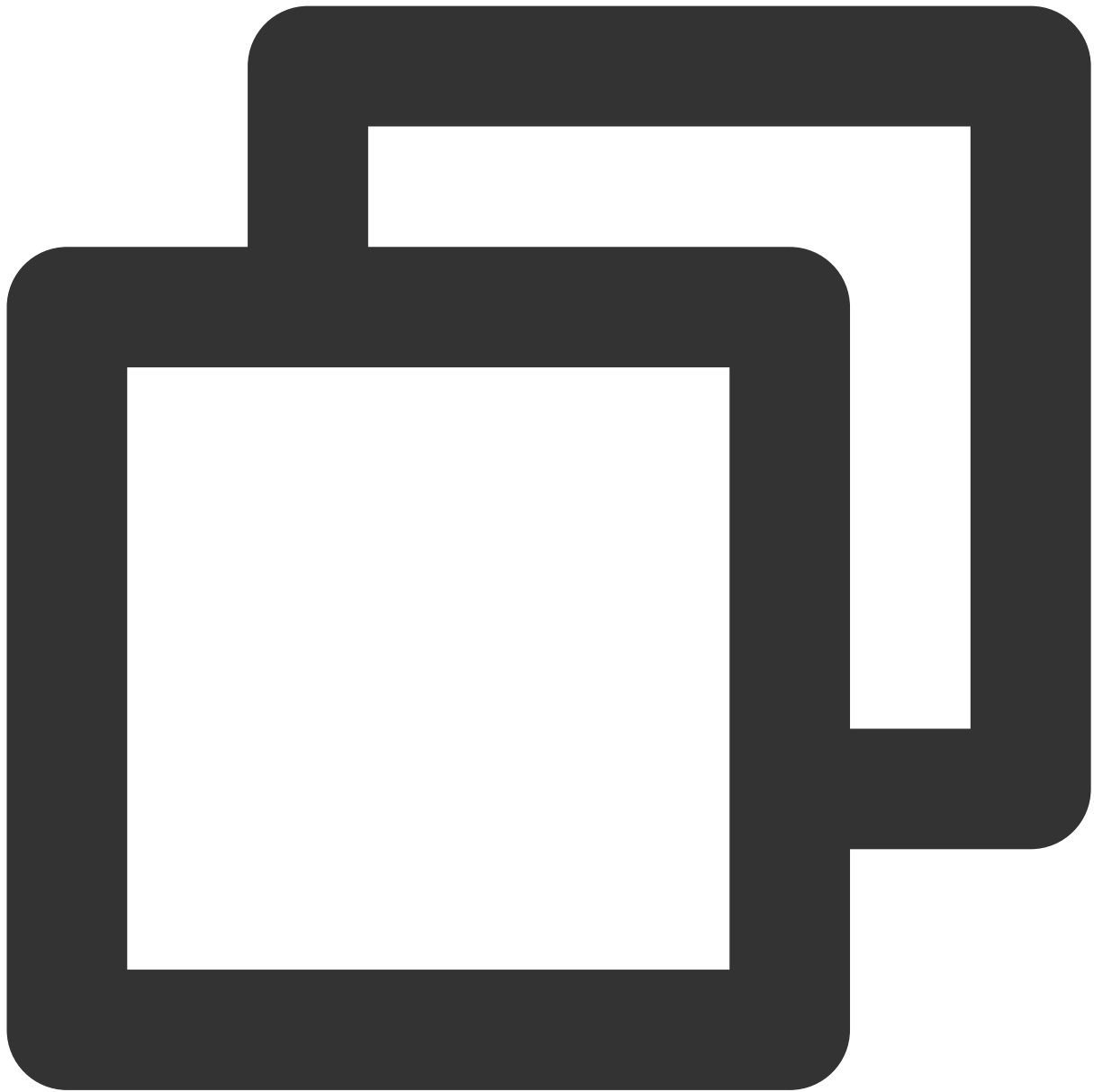


```
void onRequestCancelled(String requestId, String userId)
```

Parameter	Type	Description
requestId	String	Request ID
userId	String	Cancel signaling user ID

## onReceiveTextMessage

Received ordinary text message event.

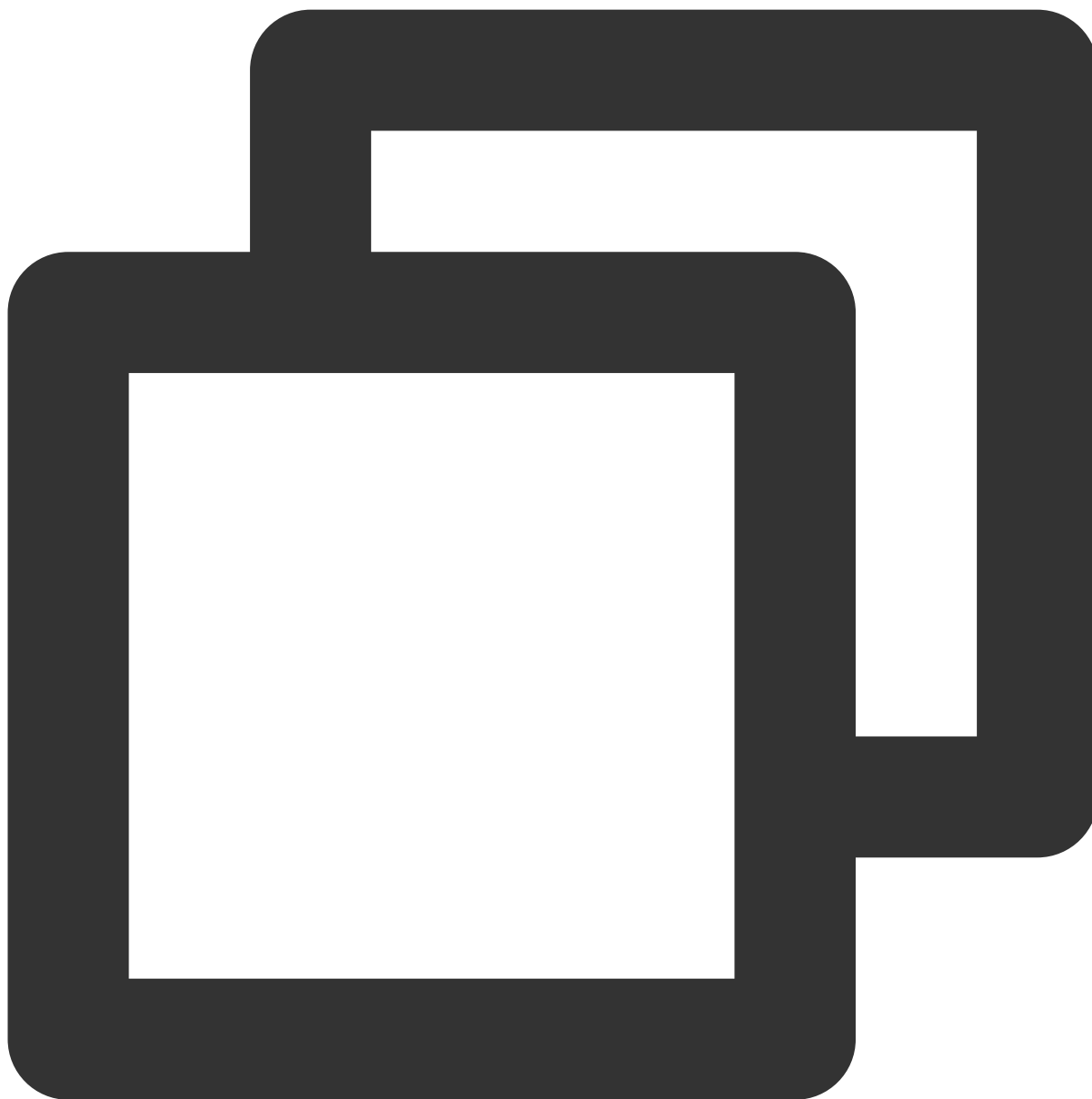


```
void onReceiveTextMessage(String roomId, TUICommonDefine.Message message)
```

Parameter	Type	Description
roomId	String	Room ID
message	<a href="#">TUICommonDefine.Message</a>	Message content

### onReceiveCustomMessage

Received custom message event.



```
void onReceiveCustomMessage(String roomId, TUICommonDefine.Message message)
```

Parameter	Type	Description
roomId	String	Room ID
message	<a href="#">TUICommonDefine.Message</a>	Message content





# Type Definition

Last updated : 2023-10-23 11:30:42

## Enumeration Definition

### TUIRoomDefine

Type	Description
<a href="#">RoomType</a>	Room Type
<a href="#">SpeechMode</a>	Room Mode
<a href="#">MediaDevice</a>	Room Media Device Type
<a href="#">Role</a>	Room Role Type
<a href="#">VideoQuality</a>	Video Quality
<a href="#">AudioQuality</a>	Audio Quality
<a href="#">VideoStreamType</a>	Video Stream Type
<a href="#">ChangeReason</a>	Change Reason (User audio and video status change operation reason: self-modification or modified by room owner/administrator)
<a href="#">RequestAction</a>	Request Type

### TUICommonDefine

Type	Description
<a href="#">NetworkQuality</a>	Network Quality

## Common Structure

### TUIRoomDefine

Type	Description
<a href="#">RoomInfo</a>	Room data
<a href="#">LoginUserInfo</a>	User Login Information
<a href="#">UserInfo</a>	Room User Information

<a href="#">SeatInfo</a>	Room Seat Information
<a href="#">SeatLockParams</a>	Lock Seat Operation Parameters
<a href="#">UserVoiceVolume</a>	Room User Volume
<a href="#">Request</a>	Signaling Request

## TUICommonDefine

Type	Description
<a href="#">NetworkInfo</a>	Network Quality Information
<a href="#">Message</a>	Message

## RoomType

Room Type

Enumeration	Value	Description
CONFERENCE	1	Conference Type Room, suitable for conference and education scenarios, this room can enable free speech, apply for speech, go live and other modes.
LIVE_ROOM	2	Live Type Room, suitable for live broadcast scenarios, this room can enable free speech, mic control mode, and the seats in this room are numbered.

## SpeechMode

Mic Control Mode

Enumeration	Value	Description
FREE_TO_SPEAK	1	Free speech mode.
APPLY_TO_SPEAK	2	Apply to speak mode. (Only effective in conference type room)
SPEAK_AFTER_TAKING_SEAT	3	Go Live mode.

## MediaDevice

Room Media Device Type

--	--	--

Enumeration	Value	Description
MICROPHONE	1	Mic
CAMERA	2	Camera
SCREEN_SHARING	3	Screen Sharing

## Role

### Room Role Types

Enumeration	Value	Description
ROOM_OWNER	0	Room Owner, generally refers to the creator of the room, the highest authority holder in the room
MANAGER	1	Room Administrator
GENERAL_USER	2	General Member in the room

## VideoQuality

### Video Quality

Enumeration	Value	Description
Q_360P	1	Low-quality 360P
Q_540P	2	Standard Definition 540P
Q_720P	3	High Definition 720P
Q_1080P	4	Ultra-clear 1080P

## AudioQuality

### Audio Quality

Enumeration	Value	Description
SPEECH	0	Speech Mode
DEFAULT	1	Default Mode
MUSIC	2	Music Mode

## VideoStreamType

### Video Stream Type

Enumeration	Value	Description
CAMERA_STREAM	0	High-quality Camera Video Stream
SCREEN_STREAM	1	Screen Sharing Video Stream
CAMERA_STREAM_LOW	2	Low-quality Camera Video Stream

## ChangeReason

Change Reason (User audio and video status change operation reason: self-modification or modification by room owner/administrator)

Enumeration	Value	Description
BY_SELF	0	Self-operation
BY_ADMIN	1	Room Owner or Administrator Operation

## RequestAction

### Request Type

Enumeration	Value	Description
INVALID_ACTION	0	Invalid Request
REQUEST_TO_OPEN_REMOTE_CAMERA	1	Request Remote User to Open Camera
REQUEST_TO_OPEN_REMOTE_MICROPHONE	2	Request Remote User to Open Microphone
REQUEST_TO_CONNECT_OTHER_ROOM	3	Request to Connect to Other Room
REQUEST_TO_TAKE_SEAT	4	Request to Go Live
REQUEST_REMOTE_USER_ON_SEAT	5	Request Remote User to Go Live
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_CAMERA	6	Request to Admin to Open Local Camera
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_MICROPHONE	7	Request to Admin to

[Open Local Microphone](#)

## NetworkQuality

### Network Quality

Enumeration	Value	Description
UNKNOWN	0	Undefined
EXCELLENT	1	Current Network is Excellent
GOOD	2	Current Network is Good
POOR	3	Current Network is Average
BAD	4	Current Network is Poor
VERY_BAD	5	Current Network is Very Poor
DOWN	6	Current Network Does Not Meet TRTC's Minimum Requirements

## RoomInfo

### Room Information

Field	Type	Description
roomId	String	Room ID
ownerId	String	Host ID, default is the room creator (read-only)
roomType	<a href="#">RoomType</a>	Room Type
name	String	Room Name, default is the room ID
speechMode	<a href="#">SpeechMode</a>	Mic Control Mode
isCameraDisableForAllUser	boolean	Whether to Disable Opening Camera (optional when creating a room), default value: false
isMicrophoneDisableForAllUser	boolean	Whether to Disable Opening Microphone (optional when creating a room), default value: false
isMessageDisableForAllUser	boolean	Whether to Disable Sending Messages (optional when creating a room), default value: false
maxSeatCount	int	Maximum Number of Mic Seats (only supported when

		entering the room and creating the room)
enableCDNStreaming	boolean	Whether to Enable CDN Live Streaming (optional when creating a room, for live streaming rooms), default value: false
cdnStreamDomain	String	Live Streaming Push Domain (optional when creating a room, for live streaming rooms), default value: empty
createTime	long	Room Creation Time (read-only)
memberCount	int	Number of Members in the Room (read-only)

## LoginUserInfo

User Login Information

Field	Type	Description
userId	String	User ID
userName	String	User Name
avatarUrl	String	User Avatar URL

## UserInfo

User Information in the Room

Field	Type	Description
userId	String	User ID
userName	String	User Name
avatarUrl	String	User Avatar URL
userRole	<a href="#">Role</a>	User Role Type
hasAudioStream	boolean	Whether There is Audio Stream, default value: false
hasVideoStream	boolean	Whether There is Video Stream, default value: false
hasScreenStream	boolean	Whether There is Screen Sharing Stream, default value: false

## SeatInfo

## Seat Information in the Room

Field	Type	Description
index	int	Mic Seat Number
userId	String	User ID
isLocked	boolean	Whether the Mic Seat is Locked, default false
isVideoLocked	boolean	Whether the Mic Seat is Prohibited from Opening Camera, default false
isAudioLocked	boolean	Whether the Mic Seat is Prohibited from Opening Microphone, default false

**SeatLockParams**

## Lock Seat Operation Parameters

Field	Type	Description
lockSeat	boolean	Lock Mic Seat, default false
lockVideo	boolean	Lock Mic Seat Camera, default false
lockAudio	boolean	Lock Mic Seat Microphone, default false

**UserVoiceVolume**

## User Voice Volume in the Room

Field	Type	Description
userId	String	User ID
volume	int	Volume Size, Value range 0 - 100

**Request**

## Signaling Request

Field	Type	Description
requestId	String	Request ID



requestAction	<a href="#">RequestAction</a>	Request Type
userId	String	User ID
content	String	Signaling Content
timestamp	int	Timestamp

## NetworkInfo

### Network Quality Information

Field	Type	Description
userId	String	User ID
quality	<a href="#">NetworkQuality</a>	Network Quality
upLoss	int	Upstream Packet Loss Rate
downLoss	int	Downstream Packet Loss Rate
delay	int	Network Delay

## Message

### Message

Field	Type	Description
messageId	String	Message ID
message	String	Message Text
timestamp	long	Message Time
userId	String	Message Sender
userName	String	Message Sender Nickname
avatarUrl	String	Message Sender Avatar

# Web

## RoomKit API

Last updated : 2024-06-14 11:49:41

### API Introduction

The TUIRoomKit API is a multi-person meeting component with an **Including UI Interface**. By using the TUIRoomKit API, you can quickly implement a meeting-like scenario through a simple interface. For more detailed integration steps, see: [Rapid Integration with TUIRoomKit](#).

### API Overview

`<ConferenceMainView />`: The component body of TUIRoomkit UI.

Conference: dependencies `ConferenceMainView` provided API.

API	Description
<a href="#">getRoomEngine</a>	Access the roomEngine instance. If roomEngine does not exist, it will return null.
<a href="#">on</a>	Listen to specified types of apid Integration with TUIRoomKit  .Event. When the event occurs, the callback function will be called.
<a href="#">off</a>	Cancel listening for events of a specified type.
<a href="#">login</a>	Log in to the conference system.
<a href="#">logout</a>	Log out of the meeting system.
<a href="#">start</a>	Start a new meeting.
<a href="#">join</a>	Join an existing meeting.
<a href="#">leave</a>	Leave the current meeting.
<a href="#">dismiss</a>	Dismiss the current meeting.
<a href="#">setSelfInfo</a>	Set your user information.
<a href="#">setLanguage</a>	Set the interface language.

<a href="#">setTheme</a>	Set the interface topic.
<a href="#">enableWatermark</a>	Enable the text messaging feature in the application. For details, see: <a href="#">Text Watermark</a> .
<a href="#">enableVirtualBackground</a>	Enable the virtual background feature in the application. After calling this function, a virtual background feature button will be displayed on the UI. For details, see: <a href="#">Virtual Background</a> .
<a href="#">disableTextMessaging</a>	Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.
<a href="#">disableScreenSharing</a>	Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.
<a href="#">hideFeatureButton</a>	Hide specific feature buttons in the application. By invoking this function and passing in the appropriate <a href="#">FeatureButton</a> enumerated values, the corresponding buttons will be hidden from the user interface.

## Introduction to ConferenceMainView attributes

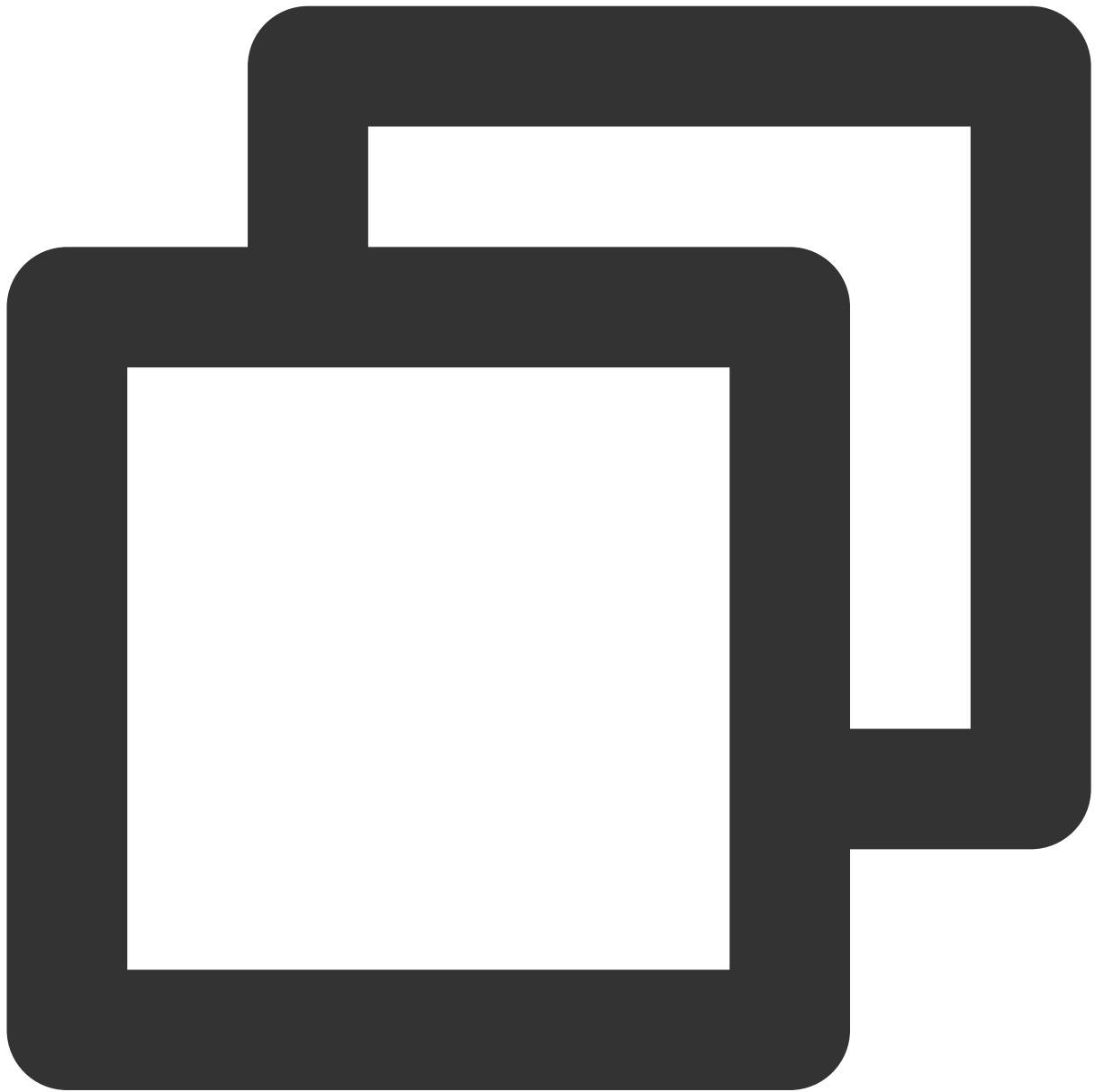
### Attribute Overview

Attribute	Description	Type	Required	Default Value
displayMode	Control of component display pattern permanent: Permanent mode. The component is always displayed; internal control of the component's visibility is not exerted. If not controlled by the business end, the component will always remain visible. wake-up: Wake-up mode. The component is activated only after calling the <a href="#">conference start/join</a> interface and officially joining the conference; it will not be displayed beforehand.	'permanent'   'wake-up'	No	permanent

### Sample code

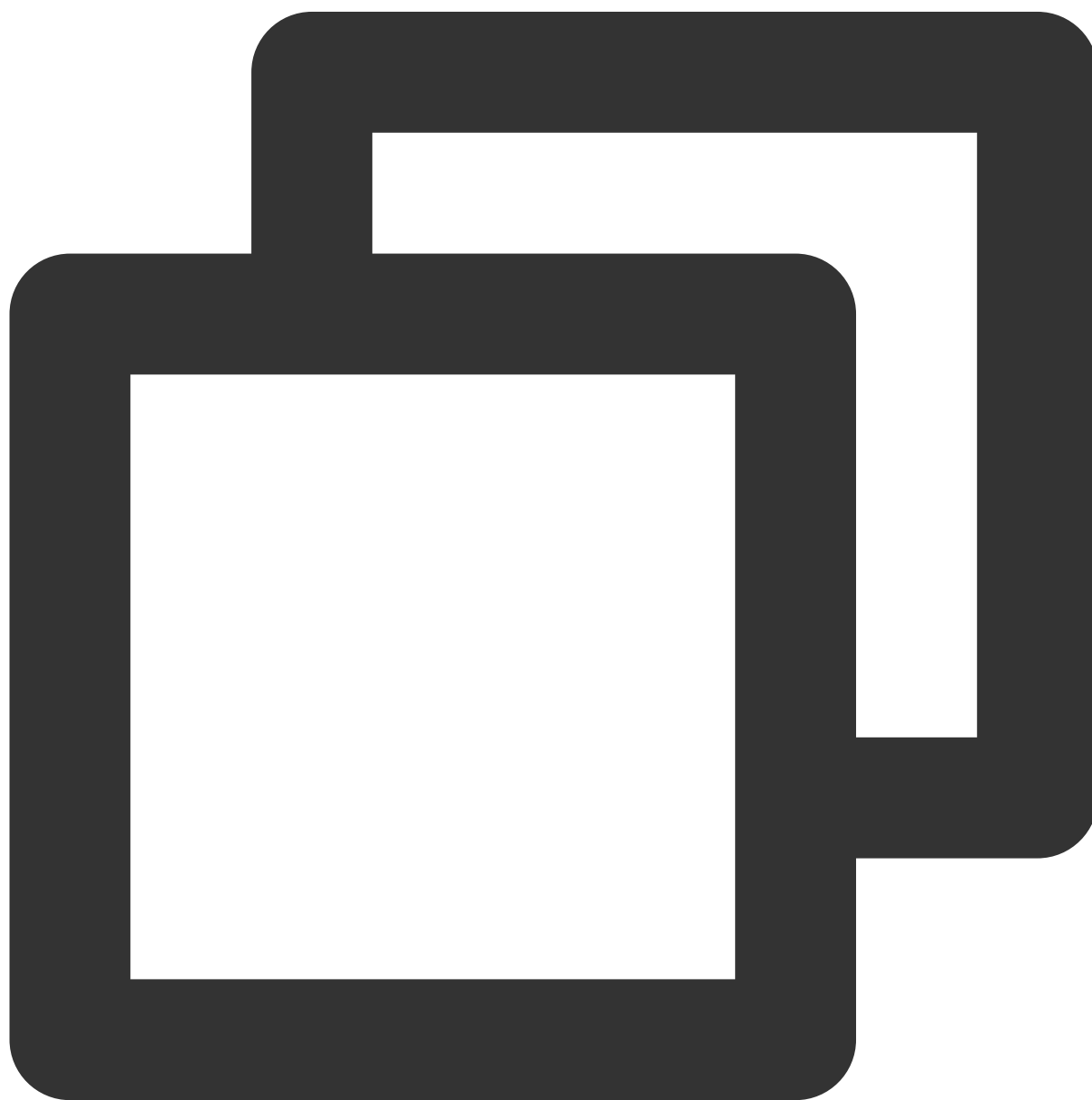
Vue3

Vue2



```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue3';
const init = async () => {
  await conference.login({
    sdkAppId: 0,
    userId: '',
  })
}
```

```
    userSig: '',  
  });  
  await conference.start('123456', {  
    isSeatEnable: false,  
    isOpenCamera: true,  
    isOpenMicrophone: true,  
  });  
}  
init();  
</script>
```



```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
  async created() {
    await conference.login({
      sdkAppId: 0,
      userId: '',
      userSig: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
  },
};
</script>
```

## Conference API Details

Conference provides a series of methods for managing and controlling online meeting features. By implementing this interface, developers can easily integrate online meeting features into their applications.

### getRoomEngine

Access the roomEngine instance. If roomEngine does not exist, it will return null.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
const roomEngine = conference.getRoomEngine();
```

Returns: *TUIRoomEngine* | *null*

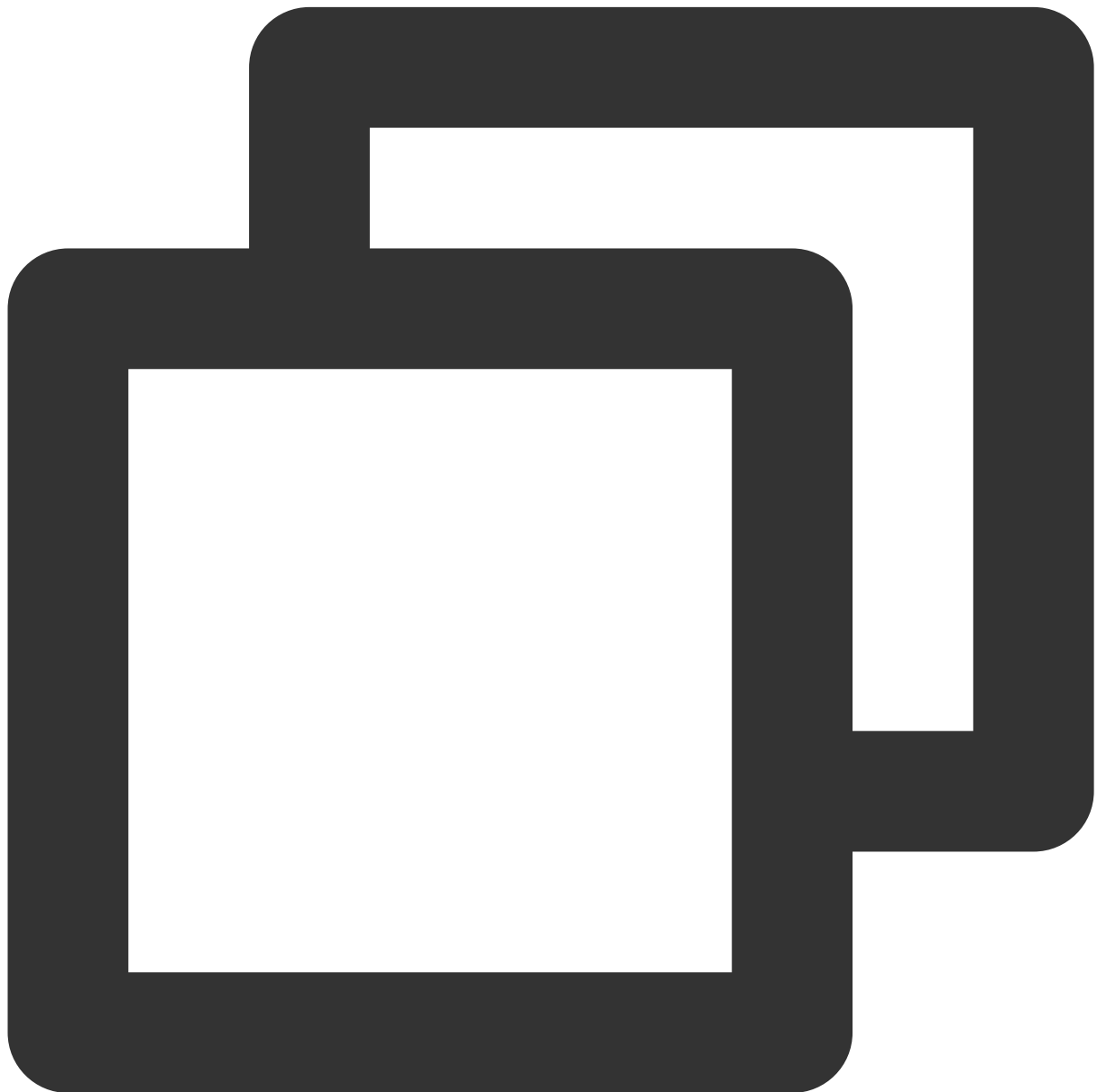
## on

Listen for a specified type of event. When the event occurs, the callback function will be called.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
eventType	<a href="#">RoomEvent</a>	-	Event Type to Listen For
callback	() => void	-	Callback Function Called When the Event Occurs



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference, RoomEvent } from '@tencentcloud/roomkit-web-vue3';
```



```
conference.on(RoomEvent.RoomStart, () => {
  console.log('[conference] The meeting has already started.')
});
conference.on(RoomEvent.ROOM_DISMISS, () => {
  console.log('[conference] The meeting has been dismissed')
});
```

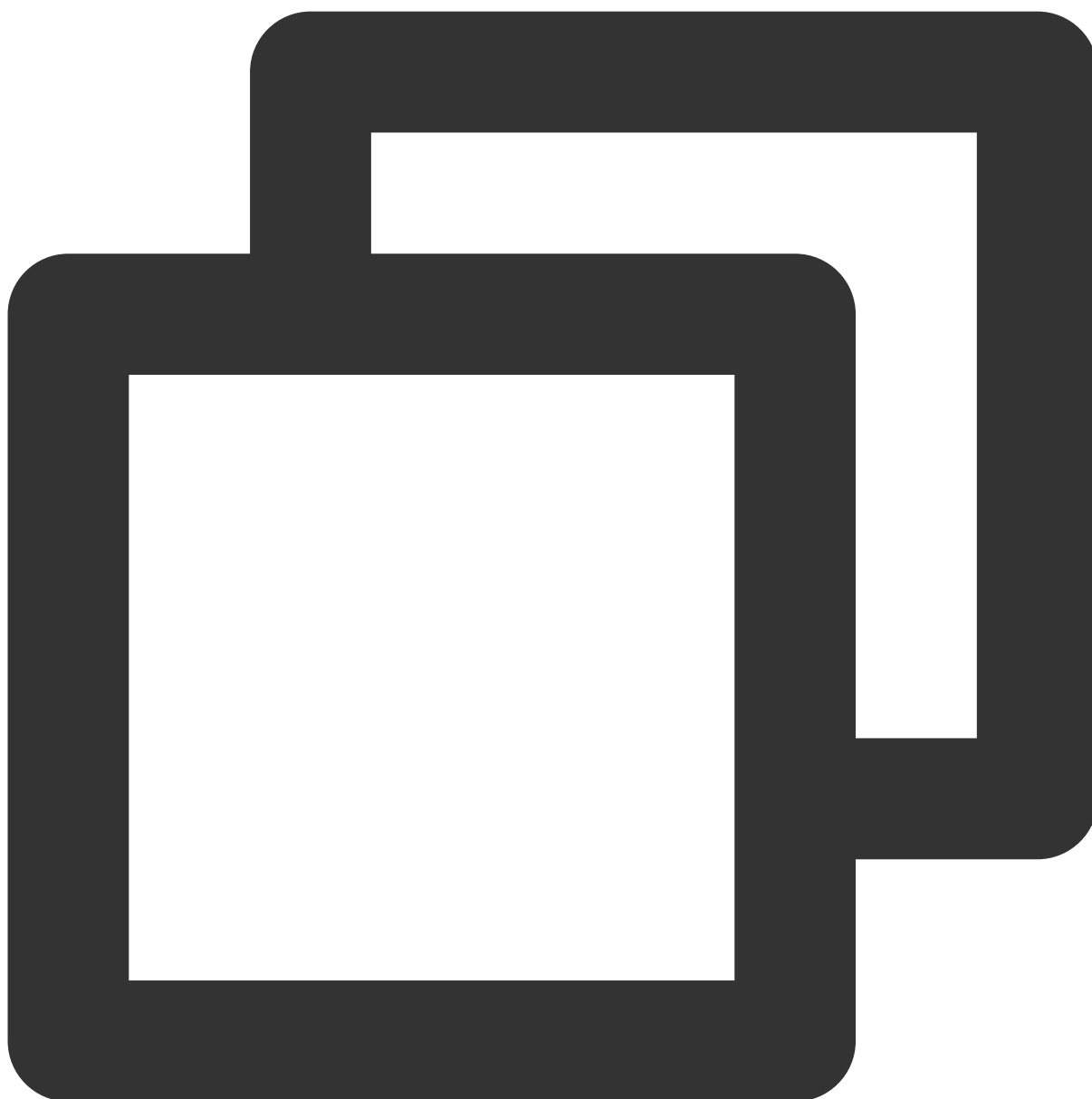
Returns: *void*

## off

Cancel listening for events of a specified type.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
eventType	<a href="#">RoomEvent</a>	-	The type of event to cancel listening for
callback	() => void	-	The callback function added previously



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.off('event', callback);
```

Returns: *void*

## login

Log in to the conference system.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
params	{sdkAppld: number; userId: string; userSig: string; tim?: ChatSDK}	-	Log in to the parameter object
sdkAppld	number	-	In the <a href="#">Real-time Audio and Video Console</a> click <b>Application Management &gt; create an application</b> . After creating a new application, you can access the sdkAppld information in <b>Application Information</b> .
userId	string	-	It's advised to limit the User ID length to 32 bytes, allowing only uppercase and lowercase letters (a-zA-Z), digits (0-9), underscores, and hyphens.
userSig	string	-	userSig Signature For the method of calculating userSig, please refer to <a href="#">UserSig Related</a> .
tim	ChatSDK (optional)	-	If you want to leverage more capabilities of the Instant Messaging SDK while integrating roomEngine, you can pass the created tim instance into TUIRoomEngine. For how to create a tim instance, please refer to <a href="#">TIM.create</a> .

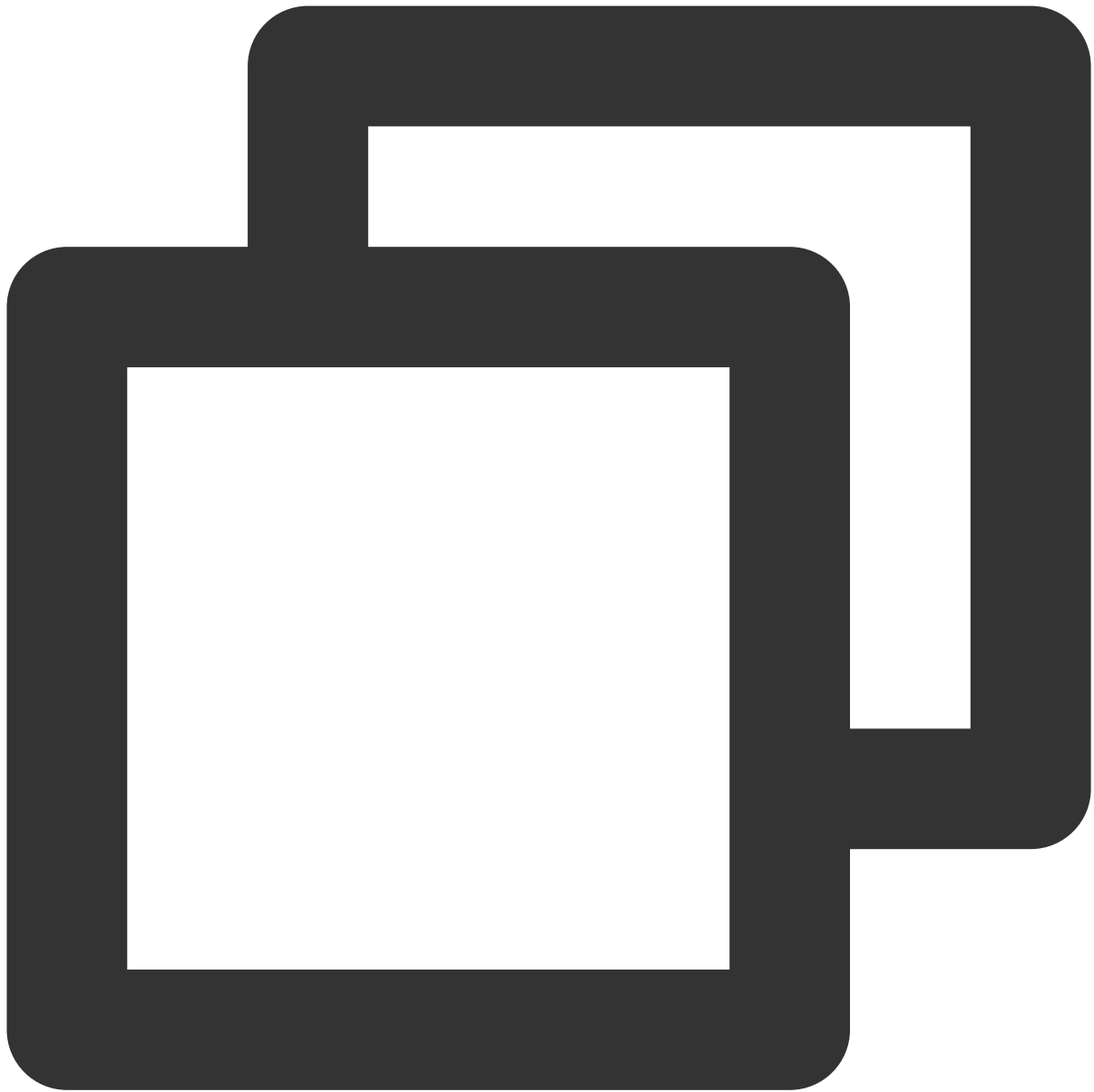


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.login({
  sdkAppId: 123456,
  userId: 'testUser',
  userSig: 'testSig'
});
```

Returns:*Promise<void>*

## logout

Log out of the meeting system.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.logout();
```

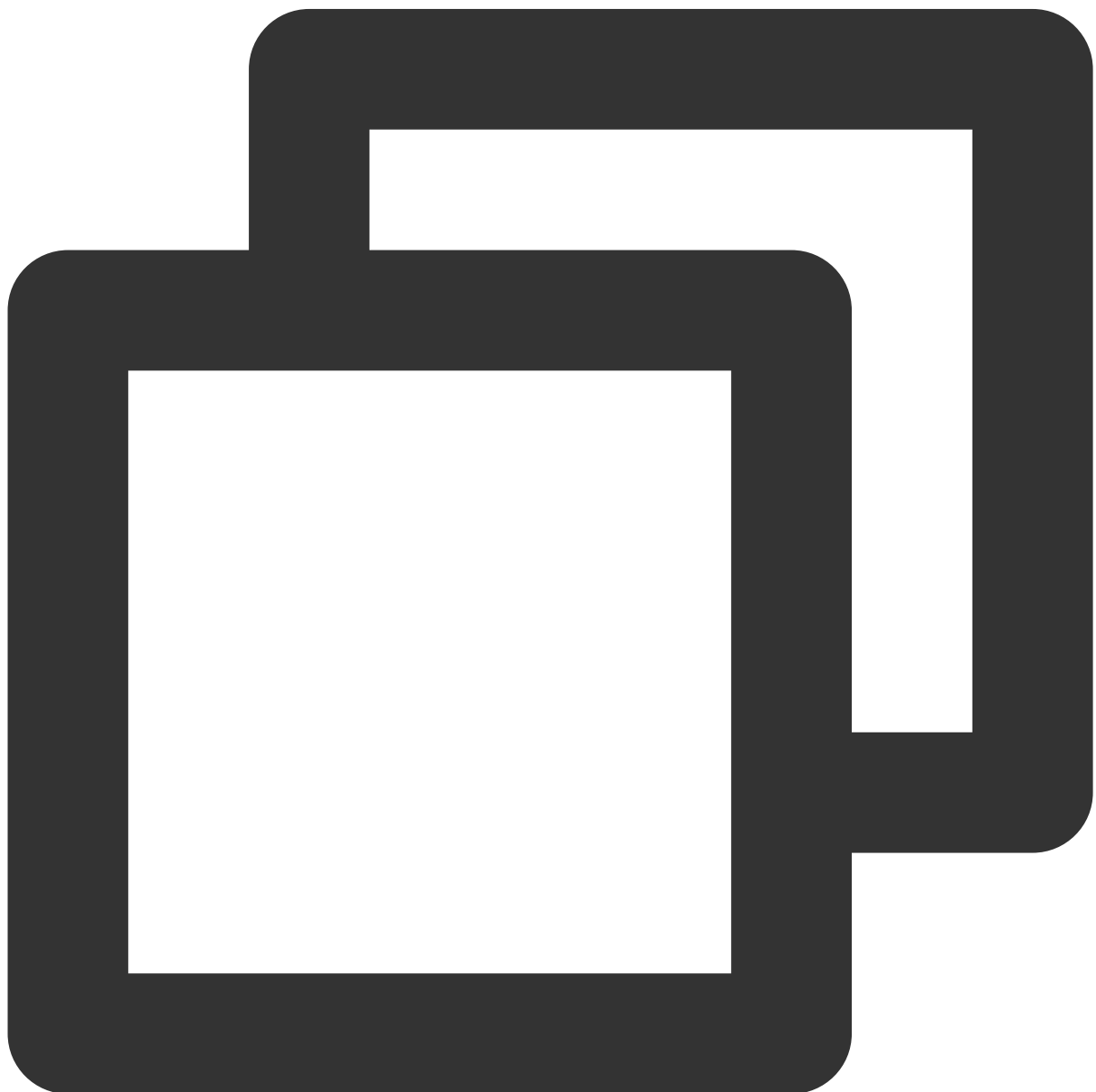
Returns:*Promise<void>*

## start

Start a new meeting.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
roomId	string	-	Meeting Room ID
params	<a href="#">StartParams</a>	-	Parameters for starting a meeting



```
// Note the package name. If you are using the vue2 version, please change the pack
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns:*Promise*<void>

## join

Join an existing meeting.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
roomId	string	-	Meeting Room ID
params	<a href="#">JoinParams</a>	-	Parameters for joining the meeting



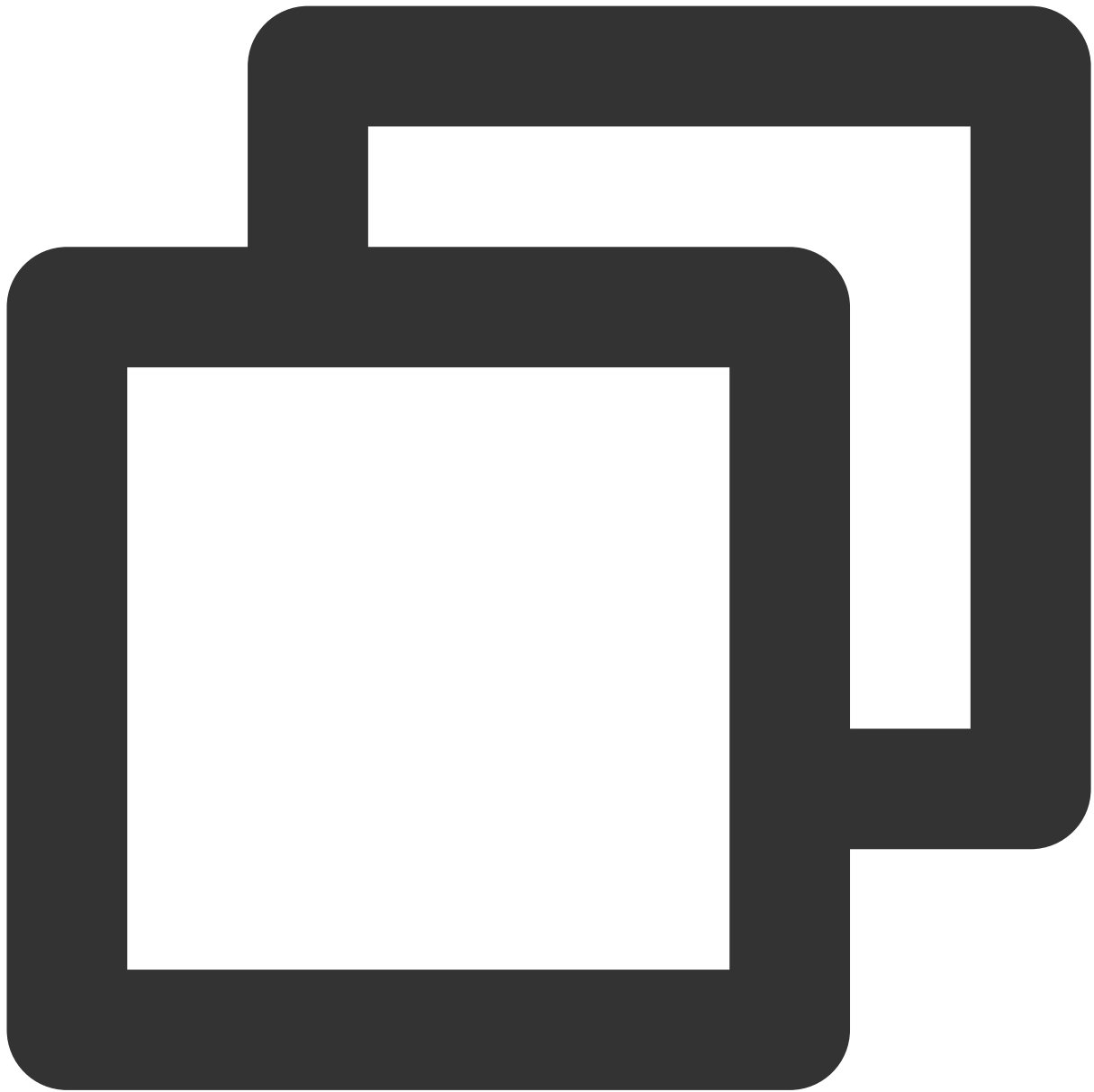
```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.join('123456', {
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns:*Promise<void>*

**leave**



Leave the current meeting.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.leave();
```

Returns:*Promise<void>*

## dismiss

Dismiss the current meeting.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.dismiss();
```

Returns:*Promise<void>*

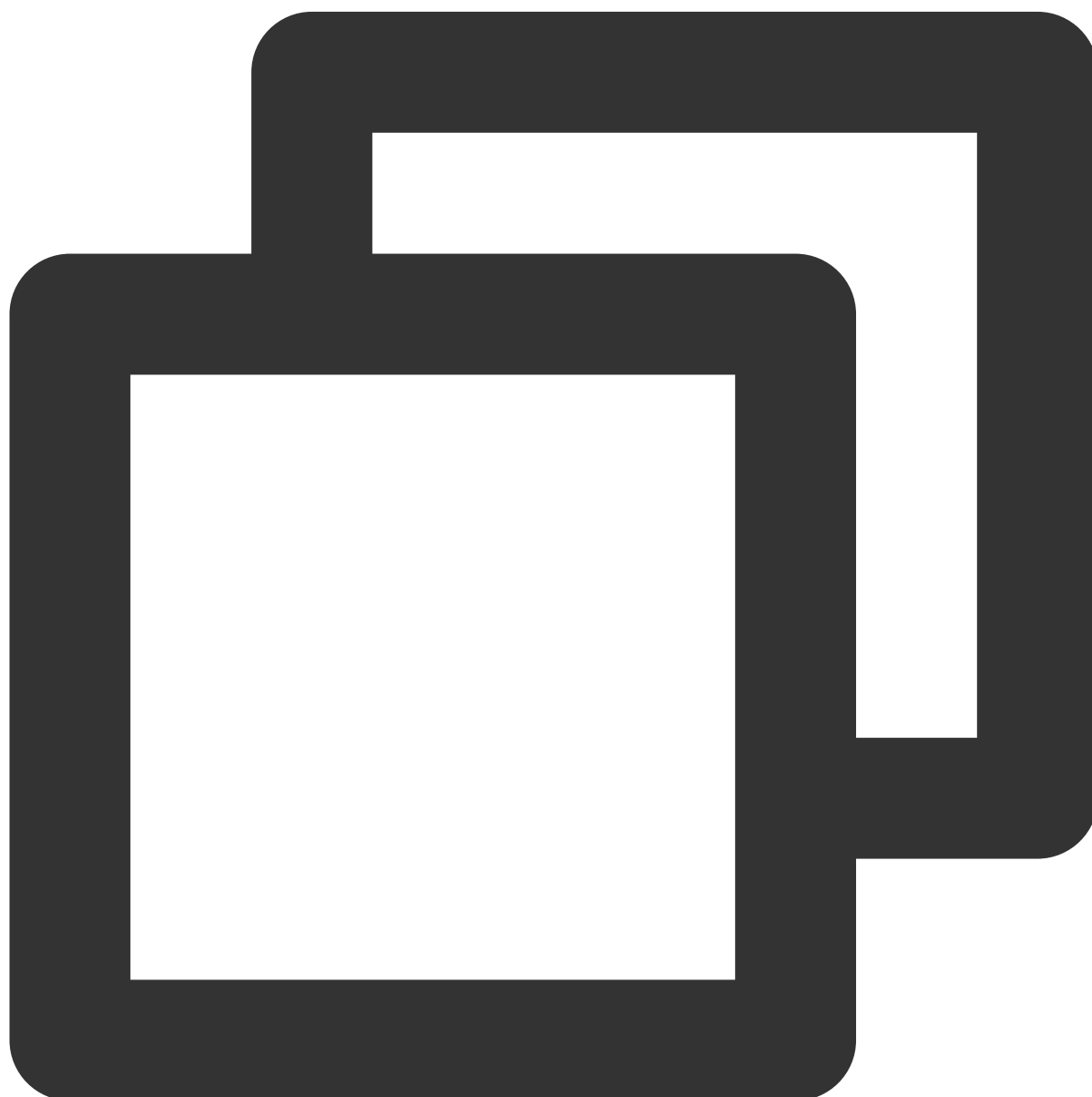
## setSelfInfo

Set your user information.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
options	{userName: string; avatarUrl: string}	-	User information object
userName	string (Optional)	-	User's nickname
avatarUrl	string (Optional)	-	User profile photo



```
// Note the package name. If you are using the vue2 version, please change the pack
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setSelfInfo({
  userName: 'test-name',
  avatarUlr: 'https://avatar.png'
});
```

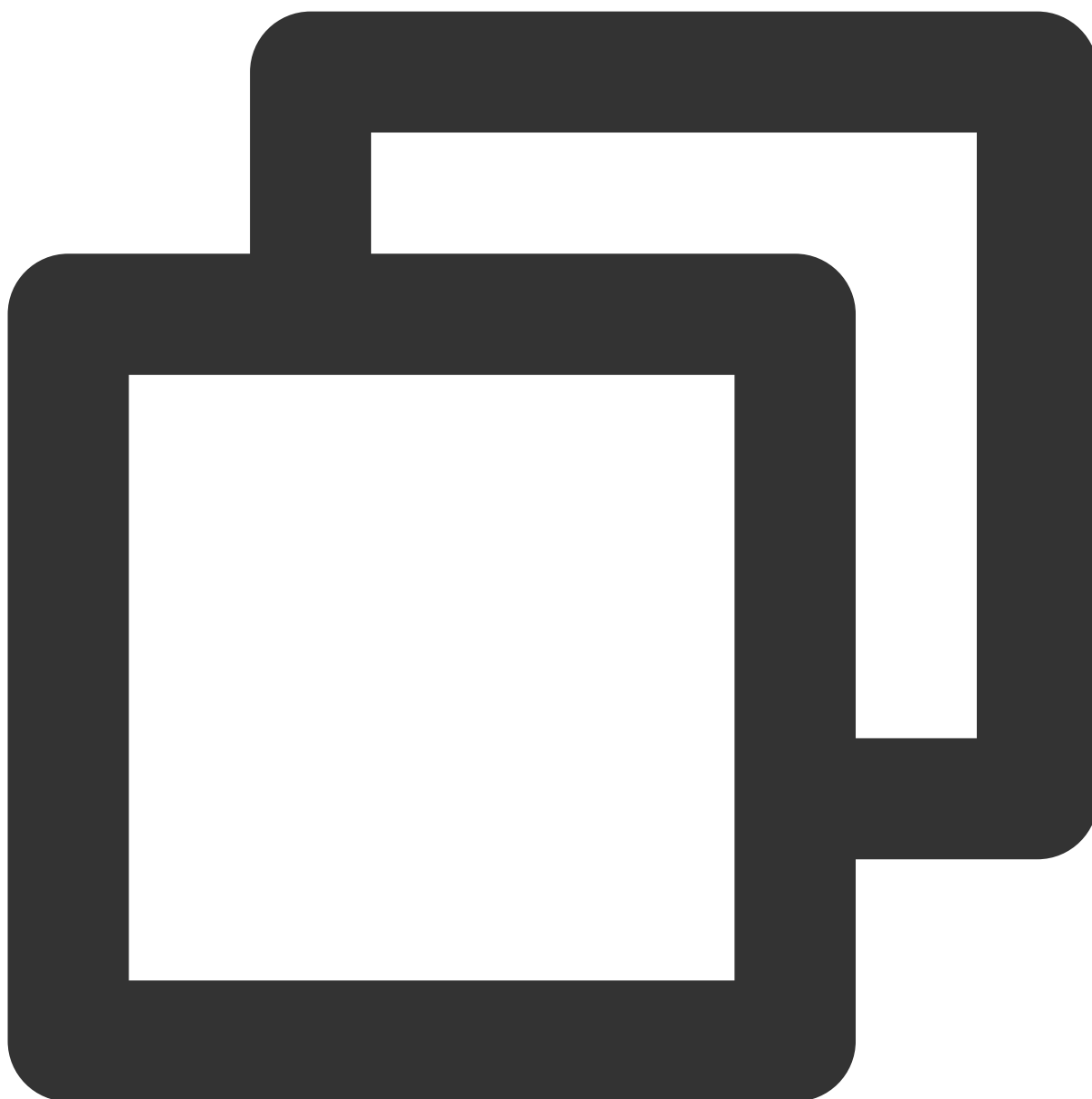
Returns:*Promise<void>*

## setLanguage

Set the interface language.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
language	'zh-CN'   'en-US'	-	Language



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setLanguage('en-US');
```

Returns: *void*

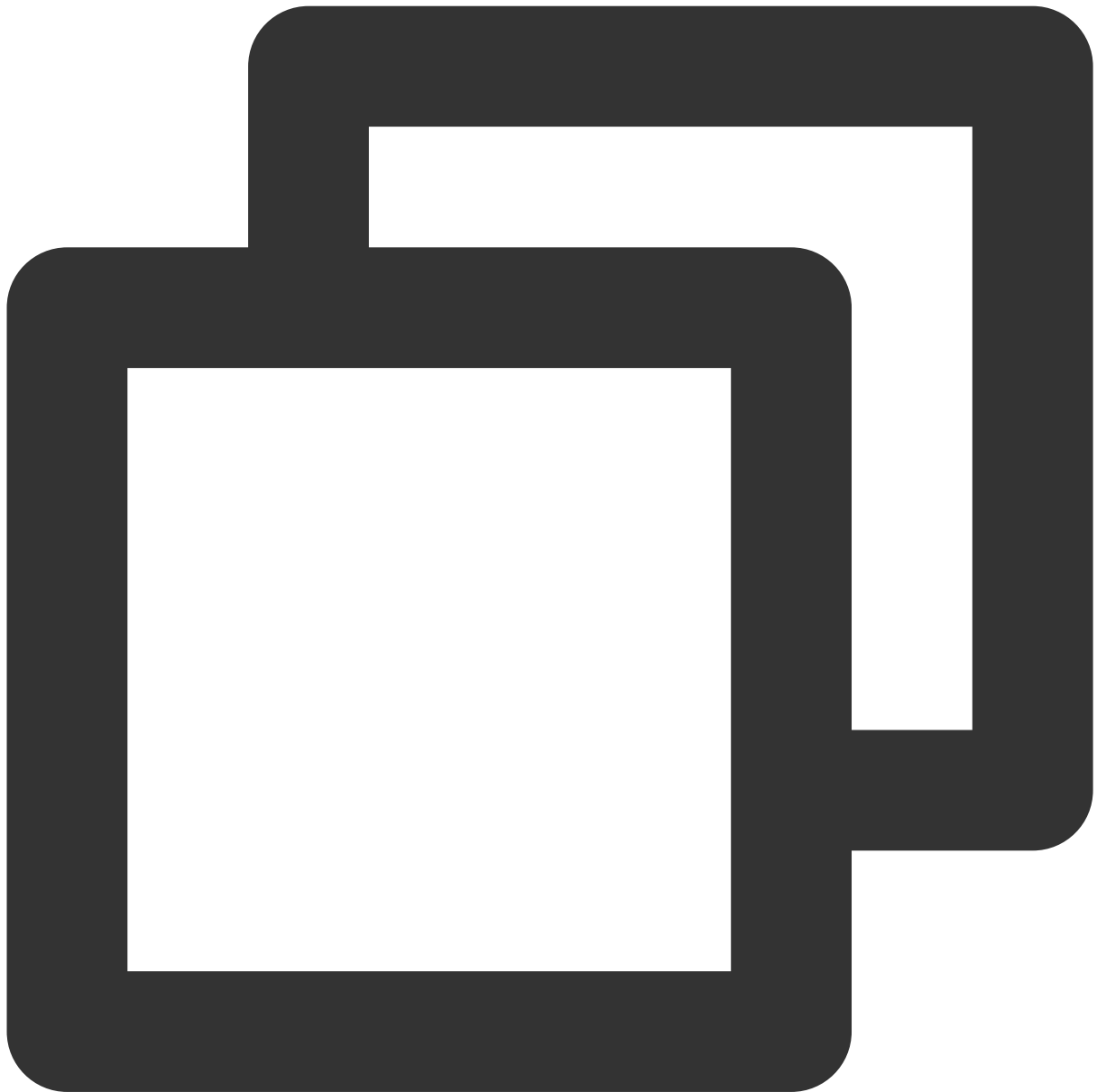
## setTheme

Set the interface topic.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
theme	'LIGHT'   'DARK'	-	Topic Type

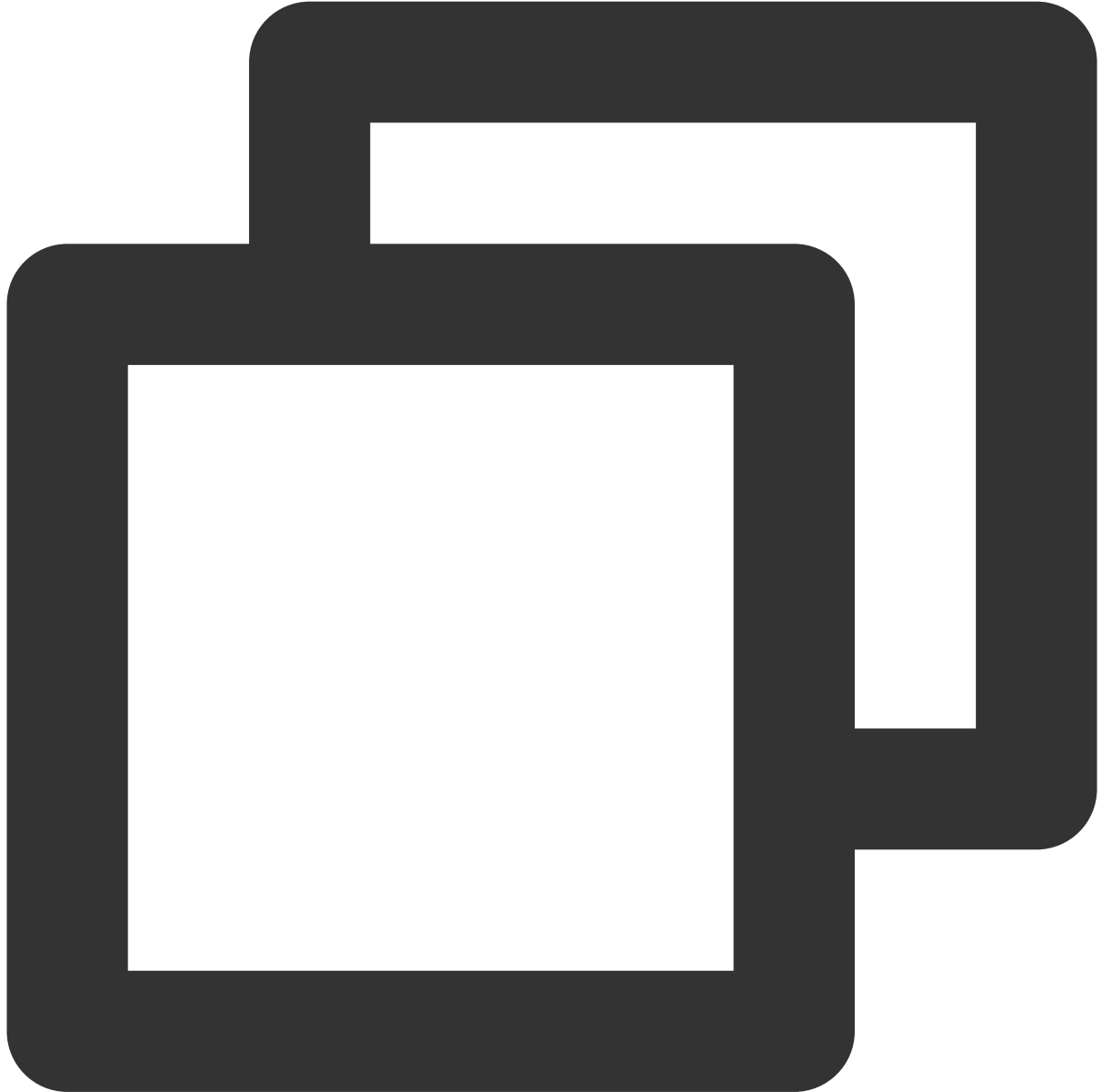


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setTheme('DARK');
```

Returns:*void*

## enableWatermark

Enable the text messaging feature in the application. For details, see: [Text Watermark](#).

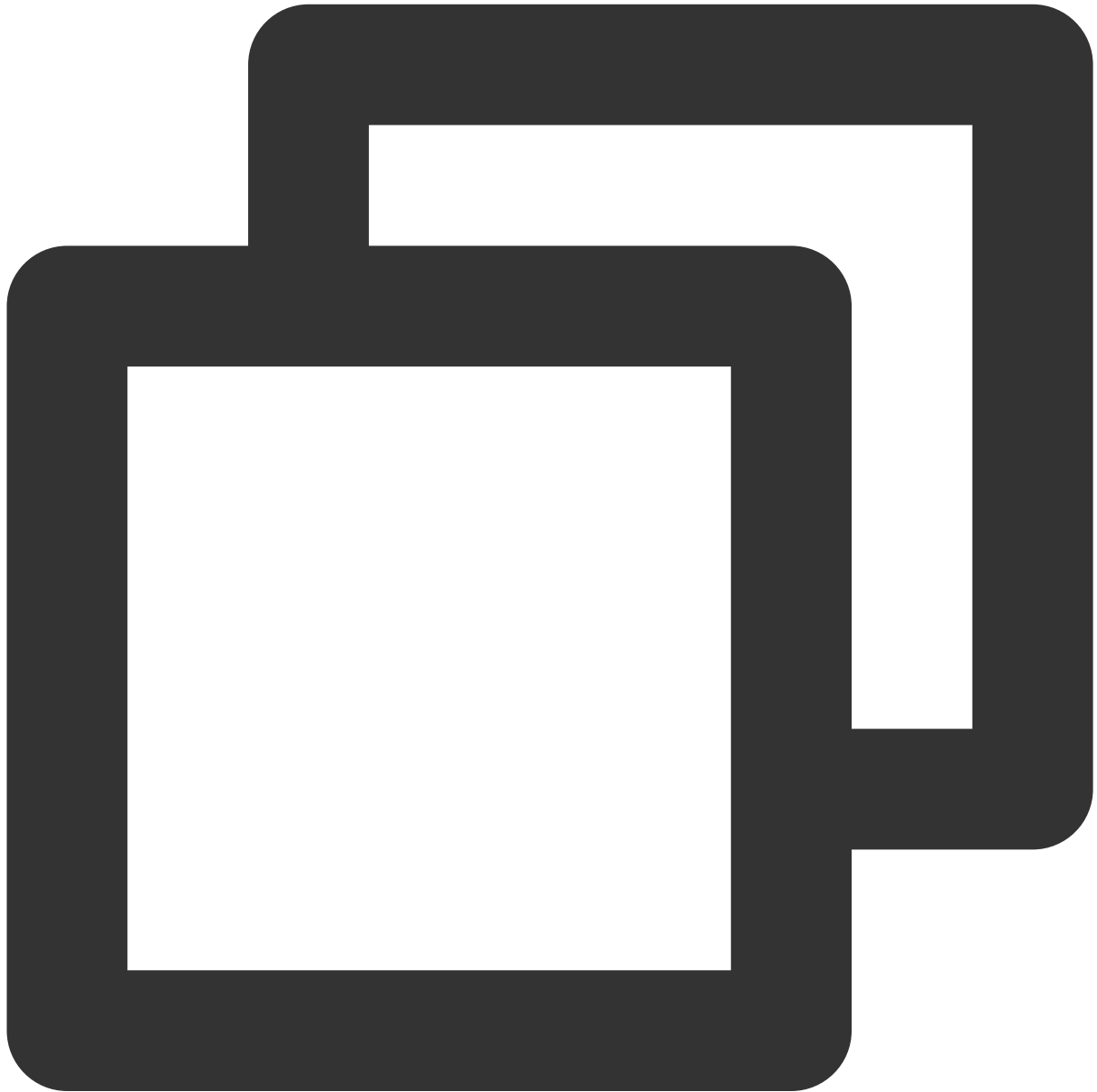


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableWatermark();
```

Returns : *void*

## enableVirtualBackground

Enable the virtual background feature in the application. After calling this function, a virtual background feature button will be displayed on the UI. For details, see: [Virtual Background](#).



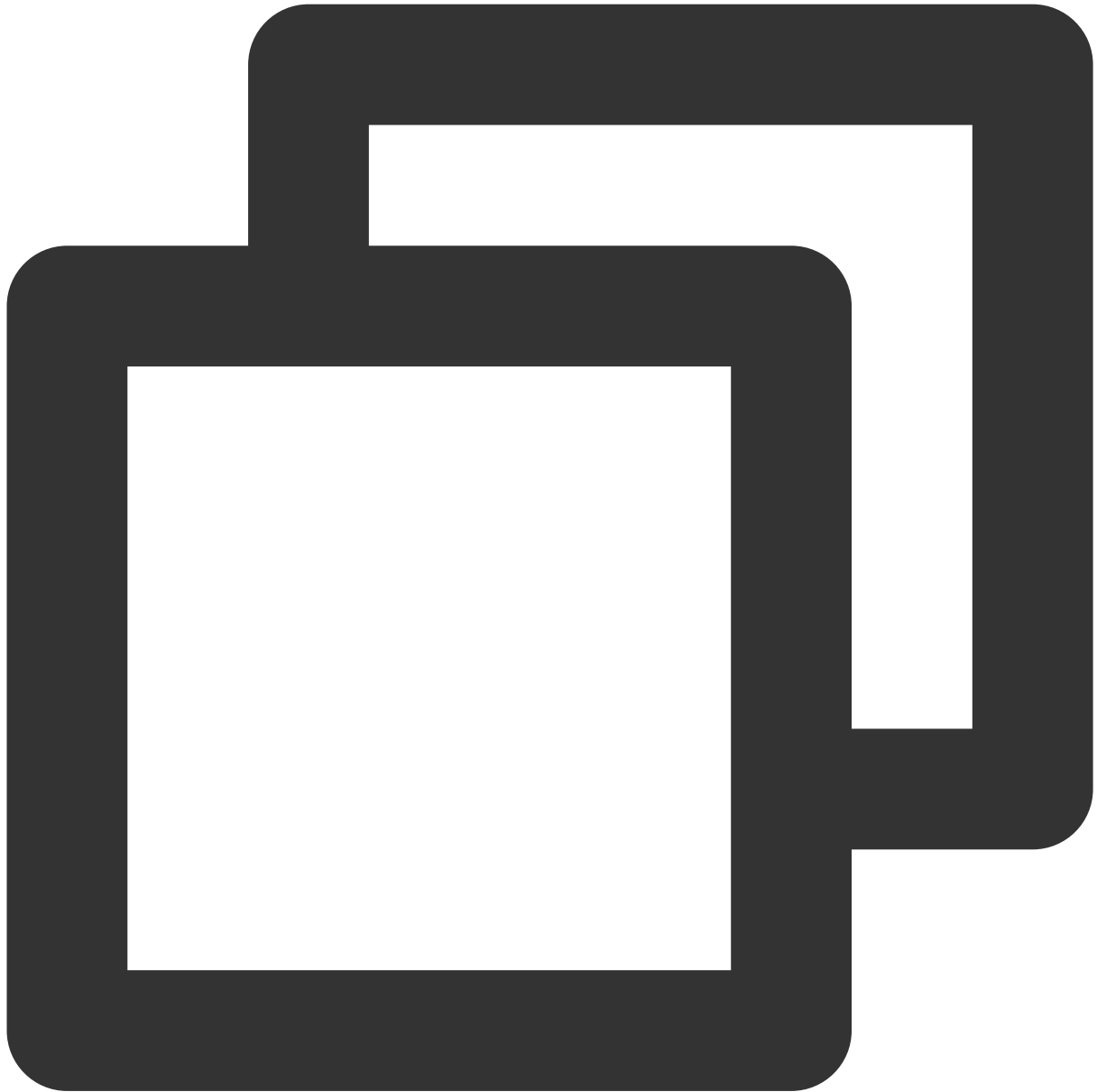
```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableVirtualBackground();
```

Returns : *void*

### **disableTextMessaging**



Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.

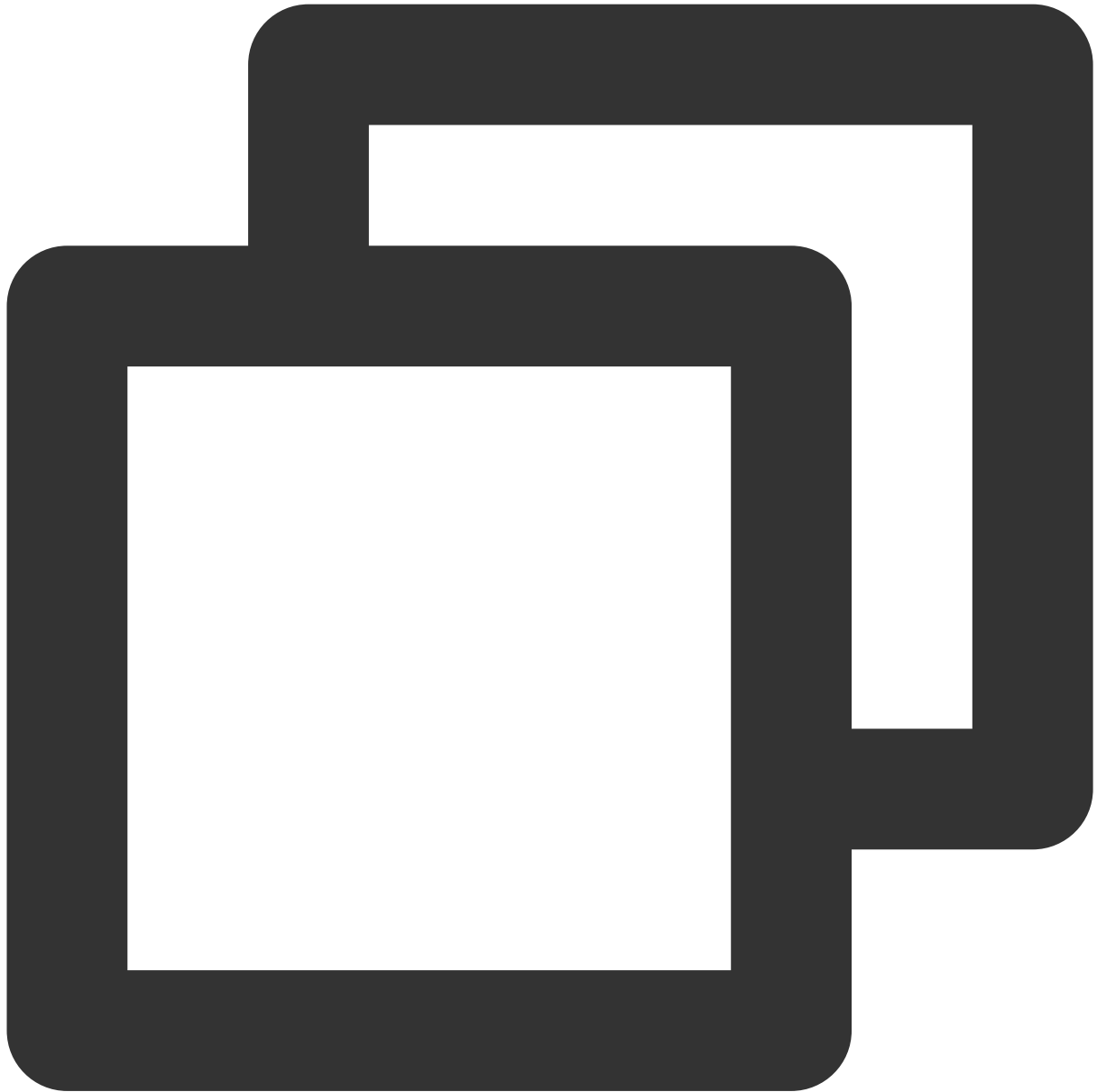


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableTextMessaging();
```

Returns: *void*

## **disableScreenSharing**

Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableScreenSharing();
```

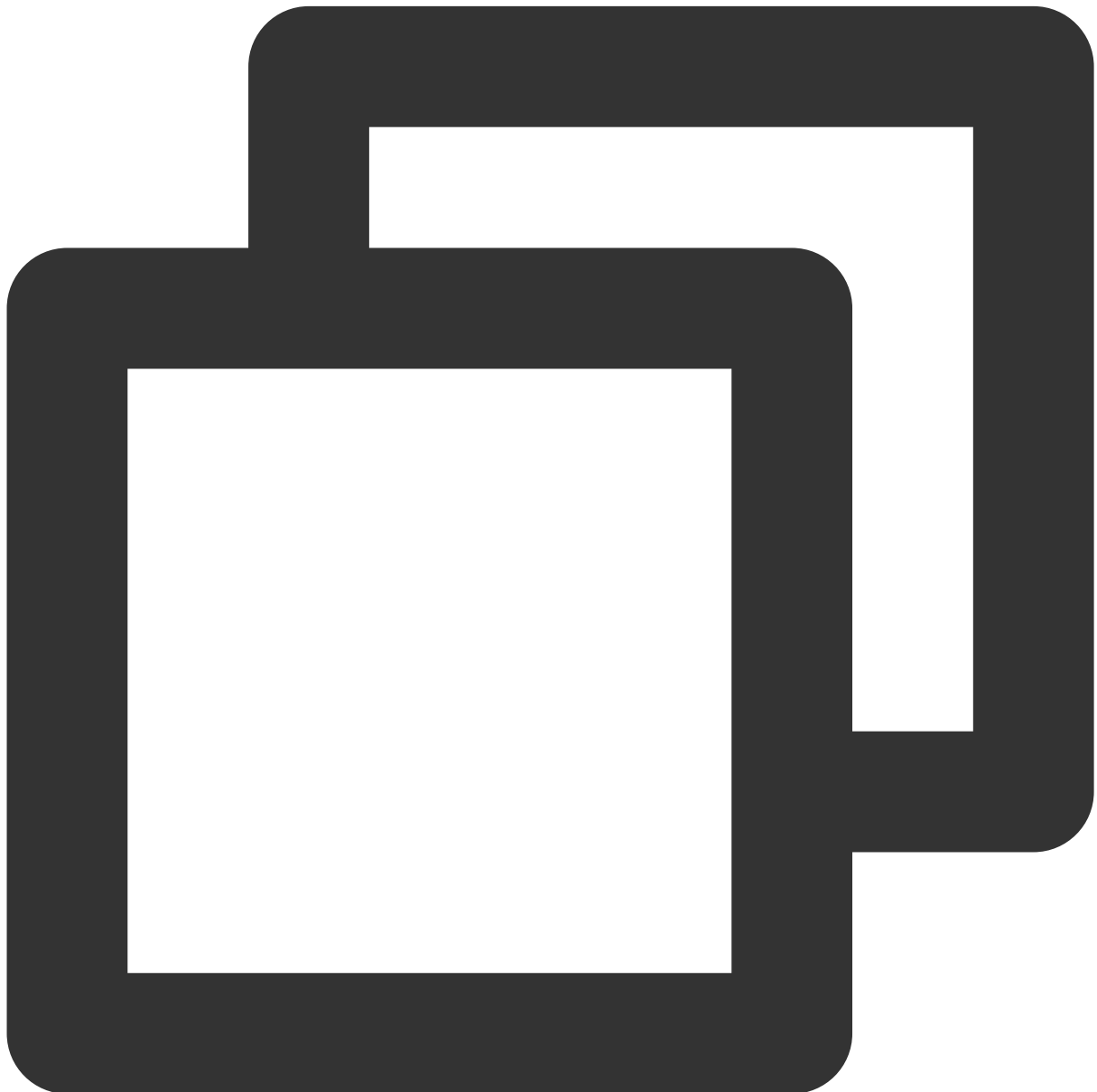
Returns: *void*

## hideFeatureButton

Hide specific feature buttons in the application. By invoking this function and passing in the appropriate [FeatureButton](#) enumerated values, the corresponding buttons will be hidden from the user interface.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
name	<a href="#">FeatureButton</a>	-	Names of Buttons to be Hidden



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue3';
```

```
conference.hideFeatureButton (FeatureButton.SwitchTheme);
```

Returns: *void*

## Type Definition

### RoomEvent (enumeration)

Parameter	Type	Description
ROOM_START	string	Creating a Meeting
ROOM_JOIN	string	Joining a Meeting
ROOM_LEAVE	string	Leave Meeting
ROOM_DISMISS	string	Meeting Dismissed
KICKED_OFFLINE	string	User kicked offline
KICKED_OUT	string	Participant removed from meeting
USER_LOGOUT	string	User logged out
ROOM_ERROR	string	Meeting error

### FeatureButton (enumerated values)

Parameter	Type	Description
SwitchTheme	string	Switch Topic Feature Button
SwitchLayout	string	Switch Layout Feature Button
SwitchLanguage	string	Switch Language Feature Button
FullScreen	string	Fullscreen Feature Button
Invitation	string	Invite Feature Button

### StartParams

Parameter	Type	Description	Default Value
roomName	string (Optional)	Room Name	-

isSeatEnabled	boolean (optional)	Enable Seats	false
isOpenCamera	boolean (optional)	Whether to enable the camera	false
isOpenMicrophone	boolean (optional)	Whether to enable the microphone	false
defaultCameraId	string (Optional)	Default Camera ID	-
defaultMicrophoneId	string (Optional)	Default Microphone ID	-
defaultSpeakerId	string (Optional)	Default Speaker ID	-

## JoinParams

Parameter	Type	Description	Default Value
isOpenCamera	boolean (optional)	Whether to enable the camera	false
isOpenMicrophone	boolean (optional)	Whether to enable the microphone	false
defaultCameraId	string (Optional)	Default Camera ID	-
defaultMicrophoneId	string (Optional)	Default Microphone ID	-
defaultSpeakerId	string (Optional)	Default Speaker ID	-

# RoomEngine API

## API Overview

Last updated : 2024-04-29 17:21:12

### TUIRoomKit (UI included Component)

#### TUIRoomKit API List

API	Description
<a href="#">getRoomEngine</a>	Get the roomEngine instance. If the roomEngine does not exist, return null.
<a href="#">on</a>	Listen for events of a specified type. When the event occurs, the callback function will be called.
<a href="#">off</a>	Stop listening for events of a specified type.
<a href="#">login</a>	Log in to the conference system.
<a href="#">logout</a>	Log out of the meeting system.
<a href="#">start</a>	Start a new meeting.
<a href="#">join</a>	Join an existing meeting.
<a href="#">leave</a>	Leave the current meeting.
<a href="#">dismiss</a>	Dismiss the current meeting.
<a href="#">setSelfInfo</a>	Set your user information.
<a href="#">setLanguage</a>	Set the interface language.
<a href="#">setTheme</a>	Set the interface topic.
<a href="#">disableTextMessaging</a>	Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.
<a href="#">disableScreenSharing</a>	Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.
<a href="#">hideFeatureButton</a>	Hide specific feature buttons in the application. By invoking this function and passing in the appropriate <a href="#">FeatureButton</a> enumerated values, the corresponding buttons will be hidden from the user interface.

## TUIRoomEngine (No UI interface)

### TUIRoomEngine API List

#### TUIRoomEngine Static method

API	Description
<a href="#">once</a>	Listen to TUIRoomEngine ready event. <b>Note: All methods except TUIRoomEngine.init must be executed after listening to the TUIRoomEngine ready event and the TUIRoomEngine.init method is executed successfully.</b>
<a href="#">login</a>	Login to TUIRoomEngine
<a href="#">setSelfInfo</a>	Set the current user's basic information (username, user avatar)
<a href="#">getSelfInfo</a>	Get the current user's basic information (username, user avatar)
<a href="#">logout</a>	Logout from TUIRoomEngine

#### RoomEngine Room Management API

API	Description
<a href="#">createRoom</a>	Create room
<a href="#">enterRoom</a>	Entered room
<a href="#">destroyRoom</a>	Close the room
<a href="#">exitRoom</a>	Leave room
<a href="#">fetchRoomInfo</a>	Get room data
<a href="#">updateRoomNameByAdmin</a>	Update the room's name (Only group owner or administrator can call)
<a href="#">updateRoomSpeechModeByAdmin</a>	Update the room's speech mode (Only group owner or administrator can call)
<a href="#">getUserList</a>	Get the current room user list
<a href="#">getUserInfo</a>	Get user Learn more

**roomEngine Audio Video API**

API	Description
<a href="#">setLocalVideoView</a>	Set the view control for local user video rendering
<a href="#">openLocalCamera</a>	Open local camera
<a href="#">closeLocalCamera</a>	Close local camera
<a href="#">openLocalMicrophone</a>	Open local microphone
<a href="#">closeLocalMicrophone</a>	Close local microphone
<a href="#">updateVideoQuality</a>	Update local video codec quality settings
<a href="#">updateAudioQuality</a>	Update local audio codec quality settings
<a href="#">startScreenSharing</a>	Start screen sharing
<a href="#">stopScreenSharing</a>	End screen sharing
<a href="#">startPushLocalVideo</a>	Start pushing local video
<a href="#">stopPushLocalVideo</a>	Stop pushing local video
<a href="#">startPushLocalAudio</a>	Start pushing local audio
<a href="#">stopPushLocalAudio</a>	Stop pushing local audio
<a href="#">setRemoteVideoView</a>	Set the view control for remote user video rendering
<a href="#">startPlayRemoteVideo</a>	Start playing remote user video
<a href="#">stopPlayRemoteVideo</a>	Stop playing remote user video
<a href="#">muteRemoteAudioStream</a>	Mute remote user

**roomEngine Member Management API**

API	Description
<a href="#">openRemoteDeviceByAdmin</a>	Request remote user to open media device
<a href="#">applyToAdminToOpenLocalDevice</a>	Participant applies to the host to open the device
<a href="#">closeRemoteDeviceByAdmin</a>	Close remote user's media device
<a href="#">cancelRequest</a>	Cancel the sent request



<a href="#">responseRemoteRequest</a>	Reply to remote user's request
<a href="#">changeUserRole</a>	Change user's role
<a href="#">kickRemoteUserOutOfRoom</a>	Kick user out of the room
<a href="#">disableDeviceForAllUserByAdmin</a>	Disable/Enable all users' media devices
<a href="#">disableSendingMessageForAllUser</a>	Disable/Enable all users to send messages
<a href="#">disableSendingMessageByAdmin</a>	Disable/Enable a user to send messages

### roomEngine Microphone Management API

API	Description
<a href="#">setMaxSeatCount</a>	Set room microphone position maximum value
<a href="#">getSeatList</a>	Get microphone position information
<a href="#">takeSeat</a>	Get microphone position
<a href="#">leaveSeat</a>	Release microphone position
<a href="#">takeUserOnSeatByAdmin</a>	Invite others to go live (only room host and administrator can call this method)
<a href="#">kickUserOffSeatByAdmin</a>	Kick others off the microphone position (only room host and administrator can call this method)
<a href="#">lockSeatByAdmin</a>	Lock a microphone position status (only room host and administrator can call this method)

### roomEngine Message Sending API

API	Description
<a href="#">sendTextMessage</a>	Send text message
<a href="#">sendCustomMessage</a>	Send custom message

### roomEngine Device Management API

API	Description
<a href="#">getCameraDevicesList</a>	Get camera device list
<a href="#">getMicDevicesList</a>	Get mic device list

<a href="#">getSpeakerDevicesList</a>	Get speaker device list
<a href="#">setCurrentCameraDevice</a>	Set the camera device to use
<a href="#">setCurrentMicDevice</a>	Set the mic device to use
<a href="#">setCurrentSpeakerDevice</a>	Set the speaker device to use
<a href="#">getCurrentCameraDevice</a>	Get the currently used camera device
<a href="#">getCurrentMicDevice</a>	Get the currently used mic device
<a href="#">getCurrentSpeakerDevice</a>	Get the currently used speaker device
<a href="#">startCameraDeviceTest</a>	Start camera device test
<a href="#">stopCameraDeviceTest</a>	Stop camera device test

### roomEngine Event Listening API

API	Description
<a href="#">on</a>	Listen to <a href="#">TUIRoomEvents</a> event
<a href="#">off</a>	Cancel listening to <a href="#">TUIRoomEvents</a> event

### roomEngine Other API

API	Description
<a href="#">getTRTCCloud</a>	Get trtcCloud instance
<a href="#">getTIM</a>	Get tim instance

### TUIRoomEngine Event Type

TUIRoomEvent is the Callback Event class corresponding to TUIRoomEngine. You can listen to the Callback Events you are interested in through this callback.

EVENT	Description
<a href="#">TUIRoomEvents.onError</a>	Error Event
<a href="#">TUIRoomEvents.onKickedOutOfRoom</a>	Kick out of room event
<a href="#">TUIRoomEvents.onKickedOffline</a>	User Kicked Offline Event

<a href="#">TUIRoomEvents.onUserSigExpired</a>	User Credential Timeout Event
<a href="#">TUIRoomEvents.onRoomDismissed</a>	Room Dismissed Event
<a href="#">TUIRoomEvents.onRoomNameChanged</a>	Room Name Change Event
<a href="#">TUIRoomEvents.onRoomSpeechModeChanged</a>	Room Microphone Control Mode Change
<a href="#">TUIRoomEvents.onAllUserCameraDisableChanged</a>	All Users' Cameras Disabled in Room Event
<a href="#">TUIRoomEvents.onAllUserMicrophoneDisableChanged</a>	All Users' Microphones Disabled in Room Event
<a href="#">TUIRoomEvents.onSendMessageForAllUserDisableChanged</a>	All Users' Text Message Sending Disabled in Room Event
<a href="#">TUIRoomEvents.onRoomMaxSeatCountChanged</a>	Maximum number of microphone slots in the room modification event
<a href="#">TUIRoomEvents.onRemoteUserEnterRoom</a>	Remote user Entered room event
<a href="#">TUIRoomEvents.onRemoteUserLeaveRoom</a>	Remote user leaving room event
<a href="#">TUIRoomEvents.onUserRoleChanged</a>	Role change event
<a href="#">TUIRoomEvents.onUserVideoStateChanged</a>	Video Status change event
<a href="#">TUIRoomEvents.onUserAudioStateChanged</a>	Audio Status change event
<a href="#">TUIRoomEvents.onSendMessageForUserDisableChanged</a>	Send message status event
<a href="#">TUIRoomEvents.onUserVoiceVolumeChanged</a>	Volume change event
<a href="#">TUIRoomEvents.onUserNetworkQualityChanged</a>	Network quality change event
<a href="#">TUIRoomEvents.onSeatListChanged</a>	Mic position list change event
<a href="#">TUIRoomEvents.onKickedOffSeat</a>	Kicked off the mic event
<a href="#">TUIRoomEvents.onRequestReceived</a>	Request received event
<a href="#">TUIRoomEvents.onRequestCancelled</a>	Request cancelled event
<a href="#">TUIRoomEvents.onReceiveTextMessage</a>	Receive text message event
<a href="#">TUIRoomEvents.onReceiveCustomMessage</a>	Receive custom message event
<a href="#">TUIRoomEvents.onDeviceChange</a>	Device change event
<a href="#">TUIRoomEvents.onUserScreenCaptureStopped</a>	Screen sharing stopped event

When a user uses the built-in browser's **stop sharing** button to end Screen Sharing, the user will receive the 'onUserScreenCaptureStopped' event to modify the Sharing status.

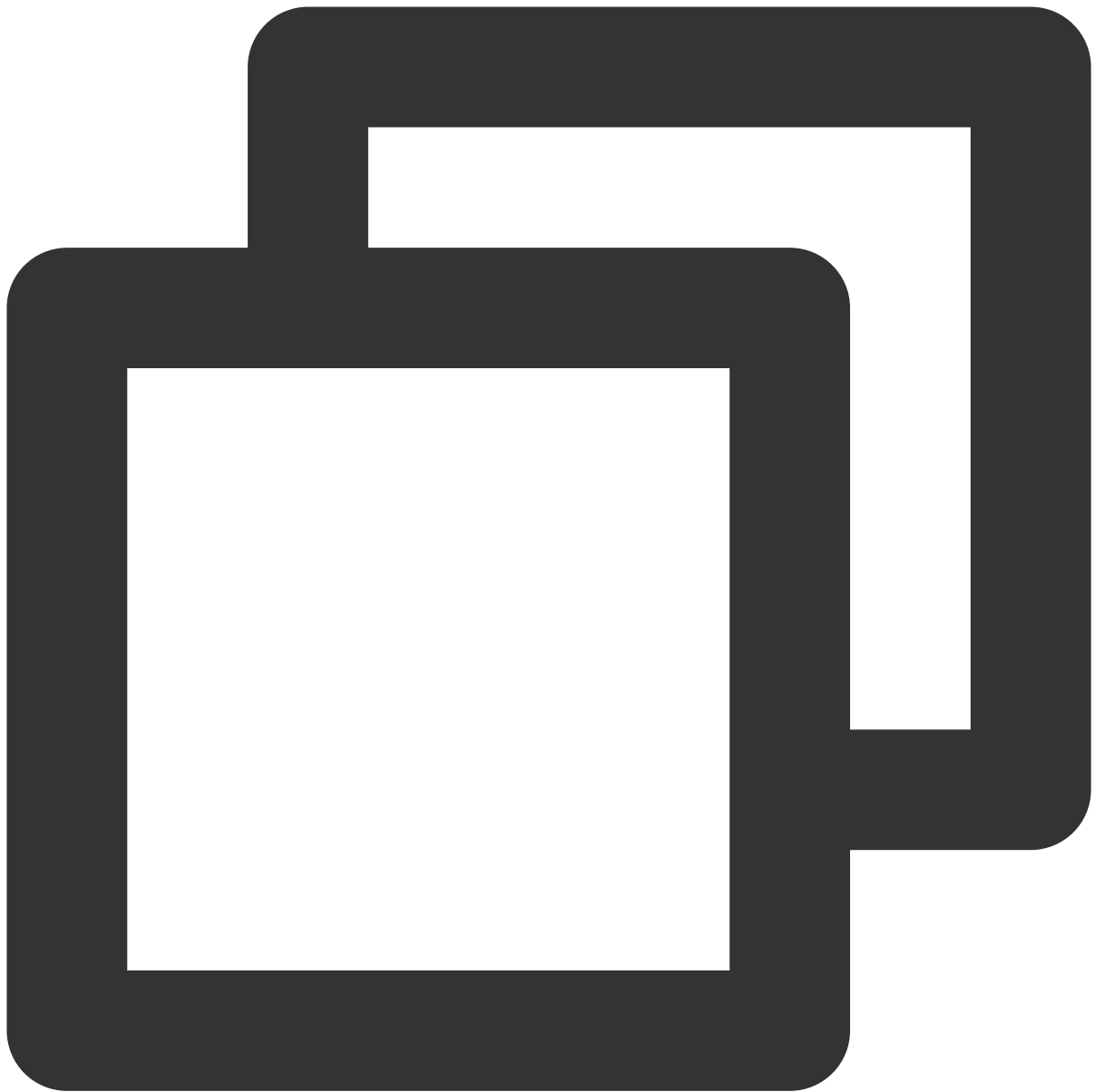
# TUIRoomEngine

Last updated : 2024-01-10 17:26:33

## TUIRoomEngine Introduction

TUIRoomEngine SDK provides Room Management, Multi-person Tencent Real-Time Communication, Screen Sharing, Member Management, Instant Messaging, and other features.

**Installation Method:**



```
// Use npm
npm i @tencentcloud/tuiroom-engine-js --save

// Use pnpm
pnpm i @tencentcloud/tuiroom-engine-js --save

// Use yarn
yarn add @tencentcloud/tuiroom-engine-js
```

# TUIRoomEngine API

## TUIRoomEngine Static Method

API	Description
<a href="#">once</a>	<b>Listening to the TUIRoomEngine ready Event.</b> <b>Note: All methods other than TUIRoomEngine.login must be executed after listening to the TUIRoomEngine ready event and the successful execution of the TUIRoomEngine.login method.</b>
<a href="#">login</a>	Login to TUIRoomEngine
<a href="#">setSelfInfo</a>	Setting the current user's Basic information (Username, User Avatar)
<a href="#">getSelfInfo</a>	Get the current user's Basic information (Username, User Avatar)
<a href="#">logout</a>	Logout of TUIRoomEngine

## roomEngine Room Management API

API	Description
<a href="#">createRoom</a>	Create Room
<a href="#">enterRoom</a>	Enter Room
<a href="#">destroyRoom</a>	Destroy Room
<a href="#">exitRoom</a>	Exit Room
<a href="#">fetchRoomInfo</a>	Get Room data
<a href="#">updateRoomNameByAdmin</a>	Update Name (Call by Group Owner or Administrator only)
<a href="#">updateRoomSpeechModeByAdmin</a>	Update Speaking Mode (Call by Group Owner or Administrator only)
<a href="#">getUserList</a>	Get User List
<a href="#">getUserInfo</a>	Learn more about the user

## roomEngine Audio Video API

API	Description

<a href="#">setLocalVideoView</a>	Set Local Stream Rendering Position
<a href="#">openLocalCamera</a>	Capturing Local Camera Video streams
<a href="#">closeLocalCamera</a>	Close Local Camera
<a href="#">openLocalMicrophone</a>	Open Local mic
<a href="#">closeLocalMicrophone</a>	Close Local mic
<a href="#">updateVideoQuality</a>	Set Local Video Parameters
<a href="#">updateAudioQuality</a>	Set Local Audio Parameters
<a href="#">startScreenSharing</a>	Start Screen Sharing
<a href="#">stopScreenSharing</a>	Stop Screen Sharing
<a href="#">startPushLocalVideo</a>	Start Pushing Local Video streams to Remote
<a href="#">stopPushLocalVideo</a>	Stop Pushing Local Video streams to Remote
<a href="#">startPushLocalAudio</a>	Start Pushing Local Audio Stream to Remote
<a href="#">stopPushLocalAudio</a>	Stop Pushing Local Audio Stream to Remote
<a href="#">setRemoteVideoView</a>	Set Remote Stream Rendering Area
<a href="#">startPlayRemoteVideo</a>	Start Playback Remote User Video streams
<a href="#">stopPlayRemoteVideo</a>	Stop Playback Remote User Video streams
<a href="#">muteRemoteAudioStream</a>	Stop Remote User Audio Stream

## roomEngine Member Management API

API	Description
<a href="#">openRemoteDeviceByAdmin</a>	Request Remote User to Open Media Device
<a href="#">applyToAdminToOpenLocalDevice</a>	Participant Apply to Host to Open Device
<a href="#">closeRemoteDeviceByAdmin</a>	Close Remote User Media Device
<a href="#">cancelRequest</a>	Cancel Sent Request
<a href="#">responseRemoteRequest</a>	Reply to Remote User Request
<a href="#">changeUserRole</a>	Change User Role



<a href="#">kickRemoteUserOutOfRoom</a>	Kick Out User from Room
<a href="#">disableDeviceForAllUserByAdmin</a>	Disable/Enable All Users' Media Device
<a href="#">disableSendingMessageForAllUser</a>	Disallow/Allow All Users to Send Message
<a href="#">disableSendingMessageByAdmin</a>	Disallow/Allow Specific User to Send Message

### roomEngine Mic Position Management API

API	Description
<a href="#">setMaxSeatCount</a>	Set Room Maximum Value
<a href="#">getSeatList</a>	Get Mic Position Information
<a href="#">takeSeat</a>	Get Mic Position
<a href="#">leaveSeat</a>	Release Mic Position
<a href="#">takeUserOnSeatByAdmin</a>	Invite Others to Go Live (Only Room Host and Administrator can invoke this method)
<a href="#">kickUserOffSeatByAdmin</a>	Kick Others Off the Mic (Only Room Host and Administrator can invoke this method)
<a href="#">lockSeatByAdmin</a>	Lock a Specific Mic Position Status (Only Room Host and Administrator can invoke this method)

### roomEngine Message Sending API

API	Description
<a href="#">sendTextMessage</a>	Send Text Message
<a href="#">sendCustomMessage</a>	Send Custom Message

### roomEngine Device Management API

API	Description
<a href="#">getCameraDevicesList</a>	Get Camera Device List
<a href="#">getMicDevicesList</a>	Get Mic Device List
<a href="#">getSpeakerDevicesList</a>	Get Speaker Device List

<a href="#">setCurrentCameraDevice</a>	Set to Use Camera
<a href="#">setCurrentMicDevice</a>	Set to Use Mic
<a href="#">setCurrentSpeakerDevice</a>	Set to Use Speaker
<a href="#">getCurrentCameraDevice</a>	Get the Currently Used Camera
<a href="#">getCurrentMicDevice</a>	Get the Currently Used Mic
<a href="#">getCurrentSpeakerDevice</a>	Get the Currently Used Speaker
<a href="#">startCameraDeviceTest</a>	Start Camera Test
<a href="#">stopCameraDeviceTest</a>	Stop Camera Test

### roomEngine Event Listening API

API	Description
<a href="#">on</a>	Listen to <a href="#">TUIRoomEvents</a>
<a href="#">off</a>	Cancel Listening to <a href="#">TUIRoomEvents</a>

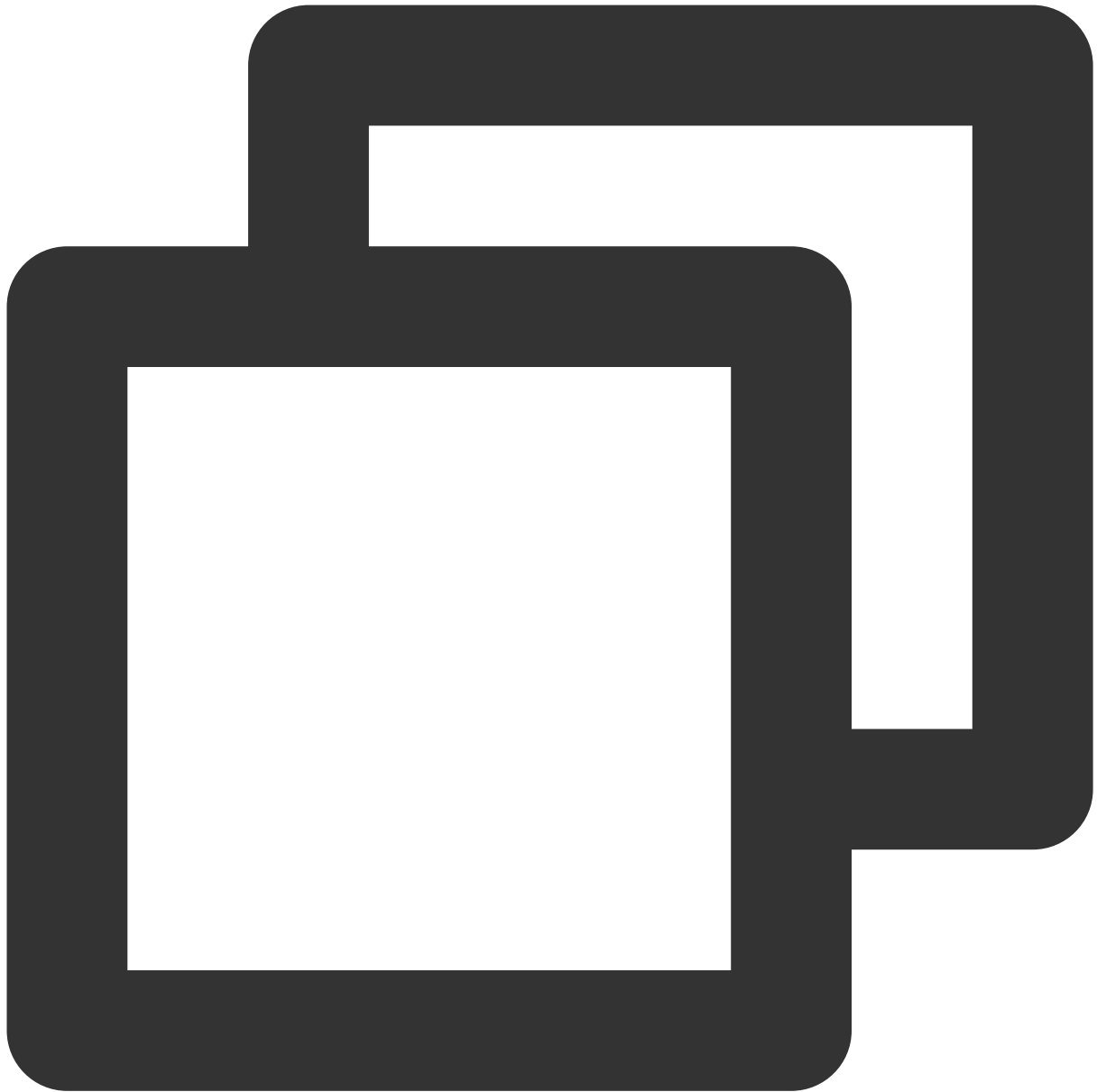
### roomEngine Other API

API	Description
<a href="#">getTRTCCloud</a>	Get trtcCloud Instance
<a href="#">getTIM</a>	Get tim Instance

## API Details

### once

Monitor TUIRoomEngine 'ready' Event



```
TUIRoomEngine.once('ready', () => {
  const roomEngine = new TUIRoomEngine();

  await TUIRoomEngine.login({
    sdkAppId: 0,    // Fill in the sdkAppId you applied for
    userId: '',     // Fill in the userId corresponding to your business
    userSig: '',    // Fill in the userSig calculated by the server or locally
  });

  await roomEngine.createRoom({
    roomId: '12345', // Enter your Room ID, note that the Room ID is required to
```

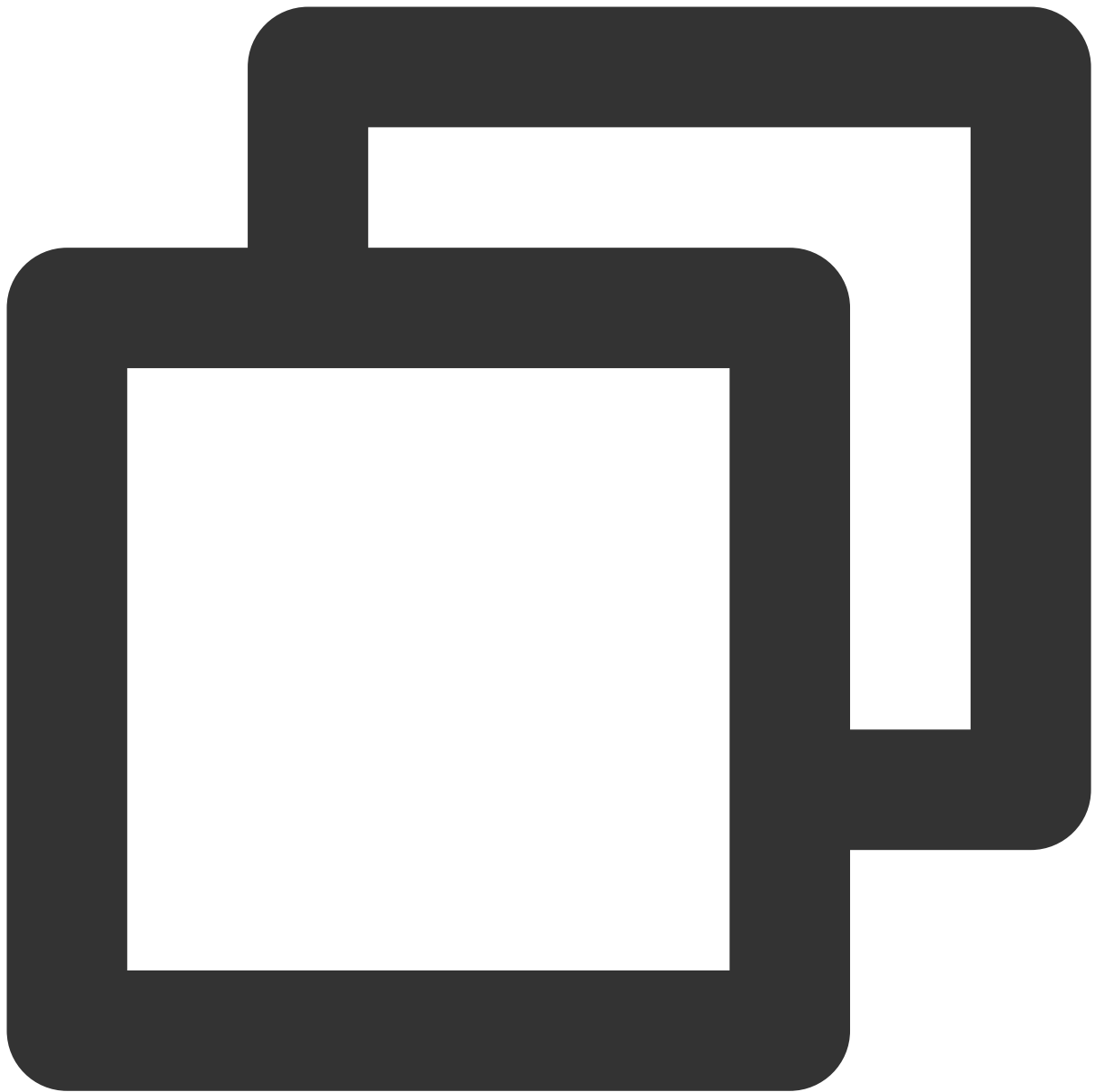
```
name: 'Test Room',      // Enter your room name, the default room name is roomId
roomType: TUIRoomType.kGroup, // Set the room type to TUIRoomType.kGroup type
});
});
```

## login

### Description:

In version v1.0.0, this interface is named TUIRoomEngine.init, please use TUIRoomEngine.login to log in TUIRoomEngine in v1.0.1 and above versions.

You must log in to TUIRoomEngine before you can call other methods of TUIRoomEngine and its instances.



```
// Login TUIRoomEngine
await TUIRoomEngine.login({
  sdkAppId: 0,    // Fill in the sdkAppId you applied for
  userId: '',     // Fill in the userId corresponding to your business
  userSig: '',    // Fill in the userSig calculated by the server or locally
});
```

Parameter:

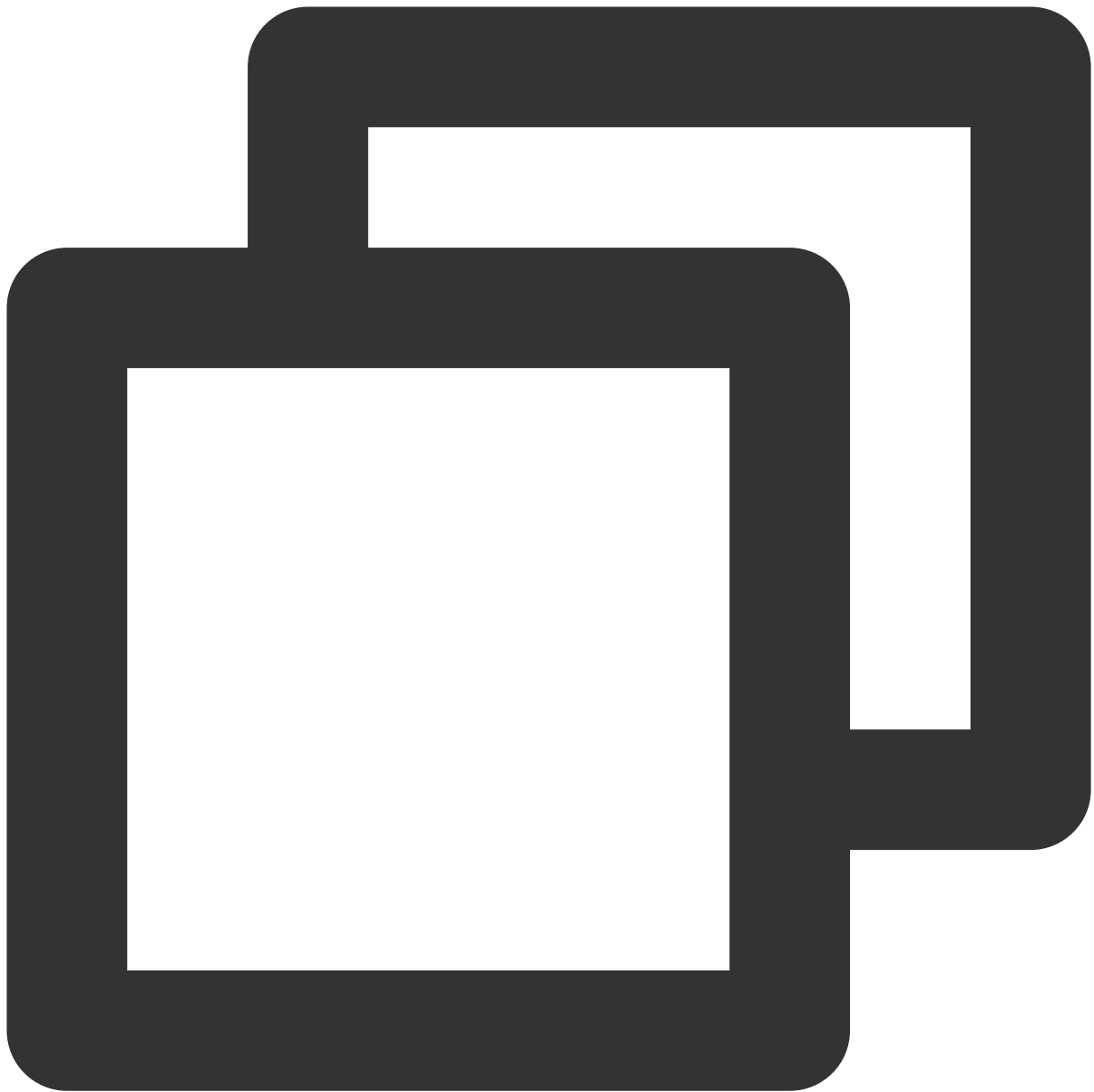
Parameter	Type	Description	Default Value	Meaning
-----------	------	-------------	---------------	---------

sdkAppId	number	Required	-	After clicking Application Management > Create Application in the Tencent Real-Time Communication Console, you can get the sdkAppId information in Application Info.
userId	string	Required	-	It is recommended to limit the user ID length to 32 bytes, and only allow uppercase and lowercase English letters (a-zA-Z), numbers (0-9), underscores, and hyphens.
userSig	string	Required	-	UserSig signature Please refer to <a href="#">UserSig related</a> for the method of calculating userSig.
tim	TIM	Not Required	-	If you want to use more capabilities of the Chat SDK while accessing roomEngine, you can pass the created tim instance into TUIRoomEngine. For the creation method of tim instance, please refer to <a href="#">TIM.create</a> .

Returns *Promise<void>*

## setSelfInfo

Set current user Basic information (Username, User avatar)



```
// Set current user Username and User avatar
await TUIRoomEngine.setSelfInfo({
  userName: '',      // Enter your New username
  avatarUrl: '',     // Enter your New avatar URL
});
```

Parameter:

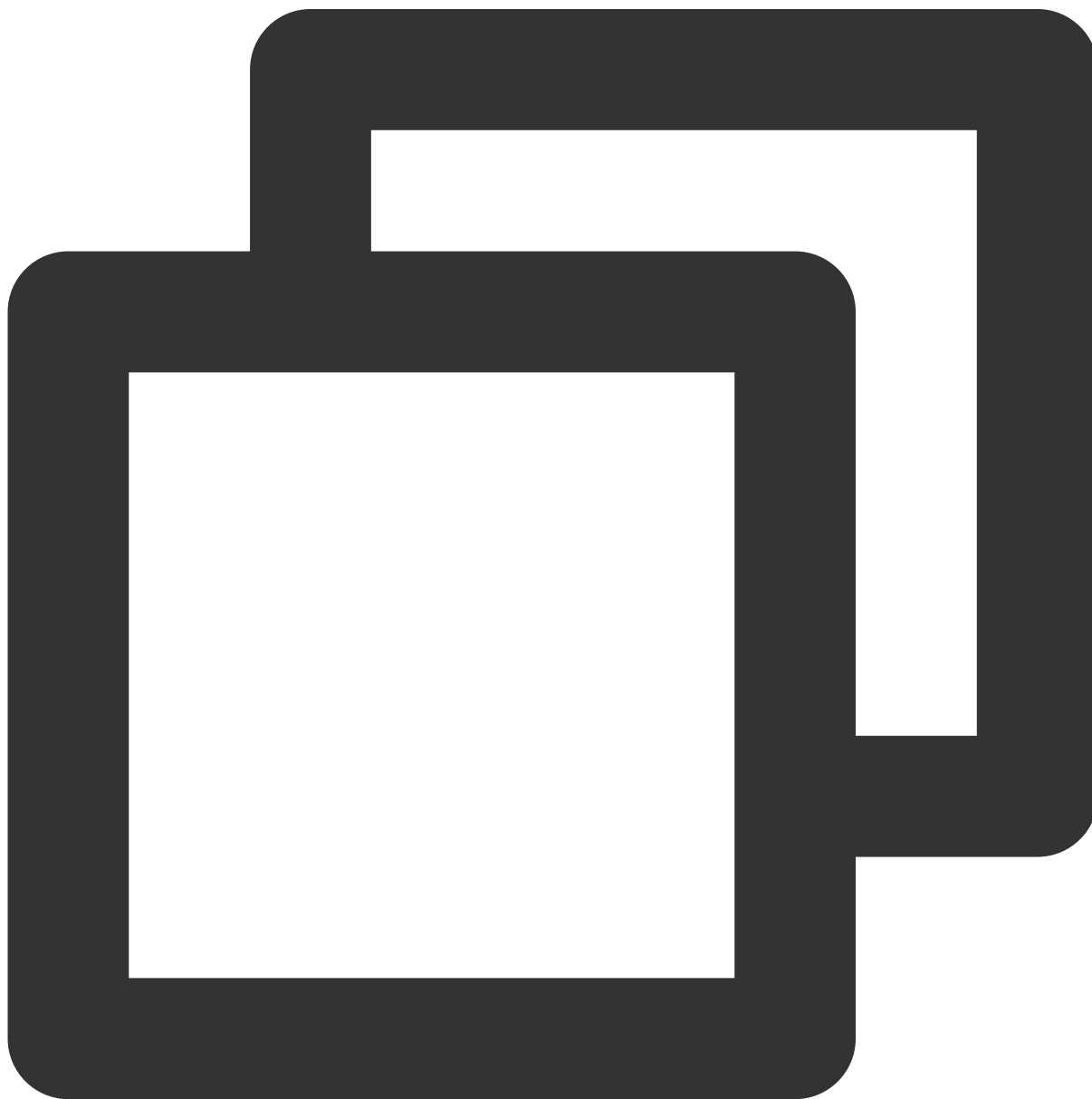
Parameter	Type	Description	Default Value	Meaning
userName	string	Required	-	User Name

avatarUrl	string	Required	-	User avatar
-----------	--------	----------	---	-------------

Returns *Promise<void>*

## getSelfInfo

Get current user Basic information (Username, User avatar)



```
// Get current user Username and User avatar
```



```
const loginUserInfo = await TUIRoomEngine.getSelfInfo();
```

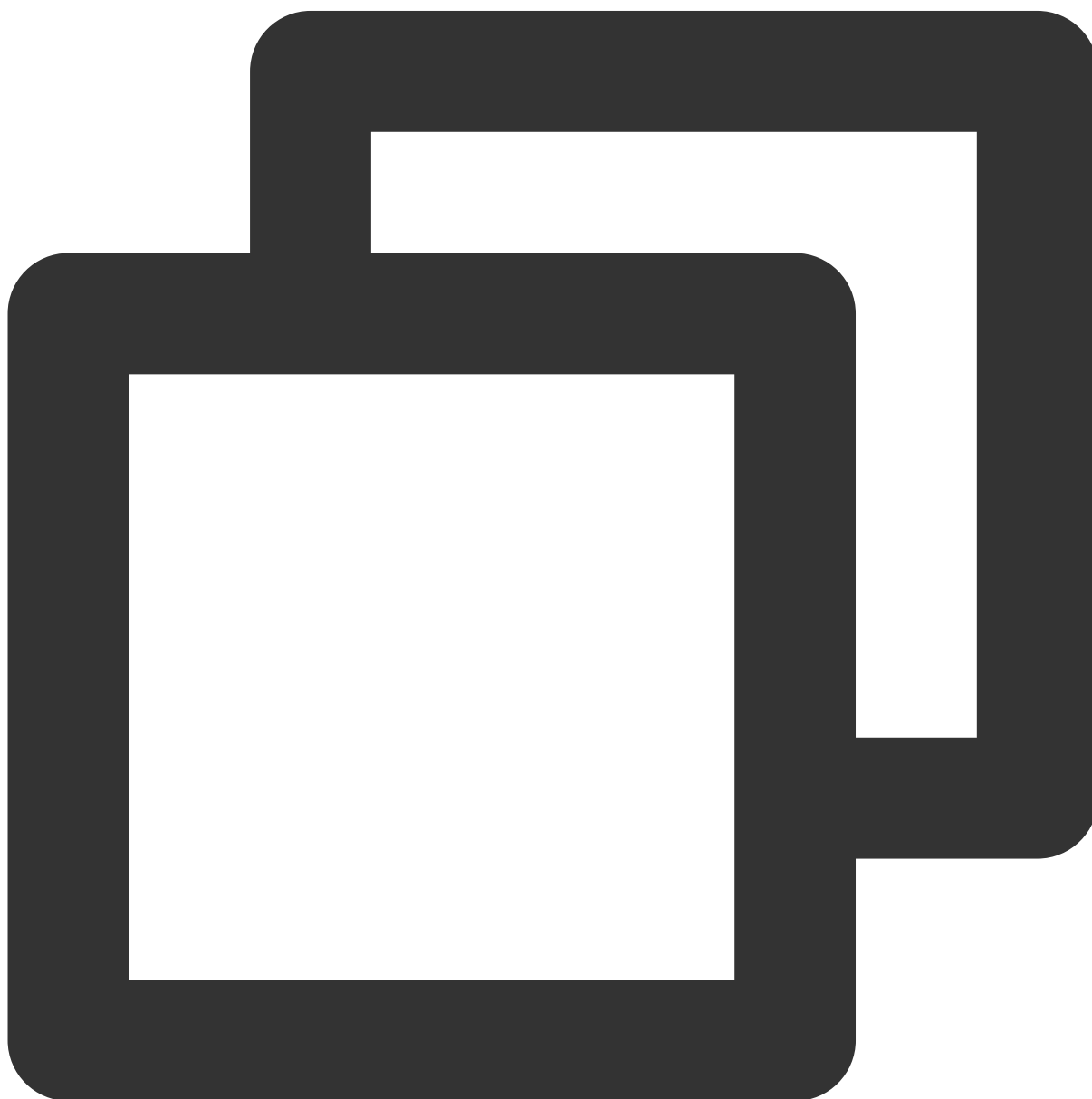
Returns *Promise*<[TUILoginUserInfo](#)> *loginUserInfo*

## logout

### Description:

Interface since v1.0.1 Version support.

Logout TUIRoomEngine

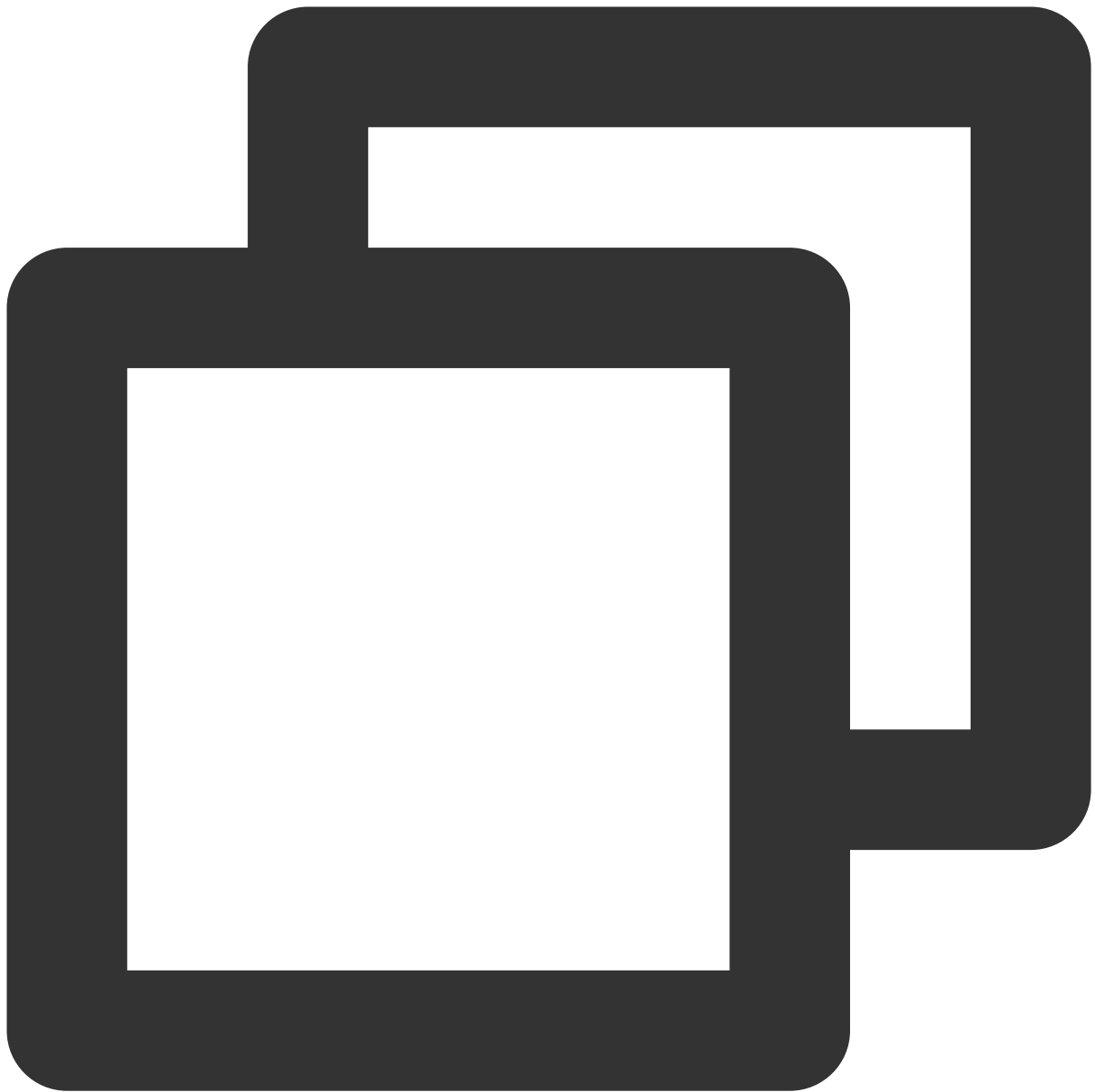


```
// Logout TUIRoomEngine  
await TUIRoomEngine.logout();
```

Returns *Promise<void>*

## createRoom

Host creates room, Call createRoom User is the room owner. When creating a room, you can set Room ID, Room name, Room type, Speaking mode, Whether to allow users to join and enable audio and video, Send message and other functions.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({
  roomId: '12345',    // Enter your Room ID, note that the Room ID must be a string
  roomName: 'Test Room',    // Enter your Room Name, the default Room Name is roomName
  roomType: TUIRoomType.kConference, // Set the Room Type to TUIRoomType.kConference
  speechMode: TUISpeechMode.kFreeToSpeak, // Set the speech mode to free speech mode
  isMicrophoneDisableForAllUser: false, // Allow users to turn on their mic when join
  isCameraDisableForAllUser: false, // Allow users to turn on their Camera when join
  isMessageDisableForAllUser: false, // Allow users to send messages when join
});
```

Parameter:

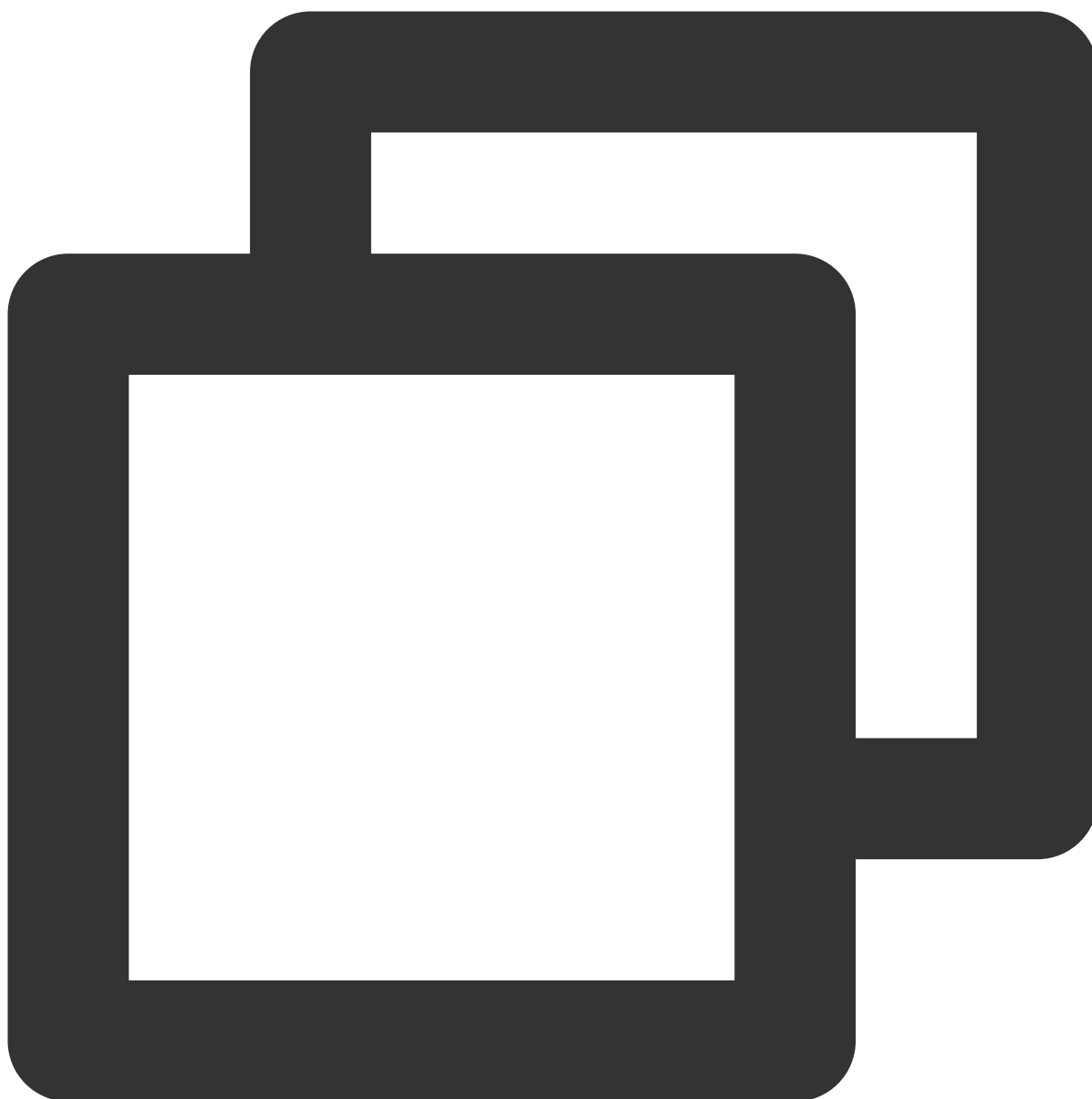
Parameter	Type	Description	Default Value	Mea
roomId	string	Required	-	Roo byte char Engl Nurr Spa { }   ·
roomName	string	Optional	roomId	Roo canr
roomType	<a href="#">TUIRoomType</a>	Optional	TUIRoomType.kConference	Roo Offic rem room TUI E-cc room TUI
speechMode	<a href="#">TUISpeechMode</a>	Optional	TUISpeechMode.kFreeToSpeak	Spe For (edu Set : TUI: can defa Set : TUI: do n defa to a or m and mod Set : TUI: user get p and For broa

				Set : TUI: for h Set : TUI: host TUI: TUI: mod
isMicrophoneDisableForAllUser	boolean	Optional	false	Enal mute
isCameraDisableForAllUser	boolean	Optional	false	Enal not e
isMessageDisableForAllUser	boolean	Optional	false	Allov proh
maxSeatCount	number	Optional	-	Max For ` (edu is nc For ` broa max
enableCDNStreaming	boolean	Optional	false	Enal
cdnStreamDomain	string	Optional	"	Pusl

Returns *Promise<void>*

## enterRoom

Entered room interface



```
const roomEngine = new TUIRoomEngine();
const roomInfo = await roomEngine.enterRoom({
  roomId: '12345',
});
```

Parameter:

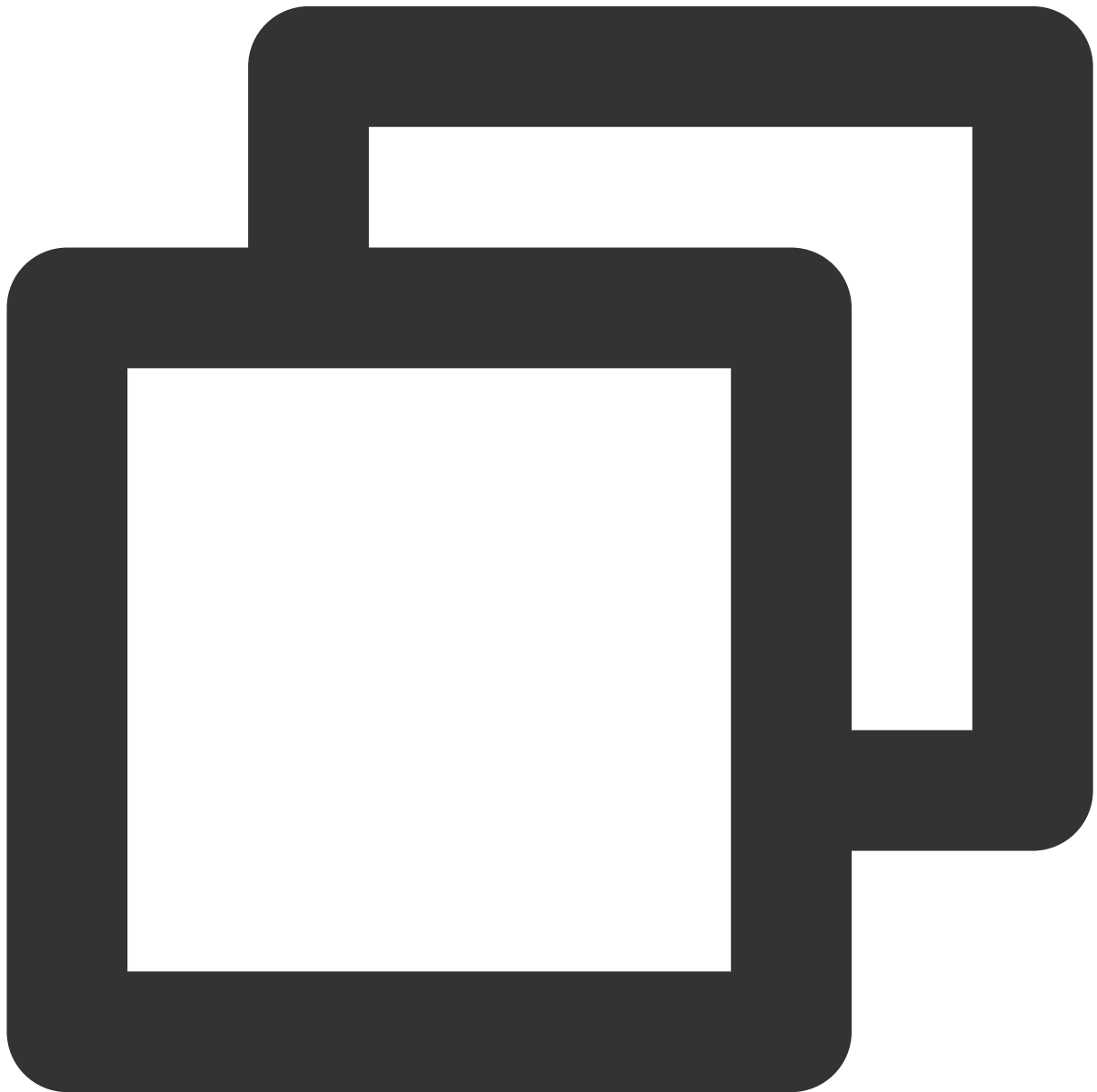
Parameter	Type	Description	Default Value	Meaning
roomId	string	Required	-	Room ID

Returns Promise<[TUIRoomInfo](#)> roomInfo

This interface returns the current Room data

## destroyRoom

Close the room interface, the room must be closed by the room owner, and the room cannot be entered after it is closed.



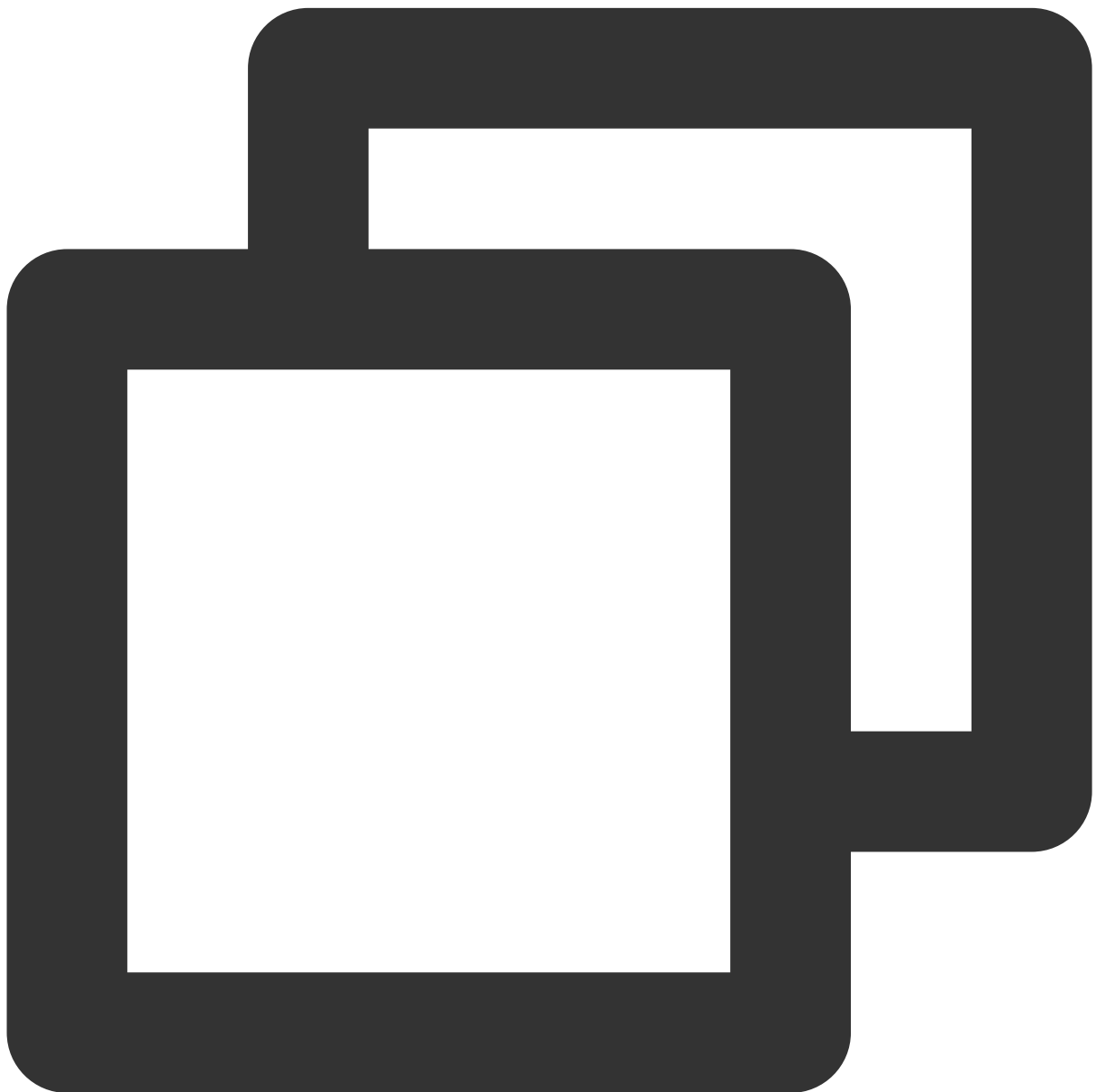
```
const roomEngine = new TUIRoomEngine();
```

```
await roomEngine.destroyRoom();
```

Returns *Promise<void>*

## exitRoom

Leave the room interface, users can leave the room through exitRoom after executing enterRoom.



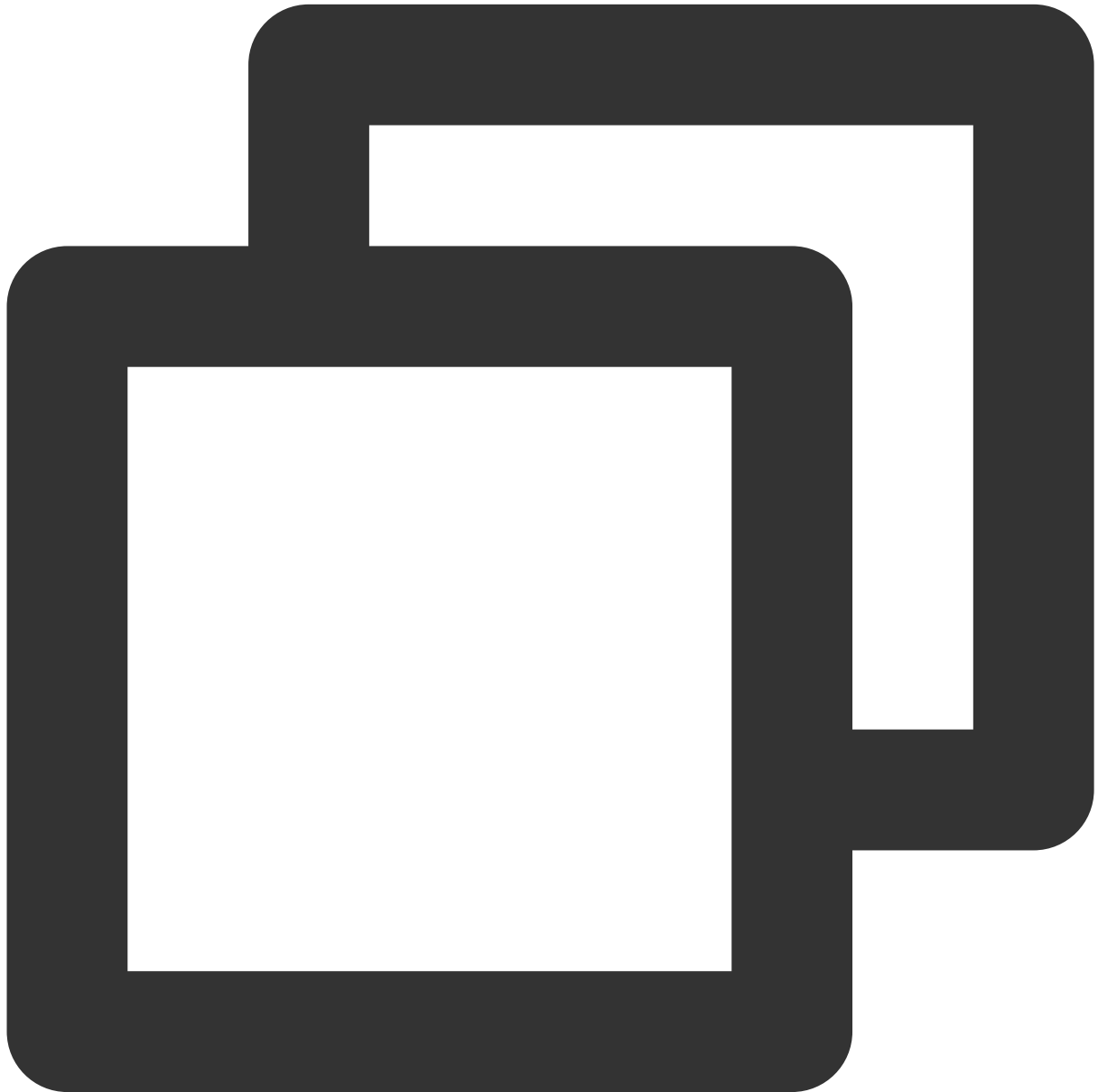
```
const roomEngine = new TUIRoomEngine();  
await roomEngine.exitRoom();
```



Returns *Promise<void>*

## fetchRoomInfo

Get Room information

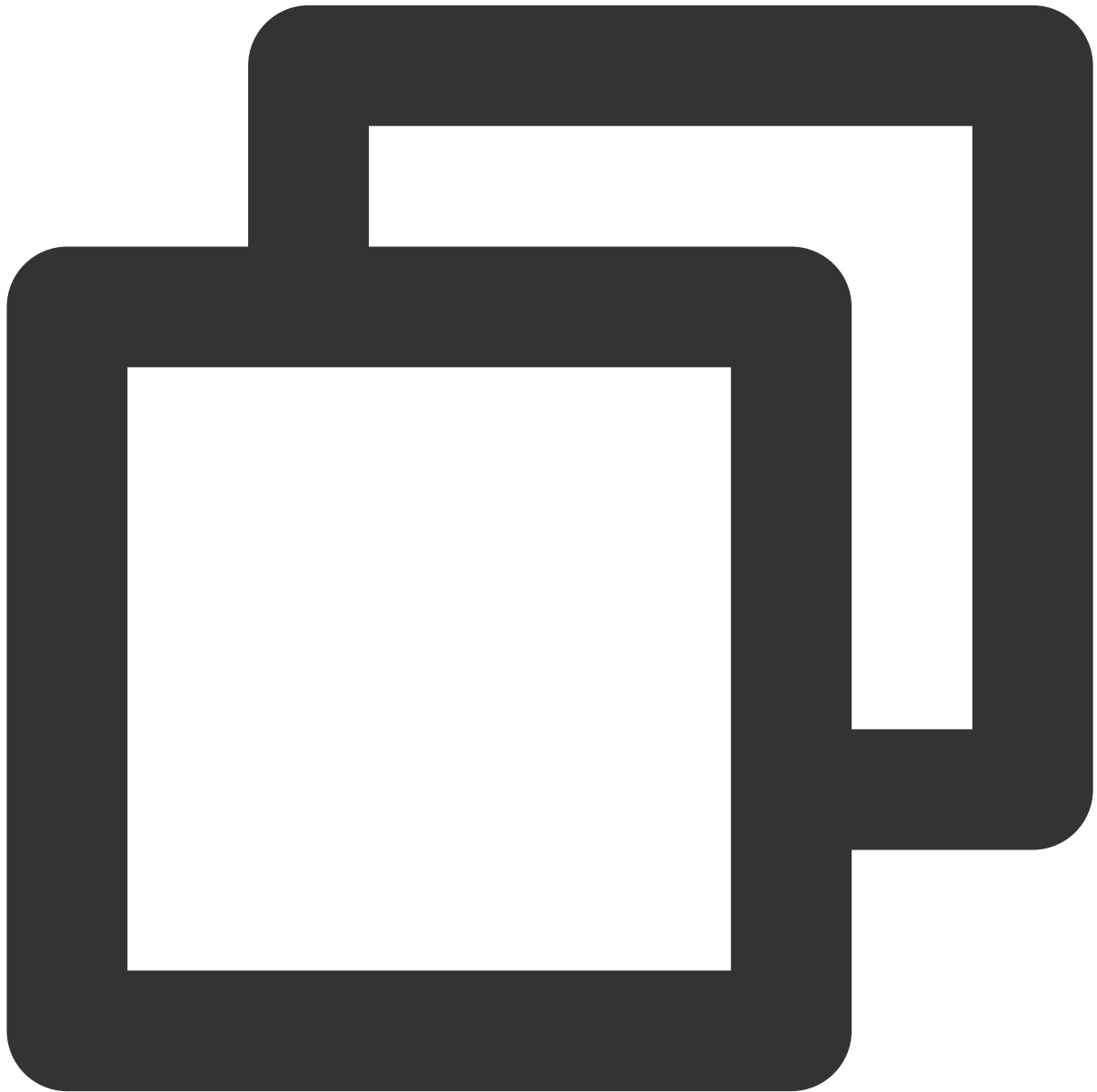


```
const roomEngine = new TUIRoomEngine();  
const roomInfo = roomEngine.fetchRoomInfo();
```

Returns : *Promise*<[TUIRoomInfo](#)> *roomInfo*

## updateRoomNameByAdmin

Update the current room's name (only group owner or admin can invoke)



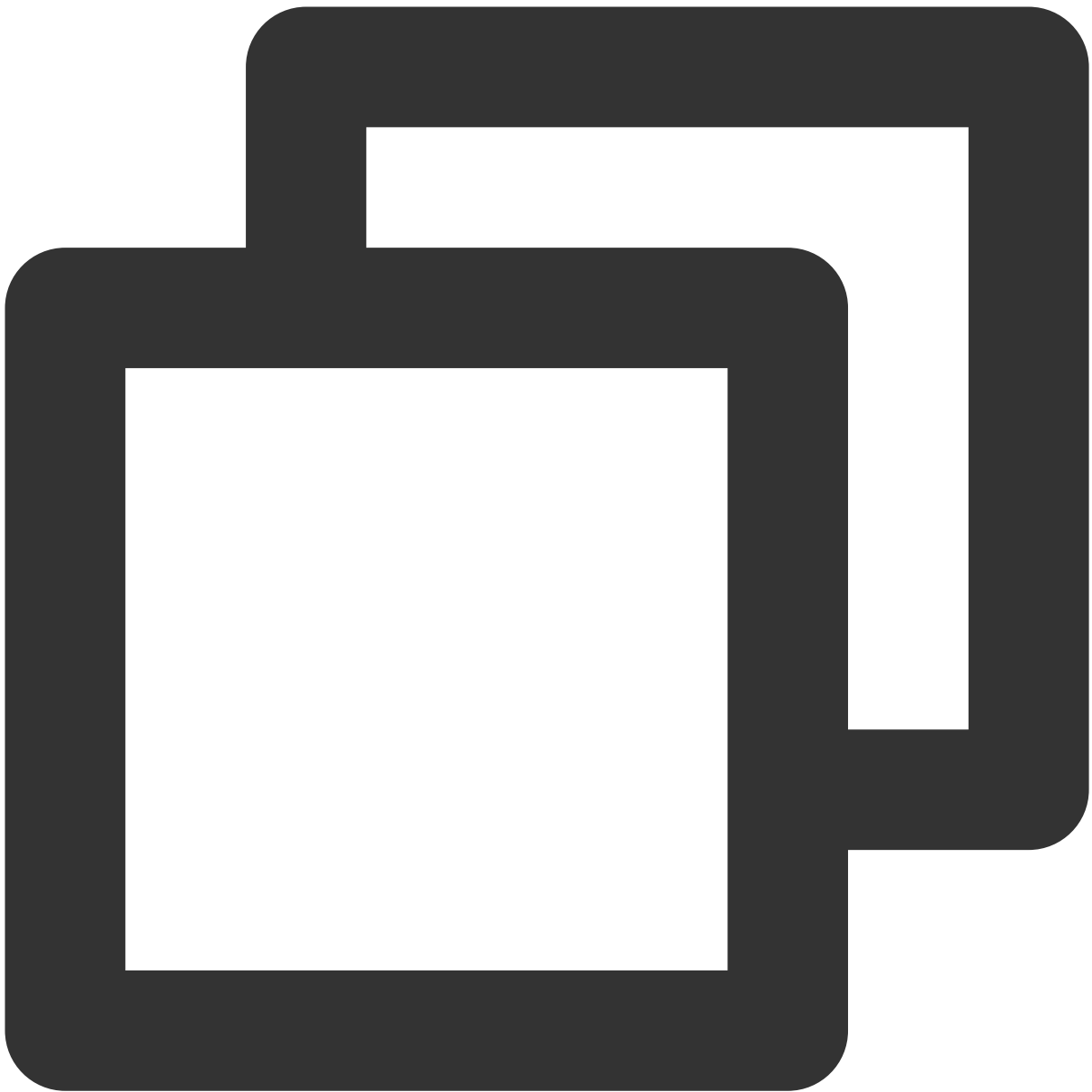
```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomNameByAdmin({ roomName: 'NewName' });
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
roomName	string	Required	-	Update the room's name, with the requirement that roomName is not an empty string

## updateRoomSpeechModeByAdmin

Update the room's speaking mode (only group owner or admin can invoke)



```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSpeechModeByAdmin({
  speechMode: TUISpeechMode.kSpeakAfterTakingSeat // Update to Go Live Speaking mode
});
```

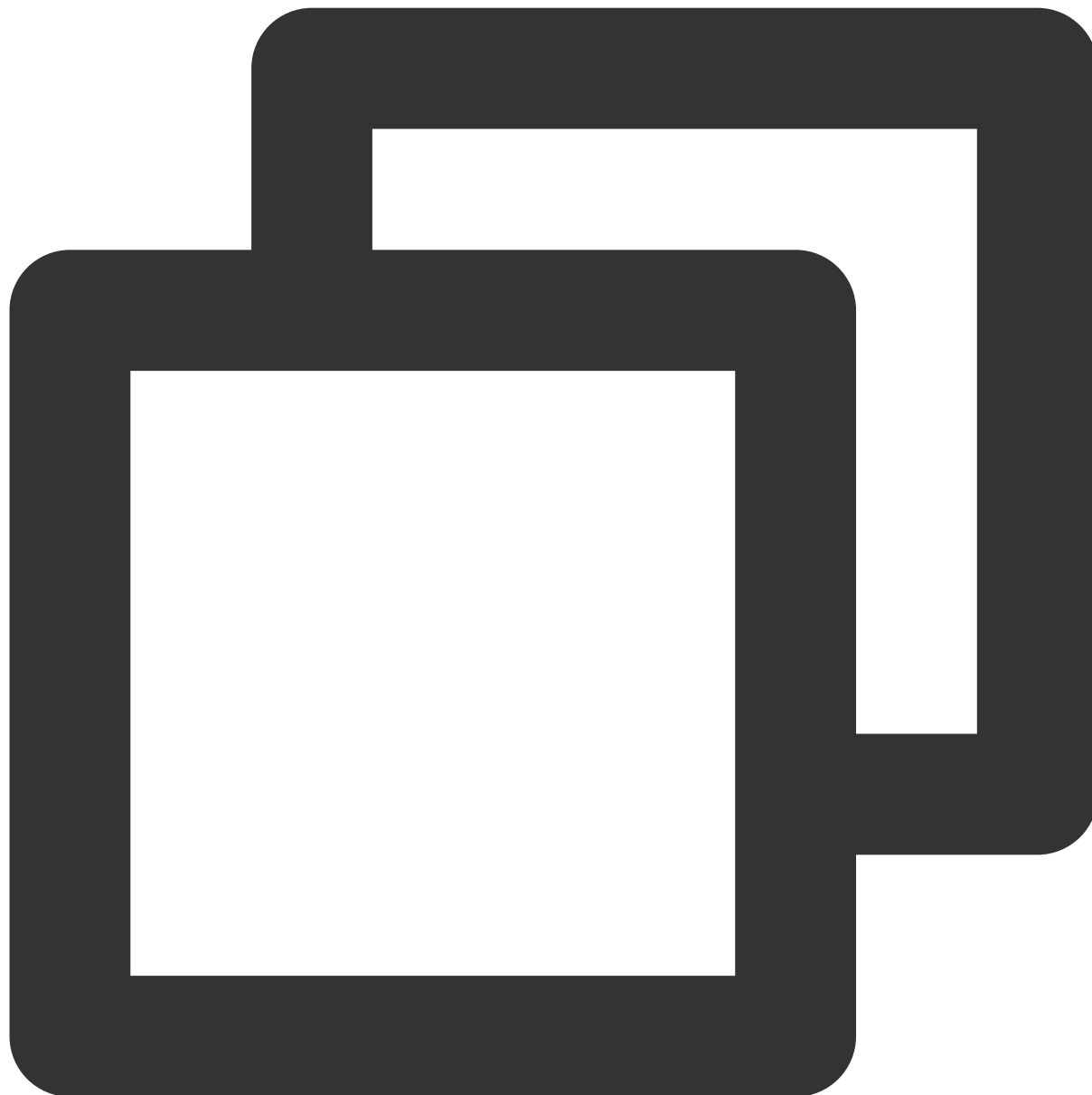
Parameter:

Parameter	Type	Description	Default Value	Meaning

speechMode	TUISpeechMode	Required	-	Update the room's speaking mode
------------	---------------	----------	---	---------------------------------

## getUserList

Get the current room's user list, note that the maximum number of user lists fetched by this interface is 100



```
const roomEngine = new TUIRoomEngine();  
const userList = [];  
let result;
```

```
do {  
  result = await globalProperties.roomEngine.getUserList();  
  userList.push(...result.userInfoList);  
} while (result.nextSequence !== 0)
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
nextSequence	number	Optional	0	Offset, default is to start fetching users from 0

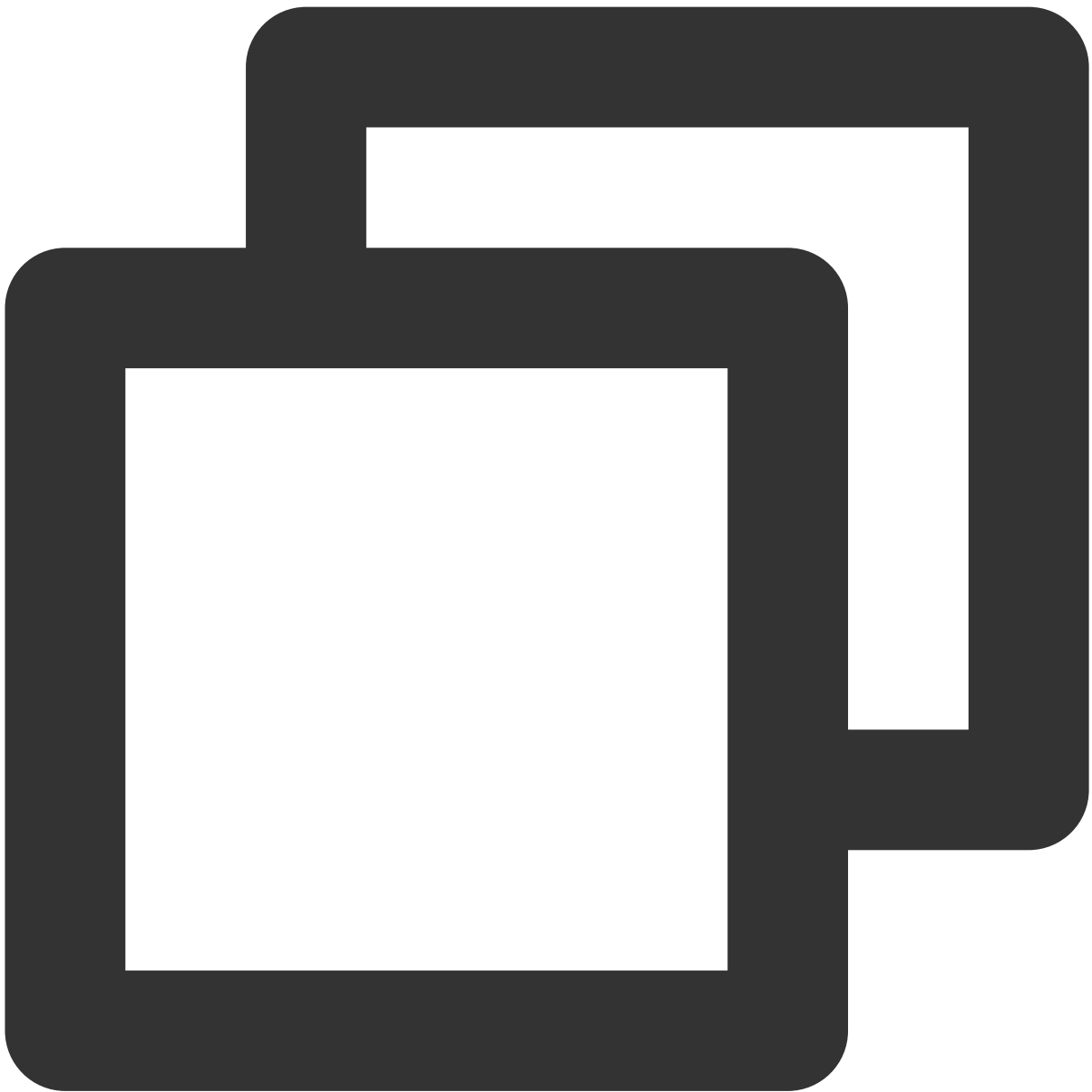
Returns : *Promise<Array> result*

result.nextSequence is the offset for fetching group users next time, if result.nextSequence is 0, it means that all userList have been fetched

result.userInfoList is the userList fetched this time

## getUserInfo

Get the detailed information of the user



```
const roomEngine = new TUIRoomEngine();
const userList = [];
const userInfo = await roomEngine.getUserInfo({
  userId: 'user_12345',
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
-----------	------	-------------	---------------	---------

userId	string	Required	-	Get the detailed information of the user according to userId
--------	--------	----------	---	--

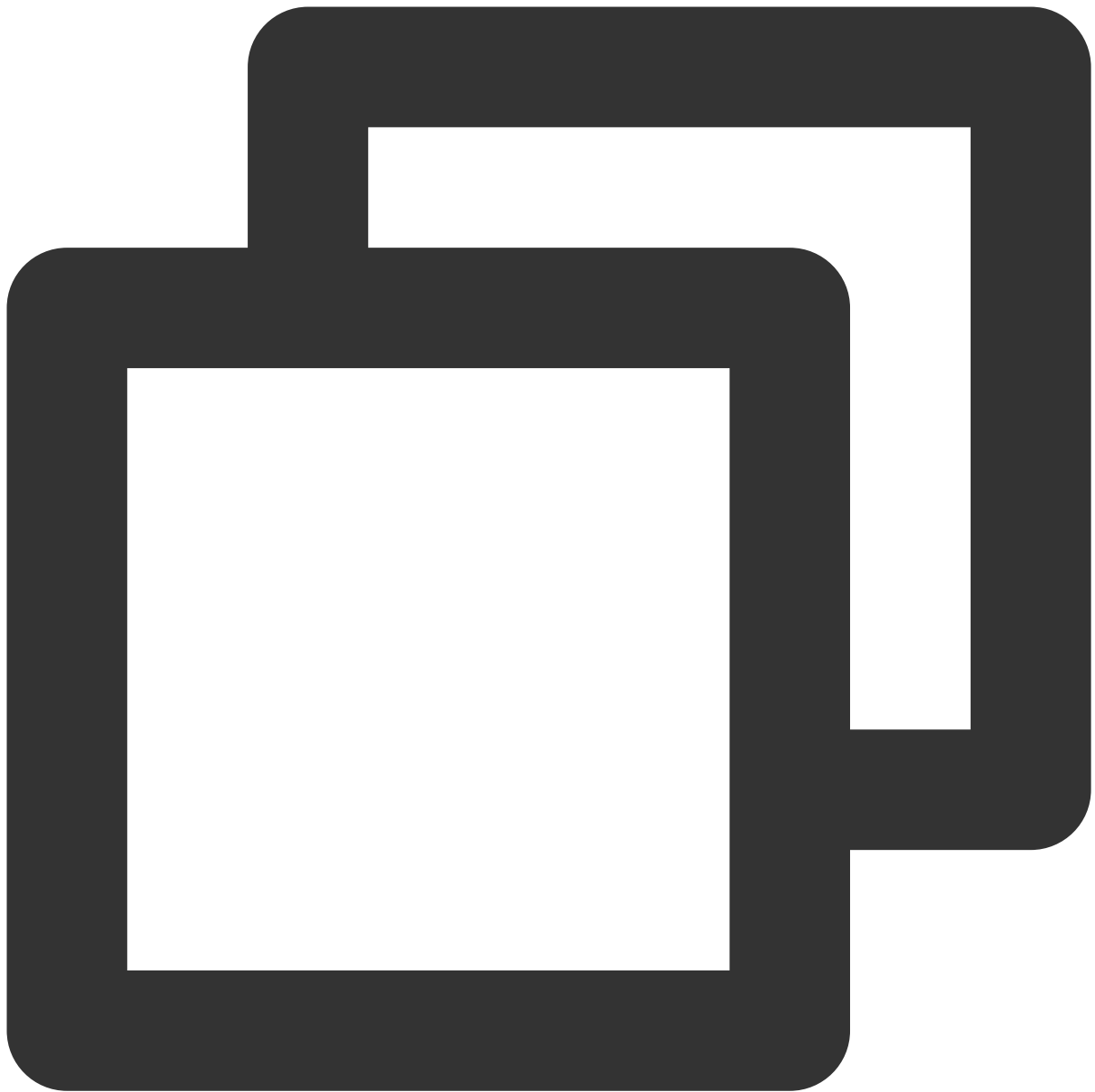
Returns : *Promise<Array<[TUIUserInfo](#)>> userInfoList*

This interface returns the user information of the specified user

## setLocalVideoView

Set the rendering position of the local stream





```
const roomEngine = new TUIRoomEngine();

// Set the playback area of the local camera stream to the div element with id 'pre
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});

// Set the playback area of the local screen sharing stream to the div element with
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kScreenStream,
```

```
view: 'preview-screen',  
});
```

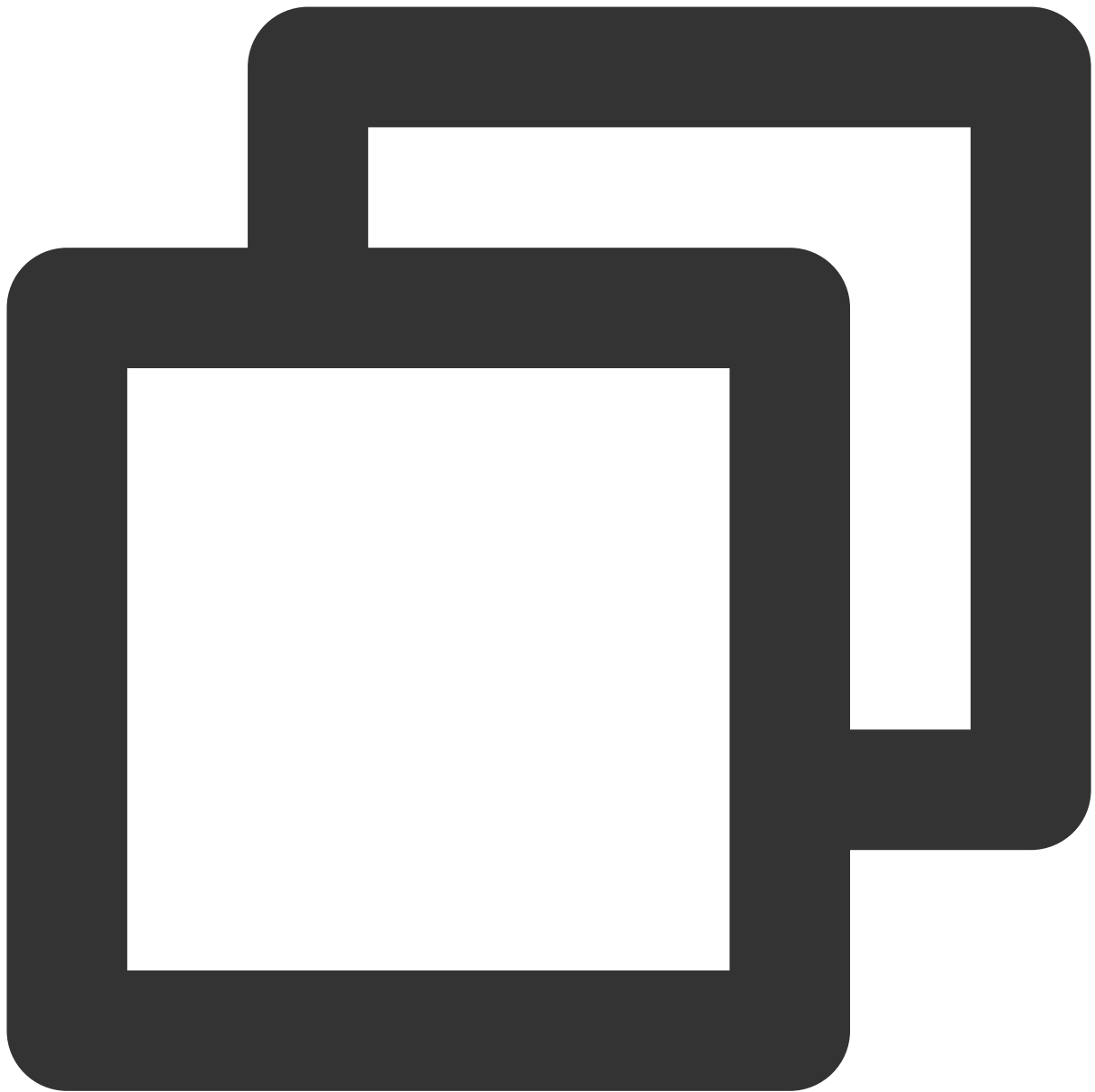
Parameter:

Parameter	Type	Description	Default Value	Meaning
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	Local stream type
view	string	Required	-	The id of the div element corresponding to the streamType

Returns : *Promise<void>*

## openLocalCamera

Open the local camera and start capturing video streams

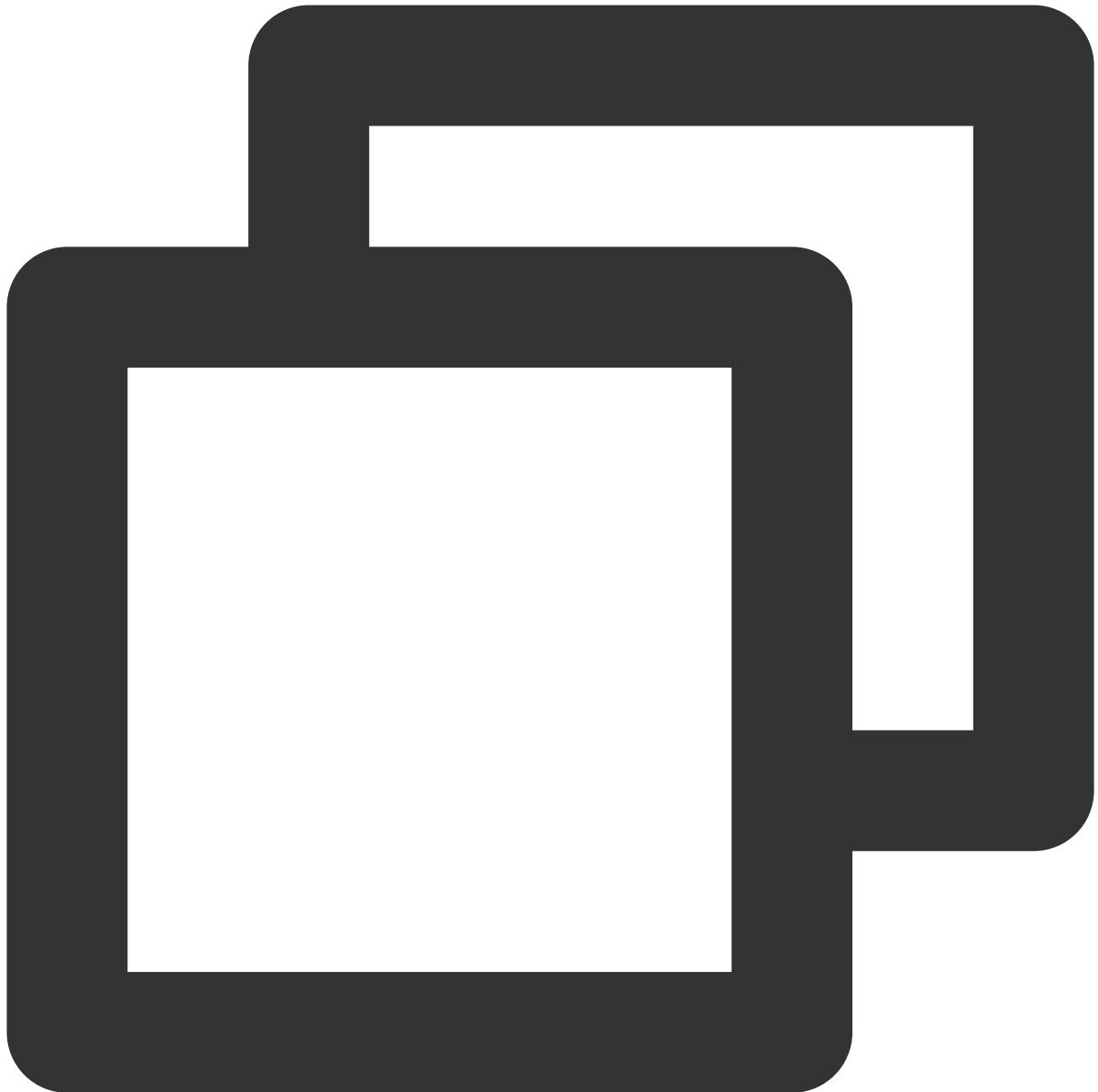


```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});
await roomEngine.openLocalCamera();
```

Returns : *Promise<void>*

## closeLocalCamera

Close the local camera

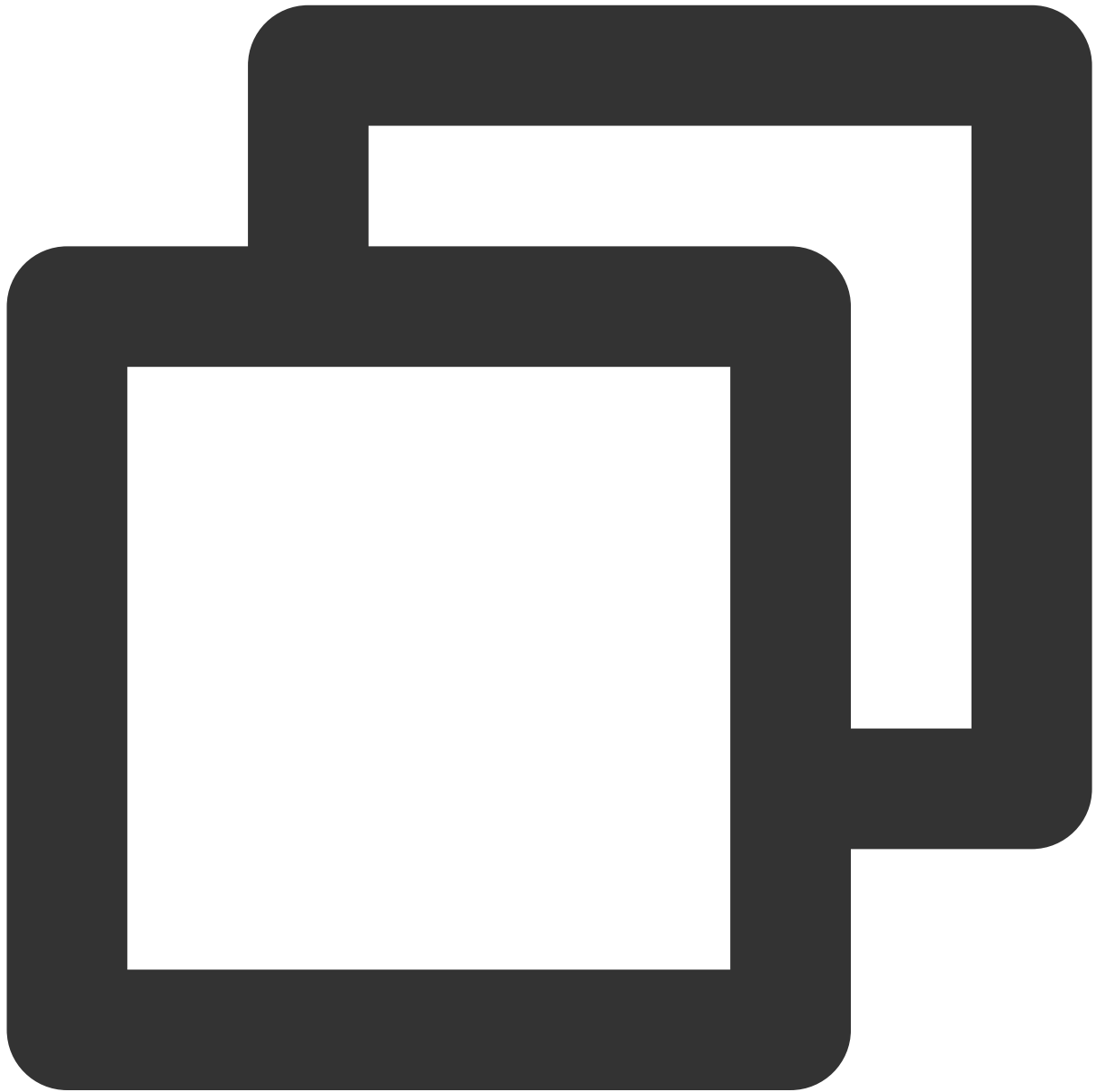


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.closeLocalCamera();
```

Returns : *Promise<void>*

## openLocalMicrophone

Open the local mic and start capturing audio streams

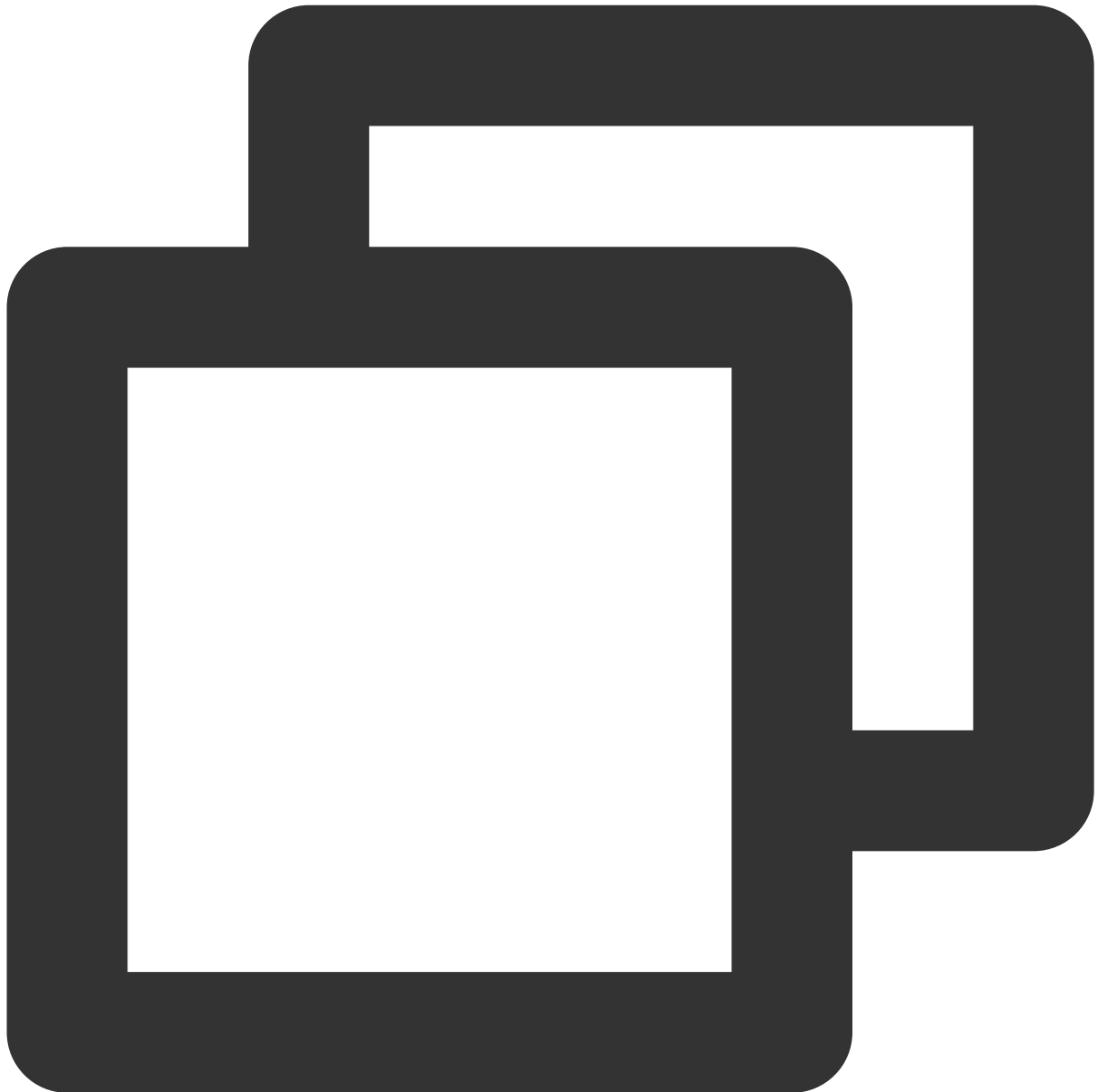


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.openLocalMicrophone();
```

Returns : *Promise<void>*

## closeLocalMicrophone

Close the local mic

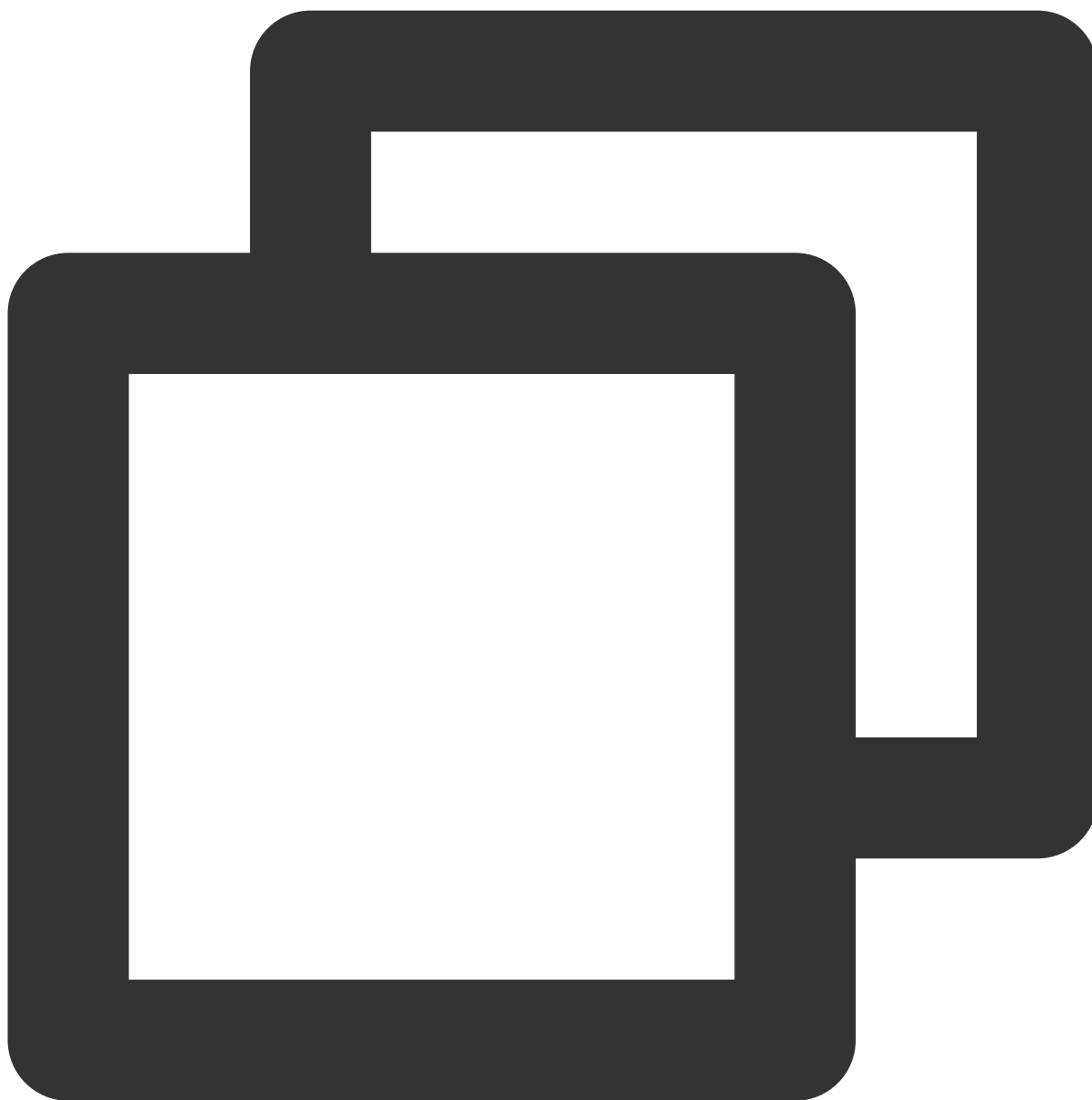


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.closeLocalMicrophone();
```

Returns : *Promise<void>*

## updateVideoQuality

Set the codec parameters of the local video stream, default is `TUIVideoProfile.kVideoQuality_720P`



```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQuality({
  quality: TUIVideoQuality.kVideoQuality_540p,
});
```

Parameter:

--	--	--	--	--

Parameter	Type	Description	Default Value	Meaning
quality	<a href="#">TUIVideoQuality</a>	Required	-	Clear TUIVideoProfile.kVideoQuality_360P SD TUIVideoProfile.kVideoQuality_540P HD TUIVideoProfile.kVideoQuality_720P Full HD TUIVideoProfile.kVideoQuality_1080P

Returns : *Promise<void>*

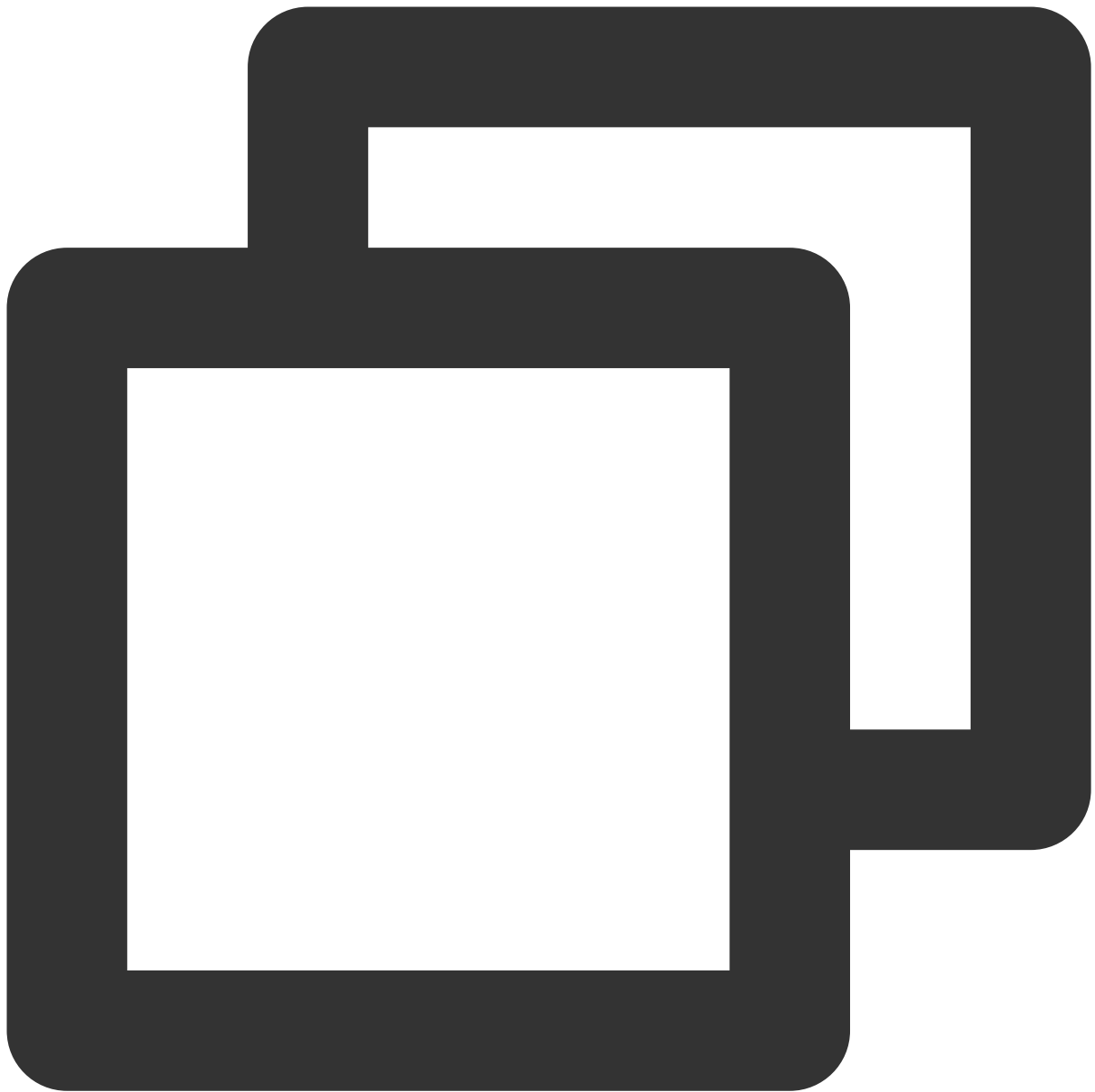
## updateAudioQuality

Set Local Audio Parameters

### Note:

This method needs to be set before openLocalMicrophone, otherwise it will not take effect.





```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalAudioProfile({
  audioProfile: TUIAudioProfile.kAudioProfileSpeech,
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
audioProfile	<a href="#">TUIAudioProfile</a>	Required	-	TUIAudioProfile.kAudioProfileSpeech: Speech

				Mode; Sample rate: 16k TUIAudioProfile.kAudioProfileDefault: Standard Mode (or Default Mode); Sample rate: 48k TUIAudioProfile.kAudioProfileMusic: Music Mode; Sample rate: 48k
--	--	--	--	--

Returns : *Promise*<void>

## startScreenSharing

Start Screen Sharing



```
const roomEngine = new TUIRoomEngine();
// Set the playback area of the local screen sharing stream to the div element with the id 'preview-screen'
await roomEngine.setLocalRenderView({
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'preview-screen',
});
// example 1: Start Screen Sharing
await roomEngine.startScreenSharing();
// example 2: Start Screen Sharing (Capturing System Audio)
await roomEngine.startScreenSharing({ screenAudio: true });
```

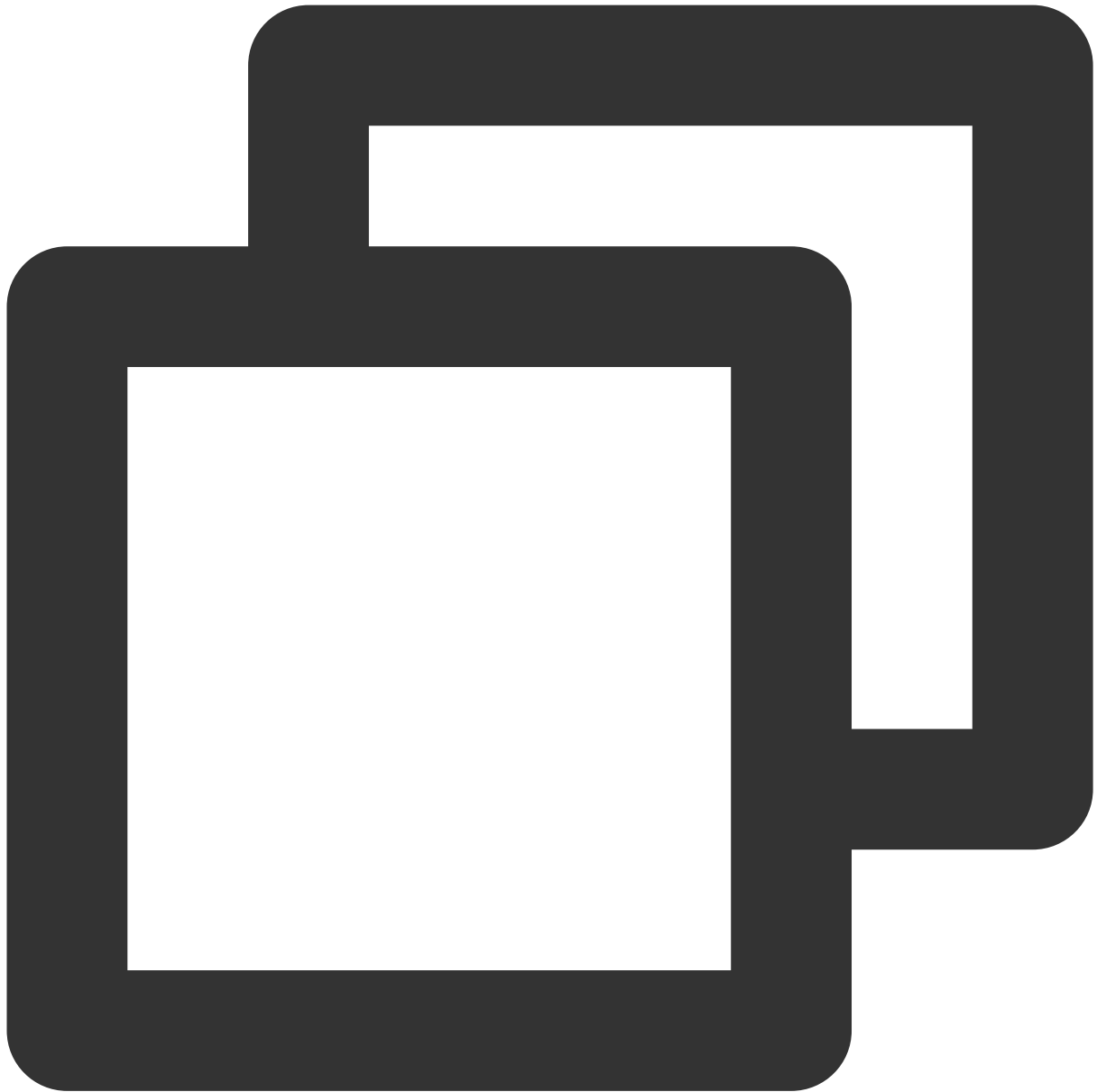
Parameter:

Parameter	Type	Description	Default Value	Meaning
screenAudio	boolean	Optional	false	Whether web screen sharing can optionally share system sound, screenAudio default value is false

Returns : *Promise<void>*

## stopScreenSharing

Stop Screen Sharing

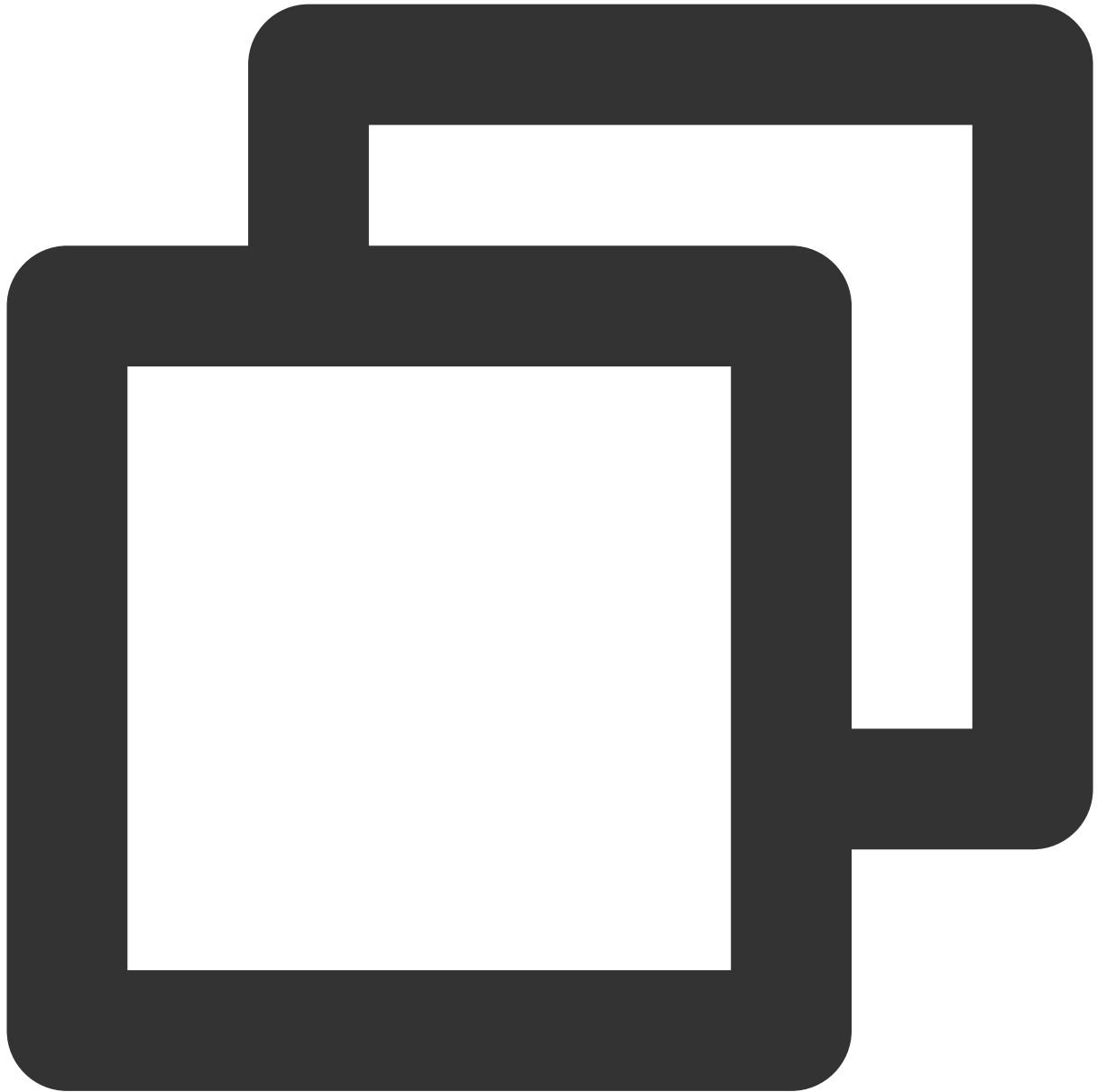


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopScreenSharing();
```

Returns : *Promise<void>*

## startPushLocalVideo

After entering the room, the local video stream will be pushed to the remote by default. This interface is used to re-push the local video stream to the remote after stopping the push.

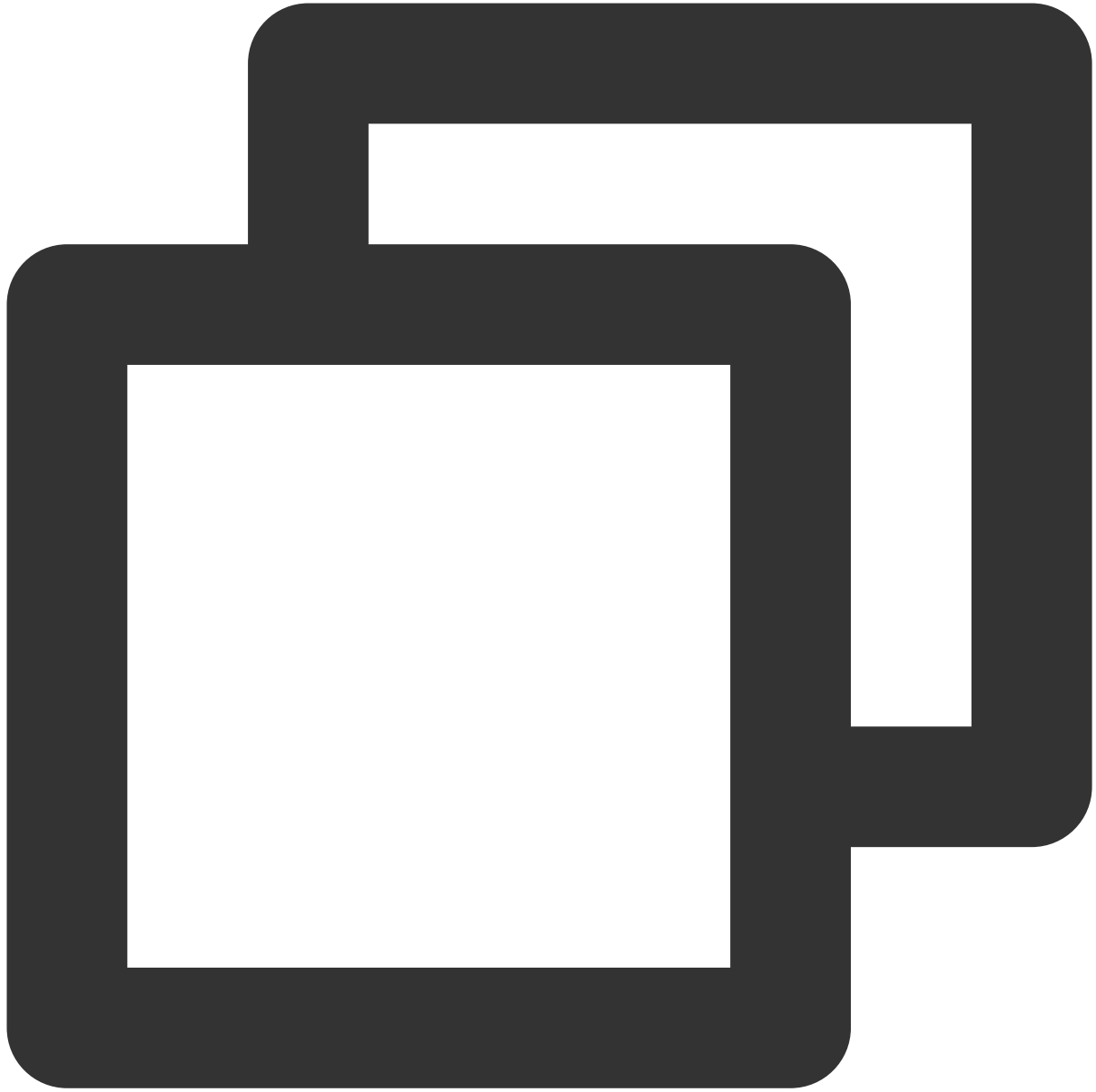


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startPushLocalVideo();
```

Returns : *Promise<void>*

## stopPushLocalVideo

## Stop Pushing Local Video Stream to Remote

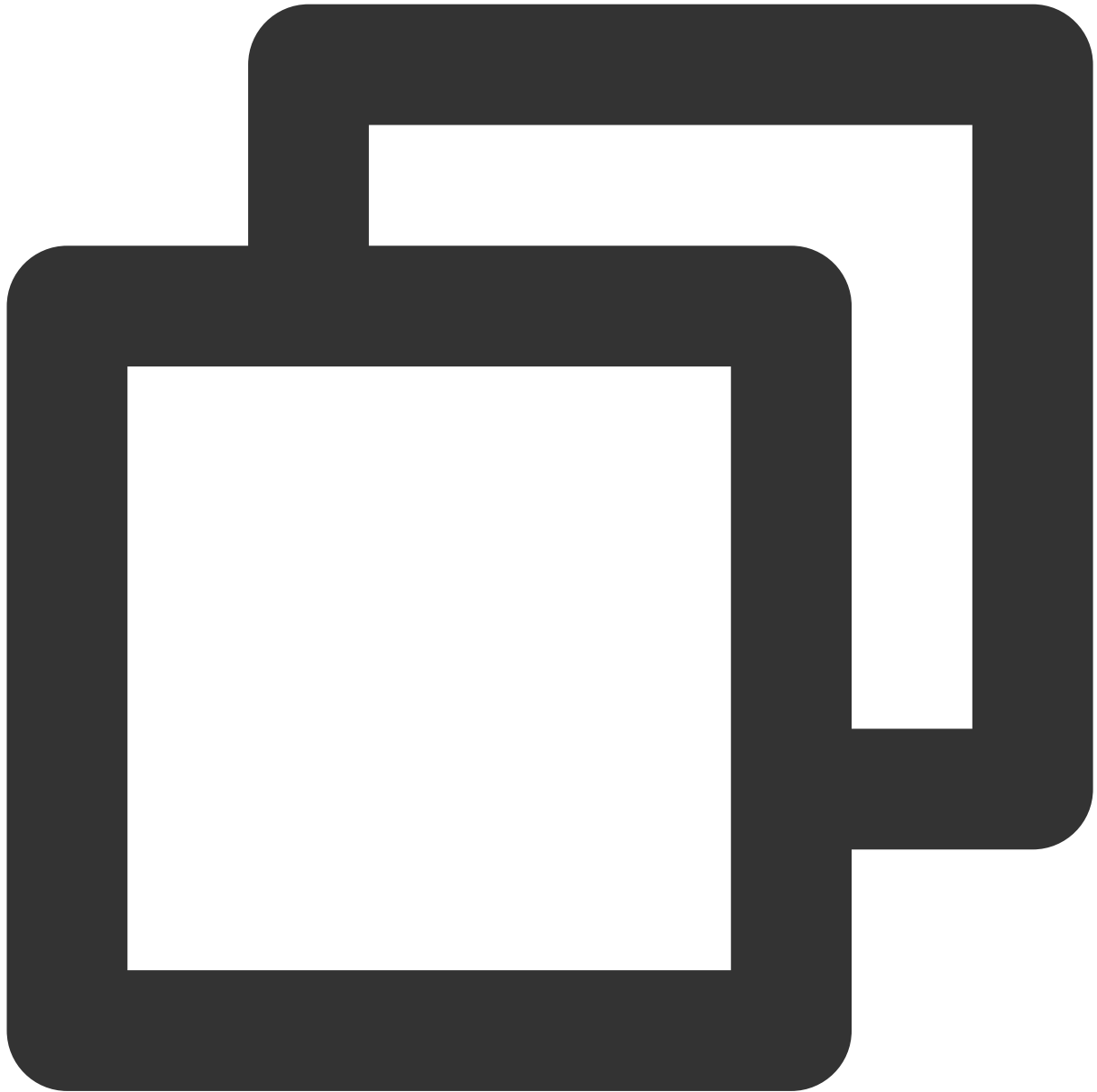


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopPushLocalVideo();
```

Returns : *Promise<void>*

## startPushLocalAudio

After entering the room, the local audio stream will be pushed to the remote by default. This interface is used to re-push the local audio stream to the remote after stopping the push.



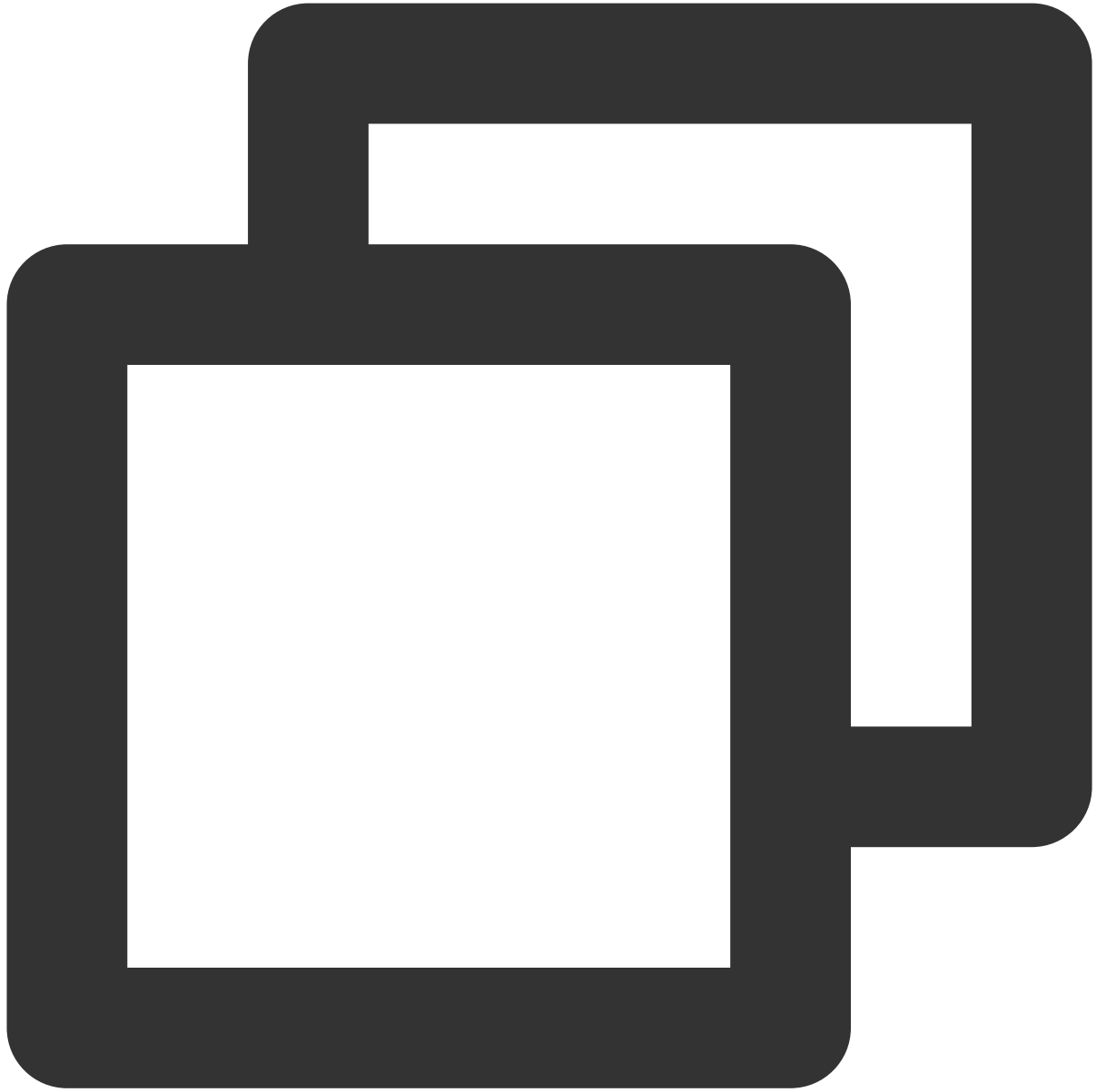
```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startPushLocalAudio();
```

Returns : *Promise<void>*

## stopPushLocalAudio



## Stop Pushing Local Audio Stream to Remote

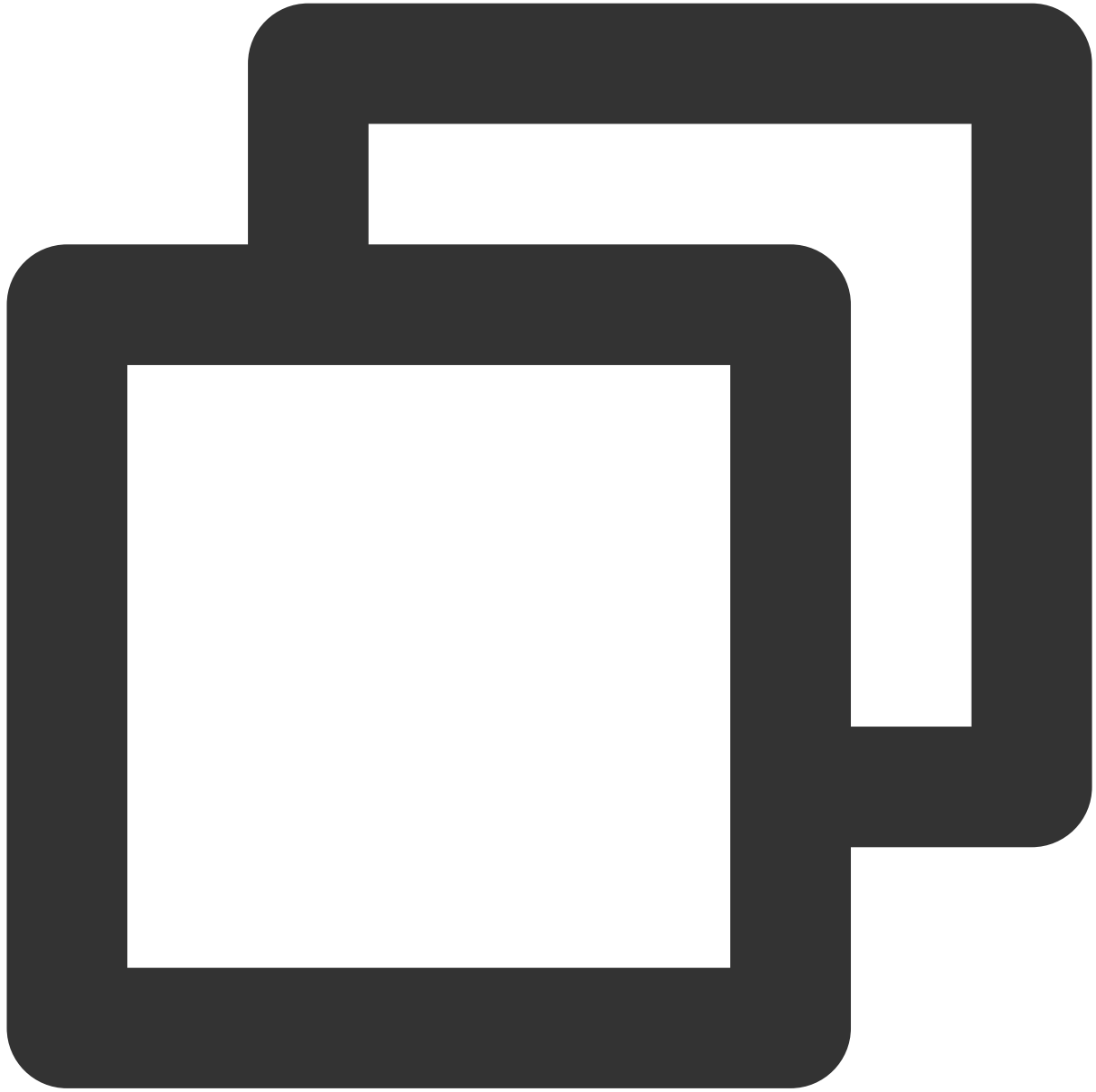


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopPushLocalAudio();
```

Returns : *Promise<void>*

## setRemoteVideoView

## Set Remote Stream Rendering Area



```
const roomEngine = new TUIRoomEngine();

// Set the remote user's video stream to play in the area with id 'remote_preview_c
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'remote_preview_camera',
});
// Set the remote user's screen sharing stream to play in the area with id 'remote_
```

```
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'remote_preview_screen',
});
```

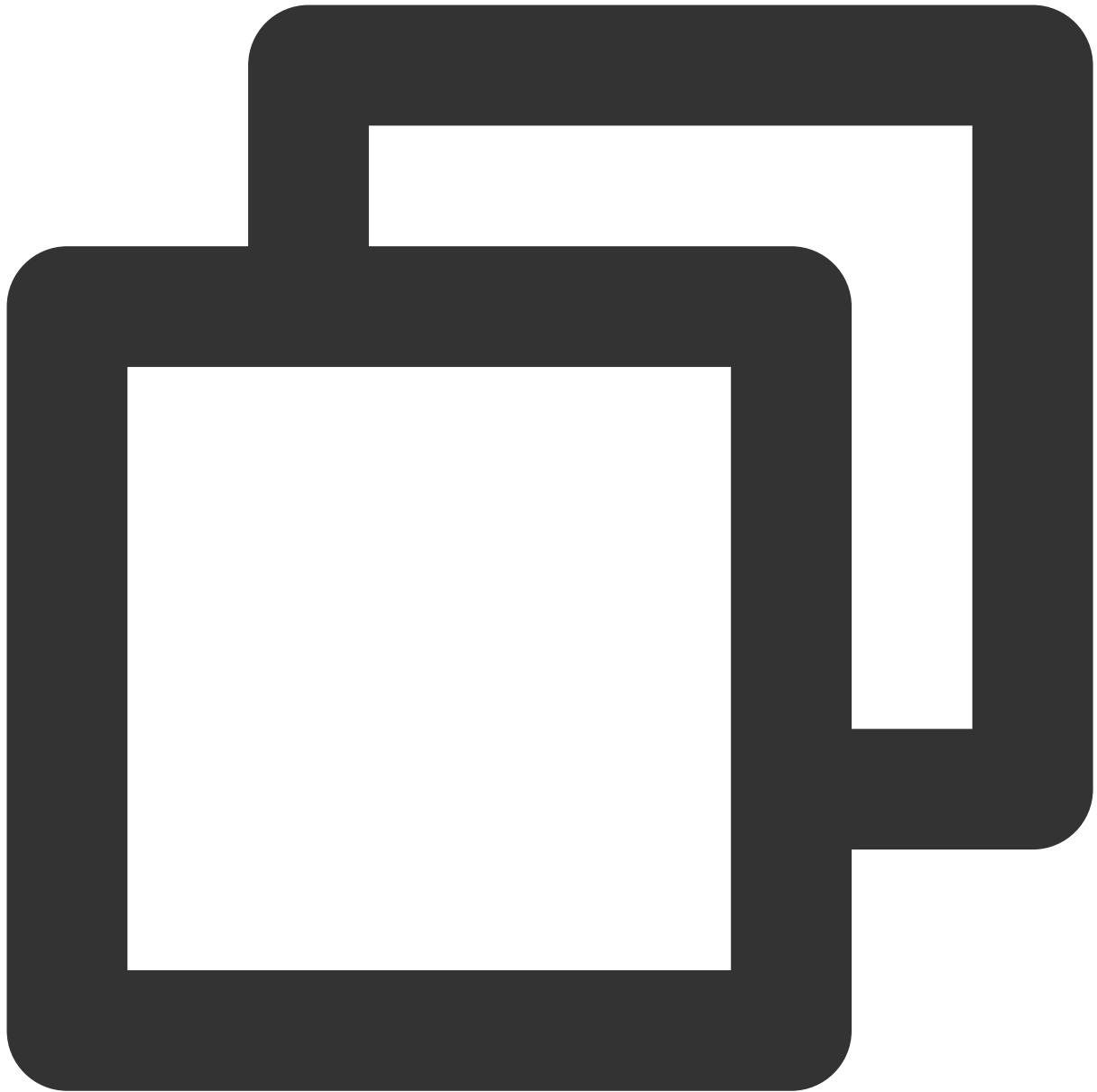
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type
view	string	Required	-	The id of the div element playing the remote user's stream

Returns : *Promise<void>*

## startPlayRemoteVideo

Start Playback of Remote User Video Stream



```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning

userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type TUIVideoStreamType.kCameraStream Video Stream TUIVideoStreamType.kScreenStream Screen Sharing Stream TUIVideoStreamType.kCameraStreamLow Low Definition Video Stream

Returns : *Promise<void>*

## stopPlayRemoteVideo

Stop Playback of Remote User Video Stream



```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

Parameter:

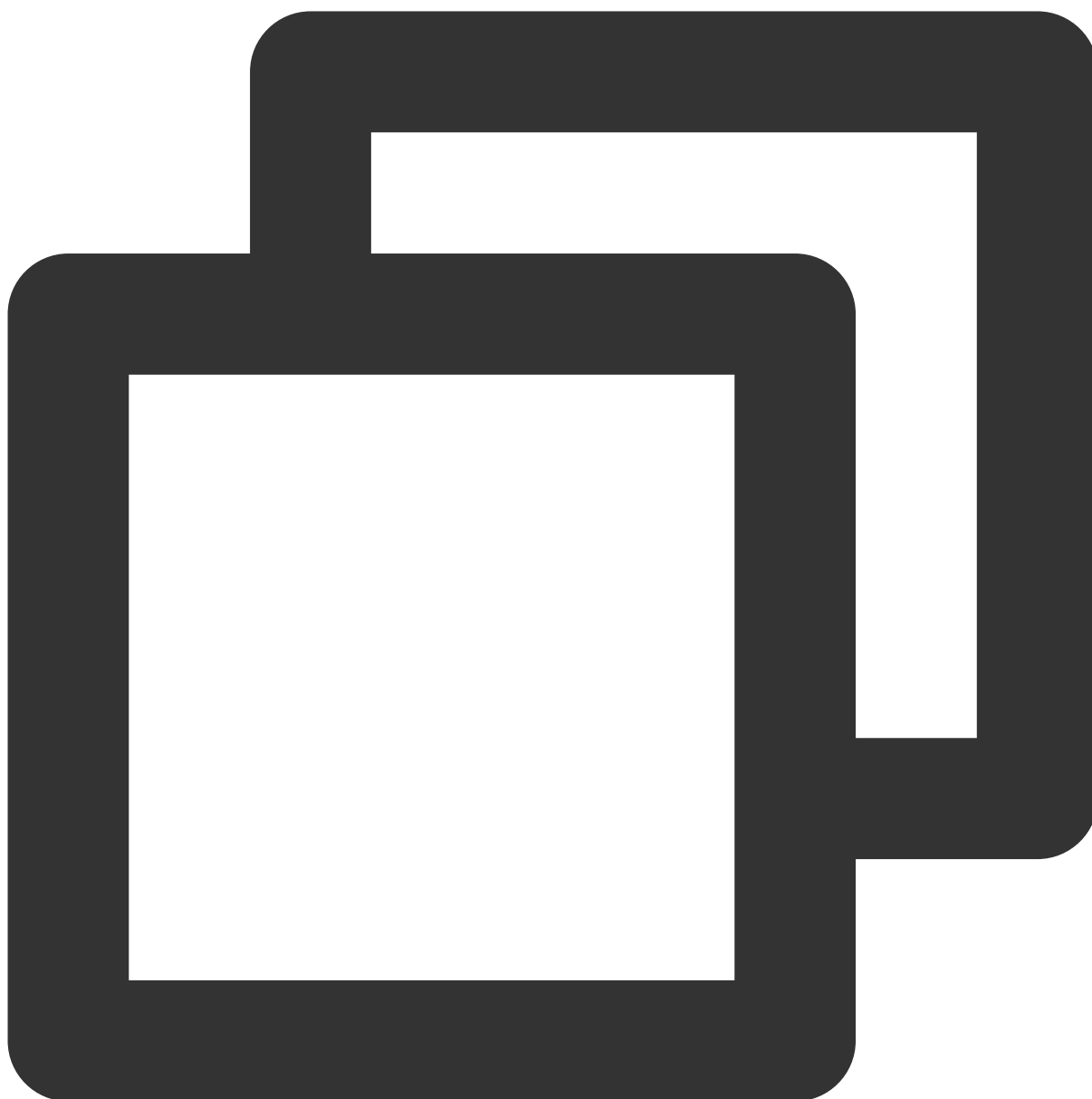
Parameter	Type	Description	Default Value	Meaning

userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type TUIVideoStreamType.kCameraStream Video Stream TUIVideoStreamType.kScreenStream Screen Sharing Stream TUIVideoStreamType.kCameraStreamLow Low Definition Video Stream

Returns : *Promise<void>*

## muteRemoteAudioStream

Stop Remote User's Audio Stream



```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteRemoteAudioStream({
  userId: 'user_1234',
  isMute: true,
});
```

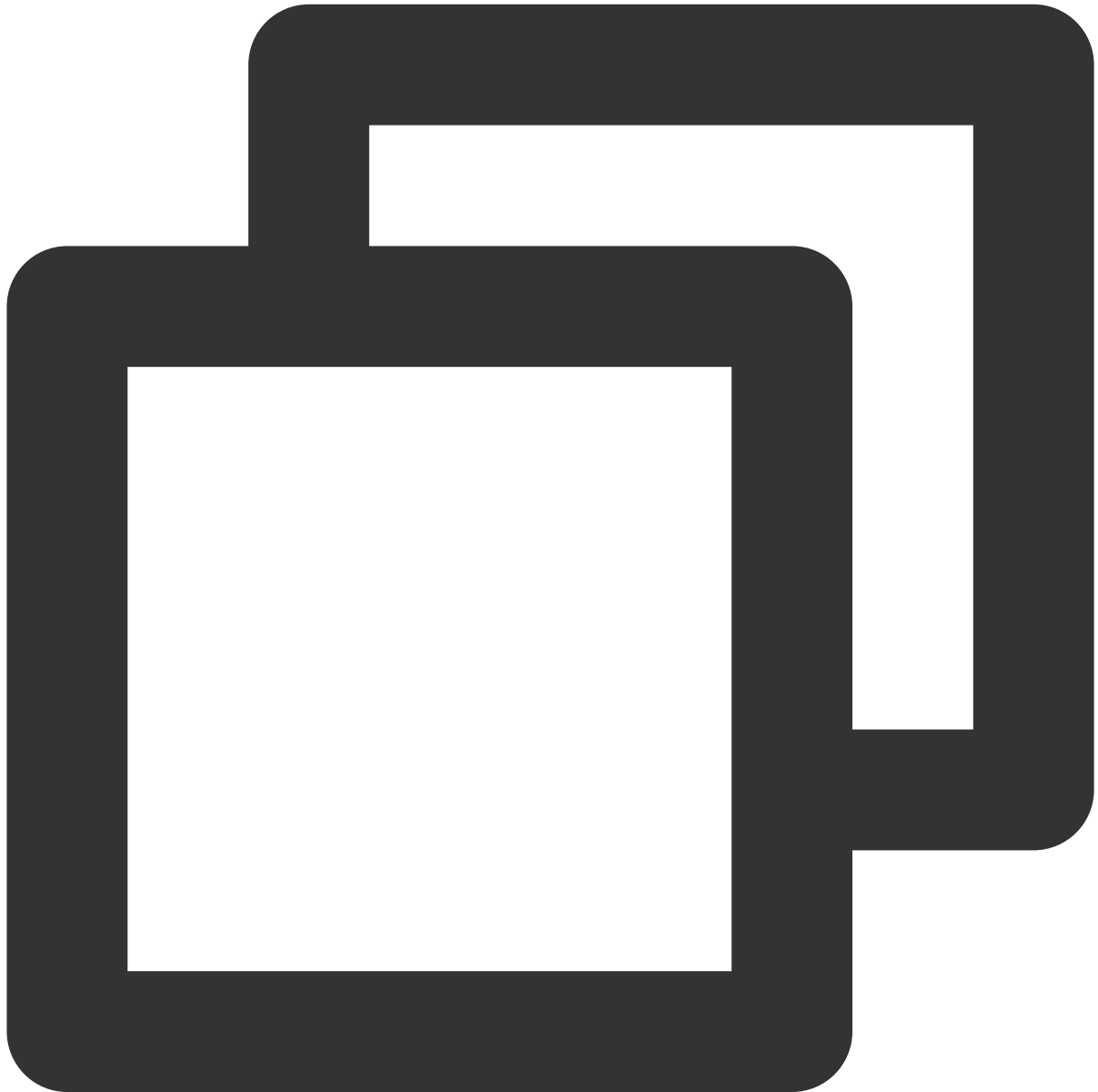
Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID



isMute	boolean	Required	-	Whether to Stop Remote User's Audio
--------	---------	----------	---	-------------------------------------

## openRemoteDeviceByAdmin

Request Remote User to Open Media Device



```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.openRemoteDeviceByAdmin({
  userId: 'user_1234',
```

```

device: TUIMediaDevice.kMicrophone //The requested device is a mic
timeout: 0,
requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
  switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // Request Accepted
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // Request Rejected
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // Request Canceled
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // Request Timeout
      break;
    case TUIRequestCallbackType.kRequestError:
      // Request Error
      break;
    default:
      break;
  }
},
});

```

**Parameter:**

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
device	TUIMediaDevice	Required	-	Media Device Type (Camera/Mic/Screen Sharing)
timeout	number	Required	-	Timeout Time. If timeout is set to 0, there is no timeout time
requestCallback	Function	Required	Empty Function	Request Callback, used to notify the initiator of the request being accepted/rejected/canceled/timeout/error

Returns : *Promise<string> requestId*

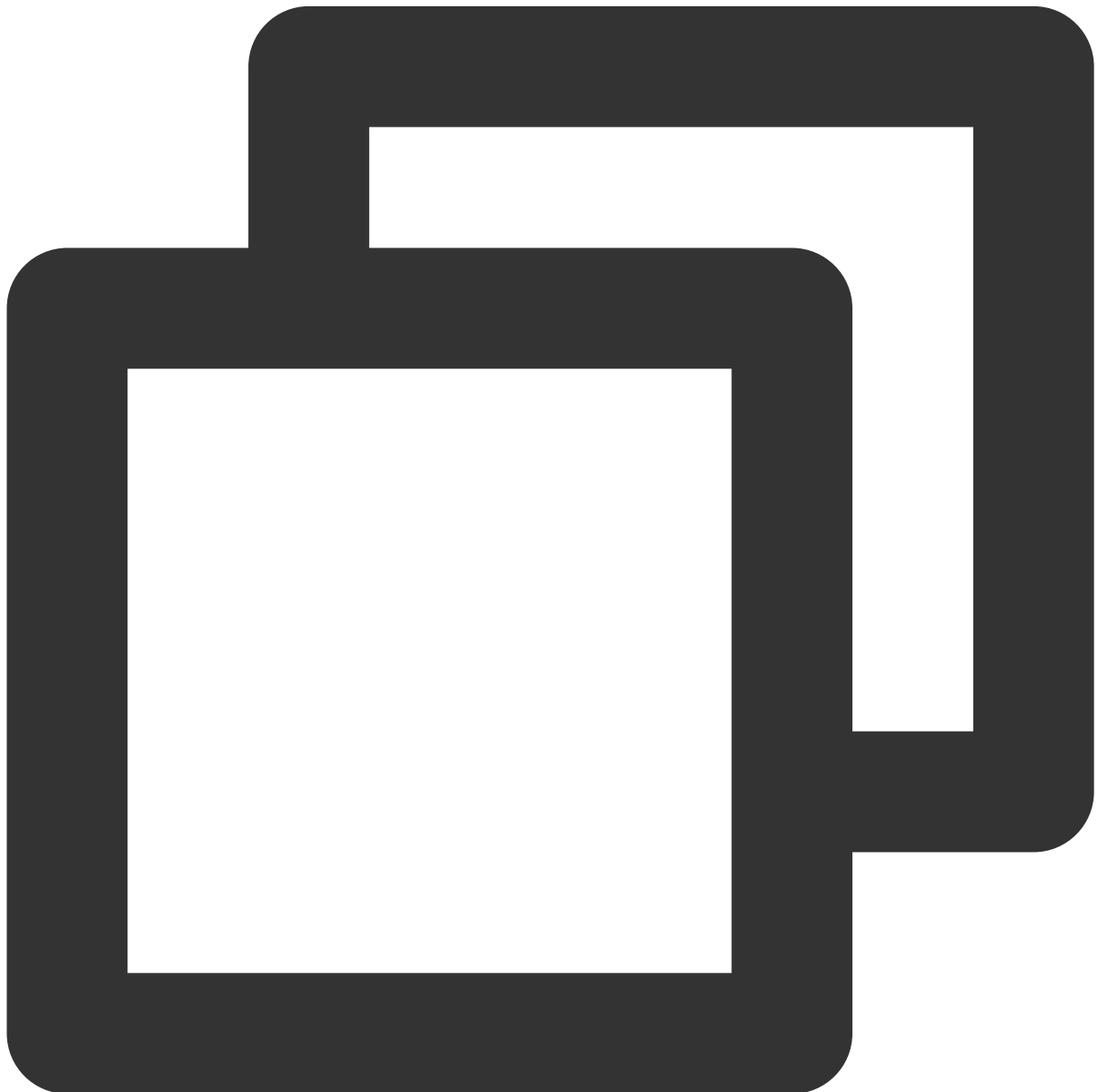
This interface returns requestId, users can use this requestId to call cancelRequest interface to cancel the request.

**Description:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## applyToAdminToOpenLocalDevice

Participant applies to the host to open the device



```
const roomEngine = new TUIRoomEngine();  
const requestId = roomEngine.applyToAdminToOpenLocalDevice({
```

```
device: TUIMediaDevice.kMicrophone //The requested device is a mic
timeout: 0,
requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
  switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // Request Accepted
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // Request Rejected
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // Request Canceled
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // Request Timeout
      break;
    case TUIRequestCallbackType.kRequestError:
      // Request Error
      break;
    default:
      break;
  }
},
});
```

Parameter:

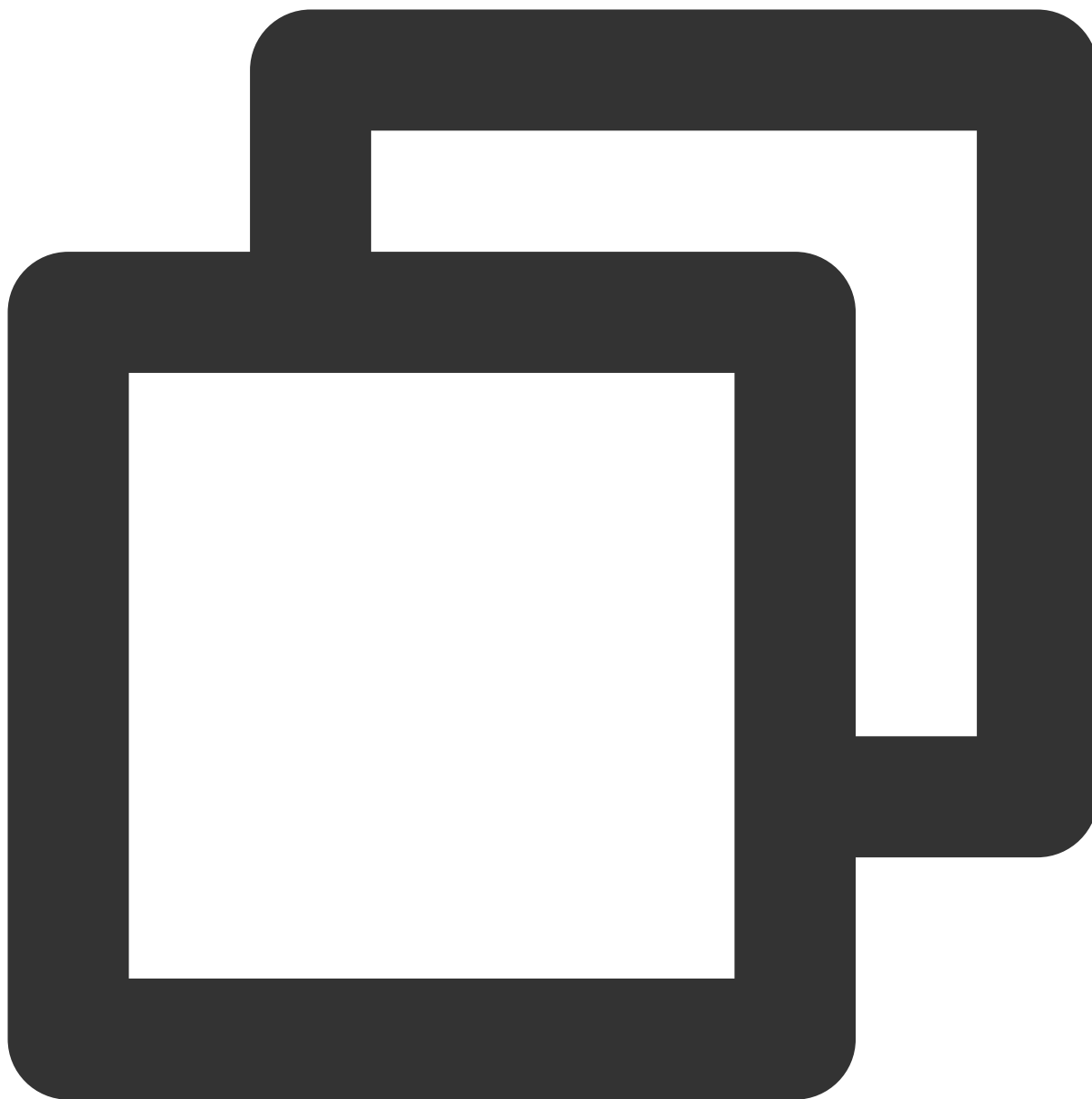
Parameter	Type	Description	Default Value	Meaning
device	TUIMediaDevice	Required	-	Media Device Type (Camera/Mic/Screen Sharing)
timeout	number	Required	-	Timeout Time. If timeout is set to 0, there is no timeout time
requestCallback	Function	Optional	Empty Function	Request Callback, used to notify the initiator of the request being accepted/rejected/canceled/timeout/error

Returns : *Promise<string> requestId*

This interface returns requestId, users can use this requestId to call cancelRequest interface to cancel the request

## closeRemoteDeviceByAdmin

## Close Remote User Media Device



```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone, //Close mic
});
```

## Parameter:

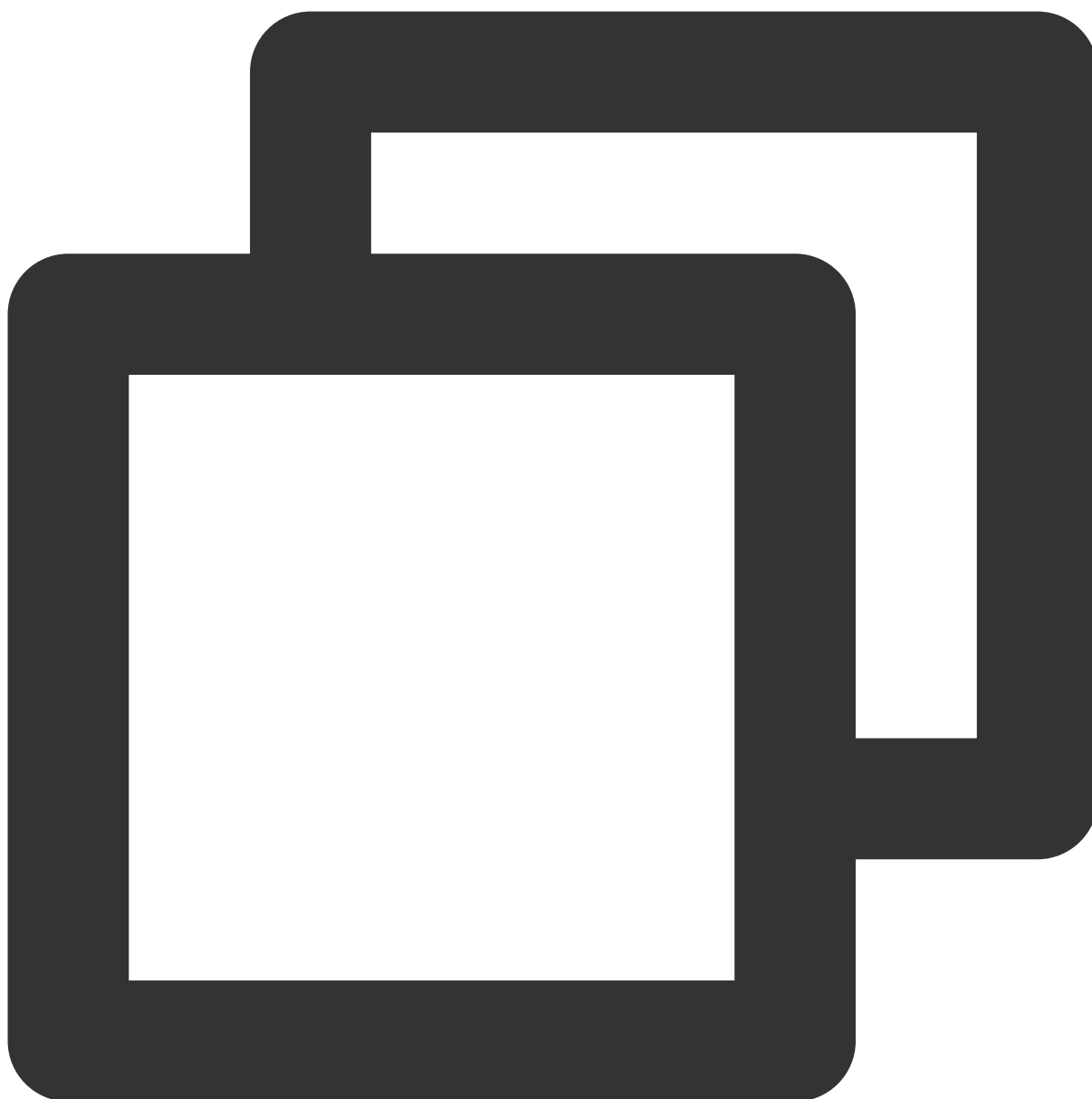
Parameter	Type	Description	Default Value	Meaning
-----------	------	-------------	---------------	---------

userId	string	Required	-	User ID
device	TUIMediaDevice	Required	-	Media Device Type (Camera/Mic/Screen Sharing)

Returns : *Promise<void>*

## cancelRequest

Cancel Already Sent Request



```
const roomEngine = new TUIRoomEngine();
await roomEngine.cancelRequest({
  requestId: '',    // Please use Actual requestId
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
requestId	string	Required	-	Request ID

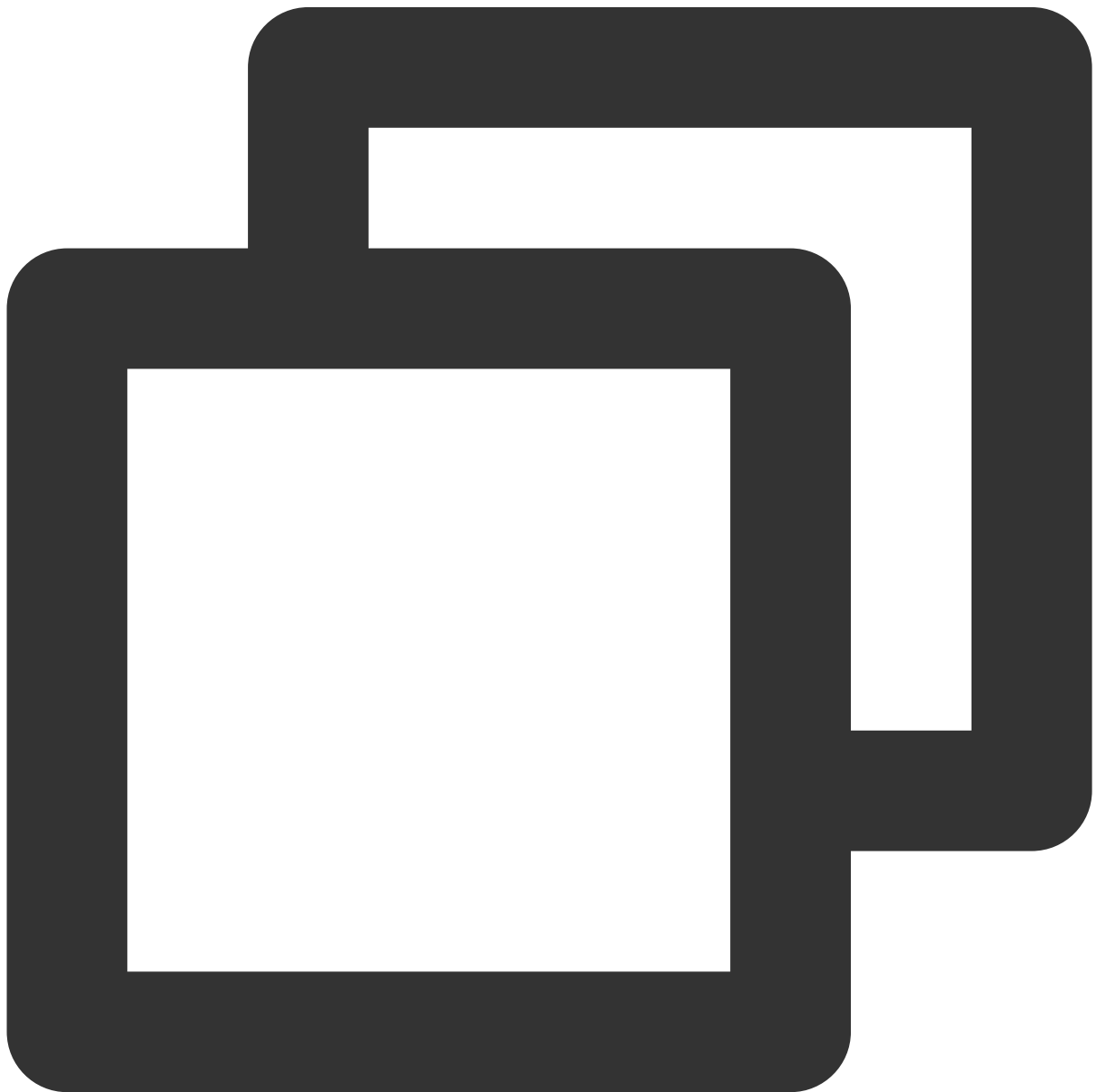
Returns : *Promise<void>*

**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## responseRemoteRequest

Reply to Remote User's Request





```
const roomEngine = new TUIRoomEngine();
// Agree to Remote User's Request
await roomEngine.responseRemoteRequest({
  requestId: '',    // Please use Actual requestId
  agree: true,
});
// Reject Remote User's Request
await roomEngine.responseRemoteRequest({
  requestId: '',    // Please use Actual requestId
  agree: false,
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
requestId	string	Required	-	Request ID
agree	boolean	Required	-	Whether to Agree

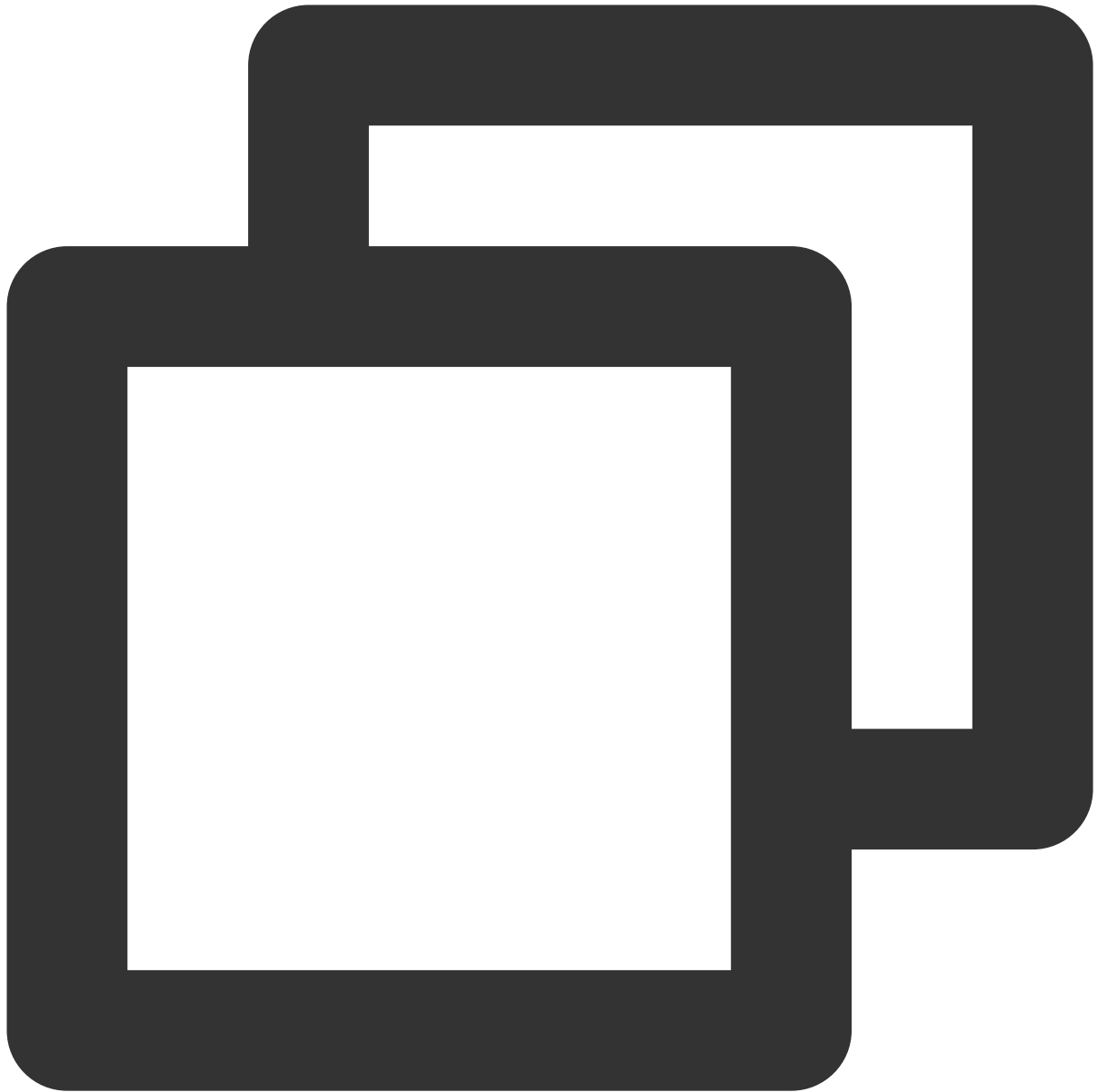
Returns : *Promise<void>*

**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## disableDeviceForAllUserByAdmin

Prohibit/Allow All Users to Open Media Device (This Interface is invalid for Room Owner and Administrator)



```
// Example 1: Prohibit All Users to Open Mic
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: true,
})
// Example 2: Allow All Users to Open Mic
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: false,
})
```

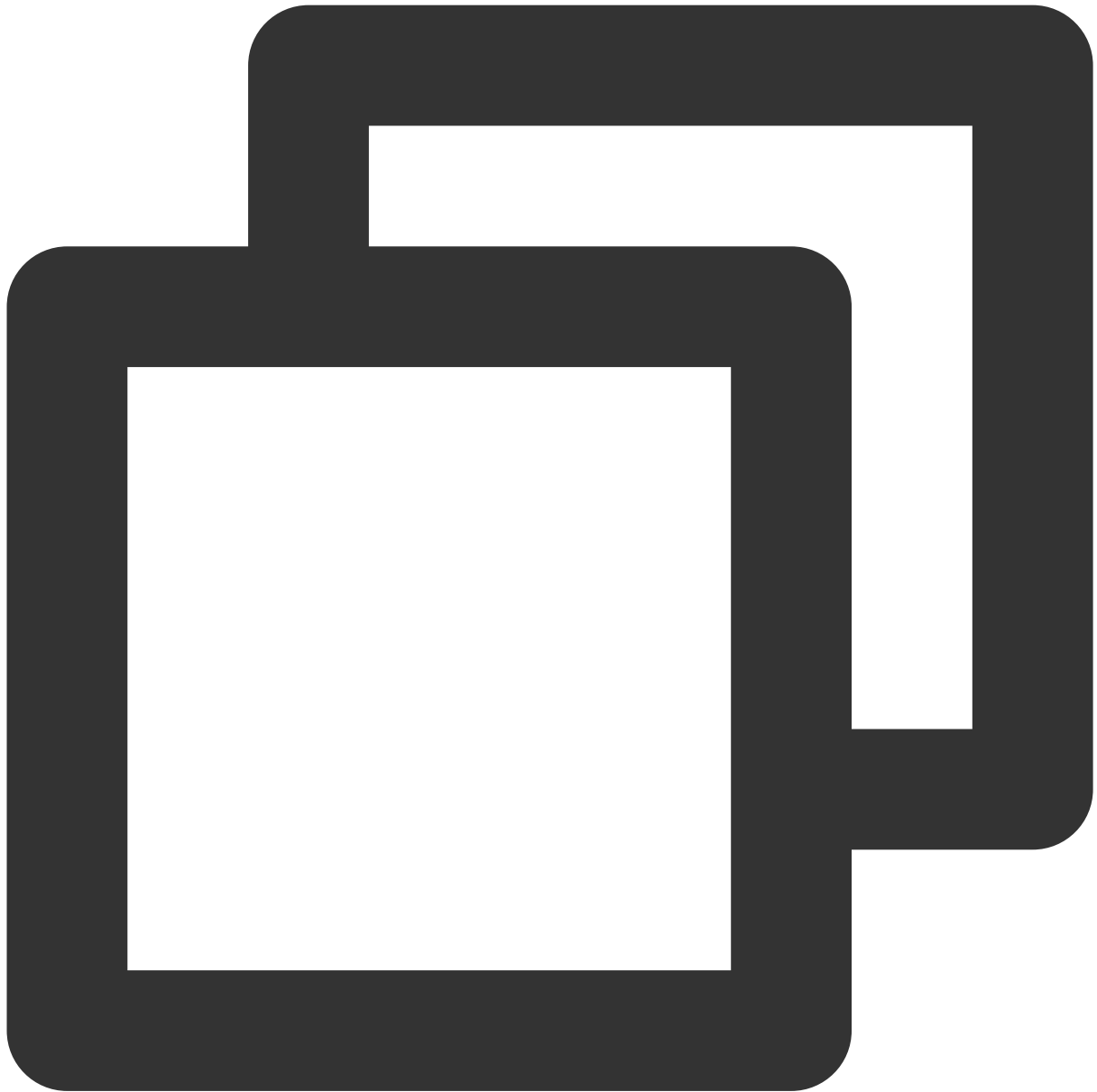
Parameter:

Parameter	Type	Description	Default Value	Meaning
device	TUIMediaDevice	Required	-	Disabled Media Device Type (Camera/Mic/Screen Sharing)
isDisable	boolean	Required	-	Whether it is Prohibited

Returns : *Promise<void>*

## disableSendingMessageForAllUser

Whether All Users are Allowed to Send Messages (This Interface is invalid for Room Owner and Administrator)



```
await roomEngine.disableSendingMessageForAllUser({  
  isDisable: true,  
});
```

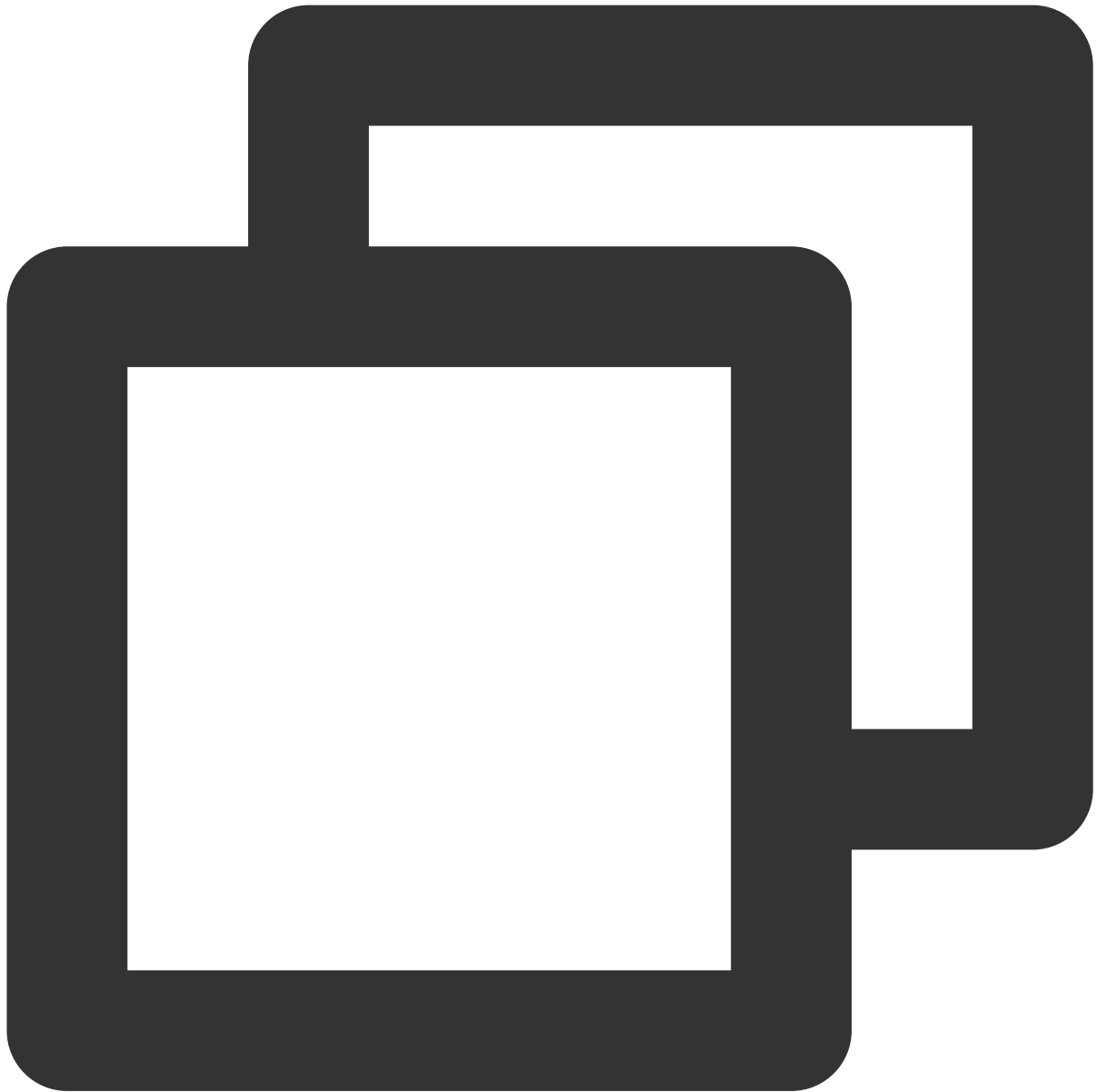
Parameter:

Parameter	Type	Description	Default Value	Meaning
isDisable	boolean	Required	-	Whether it is Disabled

Returns : *Promise<void>*

## disableSendingMessageByAdmin

Whether Specific User is Allowed to Send Messages



```
await roomEngine.disableSendingMessageByAdmin({
  userId: 'user_1234',
  isDisable: true,
```

```
});
```

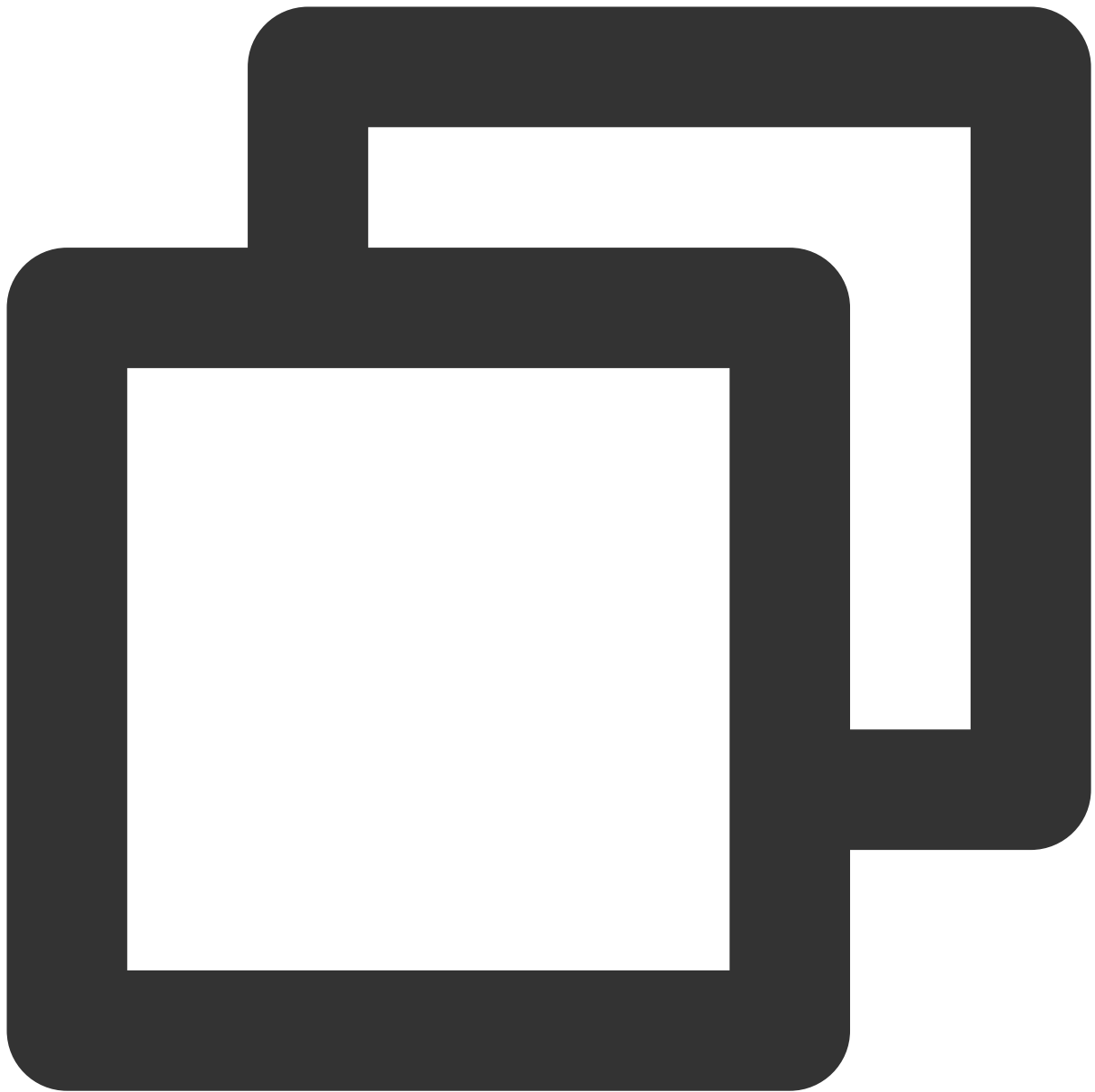
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
isDisable	boolean	Required	-	Whether it is Disabled

Returns : *Promise<void>*

## changeUserRole

Change User's Role (Only the Host can call this Interface)



```
const roomEngine = new TUIRoomEngine();

// Transfer the Room to User user_1234
await roomEngine.changeUserRole({
  userId: 'user_1234',
  role: TUIRole.kRoomOwner,
});

// Set user_1234 as Room Administrator
await roomEngine.changeUserRole({
  userId: 'user_1234',
```

```
    userRole: TUIRole.kAdministrator,  
  });
```

Parameter:

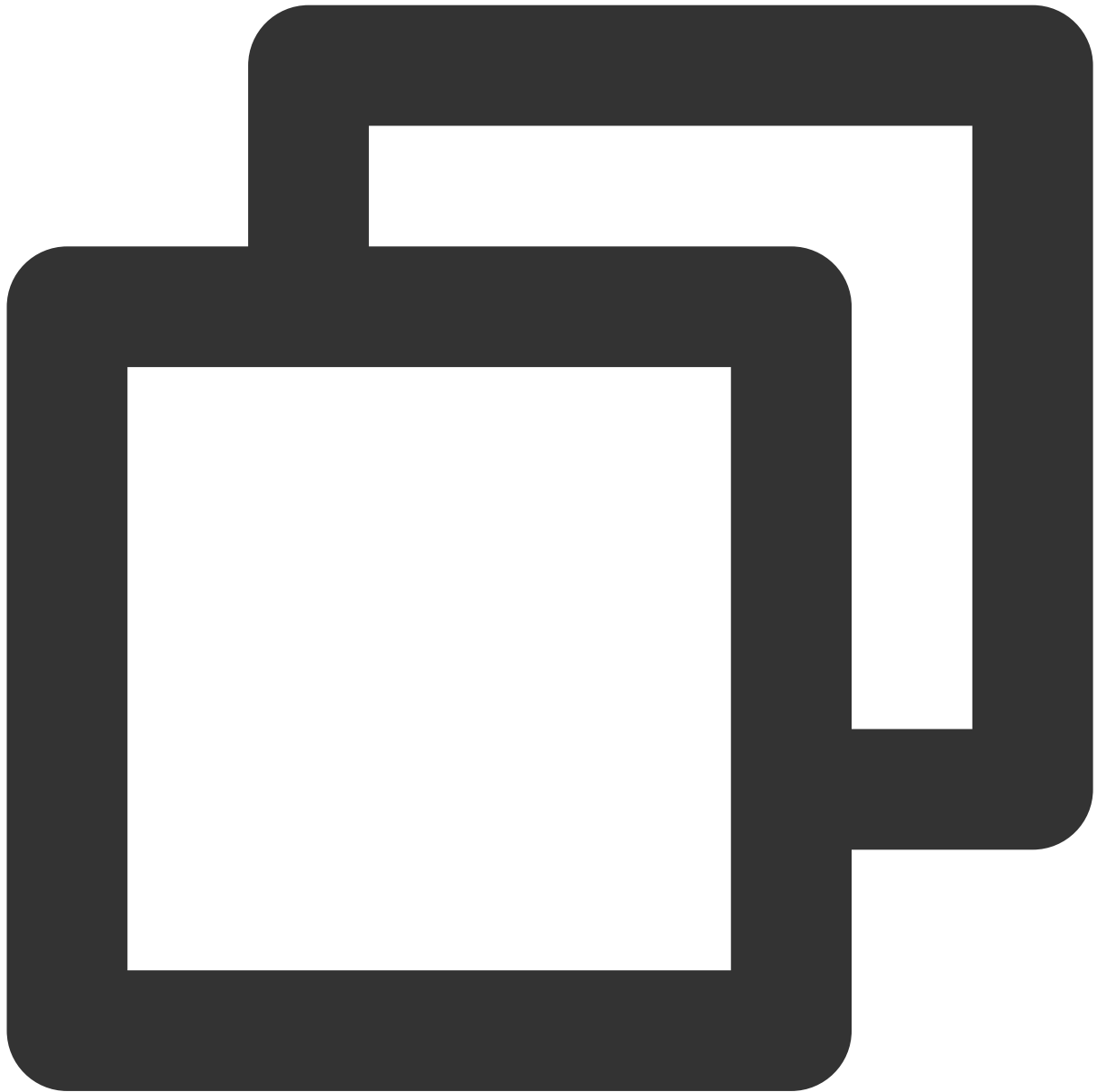
Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
userRole	TUIRole	Required	-	User Role Host TUIRole.kRoomOwner Administrator TUIRole.kAdministrator General Member TUIRole.kGeneralUser

Returns : *Promise<void>*

## kickRemoteUserOutOfRoom

Kick Out User from Room (Only Host and Administrator can call this Interface)





```
const roomEngine = new TUIRoomEngine();
await roomEngine.kickRemoteUserOutOfRoom({
  userId: 'user_1234',
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID

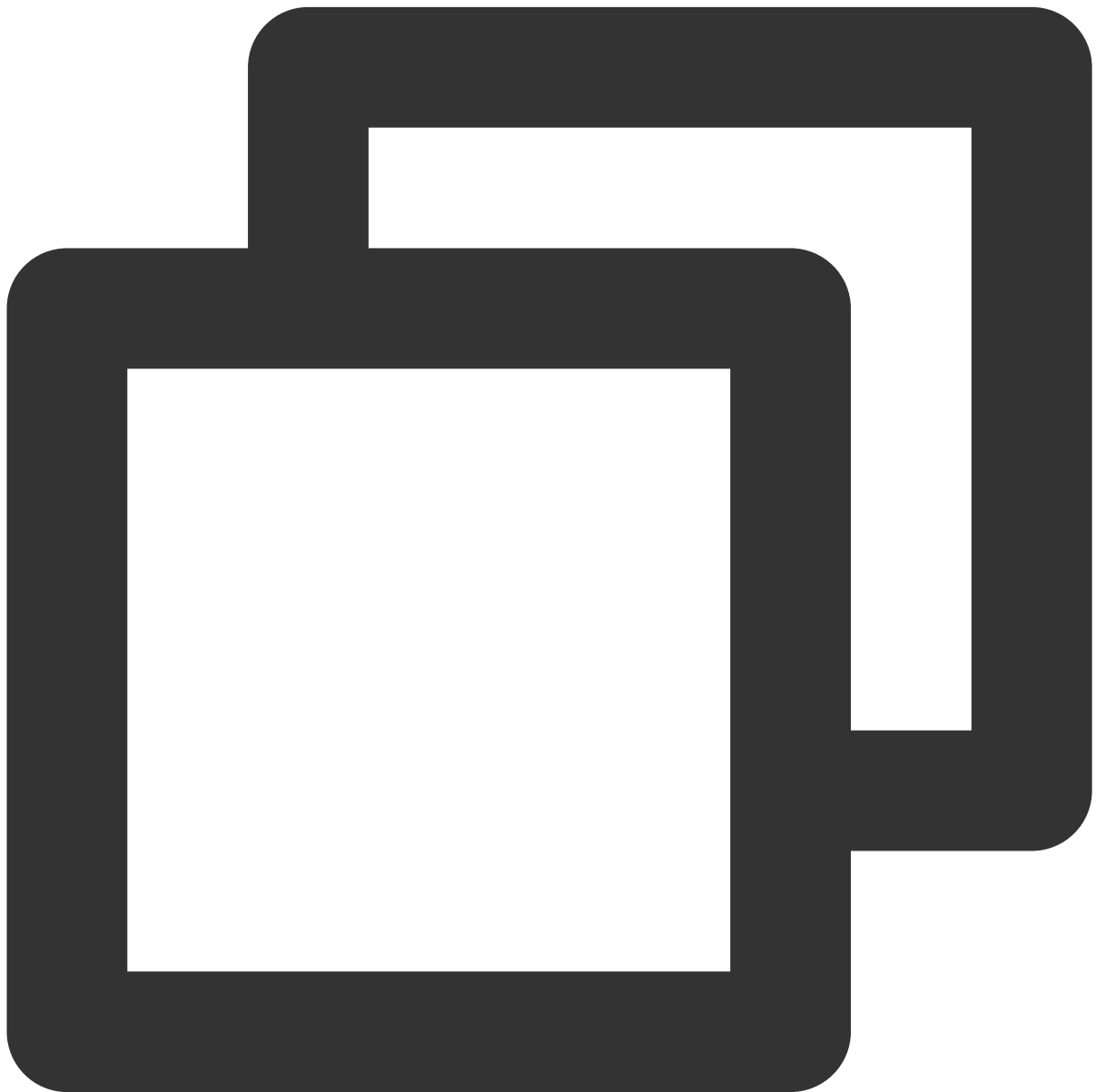
Returns : *Promise<void>*

## setMaxSeatCount

Set Room Seat Maximum Value

When roomType is TUIRoomType.kConference (Education and Conference Scene), maxSeatCount value is not limited;

When roomType is TUIRoomType.kLivingRoom (Live Scene), maxSeatCount is limited to 16;



```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.setMaxSeatCount({ maxSeatCount: 16 })
```

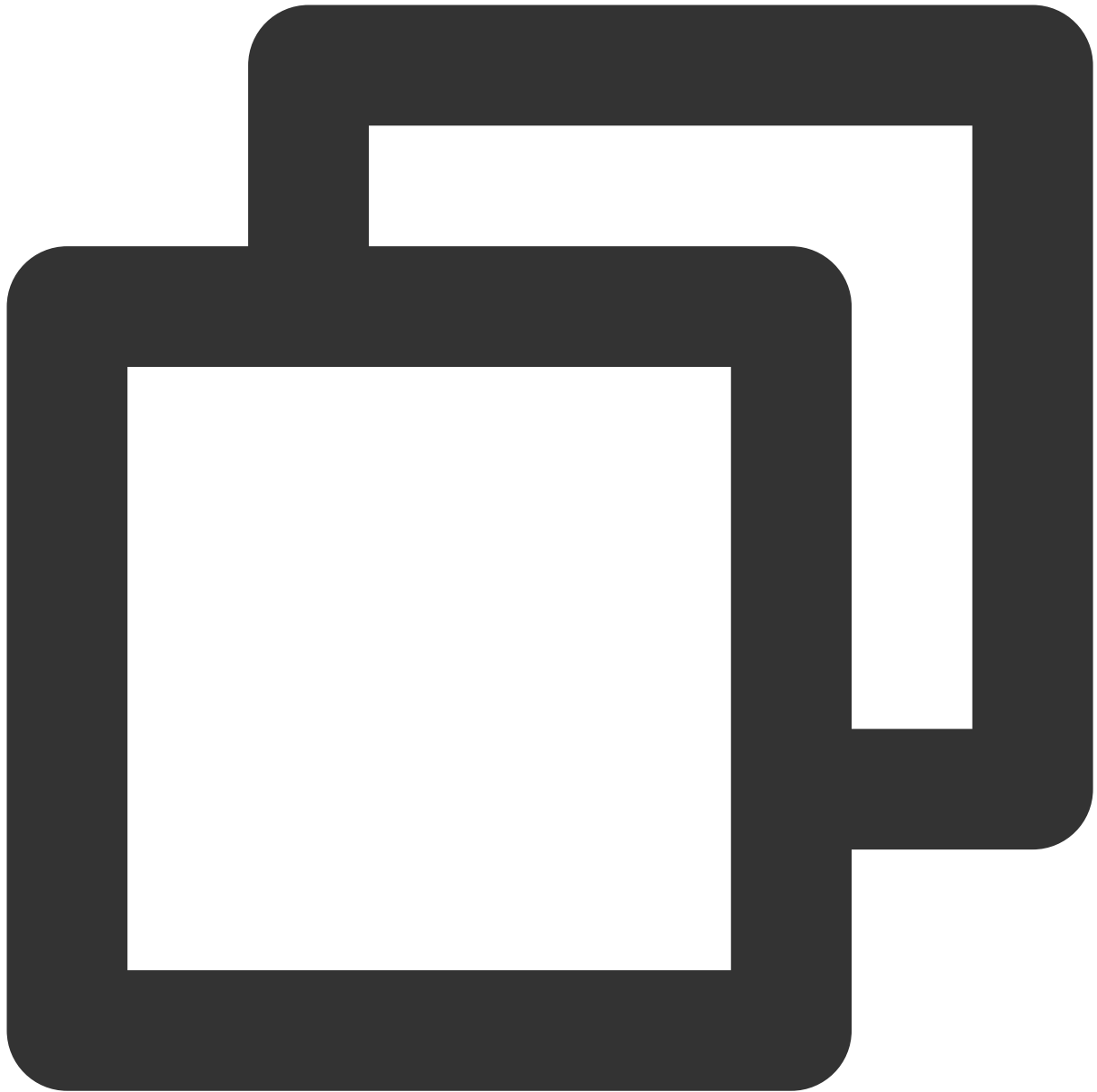
Parameter:

Parameter	Type	Description	Default Value	Meaning
maxSeatCount	number	Required	-	Set Room Seat Maximum Value

Returns : Promise<void>

## getSeatList

Get Seat List



```
const roomEngine = new TUIRoomEngine();  
const seatList = await roomEngine.getSeatList();
```

Returns : *Promise*<[TUISeatInfo](#)[]> seatList  
seatList for the Current Room's All Seat List

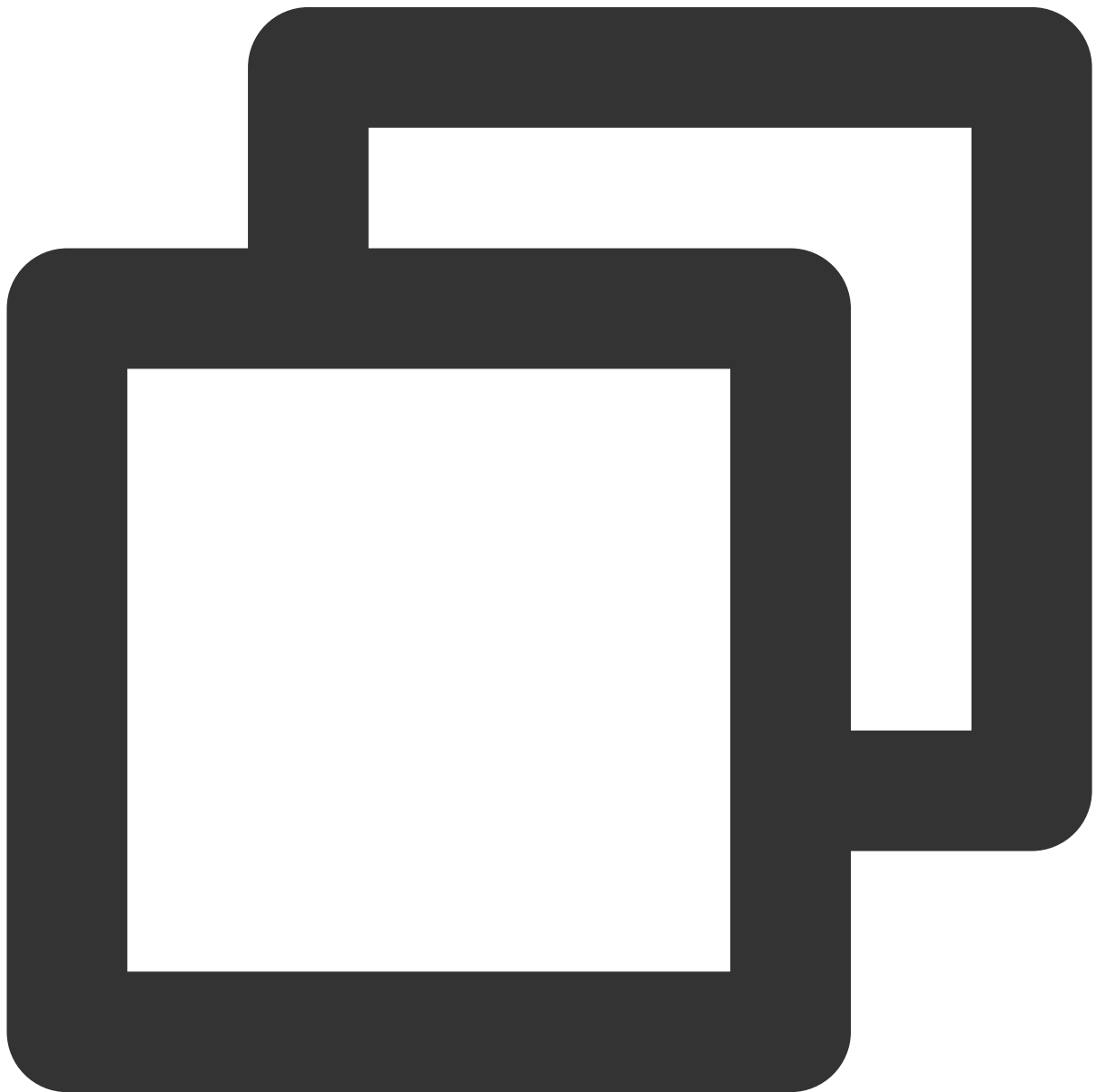
## takeSeat

Mic-off Users can call `takeSeat` to become Mic-on Users, only Mic-on Users can Publish Local stream.

When `roomInfo.roomType` is `TUIRoomType.kConference` and `roomInfo.speechMode` is `TUISpeechMode.kSpeakAfterTakingSeat`, General Users need to Wait for the Host/Administrator's Agreement to become Mic-on Users after calling `takeSeat` method.

When `roomInfo.roomType` is `TUIRoomType.kLivingRoom` and `roomInfo.speechMode` is `TUISpeechMode.kFreeToSpeak`, General Users become Mic-on Users after calling `takeSeat` method Successfully. Host & Administrator become Mic-on Users after calling `takeSeat` Successfully.

Changes of Mic-on Users are Notified to All Users through `TUIRoomEvents.onSeatListChanged`.



```
const roomEngine = new TUIRoomEngine();
```

```
// Scenario 1: Host/Administrator Go Live
// Scenario 2: When roomInfo.roomType is TUIRoomType.kConference
// and roomInfo.speechMode is TUISpeechMode.kSpeakAfterTakingSeat, General Users Go
await roomEngine.takeSeat({
  seatIndex: -1,
  timeout: 0,
});

// Scenario 3: When roomInfo.enableSeatControl is true, General Users Go Live
const requestId = await roomEngine.instance?.takeSeat({
  seatIndex: -1,
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
    switch (requestCallbackType) {
      case TUIRequestCallbackType.kRequestAccepted:
        // Request Accepted
        break;
      case TUIRequestCallbackType.kRequestRejected:
        // Request Rejected
        break;
      case TUIRequestCallbackType.kRequestCancelled:
        // Request Canceled
        break;
      case TUIRequestCallbackType.kRequestTimeout:
        // Request Timeout
        break;
      case TUIRequestCallbackType.kRequestError:
        // Request Error
        break;
      default:
        break;
    }
  },
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Seat index, set to -1 when there is No Serial Number
timeout	number	Required	-	Timeout. If timeout is set to 0, there is no Timeout
requestCallback	Function	Optional	Empty Function	Callback, used to Notify Initiator of Request being Accepted/Rejected/Cancelled/Timeout/Error

Returns : Promise<string> requestId

When roomInfo.enableSeatControl is true, General Users call this Interface to return requestId, General Users can use this requestId to call cancelRequest Interface to cancel Go Live Request.

**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## leaveSeat

Release Seat



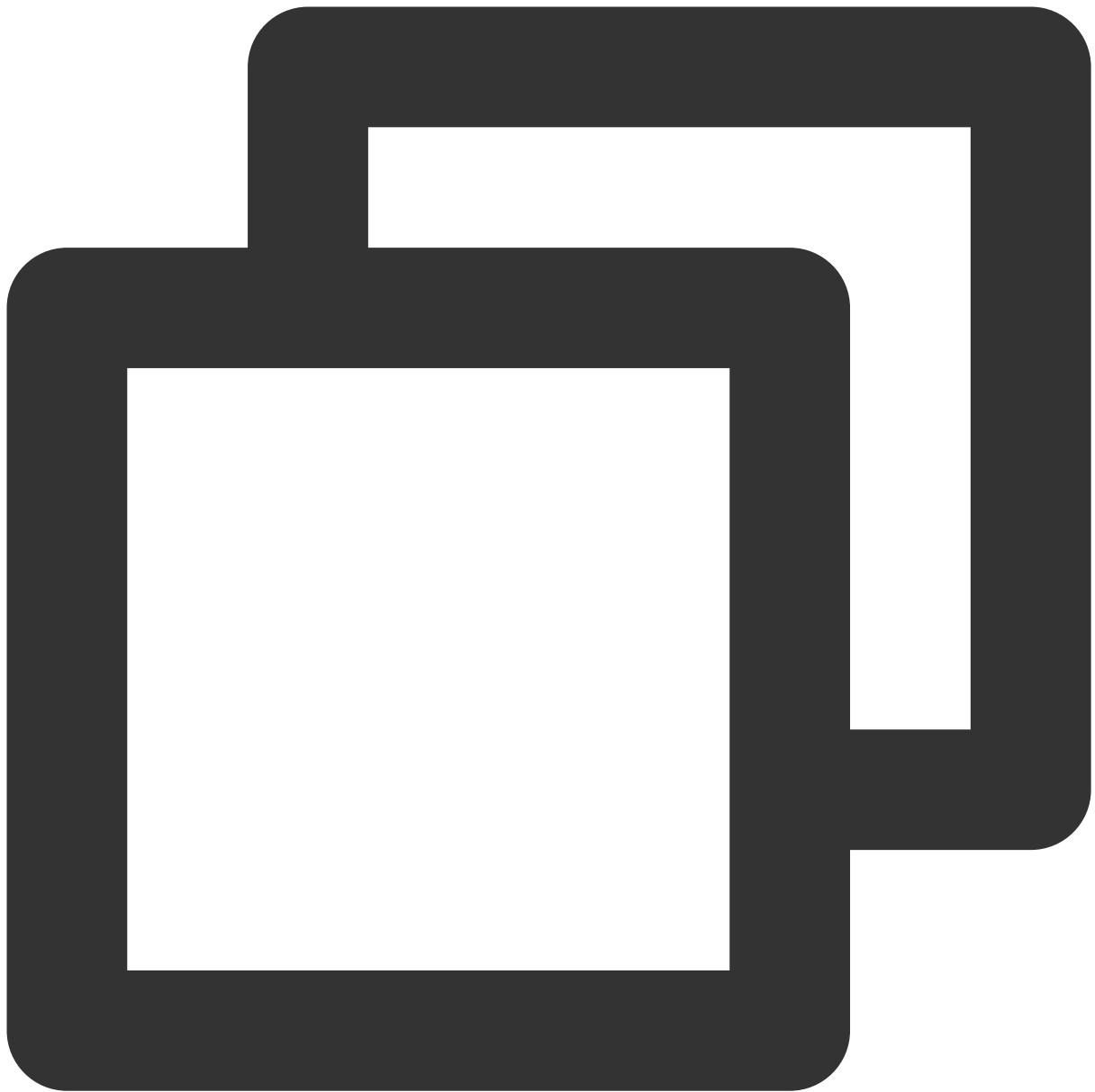
```
const roomEngine = new TUIRoomEngine();  
await roomEngine.leaveSeat();
```

Returns : *Promise<void>*

## takeUserOnSeatByAdmin

Invite Others to Go Live





```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.takeUserOnSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
    switch (requestCallbackType) {
      case TUIRequestCallbackType.kRequestAccepted:
        // Request Accepted
        break;
      case TUIRequestCallbackType.kRequestRejected:
```

```

        // Request Rejected
        break;
    case TUIRequestCallbackType.kRequestCancelled:
        // Request Canceled
        break;
    case TUIRequestCallbackType.kRequestTimeout:
        // Request Timeout
        break;
    case TUIRequestCallbackType.kRequestError:
        // Request Error
        break;
    default:
        break;
    }
},
});

```

Parameter:

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Seat index
userId	string	Required	-	User ID
timeout	number	Required	-	Timeout. If timeout is set to 0, there is no Timeout
requestCallback	Function	Optional	Empty Function	Callback, used to Notify Initiator of Request being Accepted/Rejected/Cancelled/Timeout/Error

Returns : *Promise<string>* requestId

This Interface returns requestId, Users can use this requestId to call cancelRequest Interface to cancel Request

#### Note:

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number

## kickUserOffSeatByAdmin

Ask others to get off the mic



```
const roomEngine = new TUIRoomEngine();
const requestId = await roomEngine.kickUserOffSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
});
```

Parameter:

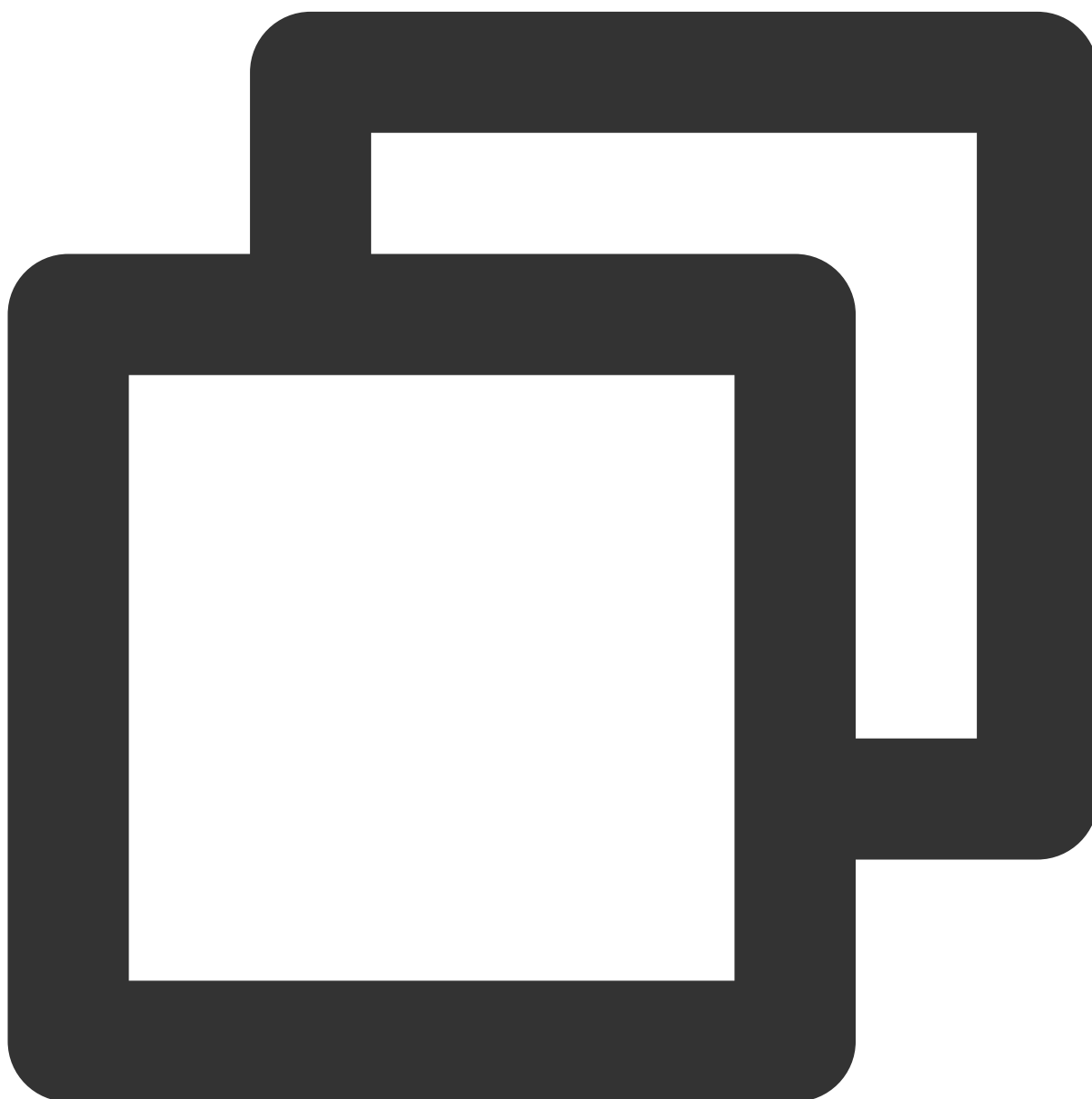
Parameter	Type	Description	Default Value	Meaning

seatIndex	number	Required	-	Seat index
userId	string	Required	-	User ID

Returns : *Promise<void>*

## lockSeatByAdmin

Lock a certain mic position status (Only the room host and administrator can call this method)



```
const roomEngine = new TUIRoomEngine();
await roomEngine.lockSeatByAdmin({
  seatIndex: 0,
  lockParams: {
    lockSeat: true,
    lockVideo: true,
    lockAudio: true,
  },
});
```

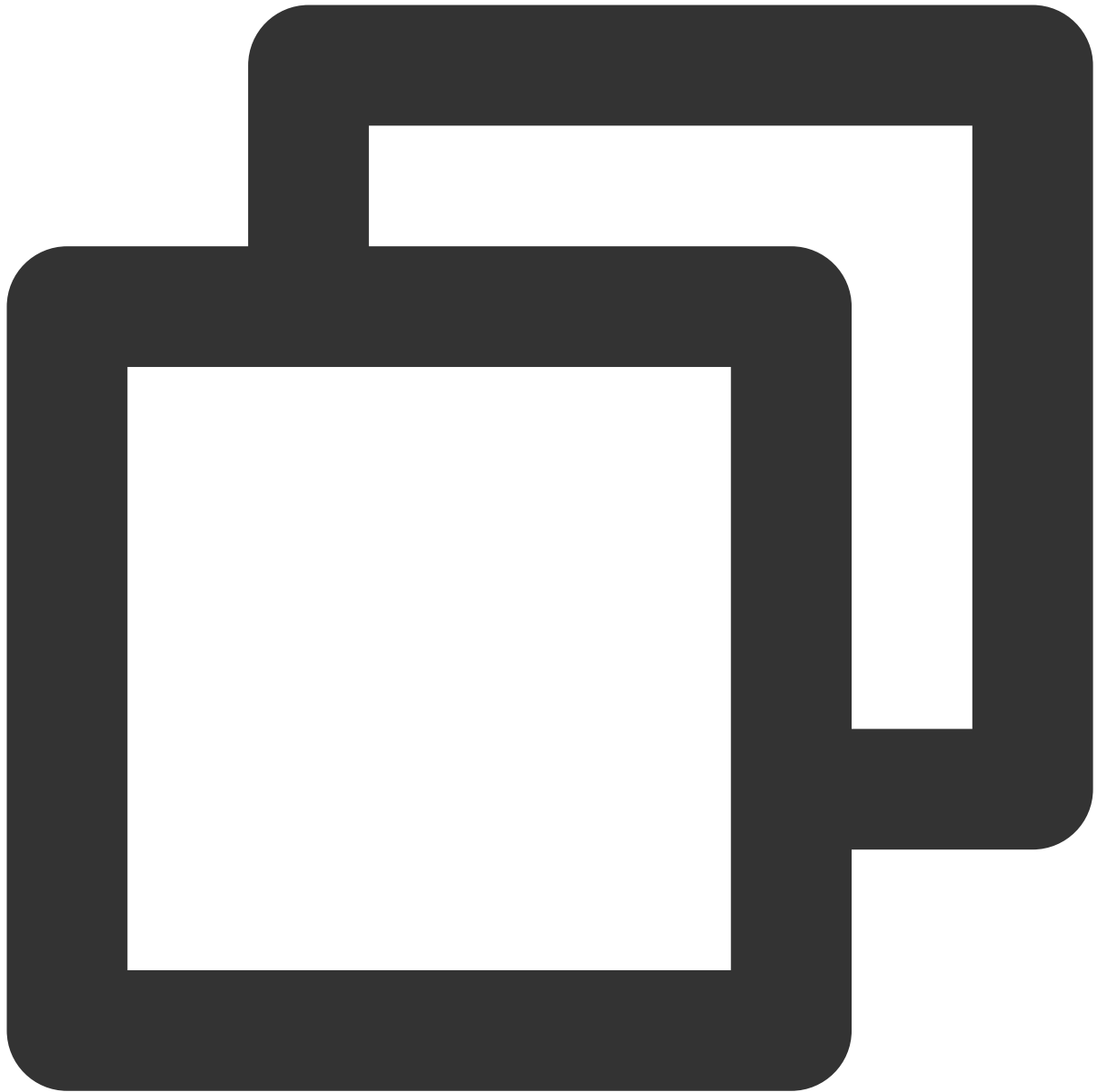
Parameter:

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Mic position index
lockParams	TUISeatLockParams	Required	-	Lock mic parameter

Returns : *Promise<void>*

## sendTextMessage

Send text message



```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendTextMessage({
  messageText: 'hello, everyone',
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
messageText	string	Required	-	Text message content

Returns : *Promise<void>*

## sendCustomMessage

Send custom message



```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendCustomMessage({
  messageText: '{ data:', description: '}',
```

```
});
```

Parameter:

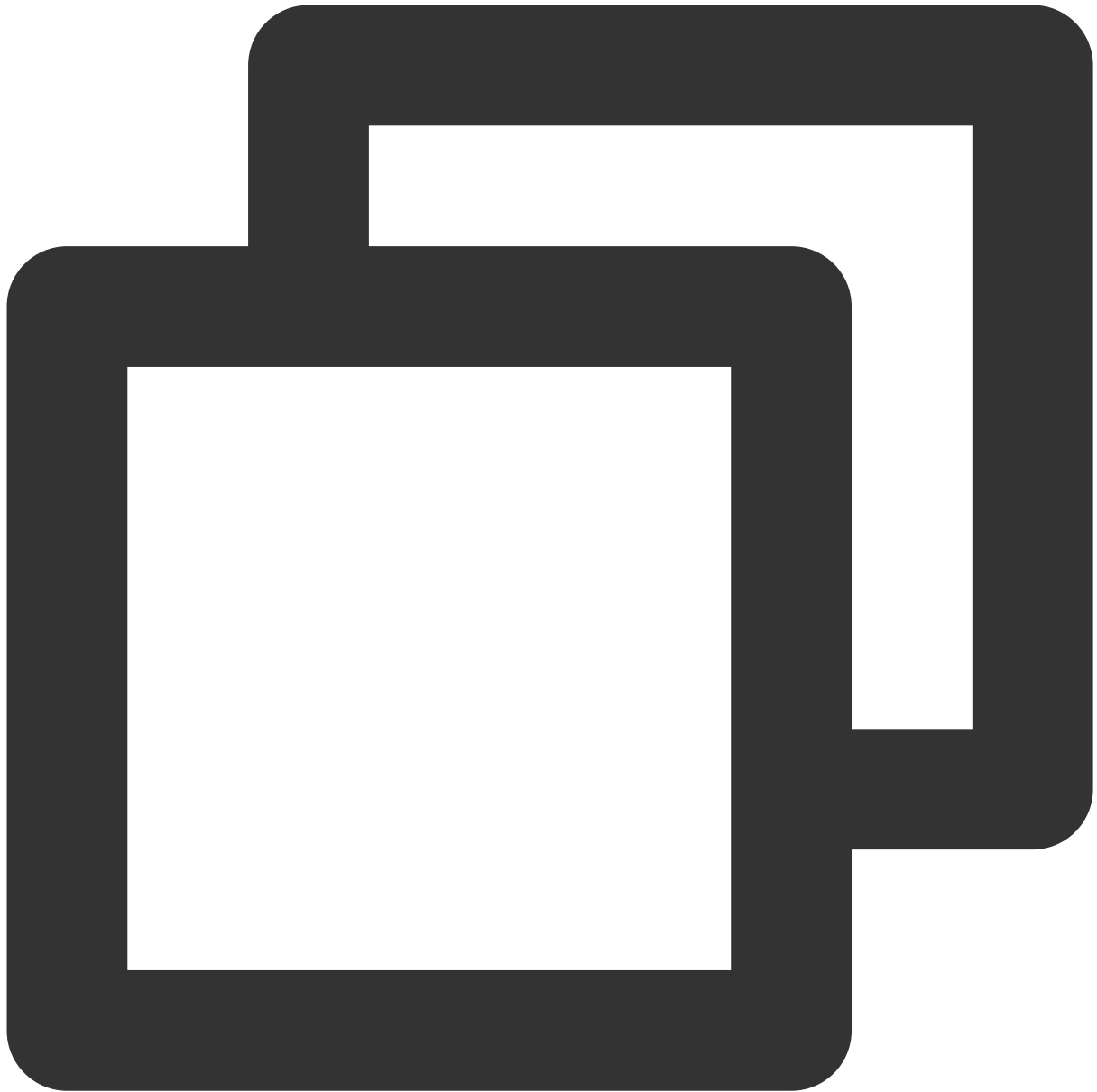
Parameter	Type	Description	Default Value	Meaning
messageText	string	Required	-	Custom message content

Returns : *Promise*<void>

## getCameraDevicesList

Get camera device list



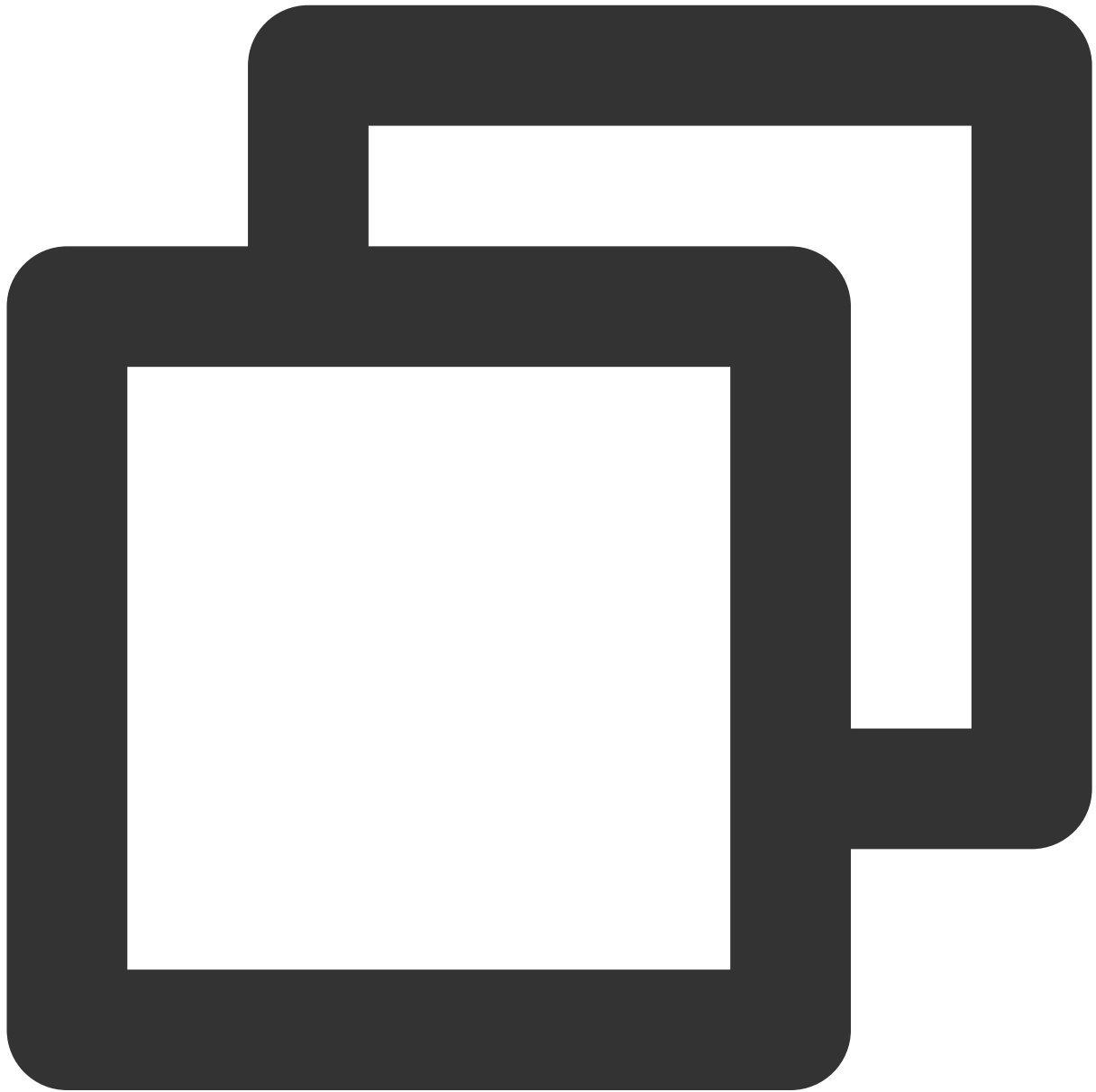


```
const roomEngine = new TUIRoomEngine();
const cameraList = await roomEngine.getCameraDevicesList();
for (i = 0; i < cameraList.length; i++) {
  var camera = cameraList[i];
  console.info("camera deviceName: " + camera.deviceName + " deviceId:" + camera.
}
```

Returns: *Promise*<[TRTCDeviceInfo\[\]](#)> cameraList

## getMicDevicesList

Get mic device list

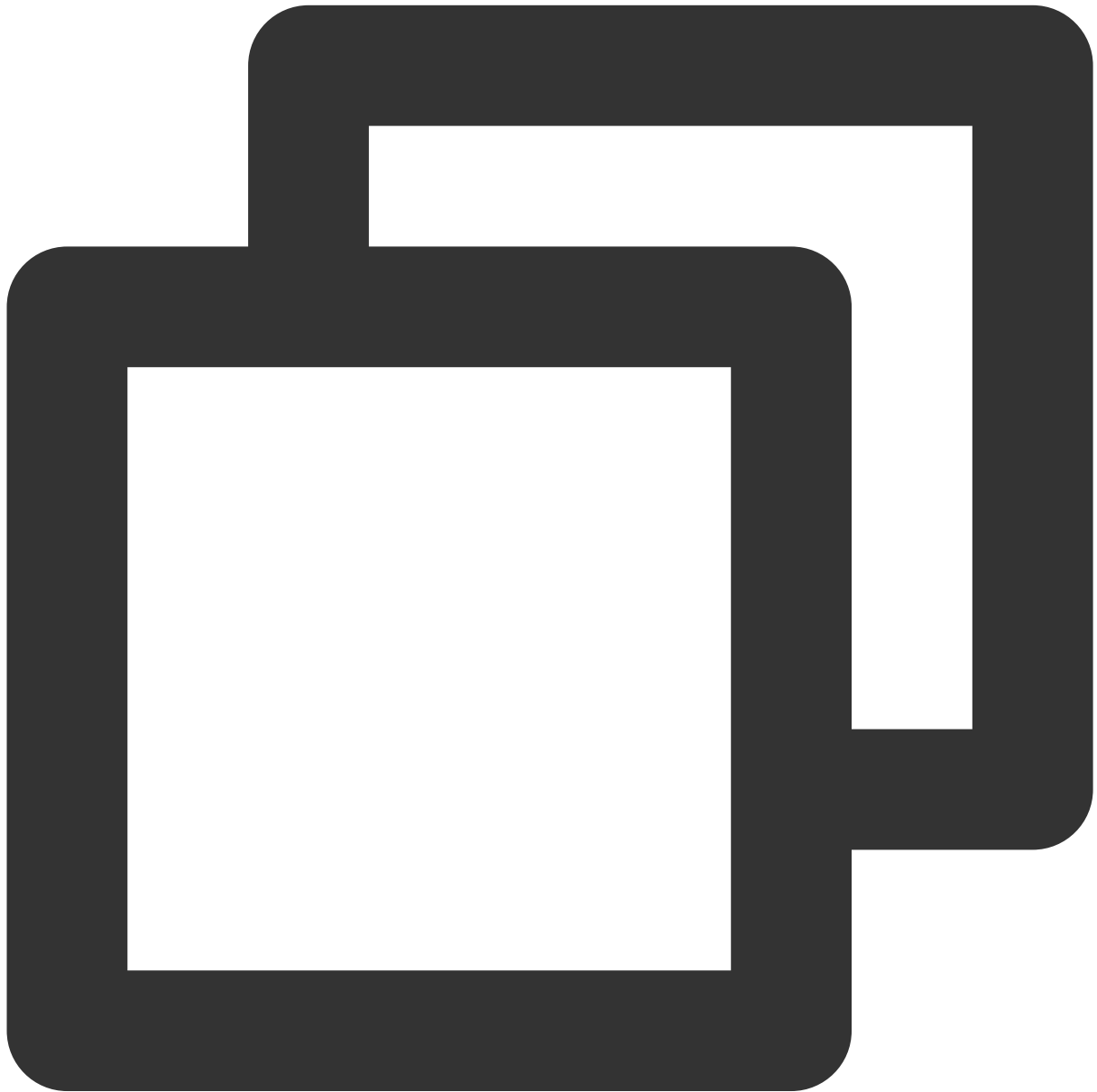


```
const roomEngine = new TUIRoomEngine();
const micList = await roomEngine.getMicDevicesList();
for (i = 0; i < micList.length; i++) {
  var mic = micList[i];
  console.info("mic deviceName: " + mic.deviceName + " deviceId:" + mic.deviceId)
}
```

Returns: *Promise*<[TRTCDDeviceInfo](#)[]> micList

## getSpeakerDevicesList

Get speaker device list



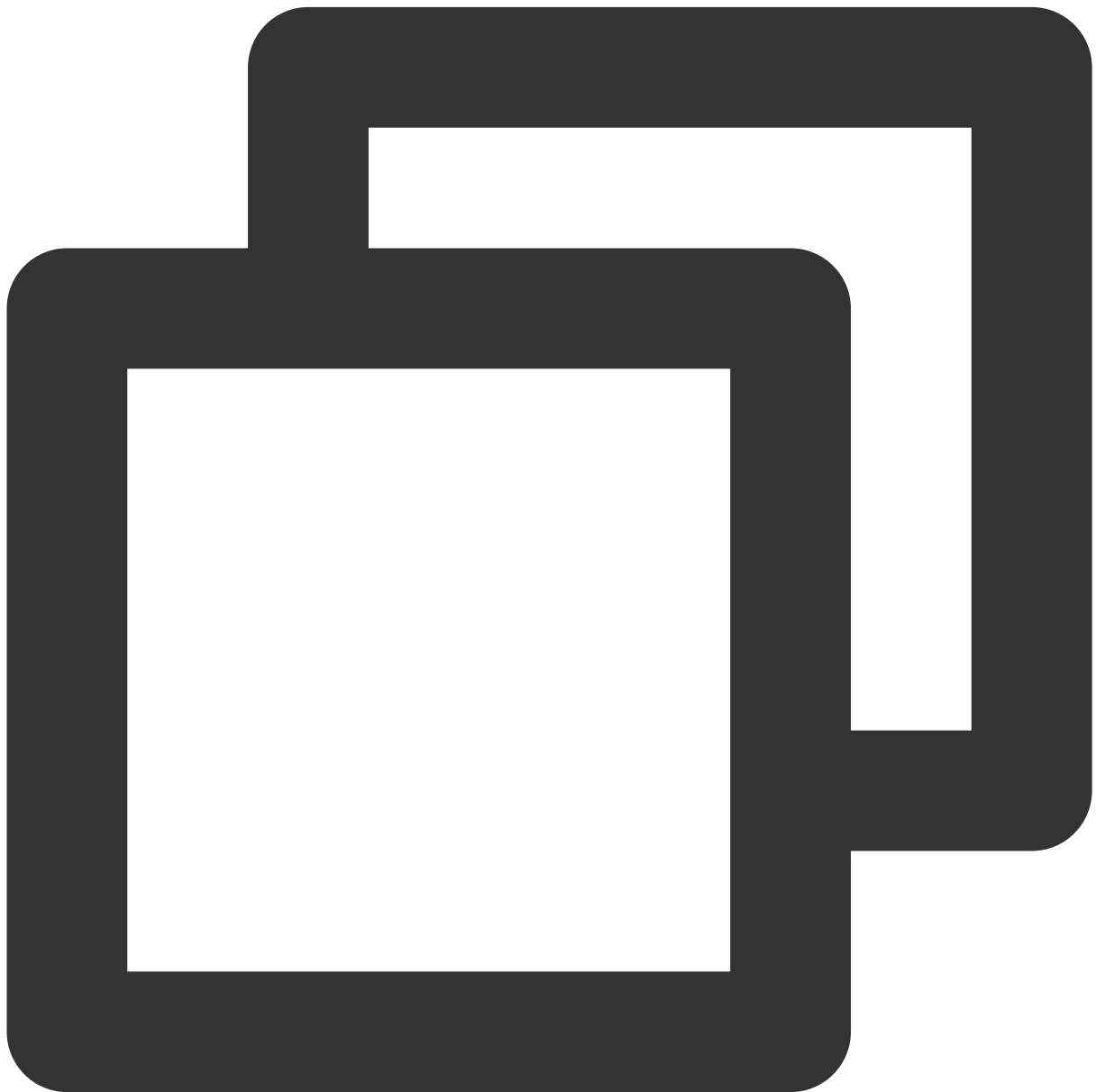
```
const roomEngine = new TUIRoomEngine();
const speakerList = await roomEngine.getSpeakerDevicesList();
for (i = 0; i < speakerList.length; i++) {
```

```
var speaker = speakerList[i];  
console.info("speaker deviceName: " + speaker.deviceName + " deviceId:" + speaker.deviceId);  
}
```

Returns: *Promise*<[TRTCDeviceInfo](#)[]> speakerList

## setCurrentCameraDevice

Set the camera device to be used



```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentCameraDevice({ deviceId: '' });
```

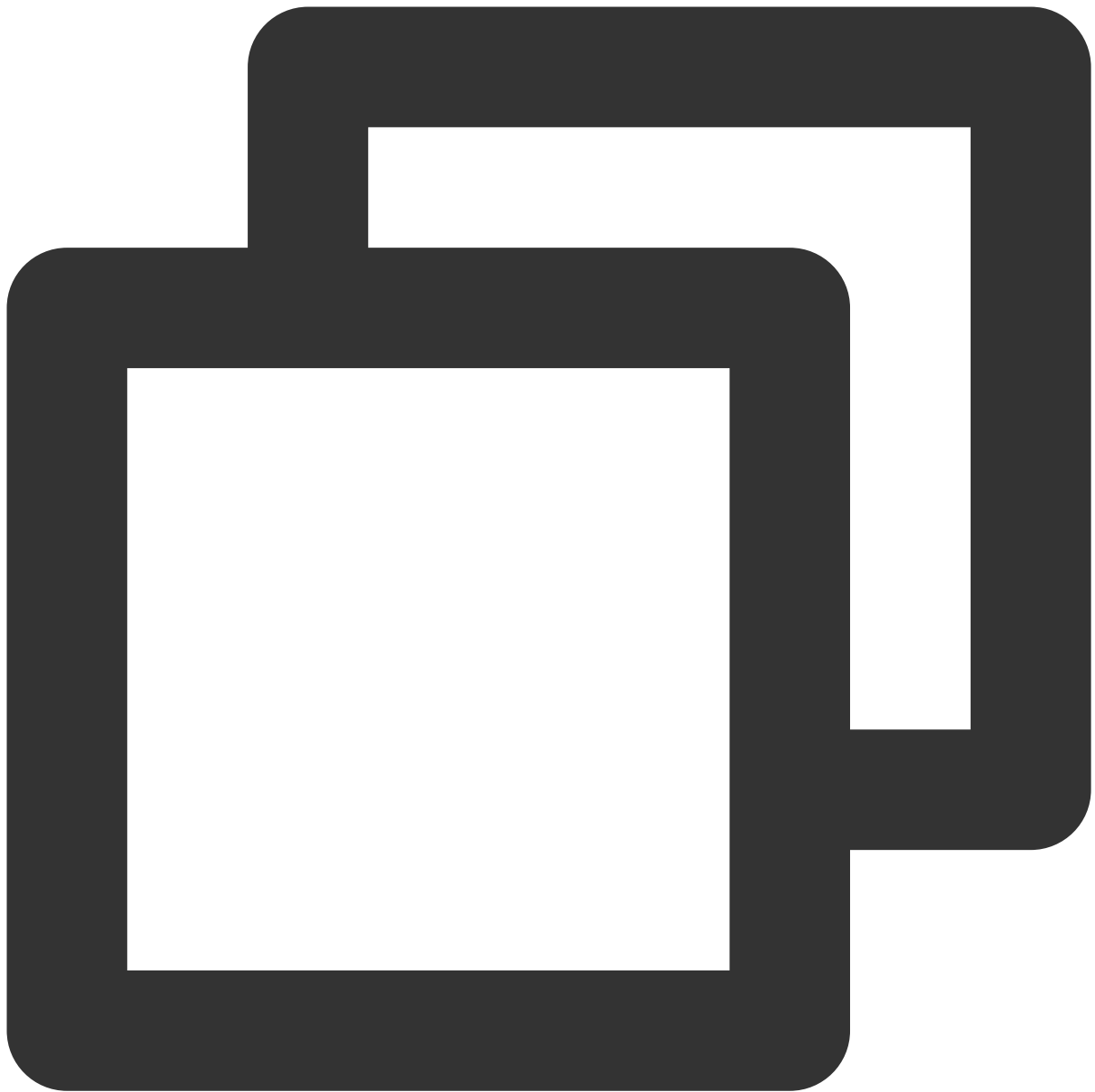
Parameter:

Parameter	Type	Description	Default Value	Meaning
deviceId	string	Required	-	Device ID obtained from getCameraDevicesList

Returns : *void*

## setCurrentMicDevice

Set the mic device to be used



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.setCurrentMicDevice({ deviceId: '' });
```

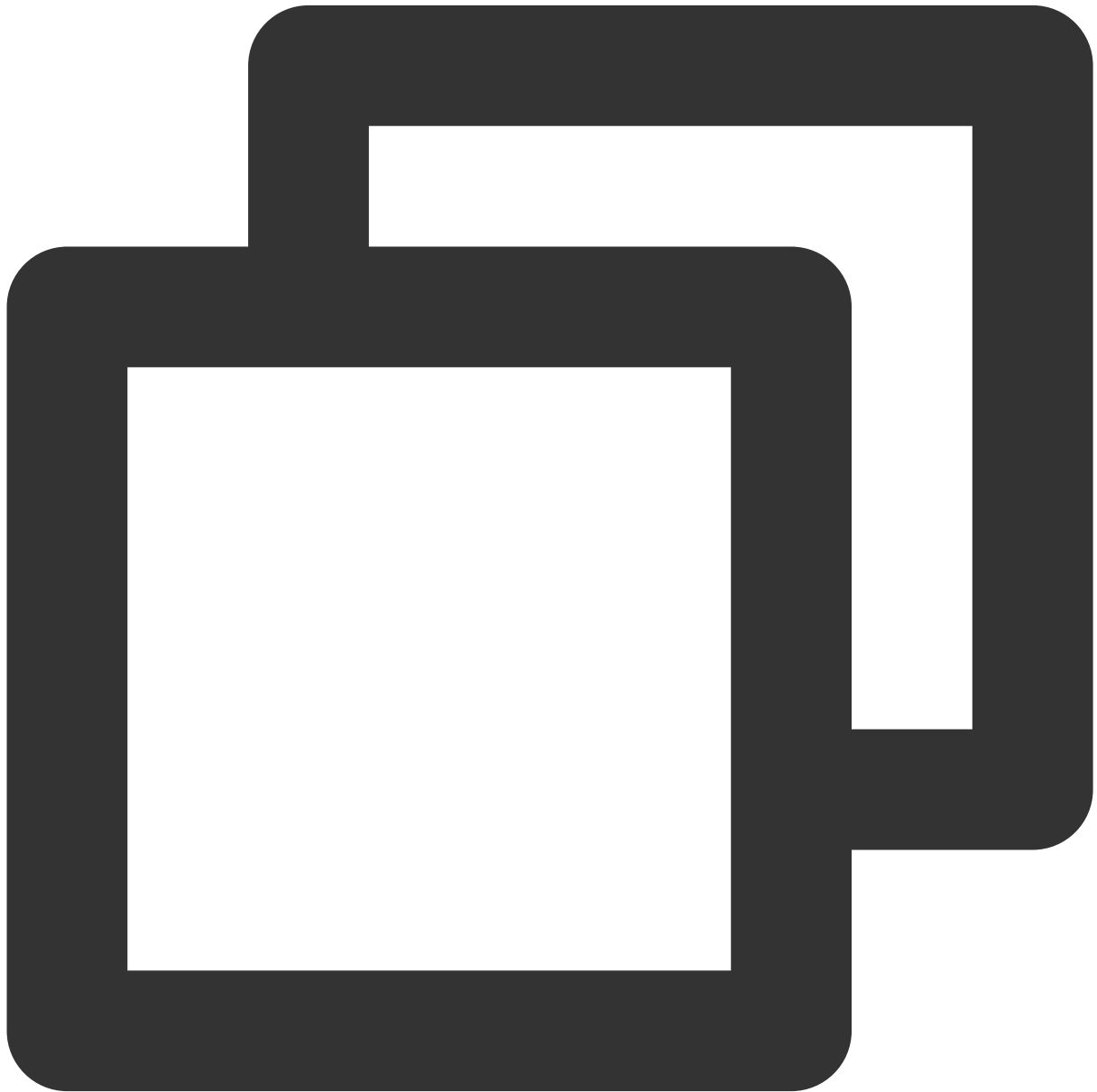
Parameter:

Parameter	Type	Description	Default Value	Meaning
deviceId	string	Required	-	Device ID obtained from getMicDevicesList

Returns : *void*

## setCurrentSpeakerDevice

Set the speaker device to be used



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.setCurrentSpeakerDevice({ deviceId: '' });
```

Parameter:

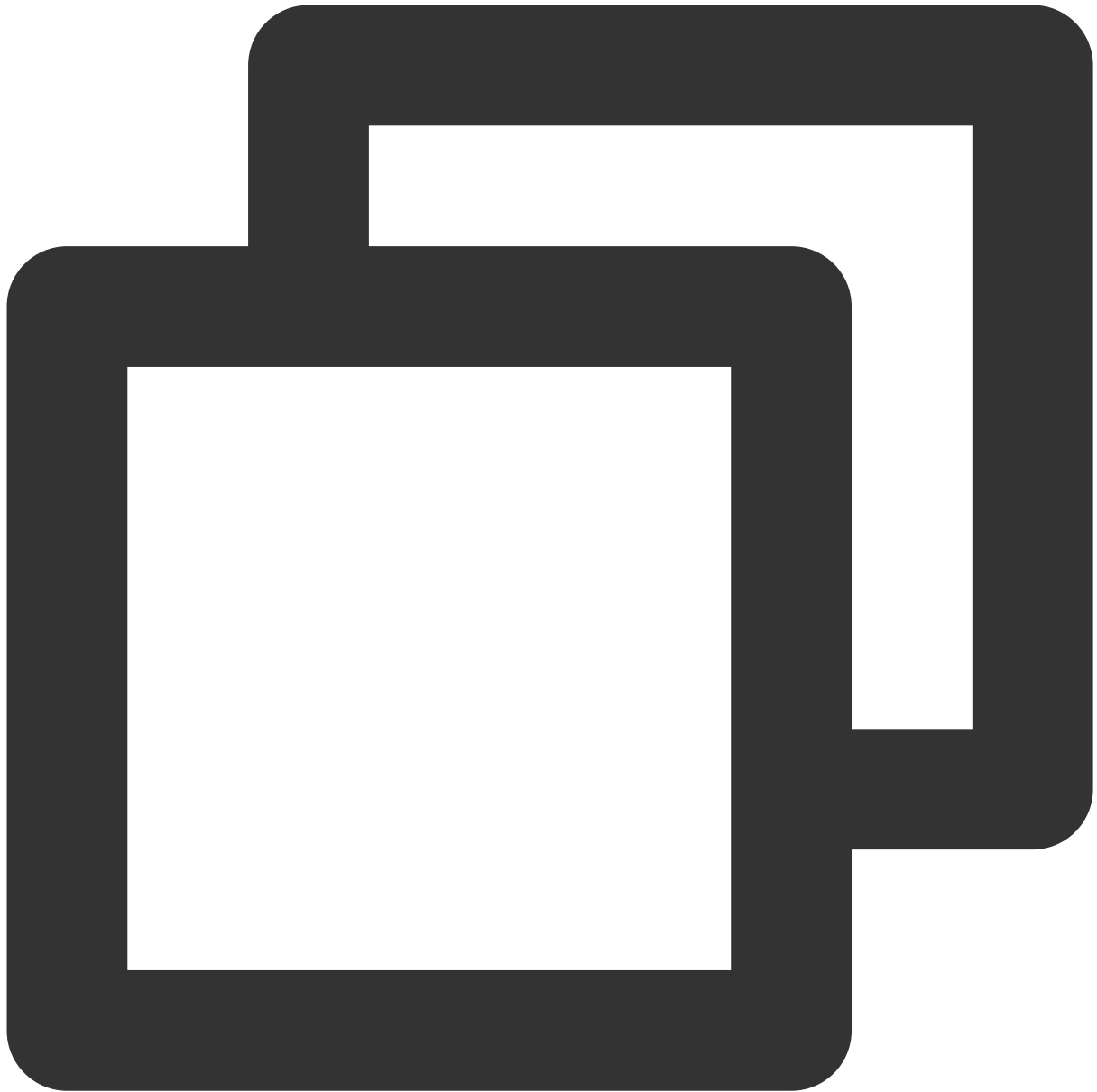
Parameter	Type	Description	Default Value	Meaning
deviceId	string	Required	-	Device ID obtained from getSpeakerDevicesList

Returns : *void*

## getCurrentCameraDevice

Get the currently used camera device





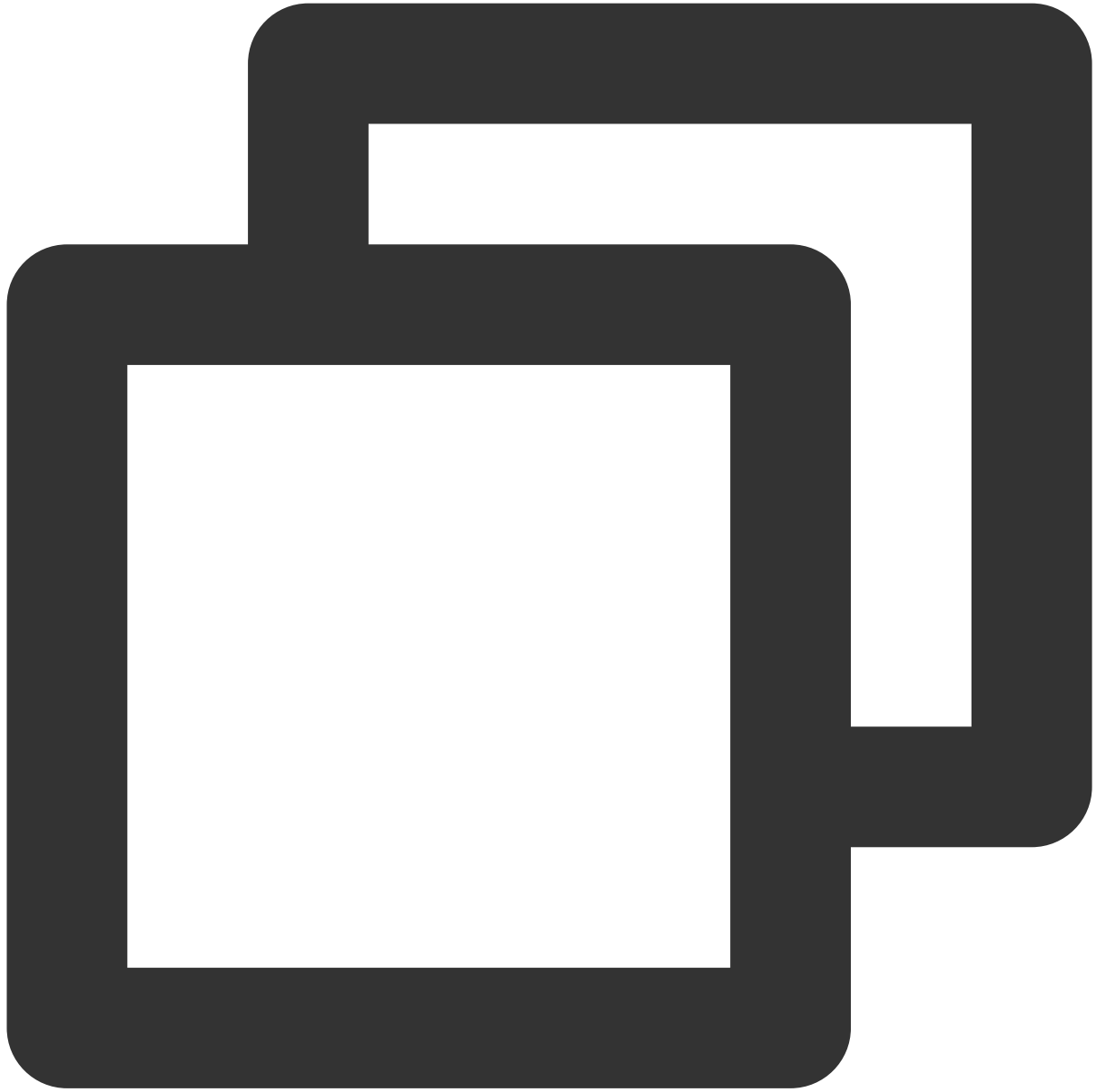
```
const roomEngine = new TUIRoomEngine();  
const currentCameraDevice = roomEngine.getCurrentCameraDevice();
```

Returns : [TRTCDeviceInfo](#)

Device information, can get device ID and device name

## getCurrentMicDevice

Get the currently used mic device



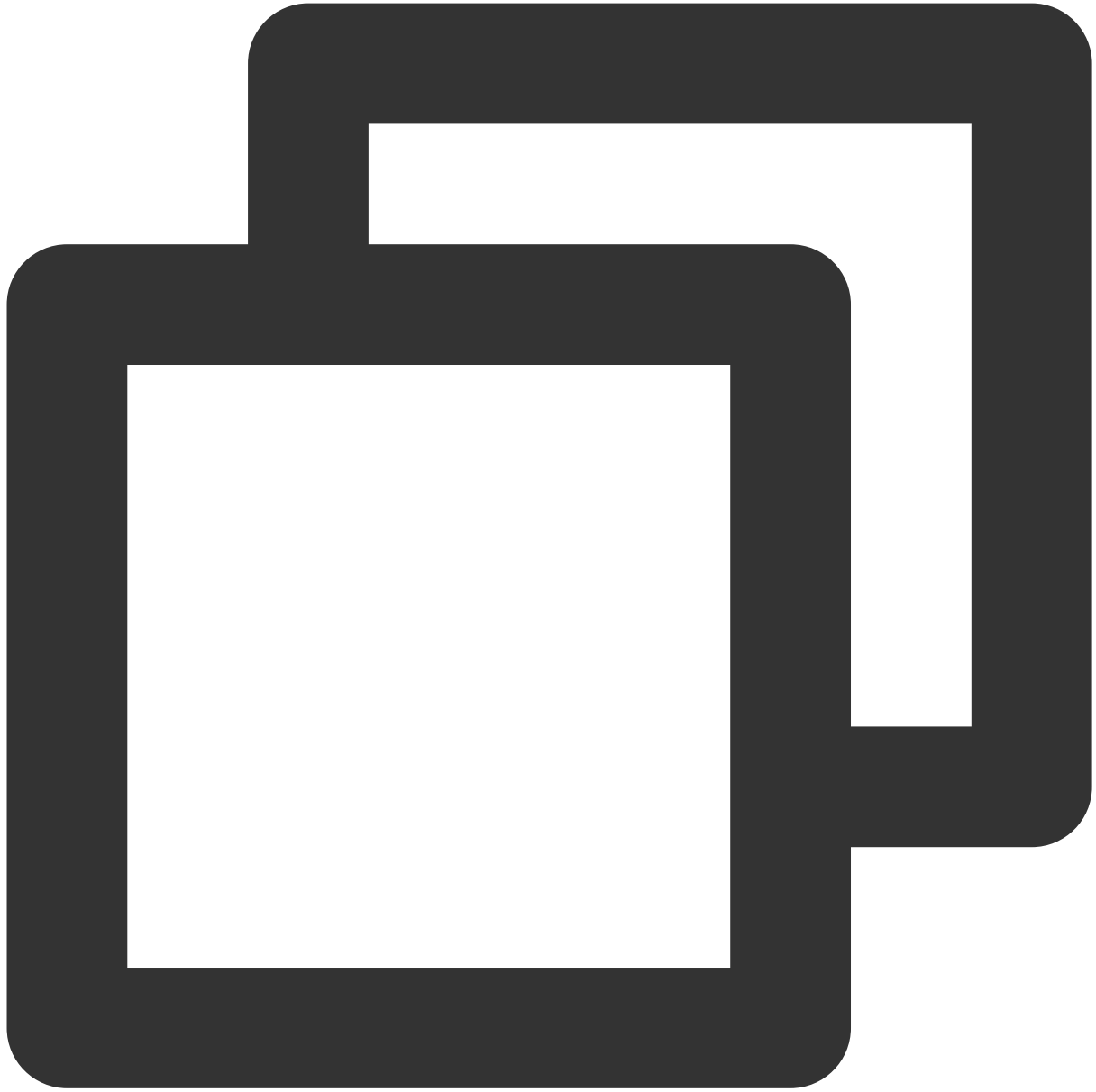
```
const roomEngine = new TUIRoomEngine();  
const currentMicDevice = roomEngine.getCurrentMicDevice();
```

Returns : [TRTCDeviceInfo](#)

Device information, can get device ID and device name

## getCurrentSpeakerDevice

Get the currently used speaker device



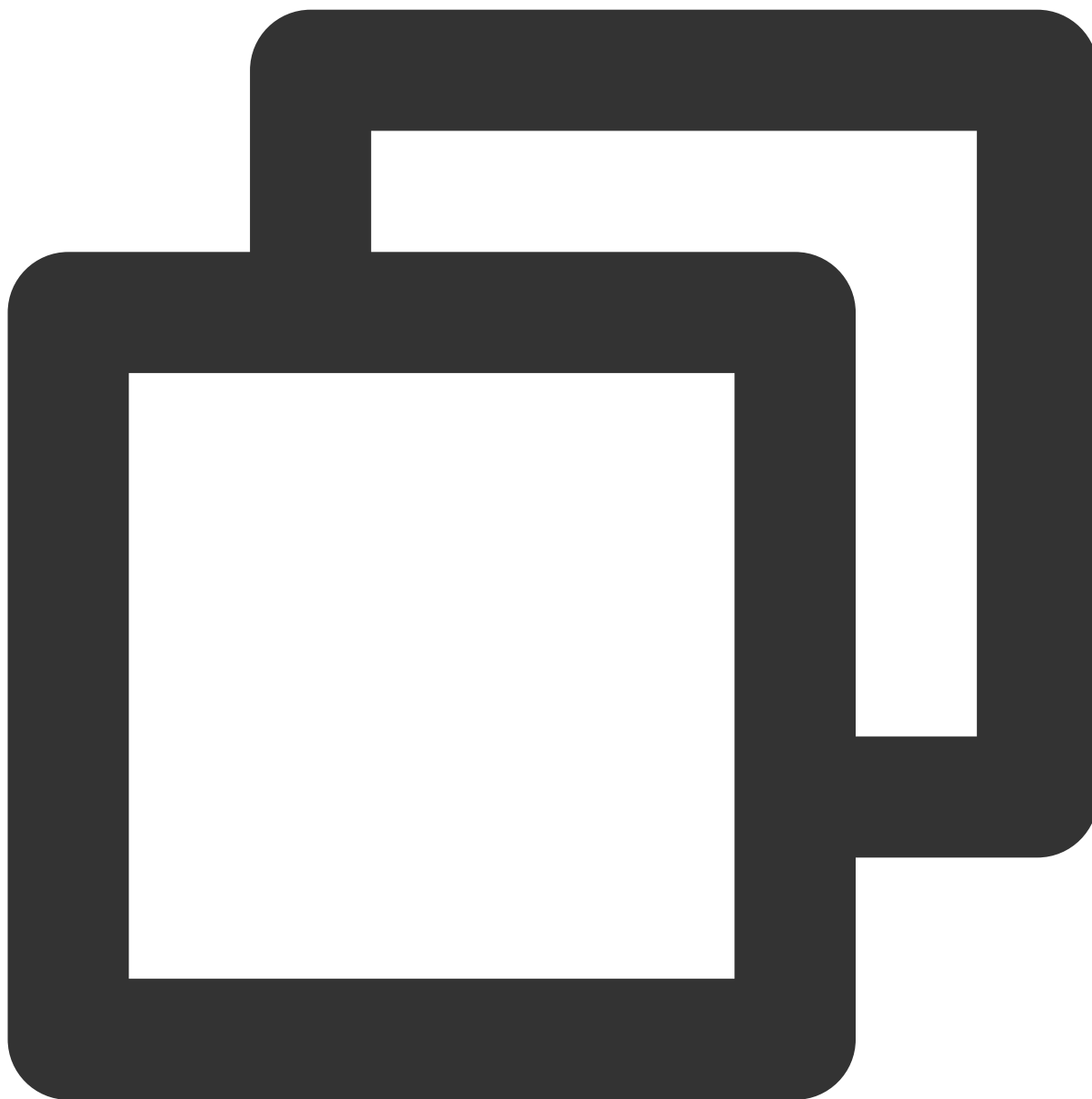
```
const roomEngine = new TUIRoomEngine();  
const currentSpeakerDevice = roomEngine.getCurrentSpeakerDevice();
```

Returns : [TRTCDeviceInfo](#)

Device information, can get device ID and device name

## startCameraDeviceTest

Start camera test



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startCameraDeviceTest({ view: 'test-preview' });
```

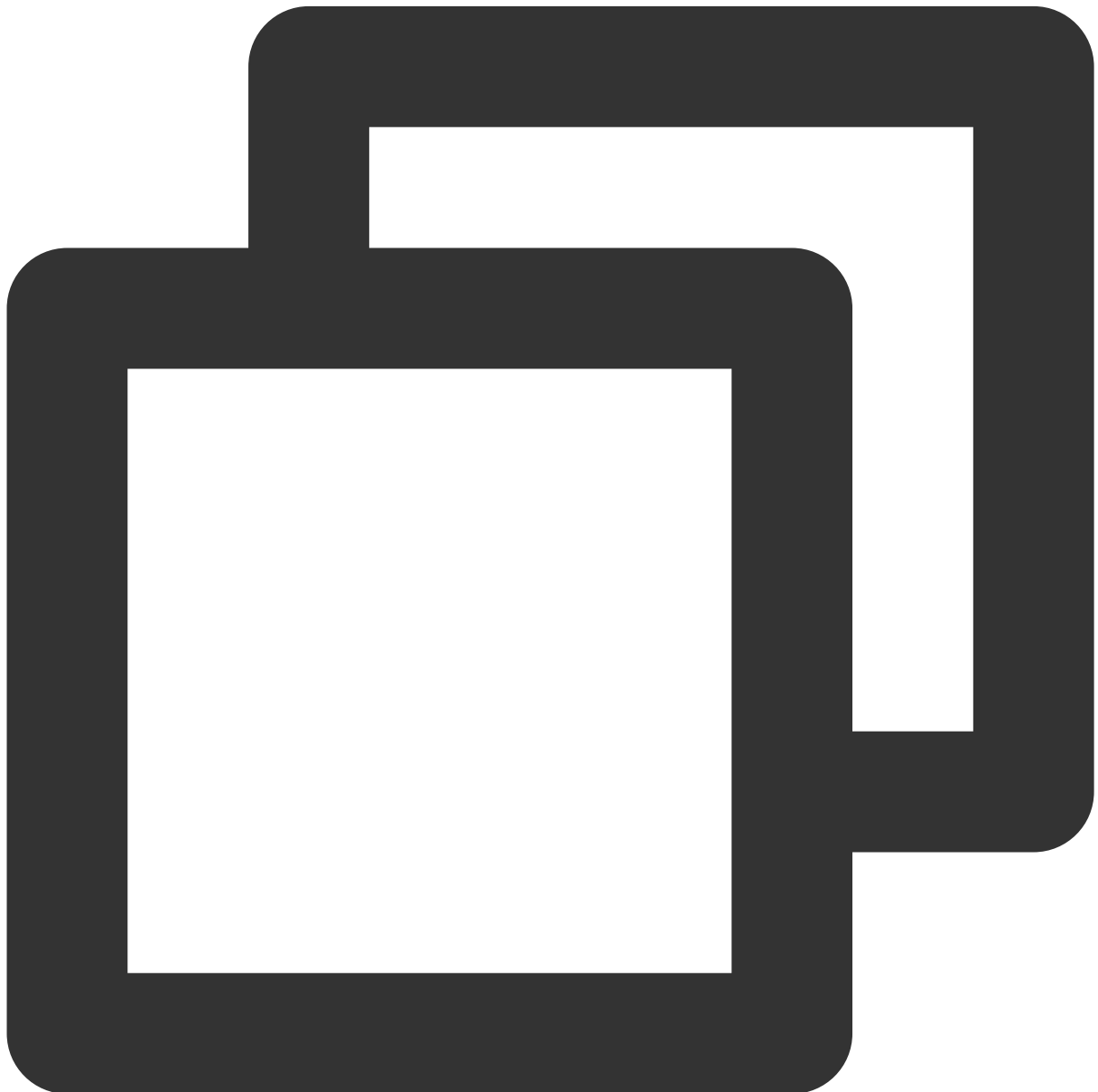
Parameter:

Parameter	Type	Description	Default Value	Meaning
view	string	Required	-	Display the video area of the camera test, the input view is the Id of the div element carrying the preview screen

Returns : *Promise<void>*

## stopCameraDeviceTest

Stop camera test

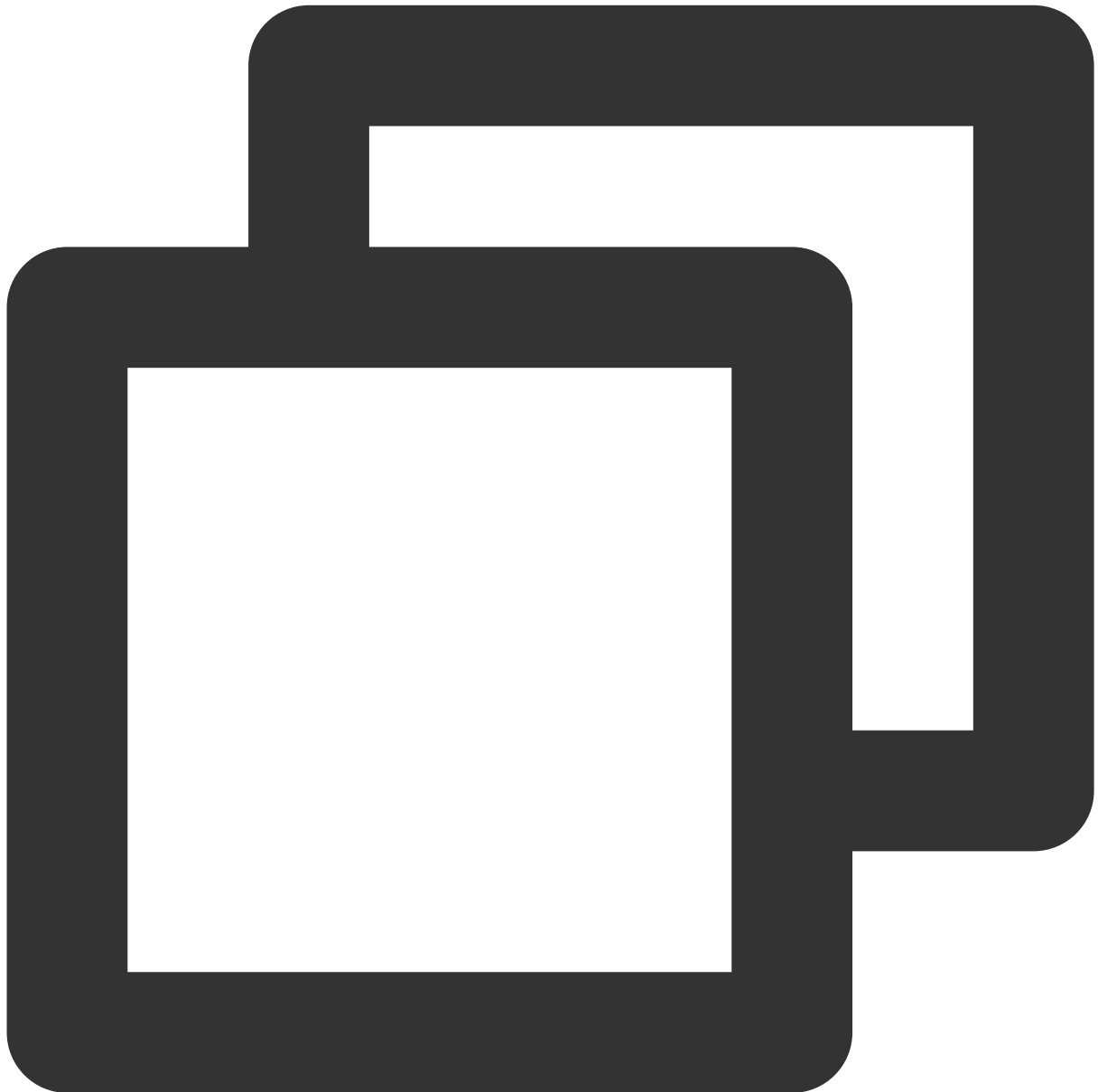


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopCameraDeviceTest();
```

Returns : *Promise<void>*

## on

Listen to roomEngine events



```
const roomEngine = new TUIRoomEngine();  
roomEngine.on(event, func);
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
event	<a href="#">TUIRoomEvents</a>	Required	-	TUIRoomEngine event list
func	Function	Required	-	Event callback function

Returns : *void*

## off

Cancel listening to roomEngine events



```
const roomEngine = new TUIRoomEngine();
roomEngine.off(event, func);
```

Parameter:

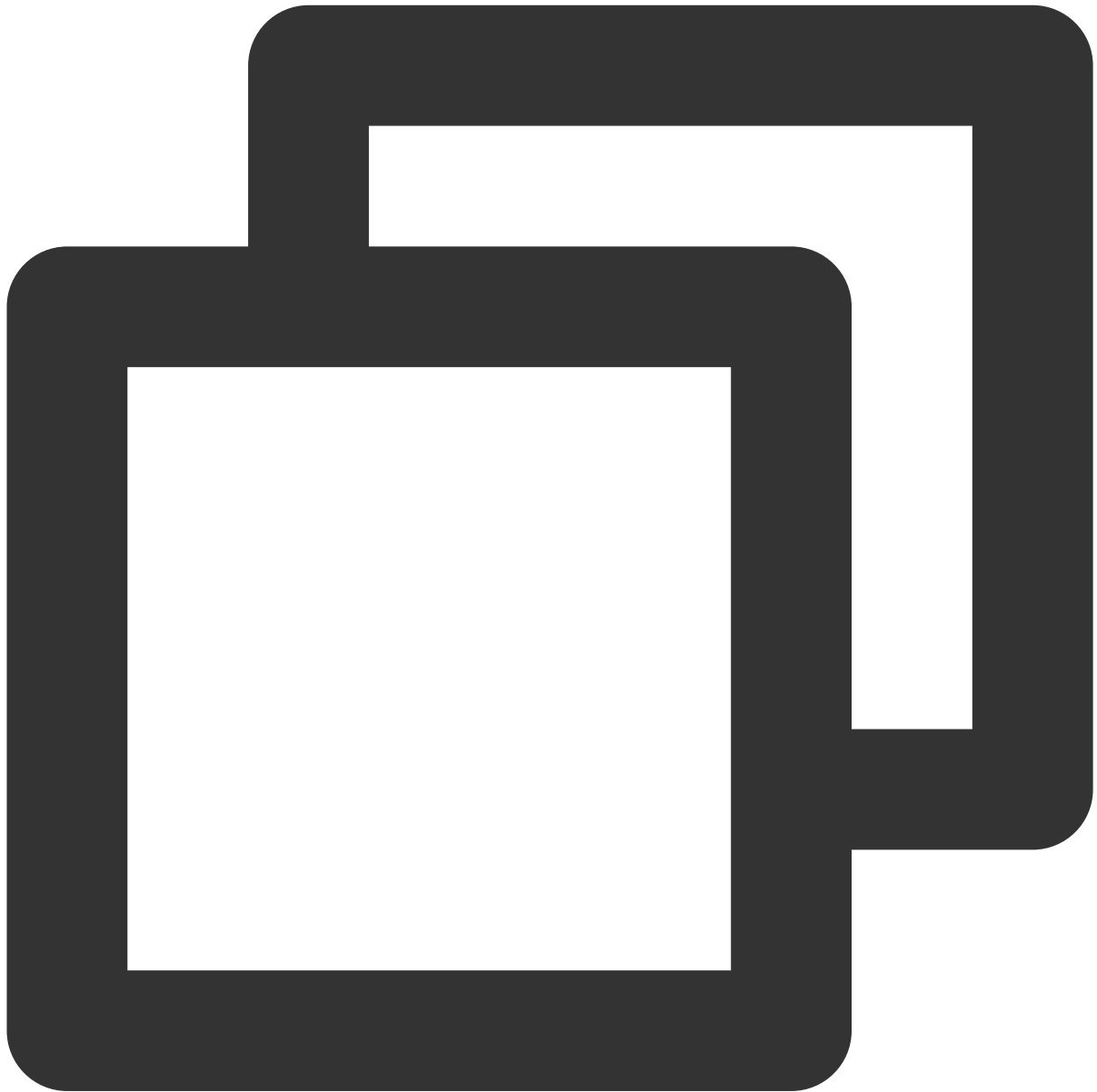
Parameter	Type	Description	Default Value	Meaning
event	<a href="#">TUIRoomEvents</a>	Required	-	TUIRoomEngine event list
func	Function	Required	-	Event callback function



Returns : *void*

## getTRTCCloud

Get trtcCloud instance, for web-side trtcCloud capabilities, please check: [TRTCCloud API documentation](#)

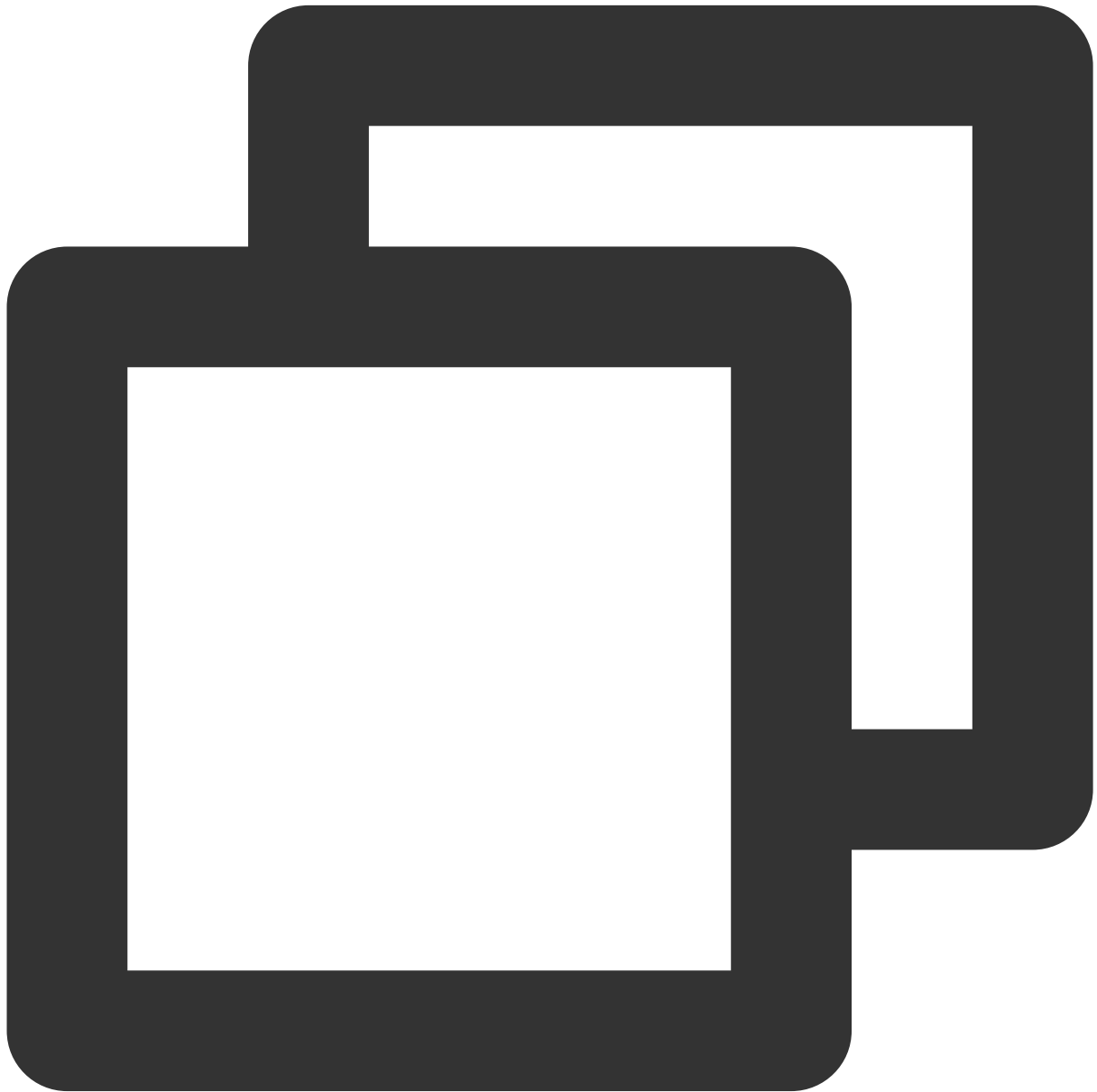


```
const roomEngine = new TUIRoomEngine();  
const trtcCloud = roomEngine.getTRTCCloud();
```

Returns : [TRTCCloud](#)

## getTIM

Get tim instance, for web-side tim capabilities, please check: [IM API documentation](#)



```
const roomEngine = new TUIRoomEngine();  
const trtcCloud = roomEngine.getTIM();
```

Returns : *TIM*

# TUIRoomEvents

Last updated : 2023-11-14 17:25:21

## TUIRoomEvent API Introduction

TUIRoomEvent API is the Event Interface for Audio/Video call Component.

### Event list

EVENT	Description
<a href="#">TUIRoomEvents.onError</a>	Error Event
<a href="#">TUIRoomEvents.onKickedOutOfRoom</a>	Kicked out of room event
<a href="#">TUIRoomEvents.onKickedOffLine</a>	Current user Kicked off line event
<a href="#">TUIRoomEvents.onUserSigExpired</a>	UserSig Expiration Event
<a href="#">TUIRoomEvents.onRoomDismissed</a>	Host Destroy Room Event
<a href="#">TUIRoomEvents.onRoomNameChanged</a>	Room Name Change Event
<a href="#">TUIRoomEvents.onRoomSpeechModeChanged</a>	Room Speech Mode Change Event
<a href="#">TUIRoomEvents.onAllUserCameraDisableChanged</a>	All members Camera Usage Permission Change Event
<a href="#">TUIRoomEvents.onAllUserMicrophoneDisableChanged</a>	All members Mic Usage Permission Change Event
<a href="#">TUIRoomEvents.onSendMessageForAllUserDisableChanged</a>	All members Send Message Status Change Event
<a href="#">TUIRoomEvents.onRoomMaxSeatCountChanged</a>	Room Maximum Seat Count Change Event
<a href="#">TUIRoomEvents.onRemoteUserEnterRoom</a>	Remote User Entered Room Event
<a href="#">TUIRoomEvents.onRemoteUserLeaveRoom</a>	Remote User Leave Room Event
<a href="#">TUIRoomEvents.onUserRoleChanged</a>	User Role Change Event
<a href="#">TUIRoomEvents.onUserVideoStateChanged</a>	User Video Status Change Event

<a href="#">TUIRoomEvents.onUserAudioStateChanged</a>	User Audio Status Change Event
<a href="#">TUIRoomEvents.onSendMessageForUserDisableChanged</a>	User Send Message Status Event
<a href="#">TUIRoomEvents.onUserVoiceVolumeChanged</a>	User Volume Change Event
<a href="#">TUIRoomEvents.onUserNetworkQualityChanged</a>	User Network Quality Change Event
<a href="#">TUIRoomEvents.onSeatListChanged</a>	Seat List Change Event
<a href="#">TUIRoomEvents.onKickedOffSeat</a>	User Kicked off Seat Event
<a href="#">TUIRoomEvents.onRequestReceived</a>	Request Received Event
<a href="#">TUIRoomEvents.onRequestCancelled</a>	Request Cancelled Event
<a href="#">TUIRoomEvents.onReceiveTextMessage</a>	Receive Text Message Event
<a href="#">TUIRoomEvents.onReceiveCustomMessage</a>	Receive Custom Message Event
<a href="#">TUIRoomEvents.onDeviceChange</a>	Device Change Event
<a href="#">TUIRoomEvents.onUserScreenCaptureStopped</a>	Screen Sharing Stopped Event When the user uses the built-in browser stop sharing button to end screen sharing, the user will receive the 'onUserScreenCaptureStopped' event to modify the screen sharing status.

## onError

Error Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onError, (error) => {
  console.log('TUIRoomError error', error);
})
```

The parameters are shown in the table below:

Parameter	Type	Meaning
code	number	Error Code

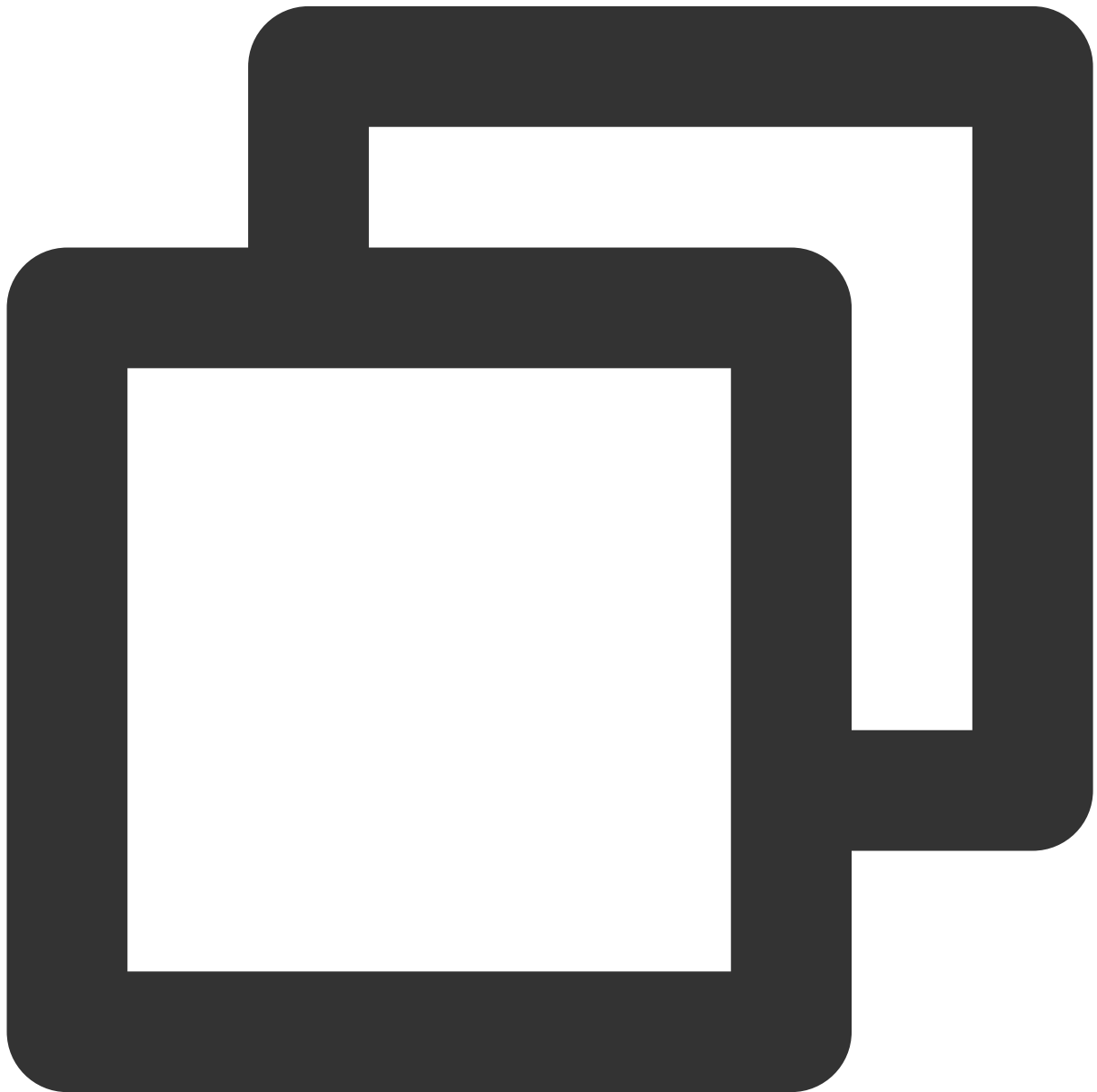
message

string

Error Information

## onKickedOutOfRoom

Kicked out of room event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOutOfRoom, ({ roomId, message }) => {
  console.log('roomEngine.onKickedOutOfRoom', roomId, message);
});
```

```
});
```

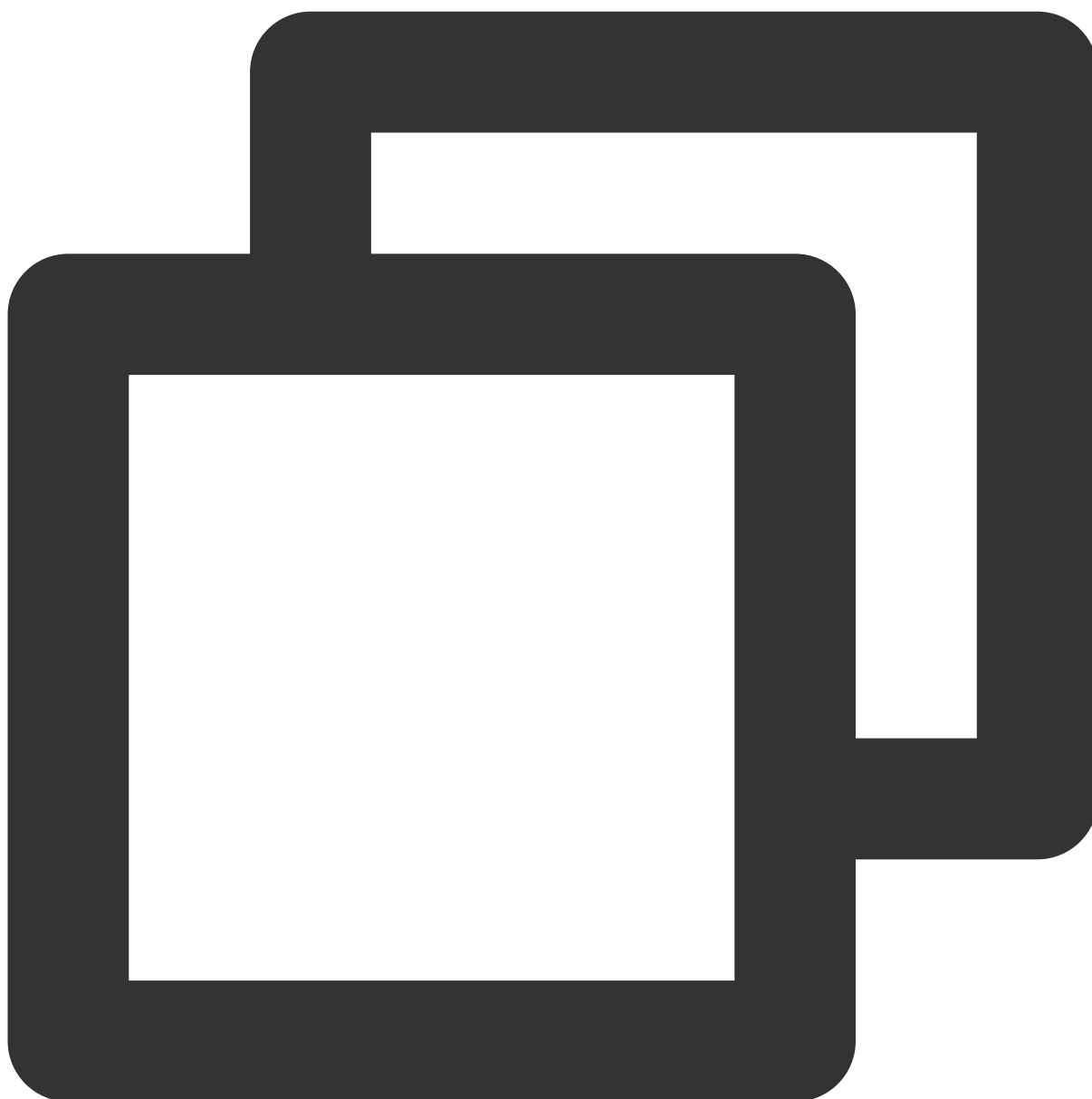
The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
message	string	Kicked out of room information

## onKickedOffline

Current user Kicked off line event





```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffLine, ({ message }) => {
  console.log('roomEngine.onKickedOffLine', message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

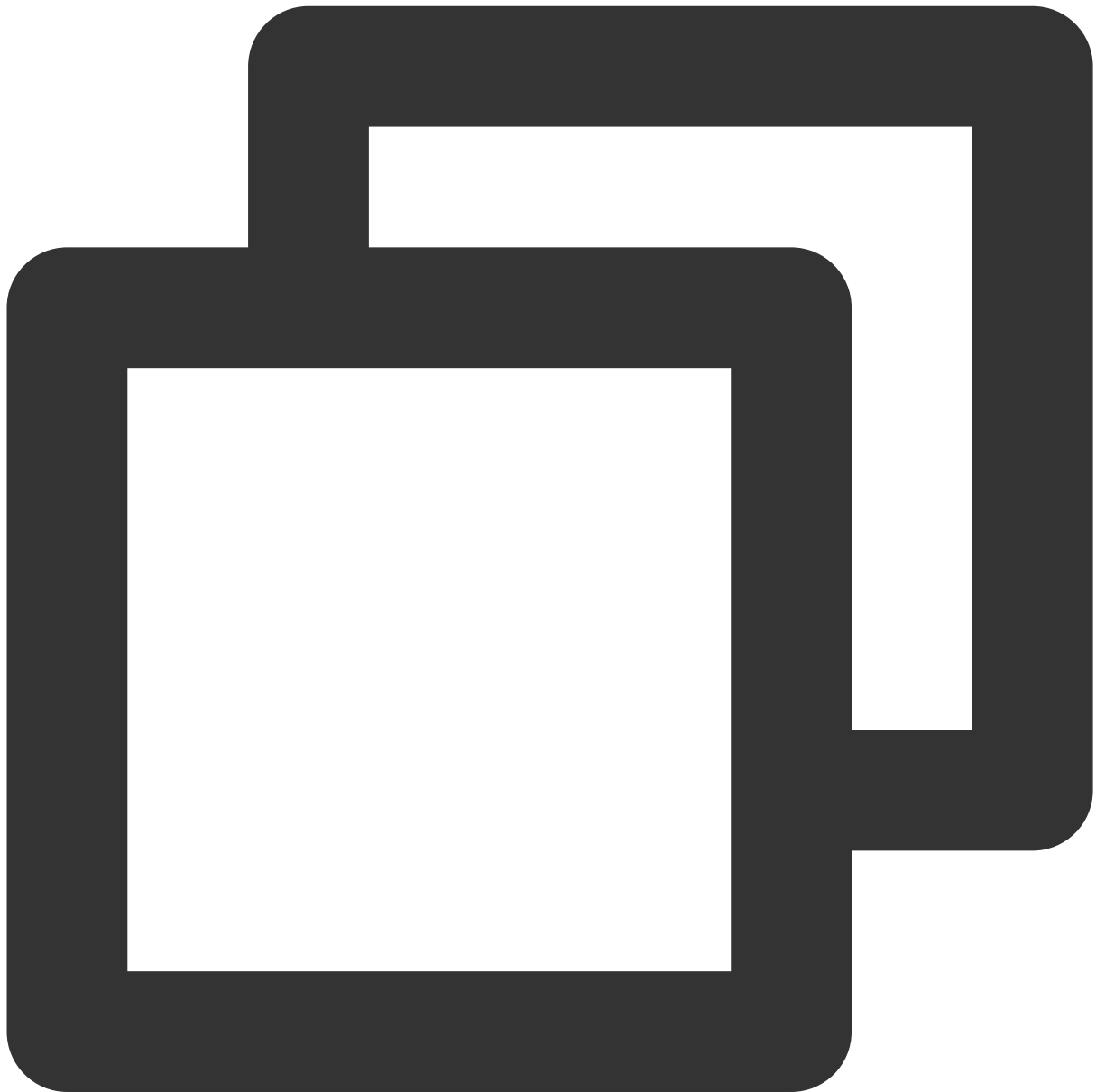
message

string

User logged in on other end information

## onUserSigExpired

UserSig Expiration Event

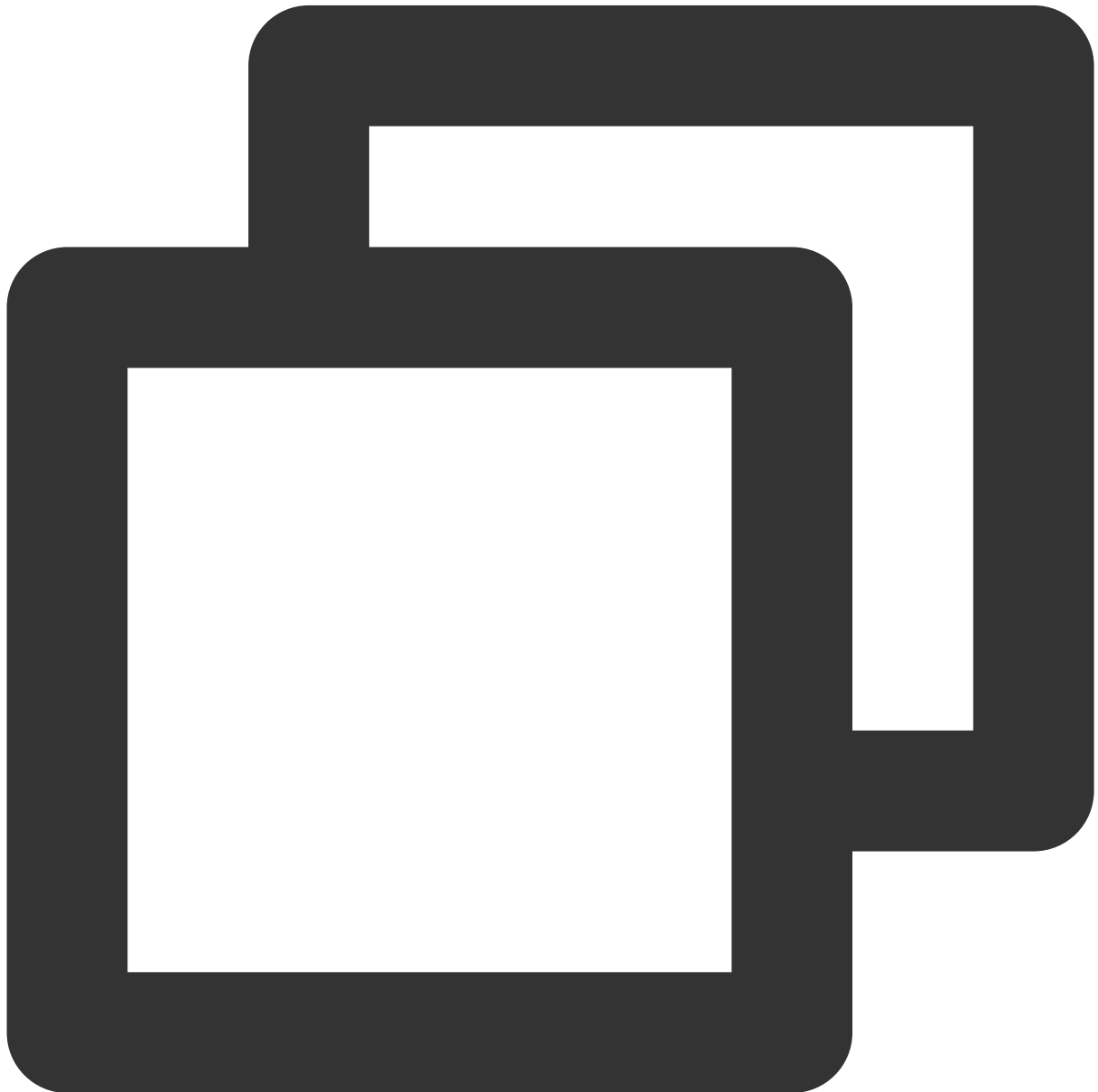


```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserSigExpired, () => {
  console.log('roomEngine.onUserSigExpired');
});
```

```
});
```

## onRoomDismissed

Host Destroy Room Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomDismissed, ({ roomId }) => {
  console.log('roomEngine.onRoomDismissed', roomId);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

## onRoomNameChanged

Room ID Modification Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomNameChanged, ({ roomId, roomName }) => {
  console.log('roomEngine.onRoomNameChanged', roomId, roomName);
});
```

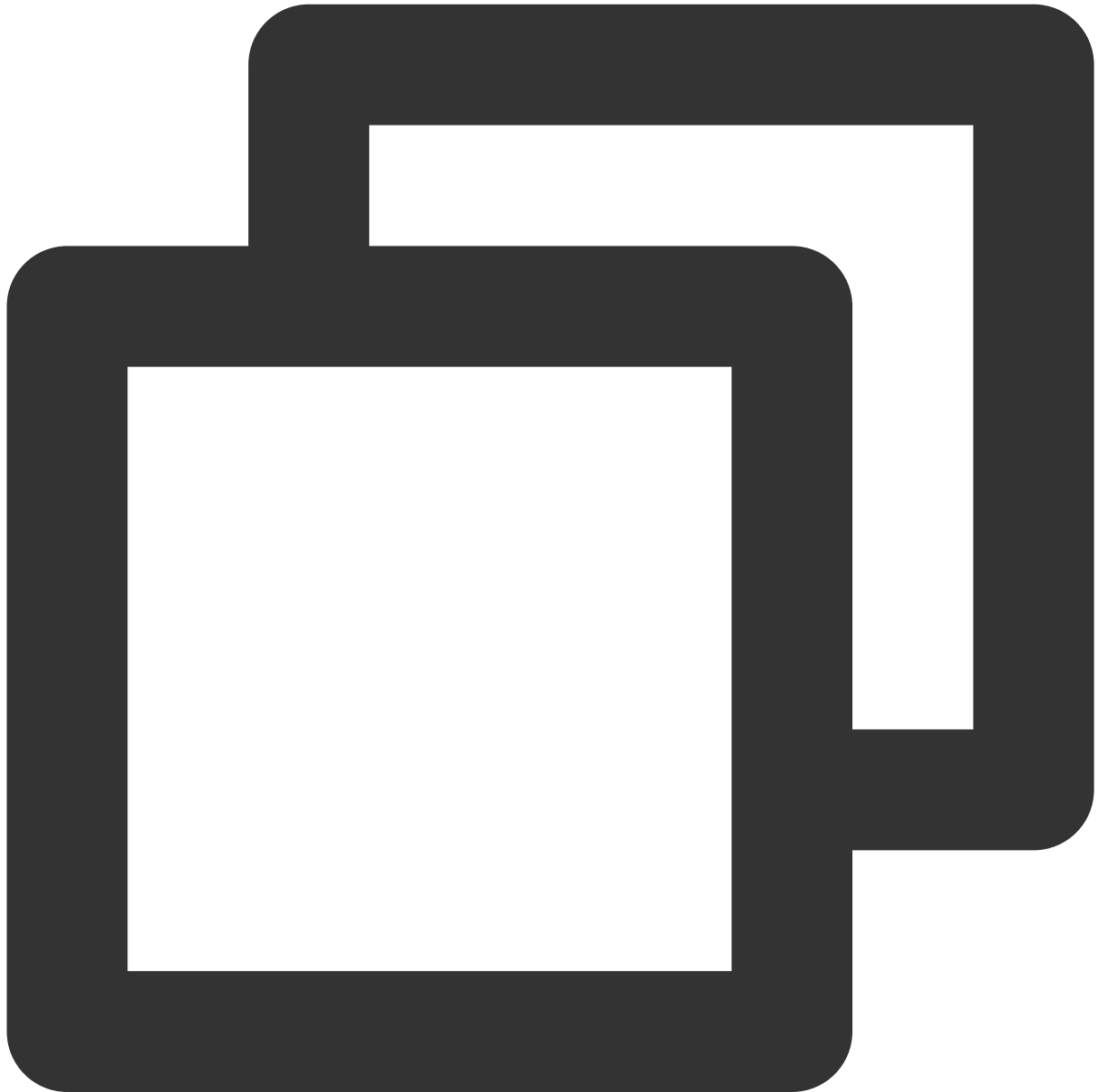
The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

roomName	string	Room Name
----------	--------	-----------

## onRoomSpeechModeChanged

Room Name Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSpeechModeChanged, ({ roomId, speechMode }) => {
  console.log('roomEngine.onRoomSpeechModeChanged', roomId, speechMode);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
speechMode	<a href="#">TUISpeechMode</a>	Speech Mode

## onAllUserCameraDisableChanged

All members Camera Usage Permission Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserCameraDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserCameraDisableChanged', isDisable);
});
```

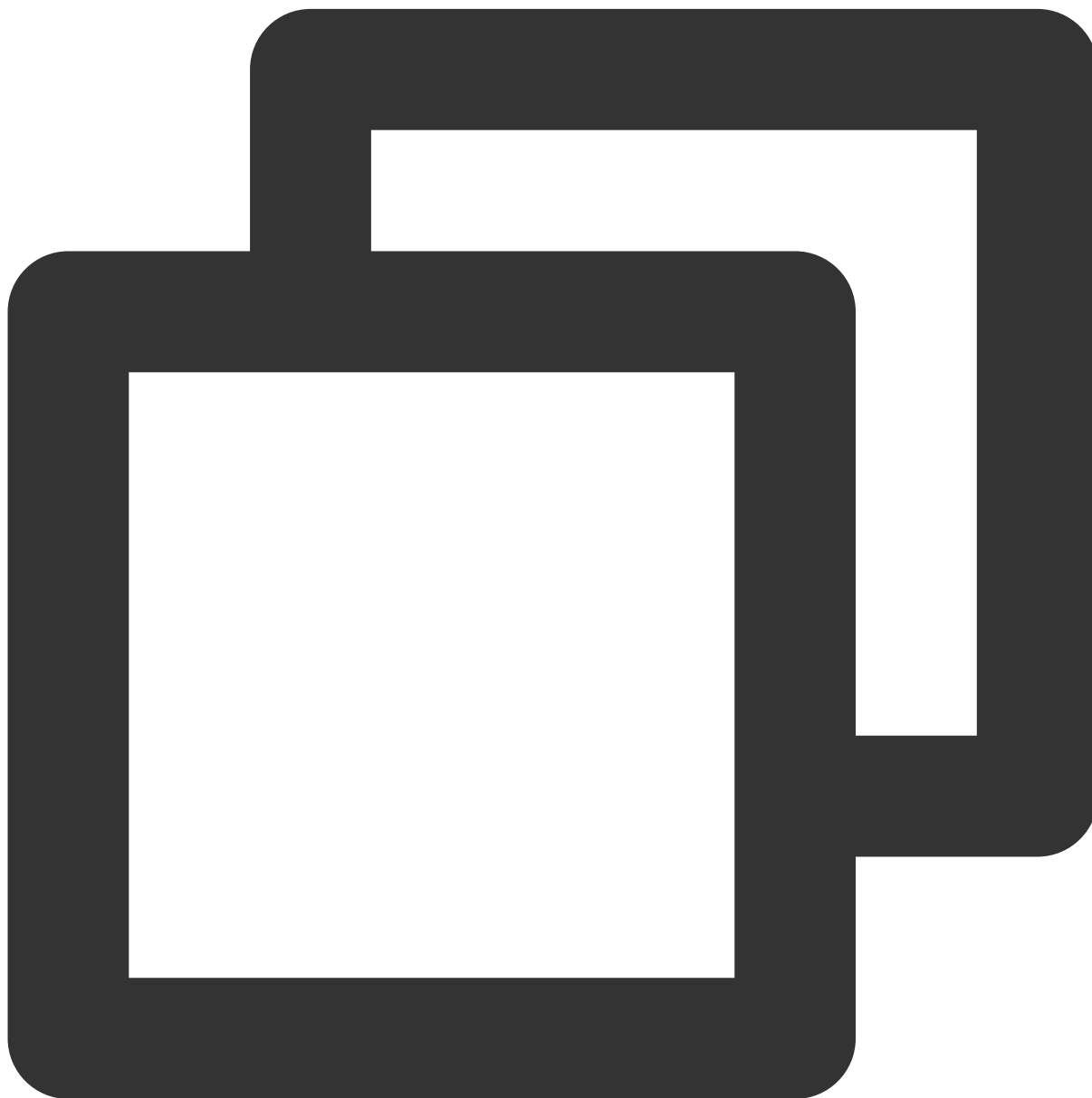
The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Camera Usage



## onAllUserMicrophoneDisableChanged

All members Mic Usage Permission Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserMicrophoneDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserMicrophoneDisableChanged', isDisable);
});
```

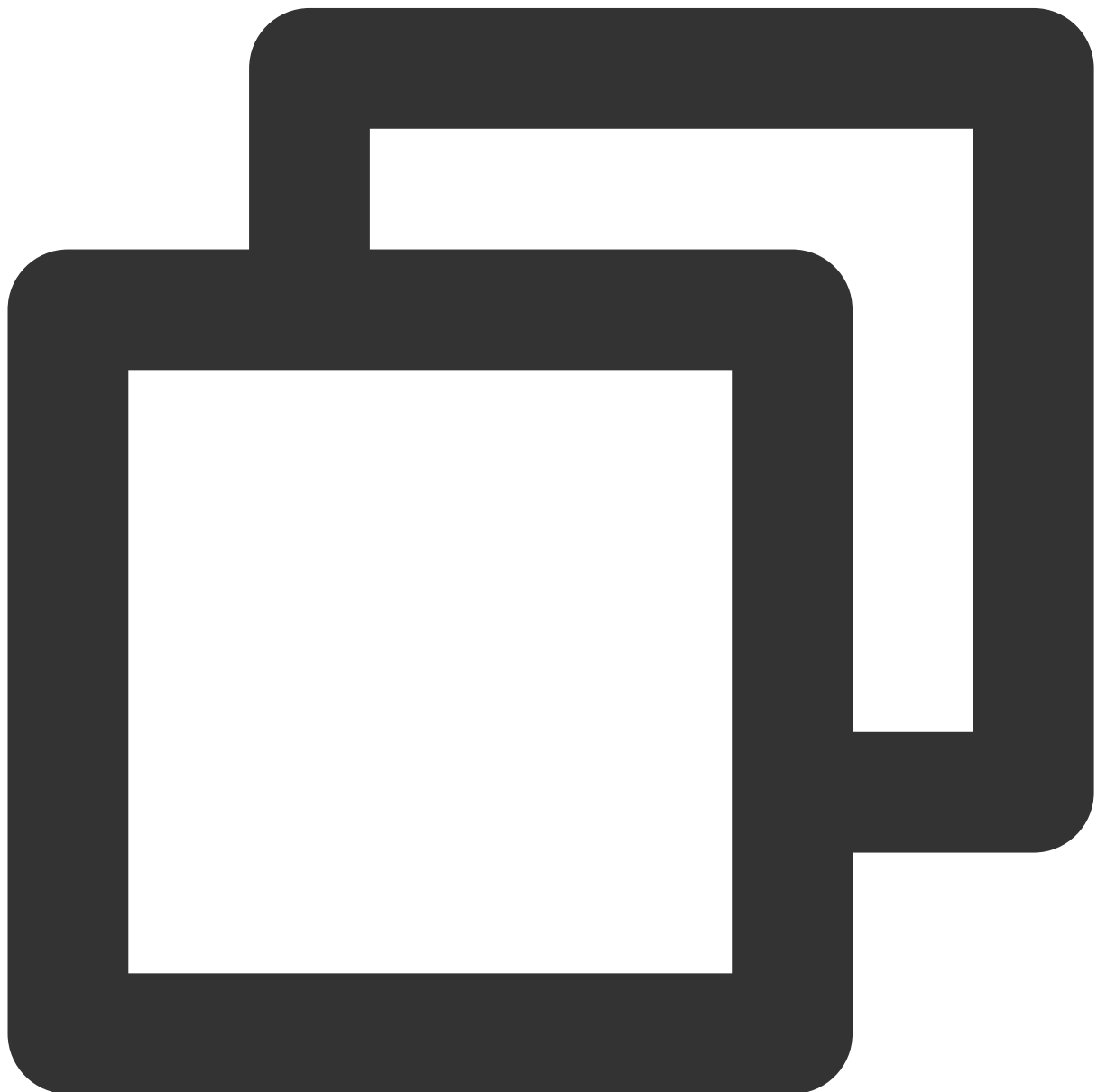
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
isDisable	boolean	Allow Mic Usage

## onSendMessageForAllUserDisableChanged

All members Send Message Permission Change Event



```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable })  
    console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);  
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Sending Text Message

## onRoomMaxSeatCountChanged

Room Maximum Seat Count Change Event



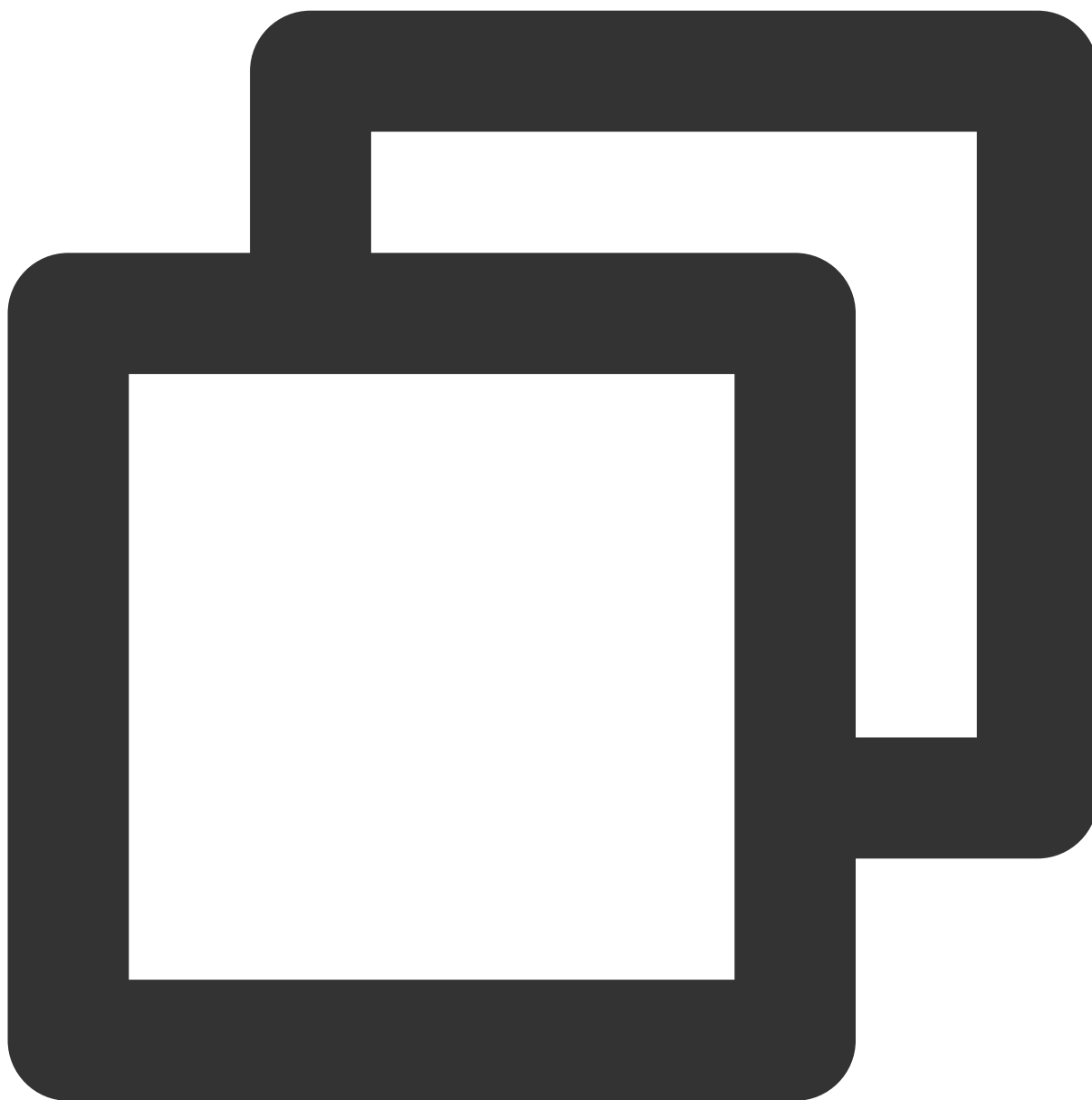
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomMaxSeatCountChanged, ({ maxSeatNumber }) => {
  console.log('roomEngine.onRoomMaxSeatCountChanged', maxSeatNumber);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
maxSeatNumber	number	Maximum Seat Count

## onRemoteUserEnterRoom

Remote User Entered Room Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserEnterRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserEnterRoom', roomId, userInfo);
});
```

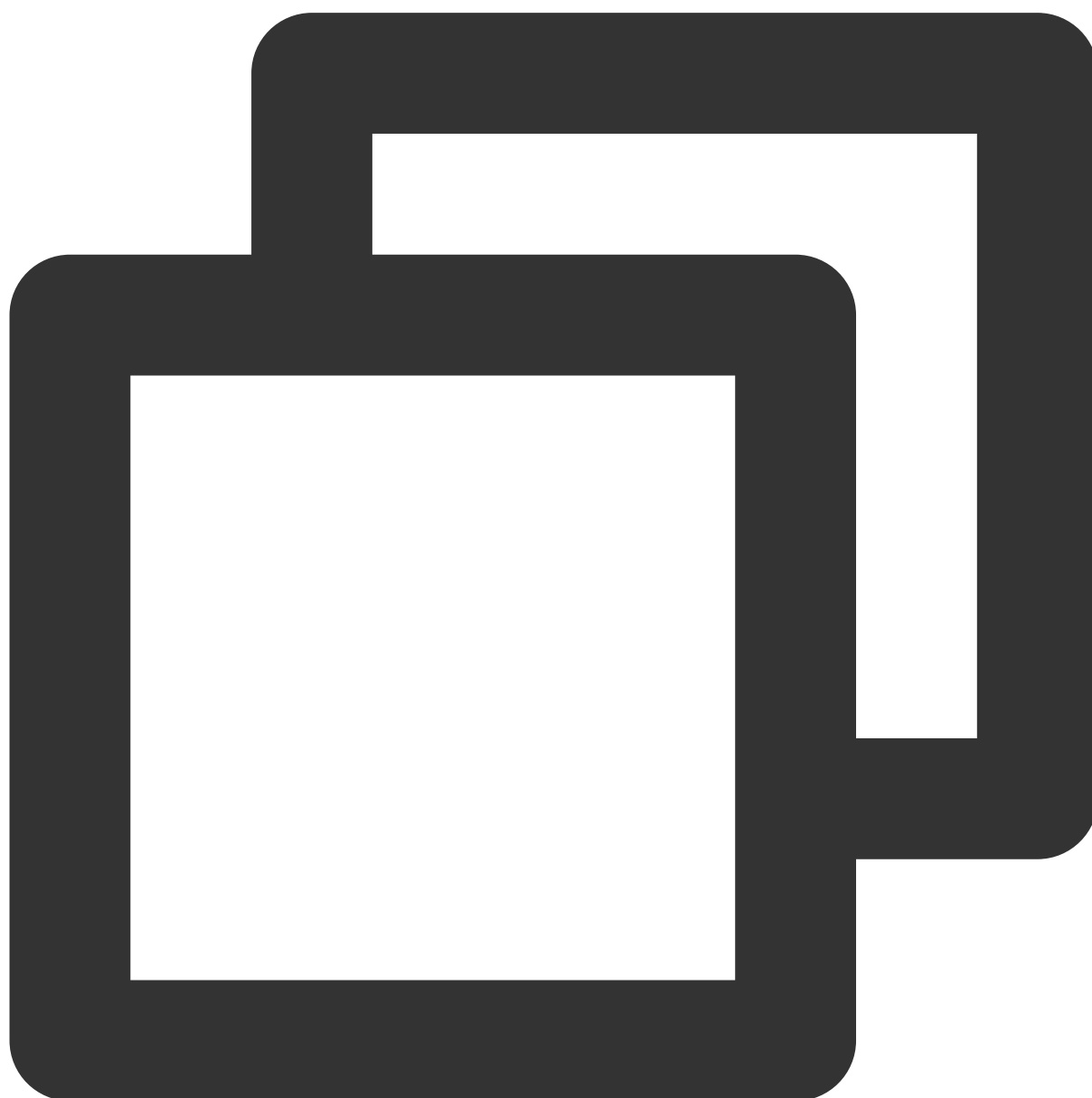
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
roomId	string	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User Information

## onRemoteUserLeaveRoom

Remote User Leave Room Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserLeaveRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserLeaveRoom', roomId, userInfo);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User Information

## onUserRoleChanged

User Role Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserRoleChanged, ({ userId, userRole }) => {
  console.log('roomEngine.onUserRoleChanged', userId, userRole);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id



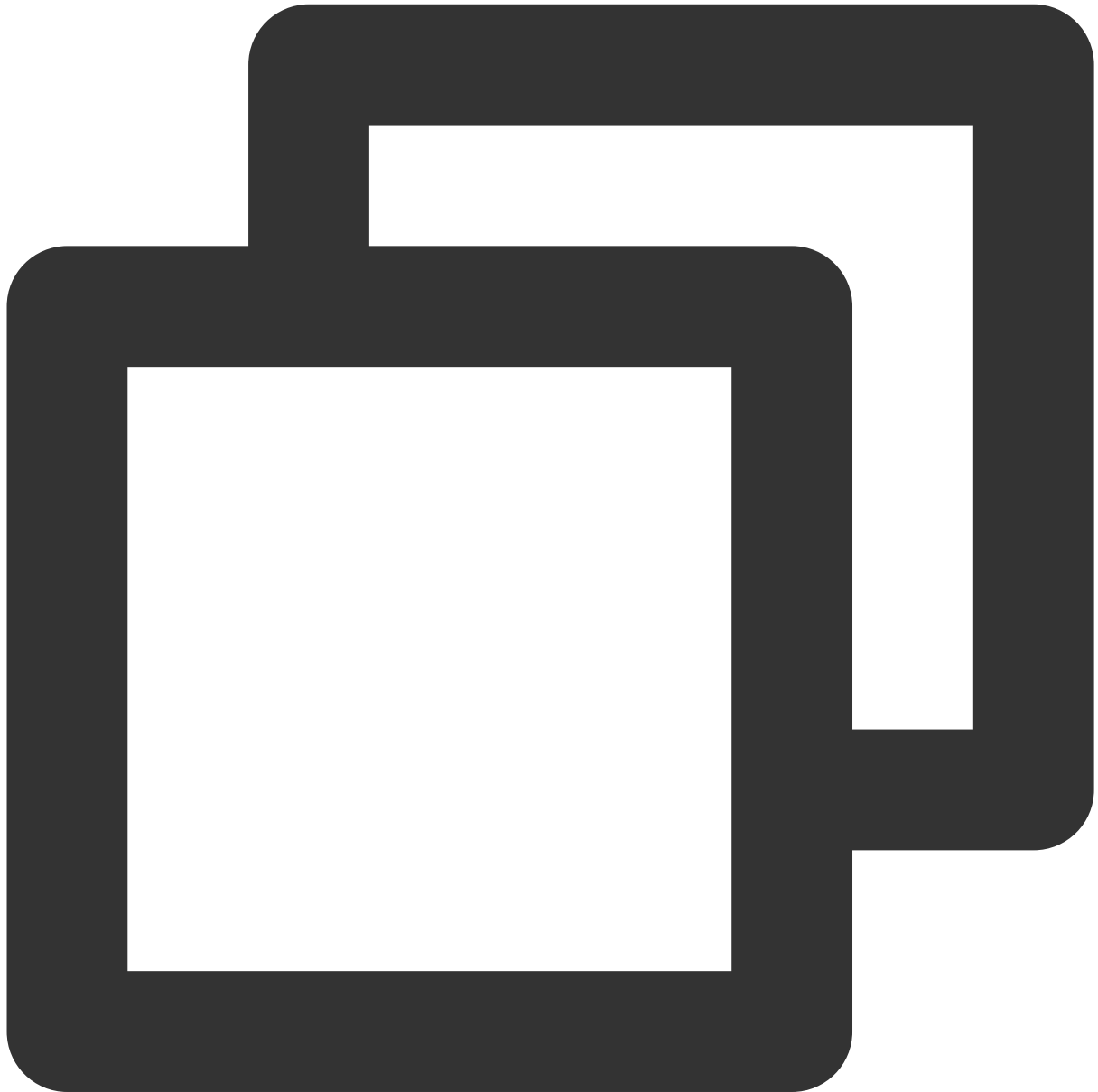
userRole

TUIRole

User Changed Role

## onUserVideoStateChanged

User Video Status Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVideoStateChanged, ({ userId, streamType, hasVideo, ...info }) => {
  console.log('roomEngine.onUserVideoStateChanged', userId, streamType, hasVideo, ...info);
});
```

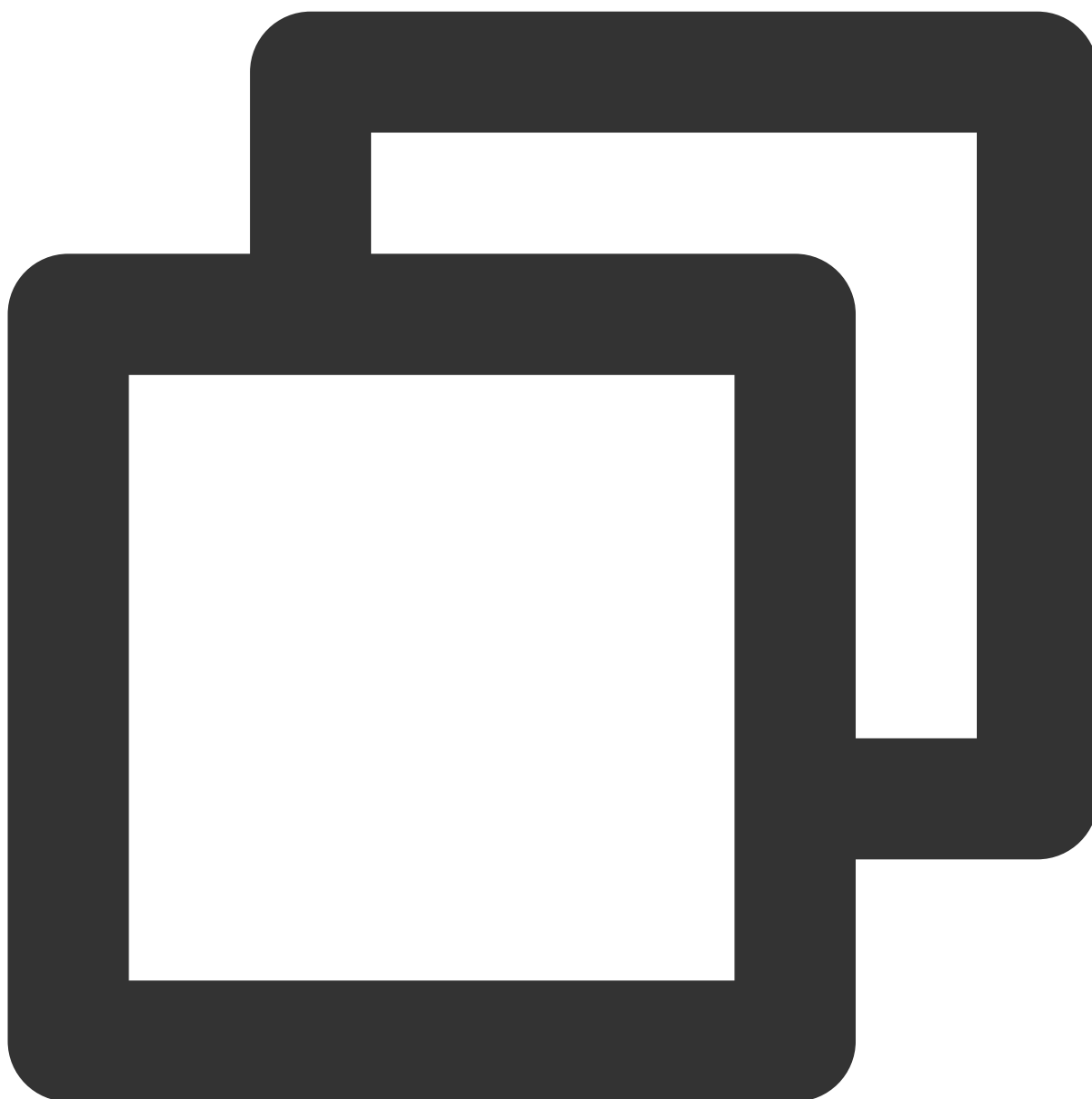
```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id
streamType	<a href="#">TUIVideoStreamType</a>	User Stream Type
hasVideo	boolean	Has Video streams
reason	<a href="#">TUIChangeReason</a>	Change Reason, Self-operation/Host-operation

## onUserAudioStateChanged

User Audio Status Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserAudioStateChanged, ({ userId, hasAudio, reason })
  console.log('roomEngine.onUserAudioStateChanged', userId, hasAudio, reason);
});
```

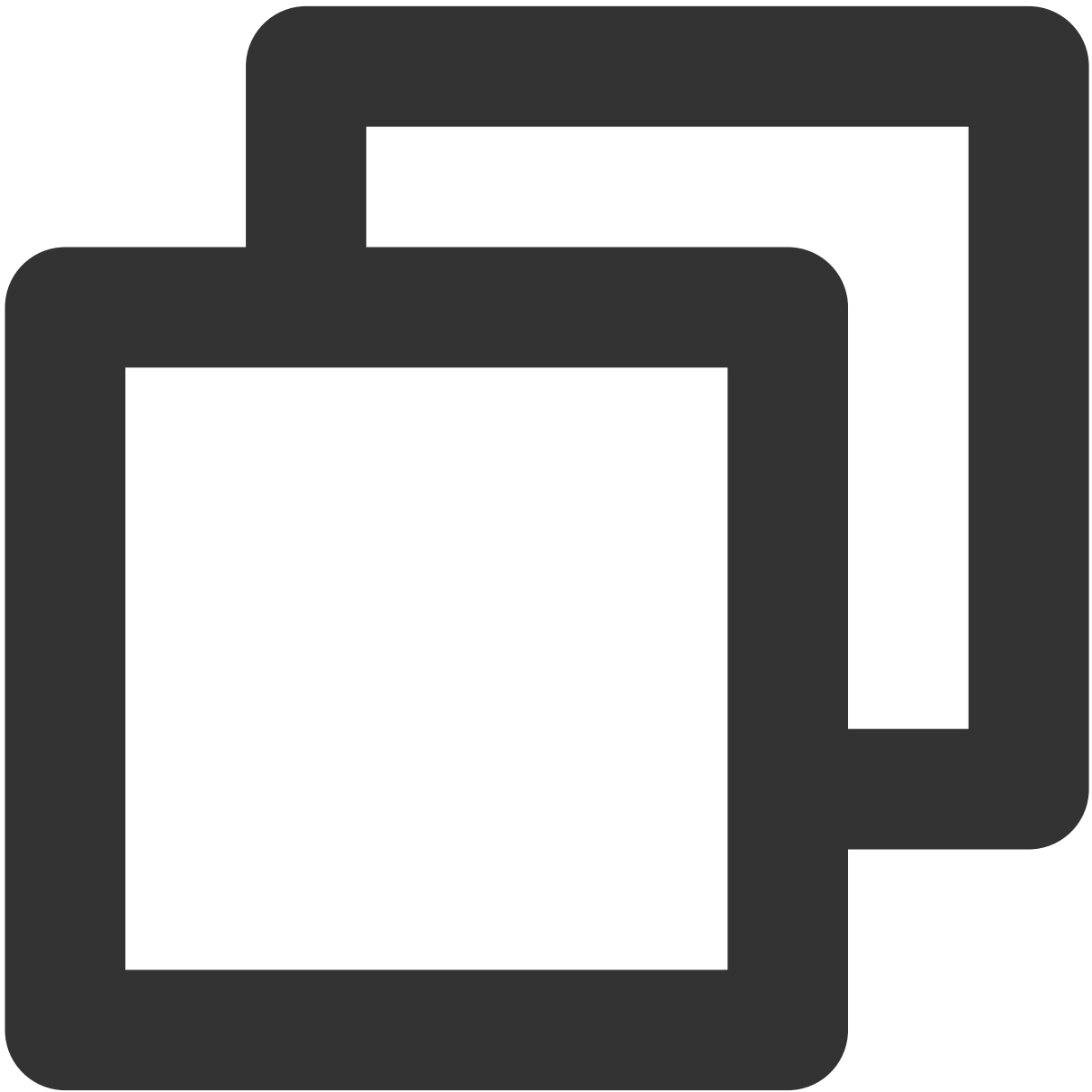
The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id

hasVideo	boolean	Has Audio streams
reason	<a href="#">TUIChangeReason</a>	Change Reason, Self-operation/Host-operation

## onSendMessageForUserDisableChanged

Member Camera Usage Permission Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable })
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Sending Text Message

## onUserVoiceVolumeChanged

User Volume Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVoiceVolumeChanged, ({ userVolumeList }) => {
  userVolumeList.forEach(userVolume => {
    console.log('roomEngine.onUserVoiceVolumeChanged', userVolume.userId, userVolume);
  })
});
```

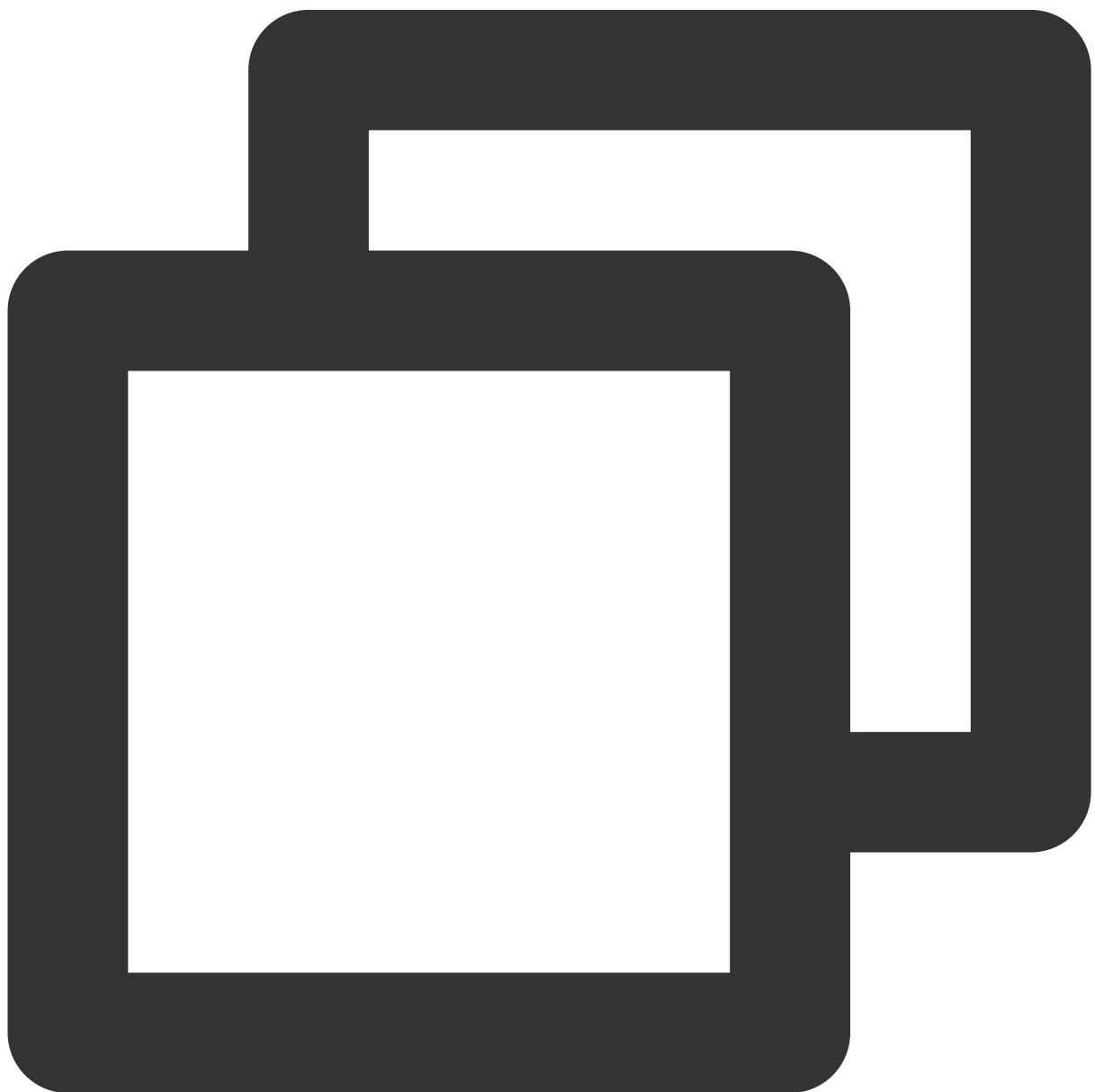
The parameters are shown in the table below:

Parameter	Type	Meaning

userVolumeList	Array.<object>	Volume of all users in the room, including userId and volume information, volume range is 1-100
----------------	----------------	---

## onUserNetworkQualityChanged

User Network Quality Change Event



```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onUserNetworkQualityChanged, ({ userNetworkList }) => {
  userNetworkList.forEach(userNetwork => {
    console.log('roomEngine.onUserNetworkQualityChanged', userNetwork.userId, userN
  })
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
networkMap	<a href="#">TUINetworkQuality</a>	Traverse Network Quality Level

## onSeatListChanged

Seat List Change Event





```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatListChanged, ({ seatList, seatedList, leftList })
  console.log('roomEngine.onSeatListChanged',seatList, seatedList, leftList);
});
```

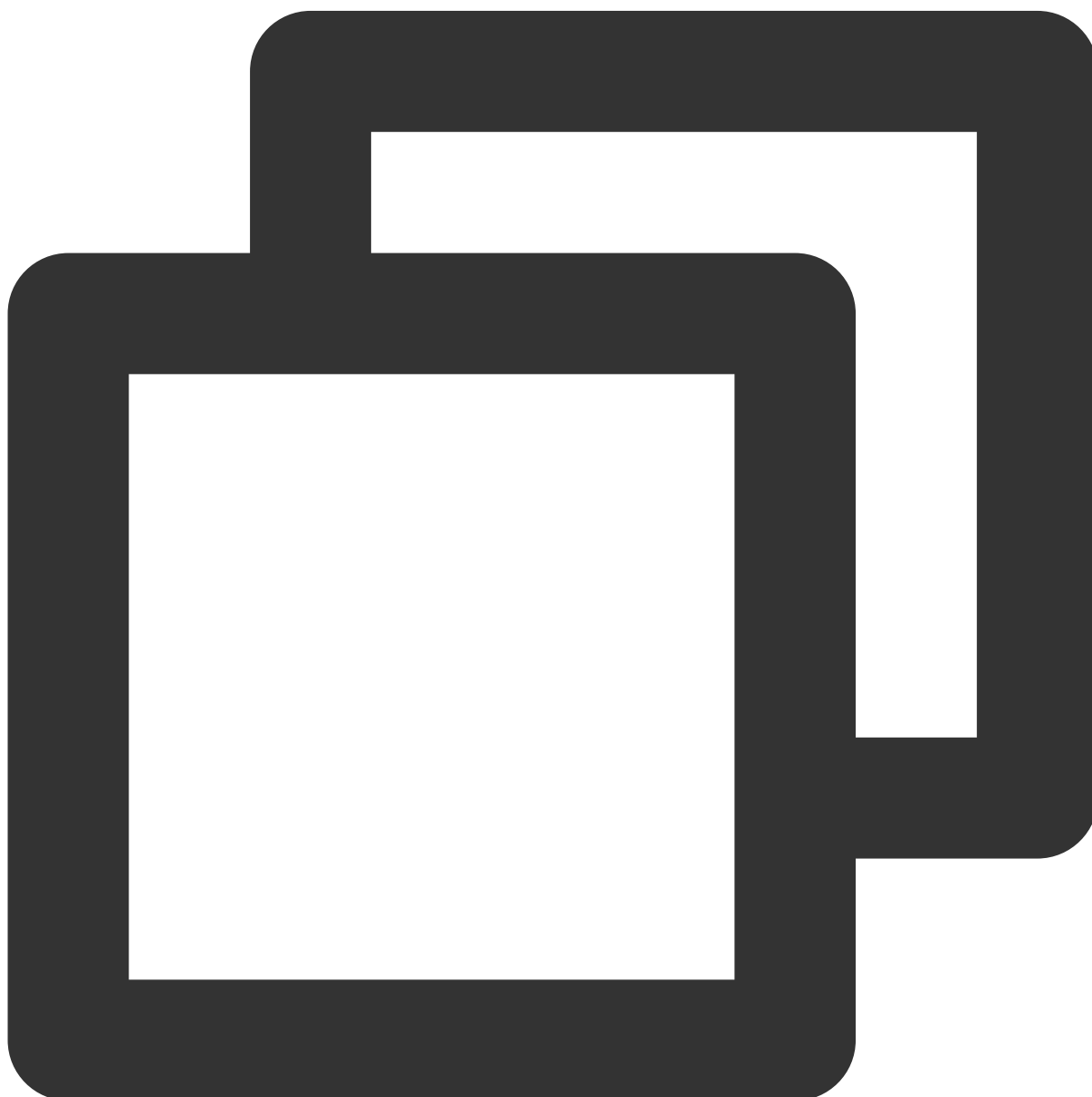
The parameters are shown in the table below:

Parameter	Type	Meaning
seatList	Array.< <a href="#">TUISeatInfo</a> >	Seat List

seatedList	Array.<TUISeatInfo>	New Seat Information
leftList	Array.<TUISeatInfo>	Left Seat Information

## onKickedOffSeat

Seat List Change Event



```
const roomEngine = new TUIRoomEngine();
```

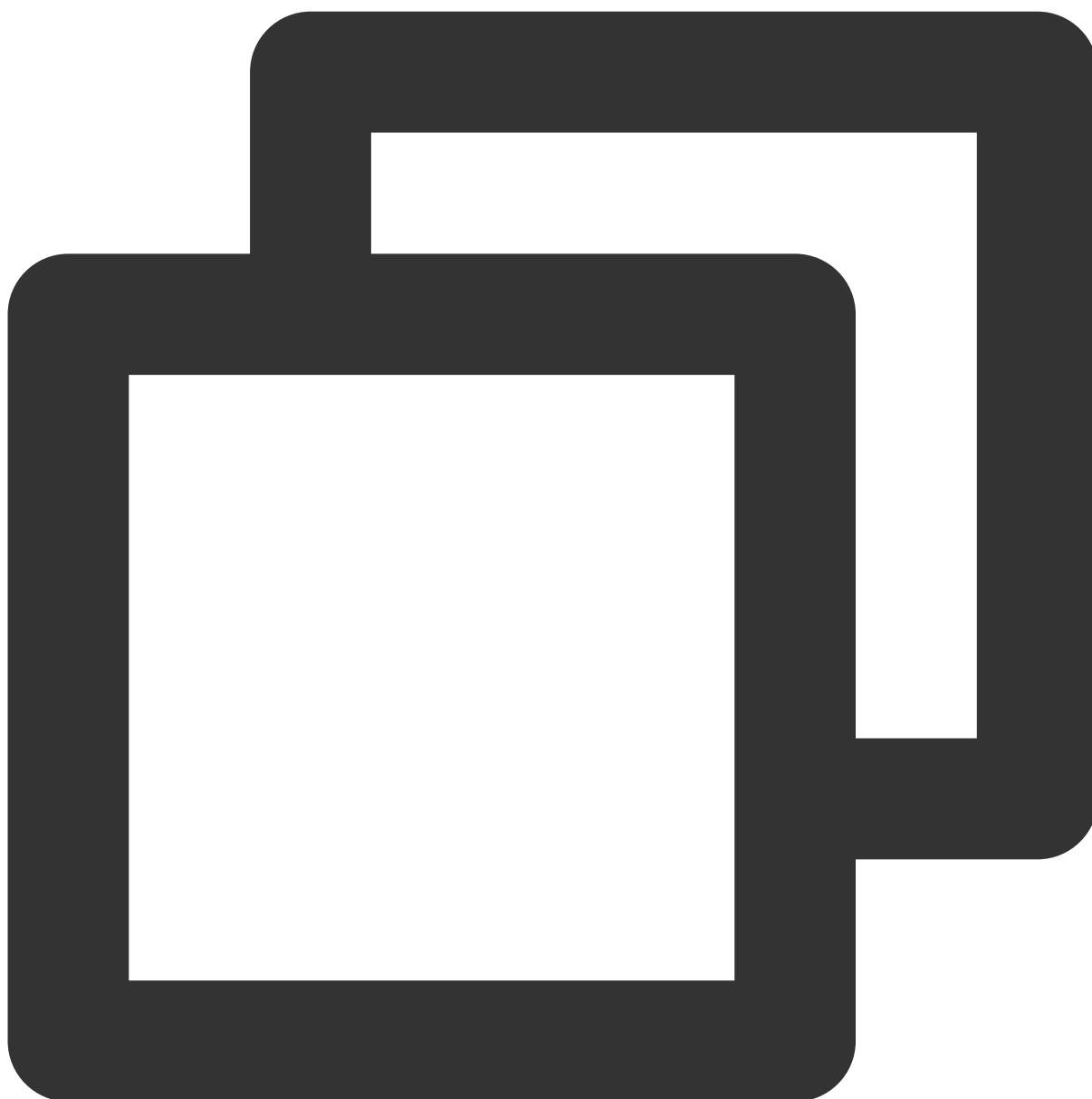
```
roomEngine.on(TUIRoomEvents.onKickedOffSeat, ({ userId }) => {  
    console.log('roomEngine.onKickedOffSeat', userId);  
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	String	Kicked off Seat User Id

## onRequestReceived

Error Event



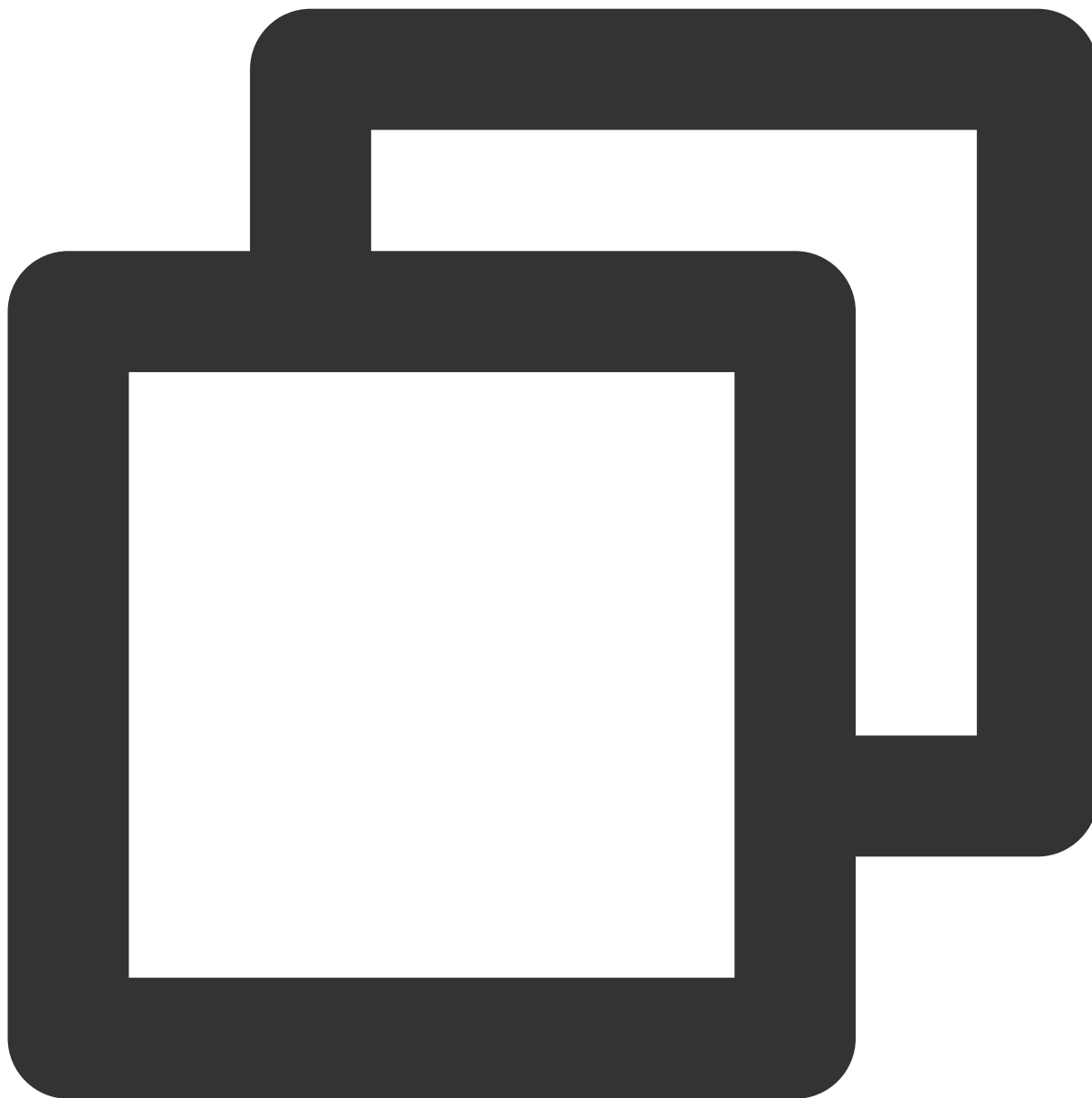
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestReceived, ({ request }) => {
  console.log('roomEngine.onRequestReceived', request);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Received

## onRequestCancelled

Request Cancelled Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestCancelled, ({ requestId, userId }) => {
  console.log('roomEngine.onRequestCancelled', requestId, userId);
});
```

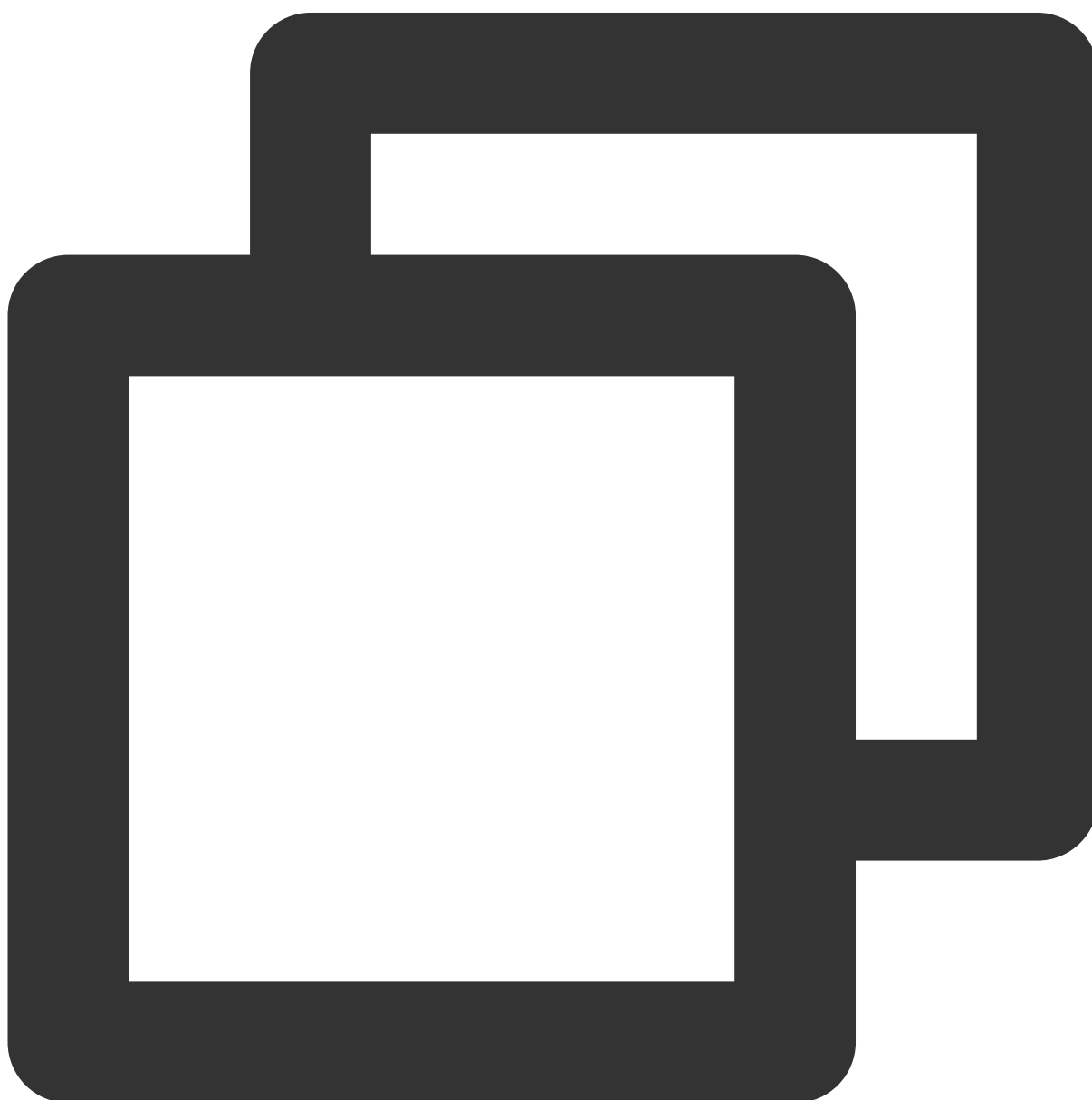
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
requestId	string	Request Id
userId	string	User Id of Cancelled Request

## onReceiveTextMessage

Receive Text Message Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveTextMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveTextMessage', roomId, message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room Id
message	<a href="#">TUIMessage</a>	Receive Text Message

## onReceiveCustomMessage

Receive Custom Message Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveCustomMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveCustomMessage', roomId, message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room Id



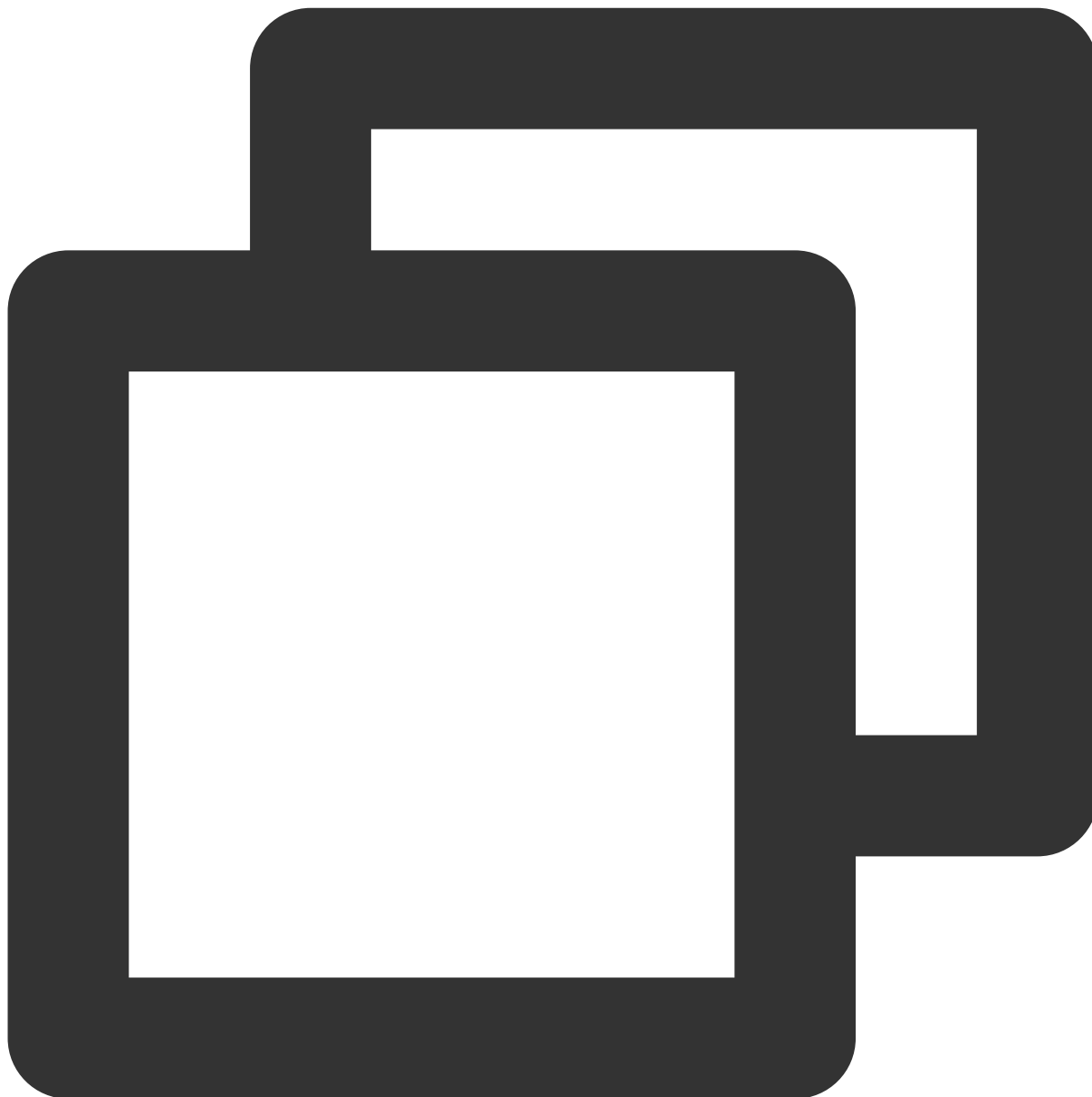
message

TUIMessage

Receive Text Message

## onDeviceChange

Device Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onDeviceChange, ({ deviceId, type, state }) => {
  console.log('roomEngine.onDeviceChange', deviceId, type, state);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
deviceId	string	Device Id
type	<a href="#">TRTCDeviceType</a>	Device Type
state	<a href="#">TRTCDeviceState</a>	Device Change Status

## onUserScreenCaptureStopped

Screen Sharing Stopped Event, when the user uses the built-in browser stop sharing button to end screen sharing, the user will receive the ' `onUserScreenCaptureStopped` ' event to modify the screen sharing status.



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserScreenCaptureStopped, () => {
  console.log('roomEngine.onUserScreenCaptureStopped', deviceId, type, state);
});
```

# TUIRoomEngine Defines

Last updated : 2023-11-14 17:05:09

Introduction to Key Type Definition of TUIRoomEngine web side.

## Enumeration Value

### TUIRole

User Role, TUIRoomEngine provides three user roles, which are Host, Administrator, and Regular User.

Field	Type	Description
kRoomOwner	number	Host Role
kAdministrator	number	Administrator Role
kGeneralUser	number	Regular User Role

### TUIVideoProfile

Video Resolution

Field	Type	Description
kLowDefinition	number	Low Resolution
kStandardDefinition	number	SD
kHighDefinition	number	HD
kSuperDefinition	number	Ultra HD

### TUIAudioProfile

Audio Resolution

Field	Type	Description
kAudioProfileSpeech	number	Voice Mode
kAudioProfileDefault	number	Standard Mode (Default Mode)
kAudioProfileMusic	number	Music Mode

## TUIVideoStreamType

### Streams Type

Field	Type	Description
kCameraStream	number	Camera Streams
kScreenStream	number	Screen Sharing Streams
kCameraStreamLow	number	Low Resolution Camera Streams

## TUINetworkQuality

### Network Status

Field	Type	Description
kQualityUnknown	number	Network Condition Unknown
kQualityExcellent	number	Network Condition Excellent
kQualityGood	number	Network Condition Good
kQualityPoor	number	Network Condition Fair
kQualityBad	number	Network Condition Poor
kQualityVeryBad	number	Network Condition Very Poor
kQualityDown	number	Network Connection Disconnected

## TUIRoomType

### Room Type

Field	Type	Description
kGroup	number	Group Type Room, suitable for conference and educational scene, the microphone position in this room is unordered and has no quantity limit
kOpen	number	Open Type Room, suitable for live streaming scene, the microphone position in this room is ordered and has a quantity limit

## TUISpeechMode

### Speech Type

--

Field	Type	Description
kFreeToSpeak	number	Free Speech Mode
kApplyToSpeak	number	Hand-raising Speech Mode
kSpeakAfterTakingSeat	number	Speak After Sitting (Grab Microphone Position)

## TUICaptureSourceType

### Screen Sharing Type

Field	Type	Description
kWindow	number	Sharing Target is a specific Windows or Mac window todo (only for electron)
kScreen	number	Sharing Target is the entire Windows desktop or Mac desktop

## TUIChangeReason

Change Reason (Audio and Video Status Change Operation Reason: Self-initiated modification or modified by room owner/administrator)

Field	Type	Description
kChangedBySelf	number	Self-operation
kChangedByAdmin	number	Room Owner or Administrator Operation

## TUIRequestAction

### Room Type

Field	Type	Description
kInvalidAction	number	Invalid Operation
kRequestToOpenRemoteCamera	number	Request Remote Camera On
kRequestToOpenRemoteMicrophone	number	Request Remote Mic On
kRequestToConnectOtherRoom	number	Request Remote Cross-room Streaming, web side temporarily unsupported
kRequestToTakeSeat	number	Request Go Live

kRequestRemoteUserOnSeat	number	Request Remote Go Live
--------------------------	--------	------------------------

## TUIRequestCallbackType

Request Type

Field	Type	Description
kRequestAccepted	number	Peer Accepted
kRequestRejected	number	Peer Rejected
kRequestCancelled	number	Request Canceled
kRequestTimeout	number	Request Timeout
kRequestError	number	Request Error

## Type Definition

### TUIRoomInfo

Room data, user can use `roomEngine.getRoomInfo` to get room data.

Field	Type	Description
roomId	string	Room Number, String Type Room Number
roomType	<a href="#">TUIRoomType</a>	Room Type
owner	string	Host's userId
name	string	Room ID
createTime	string	Creation time
roomMemberCount	number	Current total number of people in the room
maxSeatCount	number	Maximum number of microphone positions in the room
enableVideo	boolean	Allow users to join and turn on Audio
enableAudio	boolean	Allow users to join and turn on Video
enableMessage	boolean	Allow users to join and send messages
enableSeatControl	boolean	Enable microphone position control

## TUIUserInfo

### User Information

Field	Type	Description
userId	string	User Id
userName	string	User Name
avatarUrl	string	User Avatar
userRole	<a href="#">TUIRole</a>	User Role
hasAudioStream	boolean	Whether there are Audio streams
hasVideoStream	boolean	Whether there are Video streams
hasScreenStream	boolean	Whether there is Screen Sharing stream

## TUIMessage

### Message Information

Field	Type	Description
messageId	string	Message Id
message	string	Message
timestamp	number	Timestamp information, accurate to seconds
userId	string	User Id
userName	string	User name
avatarUrl	string	User Avatar

## TUIRequest

### Request Information

Field	Type	Description
requestAction	<a href="#">TUIRequestAction</a>	Request Type
timestamp	number	Request Initiation Time
requestId	string	Request Id v1.0.2 and above versions requestId type is string;



		v1.0.0 and v1.0.1 versions requestId type is number;
userId	string	Initiate Request's User Id
content	string	Other Content

## TUIRequestCallback

### Request Callback Information

Field	Type	Description
requestCallbackType	<a href="#">TUIRequestCallbackType</a>	Request Callback Type, Accept/Reject/Cancel/Timeout/Error
requestId	string	Request Id v1.0.2 and above versions requestId type is string; v1.0.0 and v1.0.1 versions requestId type is number;
userId	string	User Id
code	number	Request Response Code
message	string	Request Status Supplemental Description

## TUISeatInfo

### Microphone Position Information

Field	Type	Description
index	number	Microphone Position Number
userId	string	Microphone Position Correspondence's User Id
locked	boolean	Whether the current microphone position is locked
videoMuted	boolean	Whether the current microphone position prohibits Video
audioMuted	boolean	Whether the current microphone position prohibits Audio

# Electron RoomKit API

Last updated : 2024-06-14 11:49:41

## API Introduction

The TUIRoomKit API is a multi-person meeting component with an **Including UI Interface**. By using the TUIRoomKit API, you can quickly implement a meeting-like scenario through a simple interface. For more detailed integration steps, see: [Rapid Integration with TUIRoomKit](#).

## API Overview

`<ConferenceMainView />`: The component body of TUIRoomkit UI.

Conference: dependencies `ConferenceMainView` provided API.

API	Description
<a href="#">getRoomEngine</a>	Access the roomEngine instance. If roomEngine does not exist, it will return null.
<a href="#">on</a>	Listen to specified types of apid Integration with TUIRoomKit .Event. When the event occurs, the callback function will be called.
<a href="#">off</a>	Cancel listening for events of a specified type.
<a href="#">login</a>	Log in to the conference system.
<a href="#">logout</a>	Log out of the meeting system.
<a href="#">start</a>	Start a new meeting.
<a href="#">join</a>	Join an existing meeting.
<a href="#">leave</a>	Leave the current meeting.
<a href="#">dismiss</a>	Dismiss the current meeting.
<a href="#">setSelfInfo</a>	Set your user information.
<a href="#">setLanguage</a>	Set the interface language.
<a href="#">setTheme</a>	Set the interface topic.

<a href="#">enableWatermark</a>	Enable the text messaging feature in the application. For details, see: <a href="#">Text Watermark</a> .
<a href="#">disableScreenSharing</a>	Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.
<a href="#">disableTextMessaging</a>	Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.
<a href="#">hideFeatureButton</a>	Hide specific feature buttons in the application. By invoking this function and passing in the appropriate <a href="#">FeatureButton</a> enumerated values, the corresponding buttons will be hidden from the user interface.

## Introduction to ConferenceMainView attributes

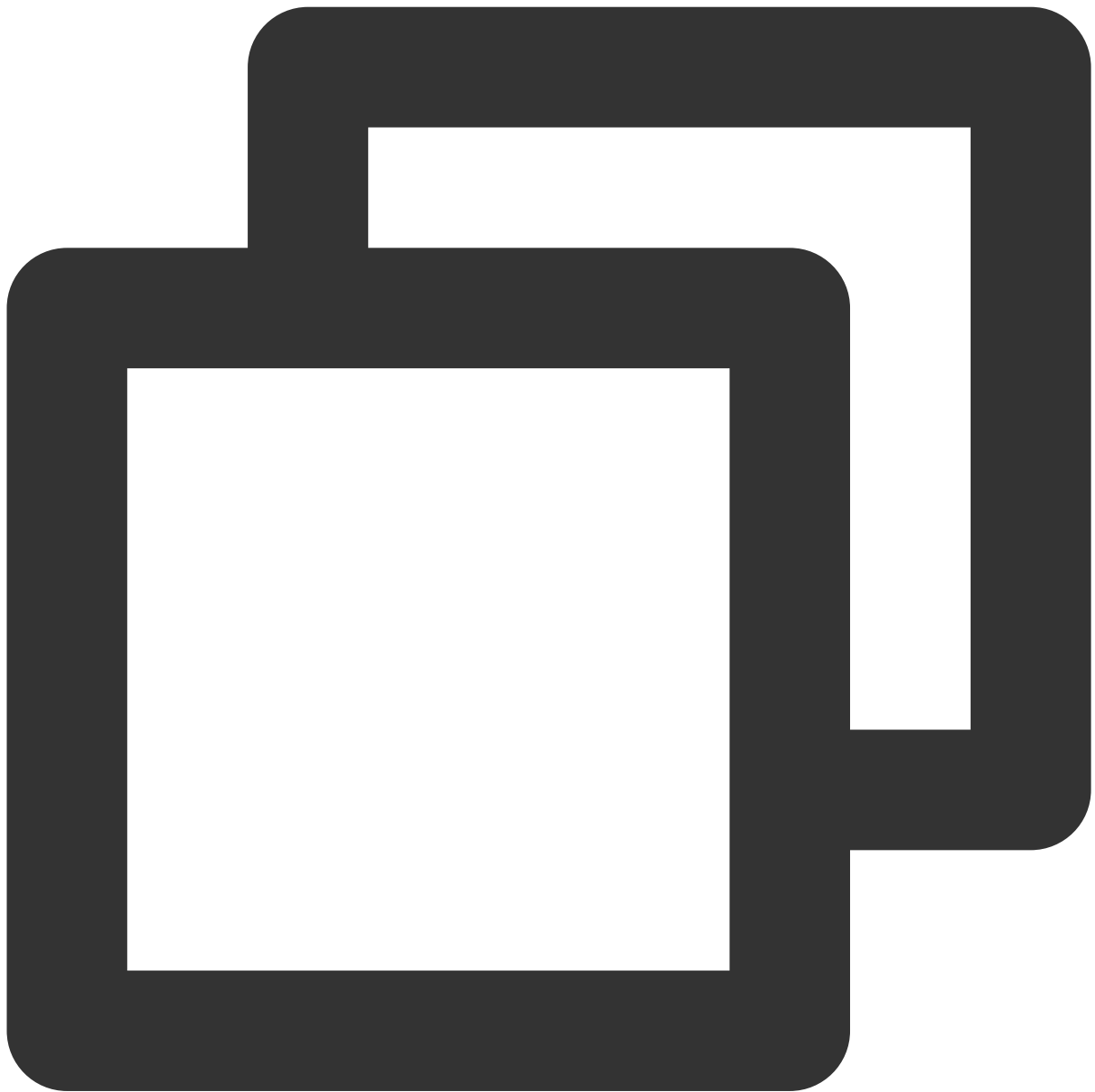
### Attribute Overview

Attribute	Description	Type	Required	Default Value
displayMode	Control of component display pattern: permanent: Permanent mode. The component is always displayed; internal control of the component's visibility is not exerted. If not controlled by the business end, the component will always remain visible. wake-up: Wake-up mode. The component is activated only after calling the <a href="#">conference start/join</a> interface and officially joining the conference; it will not be displayed beforehand.	'permanent'   'wake-up'	No	permanent

### Sample code

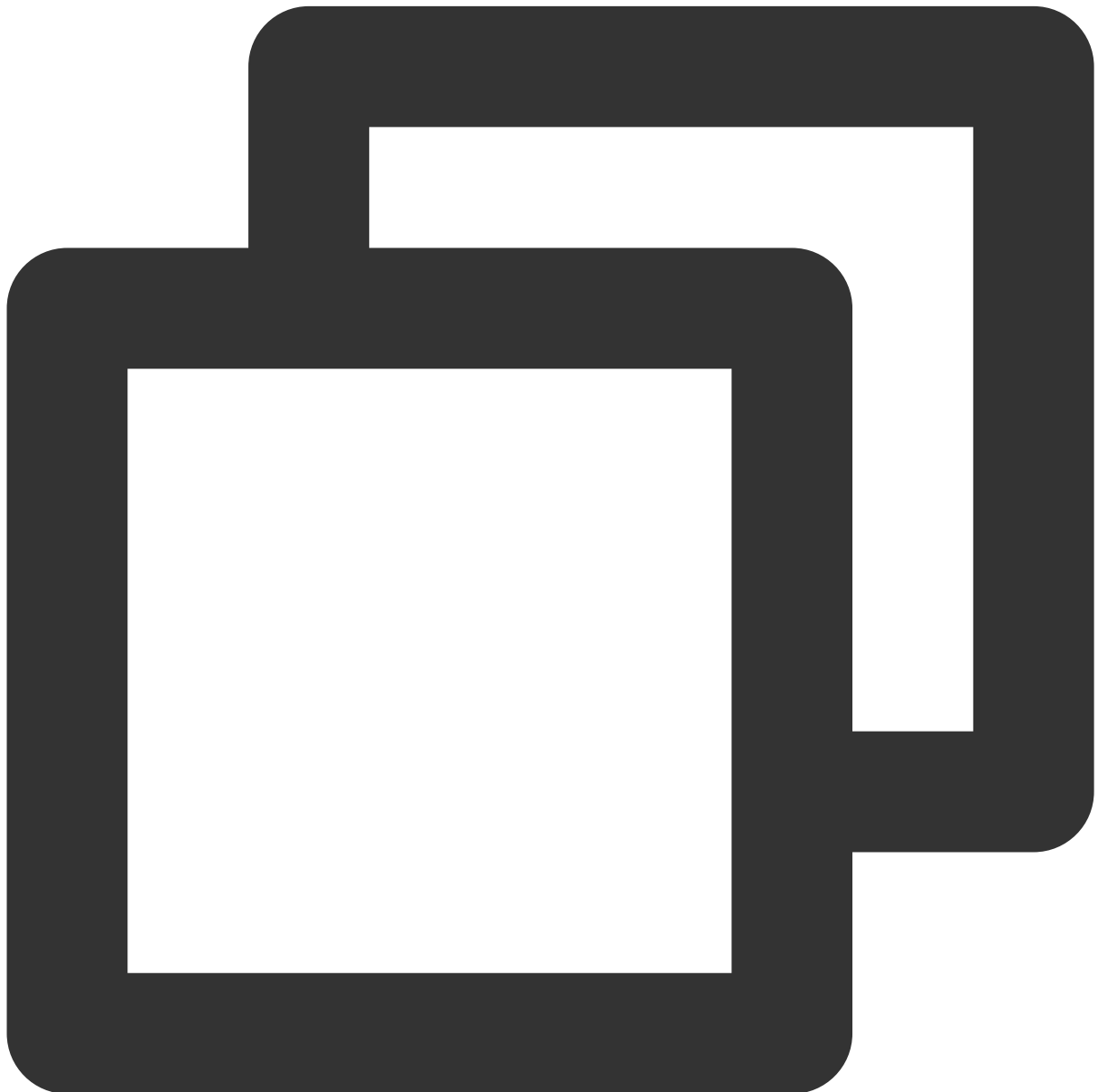
Vue3

Vue2



```
<template>
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue3';
const init = async () => {
  await conference.login({
    sdkAppId: 0,
    userId: '',
    userSig: '',
  });
};
```

```
await conference.start('123456', {  
  isSeatEnable: false,  
  isOpenCamera: true,  
  isOpenMicrophone: true,  
});  
}  
init();  
</script>
```



```
<template>  
  <ConferenceMainView display-mode="permanent"></ConferenceMainView>
```

```
</template>
<script>
import { ConferenceMainView, conference } from '@tencentcloud/roomkit-web-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
  async created() {
    await conference.login({
      sdkAppId: 0,
      userId: '',
      userSig: '',
    });
    await conference.start('123456', {
      isSeatEnable: false,
      isOpenCamera: true,
      isOpenMicrophone: true,
    });
  },
};
</script>
```

## Conference API Details

Conference provides a series of methods for managing and controlling online meeting features. By implementing this interface, developers can easily integrate online meeting features into their applications.

### **getRoomEngine**

Access the roomEngine instance. If roomEngine does not exist, it will return null.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
const roomEngine = conference.getRoomEngine();
```

Returns: *TUIRoomEngine* | *null*

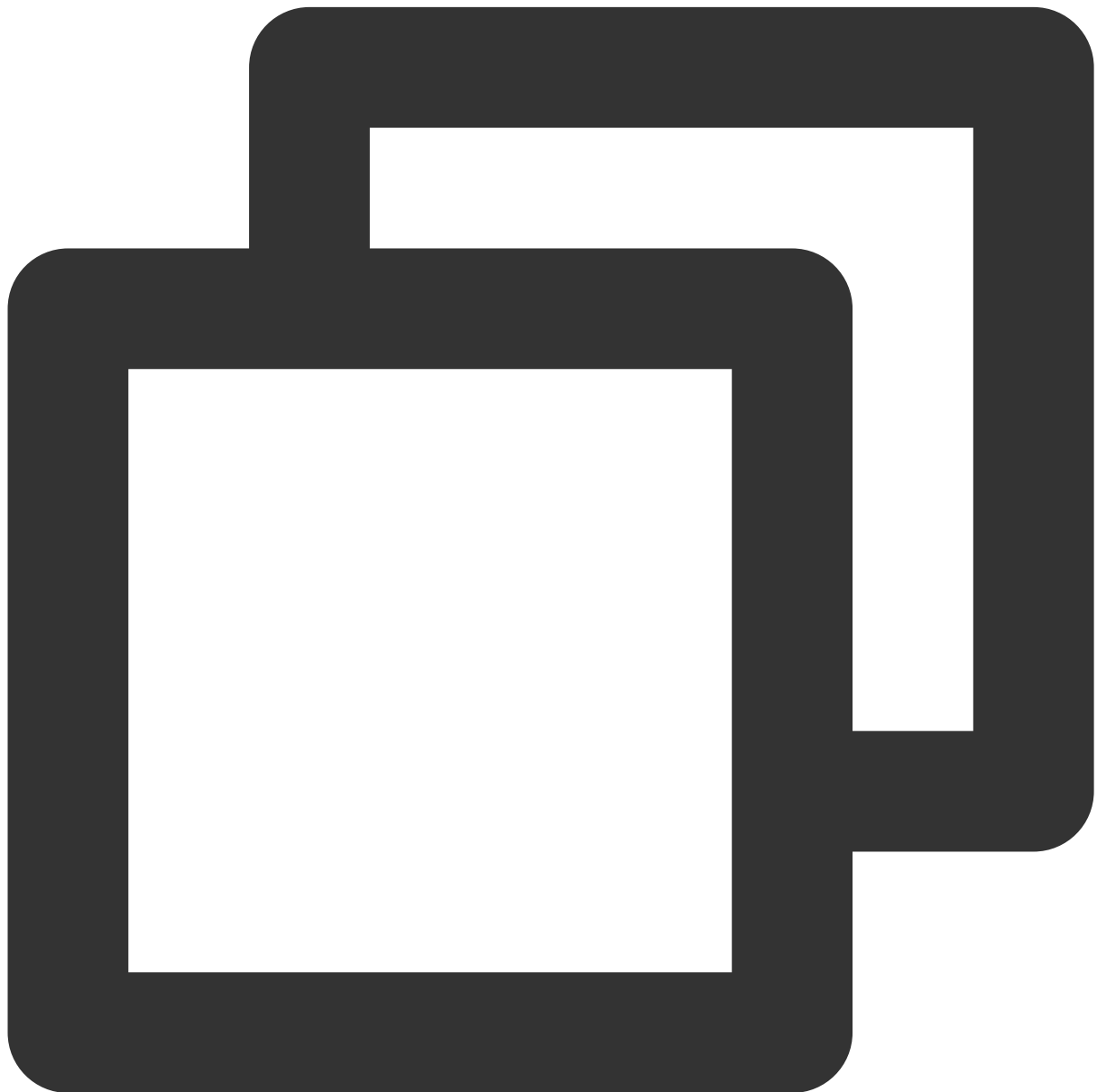
## on

Listen for a specified type of event. When the event occurs, the callback function will be called.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
eventType	<a href="#">RoomEvent</a>	-	Event Type to Listen For
callback	() => void	-	Callback Function Called When the Event Occurs



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference, RoomEvent } from '@tencentcloud/roomkit-web-vue3';
```



```
conference.on(RoomEvent.RoomStart, () => {  
  console.log('[conference] The meeting has already started.')});  
conference.on(RoomEvent.ROOM_DISMISS, () => {  
  console.log('[conference] The meeting has been dismissed')});
```

Returns: *void*

## off

Cancel listening for events of a specified type.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
eventType	<a href="#">RoomEvent</a>	-	The type of event to cancel listening for
callback	() => void	-	The callback function added previously



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.off('event', callback);
```

Returns: *void*

## login

Log in to the conference system.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
params	{sdkAppld: number; userId: string; userSig: string; tim?: ChatSDK}	-	Log in to the parameter object
sdkAppld	number	-	In the <a href="#">Real-time Audio and Video Console</a> click <b>Application Management &gt; create an application</b> . After creating a new application, you can access the sdkAppld information in <b>Application Information</b> .
userId	string	-	It's advised to limit the User ID length to 32 bytes, allowing only uppercase and lowercase letters (a-zA-Z), digits (0-9), underscores, and hyphens.
userSig	string	-	userSig Signature For the method of calculating userSig, please refer to <a href="#">UserSig Related</a> .
tim	ChatSDK (optional)	-	If you want to leverage more capabilities of the Instant Messaging SDK while integrating roomEngine, you can pass the created tim instance into TUIRoomEngine. For how to create a tim instance, please refer to <a href="#">TIM.create</a> .

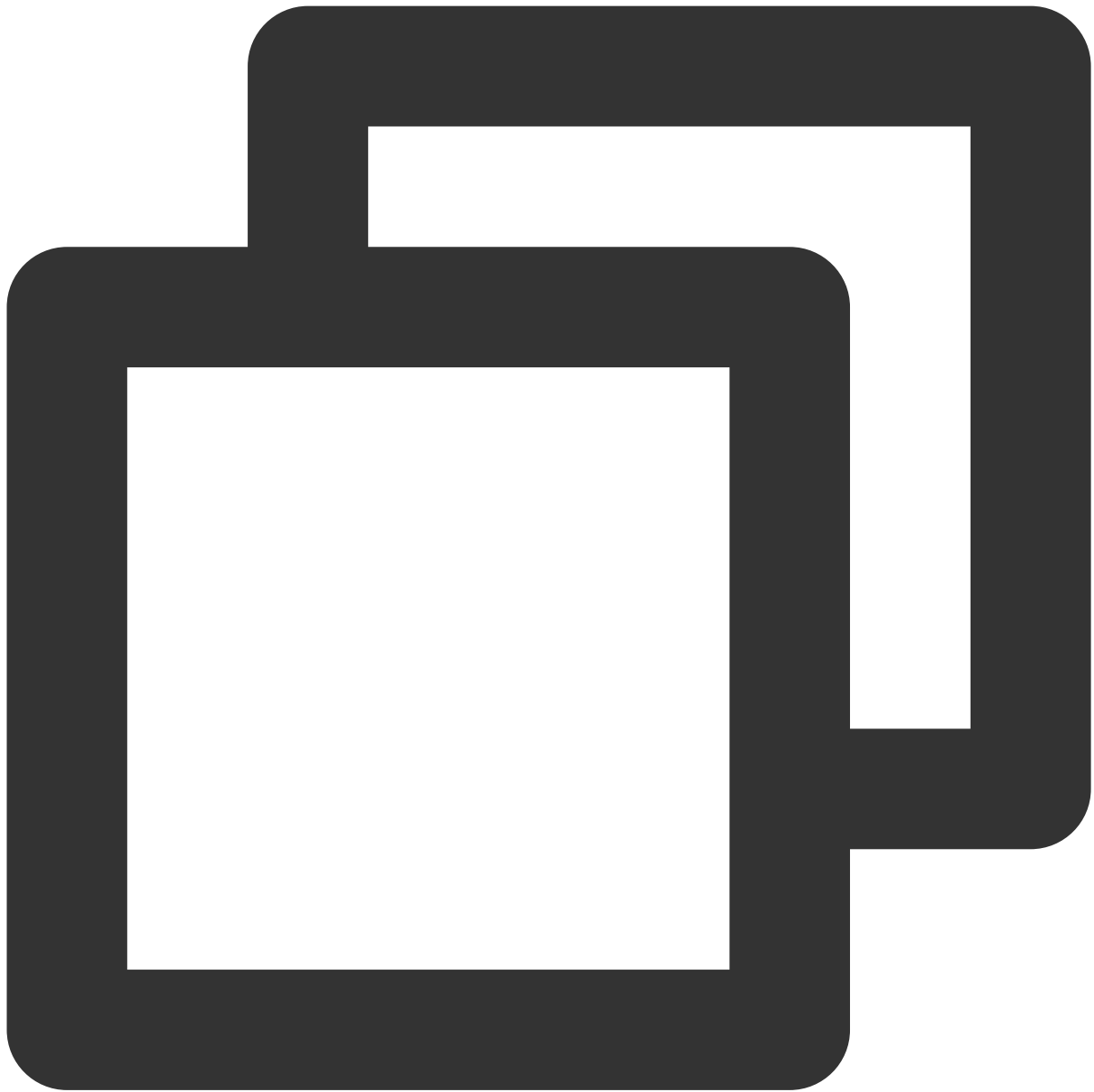


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.login({
  sdkAppId: 123456,
  userId: 'testUser',
  userSig: 'testSig'
});
```

Returns:*Promise<void>*

## logout

Log out of the meeting system.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.logout();
```

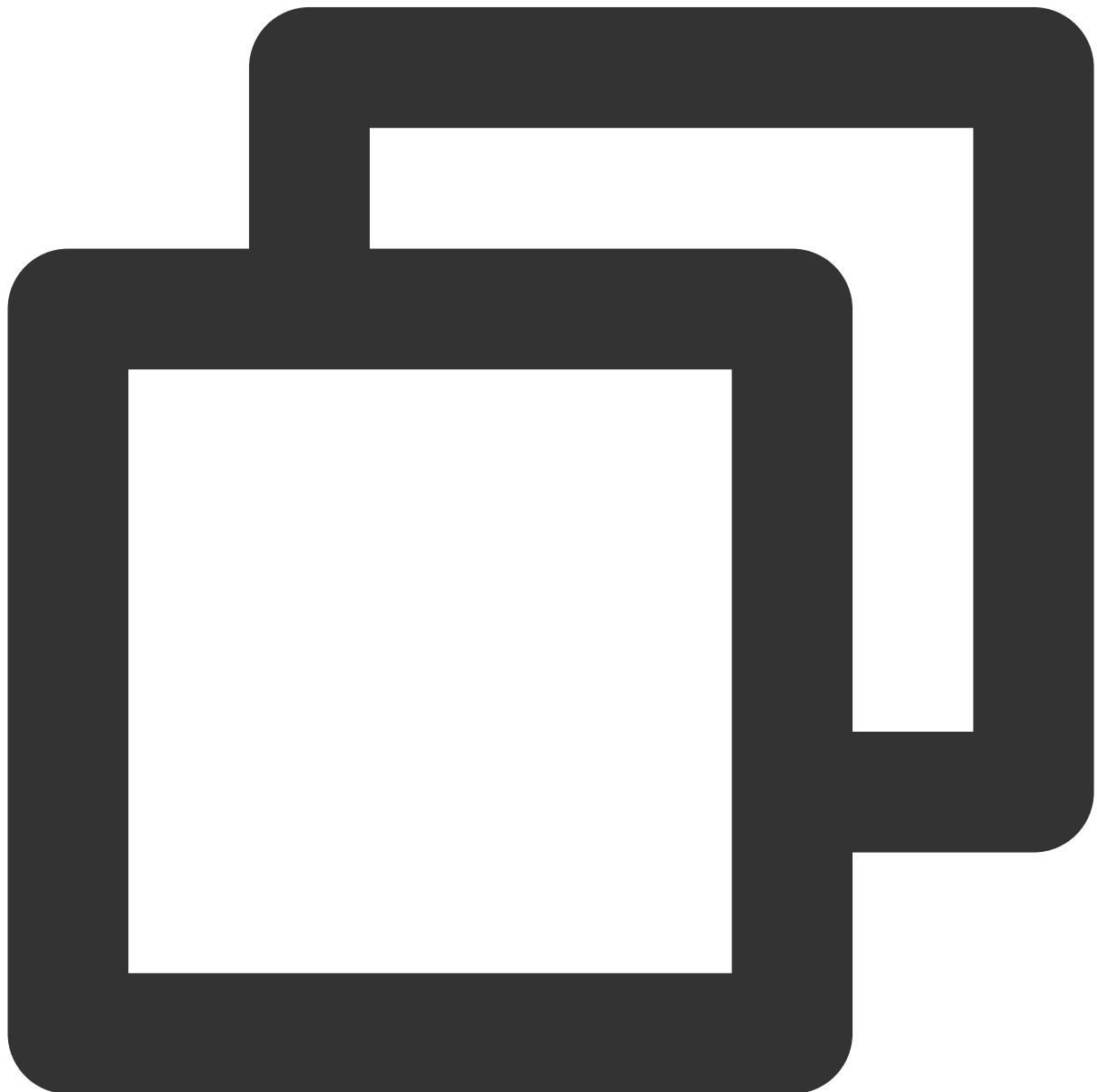
Returns:*Promise<void>*

## start

Start a new meeting.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
roomId	string	-	Meeting Room ID
params	<a href="#">StartParams</a>	-	Parameters for starting a meeting



```
// Note the package name. If you are using the vue2 version, please change the pack
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.start('123456', {
  roomName: 'TestRoom',
  isSeatEnabled: false,
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns: *Promise*<void>

## join

Join an existing meeting.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
roomId	string	-	Meeting Room ID
params	<a href="#">JoinParams</a>	-	Parameters for joining the meeting



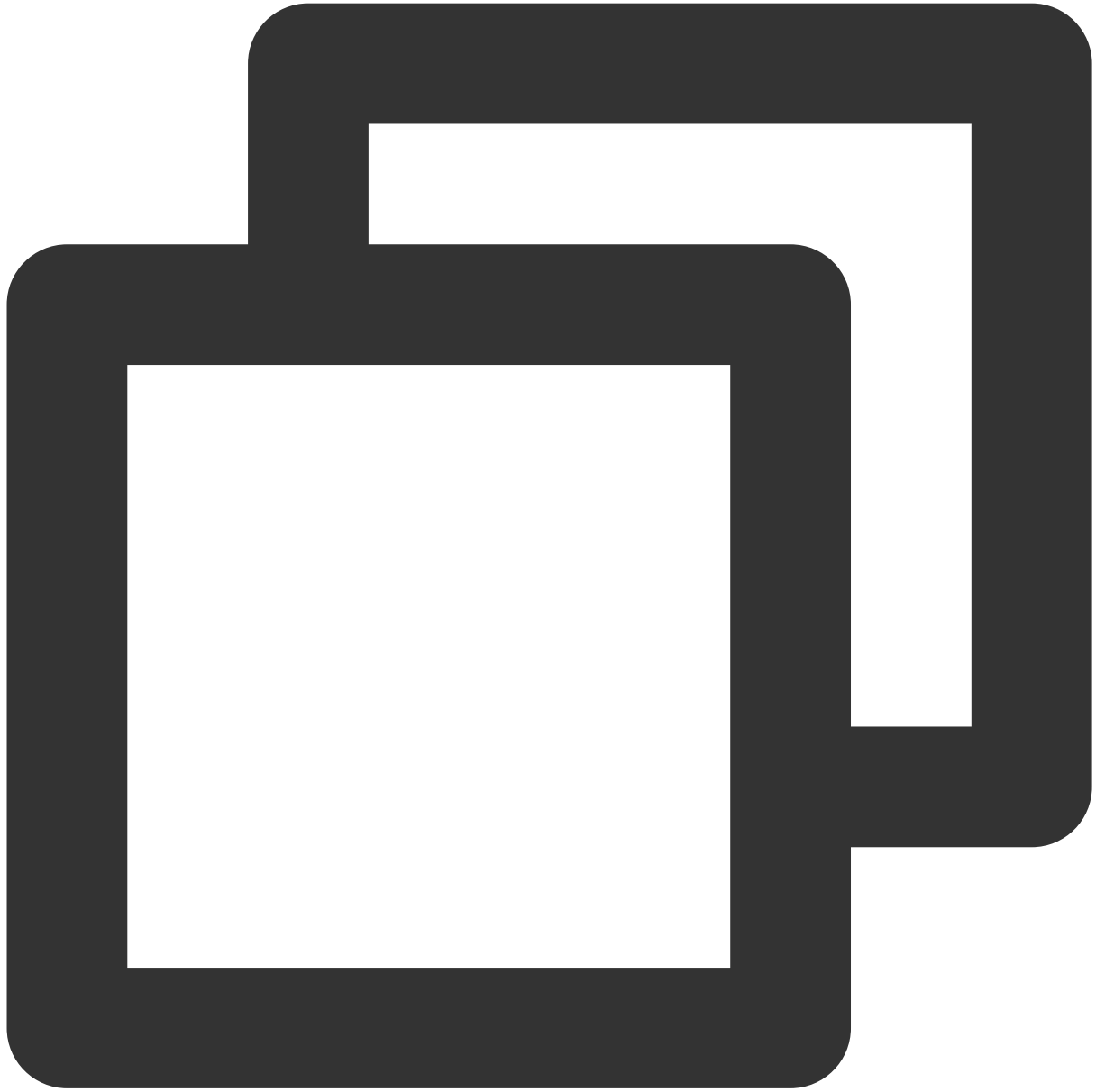
```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.join('123456', {
  isOpenCamera: false,
  isOpenMicrophone: false,
});
```

Returns:*Promise<void>*

**leave**



Leave the current meeting.

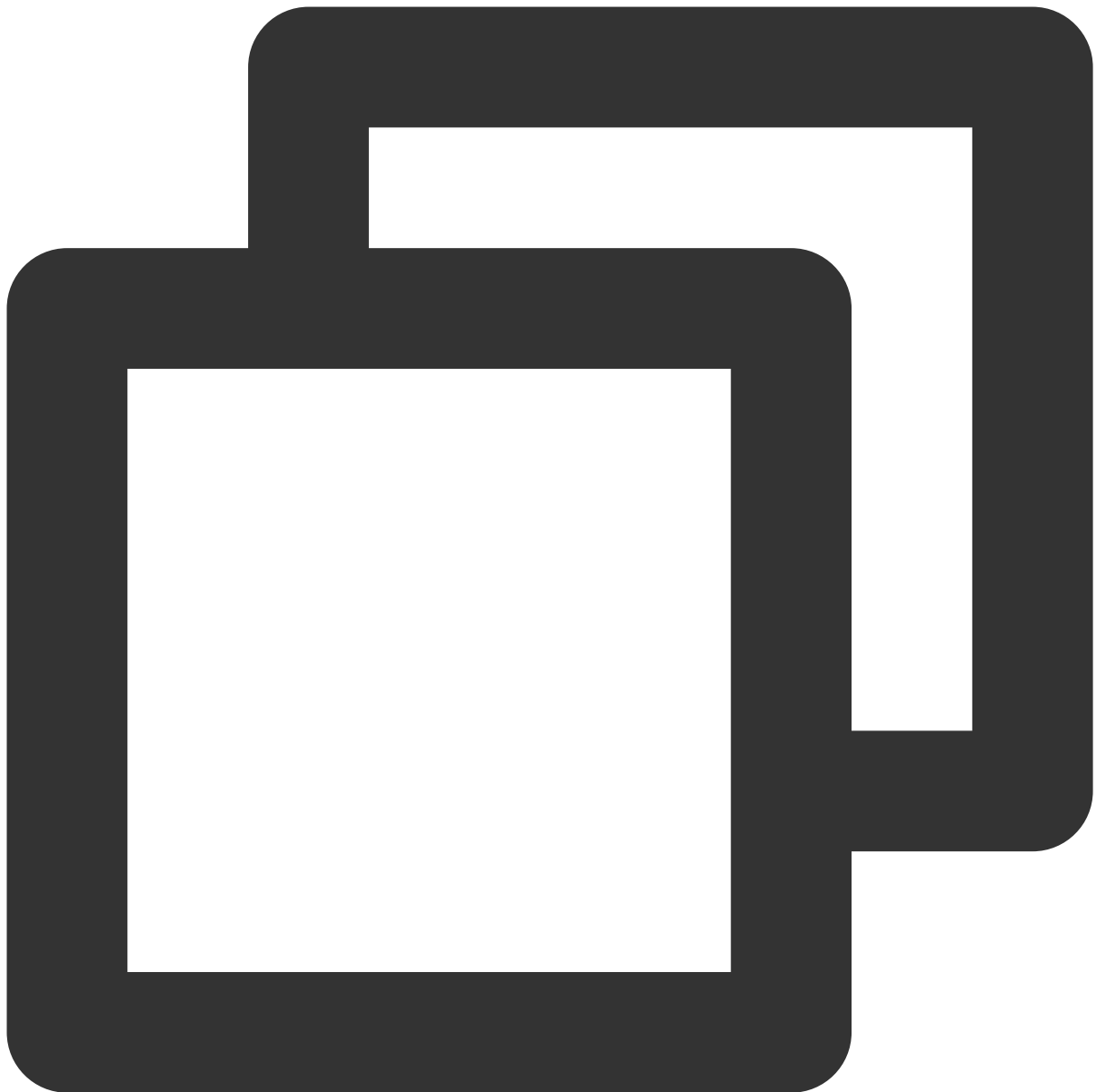


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.leave();
```

Returns:*Promise<void>*

## dismiss

Dismiss the current meeting.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.dismiss();
```

Returns:*Promise<void>*

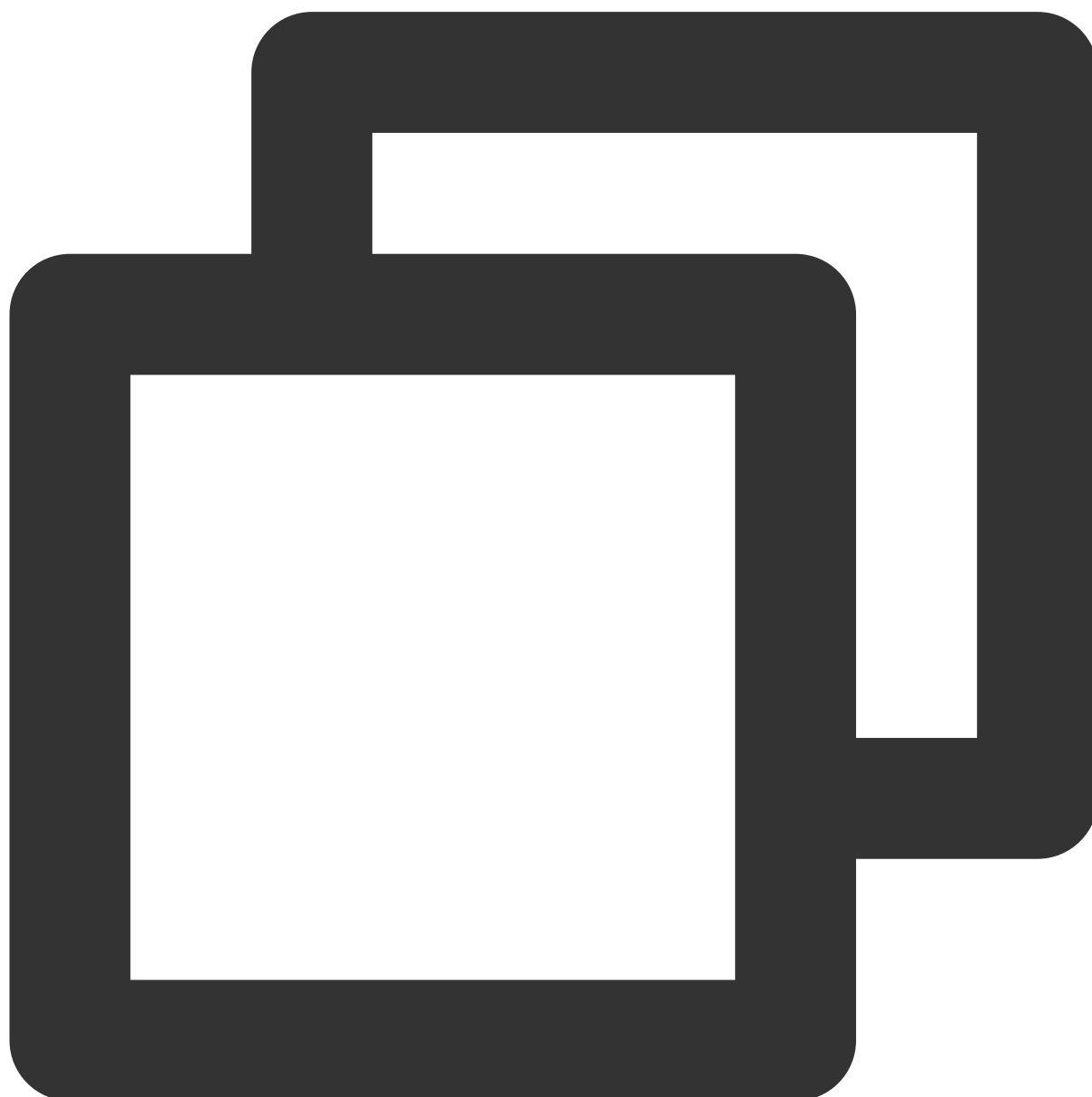
## setSelfInfo

Set your user information.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
options	{userName: string; avatarUrl: string}	-	User information object
userName	string (Optional)	-	User's nickname
avatarUrl	string (Optional)	-	User profile photo



```
// Note the package name. If you are using the vue2 version, please change the pack
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setSelfInfo({
  userName: 'test-name',
  avatarUlr: 'https://avatar.png'
});
```

Returns:*Promise<void>*

## setLanguage

Set the interface language.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
language	'zh-CN'   'en-US'	-	Language



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setLanguage('en-US');
```

Returns: *void*

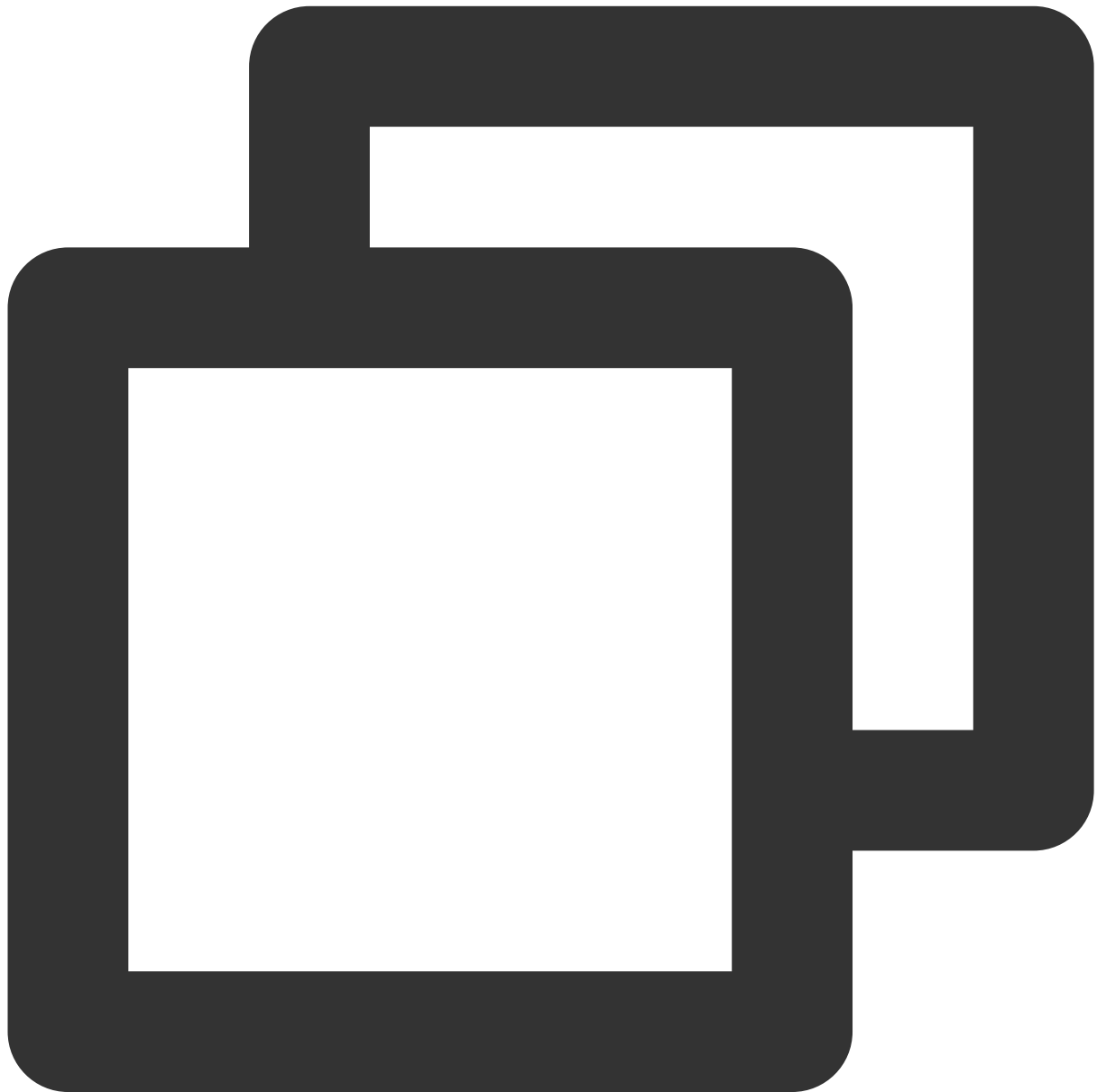
## setTheme

Set the interface topic.

The parameters are described as follows:

--	--	--	--

Parameter	Type	Default Value	Meaning
theme	'LIGHT'   'DARK'	-	Topic Type

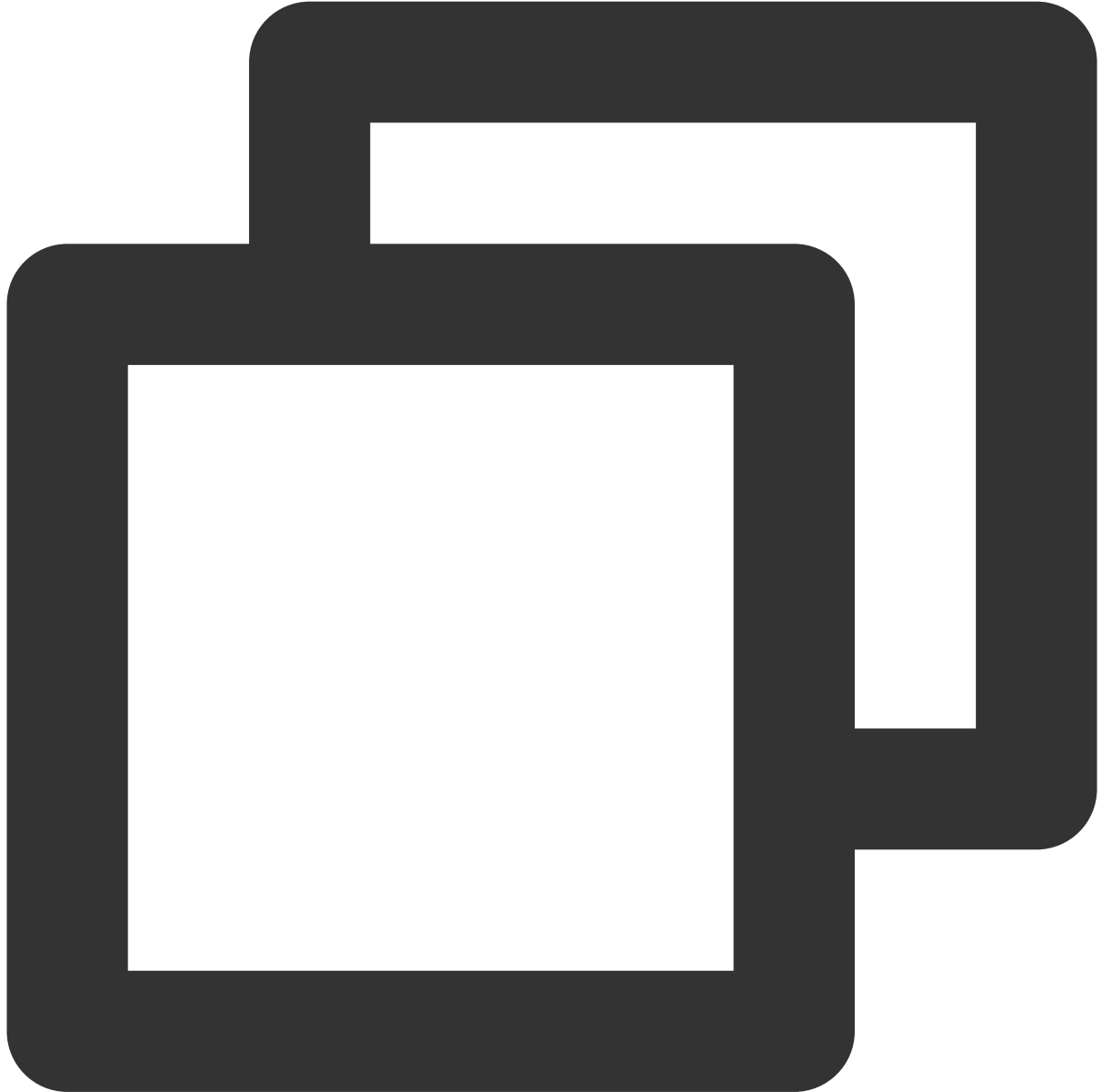


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.setTheme('DARK');
```

Returns: *void*

## enableWatermark

Enable the text messaging feature in the application. For details, see: [Text Watermark](#).

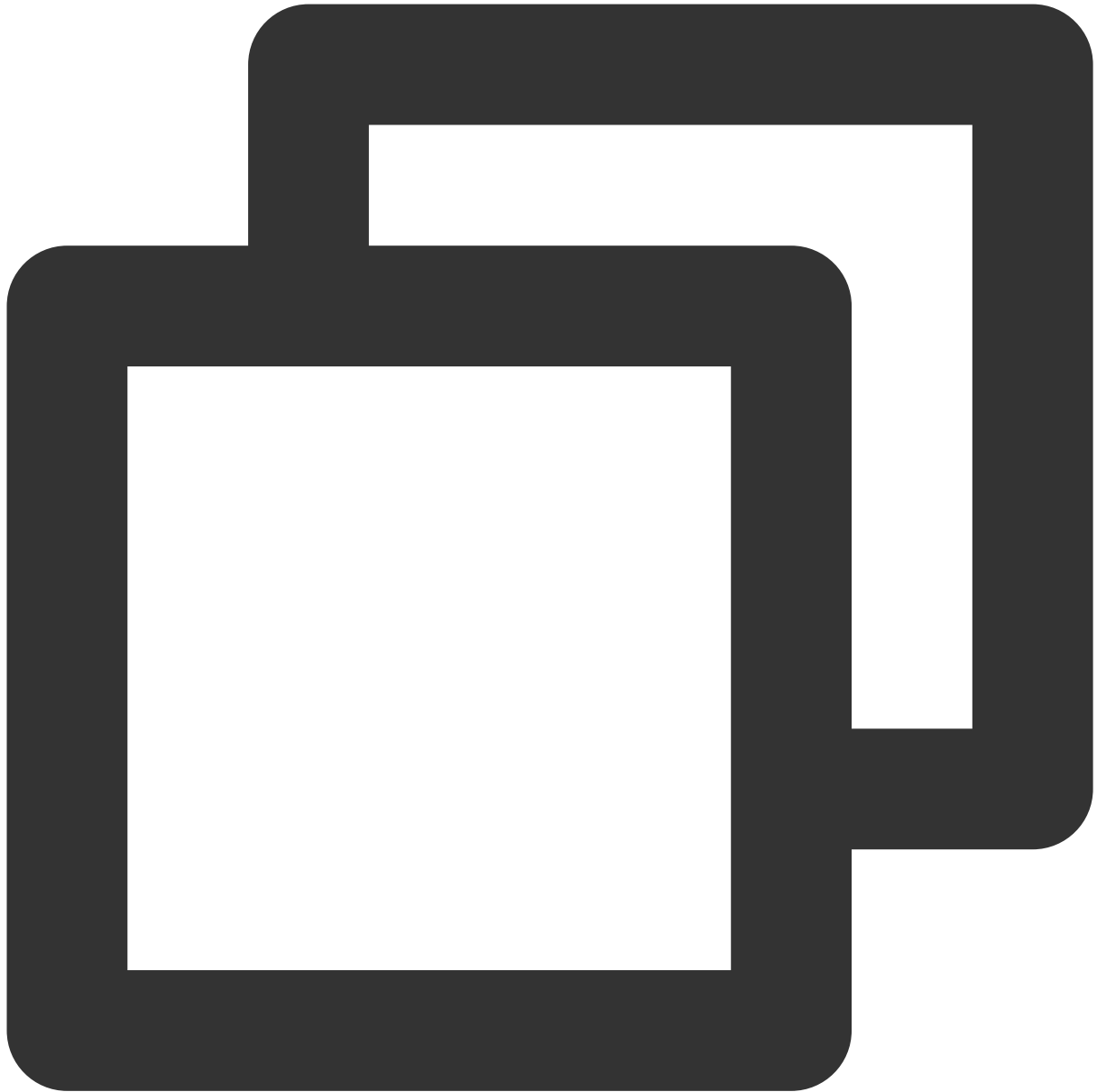


```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableWatermark();
```

Returns : *void*

## disableTextMessaging

Disable the text messaging feature in the application. After invoking this function, users will not be able to send or receive text messages.



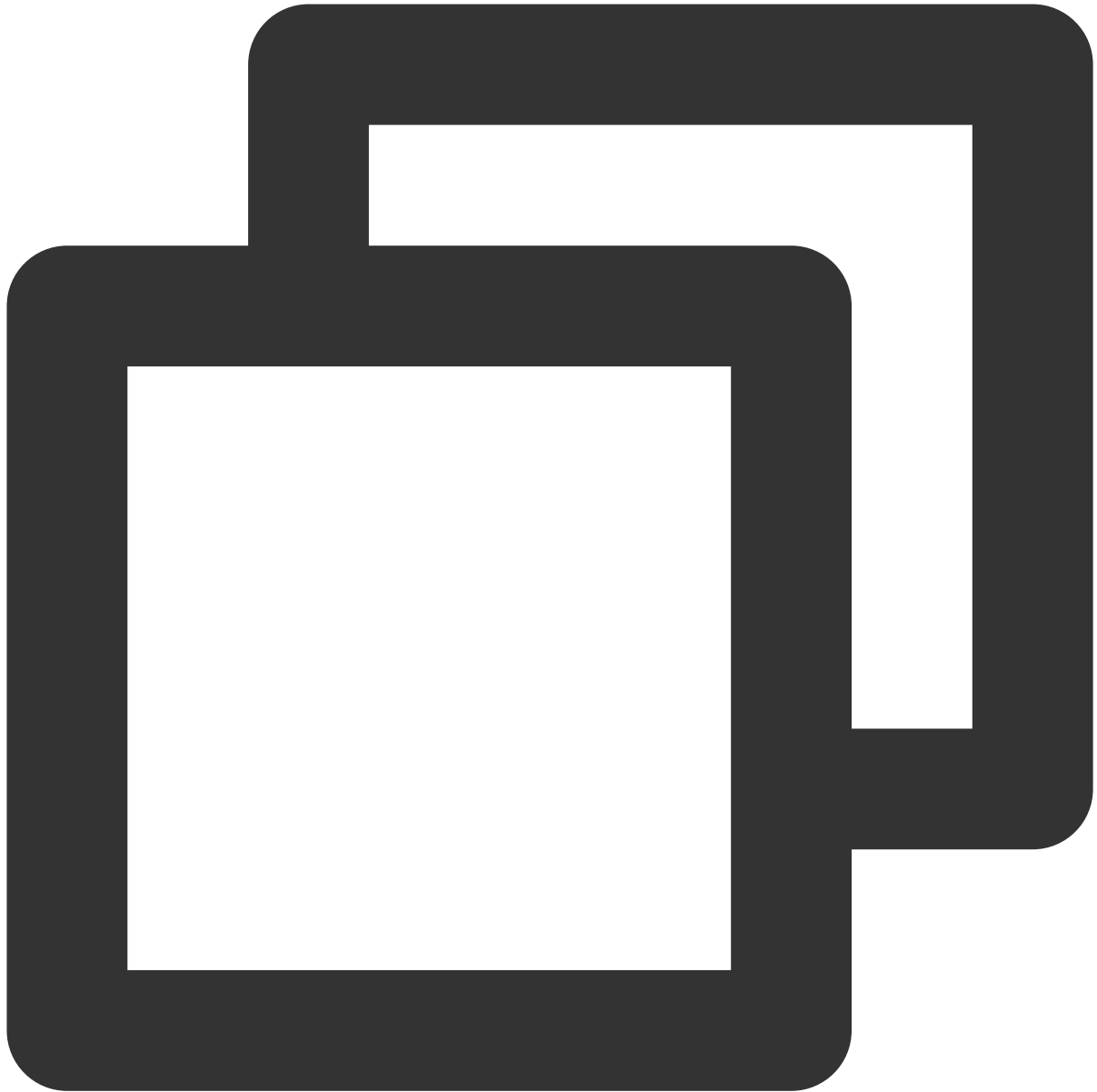
```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableTextMessaging();
```

Returns: *void*

## **disableScreenSharing**



Disable the screen sharing feature in the application. After invoking this function, users will not be able to share their screen with others.



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.disableScreenSharing();
```

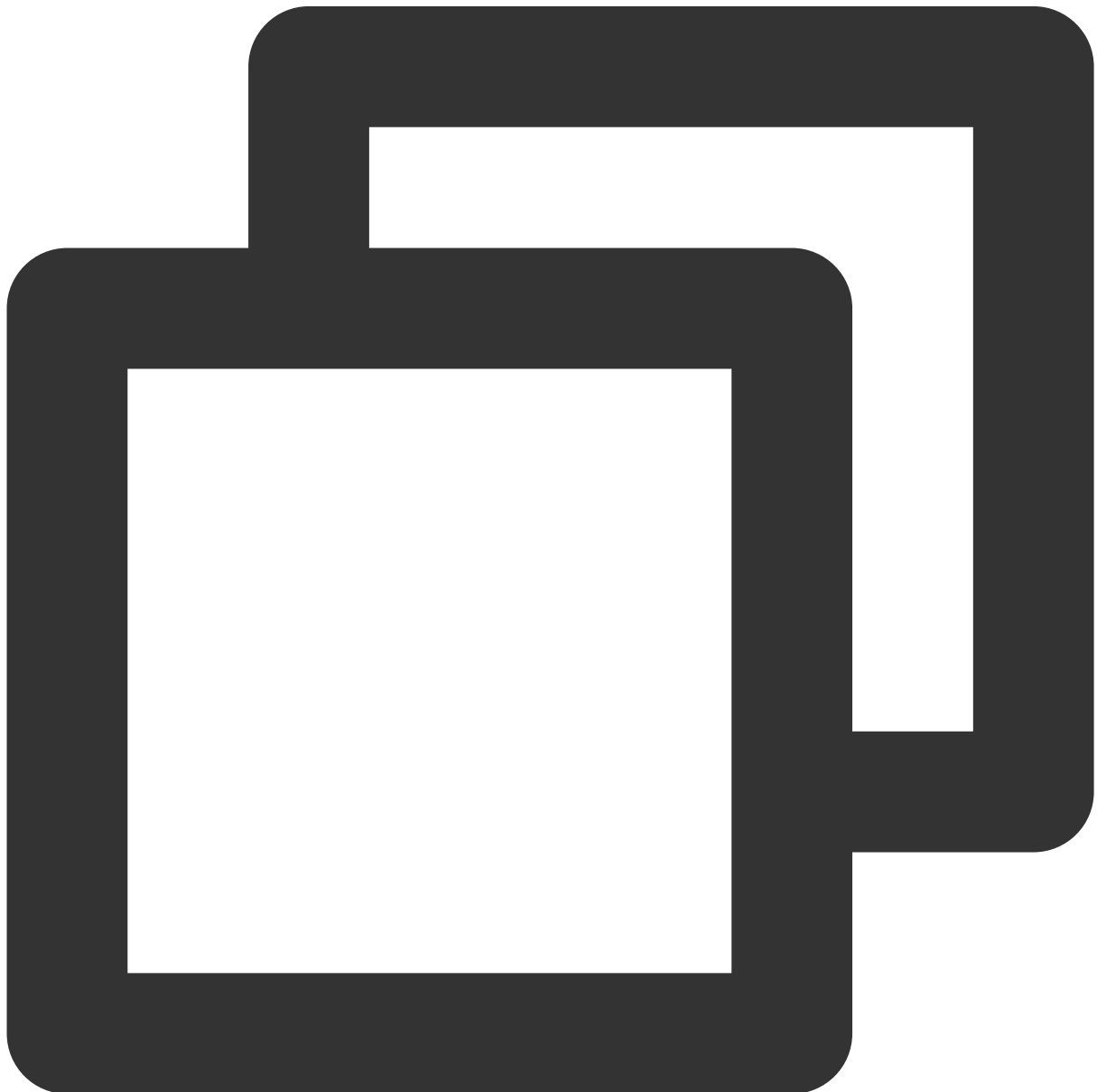
Returns: *void*

## hideFeatureButton

Hide specific feature buttons in the application. By invoking this function and passing in the appropriate [FeatureButton](#) enumerated values, the corresponding buttons will be hidden from the user interface.

The parameters are described as follows:

Parameter	Type	Default Value	Meaning
name	<a href="#">FeatureButton</a>	-	Names of Buttons to be Hidden



```
// Note the package name. If you are using the vue2 version, please change the pack
import { conference, FeatureButton } from '@tencentcloud/roomkit-web-vue3';
```

```
conference.hideFeatureButton (FeatureButton.SwitchTheme);
```

Returns: *void*

## Type Definition

### RoomEvent (enumeration)

Parameter	Type	Description
ROOM_START	string	Creating a Meeting
ROOM_JOIN	string	Joining a Meeting
ROOM_LEAVE	string	Leave Meeting
ROOM_DISMISS	string	Meeting Dismissed
KICKED_OFFLINE	string	User kicked offline
KICKED_OUT	string	Participant removed from meeting
USER_LOGOUT	string	User logged out
ROOM_ERROR	string	Meeting error

### FeatureButton (enumerated values)

Parameter	Type	Description
SwitchTheme	string	Switch Topic Feature Button
SwitchLayout	string	Switch Layout Feature Button
SwitchLanguage	string	Switch Language Feature Button
FullScreen	string	Fullscreen Feature Button
Invitation	string	Invite Feature Button

### StartParams

Parameter	Type	Description	Default Value
roomName	string (Optional)	Room Name	-

isSeatEnabled	boolean (optional)	Enable Seats	false
isOpenCamera	boolean (optional)	Whether to enable the camera	false
isOpenMicrophone	boolean (optional)	Whether to enable the microphone	false
defaultCameraId	string (Optional)	Default Camera ID	-
defaultMicrophoneId	string (Optional)	Default Microphone ID	-
defaultSpeakerId	string (Optional)	Default Speaker ID	-

## JoinParams

Parameter	Type	Description	Default Value
isOpenCamera	boolean (optional)	Whether to enable the camera	false
isOpenMicrophone	boolean (optional)	Whether to enable the microphone	false
defaultCameraId	string (Optional)	Default Camera ID	-
defaultMicrophoneId	string (Optional)	Default Microphone ID	-
defaultSpeakerId	string (Optional)	Default Speaker ID	-

# RoomEngine API

## API Overview

Last updated : 2024-02-29 10:57:30

### TUIRoomEngine (No UI interface)

#### TUIRoomEngine Static method

API	Description
<a href="#">once</a>	<b>Listen to TUIRoomEngine ready event.</b> <b>Note: All methods except TUIRoomEngine.init must be executed after listening to the TUIRoomEngine ready event and the TUIRoomEngine.init method is executed successfully.</b>
<a href="#">login</a>	Login to TUIRoomEngine
<a href="#">setSelfInfo</a>	Set the current user's basic information (username, user avatar)
<a href="#">getSelfInfo</a>	Get the current user's basic information (username, user avatar)
<a href="#">logout</a>	Logout from TUIRoomEngine

#### RoomEngine Room Management API

API	Description
<a href="#">createRoom</a>	Create room
<a href="#">enterRoom</a>	Entered room
<a href="#">destroyRoom</a>	Close the room
<a href="#">exitRoom</a>	Leave room
<a href="#">fetchRoomInfo</a>	Get room data
<a href="#">updateRoomNameByAdmin</a>	Update the room's name (Only group owner or administrator can call)
<a href="#">updateRoomSpeechModeByAdmin</a>	Update the room's speech mode (Only group owner or administrator can call)
<a href="#">getUserList</a>	Get the current room user list

[getUserInfo](#)[Get user](#) [Learn more](#)**roomEngine Audio Video API**

API	Description
<a href="#">setLocalVideoView</a>	Set the view control for local user video rendering
<a href="#">openLocalCamera</a>	Open local camera
<a href="#">closeLocalCamera</a>	Close local camera
<a href="#">openLocalMicrophone</a>	Open local microphone
<a href="#">closeLocalMicrophone</a>	Close local microphone
<a href="#">updateVideoQuality</a>	Update local video codec quality settings
<a href="#">updateAudioQuality</a>	Update local audio codec quality settings
<a href="#">startPushLocalVideo</a>	Start pushing local video
<a href="#">stopPushLocalVideo</a>	Stop pushing local video
<a href="#">startPushLocalAudio</a>	Start pushing local audio
<a href="#">stopPushLocalAudio</a>	Stop pushing local audio
<a href="#">setRemoteVideoView</a>	Set the view control for remote user video rendering
<a href="#">startPlayRemoteVideo</a>	Start playing remote user video
<a href="#">stopPlayRemoteVideo</a>	Stop playing remote user video
<a href="#">muteRemoteAudioStream</a>	Mute remote user

**roomEngine Member Management API**

API	Description
<a href="#">openRemoteDeviceByAdmin</a>	Request remote user to open media device
<a href="#">applyToAdminToOpenLocalDevice</a>	Participant applies to the host to open the device
<a href="#">closeRemoteDeviceByAdmin</a>	Close remote user's media device
<a href="#">cancelRequest</a>	Cancel the sent request

<a href="#">responseRemoteRequest</a>	Reply to remote user's request
<a href="#">changeUserRole</a>	Change user's role
<a href="#">kickRemoteUserOutOfRoom</a>	Kick user out of the room
<a href="#">disableDeviceForAllUserByAdmin</a>	Disable/Enable all users' media devices
<a href="#">disableSendingMessageForAllUser</a>	Disable/Enable all users to send messages
<a href="#">disableSendingMessageByAdmin</a>	Disable/Enable a user to send messages

**roomEngine Screen Sharing API**

API	Description
<a href="#">startScreenSharingElectron</a>	Start screen sharing
<a href="#">stopScreenSharingElectron</a>	End screen sharing
<a href="#">getScreenSharingTarget</a>	Get Screen Sharing List
<a href="#">selectScreenSharingTarget</a>	Switch Screen Sharing Window

**roomEngine Microphone Management API**

API	Description
<a href="#">setMaxSeatCount</a>	Set Room Microphone Maximum
<a href="#">getSeatList</a>	Get Microphone Information
<a href="#">takeSeat</a>	Get Microphone
<a href="#">leaveSeat</a>	Release Microphone
<a href="#">takeUserOnSeatByAdmin</a>	Invite Others to Go Live (Only the Host and Administrator can call this method)
<a href="#">kickUserOffSeatByAdmin</a>	Kick Off Microphone (Only the Host and Administrator can call this method)
<a href="#">lockSeatByAdmin</a>	Lock a Microphone Status (Only the Host and Administrator can call this method)

**roomEngine Message Sending API**

API	Description
<a href="#">sendTextMessage</a>	Send text message

[sendCustomMessage](#)

Send custom message

**roomEngine Device Management API**

API	Description
<a href="#">getCameraDevicesList</a>	Get camera device list
<a href="#">getMicDevicesList</a>	Get mic device list
<a href="#">getSpeakerDevicesList</a>	Get speaker device list
<a href="#">setCurrentCameraDevice</a>	Set the camera device to use
<a href="#">setCurrentMicDevice</a>	Set the mic device to use
<a href="#">setCurrentSpeakerDevice</a>	Set the speaker device to use
<a href="#">getCurrentCameraDevice</a>	Get the currently used camera device
<a href="#">getCurrentMicDevice</a>	Get the currently used mic device
<a href="#">getCurrentSpeakerDevice</a>	Get the currently used speaker device
<a href="#">startCameraDeviceTest</a>	Start camera device test
<a href="#">stopCameraDeviceTest</a>	Stop camera device test

**roomEngine Event Listening API**

API	Description
<a href="#">on</a>	Listen to <a href="#">TUIRoomEvents</a> event
<a href="#">off</a>	Cancel listening to <a href="#">TUIRoomEvents</a> event

**roomEngine Other API**

API	Description
<a href="#">getTRTCCloud</a>	Get trtcCloud instance
<a href="#">getTIM</a>	Get tim instance

## Event Type Definition



TUIRoomEvent is the Callback Event class corresponding to TUIRoomEngine. You can listen to Callback Events of interest through this callback.

EVENT	Description
<a href="#">TUIRoomEvents.onError</a>	Error Event
<a href="#">TUIRoomEvents.onKickedOutOfRoom</a>	Kick out of room event
<a href="#">TUIRoomEvents.onKickedOffline</a>	User Kicked Offline Event
<a href="#">TUIRoomEvents.onUserSigExpired</a>	User Credential Timeout Event
<a href="#">TUIRoomEvents.onRoomDismissed</a>	Room Dismissed Event
<a href="#">TUIRoomEvents.onRoomNameChanged</a>	Room Name Change Event
<a href="#">TUIRoomEvents.onRoomSpeechModeChanged</a>	Room Microphone Control Mode Change
<a href="#">TUIRoomEvents.onAllUserCameraDisableChanged</a>	All Users' Cameras Disabled in Room Event
<a href="#">TUIRoomEvents.onAllUserMicrophoneDisableChanged</a>	All Users' Microphones Disabled in Room Event
<a href="#">TUIRoomEvents.onSendMessageForAllUserDisableChanged</a>	All Users' Text Message Sending Disabled in Room Event
<a href="#">TUIRoomEvents.onRoomMaxSeatCountChanged</a>	Maximum number of microphone slots in the room modification event
<a href="#">TUIRoomEvents.onRemoteUserEnterRoom</a>	Remote user Entered room event
<a href="#">TUIRoomEvents.onRemoteUserLeaveRoom</a>	Remote user leaving room event
<a href="#">TUIRoomEvents.onUserRoleChanged</a>	Role change event
<a href="#">TUIRoomEvents.onUserVideoStateChanged</a>	Video Status change event
<a href="#">TUIRoomEvents.onUserAudioStateChanged</a>	Audio Status change event
<a href="#">TUIRoomEvents.onSendMessageForUserDisableChanged</a>	Send message status event
<a href="#">TUIRoomEvents.onUserVoiceVolumeChanged</a>	Volume change event
<a href="#">TUIRoomEvents.onUserNetworkQualityChanged</a>	Network quality change event
<a href="#">TUIRoomEvents.onSeatListChanged</a>	Mic position list change event
<a href="#">TUIRoomEvents.onKickedOffSeat</a>	Kicked off the mic event

<a href="#">TUIRoomEvents.onRequestReceived</a>	Request received event
<a href="#">TUIRoomEvents.onRequestCancelled</a>	Request cancelled event
<a href="#">TUIRoomEvents.onReceiveTextMessage</a>	Receive text message event
<a href="#">TUIRoomEvents.onReceiveCustomMessage</a>	Receive custom message event
<a href="#">TUIRoomEvents.onDeviceChange</a>	Device change event
<a href="#">TUIRoomEvents.onUserScreenCaptureStopped</a>	<p>Screen sharing stopped event</p> <p>When a user uses the built-in browser's stop sharing button to end Screen Sharing, the user will receive the 'onUserScreenCaptureStopped' event to modify the Sharing status.</p>

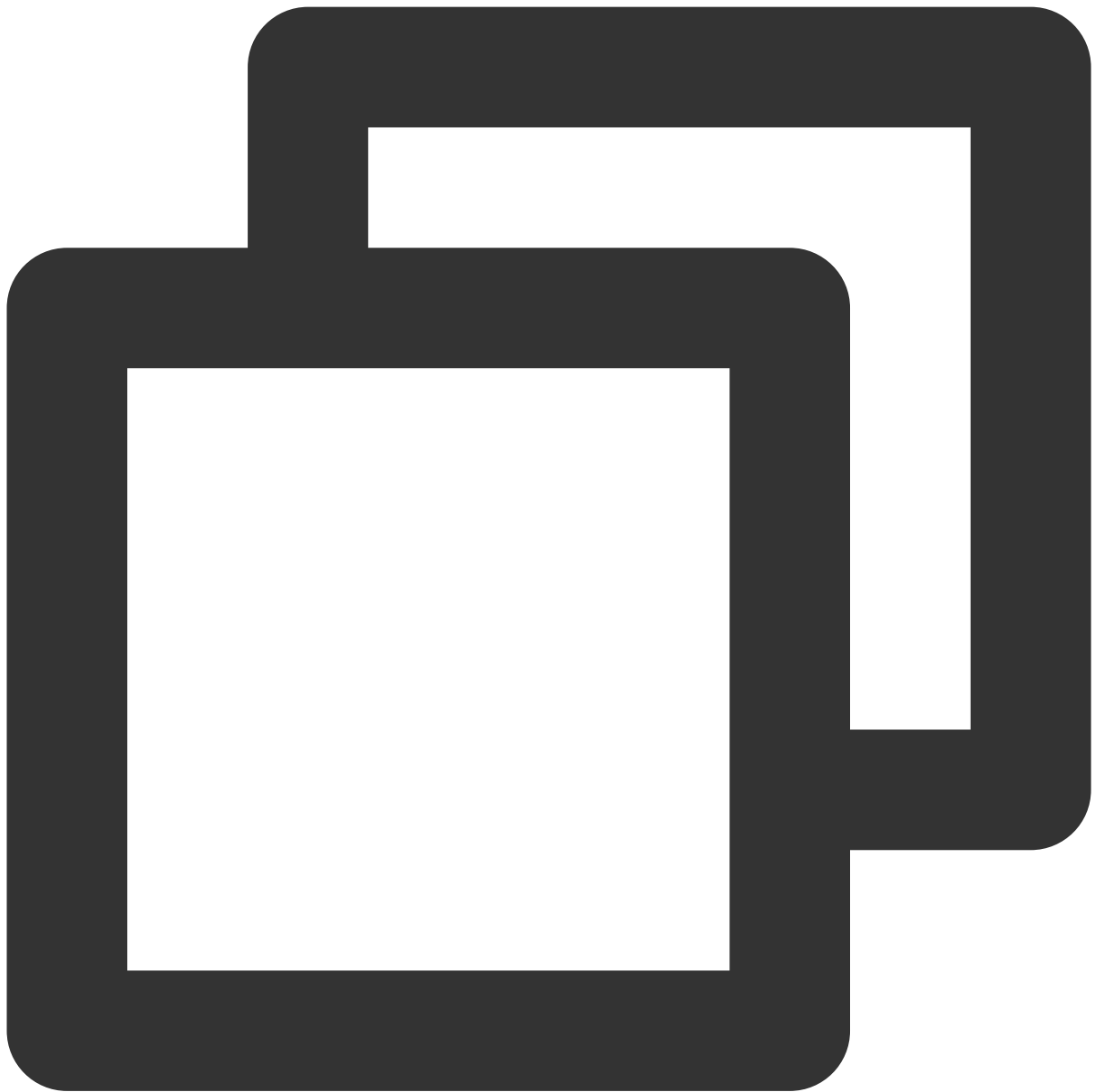
# TUIRoomEngine

Last updated : 2023-11-16 14:46:30

## TUIRoomEngine Introduction

TUIRoomEngine SDK provides Room Management, Multi-person Tencent Real-Time Communication, Screen Sharing, Member Management, Instant Messaging, and other features.

### **Installation Method:**



```
// Use npm
npm i @tencentcloud/tuiroom-engine-electron --save

// Use pnpm
pnpm i @tencentcloud/tuiroom-engine-electron --save

// Use yarn
yarn add @tencentcloud/tuiroom-engine-electron
```

# TUIRoomEngine API

## TUIRoomEngine Static Method

API	Description
<a href="#">once</a>	<b>Listening to the TUIRoomEngine ready Event.</b> <b>Note: All methods other than TUIRoomEngine.login must be executed after listening to the TUIRoomEngine ready event and the successful execution of the TUIRoomEngine.login method.</b>
<a href="#">login</a>	Login to TUIRoomEngine
<a href="#">setSelfInfo</a>	Setting the current user's Basic information (Username, User Avatar)
<a href="#">getSelfInfo</a>	Get the current user's Basic information (Username, User Avatar)
<a href="#">logout</a>	Logout of TUIRoomEngine

## roomEngine Room Management API

API	Description
<a href="#">createRoom</a>	Create Room
<a href="#">enterRoom</a>	Enter Room
<a href="#">destroyRoom</a>	Destroy Room
<a href="#">exitRoom</a>	Exit Room
<a href="#">fetchRoomInfo</a>	Get Room data
<a href="#">updateRoomNameByAdmin</a>	Update Name (Call by Group Owner or Administrator only)
<a href="#">updateRoomSpeechModeByAdmin</a>	Update Speaking Mode (Call by Group Owner or Administrator only)
<a href="#">getUserList</a>	Get User List
<a href="#">getUserInfo</a>	Learn more about the user

## roomEngine Audio Video API

API	Description
<a href="#">setLocalVideoView</a>	Set Local Stream Rendering Position

<a href="#">openLocalCamera</a>	Capturing Local Camera Video streams
<a href="#">closeLocalCamera</a>	Close Local Camera
<a href="#">openLocalMicrophone</a>	Open Local mic
<a href="#">closeLocalMicrophone</a>	Close Local mic
<a href="#">updateVideoQuality</a>	Set Local Video Parameters
<a href="#">updateAudioQuality</a>	Set Local Audio Parameters
<a href="#">startPushLocalVideo</a>	Start Pushing Local Video streams to Remote
<a href="#">stopPushLocalVideo</a>	Stop Pushing Local Video streams to Remote
<a href="#">startPushLocalAudio</a>	Start Pushing Local Audio Stream to Remote
<a href="#">stopPushLocalAudio</a>	Stop Pushing Local Audio Stream to Remote
<a href="#">setRemoteVideoView</a>	Set Remote Stream Rendering Area
<a href="#">startPlayRemoteVideo</a>	Start Playback Remote User Video streams
<a href="#">stopPlayRemoteVideo</a>	Stop Playback Remote User Video streams
<a href="#">muteRemoteAudioStream</a>	Stop Remote User Audio Stream

## roomEngine Member Management API

API	Description
<a href="#">openRemoteDeviceByAdmin</a>	Request Remote User to Open Media Device
<a href="#">applyToAdminToOpenLocalDevice</a>	Participant Apply to Host to Open Device
<a href="#">closeRemoteDeviceByAdmin</a>	Close Remote User Media Device
<a href="#">cancelRequest</a>	Cancel Sent Request
<a href="#">responseRemoteRequest</a>	Reply to Remote User Request
<a href="#">changeUserRole</a>	Change User Role
<a href="#">kickRemoteUserOutOfRoom</a>	Kick Out User from Room
<a href="#">disableDeviceForAllUserByAdmin</a>	Disable/Enable All Users' Media Device
<a href="#">disableSendingMessageForAllUser</a>	Disallow/Allow All Users to Send Message

[disableSendingMessageByAdmin](#)

Disallow/Allow Specific User to Send Message

## roomEngine Screen Sharing API

API	Description
<a href="#">startScreenSharingElectron</a>	Start Screen Sharing
<a href="#">stopScreenSharingElectron</a>	Stop Screen Sharing
<a href="#">getScreenSharingTarget</a>	Obtain Screen Sharing List
<a href="#">selectScreenSharingTarget</a>	Switch Screen Sharing Window

## roomEngine Mic Position Management API

API	Description
<a href="#">setMaxSeatCount</a>	Set Room Maximum Value
<a href="#">getSeatList</a>	Get Mic Position Information
<a href="#">takeSeat</a>	Get Mic Position
<a href="#">leaveSeat</a>	Release Mic Position
<a href="#">takeUserOnSeatByAdmin</a>	Invite Others to Go Live (Only Room Host and Administrator can invoke this method)
<a href="#">kickUserOffSeatByAdmin</a>	Kick Others Off the Mic (Only Room Host and Administrator can invoke this method)
<a href="#">lockSeatByAdmin</a>	Lock a Specific Mic Position Status (Only Room Host and Administrator can invoke this method)

## roomEngine Message Sending API

API	Description
<a href="#">sendTextMessage</a>	Send Text Message
<a href="#">sendCustomMessage</a>	Send Custom Message

## roomEngine Device Management API

--	--

API	Description
<a href="#">getCameraDevicesList</a>	Get Camera Device List
<a href="#">getMicDevicesList</a>	Get Mic Device List
<a href="#">getSpeakerDevicesList</a>	Get Speaker Device List
<a href="#">setCurrentCameraDevice</a>	Set to Use Camera
<a href="#">setCurrentMicDevice</a>	Set to Use Mic
<a href="#">setCurrentSpeakerDevice</a>	Set to Use Speaker
<a href="#">getCurrentCameraDevice</a>	Get the Currently Used Camera
<a href="#">getCurrentMicDevice</a>	Get the Currently Used Mic
<a href="#">getCurrentSpeakerDevice</a>	Get the Currently Used Speaker
<a href="#">startCameraDeviceTest</a>	Start Camera Test
<a href="#">stopCameraDeviceTest</a>	Stop Camera Test

## roomEngine Event Listening API

API	Description
<a href="#">on</a>	Listen to <a href="#">TUIRoomEvents</a>
<a href="#">off</a>	Cancel Listening to <a href="#">TUIRoomEvents</a>

## roomEngine Other API

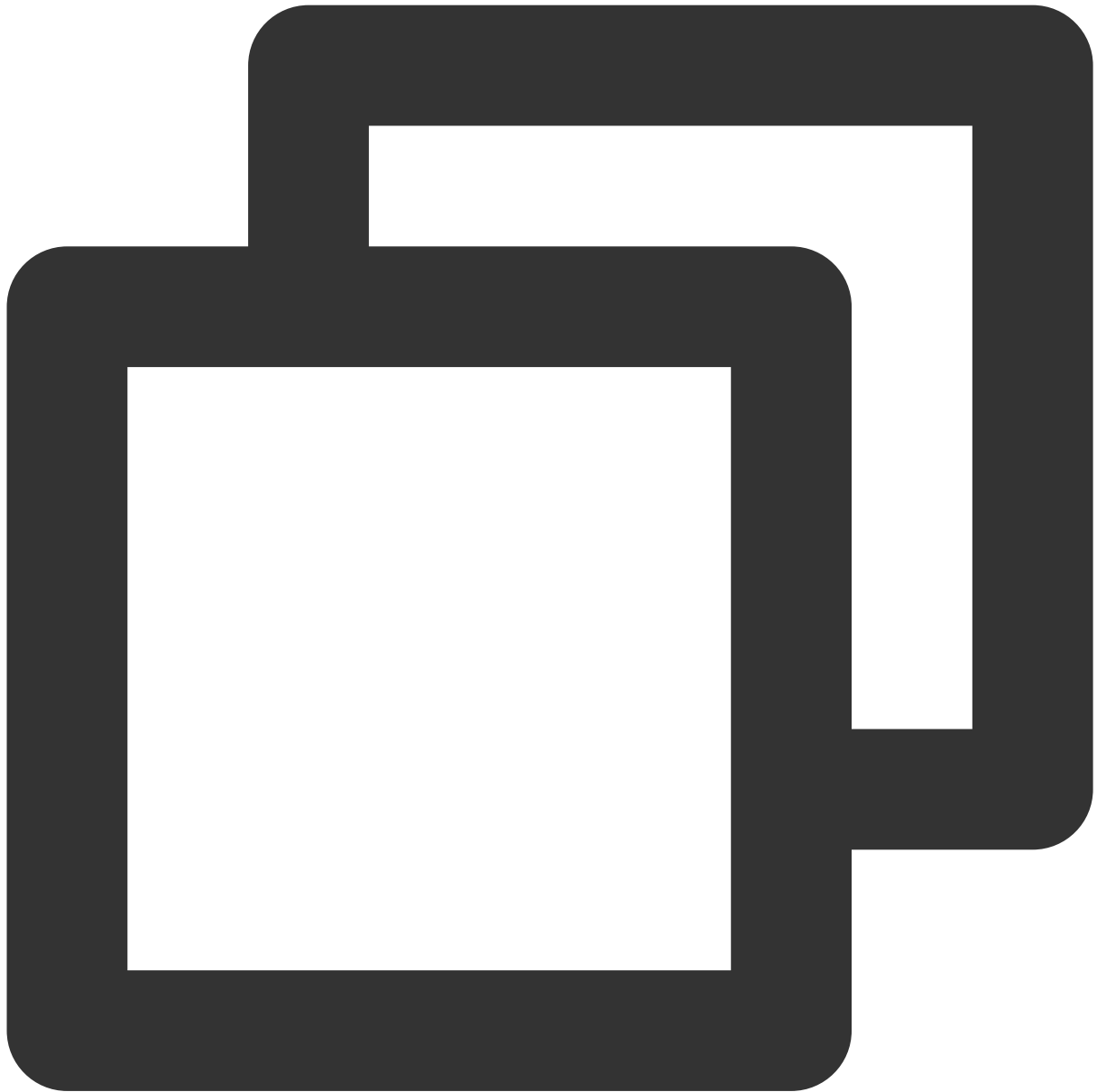
API	Description
<a href="#">getTRTCCloud</a>	Get trtcCloud Instance
<a href="#">getTIM</a>	Get tim Instance

## API Details

### once

Monitor TUIRoomEngine 'ready' Event





```
TUIRoomEngine.once('ready', () => {
  const roomEngine = new TUIRoomEngine();

  await TUIRoomEngine.init({
    sdkAppId: 0,    // Fill in the sdkAppId you applied for
    userId: '',     // Fill in the userId corresponding to your business
    userSig: '',    // Fill in the userSig calculated by the server or locally
  });

  await roomEngine.createRoom({
    roomId: '12345', // Enter your Room ID, note that the Room ID is required to
```

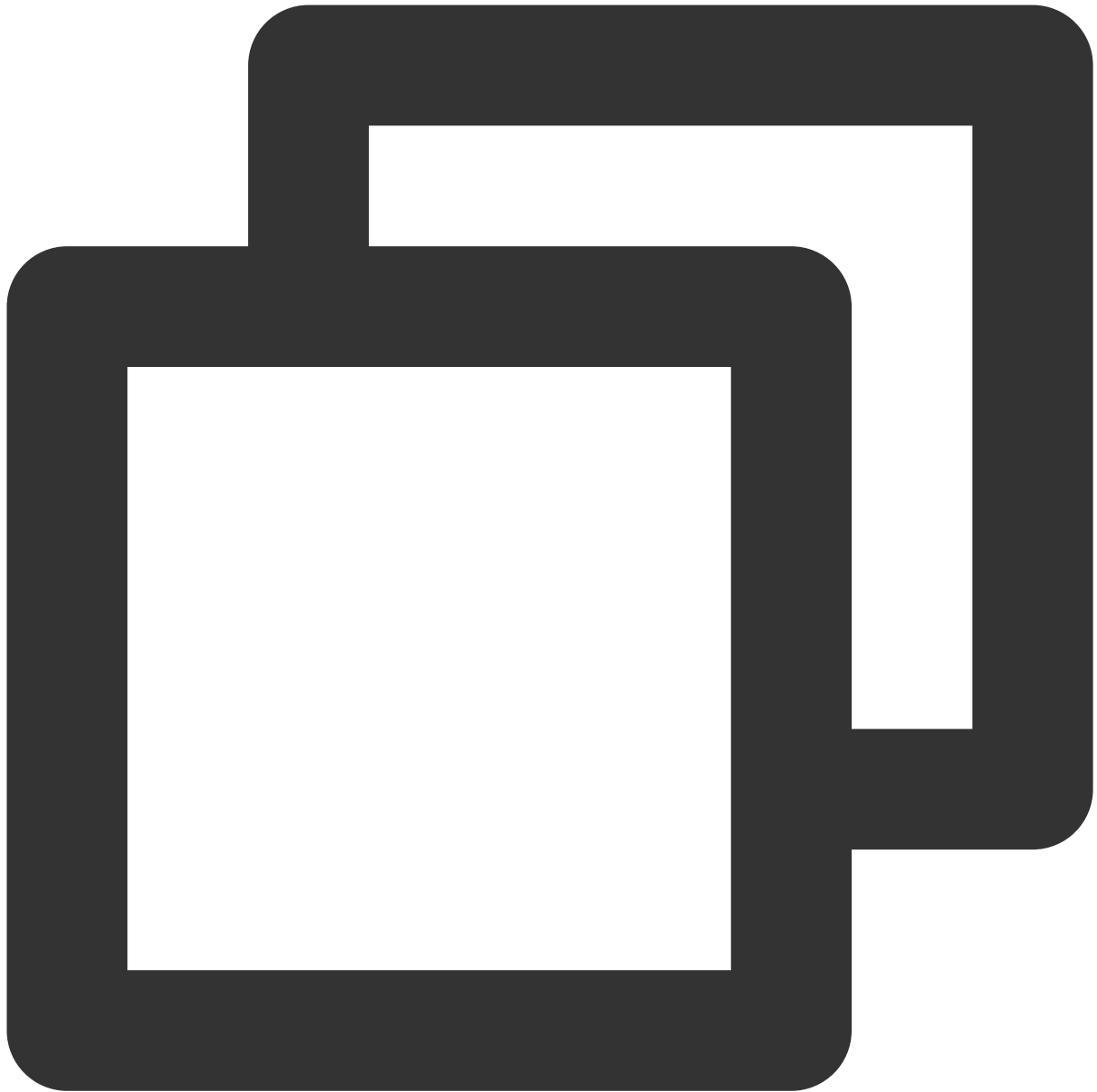
```
name: 'Test Room',      // Enter your room name, the default room name is roomId
roomType: TUIRoomType.kGroup, // Set the room type to TUIRoomType.kGroup type
});
});
```

## login

### Description:

In version v1.0.0, this interface is named TUIRoomEngine.init, please use TUIRoomEngine.login to log in TUIRoomEngine in v1.0.1 and above versions.

You must log in to TUIRoomEngine before you can call other methods of TUIRoomEngine and its instances.



```
// Login TUIRoomEngine
await TUIRoomEngine.login({
  sdkAppId: 0,    // Fill in the sdkAppId you applied for
  userId: '',     // Fill in the userId corresponding to your business
  userSig: '',    // Fill in the userSig calculated by the server or locally
});
```

Parameter:

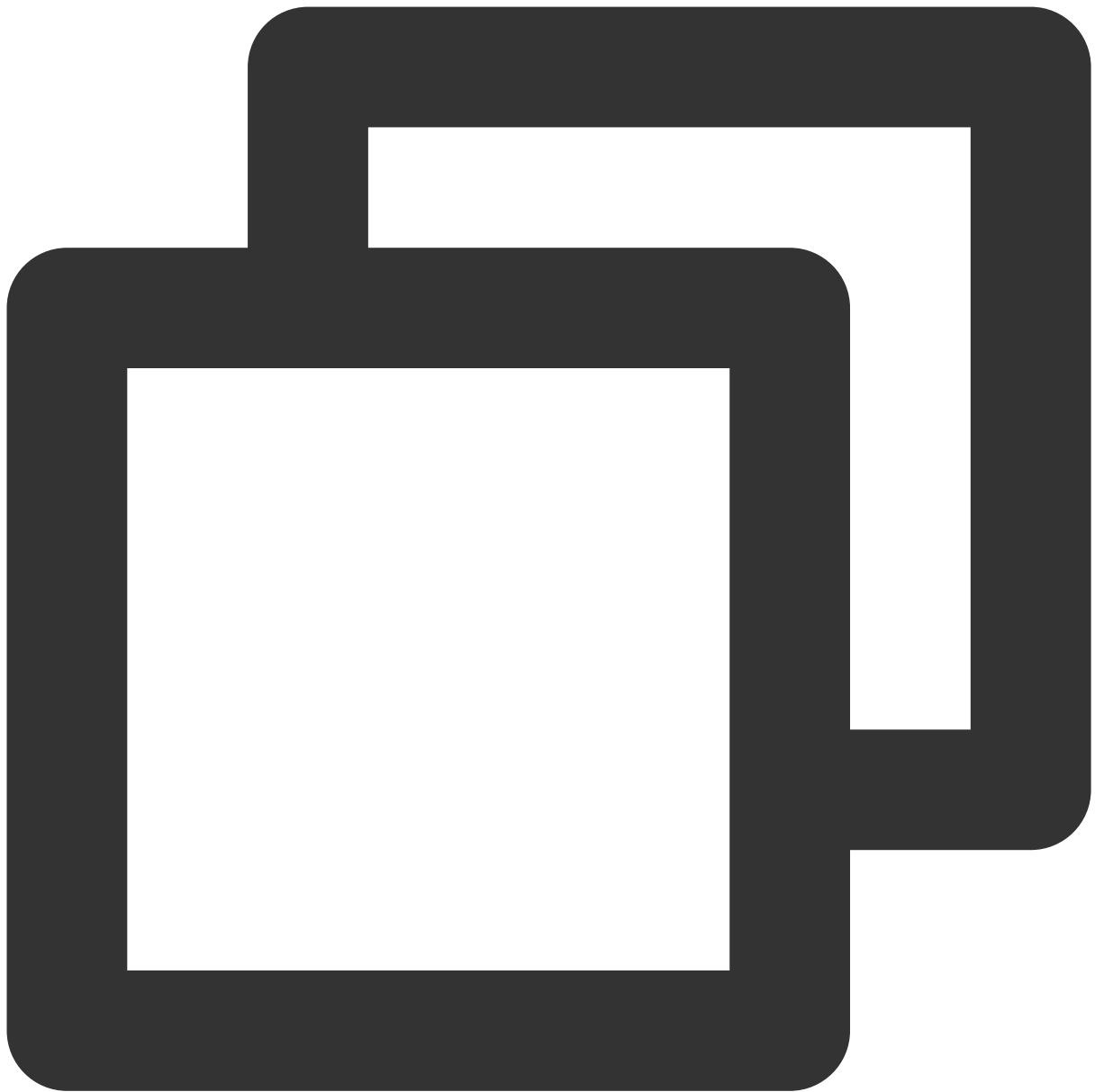
Parameter	Type	Description	Default Value	Meaning

sdkAppld	number	Required	-	After clicking Application Management > Create Application in the Tencent Real-Time Communication Console, you can get the sdkAppld information in Application Info.
userId	string	Required	-	It is recommended to limit the user ID length to 32 bytes, and only allow uppercase and lowercase English letters (a-zA-Z), numbers (0-9), underscores, and hyphens.
userSig	string	Required	-	UserSig signature Please refer to <a href="#">UserSig related</a> for the method of calculating userSig.
tim	TIM	Not Required	-	If you want to use more capabilities of the Chat SDK while accessing roomEngine, you can pass the created tim instance into TUIRoomEngine. For the creation method of tim instance, please refer to <a href="#">TIM.create</a> .

**Returns** *Promise<void>*

### setSelfInfo

Set current user Basic information (Username, User avatar).



```
// Set current user Username and User avatar
await TUIRoomEngine.setSelfInfo({
  userName: '',      // Enter your New username
  avatarUrl: '',     // Enter your New avatar URL
});
```

**Parameter:**

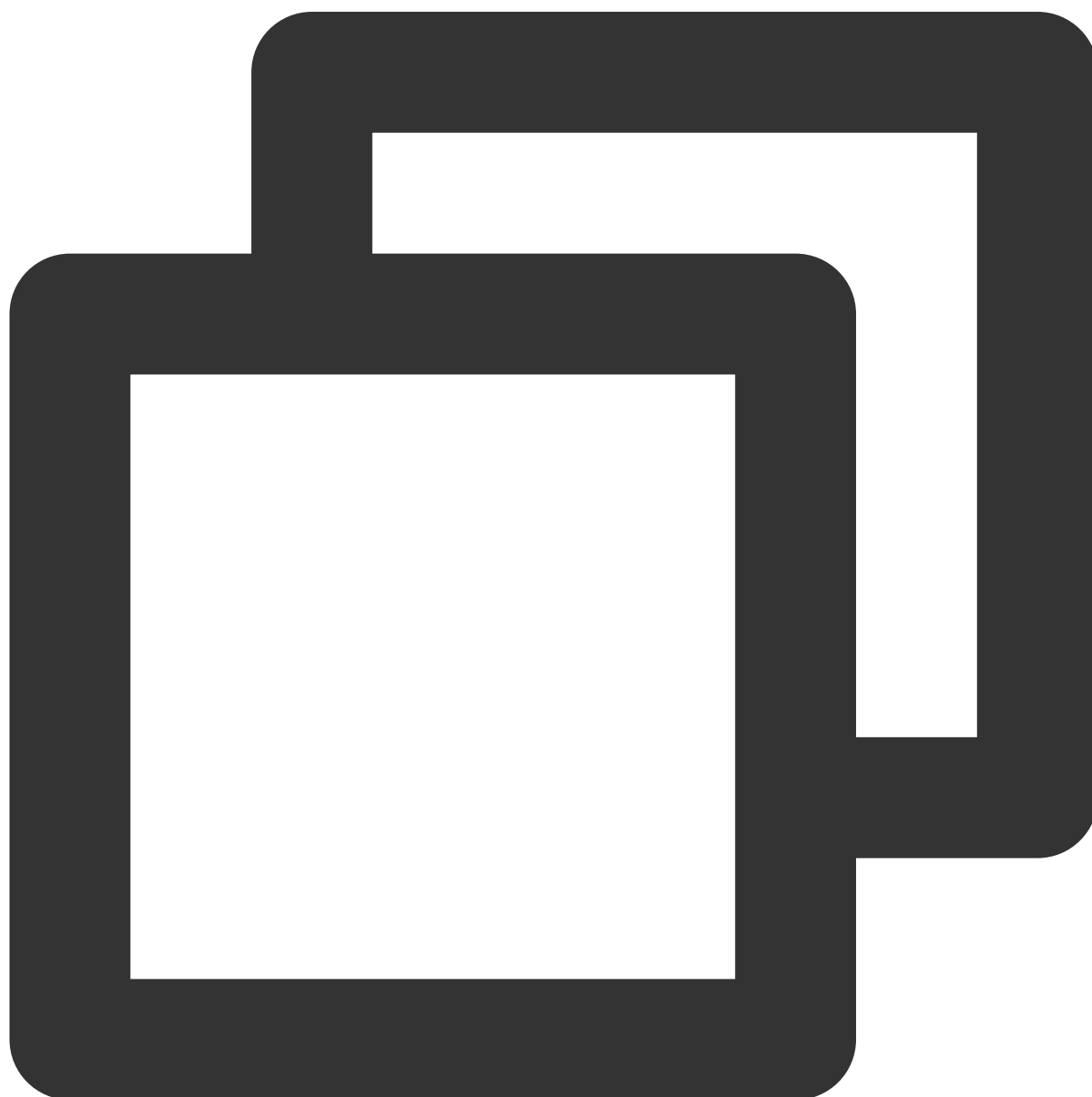
Parameter	Type	Description	Default Value	Meaning

userName	string	Required	-	User Name
avatarUrl	string	Required	-	User avatar

**Returns** *Promise<void>*

## getSelfInfo

Get current user Basic information (Username, User avatar).



```
// Get current user Username and User avatar
```

```
const loginUserInfo = await TUIRoomEngine.getSelfInfo();
```

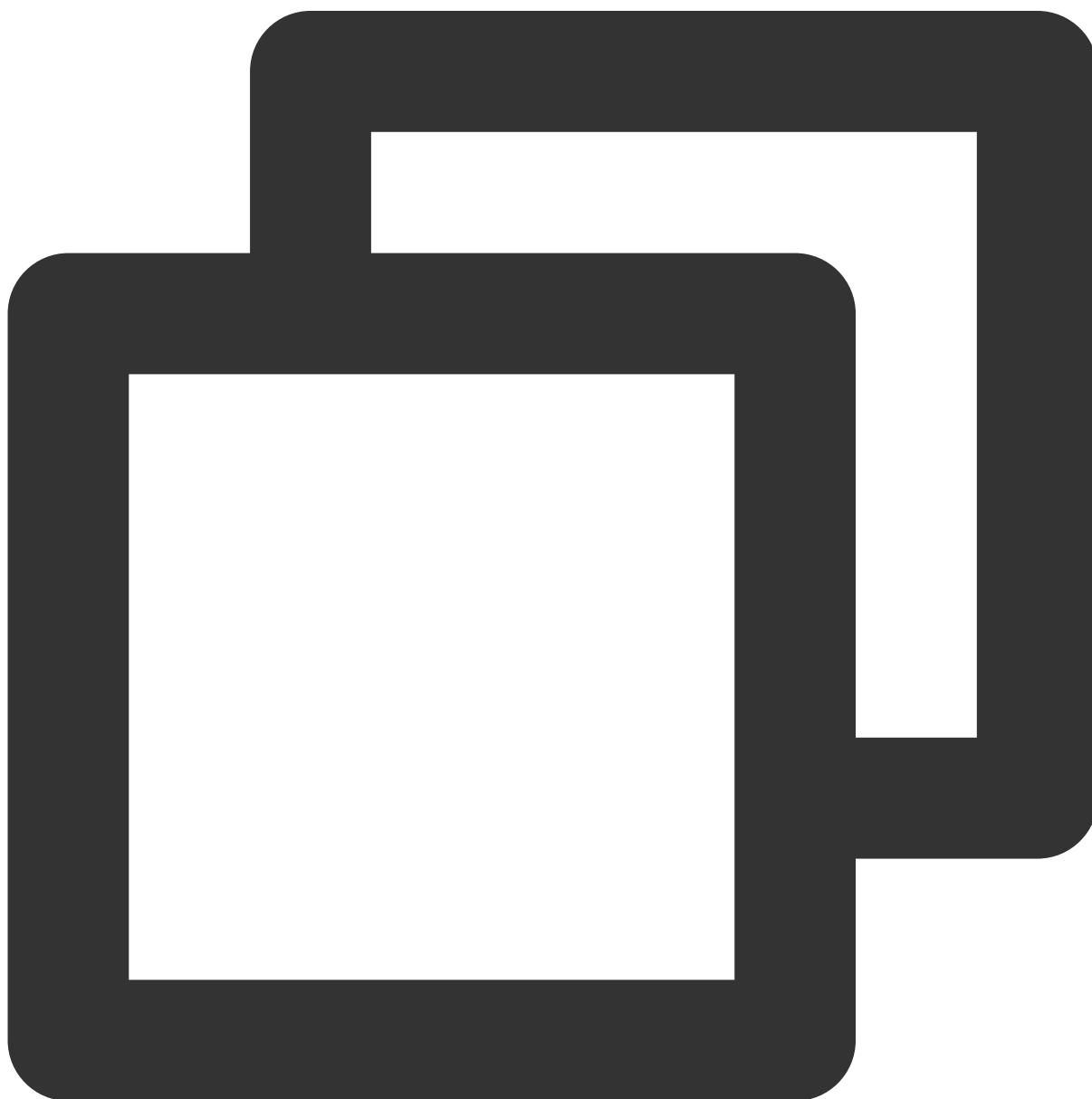
Returns *Promise*<[TUILoginUserInfo](#)> *loginUserInfo*

## logout

### Description:

Interface since v1.0.1 Version support.

Logout TUIRoomEngine



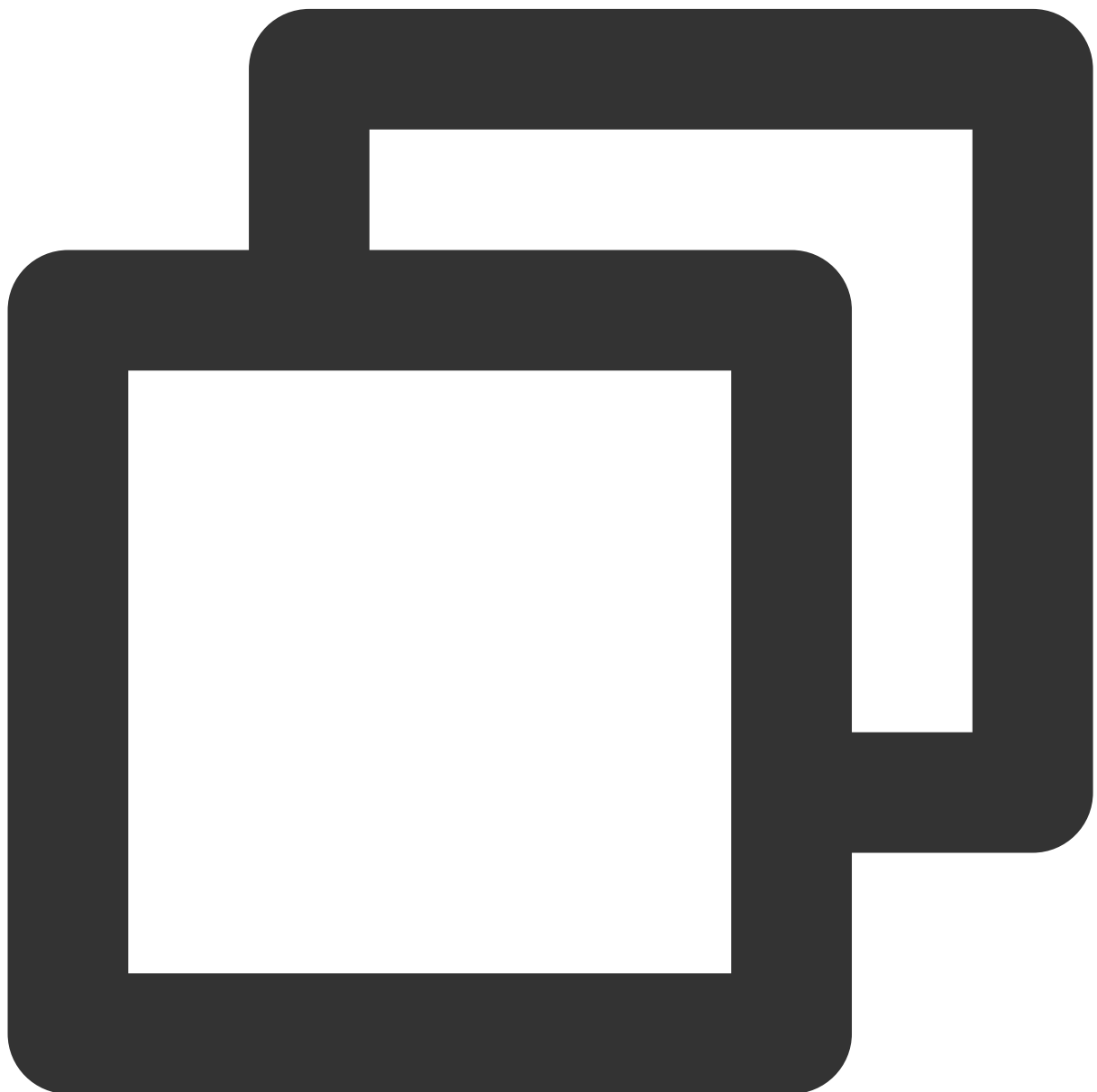
```
// Logout TUIRoomEngine
```

```
await TUIRoomEngine.logout();
```

Returns *Promise<void>*

## createRoom

Host creates room, Call createRoom User is the room owner. When creating a room, you can set Room ID, Room name, Room type, Speaking mode, Whether to allow users to join and enable audio and video, Send message and other functions.



```
const roomEngine = new TUIRoomEngine();
```



```
await roomEngine.createRoom({
  roomId: '12345',    // Enter your Room ID, note that the Room ID must be a string
  roomName: 'Test Room',    // Enter your Room Name, the default Room Name is roomId
  roomType: TUIRoomType.kConference, // Set the Room Type to TUIRoomType.kConference
  speechMode: TUISpeechMode.kFreeToSpeak, // Set the speech mode to free speech mode
  isMicrophoneDisableForAllUser: false, // Allow users to turn on their mic when
  isCameraDisableForAllUser: false, // Allow users to turn on their Camera when
  isMessageDisableForAllUser: false, // Allow users to send messages when joining
});
```

## Parameter:

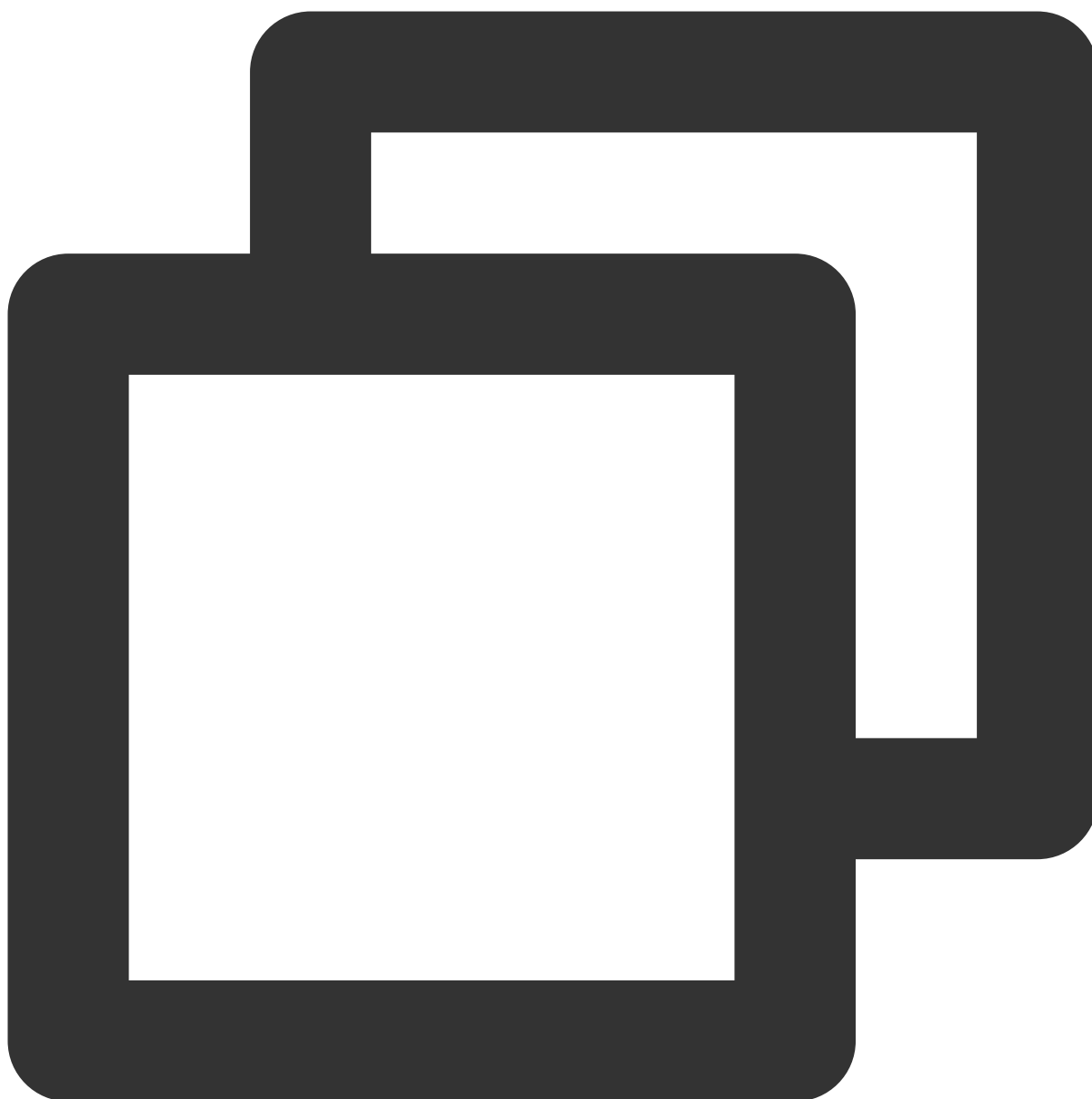
Parameter	Type	Description	Default Value	Meaning
roomId	string	Required	-	Room ID, a byte character string, 1-32 characters in length, must start with a letter, and can contain letters, digits, underscores, and hyphens.
roomName	string	Optional	roomId	Room Name, a byte character string, 1-32 characters in length, must start with a letter, and can contain letters, digits, underscores, and hyphens.
roomType	<a href="#">TUIRoomType</a>	Optional	TUIRoomType.kConference	Room Type, a byte character string, 1-32 characters in length, must start with a letter, and can contain letters, digits, underscores, and hyphens. The default value is TUIRoomType.kConference.
speechMode	TUISpeechMode	Optional	TUISpeechMode.kFreeToSpeak	Speech Mode, a byte character string, 1-32 characters in length, must start with a letter, and can contain letters, digits, underscores, and hyphens. The default value is TUISpeechMode.kFreeToSpeak.

				and mod Set : TUI: user get p and For` broa Set : TUI: for h Set : TUI: host TUI: TUI: mod
isMicrophoneDisableForAllUser	boolean	Optional	false	Enal mute
isCameraDisableForAllUser	boolean	Optional	false	Enal not e
isMessageDisableForAllUser	boolean	Optional	false	Allov proh
maxSeatCount	number	Optional	-	Max For` (edu is nc For` broa max
enableCDNStreaming	boolean	Optional	false	Enal
cdnStreamDomain	string	Optional	"	Pusl

Returns *Promise<void>*

## enterRoom

Entered room interface.



```
const roomEngine = new TUIRoomEngine();
const roomInfo = await roomEngine.enterRoom({
  roomId: '12345',
});
```

Parameter:

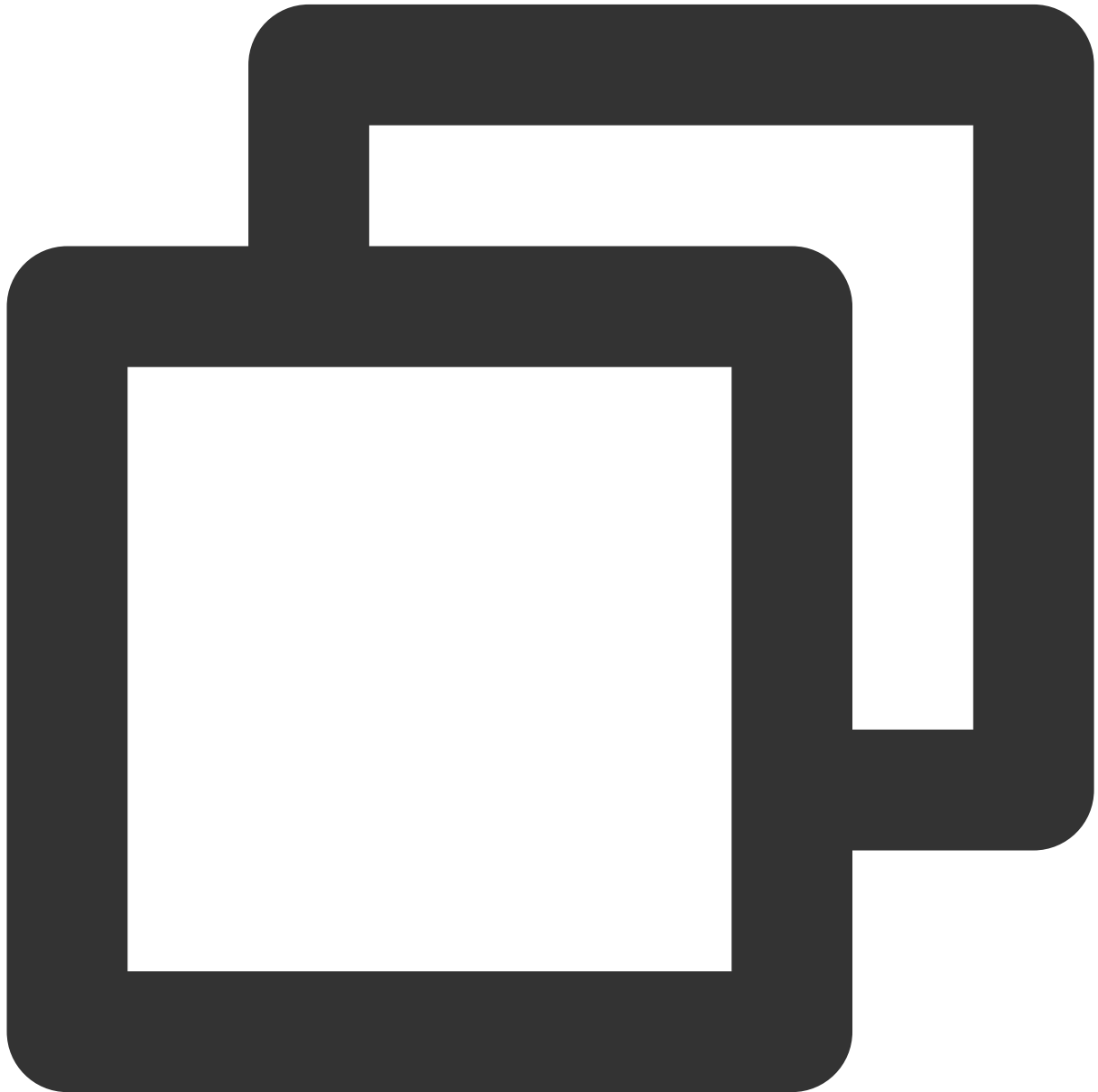
Parameter	Type	Description	Default Value	Meaning
roomId	string	Required	-	Room ID

**Returns** *Promise*<[TUIRoomInfo](#)> *roomInfo*

This interface returns the current Room data

## **destroyRoom**

Close the room interface, the room must be closed by the room owner, and the room cannot be entered after it is closed.

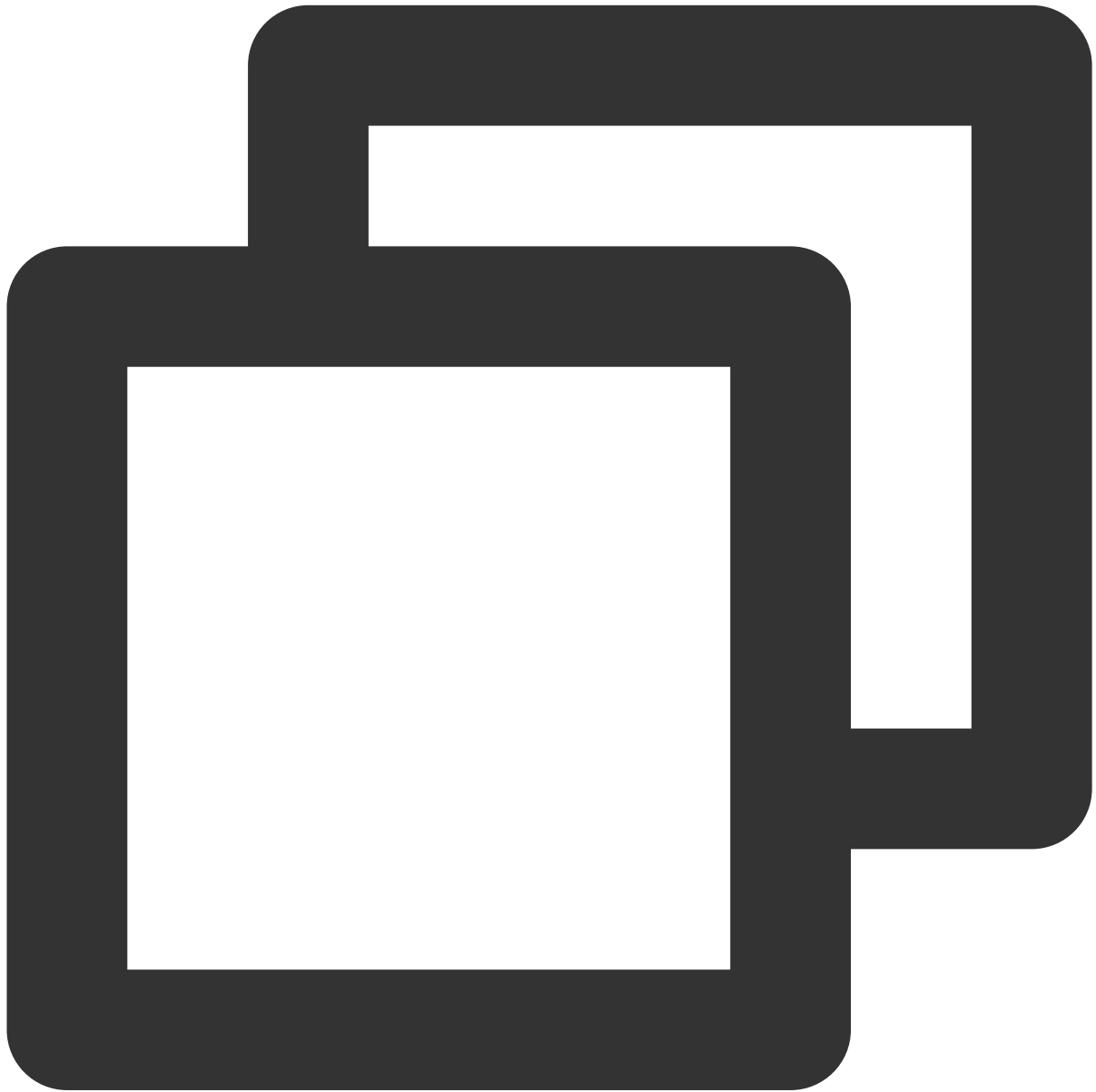


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.destroyRoom();
```

**Returns** *Promise<void>*

## exitRoom

Leave the room interface, users can leave the room through exitRoom after executing enterRoom.

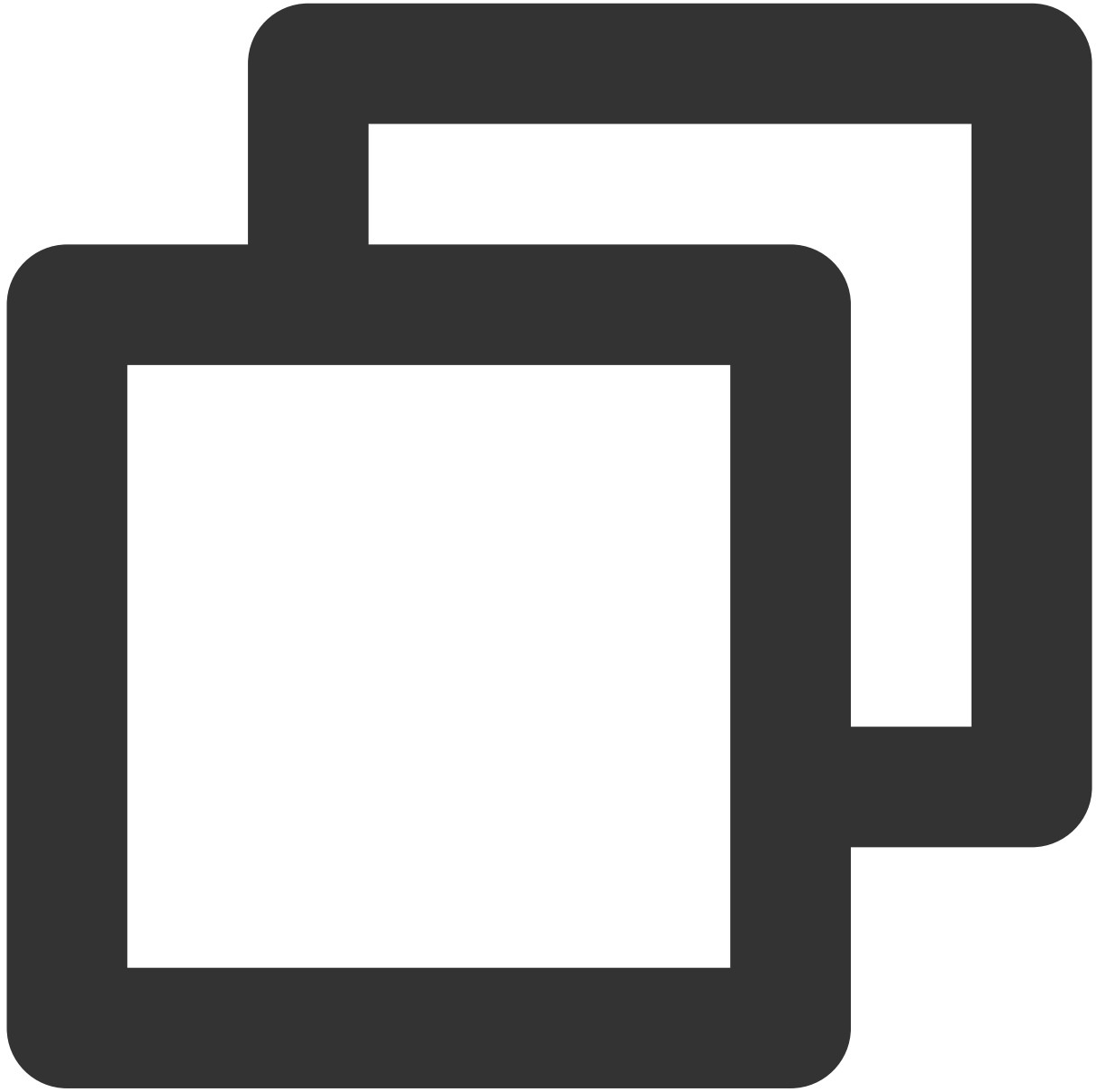


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.exitRoom();
```

**Returns** *Promise<void>*

## fetchRoomInfo

Get Room information.

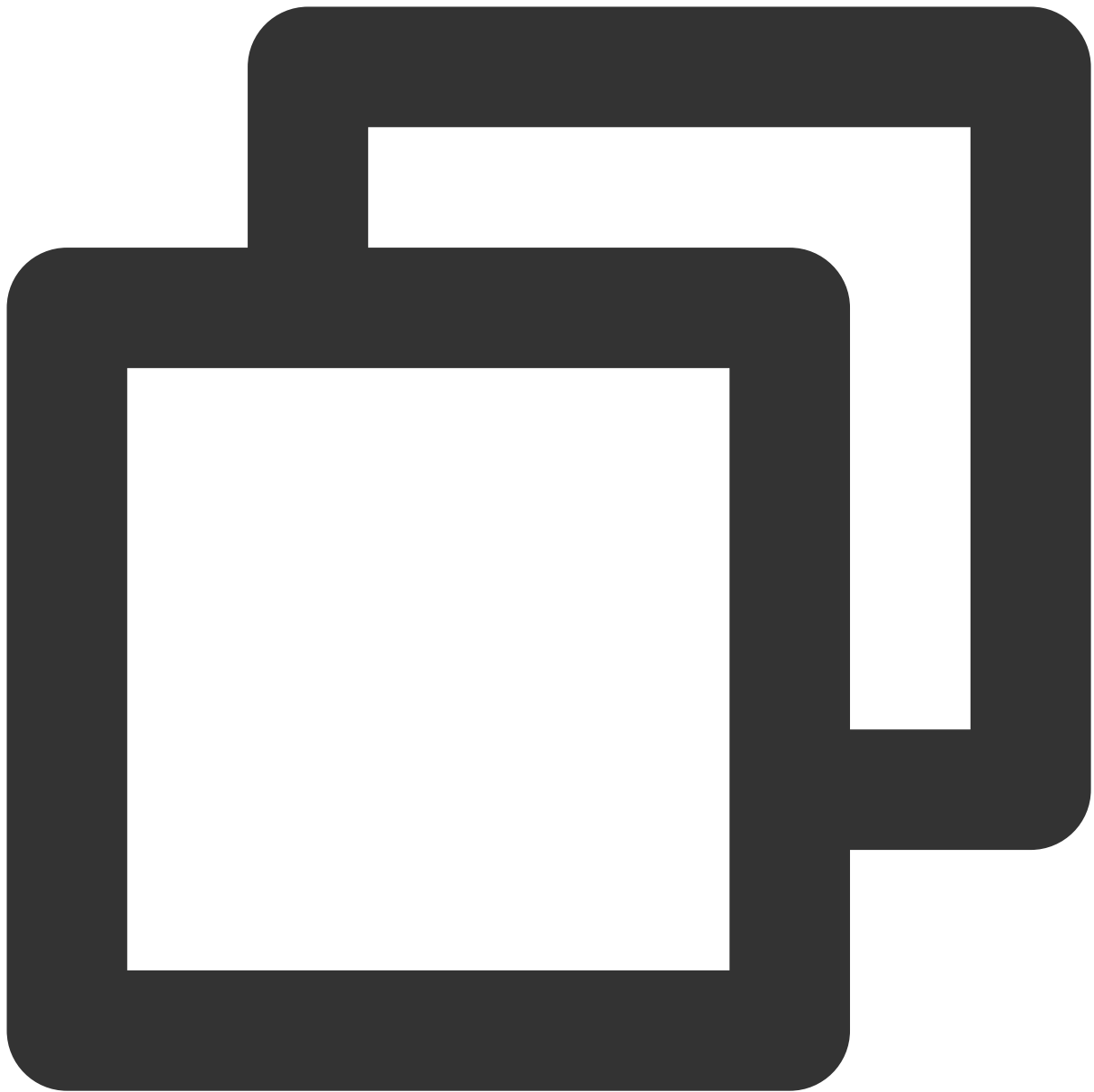


```
const roomEngine = new TUIRoomEngine();  
const roomInfo = roomEngine.fetchRoomInfo();
```

Returns : *Promise*<[TUIRoomInfo](#)> *roomInfo*

### **updateRoomNameByAdmin**

Update the current room's name (only group owner or admin can invoke).



```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomNameByAdmin({ roomName: 'new name' });
```

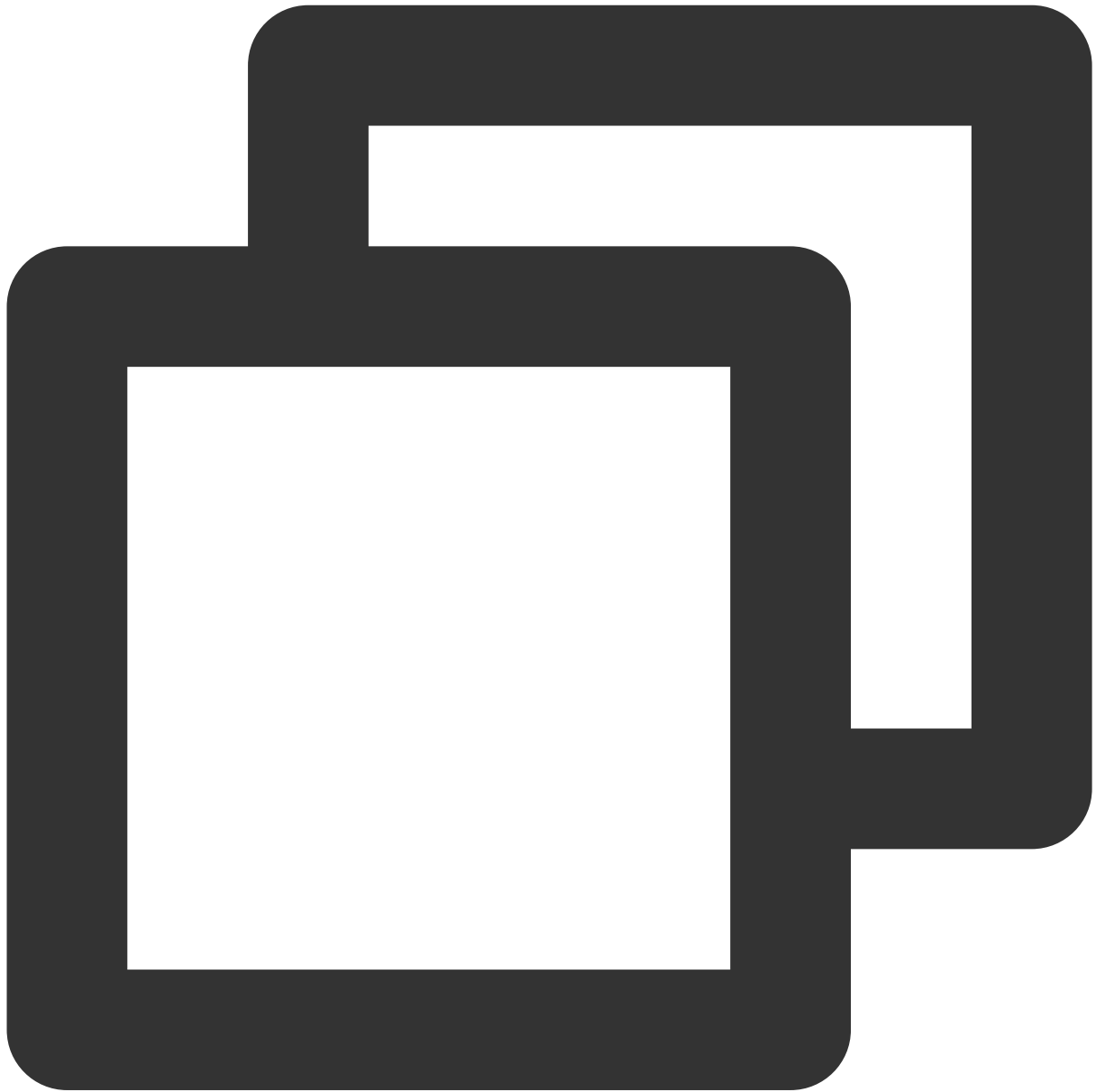
Parameter:

Parameter	Type	Description	Default Value	Meaning
roomName	string	Required	-	Update the room's name, with the requirement that roomName is not an empty string

**Returns** *Promise<void>*

### **updateRoomSpeechModeByAdmin**

Update the room's speaking mode (only group owner or admin can invoke).



```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSpeechModeByAdmin({
  speechMode: TUISpeechMode.kSpeakAfterTakingSeat // Update to Go Live Speaking mo
});
```

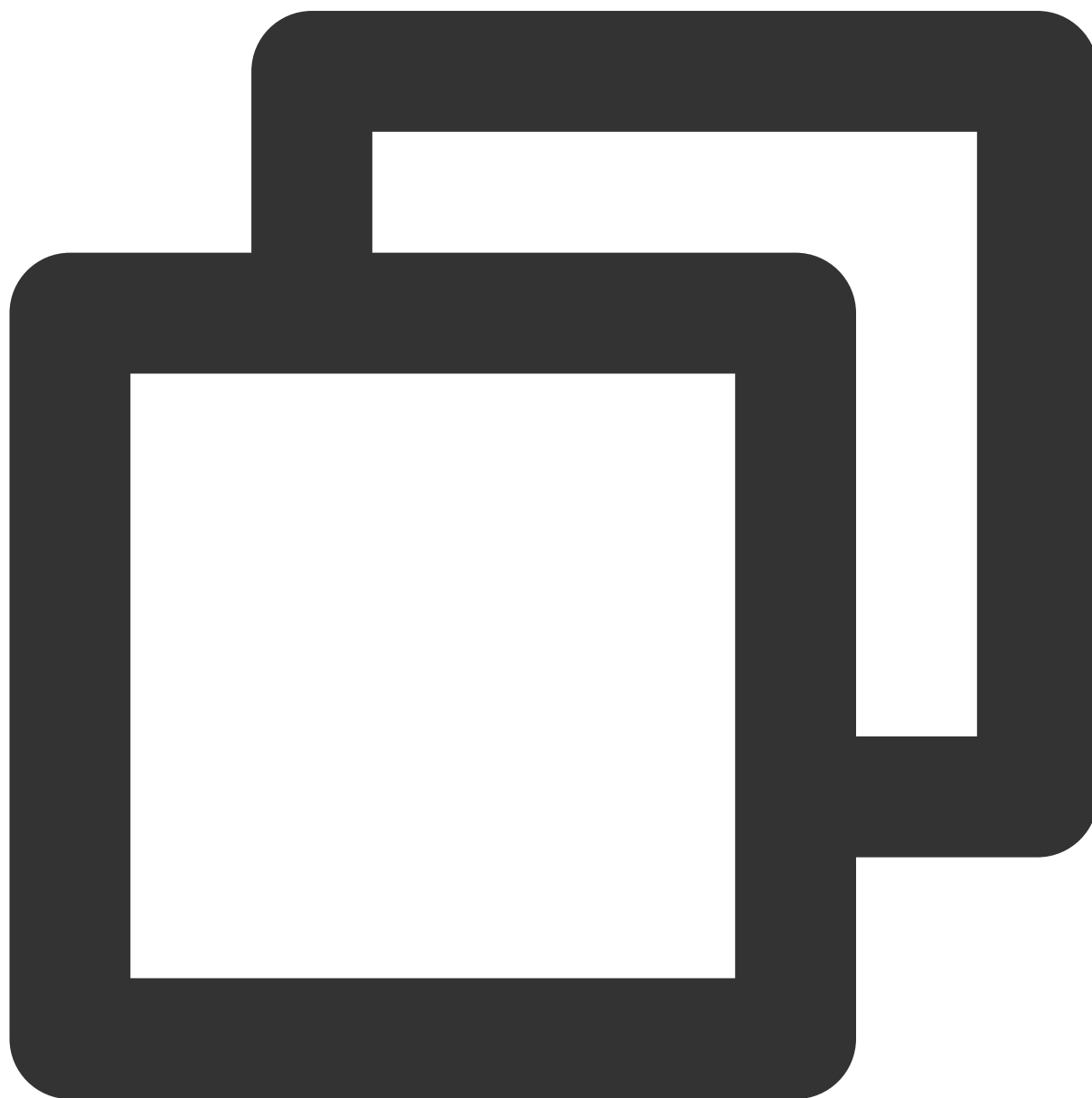


Parameter:

Parameter	Type	Description	Default Value	Meaning
speechMode	TUISpeechMode	Required	-	Update the room's speaking mode

## getUserList

Get the current room's user list, note that the maximum number of user lists fetched by this interface is 100.



```
const roomEngine = new TUIRoomEngine();
```

```
const userList = [];  
let result;  
do {  
  result = await globalProperties.roomEngine.getUserList();  
  userList.push(...result.userInfoList);  
} while (result.nextSequence !== 0)
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
nextSequence	number	Optional	0	Offset, default is to start fetching users from 0

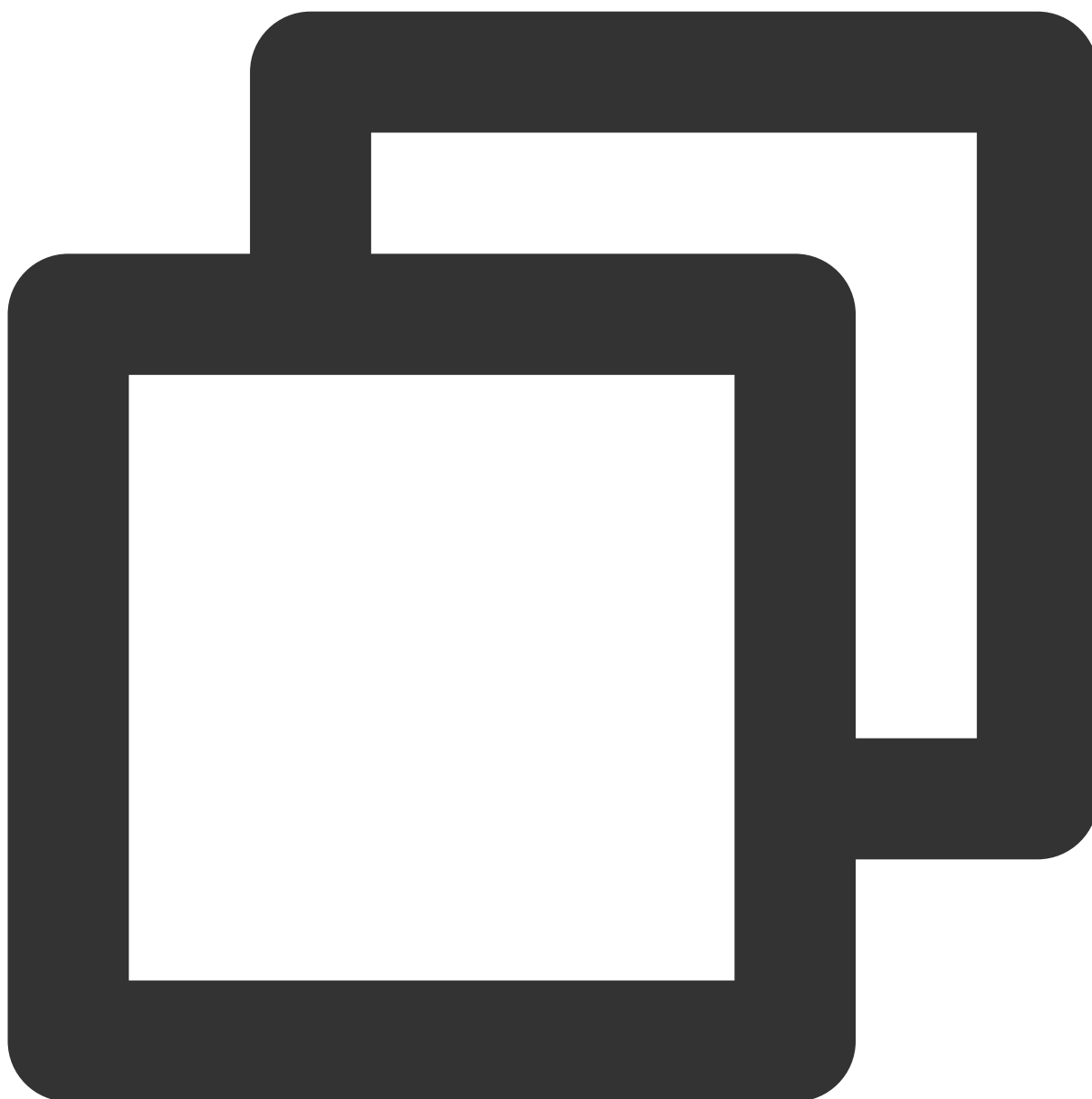
**Returns** : *Promise<object> result*

result.nextSequence is the offset for fetching group users next time, if result.nextSequence is 0, it means that all userList have been fetched

result.userInfoList is the userList fetched this time

## getUserInfo

Get the detailed information of the user.



```
const roomEngine = new TUIRoomEngine();
const userList = [];
const userInfo = await roomEngine.getUserInfo({
  userId: 'user_12345',
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	Get the detailed information of the user

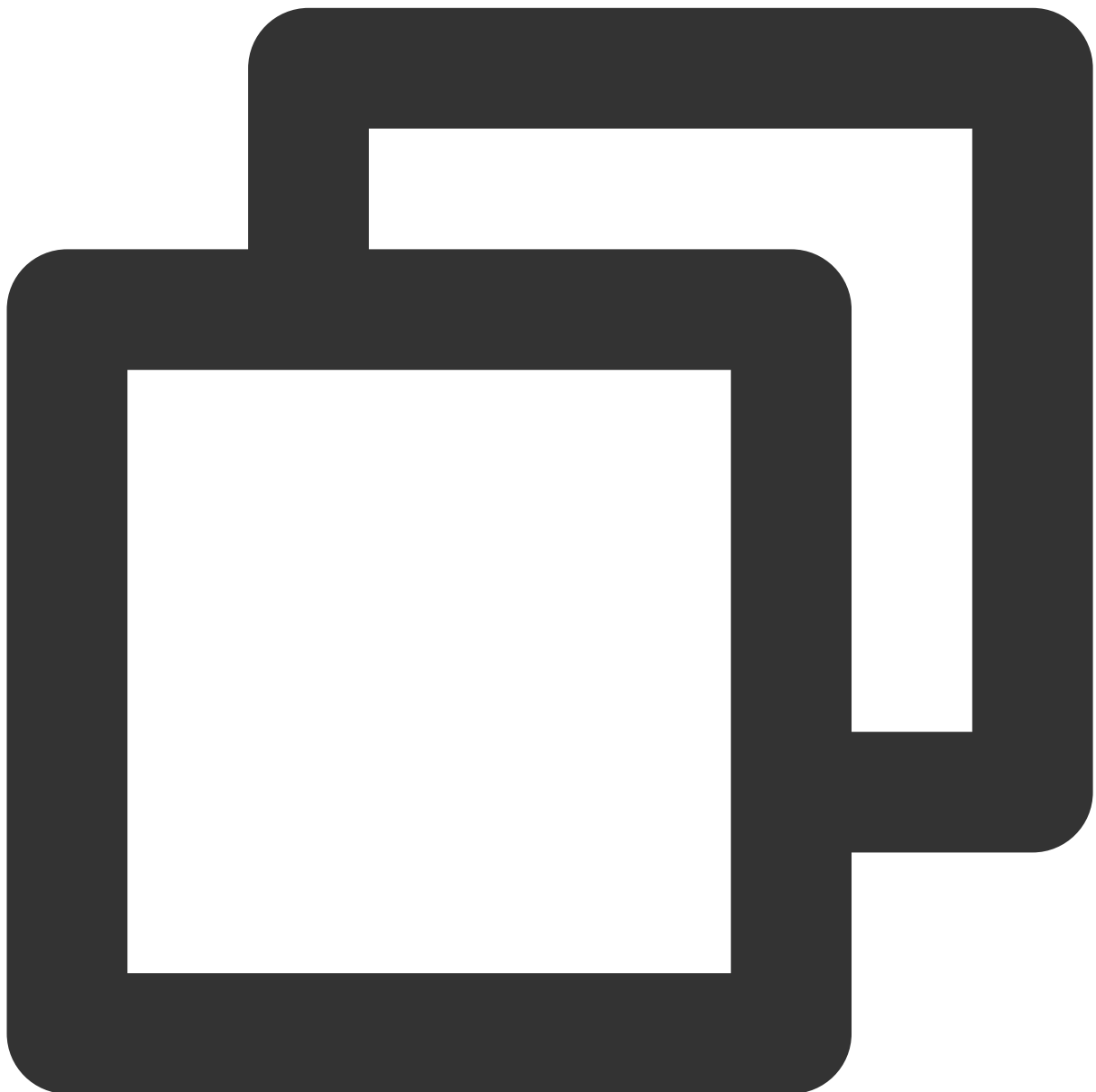
				according to userId
--	--	--	--	---------------------

**Returns** : *Promise*<[TUIUserInfo](#)> *userInfo*

This interface returns the user information of the specified user

### **setLocalVideoView**

Set the rendering position of the local stream.



```
const roomEngine = new TUIRoomEngine();
```

```
// Set the playback area of the local camera stream to the div element with id 'pre
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});

// Set the playback area of the local screen sharing stream to the div element with
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'preview-screen',
});
```

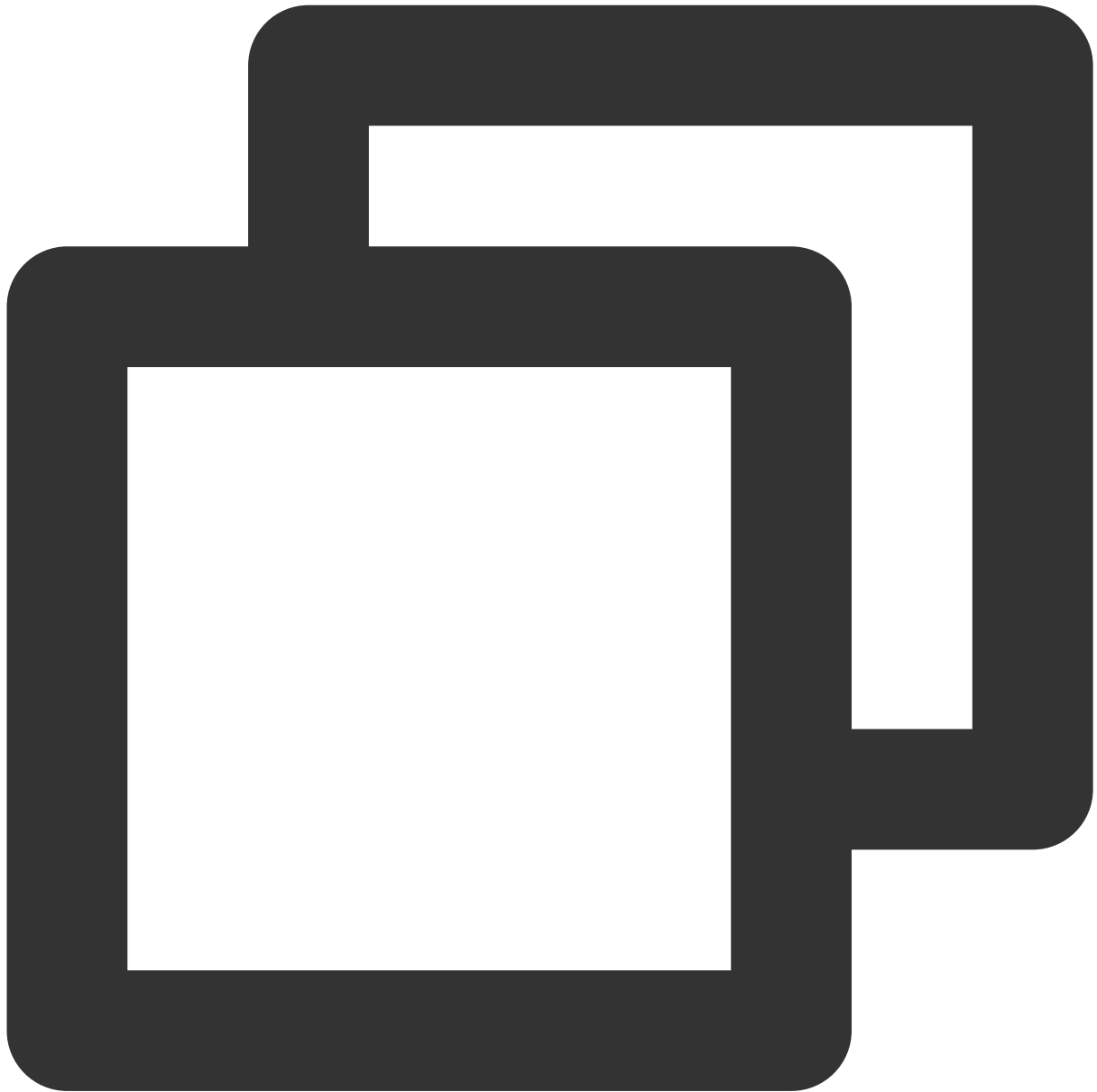
Parameter:

Parameter	Type	Description	Default Value	Meaning
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	Local stream type
view	string	Required	-	The id of the div element corresponding to the streamType

Returns : *Promise<void>*

## openLocalCamera

Open the local camera and start capturing video streams.

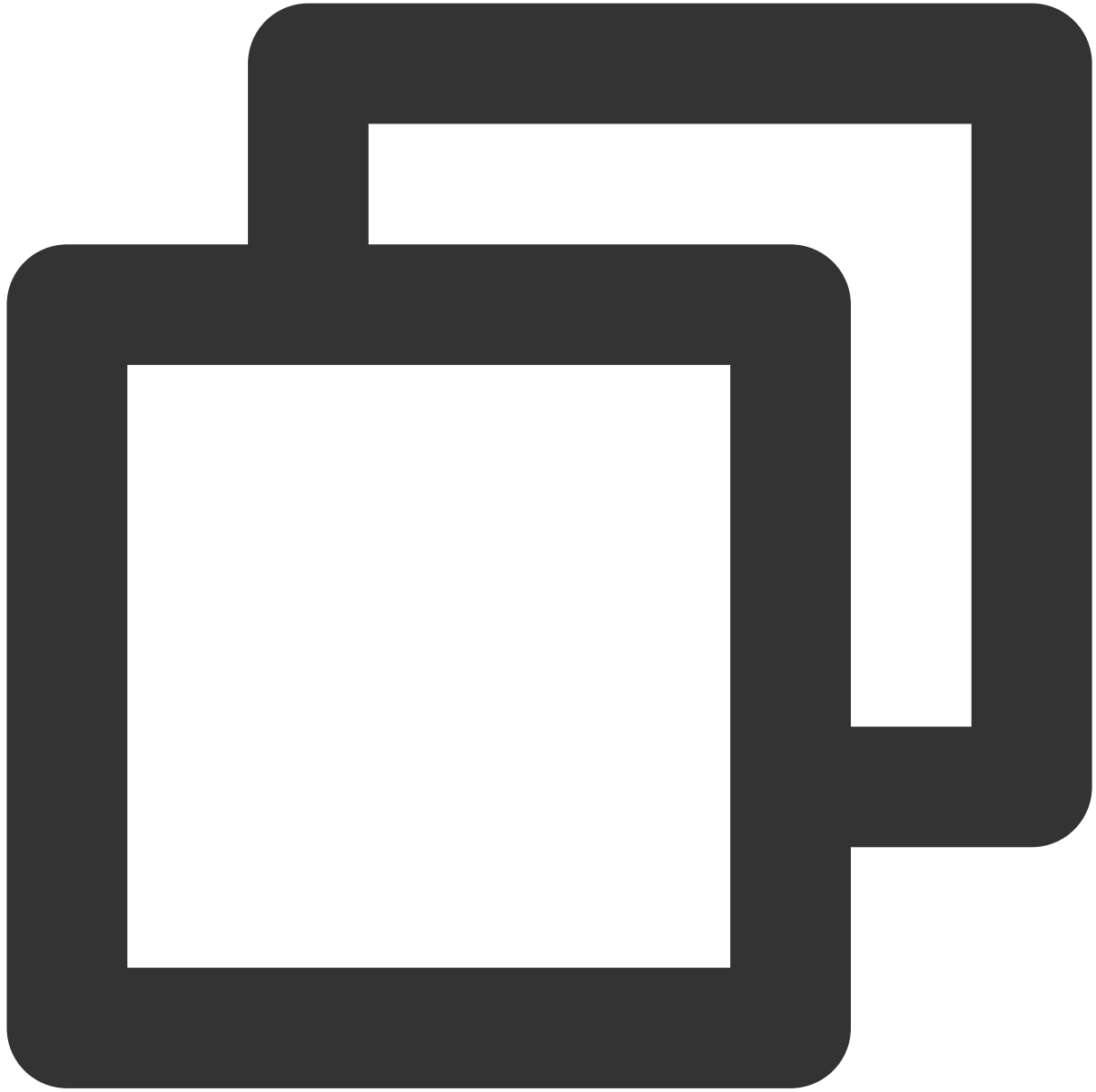


```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});
await roomEngine.openLocalCamera();
```

Returns : *Promise<void>*

### **closeLocalCamera**

Close the local camera.



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.closeLocalCamera();
```

Returns : *Promise<void>*

### **openLocalMicrophone**

Open the local mic and start capturing audio streams.



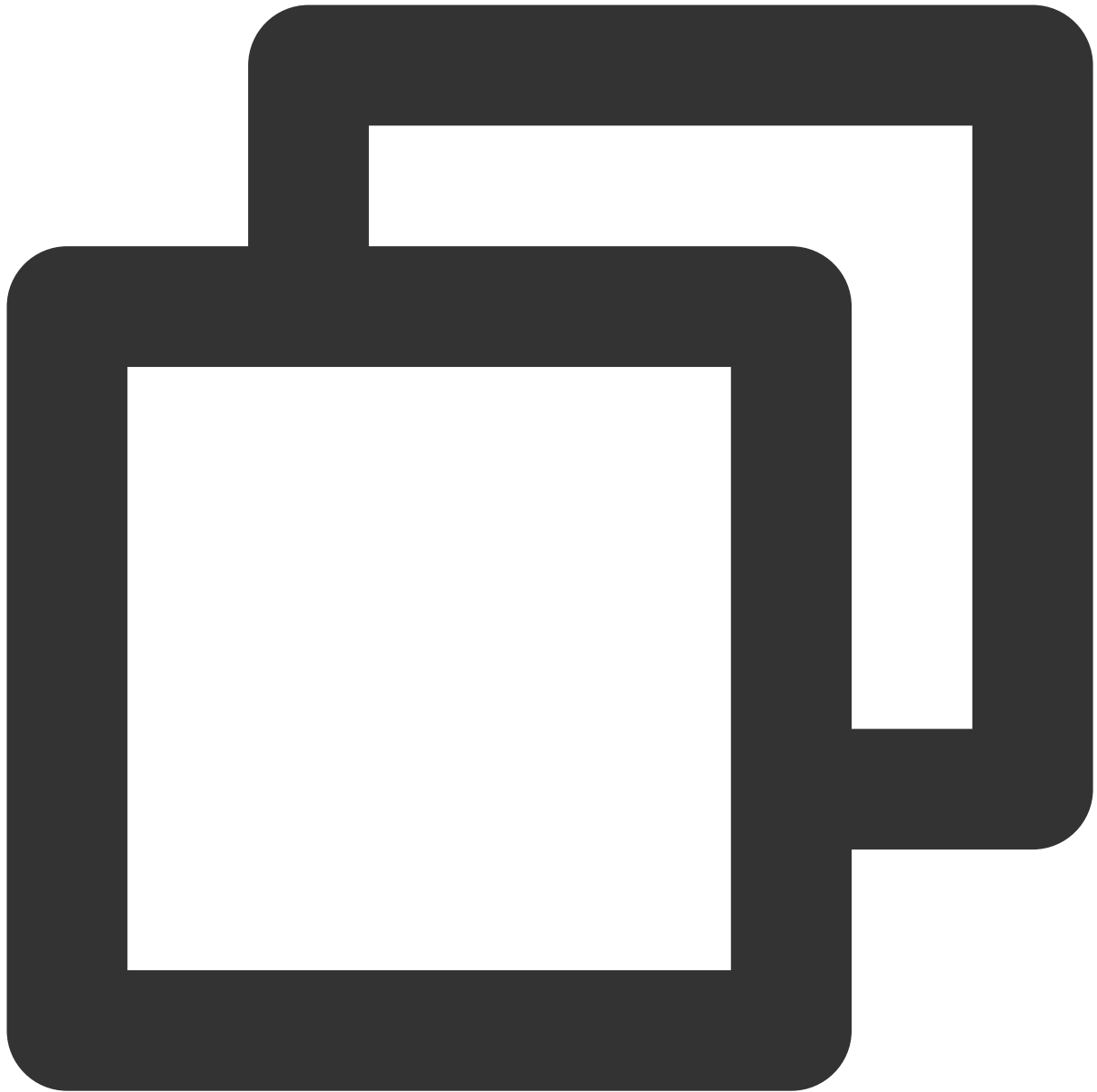
```
const roomEngine = new TUIRoomEngine();  
await roomEngine.openLocalMicrophone();
```

Returns : *Promise<void>*

### **closeLocalMicrophone**

Close the local mic.



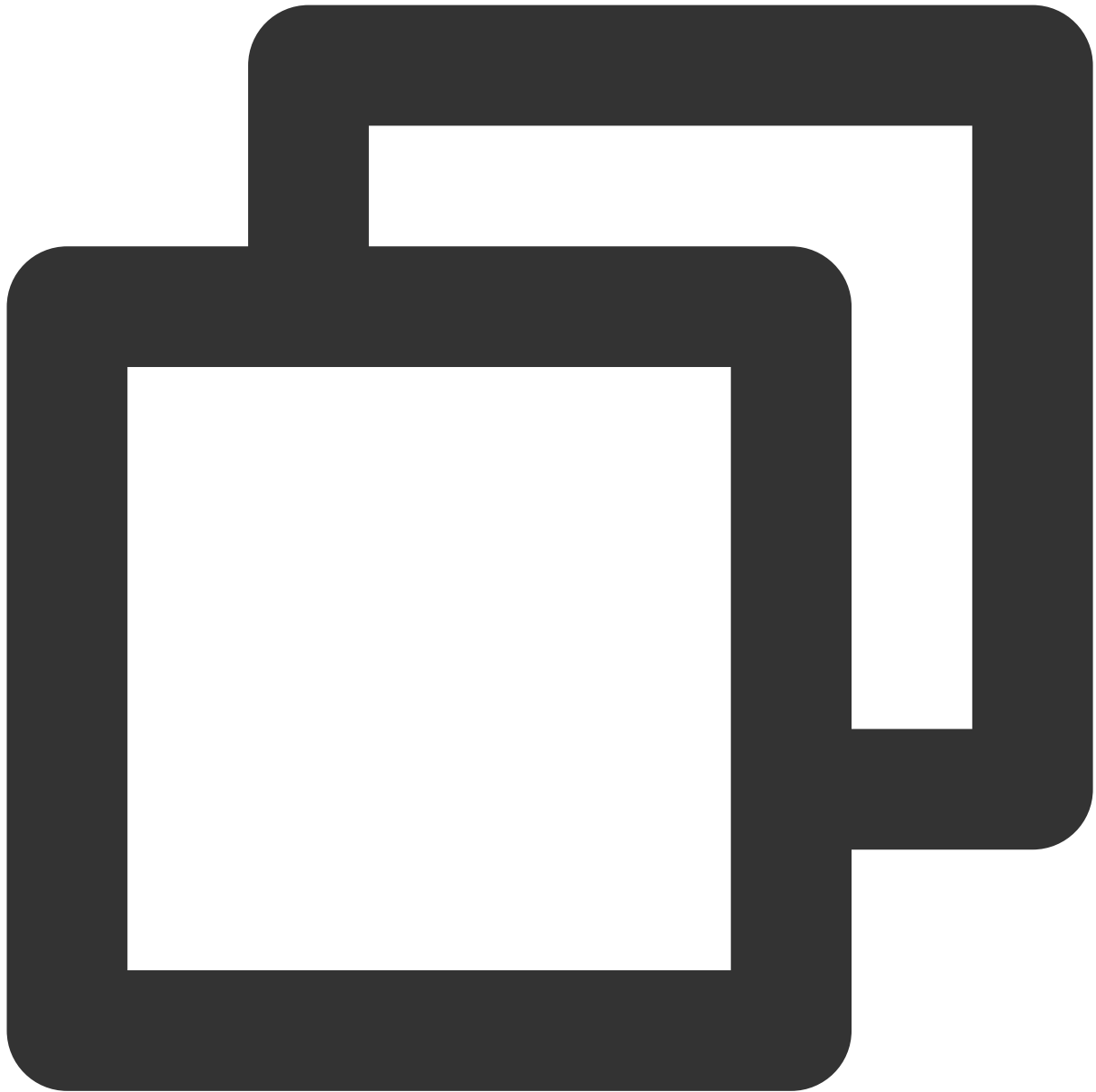


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.closeLocalMicrophone();
```

Returns : *Promise<void>*

### **updateVideoQuality**

Set the codec parameters of the local video stream, default is `TUIVideoProfile.kVideoQuality_720P`.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQuality({
  quality: TUIVideoQuality.kVideoQuality_540p,
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
quality	<a href="#">TUIVideoQuality</a>	Required	-	Clear TUIVideoProfile.kVideoQuality_360P

				SD TUIVideoProfile.kVideoQuality_540P HD TUIVideoProfile.kVideoQuality_720P Full HD TUIVideoProfile.kVideoQuality_1080P
--	--	--	--	---

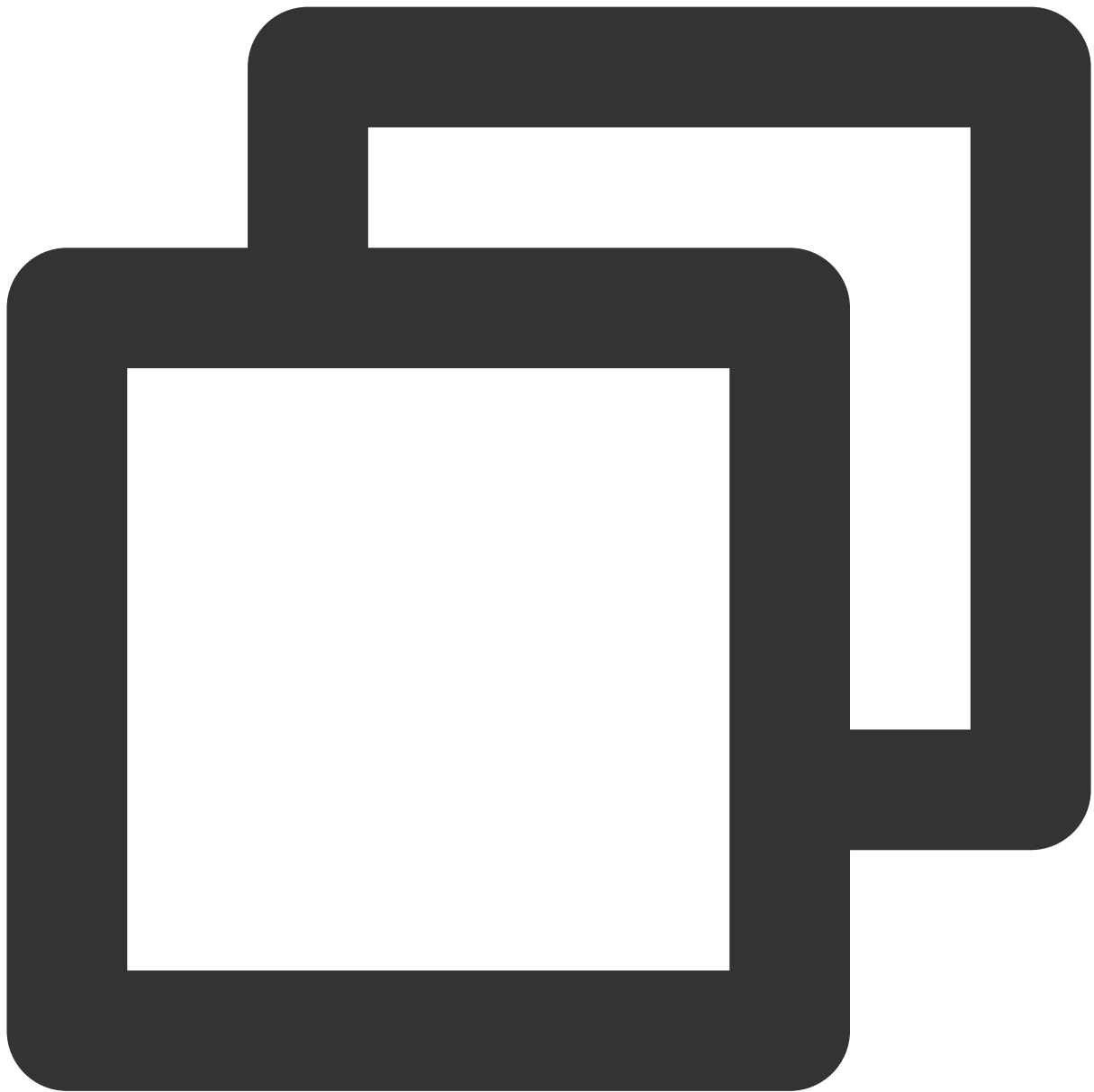
Returns : *Promise<void>*

## updateAudioQuality

Set Local Audio Parameters.

### Note:

This method needs to be set before openLocalMicrophone, otherwise it will not take effect.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalAudioProfile({
  audioProfile: TUIAudioProfile.kAudioProfileSpeech,
});
```

Parameter:

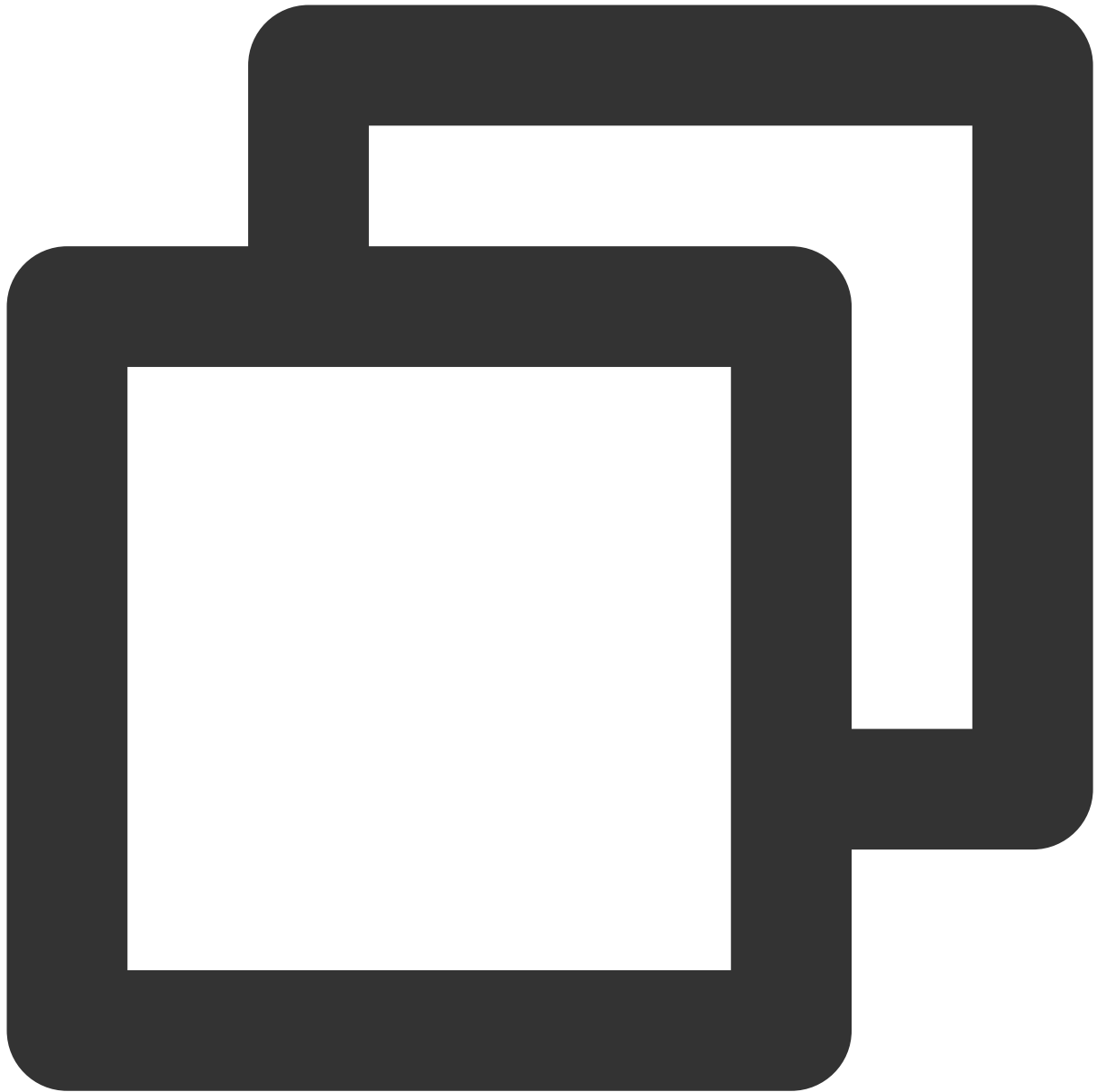
Parameter	Type	Description	Default Value	Meaning
audioProfile	<a href="#">TUIAudioProfile</a>	Required	-	TUIAudioProfile.kAudioProfileSpeech: Speech

				Mode; Sample rate: 16k TUIAudioProfile.kAudioProfileDefault: Standard Mode (or Default Mode); Sample rate: 48k TUIAudioProfile.kAudioProfileMusic: Music Mode; Sample rate: 48k
--	--	--	--	--

Returns : *Promise<void>*

## **startPushLocalVideo**

After entering the room, the local video stream will be pushed to the remote by default. This interface is used to re-push the local video stream to the remote after stopping the push.

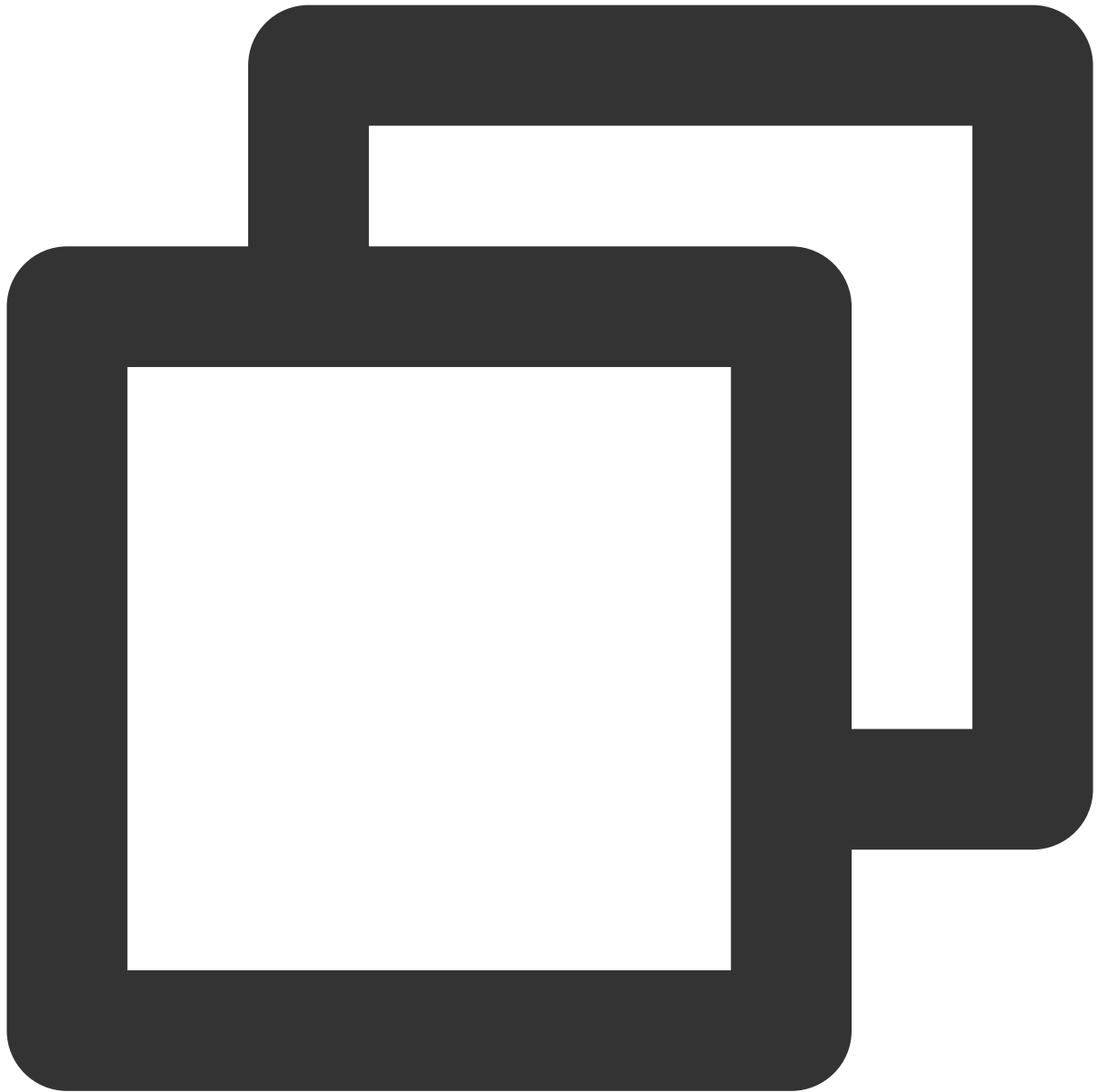


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startPushLocalVideo();
```

Returns : *Promise<void>*

### **stopPushLocalVideo**

Stop Pushing Local Video Stream to Remote.



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopPushLocalVideo();
```

Returns : *Promise<void>*

### **startPushLocalAudio**

After entering the room, the local audio stream will be pushed to the remote by default. This interface is used to re-push the local audio stream to the remote after stopping the push.



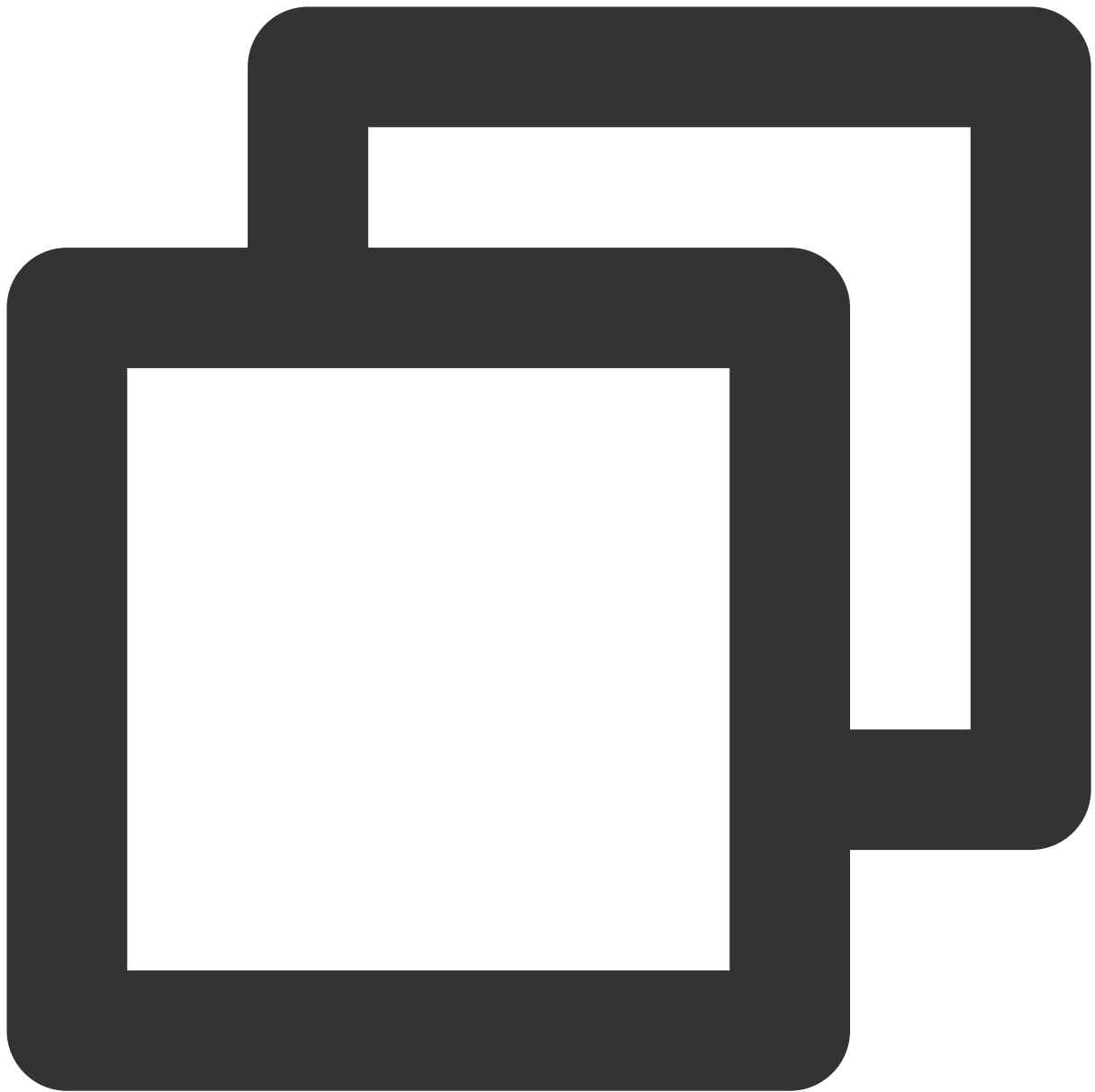
```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startPushLocalAudio();
```

Returns : *Promise<void>*

### **stopPushLocalAudio**

Stop Pushing Local Audio Stream to Remote.





```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopPushLocalAudio();
```

Returns : *Promise<void>*

### **setRemoteVideoView**

Set Remote Stream Rendering Area.



```
const roomEngine = new TUIRoomEngine();

// Set the remote user's video stream to play in the area with id 'remote_preview_c
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'remote_preview_camera',
});
// Set the remote user's screen sharing stream to play in the area with id 'remote_
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
```

```
streamType: TUIVideoStreamType.kScreenStream,  
view: 'remote_preview_screen',  
});
```

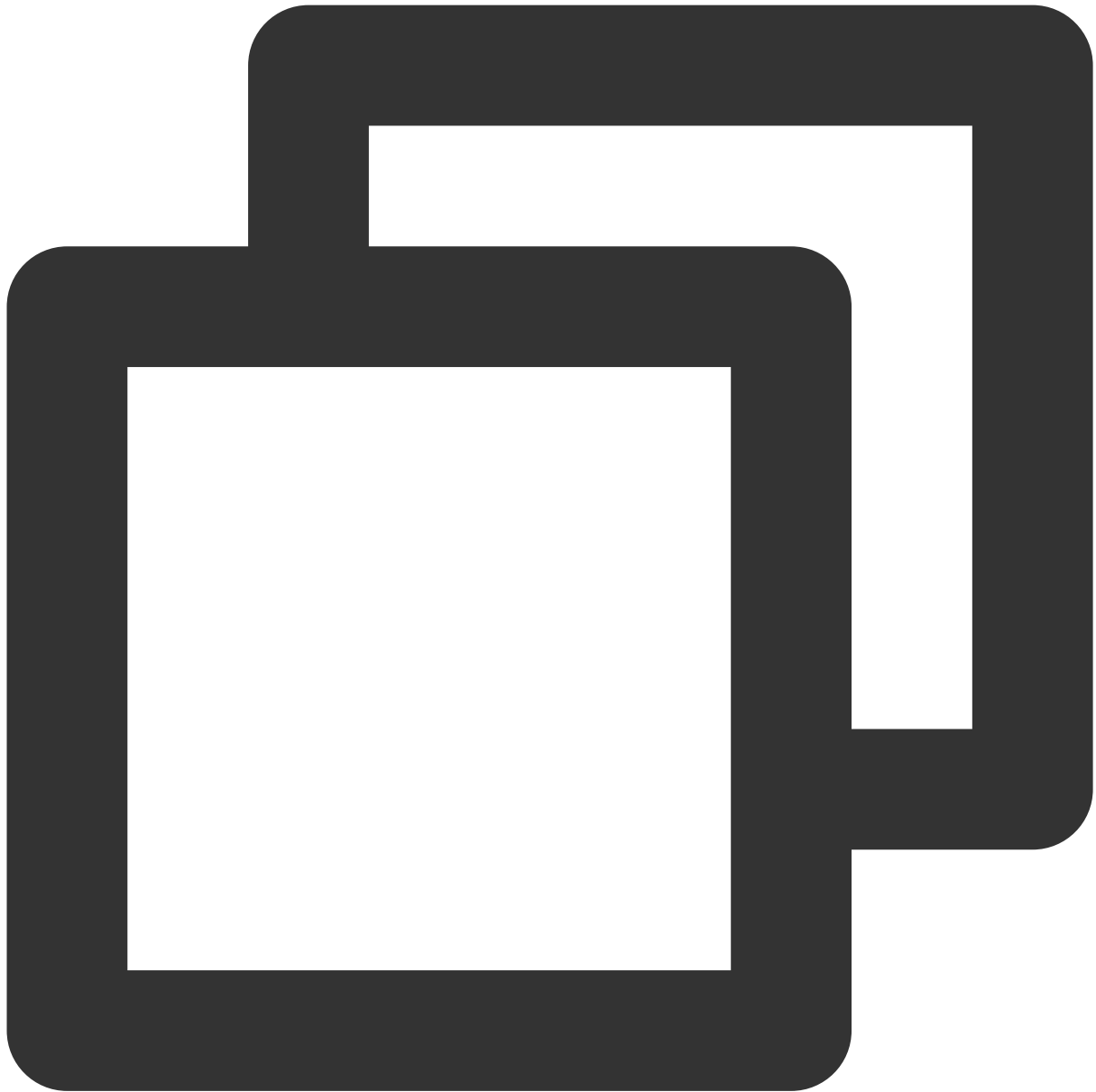
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type
view	string	Required	-	The id of the div element playing the remote user's stream

Returns : *Promise<void>*

### **startPlayRemoteVideo**

Start Playback of Remote User Video Stream.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

Parameter:

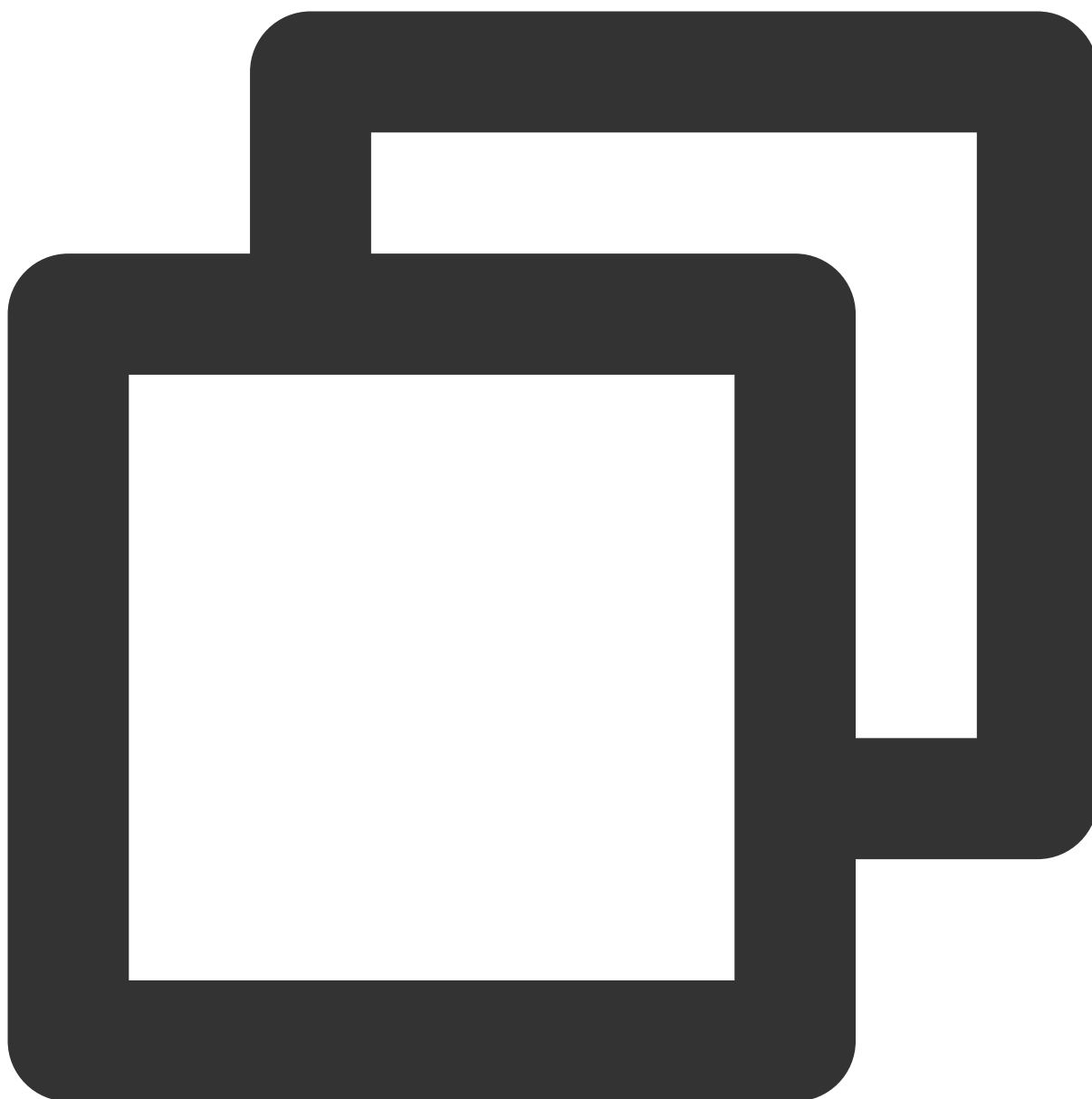
Parameter	Type	Description	Default Value	Meaning

userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type TUIVideoStreamType.kCameraStream Video Stream TUIVideoStreamType.kScreenStream Screen Sharing Stream TUIVideoStreamType.kCameraStreamLow Low Definition Video Stream

Returns : *Promise<void>*

## stopPlayRemoteVideo

Stop Playback of Remote User Video Stream.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning

userId	string	Required	-	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Required	-	User Stream Type TUIVideoStreamType.kCameraStream Video Stream TUIVideoStreamType.kScreenStream Screen Sharing Stream TUIVideoStreamType.kCameraStreamLow Low Definition Video Stream

Returns : *Promise<void>*

### **muteRemoteAudioStream**

Stop Remote User's Audio Stream.



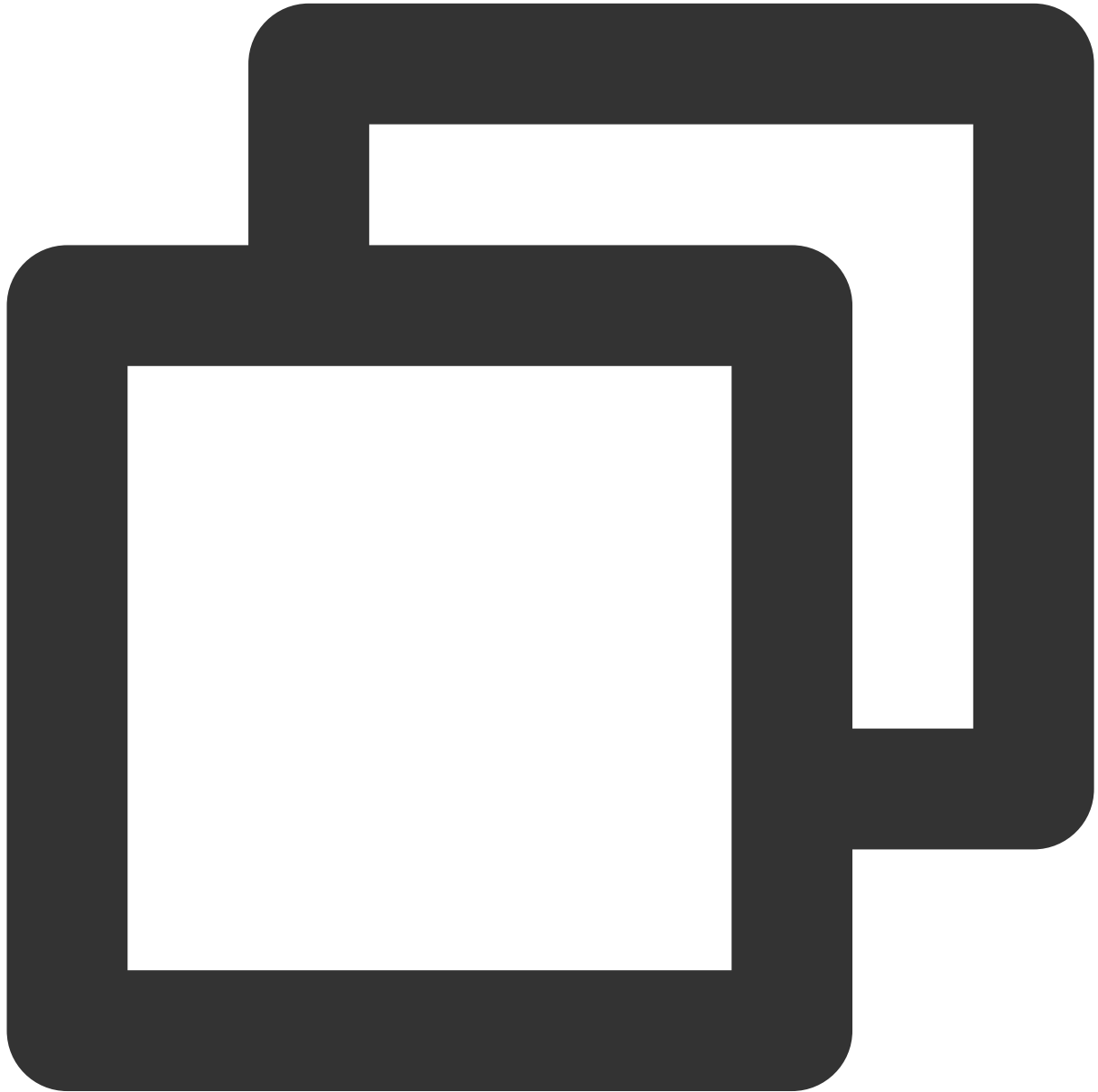
```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteRemoteAudioStream({
  userId: 'user_1234',
  isMute: true,
});
```

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
isMute	boolean	Required	-	Whether to Stop Remote User's Audio



## openRemoteDeviceByAdmin

Request Remote User to Open Media Device.



```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.openRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone //The requested device is a mic
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
```

```

switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
        // Request Accepted
        break;
    case TUIRequestCallbackType.kRequestRejected:
        // Request Rejected
        break;
    case TUIRequestCallbackType.kRequestCancelled:
        // Request Canceled
        break;
    case TUIRequestCallbackType.kRequestTimeout:
        // Request Timeout
        break;
    case TUIRequestCallbackType.kRequestError:
        // Request Error
        break;
    default:
        break;
}
},
});

```

Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
device	<a href="#">TUIMediaDevice</a>	Required	-	Media Device Type (Camera/Mic/Screen Sharing)
timeout	number	Required	-	Timeout Time. If timeout is set to 0, there is no timeout time
requestCallback	Function	Optional	Empty Function	Request Callback, used to notify the initiator of the request being accepted/rejected/canceled/timeout/error

Returns : *Promise<string> requestId*

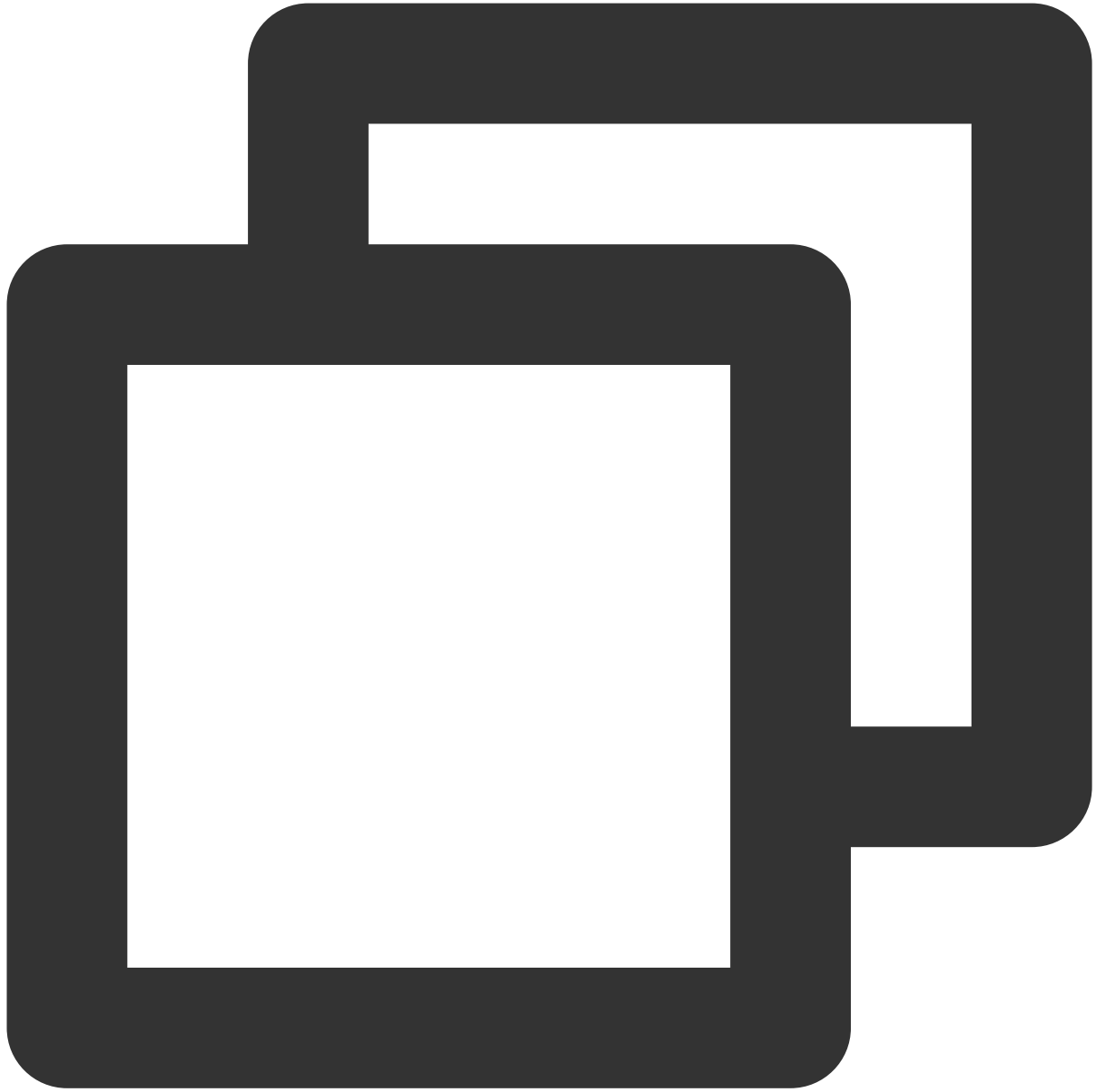
This interface returns requestId, users can use this requestId to call cancelRequest interface to cancel the request.

#### Description:

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

### applyToAdminToOpenLocalDevice

Participant applies to the host to open the device.



```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.applyToAdminToOpenLocalDevice({
  device: TUIMediaDevice.kMicrophone //The requested device is a mic
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
    switch (requestCallbackType) {
      case TUIRequestCallbackType.kRequestAccepted:
        // Request Accepted
        break;
    }
  }
});
```

```
case TUIRequestCallbackType.kRequestRejected:
    // Request Rejected
    break;
case TUIRequestCallbackType.kRequestCancelled:
    // Request Canceled
    break;
case TUIRequestCallbackType.kRequestTimeout:
    // Request Timeout
    break;
case TUIRequestCallbackType.kRequestError:
    // Request Error
    break;
default:
    break;
}
},
});
```

Parameter:

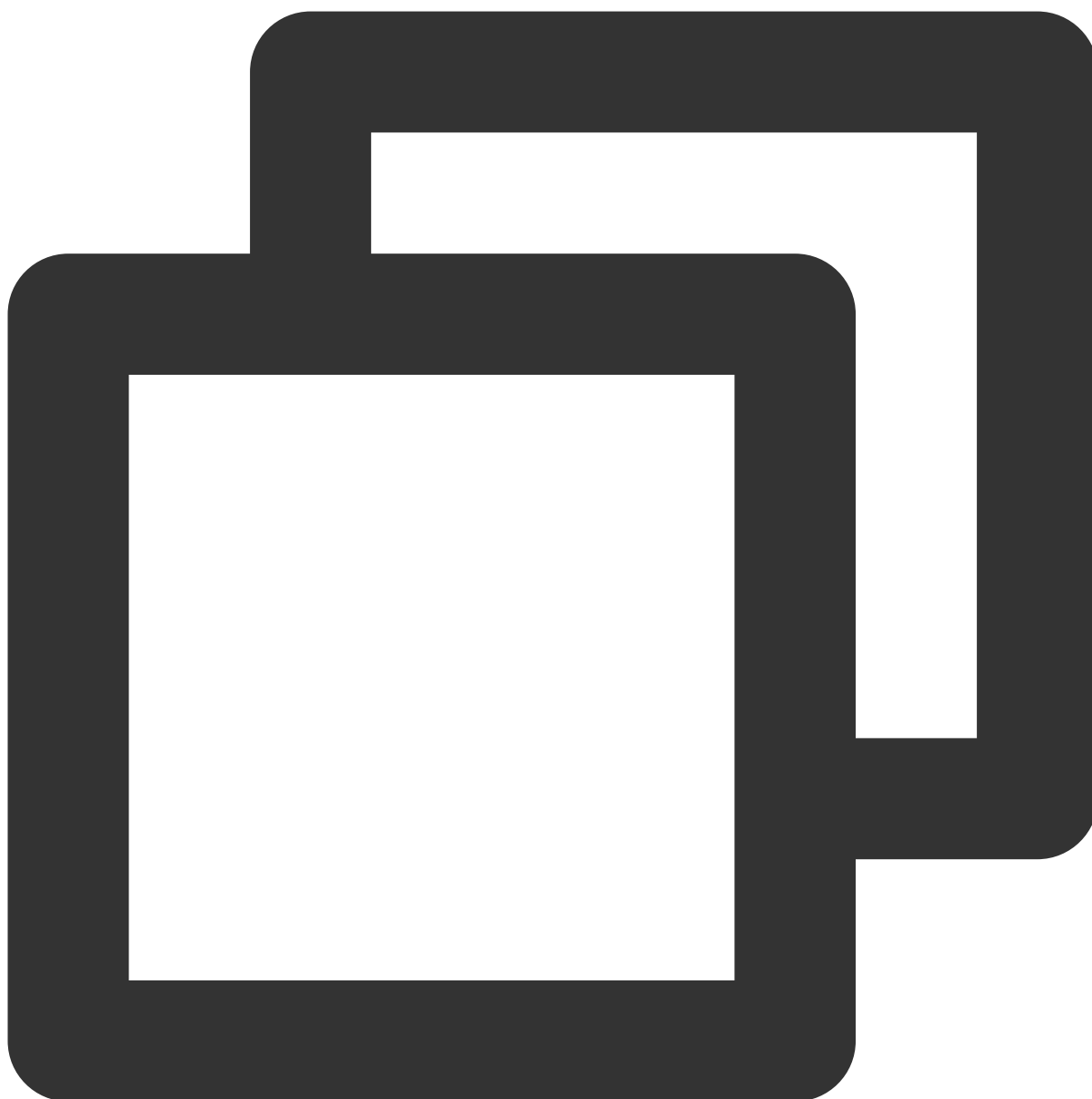
Parameter	Type	Description	Default Value	Meaning
device	<a href="#">TUIMediaDevice</a>	Required	-	Media Device Type (Camera/Mic/Screen Sharing)
timeout	number	Required	-	Timeout Time. If timeout is set to 0, there is no timeout time
requestCallback	Function	Optional	Empty Function	Request Callback, used to notify the initiator of the request being accepted/rejected/canceled/timeout/error

Returns : *Promise<string> requestId*

This interface returns requestId, users can use this requestId to call cancelRequest interface to cancel the request.

### closeRemoteDeviceByAdmin

Close Remote User Media Device.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone, //Close mic
});
```

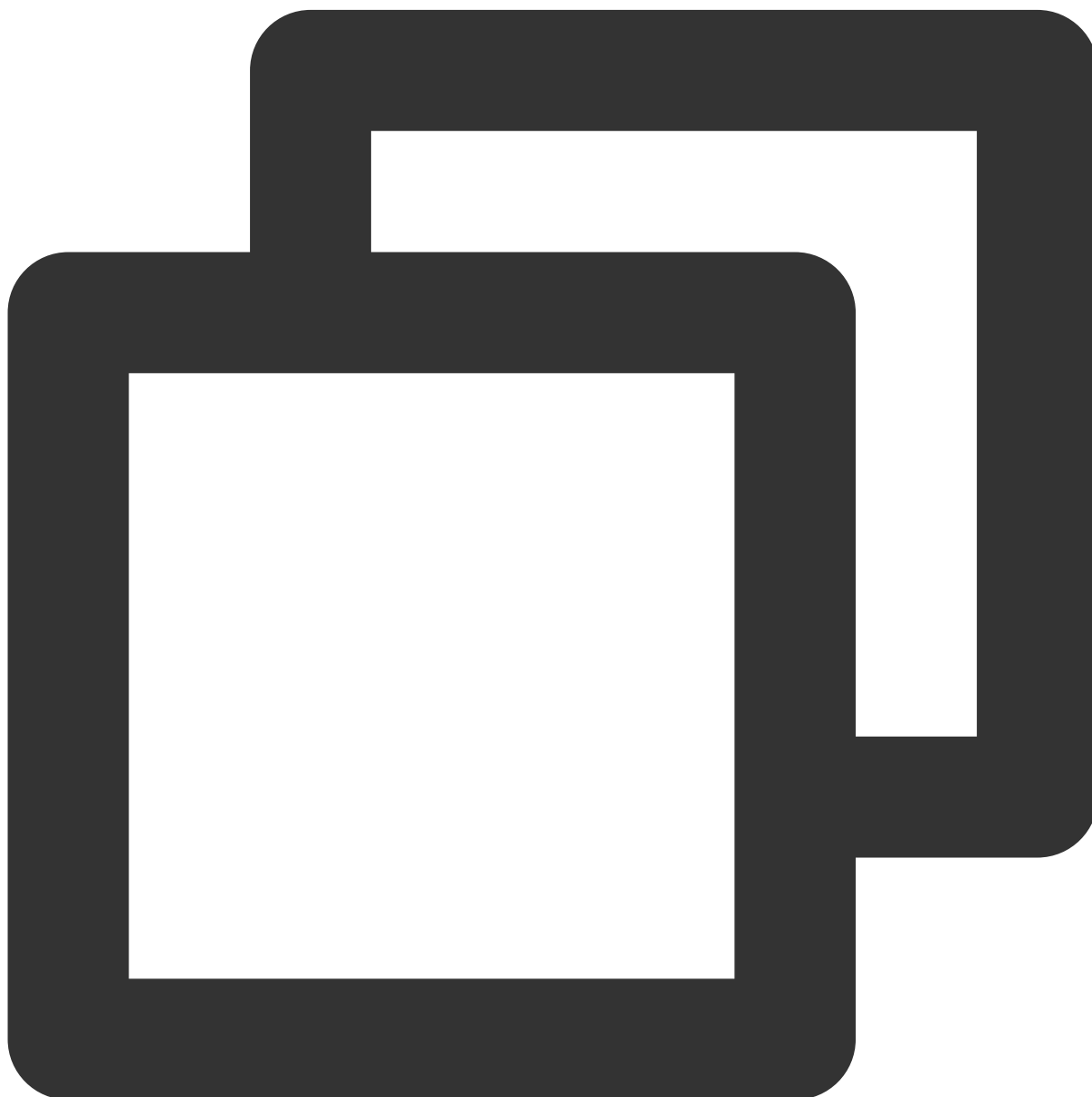
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID

device	<a href="#">TUIMediaDevice</a>	Required	-	Media Device Type (Camera/Mic/Screen Sharing)
--------	--------------------------------	----------	---	--

## cancelRequest

Cancel Already Sent Request.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.cancelRequest({
  requestId: '',    // Please use Actual requestId
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
requestId	string	Required	-	Request ID

**Returns** : *Promise<string>* requestId

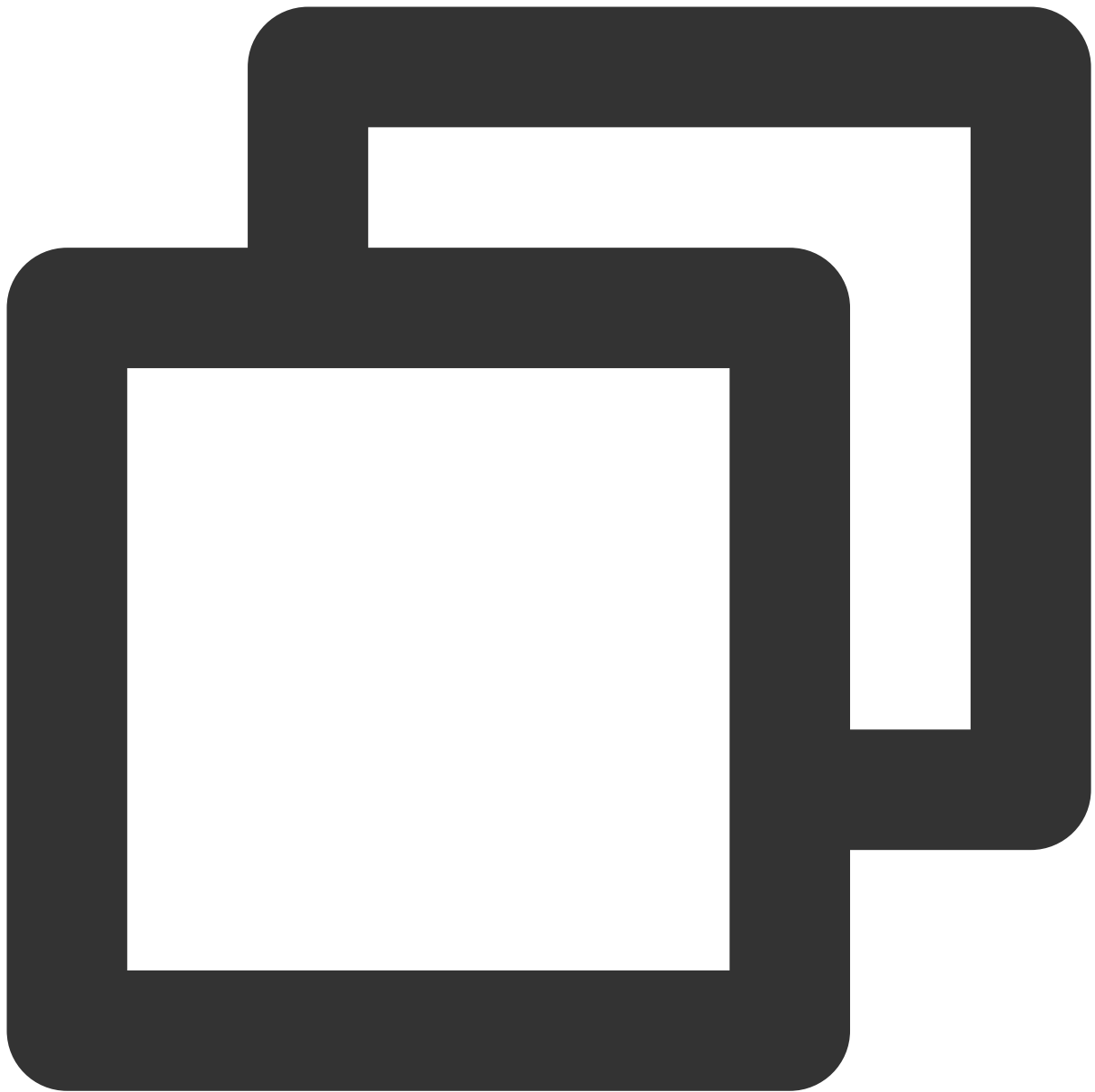
This interface returns requestId, users can use this requestId to call cancelRequest interface to cancel the request

**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## responseRemoteRequest

Reply to Remote User's Request.



```
const roomEngine = new TUIRoomEngine();
// Agree to Remote User's Request
await roomEngine.responseRemoteRequest({
  requestId: '',    // Please use Actual requestId
  agree: true,
});
// Reject Remote User's Request
await roomEngine.responseRemoteRequest({
  requestId: '',    // Please use Actual requestId
  agree: false,
});
```



Parameter:

Parameter	Type	Description	Default Value	Meaning
requestId	string	Required	-	Request ID
agree	boolean	Required	-	Whether to Agree

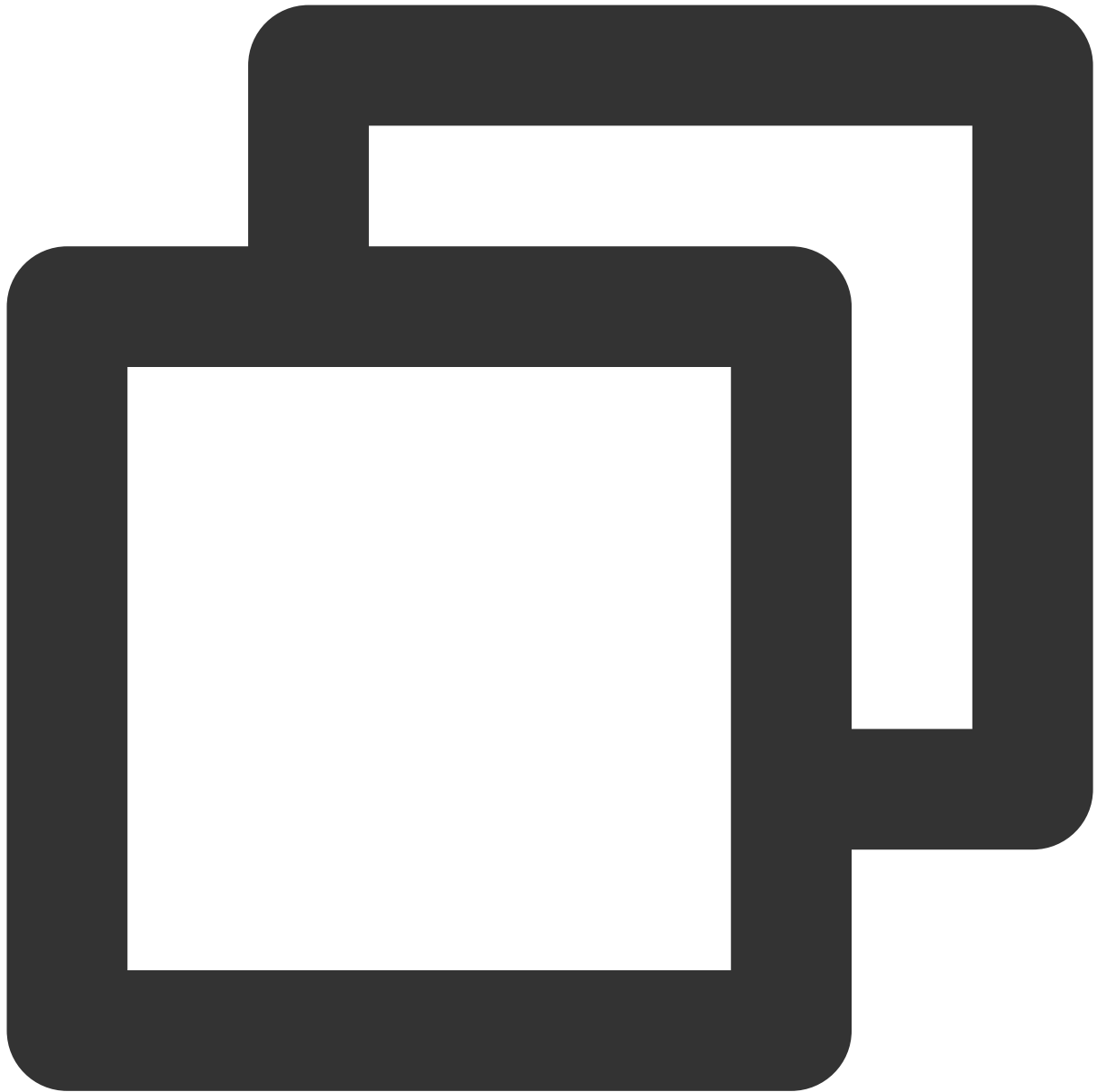
**Returns** : *Promise<void>*

**Note** :

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

### **disableDeviceForAllUserByAdmin**

Prohibit/Allow All Users to Open Media Device (This Interface is invalid for Room Owner and Administrator).



```
// Example 1: Prohibit All Users to Open Mic
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: true,
})
// Example 2: Allow All Users to Open Mic
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: false,
})
```

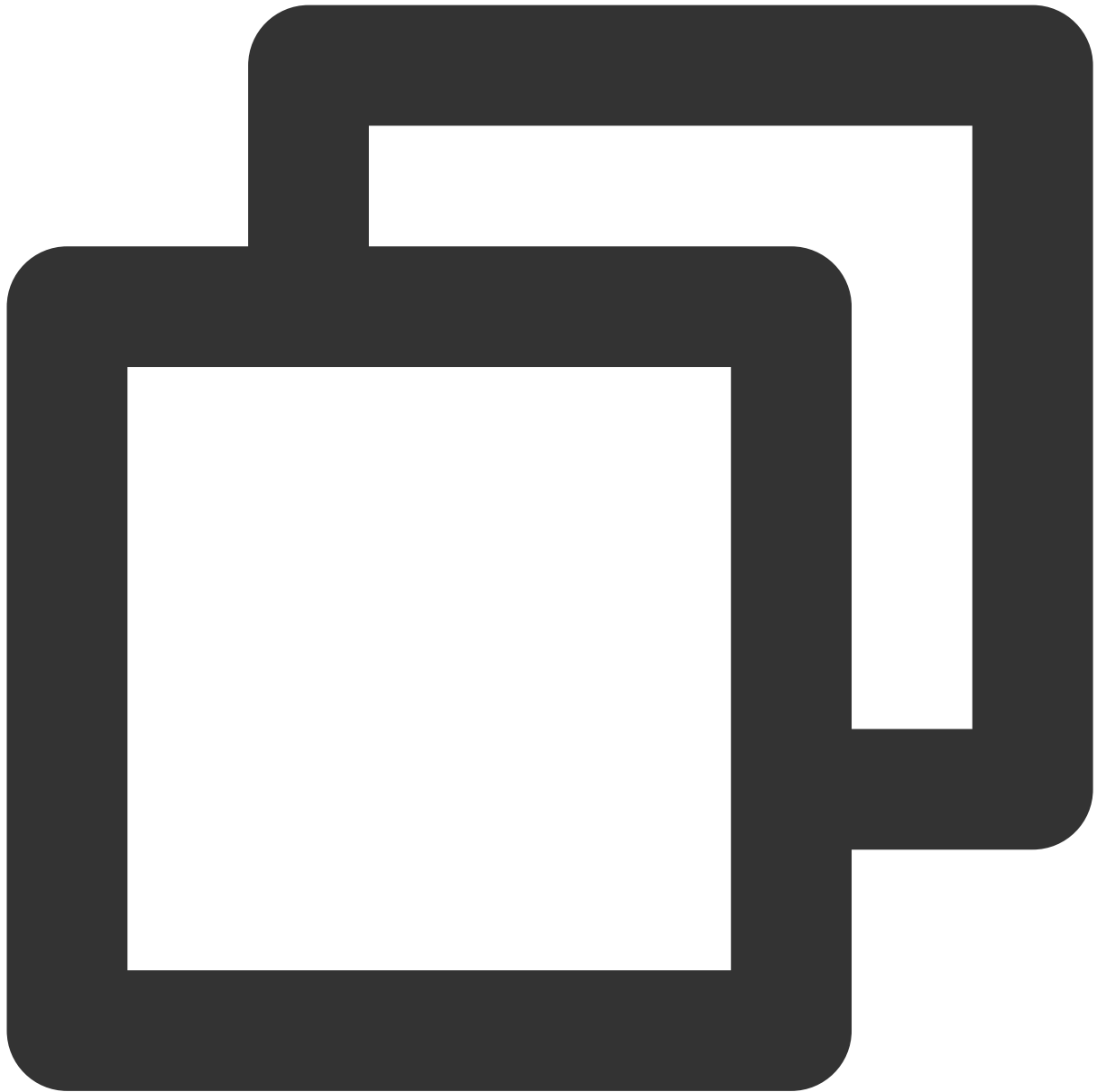
Parameter:

Parameter	Type	Description	Default Value	Meaning
device	<a href="#">TUIMediaDevice</a>	Required	-	Disabled Media Device Type (Camera/Mic/Screen Sharing)
isDisable	boolean	Required	-	Whether it is Prohibited

Returns : *Promise<void>*

### **disableSendingMessageForAllUser**

Whether All Users are Allowed to Send Messages (This Interface is invalid for Room Owner and Administrator).



```
await roomEngine.disableSendingMessageForAllUser({
  isDisable: true,
});
```

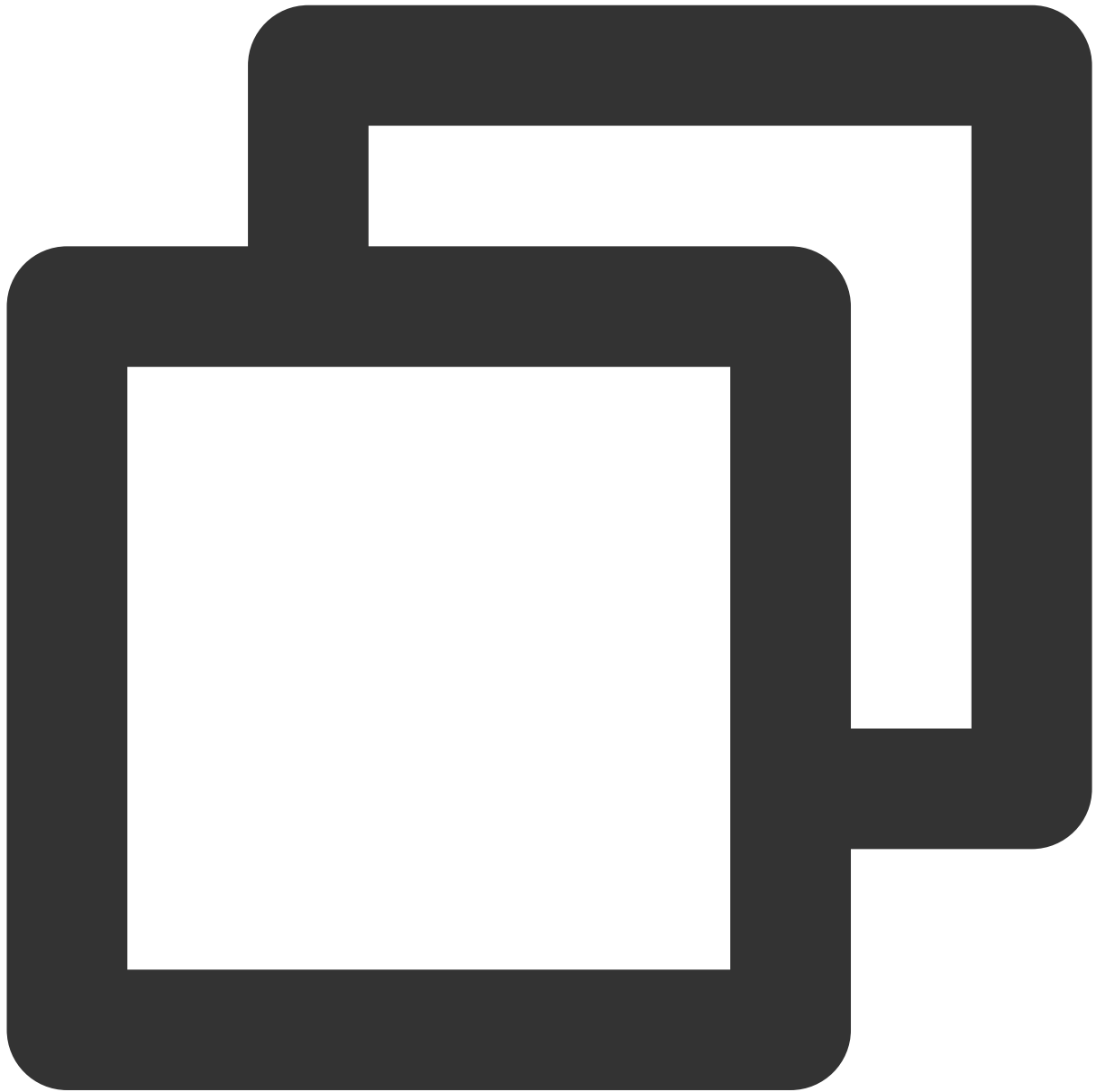
Parameter:

Parameter	Type	Description	Default Value	Meaning
isDisable	boolean	Required	-	Whether it is Disabled

Returns : *Promise<void>*

### **disableSendingMessageByAdmin**

Whether Specific User is Allowed to Send Messages.



```
await roomEngine.disableSendingMessageByAdmin({
  userId: 'user_1234',
  isDisable: true,
});
```

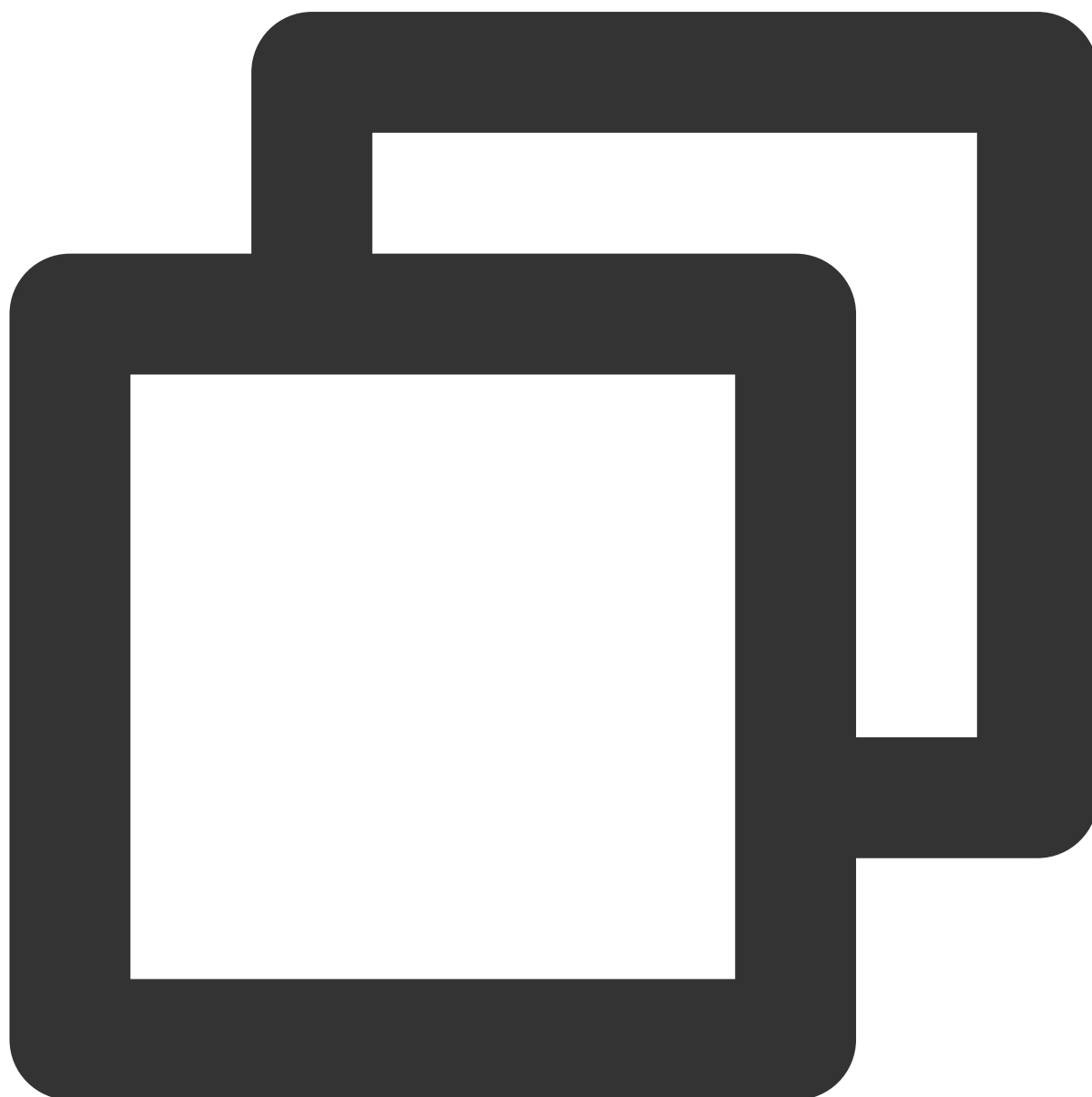
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
isDisable	boolean	Required	-	Whether it is Disabled

Returns : *Promise<void>*

## changeUserRole

Change User's Role (Only the Host can call this Interface).



```
const roomEngine = new TUIRoomEngine();

// Transfer the Room to User user_1234
await roomEngine.changeUserRole({
  userId: 'user_1234',
  role: TUIRole.kRoomOwner,
});

// Set user_1234 as Room Administrator
await roomEngine.changeUserRole({
  userId: 'user_1234',
  userRole: TUIRole.kAdministrator,
});
```

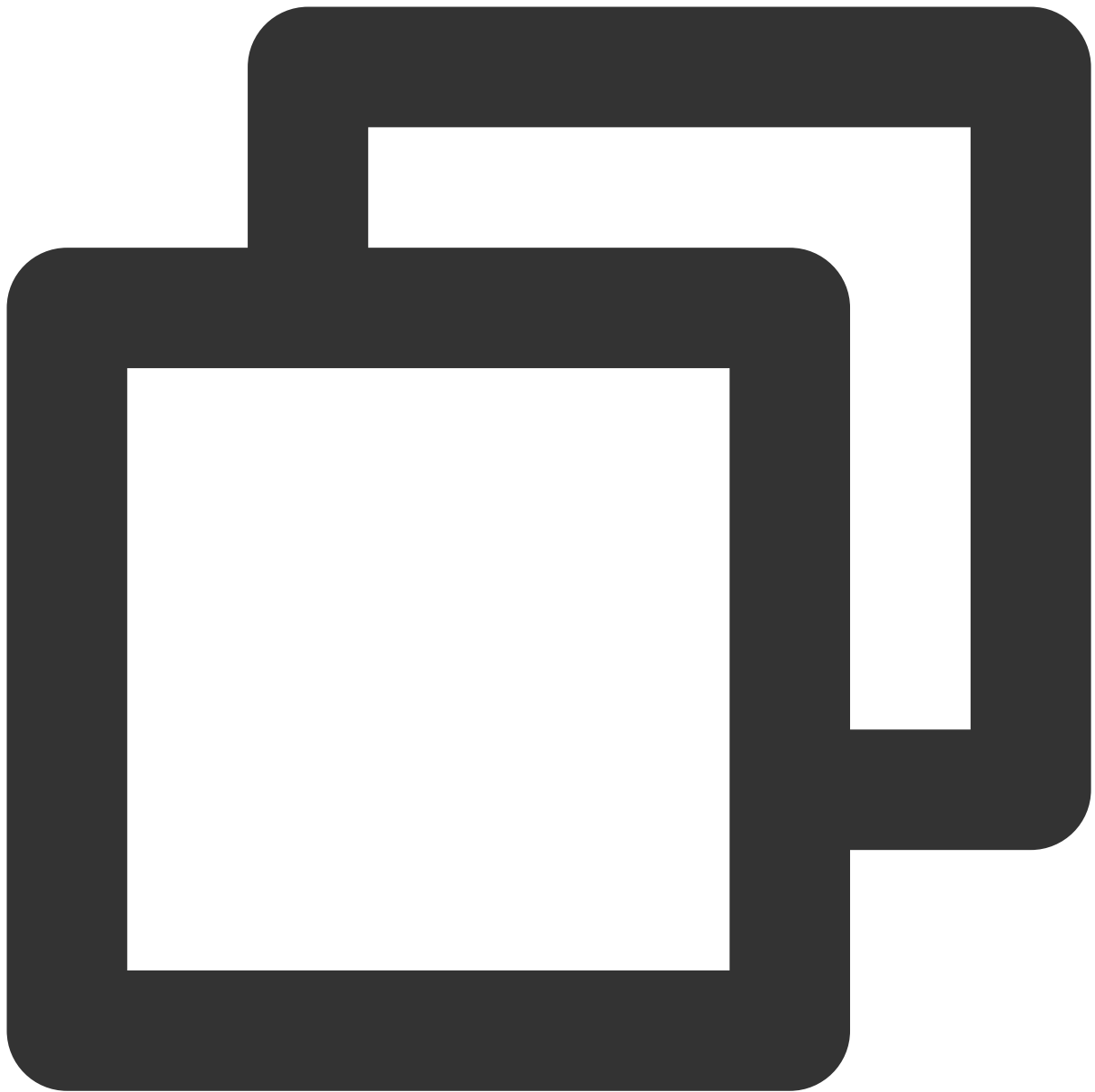
Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID
userRole	<a href="#">TUIRole</a>	Required	-	User Role Host TUIRole.kRoomOwner Administrator TUIRole.kAdministrator General Member TUIRole.kGeneralUser

Returns : *Promise<void>*

## kickRemoteUserOutOfRoom

Kick Out User from Room (Only Host and Administrator can call this Interface).



```
const roomEngine = new TUIRoomEngine();
await roomEngine.kickRemoteUserOutOfRoom({
  userId: 'user_1234',
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
userId	string	Required	-	User ID



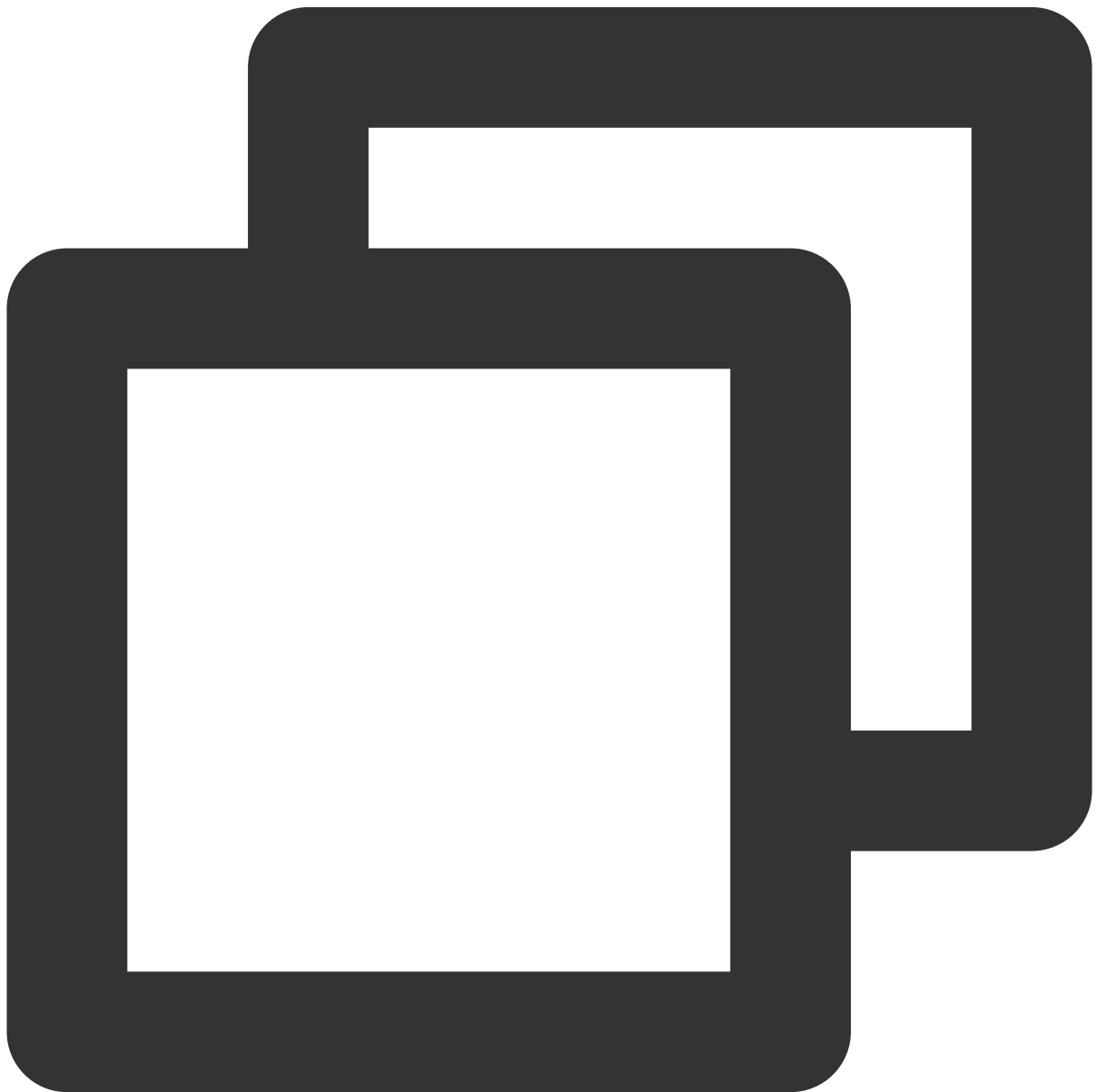
Returns : *Promise<void>*

## setMaxSeatCount

Set Room Seat Maximum Value

When roomType is TUIRoomType.kConference (Education and Conference Scene), maxSeatCount value is not limited;

When roomType is TUIRoomType.kLivingRoom (Live Scene), maxSeatCount is limited to 16;



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.createRoom({ roomId: '12345' });
```

```
await roomEngine.setMaxSeatCount({ maxSeatCount: 16 })
```

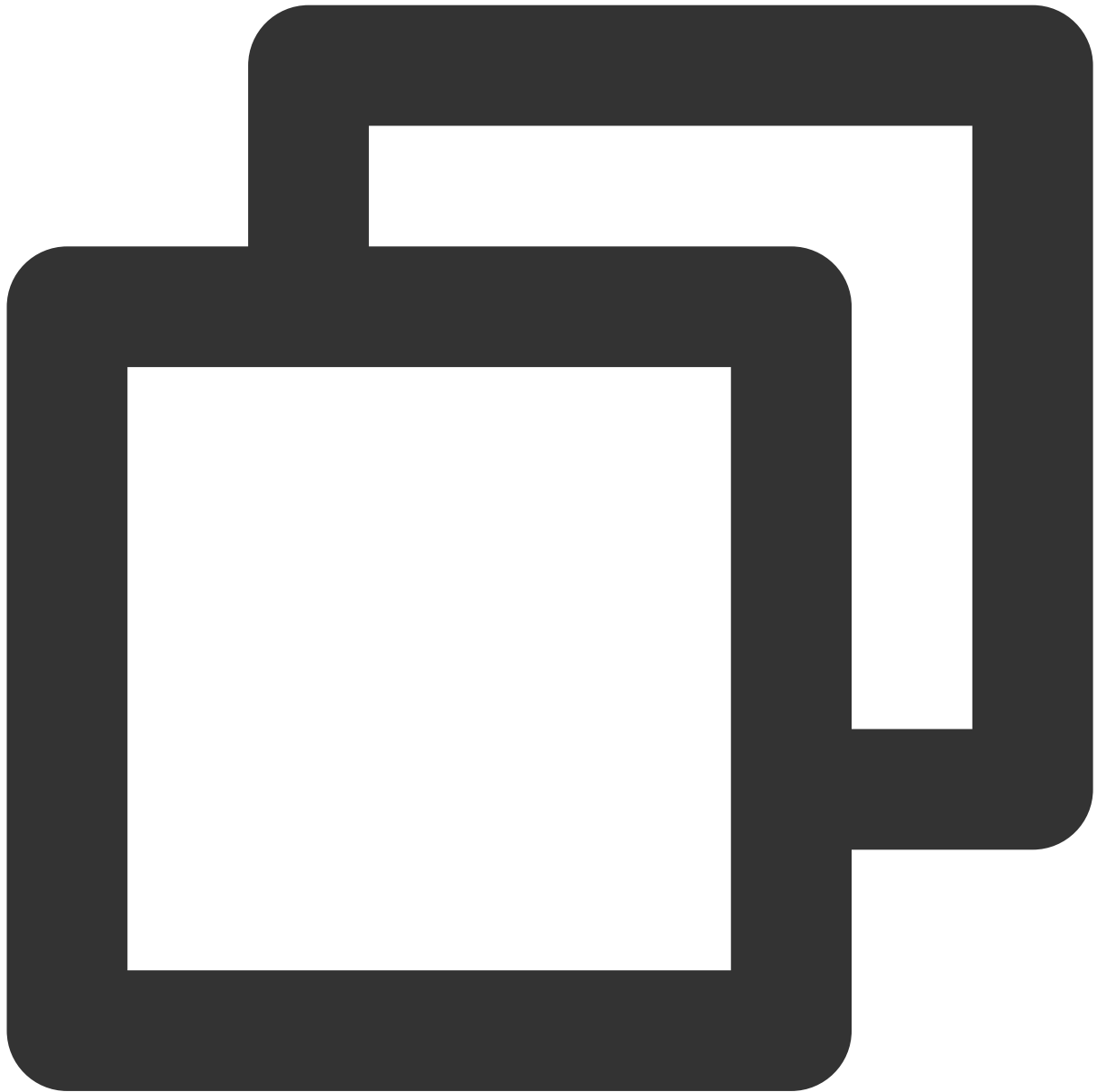
Parameter:

Parameter	Type	Description	Default Value	Meaning
maxSeatCount	number	Required	-	Set Room Seat Maximum Value

Returns : Promise<void>

## getSeatList

Get Seat List



```
const roomEngine = new TUIRoomEngine();  
const seatList = await roomEngine.getSeatList();
```

**Returns** : *Promise*<[TUISeatInfo](#)[]> seatList  
seatList for the Current Room's All Seat List

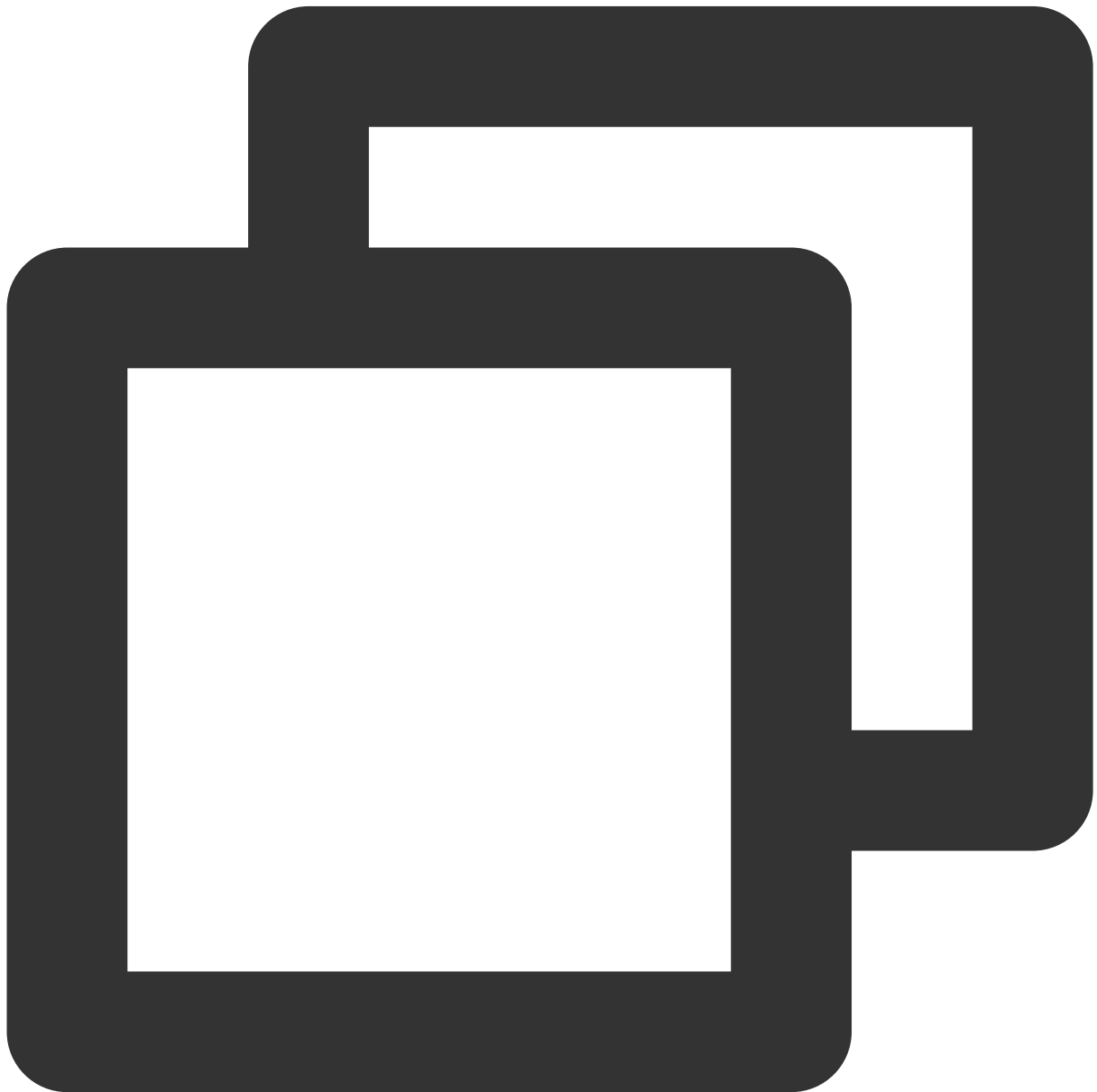
### takeSeat

Mic-off Users can call takeSeat to become Mic-on Users, only Mic-on Users can Publish Local stream.

When `roomInfo.roomType` is `TUIRoomType.kConference` and `roomInfo.speechMode` is `TUISpeechMode.kSpeakAfterTakingSeat`, General Users need to Wait for the Host/Administrator's Agreement to become Mic-on Users after calling `takeSeat` method.

When `roomInfo.roomType` is `TUIRoomType.kLivingRoom` and `roomInfo.speechMode` is `TUISpeechMode.kFreeToSpeak`, General Users become Mic-on Users after calling `takeSeat` method Successfully. Host & Administrator become Mic-on Users after calling `takeSeat` Successfully.

Changes of Mic-on Users are Notified to All Users through `TUIRoomEvents.onSeatListChanged`.



```
const roomEngine = new TUIRoomEngine();  
// Scenario 1: Host/Administrator Go Live
```

```
// Scenario 2: When roomInfo.roomType is TUIRoomType.kConference
// and roomInfo.speechMode is TUISpeechMode.kSpeakAfterTakingSeat, General Users Go
await roomEngine.takeSeat({
  seatIndex: -1,
  timeout: 0,
});

// Scenario 3: When roomInfo.enableSeatControl is true, General Users Go Live
const requestId = await roomEngine.instance?.takeSeat({
  seatIndex: -1,
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
    switch (requestCallbackType) {
      case TUIRequestCallbackType.kRequestAccepted:
        // Request Accepted
        break;
      case TUIRequestCallbackType.kRequestRejected:
        // Request Rejected
        break;
      case TUIRequestCallbackType.kRequestCancelled:
        // Request Canceled
        break;
      case TUIRequestCallbackType.kRequestTimeout:
        // Request Timeout
        break;
      case TUIRequestCallbackType.kRequestError:
        // Request Error
        break;
      default:
        break;
    }
  },
});
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Seat index, set to -1 when there is No Serial Number
timeout	number	Required	-	Timeout. If timeout is set to 0, there is no Timeout
requestCallback	Function	Optional	Empty Function	Callback, used to Notify Initiator of Request being Accepted/Rejected/Cancelled/Timeout/Error

**Returns** : Promise<string> requestId

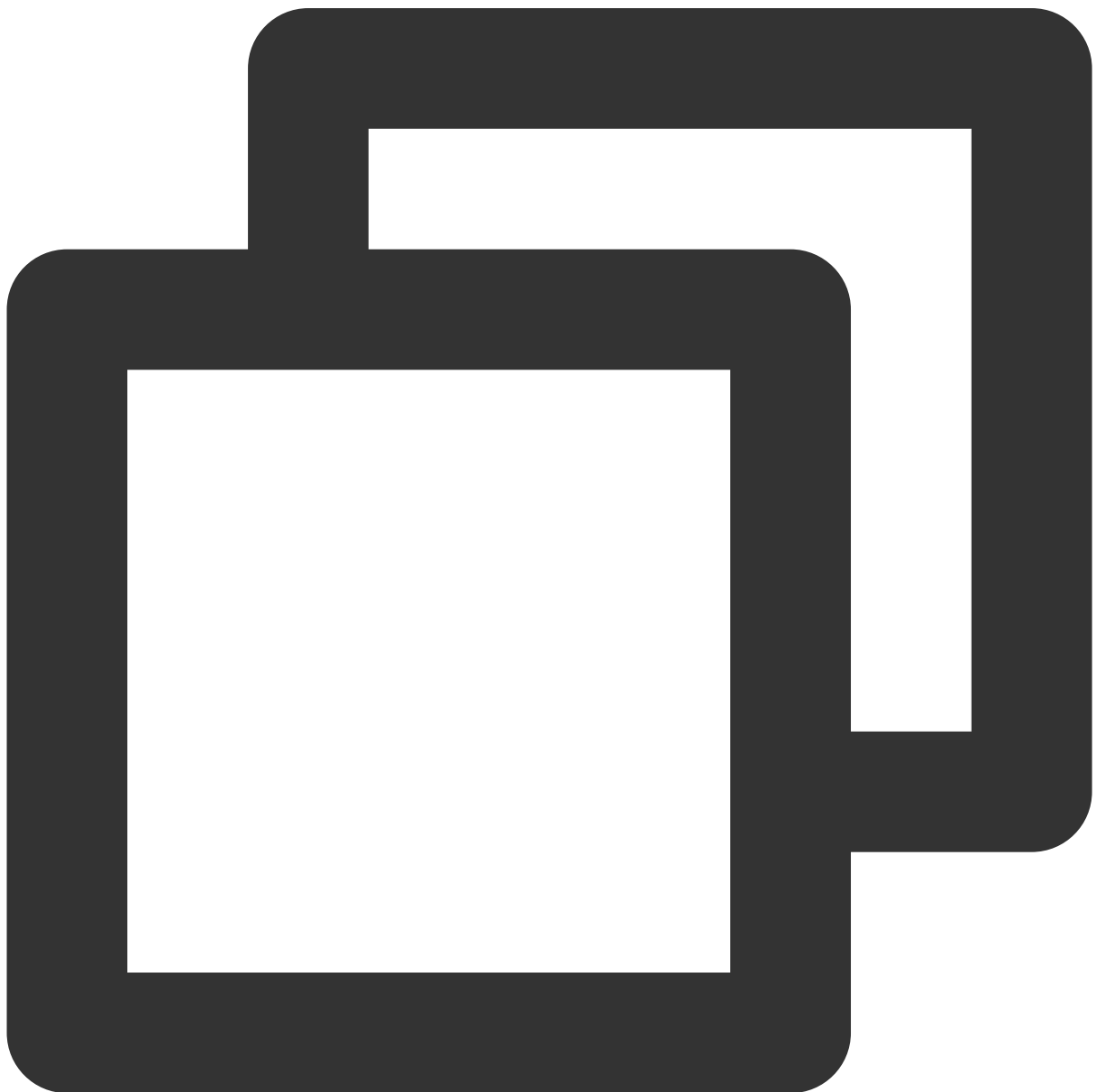
When roomInfo.enableSeatControl is true, General Users call this Interface to return requestId, General Users can use this requestId to call cancelRequest Interface to cancel Go Live Request.

**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;

## leaveSeat

Release Seat.

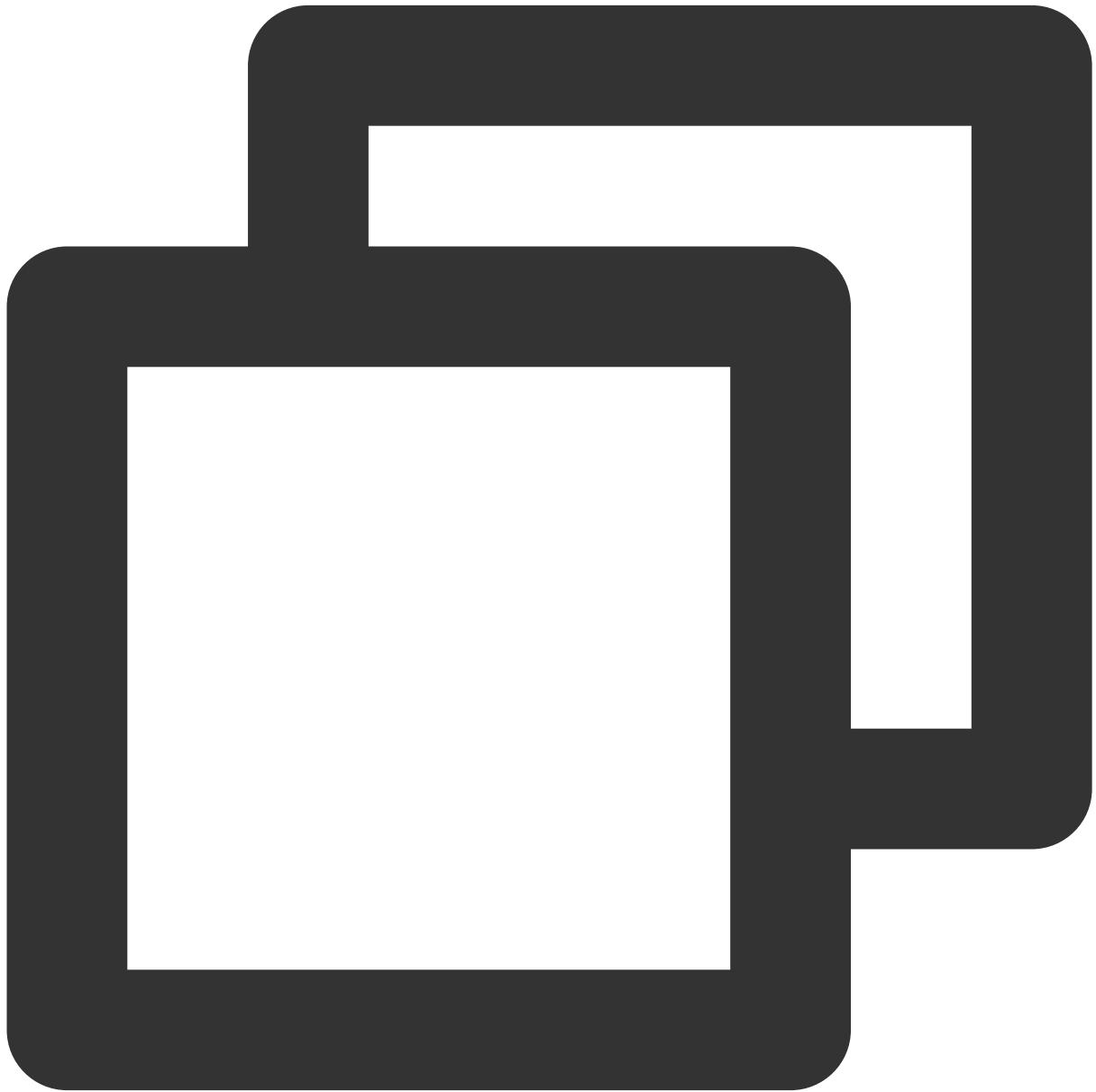


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.leaveSeat();
```

**Returns** : *Promise<void>*

### takeUserOnSeatByAdmin

Invite Others to Go Live.



```
const roomEngine = new TUIRoomEngine();  
const requestId = roomEngine.takeUserOnSeatByAdmin({  
  seatIndex: 0,
```

```
userId: 'user_1234',
timeout: 0,
requestCallback: ({ requestCallbackType, requestId, userId, code, message }) =>
  switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // Request Accepted
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // Request Rejected
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // Request Canceled
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // Request Timeout
      break;
    case TUIRequestCallbackType.kRequestError:
      // Request Error
      break;
    default:
      break;
  }
},
});
```

**Parameter:**

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Seat index
userId	string	Required	-	User ID
timeout	number	Required	-	Timeout. If timeout is set to 0, there is no Timeout
requestCallback	Function	Optional	Empty Function	Callback, used to Notify Initiator of Request being Accepted/Rejected/Cancelled/Timeout/Error

**Returns :** *Promise<string> requestId*

This Interface returns requestId, Users can use this requestId to call cancelRequest Interface to cancel Request.

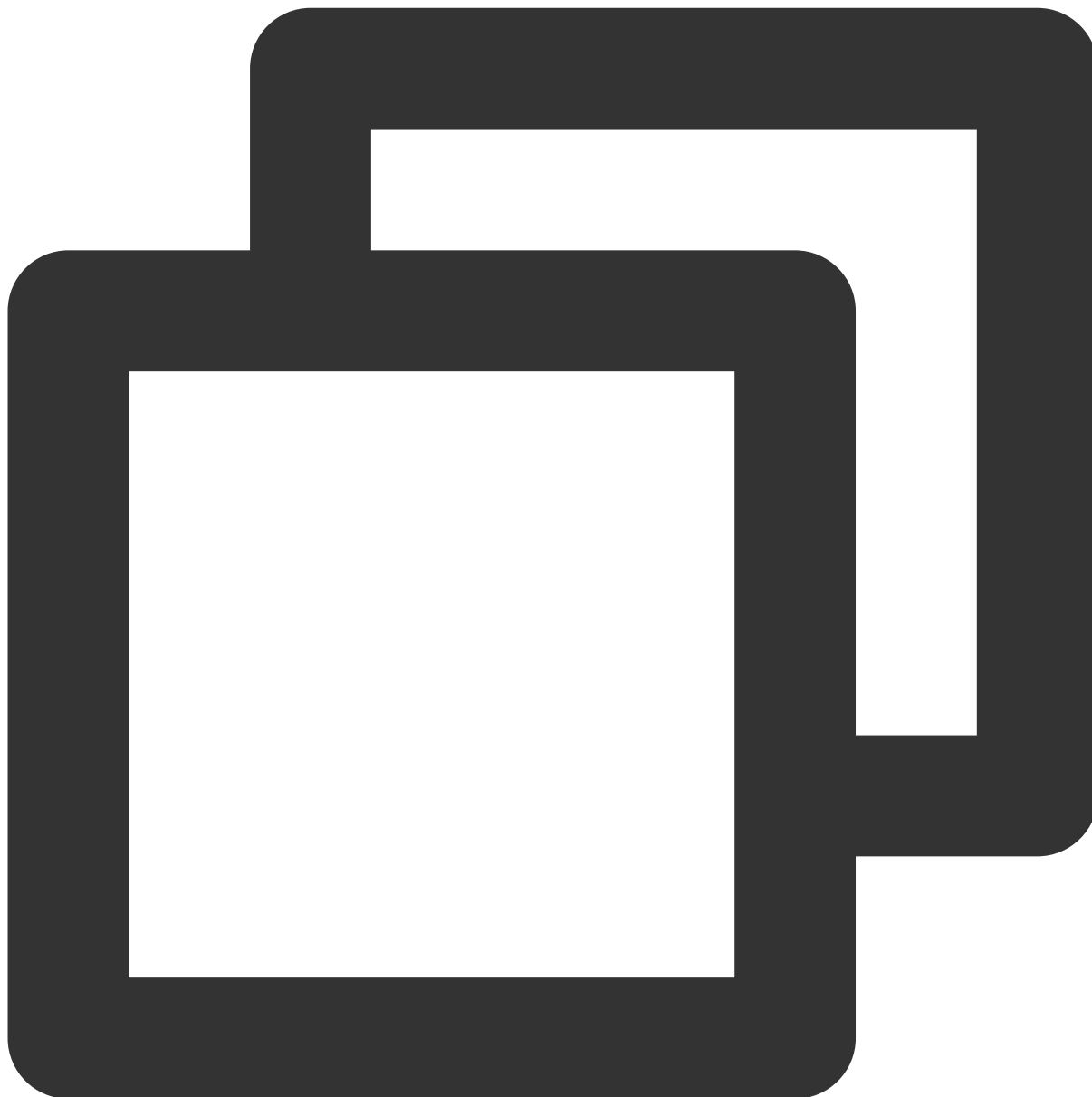
**Note:**

In v1.0.2 and above, the requestId returned by this interface is of type string; in v1.0.0 and v1.0.1, the requestId returned by this interface is of type number;



## kickUserOffSeatByAdmin

Ask others to get off the mic



```
const roomEngine = new TUIRoomEngine();
const requestId = await roomEngine.kickUserOffSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
});
```

Parameter:

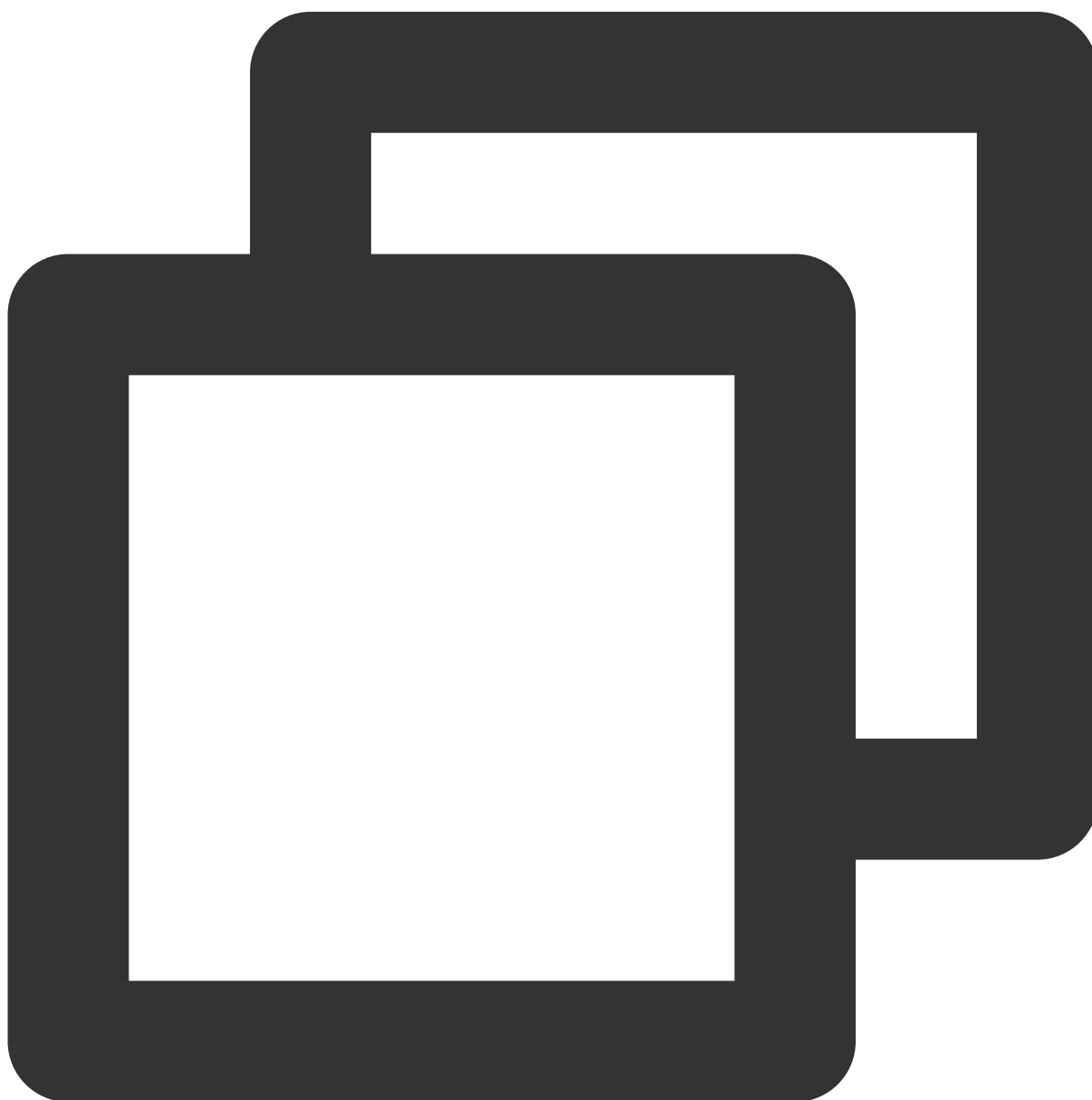
--	--	--	--	--

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Seat index
userId	string	Required	-	User ID

**Returns** : *Promise<void>*

### lockSeatByAdmin

Lock a certain mic position status (Only the room host and administrator can call this method).



```
const roomEngine = new TUIRoomEngine();
await roomEngine.lockSeatByAdmin({
  seatIndex: 0,
  lockParams: {
    lockSeat: true,
    lockVideo: true,
    lockAudio: true,
  },
});
```

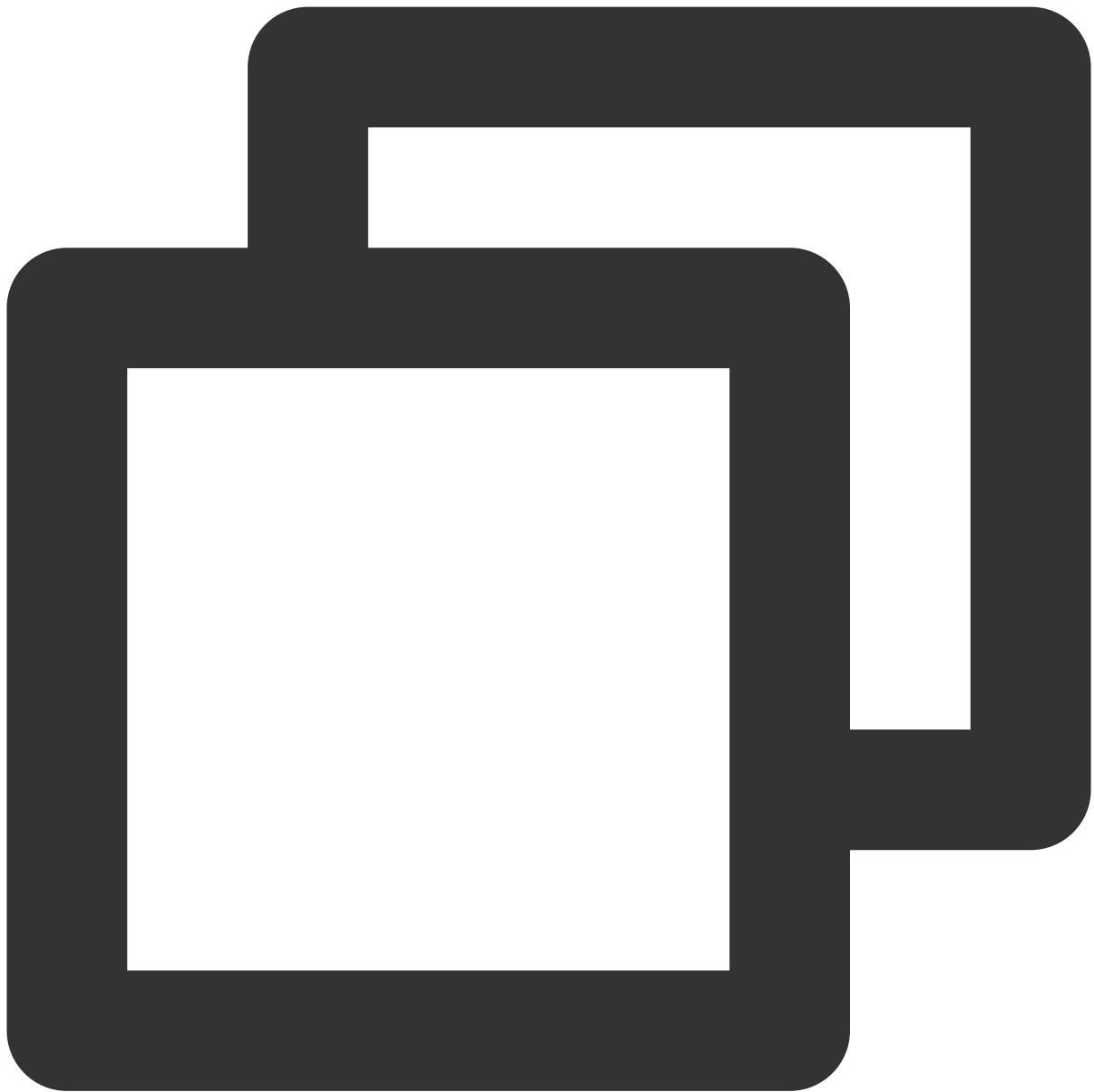
Parameter:

Parameter	Type	Description	Default Value	Meaning
seatIndex	number	Required	-	Mic position index
lockParams	<a href="#">TUISeatLockParams</a>	Required	-	Lock mic parameter

Returns : *Promise<void>*

## startScreenSharingElectron

Start Screen Sharing Electron。



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startScreenSharingElectron('targetId');
```

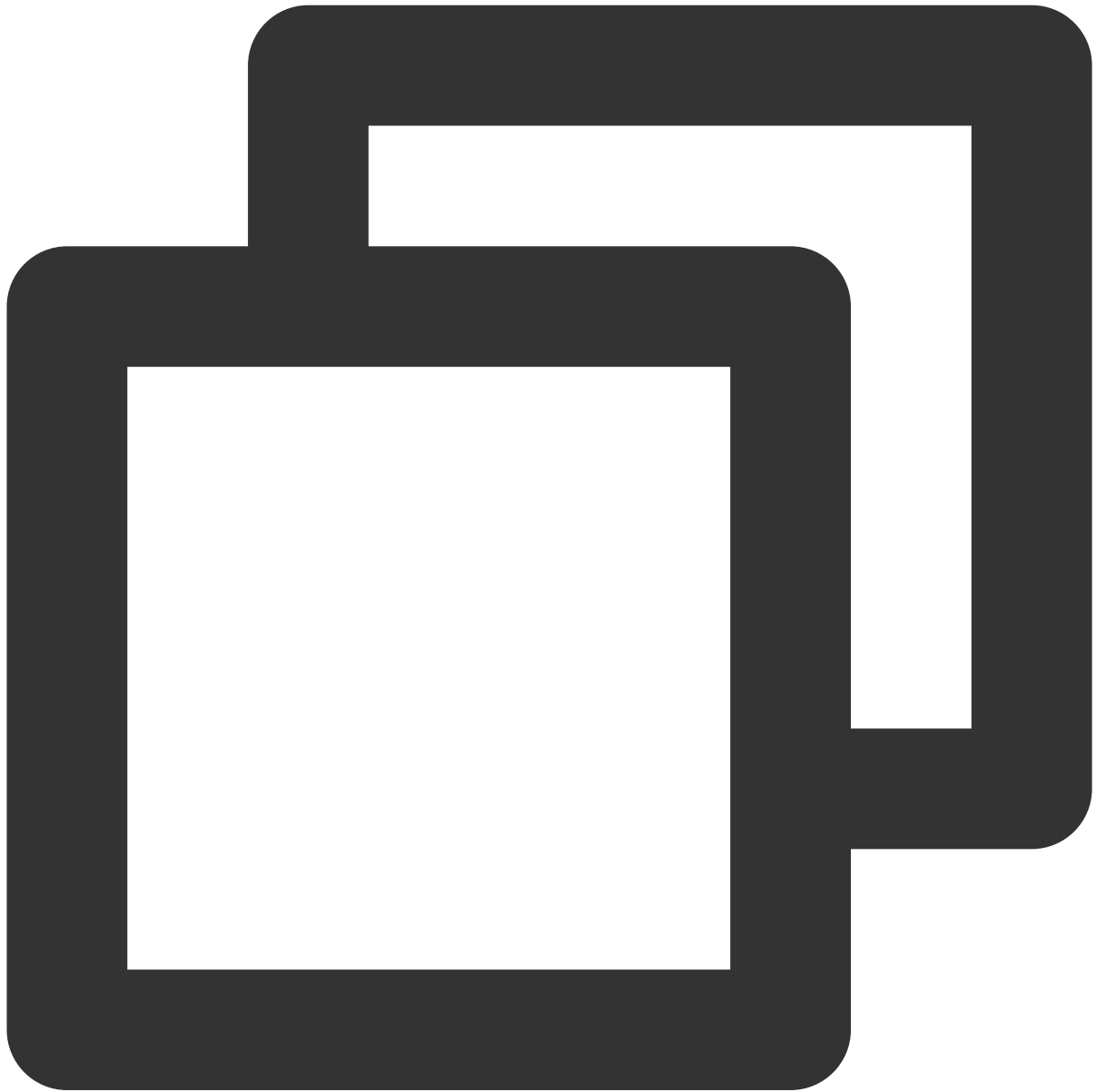
Parameter:

Parameter	Type	Description	Default Value	Meaning
targetId	string	Optional	-	Sharing Window ID, can be obtained from getScreenSharingTarget

**Returns** : *Promise<void>*

## **stopScreenSharingElectron**

Stop Screen Sharing Electron.

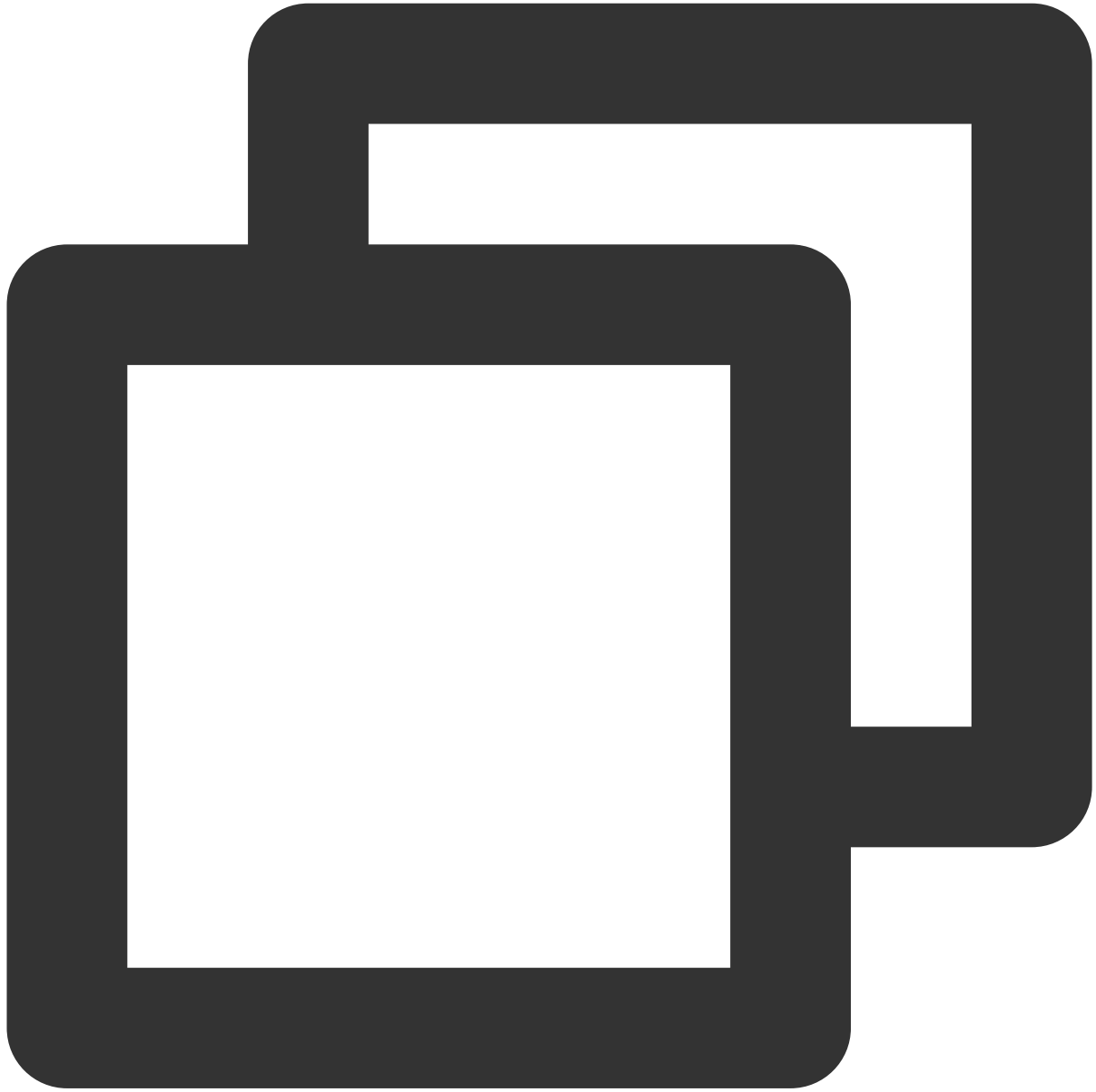


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopScreenSharingElectron();
```

**Returns** : *Promise<void>*

## **getScreenSharingTarget**

Get Screen Sharing list Electron。

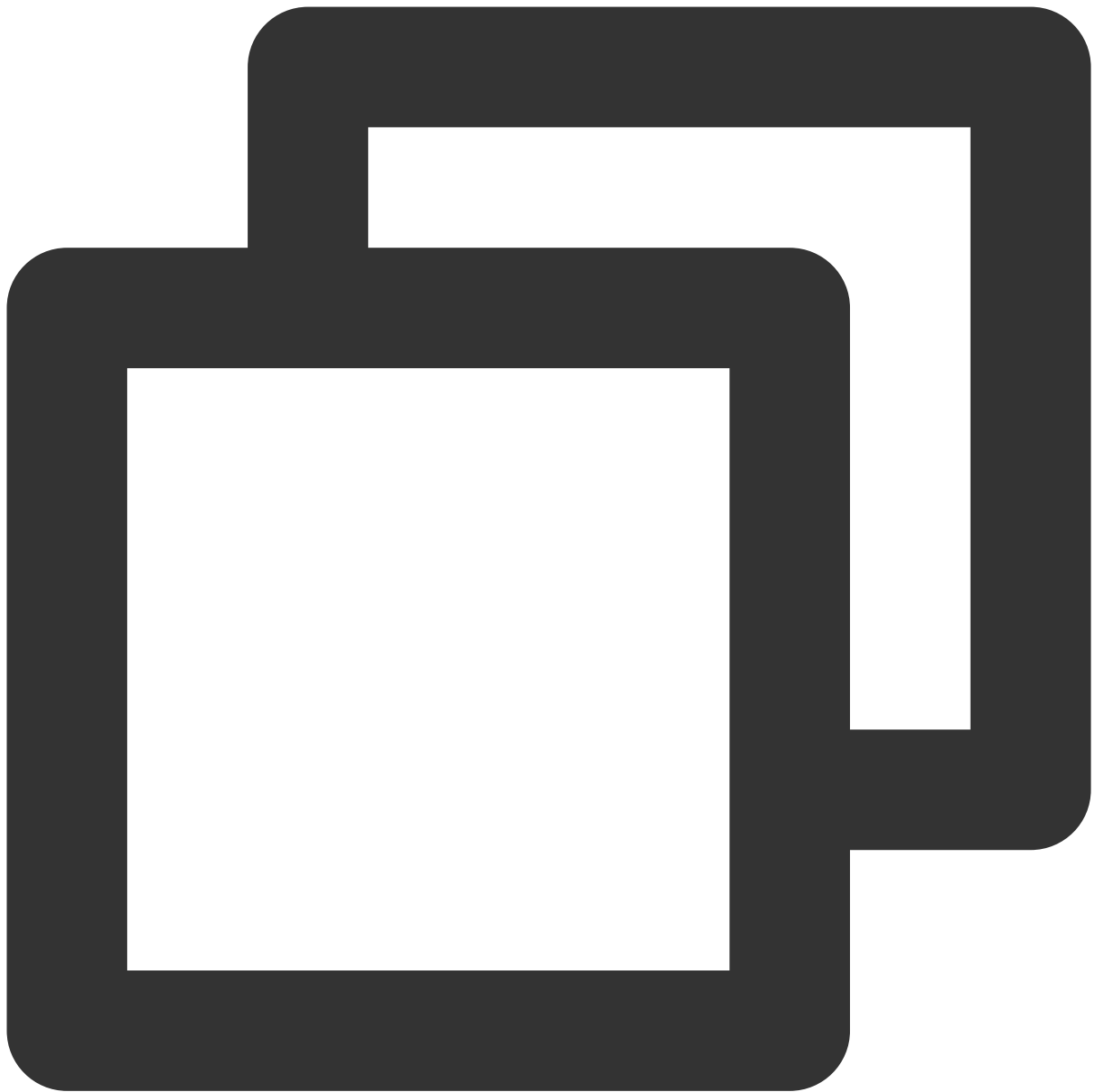


```
const roomEngine = new TUIRoomEngine();  
const screenList = await roomEngine.getScreenSharingTarget();
```

**Returns** : *Promise<void>*

### **selectScreenSharingTarget**

Switch Screen Sharing Window Electron。



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.selectScreenSharingTarget('targetId');
```

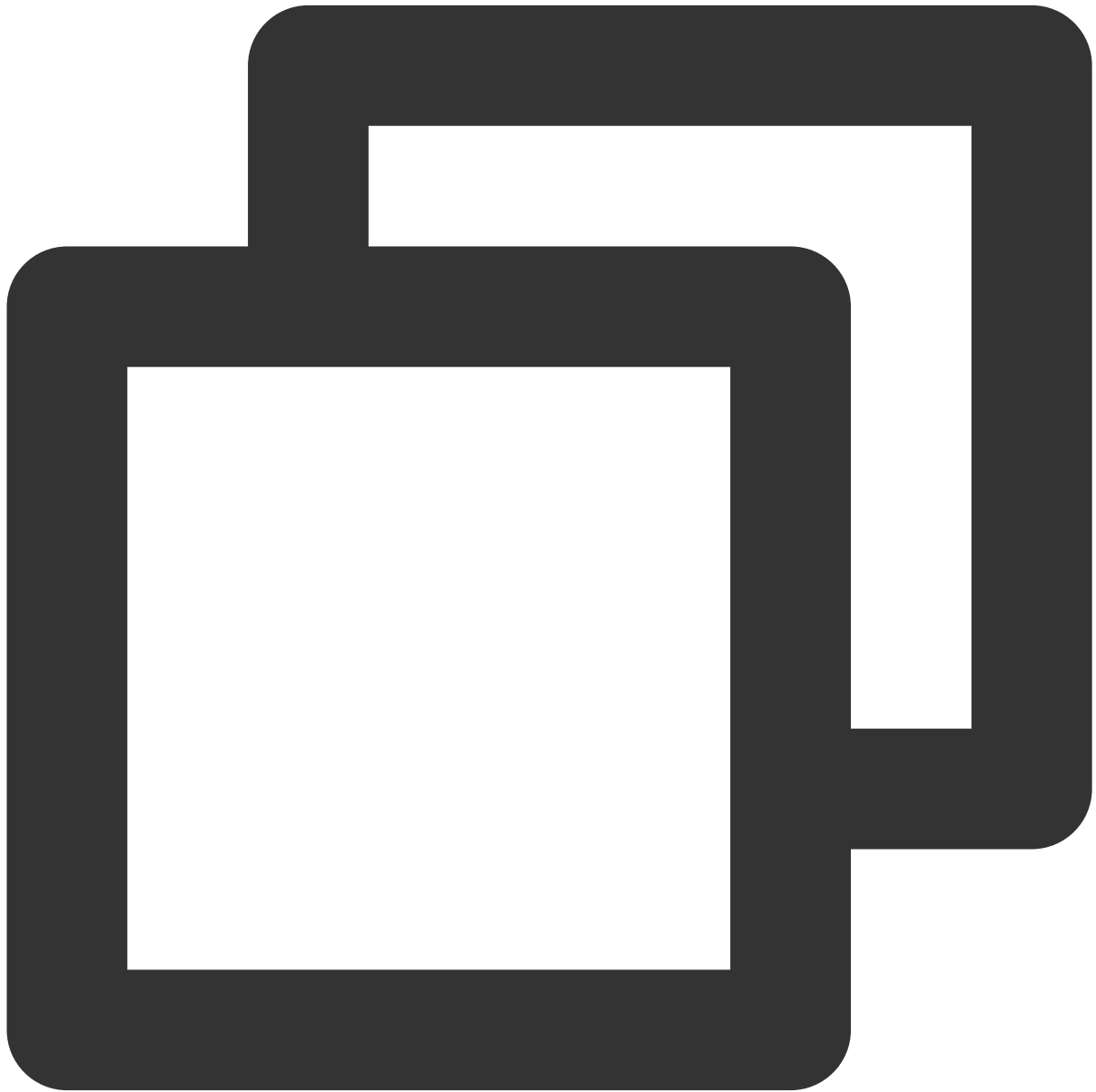
Parameter:

Parameter	Type	Description	Default Value	Meaning
targetId	string	Required	-	Sharing Window ID, can be obtained from <code>getScreenSharingTarget</code>

**Returns** : *Promise<void>*

## **sendTextMessage**

Send text message.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendTextMessage({
  messageText: 'hello, everyone',
});
```

**Parameter:**

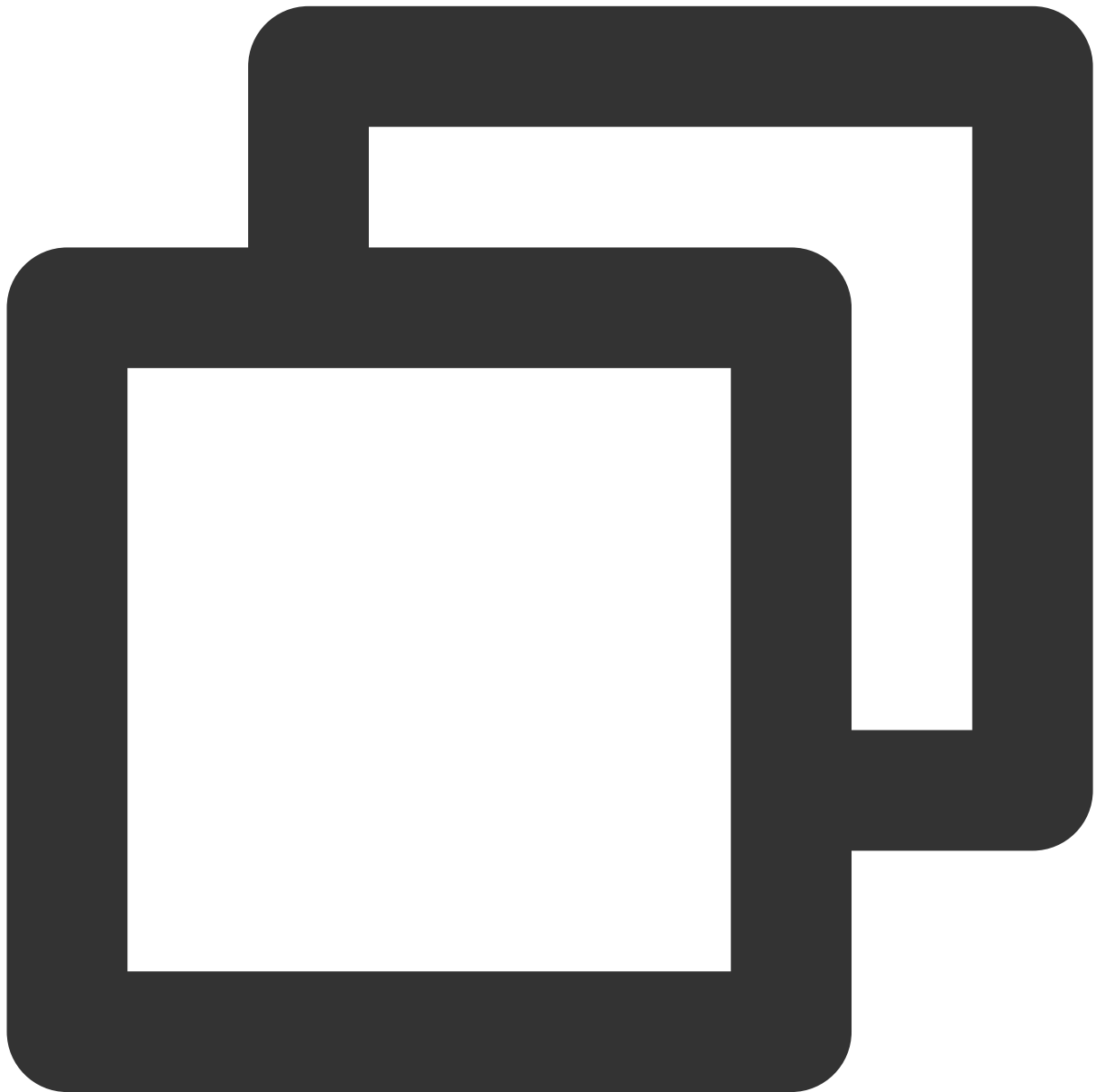


Parameter	Type	Description	Default Value	Meaning
messageText	string	Required	-	Text message content

**Returns** : *Promise<void>*

## sendCustomMessage

Send custom message.



```
const roomEngine = new TUIRoomEngine();
```

```
await roomEngine.sendCustomMessage({
  messageText: '{ data:', description: ''}',
});
```

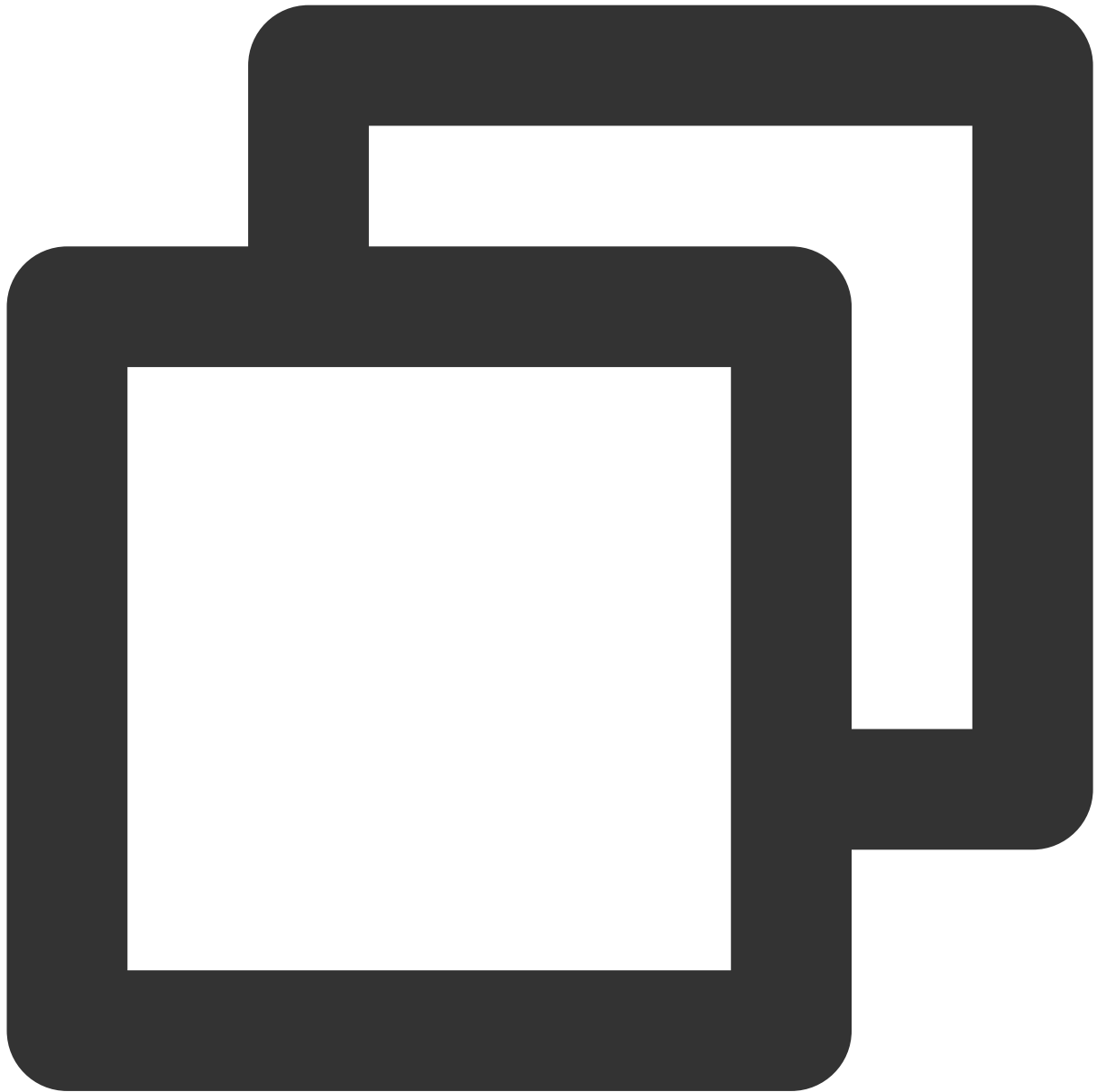
Parameter:

Parameter	Type	Description	Default Value	Meaning
messageText	string	Required	-	Custom message content

**Returns** : *Promise<void>*

## getCameraDevicesList

Get camera device list.

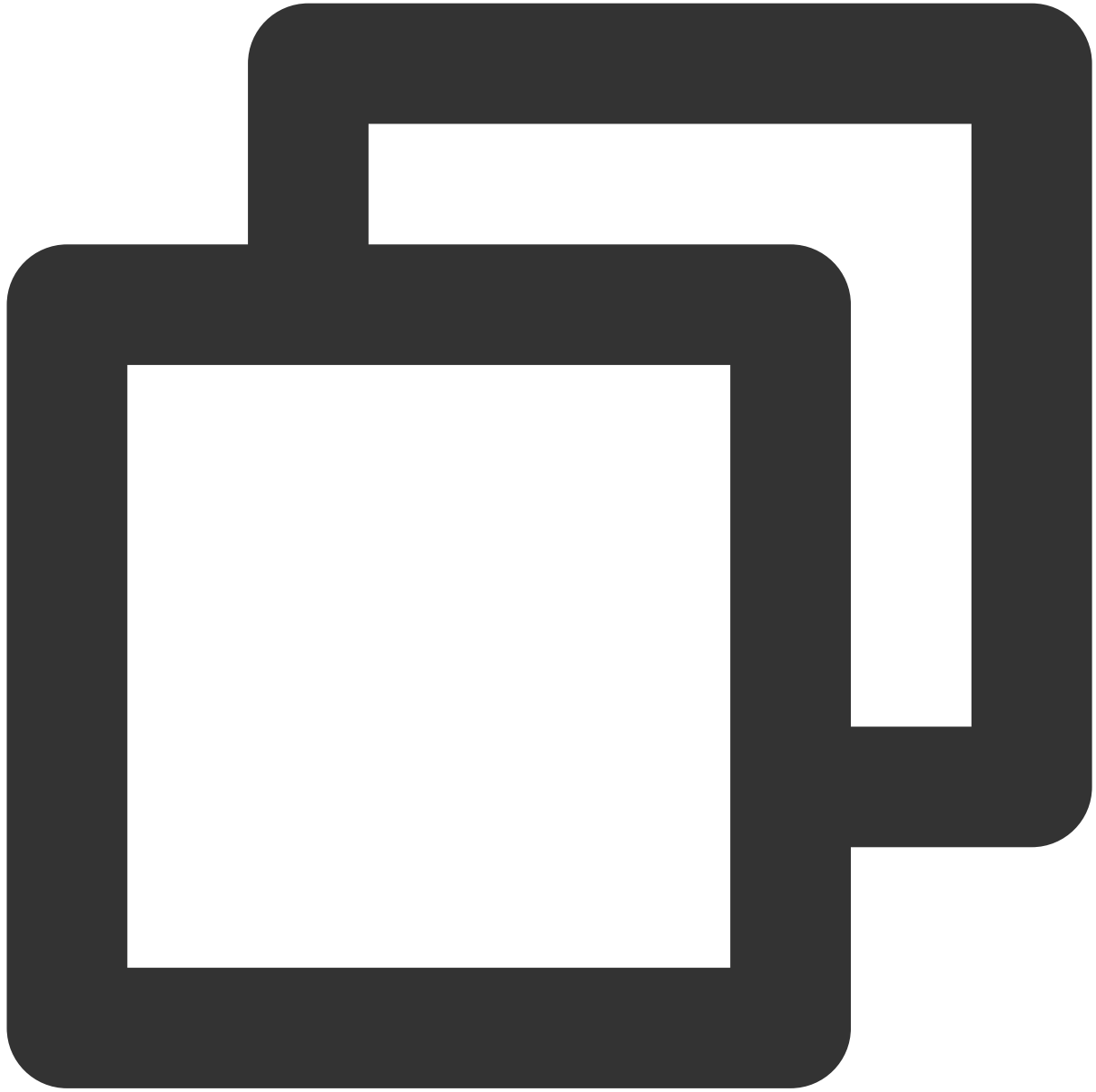


```
const roomEngine = new TUIRoomEngine();
const cameraList = await roomEngine.getCameraDevicesList();
for (i = 0; i < cameraList.length; i++) {
  var camera = cameraList[i];
  console.info("camera deviceName: " + camera.deviceName + " deviceId:" + camera.
}
```

**Returns:** *Promise*<[TRTCDeviceInfo\[\]](#)> cameraList

### getMicDevicesList

Get mic device list.

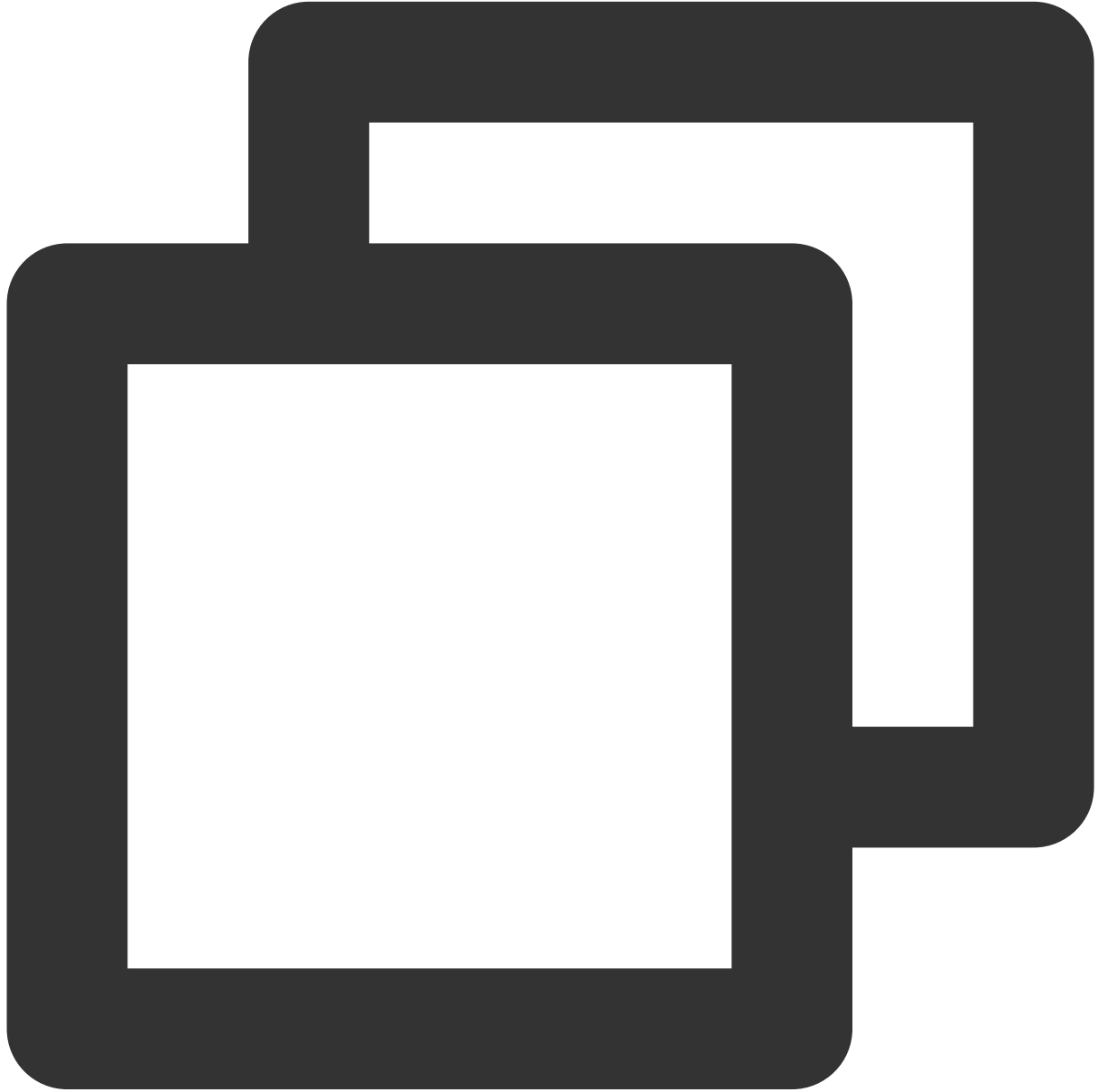


```
const roomEngine = new TUIRoomEngine();
const micList = await roomEngine.getMicDevicesList();
for (i = 0; i < micList.length; i++) {
  var mic = micList[i];
  console.info("mic deviceName: " + mic.deviceName + " deviceId:" + mic.deviceId)
}
```

**Returns:** *Promise*<[TRTCDeviceInfo\[\]](#)> micList

## getSpeakerDevicesList

Get speaker device list.

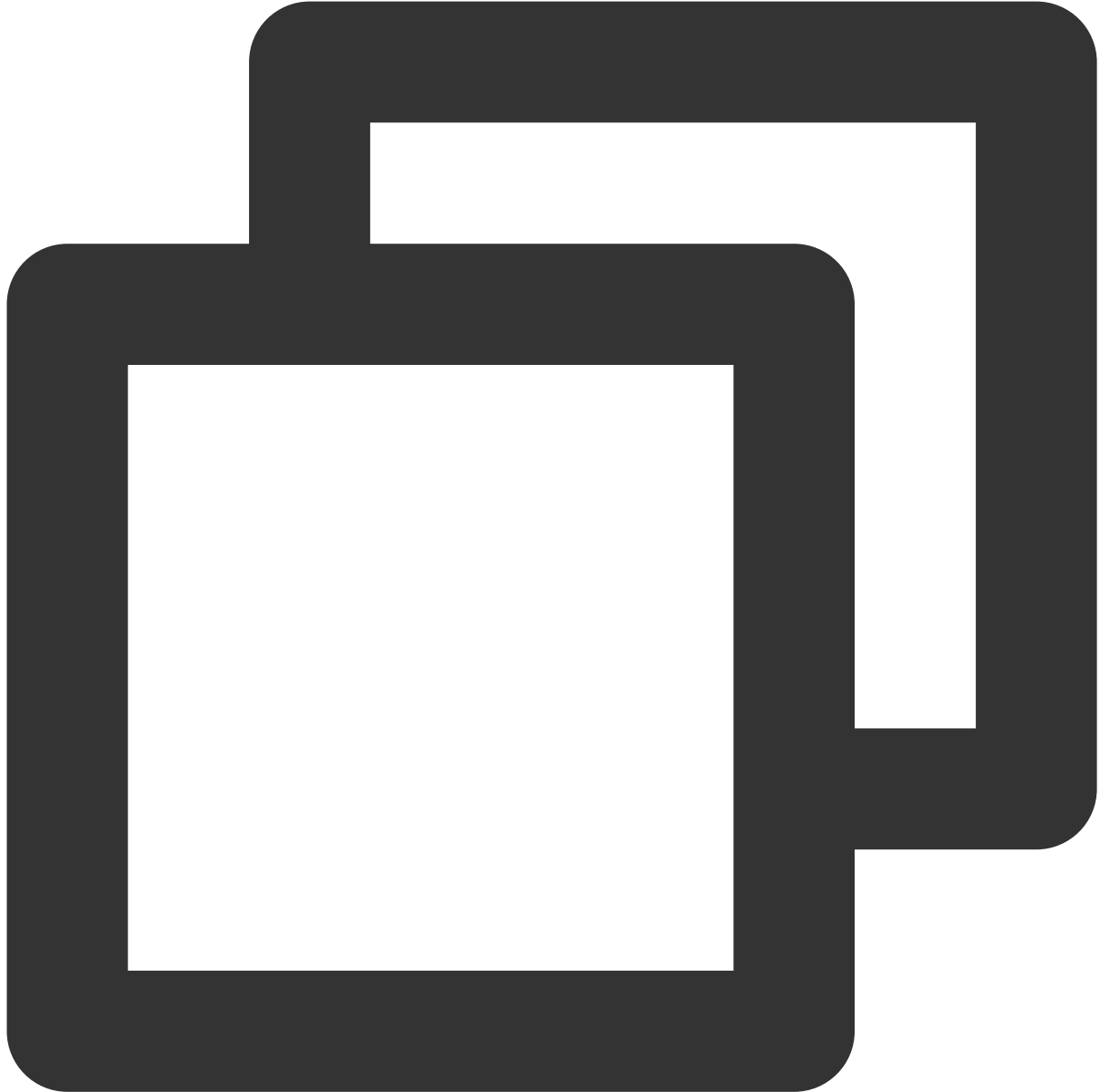


```
const roomEngine = new TUIRoomEngine();
const speakerList = await roomEngine.getSpeakerDevicesList();
for (i = 0; i < speakerList.length; i++) {
  var speaker = speakerList[i];
  console.info("speaker deviceName: " + speaker.deviceName + " deviceId:" + speaker.deviceId);
}
```

**Returns:** *Promise*<[TRTCDeviceInfo\[\]](#)> speakerList

## setCurrentCameraDevice

Set the camera device to be used.



```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentCameraDevice({ deviceId: '' });
```

Parameter:

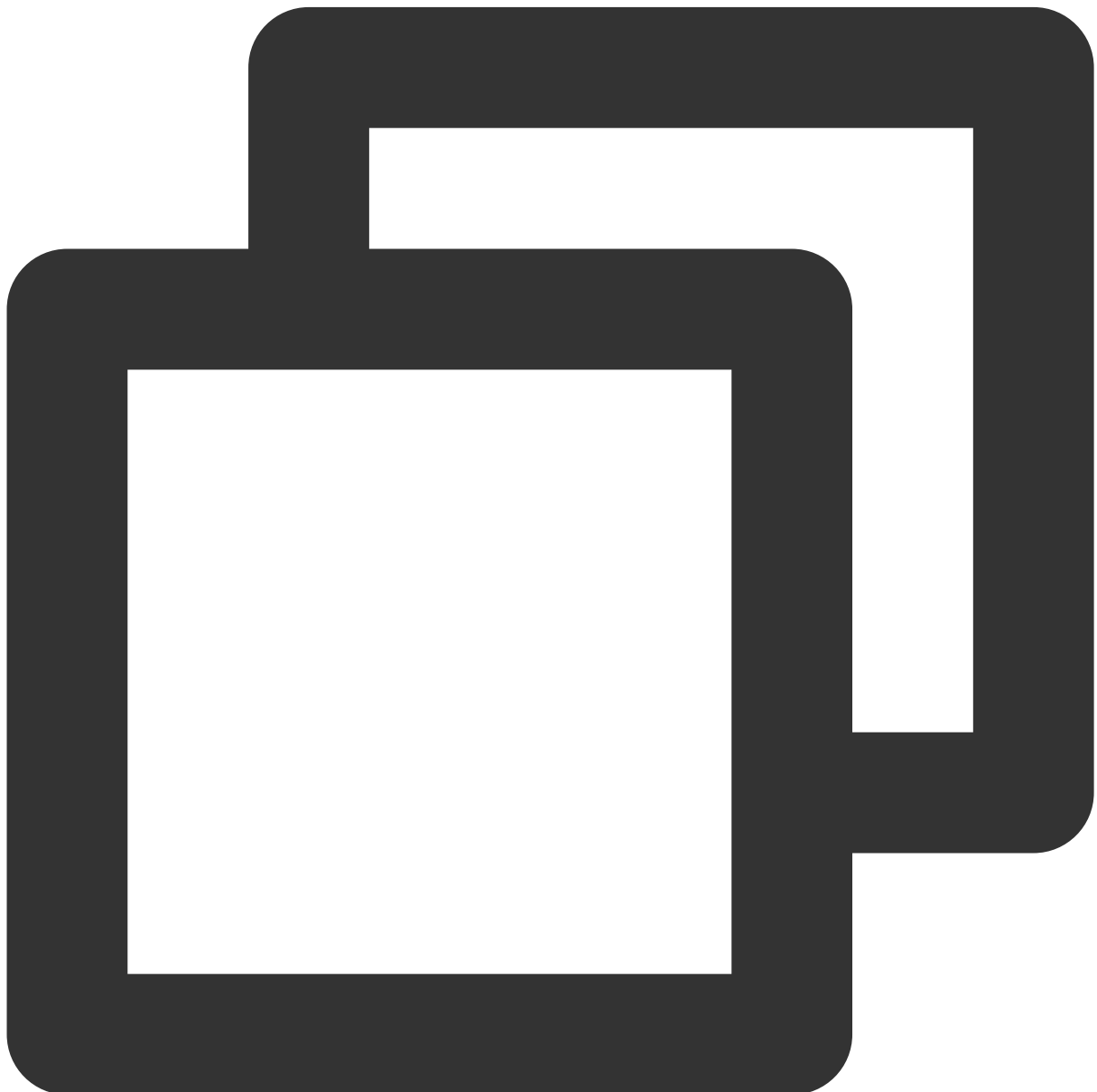
Parameter	Type	Description	Default Value	Meaning

deviceId	string	Required	-	Device ID obtained from getCameraDevicesList
----------	--------	----------	---	---

**Returns :** *void*

### **setCurrentMicDevice**

Set the mic device to be used.



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.setCurrentMicDevice({ deviceId: '' });
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
deviceId	string	Required	-	Device ID obtained from getSpeakerDevicesList

**Returns :** *void*

### **setCurrentSpeakerDevice**

Get the currently used camera device.





```
const roomEngine = new TUIRoomEngine();  
await roomEngine.setCurrentSpeakerDevice({ deviceId: '' });
```

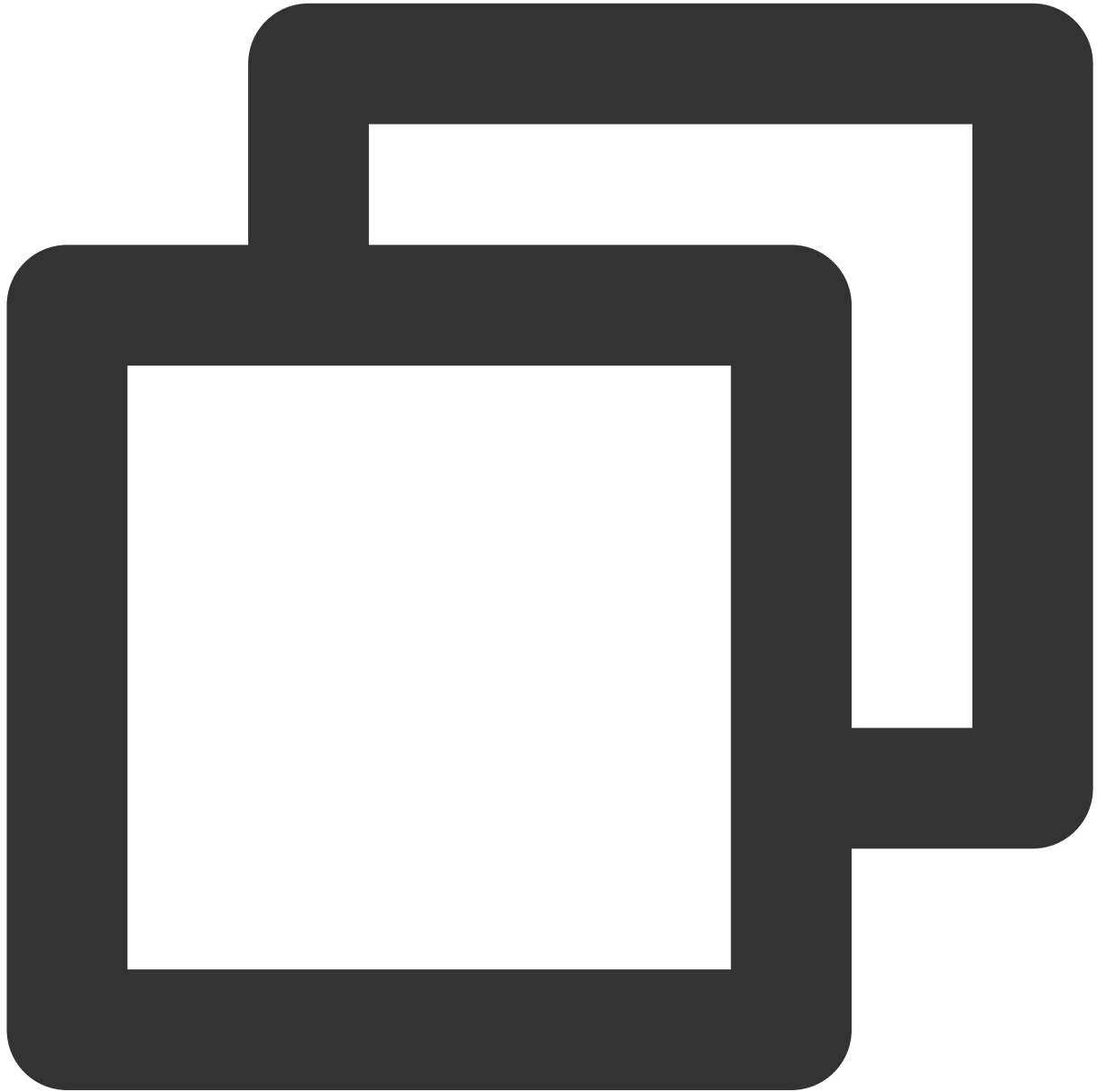
Parameter:

Parameter	Type	Description	Default Value	Meaning
deviceId	string	Required	-	Device ID obtained from getSpeakerDevicesList

**Returns :** *void*

### **getCurrentCameraDevice**

Get the currently used camera device.



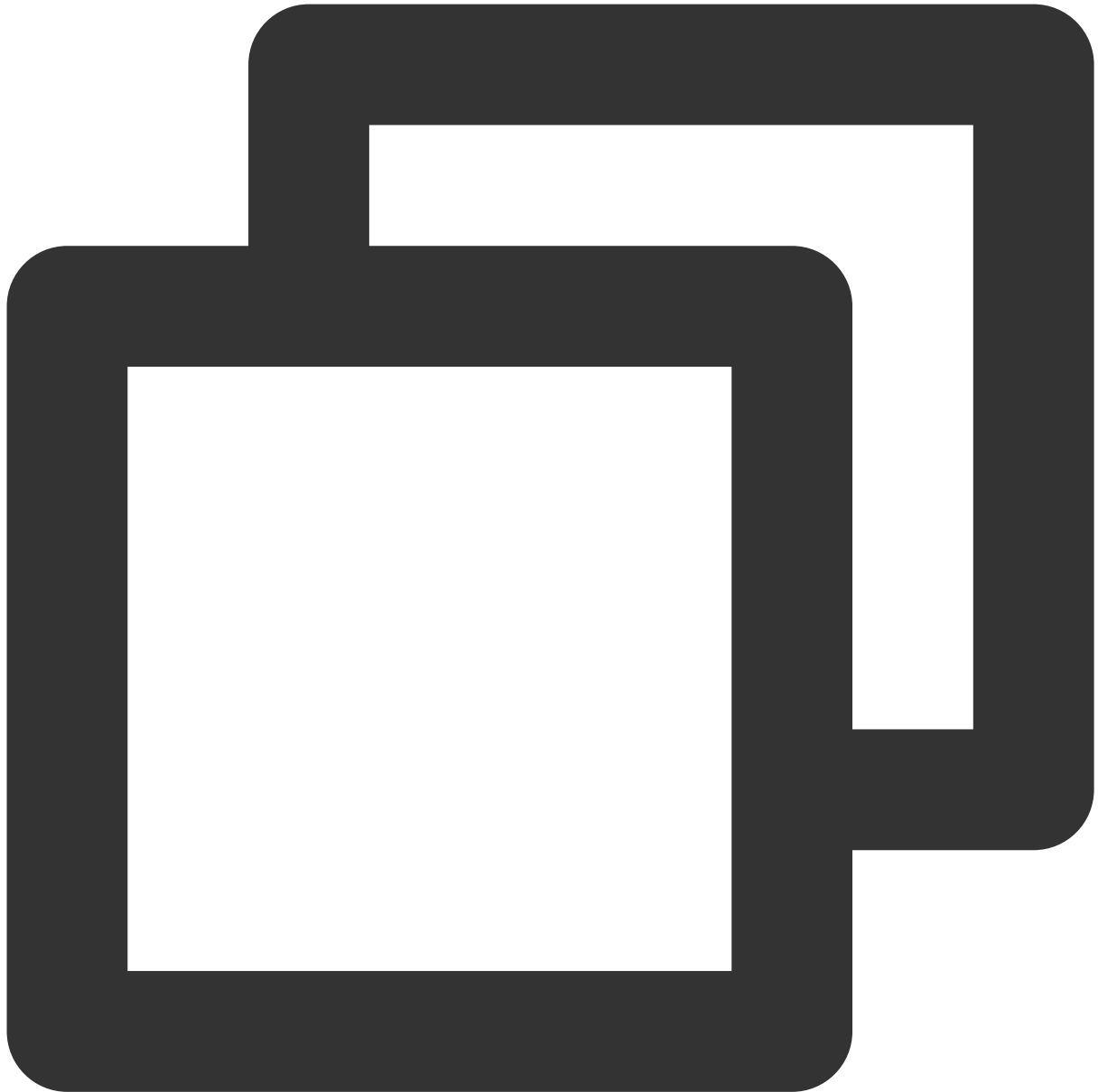
```
const roomEngine = new TUIRoomEngine();  
const currentCameraDevice = roomEngine.getCurrentCameraDevice();
```

**Returns :** [TRTCDeviceInfo](#)

Device information, can get device ID and device name

## getCurrentMicDevice

Get the currently used mic device.



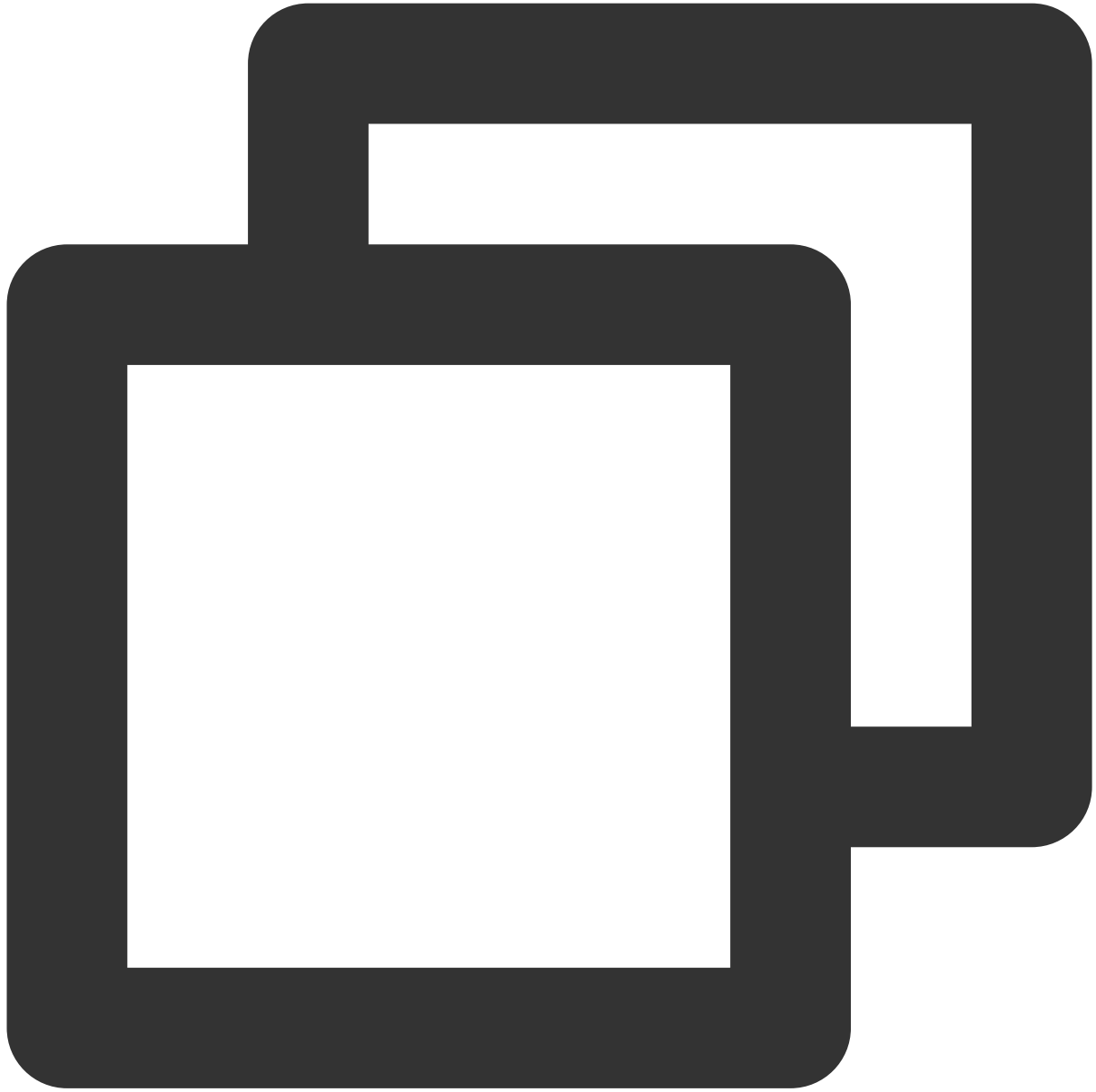
```
const roomEngine = new TUIRoomEngine();  
const currentMicDevice = roomEngine.getCurrentMicDevice();
```

**Returns :** [TRTCDeviceInfo](#)

Device information, can get device ID and device name

## getCurrentSpeakerDevice

Get the currently used speaker device.



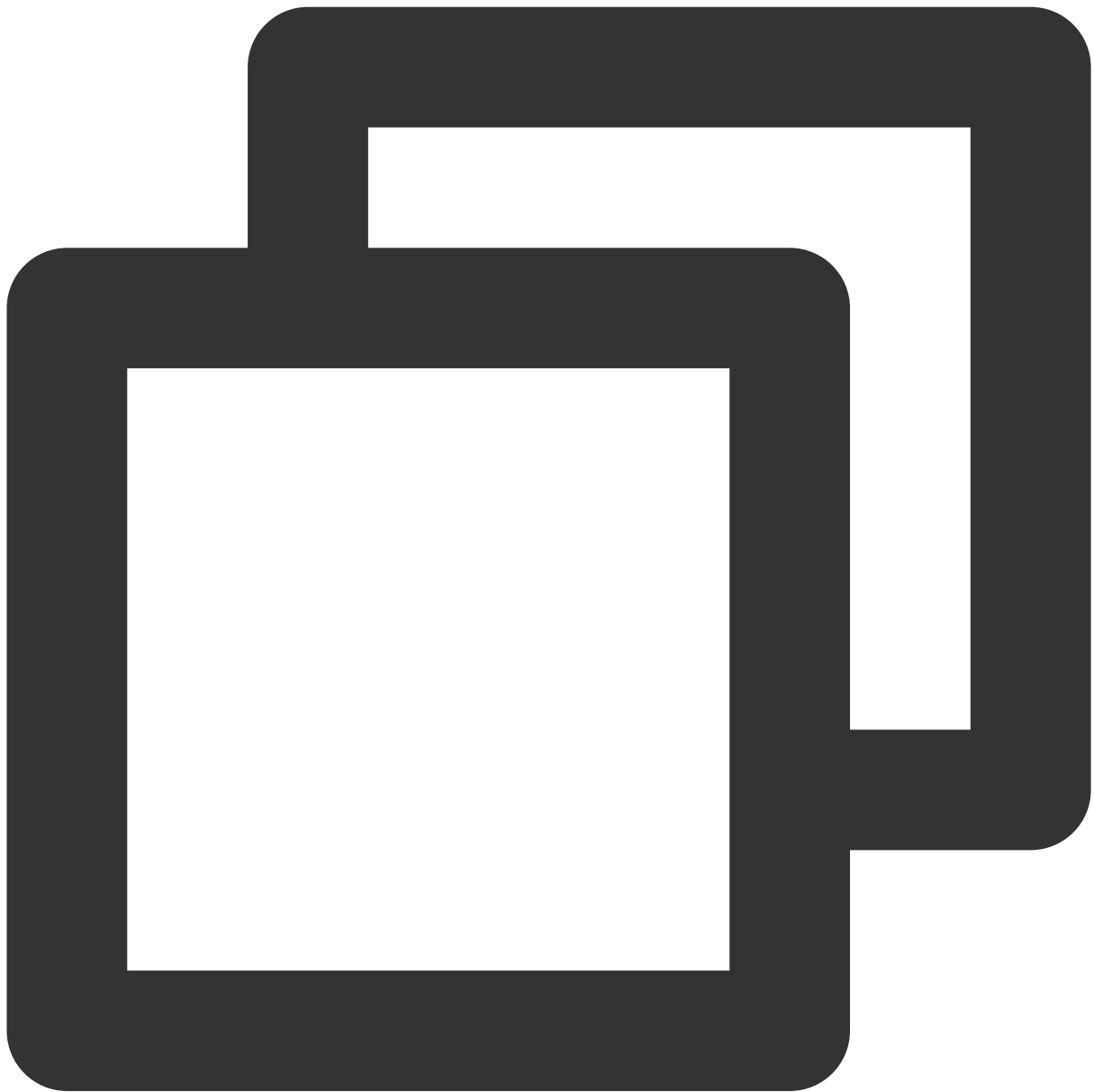
```
const roomEngine = new TUIRoomEngine();  
const currentSpeakerDevice = roomEngine.getCurrentSpeakerDevice();
```

**Returns :** [TRTCDeviceInfo](#)

Device information, can get device ID and device name

### **startCameraDeviceTest**

Start camera test.



```
const roomEngine = new TUIRoomEngine();  
await roomEngine.startCameraDeviceTest({ view: 'test-preview' });
```

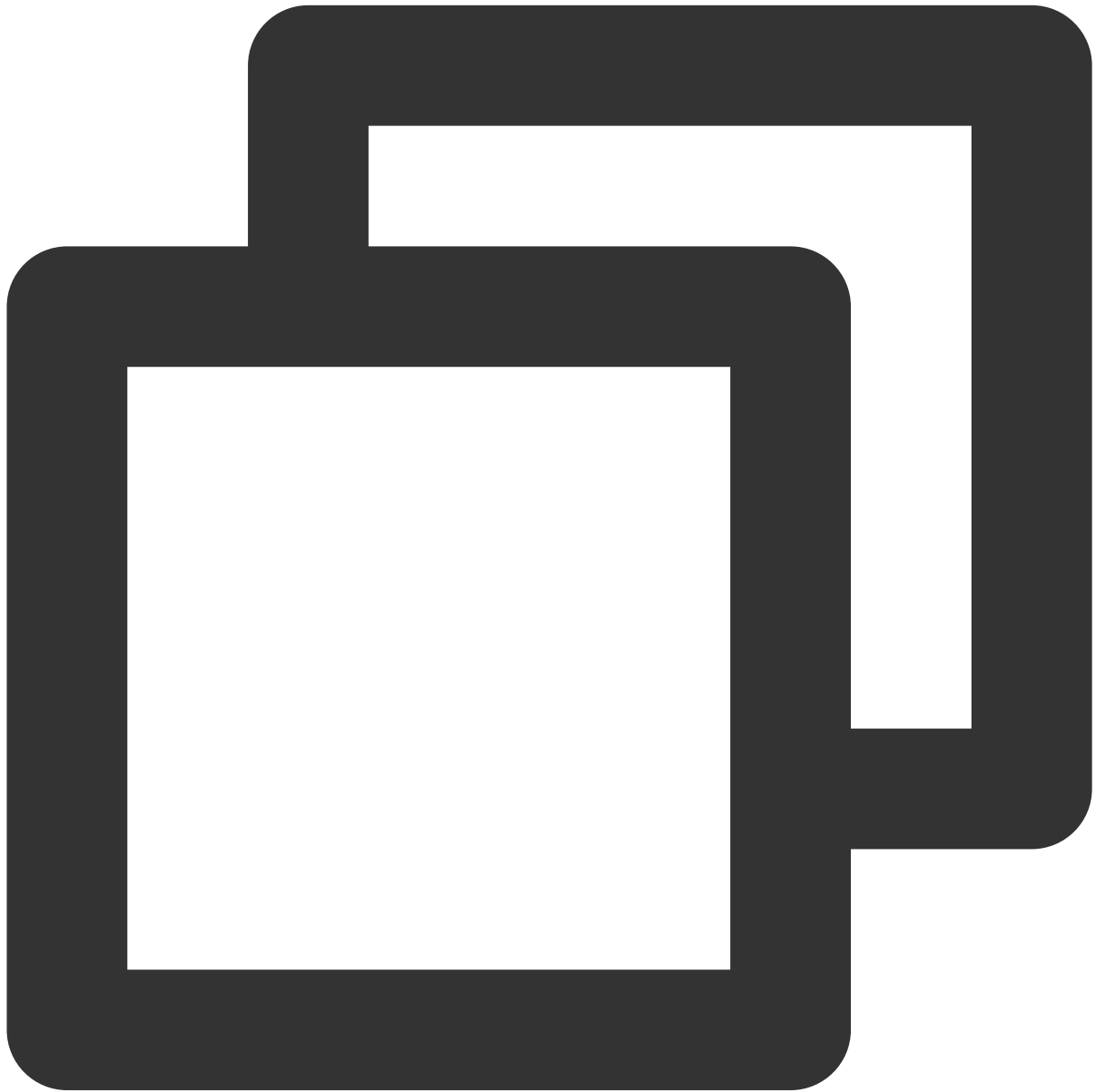
Parameter:

Parameter	Type	Description	Default Value	Meaning
view	string	Required	-	Display the video area of the camera test, the input view is the Id of the div element carrying the preview screen

**Returns** : *Promise<void>*

## **stopCameraDeviceTest**

Stop camera test.

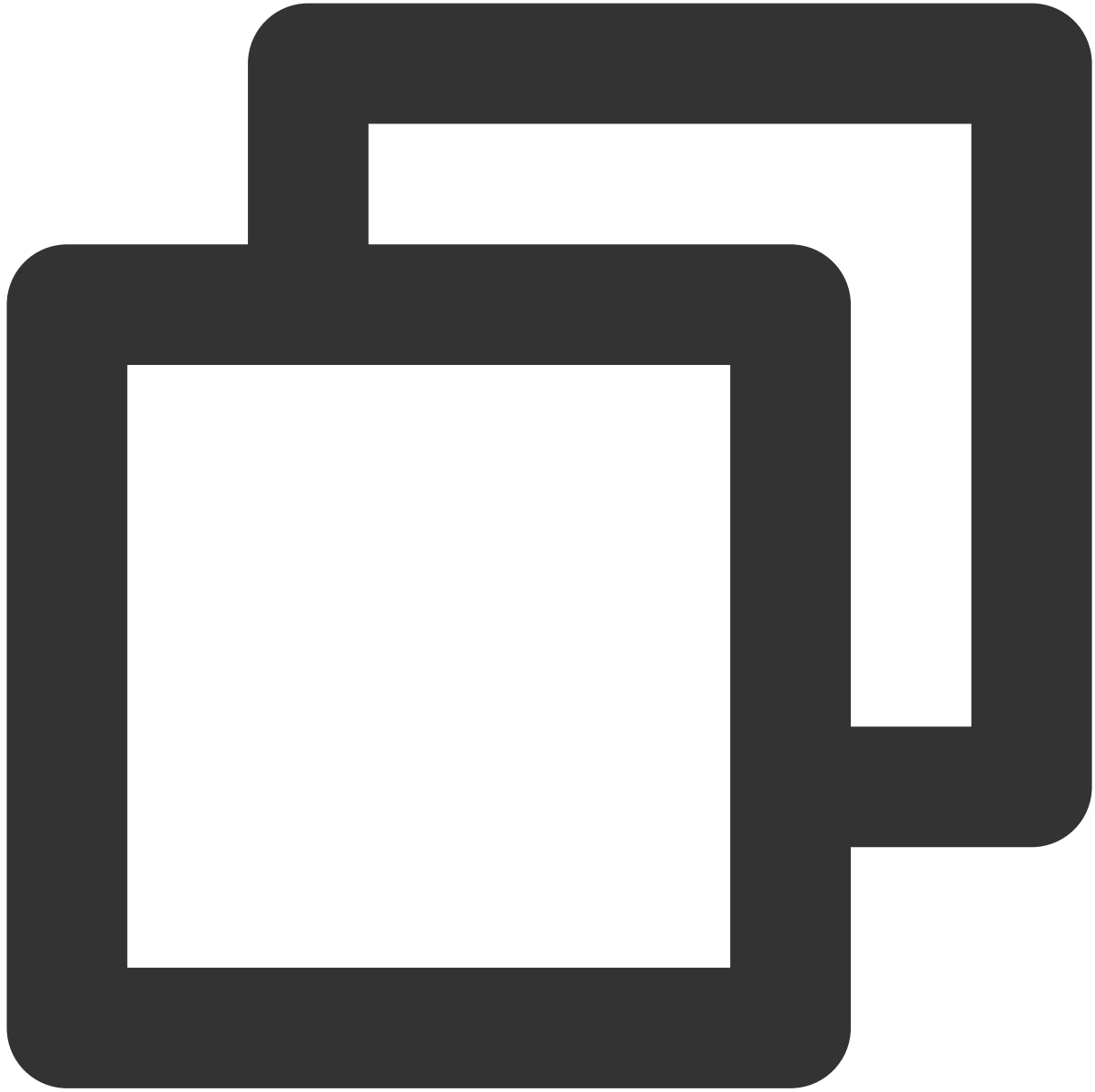


```
const roomEngine = new TUIRoomEngine();  
await roomEngine.stopCameraDeviceTest();
```

**Returns** : *Promise<void>*

**on**

Listen to roomEngine events.



```
const roomEngine = new TUIRoomEngine();  
roomEngine.on(event, func);
```

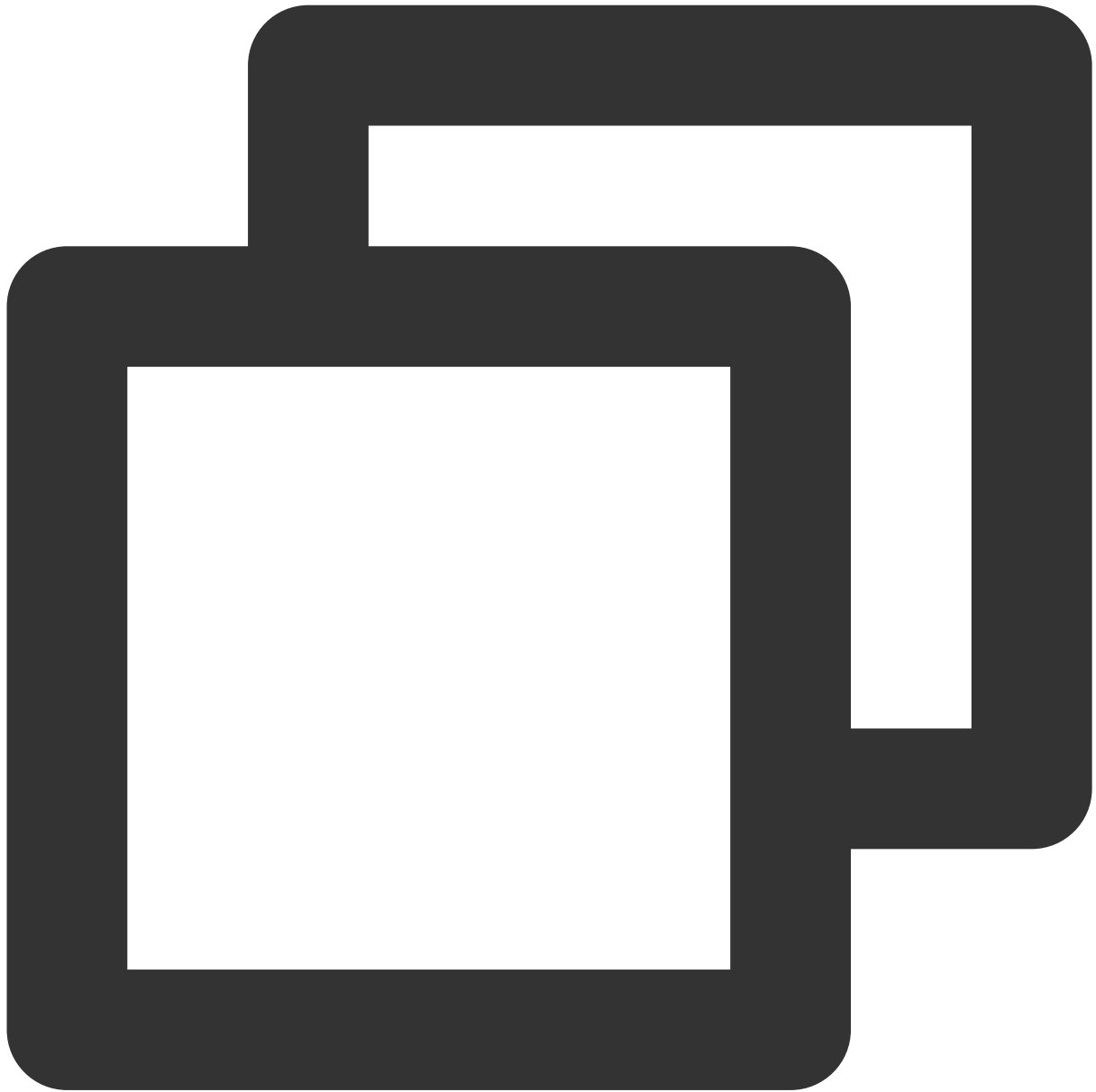
Parameter:

Parameter	Type	Description	Default Value	Meaning
event	<a href="#">TUIRoomEvents</a>	Required	-	TUIRoomEngine event list
func	Function	Required	-	Event callback function

**Returns :** *void*

## **off**

Cancel listening to roomEngine events



```
const roomEngine = new TUIRoomEngine();  
roomEngine.off(event, func);
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
-----------	------	-------------	---------------	---------

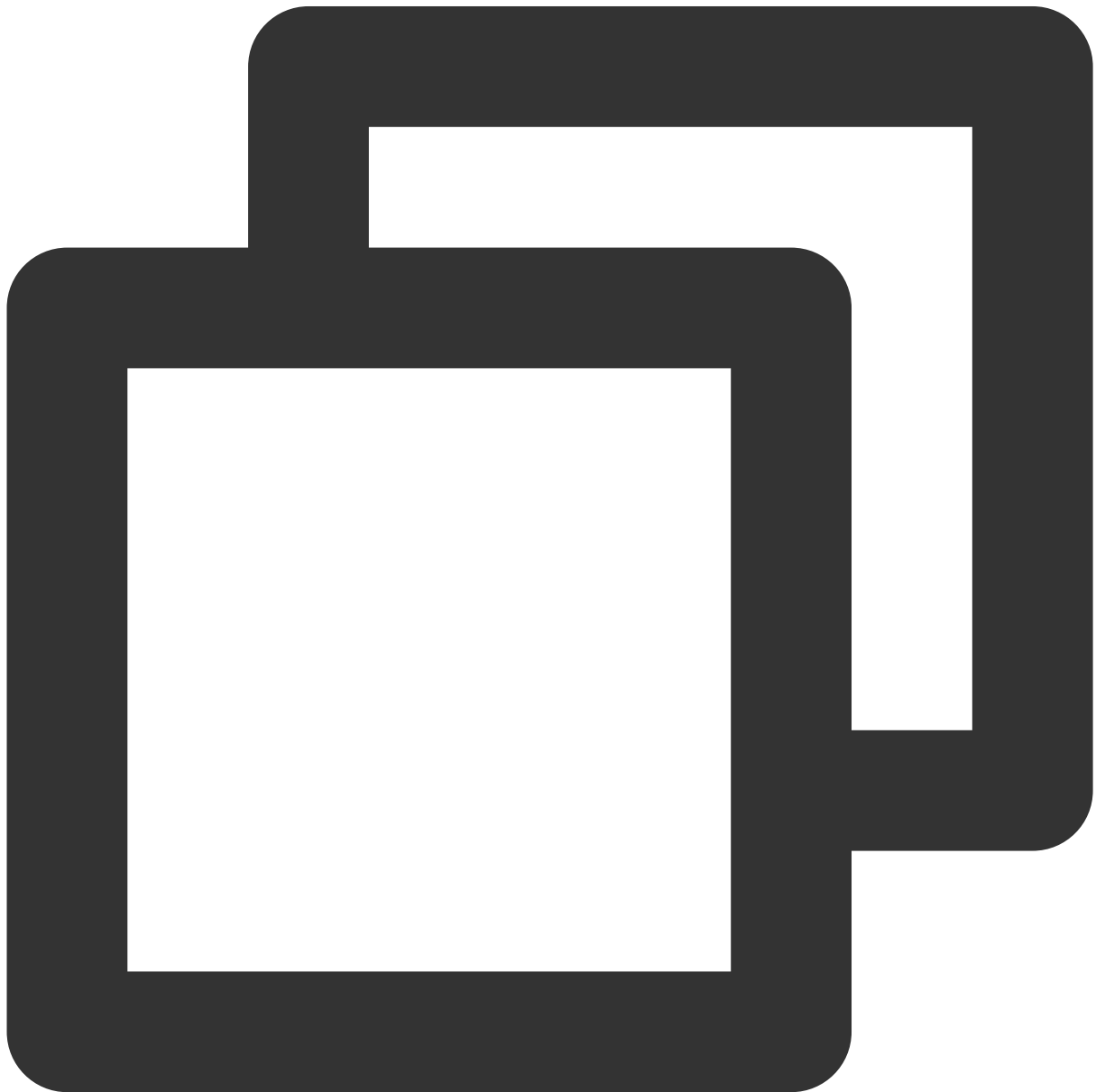


event	<a href="#">TUIRoomEvents</a>	Required	-	TUIRoomEngine event list
func	Function	Required	-	Event callback function

**Returns :** *void*

## getTRTCCloud

Get trtcCloud instance, for web-side trtcCloud capabilities, please check: [TRTCCloud API documentation](#).



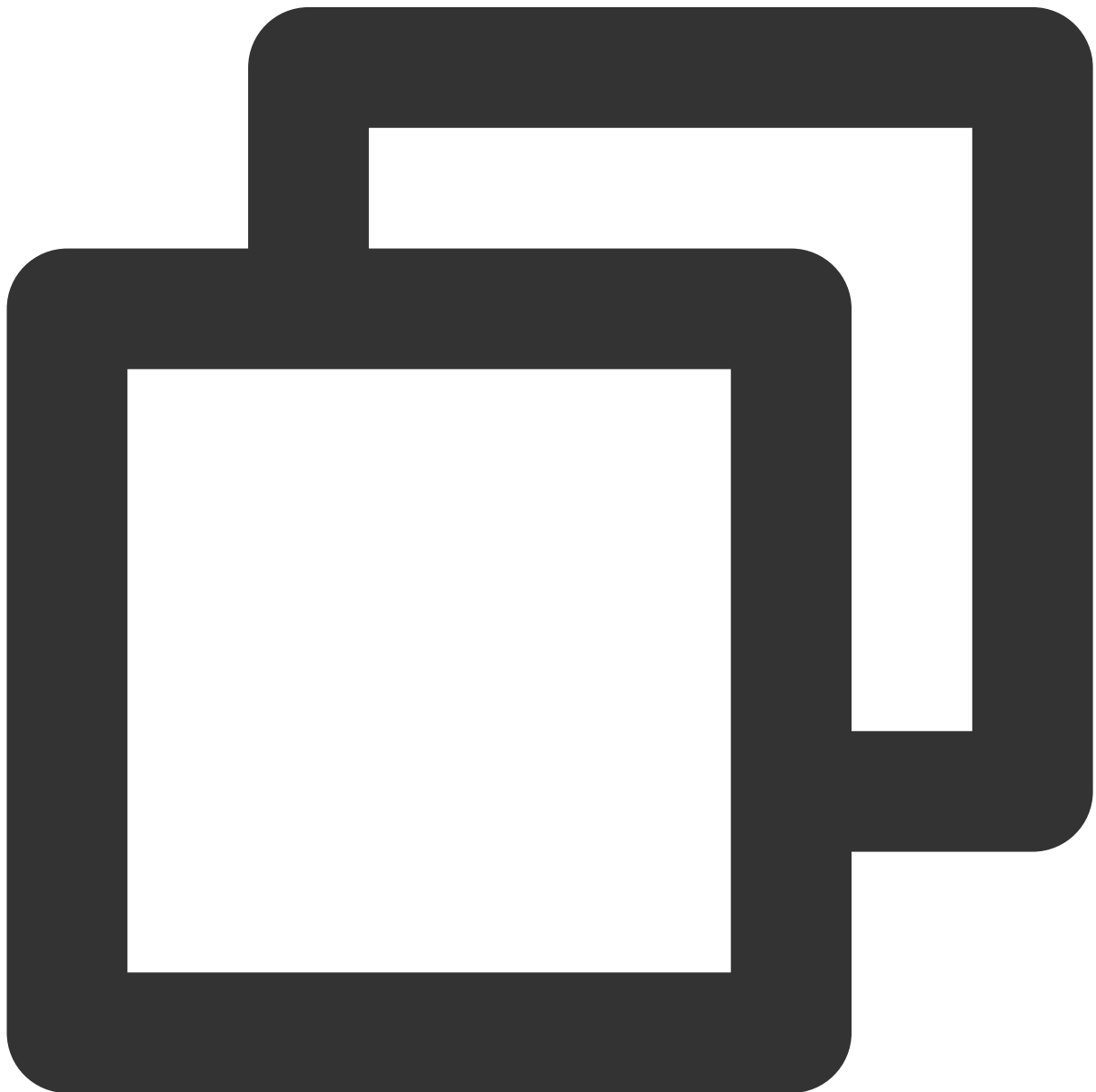
```
const roomEngine = new TUIRoomEngine();
```

```
const trtcCloud = roomEngine.getTRTCCloud();
```

**Returns :** [TRTCCloud](#)

## getTIM

Get tim instance, for web-side tim capabilities, please check: [IM API documentation](#)



```
const roomEngine = new TUIRoomEngine();  
const trtcCloud = roomEngine.getTIM();
```

**Returns :** *TIM*



# TUIRoomEvent

Last updated : 2023-11-22 11:26:33

## TUIRoomEvent API Introduction

TUIRoomEvent API is the Event Interface for Multiplayer Component.

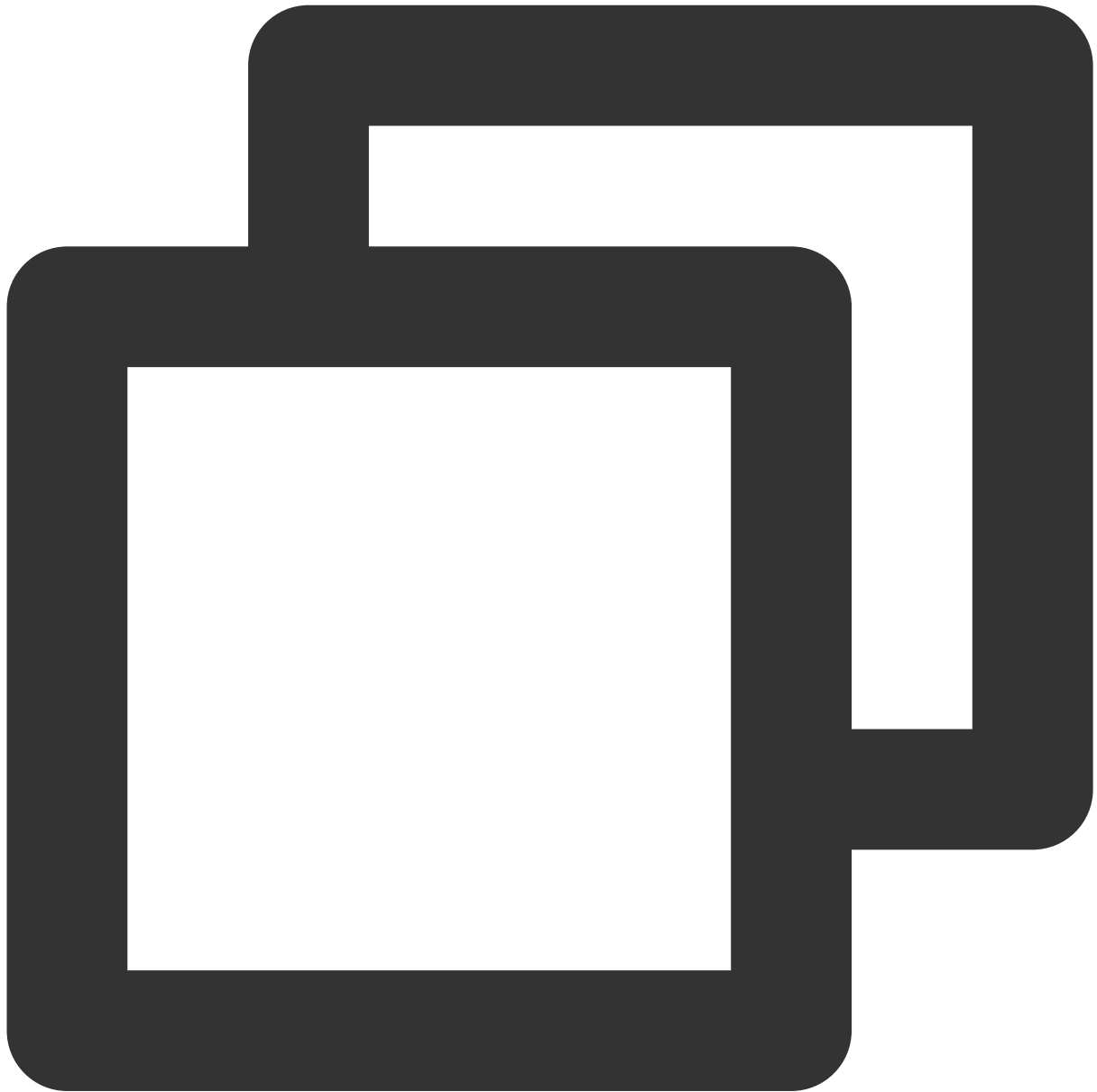
### Event list

EVENT	Description
<a href="#">TUIRoomEvents.onError</a>	Error Event
<a href="#">TUIRoomEvents.onKickedOutOfRoom</a>	Kicked out of room event
<a href="#">TUIRoomEvents.onKickedOffLine</a>	Current user Kicked off line event
<a href="#">TUIRoomEvents.onUserSigExpired</a>	UserSig Expiration Event
<a href="#">TUIRoomEvents.onRoomDismissed</a>	Host Destroy Room Event
<a href="#">TUIRoomEvents.onRoomNameChanged</a>	Room Name Change Event
<a href="#">TUIRoomEvents.onRoomSpeechModeChanged</a>	Room Speech Mode Change Event
<a href="#">TUIRoomEvents.onAllUserCameraDisableChanged</a>	All members Camera Usage Permission Change Event
<a href="#">TUIRoomEvents.onAllUserMicrophoneDisableChanged</a>	All members Mic Usage Permission Change Event
<a href="#">TUIRoomEvents.onSendMessageForAllUserDisableChanged</a>	All members Send Message Status Change Event
<a href="#">TUIRoomEvents.onRoomMaxSeatCountChanged</a>	Room Maximum Seat Count Change Event
<a href="#">TUIRoomEvents.onRemoteUserEnterRoom</a>	Remote User Entered Room Event
<a href="#">TUIRoomEvents.onRemoteUserLeaveRoom</a>	Remote User Leave Room Event
<a href="#">TUIRoomEvents.onUserRoleChanged</a>	User Role Change Event
<a href="#">TUIRoomEvents.onUserVideoStateChanged</a>	User Video Status Change Event

<a href="#">TUIRoomEvents.onUserAudioStateChanged</a>	User Audio Status Change Event
<a href="#">TUIRoomEvents.onSendMessageForUserDisableChanged</a>	User Send Message Status Event
<a href="#">TUIRoomEvents.onUserVoiceVolumeChanged</a>	User Volume Change Event
<a href="#">TUIRoomEvents.onUserNetworkQualityChanged</a>	User Network Quality Change Event
<a href="#">TUIRoomEvents.onSeatListChanged</a>	Seat List Change Event
<a href="#">TUIRoomEvents.onKickedOffSeat</a>	User Kicked off Seat Event
<a href="#">TUIRoomEvents.onRequestReceived</a>	Request Received Event
<a href="#">TUIRoomEvents.onRequestCancelled</a>	Request Cancelled Event
<a href="#">TUIRoomEvents.onReceiveTextMessage</a>	Receive Text Message Event
<a href="#">TUIRoomEvents.onReceiveCustomMessage</a>	Receive Custom Message Event
<a href="#">TUIRoomEvents.onDeviceChange</a>	Device Change Event
<a href="#">TUIRoomEvents.onUserScreenCaptureStopped</a>	Screen Sharing Stopped Event When the user uses the built-in browser stop sharing button to end screen sharing, the user will receive the 'onUserScreenCaptureStopped' event to modify the screen sharing status.

## onError

Error Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onError, (error) => {
  console.log('TUIRoomError error', error);
})
```

The parameters are shown in the table below:

Parameter	Type	Meaning
code	number	Error Code

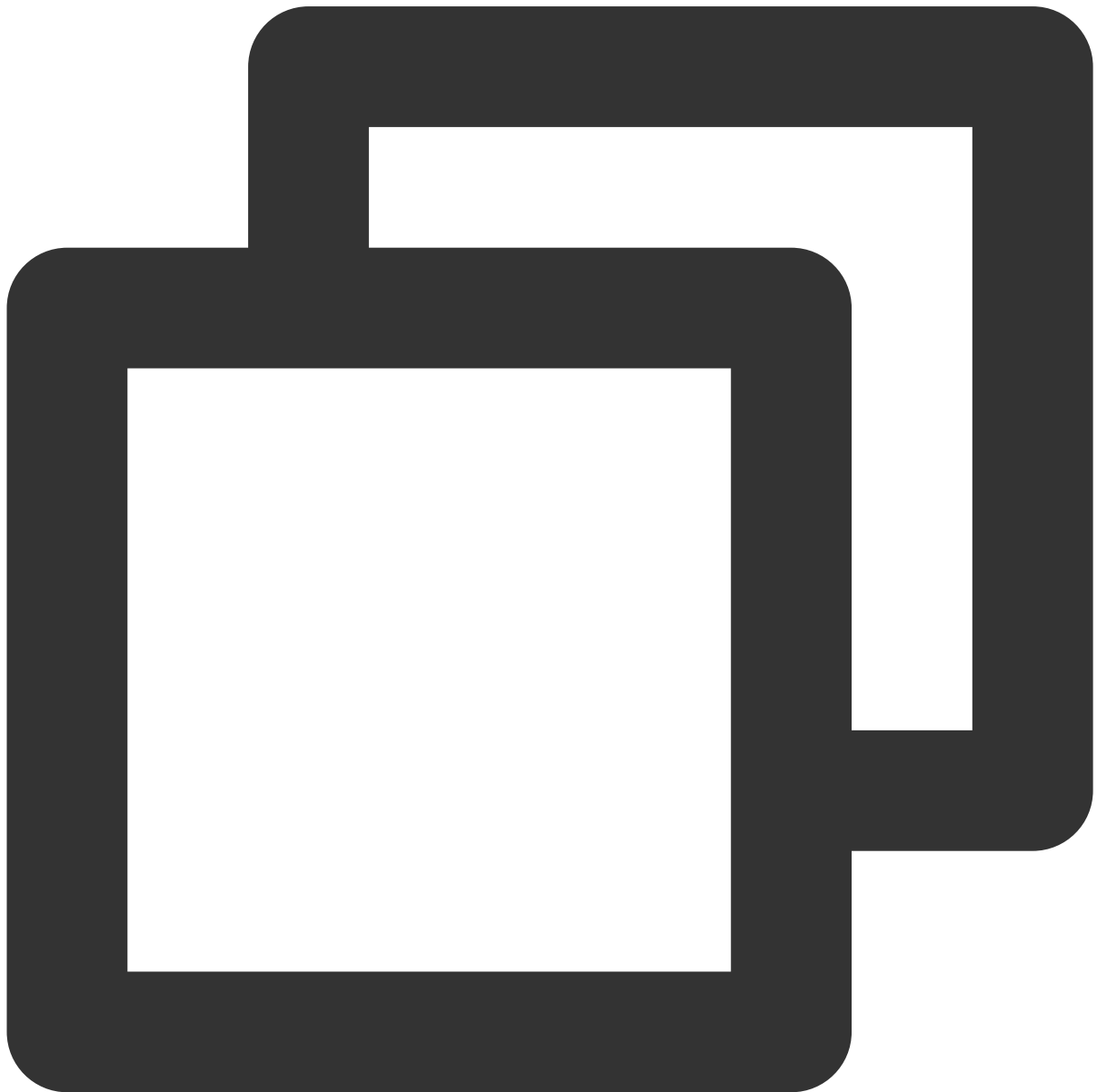
message

string

Error Information

## onKickedOutOfRoom

Kicked out of room event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOutOfRoom, ({ roomId, message }) => {
  console.log('roomEngine.onKickedOutOfRoom', roomId, message);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
message	string	Kicked out of room information

## onKickedOffLine

Current user Kicked off line event





```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffLine, ({ message }) => {
  console.log('roomEngine.onKickedOffLine', message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

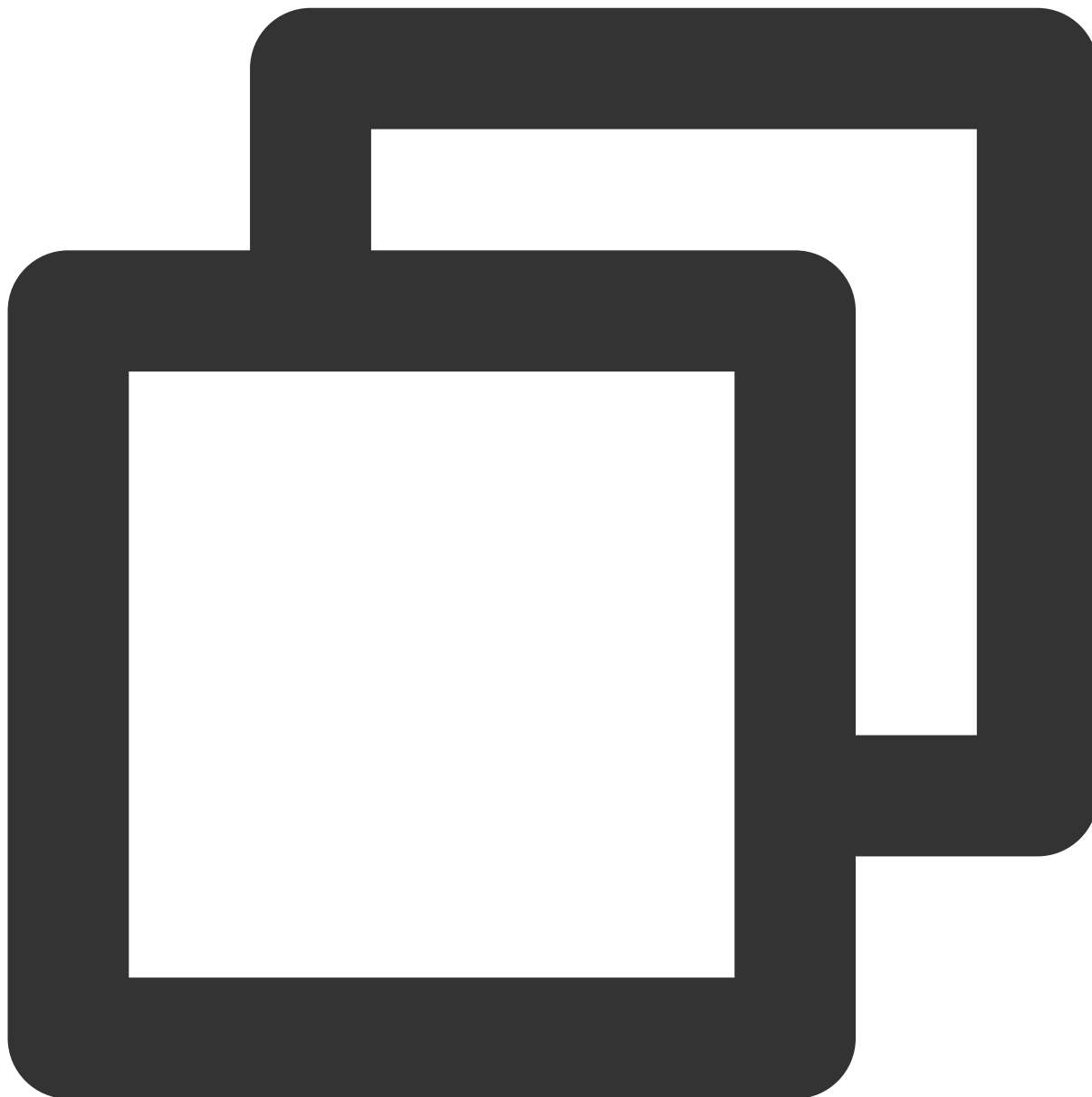
message

string

User logged in on other end information

## onUserSigExpired

UserSig Expiration Event

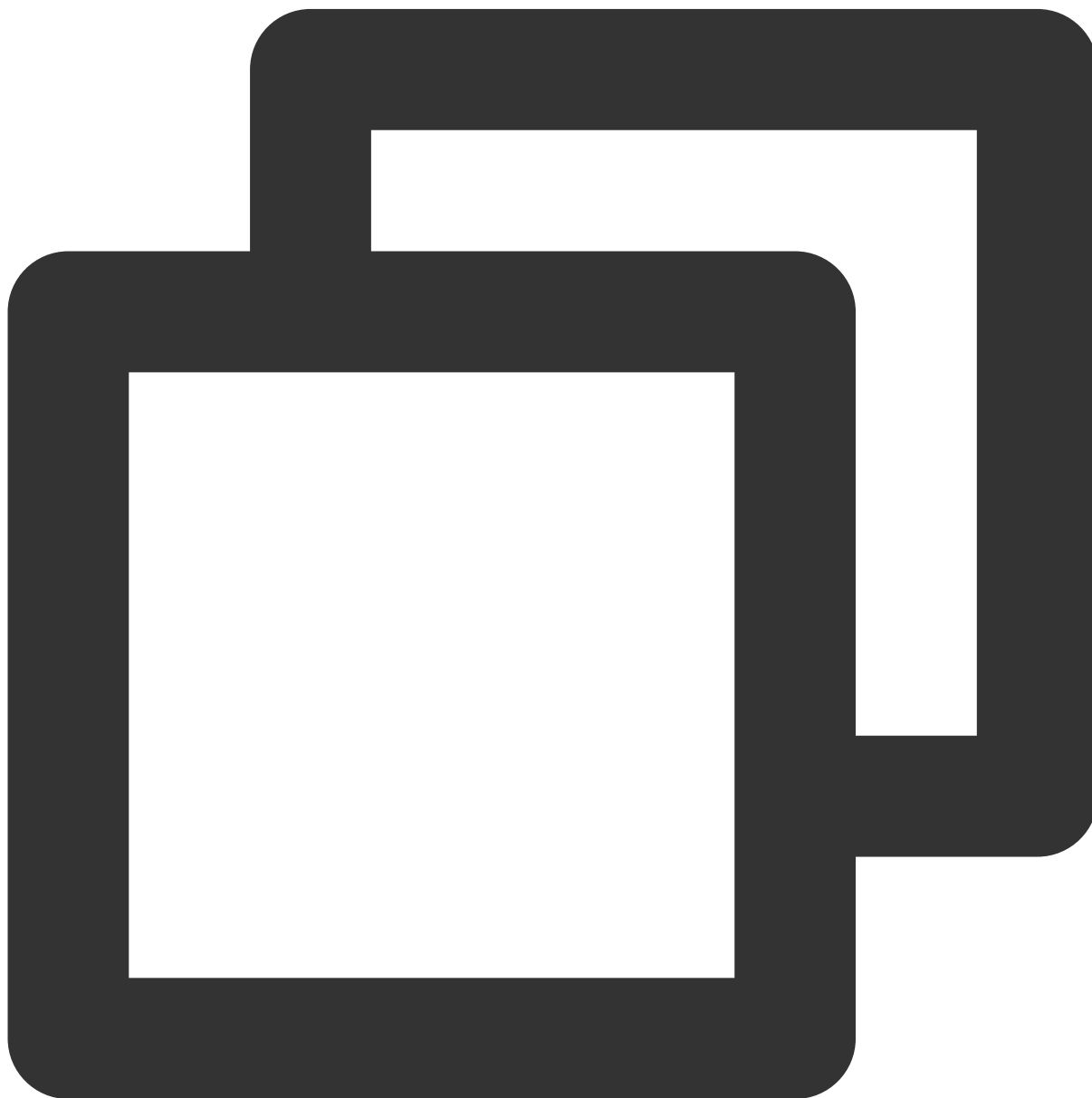


```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserSigExpired, () => {
  console.log('roomEngine.onUserSigExpired');
});
```

```
});
```

## onRoomDismissed

Host Destroy Room Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomDismissed, ({ roomId }) => {
  console.log('roomEngine.onRoomDismissed', roomId);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

## onRoomNameChanged

Room ID Modification Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomNameChanged, ({ roomId, roomName }) => {
  console.log('roomEngine.onRoomNameChanged', roomId, roomName);
});
```

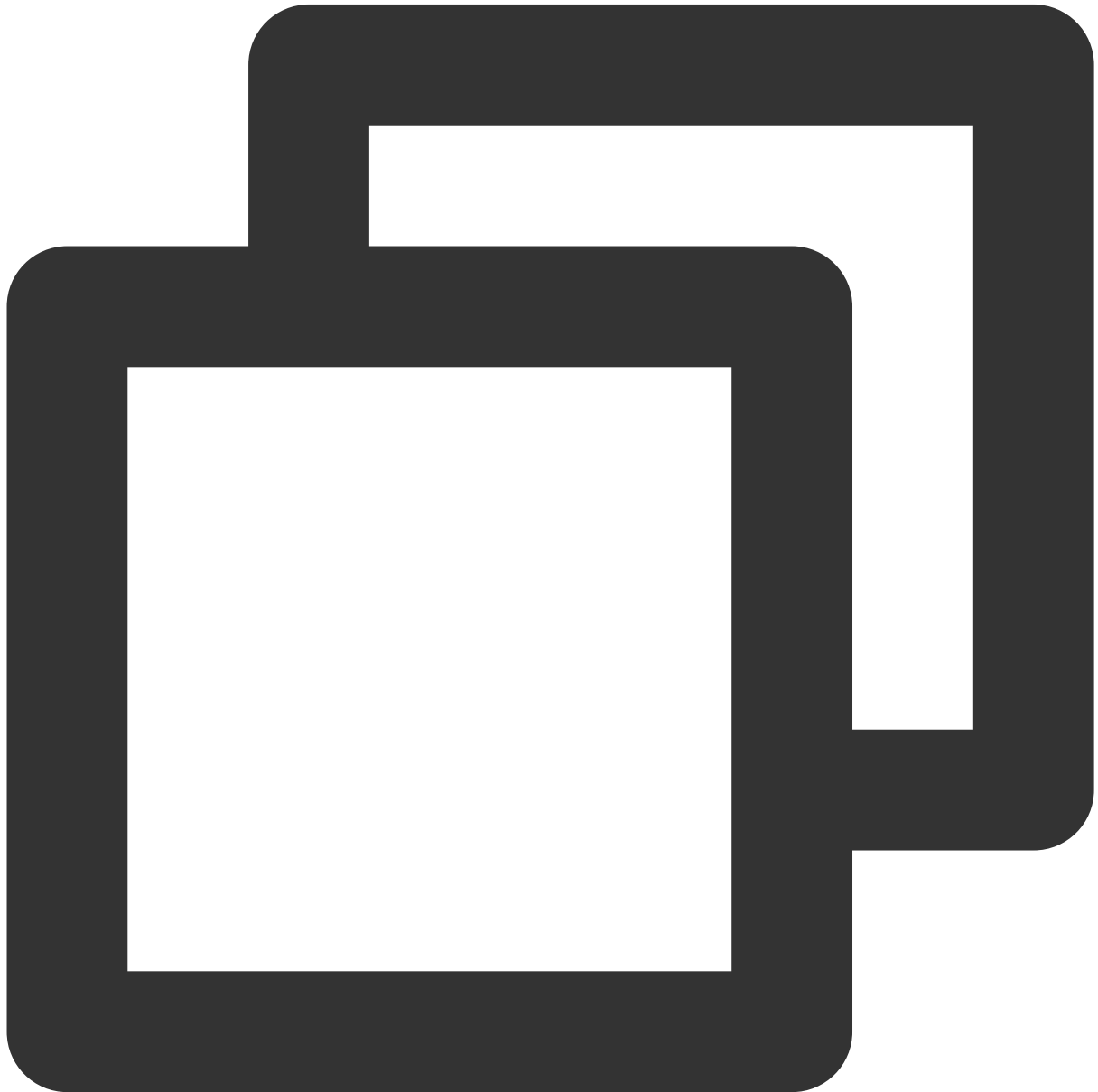
The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID

roomName	string	Room Name
----------	--------	-----------

## onRoomSpeechModeChanged

Room Name Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSpeechModeChanged, ({ roomId, speechMode }) => {
  console.log('roomEngine.onRoomSpeechModeChanged', roomId, speechMode);
});
```

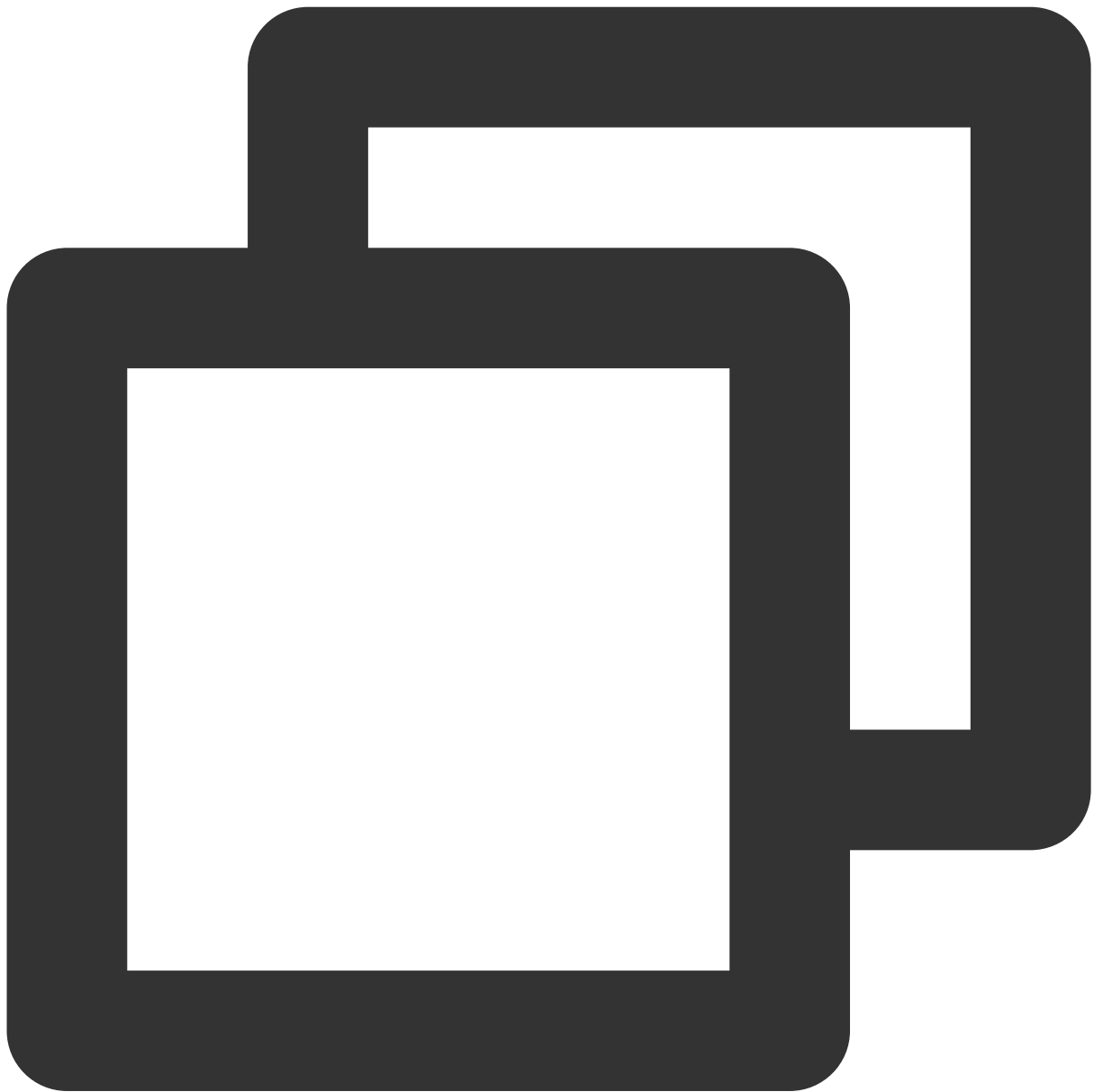
```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
speechMode	<a href="#">TUISpeechMode</a>	Speech Mode

## onAllUserCameraDisableChanged

All members Camera Usage Permission Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserCameraDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserCameraDisableChanged', isDisable);
});
```

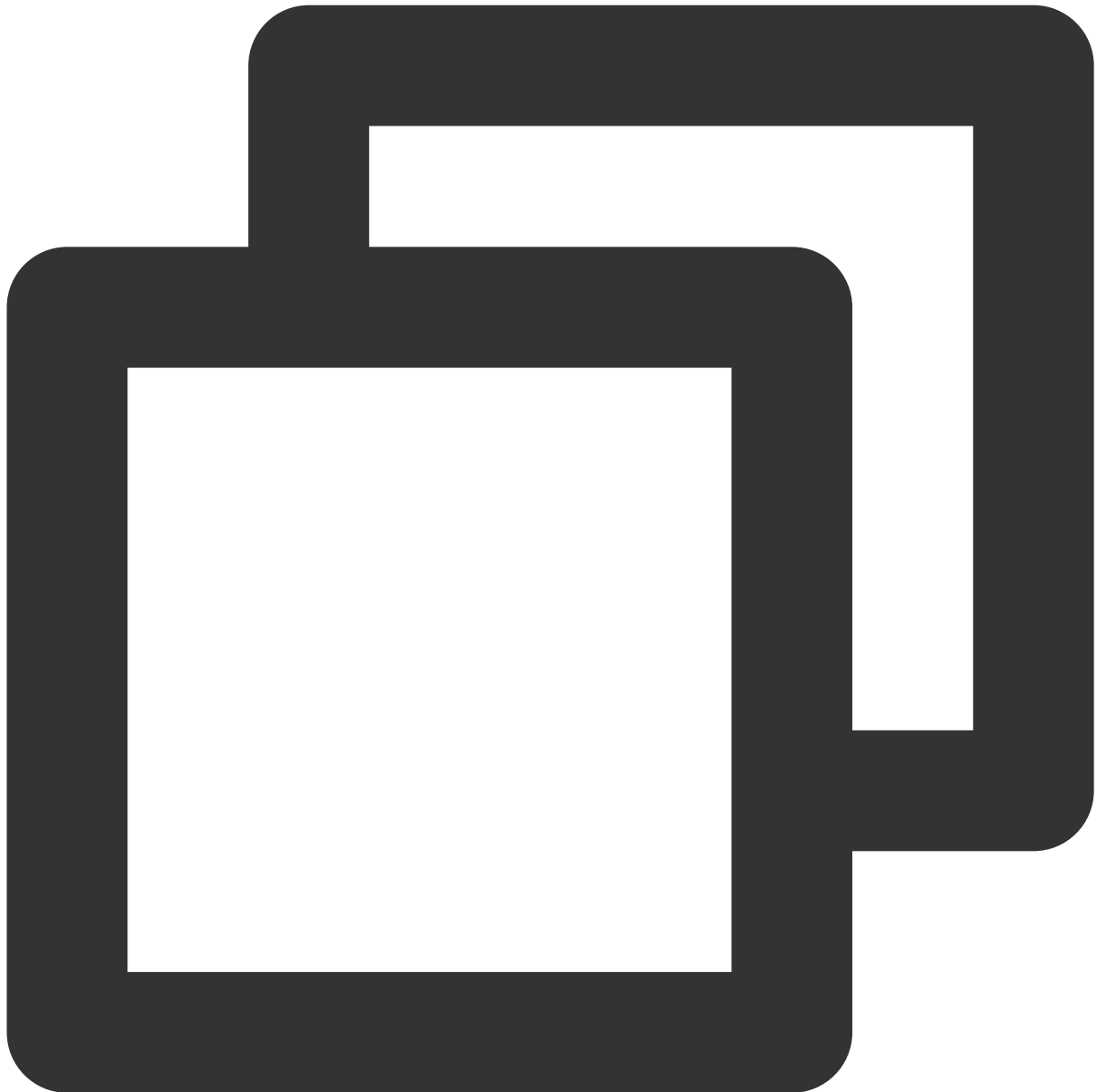
The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Camera Usage



## onAllUserMicrophoneDisableChanged

All members Mic Usage Permission Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserMicrophoneDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserMicrophoneDisableChanged', isDisable);
});
```

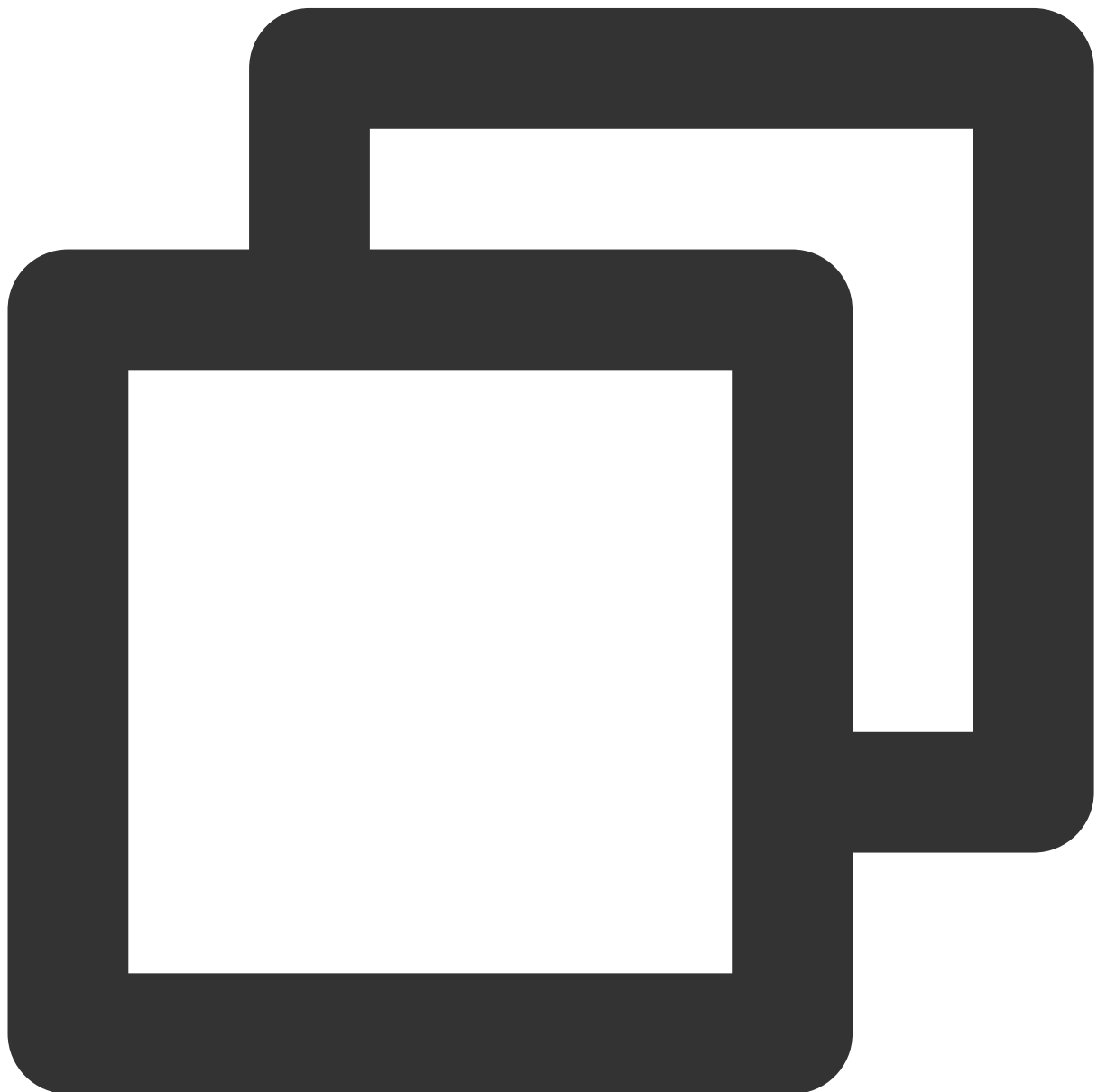
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
isDisable	boolean	Allow Mic Usage

## onSendMessageForAllUserDisableChanged

All members Send Message Permission Change Event



```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable })  
    console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);  
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Sending Text Message

## onRoomMaxSeatCountChanged

Room Maximum Seat Count Change Event



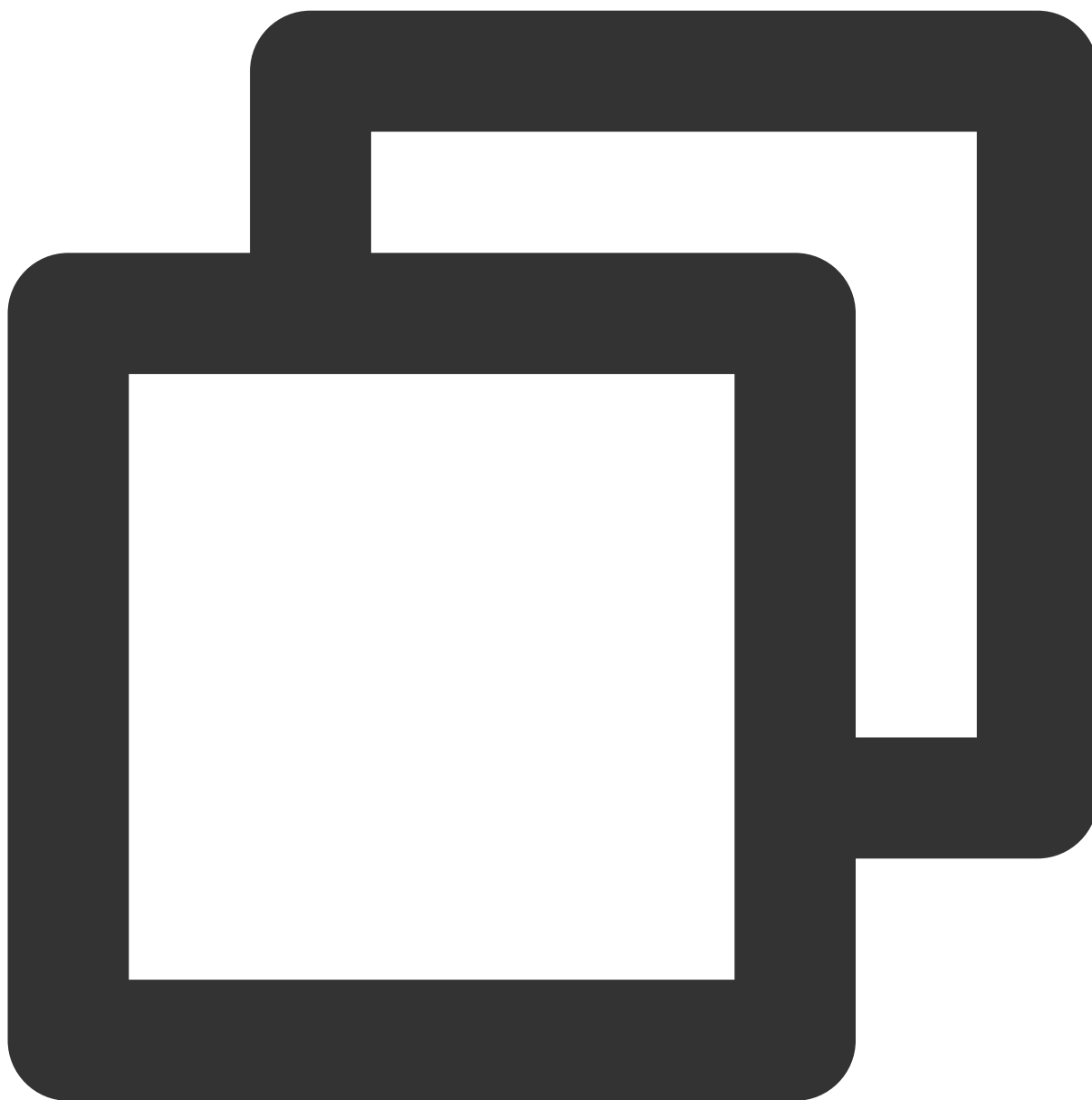
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomMaxSeatCountChanged, ({ maxSeatNumber }) => {
  console.log('roomEngine.onRoomMaxSeatCountChanged', maxSeatNumber);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
maxSeatNumber	number	Maximum Seat Count

## onRemoteUserEnterRoom

Remote User Entered Room Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserEnterRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserEnterRoom', roomId, userInfo);
});
```

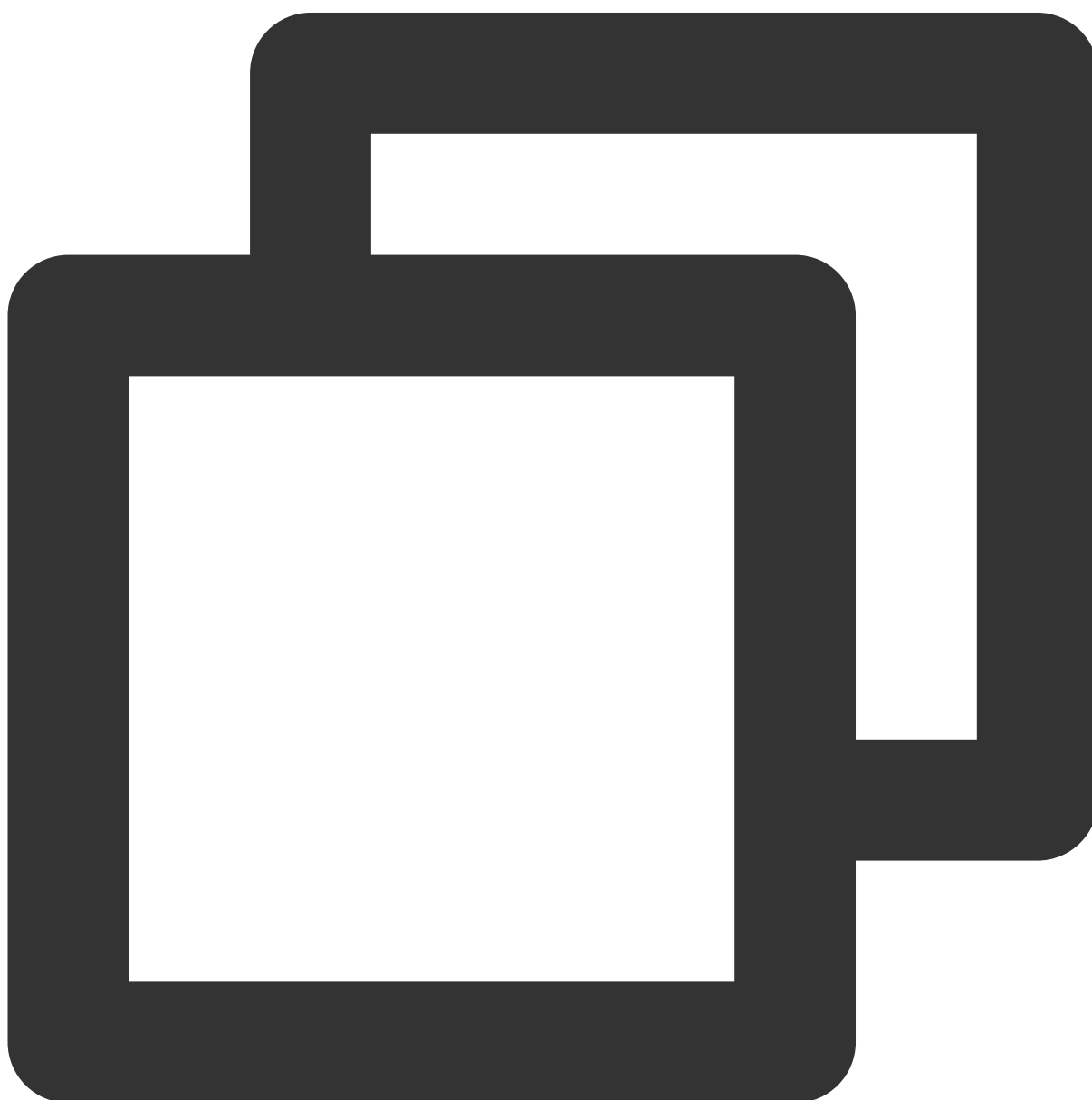
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
roomId	string	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User Information

## onRemoteUserLeaveRoom

Remote User Leave Room Event



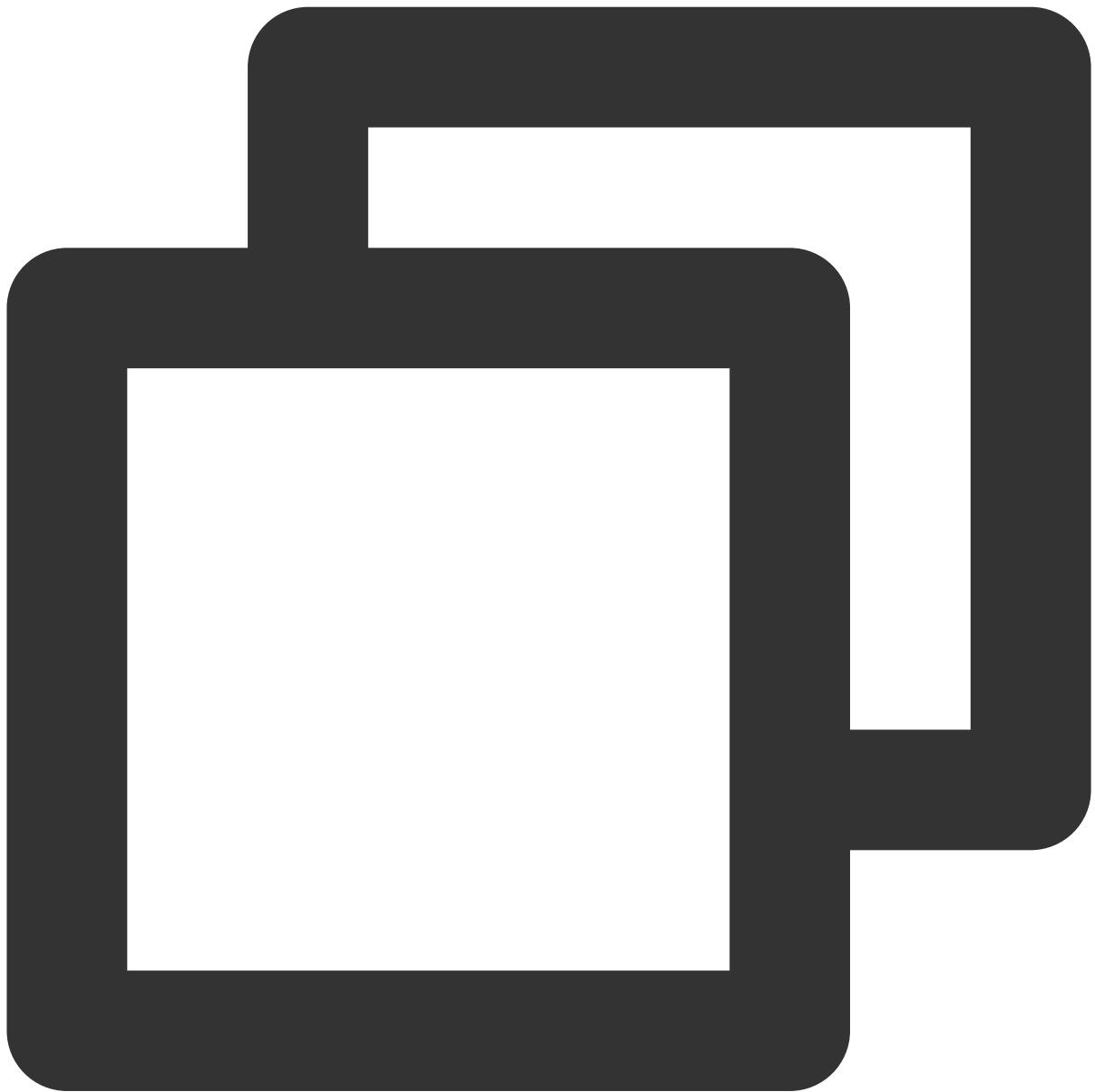
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserLeaveRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserLeaveRoom', roomId, userInfo);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User Information

## onUserRoleChanged

User Role Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserRoleChanged, ({ userId, userRole }) => {
  console.log('roomEngine.onUserRoleChanged', userId, userRole);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id



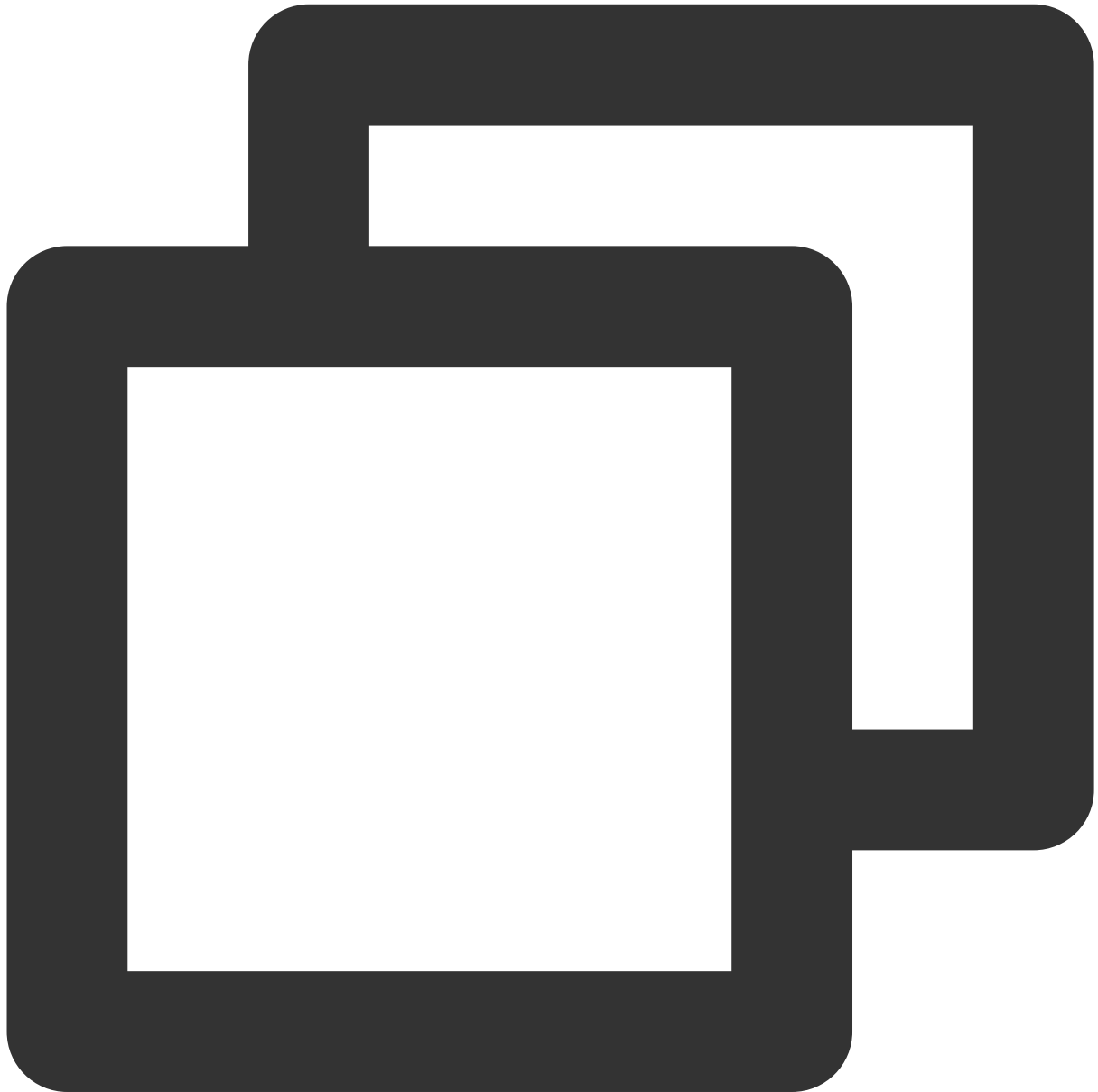
userRole

TUIRole

User Changed Role

## onUserVideoStateChanged

User Video Status Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVideoStateChanged, ({ userId, streamType, hasVideo, ...info }) => {
  console.log('roomEngine.onUserVideoStateChanged', userId, streamType, hasVideo, ...info);
});
```

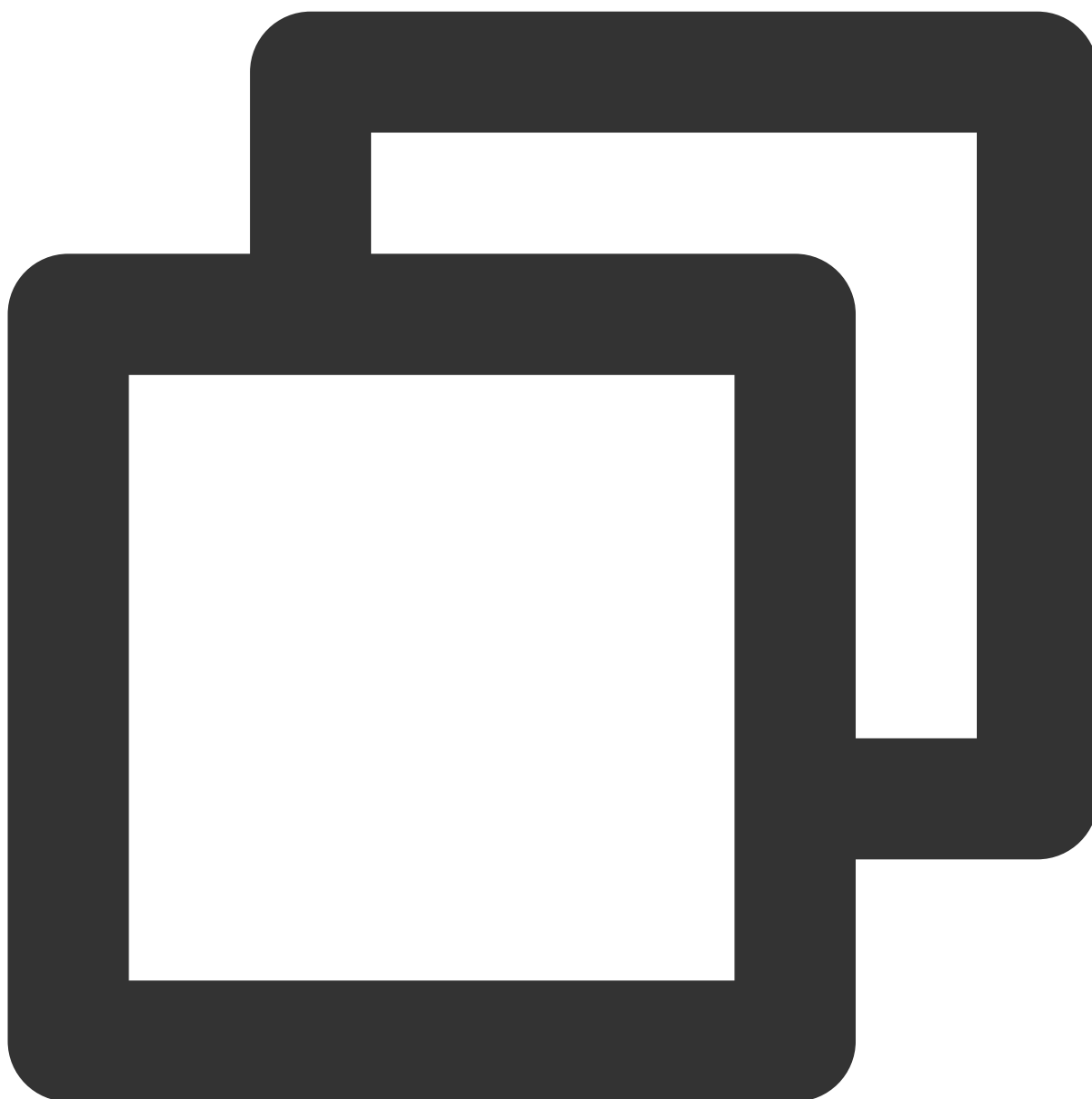
```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id
streamType	<a href="#">TUIVideoStreamType</a>	User Stream Type
hasVideo	boolean	Has Video streams
reason	<a href="#">TUIChangeReason</a>	Change Reason, Self-operation/Host-operation

## onUserAudioStateChanged

User Audio Status Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserAudioStateChanged, ({ userId, hasAudio, reason })
  console.log('roomEngine.onUserAudioStateChanged', userId, hasAudio, reason);
});
```

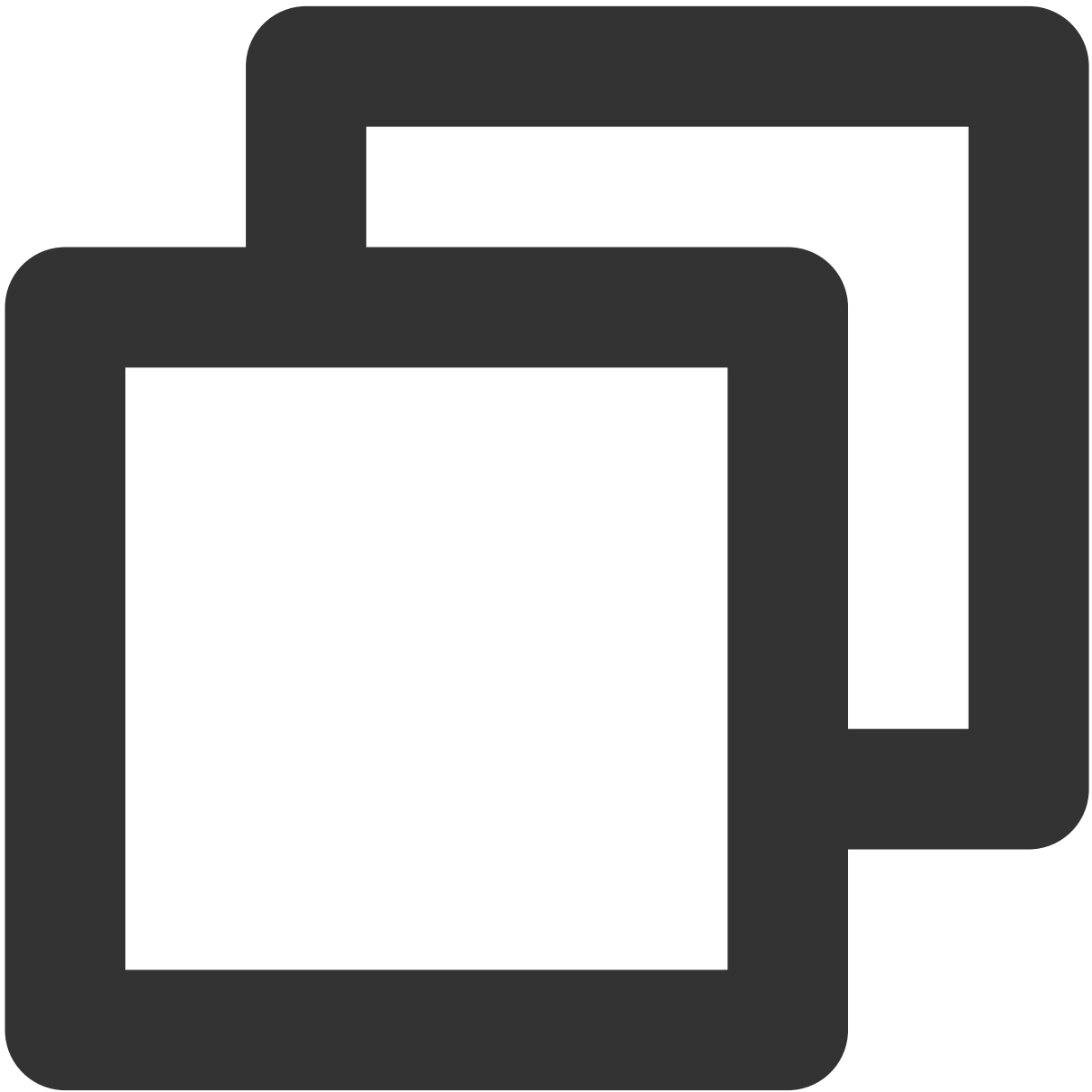
The parameters are shown in the table below:

Parameter	Type	Meaning
userId	string	User Id

hasVideo	boolean	Has Audio streams
reason	<a href="#">TUIChangeReason</a>	Change Reason, Self-operation/Host-operation

## onSendMessageForUserDisableChanged

Member Camera Usage Permission Change Event



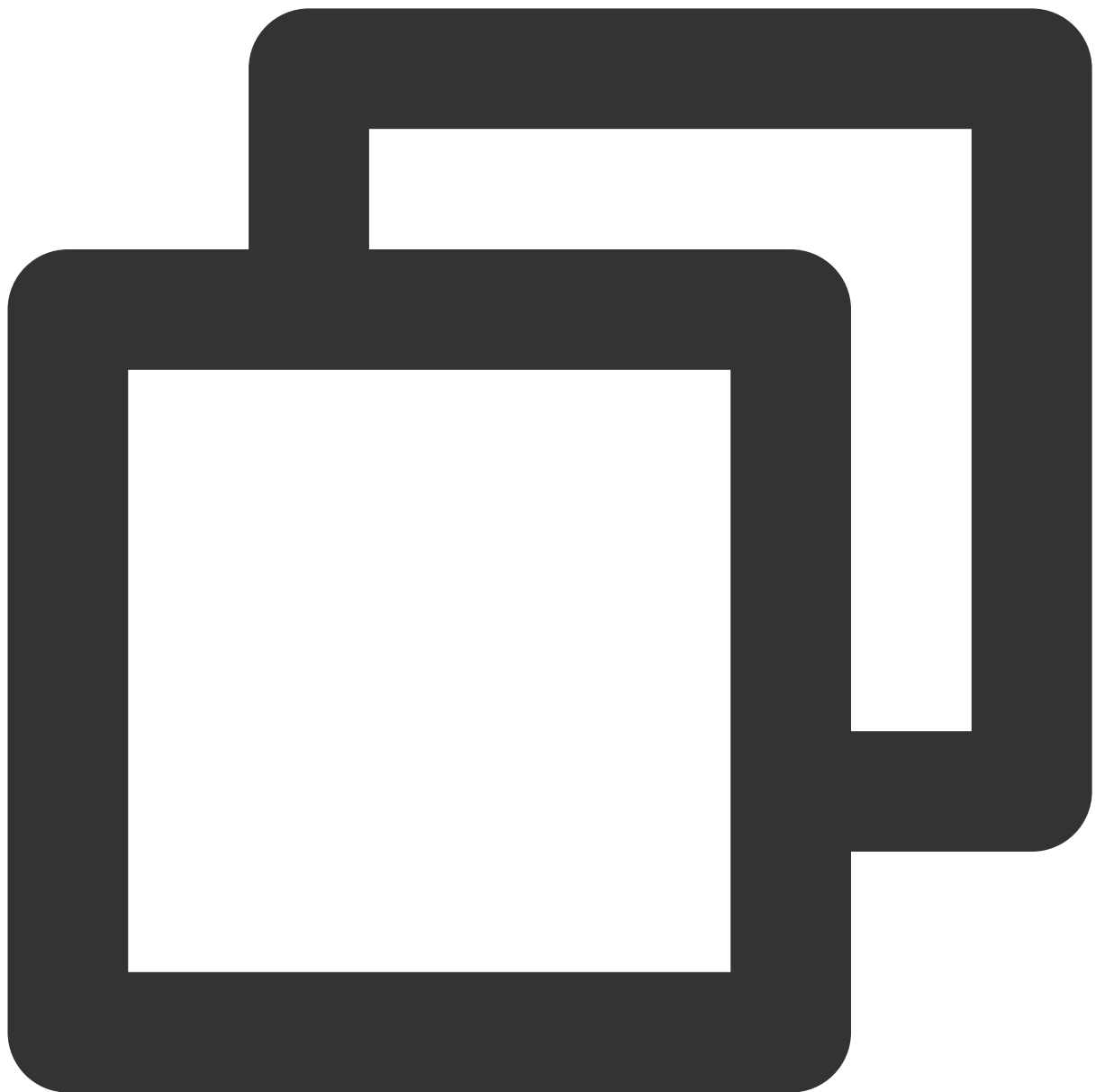
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable })
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
isDisable	boolean	Allow Sending Text Message

## onUserVoiceVolumeChanged

User Volume Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVoiceVolumeChanged, ({ userVolumeList }) => {
  userVolumeList.forEach(userVolume => {
    console.log('roomEngine.onUserVoiceVolumeChanged', userVolume.userId, userVolume);
  })
});
```

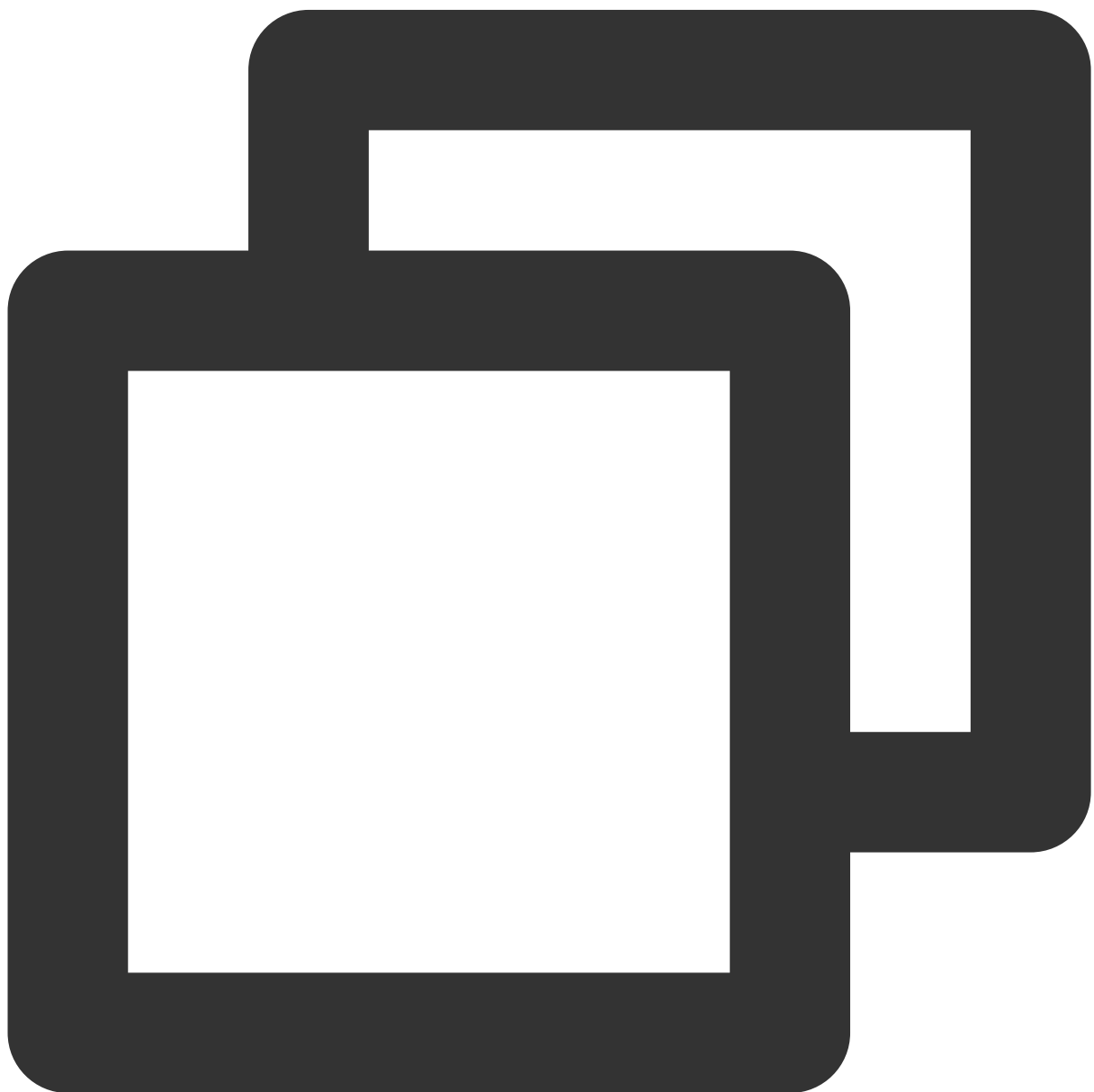
The parameters are shown in the table below:

Parameter	Type	Meaning

userVolumes	Array.<TRTCVolumeInfo>	Volume of all users in the room, including userId and volume information, volume range is 1-100
-------------	------------------------	---

## onUserNetworkQualityChanged

User Network Quality Change Event



```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onUserNetworkQualityChanged, ({ userNetworkList }) => {
  userNetworkList.forEach(userNetwork => {
    console.log('roomEngine.onUserNetworkQualityChanged', userNetwork.userId, userN
  })
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
networkMap	<a href="#">TUINetworkQuality</a>	Traverse Network Quality Level

## onSeatListChanged

Seat List Change Event





```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatListChanged, ({ seatList, seatedList, leftList })
  console.log('roomEngine.onSeatListChanged',seatList, seatedList, leftList);
});
```

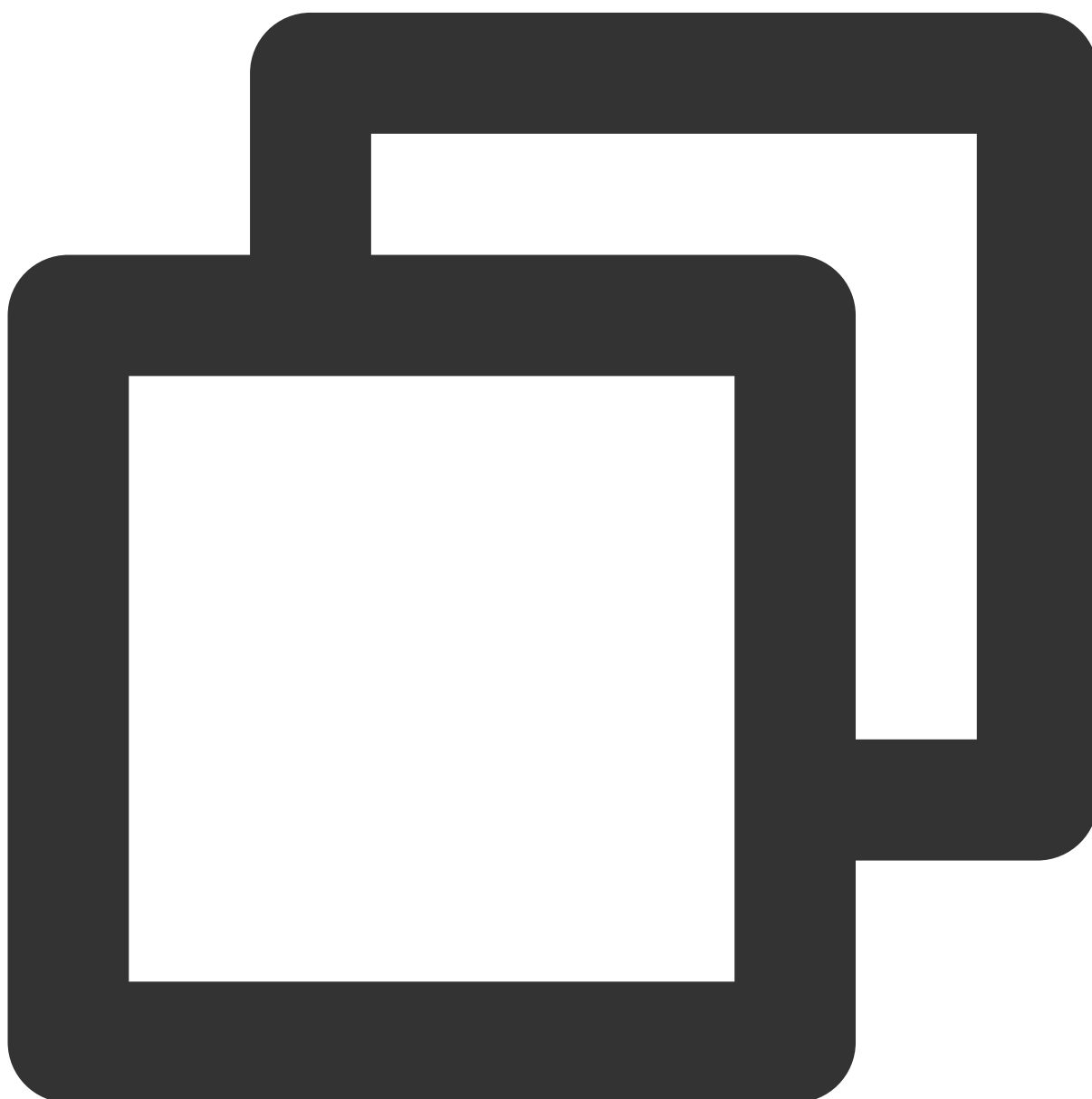
The parameters are shown in the table below:

Parameter	Type	Meaning
seatList	Array.< <a href="#">TUISeatInfo</a> >	Seat List

seatedList	Array.<TUISeatInfo>	New Seat Information
leftList	Array.<TUISeatInfo>	Left Seat Information

## onKickedOffSeat

Seat List Change Event



```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onKickedOffSeat, ({ userId }) => {  
    console.log('roomEngine.onKickedOffSeat', userId);  
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
userId	String	Kicked off Seat User Id

## onRequestReceived

Error Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestReceived, ({ request }) => {
  console.log('roomEngine.onRequestReceived', request);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
request	<a href="#">TUIRequest</a>	Request Received

# onRequestCancelled

Request Cancelled Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestCancelled, ({ requestId, userId }) => {
  console.log('roomEngine.onRequestCancelled', requestId, userId);
});
```

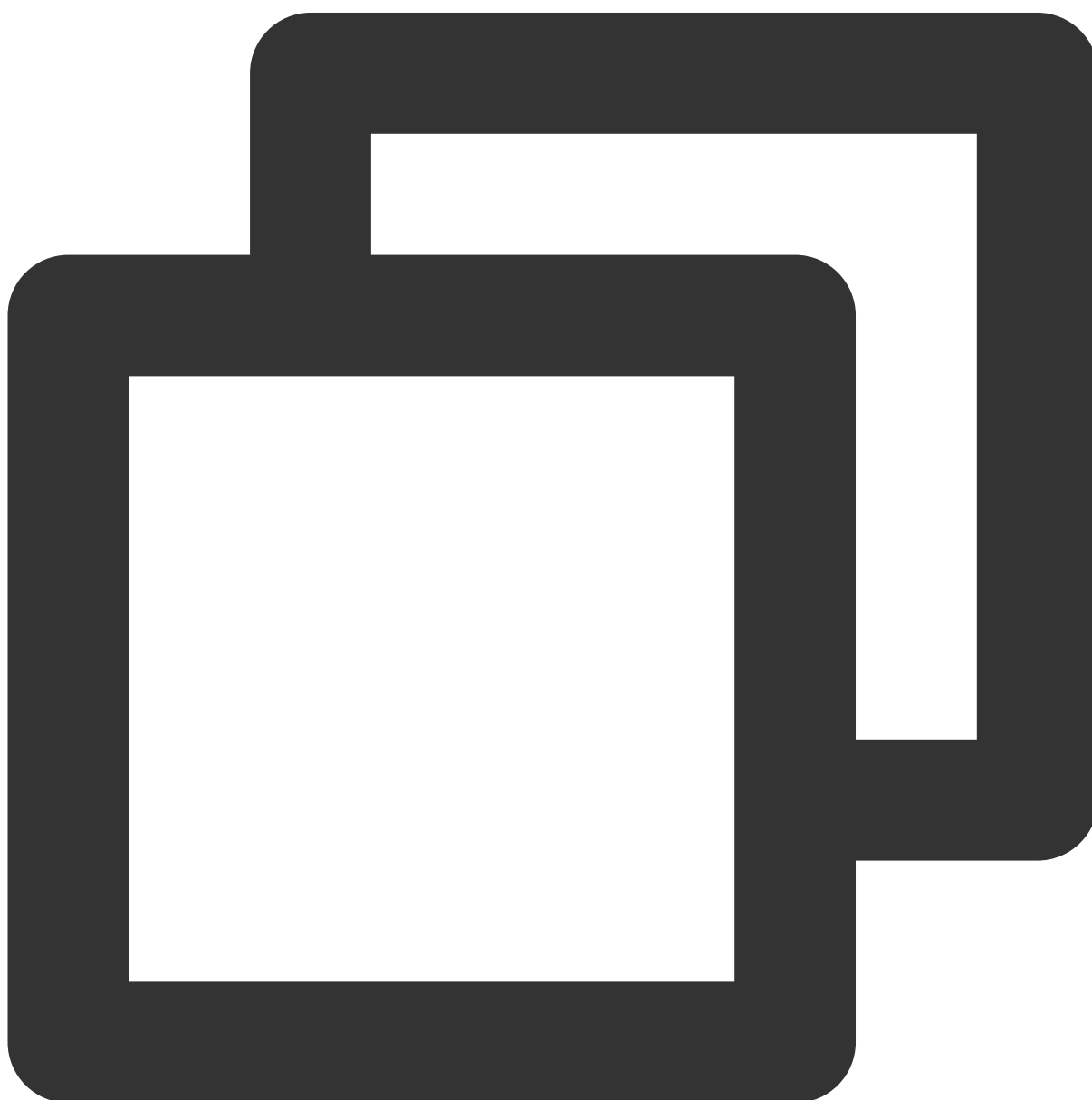
The parameters are shown in the table below:

--	--	--

Parameter	Type	Meaning
requestId	string	Request Id
userId	string	User Id of Cancelled Request

## onReceiveTextMessage

Receive Text Message Event



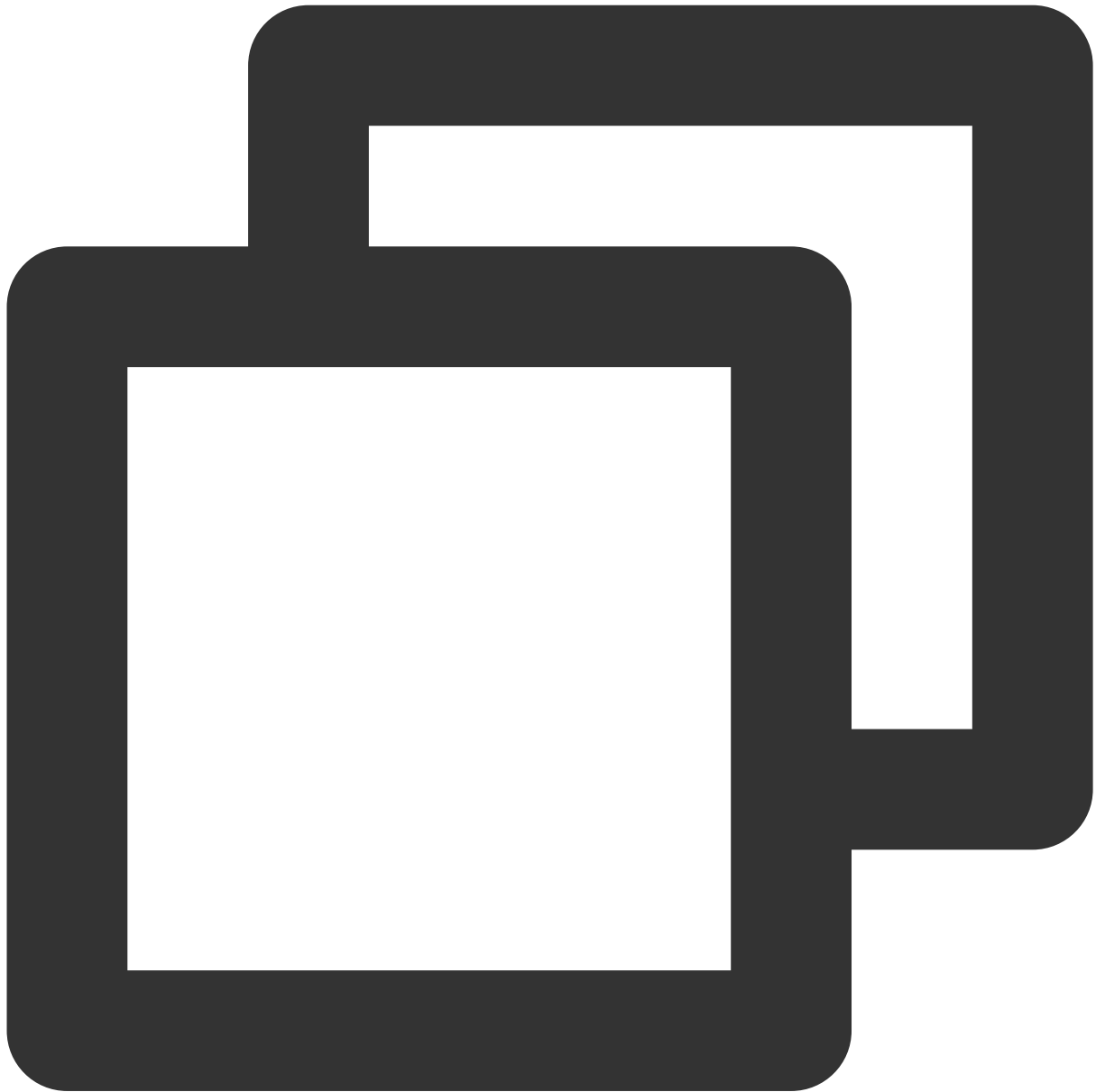
```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveTextMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveTextMessage', roomId, message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room Id
message	<a href="#">TUIMessage</a>	Receive Text Message

## onReceiveCustomMessage

Receive Custom Message Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveCustomMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveCustomMessage', roomId, message);
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
roomId	string	Room Id



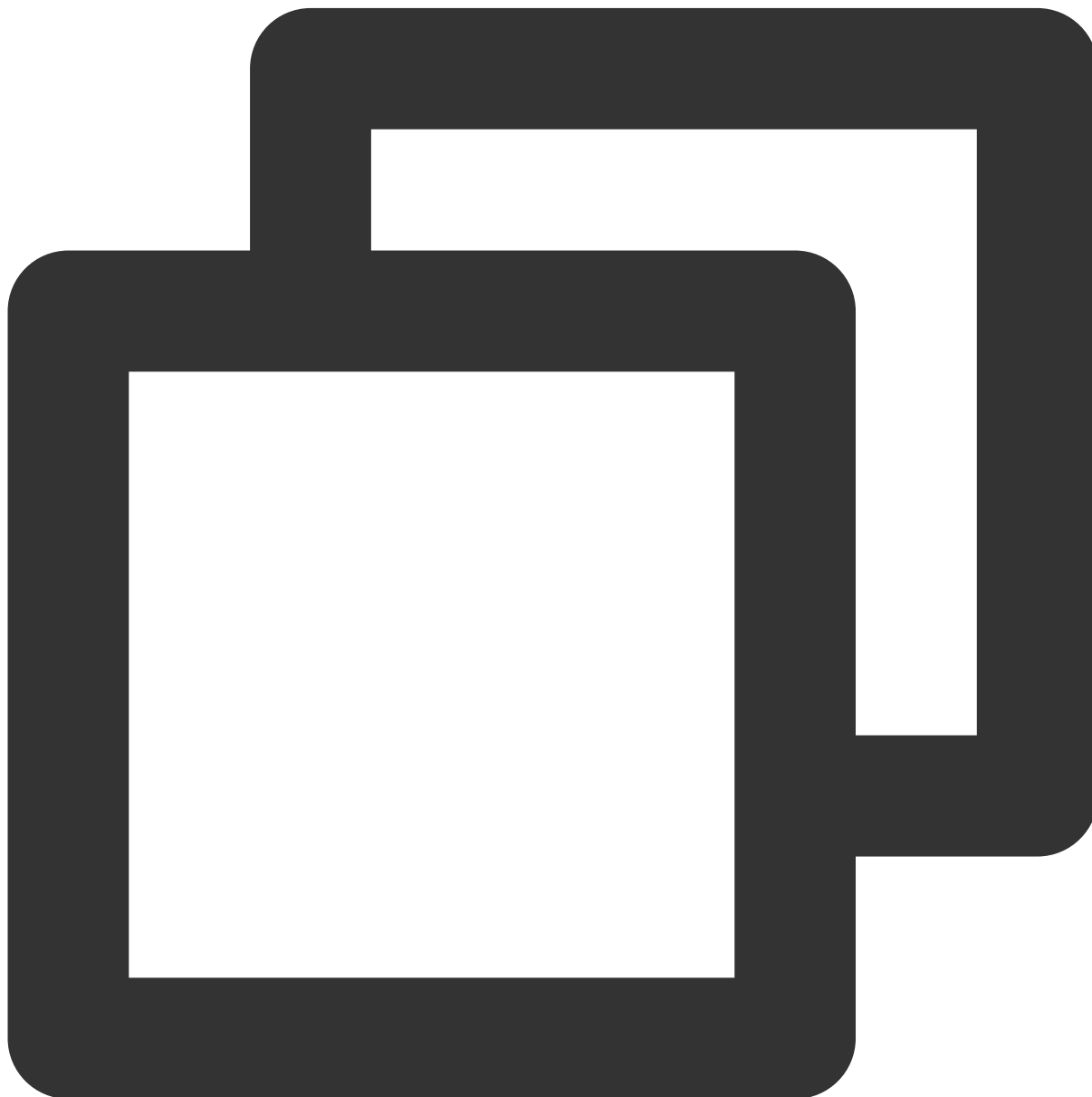
message

TUIMessage

Receive Text Message

## onDeviceChange

Device Change Event



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onDeviceChange, ({ deviceId, type, state }) => {
  console.log('roomEngine.onReceiveCustomMessage', deviceId, type, state);
});
```

```
});
```

The parameters are shown in the table below:

Parameter	Type	Meaning
deviceId	string	Device Id
type	<a href="#">TRTCDeviceType</a>	Device Type
state	<a href="#">TRTCDeviceState</a>	Device Change Status

## onUserScreenCaptureStopped

Screen Sharing Stopped Event, when the user uses the built-in browser stop sharing button to end screen sharing, the user will receive the ' `onUserScreenCaptureStopped` ' event to modify the screen sharing status.



```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserScreenCaptureStopped, () => {
  console.log('roomEngine.onReceiveCustomMessage', deviceId, type, state);
});
```

# TUIRoomEngine Defines

Last updated : 2023-11-16 14:33:35

Introduction to Key Type Definition of TUIRoomEngine Electron side.

## Enumeration Value

### TUIRole

User Role, TUIRoomEngine provides three user roles, which are Host, Administrator, and Regular User.

Field	Type	Description
kRoomOwner	number	Host Role
kAdministrator	number	Administrator Role
kGeneralUser	number	Regular User Role

### TUIVideoQuality

Video Resolution

Field	Type	Description
kVideoQuality_360p	number	Low Quality, Resolution is 640 * 360
kVideoQuality_540p	number	SD, Resolution is 960 * 540
kVideoQuality_720p	number	HD, Resolution is 1280 * 720
kVideoQuality_1080p	number	Ultra HD, Resolution is 1920 * 1080

### TUIAudioProfile

Audio Resolution

Field	Type	Description
kAudioProfileSpeech	number	Voice Mode
kAudioProfileDefault	number	Standard Mode (Default Mode)
kAudioProfileMusic	number	Music Mode

## TUIVideoStreamType

### Streams Type

Field	Type	Description
kCameraStream	number	Camera Streams
kScreenStream	number	Screen Sharing Streams
kCameraStreamLow	number	Low Resolution Camera Streams

## TUINetworkQuality

### Network Status

Field	Type	Description
kQualityUnknown	number	Network Condition Unknown
kQualityExcellent	number	Network Condition Excellent
kQualityGood	number	Network Condition Good
kQualityPoor	number	Network Condition Fair
kQualityBad	number	Network Condition Poor
kQualityVeryBad	number	Network Condition Very Poor
kQualityDown	number	Network Connection Disconnected

## TUIRoomType

### Room Type

Field	Type	Description
kGroup	number	Group Type Room, suitable for conference and educational scene, the microphone position in this room is unordered and has no quantity limit
kOpen	number	Open Type Room, suitable for live streaming scene, the microphone position in this room is ordered and has a quantity limit

## TUISpeechMode

### Speech Type

--

Field	Type	Description
kFreeToSpeak	number	Free Speech Mode
kApplyToSpeak	number	Hand-raising Speech Mode
kSpeakAfterTakingSeat	number	Speak After Sitting (Grab Microphone Position)

## TUICaptureSourceType

Screen Sharing Type

Field	Type	Description
kWindow	number	Sharing Target is a specific Windows or Mac window todo (only for electron)
kScreen	number	Sharing Target is the entire Windows desktop or Mac desktop

## TUIChangeReason

Change Reason (Audio and Video Status Change Operation Reason: Self-initiated modification or modified by room owner/administrator)

Field	Type	Description
kChangedBySelf	number	Self-operation
kChangedByAdmin	number	Room Owner or Administrator Operation

## TUIMediaDevice

Field	Type	Description
kMicrophone	number	Mic
kCamera	number	Camera
kScreen	number	Screen Sharing

## TUIRequestAction

Room Type

Field	Type	Description

kInvalidAction	number	Invalid Operation
kRequestToOpenRemoteCamera	number	Request Remote Camera On
kRequestToOpenRemoteMicrophone	number	Request Remote Mic On
kRequestToConnectOtherRoom	number	Request Remote Cross-room Streaming, web side temporarily unsupported
kRequestToTakeSeat	number	Request Go Live
kRequestRemoteUserOnSeat	number	Request Remote Go Live

## TUIRequestCallbackType

Request Type

Field	Type	Description
kRequestAccepted	number	Peer Accepted
kRequestRejected	number	Peer Rejected
kRequestCancelled	number	Request Canceled
kRequestTimeout	number	Request Timeout
kRequestError	number	Request Error

## Type Definition

### TUILoginUserInfo

Current Logged-in User Information

Field	Type	Description
userId	string	Login User's ID
userName	string	Login User's Name
avatarUrl	string	Login User's Avatar

### TUIRoomInfo

Room data, user can use `roomEngine.getRoomInfo` to get room data.

--	--	--

Field	Type	Description
roomId	string	Room Number, String Type Room Number
roomType	<a href="#">TUIRoomType</a>	Room Type
owner	string	Host's userId
name	string	Room ID
createTime	string	Creation time
roomMemberCount	number	Current total number of people in the room
maxSeatCount	number	Maximum number of microphone positions in the room
enableVideo	boolean	Allow users to join and turn on Audio
enableAudio	boolean	Allow users to join and turn on Video
enableMessage	boolean	Allow users to join and send messages
enableSeatControl	boolean	Enable microphone position control

## TUIUserInfo

### User Information

Field	Type	Description
userId	string	User Id
userName	string	User Name
avatarUrl	string	User Avatar
userRole	<a href="#">TUIRole</a>	User Role
hasAudioStream	boolean	Whether there are Audio streams
hasVideoStream	boolean	Whether there are Video streams
hasScreenStream	boolean	Whether there is Screen Sharing stream

## TUIMessage

### Message Information

Field	Type	Description



messageId	string	Message Id
message	string	Message
timestamp	number	Timestamp information, accurate to seconds
userId	string	User Id
userName	string	User name
avatarUrl	string	User Avatar

## TUIRequest

### Request Information

Field	Type	Description
requestAction	<a href="#">TUIRequestAction</a>	Request Type
timestamp	number	Request Initiation Time
requestId	string	Request Id v1.0.2 and above versions requestId type is string; v1.0.0 and v1.0.1 versions requestId type is number;
userId	string	Initiate Request's User Id
content	string	Other Content

## TUIRequestCallback

### Request Callback Information

Field	Type	Description
requestCallbackType	<a href="#">TUIRequestCallbackType</a>	Request Callback Type, Accept/Reject/Cancel/Timeout/Error
requestId	string	Request Id v1.0.2 and above versions requestId type is string; v1.0.0 and v1.0.1 versions requestId type is number;
userId	string	User Id
code	number	Request Response Code

message	string	Request Status Supplemental Description
---------	--------	---

## TUISeatInfo

### Microphone Position Information

Field	Type	Description
index	number	Microphone Position Number
userId	string	Microphone Position Correspondence's User Id
locked	boolean	Whether the current microphone position is locked
videoMuted	boolean	Whether the current microphone position prohibits Video
audioMuted	boolean	Whether the current microphone position prohibits Audio

## TUISeatLockParams

### Microphone Lock Status

Field	Type	Description
lockSeat	boolean	Lock Microphone Position
lockVideo	boolean	Lock Microphone Position Video
lockAudio	boolean	Lock Microphone Position Audio

## TUINetwork

### Network Information

Field	Type	Description
userId	string	User ID
quality	TUINetworkQuality	Network Quality
upLoss	number	Upstream Packet Loss Rate, Unit (%) The smaller the value, the better, currently only local users have this information
downLoss	number	Downstream Packet Loss Rate, Unit (%) The smaller the value, the better, currently only local users have

		this information
delay	number	Network Latency, Unit ms, currently only local users have this information

# Flutter

## RoomKit API

Last updated : 2024-05-13 11:06:01

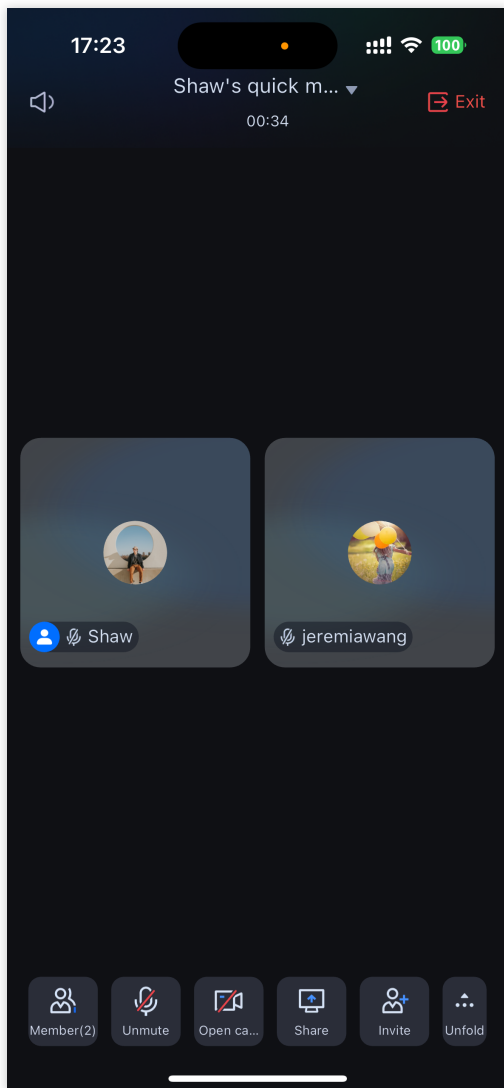
### Introduction

The TUIRoomKit API is a multi-person meeting component with an **Including UI Interface**. By using the TUIRoomKit API, you can quickly implement a meeting-like scenario through a simple interface. For more detailed integration steps, see: [Integration](#).

This document will provide a detailed introduction to the classes and related interfaces you may use in Flutter TUIRoomKit. By referring to this document, you can gain a more comprehensive understanding of how to use Flutter TUIRoomKit.

### ConferenceMainPage

Conference Main Page



Parameter	Type	Description
conferenceId	String	Conference ID required for creating or joining a conference
isCreateConference	bool	Whether it is for creating a conference (true for creating, false for joining)
conferenceParams	<a href="#">ConferenceParams</a>	Parameters related to creating or joining a conference
conferenceObserver	<a href="#">ConferenceObserver</a>	Conference status change callback listener

**Note :**

When you use [ConferenceSession](#) to create or join a conference, you do not need to pass any of the parameters here.

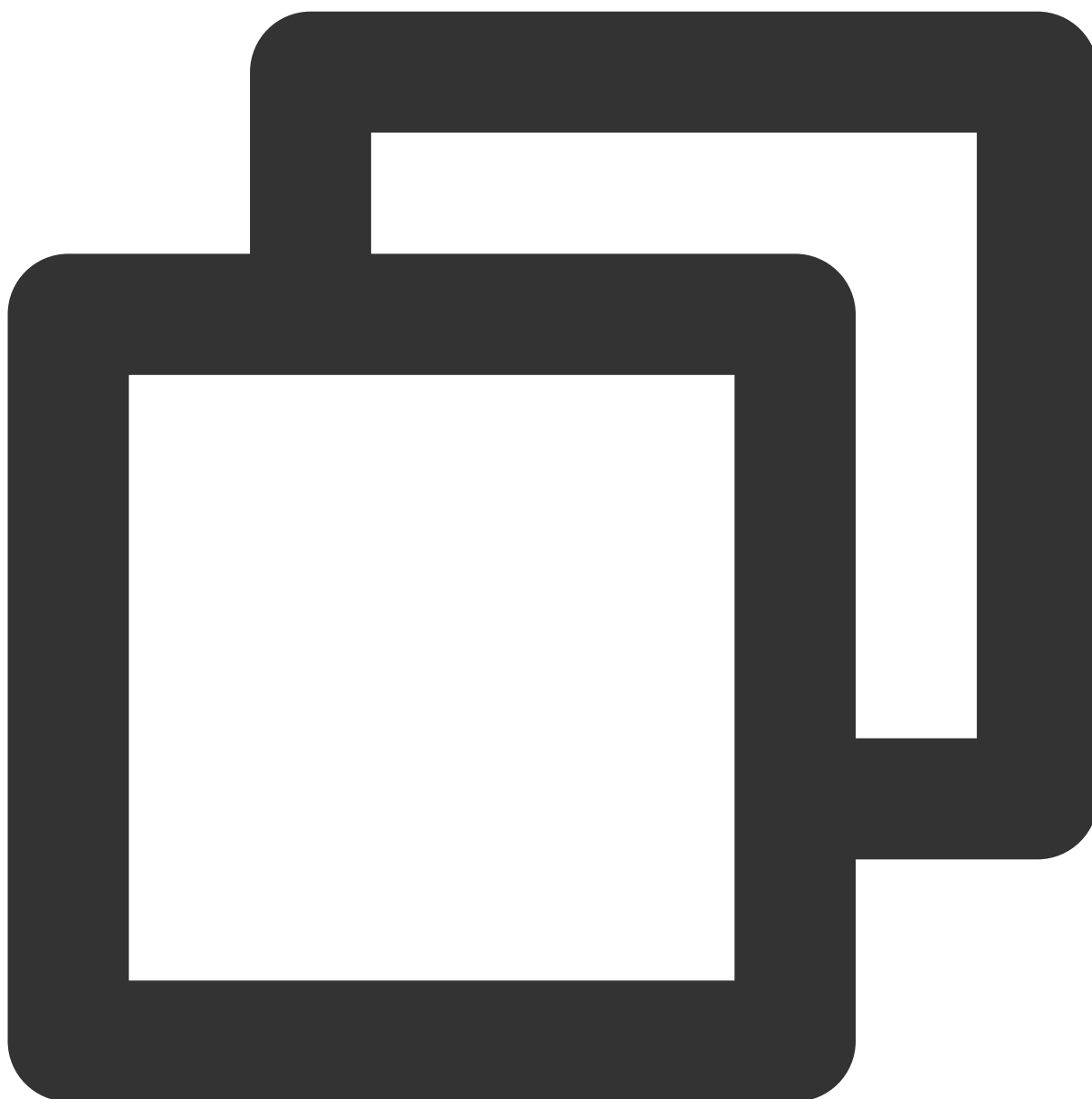
# ConferenceSession

When you expect to navigate to the conference page after successfully creating/joining a conference, you can use the `ConferenceSession` class to perform related operations.

Parameter	Type	Description
<code>isMuteMicrophone</code>	<code>bool</code>	Whether to mute the microphone (default is false)
<code>isOpenCamera</code>	<code>bool</code>	Whether to turn on the camera (default is false)
<code>isSoundOnSpeaker</code>	<code>bool</code>	Whether to use speakers (default is true)
<code>name</code>	<code>String</code>	Conference name (default is your conference ID, it is invalid when joining the conference)
<code>enableMicrophoneForAllUser</code>	<code>bool</code>	Whether to enable microphone permission for all members (default is true, invalid when joining a conference)
<code>enableCameraForAllUser</code>	<code>bool</code>	Whether to enable camera permissions for all members (default is true, invalid when joining a conference)
<code>enableMessageForAllUser</code>	<code>bool</code>	Whether to enable the speaking permission of all members (default is true, invalid when joining a conference)
<code>enableSeatControl</code>	<code>bool</code>	Whether to enable speaking mode on stage (default is false, invalid when joining a conference)
<code>onActionSuccess</code>	<code>VoidCallback</code>	Callback for successful creation/joining of a conference. You can navigate to the meeting page in this callback.
<code>onActionError</code>	Function ( <code>ConferenceError</code> , <code>String</code> )	Callback for failed creation/joining of a conference.

## newInstance

Create a new `ConferenceSession` object.

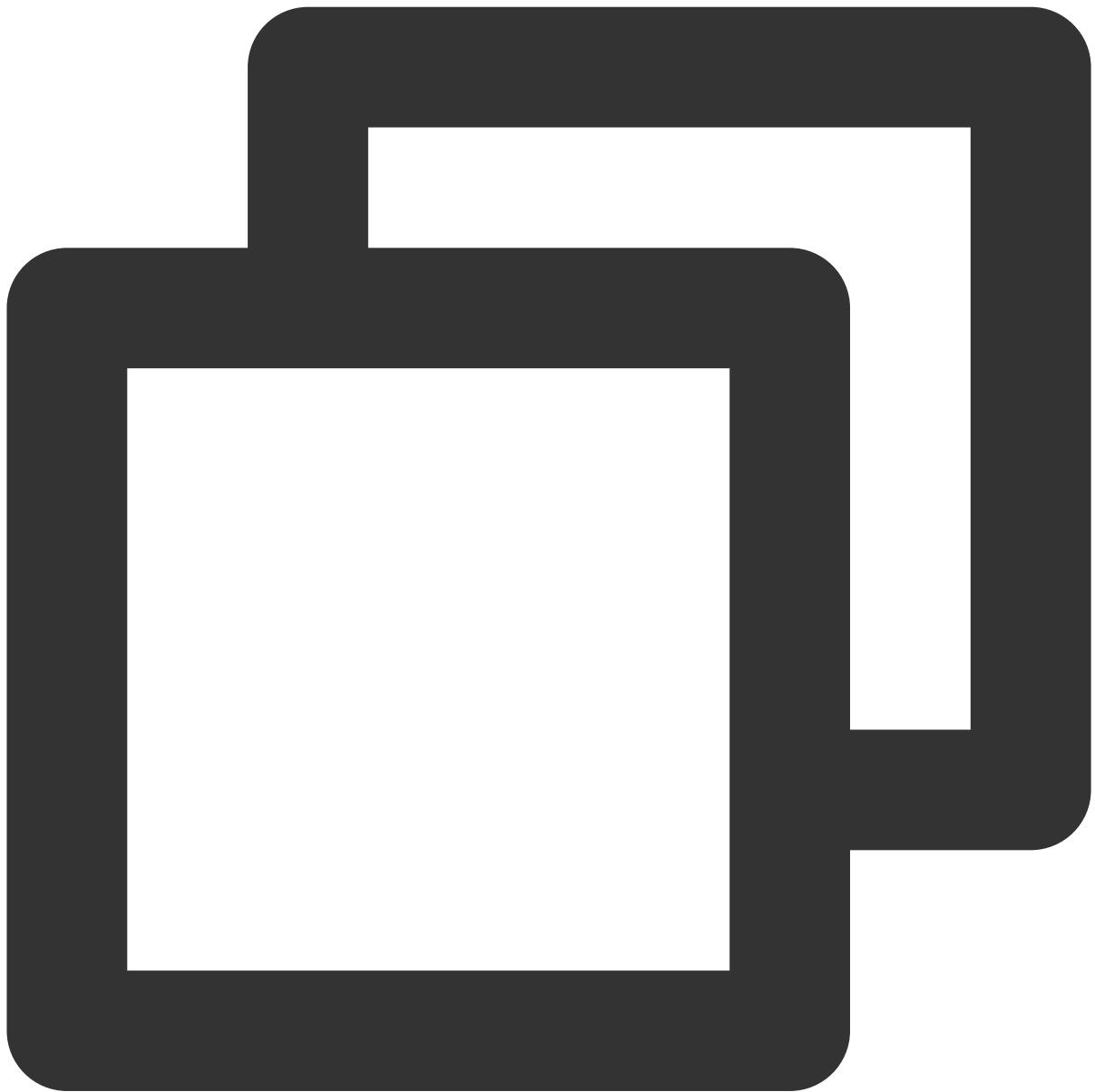


```
factory ConferenceSession.newInstance(String id)
```

Parameter	Type	Description
id	String	Conference ID required for creating or joining a conference

## quickStart

Quickly create conference interfaces.



```
Future<void> quickStart ()
```

## join

Join the conference interface.





```
Future<void> join()
```

**Note :**

Before calling the quick conference creation or joining conference interface, you need to complete all the parameters of the `ConferenceSession` that you need to set. For details, please refer to [Pre-conference Control](#).

When navigating directly to [ConferenceMainPage](#) and passing in relevant parameters to create/join a conference, there is no need to use `ConferenceSession`.

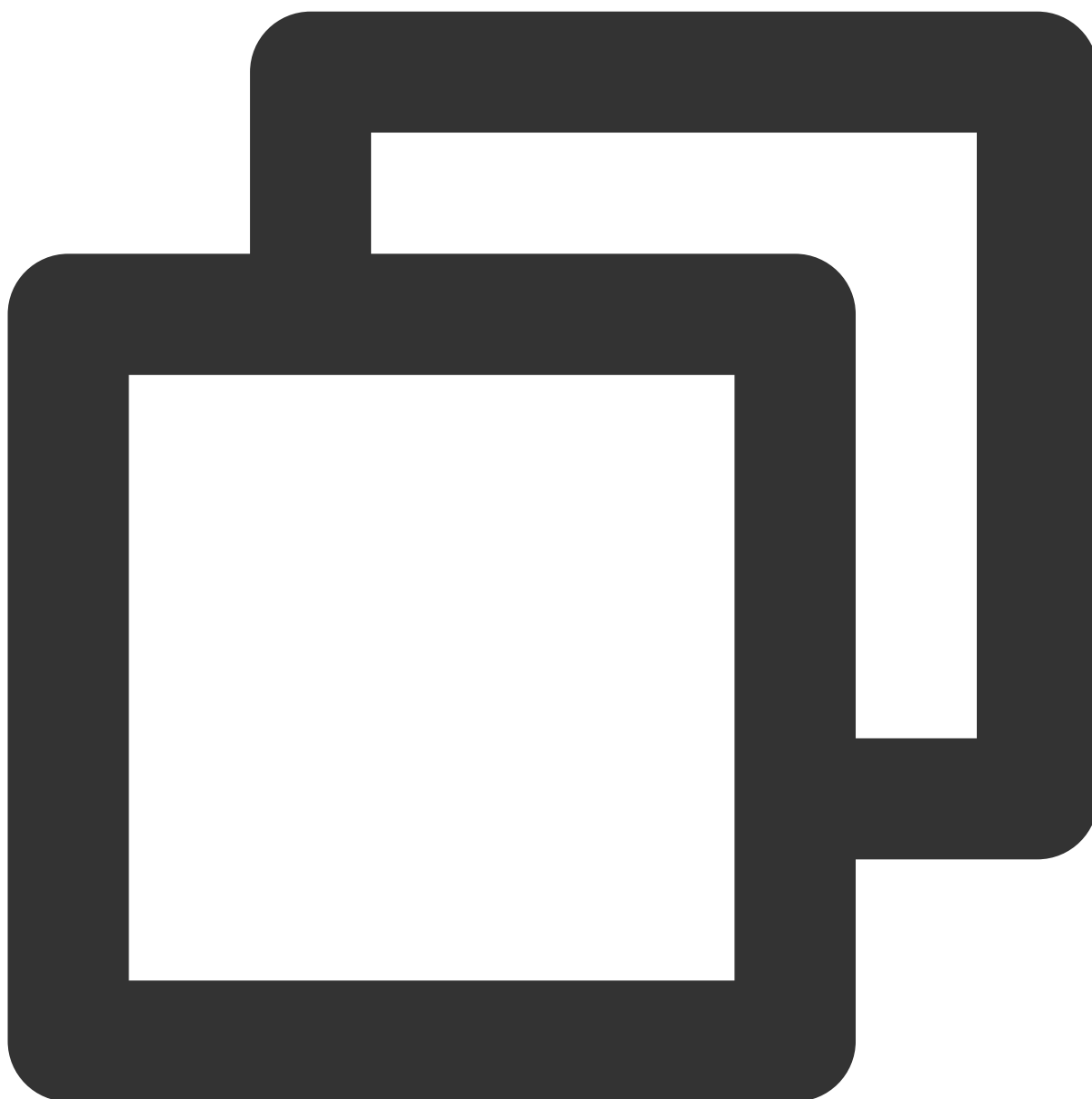
## ConferenceParams

Parameter	Type	Description
isMuteMicrophone	bool	Whether to mute the microphone (default is false)
isOpenCamera	bool	Whether to turn on the camera (default is false)
isSoundOnSpeaker	bool	Whether to use speakers (default is true)
name	String	Conference name (default is your conference ID, it is invalid when joining the conference)
enableMicrophoneForAllUser	bool	Whether to enable microphone permission for all members (default is true, invalid when joining a conference)
enableCameraForAllUser	bool	Whether to enable camera permissions for all members (default is true, invalid when joining a conference)
enableMessageForAllUser	bool	Whether to enable the speaking permission of all members (default is true, invalid when joining a conference)
enableSeatControl	bool	Whether to enable speaking mode on stage (default is false, invalid when joining a conference)
onActionSuccess	VoidCallback	Callback for successful creation/joining of a conference. You can navigate to the meeting page in this callback.
onActionError	Function ( <a href="#">ConferenceError</a> , String)	Callback for failed creation/joining of a conference.

## ConferenceObserver

### onConferenceStarted

Conference start event.

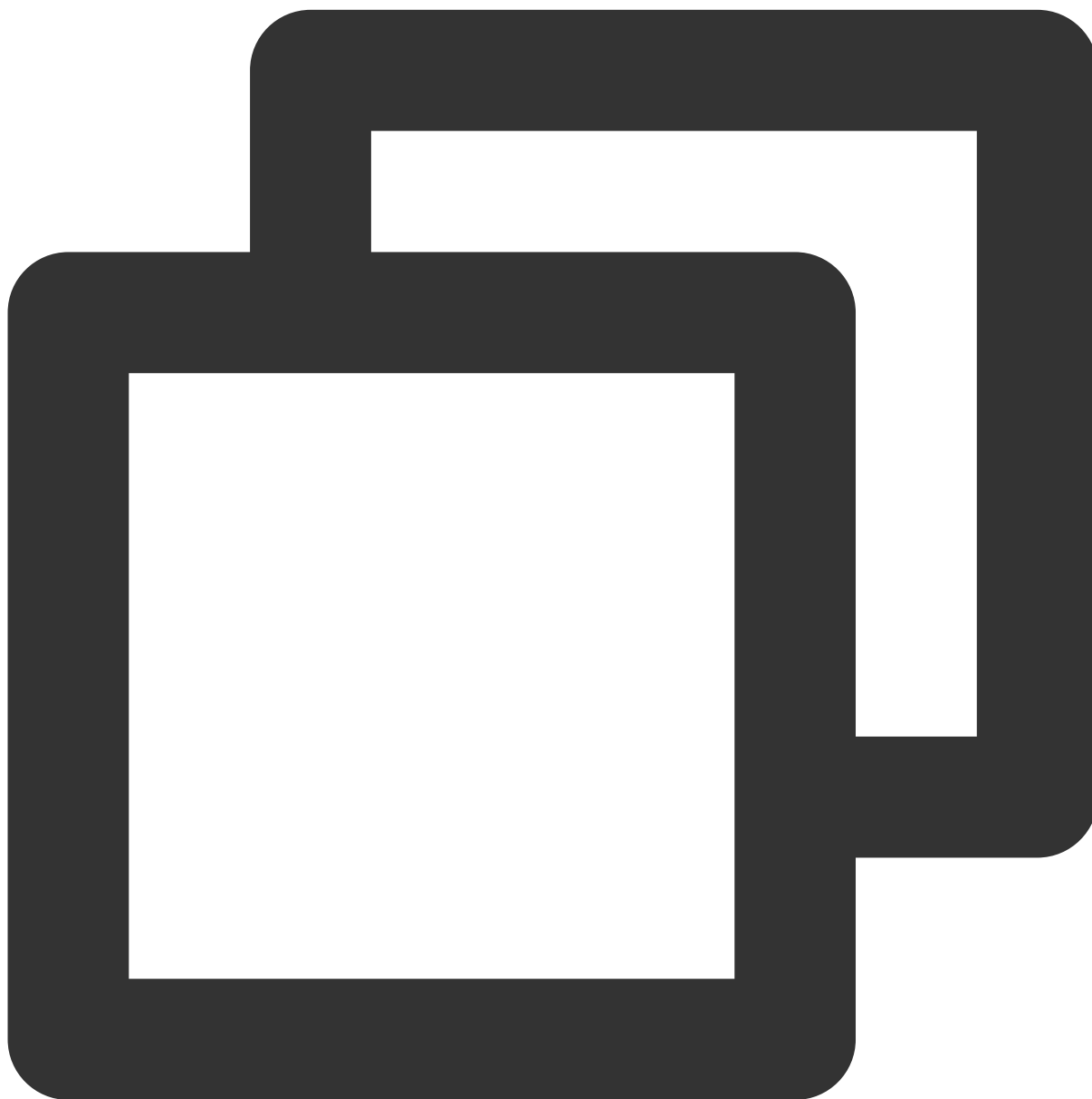


```
Function(String conferenceId, ConferenceError error) onConferenceStarted
```

Parameter	Type	Description
conferenceId	String	Conference id
error	<a href="#">ConferenceError</a>	Error code

## onConferenceJoined

Join the conference event.

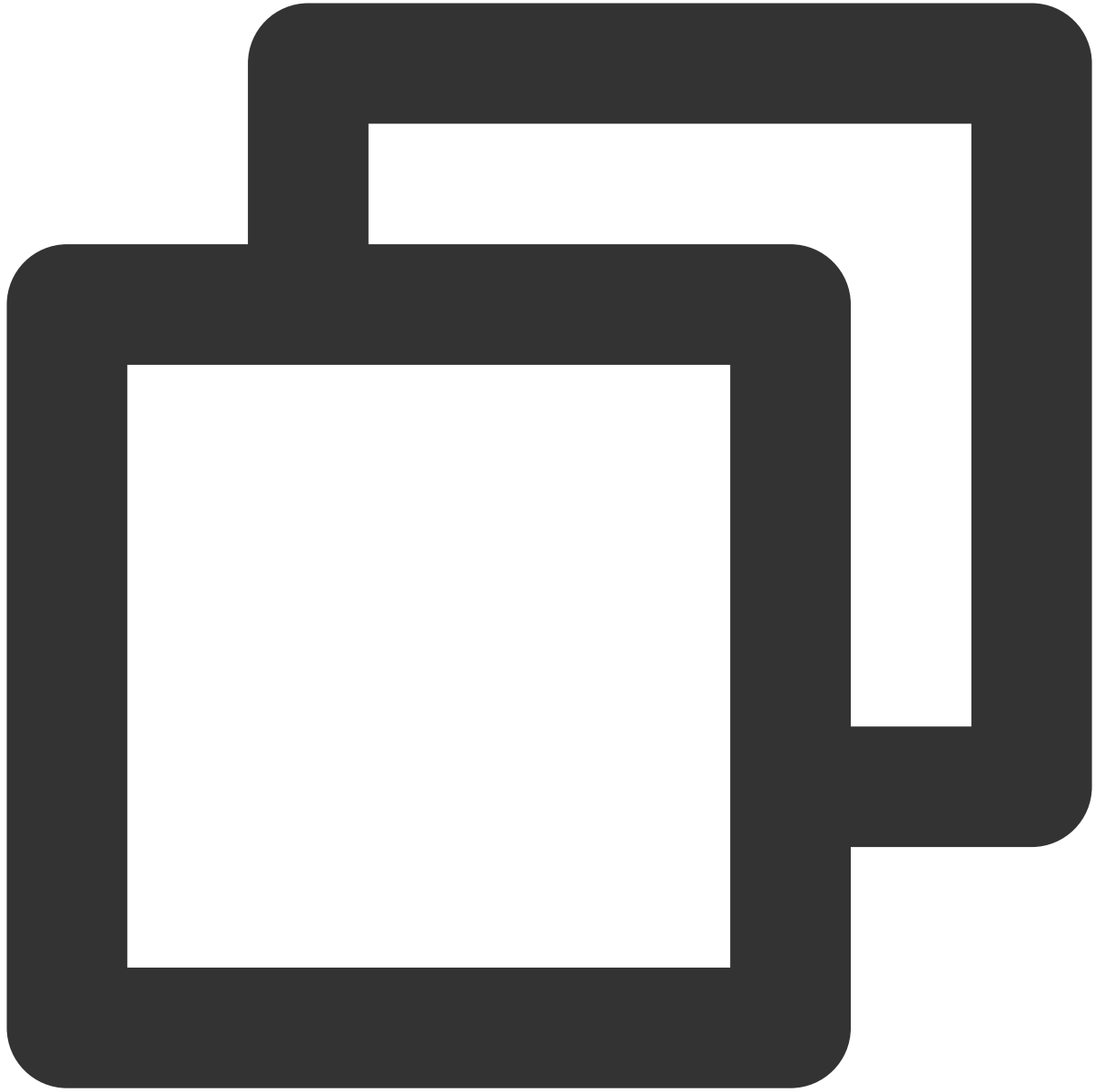


```
Function(String conferenceId, ConferenceError error) onConferenceJoined
```

Parameter	Type	Description
conferenceId	String	Conference Id
error	<a href="#">ConferenceError</a>	Error code

### onConferenceFinished

Meeting end event, this callback will be triggered when the meeting is actively ended or the meeting is dismissed.



```
Function(String conferenceId) onConferenceFinished
```

Parameter	Type	Description
conferenceId	String	Conference Id

### **onConferenceExited**

Exit meeting event, this callback will be triggered when actively exiting the meeting or being kicked out of the meeting.



```
Function(String conferenceId) onConferenceFinished
```

Parameter	Type	Description
conferenceId	String	Conference Id

## ConferenceError

## Error code

Enumeration	Value	Description
success	0	Operation Successful
errFailed	-1	Temporarily Unclassified General Error
errConferenceldNotExist	-2100	Room Does Not Exist When Entering, May Have Been Closed
errConferenceldInvalid	-2105	Illegal Custom Room ID, Must Be Printable ASCII Characters (0x20-0x7e), Up to 48 Bytes Long
errConferenceldOccupied	-2106	Room ID is Already in Use, Please Choose Another Room ID
errConferenceNameInvalid	-2107	Illegal Room Name, Maximum 30 Bytes, Must Be UTF-8 Encoding if Contains Chinese Characters

# RoomEngine API

## API Overview

Last updated : 2023-11-21 15:20:12

### TUIRoomEngine API List

TUIRoomEngine API is the Audio/Video call Component's No UI Interface, you can use this set of API to customize packaging according to your business needs.

TUIRoomEngine

#### TUIRoomEngine Core Methods

API	Description
<a href="#">createInstance</a>	Create TUIRoomEngine Instance
<a href="#">destroyInstance</a>	Destroy TUIRoomEngine Instance
<a href="#">login</a>	Login interface, you need to initialize user information before entering the room and perform a series of operations.
<a href="#">logout</a>	Logout interface, there will be actively leave room operation, destroy resources
<a href="#">setSelfInfo</a>	Set local user name and avatar
<a href="#">setLoginUserInfo</a>	Set login user information
<a href="#">getSelfInfo</a>	Get local user basic information
<a href="#">addObserver</a>	Set event callback
<a href="#">removeObserver</a>	Remove event callback

#### Room Related Active Interface

API	Description
<a href="#">createRoom</a>	Create room
<a href="#">destroyRoom</a>	Close the room



<a href="#">enterRoom</a>	Entered room
<a href="#">exitRoom</a>	Leave room
<a href="#">connectOtherRoom</a>	Connect to other room
<a href="#">disconnectOtherRoom</a>	Disconnect from other room
<a href="#">fetchRoomInfo</a>	Get room data
<a href="#">updateRoomNameByAdmin</a>	Update room name
<a href="#">updateRoomSpeechModeByAdmin</a>	Set room management mode (only administrator or group owner can call)

## Local User View Rendering, Video Management

API	Description
<a href="#">setLocalVideoView</a>	Set the view control for local user video rendering
<a href="#">openLocalCamera</a>	Open local camera
<a href="#">closeLocalCamera</a>	Close local camera
<a href="#">updateVideoQuality</a>	Update local video codec quality settings
<a href="#">updateVideoQualityEx</a>	Set the encoding parameters of video encoder
<a href="#">setVideoResolutionMode</a>	Set the resolution mode of video encoder
<a href="#">enableGravitySensor</a>	Enable the gravity sensor
<a href="#">startPushLocalVideo</a>	Start pushing local video
<a href="#">stopPushLocalVideo</a>	Stop pushing local video
<a href="#">startScreenSharing</a>	Start screen sharing
<a href="#">stopScreenSharing</a>	Stop screen sharing

## Local User Audio Management

API	Description
<a href="#">openLocalMicrophone</a>	Open local microphone
<a href="#">closeLocalMicrophone</a>	Close local microphone

<a href="#">updateAudioQuality</a>	Update local audio codec quality settings
<a href="#">muteLocalAudio</a>	Mute local audio
<a href="#">unMuteLocalAudio</a>	UnMute local audio

## Remote User View Rendering, Video Management

API	Description
<a href="#">setRemoteVideoView</a>	Set the view control for remote user video rendering
<a href="#">startPlayRemoteVideo</a>	Start playing remote user video
<a href="#">stopPlayRemoteVideo</a>	Stop playing remote user video
<a href="#">muteRemoteAudioStream</a>	Mute remote user

## Room User Information

API	Description
<a href="#">getUserList</a>	Get the member list in the room
<a href="#">getUserInfo</a>	Get member information

## Room User Management

API	Description
<a href="#">changeUserRole</a>	Modify user role (only administrator or group owner can call)
<a href="#">kickRemoteUserOutOfRoom</a>	Kick Remote User out of the Room (Only Administrator or Group Owner can call)
<a href="#">addCategoryTagForUsers</a>	Add category tags to users (Only Administrator or Group Owner can call)
<a href="#">removeCategoryTagForUsers</a>	Remove category tags to users (Only Administrator or Group Owner can call)
<a href="#">getUserListByTag</a>	Get user information in the room based on tags

## Speech Management in Room

--	--

API	Description
<a href="#">disableDeviceForAllUserByAdmin</a>	Control the permission status of whether all users in the current room can open Audio and Video streams capturing devices, such as: Prohibit all from turning on the mic, Prohibit all from turning on the Camera, Prohibit all from turning on Screen Sharing (currently only available in meeting scenes, and only administrators or group owners can invoke).
<a href="#">openRemoteDeviceByAdmin</a>	Request Remote User to Open Media Device (Only Administrator or Group Owner can call)
<a href="#">closeRemoteDeviceByAdmin</a>	Close Remote User's Media Device (Only Administrator or Group Owner can call)
<a href="#">applyToAdminToOpenLocalDevice</a>	Request to Open Local Media Device (Available for Ordinary Users)

## Microphone Seat Management in Room

API	Description
<a href="#">setMaxSeatCount</a>	Set Maximum Number of Microphone Seats (Only supported when entering the room and creating the room)
<a href="#">getSeatList</a>	Get Microphone Seat List
<a href="#">lockSeatByAdmin</a>	Lock Microphone Seat (Including Position Lock, Audio State Lock, Video State Lock)
<a href="#">takeSeat</a>	Apply to Go Live (Not Required in Free Speech Mode)
<a href="#">leaveSeat</a>	Apply to leave the live (Not Required in Free Speech Mode)
<a href="#">takeUserOnSeatByAdmin</a>	Host/Administrator invites user to go live
<a href="#">kickUserOffSeatByAdmin</a>	Host/Administrator kicks user off the microphone seat

## Signaling Management

API	Description
<a href="#">cancelRequest</a>	Cancel Request
<a href="#">responseRemoteRequest</a>	Reply to Request

## Send Message

API	Description
<a href="#">sendTextMessage</a>	Send Text Message
<a href="#">sendCustomMessage</a>	Send Custom Message
<a href="#">disableSendingMessageByAdmin</a>	Disable Remote User's Text Message Sending Ability (Only Administrator or Group Owner can call)
<a href="#">disableSendingMessageForAllUser</a>	Disable All Users' Text Message Sending Ability (Only Administrator or Group Owner can call) Advanced Feature: Get TRTC Instance

## Advanced Features

API	Description
<a href="#">switchCamera</a>	Switch front/rear camera
<a href="#">setBeautyLevel</a>	Set beauty level
<a href="#">setWhitenessLevel</a>	Set whiteness level

## Debugging related

API	Description
<a href="#">callExperimentalAPI</a>	Call experimental api

# TUIRoomObserver Callback Event

TUIRoomObserver is the Callback Event class corresponding to TUIRoomEngine. You can monitor the Callback Events you need through this Callback.

TUIRoomObserver

# TUIRoomObserver

## Error Callback

--	--

API	Description
<a href="#">onError</a>	Error Callback Event

## Login Status Event Callback

API	Description
<a href="#">onKickedOffLine</a>	User Kicked Offline Event
<a href="#">onUserSigExpired</a>	User Credential Timeout Event

## Room Event Callback

API	Description
<a href="#">onRoomNameChanged</a>	Room Name Change Event
<a href="#">onAllUserMicrophoneDisableChanged</a>	All Users' Microphones Disabled in Room Event
<a href="#">onAllUserCameraDisableChanged</a>	All Users' Cameras Disabled in Room Event
<a href="#">onSendMessageForAllUserDisableChanged</a>	All Users' Text Message Sending Disabled in Room Event
<a href="#">onKickedOutOfRoom</a>	Room Dismissed Event
<a href="#">onRoomDismissed</a>	Kicked Out of Room Event
<a href="#">onRoomSpeechModeChanged</a>	Room Microphone Control Mode Change

## Room User Event Callback

API	Description
<a href="#">onRemoteUserEnterRoom</a>	Remote User Entering Room Event
<a href="#">onRemoteUserLeaveRoom</a>	Remote User Leaving Room Event
<a href="#">onUserRoleChanged</a>	User Role Change Event
<a href="#">onUserVideoStateChanged</a>	User Video State Change Event
<a href="#">onUserAudioStateChanged</a>	User Audio State Change Event
<a href="#">onUserVoiceVolumeChanged</a>	User Volume Change Event
<a href="#">onSendMessageForUserDisableChanged</a>	User Text Message Sending Ability Change Event

<a href="#">onUserNetworkQualityChanged</a>	User Network Status Change Event
<a href="#">onUserScreenCaptureStopped</a>	Screen Sharing End Event

## Room Microphone Seat Event Callback

API	Description
<a href="#">onRoomMaxSeatCountChanged</a>	Room Maximum Microphone Seat Number Change Event (Only effective in conference type rooms)
<a href="#">onSeatListChanged</a>	Microphone Seat List Change Event
<a href="#">onKickedOffSeat</a>	Received User Kicked Off Microphone Event

## Request Signaling Event Callback

API	Description
<a href="#">onRequestReceived</a>	Received Request Message Event
<a href="#">onRequestCancelled</a>	Received Request Cancellation Event

## Room Message Event Callback

API	Description
<a href="#">onReceiveTextMessage</a>	Received Normal Text Message Event
<a href="#">onReceiveCustomMessage</a>	Received Custom Message Event

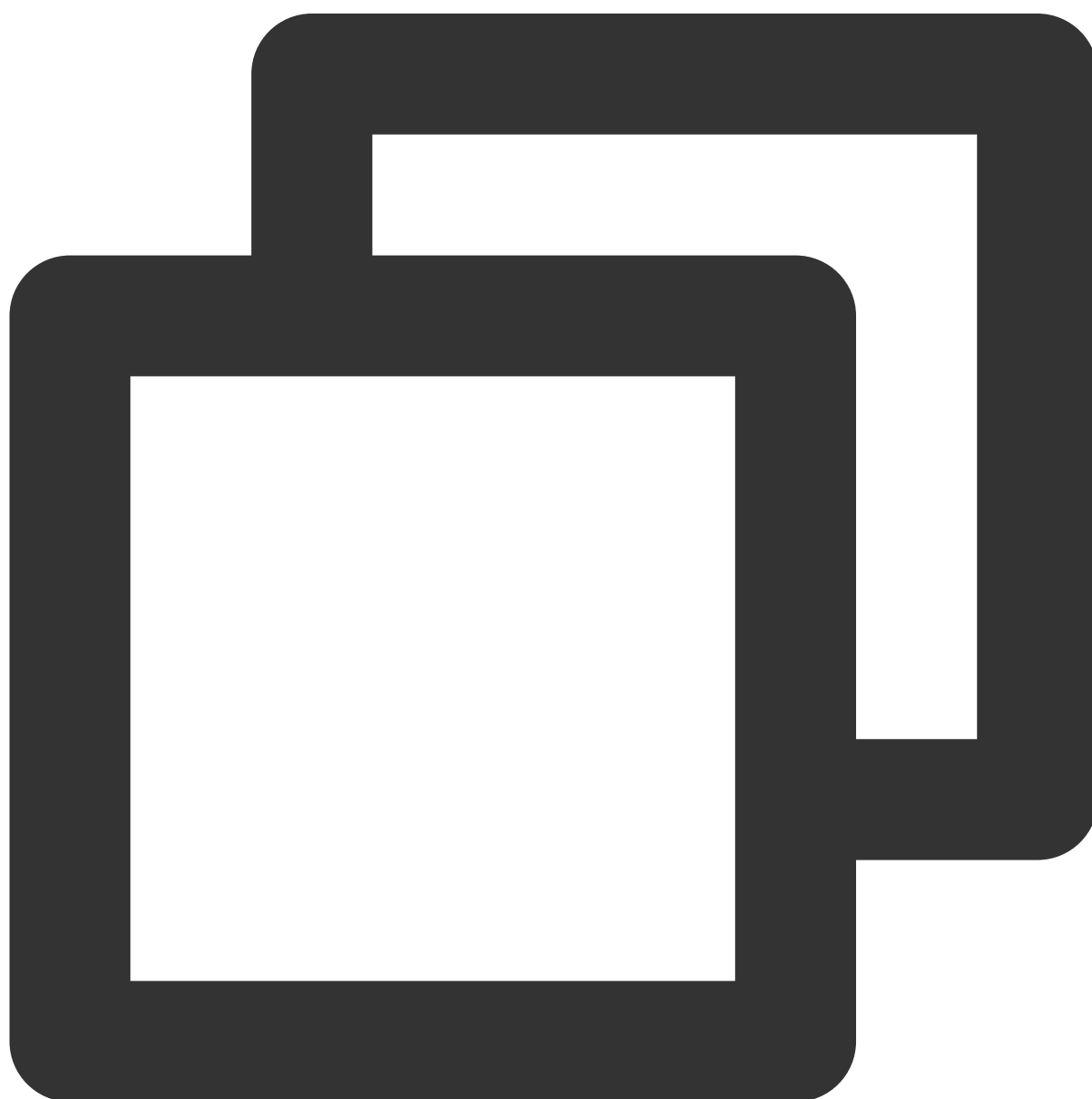
# TUIRoomObserver

Last updated : 2023-11-14 16:34:53

## TUIRoomEngine Event Callback

### **onError**

Error Event.

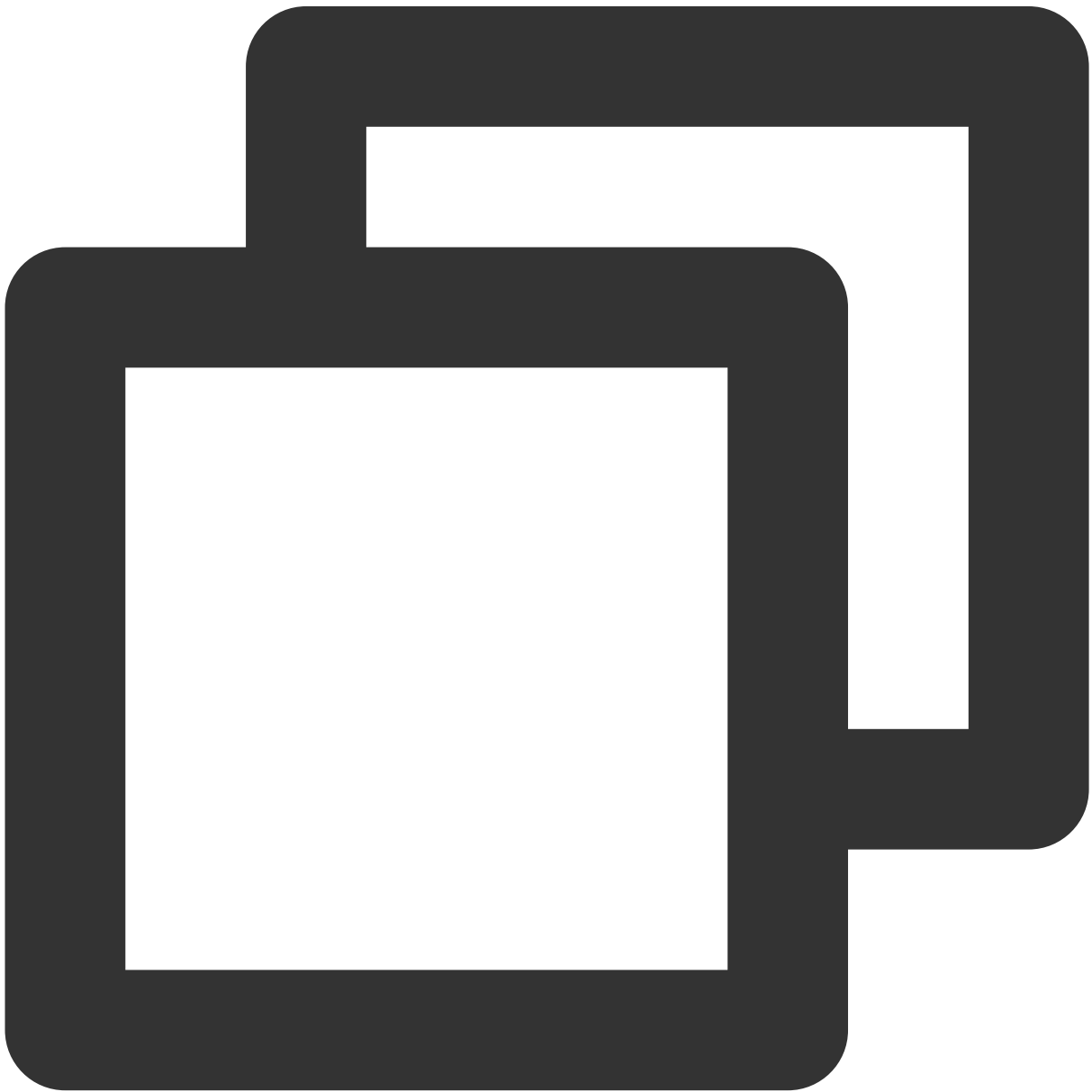


```
OnError onError = (TUIError errorCode, String message) {}
```

Parameter	Type	Description
errorCode	TUIError	Error Code
message	String	Error Message

### onKickedOffLine

Other terminals login and get kicked off event.



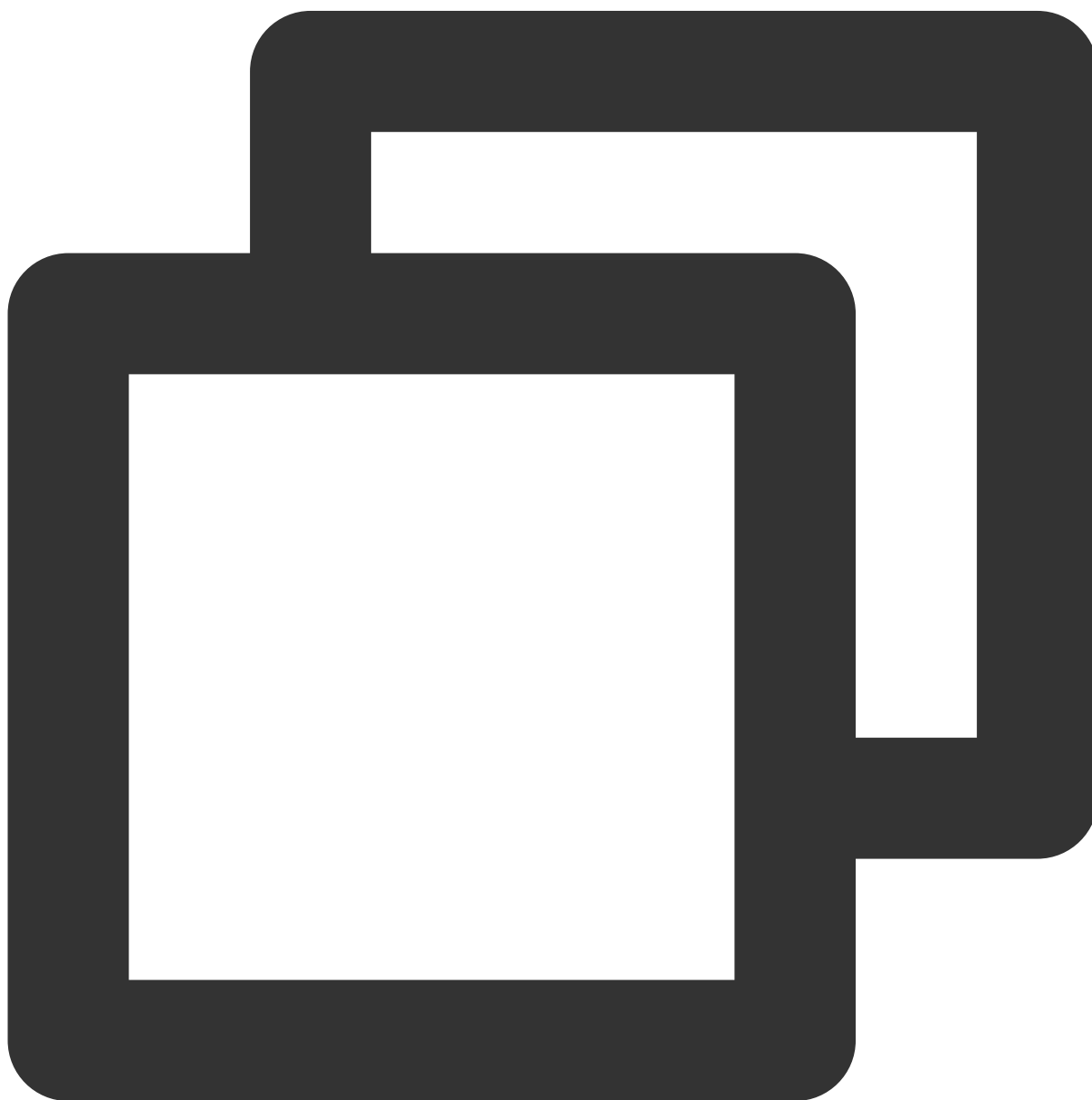


```
OnKickedOffLine onKickedOffLine = (String message) {}
```

Parameter	Type	Description
message	String	Kicked out description

## onUserSigExpired

User credential timeout event.



```
OnUserSigExpired onUserSigExpired = () {}
```

## onRoomNameChanged

Room name change event.



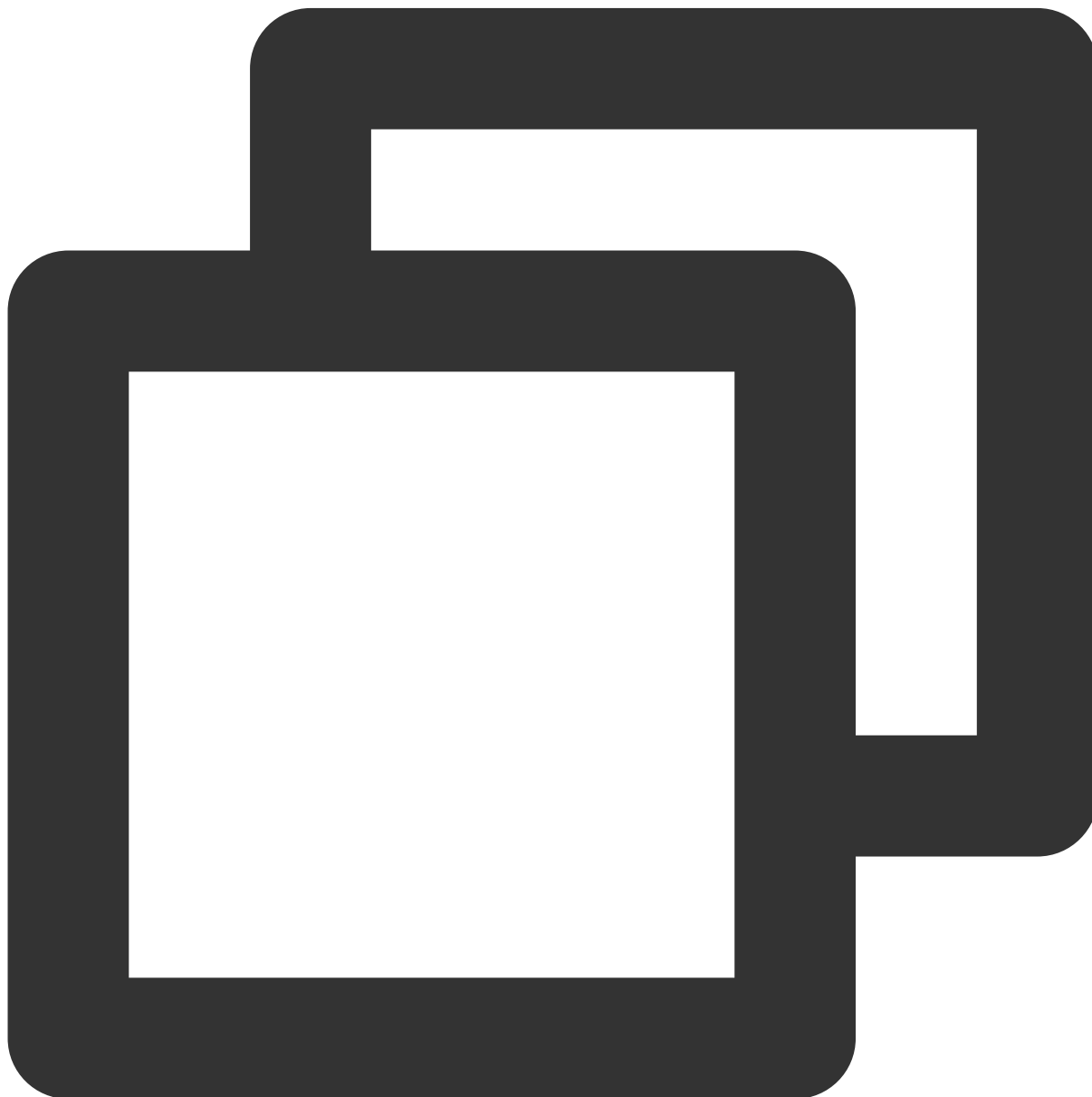
```
OnRoomNameChanged onRoomNameChanged = (String roomId, String roomName) {}
```

Parameter	Type	Description

roomId	String	Room ID
roomName	String	Room Name

### onAllUserMicrophoneDisableChanged

Inside the room, all users' mic is disabled event.



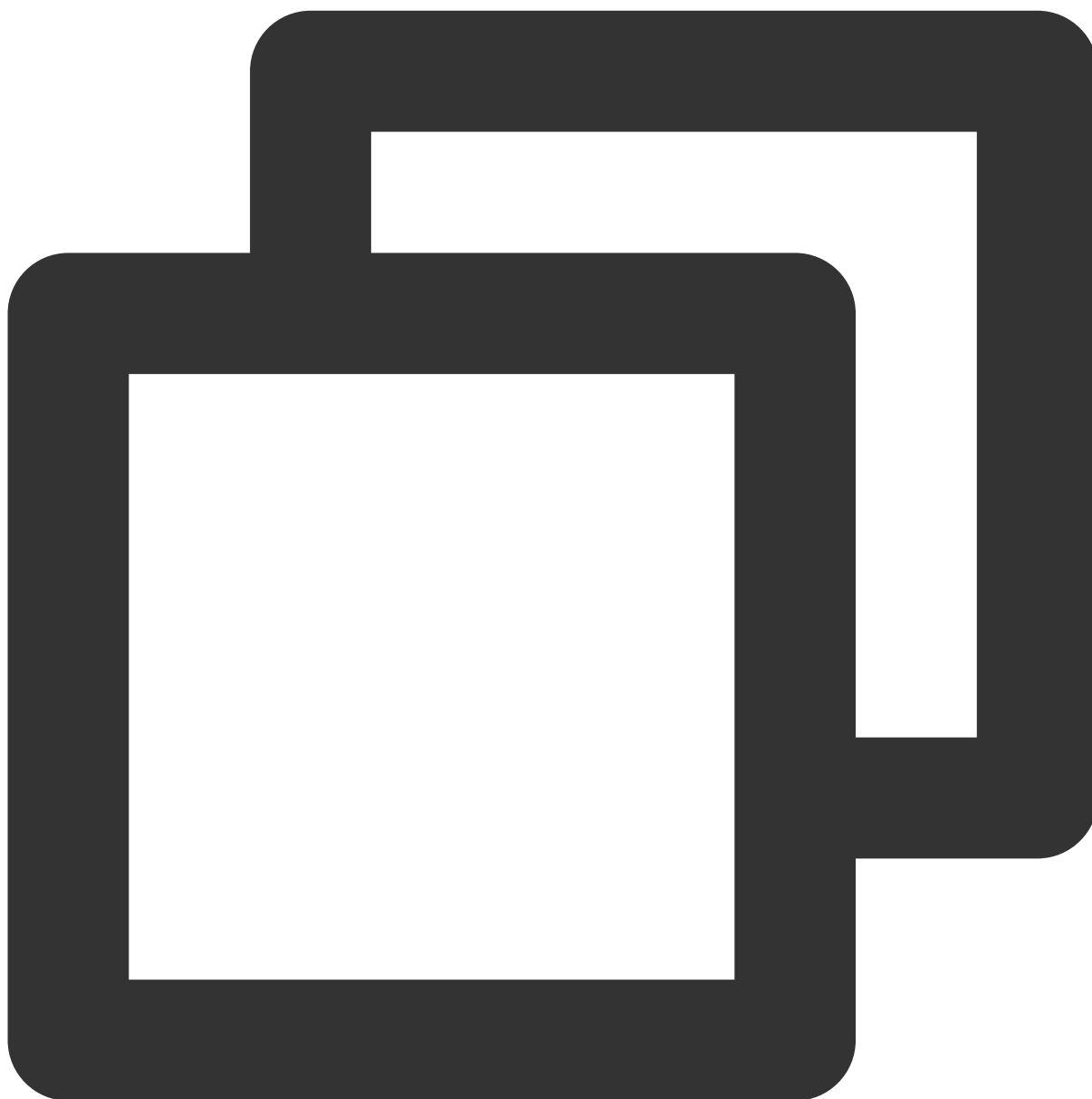
```
OnAllUserMicrophoneDisableChanged onAllUserMicrophoneDisableChanged = (String roomId
```

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

roomId	String	Room ID
isDisable	bool	Whether it is disabled

### onAllUserCameraDisableChanged

All users' Camera in the Room are disabled event.



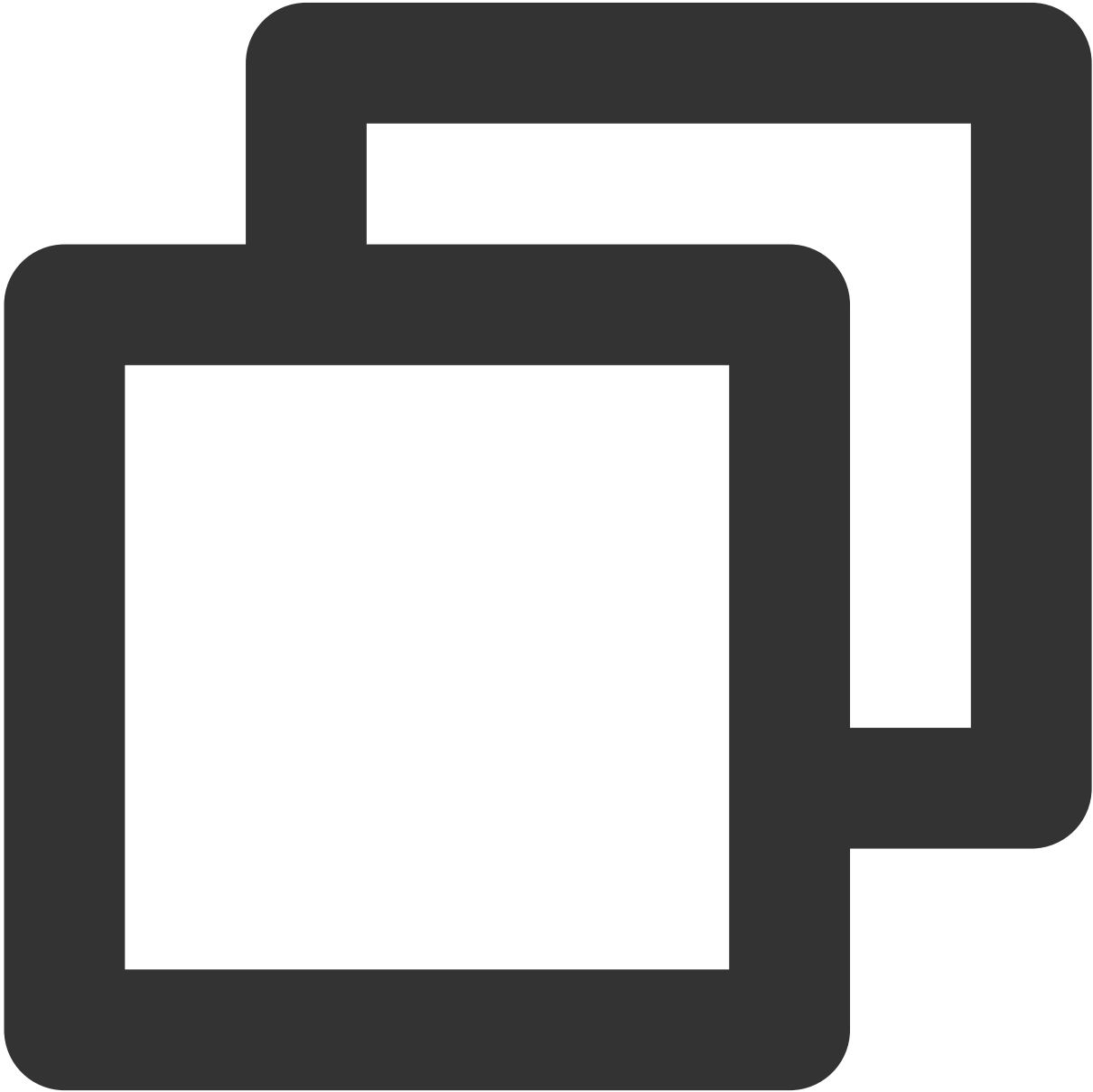
```
OnAllUserCameraDisableChanged onAllUserCameraDisableChanged = (String roomId, bool
```

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

roomId	String	Room ID
isDisable	bool	Whether it is disabled

onSendMessageForAllUserDisableChanged

Inside the room, all users' text message sending is disabled event.

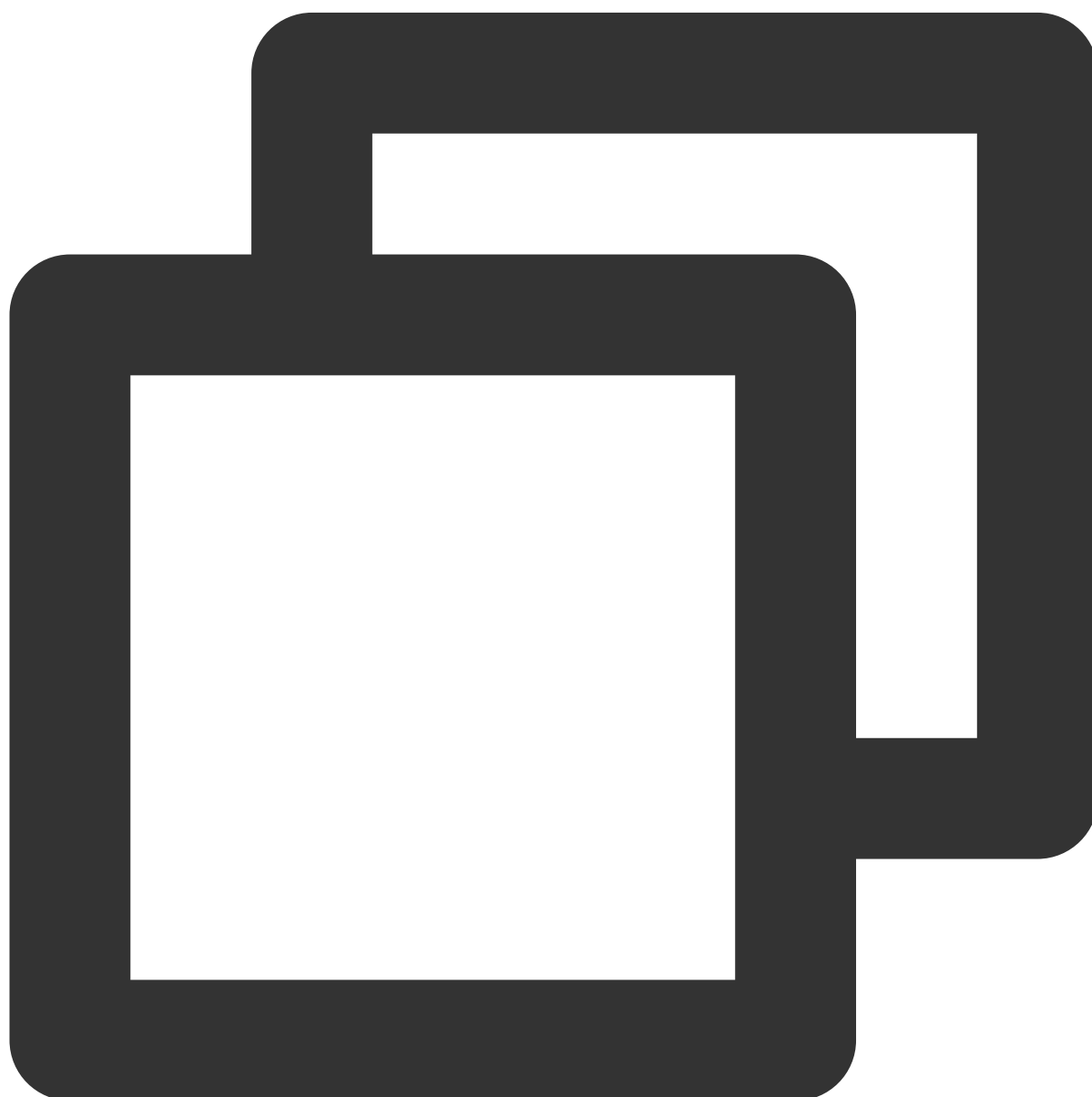


OnSendMessageForAllUserDisableChanged onSendMessageForAllUserDisableChanged = (Stri		

Parameter	Type	Description
roomId	String	Room ID
isDisable	bool	Whether it is disabled

## onRoomDismissed

Room dissolution event.

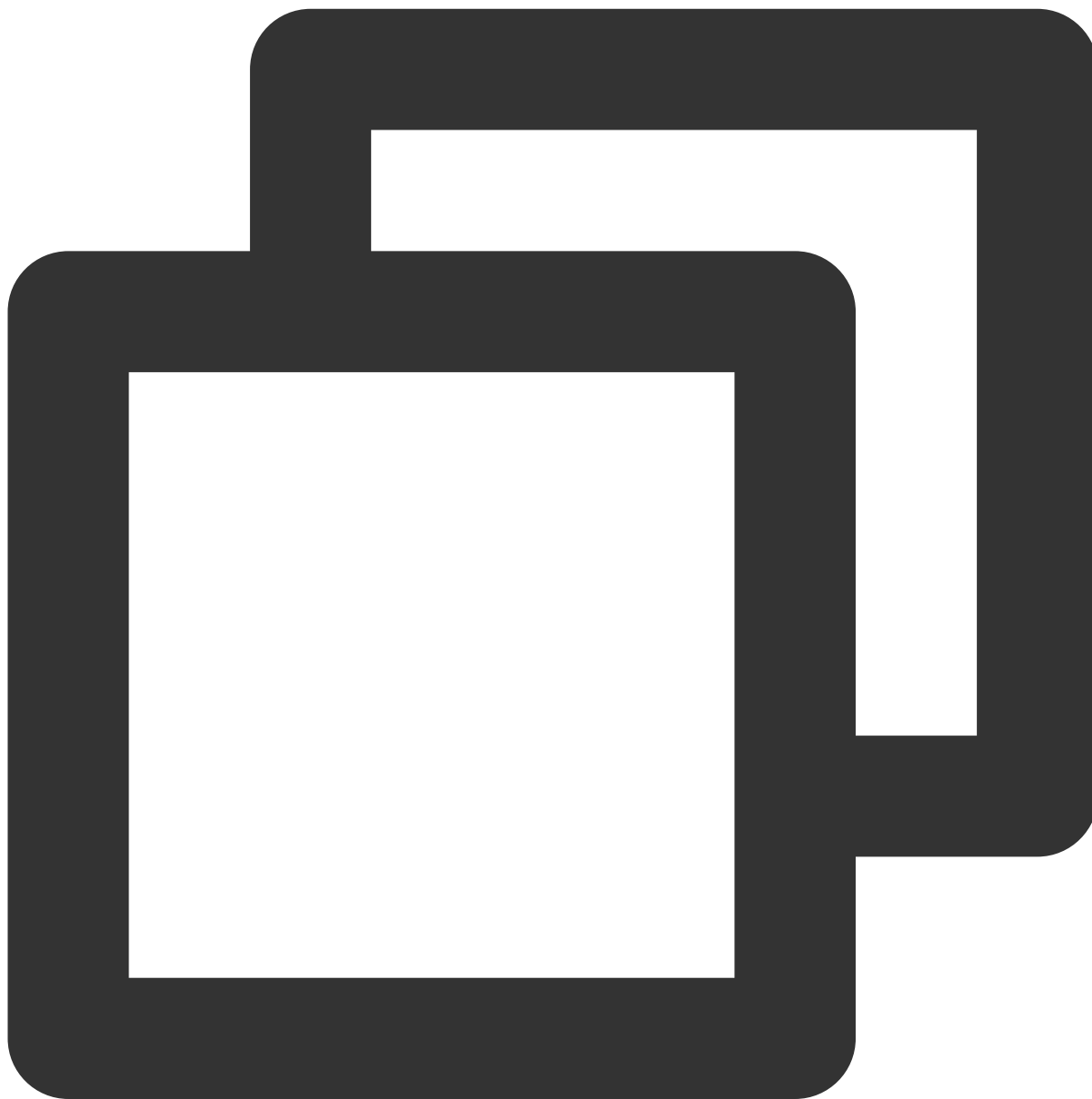


```
OnRoomDismissed onRoomDismissed = (String roomId) {}
```

Parameter	Type	Description
roomId	String	Room ID

## onKickedOutOfRoom

Kick out of the room event



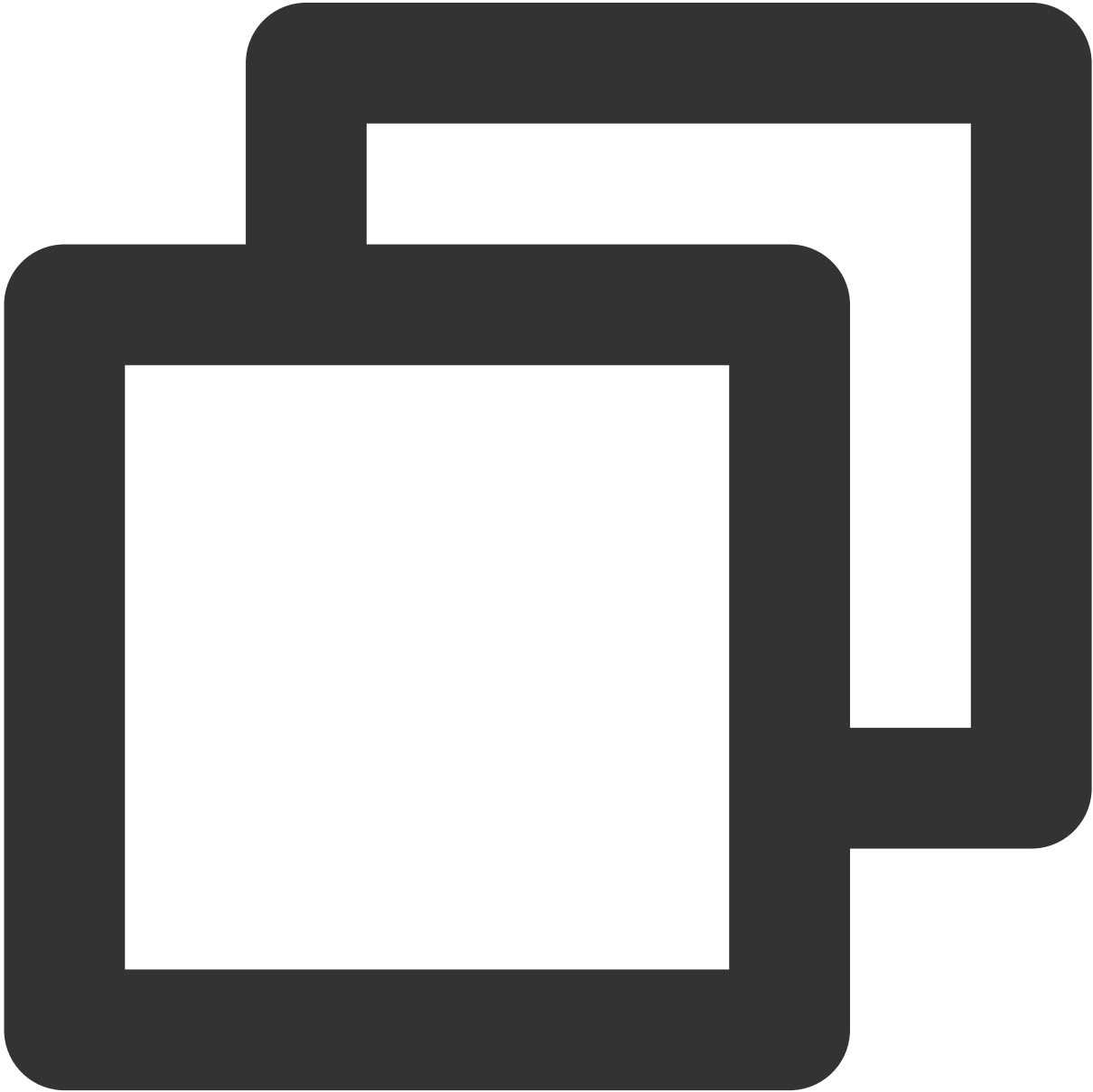
```
OnKickedOutOfRoom onKickedOutOfRoom = (String roomId, String message) {}
```

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

roomId	String	Room ID
message	String	Description of being kicked out

onRoomSpeechModeChanged

Mic control mode changes in the room.



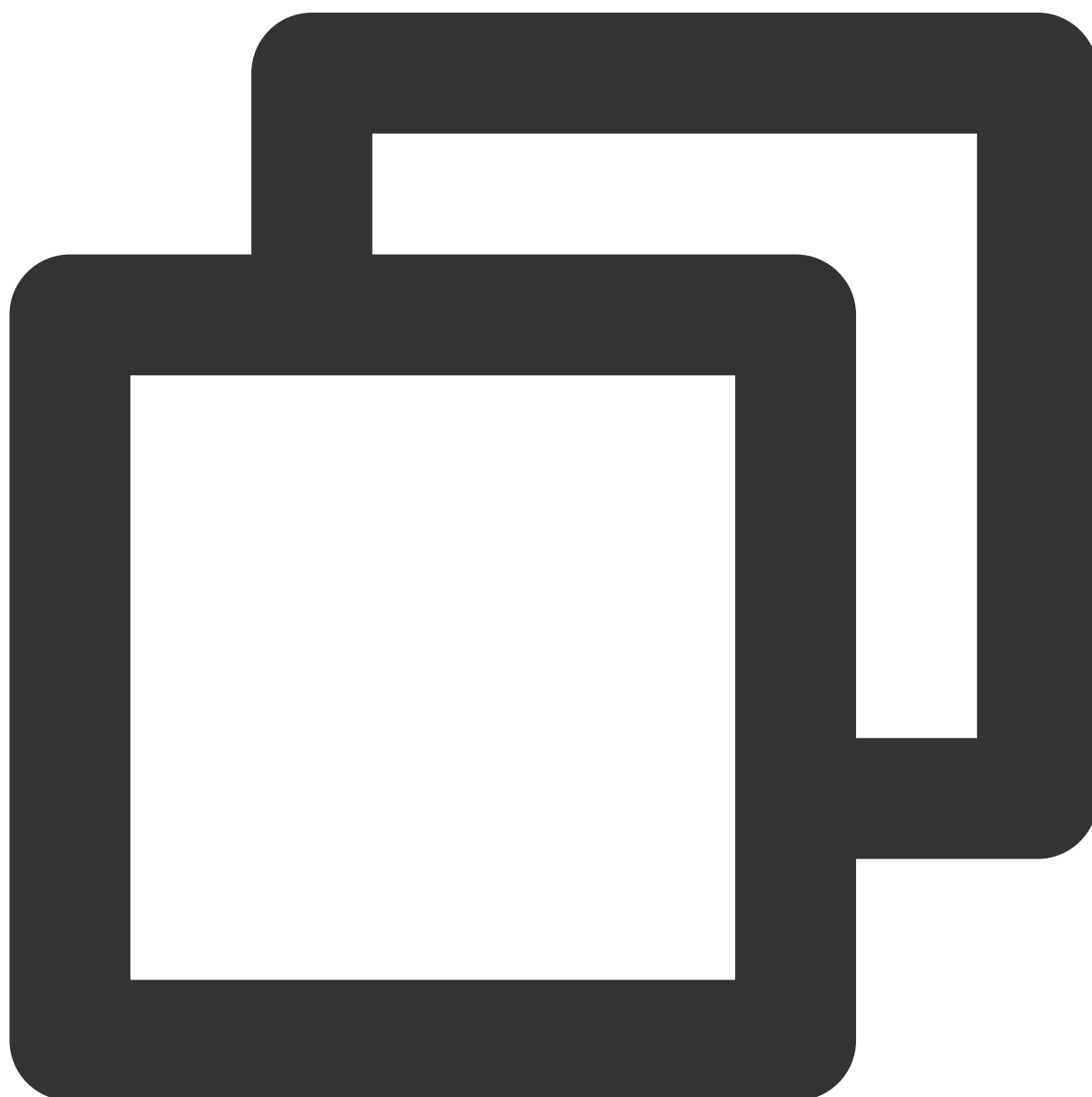
OnRoomSpeechModeChanged onRoomSpeechModeChanged = (String roomId, TUISpeechMode spe		



Parameter	Type	Description
roomId	String	Room ID
speechMode	<a href="#">TUISpeechMode</a>	Mic control mode

## onRemoteUserEnterRoom

Remote user enters the room event.

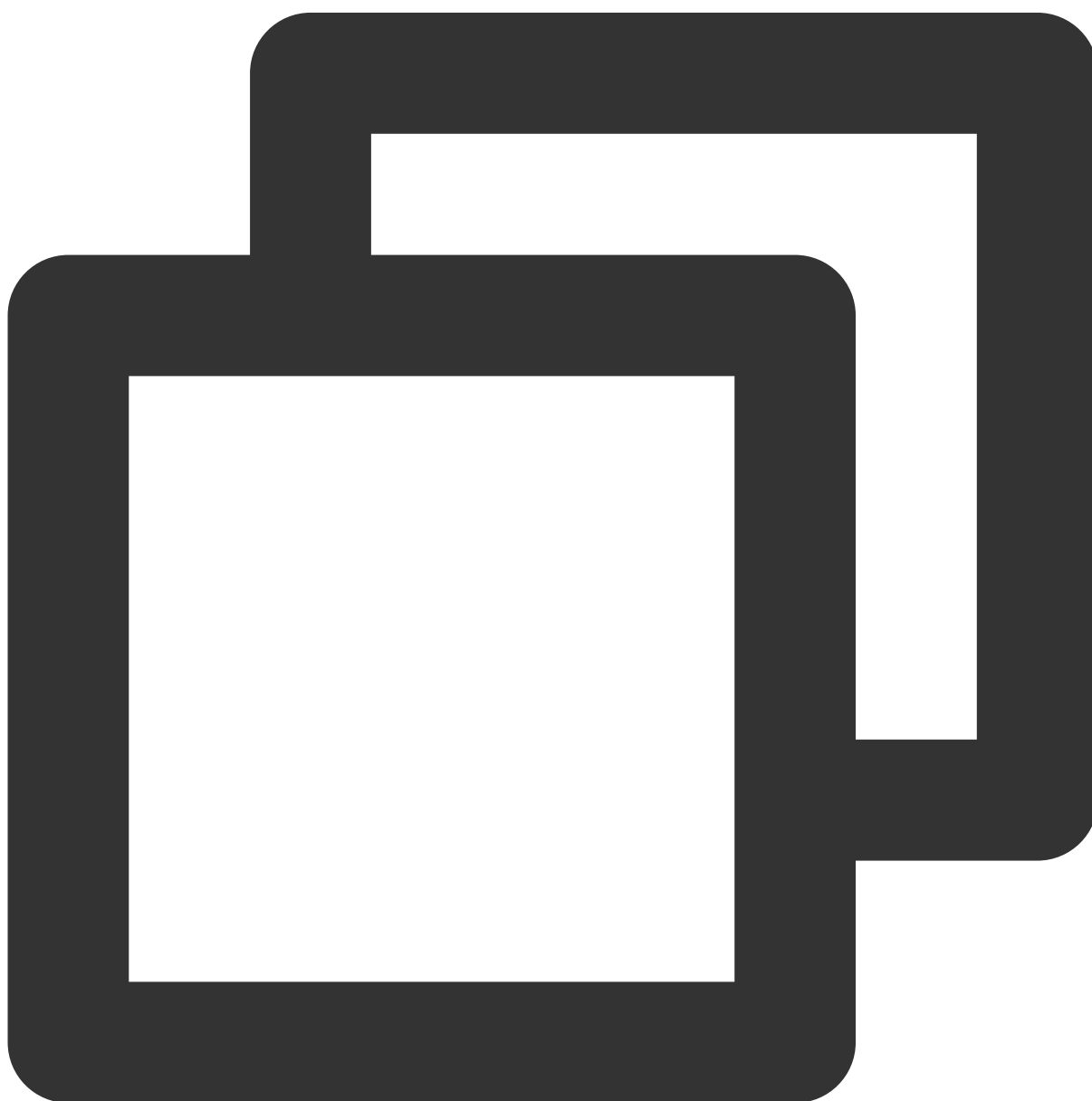


```
OnRemoteUserEnterRoom onRemoteUserEnterRoom = (String roomId, TUIUserInfo userInfo)
```

Parameter	Type	Description
roomId	String	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User information

## onRemoteUserLeaveRoom

Remote user leaves the room event.

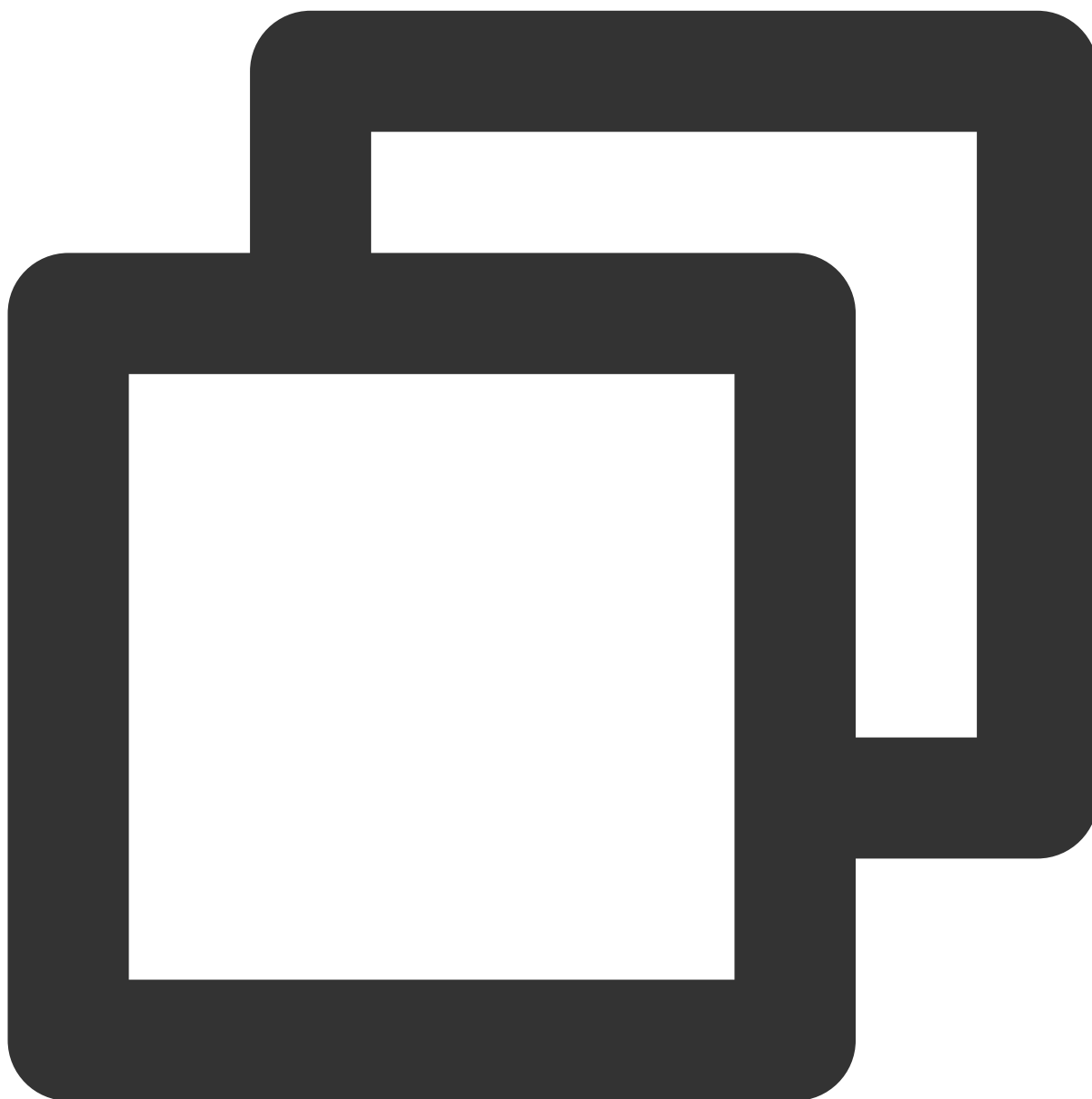


```
OnRemoteUserLeaveRoom onRemoteUserLeaveRoom = (String roomId, TUIUserInfo userInfo)
```

Parameter	Type	Description
roomId	String	Room ID
userInfo	<a href="#">TUIUserInfo</a>	User information

## onUserRoleChanged

User role changes event.

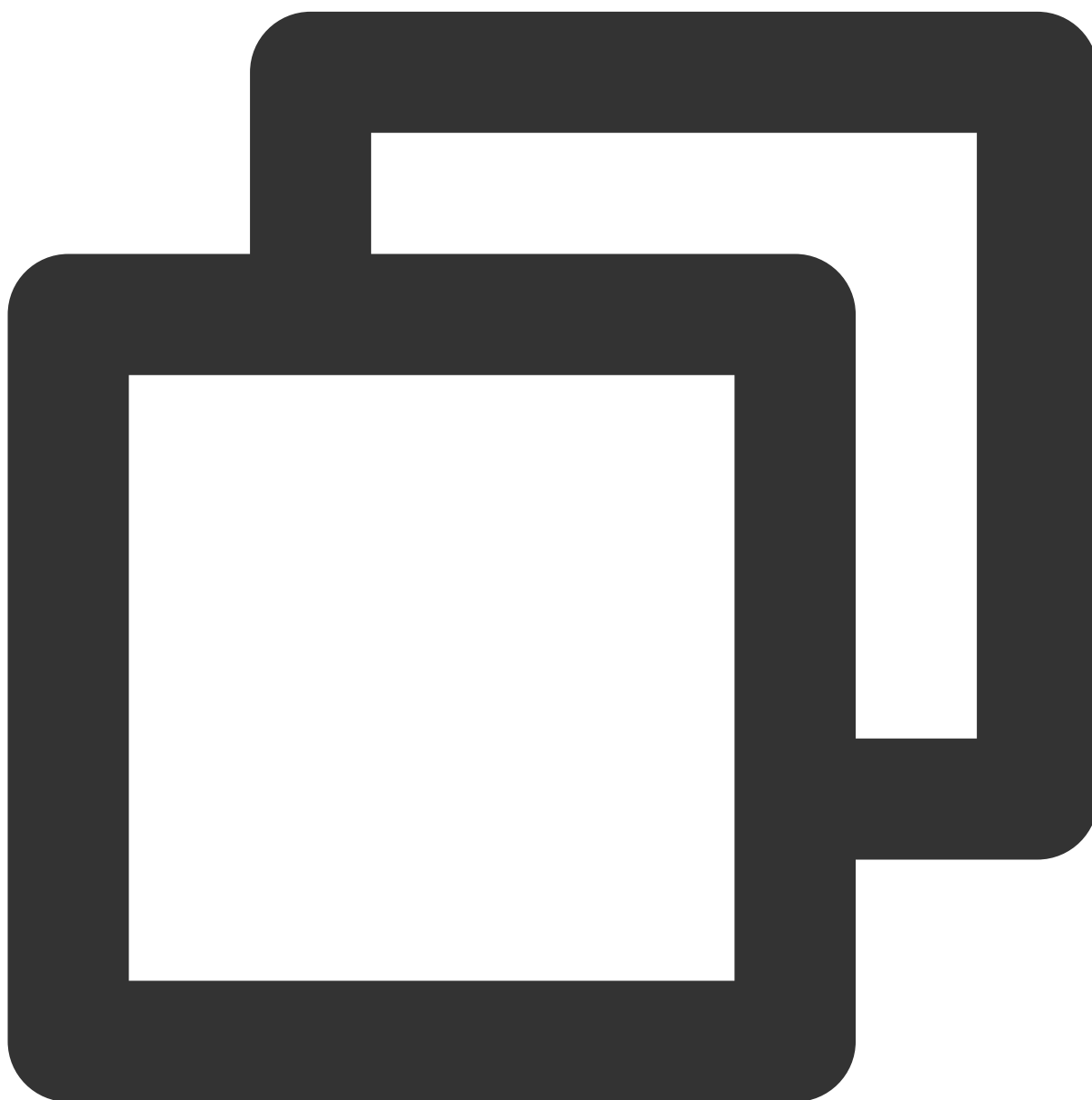


```
OnUserRoleChanged onUserRoleChanged = (String userId, TUIRole role) {}
```

Parameter	Type	Description
userId	String	User ID
role	<a href="#">TUIRole</a>	User Role

## onUserVideoStateChanged

User Video status changes event.



```
OnUserVideoStateChanged onUserVideoStateChanged = (String userId, TUIVideoStreamTyp
```

Parameter	Type	Description
userId	String	User ID
streamType	<a href="#">TUIVideoStreamType</a>	Streams type
hasVideo	bool	Whether there are streams
reason	<a href="#">TUIChangeReason</a>	Reason for streams change

## onUserAudioStateChanged

User Audio status changes event.

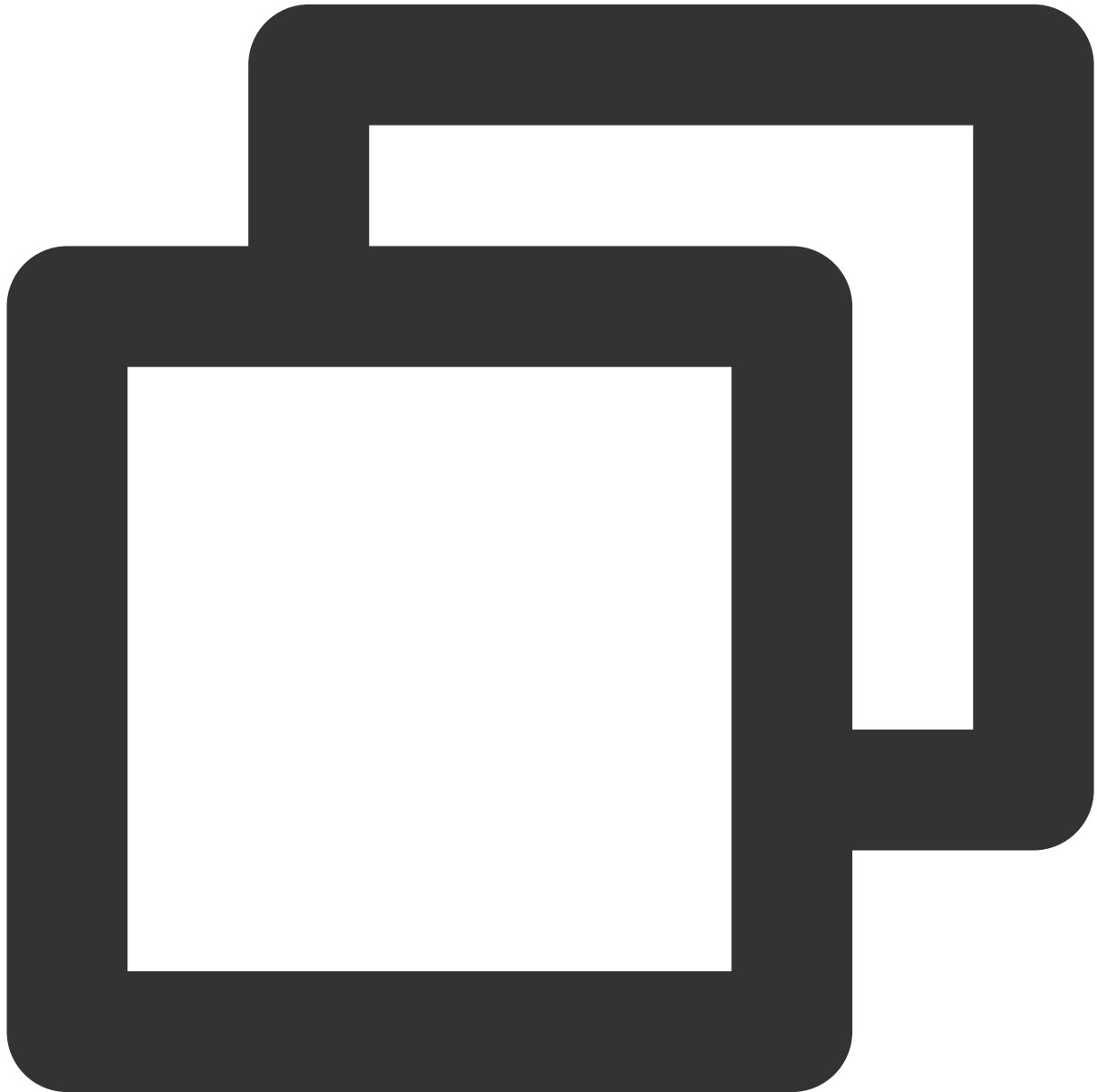


```
OnUserAudioStateChanged onUserAudioStateChanged = (String userId, bool hasAudio, TU
```

Parameter	Type	Description
userId	String	User ID
hasAudio	bool	Whether there are Audio streams
reason	<a href="#">TUIChangeReason</a>	Reason for Audio streams change

## onUserVoiceVolumeChanged

User volume change event.

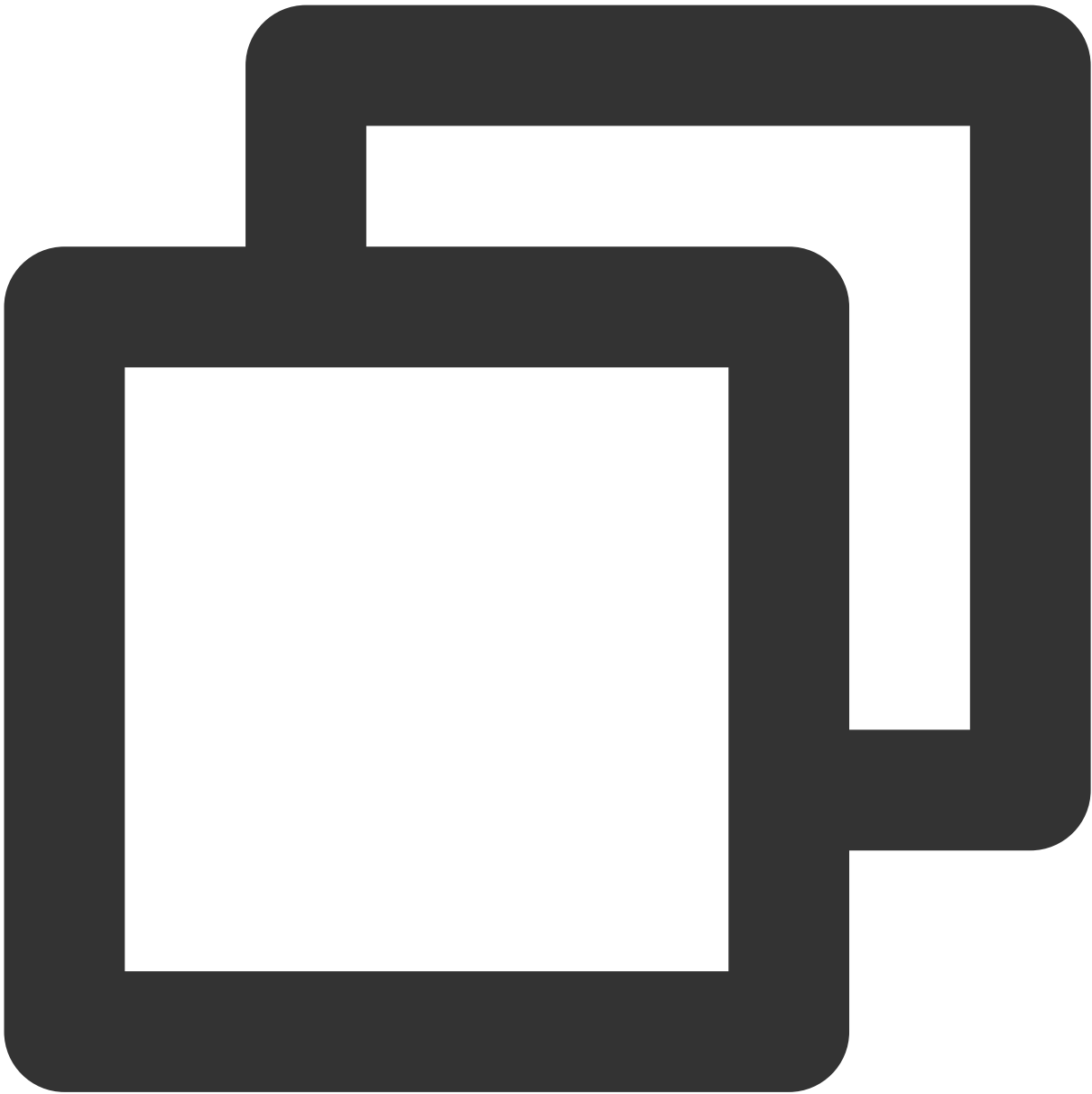


```
OnUserVoiceVolumeChanged onUserVoiceVolumeChanged = (Map<String, int> volumeMap) {}
```

Parameter	Type	Description
volumeMap	Map	User Volume Map key: userId value: Used for carrying the volume size of all speaking users, Value range 0 - 100

onSendMessageForUserDisableChanged

User text message sending ability changes event.



OnSendMessageForUserDisableChanged onSendMessageForUserDisableChanged = (String roomId, Boolean isSendMessageDisabled)

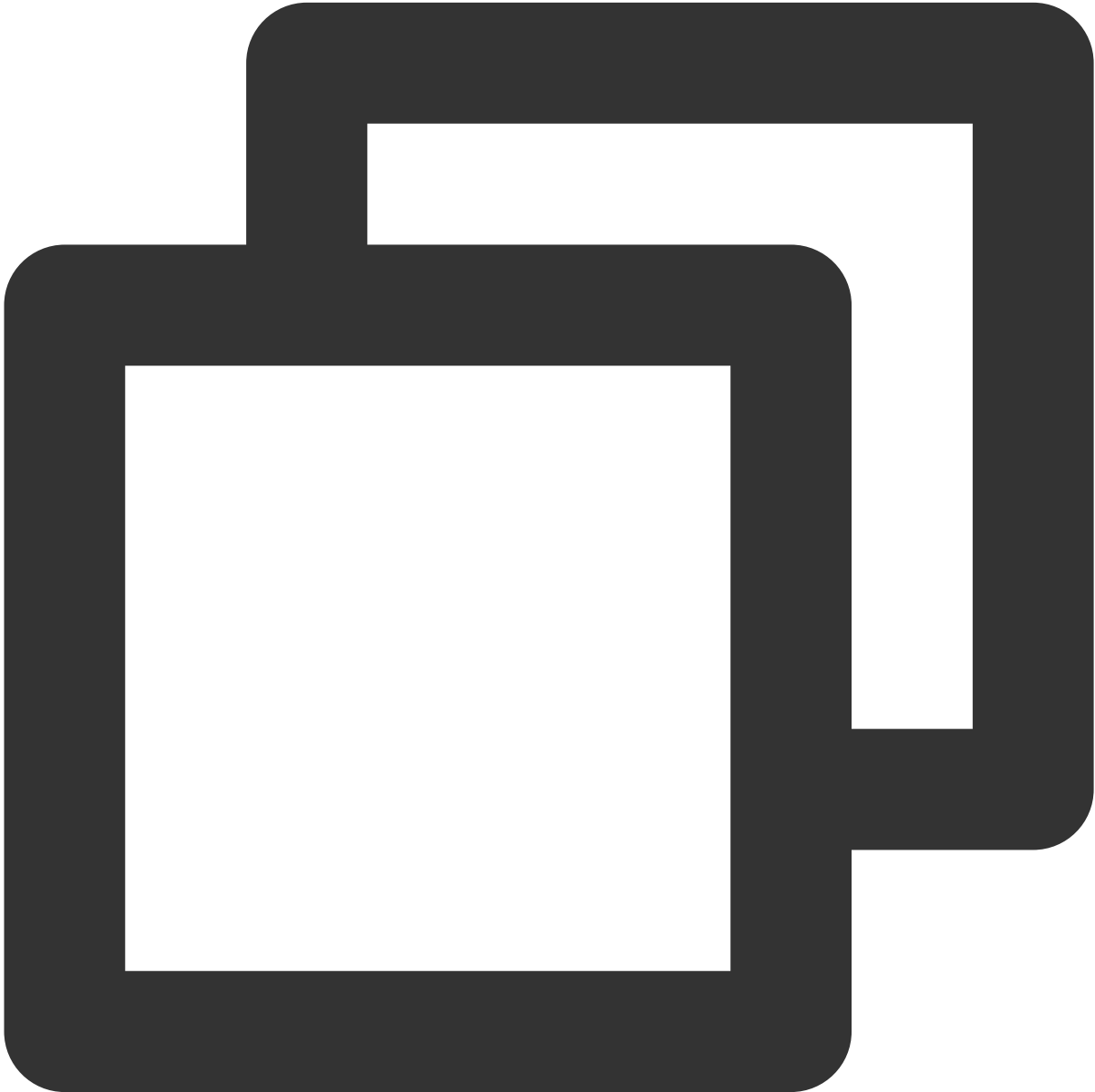
Parameter	Type	Description
roomId	String	Room ID



userId	String	User ID
isDisable	bool	Whether it is prohibited to send text messages.

onUserNetworkQualityChanged

User network status change event.



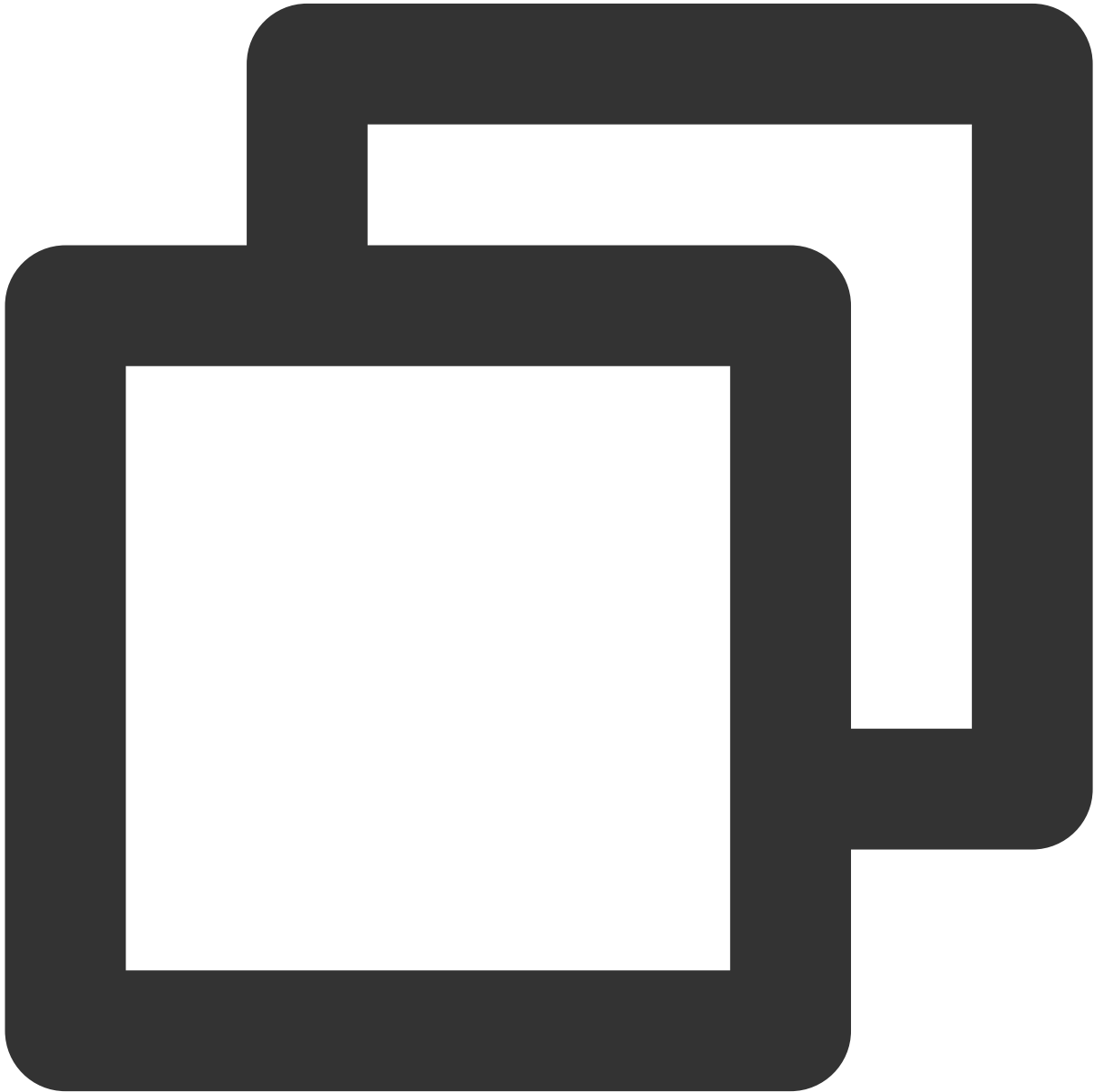
OnUserNetworkQualityChanged onUserNetworkQualityChanged = (Map<String, TUINetwork>

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

networkMap	Map	User Network Status Map key: userId value: Network Condition
------------	-----	--

**onUserScreenCaptureStopped**

Screen Sharing stopped Callback event.



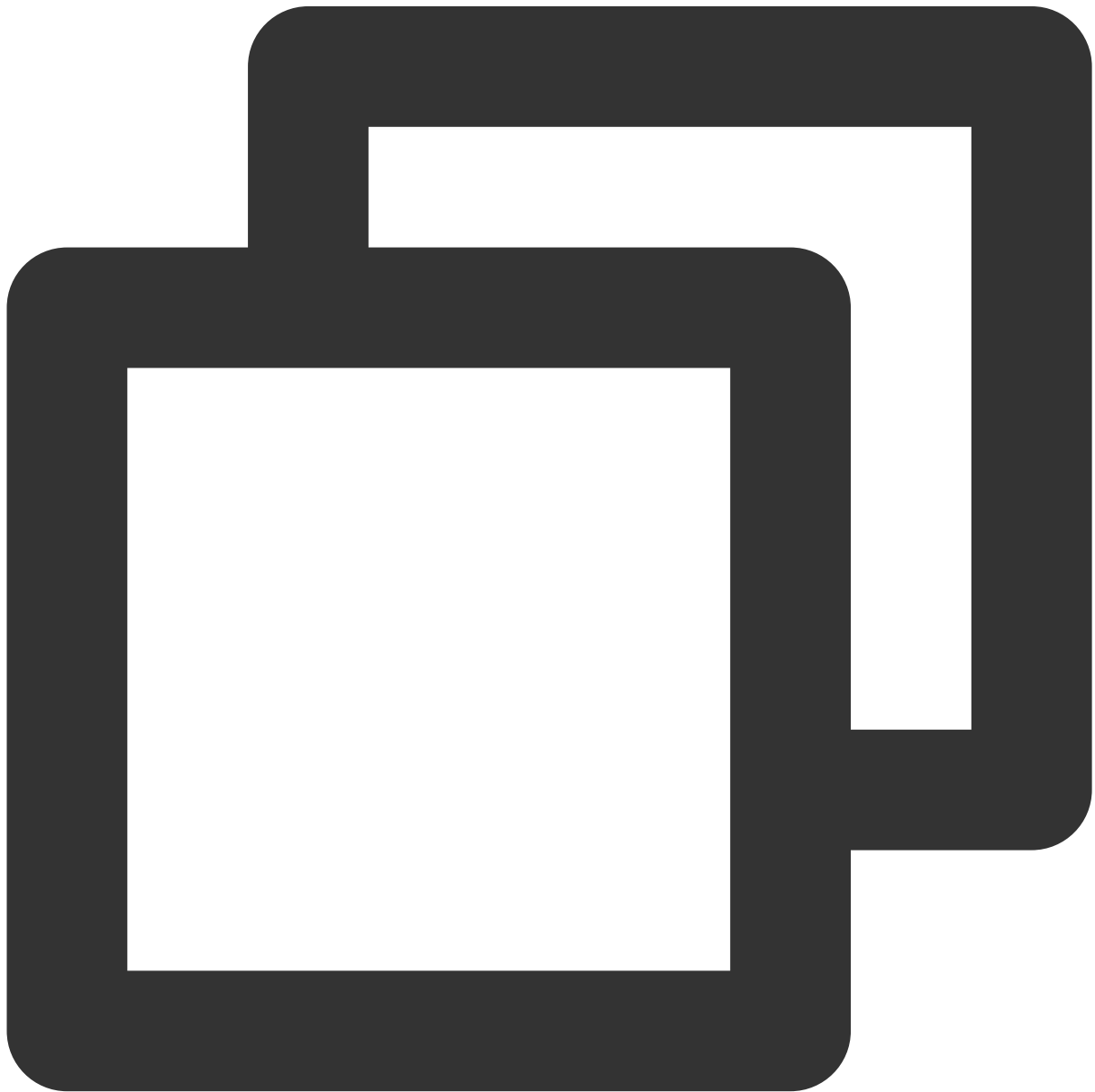
```
OnUserScreenCaptureStopped onUserScreenCaptureStopped = (int reason) {}
```

--	--	--

Parameter	Type	Description
reason	int	Stop reason: 0: User actively stops 1: Screen window closing causes the stop 2: Screen Sharing display screen status change (such as interface being unplugged, Projection mode change, etc.)

### **onRoomMaxSeatCountChanged**

Maximum number of mic slots changes event in the room (only effective in meeting type rooms).

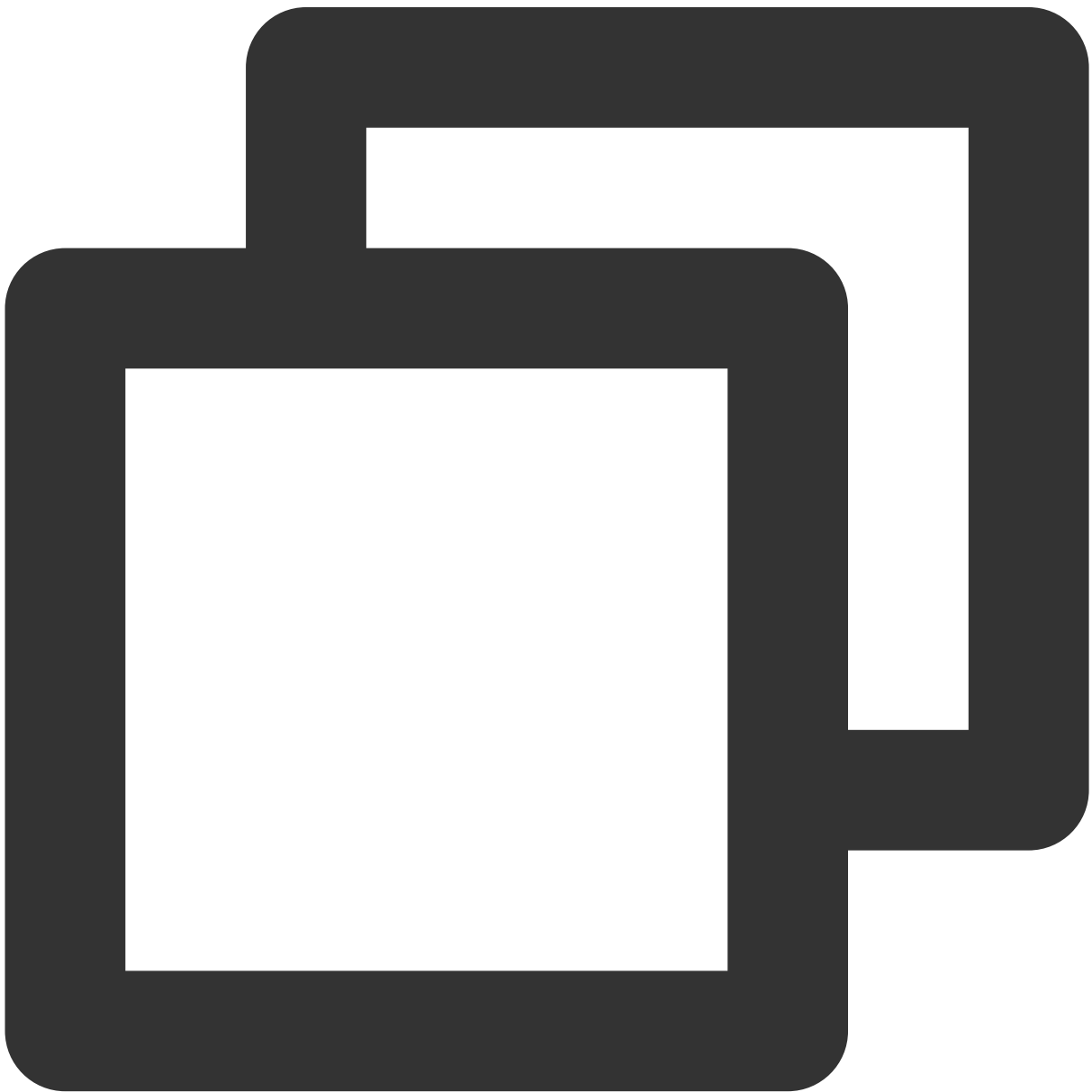


OnRoomMaxSeatCountChanged onRoomMaxSeatCountChanged = (String roomId, int maxSeatCo

Parameter	Type	Description
roomId	String	Room ID
maxSeatCount	int	Maximum number of mic slots in the room

onSeatListChanged

Mic slot list changes event.



```
OnSeatListChanged onSeatListChanged = (List<TUISeatInfo> seatList, List<TUISeatInfo>
```

Parameter	Type	Description
seatList	List< <a href="#">TUISeatInfo</a> >	The latest user list on the mic, including newly on mic users
seatedList	List< <a href="#">TUISeatInfo</a> >	Newly on mic user list
leftList	List< <a href="#">TUISeatInfo</a> >	Newly off mic user list

## onKickedOffSeat

Received the event of user being kicked off mic.



```
OnKickedOffSeat onKickedOffSeat = (String userId) {}
```

Parameter	Type	Description
userId	String	Operate Kick-out of the (Host/Administrator) User ID

## onRequestReceived

Received request message event.



```
OnRequestReceived onRequestReceived = (TUIRequest request) {}
```

Parameter	Type	Description
request	<a href="#">TUIRequest</a>	Request content

### **onRequestCancelled**

Received request cancellation event.



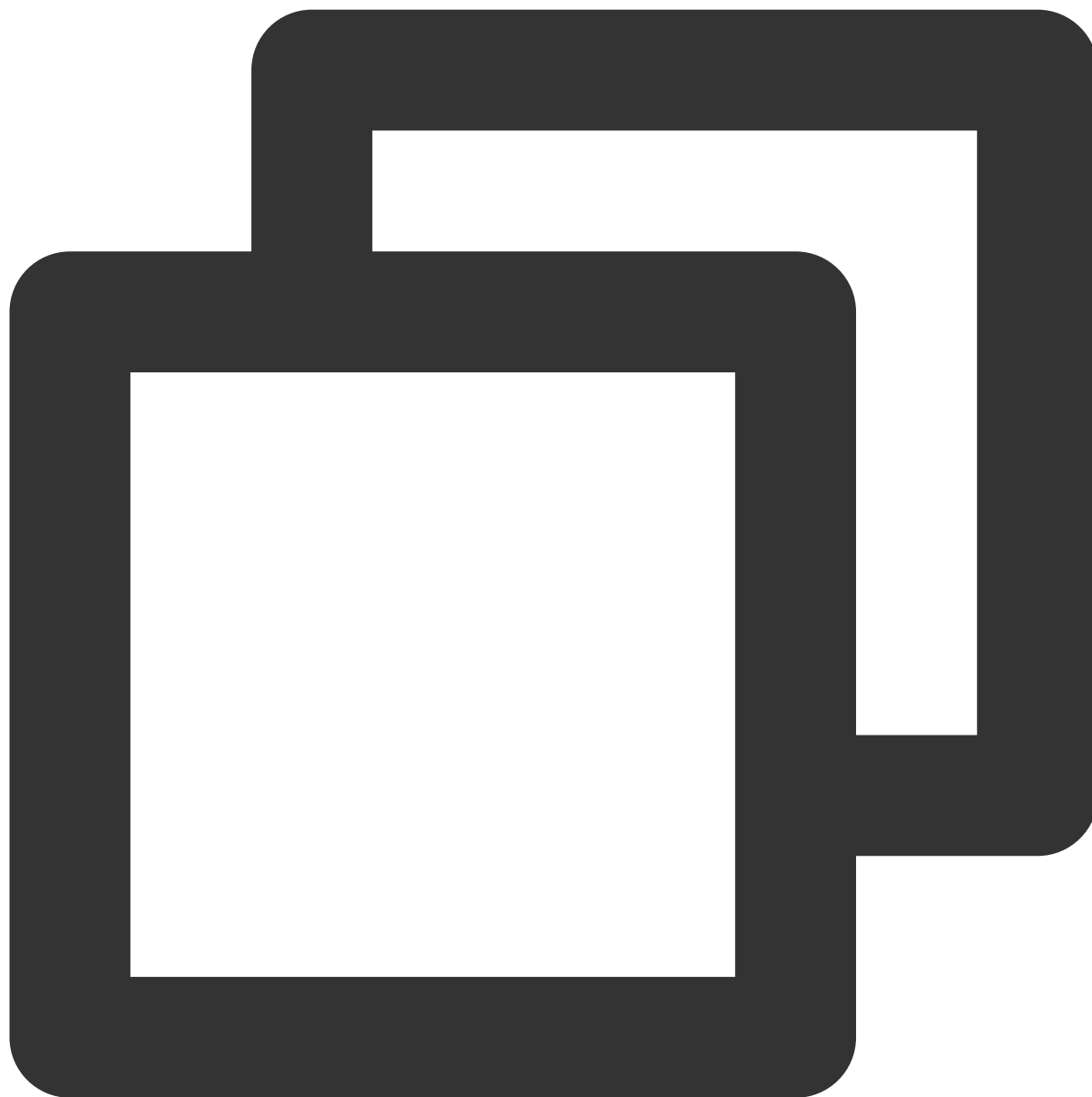
```
OnRequestCancelled onRequestCancelled = (String requestId, String userId) {}
```

Parameter	Type	Description
requestId	String	Request ID
userId	String	Cancel signaling user ID

## onReceiveTextMessage



Received ordinary text message event.

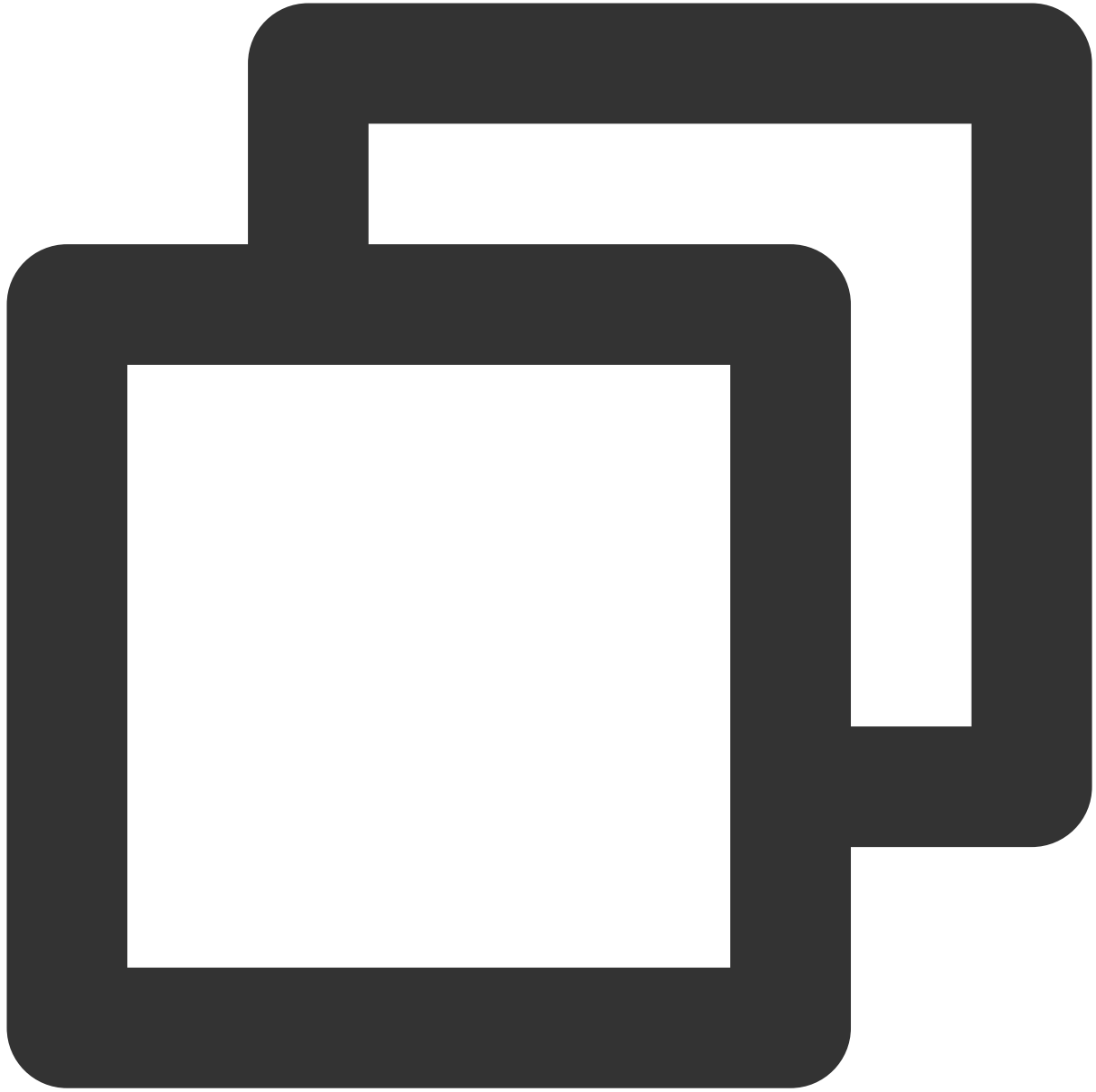


```
OnReceiveTextMessage onReceiveTextMessage = (String roomId, TUIMessage message) {}
```

Parameter	Type	Description
roomId	String	Room ID
message	<a href="#">TUIMessage</a>	Message content

### onReceiveCustomMessage

Received custom message event.



```
OnReceiveCustomMessage onReceiveCustomMessage = (String roomId, TUIMessage message)
```

Parameter	Type	Description
roomId	String	Room ID
message	<a href="#">TUIMessage</a>	Message content



# TUIRoomEngine

Last updated : 2023-11-21 15:00:32

## TUIRoomEngine API Introduction

TUIRoomEngine API is a No UI Interface for Conference Component, you can use this API to custom encapsulate according to your business needs.

### **createInstance**

Create TUIRoomEngine Instance.



```
static TUIRoomEngine createInstance()
```

**return:**TUIRoomEngine Instance

### **destroyInstance**

Destroy TUIRoomEngine Instance.



```
void destroyInstance()
```

## login

Login interface, you need to initialize user information before entering the room and perform a series of operations.



```
static Future<TUIActionCallback> login(int sdkAppId,  
                                       String userId,  
                                       String userSig)
```

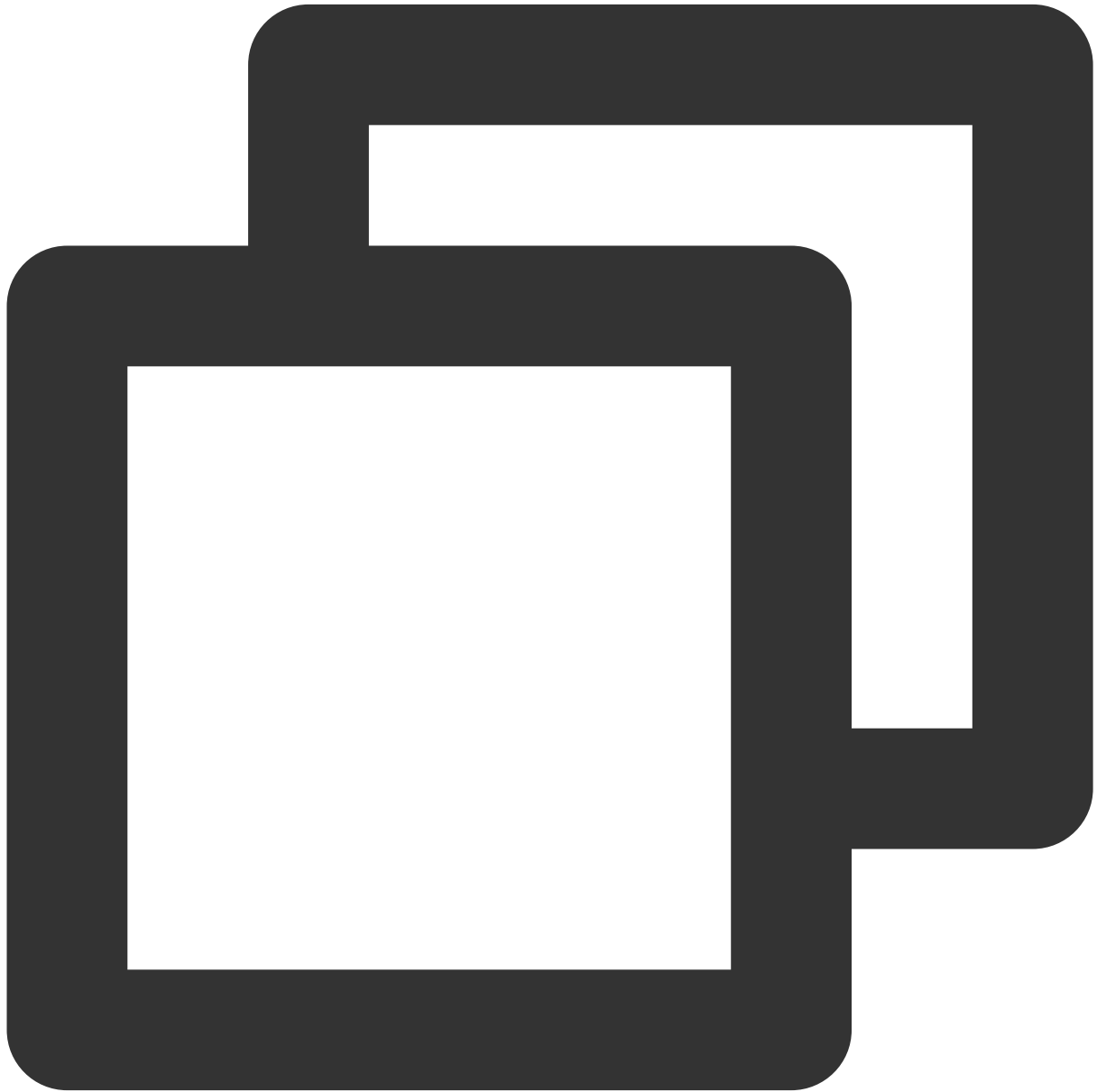
**Parameters:**

Parameter	Type	Meaning
sdkAppId	int	Get sdkAppId information from Application Info
userId	String	User ID

userSig	String	UserSig
---------	--------	---------

## logout

Logout interface, there will be actively leave the room operation, destroy resources.

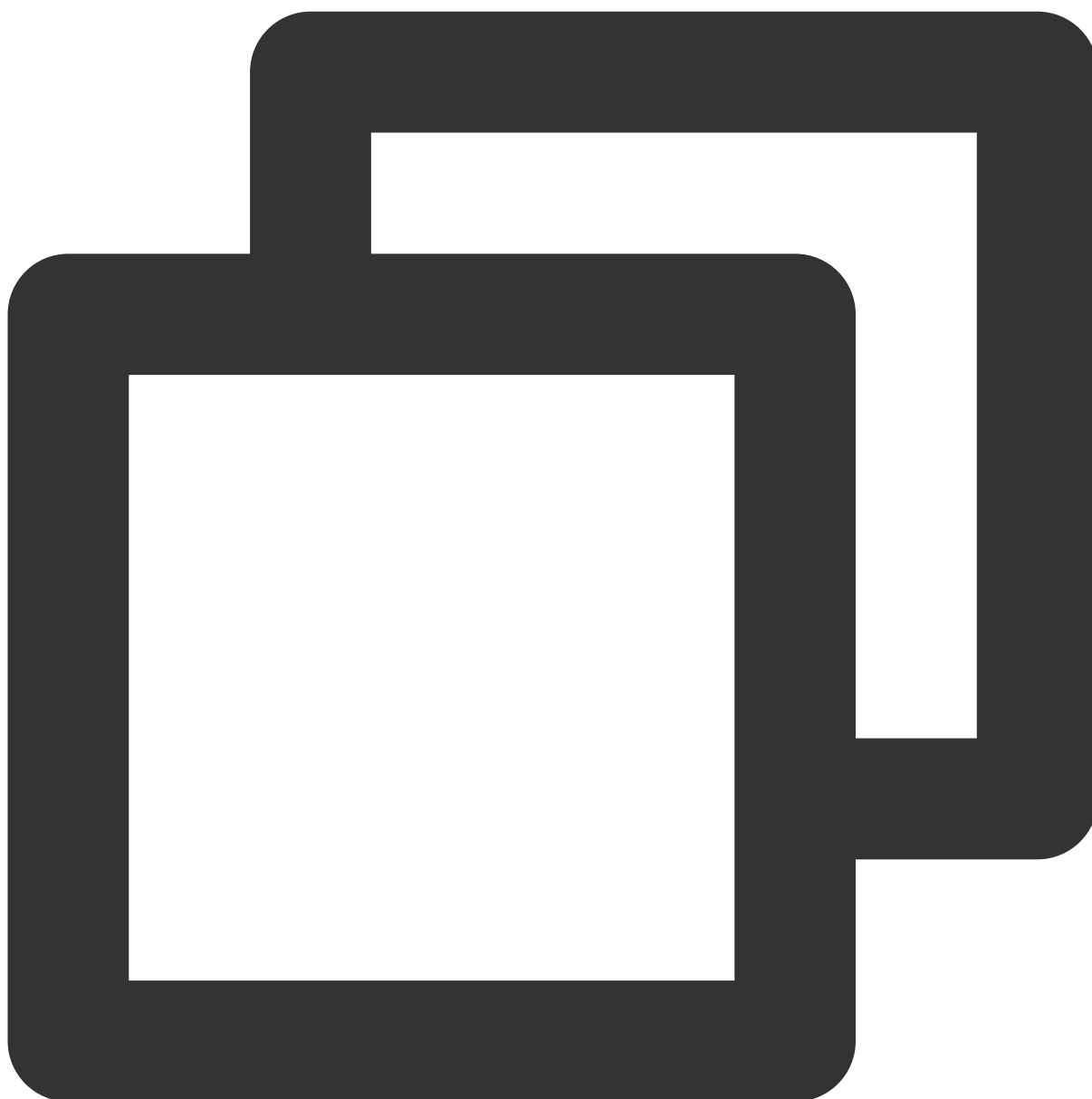


```
static Future<TUIActionCallback> logout ()
```

## setSelfInfo

Set local user name and avatar.





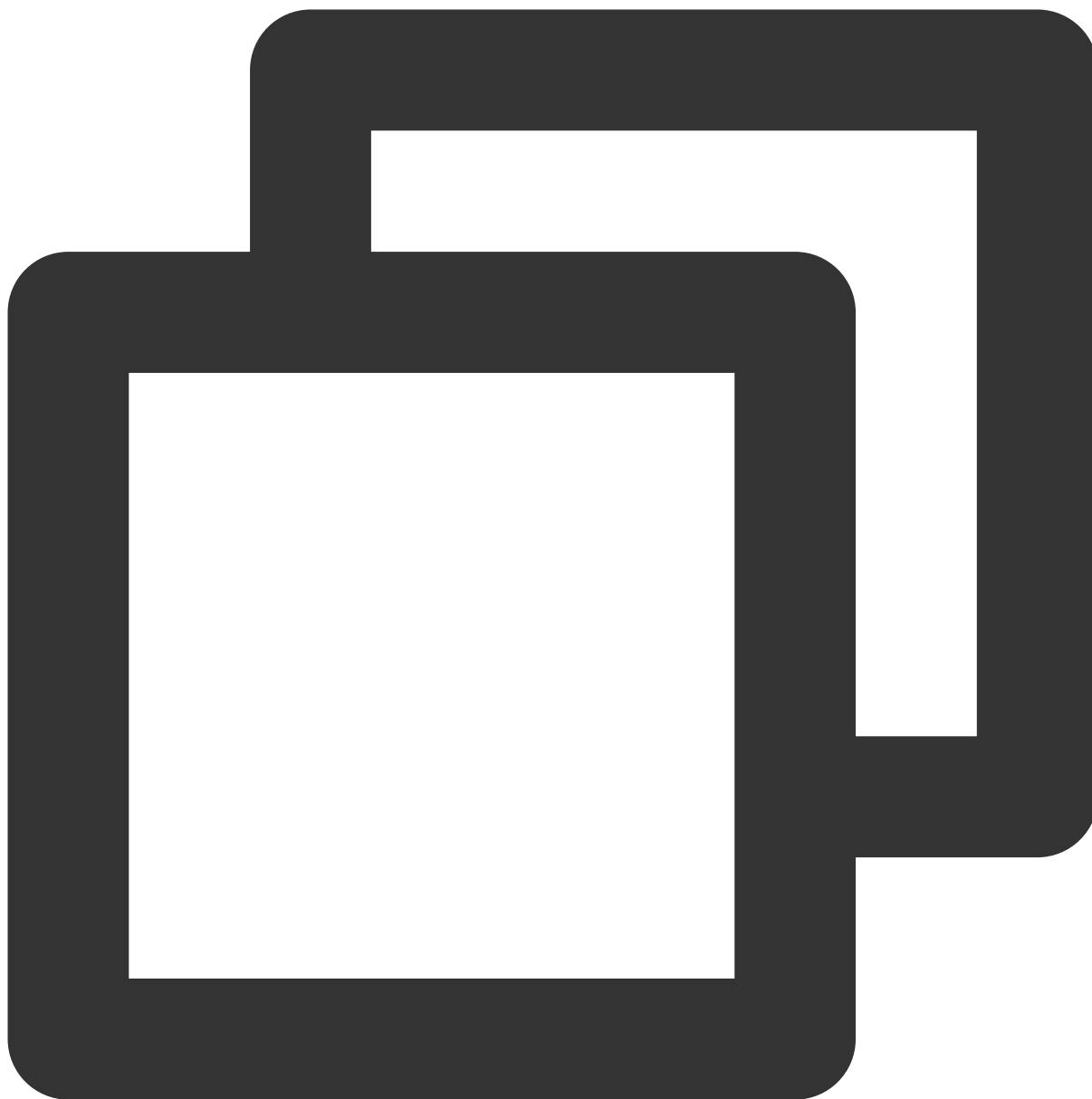
```
static Future<TUIActionCallback> setSelfInfo(String userName, String avatarURL)
```

**Parameters:**

Parameter	Type	Meaning
userName	String	User Name
avatarUrl	String	User avatar URL address

**setLoginUserInfo**

Set login user information.



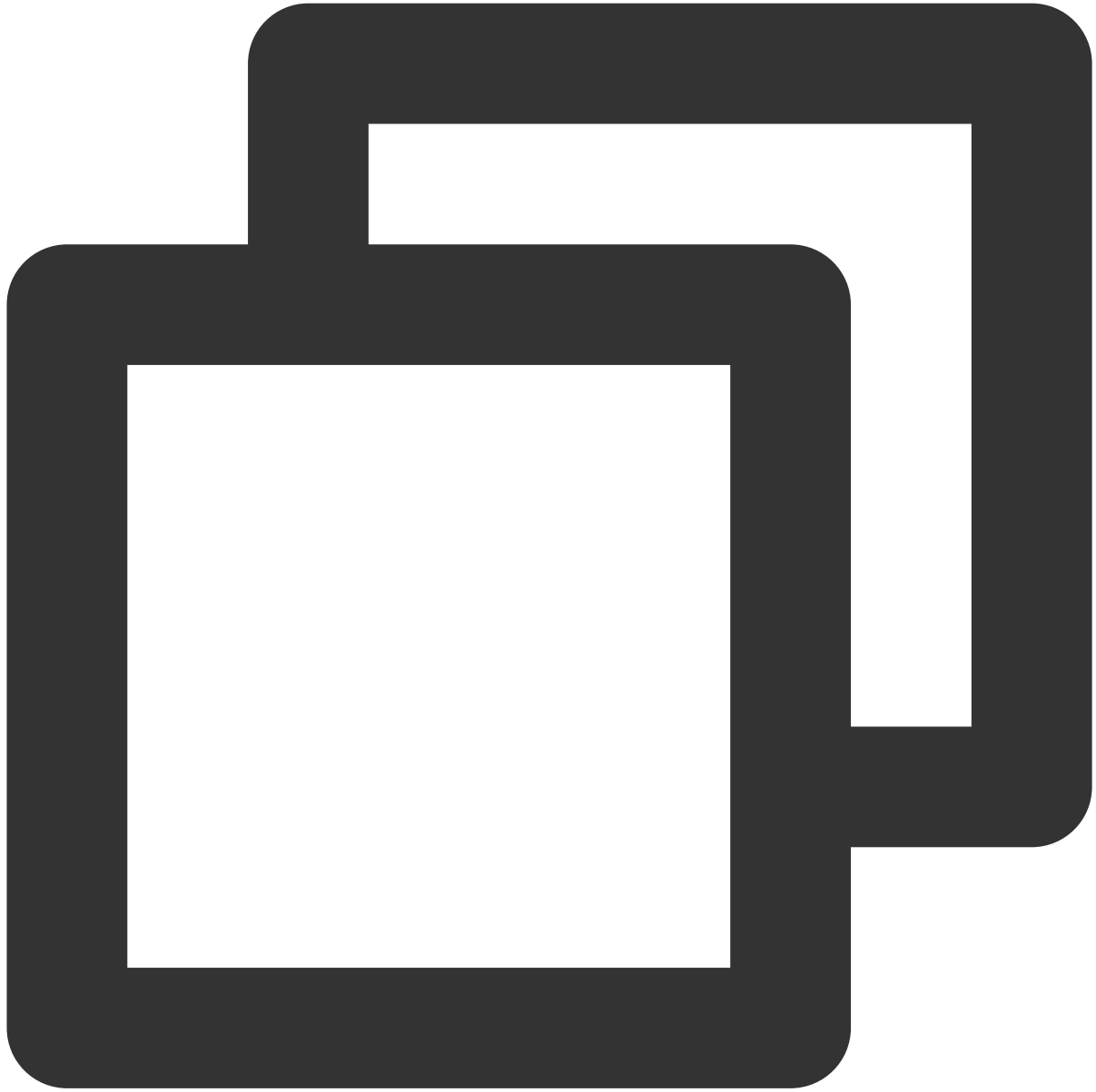
```
static Future<TUIActionCallback> setLoginUserInfo(TUILoginUserInfo userInfo)
```

**Parameters:**

Parameter	Type	Meaning
userInfo	TUILoginUserInfo	User information

**getSelfInfo**

Get the basic information of the local user login.



```
static TUILoginUserInfo getSelfInfo()
```

**return:**the basic information of the local user login

### **addObserver**

Add TUIRoomEngine Event Callback.



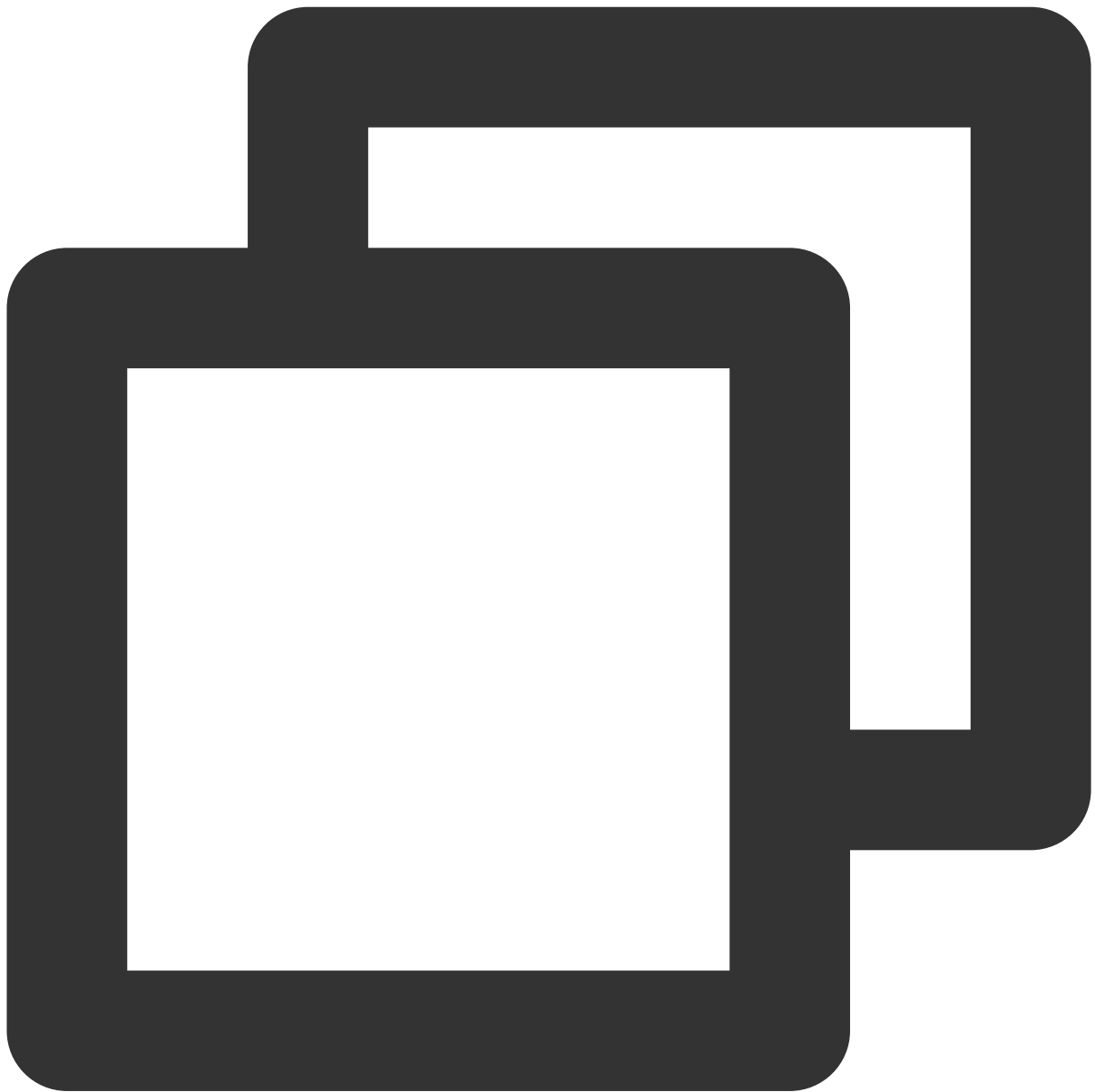
```
void addObserver(TUIRoomObserver observer)
```

**Parameters:**

Parameter	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine Event Callback

**removeObserver**

Remove TUIRoomEngine Event Callback.



```
void removeObserver(TUIRoomObserver observer)
```

Parameter	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine Event Callback

## createRoom

The host creates a room, and the user who calls createRoom is the owner of the room. When creating a room, you can set the Room ID, room name, and whether the room allows users to join, enable video and audio, send messages, and

other functions.



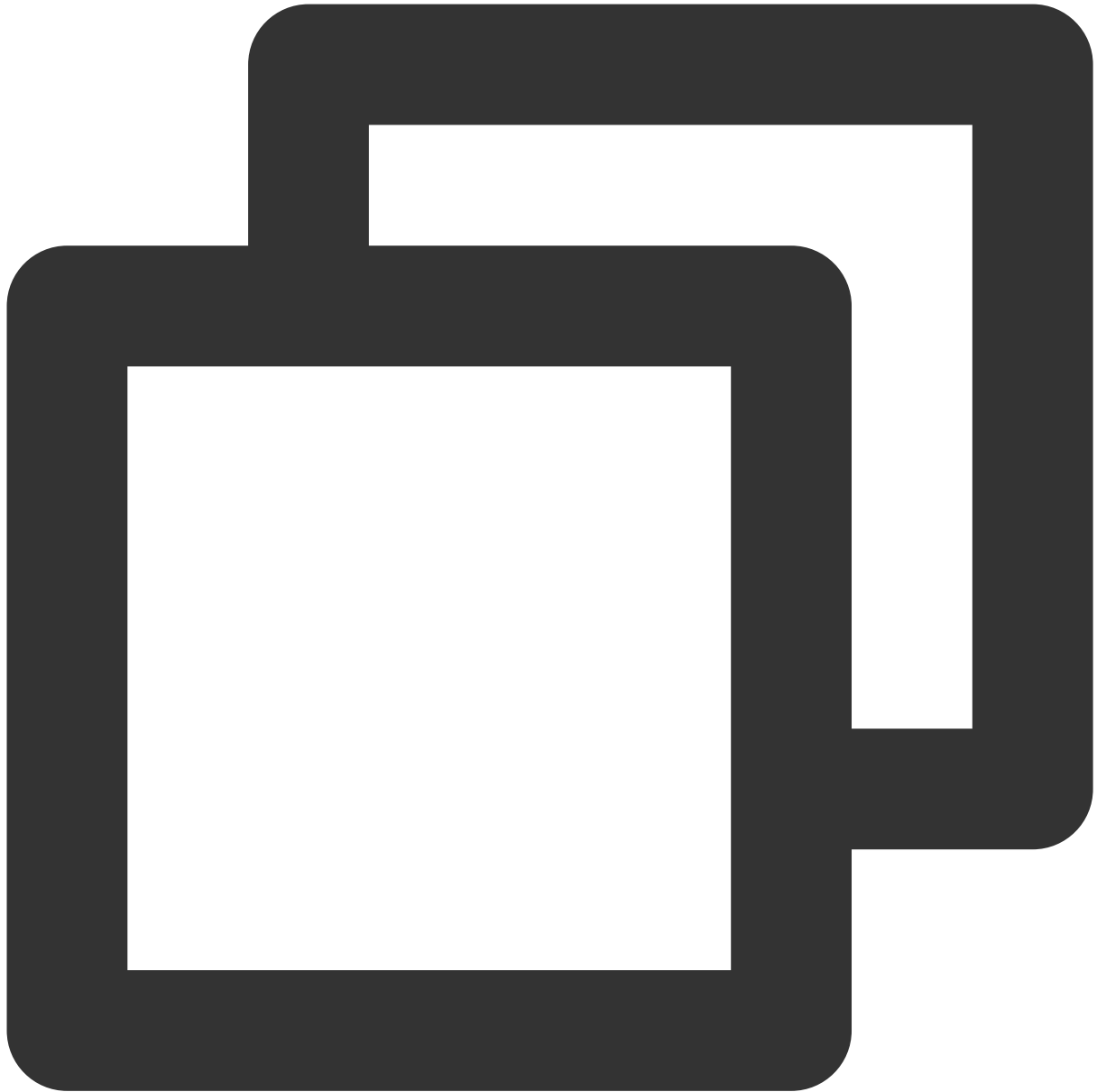
```
Future<TUIActionCallback> createRoom(TUIRoomInfo roomInfo)
```

#### Parameters:

Parameter	Type	Meaning
roomInfo	<a href="#">TUIRoomInfo</a>	Room data

#### destroyRoom

Destroy Room Interface, the room owner must initiate the destruction of the room. After the room is destroyed, it is unavailable for entry.



```
Future<TUIActionCallback> destroyRoom()
```

## enterRoom

Entered room.



```
Future<TUIValueCallBack<TUIRoomInfo>> enterRoom(String roomId)
```

**Parameters:**

Parameter	Type	Meaning
roomId	String	Room ID

**exitRoom**

Exit Room Interface, after the user executes Enter Room, they can leave the room through Leave Room.





```
Future<TUIActionCallback> exitRoom(bool syncWaiting)
```

**Parameters:**

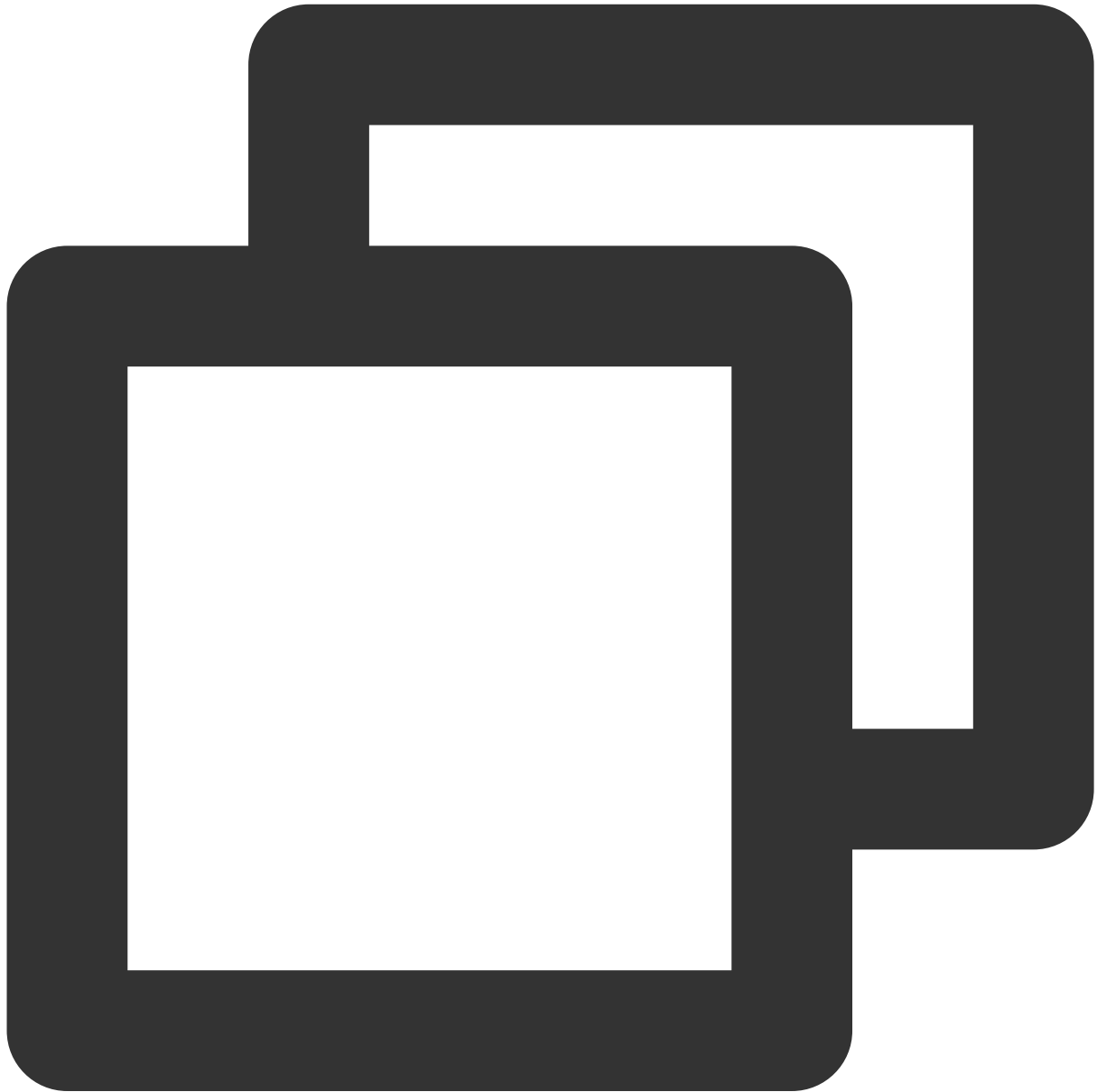
Parameter	Type	Meaning
syncWaiting	bool	Whether to synchronize leaving the room

**connectOtherRoom**

Connect to other rooms.

**Description:**

Used for live streaming scenario to apply for cross-room streaming



```
TUIRequest connectOtherRoom(String roomId,  
                             String userId,  
                             int timeout,  
                             TUIRequestCallback? requestCallback)
```

**Parameters:**

Parameter	Type	Meaning

roomId	String	Room ID
userId	String	User ID
timeout	int	Time
callback	TUIRequestCallback	Connect to other rooms Callback

**Return :** Request body

### **disconnectOtherRoom**

Disconnect from other rooms

**Description:**

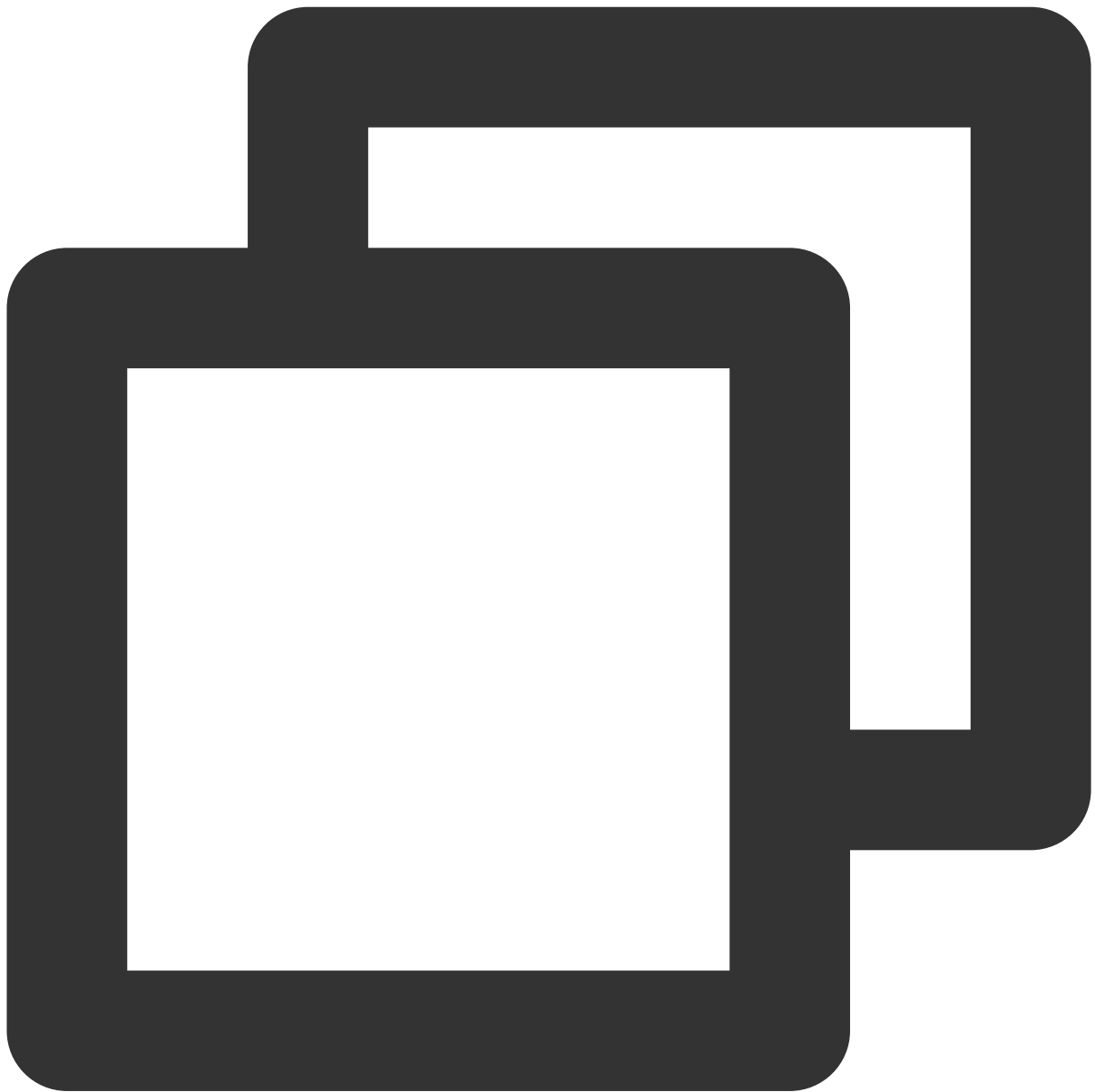
Used for live streaming scenario to disconnect cross-room streaming



```
Future<TUIActionCallback> disconnectOtherRoom()
```

### **fetchRoomInfo**

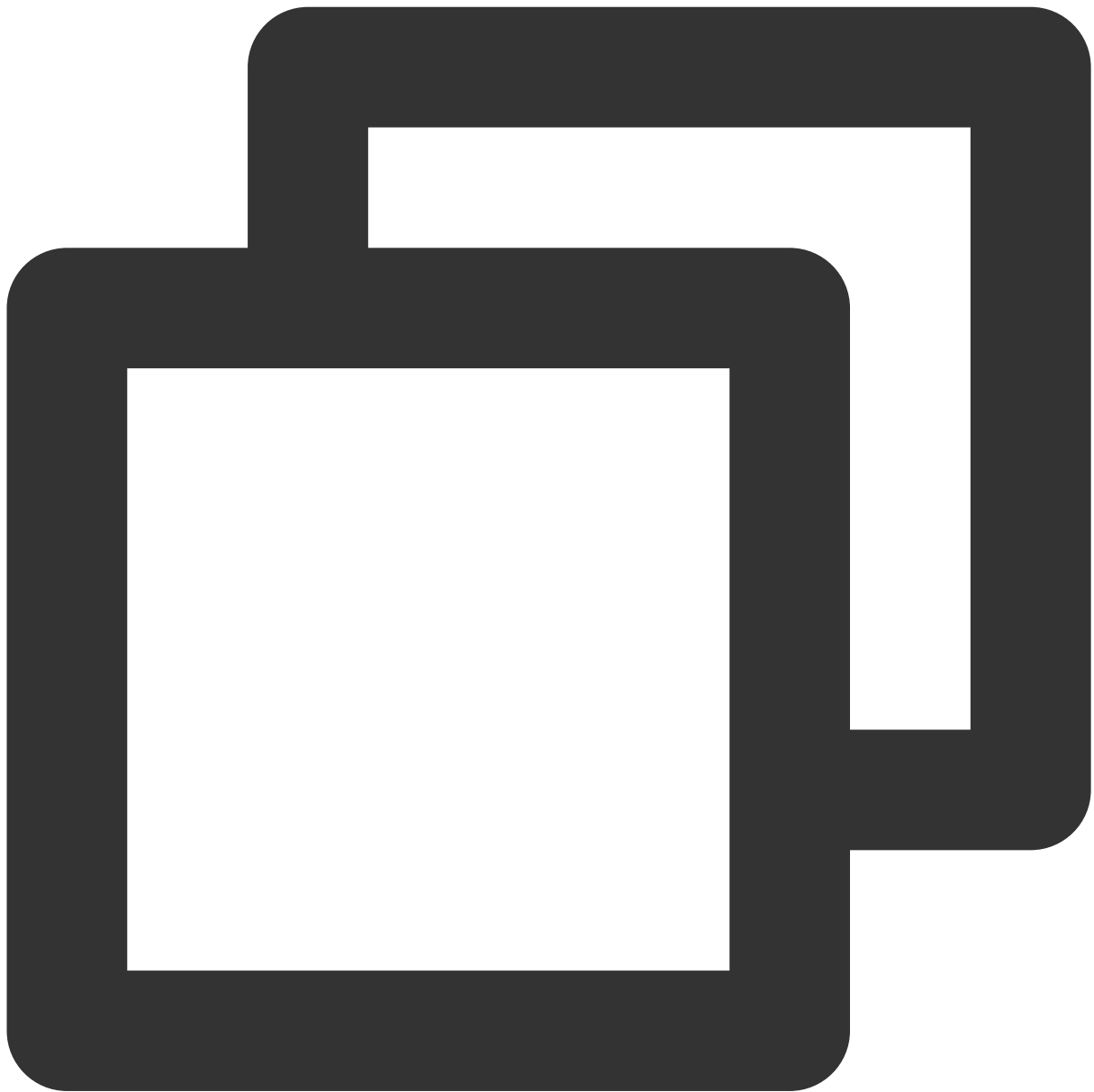
Get Room data.



```
Future<TUIValueCallBack<TUIRoomInfo>> fetchRoomInfo()
```

### **updateRoomNameByAdmin**

Update Room ID.



```
Future<TUIActionCallback> updateRoomNameByAdmin(String roomName)
```

**Parameters:**

Parameter	Type	Meaning
roomName	String	Room ID

**updateRoomSpeechModeByAdmin**

Set Management mode (only Administrator or Group owner can call).



```
Future<TUIActionCallback> updateRoomSpeechModeByAdmin(TUISpeechMode mode)
```

Parameter	Type	Meaning
mode	<a href="#">TUISpeechMode</a>	Management mode

## setLocalVideoView

Set Local user Video Rendering View control.



```
void setLocalVideoView(TUIVideoStreamType streamType,  
                       int viewId)
```

**Parameters:**

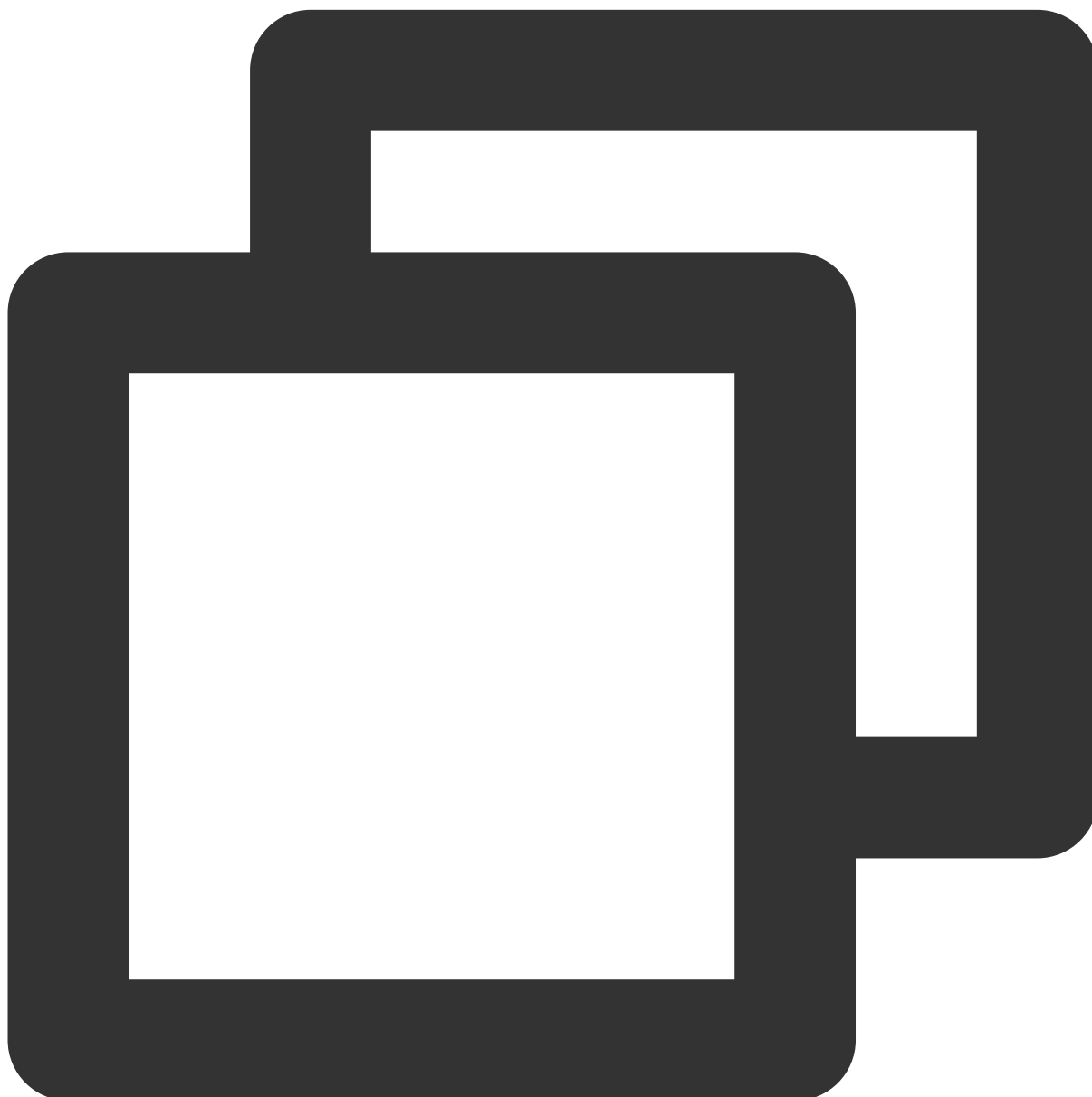
Parameter	Type	Meaning
streamType	<a href="#">TUIVideoStreamType</a>	Local streams type
viewId	int	The int64 type value of the pointer to the view to be rendered, through this viewId, can be converted to the



corresponding native platform view, and the video screen will be rendered on this view.

## openLocalCamera

Open Local Camera, Start Video Capturing.



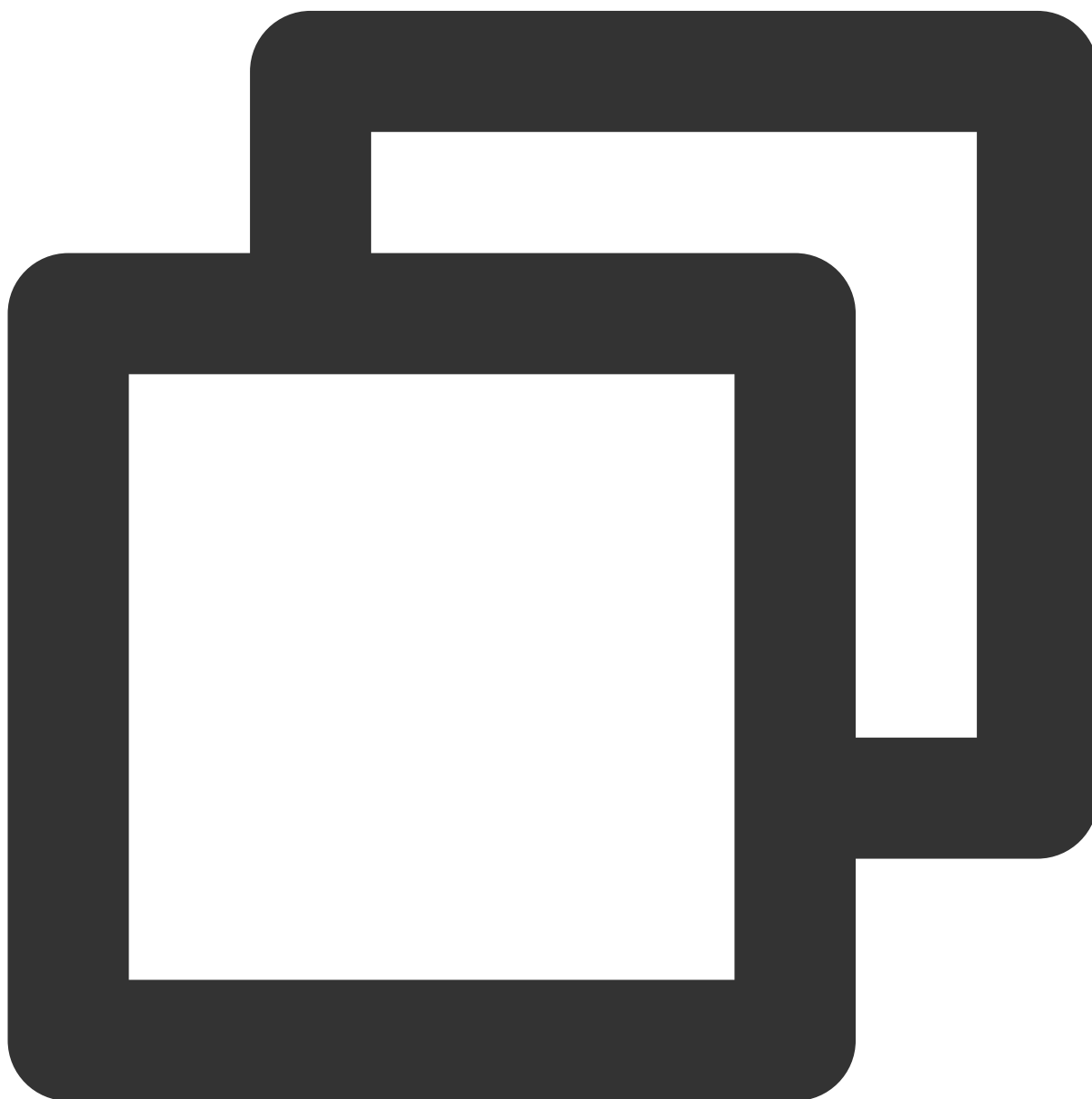
```
Future<TUIActionCallback> openLocalCamera(bool isFront,  
                                           TUIVideoQuality quality)
```

### Parameters:

Parameter	Type	Meaning
isFront	bool	Whether to use Front Camera
quality	<a href="#">TUIVideoQuality</a>	Video Quality

## closeLocalCamera

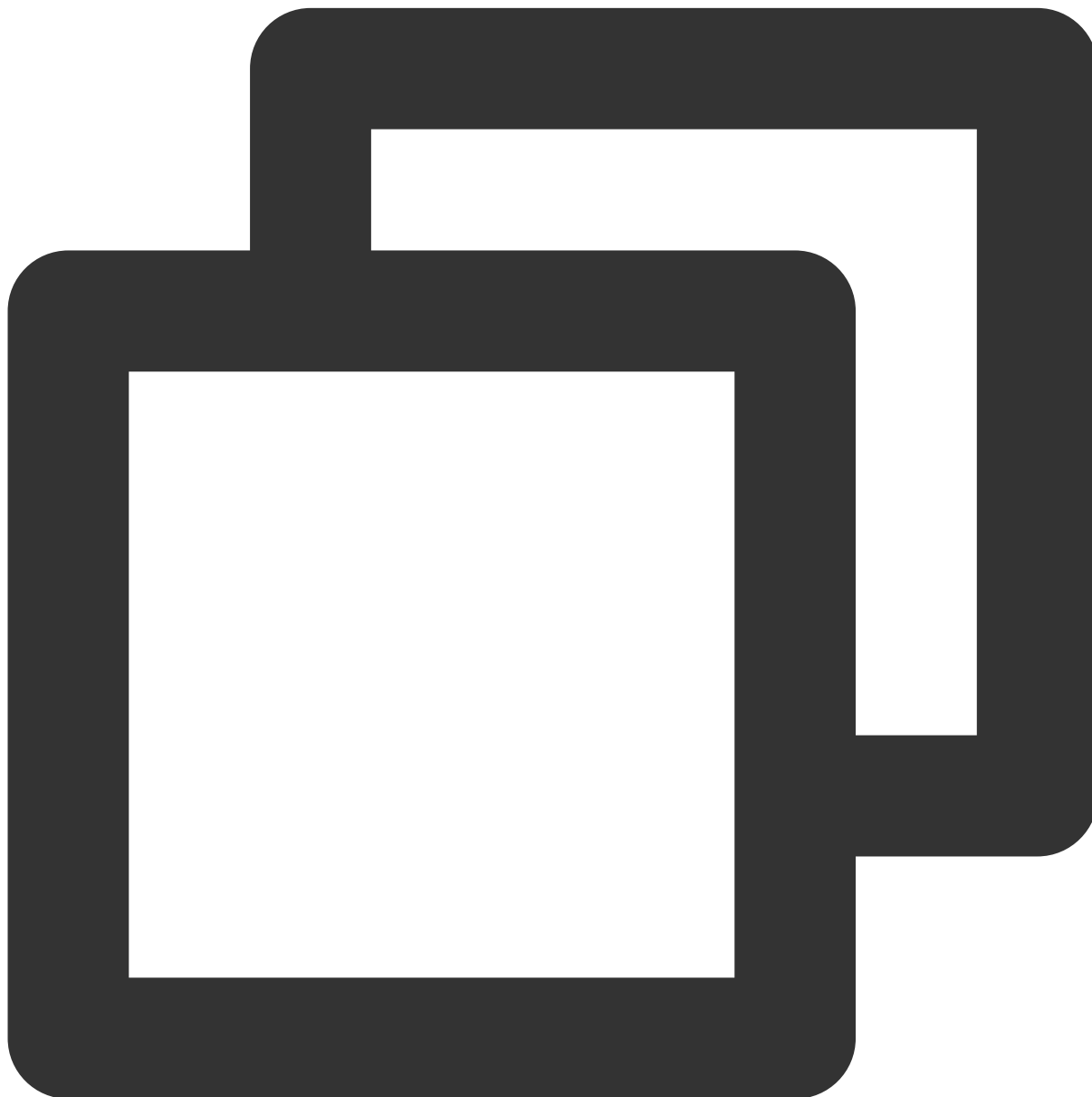
Close Local Camera.



```
void closeLocalCamera()
```

## updateVideoQuality

Set Local Video Parameter.



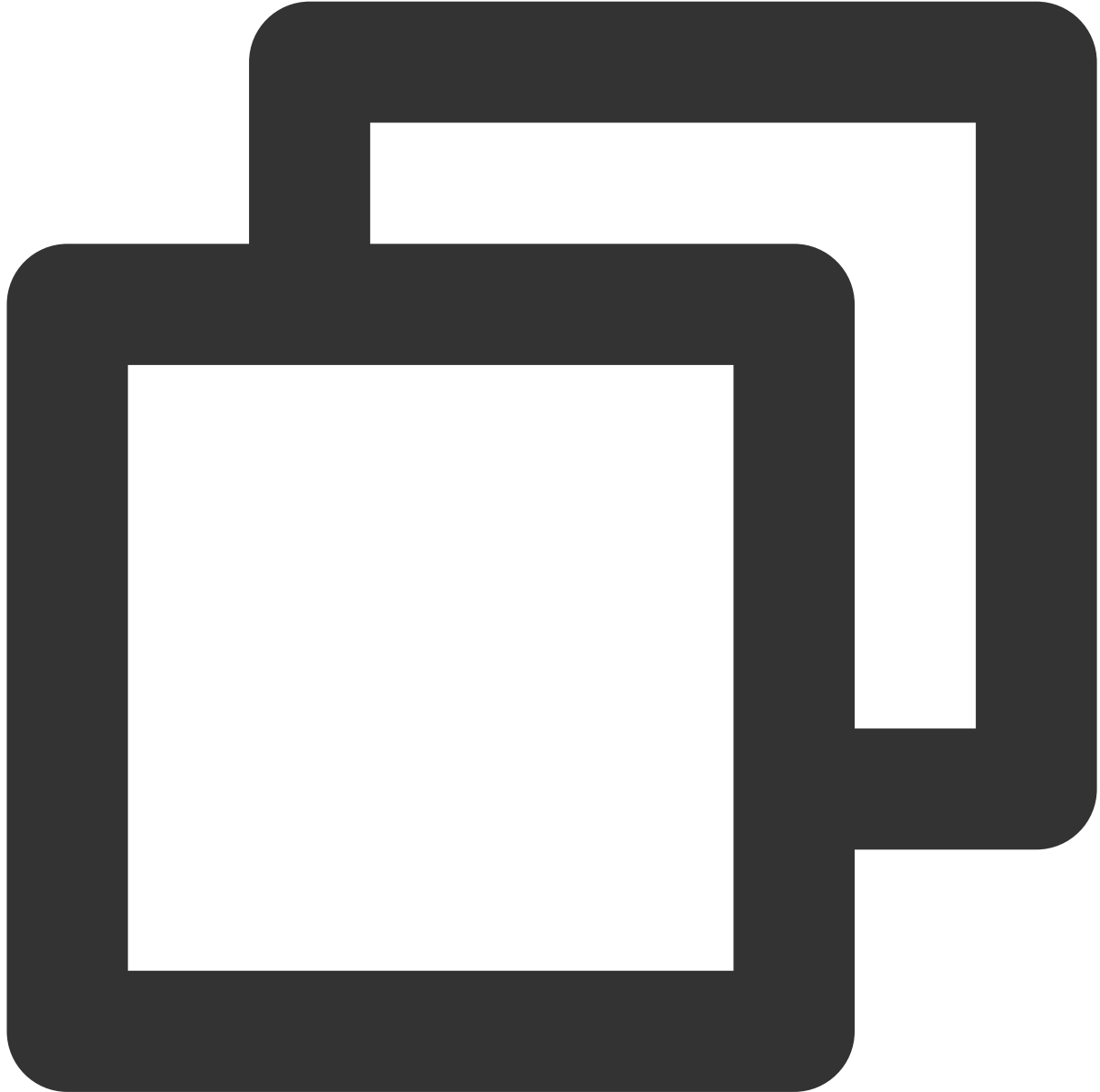
```
void updateVideoQuality(TUIVideoQuality quality)
```

### Parameters:

Parameter	Type	Meaning
quality	<a href="#">TUIVideoQuality</a>	Video Quality

## updateVideoQualityEx

Set the encoding parameters of video encoder.

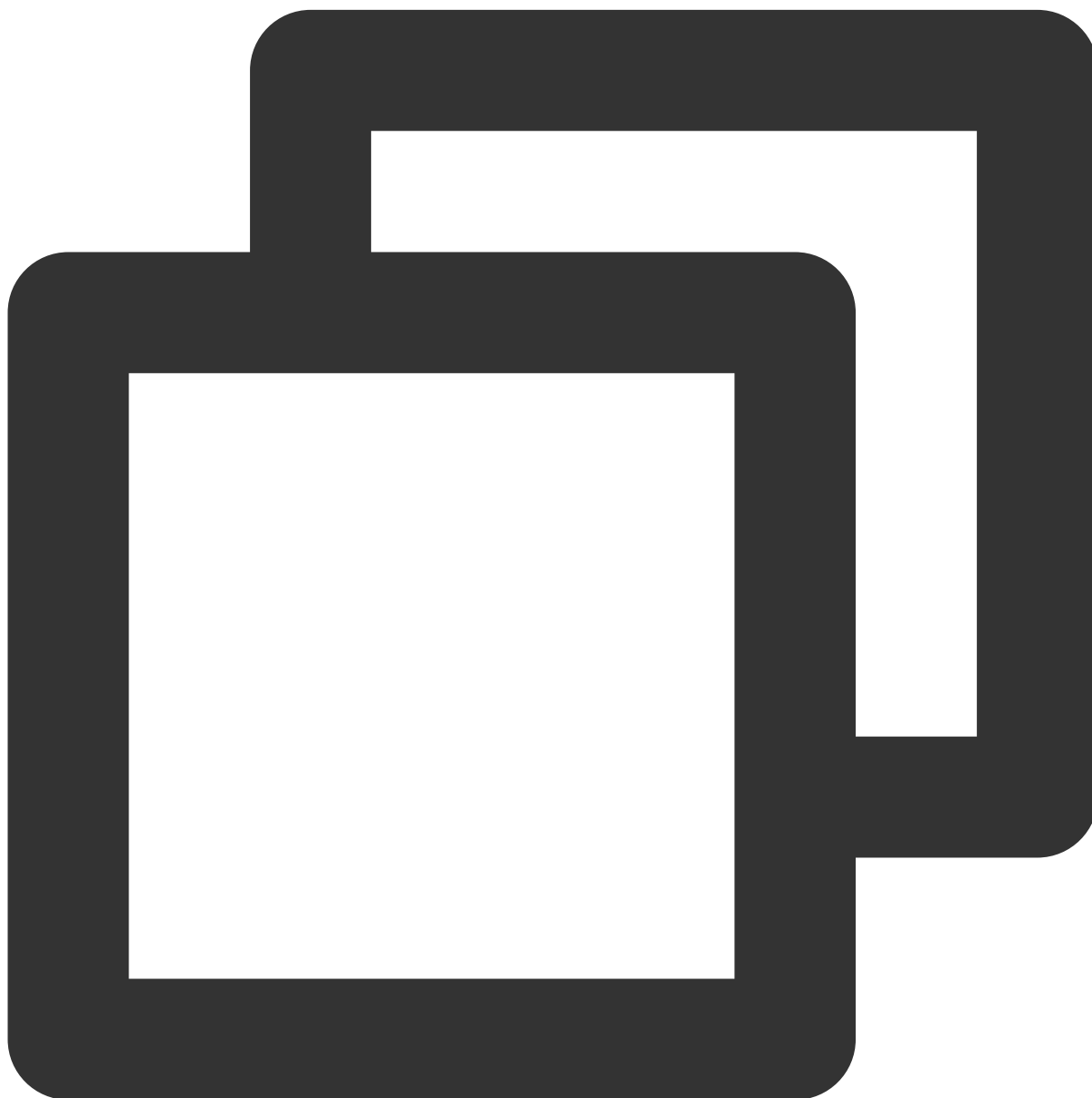


```
void updateVideoQualityEx(  
    TUIVideoStreamType streamType, TUIRoomVideoEncoderParams params);
```

Parameter	Type	Meaning
streamType	TUIVideoStreamType	Video Stream type
params	TUIRoomVideoEncoderParams	Encoding parameters of video encoder

## setVideoResolutionMode

Set the resolution mode of video encoder



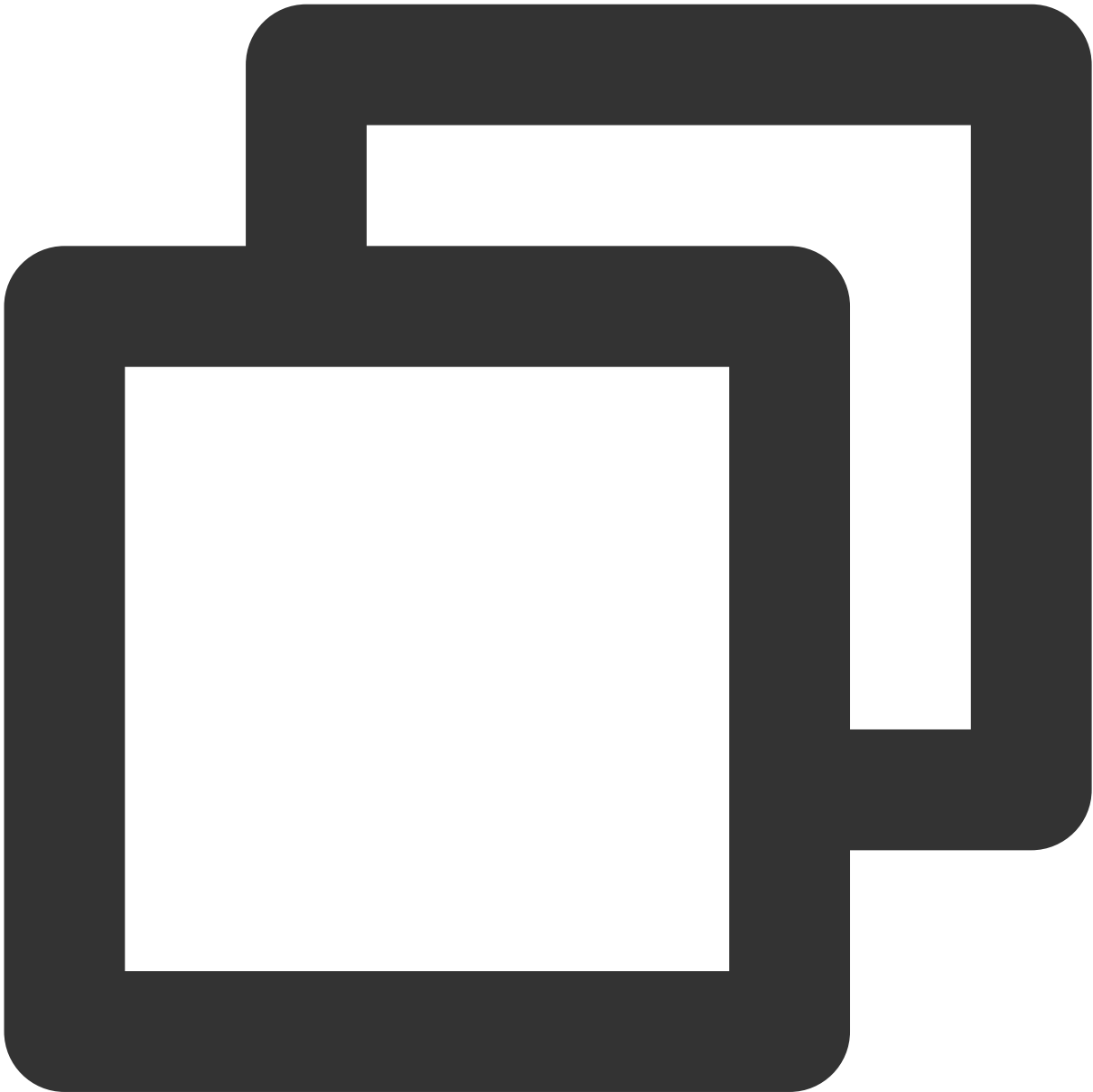
```
void setVideoResolutionMode(  
    TUIVideoStreamType streamType, TUIResolutionMode resolutionMode);
```

Parameter	Type	Meaning
streamType	TUIVideoStreamType	Video Stream type

resolutionMode	TUIResolutionMode	Video resolution mode
----------------	-------------------	-----------------------

### enableGravitySensor

Enable the gravity sensor

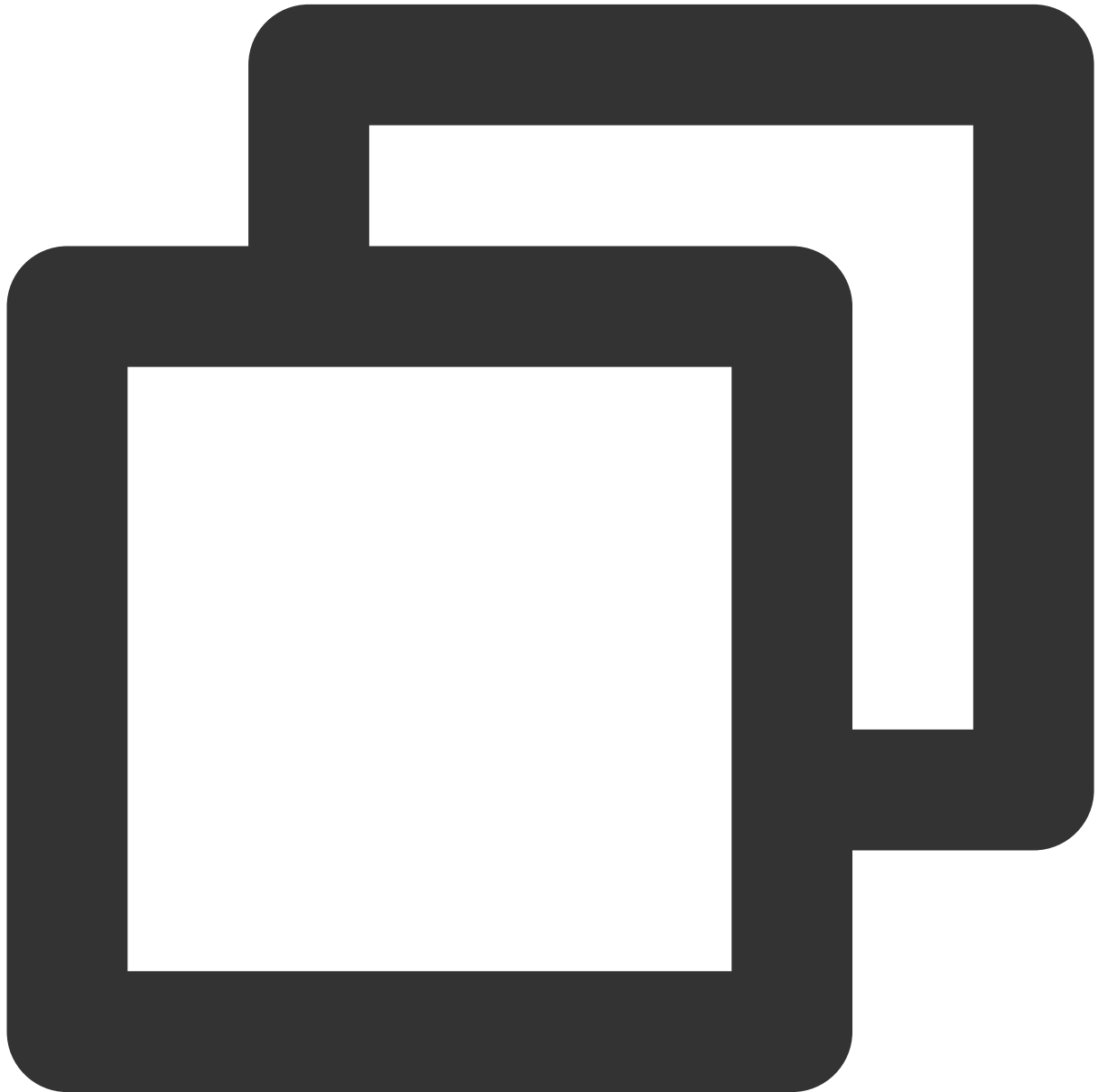


```
void enableGravitySensor(bool enable);
```

Parameter	Type	Meaning
enable	bool	Whether to enable

## **startPushLocalVideo**

Start pushing Local Video streams to Remote.



```
void startPushLocalVideo()
```

## **stopPushLocalVideo**

Stop pushing Local Video streams to Remote.



```
void stopPushLocalVideo()
```

## startScreenSharing

Start screen sharing





```
Future<void> startScreenSharing({String appGroup = ''})
```

## stopScreenSharing

Stop screen sharing



```
Future<void> stopScreenSharing()
```

## **openLocalMicrophone**

Open Local mic.



```
Future<TUIActionCallback> openLocalMicrophone(TUIAudioQuality quality)
```

Parameter	Type	Meaning
quality	<a href="#">TUIAudioQuality</a>	Audio Quality

## closeLocalMicrophone

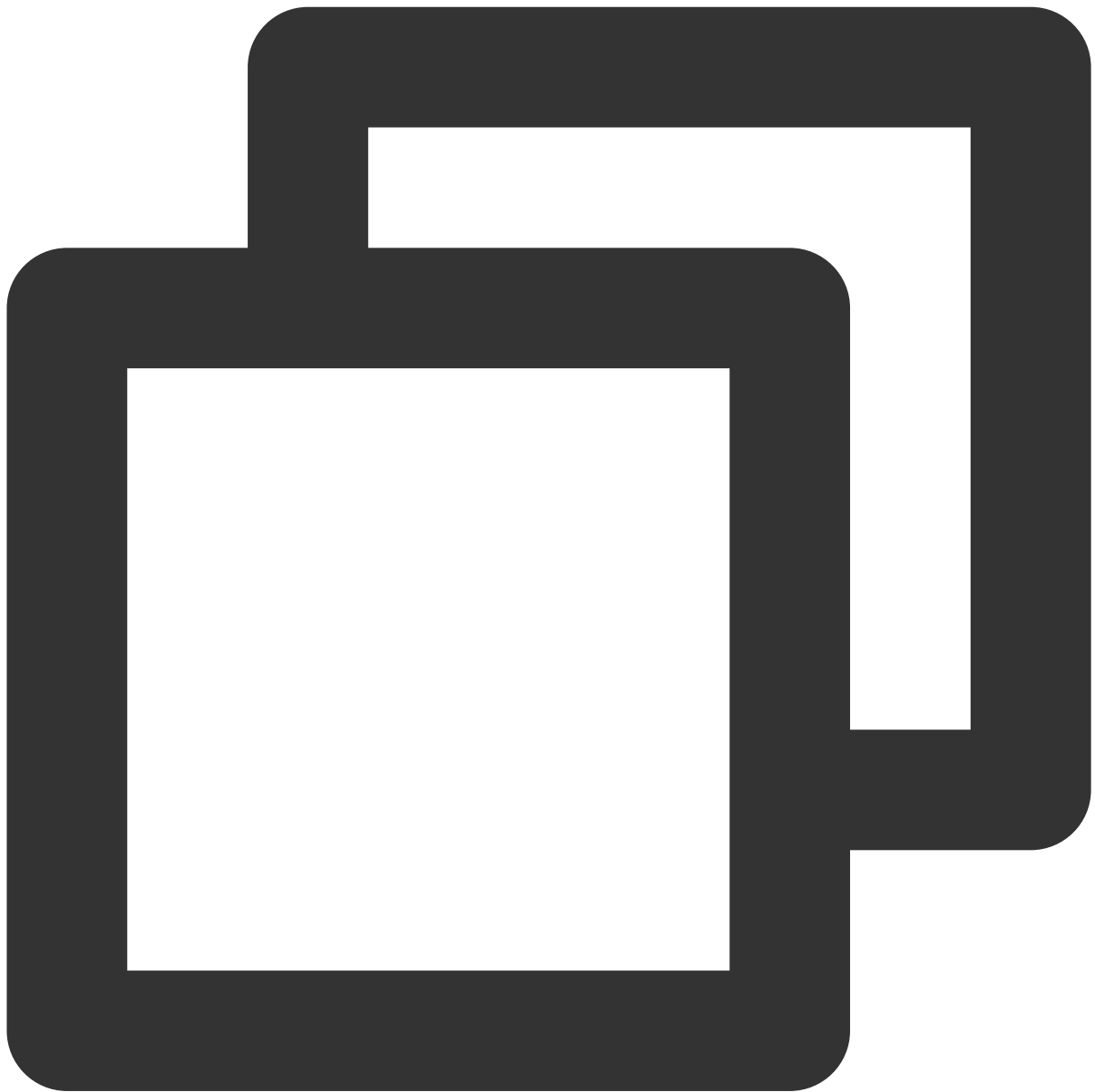
Close Local mic.



```
void closeLocalMicrophone()
```

### **updateAudioQuality**

Update Local Audio Codec Quality setting.



```
void updateAudioQuality(TUIAudioQuality quality)
```

**Parameters:**

Parameter	Type	Meaning
quality	<a href="#">TUIAudioQuality</a>	Audio Quality

**muteLocalAudio**

Mute local audio



```
Future<TUIActionCallback> muteLocalAudio()
```

## **unMuteLocalAudio**

UnMute local audio



```
Future<TUIActionCallback> unMuteLocalAudio()
```

### **setRemoteVideoView**

Set Remote user Video Rendering View control.



```
void setRemoteVideoView(String userId,
                        TUIVideoStreamType streamType,
                        int viewId)
```

**Parameters:**

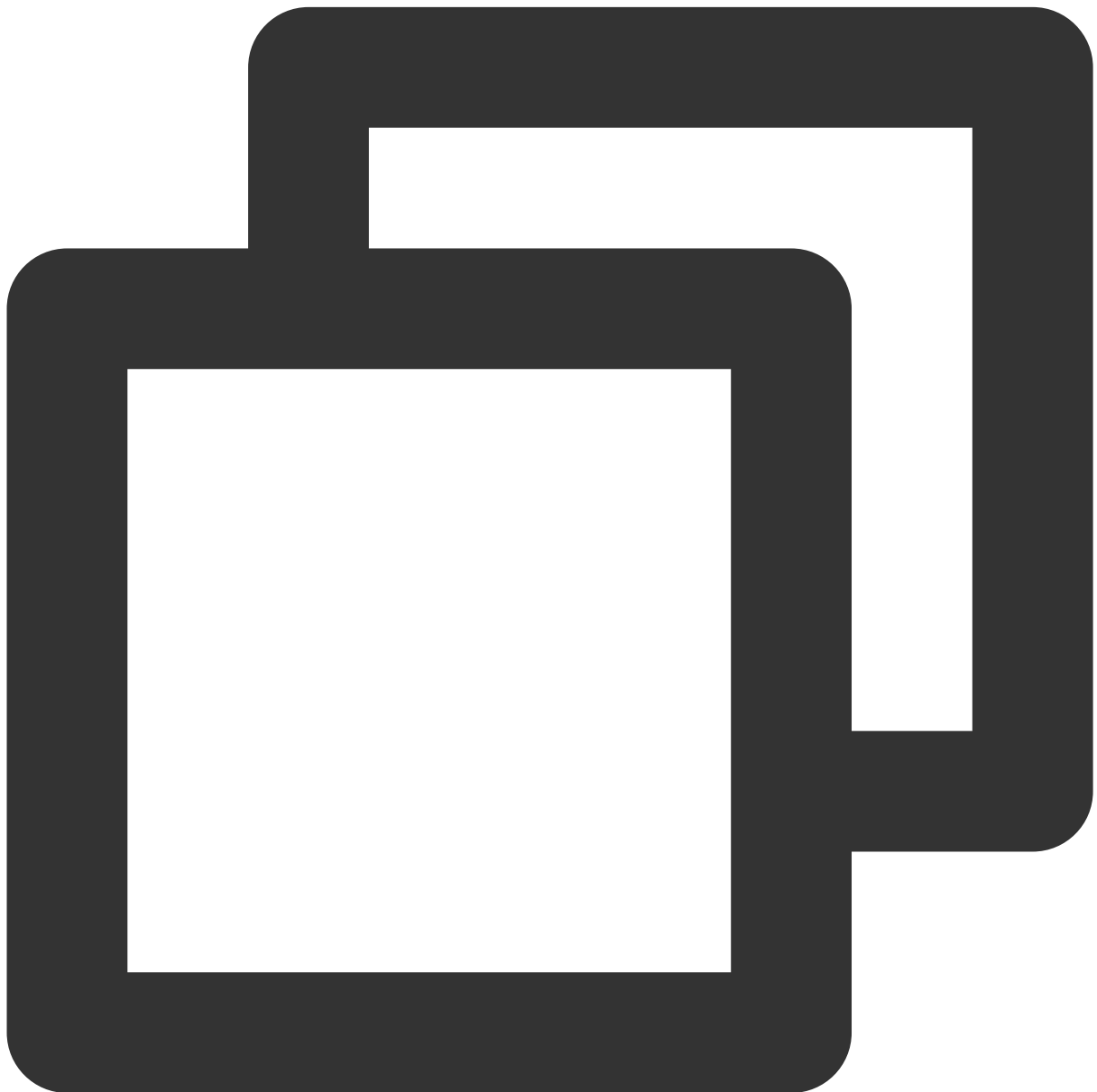
Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIVideoStreamType</a>	User streams type



viewId	int	The int64 type value of the pointer to the view to be rendered, through this viewId, can be converted to the corresponding native platform view, and the video screen will be rendered on this view.
--------	-----	--

## startPlayRemoteVideo

Start Playback Remote user Video streams.



```
void startPlayRemoteVideo(String userId,  
                           TUIVideoStreamType streamType,
```

```
TUIPlayCallback? callback)
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIVideoStreamType</a>	User streams type
callback	TUIPlayCallback?	Playback Operation result Callback

**stopPlayRemoteVideo**

Stop Playback Remote user Video streams.



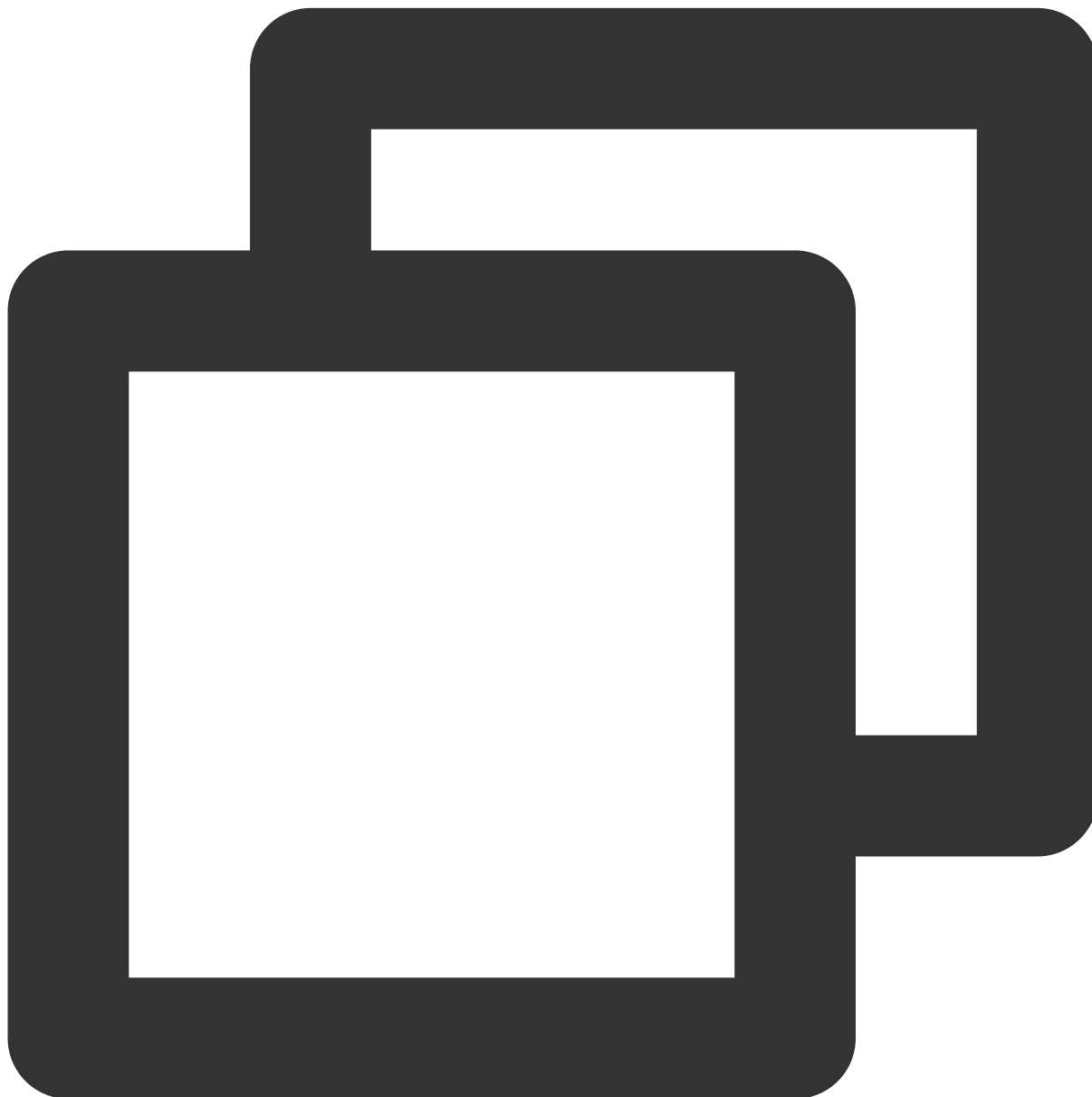
```
void stopPlayRemoteVideo(String userId,  
                          TUIVideoStreamType streamType)
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
streamType	<a href="#">TUIVideoStreamType</a>	User streams type

## **muteRemoteAudioStream**

Mute Remote user.



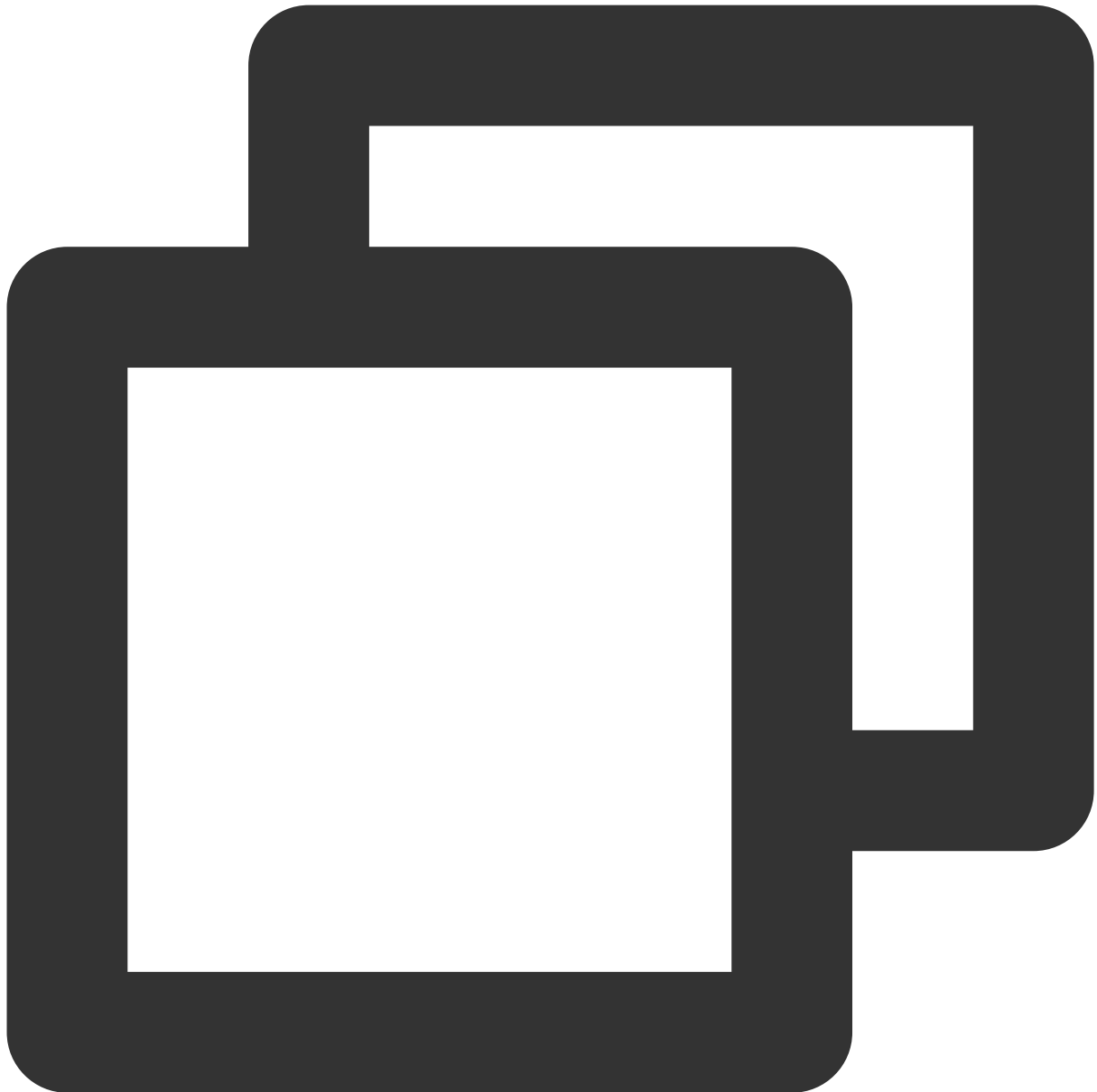
```
void muteRemoteAudioStream(String userId, boolean isMute);
```

### **Parameters:**

Parameter	Type	Meaning
userId	String	User ID
isMute	bool	Whether to mute

## getUserList

Get current User list in the room, Note that the maximum number of User list fetched by this Interface is 100.



```
Future<TUIValueCallBack<TUIUserListResult>> getUserList(int nextSequence)
```

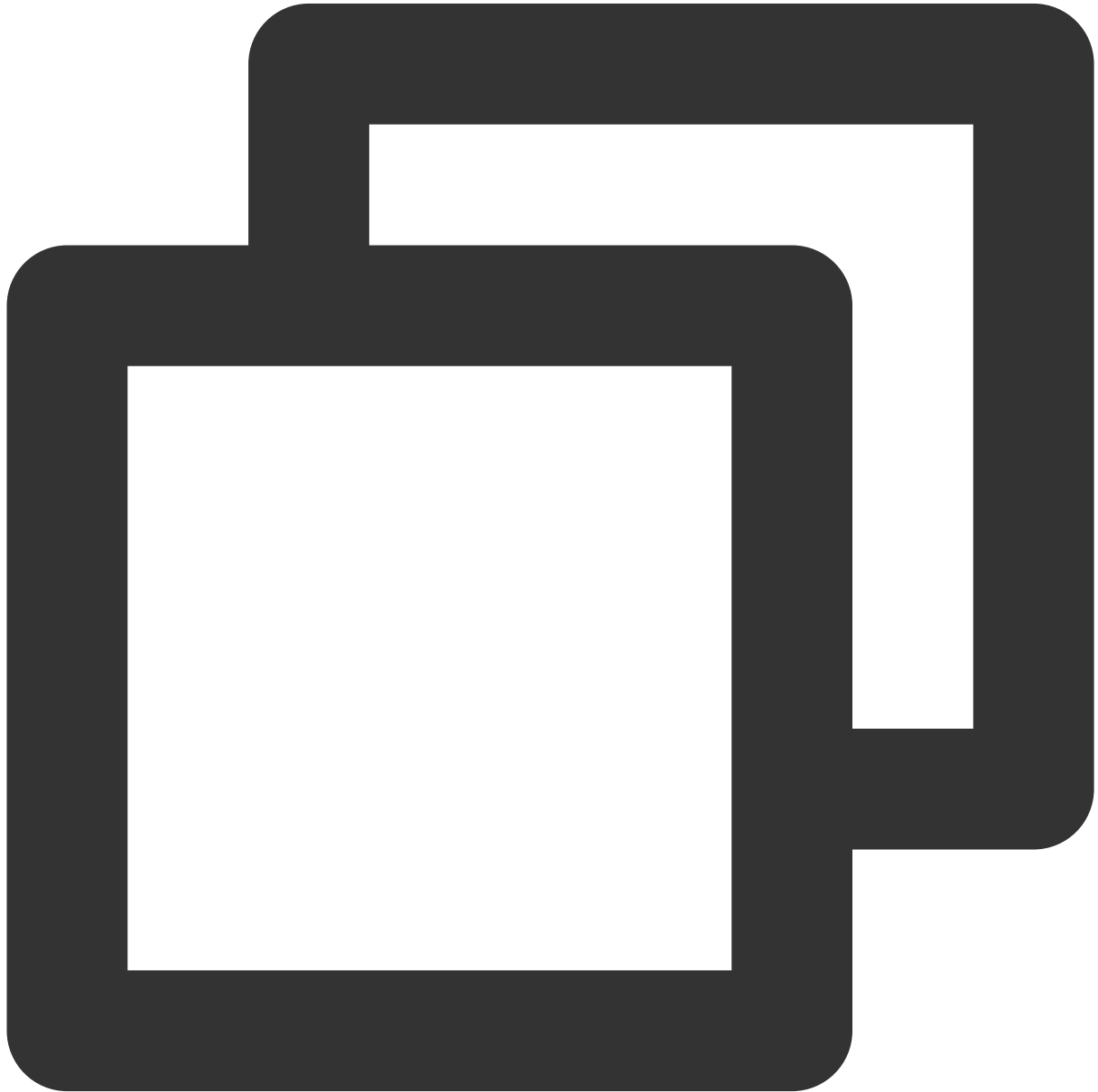
### Parameters:

Parameter	Type	Meaning
nextSequence	int	Pagination Fetch Flag, fill in 0 for the first Fetch, if the

nextSeq in the Callback is not 0, you need to do  
Pagination, pass in the nextSeq to Fetch again until the  
nextSeq in the Callback is 0

## getUserInfo

Get User [Learn more.](#)



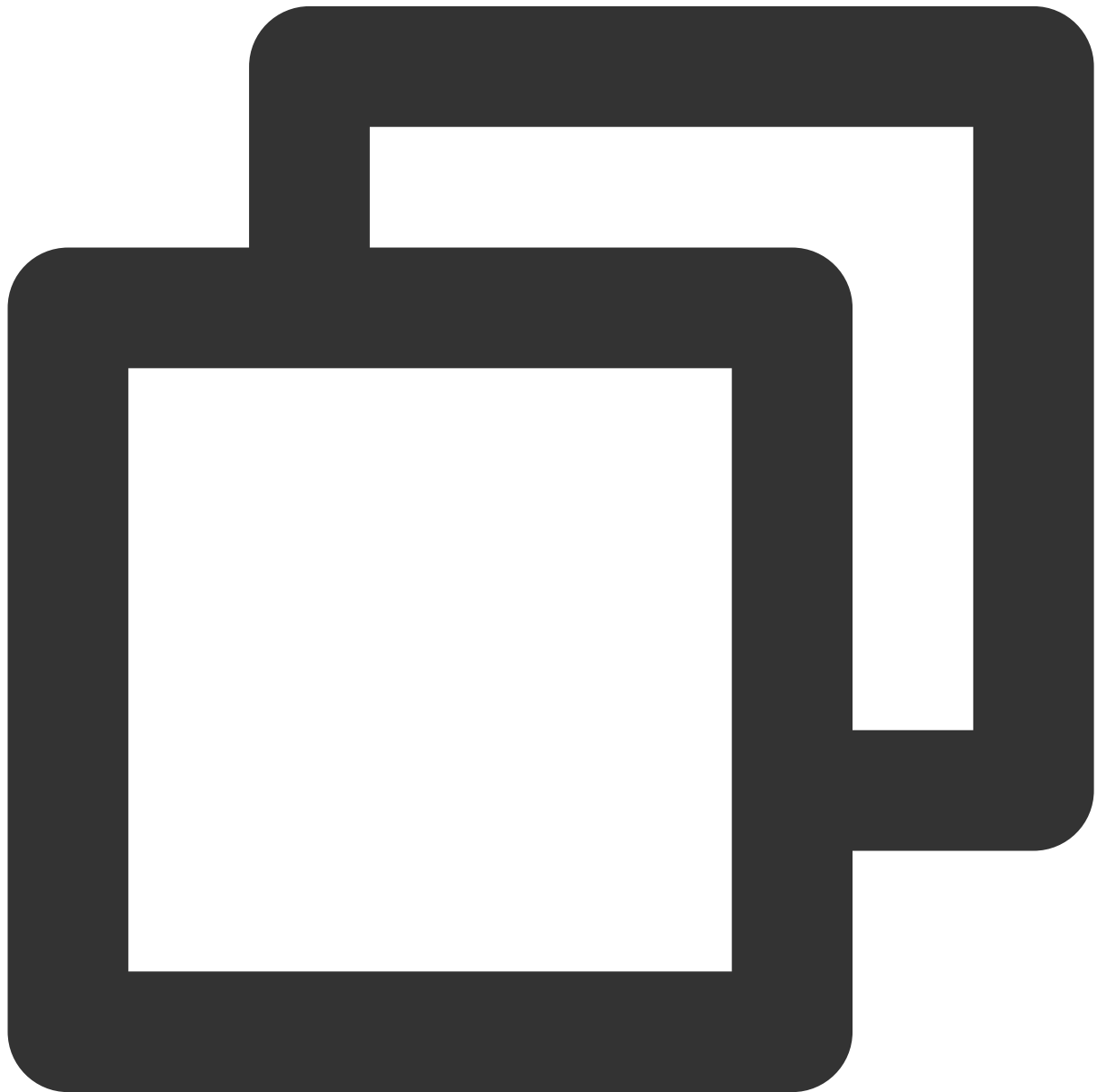
```
Future<TUIValueCallBack<TUIUserInfo>> getUserInfo(String userId)
```

### Parameters:

Parameter	Type	Meaning
userId	String	Get Learn more of the user by userId

## changeUserRole

Change user Role, only Administrator or Group owner can call.



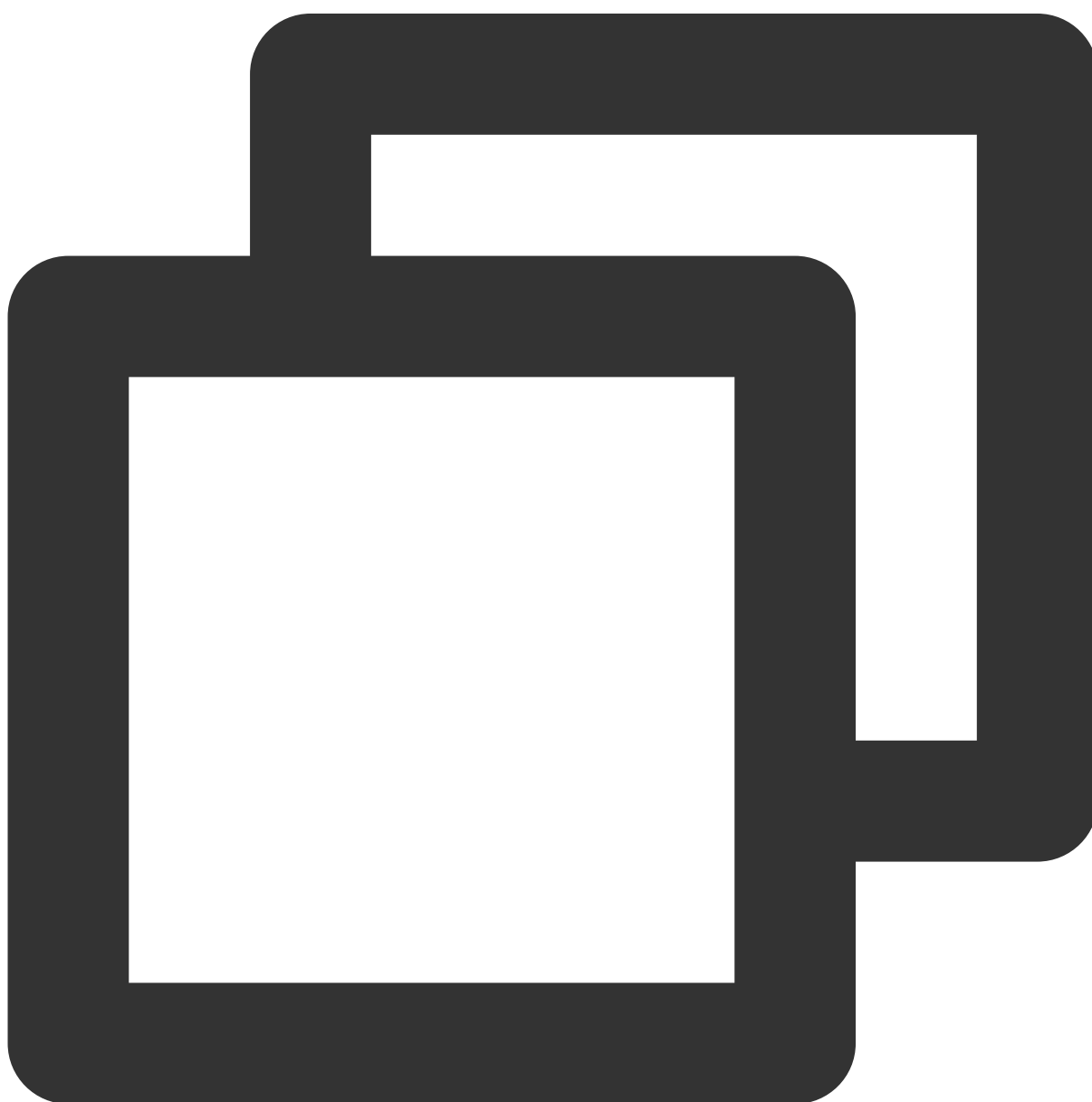
```
Future<TUIActionCallback> changeUserRole(String userId,  
                                         TUIRole role)
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID
role	<a href="#">TUIRole</a>	User Role

**kickRemoteUserOutOfRoom**

Kick user out of the room, only Administrator or Group owner can call.





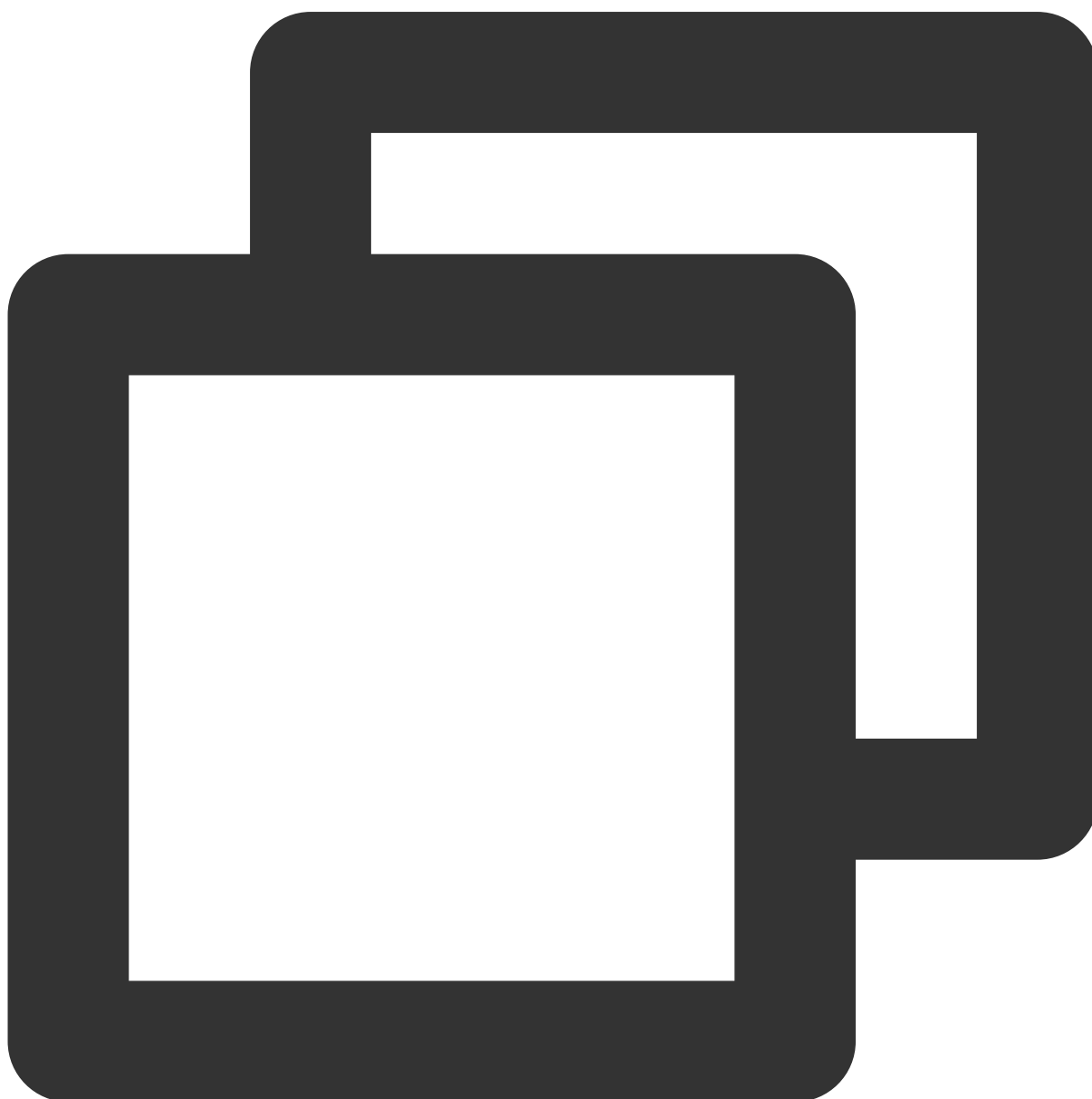
```
Future<TUIActionCallback> kickRemoteUserOutOfRoom(String userId)
```

**Parameters:**

Parameter	Type	Meaning
userId	String	User ID

**addCategoryTagForUsers**

Add category tags to users (Only Administrator or Group Owner can call)



```
Future<TUIActionCallback> addCategoryTagForUsers(int tag, List<String> userList);
```

**Parameters :**

Parameter	Type	Meaning
tag	int	Tag type. Number type, greater than or equal to 1000, you can customize
userList	List<String>	User list

**removeCategoryTagForUsers**

Remove category tags to users (Only Administrator or Group Owner can call)



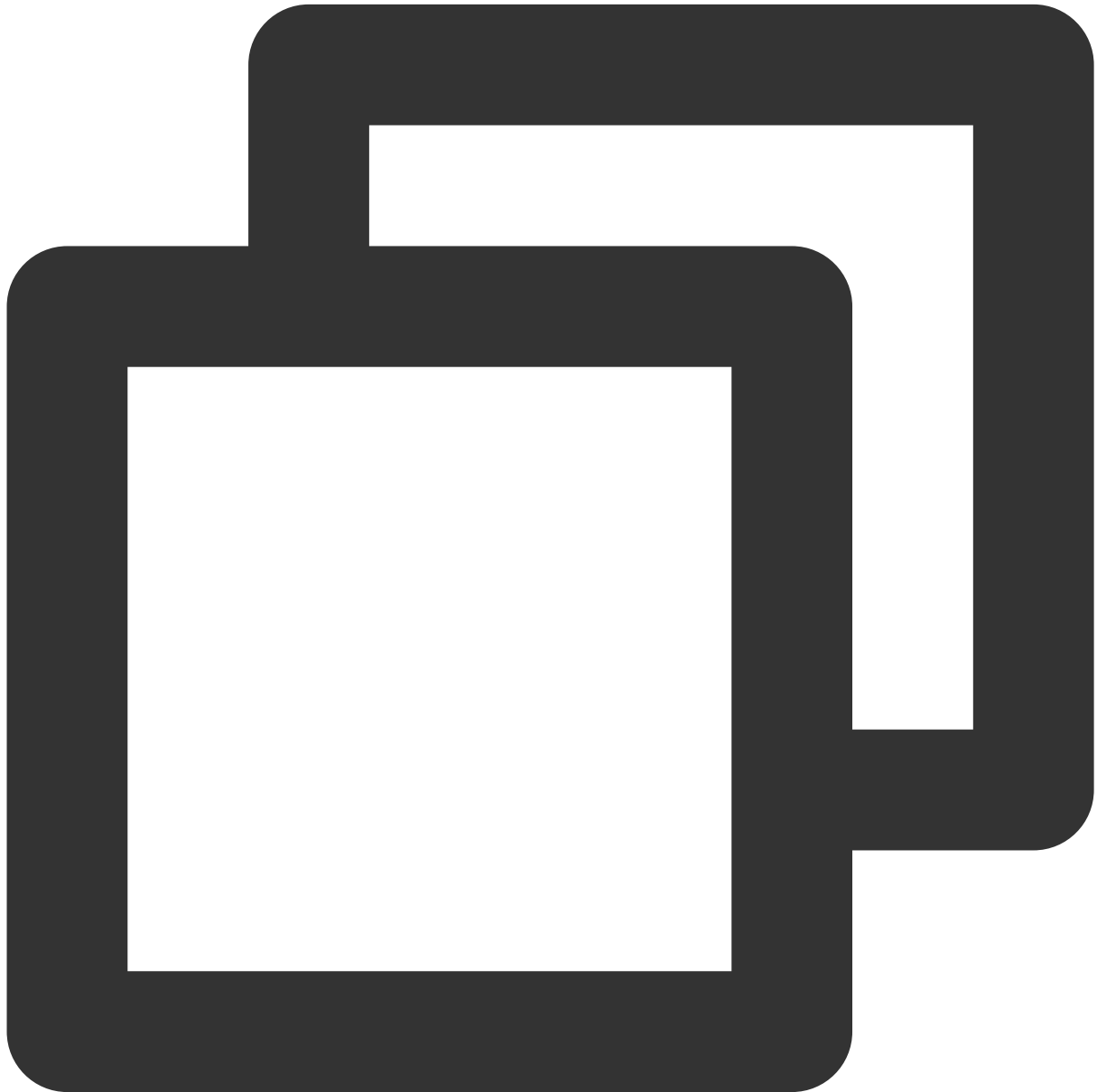
```
Future<TUIActionCallback> removeCategoryTagForUsers(int tag, List<String> userList)
```

**Parameters :**

Parameter	Type	Meaning
tag	int	Tag type. Number type, greater than or equal to 1000, you can customize
userList	List<String>	User list

## getUserListByTag

Get user information in the room based on tags



```
Future<TUIValueCallBack<TUIUserListResult>> getUserListByTag(int tag, int nextSeque
```

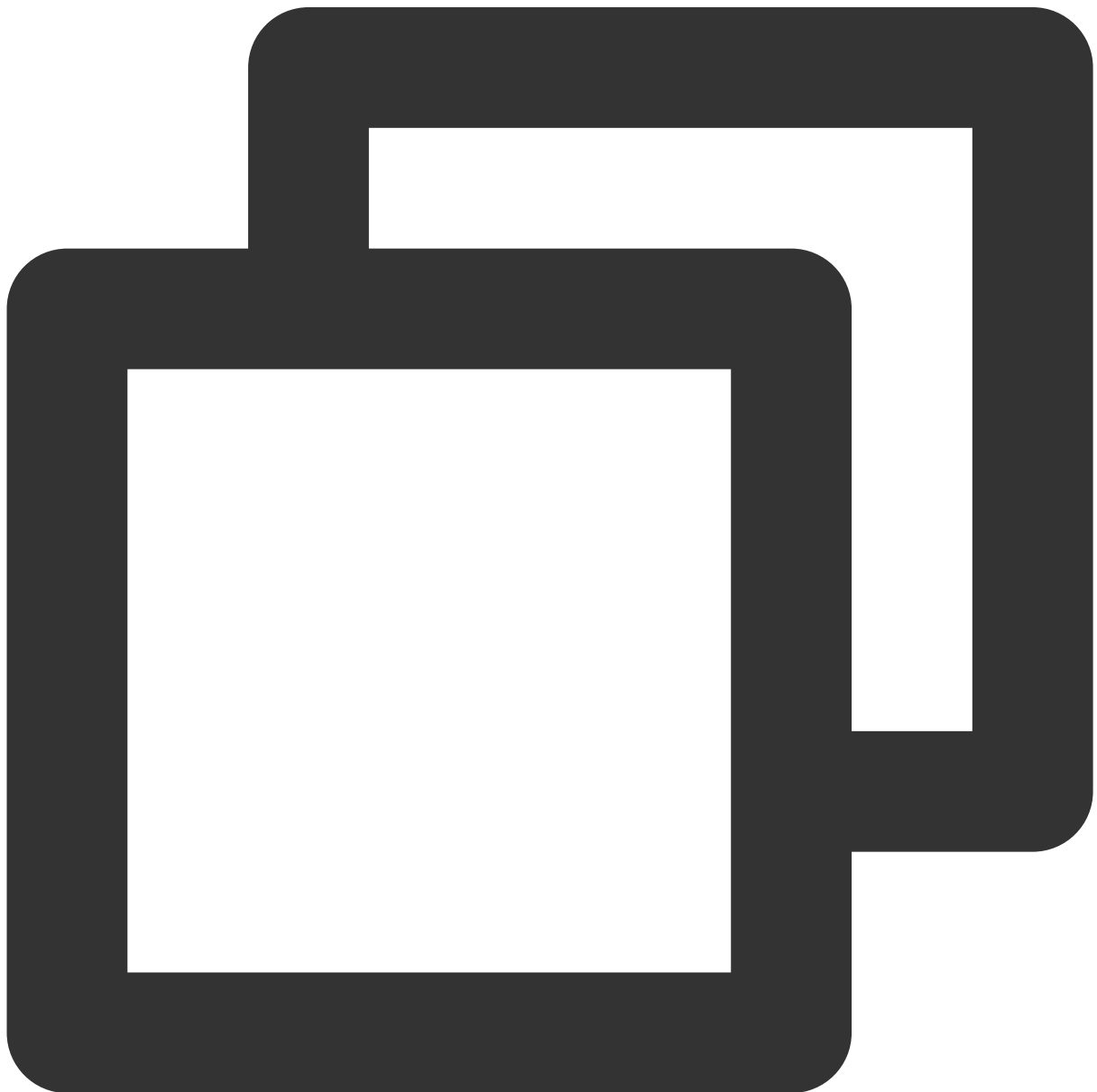
### Parameters :

Parameter	Type	Meaning
tag	int	Tag type. Number type, greater than or equal to 1000, you can customize

nextSequence	int	Pagination Fetch Flag, fill in 0 for the first Fetch, if the nextSeq in the Callback is not 0, you need to do Pagination, pass in the nextSeq to Fetch again until the nextSeq in the Callback is 0
--------------	-----	---

### **disableDeviceForAllUserByAdmin**

Manage Media device for all users, only Administrator or Group owner can call.

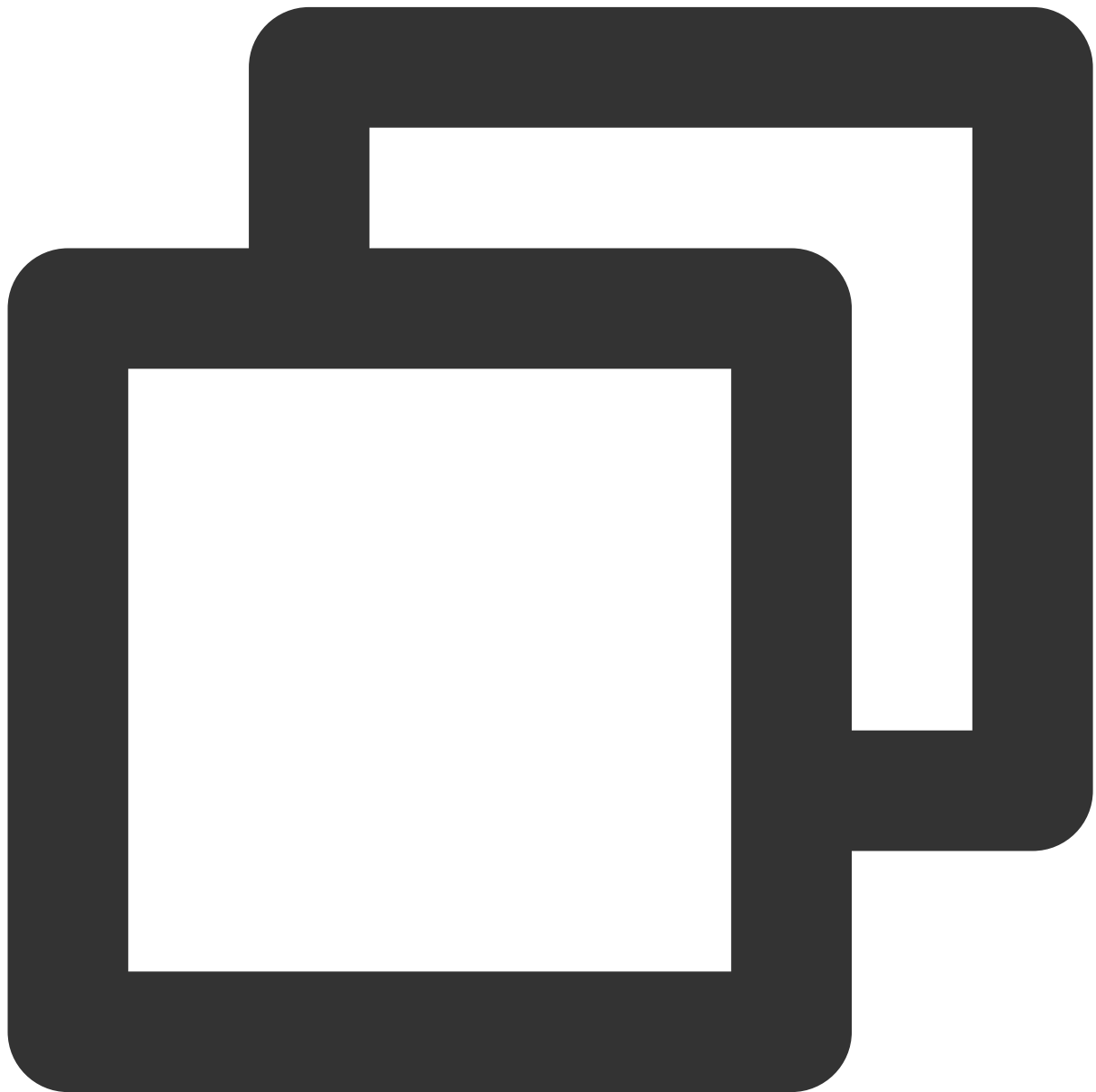


```
Future<TUIActionCallback> disableDeviceForAllUserByAdmin(TUIMediaDevice device,  
                                                         bool isDisable)
```

Parameter	Type	Meaning
device	<a href="#">TUIMediaDevice</a>	Device
isDisable	bool	Whether to Disable

## openRemoteDeviceByAdmin

Request Remote user to open Media device, only Administrator or Group owner can call.



```
TUIRequest openRemoteDeviceByAdmin(String userId,
```

```
TUIMediaDevice device,  
int timeout,  
TUIRequestCallback? requestCallback)
```

Parameter	Type	Meaning
userId	String	User ID
device	<a href="#">TUIMediaDevice</a>	Device
timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
requestCallback	TUIRequestCallback?	Operation result Callback

### closeRemoteDeviceByAdmin

Close Remote user Media device, only Administrator or Group owner can call.



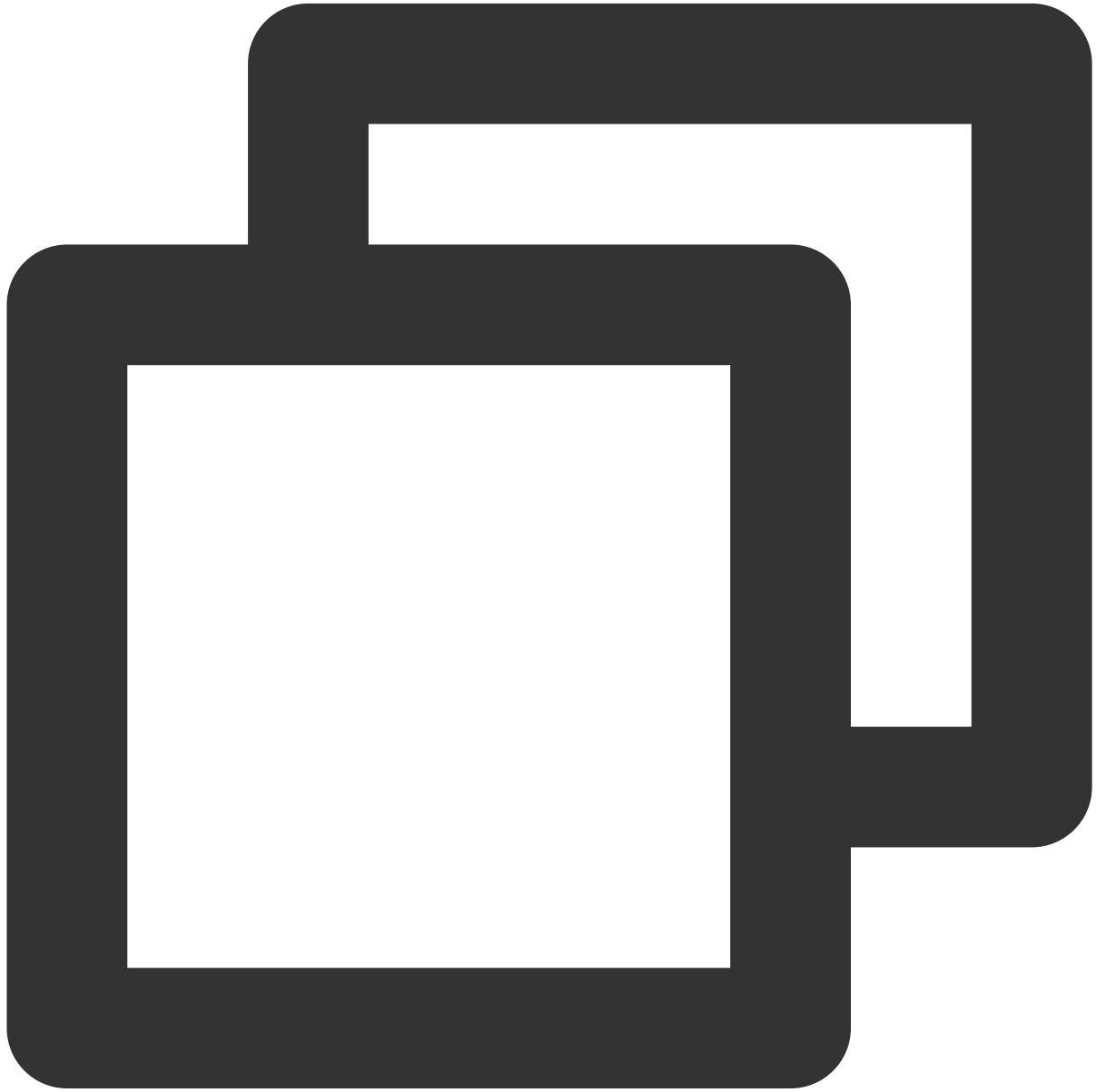
```
Future<TUIActionCallback> closeRemoteDeviceByAdmin(String userId,  
                                                    TUIMediaDevice device)
```

Parameter	Type	Meaning
userId	String	User ID
device	<a href="#">TUIMediaDevice</a>	Device

### **applyToAdminToOpenLocalDevice**



Lock all users' Media device management.



```
TUIRequest applyToAdminToOpenLocalDevice(TUIMediaDevice device,  
                                           int timeout,  
                                           TUIRequestCallback? requestCallback)
```

Parameter	Type	Meaning
device	<a href="#">TUIMediaDevice</a>	Device
timeout	int	Timeout Duration, Unit in Seconds. If Set to 0, SDK Will Not

		Perform Timeout Detection, Nor Will It Trigger Timeout Callback
callback	TUIRequestCallback	Operation Result Callback

## setMaxSeatCount

Set Maximum number of seats, only supported when entering the room and creating the room

When roomType is RoomType.CONFERENCE (Education and Conference scene), maxSeatCount value is not limited;

When roomType is RoomType.LIVE\_ROOM (Live broadcast scene), maxSeatCount is limited to 16;



```
Future<TUIActionCallback> setMaxSeatCount(int maxSeatCount)
```

Parameter	Type	Meaning
maxSeatCount	int	Maximum number of seats

### lockSeatByAdmin

Lock seat (including position lock, audio status lock, video status lock).

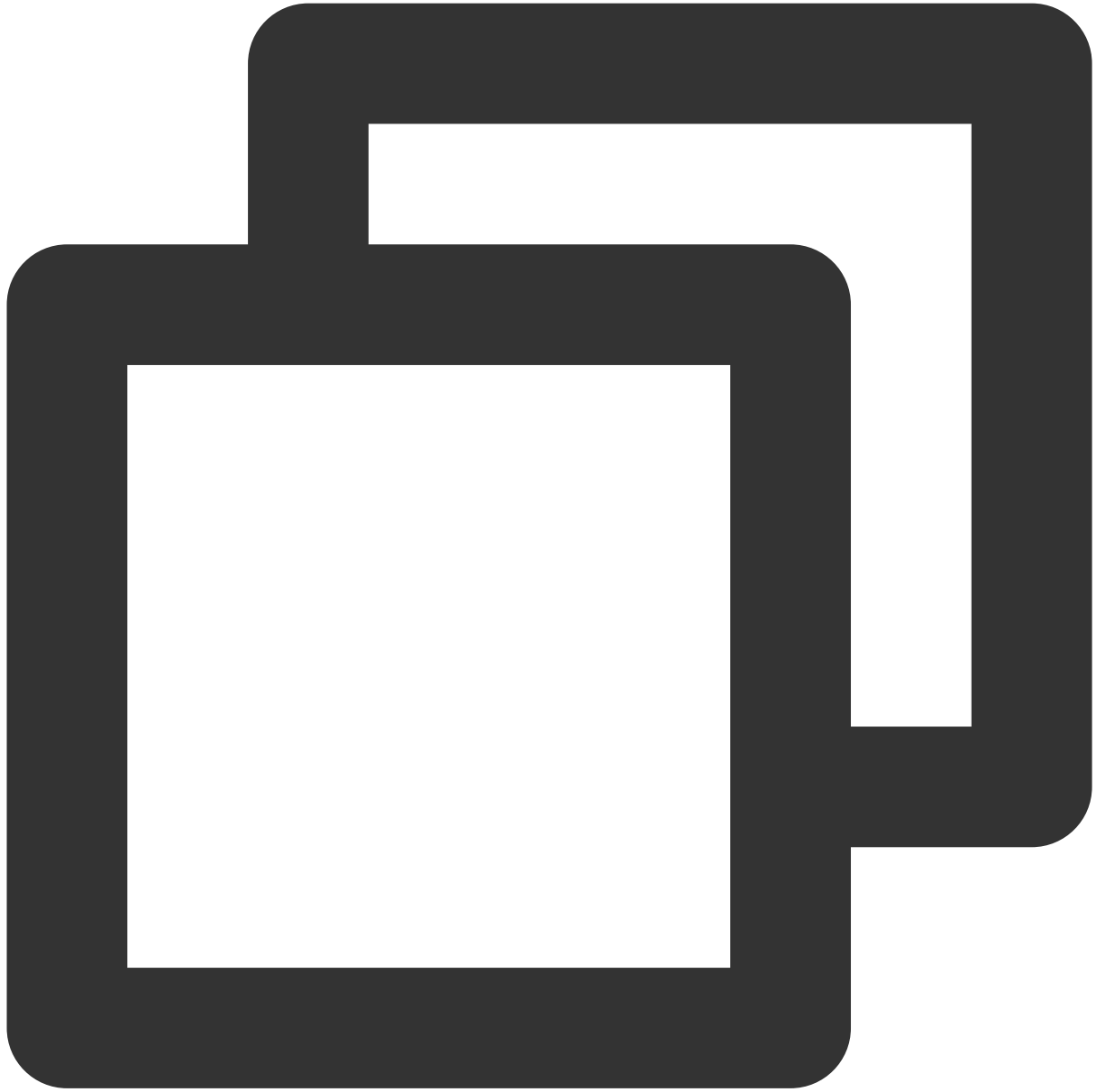


```
Future<TUIActionCallback> lockSeatByAdmin(int seatIndex,  
                                           TUISeatLockParams lockParams)
```

Parameter	Type	Meaning
seatIndex	int	Seat number
lockParams	<a href="#">TUISeatLockParams</a>	Lock microphone parameter

## getSeatList

Get seat list.



```
Future<TUIValueCallBack<List<TUISeatInfo>>> getSeatList ()
```

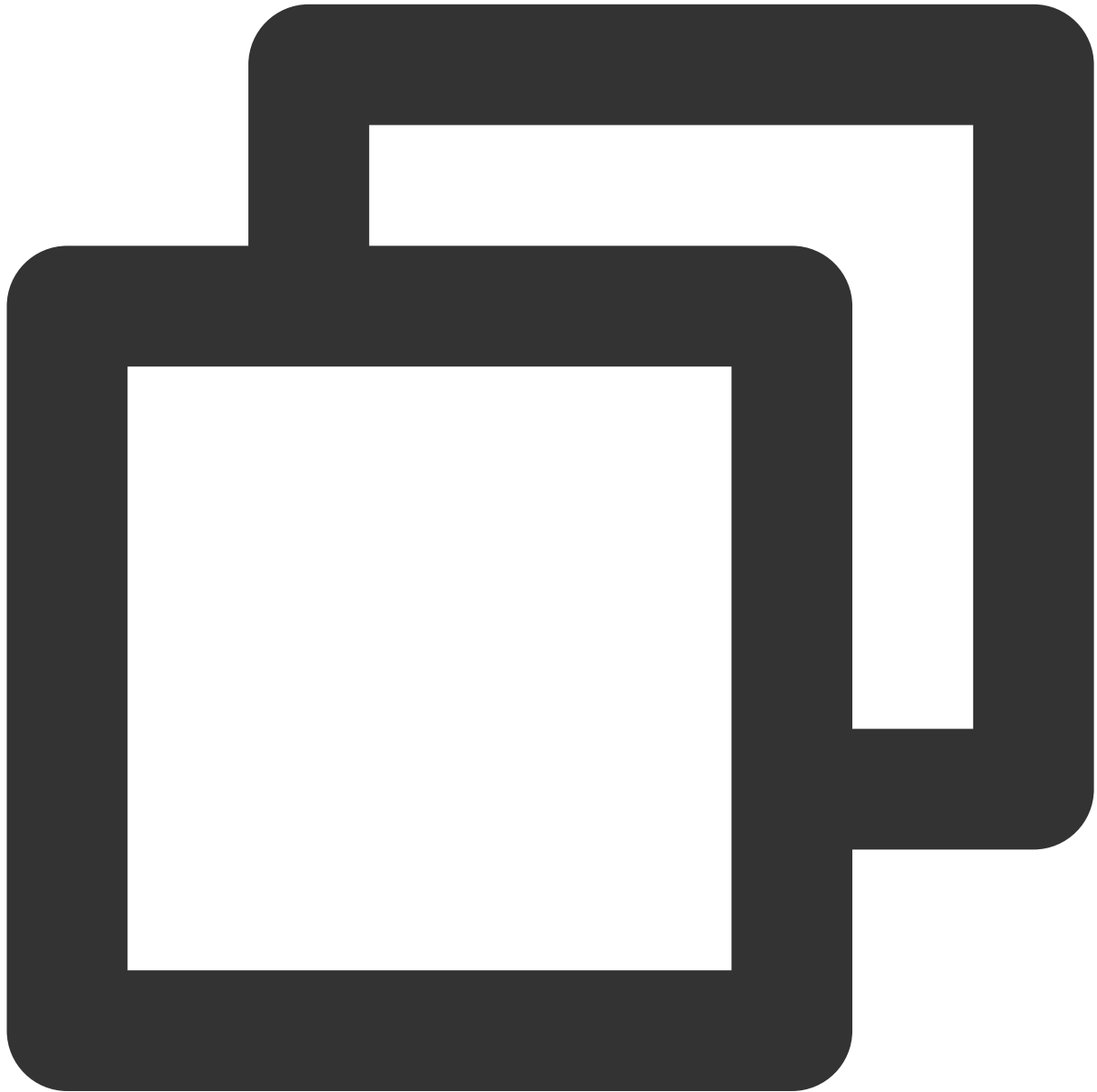
## takeSeat

Local on microphone.

### Explanation:

Conference Scene: [applyToSpeak](#) Mode needs to apply to the host or administrator to allow Go Live, other modes do not support Go Live.

Live Broadcast Scene: [freeToSpeak](#) Mode can freely Go Live, and speak after Go Live; [applySpeakAfterTakingSeat](#) Mode needs to apply to the host or administrator to allow Go Live; other modes do not support Go Live.



```
TUIRequest takeSeat(int seatIndex,  
                    int timeout,  
                    TUIRequestCallback? requestCallback)
```

**Parameters:**

Parameter	Type	Meaning
seatIndex	int	Seat number

timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
requestCallback	TUIRequestCallback?	Call interface Callback, used to notify the request Callback status

**Return:** Request body

## leaveSeat

Local off microphone.

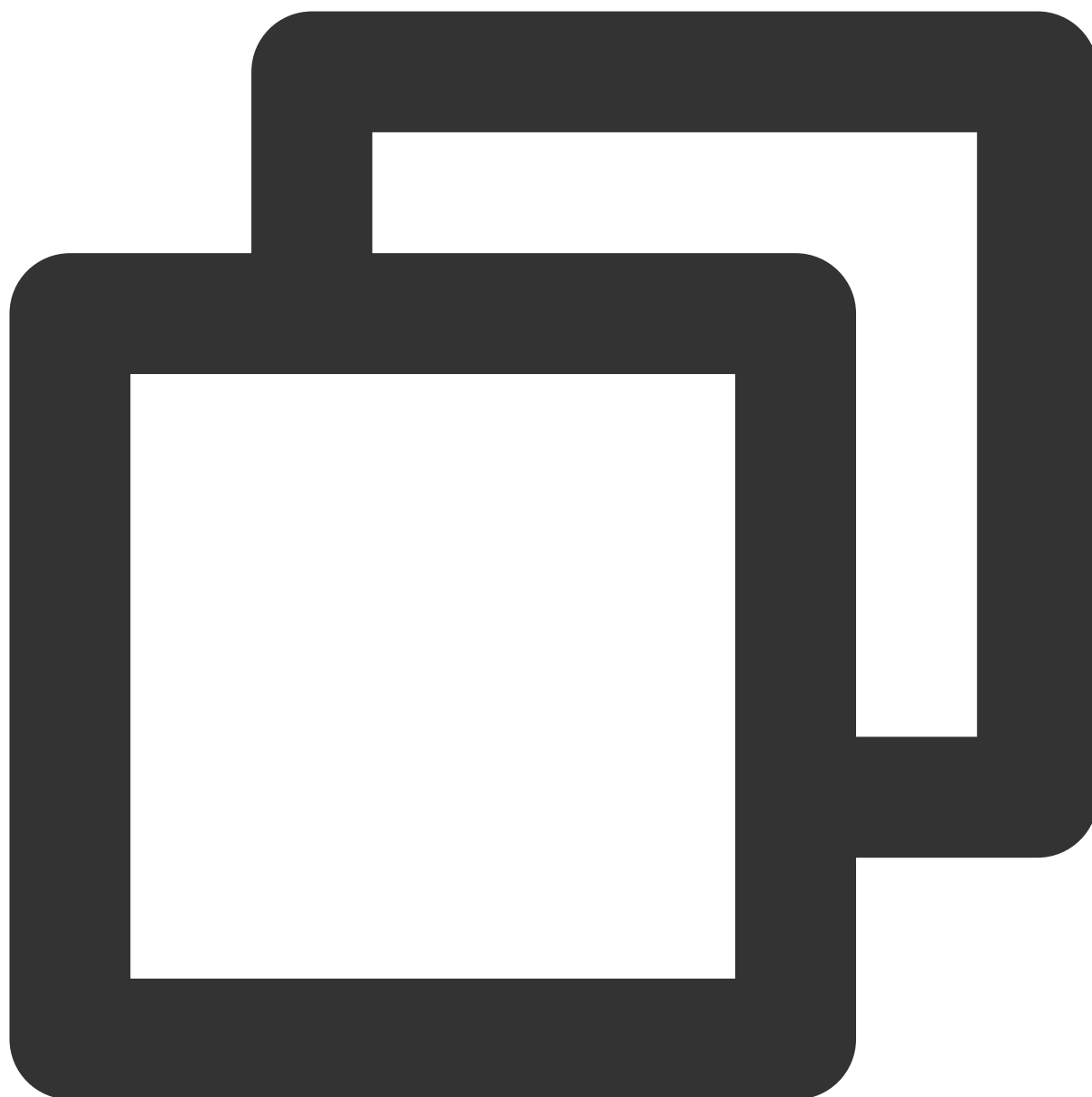


```
Future<TUIActionCallback> leaveSeat ()
```

### **takeUserOnSeatByAdmin**

Host/Administrator Invite User on microphone.





```
TUIRequest takeUserOnSeatByAdmin(int seatIndex,  
                                  String userId,  
                                  int timeout,  
                                  TUIRequestCallback? requestCallback)
```

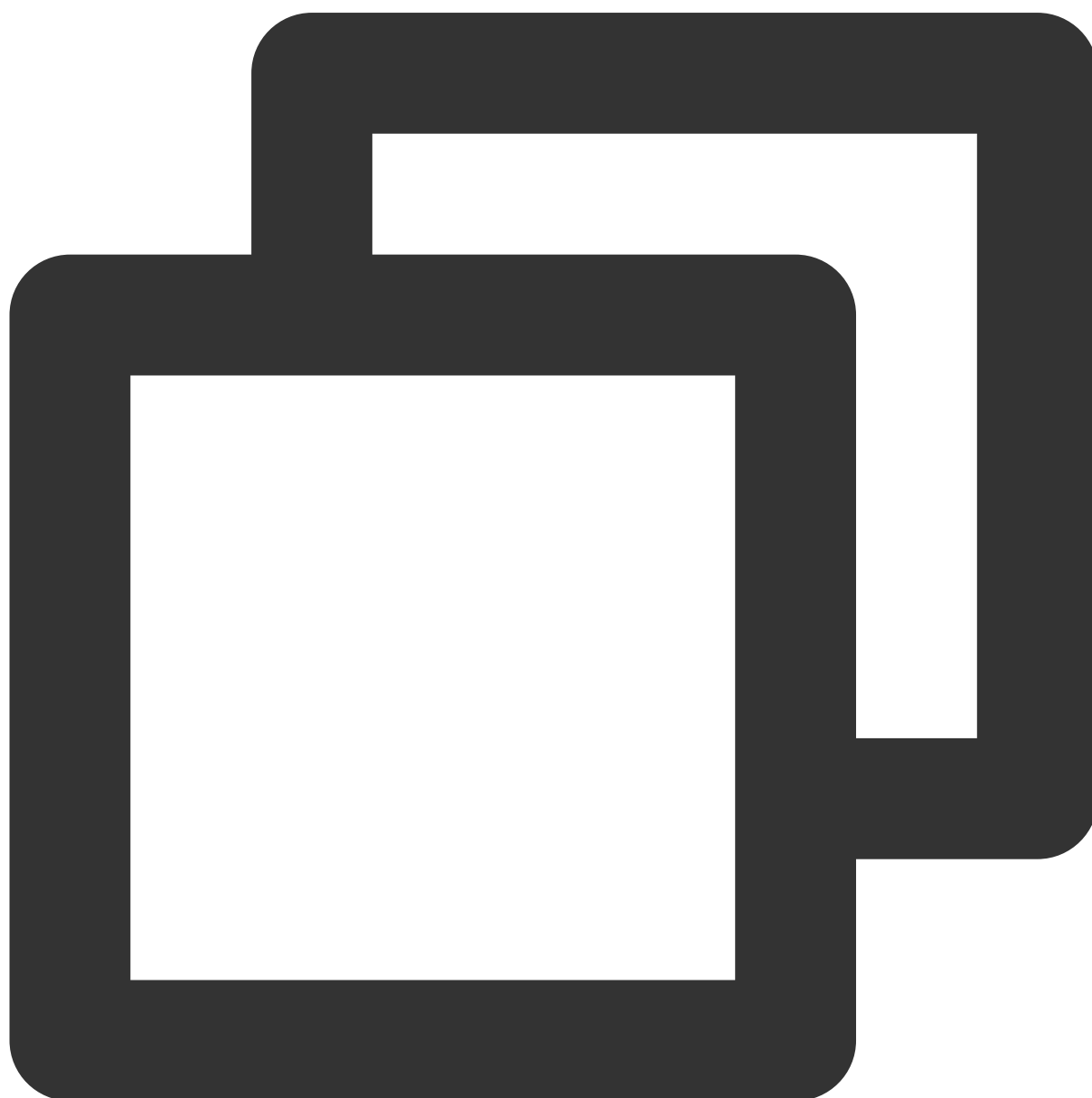
Parameter	Type	Meaning
seatIndex	int	Seat number
userId	String	User ID

timeout	int	Timeout in seconds, if set to 0, SDK will not do timeout detection and will not trigger timeout Callback
requestCallback	TUIRequestCallback?	Call interface Callback, used to notify the request Callback status

**Return:** Request body

### kickUserOffSeatByAdmin

Host/Administrator Take User off microphone.



```
Future<TUIActionCallback> kickUserOffSeatByAdmin(int seatIndex,  
                                                String userId)
```

Parameter	Type	Meaning
seatIndex	int	Seat number
userId	String	User ID

## sendTextMessage

Send Text message.



```
Future<TUIActionCallback> sendTextMessage(String message)
```

**Parameters:**

Parameter	Type	Meaning
message	String	Text message Content

**sendCustomMessage**

Send Custom message



```
Future<TUIActionCallback> sendCustomMessage(String message)
```

**Parameters:**

Parameter	Type	Meaning
message	String	Custom message Content

**disableSendingMessageByAdmin**

Disable Remote user's Text message sending Ability (only Administrator or Group owner can call).

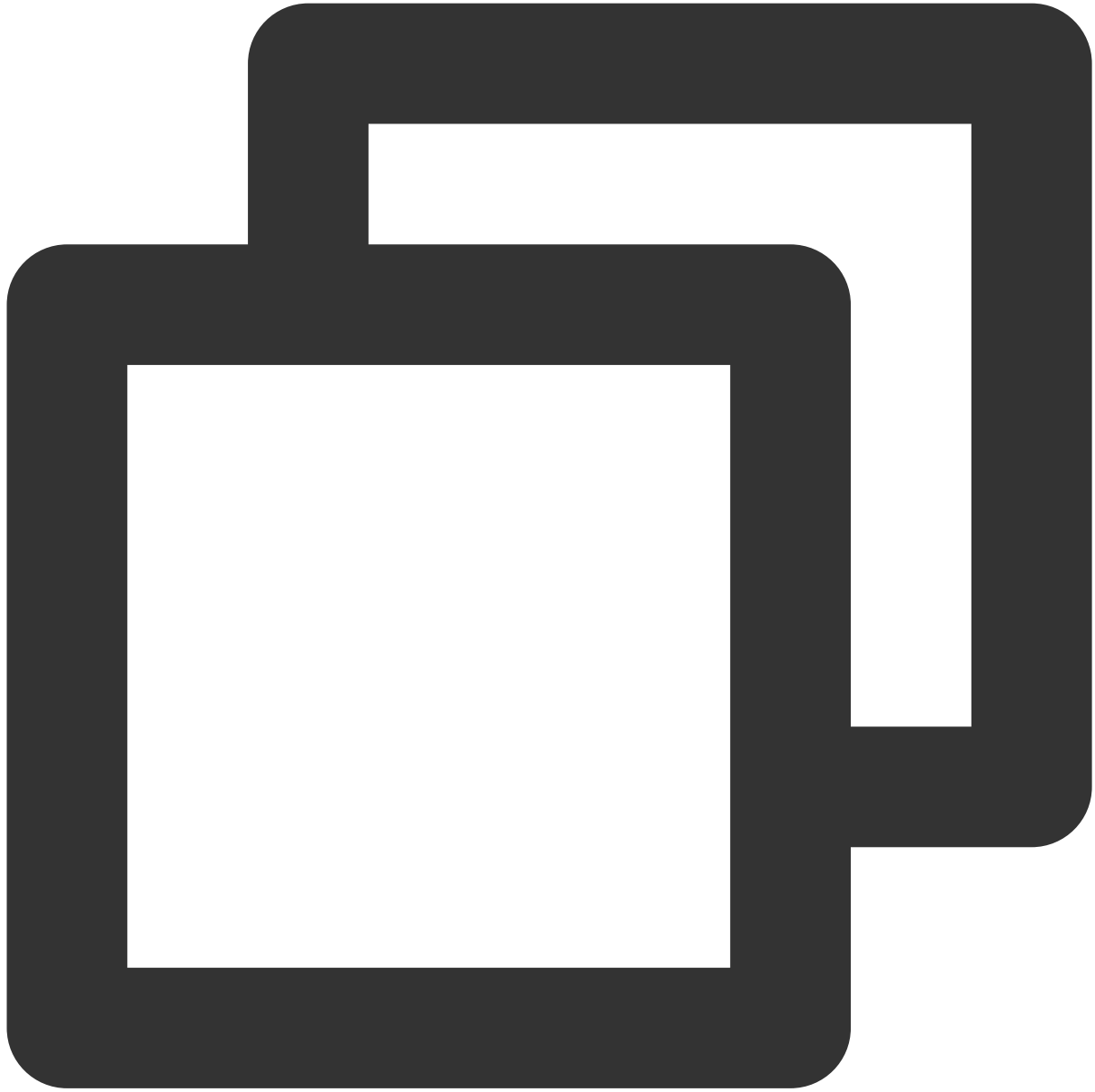


```
Future<TUIActionCallback> disableSendingMessageByAdmin(String userId,  
                                                         bool isDisable)
```

Parameter	Type	Meaning
userId	String	User ID
isDisable	bool	Whether to Disable

### **disableSendingMessageForAllUser**

Disable all users' Text message sending Ability (only Administrator or Group owner can call).



```
Future<TUIActionCallback> disableSendingMessageForAllUser(bool isDisable)
```

Parameter	Type	Meaning
isDisable	bool	Whether to Disable

### cancelRequest

Cancel sent Request.



```
Future<TUIActionCallback> cancelRequest (String requestId)
```

**Parameters:**

Parameter	Type	Meaning
requestId	String	Request ID

**responseRemoteRequest**

Reply to Remote user's Request.





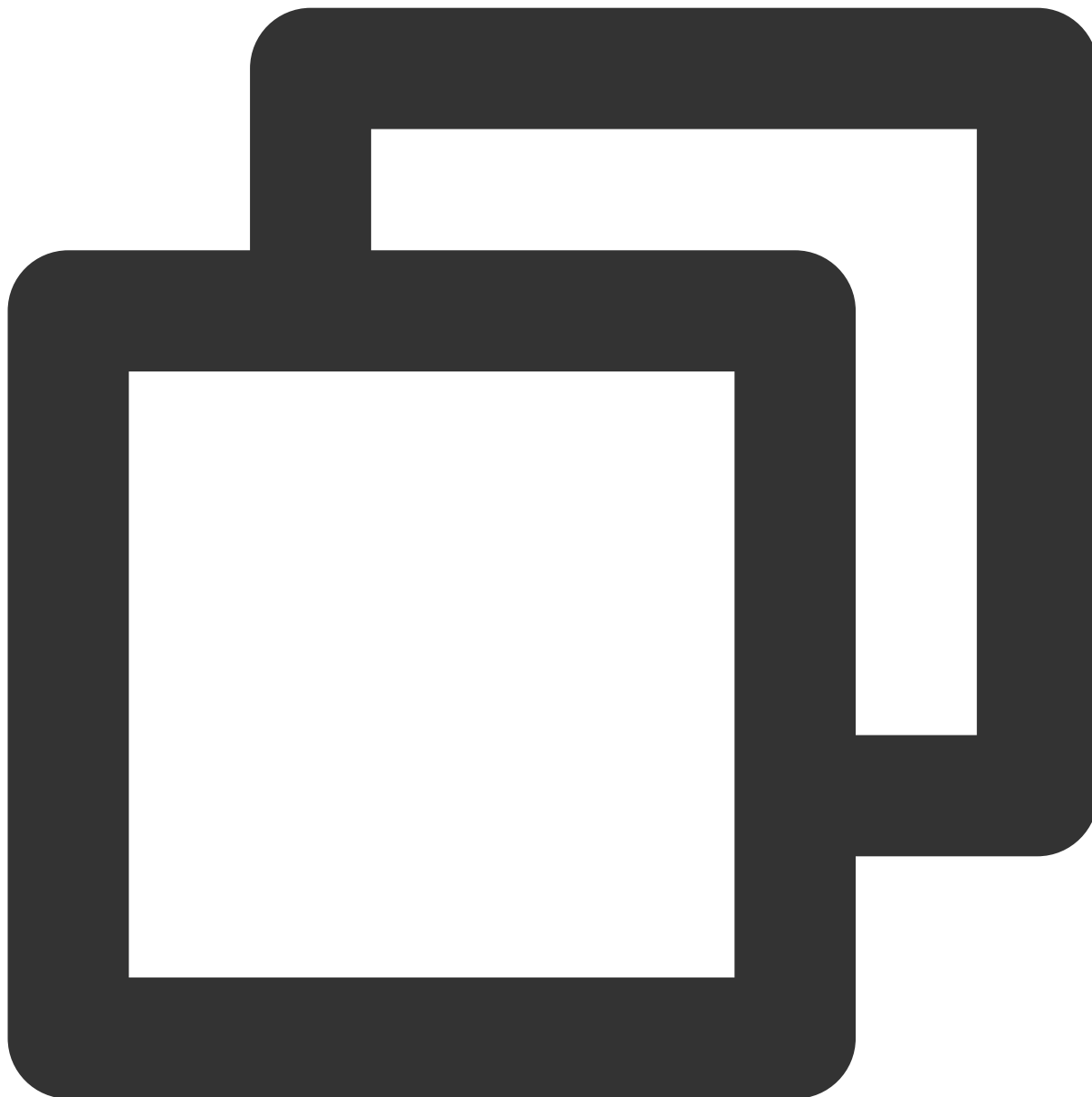
```
Future<TUIActionCallback> responseRemoteRequest(String requestId,  
                                                bool agree)
```

**Parameters:**

Parameter	Type	Meaning
requestId	String	Request ID
agree	bool	Whether to agree

## switchCamera

Switch front/rear camera

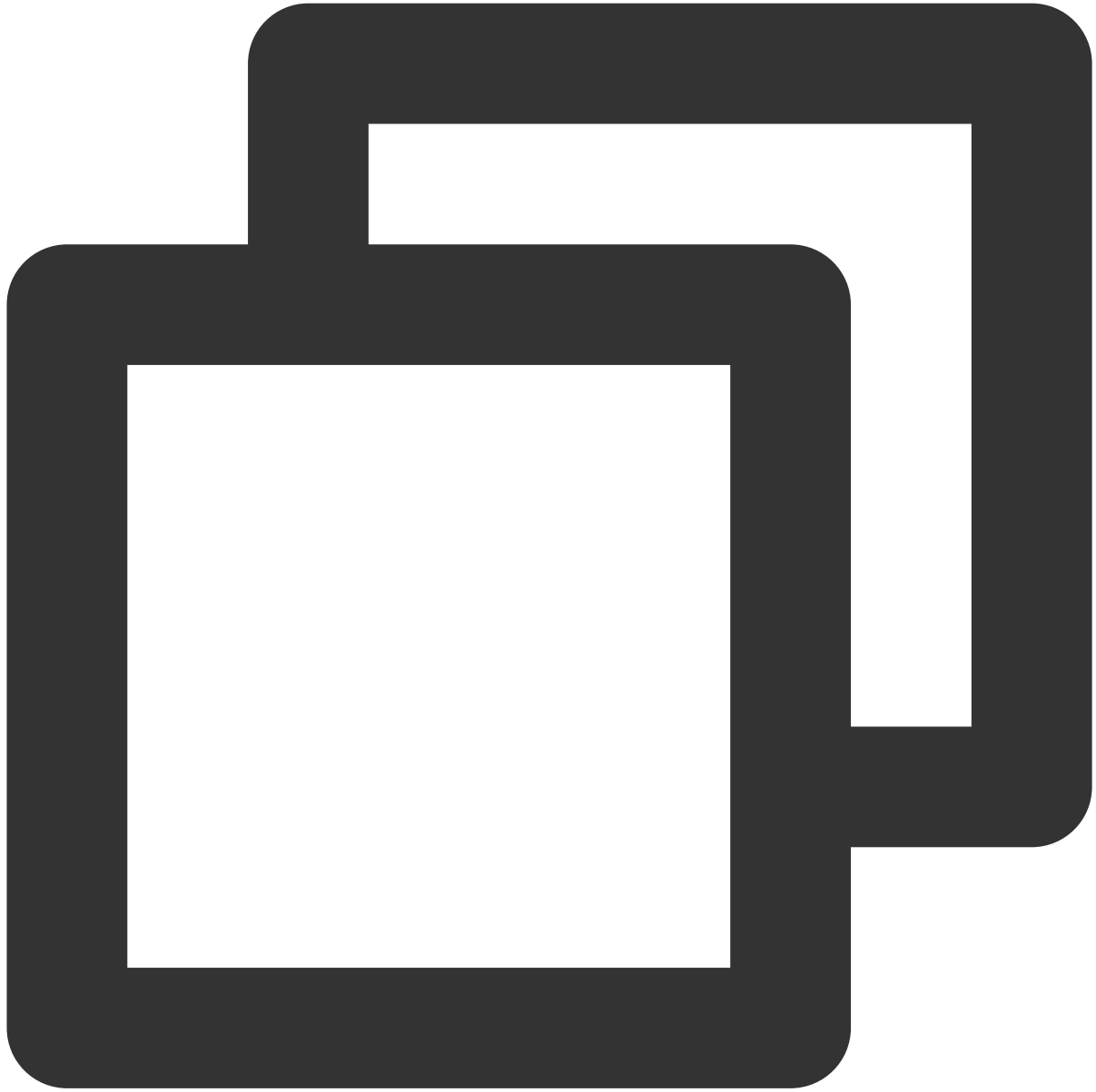


```
Future<int?> switchCamera(bool isFrontCamera);
```

Parameter	Type	Meaning
isFrontCamera	bool	Is front camera

## setBeautyLevel

Set beauty level

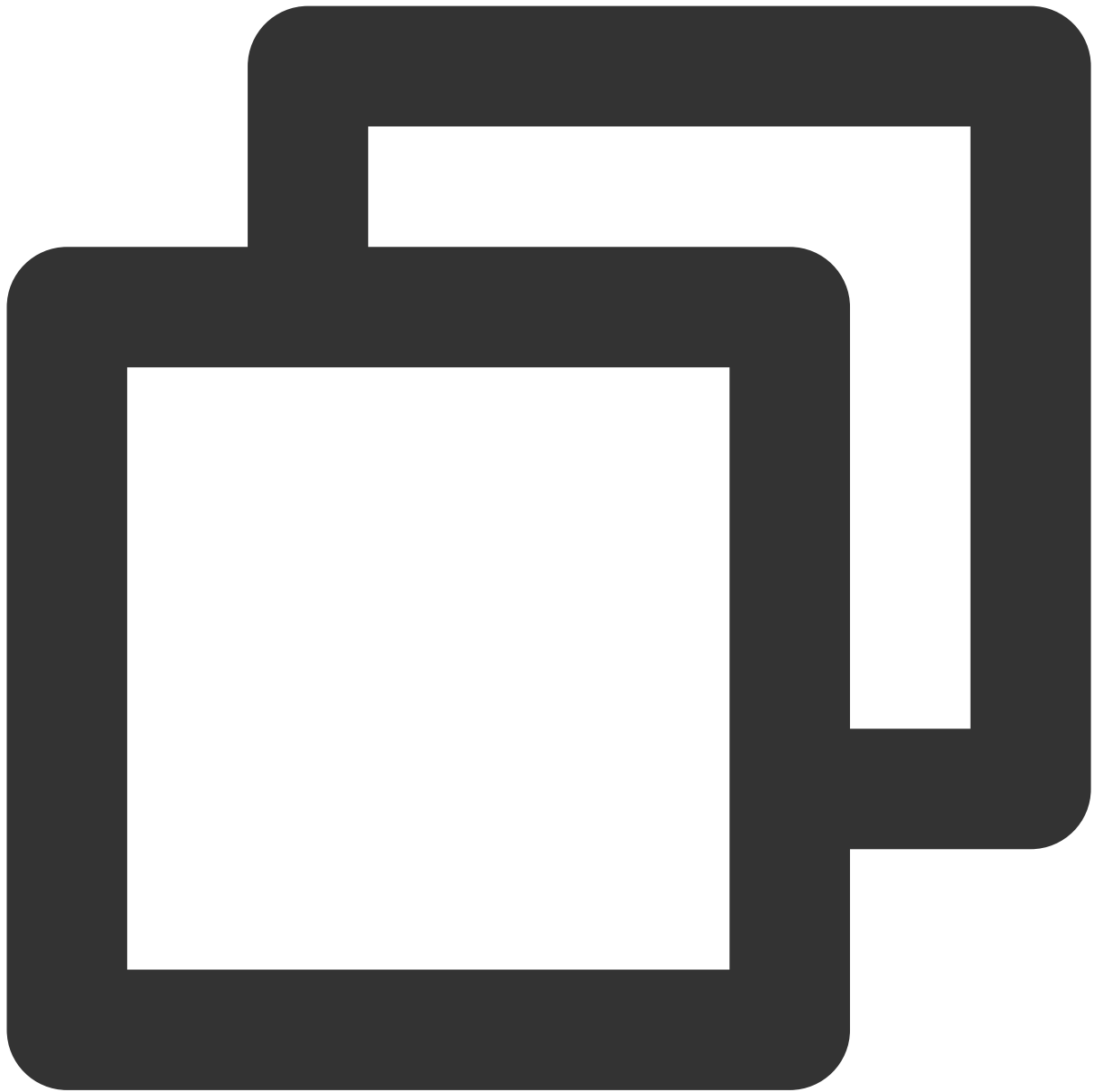


```
void setBeautyLevel(int beautyStyle, int beautyLevel);
```

Parameter	Type	Meaning
beautyStyle	int	beauty style
beautyLevel	int	beauty level

**setWhitenessLevel**

Set whiteness level



```
void setWhitenessLevel(int whitenessLevel);
```

Parameter	Type	Meaning
whitenessLevel	int	whiteness level

## callExperimentalAPI

Call experimental api



```
void callExperimentalAPI(String jsonStr);
```

Parameter	Type	Meaning
jsonStr	String	Api infomation

# Type Definition

Last updated : 2023-11-21 15:17:59

## Enumeration Definition

### TUIRoomDefine

Type	Description
<a href="#">TUIRoomType</a>	Room Type
<a href="#">TUISpeechMode</a>	Mic Control Mode
<a href="#">TUIMediaDevice</a>	Room Media Device Type
<a href="#">TUIRole</a>	Room Role Type
<a href="#">TUIVideoQuality</a>	Video Quality
<a href="#">TUIAudioQuality</a>	Audio Quality
<a href="#">TUIVideoStreamType</a>	Video Stream Type
<a href="#">TUIChangeReason</a>	Change Reason (User audio and video status change operation reason: self-modification or modified by room owner/administrator)
<a href="#">TUICaptureSourceType</a>	Screen Sharing Capture Source Type
<a href="#">TUIRequestAction</a>	Request Type
<a href="#">TUIResolutionMode</a>	Video resolution mode (landscape or portrait)

### TUICommonDefine

Type	Description
<a href="#">TUIError</a>	Error Code
<a href="#">TUINetworkQuality</a>	Network Quality

## Common Structure

### TUIRoomDefine

Type	Description

<a href="#">TUIRoomInfo</a>	Room data
<a href="#">TUILoginUserInfo</a>	User Login Information
<a href="#">TUIUserInfo</a>	Room User Information
<a href="#">TUISeatInfo</a>	Room Seat Information
<a href="#">TUISeatLockParams</a>	Lock Seat Operation Parameters
<a href="#">TUIUserVoiceVolume</a>	Room User Volume
<a href="#">TUIRequest</a>	Signaling Request
<a href="#">TUIActionCallback</a>	User Operation Callback
<a href="#">TUIPlayCallback</a>	Video Playback Callback
<a href="#">TUIRequestCallback</a>	User Request Callback
<a href="#">TUIUserListResult</a>	User List Information
<a href="#">TUIUserVoiceVolume</a>	User Volume Information
<a href="#">TUIValueCallBack&lt;T&gt;</a>	With Return Value (T) Callback
<a href="#">TUIRoomVideoEncoderParams</a>	Video encoder params

## TUICommonDefine

Type	Description
<a href="#">TUINetwork</a>	Network Quality Information
<a href="#">TUIMessage</a>	Message
<a href="#">TUIImageBuffer</a>	Image Info

## TUIRoomType

### Room Type

Enumeration	Value	Description
conference	1	Conference Type Room, suitable for conference and education scenarios, this room can enable free speech, apply for speech, go live and other modes.
livingRoom	2	Live Type Room, suitable for live broadcast scenarios, this room can

enable free speech, mic control mode, and the seats in this room are numbered.

## TUISpeechMode

### Mic Control Mode

Enumeration	Value	Description
freeToSpeak	1	Free speech mode
applyToSpeak	2	Apply to speak mode. (Only effective in conference type room)
applySpeakAfterTakingSeat	3	Go Live mode

### Explanation:

The relationship between Room Type, Mic Control Mode, and Going Live (takeSeat)

Room Type	Mic Control Mode		
	freeToSpeak	applyToSpeak	applySpeakAfterTakingSeat
conference	Not Supported	Not Supported	Need to Apply to the Host/Admin (takeSeat), Can Speak with Mic/Camera After Approval
livingRoom	Can Freely Go Live	Not Supported	Need to Apply to the Host/Admin (takeSeat), Can Speak with Mic/Camera After Approval

## TUIMediaDevice

### Room Media Device Type

Enumeration	Value	Description
microphone	1	Mic
camera	2	Camera
screen	3	Screen Sharing



## TUIRole

### Room Role Types

Enumeration	Value	Description
roomOwner	0	Room Owner. Generally refers to the creator of the room, the highest authority holder in the room
administrator	1	Room Administrator
generalUser	2	General Member in the room

## TUIVideoQuality

### Video Quality

Enumeration	Value	Description
videoQuality_360P	1	Low Definition 360P
videoQuality_540P	2	Standard Definition 540P
videoQuality_720P	3	High Definition 720P
videoQuality_1080P	4	Ultra Definition 1080P

## TUIAudioQuality

### Audio Quality

Enumeration	Value	Description
audioProfileSpeech	0	Vocal Mode
audioProfileDefault	1	Default Mode
audioProfileMusic	2	Music Mode

## TUIVideoStreamType

### Video Stream Types

Enumeration	Value	Description
cameraStream	0	HD Camera Video Stream
screenStream	1	Screen Sharing Video Stream

cameraStreamLow	2	Low Definition Camera Video Stream
-----------------	---	------------------------------------

## TUIChangeReason

Change Reason (User audio and video status change operation reason: self-initiated modification or modified by room owner/administrator)

Enumeration	Value	Description
changedBySelf	0	Self-operation
changedByAdmin	1	Room Owner or Administrator operation

## TUICaptureSourceType

Screen Sharing Capture Source Type

Enumeration	Value	Description
unknown	-1	Undefined
window	0	Window
screen	1	Screen

## TUIRequestAction

Request Type

Enumeration	Value	Description
invalidAction	0	Invalid request
requestToOpenRemoteCamera	1	Request remote user to open the camera
requestToOpenRemoteMicrophone	2	Request remote user to open the microphone
requestToConnectOtherRoom	3	Request to connect to another room
requestToTakeSeat	4	Request to go live
requestRemoteUserOnSeat	5	Request remote user to go live
applyToAdminToOpenLocalCamera	6	Request to the administrator to open the local camera
applyToAdminToOpenLocalMicrophone	7	Request to the administrator to open the local microphone

## TUIResolutionMode

Video resolution mode (landscape or portrait)

Enumeration	Value	Description
landscape	0	landscape
portrait	1	portrait

## TUIError

Error Code

Enumeration	Value	Description
success	0	Operation Successful
errFailed	-1	Temporarily Unclassified General Error
errFreqLimit	-2	Request Limited, Please Try Again Later
errRepeatOperation	-3	Repeated operation, please check whether your interface call is repeated
errSDKAppIDNotFound	-1000	SDKAppID Not Found, Please Confirm Application Info in <a href="#">TRTC Console</a>
errInvalidParameter	-1001	Illegal Input Parameters When Calling API
errSdkNotInitialized	-1002	SDK Not Initialized
errPermissionDenied	-1003	No Operation Permission
errRequirePayment	-1004	Need to Open Extra Package for This Feature
errCameraStartFailed	-1100	Failed to Open Camera
errCameraNotAuthorized	-1101	Camera Not Authorized
errCameraOccupy	-1102	Camera Occupied
errCameraDeviceEmpty	-1103	No Camera Device Currently
errMicrophoneStartFailed	-1104	Microphone Opening Failed
errMicrophoneNotAuthorized	-1105	Microphone Not Authorized
errMicrophoneOccupy	-1106	Microphone Occupied

errMicrophoneDeviceEmpty	-1107	No Microphone Device Currently
errGetScreenSharingTargetFailed	-1108	Failed to Get Screen Sharing Target
errStartScreenSharingFailed	-1109	Failed to Start Screen Sharing
errRoomIdNotExist	-2100	Room Does Not Exist When Entering, or Maybe Closed
errOperationInvalidBeforeEnterRoom	-2101	Need to Enter the Room Before Using This Feature
errExitNotSupportedForRoomOwner	-2102	Room Owner Does Not Support Exit Operation, Conference Room Type: You can transfer the room owner first, then exit the room. LivingRoom Room Type: The room owner can only close the room
errOperationNotSupportedInCurrentRoomType	-2103	This Operation is Not Supported in the Current Room Type
errOperationNotSupportedInCurrentSpeechMode	-2104	This Operation is Not Supported in the Current Speech Mode
errRoomIdInvalid	-2105	Illegal Room ID Creation, Custom ID Must Be Printable ASCII Characters (0x20-0x7e), Up to 48 Bytes
errRoomIdOccupied	-2106	Room ID Already in Use, Please Choose Another Room ID
errRoomNameInvalid	-2107	Illegal Room Name, The Name Must Be Up to 30 Bytes, The Character Encoding Must Be UTF-8, If It Contains Chinese
errAlreadyInOtherRoom	-2108	The Current User is Already in Another Room, You Need to Exit the Room First to Join a New Room. A single roomEngine instance only supports the user entering one room. If you want to enter a different room, please exit the room first or use a new roomEngine instance
errUserNotExist	-2200	User Does Not Exist
errUserNotEntered	-2201	User is Not in the Current Room
errUserNeedOwnerPermission	-2300	Room Owner Permission Required for

		Operation
errUserNeedAdminPermission	-2301	Room Owner or Administrator Permission Required for Operation
errRequestNoPermission	-2310	No Permission for Signaling Request, For Example, Canceling an Invitation Not Initiated by Yourself
errRequestIdInvalid	-2311	Invalid Signaling Request ID or Already Processed
errMaxSeatCountLimit	-2340	Maximum Seat Exceeds Package Quantity Limit
errAlreadyInSeat	-2341	Current User is Already on the Seat
errSeatOccupied	-2342	The Current Seat is Already Occupied
errSeatLocked	-2343	The Current Seat is Locked
errSeatIndexNotExist	-2344	Seat Number Does Not Exist
errUserNotInSeat	-2345	Current User is Not on the Mic
errAllSeatOccupied	-2346	The Number of People on the Mic is Full
errOpenMicrophoneNeedSeatUnlock	-2360	The Current Seat Audio is Locked
errOpenMicrophoneNeedPermissionFromAdmin	-2361	Need to Apply to the Room Owner or Administrator to Open the Microphone
errOpenCameraNeedSeatUnlock	-2370	The Current Seat Video is Locked, The Room Owner Needs to Unlock the Seat Before Opening the Camera
errOpenCameraNeedPermissionFromAdmin	-2371	Need to Apply to the Room Owner or Administrator to Open the Camera
errSendMessageDisabledForAll	-2380	The Current Room Has Enabled Mute for All
errSendMessageDisabledForCurrent	-2381	In the Current Room, You Have Been Muted

## TUINetworkQuality

### Network Quality

Enumeration	Value	Description

qualityUnknown	0	Undefined
qualityExcellent	1	The Current Network is Very Good
qualityGood	2	The Current Network is Good
qualityPoor	3	The Current Network is Average
qualityBad	4	The Current Network is Poor
qualityVeryBad	5	The Current Network is Very Poor
qualityDown	6	The Current Network Does Not Meet the Minimum Requirements of TRTC

## TUIRoomInfo

Room data

Field	Type	Description
roomId	String	Room ID
roomType	<a href="#">TUIRoomType</a>	Room Type
ownerId	String	Host ID, Default is Room Creator (Read Only)
name	String	Room Name, Default is Room ID
speechMode	<a href="#">TUISpeechMode</a>	Room Speech Mode
createTime	int	Room Creation Time (Read Only)
memberCount	int	Number of Members in the Room (Read Only)
maxSeatCount	int	Maximum Seat Quantity (Only Supported When Entering the Room and Creating the Room)
isCameraDisableForAllUser	bool	Whether to Prohibit Opening the Camera (Optional Parameter When Creating a Room). Default Value: false
isMicrophoneDisableForAllUser	bool	Whether to Prohibit Opening the Microphone (Optional Parameter When Creating a Room). Default Value: false
isMessageDisableForAllUser	bool	Whether to Prohibit Sending Messages (Optional Parameter When Creating a Room). Default Value: false

enableCDNStreaming	bool	Whether to Enable CDN Live Streaming (Optional Parameter When Creating a Room, for Live Room Use). Default Value: false
cdnStreamDomain	String	Live Streaming Push Domain (Optional Parameter When Creating a Room, for Live Room Use). Default Value: Empty

## TUILoginUserInfo

### User Login Info

Field	Type	Meaning
userId	String	User ID
userName	String	User Name
avatarUrl	String	User Avatar URL
customInfo	Map<String, String>	Custom Information

## TUIUserInfo

### User Information in the Room

Field	Type	Description
userId	String	User ID
userName	String	User Name
avatarUrl	String	User Avatar URL
userRole	<a href="#">TUIRole</a>	User Role Type
hasAudioStream	bool	Whether There is an Audio Stream. Default Value: false
hasVideoStream	bool	Whether There is a Video Stream. Default Value: false
hasScreenStream	bool	Whether There is a Screen Sharing Stream. Default Value: false

## TUISeatInfo

### Seat Information in the Room

Field	Type	Description
-------	------	-------------

index	int	Seat Number
userId	String	User ID
isLocked	bool	Whether the Seat is Locked. Default Value: false
isVideoLocked	bool	Whether the Seat is Prohibited from Opening the Camera. Default Value: false
isAudioLocked	bool	Whether the Seat is Prohibited from Opening the Microphone. Default Value: false

## TUISeatLockParams

Lock Seat Operation Parameters

Field	Type	Meaning
lockSeat	bool	Lock Seat. Default Value: false
lockVideo	bool	Lock Seat Camera. Default Value: false
lockAudio	bool	Lock Seat Microphone. Default Value: false

## TUIUserVoiceVolume

User Volume in the Room

Field	Type	Description
userId	String	User ID
volume	int	Volume. Used to Carry the Volume of All Speaking Users, Value Range 0 - 100

## TUIRequest

Signaling Request

Field	Type	Description
requestId	String	Request ID
requestAction	<a href="#">TUIRequestAction</a>	Request Type
userId	String	User ID
content	String	Signaling Content



timestamp	int	Timestamp
-----------	-----	-----------

## TUIActionCallback

Field	Type	Description
code	<a href="#">TUIError</a>	Error Code
message	String?	Error Information

## TUIPlayCallback

Field	Type	Description
onPlaying	(String userId) {}	Playing Callback
onLoading	(String userId) {}	Loading Callback
onPlayError	(String userId, TUIError code, String message) {}	Playback Error Callback

## TUIRequestCallback

Field	Type	Description
onAccepted	(String requestId, String userId) {}	Request Accepted Callback
onRejected	(String requestId, String userId, String message) {}	Request Rejected Callback
onCancelled	(String requestId, String userId) {}	Request Cancelled Callback
onTimeout	(String requestId, String userId) {}	Request Timeout Callback
onError	(String requestId, String userId, TUIError error, String message) {}	Request Error Callback

## TUIUserListResult

Field	Type	Description
nextSequence	int	Pagination Fetch Flag, if the returned nextSequence is not zero, you need to use the returned nextSequence to fetch again until it returns 0

userInfoList	List<TUIUserInfo>	The user list returned by this call
--------------	-------------------	-------------------------------------

## TUIUserVoiceVolume

Field	Type	Description
userId	String	User ID
volume	int	User Volume Size

## TUIValueCallBack<T>

Field	Type	Description
code	<a href="#">TUIError</a>	Error Code
message	String?	Error Message
data	T?	Return Data, example: if T is TUIUserInfo, then the data field type of TUIValueCallBack<TUIUserInfo> is TUIUserInfo

## TUIRoomVideoEncoderParams

Video encoder params

Field	Type	Description
videoResolution	<a href="#">TUIVideoQuality</a>	Video resolution
resolutionMode	<a href="#">TUIResolutionMode</a>	Video resolution mode
fps	int	Video capturing frame rate
bitrate	int	Target video bitrate

## TUINetwork

Network Quality Information

Field	Type	Description
userId	String	User ID
quality	<a href="#">TUINetworkQuality</a>	Network Quality
upLoss	int	Packet Loss Rate for Upstream

downLoss	int	Packet Loss Rate for Downstream
delay	int	Network Delay

## TUIMessage

### Message

Field	Type	Description
messageId	String	Message ID
message	String	Message Text
timestamp	int	Message Timestamp
userId	String	Message Sender
userName	String	Message Sender Nickname
avatarUrl	String	Message Sender Avatar

## TUIImageBuffer

### Image Info

Field	Type	Description
buffer	String	Image Data Cache Address
length	int	Length
width	int	Width
height	int	Height

# Server APIs (TUIRoomKit)

## REST API

### RESTful API Overview

Last updated : 2024-06-12 11:46:26

The RESTful API is a part of the TRTC Conference backend HTTP Management Interface, providing developers with a simplified management entry.

For security reasons, the RESTful API is only available via HTTPS Interface.

### Prerequisites

To call the RESTful API, you need purchase or acquire the necessary plan for RoomSdk.

#### **Warning:**

The new version of the RESTful API has undergone significant optimization and upgrades in backend architecture, providing enhanced capabilities. Unfortunately, this means it currently cannot interoperate with the 1.x version of the SDK.

Therefore, when integrating the latest RESTful API, please ensure to use the client SDK of the latest version 2.0 or above.

### Calling Method

#### **Request URL**

The URL format of the RESTful API is as follows:



```
https://console.tim.qq.com/$ver/$servicename/$command?sdkappid=$SDKAppID&identifier
```

The meanings and values of each parameter are as follows (both parameter names and their values are case-sensitive):

Parameter	Meaning	Fetching Value
https	Request protocol	The request protocol is HTTPS, and the request method is POST
console.tim.qq.com	Request domain	Fixed as <code>console.tim.qq.com</code>

	name	
ver	Protocol version number	Fixed as <code>v4</code>
servicename	Internal service name, different service names correspond to different service types	Example: <code>v4/room_engine_http_srv/create_room</code> , where <code>room_engine_http_srv</code> is the <code>servicename</code> For more details, please refer to the <a href="#">REST API Interface List</a> .
command	The word <code>command</code> , combined with the <code>servicename</code> , is used to indicate a specific business feature	Example: <code>v4/room_engine_http_srv/create_room</code> , where <code>create_room</code> is the <code>command</code> For more details, please refer to the <a href="#">REST API Interface List</a> .
sdkappid	The application identifier accessed in the Chat console	Obtained when applying for integration
identifier	username, must be an App Administrator Account when calling RESTful APIs	Refer to <a href="#">App Administrator</a>
usersig	password corresponding to username	Refer to <a href="#">Generating UserSig</a>
random	Identifies the random number parameter for the current request	32-bit unsigned integer random number, ranging from 0 to 4294967295
contenttype	Request format	Fixed value: <code>json</code>

### Note

When calling RESTful APIs, the App Server's identifier must be an App Administrator Account.

An App can generate a UserSig for the Administrator Account each time it calls a RESTful API, or it can generate a fixed UserSig for repeated use, but please pay special attention to the validity period of the UserSig.

### HTTP Request Body Format

The RESTful API only supports the POST method, and its request body is in JSON format. For specific body formats, refer to the detailed description of each API.

It's particularly important that the POST body cannot be empty. Even if a protocol doesn't require any information to be carried, an empty JSON object, namely `{ }`, must be included.

## HTTP Return Code

Unless a network error occurs (e.g., a 502 error), the call result of RESTful APIs is always 200. The actual error code and error message of the API call are returned in the HTTP response body.

## HTTP Response Body Format

The response body of the RESTful API is also in JSON format, and its format conforms to the following characteristics:



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-70e312f1de024af5a36714b7b71da224-O-Seq-63504"
  // Other response content of REST API
}
```

The response body must contain four attributes: ActionStatus, ErrorInfo, ErrorCode, RequestId. Their meanings are as follows:

--	--	--



Field	Type	Description
ActionStatus	String	The result of the request processing. OK for success, FAIL for failure. If it's FAIL, ErrorInfo will provide the reason for failure.
ErrorInfo	String	Cause of failure
ErrorCode	Integer	Error code. 0 for success, others for failure. You can refer to the <a href="#">Error Code Table</a> for specific reasons.
RequestId	String	Error code. 0 for success, others for failure. You can refer to the <a href="#">Error Code Table</a> for specific reasons.

## Sample call

Below is an example of [getting all groups in an app](#) through RESTful APIs.

HTTPS Request:



```
POST /v4/group_open_http_svc/get_appid_group_list?usersig=xxx&identifier=admin&sdka
Host: console.tim.qq.com
Content-Length: 22
{
  "Limit": 2
}
```

HTTPS Response:



```
HTTP/1.1 200 OK
Server: nginx/1.7.10
Date: Fri, 09 Oct 2015 02:59:55 GMT
Content-Length: 156
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Allow-Methods: POST

{
  "ActionStatus": "OK",
```

```
"ErrorCode": 0,
"GroupIdList": [
  {
    "GroupId": "@TGS#1YTTZEAEg"
  },
  {
    "GroupId": "@TGS#1KVTZEAEZ"
  }
],
"TotalCount": 58530
}
```

## Common Error Codes for RESTful APIs

Error code	Description
60002	An error occurred when parsing the HTTP request. Check the format of the HTTP request URL.
60003	An error occurred when parsing the JSON data of the HTTP request. Check the JSON format.
60004	Account or signature in the request URL or JSON packet is incorrect.
60005	Account or signature in the request URL or JSON packet is incorrect.
60006	Invalid SDKAppID. Check the validity of the SDKAppID.
60007	The RESTful API call exceeds the frequency limit. Please reduce the request frequency.
60008	The service request timed out or the format of the HTTP request is incorrect. Please check and try again.
60009	Request resource error. Please check the request URL.
60010	The request requires App administrator permission.
60011	The SDKAppID request exceeds the frequency limit. Please reduce the request frequency.
60012	SDKAppID is required when calling the RESTful API. Check the SDKAppID in the request URL.
60013	An error occurred when parsing the JSON data in the HTTP response packet.
60014	Account switching timed out.
60015	The type of the account in the request packet is incorrect. Please ensure that the UserID is in string format.

60016	The SDKAppID is disabled.
60017	The request is disabled.
60018	Too many requests. Try again later.
60019	Too many requests. Try again later.
60020	Your professional edition plan has expired and been disabled, please log in to <a href="#">Chat Purchase Page</a> to repurchase the plan. It will take effect 5 minutes after purchase.
60021	The source IP of the RESTful API call is invalid.

## FAQs

### Is there a chance that RESTful API requests timeout, receiving no response?

1. The Room backend RESTful API's timeout setting is 3s, the caller's timeout setting should be longer than 3s.
2. `telnet console.tim.qq.com 443` to confirm if the service port can be connected.
3. Use `curl -I https://console.tim.qq.com` to simply test if the status code is 200.
4. Confirm whether the machine's DNS server configuration is an internal DNS server or a public DNS server. If it is an internal DNS server, ensure that the DNS server's network egress and the region ISP of the machine's network egress IP match.
5. It is recommended for business callers to use the **Long Connection + Connection Pool** pattern.

#### Note

Due to the high time consumption of establishing HTTPS short connections, each request incurs TCP + TLS handshake overhead, so it is advised for RESTful API to use long connections.

In the scenario of using standard HTTP libraries: For HTTP1.0, it's necessary to specify the request header `Connection: keep-alive`; for HTTP1.1, long connections are supported by default. In the scenario of encapsulating HTTPS requests based on TCP, the TCP connection can be reused for sending and receiving requests.

# RESTful API List

Last updated : 2024-06-12 11:46:26

## Room Management

Feature	API
Create a Room	<a href="#">room_engine_http_srv/create_room</a>
Destroy a Room	<a href="#">room_engine_http_srv/destroy_room</a>
Update the Room Information	<a href="#">room_engine_http_srv/update_room_info</a>
Get the Room Information	<a href="#">room_engine_http_srv/get_room_info</a>

## User Management

Feature	API
Get the Room Member List	<a href="#">room_engine_http_srv/get_room_member_list</a>
Update the Room Member Information	<a href="#">room_engine_http_srv/update_room_member_info</a>
Change the Room Ownership	<a href="#">room_engine_http_srv/change_room_owner</a>
Mark Room Members	<a href="#">room_engine_http_srv/mark_room_member</a>
Ban Room Members	<a href="#">room_engine_http_srv/ban_room_member</a>
Unban Room Members	<a href="#">room_engine_http_srv/unban_room_member</a>
Get the Banned Room Member List	<a href="#">room_engine_http_srv/get_banned_member_list</a>

## Seat Management

Feature	API
Get the Seat List	<a href="#">room_engine_http_mic/get_seat_list</a>
Pick User on the Seat	<a href="#">room_engine_http_mic/pick_user_on_seat</a>

Kick User off the Seat	<a href="#">room_engine_http_mic/kick_user_off_seat</a>
Lock the Seat	<a href="#">room_engine_http_mic/lock_seat</a>

# Room Management

## Create a Room

Last updated : 2024-06-12 11:46:26

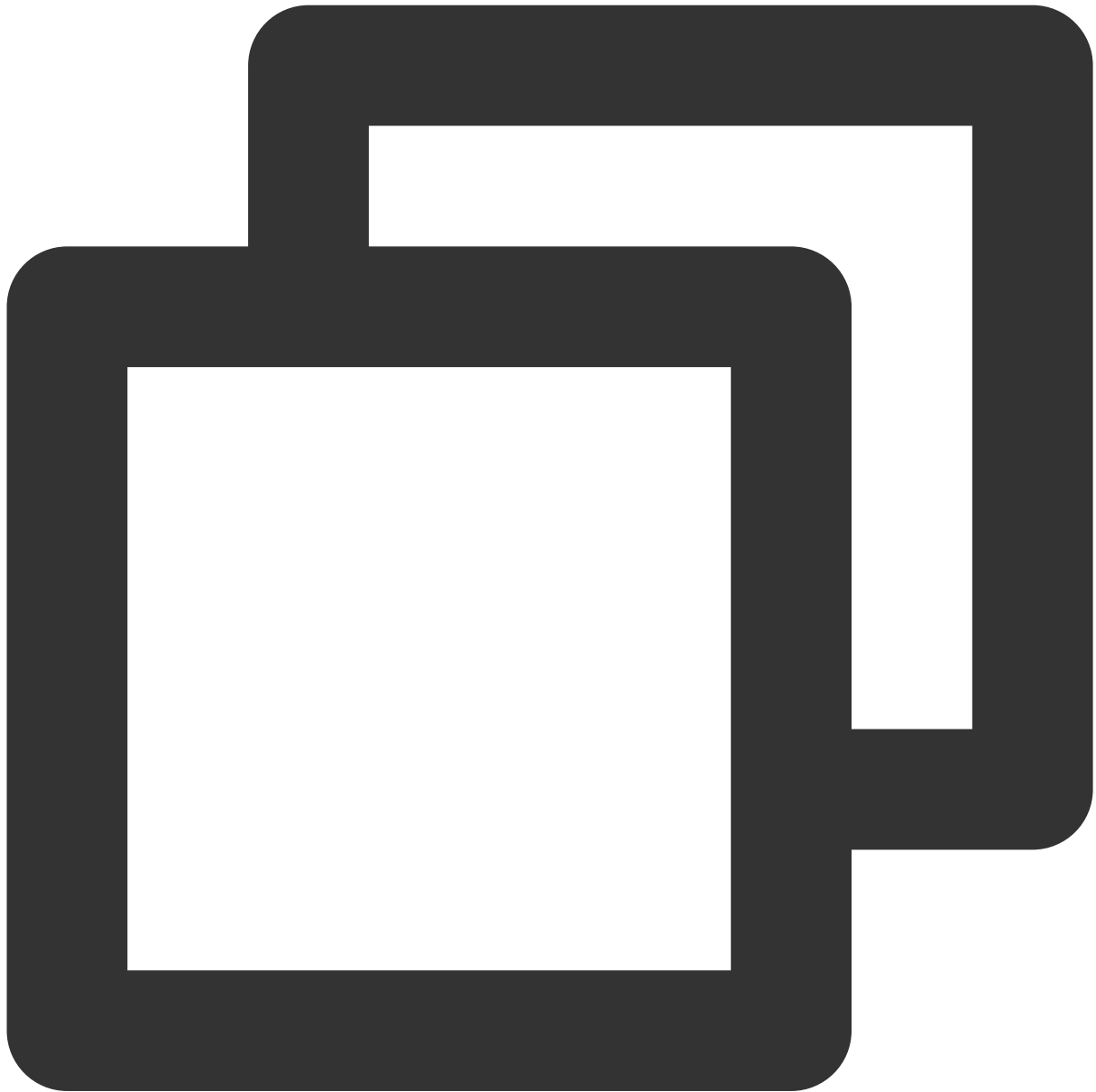
### Feature Overview

App administrators can create rooms and schedule meetings through this API.

### API Calling Description

#### Sample Request URL





```
https://xxxxxx/v4/room_engine_http_srv/create_room?sdkappid=88888888&identifier=adm
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

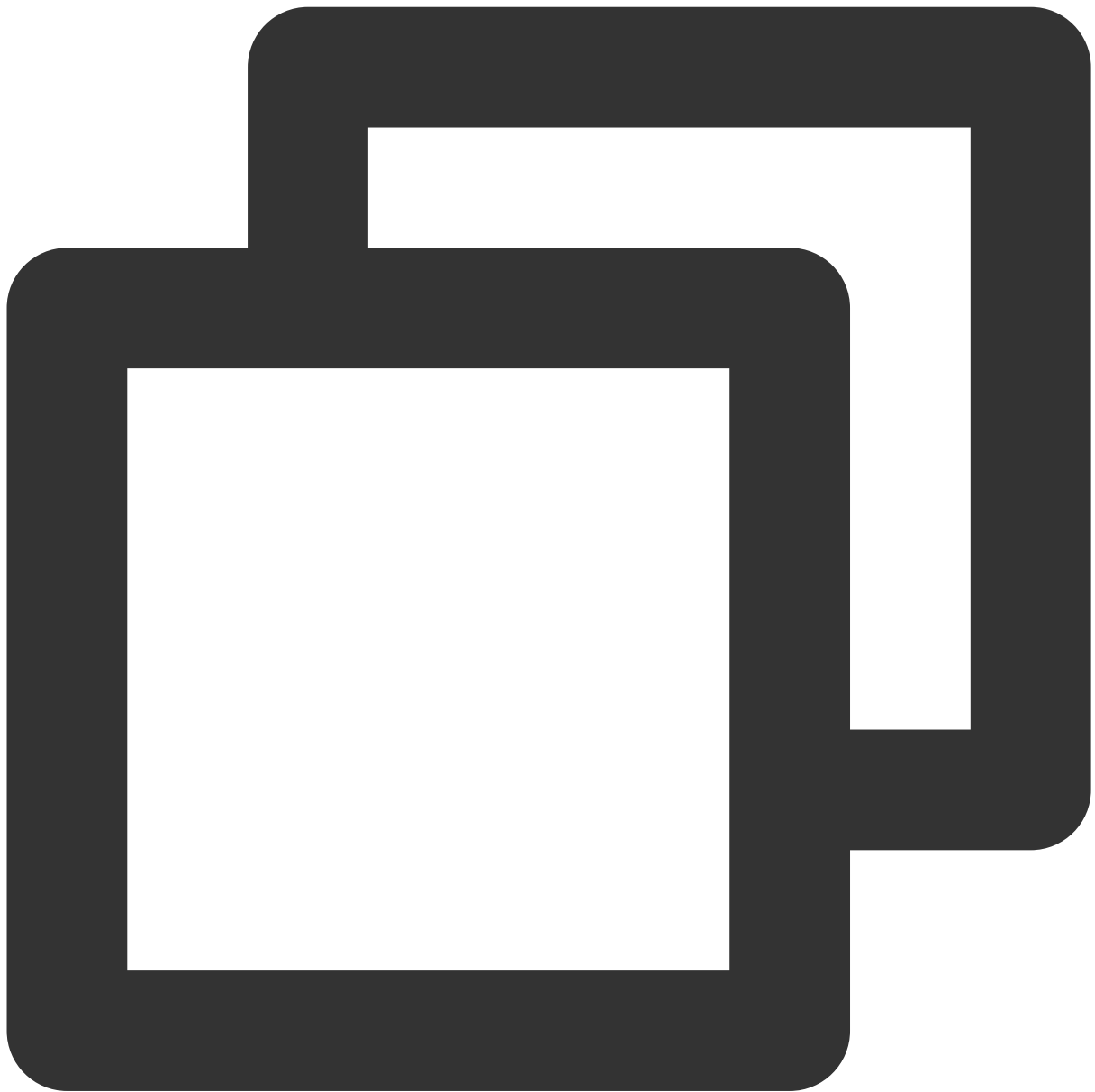
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/create_room	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomInfo": {
    "RoomId": "room-test",
    "RoomName": "room-name-test",
    "RoomType": "Conference",
    "Owner_Account": "user2",
    "MaxMemberCount": 300,
    "ScheduleStartTime" : 1693271355,
    "ScheduleEndTime" : 1693272355,
    "IsVideoDisabled": true,
    "IsAudioDisabled": true,
  }
}
```

```

    "IsMessageDisabled": true,
    "IsScreenSharingDisabled": true,
    "IsCloudRecordingDisabled": true,
    "CustomInfo": "custom123",
    "IsSeatEnabled": true,
    "MaxSeatCount": 16,
    "TakeSeatMode": "ApplyToTake"
  },
  "ScheduleInviteeList_Account": [
    "user1", "user2", "user3"
  ]
}

```

## Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID, up to 48 bytes
RoomName	String	Optional	Room Name, defaults to Room ID, up to 100 bytes
RoomType	String	Mandatory	Room Type: Conference (Meeting Room)
Owner_Account	String	Optional	Host ID (must be an <a href="#">imported</a> account), defaults to the User ID of the API caller
MaxMemberCount	Integer	Optional	Maximum number of room members, default value when not specified: Upper limit of paid package, for example, the trial version is 20, if upgrading the package, this field must be updated according to modify room information.
ScheduleStartTime	Integer	Optional	Scheduled meeting start time, the default is current time.
ScheduleEndTime	Integer	Optional	Scheduled meeting end time, default is 1 hour after the start time, the minimum meeting time cannot be less than 5 minutes, and the maximum time cannot exceed 24h.
IsVideoDisabled	Bool	Optional	Mute all video, default false
IsAudioDisabled	Bool	Optional	Mute all audio, default false
IsMessageDisabled	Bool	Optional	Disable all members from sending text messages, default false

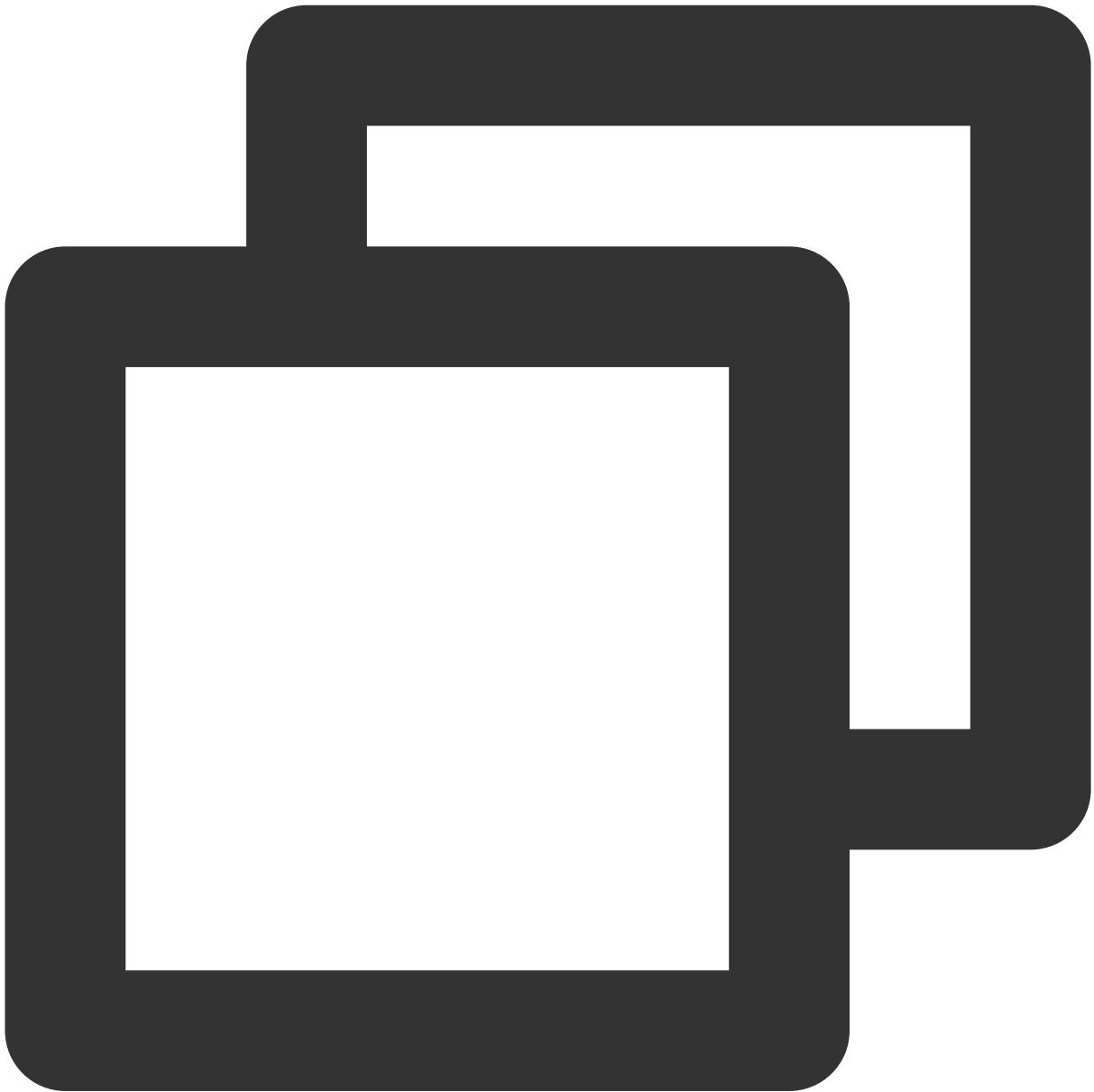
IsScreenSharingDisabled	Bool	Optional	Disable screen sharing, default false
IsCloudRecordingDisabled	Bool	Optional	Disable cloud recording, default false
CustomInfo	String	Optional	Custom Information, maximum 500 bytes
IsSeatEnabled	Bool	Optional	Whether to support seat position. Not supported by default.
MaxSeatCount	Integer	Optional	Maximum number of seats, default is the package limit (max 20 by default).
TakeSeatMode	String	Optional	Seat mode: FreeToTake (open mic), ApplyToTake (mic on request)
ScheduleInviteeList_Account	Array	Optional	Scheduled member list, up to 300 members

**Note:**

After the meeting ends, users need to manually call the [Destroy Room](#) interface to end the meeting.

If the Destroy Room interface is not called manually, the backend will try to recycle the room 6 hours after the meeting ends. The condition for recycling is that no members left in the room.

**Sample Response Packets****Basic Form**



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success; FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned for each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error Code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100003	The room ID already exists, please choose another room ID.
100007	No payment information, you need to purchase a package in the console.
100010	The room ID has been used, and the operator is the home owner, you can use it directly.
100011	The room ID has been used by Chat, you can change to another room ID or destroy the group through the Chat interface.
100012	The frequency limit for creating a room has been exceeded. The same room ID can only be created once per second.

## Possible Callbacks

[After a Room Is Created](#)

# Destroy a Room

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can destroy through this API.

## API Calling Description

### Sample Request URL





```
https://xxxxxx/v4/room_engine_http_srv/destroy_room?sdkappid=88888888&identifier=ad
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

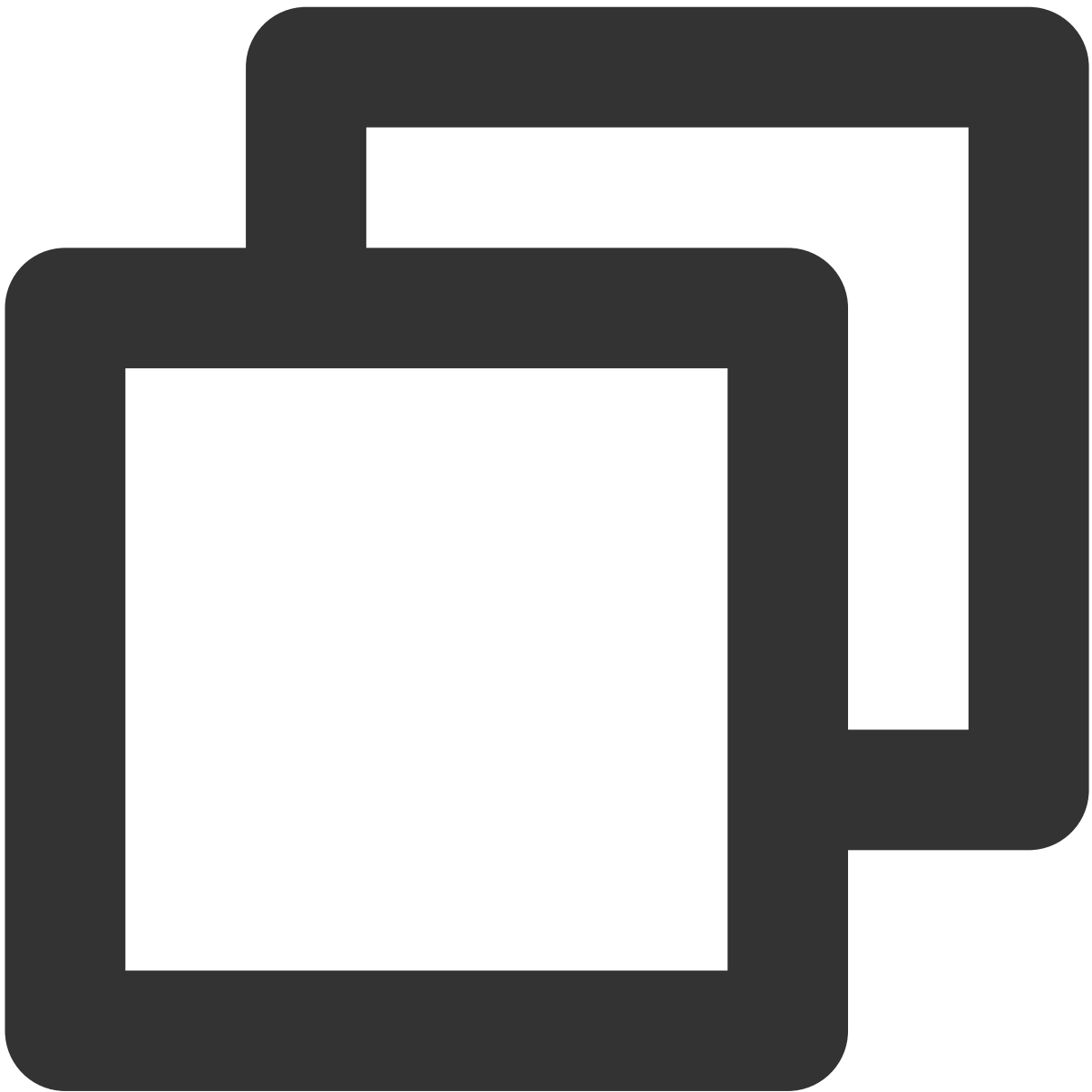
	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_engine_http_srv/destroy_room	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



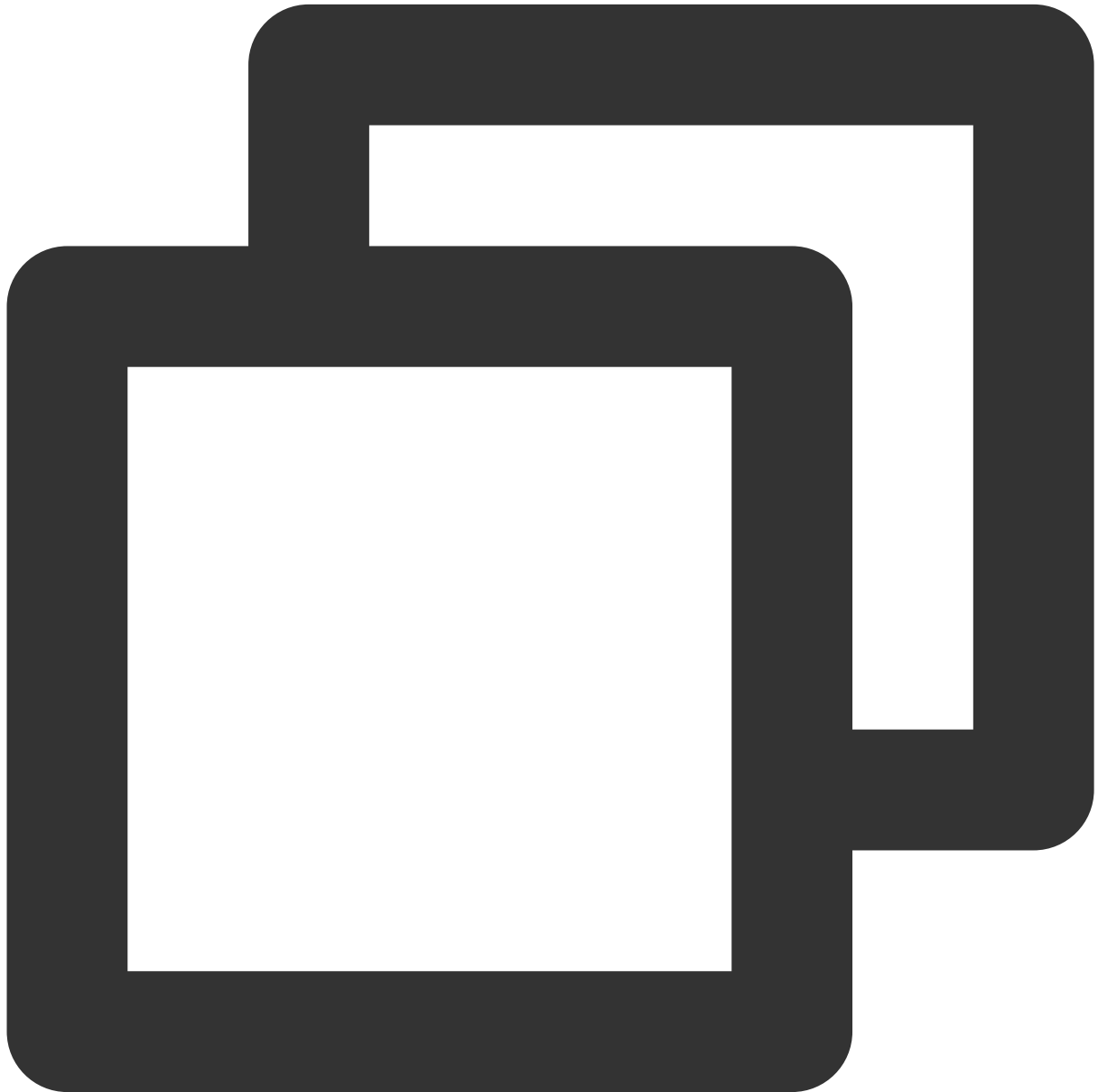
```
{
  "RoomId": "room-test"
}
```

Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

## Sample Response Packets

### Basic Form



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error Code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	The room does not exist, or it once existed but now has been destroyed.
100006	Insufficient operational permissions.

## Possible Callbacks

[After a Room Is Destroyed](#)

# Update the Room Information

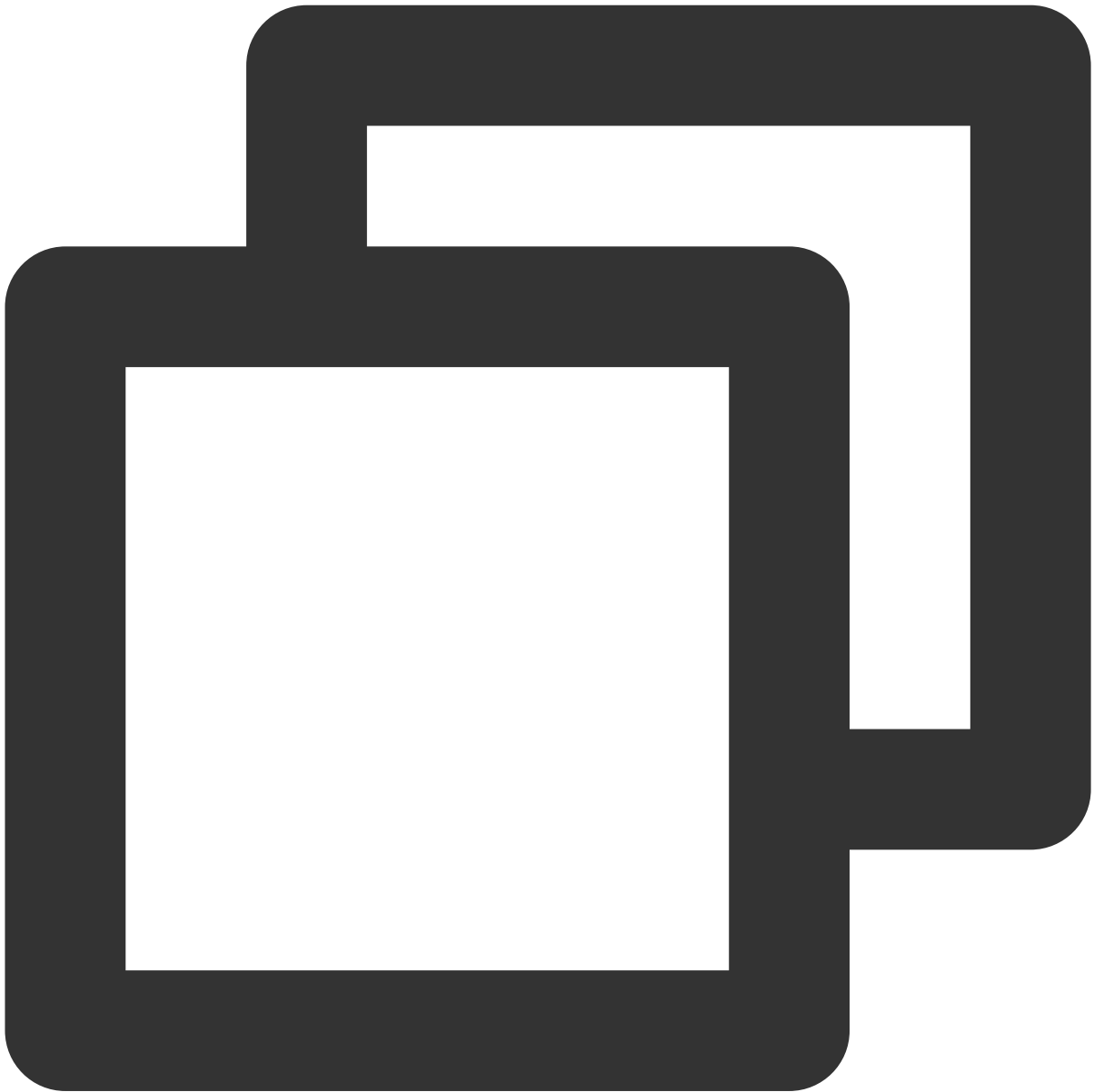
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can update room information through this API.

## API Calling Description

### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_srv/update\_room\_info?sdkappid=88888888&identifie

Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_engine_http_srv/update_room_info	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form





```
{
  "RoomInfo": {
    "RoomId": "room-test",
    "RoomName": "room-name-test",
    "MaxMemberCount": 300,
    "IsVideoDisabled": true,
    "IsAudioDisabled": true,
    "IsMessageDisabled": true,
    "IsScreenSharingDisabled": true,
    "IsCloudRecordingDisabled": true,
    "CustomInfo": "custom123",
  }
}
```

```
    "TakeSeatMode": "ApplyToTake"
  }
}
```

## Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
RoomName	String	Optional	Room Name, defaults to Room ID, up to 100 bytes
MaxMemberCount	Integer	Optional	Maximum number of room members
IsVideoDisabled	Bool	Optional	Mute all video
IsAudioDisabled	Bool	Optional	Mute all audio
IsMessageDisabled	Bool	Optional	Disable all members from sending text messages
IsScreenSharingDisabled	Bool	Optional	Disable screen sharing
IsCloudRecordingDisabled	Bool	Optional	Disable cloud recording
CustomInfo	String	Optional	Custom Information, maximum 500 bytes
TakeSeatMode	String	Optional	Seat mode: FreeToTake (open mic), ApplyToTake (mic on request)

## Sample Response Packets

### Basic Form



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned for each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

## Possible Callbacks

[After the Room Information Is Updated](#)

# Get the Room Information

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrator can get room information through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_engine_http_srv/get_room_info?sdkappid=88888888&identifier=a
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

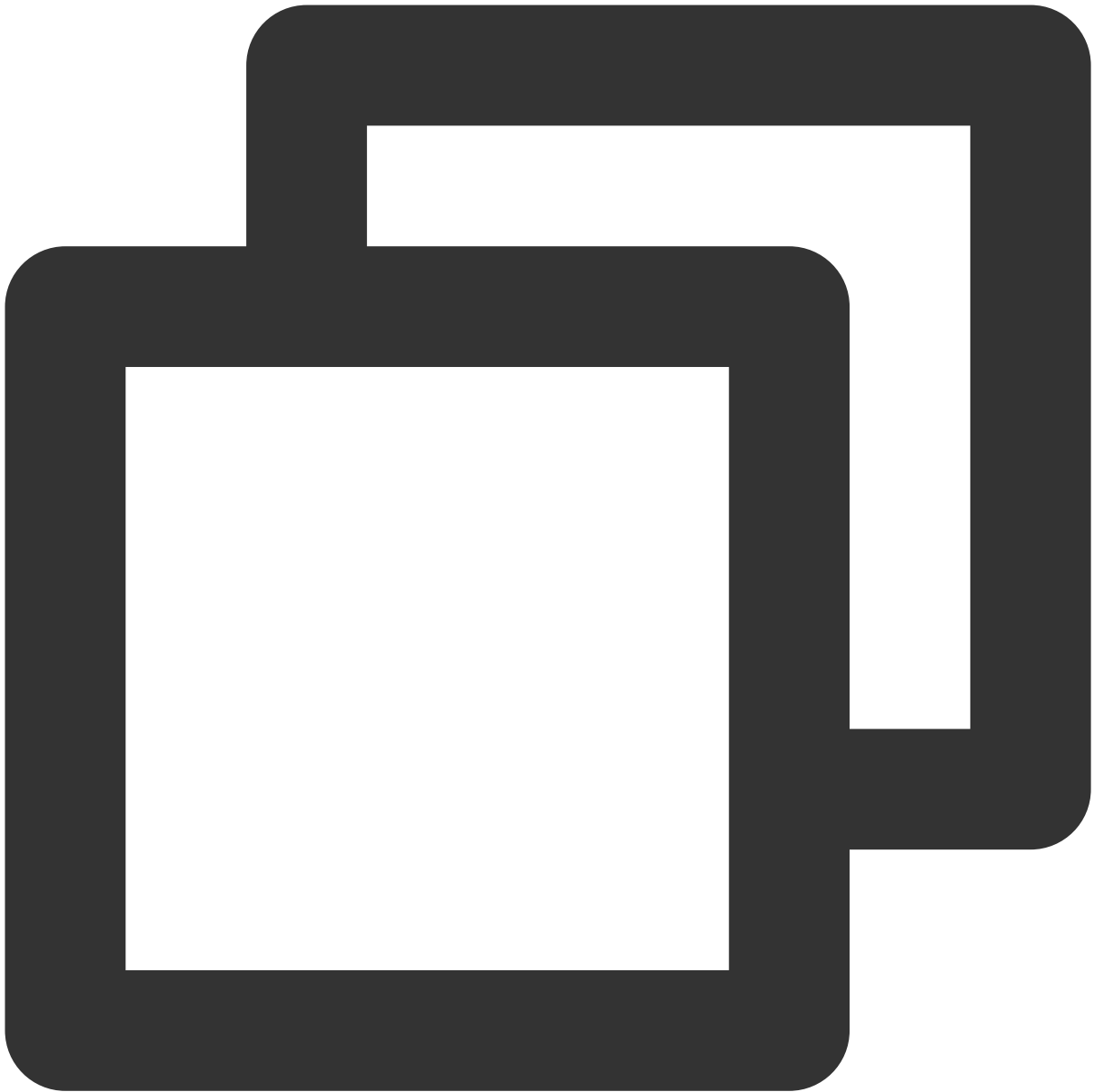
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/get_room_info	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test"
}
```

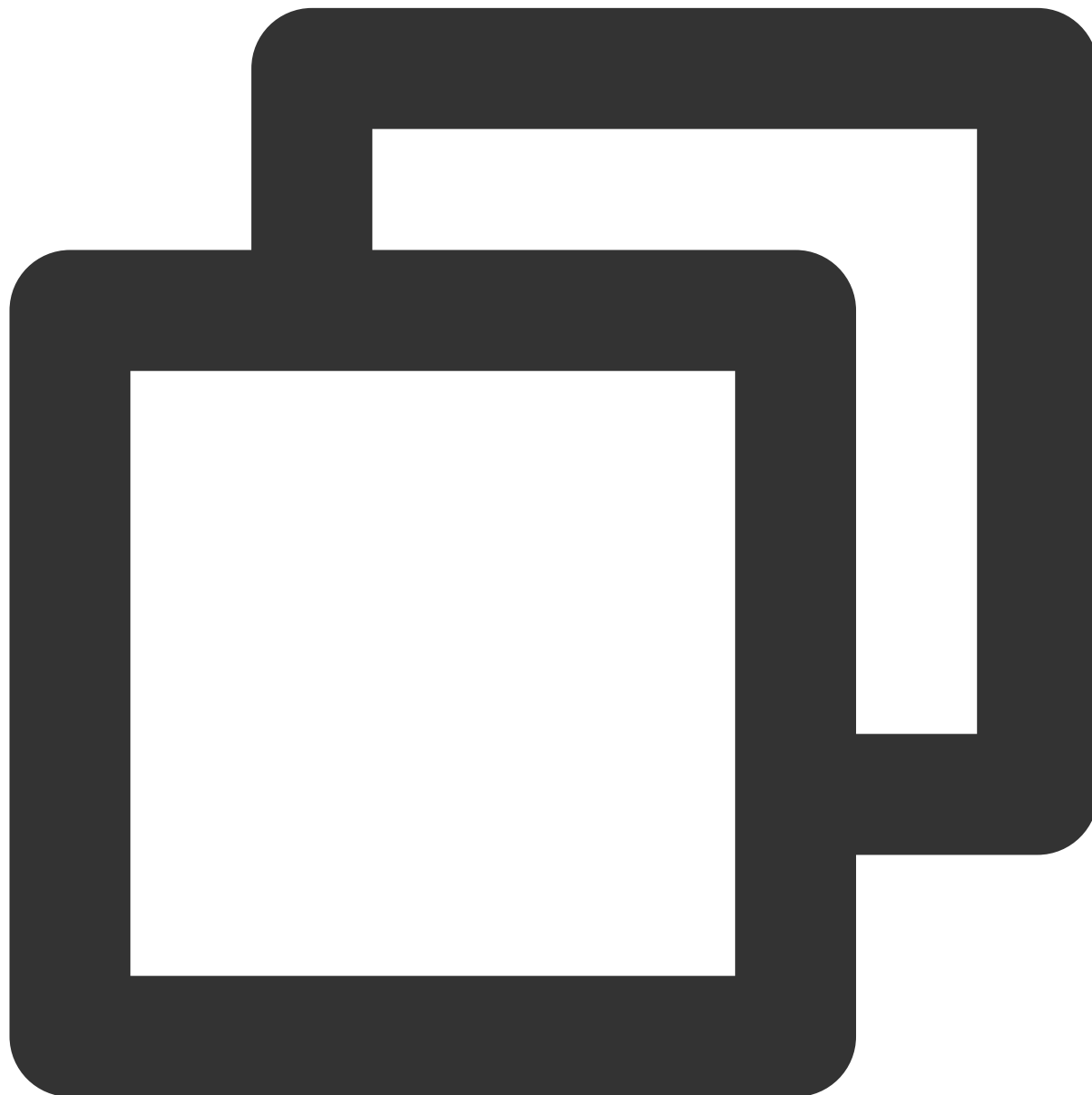
Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID



## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-81fb8ae1529f409a9ed83ef3c3071657-O-Seq-56057",
  "Response": {
    "RoomInfo": {
      "RoomId": "room-test",
```

```
    "RoomName": "room-name-test",
    "RoomType": "Conference",
    "Owner_Account": "user2",
    "MaxMemberCount": 300,
    "MaxSeatCount": 16,
    "IsVideoDisabled": true,
    "IsAudioDisabled": true,
    "IsMessageDisabled": true,
    "IsScreenSharingDisabled": true,
    "IsCloudRecordingDisabled": true,
    "CustomInfo": "custom123",
    "ScheduleStartTime": 1703491546,
    "ScheduleEndTime": 1703495146,
    "RoomStatus": "Running",
    "IsSeatEnabled": true,
    "TakeSeatMode": "ApplyToTake",
    "CreateTime": 1703491546,
    "MemberCount": 1
  }
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned for each request and required to provide this RequestId when locating issues.
RoomId	String	Room ID
RoomName	String	Room Name
RoomType	String	Room Type: Conference (Meeting Room)
Owner_Account	String	Host ID
MaxMemberCount	Integer	Maximum number of room members
ScheduleStartTime	Integer	Scheduled meeting start time
ScheduleEndTime	Integer	Scheduled meeting end time

IsVideoDisabled	Bool	Mute all video
IsAudioDisabled	Bool	Mute all audio
IsMessageDisabled	Bool	Disable all members from sending text messages
IsScreenSharingDisabled	Bool	Disable screen sharing
IsCloudRecordingDisabled	Bool	Disable cloud recording
CustomInfo	String	Custom Information
RoomStatus	String	Room Status: None, NotStarted, Running
IsSeatEnabled	Bool	Is microphone support available?
MaxSeatCount	Integer	Maximum Number of Microphones
TakeSeatMode	String	Seat Mode: None, FreeToTake (open mic), ApplyToTake (mic on request)
CreateTime	Integer	Scheduled meeting start time
MemberCount	Integer	Number of room members

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through `ErrorCode` and `ErrorInfo` in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions



# User Management

## Get the Room Member List

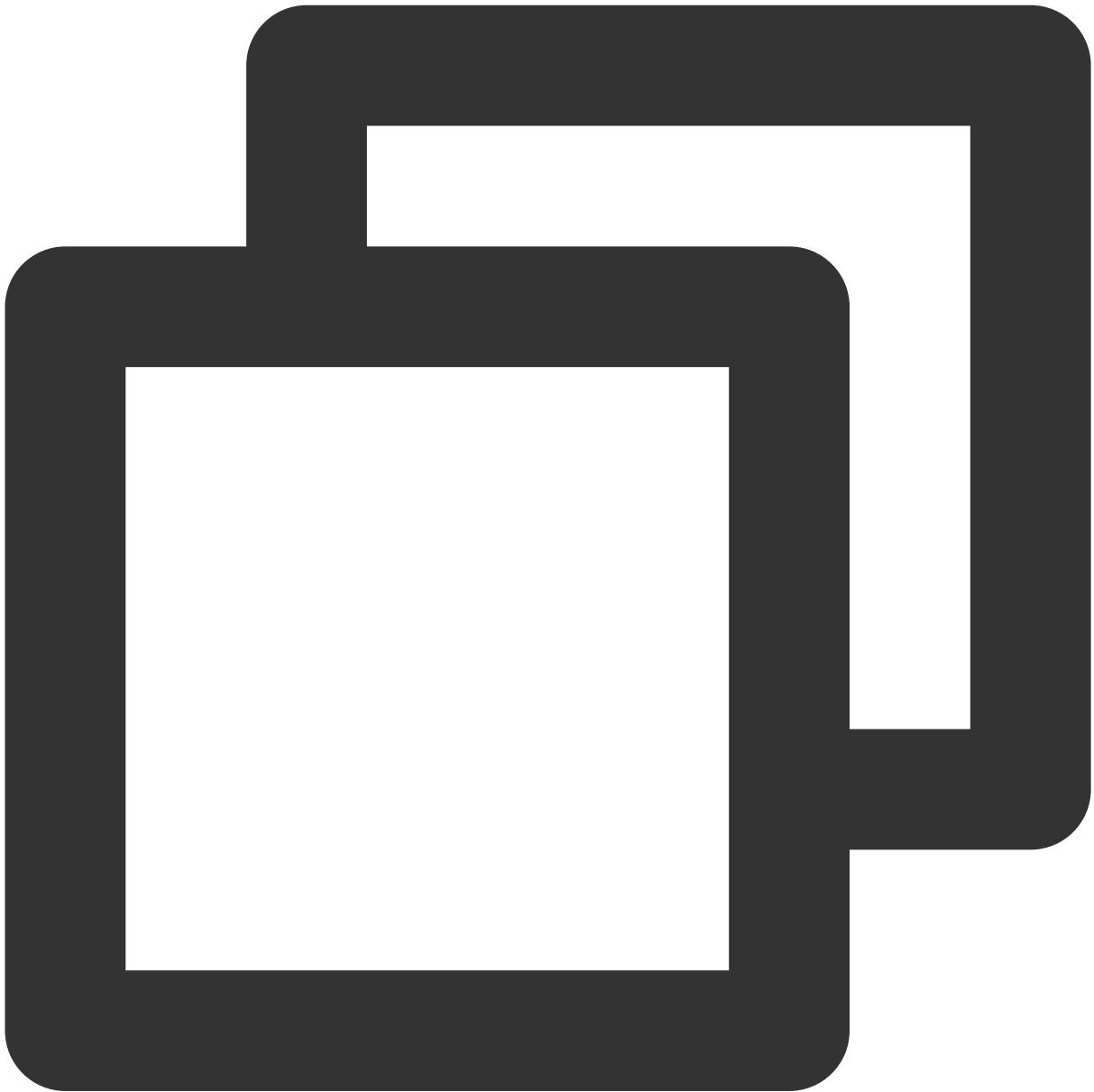
Last updated : 2024-06-12 11:46:26

### Feature Overview

App administrators can get the room member list through this API.

### API Calling Description

#### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_srv/get\_room\_member\_list?sdkappid=88888888&ident

Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

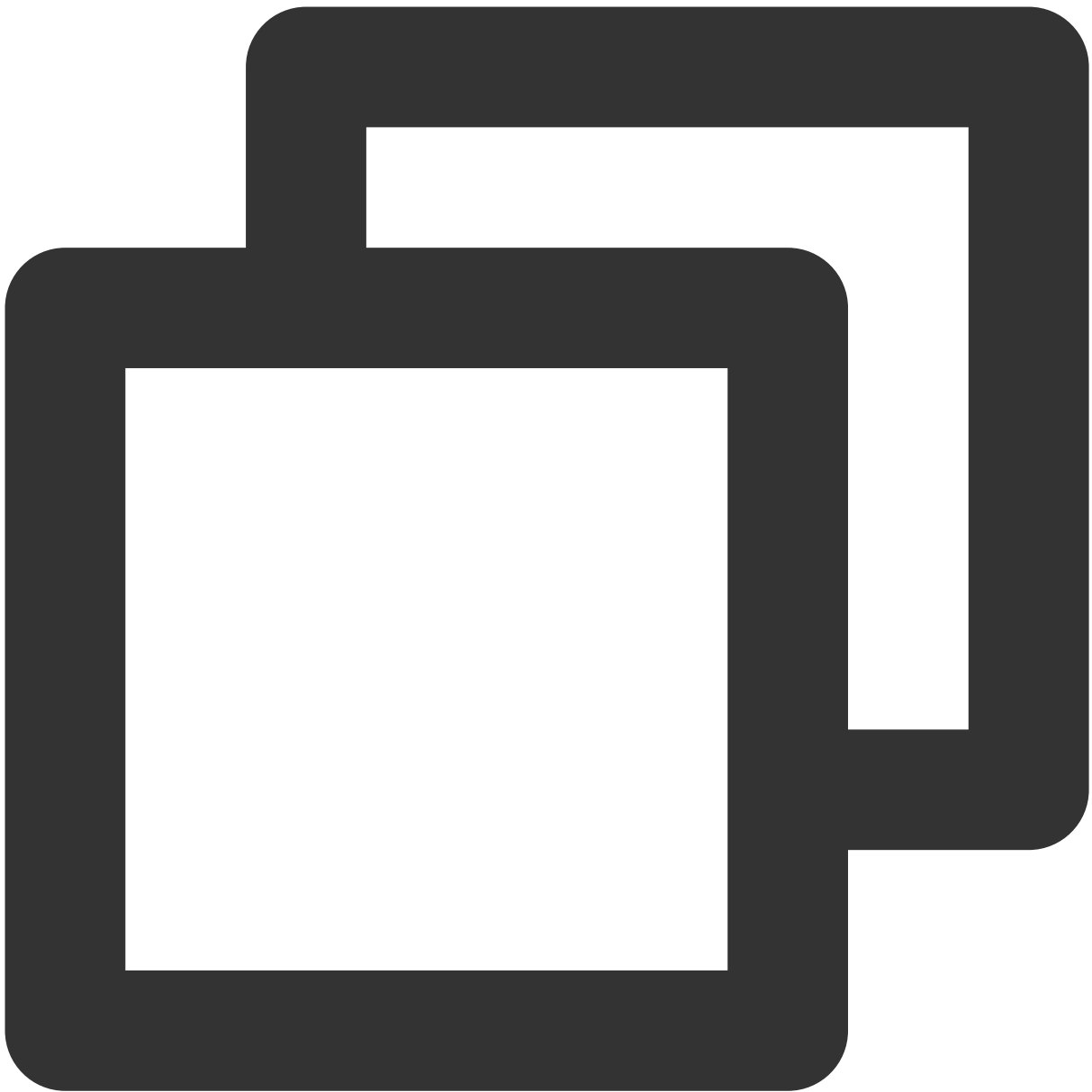
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/get_room_member_list	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "Count": 10,
  "Next": ""
}
```

Request Packet Field Description

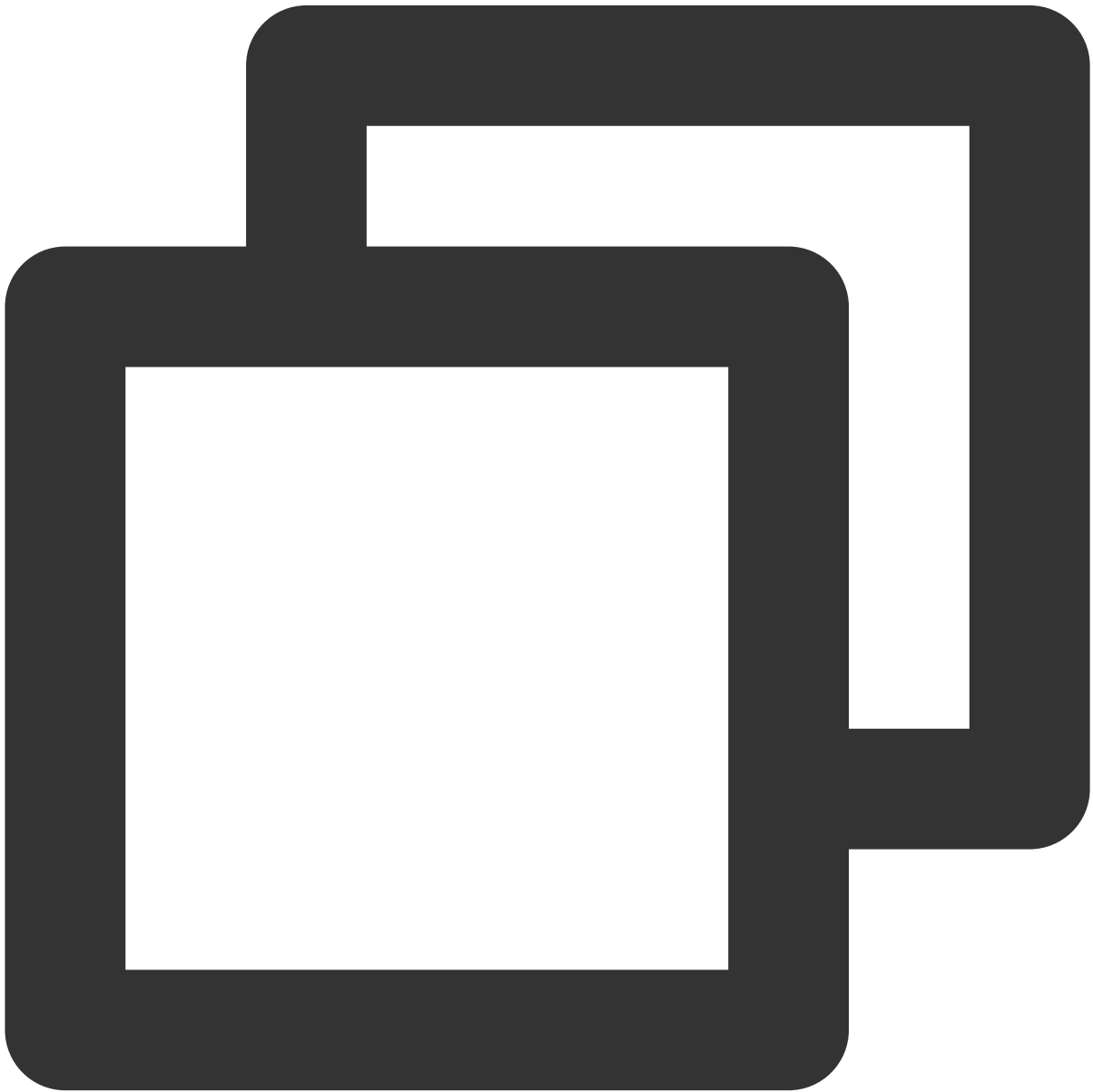
Field	Type	Attribute	Description



RoomId	String	Mandatory	Room ID
Count	Integer	Optional	Fetch number of members, default is 50
Next	String	Optional	The position of members pulled last time, use <code>" "</code> for the first call, continue pulling using the Next value returned

### Sample Response Packets

#### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-52f049ba12d741b7a84959700acdb1d5-O-Seq-56287",
  "Response": {
    "Next": "",
    "MemberList": [
      {
        "Member_Account": "user2",
        "NameCard": "123",
        "Role": "RoomOwner",
        "Marks": [
          1000,
          1002,
          1003
        ],
        "CustomInfo": ""
      }
    ]
  }
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned for each request and required to provide this RequestId when locating issues.
MemberList	Array	Room member list
Member_Account	String	Room member ID
NameCard	String	Room member nickname
Role	String	Room member permissions: Owner, Admin, Member
Marks	Array	Room member tag
CustomInfo	String	Room member custom fields

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through `ErrorCode` and `ErrorInfo` in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

# Update the Room Member Information

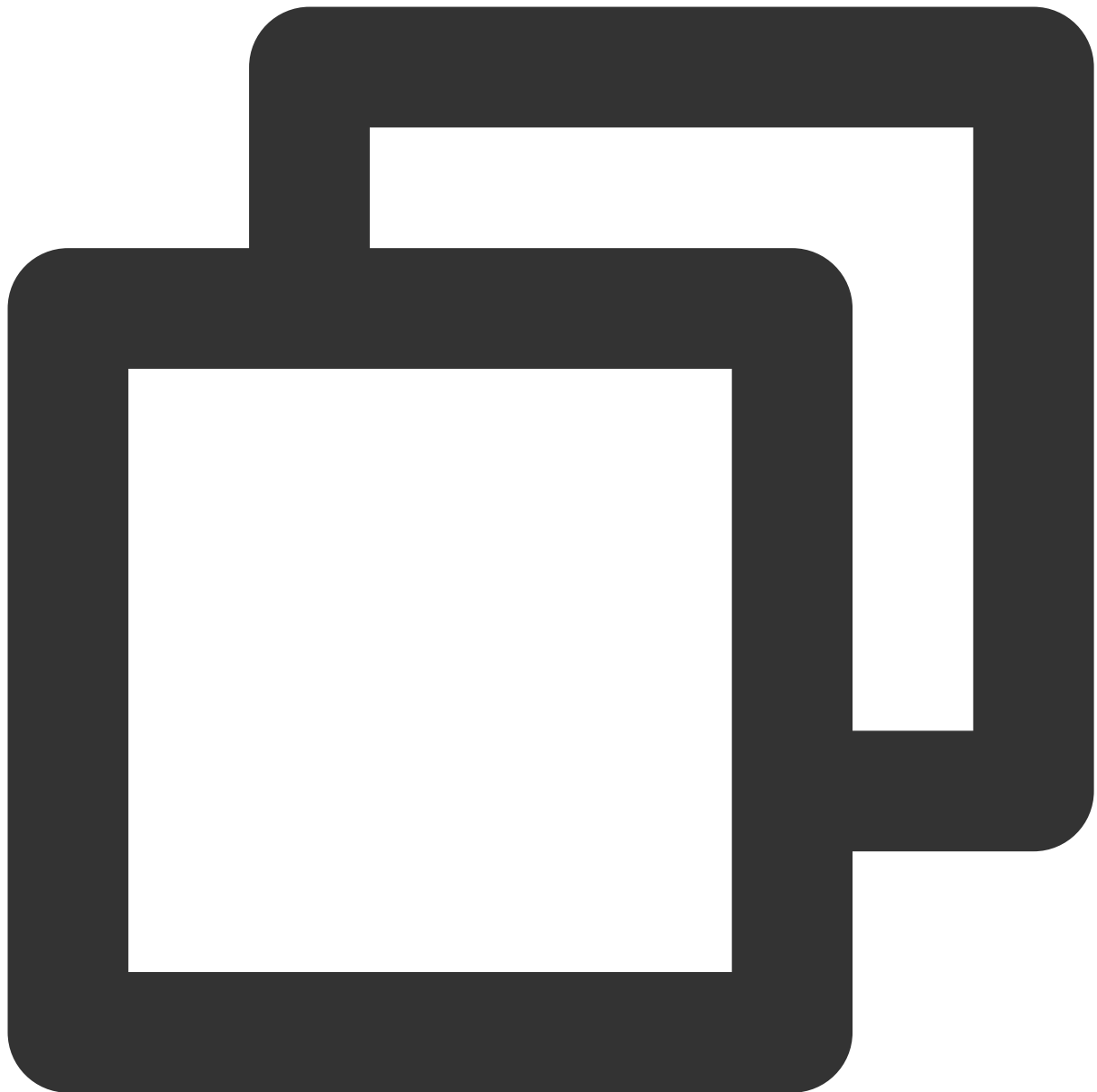
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can update room members information through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_engine_http_srv/update_room_member_info?sdkappid=888888888&id
```

## Request parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

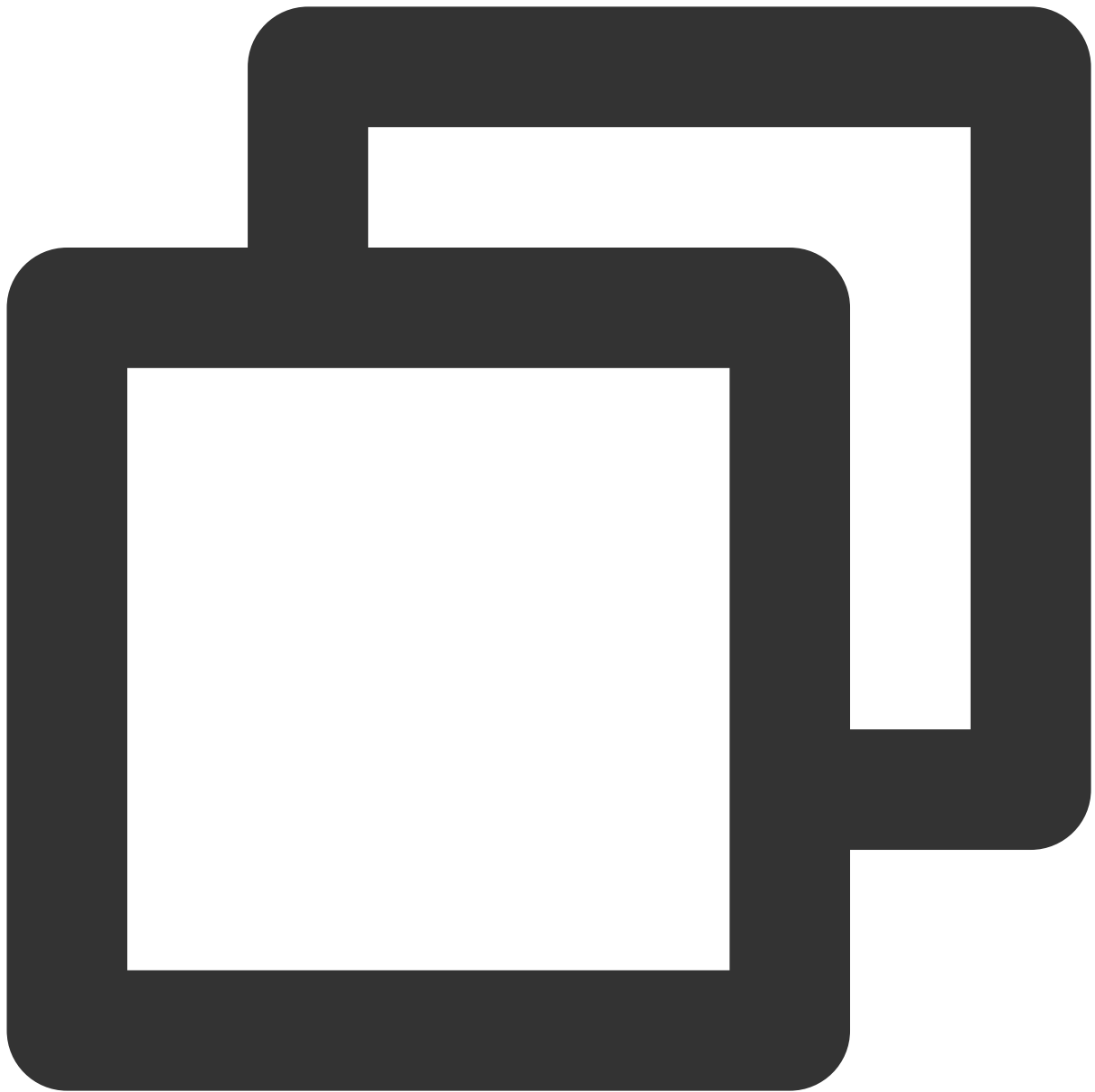
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/update_room_member_info	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "MemberInfo": {
    "Member_Account": "user4",
    "NameCard": "jack",
    "Role": "Admin",
    "CustomInfo": "custom info"
  }
}
```

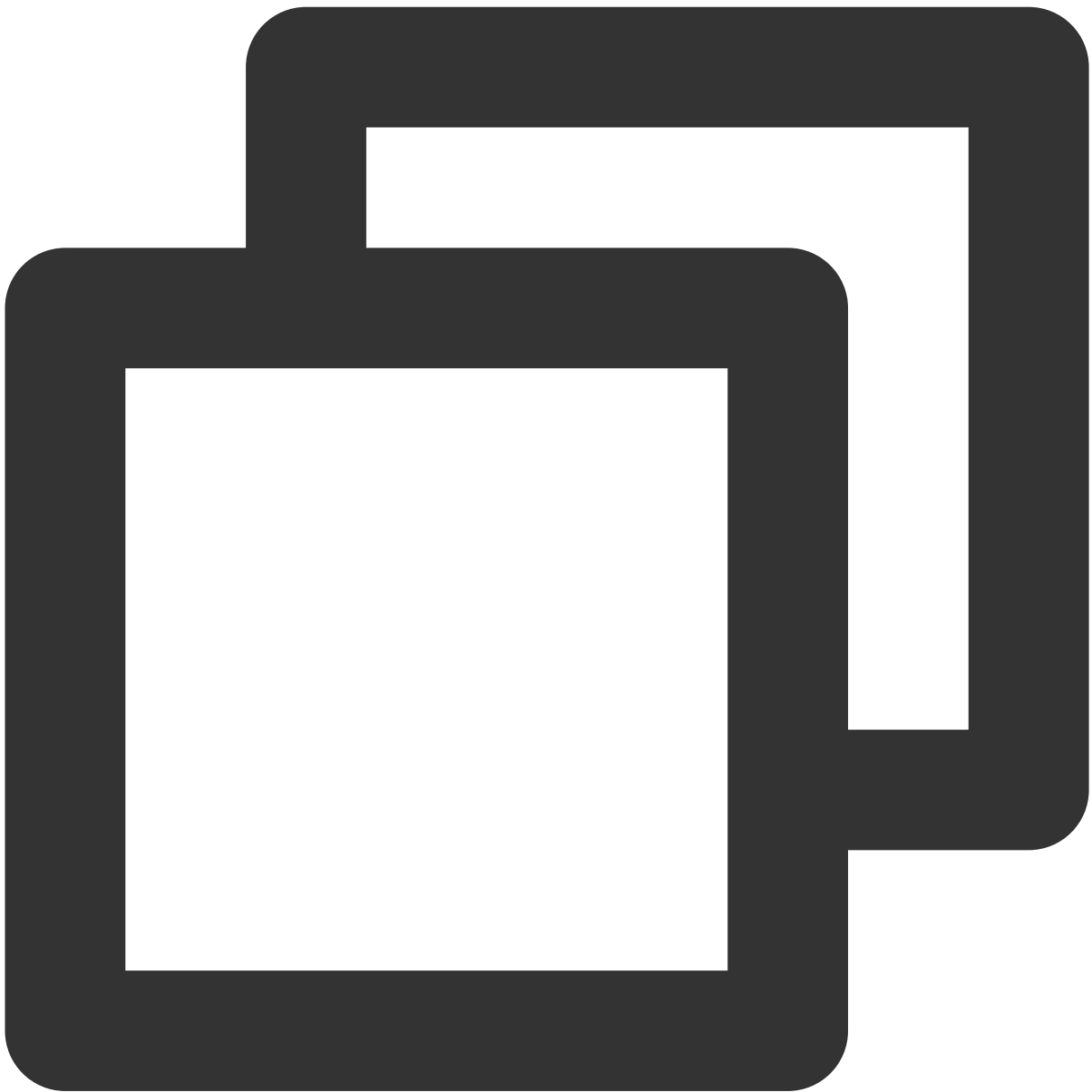
## Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
Member_Account	String	Mandatory	Room member ID
NameCard	String	Optional	Room member nickname, up to 32 Bytes
Role	String	Optional	Room member permissions: Admin, Member
CustomInfo	String	Optional	Room member custom definition fields, up to 60 Bytes

## Sample Response Packets

### Basic Form





```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-8d5f107ce00f420597a2db9fbe7f71ca-O-Seq-102831"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	The room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

# Change the Room Ownership

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrator can change room ownership through this API.

## API Calling Description

### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_srv/change\_room\_owner?sdkappid=88888888&identifi

Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

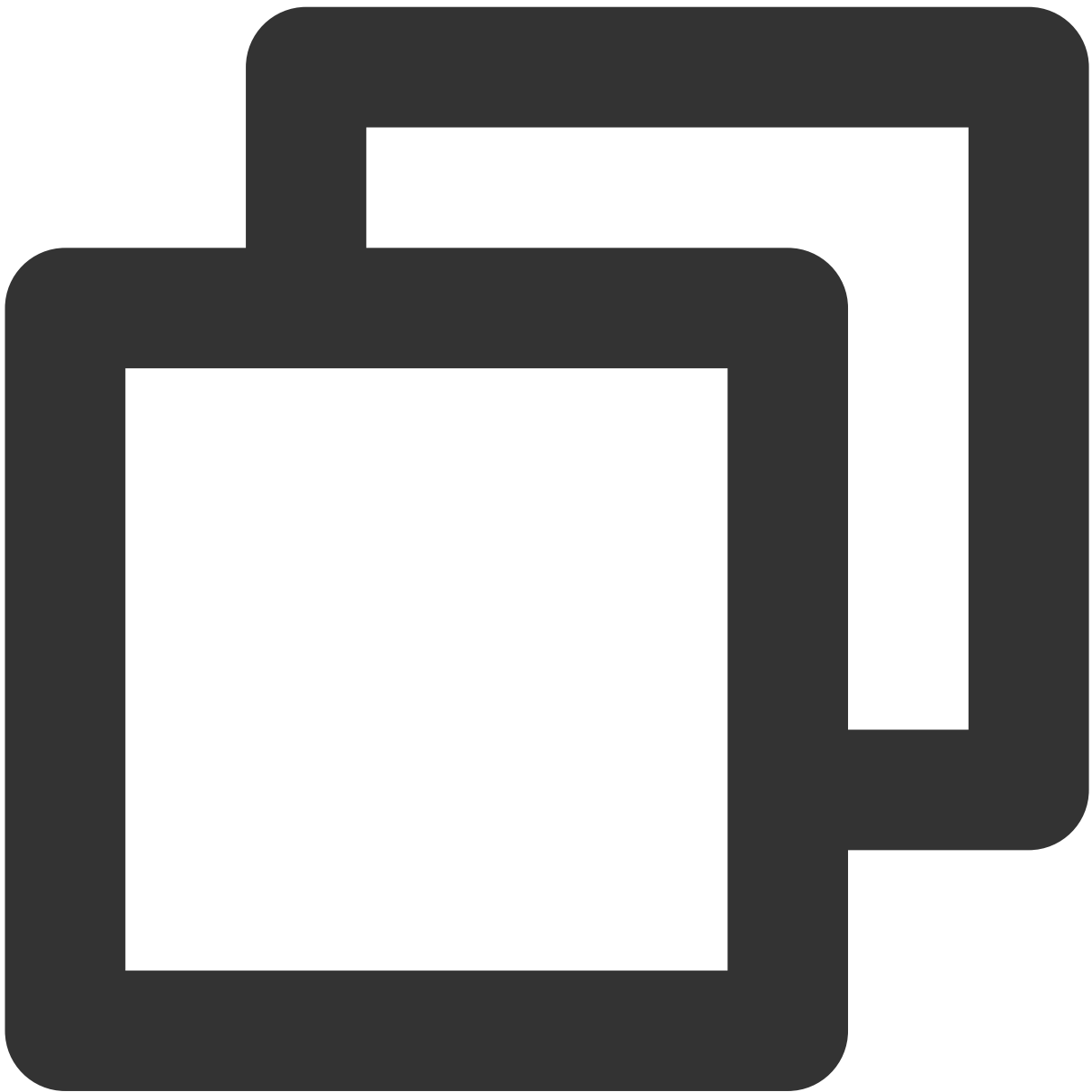
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/change_room_owner	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "From_Account": "user2",
  "NewOwner_Account": "user4"
}
```

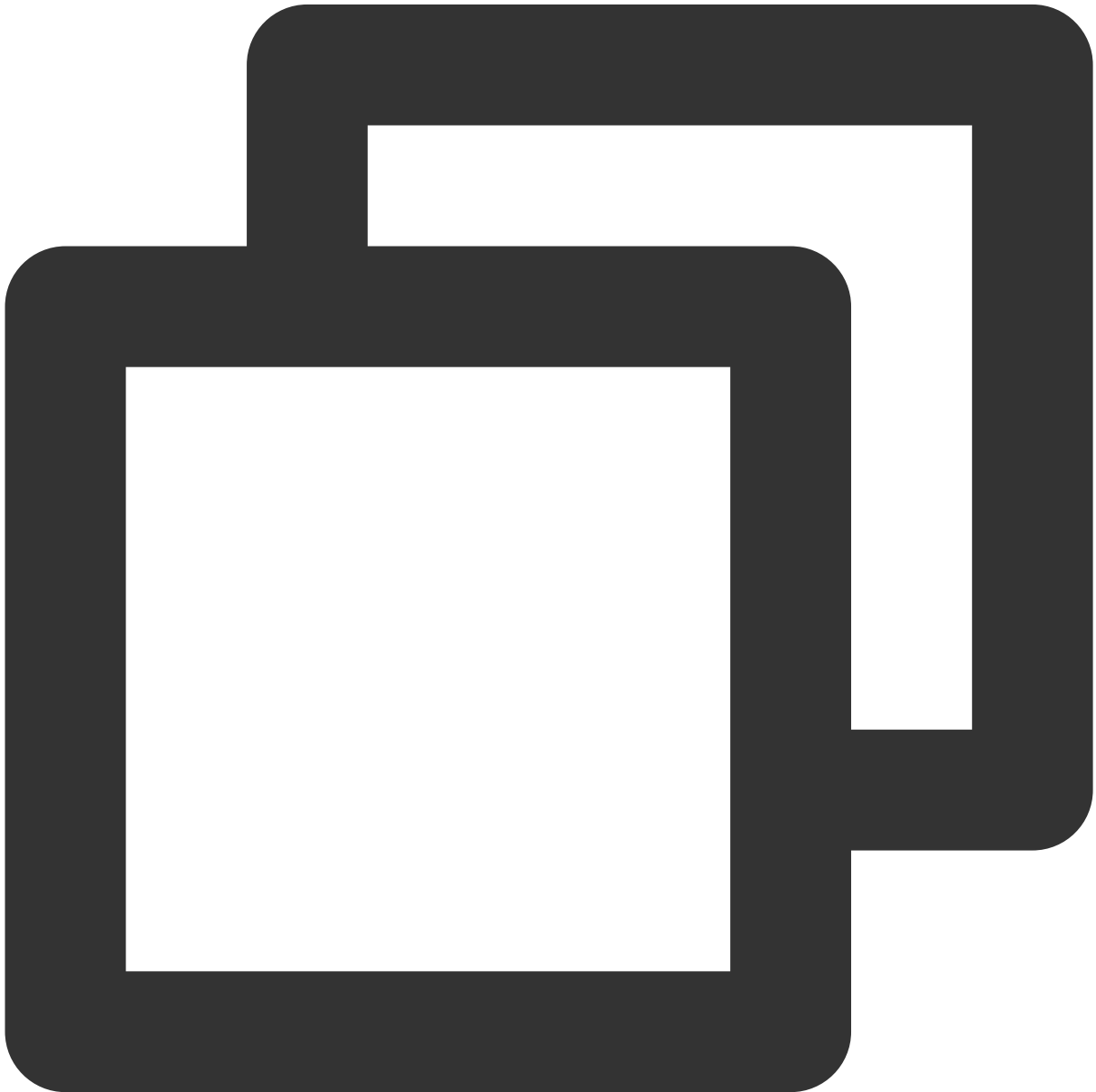
Request Packet Field Description

Field	Type	Attribute	Description

RoomId	String	Mandatory	Room ID
From_Account	String	Mandatory	Original owner
NewOwner_Account	String	Mandatory	New owner, must be a room member

Sample response packets

Basic Form



```
{
```

```
"ErrorCode": 0,  
"ErrorInfo": "",  
"ActionStatus": "OK",  
"RequestId": "Id-8d5f107ce00f420597a2db9fbe7f71ca-O-Seq-102831"  
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	The room does not exist, or it once existed but now has been destroyed.
100005	Non-room members



# Mark Room Members

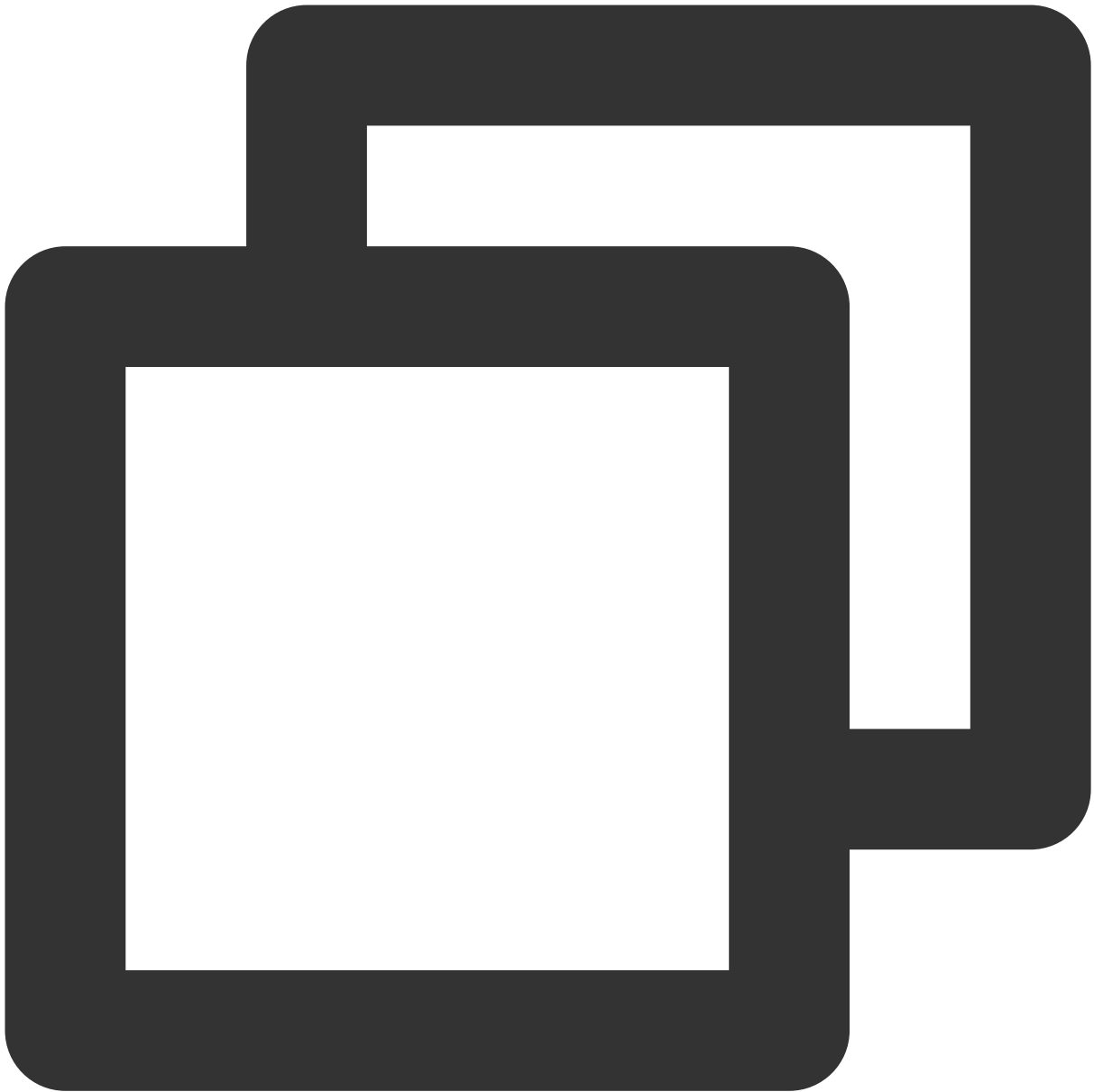
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can mark room members through this API.

## API Calling Description

### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_srv/mark\_room\_member?sdkappid=88888888&identifie

Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

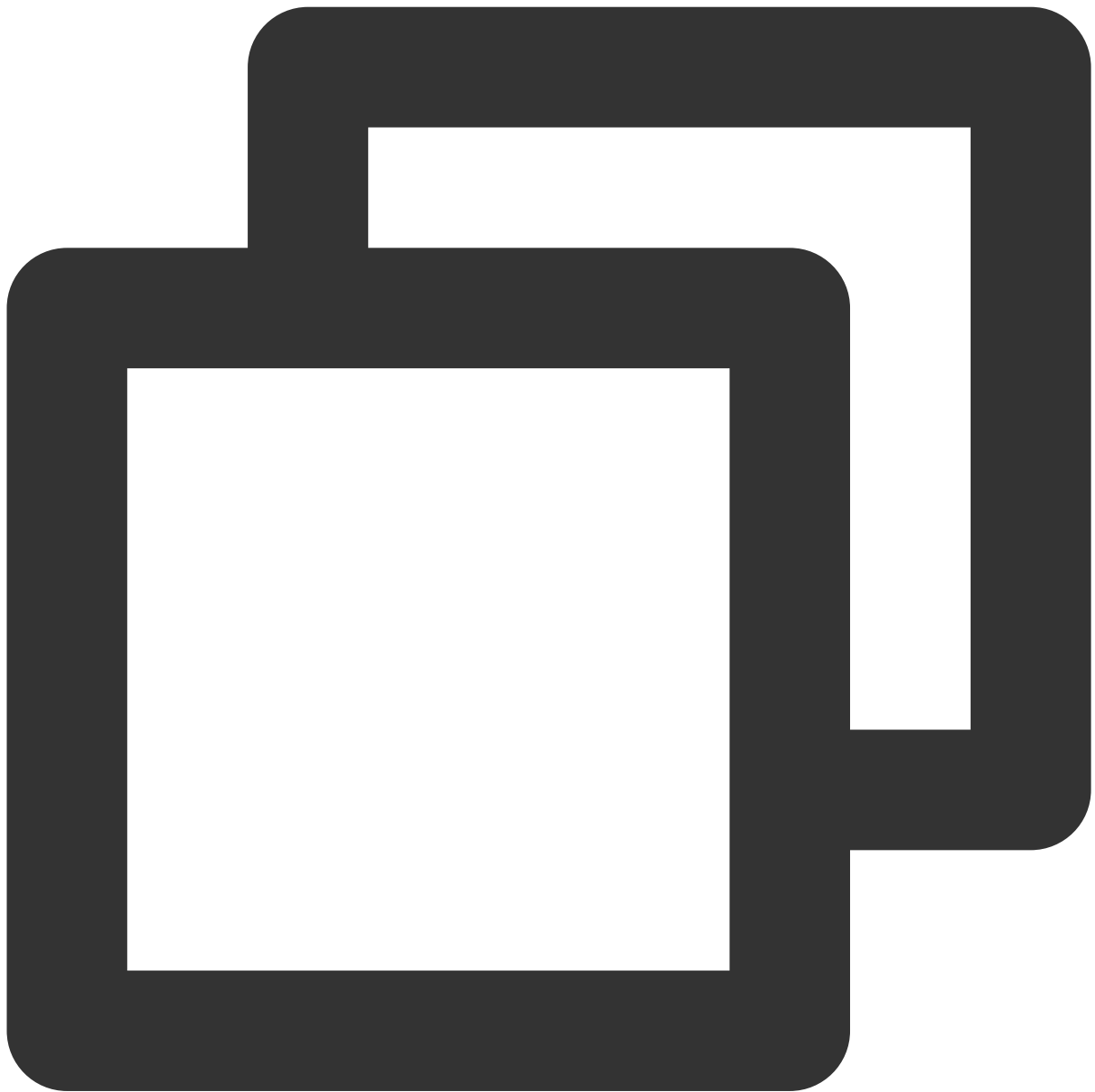
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/mark_room_member	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



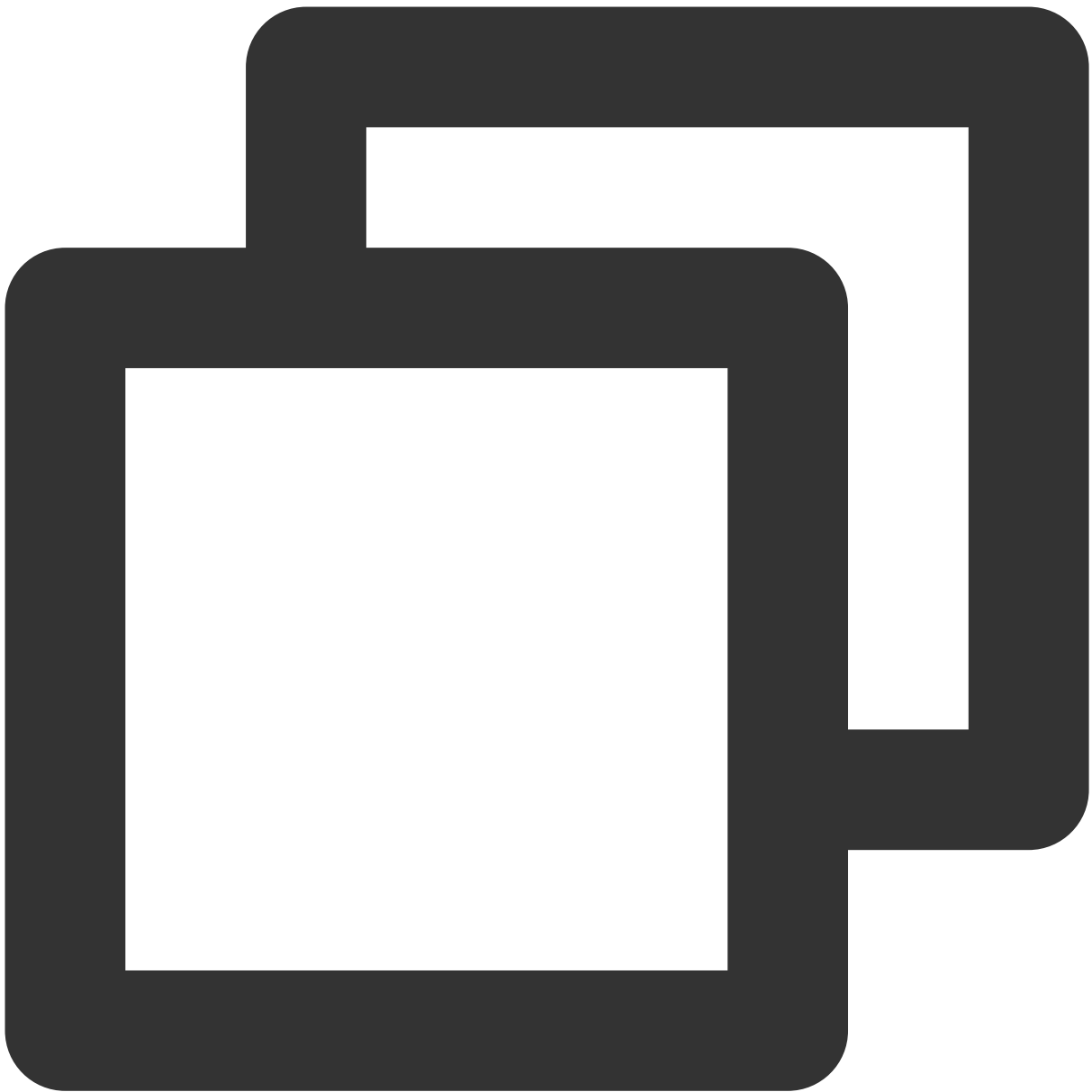
```
{
  "RoomId": "room-test",
  "CommandType": 1,
  "MemberList": [
    {
      "Member_Account": "user2",
      "Marks": [1000, 1002, 1003]
    }
  ]
}
```

## Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
CommandType	String	Mandatory	1: Set Labels, 2: Delete Labels
MemberList	Array	Mandatory	The account list to be set, up to 10 accounts at a time
Marks	Array	Mandatory	Tag must be a number greater than or equal to 1,000, and each room can only set a maximum of 10 different labels, with no more than 3 tags set at a time

## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-ee0a1a5f2c70432f8273a2b279a5fa8f-O-Seq-57703"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions
100009	Number of tags exceeds the upper limit

# Ban Room Members

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can ban room members through this API. Once banned, the members will be removed from the room and will not be able to re-enter during the ban duration.

## API Calling Description

### Sample Request URL





```
https://xxxxxx/v4/room_engine_http_srv/ban_room_member?sdkappid=88888888&identifier
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

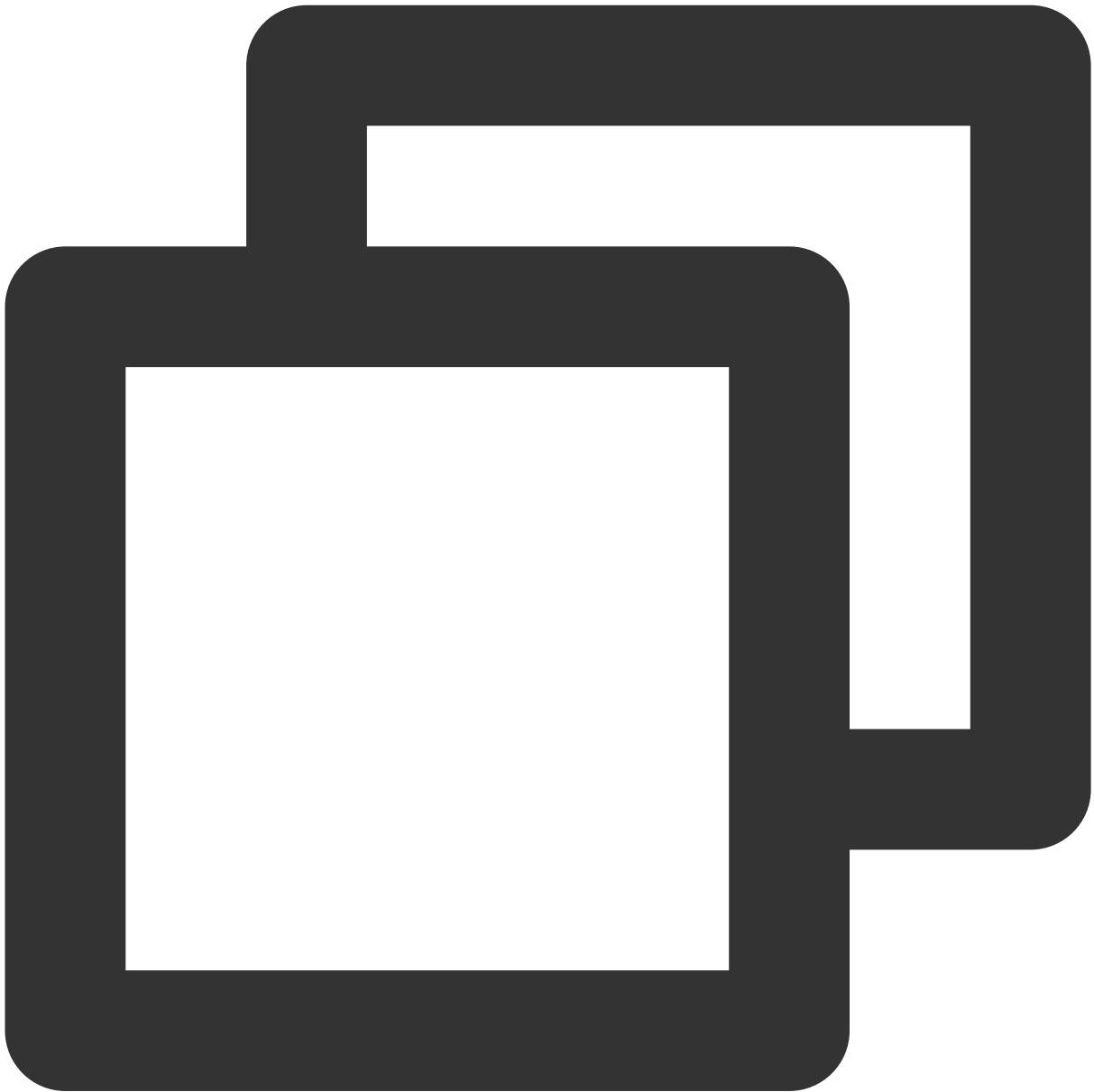
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/ban_room_member	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "MemberList_Account": ["user1"],
  "Reason": "you are banned because of irregularities",
  "Duration": 3600
}
```

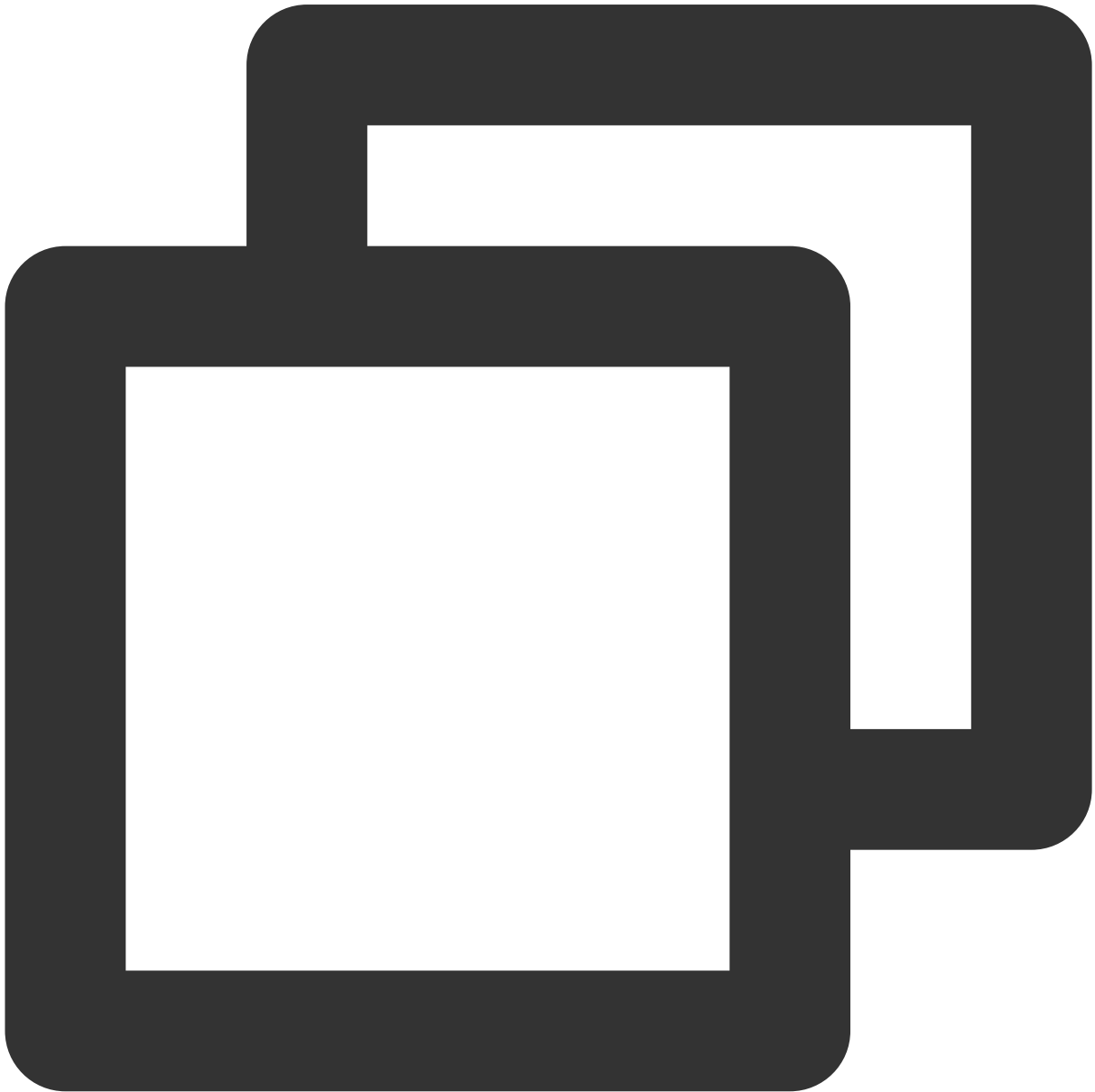
Request Packet Field Description

Field	Type	Attribute	Description
-------	------	-----------	-------------

RoomId	String	Mandatory	Room ID
MemberList_Account	Array	Mandatory	Banned members ID, up to 20 per request
Reason	String	Mandatory	Banned information, up to 1000 bytes
Duration	Integer	Mandatory	Ban duration, unit: seconds

Sample Response Packets

Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-O-Seq-47158"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

# Unban Room Members

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can unban room members through this API. After unbanning, previously banned members can re-enter the room.

## API Calling Description

### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_srv/unban\_room\_member?sdkappid=88888888&identifi

Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_engine_http_srv/unban_room_member	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

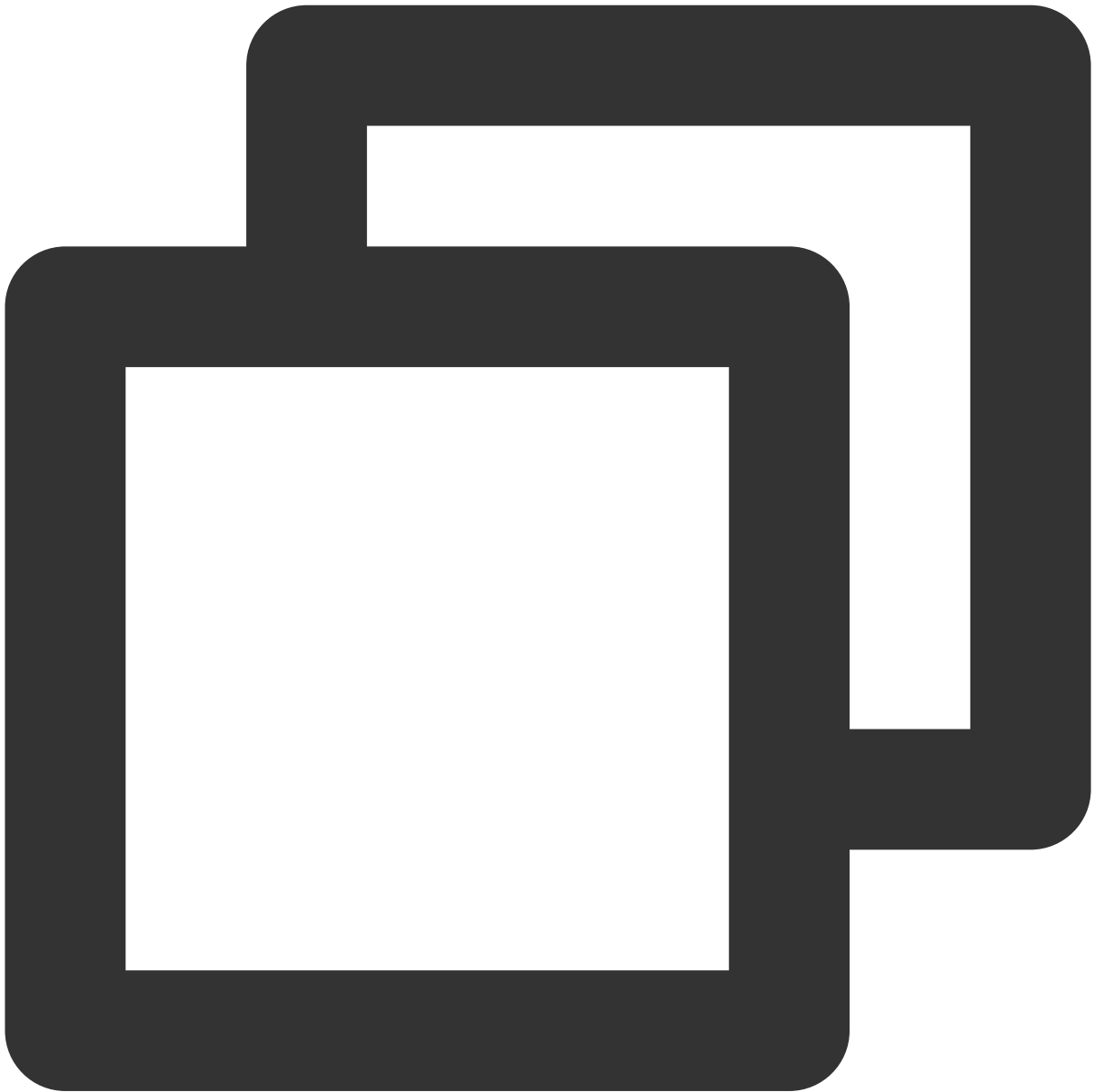
## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form





```
{
  "RoomId": "room-test",
  "MemberList_Account": ["user1"]
}
```

Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

MemberList\_Account

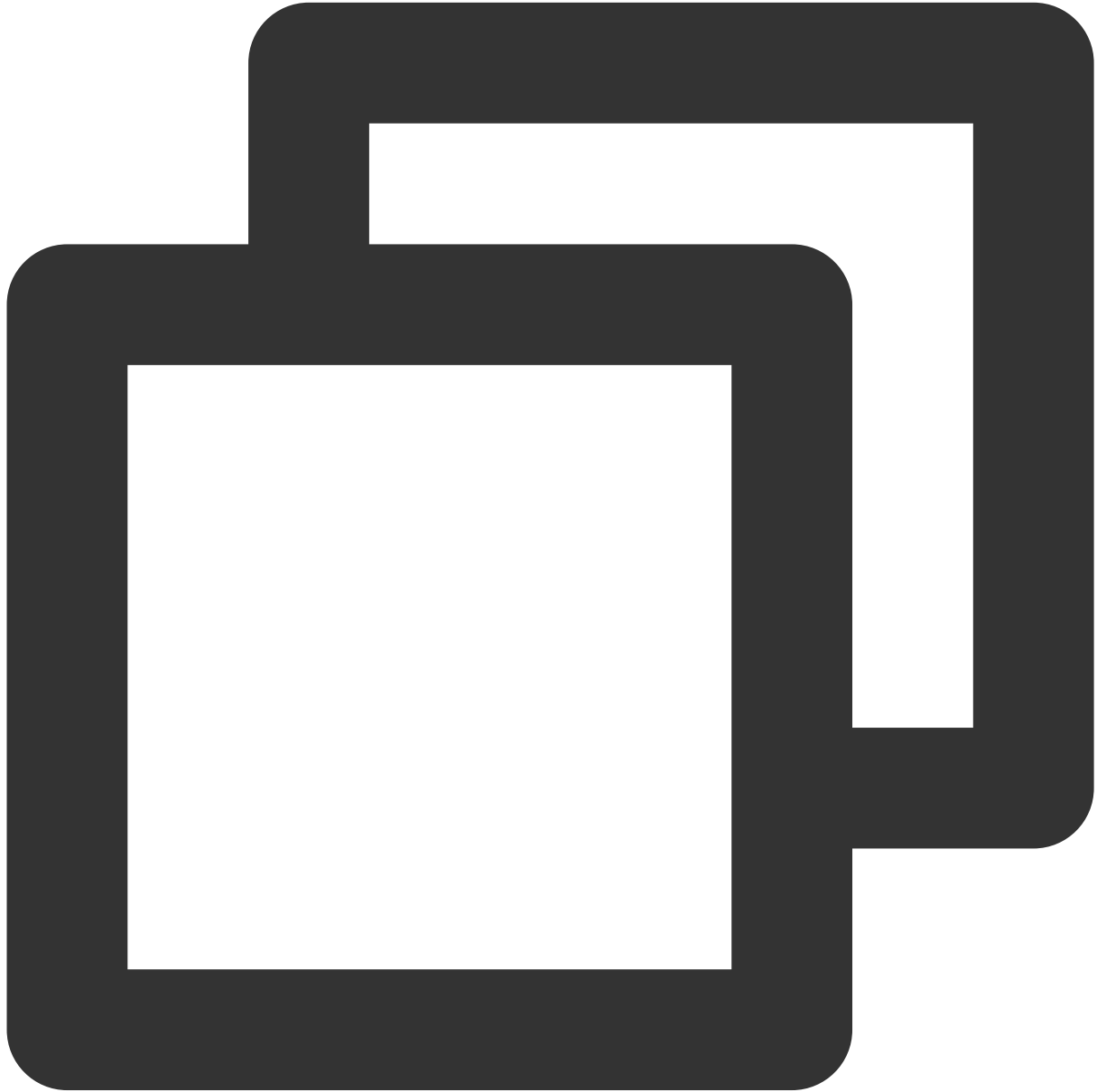
Array

Mandatory

Unbanned members ID, up to 20 per request

## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-O-Seq-47158"
```

```
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

# Get the Banned Room Member List

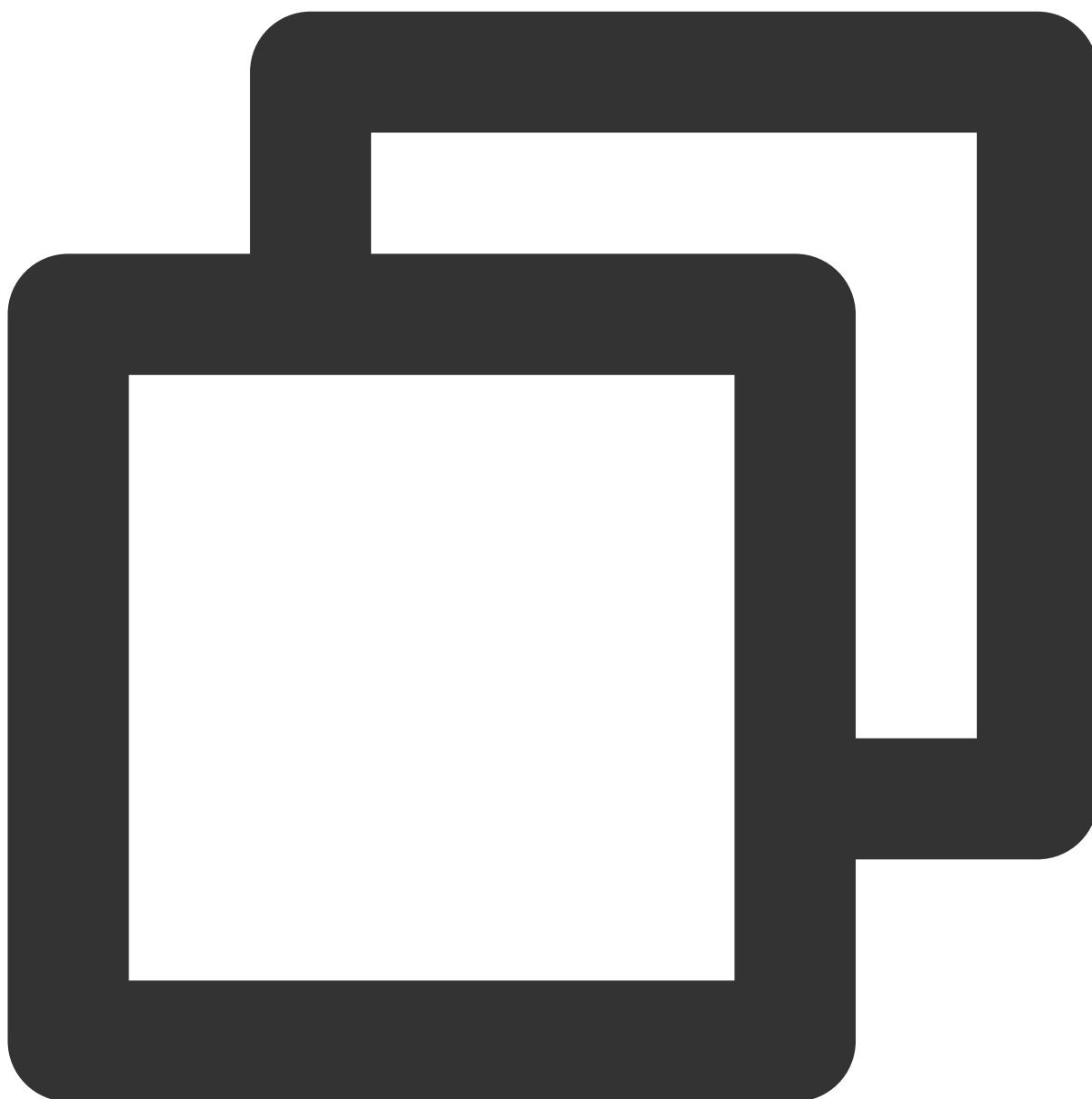
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can get the list of banned members through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_engine_http_srv/get_banned_member_list?sdkappid=888888888&ide
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located:

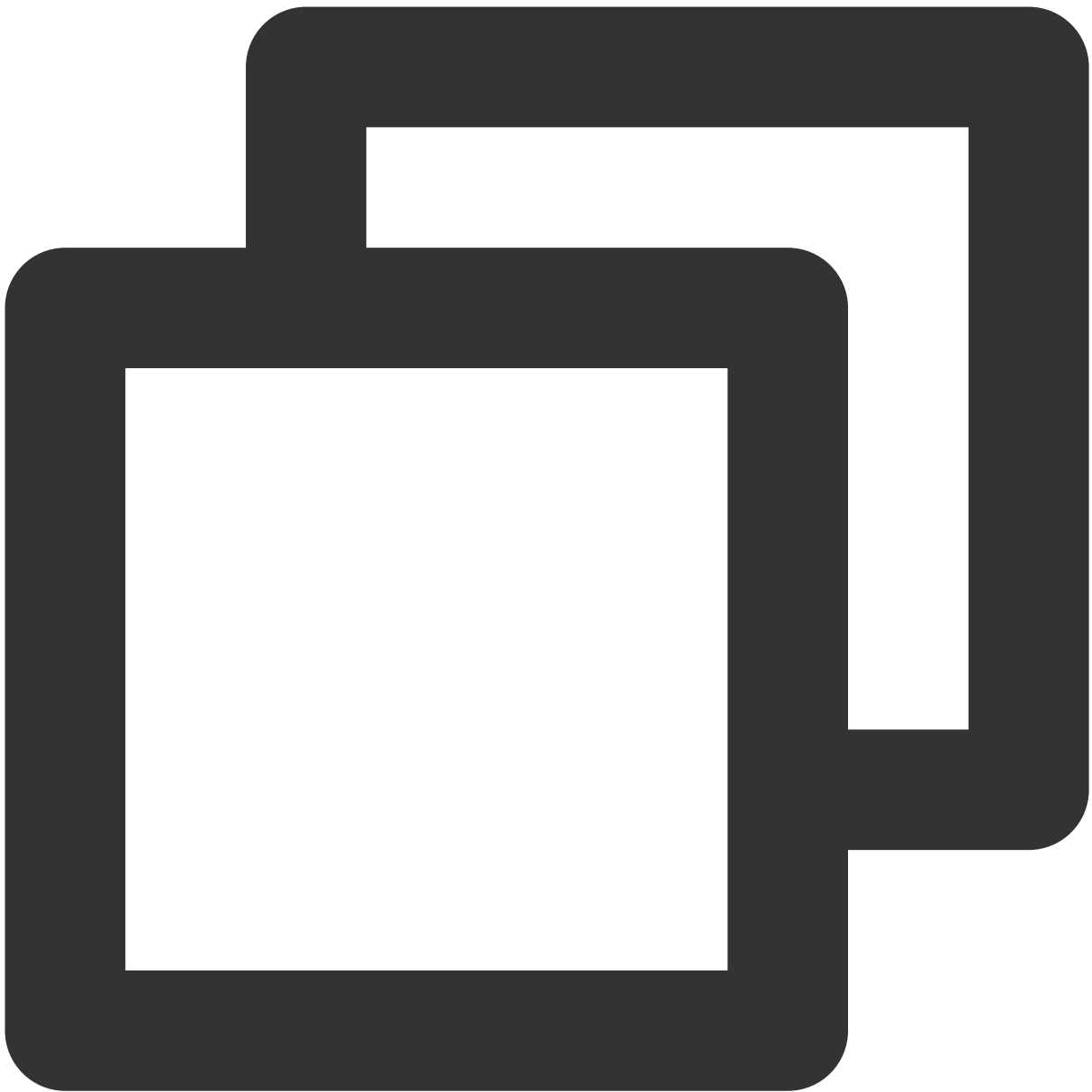
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_srv/get_banned_member_list	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "Limit": 100,
  "Offset": 0
}
```

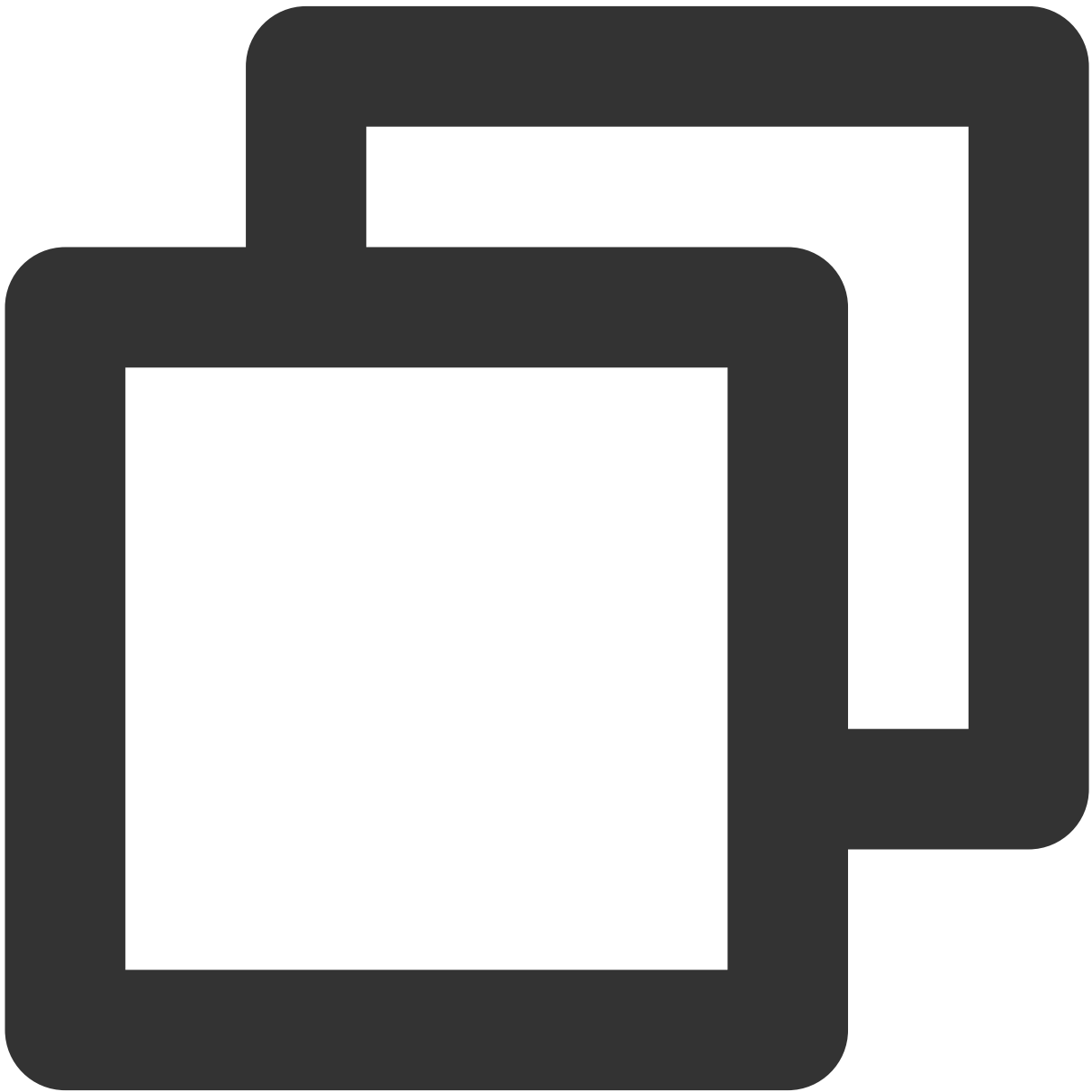
Request Packet Field Description

Field	Type	Attribute	Description

RoomId	String	Mandatory	Room ID
Limit	Integer	Optional	Get the number of banned members at a time, with a maximum of 100
Next	String	Optional	Offset, starting from 0 for the first pull, subsequent requests are based on the NextOffset value in the response packet

Sample Response Packets

Basic Form





```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-99d6df2213b54216a02bc18024a5c218-O-Seq-57681",
  "Response": {
    "BannedAccountList": [
      {
        "Member_Account": "user2",
        "BannedUntil": 1703559417
      }
    ],
    "NextOffset": 0
  }
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.
BannedAccountList	Array	Banned member information, BannedUntil is the ban expiry time of the member, Member_Account is the banned member's account ID.
NextOffset	Integer	The Offset value for the next request. If this value is 0, it means that the list of banned members has been completely pulled.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description

100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	The room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions

# Remove Room Member

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can remove members from the room through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_engine_http_srv/kick_user_out?sdkappid=88888888&identifier=a
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_engine_http_srv/kick_user_out	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

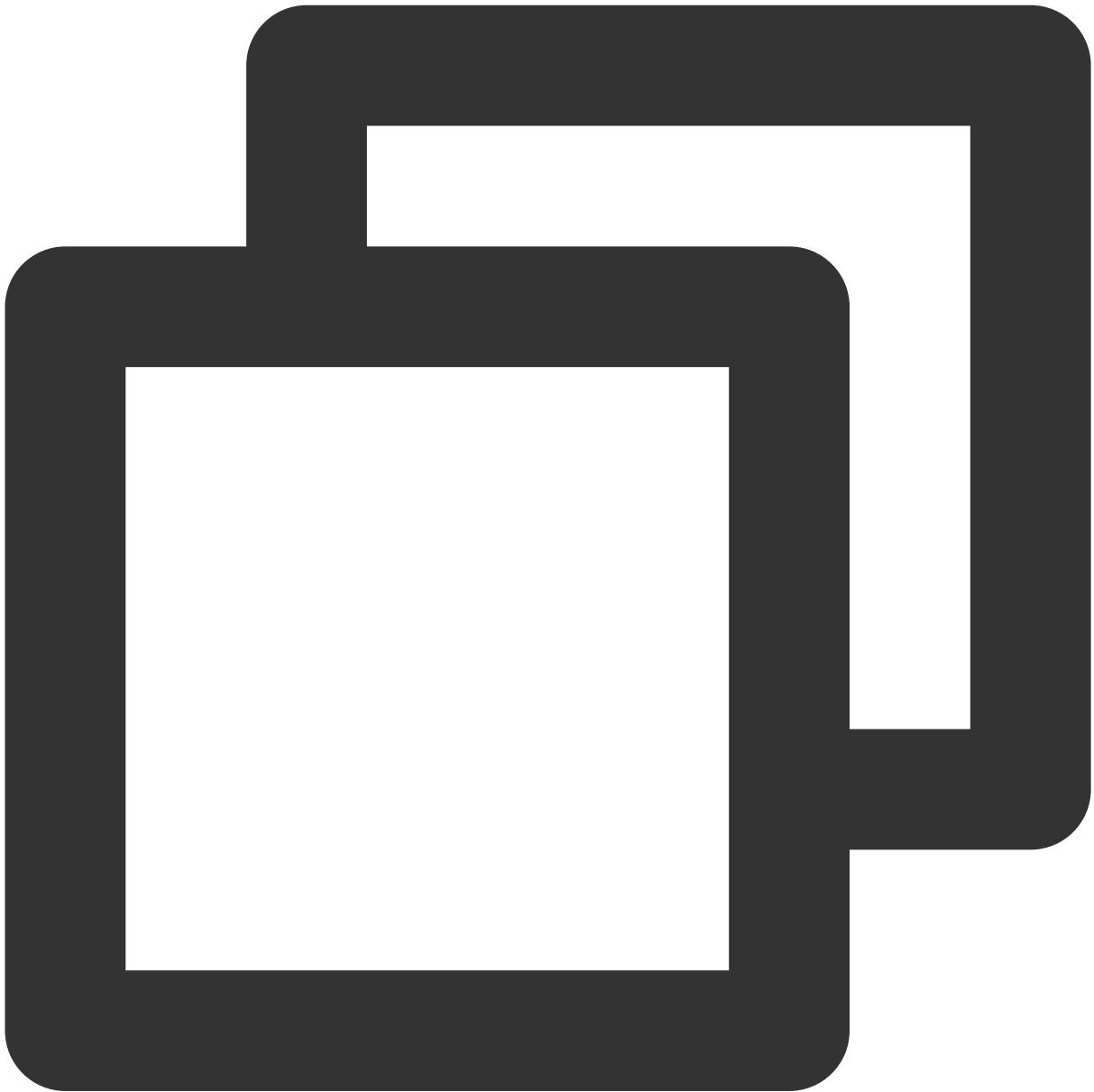
## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic form

Create a room



```
{
  "RoomId": "room-test",
  "MemberList_Account": ["user1"],
  "Reason": "you are kicked because of irregularities"
}
```

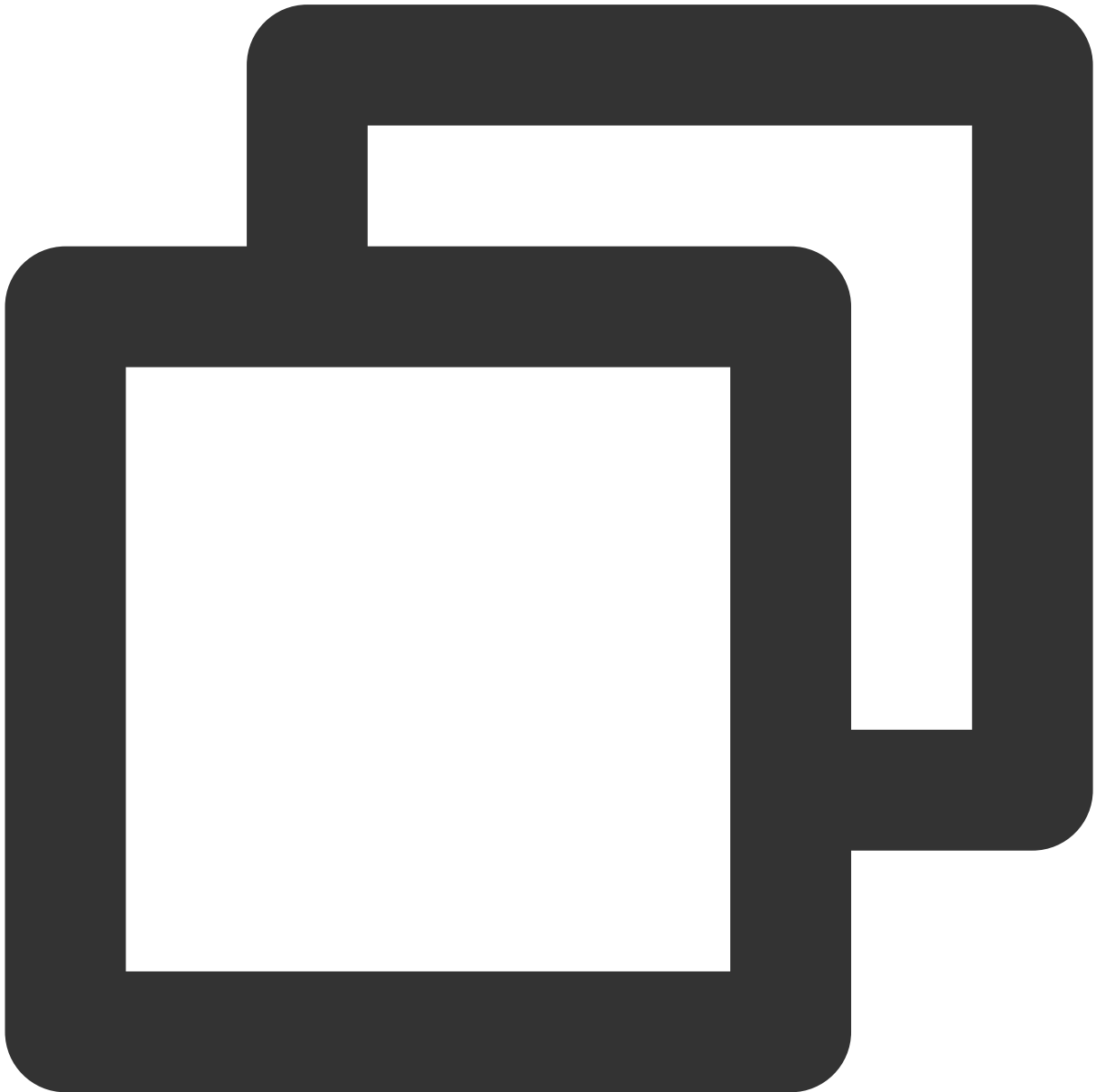
Request Packet Field Description

Field	Type	Attribute	Description

RoomId	String	Mandatory	Room ID
MemberList_Account	Array	Mandatory	The ID of members to be kicked out, up to 20 per request
Reason	String	Mandatory	Kickout information, up to 1000 bytes

Sample Response Packets

Basic Form



```
{
```

```
"ErrorCode": 0,
"ErrorInfo": "",
"ActionStatus": "OK",
"RequestId": "Id-d4e0a71869c84352ac3a730fda71cdb0-O-Seq-47158"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	The room does not exist, or it once existed but now has been destroyed.
100005	Non-room members
100006	Insufficient operational permissions



# Seat Management

## Get the Seat List

Last updated : 2024-06-12 11:46:26

### Feature Overview

App administrators can get the seat list through this API.

### API Calling Description

#### **Sample Request URL**



```
https://xxxxxx/v4/room_engine_http_mic/get_seat_list?sdkappid=88888888&identifier=a
```

## Request parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_mic/get_seat_list	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



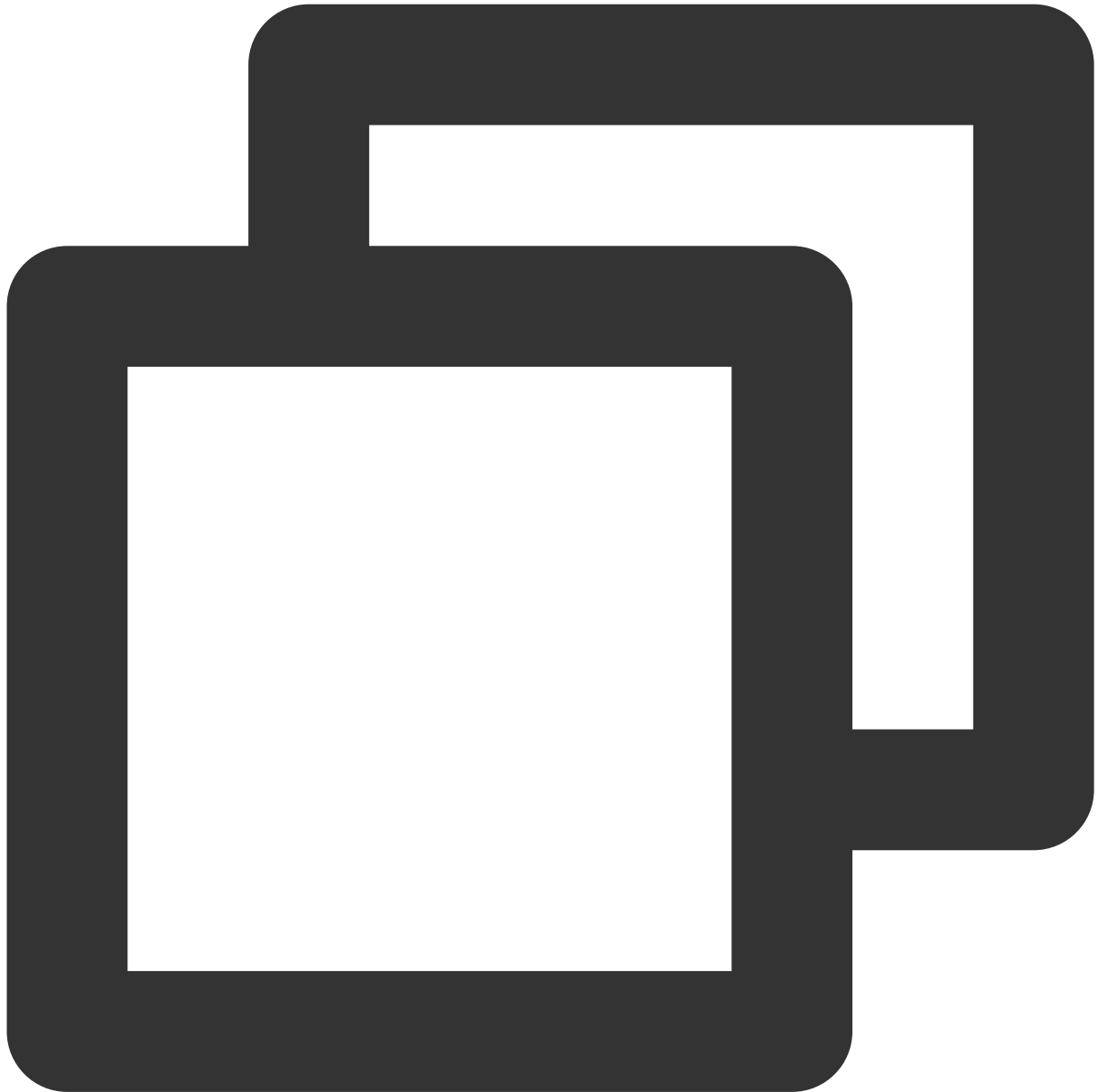
```
{
  "RoomId": "room-test"
}
```

Request Packet Field Description

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717",
  "Response": {
    "SeatList": [
      {
```

```
        "Index": 0,
        "Member_Account": "user2",
        "IsTakenDisabled": false,
        "IsVideoDisabled": false,
        "IsAudioDisabled": false
    },
    {
        "Index": 1,
        "IsTakenDisabled": false,
        "IsVideoDisabled": false,
        "IsAudioDisabled": false
    }
]
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.
SeatList	Array	The list of the seat
Index	Integer	The number of the seat
Member_Account	String	User ID on the seat, empty for no user on the seat
IsTakenDisabled	Bool	Lock seat position
IsVideoDisabled	Bool	Disable seat video stream
IsAudioDisabled	Bool	Disable seat audio stream

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.

# Pick User on the Seat

Last updated : 2024-06-12 11:46:26

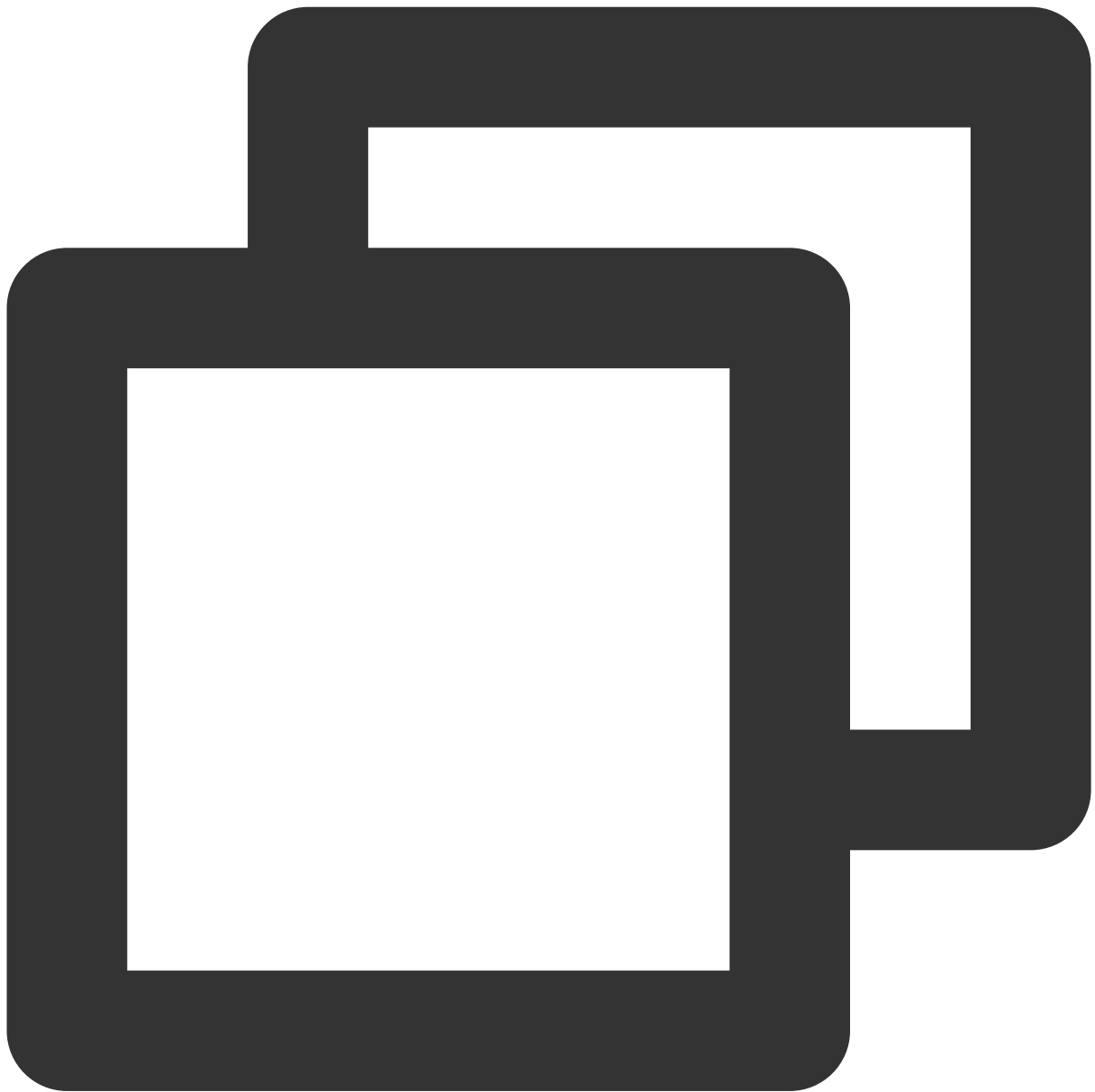
## Feature Overview

App administrators can allow members to take the seat through this API.

## API Calling Description

### Sample Request URL





```
https://xxxxxx/v4/room_engine_http_mic/pick_user_on_seat?sdkappid=88888888&identifi
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

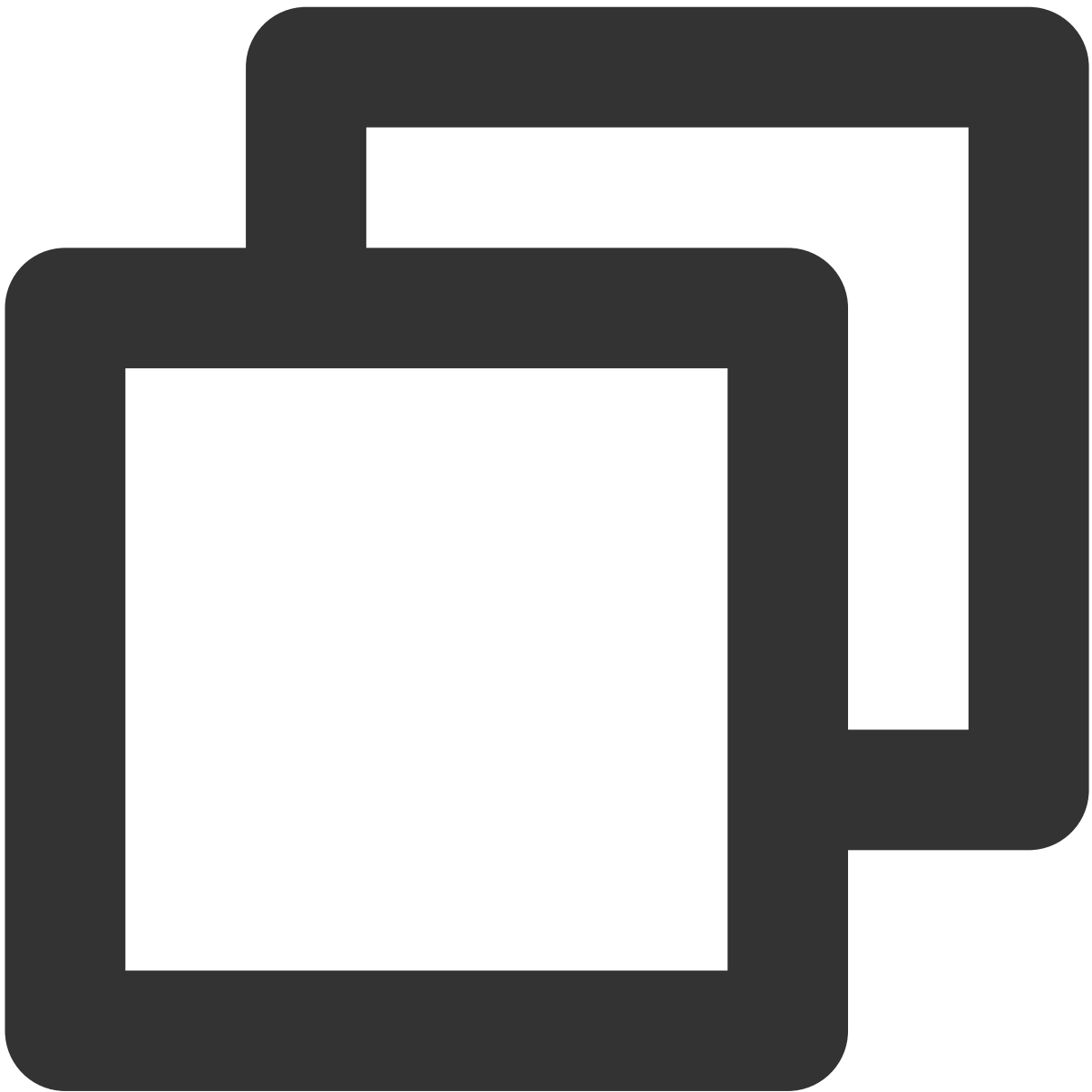
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_mic/pick_user_on_seat	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "Member_Account": "user2",
  "Index": 1
}
```

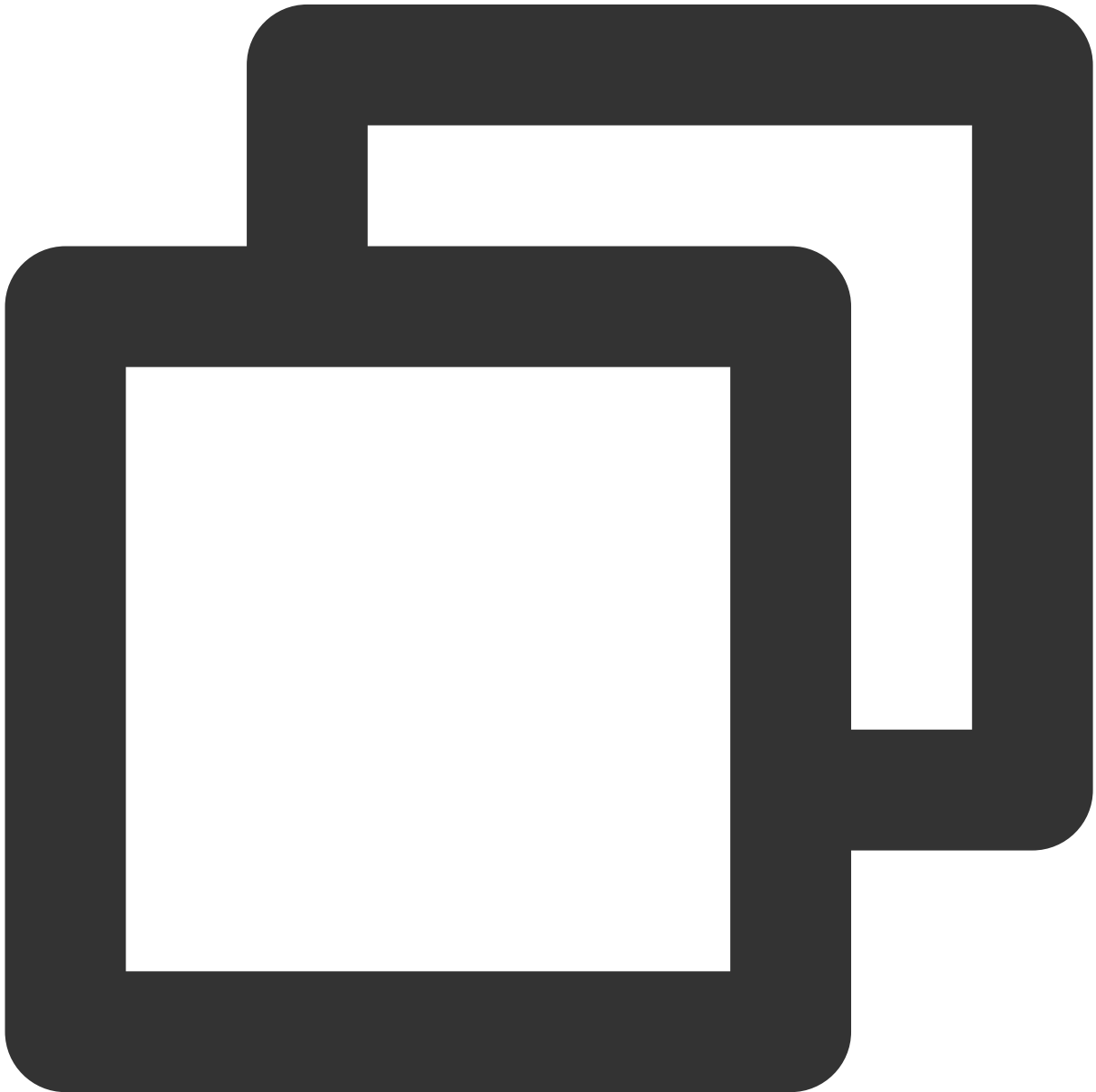
Request Packet Field Description

Field	Type	Attribute	Description

RoomId	String	Mandatory	Room ID
Index	Integer	Mandatory	Seat position number
Member_Account	String	Mandatory	Seat user ID, must be a room member

Sample Response Packets

Basic Form



```
{
```

```
"ErrorCode": 0,  
"ErrorInfo": "",  
"ActionStatus": "OK",  
"RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"  
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100005	The user on the seat is not a room member.
100200	The seat position has been locked, you can try switching to another position.
100202	Already in queue for the seat.
100203	Already on the seat.
100205	The seat position is full.
100210	There is already a user on the seat position.

100211

This room does not support seat connection.

## Possible Callbacks

[After the Seat List Is Changed](#)

# Kick User off the Seat

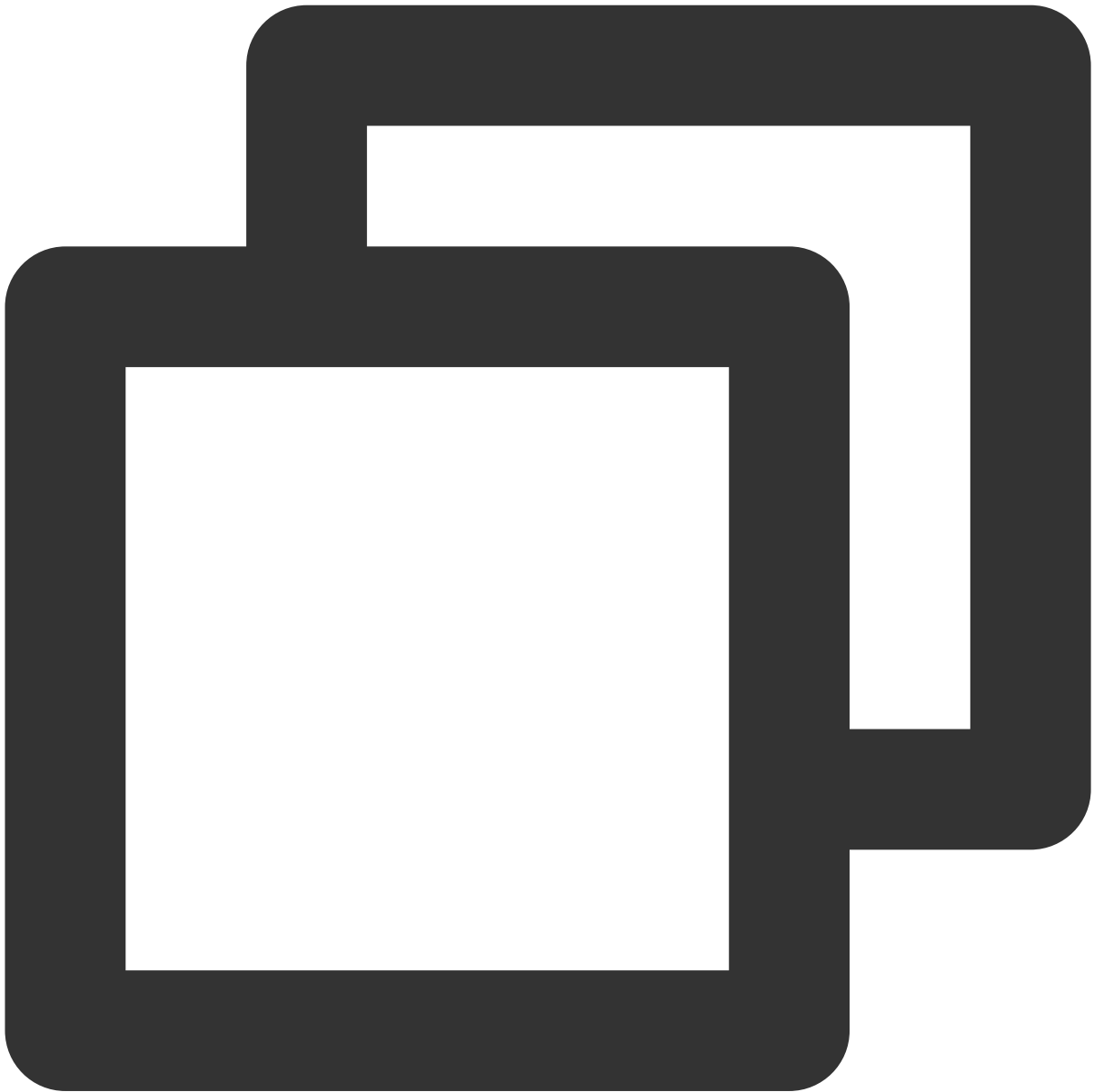
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrator can kick members off the seat through this API.

## API Calling Description

### Sample Request URL



https://xxxxxx/v4/room\_engine\_http\_mic/kick\_user\_off\_seat?sdkappid=88888888&identif

Request Parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:



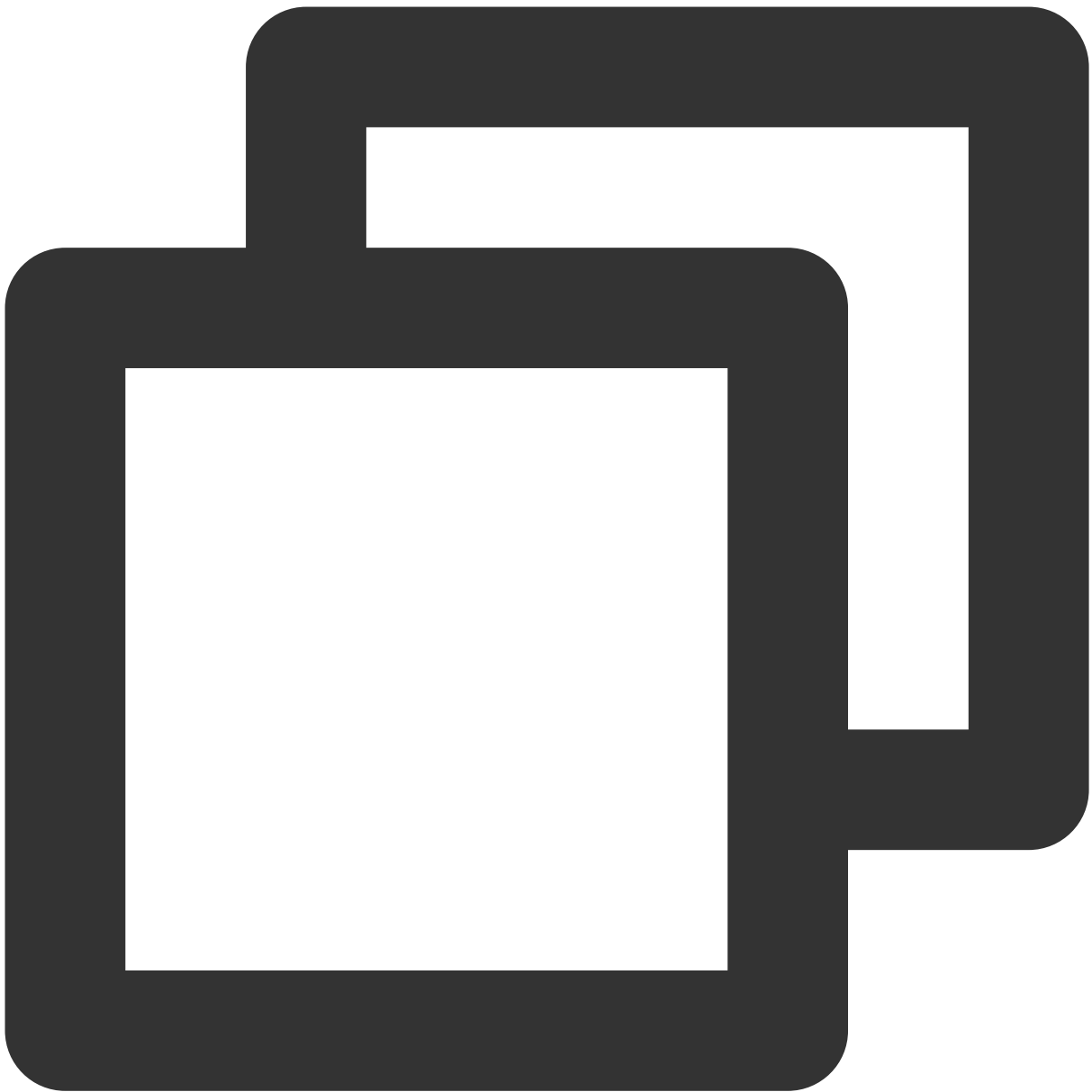
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_mic/kick_user_off_seat	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "Member_Account": "user2",
  "Data": "time over"
}
```

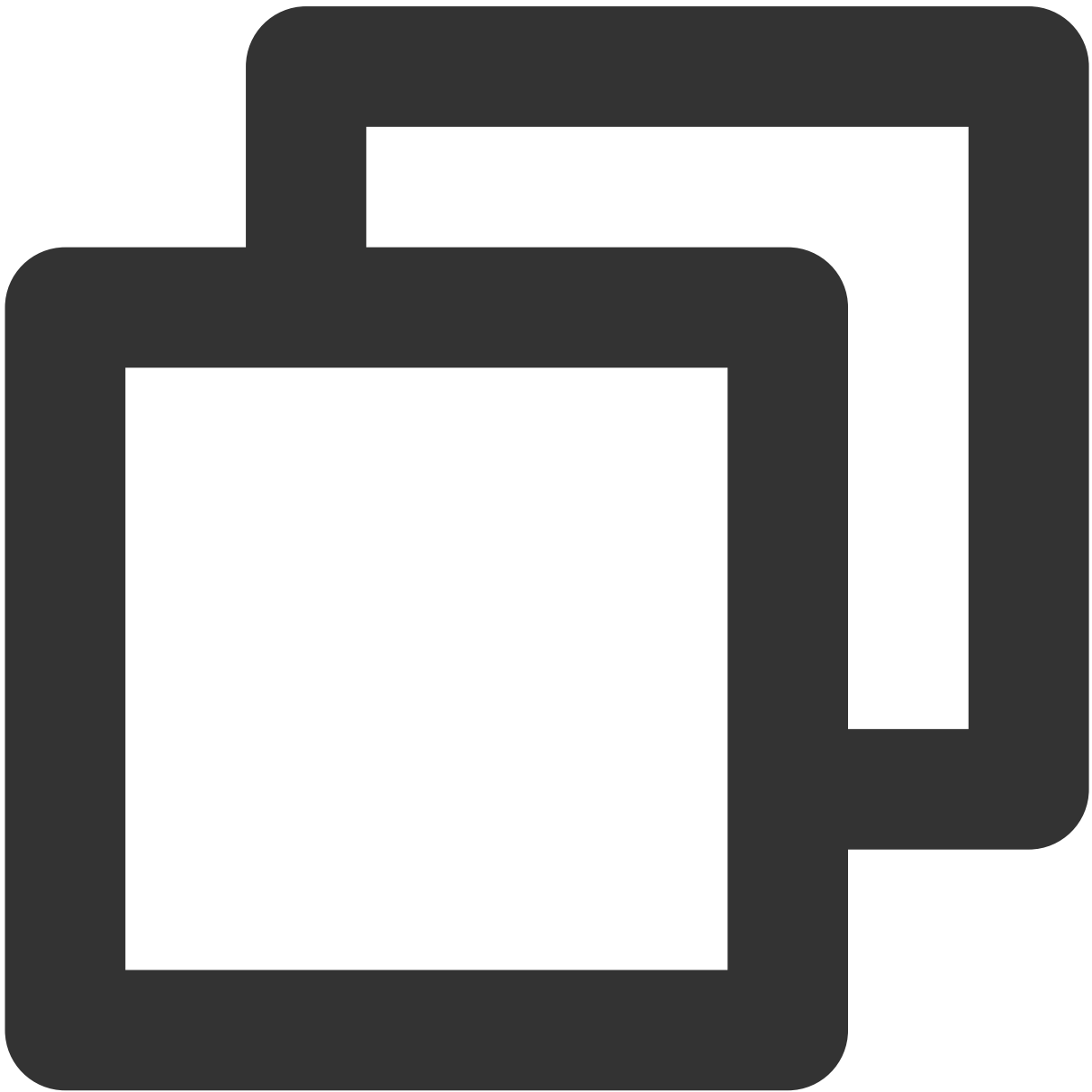
Request Packet Field Description

Field	Type	Attribute	Description

RoomId	String	Mandatory	Room ID
Member_Account	String	Mandatory	User ID on the seat
Data	String	Mandatory	Custom information, which is sent when the user is kicked off the seat and the notification is sent.

Sample Response Packets

Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.

## Possible Callbacks

[After the Seat List Is Changed](#)

# Lock the Seat

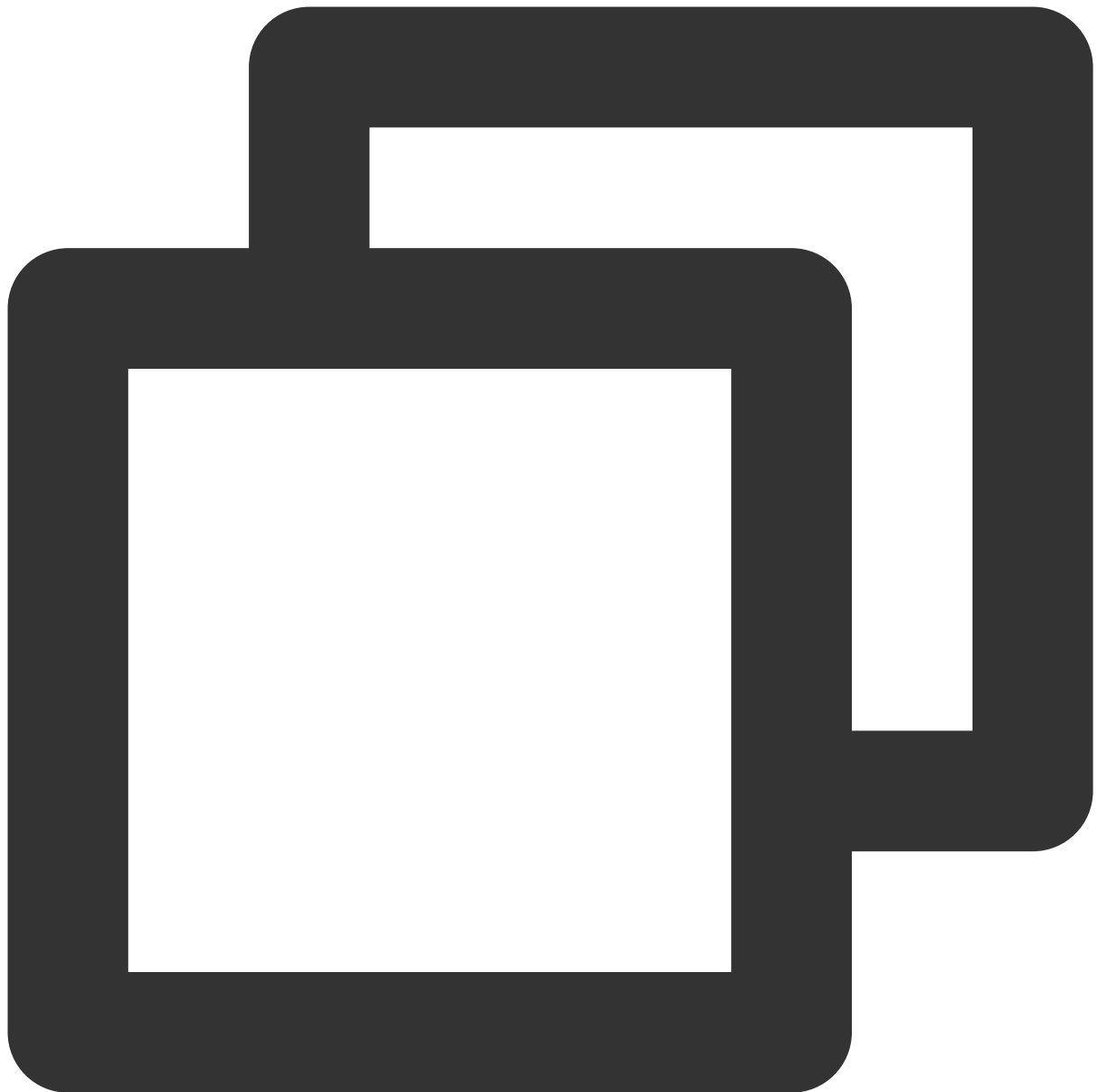
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can lock the seat position through this API. Once locked, users cannot take this position. When locking, if someone is on the seat, the user will be removed before locking.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_engine_http_mic/lock_seat?sdkappid=88888888&identifier=admin
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

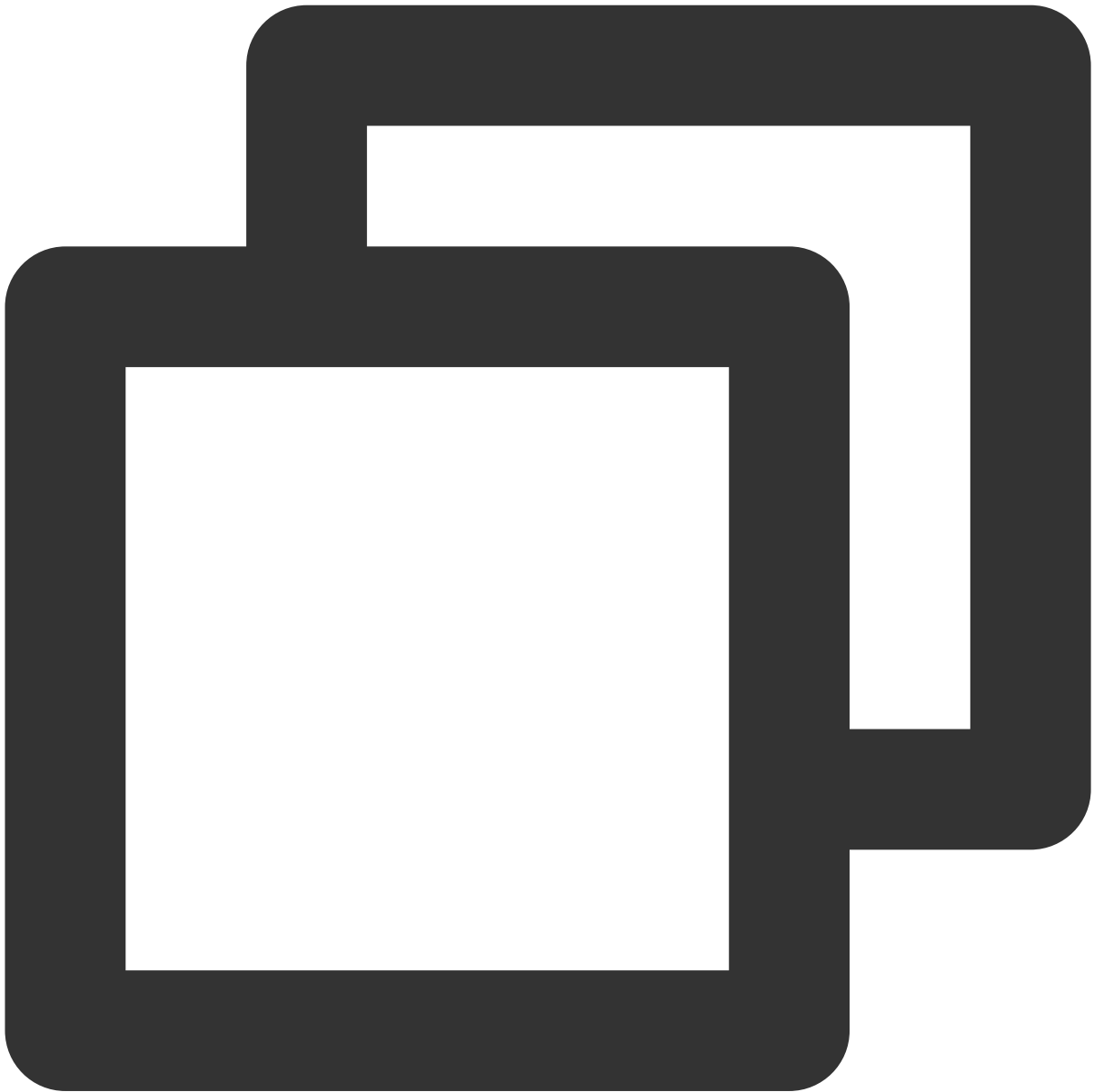
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_engine_http_mic/lock_seat	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "RoomId": "room-test",
  "Index": 2,
  "ForbidTaken": true,
  "ForbidVideo": true,
  "ForbidAudio": true
}
```

Request Packet Field Description

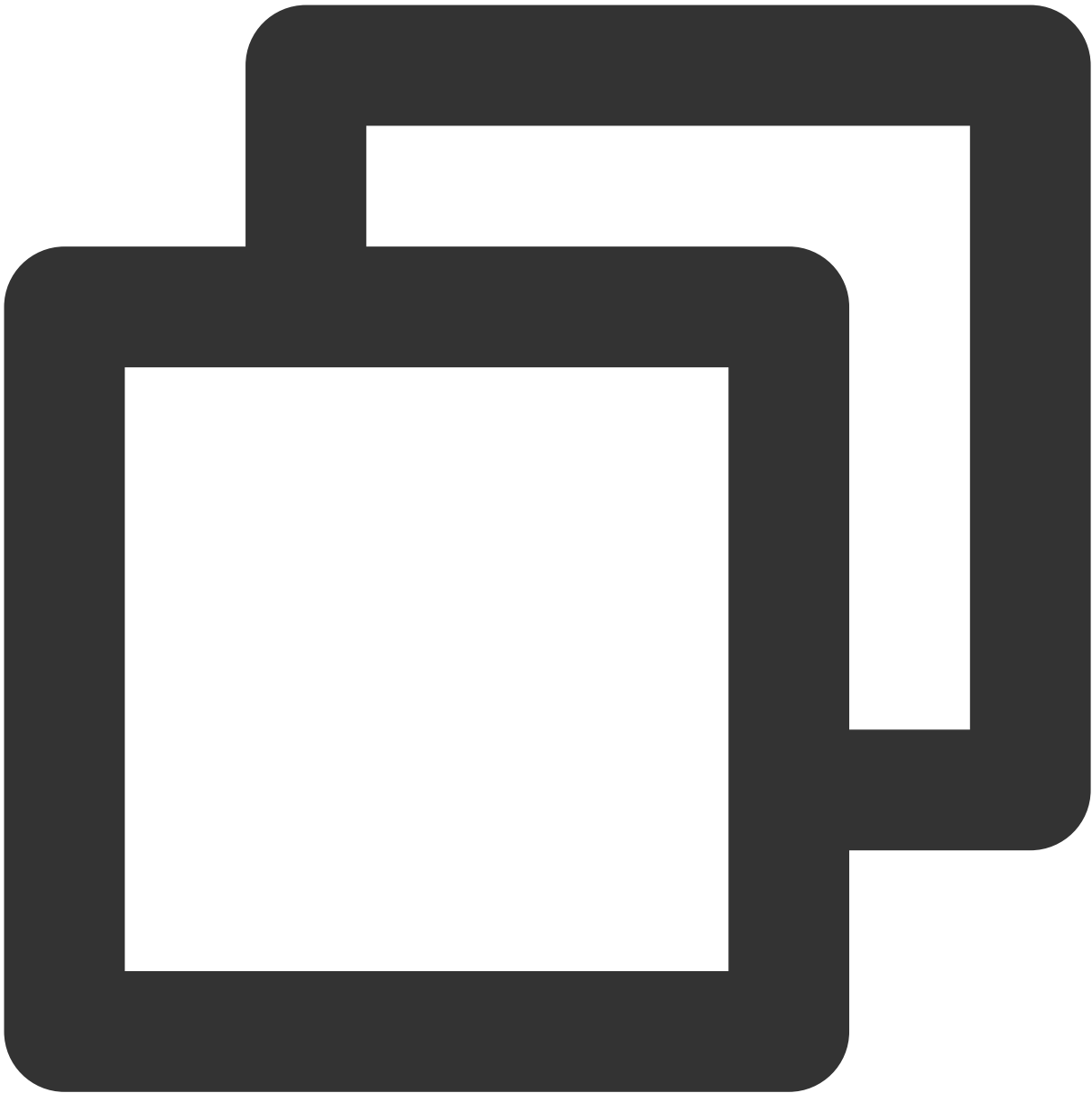
--	--	--	--



Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
Index	Integer	Mandatory	The seat position number
ForbidTaken	Bool	Optional	Lock the seat position
ForbidVideo	Bool	Optional	Disable video stream
ForbidAudio	Bool	Optional	Disable audio stream

## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	Invalid parameter, please check the request for correctness based on the error description.
100004	Room does not exist, or it once existed but now has been destroyed.
100211	This room does not support seat connection.

## Possible Callbacks

[After the Seat List Is Changed](#)

# Third-Party Callback Callback Overview

Last updated : 2024-06-12 11:46:26

To facilitate your control of the feature form of your App, RoomKit provides callback capabilities.

## Feature Overview

Users can configure a callback to a specified URL through RESTful API. When the executed CallbackCommand is on the configuration list, the callback will be triggered.

## Notes

Only one callback can be set per sdkAppId.  
Ensure that the callback URL is accessible.

# Callback Command List

Last updated : 2024-06-12 11:46:26

## Room Related

Callback Type	Callback Command Word
After a Room Is Created	<a href="#">Room.CallbackAfterCreateRoom</a>
After a Room Is Destroyed	<a href="#">Room.CallbackAfterDestroyRoom</a>
After the Room Information Is Updated	<a href="#">Room.CallbackUpdateRoomInfo</a>

## User Related

Callback Type	Callback Command Word
After a Room Is Entered	<a href="#">Room.CallbackAfterMemberEnter</a>
After a Room Is Left	<a href="#">Room.CallbackAfterMemberLeave</a>

## Seat Connection Related

Callback Type	Callback Command Word
After the Seat List Is Changed	<a href="#">Mic.CallbackAfterSeatInfoChanged</a>

# Callback Configuration

## Query Callback Configuration

Last updated : 2024-06-12 11:46:26

### Feature Overview

App administrators can get a callback through this API.

### API Calling Description

#### **Sample Request URL**



```
https://xxxxxx/v4/room_config/get_callback?sdkappid=888888888&identifier=admin&users
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_config/get_callback	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

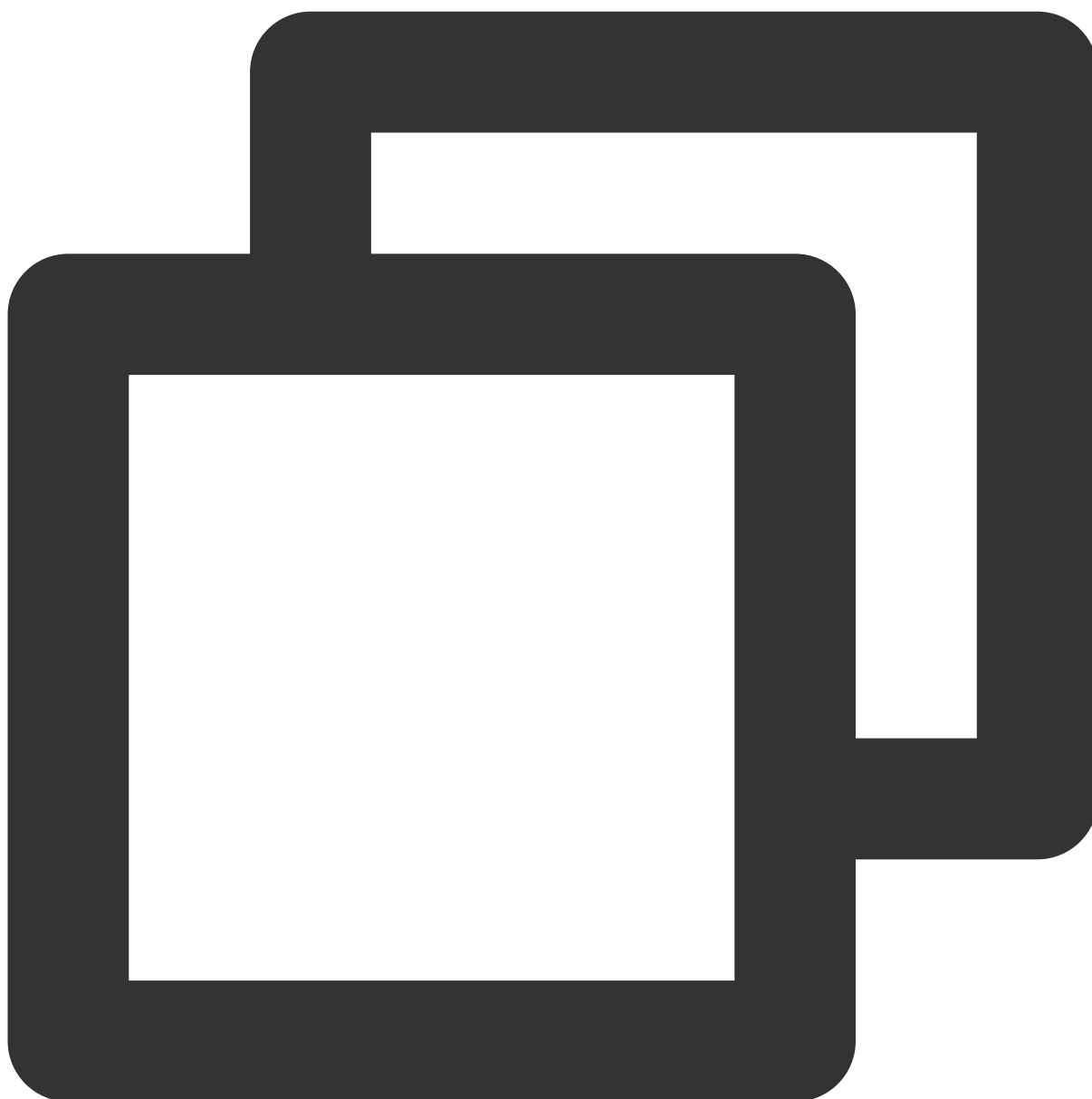
200 times per second.

## Sample Request Packets

### Basic Form

Just pass it empty.





```
{  
}
```

## Request Packet Field Description

## Sample Response Packets

### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-O-Seq-61309",
  "Response": {
    "Url": "http://www.example.com/callback",
    "CallbackCommandList": [
      "Room.CallbackAfterCreateRoom",
      "Room.CallbackAfterDestroyRoom",
      "Room.CallbackUpdateRoomInfo",
    ]
  }
}
```

```
        "Room.CallbackAfterMemberEnter",
        "Room.CallbackAfterMemberLeave",
        "Mic.CallbackAfterSeatInfoChanged"
    ]
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.
Url	String	Callback address, must start with http/https, it is recommended to use the more secure https.
CallbackCommandList	Array	For the command words that trigger the callback, refer to <a href="#">Callback Command Word List</a> .

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	The parameter is invalid, please check the request for correctness based on the error description.
100301	Callback configuration does not exist, you can create it using the Create Callback Configuration interface.



# Create Callback Configuration

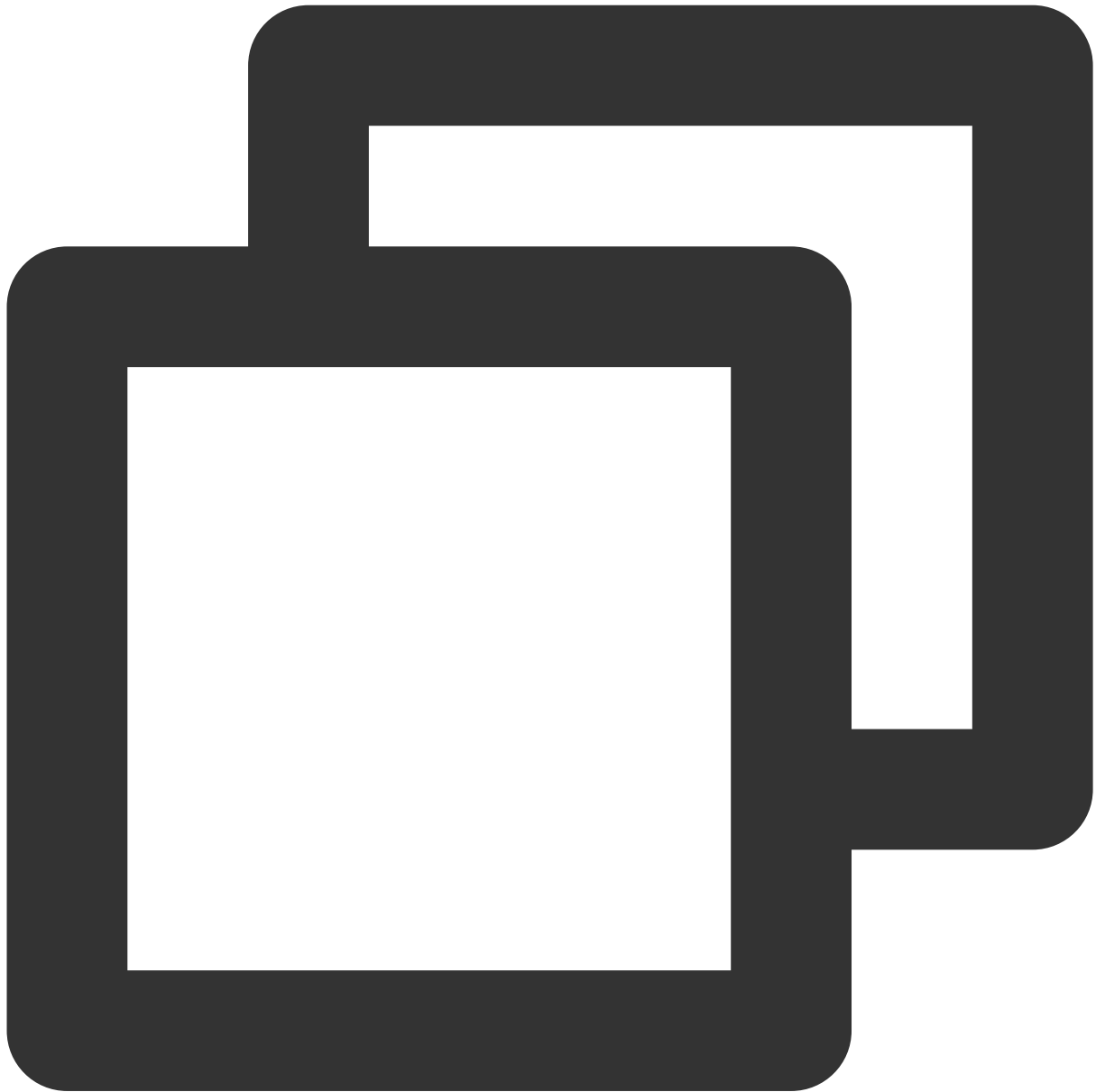
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can create callback configurations through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_config/set_callback?sdkappid=888888888&identifier=admin&users
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

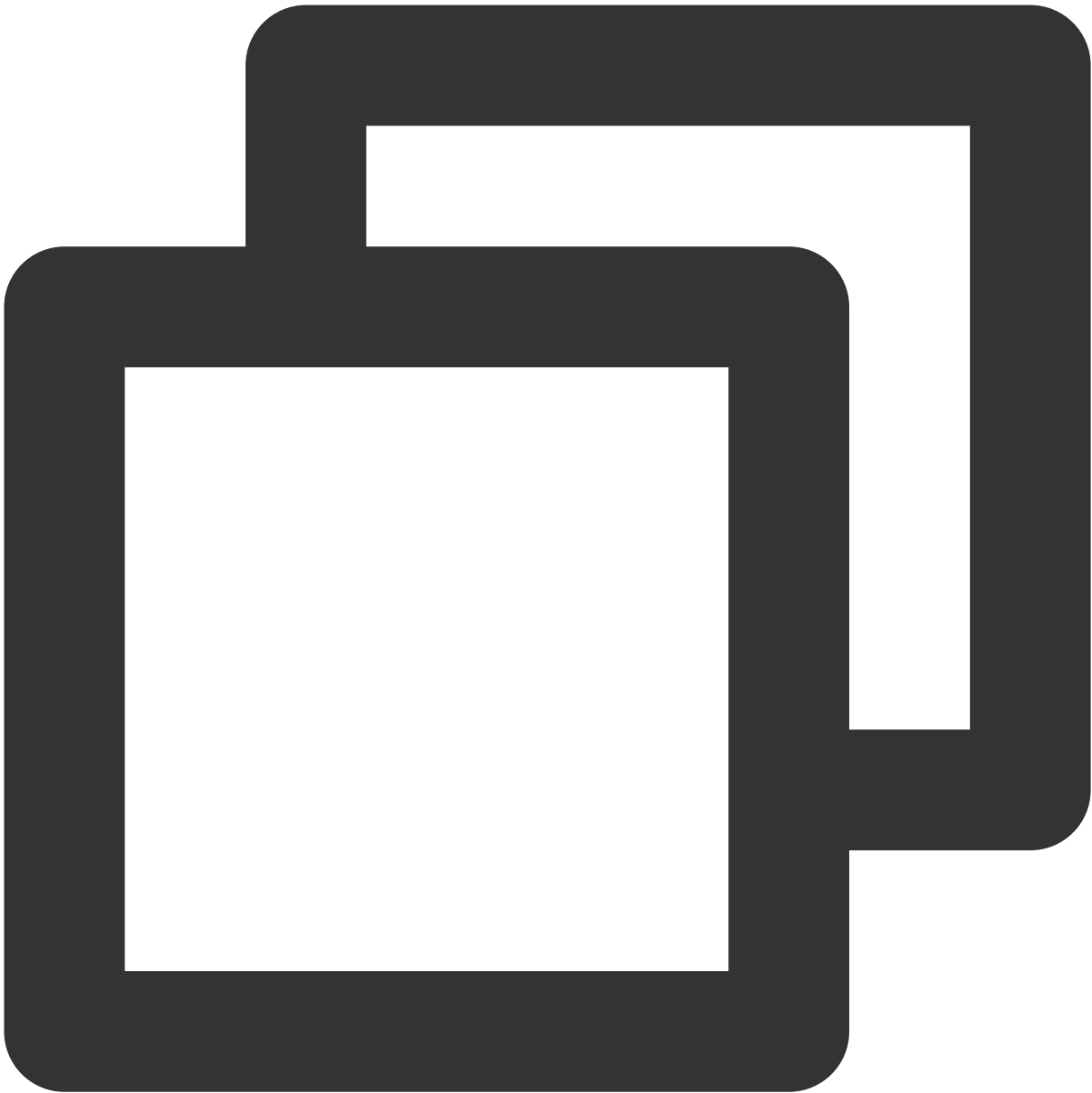
	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_config/set_callback	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "Url": "http://www.example.com/callback",
  "CallbackCommandList": [
    "Room.CallbackAfterMemberLeave",
    "Mic.CallbackAfterSeatInfoChanged"
  ]
}
```

Request Packet Field Description

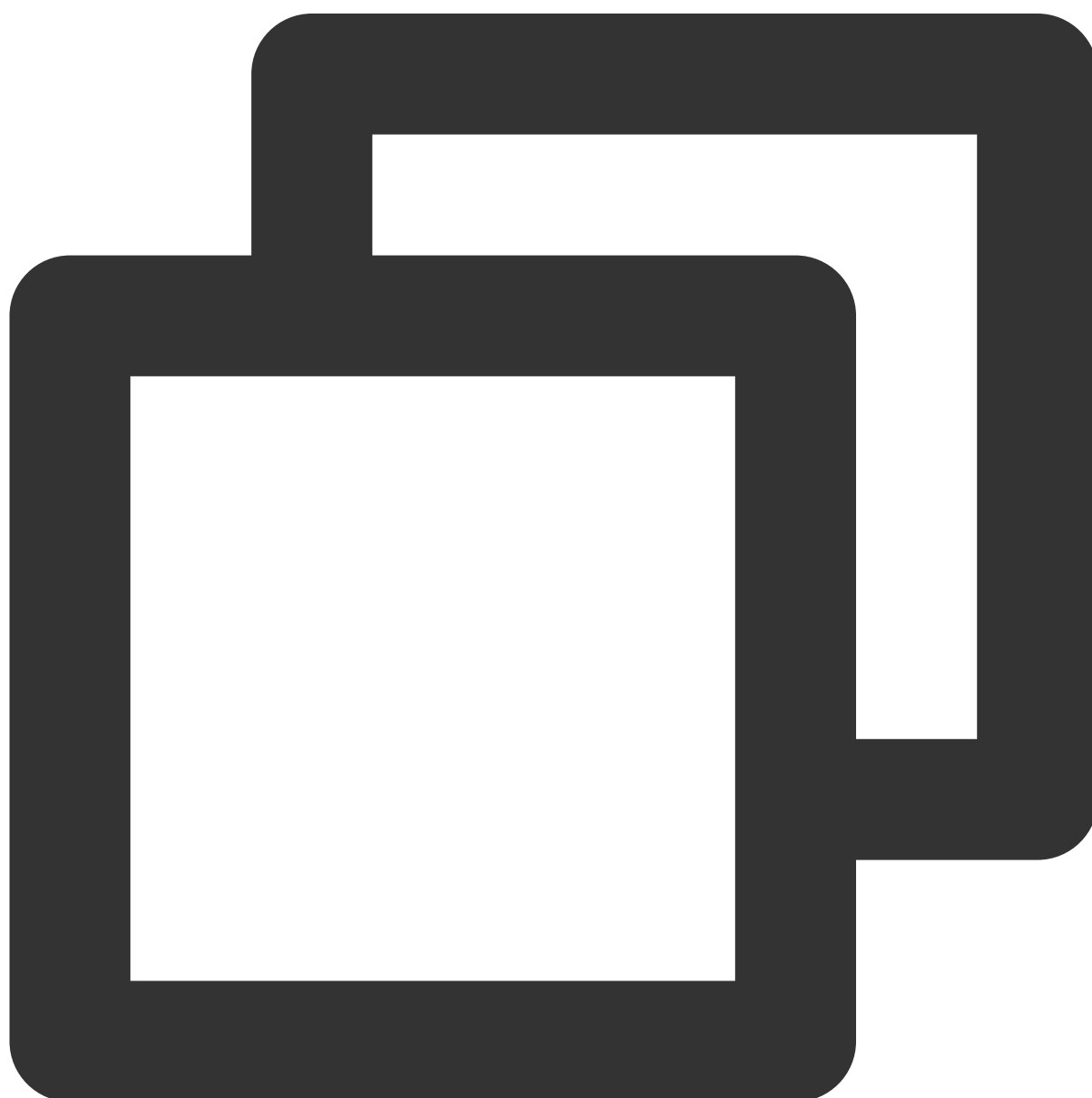
--	--	--	--



Field	Type	Attribute	Description
Url	String	Mandatory	Callback address, must start with http/https, it is recommended to use the more secure https.
CallbackCommandList	Array	Mandatory	For the command words that trigger the callback, refer to <a href="#">Callback Command Word List</a> .

## Sample Response Packets

### Basic Form



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	The parameter is invalid, please check the request for correctness based on the error description.
100300	Callback configuration already exists, you can use the Update Callback Configuration interface to update.

# Update Callback Configuration

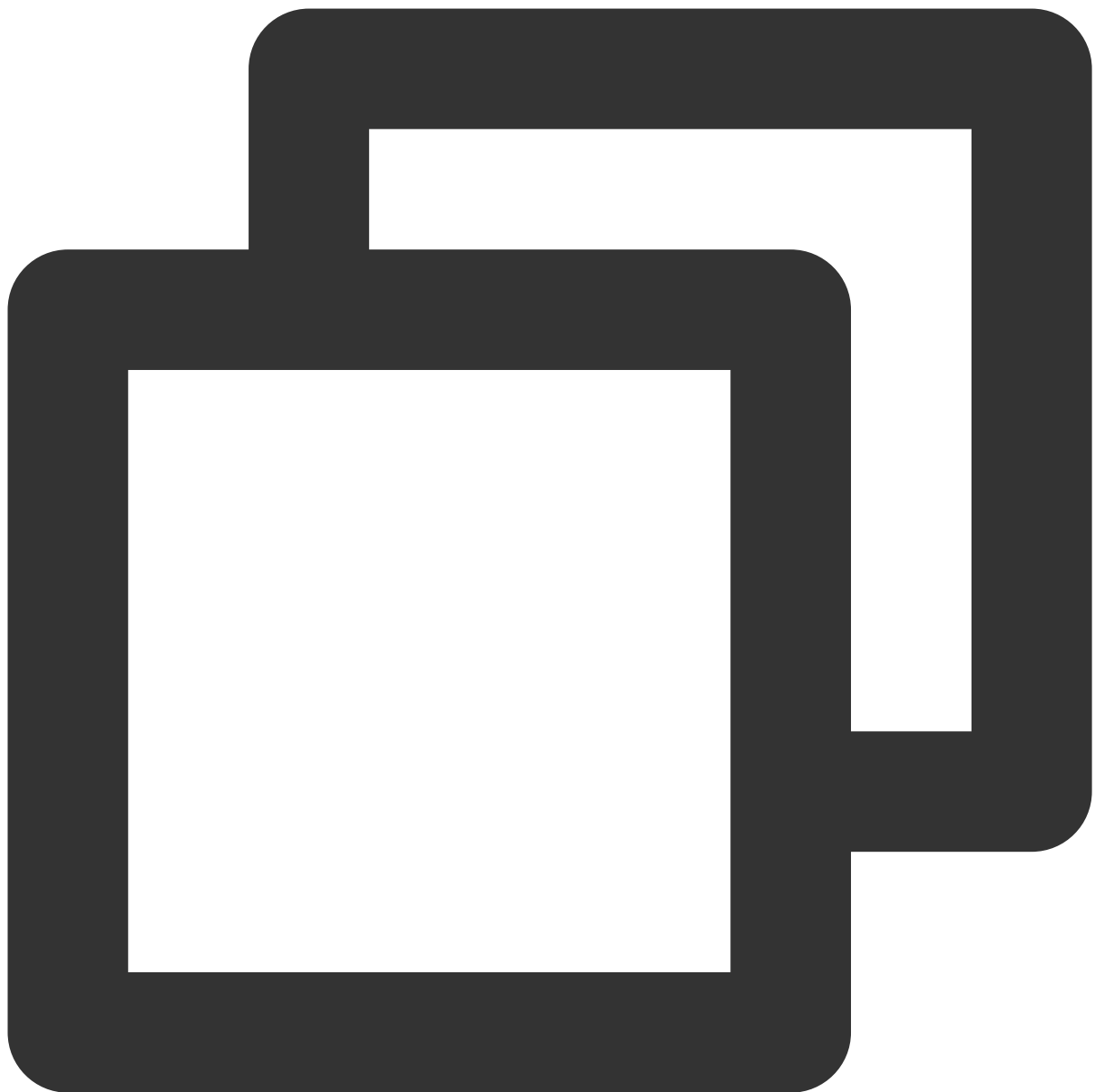
Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can create a callback through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_config/update_callback?sdkappid=888888888&identifier=admin&us
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

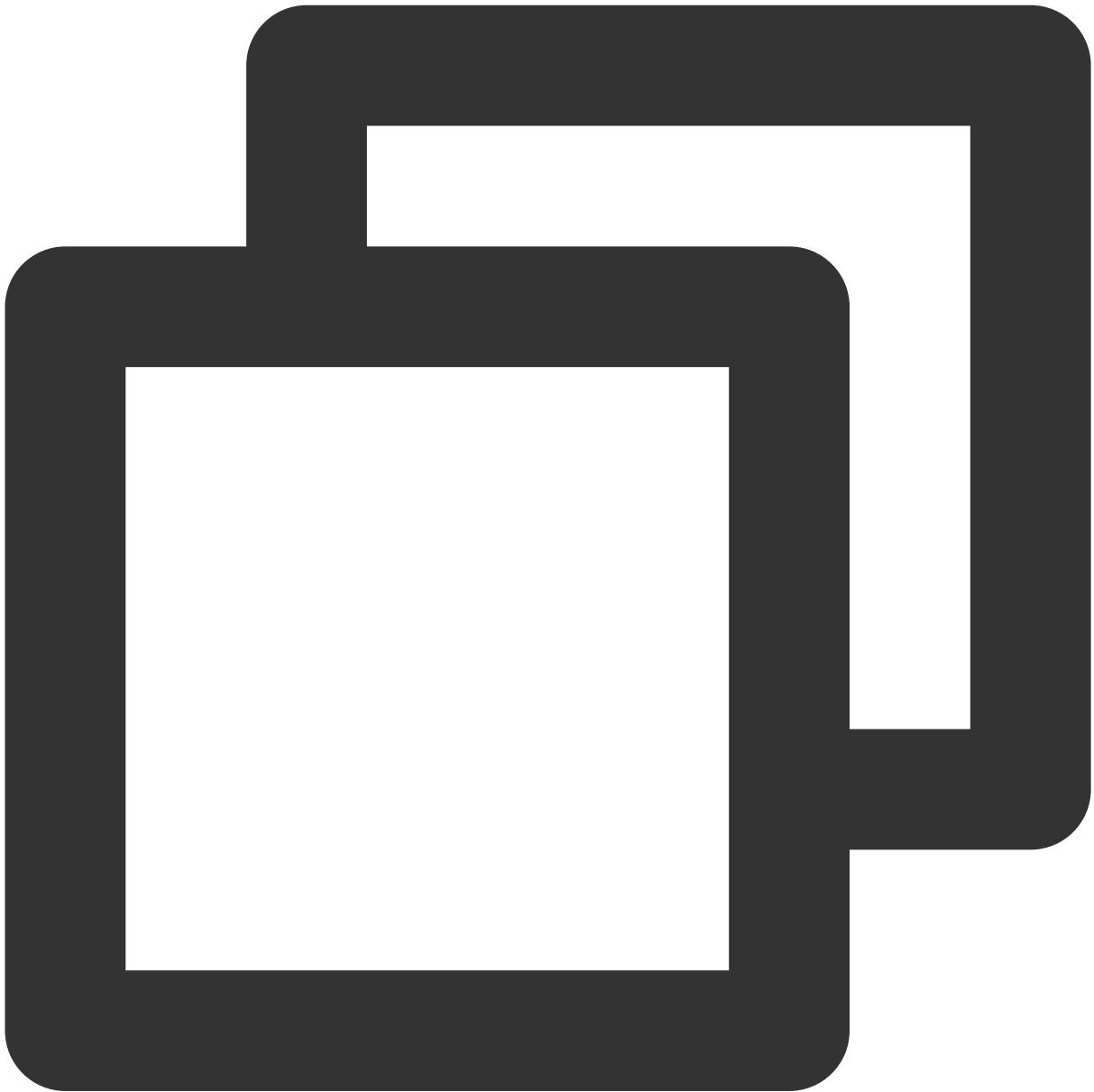
	Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/room_config/update_callback	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form



```
{
  "Url": "http://www.example.com/callback",
  "CallbackCommandList": [
    "Room.CallbackAfterMemberLeave",
    "Mic.CallbackAfterSeatInfoChanged"
  ]
}
```

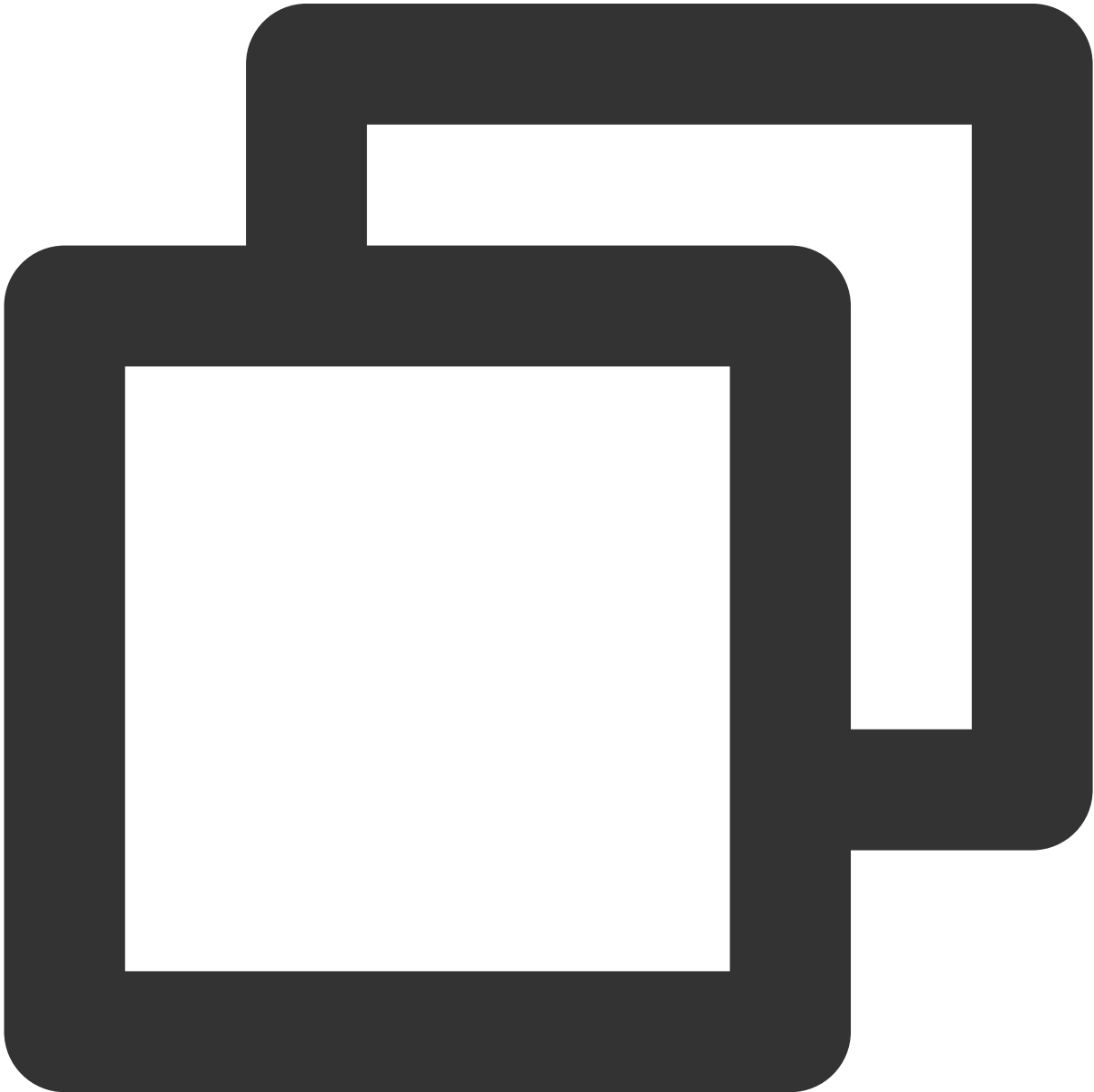
Request Packet Field Description

--	--	--	--

Field	Type	Attribute	Description
Url	String	Mandatory	Callback address, must start with http/https, it is recommended to use the more secure https.
CallbackCommandList	Array	Mandatory	For the command words that trigger the callback, refer to <a href="#">Callback Command Word List</a> .

Sample Response Packets

Basic Form



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	The unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	The parameter is invalid, please check the request for correctness based on the error description.
100301	Callback configuration does not exist, you can create it using the Create Callback Configuration interface.



# Delete Callback Configuration

Last updated : 2024-06-12 11:46:26

## Feature Overview

App administrators can delete the callback configuration through this API.

## API Calling Description

### Sample Request URL



```
https://xxxxxx/v4/room_config/delete_callback?sdkappid=888888888&identifier=admin&us
```

## Request Parameters

The table below only lists the parameters modified when calling this API. For more details, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain name for the country/region where the SDKAppID is located:

	Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/room_config/delete_callback	Request API
sdkappid	The SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App administrator account. For more details, please refer to <a href="#">App Administrator</a> .
usersig	The Signature generated by the App administrator account. For more details, please refer to <a href="#">Generating UserSig</a> .
random	Enter a random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	The request format fixed value is <code>json</code>

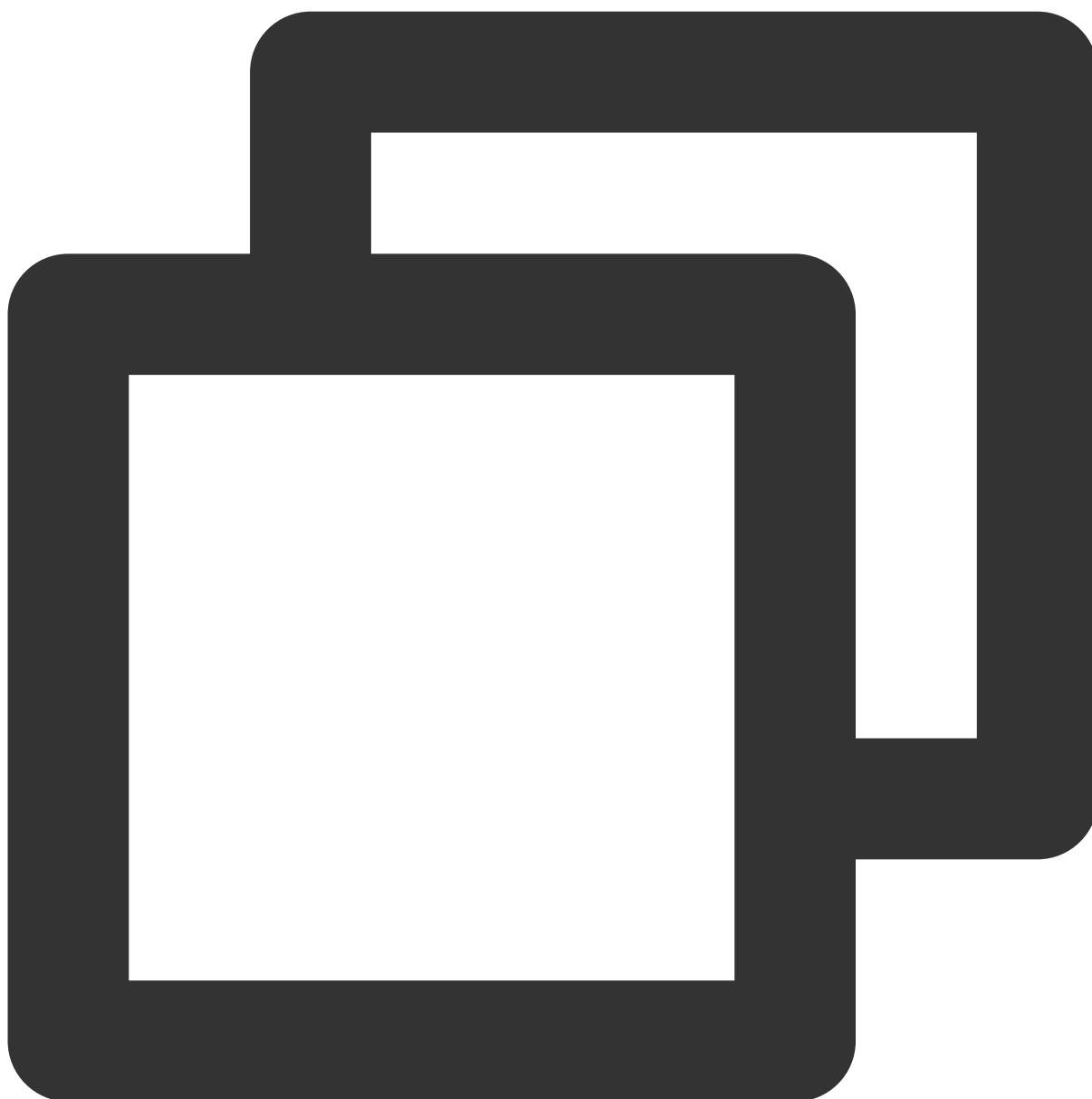
## Maximum Calling Frequency

200 times per second.

## Sample Request Packets

### Basic Form

Just pass it empty.



```
{  
}
```

### Request Packet Field Description

None.

### Sample Response Packets

#### Basic Form



```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-O-Seq-61309"
}
```

Response Packet Field Description

Field	Type	Description
-------	------	-------------

ActionStatus	String	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Error code. 0 for success, others for failure.
ErrorInfo	String	Error message
RequestId	String	UThe unique request ID is returned with each request and required to provide this RequestId when locating issues.

## Error Codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

For common error codes (60000 to 79999), see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry.
100002	The parameter is invalid, please check the request for correctness based on the error description.

# Room Related

## After a Room Is Created

Last updated : 2024-06-12 11:46:26

### Feature Overview

The App backend can view information about the room created by the user in real time through this callback, including notifications of successful room creation in the app backend, allowing for actions such as data synchronization.

### Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the App backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

### Scenarios

App users successfully create a room through the client.

App administrators successfully create a room through the RESTful API.

### Callback Trigger Time

After the room is created successfully.

### API Description

#### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com`.

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL



SdkAppid	The SDKAppID assigned by the Chat console when an app is created
CallbackCommand	Fixed as Room.CallbackAfterCreateRoom
contenttype	Fixed value is <code>JSON</code>
ClientIP	Client IP, format such as 127.0.0.1
OptPlatform	Client platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Room.CallbackAfterCreateRoom",
  "Operator_Account": "admin",
  "RoomInfo": {
    "RoomId": "tandy-test-rest",
    "RoomName": "tandy-test-rest",
    "RoomType": "Meeting",
    "Owner_Account": "user3",
    "MaxMemberCount": 300,
    "MaxSeatCount": 16,
    "IsVideoDisabled": true,
  }
}
```

```
"IsAudioDisabled":true,
"IsMessageDisabled":true,
"IsScreenSharingDisabled":true,
"IsCloudRecordingDisabled":true,
"CustomInfo":"custom123",
"ScheduleStartTime":1703589922,
"ScheduleEndTime":1703593522,
"RoomStatus":"Running",
"IsSeatEnabled":true,
"TakeSeatMode":"ApplyToTake",
"CreateTime":1703589922
},
"ScheduleInviteeList_Account":["user2", "user3", "user3"],
"EventTime":1703589922780
}
```

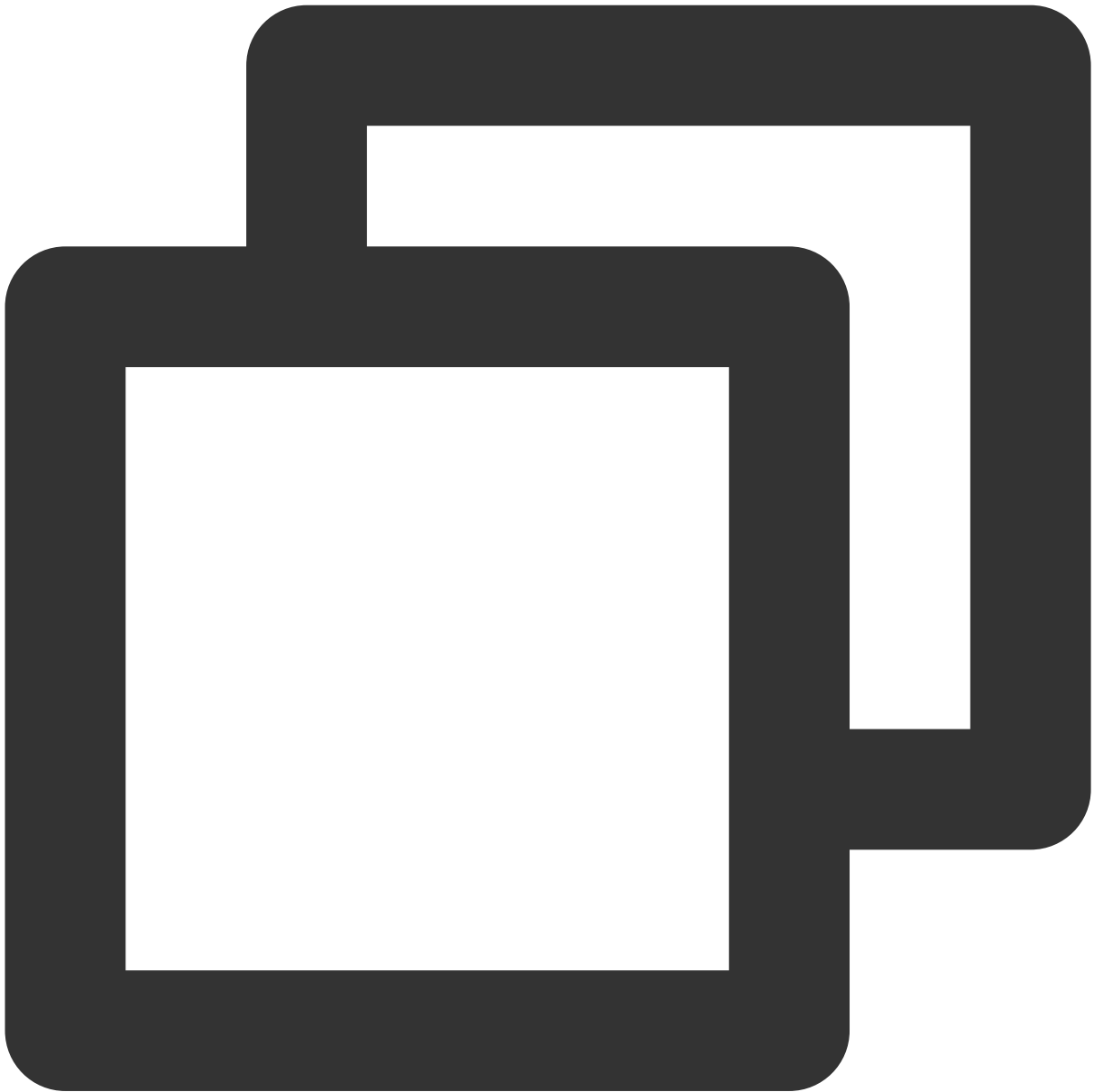
## Request Packet Field Description

Field	Type	Description
CallbackCommand	String	Callback command
Operator_Account	String	Operator UserID initiating the request to create a group
RoomId	String	Room ID
RoomName	String	Room name
RoomType	String	Room type: conference (meeting room)
Owner_Account	String	Host ID
MaxMemberCount	Integer	Maximum number of room members
ScheduleStartTime	Integer	Scheduled meeting start time
ScheduleEndTime	Integer	Scheduled meeting end time
IsVideoDisabled	Bool	Mute all video
IsAudioDisabled	Bool	Mute all audio
IsMessageDisabled	Bool	Disable all members from sending text messages
IsScreenSharingDisabled	Bool	Disable screen sharing
IsCloudRecordingDisabled	Bool	Disable cloud recording

CustomInfo	String	Custom definition fields
RoomStatus	String	Room Status: None, NotStarted, Running
IsSeatEnabled	Bool	Whether to support seat position.
MaxSeatCount	Integer	Maximum number of seats
TakeSeatMode	String	Seat Mode: None (off), FreeToTake (open mic), ApplyToTake (mic on request)
CreateTime	Integer	Scheduled meeting start time
EventTime	Integer	Event trigger timestamp in milliseconds

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

Response Packet Field Description

Field	Type	Attribute	Description

ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)

RESTful API:[Create a Room](#)

# After a Room Is Destroyed

Last updated : 2024-06-12 11:46:26

## Feature Overview

The App backend can view the destroy dynamics of the room in real time through this callback, including: real-time recording of the room destroy (for example, recording logs or synchronization to other systems).

## Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the App backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

## Scenarios

App users successfully destroy the room through the client.

App administrators successfully destroy the room through the RESTful API.

## Callback Trigger Time

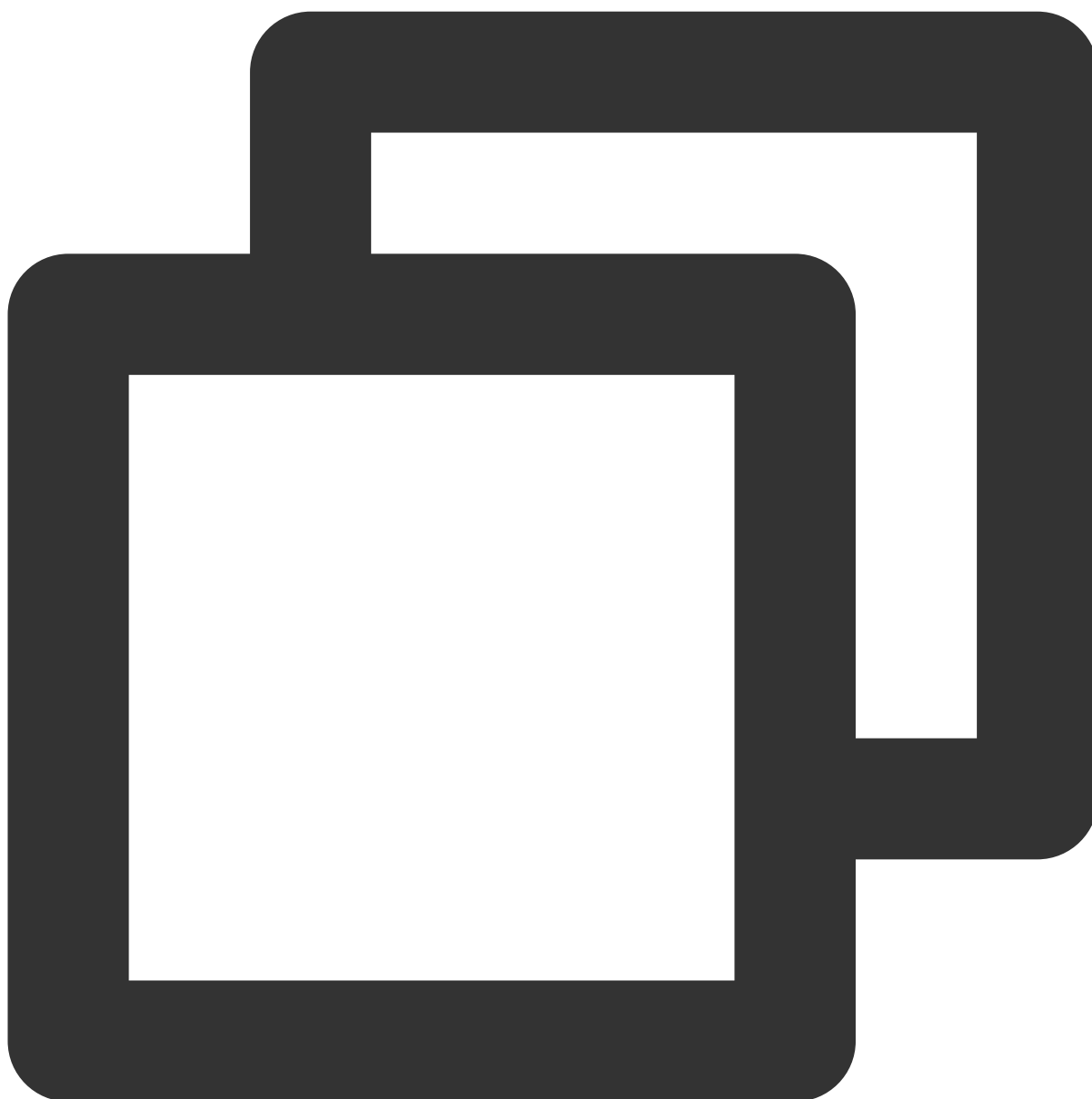
After the room is successfully destroyed.

## API Description

### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com`.

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

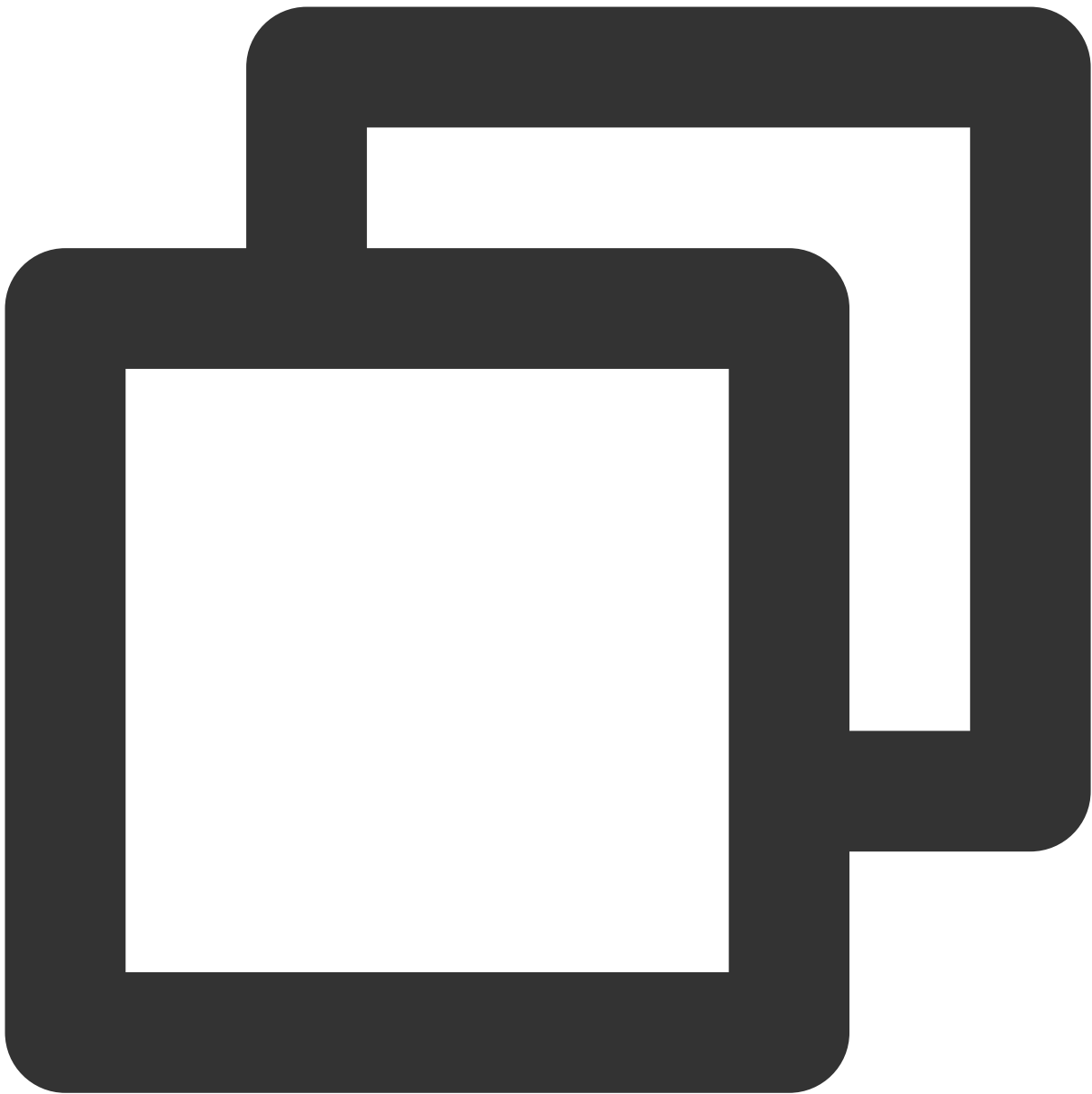
## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL



SdkAppid	The SDKAppID assigned by the Chat console when an app is created
CallbackCommand	Fixed as Room.CallbackAfterDestroyRoom
contenttype	Fixed value: JSON
ClientIP	Client IP, format such as 127.0.0.1
OptPlatform	Client platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Room.CallbackAfterDestroyRoom",
  "Operator_Account": "admin",
  "RoomId": "tandy-test-rest",
  "EventTime": 1703589922780
}
```

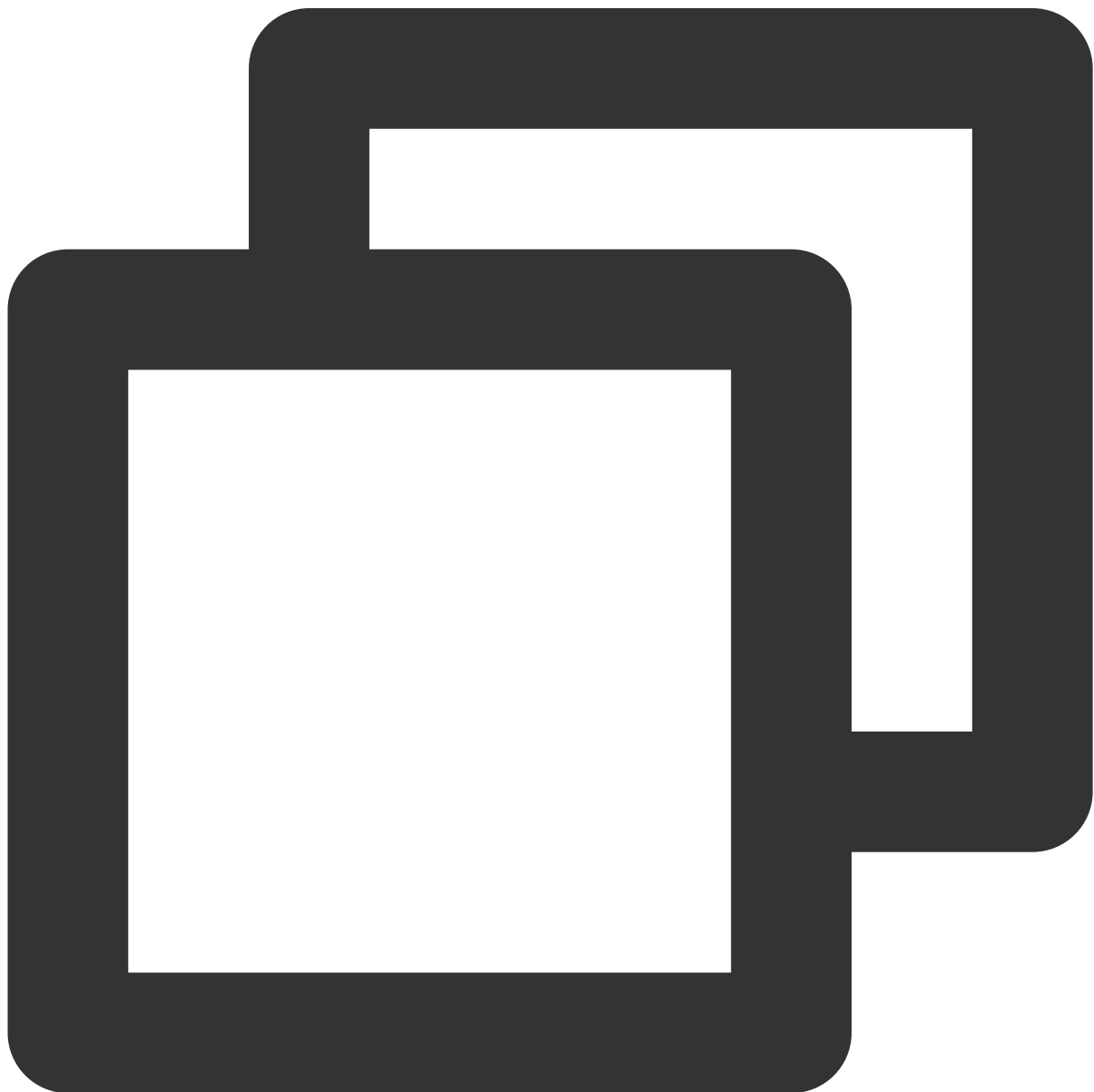
Request Packet Field Description

Field	Type	Description
-------	------	-------------

CallbackCommand	String	Callback command
Operator_Account	String	UserID of the operator initiating the request to destroy a room
RoomId	String	Room ID
EventTime	Integer	Event trigger timestamp in milliseconds

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

## Response Packet Field Description

Field	Type	Attribute	Description
ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)

RESTful API:[Destroy a Room](#)

# After the Room Information Is Updated

Last updated : 2024-06-12 11:46:26

## Feature Overview

The App backend can view changes in room information in real time through this callback, including real-time recording of changed room information (for example, recording logs or synchronization to other systems).

## Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the App backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

## Scenarios

App users update the room information through the client.

App administrators update the room information through the RESTful API.

## Callback Trigger Time

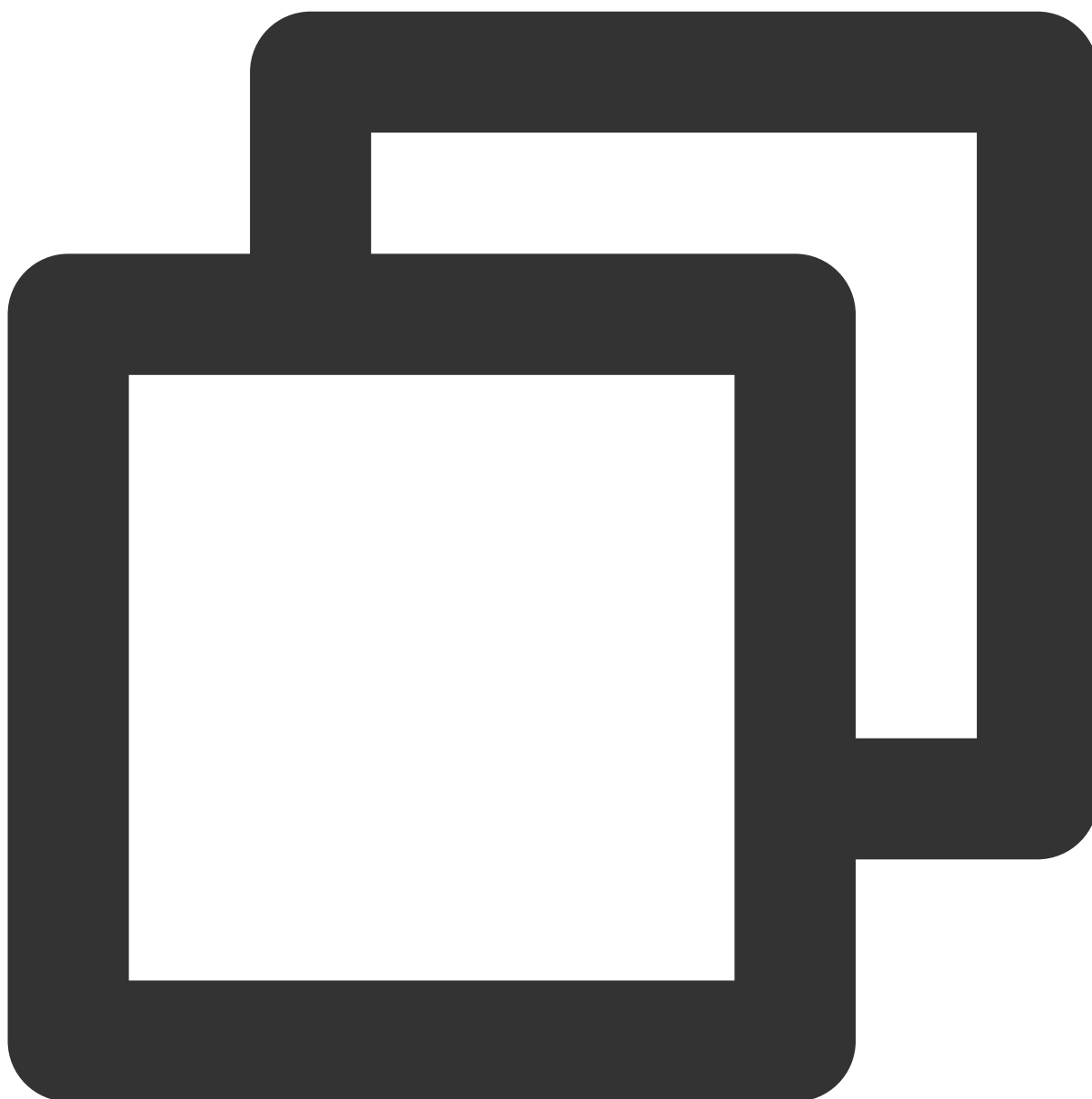
After the room information is successfully updated.

## API Description

### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com`.

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL

SdkAppid	The SDKAppID assigned by the Chat console when an application is created
CallbackCommand	Fixed as Room.CallbackAfterUpdateRoomInfo
contenttype	Fixed value: JSON
ClientIP	Client IP, format such as 127.0.0.1
OptPlatform	Client platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Room.CallbackAfterUpdateRoomInfo",
  "Operator_Account": "bob",
  "RoomInfo" : {
    "RoomId": "rid-123",
    "RoomName" : "rname-123",
    "Owner_Account" : "jack",
    "TakeSeatMode" : "FreeToTake",
    "MaxMemberCount" : 300, // Maximum room capacity
    "IsVideoDisabled" : false, // Whether to enable video for all, default false
    "IsAudioDisabled" : false, // Whether to enable mute for all, default false
  }
}
```



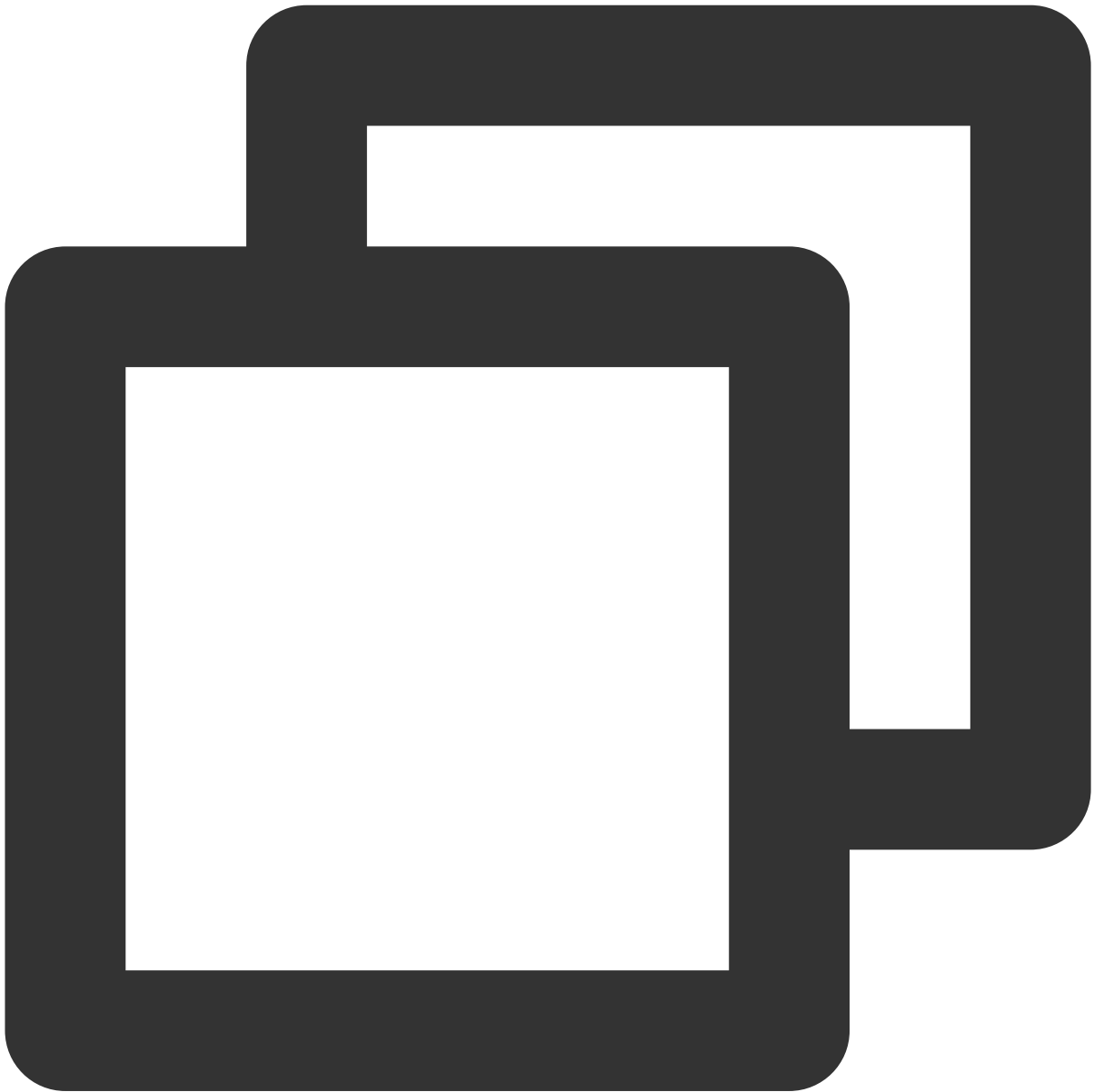
```
"IsMessageDisabled" : false, // Whether to disable sending text messages, d
"IsScreenSharingDisabled" : false, // Whether to disable screen sharing, de
"IsCloudRecordingDisabled" : "false", // Cloud recording is not disabled, d
"CustomInfo" : "123", // Room custom information
}
"EventTime":1670574414123// Millisecond level, event trigger timestamp
}
```

## Request Packet Field Description

Field	Type	Description
CallbackCommand	String	Callback command
Operator_Account	String	Operator UserID initiating the request to create a group
RoomId	String	Room ID
RoomName	String	Room name
Owner_Account	String	Host ID
MaxMemberCount	Integer	Maximum number of room members
IsVideoDisabled	Bool	Mute all video
IsAudioDisabled	Bool	Mute all audio
IsMessageDisabled	Bool	Disable all members from sending text messages
IsScreenSharingDisabled	Bool	Disable screen sharing
IsCloudRecordingDisabled	Bool	Disable cloud recording
CustomInfo	String	Custom definition fields
EventTime	Integer	Event trigger timestamp in milliseconds

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

Response Packet Field Description

Field	Type	Attribute	Description

ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)

RESTful API:[Update the Room Information](#)

# User Related

## After a Room Is Entered

Last updated : 2024-06-12 11:46:26

### Feature Overview

The App backend can view room member joining messages in real time through this callback, including: notifying the app backend of a member joining a group, allowing the app to perform necessary data synchronization.

### Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the app backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

### Scenarios

App users actively join a room through the client.

### Callback Trigger Time

After users successfully join a room.

### API Description

#### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com`.

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL

SdkAppid	The SDKAppID assigned by the Chat console when an application is created
CallbackCommand	Set as Room.CallbackAfterMemberEnter
contenttype	Fixed value: JSON
ClientIP	Client IP, format such as 127.0.0.1
OptPlatform	Client platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Room.CallbackAfterMemberEnter",
  "Operator_Account": "user1",
  "RoomId": "rid-123",
  "MemberCount": 20,
  "Type": "Enter", // Method of joining a room: Enter (join by oneself)
  "MemberList_Account": ["user1"],
  "EventTime": 1670574414123 // Millisecond level, event trigger timestamp
}
```

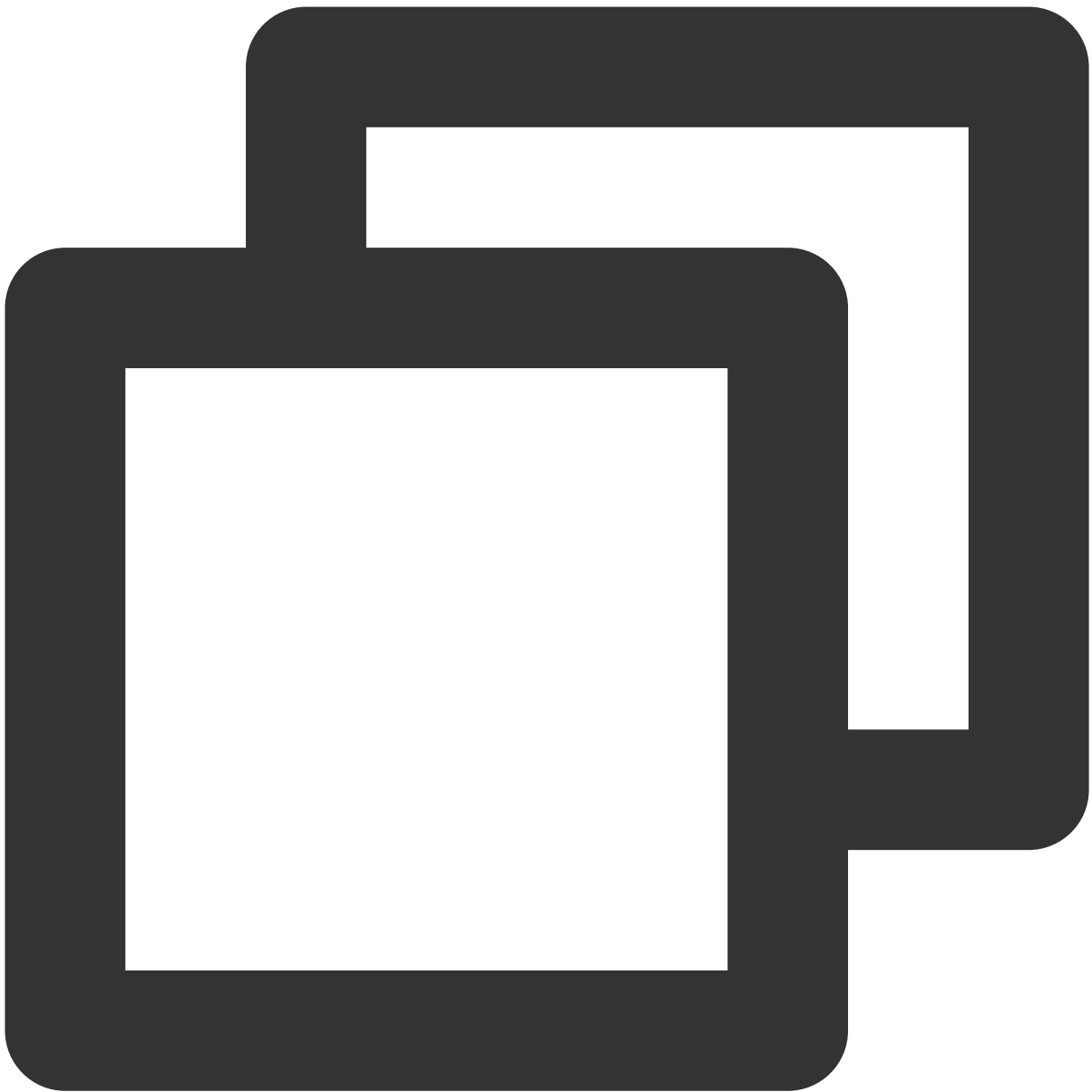
## Request Packet Field Description

Field	Type	Description
CallbackCommand	String	Callback command
Operator_Account	String	UserID of the requester
RoomId	String	Room ID
MemberCount	Integer	Room capacity
Type	String	Method of joining the room: Enter (join by self)
MemberList_Account	Array	List of room members
EventTime	Integer	Event trigger timestamp in milliseconds

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.





```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

Response Packet Field Description

Field	Type	Attribute	Description

ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)

# After a Room Is Left

Last updated : 2024-06-12 11:46:26

## Feature Overview

The App backend can view the messages of room members leaving in real-time through this callback, including: notifying the app backend that a member has left the room, allowing the app to perform necessary data synchronization.

## Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the app backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

## Scenarios

App users actively leave the room through the client.

App users are kicked out of the room.

App users are forced to leave the room due to heartbeat expiration.

## Callback Trigger Time

After users successfully leave the room.

## API Description

### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com` .

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL

SdkAppid	The SDKAppID assigned by the Chat console when an application is created
CallbackCommand	Fixed as Room.CallbackAfterMemberLeave
contenttype	Fixed value: JSON
ClientIP	Client IP, format such as 127.0.0.1
OptPlatform	Client platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Room.CallbackAfterMemberLeave",
  "Operator_Account": "user1",
  "RoomId": "rid-123",
  "MemberCount": 20,

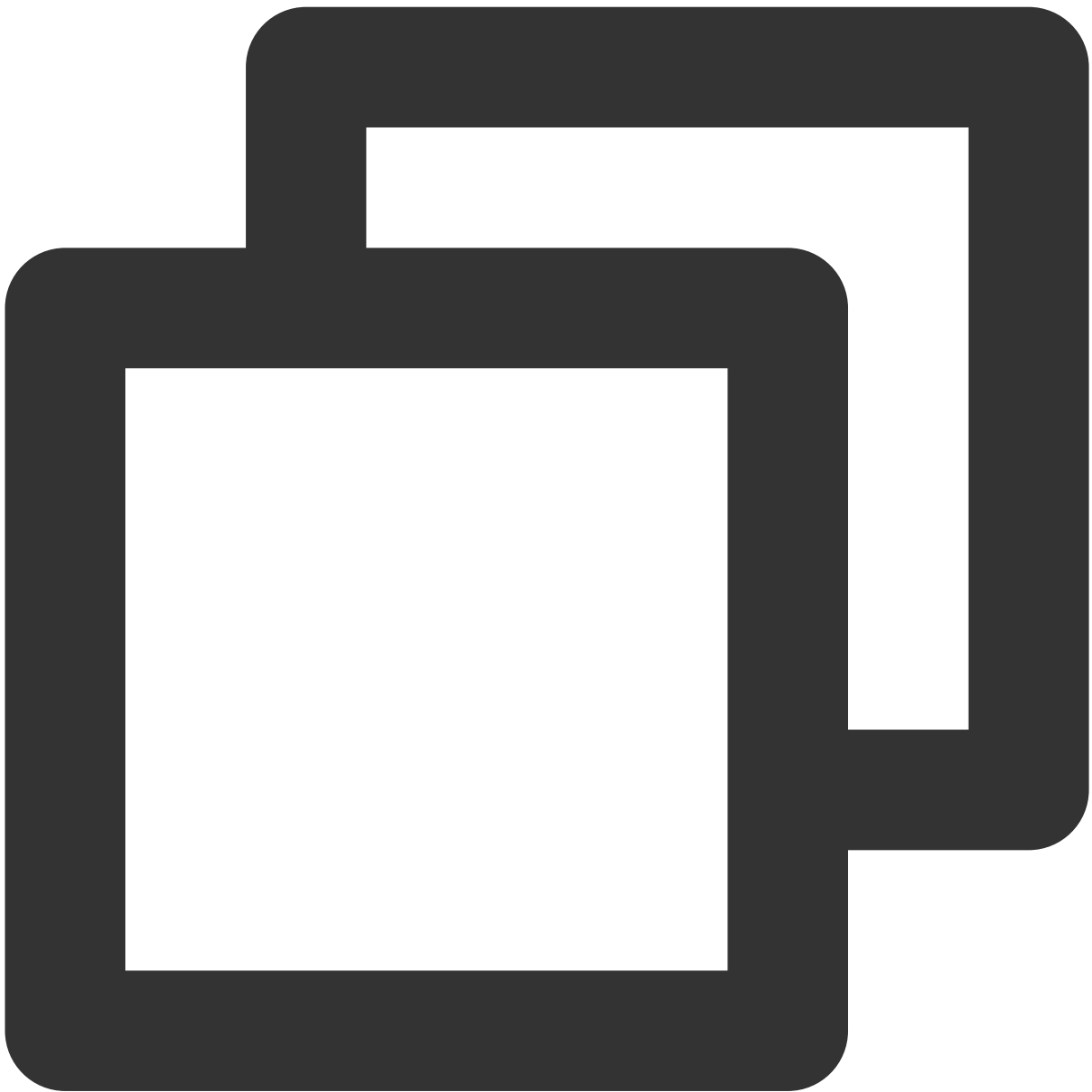
  "Type": "Leave",
  "MemberList_Account": ["user1"],
  "Reason": "xxxx",
  "EventTime": 1670574414123// Millisecond level, event trigger timestamp
}
```

## Request Packet Field Description

Field	Type	Description
CallbackCommand	String	Callback command
Operator_Account	String	UserID of the requester
RoomId	String	Room ID
MemberCount	Integer	Room capacity
Type	String	Check-out method: Leave (self left); Kicked (kicked out); Expired (heartbeat expired)
MemberList_Account	Array	Exit room member list
Reason	String	Reason for check-out
EventTime	Integer	Event trigger timestamp in milliseconds

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

Response Packet Field Description

Field	Type	Attribute	Description



ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)

# Seat Connection Related

## After the Seat List Is Changed

Last updated : 2024-06-12 11:46:26

### Feature Overview

The App backend can view position list change messages in real time through this callback.

### Notes

To enable the callback, a callback URL must be configured and the switch corresponding to this callback protocol activated. For configuration methods, see [Third-party Callback Configuration](#) document.

The direction of the callback is from the Room backend to the App backend via an HTTP POST request.

After receiving the callback request, the app backend must check whether the SDKAppID contained in the request URL is consistent with its own SDKAppID.

### Scenarios

Pick App user on the seat/Kick user off the seat.

Lock the seat position.

### Callback Trigger Time

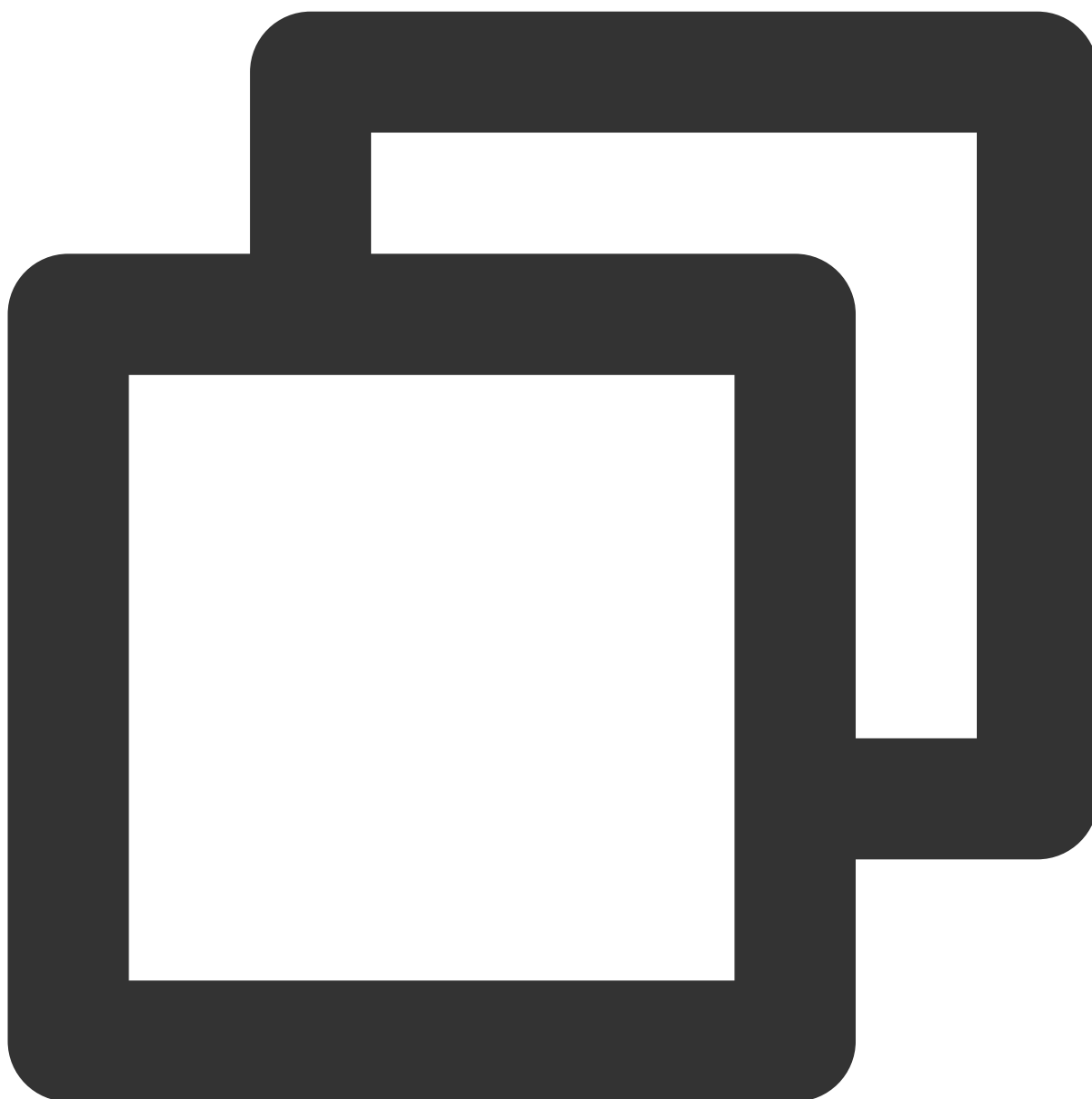
After the seat position information changes.

### API Description

#### Sample Request URL

In the following example, the callback URL configured in the app is `https://www.example.com`.

**Example:**



```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&content
```

## Request Parameters

Parameter	Description
https	The request protocol is HTTPS, and the request method is POST
www.example.com	Callback URL

SdkAppid	The SDKAppID assigned by the Chat console when an app is created
CallbackCommand	Fixed as Mic.CallbackAfterSeatInfoChanged
contenttype	Fixed value: JSON
ClientIP	Client IP, such as 127.0.0.1
OptPlatform	Client Platform. For the value, refer to the meaning of the OptPlatform parameter of <a href="#">Webhook Overview: Callback Protocol</a> .

## Sample Request Packets



```
{
  "CallbackCommand": "Mic.CallbackAfterSeatInfoChanged",
  "Operator_Account": "user1",
  "RoomId": "rid-123",
  "SeatList": [
    {
      // Seat Number
      "Index": 1,
      // If the seat is currently occupied, return the user's ID
      "Member_Account": 144115233775727695,
      // false: Mic access allowed true: Mic access prohibited
    }
  ]
}
```

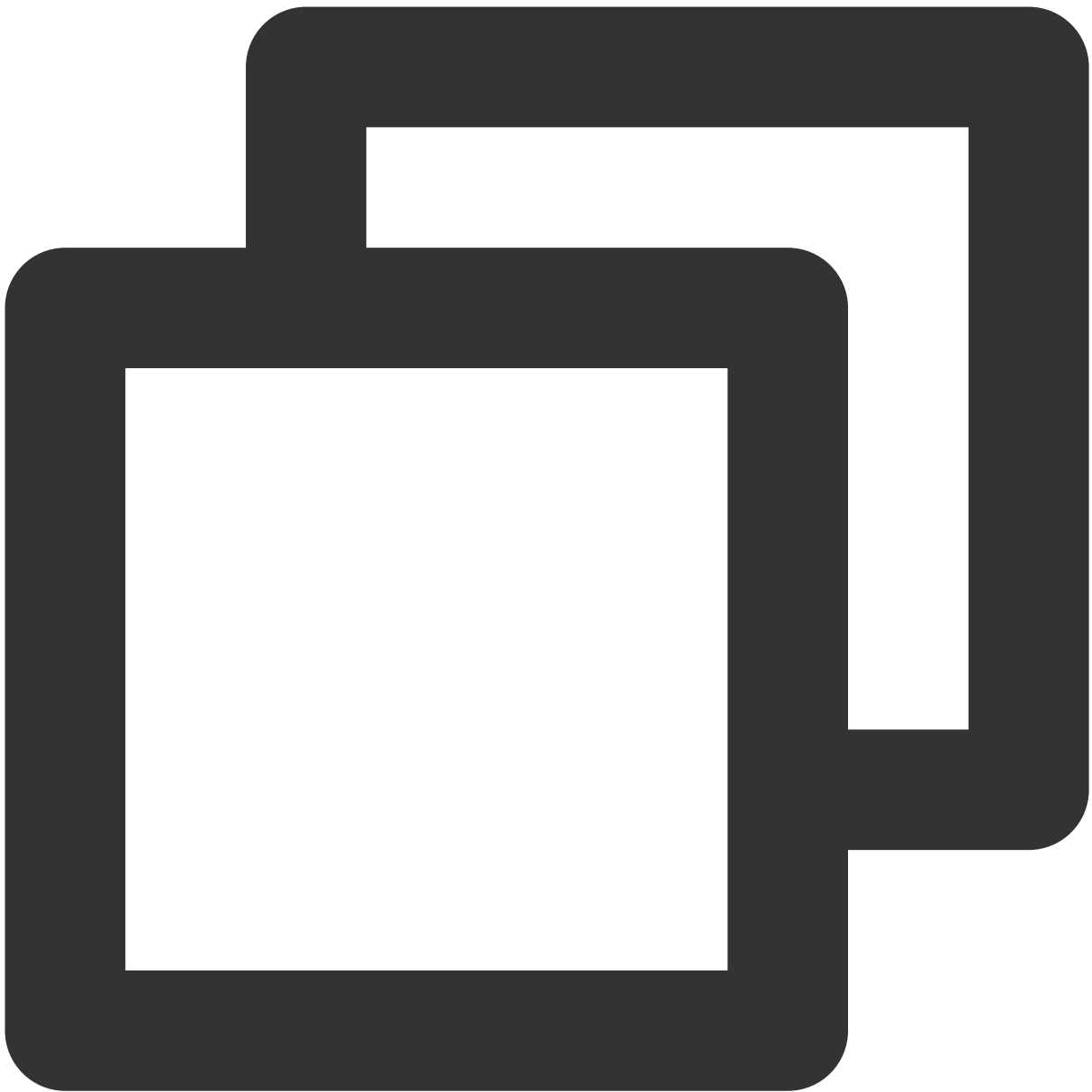
```
        "IsTakenDisabled": false,  
        // false: Pushing video stream allowed true: Pushing video stream proh  
        "IsVideoDisabled": false,  
        // false: Allow audio stream push true: Prohibit audio stream push  
        "IsAudioDisabled": false  
    }  
]  
"EventTime":1670574414123// Millisecond level, event trigger timestamp  
}
```

## Request Packet Field Description

Field	Type	Description
CallbackCommand	String	Callback command
Operator_Account	String	UserID of the requester
RoomId	String	Room ID
SeatList	Array	The list of the seat
Index	Integer	The number of the seat
Member_Account	String	User ID on the seat position, empty indicates no user on the seat
IsTakenDisabled	Bool	Lock the seat position
IsVideoDisabled	Bool	Disable microphone video stream
IsAudioDisabled	Bool	Disable microphone audio stream

## Sample Response Packets

A callback response packet is sent after the app backend synchronizes the data.



```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // Ignore callback result
}
```

Response Packet Field Description

Field	Type	Attribute	Description

ActionStatus	String	Mandatory	The result of the request process. OK for success, FAIL for failure.
ErrorCode	Integer	Mandatory	Error code, 0 means to ignore the response result
ErrorInfo	String	Mandatory	Error message

## Reference

[Webhook Overview](#)



# FAQs (TUIRoomKit)

## Web

Last updated : 2023-10-30 11:03:31

### Environment-related Issues

#### What platforms does TUIRoomKit Web support?

Please refer to [TRTC Web SDK's browser support](#) for TUIRoomEngine Web supported platforms.

For environments not listed above, you can open the [TRTC Capability Test](#) in your current browser to test if it fully supports WebRTC features.

#### Why can TUIRoomKit be used normally in local development testing, but not when deployed online?

Considering user safety and privacy issues, browsers restrict web pages to only capture mic and Camera in secure environments (such as https, localhost, file:// protocols). HTTP protocol is insecure, and browsers will prohibit media device capturing under HTTP protocol.

If everything works fine in your local development testing, but you cannot capture Camera and mic after deploying the page, please check if your web page is deployed on HTTP protocol. If so, please use HTTPS to deploy your web page and ensure a qualified HTTPS security certificate.

For more details, please refer to [the URL domain and protocol restrictions description](#).

#### Does TUIRoomKit Web support integration with iframe?

Yes, it does. To integrate TUIRoom Web in an iframe, you need to add attributes to the iframe tag to enable related permissions, as shown below.

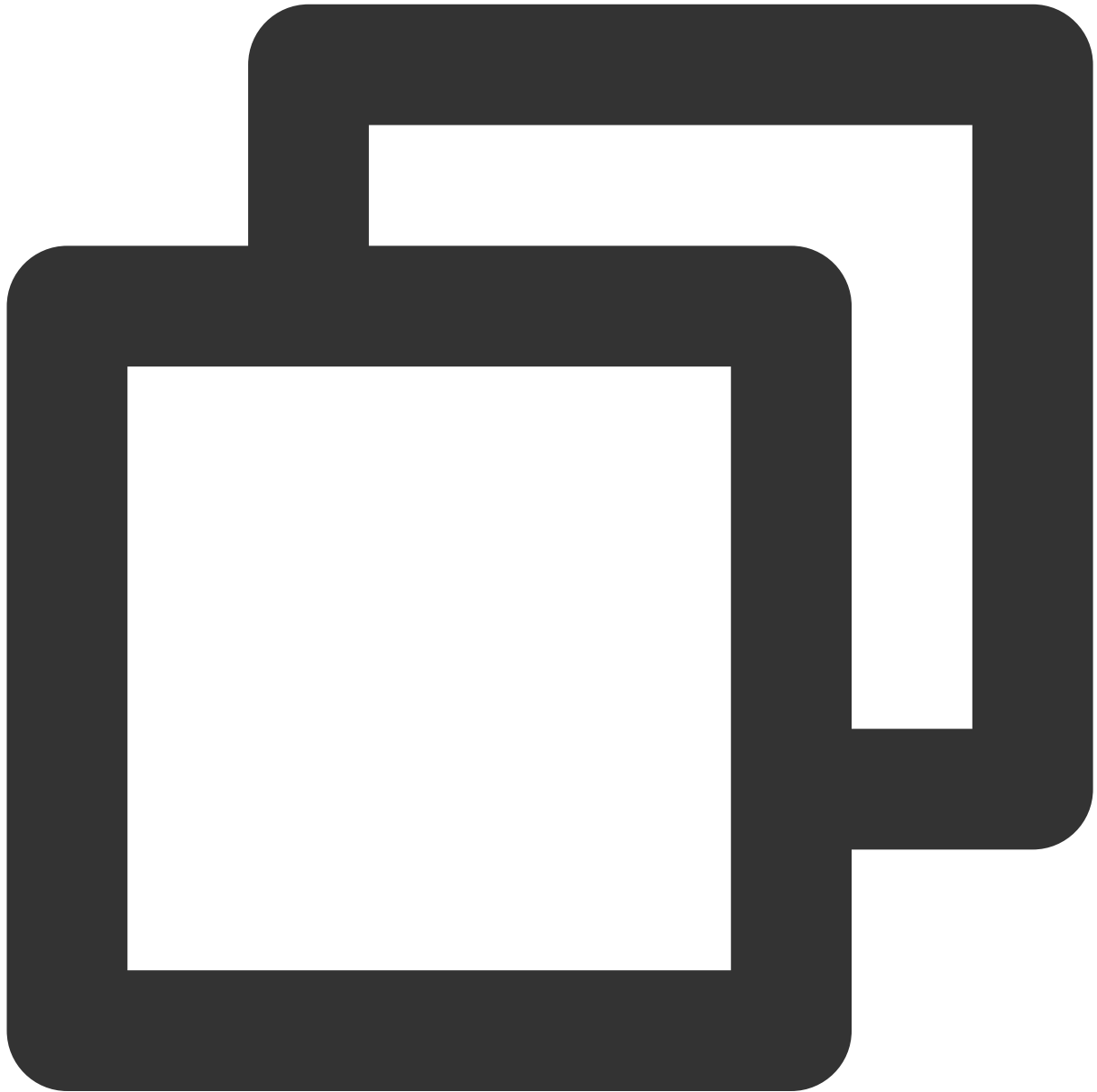


```
// Enable mic, Camera, Screen Sharing, and full-screen permissions
<iframe allow="microphone; camera; display-capture; display; fullscreen;">
```

## Compilation-related Issues

**Webpack 5 importing TUIRoomEngine SDK error: webpack < 5 used to include polyfills for node.js core modules by default. This is no longer the case. Verify if you need this module and configure a polyfill for it.**

The error is caused by the removal of the automatic inclusion of nodejs core module polyfills in webpack5. You can add `configureWebpack` configuration in `vue.config.js` to solve this problem.



```
module.exports = defineConfig({
  // ...
  configureWebpack: (config) => {
    config.resolve.fallback = {
      ...config.resolve.fallback,
      url: false,
      path: false,
      fs: false,
    }
  }
})
```

```
        crypto: false,  
      };  
    }  
  });
```

## Function-related Issues

### Can `tim-js-sdk` and `@tencentcloud/tuiroom-engine-js` be introduced in the project at the same time?

Yes, they can. You can create a `tim` instance object through `TIM` in `tim-js-sdk` and pass it to the `init` interface when initializing `TUIRoomEngine`.



```
await TUIRoomEngine.init({
  sdkAppId: 0,    // Fill in your applied sdkAppId
  userId: '',     // Fill in your business-related userId
  userSig: '',    // Fill in the userSig calculated by the server or locally
  tim,           // Pass in the tim instance
});
```

Also, you can get the tim instance used internally by TUIRoomEngine through the `roomEngine.getTIM` method.

# iOS

Last updated : 2023-10-30 14:10:38

**I need to modify the UI myself. Every time I update the pod, the source code will be refreshed, causing the modifications to be lost. How should I handle this?**

It is suggested to Fork the [TUIRoomKit repository](#) to your personal GitHub account, and then use the [local pod](#) or [git pod path](#) to reference the corresponding code in your project.

For details, please refer to the [official Pod documentation](#):

## Using the files from a folder local to the machine.

If you would like to develop a Pod in tandem with its client project you can use `:path`.

```
pod 'Alamofire', :path => '~/Documents/Alamofire'
```

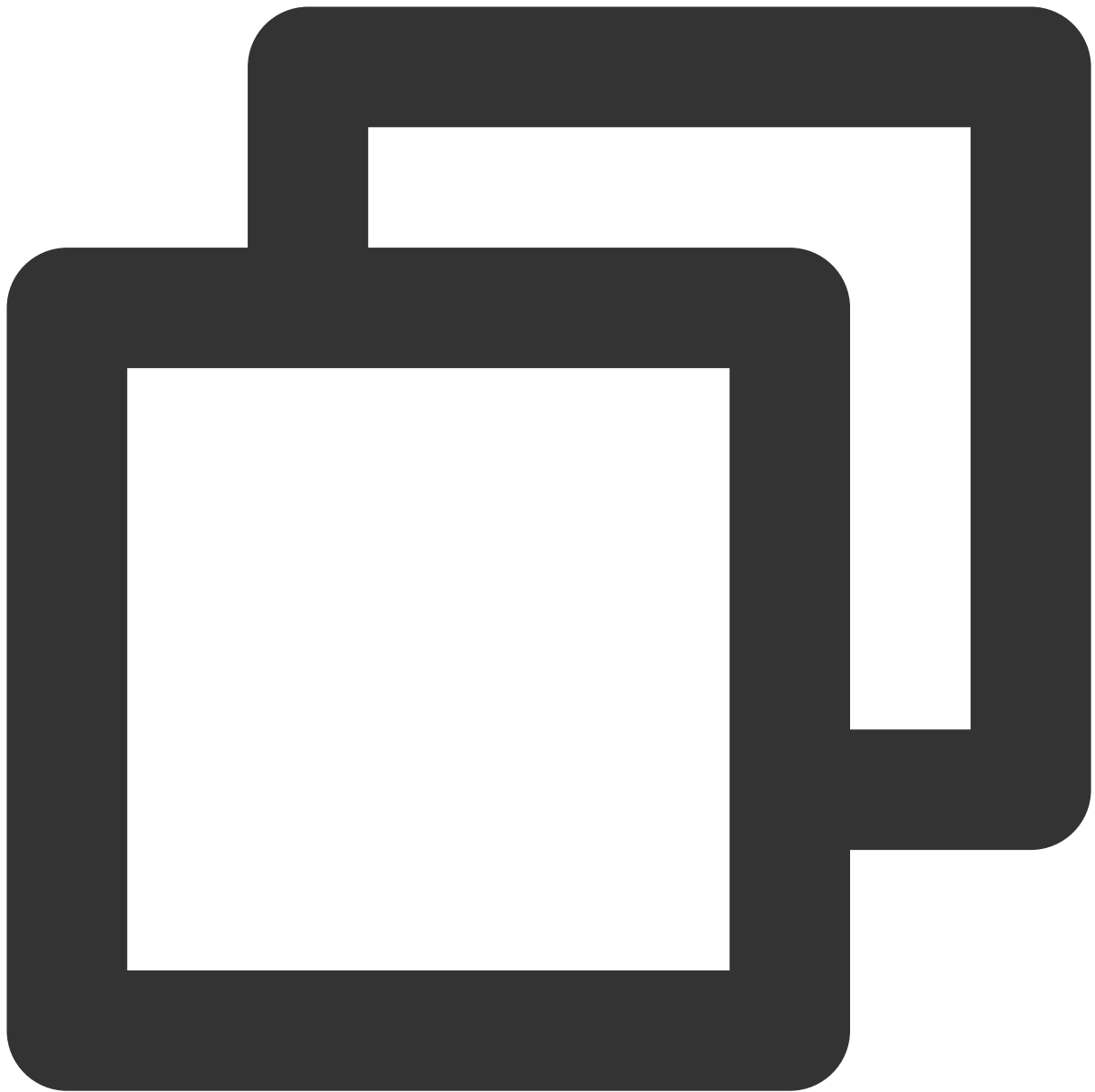
Using this option CocoaPods will assume the given folder to be the root of the Pod and will link the files directly from there in your project. This means that your edits will persist between CocoaPods installations. The referenced folder can be a checkout of your favorite framework or a git submodule of the current repo.

Note that the `podspec` of the Pod file is expected to be in that the designated folder.

## Is there a conflict between TUIRoomKit and the integrated audio and video library?

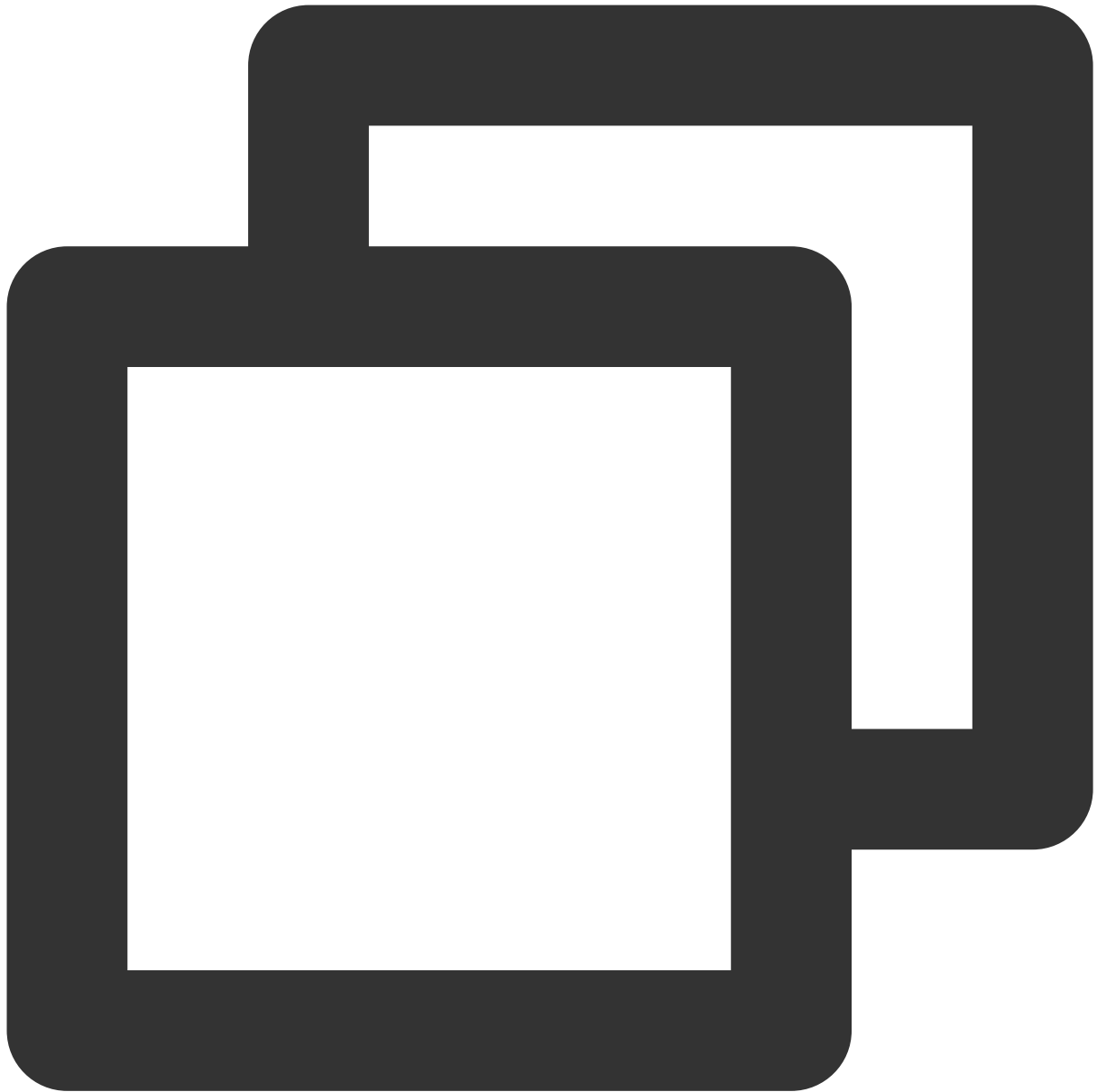
Tencent Cloud's audio and video libraries cannot be integrated at the same time, and there may be symbol conflicts. You can handle it according to the following scenarios.

1. If you are using the `TXLiteAVSDK_TRTC` library, there will be no symbol conflicts. You can directly add dependencies in the Podfile file.



```
pod 'TUIRoomKit'
```

2. If you are using the `TXLiteAVSDK_Professional` library, there will be symbol conflicts. You can add dependencies in the `Podfile` file.



```
pod 'TUIRoomKit/Professional'
```

3. If you are using the `TXLiteAVSDK_Enterprise` library, there will be symbol conflicts. It is recommended to upgrade to `TXLiteAVSDK_Professional` and then use `TUIRoomKit/Professional`.

### How to view TRTC logs?

TRTC logs are compressed and encrypted by default, with the extension `.xlog`. Whether the log is encrypted can be controlled by `setLogCompressEnabled`. The file name containing C(compressed) is encrypted and compressed, and the file name containing R(raw) is plaintext.



iOS&Mac : Sandbox's `Documents/log`

Android :

Versions 6.7 and earlier : `/sdcard/log/tencent/liteav`

Versions after 6.8 : `/sdcard/Android/data/package name/files/log/tencent/liteav/`

Versions after 8.5 : `/sdcard/Android/data/package name/files/log/liteav/`

Windows :

Versions before 8.8 : `%appdata%/tencent/liteav/log`

Versions 8.8 and later : `%appdata%/liteav/log`

Web : Open the browser Console, or use vConsole to record SDK print information.

**Note:**

To view the .xlog file, you need to download the [decryption tool](#) and run it directly in the Python 2.7 environment with the xlog file in the same directory using `python decode_mars_log_file.py`.

To view the .clog file (new log format after version 9.6), you need to download the [decryption tool](#) and run it directly in the Python 2.7 environment with the clog file in the same directory using `python decompress_clog.py`.

# Android

Last updated : 2023-10-30 11:06:45

## What are the requirements for TUIRoomKit's dependencies on TRTC and IM SDK?

TUIRoomEngine will perform version verification when logging in, prompting you to use the correct TRTC SDK and IM SDK Version.

```
W [W] [05-25/20:11:34.781+8.0] [7993,7993] [global_service.cc:239]Tuikit: warning: im recommend to use 7.2.4
W [W] [05-25/20:11:34.781+8.0] [7993,7993] [global_service.cc:248]Tuikit: warning: trtc recommend to use 11.
```

## Can TUIRoomKit remove the dependency on IM SDK?

No, Real-time Conference SDK requires IM for signaling-related communication, including mic control, bullet screen, and other functions, so it needs to have a dependency on IM SDK.

## Does TUIRoomKit support features like conference scheduling?

It is not supported yet, this feature is in planning, you can follow our update log to get the latest information on the launch status of this feature.

## After integrating TUIRoomKit, how can I quickly modify the user name and user avatar?

You can set it by calling the `setSelfInfo` interface of TUIRoomKit.

# Electron

Last updated : 2024-05-30 13:22:58

## Runtime error: "does not provide an export named 'createBlock' "?

If you encounter the following error after following the steps above, please make sure to lock the version of vite to below 3.0.0, and the version of vue to below 3.4.9.

```
/node_modules/.vite/deps/vue.js?v=b083d236' does not provide an export named 'createBlock' (at App.vt
```

The reason is that the electron project template provided supports vite versions specified below 3.0.0, but properties in vue version 3.4.9 and above require an upgrade of the vite version for support. An issue has been submitted to the official vue, <https://github.com/vuejs/core/issues/10177>.

## Is trtc-electron-sdk compatible with the official Electron v12.0.1 version?

Yes, trtc-electron-sdk does not depend on the electron's own SDK specifically, so there are no related version dependencies.

# Flutter

Last updated : 2023-11-21 14:55:33

## An error occurs when running on iOS: Error (Xcode): Library not found for -ld64 ?

Xcode 15 changed the compiled linker , and RoomEngine Flutter was adapted in 1.6.0. Old versions of Xcode cannot be compiled, so just use the latest Xcode .

The project integrates tencent\_calls\_uikit and rtc\_room\_engine at the same time. There will be problems such as the two components interfering with each other when joining the room will interrupt the call. How to Resolve ?

This is due to the exclusivity of the RTC service. We recommend that you use two flags, callState and roomState , to record whether you are currently on a call or in a conference. These two flags are used to isolate the two services. , to avoid this problem. For example, if you are currently on a call, entry into the room is prohibited.

# Error Code (TUIRoomKit)

Last updated : 2023-12-18 18:13:00

## General Error Code

Error Code	Description
0	Operation Successful
-1	Temporarily Unclassified General Error
-2	Request Rate Limited, Please Try Again Later
-1000	Not Found SDKAppID, Please Confirm Application Info in <a href="#">TRTC Console</a>
-1001	Passing illegal parameters when calling API, check if the parameters are legal
-1002	Not Logged In, Please Call Login API
-1003	Failed to Obtain Permission, Unauthorized Audio/Video Permission, Please Check if Device Permission is Enabled
-1004	This feature requires additional Package, please enable the corresponding Package as needed

## Local User Rendering, Video Management, Audio Management API Callback Error Definition

Error Code	Description
-1100	System Issue, Failed to Open Camera. Check if Camera Device is Normal
-1101	Camera has No System Authorization, Check System Authorization
-1102	Camera is Occupied, Check if Other Process is Using Camera
-1103	No Camera Device Currently, Please Insert Camera Device to Solve the Problem
-1104	System Issue, Failed to Open Mic. Check if Mic Device is Normal
-1105	Mic has No System Authorization, Check System Authorization
-1106	Mic is Occupied

-1107	No Mic Device Currently
-1108	Failed to Obtain Screen Sharing Object, Check Screen Recording Permission
-1109	Failed to Enable Screen Sharing, Check if Someone is Already Screen Sharing in the Room

## Room Management Related API Callback Error Definition

Error Code	Description
-2100	Room Does Not Exist When Entering, May Have Been Closed
-2101	This Feature Can Only Be Used After Entering the Room
-2102	Room Owner Does Not Support Leaving the Room, Conference Room Type: Transfer Room Ownership First, Then Leave the Room. Living Room Type: Room Owner Can Only Close the Room
-2103	This Operation is Not Supported in the Current Room Type
-2104	This Operation is Not Supported in the Current Speaking Mode
-2105	Illegal Custom Room ID, Must Be Printable ASCII Characters (0x20-0x7e), Up to 48 Bytes Long
-2106	Room ID is Already in Use, Please Choose Another Room ID
-2107	Illegal Room Name, Maximum 30 Bytes, Must Be UTF-8 Encoding if Contains Chinese Characters
-2108	User is Already in Another Room, Single RoomEngine Instance Only Supports User Entering One Room, To Enter Different Room, Please Leave the Room or Use New RoomEngine Instance

## Room User Information API Callback Error Definition

Error Code	Description
-2200	User Not Found
-2201	User Not Found in the Room

## Room User Speech Management API Callback Error Definition & Room Mic Seat Management API Callback Error Definition

Error	Description
-------	-------------

Code	
-2300	Room Owner Permission Required for Operation
-2301	Room Owner or Administrator Permission Required for Operation
-2310	No Permission for Signaling Request, e.g. Canceling an Invite Not Initiated by Yourself
-2311	Signaling Request ID is Invalid or Has Been Processed
-2340	Maximum Mic Seat Exceeds Package Quantity Limit
-2341	Current User is Already on Mic Seat
-2342	Mic Seat is Already Occupied
-2343	Mic Seat is Locked
-2344	Mic Seat Serial Number Does Not Exist
-2345	Current User is Not on Mic
-2346	Mic-on Capacity is Full
-2360	Current Mic Seat Audio is Locked
-2361	Need to Apply to Room Owner or Administrator to Open Mic
-2370	Current Mic Seat Video is Locked, Need Room Owner to Unlock Mic Seat Before Opening Camera
-2371	Need to Apply to Room Owner or Administrator to Open Camera
-2380	All Members Muted in the Current Room
-2381	You Have Been Muted in the Current Room

# SDK Update Log (TUIRoomKit)

Last updated : 2024-05-24 18:29:56

## April 2024

Publishing dynamics	Update Log	Release time
SDK 2.3.1 Released for iOS, Android, Windows, Web, Uniapp and Electron.	<b>RoomEngine SDK</b> Full platform: Support live broadcast scenarios with maximum room count of 10w+. Add onRoomUserCountChanged callback. Add enterRoom interface with RoomType. TUISeatInfo structure has new fields for avatar and nickname. TUIRequest structure has new fields for avatar and nickname. TUIUserInfo structure now has new isMessageDisabled field. Android&iOS : Fix startPlayRemoteVideo memory leak. Add OnKickedOffSeat callback for seatIndex and userInfo, the original callback is deprecated. OnRequestCancelled callback for request, userInfo, the original callback is deprecated. OnRequestProcessed callback of request and userInfo, the original callback is deprecated. OnUserRoleChanged callback with userInfo as new parameter, the original callback is deprecated. OnRoomDismissed callback with roomId and TUIRoomDismissedReason, the original callback is deprecated.  <b>RoomKit</b> Android&iOS : Support setting watermarks in meetings; Optimise the logout logic, listen to the logout action during the meeting, exit the current meeting and give a prompt; Android : Solve the problem of not being able to auto-upload when changing to the house owner during the uploading process; Solve the memory leak when BottomSheetDialog is still displayed when Activity finish; Remove the 3rd party library of support-v4; iOS :	2024-04-29



Solve the problem of switching between small and large windows during a two-person meeting, and the small window is black screened; Solve the problem that there is no watermark in the small window of a two-person meeting;

Electron :

Fix the problem that the default device in the camera list is not updated when unplugging the external camera under Electron.

Fix the problem that the member action panel is not far enough from the top and the display is not complete; Fix the problem that the member action panel is not far enough from the top and the display is not complete.

Fix the problem that Notification component shows the events of other members; Fix the problem that Notification component does not update the list of applications after getting the identities of hosts and administrators.

Repair the problem of not updating the list of applications after getting the identity of moderator and administrator; Repair the problem of not updating the list of applications after getting the identity of moderator and administrator.

Fix the problem of inaccurate unread count of chat messages.

Repair the problem of disable status error when clicking the microphone and camera buttons several times when transferring to host or administrator status.

Web :

Repair the problem of incomplete display of membership time in H5 mobile terminal.

Repair the problem of line breaks in the prompt message of H5 mobile;

Fix the problem that the member management list cannot scroll.

Fix the problem that the member management list cannot be scrolled in H5 mobile; Fix the problem that the member operation panel is not far enough from the top.

Fix the problem of incomplete display of member operation panel due to insufficient distance from the top of panel; Fix the problem of incomplete display of member operation panel due to insufficient distance from the top of panel.

Repair the problem of white screen when loading demo page in onePlus browser on H5 mobile; Repair the problem of white screen when loading demo page in onePlus browser on H5 mobile.

Repair the problem of notification component displaying events when it receives events handled by other members; Repair the problem of notification component not displaying events handled by other members.

Repair the problem of not updating the application list after getting the host and administrator identities; Repair the problem of not updating the application list after getting the host and administrator identities.

<p>Fix the problem of inaccurate unread count of chat messages.</p> <p>Fix the problem that the MessageBox component cannot be closed when the button is clicked after entering a room.</p> <p>Repair the problem of disable status error when clicking the microphone and camera buttons several times when transferring to host or administrator status.</p> <p>Mini Program :</p> <p>Repair the problem of incomplete display of the time of joining the applet.</p> <p>Repair the problem of line breaks in the upstage prompt message of applets.</p> <p>Repair the problem that the member management list of apps cannot be scrolled.</p> <p>Repair the problem that after creating a room in an app, clicking the home button in the upper left corner to go back to the homepage, the new room is created abnormally again.</p> <p>Fix the problem that the icon cannot be switched after transferring the host to the app; Fix the problem that the icon cannot be switched after transferring the host to the app.</p> <p>Repair the problem of not updating the application list after getting the host and administrator identities; Repair the problem of not updating the application list after getting the host and administrator identities.</p> <p>Fix the problem of inaccurate unread count of chat messages.</p> <p>Repair the problem of notification component displaying events when it receives events handled by other members; Repair the problem of notification component displaying events when it receives events handled by other members.</p> <p>Repair the problem of disable status error when clicking the microphone and camera buttons several times when transferring to host or administrator status.</p>	
--	--

**Note :**

The Multiplayer Audio/Video Room SDK (TUIRoomKit) is available from 8 April 2023 onwards. If you have accessed the product during the release of the free version, we have extended the time limit for you to 8 May 2023, so that you can upgrade to the official version of the product, and during this period, you can still use and experience the product. For more information, see: Product Bulletin.