Tencent Real-Time Communication

Voice Chat Room (with UI)

Product Documentation

Contents

Voice Chat Room (with UI) Integrating TUIVoiceRoom (Android)

Last updated：2024-01-18 11:18:54

Overview

`TUIVoiceRoom` is an open-source audio/video UI component. After integrating it into your project, you can make your application support the group audio chat scenario simply by writing a few lines of code. It also supports the iOS platform. Its basic features are as shown below:

Note

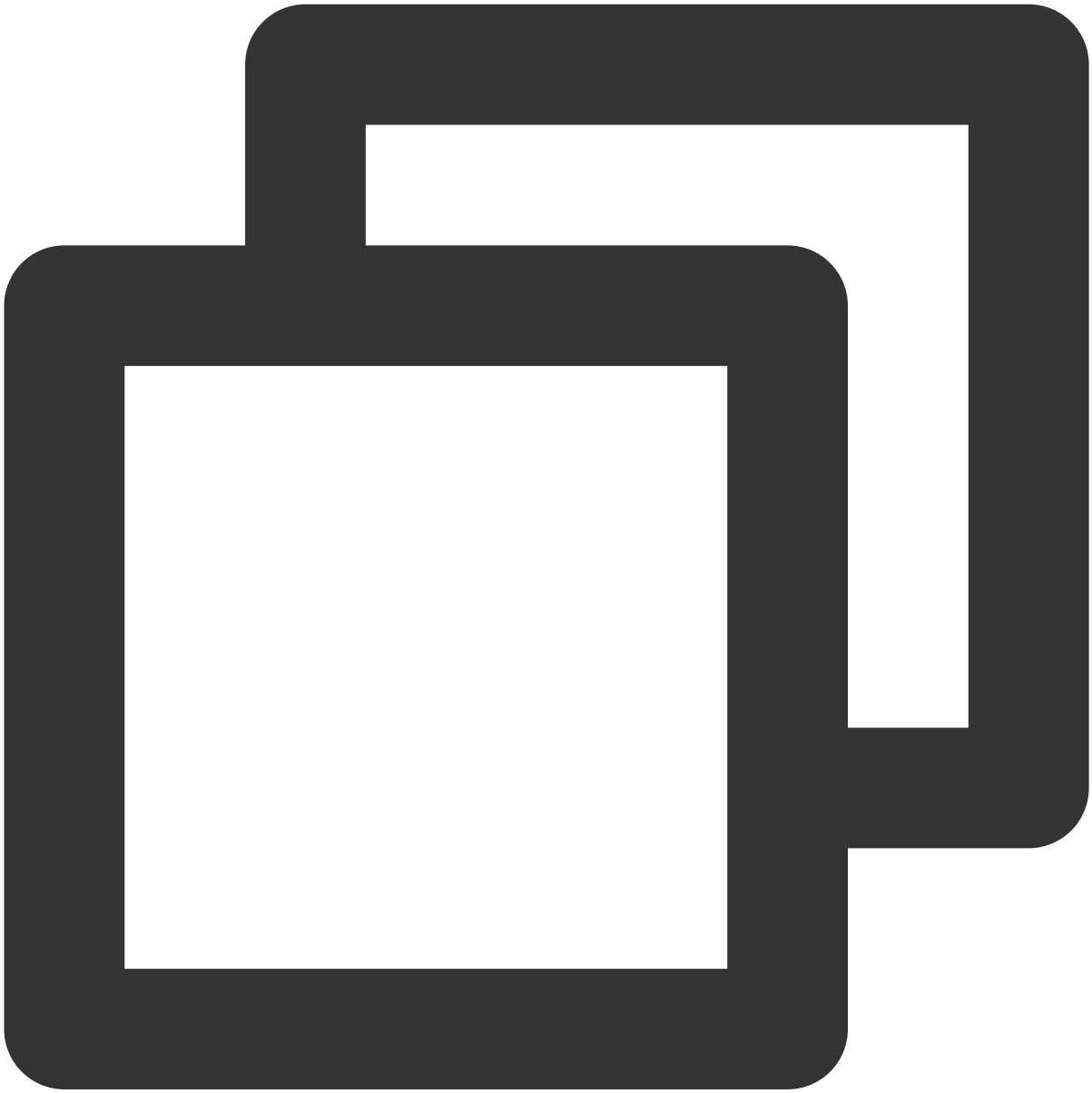
All TUIKit components are based on two PaaS services of Tencent Cloud, namely TRTC and Chat. When you activate TRTC, the Chat SDK trial edition (which supports up to 100 DAUs) will be activated automatically. For billing details of Chat, see Pricing.

Integration

Step 1. Download and import the `TUIVoiceRoom` component

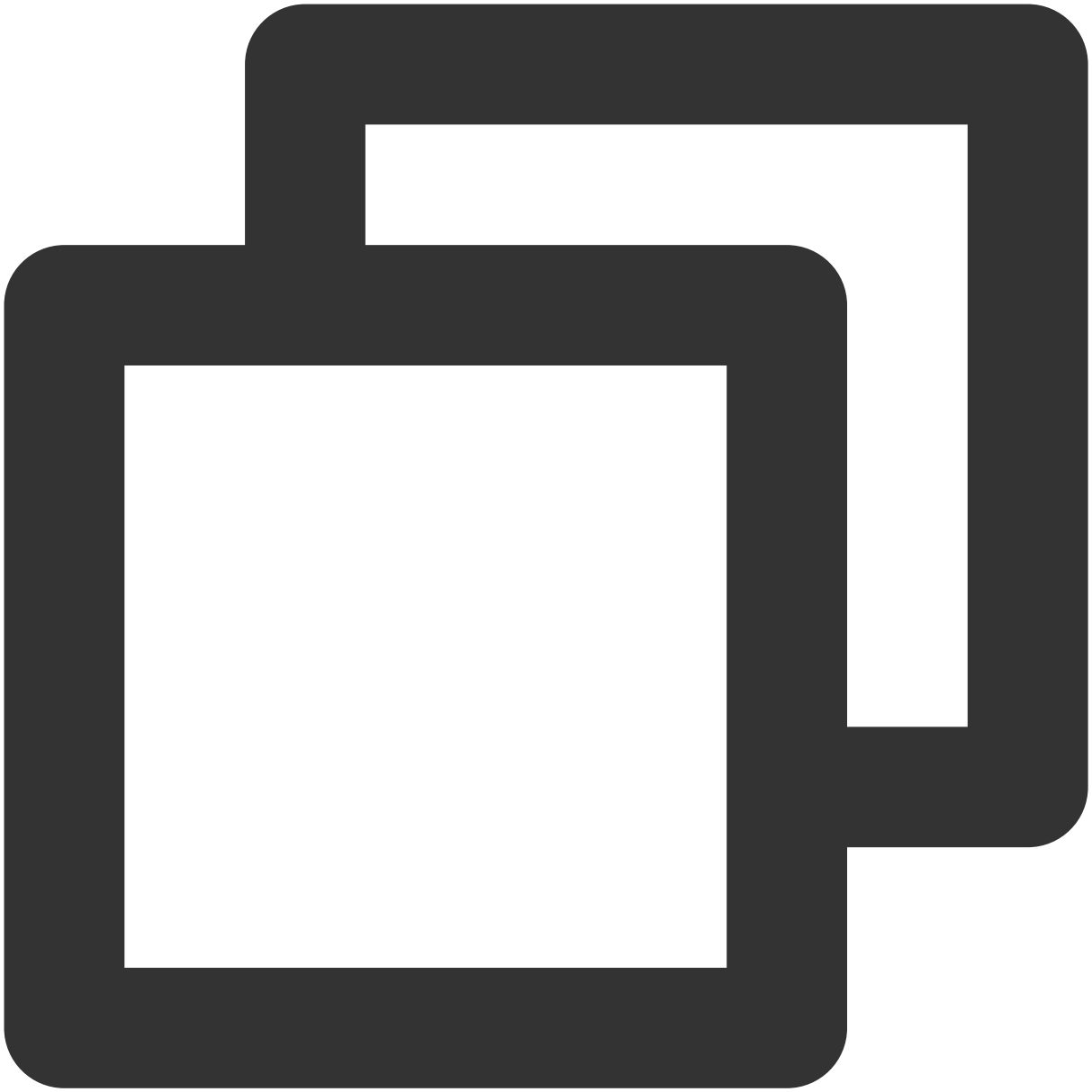
Go to GitHub, clone or download the code, copy the `Android/Source` directory to your project, and complete the following import operations:

Add the code below in `setting.gradle` :



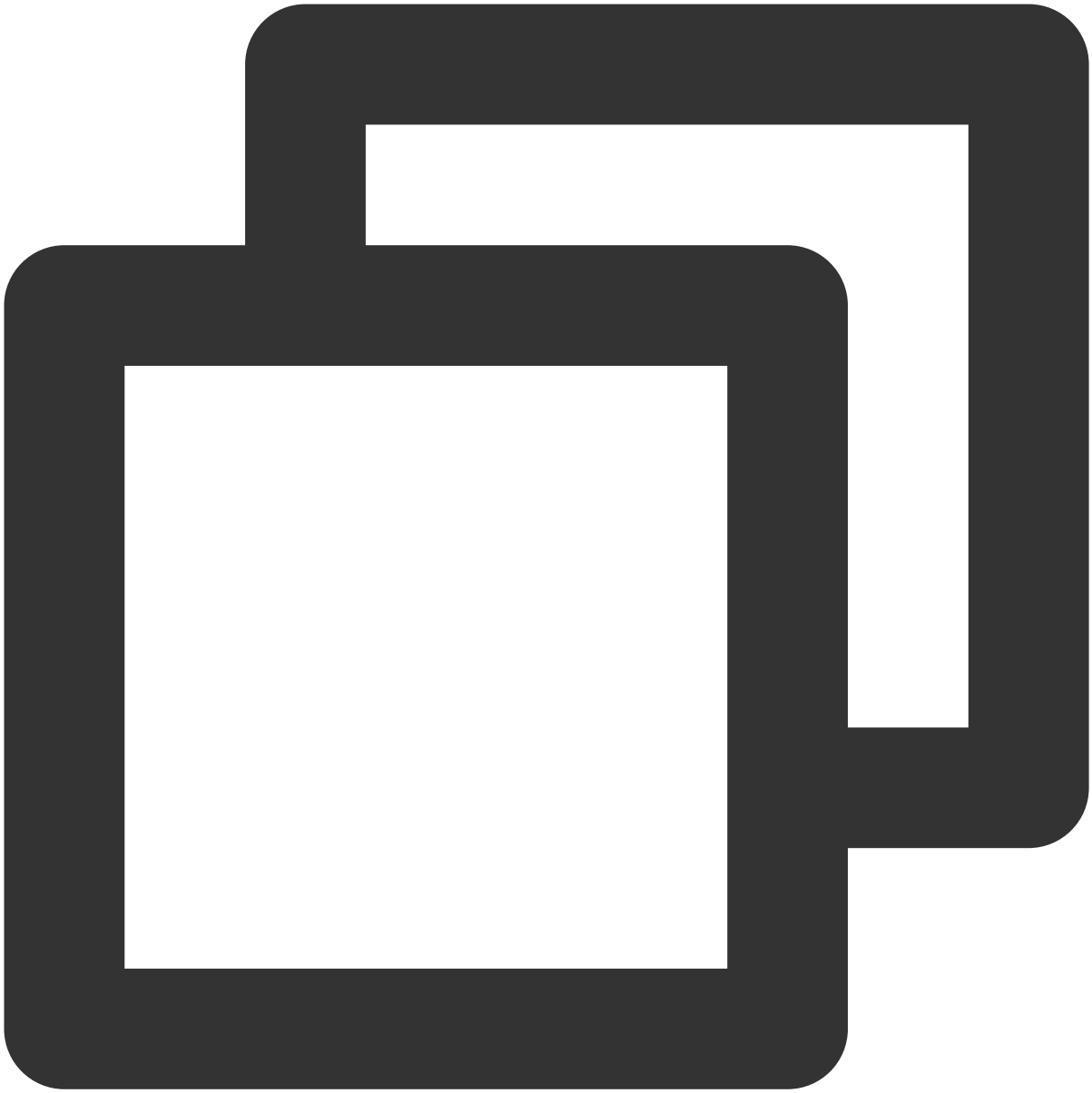
```
include ':Source'
```

Add dependencies on `Source` to the `build.gradle` file in `app` :



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

Add the TRTC SDK (`liteavSdk`) and Chat SDK (`imSdk`) dependencies in `build.gradle` in the root directory:



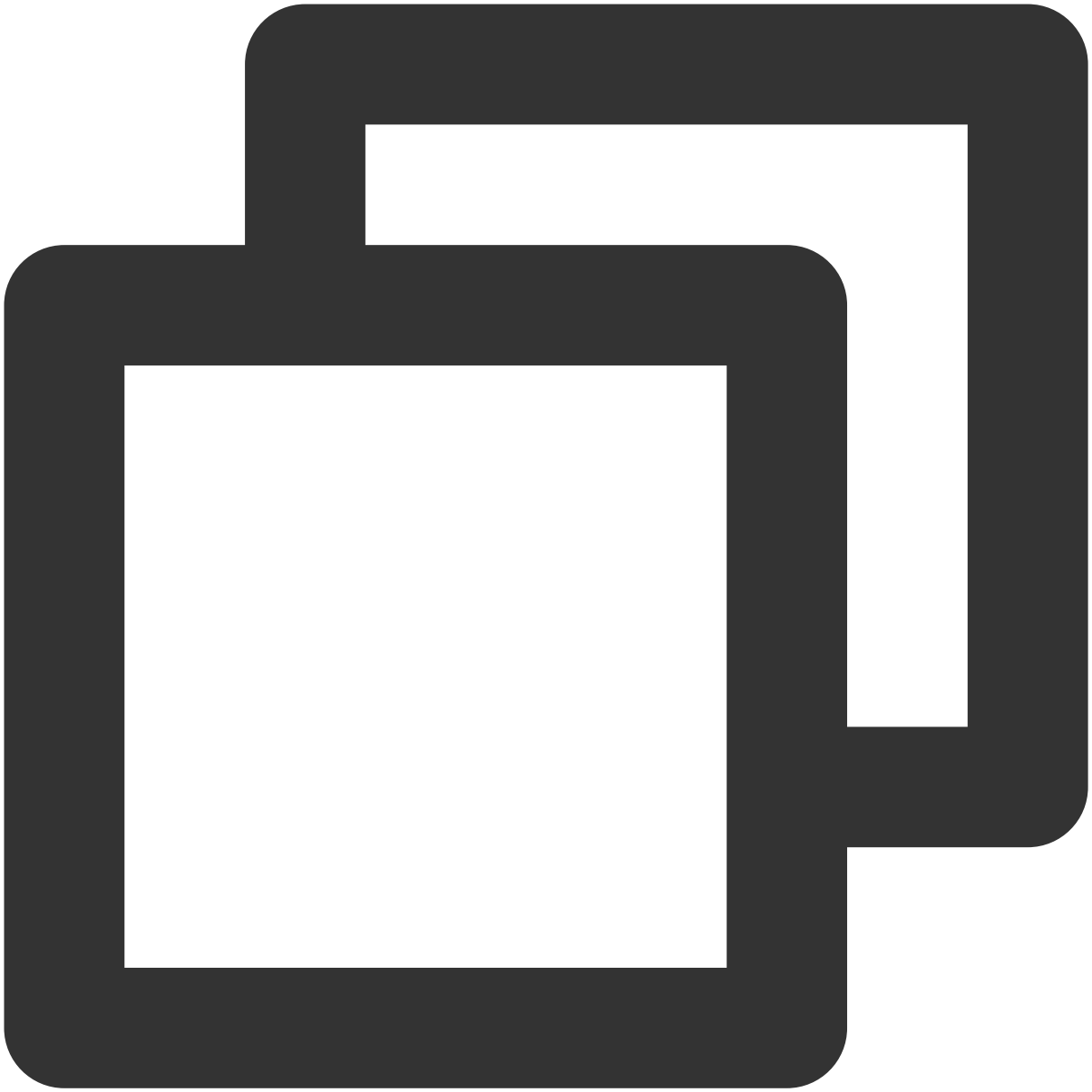
```
ext {
  liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
  imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

Step 2. Configure permission requests and obfuscation rules

Configure permission requests for your app in `AndroidManifest.xml` . The SDKs need the following permissions (on Android 6.0 and later, the mic access must be requested at runtime):

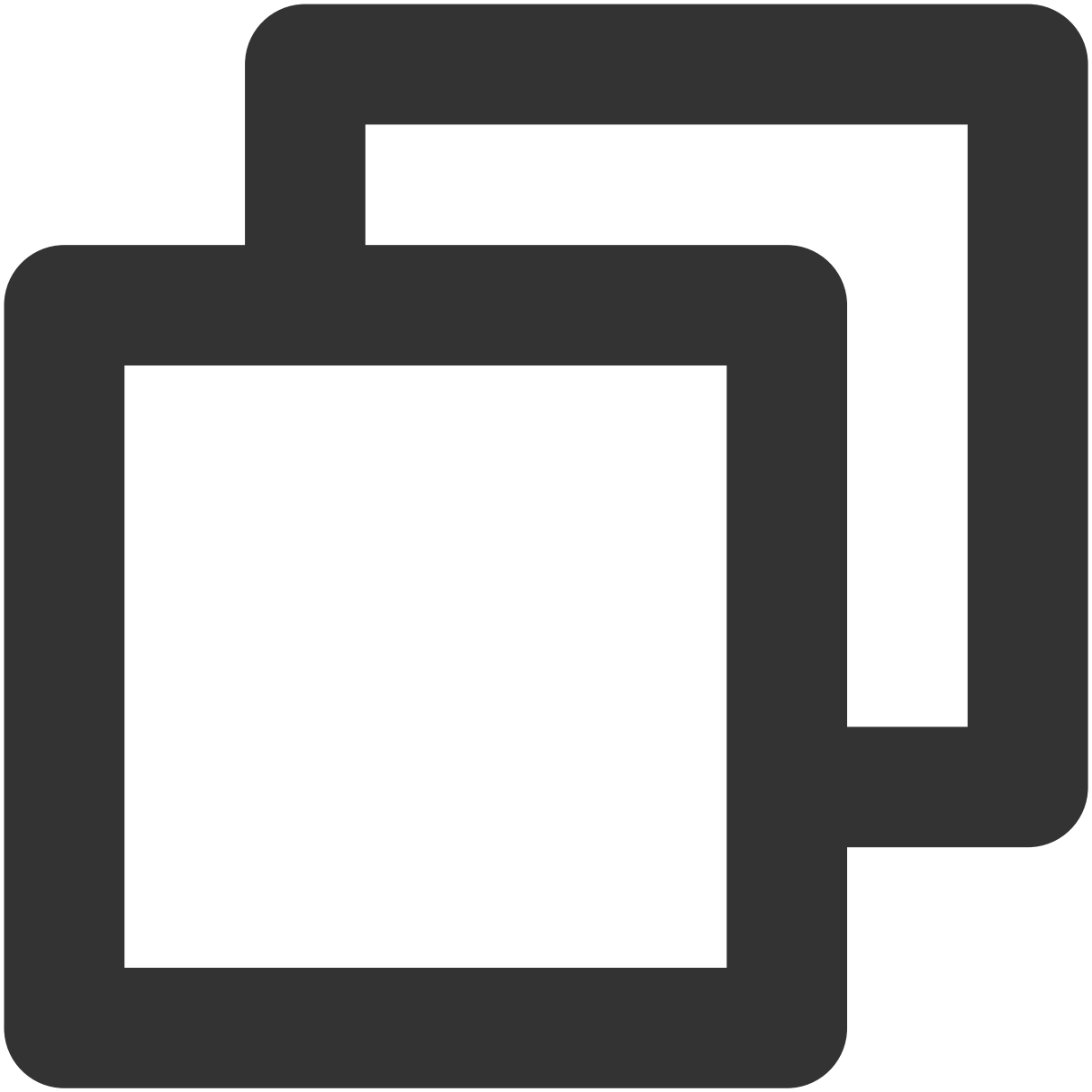
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

In the `proguard-rules.pro` file, add the SDK classes to the "do not obfuscate" list.



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

Step 3. Initialize and log in to the component

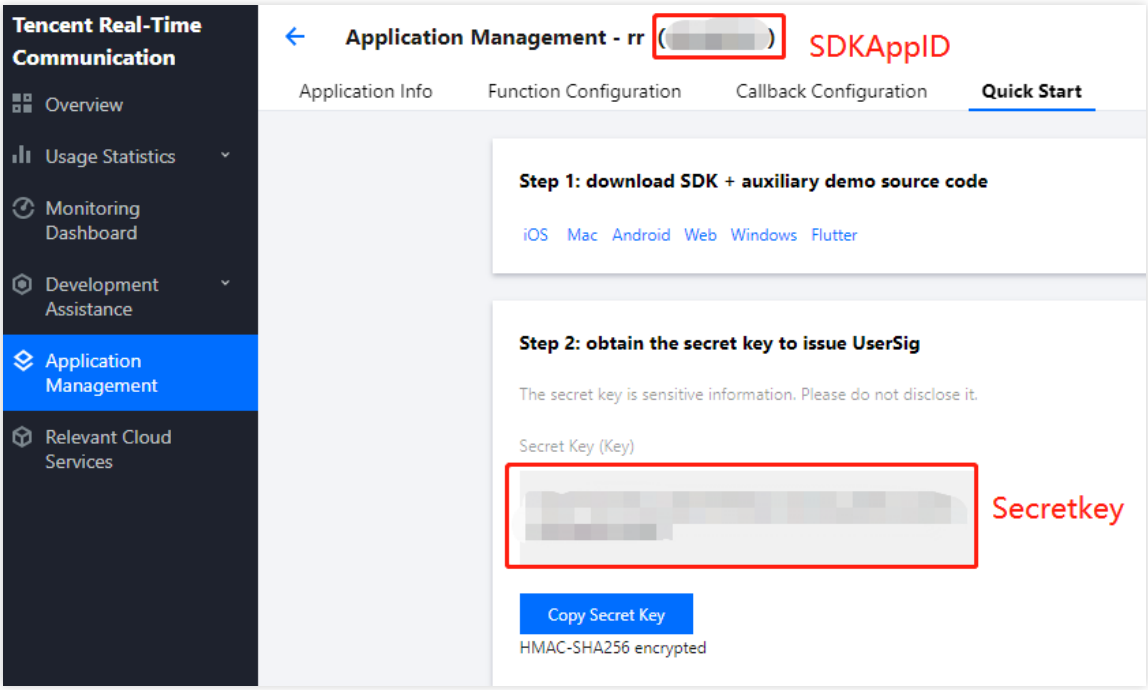


```
// 1. Initialize
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
);
      // 2. Log in
mTRTCVoiceRoom.login(SDKAppID, userId, userSig, new TRTCVoiceRoomCallback.Actio
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
        // Logged in
        }
```

```
            }
        });
```

Parameter description:

SDKAppID: The **TRTC application ID**. If you haven't activated TRTC, log in to the TRTC console, create a TRTC application, click **Application Info**, and select the **Quick Start** tab to view its `SDKAppID` .



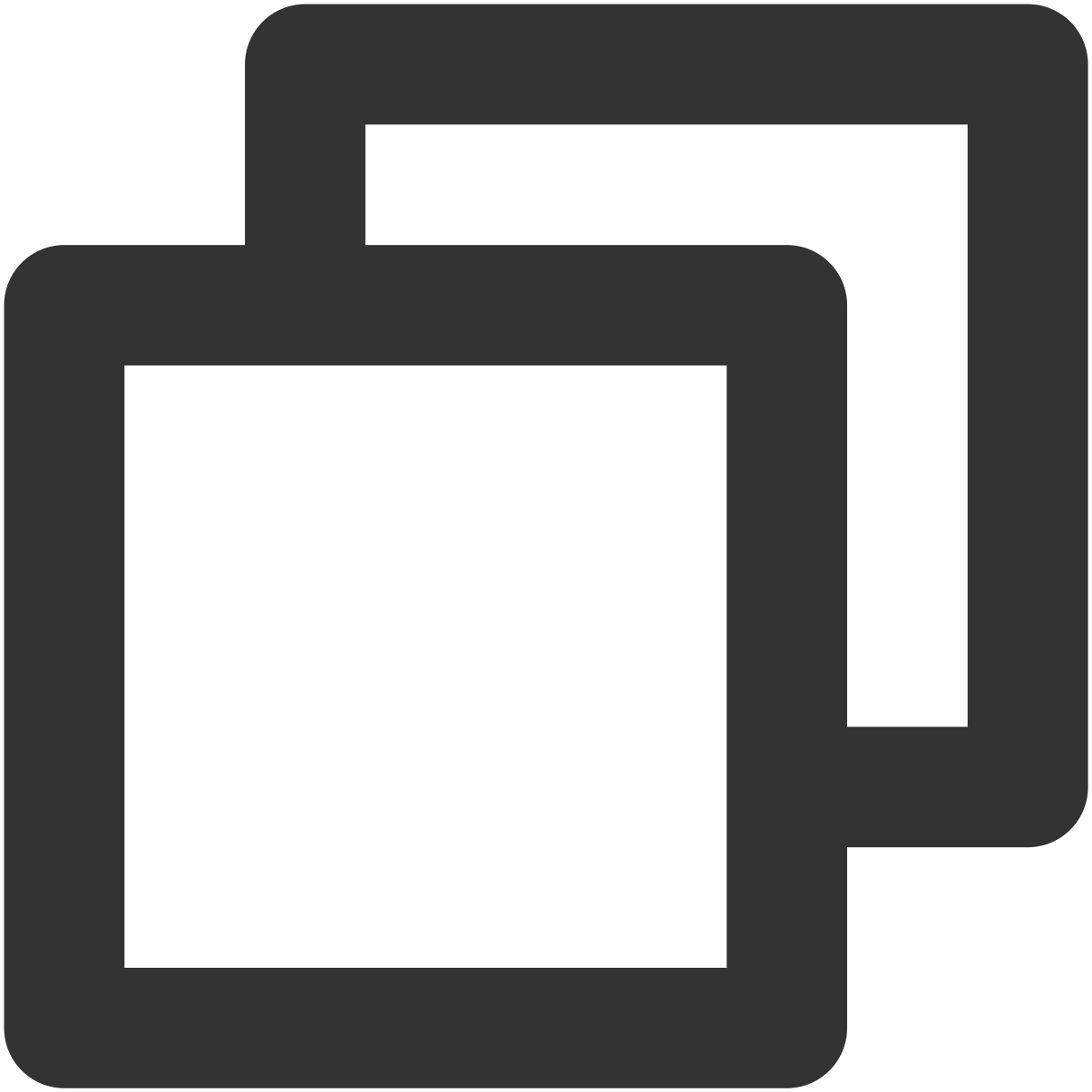
Secretkey: The **TRTC application key**. Each secret key corresponds to a `SDKAppID` . You can view your application's secret key on the Application Management page of the TRTC console.

userId: The ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_). We recommend that you keep it consistent with your user account system.

userSig: The security protection signature calculated based on `SDKAppID` , `userId` , and `Secretkey` . You can click here to directly generate a debugging `userSig` online. For more information, see UserSig.

Step 4. Implement the audio chat room

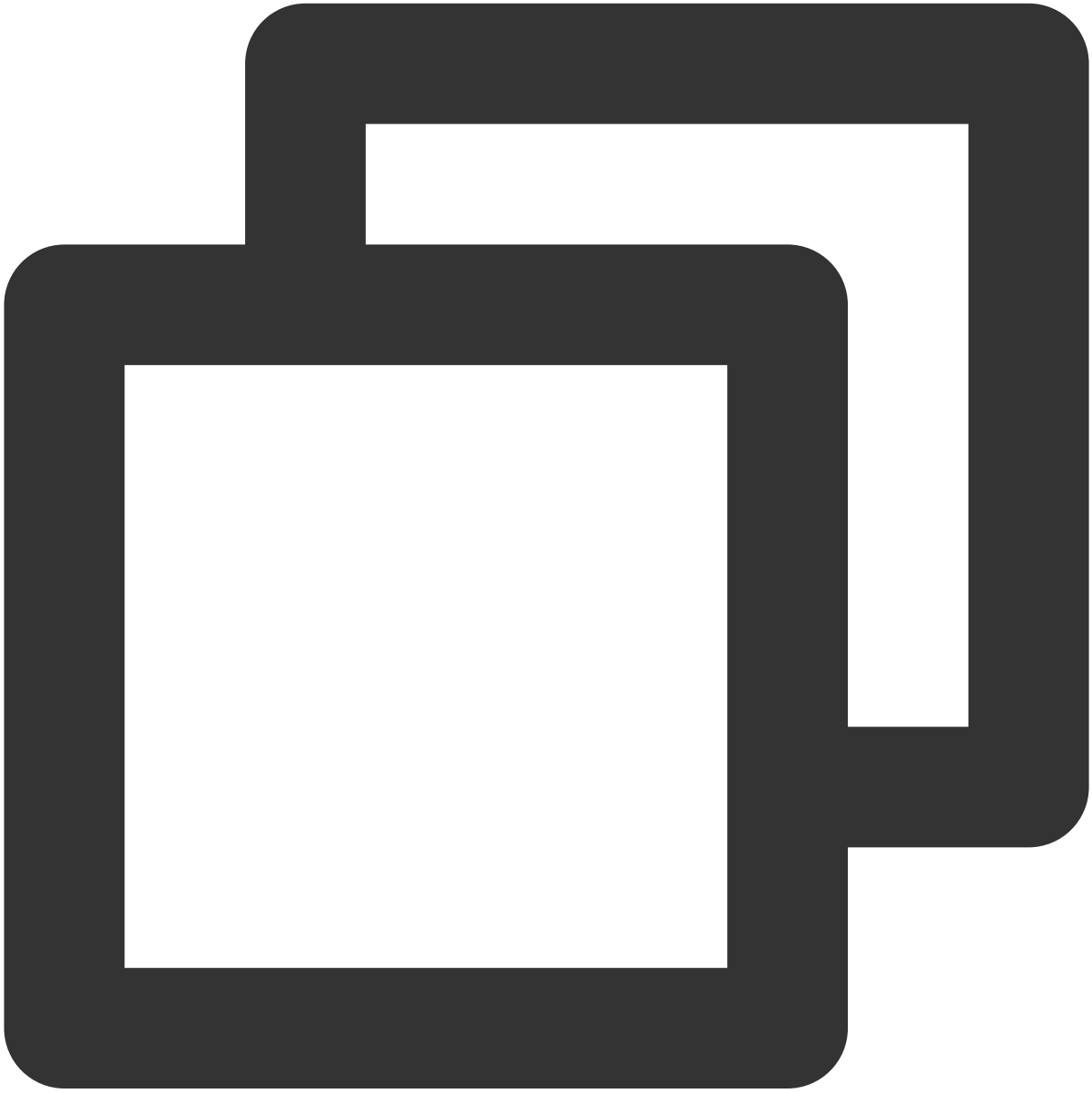
1. **The room owner creates an audio chat room through** TRTCVoiceRoom#createRoom.



```
// 1. The room owner calls an API to create a room
int roomId = 12345; // Room ID
final TRTCVoiceRoomDef.RoomParam roomParam = new TRTCVoiceRoomDef.RoomParam();
roomParam.roomName = "Room name";
roomParam.needRequest = false; // Whether the room owner’s permission is required f
roomParam.seatCount = 7; // Number of room seats. In this example, the number is 7.
roomParam.coverUrl = "URL of room cover image";
mTRTCVoiceRoom.createRoom(roomId, roomParam, new TRTCVoiceRoomCallback.ActionCallba
 @Override
 public void onCallback(int code, String msg) {
     if (code == 0) {
```

```
                // Room created successfully
        }
    }
});
```

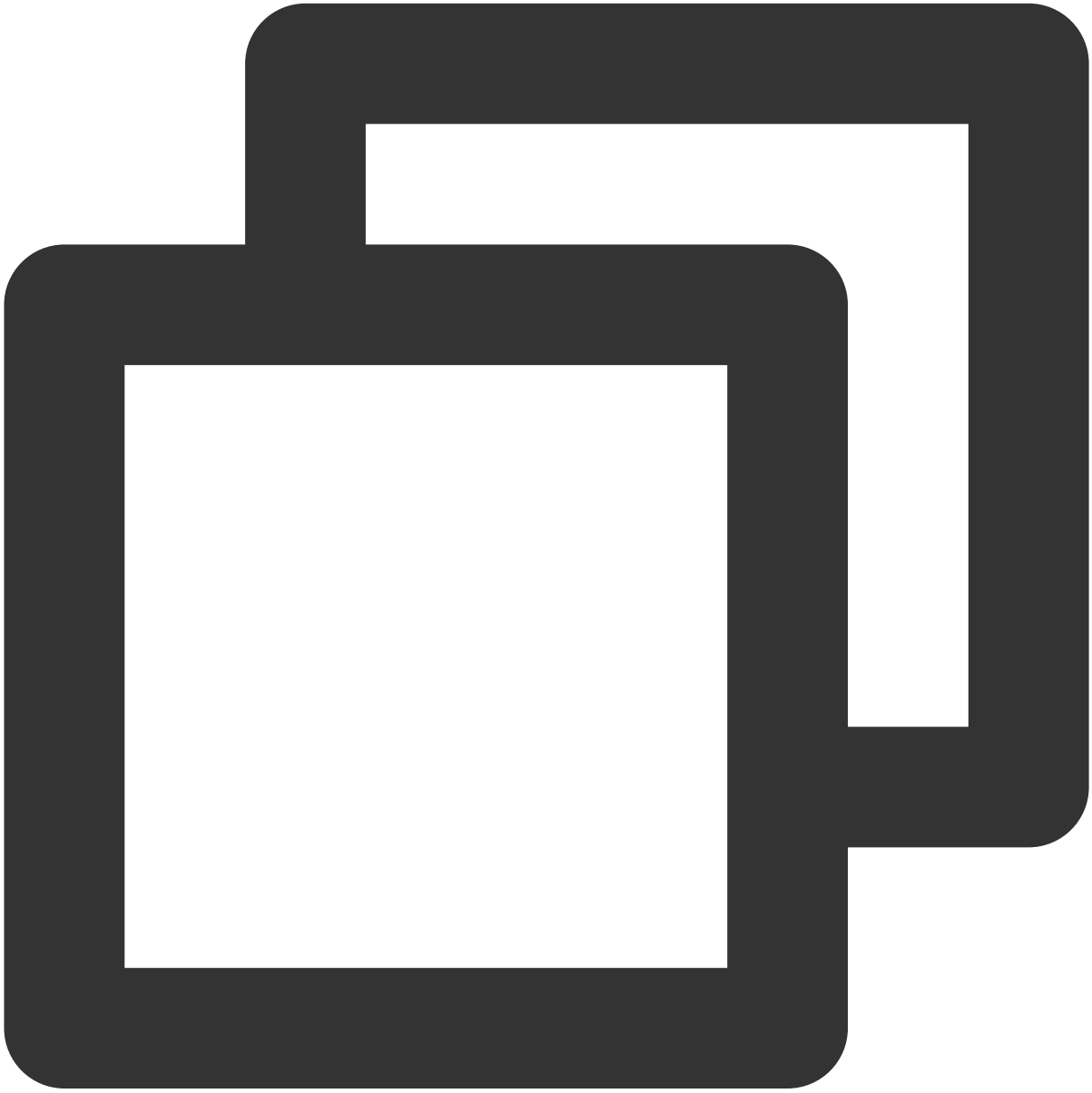
2. **A listener enters the audio chat room through** TRTCVoiceRoom#enterRoom.



```
// 1. A listener calls an API to enter the room
mTRTCVoiceRoom.enterRoom(roomId, new TRTCVoiceRoomCallback.ActionCallback() {
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
```

```
            // Entered room successfully
            }
        }
});
```

3. **A listener mics on through** TRTCVoiceRoom#enterSeat.

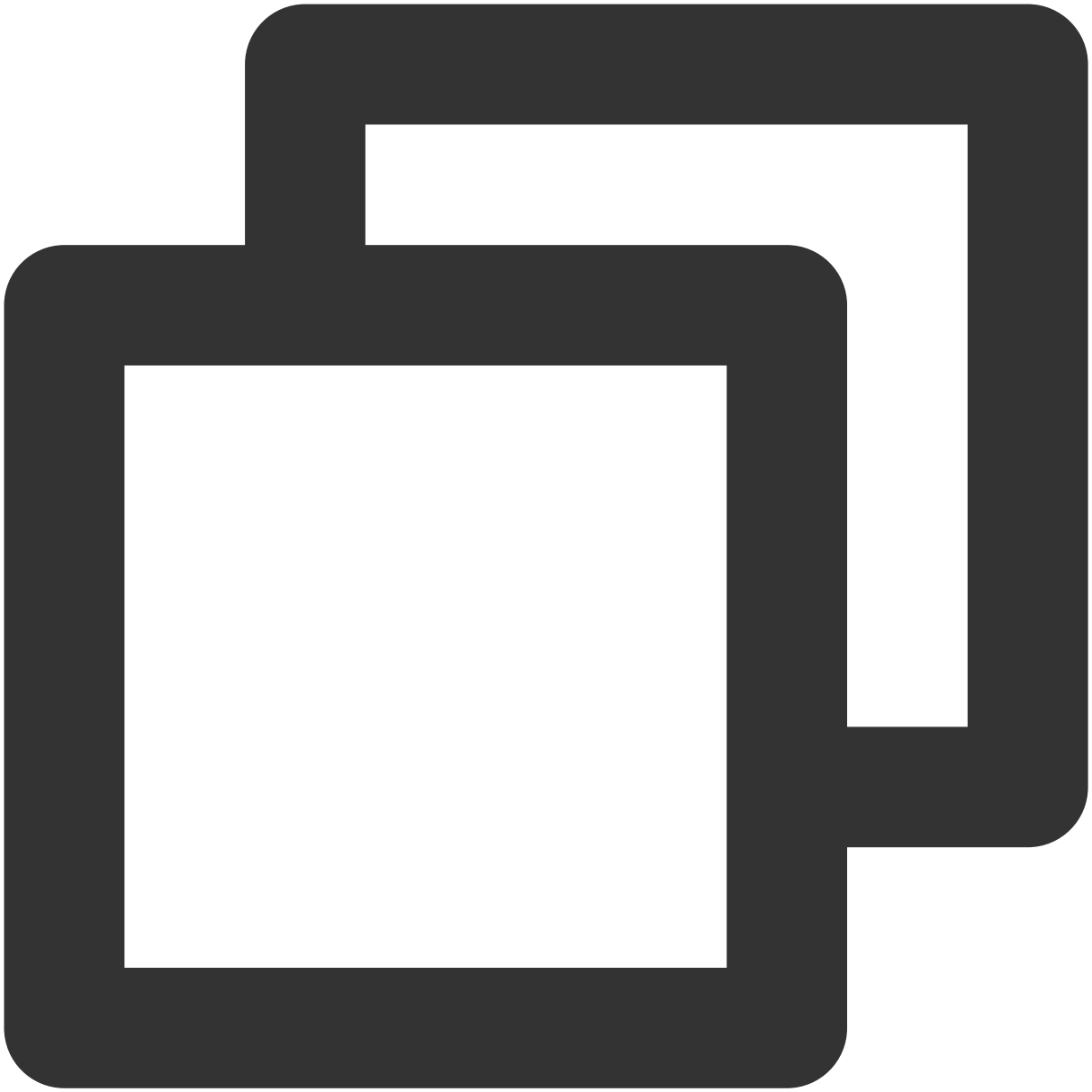


```
// 1. A listener calls an API to mic on
int seatIndex = 2; // Seat index
mTRTCVoiceRoom.enterSeat(seatIndex, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
```

```
        if (code == 0) {
        // Operation succeeded
        }
    }
});

// 2. The `onSeatListChange` callback is received, and the seat list is refreshed
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

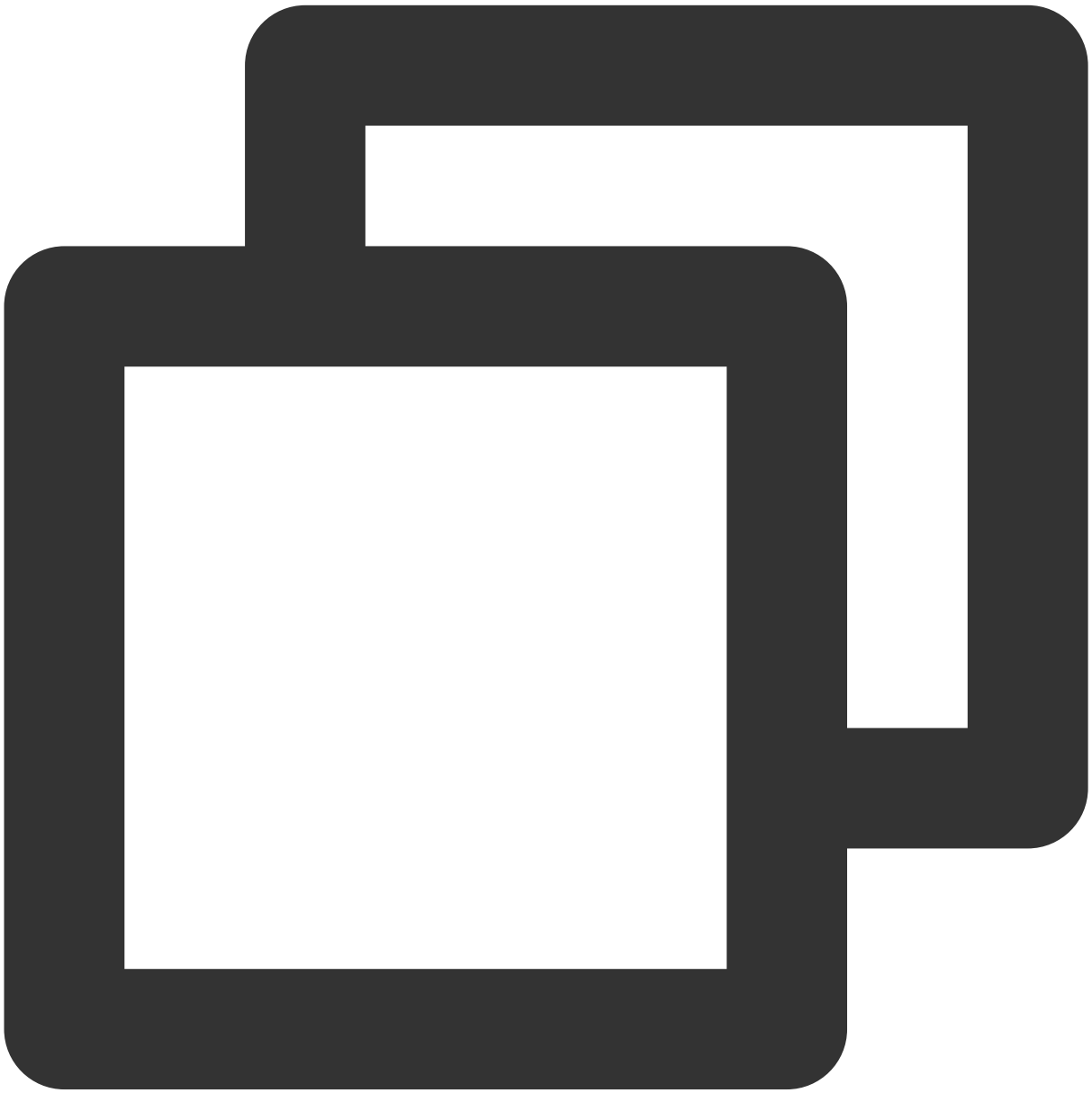
4. **The room owner makes a listener a speaker through** TRTCVoiceRoom#pickSeat.



```
// 1. The room owner makes a listener a speaker
int seatIndex = 2; // Seat index
String userId = "123"; // The ID of the user who will become a speaker
mTRTCVoiceRoom.pickSeat(1, userId, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
     if (code == 0) {
             // Operation succeeded
     }
 }
});
```

```
// 2. The `onSeatListChange` callback is received, and the seat list is refreshed
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

5. **A listener requests to speak through** TRTCVoiceRoom#sendInvitation.



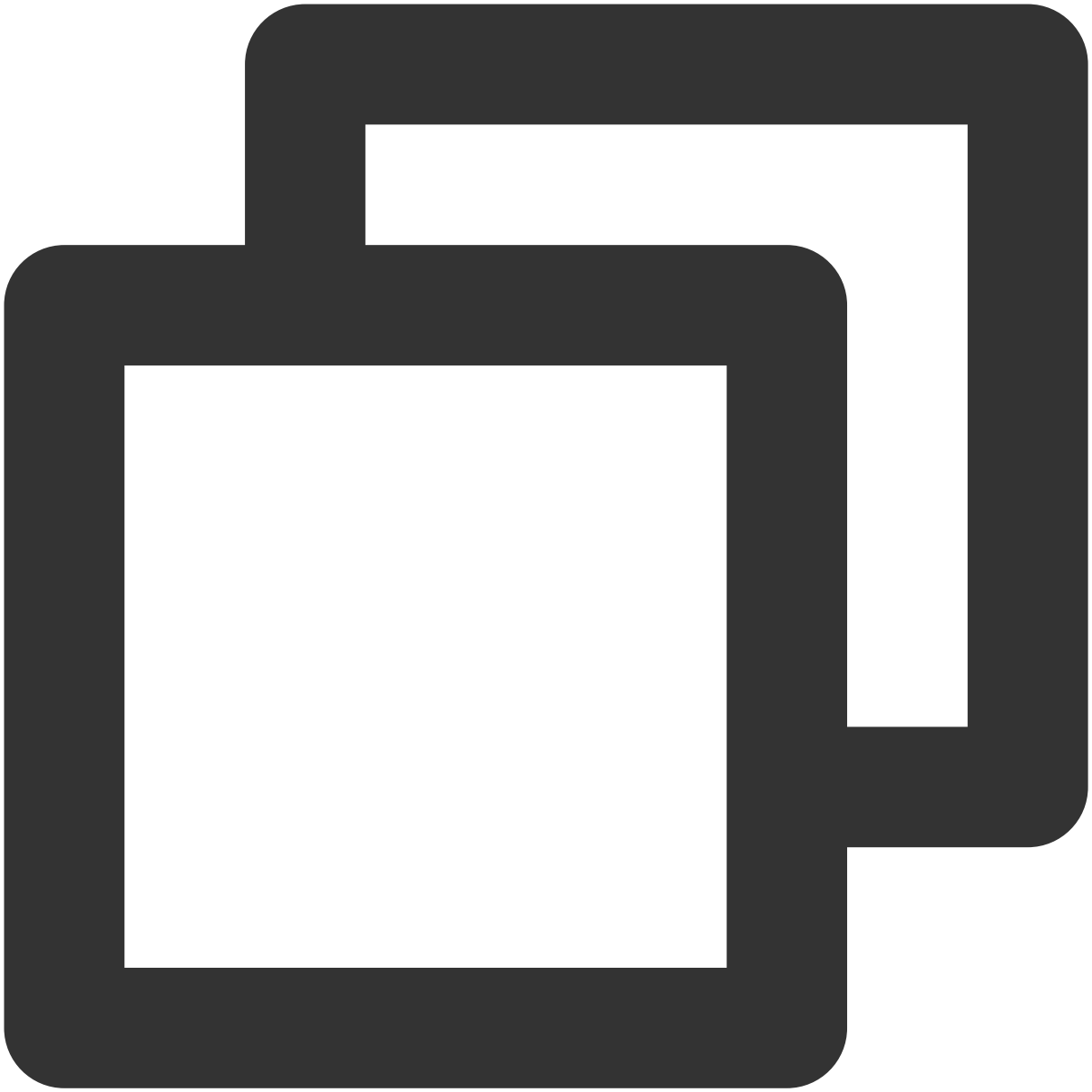
```
// Listener
// 1. A listener calls an API to request to speak
String seatIndex = "1"; // Seat index
String userId = "123"; // User ID
```

```
String inviteId = mTRTCVoiceRoom.sendInvitation("takeSeat", userId, seatIndex, null

// 2. Place the user in the seat after the invitation is accepted
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
     mTRTCVoiceRoom.enterSeat(index, null);
 }
}

// Room owner
// 1. The room owner receives the request
 @Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, fin
 if (cmd.equals("takeSeat")) {
     // 2. The room owner accepts the request
      mTRTCVoiceRoom.acceptInvitation(id, null);
 }
}
```

6. **The room owner invites a listener to speak through** TRTCVoiceRoom#sendInvitation.



```
// Room owner
// 1. The room owner calls an API to invite a listener to speak
String seatIndex = "1"; // Seat index
String userId = "123"; // User ID
String inviteId = mTRTCVoiceRoom.sendInvitation("pickSeat", userId, seatIndex, null

// 2. Place the user in the seat after the invitation is accepted
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
     mTRTCVoiceRoom.pickSeat(index, null);
```

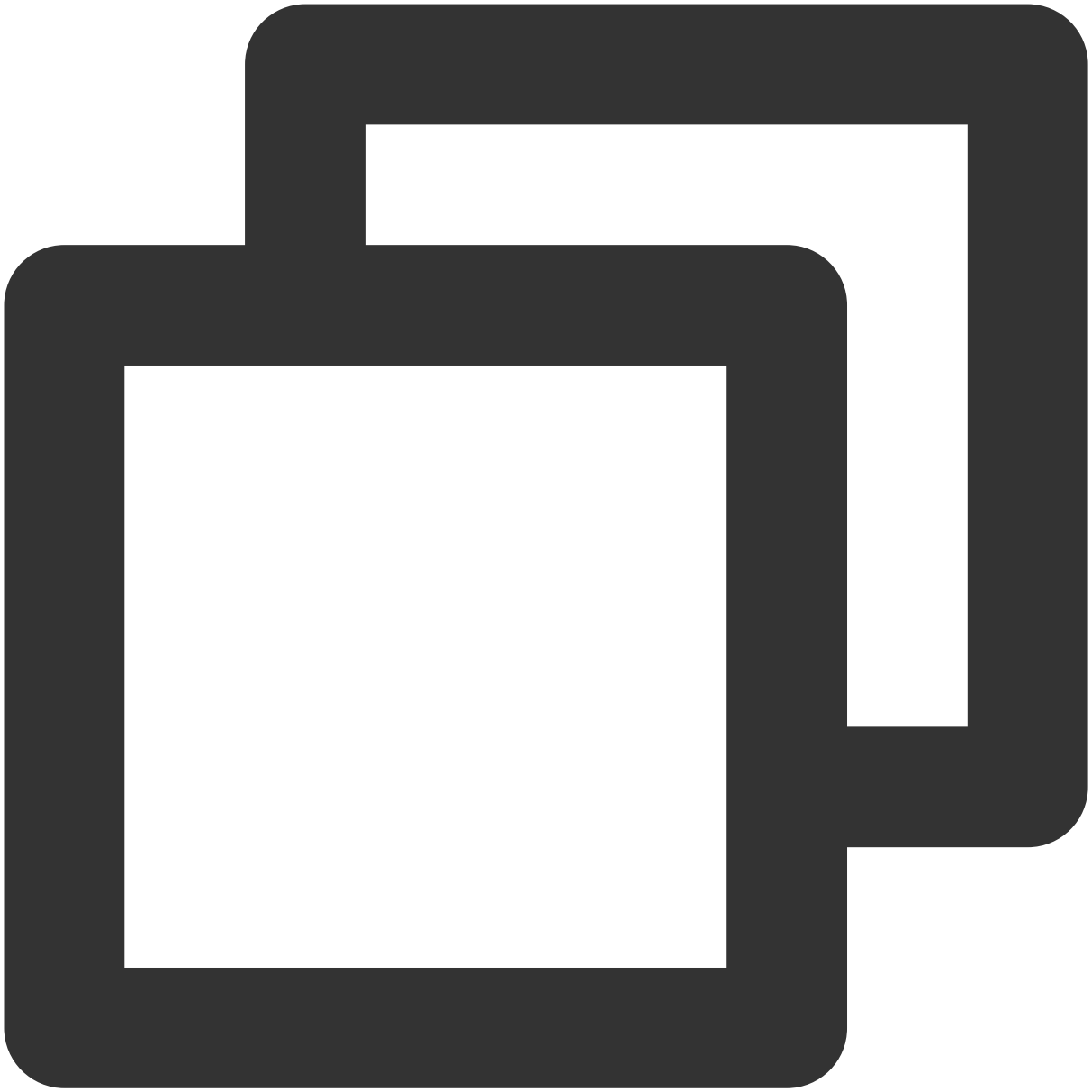
```
  }
 }

 // Listener
 // 1. The listener receives the invitation
  @Override
 public void onReceiveNewInvitation(final String id, String inviter, String cmd, fin
  if (cmd.equals("pickSeat")) {
      // 2. The listener accepts the invitation
       mTRTCVoiceRoom.acceptInvitation(id, null);
  }
 }
```

7. **Implement text chat through** TRTCVoiceRoom#sendRoomTextMsg.

```
// Sender: Sends a text chat message
mTRTCVoiceRoom.sendRoomTextMsg("Hello World!", null);
// Receiver: Listens for text chat messages
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
  @Override
  public void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo)
   Log.d(TAG, "Received a message from" + userInfo.userName + ": " + message);
  }
});
```

8. **Implement on-screen commenting through** TRTCVoiceRoom#sendRoomCustomMsg.



```
// A sender can customize CMD to distinguish on-screen comments and likes.
// For example, use "CMD_DANMU" to indicate on-screen comments and "CMD_LIKE" to in
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// Receiver: Listens for custom messages
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
 @Override
 public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserI
     if ("CMD_DANMU".equals(cmd)) {
         // An on-screen comment is received
         Log.d(TAG, "Received an on-screen comment from" + userInfo.userName + ": "
```

```
        } else if ("CMD_LIKE".equals(cmd)) {
            // A like is received
            Log.d(TAG, userInfo.userName + "liked you.");
        }
    }
});
```

Suggestions and Feedback

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

Integrating TUIVoiceRoom (iOS)

Last updated：2023-09-25 10:51:43

Overview

`TUIVoiceRoom` is an open-source audio/video UI component. After integrating it into your project, you can make your application support the group audio chat scenario simply by writing a few lines of code. It also supports the Android platform. Its basic features are as shown below:

Note

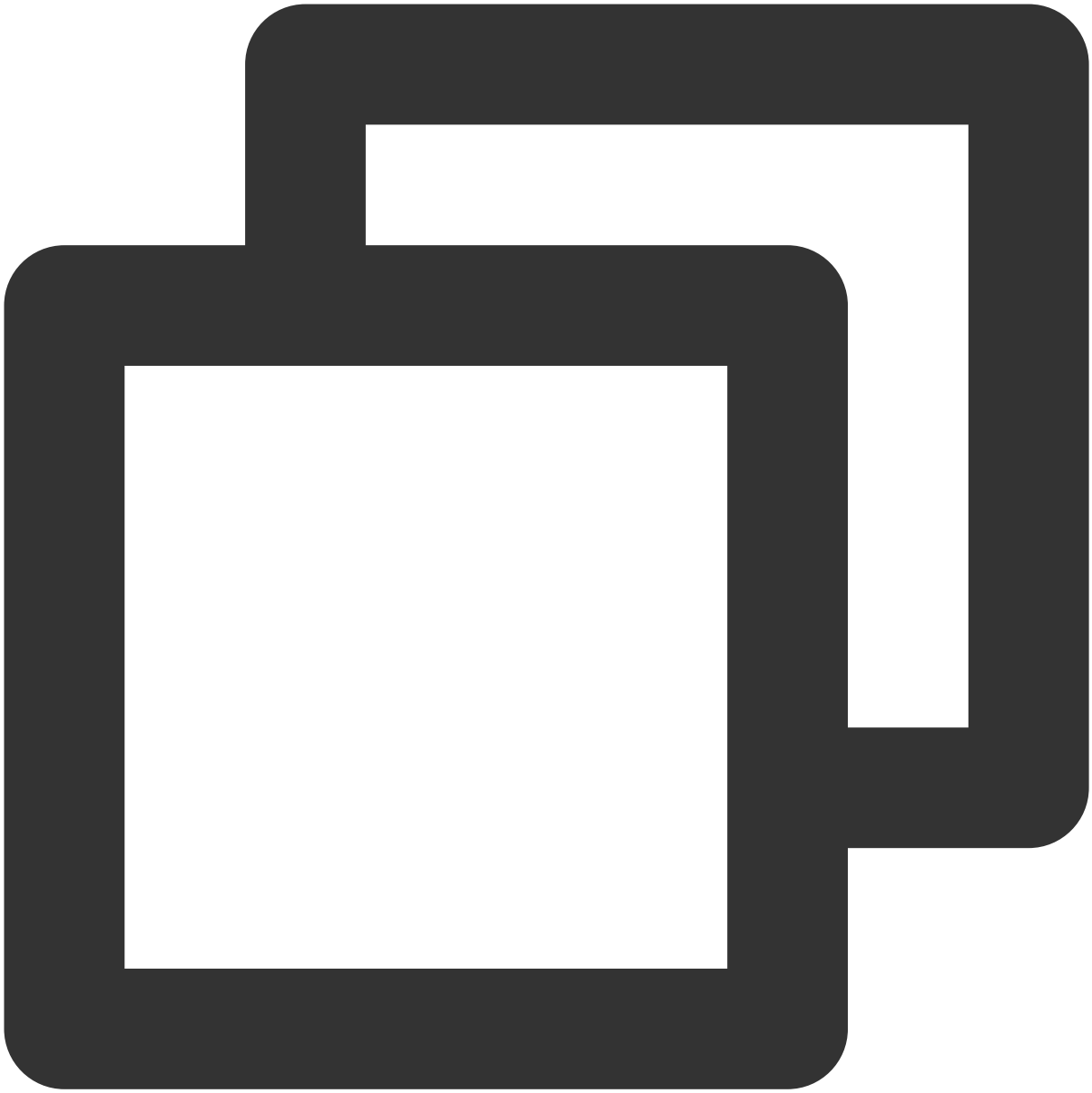
All TUIKit components are based on two PaaS services of Tencent Cloud, namely TRTC and Chat. When you activate TRTC, the Chat SDK trial edition (which supports up to 100 DAUs) will be activated automatically. For billing details of Chat, see Pricing.

Integration

Step 1. Download and import the `TUIVoiceRoom` component

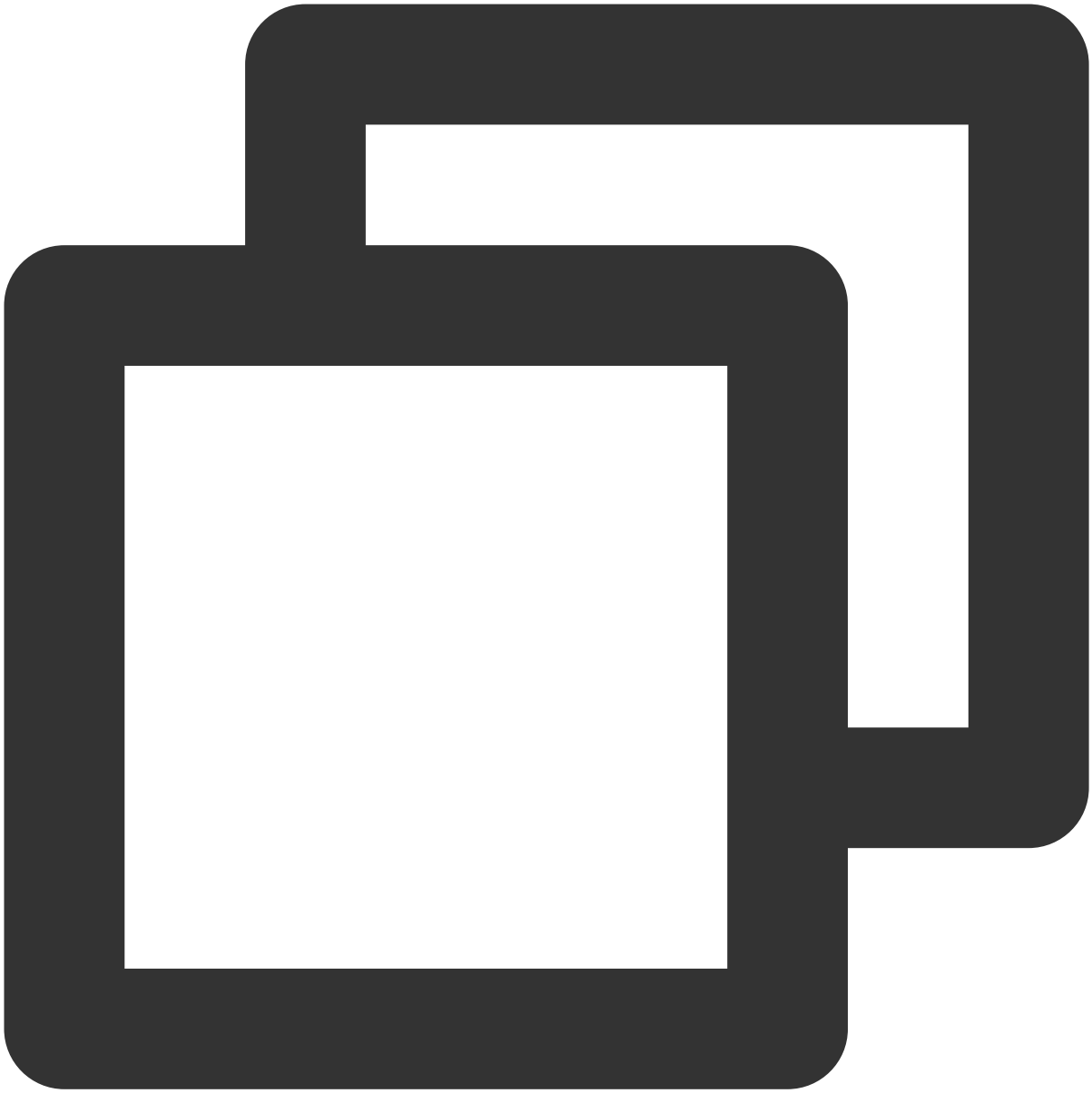
Create the `TUIVoiceRoom` folder at the same level as the `Podfile` in your Xcode project, copy TXAppBasic, Resources, Source, and TUIVoiceRoom.podspec files from the `iOS` directory in the GitHub repository to the folder, and complete the following import operations:

Open the project's `Podfile` and import `TUIVocieRoom.podspec` as follows:



```
# `path` is the path of `TXAppBasic.podspec` relative to the `Podfile`
pod 'TXAppBasic', :path => "TUIVoiceRoom/TXAppBasic/"
# `path` is the path of `TUIVoiceRoom.podspec` relative to the `Podfile`
pod 'TUIVoiceRoom', :path => "TUIVoiceRoom/", :subspecs => ["TRTC"]
```

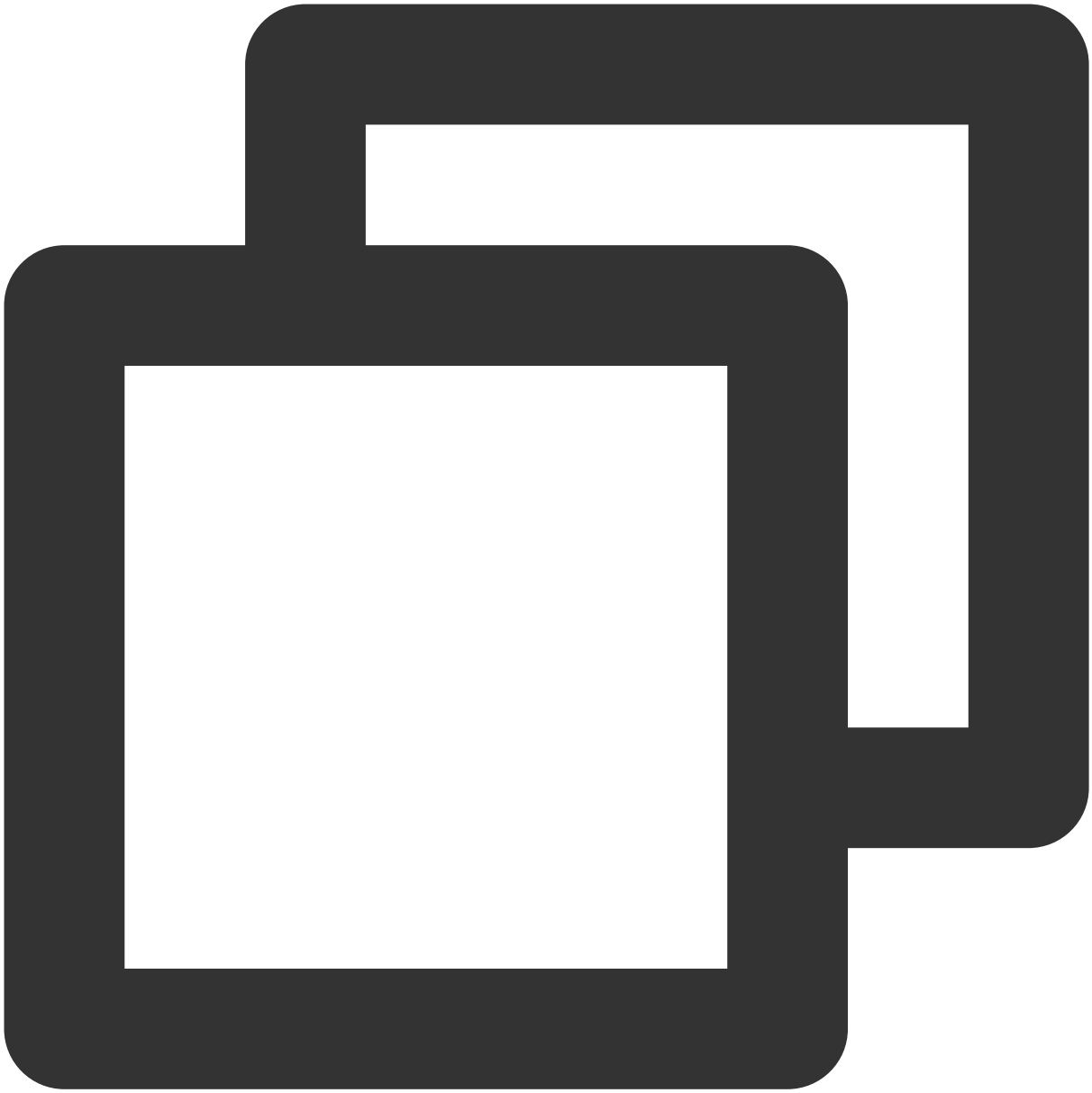
Open Terminal, enter the directory of `Podfile`, and run `pod install`.



```
pod install
```

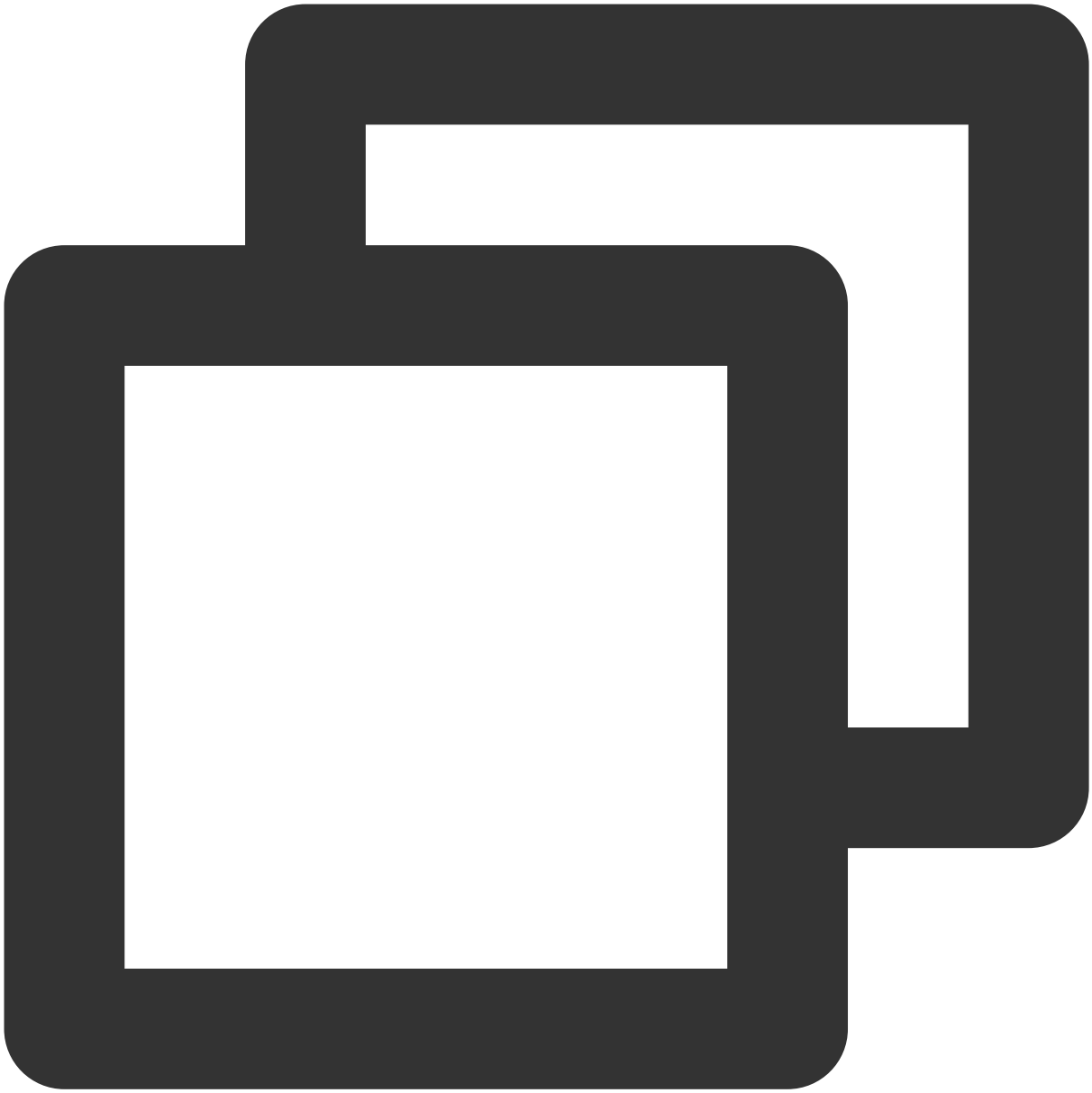
Step 2. Configure permission requests and obfuscation rules

In `info.plist`, add `Privacy > Microphone Usage Description` to request mic access.



```
<key>NSMicrophoneUsageDescription</key>
<string>`VoiceRoomApp` needs to access your mic to be able to shoot videos with aud
```

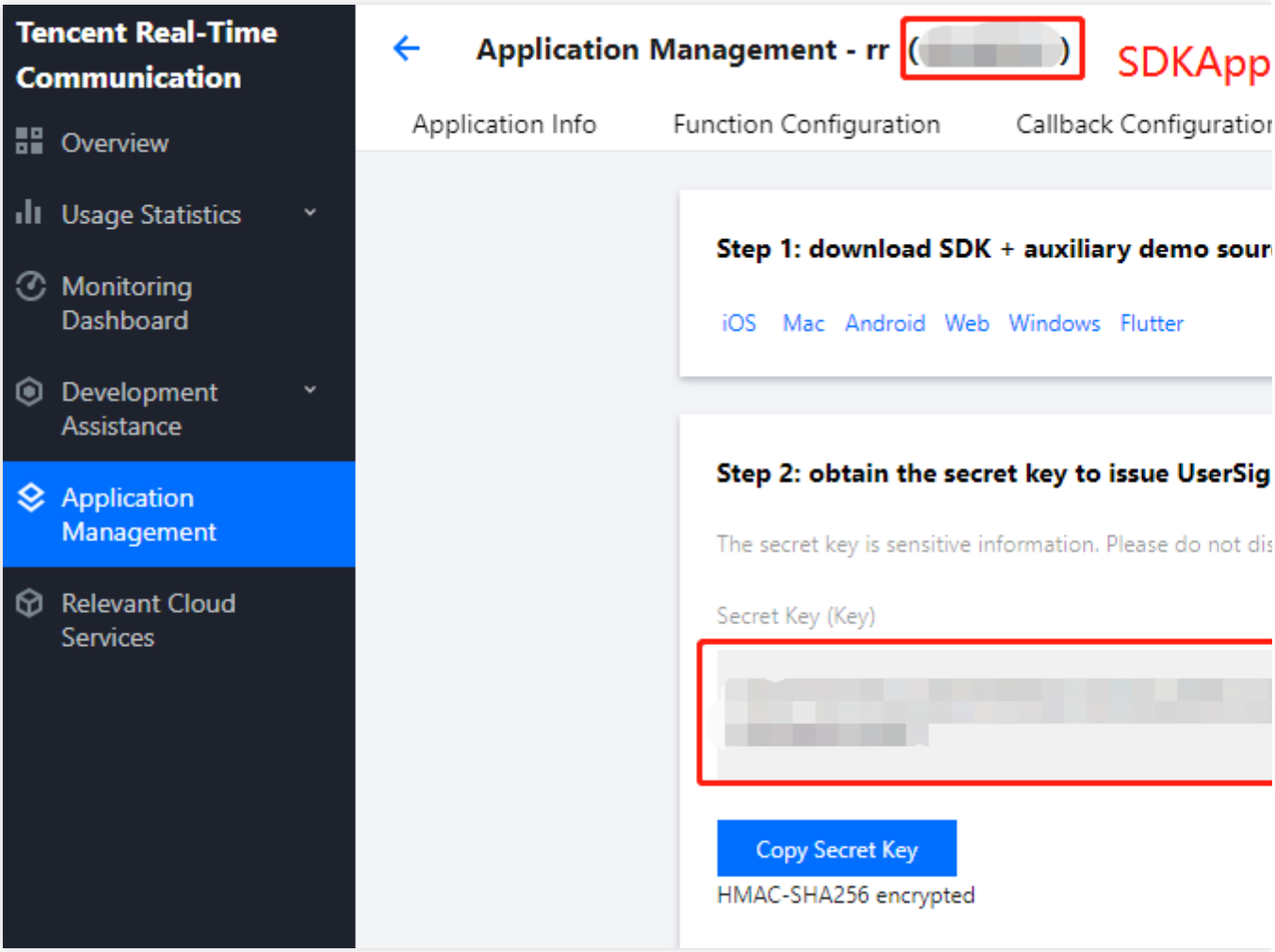
Step 3. Initialize and log in to the component



```
// Initialize TUIKit
let mTRTCVoiceRoom = TRTCVoiceRoom.shared()
// Log in
mTRTCVoiceRoom.login(sdkAppID: SDKAppID, userId: userId, userSig: userSig) { code,
    if code == 0 {
        // Logged in
    }
}
```

Parameter description:

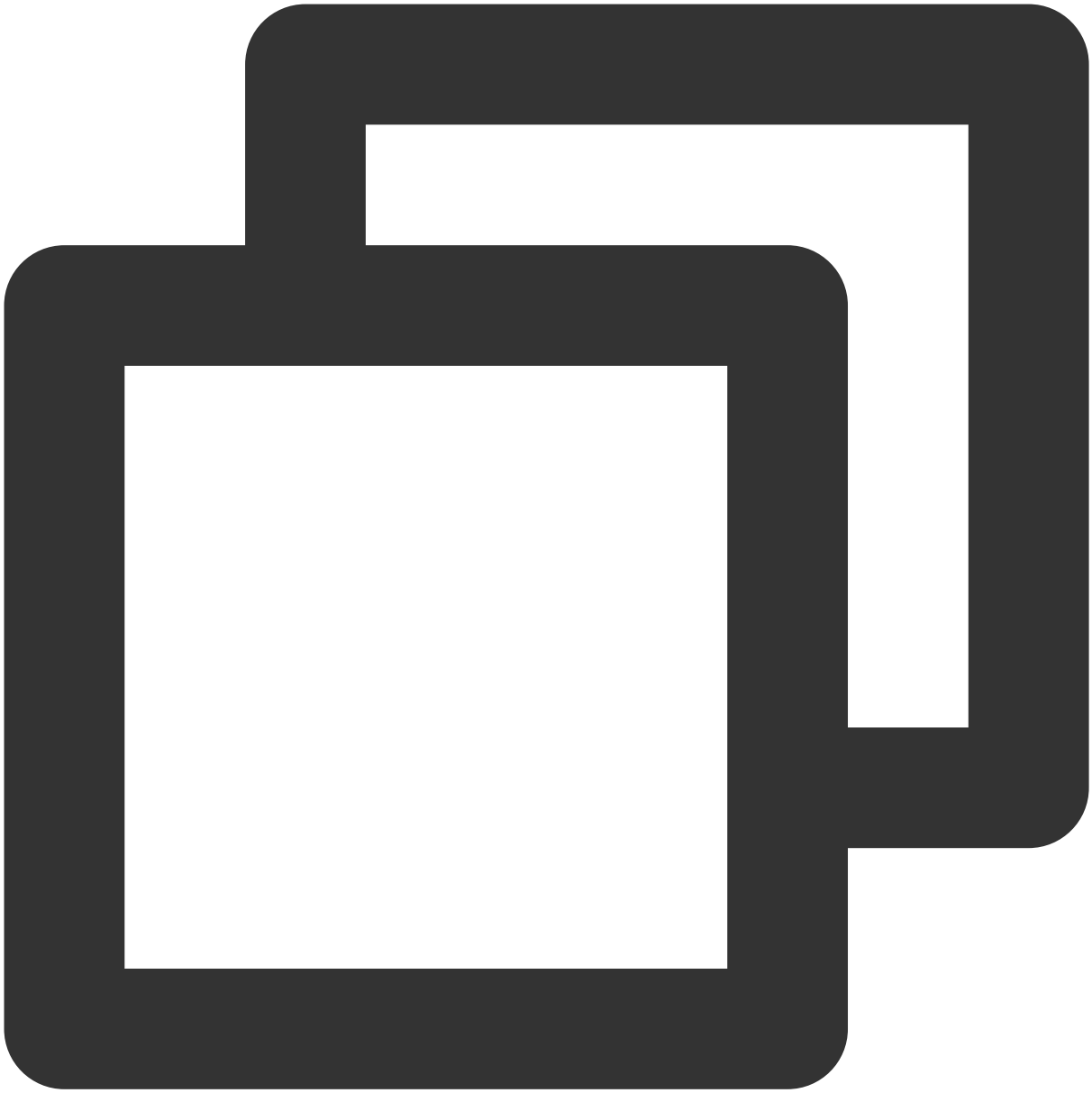
SDKAppID: The **TRTC application ID**. If you haven't activated TRTC, log in to the TRTC console, create a TRTC application, click **Application Info**, and select the **Quick Start** tab to view its `SDKAppID` .



Secretkey: The **TRTC application key**. Each secret key corresponds to a `SDKAppID` . You can view your application's secret key on the Application Management page of the TRTC console.
userId: The ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_). We recommend that you keep it consistent with your user account system.
UserSig: The security signature calculated based on `SDKAppID` , `userId` , and `Secretkey` . You can click here to quickly generate a `UserSig` for testing. For more information, see UserSig.

Step 4. Implement the audio chat room

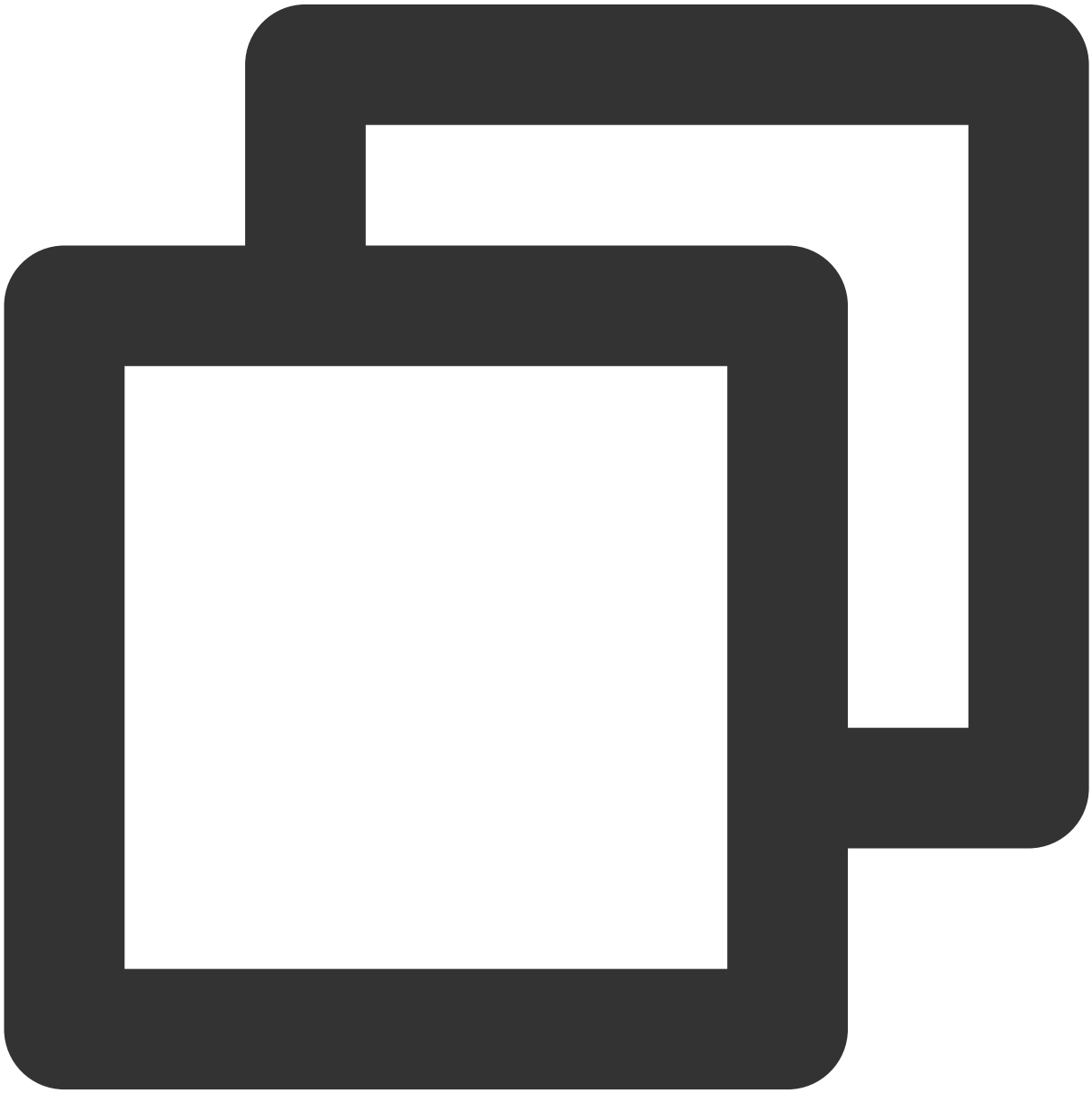
1. **The room owner creates an audio chat room through** TRTCVoiceRoom#createRoom.



```
// Initialize the audio chat room parameters
let roomParam = VoiceRoomParam()
roomParam.roomName = "Room name"
roomParam.needRequest = false // Whether the room owner’s permission is required fo
roomParam.coverUrl = "URL of room cover image"
roomParam.seatCount = 7 // Number of room seats. In this example, the number is 7.
roomParam.seatInfoList = []
// Initialize the seat information
for _ in 0..< param.seatCount {
 let seatInfo = VoiceRoomSeatInfo()
 param.seatInfoList.append(seatInfo)
```

```
}
// Create a room
mTRTCVoiceRoom.createRoom(roomID: yourRoomID, roomParam: roomParam) { (code, messag
 if code == 0 {
     // Group created successfully
 }
}
```

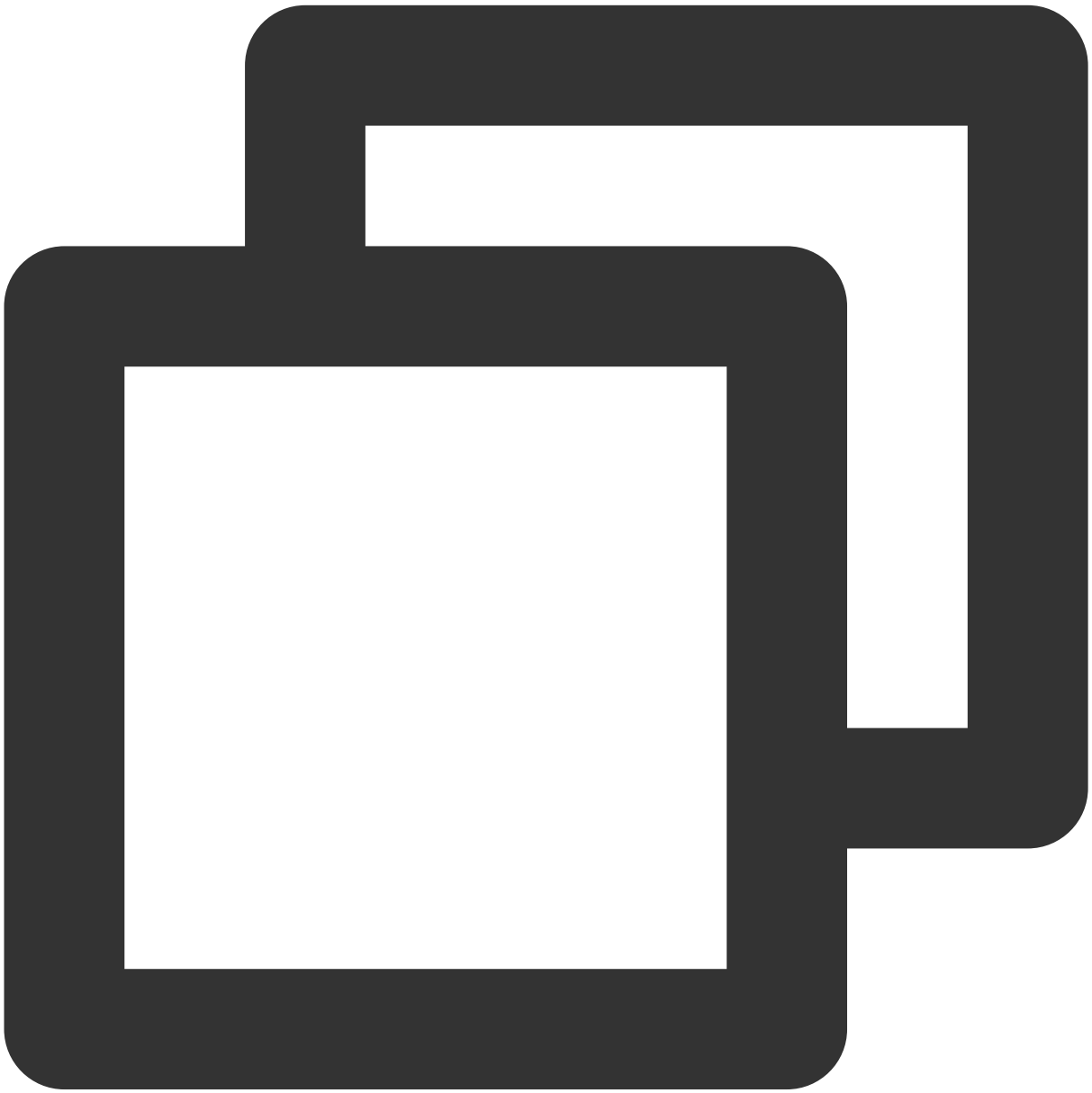
2. **A listener enters the audio chat room through** TRTCVoiceRoom#enterRoom.



```
// 1. A listener calls an API to enter the room
mTRTCVoiceRoom.enterRoom(roomID: roomID) { (code, message) in
```

```
 // Callback of the room entry result
 if code == 0 {
    // Entered room successfully
 }
}
```

3. **A listener mics on through** TRTCVoiceRoom#enterSeat.

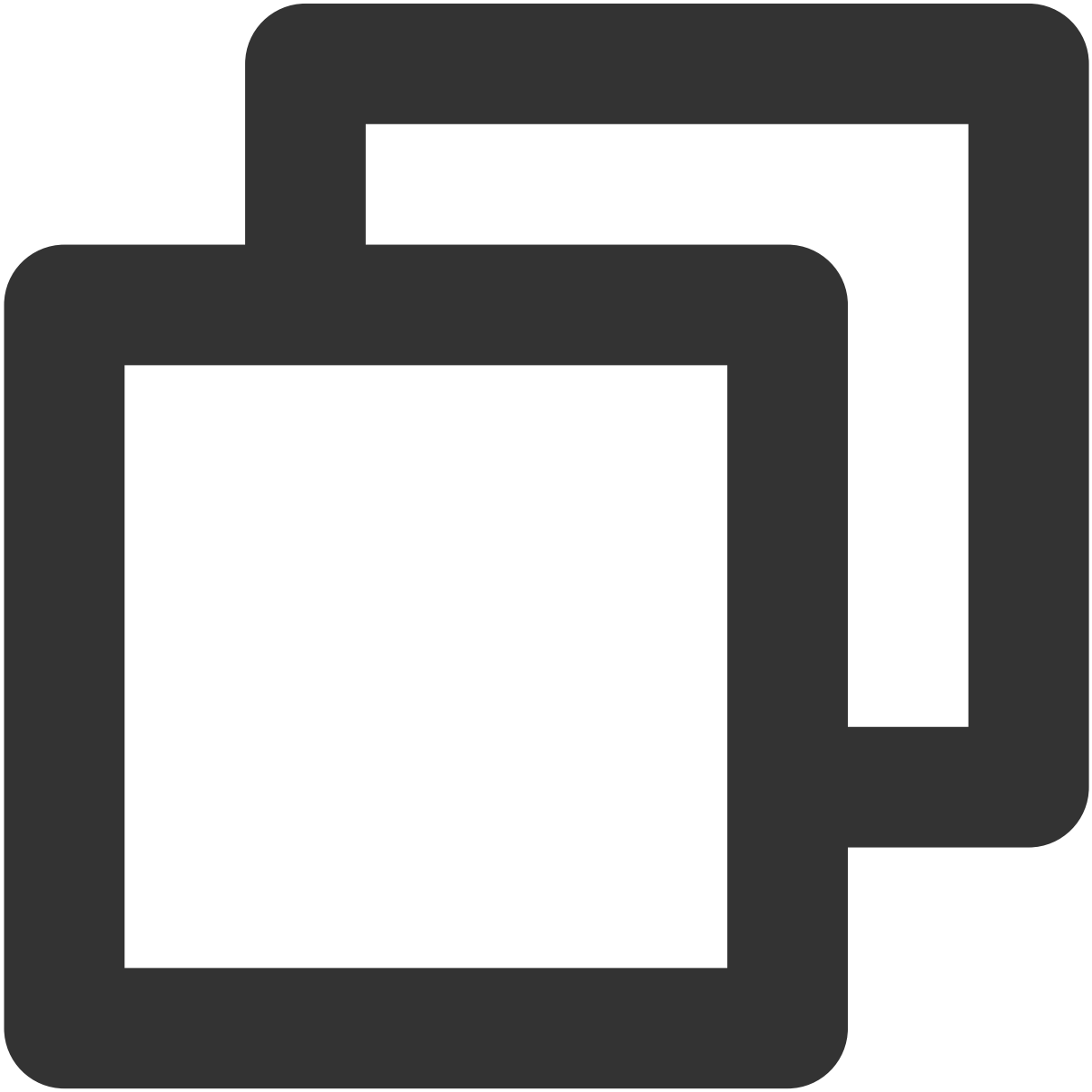


```
// 1. A listener calls an API to mic on
let seatIndex = 2; // Seat index
mTRTCVoiceRoom.enterSeat(seatIndex: 2) { (code, message) in
 if code == 0 {
```

```
      // Mic turned on successfully
  }
}

// 2. The `onSeatListChange` callback is received, and the seat list is refreshed
@Override
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
  // Refreshed seat list
}
```

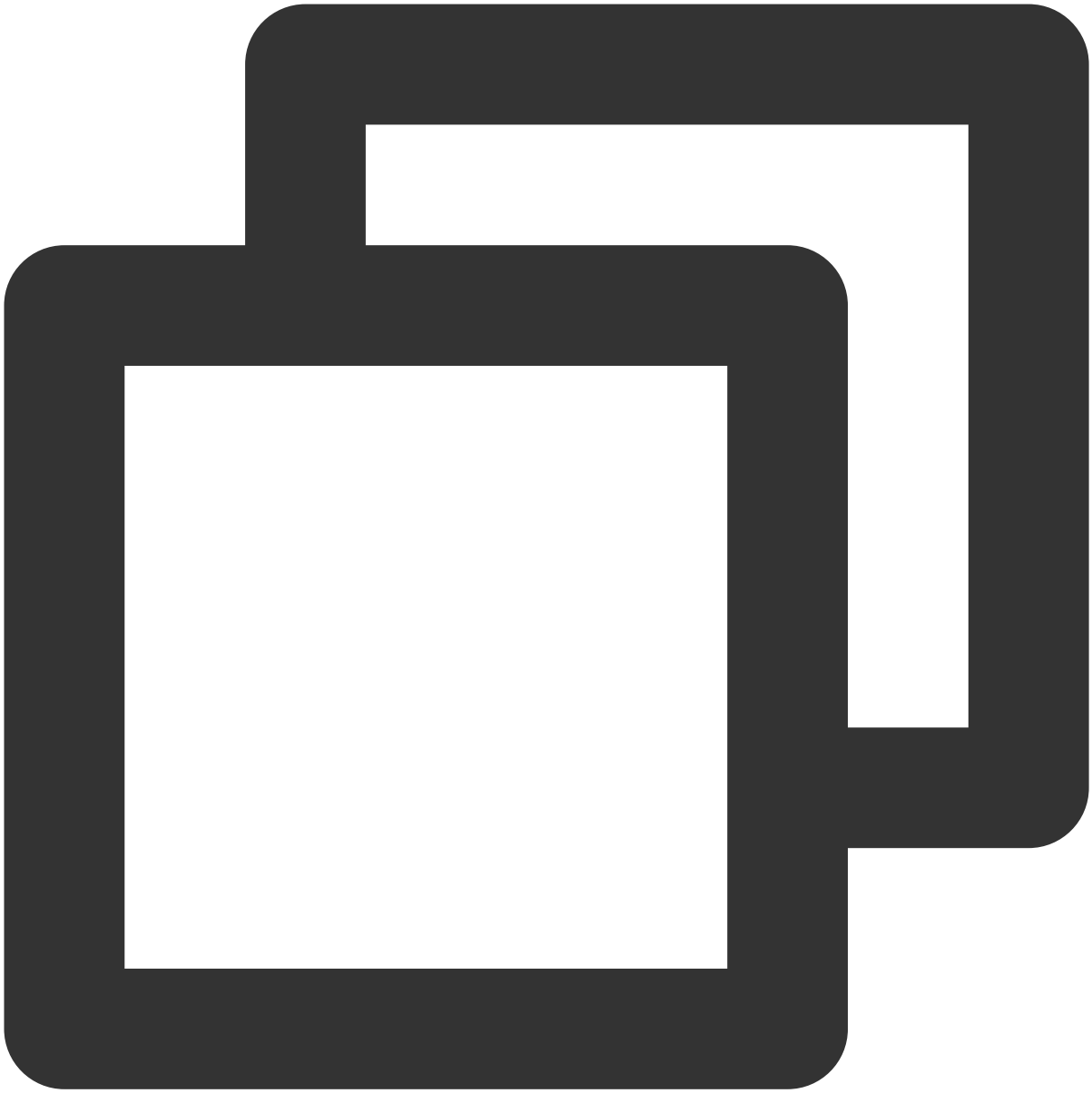
4. **The room owner makes a listener speaker through** TRTCVoiceRoom#pickSeat.



```
// 1. The room owner makes a listener a speaker
let seatIndex = 2; // Seat index
let userId = "123"; // ID of the user to speak
mTRTCVoiceRoom.pickSeat(seatIndex: 1, userId: "123") { (code, message) in
 if code == 0 {
 }
}

// 2. The `onSeatListChange` callback is received, and the seat list is refreshed
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
 // Refreshed seat list
}
```

5. **A listener requests to speak through** TRTCVoiceRoom#sendInvitation.



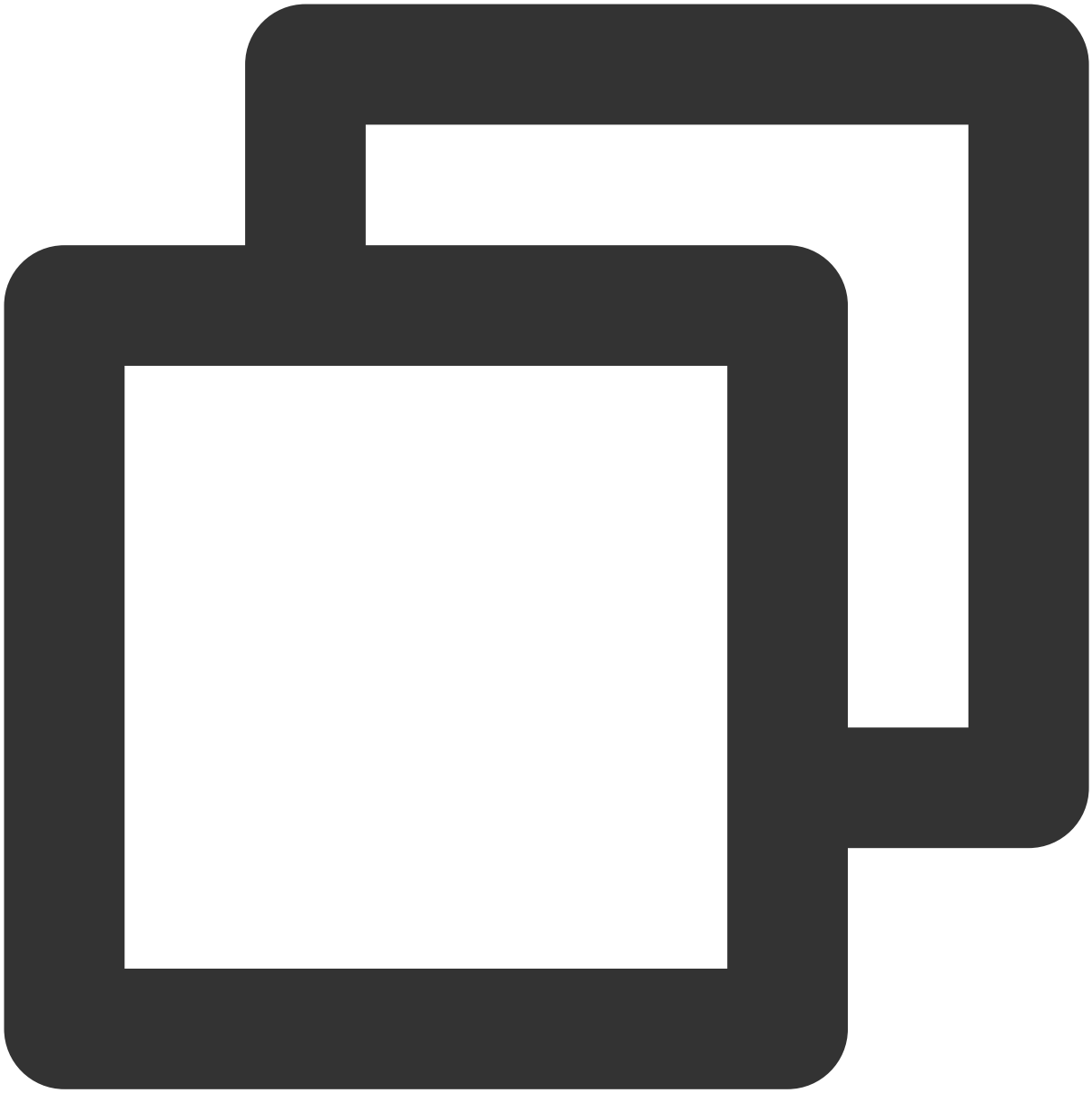
```
// Listener
// 1. A listener calls an API to request to speak
let seatIndex = "1"; // Seat index
let userId = "123"; // User ID
let inviteId = mTRTCVoiceRoom.sendInvitation(cmd: "takeSeat", userId: ownerUserId,
 // Callback of the result
}

// 2. Place the user in the seat after the invitation is accepted
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
```

```
        self.mTRTCVoiceRoom.enterSeat(seatIndex: ) { (code, message) in
            // Callback of the result
        }
    }
}

// Room owner
// 1. The room owner receives the request
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, conte
    if cmd == "takeSeat" {
        // 2. The room owner accepts the request
        self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
    }
}
```

6. **The room owner invites a listener to speak through** TRTCVoiceRoom#sendInvitation.



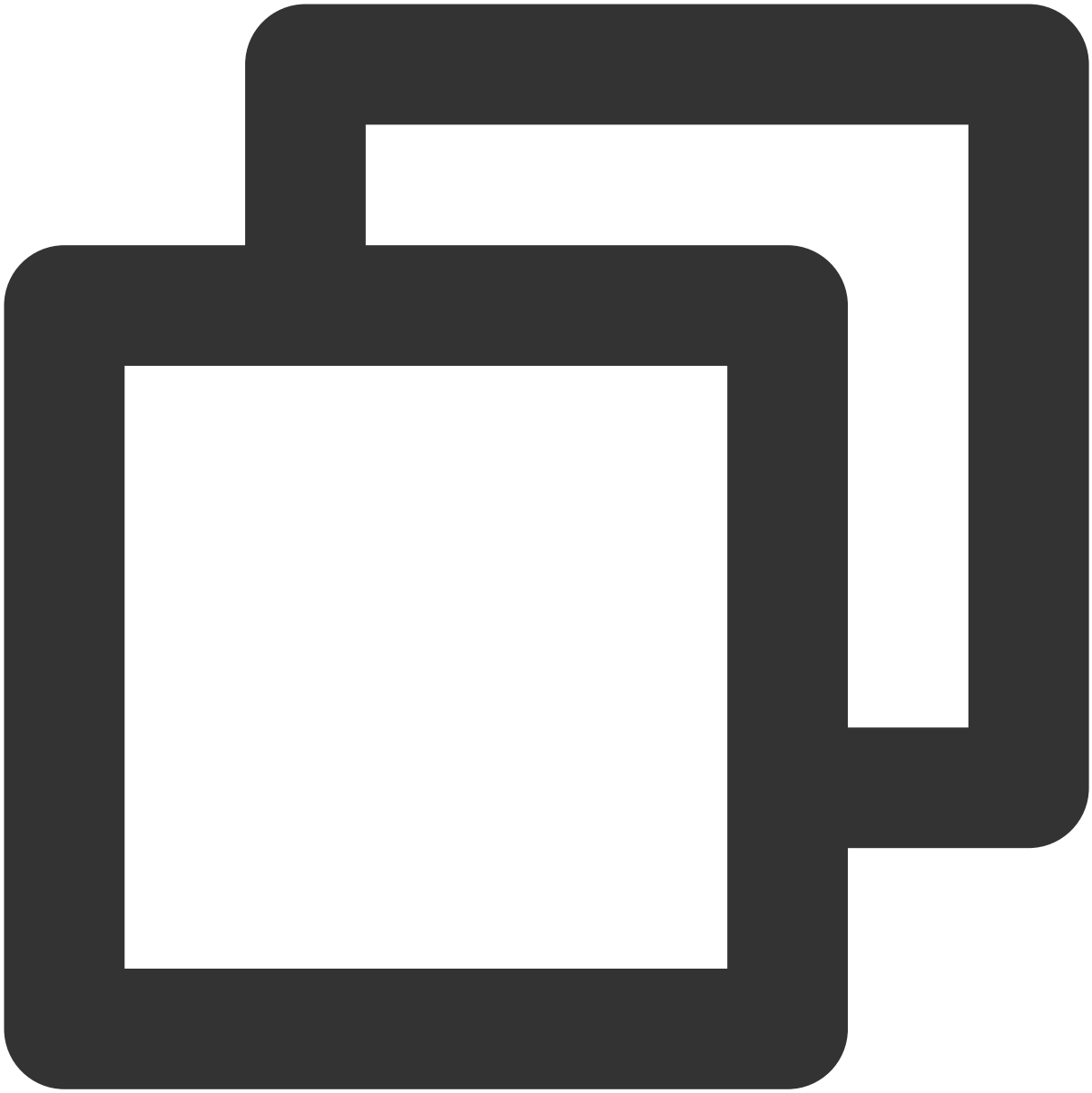
```
// Room owner
// 1. Call `sendInvitation` to invite user `123` to take seat 2
let inviteId = self.mTRTCVoiceRoom.sendInvitation(cmd: "pickSeat", userId: ownerUse
 // Callback of the result
}

// 2. Place the user in the seat after the invitation is accepted
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
     self.mTRTCVoiceRoom.pickSeat(seatIndex: ) { (code, message) in
         // Callback of the result
```

```
        }
    }
}

// Listener
// 1. The listener receives the invitation
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, conte
    if cmd == "pickSeat" {
        // 2. The listener accepts the invitation
        self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
    }
}
```

7. **Implement text chat through** TRTCVoiceRoom#sendRoomTextMsg.

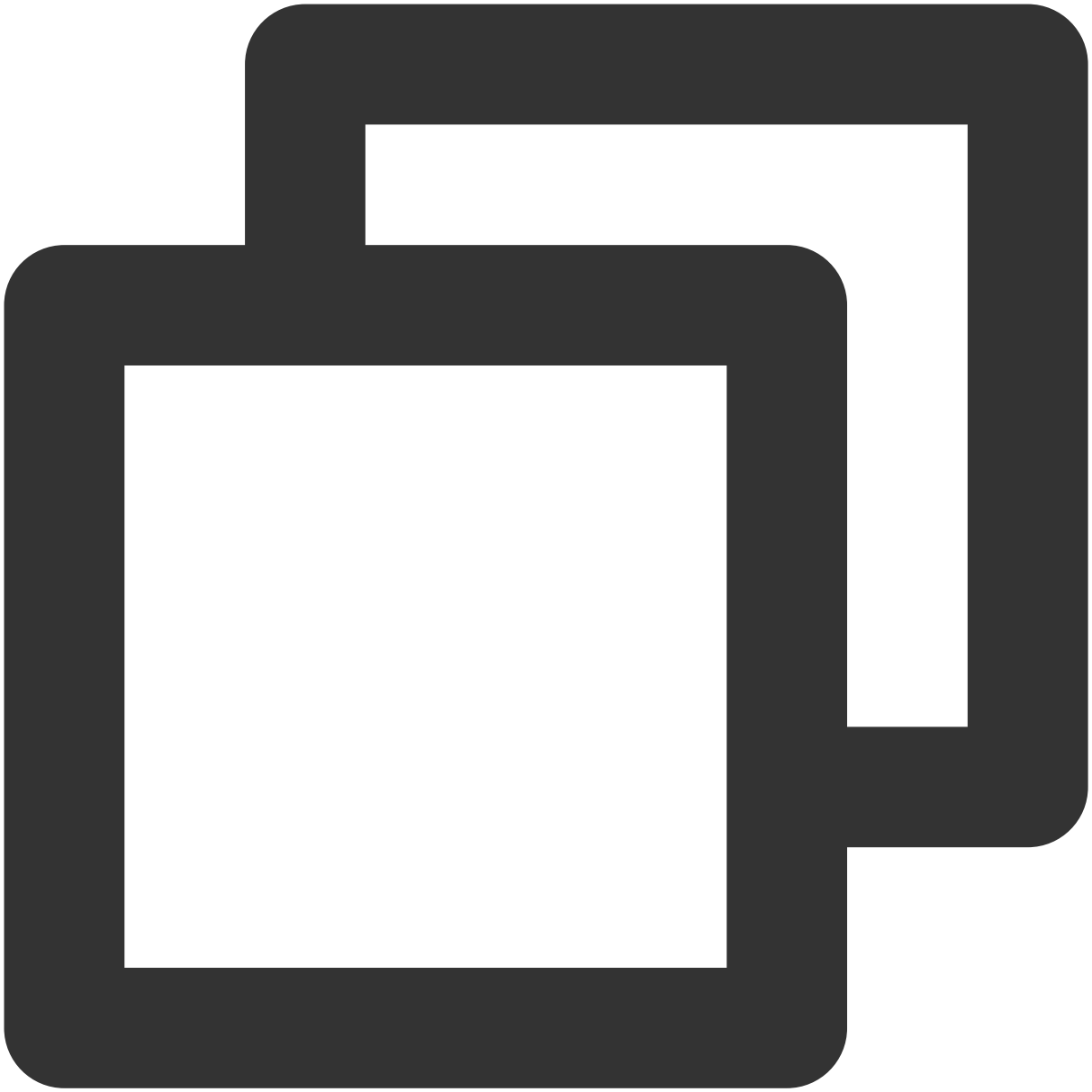


```
// Sender: Sends text chat messages
self.mTRTCVoiceRoom.sendRoomTextMsg(message: message) { (code, message) in

}

// Receiver: Listens for text chat messages
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
 // Handling of the messages received
}
```

8. **Implement on-screen commenting through** TRTCVoiceRoom#sendRoomCustomMsg.



```
// For example, a sender can customize commands to distinguish on-screen comments a
// For example, use "CMD_DANMU" to indicate on-screen comments and "CMD_LIKE" to in
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_DANMU", message: "hello world", cal
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)

// Receiver: Listens for custom messages
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserInfo)
 if cmd == "CMD_DANMU" {
     // An on-screen comment is received
 }
 if cmd == "CMD_LIKE" {
```

```
        // A like is received
    }
  }
```

Suggestions and Feedback

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

TUIVoiceRoom APIs
TRTCVoiceRoom (iOS)

Last updated：2023-09-25 10:52:31

`TRTCVoiceRoom` is based on Tencent Real-Time Communication (TRTC) and Tencent Cloud Chat. Wit:
A user can create an audio chat room and become a speaker or enter an audio chat room as a listener.
The room owner can invite a listener to speak as well as remove speaker from the seat.
The room owner can also block a seat. Listeners cannot request to take a blocked seat.
A listener can request to speak and become a speaker. A speaker can also become a listener.
All users can send text and custom messages. Custom messages can be used to send on-screen comments, give likes, and send gifts.

Note

All TUIKit components are based on two PaaS services of Tencent Cloud, namely TRTC and Chat. When you activate TRTC, the Chat SDK trial edition (which supports up to 100 DAUs) will be activated automatically. For billing details of Chat, see Pricing.
`TRTCVoiceRoom` is an open-source class depending on two closed-source Tencent Cloud SDKs. For the specific implementation process, see Audio Chat Room (iOS).
The TRTC SDK is used as a low-latency audio chat component.
The `AVChatRoom` feature of the Chat SDK is used to implement chat rooms. The attribute APIs of Chat are used to store room information such as the seat list, and invitation signaling is used to send requests to speak or invite others to speak.

TRTCVoiceRoom API Overview

Basic SDK APIs

API	Description
sharedInstance	Gets a singleton object.
destroySharedInstance	Terminates a singleton object.
setDelegate	Sets event callbacks.
setDelegateHandler	Sets the thread where event callbacks are.
login	Logs in.
logout	Logs out.

setSelfProfile	Sets profile.

Room APIs

API	Description
createRoom	Creates a room (called by room owner). If the room does not exist, the system will automatically create a room.
destroyRoom	Terminates a room (called by room owner).
enterRoom	Enters a room (called by listener).
exitRoom	Exits a room (called by listener).
getRoomInfoList	Gets room list details.
getUserInfoList	Gets the user information of the specified `userId` . If the value is `nil` , the information of all users in the room is obtained.

Seat management APIs

API	Description
enterSeat	Becomes a speaker (called by room owner or listener).
moveSeat	Changes the seat (called by speaker).
leaveSeat	Becomes a listener (called by speaker).
pickSeat	Places a user in a seat (called by room owner).
kickSeat	Removes a speaker (called by room owner).
muteSeat	Mutes/Unmutes a seat (called by room owner).
closeSeat	Blocks/Unblocks a seat (called by room owner).

Local audio APIs

API	Description
startMicrophone	Starts mic capturing.
stopMicrophone	Stops mic capturing.

setAudioQuality	Sets audio quality.
muteLocalAudio	Mutes/Unmutes local audio.
setSpeaker	Sets whether to play sound from the device's speaker or receiver.
setAudioCaptureVolume	Sets mic capturing volume.
setAudioPlayoutVolume	Sets playback volume.
setVoiceEarMonitorEnable	Enables/Disables in-ear monitoring.

Remote audio APIs

API	Description
muteRemoteAudio	Mutes/Unmutes a specified member.
muteAllRemoteAudio	Mutes/Unmutes all members.

Background music and audio effect APIs

API	Description
getAudioEffectManager	Gets the background music and audio effect management object TXAudioEffectManager.

Message sending APIs

API	Description
sendRoomTextMsg	Broadcasts a text chat message in a room. This API is generally used for on-screen comments.
sendRoomCustomMsg	Sends a custom text message.

Invitation signaling APIs

API	Description
sendInvitation	Sends an invitation.
acceptInvitation	Accepts an invitation.
rejectInvitation	Declines an invitation.

cancelInvitation	Cancels an invitation.

TRTCVoiceRoomDelegate API Overview

Common event callbacks

API	Description
onError	Callback for error.
onWarning	Callback for warning.
onDebugLog	Callback of log.

Room event callback APIs

API	Description
onRoomDestroy	The room was terminated.
onRoomInfoChange	The room information changed.
onUserVolumeUpdate	User volume

Seat list change callback APIs

API	Description
onSeatListChange	All seat changes
onAnchorEnterSeat	Someone became a speaker or was made a speaker by the room owner.
onAnchorLeaveSeat	Someone became a listener or was made a listener by the room owner.
onSeatMute	The room owner muted a seat.
onUserMicrophoneMute	Whether a user's mic is muted
onSeatClose	The room owner blocked a seat.

Callback APIs for room entry/exit by listener

API	Description
onAudienceEnter	A listener entered the room.
onAudienceExit	A listener exited the room.

Message event callback APIs

API	Description
onRecvRoomTextMsg	A text chat message was received.
onRecvRoomCustomMsg	A custom message was received.

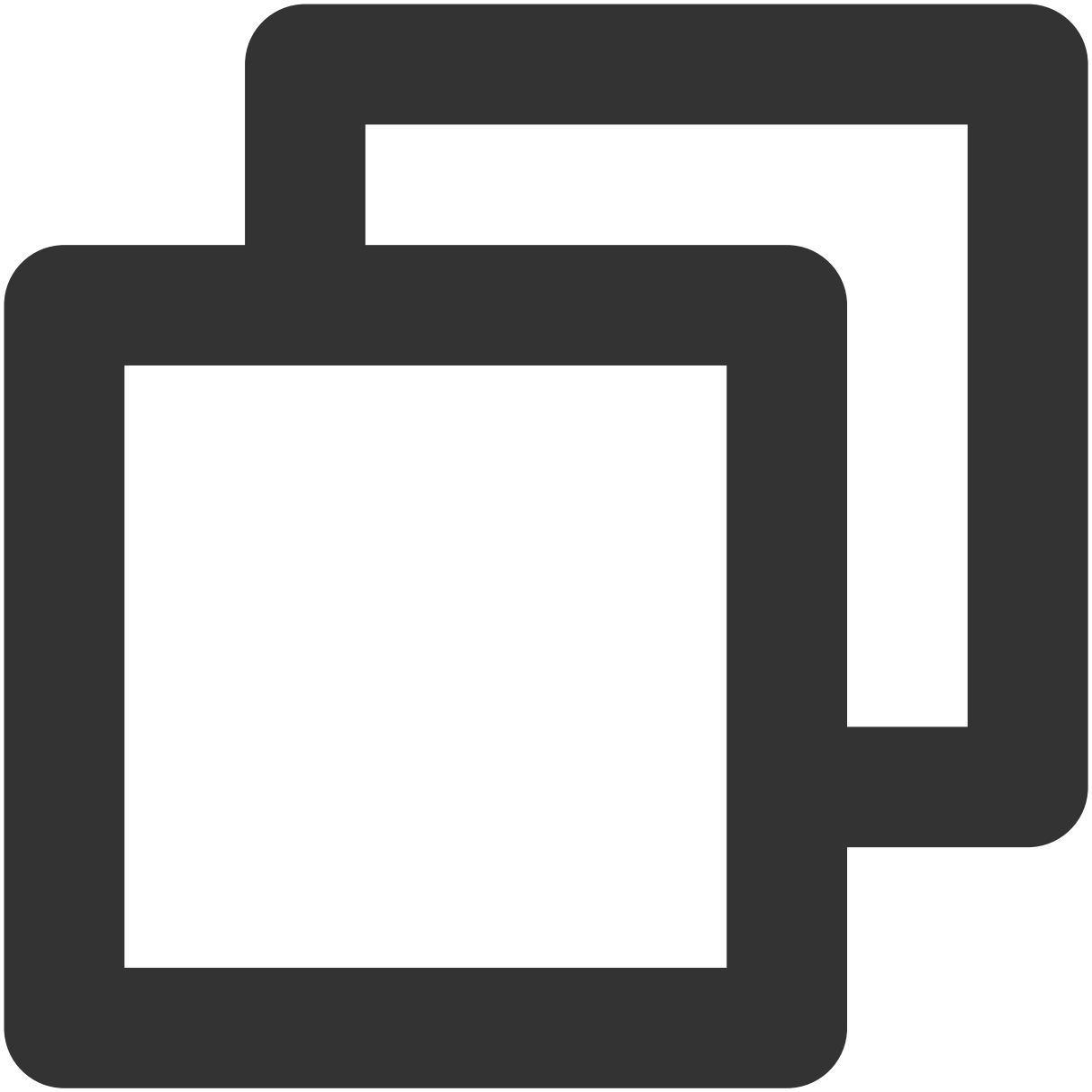
Signaling event callback APIs

API	Description
onReceiveNewInvitation	An invitation was received.
onInviteeAccepted	The invitee accepted the invitation.
onInviteeRejected	The invitee declined the invitation.
onInvitationCancelled	The inviter canceled the invitation.

Basic SDK APIs

sharedInstance

This API is used to get a TRTCVoiceRoom singleton object.



```
/**
* Get a `TRTCVoiceRoom` singleton object
*
* - returns: `TRTCVoiceRoom` instance
* - note: to terminate a singleton object, call {@link TRTCVoiceRoom#destroySharedI
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

This API is used to terminate a TRTCVoiceRoom singleton object.

explain

After the instance is terminated, the externally cached `TRTCVoiceRoom` instance can no longer be used. You need to call sharedInstance again to get a new instance.

```
/**
* Terminate a `TRTCVoiceRoom` singleton object
*
* - Note: After the instance is terminated, the externally cached `TRTCVoiceRoom` i
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

This API is used to set the event callback of TRTCVoiceRoom. You can use `TRTCVoiceRoomDelegate` to get different status notifications of TRTCVoiceRoom.

```
/**
* Set the event callbacks of the component
*
* You can use `TRTCVoiceRoomDelegate` to get different status notifications of `TRT
*
```

```
 * - parameter delegate Callback API
 * - Note: Callback events in `TRTCVoiceRoom` are called back to you in the main que
 */
 - (void)setDelegate:(id<TRTCVoiceRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(d
```

explain

`setDelegate` is the delegate callback of `TRTCVoiceRoom` .

setDelegateQueue

This API is used to set the thread queue for event callbacks. The main thread (MainQueue) is used by default.

```
/**
* Set the queue for event callbacks
*
* - parameter queue Queue. Various status callback notifications in `TRTCVoiceRoom`
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(que
```

The parameters are described below:

Parameter	Type	Description
queue	dispatch_queue_t	The status notifications of `TRTCVoiceRoom` are sent to the thread queue you specify.

login

Login

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

The parameters are described below:

Parameter	Type	Description
sdkAppId	int	You can view `SDKAppID` in Application Management > **Application Info** of

		the TRTC console.
userId	NSString	ID of the current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_)
userSig	NSString	Tencent Cloud's proprietary security signature. For how to calculate and use it, see FAQs > UserSig.
callback	ActionCallback	Callback for login. The code is 0 if login succeeds.

logout

Log out

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

The parameters are described below:

Parameter	Type	Description
callback	ActionCallback	Callback for logout. The code is `0` if logout succeeds.

setSelfProfile

This API is used to set the profile.

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callbac
```

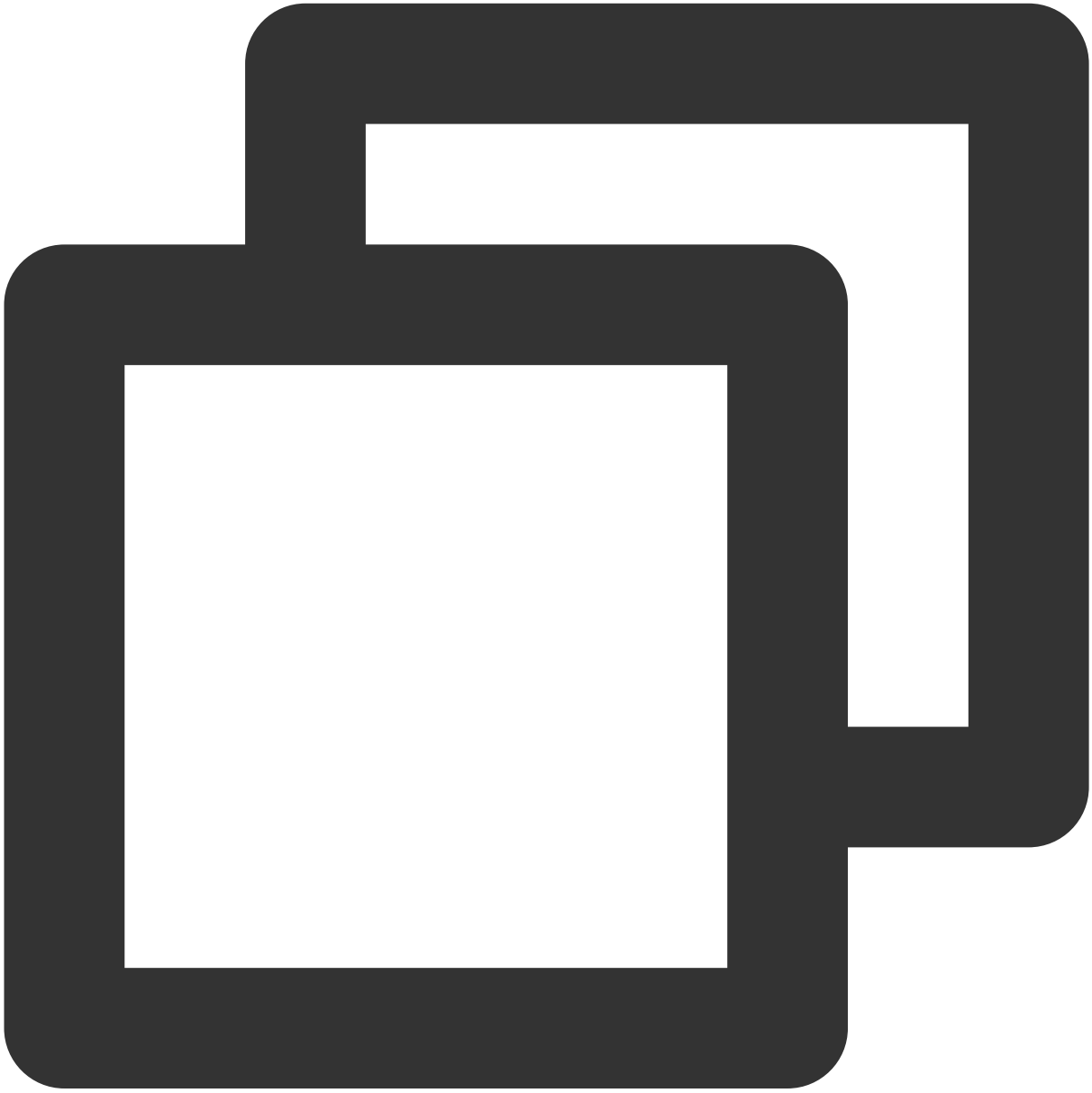
The parameters are described below:

Parameter	Type	Description
userName	NSString	Username
avatarURL	NSString	Profile picture URL
callback	ActionCallback	Callback for profile setting. The code is `0` if the operation succeeds.

Room APIs

createRoom

This API is used to create a room (called by room owner).



```
- (void)createRoom:(int)roomID roomParam:(VoiceRoomParam *)roomParam callback:(Acti
```

The parameters are described below:

Parameter	Type	Description

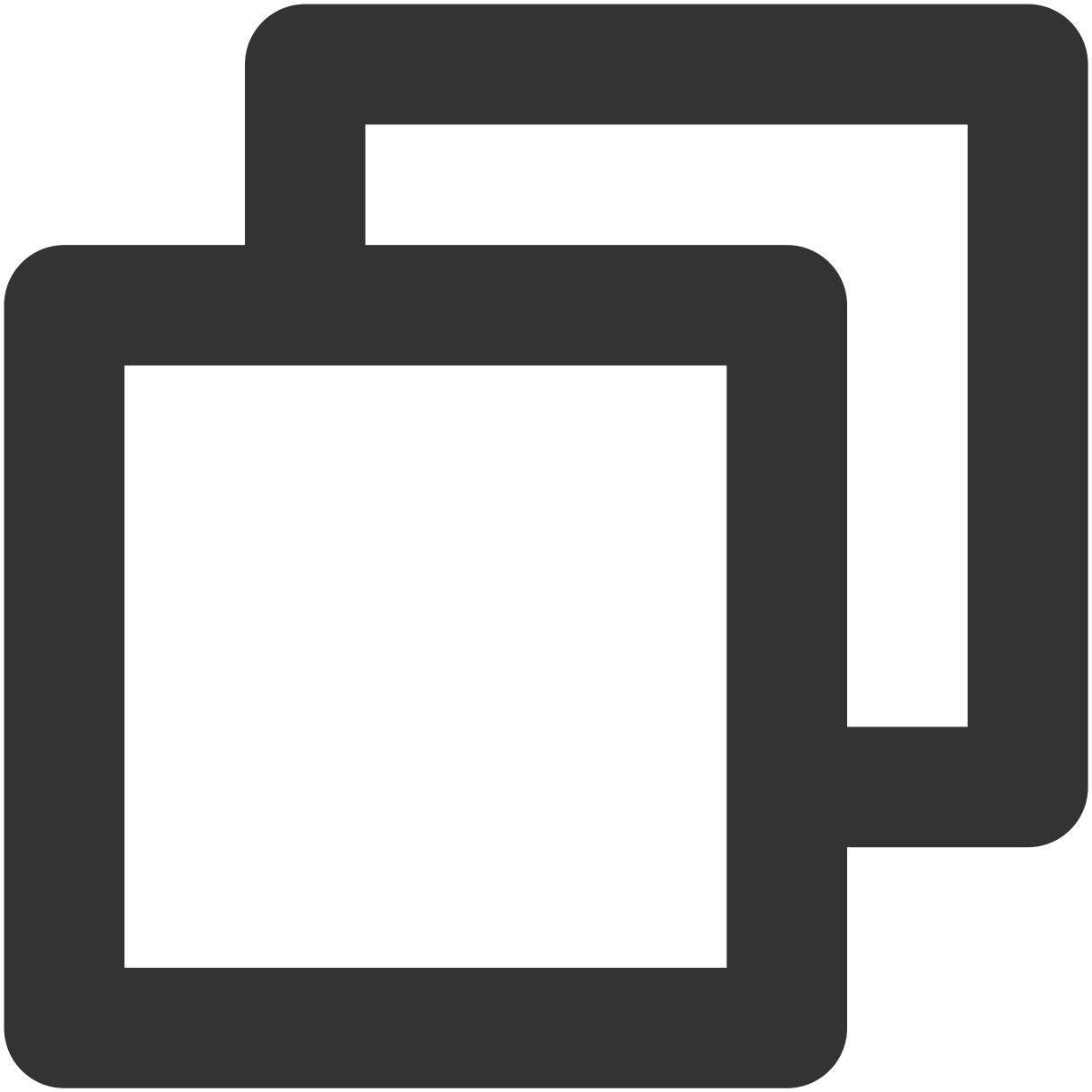
roomId	int	The room ID. You need to assign and manage the IDs in a centralized manner. Multiple `roomID` values can be aggregated into a karaoke room list. Currently, Tencent Cloud does not provide management services for room lists. Please manage your own room lists.
roomParam	VoiceRoomParam	Room information, such as room name, seat list information, and cover information. To manage seats, you must enter the number of seats in the room.
callback	ActionCallback	Callback for room creation. The code is 0 if the operation succeeds.

The process of creating a karaoke room and becoming a speaker is as follows:

1. A user calls `createRoom` to create an audio chat room, passing in room attributes (e.g. room ID, whether listeners require room owner's consent to speak, number of seats).
2. After creating the room, the user calls `enterSeat` to become a speaker.
3. The user will receive an `onSeatListChanget` notification about the change of the seat list, and can update the change to the UI.
4. The user will also receive an `onAnchorEnterSeat` notification that someone became a speaker, and mic capturing will be enabled automatically.

destroyRoom

This API is used to terminate a room (called by room owner).



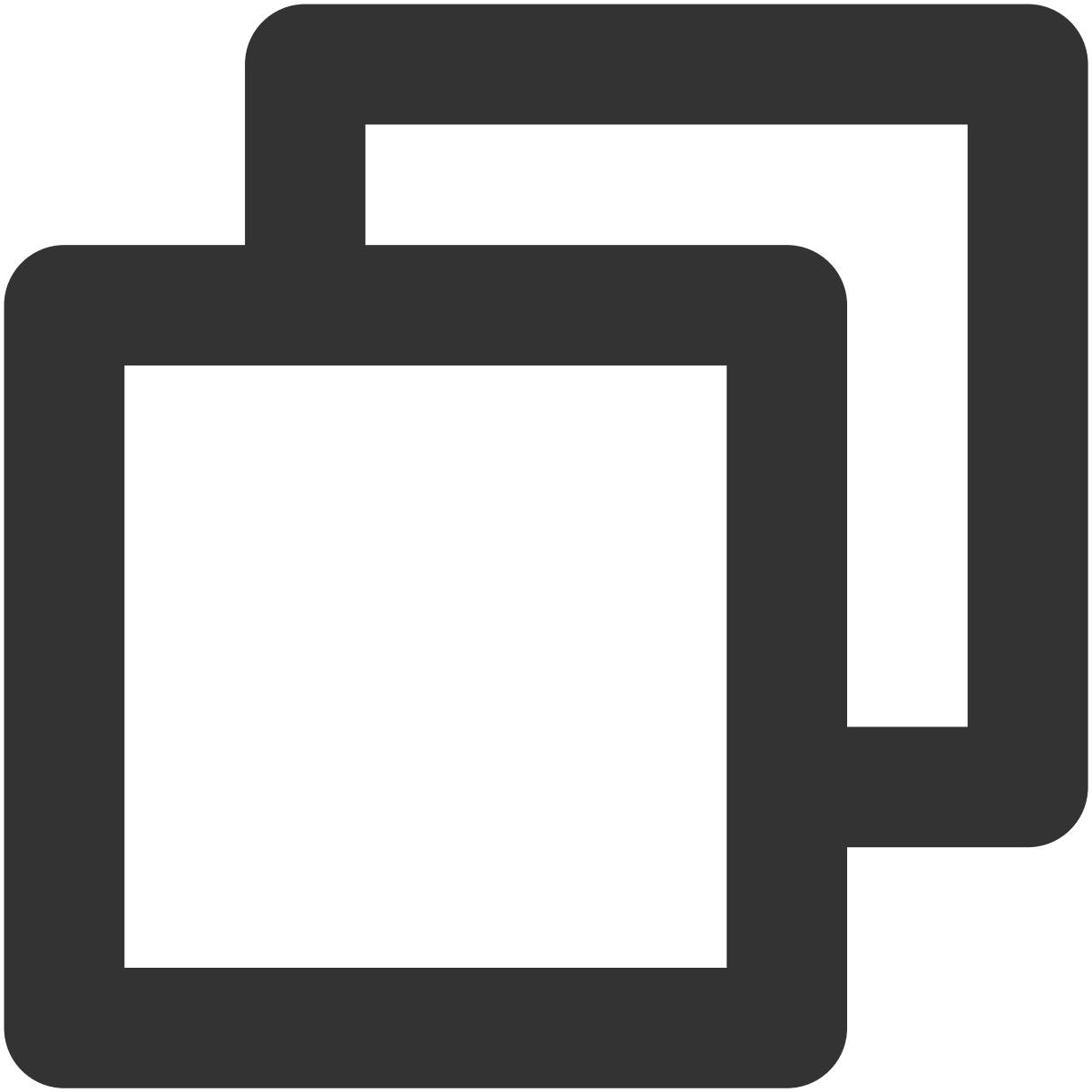
```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(ca
```

The parameters are described below:

Parameter	Type	Description
callback	ActionCallback	Callback for room termination. The code is `0` if the operation succeeds.

enterRoom

This API is used to enter a room (called by listener).



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

The parameters are described below:

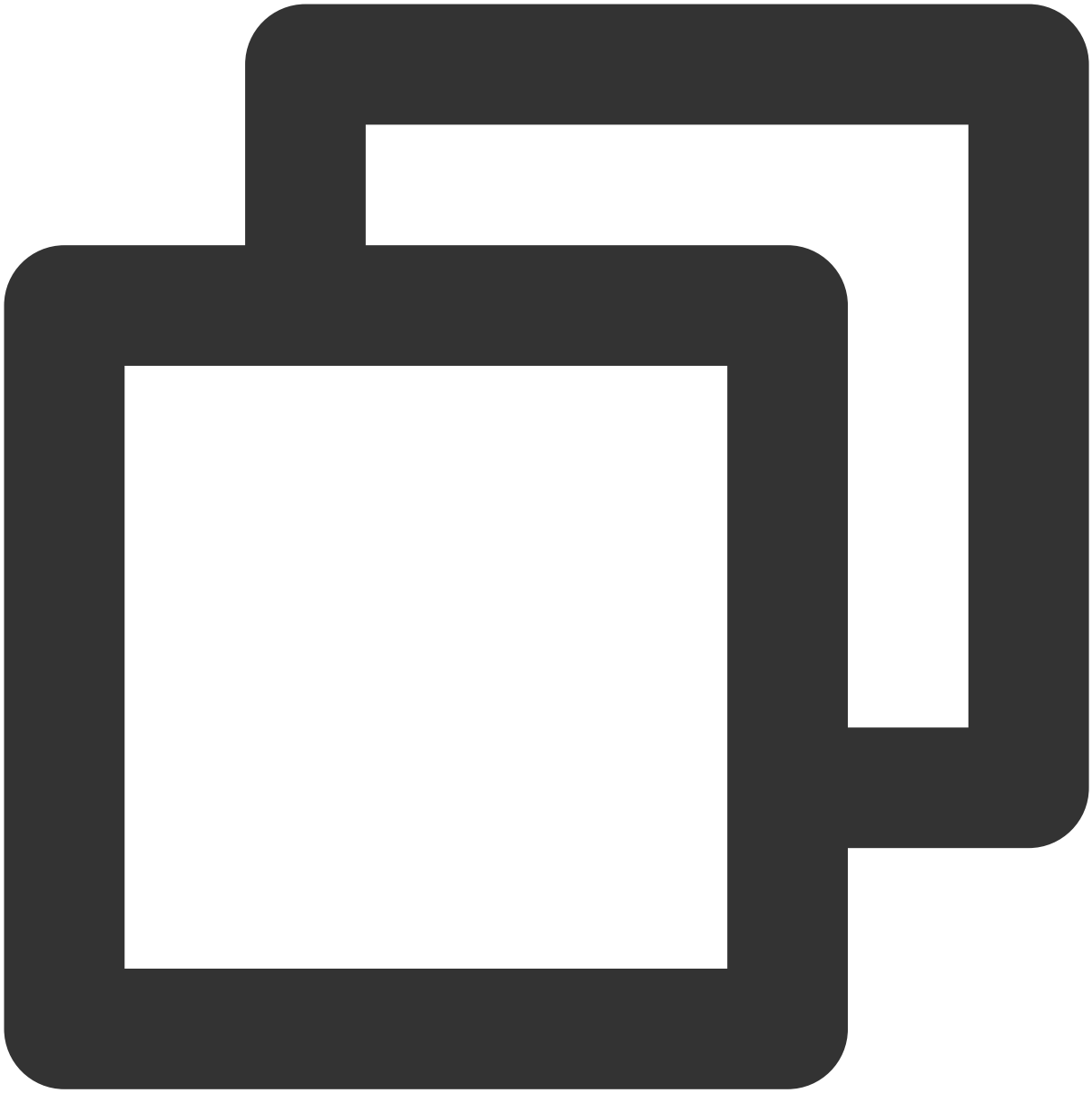
Parameter	Type	Description
roomId	NSInteger	The room ID.
callback	ActionCallback	Callback for room entry. The code is `0` if the operation succeeds.

The process of entering a room as a listener is as follows:

1. A user gets the latest audio chat room list from your server. The list may contain the `roomId` and room information of multiple audio chat rooms.
2. The user selects a room, and calls `enterRoom` with the room ID passed in to enter the room.
3. After entering the room, the user receives an `onRoomInfoChange` notification about room attribute change from the component. The attributes can be recorded, and corresponding changes can be made to the UI, including room name, whether room owner's consent is required for listeners to speak, etc.
4. The user will receive an `onSeatListChange` notification about the change of the seat list and can update the change to the UI.
5. The user will also receive an `onAnchorEnterSeat` notification that someone became a speaker.

exitRoom

Leave room



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

The parameters are described below:

Parameter	Type	Description
callback	ActionCallback	Callback for room exit. The code is `0` if the operation succeeds.

getRoomInfoList

This API is used to get room list details. The room name and cover are set by the room owner via `roomInfo` when calling `createRoom()` .

explain

You don't need this API if both the room list and room information are managed on your server.

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(VoiceRoomInfoCa
```

The parameters are described below:

Parameter	Type	Description

roomIdList	NSArray<NSNumber>	Room ID list
callback	RoomInfoCallback	Callback of room details

getUserInfoList

This API is used to get the information of specific users (`userId`).

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(Voice
```

The parameters are described below:

Parameter	Type	Description
userIdList	NSArray<NSString>	IDs of the users to query. If this parameter is `null` , the information of all users in the room is queried.
userlistcallback	UserListCallback	Callback of user details

Seat Management APIs

enterSeat

This API is used to become a speaker (called by room owner or listener).

explain

After a user becomes a speaker, all members in the room will receive an `onSeatListChange` notification and an `onAnchorEnterSeat` notification.

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

The parameters are described below:

Parameter	Type	Description
seatIndex	NSInteger	Number of the seat to take
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. In cases where listeners need the room owner's consent to take a seat, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept`, call this API.

moveSeat

This API is used to change one's seat (called by speaker).

explain

After the seat change, all users in the room will receive the `onSeatListChange`, `onAnchorLeaveSeat`, and `onAnchorEnterSeat` notifications. This API will only change the user's seat number, not the user role.

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

The parameters are described below:

Parameter	Type	Description
seatIndex	NSInteger	Number of the seat to change to
callback	ActionCallback	Callback for the operation

Response parameters:

Parameter	Type	Description
code	NSInteger	Result of seat change. `0` : operation successful; `10001` : API rate limit exceeded; other values: operation failed

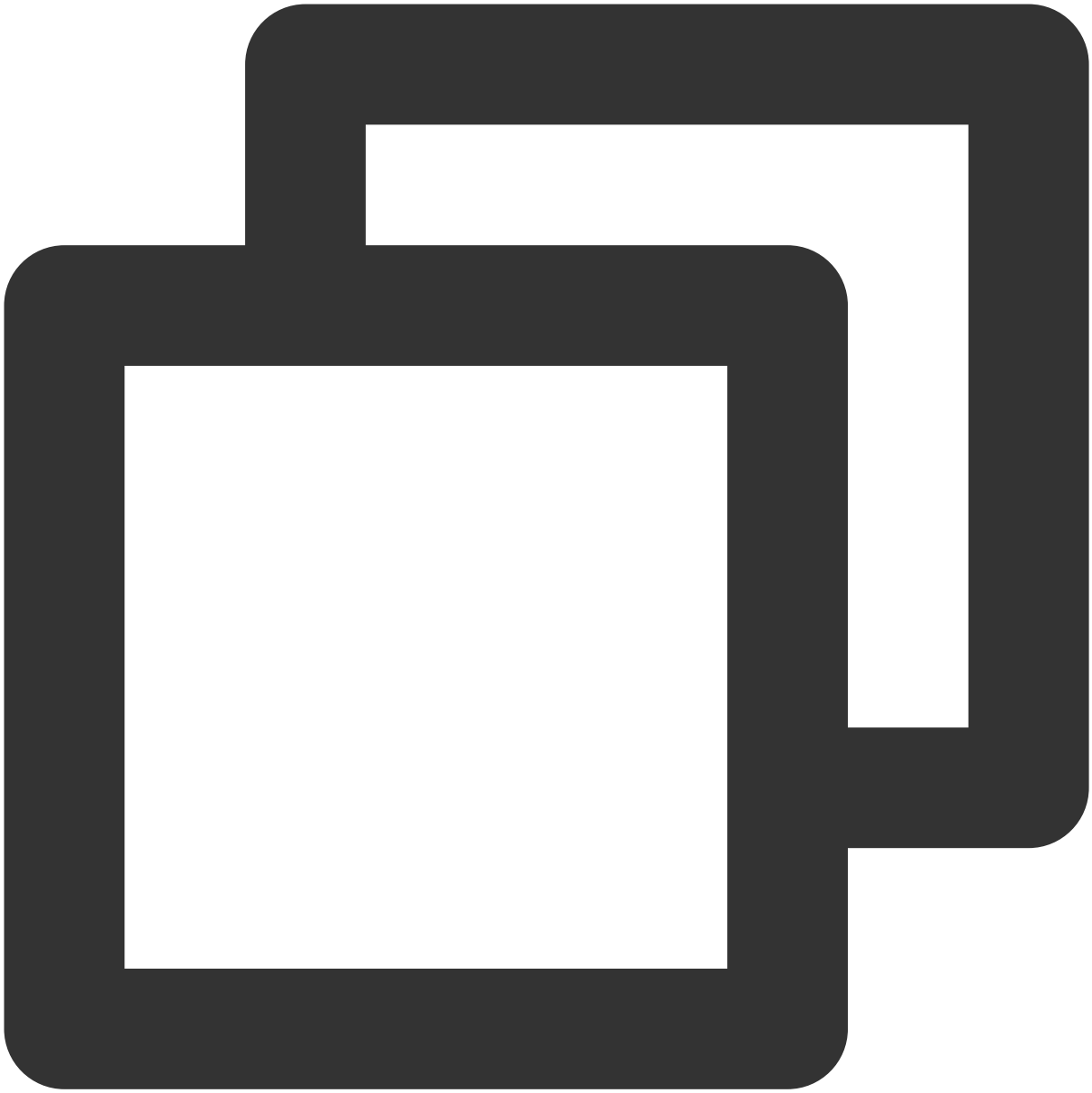
Calling this API will immediately modify the seat list. In cases where listeners need the room owner's consent to take a seat, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept` , call this API.

leaveSeat

This API is used to become a listener (called by speaker).

explain

After a speaker becomes a listener, all members in the room will receive an `onSeatListChange` notification and an `onAnchorLeaveSeat` notification.



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

The parameters are described below:

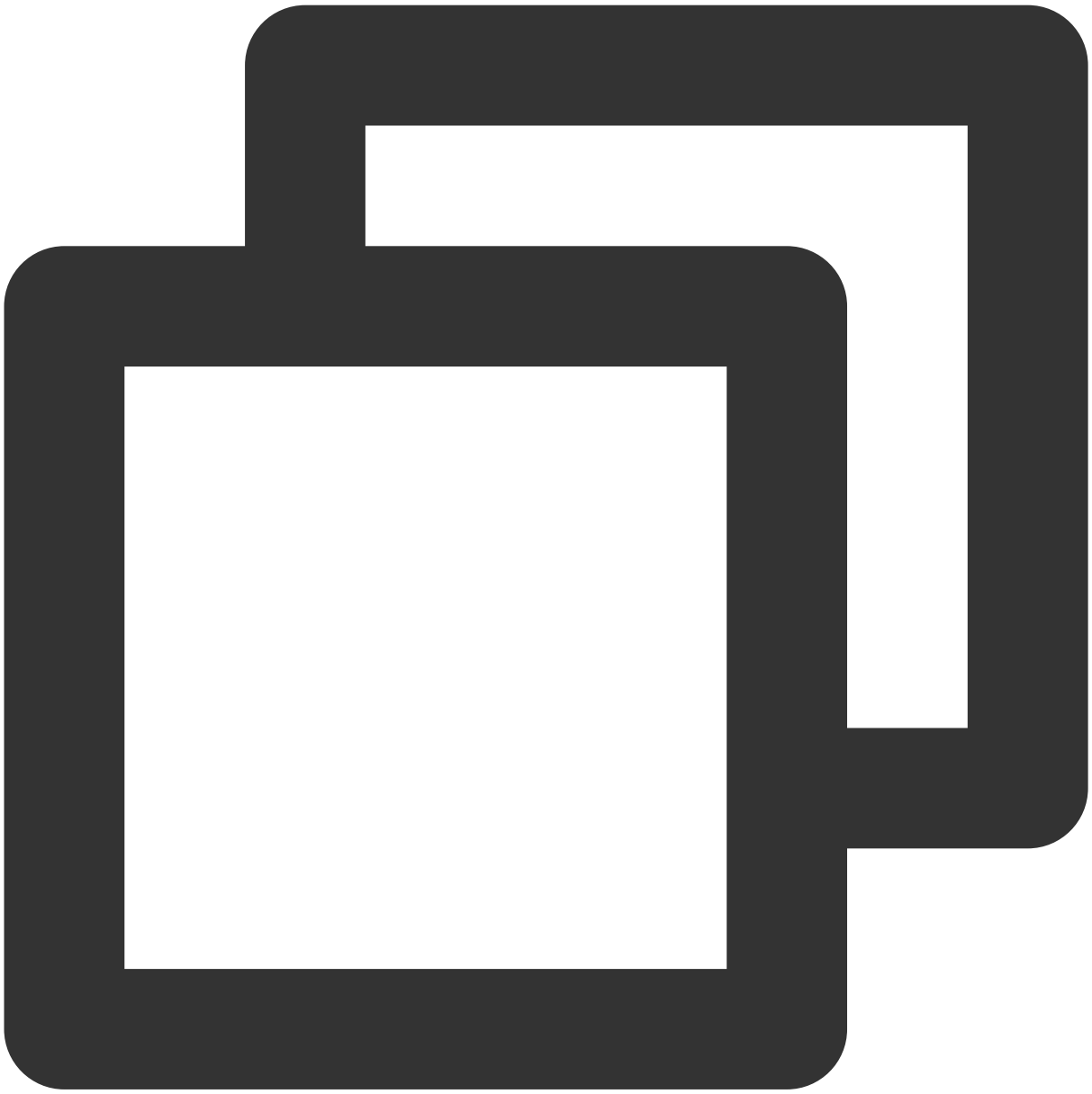
Parameter	Type	Description
callback	ActionCallback	Callback for the operation

pickSeat

This API is used to place a user in a seat (called by room owner).

explain

After the room owner places a user in a seat, all members in the room will receive an `onSeatListChange` notification and an `onAnchorEnterSeat` notification.



```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCal
```

The parameters are described below:

Parameter	Type	Description
seatIndex	NSInteger	Number of the seat to place the user in

userId	NSString	User ID
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. In cases where the room owner needs listeners' consent to make them speakers, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept` , call `pickSeat` .

kickSeat

This API is used to remove a speaker (called by room owner).

explain

After a speaker is removed, all members in the room will receive an `onSeatListChange` notification and an `onAnchorLeaveSeat` notification.

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

The parameters are described below:

Parameter	Type	Description
seatIndex	NSInteger	Seat number of the speaker to remove
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list.

muteSeat

This API is used to mute/unmute a seat (called by room owner).

explain

After a seat is muted/unmuted, all members in the room will receive an `onSeatListChange` notification and an `onSeatMute` notification.

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback
```

The parameters are described below:

Parameter	Type	Description
seatIndex	NSInteger	Number of the seat to mute/unmute
isMute	BOOL	`YES` : mute the seat; `NO` : unmute the seat
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. The speaker on the seat specified by `seatIndex` will call `muteAudio` to mute/unmute his or her audio.

closeSeat

This API is used to block/unblock a seat (called by room owner).

explain

After a seat is blocked/unblocked, all members in the room will receive an `onSeatListChange` notification and an `onSeatClose` notification.

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallba
```

The parameters are described below:

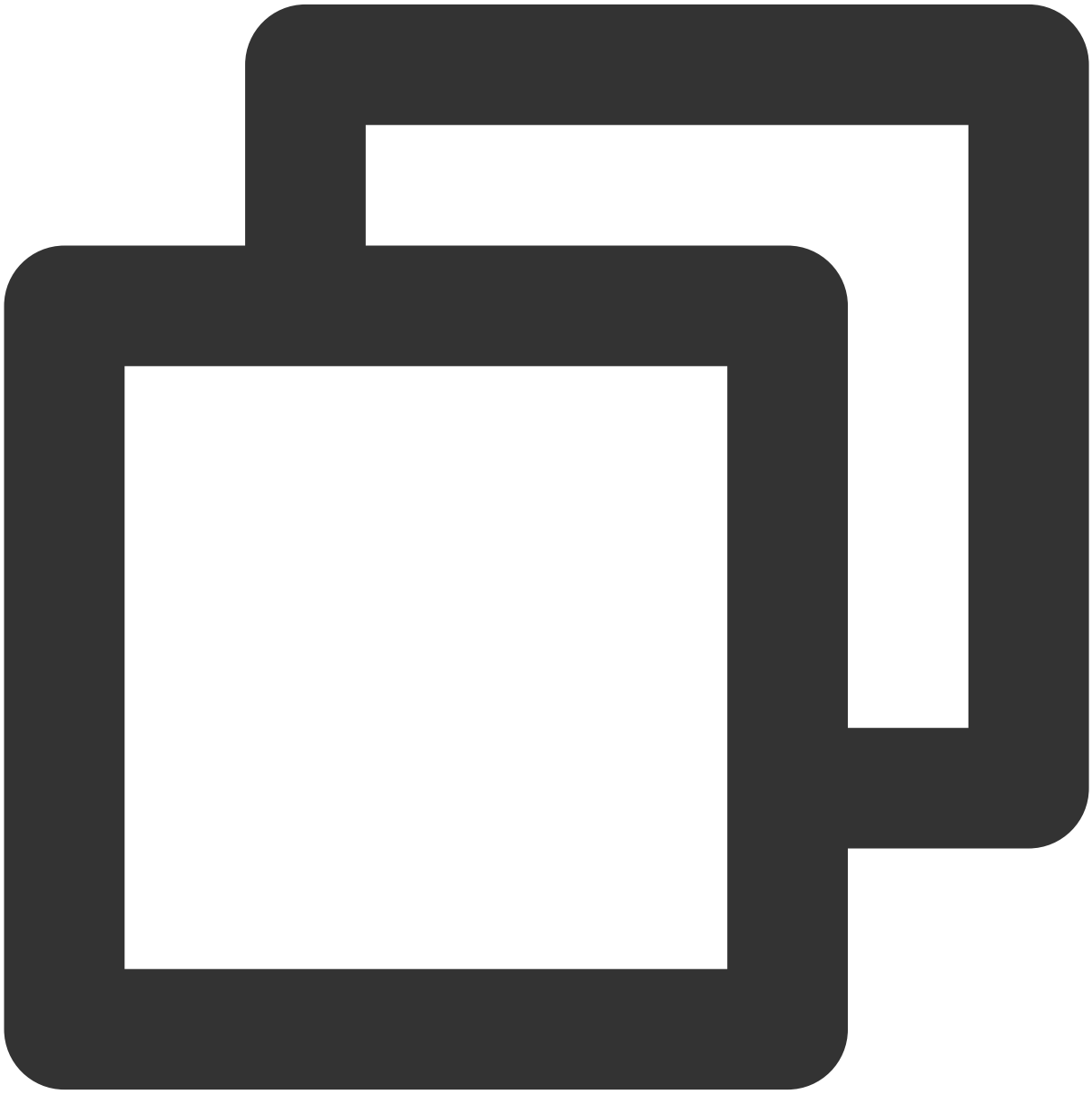
Parameter	Type	Description
seatIndex	NSInteger	Number of the seat to block/unblock
isClose	BOOL	`YES` : block the seat; `NO` : unblock the seat
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. The speaker on the seat specified by `seatIndex` will leave the seat.

Local Audio APIs

startMicrophone

This API is used to start mic capturing.



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

stopMicrophone

This API is used to stop mic capturing.

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

setAudioQuality

This API is used to set audio quality.

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

The parameters are described below:

Parameter	Type	Description
quality	NSInteger	The audio quality. For more information, see setAudioQuality().

muteLocalAudio

This API is used to mute/unmute local audio.

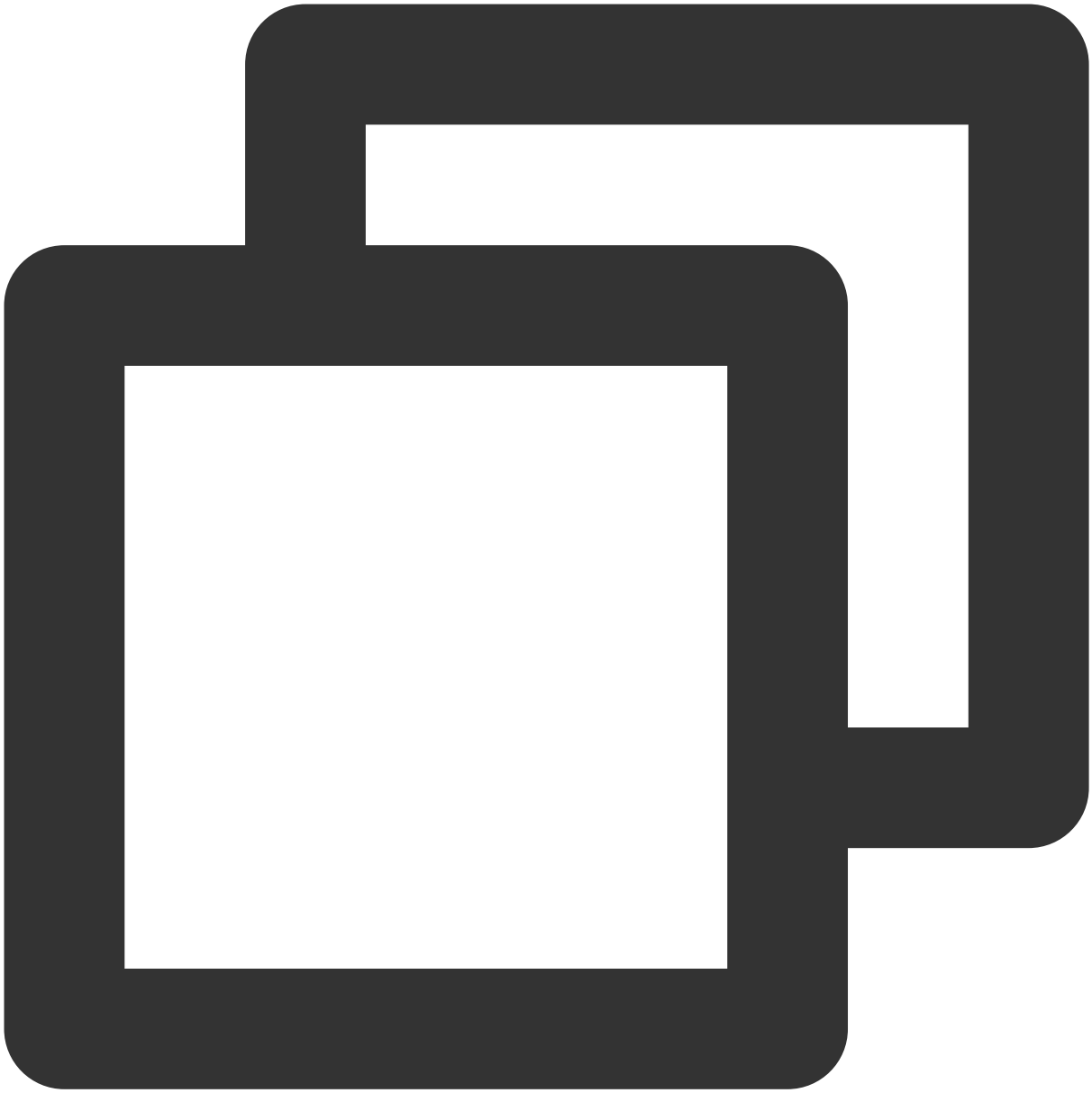
```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

The parameters are described below:

Parameter	Type	Description
mute	BOOL	Whether to mute or unmute audio. For more information, see muteLocalAudio().

setSpeaker

This API is used to set whether to play sound from the device's speaker or receiver.



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

The parameters are described below:

Parameter	Type	Description
useSpeaker	BOOL	`YES` : speaker; `NO` : receiver

setAudioCaptureVolume

This API is used to set the mic capturing volume.

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

The parameters are described below:

Parameter	Type	Description
volume	NSInteger	Capturing volume. Value range: 0-100. Default value: 100

setAudioPlayoutVolume

This API is used to set the playback volume.

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

The parameters are described below:

Parameter	Type	Description
volume	NSInteger	Playback volume. Value range: 0-100. Default value: 100

muteRemoteAudio

This API is used to mute/unmute a specified user.

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

The parameters are described below:

Parameter	Type	Description
userId	NSString	ID of the user to mute/unmute
mute	BOOL	`YES` : mute; `NO` : unmute

muteAllRemoteAudio

This API is used to mute/unmute all users.

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

The parameters are described below:

Parameter	Type	Description
mute	BOOL	`YES` : mute; `NO` : unmute

setVoiceEarMonitorEnable

This API is used to enable/disable in-ear monitoring.

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

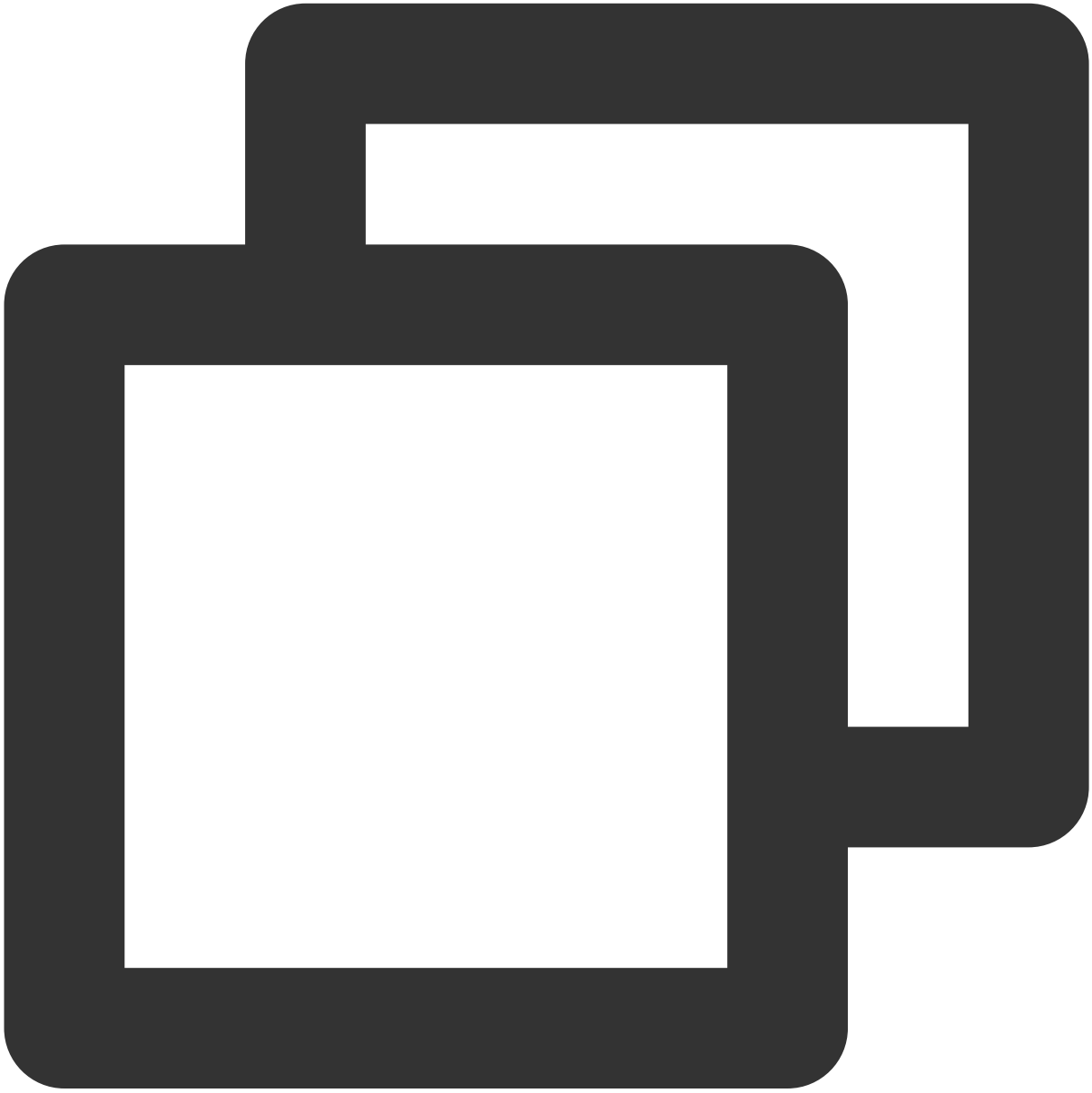
The parameters are described below:

Parameter	Type	Description
enable	BOOL	`YES` : enable in-ear monitoring; `NO` : disable in-ear monitoring

Background Music and Audio Effect APIs

getAudioEffectManager

This API is used to get the background music and audio effect management object TXAudioEffectManager.



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

Message Sending APIs

sendRoomTextMsg

This API is used to broadcast a text chat message in a room, which is generally used for on-screen comments.

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

The parameters are described below:

Parameter	Type	Description
message	NSString	Text message
callback	ActionCallback	Callback for the operation

sendRoomCustomMsg

This API is used to send a custom text message.

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(Act
```

The parameters are described below:

Parameter	Type	Description
cmd	NSString	Custom command word used to distinguish between different message types

message	NSString	Text message
callback	ActionCallback	Callback for the operation

Invitation Signaling APIs

sendInvitation

This API is used to send an invitation.

```
- (NSString *)sendInvitation:(NSString *)cmd
                      userId:(NSString *)userId
                     content:(NSString *)content
                    callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(sendI
```

The parameters are described below:

Parameter	Type	Description
cmd	NSString	Custom command of business
userId	NSString	ID of the user to invite
content	NSString	Invitation content
callback	ActionCallback	Callback for the operation

Response parameters:

Parameter	Type	Description
inviteId	NSString	Invitation ID

acceptInvitation

This API is used to accept an invitation.

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)
```

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID
callback	ActionCallback	Callback for the operation

rejectInvitation

This API is used to decline an invitation.

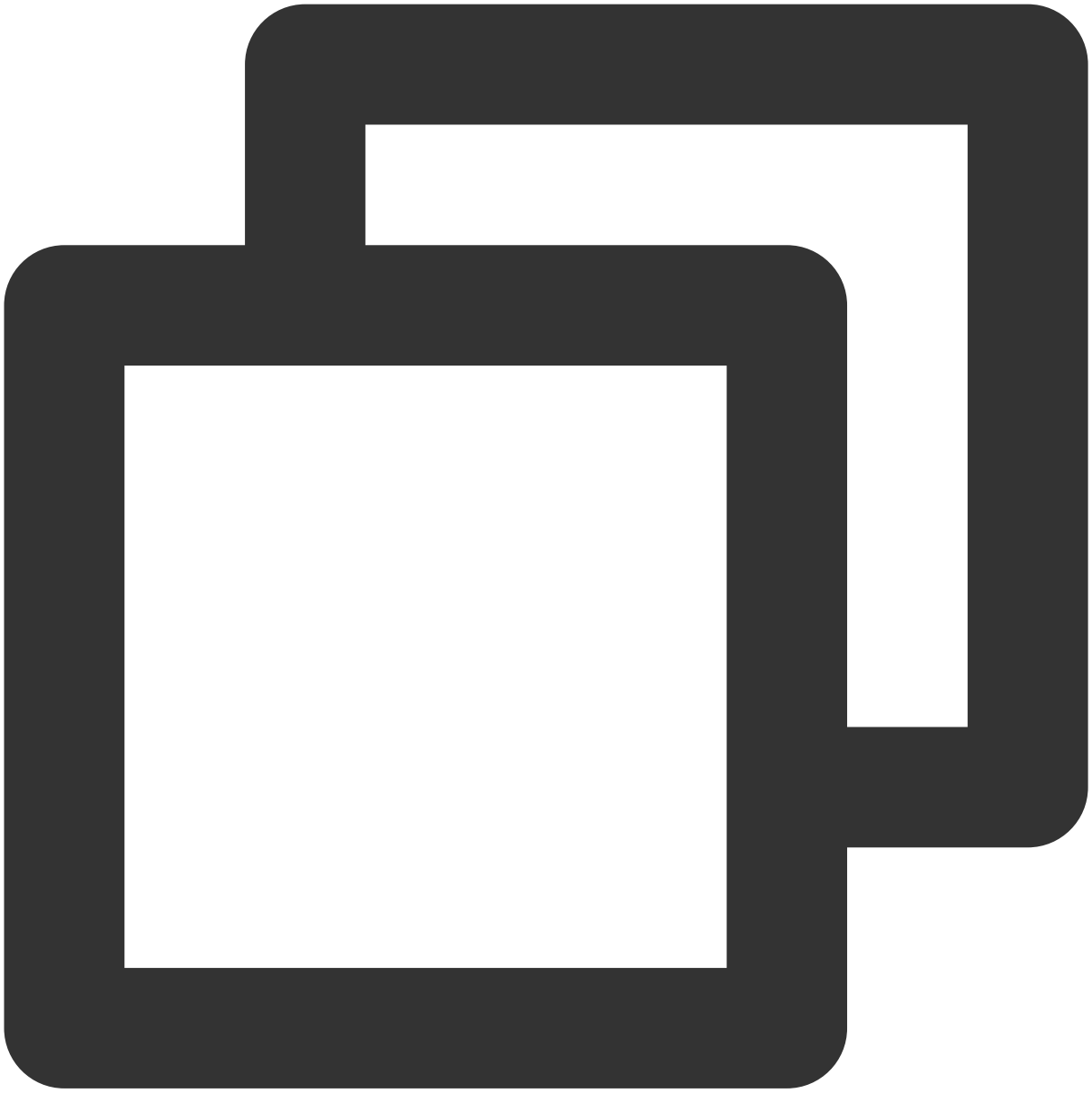
```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)
```

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID
callback	ActionCallback	Callback for the operation

cancelInvitation

This API is used to cancel an invitation.



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

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID
callback	ActionCallback	Callback for the operation

TRTCVoiceRoomDelegate Event Callback APIs

Common Event Callback APIs

onError

Callback for error.

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

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

The parameters are described below:

Parameter	Type	Description
code	int	Error code
message	NSString	Error message

onWarning

Callback for warning.

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

The parameters are described below:

Parameter	Type	Description

code	int	Error code
message	NSString	Warning message

onDebugLog

Callback for log.

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

The parameters are described below:

Parameter	Type	Description
message	NSString	Log information

Room Event Callback APIs

onRoomDestroy

Callback for room termination. When the owner terminates the room, all users in the room will receive this callback.

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

The parameters are described below:

Parameter	Type	Description
roomId	NSString	Room ID

onRoomInfoChange

Callback for change of room information. This callback is sent after successful room entry. The information in `roomInfo` is passed in by the room owner during room creation.

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

The parameters are described below:

Parameter	Type	Description
roomInfo	VoiceRoomInfo	Room information

onUserMicrophoneMute

Callback of whether a user's mic is muted. When a user calls `muteLocalAudio` , all members in the room will receive this callback.

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

The parameters are described below:

Parameter	Type	Description

userId	NSString	User ID
mute	BOOL	`YES` : muted; `NO` : unmuted

onUserVolumeUpdate

Notification to all members of the volume after the volume reminder is enabled.

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

The parameters are described below:

Parameter	Type	Description
userVolumes	NSArray	List of user volumes
totalVolume	NSInteger	Total volume. Value range: 0-100

Seat Callback APIs

onSeatListChange

Callback for all seat changes.

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

The parameters are described below:

Parameter	Type	Description
seatInfoList	NSArray<VoiceRoomSeatInfo>	Full seat list

onAnchorEnterSeat

Someone became a speaker or was made a speaker by the owner.

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

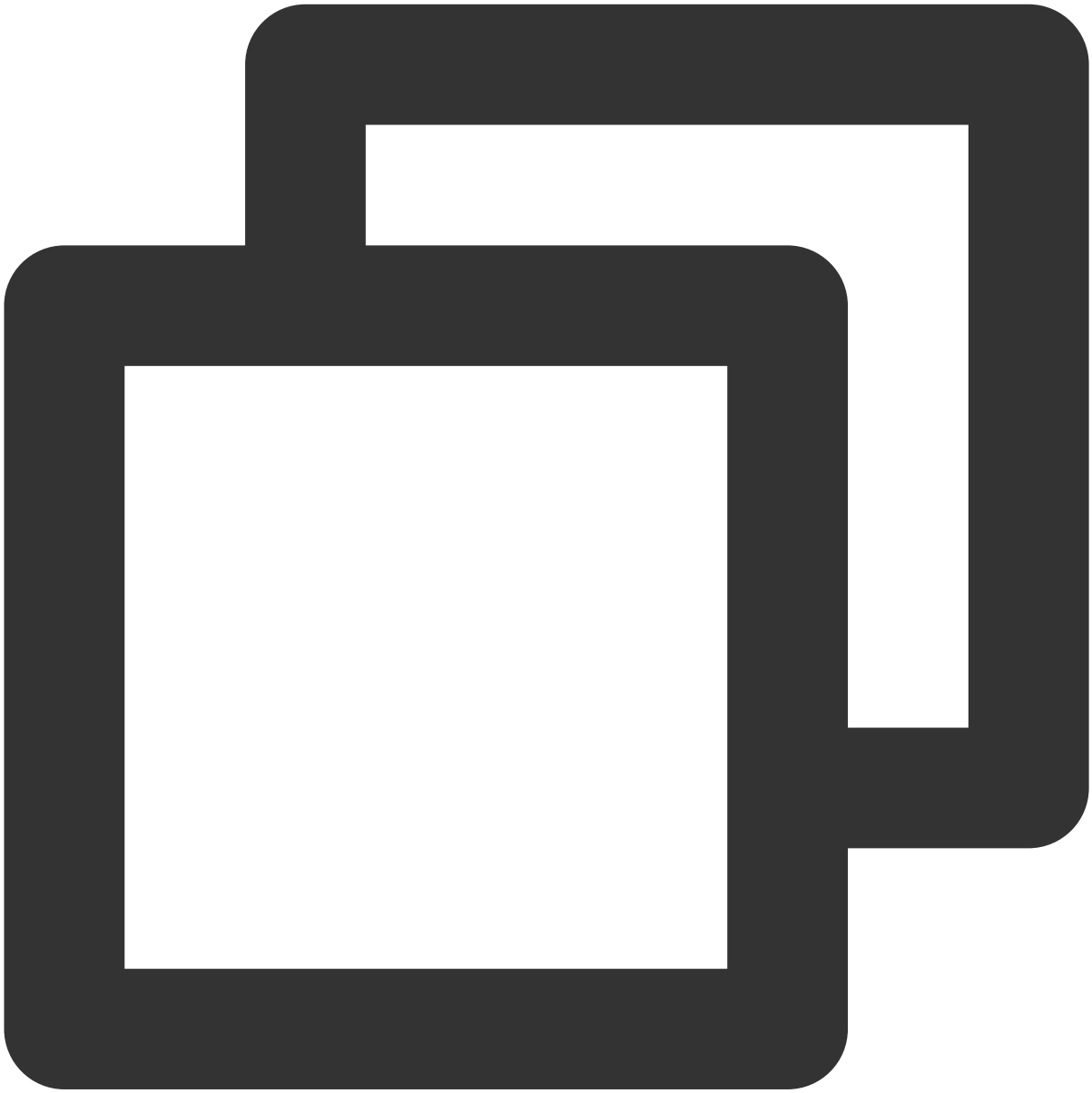
The parameters are described below:

Parameter	Type	Description
index	NSInteger	Number of the seat taken

user	VoiceRoomUserInfo	Information of the user who left the seat

onAnchorLeaveSeat

A speaker became a listener or was moved to listeners by the room owner.



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

The parameters are described below:

Parameter	Type	Description
index	NSInteger	Number of the seat the user left
user	VoiceRoomUserInfo	Information of the user who left the seat

onSeatMute

The room owner muted/unmuted a seat.

```
- (void)onSeatMute:(NSInteger)index
```

```
                isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

The parameters are described below:

Parameter	Type	Description
index	NSInteger	The seat muted/unmuted
isMute	BOOL	`YES: The seat was muted;` NO`: The seat was unmuted.

onSeatClose

The room owner blocked/unblocked a seat.

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

The parameters are described below:

Parameter	Type	Description
index	NSInteger	The seat blocked/unblocked
isClose	BOOL	`YES` : The seat was blocked; `NO` : The seat was unblocked.

Callback APIs for Room Entry/Exit by Listener

onAudienceEnter

A listener entered the room.

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

The parameters are described below:

Parameter	Type	Description
userInfo	VoiceRoomUserInfo	Information of the listener who entered

onAudienceExit

A listener exited the room.

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

The parameters are described below:

Parameter	Type	Description
userInfo	VoiceRoomUserInfo	Information of the user who left

Message Event Callback APIs

onRecvRoomTextMsg

Callback for receiving a text chat message.

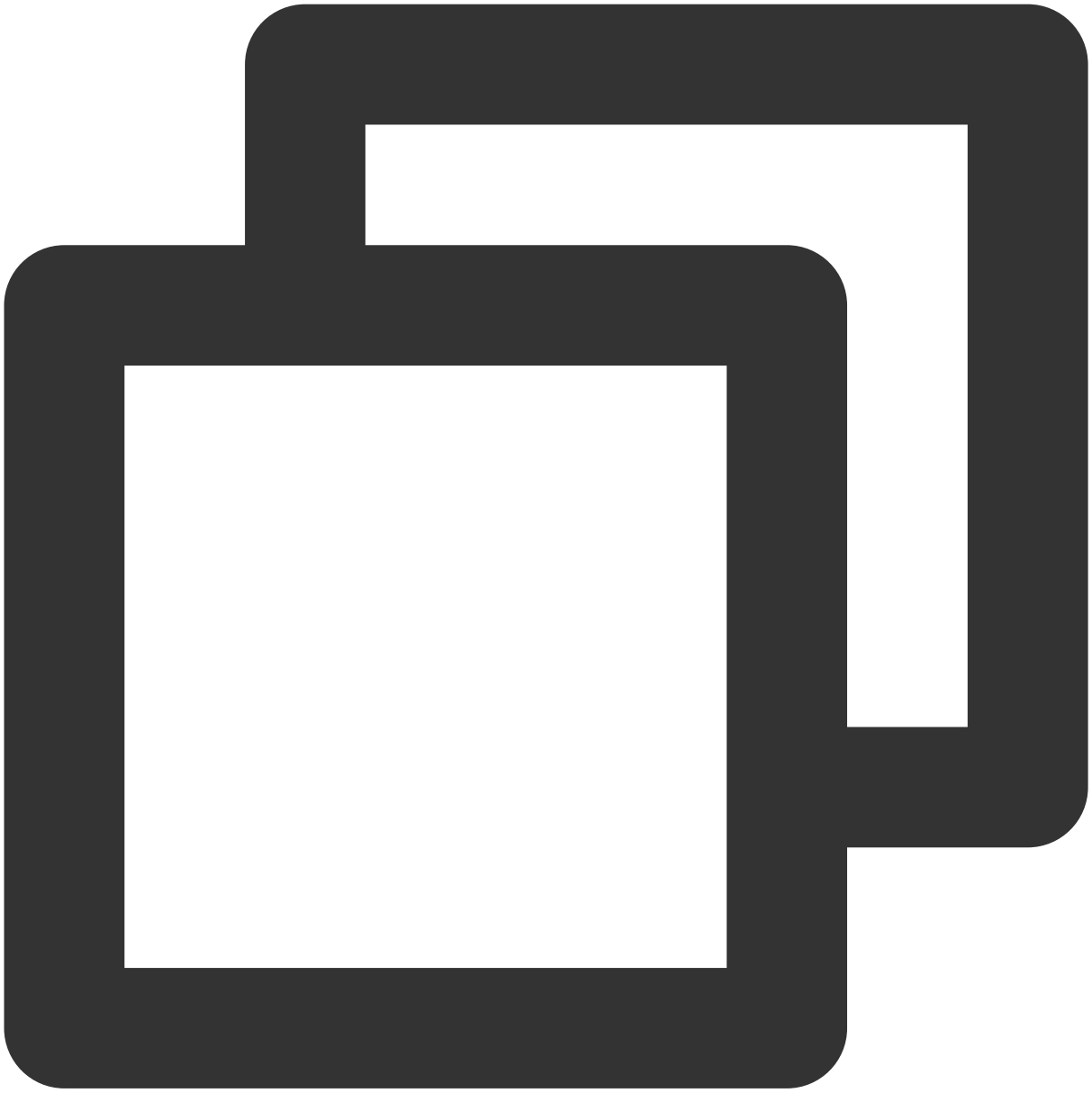
```
- (void)onRecvRoomTextMsg:(NSString *)message
                 userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

The parameters are described below:

Parameter	Type	Description
message	NSString	Text message
userInfo	VoiceRoomUserInfo	Information of the sender

onRecvRoomCustomMsg

A custom message was received.



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

The parameters are described below:

Parameter	Type	Description

command	NSString	Custom command word used to distinguish between different message types
message	NSString	Text message
userInfo	VoiceRoomUserInfo	Information of the sender

Invitation Signaling Callback APIs

onReceiveNewInvitation

An invitation was received.

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

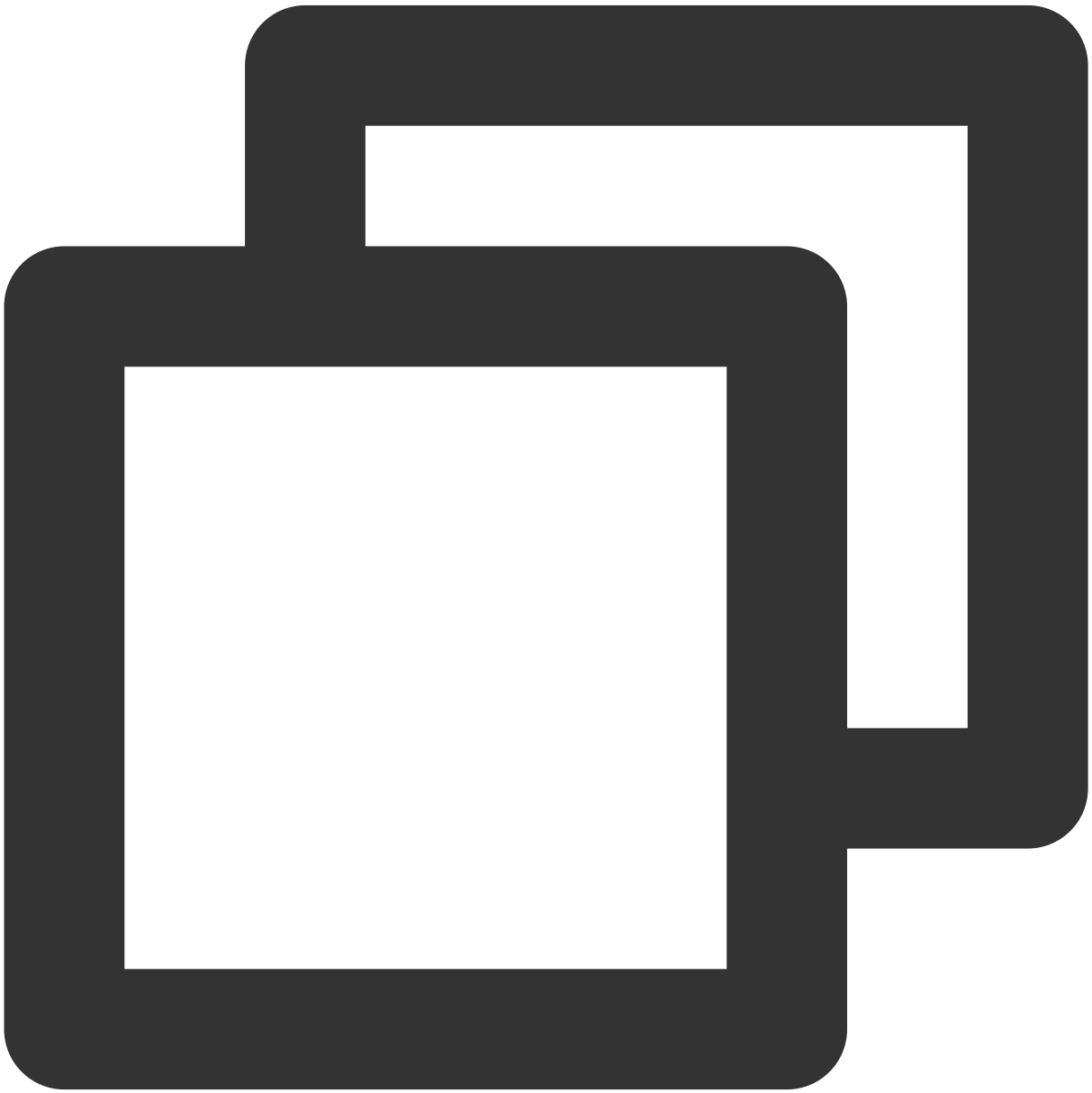
The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID

inviter	NSString	Inviter's user ID
cmd	NSString	Custom command word specified by business
content	NSString	Content specified by business

onInviteeAccepted

The invitee accepted the invitation



```
- (void)onInviteeAccepted:(NSString *)identifier
```

```
                    invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(id:invitee:));
```

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID
invitee	NSString	Invitee's user ID

onInviteeRejected

The invitee declined the invitation

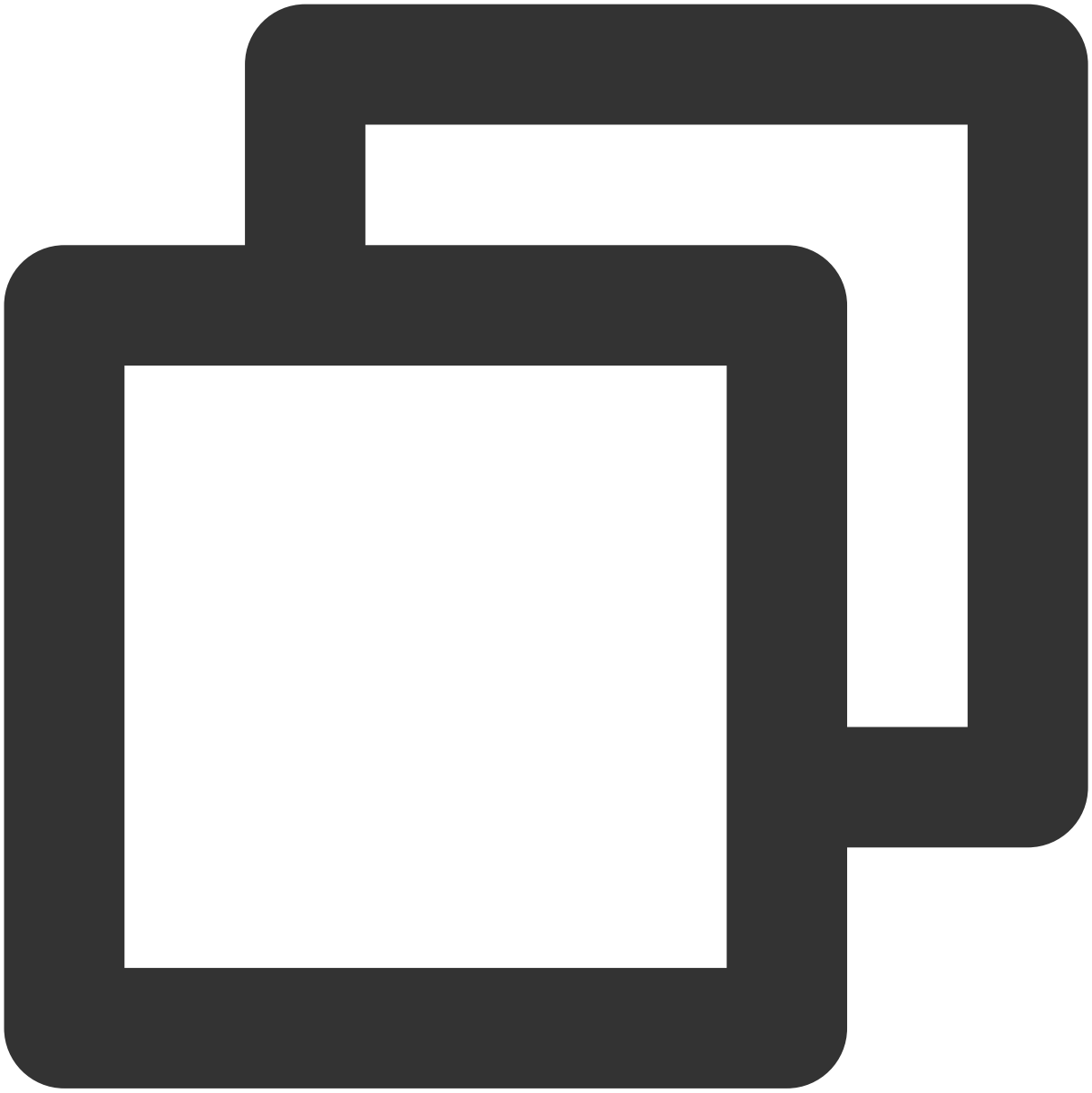
```
- (void)onInviteeRejected:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(id:invitee:));
```

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID
invitee	NSString	Invitee's user ID

onInvitationCancelled

The inviter canceled the invitation.



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

The parameters are described below:

Parameter	Type	Description
id	NSString	Invitation ID

inviter	NSString	Inviter's user ID

TRTCVoiceRoom (Android)

Last updated：2023-09-25 10:53:10

`TRTCVoiceRoom` is based on Tencent Real-Time Communication (TRTC) and Tencent Cloud Chat. With TRTCVoiceRoom:

A user can create an audio chat room and become a speaker, or enter an audio chat room as a listener.

The room owner can invite a listener to speak as well as remove a speaker from a seat.

The room owner can also block a seat. A listener cannot request to take a blocked seat to become a speaker.

A listener can request to speak and become a speaker. A speaker can also become a listener.

All users can send text and custom messages. Custom messages can be used to send on-screen comments, give likes, and send gifts.

Note

All TUIKit components are based on two PaaS services of Tencent Cloud, namely TRTC and Chat. When you activate TRTC, the Chat SDK trial edition (which supports up to 100 DAUs) will be activated automatically. For billing details of Chat, see Pricing.

`TRTCVoiceRoom` is an open-source class depending on two closed-source Tencent Cloud SDKs. For the specific implementation process, see Audio Chat Room (Android).

TRTThe TRTC SDK is used as a low-latency audio chat component.

The `AVChatRoom` feature of the Chat SDK is used to implement chat rooms. The attribute APIs of Chat are used to store room information such as the seat list, and invitation signaling is used to send requests to speak or invite others to speak.

TRTCVoiceRoom API Overview

Basic SDK APIs

API	Description
sharedInstance	Gets a singleton object.
destroySharedInstance	Terminates a singleton object.
setDelegate	Sets event callbacks.
setDelegateHandler	Sets the thread where event callbacks are.
login	Logs in.

logout	Logs out.
setSelfProfile	Sets profile.

Room APIs

API	Description
createRoom	Creates a room (called by room owner). If the room does not exist, the system will automatically create a room.
destroyRoom	Terminates a room (called by room owner).
enterRoom	Enters a room (called by listener).
exitRoom	Exits a room (called by listener).
getRoomInfoList	Gets room list details.
getUserInfoList	Gets the user information of the specified `userId`. If the value is `null`, the information of all users in the room is obtained.

Seat management APIs

API	Description
enterSeat	Becomes a speaker (called by room owner or listener).
moveSeat	Changes the seat (called by speaker).
leaveSeat	Becomes a listener (called by speaker).
pickSeat	Places a user in a seat (called by room owner).
kickSeat	Removes a speaker (called by room owner).
muteSeat	Mutes/Unmutes a seat (called by room owner).
closeSeat	Blocks/Unblocks a seat (called by room owner).

Local audio APIs

API	Description
startMicrophone	Starts mic capturing.
stopMicrophone	Stops mic capturing.

setAudioQuality	Sets audio quality.
muteLocalAudio	Mutes/Unmutes local audio.
setSpeaker	Sets whether to play sound from the device's speaker or receiver.
setAudioCaptureVolume	Sets mic capturing volume.
setAudioPlayoutVolume	Sets playback volume.
setVoiceEarMonitorEnable	Enables/Disables in-ear monitoring.

Remote audio APIs

API	Description
muteRemoteAudio	Mutes/Unmutes a specified member.
muteAllRemoteAudio	Mutes/Unmutes all members.

Background music and audio effect APIs

API	Description
getAudioEffectManager	Gets the background music and audio effect management object TXAudioEffectManager.

Message sending APIs

API	Description
sendRoomTextMsg	Broadcasts a text chat message in a room. This API is generally used for on-screen comments.
sendRoomCustomMsg	Sends a custom text message.

Invitation signaling APIs

API	Description
sendInvitation	Sends an invitation.
acceptInvitation	Accepts an invitation.
rejectInvitation	Declines an invitation.

cancelInvitation	Cancels an invitation.

TRTCVoiceRoomDelegate API Overview

Common event callbacks

API	Description
onError	Callback for error.
onWarning	Callback for warning.
onDebugLog	Callback of log.

Room event callback APIs

API	Description
onRoomDestroy	The room was terminated.
onRoomInfoChange	The room information changed.
onUserVolumeUpdate	User volume

Seat list change callback APIs

API	Description
onSeatListChange	All seat changes
onAnchorEnterSeat	Someone became a speaker or was made a speaker by the room owner.
onAnchorLeaveSeat	Someone became a listener or was moved to listeners by the room owner.
onSeatMute	The room owner muted a seat.
onUserMicrophoneMute	Whether a user's mic is muted
onSeatClose	The room owner blocked a seat.

Callback APIs for room entry/exit by listener

API	Description
onAudienceEnter	A listener entered the room.
onAudienceExit	A listener exited the room.

Message event callback APIs

API	Description
onRecvRoomTextMsg	A text chat message was received.
onRecvRoomCustomMsg	A custom message was received.

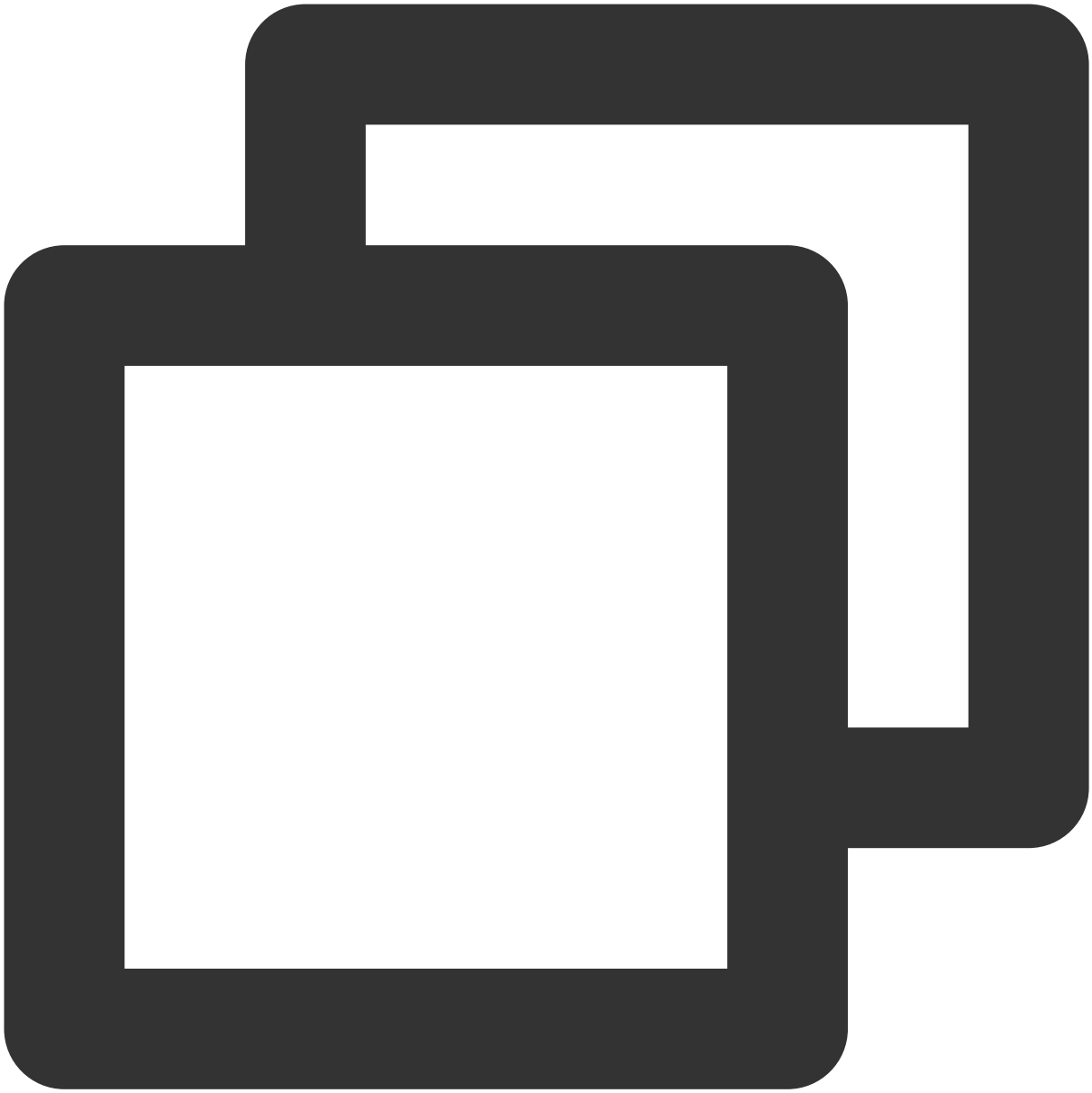
Signaling Event Callback APIs

API	Description
onReceiveNewInvitation	An invitation was received.
onInviteeAccepted	The invitee accepted the invitation.
onInviteeRejected	The invitee declined the invitation.
onInvitationCancelled	The inviter canceled the invitation.

Basic SDK APIs

sharedInstance

This API is used to get a TRTCVoiceRoom singleton object.



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

The parameters are described below:

Parameter	Type	Description
context	Context	Android context, which will be converted to `ApplicationContext` for the calling of system APIs

destroySharedInstance

This API is used to terminate a TRTCVoiceRoom singleton object.

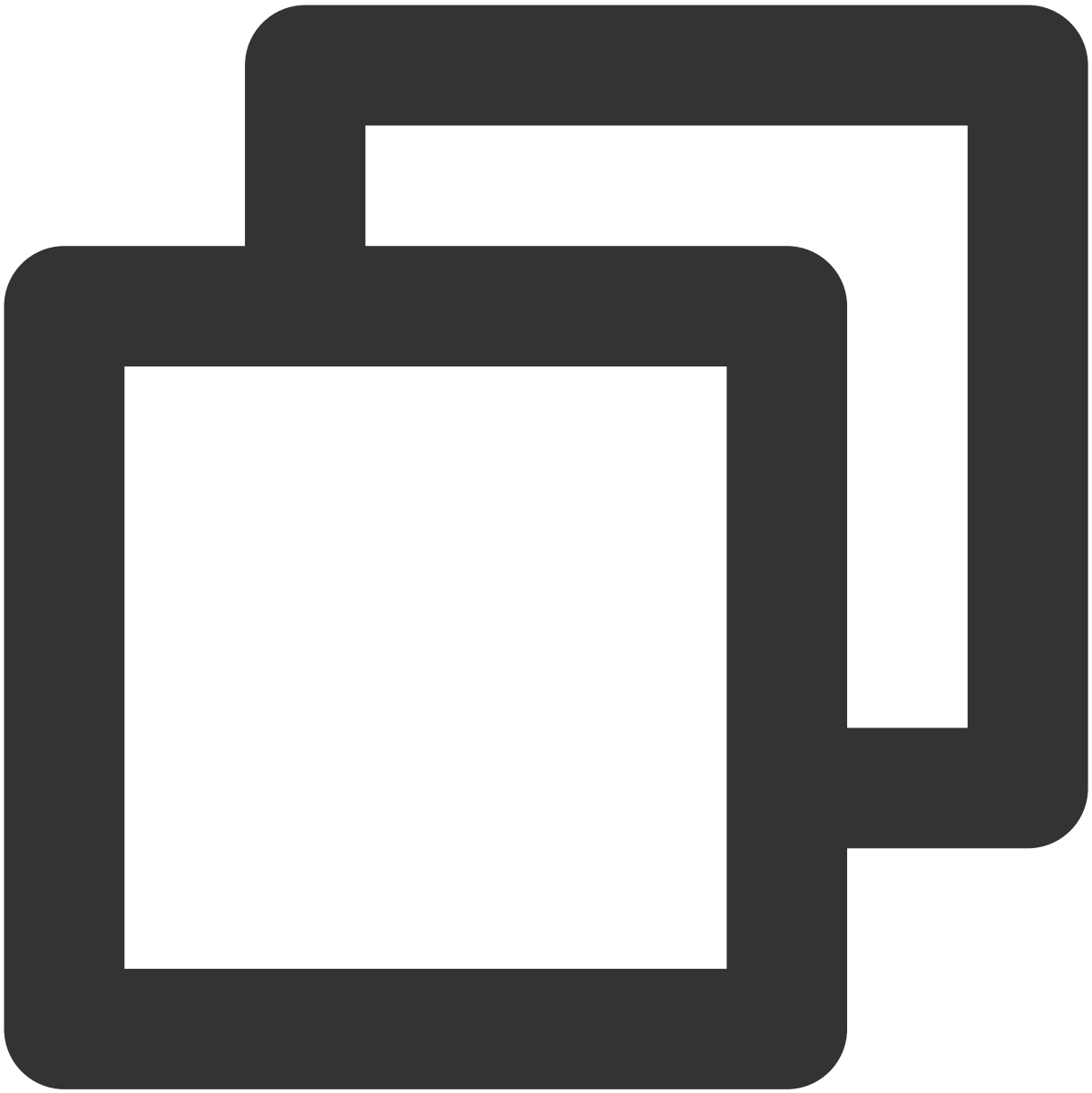
Note

After the instance is terminated, the externally cached `TRTCVoiceRoom` instance can no longer be used. You need to call sharedInstance again to get a new instance.

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

setDelegate

This API is used to set the event callback of TRTCVoiceRoom. You can use `TRTCVoiceRoomDelegate` to get different status notifications of TRTCVoiceRoom.



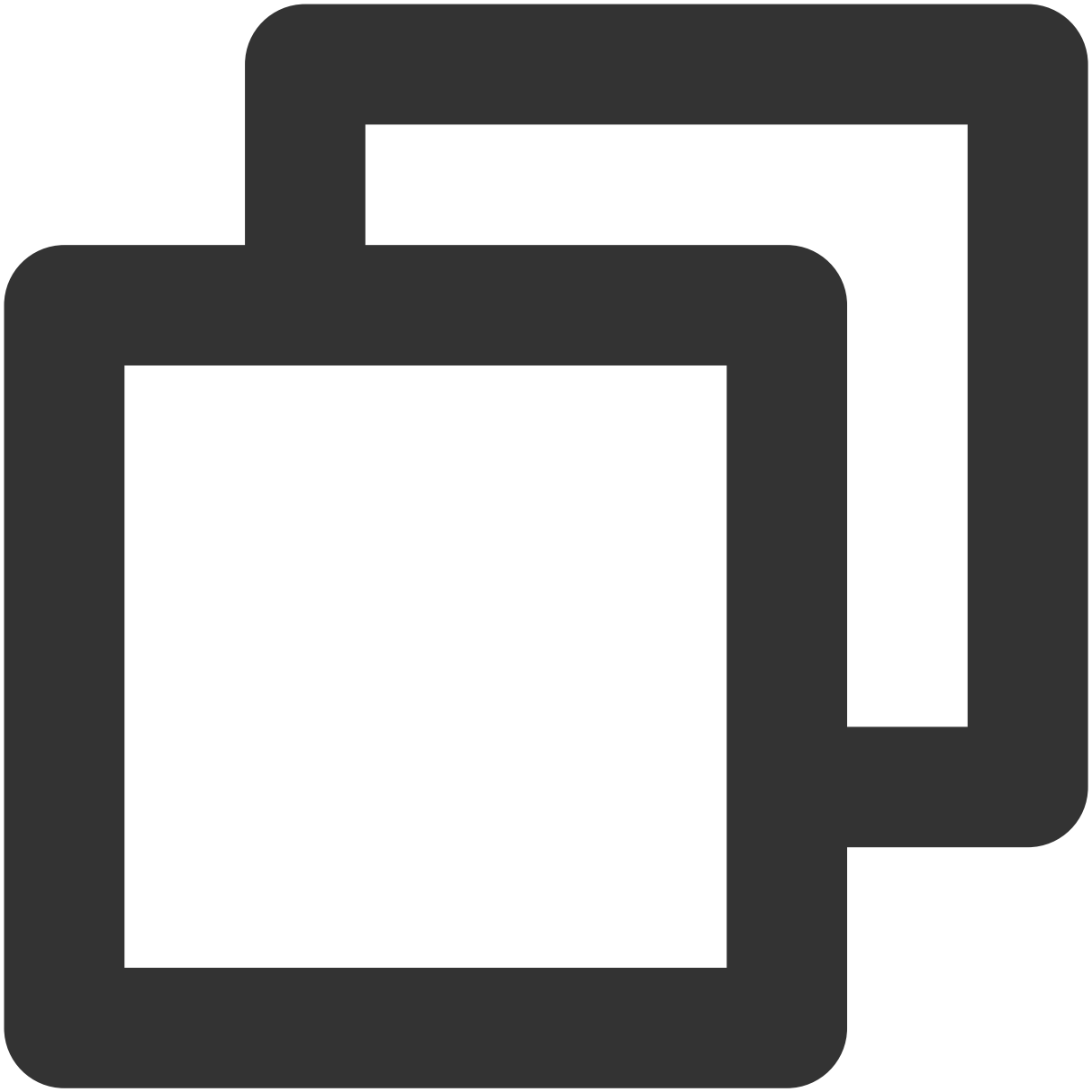
```
public abstract void setDelegate(TRTCVoiceRoomDelegate delegate);
```

Note

`setDelegate` is the delegate callback of `TRTCVoiceRoom`.

setDelegateHandler

This API is used to set the thread where event callbacks are.



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

The parameters are described below:

Parameter	Type	Description
handler	Handler	The status notifications of `TRTCVoiceRoom` are sent to the handler thread you specify.

login

Login

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

The parameters are described below:

Parameter	Type	Description
sdkAppId	int	You can view `SDKAppID` in Application Management > **Application Info** of the TRTC console.

userId	String	The ID of current user, which is a string that can contain only letters (a-z and A-Z), digits (0-9), hyphens (-), and underscores (_).
userSig	String	Tencent Cloud's proprietary security signature. For how to calculate and use it, see FAQs > UserSig.
callback	ActionCallback	The callback for login. The code is `0` if login succeeds.

logout

Log out

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

The parameters are described below:

Parameter	Type	Description
callback	ActionCallback	The callback for logout. The code is `0` if logout succeeds.

setSelfProfile

This API is used to set the profile.

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCVoiceRoo
```

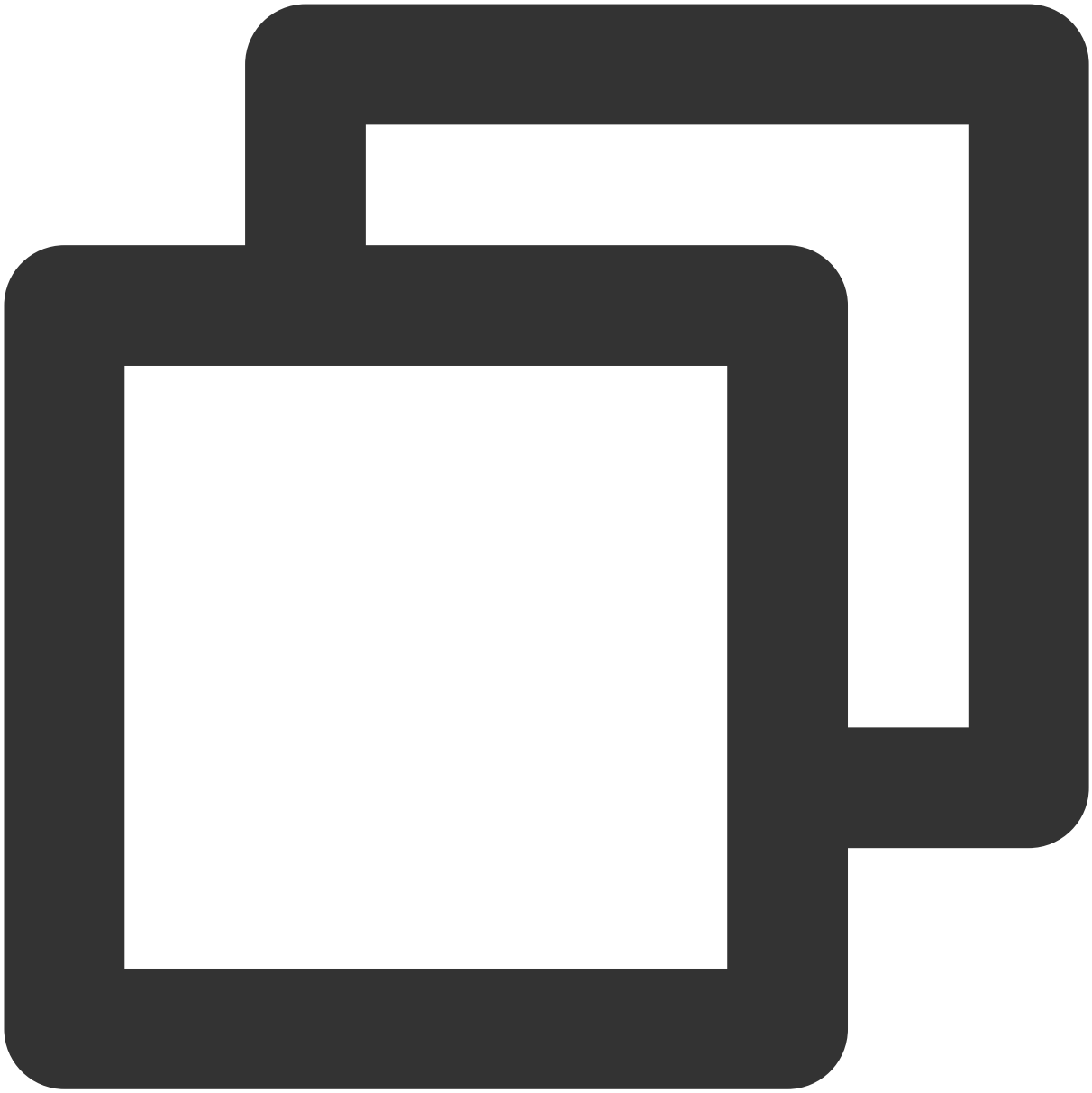
The parameters are described below:

Parameter	Type	Description
userName	String	The username.
avatar	String	The address of the profile photo.
callback	ActionCallback	Callback for profile setting. The code is `0` if the operation succeeds.

Room APIs

createRoom

This API is used to create a room (called by room owner).



```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam, T
```

The parameters are described below:

Parameter	Type	Description
roomId	int	The room ID. You need to assign and manage the IDs in a centralized manner. Multiple `roomID` values can be aggregated into a karaoke room list. Currently, Tencent Cloud does not provide management services for room lists. Please manage your own room lists.

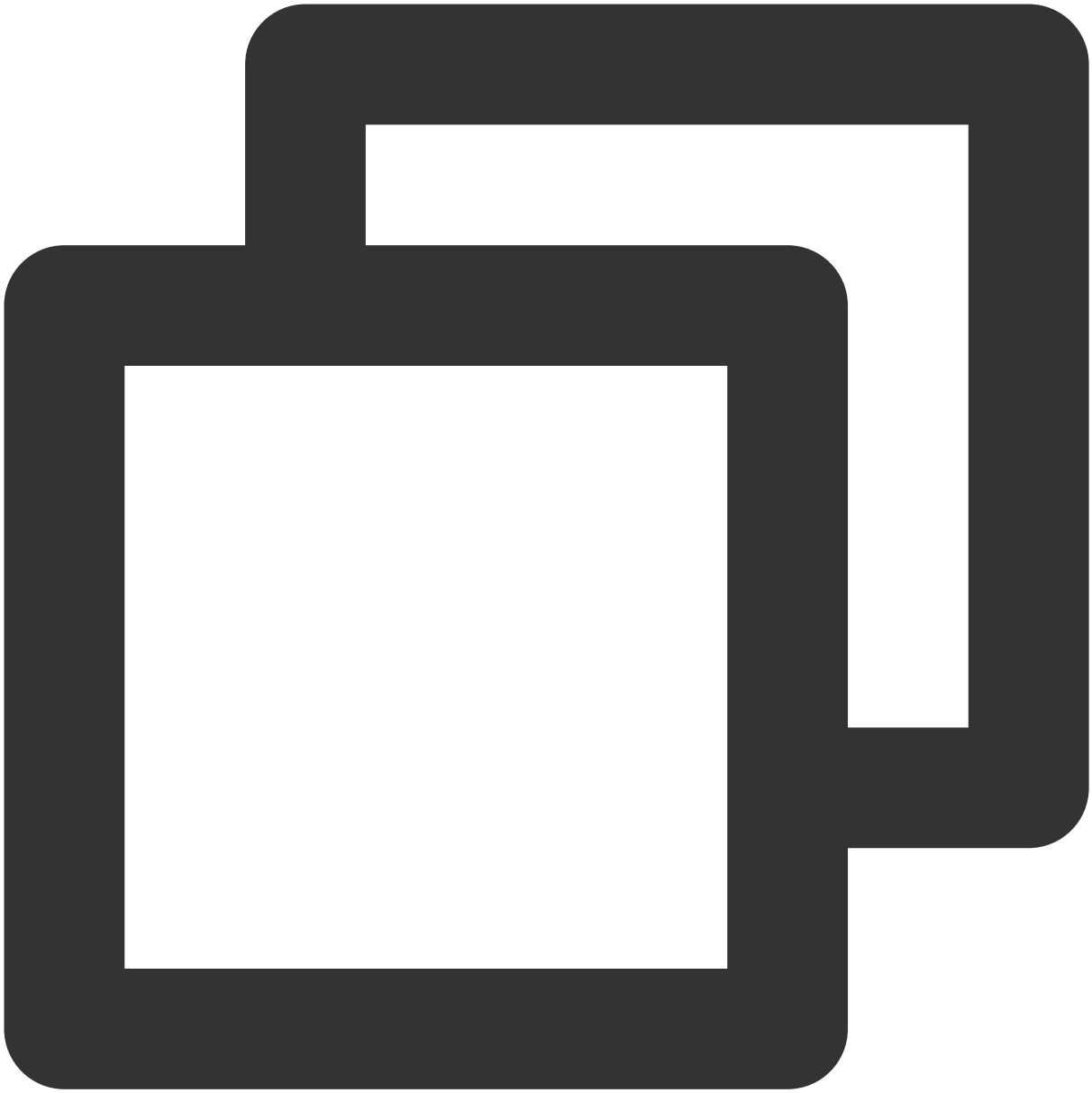
roomParam	TRTCCreateRoomParam	Room information, such as room name, seat list information, and cover information. To manage seats, you must enter the number of seats in the room.
callback	ActionCallback	Callback for room creation. The code is 0 if the operation succeeds.

The process of creating a karaoke room and becoming a speaker is as follows:

1. A user calls `createRoom` to create an audio chat room, passing in room attributes (e.g. room ID, whether listeners require room owner's consent to speak, number of seats).
2. After creating the room, the user calls `enterSeat` to become a speaker.
3. The user will receive an `onSeatListChanget` notification about the change of the seat list, and can update the change to the UI.
4. The user will also receive an `onAnchorEnterSeat` notification that someone became a speaker, and mic capturing will be enabled automatically.

destroyRoom

This API is used to terminate a room (called by room owner).



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

The parameters are described below:

Parameter	Type	Description
callback	ActionCallback	Callback for room termination. The code is `0` if the operation succeeds.

enterRoom

This API is used to enter a room (called by listener).

```
public abstract void enterRoom(int roomId, TRTCVoiceRoomCallback.ActionCallback cal
```

The parameters are described below:

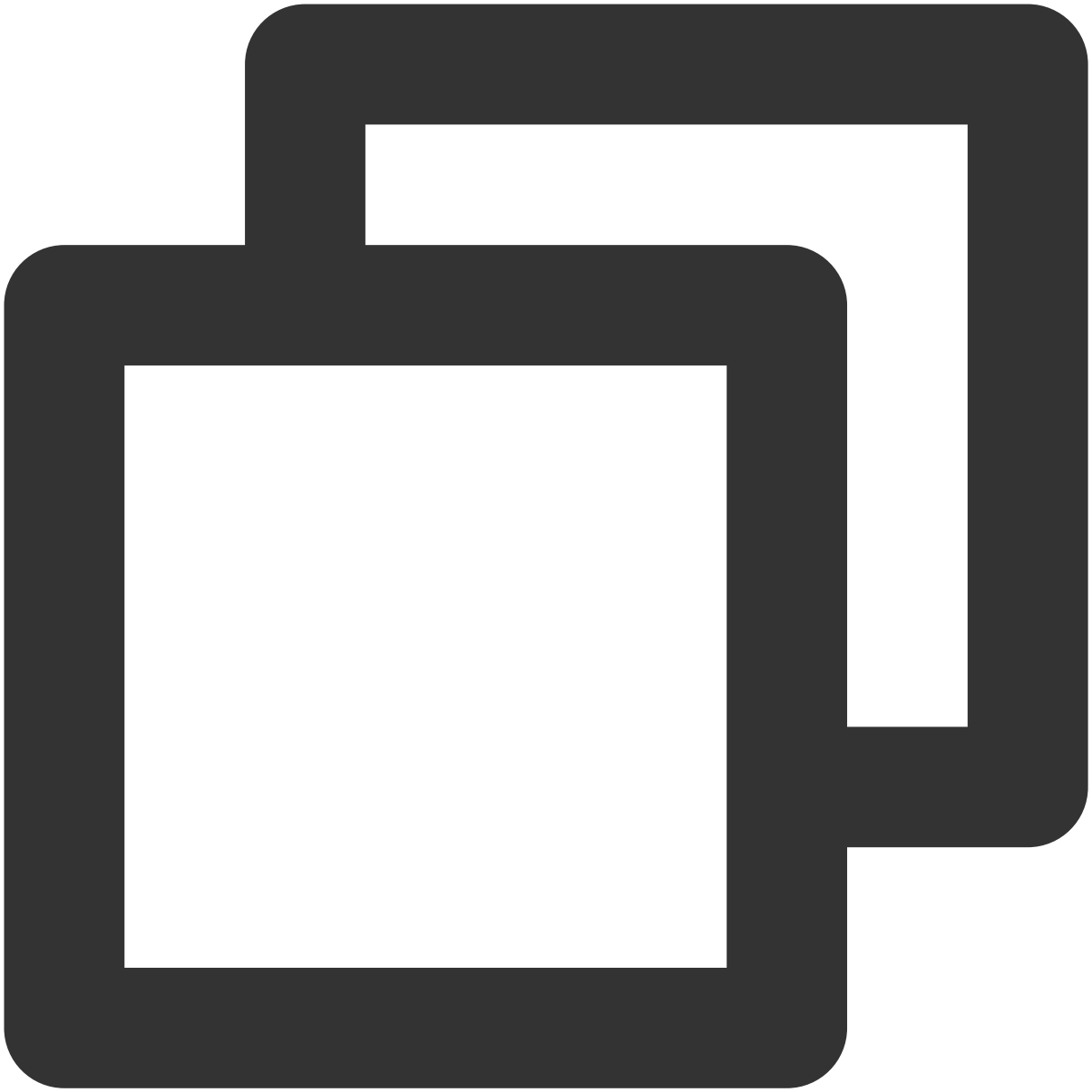
Parameter	Type	Description
roomId	int	The room ID.
callback	ActionCallback	Callback for room entry. The code is `0` if the operation succeeds.

The process of entering a room as a listener is as follows:

1. A user gets the latest audio chat room list from your server. The list may contain the `roomId` and room information of multiple audio chat rooms.
2. The user selects a room, and calls `enterRoom` with the room ID passed in to enter the room.
3. After entering the room, the user receives an `onRoomInfoChange` notification about room attribute change from the component. The attributes can be recorded, and corresponding changes can be made to the UI, including room name, whether room owner's consent is required for listeners to speak, etc.
4. The user will receive an `onSeatListChange` notification about the change of the seat list and can update the change to the UI.
5. The user will also receive an `onAnchorEnterSeat` notification that someone became a speaker.

exitRoom

Leave room



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

The parameters are described below:

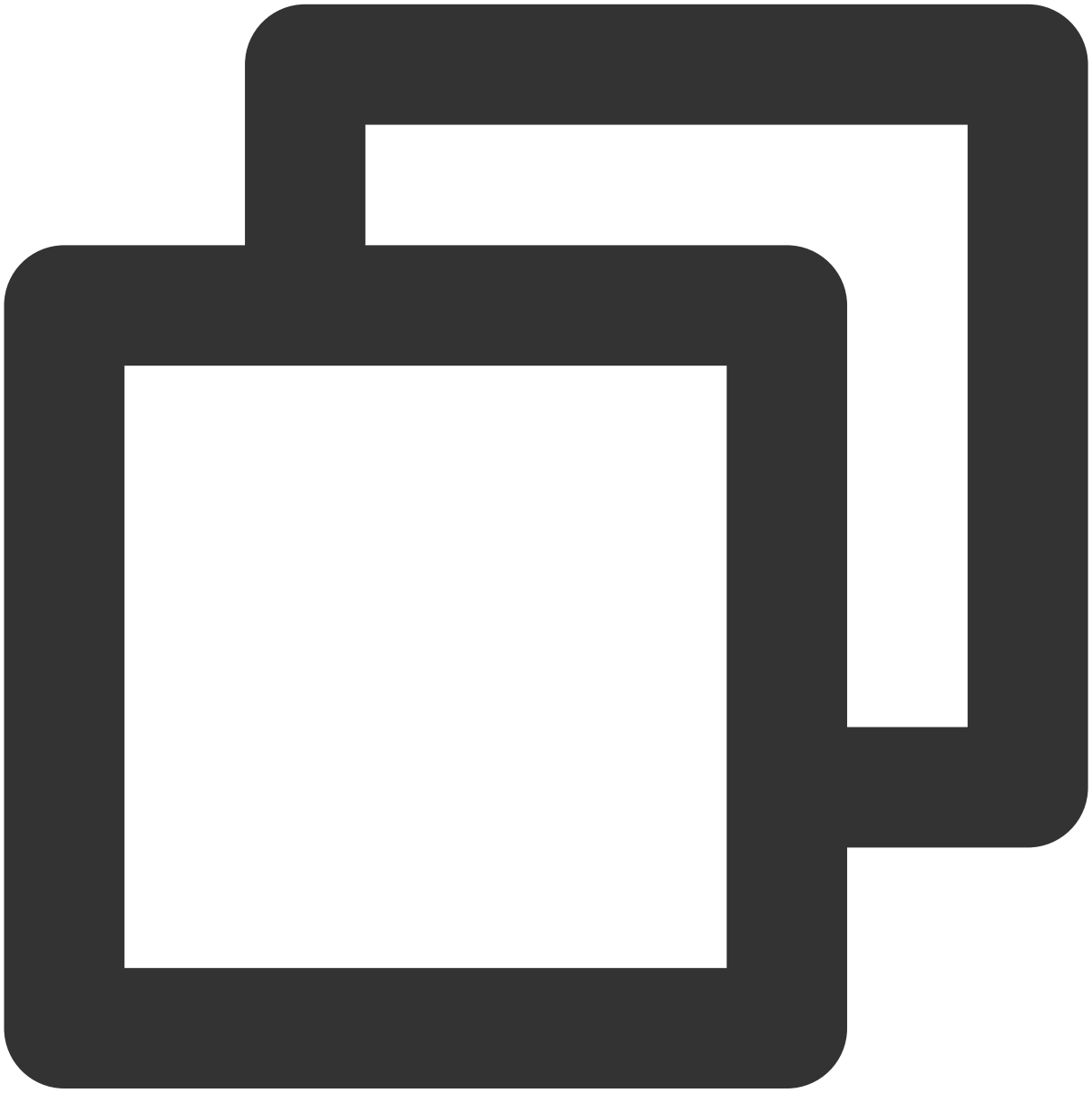
Parameter	Type	Description
callback	ActionCallback	Callback for room exit. The code is `0` if the operation succeeds.

getRoomInfoList

This API is used to get room list details. The room name and cover are set by the room owner via `roomInfo` when calling `createRoom()` .

Note

You don't need this API if both the room list and room information are managed on your server.



```
public abstract void getRoomInfoList(List<Integer> roomIdList, TRTCVoiceRoomCallbac
```

The parameters are described below:

Parameter	Type	Description

roomIdList	List<Integer>	Room ID list
callback	RoomInfoCallback	Callback of room details

getUserInfoList

This API is used to get the user information of a specified `userId` .

```
public abstract void getUserInfoList(List<String> userIdList, TRTCVoiceRoomCallback
```

The parameters are described below:

Parameter	Type	Description
userIdList	List<String>	IDs of the users to query. If this parameter is `null` , the information of all users in the room is queried.
userlistcallback	UserListCallback	Callback of user details

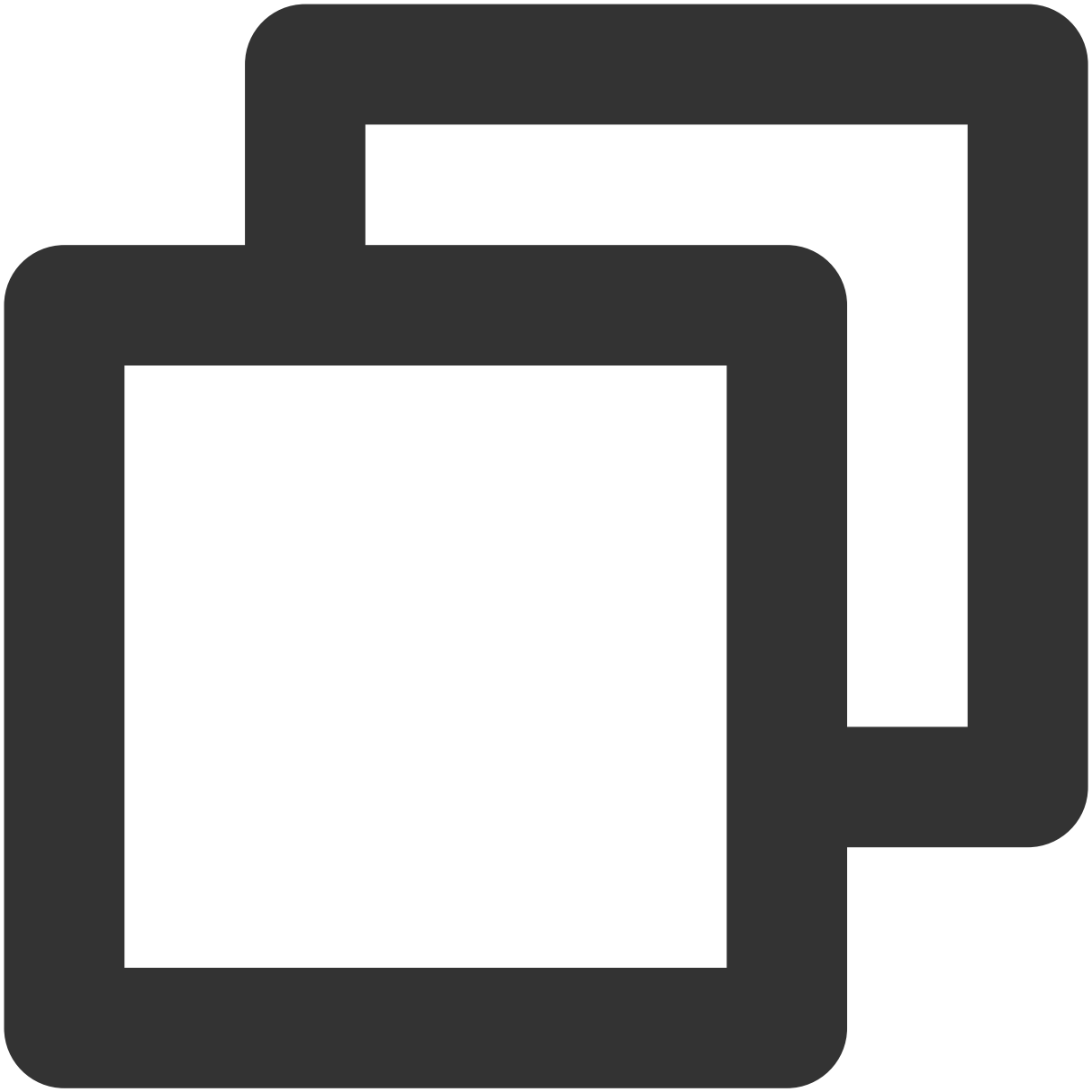
Seat Management APIs

enterSeat

This API is used to become a speaker (called by room owner or listener).

Note

After a user becomes a speaker, all members in the room will receive an `onSeatListChange` notification and an `onAnchorEnterSeat` notification.



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

The parameters are described below:

Parameter	Type	Description
seatIndex	int	The number of the seat to take
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. In cases where listeners need the room owner's consent to take a seat, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept`, call this API.

moveSeat

This API is used to change one's seat (called by speaker).

Note

After the seat change, all users in the room will receive the `onSeatListChange`, `onAnchorLeaveSeat`, and `onAnchorEnterSeat` notifications. This API will only change the user's seat number, not the user role.

```
public abstract int moveSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback ca
```

The parameters are described below:

Parameter	Type	Description
seatIndex	int	The number of the seat to change to
callback	ActionCallback	Callback for the operation

Response parameters:

Parameter	Type	Description
code	int	Result of seat change. `0` : operation successful; `10001` : API rate limit exceeded; other values: operation failed

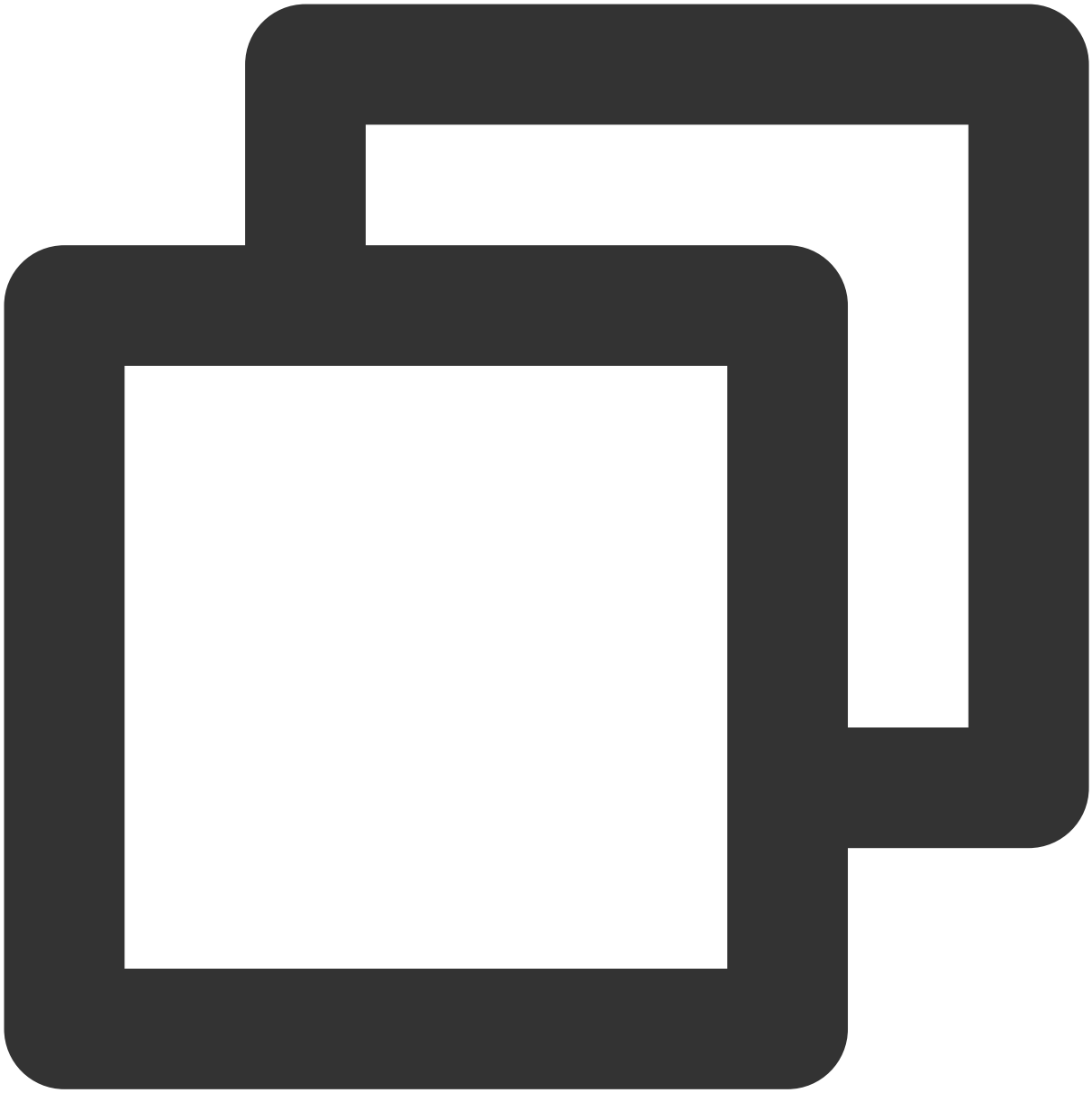
Calling this API will immediately modify the seat list. In cases where listeners need the room owner's consent to take a seat, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept` , call this API.

leaveSeat

This API is used to become a listener (called by speaker).

Note

After a speaker becomes a listener, all members in the room will receive an `onSeatListChange` notification and an `onAnchorLeaveSeat` notification.



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

The parameters are described below:

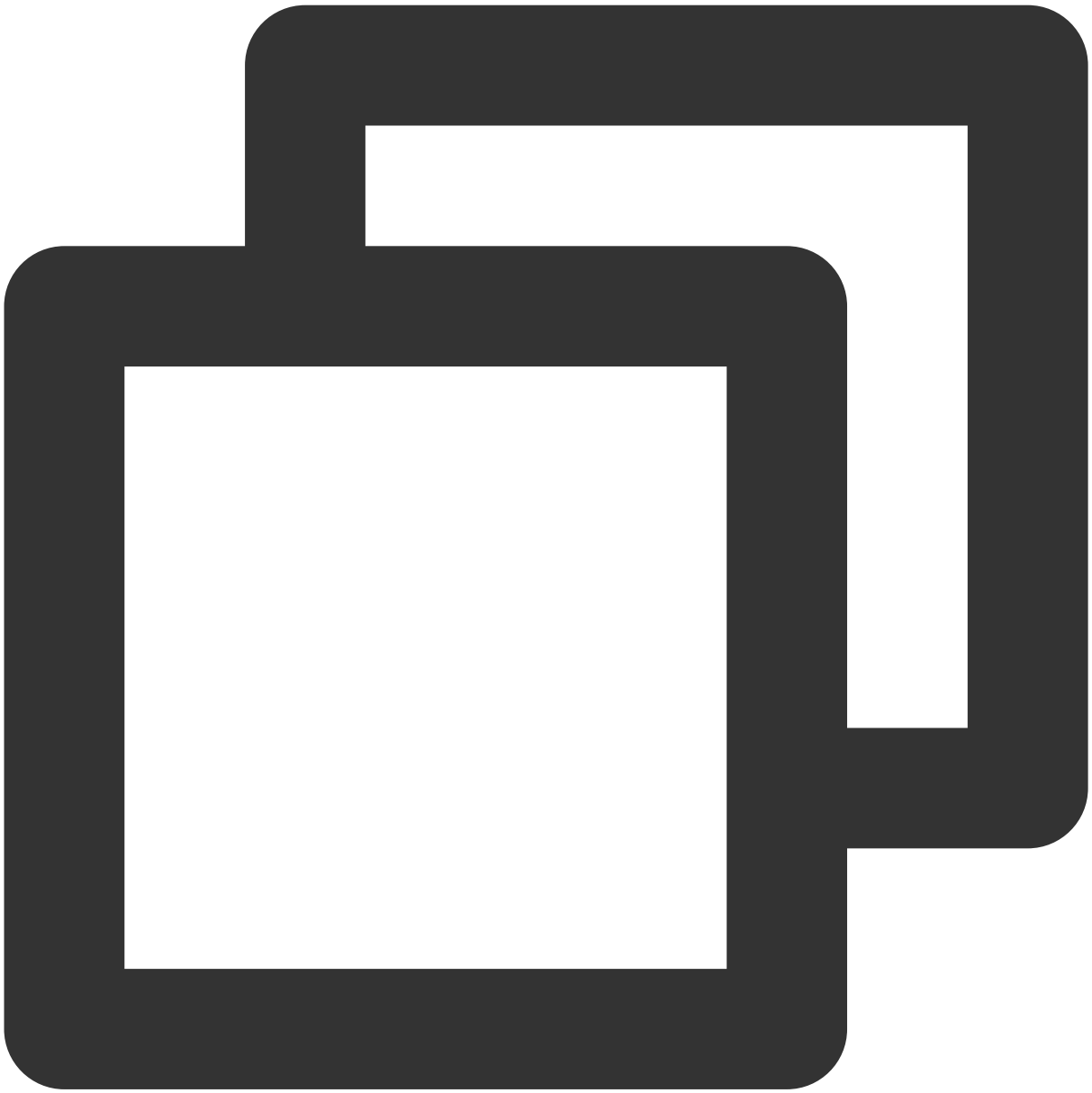
Parameter	Type	Description
callback	ActionCallback	Callback for the operation

pickSeat

This API is used to place a user in a seat (called by room owner).

Note

After the room owner places a user in a seat, all members in the room will receive an `onSeatListChange` notification and an `onAnchorEnterSeat` notification.



```
public abstract void pickSeat(int seatIndex, String userId, TRTCVoiceRoomCallback.A
```

The parameters are described below:

Parameter	Type	Description
seatIndex	int	The number of the seat to place the listener in

userId	String	User ID
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. In cases where the room owner needs listeners' consent to make them speakers, you can call `sendInvitation` first to send a request and, after receiving `onInvitationAccept`, call `pickSeat`.

kickSeat

This API is used to remove a speaker (called by room owner).

Note

After a speaker is removed, all members in the room will receive an `onSeatListChange` notification and an `onAnchorLeaveSeat` notification.

```
public abstract void kickSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback c
```

The parameters are described below:

Parameter	Type	Description
seatIndex	int	The number of the seat to remove the speaker from
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list.

muteSeat

This API is used to mute/unmute a seat (called by room owner).

Note

After a seat is muted/unmuted, all members in the room will receive an `onSeatListChange` notification and an `onSeatMute` notification.

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCVoiceRoomCallback.
```

The parameters are described below:

Parameter	Type	Description
seatIndex	int	The number of the seat to block/unblock
isMute	boolean	`true` : mute; `false` : unmute
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. The speaker on the seat specified by `seatIndex` will call `muteAudio` to mute/unmute his or her audio.

closeSeat

This API is used to block/unblock a seat (called by room owner).

Note

After a seat is blocked/unblocked, all members in the room will receive an `onSeatListChange` notification and an `onSeatClose` notification.

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCVoiceRoomCallbac
```

The parameters are described below:

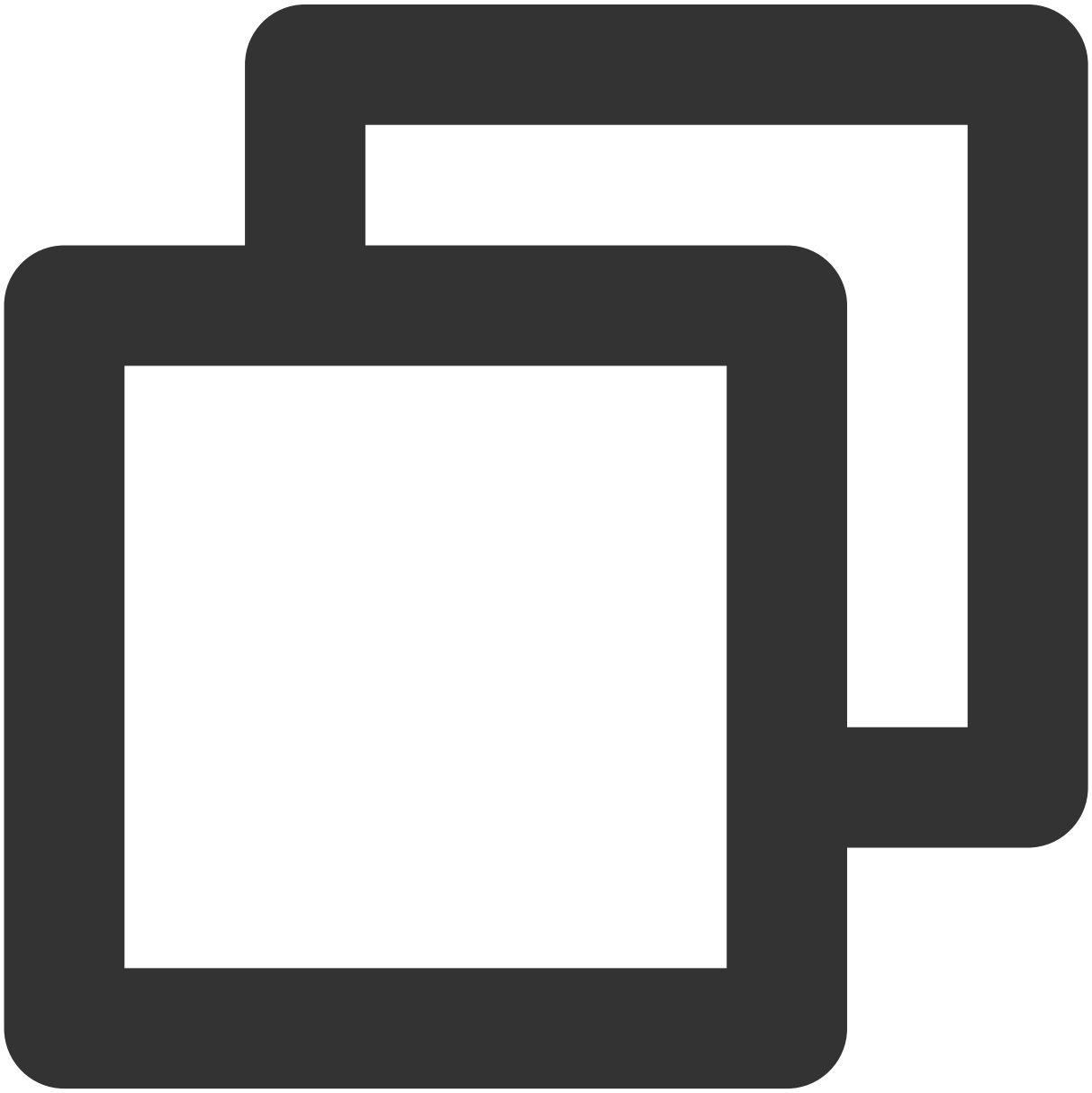
Parameter	Type	Description
seatIndex	int	The number of the seat to block/unblock
isClose	boolean	`true` : block; `false` : unblock
callback	ActionCallback	Callback for the operation

Calling this API will immediately modify the seat list. The speaker on the seat specified by `seatIndex` will leave the seat.

Local Audio APIs

startMicrophone

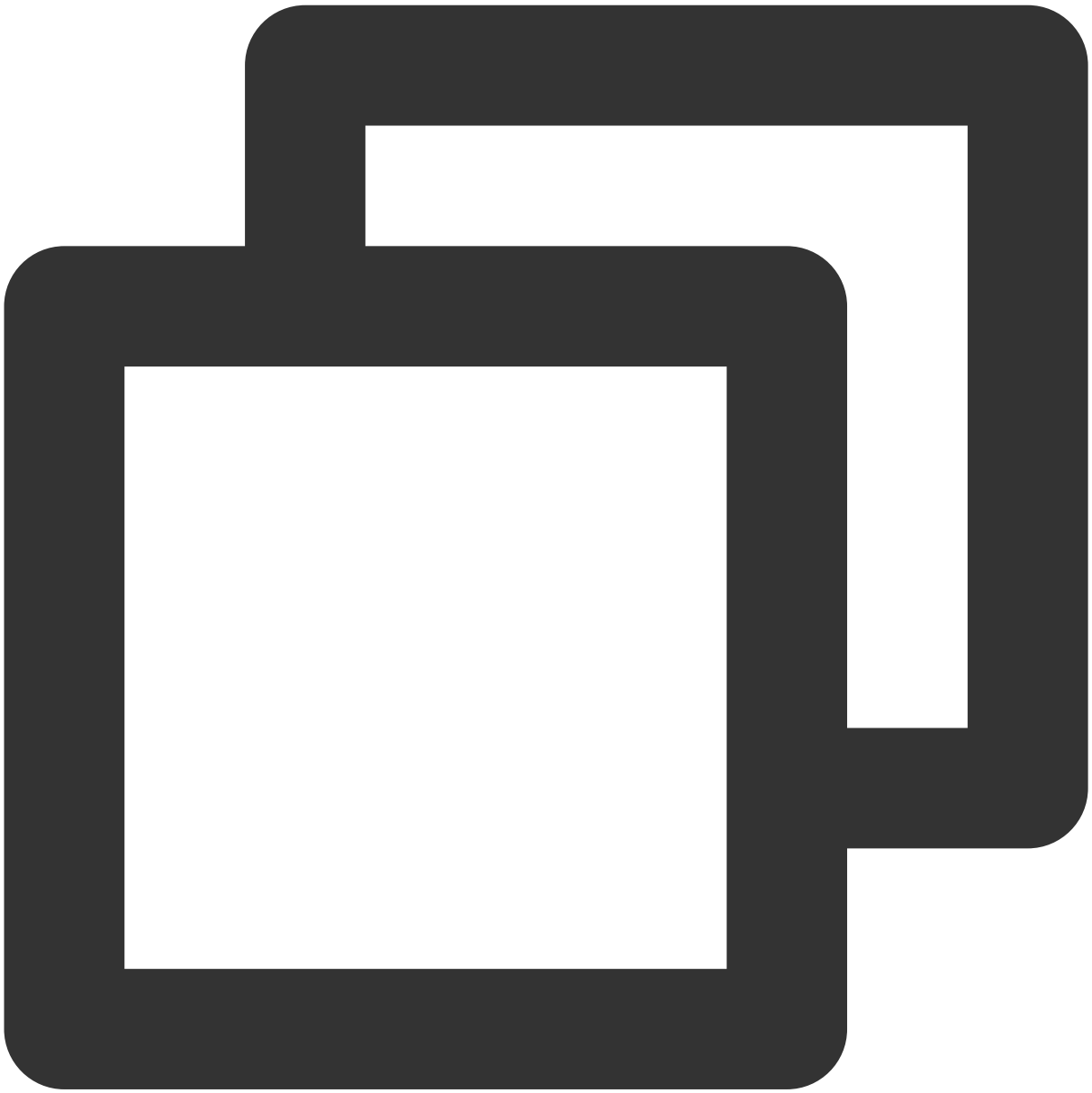
This API is used to start mic capturing.



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

stopMicrophone

This API is used to stop mic capturing.



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

setAudioQuality

This API is used to set audio quality.

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

The parameters are described below:

Parameter	Type	Description
quality	int	The audio quality. For more information, see setAudioQuality().

muteLocalAudio

This API is used to mute/unmute local audio.

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

The parameters are described below:

Parameter	Type	Description
mute	boolean	Whether to mute or unmute audio. For more information, see muteLocalAudio().

setSpeaker

This API is used to set whether to play sound from the device's speaker or receiver.

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

The parameters are described below:

Parameter	Type	Description
useSpeaker	boolean	`true` : Speaker; `false` : Receiver

setAudioCaptureVolume

This API is used to set the mic capturing volume.

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

The parameters are described below:

Parameter	Type	Description
volume	int	Capturing volume. Value range: 0-100 (default: 100)

setAudioPlayoutVolume

This API is used to set the playback volume.

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

The parameters are described below:

Parameter	Type	Description
volume	int	Playback volume. Value range: 0-100 (default: 100)

muteRemoteAudio

This API is used to mute/unmute a specified user.

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

The parameters are described below:

Parameter	Type	Description
userId	String	User ID
mute	boolean	`true` : Mute; `false` : Unmute

muteAllRemoteAudio

This API is used to mute/unmute all users.

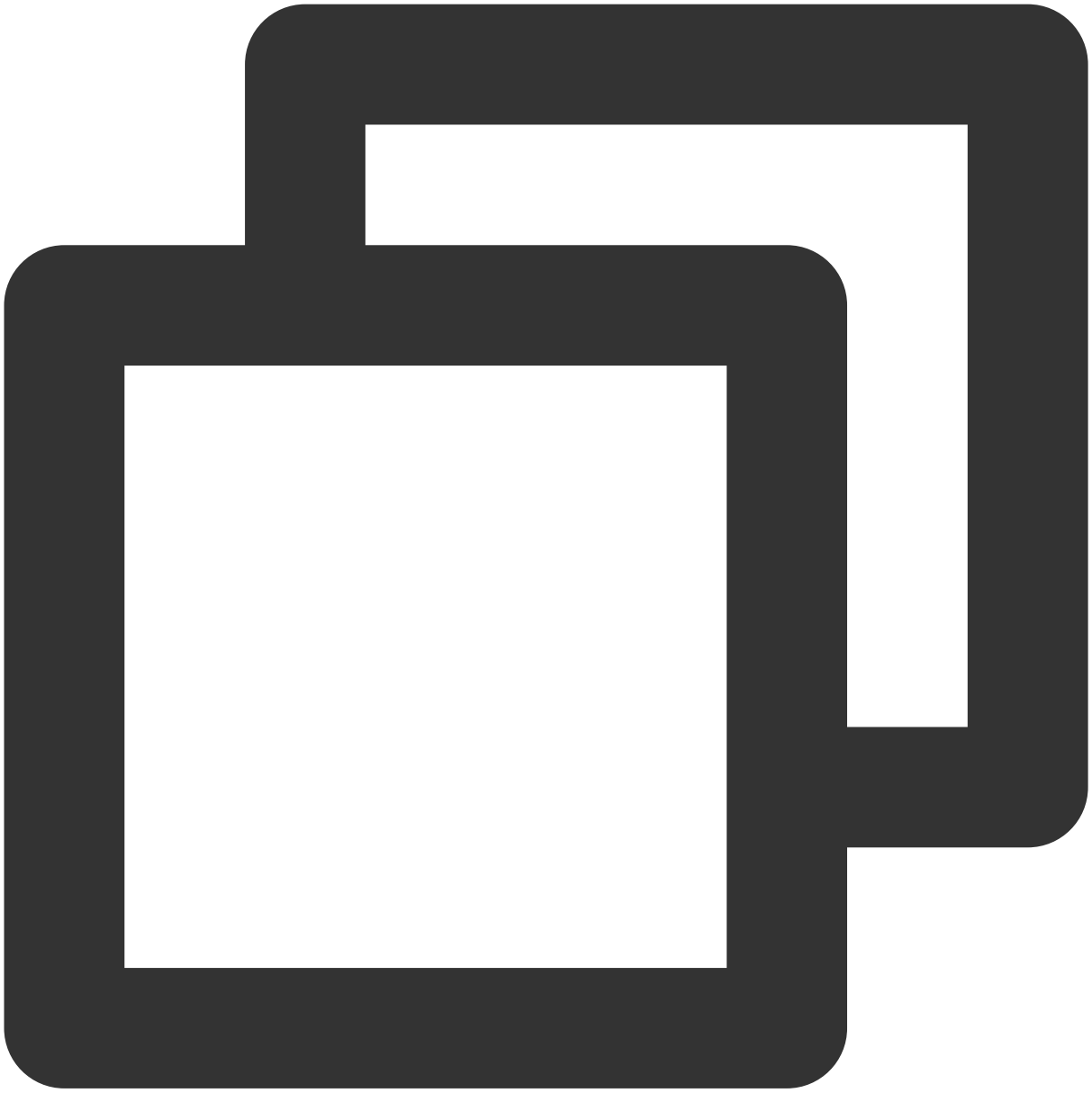
```
public abstract void muteAllRemoteAudio(boolean mute);
```

The parameters are described below:

Parameter	Type	Description
mute	boolean	`true` : Mute; `false` : Unmute

setVoiceEarMonitorEnable

This API is used to enable/disable in-ear monitoring.



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

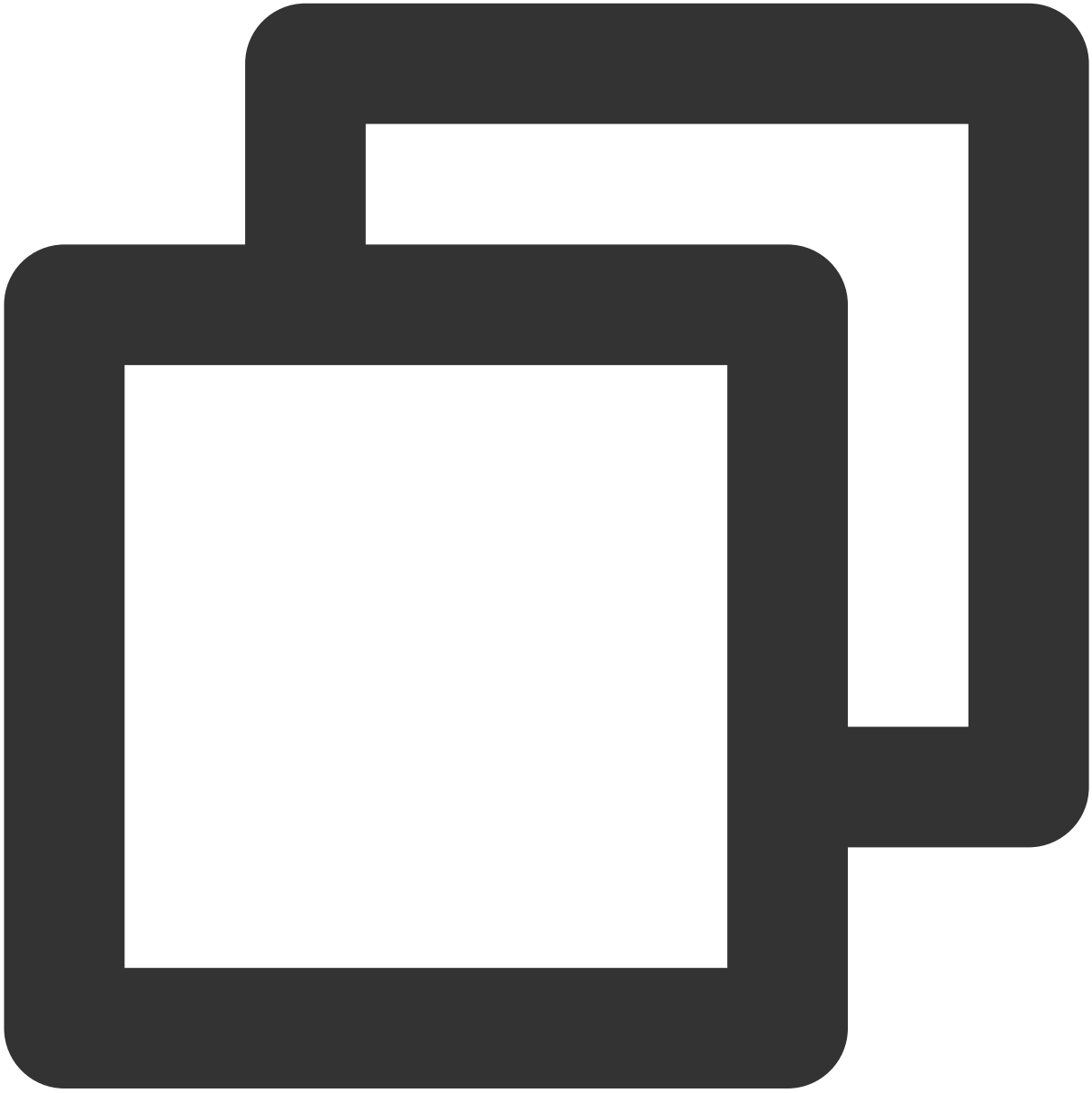
The parameters are described below:

Parameter	Type	Description
enable	boolean	`true` : Enable; `false` : Disable

Background Music and Audio Effect APIs

getAudioEffectManager

This API is used to get the background music and audio effect management object TXAudioEffectManager.

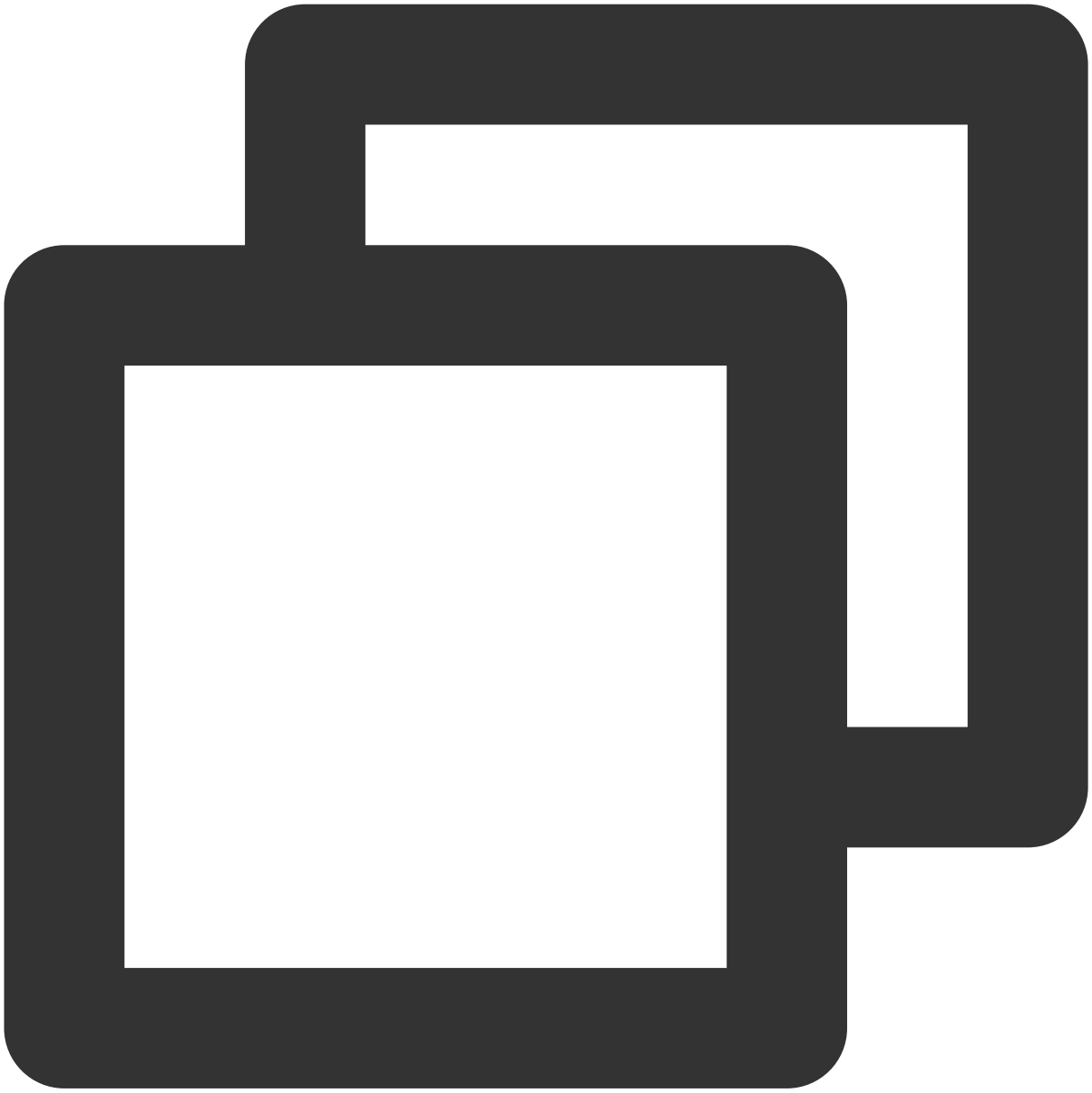


```
public abstract TXAudioEffectManager getAudioEffectManager();
```

Message Sending APIs

sendRoomTextMsg

This API is used to broadcast a text chat message in a room, which is generally used for on-screen comments.



```
public abstract void sendRoomTextMsg(String message, TRTCVoiceRoomCallback.ActionCa
```

The parameters are described below:

Parameter	Type	Description
message	String	Text message
callback	ActionCallback	Callback for the operation

sendRoomCustomMsg

This API is used to send a custom text message.

```
public abstract void sendRoomCustomMsg(String cmd, String message, TRTCVoiceRoomCal
```

The parameters are described below:

Parameter	Type	Description
cmd	String	A custom command word used to distinguish between different message

		types.
message	String	Text message
callback	ActionCallback	Callback for the operation

Invitation Signaling APIs

sendInvitation

This API is used to send an invitation.

```
public abstract String sendInvitation(String cmd, String userId, String content, TR
```

The parameters are described below:

Parameter	Type	Description
cmd	String	Custom command of business
userId	String	Invitee's user ID
content	String	Invitation content

callback	ActionCallback	Callback for the operation

Response parameters:

Parameter	Type	Description
inviteId	String	Invitation ID

acceptInvitation

This API is used to accept an invitation.

```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
callback	ActionCallback	Callback for the operation

rejectInvitation

This API is used to decline an invitation.

```
public abstract void rejectInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
callback	ActionCallback	Callback for the operation

cancelInvitation

This API is used to cancel an invitation.

```
public abstract void cancelInvitation(String id, TRTCVoiceRoomCallback.ActionCallba
```

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
callback	ActionCallback	Callback for the operation

TRTCVoiceRoomDelegate Event Callback APIs

Common Event Callback APIs

onError

Callback for error.

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

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

The parameters are described below:

Parameter	Type	Description
code	int	Error code
message	String	Error message

onWarning

Callback for warning.

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

The parameters are described below:

Parameter	Type	Description
code	int	Error code
message	String	Warning message

onDebugLog

Callback for log.

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

The parameters are described below:

Parameter	Type	Description
message	String	Log information

Room Event Callback APIs

onRoomDestroy

Callback for room termination. When the owner terminates the room, all users in the room will receive this callback.

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

The parameters are described below:

Parameter	Type	Description

roomId	String	Room ID

onRoomInfoChange

Callback for change of room information. This callback is sent after successful room entry. The information in `roomInfo` is passed in by the room owner during room creation.

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

The parameters are described below:

Parameter	Type	Description
roomInfo	RoomInfo	Room information

onUserMicrophoneMute

Callback of whether a user's mic is muted. When a user calls `muteLocalAudio` , all members in the room will receive this callback.

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

The parameters are described below:

Parameter	Type	Description
userId	String	User ID
mute	boolean	`true` : muted; `false` : unmuted

onUserVolumeUpdate

Notification to all members of the volume after the volume reminder is enabled.

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVol
```

The parameters are described below:

Parameter	Type	Description
userVolumes	ListList<TRTCCloudDef.TRTCVolumeInfo>	List of user IDs
totalVolume	int	Total volume. Value range: 0-100

Seat Callback APIs

onSeatListChange

Callback for all seat changes.

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

The parameters are described below:

Parameter	Type	Description
seatInfoList	List<SeatInfo>	Full seat list

onAnchorEnterSeat

Someone became a speaker or was made a speaker by the owner.

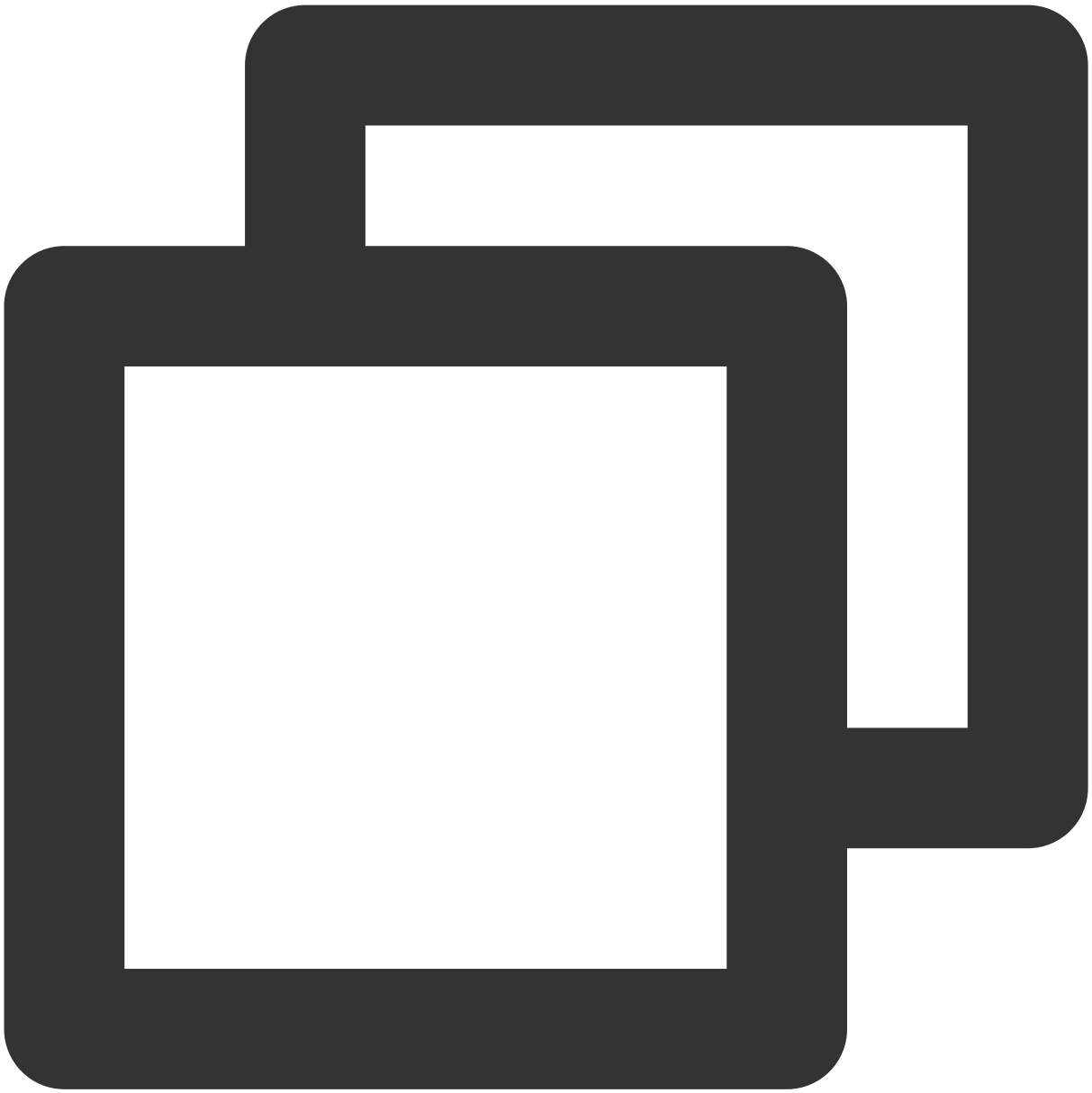
```
void onAnchorEnterSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

The parameters are described below:

Parameter	Type	Description
index	int	The seat taken
user	UserInfo	Details of the user who took the seat

onAnchorLeaveSeat

A speaker became a listener or was moved to listeners by the room owner.



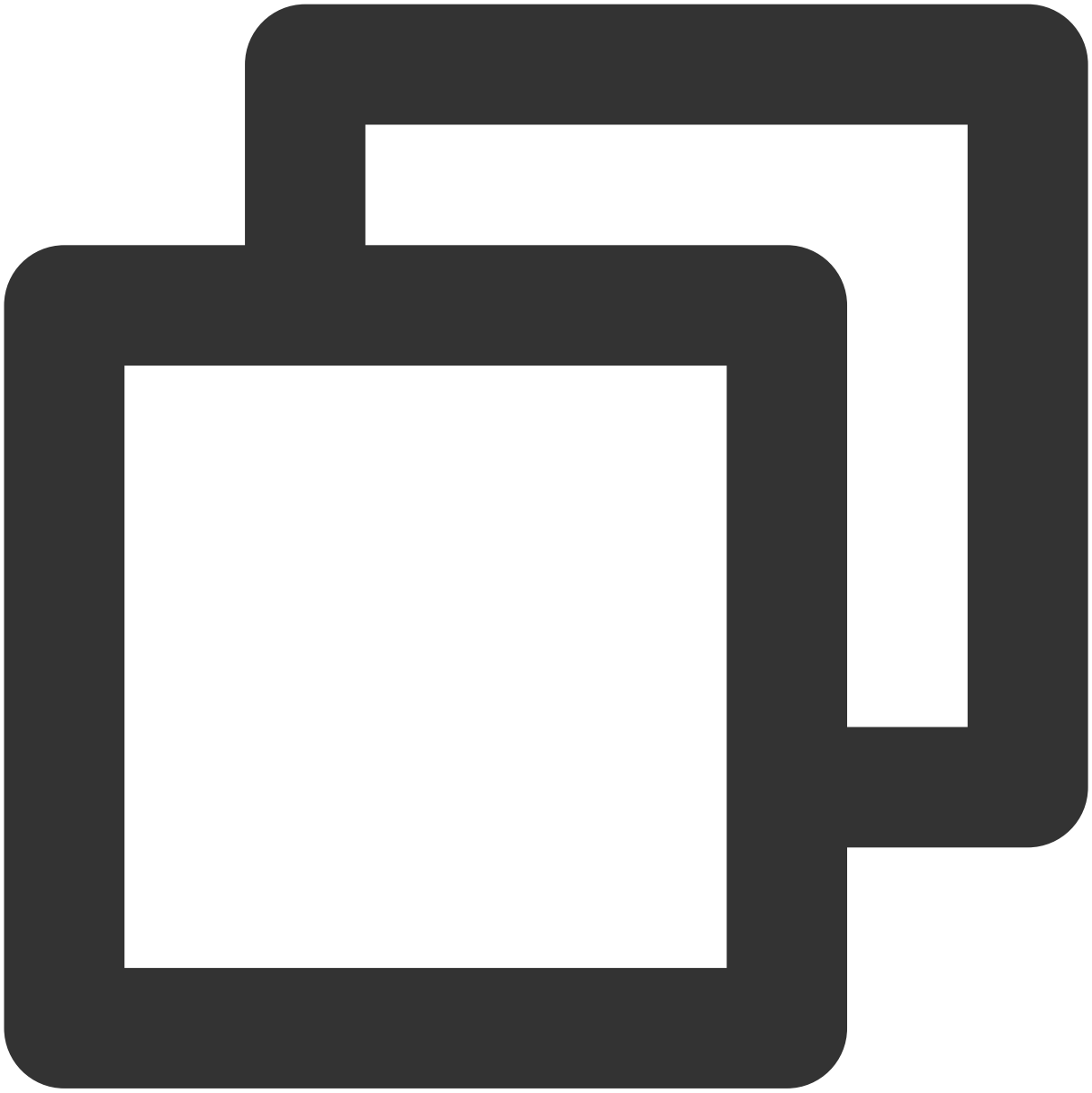
```
void onAnchorLeaveSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

The parameters are described below:

Parameter	Type	Description
index	int	The seat previously occupied by the speaker
user	UserInfo	Details of the user who took the seat

onSeatMute

The room owner muted/unmuted a seat.



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

The parameters are described below:

Parameter	Type	Description
index	int	The seat muted/unmuted
isMute	boolean	

		`true` : muted; `false` : unmuted

onSeatClose

The room owner blocked/unblocked a seat.

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

The parameters are described below:

Parameter	Type	Description

index	int	The seat blocked/unblocked
isClose	boolean	`true` : blocked; `false` : unblocked

Callback APIs for Room Entry/Exit by Listener

onAudienceEnter

A listener entered the room.

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

The parameters are described below:

Parameter	Type	Description
userInfo	UserInfo	Information of the listener who entered the room

onAudienceExit

A listener exited the room.

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

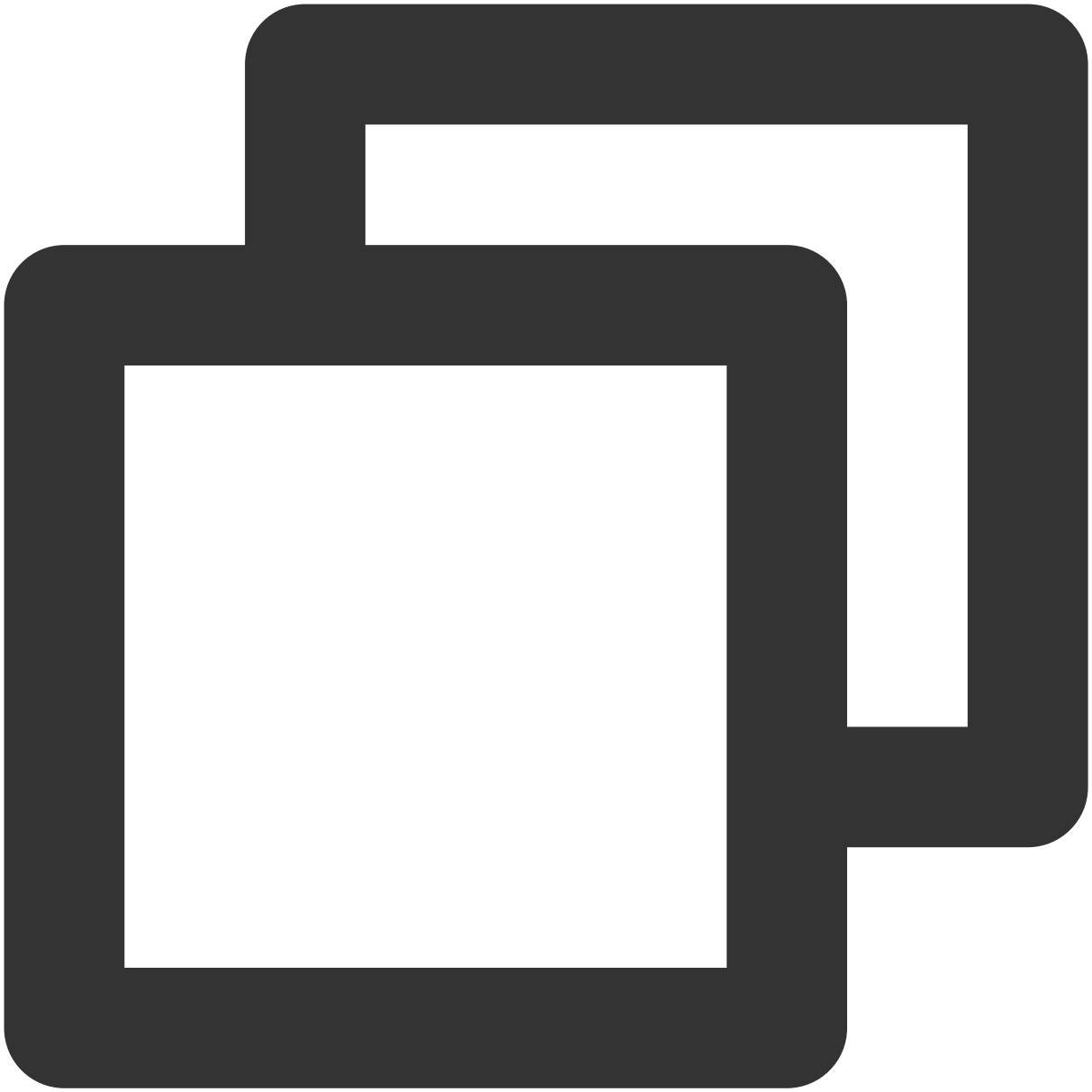
The parameters are described below:

Parameter	Type	Description
userInfo	UserInfo	Information of the listener who exited the room

Message Event Callback APIs

onRecvRoomTextMsg

Callback for receiving a text chat message.



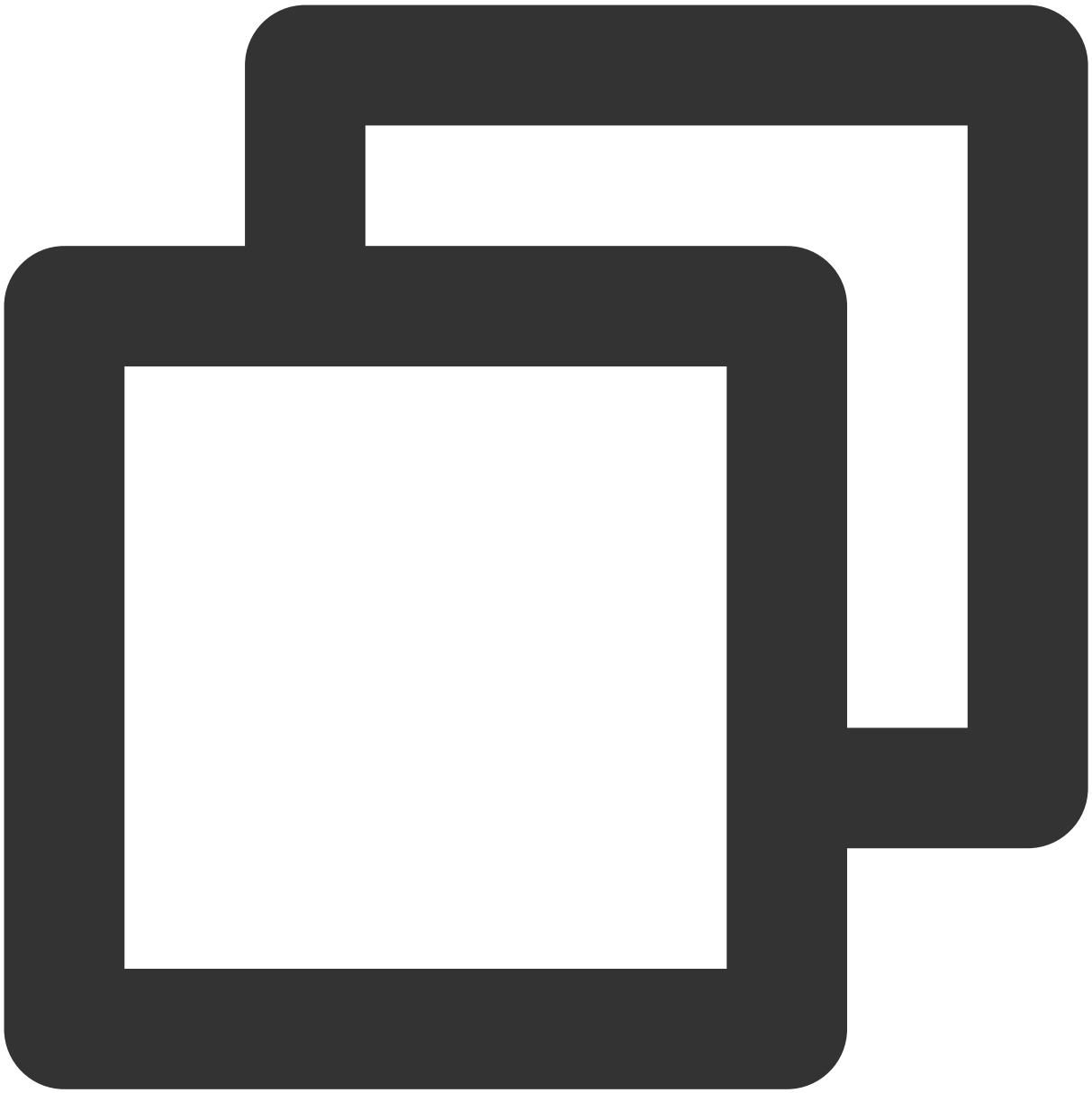
```
void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

The parameters are described below:

Parameter	Type	Description
message	String	Text message
userInfo	UserInfo	Information of the sender

onRecvRoomCustomMsg

A custom message was received.



```
void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo user
```

The parameters are described below:

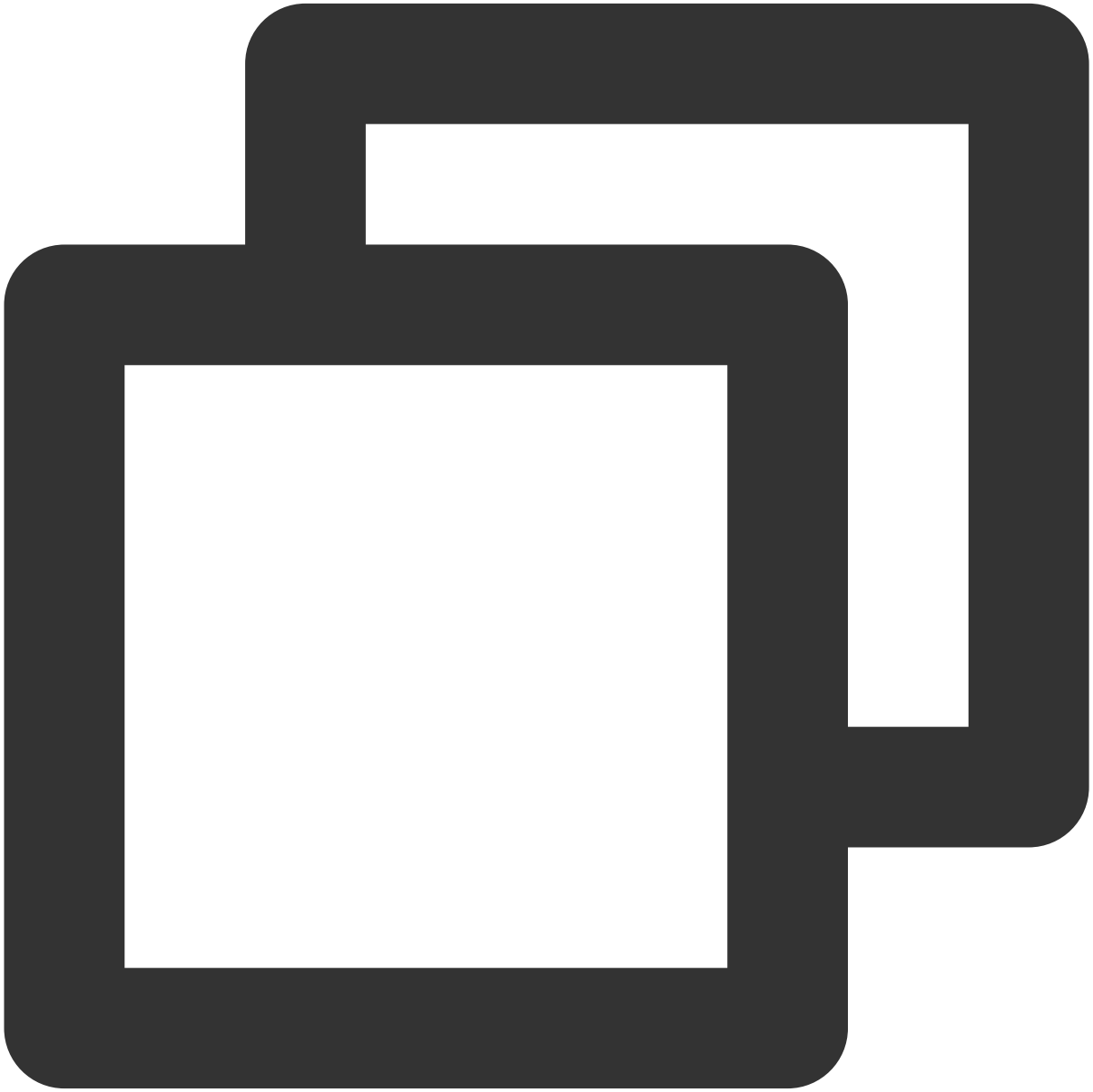
Parameter	Type	Description
cmd	String	Custom command word used to distinguish between different message types
message	String	Text message

userInfo	UserInfo	Information of the sender

Invitation Signaling Callback APIs

onReceiveNewInvitation

An invitation was received.



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

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
inviter	String	Inviter's user ID
cmd	String	Custom command word specified by business
content	String	Content specified by business

onInviteeAccepted

The invitee accepted the invitation

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

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
invitee	String	Invitee's user ID

onInviteeRejected

The invitee declined the invitation

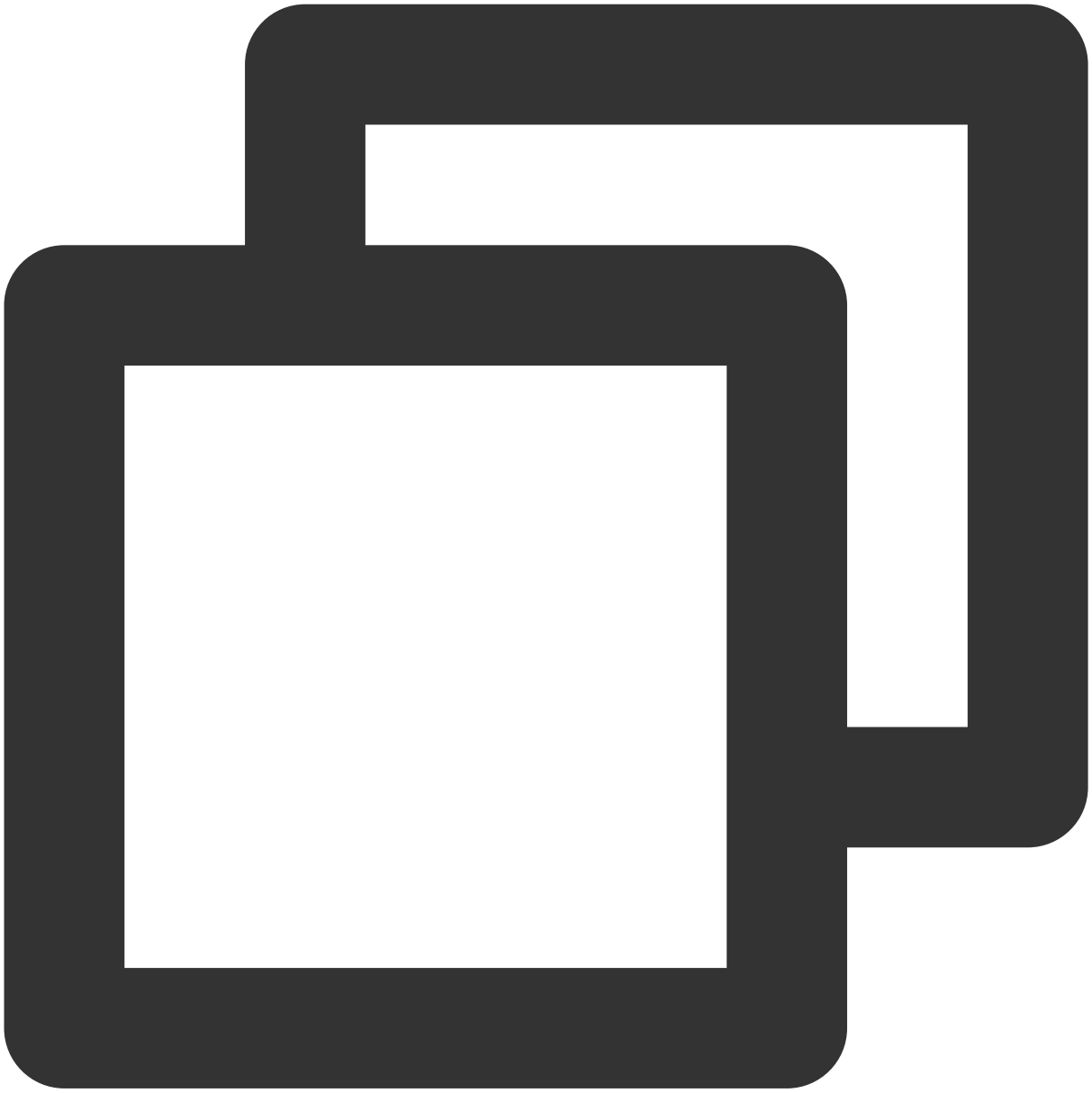
```
void onInviteeRejected(String id, String invitee);
```

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
invitee	String	Invitee's user ID

onInvitationCancelled

The inviter canceled the invitation.



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

The parameters are described below:

Parameter	Type	Description
id	String	Invitation ID
inviter	String	Inviter's user ID