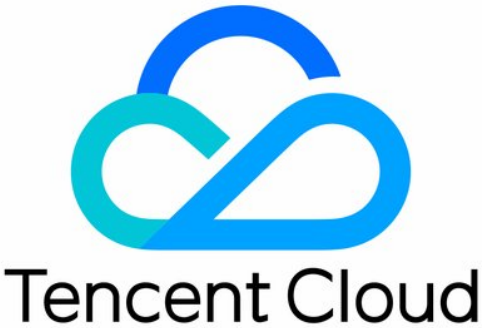
Tencent Smart Advisor-Tencent RTC Copilot

Scenario-Based Solutions

Product Documentation



©2013-2022 Tencent Cloud. All rights reserved.

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

Tencent Cloud

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

Scenario-Based Solutions
- Overview of Scenario-Based Solutions
- Social Entertainment
  - Voice Chat Room
    - Scenario Solution
    - Quick Integration Guide
      - Android
      - iOS
  - Online Karaoke
    - Scenario Solution
    - Quick Integration Guide
      - Android
      - iOS
  - Live Show Streaming
    - Scenario Solution
    - Quick Integration Guide
      - Android
      - iOS
- Live Shopping
  - Live Streaming with Goods
    - Scenario Solution
    - Quick Integration Guide
      - Android
      - iOS
- Audio/Video Call
  - 1V1 Audio and Video Call
    - Scenario Solution
    - Quick Integration Guide
      - Android
      - iOS

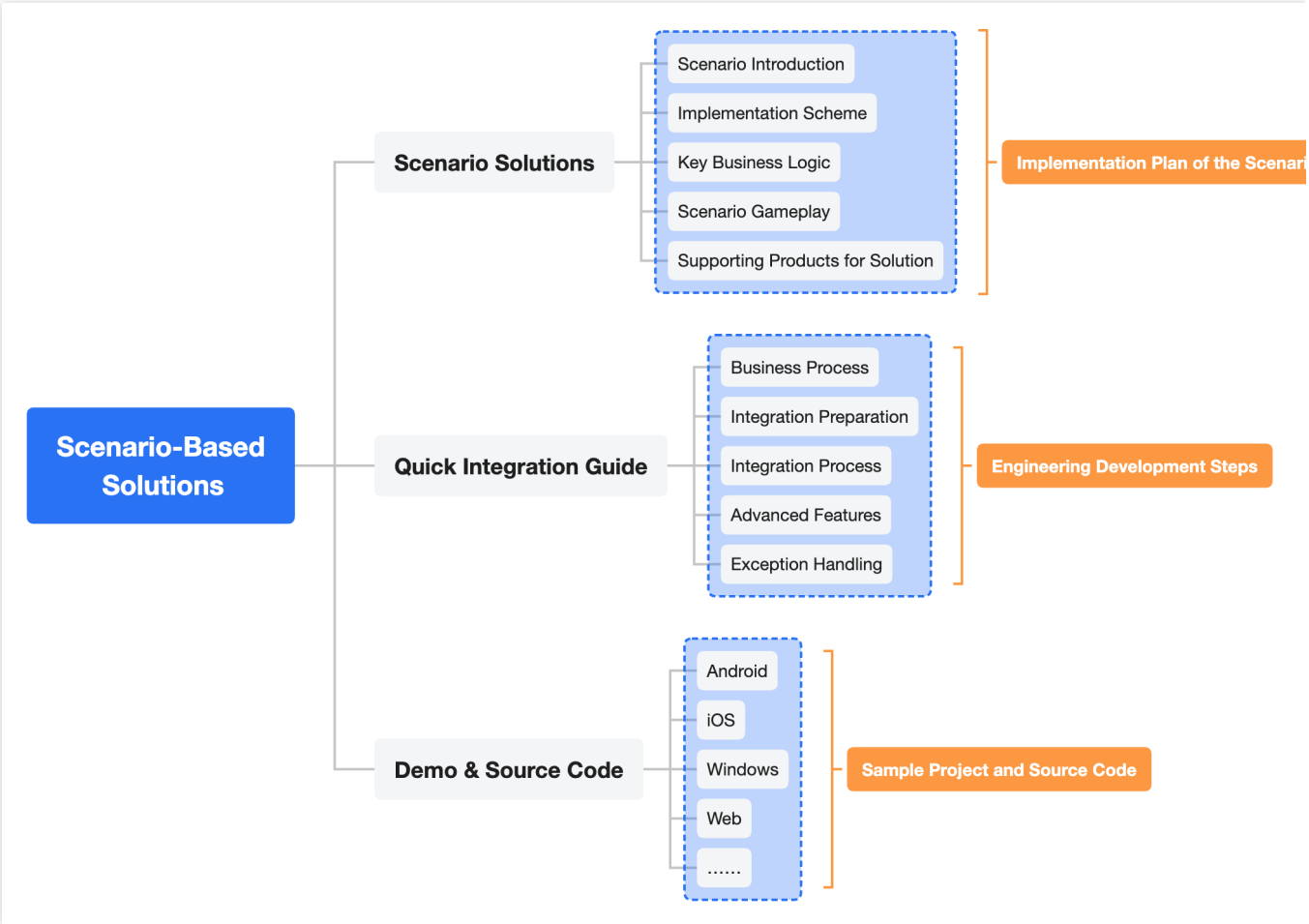
©2013-2022 Tencent Cloud. All rights reserved.

Scenario-Based Solutions
Overview of Scenario-Based Solutions

Last updated：2024-07-22 14:26:43

Overview

The scenario-based solutions systematically summarize the scenario implementation scheme and engineering development steps to help you quickly familiarize yourself with the relevant scenarios and complete the TRTC integration and launch work. Covers voice chat room, online karaoke, live show streaming, and other social entertainment scenarios, and live streaming marketing and 1V1 audio and video call, and other scenario-based solutions and quick integration guide. It can help improve integration development efficiency and facilitate the rapid implementation of business scenarios.



Project Planning Phase

At this stage, you can see the experience demo and scenario-based solutions we provided to complete the project planning.

©2013-2022 Tencent Cloud. All rights reserved.

Experience Demo: Quickly get started with related scenarios and gameplay and experience the project implementation effect.
Scenario-Based Solutions: Be familiar with the relevant scenarios, functional modules, and technical architecture, and finalize the project implementation scheme.

Project Development Phase

At this stage, you can see the demo source code and Quick Integration Guide we provided to complete project development.
Demo Source Code: Provides complete engineering source code for the relevant scenarios. You can refer to the code implementation of related modules.
Quick Integration Guide: Provides full-process integration guides for the relevant scenarios to help you quickly complete project development.

©2013-2022 Tencent Cloud. All rights reserved.

Social Entertainment
Voice Chat Room
Scenario Solution

Last updated：2024-07-25 16:45:57

Scenario Introduction

A voice chat room provides a virtual space for audio-only online social interaction. Typically, the room contains several seats where anchors and co-speakers can engage in voice conversations, while other listeners can join the room to listen in. The number of seats and listeners vary by room type. Tencent Cloud's Tencent Real-Time Communication (TRTC) supports up to 50 people chatting on the mic simultaneously, with smooth transitions between speaking and listening, and a voice chat latency of less than 300ms. It includes a variety of audio effects like voice changing, ambiance effects, and reverb to enrich the chat experience. Combined with Instant Messaging, it supports various forms of message interaction such as public chat, private chat, group chat, likes, and gift sending, creating a lively and engaging chat interaction experience.

©2013-2022 Tencent Cloud. All rights reserved.

Implementation Scheme

Implementing a complete voice chat room scenario usually involves several functional modules: Room Management, Seat Management, Audio Stream Management, On-Cloud Recording, etc. The key actions and feature points under each functional module are as follows:

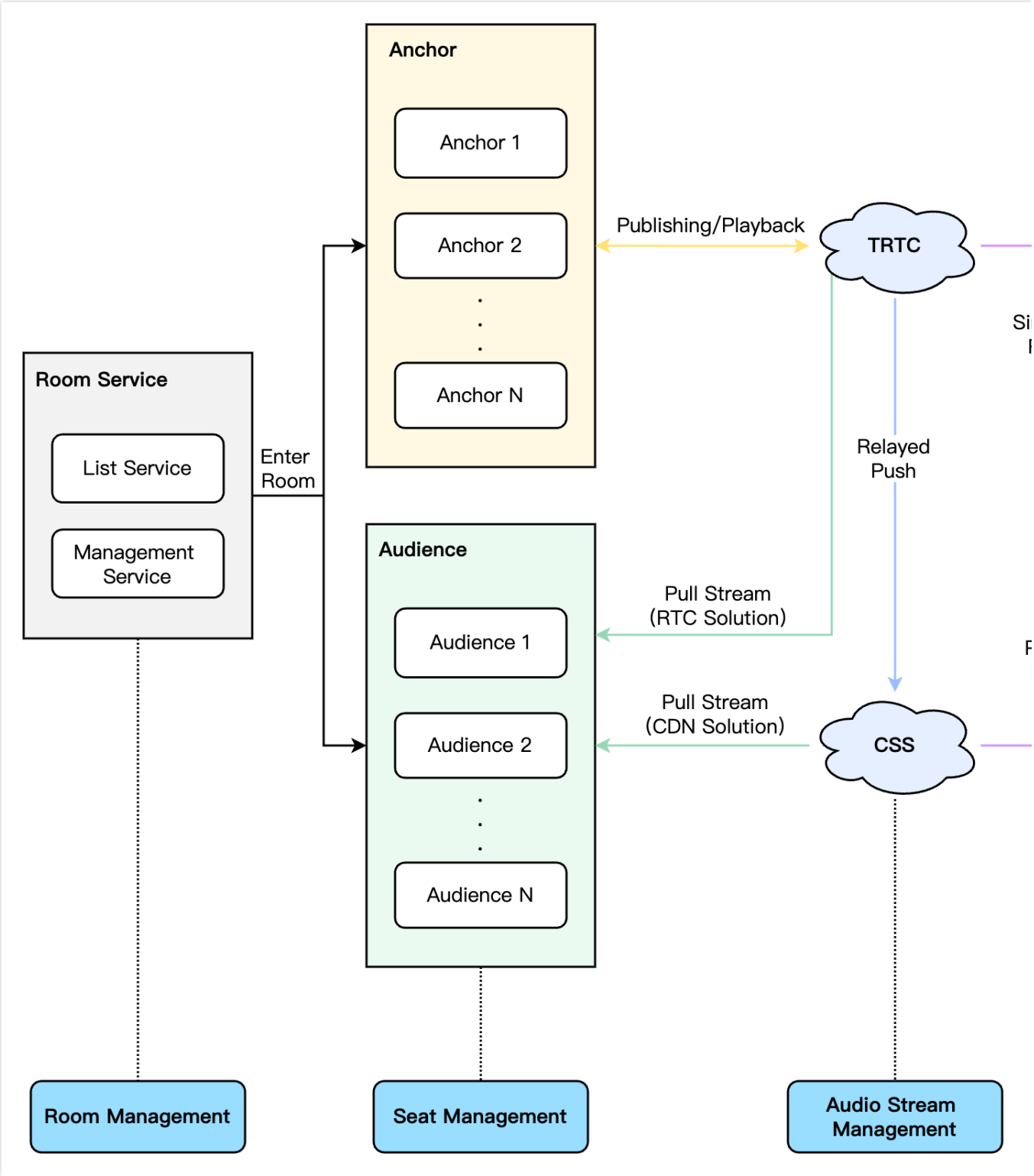
Functional Module	Key Actions and Feature Points
Room Management	Room list, create a room, enter a room, exit a room, and terminate a room.
Seat Management	Become a speaker, invite a listener to speak, become a listener, remove a speaker, mute a seat, lock a seat and move a seat.

©2013-2022 Tencent Cloud. All rights reserved.

Audio Stream Management	Publishing/Playback Architecture Solution, Real-Time Stream Subscription Mode
On-Cloud Recording	TRTC On-Cloud Recording.

The overall business architecture of the voice chat room is shown in the figure below. Room owners create voice chat rooms, and users can choose to join rooms that interest them. After entering a room, users can go on the seat and engage in voice interaction with the speakers. However, due to compliance requirements, the voice content in the room needs to be recorded and reviewed.

©2013-2022 Tencent Cloud. All rights reserved.



Room Management

The Room Management Module is primarily responsible for maintaining the room list and includes the following features:

Create Room: After users log in to the business system, they can create a room. The room list needs to be updated after a room is created.

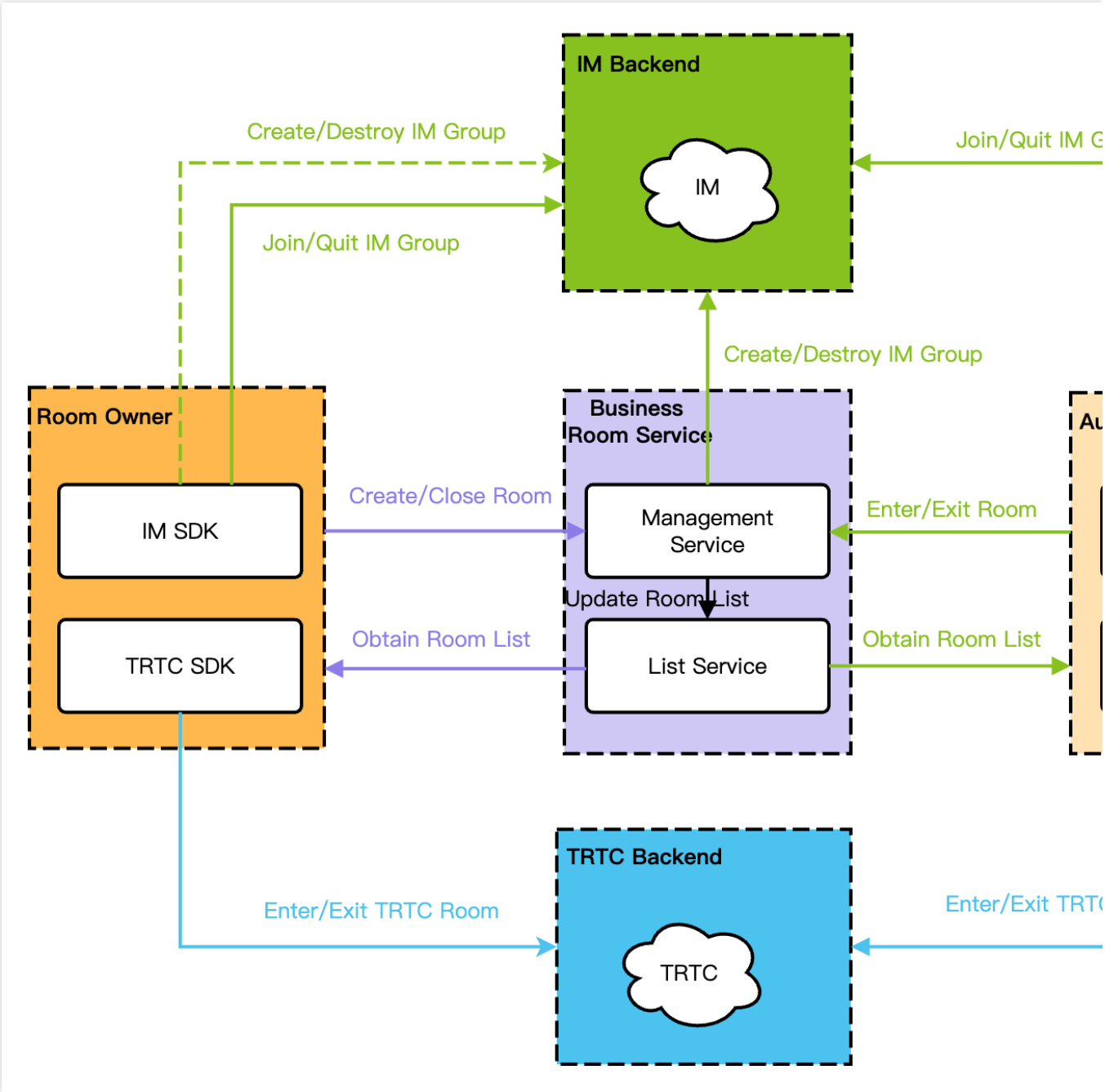
©2013-2022 Tencent Cloud. All rights reserved.

Enter Room: Users can choose to enter an existing room. Upon entering, the current list of room members should be updated.
Exit Room: Users can choose to exit the current room. Upon exiting, the current list of room members needs to be updated with a delete operation.
Terminate Room: After all users exit the room, it needs to be terminateed. Upon destruction, the room list needs to be updated with a delete operation.

Scheme Architecture

In the overall architecture of room management, there are primarily three major modules involved:
Room Management: Mainly used for the maintenance and administration of the room list, such as synchronizing the properties and status of rooms. Features include room list querying, entering/exiting rooms, and creating/terminateing rooms.
IM group management: This module is primarily used for managing room member lists, signaling transmission, and message interactions. For instance, it handles actions such as approving/rejecting a speaking request, inviting a listener to speak/removing a listener, muting/unmuting a seat, and blocking/unblocking a seat. This feature is also distinguished by group dimension, including creating groups, joining groups, exiting groups, and terminateing groups.
TRTC Room Management: This module is mainly used for the interaction and transmission of audio streams. For instance, it facilitates the sending and listening of voice/music between anchors and audiences. It is also distinguished by room dimensions. Features include entering/exiting TRTC rooms.

©2013-2022 Tencent Cloud. All rights reserved.



Specific Implementation

In room management, different user roles have different feature permissions and implementation processes. In voice chat rooms, there are mainly two roles: the room owner and the listeners. For a detailed description and differences of roles, see the table below:

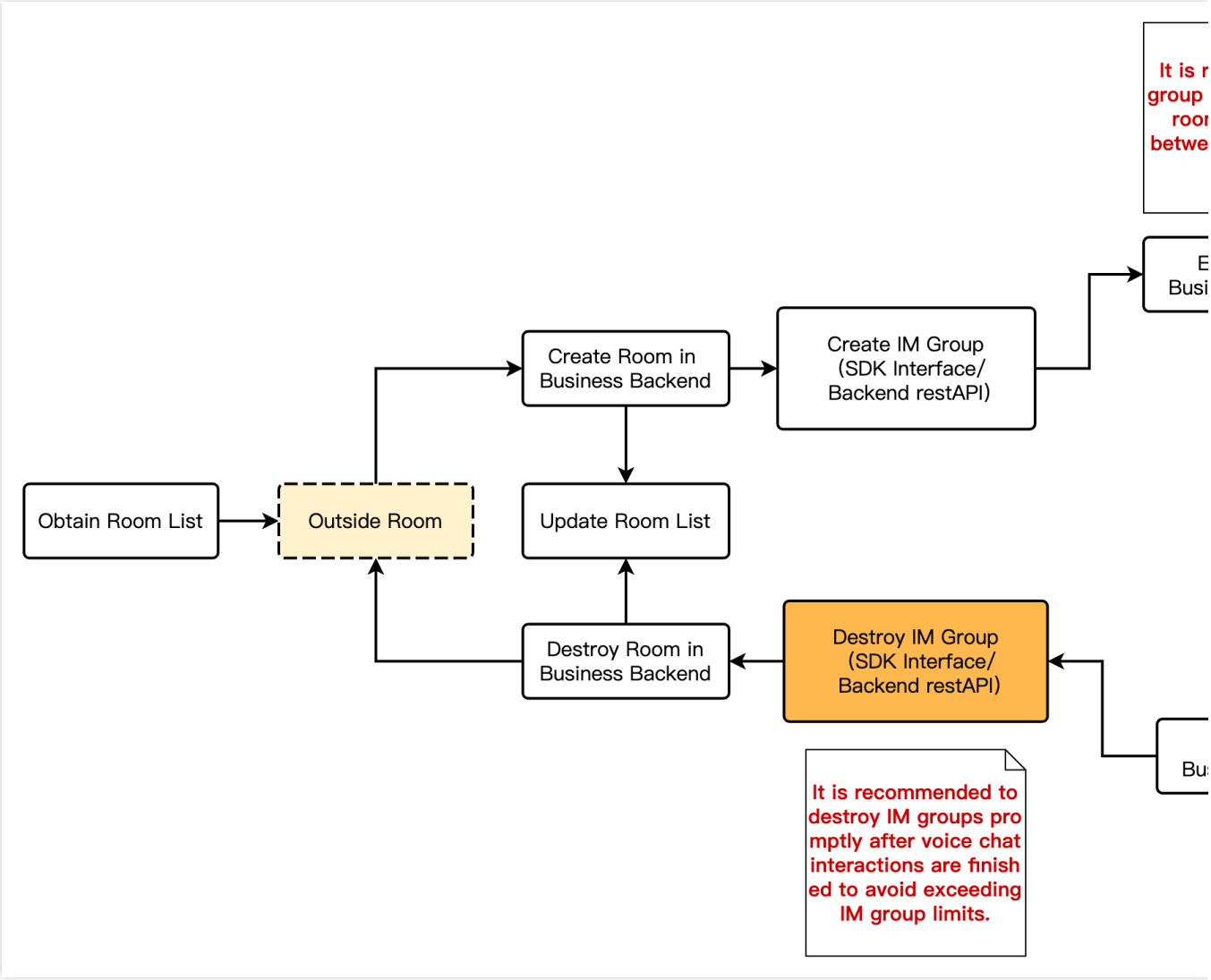
Roles	Description	Differences
Room Owner	The room owner with the highest authority in the room can create or terminate the room.	The role must be an anchor. Creates or terminates rooms/IM groups/RTC rooms

©2013-2022 Tencent Cloud. All rights reserved.

Listeners	Participants in the room can also take the mic to become the anchor.	The role can be either audience or anchor. Enter/Exit Room

Implementation Process

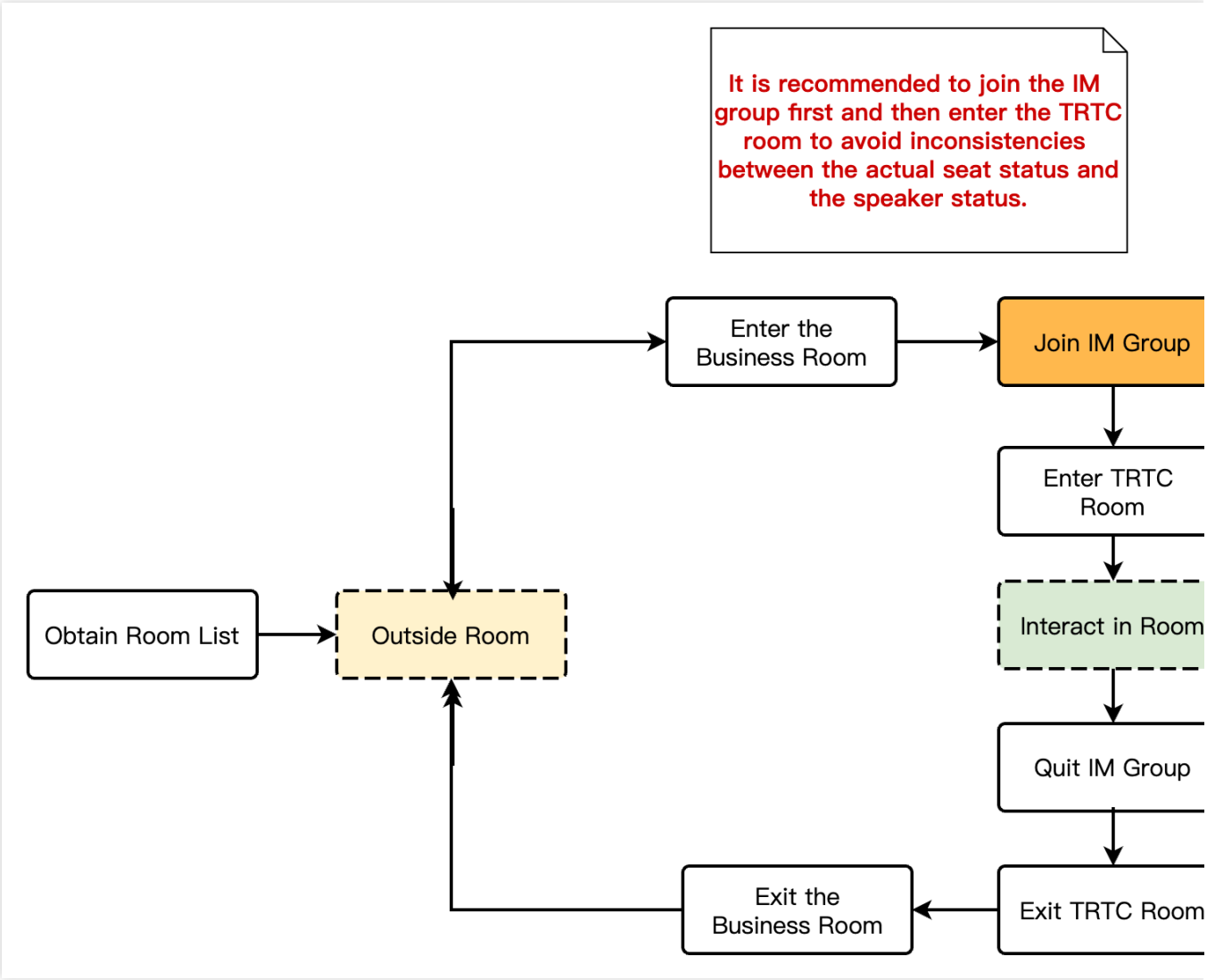
Room Owner



1. Obtain the room list.
2. Create the corresponding room through the business API.
3. Create an IM group.
4. Enter the Room/IM Room/RTC Room, and interact with others.
5. Exit the IM Room/RTC Room/Room.
6. Terminate the IM Group.

Listeners

©2013-2022 Tencent Cloud. All rights reserved.



1. Obtain the room list.
2. Enter the Room/IM Group/RTC Room, and interact with others.
3. Exit the IM Group/RTC Room/Room.

Seat Management

In a Voice Chat Room, the seats are usually ordered and limited. For example, an audience needs the room owner's approval to speak orderly. Generally, the number of seats in a room does not exceed 10. Seat management is mainly responsible for managing the number of seats in a room according to the business scenario, as well as the status of all current seats in the room. The main features of seat management include: becoming a speaker, inviting a listener to speak, becoming a listener, removing a speaker, muting a seat, locking a seat, and moving a seat.

After users enter the room, they can only request to speak when there are idle seats available.

After the room owner approves a user to become a speaker, the seat status needs to be changed to non-idle.

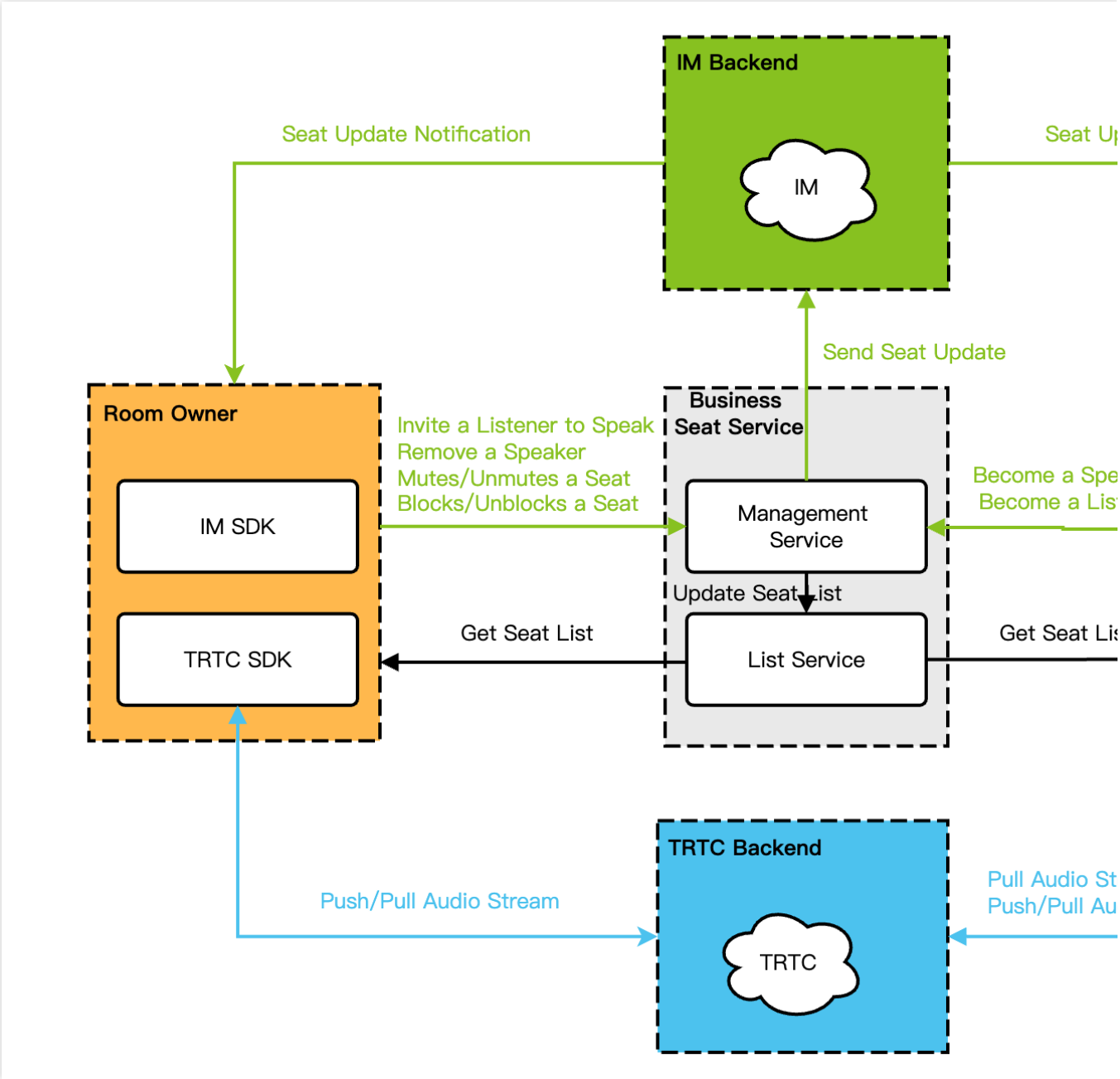
After the user stops streaming and becomes a listener, the seat status needs to be reset.

The room owner has the authority to lock the seat, invite a listener to speak, remove a speaker, mute a speaker, etc.

©2013-2022 Tencent Cloud. All rights reserved.

Scheme Architecture

The architecture of seat management will be organized by integrating Tencent Real-Time Communication (TRTC) and Instant Messaging (IM). In the entire room management architecture, the room owner has the highest authority and can invite a listener to speak, remove a speaker, mute/unmute a seat, and block/unblock a seat. Listeners can also request to speak, become speakers, and interact with other speakers in the room.



Specific Implementation

In seat management, different user roles have different feature permissions and implementation processes, primarily involving two roles: room owner and audience. For details on role descriptions and their differences, see the table

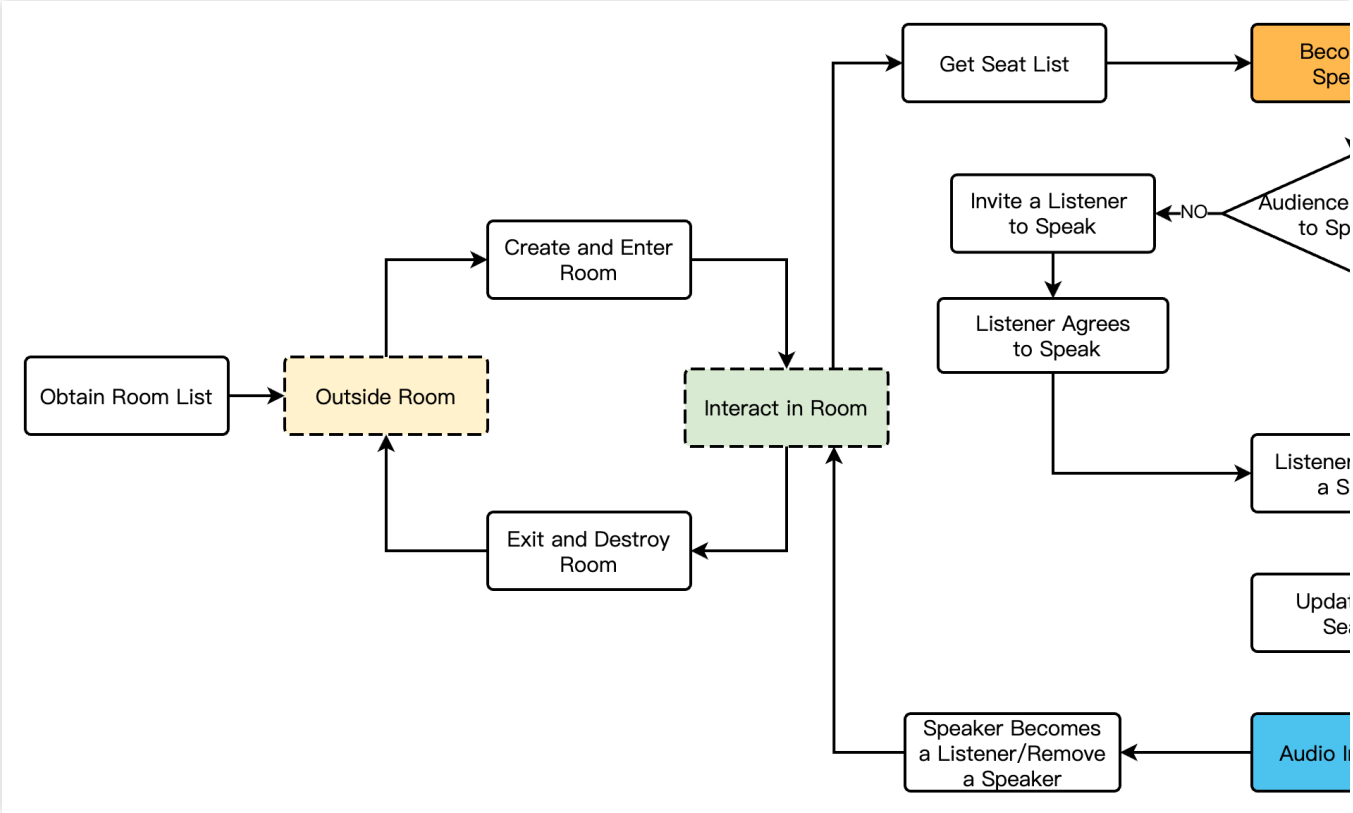
©2013-2022 Tencent Cloud. All rights reserved.

below:

Roles	Description	Differences
Room Owner	The figure with the highest authority over seats. The room owner is responsible for managing all seats. When the room owner exits the room, all seats are automatically dissolved.	The role must be an anchor. Enter the room and become a speaker Approve or reject speaking requests Invite a listener to speak/Remove a listener Mutes/Unmutes a seat Blocks/Unblocks a seat
Listeners	Room seat participants who engage in voice interactions.	The role can be either audience or anchor. Request to speak/become a listener

Implementation Process

Room Owner

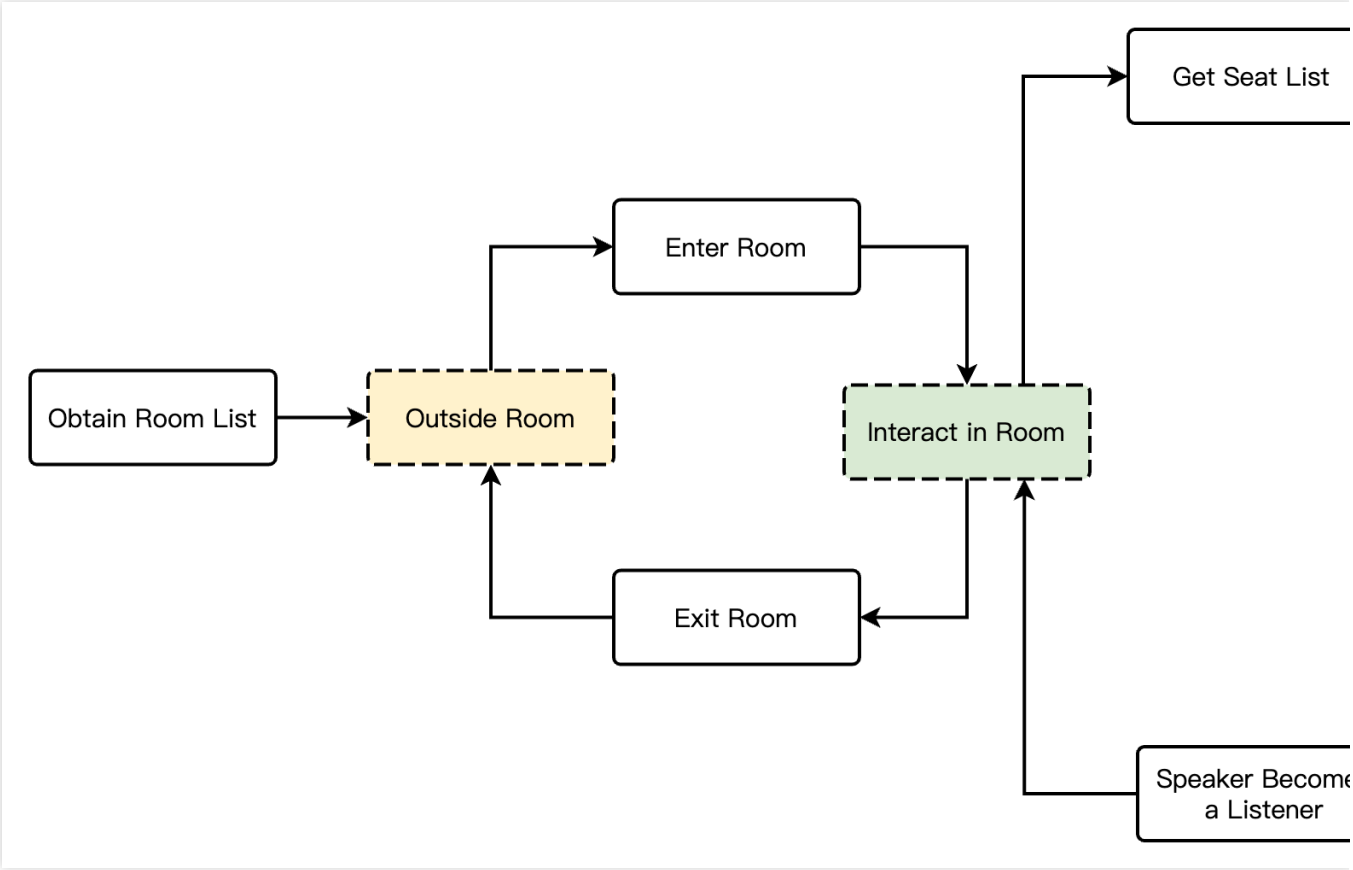


1. Anchors enter the Room Lobby and obtain the room list.
2. Anchors create and enter the room as room owners.
3. The room owner obtains the seat list based on group attributes and becomes a speaker.

©2013-2022 Tencent Cloud. All rights reserved.

4. Listeners become speakers. After becoming speakers, they can interact with other speakers. There are two ways for listeners to become speakers: they can either request to speak actively, which the room owner approves, or the room owner can invite them to speak, which they accept.
5. Speakers become listeners. There are two ways to become listeners: they can become listeners actively, or the room owner can forcibly remove them.
6. The room owner exits and terminates the room (the room is dissolved, and all speakers are forcibly removed and exit the room).

Listeners



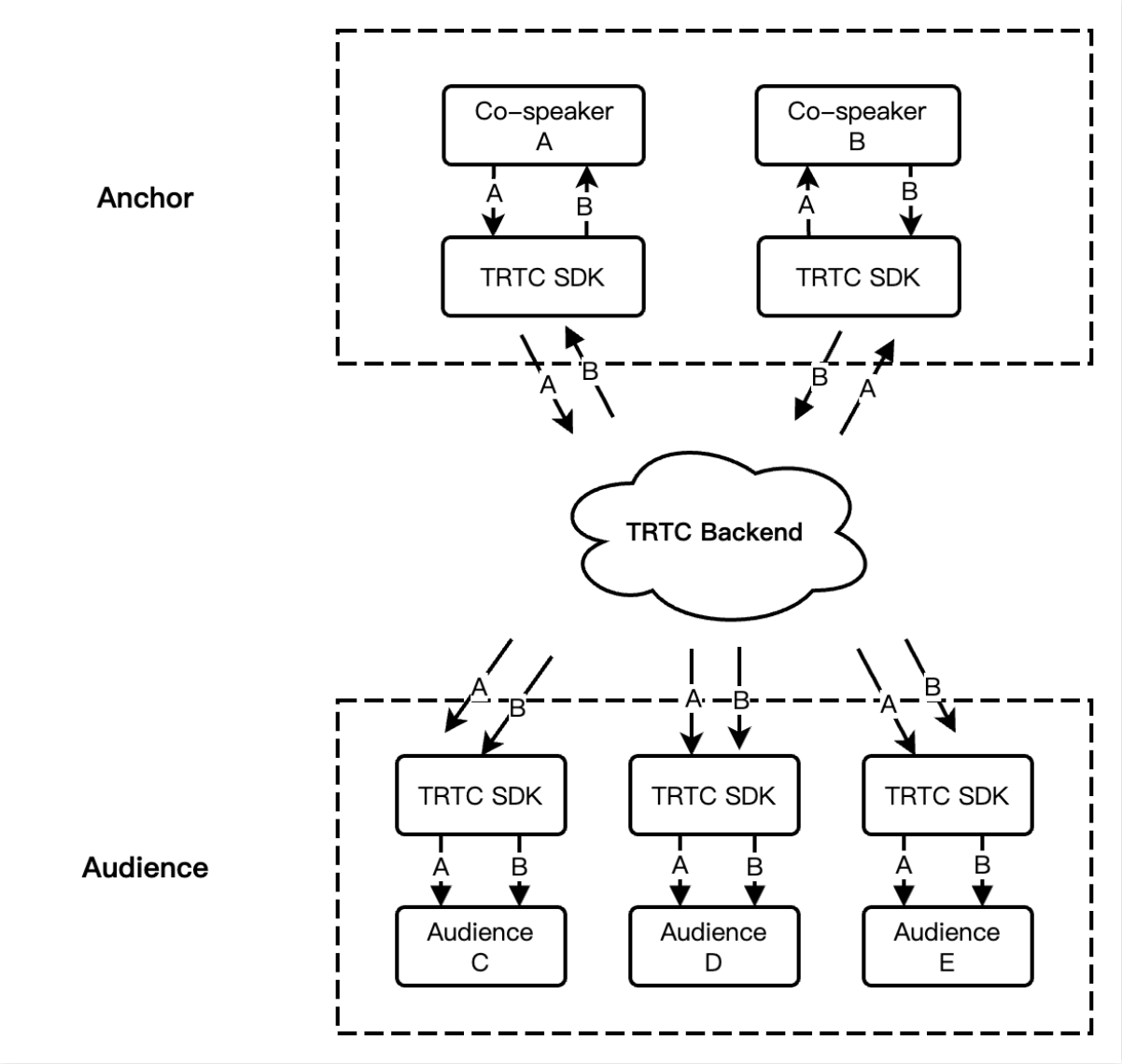
1. Listeners enter the Room Lobby and obtain the room list.
2. Listeners select and enter the room.
3. Listeners obtain the seat list based on group attributes.
4. Listeners request to speak. After approval from the room owner, they interact with other speakers.
5. Speakers become listeners and exit the room.

Audio Stream Management

The typical interactive scenario for voice chat often opts for the RTC stream access solution, as it offers simplicity and quick integration while providing the low-latency characteristics of real-time interaction. As shown in the figure below,

©2013-2022 Tencent Cloud. All rights reserved.

a classic publishing/playback architecture of real-time interactive voice chat is displayed, showcasing two roles: speakers and listeners.



For real-time stream subscription within the room, TRTC offers two subscription modes: Automatic Subscription and Manual Subscription.

Automatic Subscription: Upon entering the room, users will immediately receive the room's audio and video streams, with audio automatically playing and video automatically starting to decode.

Manual Subscription: After entering the room, users need to manually call `startRemoteView` to start the subscription and decoding of the video stream, and manually call `muteRemoteAudio` to start the playback of audio.

In most scenarios, TRTC defaults to the Automatic Subscription mode, where users subscribe to audio and video streams from all anchors in the room upon entering, achieving a better instant opening experience. The Manual

©2013-2022 Tencent Cloud. All rights reserved.

Subscription mode, on the other hand, offers greater flexibility and customizability, allowing users to selectively subscribe to audio and video streams.

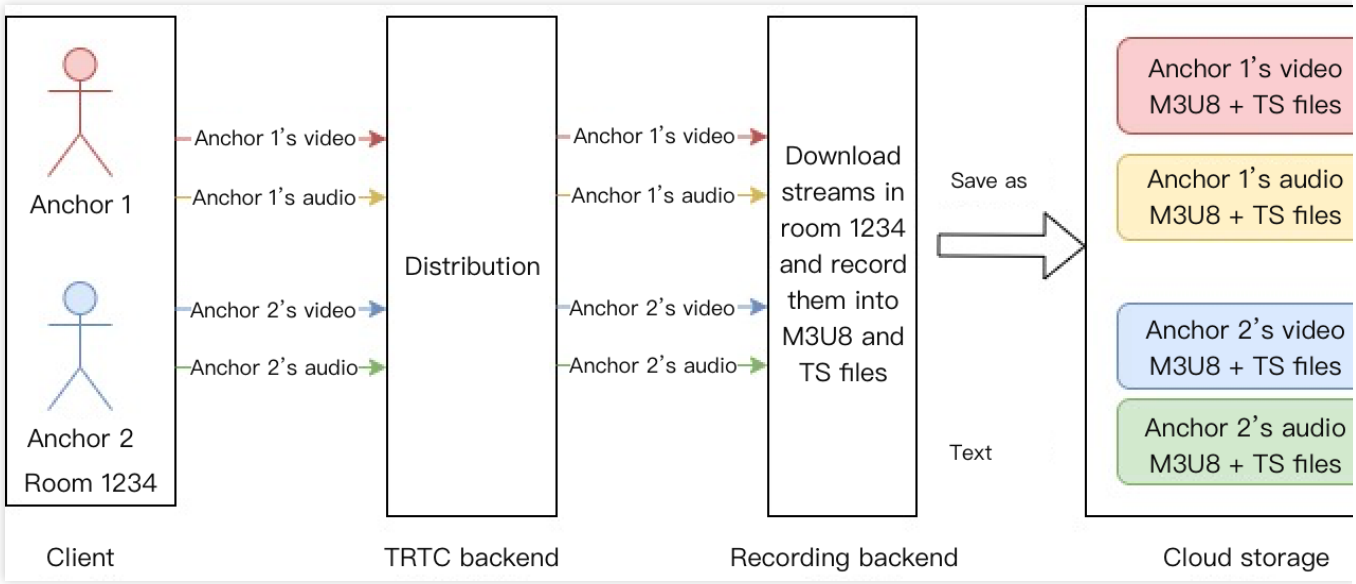
Note:

Compared to the Manual Subscription mode, Automatic Subscription does not require complex media stream subscription management. For voice chat scenarios without special requirements, Automatic Subscription is recommended.

On-Cloud Recording

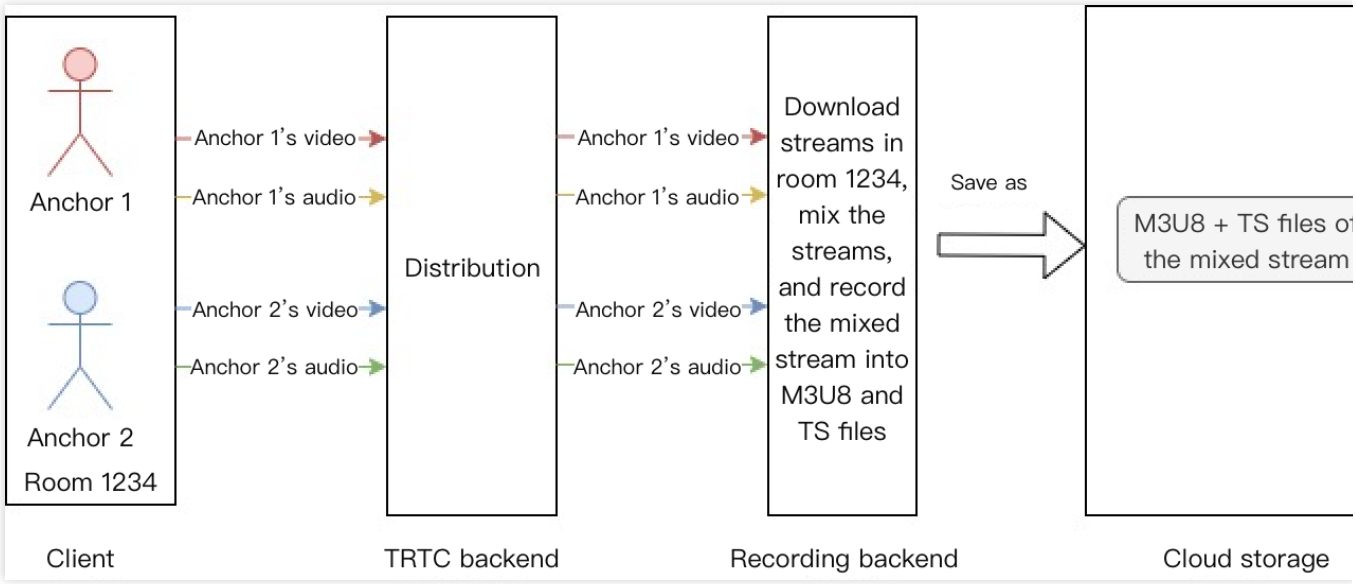
TRTC's newly upgraded On-Cloud Recording does not depend on Cloud Streaming Services. It does not require a relayed push for cloud live streaming and uses TRTC's internal real-time recording cluster for audio and video recording, offering a more comprehensive and unified recording experience.

Single Stream Recording: With TRTC's On-Cloud Recording feature, you can record each user's audio stream in the room as a separate file.



Mixed Stream Recording: Mix and record the audio media streams from the same room into a single file.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

For a detailed introduction and activation guide to TRTC On-Cloud Recording, see On-Cloud Recording.

Key Business Logic

Ganchor Mic Handling Solution

Ganchor mic, also known as blast mic or black mic, refers to the phenomenon where users not becoming speakers can speak and other users can hear their voice. The fundamental reason for the ganchor mic phenomenon is the inconsistency between the seat status and the TRTC user role status. There are several possible reasons for this issue.

When a speaker becomes a listener, the seat list is updated accordingly. However, if the seat information callback is not triggered or intercepted, the local TRTC operations for switching the audience's role and turning off the mic will not be performed. This can result in the listener being able to speak.

When a speaker becomes a listener, the seat list is updated accordingly. However, after the seat information callback is received, the audience's local call to the TRTC switch audience's role API fails, resulting in listeners being able to speak.

The app is cracked by brute force, leading to the UserSig being intercepted by hackers, allowing hackers to enter the TRTC room as an anchor and speak at will.

Detection and Handling of Ganchor Mics

By detecting ganchor mics, we can proactively identify and promptly handle them. It is recommended to use a server detection solution: Real-time anchor list comparison detection.

Principle of the Solution: In the voice chat room scenario, user roles are divided into anchors and audiences, with only the anchor being able to upstream local audio. Therefore, ganchor mics can be detected by comparing the seat

©2013-2022 Tencent Cloud. All rights reserved.

list with the TRTC role list. TRTC provides server-side room and media event callbacks. By listening for events such as entering the room, switching roles, and leaving the room, you can maintain a real-time anchor list for the current room. Then, by comparing the TRTC real-time anchor list with the full seat list, ganchor mics can be easily detected and identified, and actions such as removing from the room or muting can be performed.

1. Tencent Real-Time Communication (TRTC) Console supports self-configuration of callback information. Once the configuration is complete, event callback notifications can be received. For more details, see Callback Configuration.

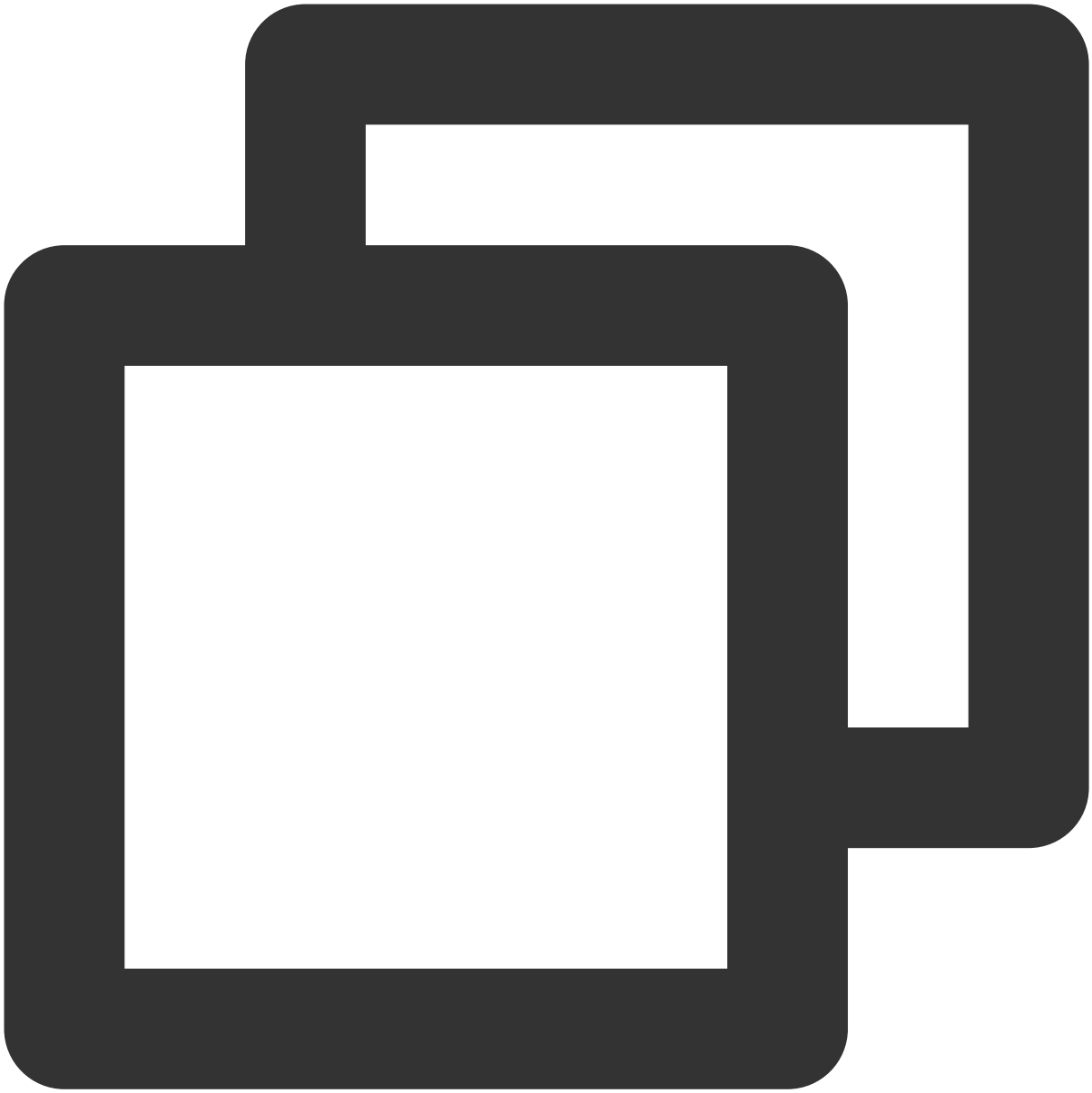
2. Receive and parse callback event packages, pay attention to events 103/104/105, and count the real-time online anchor list in the current room. For more details, see Callback Event.

103

104

105

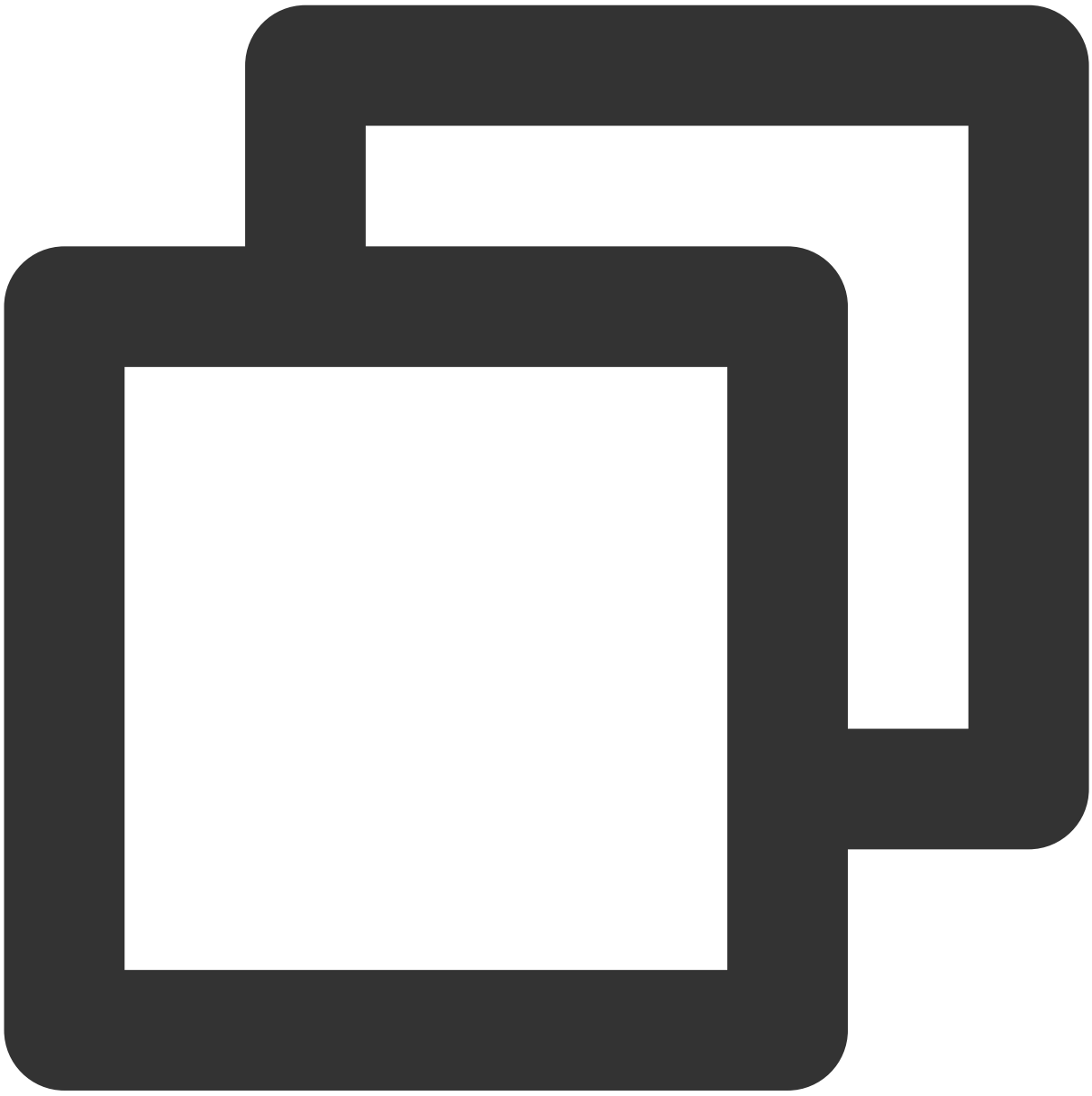
©2013-2022 Tencent Cloud. All rights reserved.



```
{
        "EventGroupId": 1,                          #Room event group
        "EventType":    103,                        #Enter room event
        "CallbackTs":   1687679847972,              #Callback time, in milliseconds
        "EventInfo":    {
                "RoomId":       "123456",                   #Room ID
                "EventTs":      1687679847,                 #Event occurrence time, i
                "EventMsTs":  1687679847899,            #Event occurrence time, in mi
                "UserId":  "1a99b0a9",                  #Username
                "Role":  20,                            #User role 20:Anchor; 21:Audi
        "TerminalType": 2,                      #Terminal Type
```

©2013-2022 Tencent Cloud. All rights reserved.

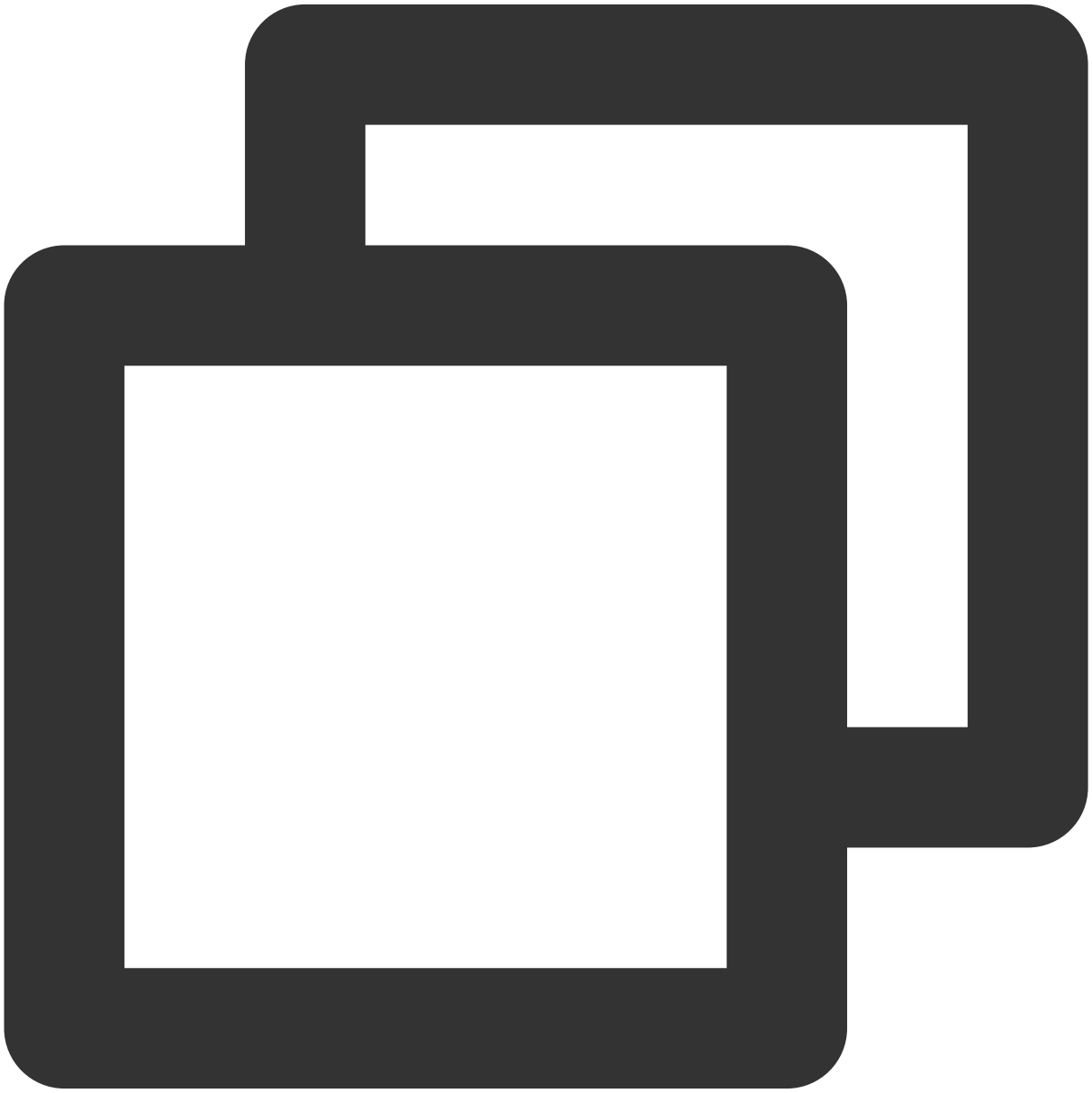
```
        "UserType": 3,                          #User Type
        "Reason": 1                             #Specific reasons
        }
}
```



```
{
        "EventGroupId": 1,                      #Room event group
        "EventType":    104,                    #Exit room event
        "CallbackTs":   1687679847972,          #Callback time, in milliseconds
        "EventInfo":    {
                "RoomId":       "123456",               #Room ID
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                "EventTs":      1687679847,              #Event occurrence time, i
                "EventMsTs":  1687679847899,          #Event occurrence time, in mi
                "UserId":  "1a99b0a9",                #Username
                "Role":  20,                          #User role 20:Anchor; 21:Audi
        "Reason": 1                           #Specific reasons
        }
}
```



```
{
        "EventGroupId": 1,                        #Room event group
        "EventType":    105,                      #Switch role event
```

```
        "CallbackTs":   1687679847972,            #Callback time, in milliseconds
        "EventInfo":    {
                "RoomId":       "123456",               #Room ID
                "EventTs":      1687679847,             #Event occurrence time, i
                "EventMsTs":  1687679847899,          #Event occurrence time, in mi
                "UserId":  "1a99b0a9",                #Username
                "Role":  20                           #User role 20: Anchor; 21: Au
        }
}
```

Note:

105-Switch Role Event is only triggered by changes in the user role after entering the room. Therefore, you also need to supplement the user role list based on the initial role information in **103-Enter Room Event**, as well as update the user role list according to **104-Exit Room Event**, to maintain a more accurate list of room user roles.

3. Periodically compare the seat list and the real-time TRTC anchor list for each room, identify ganchor mics, and mute or remove them accordingly.

Anti-Stutter Solution When Switching on and off the Mic

Problem Description

Due to differences in the mechanisms of mobile device systems, the performance of switching on and off the mic in the voice chat scenario may differ between Android and iOS. On iOS devices, there may be brief audio stutters when switching on and off the mic.

Cause Analysis

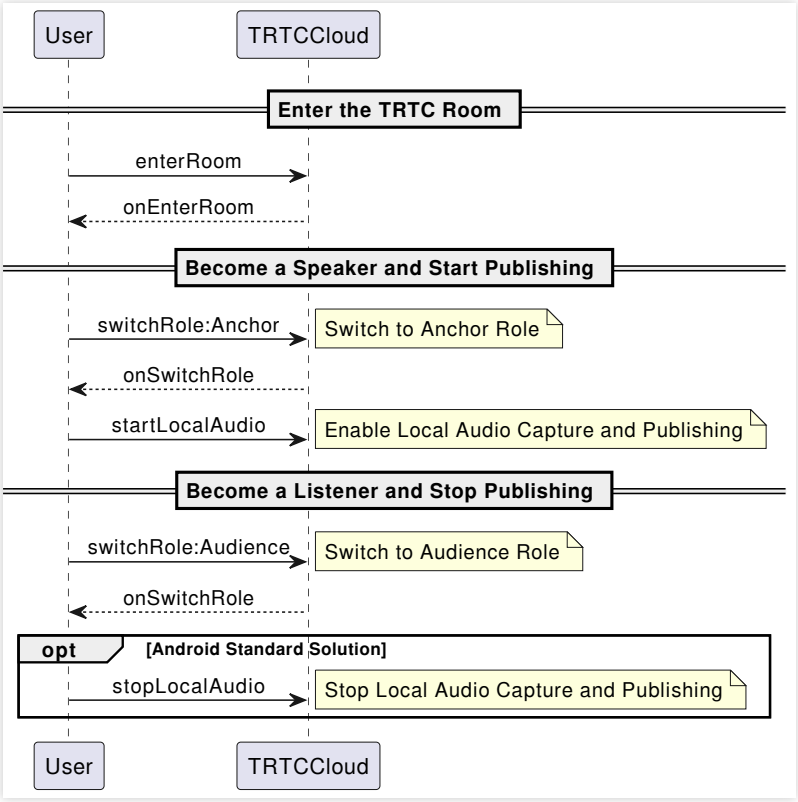
This is related to the iOS system's audio mechanism. The `startLocalAudio` and `stopLocalAudio` operations access and release the microphone device permissions, respectively. SDK's audio re-capturing causes the AVAudioSession to restart the audio driver, resulting in a temporary audio stutter when switching on and off the mic.

Solutions

The typical sequence of switching on and off the mic in TRTC is shown in the figure below. Switching roles simultaneously starts or stops local audio capture and publishing. This solution works normally on the Android platform.

On iOS, during the mic off operation, it is possible to stop streaming simply by switching the audience role, without the need to call `stopLocalAudio` to stop audio capture and release mic permissions, thereby avoiding audio stutters during mic on/off.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

In the anti-stutter solution, not calling `stopLocalAudio` will keep the mic in a continuous capturing state, which may lead to user misunderstanding.

Best Practices for Audio Configuration

In audio configuration, audio quality and volume type are two distinct concepts. In TRTC, audio quality can be set during the enabling of local audio capture and publishing by using `startLocalAudio(TRTCAudioQuality)` or `setAudioQuality(TRTCAudioQuality)` to individually set audio quality; volume type is determined by a combination of factors such as the room entry scenario and audio quality settings. Additionally, it can be forcibly specified through `setSystemVolumeType(TRTCSystemVolumeType)` .

Best Practices for Audio Quality Configuration

The TRTC SDK provides three finely tuned audio quality modes to meet the diverse audio quality needs of various vertical scenarios.

Audio Quality Mode	Audio Quality Enumeration Values	Audio Quality Parameters	Audio Quality Explanation
Voice Mode	TRTCAudioQualitySpeech	Sampling Rate: 16 k; Mono; Encoding Bitrate: 16 kbps	It has strong network resilience and performs well in poor network environments, making it suitable for applications primarily

©2013-2022 Tencent Cloud. All rights reserved.

			focused on voice communication, such as online meetings, voice calls, etc.
Default Mode	TRTCAudioQualityDefault	Sampling Rate: 48k; Mono; Encoding Bitrate: 50 kbps	The default SDK settings. It offers better fidelity for music than the Voice Mode, while also transmitting much less data volume than the Music Mode. This makes it versatile and suitable for various scenarios.
Music Mode	TRTCAudioQualityMusic	Sampling Rate: 48 k; Full-Band Stereo; Encoding Bitrate: 128 kbps	Under this mode, the audio transmission consumes a significant data volume, ensuring that the music signal achieves high fidelity in detail restoration across all frequency bands, suitable for scenarios requiring high-quality music transmission.

As can be seen from the table, from Voice Mode to Music Mode, the audio quality effect improves, but the data volume of audio transmission also increases.
In the scenario of a voice chat room, it is recommended to choose the Voice Mode for pure voice communication, which can achieve better smoothness under weak network conditions.
For voice chat rooms with a need to play background music, it is recommended to choose the Default Mode or Music Mode to achieve good audio detail restoration.
Considering the bandwidth pressure on downstream audience networks, to ensure a good user experience, it is advisable to use the Music Mode cautiously in business scenarios with ten or more seats.
Note:
TRTC audio quality supports dynamic adjustment, which means audio quality can be dynamically adjusted during the streaming process by calling `setAudioQuality(TRTCAudioQuality)` .

Best Practices for Volume Type Configuration

The TRTC SDK provides three control modes for system volume types to meet the different needs of volume types in various scenarios.

Volume Type Mode	Volume Type Mode Enumerated Values	Volume Type Mode Description
Full Call Volume	TRTCSystemVolumeTypeVOIP	The advantage of this solution is that the audio module does not need to switch working modes during mic on/off, enabling seamless mic switching. It is suitable for applications where users frequently switch mics. If the scenario selected upon entering the room is TRTCAppSceneVideoCall or TRTCAppSceneAudioCall, the SDK will automatically use this mode.

©2013-2022 Tencent Cloud. All rights reserved.

Automatic Switching Mode	TRTCSystemVolumeTypeAuto	Also known as Voice Call on Mic, Media Off Mic. This mode ensures that the anchor uses call volume when on the mic, while the off-mic audience uses media volume, suitable for live streaming scenarios. If the scenario selected upon entering the room is TRTCAppSceneLIVE or TRTCAppSceneVoiceChatRoom, the SDK will automatically use this mode.
Full Media Volume	TRTCSystemVolumeTypeMedia	Use Media Volume for the entire call. It is suitable for music scenarios where demanding audio quality is required. If your users mainly use external devices (such as external sound cards), this mode can be adopted.

In call scenarios, it is recommended to choose the default Full Call Volume, where the audio module does not need to switch;
In voice chat room scenarios, it is recommended to use the default Automatic Switching mode for pure voice communication, that is, Voice Call on Mic, Media Off Mic.
Voice chat rooms that need background music can consider setting the volume to Full Media Volume throughout to avoid users perceiving stuttering or sudden volume changes in music when going on and off the mic.
Note:
If you need to specify a volume type, it is recommended to call `setSystemVolumeType` once after entering the room and before starting to stream. Do not call it during the mic on/off.
Call Volume supports the phone's built-in AEC feature and allows audio pickup via the mic on Bluetooth headphones, but the disadvantage is the audio quality is relatively average.
Media Volume does not support the phone's built-in AEC feature and does not support audio pickup via the mic on Bluetooth headphones, but it has better music playback performance.

Single-Stream Volume Evaluation

In voice chat room scenarios, some customers may opt to push and pull RTC single streams for speakers, while pulling mixed streams from the room for audiences, aiming to save bandwidth costs. However, in voice chat room scenarios, it is usually necessary to provide UI prompts based on the volume level of the speakers, such as sound waves or volume bars. While volume evaluation and feedback for single-channel audio is straightforward to implement in TRTC rooms, achieving this in audio-only mix streams requires specialized techniques. Below are the specific implementations of two solutions.

Single-Stream Volume Evaluation in RTC Room

Step One: Enable Volume Prompts

Enable the volume callback through the `enableAudioVolumeEvaluation` API, and optionally enable the local voice detection feature. After enabling this feature, the SDK will provide feedback in the `onUserVoiceVolume`

©2013-2022 Tencent Cloud. All rights reserved.

callback about the volume of both local users and remote streaming users, the maximum volume value, as well as the local voice detection result.

Note:

Starting from TRTC SDK version 10.2, the local voice detection feature has been added. Once enabled, the local voice detection result will be displayed in `TRTCVolumeInfo.vad` (for users in the anchor role). Operations such as `muteLocalAudio` and `setAudioCaptureVolume(0)` will not affect the voice detection result, making it convenient to remind users to turn on their mics.

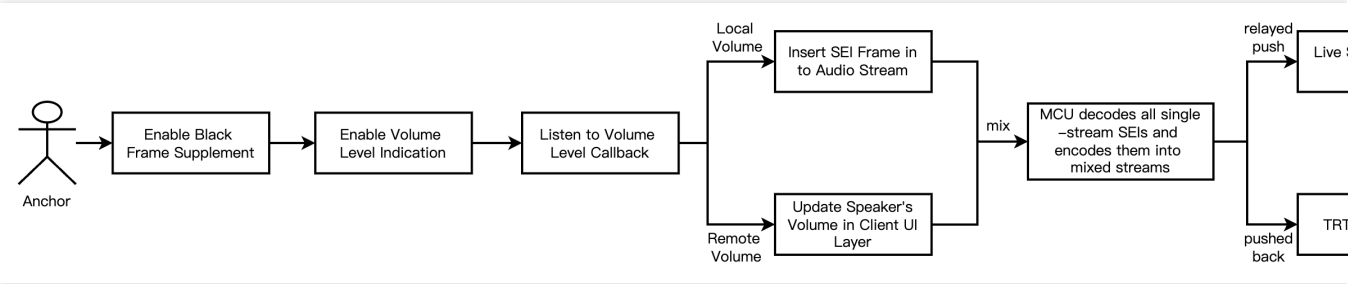
Step Two: Listen to the Volume Callback

Listen to the `onUserVoiceVolume` callback in `TRTCCloudListener`. This callback provides information on the volume levels of both local and remote users' streams, as well as the maximum volume value of remote users. Based on these volume levels, you can adjust the UI to display corresponding voice waveforms.

Note:

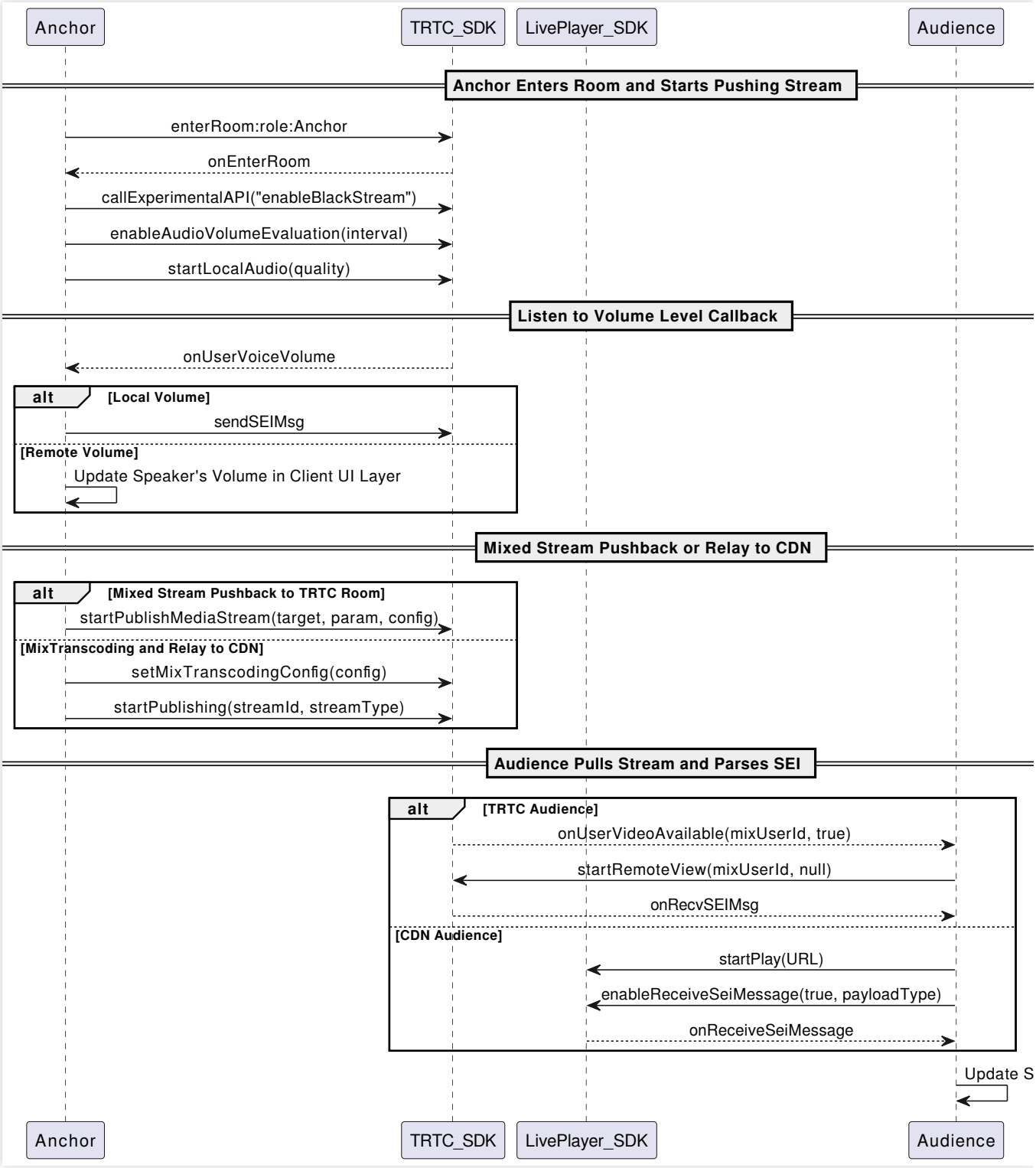
Rendering voice waveform animations for speakers can be determined by the volume level in the onUserVoiceVolume callback. The activation and deactivation of voice waveform animations (user's mic on/off state) are recommended to be determined based on the onUserAudioAvailable callback.

Evaluation of Single Stream Volume in Audio-Only Mixed Stream



The implementation process of evaluating single stream volume in audio-only mixed stream is shown in the diagram above. Speakers need to listen for volume level callback and determine both local and remote volume levels. Then, insert the local volume value and user information into the audio stream in the form of SEI messages. After mixing, these messages are transmitted to listeners. **Alternatively, the room owner can send out speakers' callback volume values through SEI messages**. The diagram below shows the sequence diagram of the entire process:

©2013-2022 Tencent Cloud. All rights reserved.



Note:

If there is a requirement for mixing and relaying to CDN while transmitting SEI:

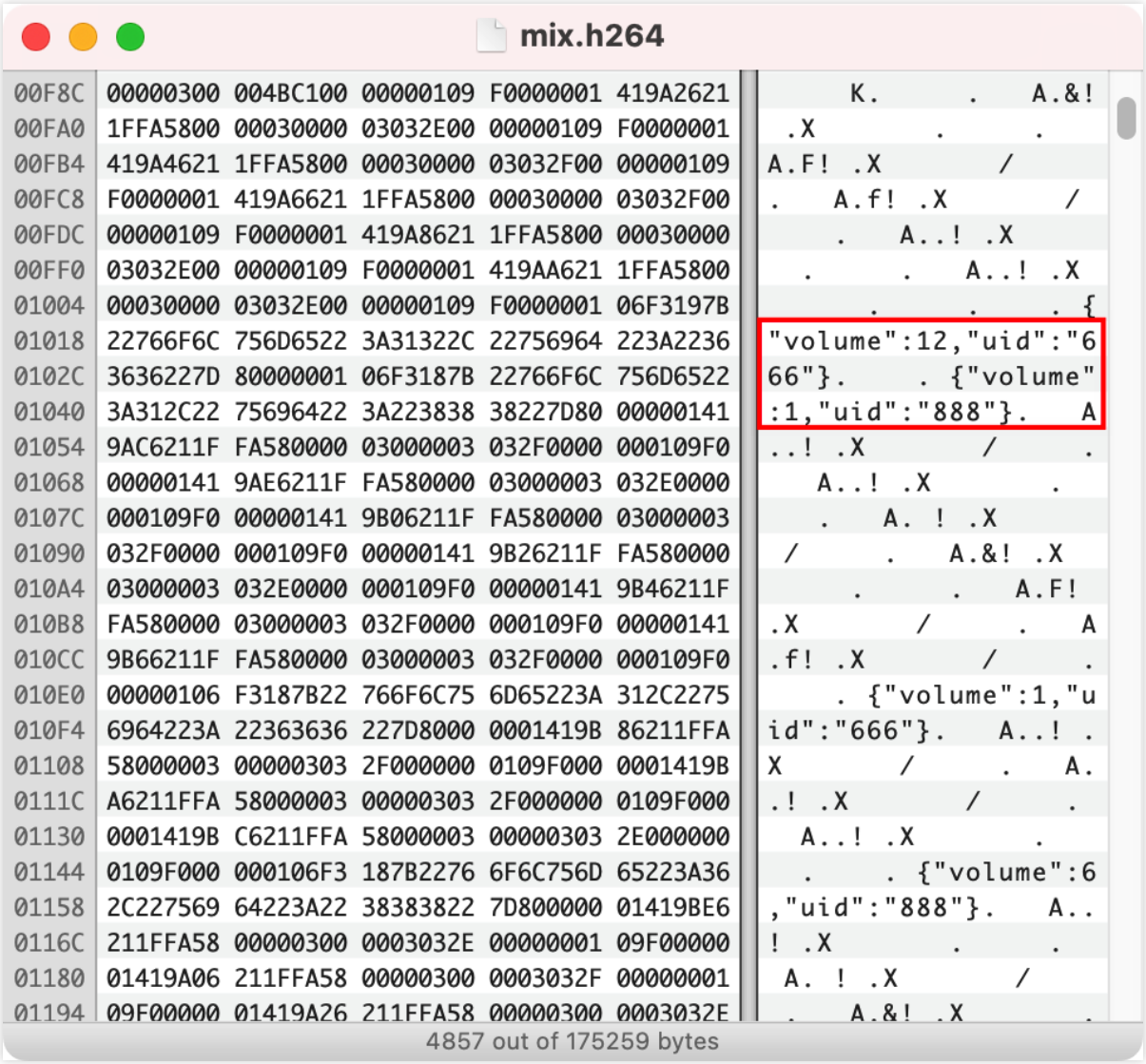
The room entry scenario must be set to LIVE and cannot be set to pure audio entry, otherwise SEI messages will not be transmitted.

If the mixing API adopts `setMixTranscodingConfig`, then the mixing mode cannot use the `PureAudio` audio-only mode.

©2013-2022 Tencent Cloud. All rights reserved.

If the mixing API adopts `startPublishMediaStream`, then the media stream transcoding configuration parameters must carry the `TRTCVideoLayout` parameter.

As shown below, the audience will see the volume levels of the respective speakers in the SEI messages parsed from the mixed stream.



Scenario Gameplay

In the voice chat room scenario, the room owner and several speakers interact online through voice, and there may also be listeners who cannot speak but only listen and interact through sending gifts and chat messages. Different room themes are usually set to attract users with similar interests for viewing and interaction. Common themes include FM radio, Karaoke chat, game interaction, and sports event streaming.

©2013-2022 Tencent Cloud. All rights reserved.

FM Radio Room

There may be a solo live broadcast by an anchor or a host with several fixed chatting guests, while background music and sound effects are played simultaneously. Listeners can request to speak by giving gifts to participate in voice interaction.

This scenario typically involves a large number of audiences, with infrequent mic switching. It is suitable to use the solution of speakers pushing and pulling RTC streams while the audience pulls CDN mixed streams. When audiences become speakers, they switch from the RTMP channel to the TRTC channel to enter the room and stream in real time. This solution balances real-time interaction with cost.

Note:

For this scenario, it is recommended to use the solution of speakers pushing and pulling RTC streams while the audience pulls CDN mixed streams.

When audiences switch between the RTMP and RTC protocol for mic-connecting, ensure a smooth transition between on-mic and off-mic states.

Karaoke Voice Chat Room

Usually, there is one administrator, and everyone can select songs, comment, guess songs, continue singing, etc. It mainly consists of two models: Multi-Person Co-Anchoring and Multi-Person Mic Rotation. In the Multi-Person Co-Anchoring mode, one person sings while other co-anchoring users can listen and speak simultaneously, but the lead singer cannot hear the other speakers. However, the audience in the room can hear all the voices. The Multi-Person Mic Rotation mode allows a person to sing a portion of a song, after which it automatically transitions to the next person; meanwhile, other users can only listen and comment during the waiting period, without participating in voice chat.

Online Karaoke scenarios require high synchronization and allow audiences to join in singing at any time, making it suitable to use the solution of the speaker pushing and pulling RTC streams while the audience pulls RTC mixed streams. Here, a mixing robot is needed to initiate the mixing command and push the mixed stream back to the TRTC room for the listeners to pull and watch.

Note:

For this scenario, it is recommended to use the solution of the speaker pushing and pulling RTC streams while the audience pulls RTC mixed streams.

For specific technical details and precautions regarding the implementation of Karaoke Voice Chat Rooms, please refer to Online Karaoke Scenario Solutions.

Interactive Gaming Room

In scenarios like Werewolf, Murder Mystery, Dubbing, Truth or Dare, and Draw and Guess, rooms are created based on the game's progression, and the speaking permissions of players are controlled in sequence according to the game's progress.

In interactive gaming scenarios, the number of participants is typically limited, and there is a need for frequent joining and leaving of the mic for gaming purposes. This scenario is suitable for the conventional approach of having the

©2013-2022 Tencent Cloud. All rights reserved.

speaker pull and push RTC streams while the audience pulls RTC single streams. Game participants can request to speak at any time or choose to mute themselves until their character dies, forcing them to become listeners or exit the room.

Note:

This scenario recommends the solution of the speaker pulling and pushing RTC streams while the audience pulls RTC single streams.

Interactive gaming rooms usually include the playing of local game audio effects, so attention must be paid to AEC processing and the selection of volume types.

Supporting Products for the Solution

System Level	Product Name	Application Scenarios
Access Layer	Tencent Real-Time Communication (TRTC)	Provides low-latency, high-quality real-time interactive live streaming solutions for multi-person voice interaction, serving as the foundational capability for voice chat scenarios.
Access Layer	Instant Messaging (IM)	Provides room management and seat management capabilities based on group features, enables the sending and receiving of rich media messages such as live streaming room-wide messaging, public screen messages, as well as custom signaling and other communication needs.
Cloud Services	Cloud Streaming Services (CSS)	Provides real-time audio and video relayed push, along with accelerated media stream distribution services, as well as additional capabilities such as recording and pornography detection.
Cloud Services	Video on Demand (VOD)	For audio-video media, it offers an integrated high-quality media service that includes production and upload, storage, transcoding, media processing, media AI, accelerated distribution and playback, and copyright protection
Data Storage	Cloud Object Storage (COS)	Provides storage services for audio recording files and audio slicing files.

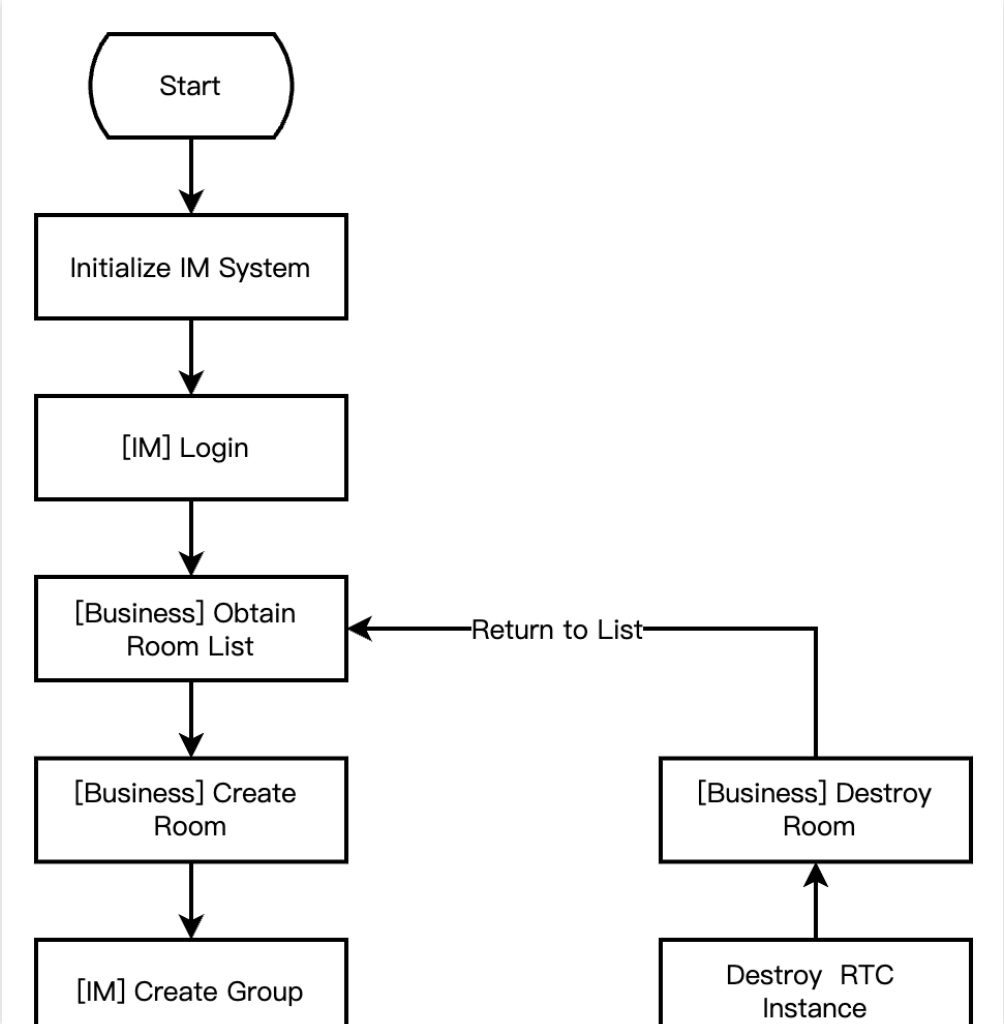
©2013-2022 Tencent Cloud. All rights reserved.

Quick Integration Guide
Android

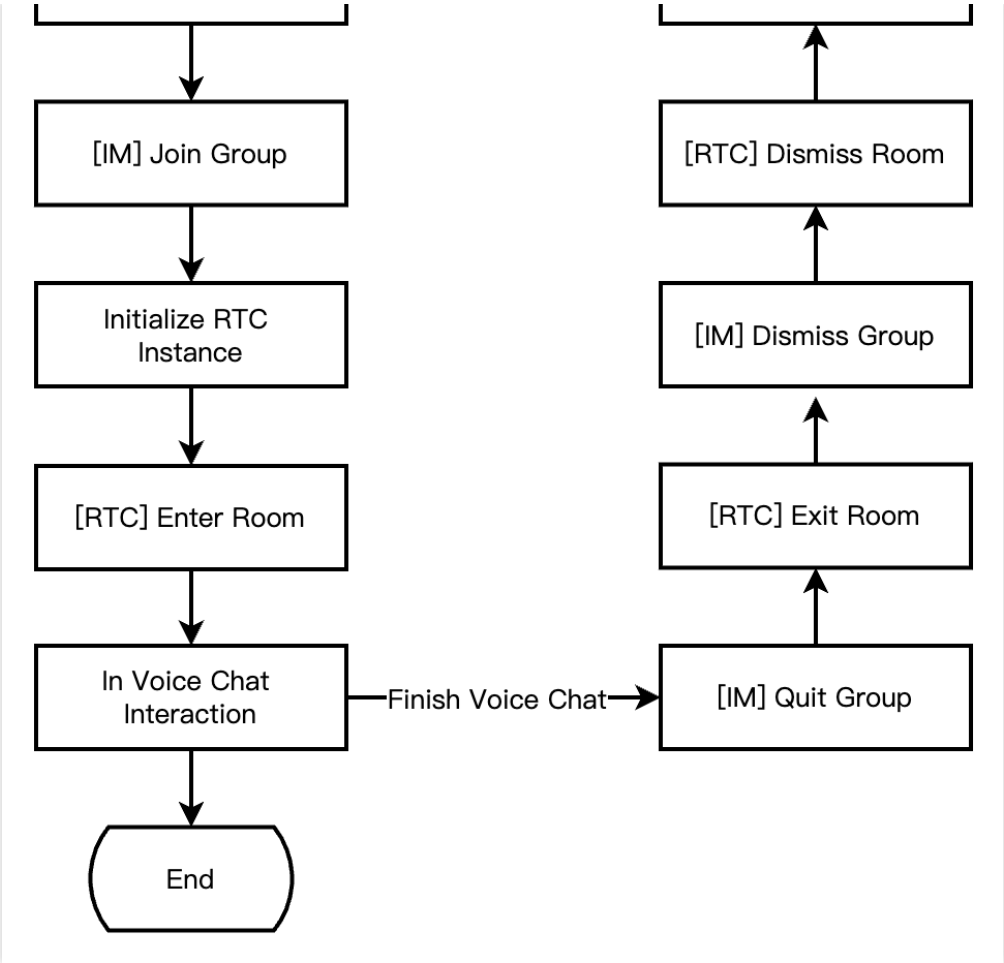
Last updated：2024-07-18 14:26:14

Business Process

This section summarizes some common business processes in voice chat rooms to help you better understand the entire scenario implementation process.
Room management process
Room owner seat management process
Audience seat management process
The following figure shows the room management process, including the creation, joining, exiting, and dissolvement of rooms.

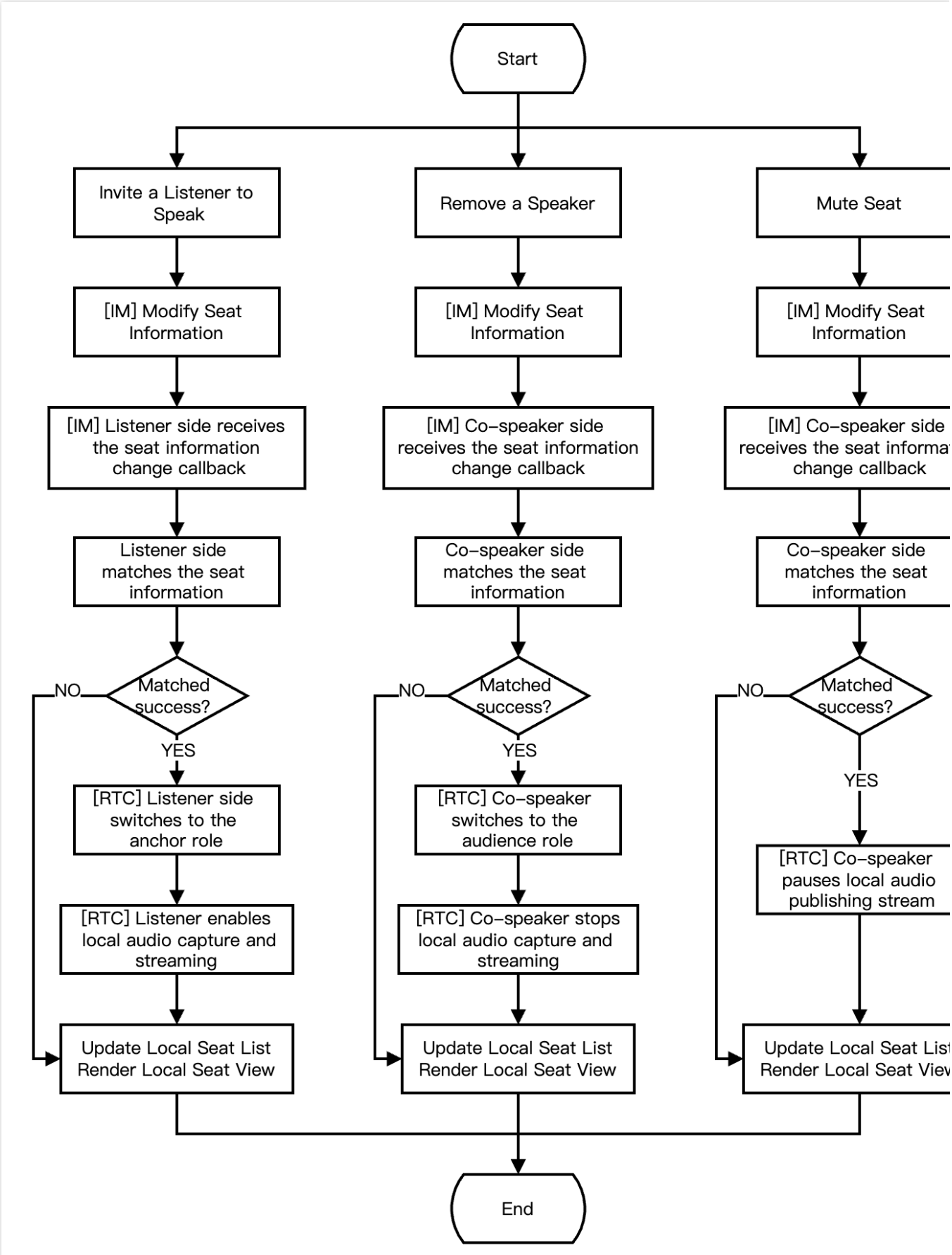


©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the room owner seat management process, including inviting a listener to speak, removing a speaker, and muting a seat.

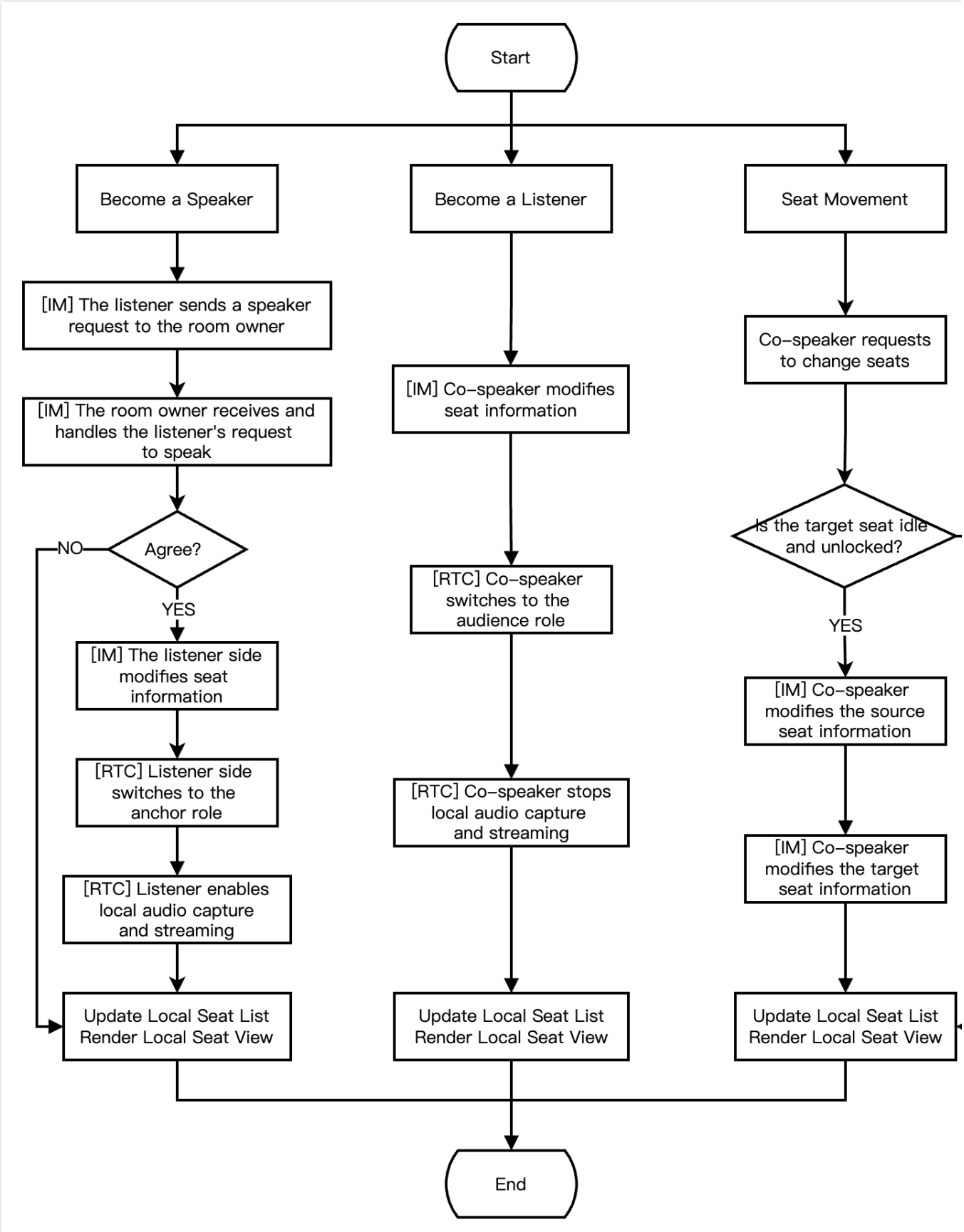
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

The following figure shows the audience seat management process, including becoming a speaker, become a listener, and moving a seat.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

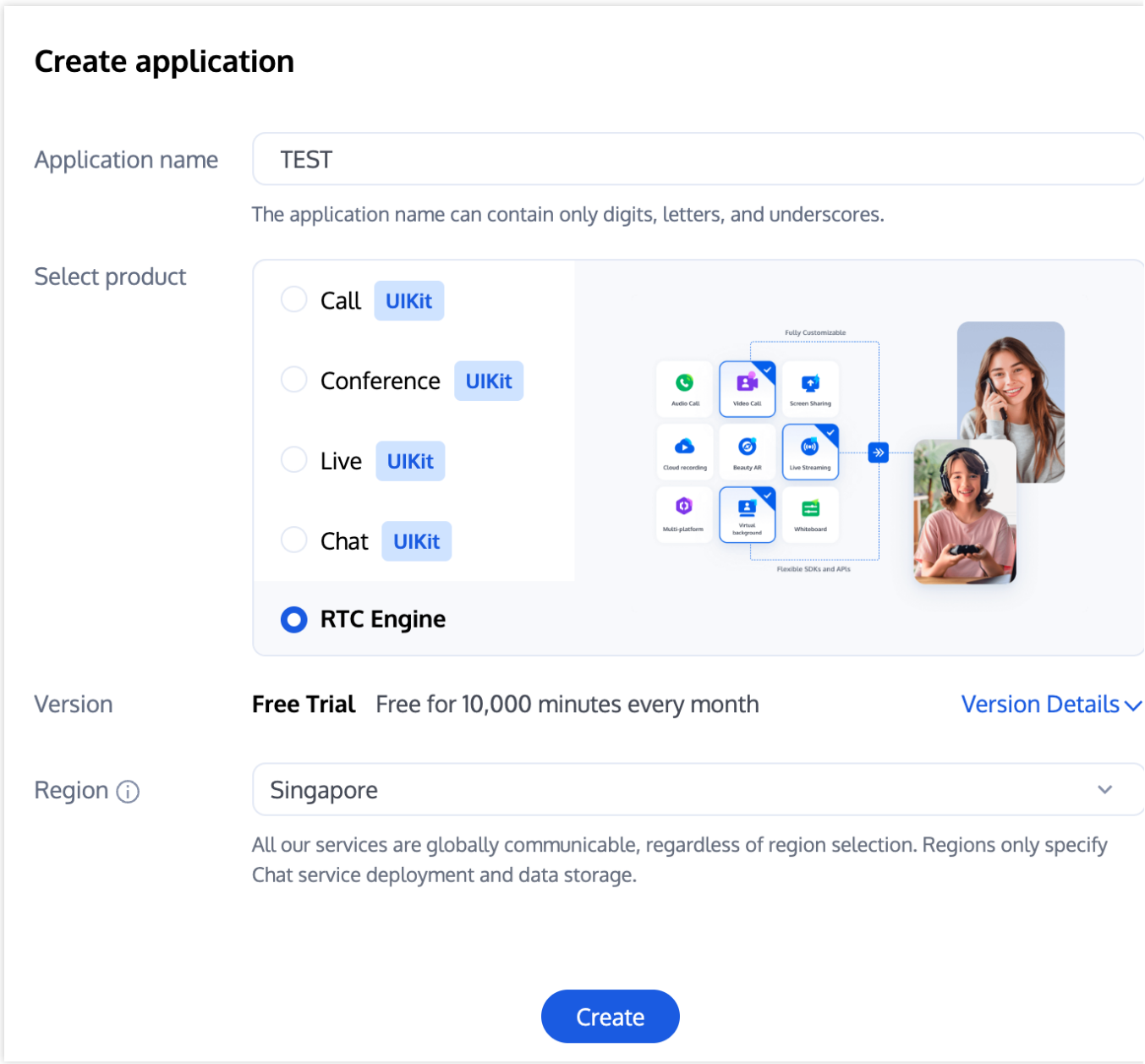
Integration Preparation

Step 1. Activating the service.

Voice chat room scenarios usually require dependencies on two paid PaaS services from Tencent Cloud, Instant Messaging (IM) and Tencent Real-Time Communication (TRTC) for construction.

1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. At this time, in the Instant Messaging (IM) console, an IM experience edition application with the same SDKAppID as the current TRTC application will be automatically created. The account and authentication system of the two can be reused. Subsequently, you can choose to upgrade the TRTC or IM application version as needed. For example, advanced versions can unlock more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



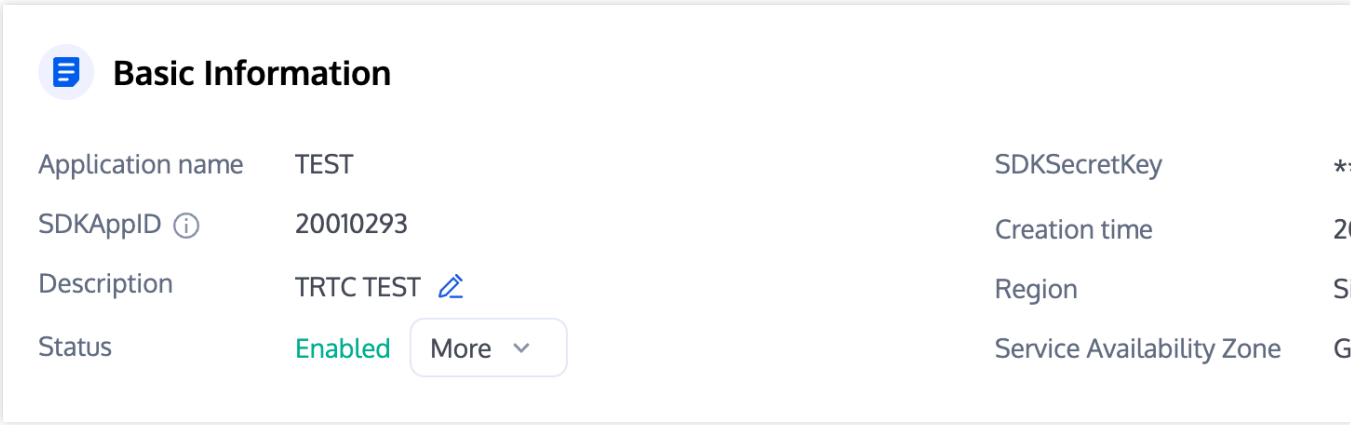
Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

2. After an application is created, you can see the basic information of the application in the **Application Management - Application Overview** section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.

©2013-2022 Tencent Cloud. All rights reserved.

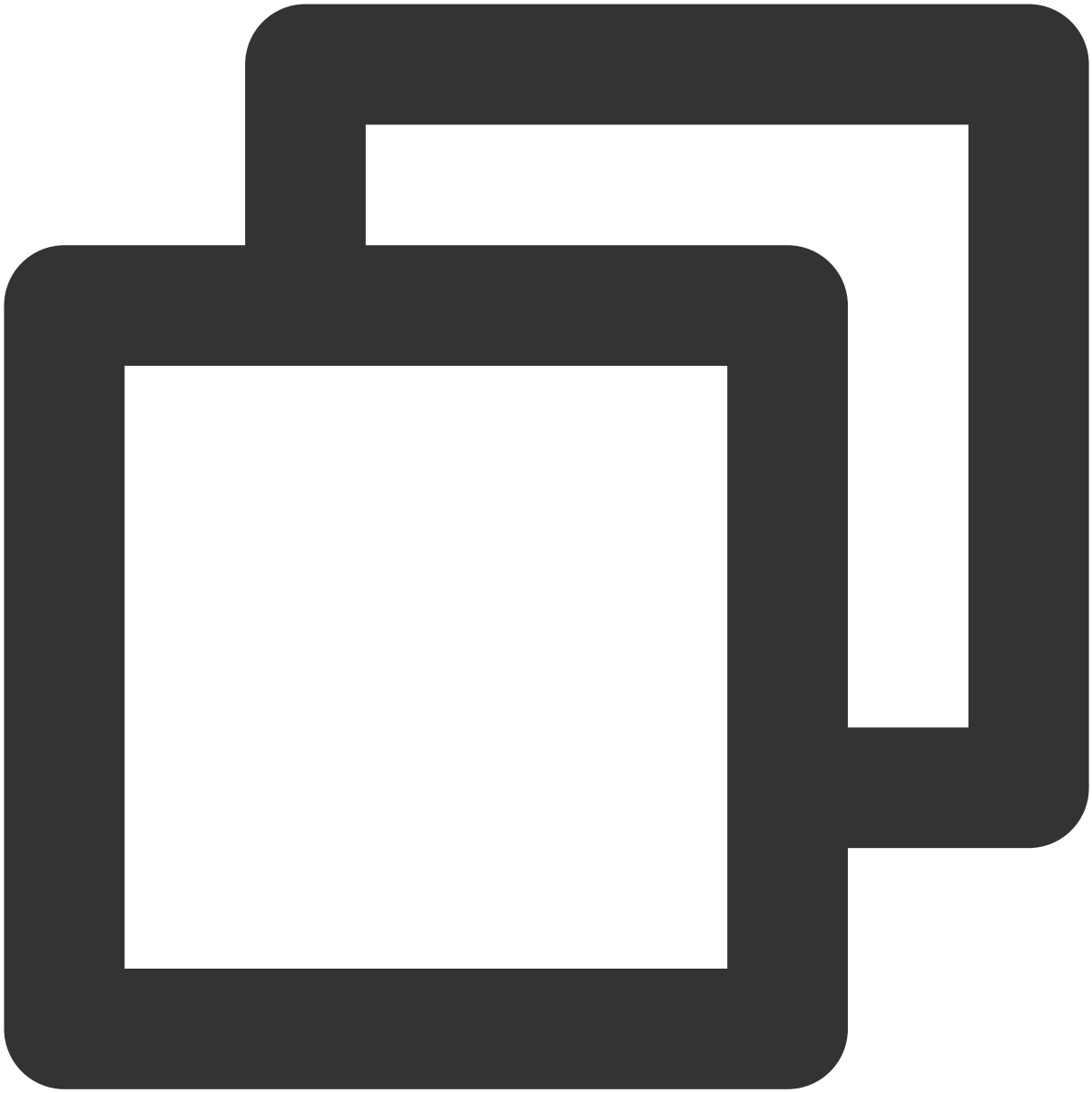


Step 2: Importing SDK.

The TRTC SDK and IM SDK have been released to the **mavenCentral** repository. You can configure Gradle to download and update automatically.

1. Add the dependency for the appropriate version of the SDK in dependencies.

©2013-2022 Tencent Cloud. All rights reserved.



```
dependencies {
    // TRTC SDK Simplified Edition. It has two features of the TRTC and live stream
    implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'

    // Add IM SDK. Entering the latest Version No. is recommended.
    implementation 'com.tencent.imsdk:imsdk-plus:Version number'

    // If you need to add the Quic plugin, uncomment the next line (Note: Version N
    // implementation 'com.tencent.imsdk:timquic-plugin:Version number'
}
```

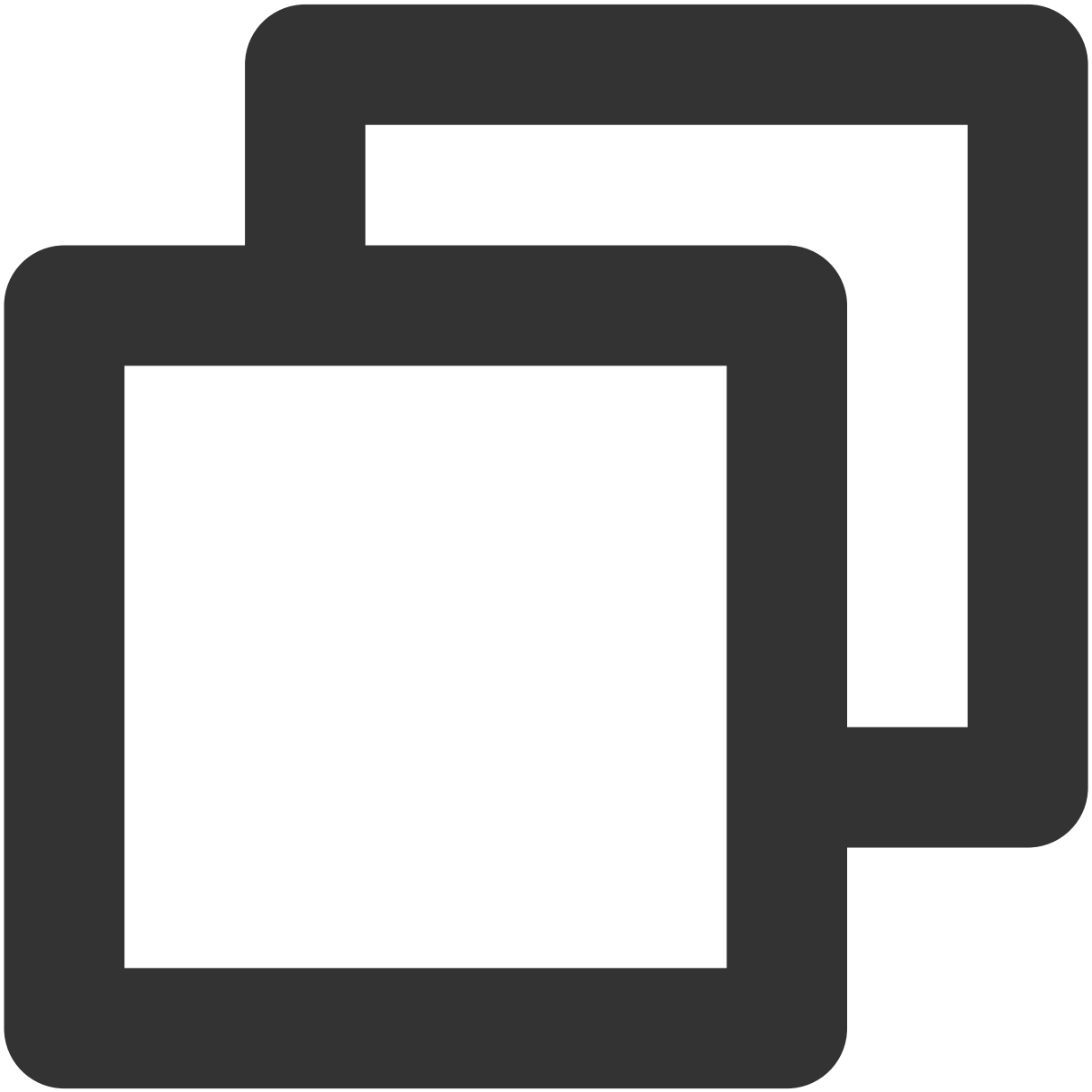
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manually Integrating IM SDK.

Quic plugin offers axp-quic Multiplexing Transmission Protocol, providing better resistance to poor networks. Even with a packet loss rate of 70%, it still can offer services. **Available only for Flagship users**. For non-Flagship users, purchase the Flagship package before use, and see Pricing Instructions. To ensure proper functionality, update **Terminal SDK** to **version 7.7.5282 or above.**

2. Specify the CPU architecture used by the app in defaultConfig.



```
defaultConfig {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        ndk {
                abiFilters "armeabi-v7a", "arm64-v8a"
        }
    }
```

Note:

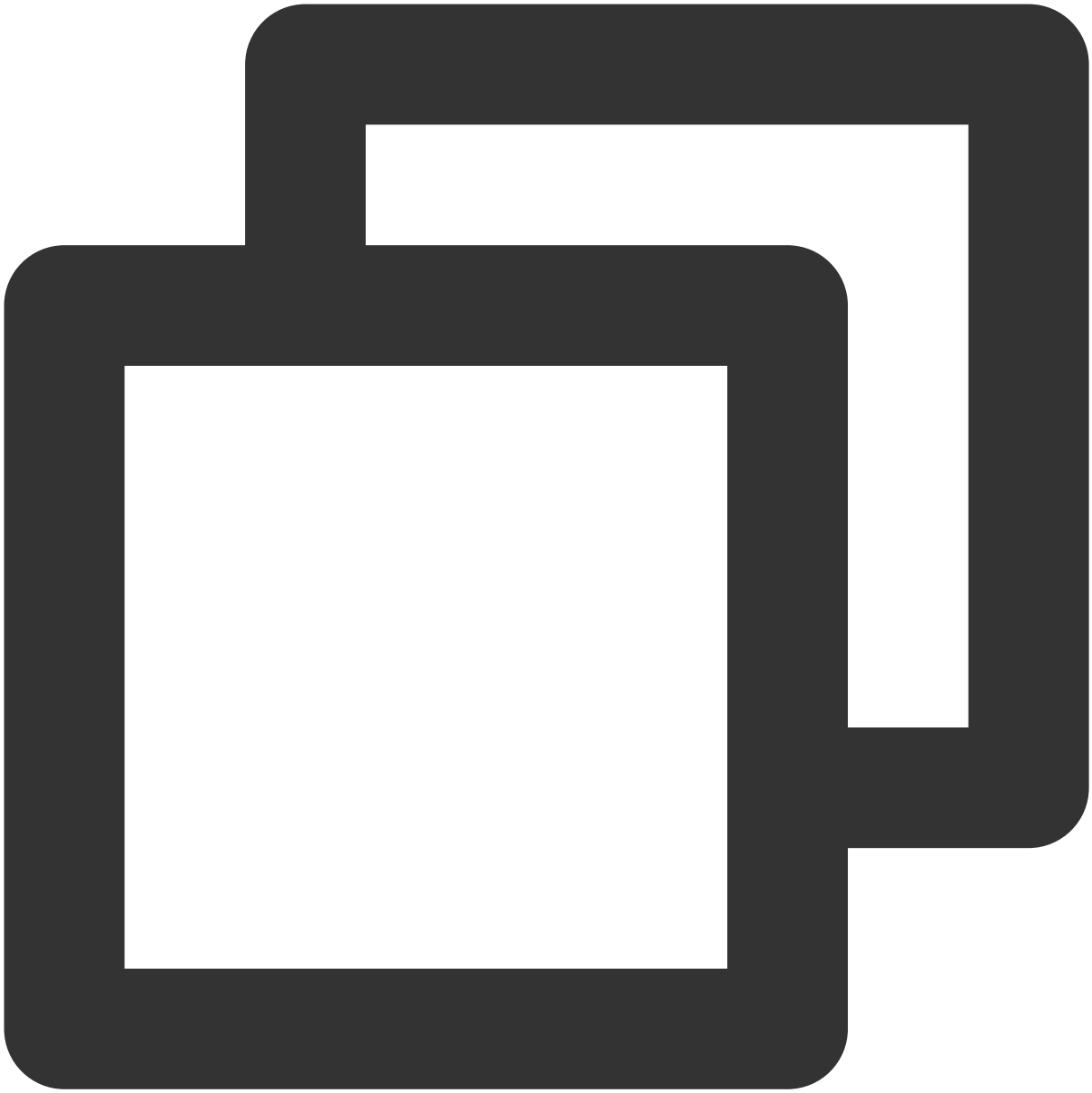
The TRTC SDK supports architectures including armeabi, armeabi-v7a and arm64-v8a. Additionally, it supports architectures for simulators including x86 and x86_64.

The IM SDK supports architectures including armeabi-v7a, arm64-v8a, x86, and x86_64. To reduce the size of the installer package, you can choose to package SO files for only a subset of these architectures.

Step 3: Project configuration.

1. To configure app permissions in AndroidManifest.xml, for voice chat scenarios, both the TRTC SDK and IM SDK require the following permissions:

©2013-2022 Tencent Cloud. All rights reserved.



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

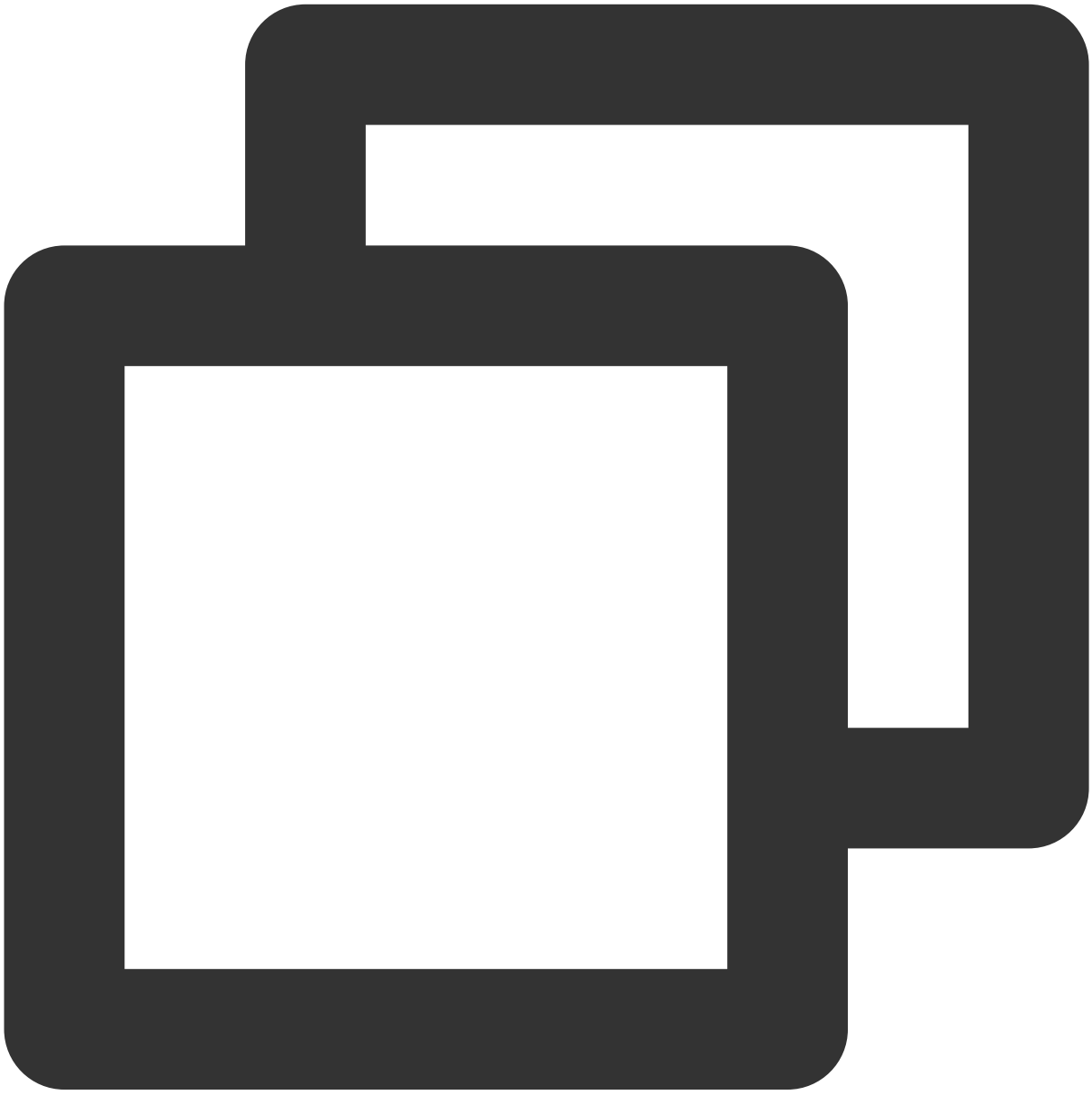
Note:

The TRTC SDK does not have built-in permission request logic. You need to declare the corresponding permissions and features yourself. Some permissions (such as storage and recording), also require runtime dynamic requests.

©2013-2022 Tencent Cloud. All rights reserved.

If the Android project's `targetSdkVersion` is 31 or higher, or if the target device runs Android 12 or a newer version, the official requirement is to dynamically request `android.permission.BLUETOOTH_CONNECT` permission in the code to use the Bluetooth feature properly. For more information, see Bluetooth Permissions.

2. Since we use Java's reflection features inside the SDK, you need to add relevant TRTC SDK classes to the non-obfuscation list in the proguard-rules.pro file:



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

©2013-2022 Tencent Cloud. All rights reserved.

Integration Process

Step 1: Generate authentication credentials.

UserSig is a security protection signature designed by Tencent Cloud. Its purpose is to prevent malicious attackers from misappropriating your cloud service usage rights. Tencent Cloud's Tencent Real-Time Communication (TRTC) and Instant Messaging (IM) services both implement this security mechanism. TRTC authentication when entering a room, and IM authentication when logging in.

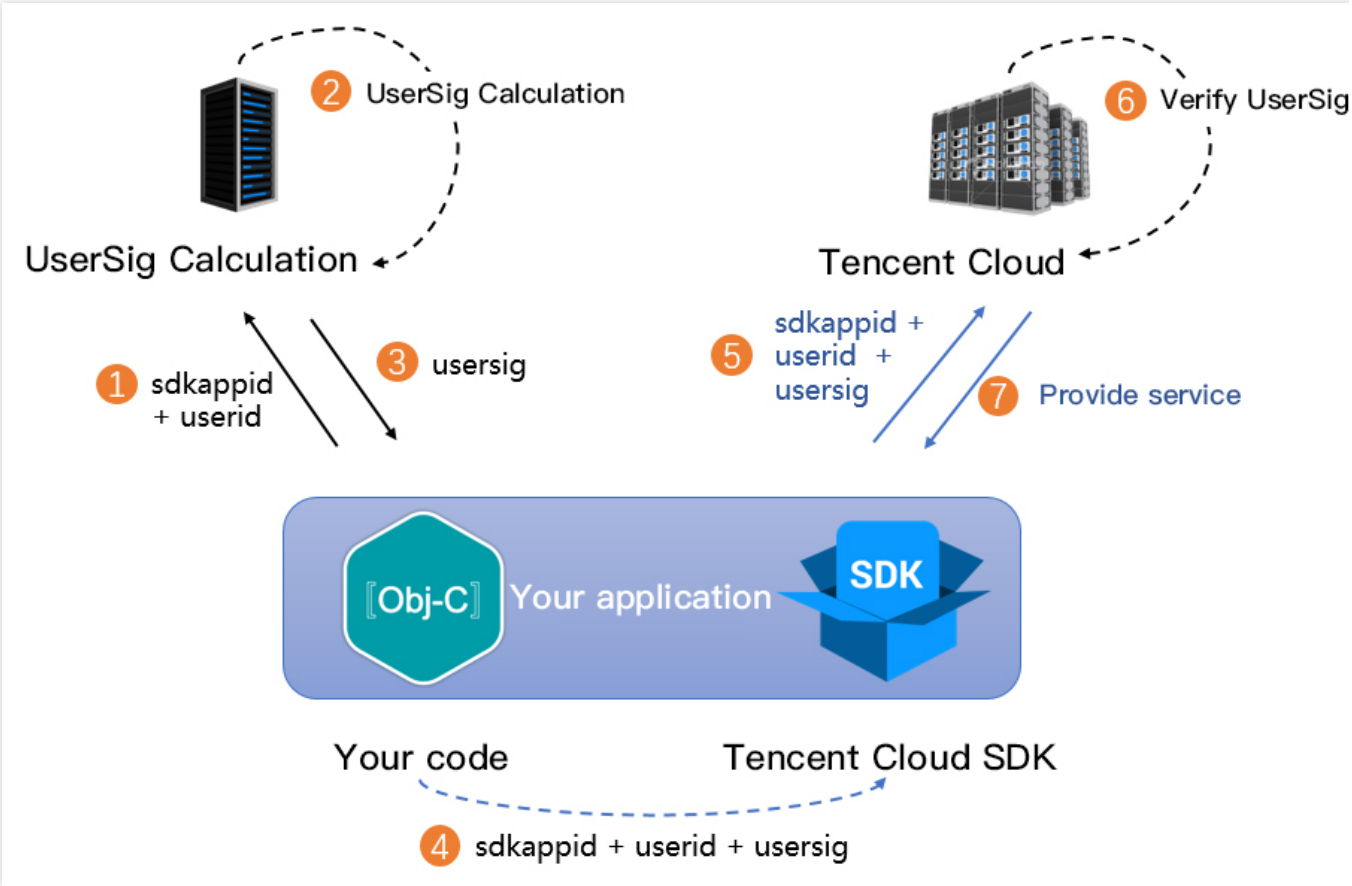
Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.

Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.

The specific implementation process is as follows:

1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.
3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. Once the verification is passed, it will provide instant communication services to the IM SDK and Tencent Real-Time Communication (TRTC) services to the TRTC SDK.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

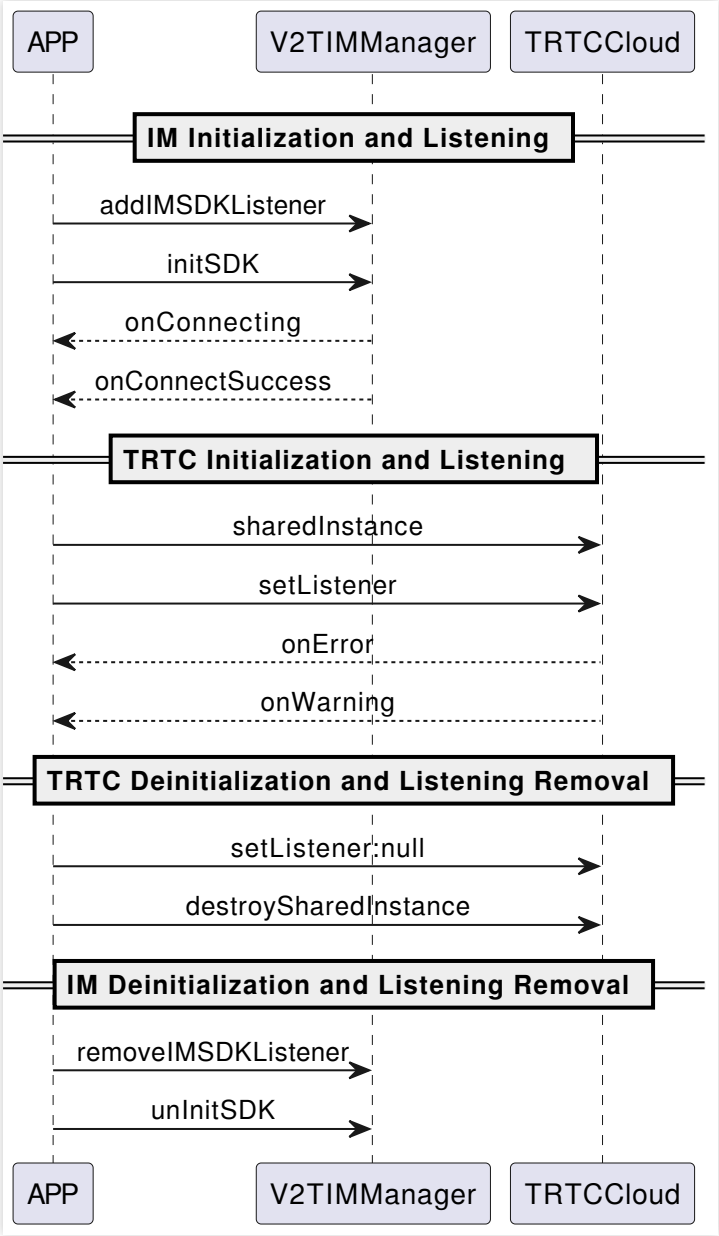
The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Step 2: Initialization and listening.

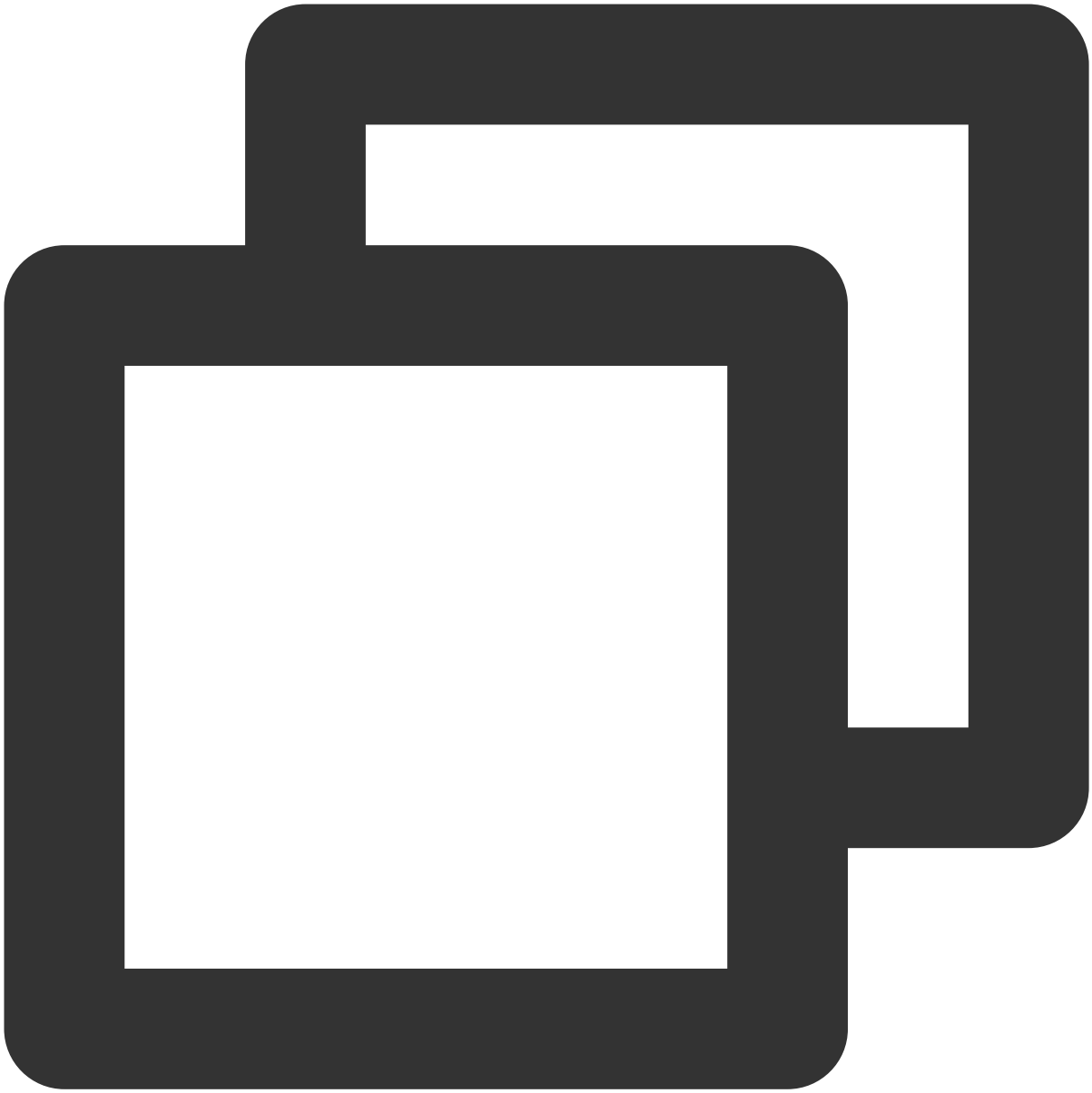
Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.



1. Initialize the IM SDK and add event listeners.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Add event listener.
V2TIMManager.getInstance().addIMSDKListener(imSdkListener);
// Initialize the IM SDK. After calling this API, you can immediately call the log-
V2TIMManager.getInstance().initSDK(context, sdkAppID, null);

// After the SDK is initialized, it will trigger various events, such as connection
private V2TIMSDKListener imSdkListener = new V2TIMSDKListener() {
    @Override
    public void onConnecting() {
        Log.d(TAG, "IM SDK is connecting to Tencent cloud service");
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

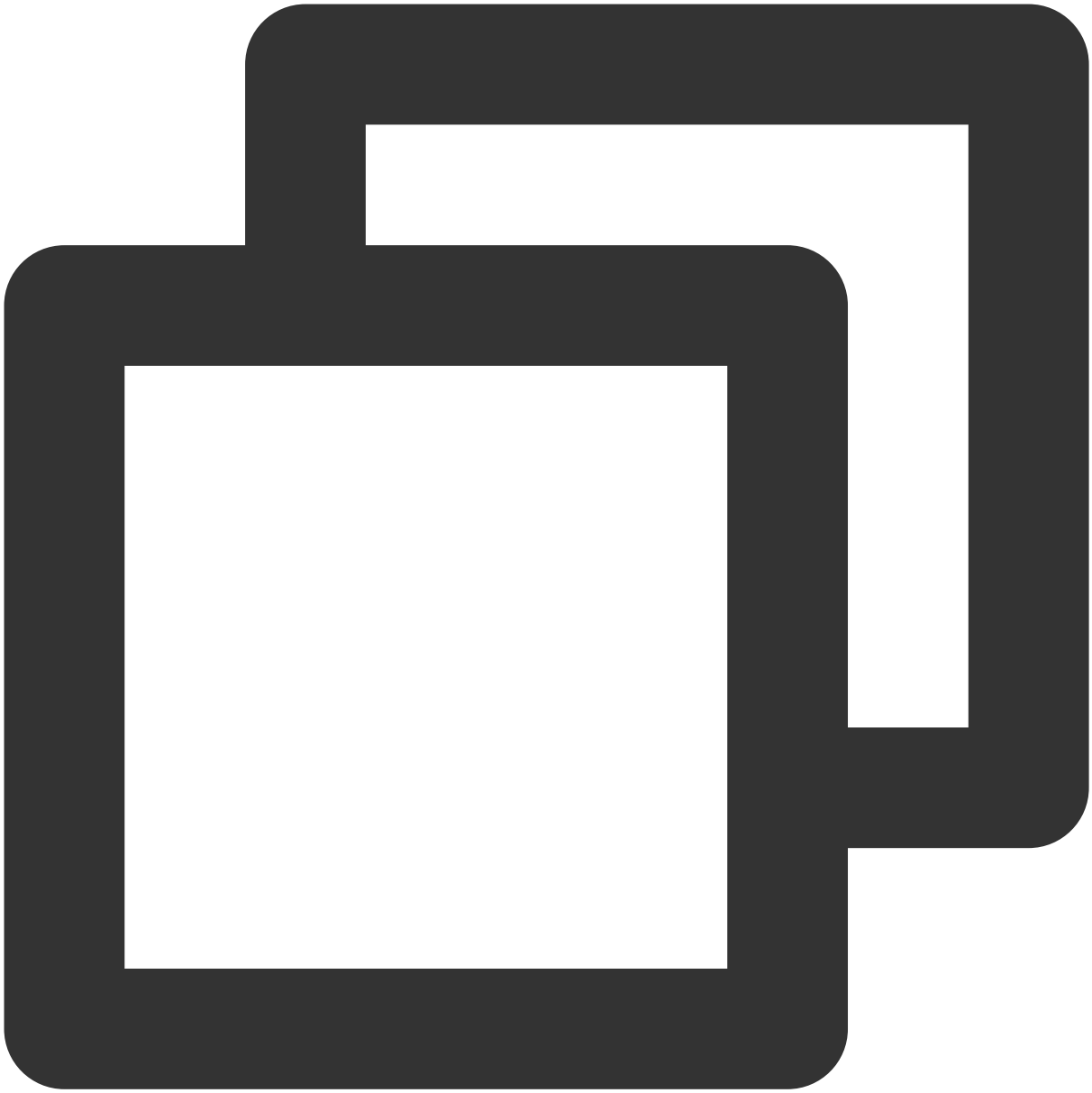
```
    @Override
    public void onConnectSuccess() {
        Log.d(TAG, "IM SDK has successfully connected to Tencent cloud service");
    }
};

// Remove event listener.
V2TIMManager.getInstance().removeIMSDKListener(imSdkListener);
// Deinitialize the IM SDK.
V2TIMManager.getInstance().unInitSDK();
```

Note:

If your application's lifecycle is consistent with the SDK's lifecycle, you do not need to deinitialize before exiting the application. If you only initialize the SDK after entering a specific interface and no longer use it after exiting, you may deinitialize the SDK.

2. Create TRTC SDK instances and set event listeners.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (Single Instance Pattern).
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Set event listeners.
mTRTCCloud.setListener(trtcSdkListener);

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
private TRTCCloudListener trtcSdkListener = new TRTCCloudListener() {
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        Log.d(TAG, errCode + errMsg);
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        Log.d(TAG, warningCode + warningMsg);
    }
};

// Remove event listener.
mTRTCCloud.setListener(null);
// Terminate TRTC SDK instance (Singleton Pattern).
TRTCCloud.destroySharedInstance();
```

Note:

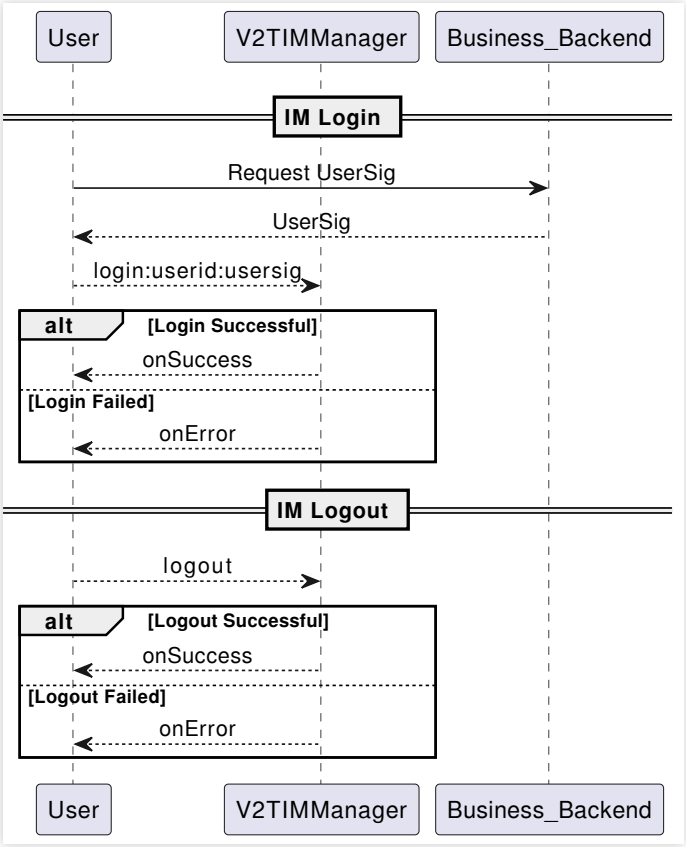
It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

Step 3: Log in and log out.

After initializing the IM SDK, you need to call the SDK log-in API to authenticate your account identity and have permissions to use features. Before using any other features, ensure you are successfully logged in, or you might encounter feature malfunctions or unavailability. If you only need to use TRTC's audio and video services, you can skip this step.

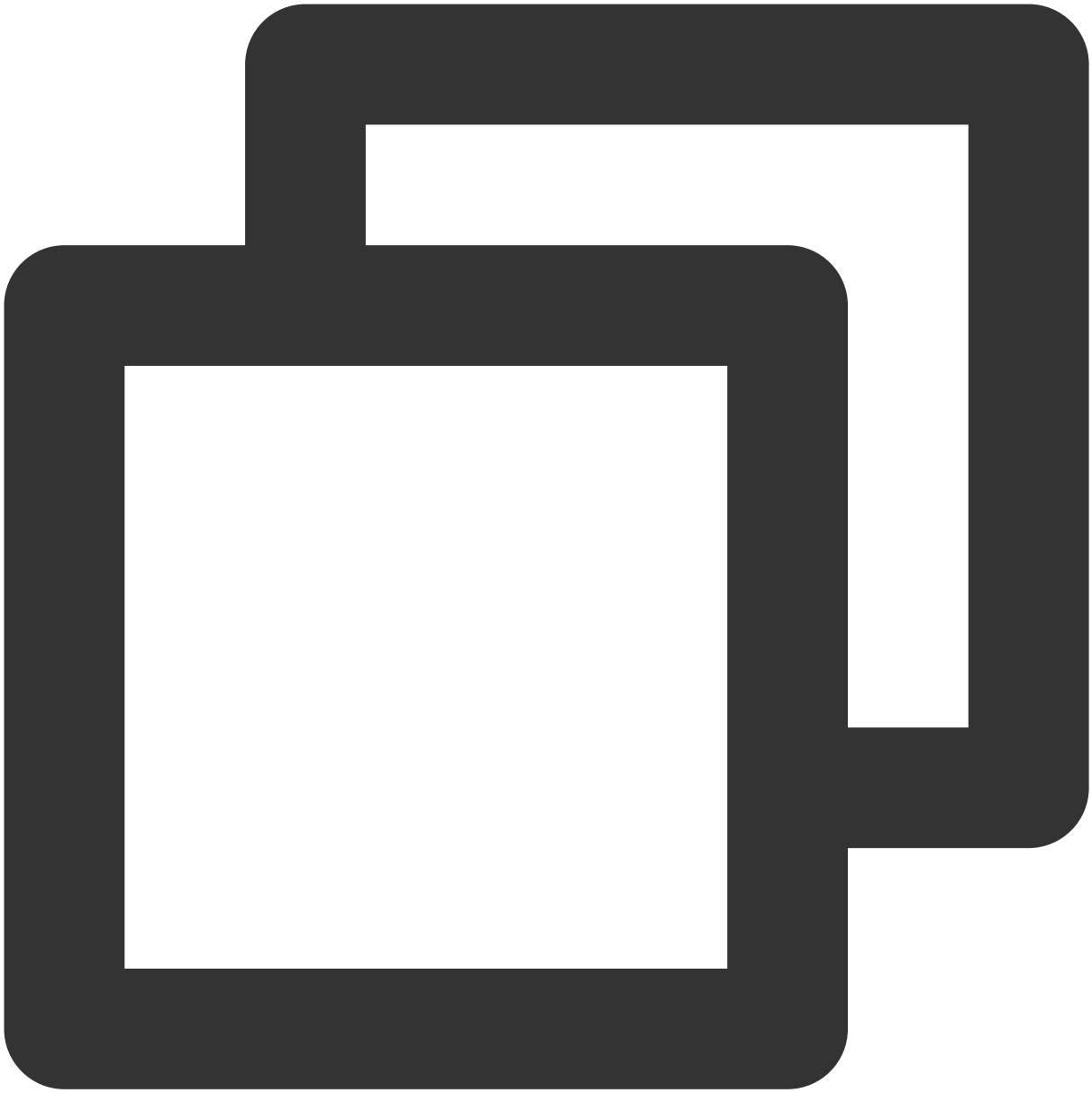
Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.



1. Log in.

©2013-2022 Tencent Cloud. All rights reserved.

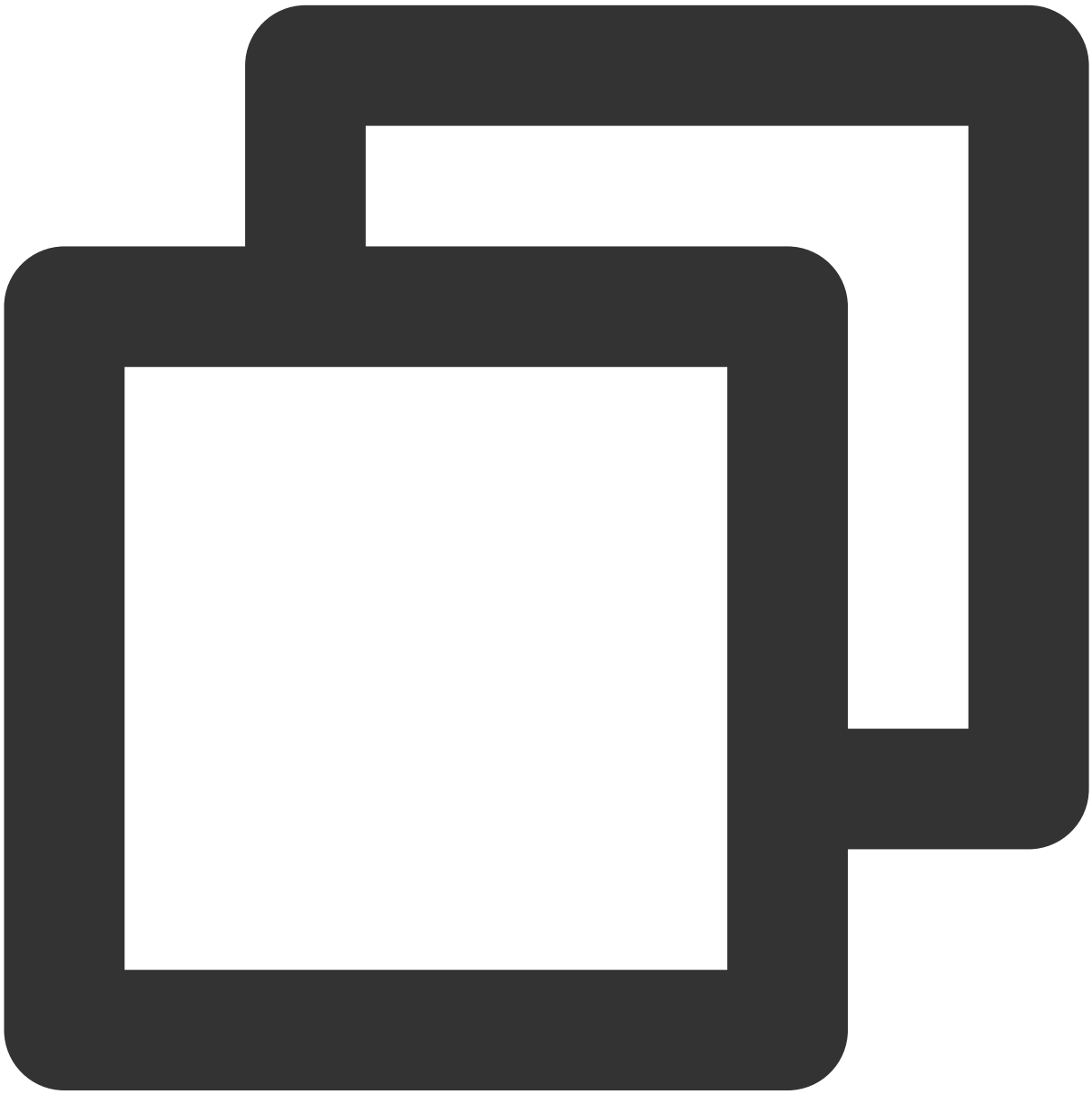


```
// Log in: userID can be defined by the user and userSig can be generated as per St
V2TIMManager.getInstance().login(userID, userSig, new V2TIMCallback() {
    @Override
    public void onSuccess() {
        Log.i("imsdk", "success");
    }

    @Override
    public void onError(int code, String desc) {
        // The following error codes mean an expired UserSig, and you need to gener
        // 1. ERR_USER_SIG_EXPIRED (6206)
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // 2. ERR_SVR_ACCOUNT_USERSIG_EXPIRED (70001)
        // Note: Do not call the log-in API in case of other error codes. Otherwise
        Log.i("imsdk", "failure, code:" + code + ", desc:" + desc);
    }
});
```

2. Log out.



```
// Log out.
V2TIMManager.getInstance().logout(new V2TIMCallback() {
    @Override
    public void onSuccess() {
```

©2013-2022 Tencent Cloud. All rights reserved.

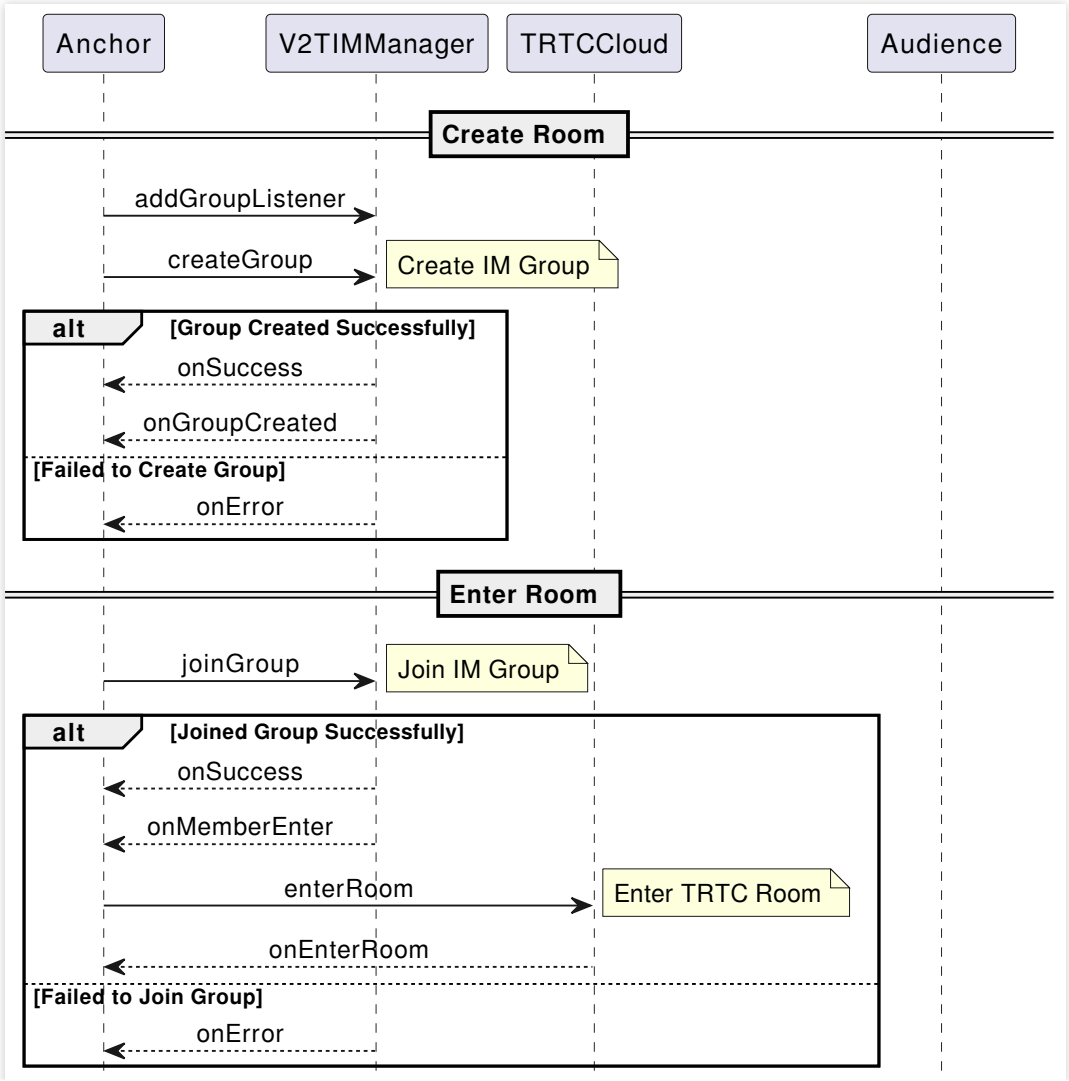
```
            Log.i("imsdk", "success");
        }

        @Override
        public void onError(int code, String desc) {
            Log.i("imsdk", "failure, code:" + code + ", desc:" + desc);
        }
    });
```

Note:

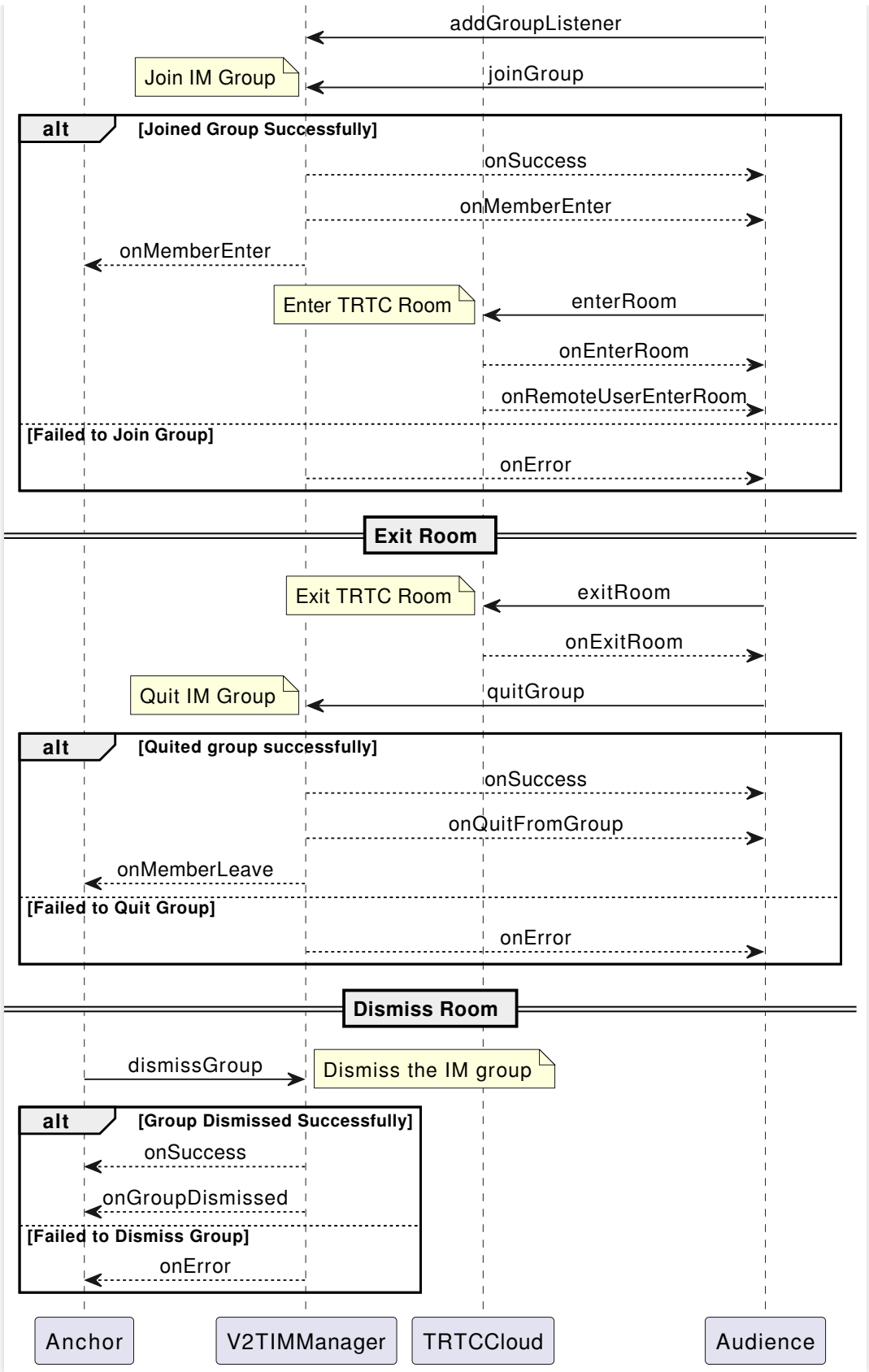
If your application's lifecycle matches the IM SDK's lifecycle, logging out before exiting the application is not necessary. However, if you only use the IM SDK after entering a specific interface and stop using it after exiting, you can log out and deinitialize the IM SDK.

Step 4: Room management.

Sequence diagram



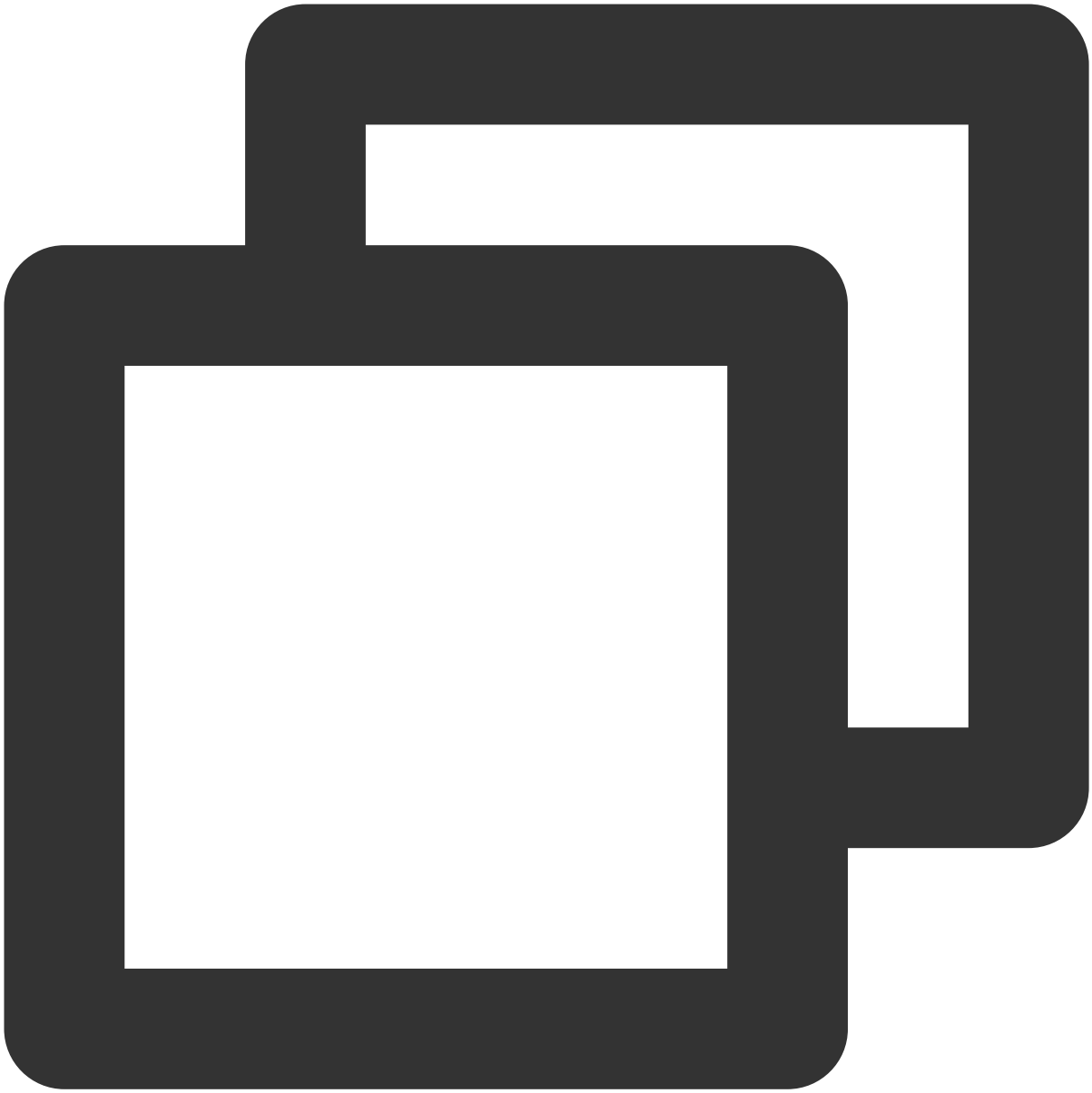
©2013-2022 Tencent Cloud. All rights reserved.



1. Create a room.

When the anchor (room owner) starts live streaming, a room needs to be created. The concept of "room" here corresponds to "group" in IM. This example only shows how to create an IM group on the client, but it is also possible to create a group on the server.

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMManager.getInstance().createGroup(V2TIMManager.GROUP_TYPE_AVCHATROOM, groupID,
  @Override
  public void onSuccess(String s) {
      // Group creation successful
  }

  @Override
  public void onError(int code, String desc) {
      // Group creation failed
  }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Listen for group creation notifications.
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
  @Override
  public void onGroupCreated(String groupID) {
      // Group creation callback. groupID is the ID of the newly created group.
  }
});
```

Note:

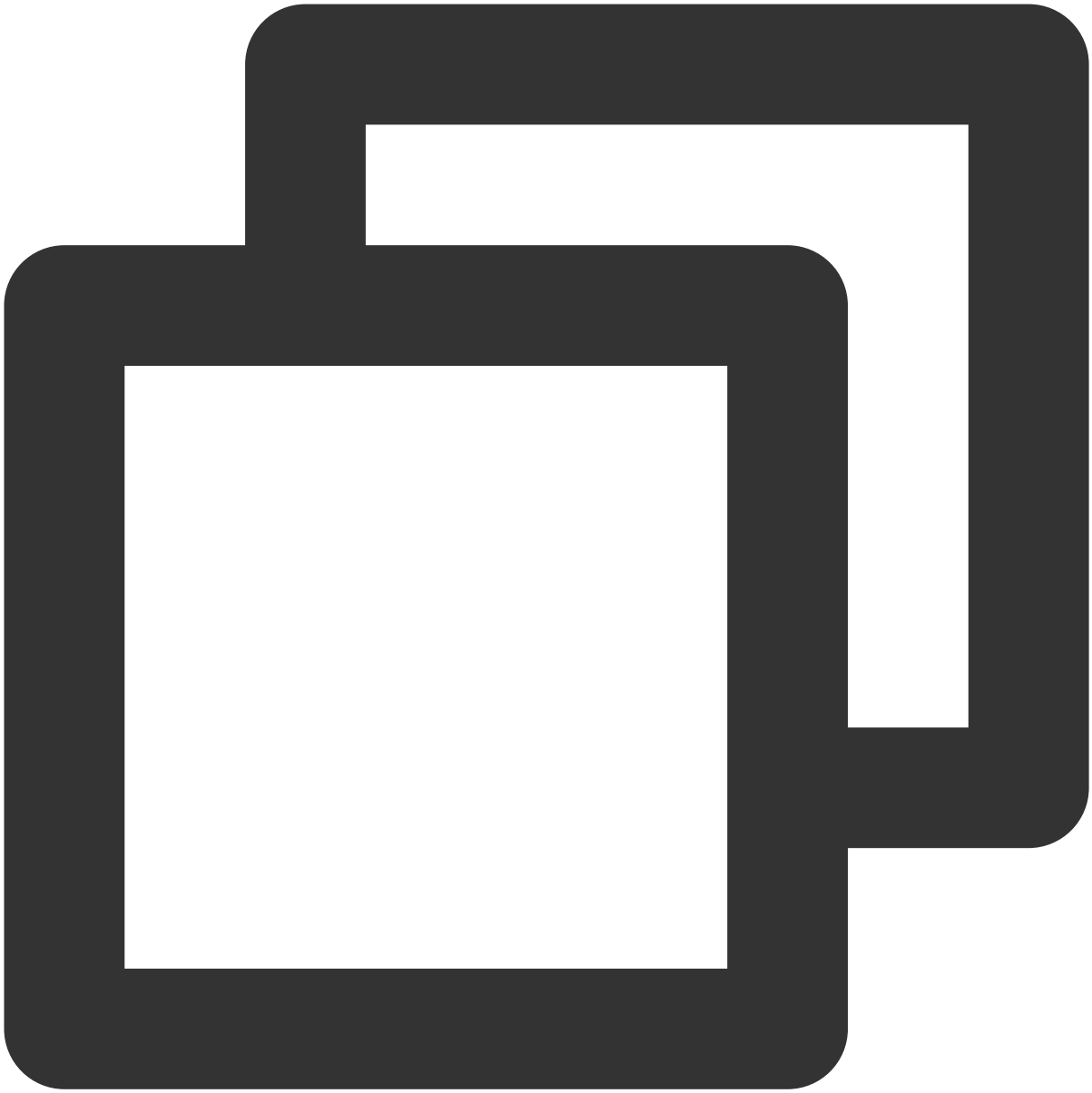
For voice chat room scenarios, when creating IM groups, you need to choose the type of live streaming groups: `GROUP_TYPE_AVCHATROOM`.

TRTC does not have a room-creation API, so when a user attempts to join a room that does not exist, the backend automatically creates a room.

2. Join a room.

Join IM group.

©2013-2022 Tencent Cloud. All rights reserved.

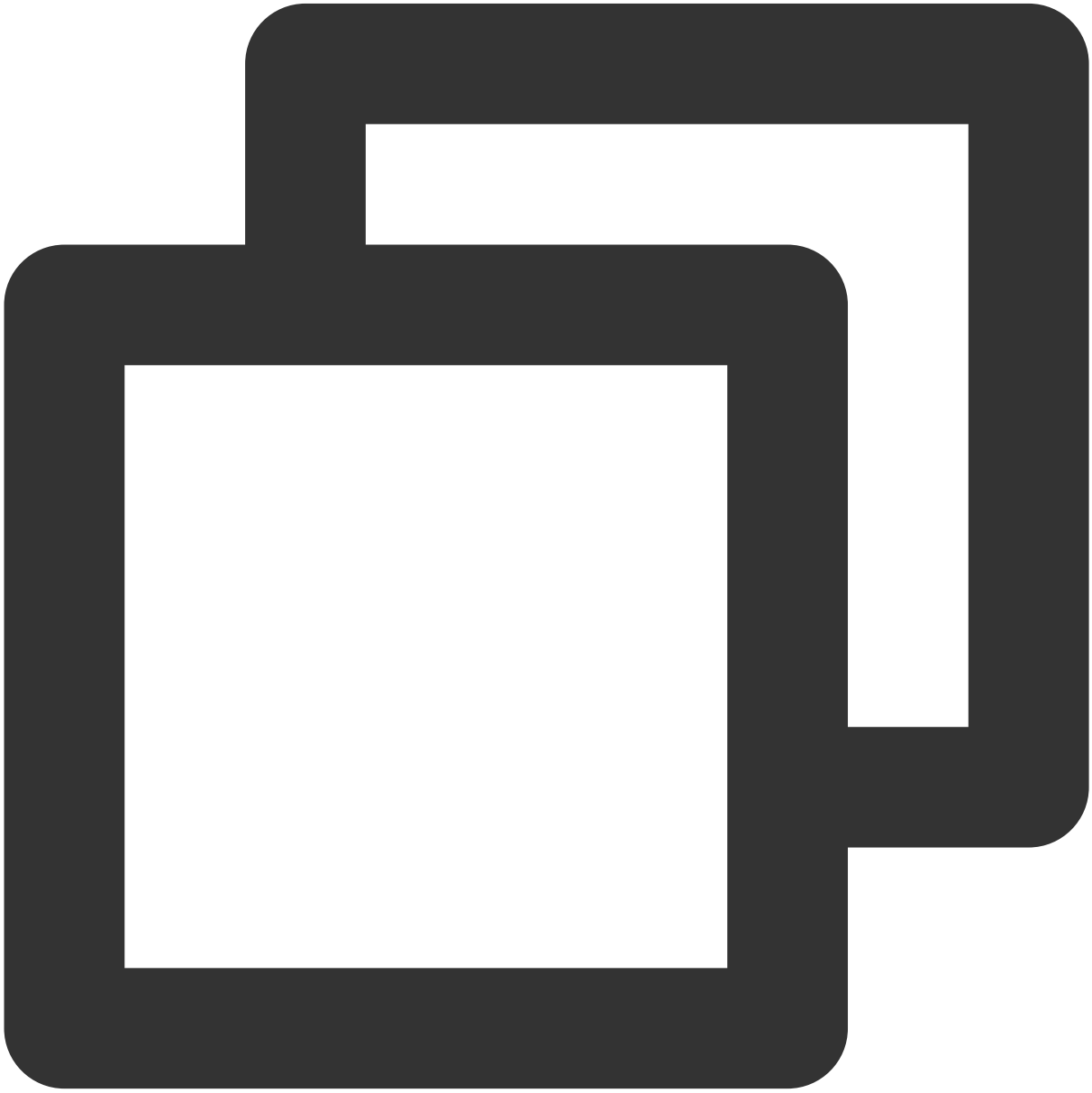


```
V2TIMManager.getInstance().joinGroup(groupID, message, new V2TIMCallback() {
  @Override
  public void onSuccess() {
      // Successfully joined the group.
  }

  @Override
  public void onError(int code, String desc) {
      // Failed to join the group.
  }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Listen for the event of joining a group.
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
  @Override
  public void onMemberEnter(String groupID, List<V2TIMGroupMemberInfo> memberList)
      // Someone joined the group.
  }
});
```

Join a TRTC room.



```
private void enterRoom(String roomId, String userId) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Using a string as the room ID for example, it is recommended to keep it cons
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // For entering a room in voice chat interaction scenarios, specify the user's
    params.role = TRTCCloudDef.TRTCRoleAudience;
    // Use room entry in voice chat interaction scenarios as an example.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_VOICE_CHATROOM);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

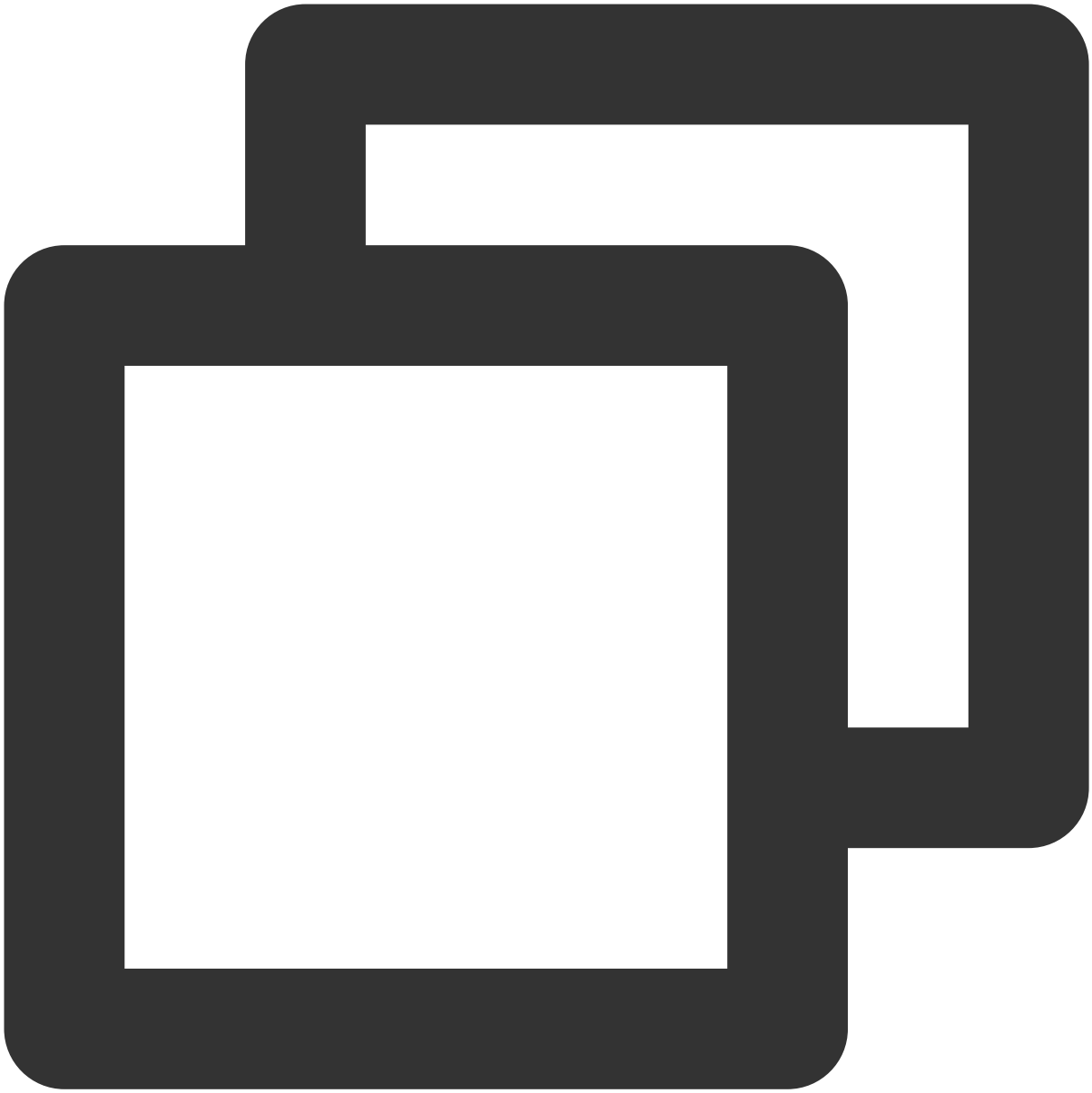
When entering a room in voice chat interaction scenarios, it is necessary to specify the user's role (anchor/audience). Only anchors have permissions to push streams. If not specified, the default role is anchor.

For voice chat interaction room-entering scenarios, it is recommended to use `TRTC_APP_SCENE_VOICE_CHATROOM`.

3. Exit the room.

Exit the IM group.

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMManager.getInstance().quitGroup(groupID, new V2TIMCallback() {
  @Override
  public void onSuccess() {
      // Exiting the group successful.
  }

  @Override
  public void onError(int code, String desc) {
      // Exiting the group failed.
  }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

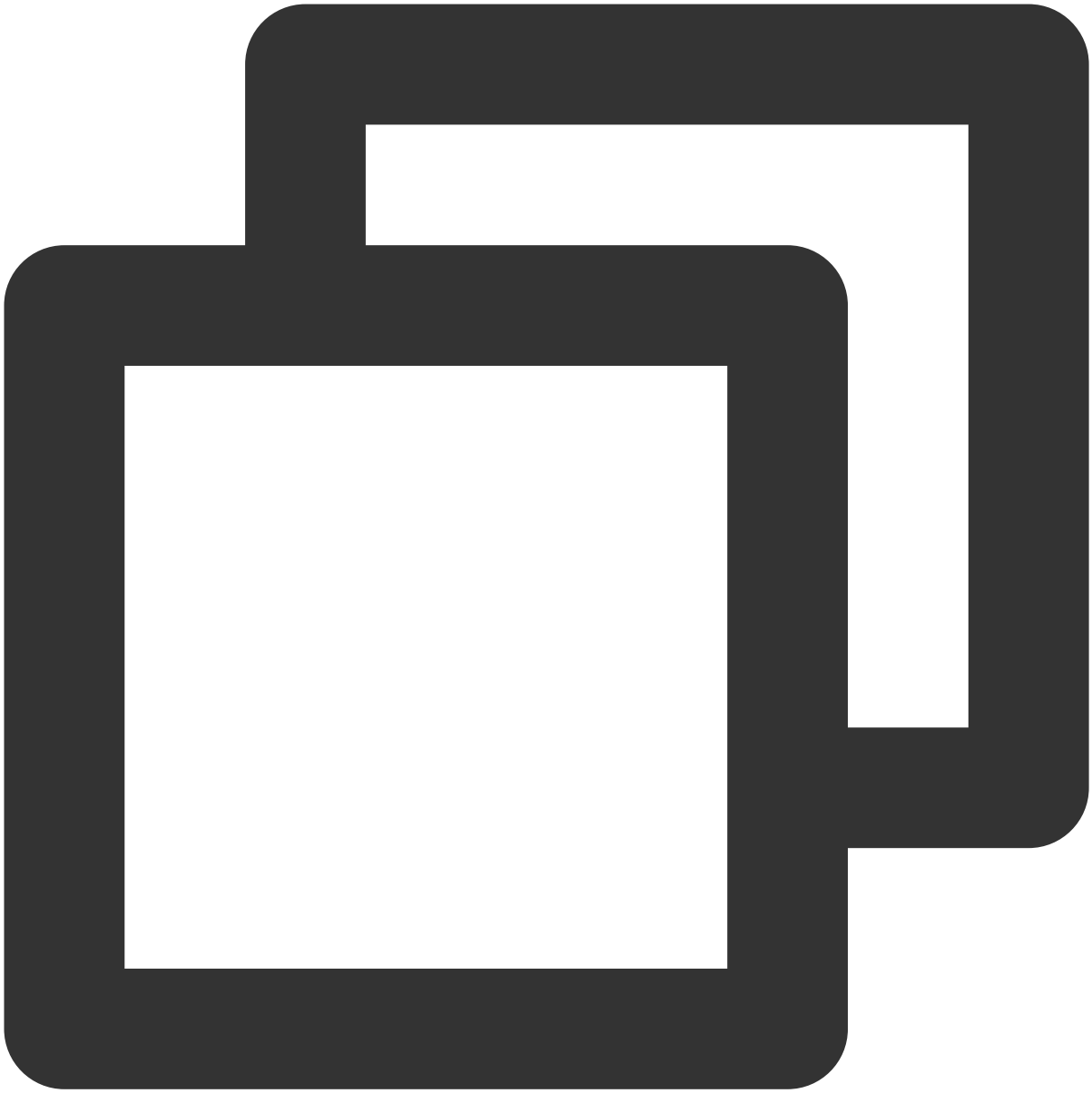
```
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
  @Override
  public void onMemberLeave(String groupID, V2TIMGroupMemberInfo member) {
        // Group member leave callback.
  }
});
```

Note:

In a live streaming group (AVChatRoom), the group owner cannot exit the group. The owner can only dissolve the group by calling `dismissGroup` .

Exit the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



```
private void exitRoom() {
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.exitRoom();
}

// Event callback for exiting the room.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            Log.d(TAG, "Removed from the current room by the server.");
        } else if (reason == 2) {
            Log.d(TAG, "The current room has been dissolved.");
        }
    }
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

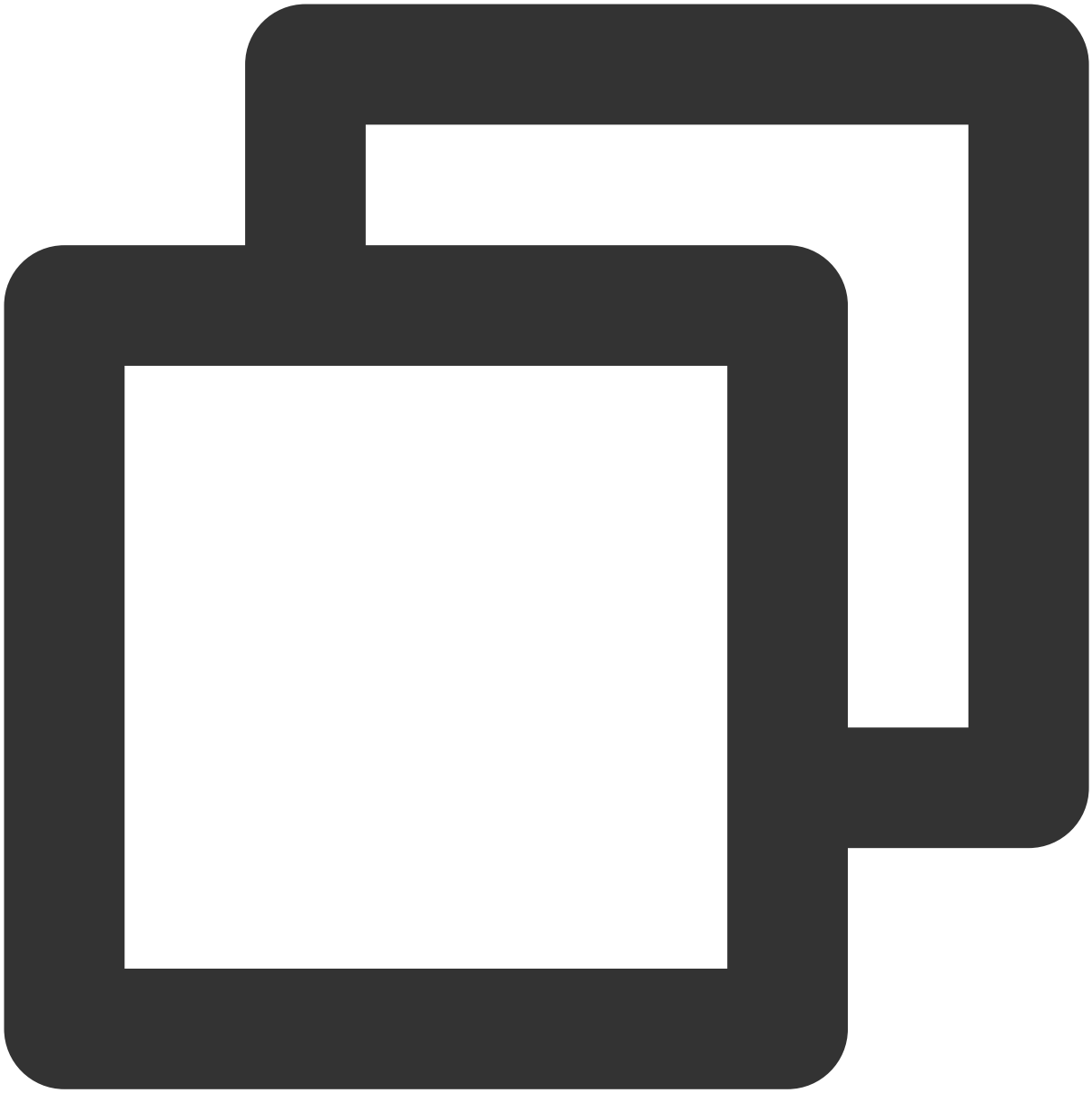
If you want to call `enterRoom` again or switch to another audio/video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter exceptions such as the camera or microphone being forcefully occupied.

4. Dissolve the room.

Dissolve the IM group.

This example only shows the client method of dissolving an IM group. It can also be done through server dissolves the group.

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMManager.getInstance().dismissGroup(groupID, new V2TIMCallback() {
  @Override
  public void onSuccess() {
      // Dissolving group successful.
  }

  @Override
  public void onError(int code, String desc) {
      // Dissolving the group failed.
  }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

```
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
  @Override
  public void onGroupDismissed(String groupID, V2TIMGroupMemberInfo opUser) {
      // Group dissolved callback.
  }
});
```

Dissolve TRTC room.

Server dissolvement: TRTC provides the Server dissolves the room API `DismissRoom` (differentiating between numeric room ID and string room ID). You can call this API to remove all users from the room and dissolve the room.

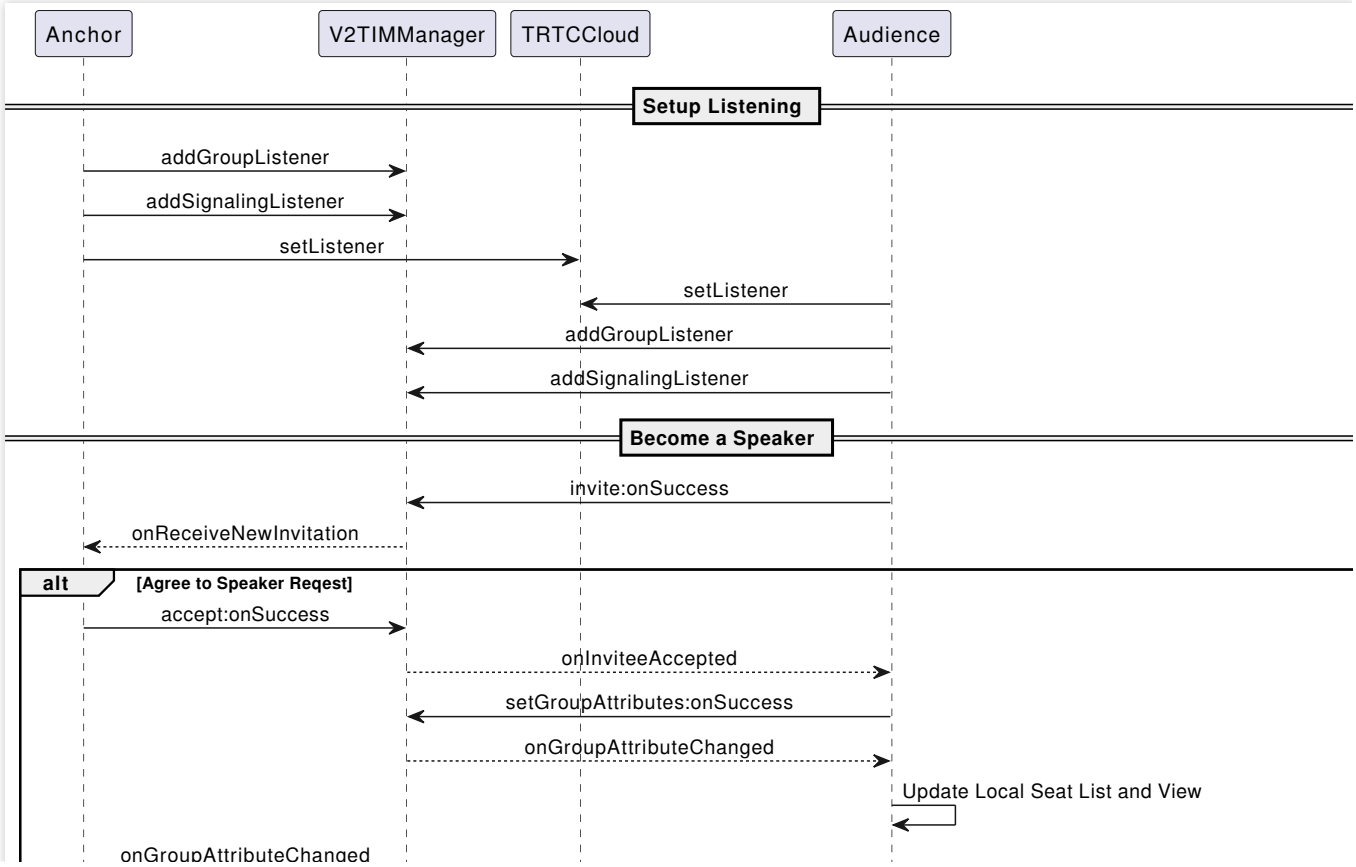
Client dissolvement: Through the room exit `exitRoom` API of each client, all the anchors and audiences in the room can be completed of room exit. After room exit, according to TRTC room lifecycle rules, the room will automatically be dissolved. For details, see Exit Room.

Warning:

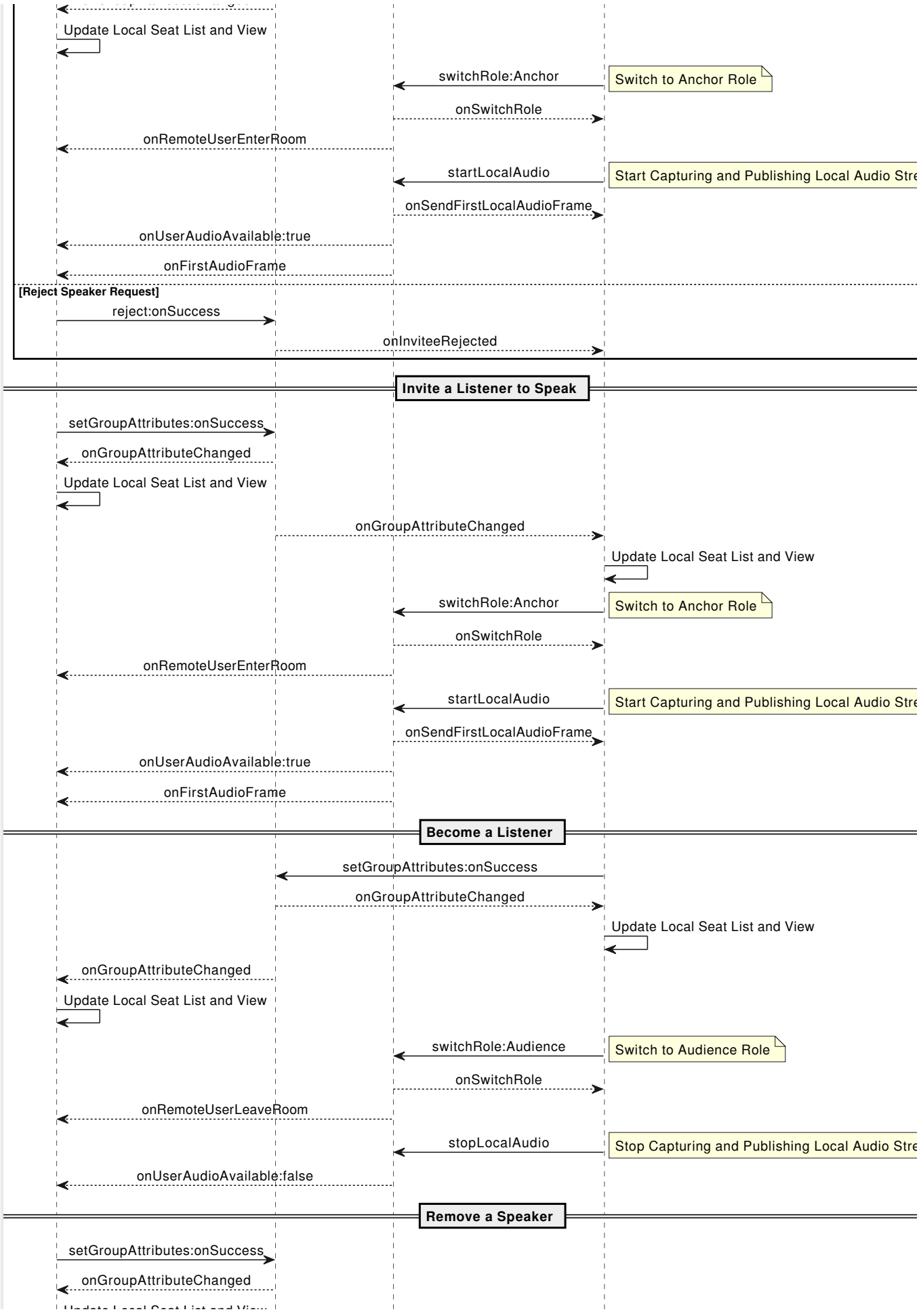
It is recommended that after the end of live streaming, you call the room dissolvement API to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

Step 5: Seat management.

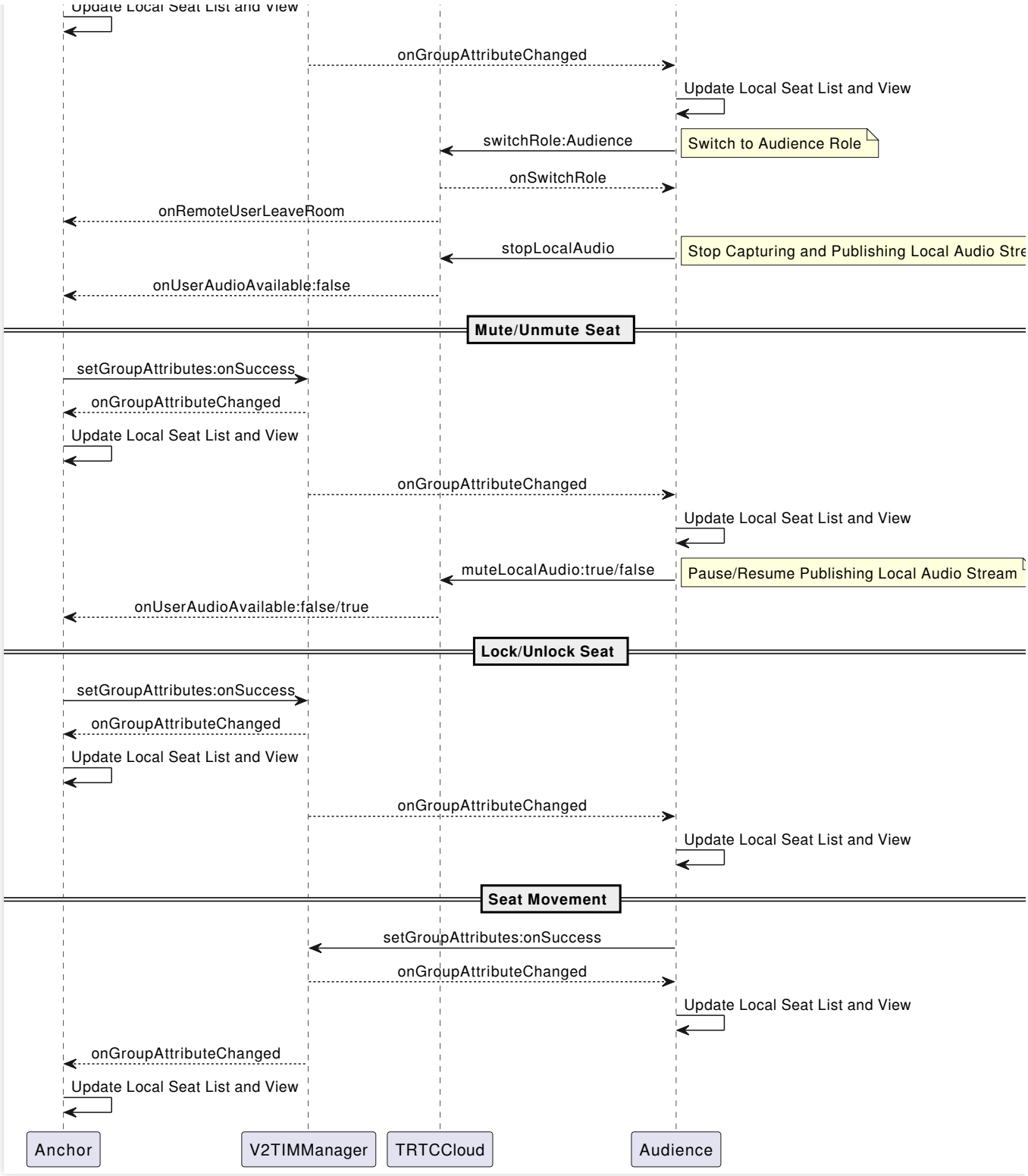
Sequence diagram



©2013-2022 Tencent Cloud. All rights reserved.

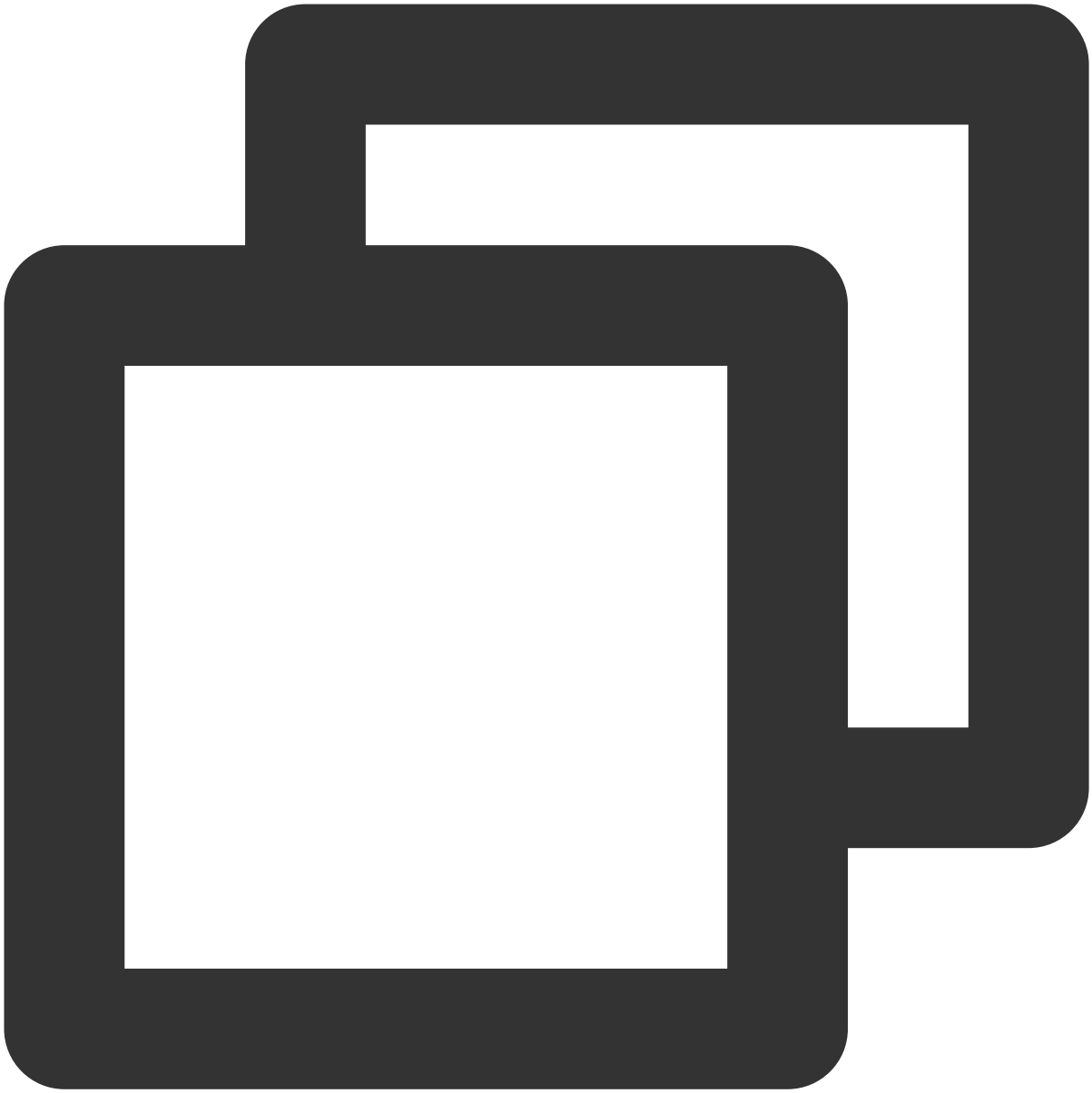


©2013-2022 Tencent Cloud. All rights reserved.



First, we can create a JavaBean to save seat information.

©2013-2022 Tencent Cloud. All rights reserved.



```
public class SeatInfo implements Serializable {
    public static final transient int STATUS_UNUSED = 0;
    public static final transient int STATUS_USED = 1;
    public static final transient int STATUS_LOCKED = 2;

    // The status of seats. There are three corresponding statuses.
    public int status;
    // Whether the seat is muted.
    public boolean mute;
    // When the seat is occupied, the user information is stored.
    public String userId;
```

©2013-2022 Tencent Cloud. All rights reserved.

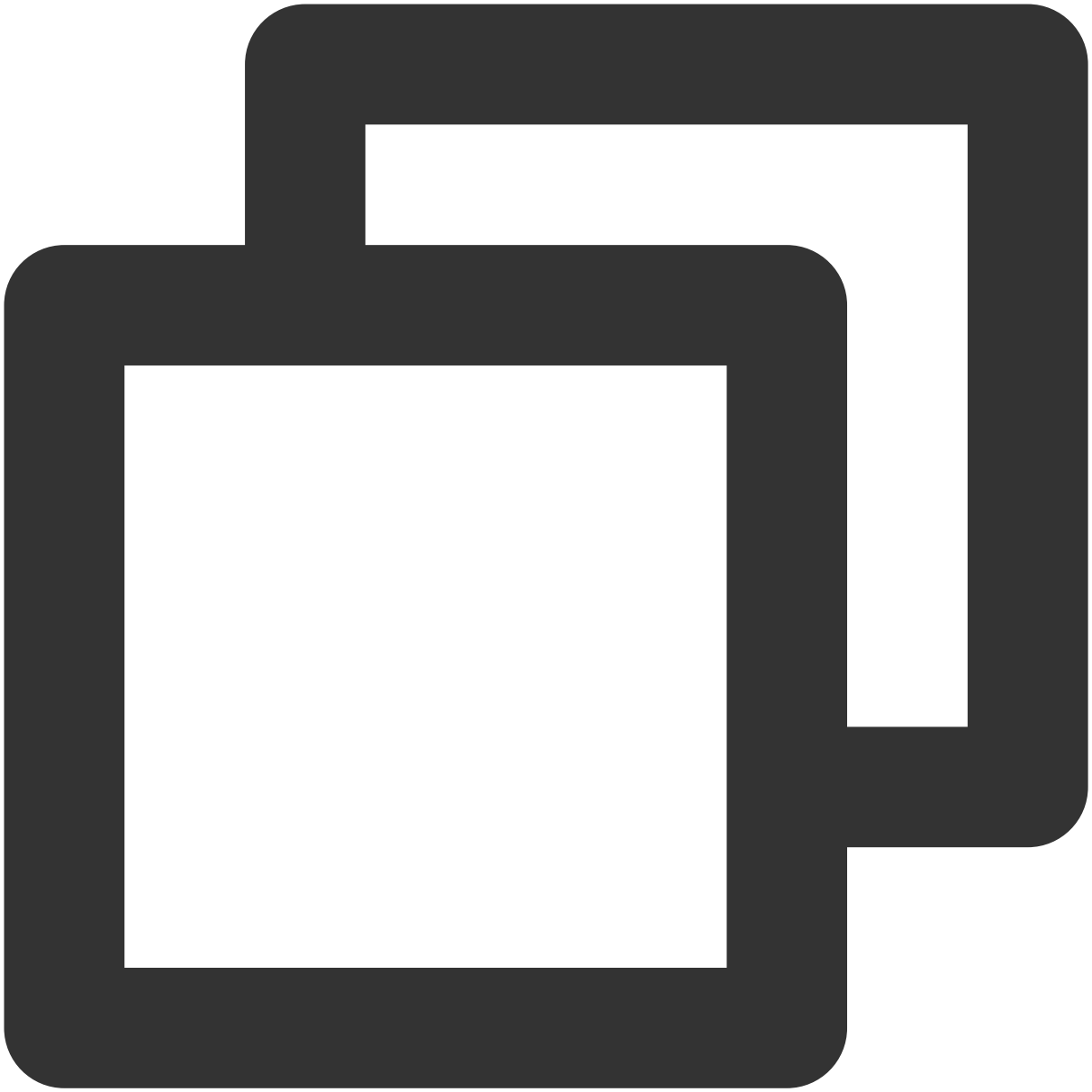
```
    @Override
    public String toString() {
        return "TXSeatInfo{"
                + "status=" + status
                + ", mute=" + mute
                + ", user='" + userId + '\\''
                + '}';
    }
}
```

1. Become a speaker.

Becoming a speaker refers to off-mic audience sending a request to speak to the room owner or administrator. The audience can speak once the approval signaling is received. In a free-speaking mode, the signaling request part can be skipped.

Audience sends a request to speak.

©2013-2022 Tencent Cloud. All rights reserved.



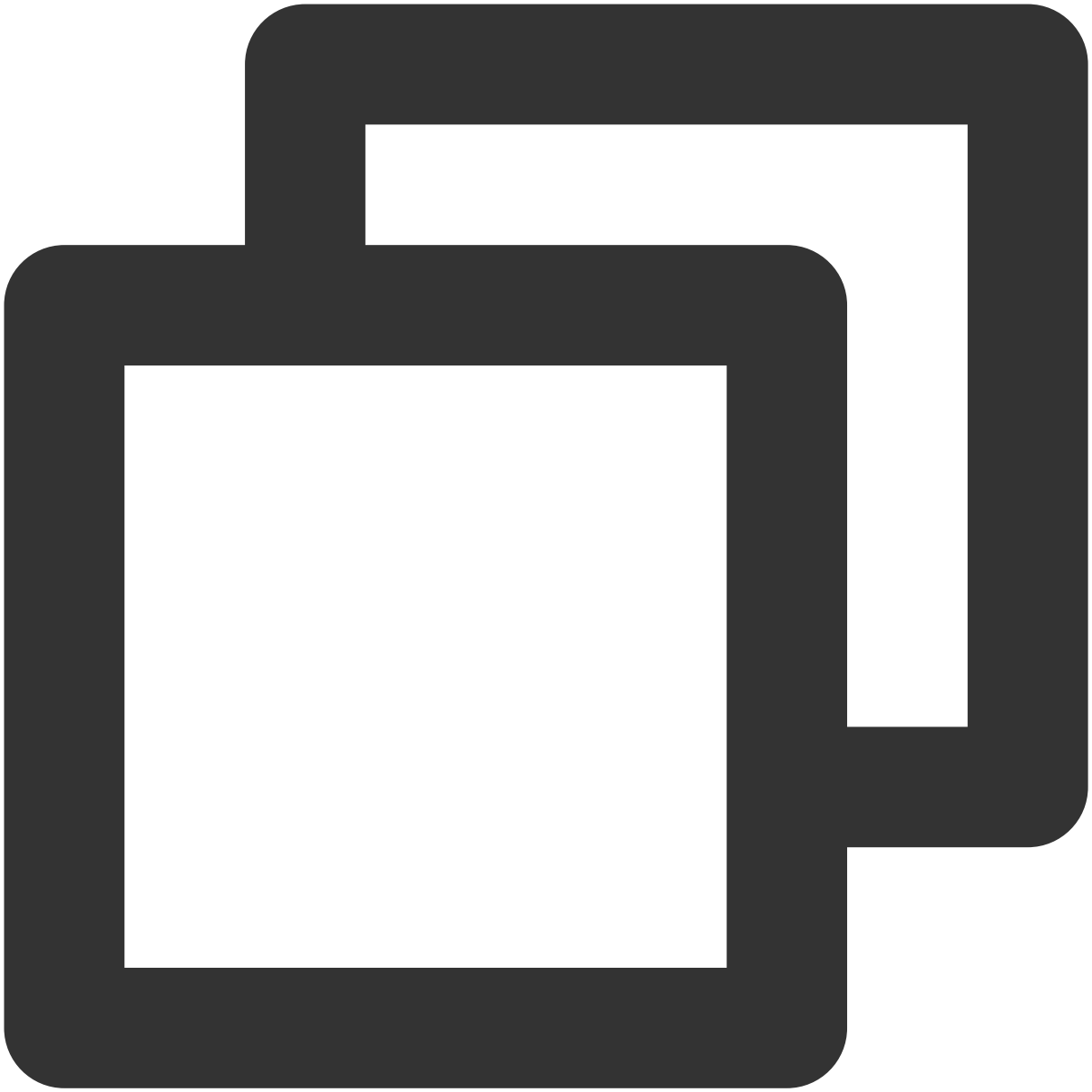
```
// Audience sends a request to speak. userId is the Anchor ID, and data can pass in
private String sendInvitation(String userId, String data) {
    return V2TIMManager.getSignalingManager().invite(userId, data, true, null, 0, n
        @Override
        public void onError(int i, String s) {
            Log.e(TAG, "sendInvitation error " + i);
        }

        @Override
        public void onSuccess() {
            Log.i(TAG, "sendInvitation success ");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    });
}

// Anchor receives the request to speak. inviteID is the request ID, and inviter is
V2TIMManager.getSignalingManager().addSignalingListener(new V2TIMSignalingListener(
    @Override
    public void onReceiveNewInvitation(String inviteID, String inviter,
                                       String groupId, List<String> inviteeList, St
        Log.i(TAG, "received invitation: " + inviteID + " from " + inviter);
    }
});
```

Anchor processes the request to speak.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Agree to the request to speak.
private void acceptInvitation(String inviteID, String data) {
    V2TIMManager.getSignalingManager().accept(inviteID, data, new V2TIMCallback() {
        @Override
        public void onError(int i, String s) {
            Log.e(TAG, "acceptInvitation error " + i);
        }

        @Override
        public void onSuccess() {
            Log.i(TAG, "acceptInvitation success ");
```

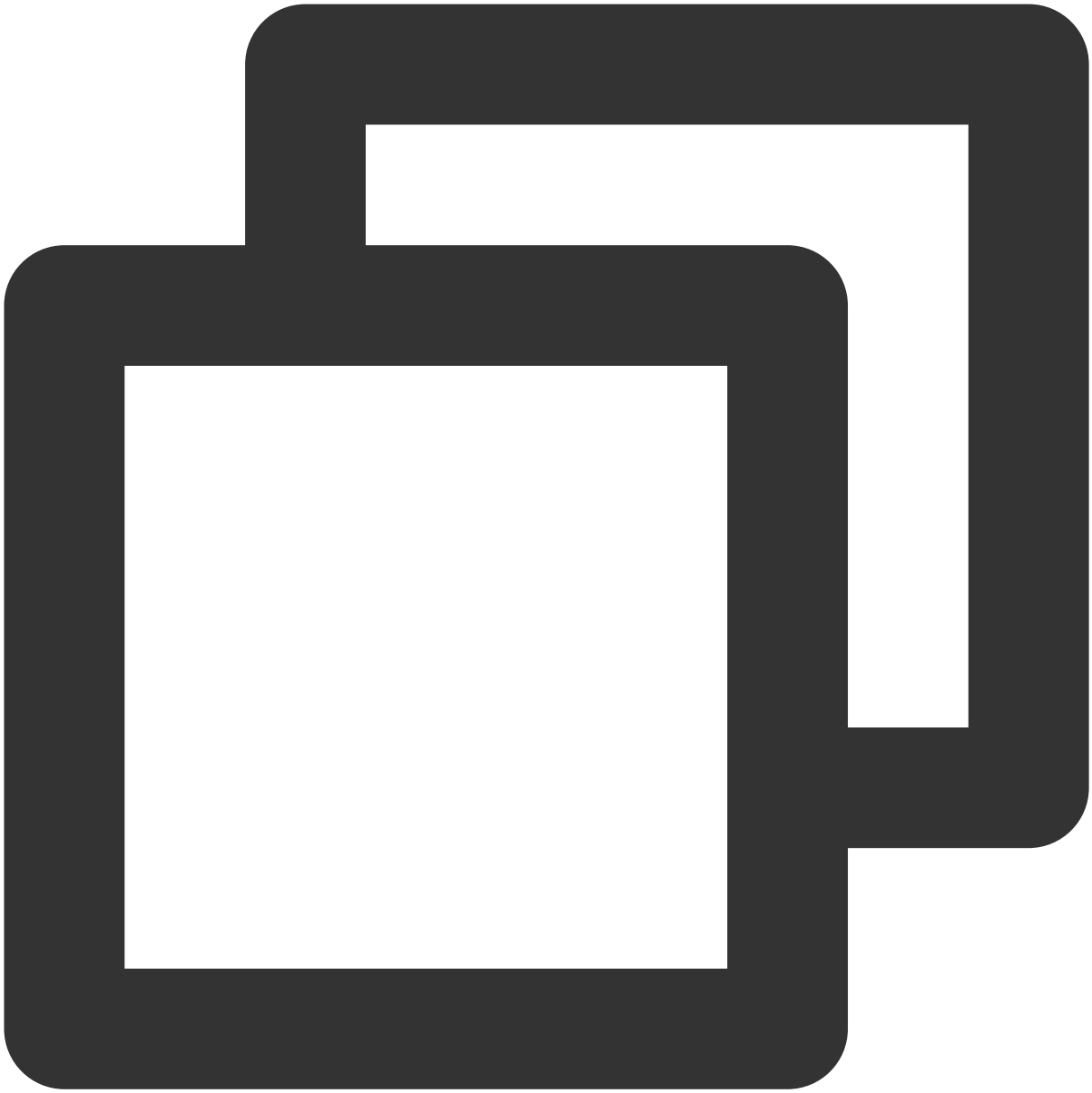
©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    });
}

// Reject the request to speak.
private void rejectInvitation(String inviteID, String data) {
    V2TIMManager.getSignalingManager().reject(inviteID, data, new V2TIMCallback() {
        @Override
        public void onError(int i, String s) {
            Log.e(TAG, "rejectInvitation error " + i);
        }

        @Override
        public void onSuccess() {
            Log.i(TAG, "rejectInvitation success ");
        }
    });
}
```

Audience to speak.

If the anchor agrees to the audience's request to speak, the audience can add seat information by modifying group attributes. Other users will receive a callback for the change in group attributes. Update the local seat information.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// Callback for agreeing to the request to speak.
V2TIMManager.getSignalingManager().addSignalingListener(new V2TIMSignalingListener(
    @Override
    public void onInviteeAccepted(String inviteID, String invitee, String data) {
        Log.i(TAG, "received accept invitation: " + inviteID + " from " + invitee);
        takeSeat(seatIndex);
    }
});
```

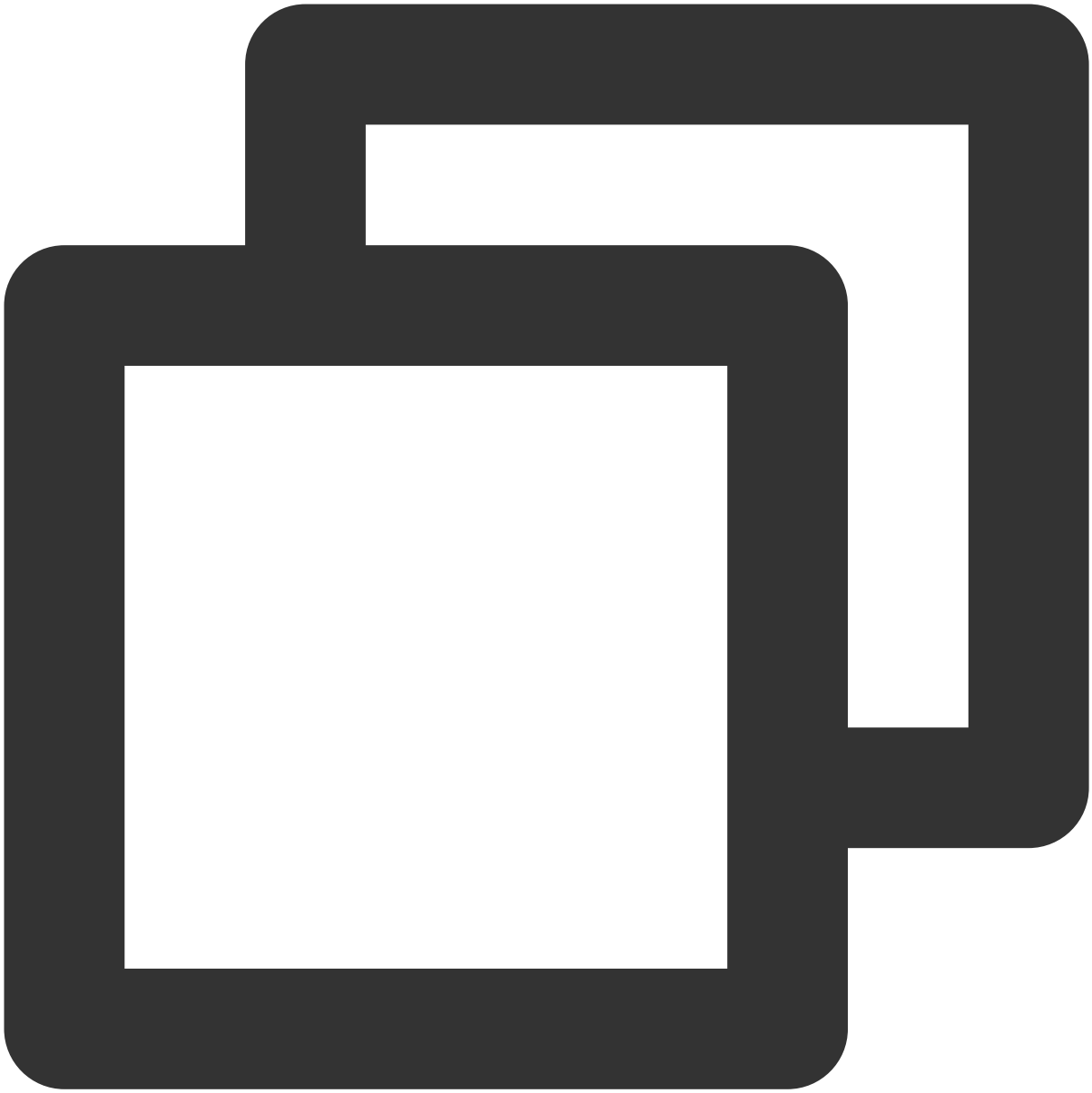
©2013-2022 Tencent Cloud. All rights reserved.

```
// Audience begins to speak.
private void takeSeat(int seatIndex) {
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = SeatInfo.STATUS_USED;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = mUserId;

    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    String json = gson.toJson(seatInfo, SeatInfo.class);
    HashMap<String, String> map = new HashMap<>();
    map.put("seat" + seatIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to become a speaker.
        }

        @Override
        public void onSuccess() {
            // Successfully modified group attributes. Switch TRTC role and start s
            mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
            mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
        }
    });
}
```

2. Invite a listener to speak.

Anchor invites a listener to speak (without the need for audience's consent). Directly modify group attributes saved for seats, and corresponding audiences will receive a callback for the change in group attributes. After matching the userId successfully, they can switch TRTC role and start streaming. In an invite-to-speak mode, see the implementation logic of becoming a speaker. Just switch the sender and receiver of the signaling.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// Anchor calls this API to modify the seat information saved in group attributes.
private void pickSeat(String userId, int seatIndex) {
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = SeatInfo.STATUS_USED;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = userId;
```

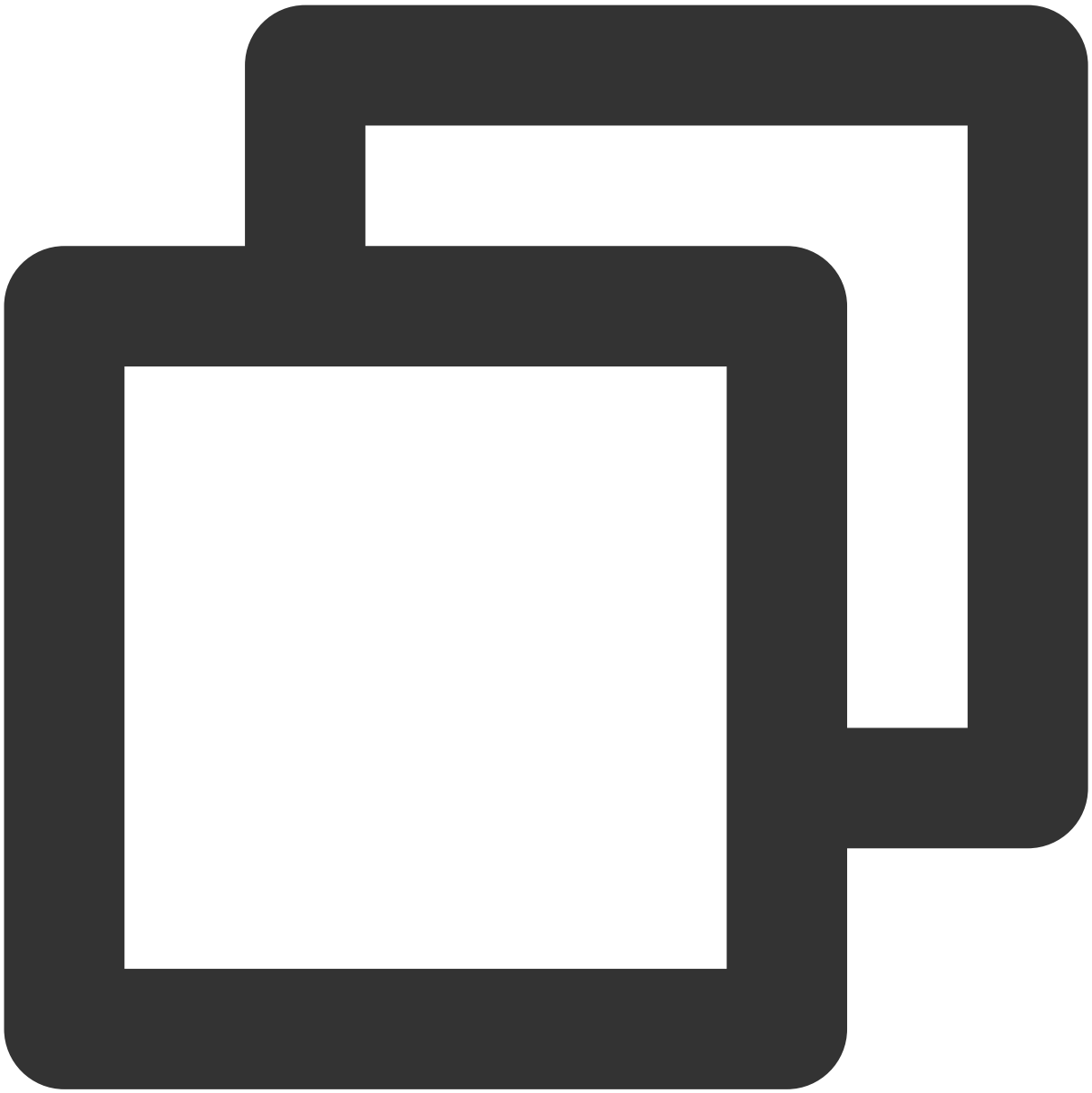
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    String json = gson.toJson(seatInfo, SeatInfo.class);
    HashMap<String, String> map = new HashMap<>();
    map.put("seat" + seatIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to be invited to the micro
        }

        @Override
        public void onSuccess() {
            // Successfully modified group attributes and it triggers onGroupAttrib
        }
    });
}

// Audience receives group attribute change callback. Audience starts streaming aft
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
    @Override
    public void onGroupAttributeChanged(String groupID, Map<String, String> groupAt
        // Last locally saved full list of seats.
        final List<SeatInfo> oldSeatInfoList = mSeatInfoList;
        // The most recent full list of seats parsed from groupAttributeMap.
        final List<SeatInfo> newSeatInfoList = getSeatListFromAttr(groupAttributeMa
        // Iterate through the full list of seats. Compare old and new seat informa
        for (int i = 0; i < seatSize; i++) {
            SeatInfo oldInfo = oldSeatInfoList.get(i);
            SeatInfo newInfo = newSeatInfoList.get(i);
            if (oldInfo.status != newInfo.status && newInfo.status == SeatInfo.STAT
                if (newInfo.userId.equals(mUserId)) {
                    // Match own information successfully. Switch TRTC role and sta
                    mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
                    mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFA
                } else {
                    // Update local seat list. Render local seat view.
                }
            }
        }
    }
});
```

3. Become a listener.

©2013-2022 Tencent Cloud. All rights reserved.

Mic-connecting audiences can reset seat information by modifying group attributes. Other users will receive a group attribute change callback. Update local seat information.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

private void leaveSeat(int seatIndex) {
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = SeatInfo.STATUS_UNUSED;
```

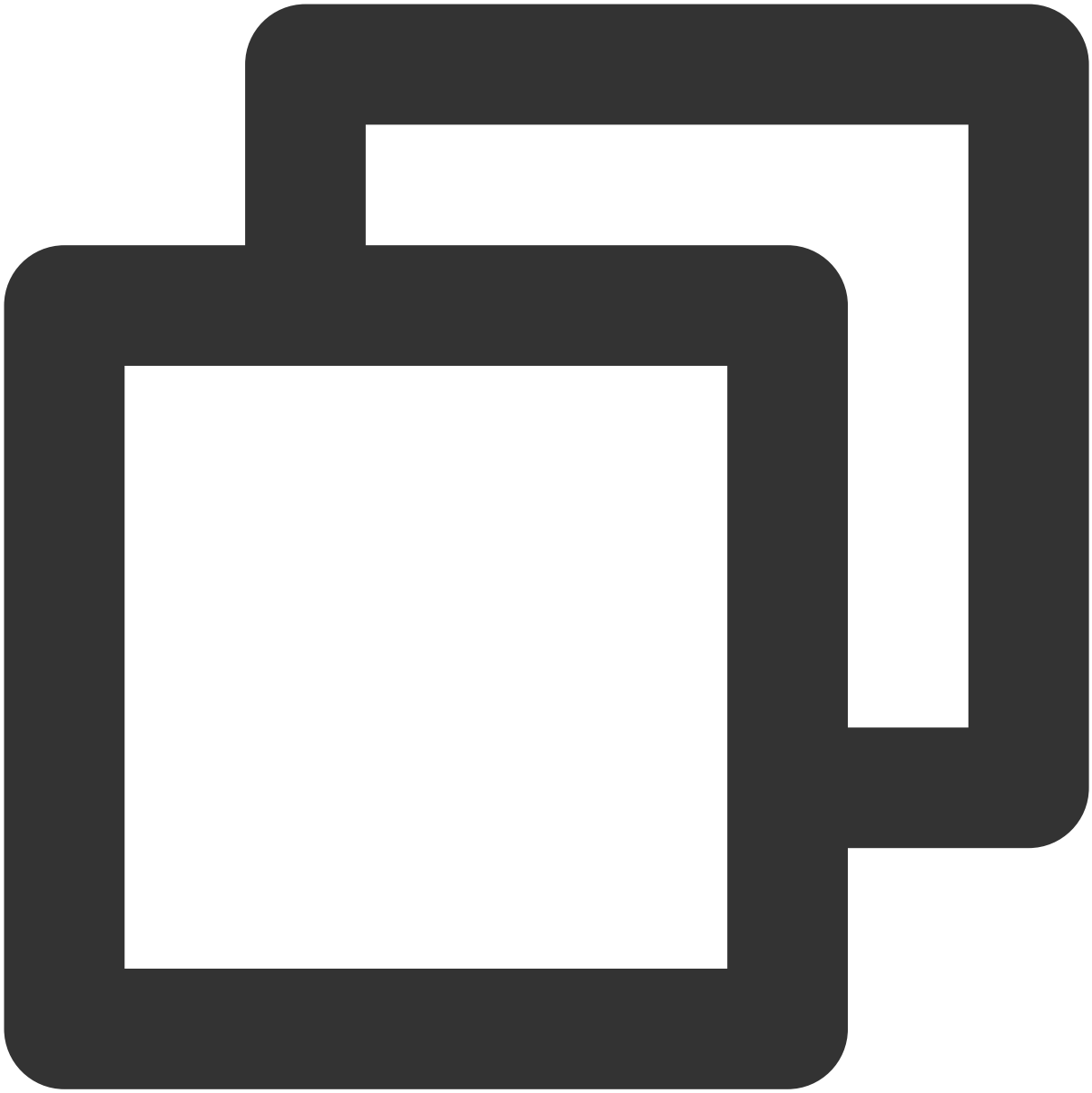
©2013-2022 Tencent Cloud. All rights reserved.

```
        seatInfo.mute = localInfo.mute;
        seatInfo.userId = "";

        // Serialize the seat information object into JSON format.
        Gson gson = new Gson();
        String json = gson.toJson(seatInfo, SeatInfo.class);
        HashMap<String, String> map = new HashMap<>();
        map.put("seat" + seatIndex, json);

        // Set group attributes. If the group attribute already exists, its value is up
        V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
            @Override
            public void onError(int code, String message) {
                // Failed to modify group attributes. Failed to become a listener.
            }

            @Override
            public void onSuccess() {
                // Successfully modified group attributes. Switch TRTC role and stop st
                mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);
                mTRTCCloud.stopLocalAudio();
            }
        });
    }
```

4. Remove a speaker.

Anchor removes a speaker. Directly modify the seat information saved in group attributes. Corresponding mic-connecting audience receives group attribute change callback. After successfully matching userId, they switch TRTC role and stop streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// Anchor calls this API to modify the seat information saved in group attributes.
private void kickSeat(int seatIndex) {
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = SeatInfo.STATUS_UNUSED;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = "";
```

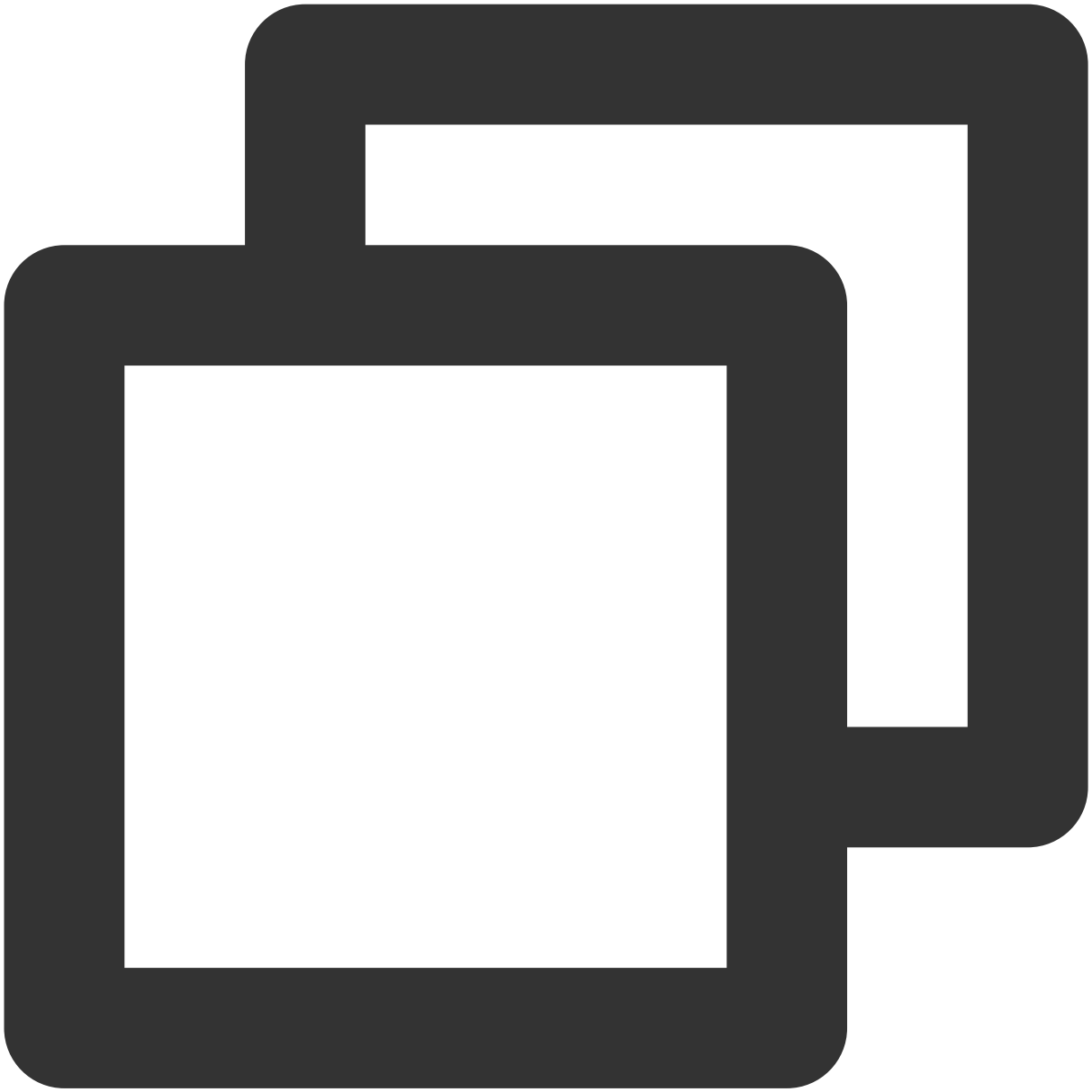
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    String json = gson.toJson(seatInfo, SeatInfo.class);
    HashMap<String, String> map = new HashMap<>();
    map.put("seat" + seatIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to remove the speaker.
        }

        @Override
        public void onSuccess() {
            // Successfully modified group attributes and it triggers onGroupAttrib
        }
    });
}

// Mic-connecting audience receives group attribute change callback. It stops strea
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
    @Override
    public void onGroupAttributeChanged(String groupID, Map<String, String> groupAt
        // Last locally saved full list of seats.
        final List<SeatInfo> oldSeatInfoList = mSeatInfoList;
        // The most recent full list of seats parsed from groupAttributeMap.
        final List<SeatInfo> newSeatInfoList = getSeatListFromAttr(groupAttributeMa
        // Iterate through the full list of seats. Compare old and new seat informa
        for (int i = 0; i < seatSize; i++) {
            SeatInfo oldInfo = oldSeatInfoList.get(i);
            SeatInfo newInfo = newSeatInfoList.get(i);
            if (oldInfo.status != newInfo.status && newInfo.status == SeatInfo.STAT
                if (oldInfo.userId.equals(mUserId)) {
                    // Match own information successfully. Switch TRTC role and sto
                    mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);
                    mTRTCCloud.stopLocalAudio();
                } else {
                    // Update local seat list. Render local seat view.
                }
            }
        }
    }
});
```

5. Mute a seat.

©2013-2022 Tencent Cloud. All rights reserved.

Anchor mutes/unmutes a specific seat. Directly modify the seat information saved in group attributes. Corresponding mic-connecting audience receives group attribute change callback. After successfully matching userId, they pause/resume local streaming.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// Anchor calls this API to modify the seat information saved in group attributes.
private void muteSeat(int seatIndex, boolean mute) {
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
```

©2013-2022 Tencent Cloud. All rights reserved.

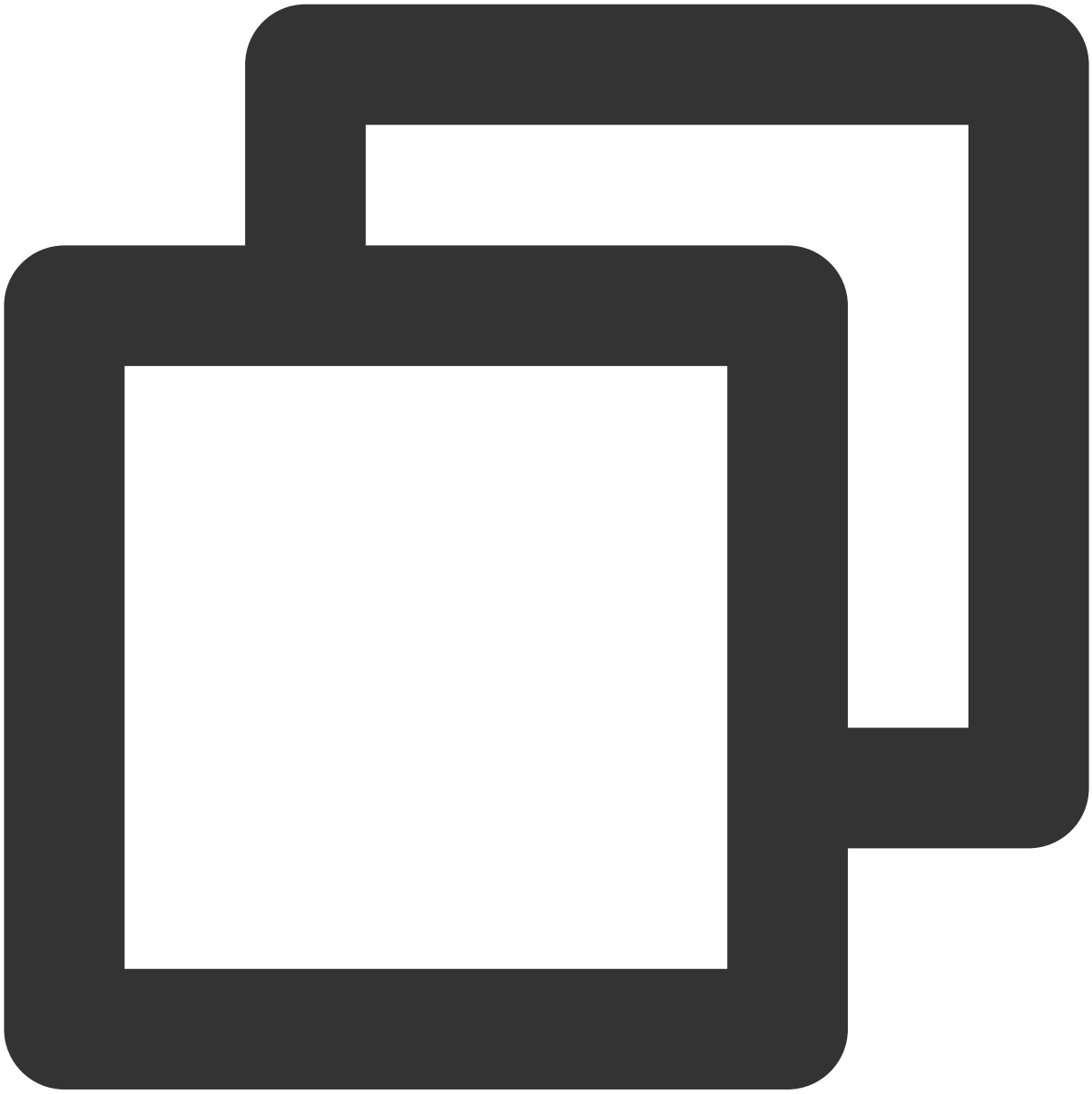
```
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = localInfo.status;
    seatInfo.mute = mute;
    seatInfo.userId = localInfo.userId;

    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    String json = gson.toJson(seatInfo, SeatInfo.class);
    HashMap<String, String> map = new HashMap<>();
    map.put("seat" + seatIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to mute the seat.
        }

        @Override
        public void onSuccess() {
            // Successfully modified group attributes and it triggers onGroupAttrib
        }
    });
}

// The mic-connecting audience receives the group attribute change callback. The au
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
    @Override
    public void onGroupAttributeChanged(String groupID, Map<String, String> groupAt
        // Last locally saved full list of seats.
        final List<SeatInfo> oldSeatInfoList = mSeatInfoList;
        // The most recent full list of seats parsed from groupAttributeMap.
        final List<SeatInfo> newSeatInfoList = getSeatListFromAttr(groupAttributeMa
        // Iterate through the full list of seats. Compare old and new seat informa
        for (int i = 0; i < seatSize; i++) {
            SeatInfo oldInfo = oldSeatInfoList.get(i);
            SeatInfo newInfo = newSeatInfoList.get(i);
            if (oldInfo.mute != newInfo.mute) {
                if (oldInfo.userId.equals(mUserId)) {
                    // Match own information successfully. Pause/resume local strea
                    mTRTCCloud.muteLocalAudio(newInfo.mute);
                } else {
                    // Update local seat list. Render local seat view.
                }
            }
        }
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
});
```

6. Lock a seat.

Anchor locks/unlocks a seat by directly modifying the seat information saved in group attributes. Audience updates the corresponding seat view after receiving the group attribute change callback.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// Anchor calls this API to modify the seat information saved in group attributes.
private void lockSeat(int seatIndex, boolean isLock) {
```

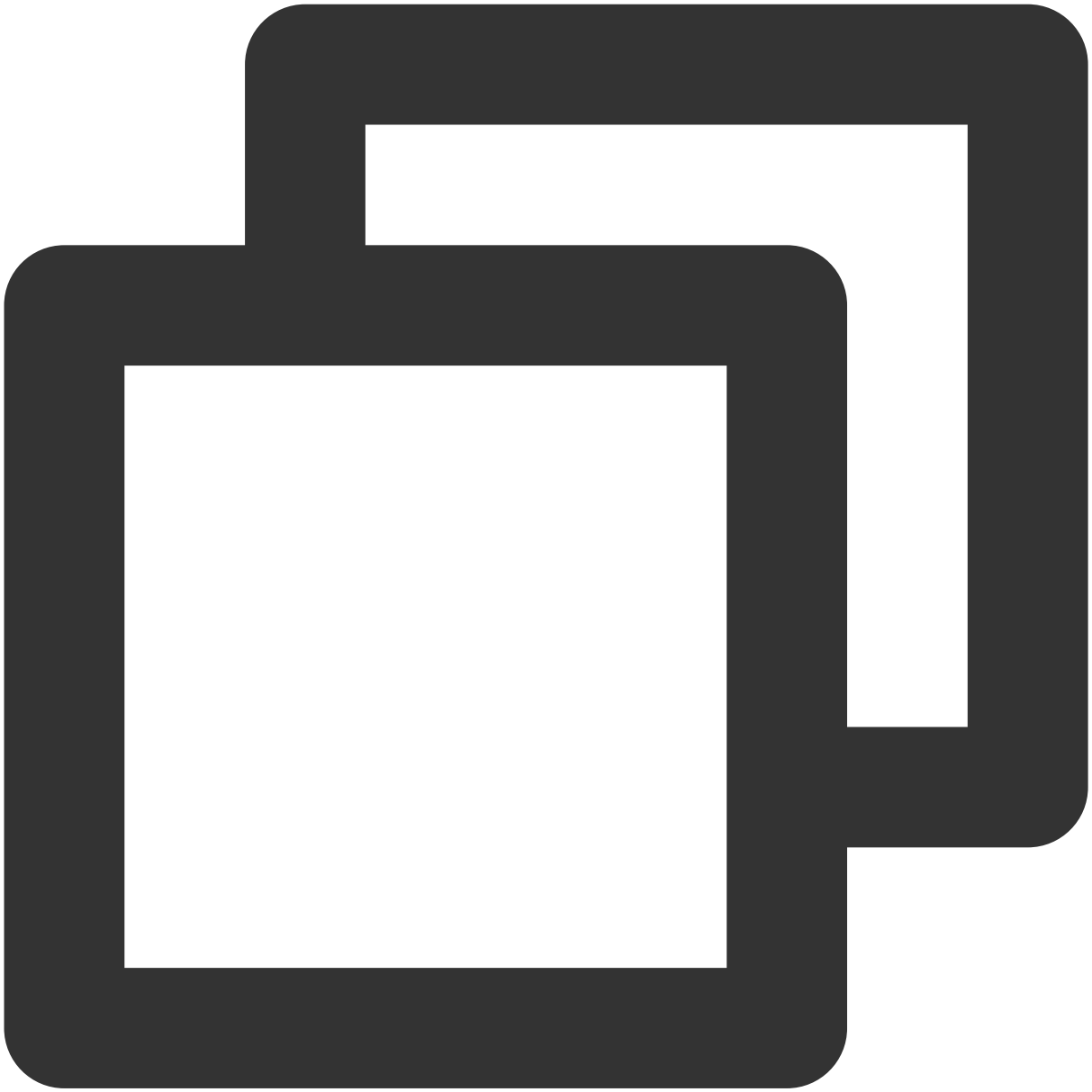
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Create a seat information instance. Store the modified seat information.
    SeatInfo localInfo = mSeatInfoList.get(seatIndex);
    SeatInfo seatInfo = new SeatInfo();
    seatInfo.status = isLock ? SeatInfo.STATUS_LOCKED : SeatInfo.STATUS_UNUSED;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = "";

    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    String json = gson.toJson(seatInfo, SeatInfo.class);
    HashMap<String, String> map = new HashMap<>();
    map.put("seat" + seatIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to lock the seat.
        }

        @Override
        public void onSuccess() {
            // Successfully modified group attributes and it triggers onGroupAttrib
        }
    });
}

// The audience receives the group attribute change callback. Update the correspond
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
    @Override
    public void onGroupAttributeChanged(String groupID, Map<String, String> groupAt
        // Last locally saved full list of seats.
        final List<SeatInfo> oldSeatInfoList = mSeatInfoList;
        // The most recent full list of seats parsed from groupAttributeMap.
        final List<SeatInfo> newSeatInfoList = getSeatListFromAttr(groupAttributeMa
        // Iterate through the full list of seats. Compare old and new seat informa
        for (int i = 0; i < seatSize; i++) {
            SeatInfo oldInfo = oldSeatInfoList.get(i);
            SeatInfo newInfo = newSeatInfoList.get(i);
            if (oldInfo.status == SeatInfo.STATUS_LOCKED && newInfo.status == SeatI
                // Unlock a seat.
            } else if (oldInfo.status != newInfo.status && newInfo.status == SeatIn
                // Lock a seat.
            }
        }
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

7. Move a seat.

On-mic anchor moves a seat by necessarily and separately modifying the source and target seat information saved in group attributes. Audience updates the corresponding seat view after receiving the group attribute change callback.



```
// Locally saved full list of seats.
private List<SeatInfo> mSeatInfoList;

// On-mic anchor calls this API to modify the seat information saved in group attri
private void moveSeat(int dstIndex) {
    // Obtain the source seat ID by userId.
```

©2013-2022 Tencent Cloud. All rights reserved.

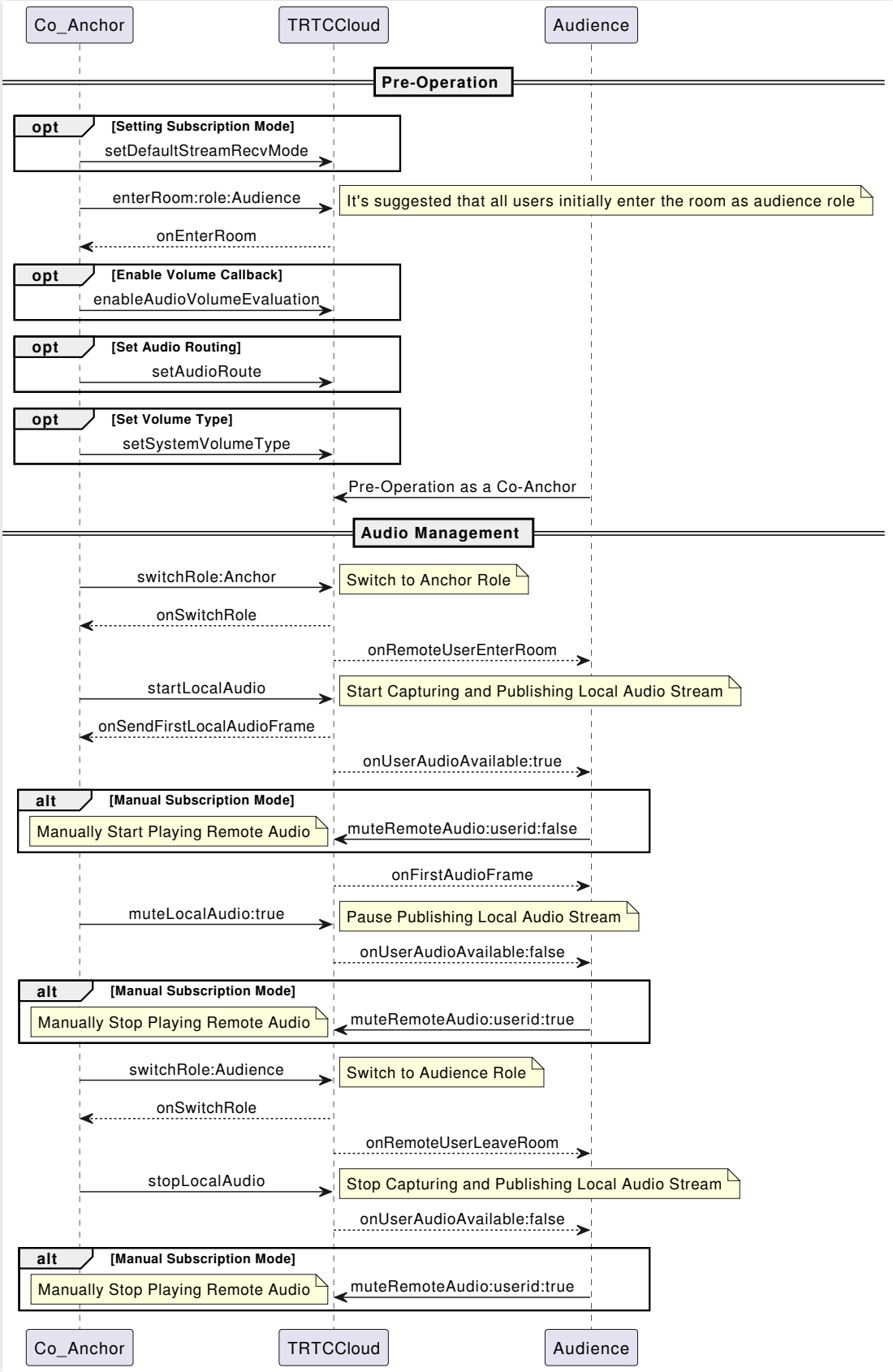
```
    int srcIndex = -1;
    for (int i = 0; i < mSeatInfoList.size(); i++) {
        SeatInfo seatInfo = mSeatInfoList.get(i);
        if (seatInfo != null && mUserId.equals(seatInfo.userId)) {
            srcIndex = i;
            break;
        }
    }

    // Obtain the corresponding seat information by its ID.
    SeatInfo srcSeatInfo = mSeatInfoList.get(srcIndex);
    SeatInfo dstSeatInfo = mSeatInfoList.get(dstIndex);

    // Create a seat information instance to store the modified source seat data.
    SeatInfo srcChangeInfo = new SeatInfo();
    srcChangeInfo.status = SeatInfo.STATUS_UNUSED;
    srcChangeInfo.mute = srcSeatInfo.mute;
    srcChangeInfo.userId = "";

    // Create a seat information instance to store the modified target seat data.
    SeatInfo dstChangeInfo = new SeatInfo();
    dstChangeInfo.status = SeatInfo.STATUS_USED;
    dstChangeInfo.mute = dstSeatInfo.mute;
    dstChangeInfo.userId = mUserId;

    // Serialize the seat information object into JSON format.
    Gson gson = new Gson();
    HashMap<String, String> map = new HashMap<>();
    String json = gson.toJson(srcChangeInfo, SeatInfo.class);
    map.put("seat" + srcIndex, json);
    json = gson.toJson(dstChangeInfo, SeatInfo.class);
    map.put("seat" + dstIndex, json);

    // Set group attributes. If the group attribute already exists, its value is up
    V2TIMManager.getGroupManager().setGroupAttributes(groupId, map, new V2TIMCallba
        @Override
        public void onError(int code, String message) {
            // Failed to modify group attributes. Failed to move the seat.
        }

        @Override
        public void onSuccess() {
            // Modify group attributes successfully. Move the seat successfully.
        }
    });
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Step 6: Audio management.

Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.

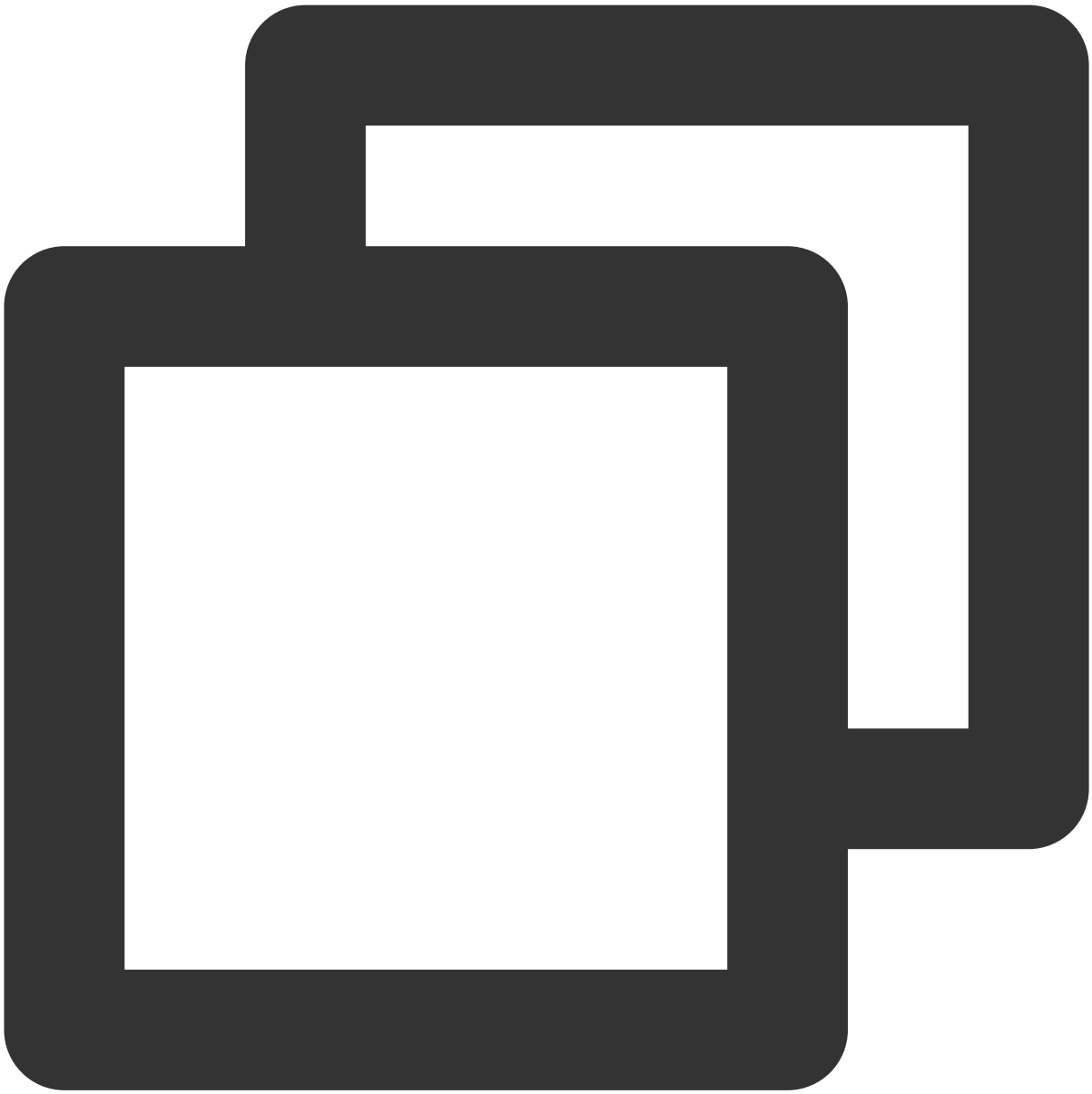


1. Subscription mode.

By default, the TRTC SDK uses an automatic subscription mode for audio streams. When users enter the room, the system will automatically play remote users' voice. If manual subscription to audio streams is needed, calling

©2013-2022 Tencent Cloud. All rights reserved.

`muteRemoteAudio(userId, mute)` to subscribe to and play remote users' audio streams is required.



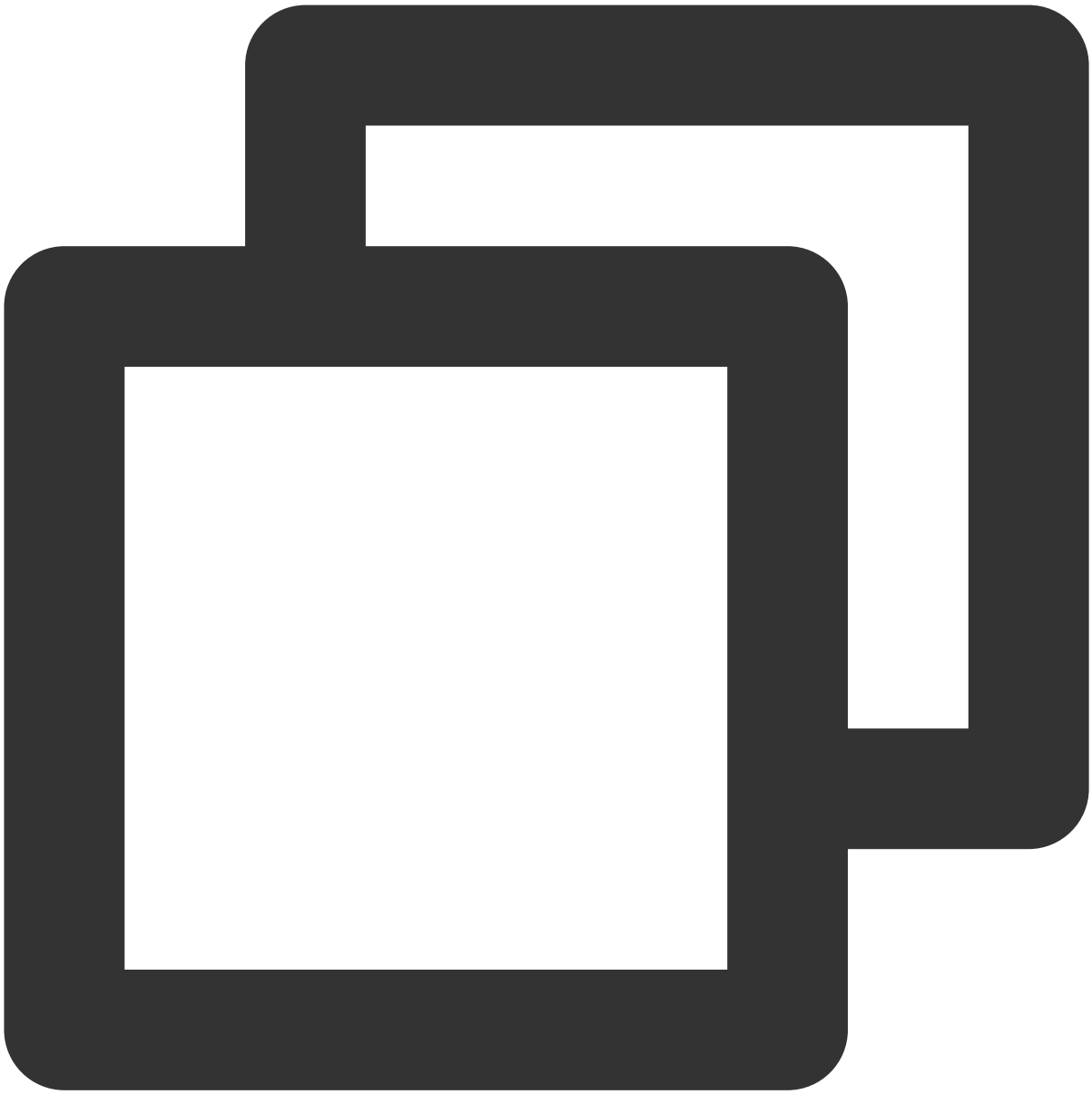
```
// Automatic subscription mode (default).
mTRTCCloud.setDefaultStreamRecvMode(true, true);

// Manual subscription mode (custom).
mTRTCCloud.setDefaultStreamRecvMode(false, false);
```

Note:

Set the subscription mode `setDefaultStreamRecvMode` before entering the room `enterRoom` to ensure to take effect.

©2013-2022 Tencent Cloud. All rights reserved.

2. Capture and publish.



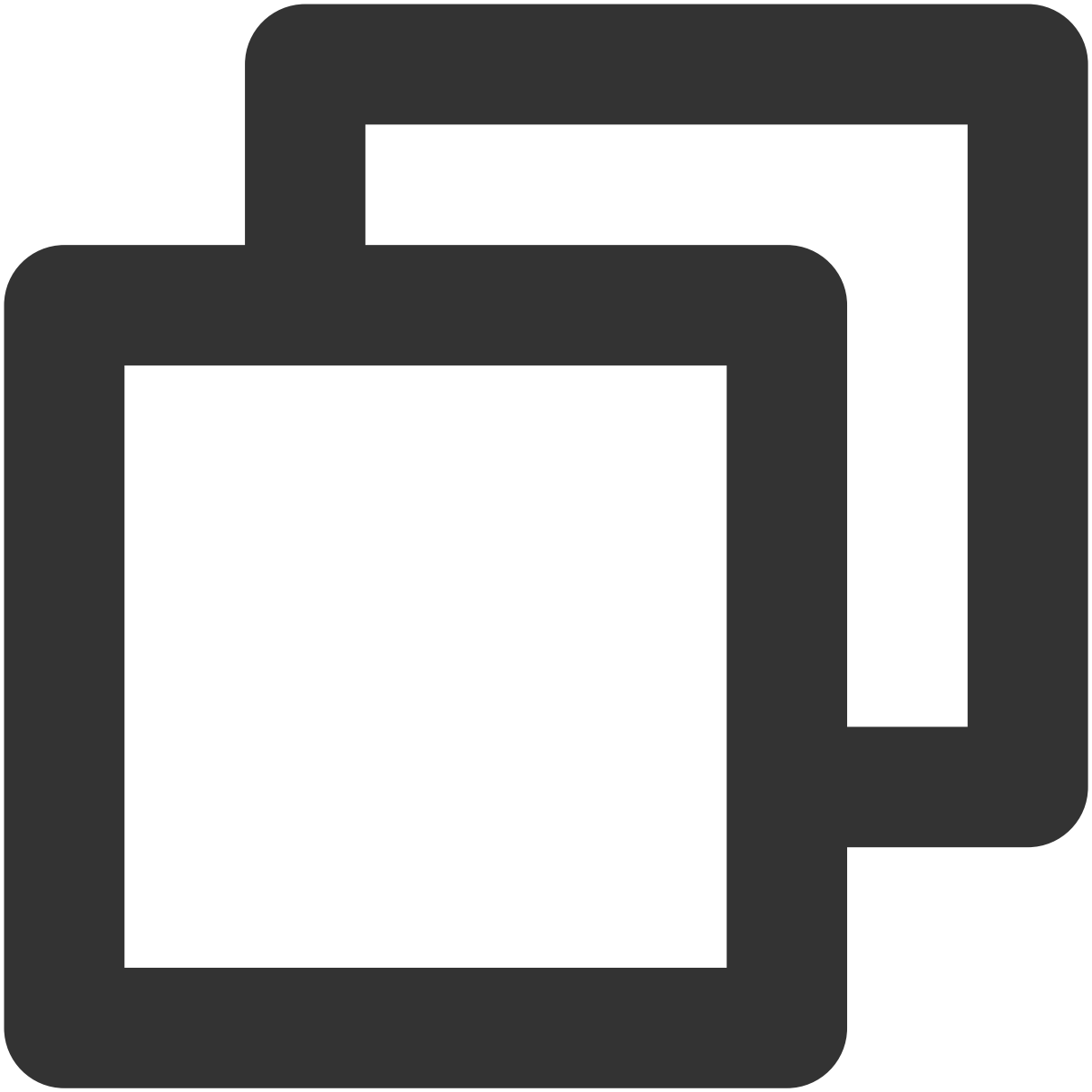
```
// Enable local audio capture and publishing.
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT)

// Stop local audio capture and publishing.
mTRTCCloud.stopLocalAudio();
```

Note:

`startLocalAudio` requests mic permissions, and `stopLocalAudio` releases them.

3. Mute and unmute.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Pause publishing local audio streams (mute).
mTRTCCloud.muteLocalAudio(true);
// Resume publishing local audio streams (unmute).
mTRTCCloud.muteLocalAudio(false);

// Pause the subscription and playback of a specific remote user's audio streams.
mTRTCCloud.muteRemoteAudio(userId, true);
// Resume the subscription and playback of a specific remote user's audio streams.
mTRTCCloud.muteRemoteAudio(userId, false);

// Pause the subscription and playback of all remote users' audio streams.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
mTRTCCloud.muteAllRemoteAudio(true);
// Resume the subscription and playback of all remote users' audio streams.
mTRTCCloud.muteAllRemoteAudio(false);
```

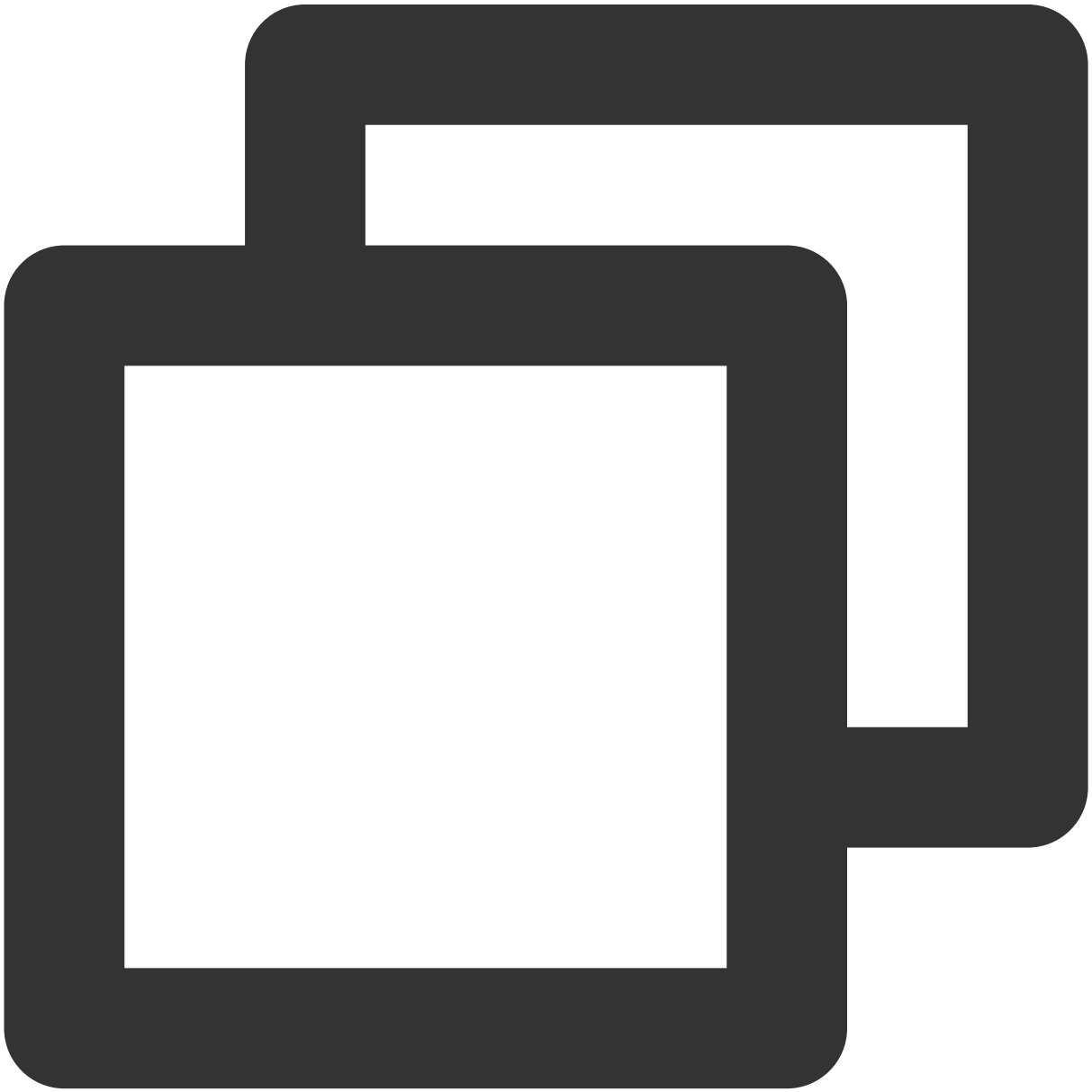
Note:

In comparison, `muteLocalAudio` only requires a pause or release of the data stream at the software level, thus it is more efficient and smoother. And it is better suited for scenarios that require frequent muting and unmuting.

4.

Audio quality and volume type

Audio quality setting



©2013-2022 Tencent Cloud. All rights reserved.

```
// Set audio quality during local audio capture and publishing.
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);

// Dynamically set audio quality during audio streaming.
mTRTCCloud.setAudioQuality(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

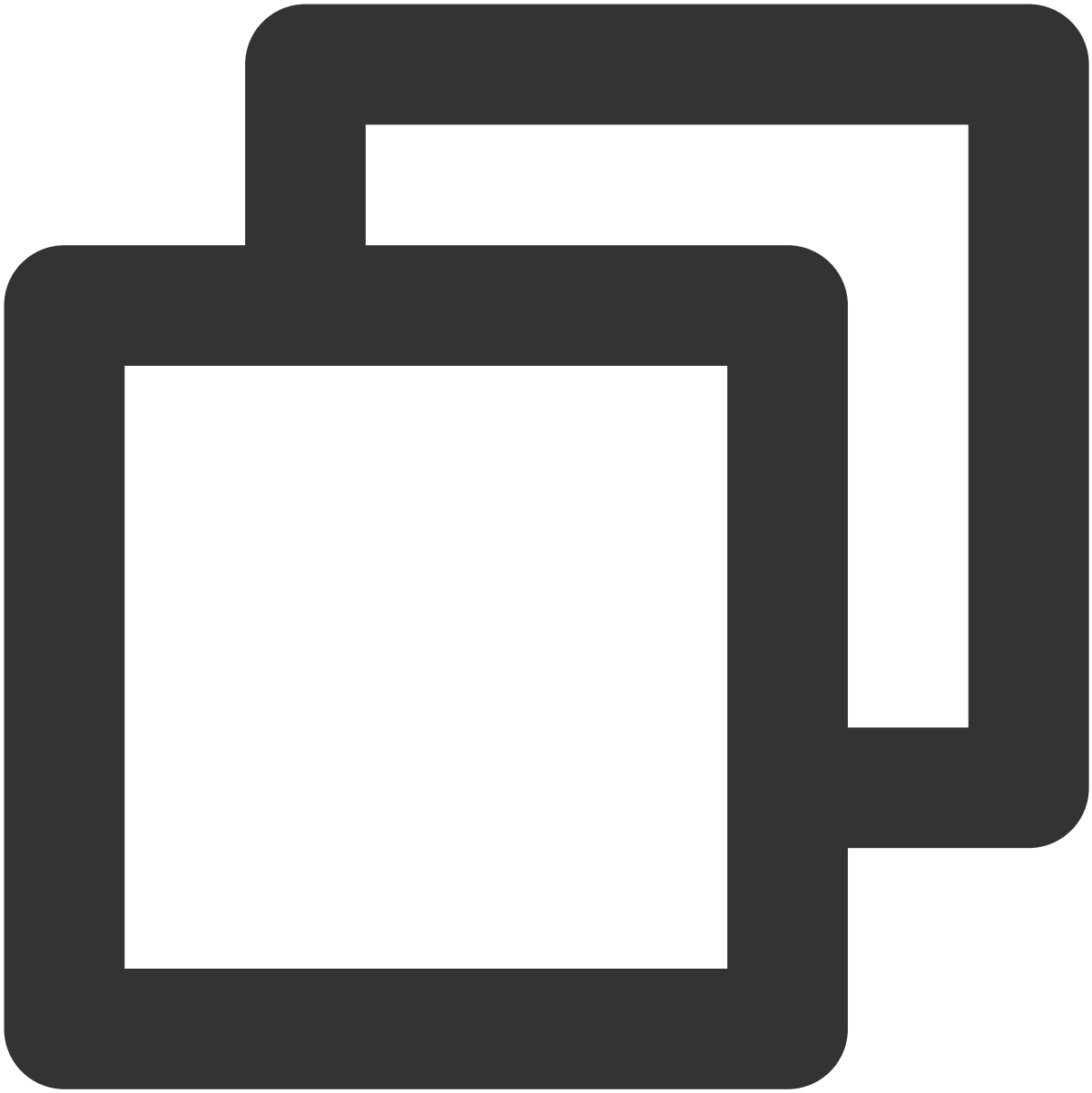
Note:

TRTC's preset audio quality is divided into three levels (Speech/Default/Music), each corresponding to different audio parameters. See TRTCAudioQuality for details.

Volume type setting.

Each TRTC audio quality level corresponds to a default volume type. If you need to forcibly specify a volume type, you can use the following API.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set volume type.
mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeAuto);
```

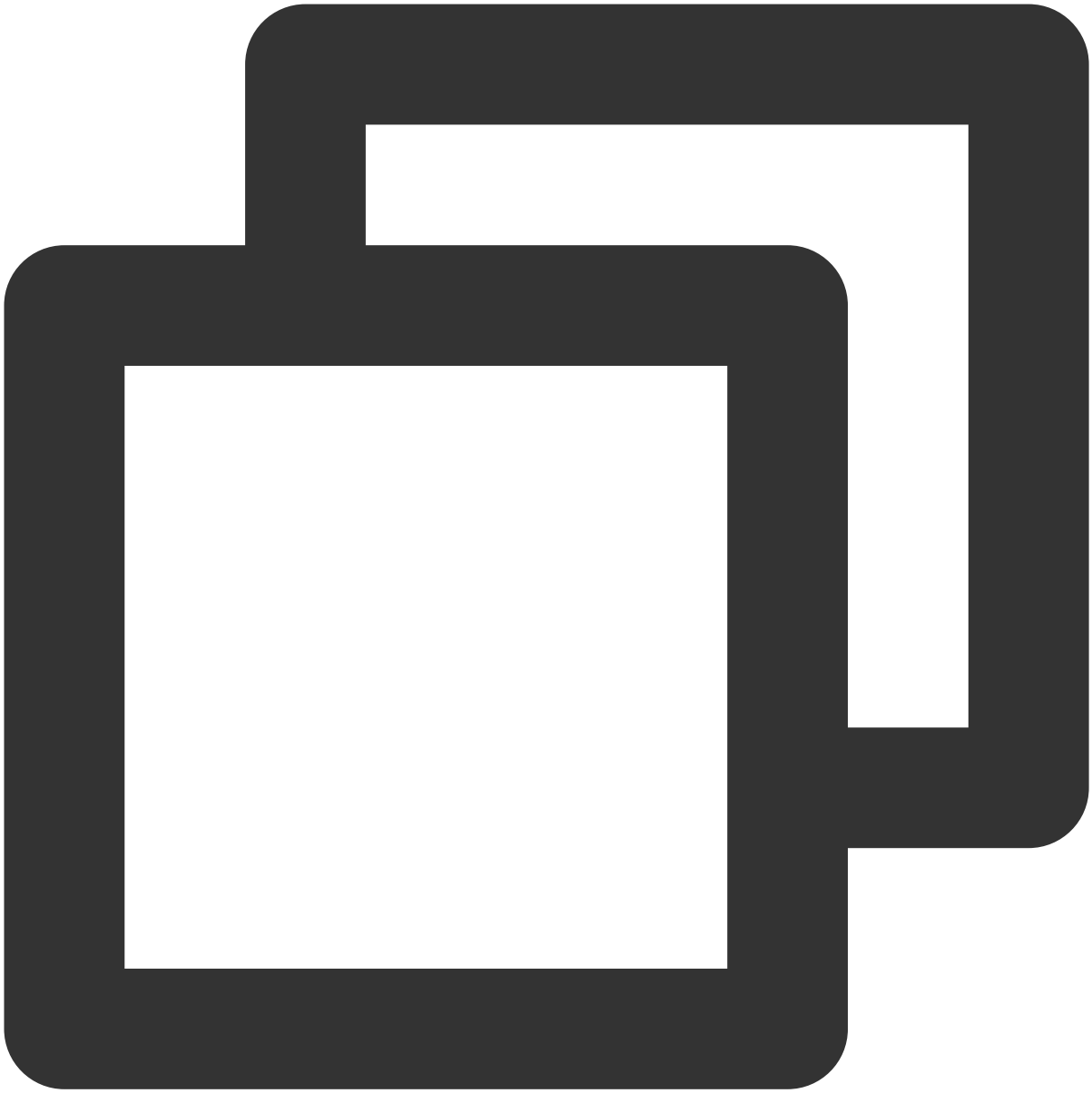
Note:

TRTC volume types are divided into three levels (VOIP/Auto/Media), each corresponding to different volume channels. See TRTCSystemVolumeType for details.

Audio routing setting.

Mobile devices such as smartphones usually have two playback locations: the speaker and the earpiece. If you need to forcibly specify the audio routing, you can use the following API.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set audio routing.
mTRTCCloud.setAudioRoute(TRTCCloudDef.TRTC_AUDIO_ROUTE_SPEAKER);
```

Note:

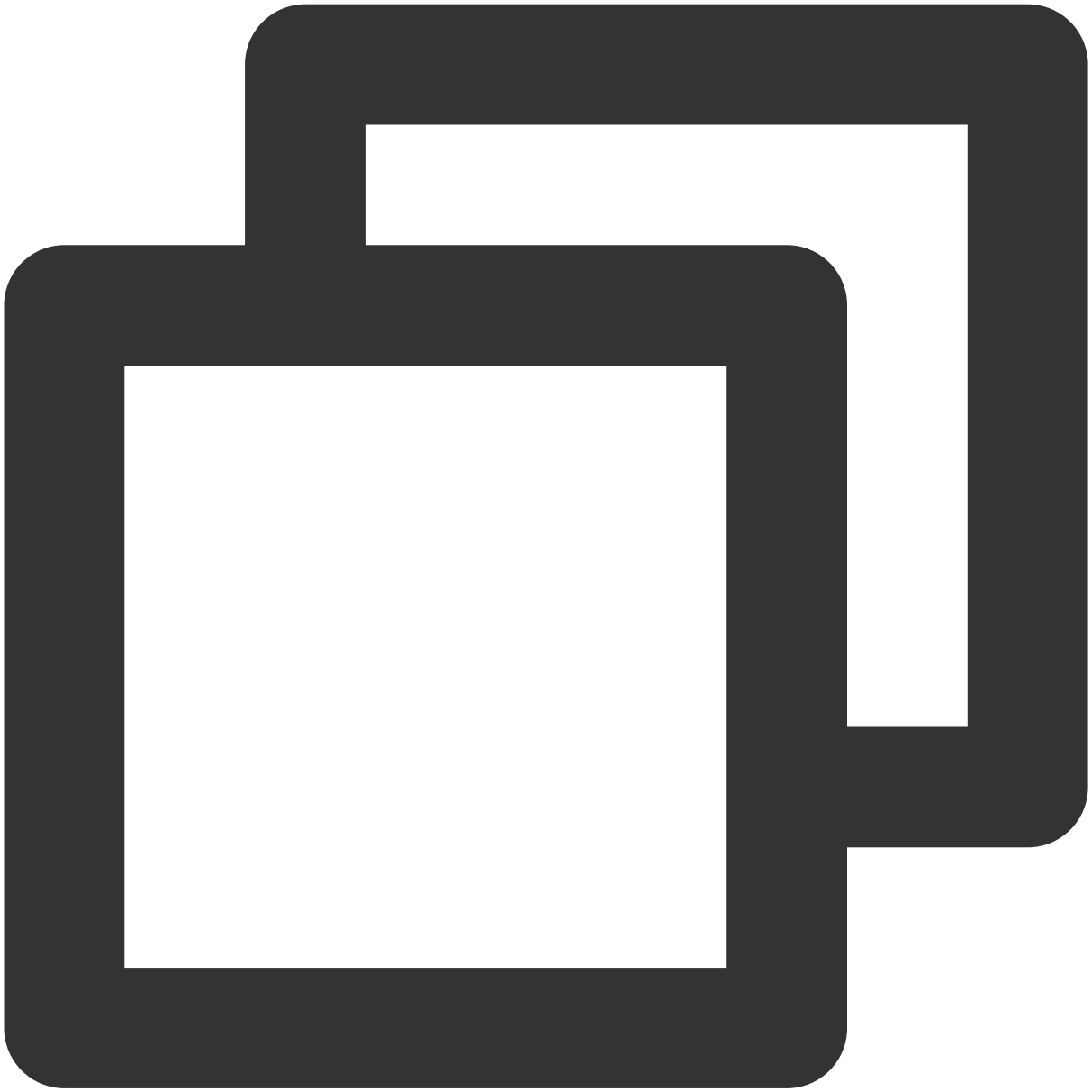
TRTC audio routing is divided into two types (Speaker/Earpiece), each corresponding to a different sound emission location. See TRTCAudioRoute for details.

Advanced Features

©2013-2022 Tencent Cloud. All rights reserved.

Bullet screen message interaction.

Voice chat live streaming rooms usually have text-based bullet screen message interactions. This can be achieved through the sending and receiving of group chat regular text messages via IM.



```
// Send public screen bullet screen messages.
V2TIMManager.getInstance().sendGroupTextMessage(text, groupID, V2TIMMessage.V2TIM_P
    @Override
    public void onError(int i, String s) {
        // Failed to send bullet screen messages.
    }
```

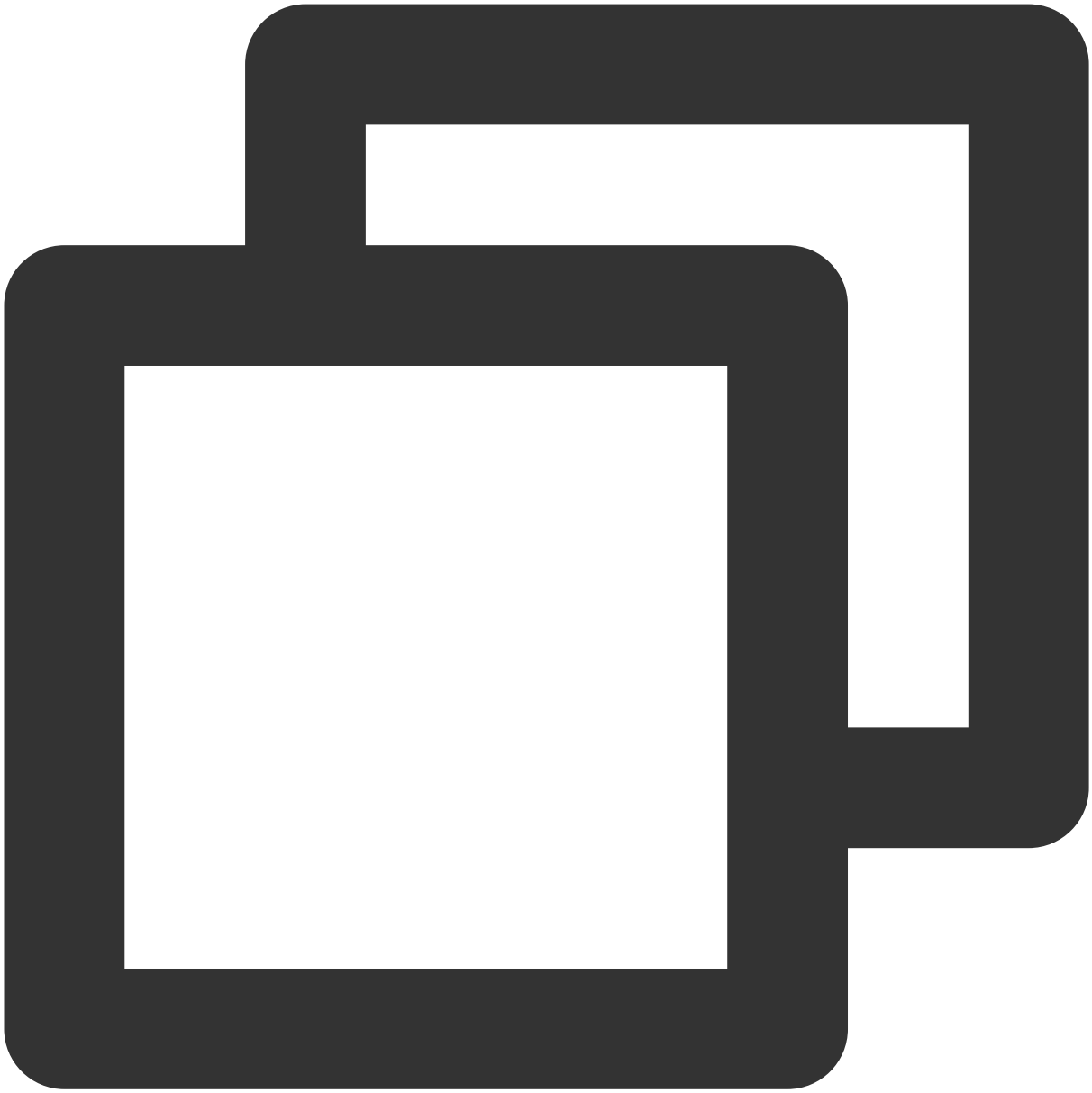
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onSuccess(V2TIMMessage v2TIMMessage) {
        // Successfully sent bullet screen messages.
    }
});

// Receive public screen bullet screen messages.
V2TIMManager.getInstance().addSimpleMsgListener(new V2TIMSimpleMsgListener() {
    @Override
    public void onRecvGroupTextMessage(String msgID, String groupID, V2TIMGroupMemb
        Log.i(TAG, sender.getNickName + ": " + text);
    }
});
```

Volume level callback.

TRTC can callback the volume levels of the on-mic anchor at a fixed frequency. It is usually used to display sound waves and indicate the speaking anchor.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enable volume level callback. It is recommended to be enabled immediately after
// interval: Callback interval (ms). enable_vad: Whether to enable voice detection.
mTRTCCloud.enableAudioVolumeEvaluation(int interval, boolean enable_vad);

private class TRTCCloudImplListener extends TRTCCloudListener {
    public void onUserVoiceVolume(ArrayList<TRTCCloudDef.TRTCVolumeInfo> userVolume
        super.onUserVoiceVolume(userVolumes, totalVolume);
        // userVolumes is used to hold the volume levels of all speaking users, inc
        // totalVolume is used to report the maximum volume value among remote stre
        ...
        // Adjust the corresponding visual representation of sound waves on the UI
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            ...
        }
    }
```

Note:

Voice detection only provides local voice detection results. The user's role must be an anchor to make it convenient to remind users to turn on their mics.

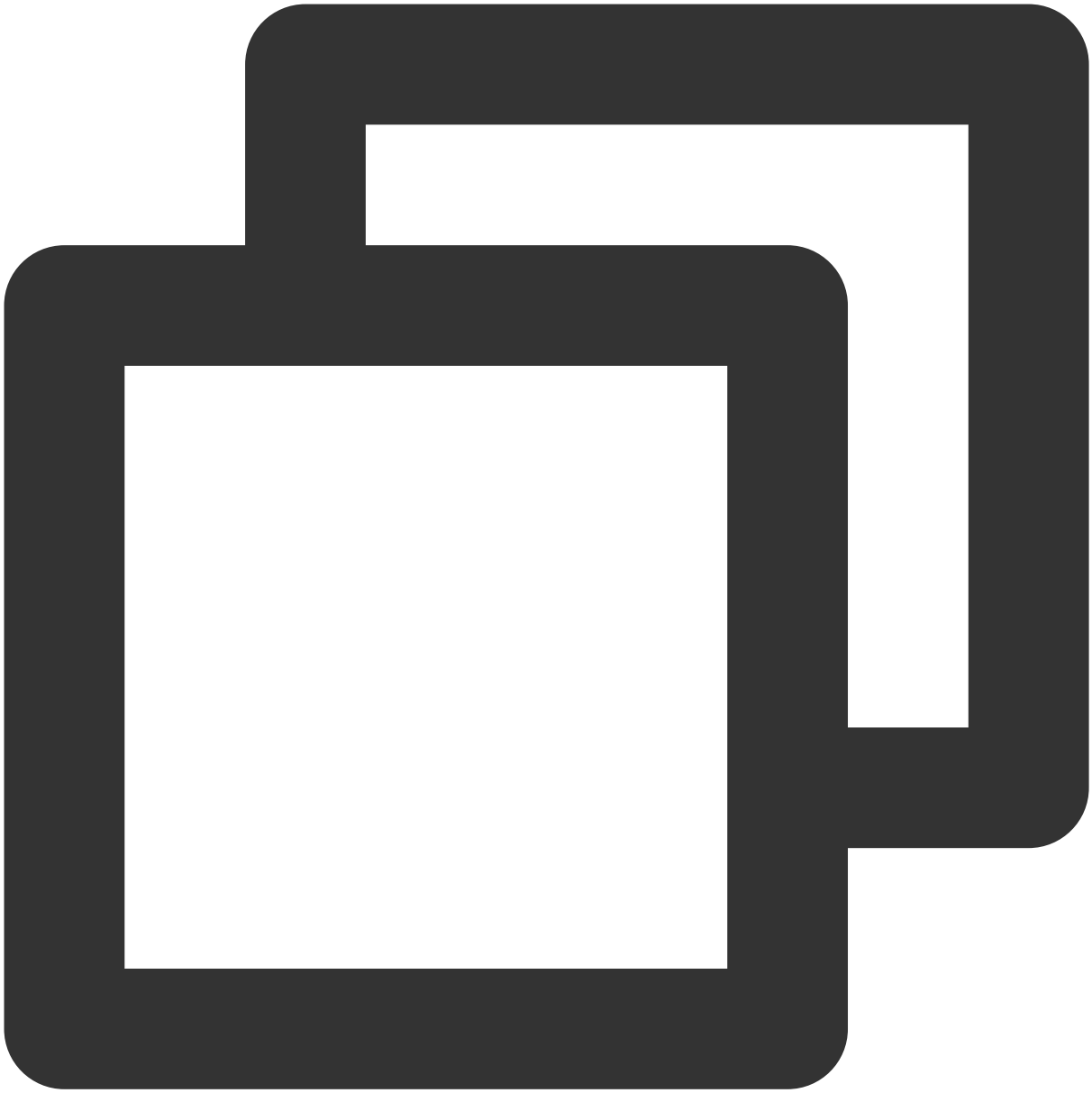
`userVolumes` is an array. For each element in the array, when userId is oneself, it indicates the volume captured by the local microphone; when userId is others, it indicates the volume of remote users.

Music and sound effect playback.

Playing background music and sound effects is a high-frequency demand in voice chat room scenarios. Below, we will explain the use of and precautions for commonly used background music APIs.

1. Start/stop/pause/resume playback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the management class for configuring background music, short sound effect
TXAudioEffectManager mTXAudioEffectManager = mTRTCCloud.getAudioEffectManager();

TXAudioEffectManager.AudioMusicParam param = new TXAudioEffectManager.AudioMusicPar
// Whether to publish the music to remote (otherwise play locally only).
param.publish = true;
// Whether the playback is from a short sound effect file.
param.isShortFile = false;

// Start background music playback.
mTXAudioEffectManager.startPlayMusic(param);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Stop background music playback.
mTXAudioEffectManager.stopPlayMusic(musicID);
// Pause background music playback.
mTXAudioEffectManager.pausePlayMusic(musicID);
// Resume background music playback.
mTXAudioEffectManager.resumePlayMusic(musicID);
```

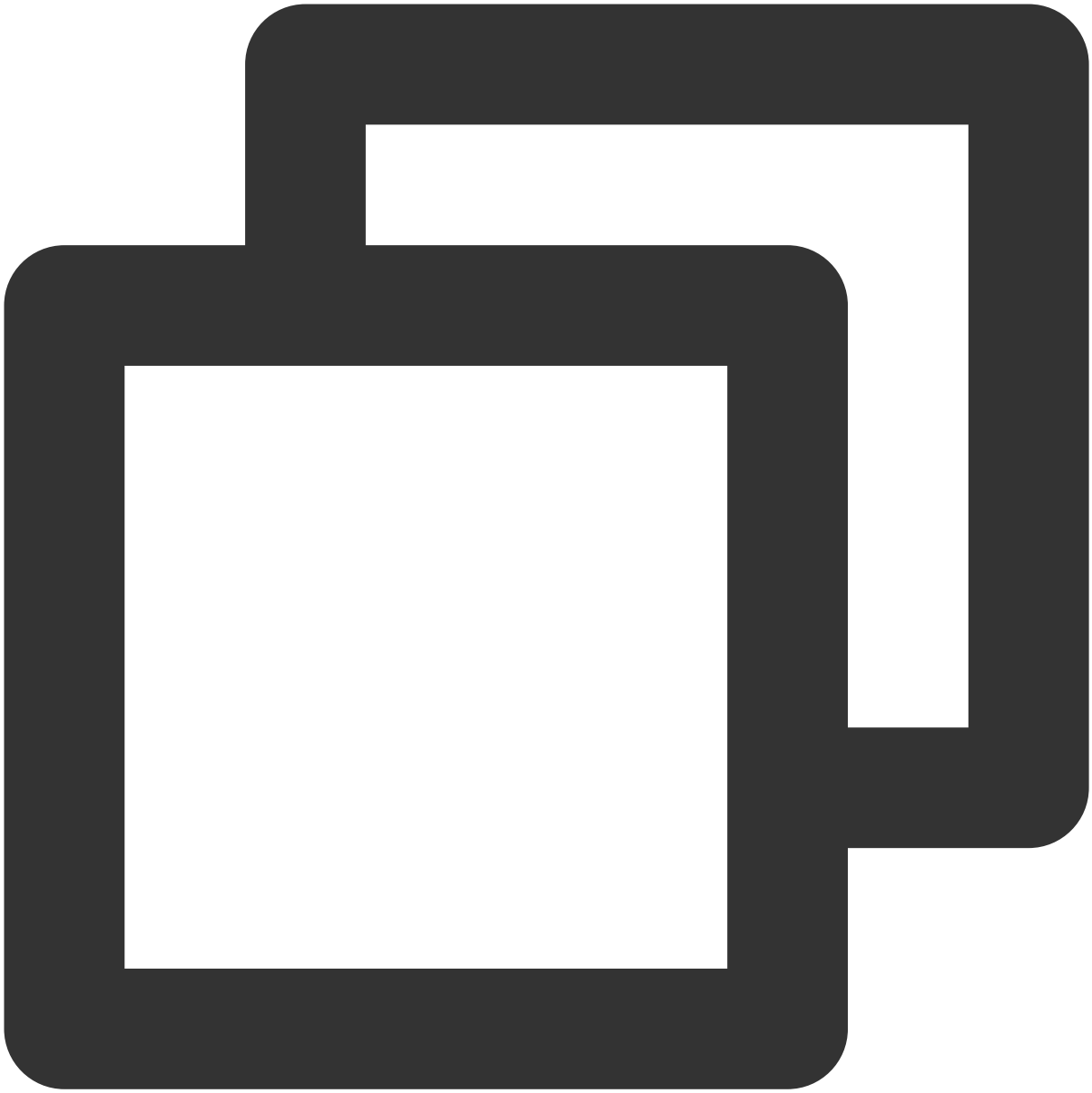
Note:

TRTC supports playing multiple pieces of music simultaneously, each identified uniquely by a musicID. If you want to play only one piece of music at a time, be sure to stop other music before starting playback, or you can use the same musicID to play different music. In this way, the SDK will stop the old music first, and then play the new one.

TRTC supports playing both local and online audio files, by passing in a local absolute path or URL address through `musicPath` . MP3/AAC/M4A/WAV formats are supported.

2. Adjust the ratio of music and voice volume.

©2013-2022 Tencent Cloud. All rights reserved.

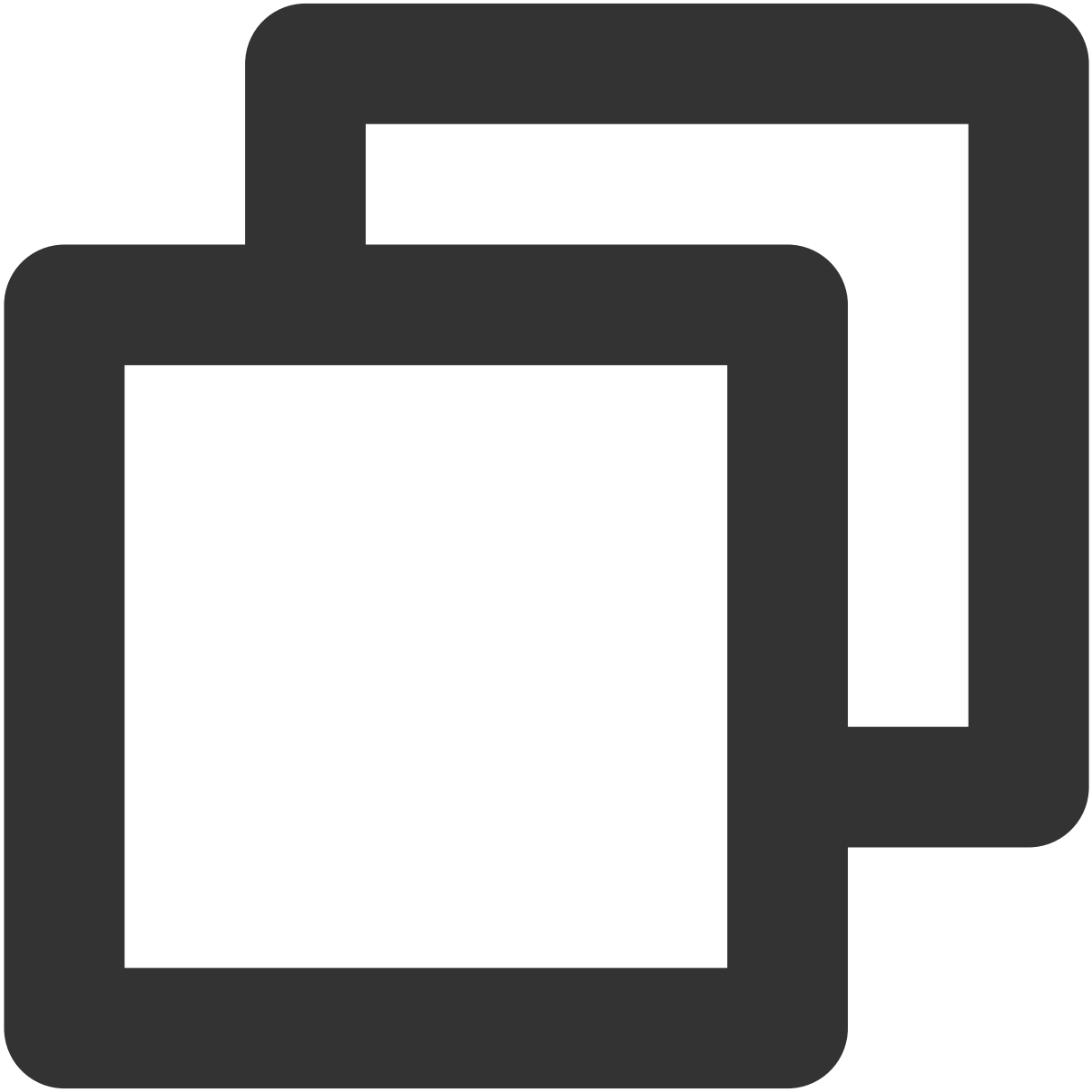


```
// Set the local playback volume of a piece of background music.
mTXAudioEffectManager.setMusicPlayoutVolume(musicID, volume);
// Set the remote playback volume of a specific background music.
mTXAudioEffectManager.setMusicPublishVolume(musicID, volume);
// Set the local and remote volume of all background music.
mTXAudioEffectManager.setAllMusicVolume(volume);
// Set the volume of voice capture.
mTXAudioEffectManager.setVoiceCaptureVolume(volume);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Volume value's normal range is 0-100, with a default of 60 and a maximum setting of 150, but there is a risk of audio clipping.
If background music is overwhelming vocals, consider lowering the music playback volume and increasing the voice capture volume.
Mute microphone without muting background music: Use `setVoiceCaptureVolume(0)` to replace `muteLocalAudio(true)` .
3. Set music playback event callback.



```
mTXAudioEffectManager.setMusicObserver(mCurPlayMusicId, new TXAudioEffectManager.TX
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Background music starts playing.
    public void onStart(int id, int errCode) {
        // -4001: Path opening failed.
        // -4002: Decoding failed.
        // -4003: Invalid URL address.
        // -4004: Playback not stopped.
        if (errCode < 0) {
            // Before replaying after playback failure, you must first stop the cur
            mTXAudioEffectManager.stopPlayMusic(id);
        }
    }

    @Override
    // The playback progress of background music.
    public void onPlayProgress(int id, long curPtsMs, long durationMs) {
        // curPtsMS current playback duration (in milliseconds).
        // durationMs: Total duration of the current music (milliseconds).
    }

    @Override
    // Background music has finished playing.
    public void onComplete(int id, int errCode) {
        // Playback failure due to weak network during playback will also throw thi
        // Pausing or stopping playback midway will not trigger the onComplete call
    }
});
```

Note:

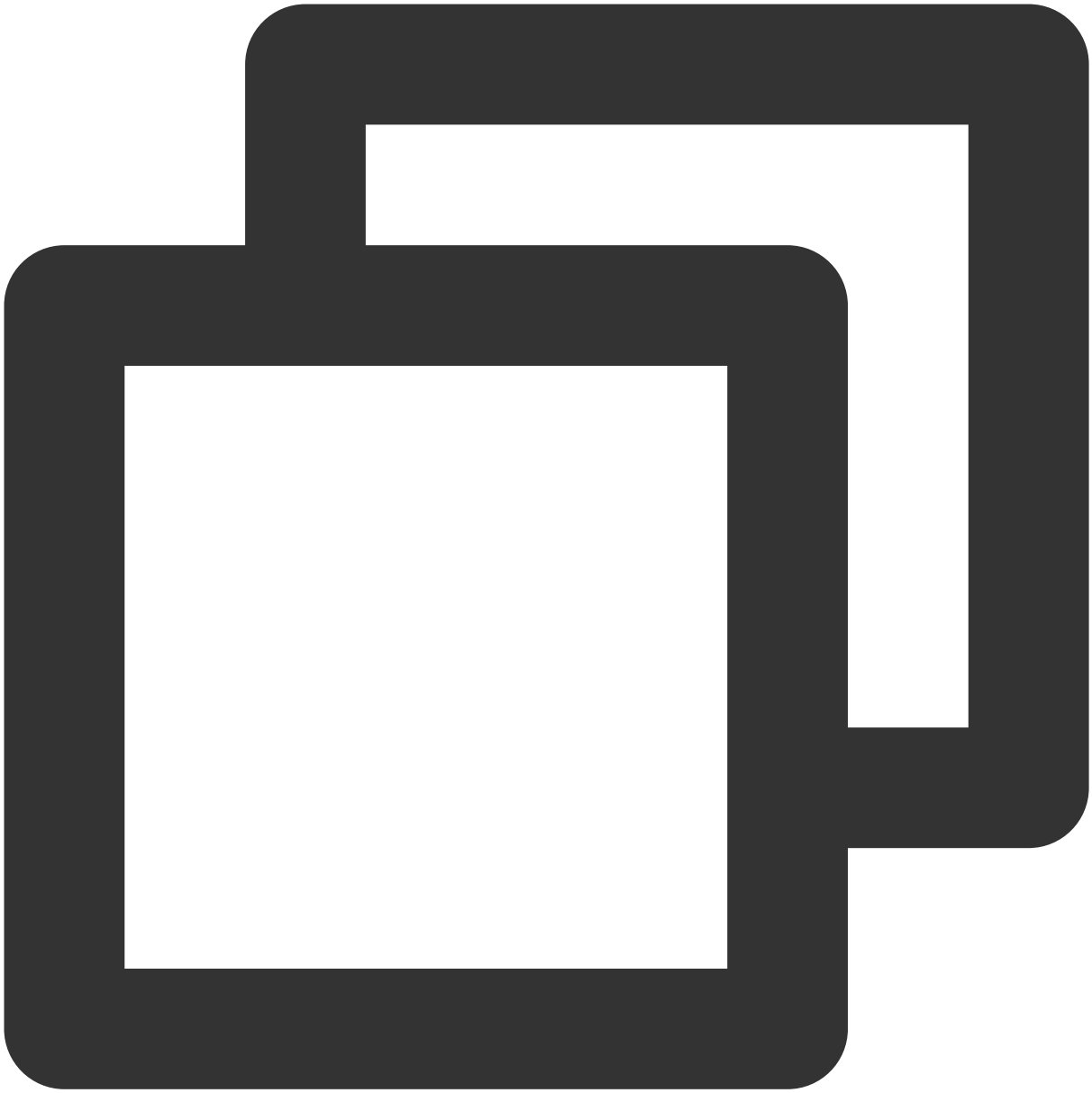
Use this API to set the playback event callback before playing background music to monitor the progress of the music; If the MusicId does not need to be reused, you can execute `setMusicObserver(musicId, null)` after playback is finished to completely release the Observer.

4. Loop playback of background music and sound effects.

Solution 1: Use the `AudioMusicParam` 's `loopCount` parameter to set the number of loop playbacks.

The value range is from 0 to any positive integer. The default value is 0. 0 means play the music once; 1 means play the music twice; and so on.

©2013-2022 Tencent Cloud. All rights reserved.



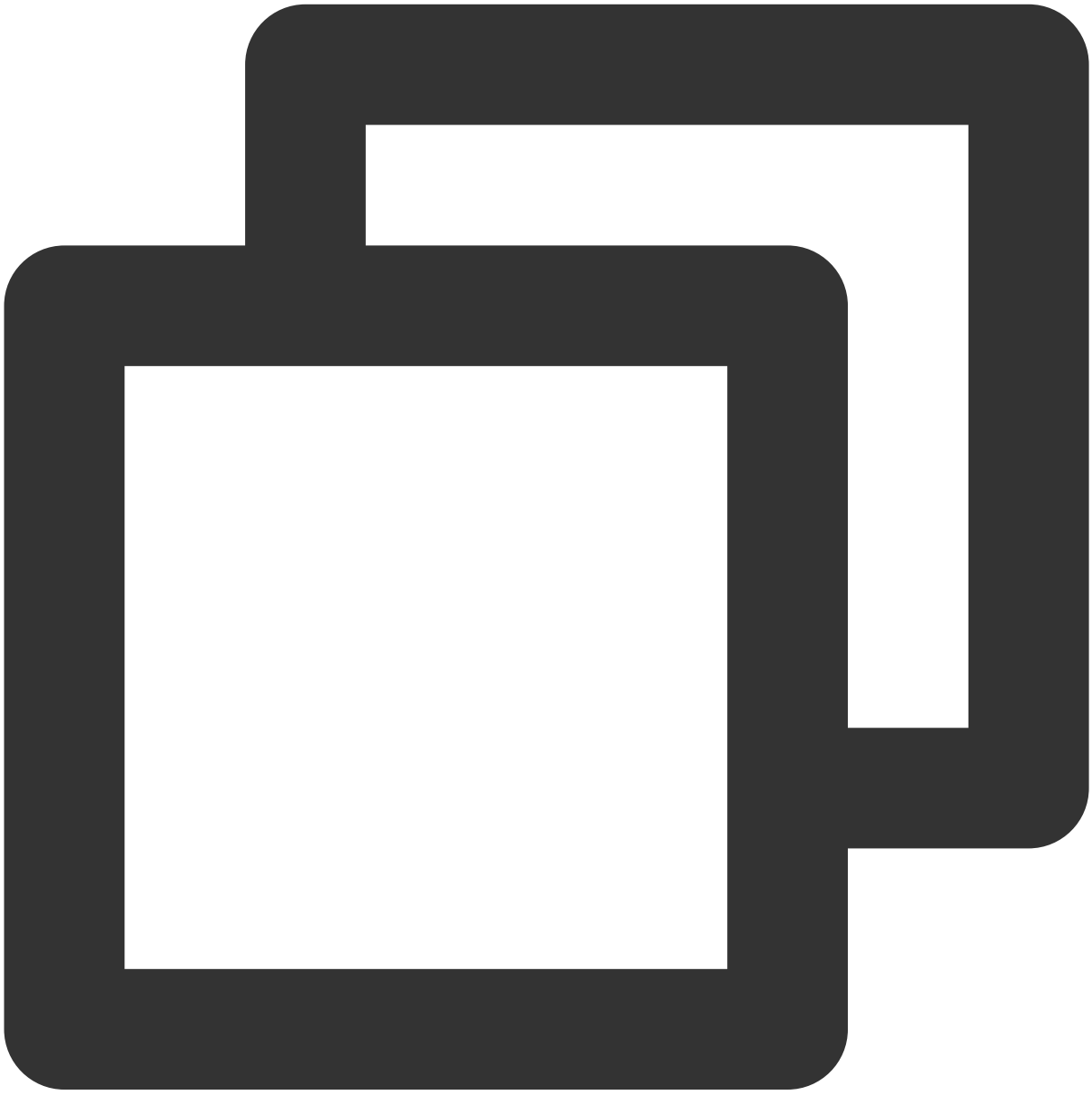
```
private void startPlayMusic(int id, String path, int loopCount) {
    TXAudioEffectManager.AudioMusicParam param = new TXAudioEffectManager.AudioMusi
    // Whether to publish music to the remote.
    param.publish = true;
    // Whether the playback is from a short sound effect file.
    param.isShortFile = true;
    // Set the number of loop playbacks. Negative number means an infinite loop.
    param.loopCount = loopCount < 0 ? Integer.MAX_VALUE : loopCount;
    mTRTCCloud.getAudioEffectManager().startPlayMusic(param);
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

Solution 1 will not trigger the `onComplete` callback after each loop playback. It will only be triggered after all the set loop counts have been played.

Solution 2: Implement loop playback through the "Background music has finished playing" event callback `onComplete`. It is usually used for list loop or single track loop.



```
// The member variable used for indicating whether to loop playback.
private boolean loopPlay;

private void startPlayMusic(int id, String path) {
```

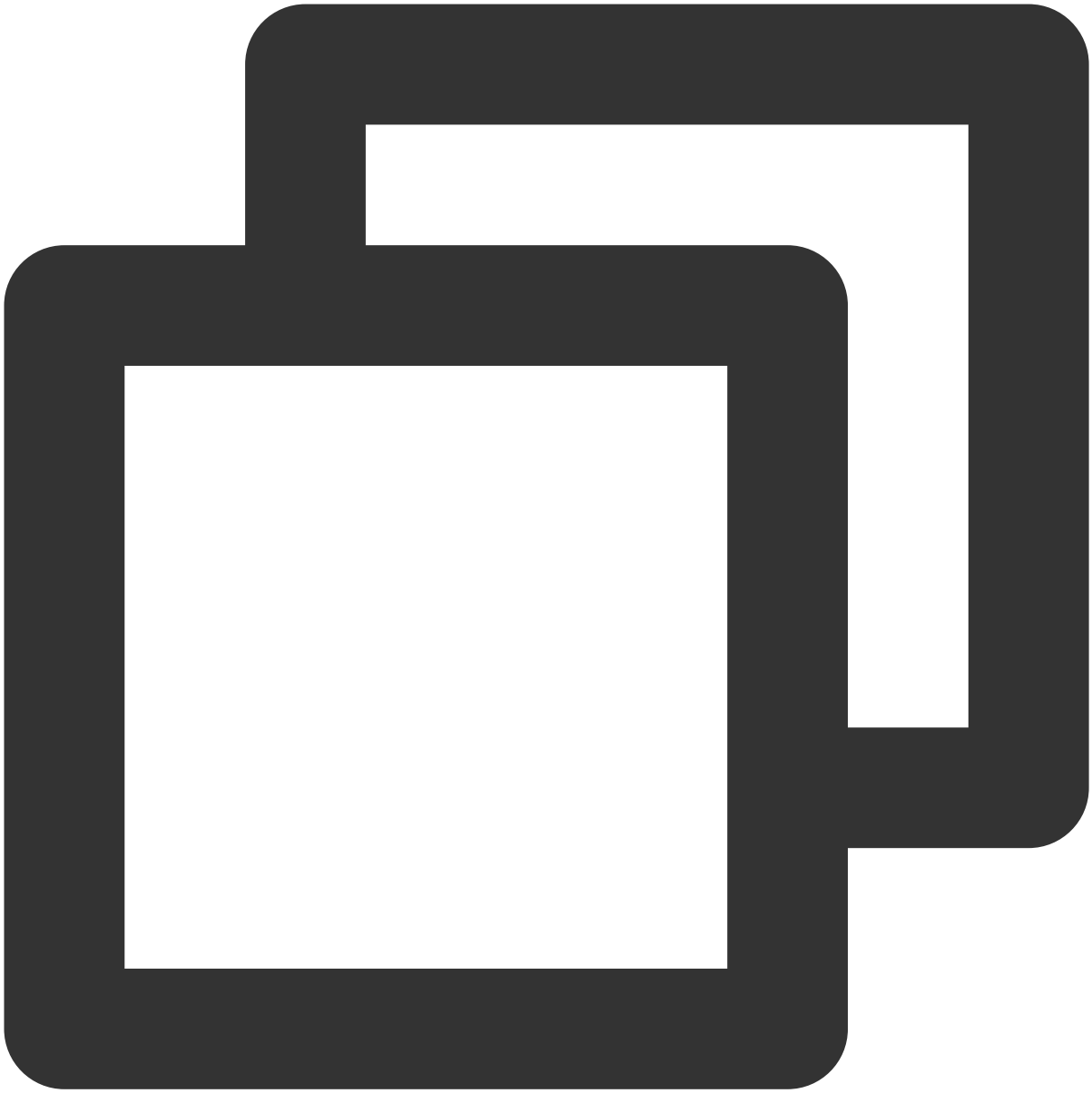
©2013-2022 Tencent Cloud. All rights reserved.

```
    TXAudioEffectManager.AudioMusicParam param = new TXAudioEffectManager.AudioMusi
    mTXAudioEffectManager.setMusicObserver(id, new MusicPlayObserver(id, path));
    mTXAudioEffectManager.startPlayMusic(param);
}

private class MusicPlayObserver implements TXAudioEffectManager.TXMusicPlayObserver
    private final int mId;
    private final String mPath;

    public MusicPlayObserver(int id, String path) {
        mId = id;
        mPath = path;
    }

    @Override
    public void onStart(int i, int i1) {

    }

    @Override
    public void onPlayProgress(int i, long l, long l1) {

    }

    @Override
    public void onComplete(int i, int i1) {
        mTXAudioEffectManager.stopPlayMusic(i);
        if (i1 >= 0 && loopPlay) {
            // Here, you can replace the ID or Path of the music in the loop playli
            startPlayMusic(mId, mPath);
        }
    }
}
```

Mixed stream relay and push back.

1. Live streaming CDN with mixed stream relay.

©2013-2022 Tencent Cloud. All rights reserved.



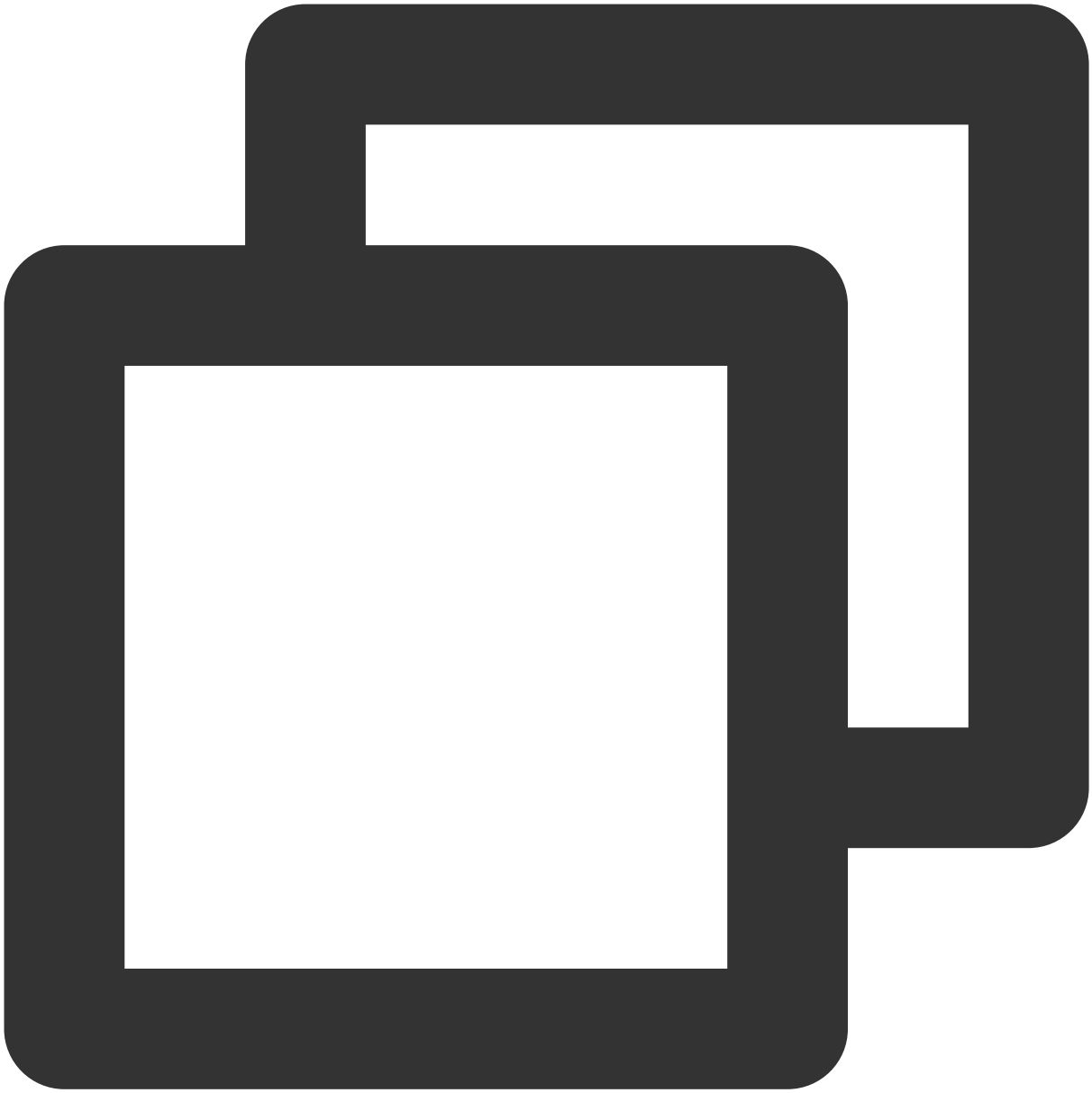
```
private void startPublishMediaToCDN(String streamName) {
    // Expiration time of the streaming URL. By default, it is one day.
    long txTime = (System.currentTimeMillis() / 1000) + (24 * 60 * 60);
    // LIVE_URL_KEY authentication key. Obtain from the streaming URL configuration
    String secretParam = UrlHelper.getSafeUrl(LIVE_URL_KEY, streamName, txTime);

    // The target URLs for media stream publication.
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // Publish to CDN after mixing.
    target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToCdn;
    TRTCCloudDef.TRTCPublishCdnUrl cdnUrl = new TRTCCloudDef.TRTCPublishCdnUrl();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Streaming URL must include parameters. Otherwise, streaming fails.
    cdnUrl.rtmpUrl = "rtmp://" + PUSH_DOMAIN + "/live/" + streamName + "?" + secret
    // True means Tencent CSS push URLs, and false means third-party services.
    cdnUrl.isInternalLine = true;
    // Multiple CDN push URLs can be added.
    target.cdnUrlList.add(cdnUrl);

    // Set the encoding parameters of the transcoded audio stream (can be customize
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
    trtcStreamEncoderParam.audioEncodedChannelNum = 1;
    trtcStreamEncoderParam.audioEncodedKbps = 50;
    trtcStreamEncoderParam.audioEncodedCodecType = 0;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;

    // Set the encoding parameters of the transcoded video stream (must be filled i
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 3;
    trtcStreamEncoderParam.videoEncodedKbps = 30;
    trtcStreamEncoderParam.videoEncodedWidth = 64;
    trtcStreamEncoderParam.videoEncodedHeight = 64;

    // Configuration parameters for media stream transcoding.
    TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new TRTCCloudDef.T
    // By default, leave this field empty. It indicates that all audio in the room
    trtcStreamMixingConfig.audioMixUserList = null;

    // Must have TRTCVideoLayout parameters if mixing black frames (can be ignored
    TRTCCloudDef.TRTCVideoLayout videoLayout = new TRTCCloudDef.TRTCVideoLayout();
    trtcStreamMixingConfig.videoLayoutList.add(videoLayout);

    // Start mixing and relaying mixed streams.
    mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam, trtcStreamMi
}
```

2. Push the mixed stream back to the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



```
private void startPublishMediaToRoom(String roomId, String userId) {
    // Create TRTCPublishTarget object.
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // After mixing, the stream is relayed back to the room.
    target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToRoom;
    target.mixStreamIdentity.strRoomId = roomId;
    // Mixed stream robot's userid, must not duplicate with other users' userid in
    target.mixStreamIdentity.userId = userId + MIX_ROBOT;

    // Set the encoding parameters of the transcoded audio stream (can be customize
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
```

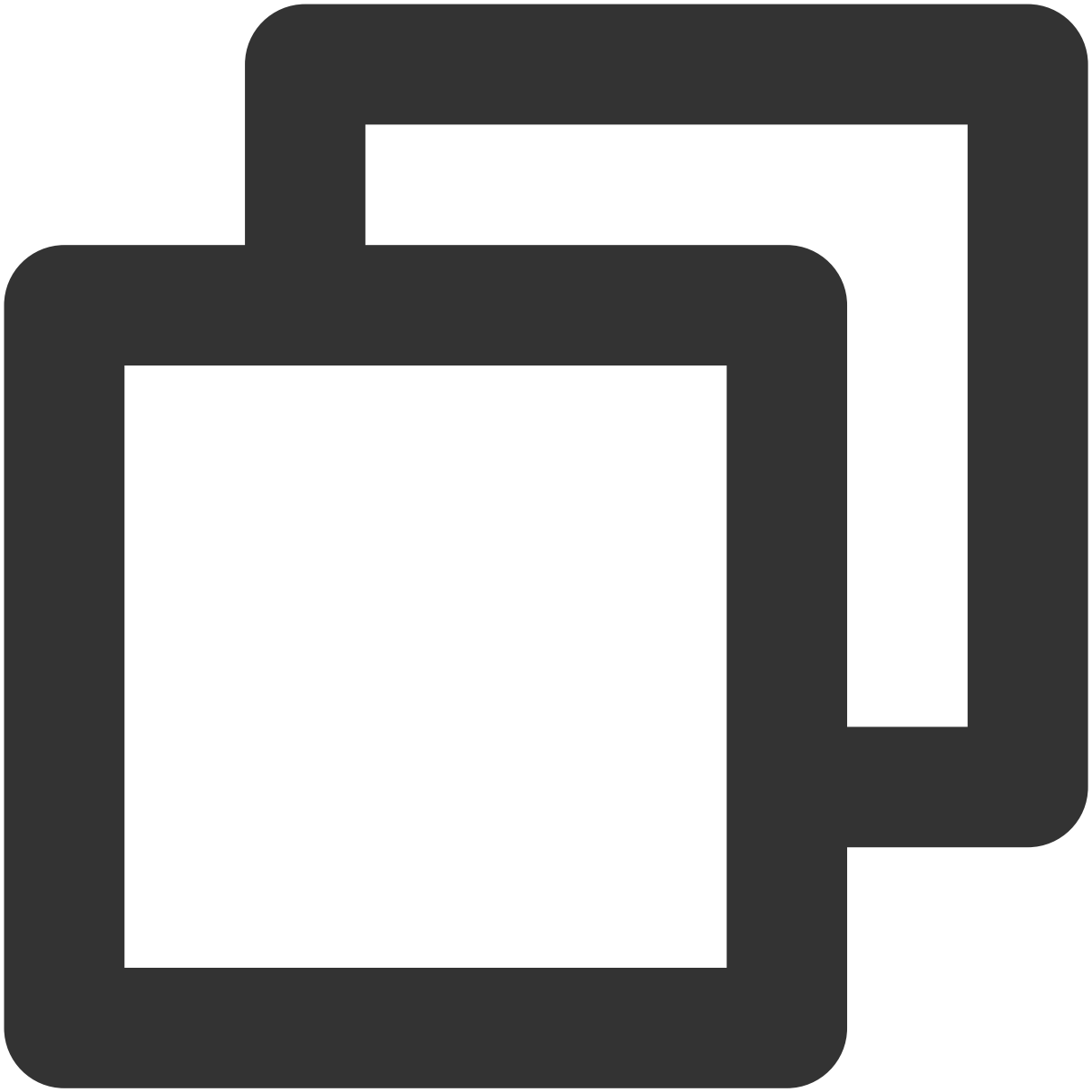
©2013-2022 Tencent Cloud. All rights reserved.

```
    trtcStreamEncoderParam.audioEncodedChannelNum = 2;
    trtcStreamEncoderParam.audioEncodedKbps = 64;
    trtcStreamEncoderParam.audioEncodedCodecType = 2;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;

    // Set the encoding parameters of the transcoded video stream (can be ignored f
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 3;
    trtcStreamEncoderParam.videoEncodedKbps = 30;
    trtcStreamEncoderParam.videoEncodedWidth = 64;
    trtcStreamEncoderParam.videoEncodedHeight = 64;

    // Set audio mixing parameters.
    TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new TRTCCloudDef.T
    // By default, leave this field empty. It indicates that all audio in the room
    trtcStreamMixingConfig.audioMixUserList = null;

    // Configure video mixing template (can be ignored for pure audio mix stream).
    TRTCCloudDef.TRTCVideoLayout videoLayout = new TRTCCloudDef.TRTCVideoLayout();
    trtcStreamMixingConfig.videoLayoutList.add(videoLayout);

    // Start mixing and pushing back.
    mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam, trtcStreamMi
}
```

3. Event callback and update stop task.

Task result event callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
private class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onStartPublishMediaStream(String taskId, int code, String message,
        // taskId: When the request is successful, TRTC backend will provide the ta
        // code: Callback result. 0 means success and other values mean failure.
    }

    @Override
    public void onUpdatePublishMediaStream(String taskId, int code, String message,
        // When you call the publish media stream API (updatePublishMediaStream), t
        // code: Callback result. 0 means success and other values mean failure.
```

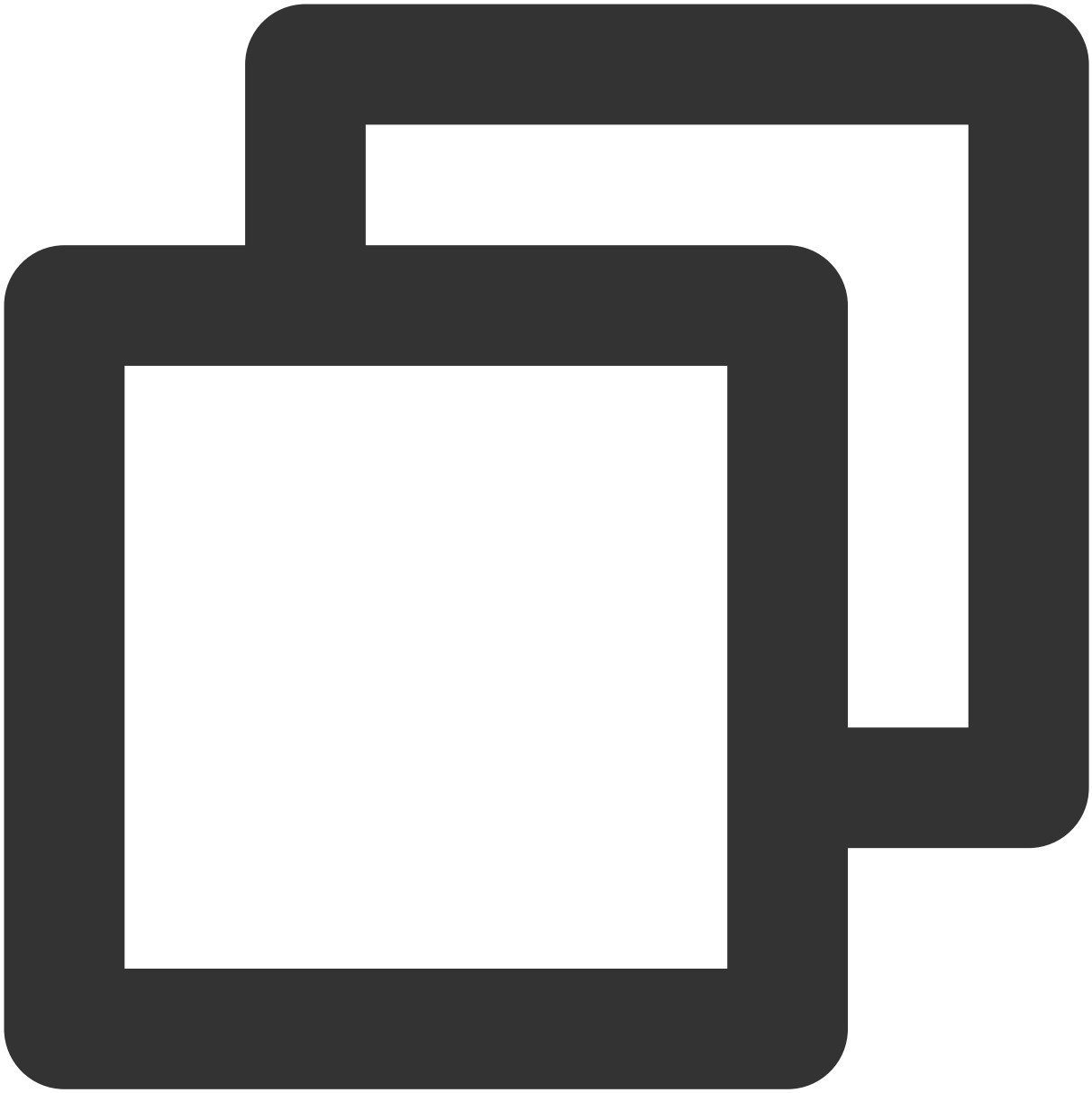
©2013-2022 Tencent Cloud. All rights reserved.

```
    }

    @Override
    public void onStopPublishMediaStream(String taskId, int code, String message, B
        // When you call the stop publishing media stream API (stopPublishMediaStre
        // code: Callback result. 0 means success and other values mean failure.
    }
}
```

Update the published media stream.

This API sends a command to the TRTC server to update the media stream initiated by `startPublishMediaStream` .

©2013-2022 Tencent Cloud. All rights reserved.



```
// taskId: Task ID returned by the onStartPublishMediaStream callback.
// target: For example, add or remove the published CDN URLs.
// params: It is recommended to maintain consistency in the encoding output paramet
// config: Update the list of users involved in mix stream transcoding, such as cro
mTRTCCloud.updatePublishMediaStream(taskId, target, trtcStreamEncoderParam, trtcStr
```

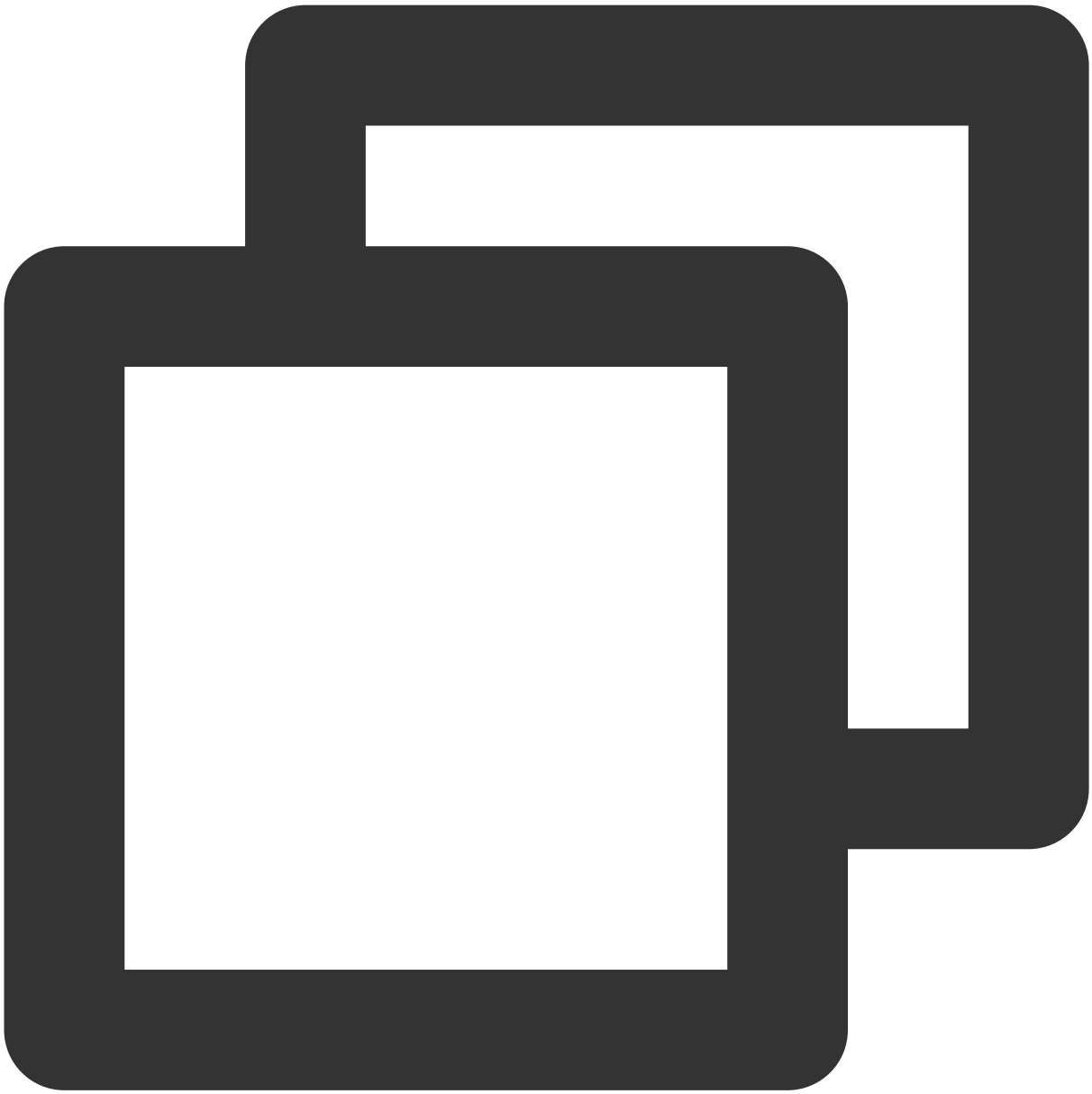
Note:

Switching between audio only, audio and video, and video only is not supported within the same task.

Stop publishing media stream.

©2013-2022 Tencent Cloud. All rights reserved.

This API sends a command to the TRTC server to stop the media stream initiated by `startPublishMediaStream`.



```
// taskId: Task ID returned by the onStartPublishMediaStream callback.
mTRTCCloud.stopPublishMediaStream(taskId);
```

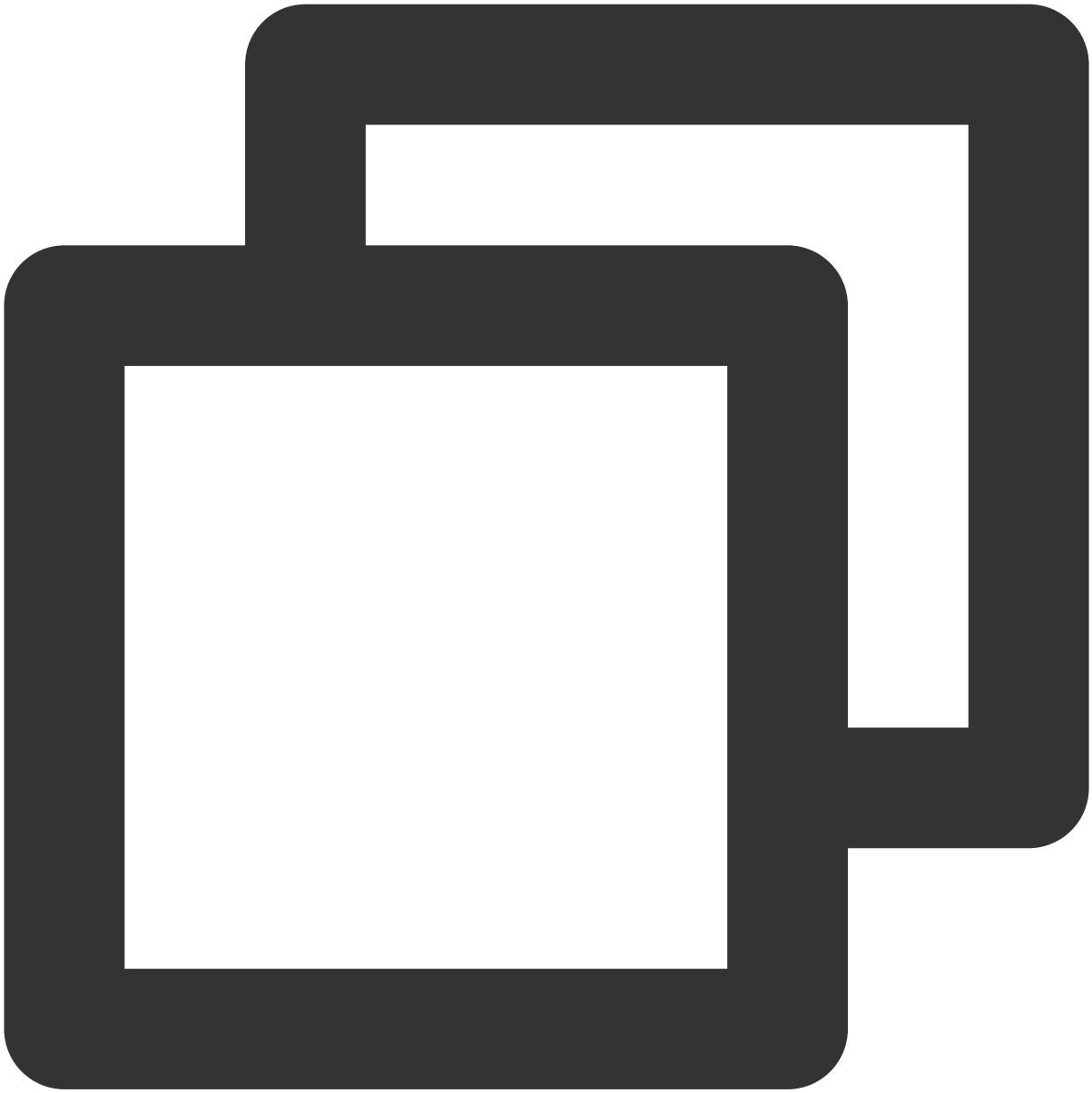
Note:

If taskId is filled with an empty string, it will stop all media streams initiated by the user through `startPublishMediaStream`. If you have only initiated one media stream or want to stop all media streams initiated by you, this method is recommended.

©2013-2022 Tencent Cloud. All rights reserved.

Real-time network quality callback

You can listen to `onNetworkQuality` to real-time monitor the network quality of both local and remote users. This callback is thrown every 2 seconds.



```
private class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality,
                                 ArrayList<TRTCCloudDef.TRTCQuality> remoteQuality)
        // localQuality userId is empty. It represents the local user's network qua
        // remoteQuality represents the remote user's network quality evaluation re
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        switch (localQuality.quality) {
            case TRTCCloudDef.TRTC_QUALITY_Excellent:
                Log.i(TAG, "The current network is excellent.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Good:
                Log.i(TAG, "The current network is good.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Poor:
                Log.i(TAG, "The current network is moderate.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Bad:
                Log.i(TAG, "The current network is poor.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Vbad:
                Log.i(TAG, "The current network is very poor.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Down:
                Log.i(TAG, "The current network does not meet the minimum requireme
                break;
            default:
                Log.i(TAG, "Undefined");
                break;
        }
    }
}
```

Advanced permission control

TRTC advanced permission control can be used to set different entry permissions for different rooms, such as advanced VIP rooms. It can also be used to control the permission for the audience to speak, such as handling ghost microphones.

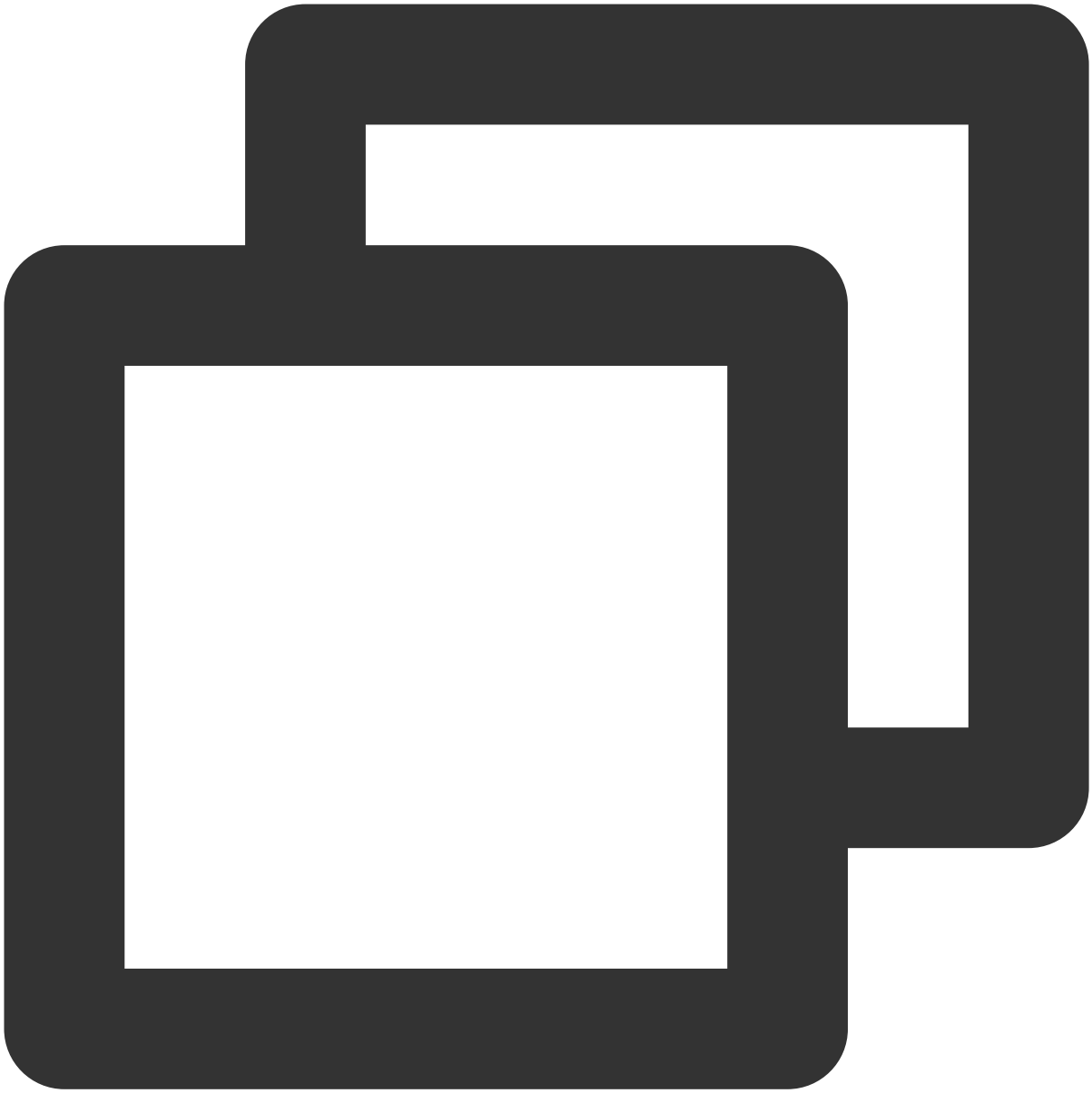
Step 1: Enable the Advanced Permission Control Switch in the TRTC console application's feature configuration page.

Step 2: Generate privateMapKey on the backend. For sample code, see privateMapKey computation source code.

Step 3: Room entry verification & speaking permission verification with PrivateMapKey.

Room entry verification

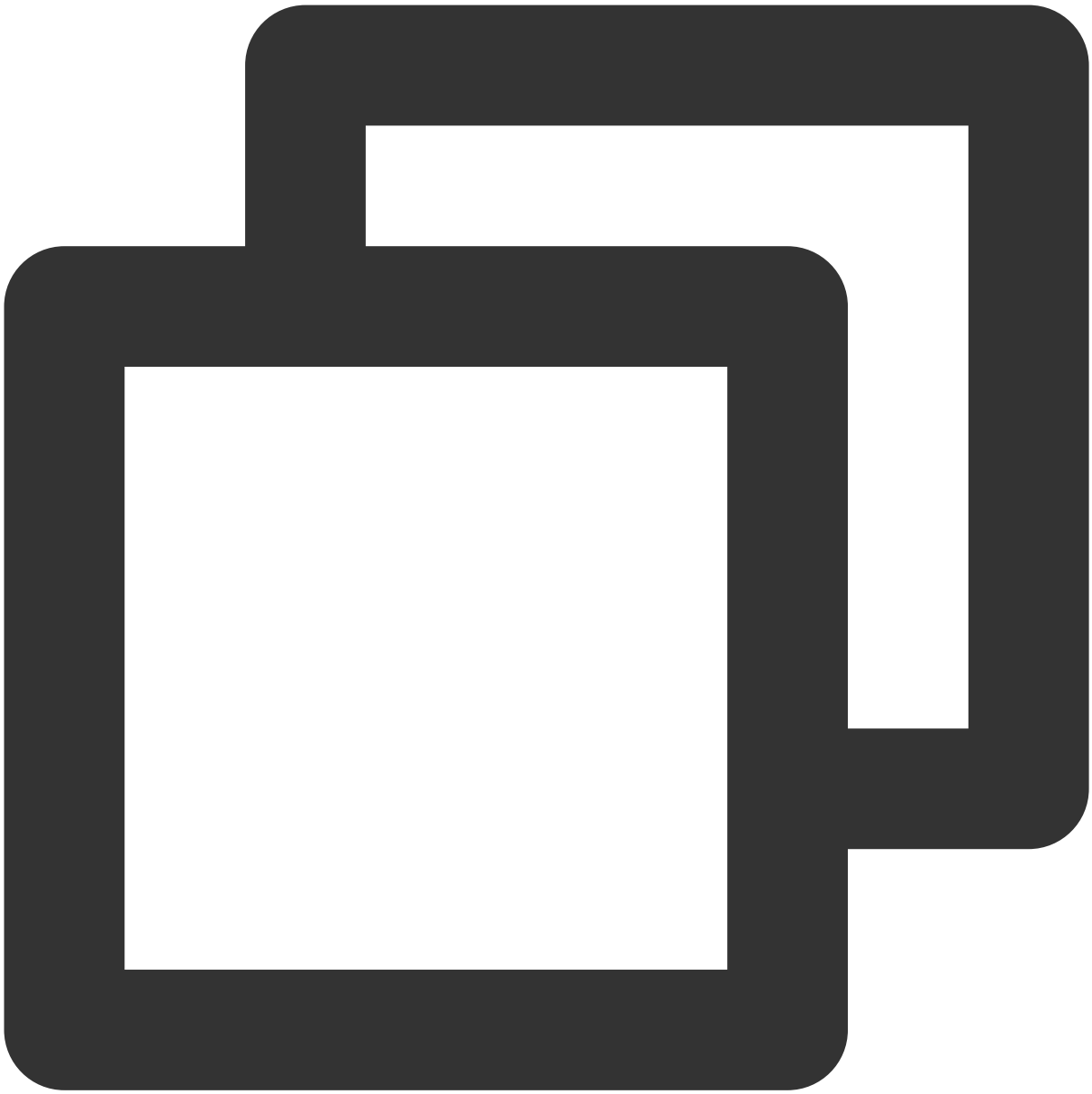
©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCParams mTRTCParams = new TRTCCloudDef.TRTCParams();
mTRTCParams.sdkAppId = SDKAPPID;
mTRTCParams.userId = mUserId;
mTRTCParams.strRoomId = mRoomId;
// UserSig obtained from the business backend.
mTRTCParams.userSig = getUserSig();
// PrivateMapKey obtained from the backend.
mTRTCParams.privateMapKey = getPrivateMapKey();
mTRTCParams.role = TRTCCloudDef.TRTCRoleAudience;
mTRTCCloud.enterRoom(mTRTCParams, TRTCCloudDef.TRTC_APP_SCENE_VOICE_CHATROOM);
```

©2013-2022 Tencent Cloud. All rights reserved.

Speaking permission verification



```
// Pass in the latest PrivateMapKey obtained from the backend into the role switchi
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor, getPrivateMapKey());
```

Exception Handling

Exception error handling

©2013-2022 Tencent Cloud. All rights reserved.

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description

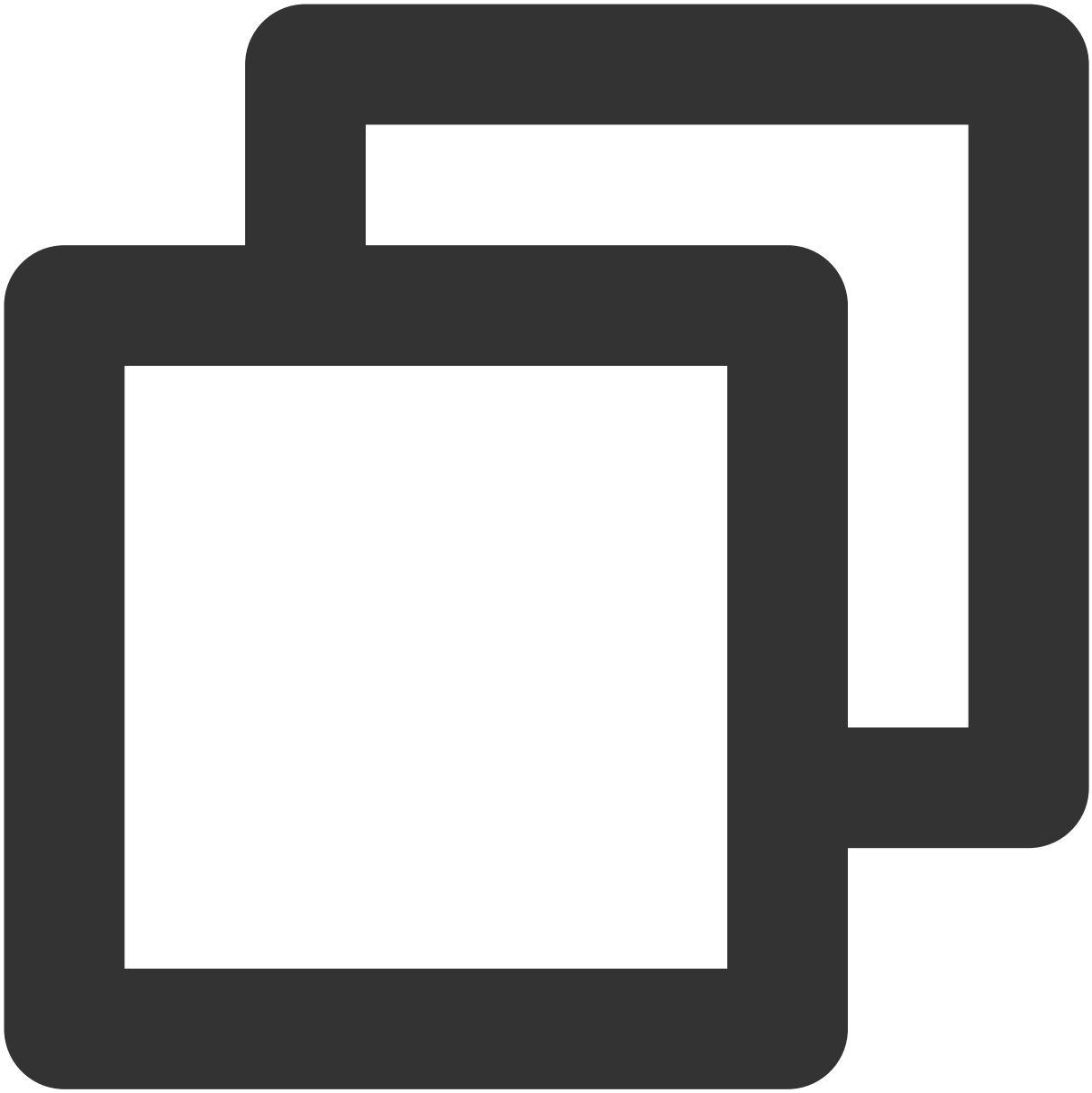
©2013-2022 Tencent Cloud. All rights reserved.

ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_SPEAKER_START_FAIL	-1321	Failed to open the speaker. For example, if there is an exception for the speaker's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Exception exit handling.

1. Network disconnection detection and timeout room exit.
You can listen for TRTC disconnection and reconnection events through the following callback notifications.
Upon receiving the `onConnectionLost` callback, display a network disconnection icon on the local seat UI to notify the user. Simultaneously, initiate a local timer. If the `onConnectionRecovery` callback is not received after exceeding the set time threshold, it means the network remains disconnected. Then, locally initiate leaving the seatmic and room exit process. Pop up a window to inform the user that they have exited the room and the page will be closed.
If the disconnection exceeds 90 seconds (default), a timeout room-exit will be triggered, and the TRTC server will remove the user from the room. If the user has an anchor role, other users in the room will receive the `onRemoteUserLeaveRoom` callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
private class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onConnectionLost() {
        // The connection between the SDK and the cloud has been disconnected.
    }

    @Override
    public void onTryToReconnect() {
        // The SDK is attempting to reconnect to the cloud.
    }
```

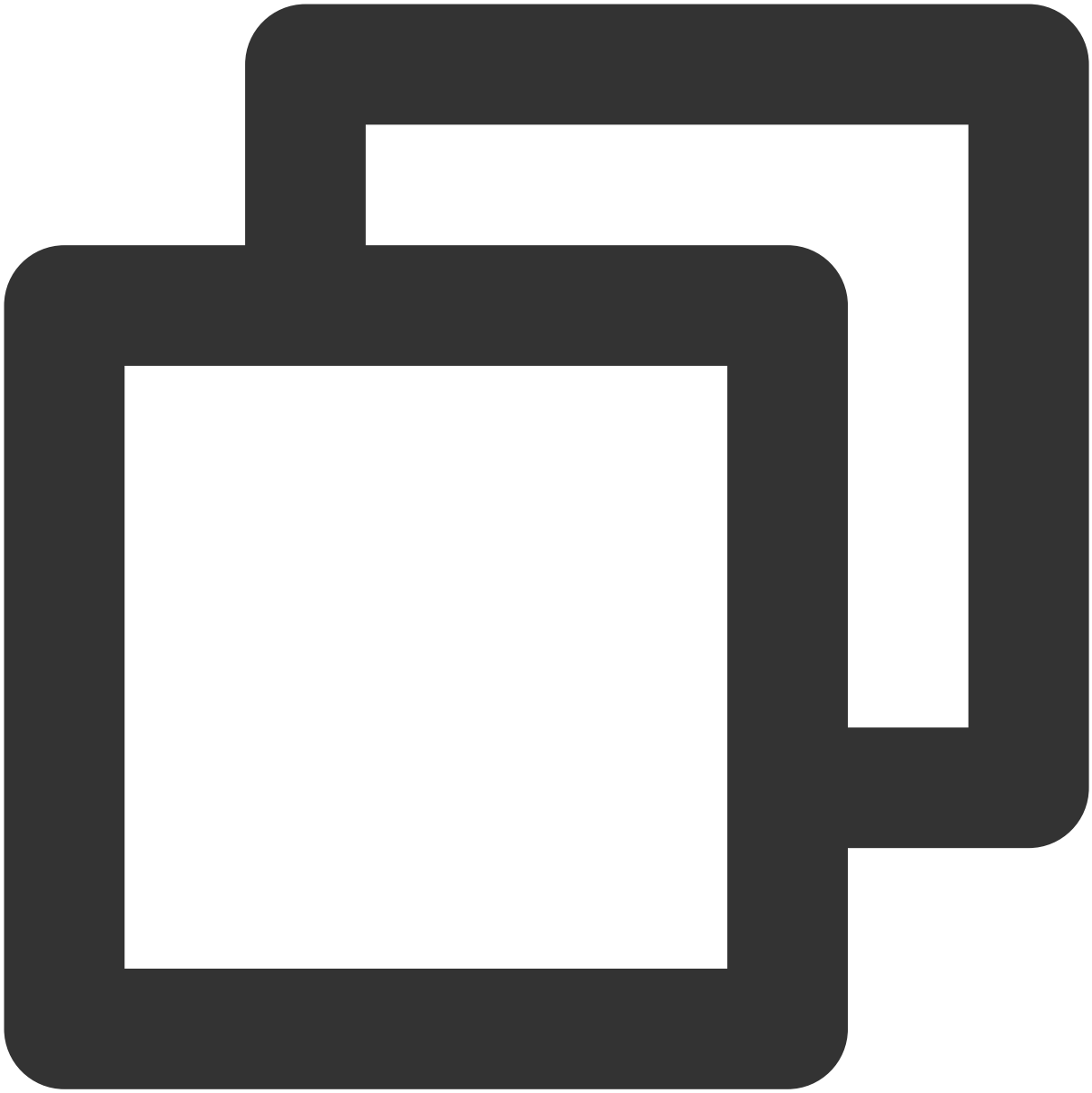
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onConnectionRecovery() {
        // The connection between the SDK and the cloud has been restored.
    }
}
```

2. Automatically remove an offline-status user.

The regular statuses of IM users include online (ONLINE), offline (OFFLINE), and not logged in (UNLOGINED). The offline status typically results from the user force-stopping the process or experiencing an abnormal network disruption. You may use the feature of anchors subscribing to the connection status of mic-connecting audiences to detect offline mic-connecting audiences. And then you may remove them.

©2013-2022 Tencent Cloud. All rights reserved.

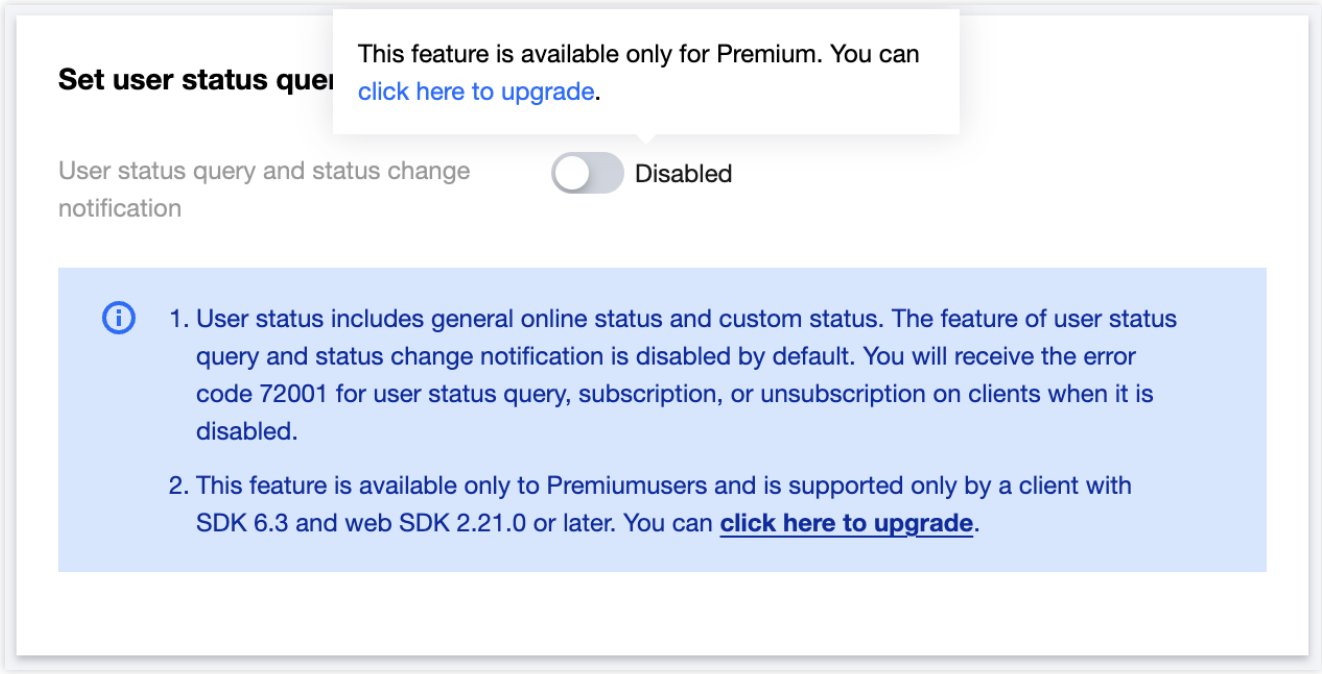


```
// Anchor subscribes to the connection status of mic-connecting audiences.
V2TIMManager.getInstance().subscribeUserStatus(userList, new V2TIMCallback() {
    @Override
    public void onSuccess() {
        // Subscription of user status succeeded.
    }

    @Override
    public void onError(int code, String message) {
        // Subscription of user status failed.
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
});

// Anchor unsubscribes from the connection status of audiences leaving the seat.
V2TIMManager.getInstance().unsubscribeUserStatus(userList, new V2TIMCallback() {
    @Override
    public void onSuccess() {
        // Unsubscription of user status succeeded.
    }

    @Override
    public void onError(int code, String message) {
        // Failed to unsubscription of user status.
    }
});

// User status change notification and processing.
V2TIMManager.getInstance().addIMSDKListener(new V2TIMSDKListener() {
    @Override
    public void onUserStatusChanged(List<V2TIMUserStatus> userStatusList) {
        for (V2TIMUserStatus userStatus : userStatusList) {
            final String userId = userStatus.getUserID();
            int status = userStatus.getStatusType();
            if (status == V2TIMUserStatus.V2TIM_USER_STATUS_OFFLINE) {
                // Remove an offline-status user.
                kickSeat(getSeatIndexFromUserId(userId));
            }
        }
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.



Note:

User-status subscription needs to be upgraded to the advanced package. For details, see Basic Service Details.

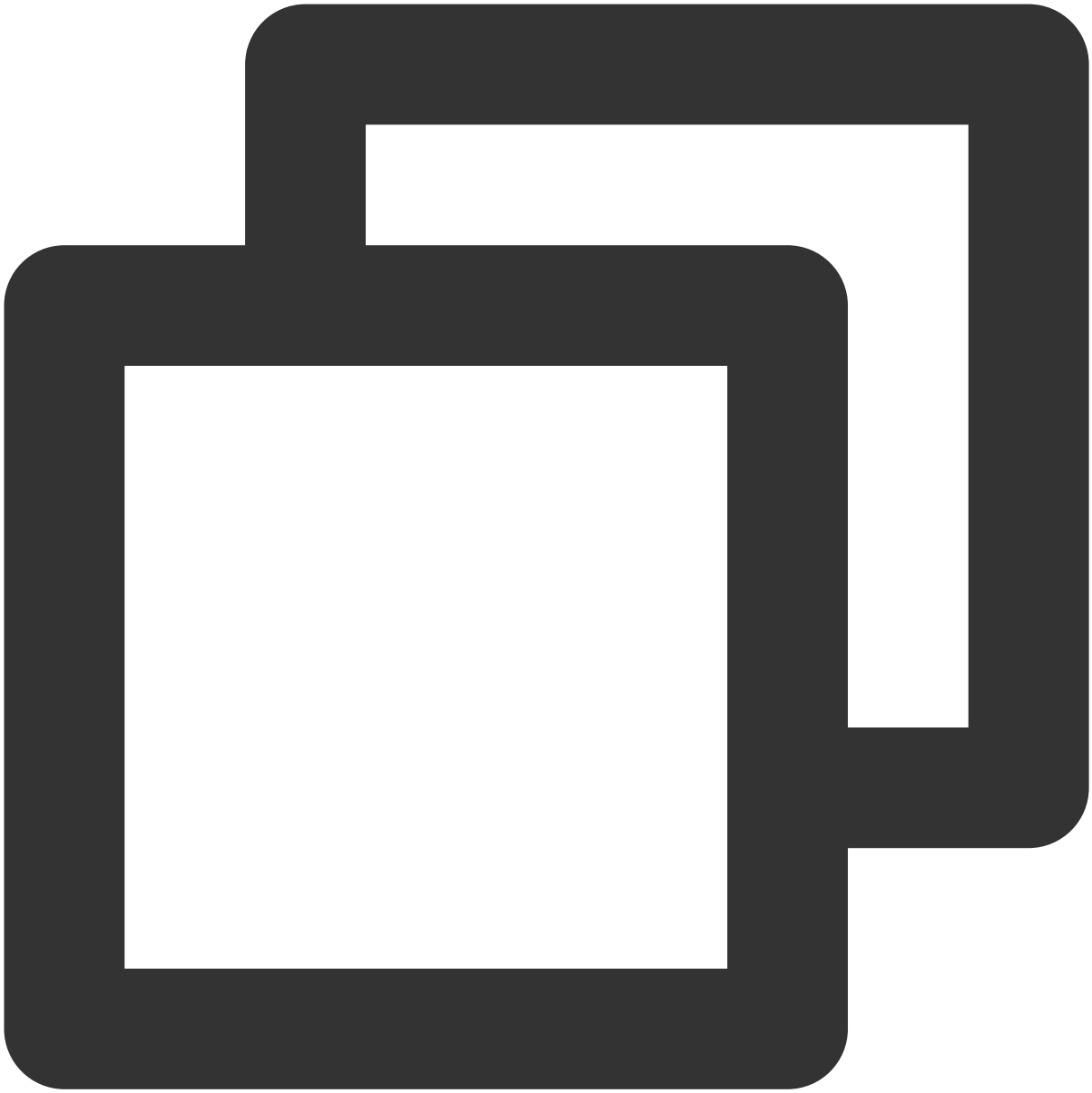
User-status subscription needs **User status query and status change notification configuration** to be enabled in Instant Messaging (IM) console in advance. Failure to enable will result in an error when calling `subscribeUserStatus` .

Server removes users from and dissolve the room.

1. Server removes users.

First, call the TRTC server user-removing API RemoveUser (for integer room IDs) or RemoveUserByStrRoomId (for string room IDs) to remove the target user from the TRTC room. The input example is as follows:

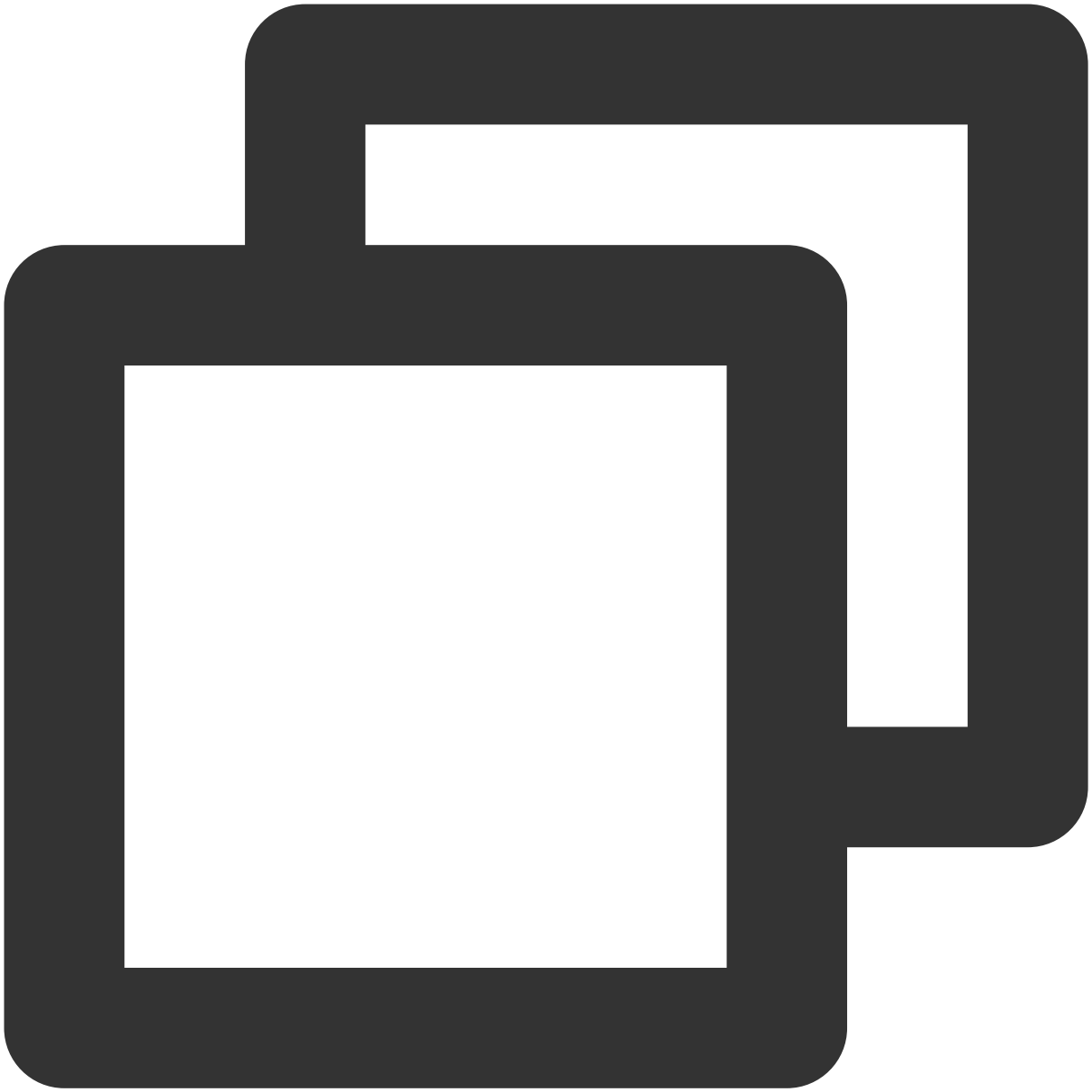
©2013-2022 Tencent Cloud. All rights reserved.



```
https://trtc.tencentcloudapi.com/?Action=RemoveUser
&SdkAppId=1400000001
&RoomId=1234
&UserIds.0=test1
&UserIds.1=test2
&<Common request parameters>
```

After removing the user successfully, the target user will receive the `onExitRoom()` callback on the client, with the `reason` value being 1. At this moment, you can handle leaving the seat and exiting the IM group in this callback.

©2013-2022 Tencent Cloud. All rights reserved.



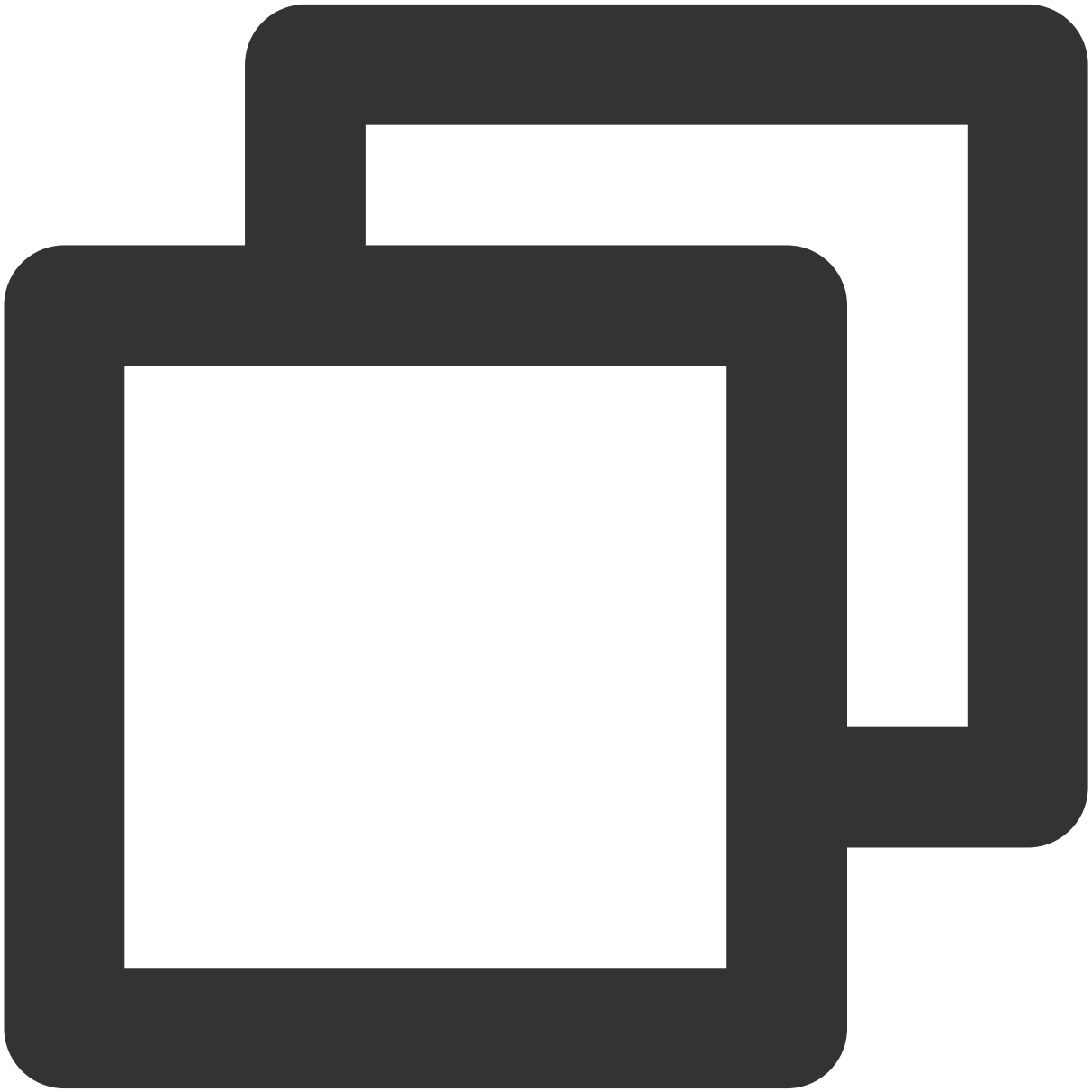
```
// Exit TRTC room event callback.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else {
        // reason 1: Removed from the current room by the server.
        // reason 2: The current room is dissolved.
        Log.d(TAG, "Removed from the room by the server, or the current room has be
        // Leave the seat.
        leaveSeat(seatIndex);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Exit IM group.
        quitGroup(groupID, new V2TIMCallback() {});
    }
}
```

2. Server dissolves the room.

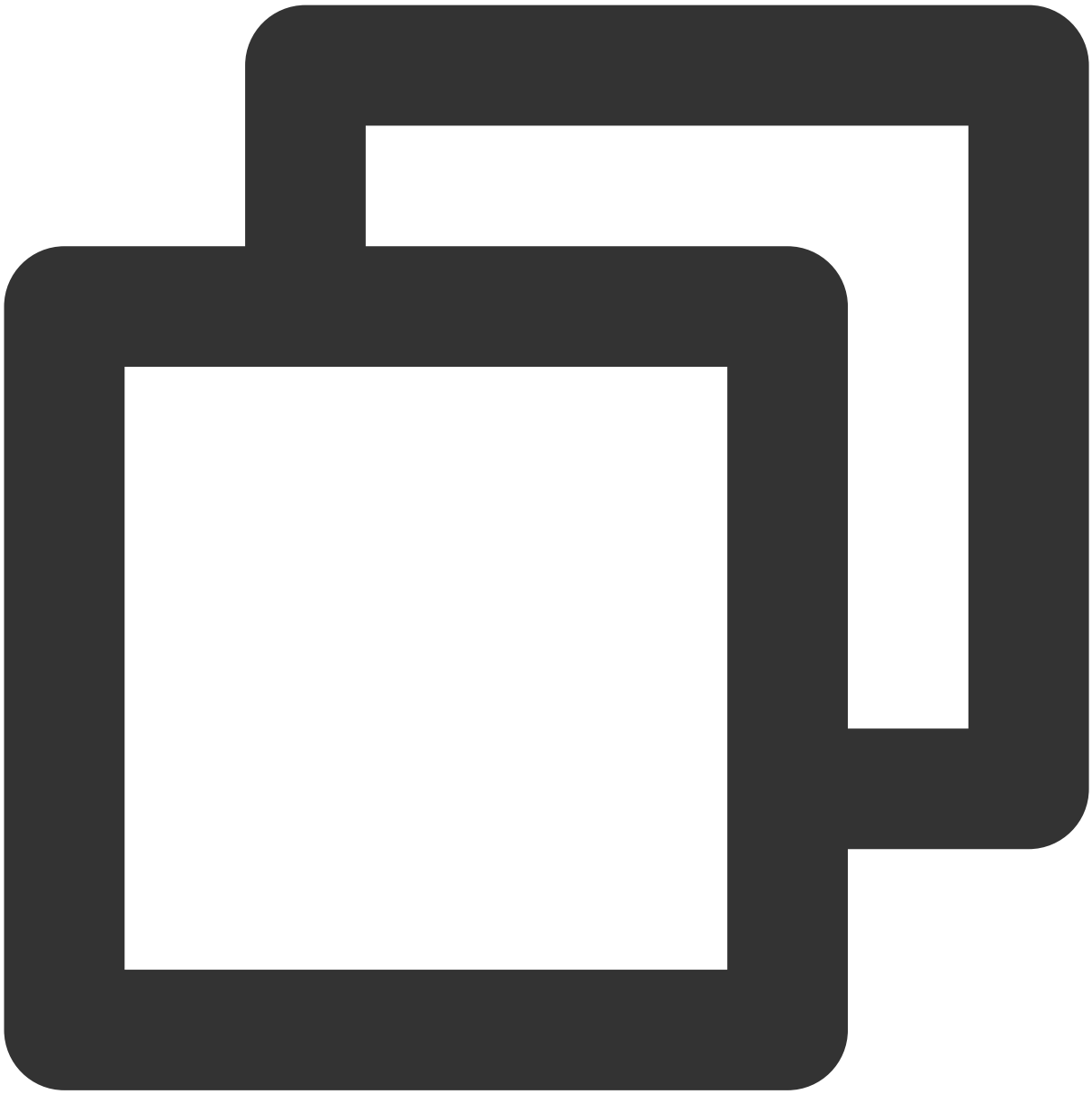
First, call the IM server group dissolvement API destroy_group to dissolve the target group. The example request URL is as follows:



```
https://xxxxxx/v4/group_open_http_svc/destroy_group?sdkappid=88888888&identifier=ad
```

©2013-2022 Tencent Cloud. All rights reserved.

After the group is dissolved, all members within the target group will receive the `onGroupDismissed()` callback on clients. At this point, you can handle operations such as exiting the TRTC room in this callback.



```
// Group dissolved callback.
V2TIMManager.getInstance().addGroupListener(new V2TIMGroupListener() {
  @Override
  public void onGroupDismissed(String groupID, V2TIMGroupMemberInfo opUser) {
      // Exit TRTC room.
      mTRTCCloud.stopLocalAudio();
      mTRTCCloud.exitRoom();
  }
```

©2013-2022 Tencent Cloud. All rights reserved.

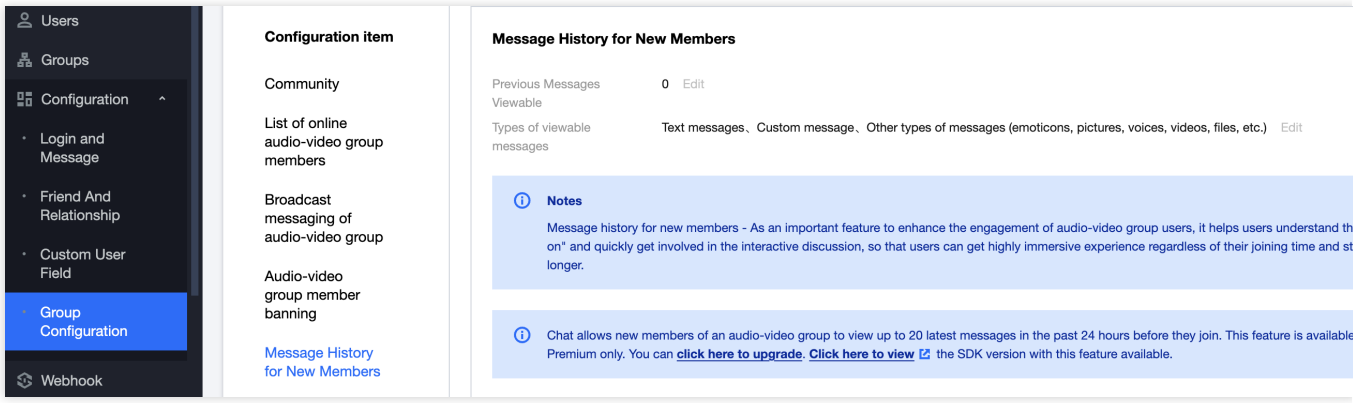
```
});
```

Note:

When all users in the room have completed exiting by calling `exitRoom()`, the TRTC room will be automatically dissolved. Of course, you can also mandatorily dissolve the TRTC room by calling the server API DismissRoom (for integer room IDs) or DismissRoomByStrRoomId (for string room IDs).

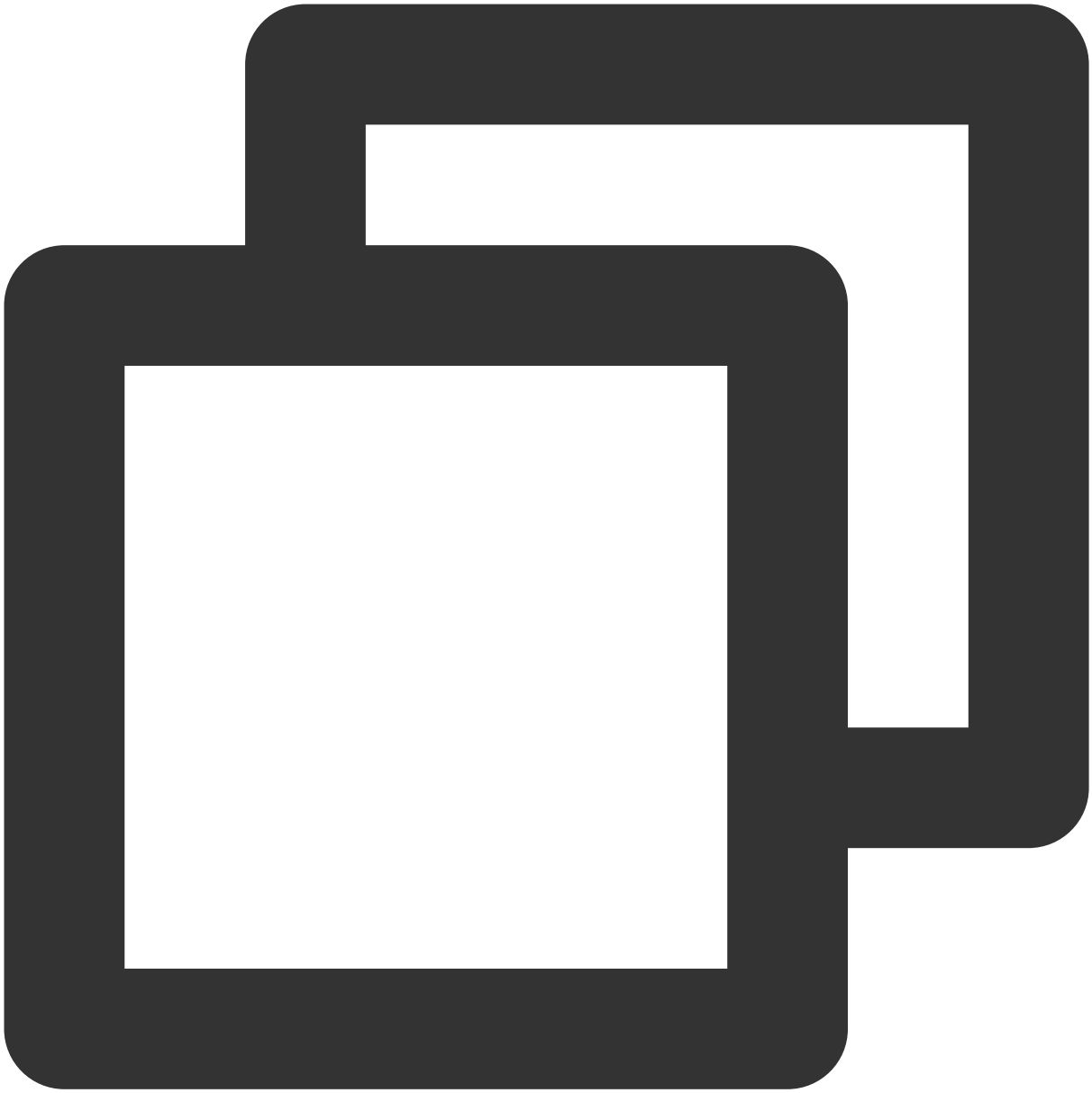
View live streaming room's historical messages upon room entry.

By default, using AVChatRoom does not store live streaming room's historical messages. Therefore, when new users enter the live streaming room, they can only see messages sent after their entry. To optimize the experience for new users joining the group, you can configure the number of messages new live streaming group users can pull before joining the group in the console, as shown in the figure:



Pulling historical messages before joining the live streaming group for live group users is the same as pulling historical messages for other groups, as shown in the sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMMessageListGetOption option = new V2TIMMessageListGetOption();
option.setGetType(V2TIMMessageListGetOption.V2TIM_GET_CLOUD_OLDER_MSG); // Pull ear
option.setGetTimeBegin(1640966400); // Starting from midnight January 1, 2022.
option.setGetTimePeriod(1 * 24 * 60 * 60); // Pull messages from a 24-hour period.
option.setCount(Integer.MAX_VALUE); // Return all messages within the time range.
option.setGroupID(#your group id#); // Pull messages for the group chat.
V2TIMManager.getMessageManager().getHistoryMessageList(option, new V2TIMValueCallba
    @Override
    public void onSuccess(List<V2TIMMessage> v2TIMMessages) {
        Log.i("imsdk", "success");
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onError(int code, String desc) {
        Log.i("imsdk", "failure, code:" + code + ", desc:" + desc);
    }
});
```

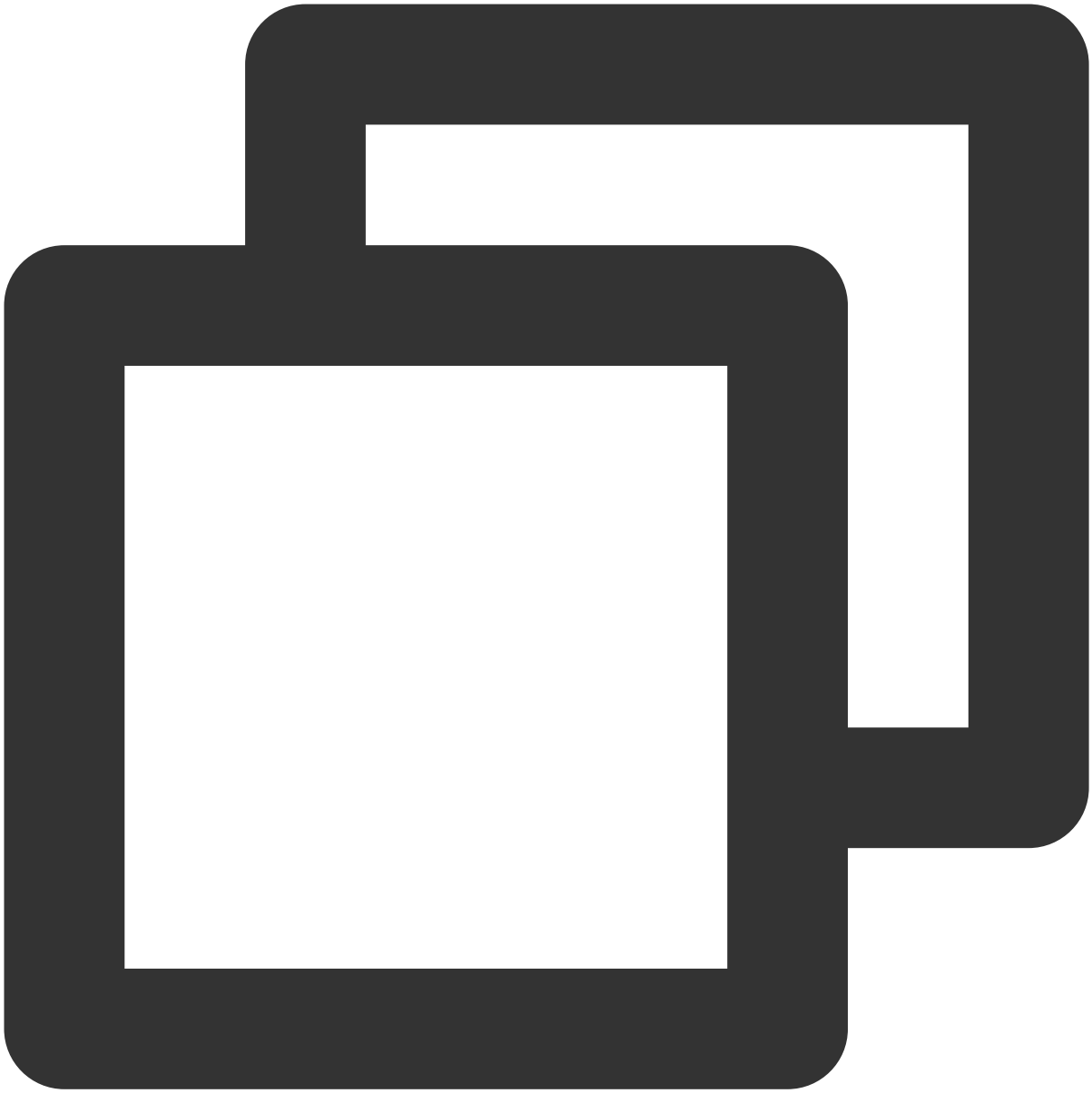
Note:

This feature is only available to advanced users. It only supports pulling up to 20 historical messages within 24 hours from the group.

Enter the room to sense the mute status of on-mic anchors.

Solution 1: Default all anchors to mute status upon room entry, then unmute the corresponding anchors based on the `onUserAudioAvailable(userId, true)` callback.

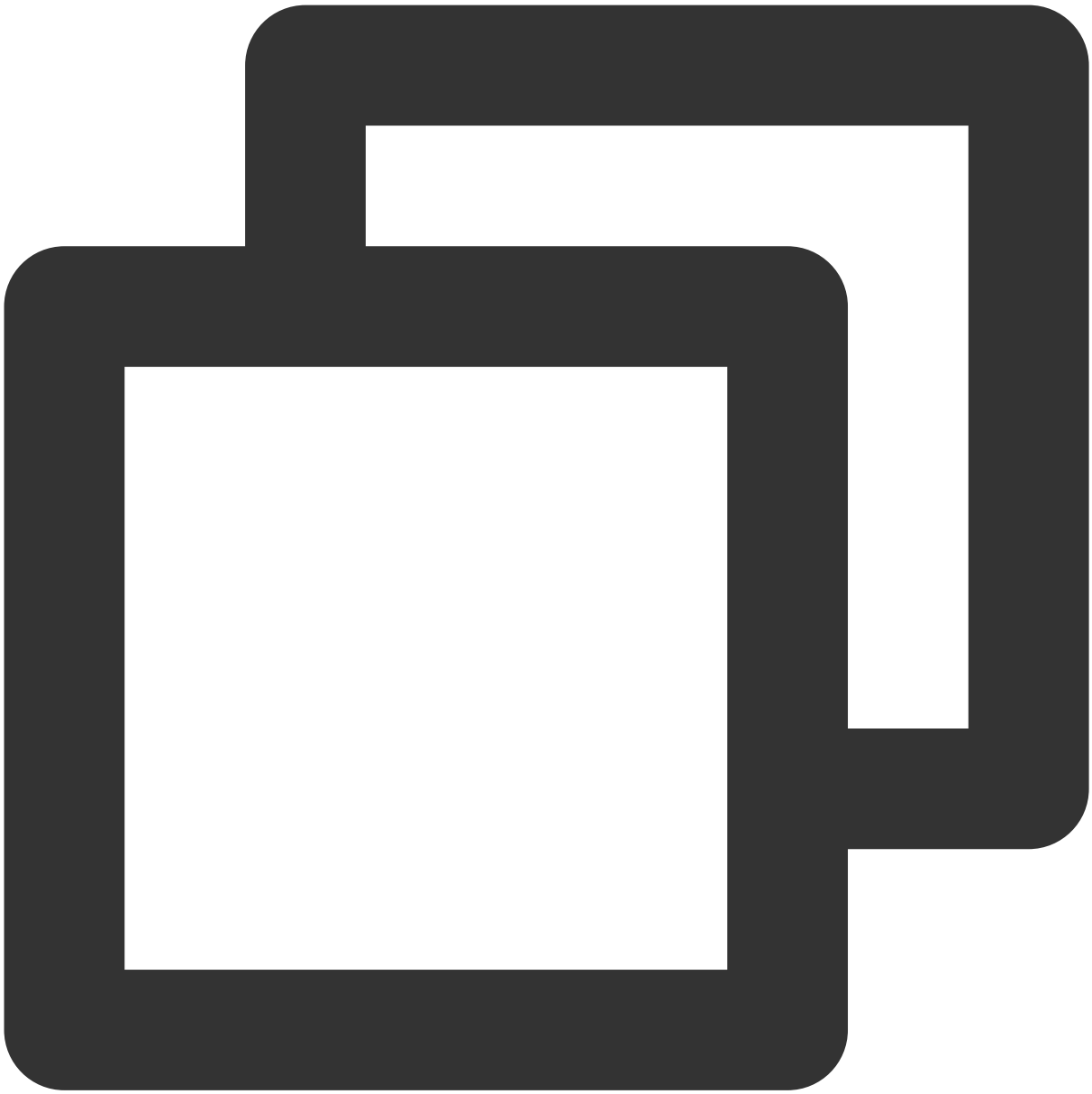
©2013-2022 Tencent Cloud. All rights reserved.



```
private class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onUserAudioAvailable(String userId, boolean available) {
        if (available) {
            // Unmute the corresponding anchors.
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Solution 2: Store the mute status of the anchors in the IM group attributes. Audiences entering the room obtain all group attributes to parse the mute status of on-mic anchors.



```
V2TIMManager.getGroupManager().getGroupAttributes(groupID, null, new V2TIMValueCall
    @Override
    public void onError(int i, String s) {
        // Failed to obtain the group attributes.
    }

    @Override
    public void onSuccess(Map<String, String> attrMap) {
```

©2013-2022 Tencent Cloud. All rights reserved.

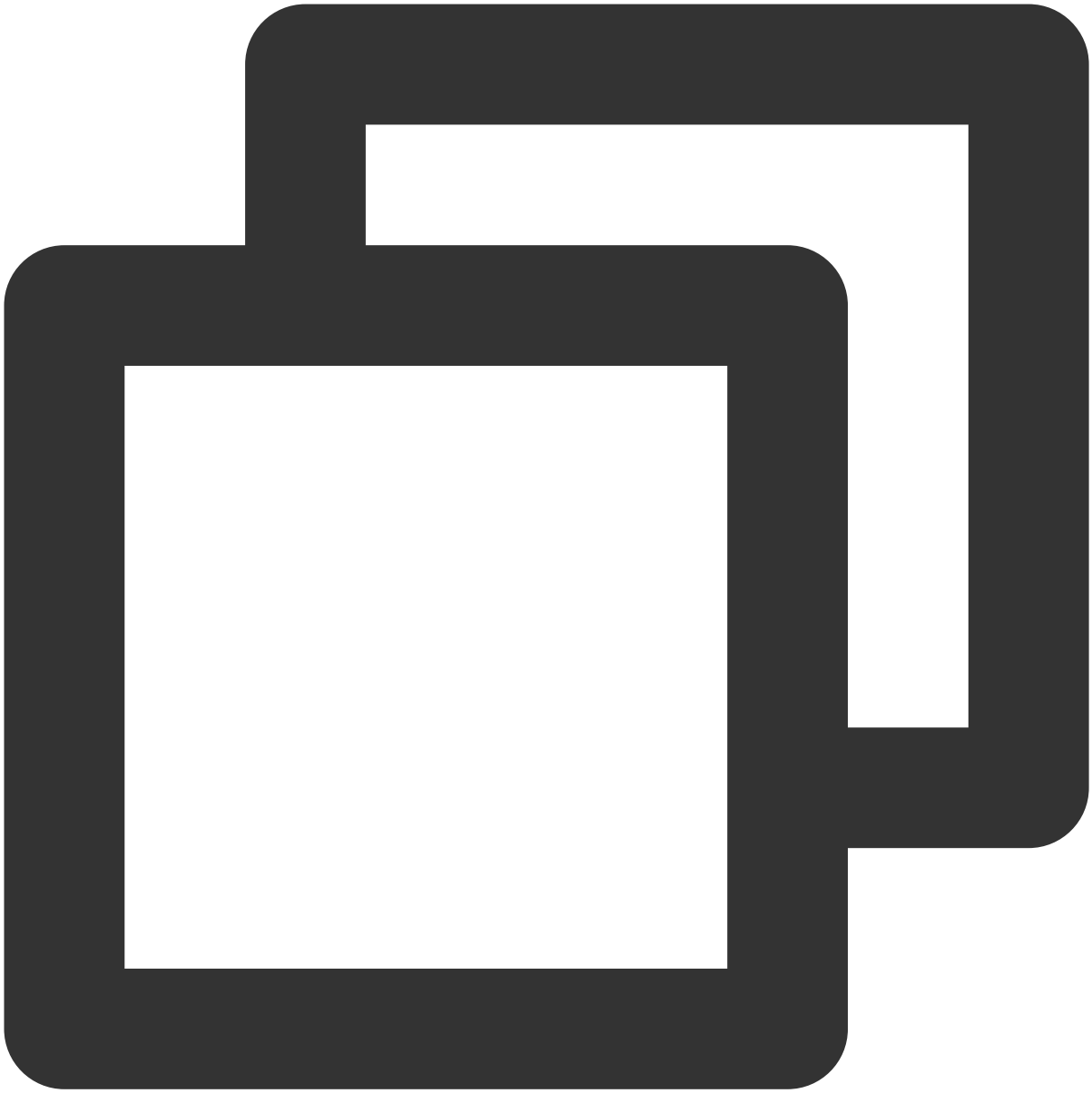
```
            // Successfully obtained group attributes. It is assumed that the key used
            String muteStatus = attrMap.get("muteStatus");
            // Parse muteStatus, and obtain the mute status of each on-mic anchor.
        }
    });
```

Issues with audio input and output of Bluetooth headphones.

The mobile phone has successfully connected to the Bluetooth headphones, but the audio input or output of the TRTC application still uses the mobile phone's microphone or speaker.

1. If the audio output is working fine with the Bluetooth headphones, but only the audio input is still using the mobile phone's microphone, check the settings for the audio volume type. Only under the call volume mode does it support capturing audio through the microphone on the Bluetooth headphones. For details, see Audio Management - Audio Quality and Volume Type.

©2013-2022 Tencent Cloud. All rights reserved.

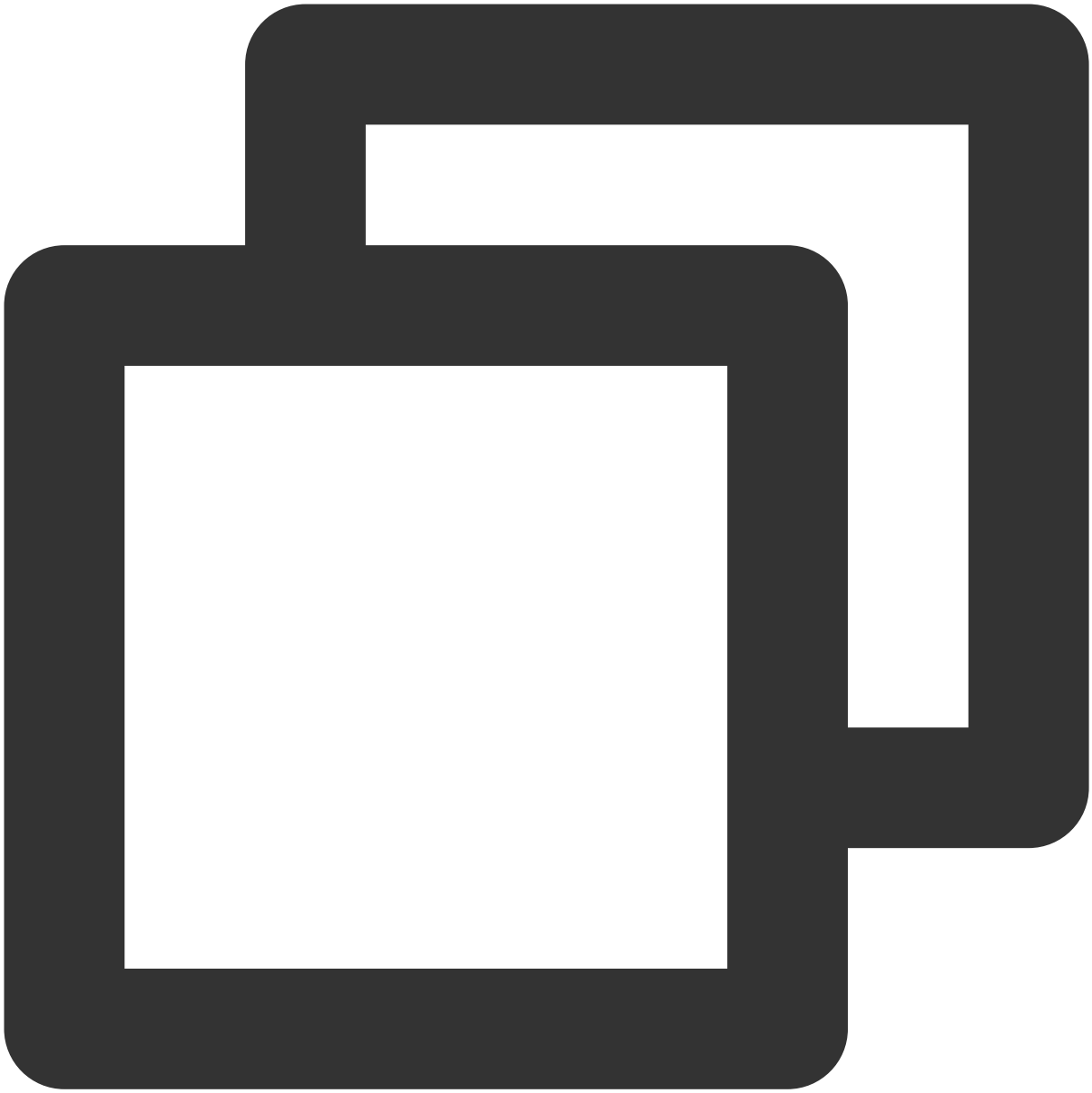


```
mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeVOIP);
```

2. If both audio input and output fail to use the Bluetooth headphones, check if the app has been granted Bluetooth permissions. Devices running on systems below Android 12 require at least the `BLUETOOTH` permission. Devices running on Android 12 and above require at least the `BLUETOOTH_CONNECT` permission and also the permissions to be requested dynamically in the code.

To configure Bluetooth permissions in the AndroidManifest.xml for compatibility with systems below Android 12, it is recommended to declare permissions as follows:

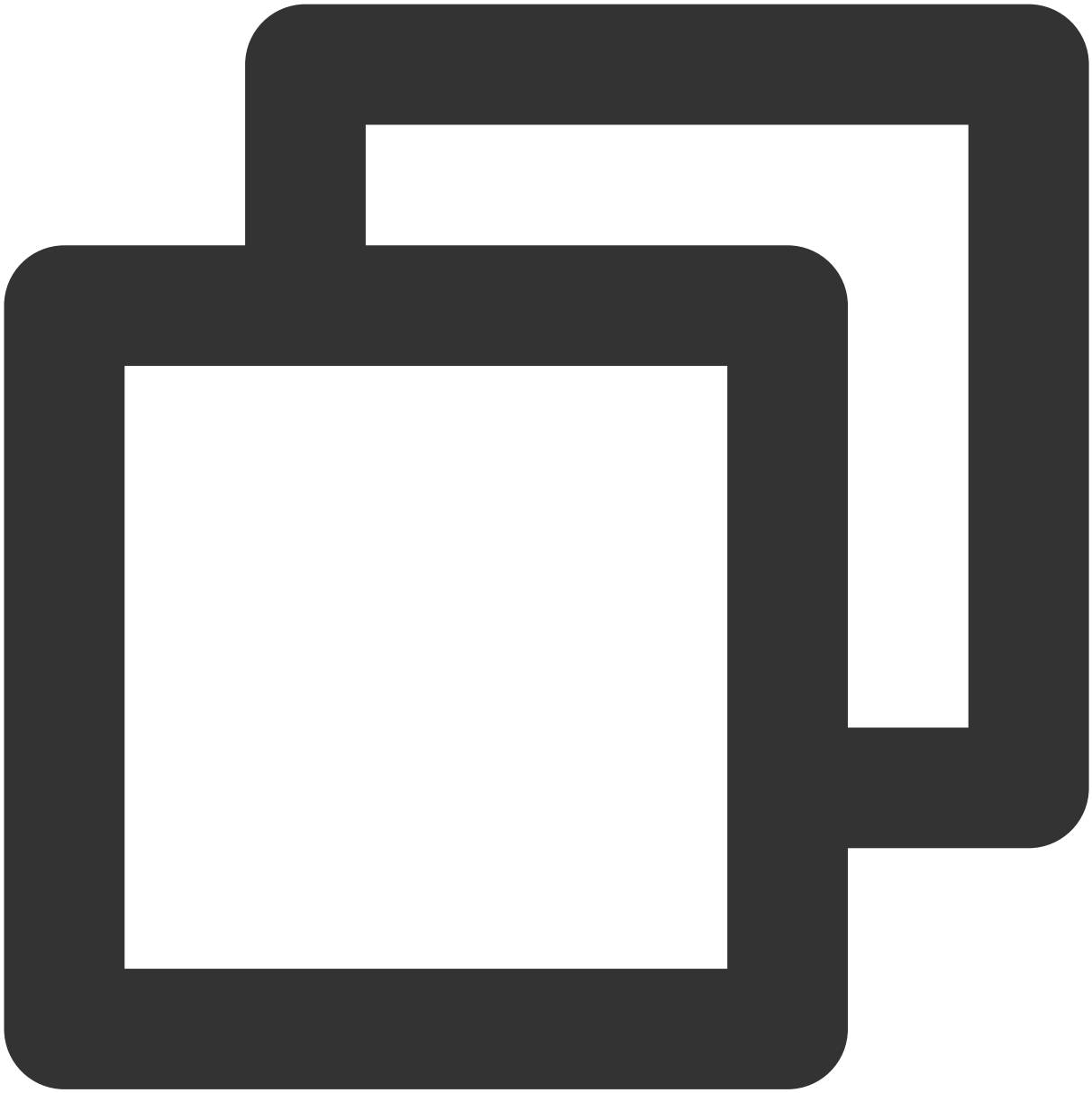
©2013-2022 Tencent Cloud. All rights reserved.



```
<!--Normal Permission: basic Bluetooth connection permissions-->
<uses-permission android:name="android.permission.BLUETOOTH" android:maxSdkVersion=
<!--Normal Permission: Bluetooth management and scan permissions-->
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" android:maxSdkVe

<!--Runtime Permission: Android 12 Bluetooth permissions for discovering Bluetooth
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<!--Runtime Permission: Android 12 Bluetooth permissions for making the current dev
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<!--Runtime Permission: Android 12 Bluetooth permissions for communicating with pai
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
```

©2013-2022 Tencent Cloud. All rights reserved.

For Android 12 and above systems, the method to dynamically request the newly added fine-grained Bluetooth permissions, is as follows:



```
private List<String> permissionList = new ArrayList<>();

protected void initPermission() {
    // Check if the Android SDK version is Android 12 and later.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        // Add the required permissions to be requested dynamically according to th
        permissionList.add(Manifest.permission.BLUETOOTH_SCAN);
```

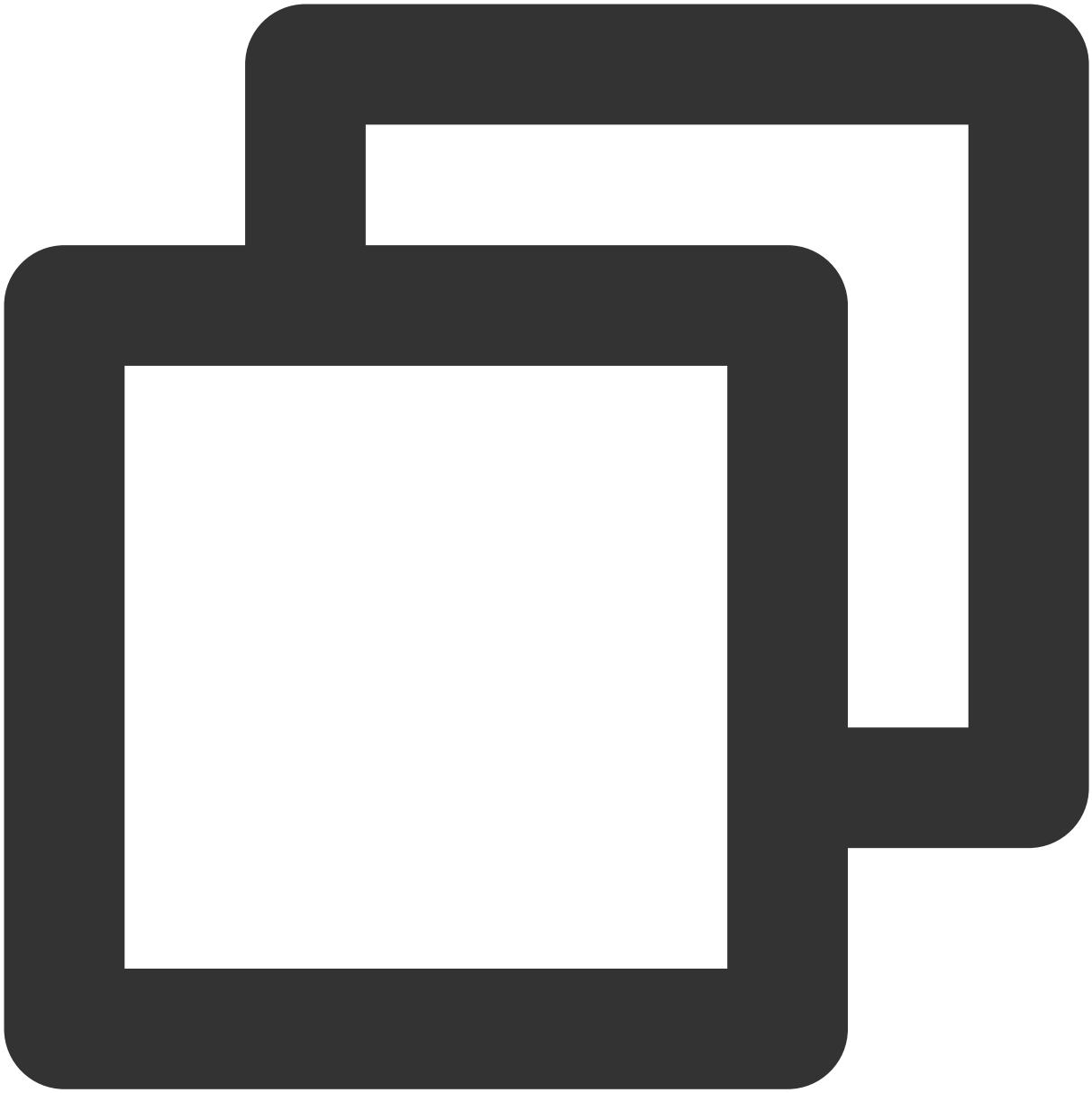
©2013-2022 Tencent Cloud. All rights reserved.

```
            permissionList.add(Manifest.permission.BLUETOOTH_ADVERTISE);
            permissionList.add(Manifest.permission.BLUETOOTH_CONNECT);
        }
        if (permissionList.size() != 0) {
            // Dynamically request permissions.
            ActivityCompat.requestPermissions(this, permissionList.toArray(new String[0
        }
    }
```

Issues with supported resource paths for playing music.

When using the TRTC SDK API `startPlayMusic` to play background music, the music resource path parameter `path` does not support for file paths from directories such as assets or raw used in Android development for storing application resources. This is because files in these directories are bundled into the APK and are not extracted to the device's file system after installation. Currently, only absolute paths to network resources URLs, external storage on Android devices, and files in the application's private directory are supported.
You can work around this issue by copying resource files from the assets directory to either the device external storage or the application private directory beforehand. Sample code is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
public static void copyAssetsToFile(Context context, String name) {
    // The files directory under the application's directory.
    String savePath = ContextCompat.getExternalFilesDirs(context, null)[0].getAbsol
    // The cache directory under the application's directory.
    // String savePath = getApplication().getExternalCacheDir().getAbsolutePath();
    // The files directory under the application's private storage directory.
    // String savePath = getApplication().getFilesDir().getAbsolutePath();
    String filename = savePath + "/" + name;
    File dir = new File(savePath);
    // Create the directory if it does not exist.
    if (!dir.exists()) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            dir.mkdir();
        }
        try {
            if (!(new File(filename)).exists()) {
                InputStream is = context.getResources().getAssets().open(name);
                FileOutputStream fos = new FileOutputStream(filename);
                byte[] buffer = new byte[1024];
                int count = 0;
                while ((count = is.read(buffer)) > 0) {
                    fos.write(buffer, 0, count);
                }
                fos.close();
                is.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Application private storage files directory path: `/data/user/0/<package_name>/files/<file_name>` .

Application external storage files directory path: `/storage/emulated/0/Android/data/<package_name>/files/<file_name>` .

Application external storage cache directory path: `/storage/emulated/0/Android/data/<package_name>/cache/<file_name>` .

Note:

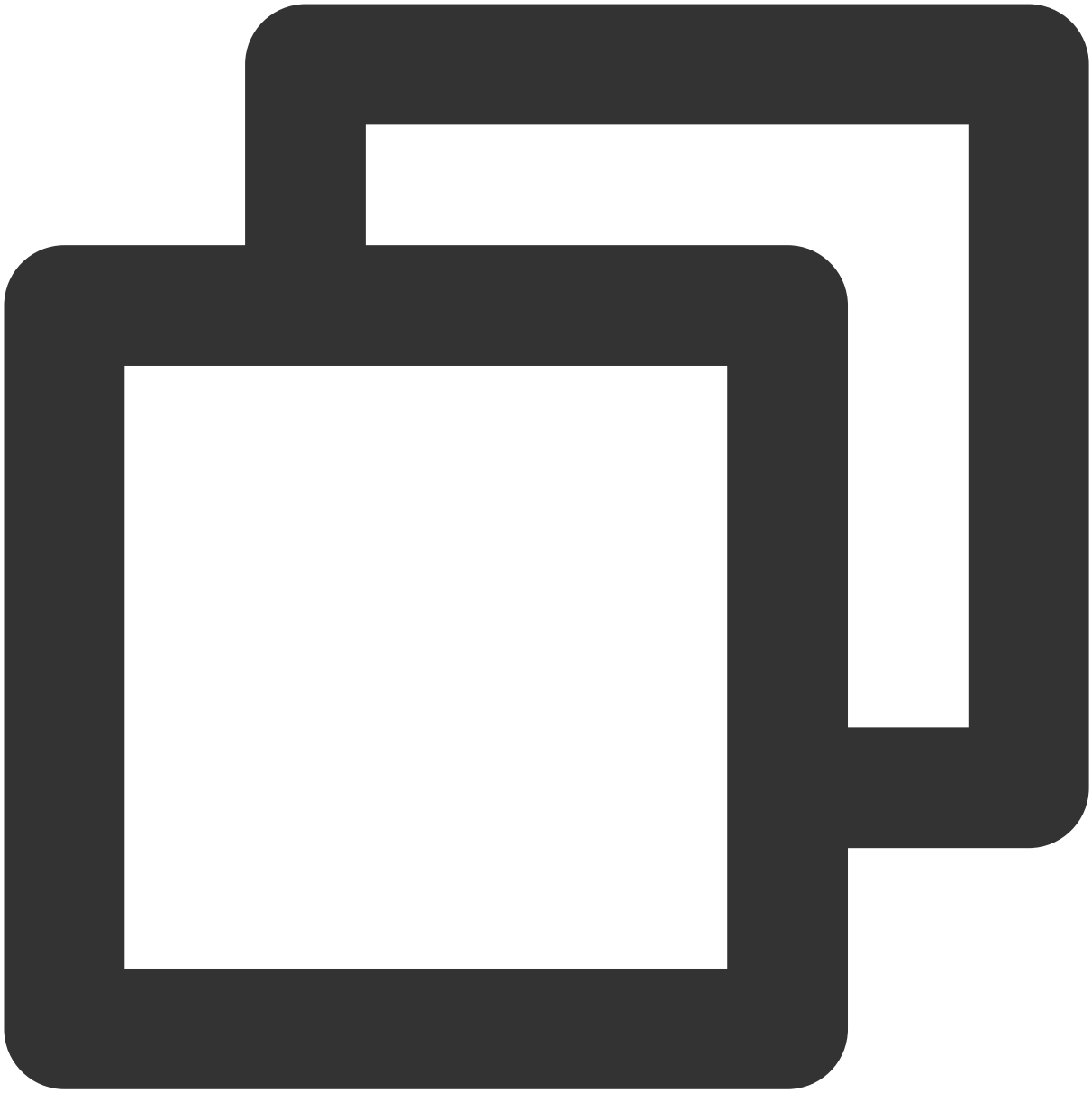
If you provide a path that is an external storage path outside of the application's specific directories, on Android 10 and above devices, you may face denial of access to the resource. This is due to Google introducing Partition Storage, a new storage management system. You can temporarily bypass this by adding the following code inside the <application> tag in the AndroidManifest.xml file: `android:requestLegacyExternalStorage="true"` . This attribute only takes effect on applications with targetSdkVersion 29 (Android 10), and applications with a higher version targetSdkVersion are still recommended to use the application's private or external storage paths.

For TRTC SDK v11.5 and later, playback of local music resources on Android devices via Content URI from Content Provider components is supported.

On Android 11 and HarmonyOS 3.0 or later, if you cannot access resource files in the external storage directory, you need to request the `MANAGE_EXTERNAL_STORAGE` permission:

First, you need to add the following entry in your application's AndroidManifest file.

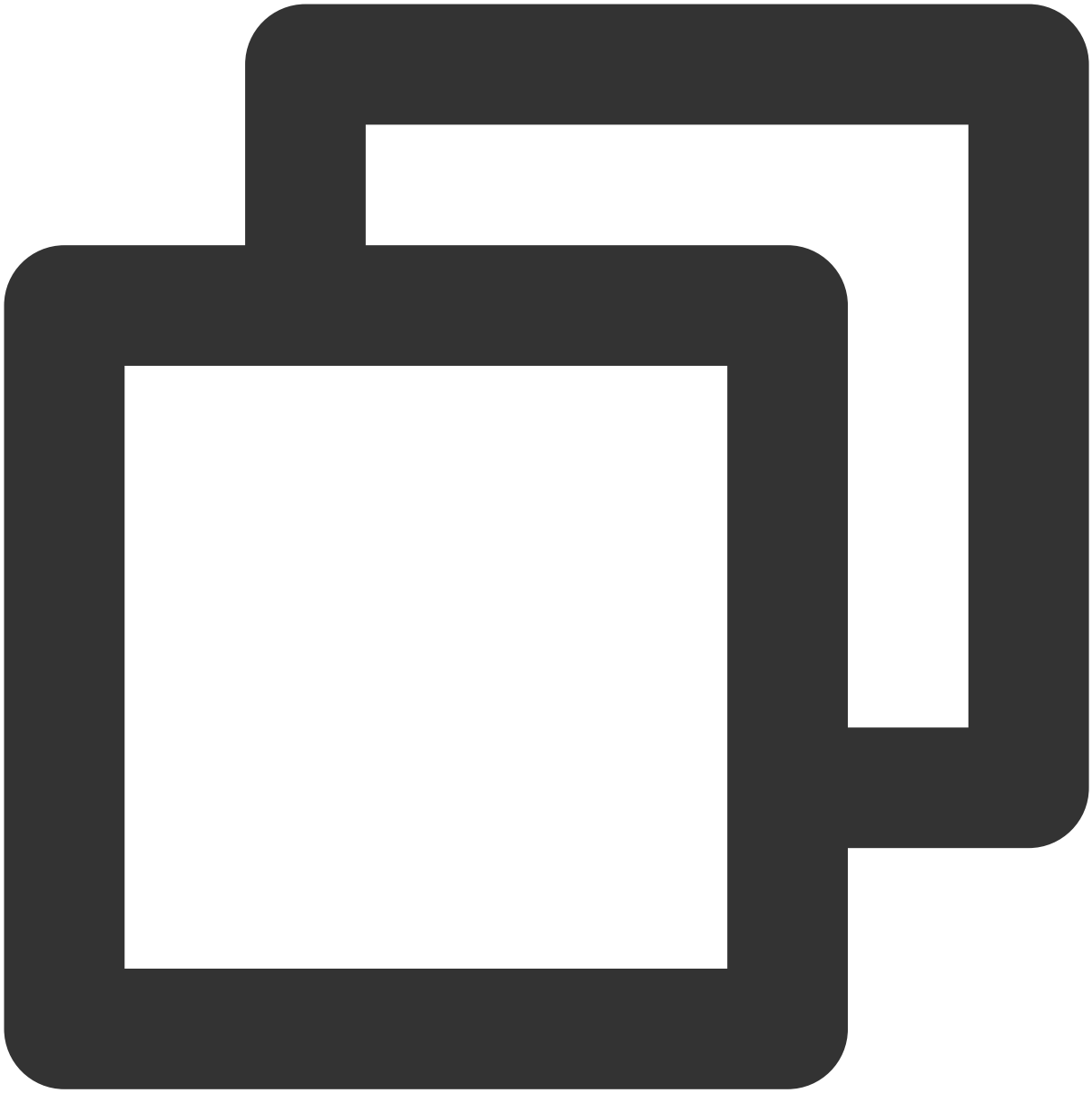
©2013-2022 Tencent Cloud. All rights reserved.



```
<manifest ...>
    <!-- This is the permission itself -->
    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />

    <application ...>
        ...
    </application>
</manifest>
```

Then, guide users to manually grant this permission at the point in your application where it is needed.

©2013-2022 Tencent Cloud. All rights reserved.



```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
    if (!Environment.isExternalStorageManager()) {
        Intent intent = new Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERM
        Uri uri = Uri.fromParts("package", getPackageName(), null);
        intent.setData(uri);
        startActivity(intent);
    }
} else {
    // For Android versions less than Android 11, you can use the old permissions m
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.WRITE_
}
```

©2013-2022 Tencent Cloud. All rights reserved.

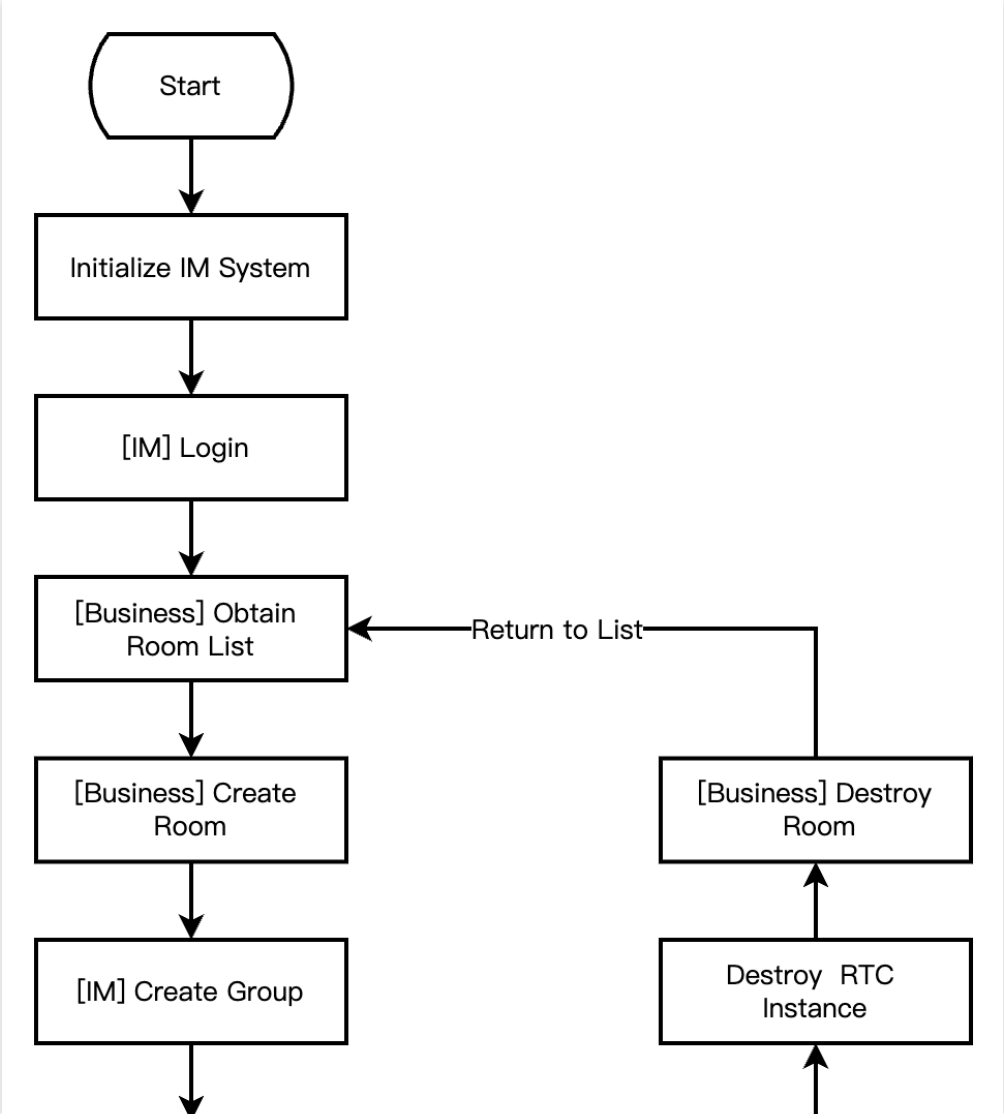
©2013-2022 Tencent Cloud. All rights reserved.

iOS

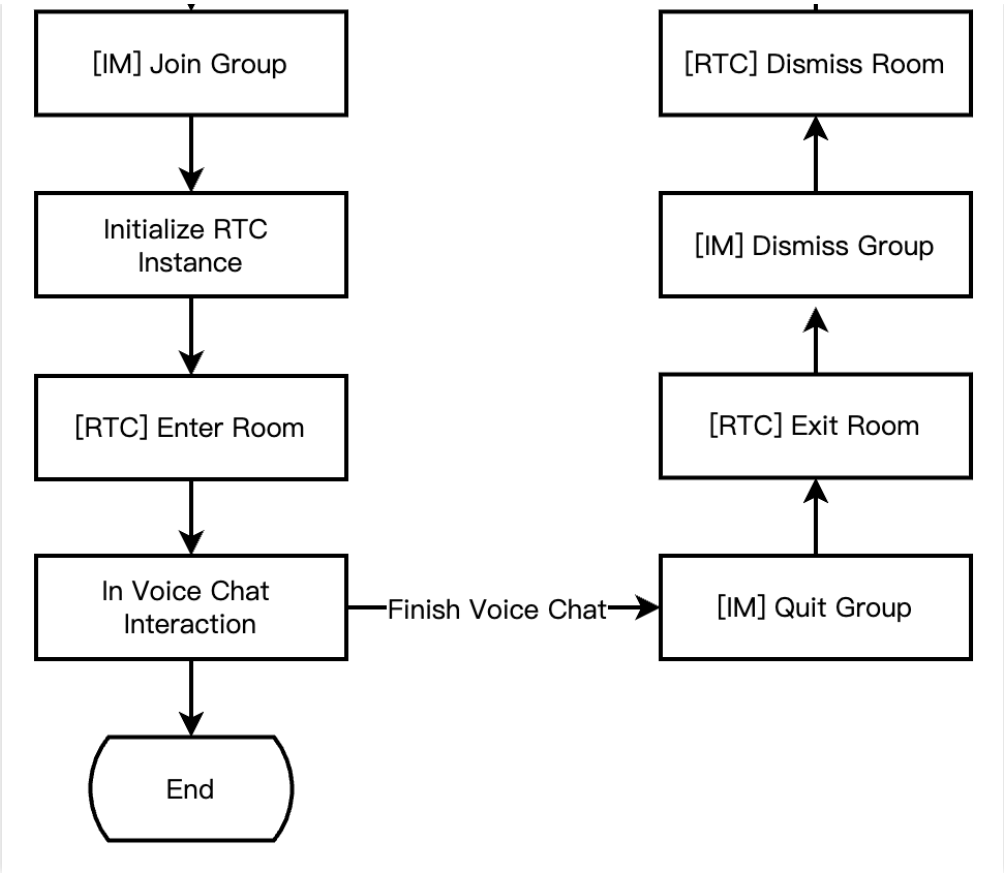
Last updated：2024-07-18 14:26:14

Business Process

This document summarizes some common business processes in voice chat rooms to help you better understand the entire scenario implementation process.
Room management process.
Room owner seat management process.
Audience seat management process.
The following figure shows the room management process, including the creation, joining, exiting, and dissolvement of rooms.

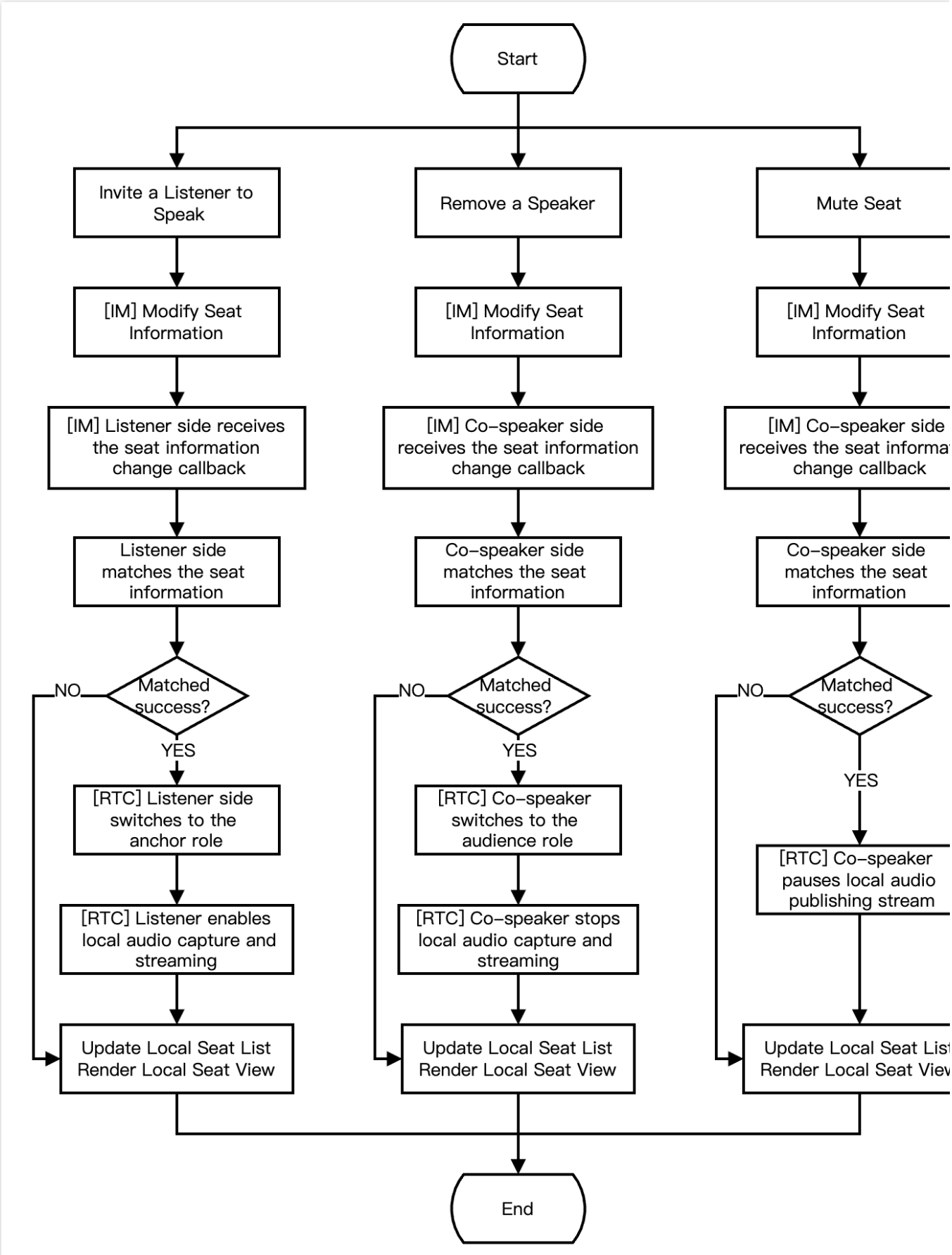


©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the room owner seat management process, including inviting a listener to speak, removing a speaker, and muting a seat.

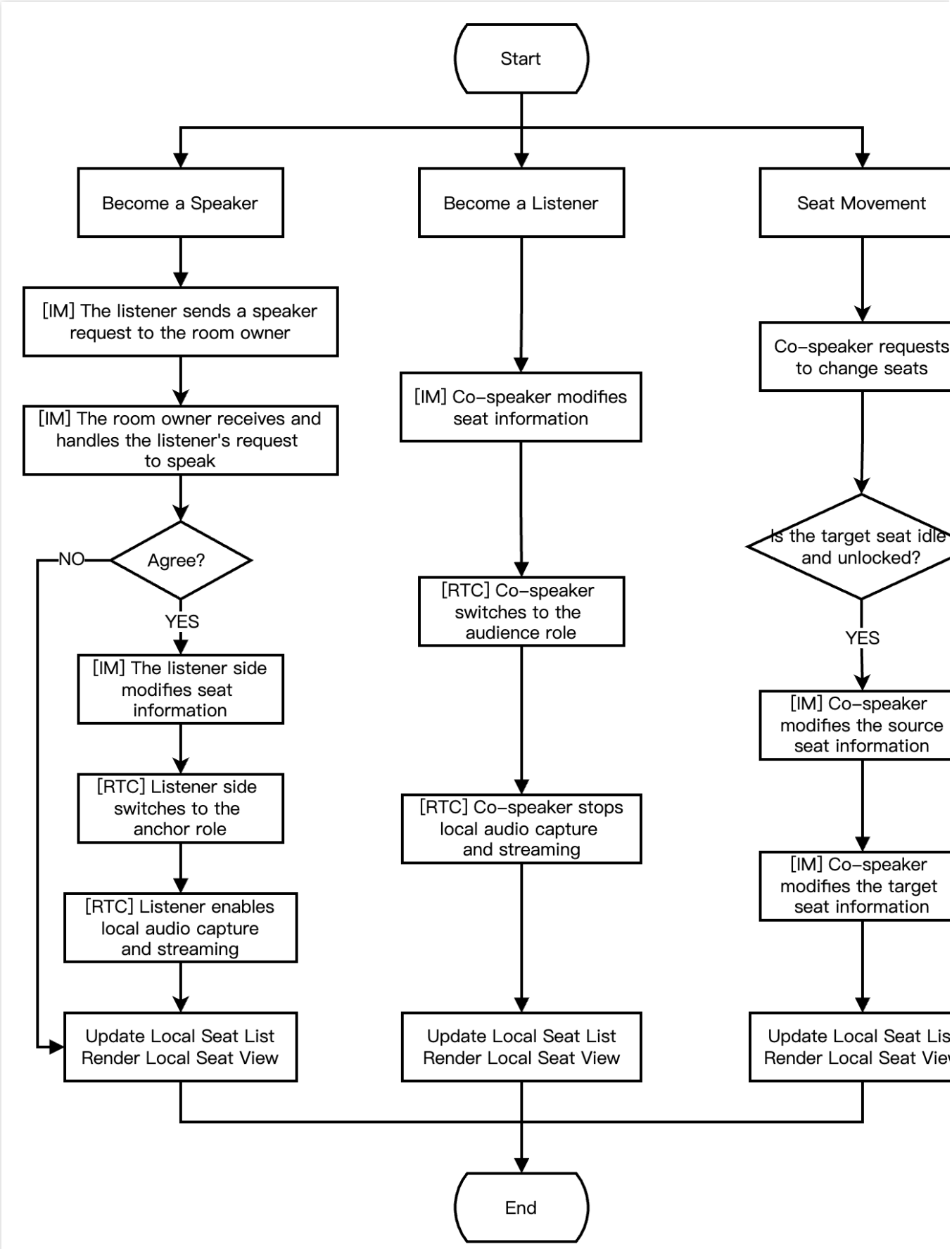
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

The following figure shows the audience seat management process, including becoming a speaker, become a listener, and moving a seat.

©2013-2022 Tencent Cloud. All rights reserved.



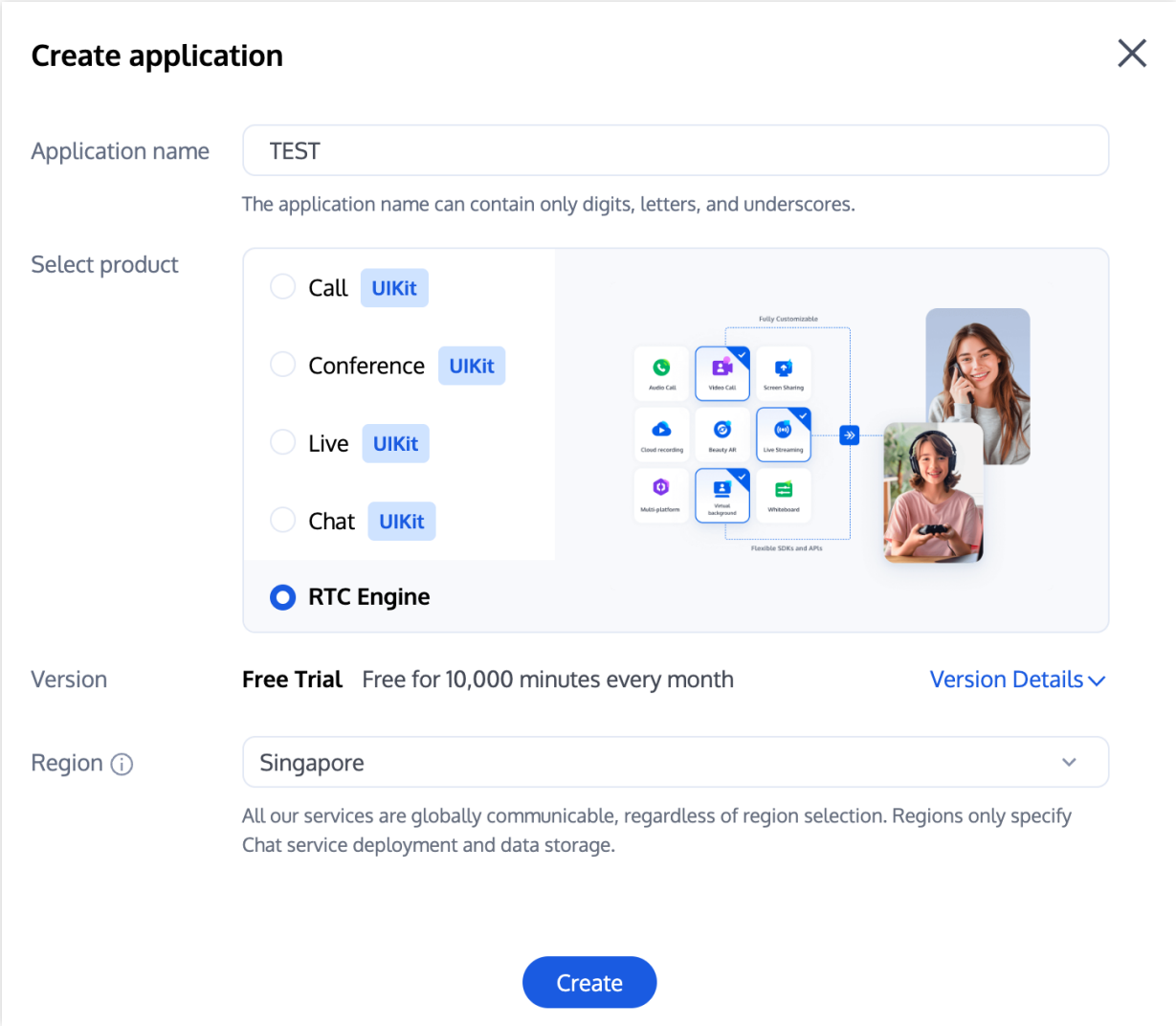
©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparation

Step 1. Activating the service.

Voice chat room scenarios usually require dependencies on two paid PaaS services from Tencent Cloud, Instant Messaging (IM) and Tencent Real-Time Communication (TRTC) for construction.

1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. At this time, in the Instant Messaging (IM) console, an IM experience edition application with the same SDKAppID as the current TRTC application will be automatically created. The account and authentication system of the two can be reused. Subsequently, you can choose to upgrade the TRTC or IM application version as needed. For example, advanced versions can unlock more value-added feature services.



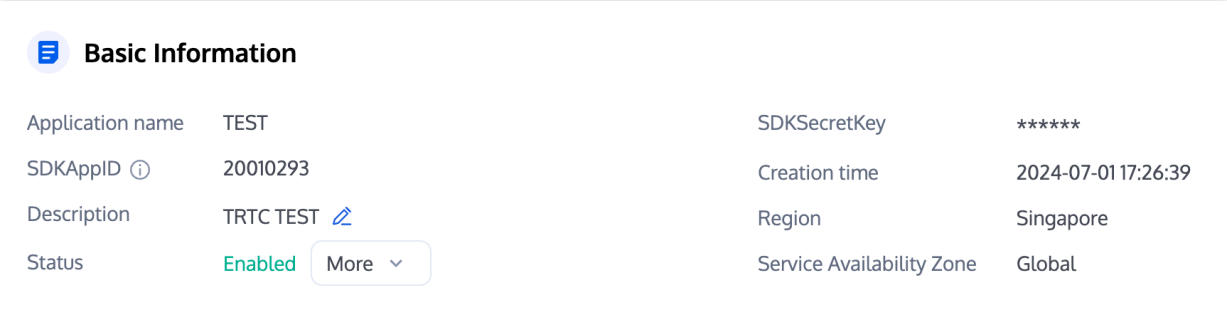
Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

©2013-2022 Tencent Cloud. All rights reserved.

TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

2. After creating the application, you can see its basic information in the Application Management - Application Overview section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.



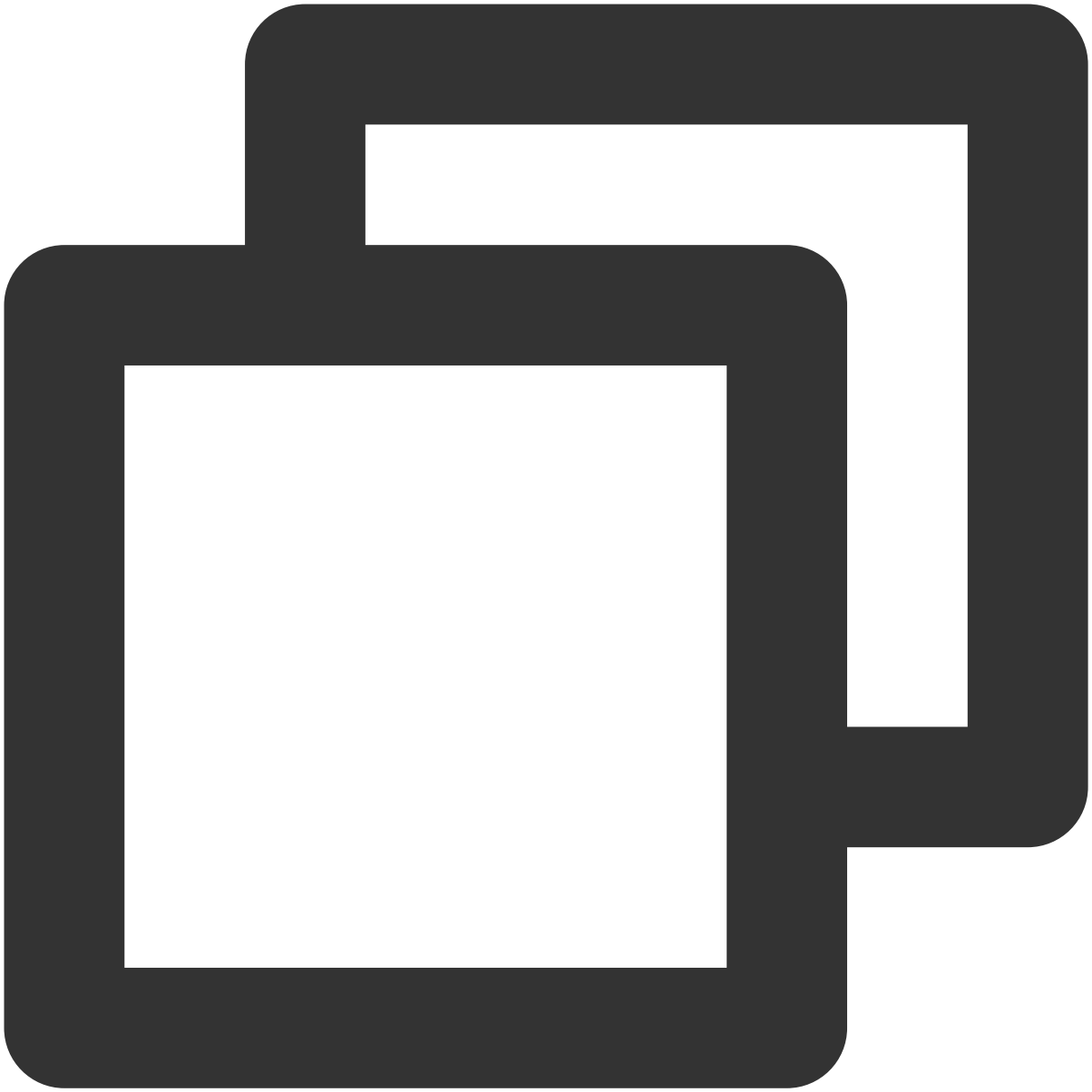
Step 2: Importing SDK.

The TRTC SDK and IM SDK are now available on CocoaPods. It is recommended to integrate the SDK through CocoaPods.

1. Install CocoaPods.

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

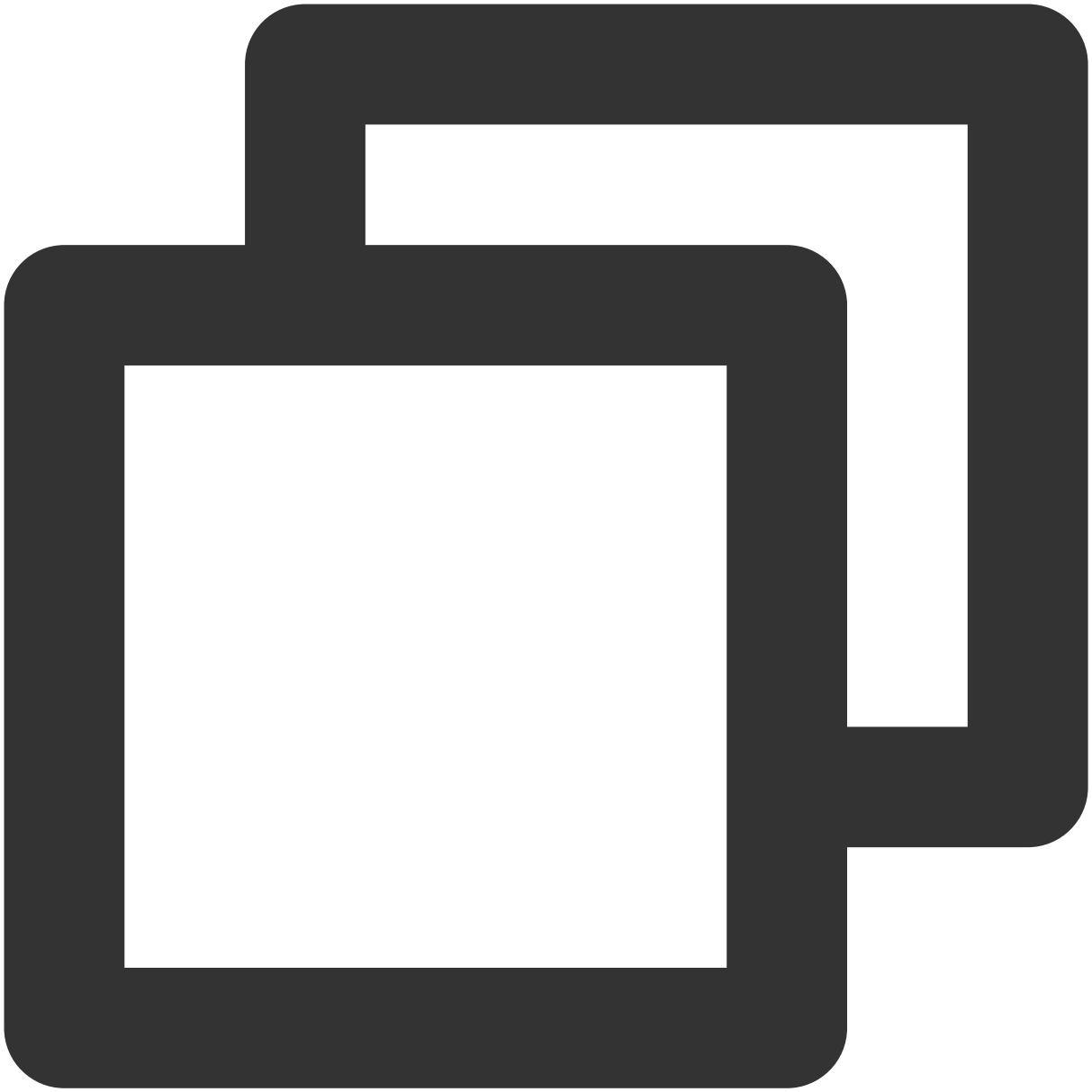


```
sudo gem install cocoapods
```

2. Create a Podfile.

Go to the project directory, and enter the following command. A Podfile file will then be created in the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

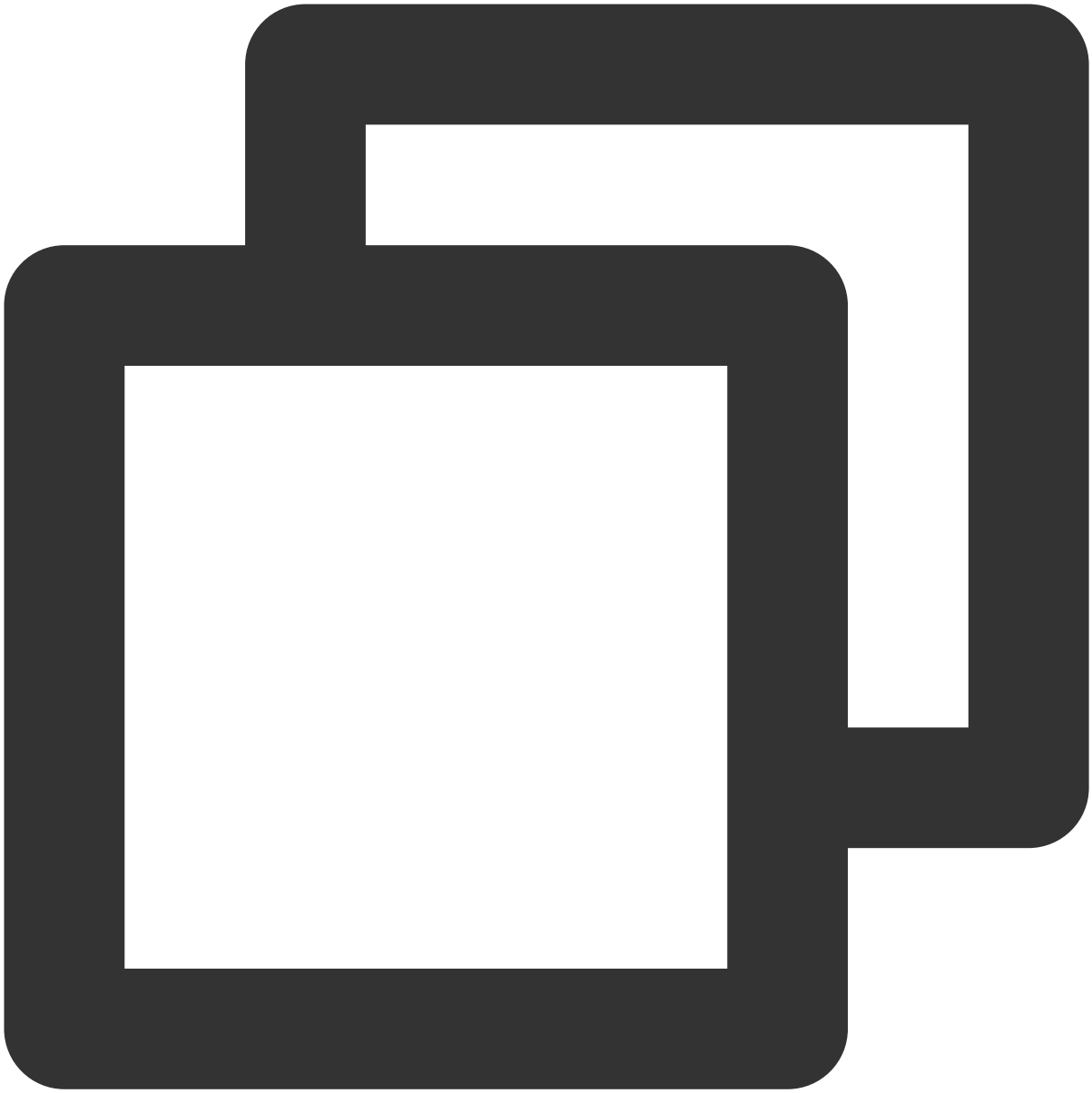


```
pod init
```

3. Edit the Podfile.

Choose the appropriate version for your project and edit the Podfile.

©2013-2022 Tencent Cloud. All rights reserved.



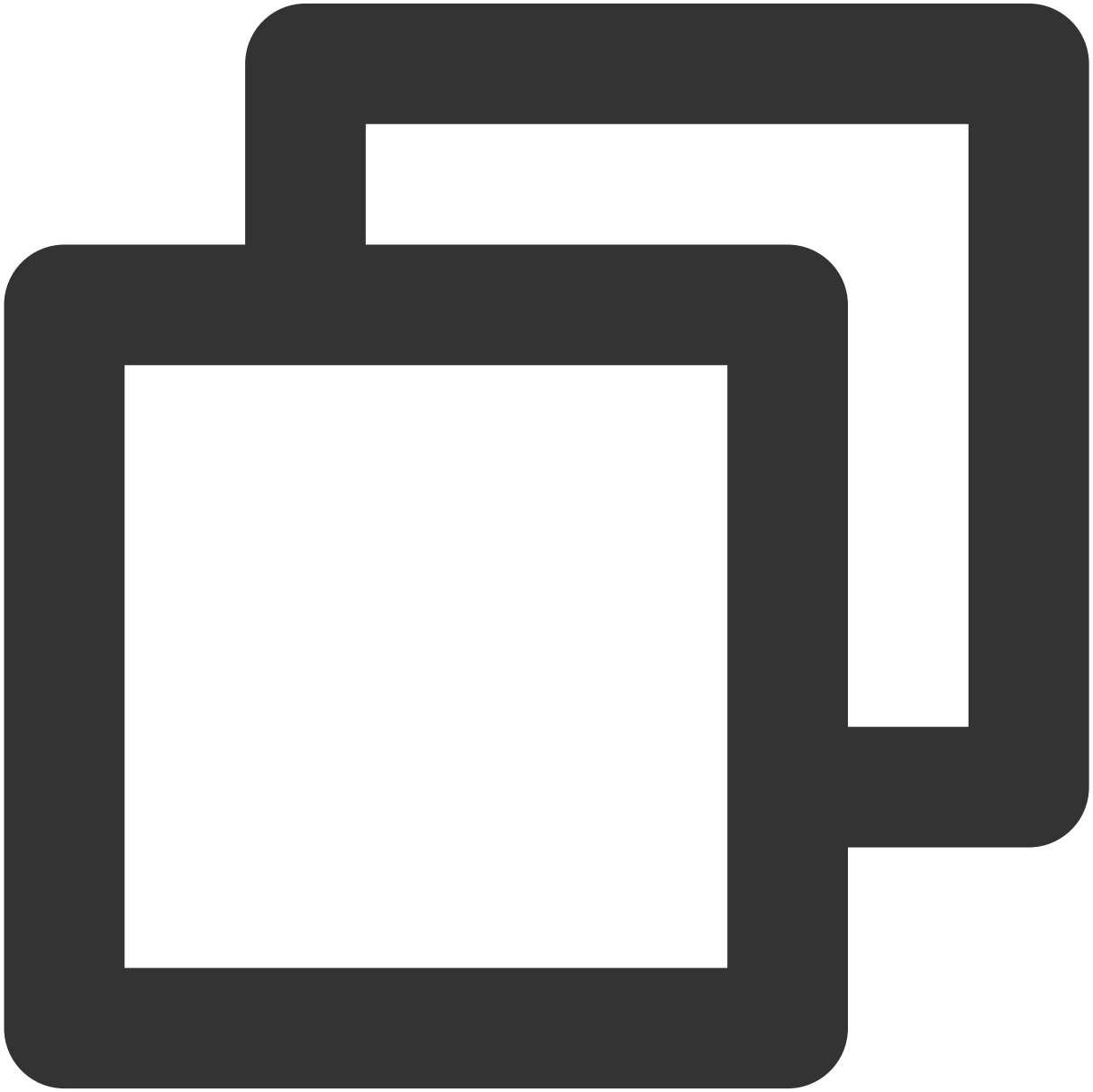
```
platform :ios, '8.0'
target 'App' do

    # TRTC Lite Edition
    # The installation package has the minimum incremental size. It only supports t
    pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd

    # Add the IM SDK
    pod 'TXIMSDK_Plus_iOS'
    # pod 'TXIMSDK_Plus_iOS_XCFramework'
    # pod 'TXIMSDK_Plus_Swift_iOS_XCFramework'
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    # If you need to add the Quic plugin, please uncomment the next line.
    # Note: This plugin must be used with the Objective-C edition or XCFramework ed
    # pod 'TXIMSDK_Plus_QuicPlugin'

end
```

4. Update and install the SDK.

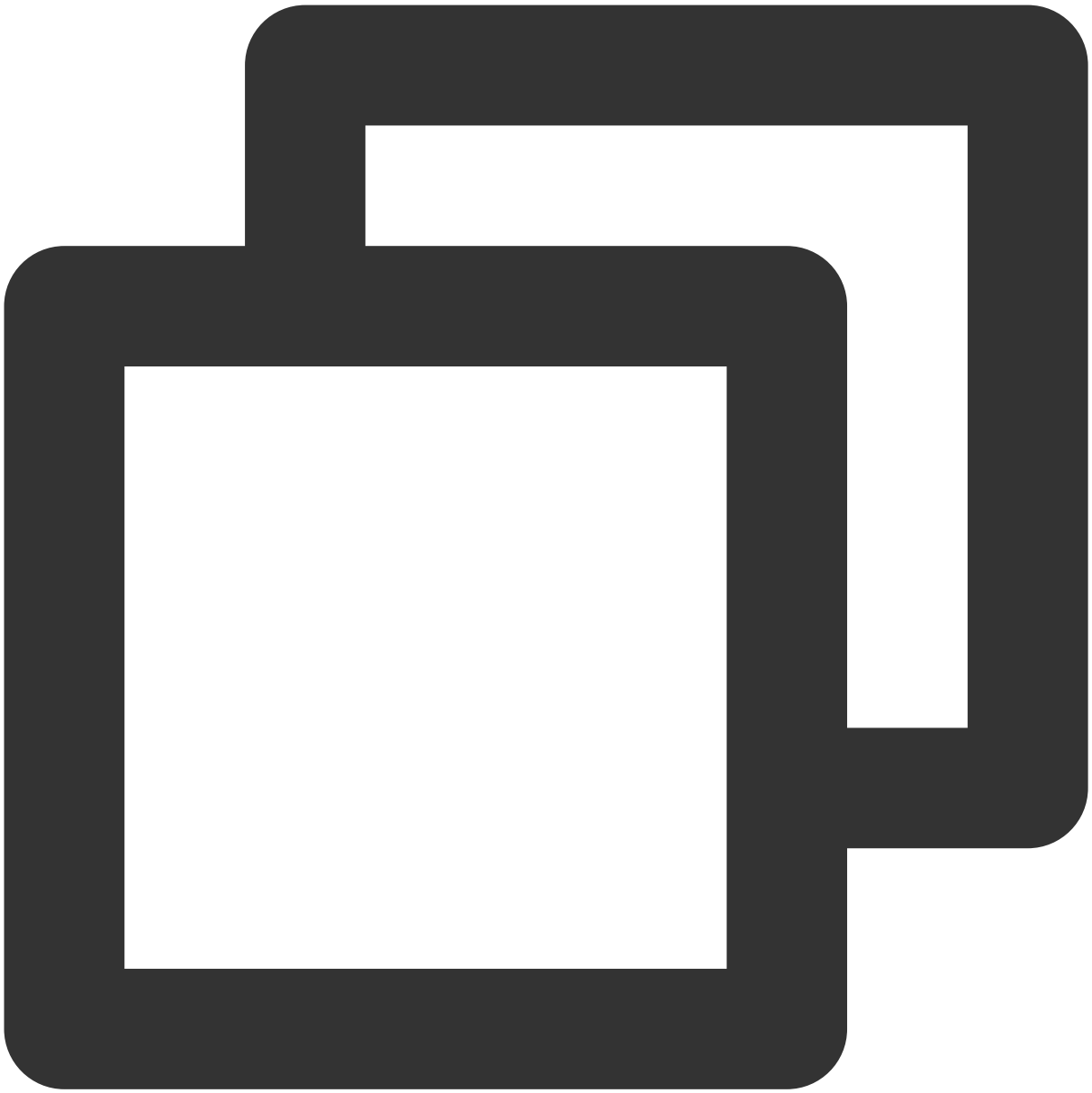
Enter the following command in a terminal window to update the local repository files and install the SDK.



```
pod install
```

©2013-2022 Tencent Cloud. All rights reserved.

Or use the following command to update the local repository.



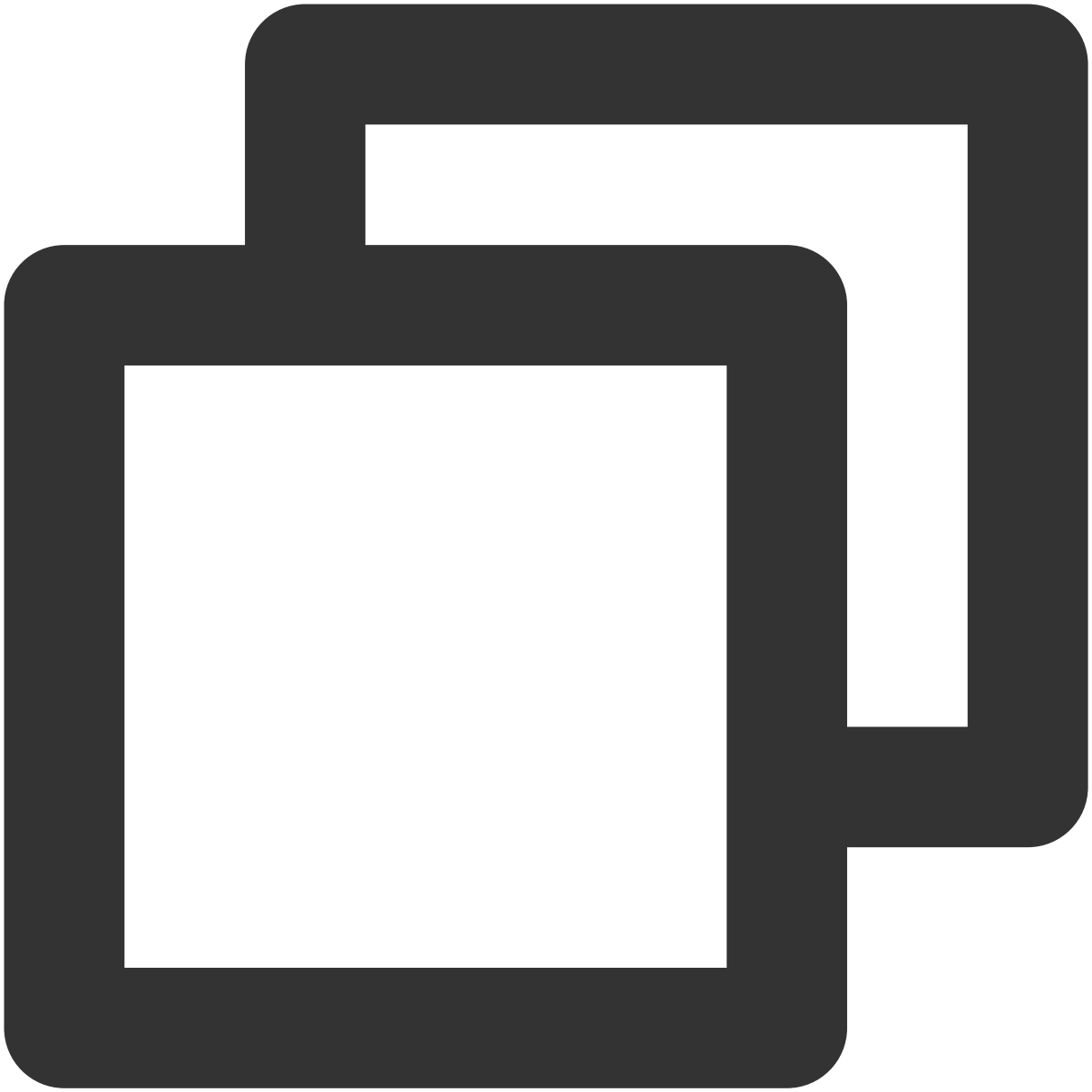
```
pod update
```

Upon the completion of pod command execution, an .xcworkspace project file integrated with the SDK will be generated. Double-click to open it.

Note:

If the pod search fails, it is recommended to try updating the local repo cache of pod. The update command is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



```
pod setup
pod repo update
rm ~/Library/Caches/CocoaPods/search_index.json
```

Besides CocoaPods integration, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manual Integration of IM SDK.

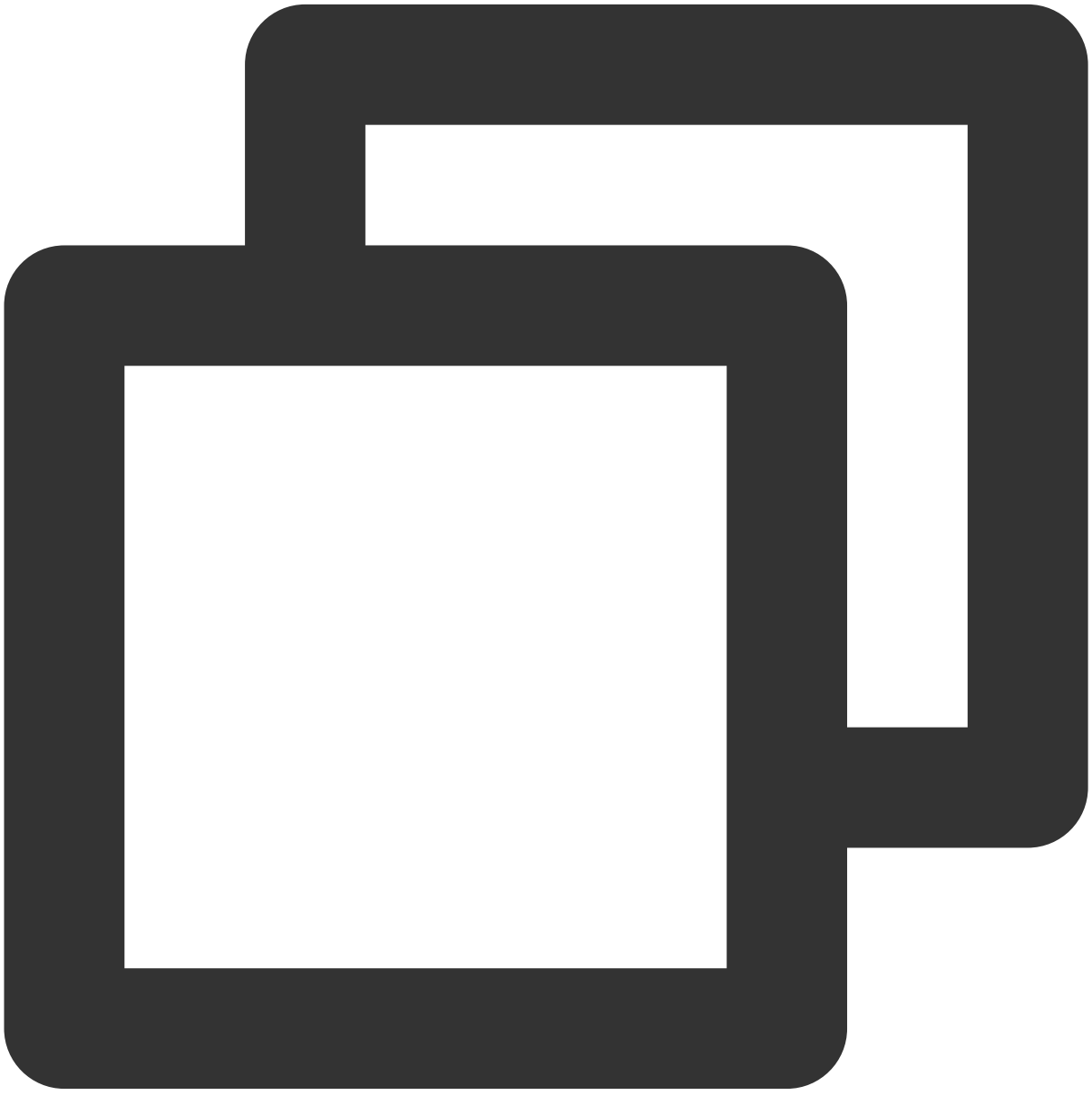
Quic plugin offers axp-quic Multiplexing Transmission Protocol, providing better resistance to poor networks. Even with a packet loss rate of 70%, it still can offer services. **Available only for Flagship users**. For non-Flagship users,

©2013-2022 Tencent Cloud. All rights reserved.

purchase the Flagship package before use, and see Pricing Instructions. To ensure proper functionality, update **Terminal SDK** to **version 7.7.5282 or above.**

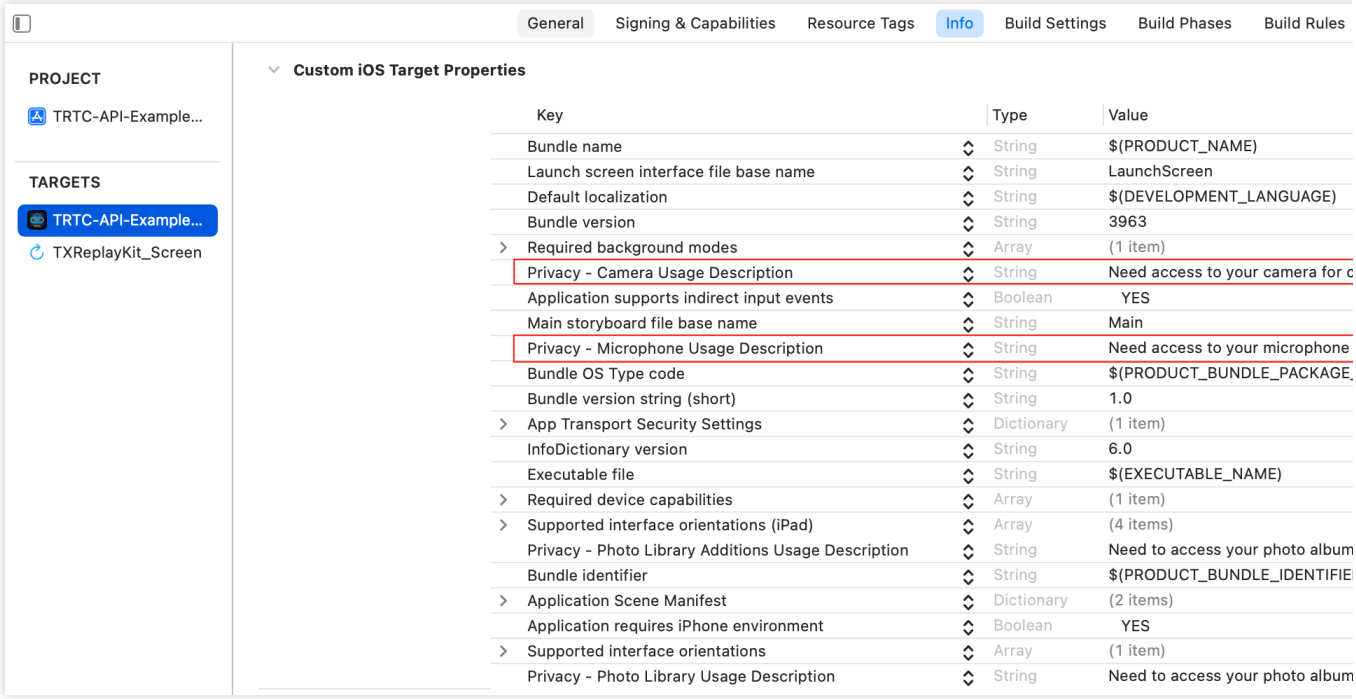
Step 3: Project configuration.

1. In voice chat scenarios, the TRTC SDK and IM SDK need to be authorized for mic permissions. Add the following content to your app's Info.plist. It corresponds to the system's prompt message in the dialog box when mic permissions are requested.



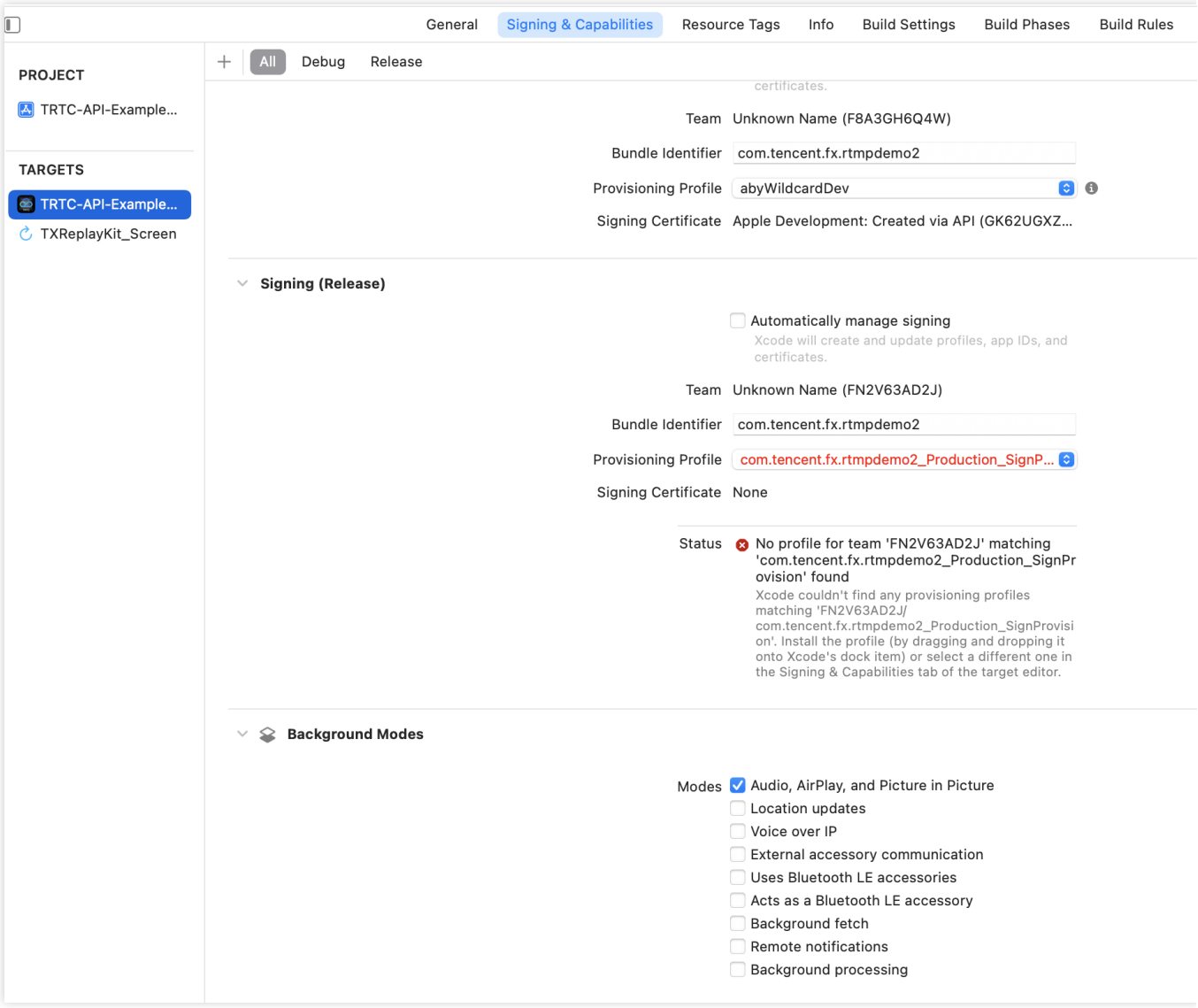
```
Privacy - Microphone Usage Description. Also enter a prompt specifying the purpose
```

©2013-2022 Tencent Cloud. All rights reserved.



2. If you need your App to continue running certain features in the background, go to XCode. Choose your current project. Under Capabilities, set the settings for Background Modes to ON, and check Audio, AirPlay, and Picture in Picture, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Integration Process

Step 1: Generate authentication credentials.

UserSig is a security protection signature designed by Tencent Cloud. Its purpose is to prevent malicious attackers from misappropriating your cloud service usage rights. Tencent Cloud's Tencent Real-Time Communication (TRTC) and Instant Messaging (IM) services both implement this security mechanism. TRTC authentication when entering a room, and IM authentication when logging in.

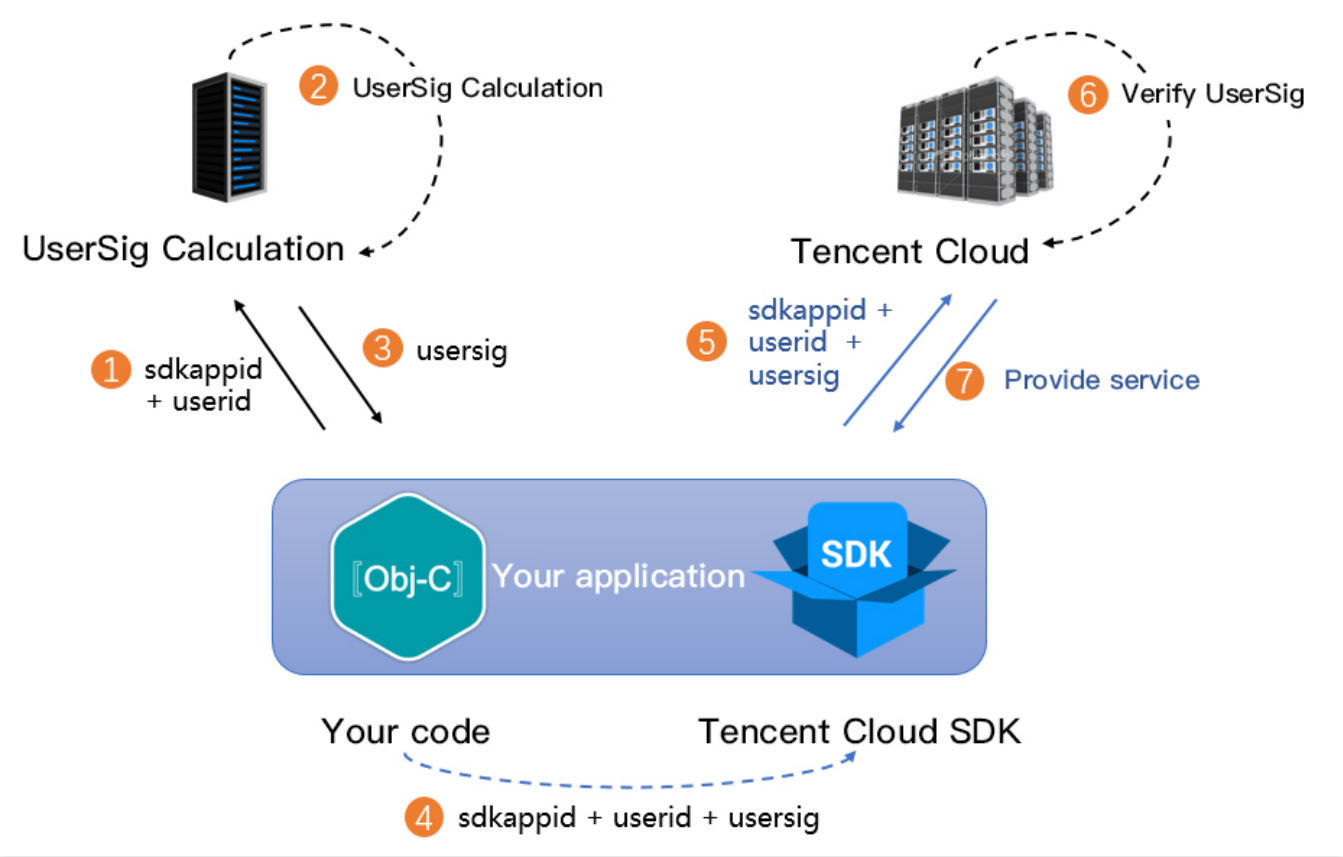
Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.

Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.

The specific implementation process is as follows:

©2013-2022 Tencent Cloud. All rights reserved.

1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.
3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. Once the verification is passed, it will provide instant communication services to the IM SDK and Tencent Real-Time Communication (TRTC) services to the TRTC SDK.



Note:

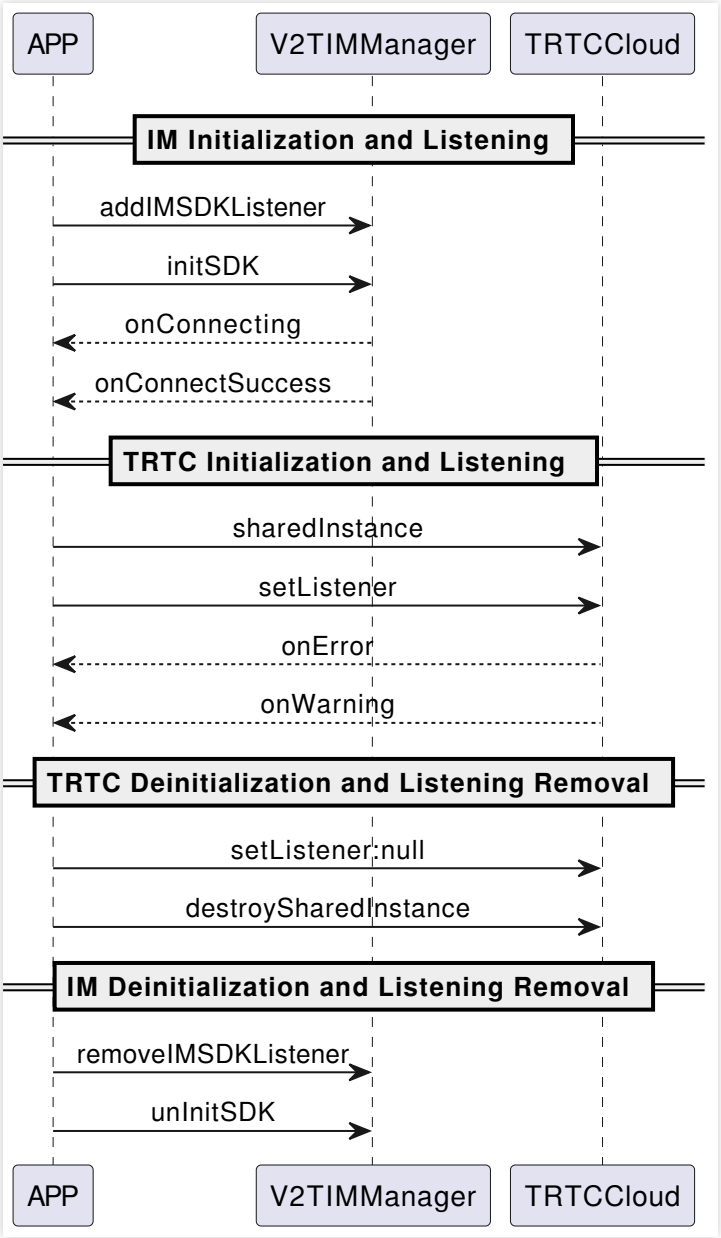
The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Step 2: Initialization and listening.

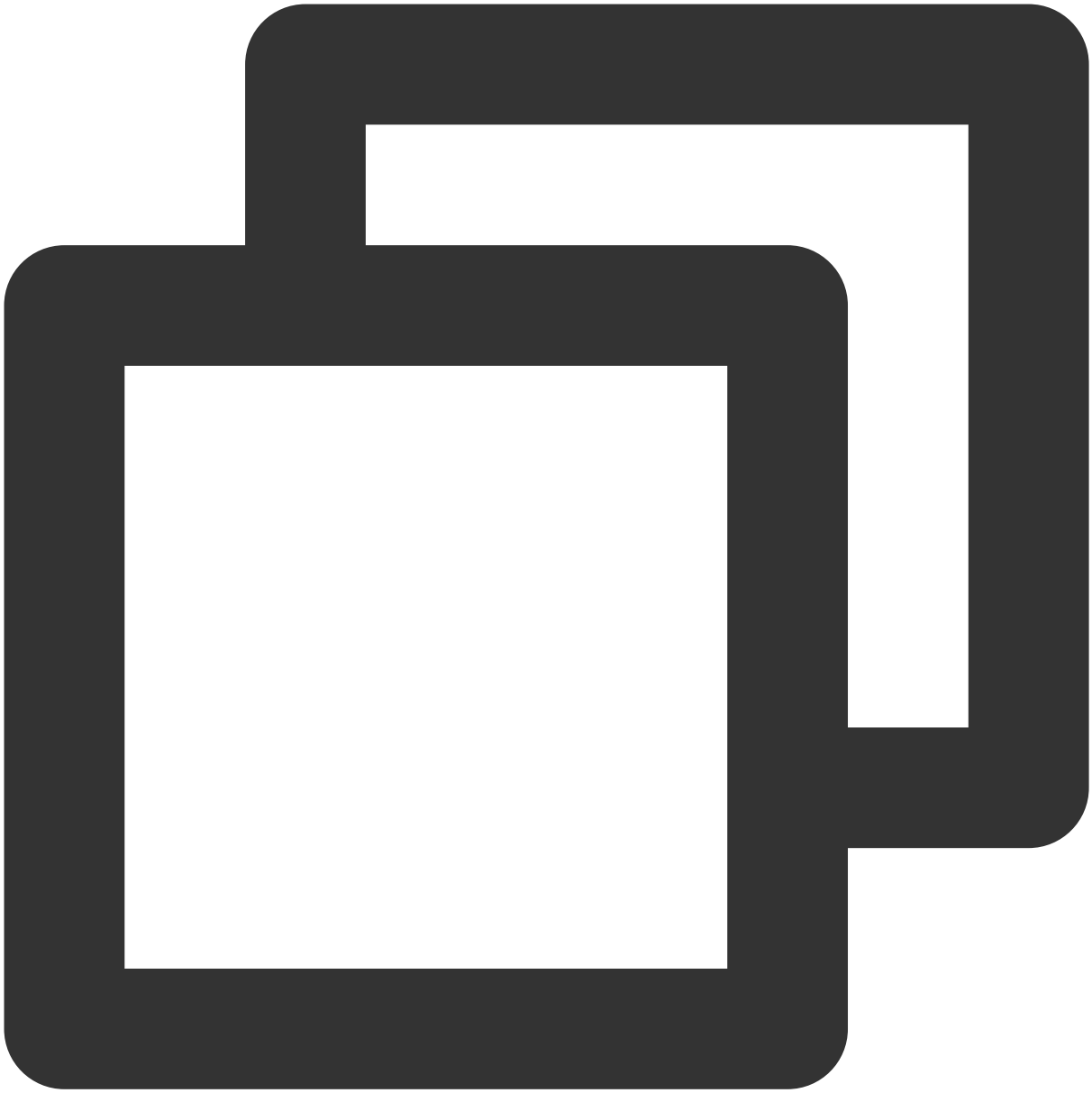
Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.



1. Initialize the IM SDK and add event listeners.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the SDKAppID from the Instant Messaging (IM) console.
// Add a V2TIMSDKListener event listener. self is the implementation class of id <V
[[V2TIMManager sharedInstance] addIMSDKListener:self];
// Initialize the IM SDK. After calling this API, you can immediately call the log-
[[V2TIMManager sharedInstance] initSDK:sdkAppID config:config];

// After the SDK is initialized, it will trigger various events, such as connection
- (void)onConnecting {
    NSLog(@"The IM SDK is connecting to Tencent Cloud CVM.");
}
```

©2013-2022 Tencent Cloud. All rights reserved.

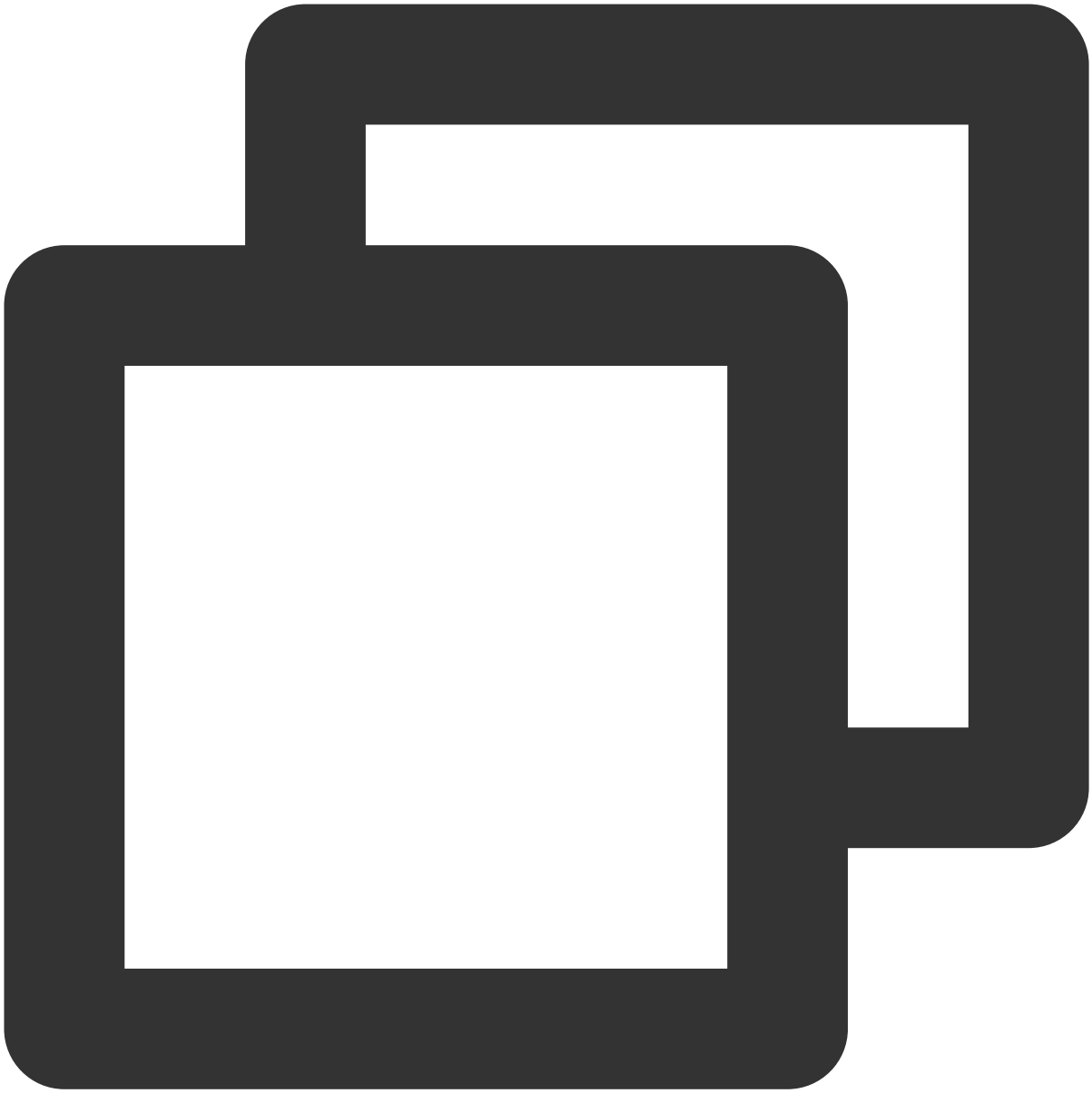
```
- (void)onConnectSuccess {
    NSLog(@"The IM SDK has successfully connected to Tencent Cloud CVM.");
}

// Remove event listener.
// self is the implementation class of id <V2TIMSDKListener>.
[[V2TIMManager sharedInstance] removeIMSDKListener:self];
// Deinitialize the SDK.
[[V2TIMManager sharedInstance] unInitSDK];
```

Note:

If your application's lifecycle is consistent with the SDK's lifecycle, you do not need to deinitialize before exiting the application. If you only initialize the SDK after entering a specific interface and no longer use it after exiting, you may deinitialize the SDK.

2. Create TRTC SDK instances and set event listeners.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (Single Instance Pattern).
_trtcCloud = [TRTCCloud sharedInstance];
// Set event listeners.
_trtcCloud.delegate = self;

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
- (void)onError:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg extInfo:(
    NSLog(@"%d: %@", errCode, errMsg);
}

- (void)onWarning:(TXLiteAVWarning)warningCode warningMsg:(nullable NSString *)warn
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    NSLog(@"%d: %@", warningCode, warningMsg);
}

// Remove event listener.
_trtcCloud.delegate = nil;
// Terminate TRTC SDK instance (Singleton Pattern).
[TRTCCloud destroySharedIntance];
```

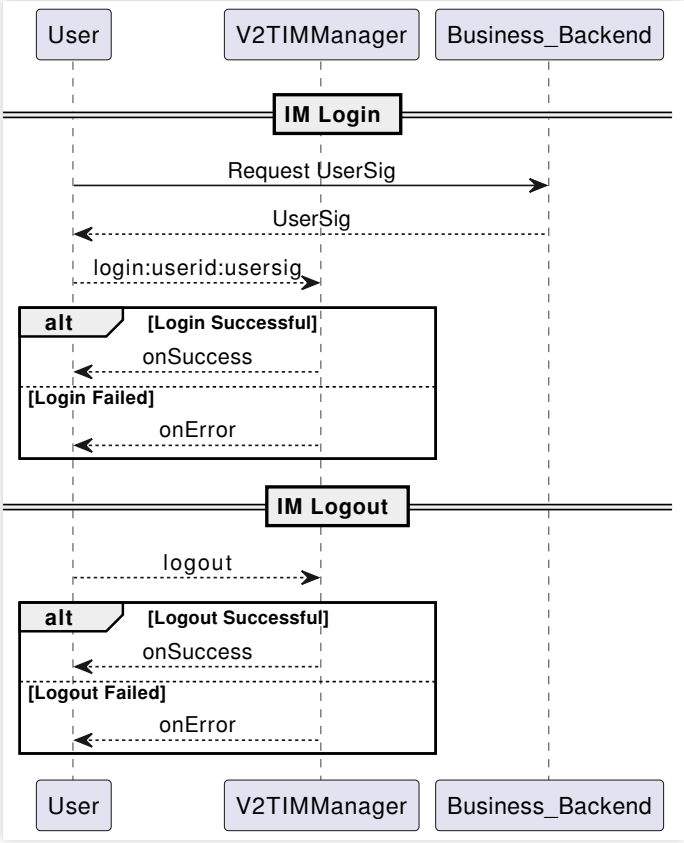
Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

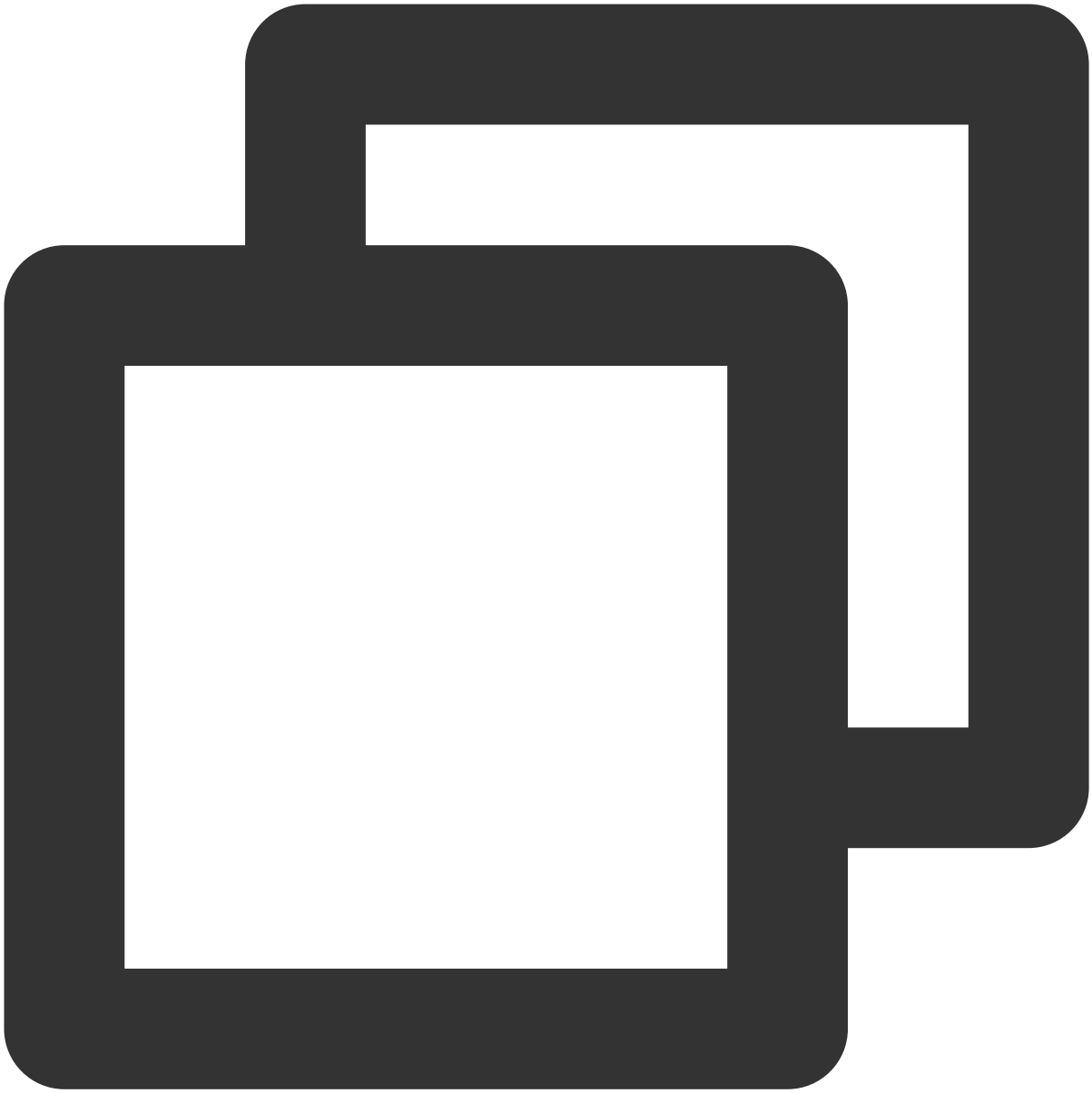
Step 3: Log in and log out.

After initializing the IM SDK, you need to call the SDK log-in API to authenticate your account identity and have permissions to use features. Before using any other features, ensure you are successfully logged in, or you might encounter feature malfunctions or unavailability. If you only need to use TRTC's audio and video services, you can skip this step.

Sequence diagram



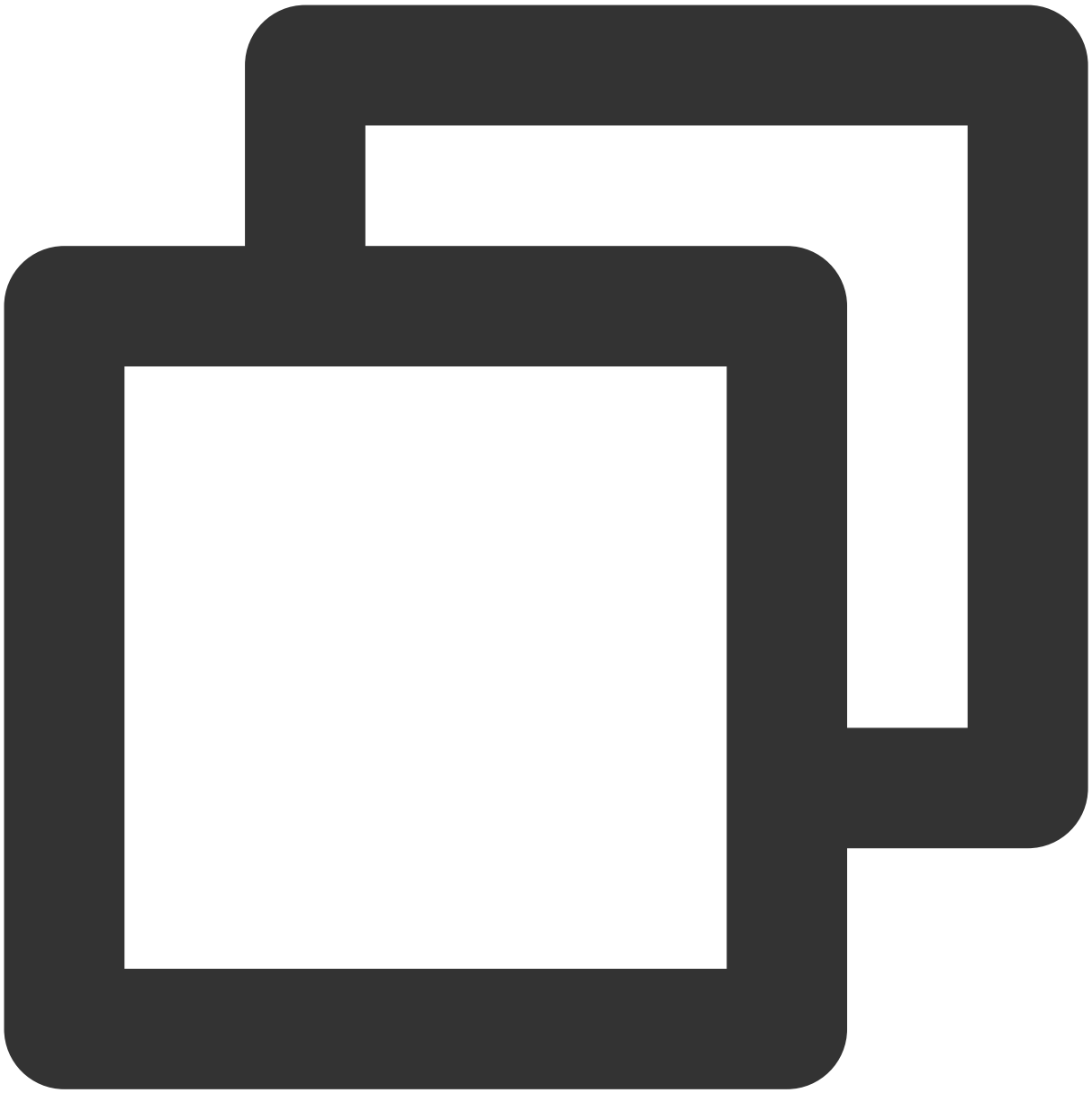
1. Log in.



```
// Log in: userID can be defined by the user and userSig can be generated as per St
[[V2TIMManager sharedInstance] login:userID userSig:userSig succ:^{
    NSLog(@"success");
} fail:^(int code, NSString *desc) {
    // The following error codes mean an expired UserSig, and you need to generate
    // 1. ERR_USER_SIG_EXPIRED(6206).
    // 2. ERR_SVR_ACCOUNT_USERSIG_EXPIRED(70001).
    // Note: Do not call the log-in API in case of other error codes. Otherwise, th
    NSLog(@"failure, code:%d, desc:%@", code, desc);
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

2. Log out.



```
// Log out.
[[V2TIMManager sharedInstance] logout:^{
    NSLog(@"success");
} fail:^(int code, NSString *desc) {
    NSLog(@"failure, code:%d, desc:%@", code, desc);
}];
```

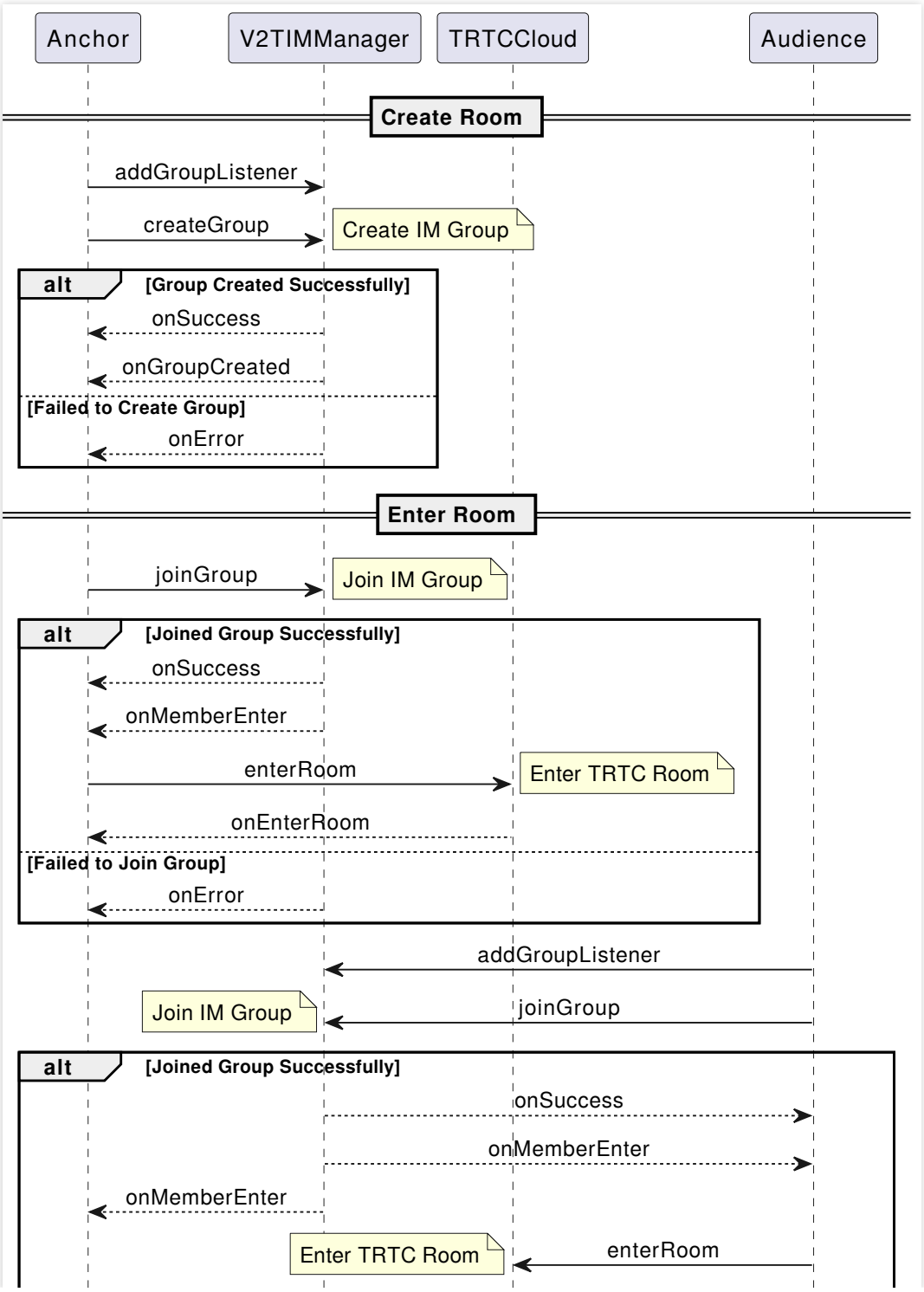
Note:

©2013-2022 Tencent Cloud. All rights reserved.

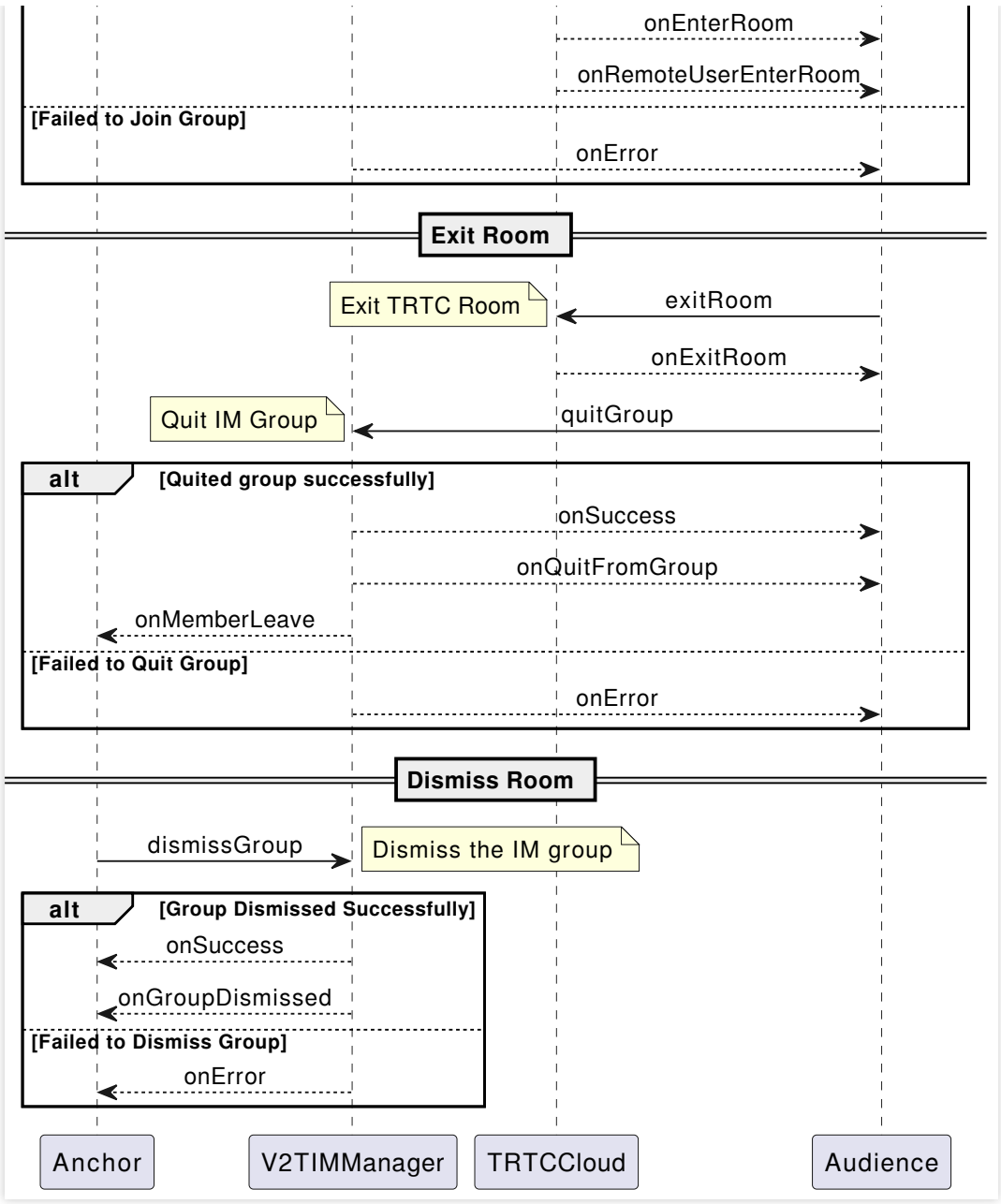
If your application's lifecycle matches the IM SDK's lifecycle, logging out before exiting the application is not necessary. However, if you only use the IM SDK after entering a specific interface and stop using it after exiting, you can log out and deinitialize the IM SDK.

Step 4: Room management.

Sequence diagram



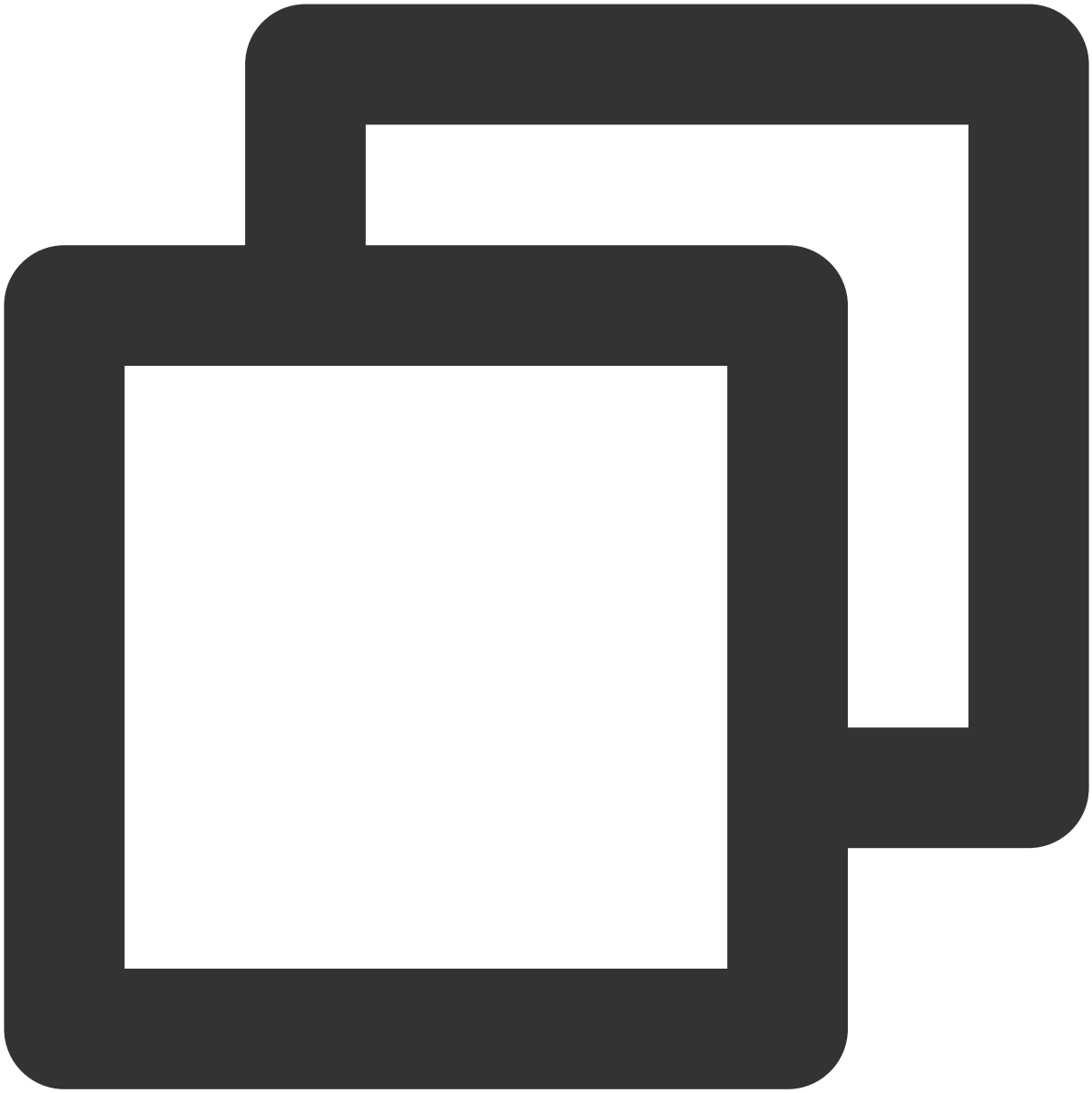
©2013-2022 Tencent Cloud. All rights reserved.



1. Create a room.

When the anchor (room owner) starts live streaming, a room needs to be created. The concept of "room" here corresponds to "group" in IM. This example only shows how to create an IM group on the client, but it is also possible to create a group on the server.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create a group.
[[V2TIMManager sharedInstance] createGroup:GroupType_AVChatRoom groupID:groupID gro
    // Group created successfully.
} fail:^(int code, NSString *desc) {
    // Group creation failed.
}];


// Listen for group creation notifications.
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupCreated:(NSString *)groupID {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Group creation callback. groupID is the ID of the newly created group.
}
```

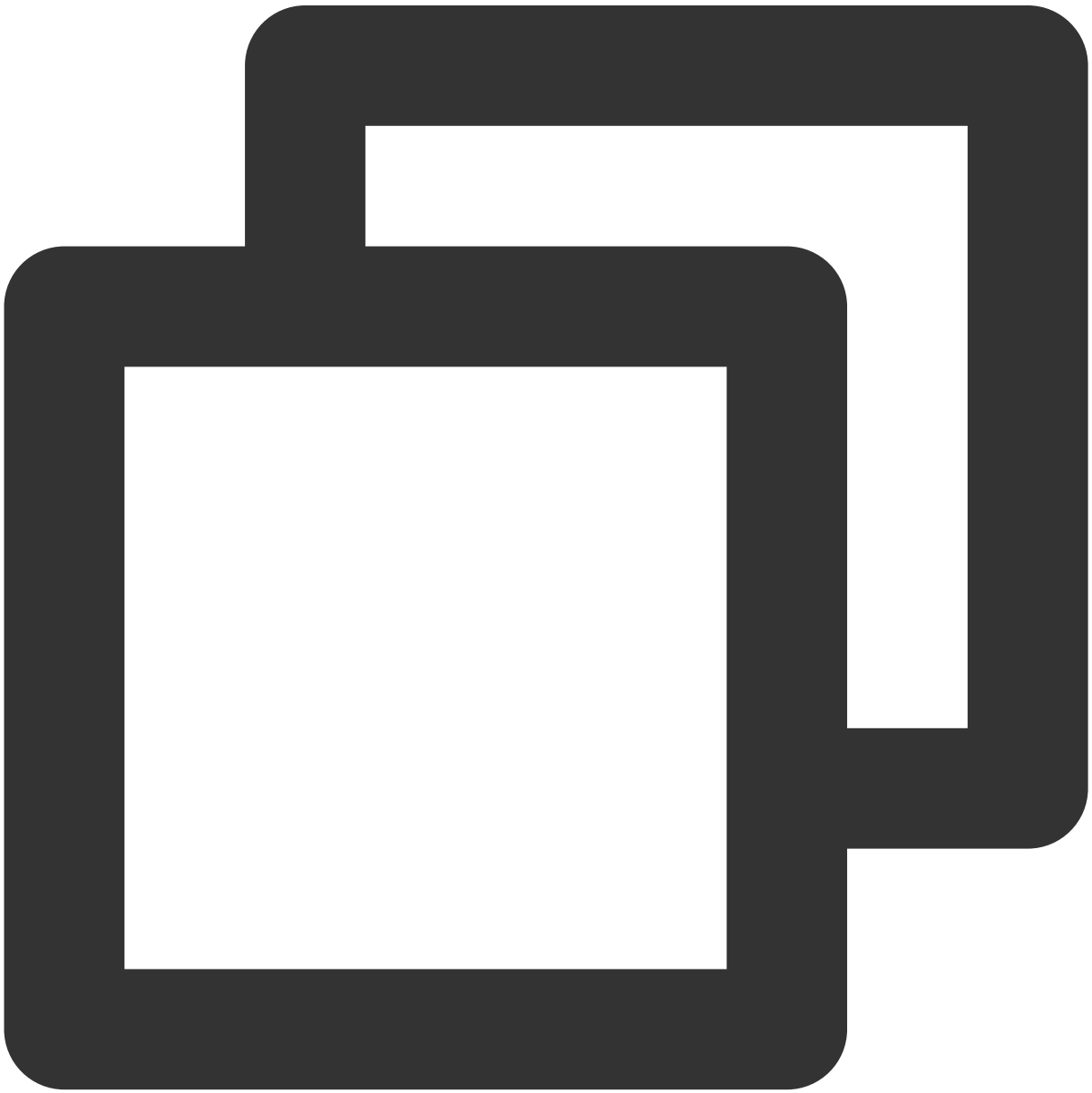
Note:

For creating an IM group in the voice chat room scenarios, select the live streaming group type: `GroupType_AVChatRoom`.

TRTC does not have a room-creation API, so when a user attempts to join a room that does not exist, the backend automatically creates a room.

2. Join a room.

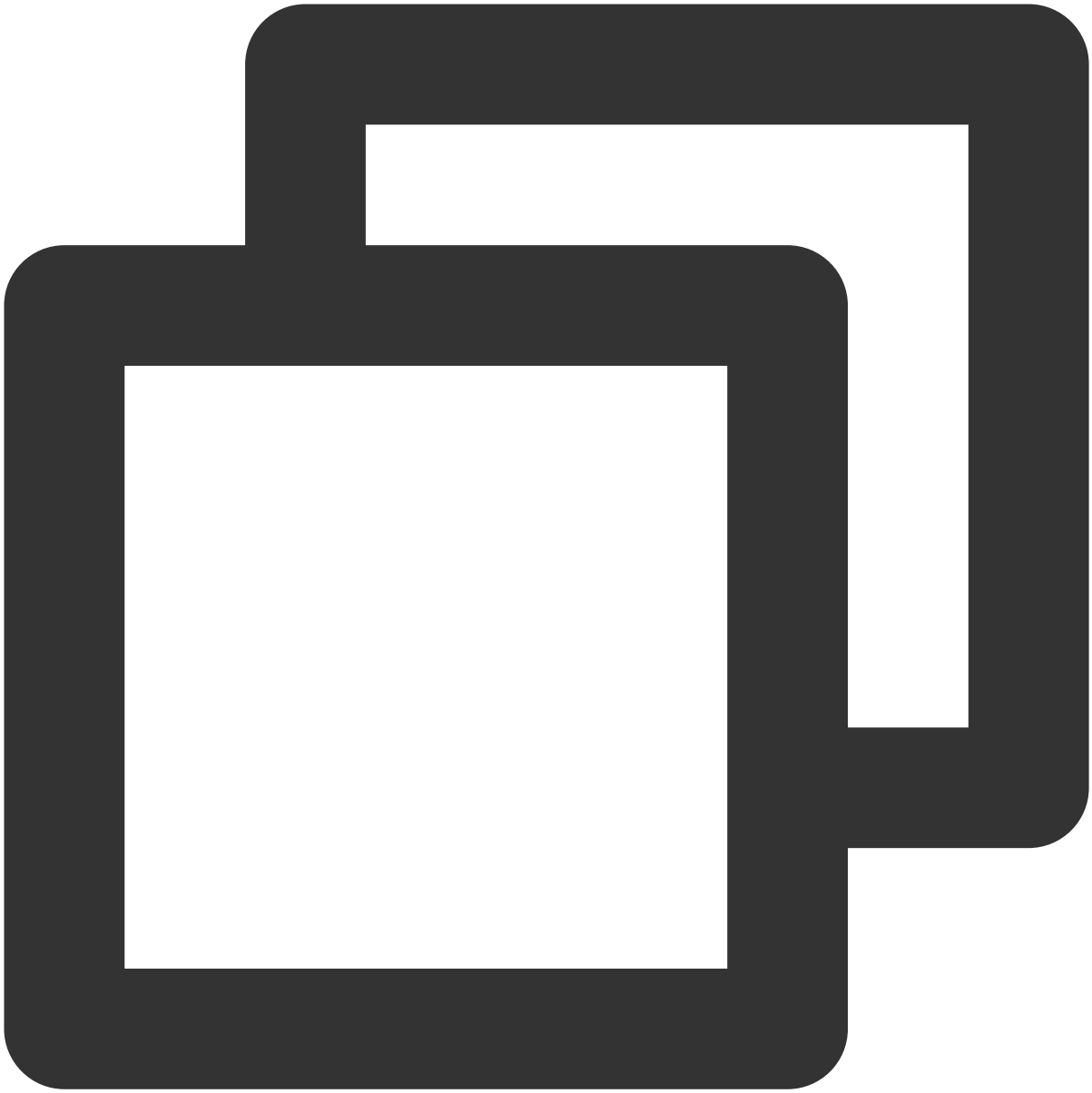
Join IM group.



```
// Join a group.
[[V2TIMManager sharedInstance] joinGroup:groupID msg:message succ:^{
    // Successfully joined the group.
} fail:^(int code, NSString *desc) {
    // Failed to join the group.
}];

// Listen for the event of joining a group.
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onMemberEnter:(NSString *)groupID memberList:(NSArray<V2TIMGroupMemberInfo
    // Someone joined the group.
}
```

Join a TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Using a string as the room ID for example, it is recommended to keep it cons
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // For entering a room in voice chat interaction scenarios, specify the user's
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Use room entry in voice chat interaction scenarios as an example.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneVoiceChatRoom];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        [self toastTip:@"Enter room succeed!"];
    } else {
        // result indicates the error code when you fail to enter the room.
        [self toastTip:@"Enter room failed!"];
    }
}
```

Note:

TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

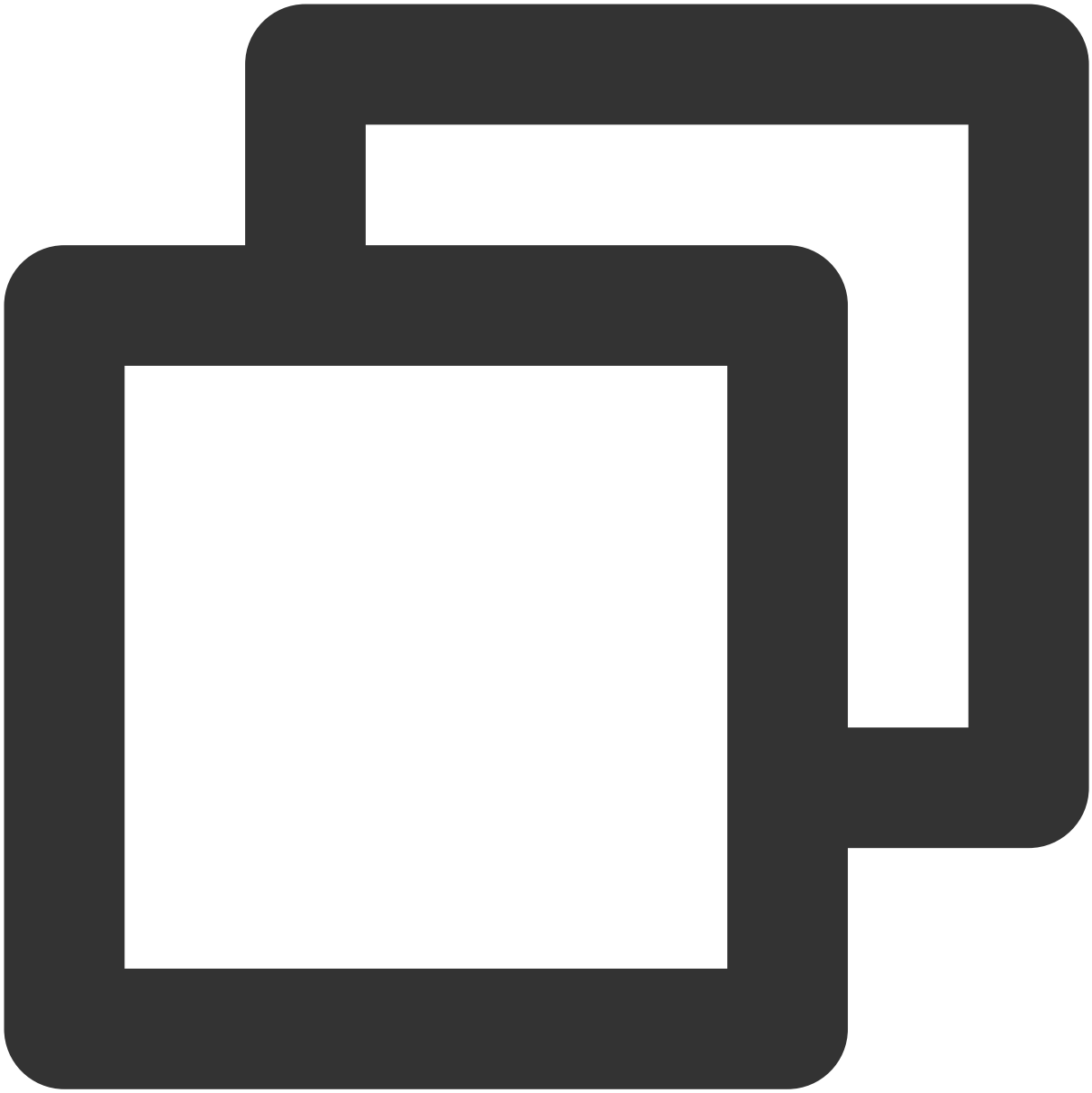
When entering a room in voice chat interaction scenarios, it is necessary to specify the user's role (anchor/audience). Only anchors have permissions to push streams. If not specified, the default role is anchor.

For entering a room in voice chat interaction scenarios, it is recommended to select `TRTCAppSceneVoiceChatRoom`.

3. Exit the room.

Exit the IM group.

©2013-2022 Tencent Cloud. All rights reserved.



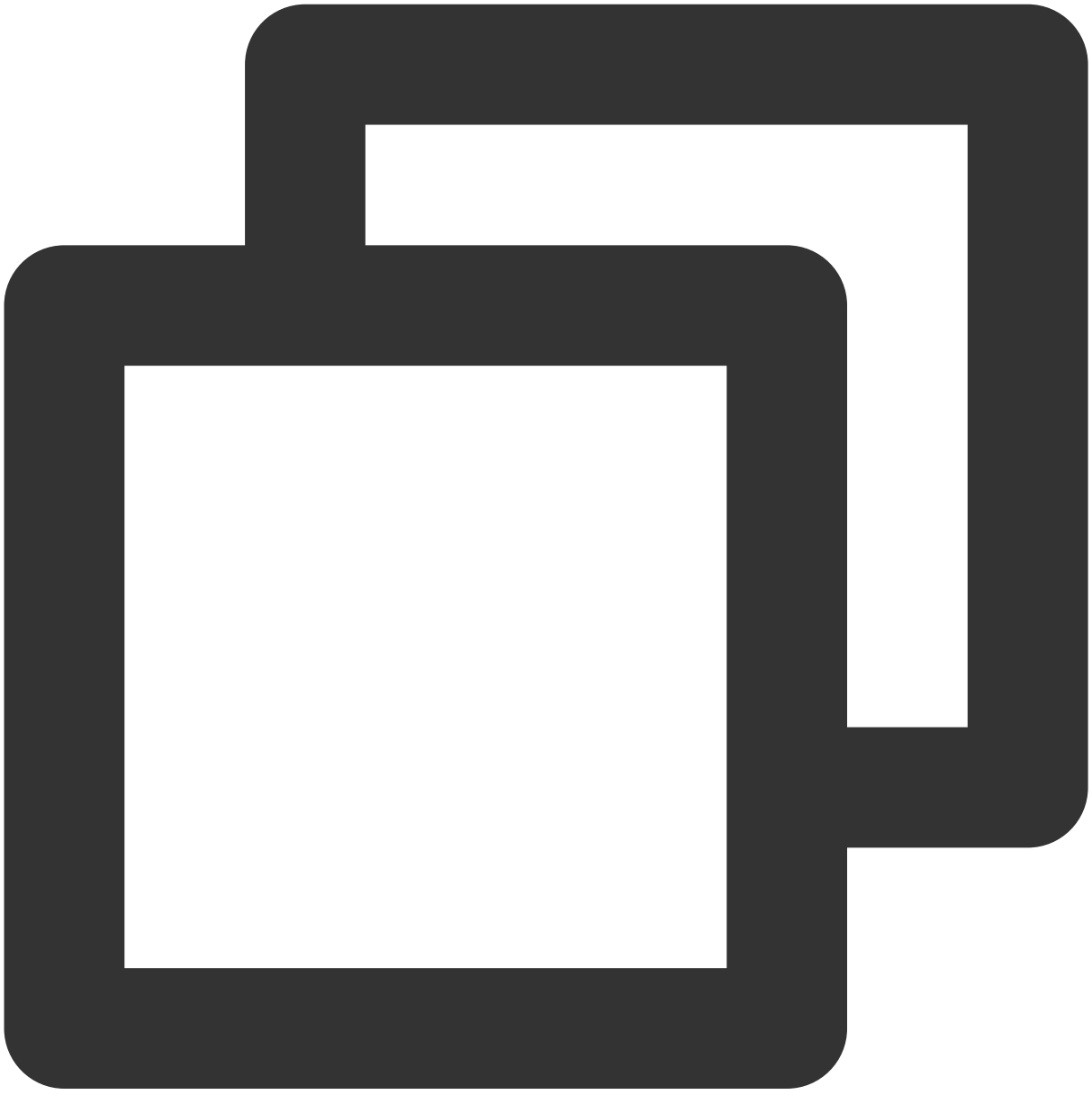
```
[[V2TIMManager sharedInstance] quitGroup:groupID succ:^{
    // Exiting the group successful.
} fail:^(int code, NSString *desc) {
    // Exiting the group failed.
}];

[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onMemberLeave:(NSString *)groupID member:(V2TIMGroupMemberInfo *)member {
    // Group member leave callback.
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

In a live streaming group (AVChatRoom), the group owner cannot exit the group. The owner can only dissolve the group by calling `dismissGroup` .

Exit the TRTC room.



```
- (void)exitTrtcRoom {
    self.trtcCloud = [TRTCCloud sharedInstance];
    [self.trtcCloud stopLocalAudio];
    [self.trtcCloud exitRoom];
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Listen for the onExitRoom callback to get the reason for exiting the room.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        // Actively call exitRoom to exit the room.
        NSLog(@"Exit current room by calling the 'exitRoom' api of sdk ...");
    } else if (reason == 1) {
        // Removed from the current room by the server.
        NSLog(@"Kicked out of the current room by server through the restful api...
    } else if (reason == 2) {
        // The current room is dissolved.
        NSLog(@"Current room is dissolved by server through the restful api...");
    }
}
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

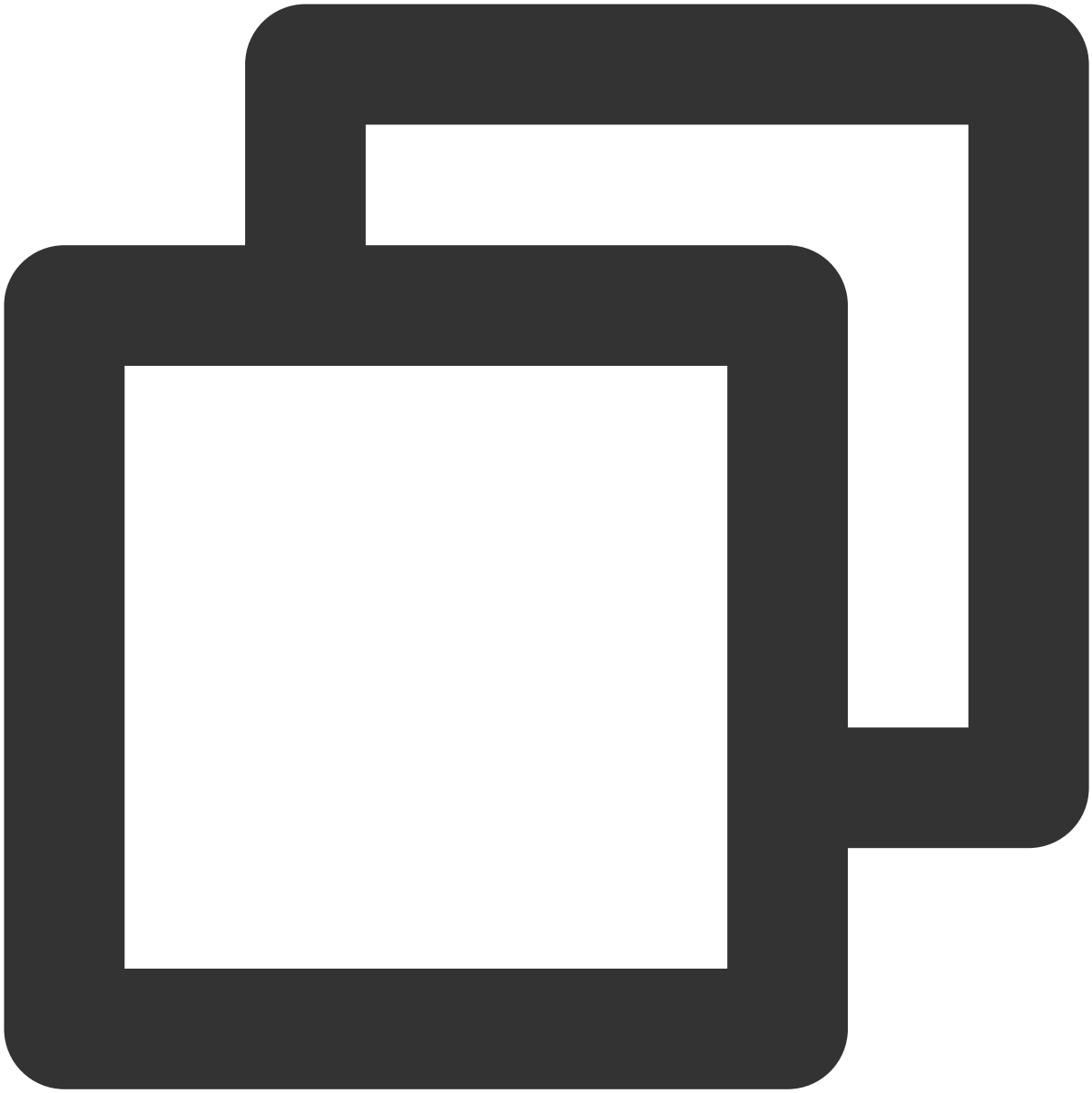
If you want to call `enterRoom` again or switch to another audio/video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter exceptions such as the camera or microphone being forcefully occupied.

4. Dissolve the room.

Dissolve the IM group.

This example only shows the client method of dissolving an IM group. It can also be done through server dissolves the group.

©2013-2022 Tencent Cloud. All rights reserved.



```
[[V2TIMManager sharedInstance] dismissGroup:groupID succ:^{
    // Dissolving group successful.
} fail:^(int code, NSString *desc) {
    // Dissolving the group failed.
}];

[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupDismissed:(NSString *)groupID opUser:(V2TIMGroupMemberInfo *)opUser
    // Group dissolved callback.
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Dissolve TRTC room.

Server dissolvement: TRTC provides the Server dissolves the room API `DismissRoom` (differentiating between numeric room ID and string room ID). You can call this API to remove all users from the room and dissolve the room.

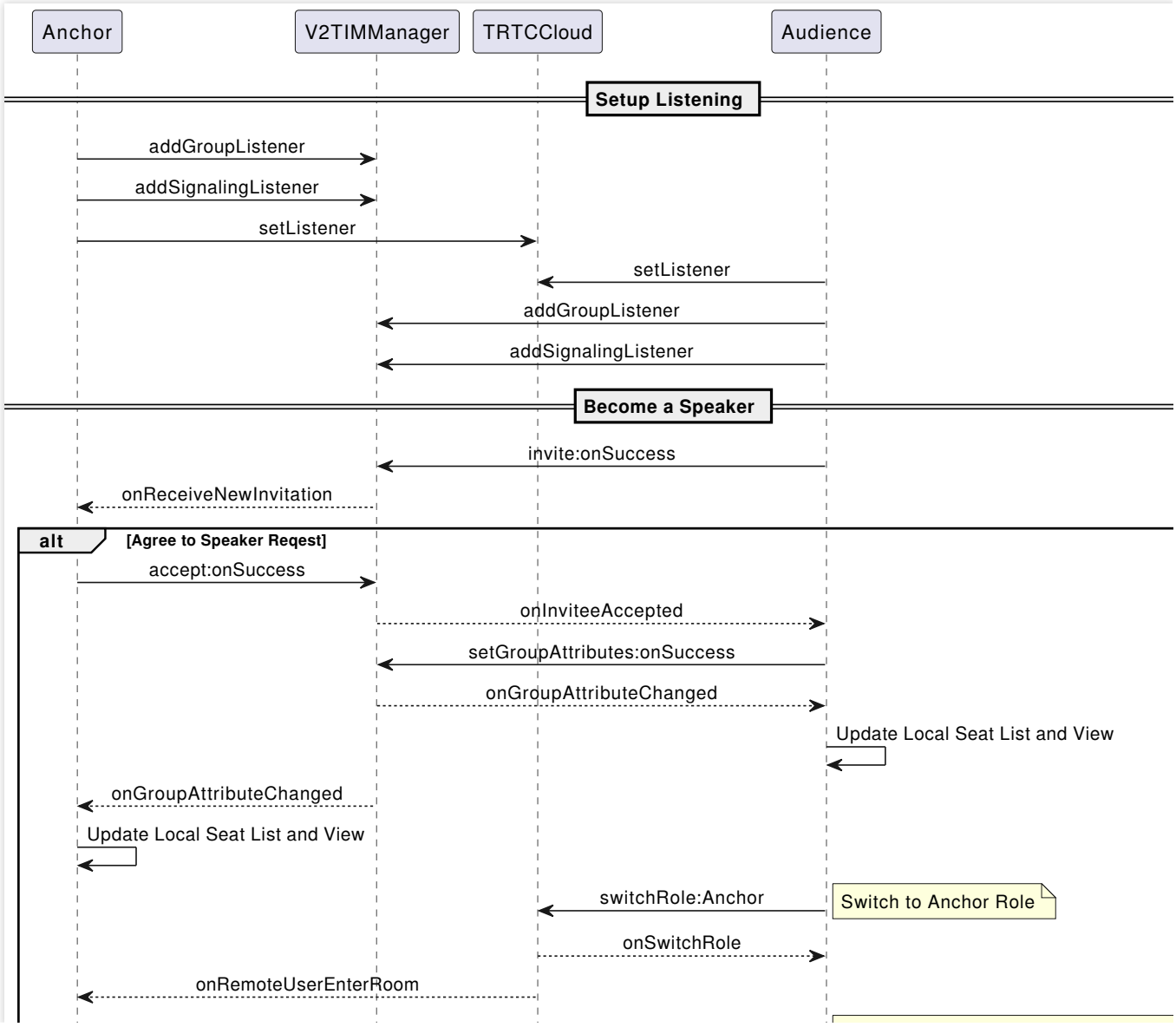
Client dissolvement: Through the room exit `exitRoom` API of each client, all the anchors and audiences in the room can be completed of room exit. After room exit, according to TRTC room lifecycle rules, the room will automatically be dissolved. For details, see Exit Room.

Warning:

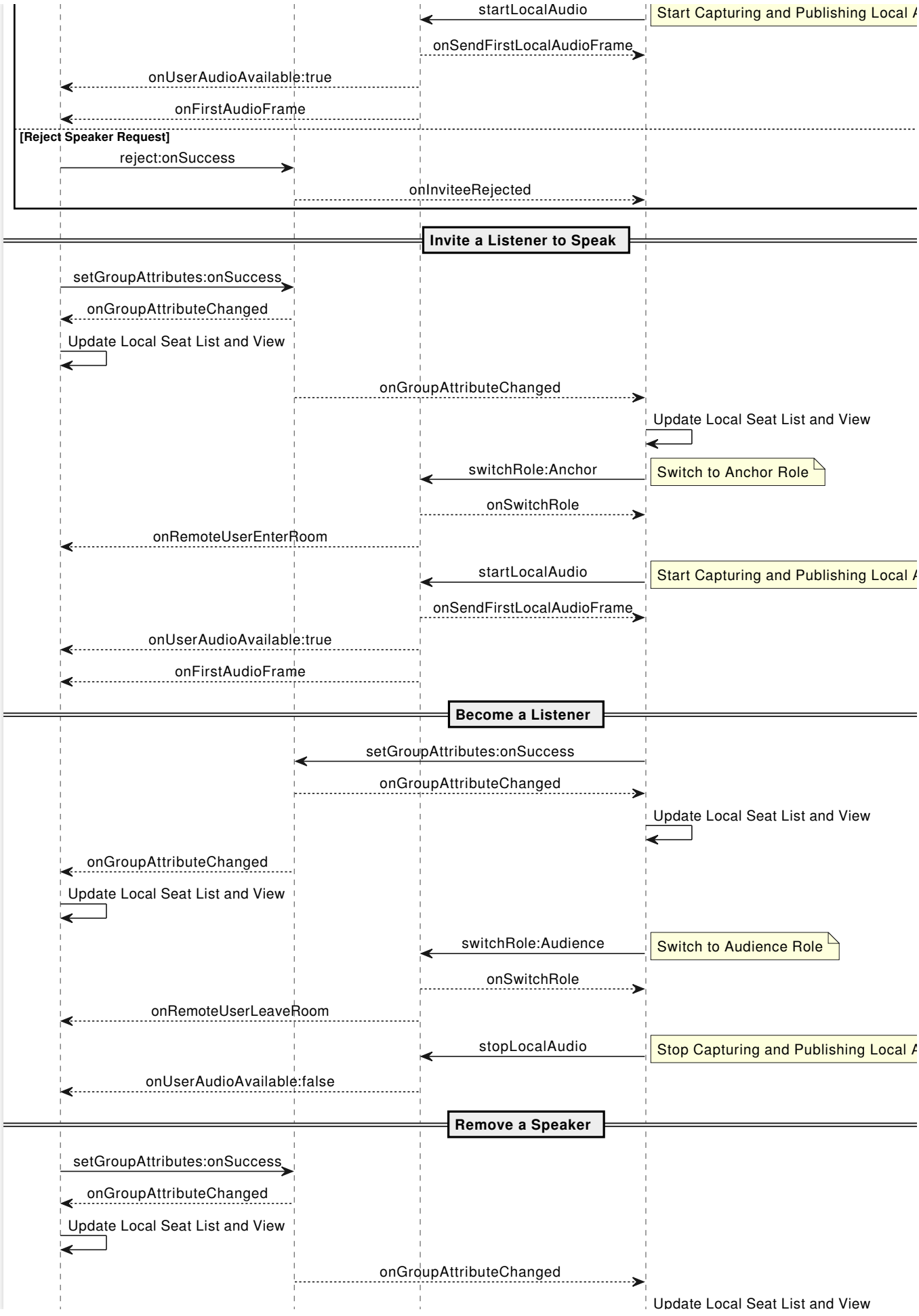
It is recommended that after the end of live streaming, you call the room dissolvement API to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

Step 5: Seat management.

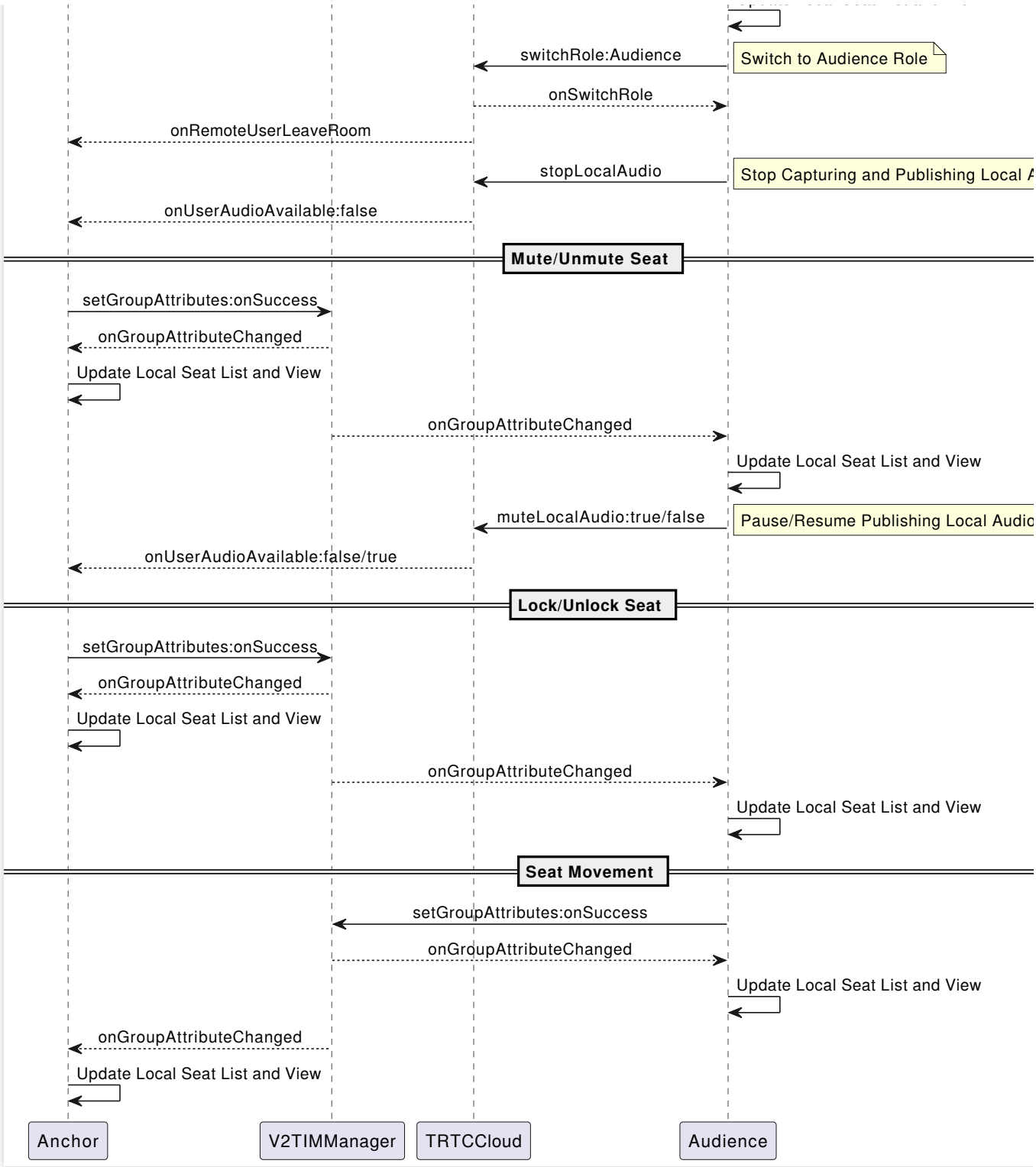
Sequence diagram



©2013-2022 Tencent Cloud. All rights reserved.

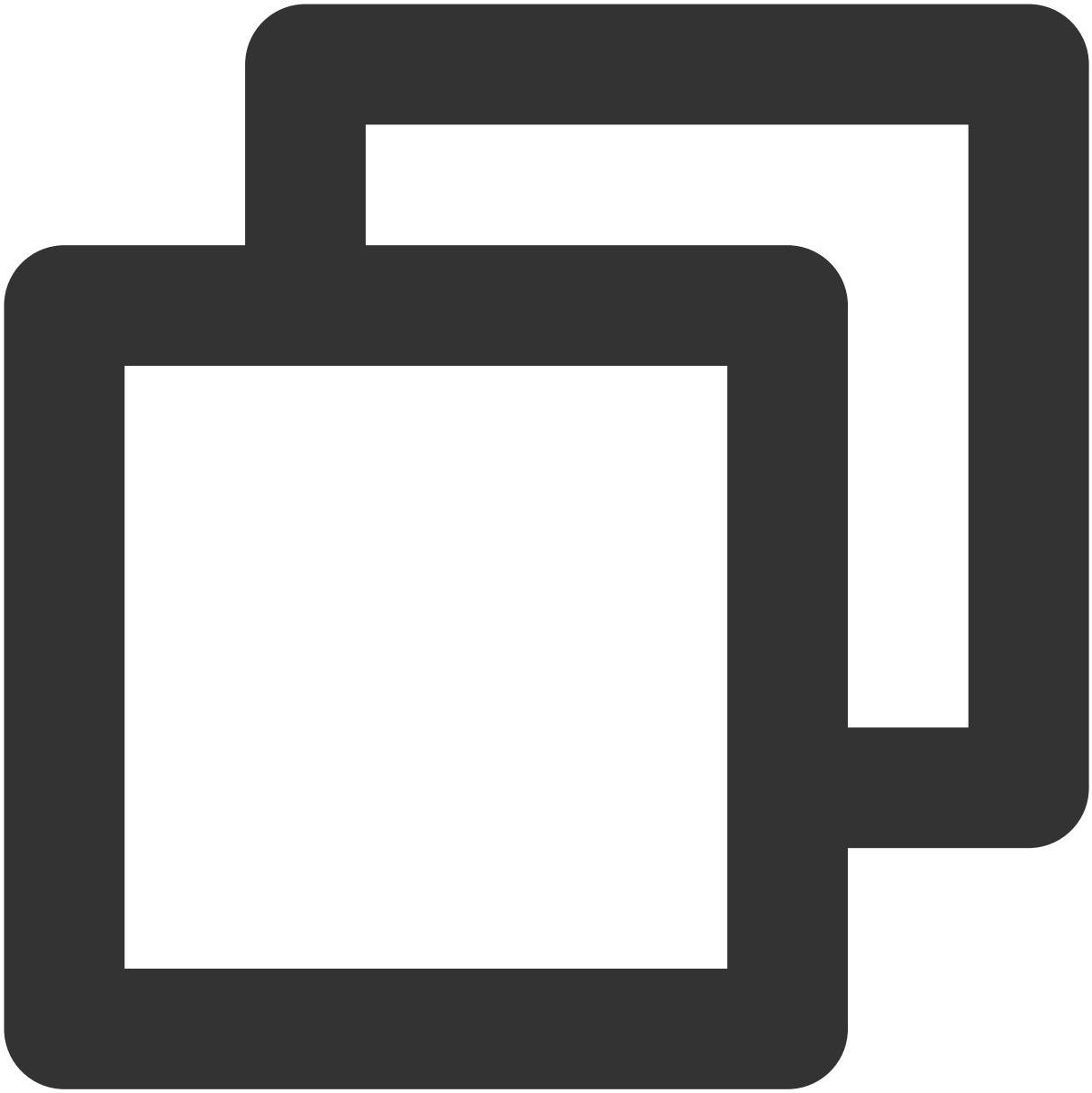


©2013-2022 Tencent Cloud. All rights reserved.



Firstly, we can create a model to save seat information.

©2013-2022 Tencent Cloud. All rights reserved.



```
#import "JSONModel.h"

typedef NS_ENUM(NSUInteger, SeatInfoStatus) {
    SeatInfoStatusUnused = 0,
    SeatInfoStatusUsed = 1,
    SeatInfoStatusLocked = 2,
};

NS_ASSUME_NONNULL_BEGIN

@interface SeatInfoModel : JSONModel
```

©2013-2022 Tencent Cloud. All rights reserved.

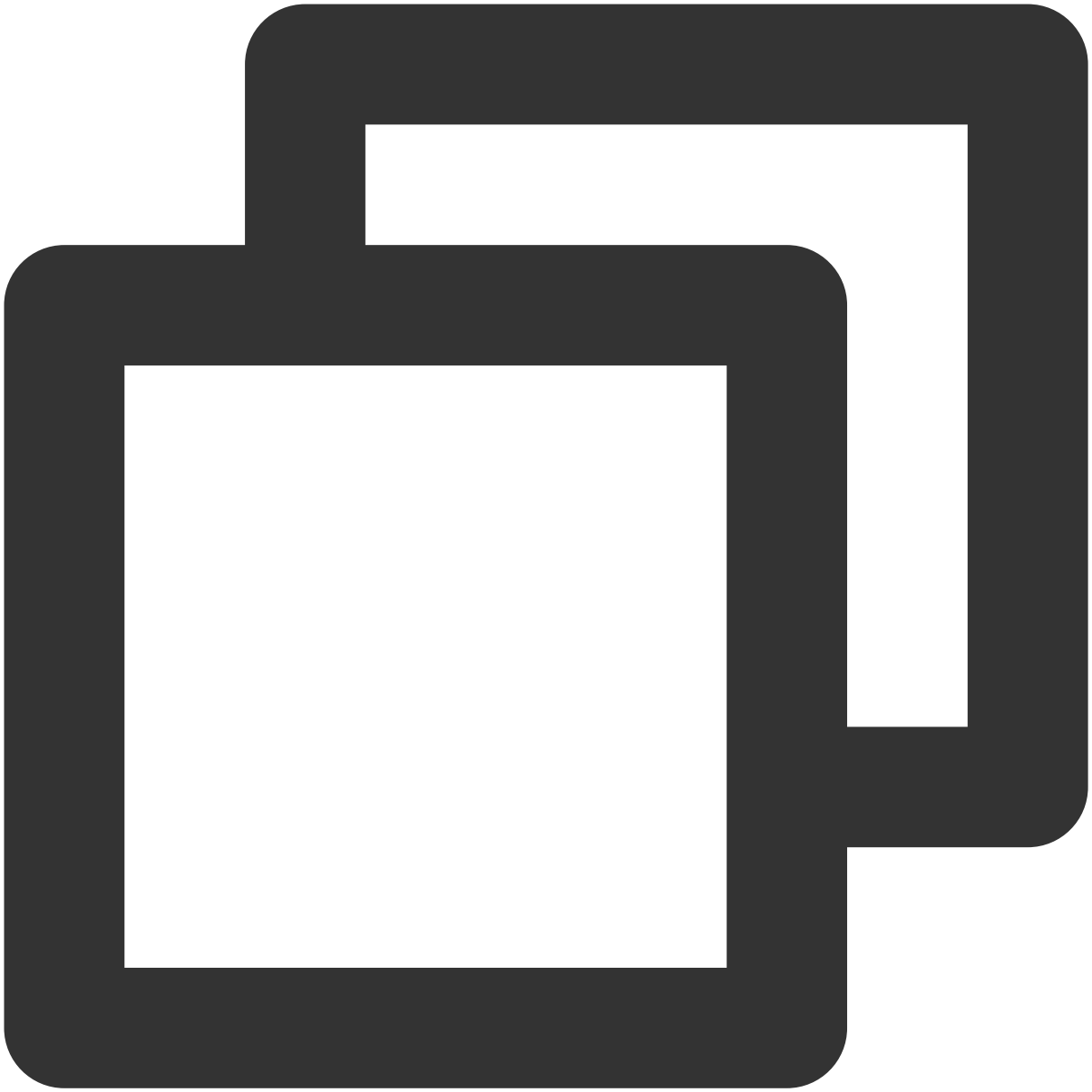
```
/// Seat status, corresponding to three statuses.
@property (nonatomic, assign) SeatInfoStatus status;
/// Whether the seat is muted.
@property (nonatomic, assign) BOOL mute;
/// When the seat is occupied, store the user information.
@property (nonatomic, copy) NSString *userId;

@end

NS_ASSUME_NONNULL_END
```

1. Become a speaker.

Becoming a speaker refers to off-mic audience sending a request to speak to the room owner or administrator. The audience can speak once the approval signaling is received. In a free-speaking mode, the signaling request part can be skipped.

Audience sends a request to speak.

©2013-2022 Tencent Cloud. All rights reserved.

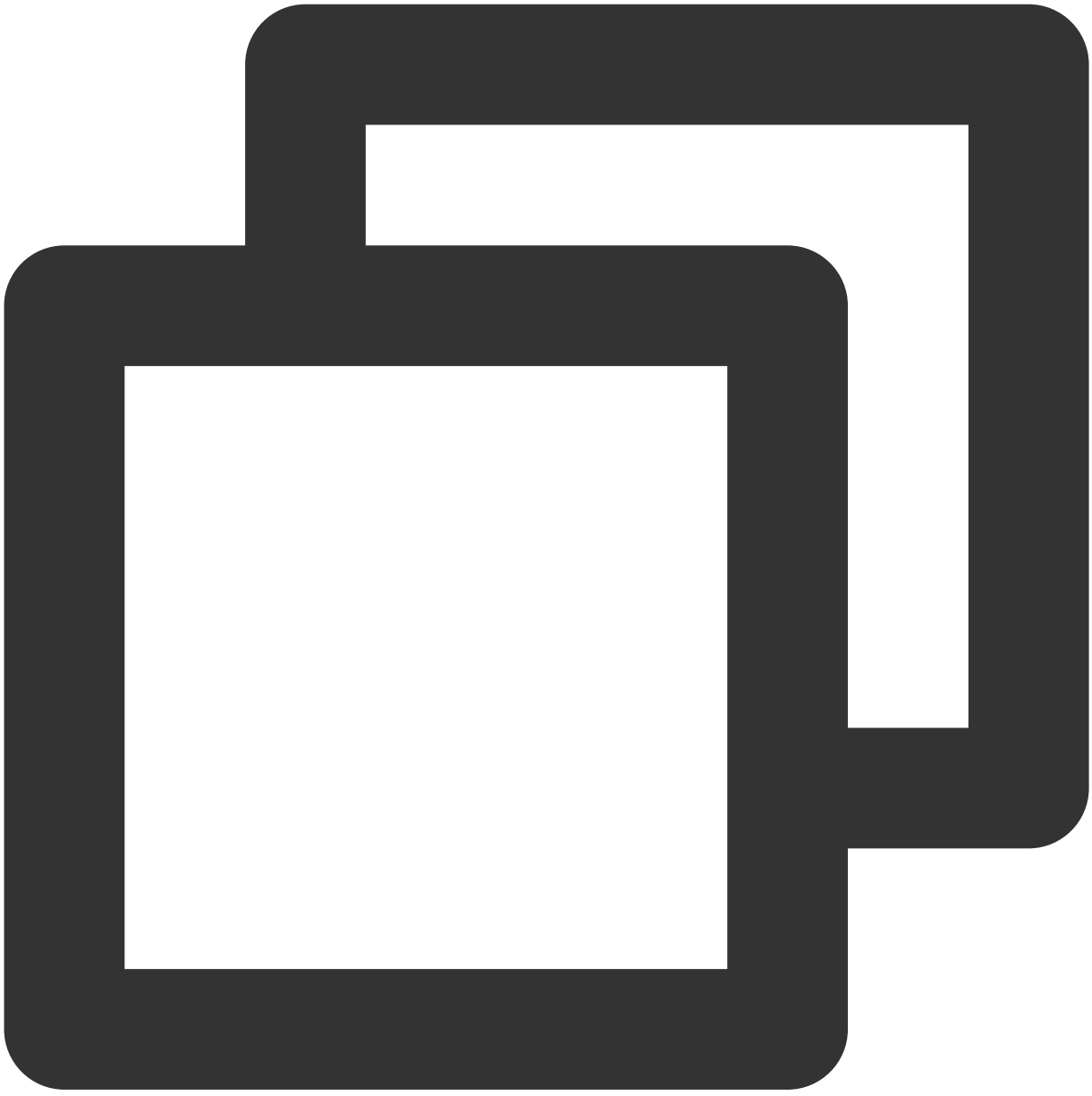


```
// Audience sends a request to speak. userId is the Anchor ID, and data can pass in
- (void)sendInvitationWithUserId:(NSString *)userId data:(NSString *)data {
    [[V2TIMManager sharedInstance] invite:userId data:data onlineUserOnly:YES offli
        NSLog(@"sendInvitation success");
    } fail:^(int code, NSString *desc) {
        NSLog(@"sendInvitation error %d", code);
    }];
}

// Anchor receives the request to speak. inviteID is the request ID, and inviter is
[[V2TIMManager sharedInstance] addSignalingListener:self];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)onReceiveNewInvitation:(NSString *)inviteID inviter:(NSString *)inviter gro
    NSLog(@"received invitation: %@ from %@", inviteID, inviter);
}
```

Anchor processes the request to speak.



```
// Agree to the request to speak.
- (void)acceptInvitationWithInviteID:(NSString *)inviteID data:(NSString *)data {
    [[V2TIMManager sharedInstance] accept:inviteID data:data succ:^{
        NSLog(@"acceptInvitation success");
    } fail:^(int code, NSString *desc) {
        NSLog(@"acceptInvitation error %d", code);
```

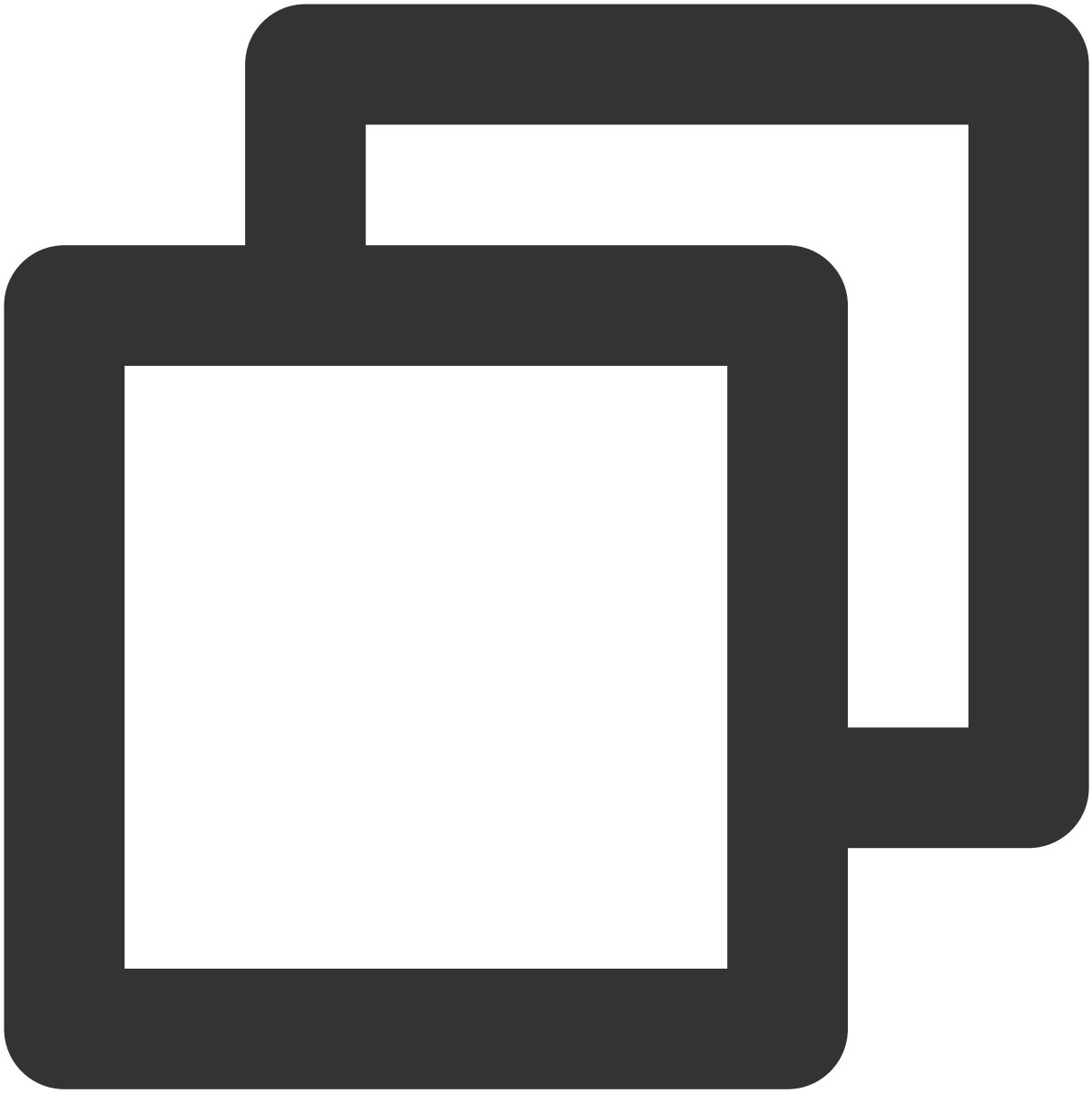
©2013-2022 Tencent Cloud. All rights reserved.

```
    }];
}

// Reject the request to speak.
- (void)rejectInvitationWithInviteID:(NSString *)inviteID data:(NSString *)data {
    [[V2TIMManager sharedInstance] reject:inviteID data:data succ:^{
        NSLog(@"rejectInvitation success");
    } fail:^(int code, NSString *desc) {
        NSLog(@"rejectInvitation error %d", code);
    }];
}
```

Audience to speak.

If the anchor agrees to the audience's request to speak, the audience can add seat information by modifying group attributes. Other users will receive a callback for the change in group attributes. Update the local seat information.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// Callback for agreeing to the request to speak.
- (void)onInviteeAccepted:(NSString *)inviteID invitee:(NSString *)invitee data:(NS
    NSLog(@"received accept invitation: %@ from %@", inviteID, invitee);
    NSInteger seatIndex = [self findSeatIndex:inviteID];
    [self takeSeatWithIndex:seatIndex];
}

// Audience begins to speak.
```

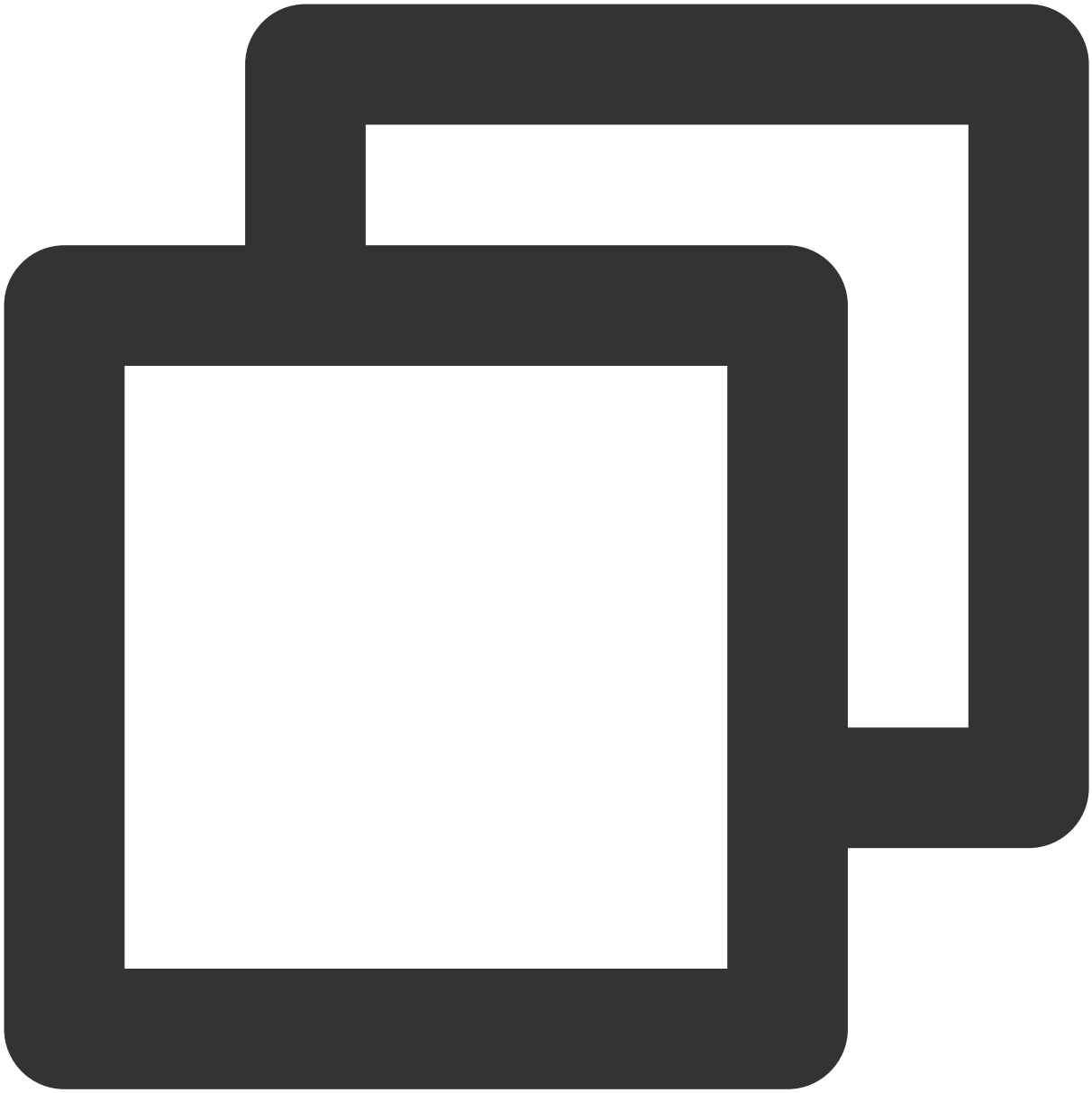
©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)takeSeatWithIndex:(NSInteger)seatIndex {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = SeatInfoStatusUsed;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = self.userId;

    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes. Switch TRTC role and start strea
        [self.trtcCloud switchRole:TRTCRoleAnchor];
        [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to become a speaker.
    }];
}
```

2. Invite a listener to speak.

Anchor invites a listener to speak (without the need for audience's consent). Directly modify group attributes saved for seats, and corresponding audiences will receive a callback for the change in group attributes. After matching the userId successfully, they can switch TRTC role and start streaming. In an invite-to-speak mode, see the implementation logic of becoming a speaker. Just switch the sender and receiver of the signaling.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// Anchor calls this API to modify the seat information saved in group attributes.
- (void)pickSeatWithUserId:(NSString *)userId seatIndex:(NSInteger)seatIndex {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = SeatInfoStatusUsed;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = self.userId;
```

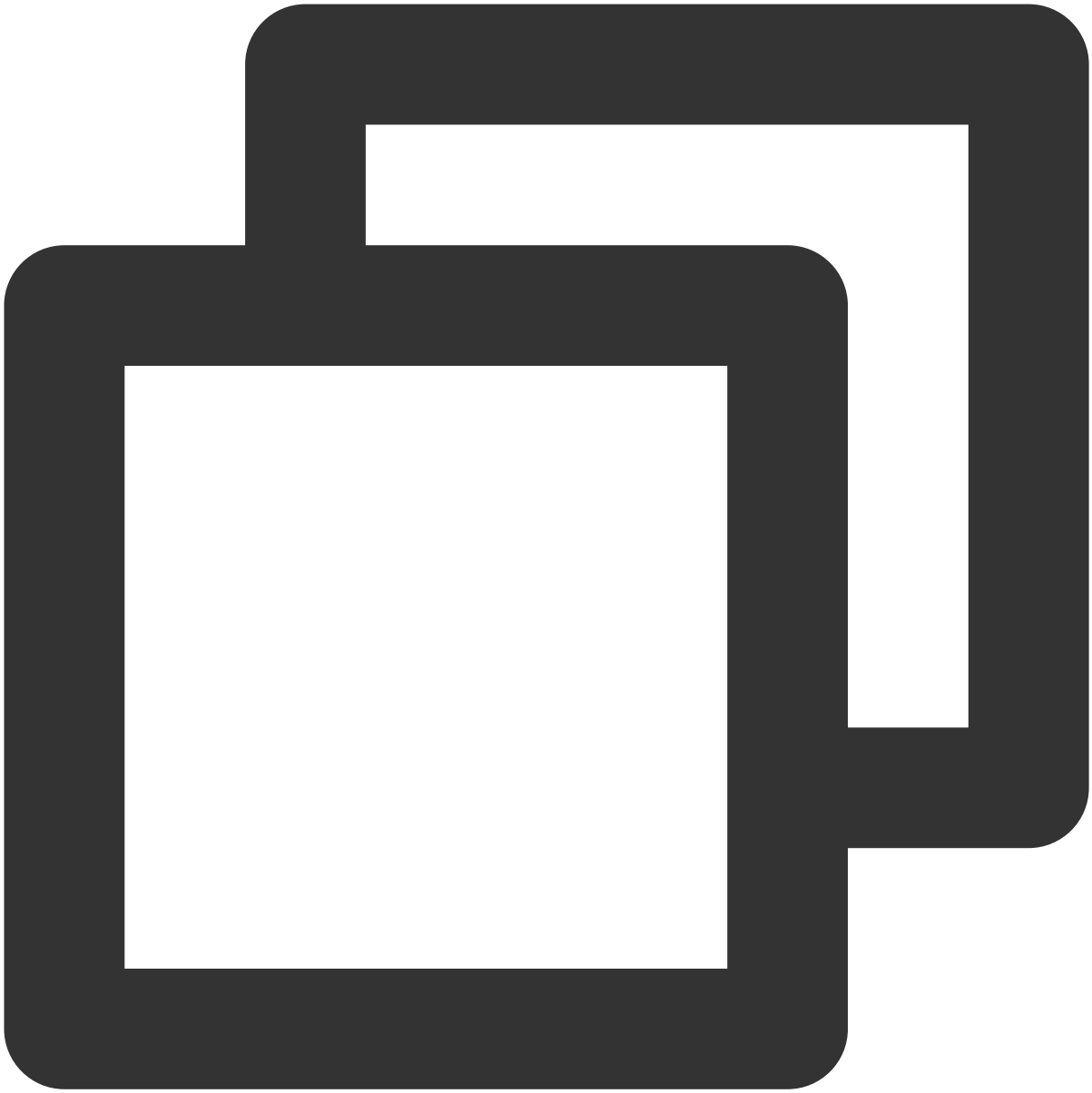
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes and it triggers onGroupAttributeC
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to become a speaker.
    }];
}

// Audience receives group attribute change callback. Audience starts streaming aft
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupAttributeChanged:(NSString *)groupID attributes:(NSMutableDictionary
    // Last locally saved full list of seats.
    NSArray *oldSeatArray = self.seatInfoArray;
    // The most recent full list of seats parsed from groupAttributeMap.
    NSArray *newSeatArray = [self getSeatListFromAttr:attributes seatSize:self.seat
    // Iterate through the full list of seats. Compare old and new seat information
    for (int i = 0; i < self.seatSize; i++) {
        SeatInfoModel *oldInfo = oldSeatArray[i];
        SeatInfoModel *newInfo = newSeatArray[i];
        if (oldInfo.status != newInfo.status && newInfo.status == SeatInfoStatusUse
            if ([newInfo.userId isEqualToString:self.userId]) {
                // Match own information successfully. Switch TRTC role and start s
                [self.trtcCloud switchRole:TRTCRoleAnchor];
                [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
            } else {
                // Update local seat list. Render local seat view.
            }
        }
    }
}
```

3. Become a listener.

Mic-connecting audiences can reset seat information by modifying group attributes. Other users will receive a group attribute change callback. Update local seat information.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

- (void)leaveSeatWithIndex:(NSInteger)seatIndex {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = SeatInfoStatusUnused;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = @"";
```

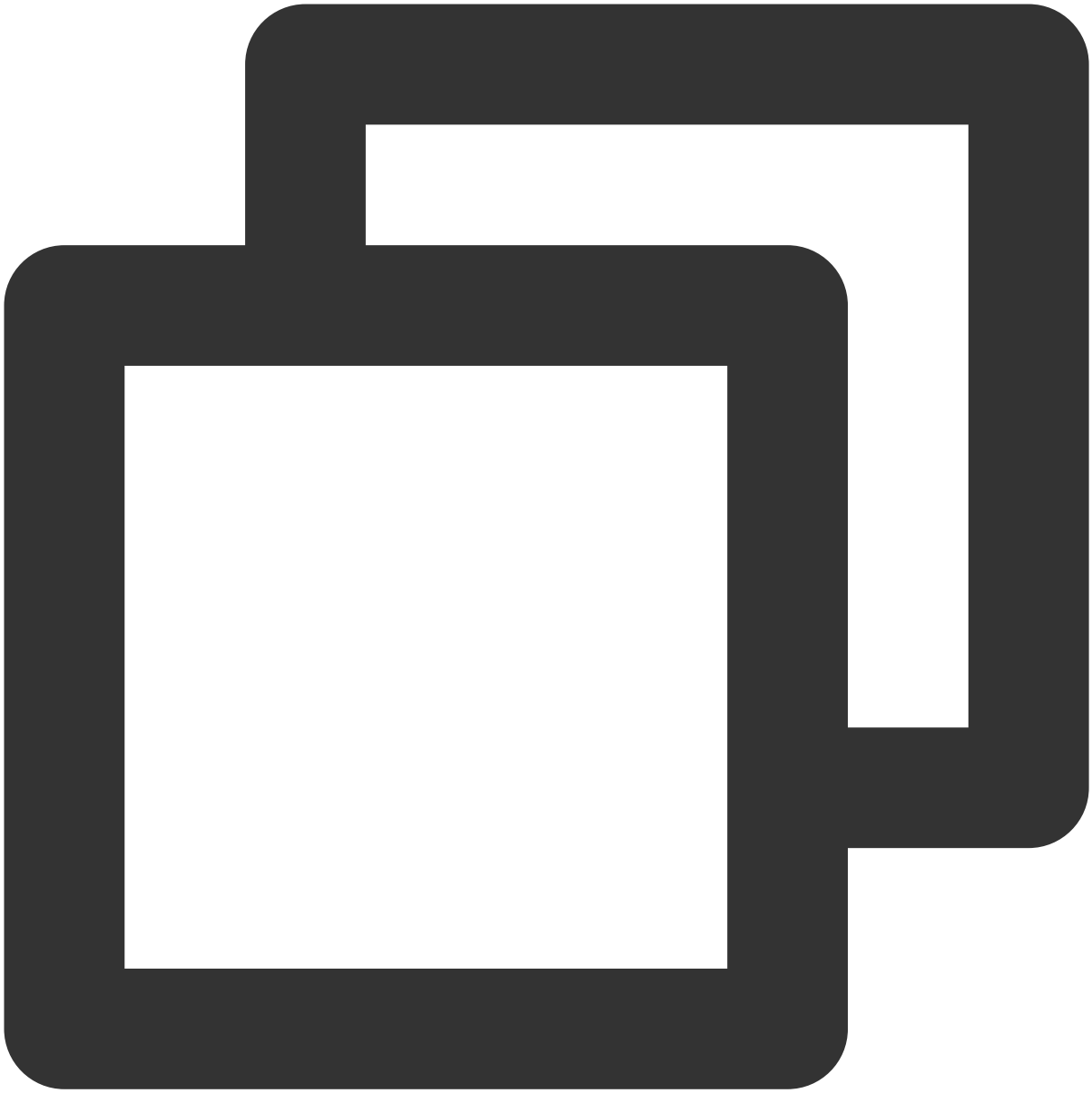
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes. Switch TRTC role and stop stream
        [self.trtcCloud switchRole:TRTCRoleAudience];
        [self.trtcCloud stopLocalAudio];
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to become a listener.
    }];
}
```

4. Remove a speaker.

Anchor removes a speaker. Directly modify the seat information saved in group attributes. Corresponding mic-connecting audience receives group attribute change callback. After successfully matching userId, they switch TRTC role and stop streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// Anchor calls this API to modify the seat information saved in group attributes.
- (void)kickSeatWithIndex:(NSInteger)seatIndex {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = SeatInfoStatusUnused;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = @"";
```

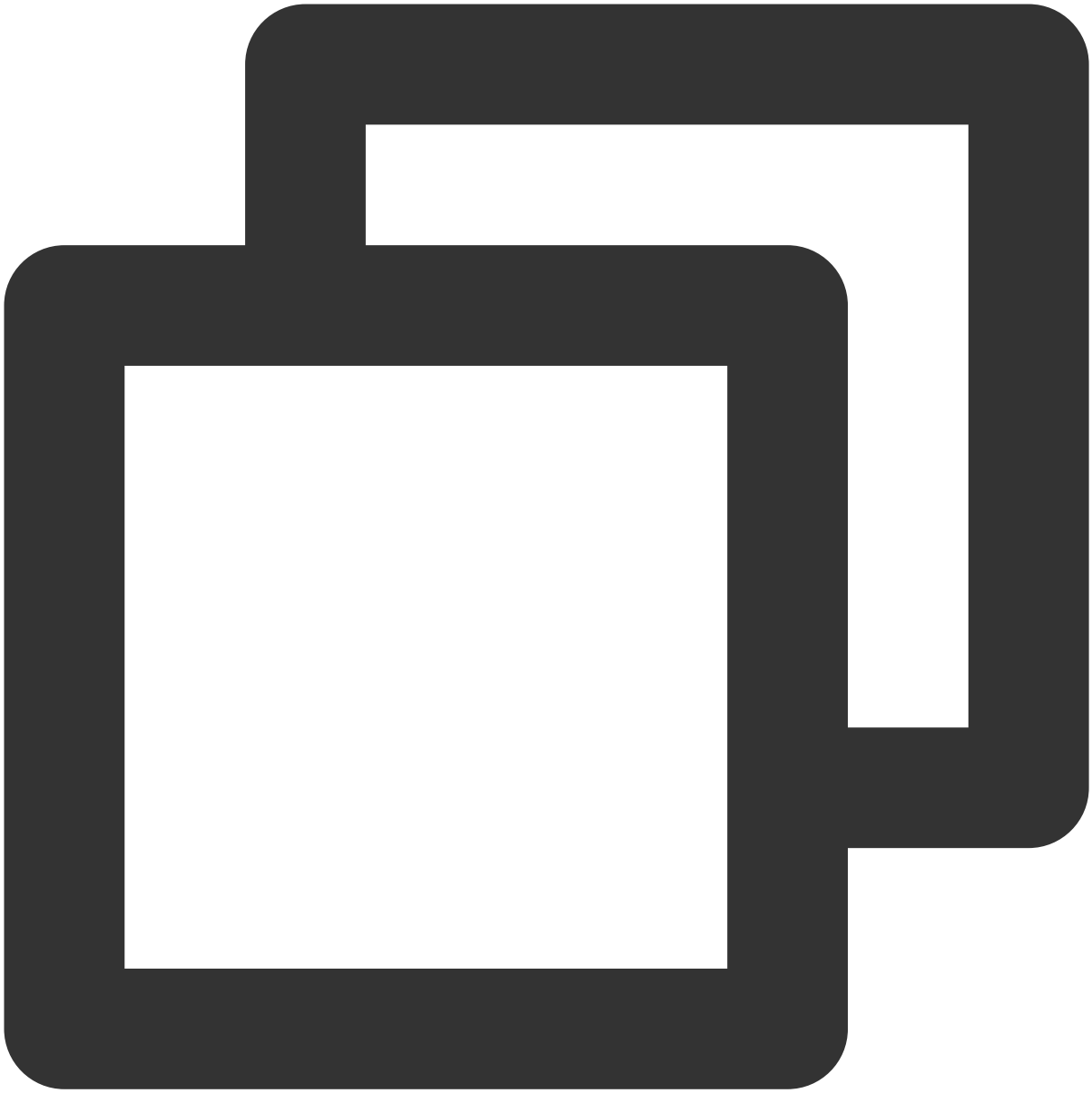
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes and it triggers onGroupAttributeC
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to remove the speaker.
    }];
}

// Mic-connecting audience receives group attribute change callback. It stops strea
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupAttributeChanged:(NSString *)groupID attributes:(NSMutableDictionary
    // Last locally saved full list of seats.
    NSArray *oldSeatArray = self.seatInfoArray;
    // The most recent full list of seats parsed from groupAttributeMap.
    NSArray *newSeatArray = [self getSeatListFromAttr:attributes seatSize:self.seat
    // Iterate through the full list of seats. Compare old and new seat information
    for (int i = 0; i < self.seatSize; i++) {
        SeatInfoModel *oldInfo = oldSeatArray[i];
        SeatInfoModel *newInfo = newSeatArray[i];
        if (oldInfo.status != newInfo.status && newInfo.status == SeatInfoStatusUnu
            if ([newInfo.userId isEqualToString:self.userId]) {
                // Match own information successfully. Switch TRTC role and stop st
                [self.trtcCloud switchRole:TRTCRoleAudience];
                [self.trtcCloud stopLocalAudio];
            } else {
                // Update local seat list. Render local seat view.
            }
        }
    }
}
```

5. Mute a seat.

Anchor mutes/unmutes a specific seat. Directly modify the seat information saved in group attributes. Corresponding mic-connecting audience receives group attribute change callback. After successfully matching userId, they pause/resume local streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// Anchor calls this API to modify the seat information saved in group attributes.
- (void)muteSeatWithIndex:(NSInteger)seatIndex mute:(BOOL)mute {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = localInfo.status;
    seatInfo.mute = mute;
    seatInfo.userId = localInfo.userId;
```

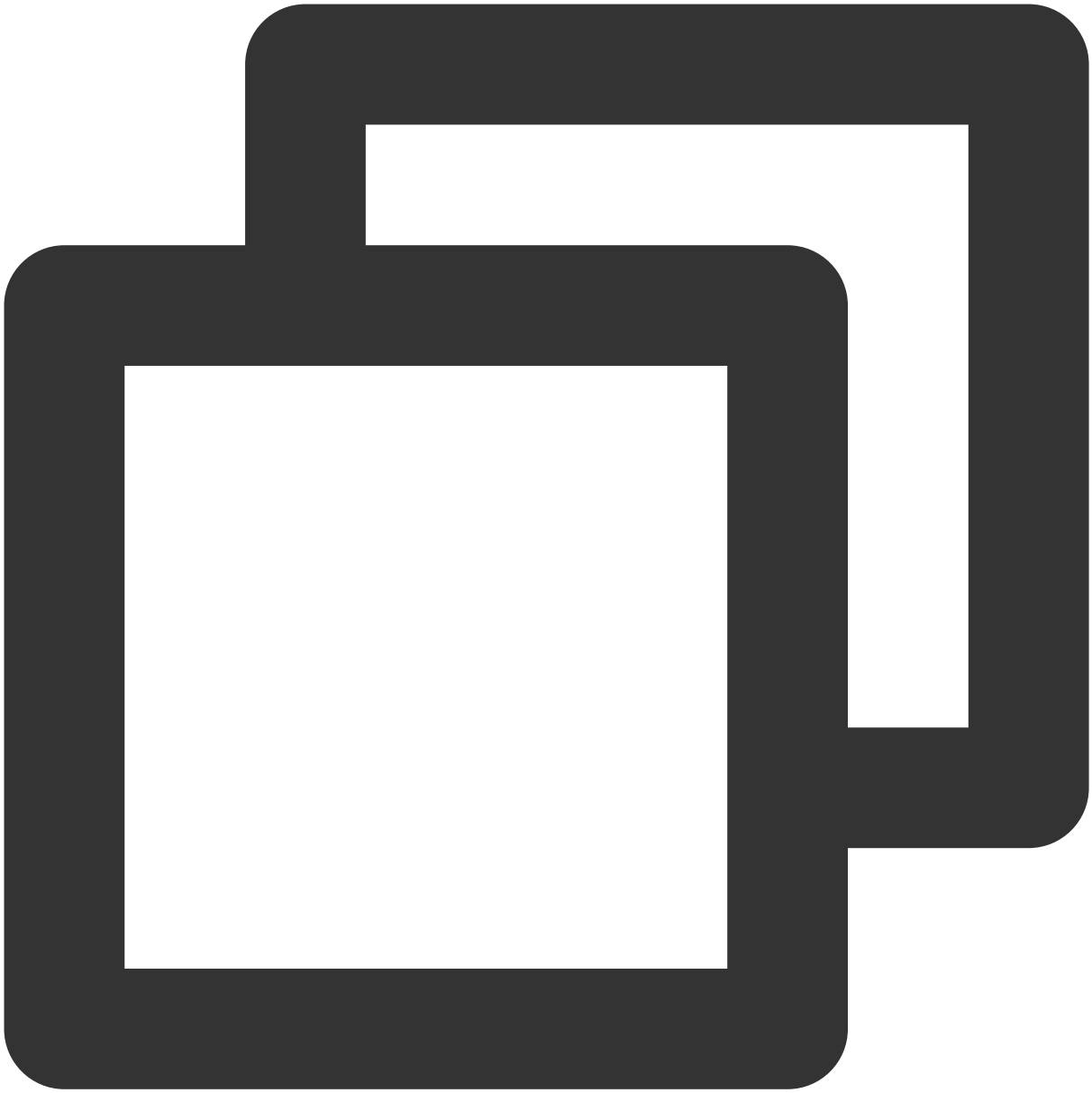
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes and it triggers onGroupAttributeC
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to mute the seat.
    }];
}

// The mic-connecting audience receives the group attribute change callback. The au
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupAttributeChanged:(NSString *)groupID attributes:(NSMutableDictionary
    // Last locally saved full list of seats.
    NSArray *oldSeatArray = self.seatInfoArray;
    // The most recent full list of seats parsed from groupAttributeMap.
    NSArray *newSeatArray = [self getSeatListFromAttr:attributes seatSize:self.seat
    // Iterate through the full list of seats. Compare old and new seat information
    for (int i = 0; i < self.seatSize; i++) {
        SeatInfoModel *oldInfo = oldSeatArray[i];
        SeatInfoModel *newInfo = newSeatArray[i];
        if (oldInfo.mute != newInfo.mute) {
            if ([newInfo.userId isEqualToString:self.userId]) {
                // Match own information successfully. Pause/resume local streaming
                [self.trtcCloud muteLocalAudio:newInfo.mute];
            } else {
                // Update local seat list. Render local seat view.
            }
        }
    }
}
```

6. Lock a seat.

Anchor locks/unlocks a seat by directly modifying the seat information saved in group attributes. Audience updates the corresponding seat view after receiving the group attribute change callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// Anchor calls this API to modify the seat information saved in group attributes.
- (void)lockSeatWithIndex:(NSInteger)seatIndex isLock:(BOOL)isLock {
    // Create a seat information instance. Store the modified seat information.
    SeatInfoModel *localInfo = self.seatInfoArray[seatIndex];
    SeatInfoModel *seatInfo = [[SeatInfoModel alloc] init];
    seatInfo.status = isLock? SeatInfoStatusLocked : SeatInfoStatusUnused;
    seatInfo.mute = localInfo.mute;
    seatInfo.userId = @"";
```

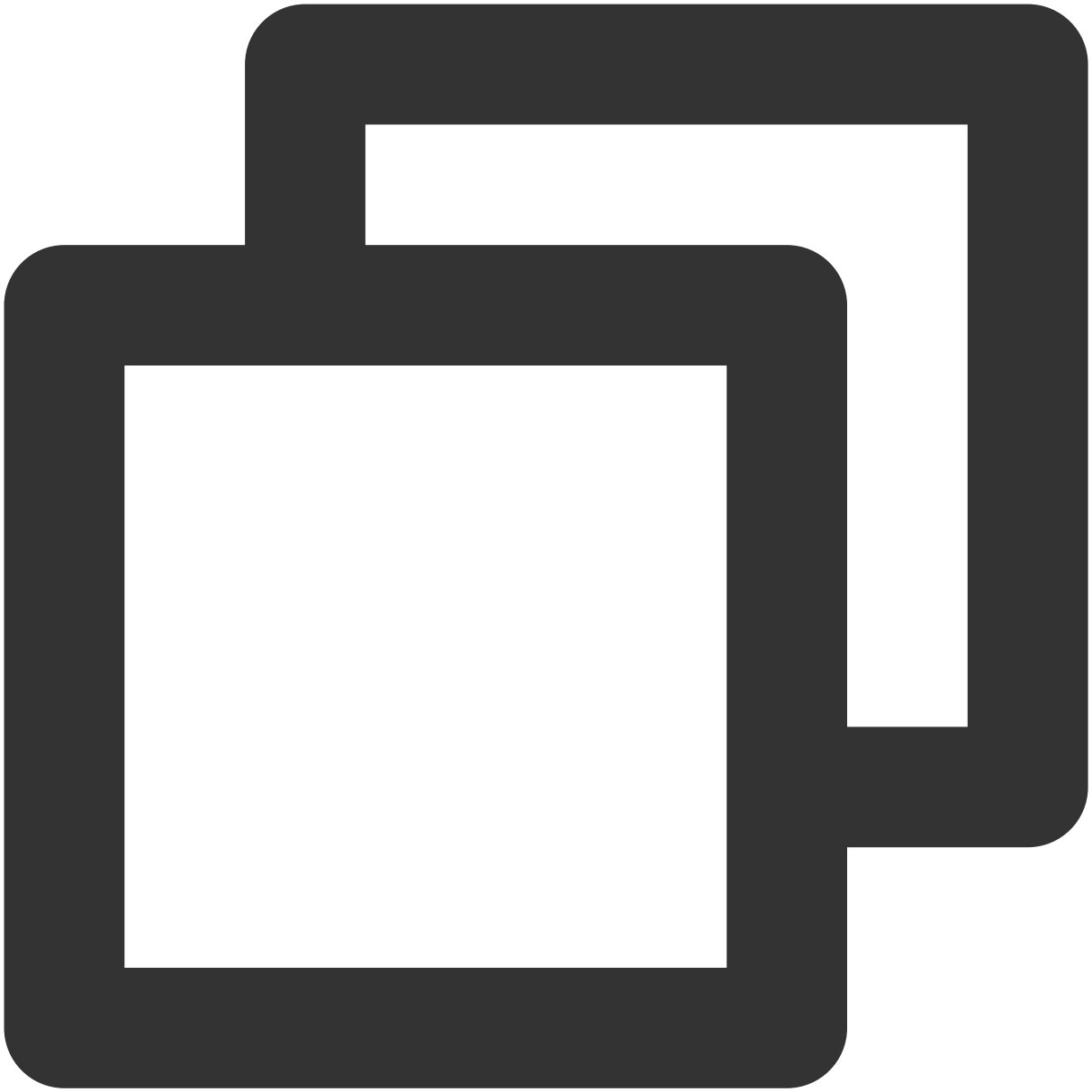
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Serialize the seat information object into JSON format.
    NSString *jsonStr = seatInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", seatIndex]: json

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Successfully modified group attributes and it triggers onGroupAttributeC
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to lock the seat.
    }];
}

// The audience receives the group attribute change callback. Update the correspond
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupAttributeChanged:(NSString *)groupID attributes:(NSMutableDictionary
    // Last locally saved full list of seats.
    NSArray *oldSeatArray = self.seatInfoArray;
    // The most recent full list of seats parsed from groupAttributeMap.
    NSArray *newSeatArray = [self getSeatListFromAttr:attributes seatSize:self.seat
    // Iterate through the full list of seats. Compare old and new seat information
    for (int i = 0; i < self.seatSize; i++) {
        SeatInfoModel *oldInfo = oldSeatArray[i];
        SeatInfoModel *newInfo = newSeatArray[i];
        if (oldInfo.status == SeatInfoStatusLocked && newInfo.status == SeatInfoSta
            // Unlock a seat.
        } else if (oldInfo.status != newInfo.status && newInfo.status == SeatInfoSt
            // Lock a seat.
        }
    }
}
```

7. Move a seat.

On-mic anchor moves a seat by necessarily and separately modifying the source and target seat information saved in group attributes. Audience updates the corresponding seat view after receiving the group attribute change callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Locally saved full list of seats.
@property (nonatomic, copy) NSArray<SeatInfoModel *> *seatInfoArray;

// On-mic anchor calls this API to modify the seat information saved in group attri
- (void)moveSeatToIndex:(NSInteger)dstIndex {
    // Obtain the source seat ID by userId.
    __block NSInteger srcIndex = -1;
    [self.seatInfoArray enumerateObjectsUsingBlock:^(SeatInfoModel * _Nonnull seatI
        if ([seatInfo.userId isEqualToString:self.userId]) {
            srcIndex = idx;
            *stop = YES;
```

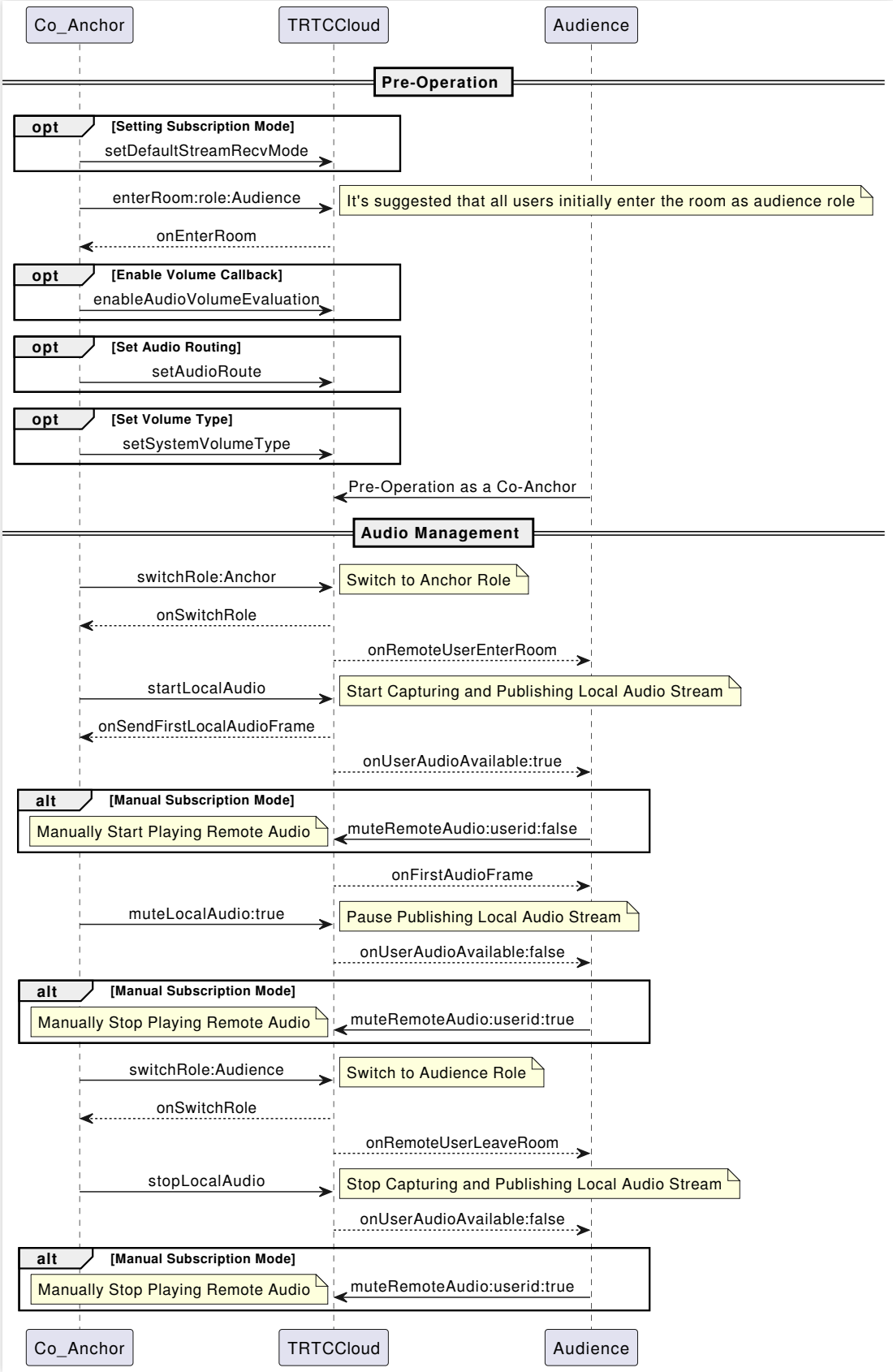
©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }];
    if (srcIndex < 0 || dstIndex < 0 || dstIndex >= self.seatInfoArray.count) {
        return;
    }

    // Obtain the corresponding seat information by its ID.
    SeatInfoModel *srcSeatInfo = self.seatInfoArray[srcIndex];
    SeatInfoModel *dstSeatInfo = self.seatInfoArray[dstIndex];

    // Create a seat information instance to store the modified source seat data.
    SeatInfoModel *srcChangeInfo = [[SeatInfoModel alloc] init];
    srcChangeInfo.status = SeatInfoStatusUnused;
    srcChangeInfo.mute = srcSeatInfo.mute;
    srcChangeInfo.userId = @"";

    // Create a seat information instance to store the modified target seat data.
    SeatInfoModel *dstChangeInfo = [[SeatInfoModel alloc] init];
    dstChangeInfo.status = SeatInfoStatusUsed;
    dstChangeInfo.mute = dstSeatInfo.mute;
    dstChangeInfo.userId = self.userId;

    // Serialize the seat information object into JSON format.
    NSString *srcJsonStr = srcChangeInfo.toJSONString;
    NSString *dstJsonStr = dstChangeInfo.toJSONString;
    NSDictionary *dict = @{ [NSString stringWithFormat:@"seat%ld", srcIndex]: srcJs
                            [NSString stringWithFormat:@"seat%ld", dstIndex]: dstJs
    };

    // Set group attributes. If the group attribute already exists, its value is up
    [[V2TIMManager sharedInstance] setGroupAttributes:self.groupId attributes:dict
        // Modify group attributes successfully. Move the seat successfully.
    } fail:^(int code, NSString *desc) {
        // Failed to modify group attributes. Failed to move the seat.
    }];
}
```

Step 6: Audio management.

Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.

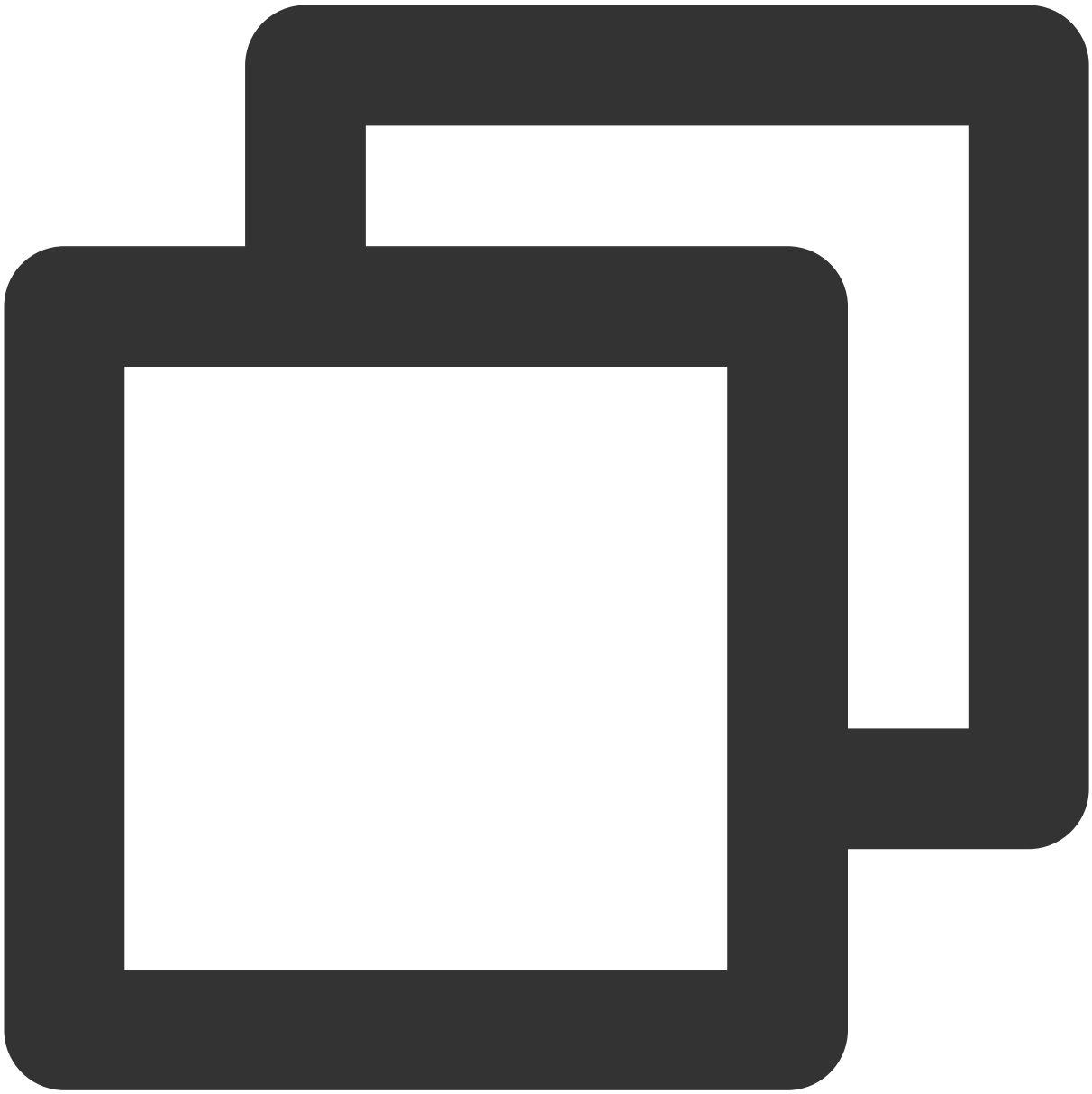


1. Subscription mode.

By default, the TRTC SDK uses an automatic subscription mode for audio streams. When users enter the room, the system will automatically play remote users' voice. If manual subscription to audio streams is needed, calling

©2013-2022 Tencent Cloud. All rights reserved.

`muteRemoteAudio(userId, mute)` to subscribe to and play remote users' audio streams is required.



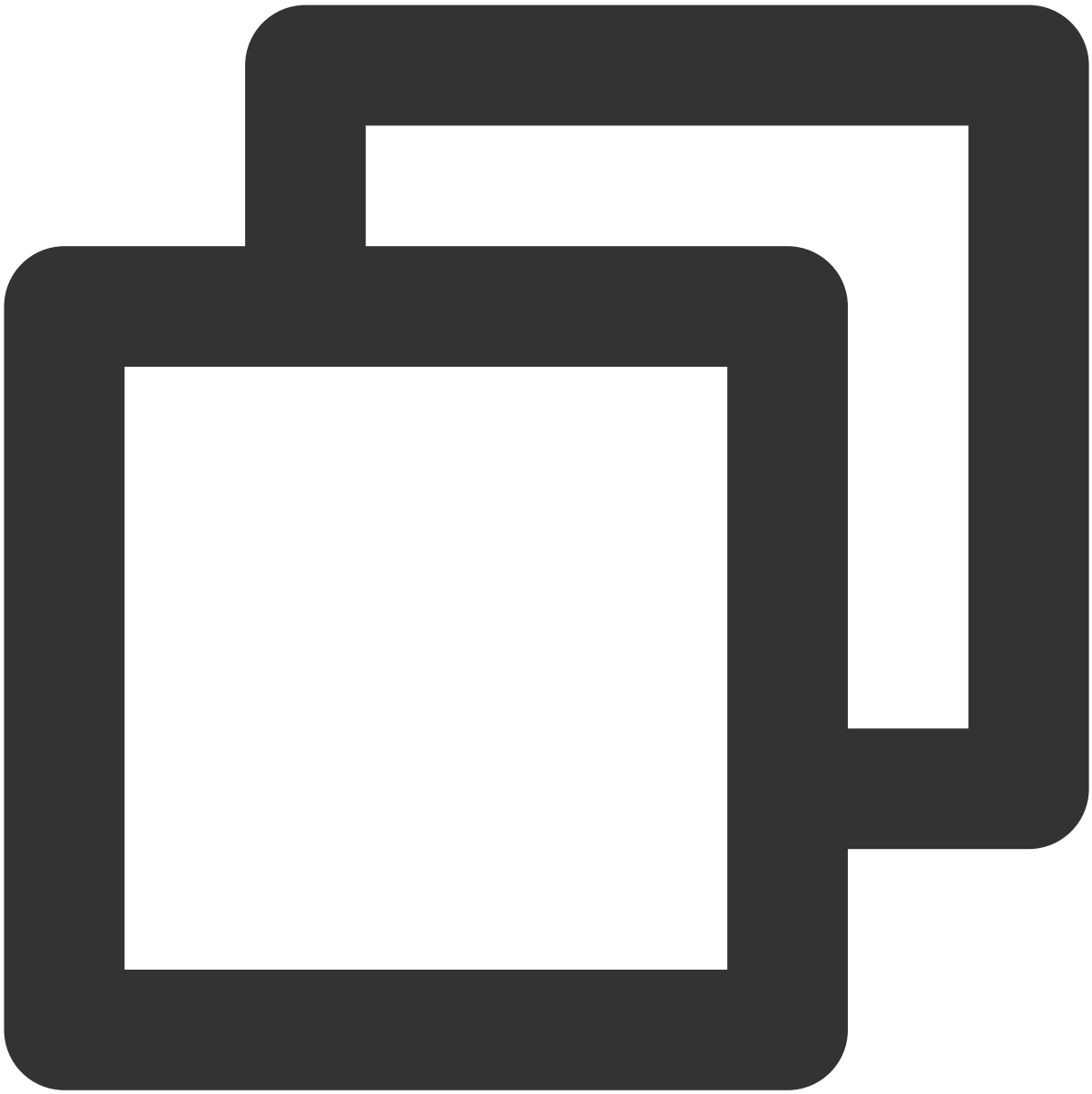
```
// Automatic subscription mode (default).
[self.trtcCloud setDefaultStreamRecvMode:YES video:YES];

// Manual subscription mode (custom).
[self.trtcCloud setDefaultStreamRecvMode:NO video:NO];
```

Note:

Set the subscription mode `setDefaultStreamRecvMode` before entering the room `enterRoom` to ensure to take effect.

©2013-2022 Tencent Cloud. All rights reserved.

2. Capture and publish.



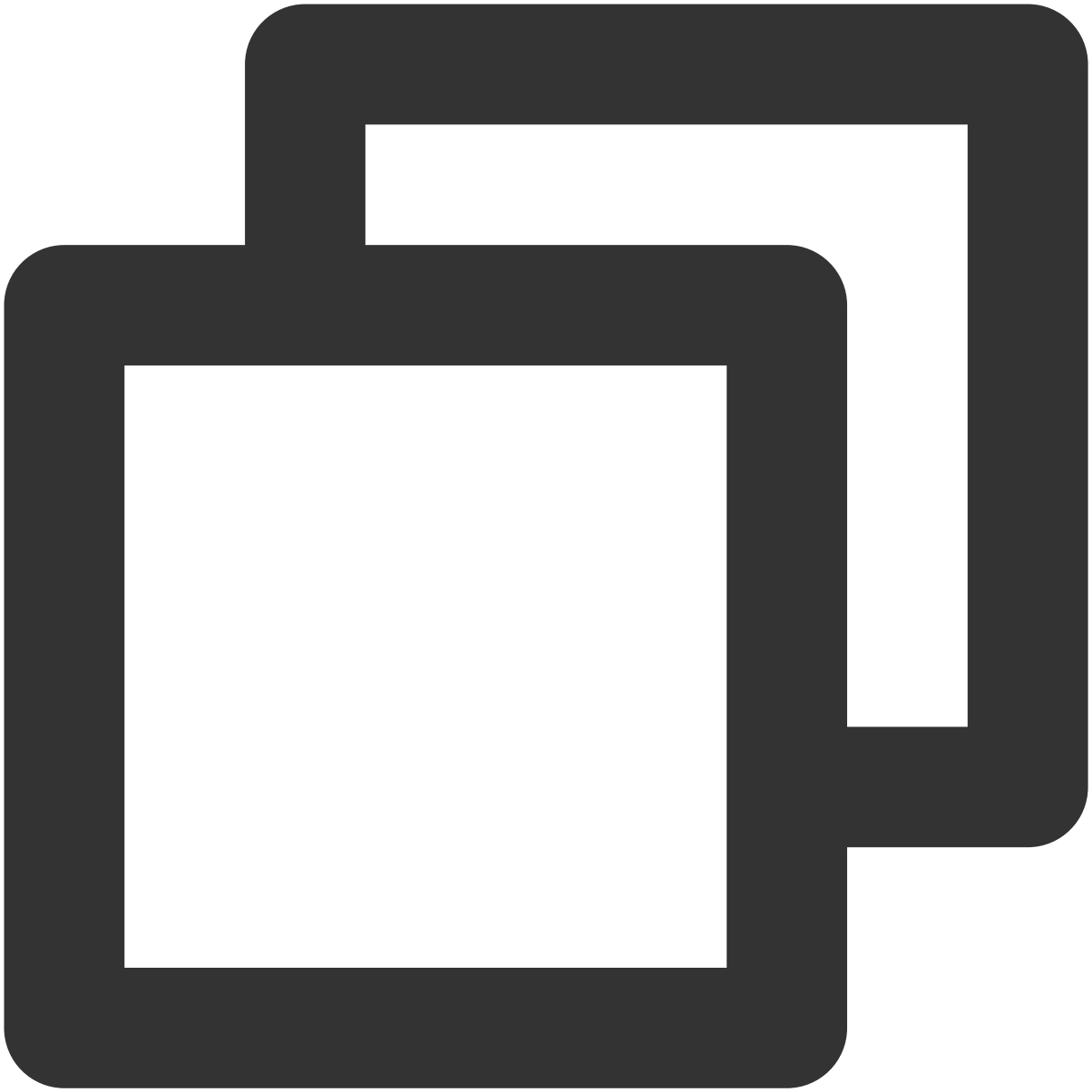
```
// Enable local audio capture and publishing.
[self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];

// Stop local audio capture and publishing.
[self.trtcCloud stopLocalAudio];
```

Note:

`startLocalAudio` requests mic permissions, and `stopLocalAudio` releases them.

3. Mute and unmute.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Pause publishing local audio streams (mute).
[self.trtcCloud muteLocalAudio:YES];
// Resume publishing local audio streams (unmute).
[self.trtcCloud muteLocalAudio:NO];

// Pause the subscription and playback of a specific remote user's audio streams.
[self.trtcCloud muteRemoteAudio:userId mute:YES];
// Resume the subscription and playback of a specific remote user's audio streams.
[self.trtcCloud muteRemoteAudio:userId mute:NO];

// Pause the subscription and playback of all remote users' audio streams.
```

©2013-2022 Tencent Cloud. All rights reserved.

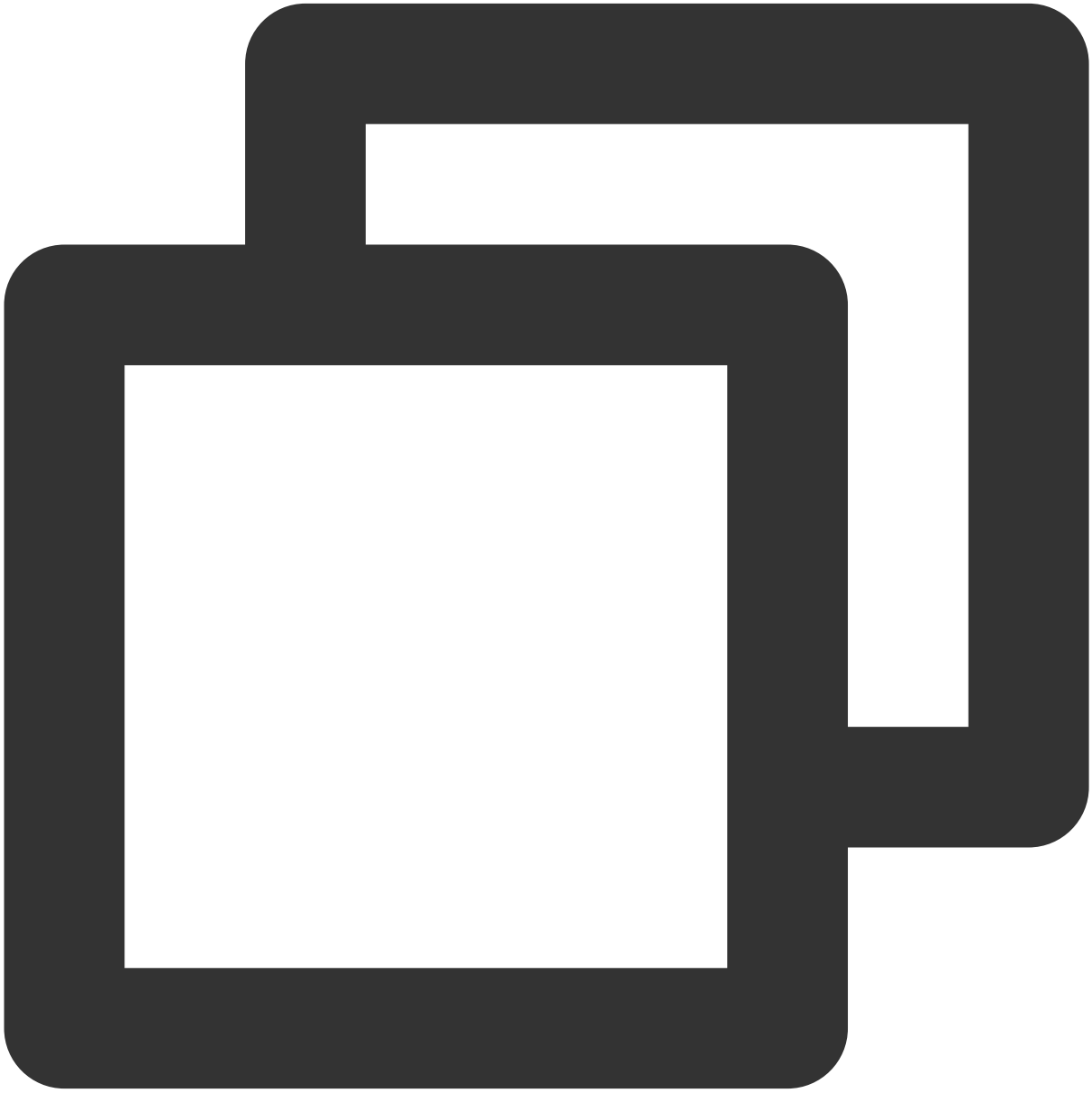
```
[self.trtcCloud muteAllRemoteAudio:YES];
// Resume the subscription and playback of all remote users' audio streams.
[self.trtcCloud muteAllRemoteAudio:NO];
```

Note:

In comparison, `muteLocalAudio` only requires a pause or release of the data stream at the software level, thus it is more efficient and smoother. And it is better suited for scenarios that require frequent muting and unmuting.

4. Audio Quality and Volume Type

Audio quality setting



```
// Set audio quality during local audio capture and publishing.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
[self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];

// Dynamically set audio quality during audio streaming.
[self.trtcCloud setAudioQuality:TRTCAudioQualityDefault];
```

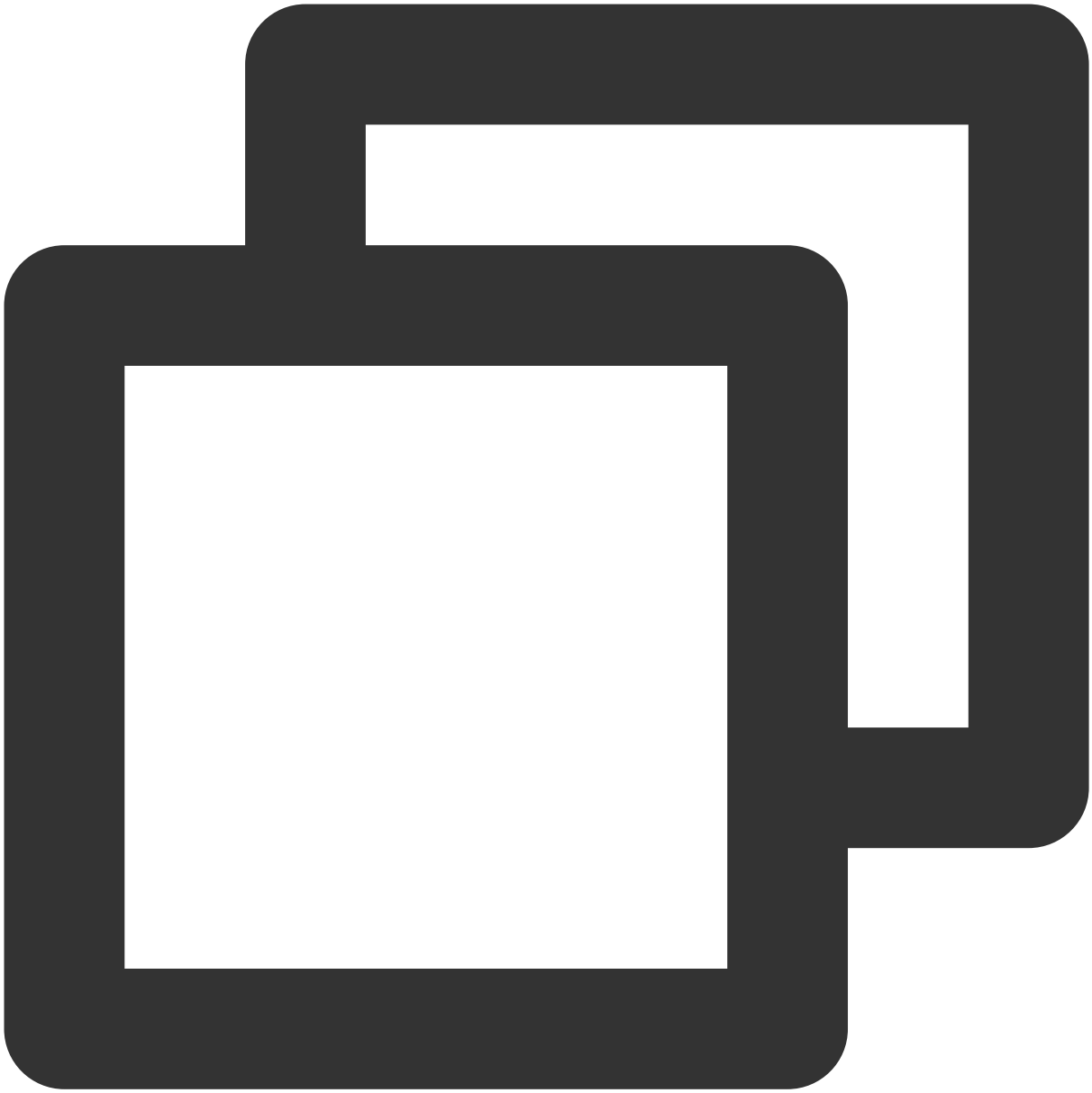
Note:

TRTC's preset audio quality is divided into three levels (Speech/Default/Music), each corresponding to different audio parameters. See TRTCAudioQuality for details.

Volume type setting.

Each TRTC audio quality level corresponds to a default volume type. If you need to forcibly specify a volume type, you can use the following API.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set volume type.
[self.trtcCloud setSystemVolumeType:TRTCSystemVolumeTypeAuto];
```

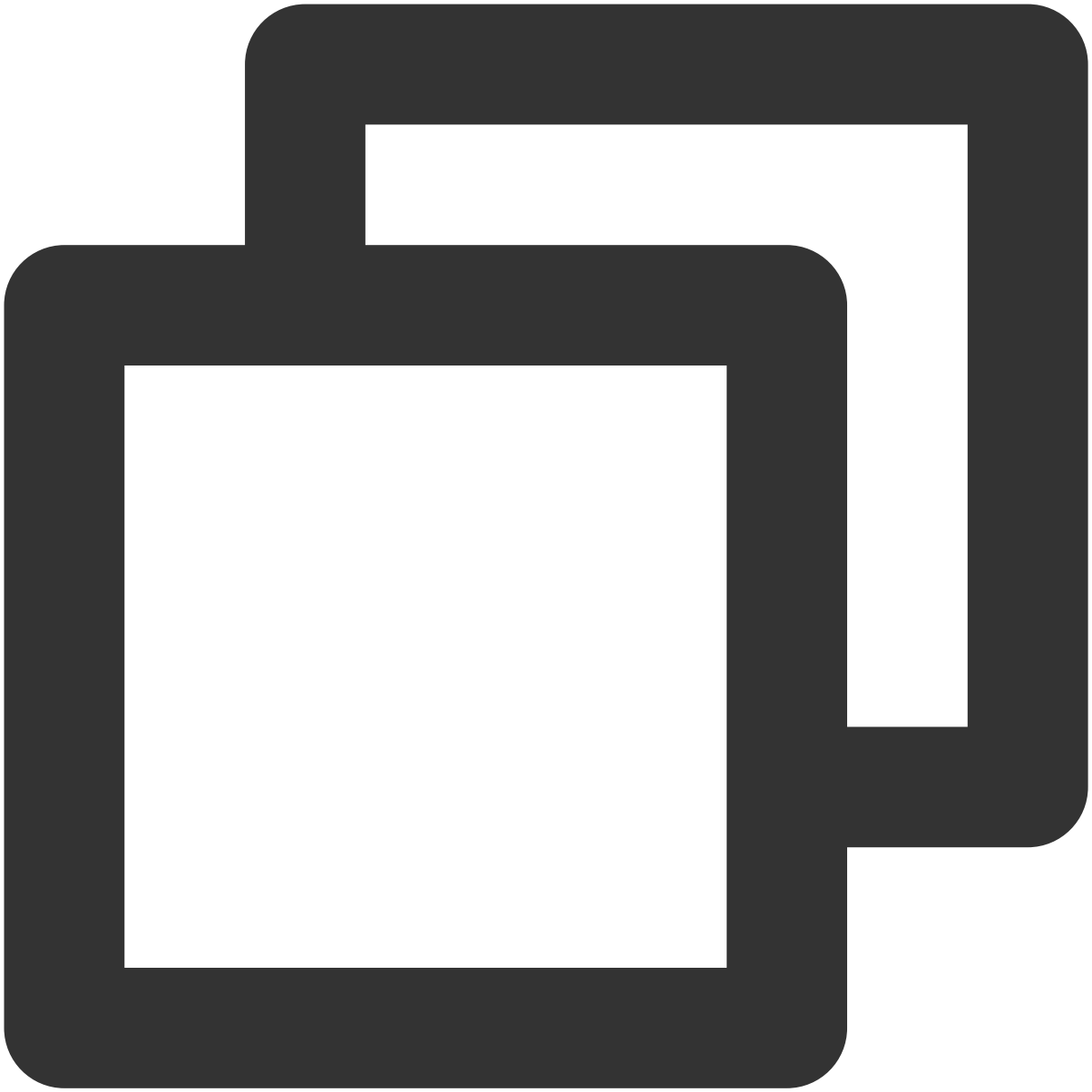
Note:

TRTC volume types are divided into three levels (VOIP/Auto/Media), each corresponding to different volume channels. See TRTCSystemVolumeType for details.

Audio routing setting.

Mobile devices such as smartphones usually have two playback locations: the speaker and the earpiece. If you need to forcibly specify the audio routing, you can use the following API.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set audio routing.
[self.trtcCloud setAudioRoute:TRTCAudioModeSpeakerphone];
```

Note:

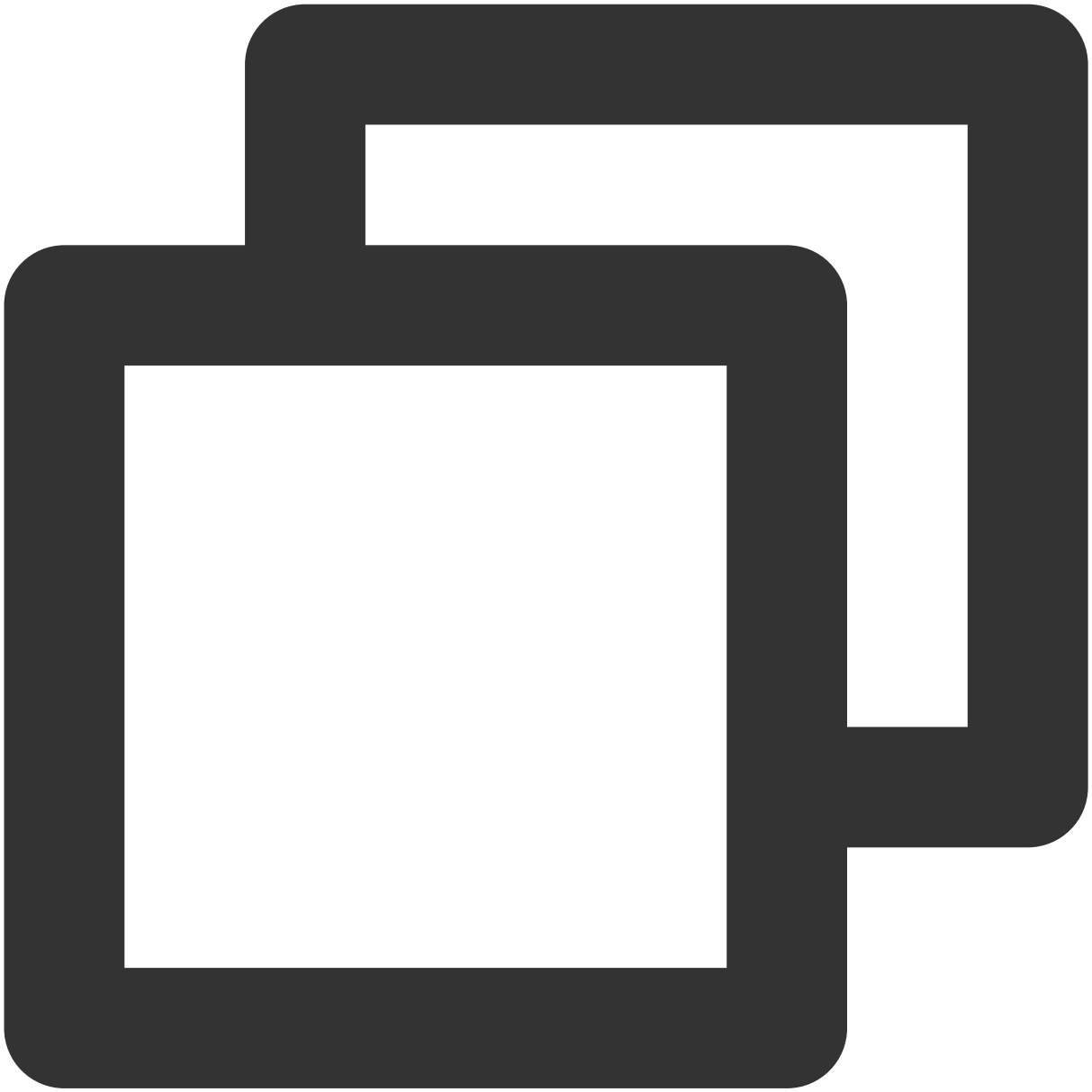
TRTC audio routing is divided into two types (Speaker/Earpiece), each corresponding to a different sound emission location. See TRTCAudioRoute for details.

Advanced Features

©2013-2022 Tencent Cloud. All rights reserved.

Bullet screen message interaction.

Voice chat live streaming rooms usually have text-based bullet screen message interactions. This can be achieved through the sending and receiving of group chat regular text messages via IM.



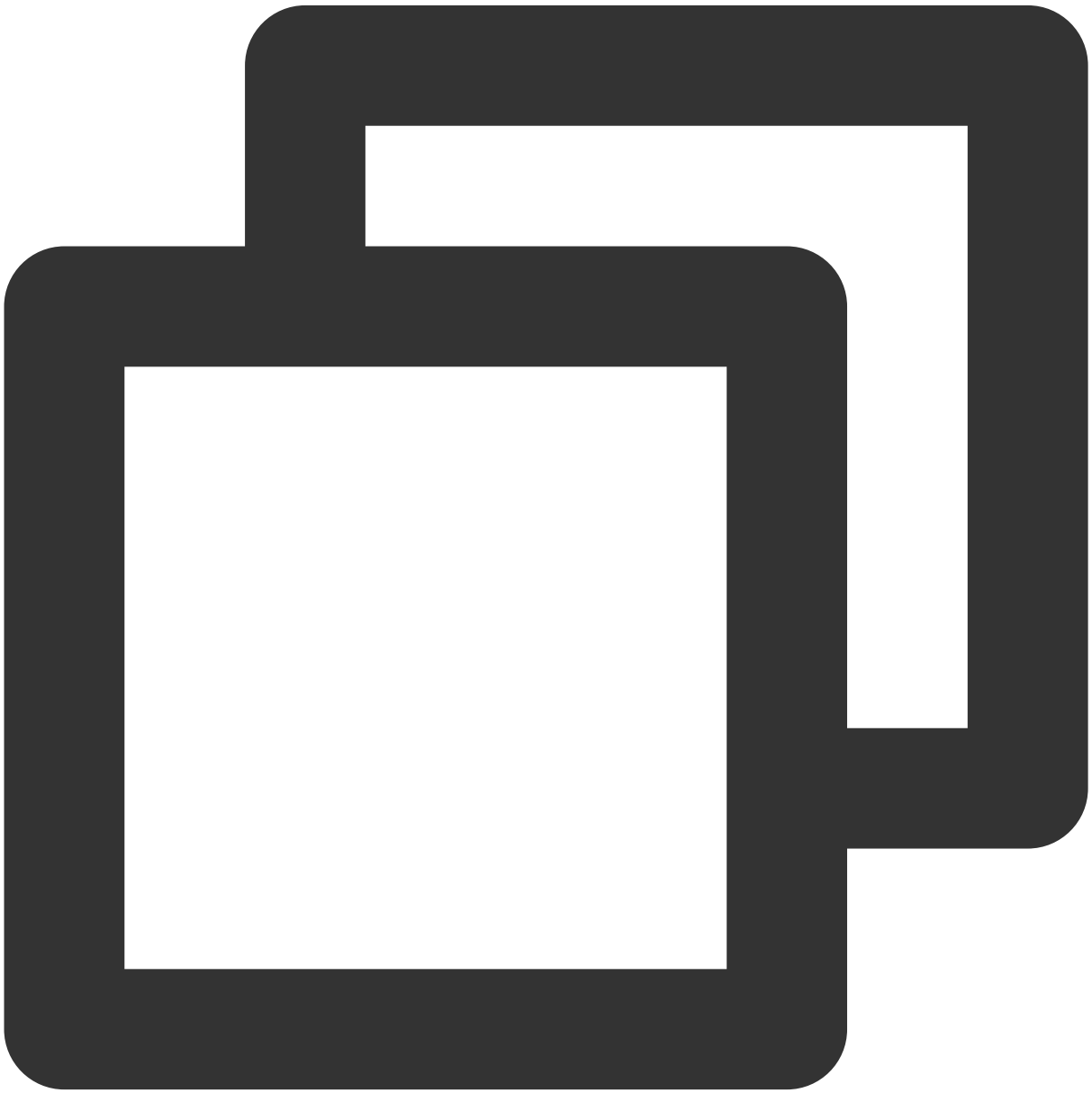
```
// Send public screen bullet screen messages.
[[V2TIMManager sharedInstance] sendGroupTextMessage:text to:groupID priority:V2TIM_
    // Successfully sent bullet screen messages.
} fail:^(int code, NSString *desc) {
    // Failed to send bullet screen messages.
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Receive public screen bullet screen messages.
[[V2TIMManager sharedInstance] addSimpleMsgListener:self];
- (void)onRecvGroupTextMessage:(NSString *)msgID groupID:(NSString *)groupID sender
    NSLog(@"%@: %@", info.nickName, text);
}
```

Volume level callback.

TRTC can callback the volume levels of the on-mic anchor at a fixed frequency. It is usually used to display sound waves and indicate the speaking anchor.



©2013-2022 Tencent Cloud. All rights reserved.

```
// Enable volume level callback. It is recommended to be enabled immediately after
// interval: Callback interval (ms). enable_vad: Whether to enable voice detection.
[self.trtcCloud enableAudioVolumeEvaluation:interval enable_vad:enable_vad];
self.trtcCloud.delegate = self;

- (void)onUserVoiceVolume:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSI
    // userVolumes is used to hold the volume levels of all speaking users, includi
    // totalVolume is used to report the maximum volume value among remote streamin
    ...
    // Adjust the corresponding visual representation of sound waves on the UI base
    ...
}
```

Note:

Voice detection only provides local voice detection results. The user's role must be an anchor to make it convenient to remind users to turn on their mics.

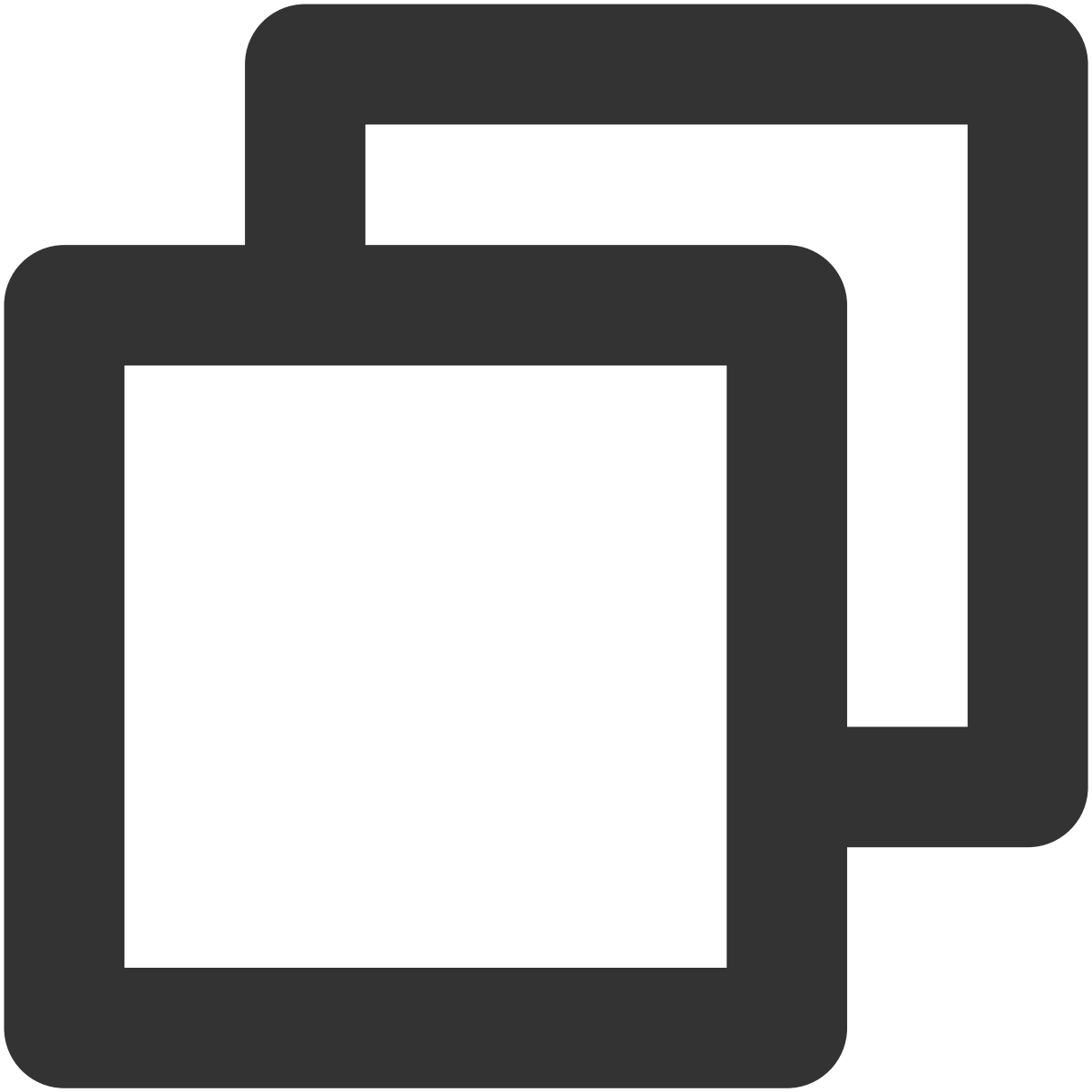
`userVolumes` is an array. For each element in the array, when userId is empty, it means the volume captured by the local mic; when userId is not empty, it means the volume of remote users.

Music and sound effect playback.

Playing background music and sound effects is a high-frequency demand in voice chat room scenarios. Below, we will explain the use of and precautions for commonly used background music APIs.

1. Start/stop/pause/resume playback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the management class for configuring background music, short sound effect
self.audioEffectManager = [self.trtcCloud getAudioEffectManager];

TXAudioMusicParam *param = [[TXAudioMusicParam alloc] init];
param.ID = musicID;
param.path = musicPath;
// Whether to publish the music to remote (otherwise play locally only).
param.publish = YES;
// Whether the playback is from a short sound effect file.
param.isShortFile = NO;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Start background music playback.
__weak typeof(self) weakSelf = self;
[self.audioEffectManager startPlayMusic:param onStart:^(NSInteger errCode) {
    __strong typeof(weakSelf) strongSelf = weakSelf;
    // Playback start callback.
    // -4001: Path opening failed.
    // -4002: Decoding failed.
    // -4003: Invalid URL address.
    // -4004: Playback not stopped.
    if (errCode < 0) {
        // Before replaying after playback failure, you must first stop the current
        [strongSelf.audioEffectManager stopPlayMusic:musicID];
    }
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // Playback progress callback.
    // progressMs: Current playback duration (milliseconds).
    // durationMs: Total duration of the current music (milliseconds).
} onComplete:^(NSInteger errCode) {
    // Playback end callback.
    // Playback failure due to weak network during playback will also throw this ca
    // Pausing or stopping playback midway will not trigger the onComplete callback
}];
// Stop background music playback.
[self.audioEffectManager stopPlayMusic:musicID];
// Pause background music playback.
[self.audioEffectManager pausePlayMusic:musicID];
// Resume background music playback.
[self.audioEffectManager resumePlayMusic:musicID];
```

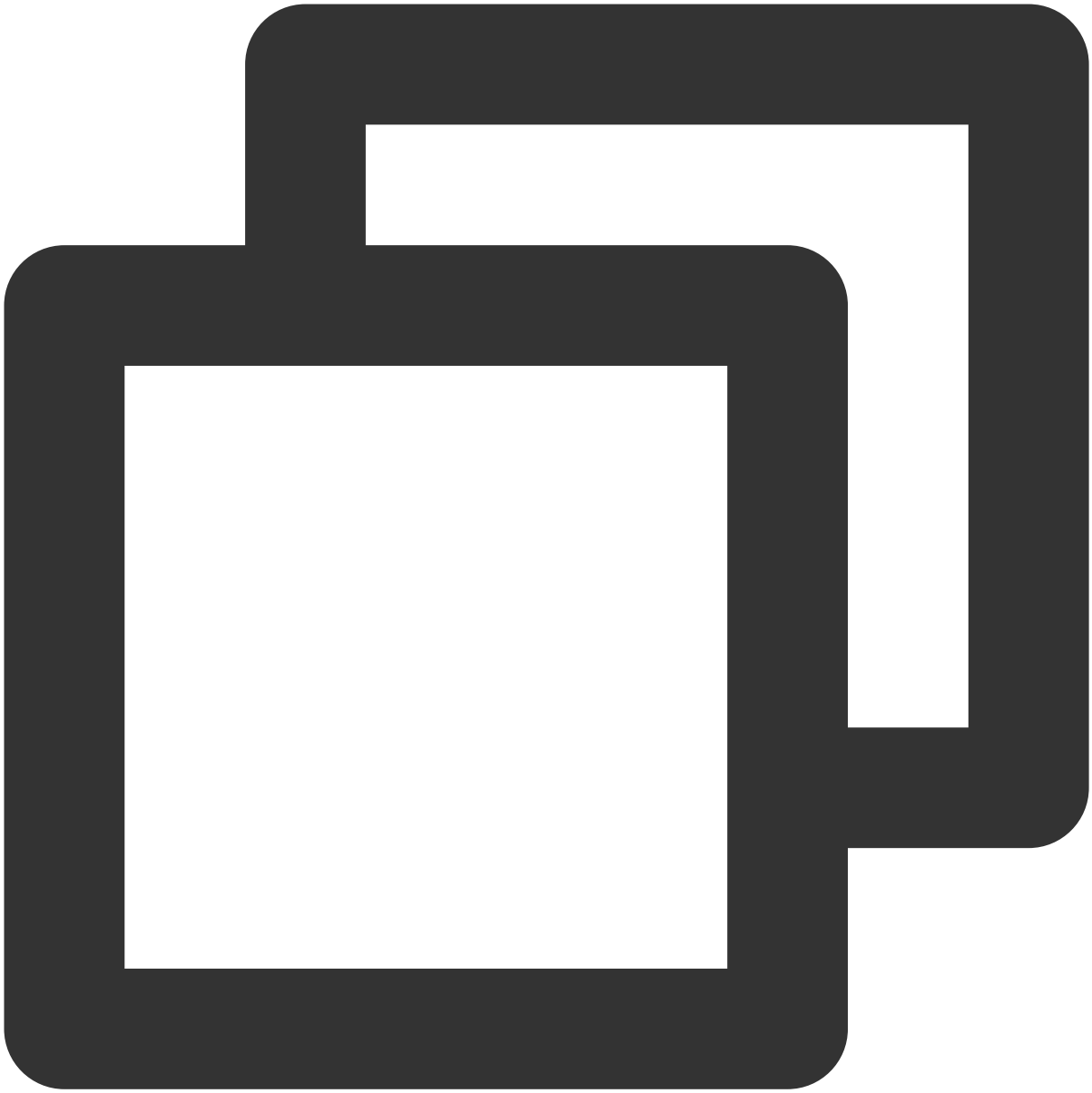
Note:

TRTC supports playing multiple pieces of music simultaneously, each identified uniquely by a musicID. If you want to play only one piece of music at a time, be sure to stop other music before starting playback, or you can use the same musicID to play different music. In this way, the SDK will stop the old music first, and then play the new one.

TRTC supports playing both local and online audio files, by passing in a local absolute path or URL address through `musicPath`. MP3/AAC/M4A/WAV formats are supported.

2. Adjust the ratio of music and voice volume.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the local playback volume of a piece of background music.
[self.audioEffectManager setMusicPlayoutVolume:musicID volume:volume];
// Set the remote playback volume of a specific background music.
[self.audioEffectManager setMusicPublishVolume:musicID volume:volume];
// Set the local and remote volume of all background music.
[self.audioEffectManager setAllMusicVolume:volume];
// Set the volume of voice capture.
[self.audioEffectManager setVoiceVolume:volume];
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Volume value's normal range is 0-100, with a default of 60 and a maximum setting of 150, but there is a risk of audio clipping.

If background music is overwhelming vocals, consider lowering the music playback volume and increasing the voice capture volume.

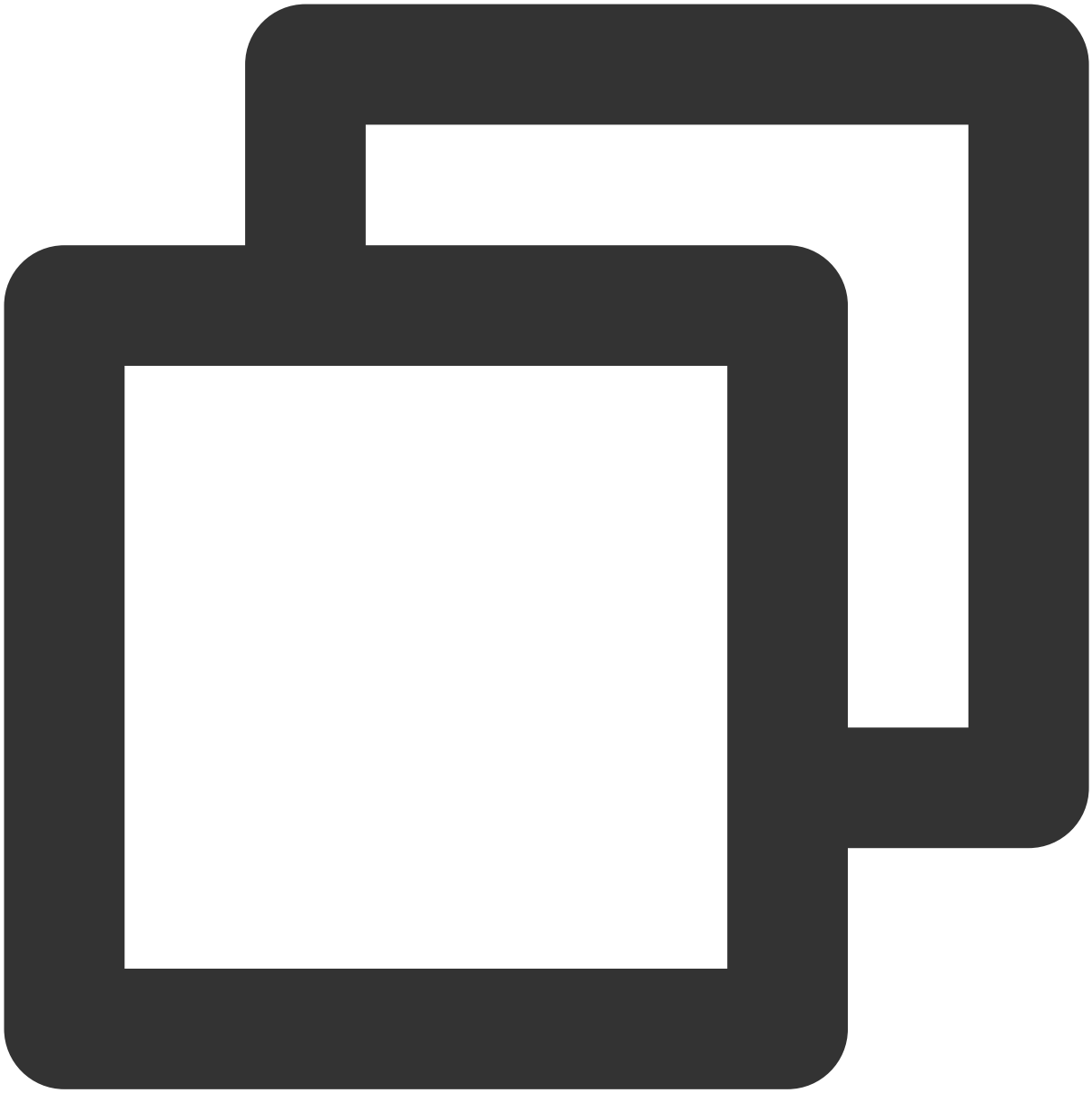
Mute the mic without muting background music: Use `setVoiceVolume(0)` to replace `muteLocalAudio(true)` .

3. Loop playback of background music and sound effects.

Solution 1: Use the `AudioMusicParam` 's `loopCount` parameter to set the number of loop playbacks.

The value range is from 0 to any positive integer. The default value is 0. 0 means play the music once; 1 means play the music twice; and so on.

©2013-2022 Tencent Cloud. All rights reserved.



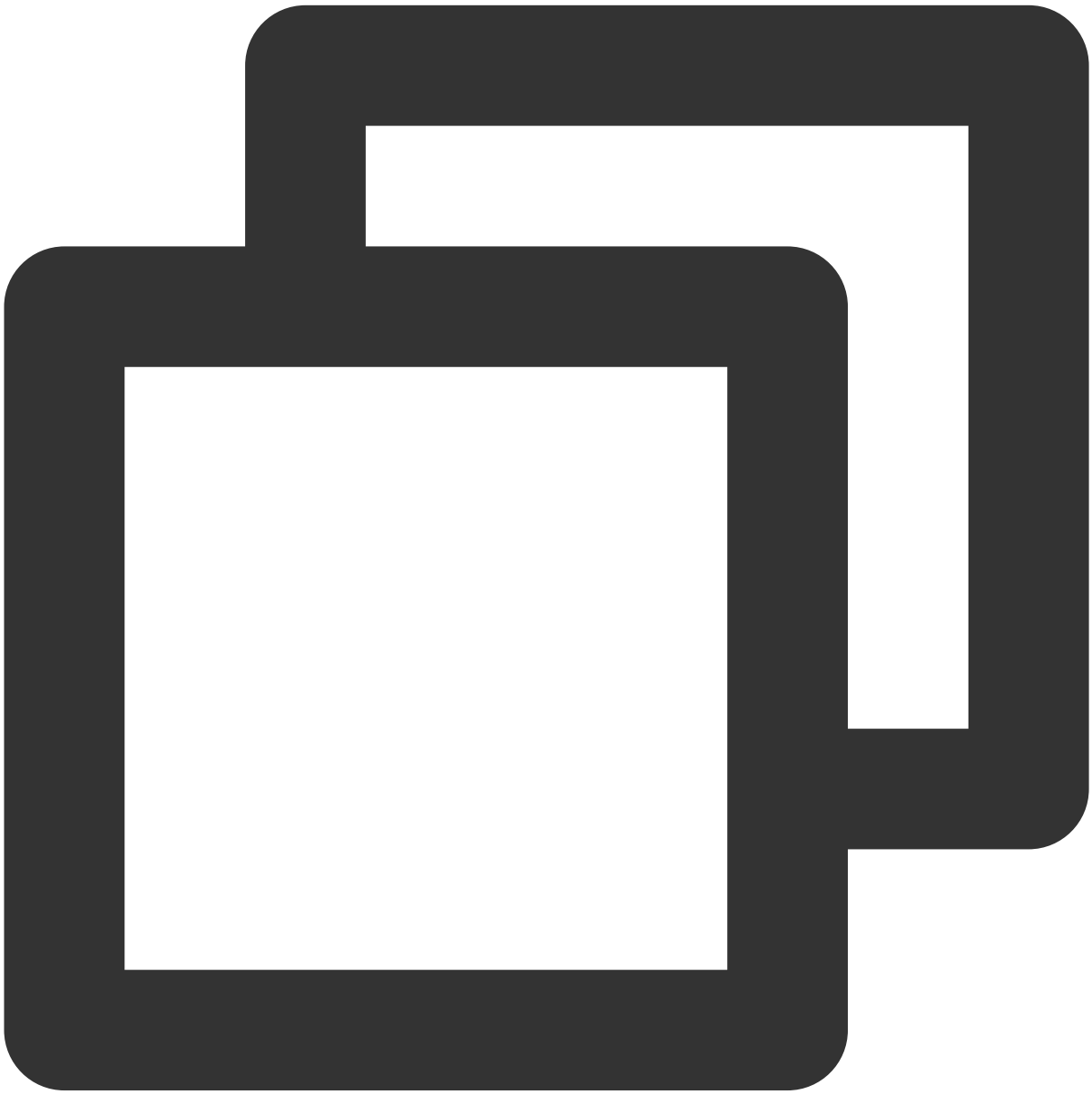
```
- (void)startPlayMusicWithId:(int32_t)musicId path:(NSString *)path loopCount:(NSIn
    TXAudioMusicParam *param = [[TXAudioMusicParam alloc] init];
    param.ID = musicId;
    param.path = path;
    param.publish = YES;
    // Whether the playback is from a short sound effect file.
    param.isShortFile = YES;
    // Set the number of loop playbacks. Negative number means an infinite loop.
    param.loopCount = loopCount < 0 ? NSIntegerMax : loopCount;
    [self.audioEffectManager startPlayMusic:param onStart:nil onProgress:nil onComp
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

Solution 1 will not trigger the `onComplete` callback after each loop playback. It will only be triggered after all the set loop counts have been played.

Solution 2: Implement loop playback through the "Background music has finished playing" event callback `onComplete`. It is usually used for list loop or single track loop.



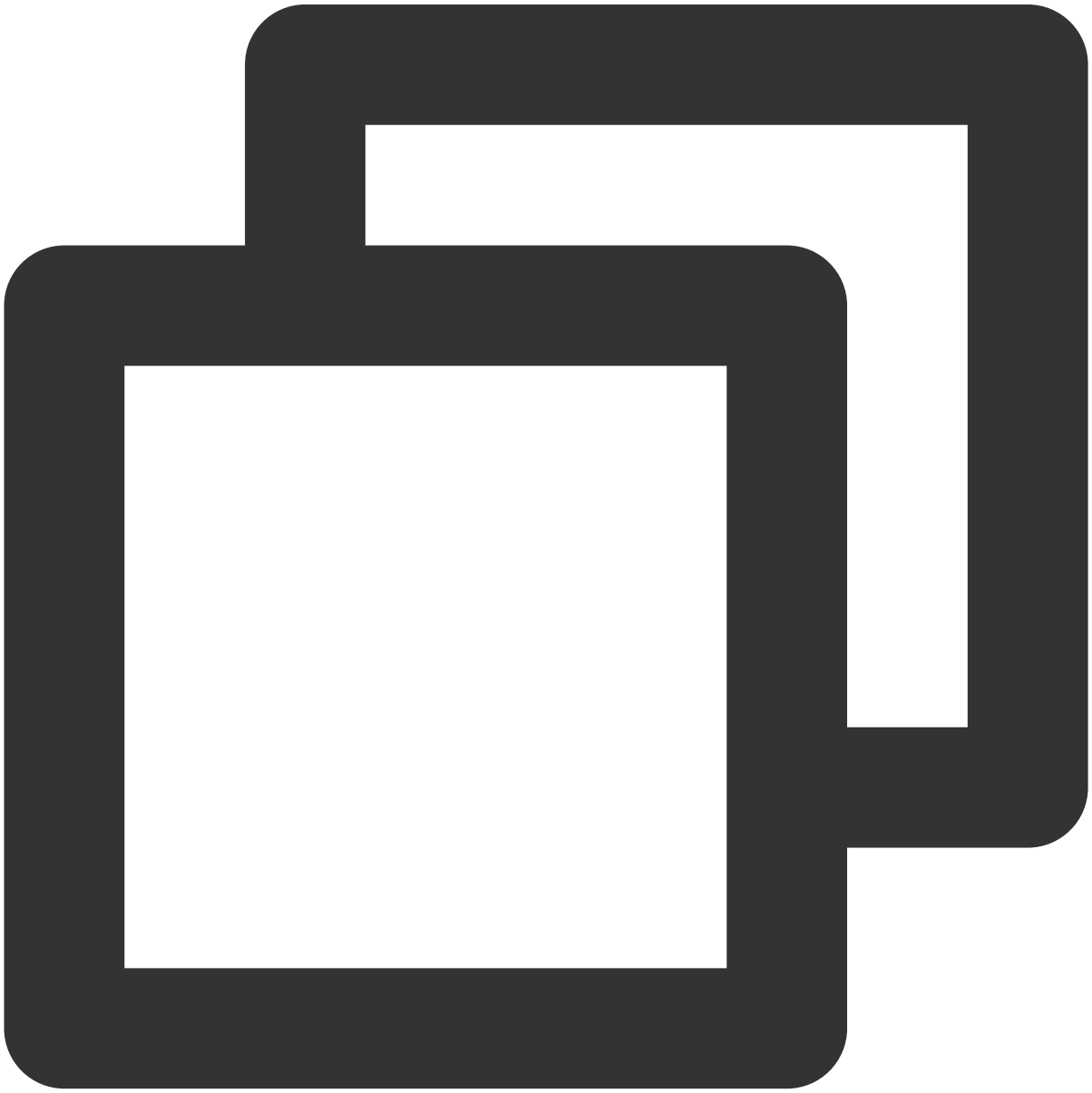
```
- (void)repeatPlayMusicWithParam:(TXAudioMusicParam *)param {
    __weak typeof(self) weakSelf = self;
    [self.audioEffectManager startPlayMusic:param onStart:nil onProgress:nil onComp
        __strong typeof(weakSelf) strongSelf = weakSelf;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Here you can re-call the playback API to achieve loop playback of the mu
        if (errCode >= 0) {
            [strongSelf repeatPlayMusicWithParam:param];
        }
    }];
}
```

Mixed stream relay and push back.

1. Live streaming CDN with mixed stream relay.

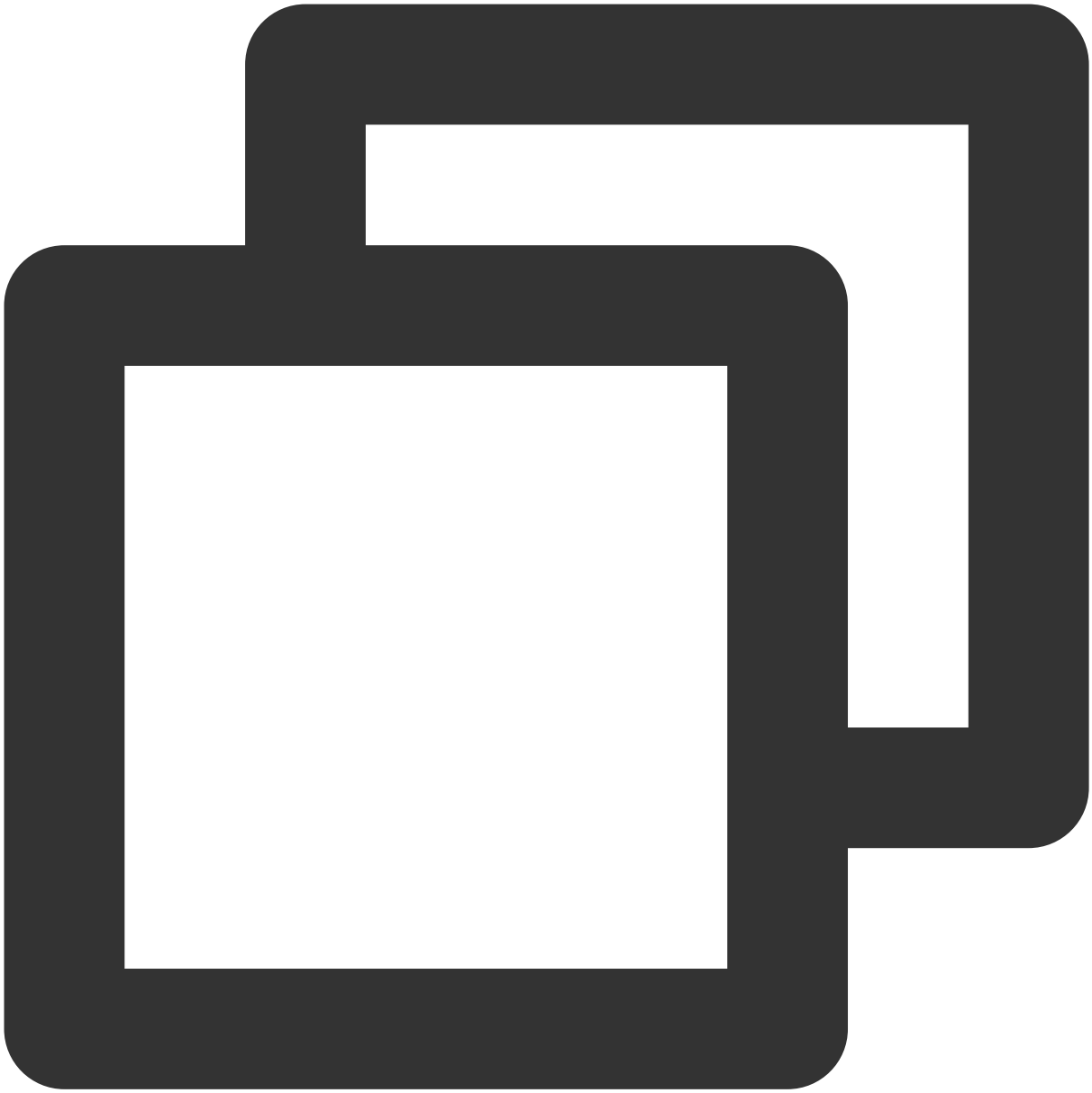


©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)startPublishMediaToCDN:(NSString *)streamName {
    NSDate *date = [NSDate dateWithTimeIntervalSinceNow:0];
    // Expiration time of the streaming URL. By default, it is one day.
    NSTimeInterval time = [date timeIntervalSince1970] + (24 * 60 * 60);
    // LIVE_URL_KEY authentication key. Obtain from the streaming URL configuration
    NSString *secretParam = [self getSafeUrl:LIVE_URL_KEY streamName:streamName tim

    // The target URLs for media stream publication.
    TRTCPublishTarget *target = [[TRTCPublishTarget alloc] init];
    // Publish to CDN after mixing.
    target.mode = TRTCPublishMixStreamToCdn;
    TRTCPublishCdnUrl* cdnUrl = [[TRTCPublishCdnUrl alloc] init];
    // Streaming URL must include parameters. Otherwise, streaming fails.
    cdnUrl.rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@?%@", PUSH_DOMAI
    // True means Tencent CSS push URLs, and false means third-party services.
    cdnUrl.isInternalLine = YES;
    NSMutableArray* cdnUrlList = [NSMutableArray new];
    // Multiple CDN push URLs can be added.
    [cdnUrlList addObject:cdnUrl];
    target.cdnUrlList = cdnUrlList;

    TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
    // Set the encoding parameters of the transcoded audio stream (can be customize
    encoderParam.audioEncodedSampleRate = 48000;
    encoderParam.audioEncodedChannelNum = 1;
    encoderParam.audioEncodedKbps = 50;
    encoderParam.audioEncodedCodecType = 0;

    // Set the encoding parameters of the transcoded video stream (must be filled i
    encoderParam.videoEncodedWidth = 64;
    encoderParam.videoEncodedHeight = 64;
    encoderParam.videoEncodedFPS = 15;
    encoderParam.videoEncodedGOP = 3;
    encoderParam.videoEncodedKbps = 30;

    // Configuration parameters for media stream transcoding.
    TRTCStreamMixingConfig *config = [[TRTCStreamMixingConfig alloc] init];
    // By default, leave this field empty. It indicates that all audio in the room
    config.audioMixUserList = nil;
    // Must have TRTCVideoLayout parameters if mixing black frames (can be ignored
    TRTCVideoLayout *layout = [[TRTCVideoLayout alloc] init];
    config.videoLayoutList = @[layout];

    // Start mixing and relaying mixed streams.
    [self.trtcCloud startPublishMediaStream:target encoderParam:encoderParam mixing
}
```

©2013-2022 Tencent Cloud. All rights reserved.

2. Push the mixed stream back to the TRTC room.



```
- (void)startPublishMediaToRoom:(NSString *)roomId userID:(NSString *)userId {
    // The target URLs for media stream publication.
    TRTCPublishTarget *target = [[TRTCPublishTarget alloc] init];
    // After mixing, the stream is relayed back to the room.
    target.mode = TRTCPublishMixStreamToRoom;
    target.mixStreamIdentity.strRoomId = roomId;
    // Mixed stream robot's userid, must not duplicate with other users' userid in
    target.mixStreamIdentity.userId = [NSString stringWithFormat:@"%@%@", userId, M
```

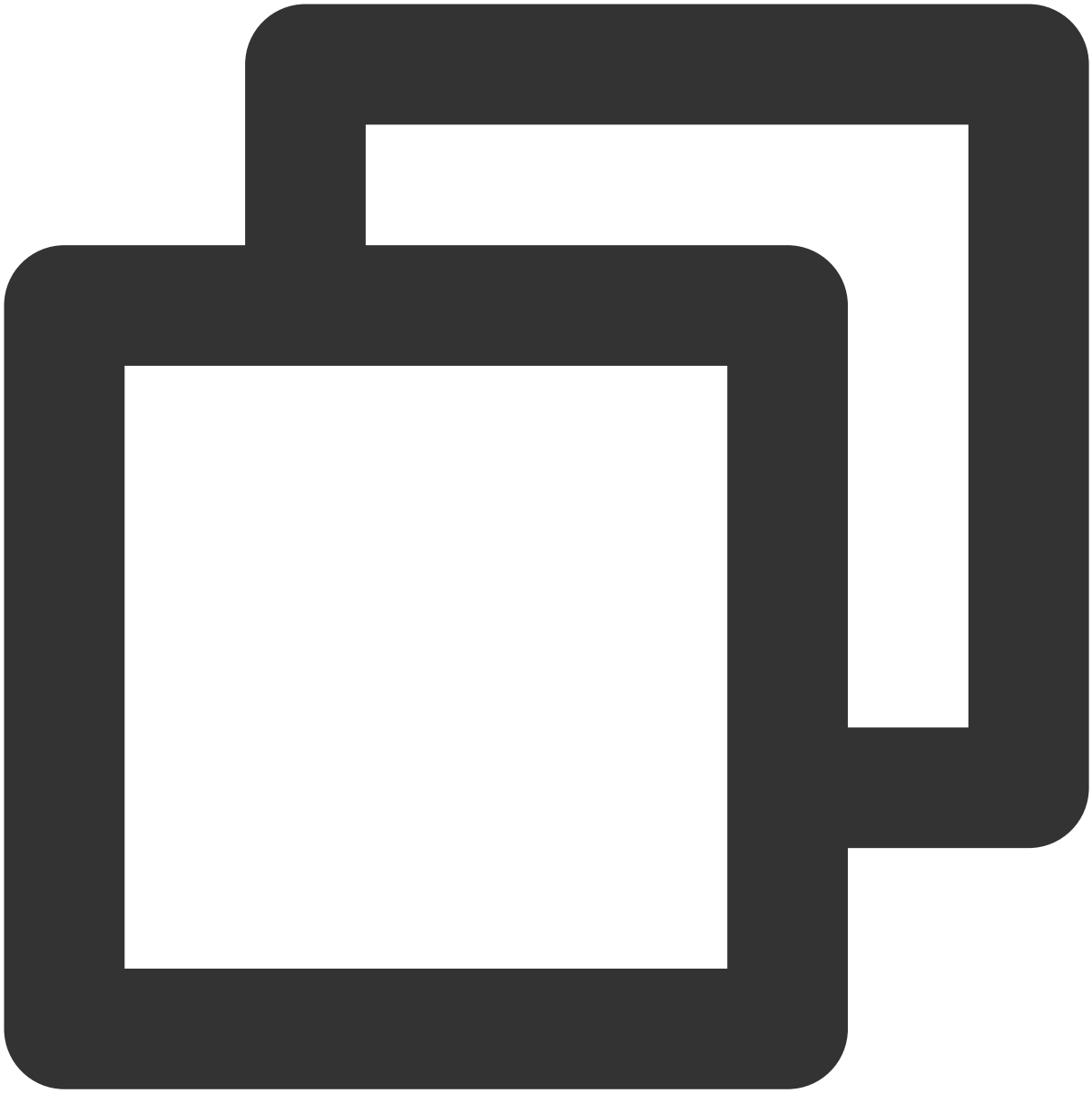
©2013-2022 Tencent Cloud. All rights reserved.

```
    TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
    // Set the encoding parameters of the transcoded audio stream (can be customize
    encoderParam.audioEncodedSampleRate = 48000;
    encoderParam.audioEncodedChannelNum = 2;
    encoderParam.audioEncodedKbps = 64;
    encoderParam.audioEncodedCodecType = 2;

    // Set the encoding parameters of the transcoded video stream (can be ignored f
    encoderParam.videoEncodedWidth = 64;
    encoderParam.videoEncodedHeight = 64;
    encoderParam.videoEncodedFPS = 15;
    encoderParam.videoEncodedGOP = 3;
    encoderParam.videoEncodedKbps = 30;

    // Set audio mixing parameters.
    TRTCStreamMixingConfig *config = [[TRTCStreamMixingConfig alloc] init];
    // By default, leave this field empty. It indicates that all audio in the room
    config.audioMixUserList = nil;

    // Configure video mixing template (can be ignored for pure audio mix stream).
    TRTCVideoLayout *layout = [[TRTCVideoLayout alloc] init];
    config.videoLayoutList = @[layout];

    // Start mixing and relaying mixed streams.
    [self.trtcCloud startPublishMediaStream:target encoderParam:encoderParam mixing
}
```

3. Event callback and update stop task.

Task result event callback.

©2013-2022 Tencent Cloud. All rights reserved.



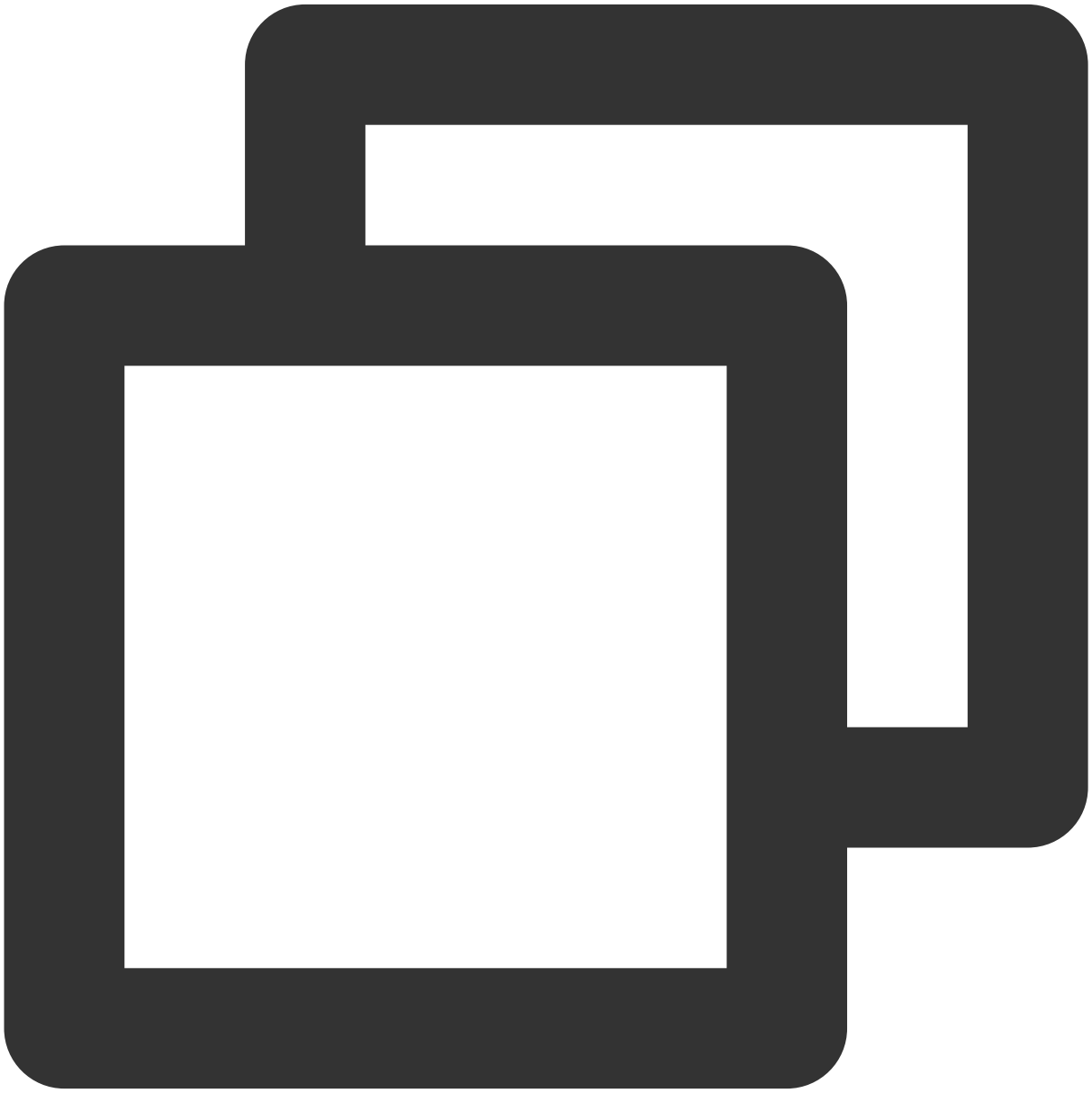
```
#pragma mark - TRTCCloudDelegate

- (void)onStartPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStri
    // taskId: When the request is successful, TRTC backend will provide the taskId
    // code: Callback result. 0 means success and other values mean failure.
}

- (void)onUpdatePublishMediaStream:(NSString *)taskId code:(int)code message:(NSStr
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure.
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)onStopPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStrin
    // When you call the stop publishing media stream API (stopPublishMediaStream),
    // code: Callback result. 0 means success and other values mean failure.
}
```

Update the published media stream.

This API sends a command to the TRTC server to update the media stream initiated by `startPublishMediaStream`.



```
// taskId: Task ID returned by the onStartPublishMediaStream callback.
```

©2013-2022 Tencent Cloud. All rights reserved.

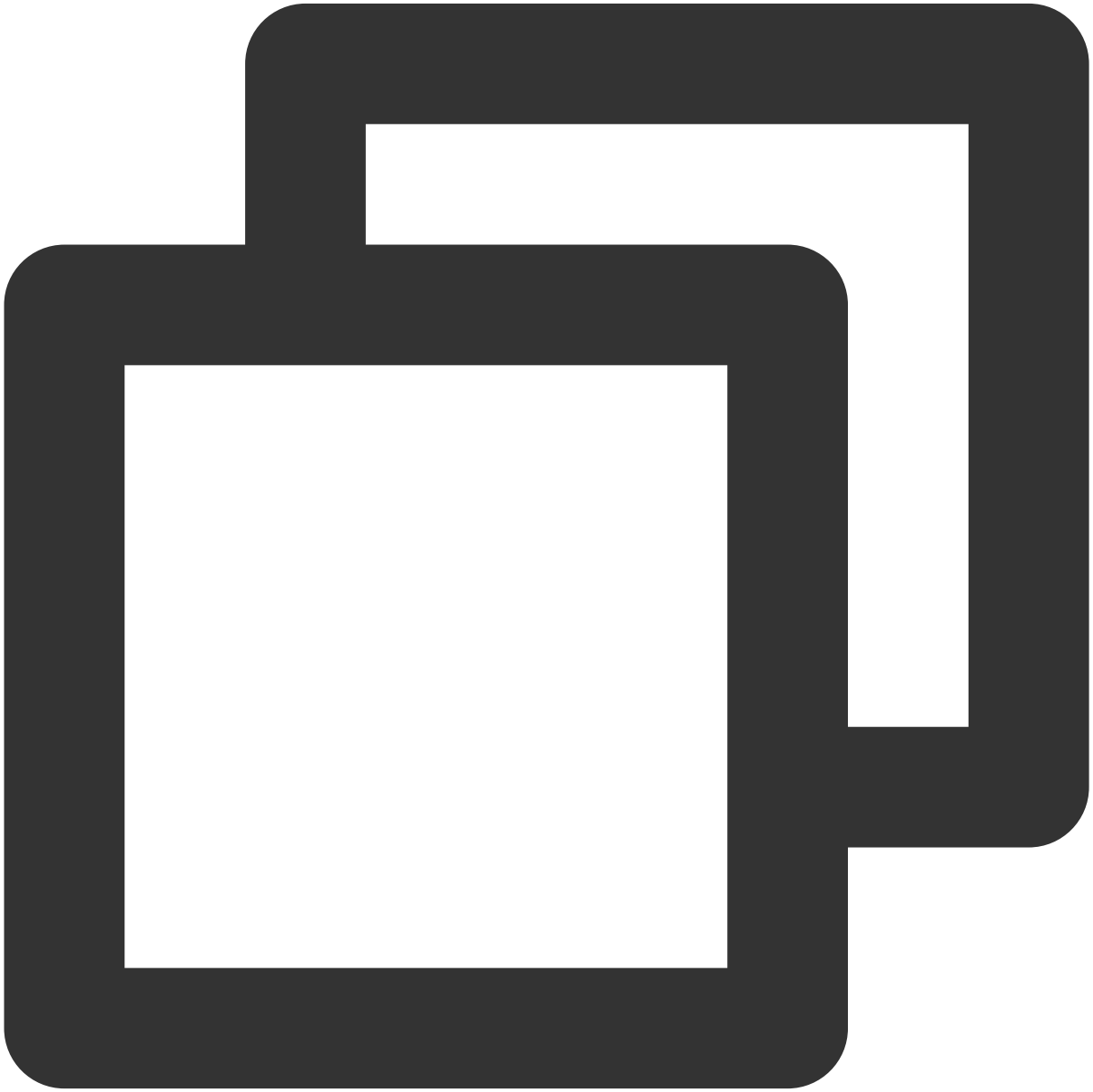
```
// target: For example, add or remove the published CDN URLs.
// params: It is recommended to maintain consistency in the encoding output paramet
// config: Update the list of users involved in mix stream transcoding, such as cro
[self.trtcCloud updatePublishMediaStream:taskId publishTarget:target encoderParam:t
```

Note:

Switching between audio only, audio and video, and video only is not supported within the same task.

Stop publishing media stream.

This API sends a command to the TRTC server to stop the media stream initiated by `startPublishMediaStream` .



©2013-2022 Tencent Cloud. All rights reserved.

```
// taskId: Task ID returned by the onStartPublishMediaStream callback.
[self.trtcCloud stopPublishMediaStream:taskId];
```

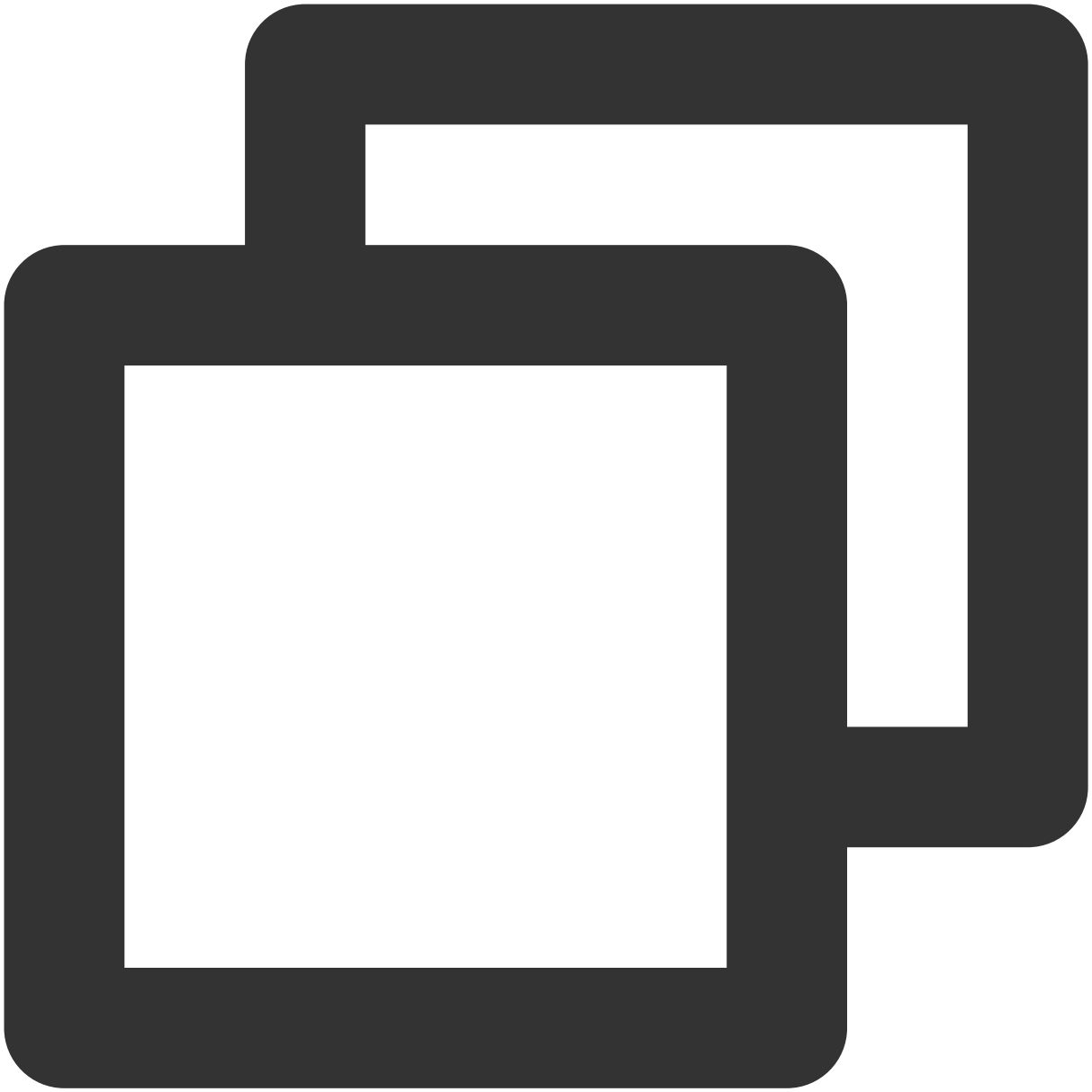
Note:

If taskId is filled with an empty string, it will stop all media streams initiated by the user through `startPublishMediaStream`. If you have only initiated one media stream or want to stop all media streams initiated by you, this method is recommended.

Real-time network quality callback

You can listen to `onNetworkQuality` to real-time monitor the network quality of both local and remote users. This callback is thrown every 2 seconds.

©2013-2022 Tencent Cloud. All rights reserved.



```
#pragma mark - TRTCCloudDelegate

- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<TRT
    // localQuality userId is empty. It represents the local user's network quality
    // remoteQuality represents the remote user's network quality evaluation result
    switch(localQuality.quality) {
        case TRTCQuality_Unknown:
            NSLog(@"Undefined.");
            break;
        case TRTCQuality_Excellent:
            NSLog(@"The current network is excellent.");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            break;
        case TRTCQuality_Good:
            NSLog(@"The current network is good.");
            break;
        case TRTCQuality_Poor:
            NSLog(@"The current network is moderate.");
            break;
        case TRTCQuality_Bad:
            NSLog(@"The current network is poor.");
            break;
        case TRTCQuality_Vbad:
            NSLog(@"The current network is very poor.");
            break;
        case TRTCQuality_Down:
            NSLog(@"The current network does not meet the minimum requirements of T
            break;
        default:
            break;
    }
}
```

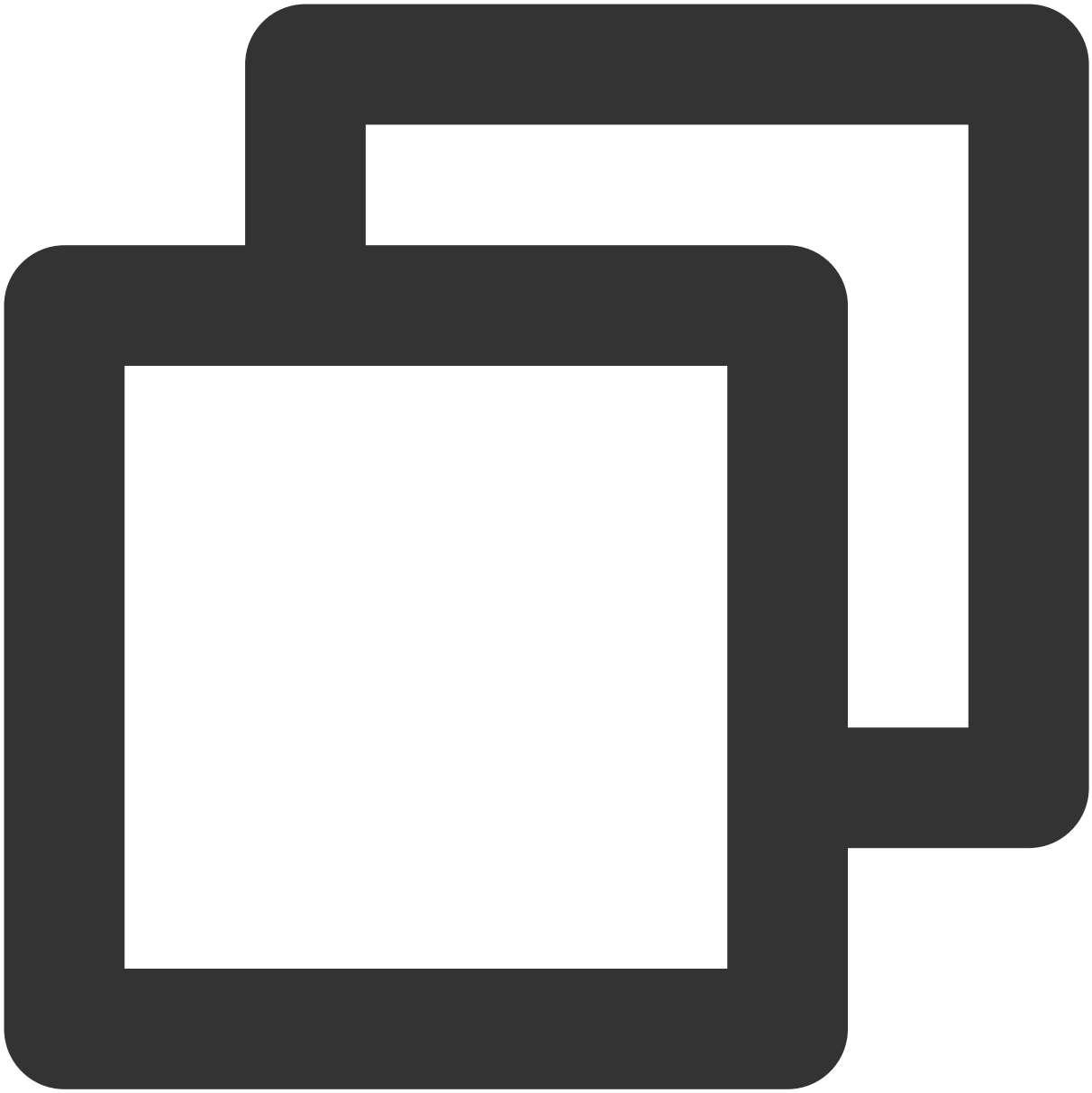
Advanced permission control

TRTC advanced permission control feature can be used to set different entry permissions for different rooms, such as for advanced VIP rooms. It can also be used to control the permission for audience to speak, such as handling ghost mics. The detailed directions are as follows:

1. Enable the Advanced Permission Control Switch in the TRTC console application's feature configuration page.
2. Generate privateMapKey on the backend. For sample code, see privateMapKey computation source code.
3. Room entry verification & speaking permission verification with PrivateMapKey.

Room entry verification

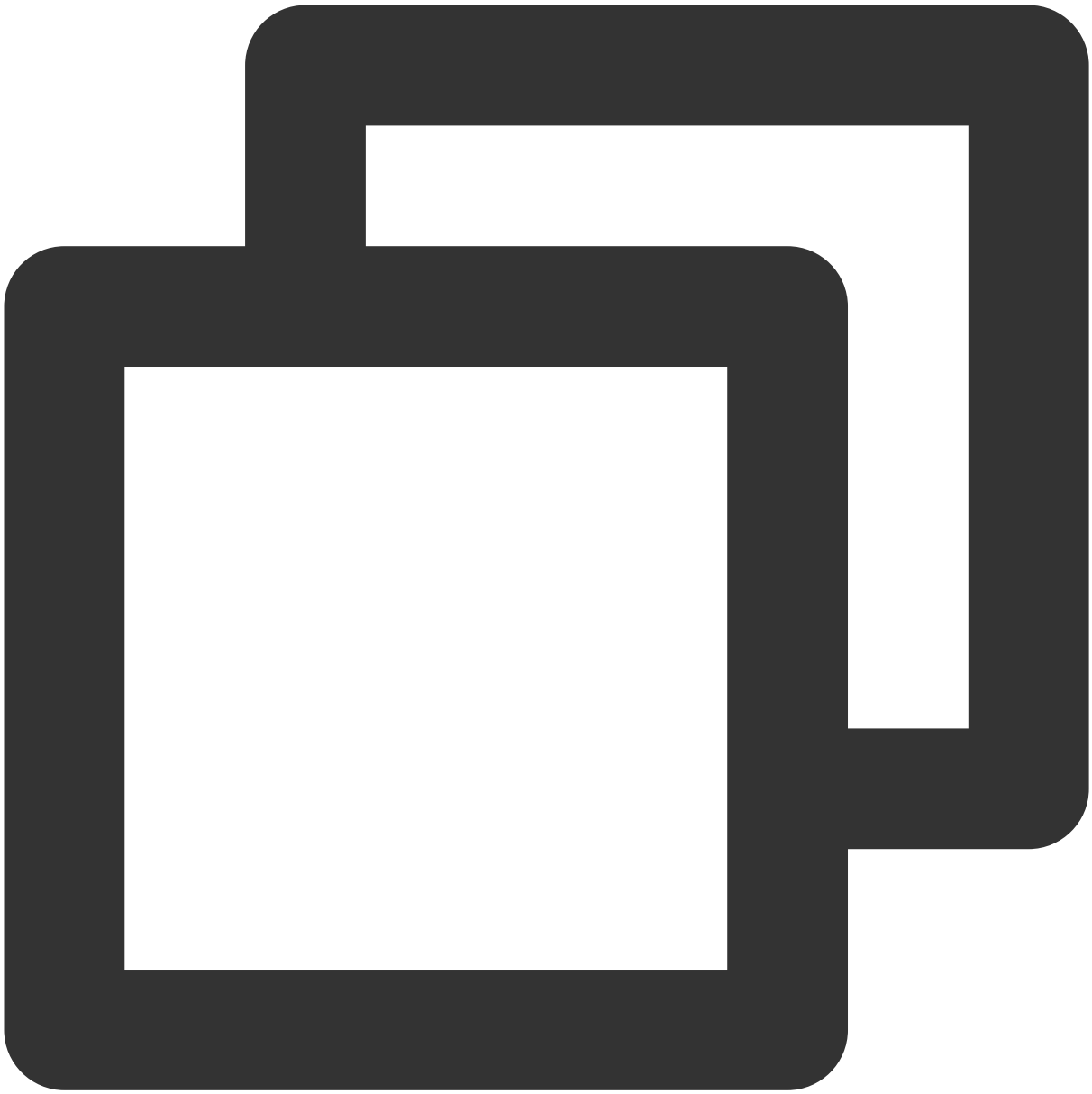
©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCParams *params = [[TRTCParams alloc] init];
params.sdkAppId = SDKAppID;
params.roomId = self.roomId;
params.userId = self.userId;
// UserSig obtained from the business backend.
params.userSig = [self getUserSig];
// PrivateMapKey obtained from the backend.
params.privateMapKey = [self getPrivateMapKey];
params.role = TRTCRoleAudience;
[self.trtcCloud enterRoom:params appScene:TRTCAppSceneVoiceChatRoom];
```

©2013-2022 Tencent Cloud. All rights reserved.

Speaking permission verification



```
// Pass in the latest PrivateMapKey obtained from the backend into the role switchi
[self.trtcCloud switchRole:TRTCRoleAnchor privateMapKey:[self getPrivateMapKey]];
```

Exception Handling

Exception error handling

©2013-2022 Tencent Cloud. All rights reserved.

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description

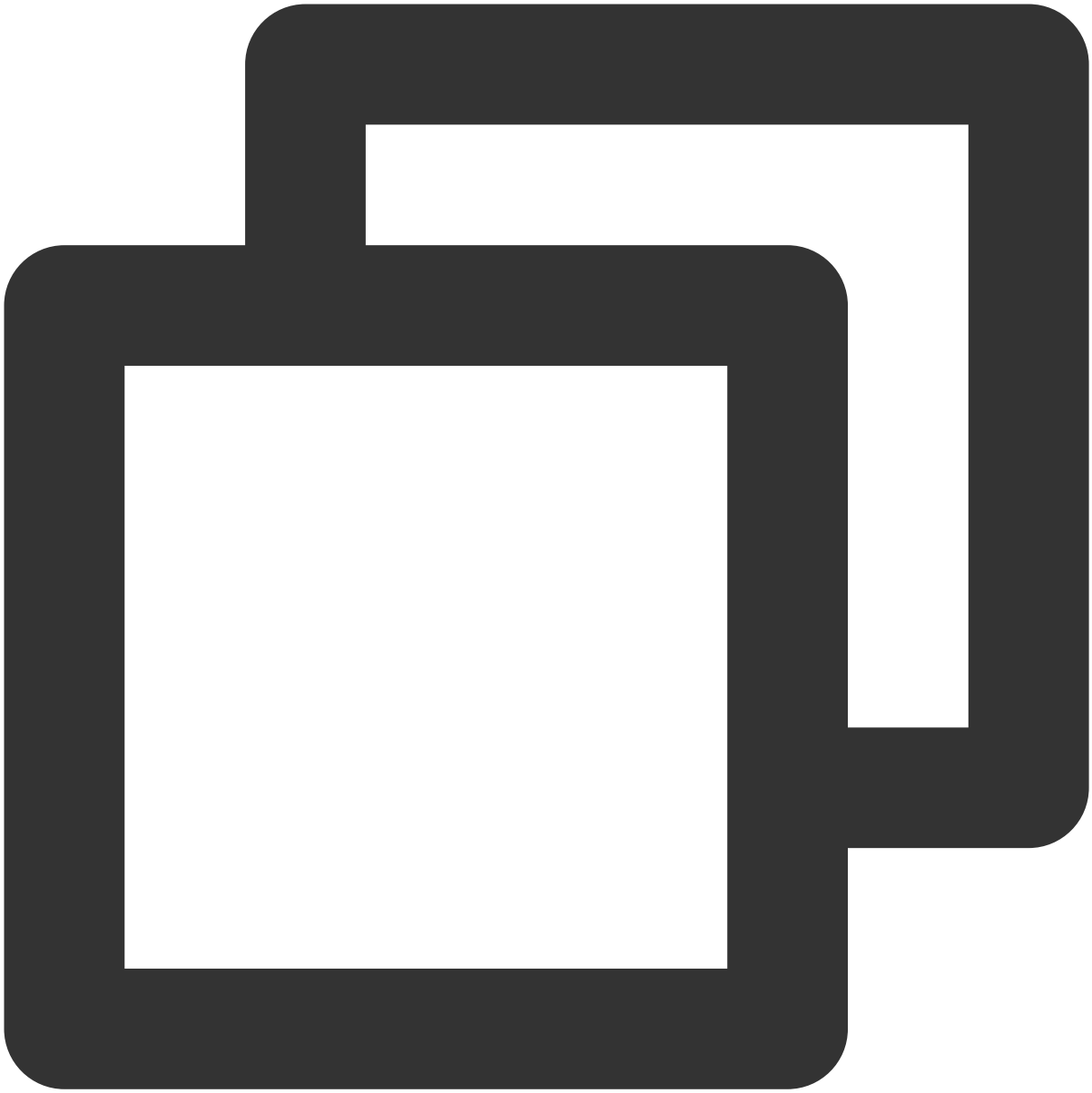
©2013-2022 Tencent Cloud. All rights reserved.

ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_SPEAKER_START_FAIL	-1321	Failed to open the speaker. For example, if there is an exception for the speaker's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Exception exit handling.

1. Network disconnection detection and timeout room exit.
You can listen for TRTC disconnection and reconnection events through the following callback notifications.
Upon receiving the `onConnectionLost` callback, display a network disconnection icon on the local seat UI to notify the user. Simultaneously, initiate a local timer. If the `onConnectionRecovery` callback is not received after exceeding the set time threshold, it means the network remains disconnected. Then, locally initiate leaving the seatmic and room exit process. Pop up a window to inform the user that they have exited the room and the page will be closed.
If the disconnection exceeds 90 seconds (default), a timeout room-exit will be triggered, and the TRTC server will remove the user from the room. If the user has an anchor role, other users in the room will receive the `onRemoteUserLeaveRoom` callback.

©2013-2022 Tencent Cloud. All rights reserved.



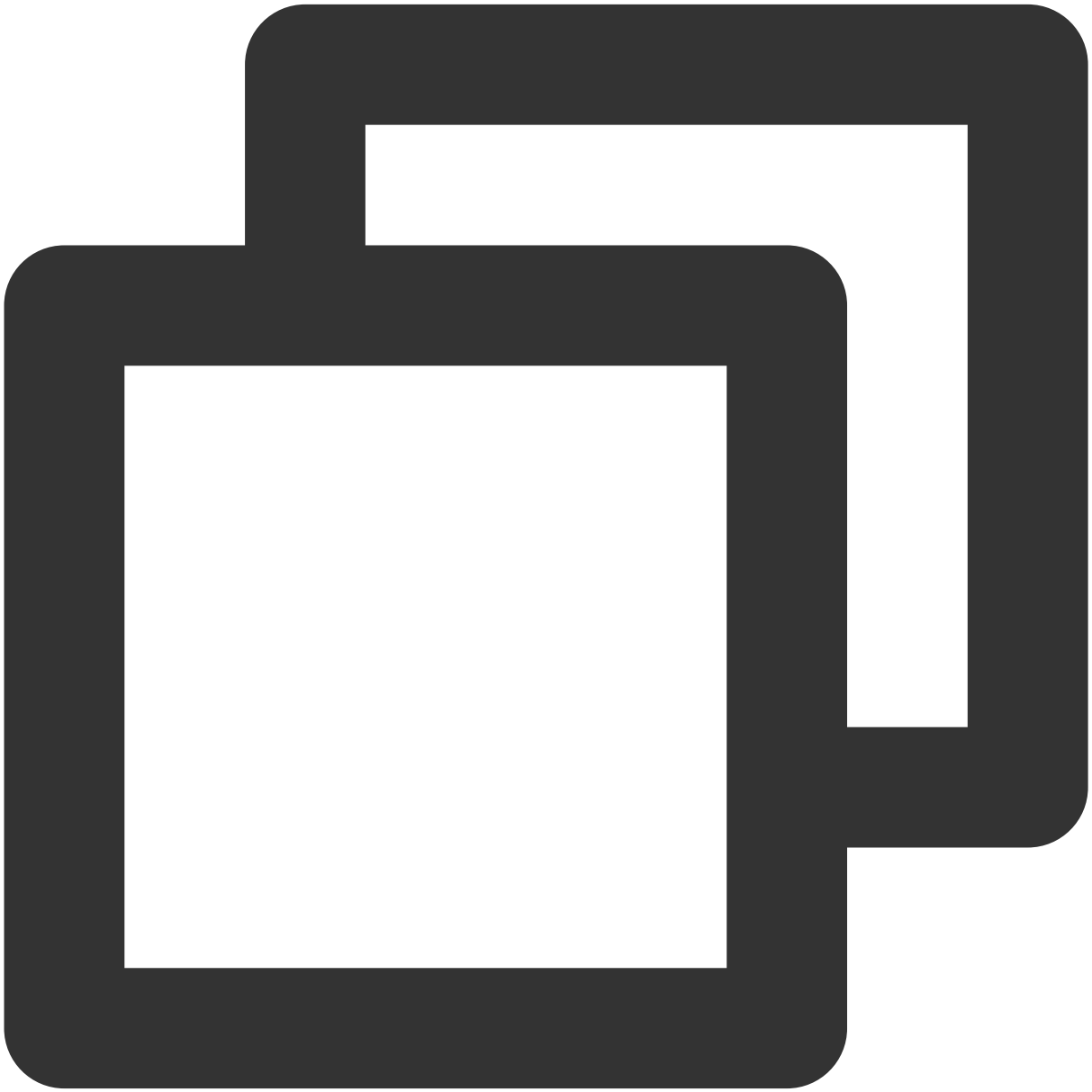
```
#pragma mark - TRTCCloudDelegate

- (void)onConnectionLost {
    // The connection between the SDK and the cloud has been disconnected.
}

- (void)onTryToReconnect {
    // The SDK is attempting to reconnect to the cloud.
}

- (void)onConnectionRecovery {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // The connection between the SDK and the cloud has been restored.
}
```

2. Automatically remove an offline-status user.

The regular statuses of IM users include online (ONLINE), offline (OFFLINE), and not logged in (UNLOGINED). The offline status typically results from the user force-stopping the process or experiencing an abnormal network disruption. You may use the feature of anchors subscribing to the connection status of mic-connecting audiences to detect offline mic-connecting audiences. And then you may remove them.

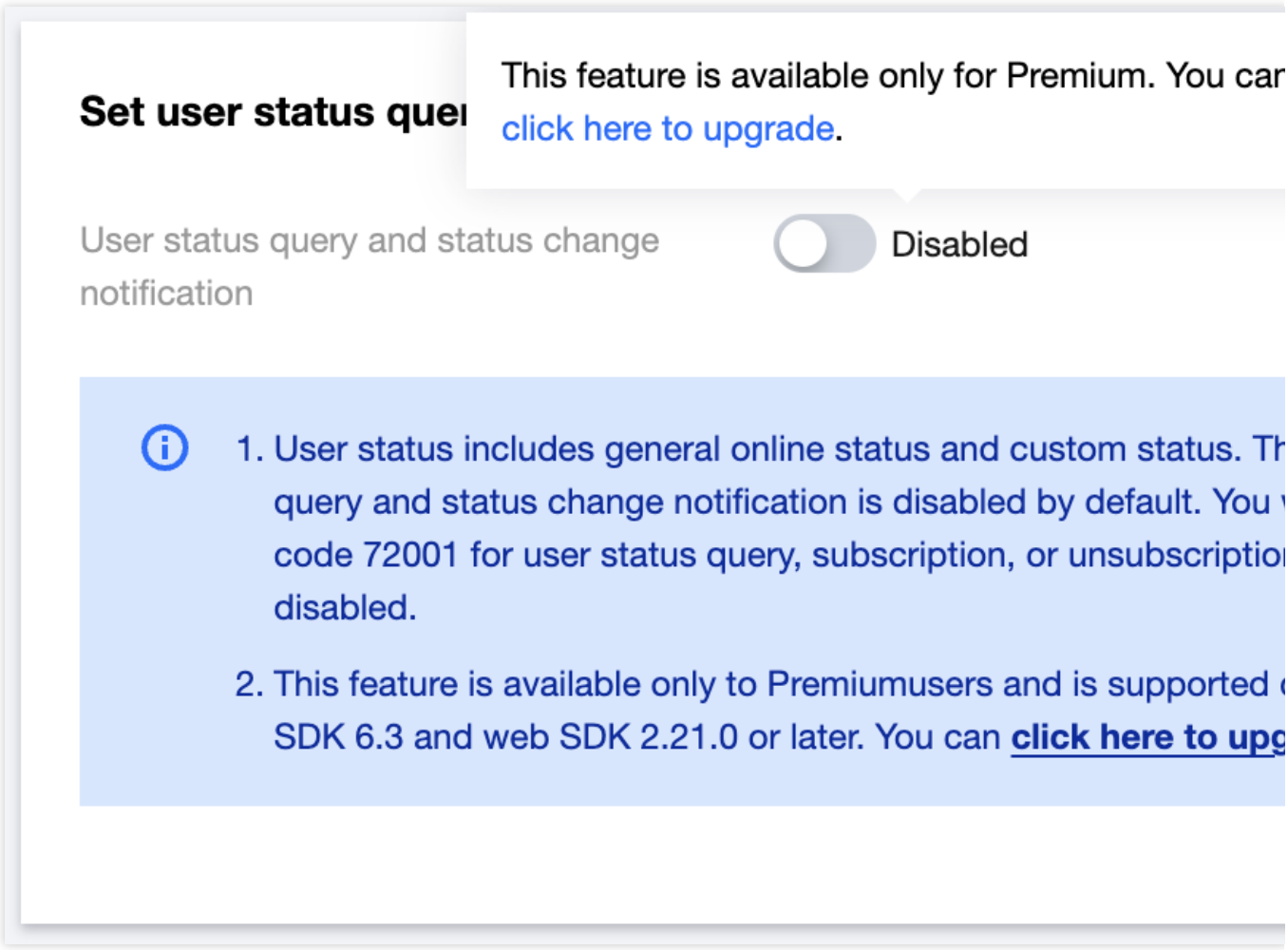


```
// Anchor subscribes to the connection status of mic-connecting audiences.
[[V2TIMManager sharedInstance] subscribeUserStatus:userList succ:^{
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Subscription of user status succeeded.
} fail:^(int code, NSString *desc) {
    // Subscription of user status failed.
}];

// Anchor unsubscribes from the connection status of audiences leaving the seat.
[[V2TIMManager sharedInstance] unsubscribeUserStatus:userList succ:^{
    // Unsubscription of user status succeeded.
} fail:^(int code, NSString *desc) {
    // Failed to unsubscription of user status.
}];

// User status change notification and processing.
[[V2TIMManager sharedInstance] addIMSDKListener:self];

- (void)onUserStatusChanged:(NSArray<V2TIMUserStatus *> *)userStatusList {
    for (V2TIMUserStatus *userStatus in userStatusList) {
        NSString *userId = userStatus.userID;
        V2TIMUserStatusType status = userStatus.statusType;
        if (status == V2TIM_USER_STATUS_OFFLINE) {
            // Remove an offline-status user.
            [self kickSeatWithIndex:[self getSeatIndexWithUserId:userId]];
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.



Note:

User-status subscription needs to be upgraded to the advanced package. For details, see Basic Service Details.

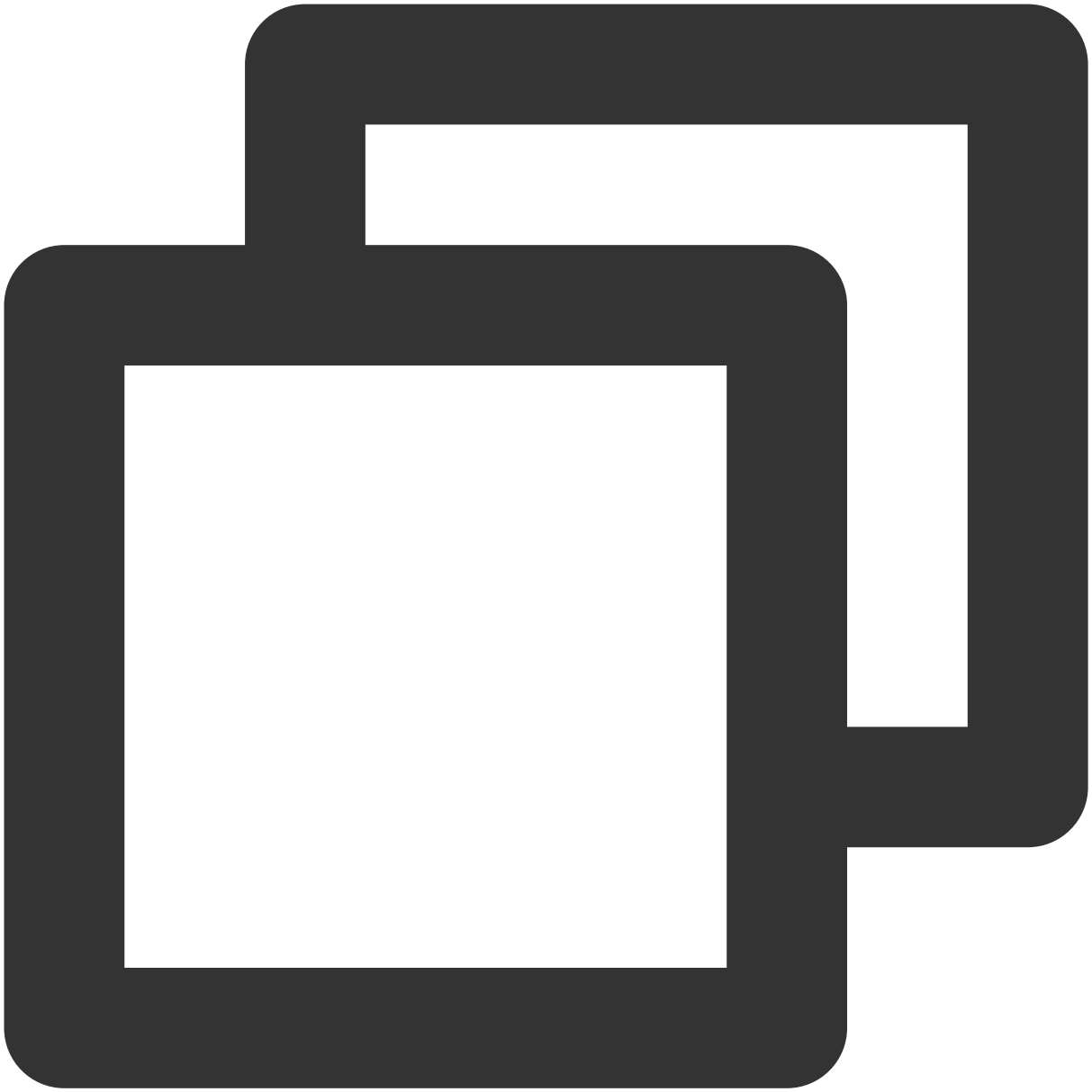
User-status subscription needs **User status query and status change notification configuration** to be enabled in Instant Messaging (IM) console in advance. Failure to enable will result in an error when calling `subscribeUserStatus` .

Server removes users from and dissolve the room.

1. Server removes users.

First, call the TRTC server user-removing API RemoveUser (for integer room IDs) or RemoveUserByStrRoomId (for string room IDs) to remove the target user from the TRTC room. The input example is as follows.

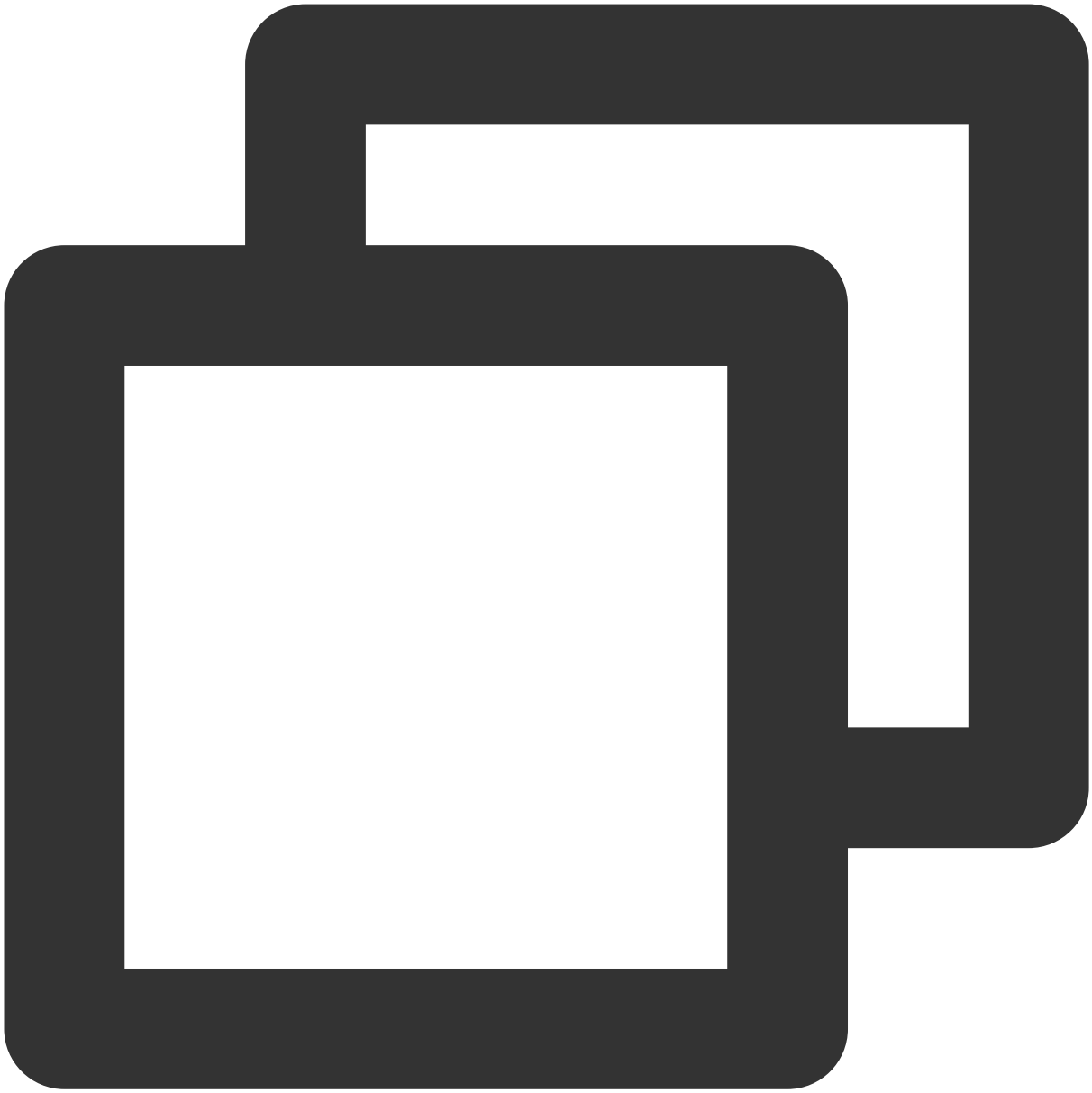
©2013-2022 Tencent Cloud. All rights reserved.



```
https://trtc.tencentcloudapi.com/?Action=RemoveUser
&SdkAppId=1400000001
&RoomId=1234
&UserIds.0=test1
&UserIds.1=test2
&<Common request parameters>
```

After removing the user successfully, the target user will receive the `onExitRoom()` callback on the client, with the `reason` value being 1. At this moment, you can handle leaving the seat and exiting the IM group in this callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Exit TRTC room event callback.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        // Actively call exitRoom to exit the room.
        NSLog(@"Exit current room by calling the 'exitRoom' api of sdk ...");
    } else {
        // reason 1: Removed from the current room by the server.
        // reason 2: The current room is dissolved.
        NSLog(@"Kicked out of the current room by server or current room is dissolv
        // Leave the seat.
        [self leaveSeatWithIndex:seatIndex];
```

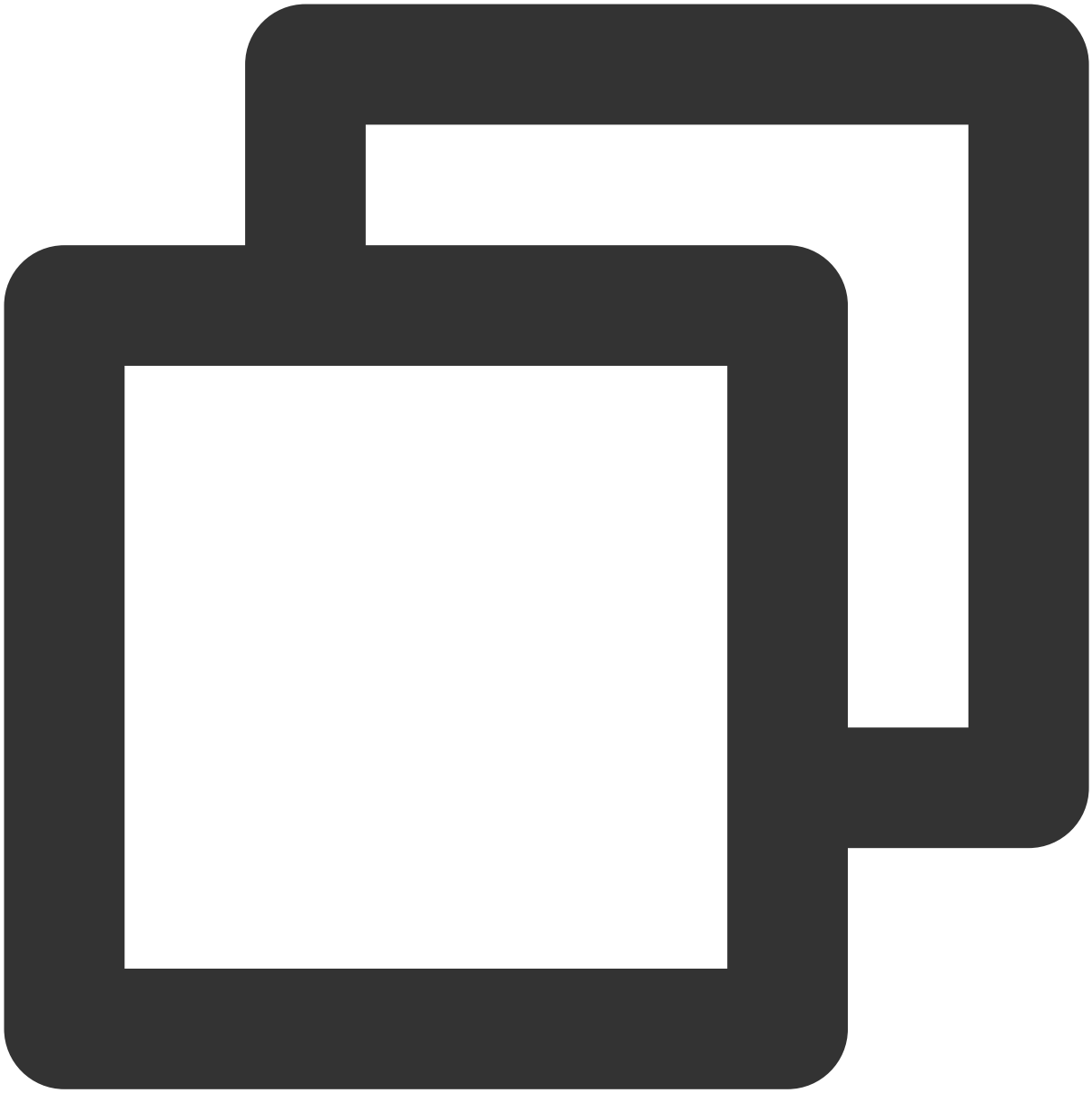
©2013-2022 Tencent Cloud. All rights reserved.

```
        // Exit IM group.
        [[V2TIMManager sharedInstance] quitGroup:groupID succ:^{
            // Exiting the group successful.
        } fail:^(int code, NSString *desc) {
            // Exiting the group failed.
        }];
    }
}
```

2. Server dissolves the room.

First, call the IM server group dissolvement API destroy_group to dissolve the target group. The example request URL is as follows.

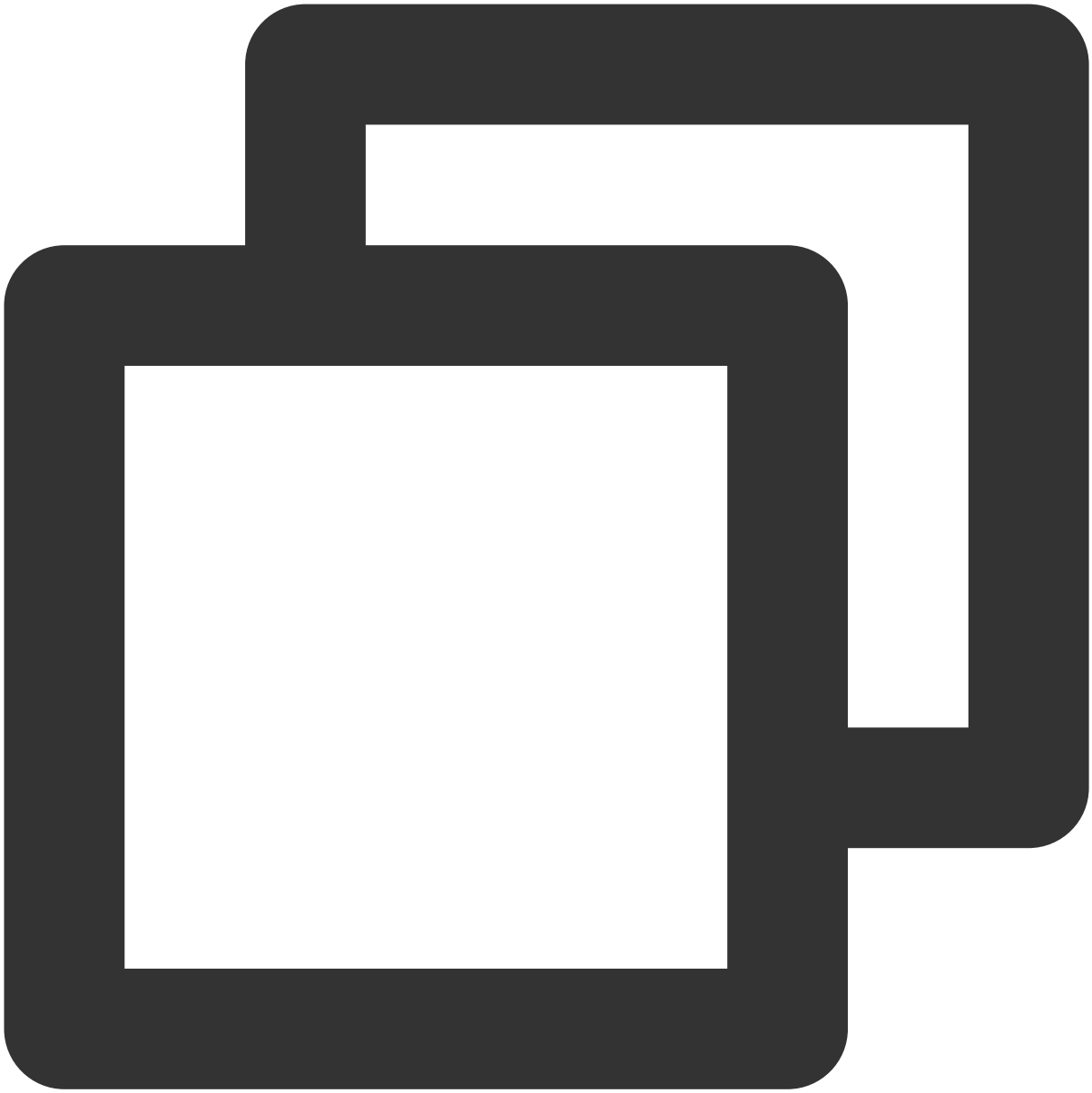
©2013-2022 Tencent Cloud. All rights reserved.



```
https://xxxxxx/v4/group_open_http_svc/destroy_group?sdkappid=88888888&identifier=ad
```

After the group is dissolved, all members within the target group will receive the `onGroupDismissed()` callback on clients. At this point, you can handle operations such as exiting the TRTC room in this callback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Group dissolved callback.
[[V2TIMManager sharedInstance] addGroupListener:self];
- (void)onGroupDismissed:(NSString *)groupID opUser:(V2TIMGroupMemberInfo *)opUser
    // Exit TRTC room.
    [self.trtcCloud stopLocalAudio];
    [self.trtcCloud exitRoom];
}
```

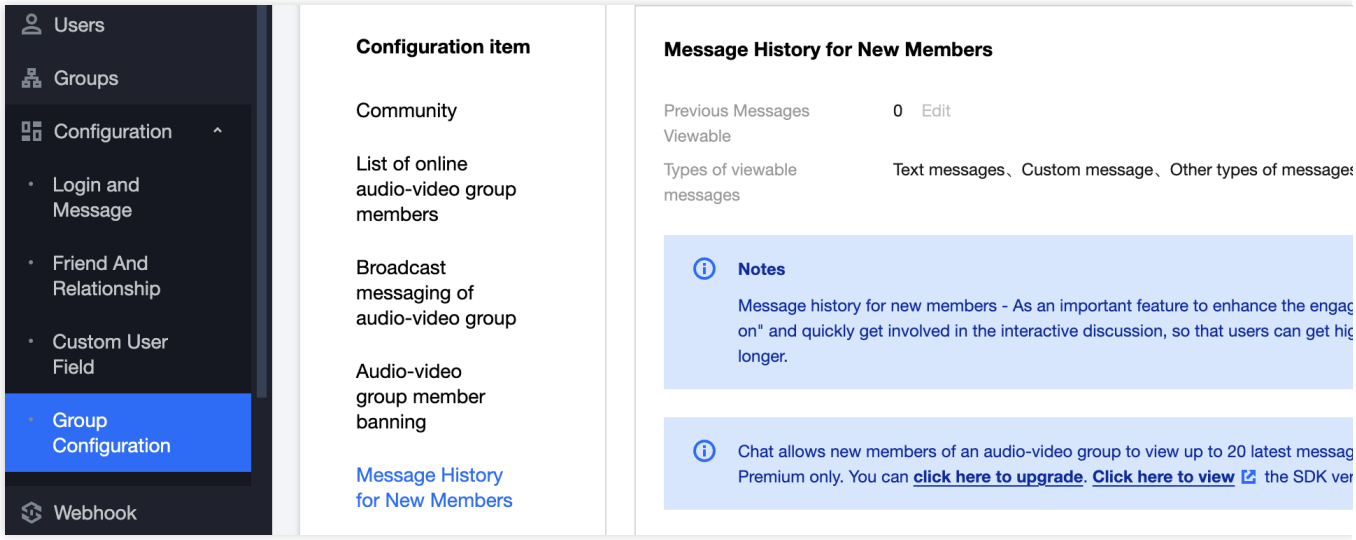
Note:

©2013-2022 Tencent Cloud. All rights reserved.

When all users in the room have completed exiting by calling `exitRoom()`, the TRTC room will be automatically dissolved. Of course, you can also mandatorily dissolve the TRTC room by calling the server API DismissRoom (for integer room IDs) or DismissRoomByStrRoomId (for string room IDs).

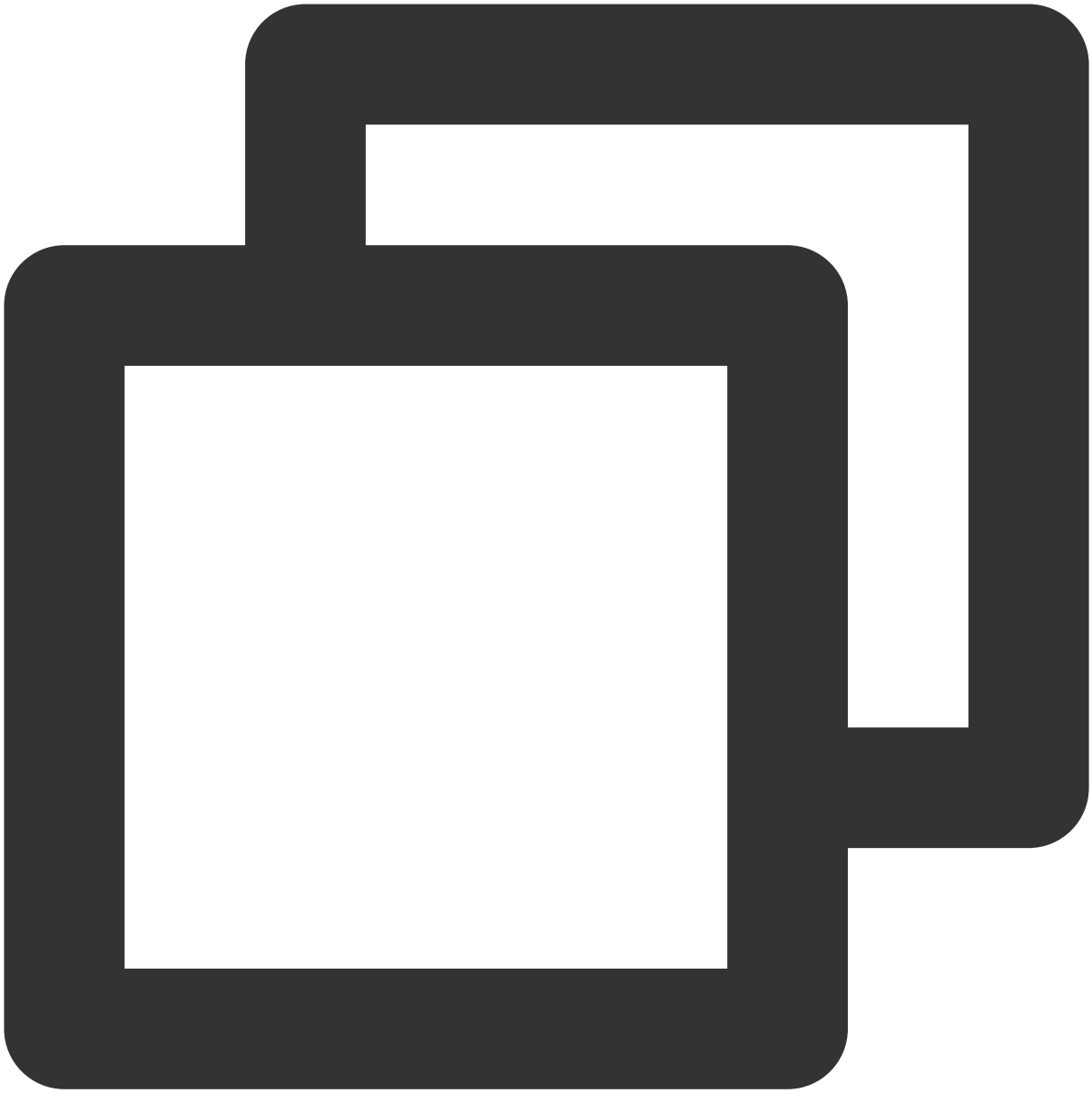
View live streaming room's historical messages upon room entry.

By default, using AVChatRoom does not store live streaming room's historical messages. Therefore, when new users enter the live streaming room, they can only see messages sent after their entry. To optimize the experience for new users joining the group, you can configure the number of messages new live streaming group users can pull before joining the group in the console, as shown in the figure:



Pulling historical messages before joining the live streaming group for live group users is the same as pulling historical messages for other groups, as shown in the sample code:

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMMessageListGetOption *option = [[V2TIMMessageListGetOption alloc] init];
option.getType = V2TIM_GET_CLOUD_OLDER_MSG; // Pull earlier existing messages from
option.getTimeBegin = 1640966400;        // Starting from midnight January 1, 2022.
option.getTimePeriod = 1 * 24 * 60 * 60; // Pull messages from a 24-hour period.
option.count = INT_MAX;                  // Return all messages within the time ran
option.groupID = #your group id#;        // Pull messages for the group chat.
[V2TIMManager.sharedInstance getHistoryMessageList:option succ:^(NSArray<V2TIMMessa
    NSLog(@"success");
} fail:^(int code, NSString *desc) {
    NSLog(@"failure, code:%d, desc:%@", code, desc);
}];
```

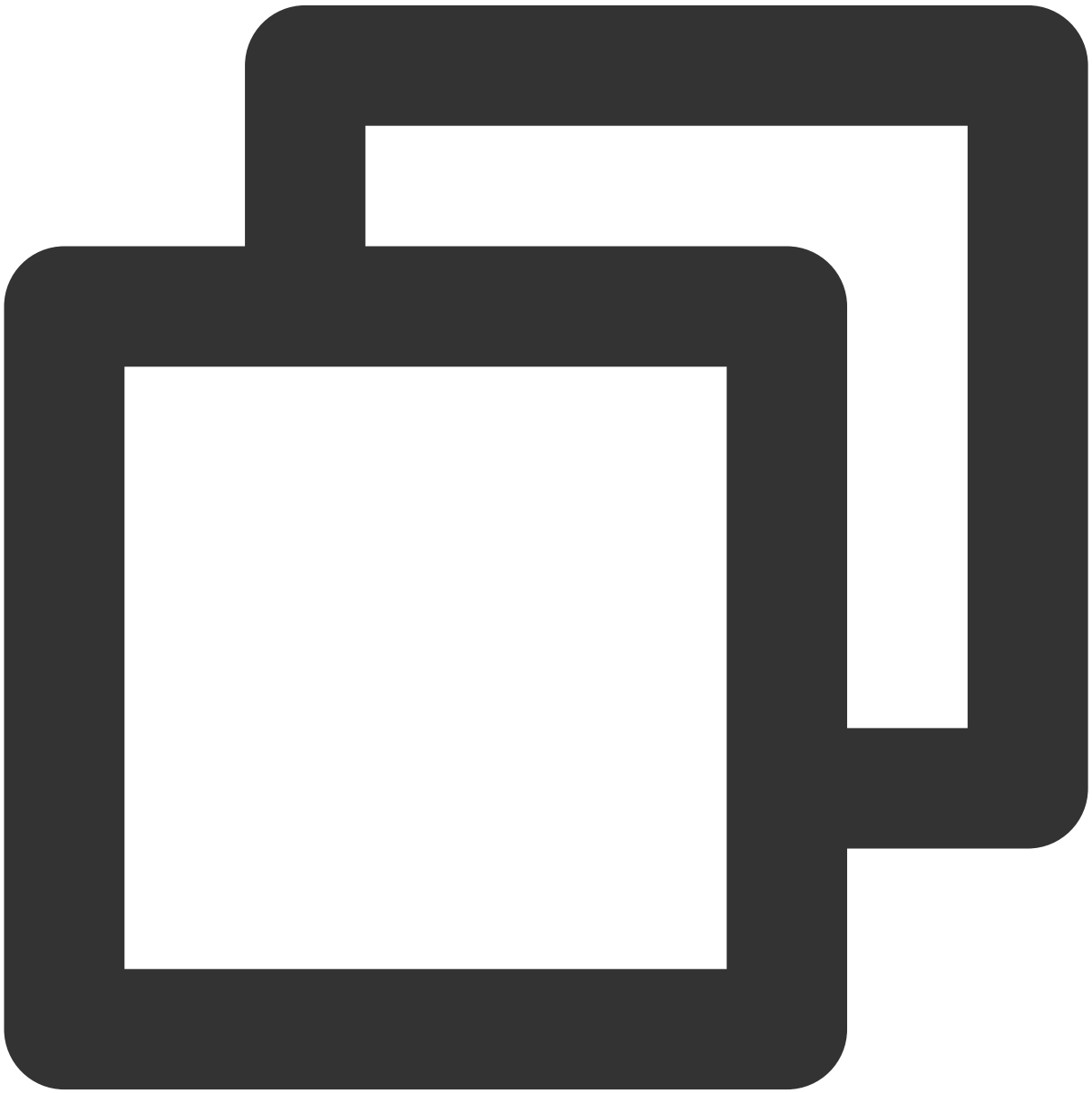
©2013-2022 Tencent Cloud. All rights reserved.

Note:

This feature is only available to advanced users. It only supports pulling up to 20 historical messages within 24 hours from the group.

Enter the room to sense the mute status of on-mic anchors.

Solution 1: Default all anchors to mute status upon room entry, then unmute the corresponding anchors based on the `onUserAudioAvailable(userId, true)` callback.

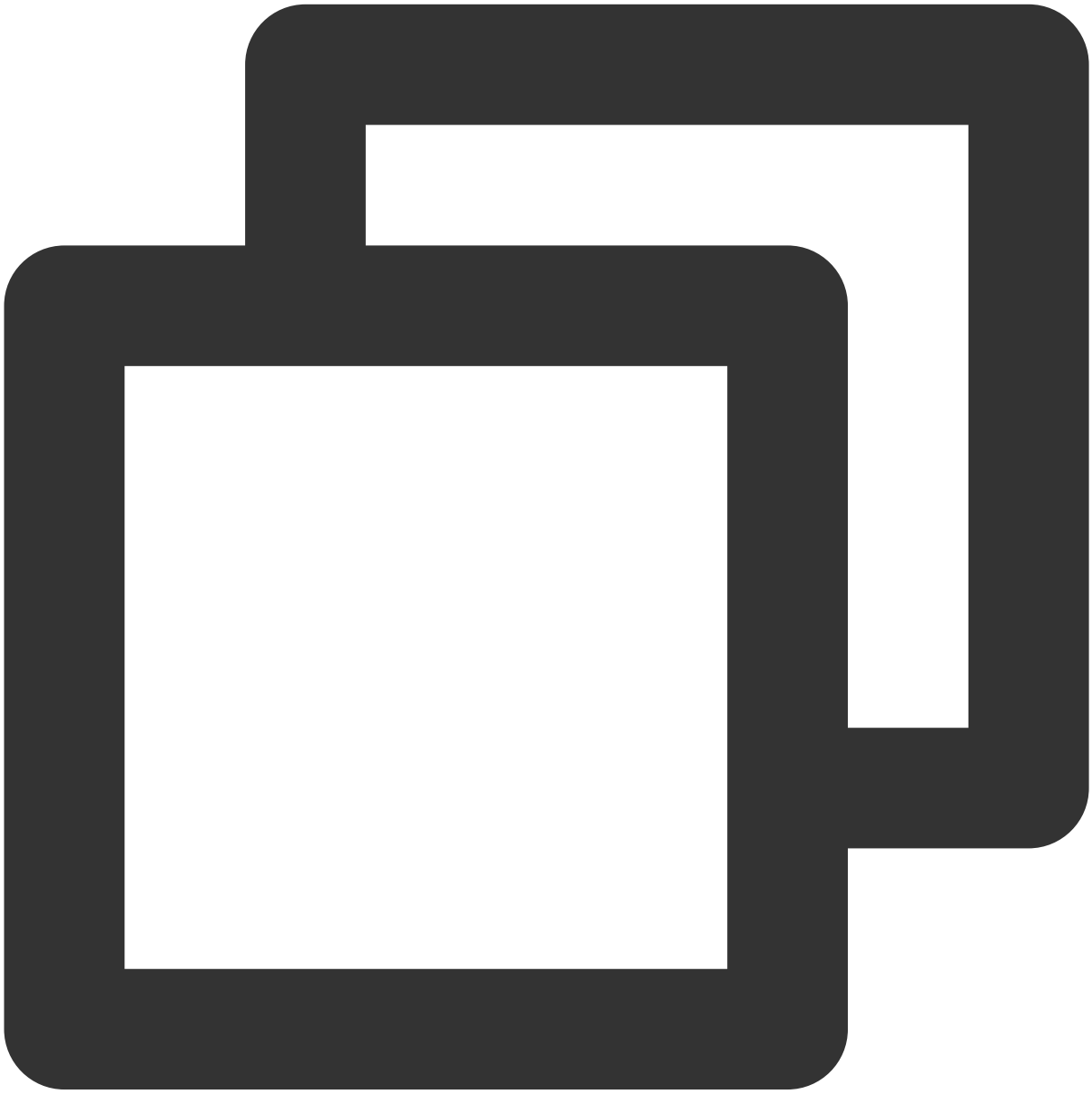


```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    if (available) {
        // Unmute the corresponding anchors.
    }
}
```

Solution 2: Store the mute status of the anchors in the IM group attributes. Audiences entering the room obtain all group attributes to parse the mute status of on-mic anchors.



```
[[V2TIMManager sharedInstance] getGroupAttributes:groupID keys:nil succ:^(NSMutable
    // Successfully obtained group attributes. It is assumed that the key used to s
    NSString *muteStatus = groupAttributeList[@"muteStatus"];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Parse muteStatus, and obtain the mute status of each on-mic anchor.
} fail:^(int code, NSString *desc) {
    // Failed to obtain the group attributes.
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

Online Karaoke Scenario Solution

Last updated：2024-07-25 17:01:48

Scenario Introduction

According to data from iiMedia Research, in 2021, the number of online Karaoke users in China was about 510 million, with a penetration rate of approximately 49.7%. Online Karaoke offers a more immersive experience and its diverse gameplay caters to the personalized needs of different user groups, becoming one of the main projects in the online pan-entertainment field. Based on network technology innovations, online Karaoke apps continue to launch diverse singing patterns and gameplay, and the continuously enriched features have enhanced the practicality and playability of online Karaoke apps. This document will provide a detailed introduction to the online Karaoke scenario-based solution based on Tencent Real-Time Communication (TRTC) in the following sections.

©2013-2022 Tencent Cloud. All rights reserved.

Implementation Scheme

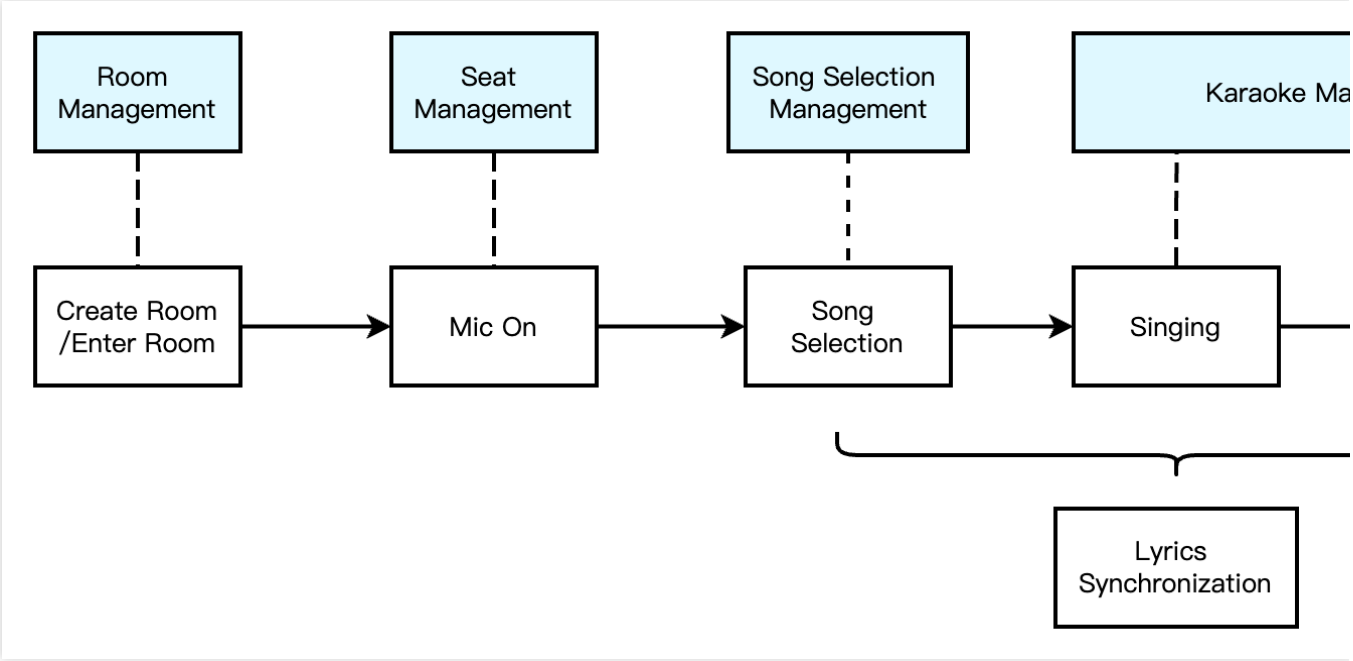
Typically, implementing a complete online Karaoke scenario involves multiple functional modules: room management, seat management, song selection management, karaoke management, scoring management, etc. Key actions and feature points under each functional module are shown in the following table:

Functional Module	Key Actions and Feature Points
Room Management	Room list, create a room, enter a room, exit a room, and terminate a room.
Seat Management	Become a speaker/listener, seat control, change a speaker, lock the seat, invite a listener to speak, and mute a speaker.

©2013-2022 Tencent Cloud. All rights reserved.

Song Selection Management	Song list display, searching songs, song selection, queue management, and selected song list.
Karaoke Management	Karaoke play mode, start/stop/switch songs, accompaniment and vocal volume adjustment, reverb/sound effects, original sound and accompaniment switch, and lyrics synchronization
Scoring Management	Singing Scoring and Pitch Line Display

The overall business process of the online Karaoke scene is shown in the following diagram. The room owner creates a Karaoke room, and users can choose the Karaoke room they are interested in to enter. After entering the room, users can request to speak to participate in the interaction. After becoming a speaker, they can also choose their favorite songs to sing and wait in line. When it is their turn, they can sing along with the accompaniment. Of course, users can also choose to become a speaker directly to participate in a chorus. These are two different Karaoke play modes. During the singing process, there will be pitch scoring for individual sentences, and there will also be singing scoring for the entire song after the singing is finished.

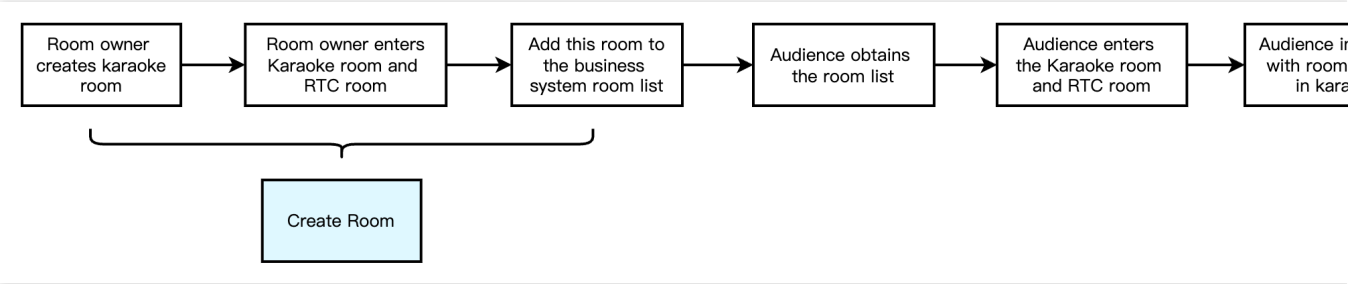


Room Management

The Room Management module is primarily responsible for maintaining the room list, which includes functions such as create a room, enter a room, exit a room, and terminate a room. Additionally, a Karaoke room differs from regular rooms in that it requires a separate Karaoke room identifier to initiate related component management: Song Selection Management, Karaoke Management, Scoring Management, etc.

©2013-2022 Tencent Cloud. All rights reserved.

Create Room: After users log in to the business system, they can create a room. The room list needs to be updated after a room is created.
Enter a Room: Users can choose to enter an existing room. Upon entering, the current list of room members should be updated.
Exit a Room: Users can choose to exit the current room. Upon exiting, the current list of room members needs to be updated with a delete operation.
Terminate Room: After all users exit the room, it needs to be terminateed. Upon destruction, the room list needs to be updated with a delete operation.



Note:
Room Management is a necessary functional module for implementing online Karaoke but is not the main functional module. Specific implementation can be achieved through integration with business systems and IM&TRTC SDKs. For details, see Voice Chat Room > Room Management.

Seat Management

In a Karaoke room, seats are generally orderly and limited. Seat management primarily involves defining the number of seats in the room based on the business scenario, as well as managing the status of all seats in the current room. Seat management includes features such as become a speaker/listener, seat control, change a speaker, lock the seat, invite a listener to speak, and mute a speaker.
After users enter a room, only idle seats can be applied for.
After the room owner approves a user's speaker request, the corresponding seat status should change to occupied.
When a user stops streaming and becomes a listener, the corresponding seat status should revert to idle.
The room owner has the authority to lock the seat, invite a listener to speak, remove a speaker, mute a speaker, etc.
Note:
Seat management is a necessary functional module for implementing online Karaoke but is not the main functional module. Specific implementation can be achieved through integration with business systems and IM&TRTC SDKs. For details, see Voice Chat Room -> Seat Management.

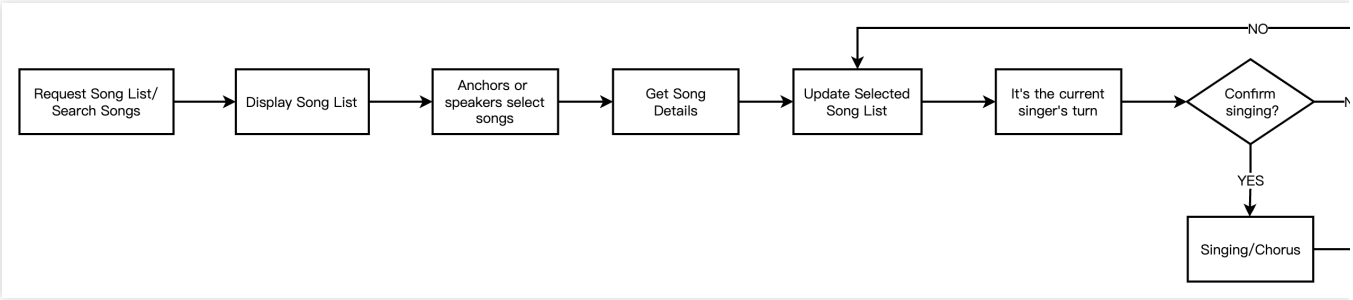
Song Selection Management

Basic Introduction

©2013-2022 Tencent Cloud. All rights reserved.

Song selection management is an important part of the online Karaoke scenario, mainly including features such as song list display, search for songs, song selection, and queue management, selected song list. Each Karaoke room needs to maintain a selected song list and an auto queue management feature, which needs to be implemented by the business backend. Meanwhile, aspects related to accompaniment resources such as song list display and song search are recommended to be implemented with accompaniment library products for overseas users.

Implementation Process



The entire song selection management, mainly involves the business-side app, business backend, and music library, where their respective functions are as follows:

Business-Side App:

Call the song selection API to report song information.

Call the change song-switching API to notify the business backend.

Call the singing confirmation API to notify the business backend.

Business Backend:

Maintain the selected song list.

Send notifications to tell the business-side app to switch the song.

Music Library:

Provide authorized music resources for TRTC to play.

Provide lyric files and pitch files matching the music resources.

Karaoke Management

The Karaoke system primarily includes functions such as: Karaoke play mode, start/stop/switch songs, accompaniment and vocal volume adjustment, reverb/sound effects, original sound and accompaniment switch, and lyrics synchronization. Below, we will introduce the implementation process of the Karaoke management module in detail through two typical Karaoke gameplay: solo singing and real-time chorus.

Solo Singing

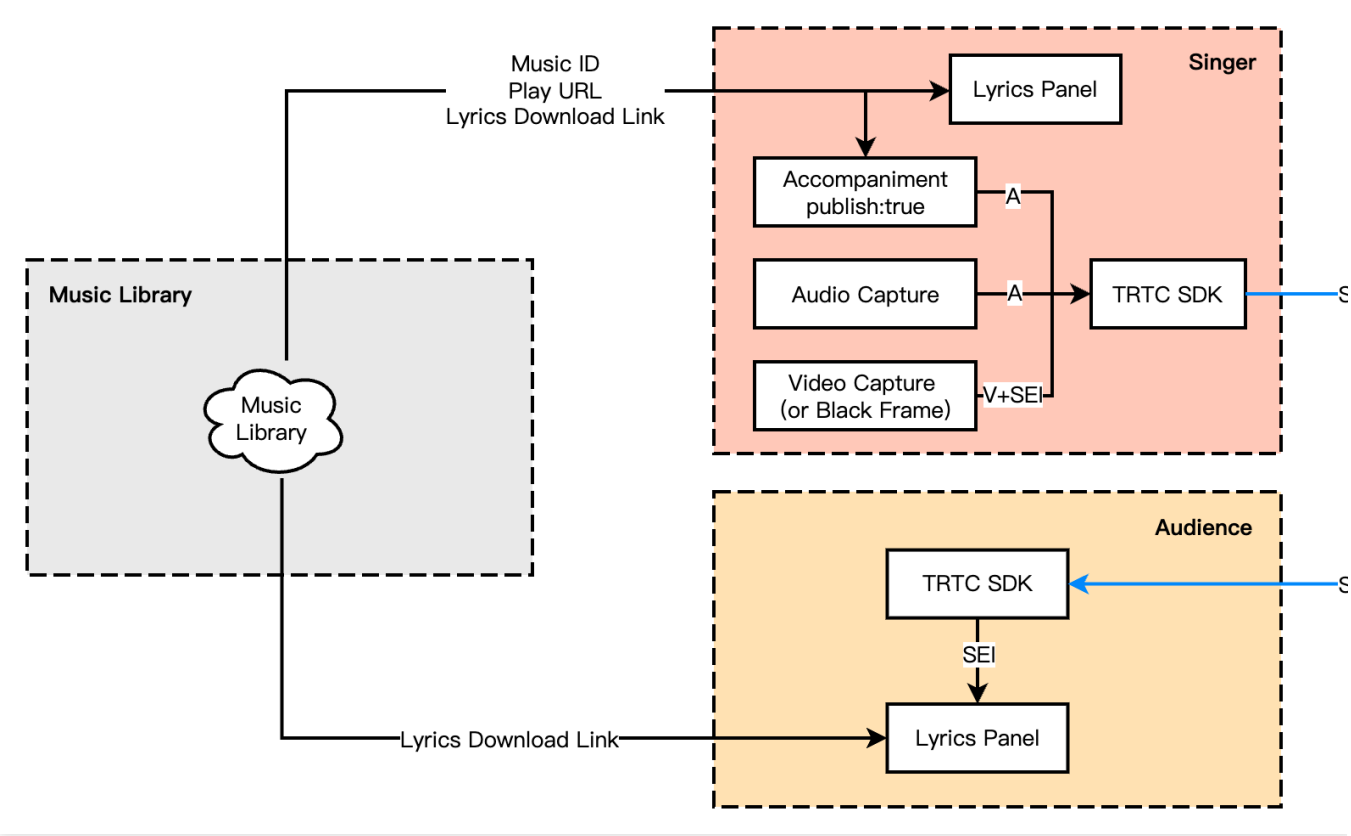
Solo singing: Primarily in the interactive Karaoke scenario with multiple participants, after the anchor/audience members become speakers, they can proceed to select songs. Once a song selection is successful, it will be

©2013-2022 Tencent Cloud. All rights reserved.

displayed collectively on the song selection platform. When it's someone's turn to select a song, the corresponding individual will play the song's accompaniment, start singing, and undergo scoring.

Solution Architecture

The overall solution primarily relies on the music library for song and lyric resources and TRTC for streaming the singer's vocals, song accompaniment, and streaming. The solution architecture is as follows:



Specific Implementation

In the solo singing scenario, different roles have different implementation processes. There are two roles: singer and audience. The description and differences of their roles are detailed in the table below:

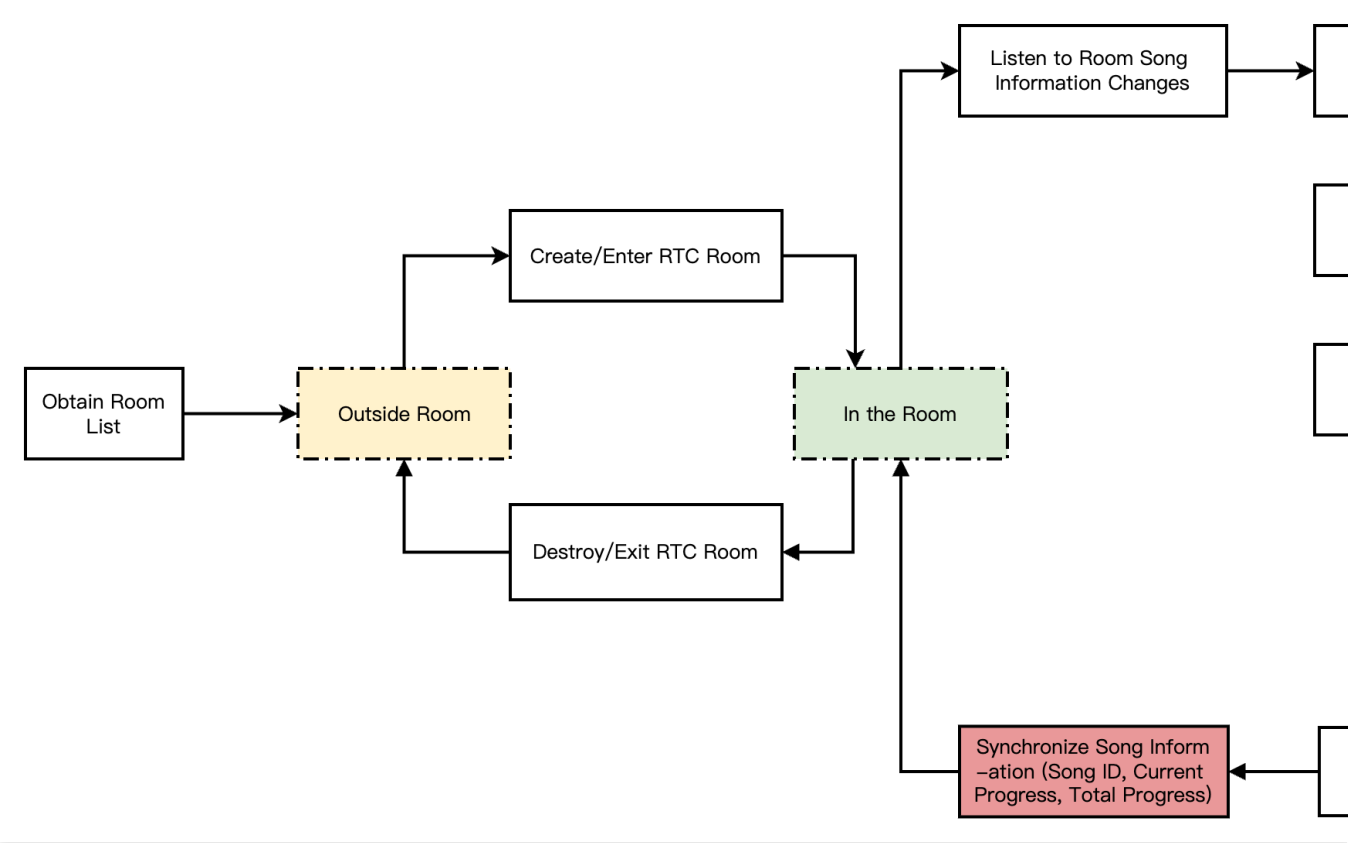
Roles	Description	Differences
Singer	The singer in the Karaoke room is evolved from the anchor/audience who selects songs and sings after becoming a speaker. After leaving the room, the room is automatically dissolved and the list of selected songs is automatically cleared.	The role must be an anchor. Upstream audio and video (no video upstream black frame) Play BGM Send SEI information (sending lyric information) Song Selection
Audience	The audience in the Karaoke room plays the media stream of the singer or other people.	The role is an audience, but can also become an anchor by becoming a speaker.

©2013-2022 Tencent Cloud. All rights reserved.

		Downstream Audio and Video Streams Receive SEI information (receive lyric information)

Implementation Process

Singer



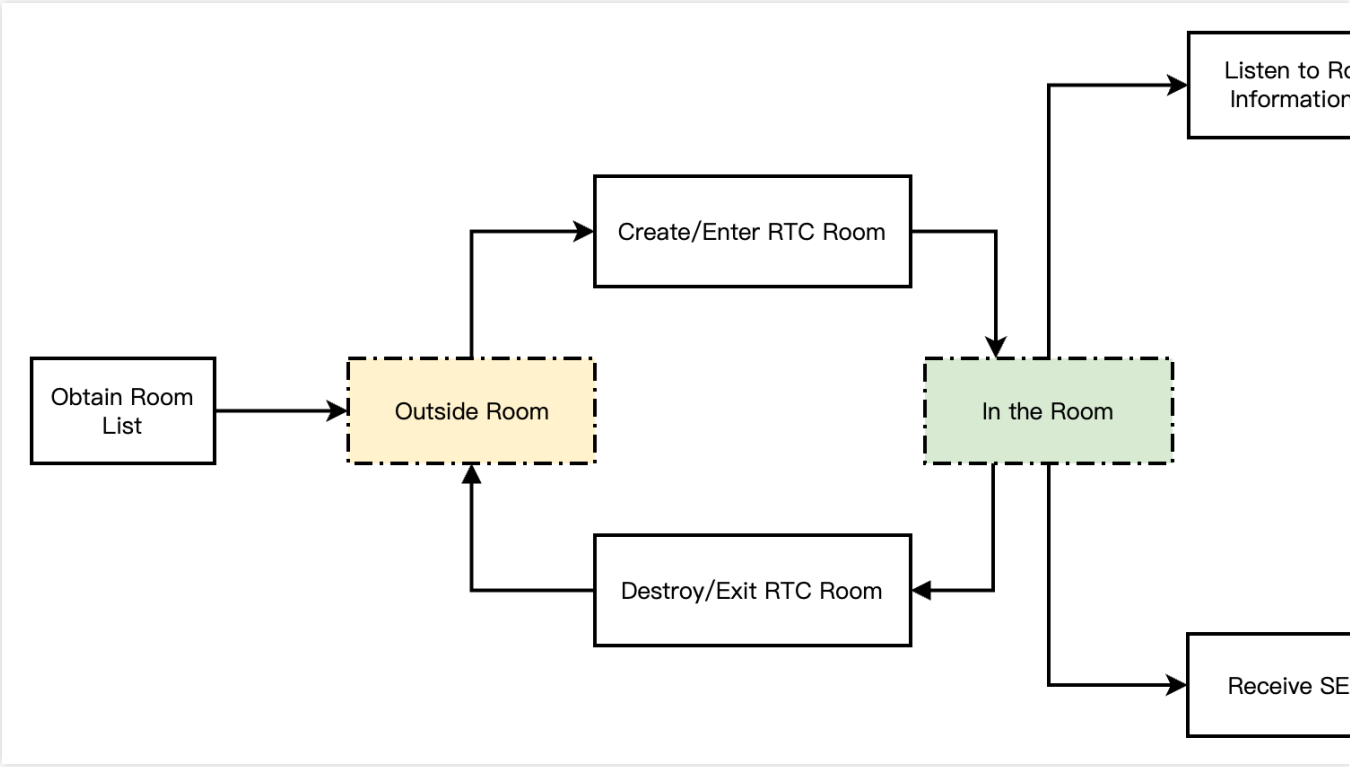
1. The anchor/audience creates/enters a TRTC room, and automatically becomes a singer after selecting a song.
2. After the singer selects a song, the song/lyric is downloaded, and then the song is played through the BGM interface.
3. If the singer does not bring up the video upstream, they need to enable the video upstream.
4. Synchronize the lyric progress of everyone through SEI information.
5. When the singer becomes a listener, all songs they selected will be cleared, and they revert to their original role.
6. After the anchor/audience exits the room, the TRTC room will be dissolved.

Note:

Anchors/Audiences on the seat can select songs for themselves or others, but the corresponding singer must play the BGM; otherwise, it may cause asynchrony between the singer and the song due to latency (about 300 ms or more).

Audience

©2013-2022 Tencent Cloud. All rights reserved.



1. Anchor/Co-Anchor/Audience can create/enter a TRTC room.
2. Monitor room song changes and load lyrics.
3. Pull the singer's stream.
4. Parse the SEI messages sent by the singer and synchronize the lyrics.

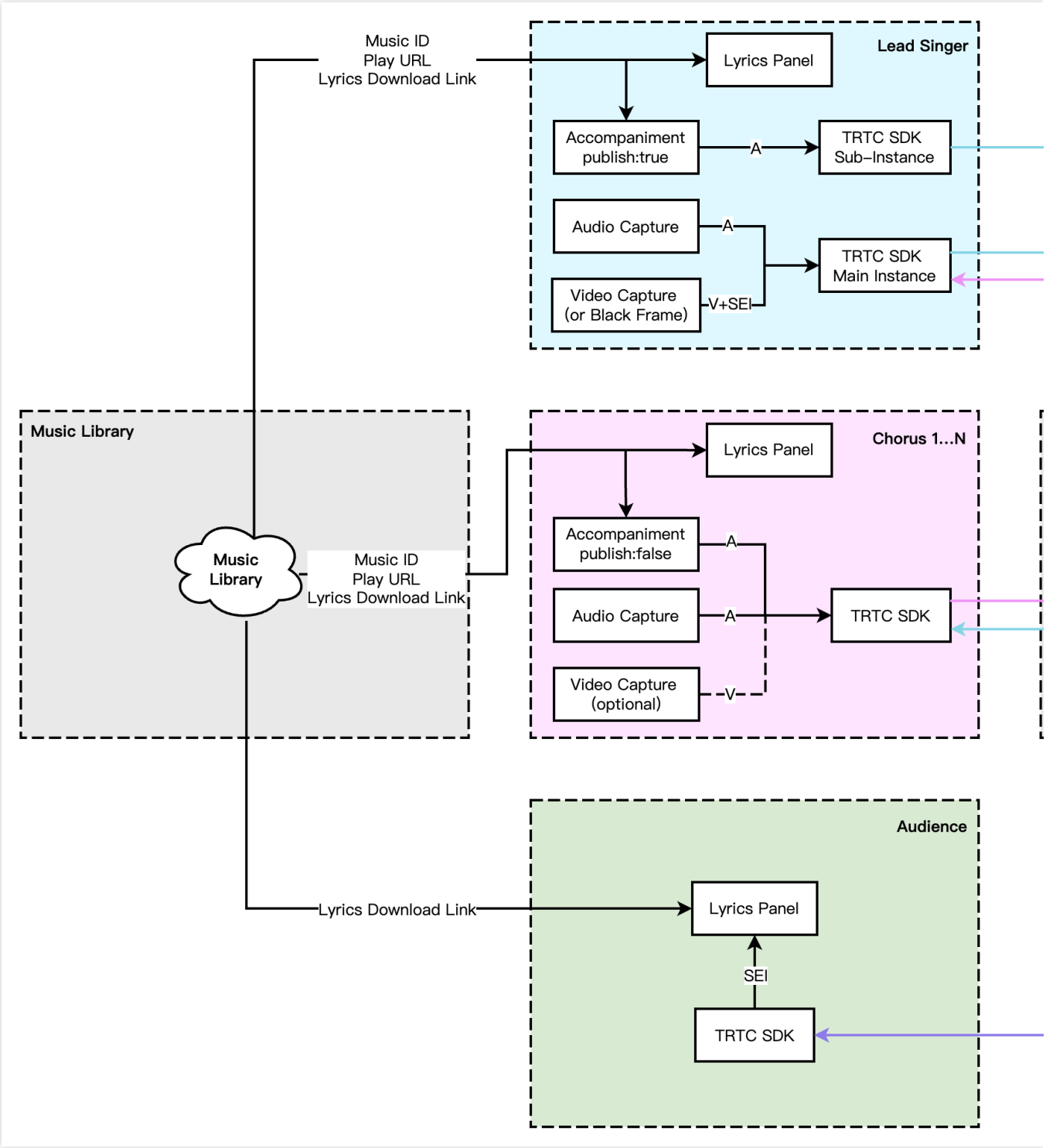
Real-Time Chorus

Real-time chorus refers to playing the song accompaniment simultaneously on all ends based on co-mic, and then performing the chorus on the mic. In a duo pattern, the lead and backing vocals can hear each other; in a multi-person pattern, all choristers can hear each other with almost no delay, achieving true real-time chorus.

Solution Architecture

In terms of media streams, the singers publish/playback streams to each other, while one leading singer uploads accompaniment, and the other singers play accompaniment locally, synchronized via NTP. Additionally, the accompaniment and all singers' voices are mixed through a mixing bot to form a single stream, which is then pushed back to the TRTC room, allowing the audience to hear the synchronized voices from all ends by pulling a single stream, achieving a multi-person chorus effect. The real-time chorus solution architecture is shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



The advantages of this solution are:

It reduces end-to-end latency.

It provides a solution for users to join the chorus midway.

It accurately synchronizes accompaniment, lyrics, and vocals between different ends.

It improves the performance of devices on different ends and the accuracy of local time, and reduces the impact of network environment latency.

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Depending on business needs, you can choose a real-time chorus solution for audio-only or audio and video scenarios. If it is a pure audio scenario, black frames need to be added to send SEI messages for lyric synchronization.

The lead singer needs to use a sub-instance to upstream both the accompaniment and vocals at the same time; other singers only need to pull each other's vocal streams and play accompaniment locally; the audience only needs to pull one mixed stream.

Specific Implementation

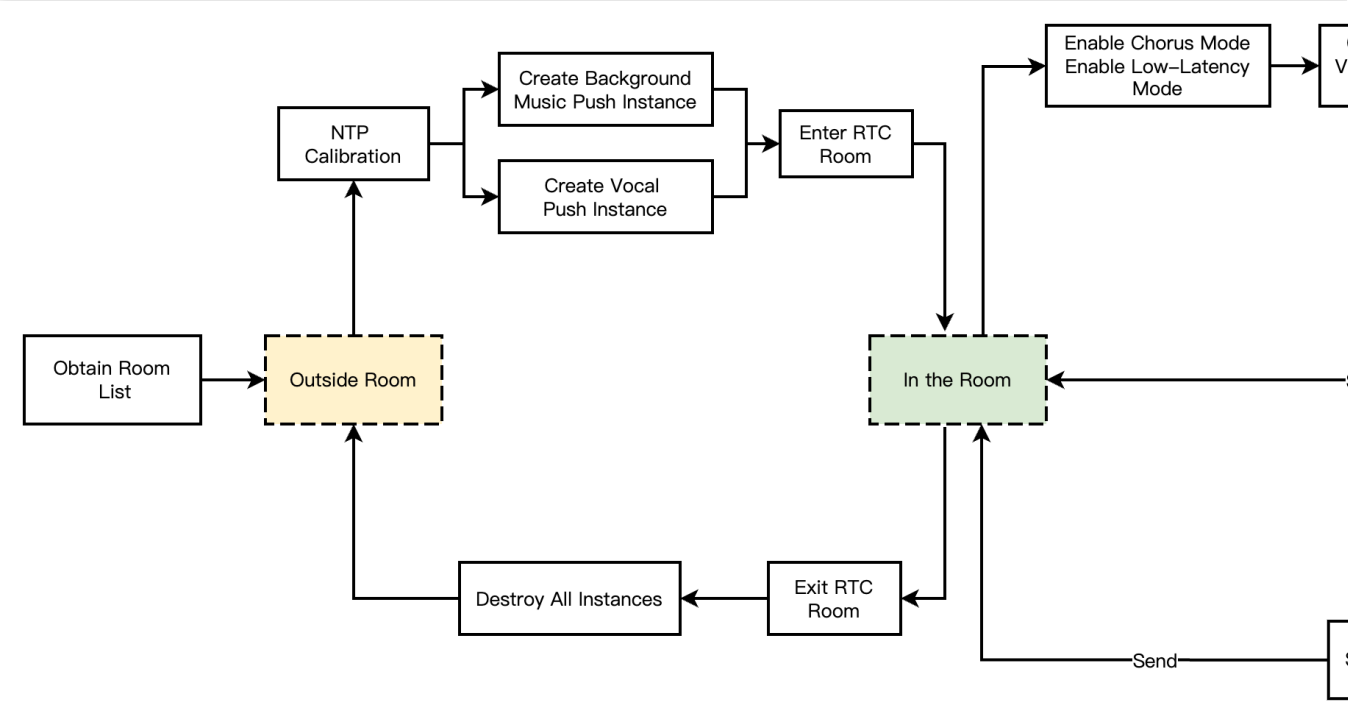
In online Karaoke rooms, different roles have different feature permissions and implementation processes, divided into three roles: lead singer, chorus, and audience, as shown in the table below:

Roles	Description	Differences
Lead Singer	The lead singer is responsible for selecting songs, sending chorus signalings, and sending SEI messages.	The role must be an Anchor. Upstream accompaniment and vocals. Song selection and initiating chorus. Push Back Mixed Stream. Send SEI Message
Chorus	Chorus can receive and process chorus signalings, and participate in the chorus on the seat.	The role must be an Anchor. Upstream Vocals Play Accompaniment Locally Receive Chorus Signals
Audience	After entering the Karaoke room, the audience can pull the stream from the seat and also participate in the chorus on the seat.	The role must be an audience. Downstream mixed stream Receive SEI messages Request to Become an Anchor

Implementation Process

Lead Singer

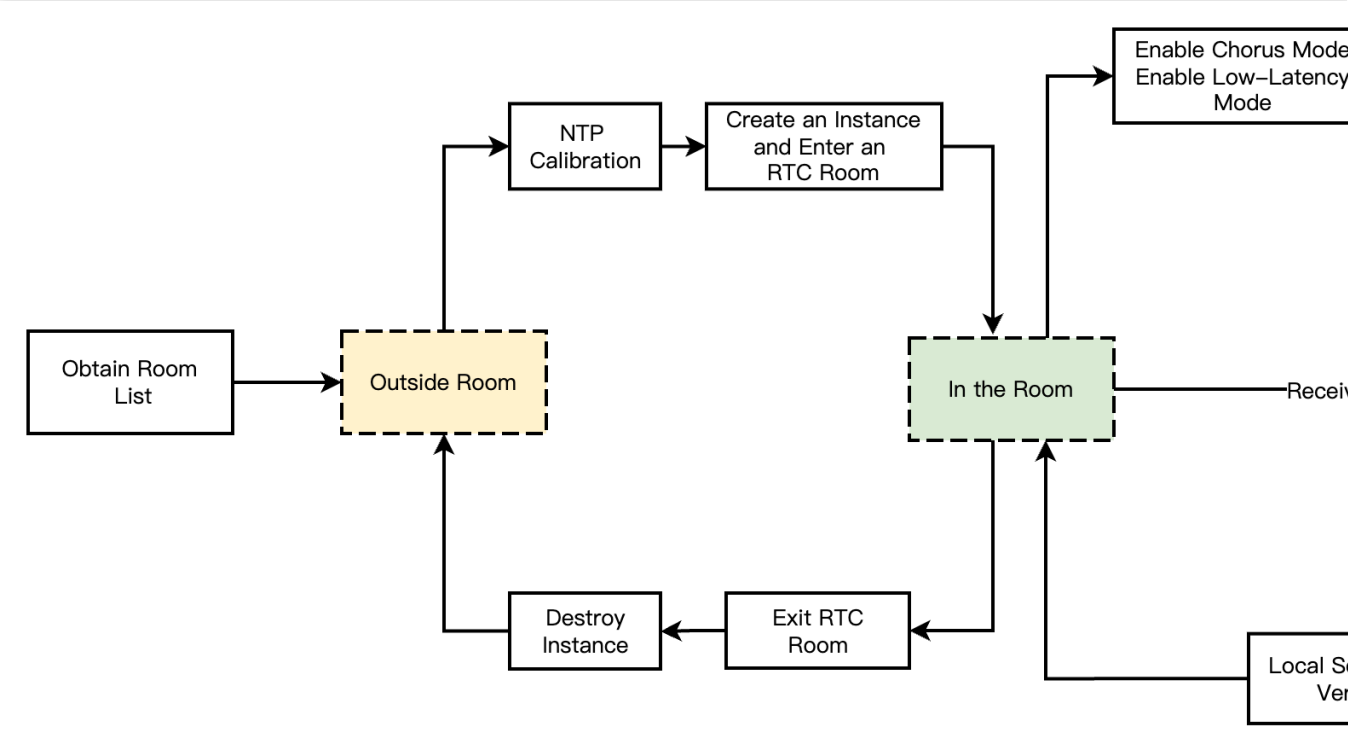
©2013-2022 Tencent Cloud. All rights reserved.



1. The lead singer needs to select songs on-demand and send chorus signalings.
2. The lead singer creates a sub-instance to push vocals and accompaniment and pulls the vocals of other singers.
3. After streaming, the lead singer is responsible for initiating the mixed stream push task.
4. After starting to sing, play the accompaniment and synchronize the lyrics through the playback progress callback.
5. SEI messages need to be sent to synchronize the song progress on the audience end.
6. AI singers need to calibrate the local song playback progress according to NTP.

Chorus

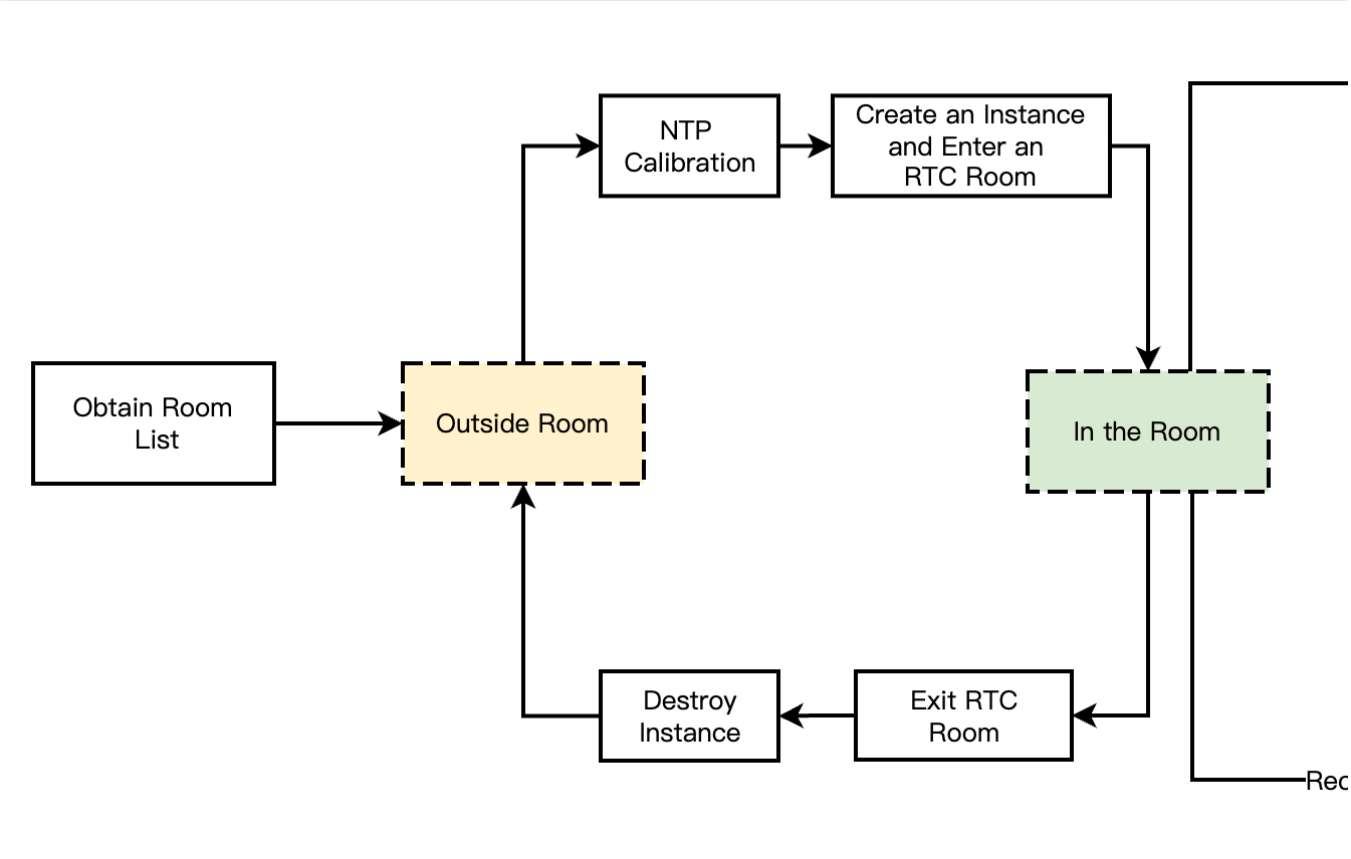
©2013-2022 Tencent Cloud. All rights reserved.



1. The chorus pushes a vocal stream, pulling the vocal stream of chorus users on the seat.
2. The singers need to listen and receive chorus signalings, preloading accompaniment resources.
3. After they start to sing and play accompaniment locally, the singers synchronizes the lyrics through the playback progress callback.
4. AI singers need to calibrate the local song playback progress according to NTP.

Audience

©2013-2022 Tencent Cloud. All rights reserved.



1. Upon entering the TRTC room, the audience receives the mixed chorus stream.
2. Parse the song progress information in the SEI of the mixed stream for lyric synchronization.
3. After the audience becomes a speaker, the mixed stream is stopped and switched to pulling the vocal stream on the seat, and the chorus mode is started.

Scoring Management

Basic Introduction

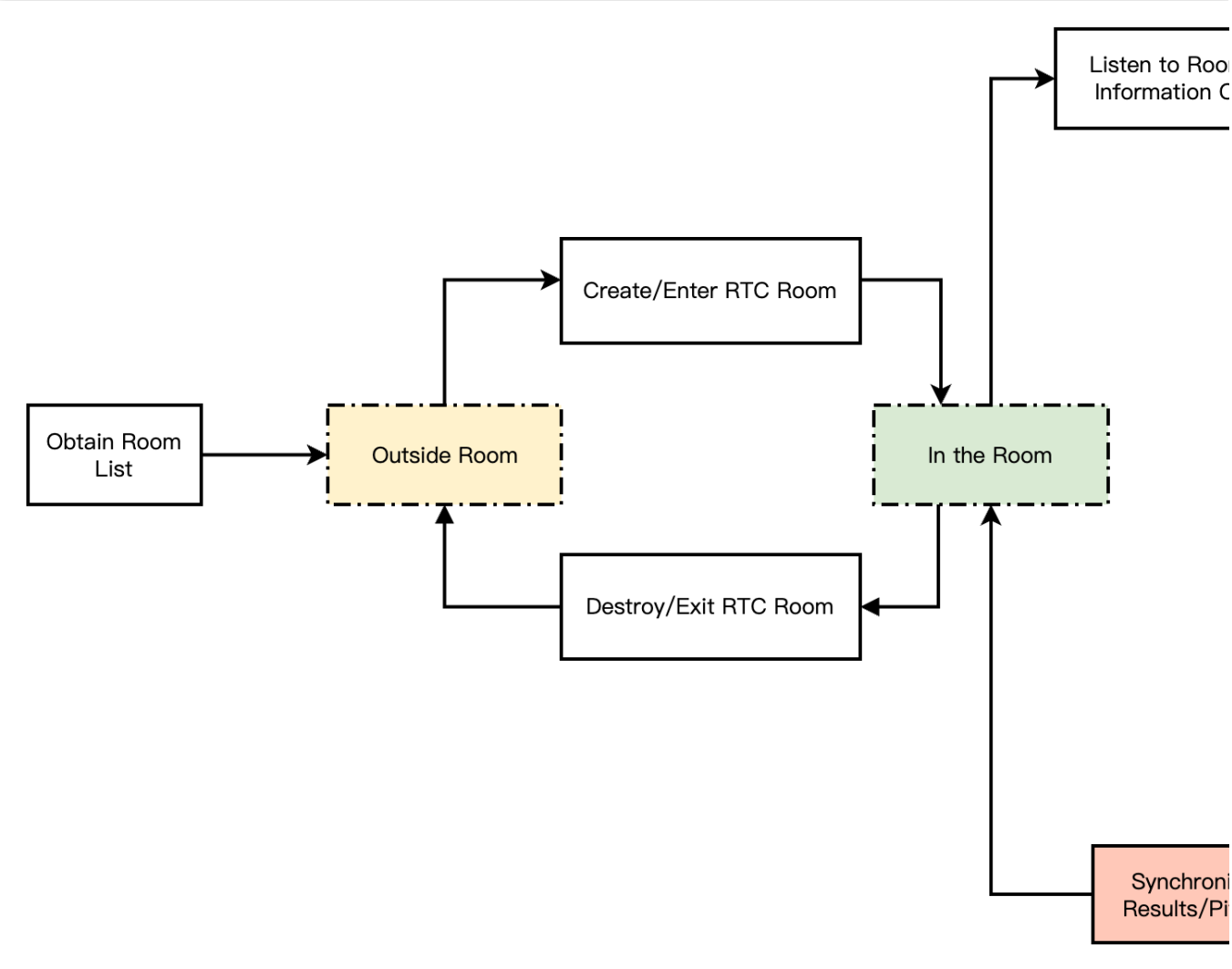
The scoring function is also one of the mainstream play methods in the Karaoke scenario, mainly judging the results of pitch accuracy and sound quality during the actual singing process. It can be used for score comparisons after multi-person singing.

Implementation Process

In the entire scoring management process, singers and audiences have different implementations based on their user roles. Scoring is usually done locally by the singer and synchronized with other people in the room.

Singer

©2013-2022 Tencent Cloud. All rights reserved.



The anchor/audience creates/enters a TRTC room, and automatically becomes a singer after selecting a song.
After the singer selects a song, the song/lyric is downloaded, and then the song is played through the BGM interface.
The vocals captured by TRTC and the progress of the BGM playback are transmitted in real-time to the rating module.
After the rating module produces the data in real time, it synchronizes with everyone in the room through SEI.

Audience

The audience side process is identical to the solo singing audience role action process; you can refer to the audience implementation process.

Key Business Logic

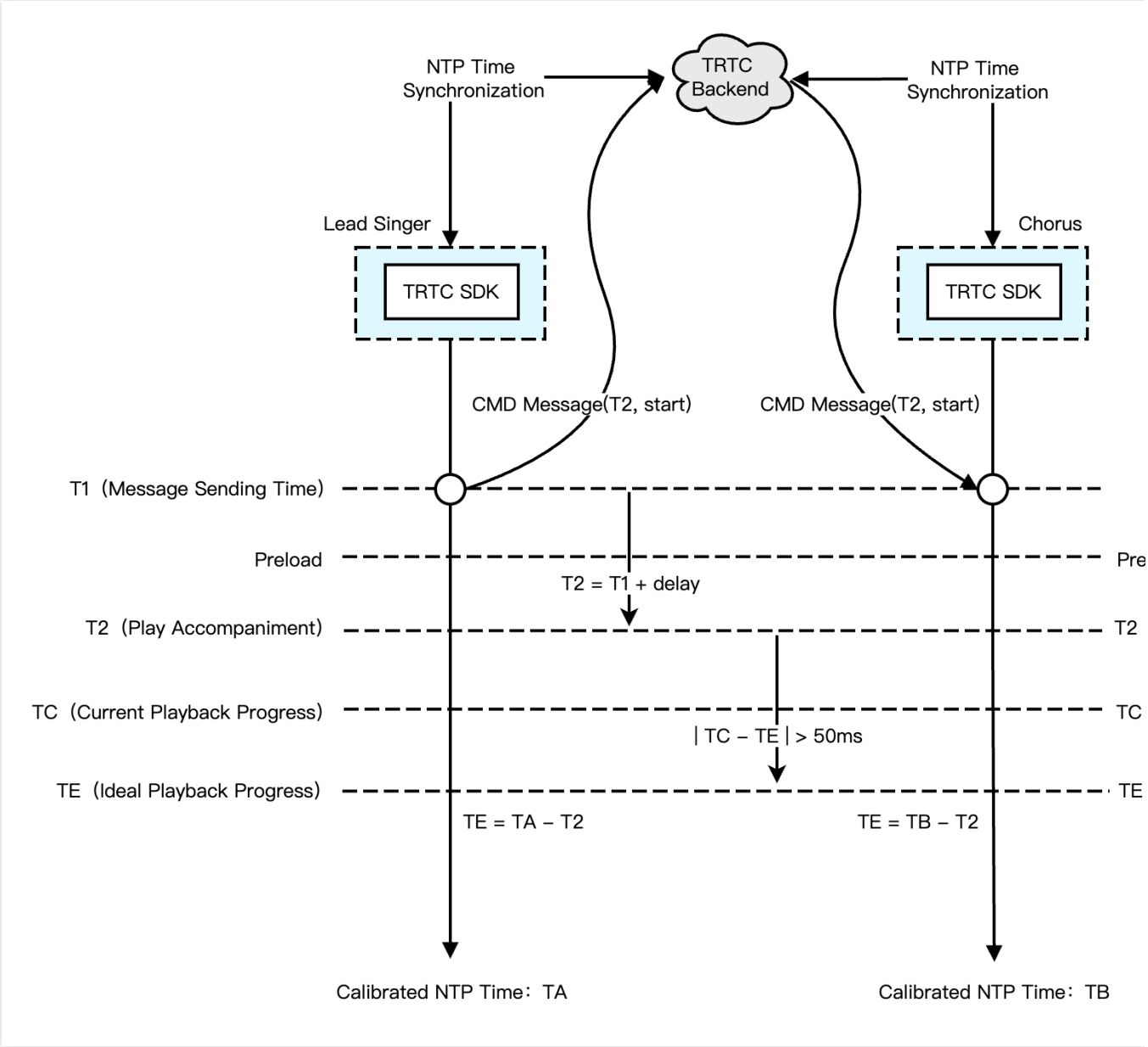
Accompaniment Synchronization Solution

In real-time scenarios, it is necessary to synchronize the accompaniment progress in real-time after starting the performance to avoid increasing end-to-end latency due to accompaniment errors. Synchronizing the accompaniment requires NTP time-based synchronization because the local clocks of different devices are not consistent, resulting in

©2013-2022 Tencent Cloud. All rights reserved.

some errors. Therefore, Tencent Cloud's self-developed NTP service is introduced. Additionally, users who join the ensemble midway also need to synchronize the accompaniment progress. Only after synchronizing the progress can they join in the chorus.

The approach to accompaniment synchronization is: the lead singer convention starts to play the accompaniment at a future point in time (e.g., after a 3-second latency), and other users join in the chorus. All ends' time is based on NTP time, which is synchronized after the TRTC SDK initialization.



The specific process is as follows:

1. All ends calibrate the NTP time, update, and access the latest NTP time T from the TRTC cloud.
2. The lead singer sends the chorus signalings (custom message), agreeing on the chorus start time T2.
3. Preload the accompaniment locally based on T2, and schedule playback.

©2013-2022 Tencent Cloud. All rights reserved.

4. Other chorus participants follow step 3 upon receiving the chorus signalings.
5. During the process, verify the local accompaniment playback progress, and perform seek calibration when the difference between TE and TC exceeds 50 ms.

Note:

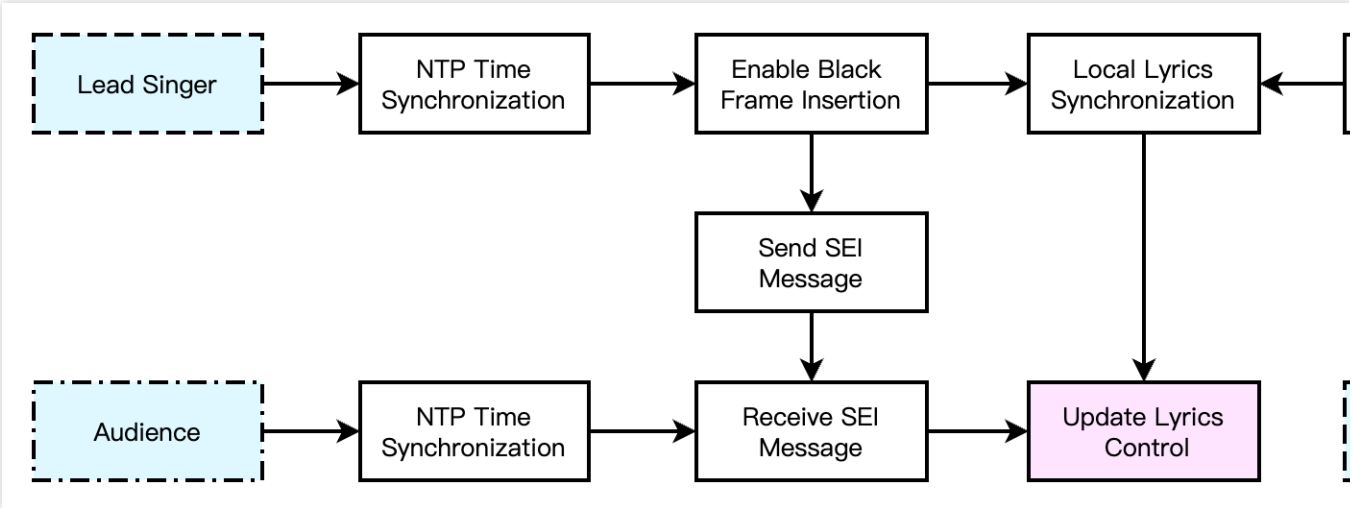
The 50 ms deviation here is a typical value, which can be adjusted appropriately based on the business tolerance, with a recommendation to fluctuate around 50 ms.

Lyrics Synchronization Solution

In the lyrics synchronization solution, the actions of three different roles are as follows:

Lead Singer	Chorus	Audience
NTP Time Synchronization Enable Black Frame Insertion Send SEI Message Local Lyric Synchronization. Update Lyrics Control	NTP Time Synchronization Local Lyric Synchronization. Update Lyrics Control	NTP Time Synchronization Receive SEI messages Update Lyrics Control

Among them, the lead singer and chorus update the lyric progress locally based on the playback progress of the synchronized accompaniment; the audience needs to receive SEI messages sent from the lead vocalist containing the latest lyric progress to update the local lyric progress. The overall process of the lyrics synchronization solution is shown in the following diagram.

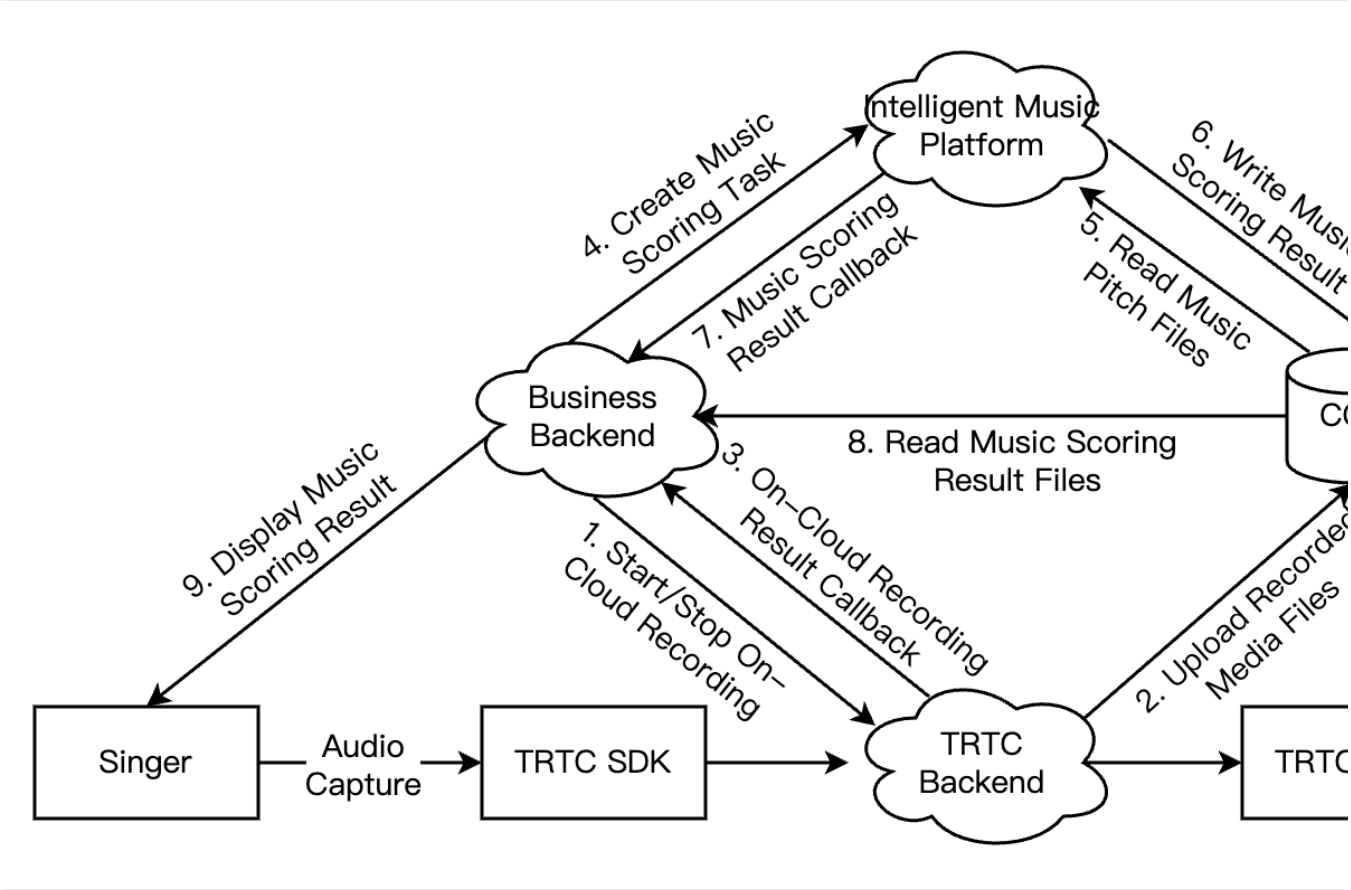


Music Scoring Integration Solution

The accompaniment scoring feature is an indispensable feature in the Online Karaoke scenario. You need to access the standardized audio file and MIDI pitch file of the music resources in advance, and then it is recommended to use

©2013-2022 Tencent Cloud. All rights reserved.

the music scoring feature of the Intelligent Music Platform to score the singer's voice from TRTC on-cloud recording. The overall process of the Music Scoring Integration Solution is shown in the following diagram.



1. The business backend starts the on-cloud recording task when the singer begins to sing, and stops the on-cloud recording task when the singer finishes singing.
2. The TRTC backend will upload the recorded singing clip media file to the COS bucket specified when initiating the recording task.
3. After the recording file is uploaded, the TRTC backend will callback the on-cloud recording results to the business backend.
4. The business backend uses the Intelligent Music Platform's music scoring feature to create a music scoring task.
5. The Intelligent Music Platform reads the singing clip and the standard pitch file from the COS bucket for scoring.
6. The Intelligent Music Platform will write the JSON file containing the scoring results into the specified path in COS.
7. After the music scoring is completed, the Intelligent Music Platform will call back the music scoring results to the business backend.
8. The business backend reads the music scoring results JSON file from COS according to the callback path.
9. The business backend analyzes the music scoring results and displays the scoring results on the singer's App.

Note:

©2013-2022 Tencent Cloud. All rights reserved.

The input file format for the Intelligent Music Platform's music scoring should use MP3 or WAV. If the on-cloud recording file format is HLS or AAC, audio transcoding is required.

Best Practices for Audio Tuning Strategies

In the entire Karaoke scenario, the audio quality is mainly affected by parameters such as sampling rate, number of channels, bitrate, and 3A. According to different room scenarios, we recommend various audio parameters and volume mixing schemes, as well as commonly used vocal and accompaniment synchronization alignment solutions.

1. **Best Parameter Configuration for Different Scenarios**

Room Scenario	Entry Mode	Audio Quality	Volume Type	Hidden Interface
Solo Singing	Video or CDN Push Requirements: LIVE Audio-only or Pure RTC Requirements: VOICE_CHATROOM	MUSIC	TRTCSystemVolumeTypeMedia	enableBlackStream
Real-Time Chorus	Video or CDN Push Requirements: LIVE Audio-only or Pure RTC Requirements: VOICE_CHATROOM	MUSIC	TRTCSystemVolumeTypeMedia	enableBlackStream enableChorus setLowLatencyModeEn
Chat and Listen to Music	VOICE_CHATROOM	DEFAULT	TRTCSystemVolumeTypeAuto	No

2. **Best Volume Ratio for Different Scenarios**

The TRTC SDK has initial default values for voice collection and music playback. If in the default situation, there is suppression of voice by accompaniment in the live streaming room, leading to the voice being masked by music, you can adjust the voice and music volume ratio according to the recommended values in the table below.

Room Scenario	Recommended Configuration for Voice/Music/Sound Effects
Solo Singing	Voice Capture Volume: 60 Music Playback Volume: 50 Enable Reverb Effect: Yes
Real-Time Chorus	
Chat and Listen to Music	Vocal Capture Volume: 100 Music Playback Volume: 30 Enable Reverb Effect: No

©2013-2022 Tencent Cloud. All rights reserved.

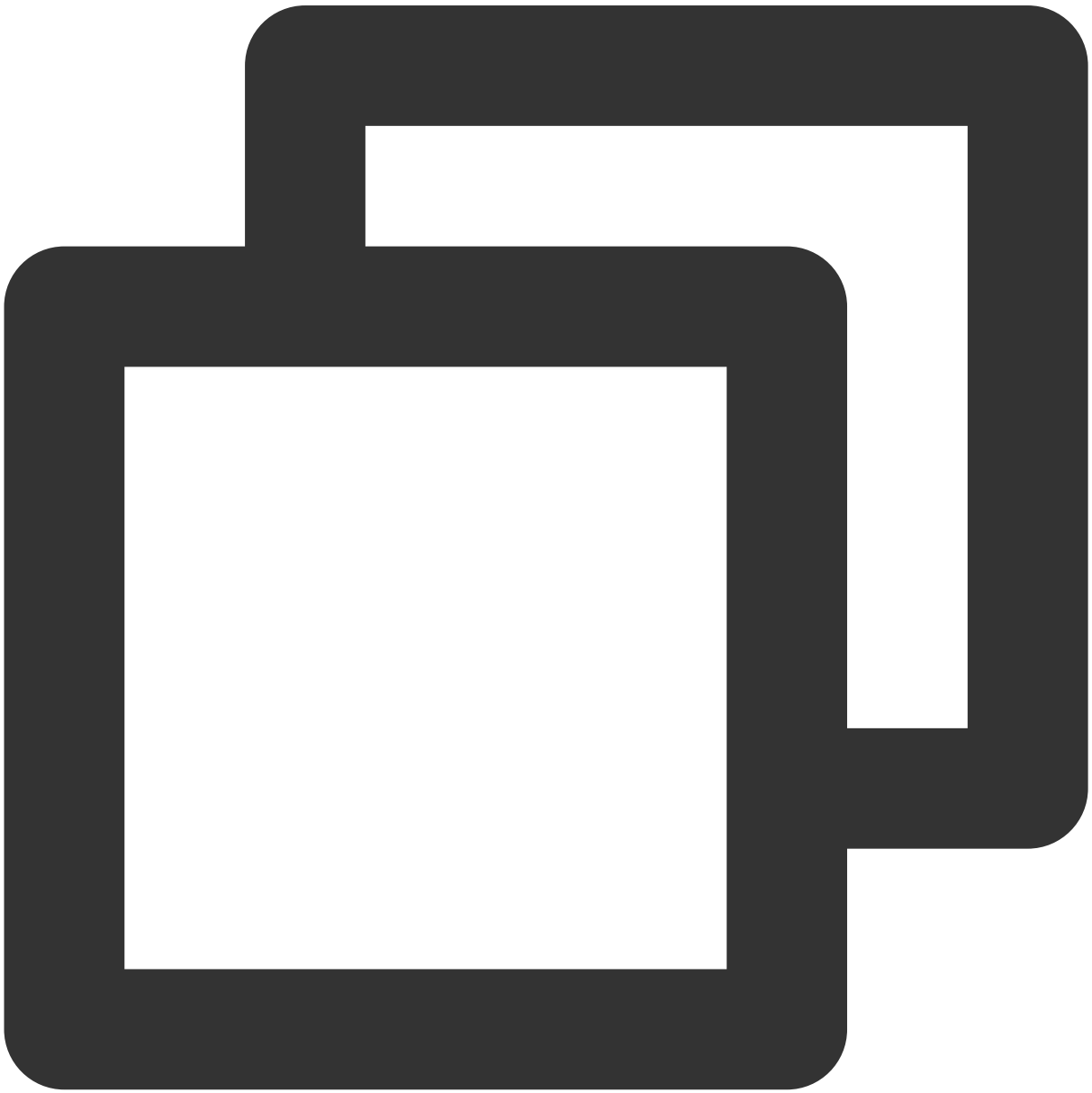
3. **Voice and Accompaniment Synchronization Alignment**

Due to the JitterBuffer for local vocal capture, the JitterBuffer for song playback mixing, and the GAP that exists from when the human ear receives the accompaniment to when singing begins if the singer sings entirely in sync with the lyrics and accompaniment, the remote audience may perceive a certain latency and misalignment between the vocal, accompaniment, and lyrics. This issue can be improved through the following two methods.

Enable Chorus Mode

Android

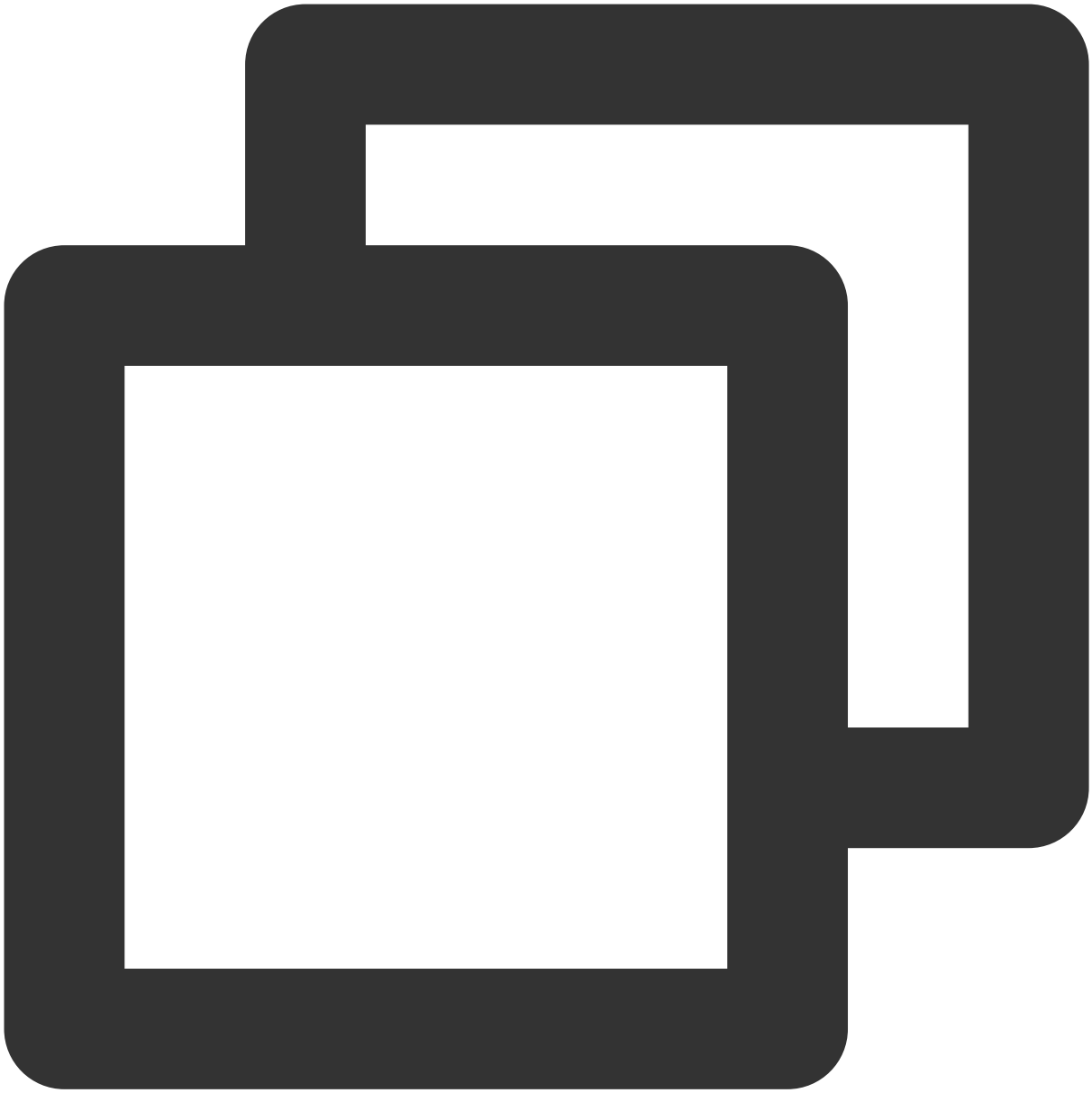
iOS



©2013-2022 Tencent Cloud. All rights reserved.

```
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("api", "enableChorus");
    JSONObject params = new JSONObject();
    params.put("enable", true);
    params.put("audioSource", 0);
    jsonObject.put("params", params);
    mTRTCCloud.callExperimentalAPI(String.format(Locale.ENGLISH, jsonObject.toStrin
} catch (JSONException e) {
    e.printStackTrace();
}
```

©2013-2022 Tencent Cloud. All rights reserved.



```
NSDictionary *jsonDic = @{
                            @"api": @"enableChorus",
                            @"params": @{
                                        @"enable": @(YES),
                                        @"audioSource": @(0)
                                      }
                        };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWr
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8Strin
[trtcCloud callExperimentalAPI:jsonString];
```

©2013-2022 Tencent Cloud. All rights reserved.

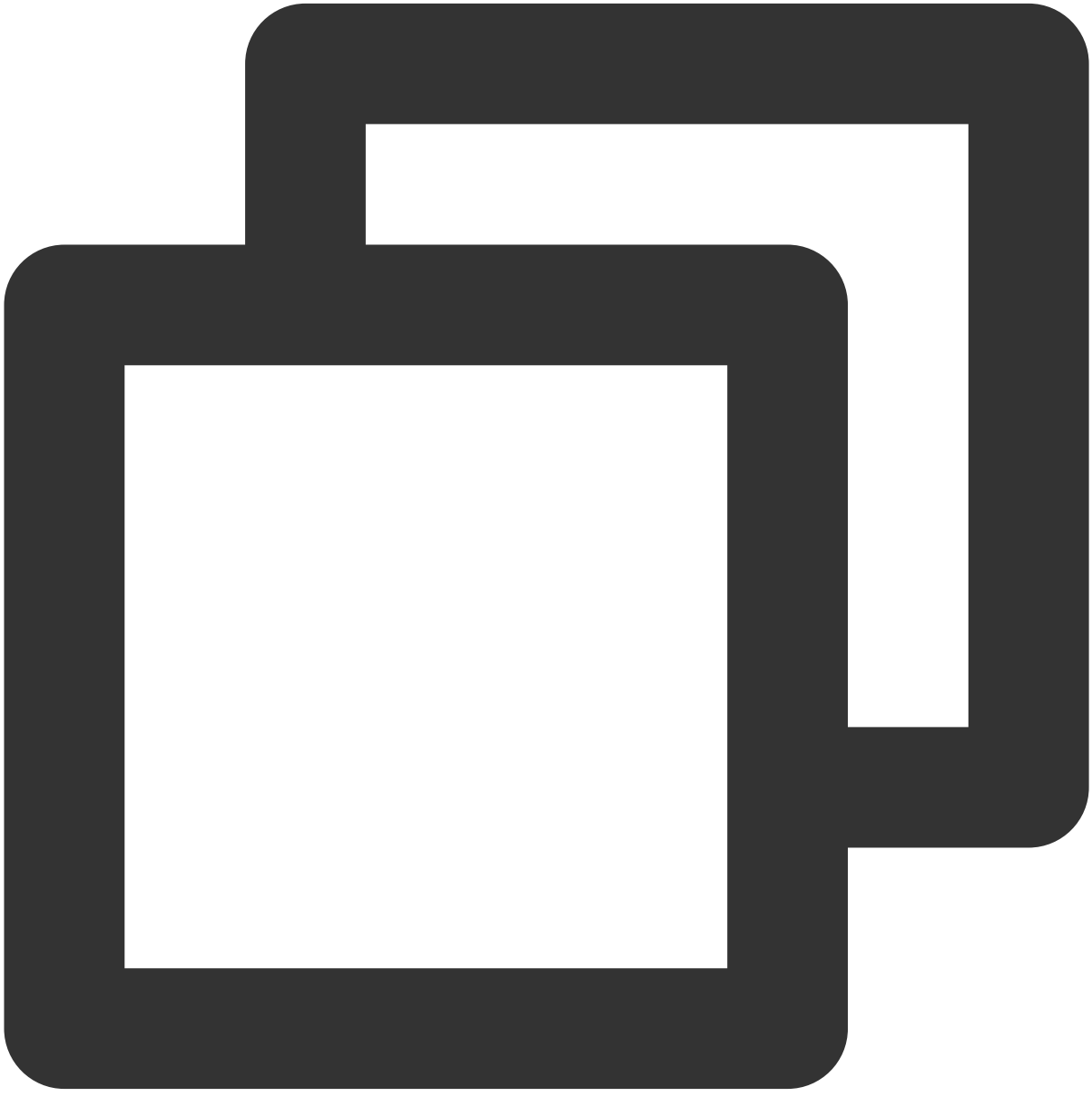
Note:

audioSource: 0 (Vocal), audioSource: 1 (Accompaniment); If using single instance streaming, set audioSource to 0 for all streams.

Enable Low Latency Mode

Android

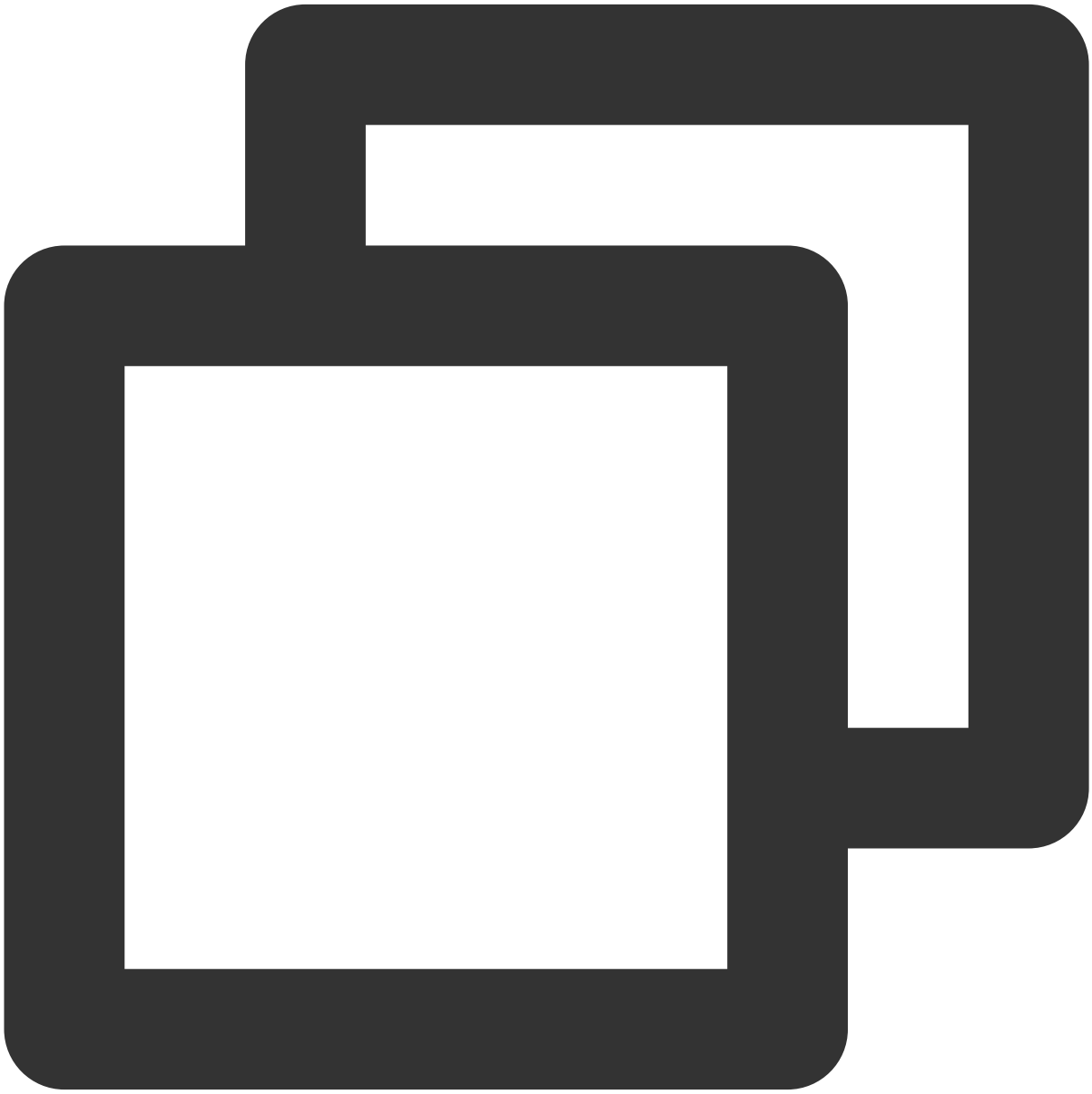
iOS



```
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("api", "setLowLatencyModeEnabled");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    JSONObject params = new JSONObject();
    params.put("enable", true);
    jsonObject.put("params", params);
    mTRTCCloud.callExperimentalAPI(String.format(Locale.ENGLISH, jsonObject.toStrin
} catch (JSONException e) {
    e.printStackTrace();
}
```



```
NSDictionary *jsonDic = @{
                            @"api": @"setLowLatencyModeEnabled",
                            @"params": @{
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                @"enable": @(1)
                                            }
                                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWr
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8Strin
[trtcCloud callExperimentalAPI:jsonString];
```

Scenario Gameplay

Solo Singing

After becoming a speaker, the audience can select songs and wait in line. After the song starts to play, they can sing solo. This game mode is relatively simple and can be achieved using TRTC single-instance mixing and streaming.

Real-Time Chorus

After becoming a speaker, the audience sings a song with the lead singer at the same time. This game mode is relatively complex. The lead singer side needs to use TRTC dual-instance streaming, and all ends also need to pay attention to accompaniment synchronization and lyric synchronization.

Mass Singing Competition

Users can choose song rooms of different categories according to their preferences. The room will randomly play music clips, and users in the room can grab the microphone at any time to sing the music clips.

Segmental Singing

The same song is divided into segments and assigned to different speakers. After the lead singer sings a segment, other speakers sing the assigned music clips respectively.

Cross-Room Singing Competition

Anchors from different rooms sing, and the audience in their respective rooms helps their anchors. In addition to the Karaoke scenario, this game mode also involves TRTC cross-room competition, and it is necessary to pay attention to the subscription logic of audio and video streams in different rooms.

Supporting Products for the Solution

System Level	Product Name	Application Scenarios
Access	Tencent Real-	Provides a low-latency, high-quality real-time interactive live streaming

©2013-2022 Tencent Cloud. All rights reserved.

Layer	Time Communication (TRTC)	solution for multiple people's audio, which is the basic foundation for online Karaoke scenarios.
Access Layer	Instant Messaging (IM)	Provides room management and seat management capabilities based on group features, enables the sending and receiving of rich media messages such as live streaming room-wide messaging, public screen messages, as well as custom signaling and other communication needs.
Access Layer	Intelligent Music Platform	Based on the self-developed music understanding technology of Tencent Media Lab, it helps users to deeply understand, analyze and create music, and provides capabilities such as lyric recognition, intelligent composition, song recognition, and music scoring.

©2013-2022 Tencent Cloud. All rights reserved.

Quick Integration Guide
Android

Last updated：2024-07-18 14:26:14

Business Process

This section summarizes some common business processes in online karaoke, helping you better understand the implementation process of the entire scenario.

Song request process

Solo singing process

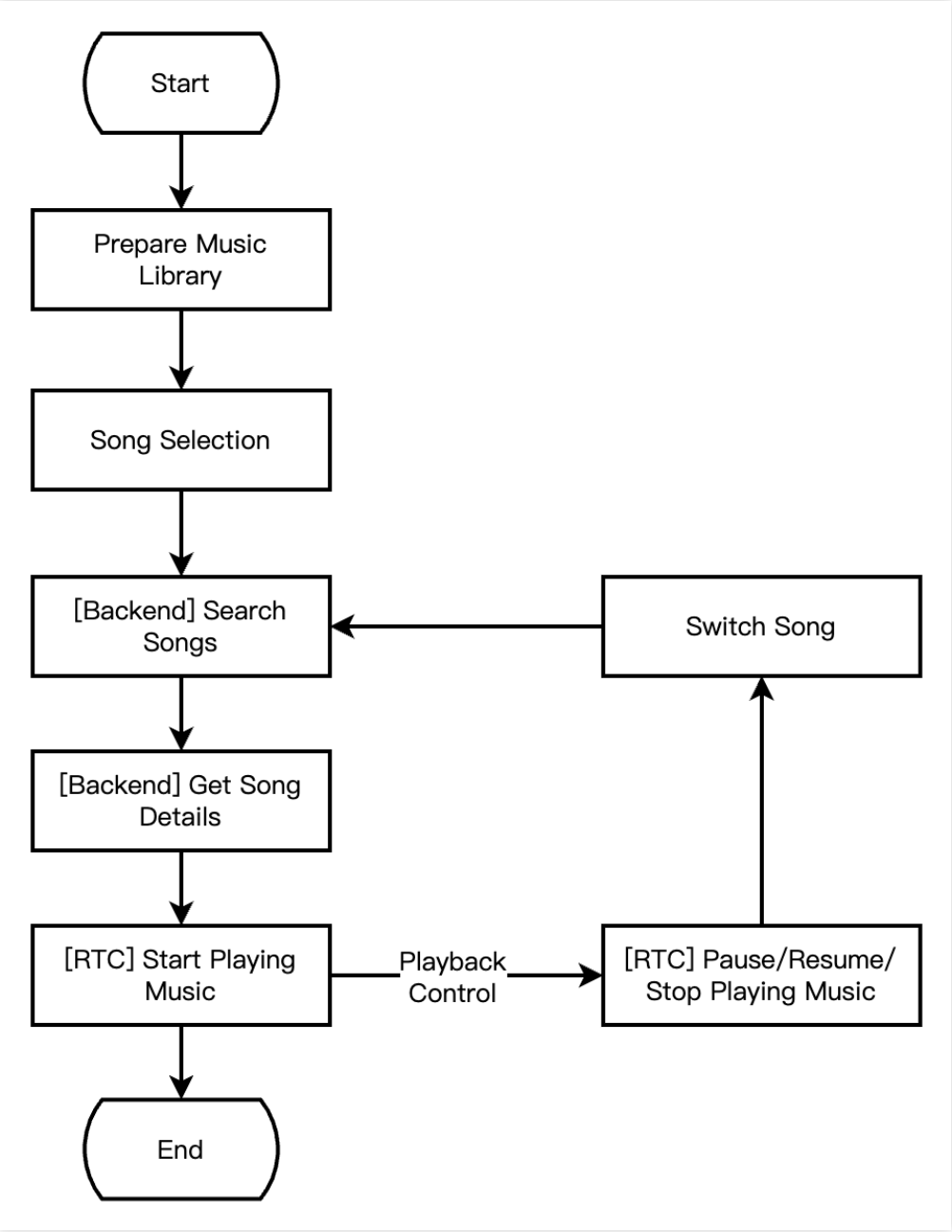
Lead singer process

Chorus process

Audience process

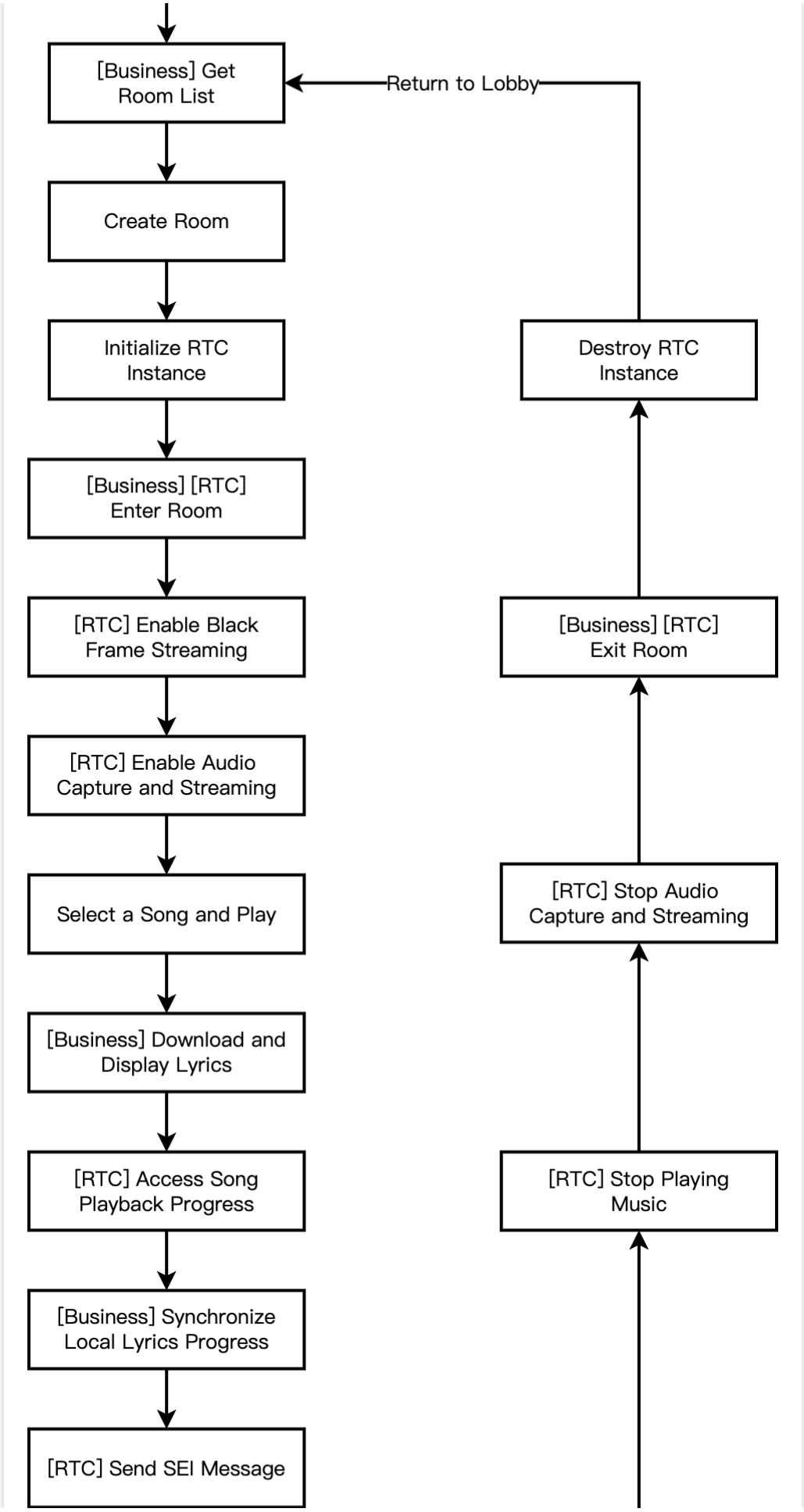
The following figure shows the process of requesting songs from a music repository on the business side and playing them using the TRTC SDK.

©2013-2022 Tencent Cloud. All rights reserved.

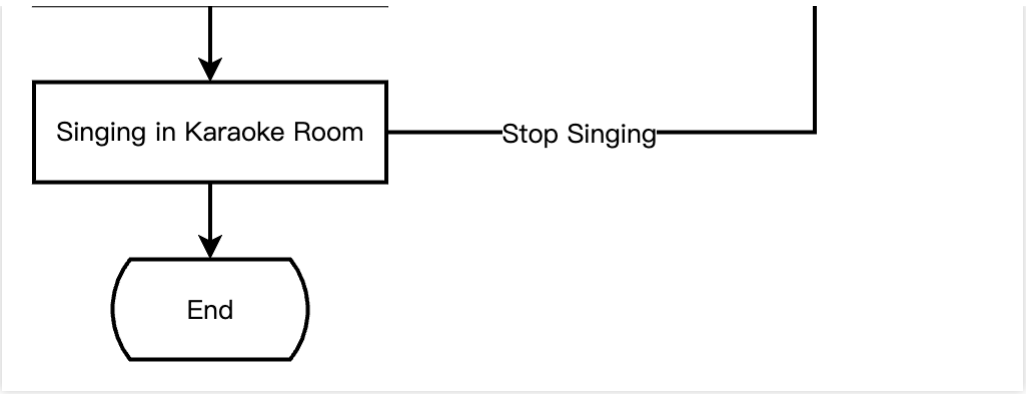


The following figure shows the process of a solo singing turn-taking game, that is, the performer enters a room to perform, stops performing, and exits the room.

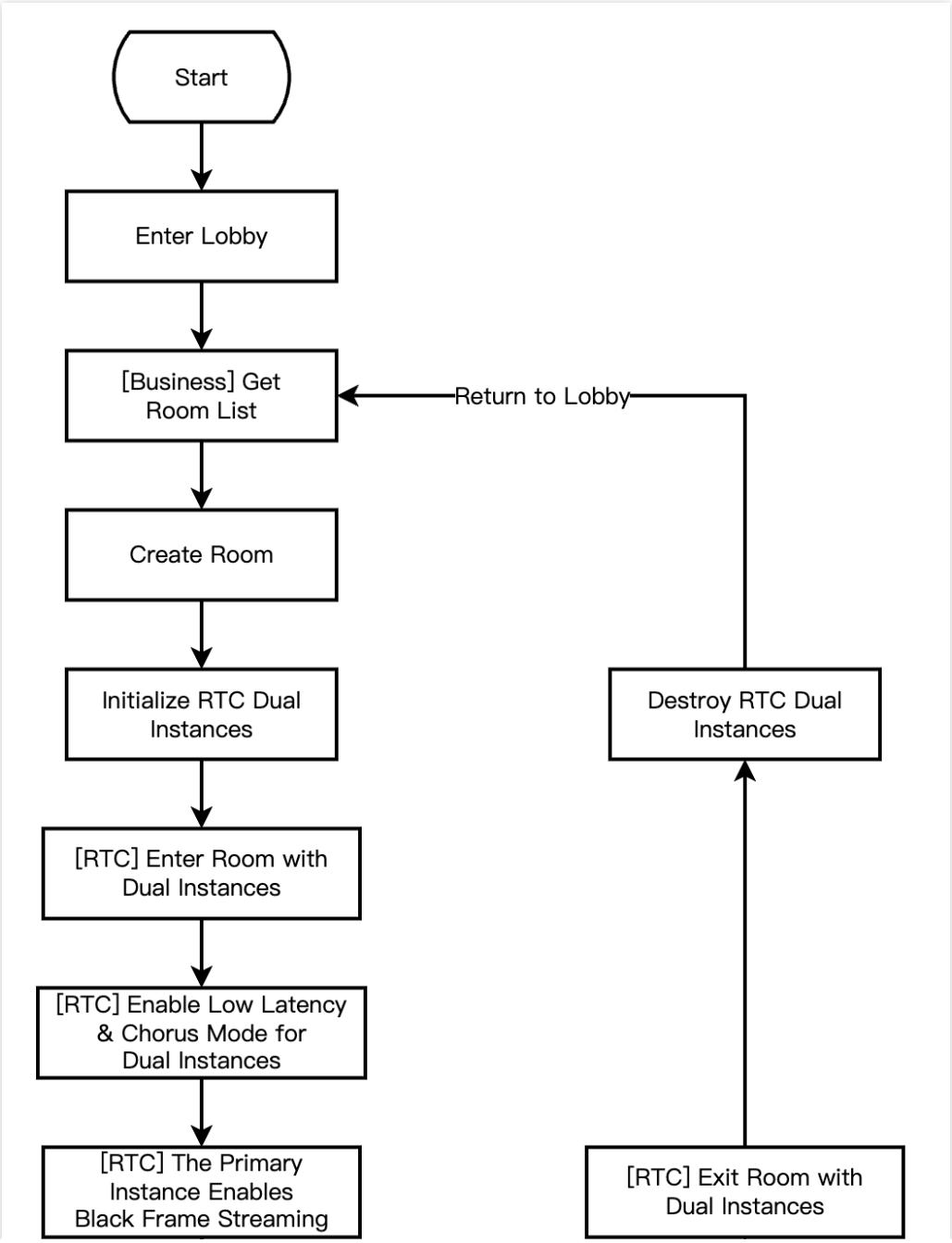
©2013-2022 Tencent Cloud. All rights reserved.



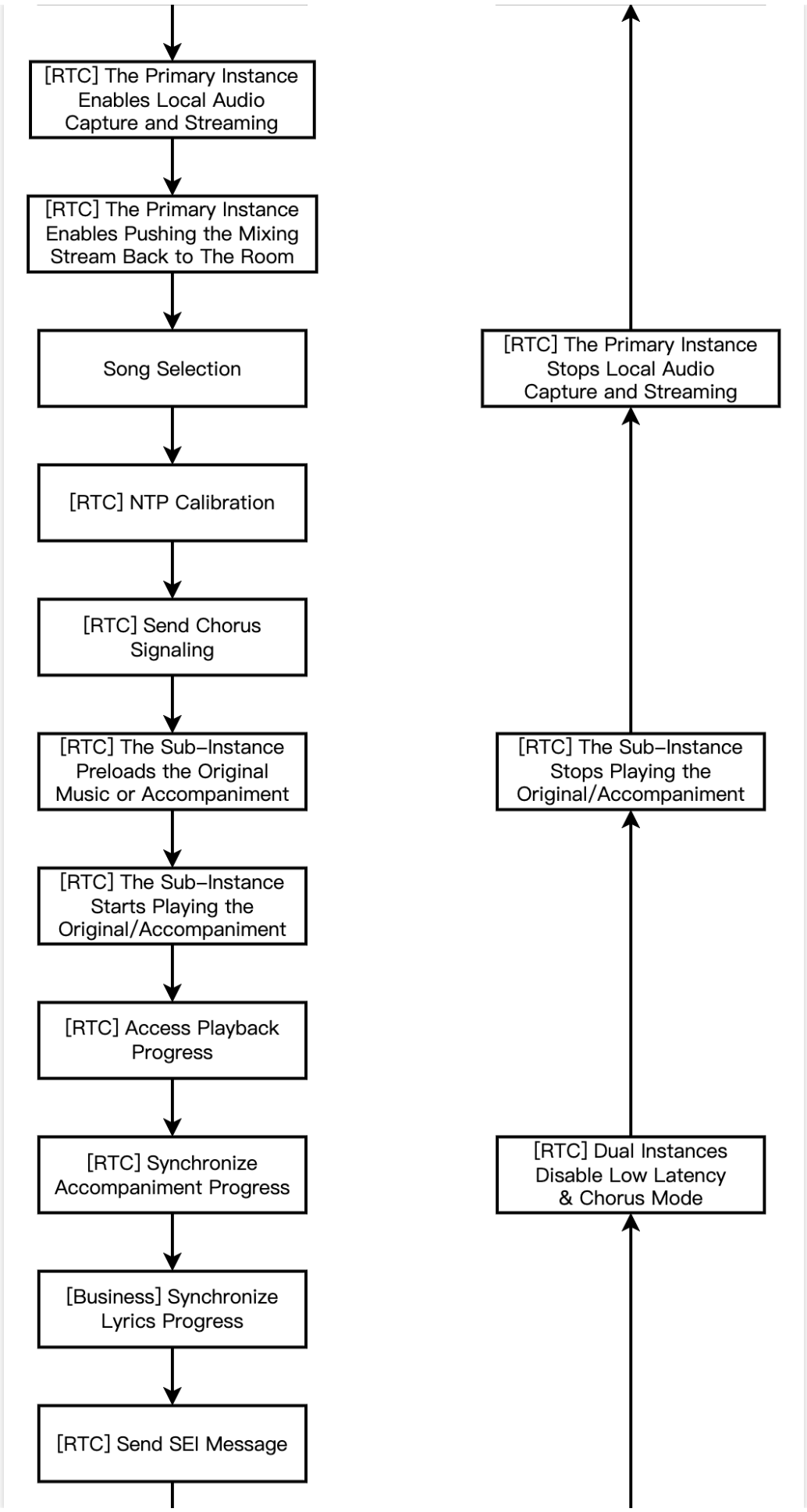
©2013-2022 Tencent Cloud. All rights reserved.



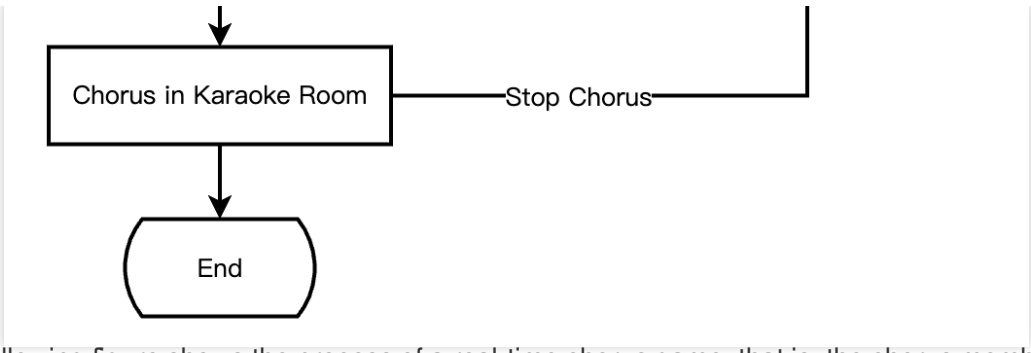
The following figure shows the process of a real-time chorus game, that is, the lead singer initiates a chorus, stops the chorus, and exits the room.



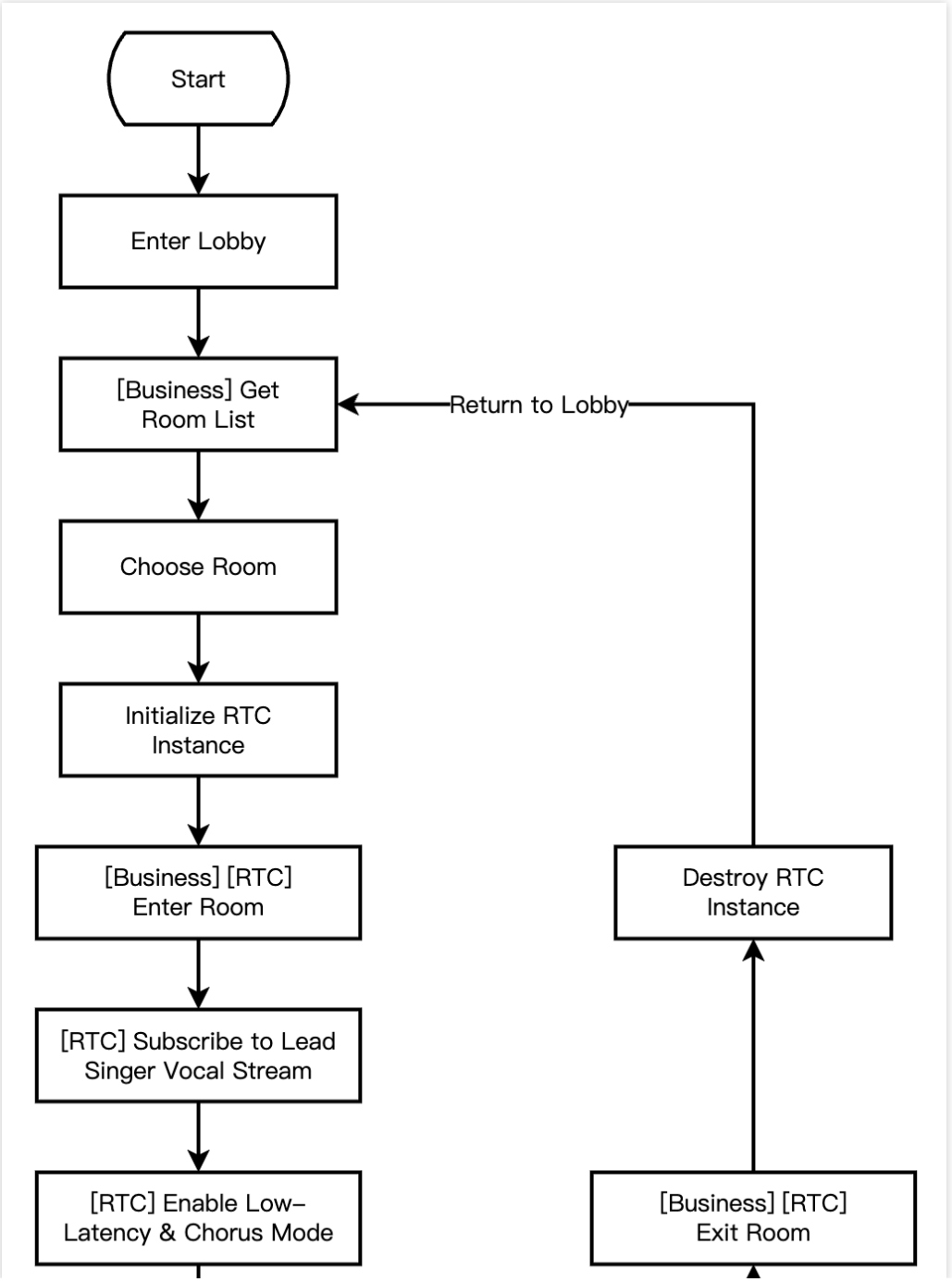
©2013-2022 Tencent Cloud. All rights reserved.



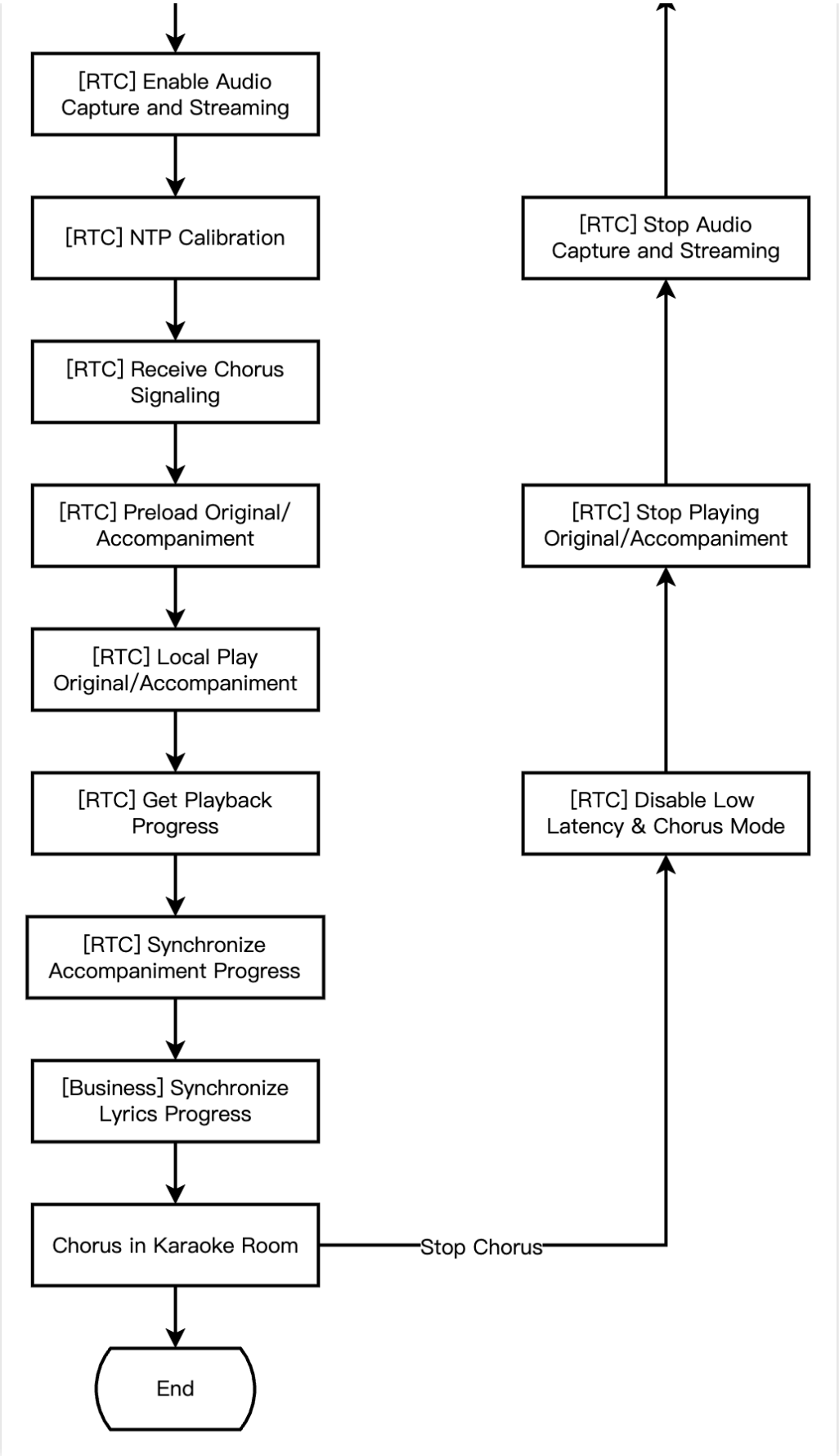
©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the process of a real-time chorus game, that is, the chorus members join the chorus, stop the chorus, and exit the room.

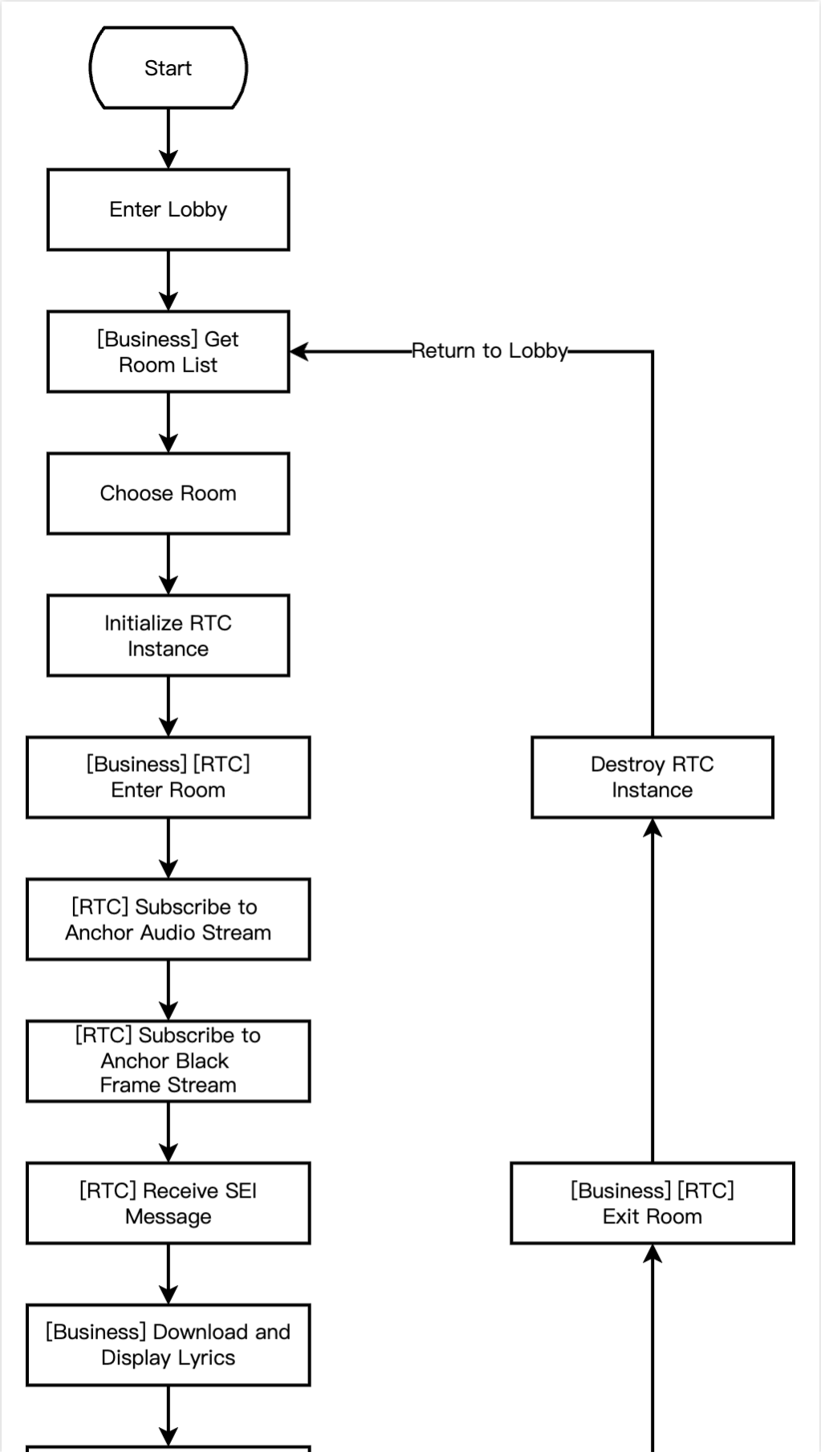


©2013-2022 Tencent Cloud. All rights reserved.

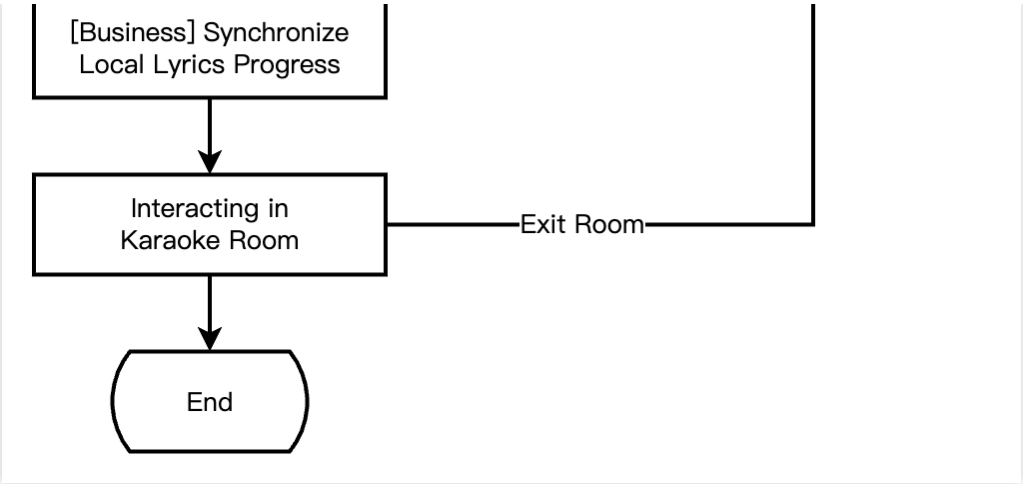


The following figure shows the process of an online karaoke scenario, that is, the audience enters the room to listen to songs and synchronizes lyrics.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



Integration Preparation

Step 1. Activating the service.

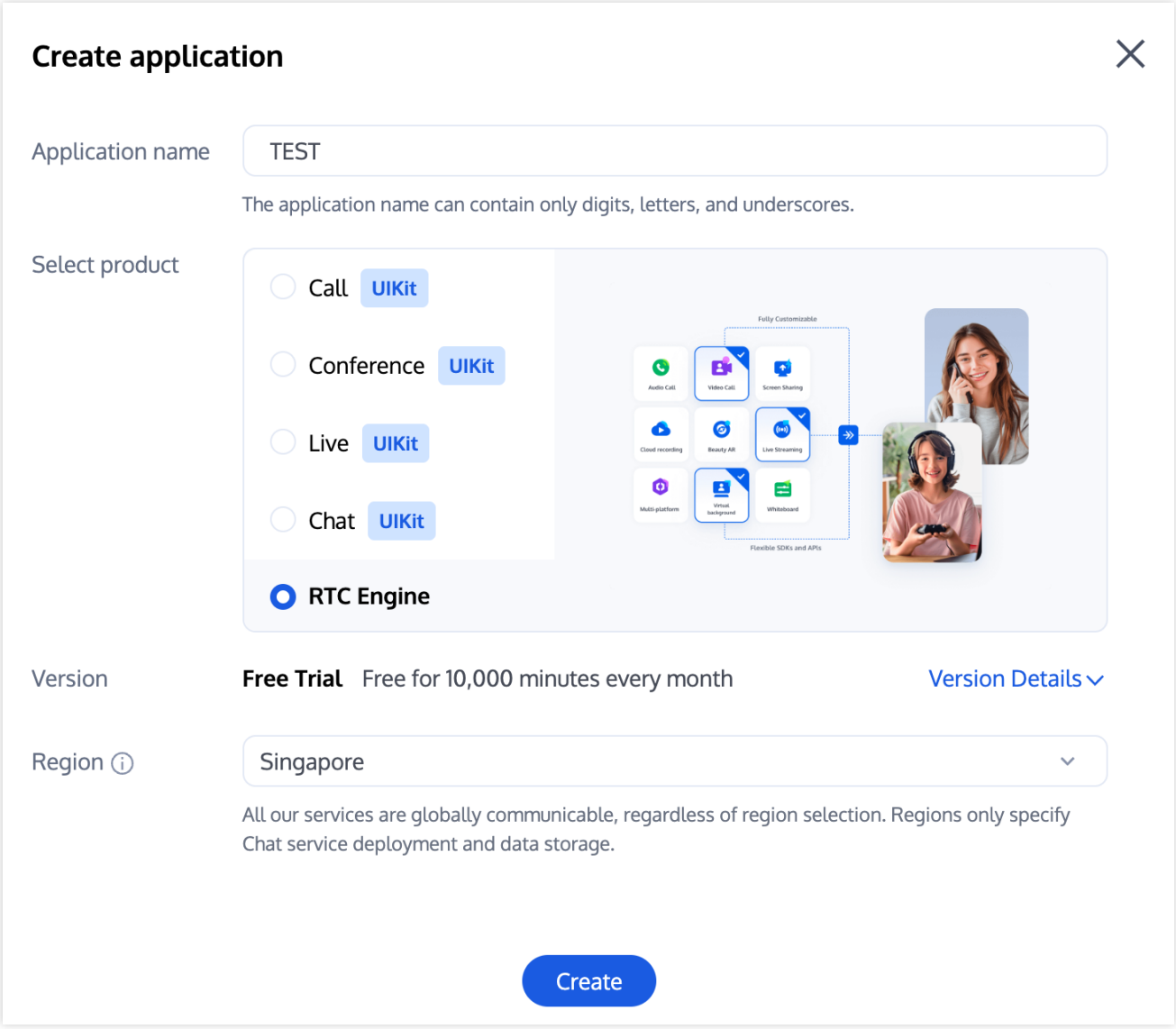
The online karaoke scenarios usually require two paid PaaS services from Tencent Cloud: Tencent Real-Time Communication (TRTC) and Intelligent Music Solution for construction. TRTC is responsible for providing real-time audio and video interaction capabilities. Intelligent Music Solution is responsible for providing lyric recognition, smart composition, music recognition, and music scoring capabilities.

Activate TRTC service.

Activate the Intelligent Music service.

1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



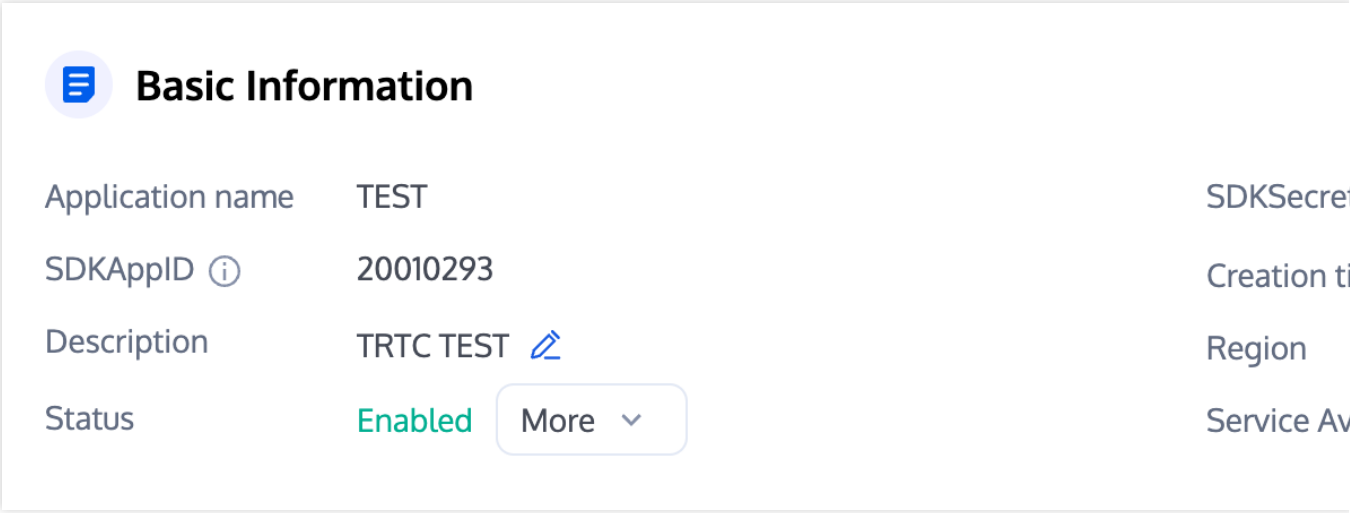
Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

2. After an application is created, you can see the basic information of the application in the Application Management - Application Overview section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.

©2013-2022 Tencent Cloud. All rights reserved.



Preparation

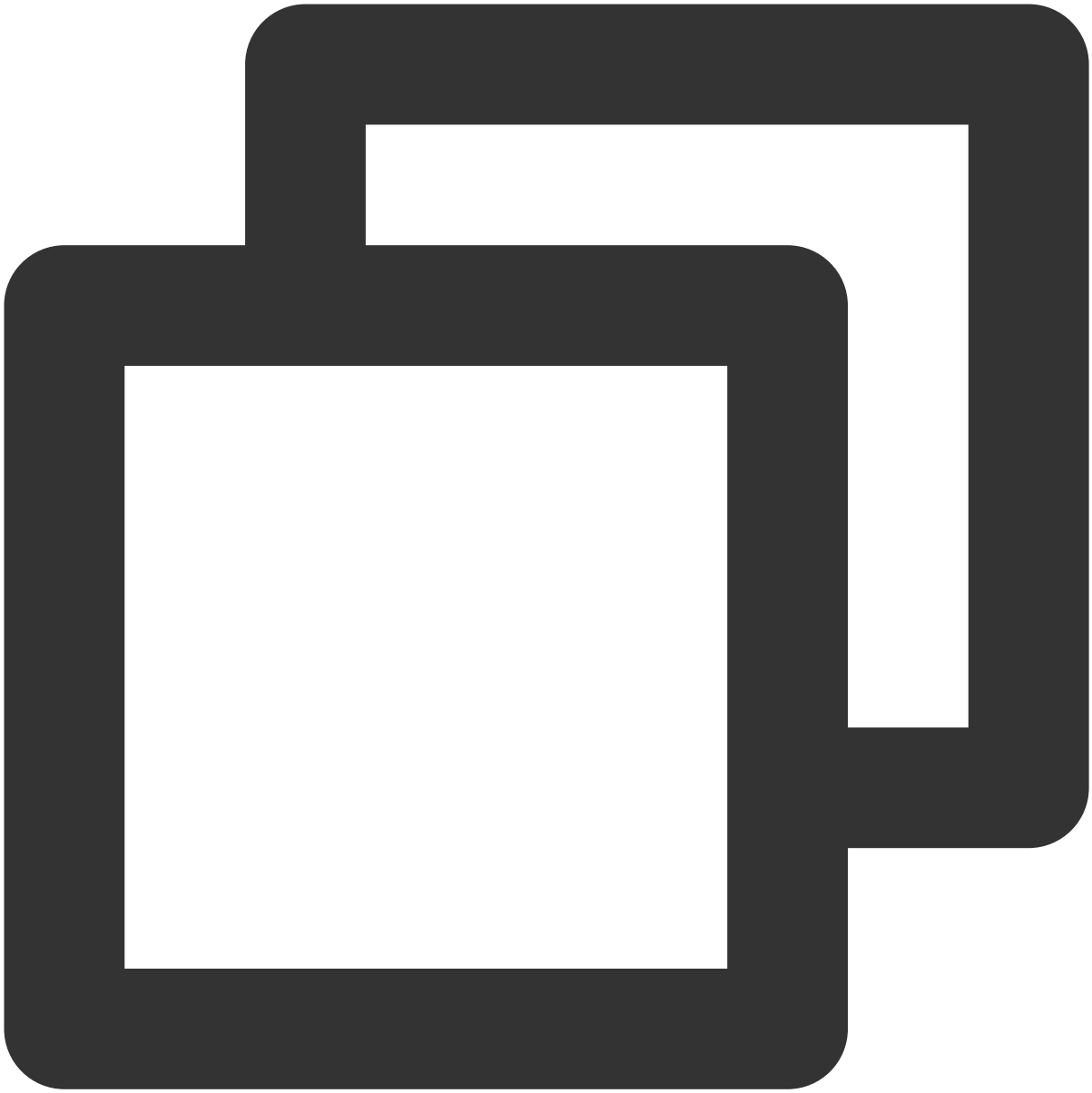
1. Go to the Purchase Page to activate the music service, and choose the appropriate features such as music scoring to activate.
2. Create an AK/SK Key Pair in CAM (namely, a programmable access user that does not require log-in or any user permissions).
3. Create a COS Bucket, and in the COS Bucket Management interface, authorize the read and write permissions of the COS Bucket to the created programmable access user.
4. Prepare the parameters.

operateUin: Tencent Cloud sub-user's account ID.
cosConfig: COS related parameters.
secretId: Bucket's secretId.
secretKey: Bucket's secretKey.
bucket: Bucket's name.
region: Bucket's region, for example, ap-guangzhou.

Activation and registration.

After the preparation is completed, proceed with registration activation by initiating a request, with an estimated wait time of about 2 minutes.
Initiate request.
Request result:

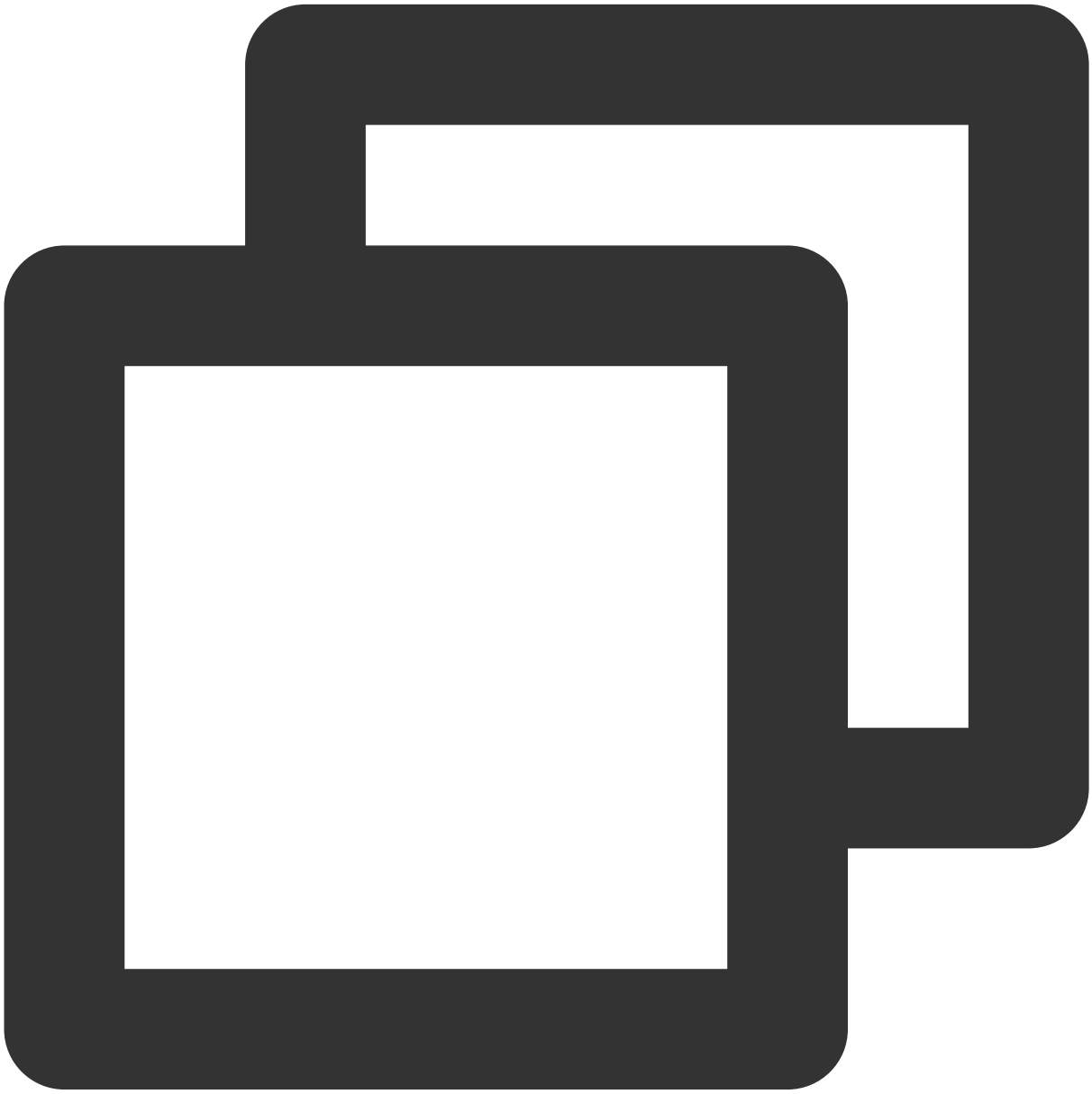
©2013-2022 Tencent Cloud. All rights reserved.



```
curl -X POST \\
  http://service-mqk0mc83-1257411467.bj.apigw.tencentcs.com/release/register \\
  -H 'Content-Type: application/json' \\
  -H 'Cache-control: no-cache' \\
  -d '{
    "requestId": "test-regisiter-service",
    "action": "Register",
    "registerRequest": {
      "operateUin": <operateUin>,
      "userName": <customedName>,
      "cosConfig": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "secretId": <CosConfig.secretId>,
        "secretKey": <CosConfig.secretKey>,
        "bucket": <CosConfig.bucket>,
        "region": <CosConfig.region>
      }
    }
  }'
```



```
{
    "requestId": "test-regisiter-service",
    "registerInfo": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "tmpContentId": <tmpContentId>,
        "tmpSecretId": <tmpSecretId>,
        "tmpSecretKey": <tmpSecretKey>,
        "apiGateSecretId": <apiGateSecretId>,
        "apiGateSecretKey": <apiGateSecretKey>,
        "demoCosPath": "UIN_demo/run_musicBeat.py",
        "usageDescription": "Download the python version demo file [UIN_demo/run_mu
        "message": "Registration successful, and thank you for registering.",
        "createdAt": <createdAt>,
        "updatedAt": <updatedAt>
    }
}
```

Run verification.

After the above activation and registration service are completed, a python version executable demo example based on music beat recognition capability will be generated in the `demoCosPath` directory. Execute the command `python run_musicBeat.py` in a networked environment for verification.

Note:

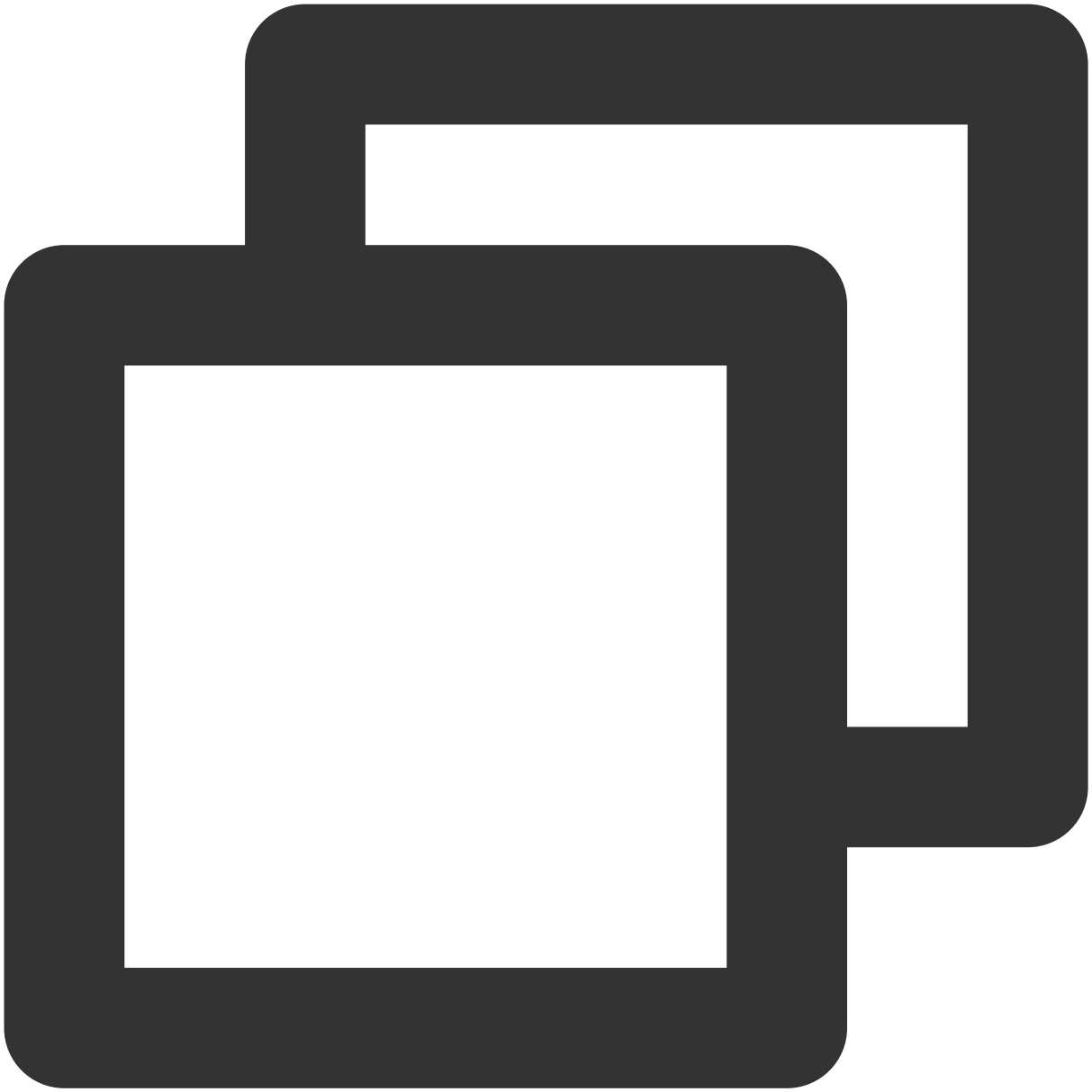
For more detailed intelligent music solution integration instructions, see Integration Guide.

Step 2: Importing SDK.

The TRTC SDK has been released to the **mavenCentral** repository, and you can configure Gradle to download and update automatically.

1. Add the dependency for the appropriate version of the SDK in dependencies.

©2013-2022 Tencent Cloud. All rights reserved.



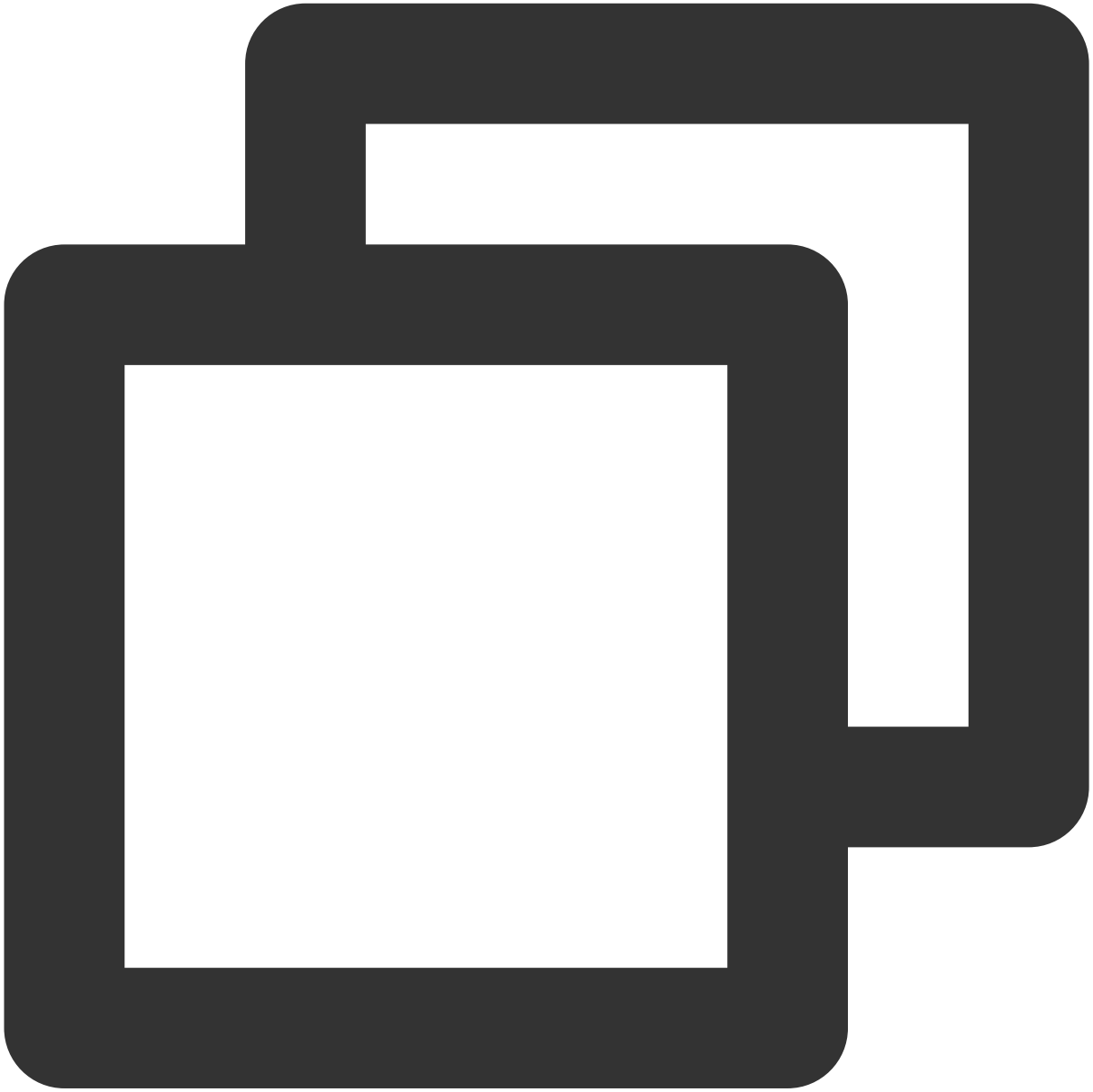
```
dependencies {
    // TRTC Lite SDK. It includes TRTC and live streaming playback features and is
    implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'

    // TRTC Professional SDK. It also includes live streaming, short video, video o
    // implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK.

2. Specify the CPU architecture used by the app in defaultConfig.



```
defaultConfig {
      ndk {
              abiFilters "armeabi-v7a", "arm64-v8a"
      }
}
```

Note:

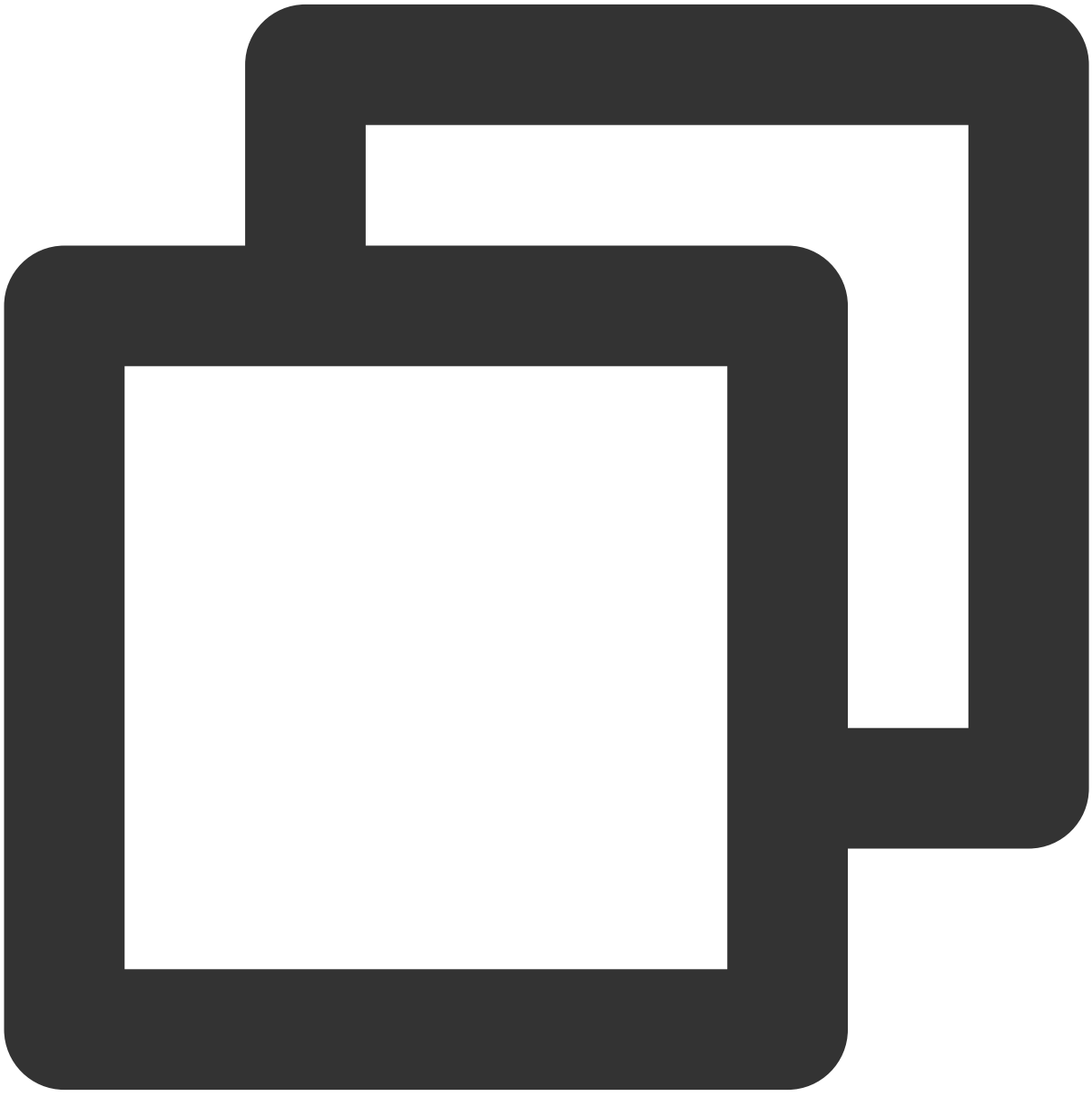
©2013-2022 Tencent Cloud. All rights reserved.

The TRTC SDK supports architectures including armeabi, armeabi-v7a and arm64-v8a. Additionally, it supports architectures for simulators including x86 and x86_64.

Step 3: Project configuration.

1. Configure permissions.

To configure app permissions in AndroidManifest.xml, for karaoke scenarios, the TRTC SDK requires the following permissions:



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

©2013-2022 Tencent Cloud. All rights reserved.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Note:

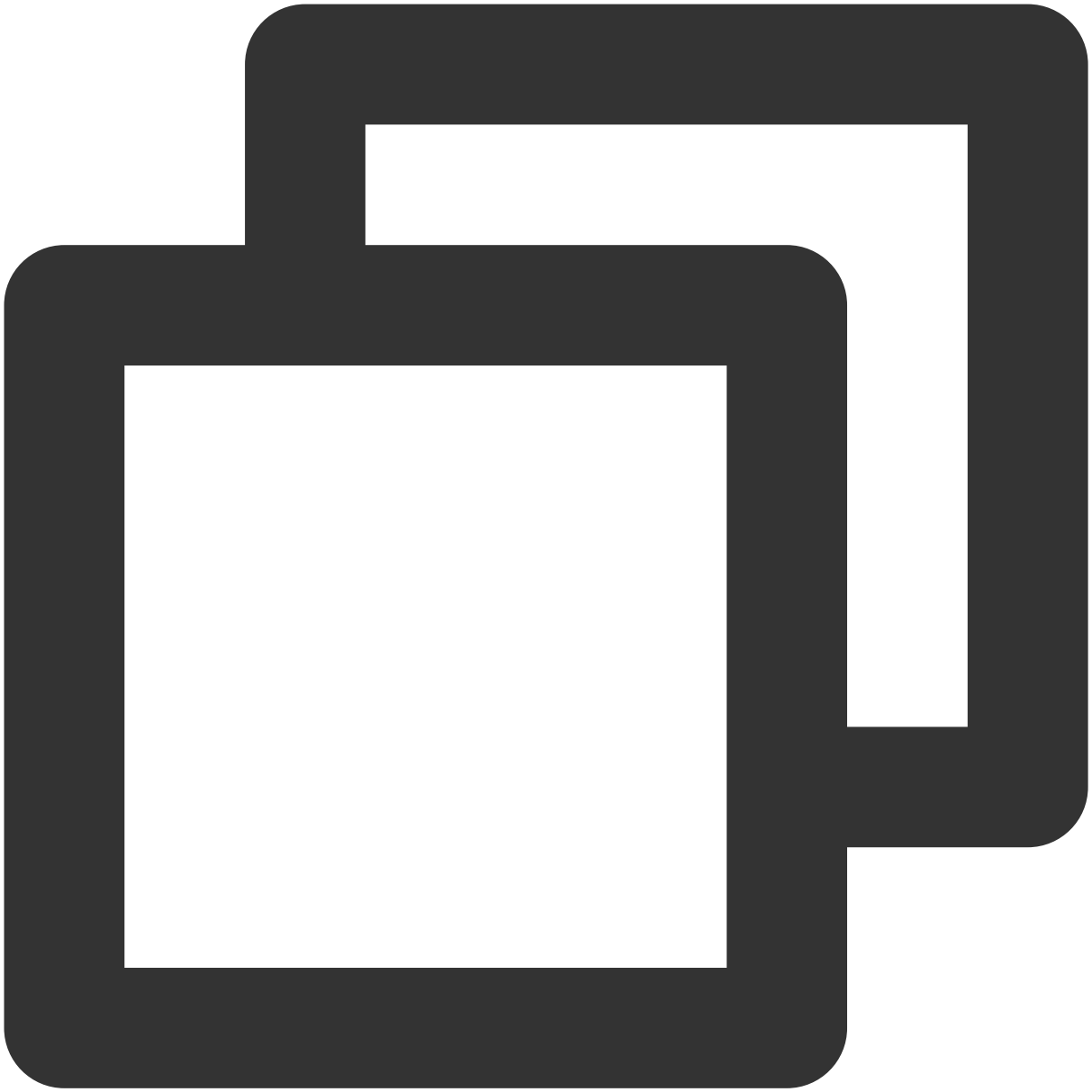
The TRTC SDK does not have built-in permission request logic. You need to declare the corresponding permissions and features yourself. Some permissions (such as storage and recording), also require runtime dynamic requests.

If the Android project's `targetSdkVersion` is 31 or higher, or if the target device runs Android 12 or a newer version, the official requirement is to dynamically request `android.permission.BLUETOOTH_CONNECT` permission in the code to use the Bluetooth feature properly. For more information, see Bluetooth Permissions.

2. Obfuscation configuration.

Since we use Java's reflection features inside the SDK, you need to add relevant SDK classes to the non-obfuscation list in the proguard-rules.pro file:

©2013-2022 Tencent Cloud. All rights reserved.



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

Step 4: Authentication and authorization.

UserSig is a security protection signature designed by Tencent Cloud to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when it enters the room.

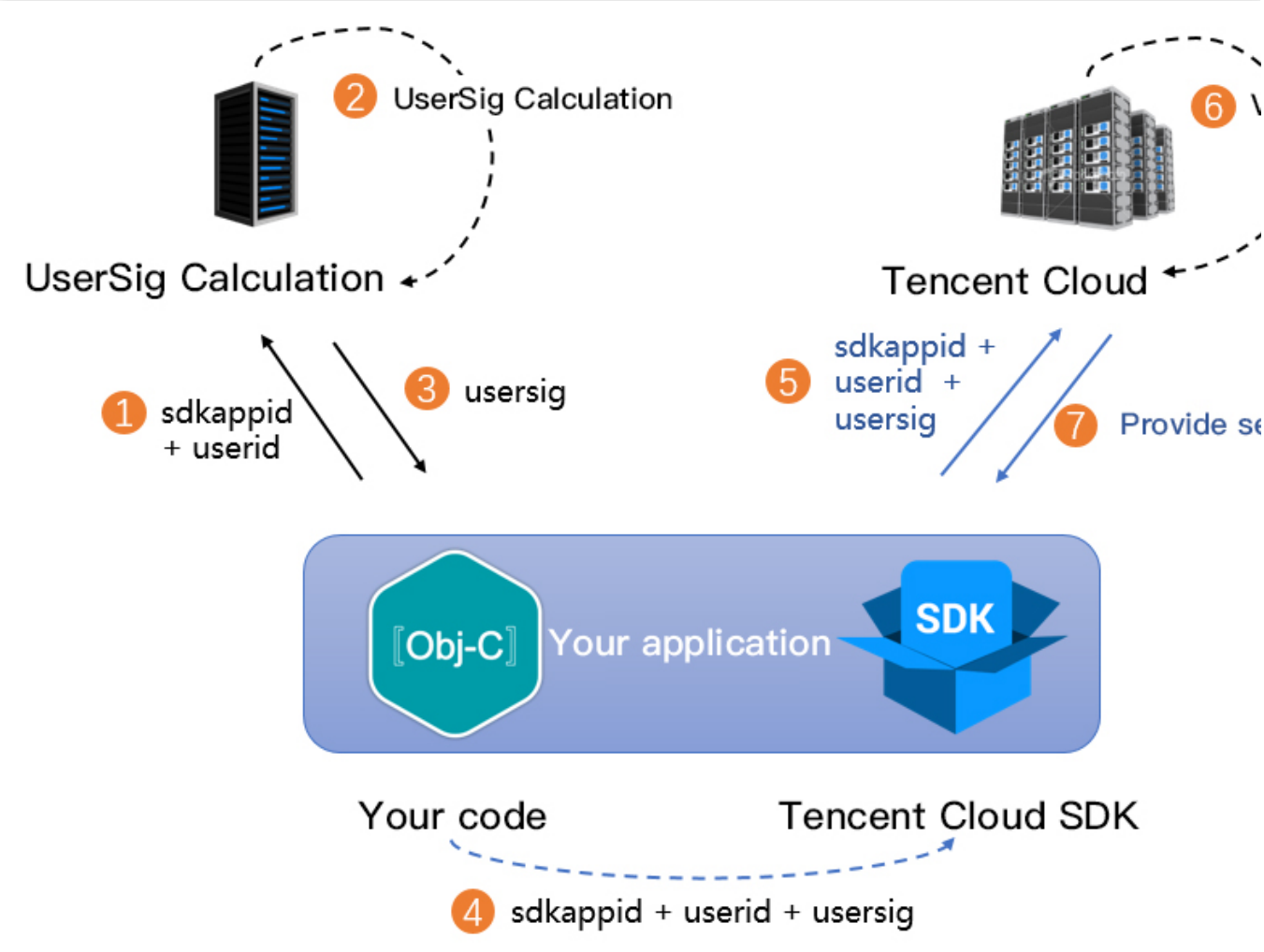
Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.

©2013-2022 Tencent Cloud. All rights reserved.

Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.

The specific implementation process is as follows:

1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.
3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.



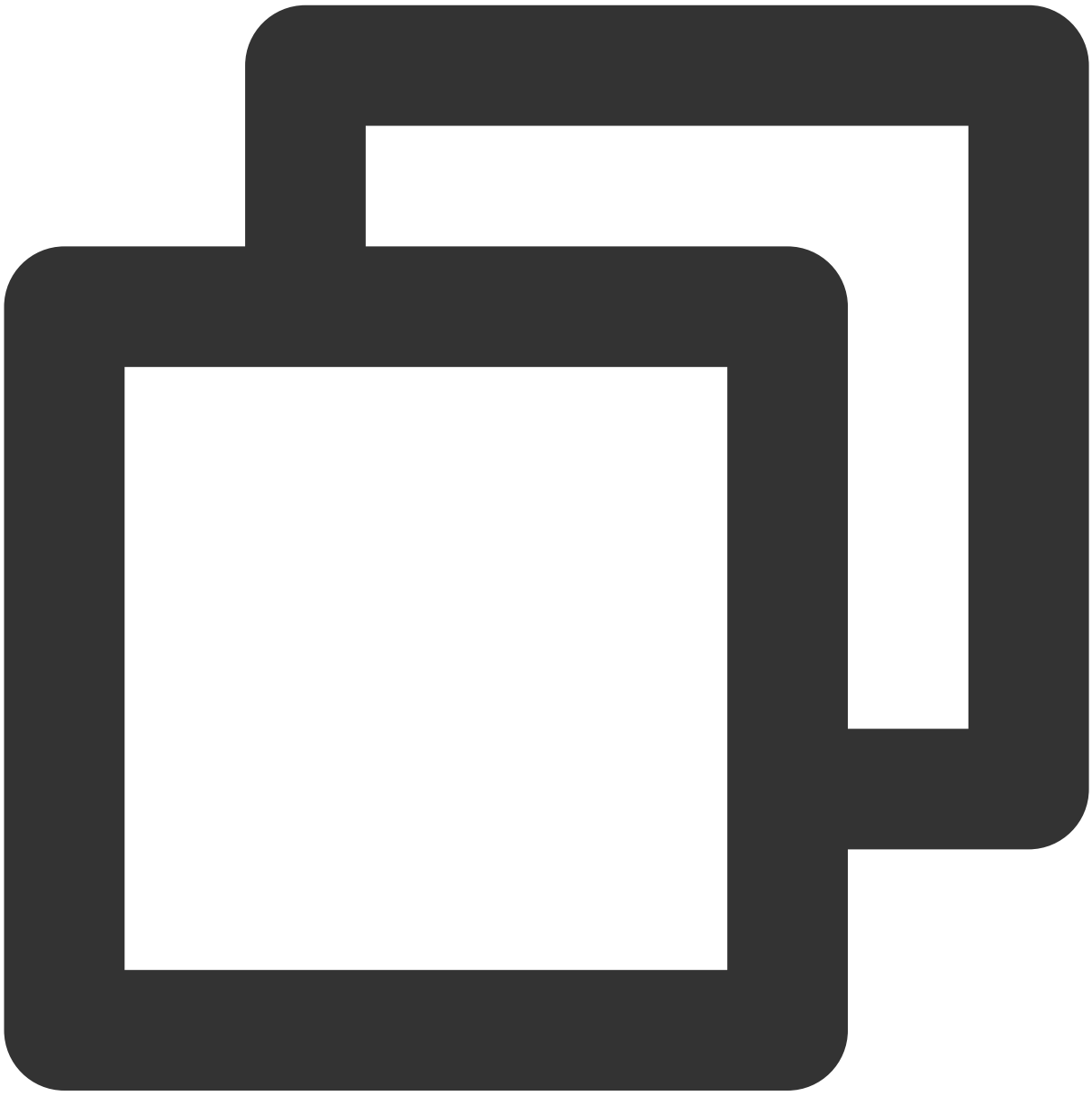
Note:

The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

©2013-2022 Tencent Cloud. All rights reserved.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Step 5: Initializing the SDK.



```
// Create TRTC SDK instance (Single Instance Pattern).
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Set event listeners.
mTRTCCloud.addListener(trtcSdkListener);

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
```

©2013-2022 Tencent Cloud. All rights reserved.

```
private TRTCCloudListener trtcSdkListener = new TRTCCloudListener() {
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        Log.d(TAG, errCode + errMsg);
    }

    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        Log.d(TAG, warningCode + warningMsg);
    }
};

// Remove event listener.
mTRTCCloud.removeListener(trtcSdkListener);
// Terminate TRTC SDK instance (Singleton Pattern).
TRTCCloud.destroySharedInstance();
```

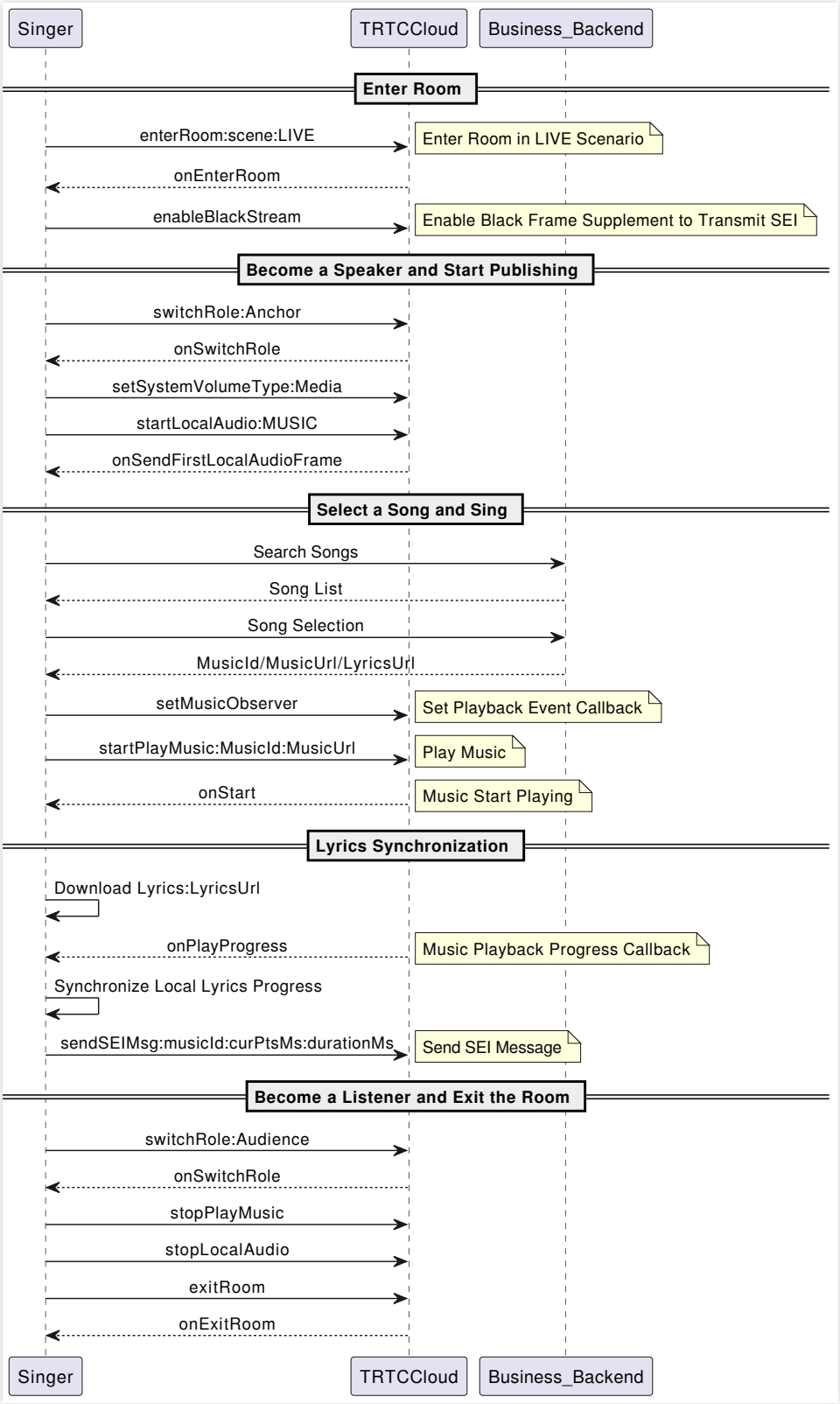
Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

Scenario 1: Solo singing turn-taking

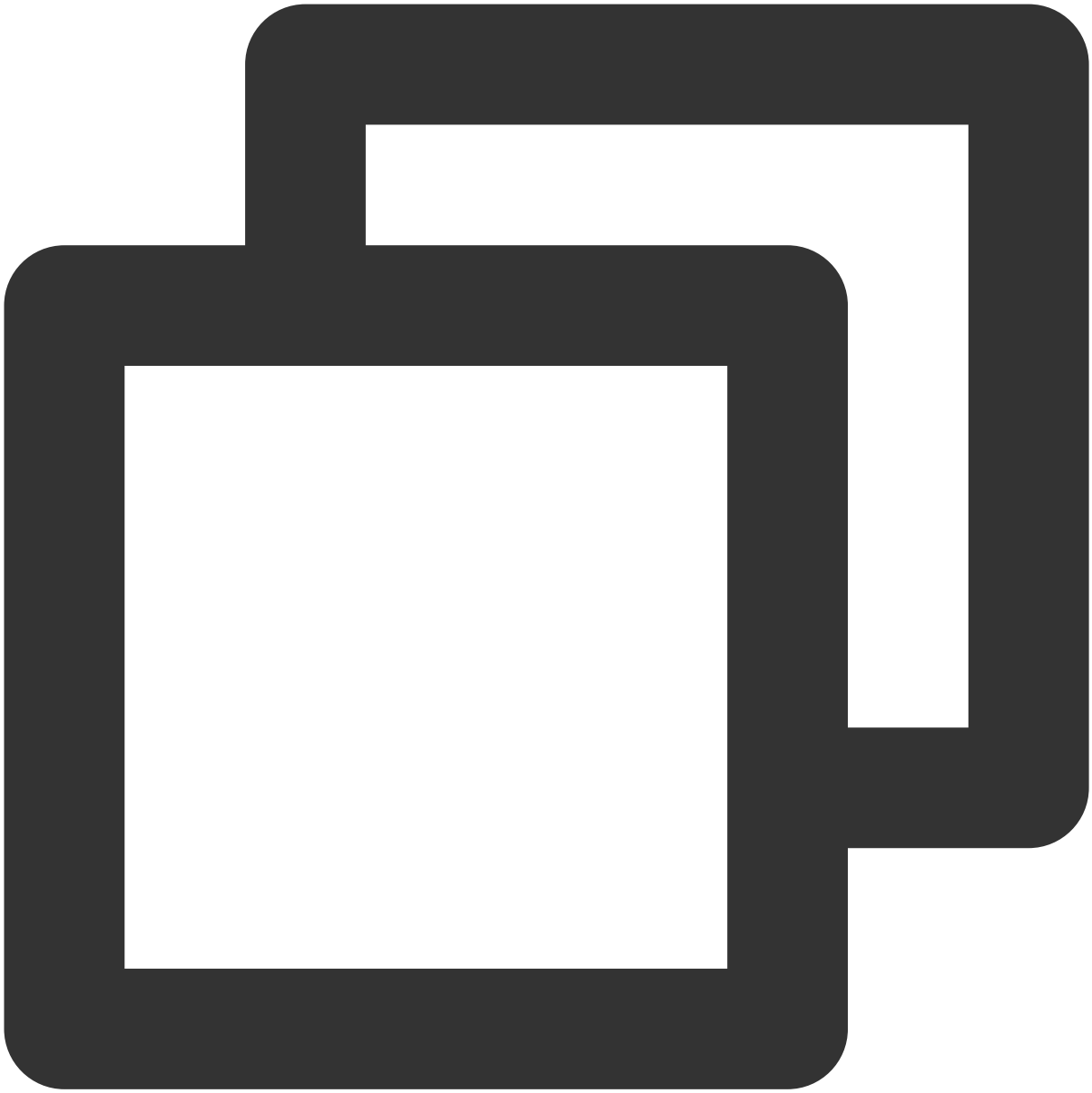
Perspective 1: Performer actions

Sequence diagram



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



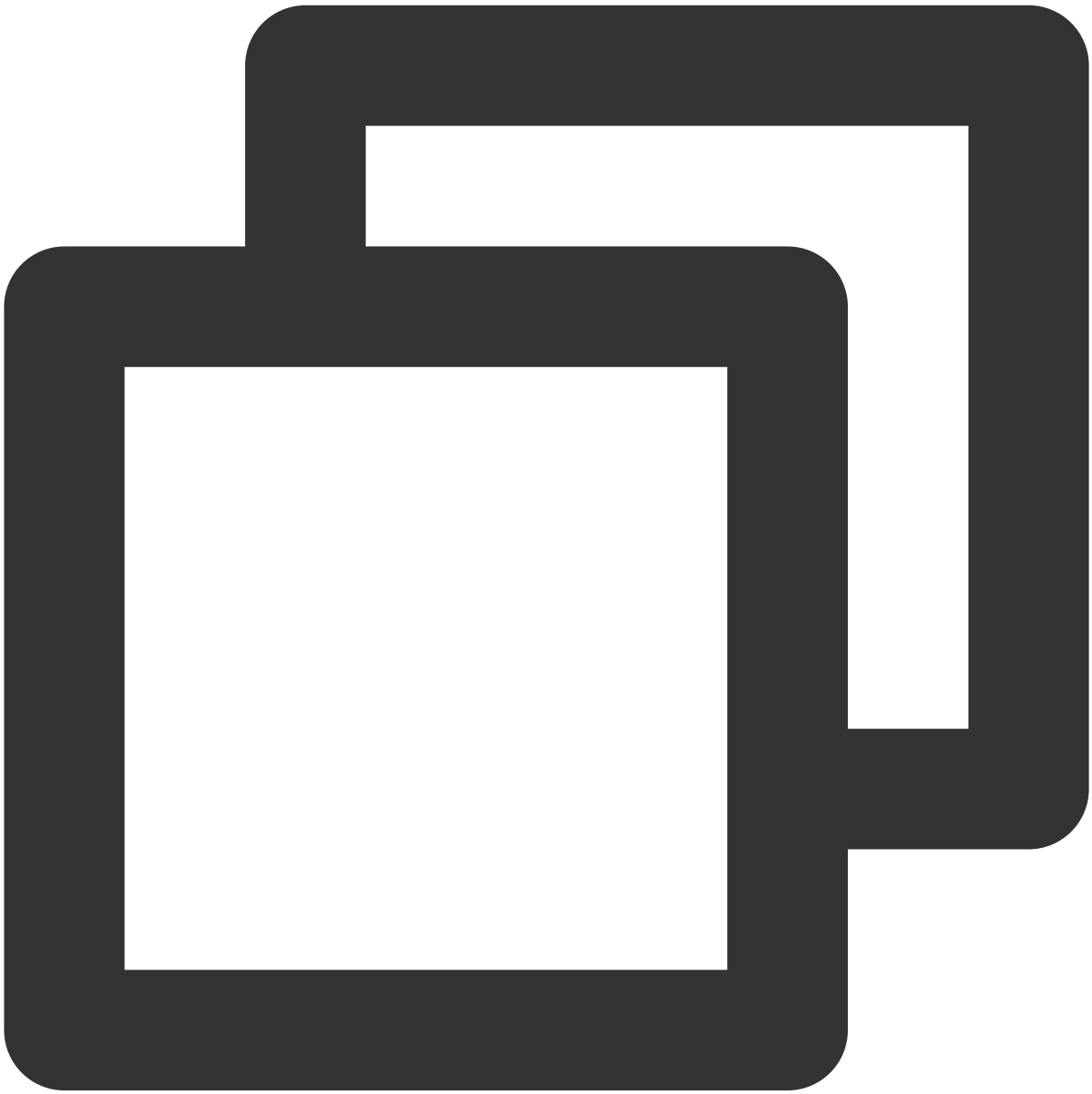
```
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}
```

Note:

To better transmit SEI messages for lyrics synchronization, it is recommended to choose `TRTC_APP_SCENE_LIVE` for room-entry scenarios.



```
// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
```

©2013-2022 Tencent Cloud. All rights reserved.

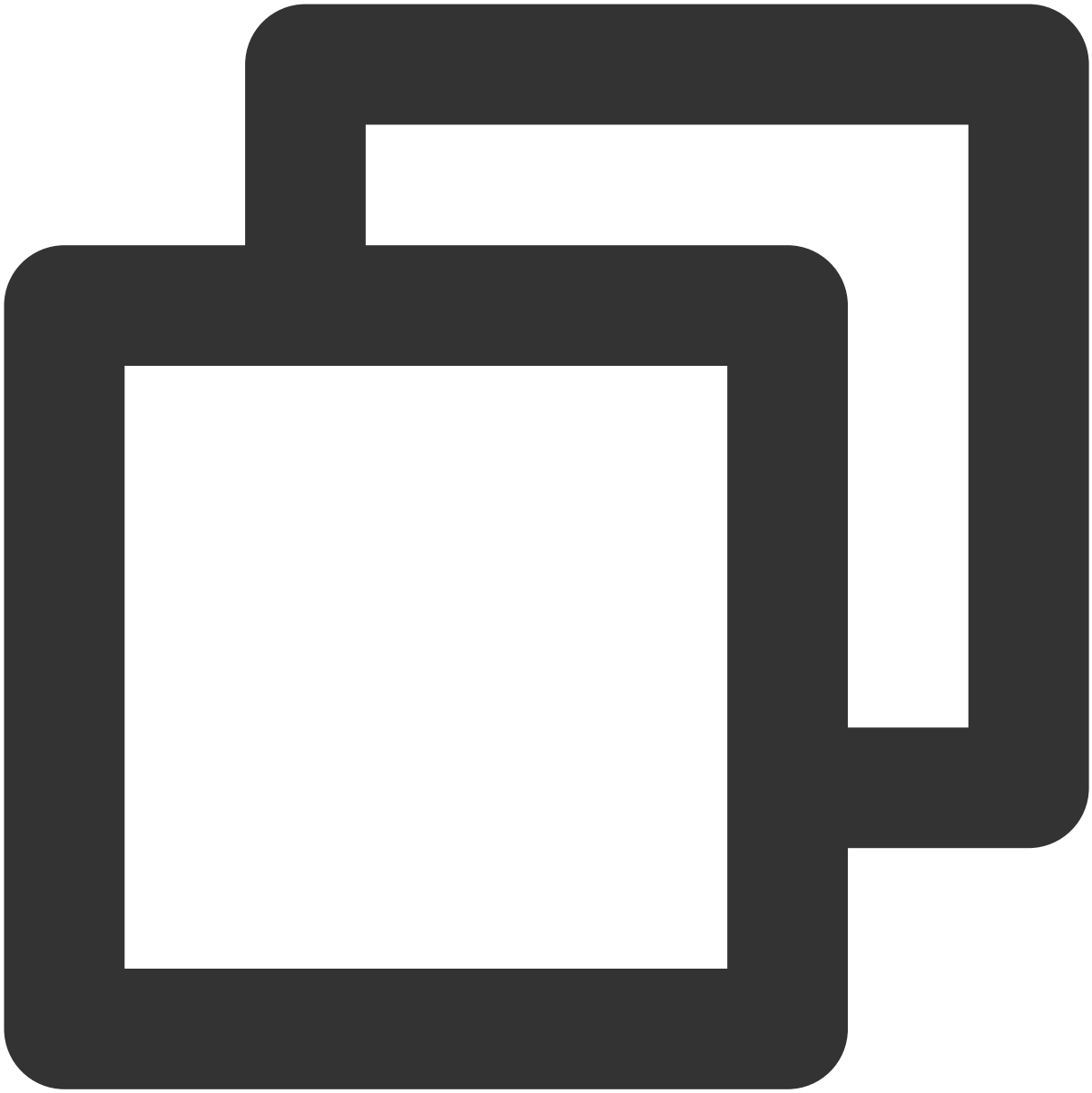
```
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
        // Enable the experimental API for black frame insertion.
        mTRTCCloud.callExperimentalAPI("{\\"api\\":\\"enableBlackStream\\",\\"param
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

Under the pure audio mode, the performer needs to enable the insertion of black frames to carry SEI messages. This API should be called after successfully entering the room.

2. Go live on streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switched to the anchor role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);

// Event callback for switching the role.
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Set media volume type.
        mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);
        // Upstream local audio streams and set audio quality.
        mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }
```

Note:

In karaoke scenarios, it is recommended to set the full-range media volume and music quality to achieve a high-fidelity listening experience.

3. Song selection and performance.

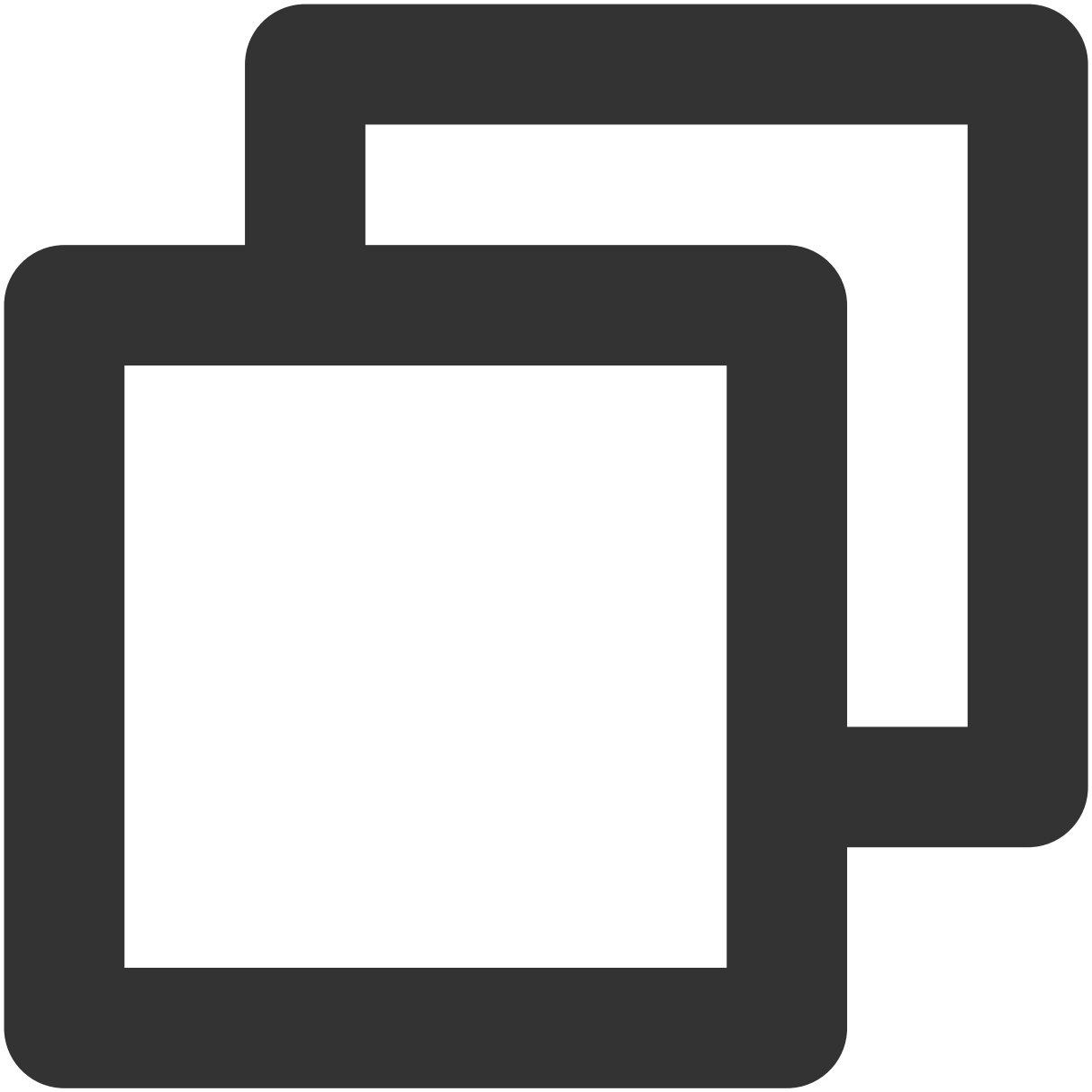
Search for songs, and obtain music resources.

Search for songs and acquire music resources through the business backend. Obtain identifiers such as the MusicId, the song's URL (MusicUrl), and the lyrics URL (LyricsUrl).

It is recommended that the business side select an appropriate music repository production to provide licensed music resources.

Play accompaniment and start singing.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain audio effects management.
TXAudioEffectManager mTXAudioEffectManager = mTRTCCloud.getAudioEffectManager();

// originMusicId: Custom identifier for the original vocal music. originMusicUrl: U
TXAudioEffectManager.AudioMusicParam originMusicParam = new TXAudioEffectManager.Au
// Whether to publish the original vocal music to remote (otherwise play locally on
originMusicParam.publish = true;

// accompMusicId: Custom identifier for the accompaniment music. accompMusicUrl: UR
TXAudioEffectManager.AudioMusicParam accompMusicParam = new TXAudioEffectManager.Au
// Whether to publish the accompaniment to remote (otherwise play locally only).
```

©2013-2022 Tencent Cloud. All rights reserved.

```
accompMusicParam.publish = true;

// Start playing the original vocal music.
mTXAudioEffectManager.startPlayMusic(originMusicParam);
// Start playing the accompaniment music.
mTXAudioEffectManager.startPlayMusic(accompMusicParam);

// Switch to the original vocal music.
mTXAudioEffectManager.setMusicPlayoutVolume(originMusicId, 100);
mTXAudioEffectManager.setMusicPlayoutVolume(accompMusicId, 0);
mTXAudioEffectManager.setMusicPublishVolume(originMusicId, 100);
mTXAudioEffectManager.setMusicPublishVolume(accompMusicId, 0);

// Switch to the accompaniment music.
mTXAudioEffectManager.setMusicPlayoutVolume(originMusicId, 0);
mTXAudioEffectManager.setMusicPlayoutVolume(accompMusicId, 100);
mTXAudioEffectManager.setMusicPublishVolume(originMusicId, 0);
mTXAudioEffectManager.setMusicPublishVolume(accompMusicId, 100);
```

Note:

In karaoke scenarios, both the original vocal and accompaniment need to be played simultaneously (distinguished by MusicID). The switch between the original vocal and accompaniment is achieved by adjusting the local and remote playback volumes.

If the music being played has dual audio tracks (including both the original vocal and accompaniment), switching between them can be achieved by specifying the music's playback track using setMusicTrack.

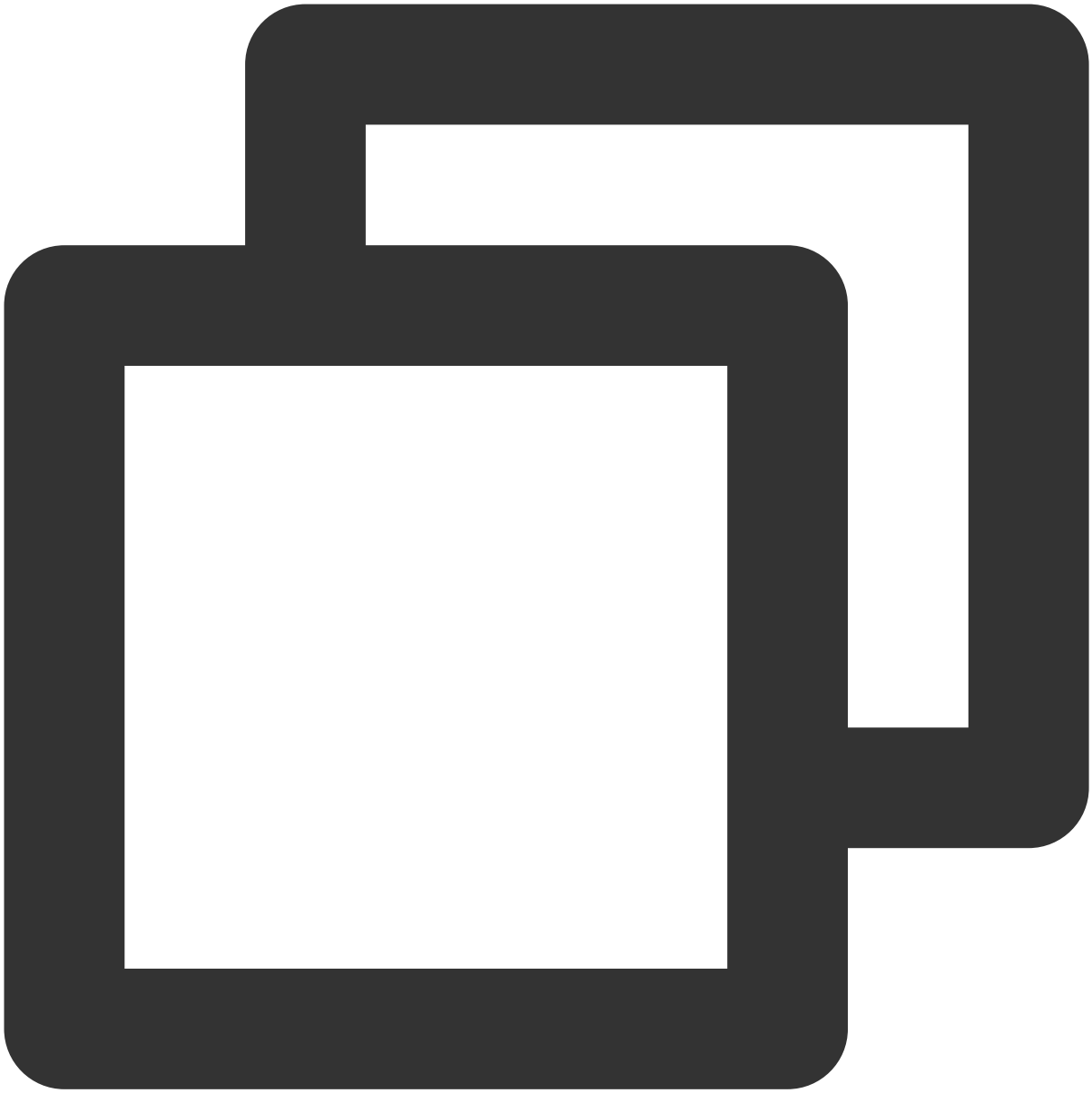
4. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Synchronize local lyrics, and transmit song progress via SEI.

©2013-2022 Tencent Cloud. All rights reserved.



```
mTXAudioEffectManager.setMusicObserver(musicId, new TXAudioEffectManager.TXMusicPla
    @Override
    public void onStart(int id, int errCode) {
        // Start playing music.
    }

    @Override
    public void onPlayProgress(int id, long curPtsMs, long durationMs) {
        // Determine whether seek is needed based on the latest progress and the lo
        // Song progress is transmitted by sending an SEI message.
        try {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("musicId", id);
            jsonObject.put("progress", curPtsMs);
            jsonObject.put("duration", durationMs);
            mTRTCCloud.sendSEIMsg(jsonObject.toString().getBytes(), 1);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onComplete(int id, int errCode) {
        // Music playback completed.
    }
});
```

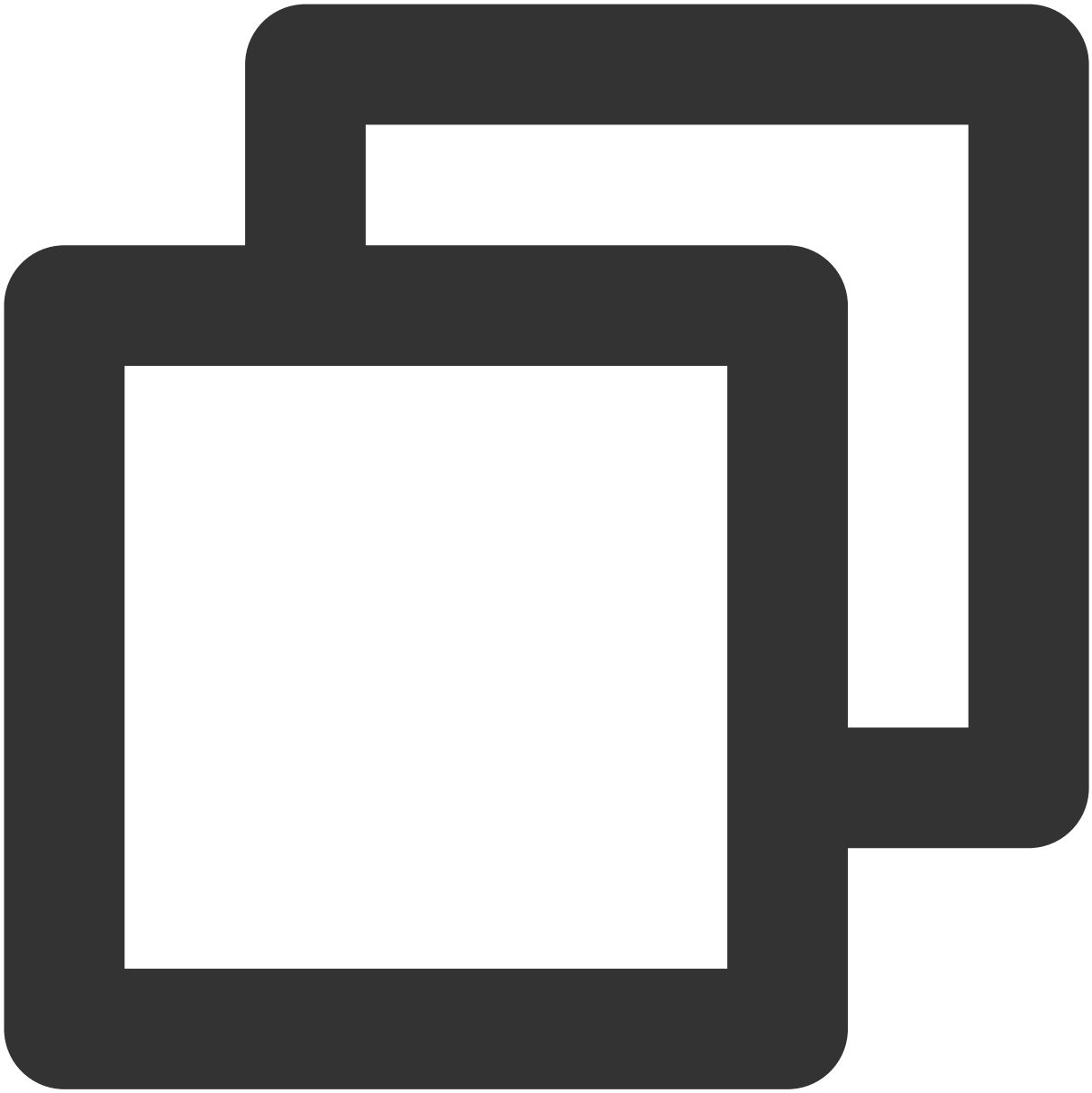
Note:

Ensure to set the playback event callback using this API before playing the background music. This allows to be aware of the background music's playback progress.

The frequency of the SEI messages sent by the performer is determined by the event callback frequency. Also, the playback progress can be actively synchronized on a schedule through getMusicCurrentPosInMS.

5. Become a listener and exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switched to the audience role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);

// Event callback for switching the role.
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Stop playing accompaniment music.
        mTRTCCloud.getAudioEffectManager().stopPlayMusic(musicId);
        // Stop local audio capture and publishing.
        mTRTCCloud.stopLocalAudio();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
}

// Exit the room.
mTRTCCloud.exitRoom();

// Exit room event callback.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server.");
    } else if (reason == 2) {
        Log.d(TAG, "The current room has been dissolved.");
    }
}
```

Note:

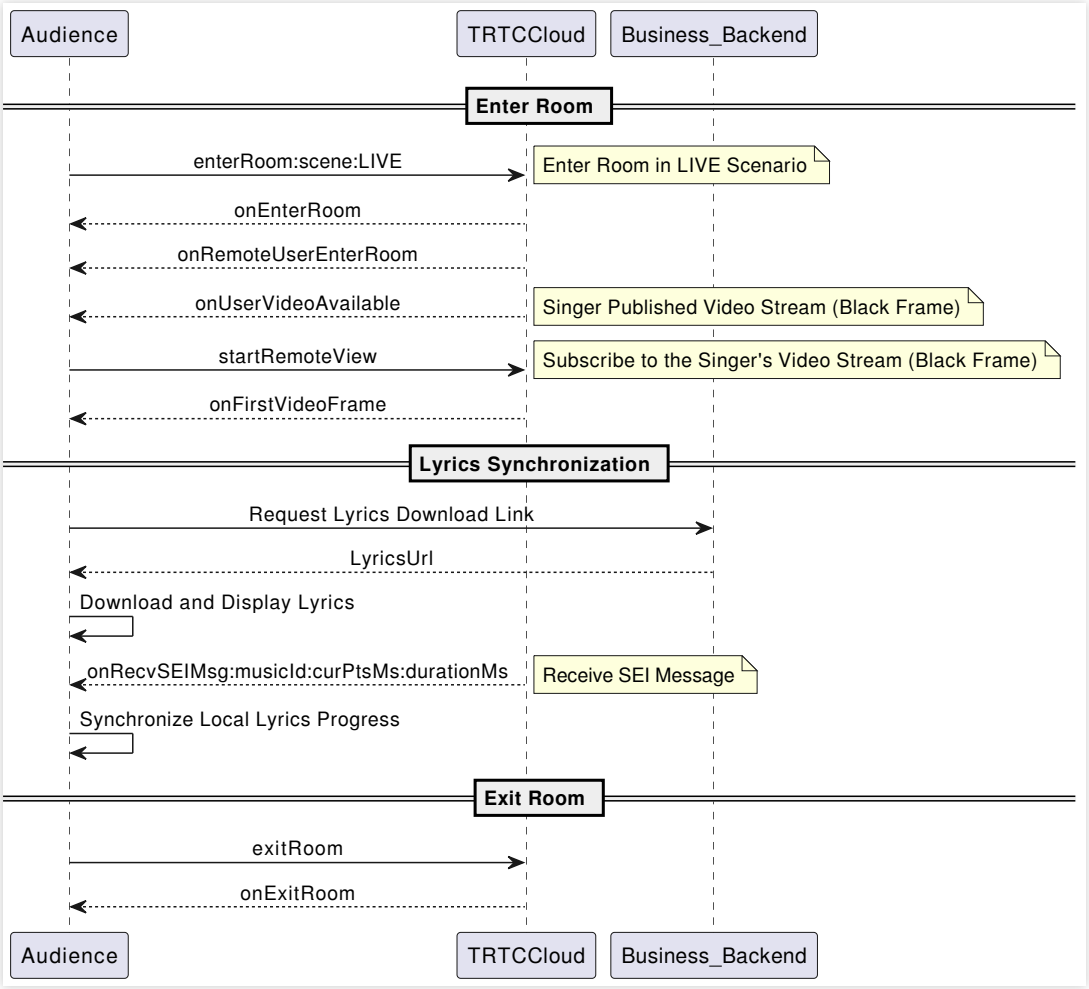
After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you want to call `enterRoom` again or switch to another audio and video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter various exceptional issues such as the camera, microphone device being forcibly occupied.

Perspective 2: Listener actions

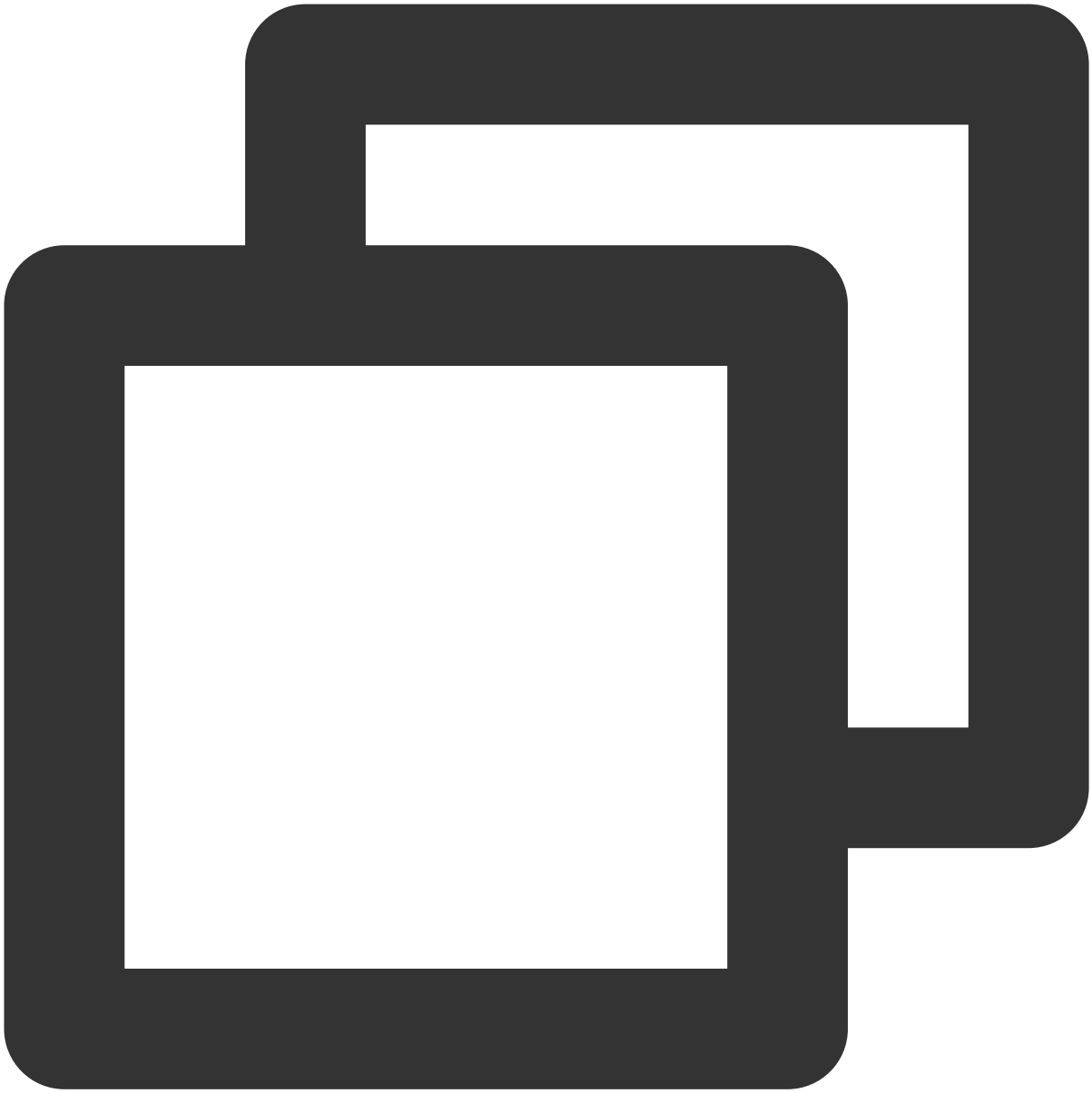
Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

To better transmit SEI messages for lyrics synchronization, it is recommended to choose `TRTC_APP_SCENE_LIVE` for room-entry scenarios.

Under the automatic subscription mode (default), audiences automatically subscribe and play the on-mic anchor's audio and video streams upon entering the room.

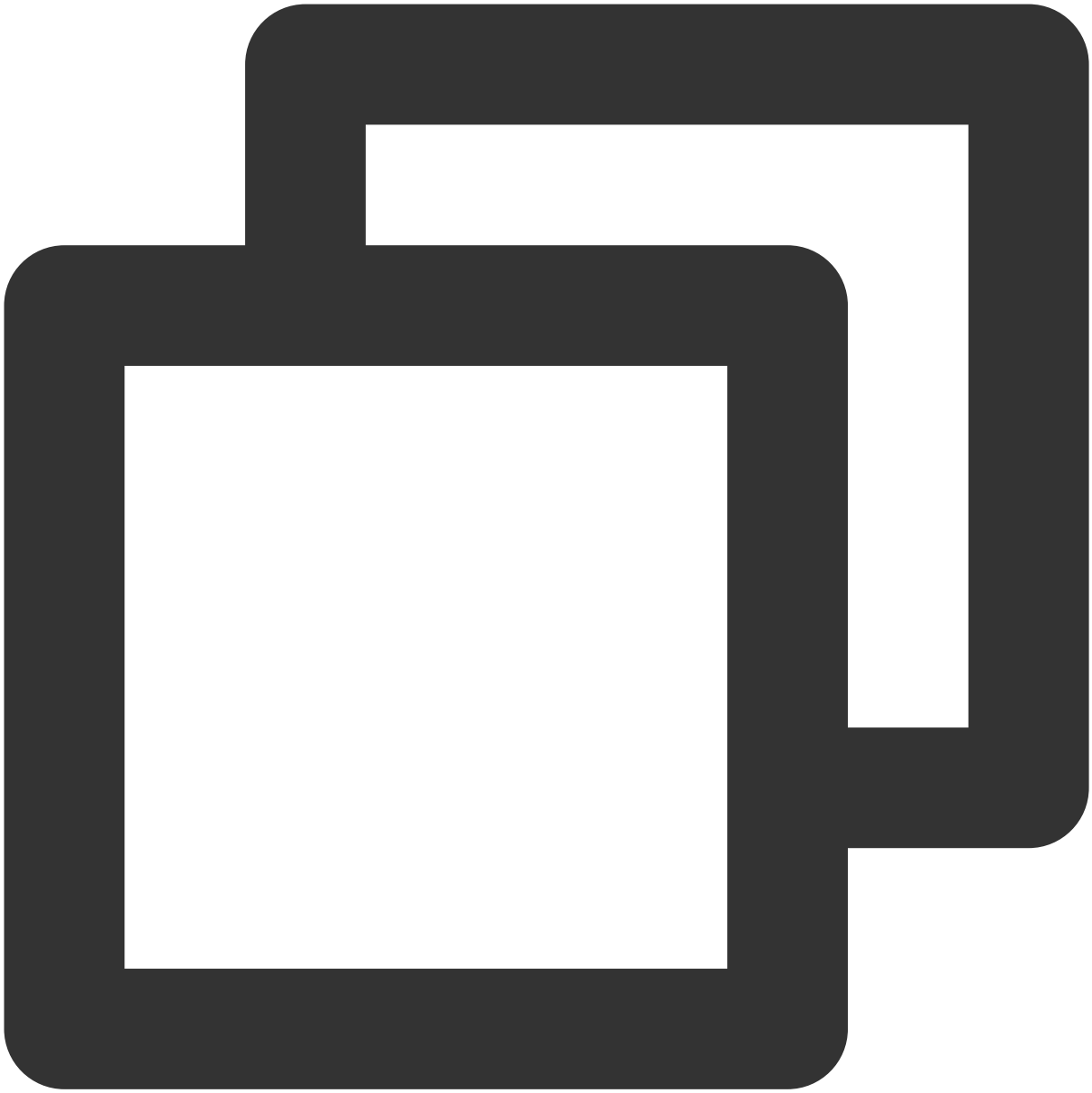
2. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Listener end lyric synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onUserVideoAvailable(String userId, boolean available) {
    if (available) {
        mTRTCCloud.startRemoteView(userId, null);
    } else {
        mTRTCCloud.stopRemoteView(userId);
    }
}

@Override
public void onRecvSEIMsg(String userId, byte[] data) {
```

©2013-2022 Tencent Cloud. All rights reserved.

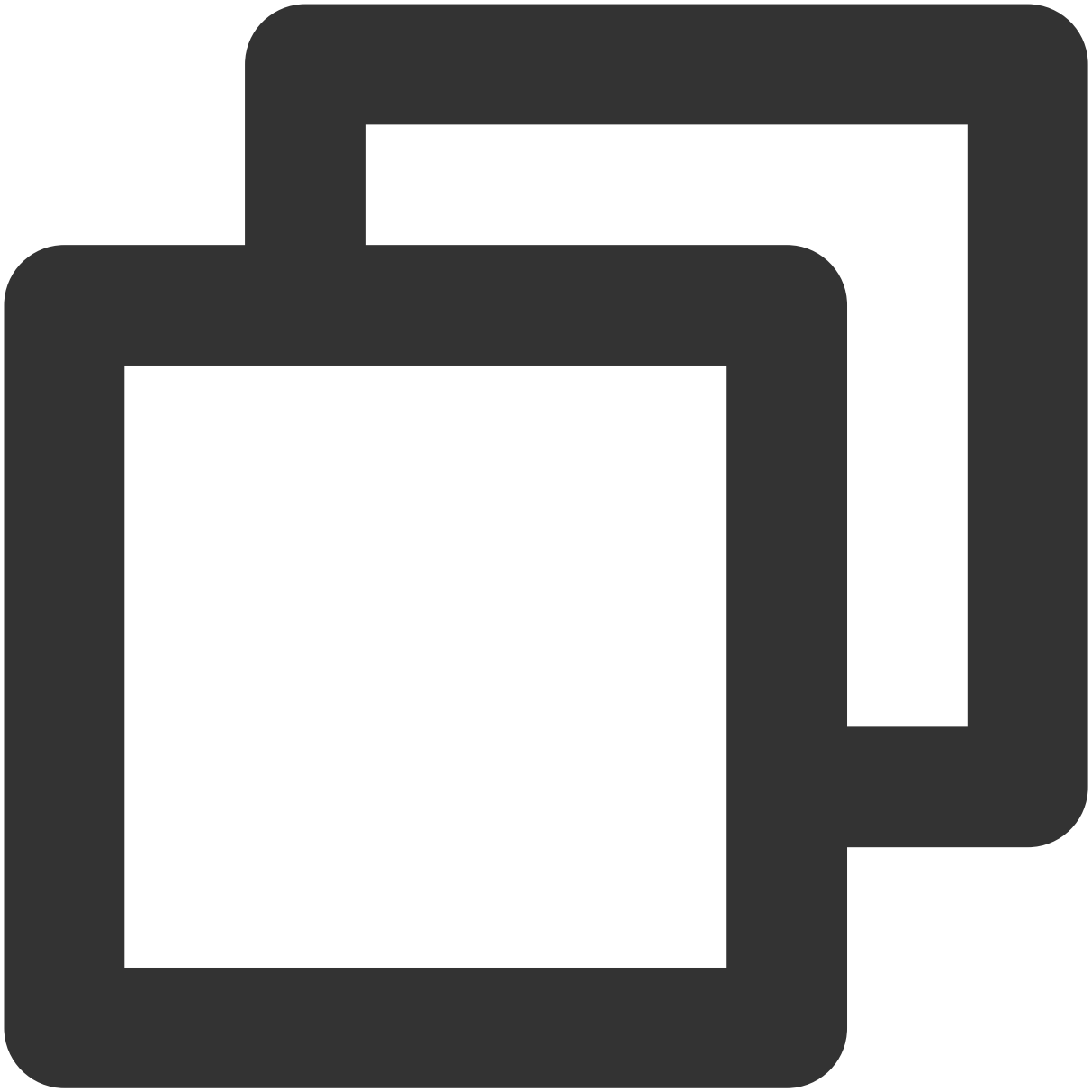
```
    String result = new String(data);
    try {
        JSONObject jsonObject = new JSONObject(result);
        int musicId = jsonObject.getInt("musicId");
        long progress = jsonObject.getLong("progress");
        long duration = jsonObject.getLong("duration");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    ...
    // TODO: The logic of updating the lyric control.
    // Based on the received latest progress and the local lyrics progress deviatio
    ...
}
```

Note:

Listeners need to actively subscribe to the performer's video streams in order to receive the SEI messages carried by black frames.

3. Exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Exit the room.
mTRTCCloud.exitRoom();

// Exit room event callback.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server.");
    } else if (reason == 2) {
```

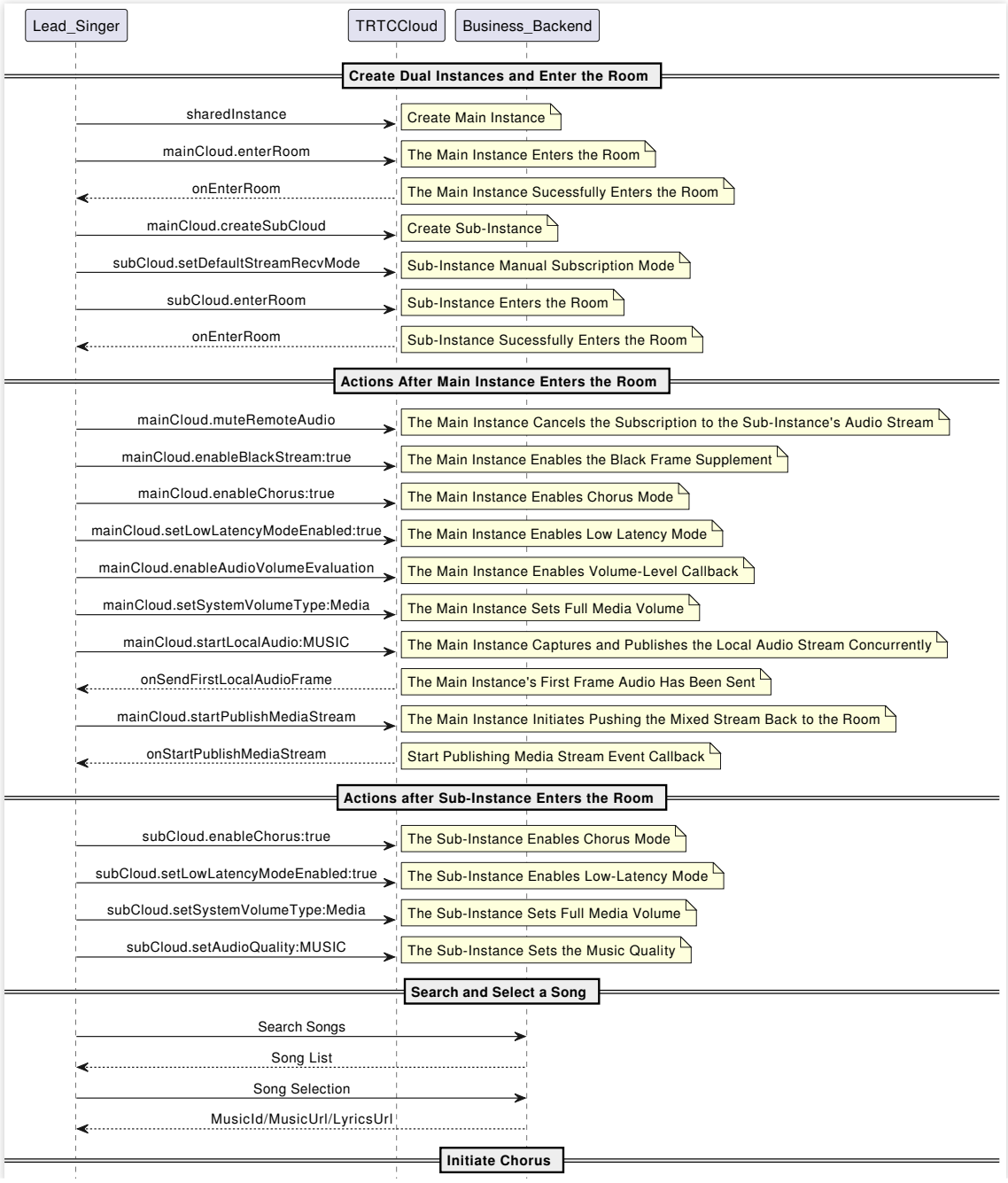
©2013-2022 Tencent Cloud. All rights reserved.

```
            Log.d(TAG, "The current room has been dissolved.");
        }
    }
```

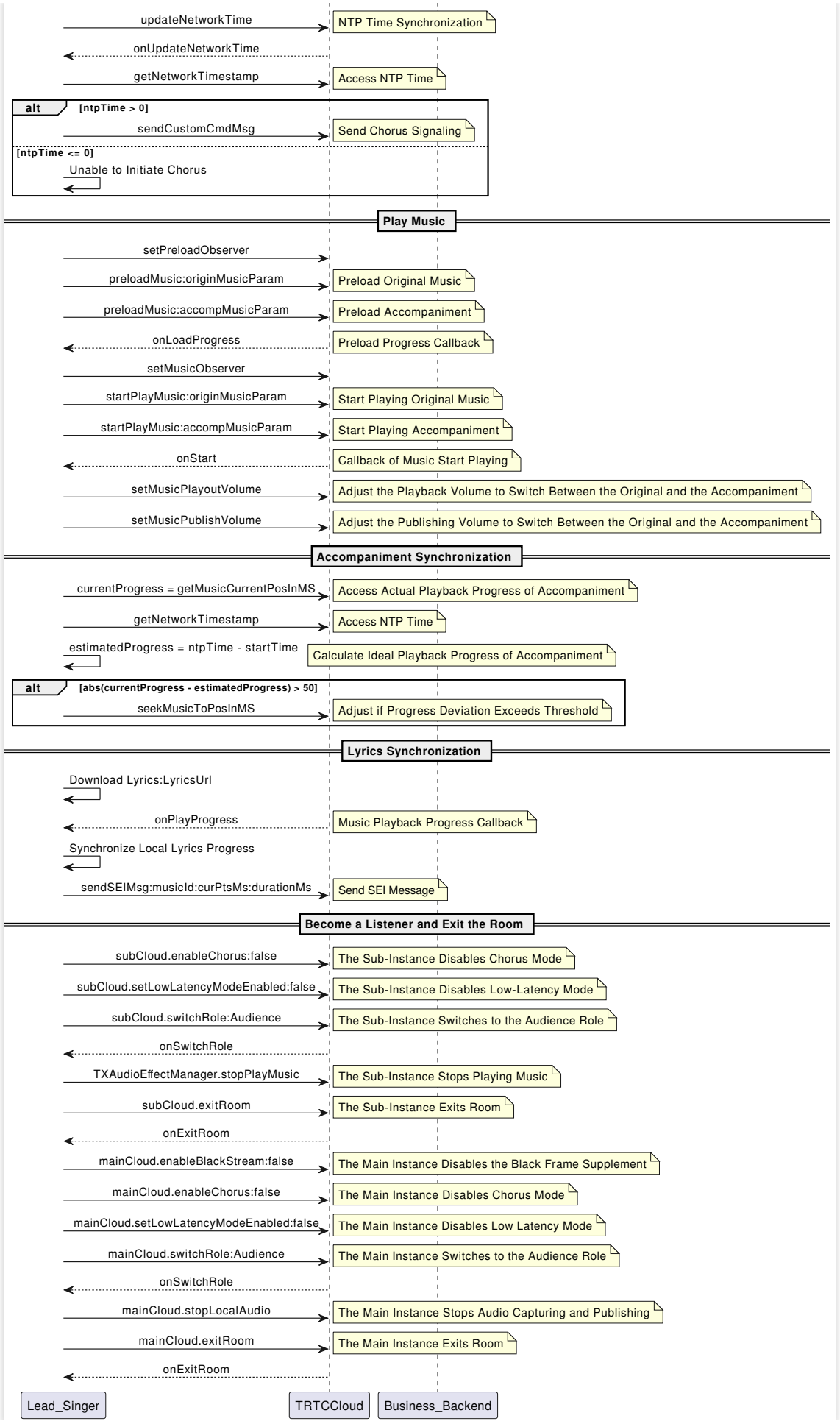
Scenario 2: Real-time chorus

Perspective 1: Lead singer actions

Sequence diagram

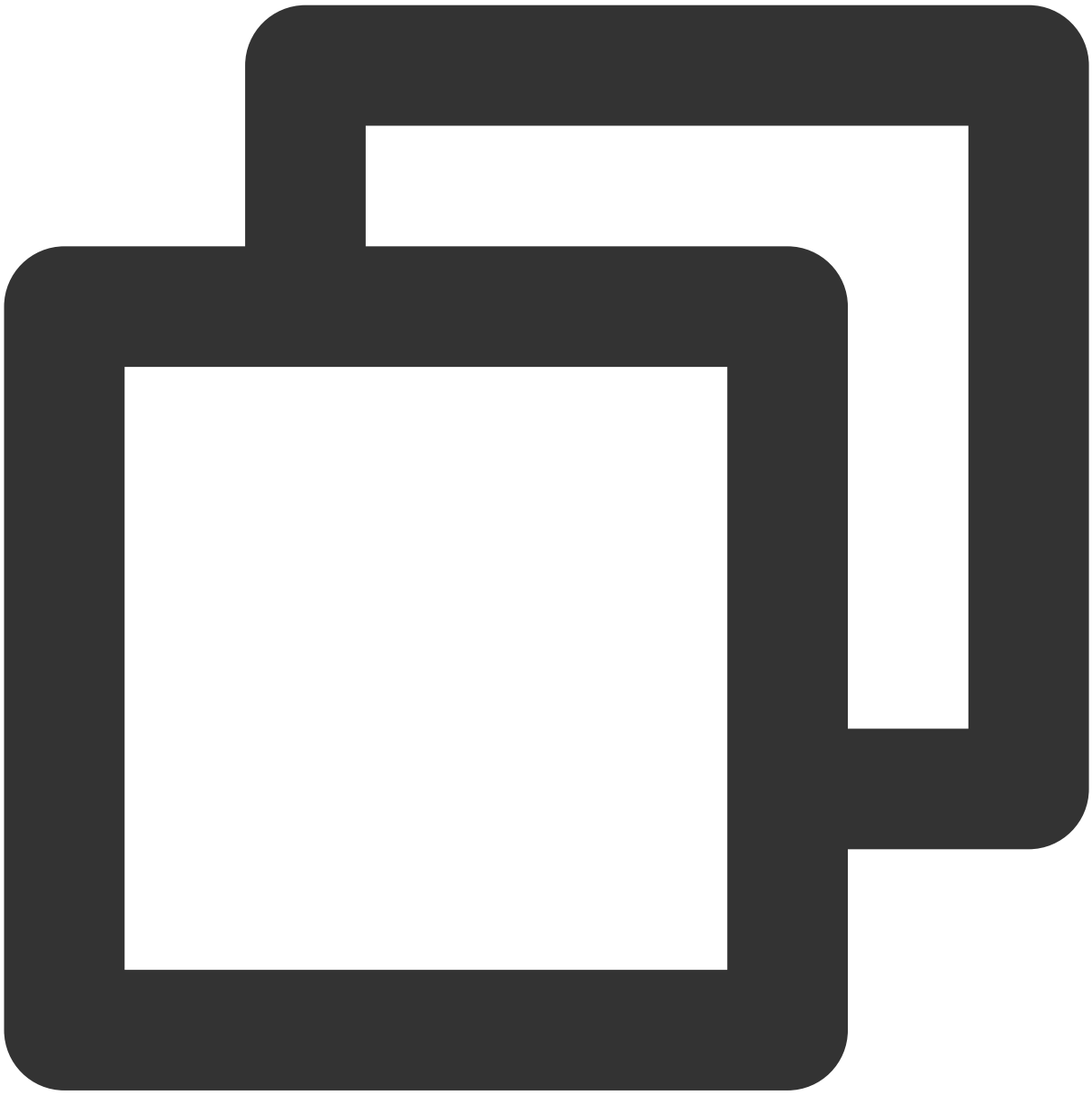


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

1. Dual instances enter the room.



```
// Create a TRTCCloud primary instance (vocal instance).
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Create a TRTCCloud sub-instance (music instance).
TRTCCloud subCloud = mTRTCCloud.createSubCloud();

// The primary instance (vocal instance) enters the room.
TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
params.sdkAppId = SDKAppId;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
params.userId = UserId;
params.userSig = UserSig;
params.role = TRTCCloudDef.TRTCRoleAnchor;
params.strRoomId = RoomId;
mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);

// The sub-instance enables manual subscription mode. By default it does not subscr
subCloud.setDefaultStreamRecvMode(false, false);

// The sub-instance (music instance) enters the room.
TRTCCloudDef.TRTCParams bgmParams = new TRTCCloudDef.TRTCParams();
bgmParams.sdkAppId = SDKAppId;
// The sub-instance username must not duplicate with other users in the room.
bgmParams.userId = UserId + "_bgm";
bgmParams.userSig = UserSig;
bgmParams.role = TRTCCloudDef.TRTCRoleAnchor;
bgmParams.strRoomId = RoomId;
subCloud.enterRoom(bgmParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

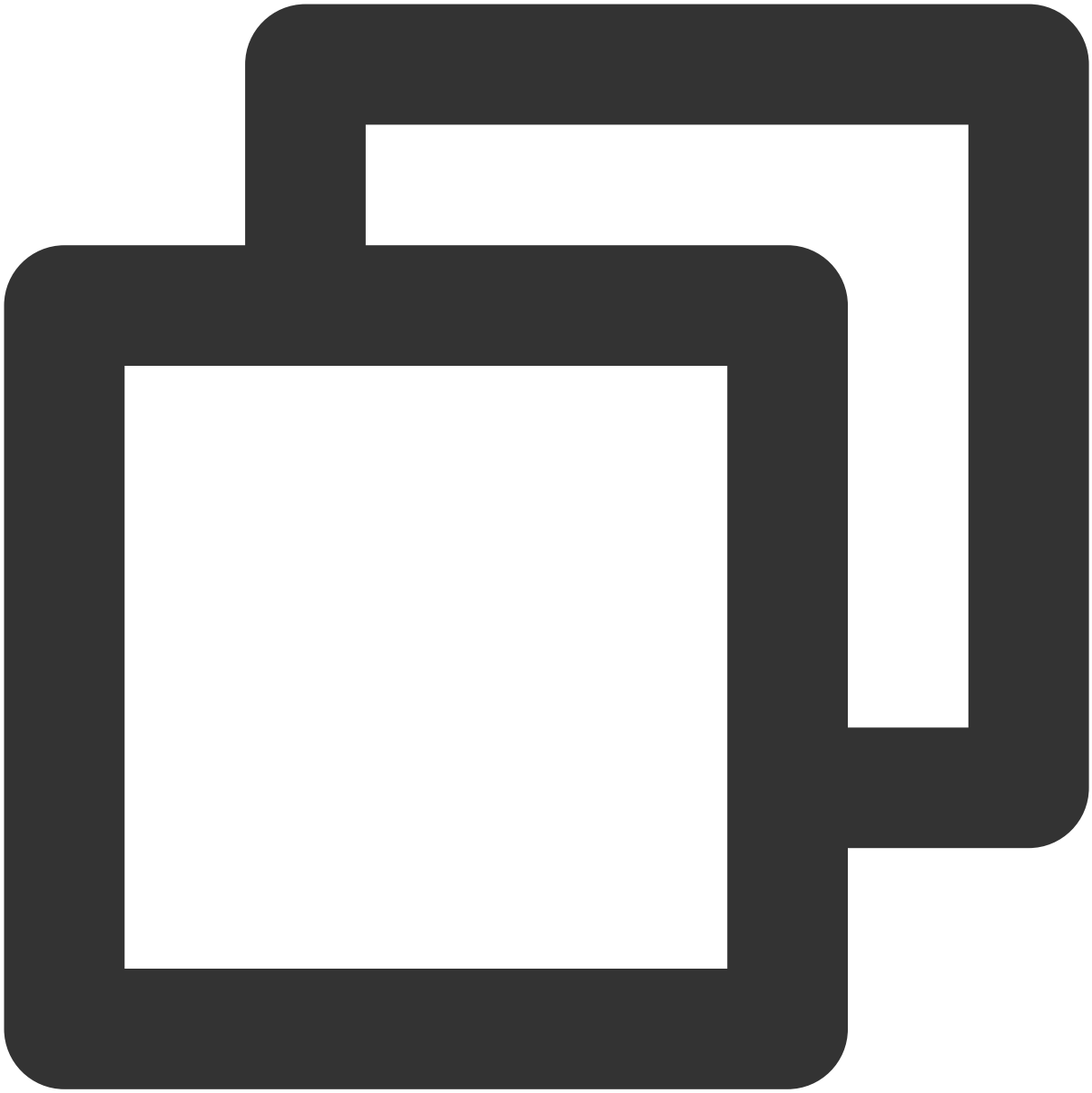
Note:

In a real-time chorus solution, the lead singer end must create primary instance and sub-instance for upstream voice and accompaniment music, respectively.

Sub-instances do not need to subscribe to other users' audio streams in the room. Therefore, it is recommended to enable manual subscription mode, and it must be activated before entering the room.

2. Set the settings after entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the result of primary instance entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The primary instance unsubscribe from music streams published by sub-ins
        mTRTCCloud.muteRemoteAudio(UserId + "_bgm", true);
        // The primary instance uses the experimental API to enable black frame ins
        mTRTCCloud.callExperimentalAPI(
        "{\\"api\\":\\"enableBlackStream\\",\\"params\\": {\\"enable\\":true}}");
        // The primary instance uses the experimental API to enable chorus mode.
        mTRTCCloud.callExperimentalAPI(
```

©2013-2022 Tencent Cloud. All rights reserved.

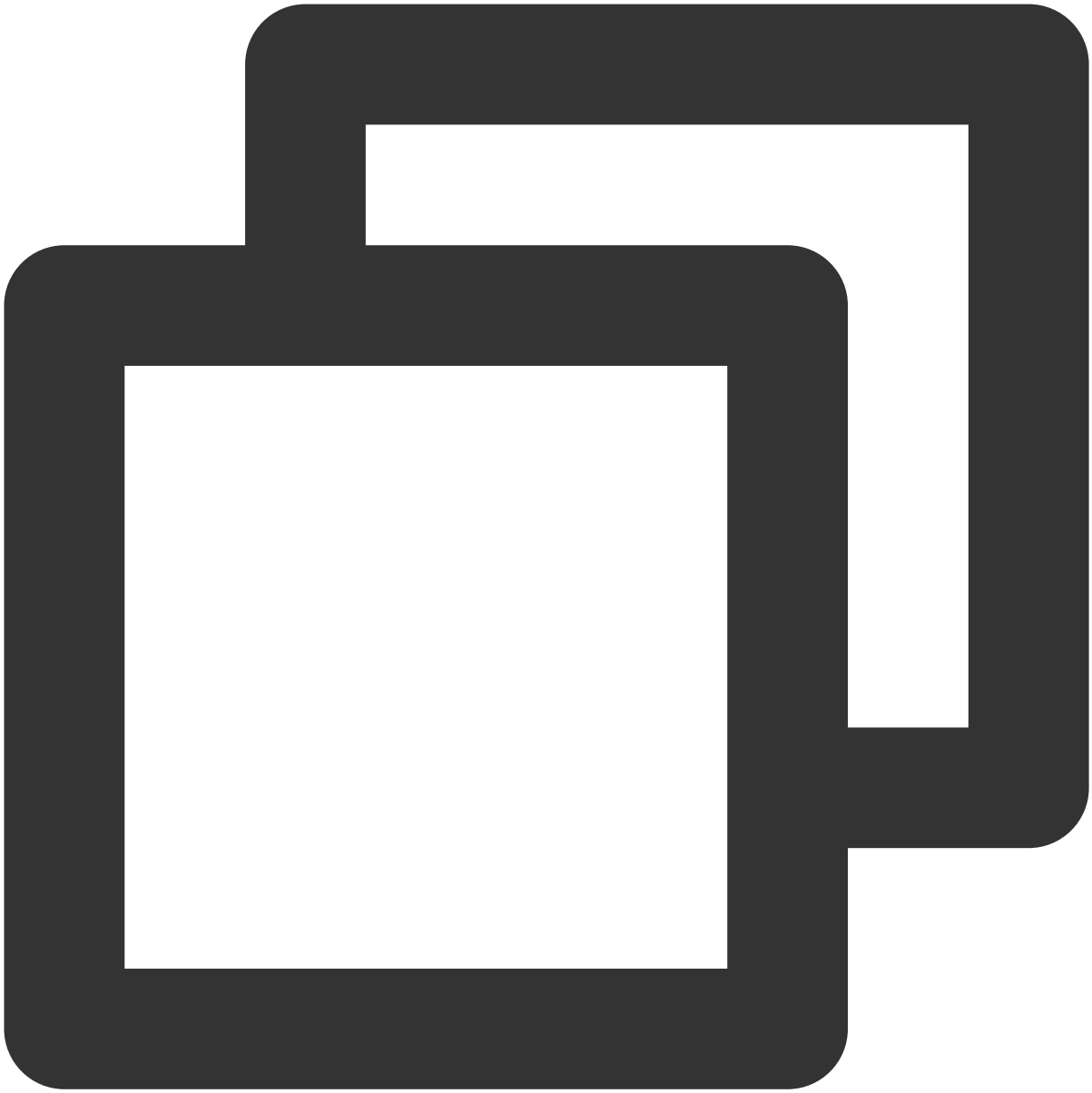
```
        "{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":true,\\"audioSour
        // The primary instance uses the experimental API to enable low-latency mod
        mTRTCCloud.callExperimentalAPI(
        "{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":true}
        // The primary instance enables volume level callback.
        mTRTCCloud.enableAudioVolumeEvaluation(300, false);
        // The primary instance sets the global media volume type.
        mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);
        // The primary instance captures and publishes local audio, and sets audio
        mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}

// Event callback for the result of sub-instance entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The sub-instance uses the experimental API to enable chorus mode.
        subCloud.callExperimentalAPI(
        "{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":true,\\"audioSour
        // The sub-instance uses the experimental API to enable low-latency mode.
        subCloud.callExperimentalAPI(
        "{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":true}
        // The sub-instance sets global media volume type.
        subCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);
        // The sub-instance sets audio quality.
        subCloud.setAudioQuality(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

Both the primary instance and sub-instance must use the experimental APIs to enable chorus mode and low-latency mode to optimize the chorus experience. Note the difference in the `audioSource` parameter.

3. Push the mixed stream back to the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
private void startPublishMediaToRoom(String roomId, String userId) {
    // Create TRTCPublishTarget object.
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // After mixing, the stream is relayed back to the room.
    target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToRoom;
    target.mixStreamIdentity.strRoomId = roomId;
    // The mixing stream robot's username must not duplicate with other users in th
    target.mixStreamIdentity.userId = userId + "_robot";

    // Set the encoding parameters of the transcoded audio stream (can be customize
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    trtcStreamEncoderParam.audioEncodedChannelNum = 2;
    trtcStreamEncoderParam.audioEncodedKbps = 64;
    trtcStreamEncoderParam.audioEncodedCodecType = 2;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;

    // Set the encoding parameters of the transcoded video stream (black frame mixi
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 3;
    trtcStreamEncoderParam.videoEncodedKbps = 30;
    trtcStreamEncoderParam.videoEncodedWidth = 64;
    trtcStreamEncoderParam.videoEncodedHeight = 64;

    // Set audio mixing parameters.
    TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new TRTCCloudDef.T
    // By default, leave this field empty. It indicates that all audio in the room
    trtcStreamMixingConfig.audioMixUserList = null;

    // Configure video mixed-stream template (black frame mixing required).
    TRTCCloudDef.TRTCVideoLayout videoLayout = new TRTCCloudDef.TRTCVideoLayout();
    trtcStreamMixingConfig.videoLayoutList.add(videoLayout);

    // Start mixing and pushing back.
    mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam, trtcStreamMi
}
```

Note:

To maintain alignment between chorus vocals and accompaniment music, it is recommended to enable pushing the mixed stream back to the room. The on-mic chorus members mutually subscribe to single streams, and off-mic audiences by default only subscribe to mixed streams.

The mixing stream robot, acting as an independent user, enters the room to pull, mix, and push streams. Its username must not duplicate with other usernames in the room. Otherwise, it may lead to mutual deletion from the room.

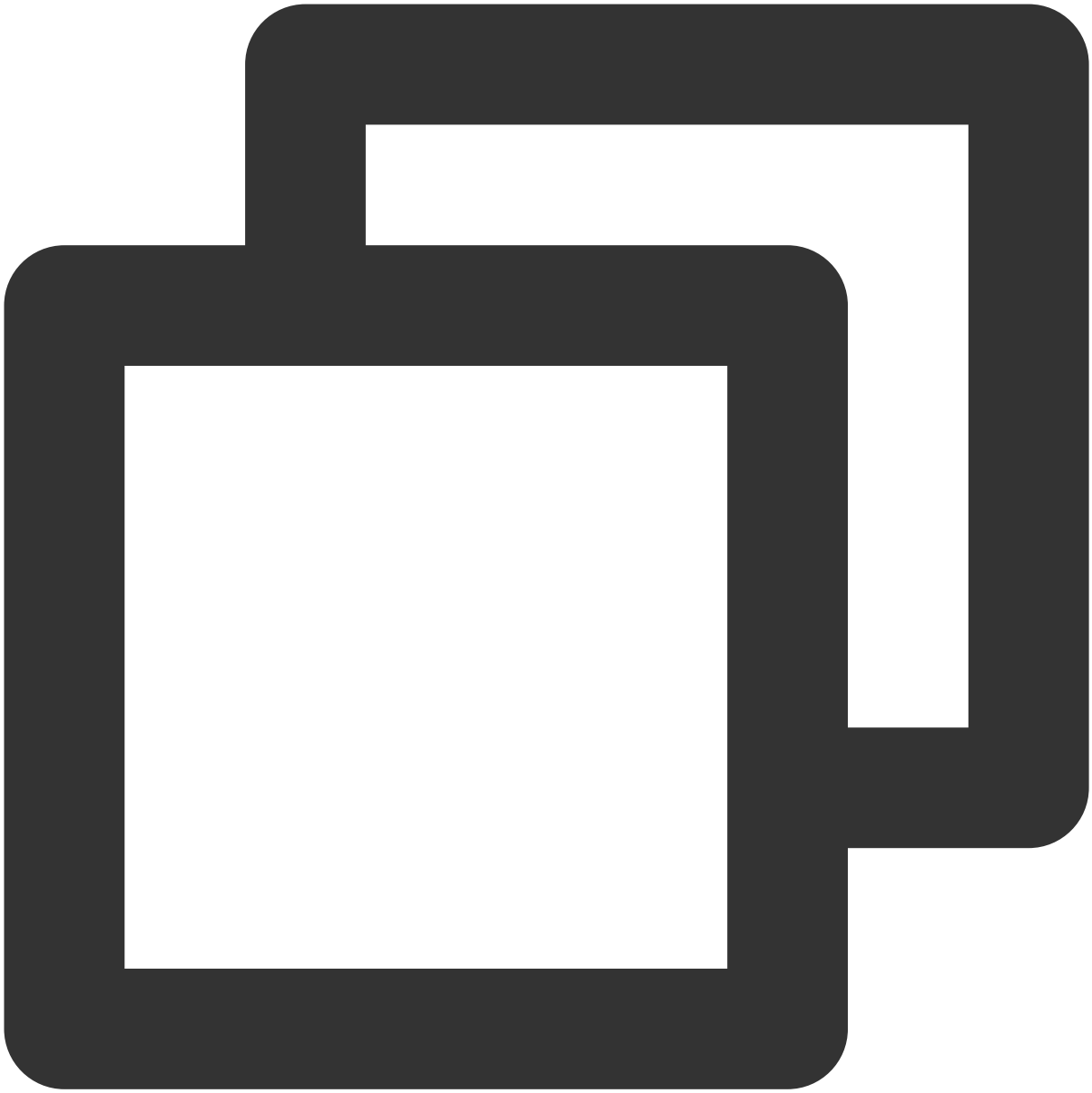
4. Search for and request songs.

Search for songs and acquire music resources through the business backend. Obtain identifiers such as the MusicId, the song's URL (MusicUrl), and the lyrics URL (LyricsUrl).

It is recommended that the business side select an appropriate music repository production to provide licensed music resources.

5. NTP synchronization.

©2013-2022 Tencent Cloud. All rights reserved.



```
TXLiveBase.setListener(new TXLiveBaseListener() {
    @Override
    public void onUpdateNetworkTime(int errCode, String errMsg) {
        super.onUpdateNetworkTime(errCode, errMsg);
        // errCode 0: Time synchronization successful and deviation within 30 ms. 1
        if (errCode == 0) {
            // Time synchronization successful and NTP timestamp obtained.
            long ntpTime = TXLiveBase.getNetworkTimestamp();
        } else {
            // If time synchronization fails, an attempt to resynchronize can be ma
            TXLiveBase.updateNetworkTime();
```

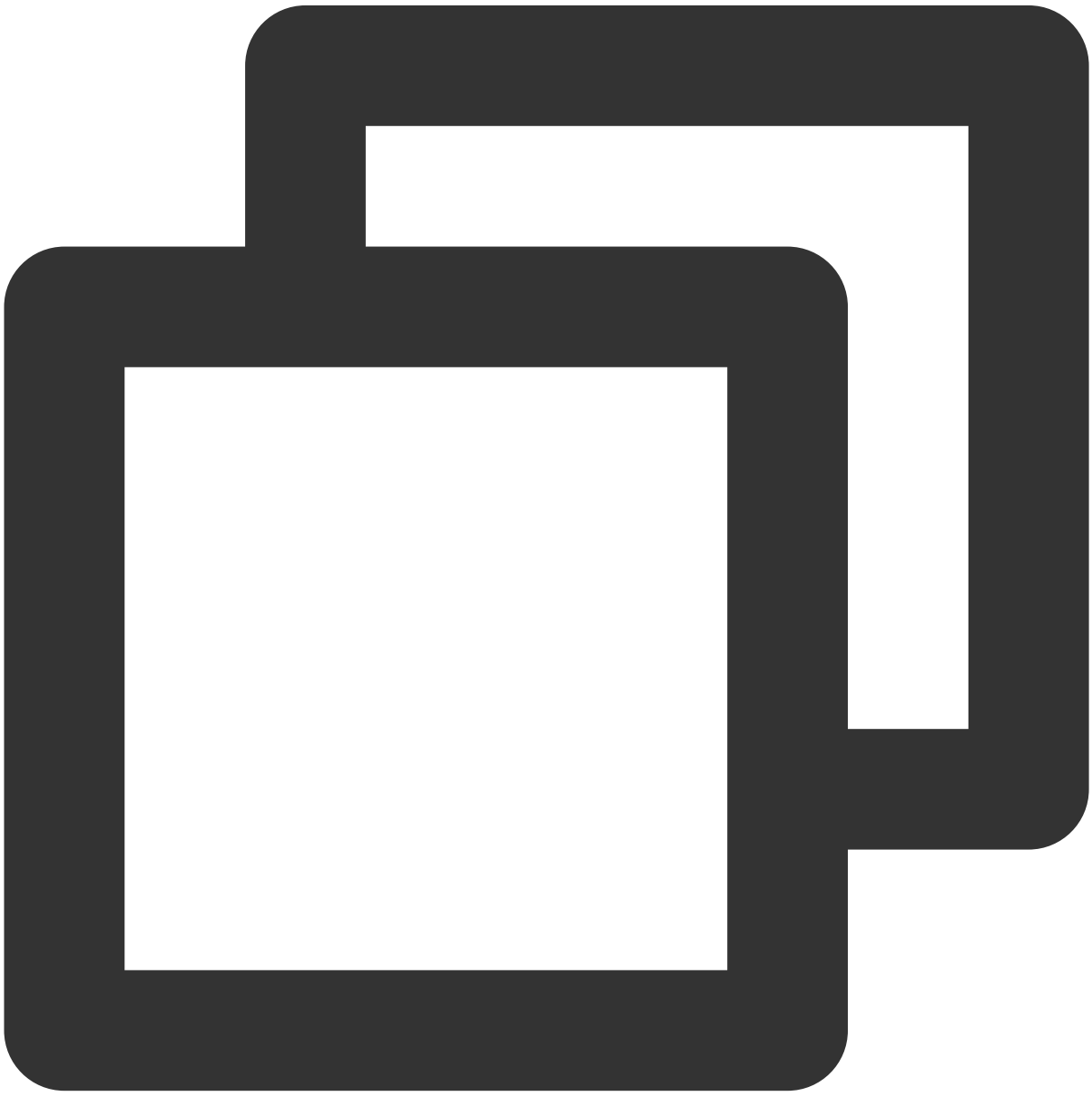
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        }
    });

    TXLiveBase.updateNetworkTime();
```

Note:

NTP time synchronization results can reflect the current network quality of the application user. To ensure a good chorus experience, it is recommended not to allow users to initiate chorus if time synchronization fails.

6. Send chorus signaling.



©2013-2022 Tencent Cloud. All rights reserved.

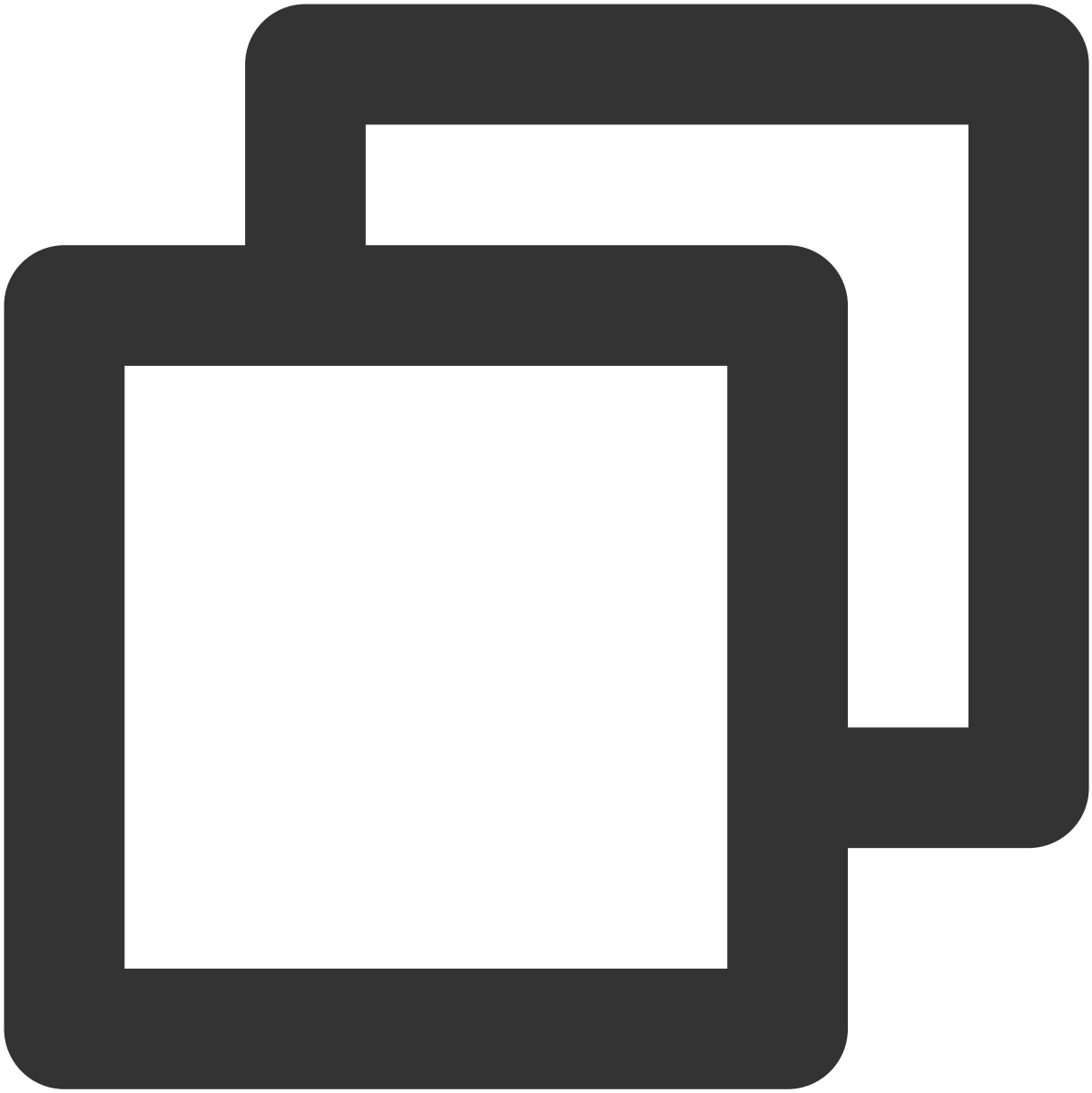
```
Timer mTimer = new Timer();
mTimer.schedule(new TimerTask() {
    @Override
    public void run() {
        try {
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("cmd", "startChorus");
            // Agreed chorus start time: Current NTP time + delayed playback time (
            jsonObject.put("startPlayMusicTS", TXLiveBase.getNetworkTimestamp() + 3
            jsonObject.put("musicId", musicId);
            jsonObject.put("musicDuration", subCloud.getAudioEffectManager().getMus
            mTRTCCloud.sendCustomCmdMsg(1, jsonObject.toString().getBytes(), false,
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}, 0, 1000);
```

Note:

The lead singer needs to cyclically broadcast chorus signaling to the room at a fixed time interval (e.g., every 1 second), so that new users who join mid-session can also participate in the chorus.

7. Load and play accompaniment.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain audio effects management.
TXAudioEffectManager mTXAudioEffectManager = subCloud.getAudioEffectManager();

// originMusicId: Custom identifier for the original vocal music. originMusicUrl: U
TXAudioEffectManager.AudioMusicParam originMusicParam = new TXAudioEffectManager.Au
// Publish original music to the remote.
originMusicParam.publish = true;
// Music start playing time point (in milliseconds).
originMusicParam.startTimeMS = 0;

// accompMusicId: Custom identifier for the accompaniment music. accompMusicUrl: UR
```

©2013-2022 Tencent Cloud. All rights reserved.

```
TXAudioEffectManager.AudioMusicParam accompMusicParam = new TXAudioEffectManager.Au
// Publish accompaniment music to the remote.
accompMusicParam.publish = true;
// Music start playing time point (in milliseconds).
accompMusicParam.startTimeMS = 0;

// Preload the original vocal music.
mTXAudioEffectManager.preloadMusic(originMusicParam);
// Preload the accompaniment music.
mTXAudioEffectManager.preloadMusic(accompMusicParam);

// Start playing the original vocal music after a delayed playback time (for exampl
mTXAudioEffectManager.startPlayMusic(originMusicParam);
// Start playing the accompaniment music after a delayed playback time (for example
mTXAudioEffectManager.startPlayMusic(accompMusicParam);

// Switch to the original vocal music.
mTXAudioEffectManager.setMusicPlayoutVolume(originMusicId, 100);
mTXAudioEffectManager.setMusicPlayoutVolume(accompMusicId, 0);
mTXAudioEffectManager.setMusicPublishVolume(originMusicId, 100);
mTXAudioEffectManager.setMusicPublishVolume(accompMusicId, 0);

// Switch to the accompaniment music.
mTXAudioEffectManager.setMusicPlayoutVolume(originMusicId, 0);
mTXAudioEffectManager.setMusicPlayoutVolume(accompMusicId, 100);
mTXAudioEffectManager.setMusicPublishVolume(originMusicId, 0);
mTXAudioEffectManager.setMusicPublishVolume(accompMusicId, 100);
```

Note:

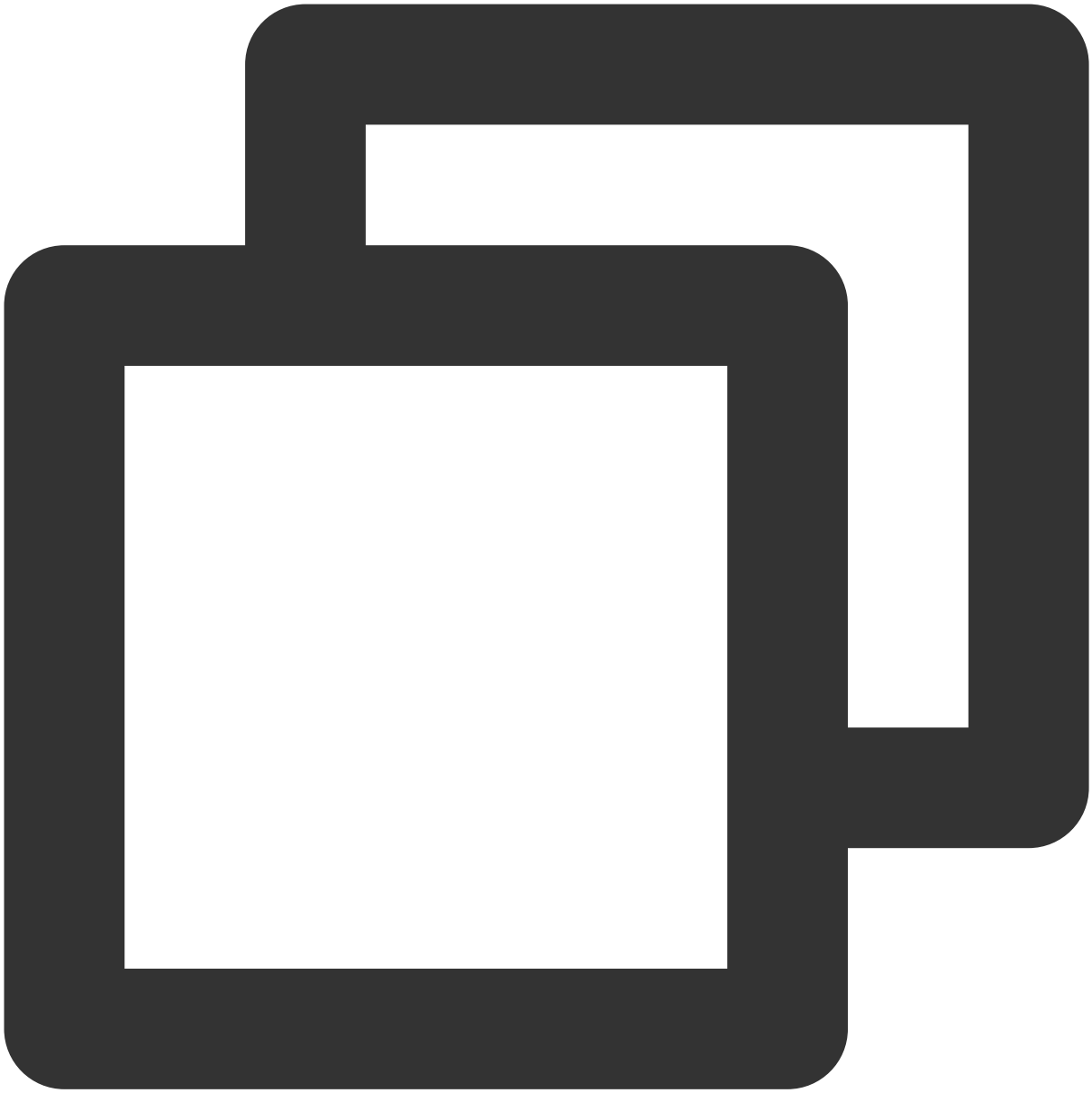
It is recommended to preload music before starting playback. By loading music resources into memory in advance, you can effectively reduce the load delay of music playback.

In karaoke scenarios, both the original vocal and accompaniment need to be played simultaneously (distinguished by MusicID). The switch between the original vocal and accompaniment is achieved by adjusting the local and remote playback volumes.

If the music being played has dual audio tracks (including both the original vocal and accompaniment), switching between them can be achieved by specifying the music's playback track using setMusicTrack.

8. Accompaniment Synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
// Agreed chorus start time.
long mStartPlayMusicTs = jsonObject.getLong("startPlayMusicTS");
// Actual playback progress of the current accompaniment music.
long currentProgress = subCloud.getAudioEffectManager().getMusicCurrentPosInMS(musi
// Ideal playback progress of the current accompaniment music.
long estimatedProgress = TXLiveBase.getNetworkTimestamp() - mStartPlayMusicTs;
// When the progress difference exceeds 50 ms, corrections are made.
if (estimatedProgress >= 0 && Math.abs(currentProgress - estimatedProgress) > 50) {
    subCloud.getAudioEffectManager().seekMusicToPosInMS(musicId, (int) estimatedPro
}
```

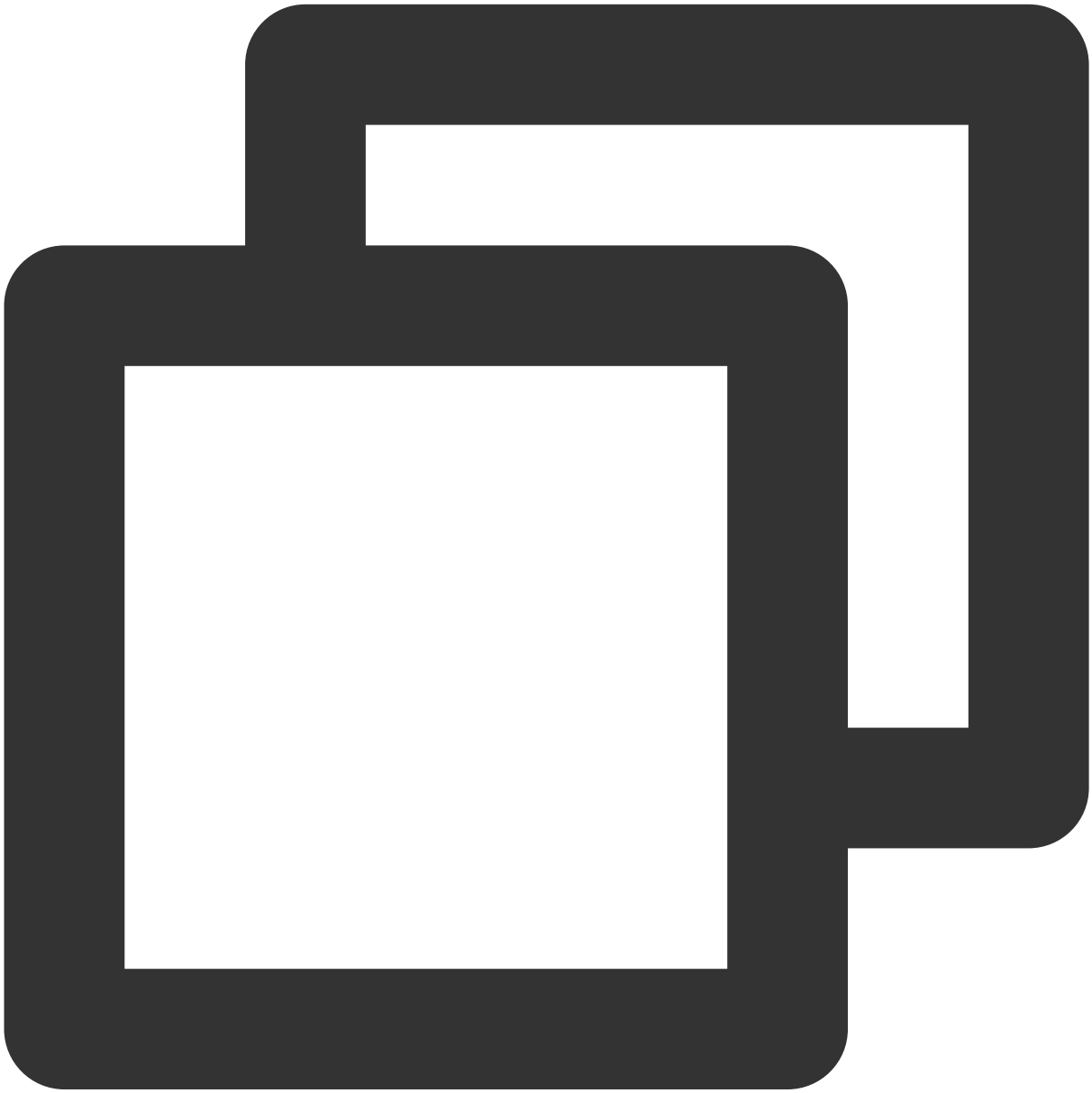
©2013-2022 Tencent Cloud. All rights reserved.

9. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Synchronize local lyrics, and transmit song progress via SEI.



```
mTXAudioEffectManager.setMusicObserver(musicId, new TXAudioEffectManager.TXMusicPla
    @Override
    public void onStart(int id, int errCode) {
        // Start playing music.
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onPlayProgress(int id, long curPtsMs, long durationMs) {
        // Determine whether seek is needed based on the latest progress and the lo
        // Song progress is transmitted by sending an SEI message.
        try {
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("musicId", id);
            jsonObject.put("progress", curPtsMs);
            jsonObject.put("duration", durationMs);
            mTRTCCloud.sendSEIMsg(jsonObject.toString().getBytes(), 1);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onComplete(int id, int errCode) {
        // Music playback completed.
    }
});
```

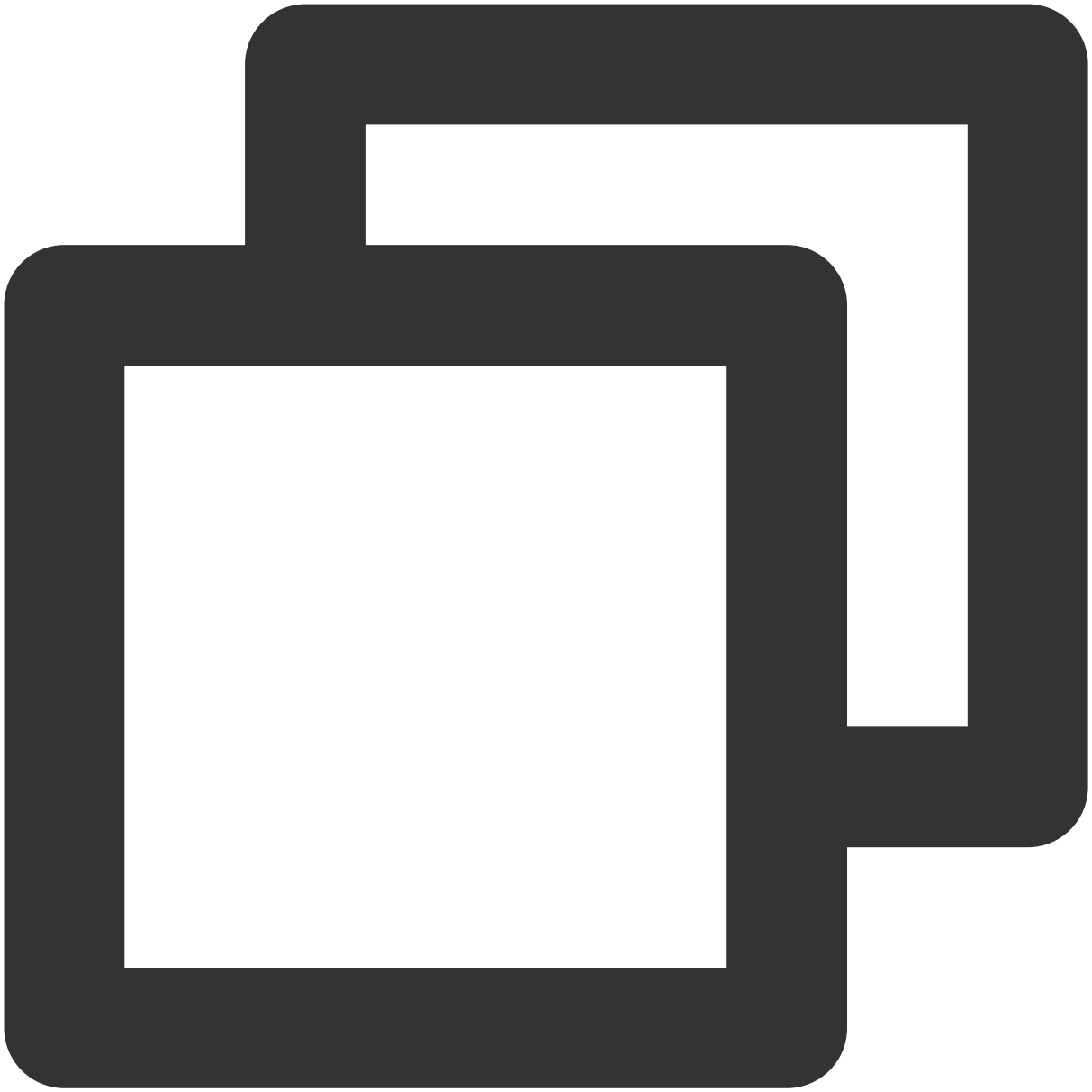
Note:

Ensure to set the playback event callback using this API before playing the background music. This allows to be aware of the background music's playback progress.

The frequency of the SEI messages sent by the performer is determined by the event callback frequency. Also, the playback progress can be actively synchronized on a schedule through getMusicCurrentPosInMS.

10. Become a listener and exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



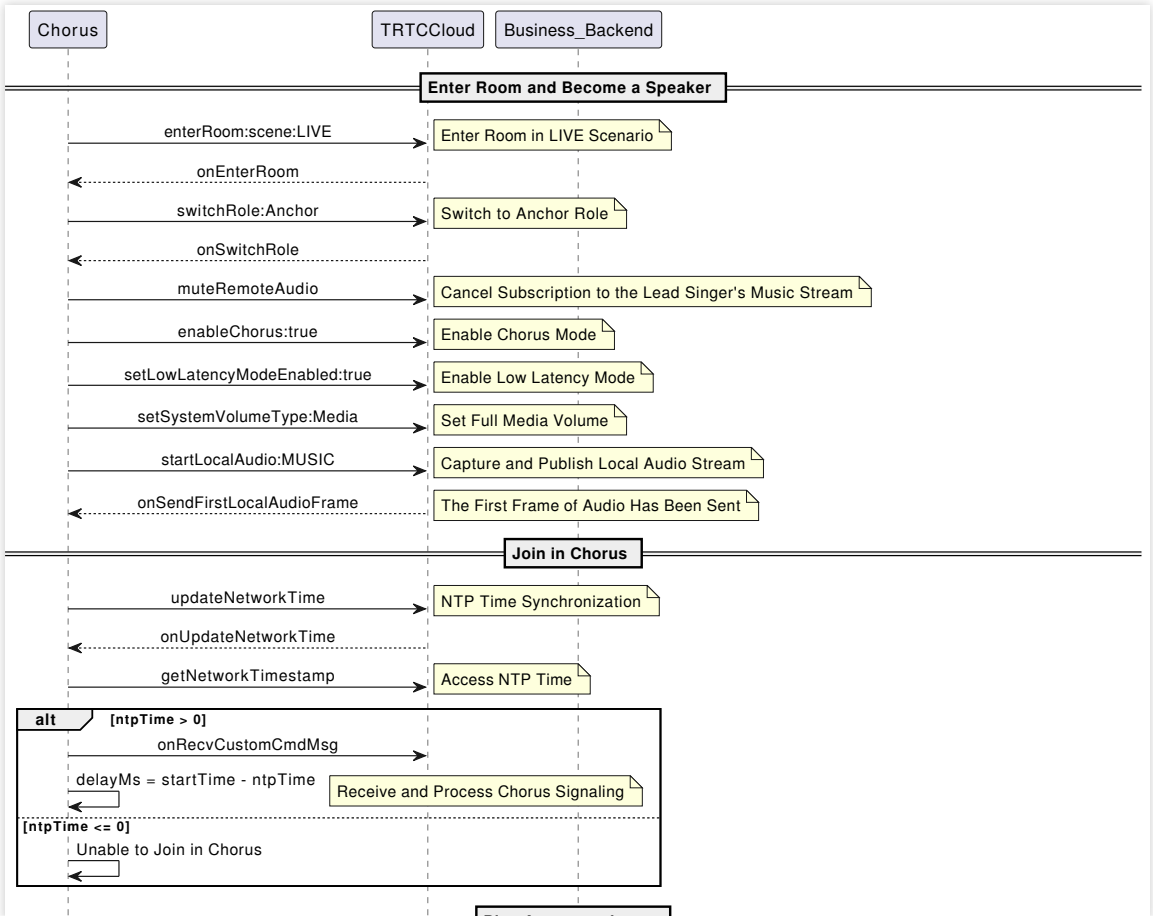
```
// The sub-instance uses the experimental API to disable chorus mode.
subCloud.callExperimentalAPI(
"{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":false,\\"audioSource\\":1
// The sub-instance uses the experimental API to disable low-latency mode.
subCloud.callExperimentalAPI(
"{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":false}}");
// The sub-instance switches to the audience role.
subCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);
// The sub-instance stops playing accompaniment music.
subCloud.getAudioEffectManager().stopPlayMusic(musicId);
// The sub-instance exits the room.
```

©2013-2022 Tencent Cloud. All rights reserved.

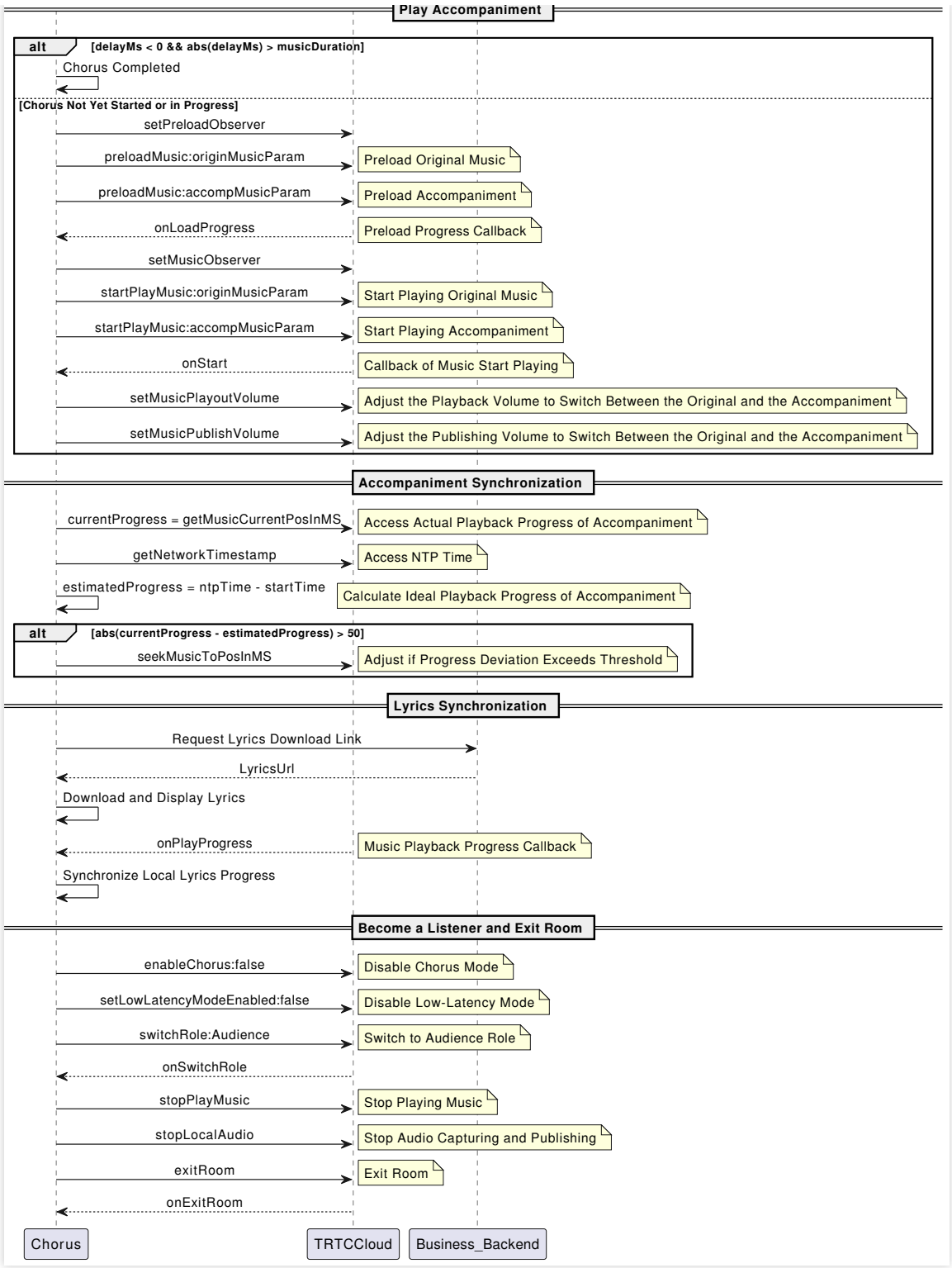
```
subCloud.exitRoom();

// The primary instance uses the experimental API to disable black frame insertion.
mTRTCCloud.callExperimentalAPI(
"{\\"api\\":\\"enableBlackStream\\",\\"params\\": {\\"enable\\":false}}");
// The primary instance uses the experimental API to disable chorus mode.
mTRTCCloud.callExperimentalAPI(
"{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":false,\\"audioSource\\":0
// The primary instance uses the experimental API to disable low-latency mode.
mTRTCCloud.callExperimentalAPI(
"{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":false}}");
// The primary instance switches to the audience role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);
// The primary instance stops local audio capture and publishing.
mTRTCCloud.stopLocalAudio();
// The primary instance exits the room.
mTRTCCloud.exitRoom();
```

Perspective 2: Chorus actions

Sequence diagram

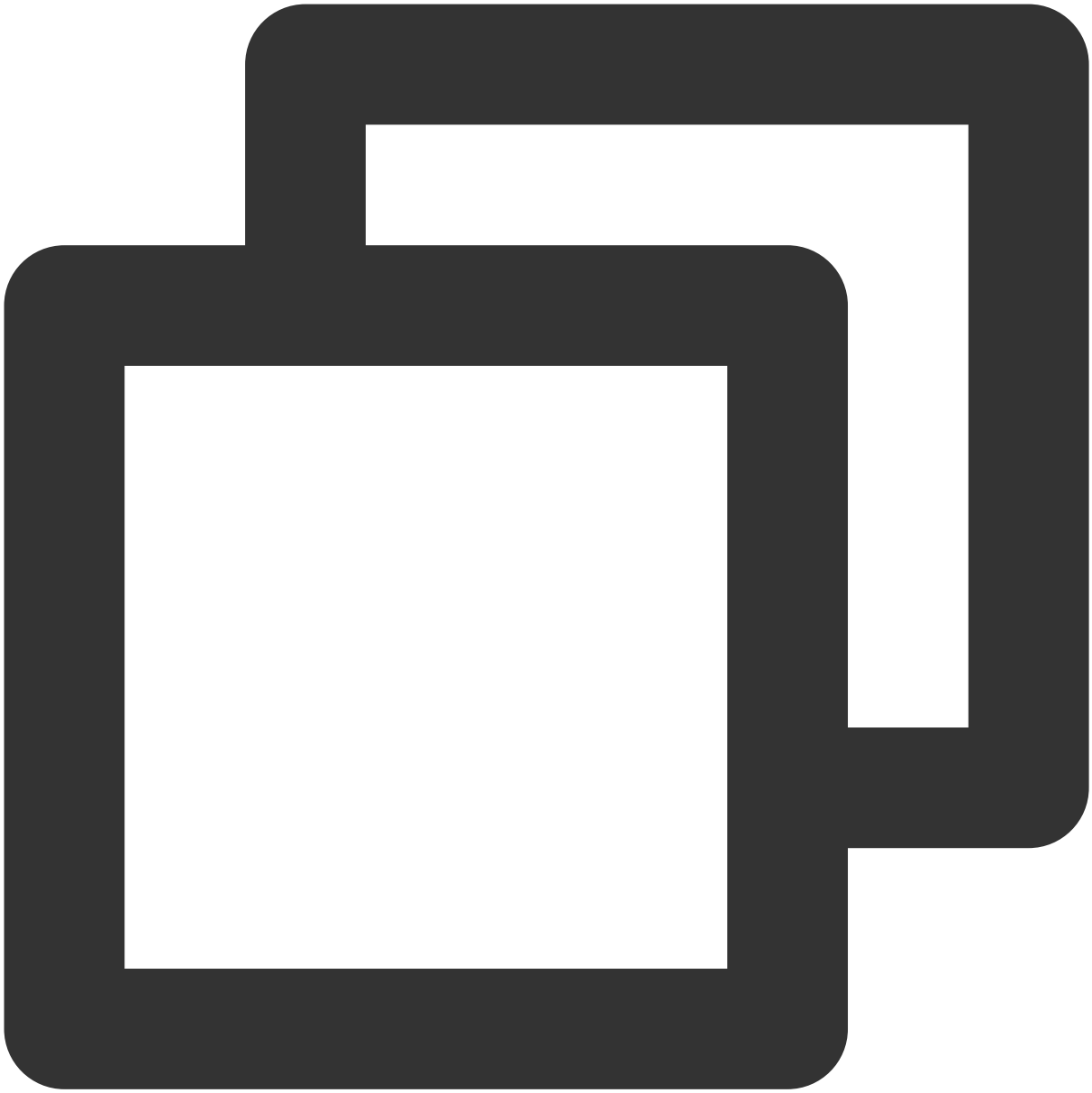


©2013-2022 Tencent Cloud. All rights reserved.



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



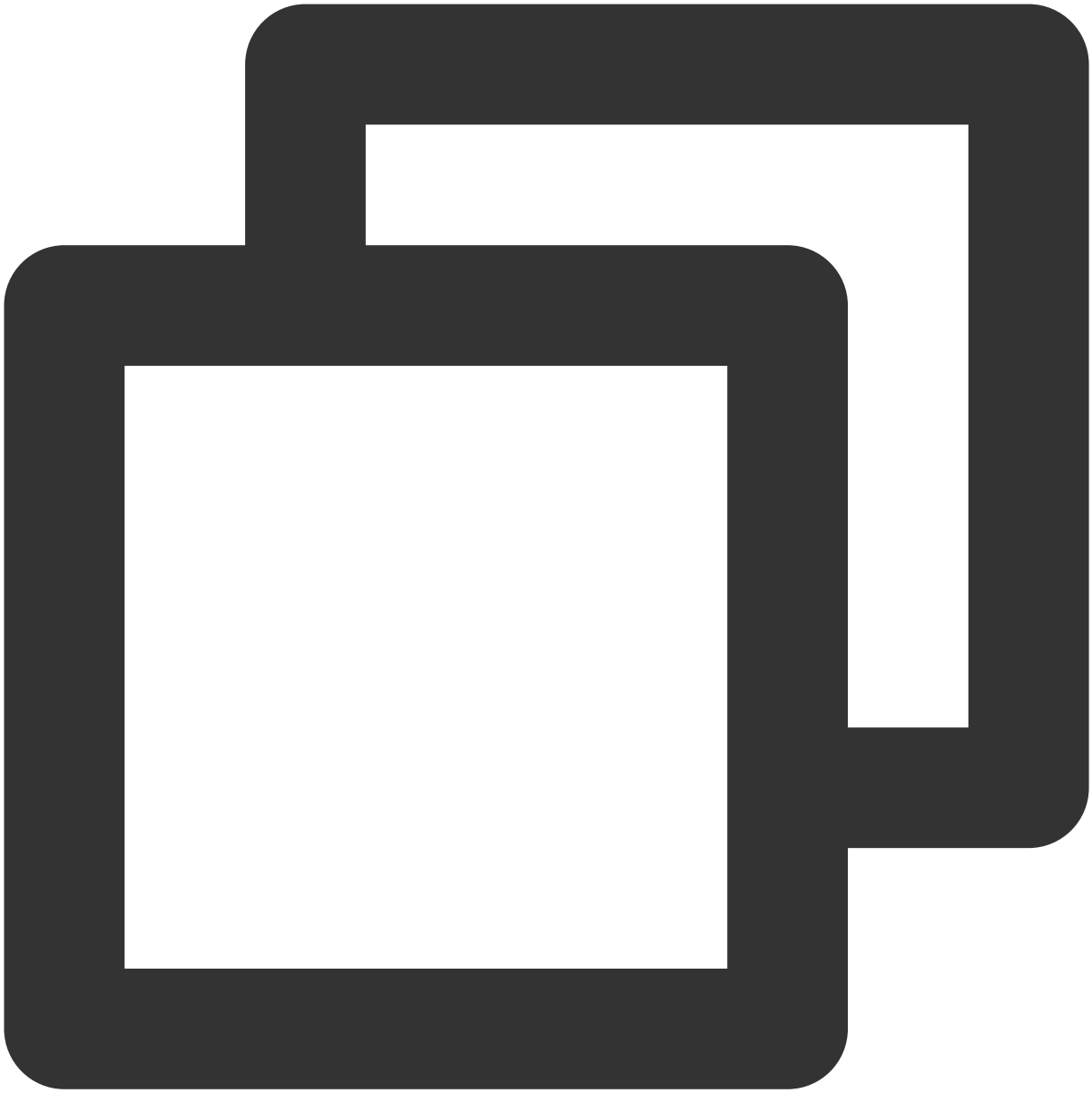
```
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Example of entering the room as an audience role.
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

2. Go live on streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switched to the anchor role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);

// Event callback for switching the role.
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Cancel subscription to music streams published by the lead singer sub-in
        mTRTCCloud.muteRemoteAudio(mBgmUserId, true);
        // Use the experimental API to enable chorus mode.
        mTRTCCloud.callExperimentalAPI(
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":true,\\"audioSour
        // Use the experimental API to enable low-latency mode.
        mTRTCCloud.callExperimentalAPI(
        "{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":true}
        // Set media volume type.
        mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);
        // Upstream local audio streams and set audio quality.
        mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
    }
}
```

Note:

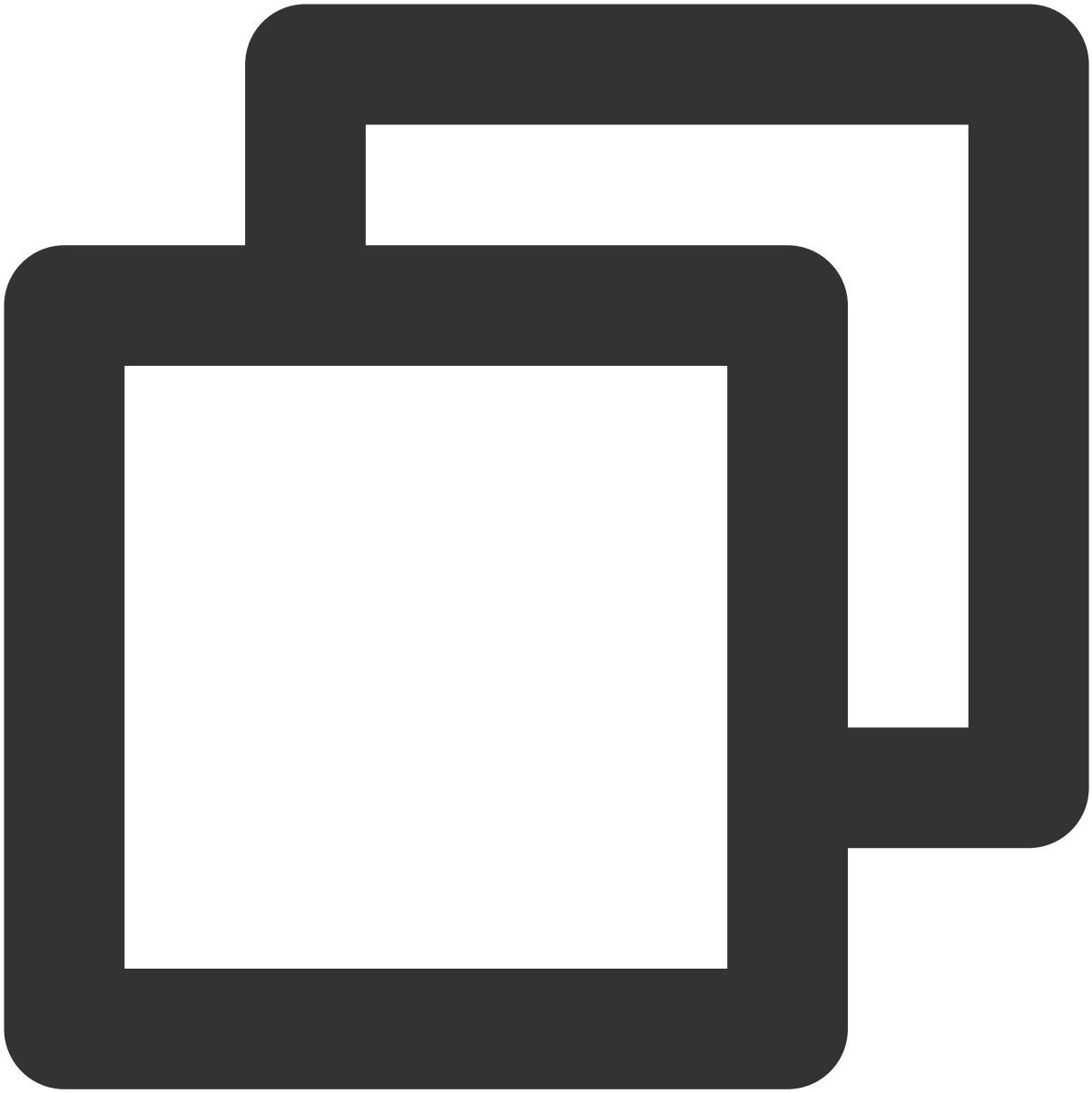
To minimize delay, all chorus members play the accompaniment music locally. Therefore, it is necessary to cancel subscriptions to music streams published by the lead singer.

Chorus members also need to use the experimental API to enable chorus mode and low-latency mode to optimize the chorus experience.

In karaoke scenarios, it is recommended to set the full-range media volume and music quality to achieve a high-fidelity listening experience.

3. NTP synchronization.

©2013-2022 Tencent Cloud. All rights reserved.



```
TXLiveBase.setListener(new TXLiveBaseListener() {
    @Override
    public void onUpdateNetworkTime(int errCode, String errMsg) {
        super.onUpdateNetworkTime(errCode, errMsg);
        // errCode 0: Time synchronization successful and deviation within 30 ms. 1
        if (errCode == 0) {
            // Time synchronization successful and NTP timestamp obtained.
            long ntpTime = TXLiveBase.getNetworkTimestamp();
        } else {
            // If time synchronization fails, an attempt to resynchronize can be ma
            TXLiveBase.updateNetworkTime();
```

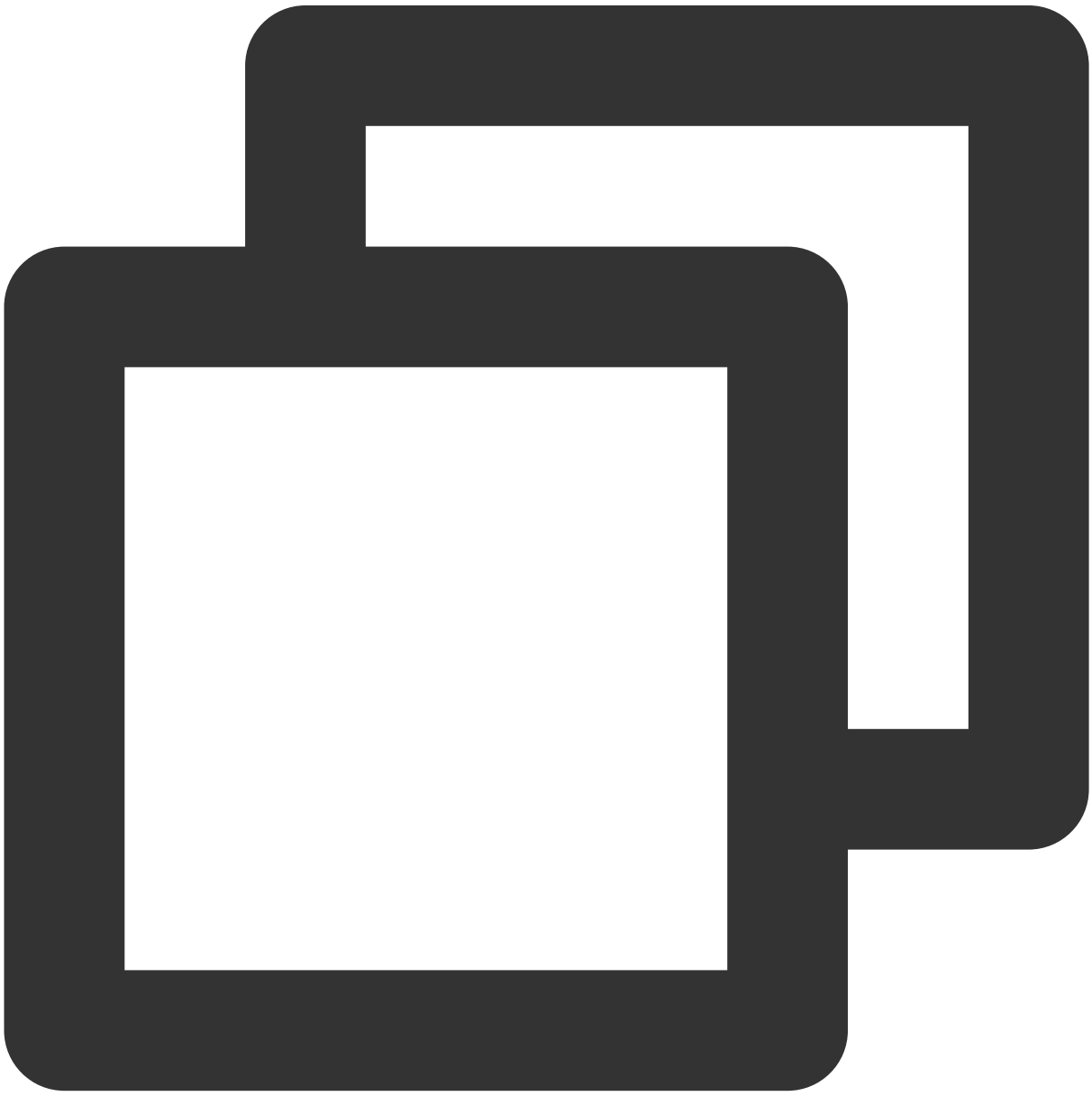
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        }
    });

    TXLiveBase.updateNetworkTime();
```

Note:

NTP time synchronization results can reflect the current network quality of the application user. To ensure a good chorus experience, it is recommended not to allow users to participate in the chorus if time synchronization fails.

4. Receive chorus signaling.



©2013-2022 Tencent Cloud. All rights reserved.

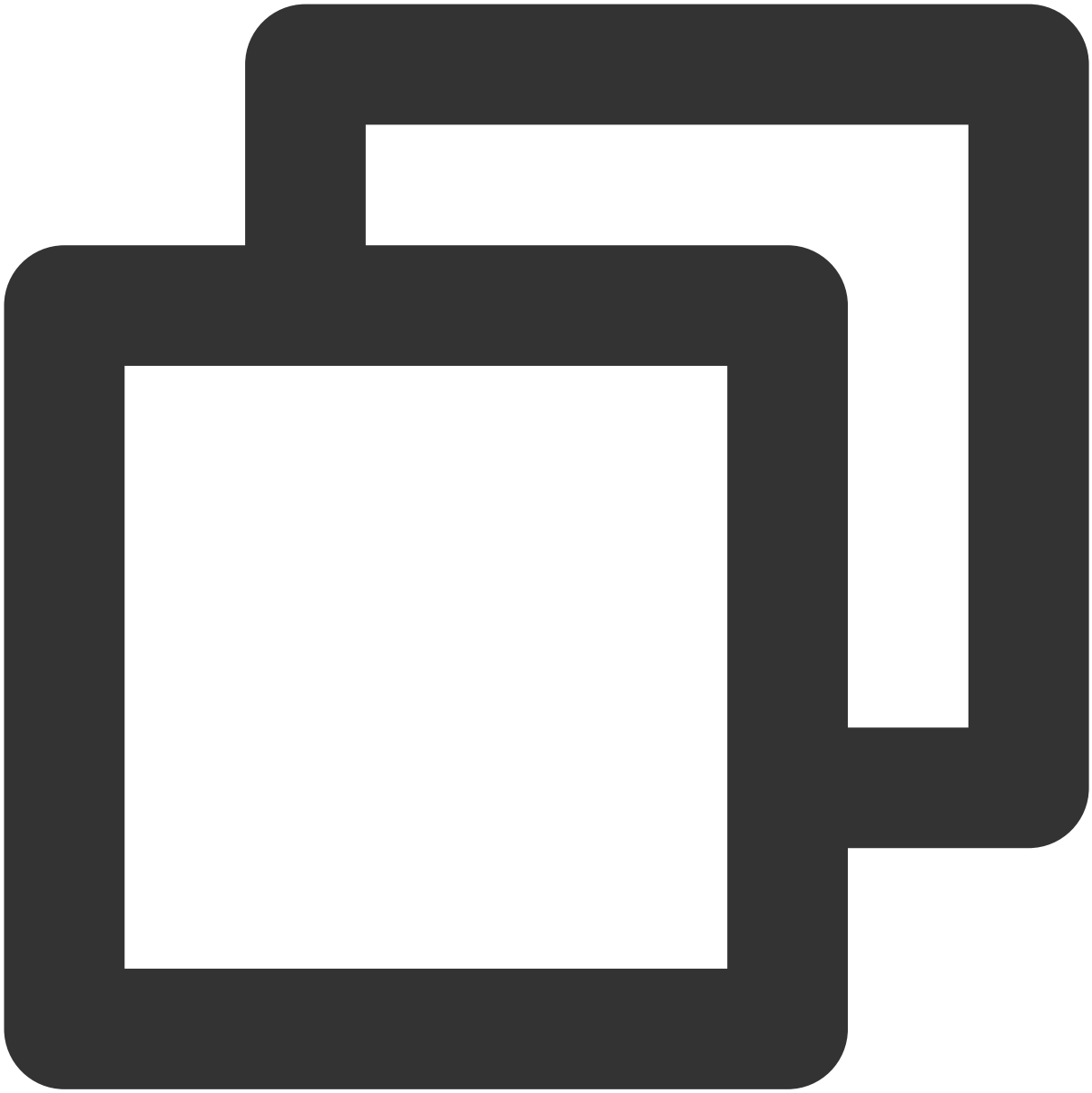
```
@Override
public void onRecvCustomCmdMsg(String userId, int cmdID, int seq, byte[] message) {
    try {
        JSONObject json = new JSONObject(new String(message, "UTF-8"));
            // Match the chorus signaling.
        if (json.getString("cmd").equals("startChorus")) {
            long startPlayMusicTs = json.getLong("startPlayMusicTS");
            int musicId = json.getInt("musicId");
            long musicDuration = json.getLong("musicDuration");
            // Agree on the time difference between chorus time and current time.
            long delayMs = startPlayMusicTs - TXLiveBase.getNetworkTimestamp();
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Note:

Once the chorus members receive the chorus signaling and join in, the status should be changed to Chorus In Progress. Chorus signaling would not be responded to again before the end of this chorus round.

5. Play accompaniment, and start the chorus.

©2013-2022 Tencent Cloud. All rights reserved.



```
if (delayMs > 0) {    // The chorus has not started.
    // Begin to preload music.
    preloadMusic(musicId, 0L);
    // Play music after a delay of delayMs.
    startPlayMusic(musicId, 0L);
} else if (Math.abs(delayMs) < musicDuration) {    // The chorus is in progress.
    // Play start time: Absolute value of the time difference + preload delay (e.g.
    long startTimeMS = Math.abs(delayMs) + 400;
    // Begin to preload music.
    preloadMusic(musicId, startTimeMS);
    // Start playing music after a preload delay (e.g., 400 ms).
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    startPlayMusic(musicId, startTimeMS);
} else {    // The chorus has ended.
    // Joining the chorus is not allowed.
}

// Preload music.
public void preloadMusic(int musicId, long startTimeMS) {
    // musicId: Obtained from chorus signaling. musicUrl: Corresponding music resou
    TXAudioEffectManager.AudioMusicParam musicParam = new
    TXAudioEffectManager.AudioMusicParam(musicId, musicUrl);
    // Only local music playback.
    musicParam.publish = false;
    // Music start playing time point (in milliseconds).
    musicParam.startTimeMS = startTimeMS;

    mTRTCCloud.getAudioEffectManager().preloadMusic(musicParam);
}

// Begin to play music.
public void startPlayMusic(int musicId, long startTimeMS) {
    // musicId: Obtained from chorus signaling. musicUrl: Corresponding music resou
    TXAudioEffectManager.AudioMusicParam musicParam = new
    TXAudioEffectManager.AudioMusicParam(musicId, musicUrl);
    // Only local music playback.
    musicParam.publish = false;
    // Music start playing time point (in milliseconds).
    musicParam.startTimeMS = startTimeMS;

    mTRTCCloud.getAudioEffectManager().startPlayMusic(musicParam);
}
```

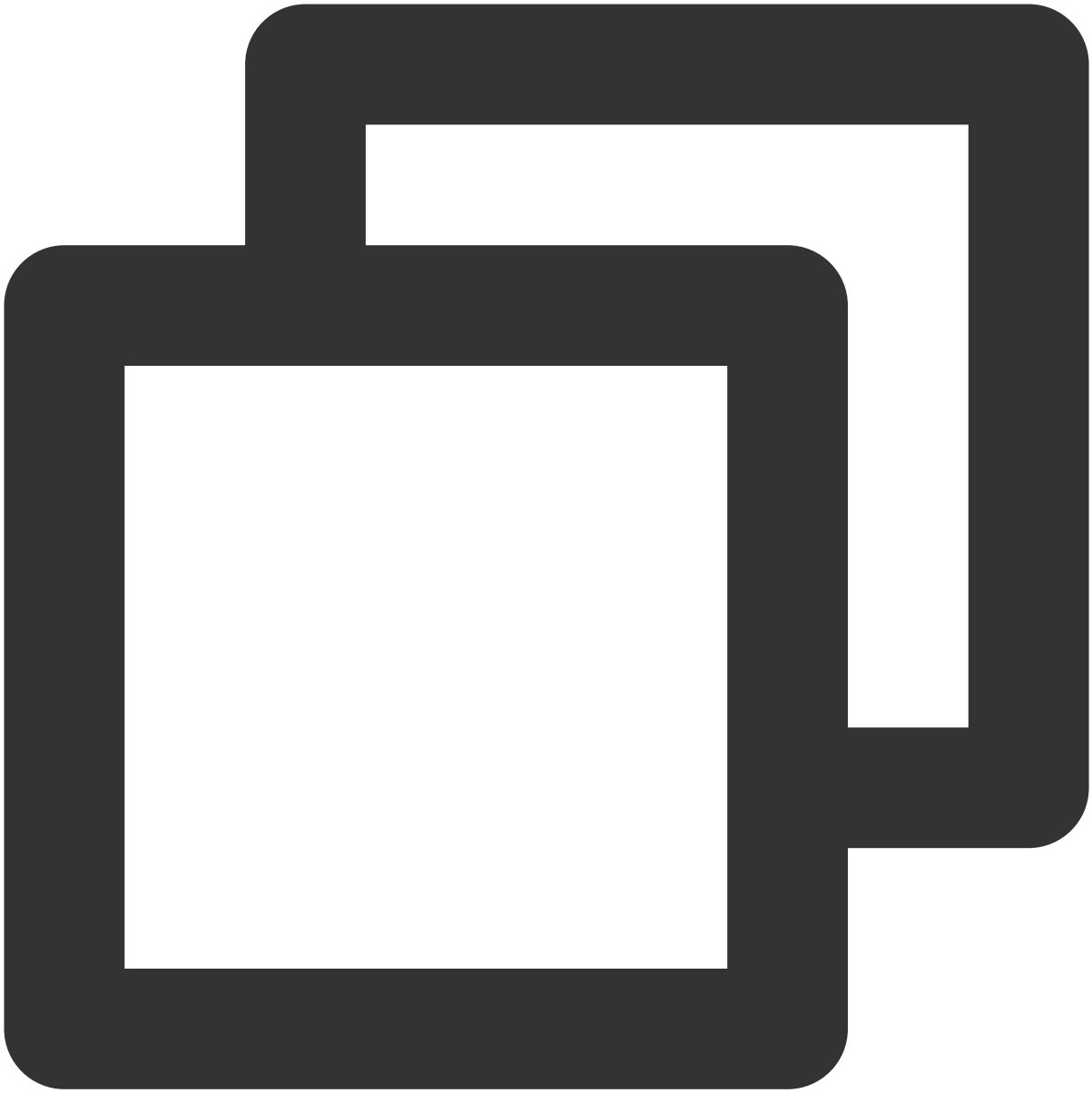
Note:

To minimize transmission delay as much as possible, chorus members perform along with the local playback of accompaniment music, and they do not need to publish or receive remote music.

Based on `delayMs`, the current chorus status can be determined. Developers must implement the `startPlayMusic` delayed call for different statuses on their own.

6. Accompaniment Synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
// Agreed chorus start time.
long mStartPlayMusicTs = jsonObject.getLong("startPlayMusicTS");
// Actual playback progress of the current accompaniment music.
long currentProgress = mTRTCCloud.getAudioEffectManager().getMusicCurrentPosInMS(mu
// Ideal playback progress of the current accompaniment music.
long estimatedProgress = TXLiveBase.getNetworkTimestamp() - mStartPlayMusicTs;
// When the progress difference exceeds 50 ms, corrections are made.
if (estimatedProgress >= 0 && Math.abs(currentProgress - estimatedProgress) > 50) {
    mTRTCCloud.getAudioEffectManager().seekMusicToPosInMS(musicId, (int) estimatedP
}
```

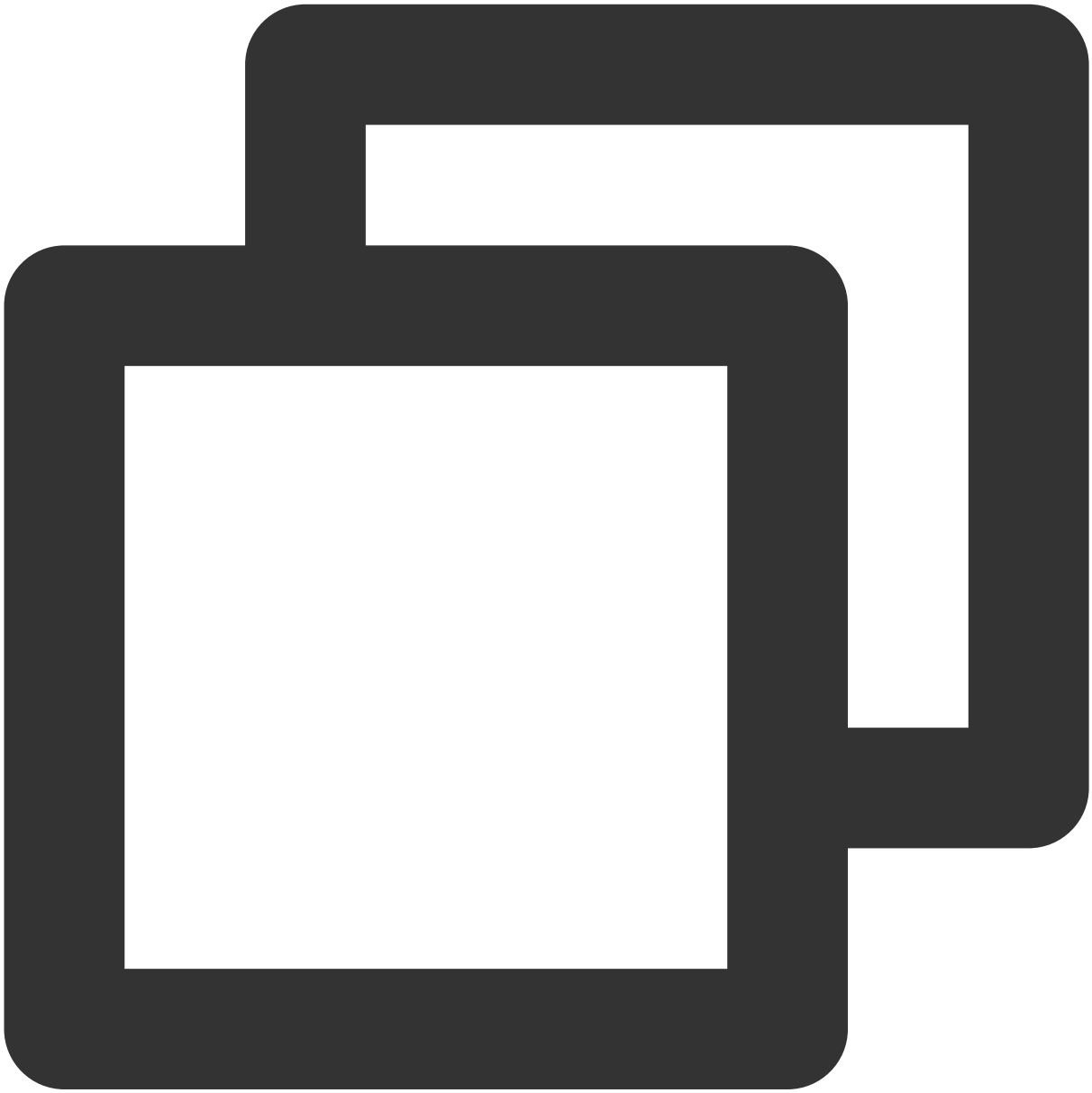
©2013-2022 Tencent Cloud. All rights reserved.

7. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Local lyric synchronization.



```
mTXAudioEffectManager.setMusicObserver(musicId, new TXAudioEffectManager.TXMusicPla
    @Override
    public void onStart(int id, int errCode) {
        // Start playing music.
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

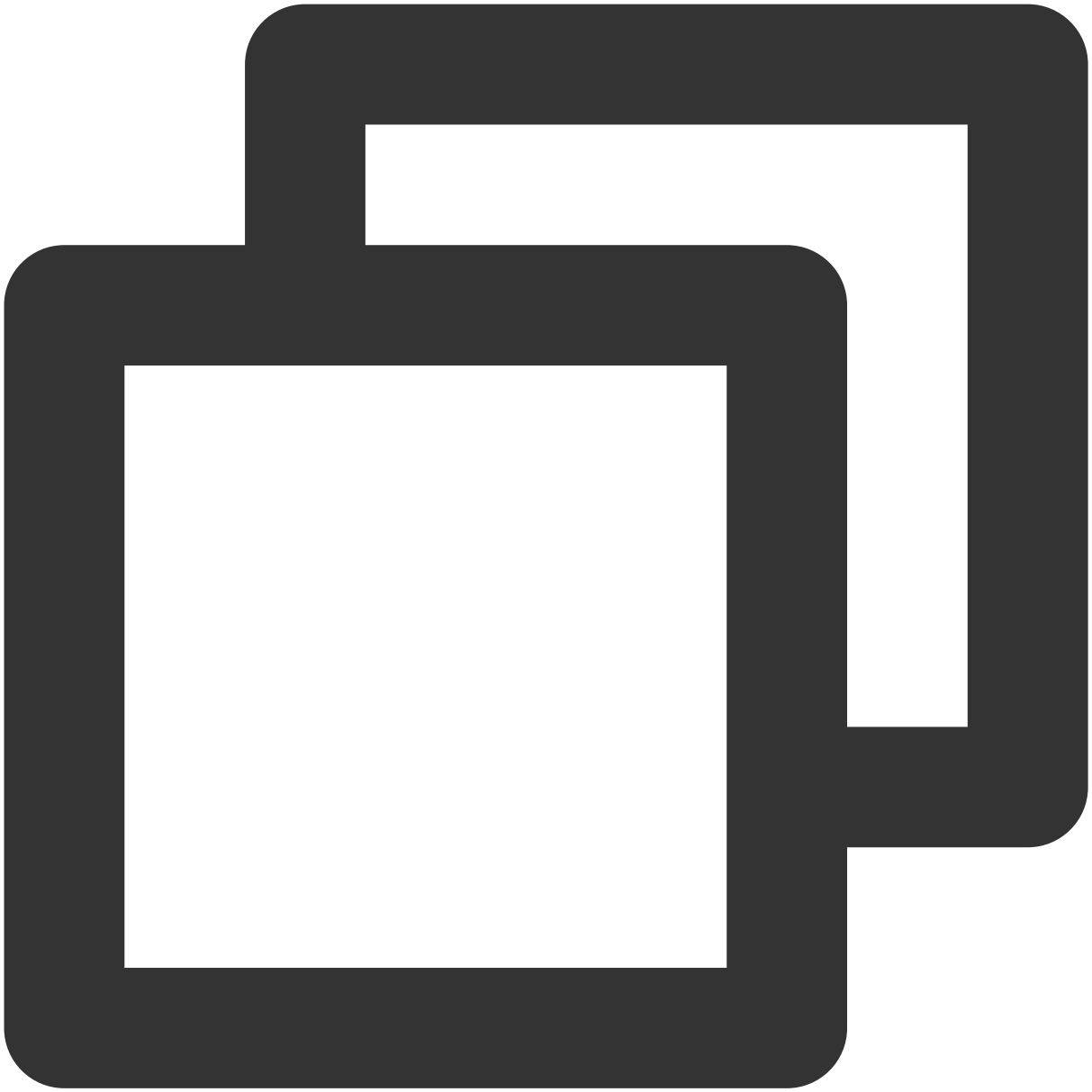
```
    @Override
    public void onPlayProgress(int id, long curPtsMs, long durationMs) {
        // TODO: The logic of updating the lyric control.
        // Determine whether seek in the lyrics control is needed based on the late

    }

    @Override
    public void onComplete(int id, int errCode) {
        // Music playback completed.
    }
});
```

Note:

Ensure to set the playback event callback using this API before playing the background music. This allows to be aware of the background music's playback progress.

8. Become a listener and exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



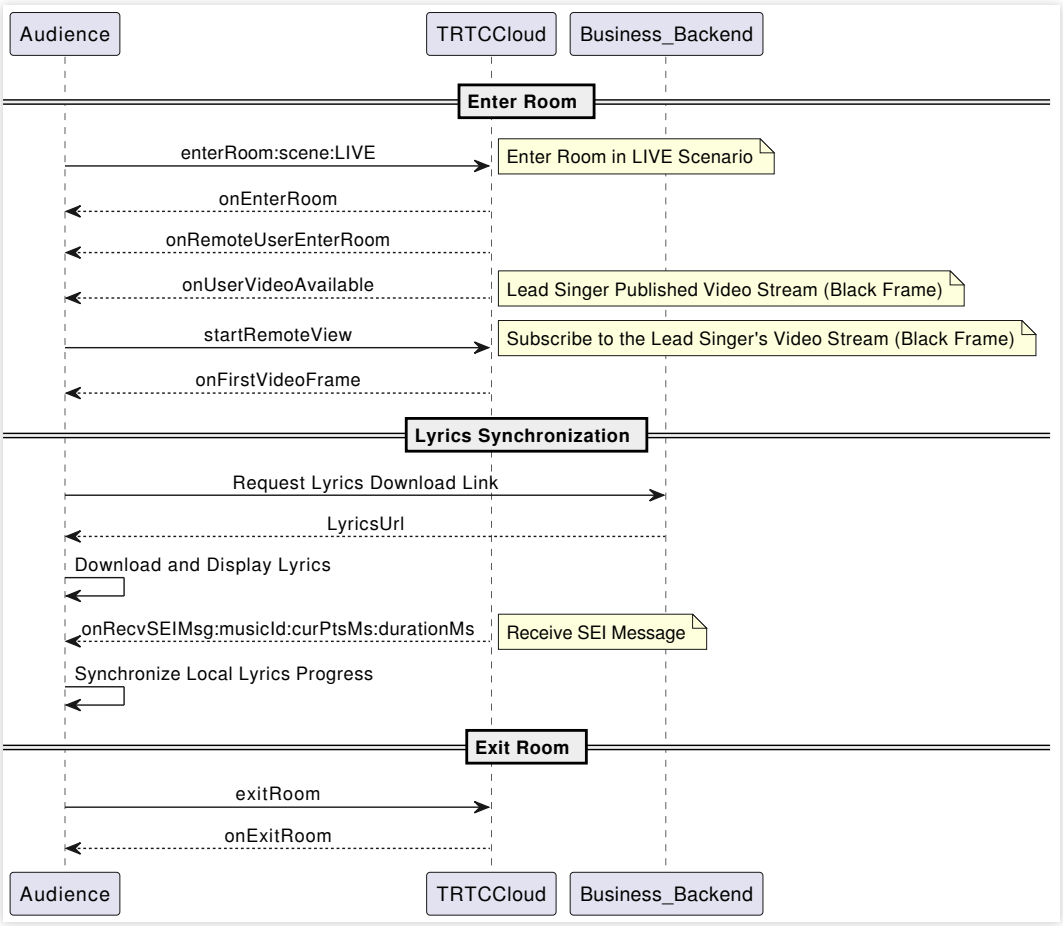
```
// Use the experimental API to disable chorus mode.
mTRTCCloud.callExperimentalAPI(
"{\\"api\\":\\"enableChorus\\",\\"params\\":{\\"enable\\":false,\\"audioSource\\":0
// Use the experimental API to disable low-latency mode.
mTRTCCloud.callExperimentalAPI(
"{\\"api\\":\\"setLowLatencyModeEnabled\\",\\"params\\":{\\"enable\\":false}}");
// Switched to the audience role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);
// Stop playing accompaniment music.
mTRTCCloud.getAudioEffectManager().stopPlayMusic(musicId);
// Stop local audio capture and publishing.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
mTRTCCloud.stopLocalAudio();
// Exit the room.
mTRTCCloud.exitRoom();
```

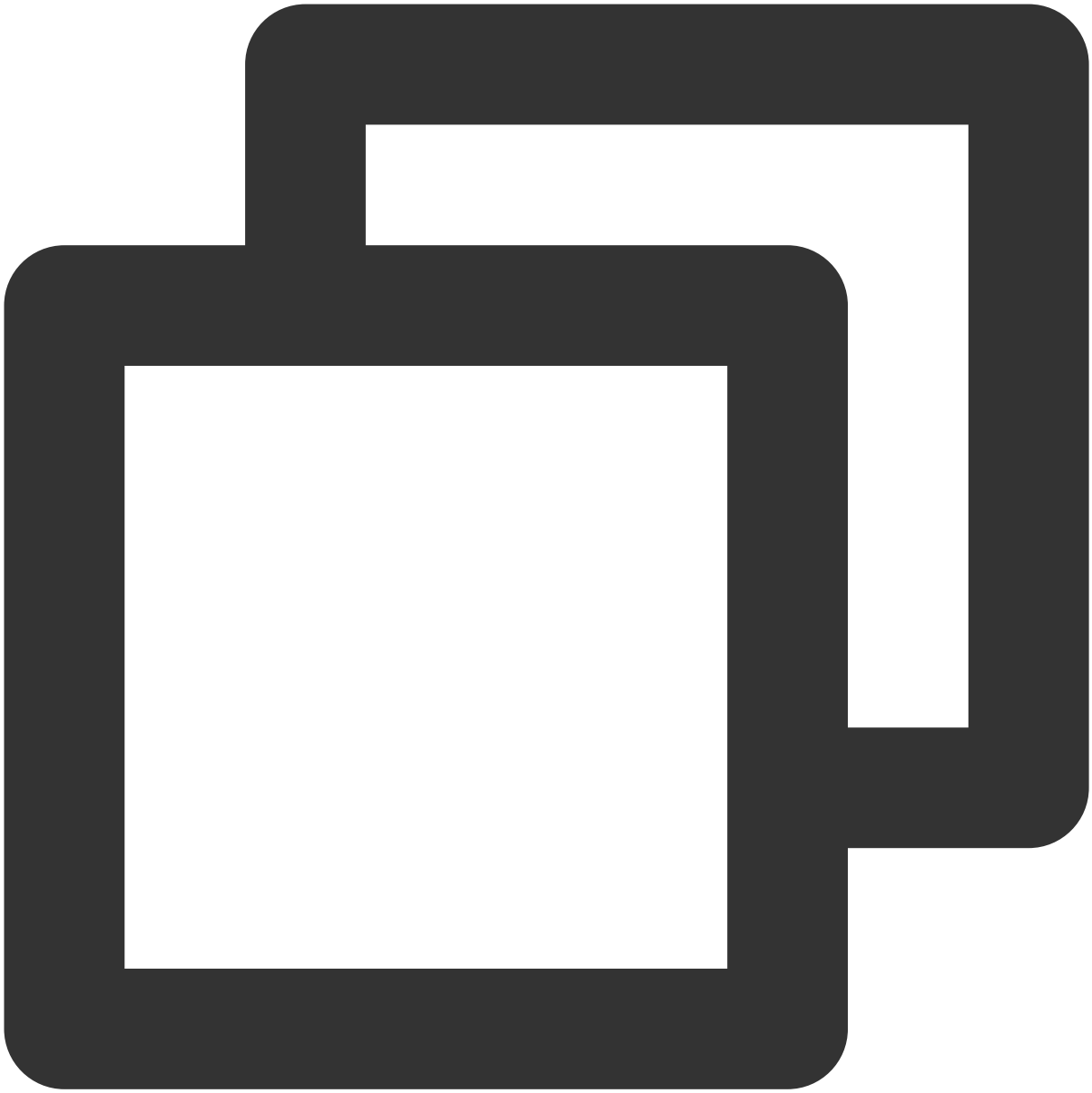
Perspective 3: Listener actions

Sequence diagram



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

To better transmit SEI messages for lyrics synchronization, it is recommended to choose `TRTC_APP_SCENE_LIVE` for room-entry scenarios.

Under the automatic subscription mode (default), audiences automatically subscribe and play the on-mic anchor's audio and video streams upon entering the room.

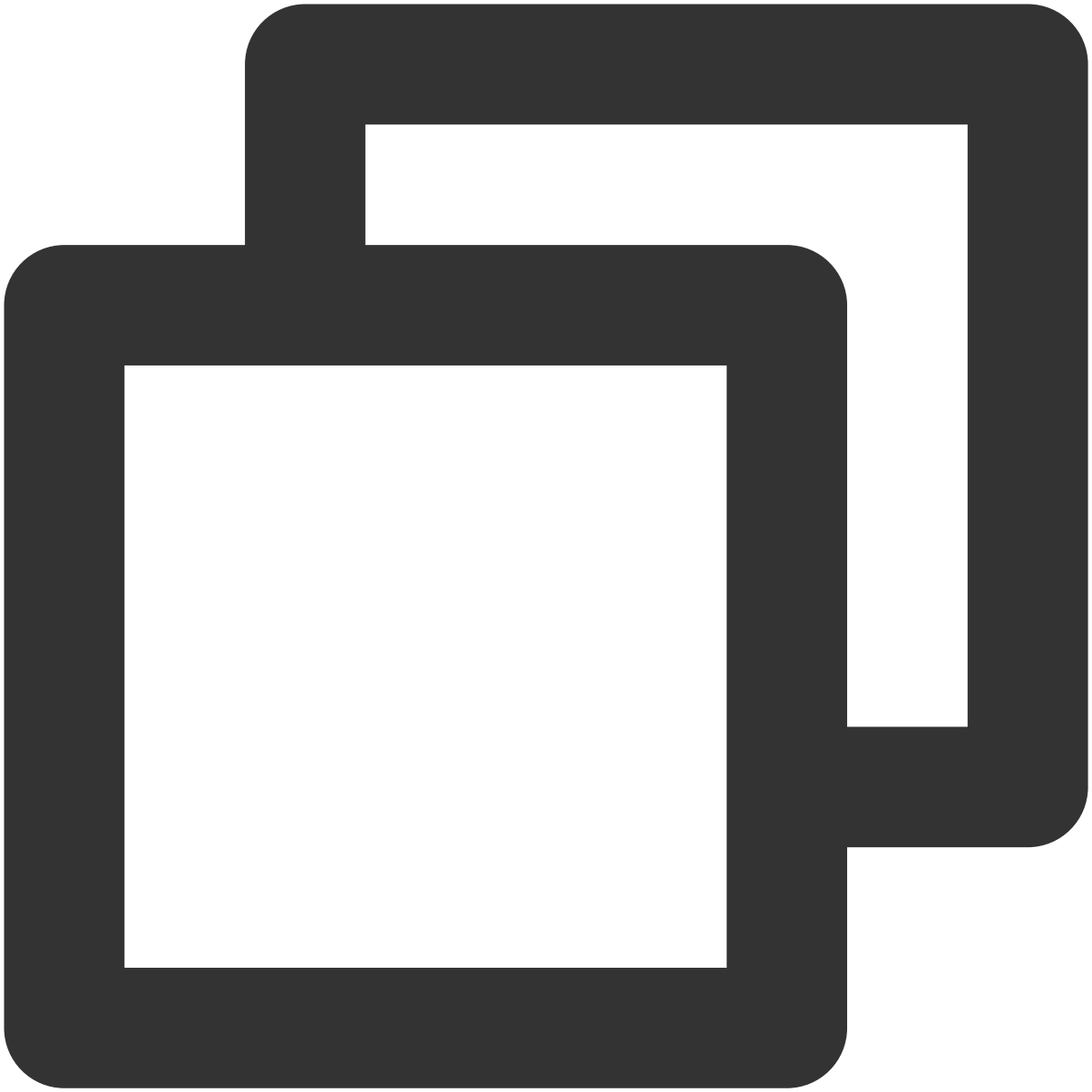
2. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Listener end lyric synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onUserVideoAvailable(String userId, boolean available) {
    if (available) {
        mTRTCCloud.startRemoteView(userId, null);
    } else {
        mTRTCCloud.stopRemoteView(userId);
    }
}

@Override
public void onRecvSEIMsg(String userId, byte[] data) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    String result = new String(data);
    try {
        JSONObject jsonObject = new JSONObject(result);
        int musicId = jsonObject.getInt("musicId");
        long progress = jsonObject.getLong("progress");
        long duration = jsonObject.getLong("duration");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    ...
    // TODO: The logic of updating the lyric control.
    // Based on the received latest progress and the local lyrics progress deviatio
    ...
}
```

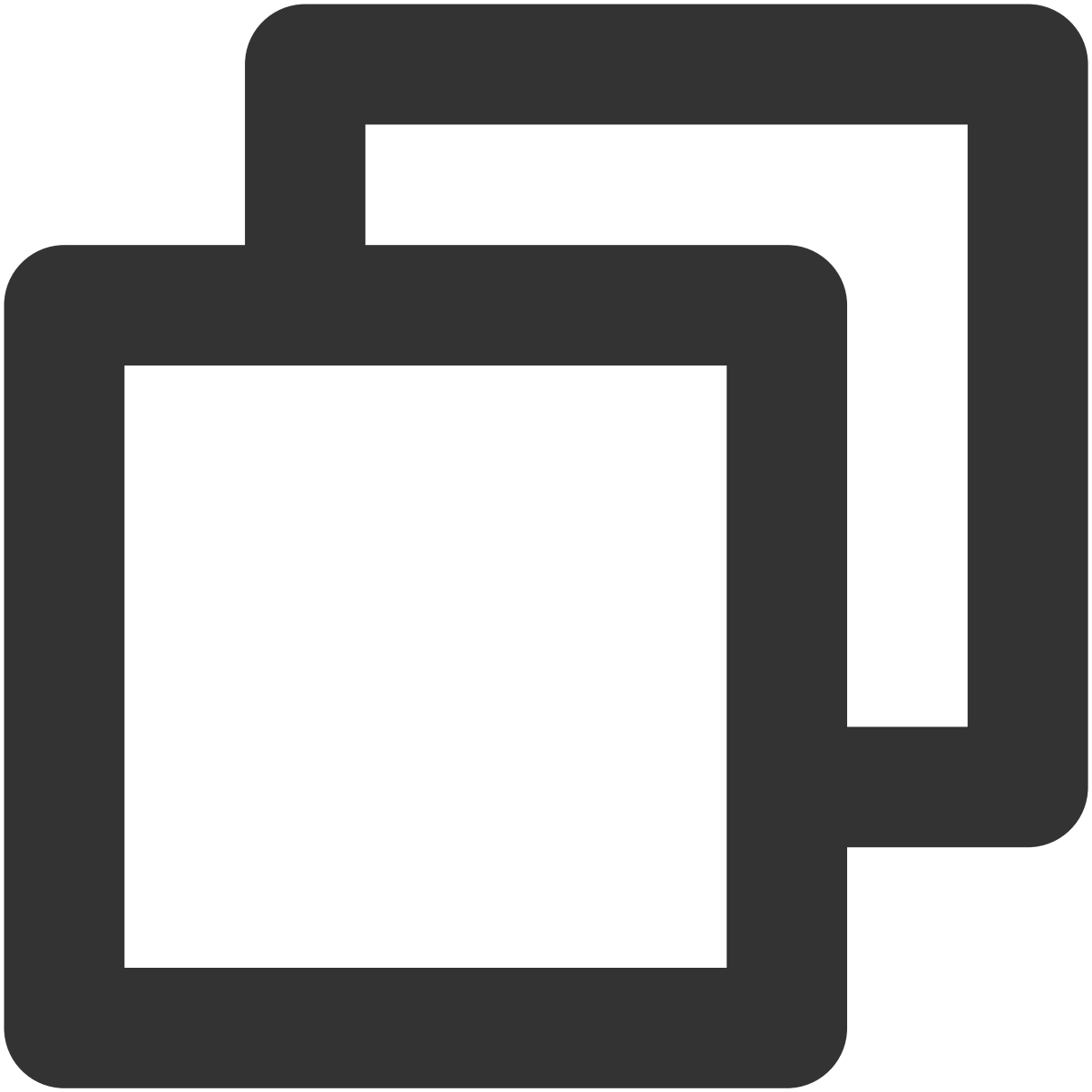
Note:

Listeners need to actively subscribe to the lead singer's video streams in order to receive the SEI messages carried by black frames.

If the lead singer's mixed stream also mixes in black frames, then only subscribing to the mixing stream robot's video stream is required.

3. Exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Exit the room.
mTRTCCloud.exitRoom();

// Exit room event callback.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server.");
    } else if (reason == 2) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            Log.d(TAG, "The current room has been dissolved.");
        }
    }
```

Advanced Features

Music scoring module integration

Music scoring provides users with multi-dimensional singing scoring capabilities. Currently, supported scoring dimensions include intonation and rhythm.

1. Prepare scoring-related files.

Prepare in advance the performance recording files to be scored, original music standard files, MIDI pitch files, and upload them to COS storage.

2. Create a music scoring task.

Request Method: POST(HTTP).

Request Address: http://service-mqk0mc83-1257411467.bj.apigw.tencentcs.com/release/job.

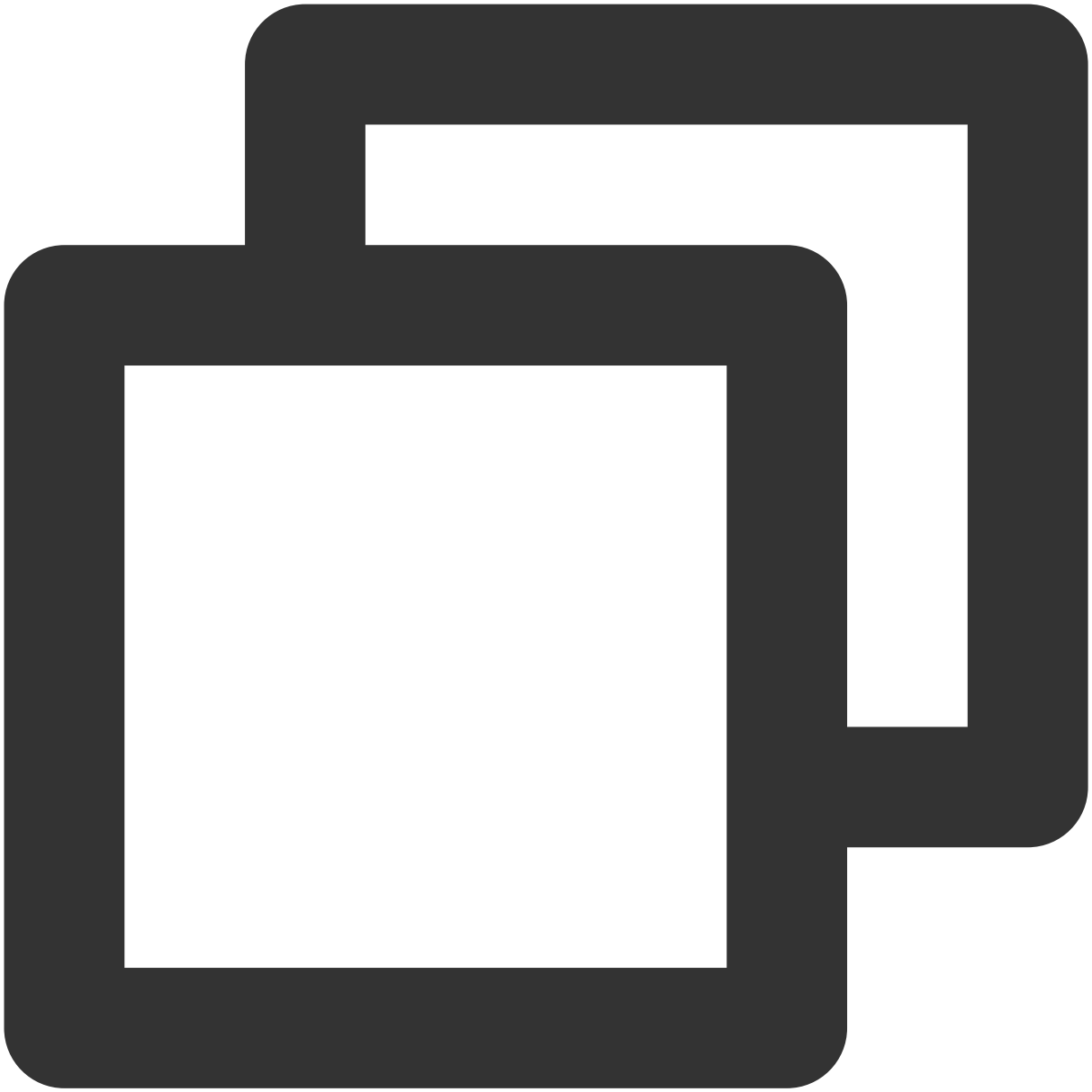
Request Header: Content-Type: application/json.

A request sample is as follows:

Request sample:

Response sample:

©2013-2022 Tencent Cloud. All rights reserved.

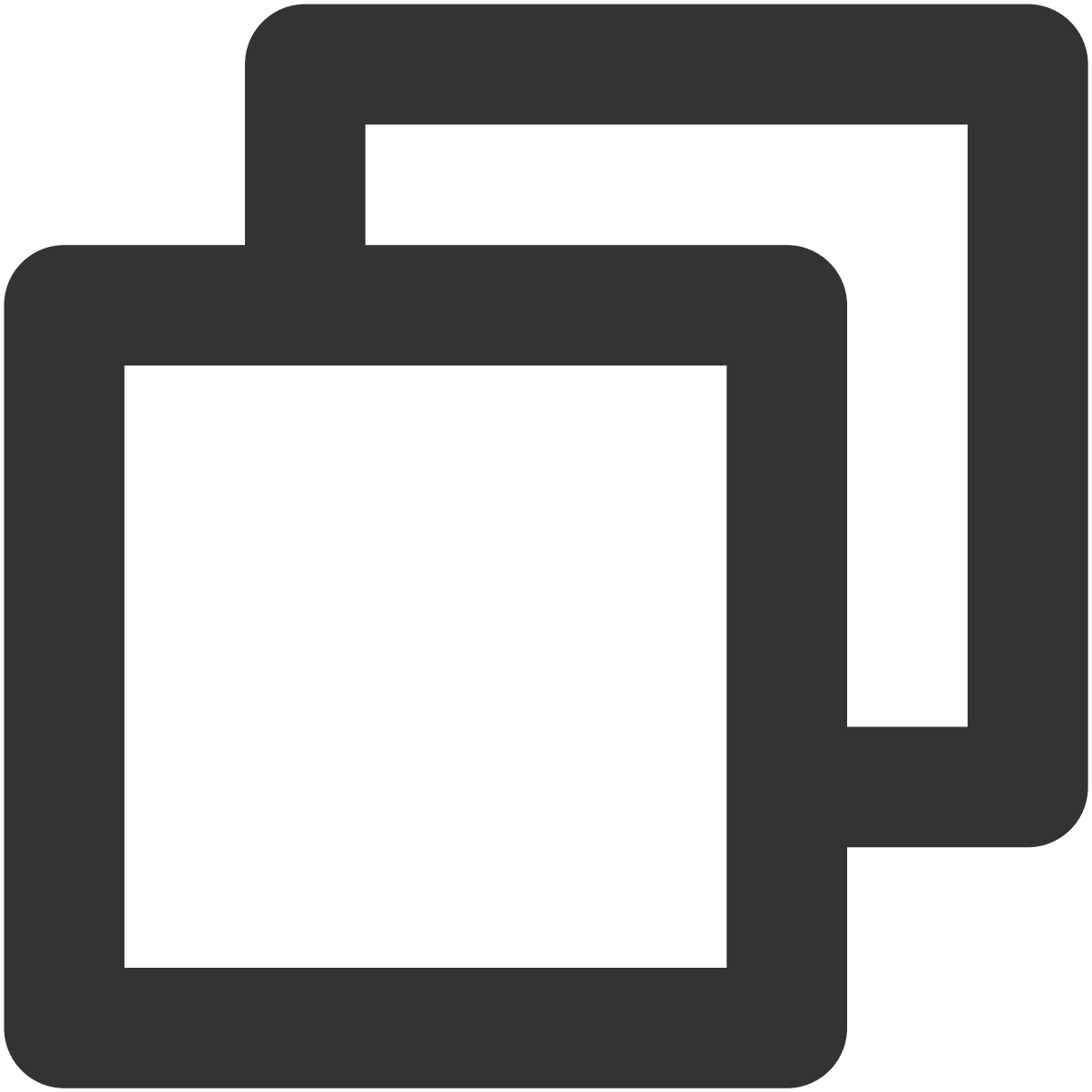


```
{
  "action": "CreateJob",
  "secretId": "{secretId}",
  "secretKey": "{secretKey}",
  "createJobRequest": {
    "customId": "{customId}",
    "callback": "{callback}",
    "inputs": [{ "url": "{url}" }],
    "outputs": [
      {
        "contentId": "{contentId}",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "destination": "{destination}",
        "inputSelectors": [0],
        "smartContentDescriptor": {
          "outputPrefix": "{outputPrefix}",
          "vocalScore": {
            "standardAudio": {
              "midi": {"url":"{url}"},
              "standardWav": {"url":"{url}"},
              "alignWav": {"url":"{url}"}
            }
          }
        }
      }
    ]
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.




```
{
    "requestId": "ac004192-110b-46e3-ade8-4e449df84d60",
    "createJobResponse": {
        "job": {
            "id": "13f342e4-6866-450e-b44e-3151431c578b",
            "state": 1,
            "customId": "{customId}",
            "callback": "{callback}",
            "inputs": [{ "url": "{url}" }],
            "outputs": [
                {
```


©2013-2022 Tencent Cloud. All rights reserved.

```
                    "contentId": "{contentId}",
                    "destination": "{destination}",
                    "inputSelectors": [0],
                    "smartContentDescriptor": {
                        "outputPrefix": "{outputPrefix}",
                        "vocalScore": {
                          "standardAudio": {
                            "midi": {"url":"{url}"},
                            "standardWav": {"url":"{url}"},
                            "alignWav": {"url":"{url}"}
                          }
                        }
                    }
                }
            ],
            "timing": {
                "createdAt": "1603432763000",
                "startedAt": "0",
                "completedAt": "0"
            }
        }
    }
}
```

3. Obtain music scoring results.

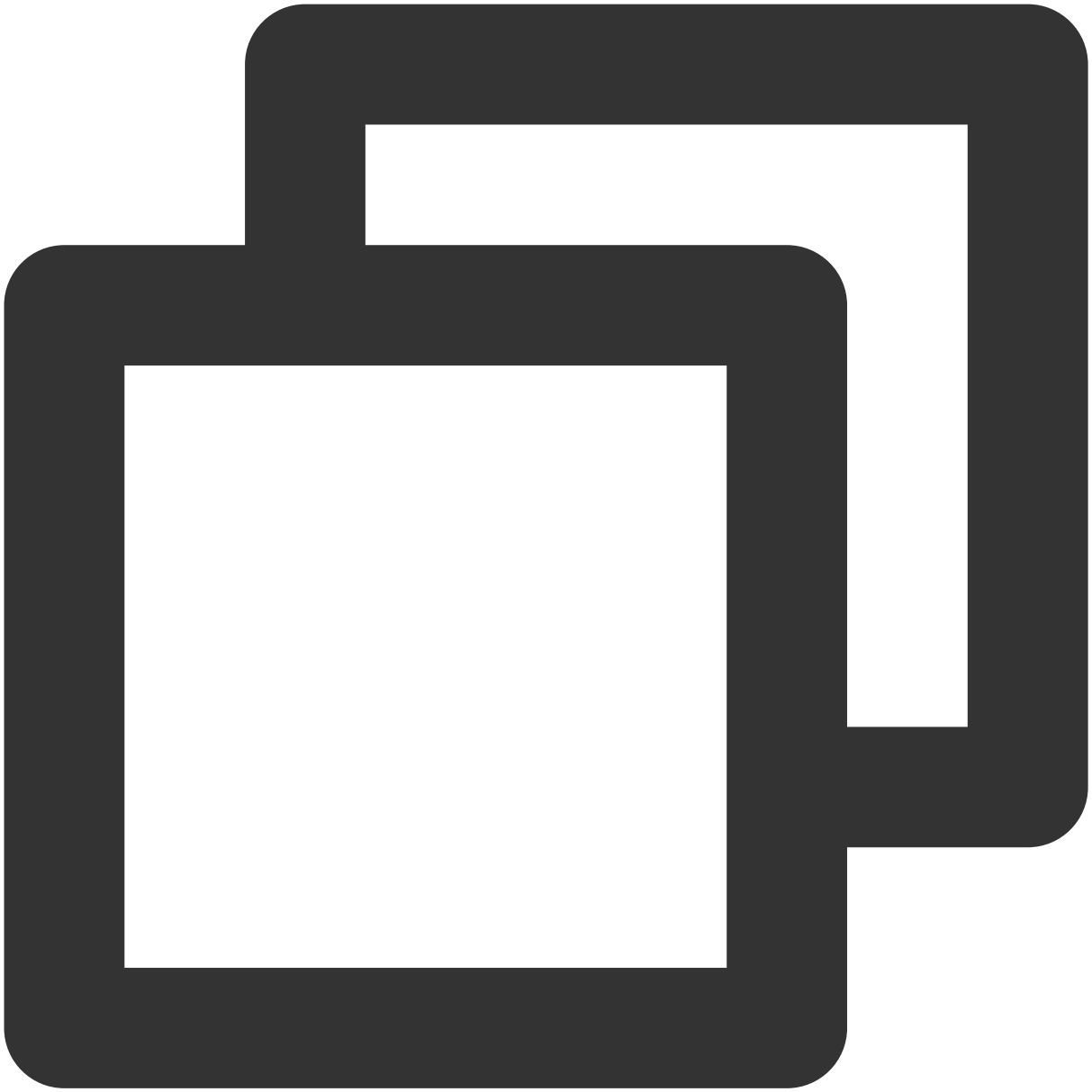
Obtain Method: Divided into active acquisition and passive callback.

By querying with the ID obtained from the response packet after creating the task, if the queried task is successful (state=3), the task's Output will carry the smartContentResult structure, in which the vocalScore field stores the result JSON file name. Users can construct the output file's COS path based on the information in Output's COS and destination.

Request sample:

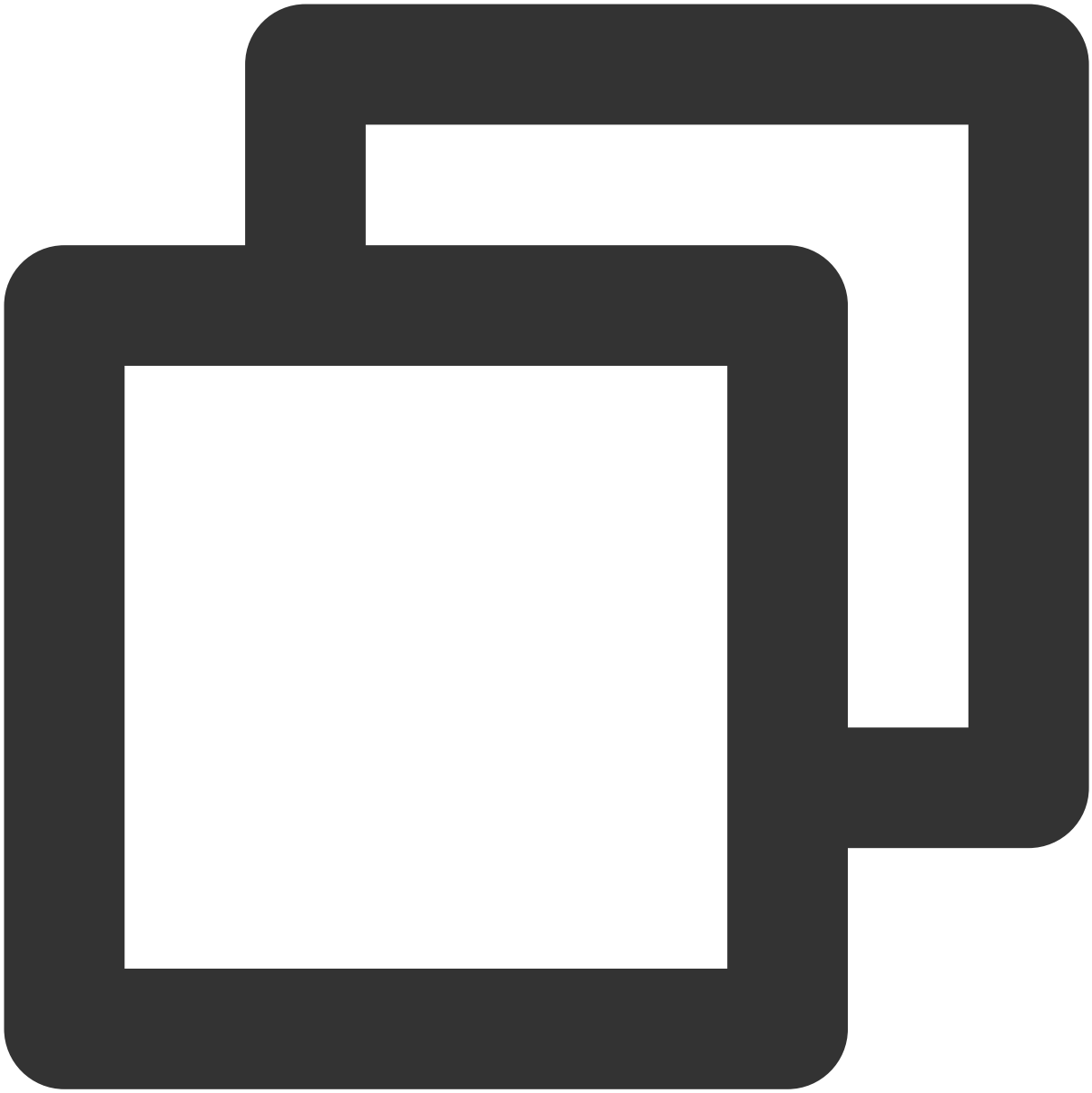
Response sample:

©2013-2022 Tencent Cloud. All rights reserved.



```
{
  "action": "GetJob",
  "secretId": "{secretId}",
  "secretKey": "{secretKey}",
  "getJobRequest": {
    "id": "{id}"
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.




```
{
  "requestId": "c9845a99-34e3-4b0f-80f5-f0a2a0ee8896",
  "getJobResponse": {
    "job": {
      "id": "a95e9d74-6602-4405-a3fc-6408a76bcc98",
      "state": 3,
      "customId": "{customId}",
      "callback": "{callback}",
      "timing": {
        "createdAt": "1610513575000",
        "startedAt": "1610513575000",
```


©2013-2022 Tencent Cloud. All rights reserved.

```
        "completedAt": "1610513618000"
      },
      "inputs": [{ "url": "{url}" }],
      "outputs": [
        {
          "contentId": "{contentId}",
          "destination": "{destination}",
          "inputSelectors": [0],
          "smartContentDescriptor": {
            "outputPrefix": "{outputPrefix}",
            "vocalScore": {
              "standardAudio": {
                "midi": {"url":"{url}"},
                "standardWav": {"url":"{url}"},
                "alignWav": {"url":"{url}"}
              }
            }
          },
          "smartContentResult": {
            "vocalScore": "out.json"
          }
        }
      ]
    }
  }
}
```

Passive callbacks need to fill in the callback field when creating a task. The platform will send the entire Job structure to the address specified by the callback after the task reaches the Completed state (COMPLETED/ERROR). It is recommended to obtain task results using passive callbacks. The entire Job structure of tasks that have reached the Completed state (COMPLETED/ERROR) will be sent to the address corresponding to the callback field specified when the task was created. See the active query sample for the Job structure (under getJobResponse).

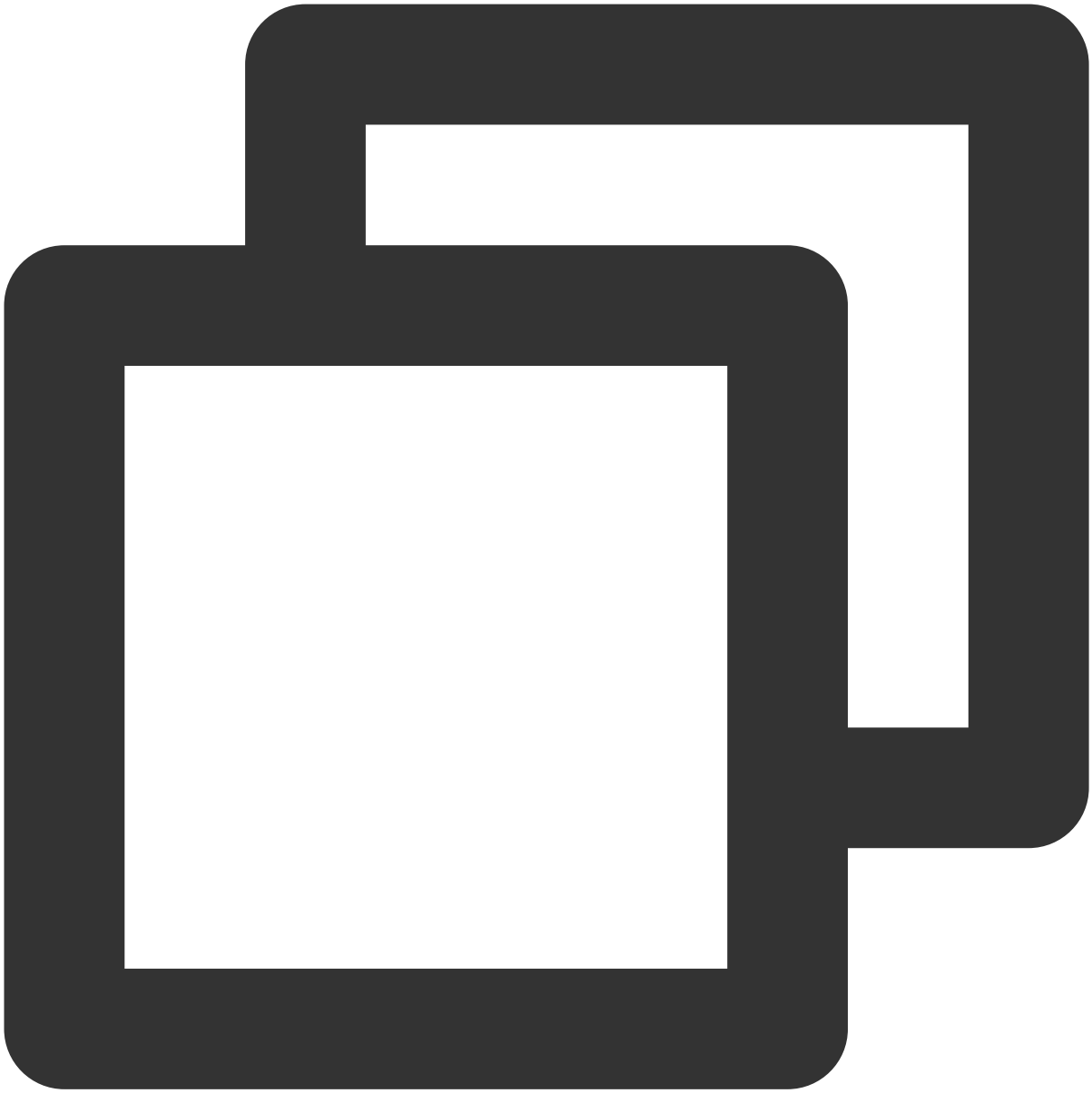
Note:

For more detailed intelligent music solution integration instructions for the music scoring module, see Music Scoring Integration.

Transparent transmission of single stream volume in mixed streams.

After the mixed streaming is enabled, the audience cannot directly obtain the on-mic anchor's single stream volume. In order to transparently transmit the single stream volume, the room owner may employ SEI to transmit the callback volume values of all on-mic anchors.

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onUserVoiceVolume(ArrayList<TRTCCloudDef.TRTCVolumeInfo> userVolumes, i
    super.onUserVoiceVolume(userVolumes, totalVolume);
    if (userVolumes != null && userVolumes.size() > 0) {
        // For storing volume values corresponding to on-mic users.
        HashMap<String, Integer> volumesMap = new HashMap<>();
        for (TRTCCloudDef.TRTCVolumeInfo user : userVolumes) {
            // Can set an appropriate volume threshold.
            if (user.volume > 10) {
                volumesMap.put(user.userId, user.volume);
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
        Gson gson = new Gson();
        String body = gson.toJson(volumesMap);
        // Transmit a collection of on-mic users' volume via SEI messages.
        mTRTCCloud.sendSEIMsg(body.getBytes(), 1);
    }
}

@Override
public void onRecvSEIMsg(String userId, byte[] data) {
    Gson gson = new Gson();
    HashMap<String, Integer> volumesMap = new HashMap<>();
    try {
        String message = new String(data, "UTF-8");
        volumesMap = gson.fromJson(message, volumesMap.getClass());
        for (String userId : volumesMap.keySet()) {
            // Print the volume levels of single streams of all on-mic users.
            Log.i(userId, String.valueOf(volumesMap.get(userId)));
        }
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
```

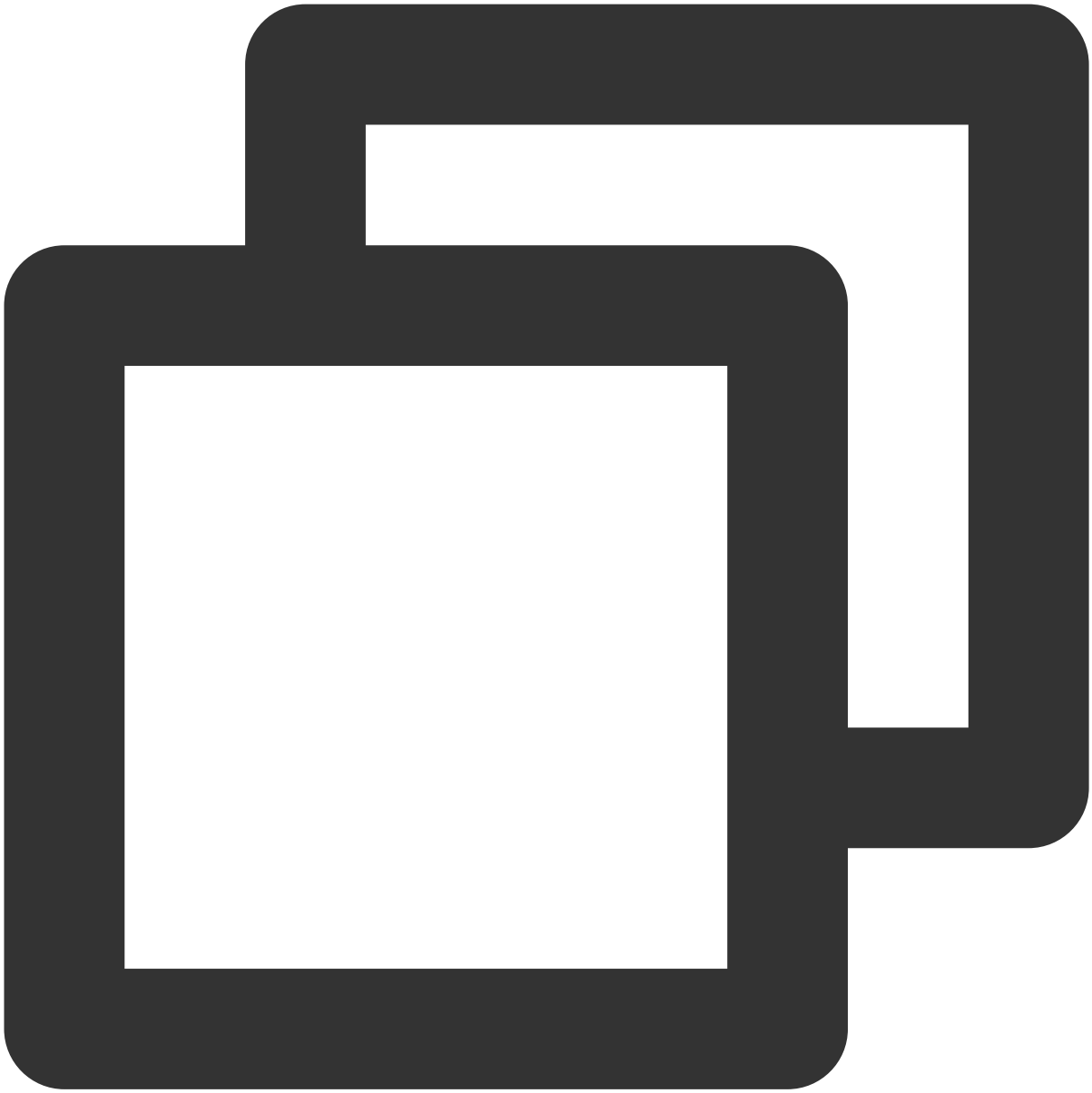
Note:

The prerequisite for using SEI messages to transparently transmit single stream volume through a mixed stream is that the room owner must either be video streaming or have black frame insertion enabled and furthermore, the audiences must actively subscribe to the room owner's video stream.

Real-time network quality callback

You can listen to `onNetworkQuality` to real-time monitor the network quality of both local and remote users. This callback is thrown every 2 seconds.

©2013-2022 Tencent Cloud. All rights reserved.



```
private class TRTCCloudImplListener extends TRTCCloudListener {
    @Override
    public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality,
                                 ArrayList<TRTCCloudDef.TRTCQuality> remoteQuality)
        // localQuality userId is empty. It represents the local user's network qua
        // remoteQuality represents the remote user's network quality evaluation re
        switch (localQuality.quality) {
            case TRTCCloudDef.TRTC_QUALITY_Excellent:
                Log.i(TAG, "The current network is excellent.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Good:
```

©2013-2022 Tencent Cloud. All rights reserved.

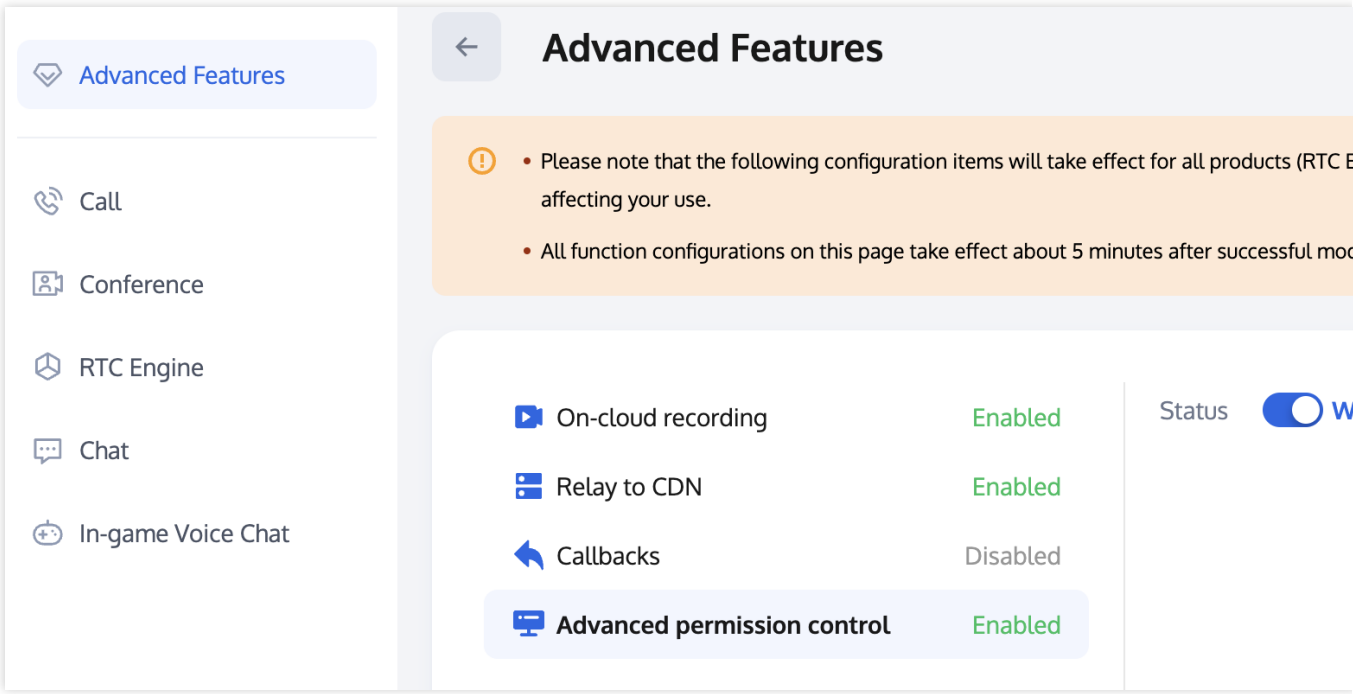
```
                Log.i(TAG, "The current network is good.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Poor:
                Log.i(TAG, "The current network is moderate.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Bad:
                Log.i(TAG, "The current network is poor.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Vbad:
                Log.i(TAG, "The current network is very poor.");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Down:
                Log.i(TAG, "The current network does not meet the minimum requireme
                break;
            default:
                Log.i(TAG, "Undefined.");
                break;
        }
    }
}
```

Advanced permission control

TRTC advanced permission control can be used to set different entry permissions for different rooms, such as advanced VIP rooms. It can also be used to control the permission for the audience to speak, such as handling ghost microphones.

Step 1: Enable the Advanced Permission Control Switch in the TRTC console application's advanced features page.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

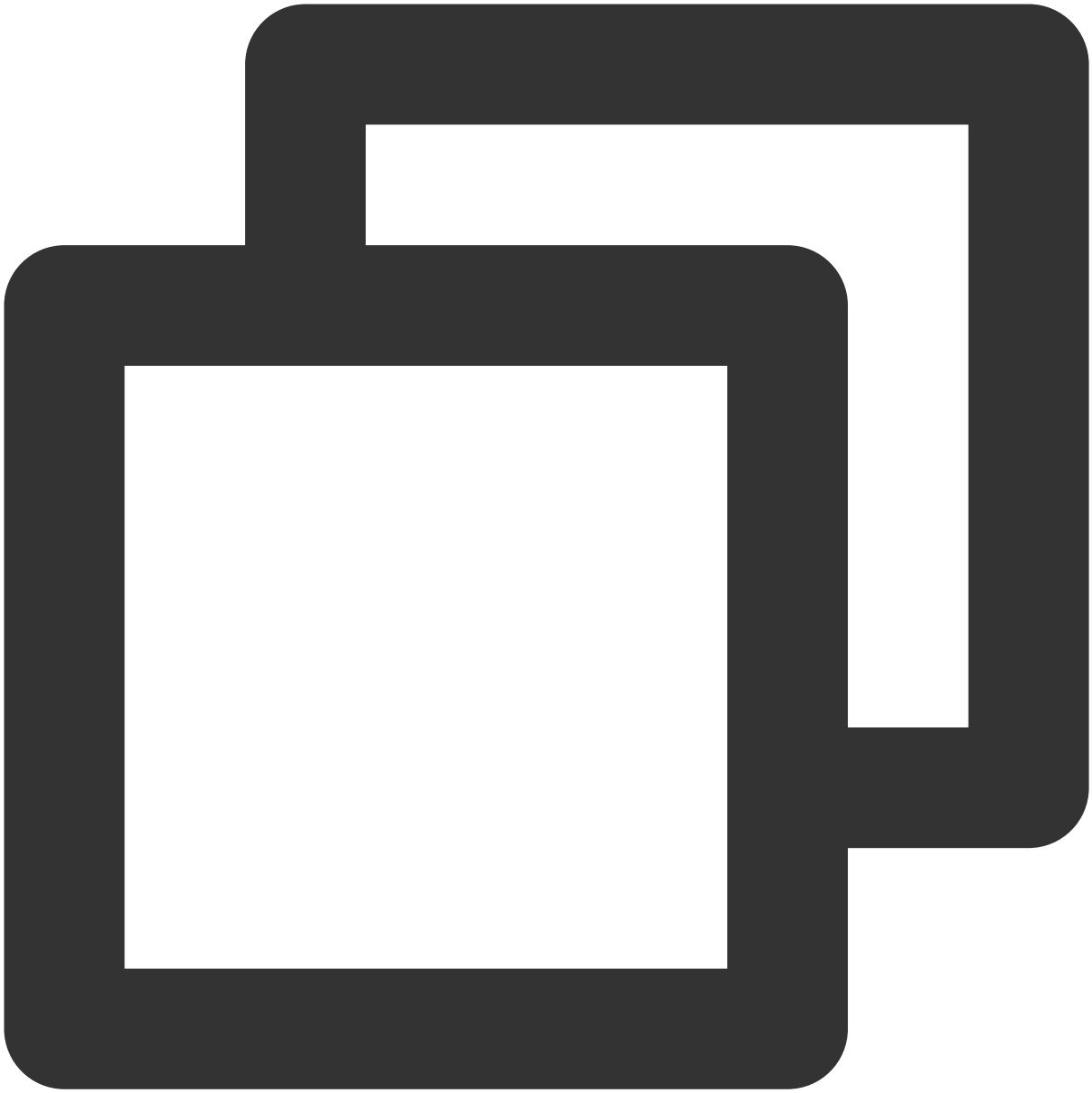
Once advanced permission control is enabled for a certain SDKAppID, all users using that SDKAppID need to pass in the `privateMapKey` parameter in `TRTCParams` to successfully enter the room. Therefore, if you have users online using this SDKAppID, do not enable this feature.

Step 2: Generate privateMapKey on the backend. For sample code, see privateMapKey computation source code.

Step 3: Room entry verification & speaking permission verification with PrivateMapKey.

Room entry verification

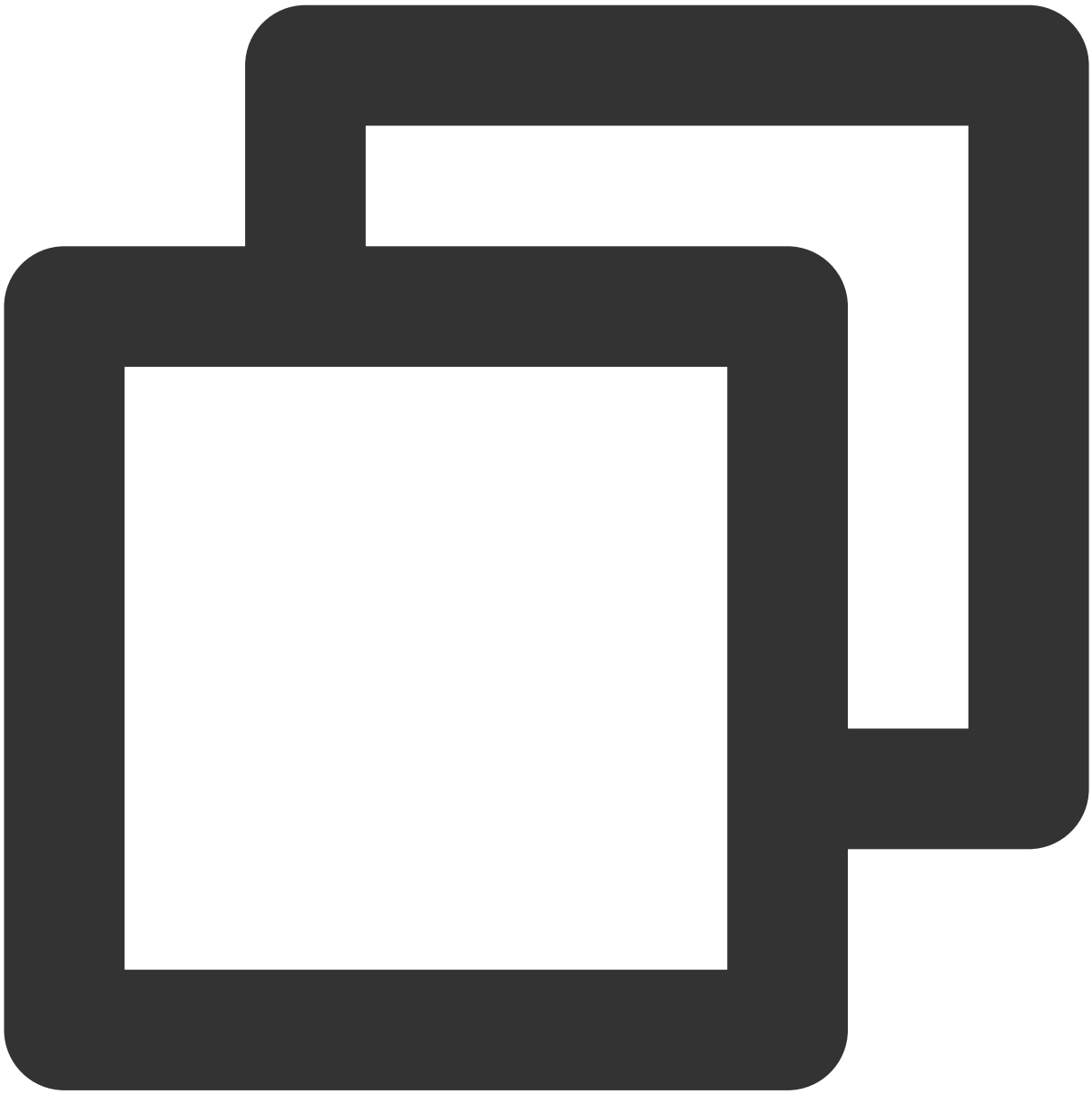
©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCParams mTRTCParams = new TRTCCloudDef.TRTCParams();
mTRTCParams.sdkAppId = SDKAPPID;
mTRTCParams.userId = mUserId;
mTRTCParams.strRoomId = mRoomId;
// UserSig obtained from the business backend.
mTRTCParams.userSig = getUserSig();
// PrivateMapKey obtained from the backend.
mTRTCParams.privateMapKey = getPrivateMapKey();
mTRTCParams.role = TRTCCloudDef.TRTCRoleAudience;
mTRTCCloud.enterRoom(mTRTCParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

©2013-2022 Tencent Cloud. All rights reserved.

Speaking permission verification



```
// Pass in the latest PrivateMapKey obtained from the backend into the role switchi
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor, getPrivateMapKey());
```

Exception Handling

Exception error handling

©2013-2022 Tencent Cloud. All rights reserved.

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description

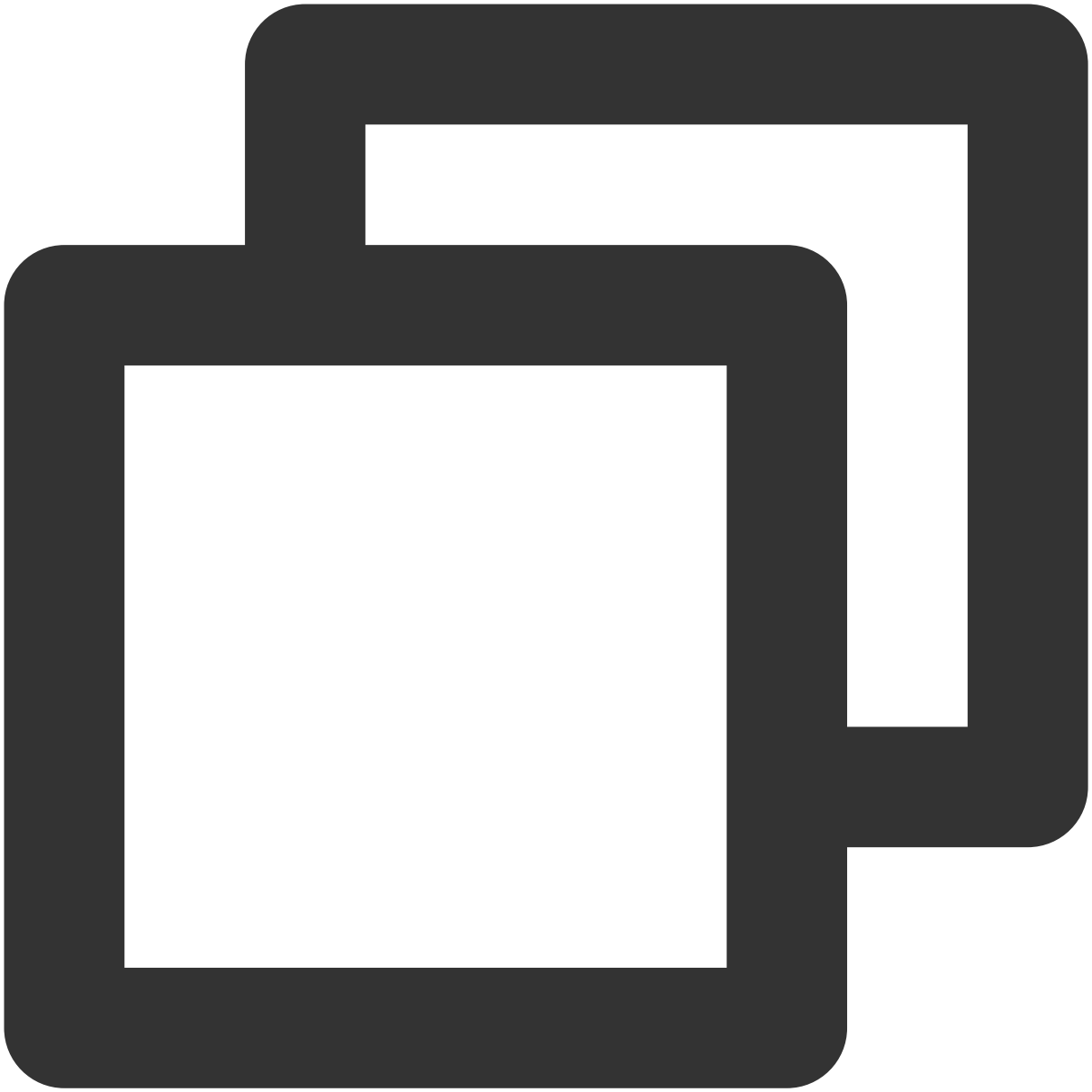
©2013-2022 Tencent Cloud. All rights reserved.

ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_SPEAKER_START_FAIL	-1321	Failed to open the speaker. For example, if there is an exception for the speaker's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Issues with IEMs

1. How to enable IEMs feature and set the volume?

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enable IEMs.
mTRTCCloud.getAudioEffectManager().enableVoiceEarMonitor(true);
// Set the volume of IEMs.
mTRTCCloud.getAudioEffectManager().setVoiceEarMonitorVolume(int volume);
```

Note:

The IEMs can be set in advance without having to monitor audio routing changes. Once headphones are connected, the IEMs feature will automatically take effect.

2. The IEMs feature does not take effect after enabled.

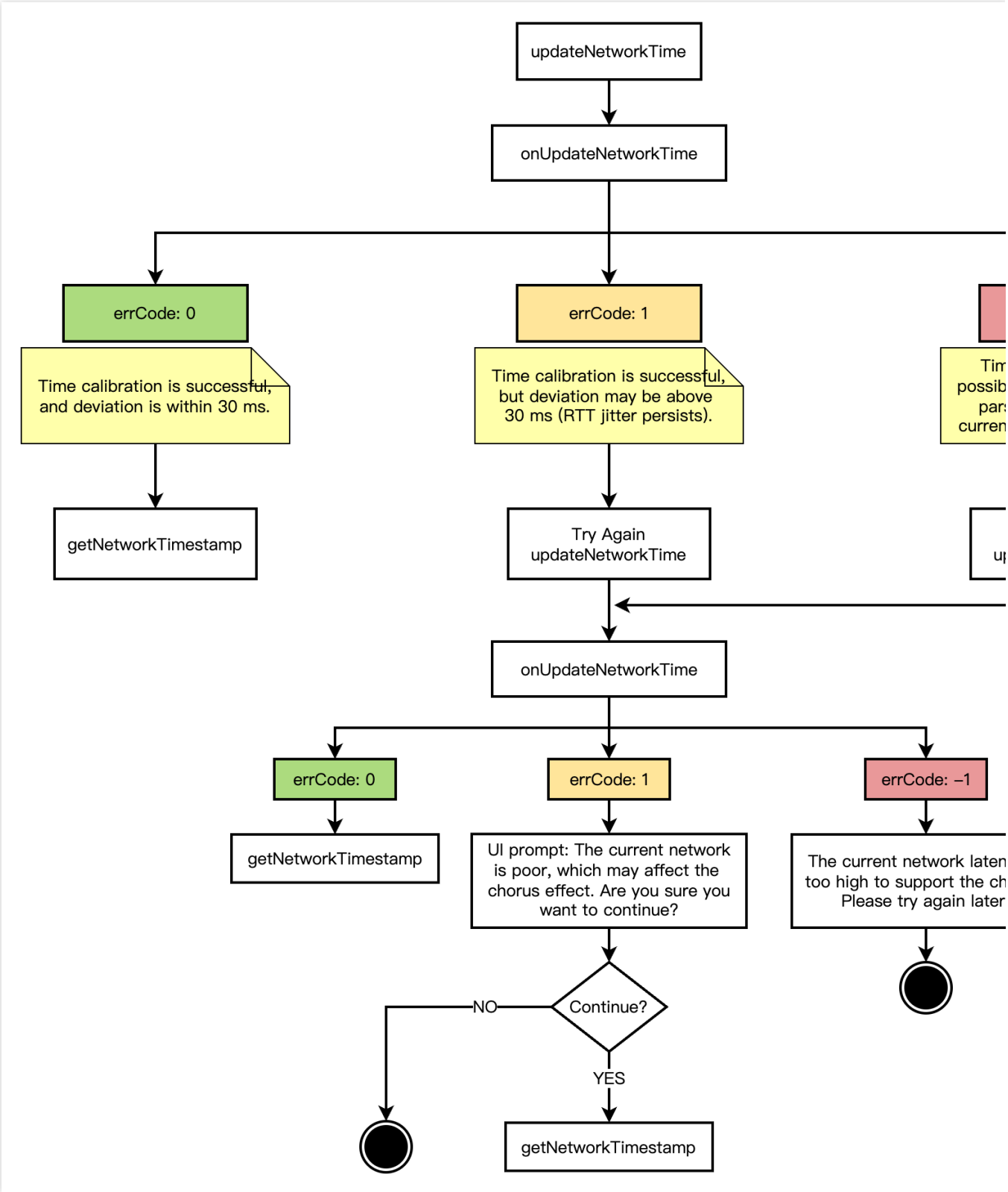
©2013-2022 Tencent Cloud. All rights reserved.

Due to the high hardware delay of Bluetooth headphones, it is recommended to prompt the anchor to wear wired headphones on the user interface. Also, it should be noted that not all smartphones will achieve excellent IEMs effect after this feature is enabled. TRTC SDK has already blocked this feature on some smartphones with poor effect.
3. High IEM delay
Check if Bluetooth headphones are in use. Due to the high hardware delay of Bluetooth headphones, wired headphones are recommended. Additionally, you can try improving the issue of high IEM delay by enabling hardware IEM through the experimental API `setSystemAudioKitEnabled`. Hardware IEMs have better performance and lower delay. Software IEMs have higher delay but better compatibility. Currently, for Huawei and VIVO devices, SDK defaults to hardware IEMs. Other devices default to software IEMs. If there are compatibility issues with hardware IEMs, contact us to configure forced use of software IEMs.

Issues with NTP sync

1. NTP time sync finished, but result maybe inaccurate
NTP sync is successful, but the deviation may still be more than 30 milliseconds. This indicates a poor client network environment with persistent RTT jitter.
2. Error in AddressResolver: No address associated with hostname
NTP sync has failed, possibly due to a temporary exception in local ISP DNS resolution under the current network environment. Try again later.
3. NTP service retry processing logic.

©2013-2022 Tencent Cloud. All rights reserved.



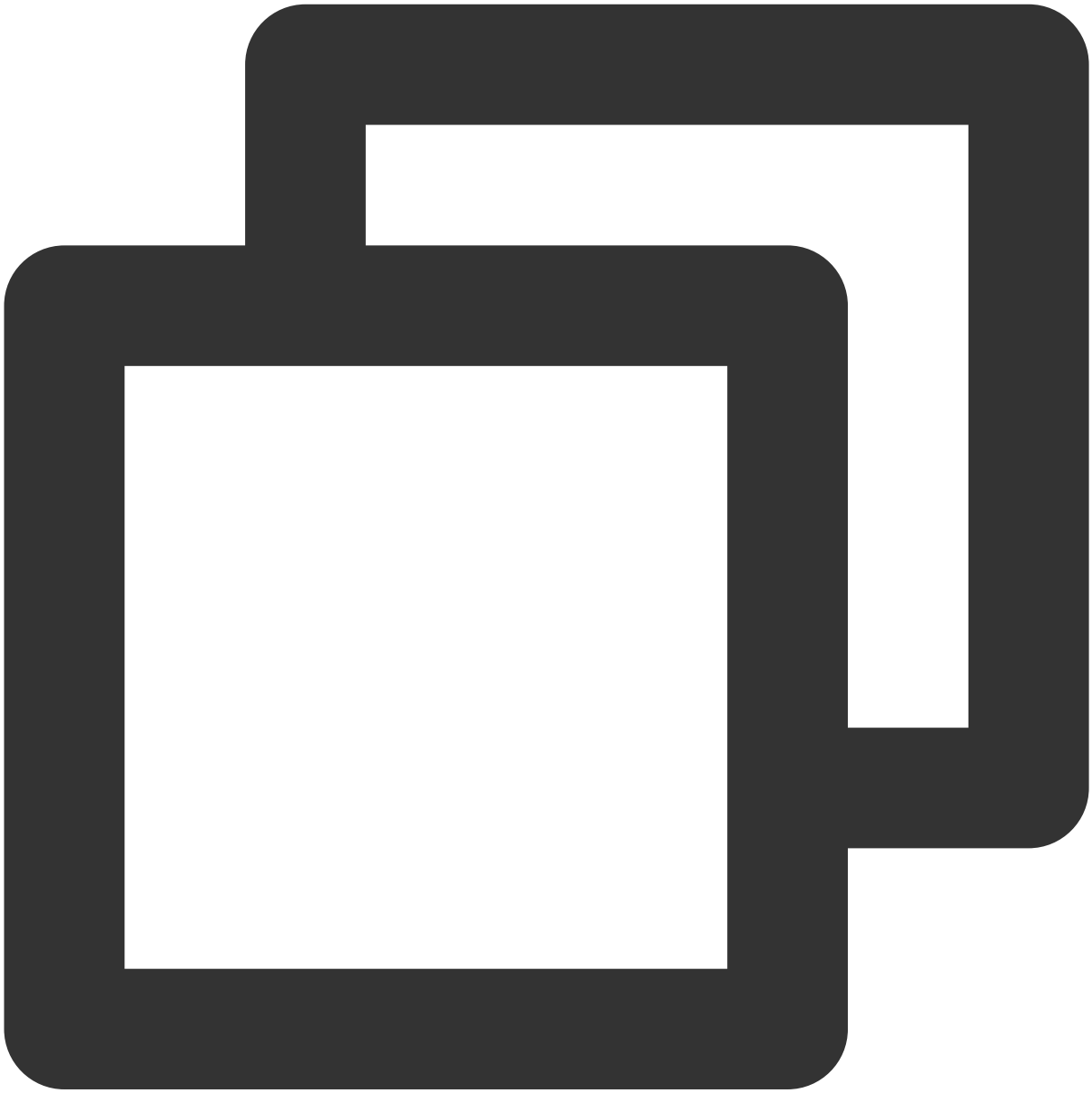
Issue with resource paths for playing music

In karaoke scenarios, when using TRTC SDK to play accompaniment music, you can choose to play from local or online music resources. Currently, playback paths only support URLs of online resources, absolute paths of music

©2013-2022 Tencent Cloud. All rights reserved.

files in device external storage, and application private directories. It does not support paths in directories like assets in Android development.

You can work around this issue by copying resource files from the assets directory to either the device external storage or the application private directory beforehand. Sample code is as follows:



```
public static void copyAssetsToFile(Context context, String name) {
    // The files directory under the application's directory.
    String savePath = ContextCompat.getExternalFilesDirs(context, null)[0].getAbsol
    // The cache directory under the application's directory.
    // String savePath = getApplication().getExternalCacheDir().getAbsolutePath();
    // The files directory under the application's private storage directory.
```

©2013-2022 Tencent Cloud. All rights reserved.

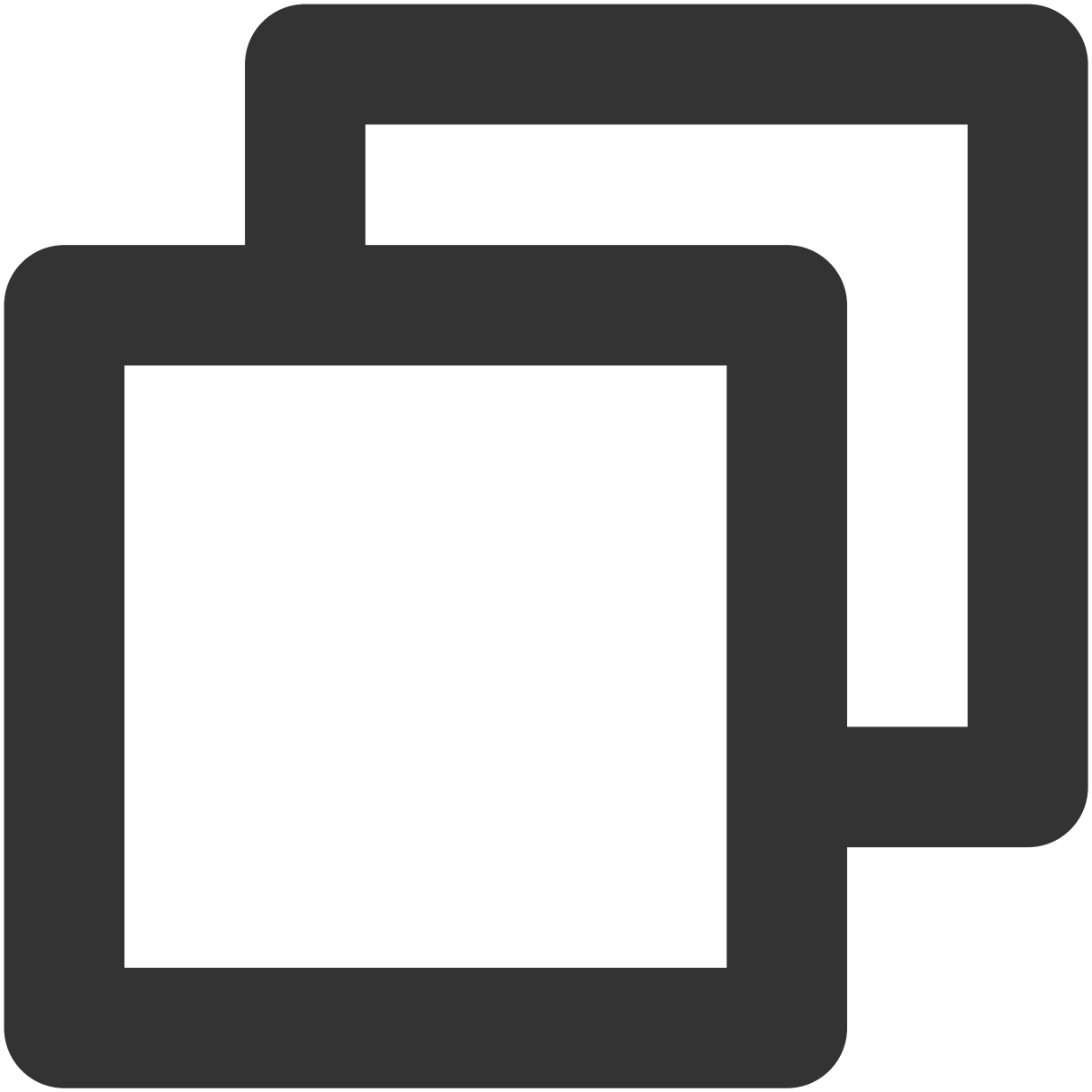
```
    // String savePath = getApplication().getFilesDir().getAbsolutePath();
    String filename = savePath + "/" + name;
    File dir = new File(savePath);
    // Create the directory if it does not exist.
    if (!dir.exists()) {
        dir.mkdir();
    }
    try {
        if (!(new File(filename)).exists()) {
            InputStream is = context.getResources().getAssets().open(name);
            FileOutputStream fos = new FileOutputStream(filename);
            byte[] buffer = new byte[1024];
            int count = 0;
            while ((count = is.read(buffer)) > 0) {
                fos.write(buffer, 0, count);
            }
            fos.close();
            is.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Application External Storage Files Directory Path:
/storage/emulated/0/Android/data/<package_name>/files/<file_name>.
Application External Storage Cache Directory Path:
/storage/emulated/0/Android/data/<package_name>/cache/file_name>.
Application Private Storage Files Directory Path: /data/user/0/<package_name>/files/<file_name>.

Note:

If you provide a path that is an external storage path outside of the application's specific directories, on Android 10 and above devices, you may face denial of access to the resource. This is due to Google introducing Partition Storage, a new storage management system. You can temporarily bypass this by adding the following code inside the <application> tag in the AndroidManifest.xml file: `android:requestLegacyExternalStorage="true"` . This attribute only takes effect on applications with targetSdkVersion 29 (Android 10), and applications with a higher version targetSdkVersion are still recommended to use the application's private or external storage paths.
For TRTC SDK v11.5 and later, playback of local music resources on Android devices via Content URI from Content Provider components is supported.
On Android 11 and HarmonyOS 3.0 or later, if you cannot access resource files in the external storage directory, you need to request the `MANAGE_EXTERNAL_STORAGE` permission:
First, you need to add the following entry in your application's AndroidManifest file.

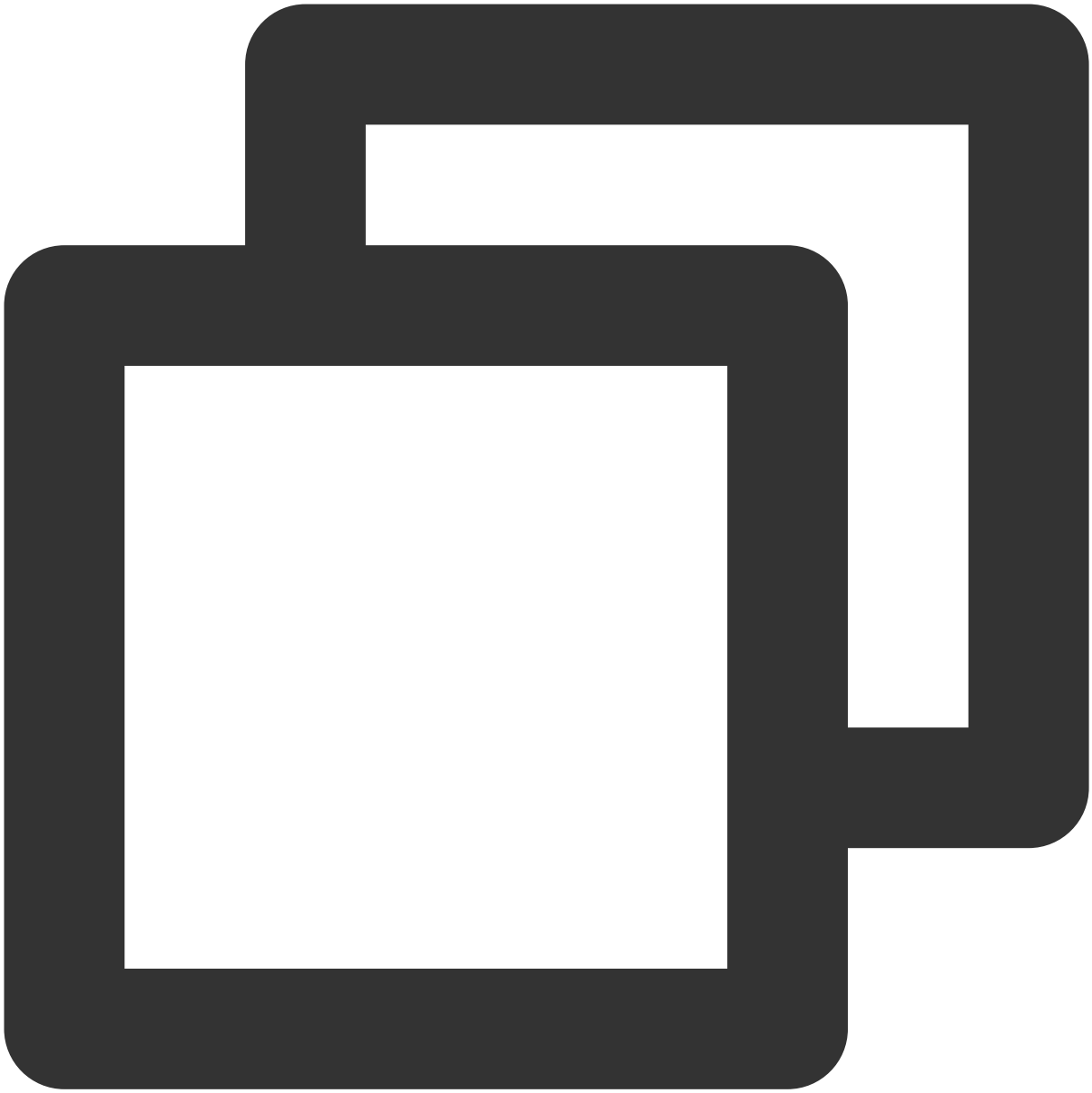
©2013-2022 Tencent Cloud. All rights reserved.



```
<manifest ...>
    <!-- This is the permission itself -->
    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />

    <application ...>
        ...
    </application>
</manifest>
```

Then, guide users to manually grant this permission at the point in your application where it is needed.

©2013-2022 Tencent Cloud. All rights reserved.



```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
    if (!Environment.isExternalStorageManager()) {
        Intent intent = new Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERM
        Uri uri = Uri.fromParts("package", getPackageName(), null);
        intent.setData(uri);
        startActivity(intent);
    }
} else {
    // For Android versions less than Android 11, you can use the old permissions m
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.WRITE_
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Issues with real-time chorus usage

1. **Why does the lead singer in real-time chorus scenarios need to use dual-instance streaming?**

In real-time chorus scenarios, to minimize end-to-end delay and achieve sync between vocals and accompaniment, a common approach is to use dual instances at the lead singer's end to separately upload vocal and accompaniment streams, while other chorus participants only upload their vocal streams and locally play the accompaniment. In this case, each chorus participant needs to subscribe to the lead singer's vocal stream, while refraining from subscribing to the lead singer's music stream. This setup can only be achieved by implementing dual-instance separate streaming.

2. **Why is it recommended to enable mixing pushback in real-time chorus scenarios?**

Having the audience pull multiple single streams at the same time is likely to result in misalignment between multiple vocal streams and accompaniment streams. Pulling a mixed stream can ensure absolute alignment of all streams and reduce downstream bandwidth.

3. **What are the uses of SEI in real-time chorus scenarios?**

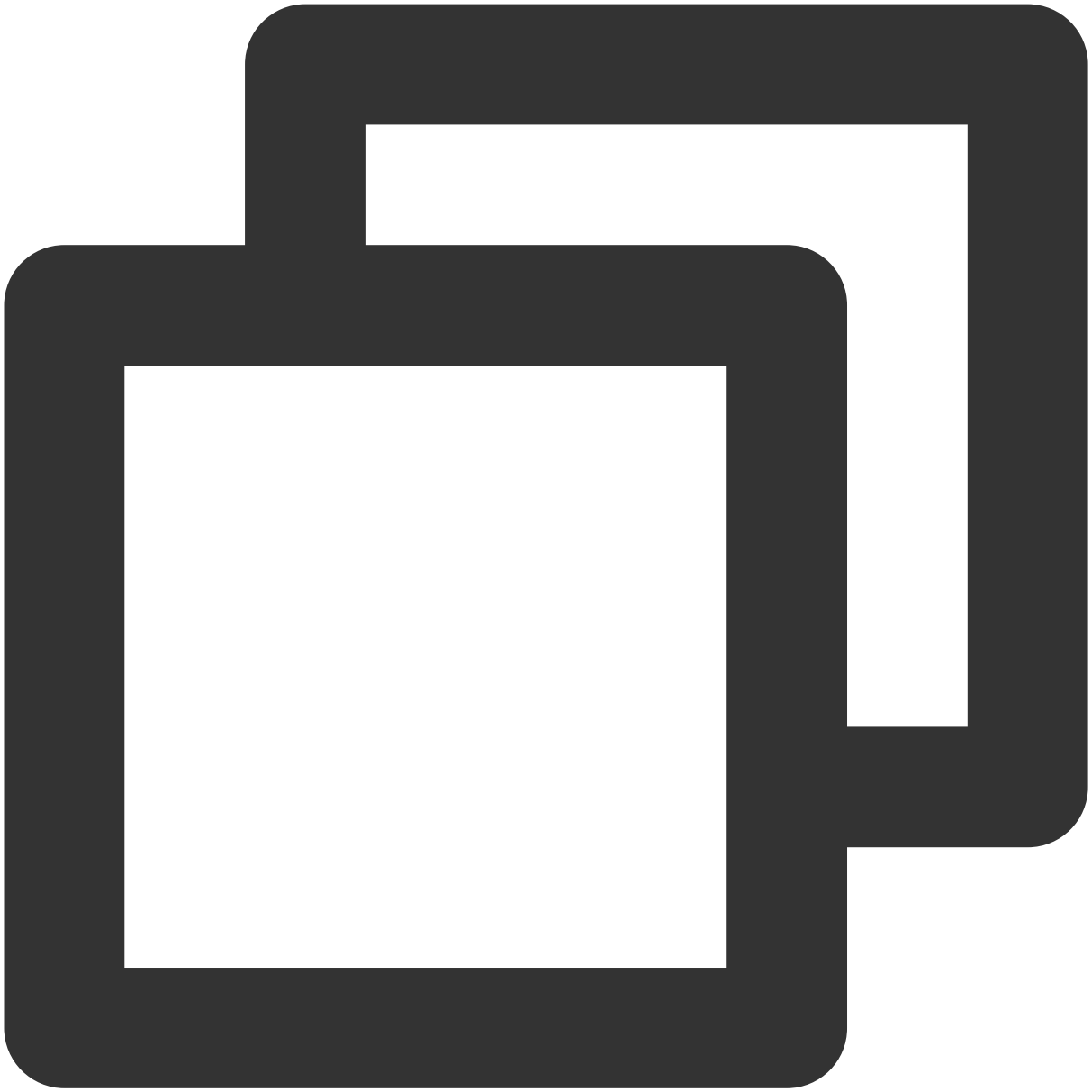
Transmitting accompaniment music progress, for lyric sync on the audience's end.

Transparently transmitting single stream volume through a mixed stream, for display as sound waves on the listener's end.

4. **Loading accompaniment music takes a long duration, causing significant playback delay?**

Loading network music resources via the SDK incurs a certain delay. It is recommended to initiate music pre-loading before starting playback.

©2013-2022 Tencent Cloud. All rights reserved.

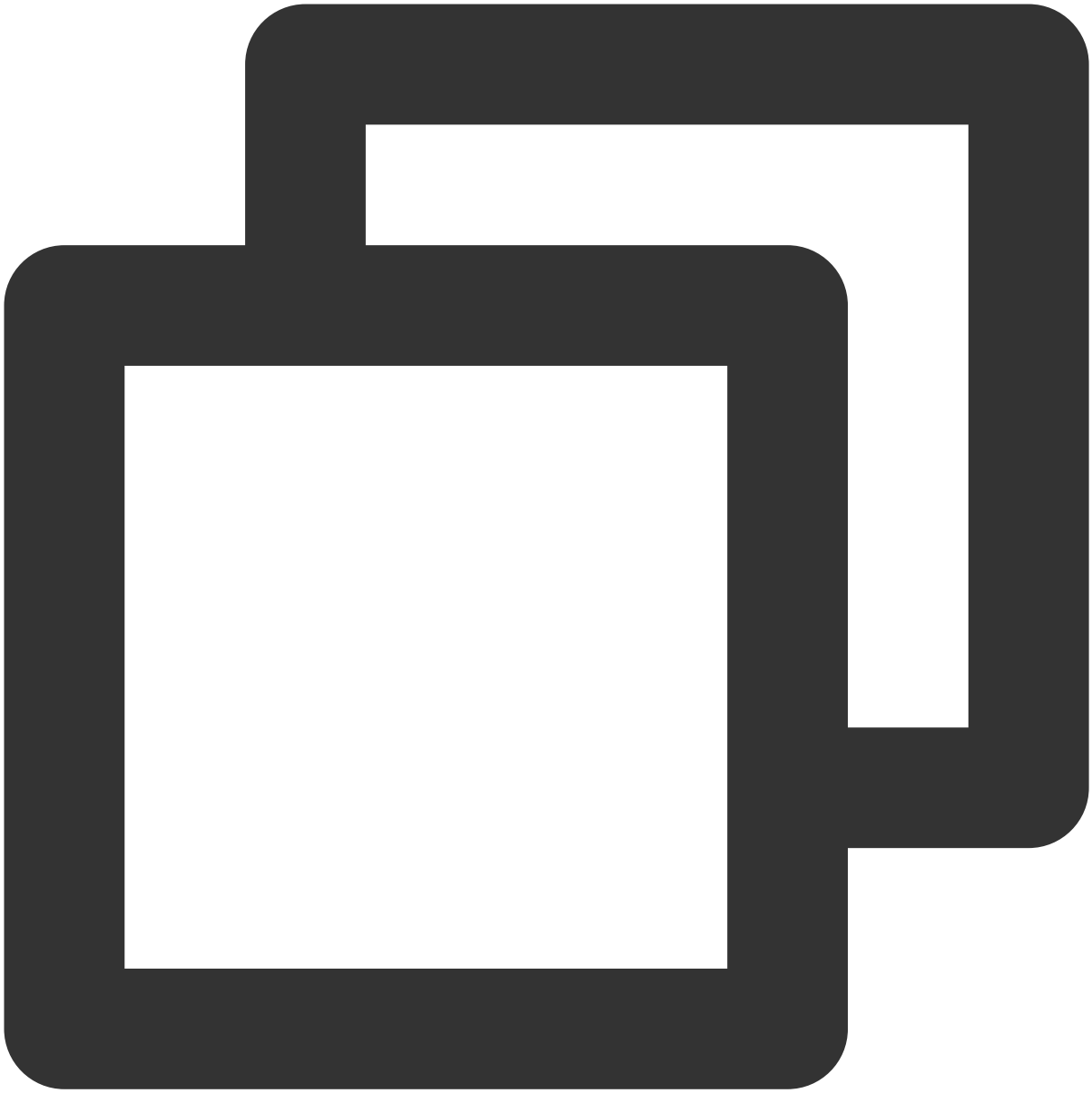


```
mTRTCCloud.getAudioEffectManager().preloadMusic(musicParam);
```

5. **When singing along with accompaniment, the vocals are barely audible. Is the music overwhelming the vocals?**

If the default volume settings result in the accompaniment overwhelming the vocals, it is recommended to adjust the volume balance between the music and vocals accordingly.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the local playback volume of a piece of background music.
mTRTCCloud.getAudioEffectManager().setMusicPlayoutVolume(musicID, volume);
// Set the remote playback volume of a specific background music.
mTRTCCloud.getAudioEffectManager().setMusicPublishVolume(musicID, volume);
// Set the local and remote volume of all background music.
mTRTCCloud.getAudioEffectManager().setAllMusicVolume(volume);
// Set the volume of voice capture.
mTRTCCloud.getAudioEffectManager().setVoiceCaptureVolume(volume);
```

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-18 14:26:14

Business Process

This section summarizes some common business processes in online karaoke, helping you better understand the implementation process of the entire scenario.

Song request process

Solo singing process

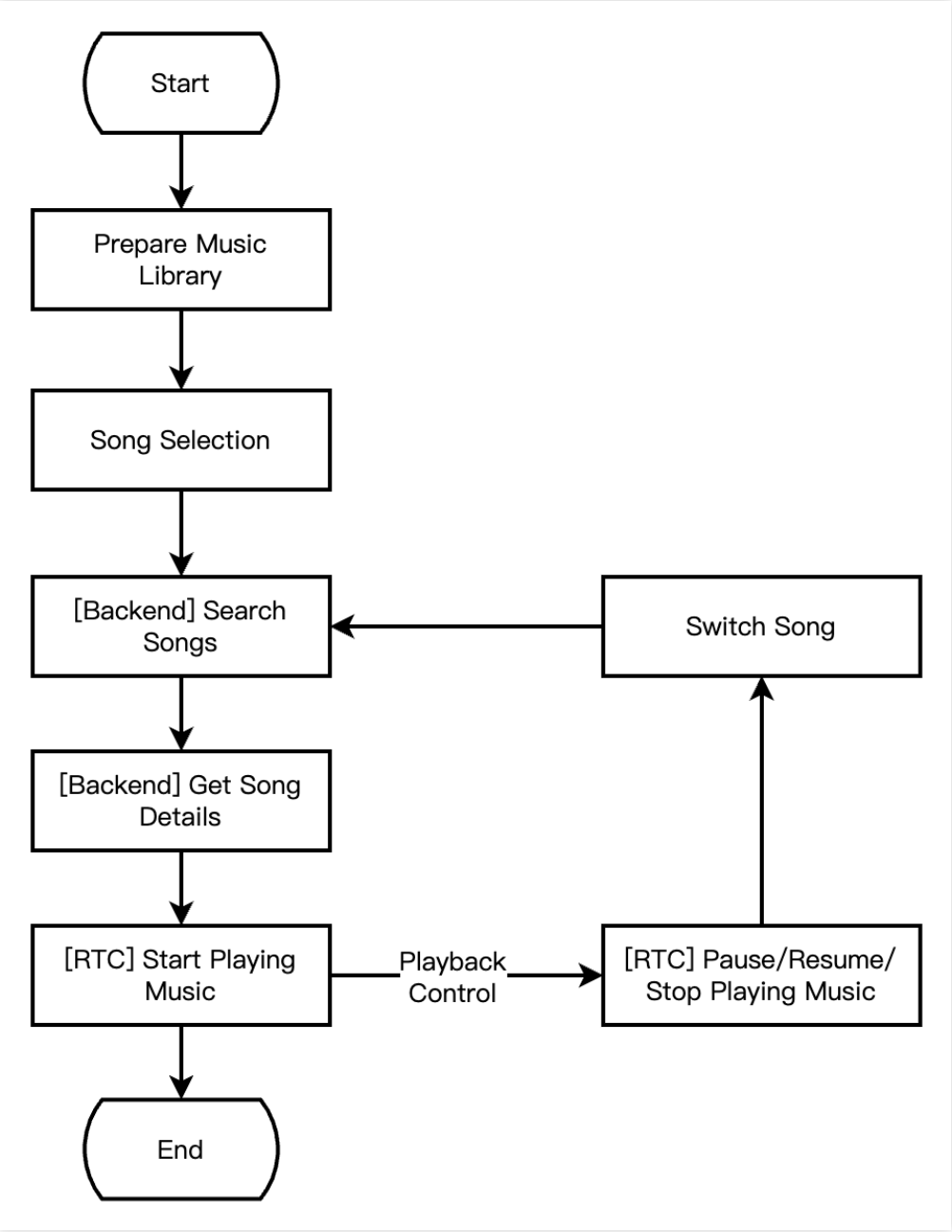
Lead singer process

Chorus process

Audience process

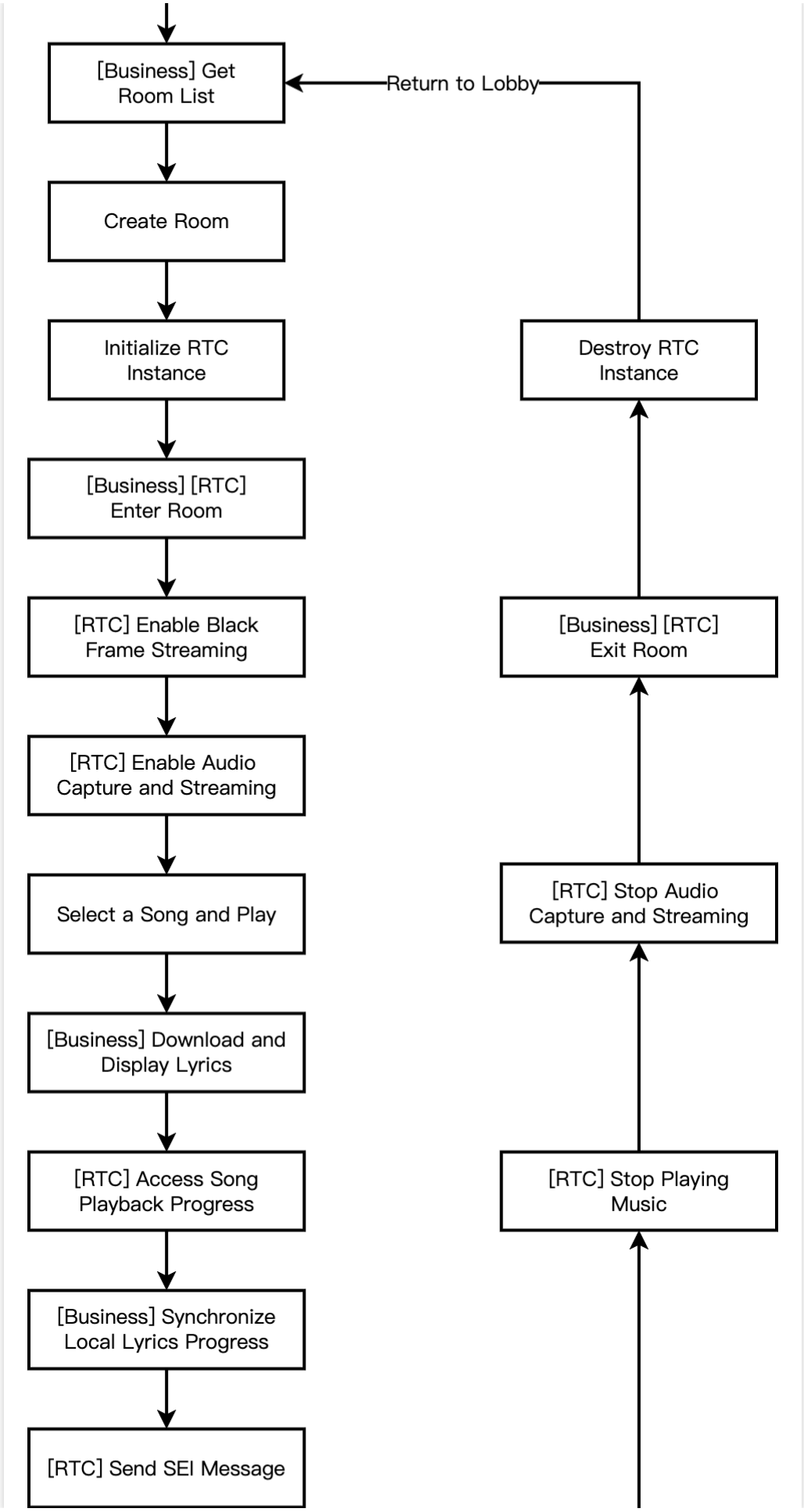
The following figure shows the process of requesting songs from a music repository on the business side and playing them using the TRTC SDK.

©2013-2022 Tencent Cloud. All rights reserved.

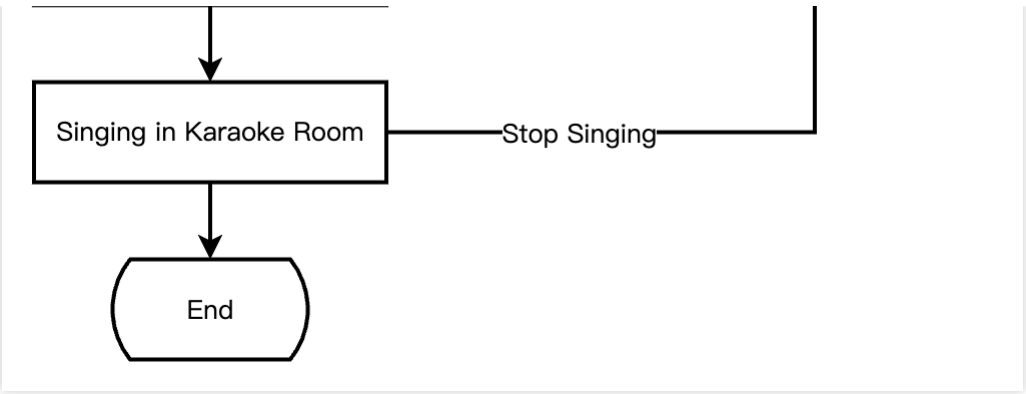


The following figure shows the process of a solo singing turn-taking game, that is, the performer enters a room to perform, stops performing, and exits the room.

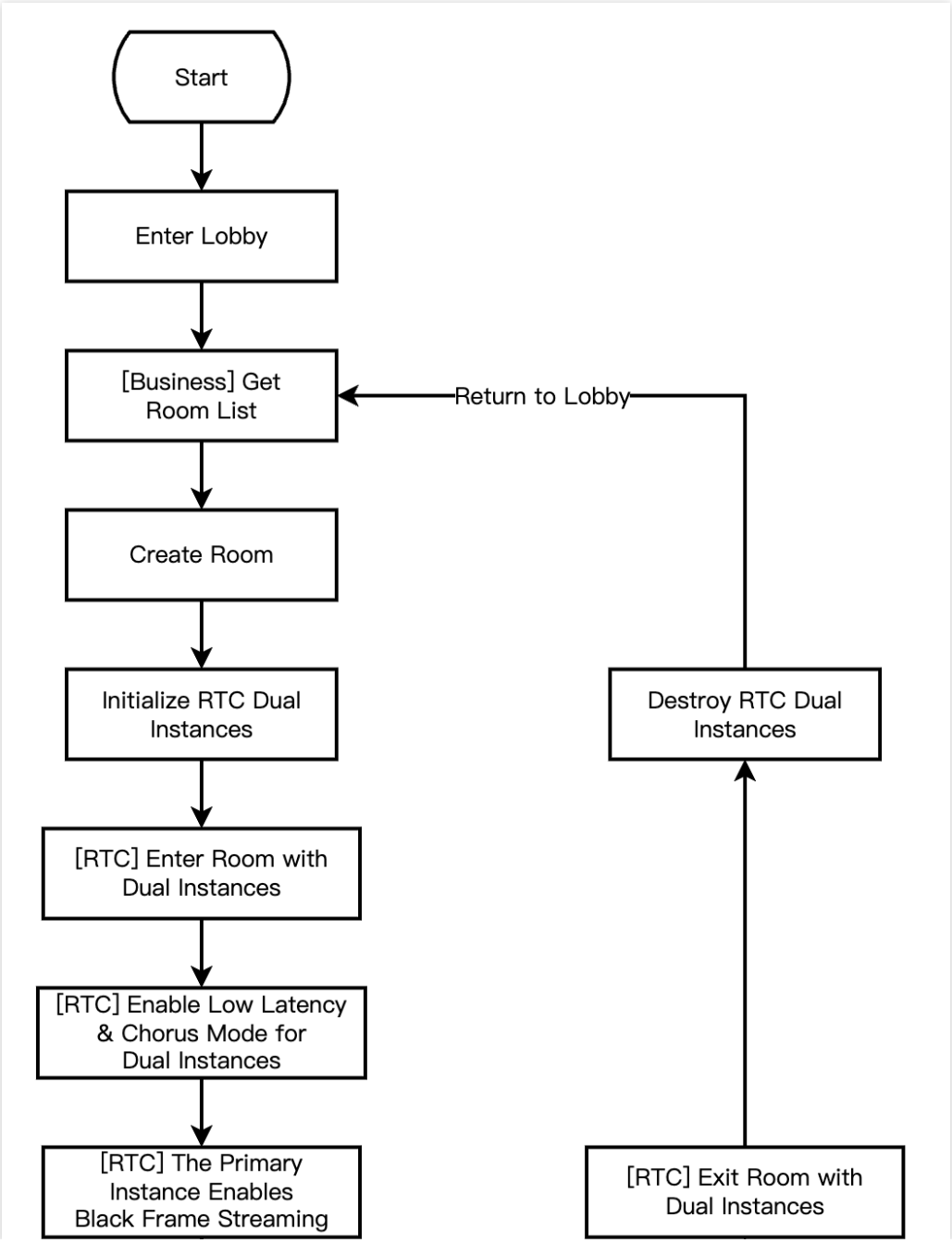
©2013-2022 Tencent Cloud. All rights reserved.



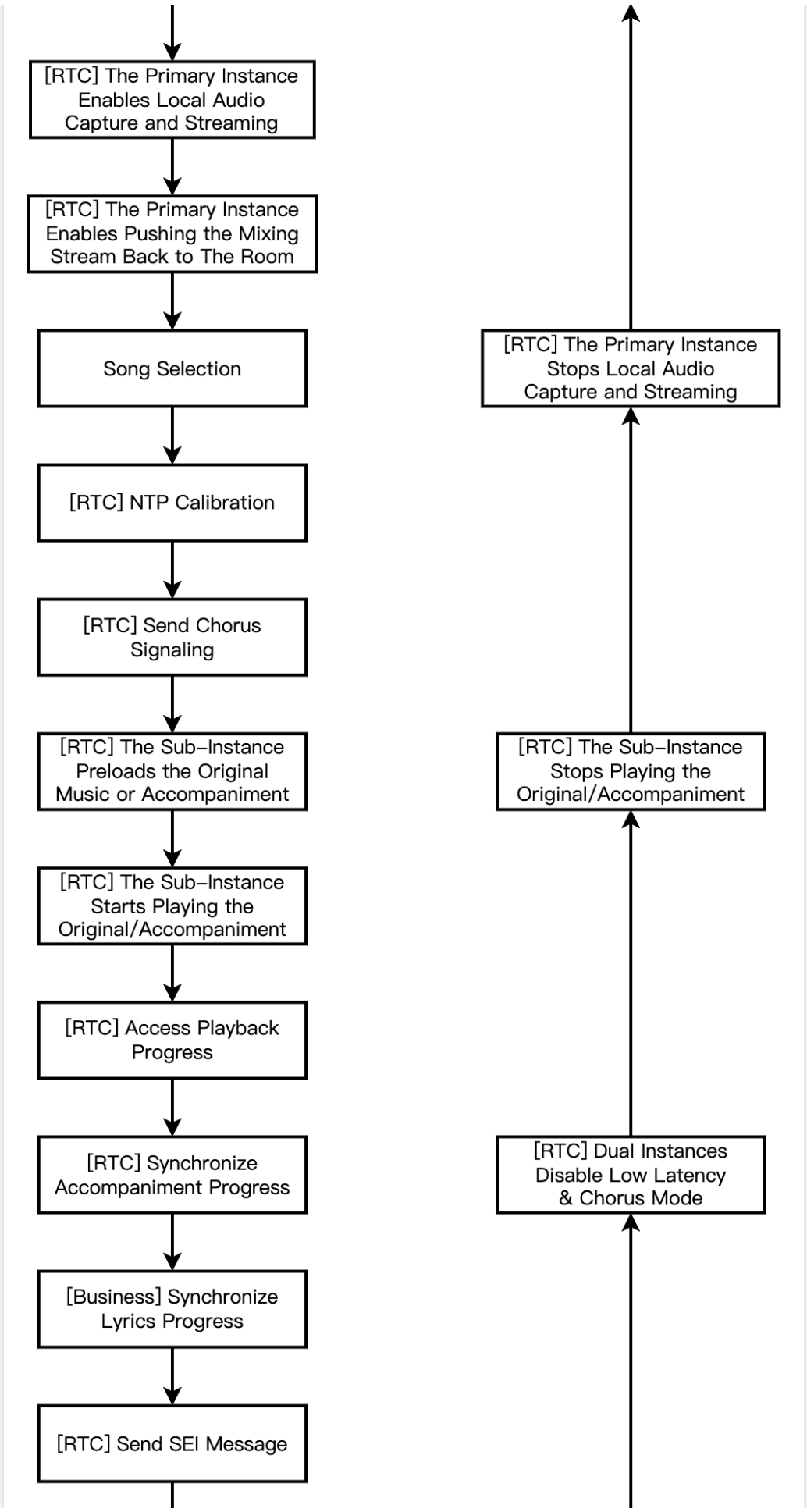
©2013-2022 Tencent Cloud. All rights reserved.



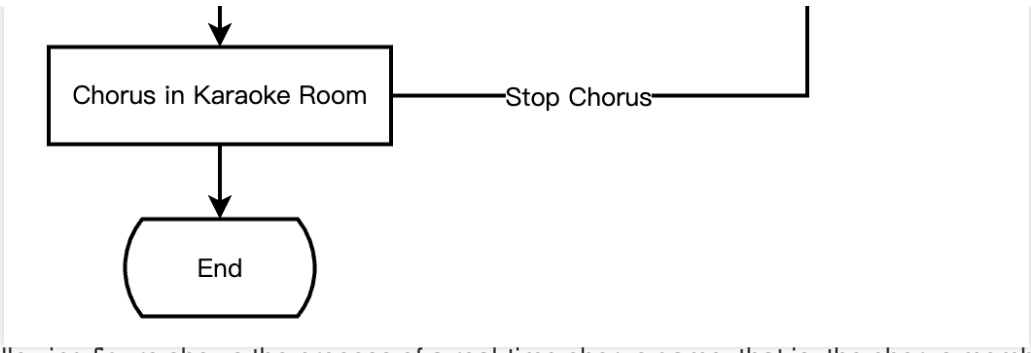
The following figure shows the process of a real-time chorus game, that is, the lead singer initiates a chorus, stops the chorus, and exits the room.



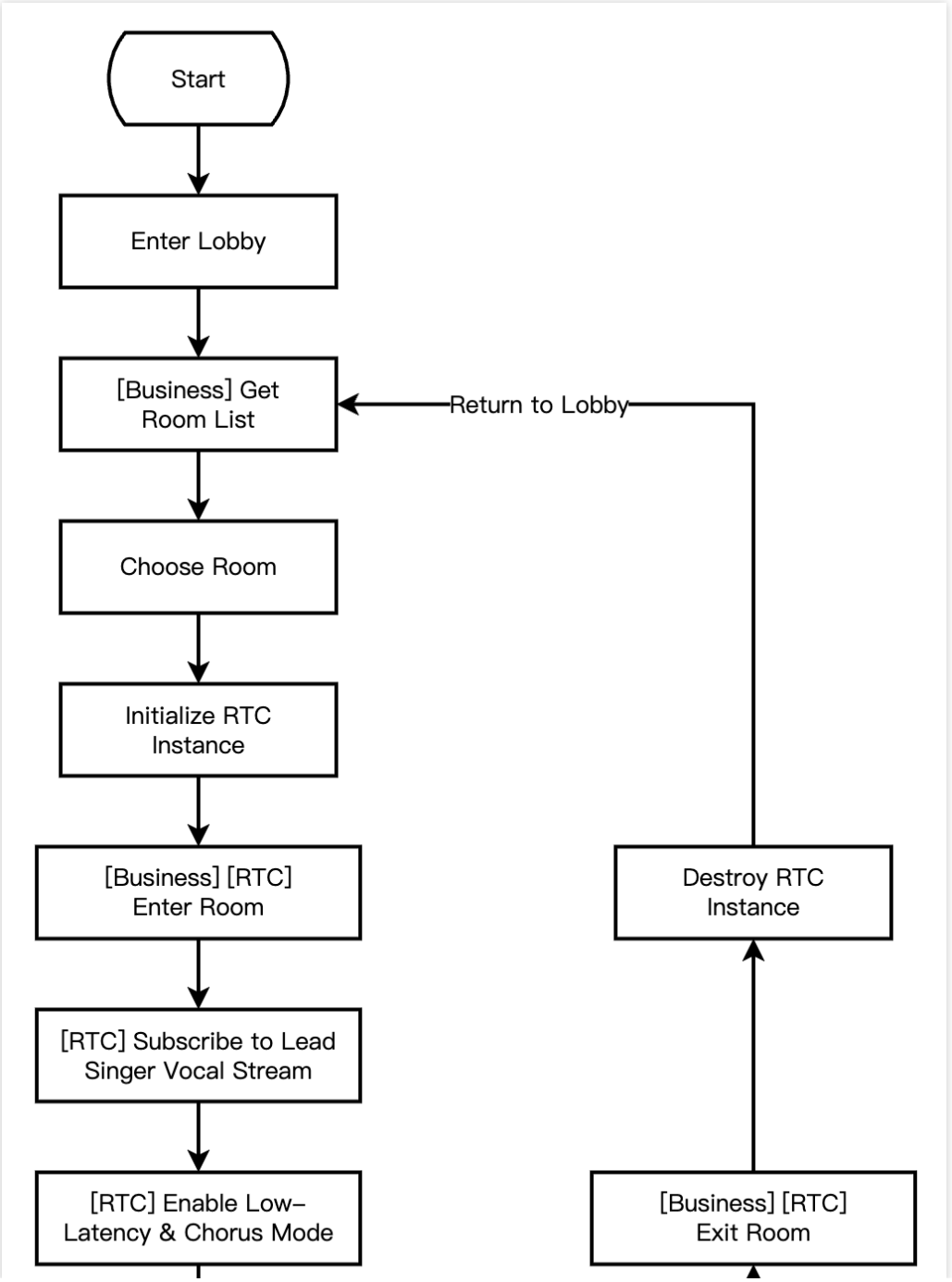
©2013-2022 Tencent Cloud. All rights reserved.



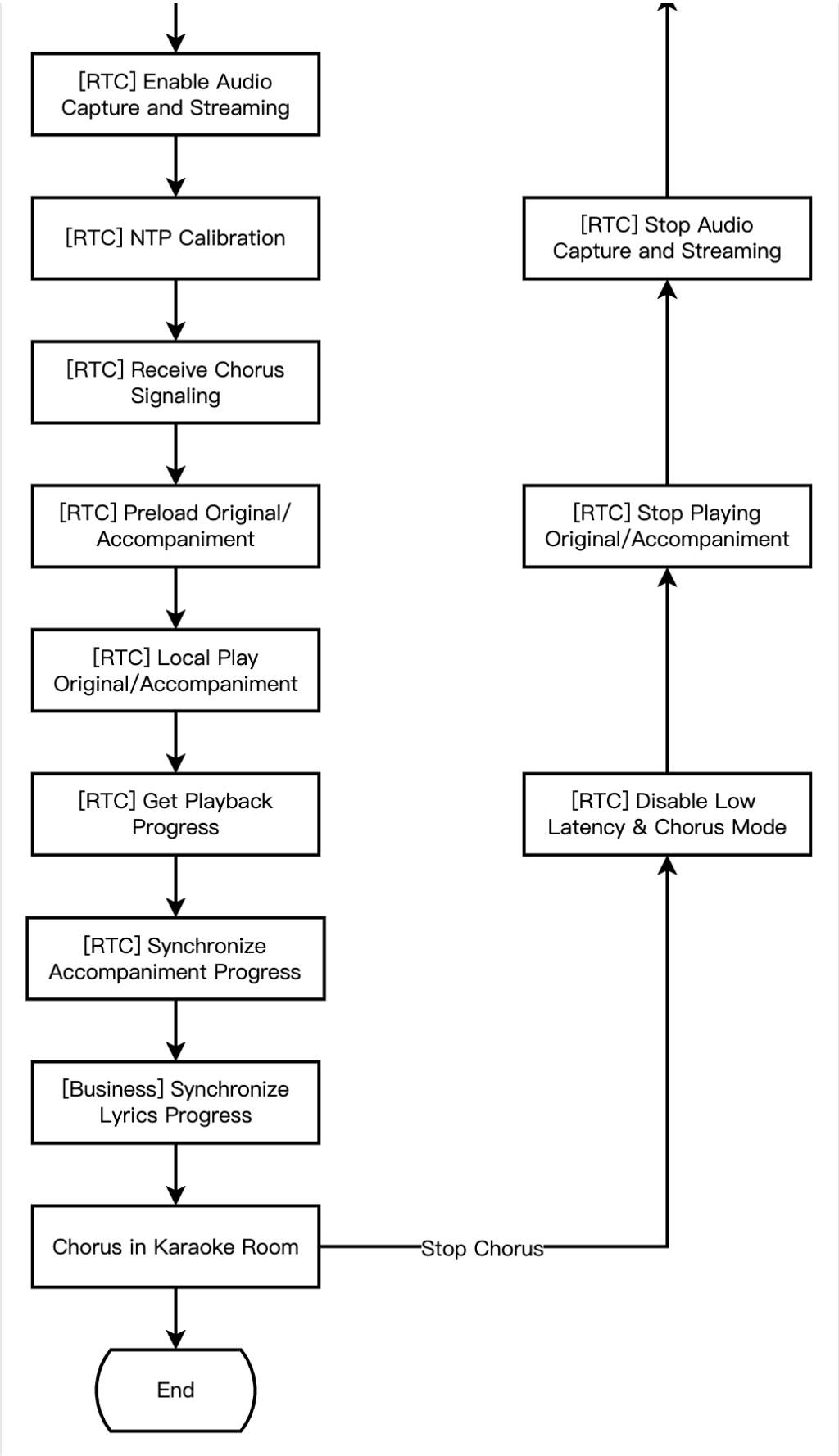
©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the process of a real-time chorus game, that is, the chorus members join the chorus, stop the chorus, and exit the room.

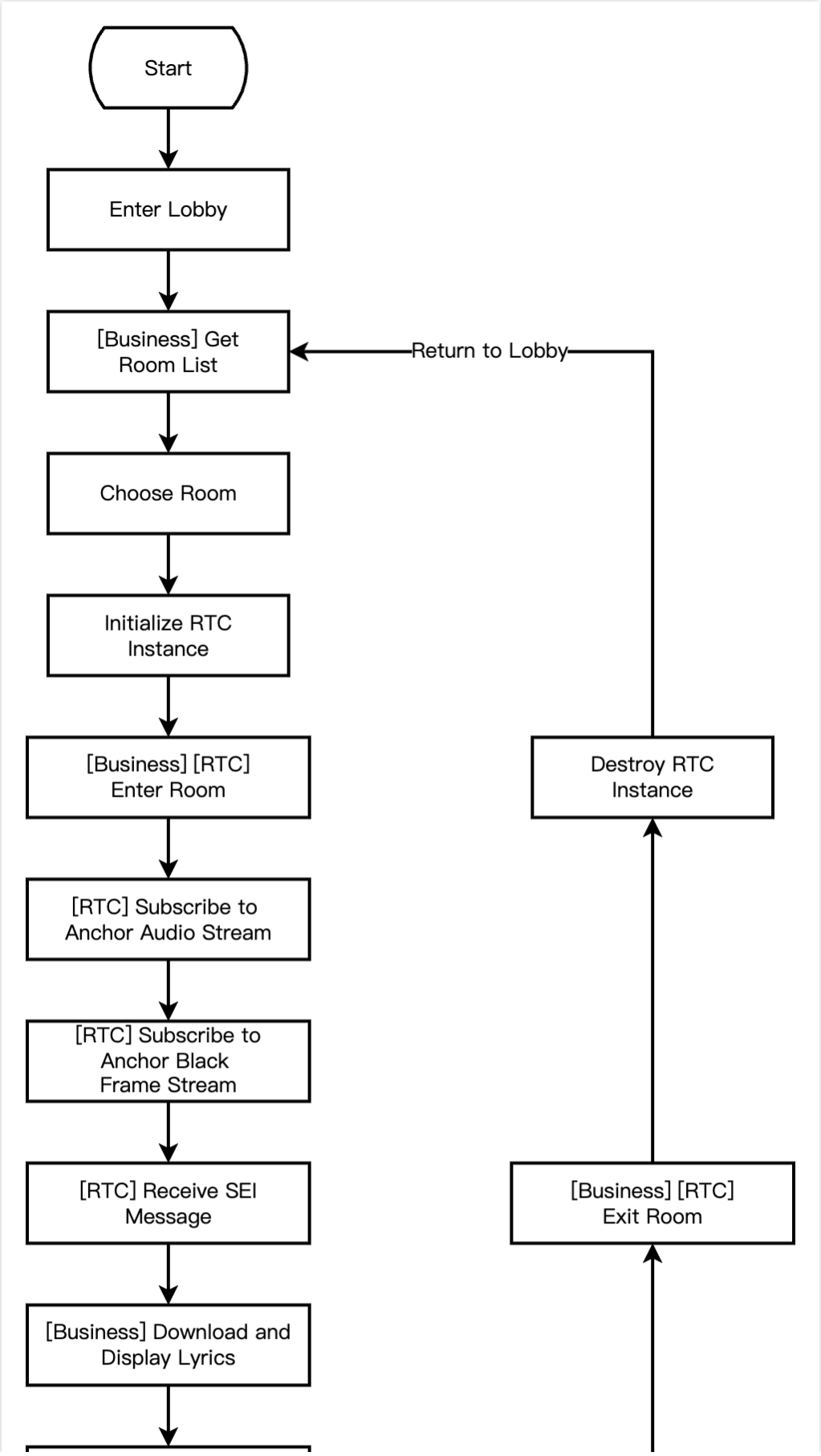


©2013-2022 Tencent Cloud. All rights reserved.

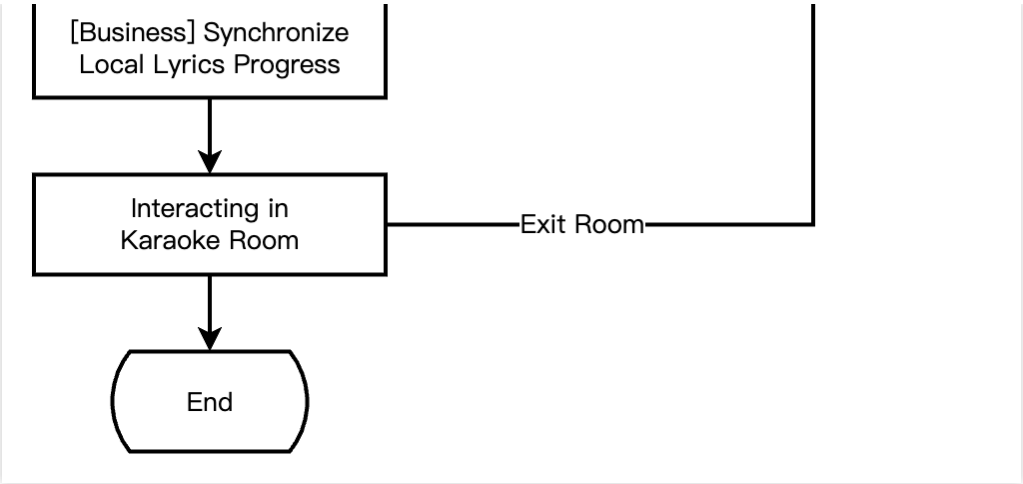


The following figure shows the process of an online karaoke scenario, that is, the audience enters the room to listen to songs and synchronizes lyrics.

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



Integration Preparation

Step 1. Activating the service.

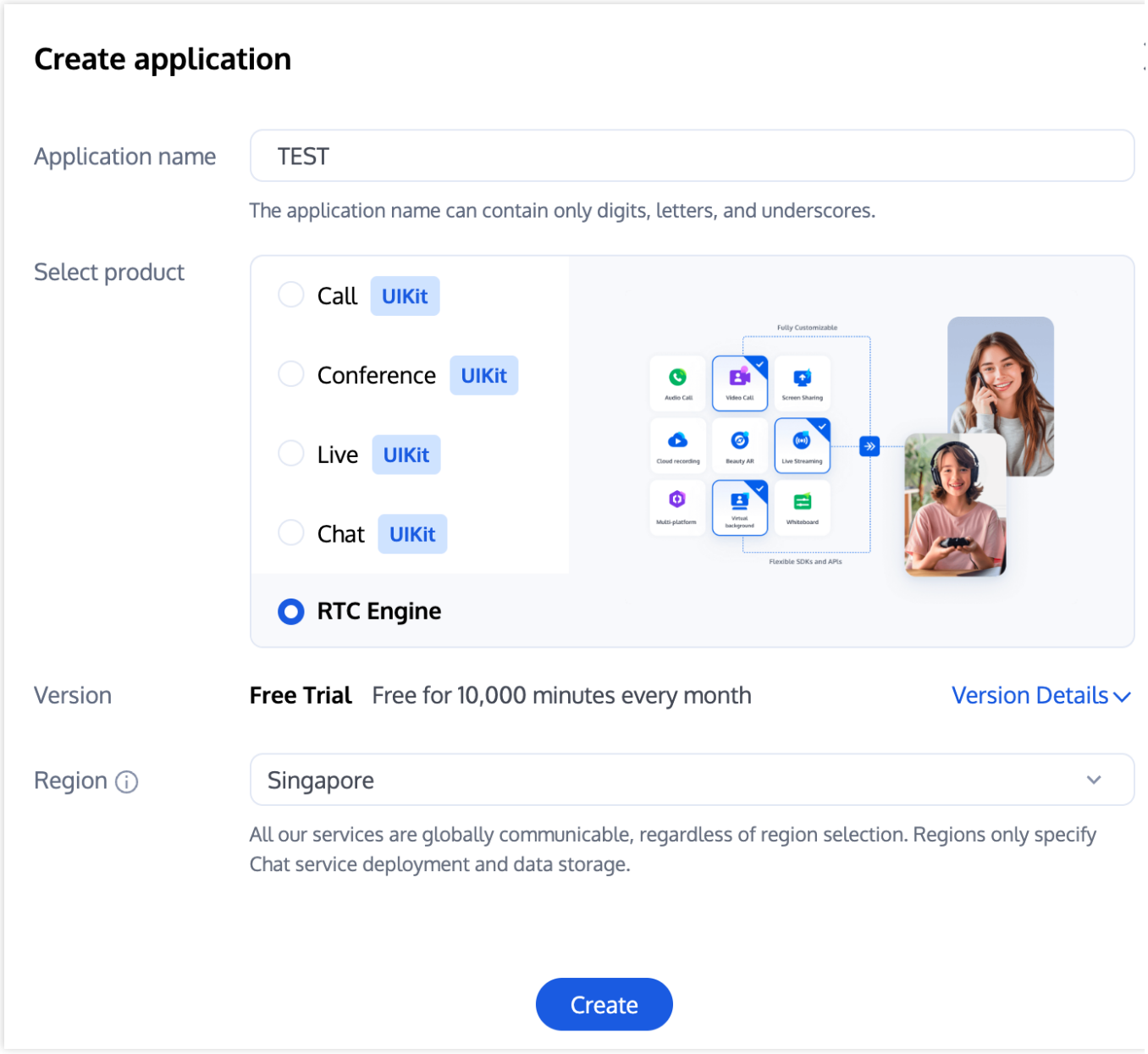
The online karaoke scenarios usually require two paid PaaS services from Tencent Cloud: Tencent Real-Time Communication (TRTC) and Intelligent Music Solution for construction. TRTC is responsible for providing real-time audio and video interaction capabilities. Intelligent Music Solution is responsible for providing lyric recognition, smart composition, music recognition, and music scoring capabilities.

Activate TRTC service.

Activate the Intelligent Music service.

1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



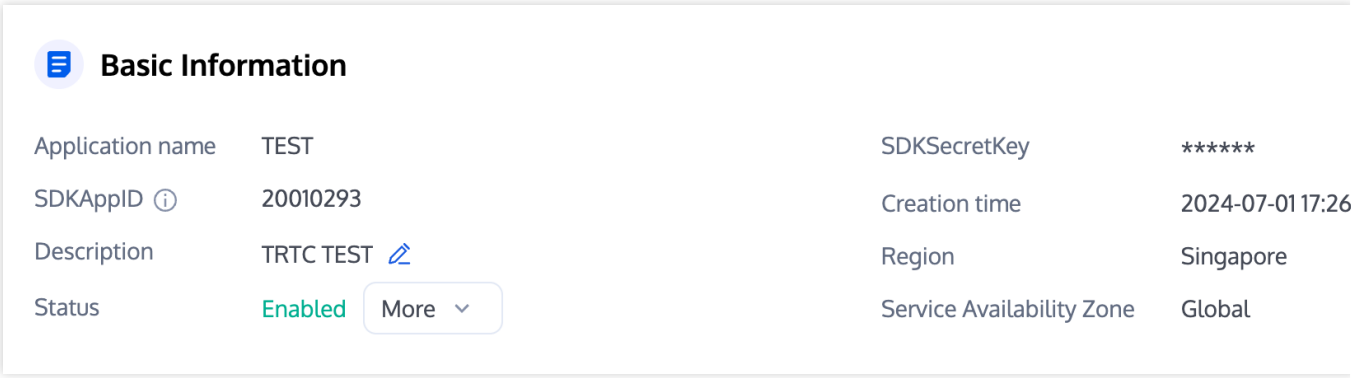
Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

2. After an application is created, you can see the basic information of the application in the Application Management - Application Overview section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.

©2013-2022 Tencent Cloud. All rights reserved.



Preparation

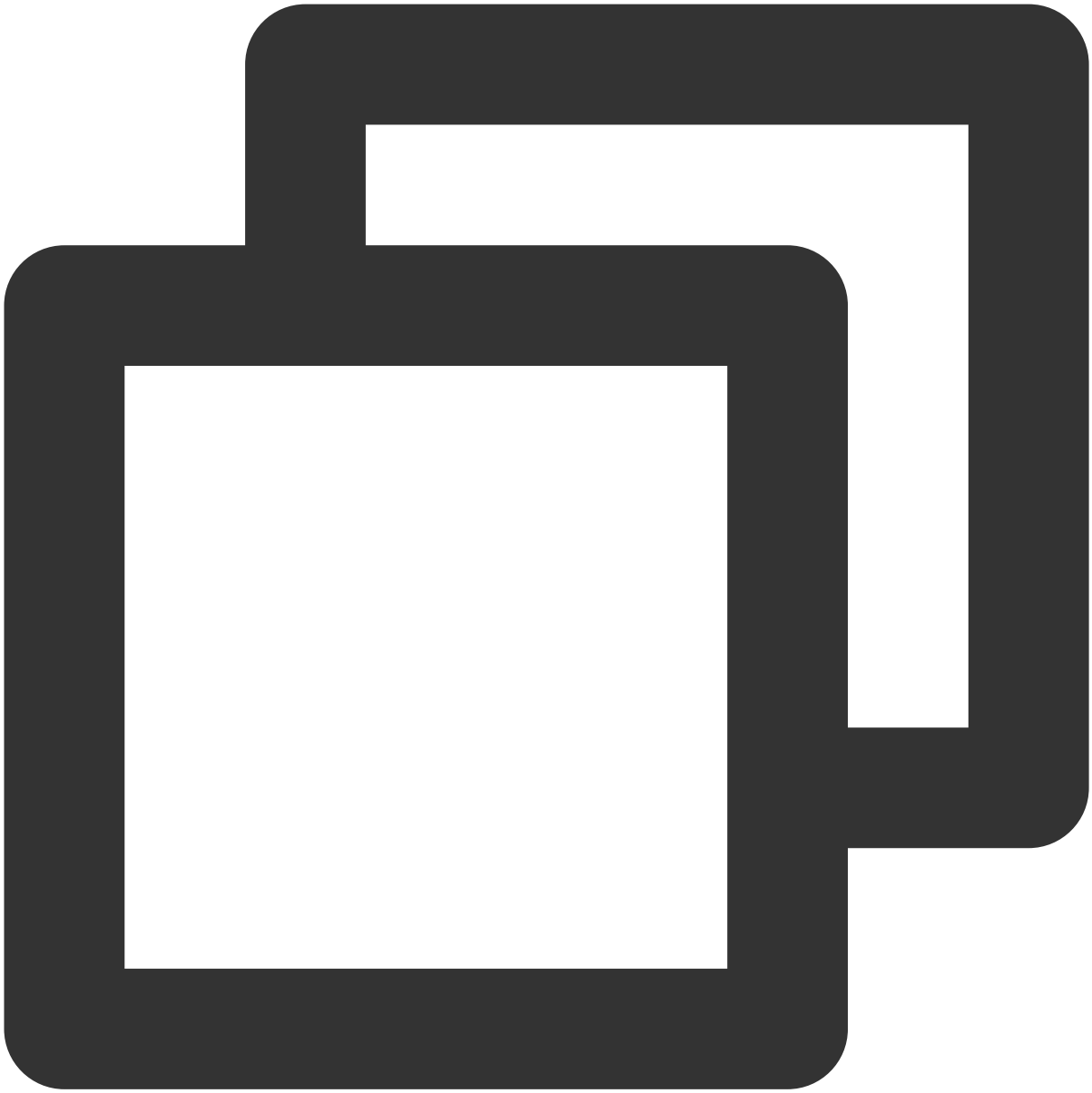
1. Go to the Purchase Page to activate the music service, and choose the appropriate features such as music scoring to activate.
2. Create an AK/SK Key Pair in CAM (namely, a programmable access user that does not require log-in or any user permissions).
3. Create a COS Bucket, and in the COS Bucket Management interface, authorize the read and write permissions of the COS Bucket to the created programmable access user.
4. Prepare the parameters.

operateUin: Tencent Cloud sub-user's account ID.
cosConfig: COS related parameters.
secretId: Bucket's secretId.
secretKey: Bucket's secretKey.
bucket: Bucket's name.
region: Bucket's region, for example, ap-guangzhou.

Activation and registration.

After the preparation is completed, proceed with registration activation by initiating a request, with an estimated wait time of about 2 minutes.
Initiate request.
Request result:

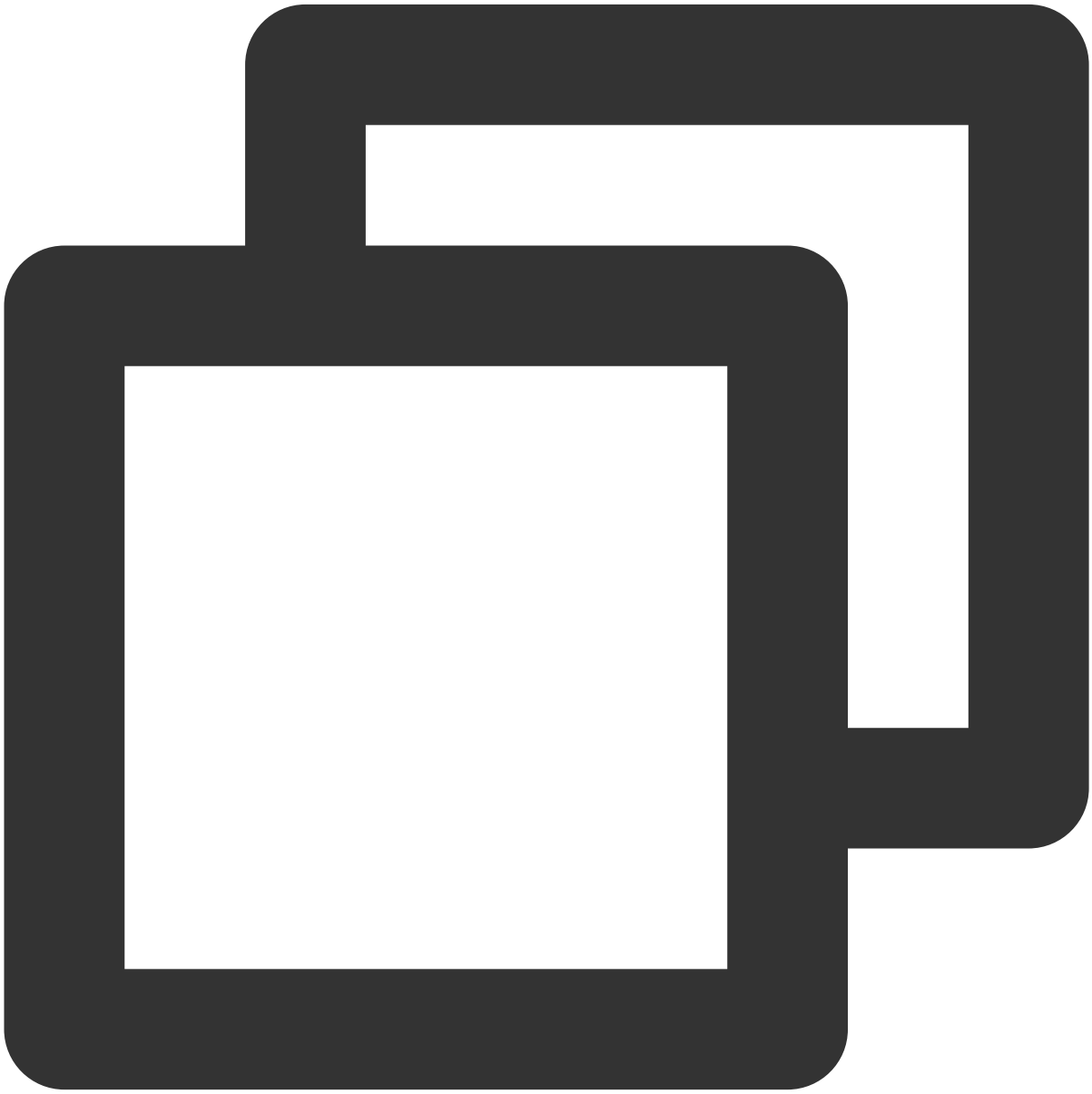
©2013-2022 Tencent Cloud. All rights reserved.



```
curl -X POST \\
  http://service-mqk0mc83-1257411467.bj.apigw.tencentcs.com/release/register \\
  -H 'Content-Type: application/json' \\
  -H 'Cache-control: no-cache' \\
  -d '{
    "requestId": "test-regisiter-service",
    "action": "Register",
    "registerRequest": {
      "operateUin": <operateUin>,
      "userName": <customedName>,
      "cosConfig": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "secretId": <CosConfig.secretId>,
        "secretKey": <CosConfig.secretKey>,
        "bucket": <CosConfig.bucket>,
        "region": <CosConfig.region>
      }
    }
  }'
```



```
{
    "requestId": "test-regisiter-service",
    "registerInfo": {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "tmpContentId": <tmpContentId>,
        "tmpSecretId": <tmpSecretId>,
        "tmpSecretKey": <tmpSecretKey>,
        "apiGateSecretId": <apiGateSecretId>,
        "apiGateSecretKey": <apiGateSecretKey>,
        "demoCosPath": "UIN_demo/run_musicBeat.py",
        "usageDescription": "Download the python version demo file [UIN_demo/run_mu
        "message": "Registration successful, and thank you for registering.",
        "createdAt": <createdAt>,
        "updatedAt": <updatedAt>
    }
}
```

Run verification.

After the above activation and registration service are completed, a python version executable demo example based on music beat recognition capability will be generated in the `demoCosPath` directory. Execute the command `python run_musicBeat.py` in a networked environment for verification.

Note:

For more detailed intelligent music solution integration instructions, see Integration Guide.

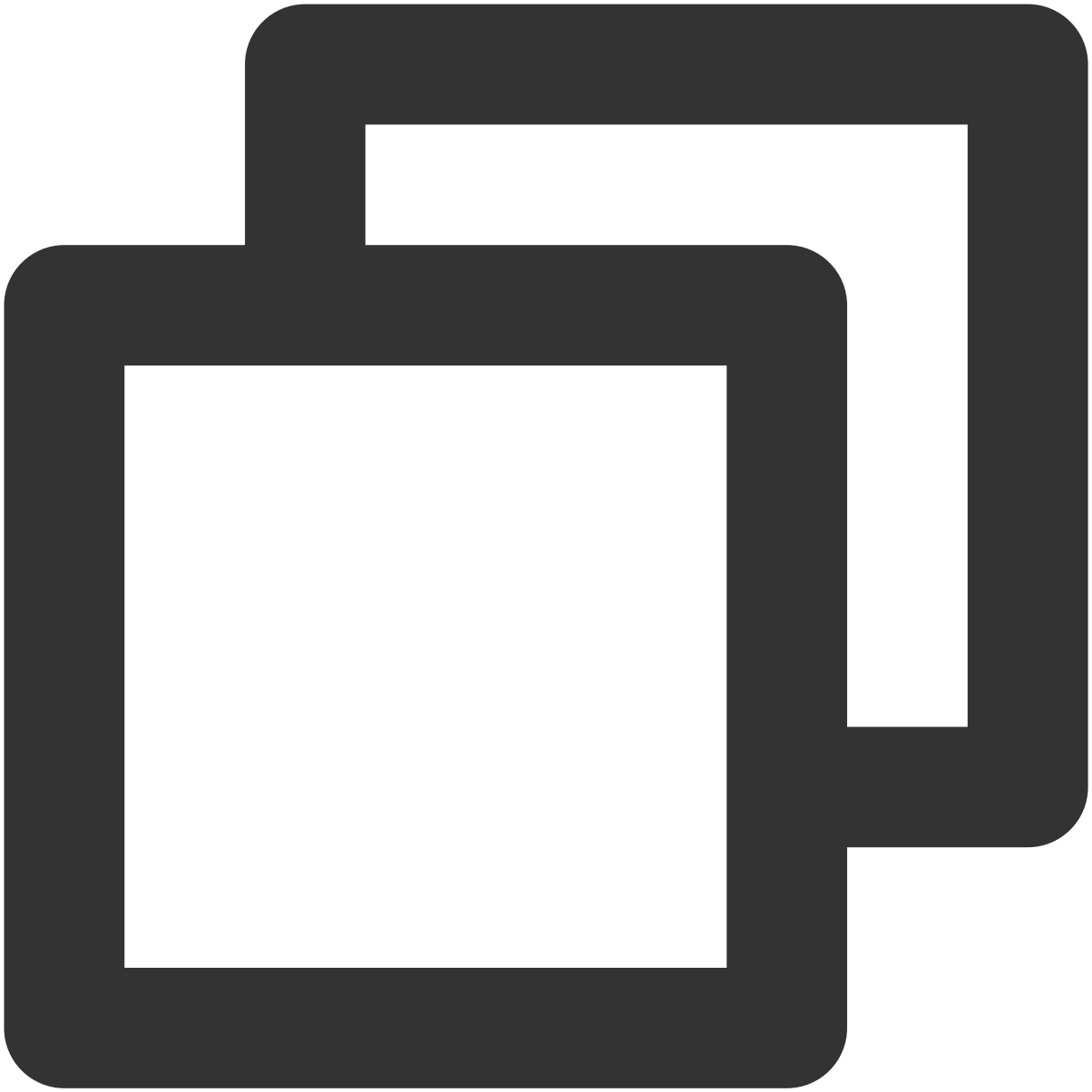
Step 2: Importing SDK.

The TRTC SDK is now available on **CocoaPods**. We recommend integrating the SDK via CocoaPods.

1. Install CocoaPods.

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

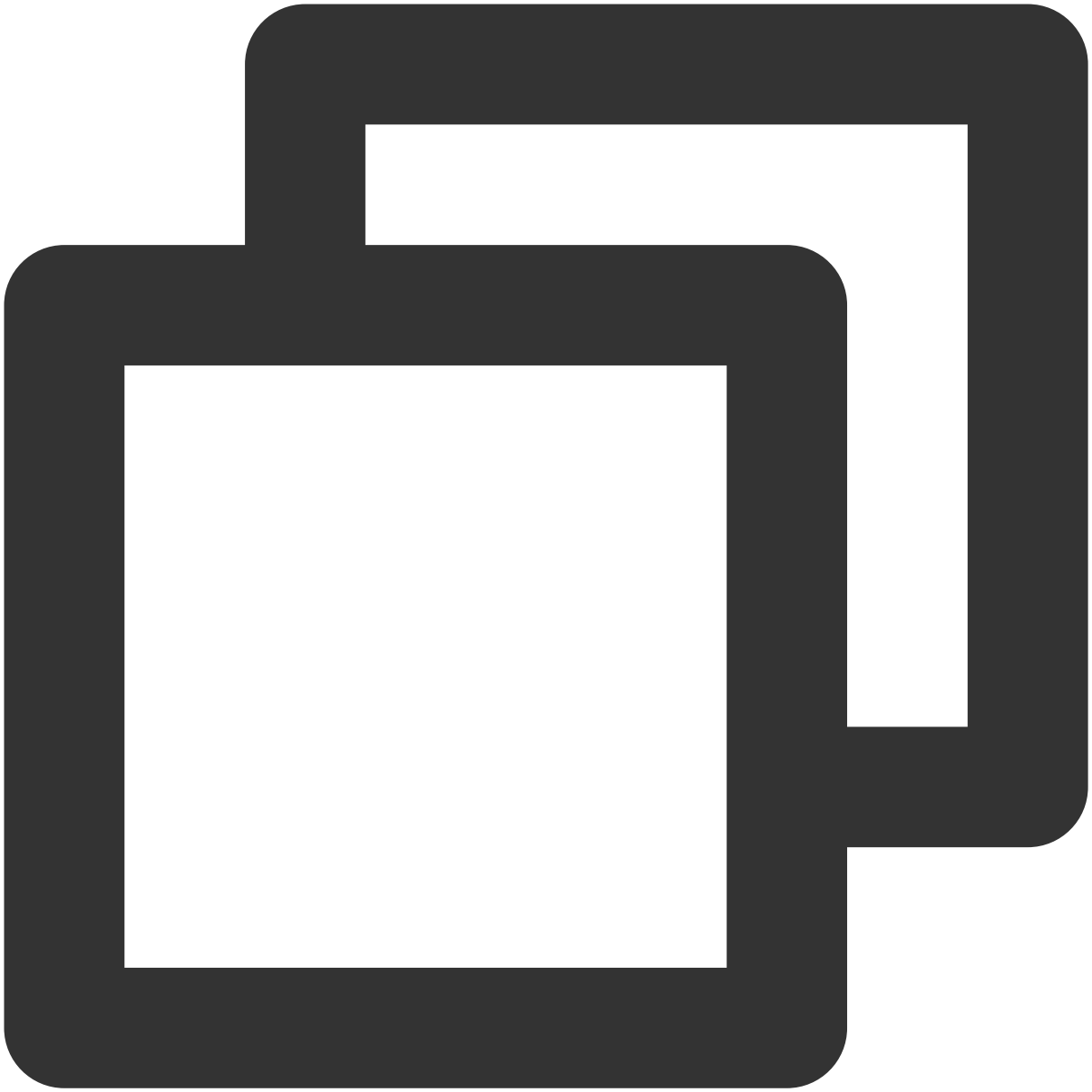


```
sudo gem install cocoapods
```

2. Create a Podfile.

Go to the project directory, and enter the following command. A Podfile file will then be created in the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

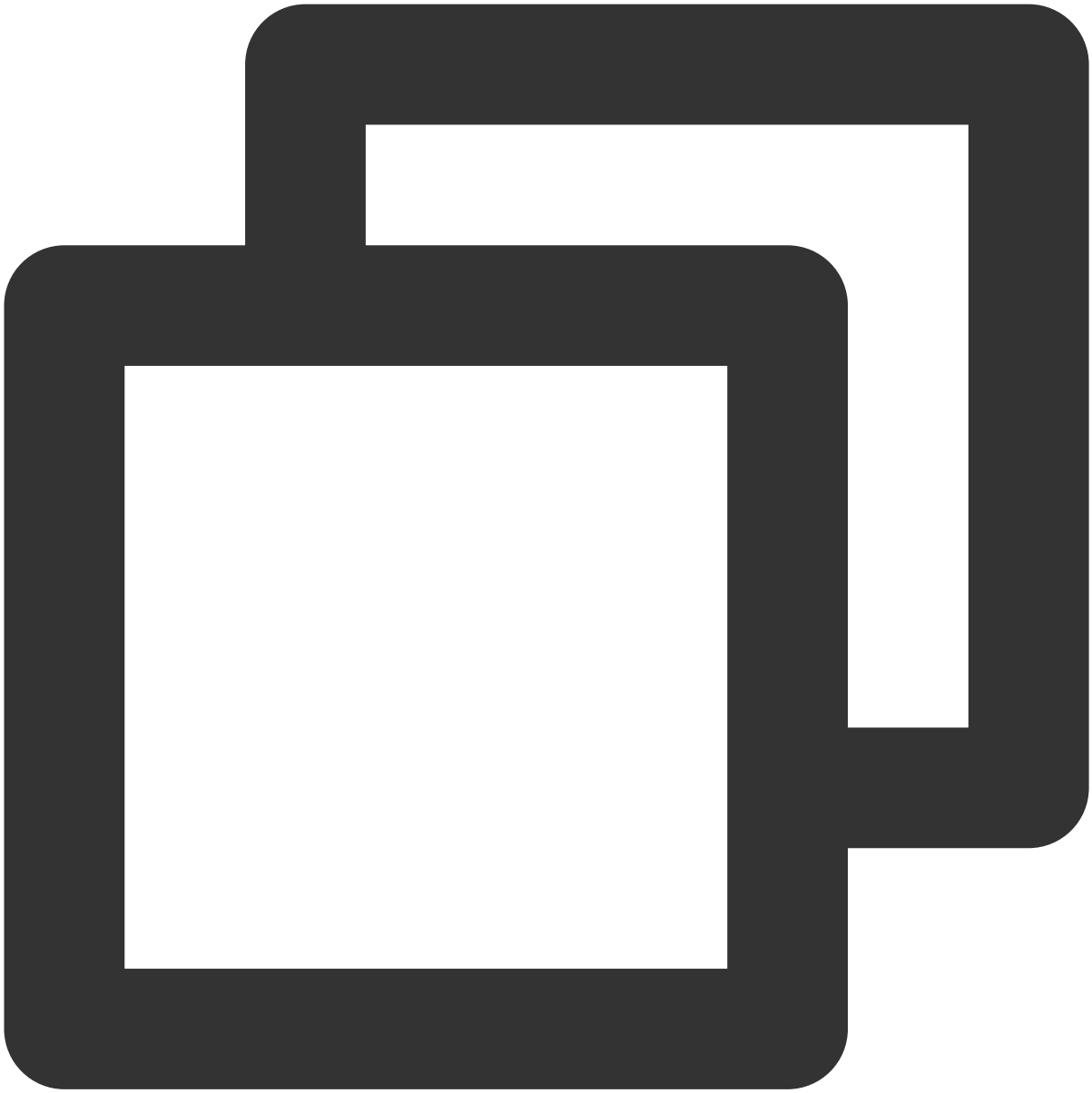


```
pod init
```

3. Edit the Podfile.

Choose the appropriate version for your project and edit the Podfile.

©2013-2022 Tencent Cloud. All rights reserved.



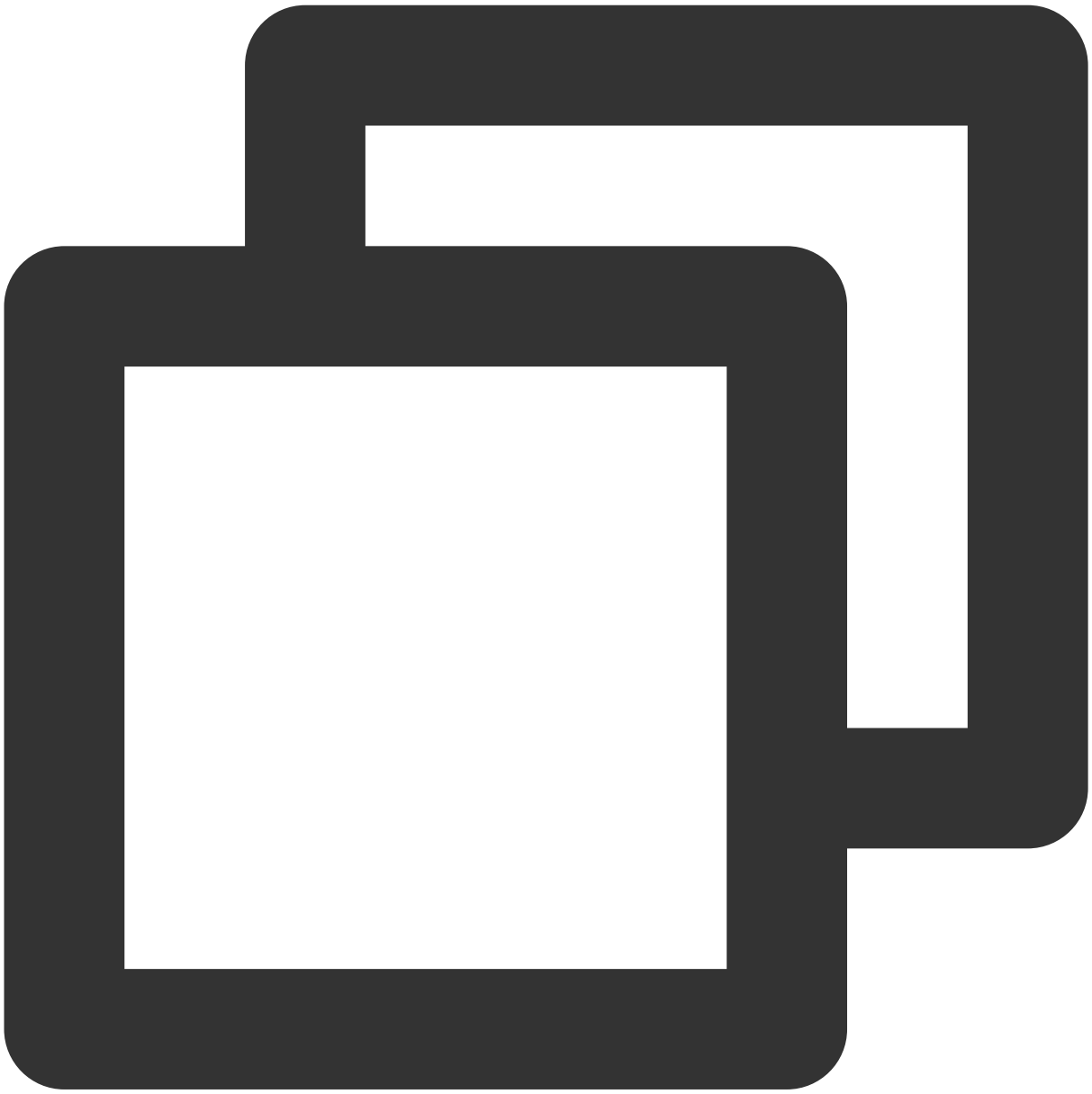
```
platform :ios, '8.0'
    target 'App' do

    # TRTC Lite Edition
    # The installation package has the minimum incremental size. But it only suppor
    pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd

    # Pro Edition
    # Includes a wide range of features such as Real-Time Communication (TRTC), TXL
    # pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/po
```

©2013-2022 Tencent Cloud. All rights reserved.

```
end
```

4. Update and install the SDK.

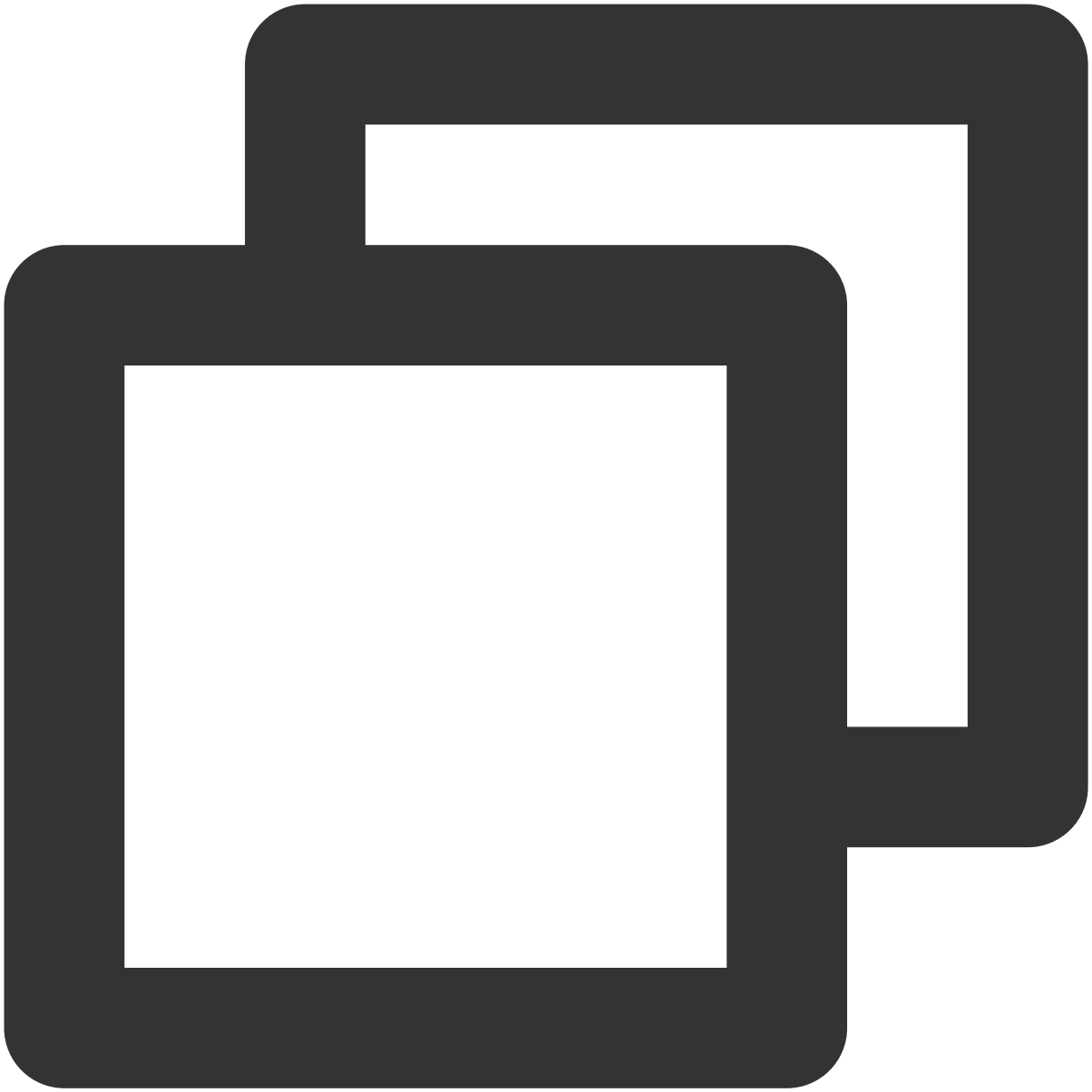
Enter the following command in a terminal window to update the local repository files and install the SDK.



```
pod install
```

Or use the following command to update the local repository.

©2013-2022 Tencent Cloud. All rights reserved.



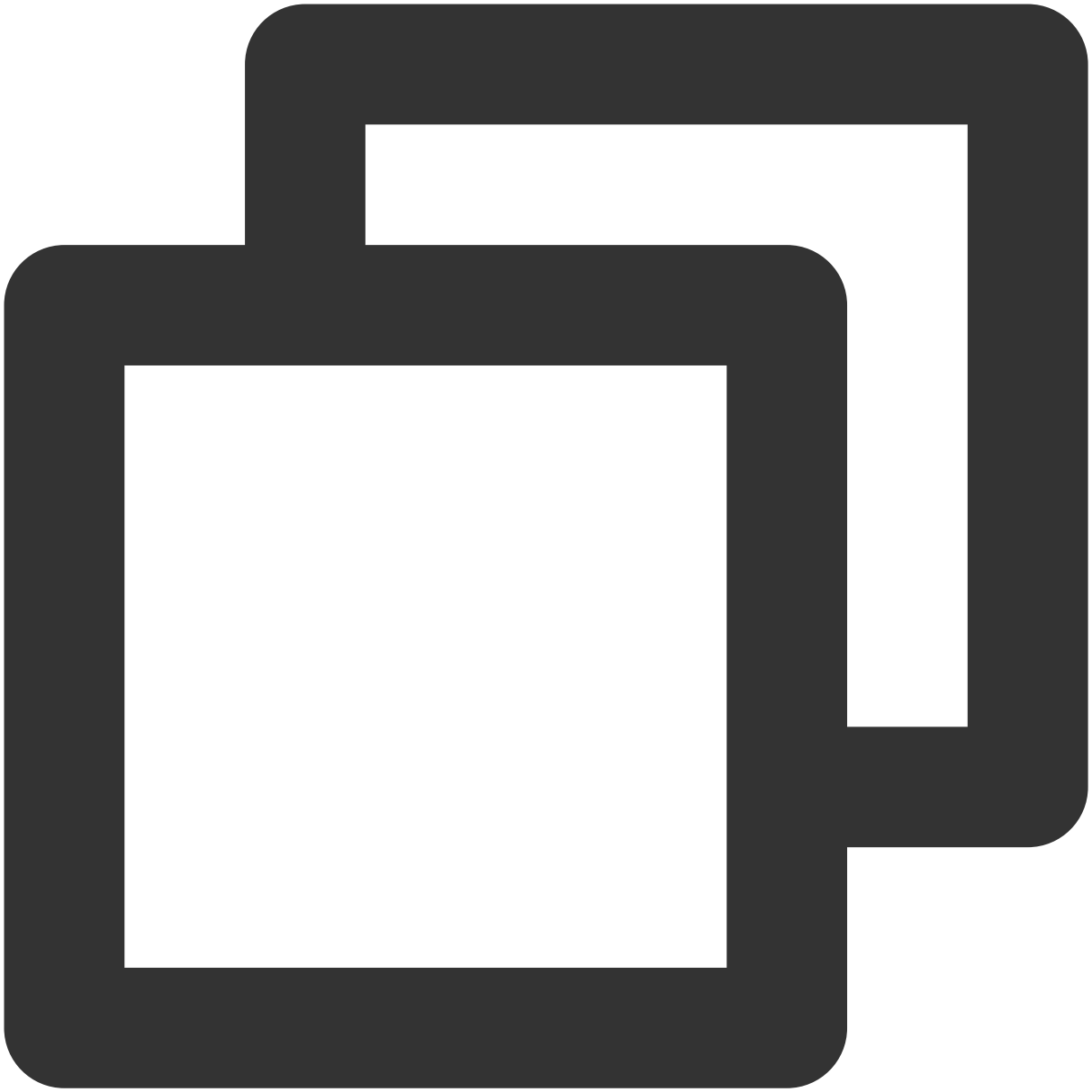
```
pod update
```

Upon the completion of pod command execution, an .xcworkspace project file integrated with the SDK will be generated. Double-click to open it.

Note:

If the pod search fails, it is recommended to try updating the local repo cache of pod. The update command is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



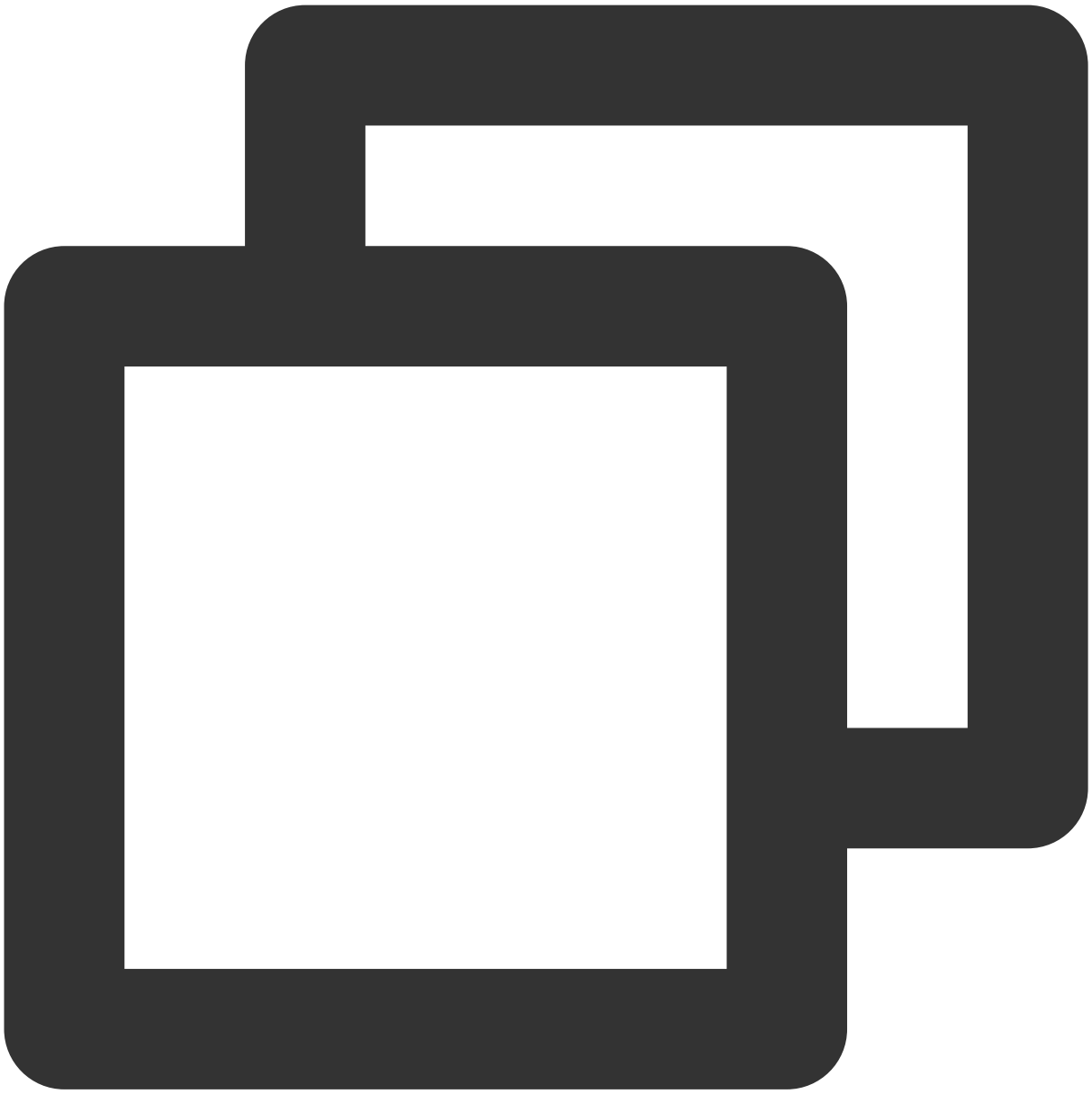
```
pod setup
pod repo update
rm ~/Library/Caches/CocoaPods/search_index.json
```

Besides CocoaPods integration, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK.

Step 3: Project configuration.

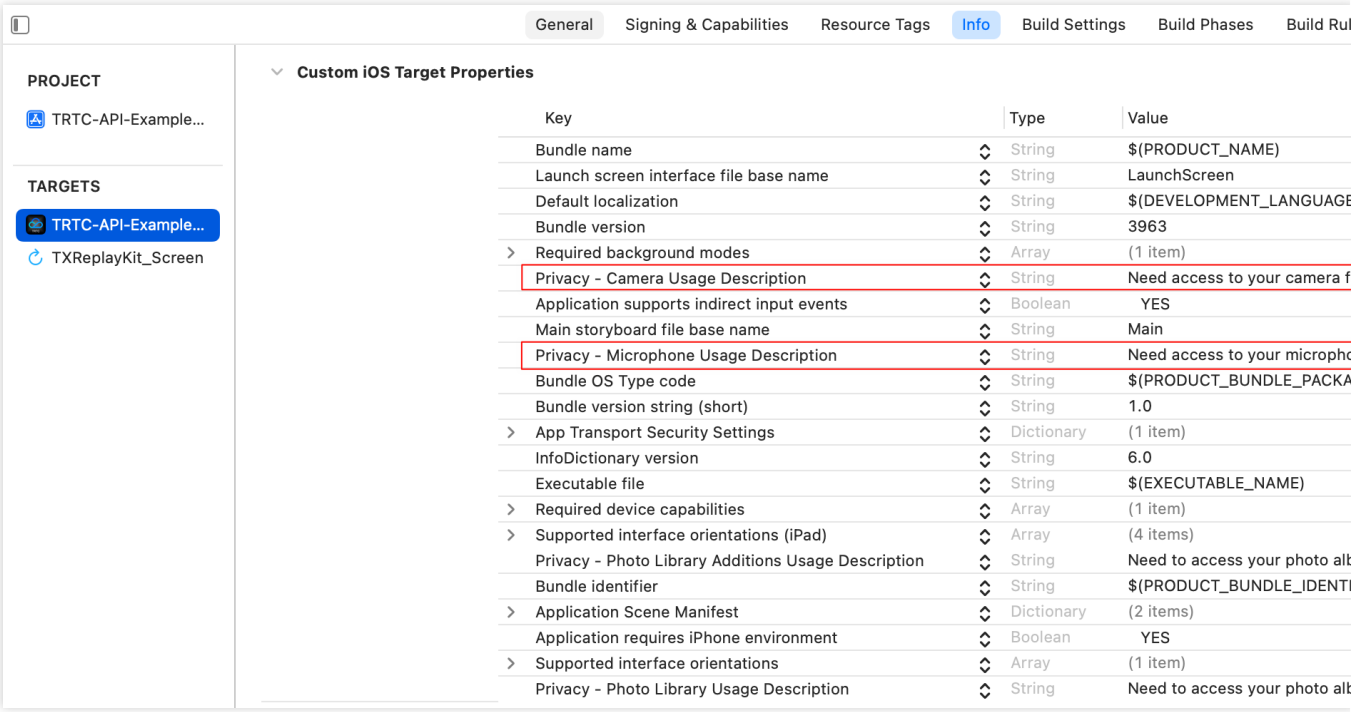
©2013-2022 Tencent Cloud. All rights reserved.

1. In karaoke scenarios, the TRTC SDK needs to be authorized for microphone permissions. Add the following content to your app's Info.plist. It corresponds to the system's prompt message when microphone permission is requested.



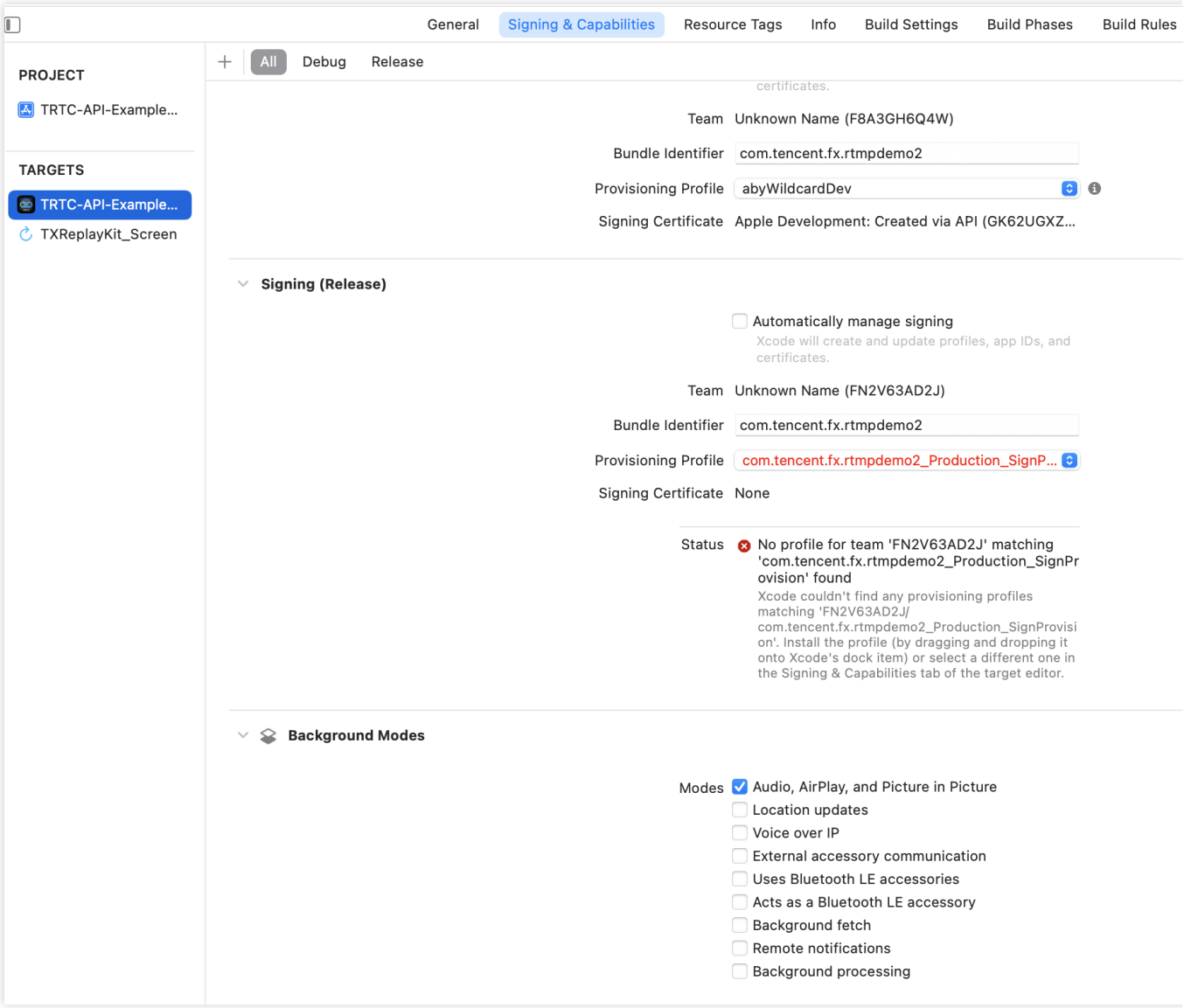
```
Privacy - Microphone Usage Description. Also enter a prompt specifying the purpose
```

©2013-2022 Tencent Cloud. All rights reserved.



2. If you need your App to continue running certain features in the background, go to XCode. Choose your current project. Under Capabilities, set the settings for Background Modes to ON, and check Audio, AirPlay, and Picture in Picture, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Step 4: Authentication and authorization.

UserSig is a security protection signature designed by Tencent Cloud to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when it enters the room.

Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.

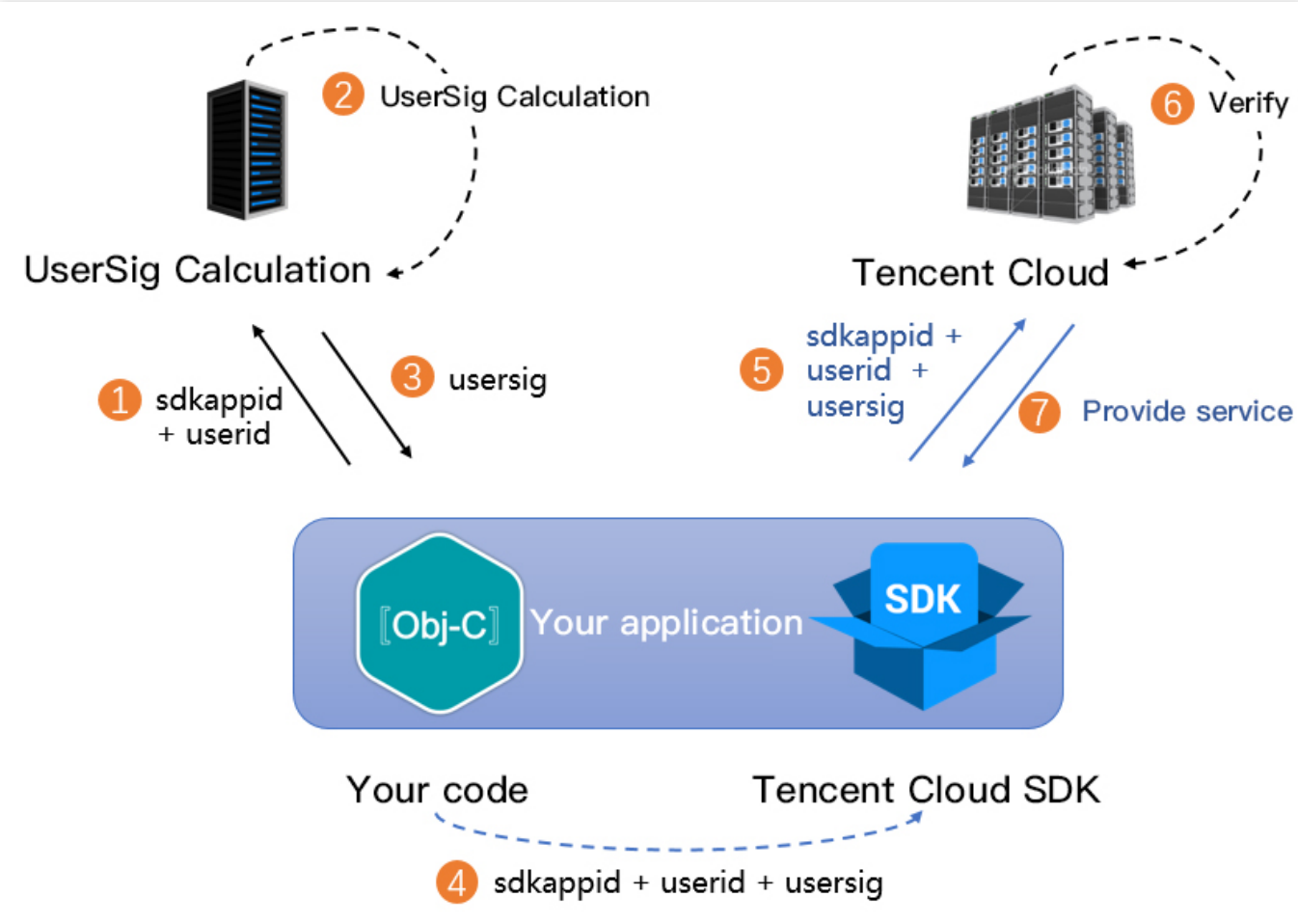
Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.

The specific implementation process is as follows:

1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.
3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.

©2013-2022 Tencent Cloud. All rights reserved.

5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.



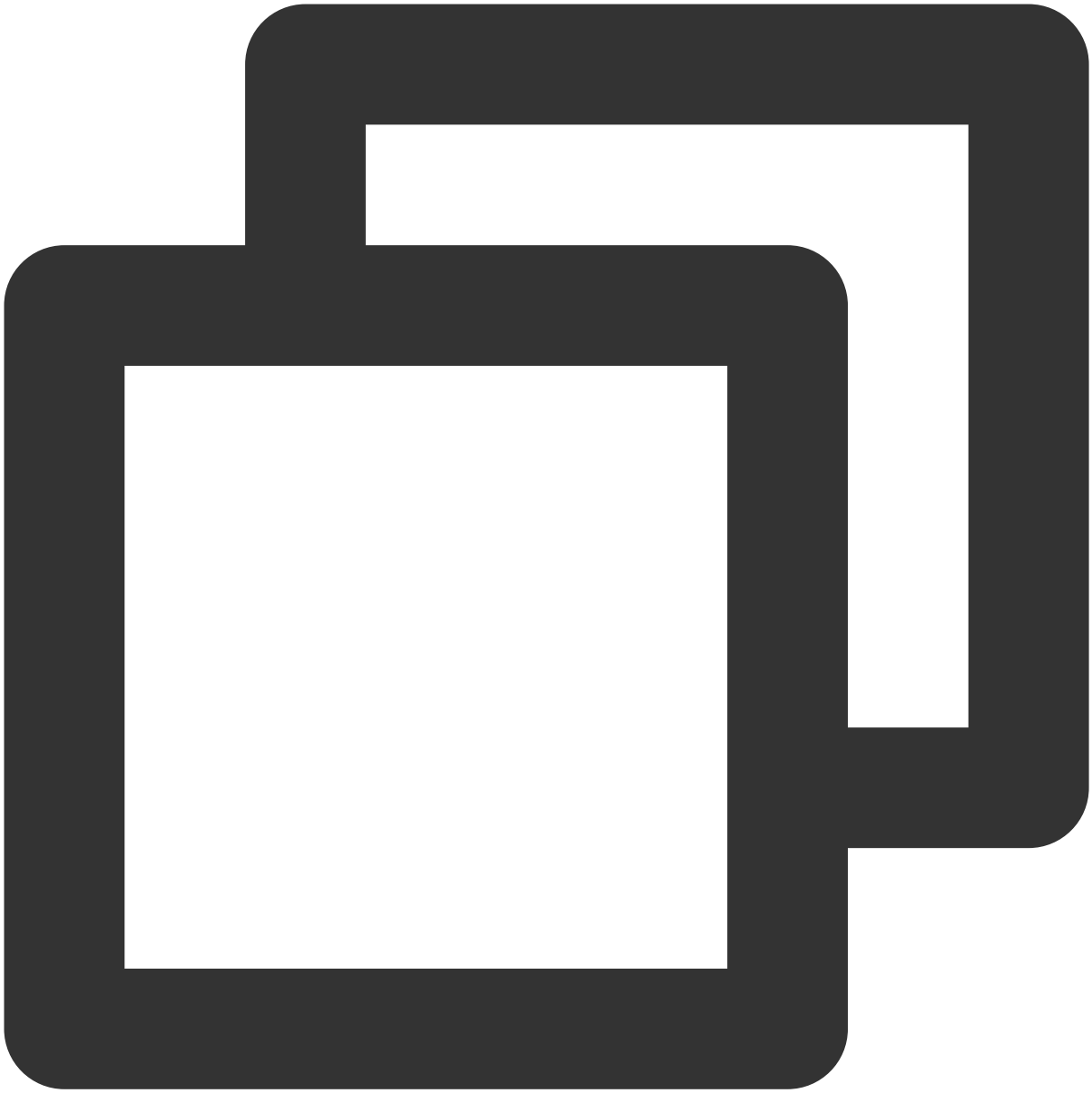
Note:

The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Step 5: Initializing the SDK.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (Single Instance Pattern).
self.trtcCloud = [TRTCCloud sharedInstance];
// Set event listeners.
self.trtcCloud.delegate = self;

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
- (void)onError:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg extInfo:(
    NSLog(@"%d: %@", errCode, errMsg);
}

- (void)onWarning:(TXLiteAVWarning)warningCode warningMsg:(nullable NSString *)warn
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    NSLog(@"%d: %@", warningCode, warningMsg);
}

// Remove event listener.
self.trtcCloud.delegate = nil;
// Terminate TRTC SDK instance (Singleton Pattern).
[TRTCCloud destroySharedIntance];
```

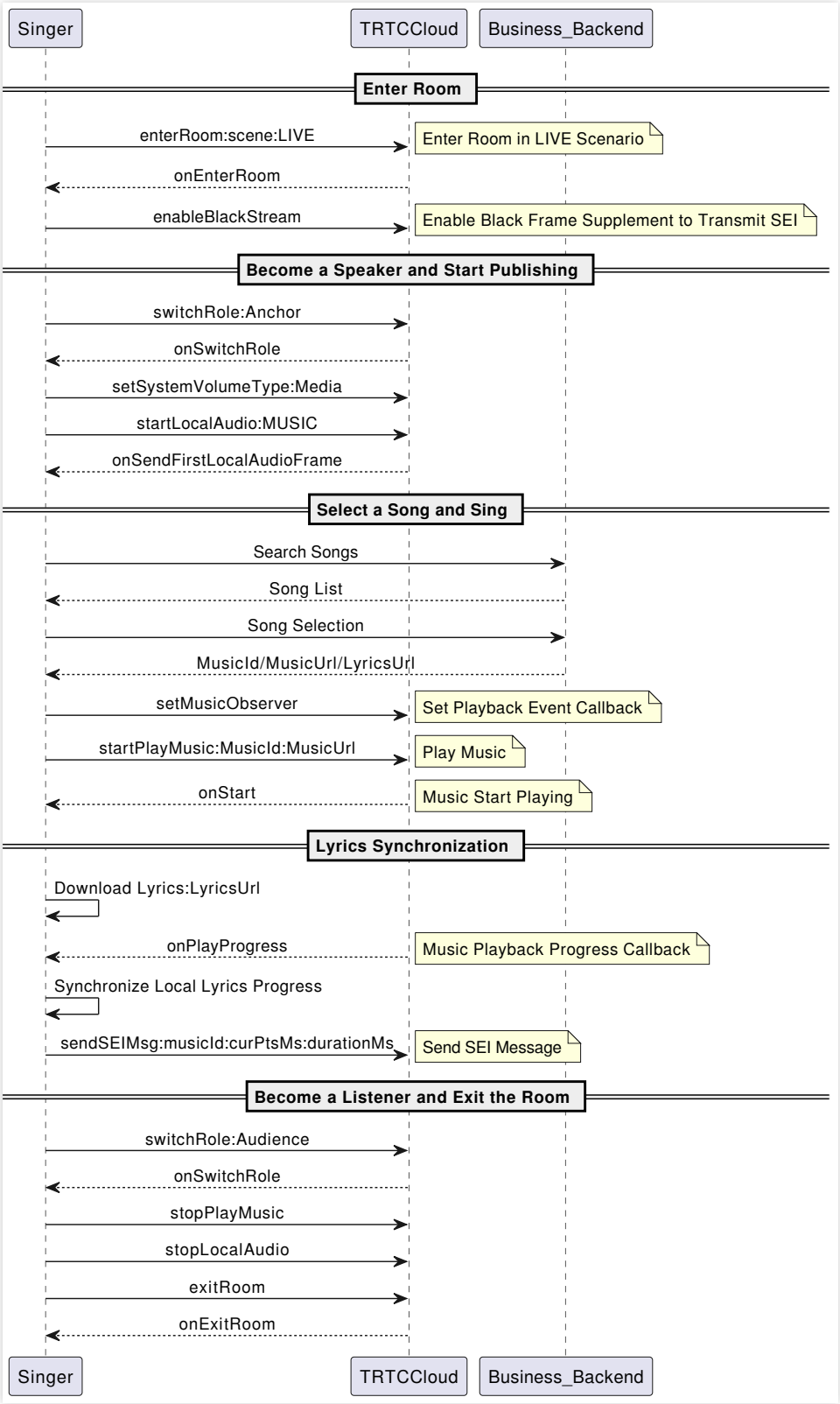
Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

Scenario 1: Solo singing turn-taking

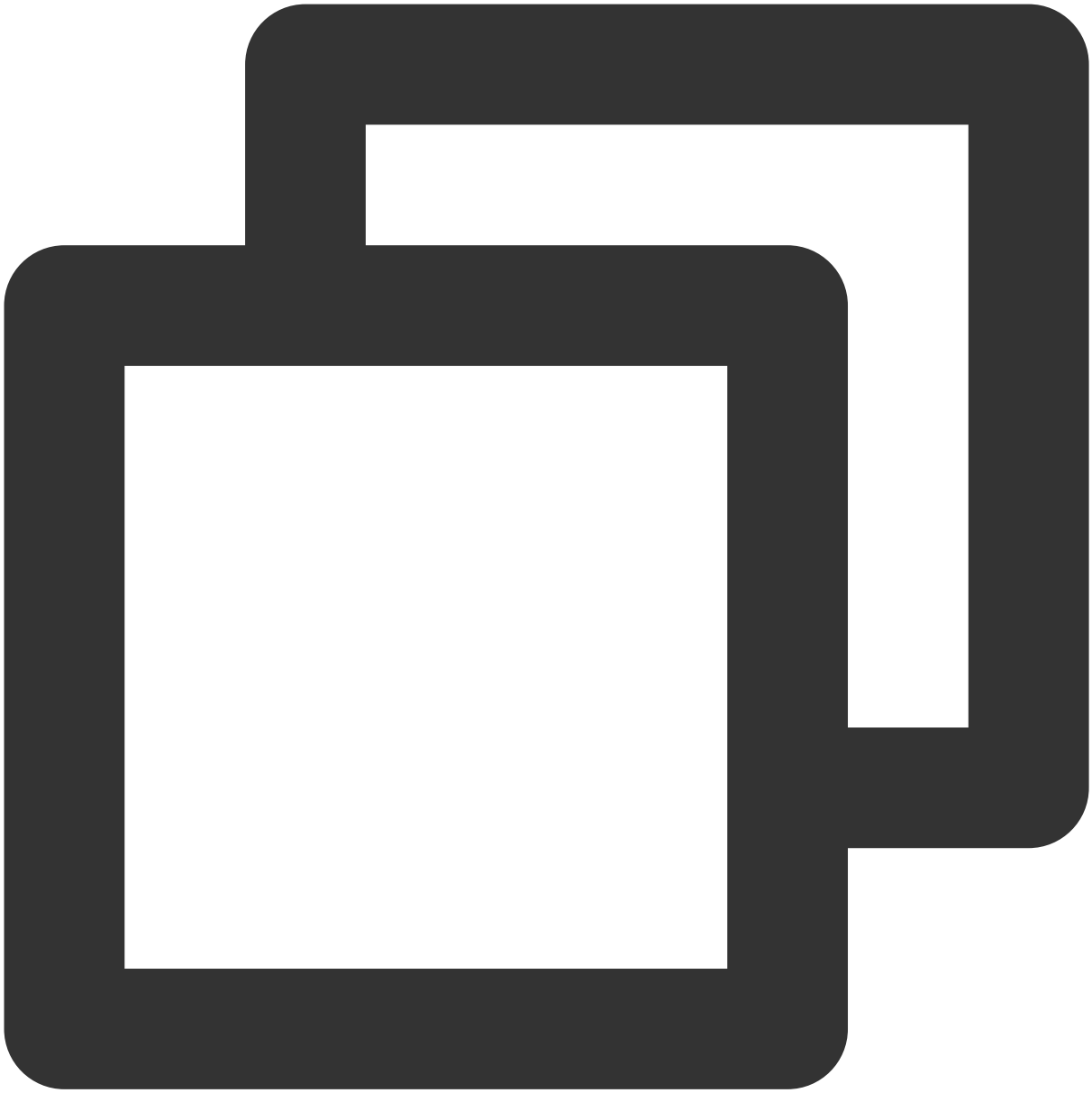
Perspective 1: Performer actions

Sequence diagram



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



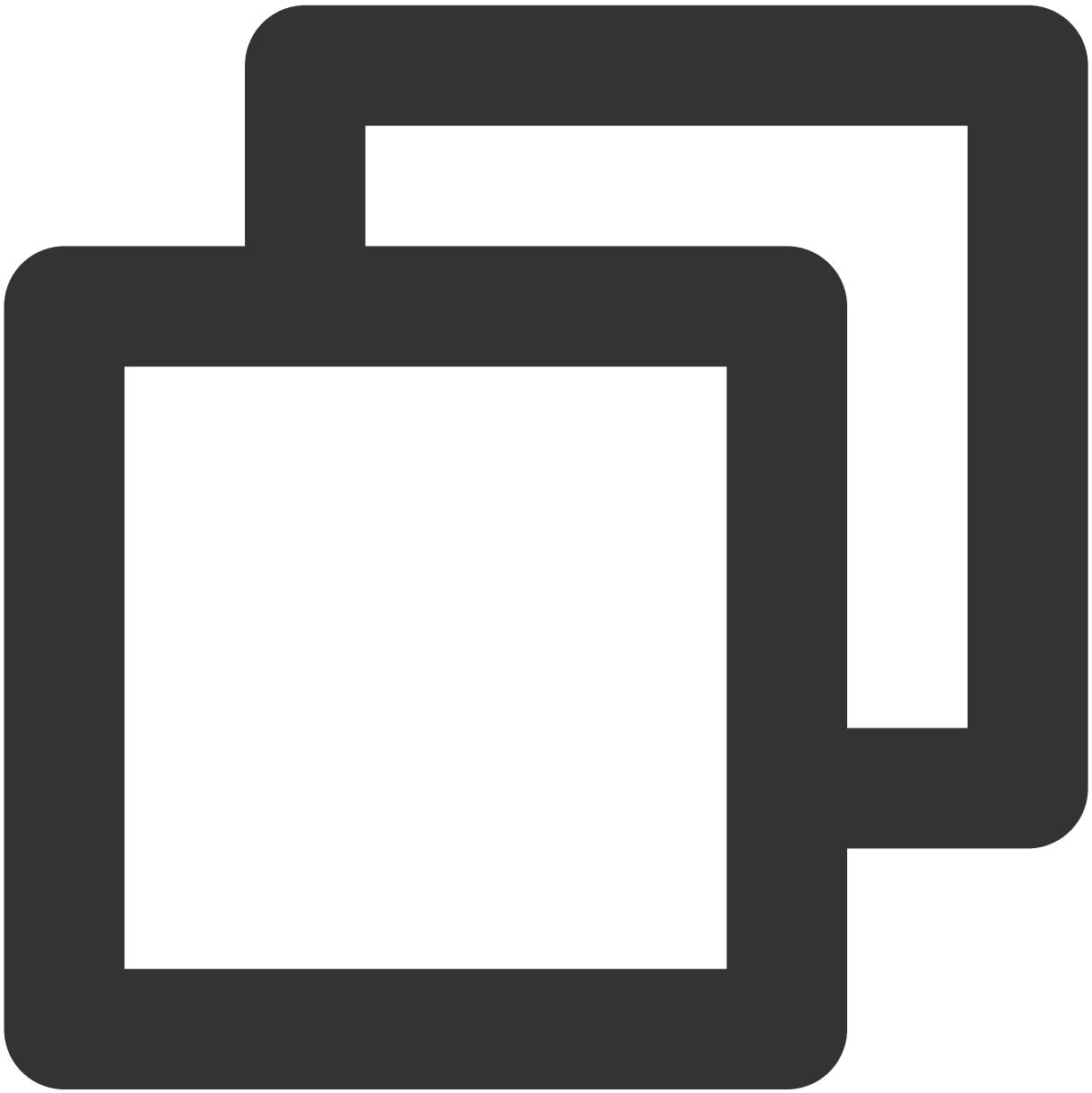
```
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = [self generateUserSig:userId];
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}
```

Note:

To better transmit SEI messages for lyric synchronization, it is recommended to choose `TRTCAppSceneLIVE` for room entry scenarios.



```
// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
```

©2013-2022 Tencent Cloud. All rights reserved.

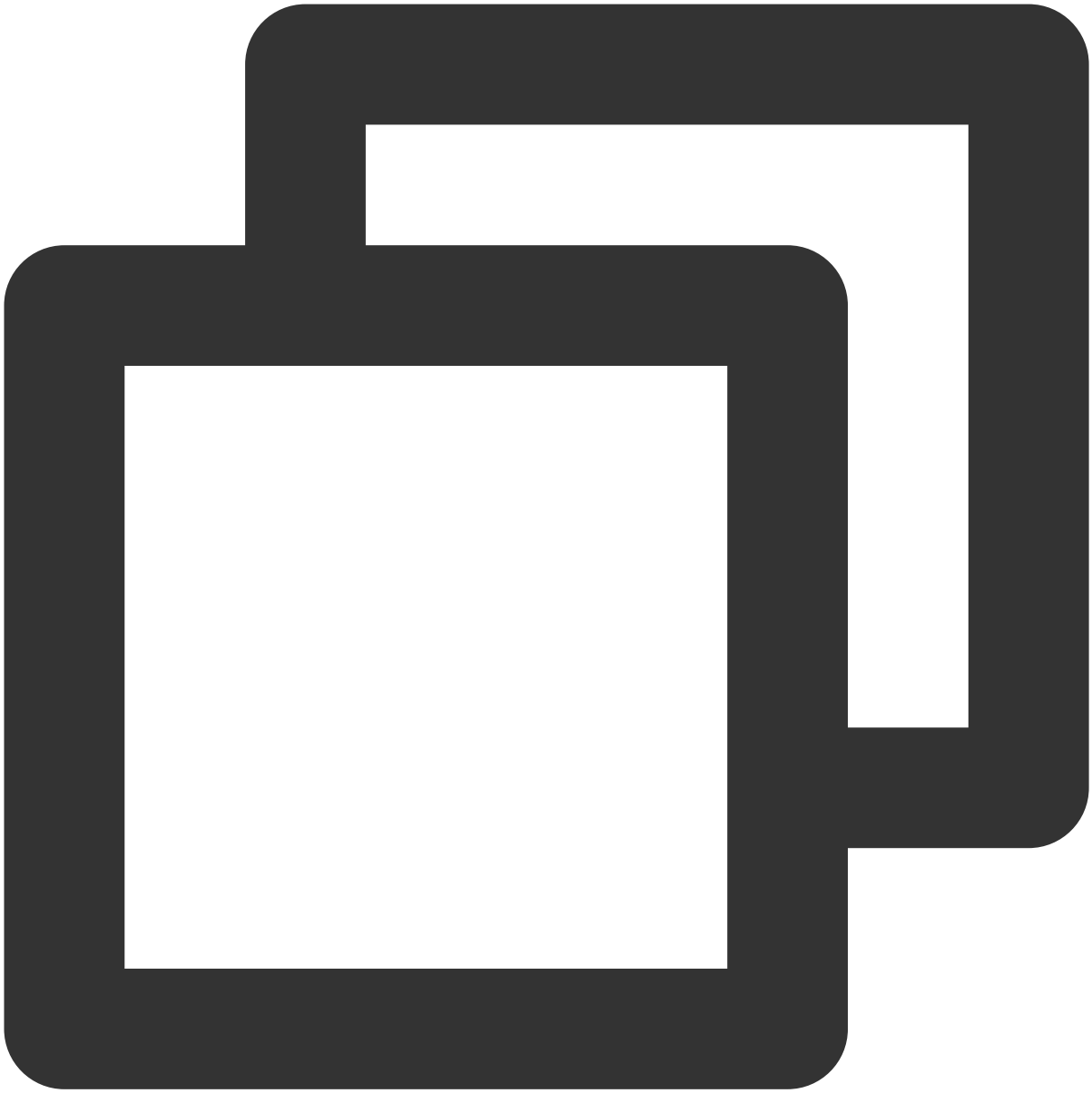
```
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
        // Enable the experimental API for black frame insertion.
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableBlackStream\\",\\
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

Under the pure audio mode, the performer needs to enable the insertion of black frames to carry SEI messages. This API should be called after successfully entering the room.

2. Go live on streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switched to the anchor role.
[self.trtcCloud switchRole:TRTCRoleAnchor];

// Event callback for switching the role.
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Set media volume type.
        [self.trtcCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
        // Upstream local audio streams and set audio quality.
        [self.trtcCloud startLocalAudio:TRTCAudioQualityMusic];
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Note:

In karaoke scenarios, it is recommended to set the full-range media volume and music quality to achieve a high-fidelity listening experience.

3. Song selection and performance.

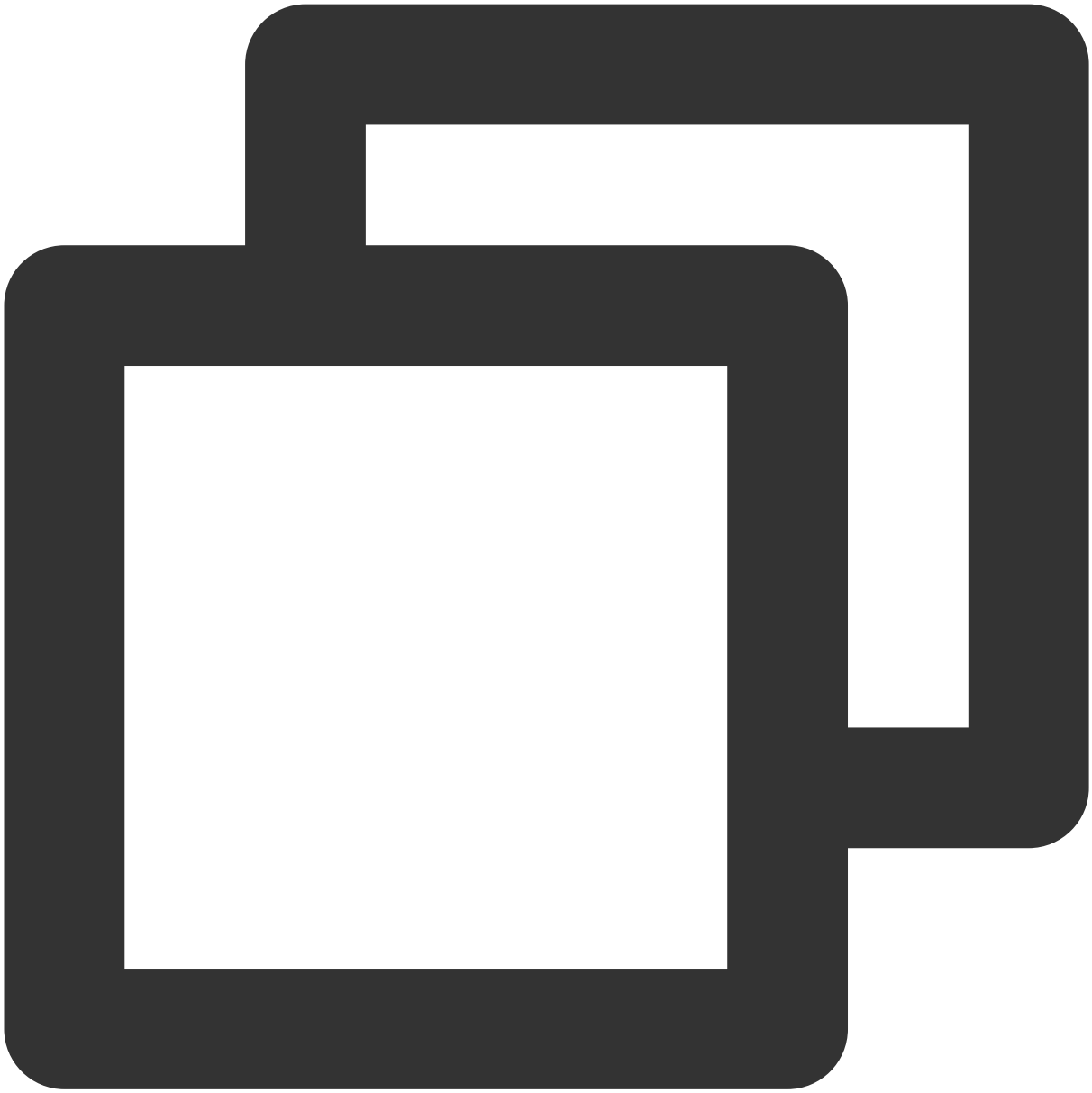
Search for songs, and obtain music resources.

Search for songs and acquire music resources through the business backend. Obtain identifiers such as the MusicId, the song's URL (MusicUrl), and the lyrics URL (LyricsUrl).

It is recommended that the business side select an appropriate music repository production to provide licensed music resources.

Play accompaniment and start singing.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain audio effects management.
self.audioEffectManager = [self.trtcCloud getAudioEffectManager];

// originMusicId: Custom identifier for the original vocal music. originMusicUrl: U
TXAudioMusicParam *originMusicParam = [[TXAudioMusicParam alloc] init];
originMusicParam.ID = originMusicId;
originMusicParam.path = originMusicUrl;
// Whether to publish the original vocal music to remote (otherwise play locally on
originMusicParam.publish = YES;

// accompMusicId: Custom identifier for the accompaniment music. accompMusicUrl: UR
```

©2013-2022 Tencent Cloud. All rights reserved.

```
TXAudioMusicParam *accompMusicParam = [[TXAudioMusicParam alloc] init];
accompMusicParam.ID = accompMusicId;
accompMusicParam.path = accompMusicUrl;
// Whether to publish the accompaniment to remote (otherwise play locally only).
accompMusicParam.publish = YES;

// Start playing the original vocal music.
[self.audioEffectManager startPlayMusic:originMusicParam onStart:^(NSInteger errCod
    // onStart
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // onProgress
} onComplete:^(NSInteger errCode) {
    // onComplete
}];

// Start playing the accompaniment music.
[self.audioEffectManager startPlayMusic:originMusicParam onStart:^(NSInteger errCod
    // onStart
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // onProgress
} onComplete:^(NSInteger errCode) {
    // onComplete
}];

// Switch to the original vocal music.
[self.audioEffectManager setMusicPlayoutVolume:originMusicId volume:100];
[self.audioEffectManager setMusicPublishVolume:originMusicId volume:100];
[self.audioEffectManager setMusicPlayoutVolume:accompMusicId volume:0];
[self.audioEffectManager setMusicPublishVolume:accompMusicId volume:0];

// Switch to the accompaniment music.
[self.audioEffectManager setMusicPlayoutVolume:originMusicId volume:0];
[self.audioEffectManager setMusicPublishVolume:originMusicId volume:0];
[self.audioEffectManager setMusicPlayoutVolume:accompMusicId volume:100];
[self.audioEffectManager setMusicPublishVolume:accompMusicId volume:100];
```

Note:

In karaoke scenarios, both the original vocal and accompaniment need to be played simultaneously (distinguished by MusicID). The switch between the original vocal and accompaniment is achieved by adjusting the local and remote playback volumes.

If the music being played has dual audio tracks (including both the original vocal and accompaniment), switching between them can be achieved by specifying the music's playback track using setMusicTrack.

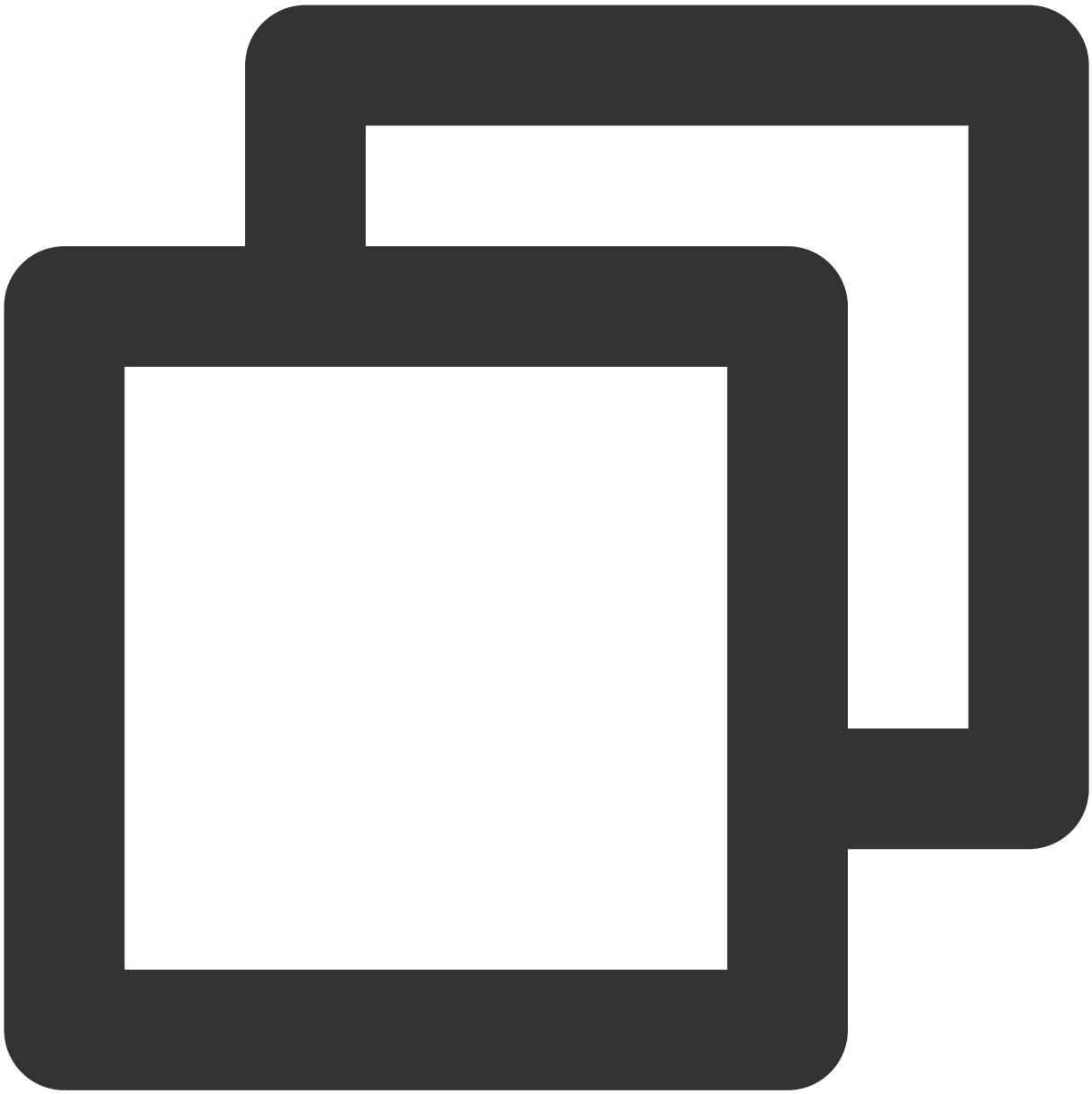
4. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

©2013-2022 Tencent Cloud. All rights reserved.

Synchronize local lyrics, and transmit song progress via SEI.



```
[self.audioEffectManager startPlayMusic:musicParam onStart:^(NSInteger errCode) {
    // Start playing music.
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // Determine whether seek is needed based on the latest progress and the local
    // Song progress is transmitted by sending an SEI message.
    NSDictionary *dic = @{
        @"musicId": @(self.musicId),
        @"progress": @(progressMs),
        @"duration": @(durationMs),
```

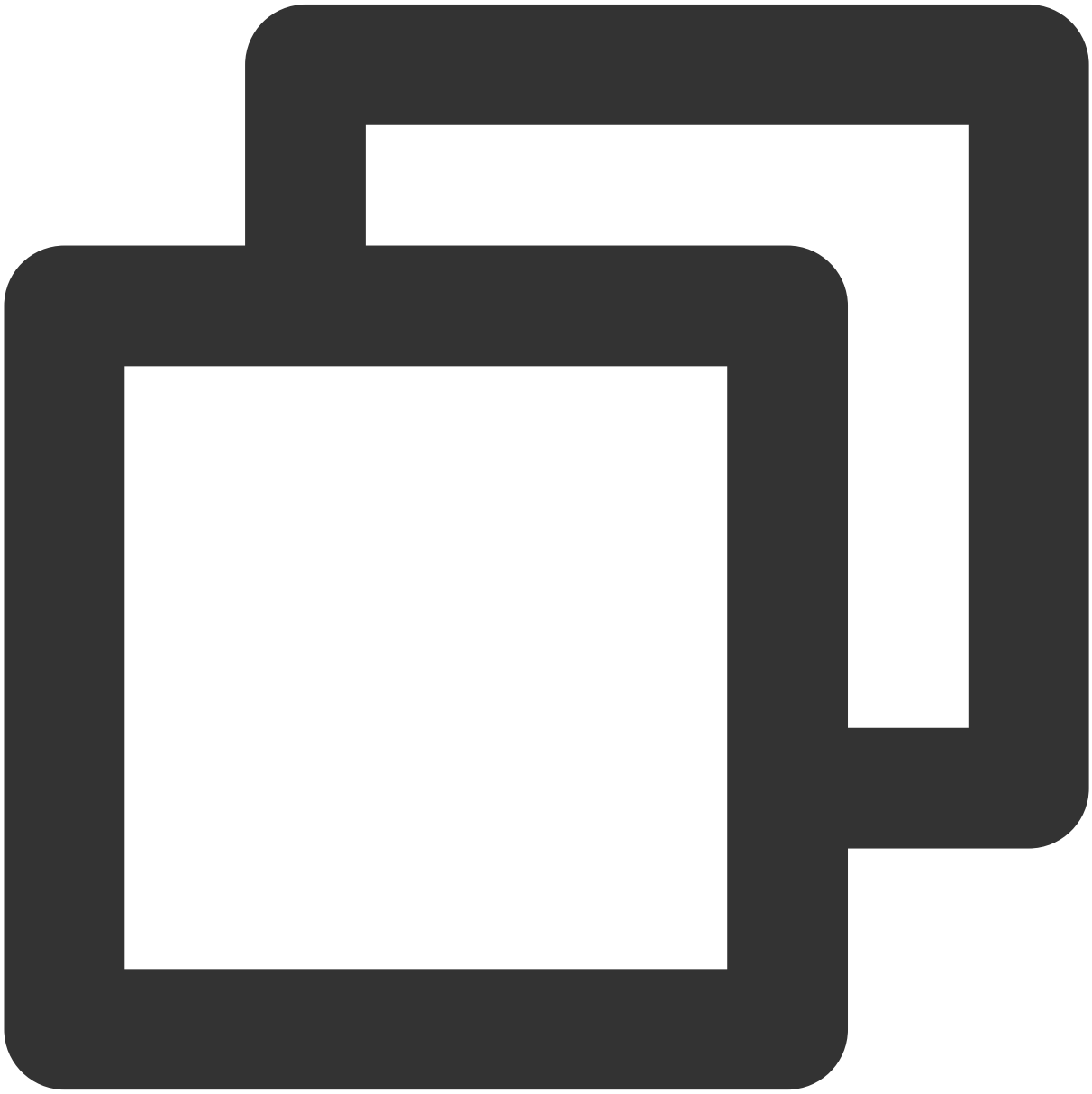
©2013-2022 Tencent Cloud. All rights reserved.

```
    };
    JSONModel *json = [[JSONModel alloc] initWithDictionary:dic error:nil];
    [self.trtcCloud sendSEIMsg:json.toJSONData repeatCount:1];
} onComplete:^(NSInteger errCode) {
    // Music playback completed.
}];
```

Note:

The frequency of the SEI messages sent by the performer is determined by the event callback frequency. Also, the playback progress can be actively synchronized on a schedule through getMusicCurrentPosInMS.

5. Become a listener and exit the room.



©2013-2022 Tencent Cloud. All rights reserved.

```
// Switched to the audience role.
[self.trtcCloud switchRole:TRTCRoleAudience];

// Event callback for switching the role.
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Stop playing accompaniment music.
        [[self.trtcCloud getAudioEffectManager] stopPlayMusic:self.musicId];
        // Stop local audio capture and publishing.
        [self.trtcCloud stopLocalAudio];
    }
}

// Exit the room.
[self.trtcCloud exitRoom];

// Exit room event callback.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room.");
    } else if (reason == 1) {
        NSLog(@"Removed from the current room by the server.");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved.");
    }
}
```

Note:

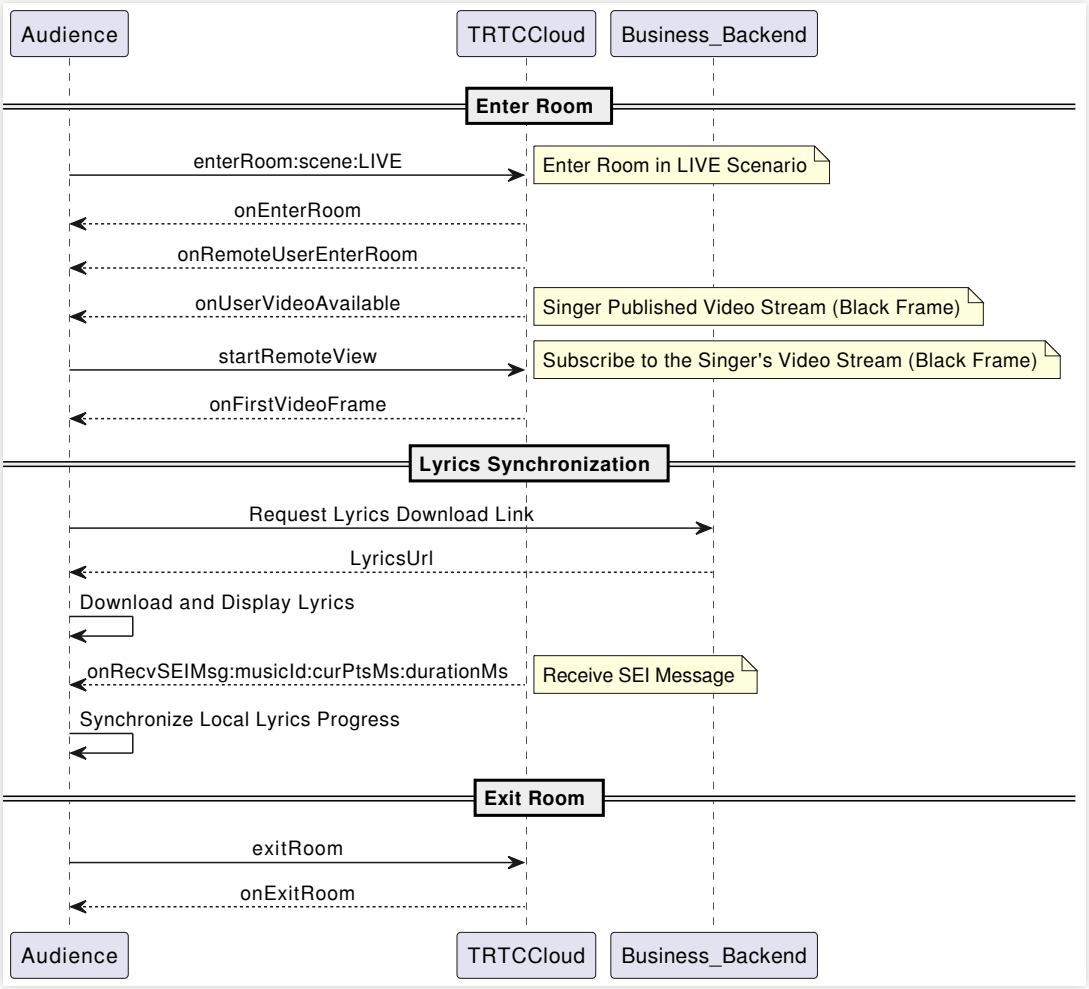
After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you want to call `enterRoom` again or switch to another audio and video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter various exceptional issues such as the camera, microphone device being forcibly occupied.

Perspective 2: Listener actions

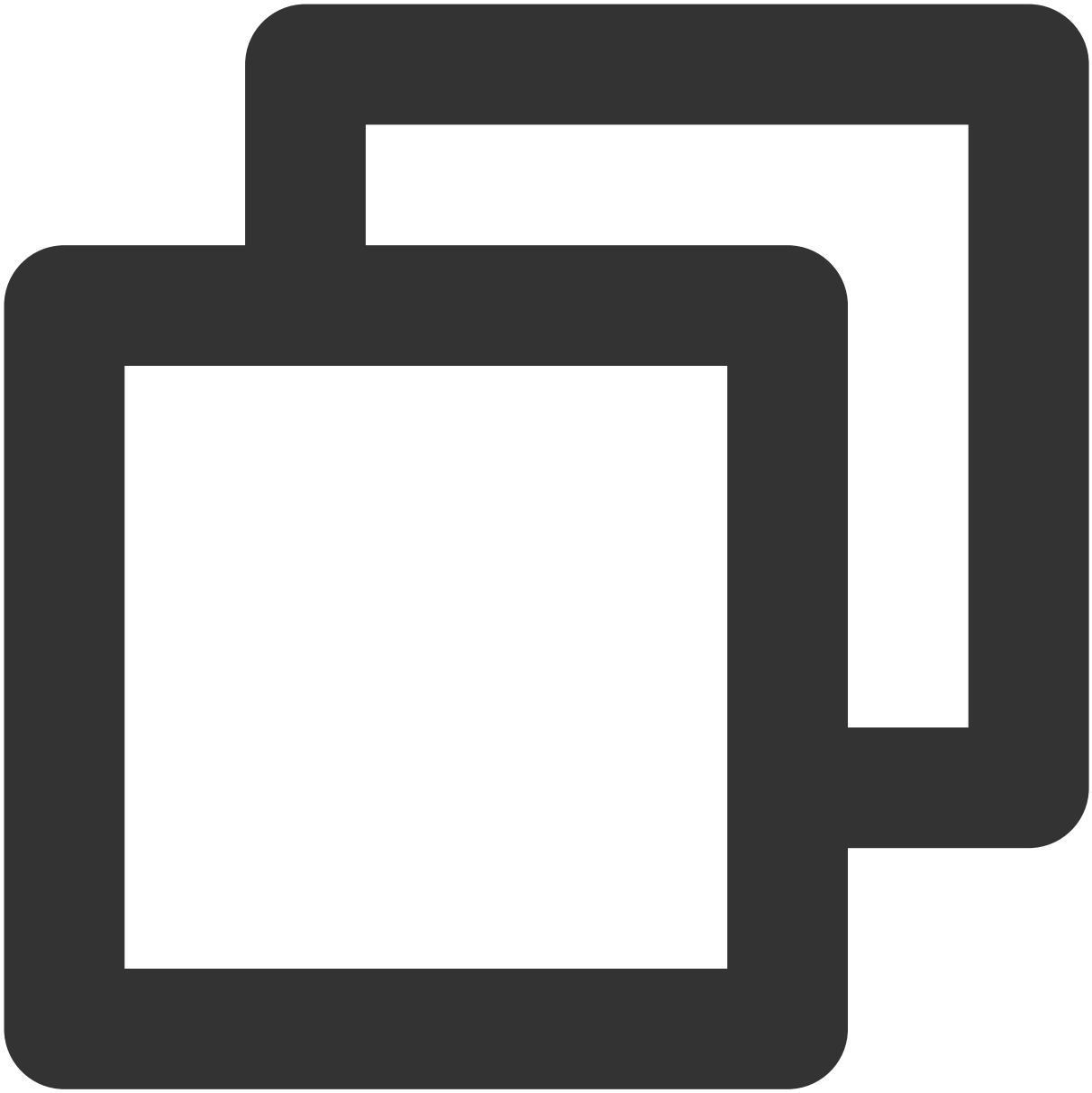
Sequence diagram

©2013-2022 Tencent Cloud. All rights reserved.



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enter the room.
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = [self generateUserSig:userId];
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.role = TRTCRoleAudience;
    // LIVE should be selected for the room entry scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

To better transmit SEI messages for lyric synchronization, it is recommended to choose `TRTCAppSceneLIVE` for room entry scenarios.

Under the automatic subscription mode (default), audiences automatically subscribe and play the on-mic anchor's audio and video streams upon entering the room.

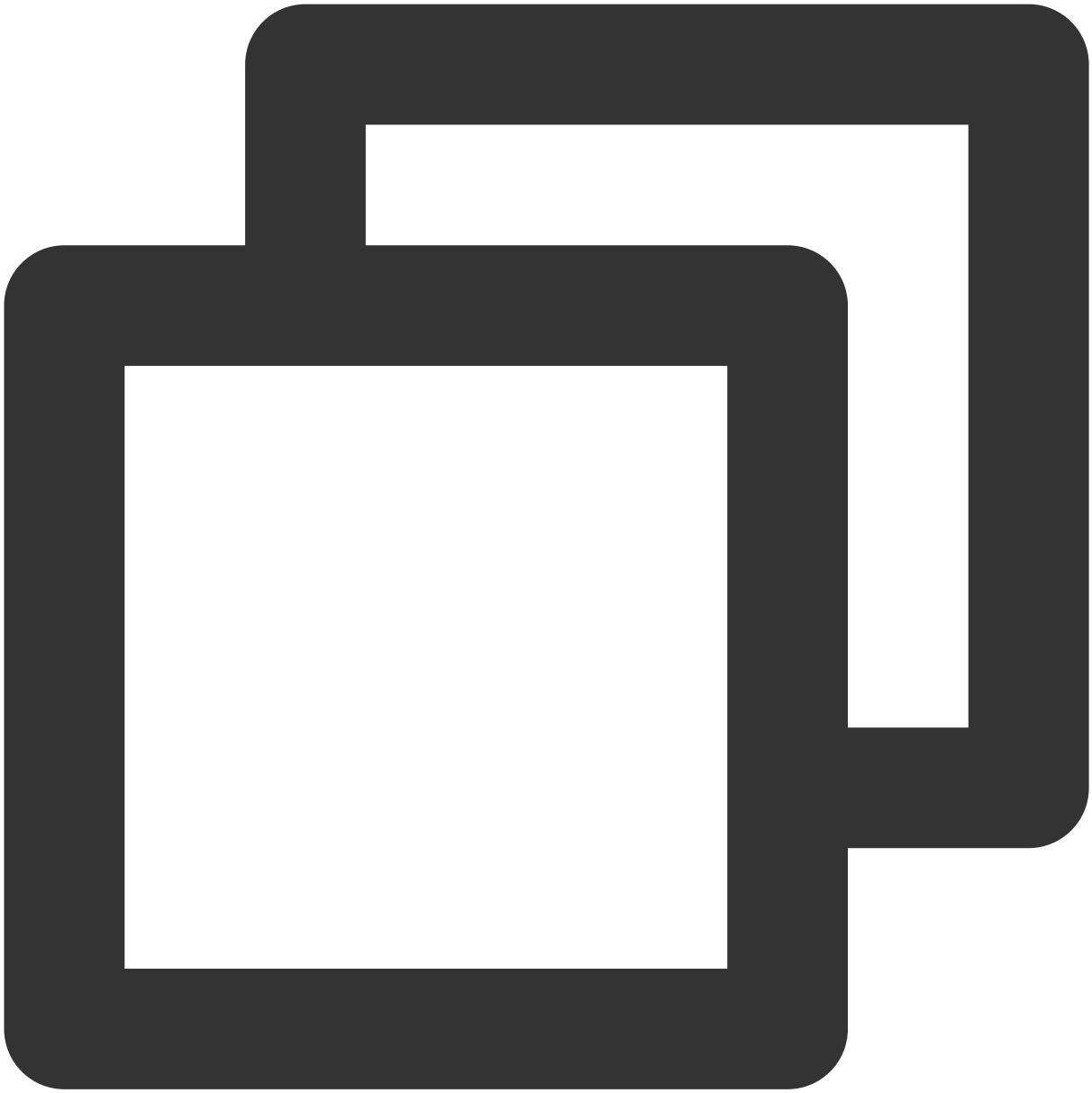
2. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Listener end lyric synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    if (available) {
        [self.trtcCloud startRemoteView:userId view:nil];
    } else {
        [self.trtcCloud stopRemoteView:userId];
    }
}

- (void)onRecvSEIMsg:(NSString *)userId message:(NSData *)message {
    JSONModel *json = [[JSONModel alloc] initWithData:message error:nil];
    NSDictionary *dic = json.toDictionary;
```

©2013-2022 Tencent Cloud. All rights reserved.

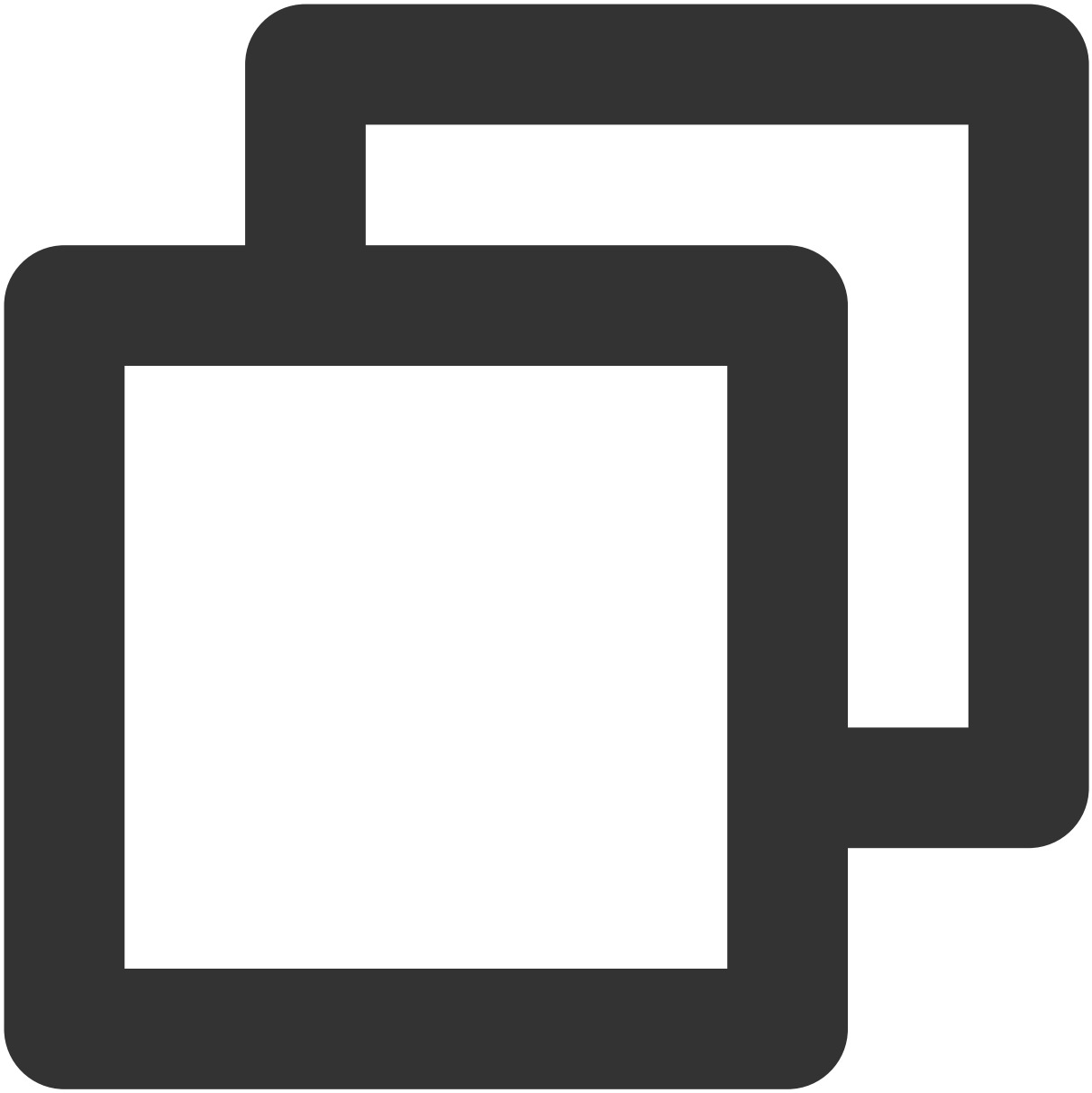
```
    int32_t musicId = [dic[@"musicId"] intValue];
    NSInteger progress = [dic[@"progress"] integerValue];
    NSInteger duration = [dic[@"duration"] integerValue];
    // ......
    // TODO: The logic of updating the lyric control.
    // Based on the received latest progress and the local lyrics progress deviatio
    // ......
}
```

Note:

Listeners need to actively subscribe to the performer's video streams in order to receive the SEI messages carried by black frames.

3. Exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Exit the room.
[self.trtcCloud exitRoom];

// Exit room event callback.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room.");
    } else if (reason == 1) {
        NSLog(@"Removed from the current room by the server.");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved.");
```

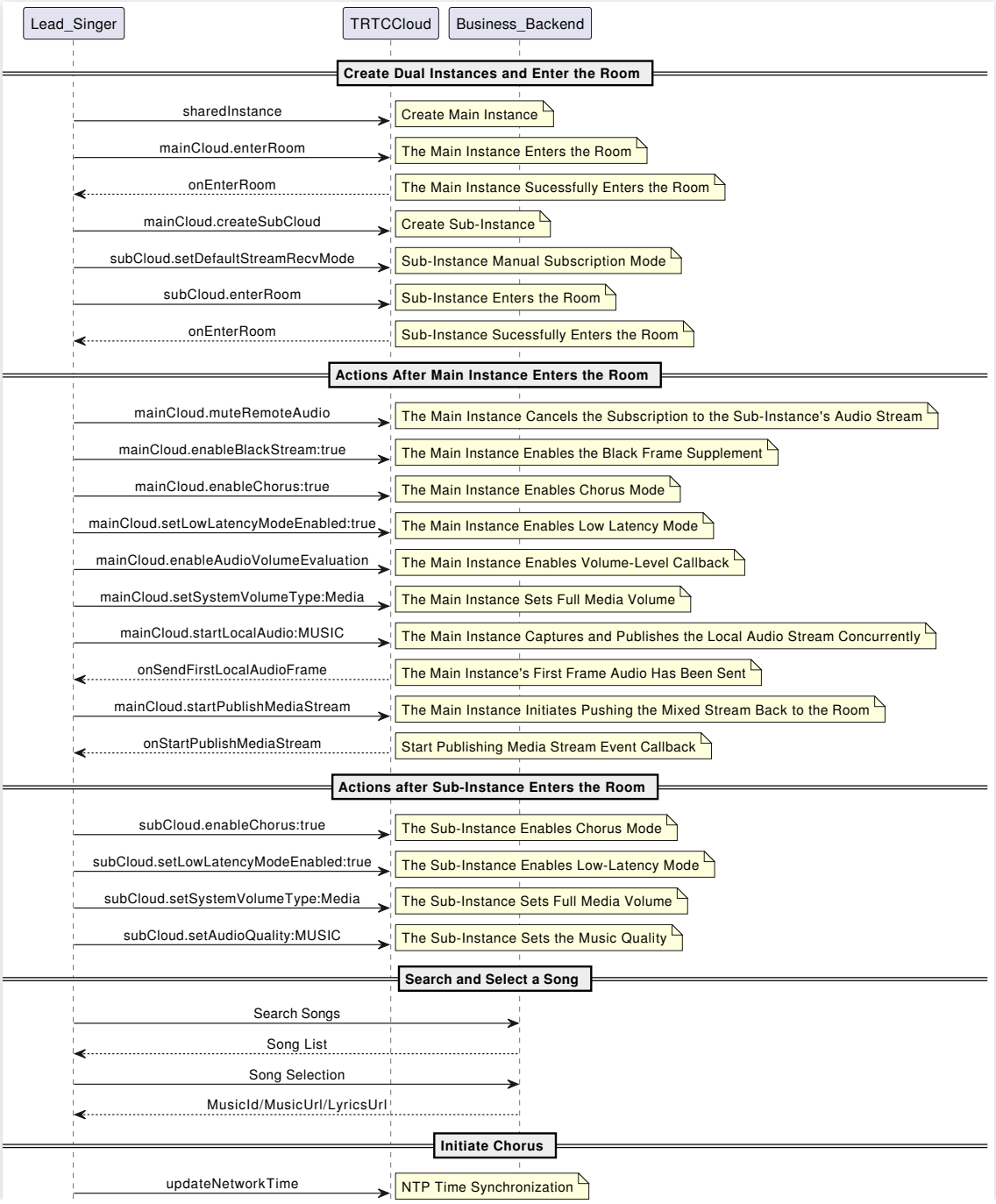
©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }
```

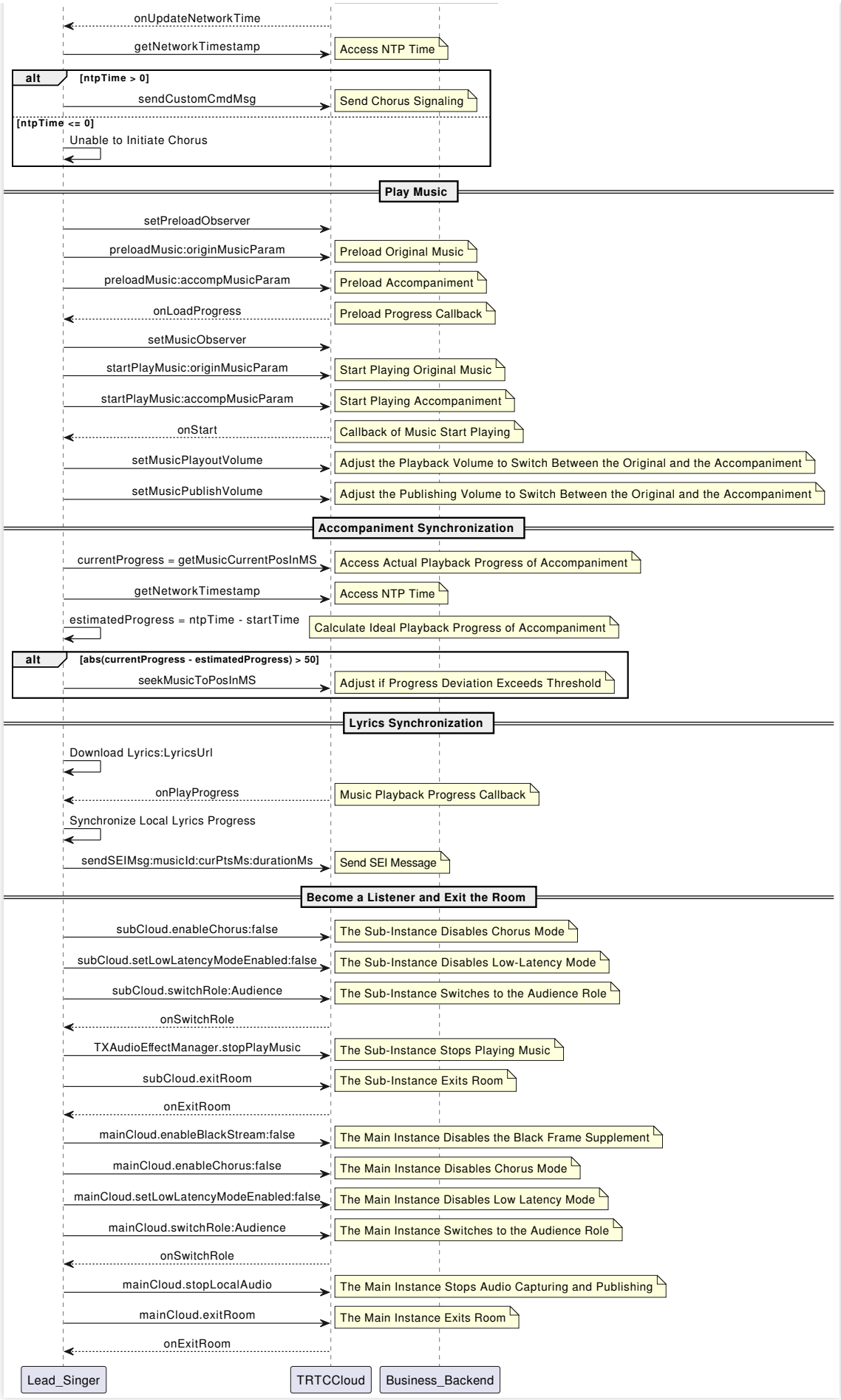
Scenario 2: Real-time chorus

Perspective 1: Lead singer actions

Sequence diagram

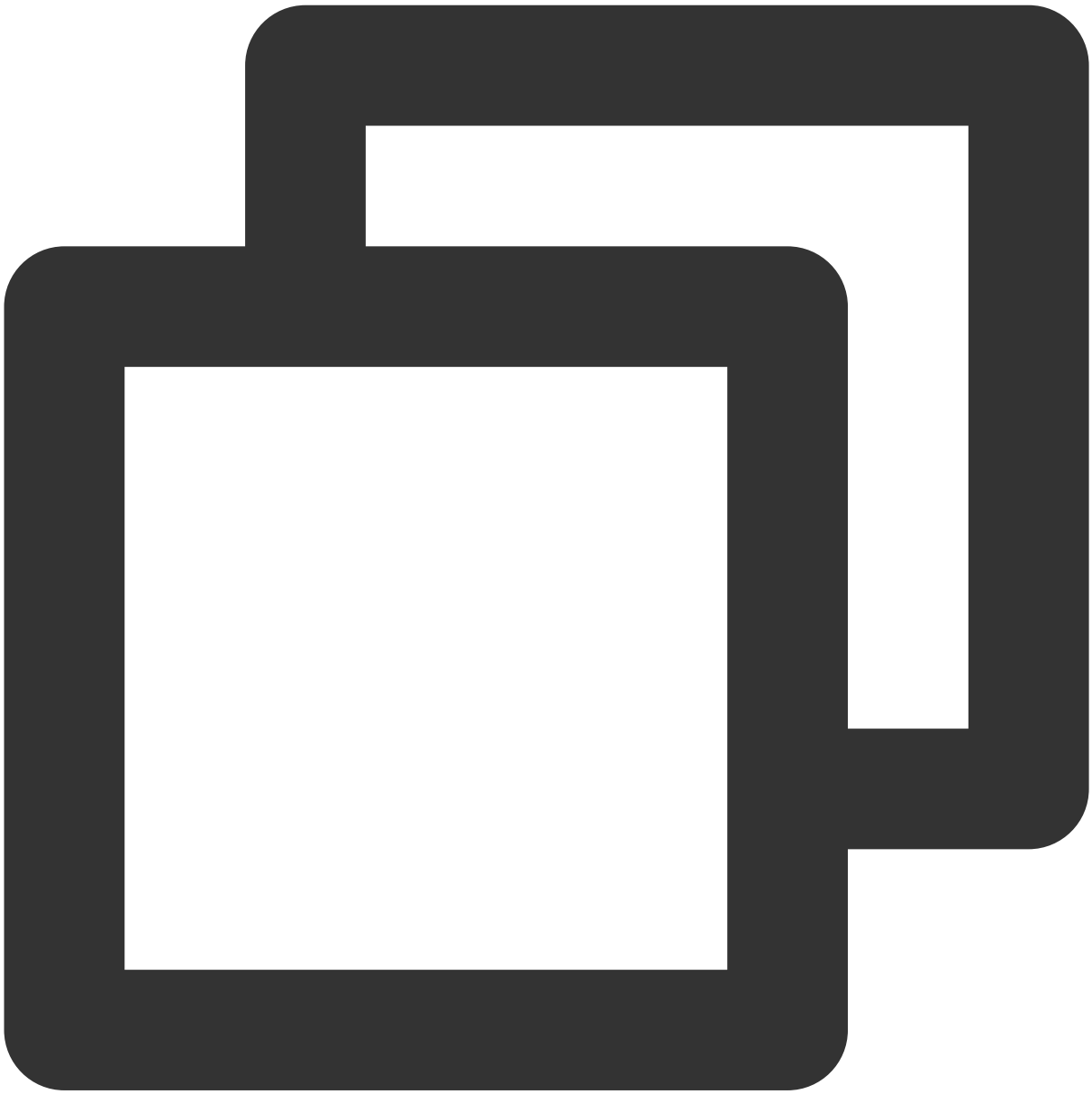


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

1. Dual instances enter the room.



```
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    // Create a TRTCCloud primary instance (vocal instance).
    TRTCCloud *mainCloud = [TRTCCloud sharedInstance];
    // Create a TRTCCloud sub-instance (music instance).
    TRTCCloud *subCloud = [mainCloud createSubCloud];

    // The primary instance (vocal instance) enters the room.
    TRTCParams *params = [[TRTCParams alloc] init];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.strRoomId = roomId;
    params.userId = userId;
    params.userSig = userSig;
    params.sdkAppId = SDKAppID;
    params.role = TRTCRoleAnchor;
    [mainCloud enterRoom:params appScene:TRTCAppSceneLIVE];

    // The sub-instance enables manual subscription mode. By default it does not su
    [subCloud setDefaultStreamRecvMode:NO video:NO];
    // The sub-instance (music instance) enters the room.
    TRTCParams *bgmParams = [[TRTCParams alloc] init];
    bgmParams.strRoomId = roomId;
    // The sub-instance username must not duplicate with other users in the room.
    bgmParams.userId = [userId stringByAppendingString:@"_bgm"];
    bgmParams.userSig = userSig;
    bgmParams.sdkAppId = SDKAppID;
    bgmParams.role = TRTCRoleAnchor;
    [subCloud enterRoom:bgmParams appScene:TRTCAppSceneLIVE];
}
```

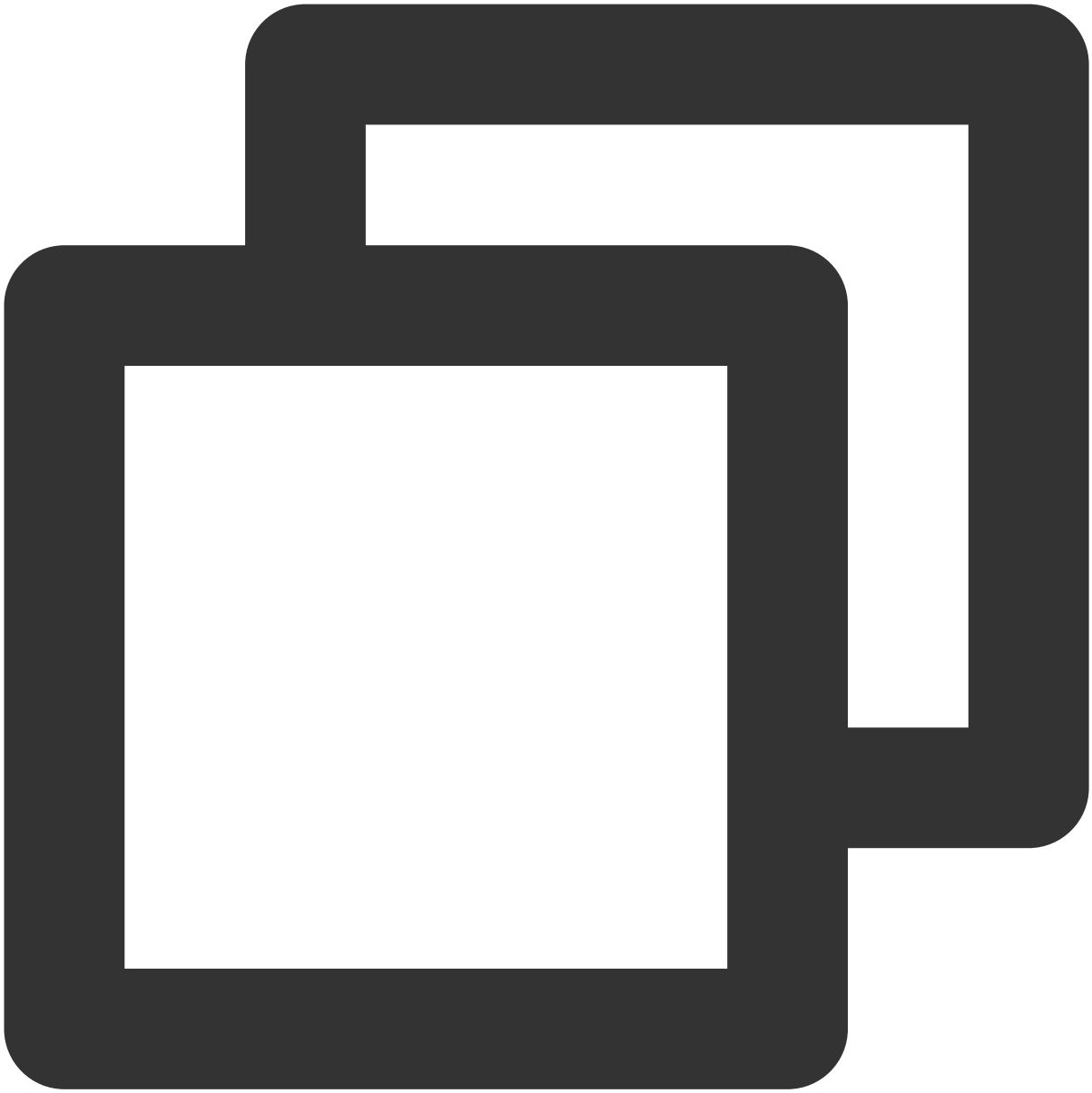
Note:

In a real-time chorus solution, the lead singer end must create primary instance and sub-instance for upstream voice and accompaniment music, respectively.

Sub-instances do not need to subscribe to other users' audio streams in the room. Therefore, it is recommended to enable manual subscription mode, and it must be activated before entering the room.

2. Set the settings after entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the result of primary instance entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The primary instance unsubscribe from music streams published by sub-ins
        [self.trtcCloud muteRemoteAudio:[self.userId stringByAppendingString:@"_bgm
        // The primary instance uses the experimental API to enable black frame ins
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableBlackStream\\",\\
        // The primary instance uses the experimental API to enable chorus mode.
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"para
        // The primary instance uses the experimental API to enable low-latency mod
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnable
```

©2013-2022 Tencent Cloud. All rights reserved.

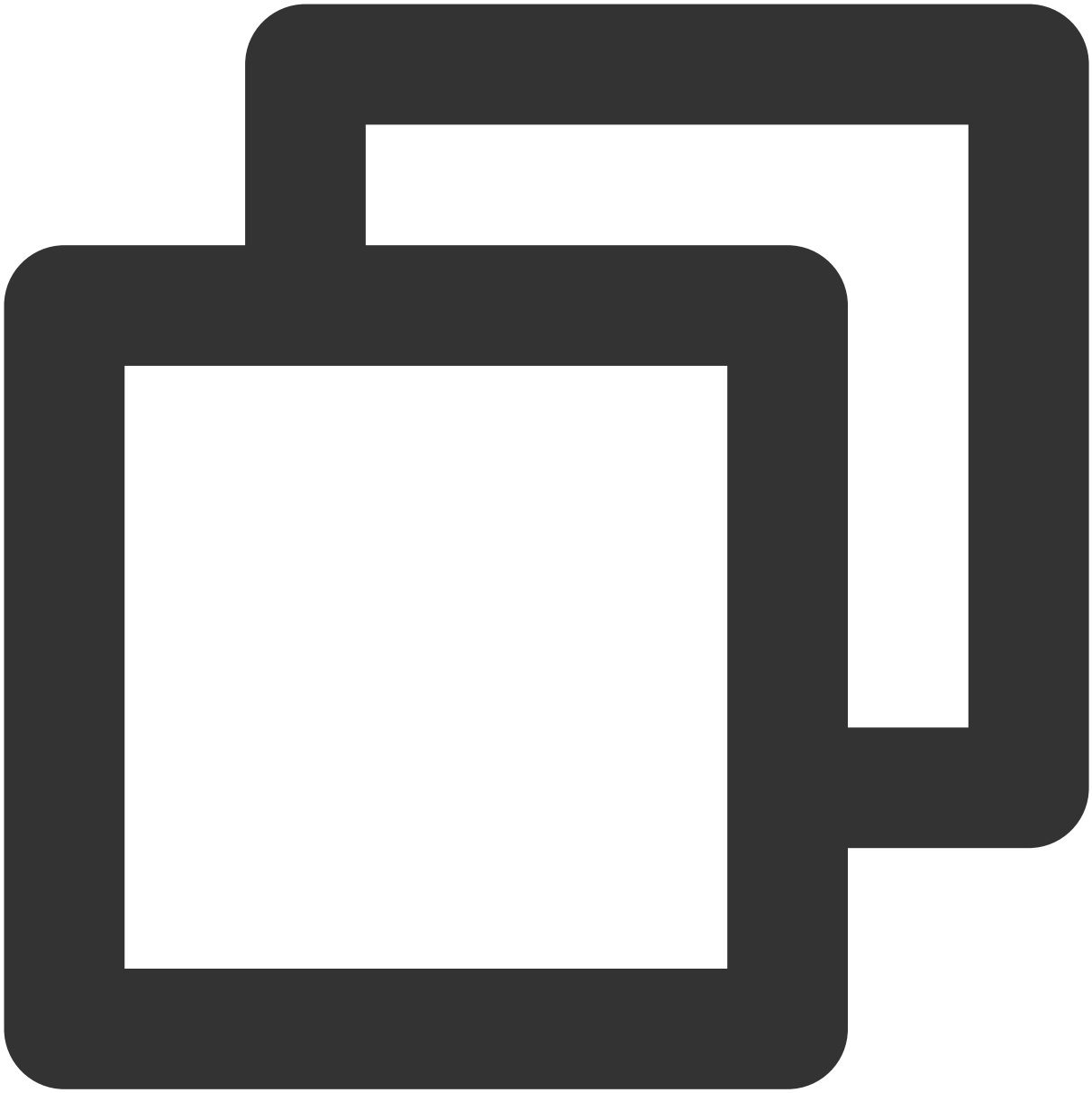
```
        // The primary instance enables volume level callback.
        TRTCAudioVolumeEvaluateParams *aveParams = [[TRTCAudioVolumeEvaluateParams
        aveParams.interval = 300;
        [self.trtcCloud enableAudioVolumeEvaluation:YES withParams:aveParams];
        // The primary instance sets the global media volume type.
        [self.trtcCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
        // The primary instance captures and publishes local audio, and sets audio
        [self.trtcCloud startLocalAudio:TRTCAudioQualityMusic];
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed");
    }
}

// Event callback for the result of sub-instance entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The sub-instance uses the experimental API to enable chorus mode.
        [self.subCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"param
        // The sub-instance uses the experimental API to enable low-latency mode.
        [self.subCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnabled
        // The sub-instance sets global media volume type.
        [self.subCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
        // The sub-instance sets audio quality.
        [self.subCloud setAudioQuality:TRTCAudioQualityMusic];
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed");
    }
}
```

Note:

Both the primary instance and sub-instance must use the experimental APIs to enable chorus mode and low-latency mode to optimize the chorus experience. Note the difference in the `audioSource` parameter.

3. Push the mixed stream back to the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)startPublishMediaToRoomWithRoomId:(NSString *)roomId userId:(NSString *)use
    // Create TRTCPublishTarget object.
    TRTCPublishTarget *target = [[TRTCPublishTarget alloc] init];
    // After mixing, the stream is relayed back to the room.
    target.mode = TRTCPublishMixStreamToRoom;
    TRTCUser *mixStreamIdentity = [[TRTCUser alloc] init];
    mixStreamIdentity.strRoomId = roomId;
    // The mixing stream robot's username must not duplicate with other users in th
    mixStreamIdentity.userId = [userId stringByAppendingString:@"_robot"];
    target.mixStreamIdentity = mixStreamIdentity;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Set the encoding parameters of the transcoded audio stream (can be customize
    TRTCStreamEncoderParam *encoderParam = [[TRTCStreamEncoderParam alloc] init];
    encoderParam.audioEncodedChannelNum = 2;
    encoderParam.audioEncodedKbps = 64;
    encoderParam.audioEncodedCodecType = 2;
    encoderParam.audioEncodedSampleRate = 48000;

    // Set the encoding parameters of the transcoded video stream (black frame mixi
    encoderParam.videoEncodedFPS = 15;
    encoderParam.videoEncodedGOP = 3;
    encoderParam.videoEncodedKbps = 30;
    encoderParam.videoEncodedWidth = 64;
    encoderParam.videoEncodedHeight = 64;

    // Set audio mixing parameters.
    TRTCStreamMixingConfig *mixingConfig = [[TRTCStreamMixingConfig alloc] init];
    // By default, leave this field empty. It indicates that all audio in the room
    mixingConfig.audioMixUserList = nil;

    // Configure video mixed-stream template (black frame mixing required).
    TRTCVideoLayout *layout = [[TRTCVideoLayout alloc] init];
    mixingConfig.videoLayoutList = @[layout];

    // Start mixing and pushing back.
    [self.trtcCloud startPublishMediaStream:target encoderParam:encoderParam mixing
}
```

Note:

To maintain alignment between chorus vocals and accompaniment music, it is recommended to enable pushing the mixed stream back to the room. The on-mic chorus members mutually subscribe to single streams, and off-mic audiences by default only subscribe to mixed streams.

The mixing stream robot, acting as an independent user, enters the room to pull, mix, and push streams. Its username must not duplicate with other usernames in the room. Otherwise, it may lead to mutual deletion from the room.

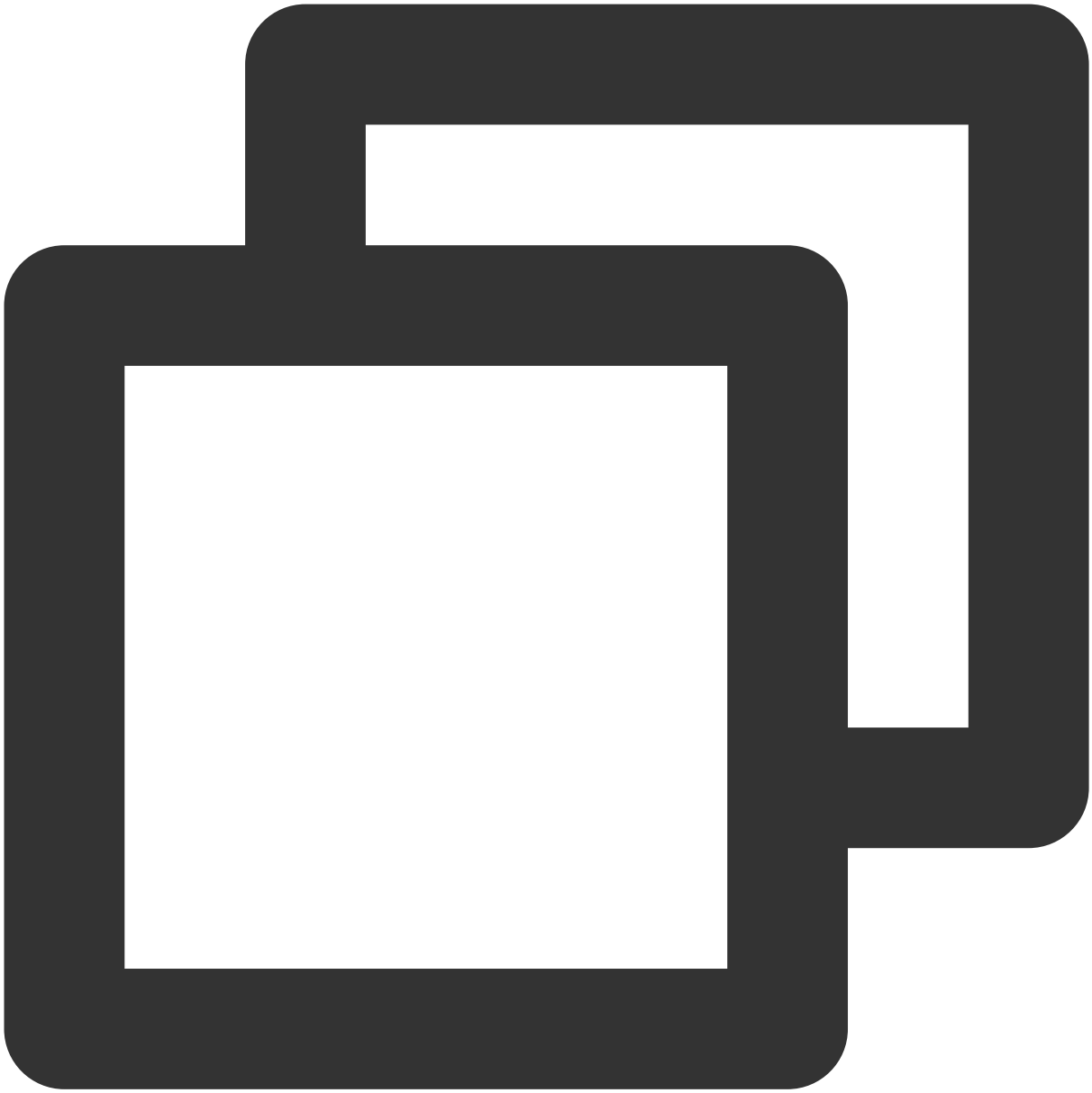
4. Search for and request songs.

Search for songs and acquire music resources through the business backend. Obtain identifiers such as the MusicId, the song's URL (MusicUrl), and the lyrics URL (LyricsUrl).

It is recommended that the business side select an appropriate music repository production to provide licensed music resources.

5. NTP synchronization.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)updateNetworkTimeExample {
    [TXLiveBase sharedInstance].delegate = self;
    [TXLiveBase updateNetworkTime];
}

- (void)onUpdateNetworkTime:(int)errCode message:(NSString *)errMsg {
    // errCode 0: Time synchronization successful and deviation within 30 ms. 1: Ti
    if (errCode == 0) {
        // Time synchronization successful and NTP timestamp obtained.
        NSInteger ntpTime = [TXLiveBase getNetworkTimestamp];
    } else {
```

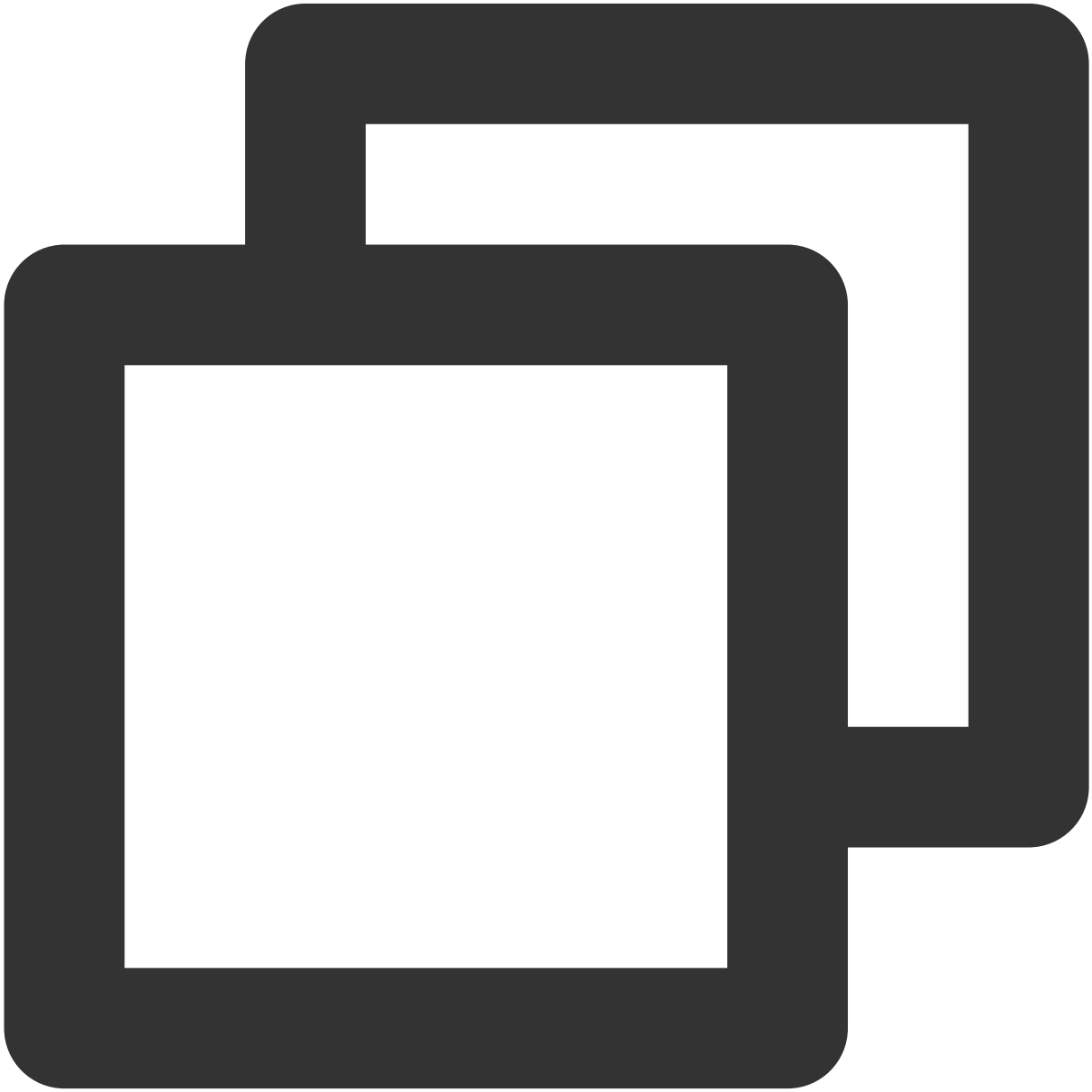
©2013-2022 Tencent Cloud. All rights reserved.

```
        NSLog(@"Time synchronization failed, and you can try re-synchronization.");
    }
}
```

Note:

NTP time synchronization results can reflect the current network quality of the application user. To ensure a good chorus experience, it is recommended not to allow users to initiate chorus if time synchronization fails.

6. Send chorus signaling.



```
- (void)sendChorusSignalExample {
    __weak typeof(self) weakSelf = self;
```

©2013-2022 Tencent Cloud. All rights reserved.

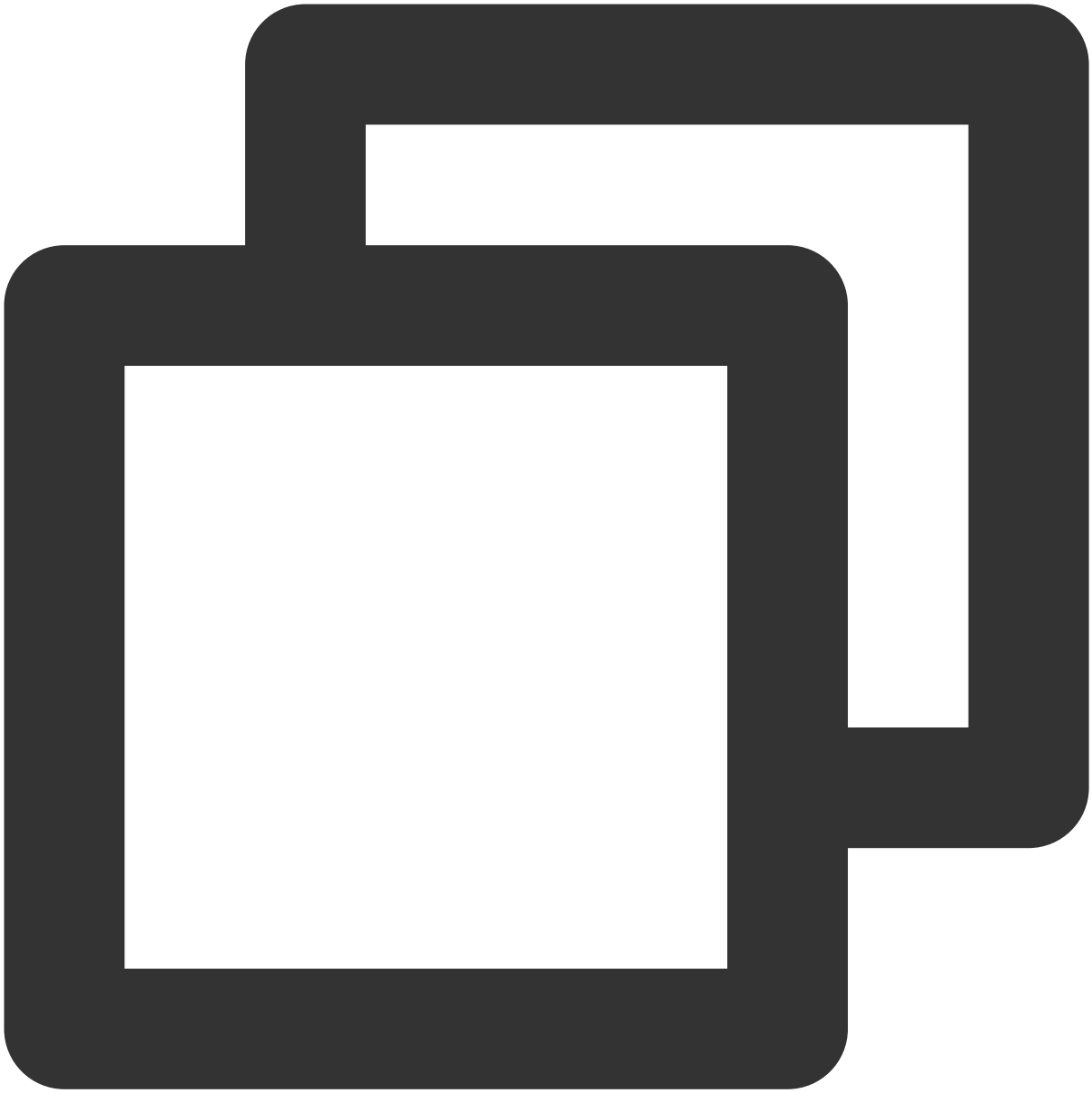
```
    NSTimer *timer = [NSTimer timerWithTimeInterval:1.0 repeats:YES block:^(NSTimer
        __strong typeof(weakSelf) strongSelf = weakSelf;
        NSDictionary *dic = @{
            @"cmd": @"startChorus",
            // Agreed chorus start time: Current NTP time + delayed playback time (
            @"startPlayMusicTS": @([TXLiveBase getNetworkTimestamp] + 3000),
            @"musicId": @(self.musicId),
            @"musicDuration": @([[strongSelf.subCloud getAudioEffectManager] getMus
        };
        JSONModel *json = [[JSONModel alloc] initWithDictionary:dic error:nil];
        [strongSelf.trtcCloud sendCustomCmdMsg:1 data:json.toJSONData reliable:NO o
    }];
    [[NSRunLoop currentRunLoop] addTimer:timer forMode:NSRunLoopCommonModes];
}
```

Note:

The lead singer needs to cyclically broadcast chorus signaling to the room at a fixed time interval (e.g., every 1 second), so that new users who join mid-session can also participate in the chorus.

7. Load and play accompaniment.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain audio effects management.
TXAudioEffectManager *audioEffectManager = [self.subCloud getAudioEffectManager];

// originMusicId: Custom identifier for the original vocal music. originMusicUrl: U
TXAudioMusicParam *originMusicParam = [[TXAudioMusicParam alloc] init];
originMusicParam.ID = originMusicId;
originMusicParam.path = originMusicUrl;
// Whether to publish the original vocal music to remote (otherwise play locally on
originMusicParam.publish = YES;
// Music start playing time point (in milliseconds).
originMusicParam.startTimeMS = 0;
```

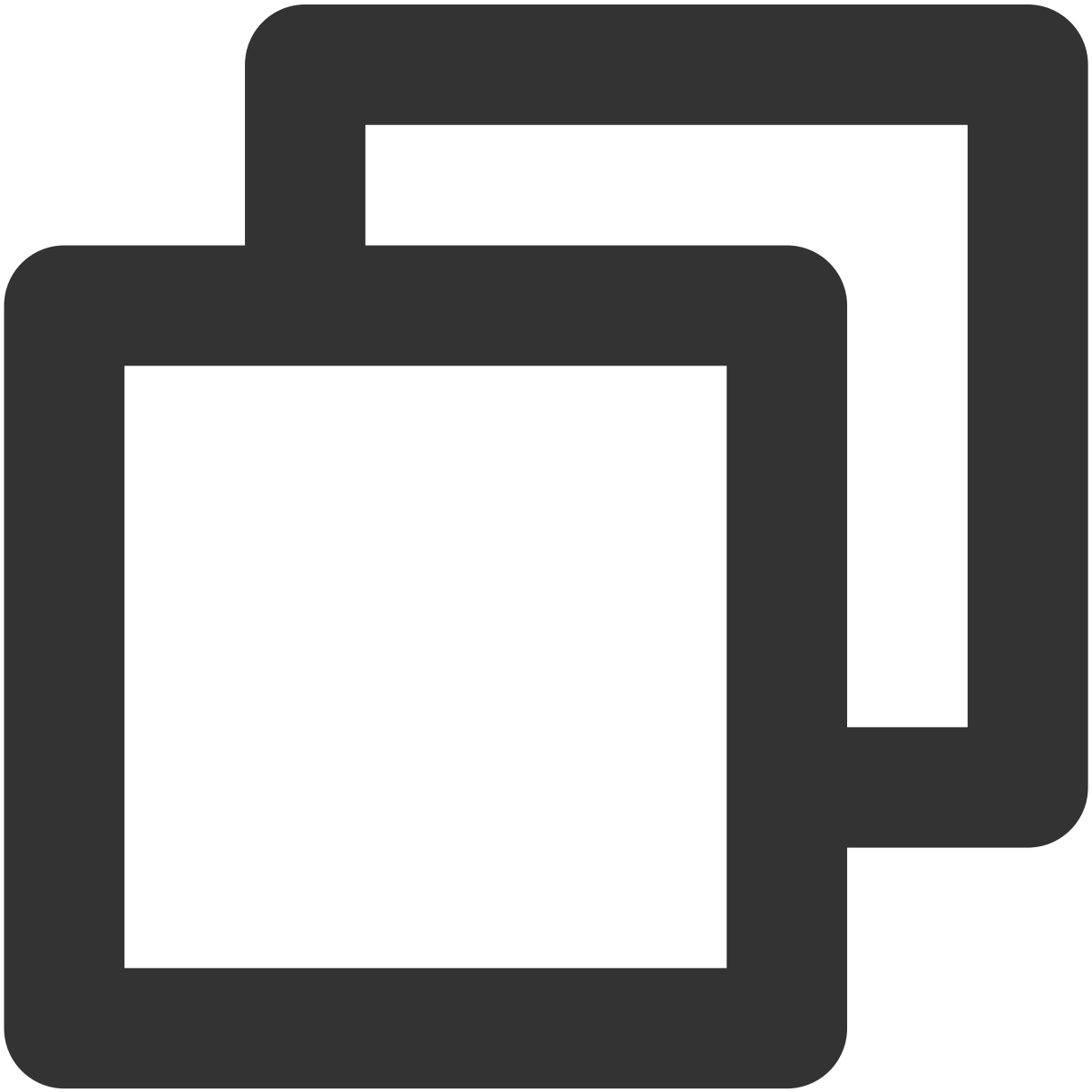
©2013-2022 Tencent Cloud. All rights reserved.

```
// accompMusicId: Custom identifier for the accompaniment music. accompMusicUrl: UR
TXAudioMusicParam *accompMusicParam = [[TXAudioMusicParam alloc] init];
accompMusicParam.ID = accompMusicId;
accompMusicParam.path = accompMusicUrl;
// Whether to publish the accompaniment to remote (otherwise play locally only).
accompMusicParam.publish = YES;
// Music start playing time point (in milliseconds).
accompMusicParam.startTimeMS = 0;

// Preload the original vocal music.
[audioEffectManager preloadMusic:originMusicParam onProgress:nil onError:nil];
// Preload the accompaniment music.
[audioEffectManager preloadMusic:accompMusicParam onProgress:nil onError:nil];

// Start playing the original vocal music after a delayed playback time (for exampl
[self.audioEffectManager startPlayMusic:originMusicParam onStart:^(NSInteger errCod
    // onStart
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // onProgress
} onComplete:^(NSInteger errCode) {
    // onComplete
}];

// Start playing the accompaniment music after a delayed playback time (for example
[self.audioEffectManager startPlayMusic:originMusicParam onStart:^(NSInteger errCod
    // onStart
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // onProgress
} onComplete:^(NSInteger errCode) {
    // onComplete
}];

// Switch to the original vocal music.
[self.audioEffectManager setMusicPlayoutVolume:originMusicId volume:100];
[self.audioEffectManager setMusicPublishVolume:originMusicId volume:100];
[self.audioEffectManager setMusicPlayoutVolume:accompMusicId volume:0];
[self.audioEffectManager setMusicPublishVolume:accompMusicId volume:0];

// Switch to the accompaniment music.
[self.audioEffectManager setMusicPlayoutVolume:originMusicId volume:0];
[self.audioEffectManager setMusicPublishVolume:originMusicId volume:0];
[self.audioEffectManager setMusicPlayoutVolume:accompMusicId volume:100];
[self.audioEffectManager setMusicPublishVolume:accompMusicId volume:100];
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

It is recommended to preload music before starting playback. By loading music resources into memory in advance, you can effectively reduce the load delay of music playback.
In karaoke scenarios, both the original vocal and accompaniment need to be played simultaneously (distinguished by MusicID). The switch between the original vocal and accompaniment is achieved by adjusting the local and remote playback volumes.
If the music being played has dual audio tracks (including both the original vocal and accompaniment), switching between them can be achieved by specifying the music's playback track using setMusicTrack.
8. Accompaniment Synchronization



```
// Agreed chorus start time.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
@property (nonatomic, assign) NSInteger startPlayMusicTS;

- (void)syncBgmExample {
    // Actual playback progress of the current accompaniment music.
    NSInteger currentProgress = [[self.subCloud getAudioEffectManager] getMusicCurr
    // Ideal playback progress of the current accompaniment music.
    NSInteger estimatedProgress = [TXLiveBase getNetworkTimestamp] - self.startPlay
    // When the progress difference exceeds 50 ms, corrections are made.
    if (estimatedProgress >= 0 && labs(currentProgress - estimatedProgress) > 50) {
        [[self.subCloud getAudioEffectManager] seekMusicToPosInMS:self.musicId pts:
    }
}
```

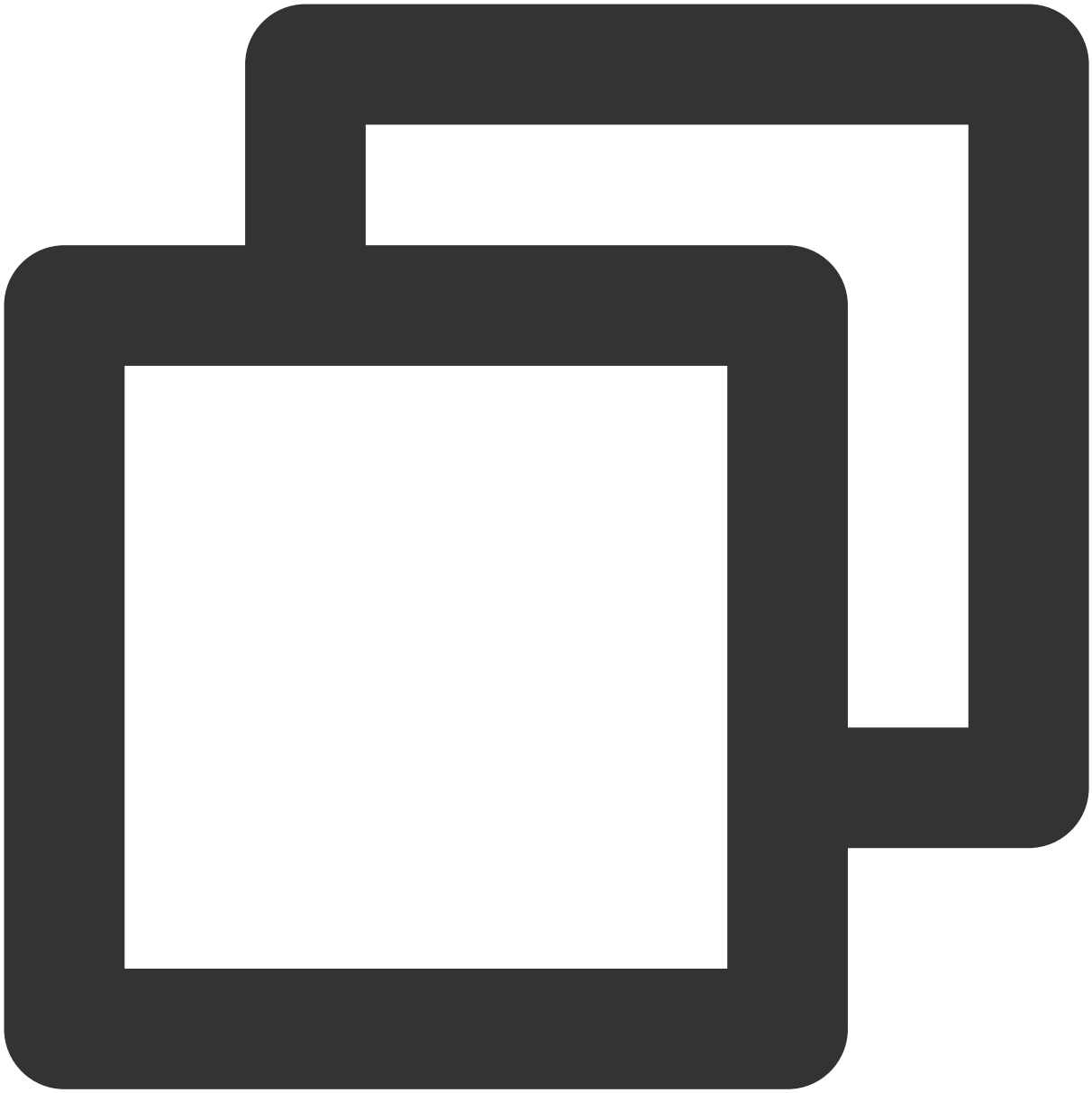
9. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Synchronize local lyrics, and transmit song progress via SEI.

©2013-2022 Tencent Cloud. All rights reserved.



```
[[self.subCloud getAudioEffectManager] startPlayMusic:musicParam onStart:^(NSIntege
    // Start playing music.
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
    // Determine whether seek is needed based on the latest progress and the local
    // Song progress is transmitted by sending an SEI message.
    NSDictionary *dic = @{
        @"musicId": @(self.musicId),
        @"progress": @(progressMs),
        @"duration": @(durationMs),
    };
    JSONModel *json = [[JSONModel alloc] initWithDictionary:dic error:nil];
```

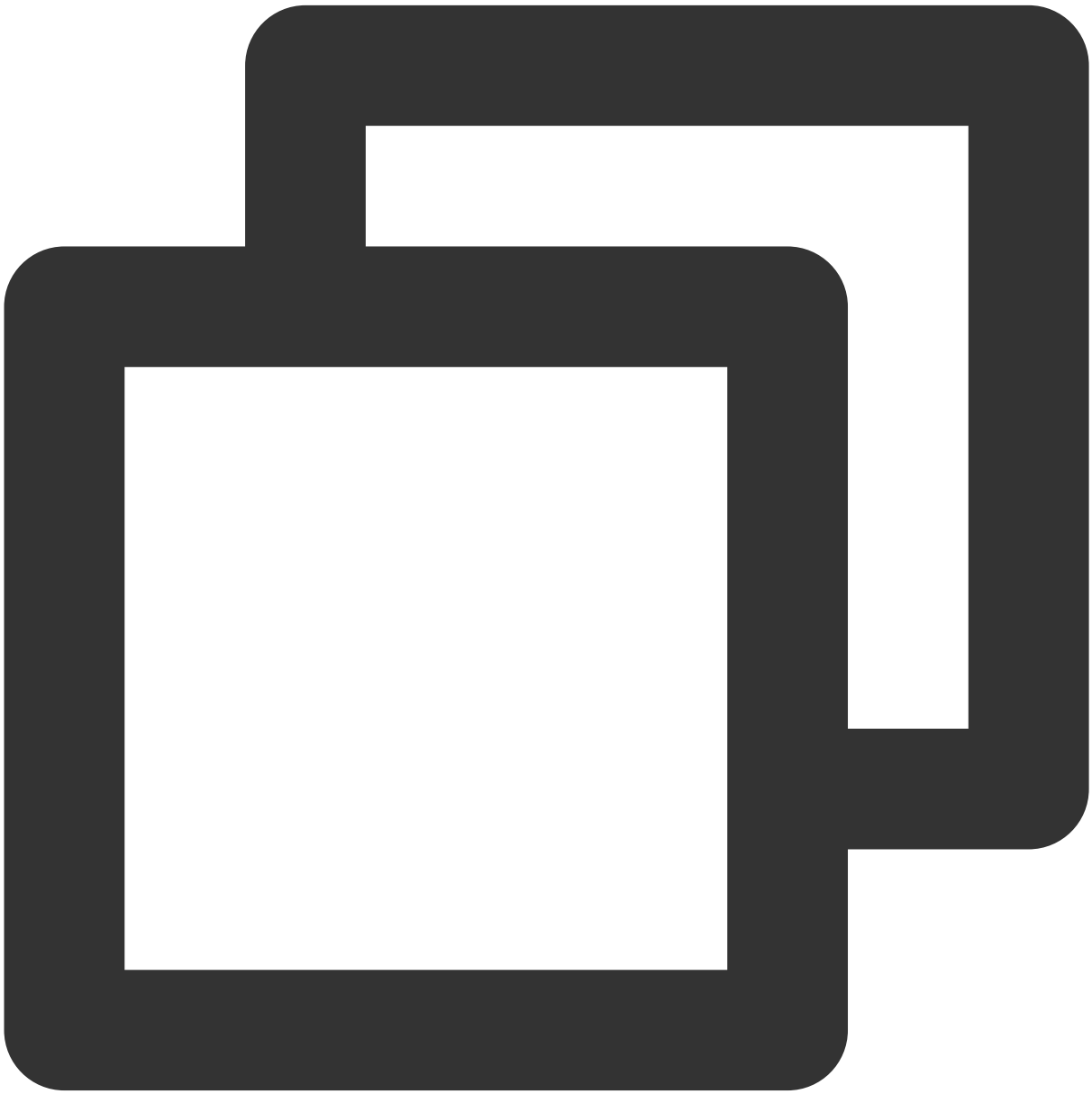
©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud sendSEIMsg:json.toJSONData repeatCount:1];
} onComplete:^(NSInteger errCode) {
    // Music playback completed.
}];
```

Note:

The frequency of the SEI messages sent by the performer is determined by the event callback frequency. Also, the playback progress can be actively synchronized on a schedule through getMusicCurrentPosInMS.

10. Become a listener and exit the room.



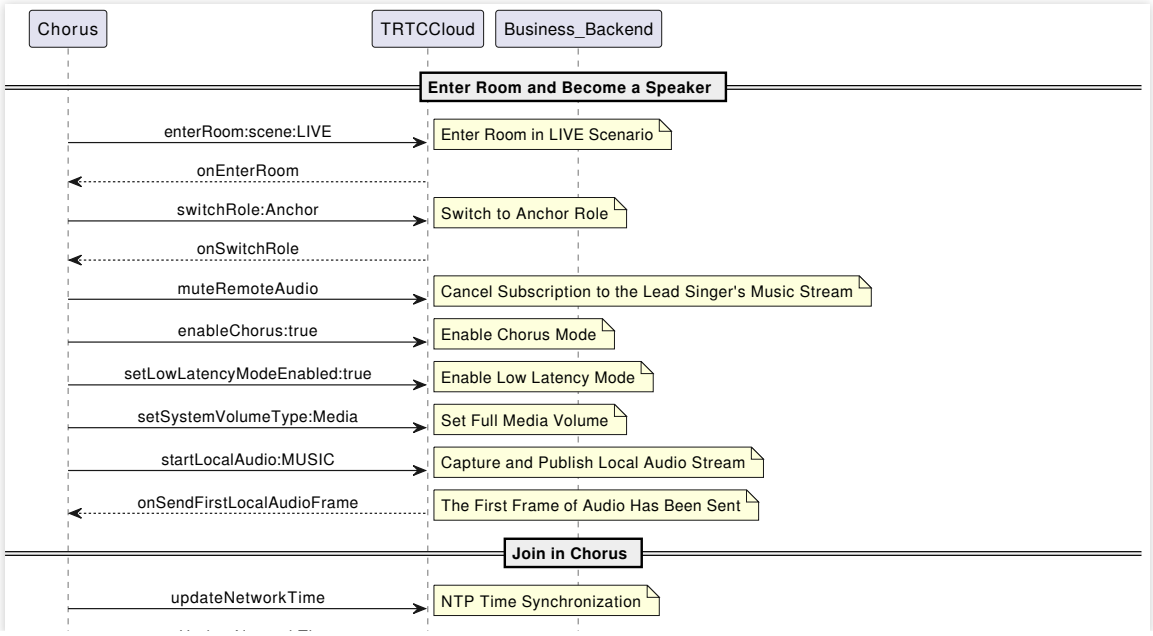
```
- (void)exitRoomExample {
```

©2013-2022 Tencent Cloud. All rights reserved.

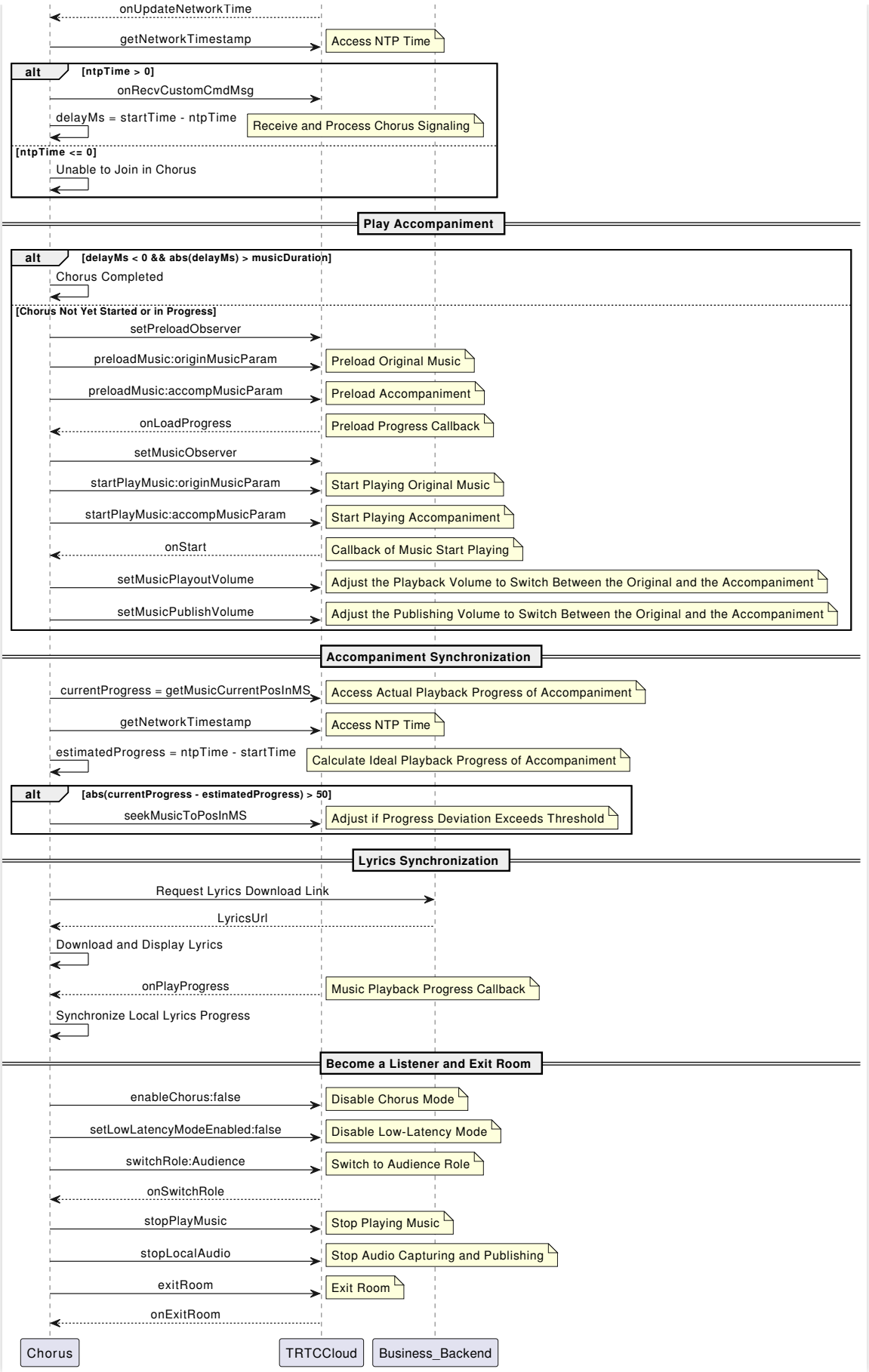
```
    // The sub-instance uses the experimental API to disable chorus mode.
    [self.subCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"params\\"
    // The sub-instance uses the experimental API to disable low-latency mode.
    [self.subCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnabled\\",
    // The sub-instance switches to the audience role.
    [self.subCloud switchRole:TRTCRoleAudience];
    // The sub-instance stops playing accompaniment music.
    [[self.subCloud getAudioEffectManager] stopPlayMusic:self.musicId];
    // The sub-instance exits the room.
    [self.subCloud exitRoom];

    // The primary instance uses the experimental API to disable black frame insert
    [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableBlackStream\\",\\"par
    // The primary instance uses the experimental API to disable chorus mode.
    [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"params\\
    // The primary instance uses the experimental API to disable low-latency mode.
    [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnabled\\"
    // The primary instance switches to the audience role.
    [self.trtcCloud switchRole:TRTCRoleAudience];
    // The primary instance stops local audio capture and publishing.
    [self.trtcCloud stopLocalAudio];
    // The primary instance exits the room.
    [self.trtcCloud exitRoom];
}
```

Perspective 2: Chorus actions

Sequence diagram

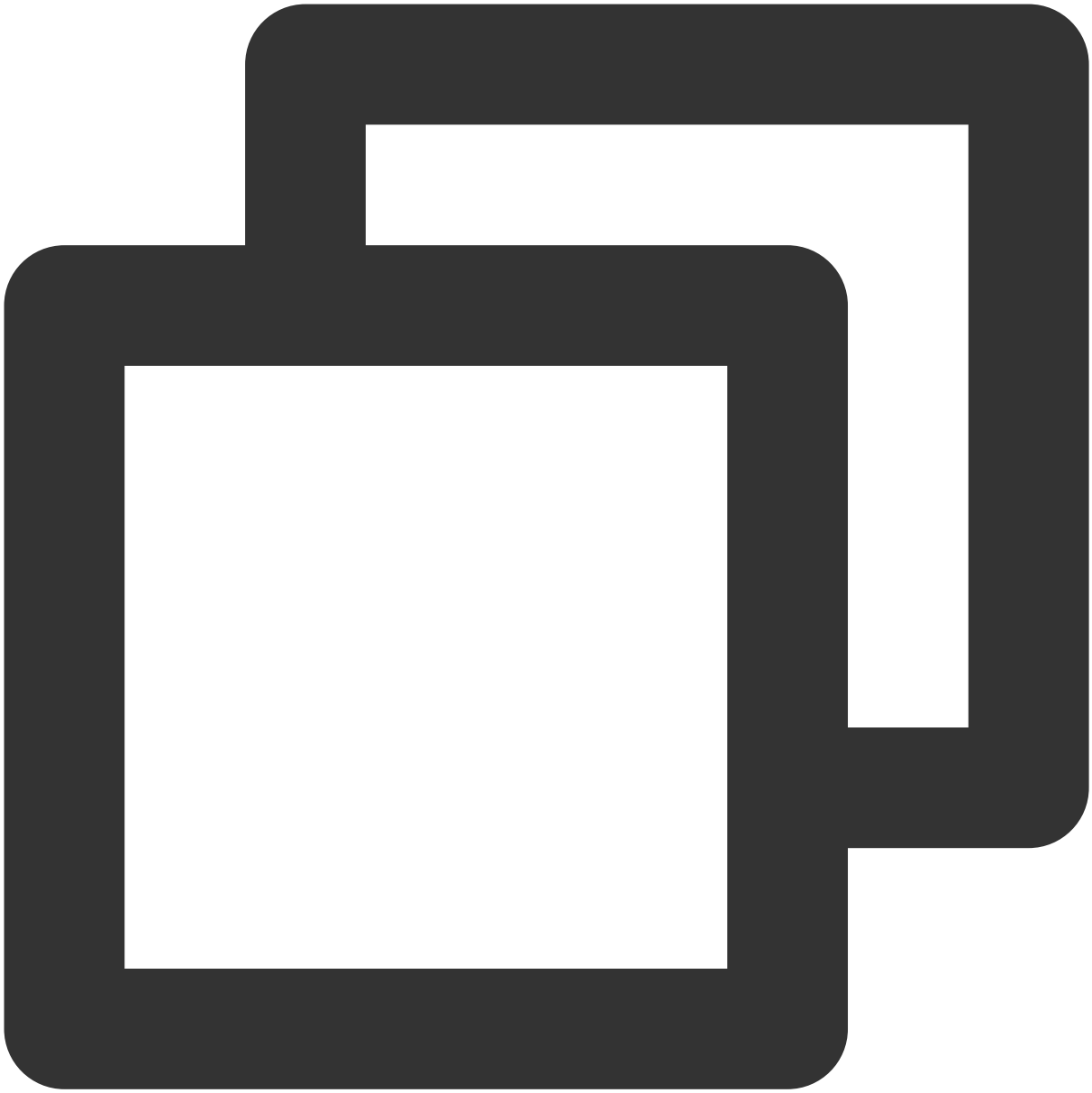


©2013-2022 Tencent Cloud. All rights reserved.



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



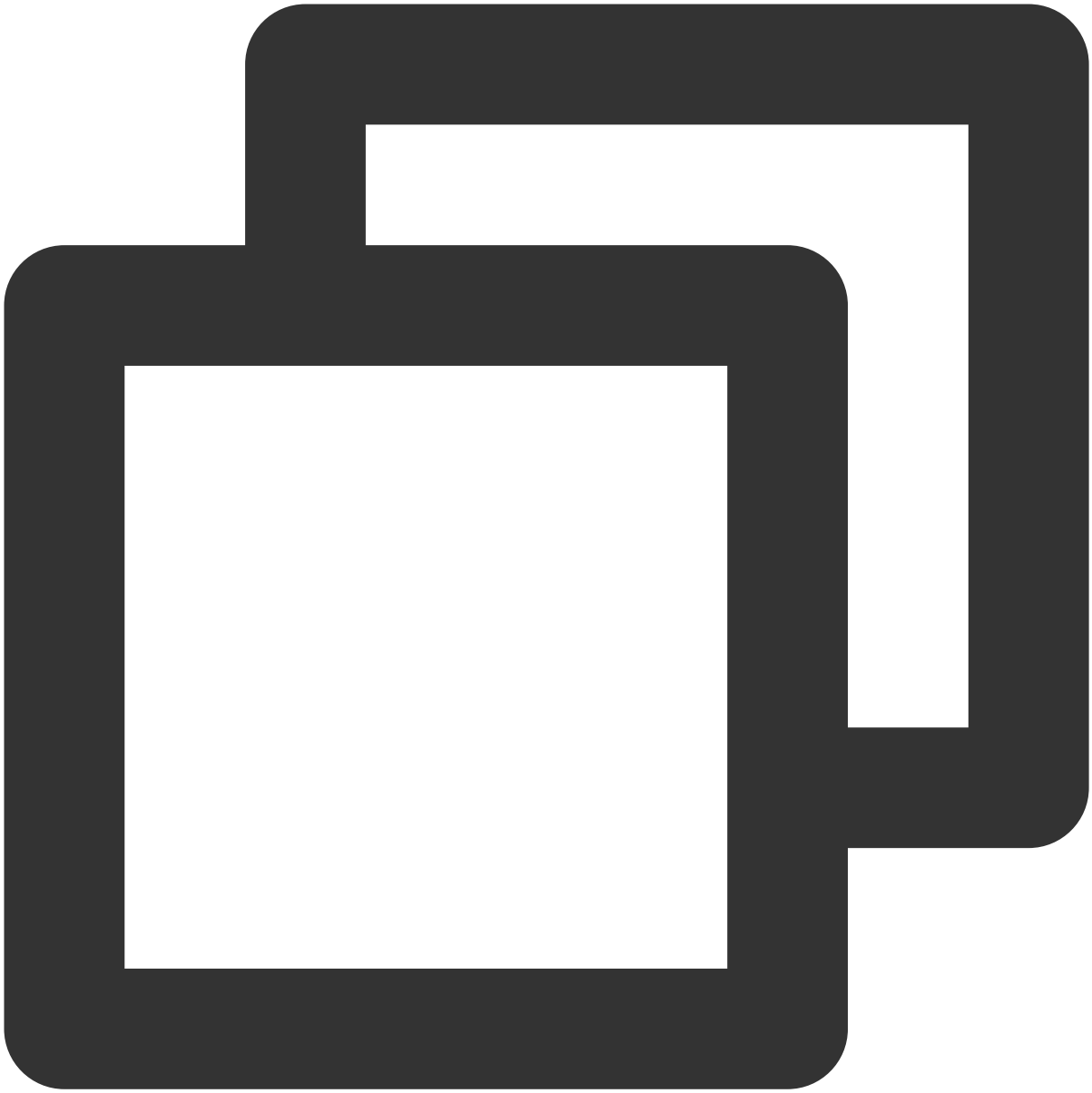
```
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = [self generateUserSig:userId];
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Example of entering the room as an audience role.
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

2. Go live on streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switched to the anchor role.
[self.trtcCloud switchRole:TRTCRoleAnchor];

// Event callback for switching the role.
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Cancel subscription to music streams published by the lead singer sub-in
        [self.trtcCloud muteRemoteAudio:self.bgmUserId mute:YES];
        // Use the experimental API to enable chorus mode.
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"para
        // Use the experimental API to enable low-latency mode.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnable
        // Set media volume type.
        [self.trtcCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
        // Upstream local audio streams and set audio quality.
        [self.trtcCloud startLocalAudio:TRTCAudioQualityMusic];
    }
}
```

Note:

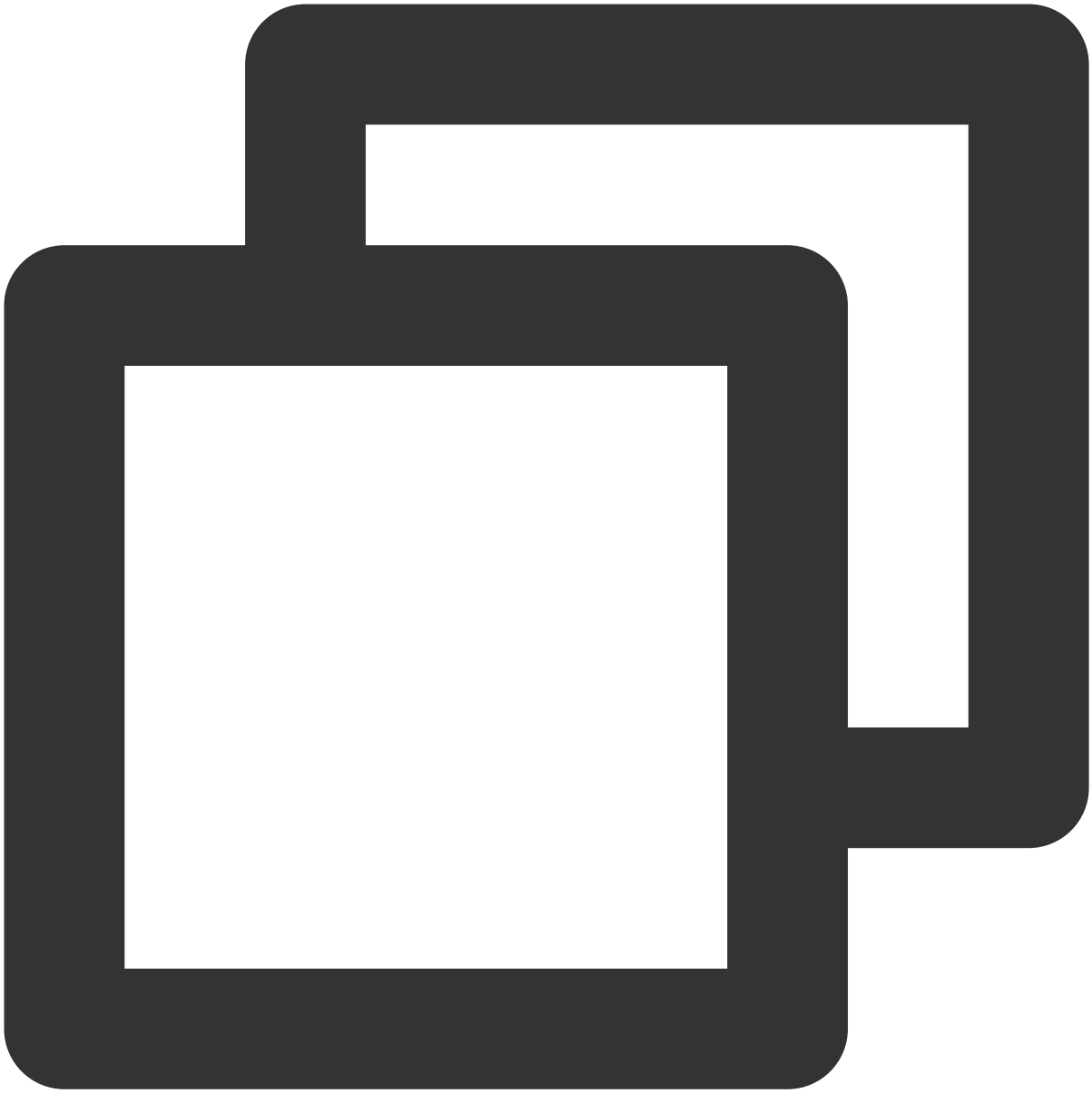
To minimize delay, all chorus members play the accompaniment music locally. Therefore, it is necessary to cancel subscriptions to music streams published by the lead singer.

Chorus members also need to use the experimental API to enable chorus mode and low-latency mode to optimize the chorus experience.

In karaoke scenarios, it is recommended to set the full-range media volume and music quality to achieve a high-fidelity listening experience.

3. NTP synchronization.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)updateNetworkTimeExample {
    [TXLiveBase sharedInstance].delegate = self;
    [TXLiveBase updateNetworkTime];
}

- (void)onUpdateNetworkTime:(int)errCode message:(NSString *)errMsg {
    // errCode 0: Time synchronization successful and deviation within 30 ms. 1: Ti
    if (errCode == 0) {
        // Time synchronization successful and NTP timestamp obtained.
        NSInteger ntpTime = [TXLiveBase getNetworkTimestamp];
    } else {
```

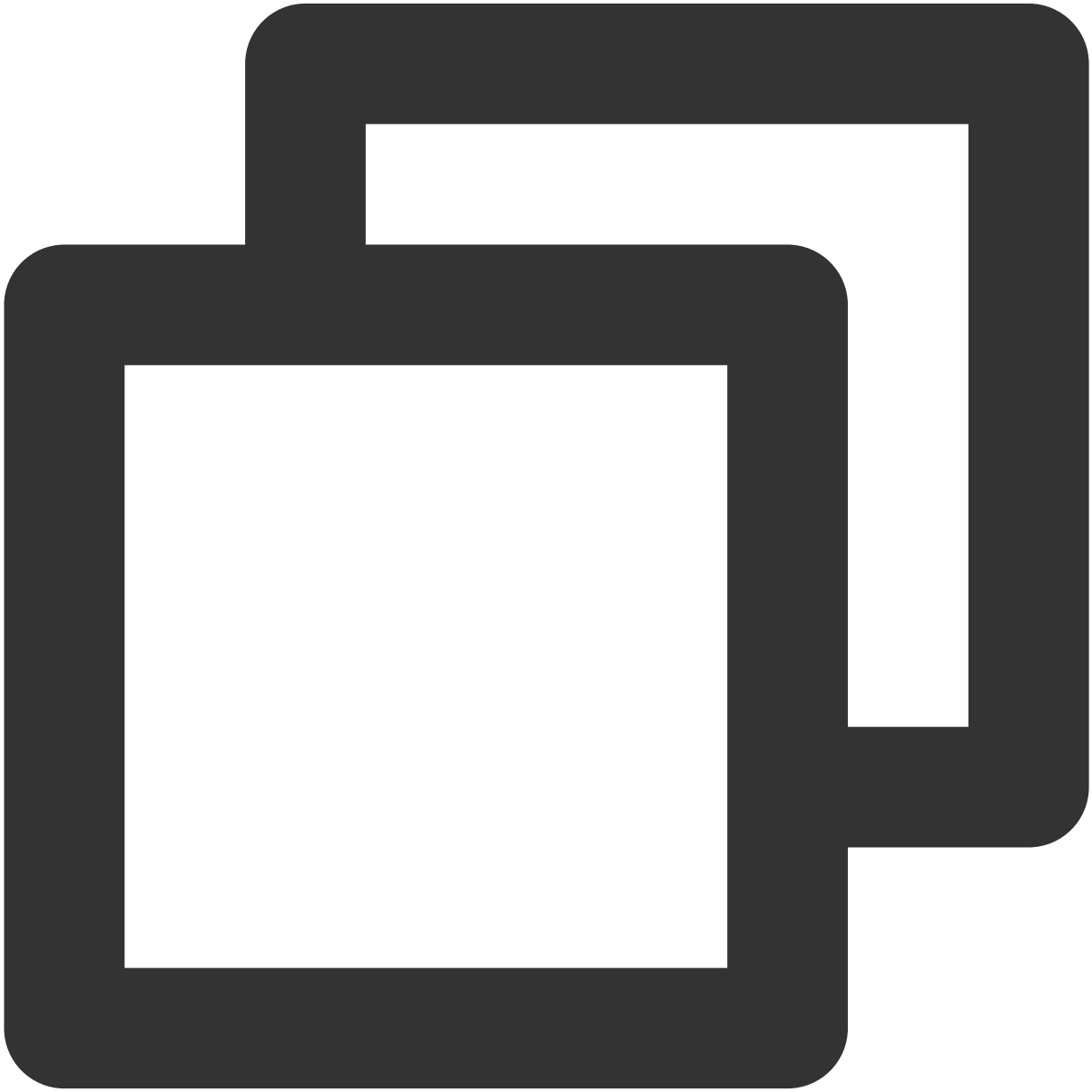
©2013-2022 Tencent Cloud. All rights reserved.

```
        NSLog(@"Time synchronization failed, and you can try re-synchronization.");
    }
}
```

Note:

NTP time synchronization results can reflect the current network quality of the application user. To ensure a good chorus experience, it is recommended not to allow users to participate in the chorus if time synchronization fails.

4. Receive chorus signaling.



```
- (void)onRecvCustomCmdMsgUserId:(NSString *)userId cmdID:(NSInteger)cmdID seq:(UIn
    JSONModel *json = [[JSONModel alloc] initWithData:message error:nil];
```

©2013-2022 Tencent Cloud. All rights reserved.

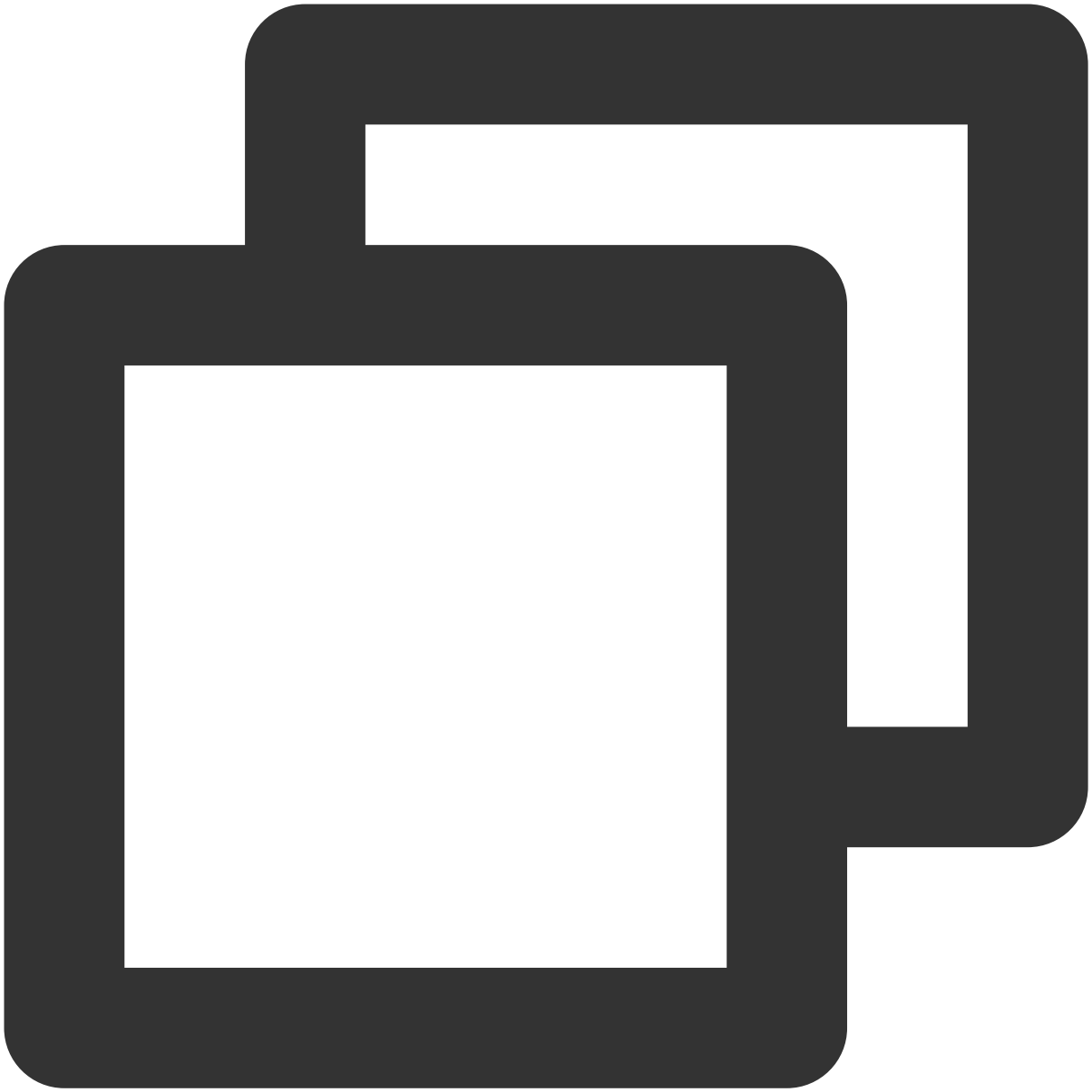
```
    NSDictionary *dic = json.toDictionary;
    // Match the chorus signaling.
    if ([dic[@"cmd"] isEqualToString:@"startChorus"]) {
        self.startPlayMusicTS = [dic[@"startPlayMusicTS"] integerValue];
        self.musicId = [dic[@"musicId"] intValue];
        self.musicDuration = [dic[@"musicDuration"] intValue];
        // Agree on the time difference between chorus time and current time.
        self.delayMs = self.startPlayMusicTS - [TXLiveBase getNetworkTimestamp];
    }
}
```

Note:

Once the chorus members receive the chorus signaling and join in, the status should be changed to Chorus In Progress. Chorus signaling would not be responded to again before the end of this chorus round.

5. Play accompaniment, and start chorus.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)playBmgExample {
    // Chorus has not started.
    if (self.delayMs > 0) {
        // Begin to preload music.
        [self preloadMusicWithStartTimeMS:0];
        // Play music after a delay of delayMs.
        [self startPlayMusicWithStartTimeMS:0];
    } else if (labs(self.delayMs) < self.musicDuration) {
        // Chorus is in progress.
        // Play start time: Absolute value of the time difference + preload delay (
        NSInteger startTimeMS = labs(self.delayMs) + 400;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Begin to preload music.
        [self preloadMusicWithStartTimeMS:startTimeMS];
        // Start playing music after a preload delay (e.g., 400 ms).
        [self startPlayMusicWithStartTimeMS:startTimeMS];
    } else {
        // Chorus has ended.
        // Joining the chorus is not allowed.
    }
}

// Preload music.
- (void)preloadMusicWithStartTimeMS:(NSInteger)startTimeMS {
    // musicId: Obtained from chorus signaling. musicUrl: Corresponding music resou
    TXAudioMusicParam *musicParam = [[TXAudioMusicParam alloc] init];
    musicParam.ID = self.musicId;
    musicParam.path = self.musicUrl;
    // Only local music playback.
    musicParam.publish = NO;
    musicParam.startTimeMS = startTimeMS;
    [self.audioEffectManager preloadMusic:musicParam onProgress:nil onError:nil];
}

// Begin to play music.
- (void)startPlayMusicWithStartTimeMS:(NSInteger)startTimeMS {
    // musicId: Obtained from chorus signaling. musicUrl: Corresponding music resou
    TXAudioMusicParam *musicParam = [[TXAudioMusicParam alloc] init];
    musicParam.ID = self.musicId;
    musicParam.path = self.musicUrl;
    // Only local music playback.
    musicParam.publish = NO;
    musicParam.startTimeMS = startTimeMS;
    [self.audioEffectManager startPlayMusic:musicParam onStart:nil onProgress:nil o
}
```

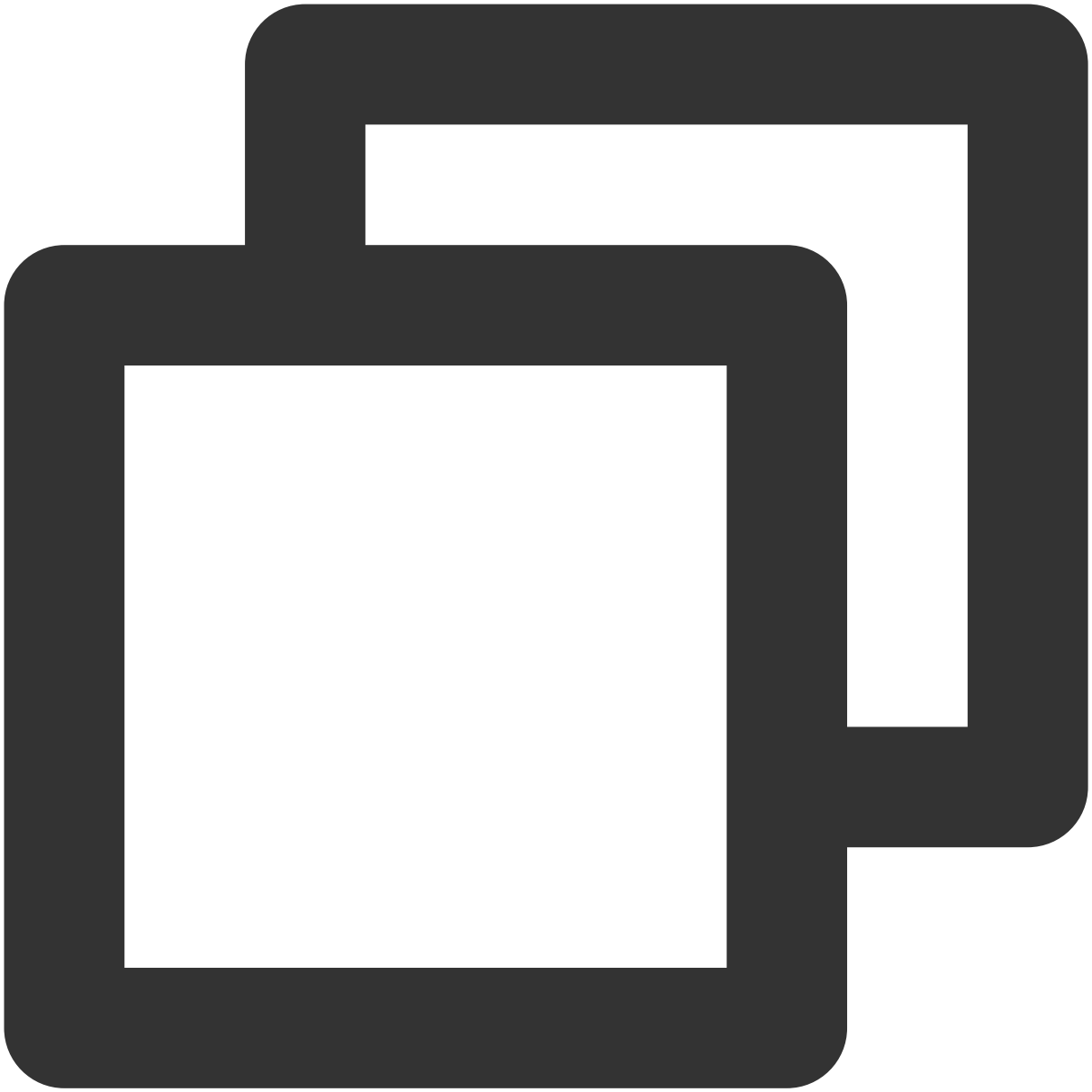
Note:

To minimize transmission delay as much as possible, chorus members perform along with the local playback of accompaniment music, and they do not need to publish or receive remote music.

Based on `delayMs`, the current chorus status can be determined. Developers must implement the `startPlayMusic` delayed call for different statuses on their own.

6. Accompaniment Synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
// Agreed chorus start time.
@property (nonatomic, assign) NSInteger startPlayMusicTS;

- (void)syncBgmExample {
    // Actual playback progress of the current accompaniment music.
    NSInteger currentProgress = [[self.trtcCloud getAudioEffectManager] getMusicCur
    // Ideal playback progress of the current accompaniment music.
    NSInteger estimatedProgress = [TXLiveBase getNetworkTimestamp] - self.startPlay
    // When the progress difference exceeds 50 ms, corrections are made.
    if (estimatedProgress >= 0 && labs(currentProgress - estimatedProgress) > 50) {
        [[self.trtcCloud getAudioEffectManager] seekMusicToPosInMS:self.musicId pts
```

©2013-2022 Tencent Cloud. All rights reserved.

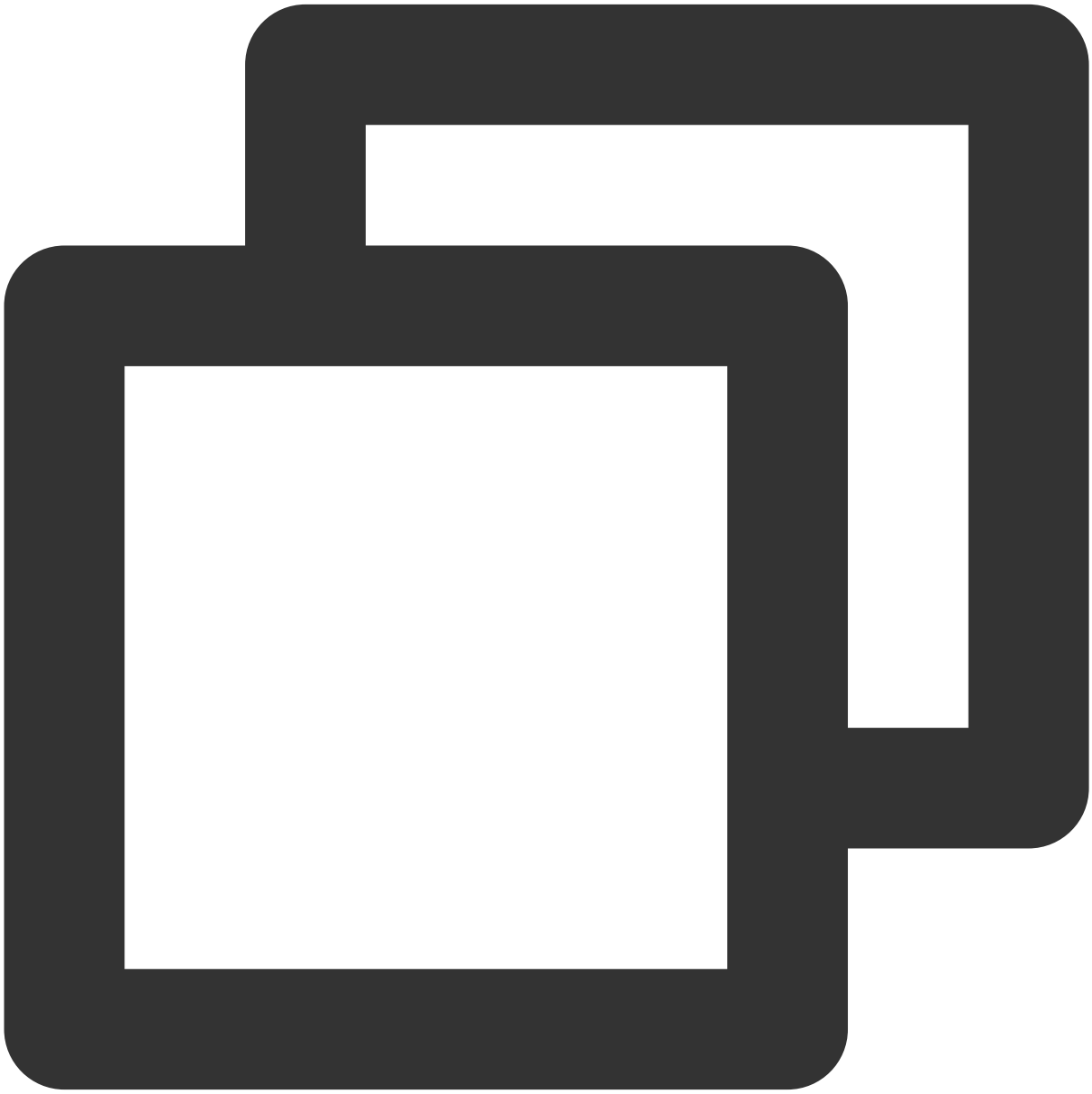
```
    }
}
```

7. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Local lyric synchronization.

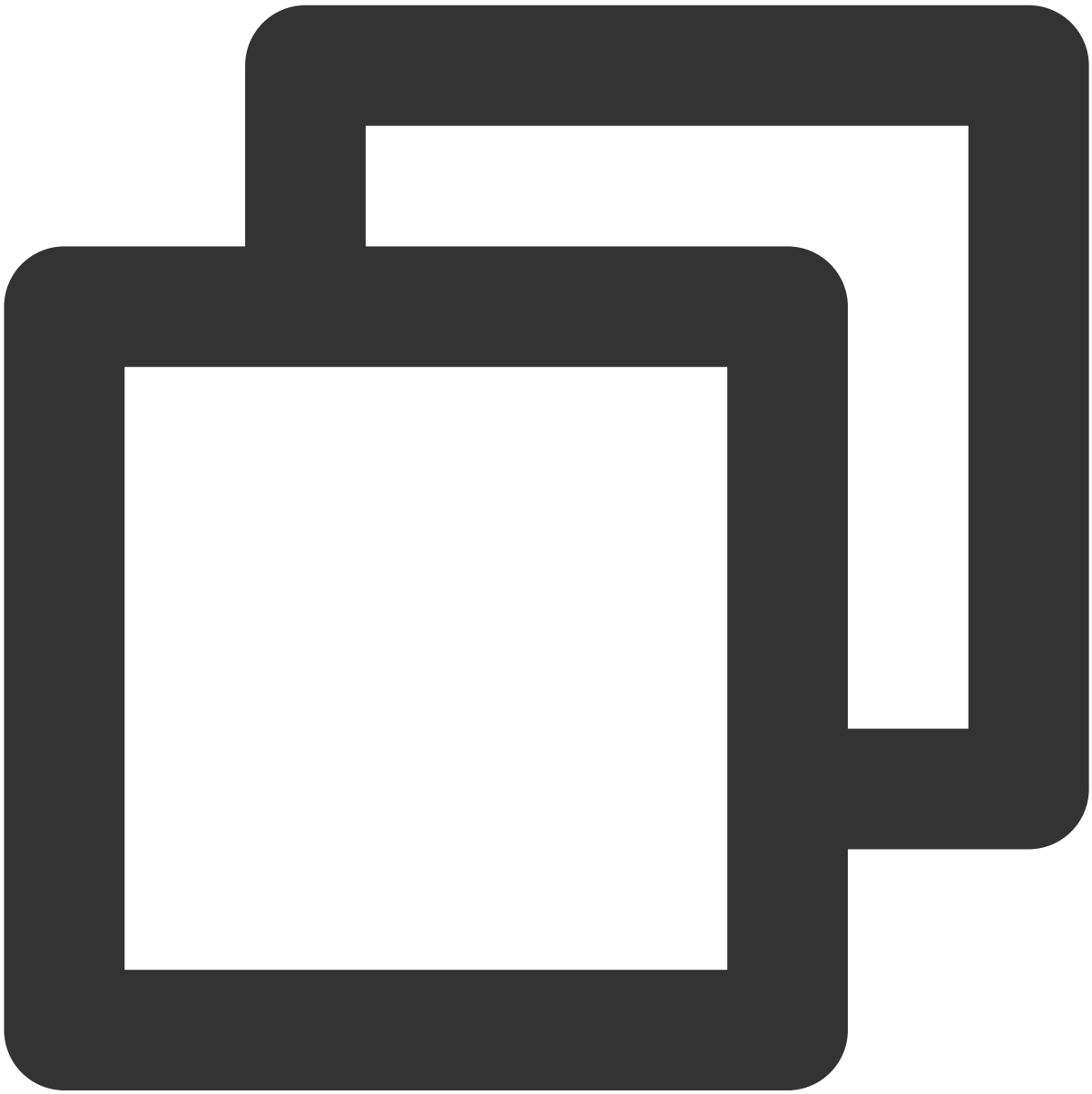


```
[self.audioEffectManager startPlayMusic:musicParam onStart:^(NSInteger errCode) {
    // Start playing music.
} onProgress:^(NSInteger progressMs, NSInteger durationMs) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // TODO: The logic of updating the lyric control.
    // Determine whether seek in the lyrics control is needed based on the latest p
} onComplete:^(NSInteger errCode) {
    // Music playback completed.
}];
```

8. Become a listener and exit the room.



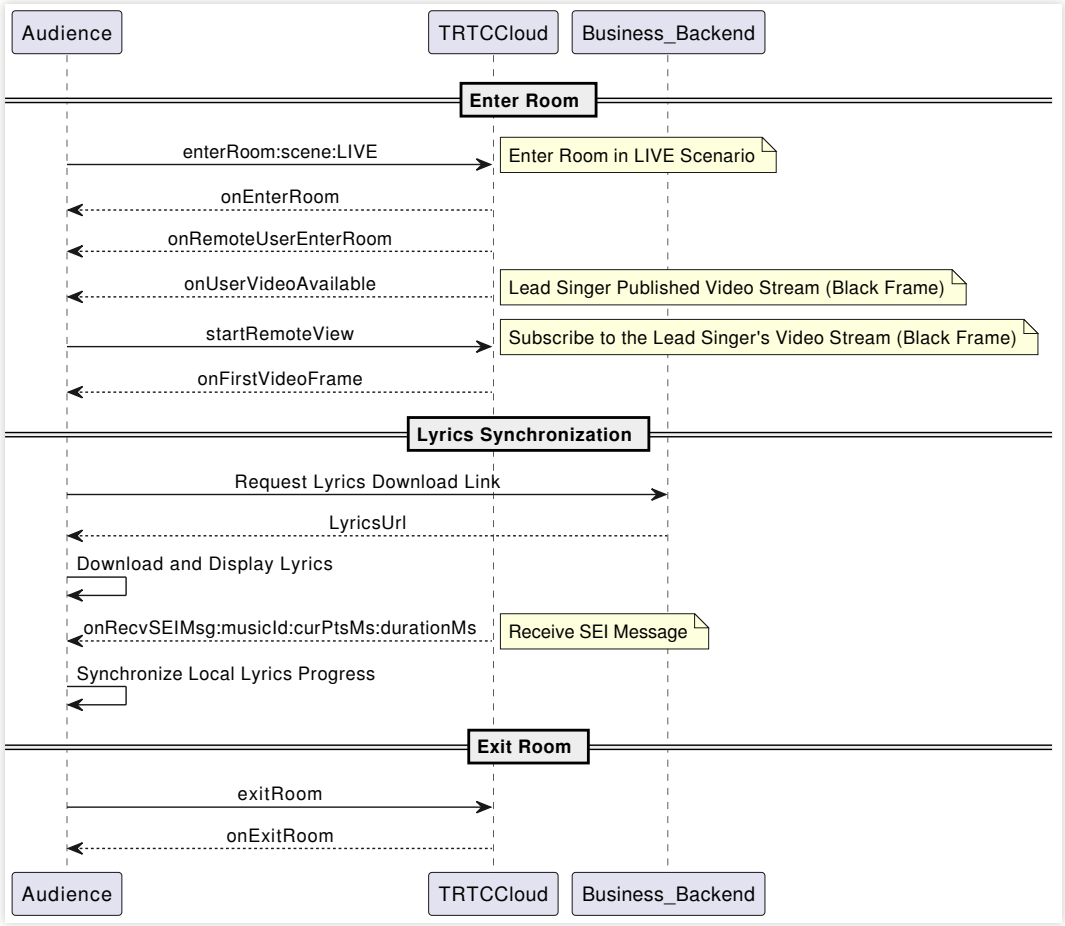
```
- (void)exitRoomExample {
    // Use the experimental API to disable chorus mode.
    [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"enableChorus\\",\\"params\\
    // Use the experimental API to disable low-latency mode.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"setLowLatencyModeEnabled\\"
    // Switched to the audience role.
    [self.trtcCloud switchRole:TRTCRoleAudience];
    // Stop playing accompaniment music.
    [[self.trtcCloud getAudioEffectManager] stopPlayMusic:self.musicId];
    // Stop local audio capture and publishing.
    [self.trtcCloud stopLocalAudio];
    // Exit the room.
    [self.trtcCloud exitRoom];
}
```

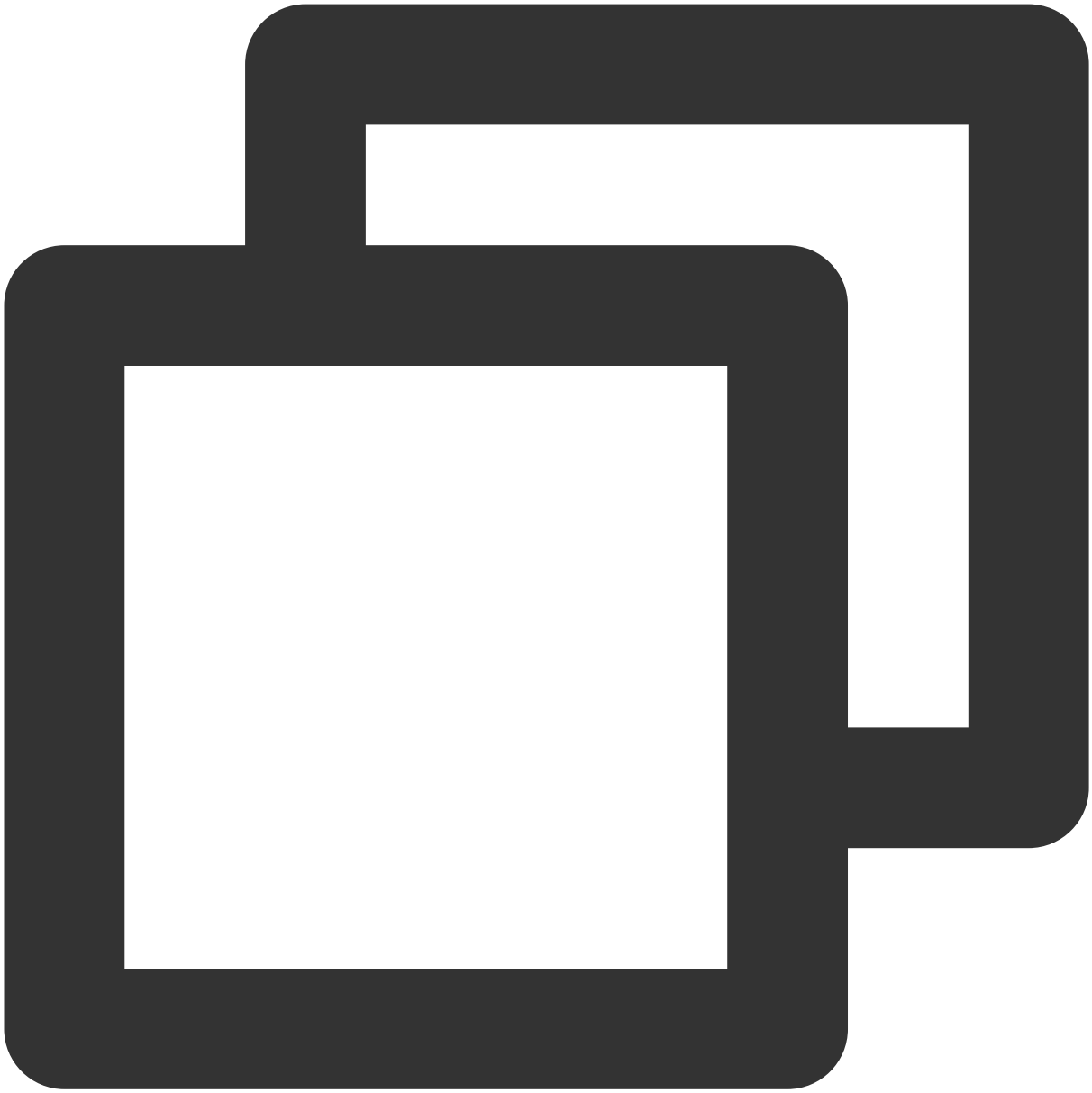
Perspective 3: Listener actions

Sequence diagram



1. Enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)enterRoomWithRoomId:(NSString *)roomId userID:(NSString *)userId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = [self generateUserSig:userId];
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // It is recommended to enter the room as an audience role.
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // LIVE should be selected for the room entry scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

To better transmit SEI messages for lyric synchronization, it is recommended to choose `TRTCAppSceneLIVE` for room entry scenarios.

Under the automatic subscription mode (default), audiences automatically subscribe and play the on-mic anchor's audio and video streams upon entering the room.

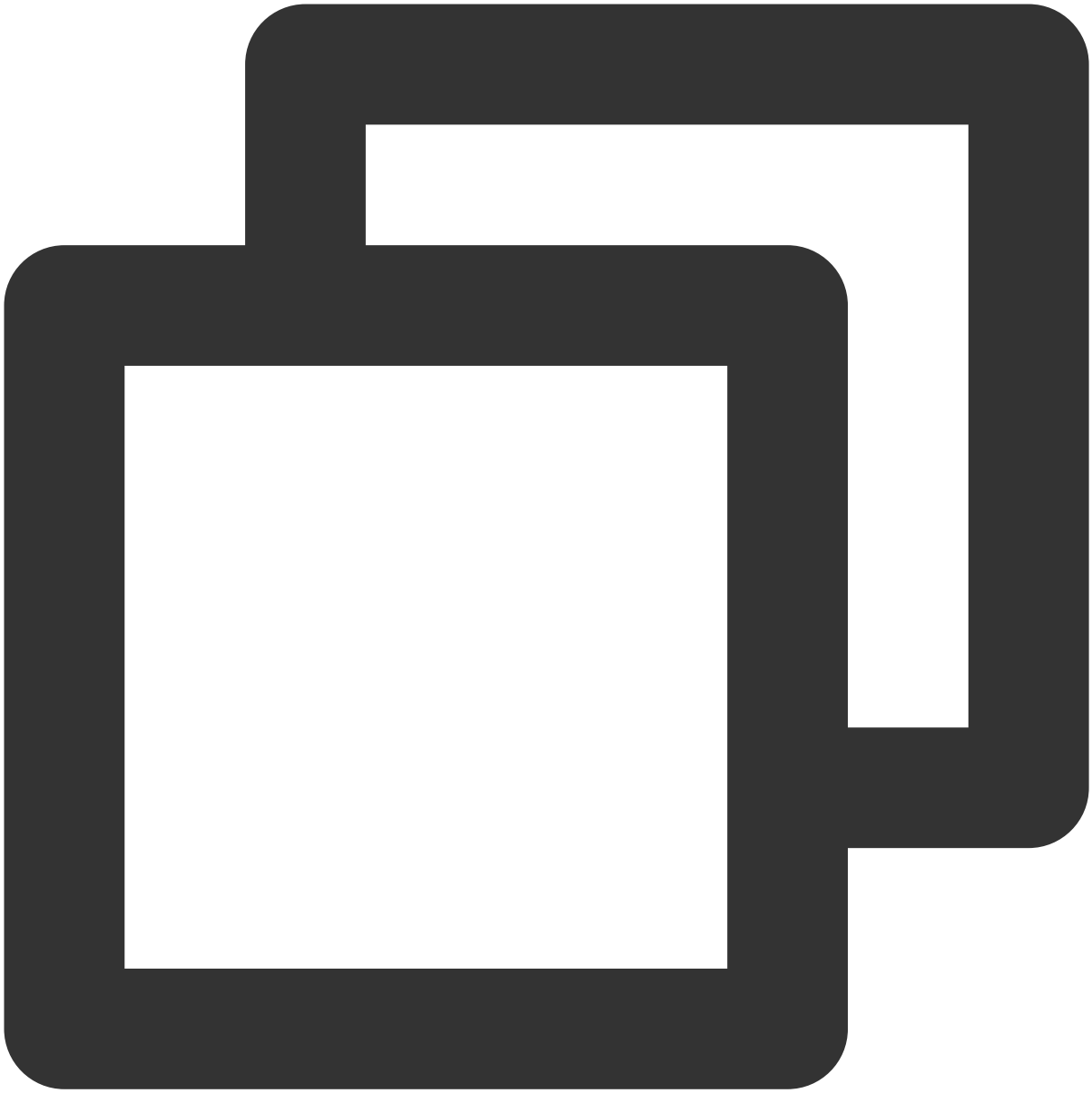
2. Lyric synchronization

Download lyrics.

Obtain the target lyrics download link, LyricsUrl, from the business backend, and cache the target lyrics locally.

Listener end lyric synchronization

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    if (available) {
        [self.trtcCloud startRemoteView:userId view:nil];
    } else {
        [self.trtcCloud stopRemoteView:userId];
    }
}

- (void)onRecvSEIMsg:(NSString *)userId message:(NSData *)message {
    JSONModel *json = [[JSONModel alloc] initWithData:message error:nil];
    NSDictionary *dic = json.toDictionary;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    int32_t musicId = [dic[@"musicId"] intValue];
    NSInteger progress = [dic[@"progress"] integerValue];
    NSInteger duration = [dic[@"duration"] integerValue];
    // ......
    // TODO: The logic of updating the lyric control.
    // Based on the received latest progress and the local lyrics progress deviatio
    // ......
}
```

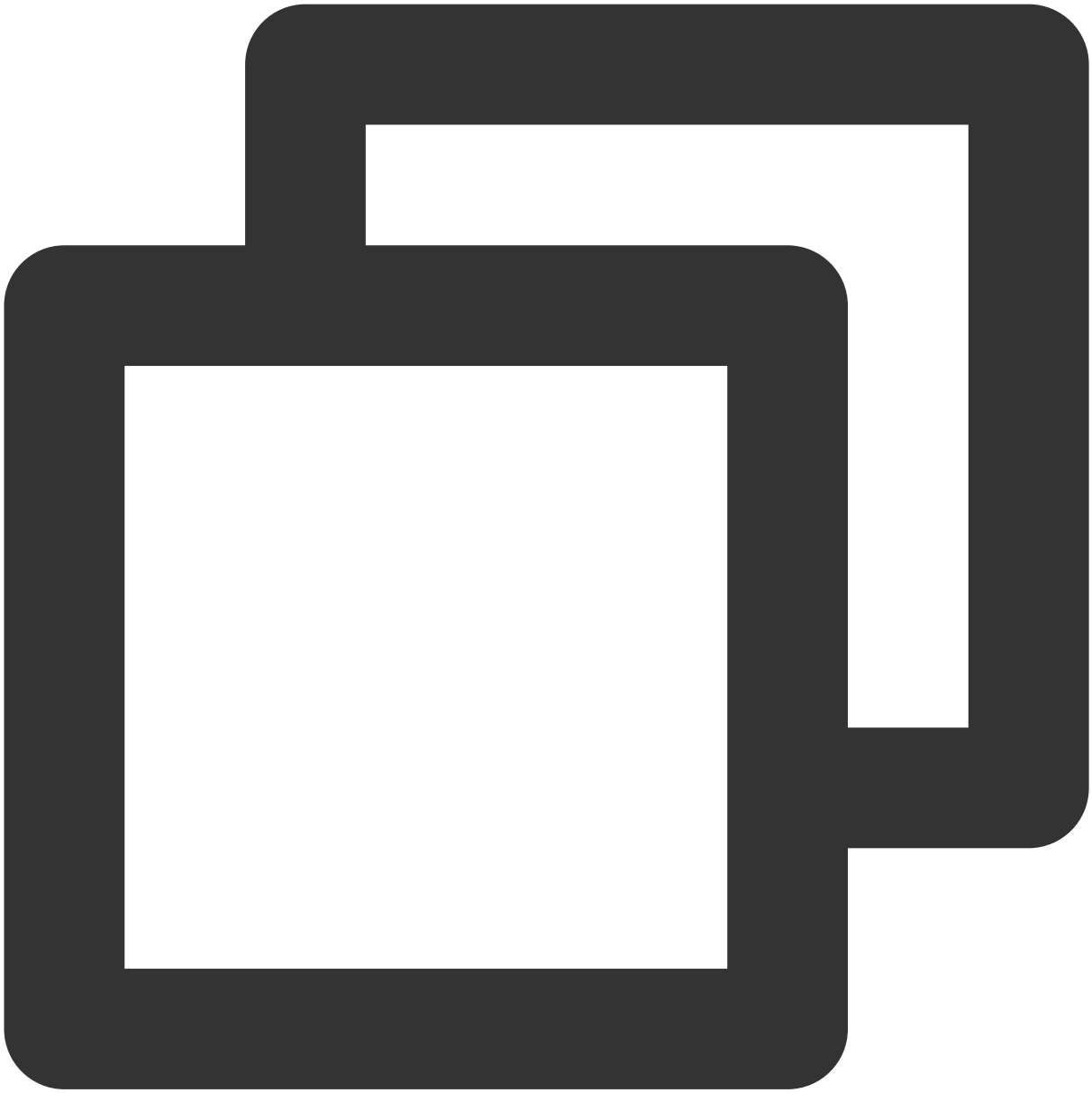
Note:

Listeners need to actively subscribe to the lead singer's video streams in order to receive the SEI messages carried by black frames.

If the lead singer's mixed stream also mixes in black frames, then only subscribing to the mixing stream robot's video stream is required.

3. Exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Exit the room.
[self.trtcCloud exitRoom];

// Exit room event callback.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room.");
    } else if (reason == 1) {
        NSLog(@"Removed from the current room by the server.");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved.");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }
```

Advanced Features

Music scoring module integration

Music scoring provides users with multi-dimensional singing scoring capabilities. Currently, supported scoring dimensions include intonation and rhythm.

1. Prepare scoring-related files.

Prepare in advance the performance recording files to be scored, original music standard files, MIDI pitch files, and upload them to COS storage.

2. Create a music scoring task.

Request Method: POST(HTTP).

Request Address: http://service-mqk0mc83-1257411467.bj.apigw.tencentcs.com/release/job.

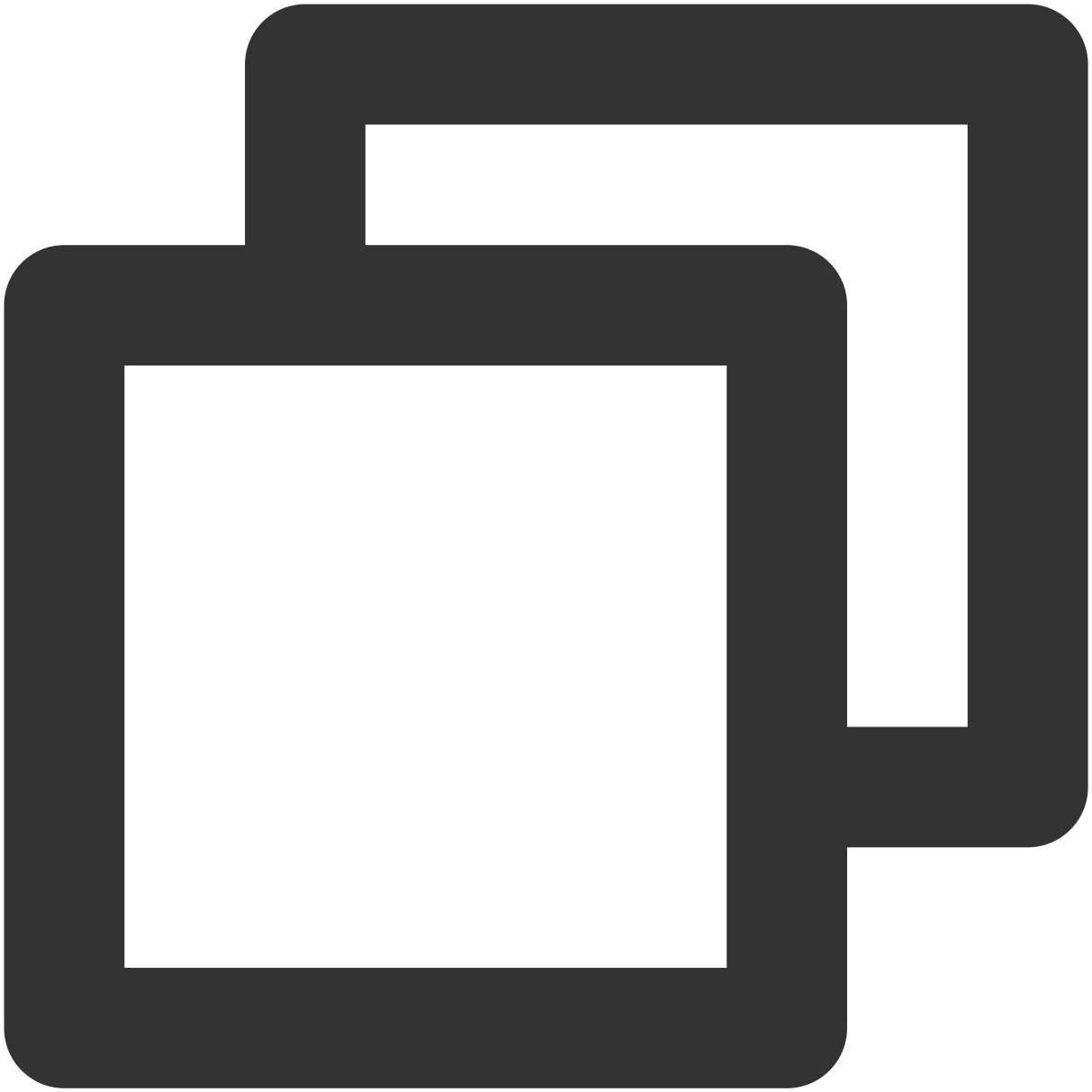
Request Header: Content-Type: application/json.

A request sample is as follows:

Request sample:

Response sample:

©2013-2022 Tencent Cloud. All rights reserved.

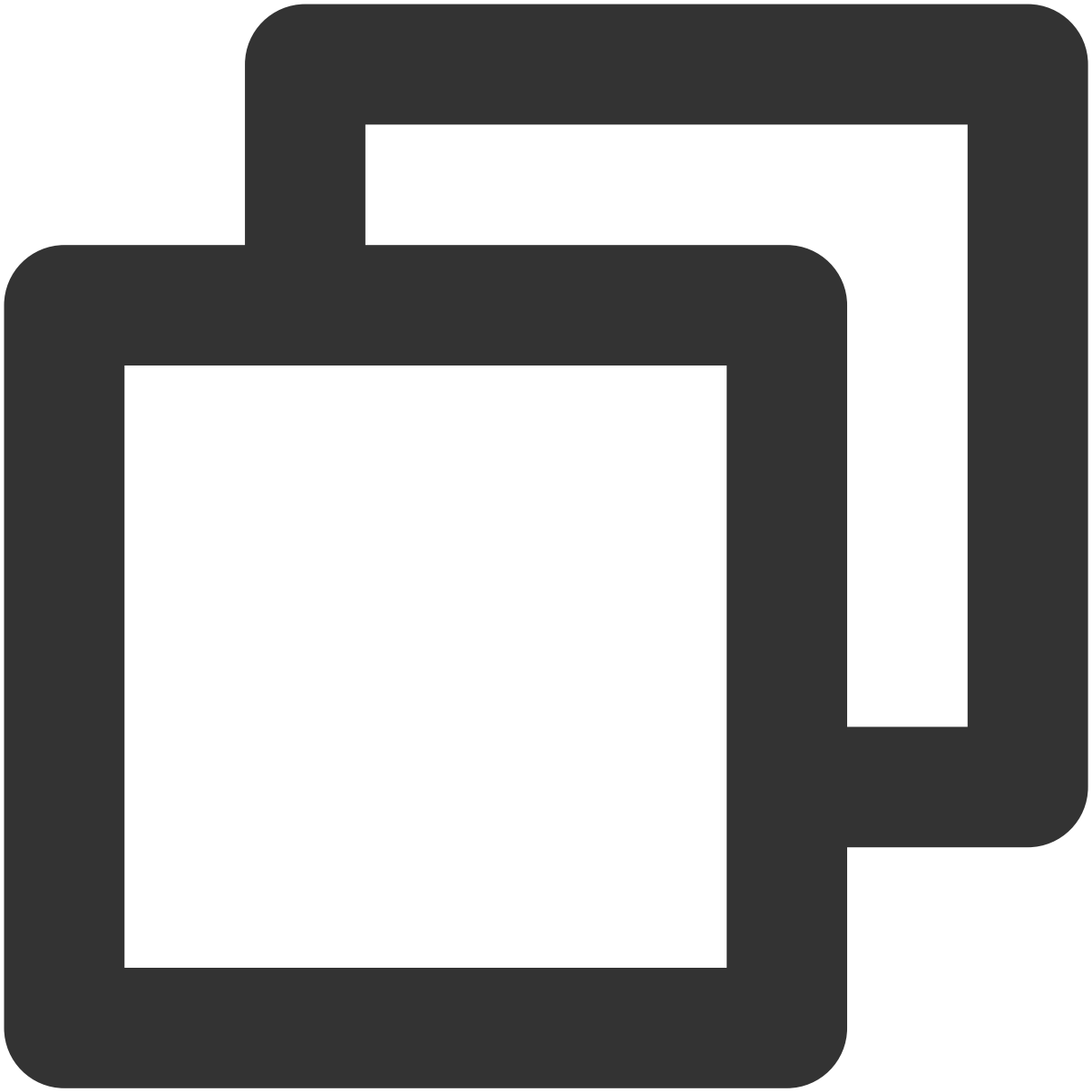


```
{
  "action": "CreateJob",
  "secretId": "{secretId}",
  "secretKey": "{secretKey}",
  "createJobRequest": {
    "customId": "{customId}",
    "callback": "{callback}",
    "inputs": [{ "url": "{url}" }],
    "outputs": [
      {
        "contentId": "{contentId}",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        "destination": "{destination}",
        "inputSelectors": [0],
        "smartContentDescriptor": {
          "outputPrefix": "{outputPrefix}",
          "vocalScore": {
            "standardAudio": {
              "midi": {"url":"{url}"},
              "standardWav": {"url":"{url}"},
              "alignWav": {"url":"{url}"}
            }
          }
        }
      }
    ]
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.




```
{
    "requestId": "ac004192-110b-46e3-ade8-4e449df84d60",
    "createJobResponse": {
        "job": {
            "id": "13f342e4-6866-450e-b44e-3151431c578b",
            "state": 1,
            "customId": "{customId}",
            "callback": "{callback}",
            "inputs": [{ "url": "{url}" }],
            "outputs": [
                {
```


©2013-2022 Tencent Cloud. All rights reserved.

```
                    "contentId": "{contentId}",
                    "destination": "{destination}",
                    "inputSelectors": [0],
                    "smartContentDescriptor": {
                        "outputPrefix": "{outputPrefix}",
                        "vocalScore": {
                          "standardAudio": {
                            "midi": {"url":"{url}"},
                            "standardWav": {"url":"{url}"},
                            "alignWav": {"url":"{url}"}
                          }
                        }
                    }
                }
            ],
            "timing": {
                "createdAt": "1603432763000",
                "startedAt": "0",
                "completedAt": "0"
            }
        }
    }
}
```

3. Obtain music scoring results.

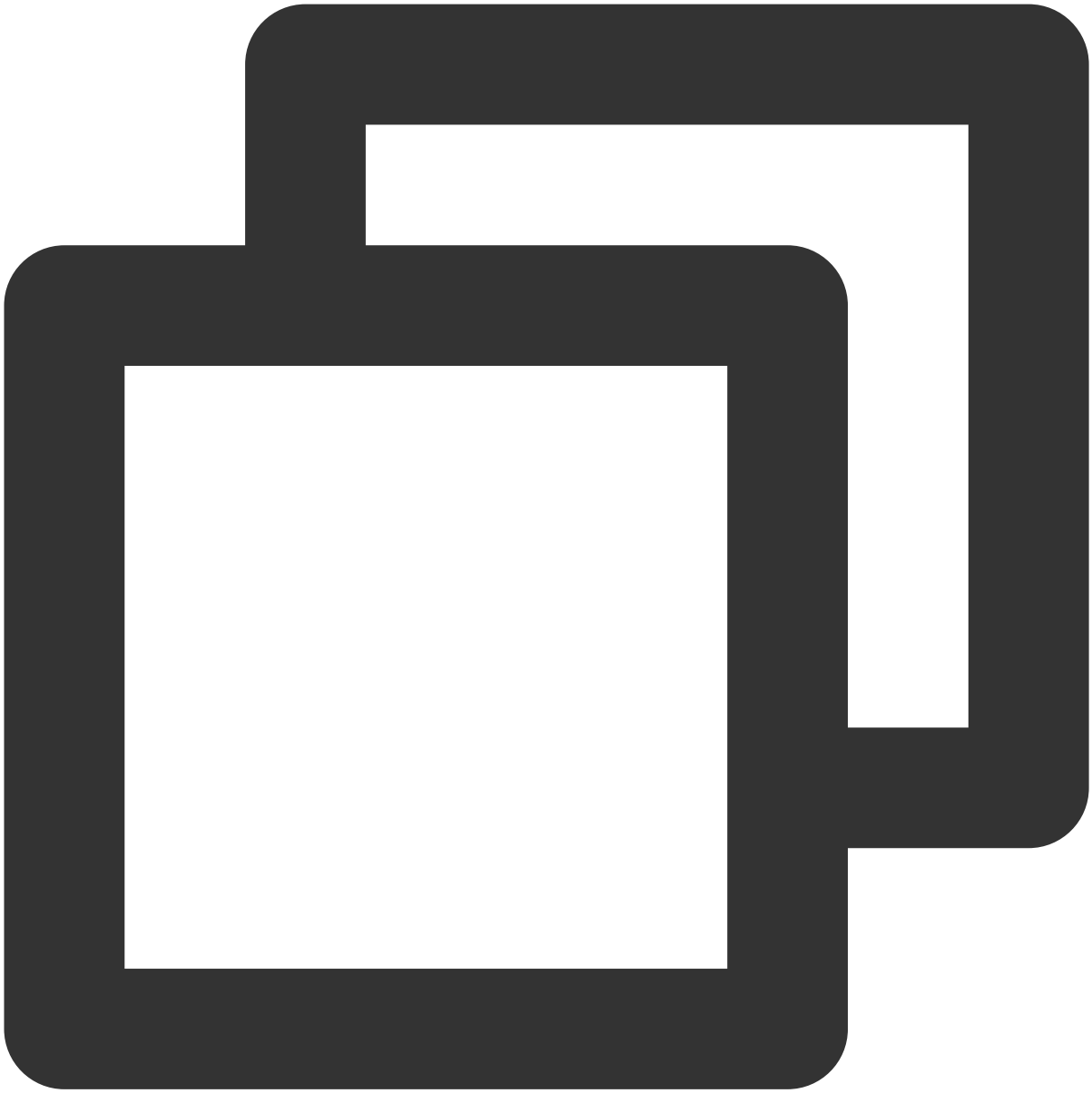
Obtain Method: Divided into active acquisition and passive callback.

By querying with the ID obtained from the response packet after creating the task, if the queried task is successful (state=3), the task's Output will carry the smartContentResult structure, in which the vocalScore field stores the result JSON file name. Users can construct the output file's COS path based on the information in Output's COS and destination.

Request sample:

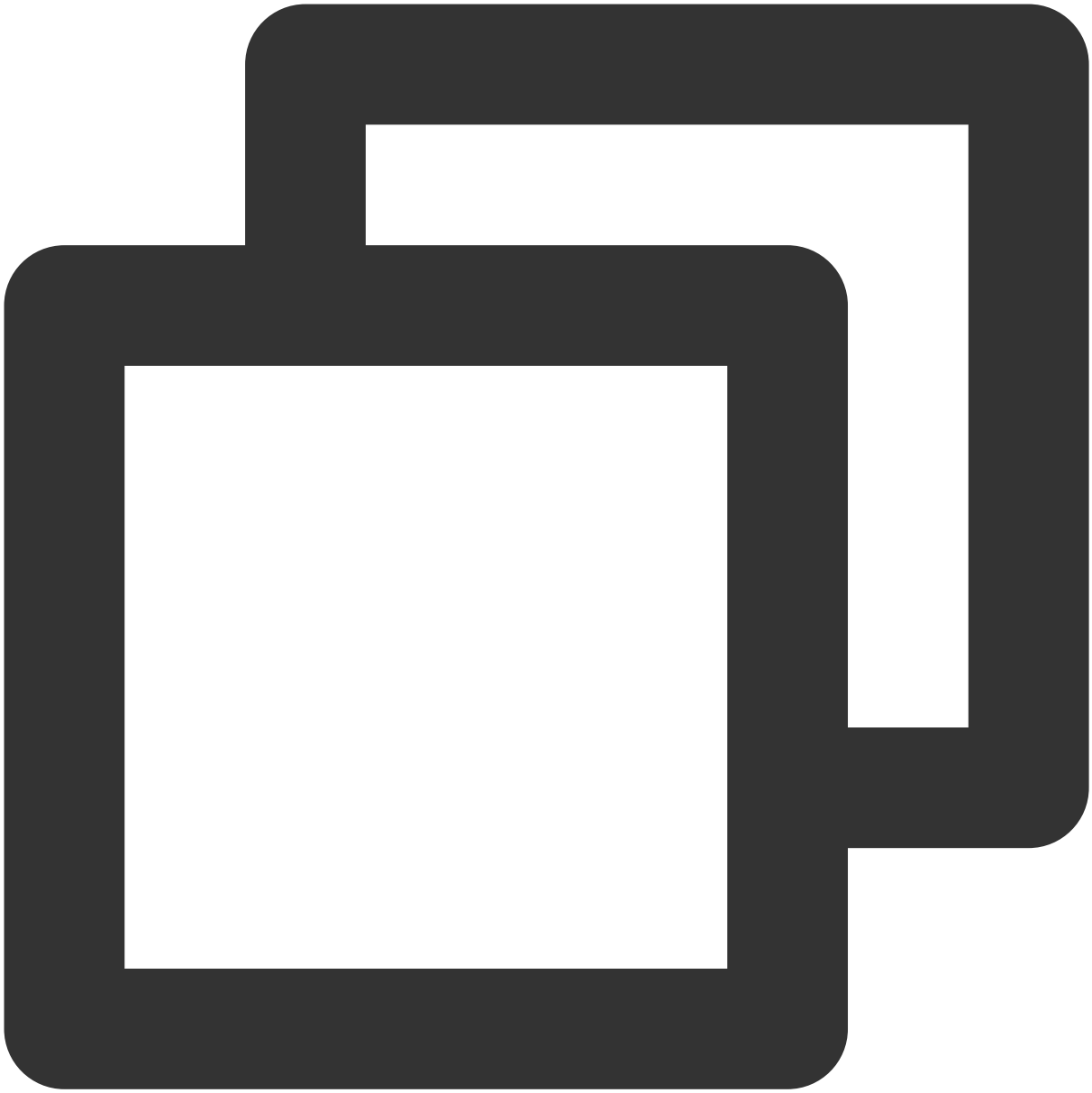
Response sample:

©2013-2022 Tencent Cloud. All rights reserved.



```
{
  "action": "GetJob",
  "secretId": "{secretId}",
  "secretKey": "{secretKey}",
  "getJobRequest": {
    "id": "{id}"
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.




```
{
  "requestId": "c9845a99-34e3-4b0f-80f5-f0a2a0ee8896",
  "getJobResponse": {
    "job": {
      "id": "a95e9d74-6602-4405-a3fc-6408a76bcc98",
      "state": 3,
      "customId": "{customId}",
      "callback": "{callback}",
      "timing": {
        "createdAt": "1610513575000",
        "startedAt": "1610513575000",
```


©2013-2022 Tencent Cloud. All rights reserved.

```
          "completedAt": "1610513618000"
        },
        "inputs": [{ "url": "{url}" }],
        "outputs": [
          {
            "contentId": "{contentId}",
            "destination": "{destination}",
            "inputSelectors": [0],
            "smartContentDescriptor": {
              "outputPrefix": "{outputPrefix}",
              "vocalScore": {
                "standardAudio": {
                  "midi": {"url":"{url}"},
                  "standardWav": {"url":"{url}"},
                  "alignWav": {"url":"{url}"}
                }
              }
            },
            "smartContentResult": {
              "vocalScore": "out.json"
            }
          }
        ]
      }
    }
  }
```

Passive callbacks need to fill in the callback field when creating a task. The platform will send the entire Job structure to the address specified by the callback after the task reaches the Completed state (COMPLETED/ERROR). It is recommended to obtain task results using passive callbacks. The entire Job structure of tasks that have reached the Completed state (COMPLETED/ERROR) will be sent to the address corresponding to the callback field specified when the task was created. See the active query sample for the Job structure (under getJobResponse).

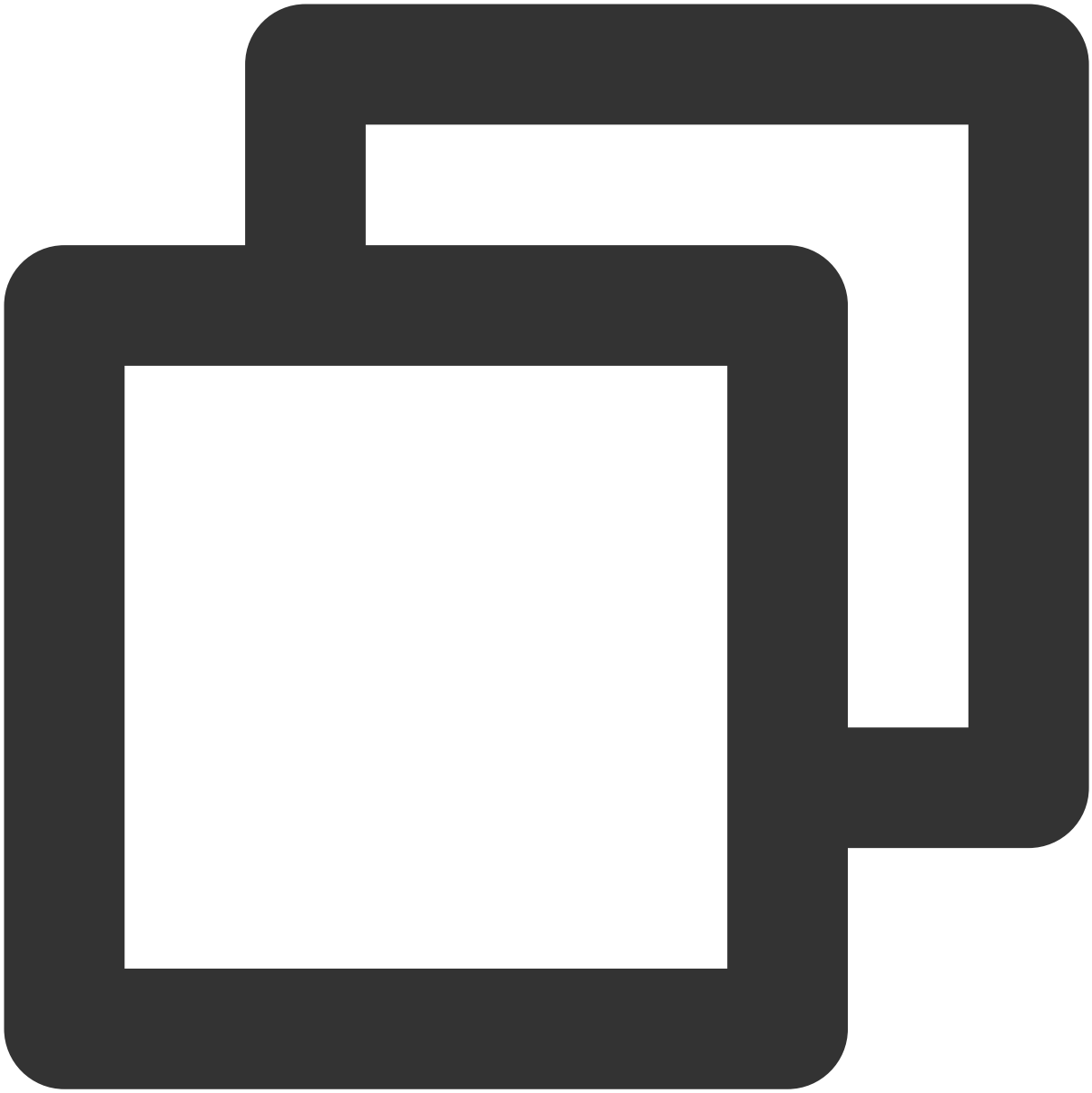
Note:

For more detailed intelligent music solution integration instructions for the music scoring module, see Music Scoring Integration.

Transparent transmission of single stream volume in mixed streams.

After the mixed streaming is enabled, the audience cannot directly obtain the on-mic anchor's single stream volume. In order to transparently transmit the single stream volume, the room owner may employ SEI to transmit the callback volume values of all on-mic anchors.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onUserVoiceVolume:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:(NSI
    if (userVolumes.count) {
        // For storing volume values corresponding to on-mic users.
        NSMutableDictionary *volumesMap = [NSMutableDictionary dictionary];
        for (TRTCVolumeInfo *user in userVolumes) {
            // Can set an appropriate volume threshold.
            if (user.volume > 10) {
                volumesMap[user.userId] = @(user.volume);
            }
        }
        JSONModel *json = [[JSONModel alloc] initWithDictionary:volumesMap error:ni
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Transmit a collection of on-mic users' volume via SEI messages.
        [self.trtcCloud sendSEIMsg:json.toJSONData repeatCount:1];
    }
}

- (void)onRecvSEIMsg:(NSString *)userId message:(NSData *)message {
    JSONModel *json = [[JSONModel alloc] initWithData:message error:nil];
    NSDictionary *dic = json.toDictionary;
    for (NSString *userId in dic.allKeys) {
        // Print the volume levels of single streams of all on-mic users.
        NSLog(@"%@: %@", userId, dic[userId]);
    }
}
```

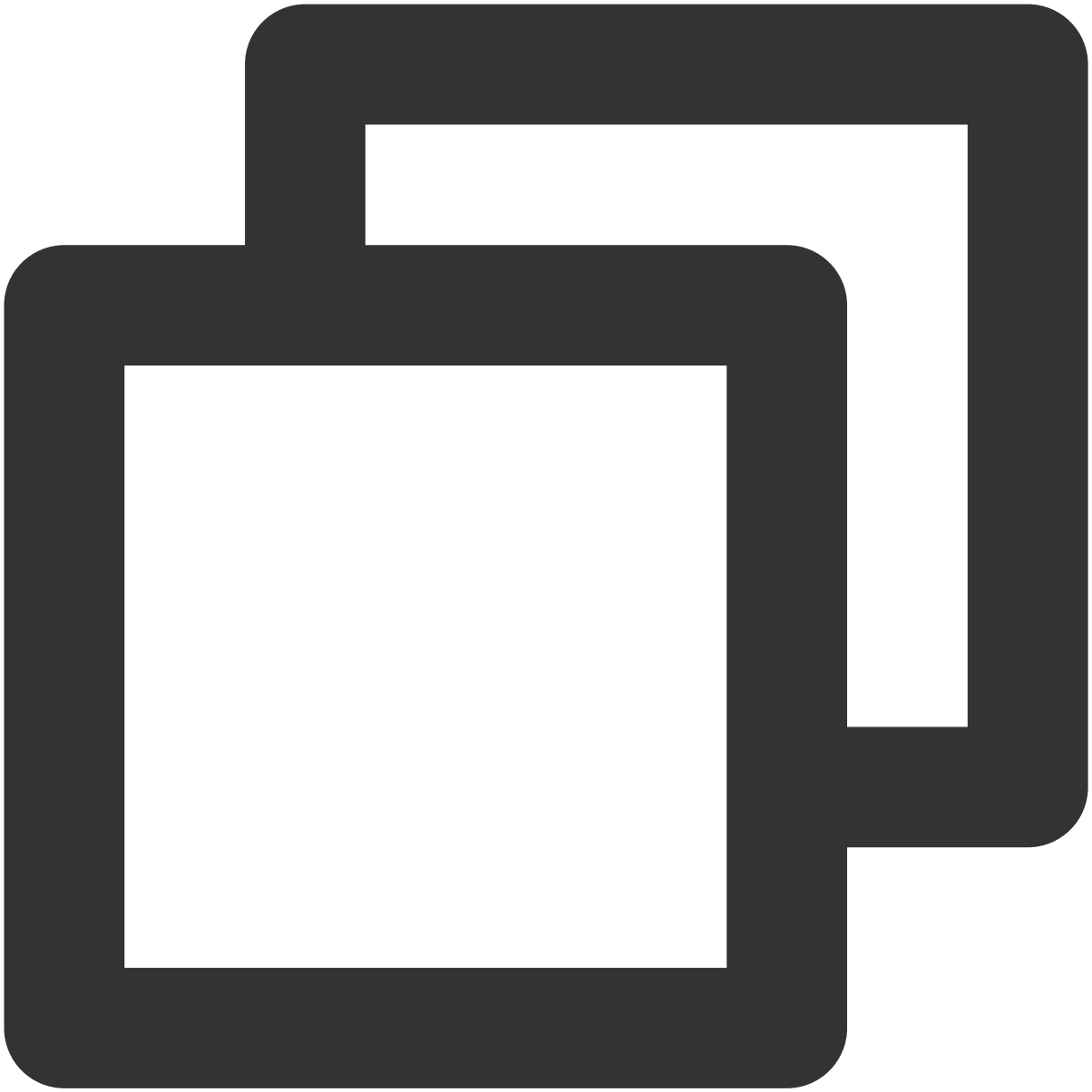
Note:

The prerequisite for using SEI messages to transparently transmit single stream volume through a mixed stream is that the room owner must either be video streaming or have black frame insertion enabled and furthermore, the audiences must actively subscribe to the room owner's video stream.

Real-time network quality callback

You can listen to `onNetworkQuality` to real-time monitor the network quality of both local and remote users. This callback is thrown every 2 seconds.

©2013-2022 Tencent Cloud. All rights reserved.



```
#pragma mark - TRTCCloudDelegate

- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<TRT
    // localQuality represents the local user's network quality evaluation result.
    // remoteQuality represents the remote user's network quality evaluation result
    switch(localQuality.quality) {
        case TRTCQuality_Unknown:
            NSLog(@"Undefined.");
            break;
        case TRTCQuality_Excellent:
            NSLog(@"The current network is excellent.");
```

©2013-2022 Tencent Cloud. All rights reserved.

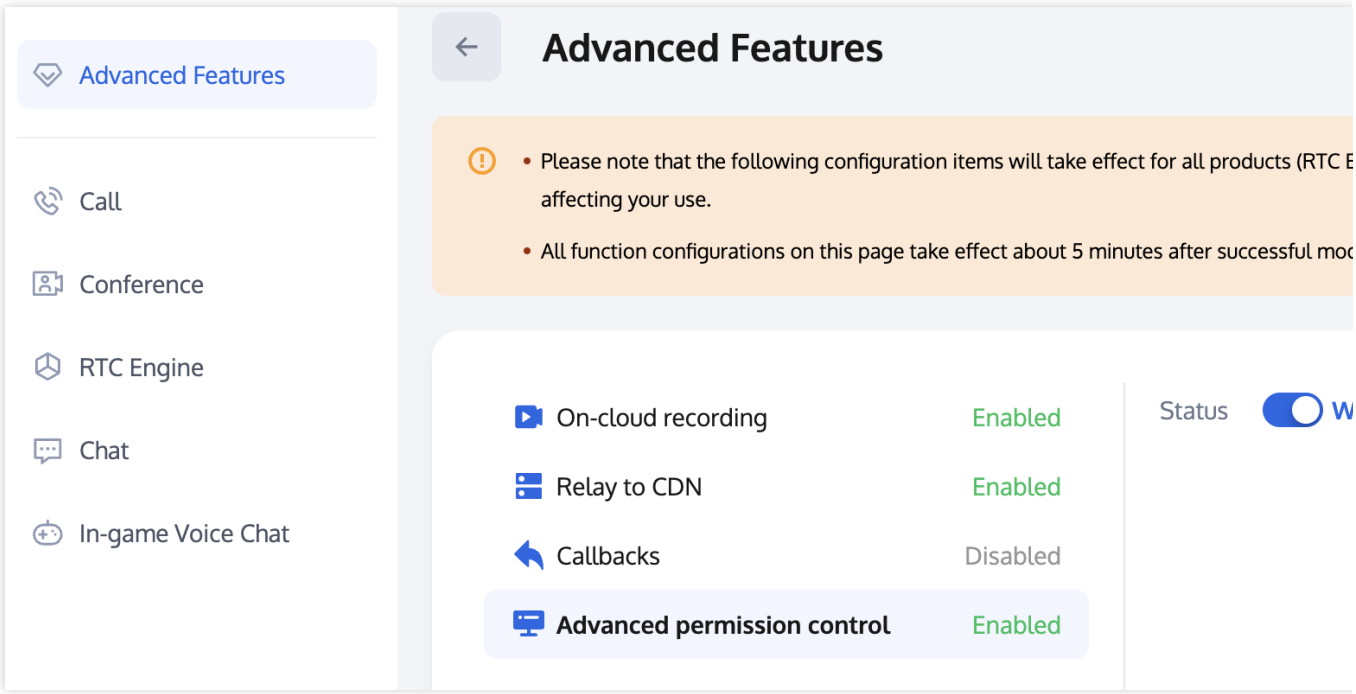
```
            break;
        case TRTCQuality_Good:
            NSLog(@"The current network is good.");
            break;
        case TRTCQuality_Poor:
            NSLog(@"The current network is moderate.");
            break;
        case TRTCQuality_Bad:
            NSLog(@"The current network is poor.");
            break;
        case TRTCQuality_Vbad:
            NSLog(@"The current network is very poor.");
            break;
        case TRTCQuality_Down:
            NSLog(@"The current network does not meet the minimum requirements of T
            break;
        default:
            break;
    }
}
```

Advanced permission control

TRTC advanced permission control can be used to set different entry permissions for different rooms, such as advanced VIP rooms. It can also be used to control the permission for the audience to speak, such as handling ghost microphones.

Step 1: Enable the Advanced Permission Control Switch in the TRTC console application's advanced features page.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

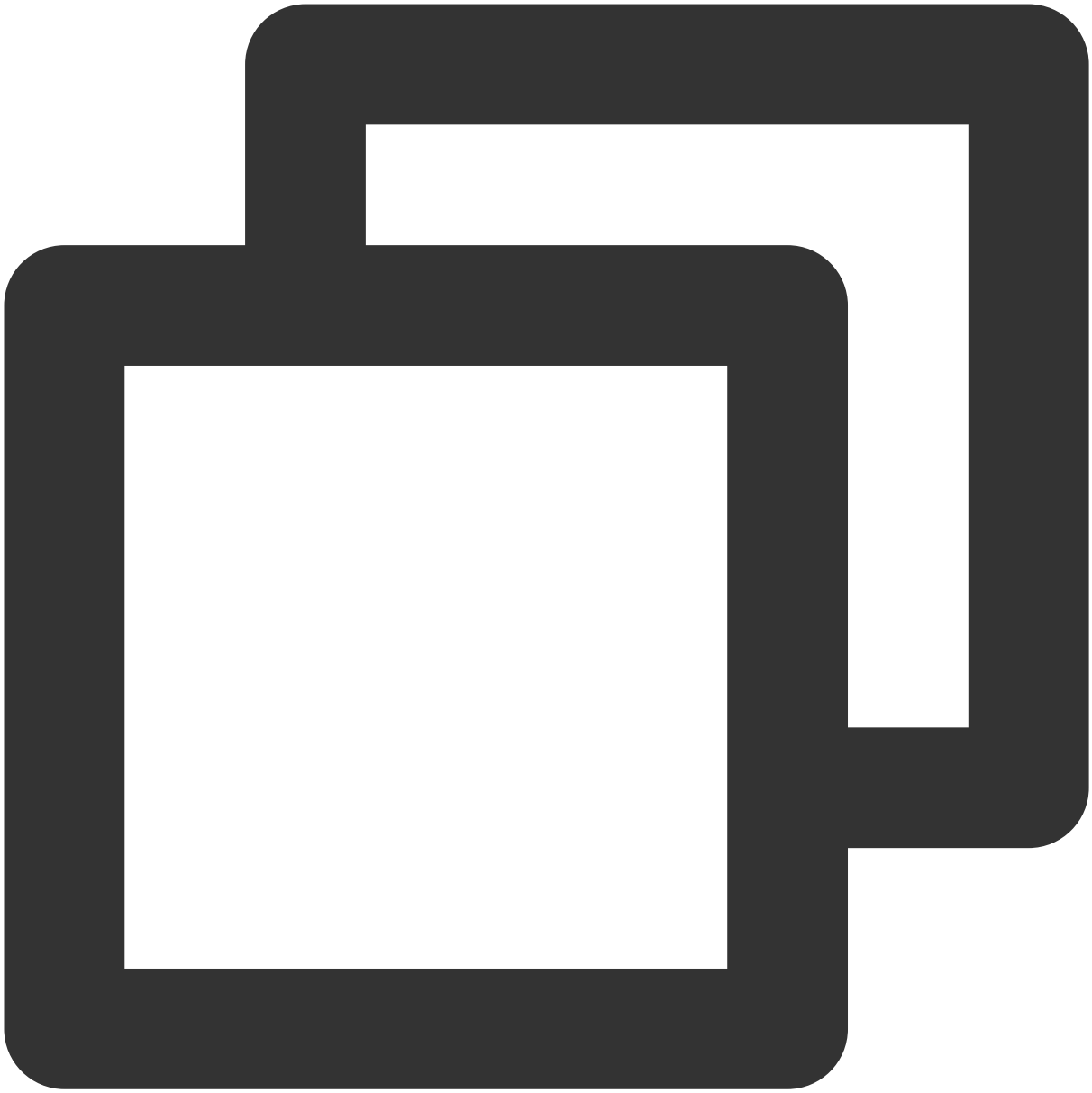
Once advanced permission control is enabled for a certain SDKAppID, all users using that SDKAppID need to pass in the `privateMapKey` parameter in `TRTCParams` to successfully enter the room. Therefore, if you have users online using this SDKAppID, do not enable this feature.

Step 2: Generate privateMapKey on the backend. For sample code, see privateMapKey computation source code.

Step 3: Room entry verification & speaking permission verification with PrivateMapKey.

Room entry verification

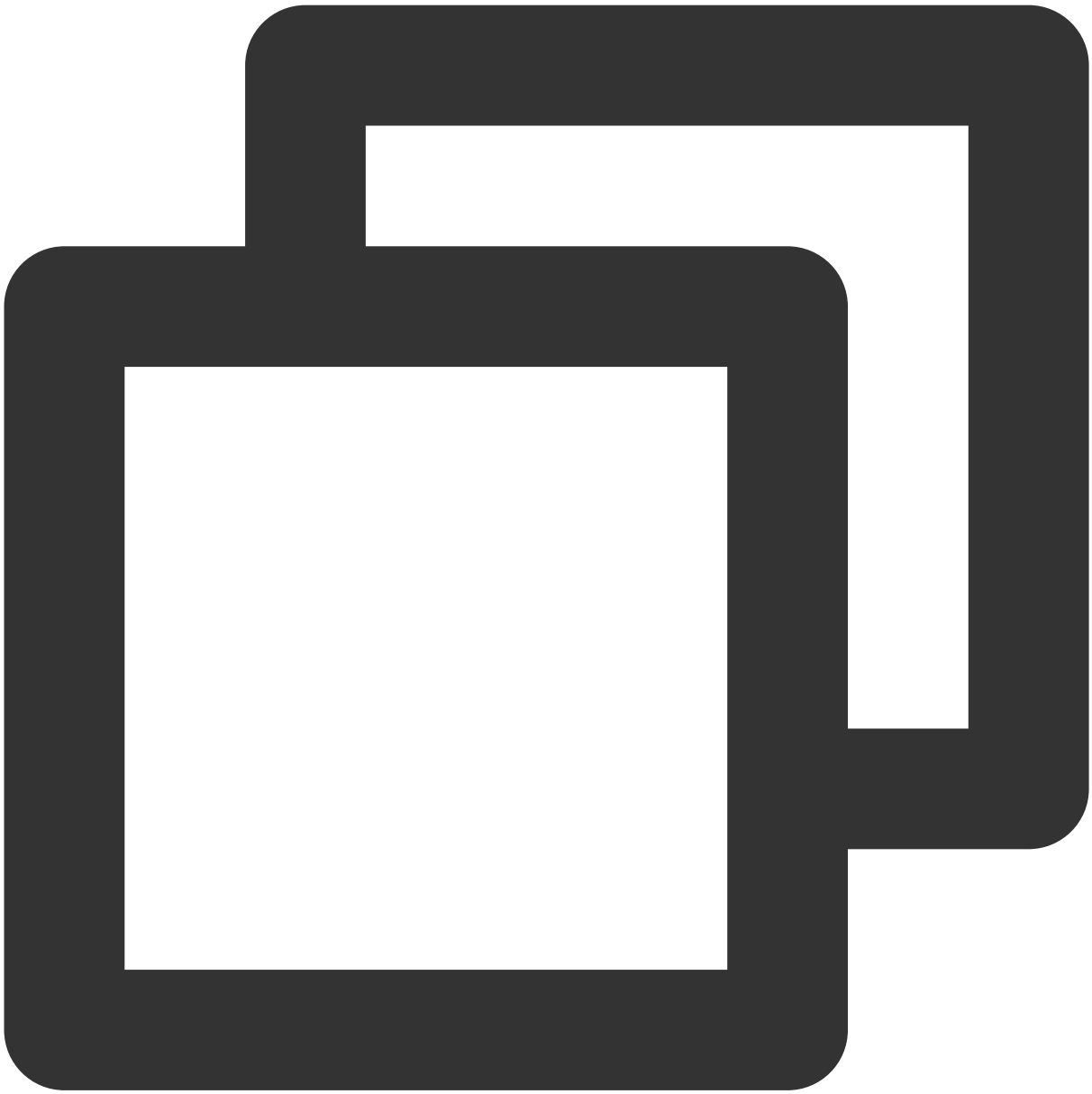
©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCParams *params = [[TRTCParams alloc] init];
params.sdkAppId = SDKAppID;
params.roomId = self.roomId;
params.userId = self.userId;
// UserSig obtained from the business backend.
params.userSig = [self getUserSig];
// PrivateMapKey obtained from the backend.
params.privateMapKey = [self getPrivateMapKey];
params.role = TRTCRoleAudience;
[self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
```

©2013-2022 Tencent Cloud. All rights reserved.

Speaking permission verification



```
// Pass in the latest PrivateMapKey obtained from the backend into the role switchi
[self.trtcCloud switchRole:TRTCRoleAnchor privateMapKey:[self getPrivateMapKey]];
```

Exception Handling

Exception error handling

©2013-2022 Tencent Cloud. All rights reserved.

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description

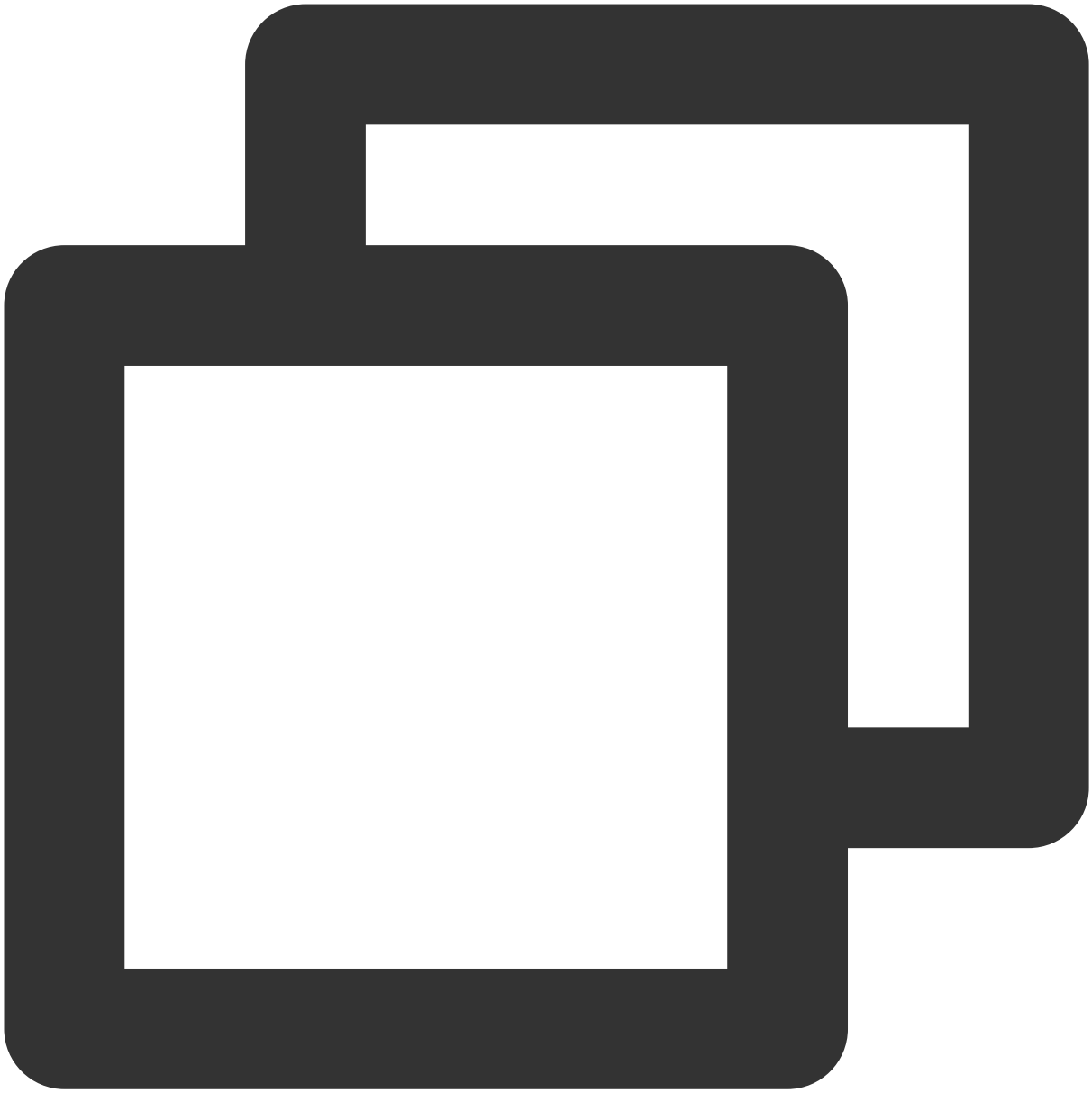
©2013-2022 Tencent Cloud. All rights reserved.

ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_SPEAKER_START_FAIL	-1321	Failed to open the speaker. For example, if there is an exception for the speaker's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Issues with IEMs

1. How to enable IEMs feature and set the volume?

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enable IEMs.
[[self.trtcCloud getAudioEffectManager] enableVoiceEarMonitor:YES];
// Set the volume of IEMs.
[[self.trtcCloud getAudioEffectManager] setVoiceEarMonitorVolume:volume];
```

Note:

The IEMs can be set in advance without having to monitor audio routing changes. Once headphones are connected, the IEMs feature will automatically take effect.

2. The IEMs feature does not take effect after enabled.

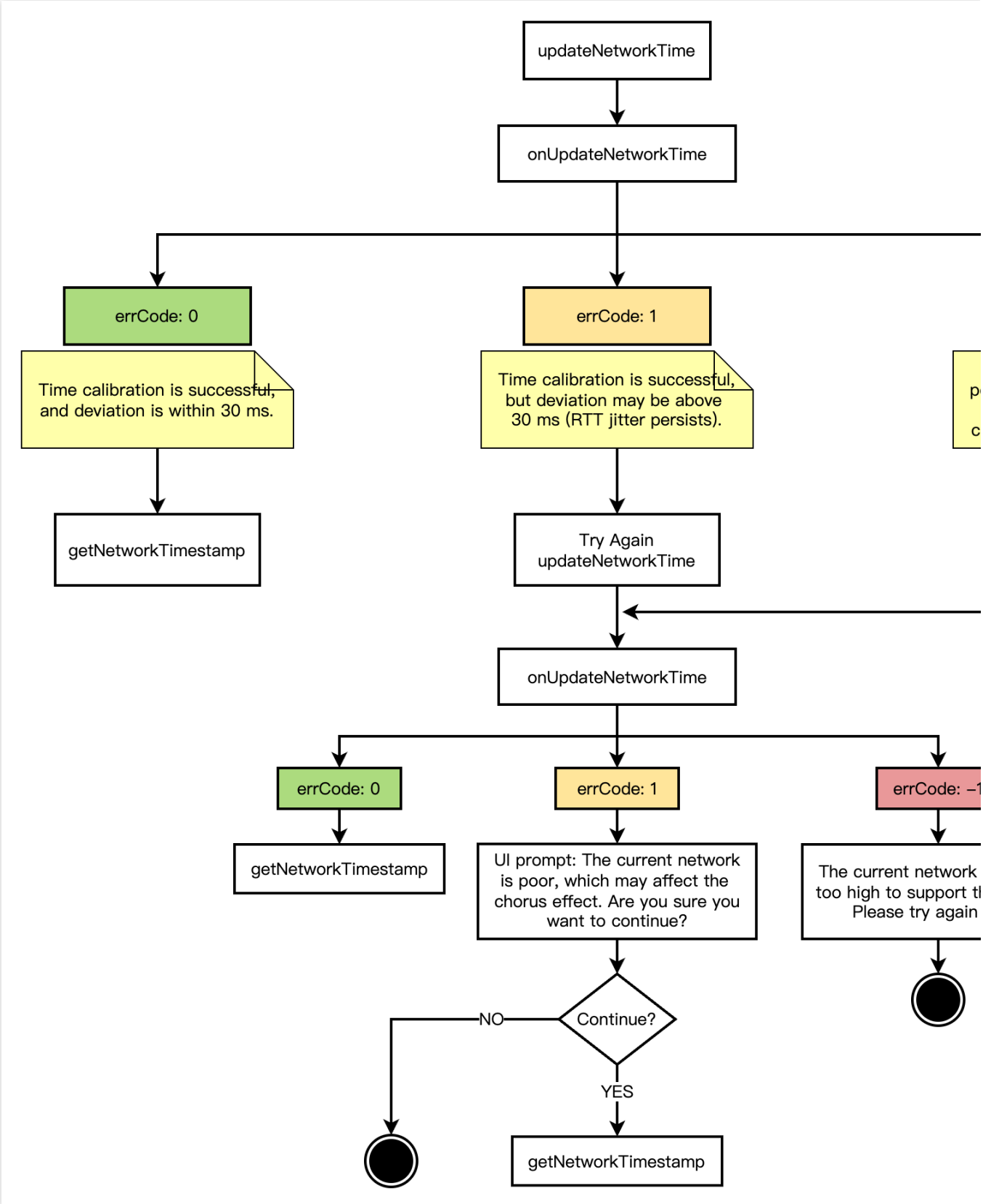
©2013-2022 Tencent Cloud. All rights reserved.

Due to the high hardware delay of Bluetooth headphones, it is recommended to prompt the anchor to wear wired headphones on the user interface. Also, it should be noted that not all smartphones will achieve excellent IEMs effect after this feature is enabled. TRTC SDK has already blocked this feature on some smartphones with poor effect.
3. High IEM delay
Check if Bluetooth headphones are in use. Due to the high hardware delay of Bluetooth headphones, wired headphones are recommended. Additionally, you can try improving the issue of high IEM delay by enabling hardware IEM through the experimental API `setSystemAudioKitEnabled`. Hardware IEMs have better performance and lower delay. Software IEMs have higher delay but better compatibility. Currently, for Huawei and VIVO devices, SDK defaults to hardware IEMs. Other devices default to software IEMs. If there are compatibility issues with hardware IEMs, contact us to configure forced use of software IEMs.

Issues with NTP sync

1. NTP time sync finished, but result maybe inaccurate.
NTP sync is successful, but the deviation may still be more than 30 milliseconds. This indicates a poor client network environment with persistent RTT jitter.
2. Error in AddressResolver: No address associated with hostname
NTP sync has failed, possibly due to a temporary exception in local ISP DNS resolution under the current network environment. Try again later.
3. NTP service retry processing logic.

©2013-2022 Tencent Cloud. All rights reserved.



Issues with real-time chorus usage

1. **Why does the lead singer in real-time chorus scenarios need to use dual-instance streaming?**

©2013-2022 Tencent Cloud. All rights reserved.

In real-time chorus scenarios, to minimize end-to-end delay and achieve sync between vocals and accompaniment, a common approach is to use dual instances at the lead singer's end to separately upload vocal and accompaniment streams, while other chorus participants only upload their vocal streams and locally play the accompaniment. In this case, each chorus participant needs to subscribe to the lead singer's vocal stream, while refraining from subscribing to the lead singer's music stream. This setup can only be achieved by implementing dual-instance separate streaming.

2. **Why is it recommended to enable mixing pushback in real-time chorus scenarios?**

Having the audience pull multiple single streams at the same time is likely to result in misalignment between multiple vocal streams and accompaniment streams. Pulling a mixed stream can ensure absolute alignment of all streams and reduce downstream bandwidth.

3. **What are the uses of SEI in real-time chorus scenarios?**

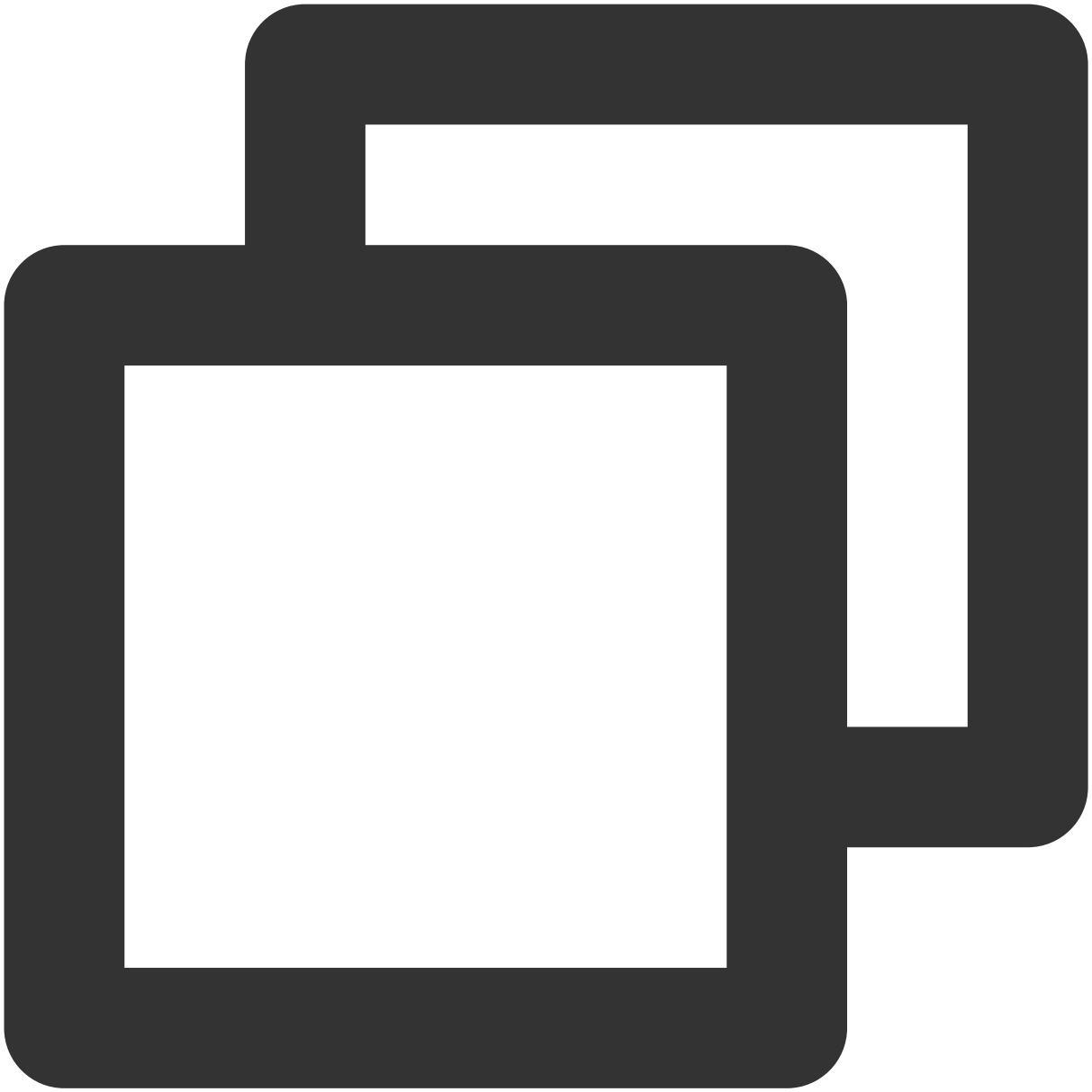
Transmitting accompaniment music progress, for lyric sync on the audience's end.

Transparently transmitting single stream volume through a mixed stream, for display as sound waves on the listener's end.

4. **Loading accompaniment music takes a long duration, causing significant playback delay?**

Loading network music resources via the SDK incurs a certain delay. It is recommended to initiate music pre-loading before starting playback.

©2013-2022 Tencent Cloud. All rights reserved.

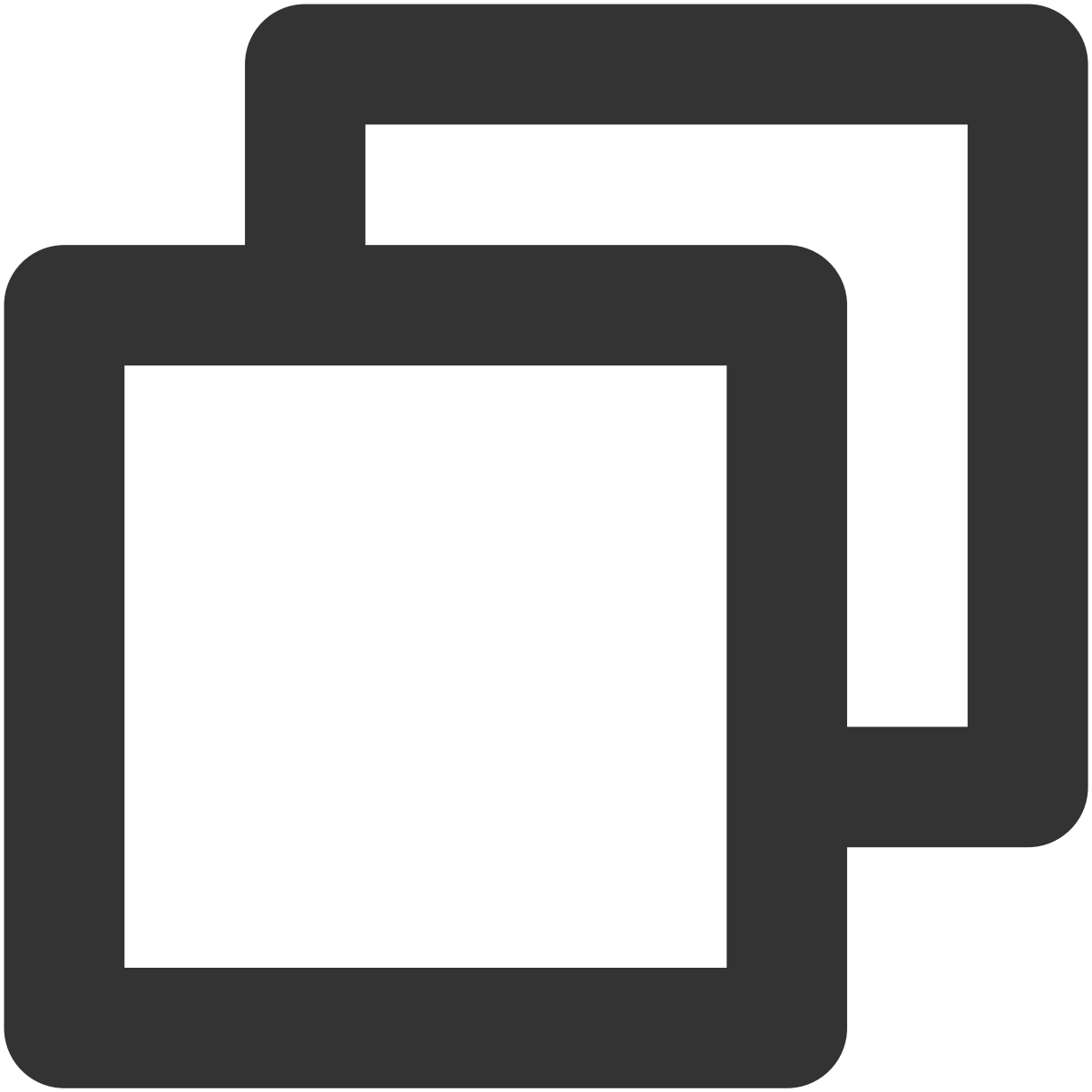


```
[[self.trtcCloud getAudioEffectManager] preloadMusic:musicParam onProgress:nil onEr
```

5. **When singing along with accompaniment, the vocals are barely audible. Is the music overwhelming the vocals?**

If the default volume settings result in the accompaniment overwhelming the vocals, it is recommended to adjust the volume balance between the music and vocals accordingly.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the local playback volume of a piece of background music.
[[self.trtcCloud getAudioEffectManager] setMusicPlayoutVolume:self.musicId volume:v
// Set the remote playback volume of a specific background music.
[[self.trtcCloud getAudioEffectManager] setMusicPublishVolume:self.musicId volume:v
// Set the local and remote volume of all background music.
[[self.trtcCloud getAudioEffectManager] setAllMusicVolume:volume];
// Set the volume of voice capture.
[[self.trtcCloud getAudioEffectManager] setVoiceVolume:volume];
```

©2013-2022 Tencent Cloud. All rights reserved.

Live Show Streaming Scenario Solution

Last updated：2024-07-25 17:02:41

Scene Introduction

The live showroom scenario is a video interaction scenario under the social entertainment mode. It supports multi-user video mic connect, making it easier to engage users in mic connect, thus boosting user spending willingness and stickiness. Moreover, the live showroom also supports cross-room competition between anchors from different rooms, further enhancing the fun of live streaming. Cross-room competition with latency below 300 ms, supporting audience and anchor mic connecting, smooth on/off mic switching, meets the high-frequency interaction demands of the live showroom scenario. The intelligent beauty feature also meets the personalized needs of anchors, making live streaming more appealing.

©2013-2022 Tencent Cloud. All rights reserved.

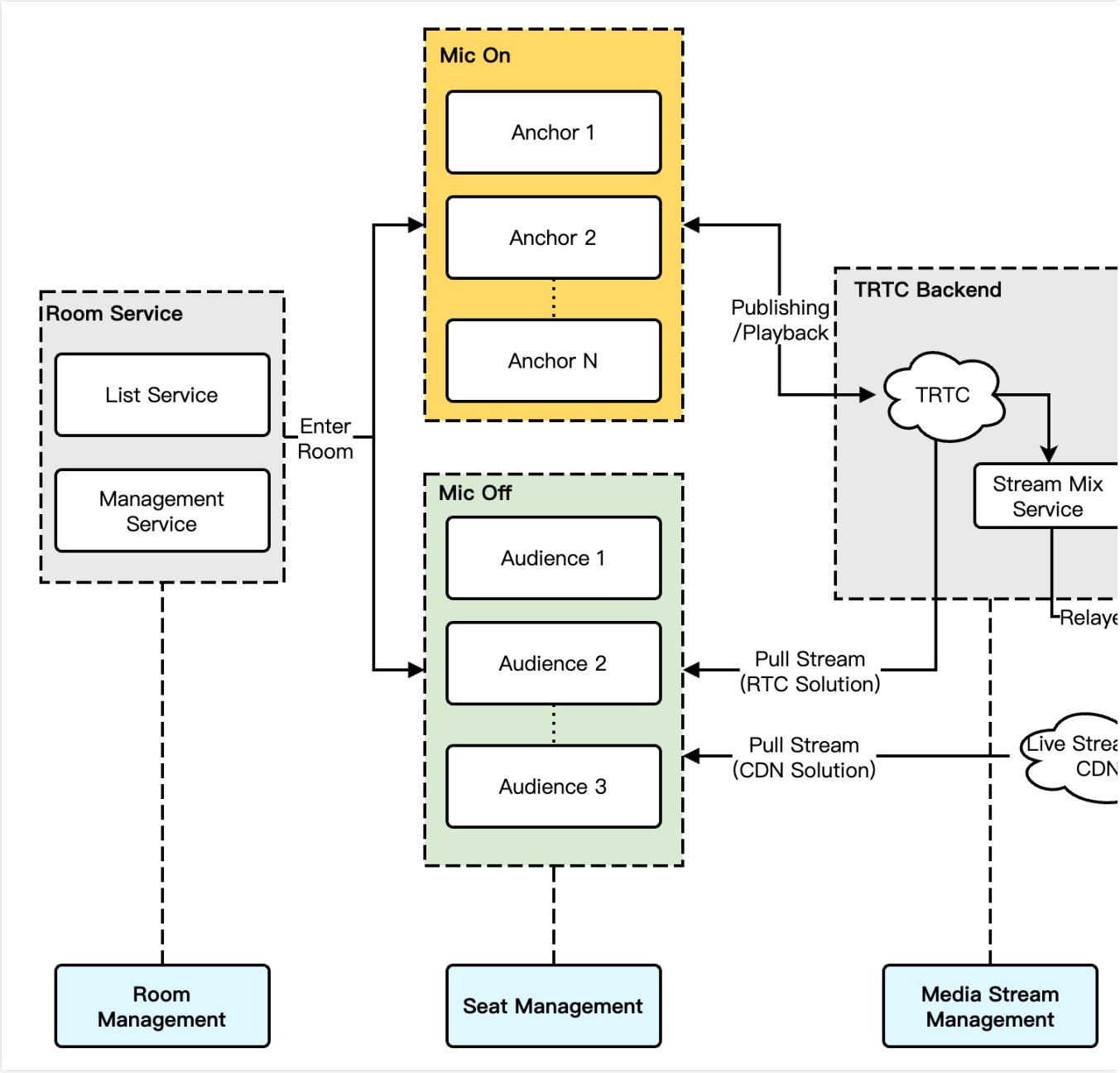
Implementation Scheme

Typically, implementing a complete live showroom scenario involves several functional modules: Room Management, Seat Management, Media Stream Management, On-Cloud Recording, etc. The key actions and feature points under each feature module are shown in the table below. Each functional module will be introduced individually to provide a comprehensive understanding of the functionalities required for building a live showroom scenario.

Feature Module	Key Actions and Feature Points
Room management	Room list, create a room, enter a room, exit a room, and terminate a room.
Seat Management	Request to speak, become a listener, invite a listener to speak, remove a speaker, and mute a speaker.
Media Stream Management	RTC Real-Time Interaction Solution
On-cloud recording	TRTC on-cloud recording.

©2013-2022 Tencent Cloud. All rights reserved.

The overall business architecture of the live showroom scenario is shown in the figure below. The room owner creates a room, and users can choose to enter rooms that interest them. Users who enter the room can become speakers to participate in audio and video interactions with the anchor via the mic. Typically, due to compliance requirements, the audio and video content in the room needs to be recorded and submitted for review.



Note:

Relayed push and recording reviews can be configured to use either target single streams or mix streams based on specific business requirements.

The recording and review process can be implemented using either RTC Recording Review or CDN Recording Review, depending on specific business requirements.

©2013-2022 Tencent Cloud. All rights reserved.

Room management

The Room Management Module is primarily responsible for maintaining the room list and includes the following features:
Create Room: After users log in to the business system, they can create a room. The room list needs to be updated after a room is created.
Enter a Room: Users can choose to enter an existing room. Upon entering, the current list of room members should be updated.
Exit a Room: Users can choose to exit the current room. Upon exiting, the current list of room members needs to be updated with a delete operation.
Destroy a Room: After all users exit the room, it needs to be destroyed. Upon destruction, the room list needs to be updated with a delete operation.
Note:
Room Management is a necessary module for implementing a live showroom, but it is not the main functional module. It can be implemented in conjunction with the business system and IM&TRTC SDK, see Voice Chat Room - Room Management for details.

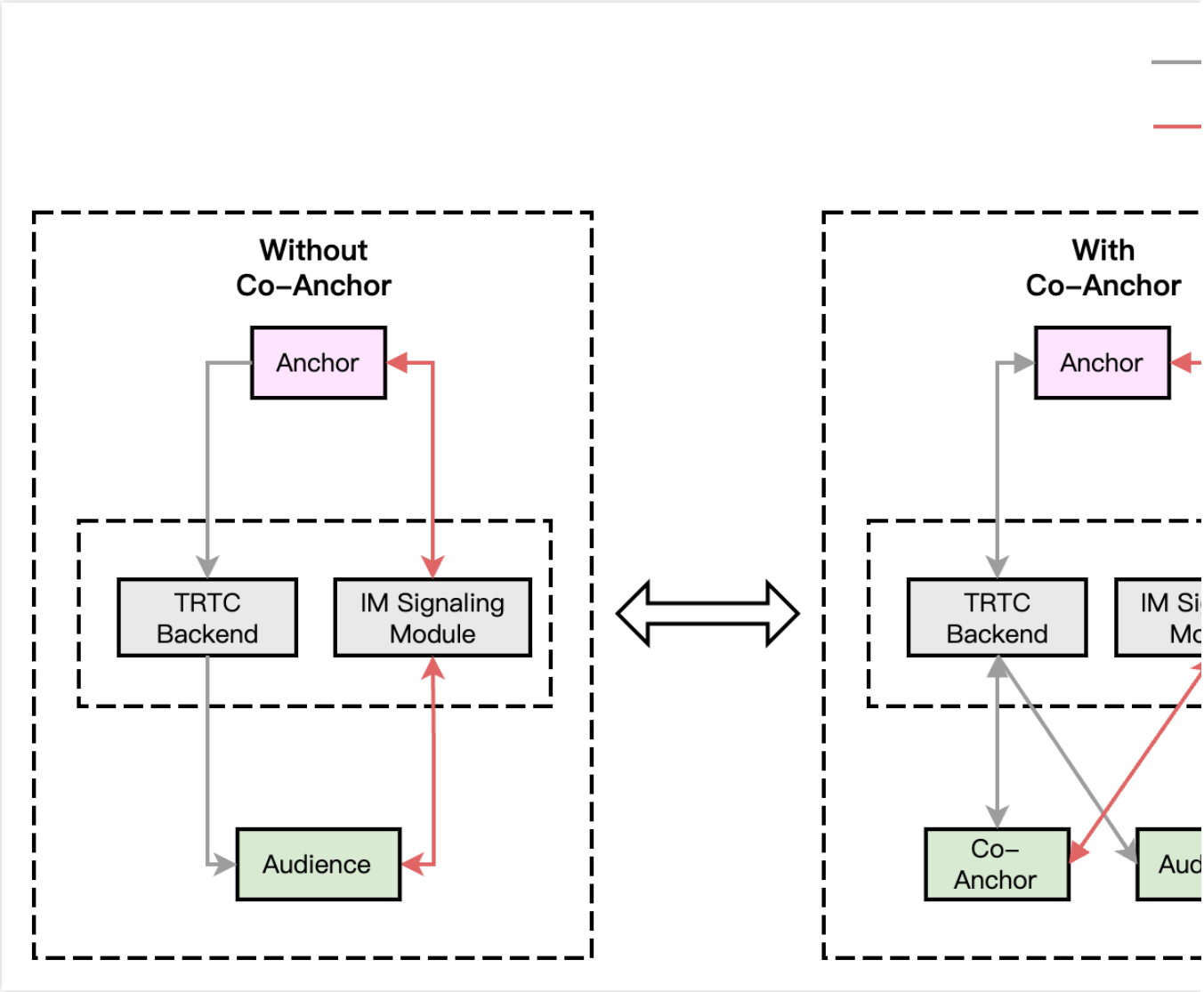
Seat Management

Seats in a live streaming room are generally ordered and limited. Seat Management is primarily responsible for defining the number of seats in a room based on the business scene, as well as managing the status of all seats in the current room. Seat Management mainly includes: request to speak, become a listener, invite a listener to speak, remove a speaker, and mute a speaker.
After users enter a room, only idle seats can be applied for.
After the room owner approves a user to become a speaker, the seat status needs to be changed to non-idle.
After the user stops streaming and becomes a listener, the seat status needs to be reset.
The room owner has the authority to lock the seat, invite a listener to speak, remove a speaker, mute a speaker, etc.
Note:
Seat Management is a necessary module for implementing a live showroom, but it is not the main functional module. It can be implemented in conjunction with the business system and IM&TRTC SDK, see Voice Chat Room - Seat Management for details.

Media Stream Management

For ordinary live showroom scenarios, we recommend the RTC real-time interaction solution: both anchors and audience use the RTC protocol for publishing/playback, minimizing end-to-end latency and ensuring a smoother experience for the audience when joining and leaving the mic, without abrupt changes such as image fast-forwarding or rewinding. Taking the example of multi-person co-anchoring live streaming interactions, the main architecture of live showroom in a pure RTC publishing/playback scenario is shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



The overall process of this solution is as follows:

1. Both the anchor and audience connect through the signaling module, which is mainly responsible for controlling the live streaming process and synchronizing the live streaming status.
2. Regardless of whether there are mic-connecting audiences or not, both the anchor and the audience use the TRTC audio and video cloud service for publishing/playback.
3. After the audience requests to mic connect with the anchor, the signaling module will notify the anchor and synchronize the personal information of the co-speakers.
4. Once the anchor accepts the mic connection request, the mic-connecting audience starts streaming, and all members in the room receive stream update notifications and pull the audio-video stream of the mic-connecting audience.
5. When a mic-connecting audience member requests to disconnect, they stop streaming. All members in the room will receive stream update notifications and stop pulling that audience's audio and video stream.

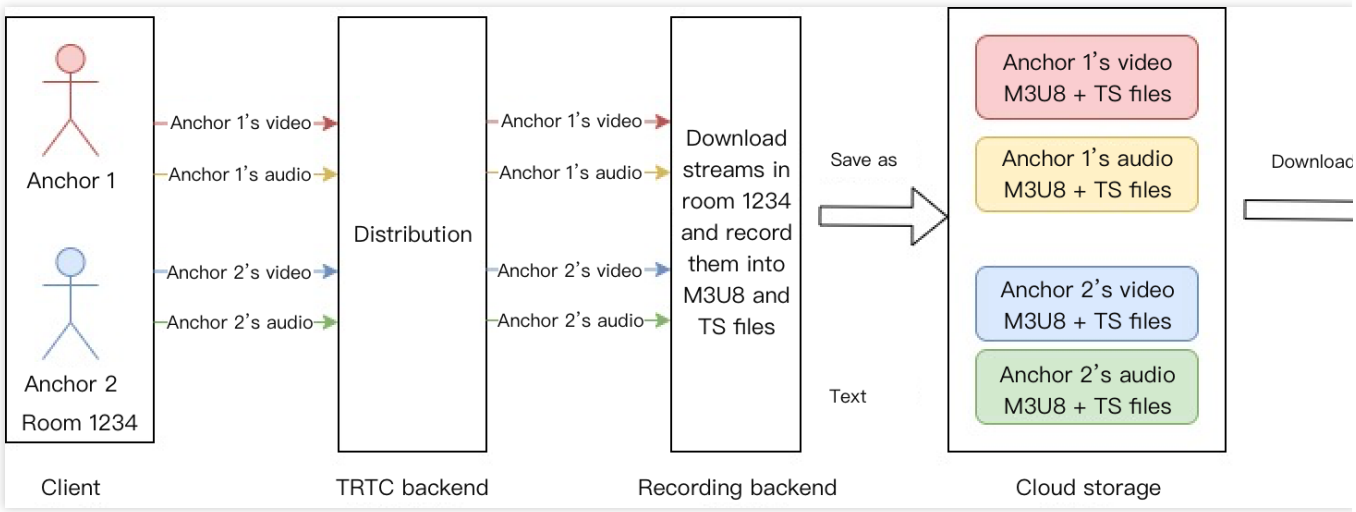
Note:

©2013-2022 Tencent Cloud. All rights reserved.

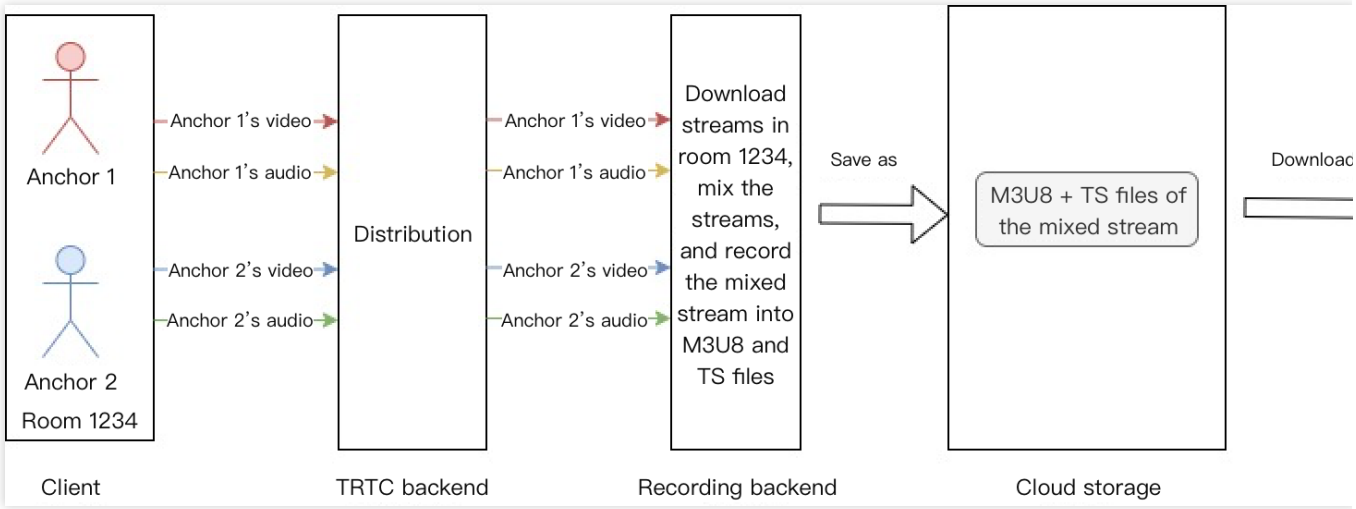
The signaling module can be a custom-developed signaling channel, and it is also recommended to use Instant Messaging (IM) for signaling interaction.

On-cloud recording

TRTC's newly upgraded on-cloud recording does not depend on cloud streaming services. It does not require a relayed push for cloud live streaming and uses TRTC's internal real-time recording cluster for audio and video recording, offering a more comprehensive and unified recording experience.
Single Stream Recording: With TRTC's on-cloud recording feature, you can record the audio and video stream of each user in the room into separate files.



Mix Stream Recording: Record the audio-video media streams of the same room as a single file.



Note:

For a detailed introduction and activation guide to TRTC On-Cloud Recording, see On-Cloud Recording.

©2013-2022 Tencent Cloud. All rights reserved.

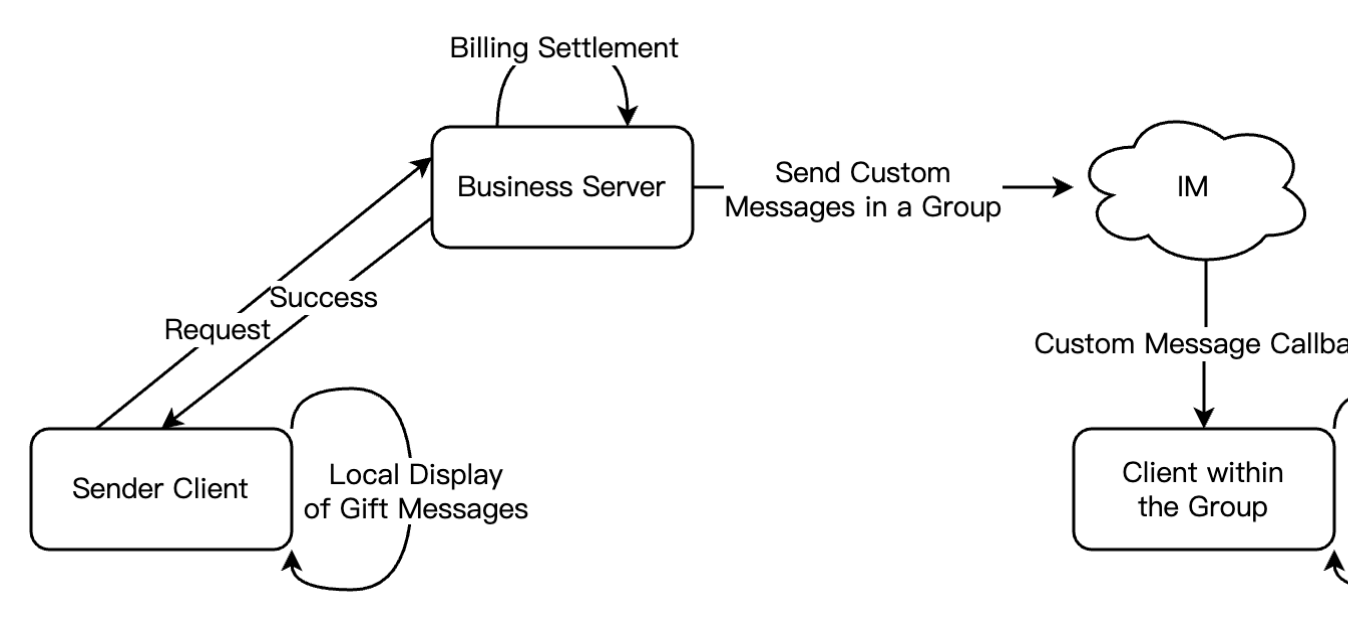
In the live showroom scenario, a common approach for recording is the mixed-stream recording solution, while the single stream recording solution can be chosen if there is a need for single stream review of the anchor or the mic-connecting audience.

Key Business Logic

Gift and Like Messages

In the live showroom scenario, gifts and likes are common ways of interaction. Audiences can express their love and support for the anchor by giving gifts and likes, and the anchor can also earn revenue from them. Below, we will introduce the implementation of gift and like messages based on Tencent Cloud Instant Messaging.

Gift message

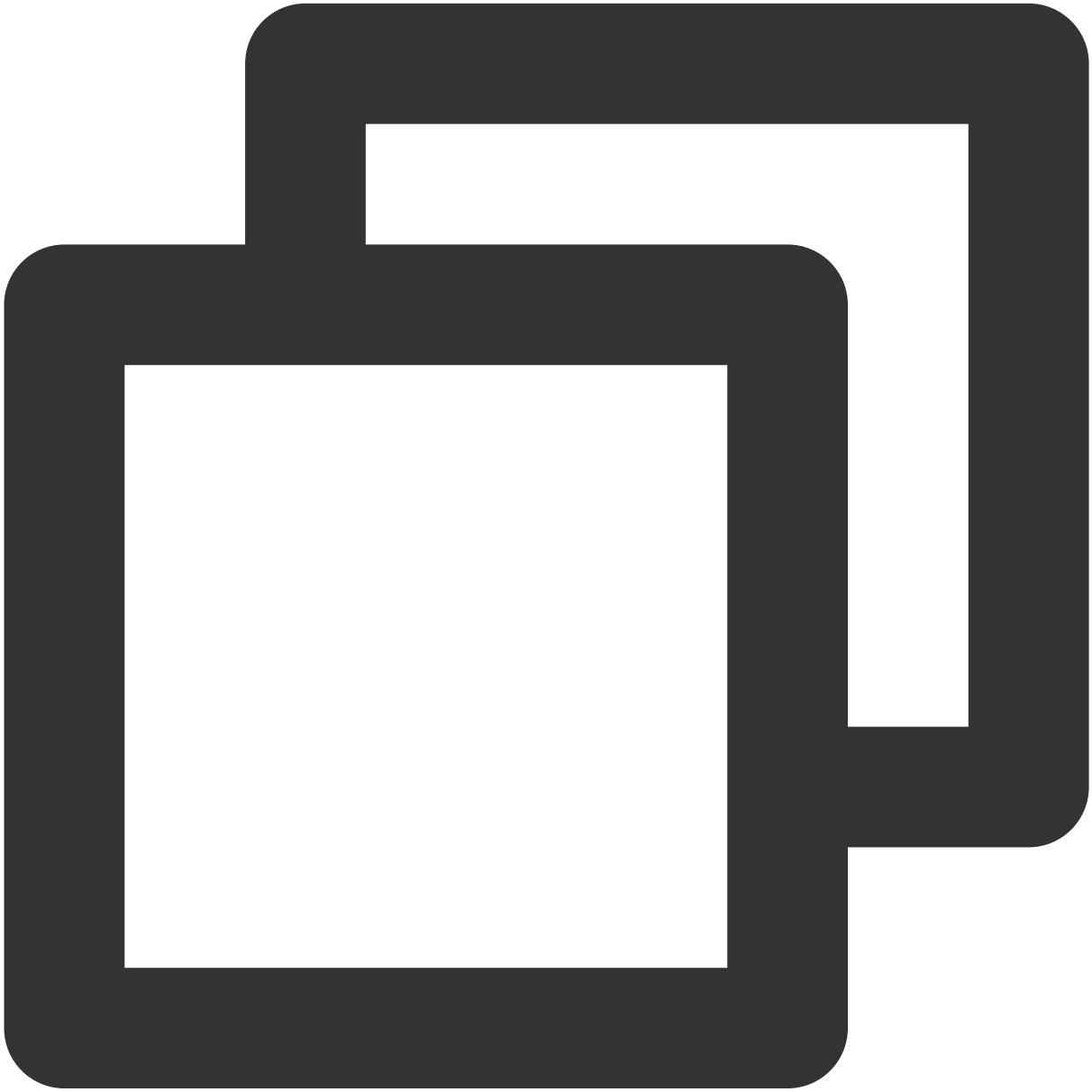


1. Non-persistent connection requests from the client to their business server involve gift billing logic.
2. After billing, the sender directly sees XXX gave XXX a gift (to ensure the sender sees the gift they sent themselves; when the message volume is large, it may trigger an abandonment strategy).
3. After billing is settled, the client calls the server API Sending Custom Messages in a Group (gift).
4. If encountering scenarios of rapid gift giving, you need to merge the messages:
If users directly select the number of gifts, such as choosing 99 gifts, the system should send a single message with the parameter set to 99 gifts.
If the gifts are in a combo and it's uncertain how many there will be, the business backend can merge every 20 (quantity adjustable) or send one for combos lasting over 1 second. Following this logic, for example, for a combo of

©2013-2022 Tencent Cloud. All rights reserved.

99 gifts, only 5 messages would need to be sent after optimization.

An example request package for server API sending custom messages in a group is as follows, where MsgBody supports customizing message elements.

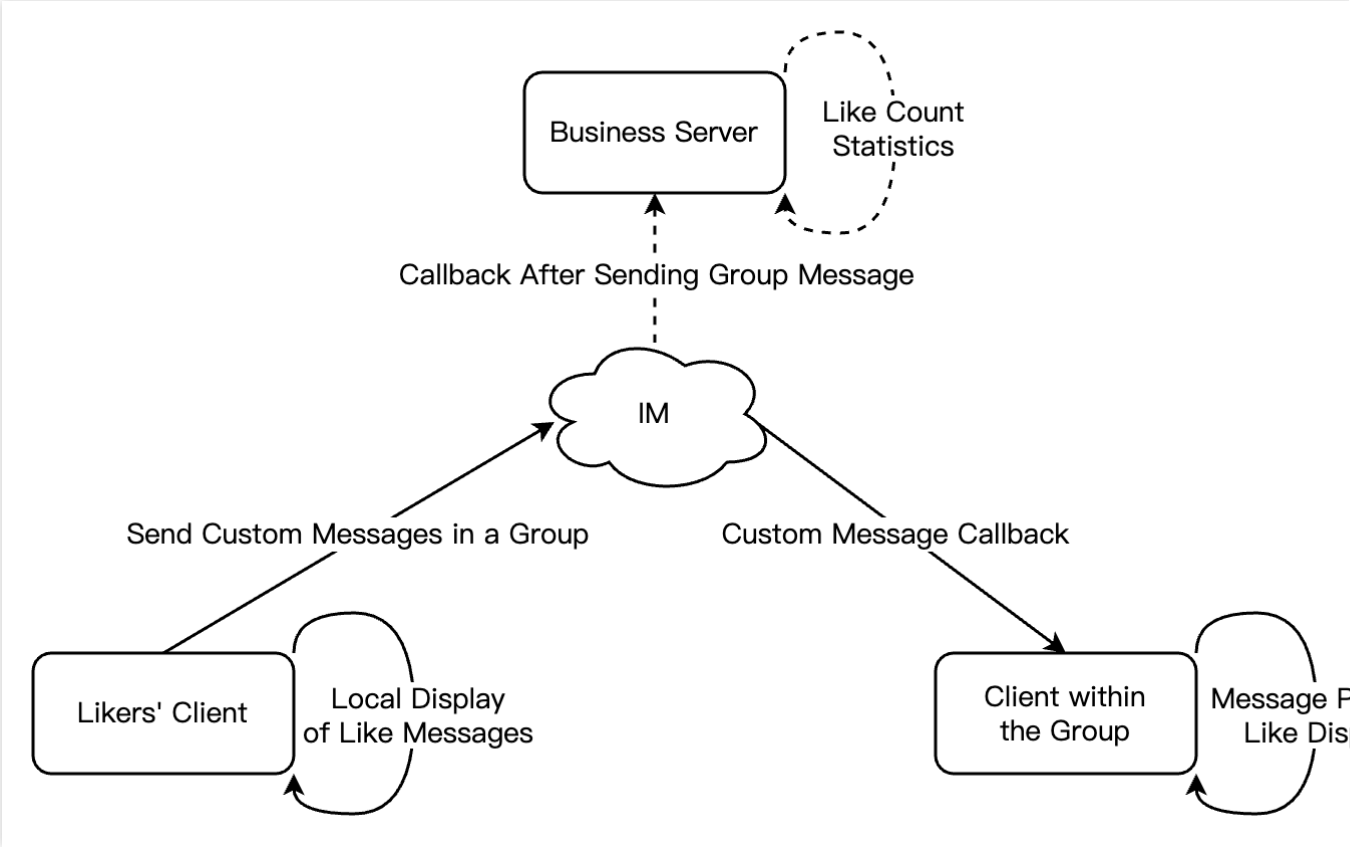


```
{
    "GroupId": "@TGS#12DEVUDHQ",
    "Random": 2784275388,
    "MsgPriority": "High",  // The priority of the message. Gift messages should be
    "MsgBody": [
        {
            "MsgType": "TIMCustomElem",
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            "MsgContent": {
                // type: gift type; giftUrl: gift resource URL; giftName: gift name
                "Data": "{\\"cmd\\": \\"gift_msg\\", \\"msg\\": {\\"type\\": 1, \\"
            }
        }
    ]
}
```

Like message



1. Likes do not involve billing and are typically sent directly from the client using sending custom messages in a group.
2. For like messages that need to be counted on the server, after traffic throttling is performed on the client, likes on the client are counted, and like messages within a short period of time are merged into one. The business server gets the count of likes in the callback after sending a group message for statistics.
3. For like messages that do not require counting, follow the same logic as step 2. Throttle the likes messages on the client side and send them. There is no need to count them in the callback after sending the group message.

Note:

Please set important messages to high priority (e.g., gift messages) and high-frequency but not important messages to low priority (e.g., like messages).

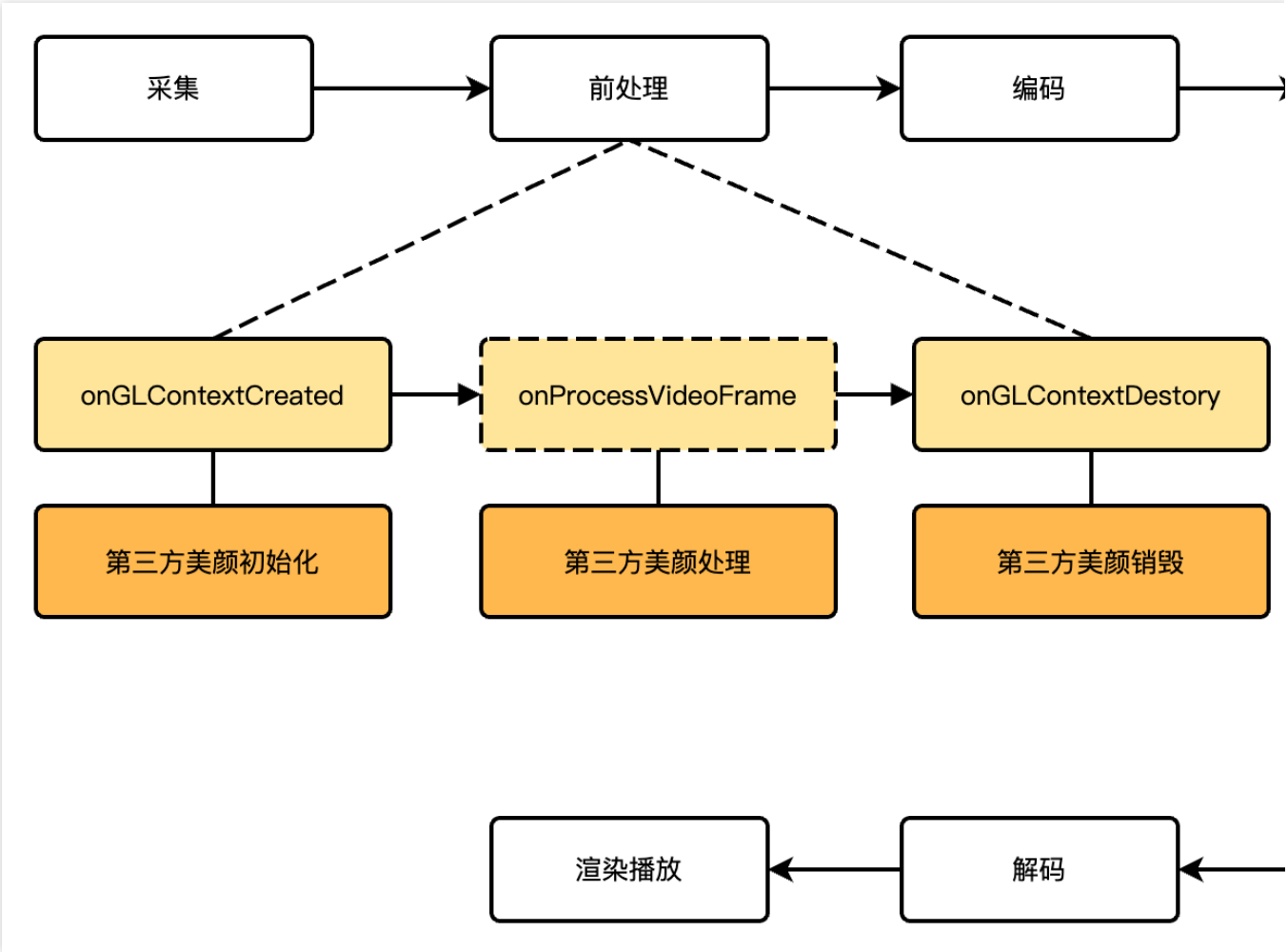
©2013-2022 Tencent Cloud. All rights reserved.

Specific implementation steps for live streaming interaction features (such as likes, gifts, and on-screen comment chat) can be seen in Quick Integration Guide - Live Streaming Interactive Messages.

Integrating Beauty Effect

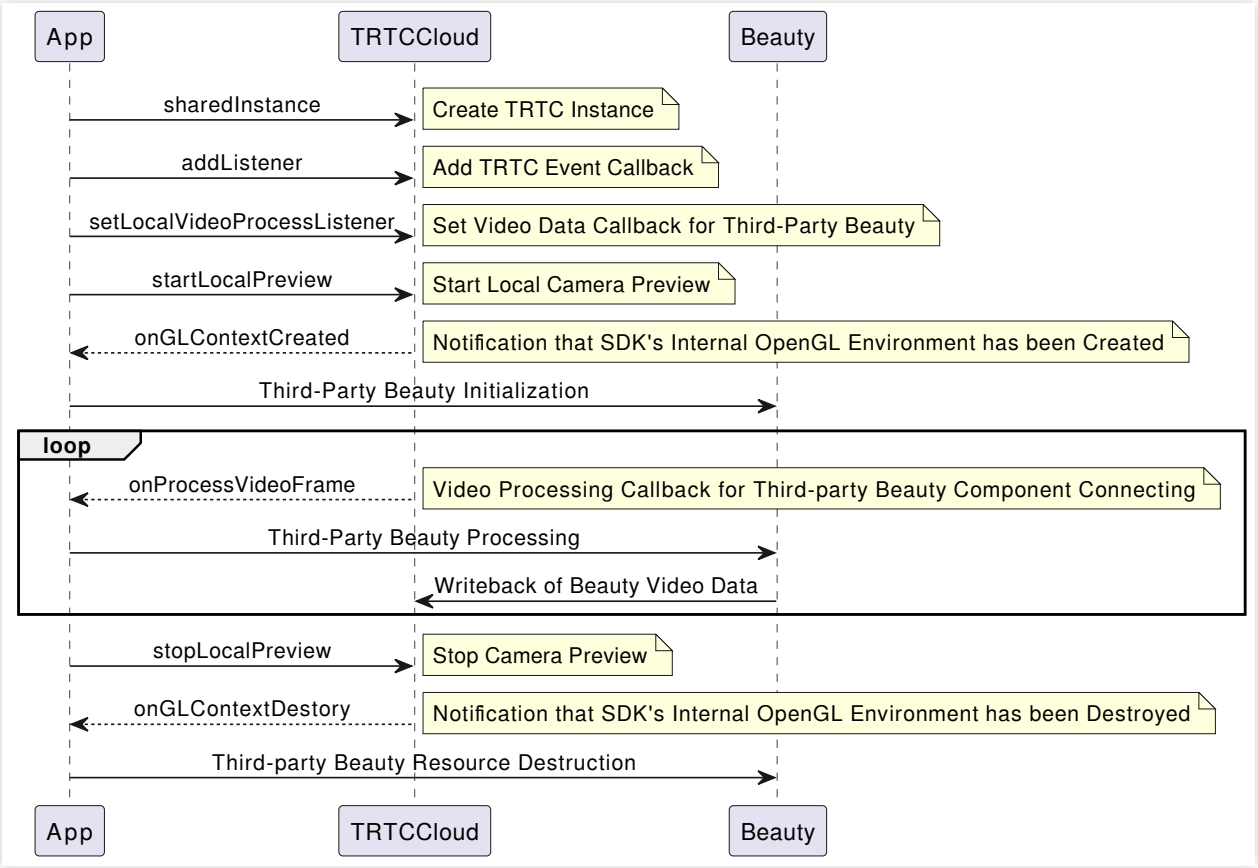
In the live showroom scenario, the beauty effect is a frequently used feature. It not only improves the beauty of the anchor but also adds fun to live interaction through various sticker effects. TRTC supports the integration of Tencent Effect SDK as well as the integration of mainstream third-party beauty effect products in the market, such as Volcano Beauty, FaceUnity, etc.

Beauty Effect Integration Process



API Call Sequence

©2013-2022 Tencent Cloud. All rights reserved.



Comparison of Beauty Products

Beauty Type	Beauty Effect	Access Costs	Fees	Virtual AI Digital Human	Support Terminal
Tencent Effect SDK	The basic effect is good, advanced effect for big eyes/slim faces is significant.	Low	Moderate	Supported	Android/iOS/PC/Flutter/Web/Mini Program
FaceUnity Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are average.	Moderately high	Moderate	Supported	Android/iOS/PC/Untiy

©2013-2022 Tencent Cloud. All rights reserved.

Volcano Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are relatively good.	Moderately high	Relatively High	Supported	Android/iOS/PC/Linux

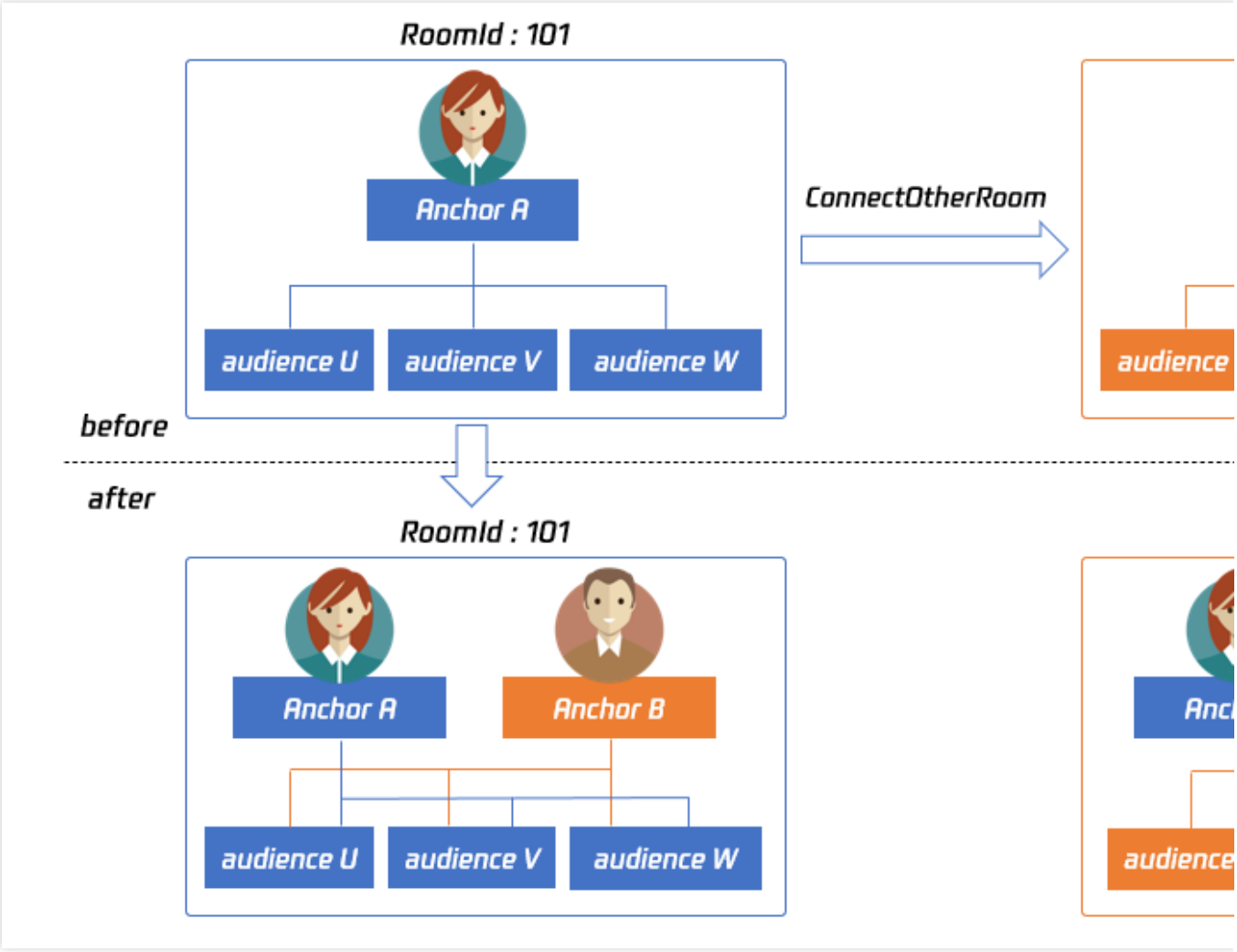
Cross-Room Competition

Cross-room competition between anchors is a common gameplay in live showroom scenarios. It enhances the fun of interactive live streaming and stimulates the audience's desire to rank and give gifts. TRTC supports multiple rooms and cross-room competition between multiple anchors. Below are the specific implementation methods.

1. **How It Works**

By default, only users in the same room can have audio and video calls, and the audio and video streams between different rooms are isolated. Through the cross-room competition, the audio and video streams of an anchor in another room can be published in the current room, while the audio and video streams of the current anchor will also be published in the target anchor's room. This allows anchors in different rooms to share audio and video streams across rooms, enabling audiences in each room to watch the audio and video of both anchors.

©2013-2022 Tencent Cloud. All rights reserved.



The figure above shows the main process of cross-room competition. For example: After anchor A in room 101 establishes a cross-room call with anchor B in room 102 using `ConnectOtherRoom()` :

Users in room 101 will receive two event callbacks from anchor B: `onRemoteUserEnterRoom(B)` and `onUserVideoAvailable(B,true)` . Therefore, users in room 101 can subscribe to the audio and video of anchor B.

Users in room 102 will receive two event callbacks from anchor A: `onRemoteUserEnterRoom(A)` and `onUserVideoAvailable(A,true)` . Therefore, users in room 102 can subscribe to the audio and video of anchor A.

Note:

Both local and peer users participating in cross-room competition must be in the anchor role and must have audio/video uplink capabilities.

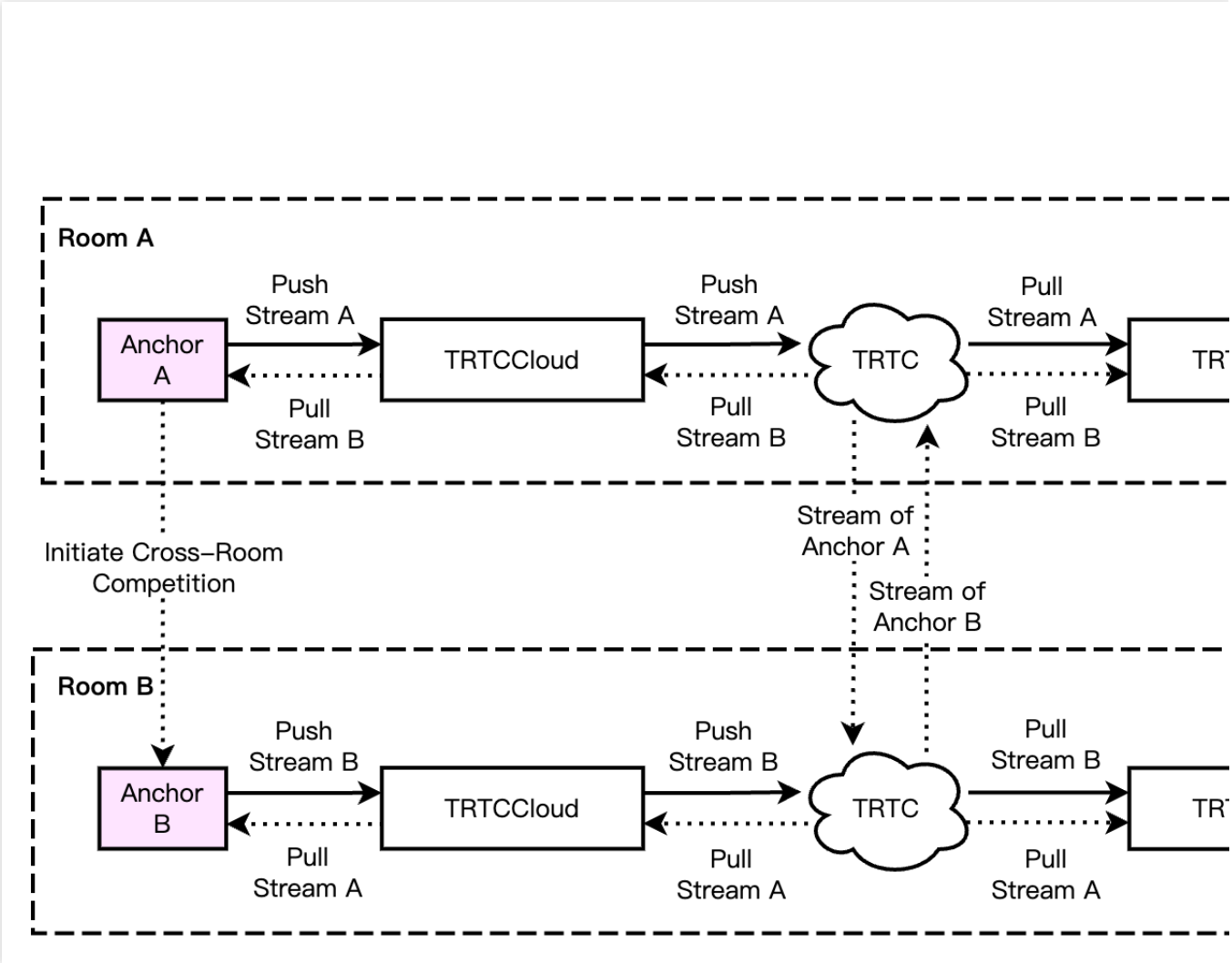
Cross-room mic-connection PK with multiple room anchors can be achieved by calling `ConnectOtherRoom()` multiple times. Currently, a room can connect with up to three other room anchors at most, and up to 10 anchors in a room can conduct cross-room mic-connection competition with anchors in other rooms.

©2013-2022 Tencent Cloud. All rights reserved.

TRTC cross-room competition can also be achieved by `createSubCloud()` to create a sub-instance and join the publishing/playback of another room. Currently, there is no limit to the number of sub-instances, facilitating the future expansion of business scenes involving PK between multiple rooms or anchors.

2. **Real-Time Interactive Cross-Room Competition Process**

In a pure RTC scenario, the cross-room competition process is straightforward. Anchors and cross-room competition anchors mutually pull RTC single streams, and the audience simultaneously pulls the RTC single streams of both anchors and cross-room competition anchors. The audience can independently control the subscription logic of the media streams of anchors and cross-room connecting anchors. The real-time interactive cross-room call process is shown in the diagram below.



Note:

In real-time interactive cross-room competition scenes, audiences in the room can independently control the logic of subscribing to the media streams of cross-room connecting anchors, or it can be changed by the room owner to <1>change the uplink capability of a cross-room anchor in their rooms<1>.

©2013-2022 Tencent Cloud. All rights reserved.

Scene Approach

Single-Anchor Live Streaming

A live streaming room with only one anchor is called single-anchor live streaming. In this scenario, the room owner is the only anchor. The audience can join the live stream, watch the live streaming, send messages, and give gifts to the room owner.

Multi-Person Co-Anchoring Live Streaming

Multi-person co-anchoring live streaming refers to a scenario where multiple anchors engage in real-time audio and video interactions within the live streaming room. The room owner can invite a listener to speak and control the seats; audiences can also request to speak to interact with the anchor.

Cross-Room Competition Live Streaming

In the live streaming room, to enhance the atmosphere and quickly attract followers, the room owner can invite another anchor from a different live room to engage in mic connecting or online PK. The audience in the mic-connected live streaming room can simultaneously watch the interaction between the two anchors and send gifts based on their performance, or quickly switch between live streaming rooms to vote for different room owners. This is a typical scenario for video competition live streaming.

Live Shopping

Live shopping combines e-commerce with video interactive live streaming. In the live streaming room, the anchor introduces products to the audience and provides product lists and links; the audience can click the link to place orders quickly for products they like. During the live streaming, the audience can become a speaker to interact with the room owner in real-time, such as asking about product details, negotiating prices, and sharing their experience of using the products; the room owner can also engage in competition live streaming with another room's owner, showcasing their products to inspire the audience's purchasing enthusiasm and add fun to the live streaming.

Alternative Solutions

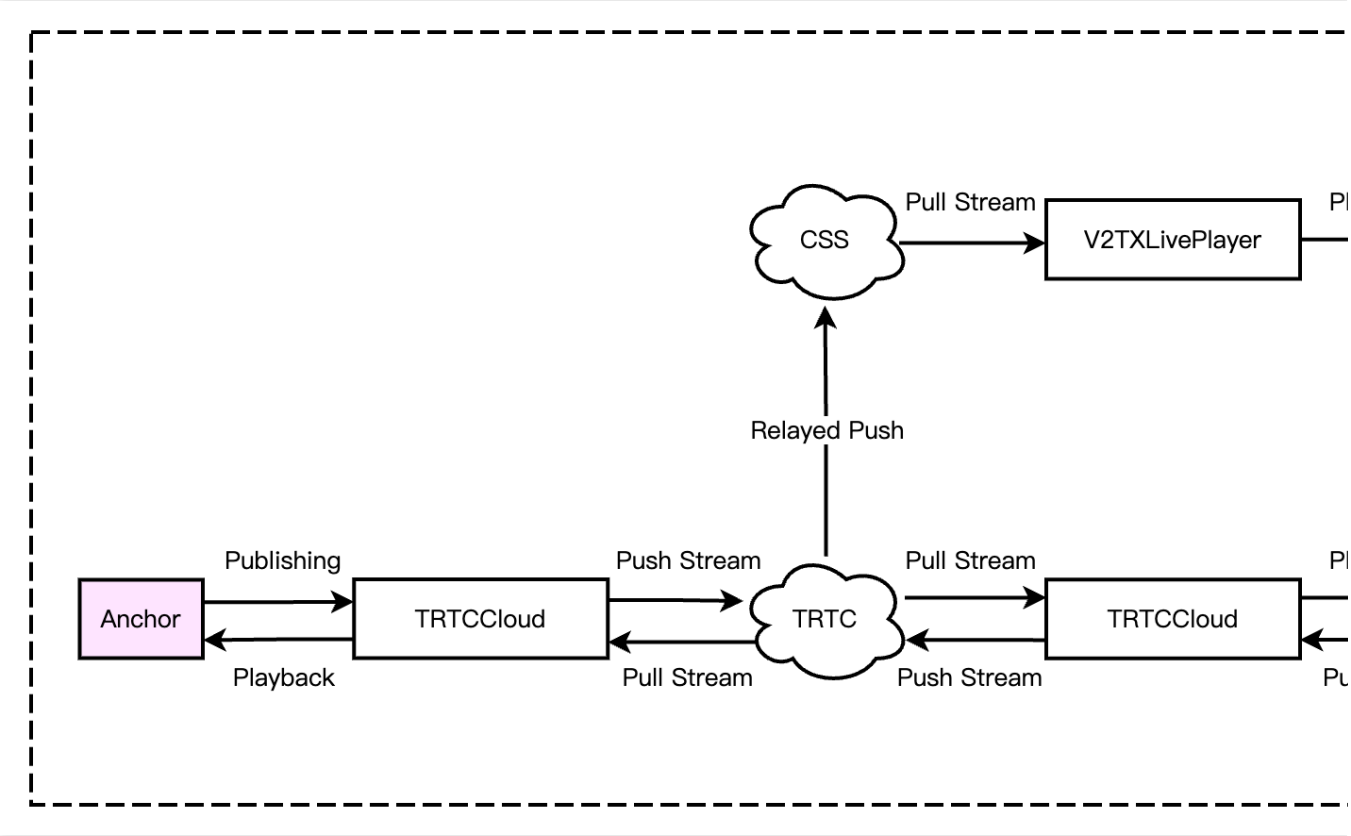
In addition to the recommended RTC Real-Time Interaction Solution, live showroom scenarios usually have an alternative solution: RTC CDN Live Streaming Solution.

RTC CDN Live Streaming Solution

The anchor uses the RTC protocol for streaming and relayed push of Tencent Cloud Streaming Services or third-party live streaming. General audiences pull the CDN stream to watch while mic-connecting audiences engage in interactive co-anchoring by switching to the RTC protocol for streaming. This solution is a commonly used compromise, with

©2013-2022 Tencent Cloud. All rights reserved.

higher latency for both watching and mic connecting on the audience side. However, it offers advantages in terms of cost-effectiveness and audience scalability. The publishing/playback architecture is shown in the diagram below:



The overall process of this solution is as follows:

1. The anchor enters the TRTC room, streams via the RTC protocol, and relays to Cloud Streaming Services.
2. Non-mic audience pull the CDN accelerated stream for watching through `V2TXLivePlayer`.
3. Audiences can request to speak and become the mic-connecting audiences. They stop the CDN streaming and switch to the RTC protocol for publishing/playback.
4. Audiences off the mic become general audiences. They stop the RTC protocol publishing/playback and switch to the CDN streaming.

CDN playback addresses support multiple protocols such as RTMP/FLV/HLS/WebRTC, with splicing rules detailed in Splicing Playback URL.

Different live streaming playback protocols have varying compatible platforms, play delays, and billing rules. See the table below for details:

Live Streaming Protocol	Advantage	Disadvantage	Playback Latency
FLV	Mature, suited for high-concurrency scenarios	Requires integration of SDK for playback	2s - 3s

©2013-2022 Tencent Cloud. All rights reserved.

RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1s - 3s
HLS(M3U8)	Well supported on mobile browsers	High latency	10s - 30s
WebRTC	Lowest latency	Requires integration of SDK for playback	< 1s

Note:

Tencent Cloud Streaming Services supports multiple playback protocols. You can choose the appropriate pull stream solution based on your business needs. For example, for delay-sensitive businesses, LEB pulling stream is recommended.

HLS and WebRTC playback protocols support the adaptive bitrate feature, which allows smooth switching of playback bitrate under different network conditions. See Adaptive Bitrate for details.

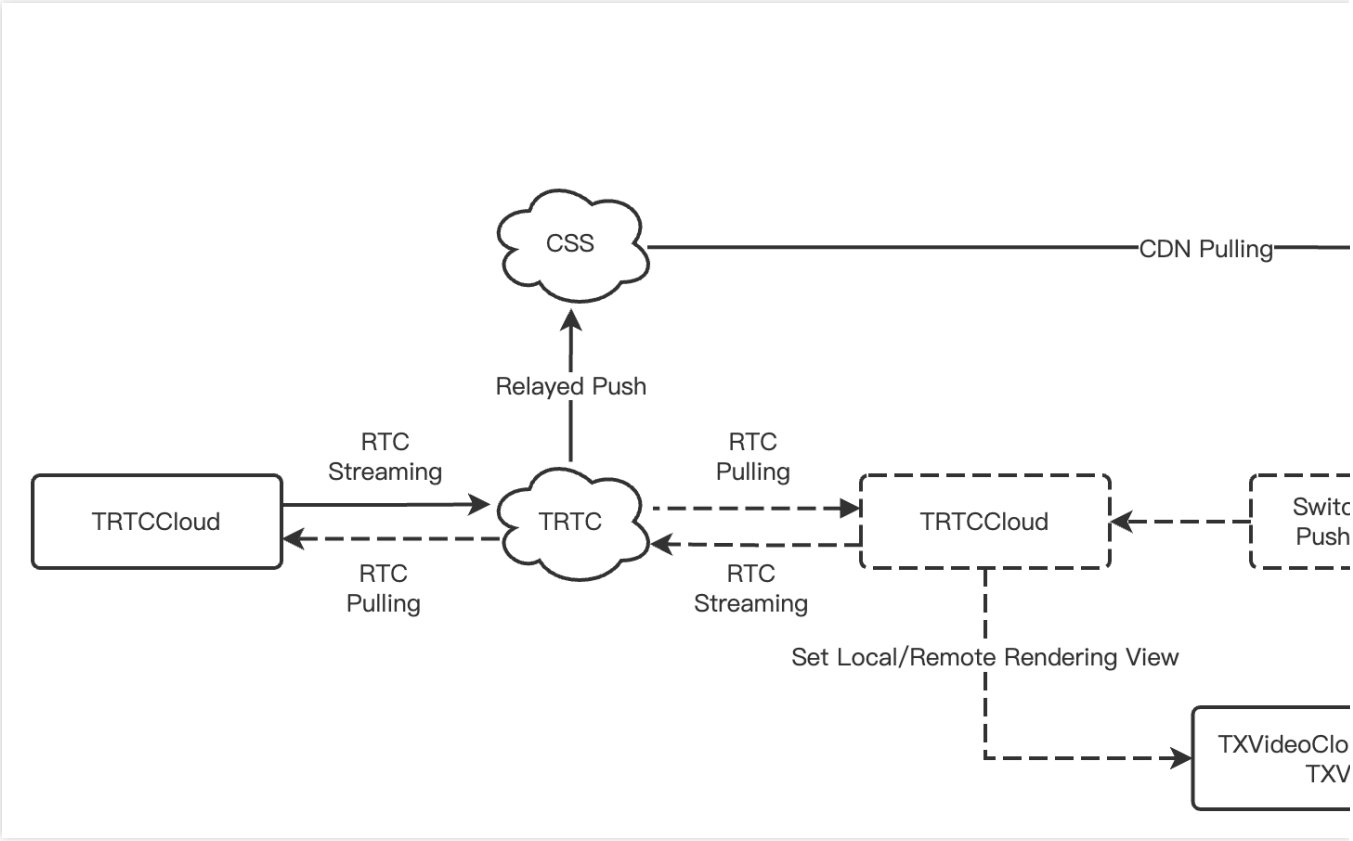
Smooth Mic On/Off Handling

In single-anchor low-frequency mic connection live streaming scenes, due to cost considerations, RTC CDN live streaming solutions or third-party live streaming publishing/playback solutions are often used. In single-anchor streaming, the anchor streams via RTC or third-party live streaming, while audiences pull streams via CDN. In interactive co-anchoring scenes, both the anchor and audiences stream via RTC. This involves switching between publishing/playback tools while maintaining a seamless experience for users. Below are the specific methods for smooth mic on/off handling in both the RTC CDN live streaming and third-party live streaming publishing/playback solutions.

1. RTC CDN Live Streaming Solution

Under the RTC CDN live streaming solution, the anchor always uses the TRTC SDK for publishing/playback. During mic connects, only the mic-connecting audience needs to switch the publishing/playback tools, and the rendering control can be reused.

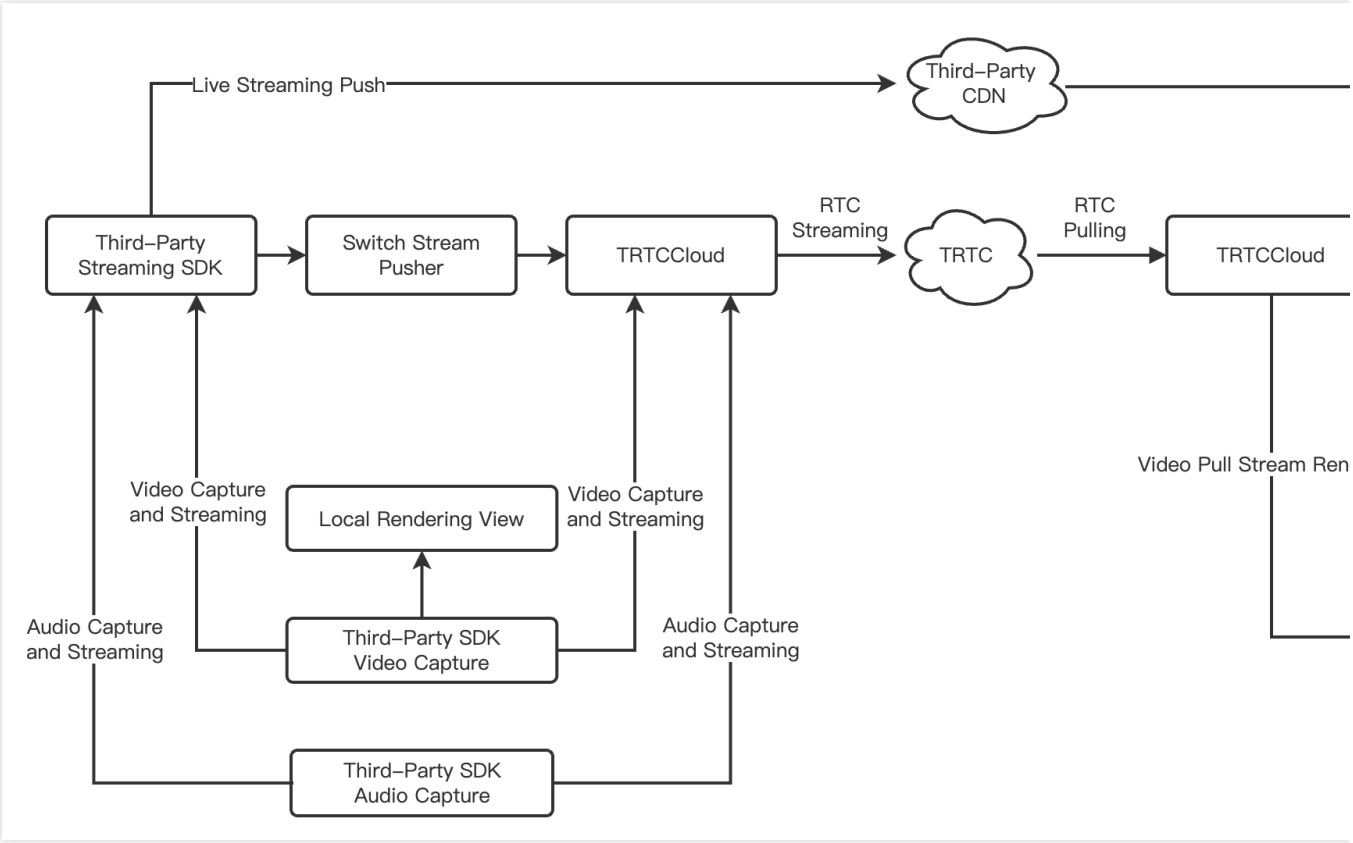
©2013-2022 Tencent Cloud. All rights reserved.



2. Third-Party Live Streaming Publishing/Playback Solution

In the third-party live streaming publishing/playback solution, during mic connects, both the anchor and the mic-connecting audience need to switch the publishing/playback control. It is recommended to use TRTC's custom capture and rendering features.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

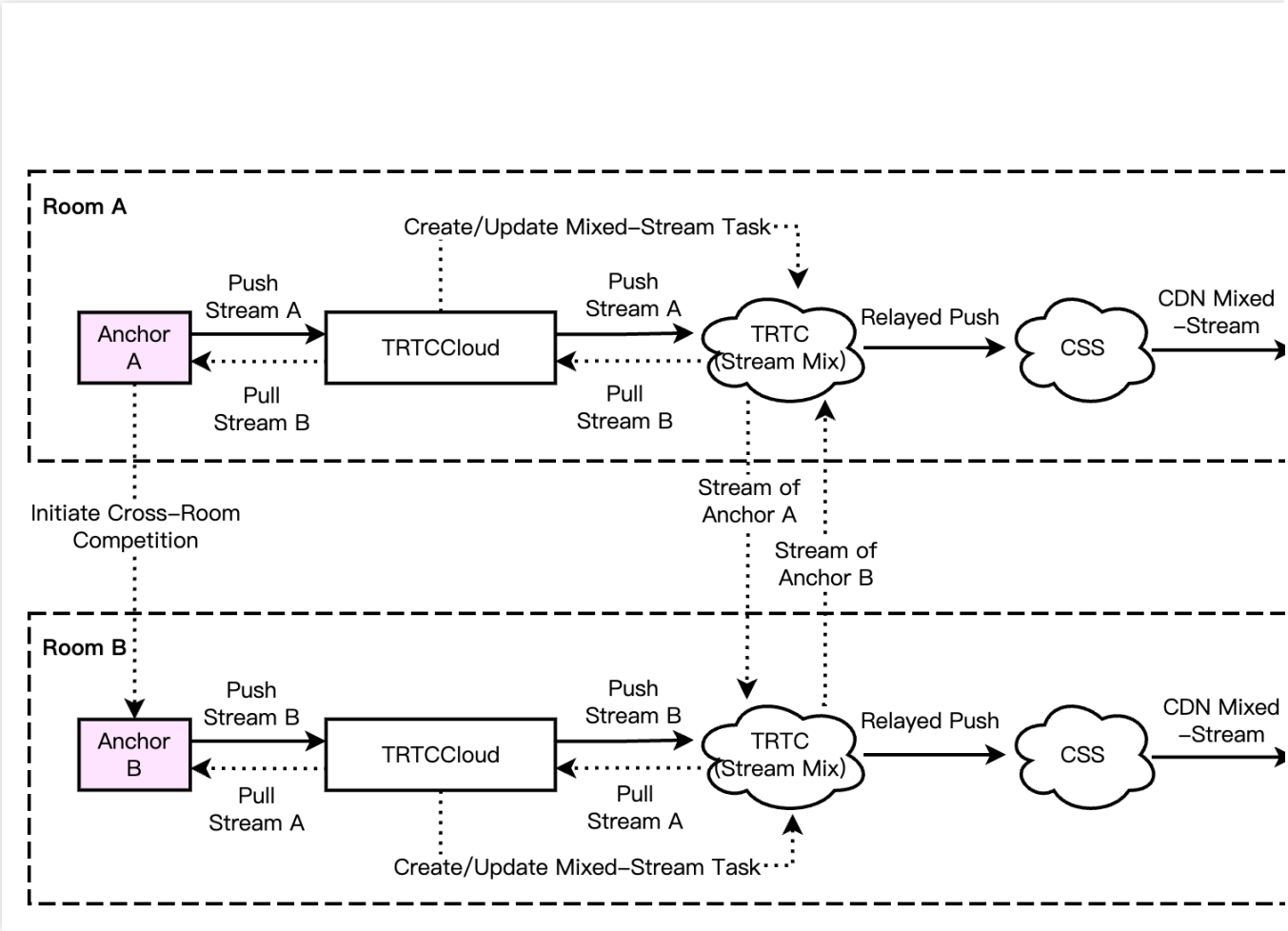
Smooth Mic on: To avoid screen interrupts when switching the stream puller, it is suggested to wait for the TRTC's first frame callback `onFirstVideoFrame` before stopping the CDN streaming.

Smooth Mic off: To avoid screen interrupts when switching the stream puller, it is suggested to wait for the video play event `onVideoPlaying` before stopping the RTC streaming.

CDN Live Streaming in Cross-Room Competition

In the RTC CDN live streaming scenario, the process of cross-room competition is relatively complex. Anchors pull RTC single streams from each other and CDN audiences pull the mixed streams from both the anchor and the cross-room competition anchor. Audiences cannot independently control the subscription logic of the anchor and cross-room competition anchor's media streams. The process for cross-room calls in a CDN live streaming scenario is shown in the diagram below:

©2013-2022 Tencent Cloud. All rights reserved.



Note:

In the CDN live streaming in cross-room competition scene, CDN audiences cannot independently control the subscription logic of the cross-room connected anchor's media stream. It needs to be uniformly controlled by the anchor through updating and publishing the media stream.

Supporting Products for the Solution

System Level	Product	Application Scenes
Access layer	Tencent Real-Time Communication (TRTC)	Provides a low-latency, high-quality multi-person audio and video real-time interaction live streaming solution, which is a foundational capability for live showroom scenarios.
Access layer	Instant Messaging (IM)	Provides room management and seat management capabilities based on group features, enables the sending and receiving of rich media messages

©2013-2022 Tencent Cloud. All rights reserved.

		such as live streaming room-wide messaging, public screen messages, as well as custom signaling and other communication needs.
Access layer	Tencent Effect SDK	Provides real-time effects processing capabilities such as beauty, filtering, makeup, fun stickers, emojis, and virtual avatars.
Cloud Services	Cloud Streaming Services (CSS)	Provides real-time audio and video relayed push, along with accelerated media stream distribution services, as well as additional capabilities such as recording and pornography detection.
Cloud Services	Video on Demand (VOD)	Catering to media such as audio, video, and images, it offers an integrated high-quality media service that includes creation, upload, storage, transcoding, media processing service, media AI, accelerated distribution and playback, and copyright protection.
Data storage	Cloud Object Storage (COS)	Provides storage services for audio and video recording files, as well as audio and video slicing files.

©2013-2022 Tencent Cloud. All rights reserved.

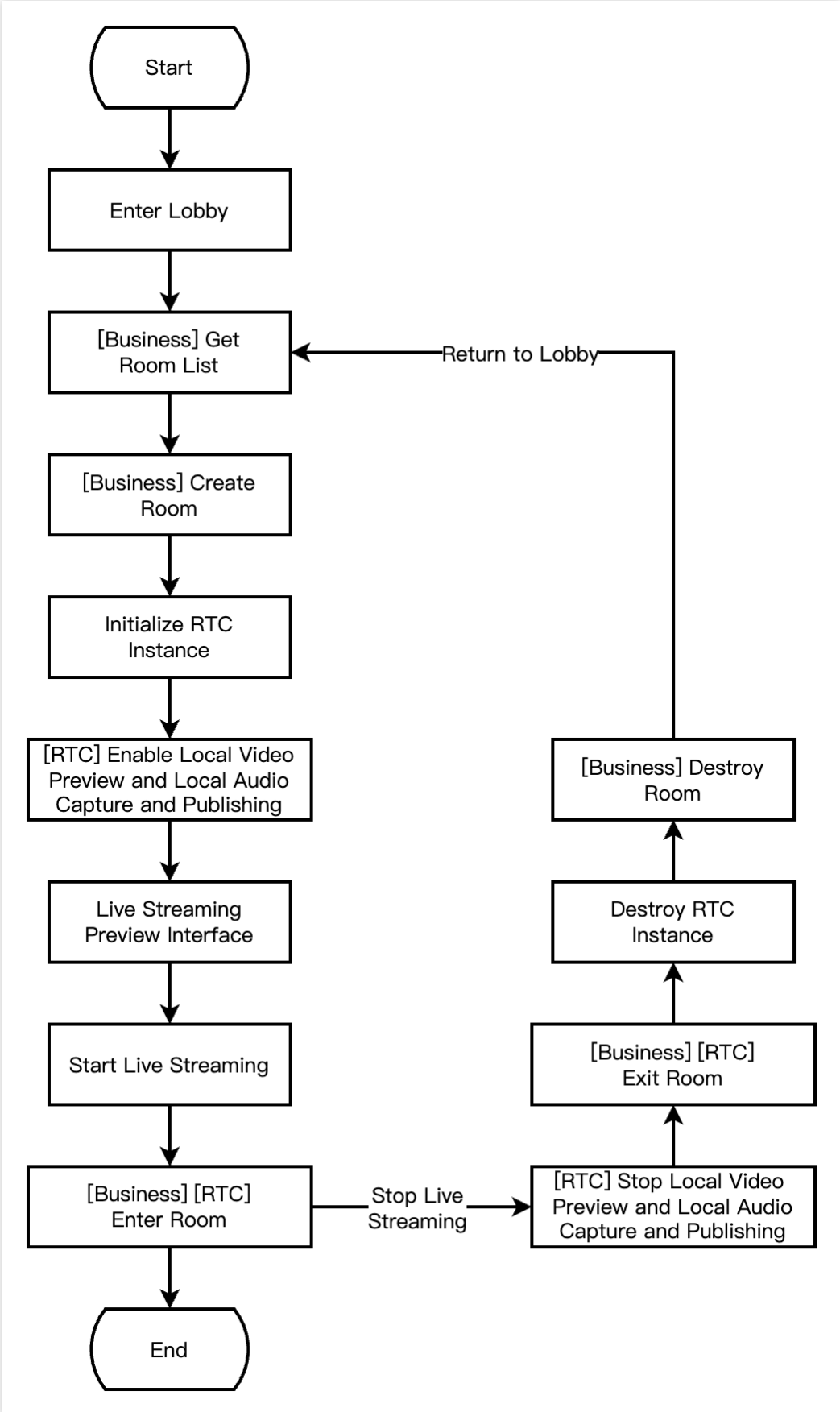
Quick Integration Guide
Android

Last updated：2024-07-18 14:26:14

Business Process

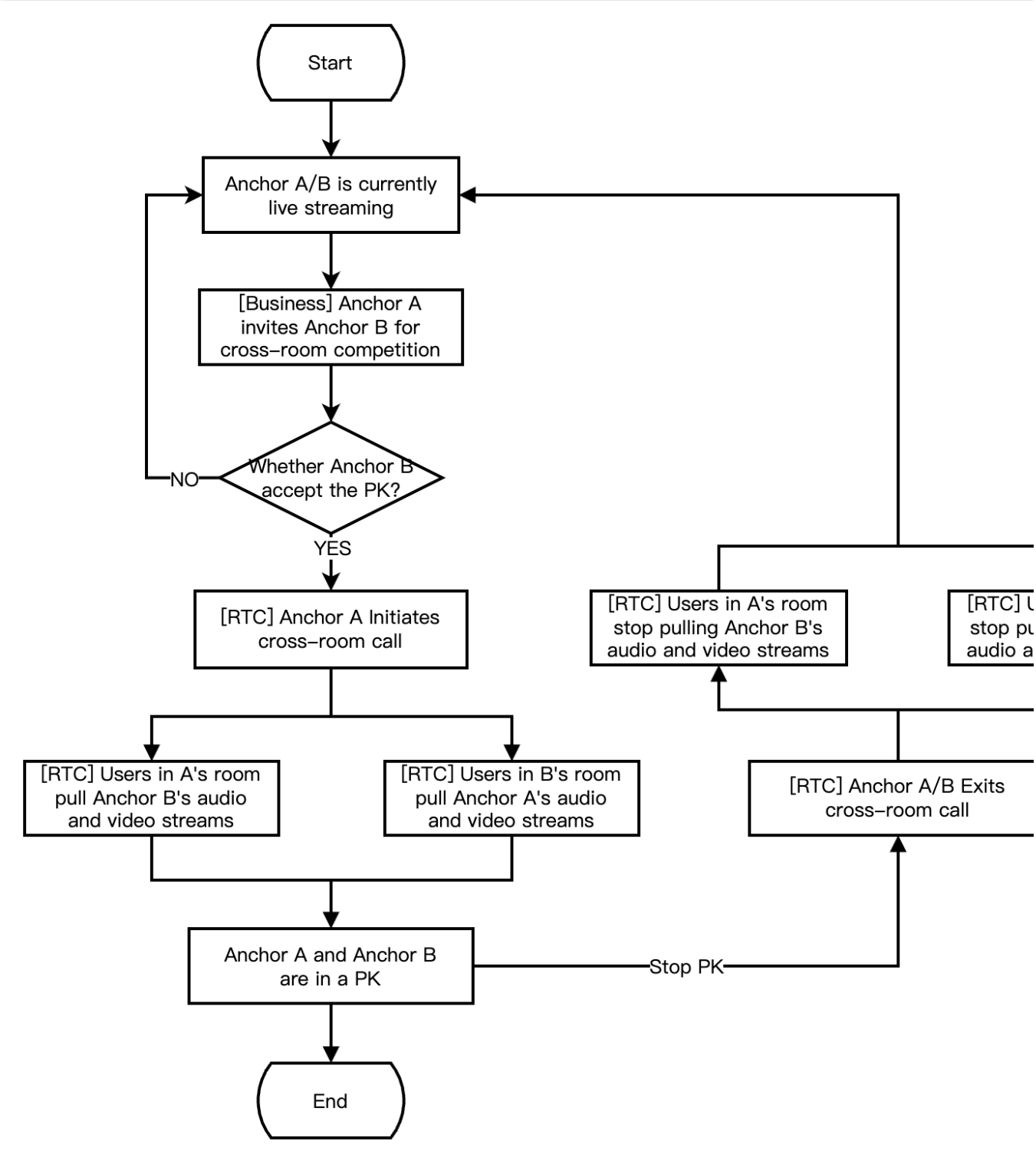
This section summarizes some common business processes in live showroom, helping you better understand the implementation process of the entire scenario.
Anchor starts and ends the live streaming.
The anchor initiates the cross-room competition.
The RTC audience enters the room for mic-connection.
The CDN audience enters the room for mic-connection.
The following figure shows the process of an anchor (room owner) local preview, creating a room, entering the room to start the live streaming, and leaving the room to end the live streaming.

©2013-2022 Tencent Cloud. All rights reserved.



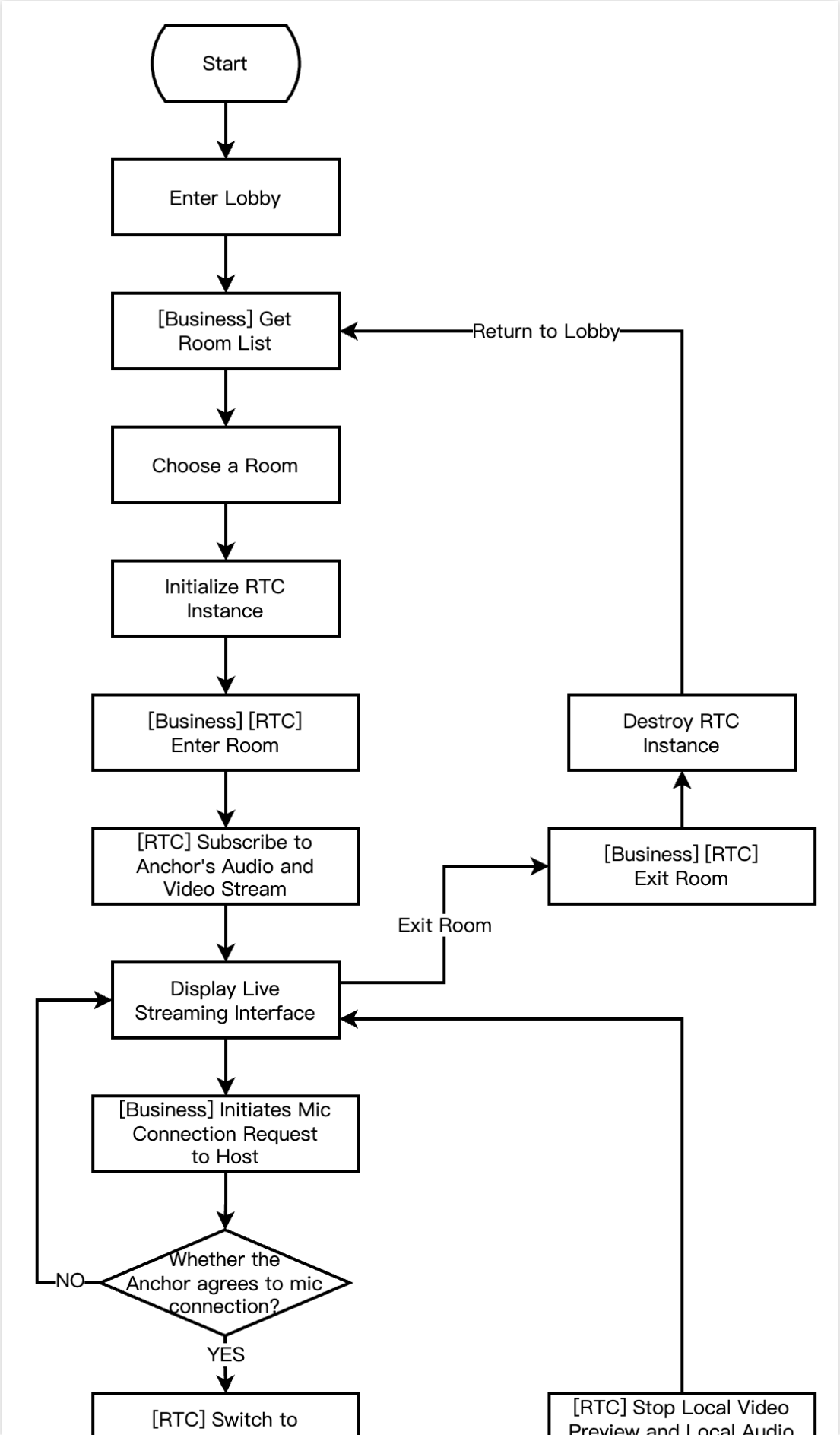
©2013-2022 Tencent Cloud. All rights reserved.

The following figure shows the process of Anchor A inviting Anchor B for a cross-room competition. During the cross-room competition, the audiences in both rooms can see the PK mic-connection live streaming of the two room owners.

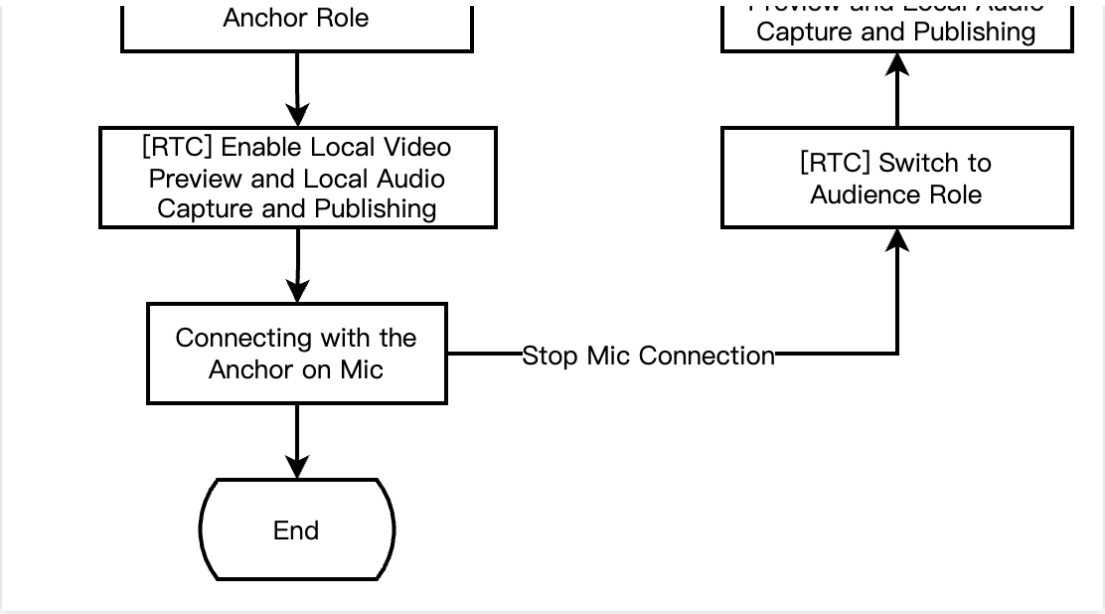


The following figure shows the process for RTC live interactive streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

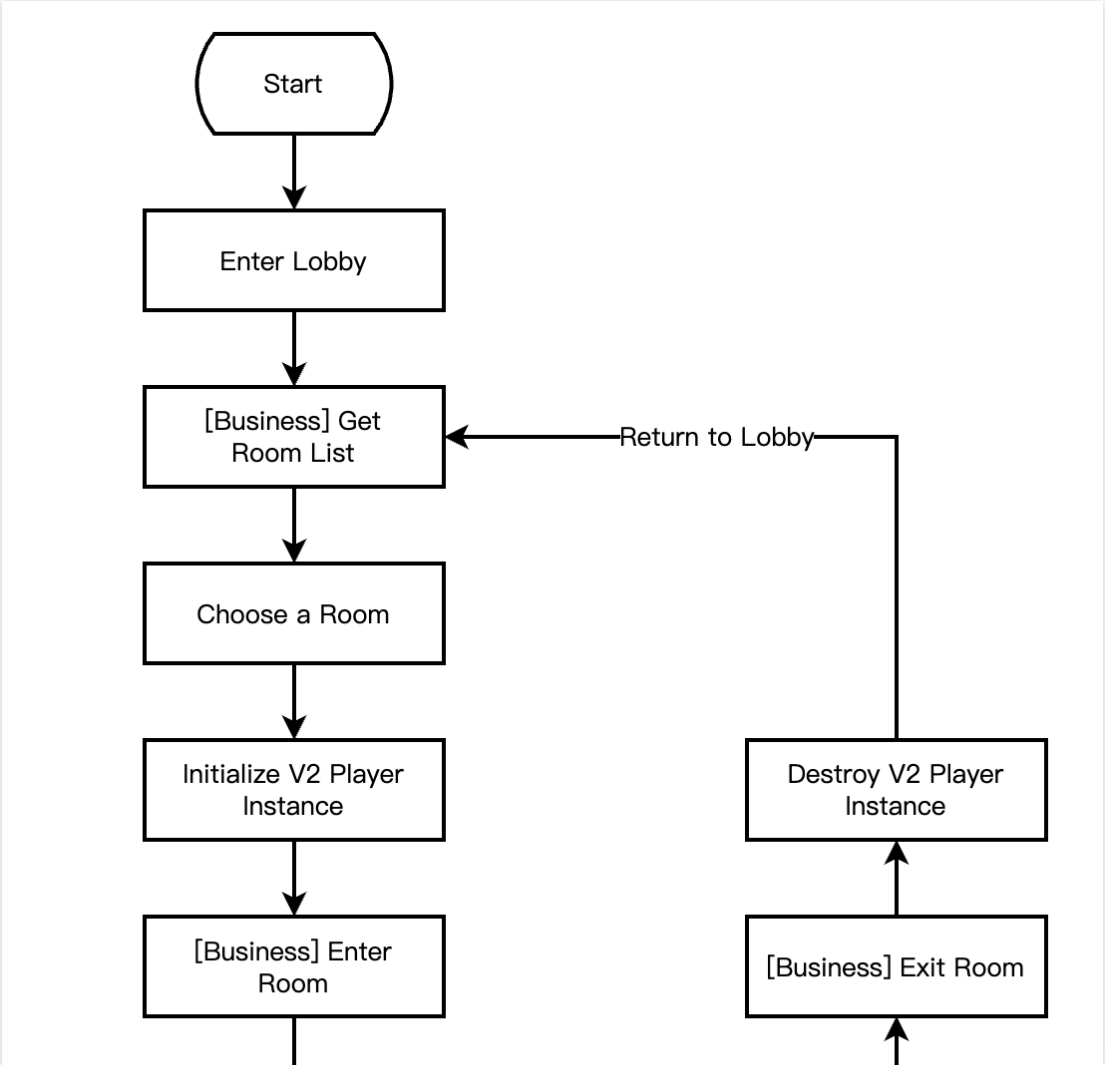
©2013-2022 Tencent Cloud. All rights reserved.



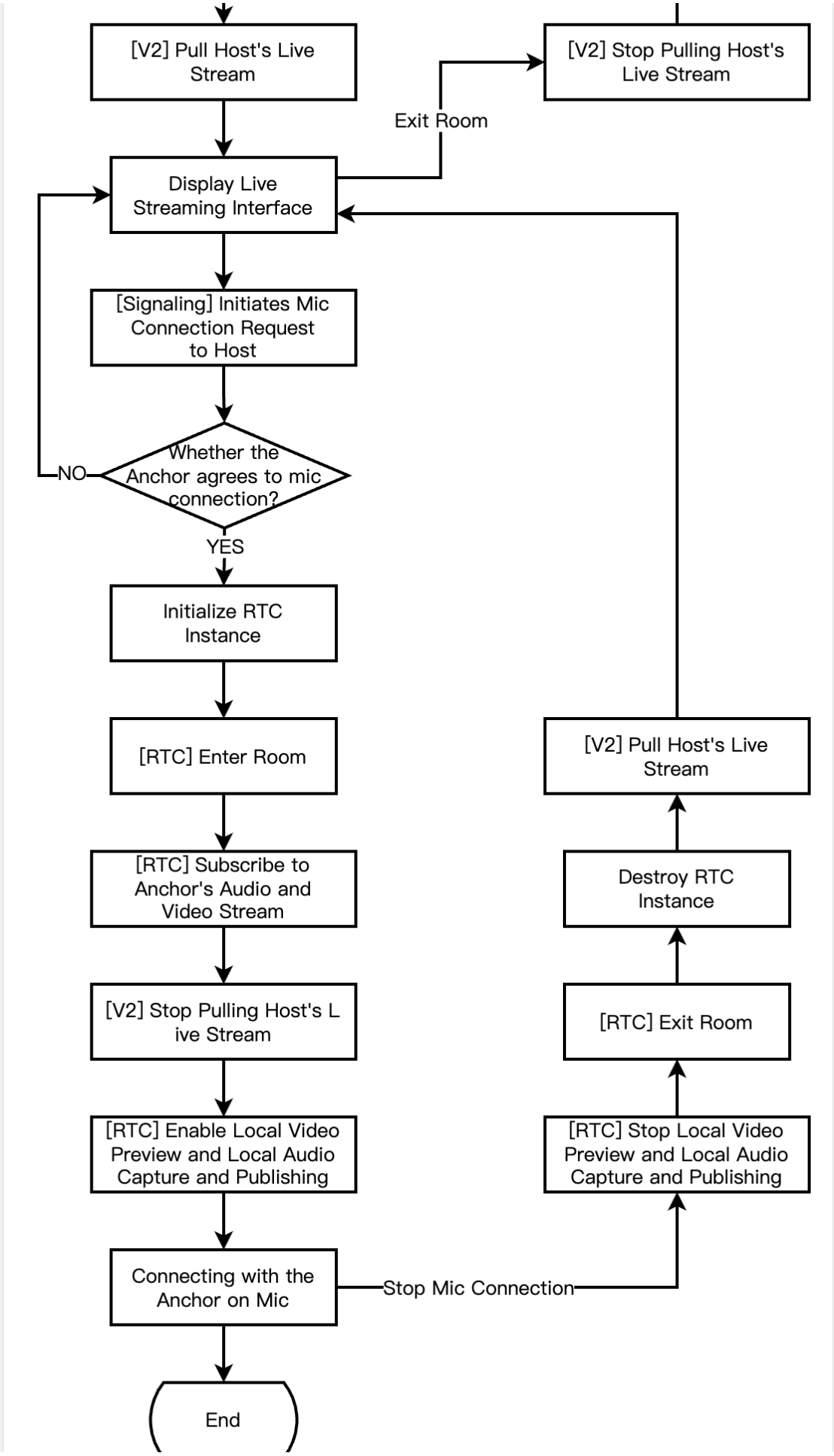
©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the process for RTC CDN live streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.



©2013-2022 Tencent Cloud. All rights reserved.



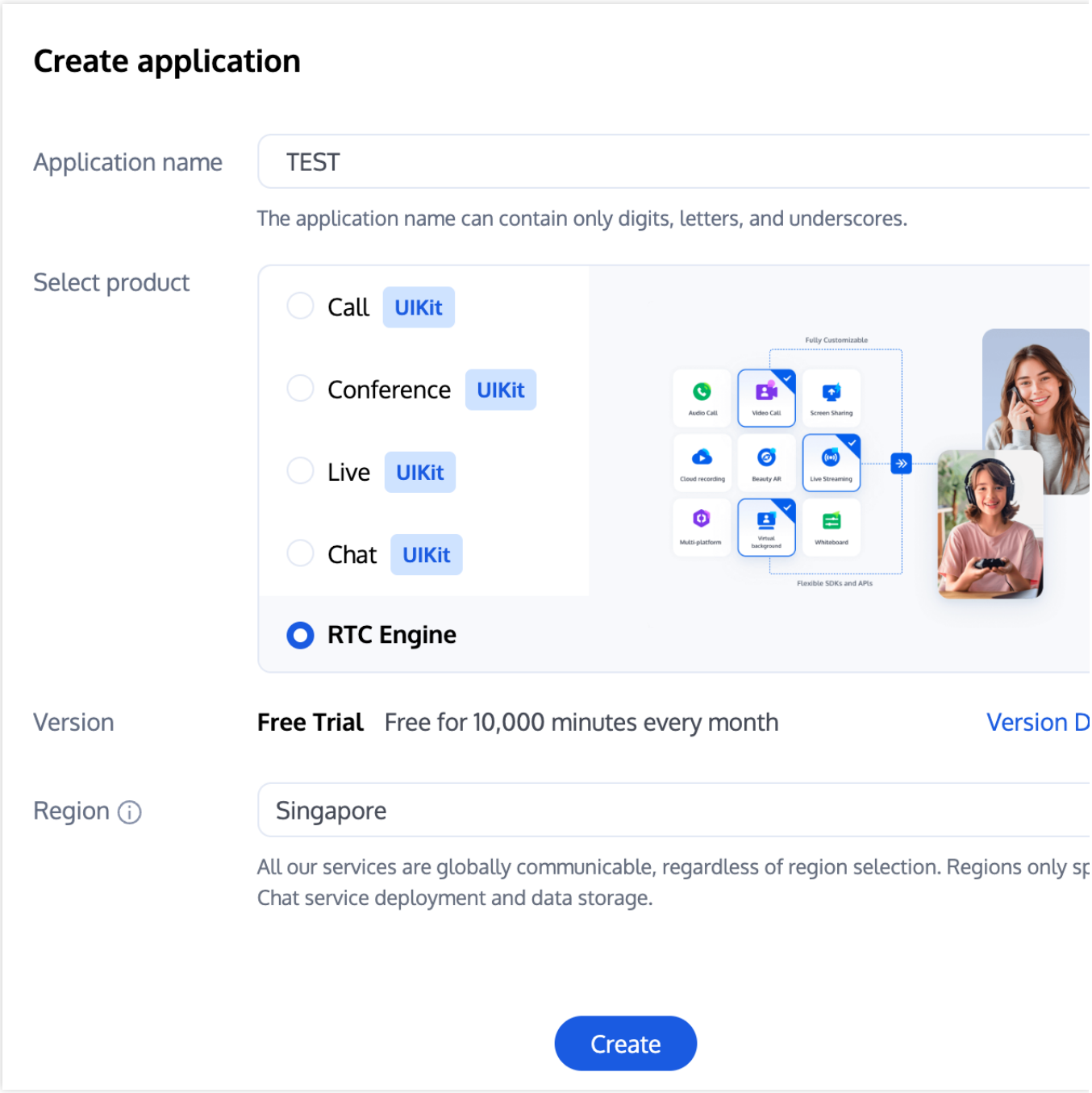
©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparation

Step 1. Activating the service.

Live showroom scenarios usually require two paid PaaS services from Tencent Cloud Tencent Real-Time Communication (TRTC) and Tencent Effect for construction. TRTC is responsible for providing real-time audio and video interaction capabilities. Tencent Effect is responsible for providing beauty effects capabilities. If you use a third-party beauty effect product, you can disregard the Tencent Effect integration part.
Activate TRTC service.
Activate Tencent Effect service.
1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

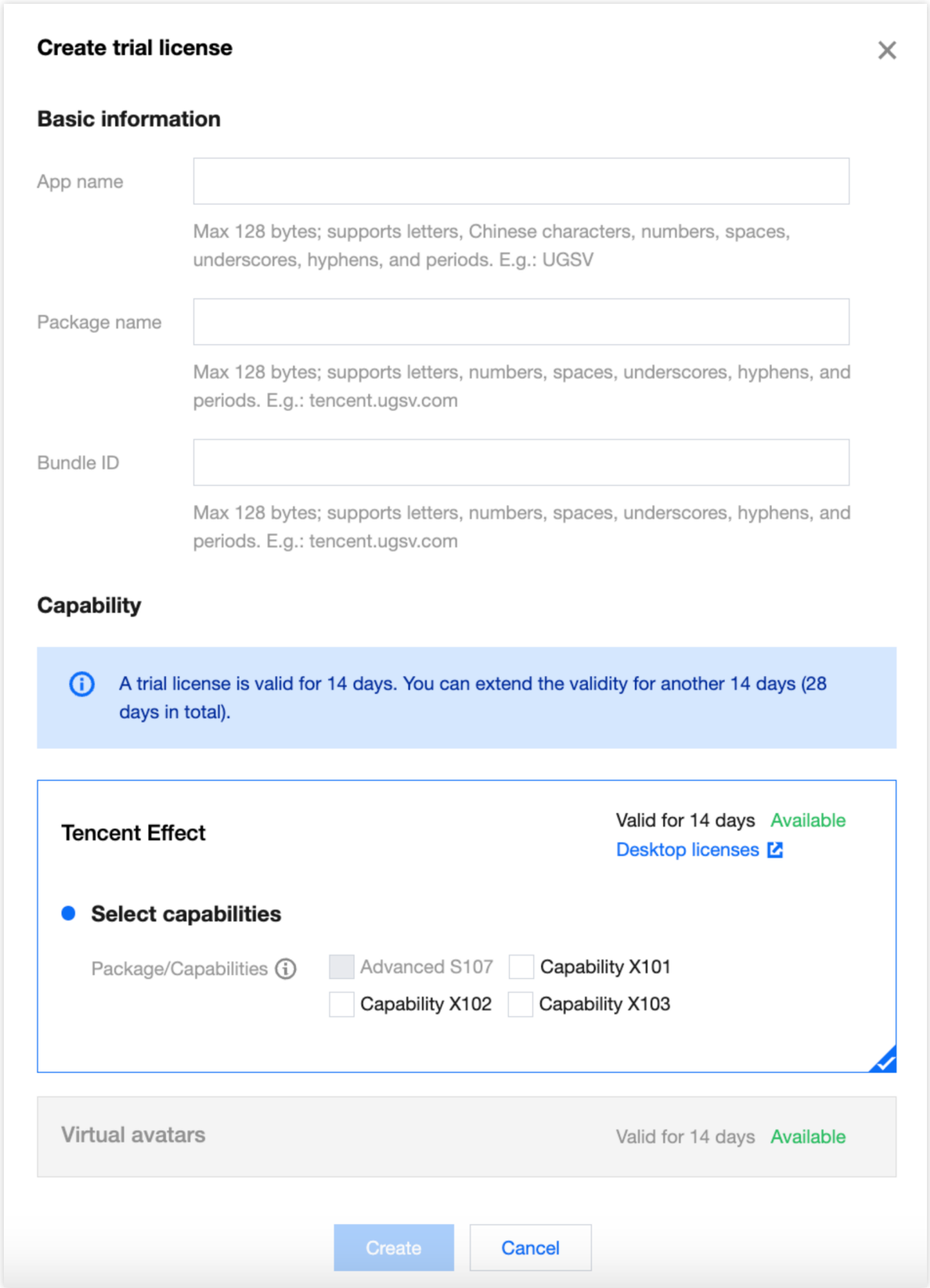
TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

2. After an application is created, you can see the basic information of the application in the Application Management - Application Overview section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.

©2013-2022 Tencent Cloud. All rights reserved.

1. Log in to Tencent Cloud Tencent Effect console > Mobile License. Click **Create Trial License** (the free trial validity period for the trial version License is 14 days. It is extendable once for a total of 28 days). Fill in the actual requirements for `App Name` , `Package Name` and `Bundle ID` . Choose **Tencent Effect**, and choose the capabilities to be tested: Advanced Package S1-07，Atomic Capability X1-01，Atomic Capability X1-02, and Atomic Capability X1-03. **After checking, accurately fill in the company name, and industry type. Upload** company service license，click **Confirm** to submit the review application, and wait for the manual review process.

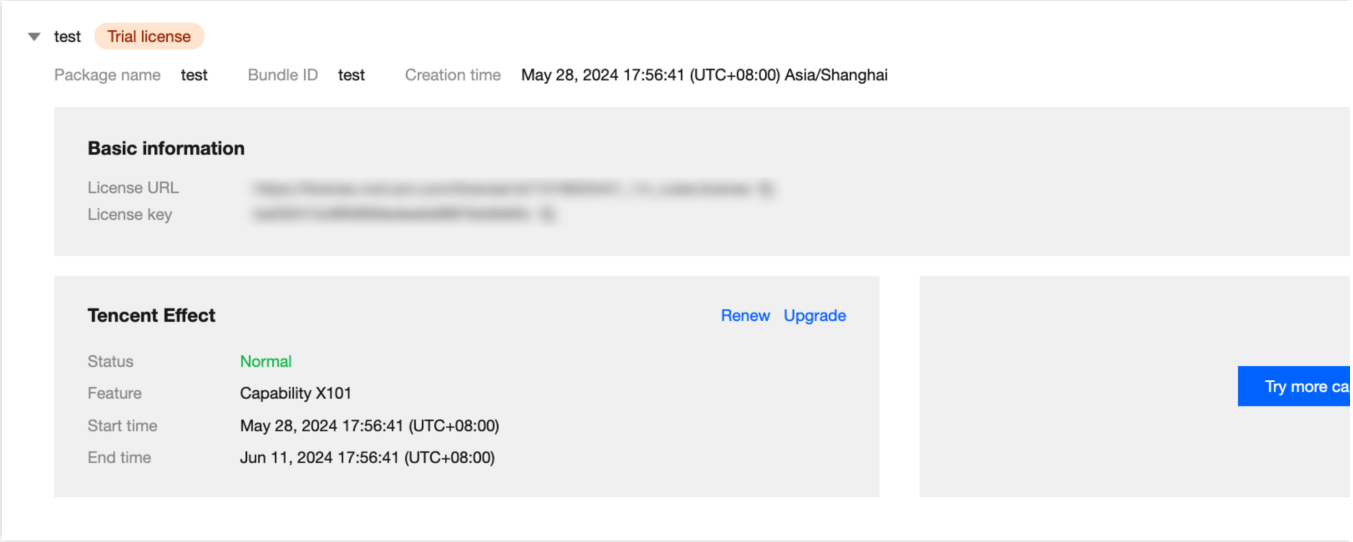
©2013-2022 Tencent Cloud. All rights reserved.



2. After the trial version License is successfully created, the page will display the generated License information. At this time, the License URL and License Key parameters are not yet effective and will only become active after the

©2013-2022 Tencent Cloud. All rights reserved.

submission is approved. **When configuring SDK initialization, you need to input both the License URL and License Key parameters. Keep the following information secure.**

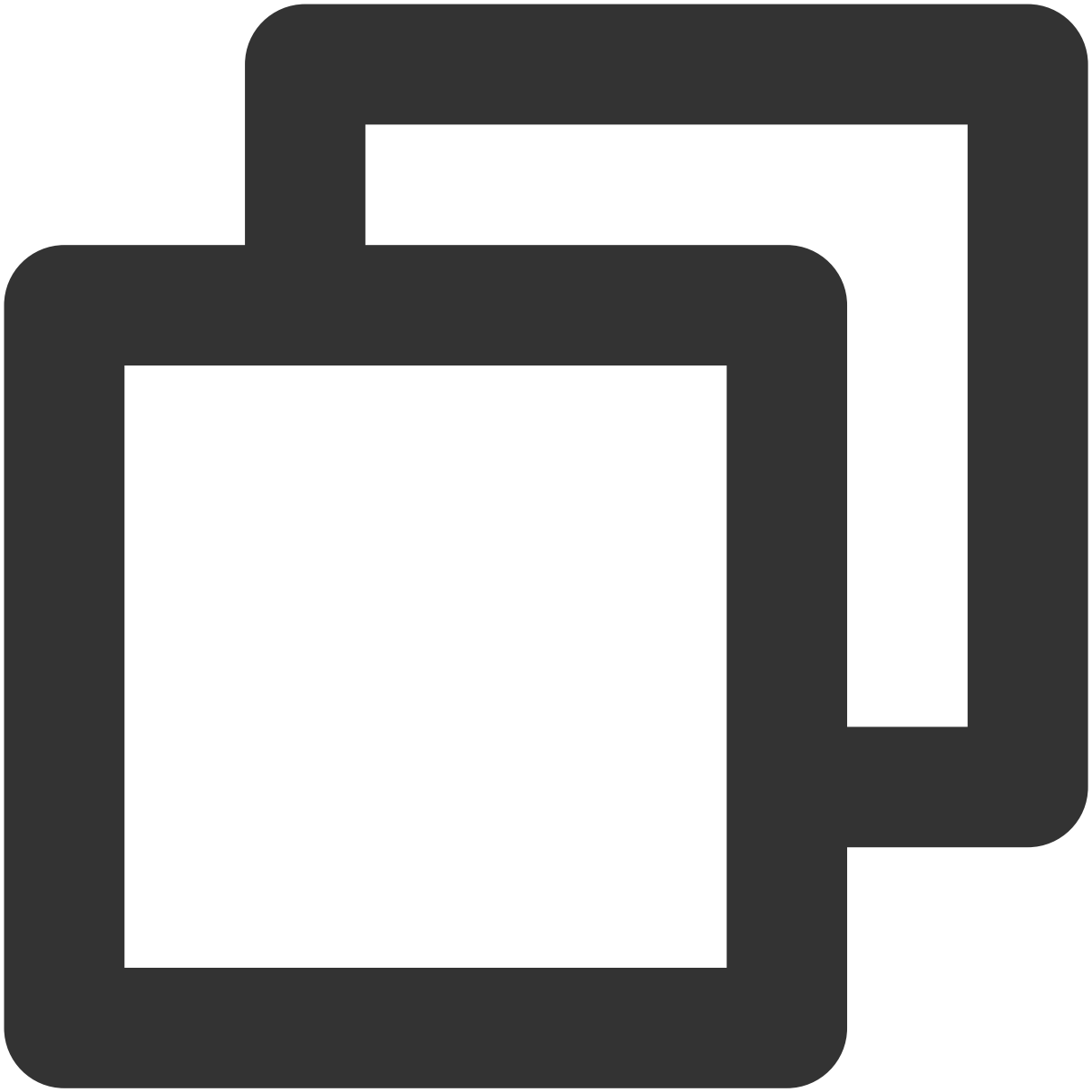


Step 2: Importing SDK.

The TRTC SDK and the Tencent Effect SDK have been released to the **mavenCentral** repository. You can configure Gradle to download and update automatically.

1. Add the dependency for the appropriate version of the SDK in dependencies.

©2013-2022 Tencent Cloud. All rights reserved.



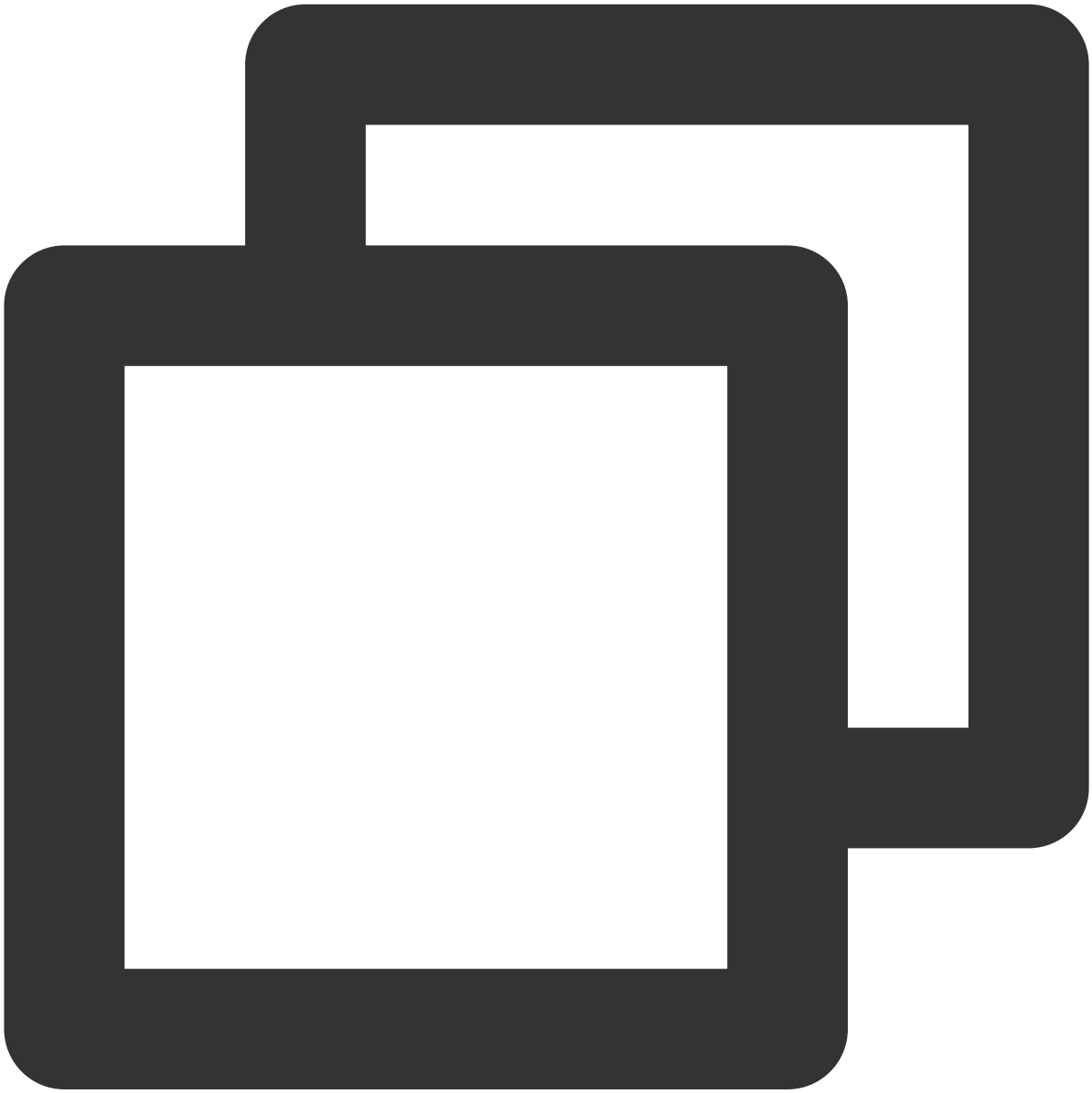
```
dependencies {
    // TRTC Lite SDK. It includes TRTC and live streaming playback features and is
    implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'

    // TRTC Professional SDK. It also includes live streaming, short video, video o
    // implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'

    // Tencent Effect SDK example of S1-07 package is as follows:
    implementation 'com.tencent.mediacloud:TencentEffect_S1-07:latest.release'
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manually Integrating Tencent Effect SDK.

2. Specify the CPU architecture used by the app in defaultConfig.



```
defaultConfig {
     ndk {
             abiFilters "armeabi-v7a", "arm64-v8a"
     }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

The TRTC SDK supports architectures including armeabi, armeabi-v7a and arm64-v8a. Additionally, it supports architectures for simulators including x86 and x86_64.

The Tencent Effect SDK currently only supports architectures including armeabi-v7a and arm64-v8a.

3. Click **Sync Now** to automatically download the SDK and integrate it into your project. If your Tencent effect package includes dynamic effect and filter features, then you need to download the corresponding package from the SDK Download Page. Unzip the free filters (./assets/lut) and animated stickers (./MotionRes) from the package and place them in the following directories in your project:

Dynamic Effect: `../assets/MotionRes` .

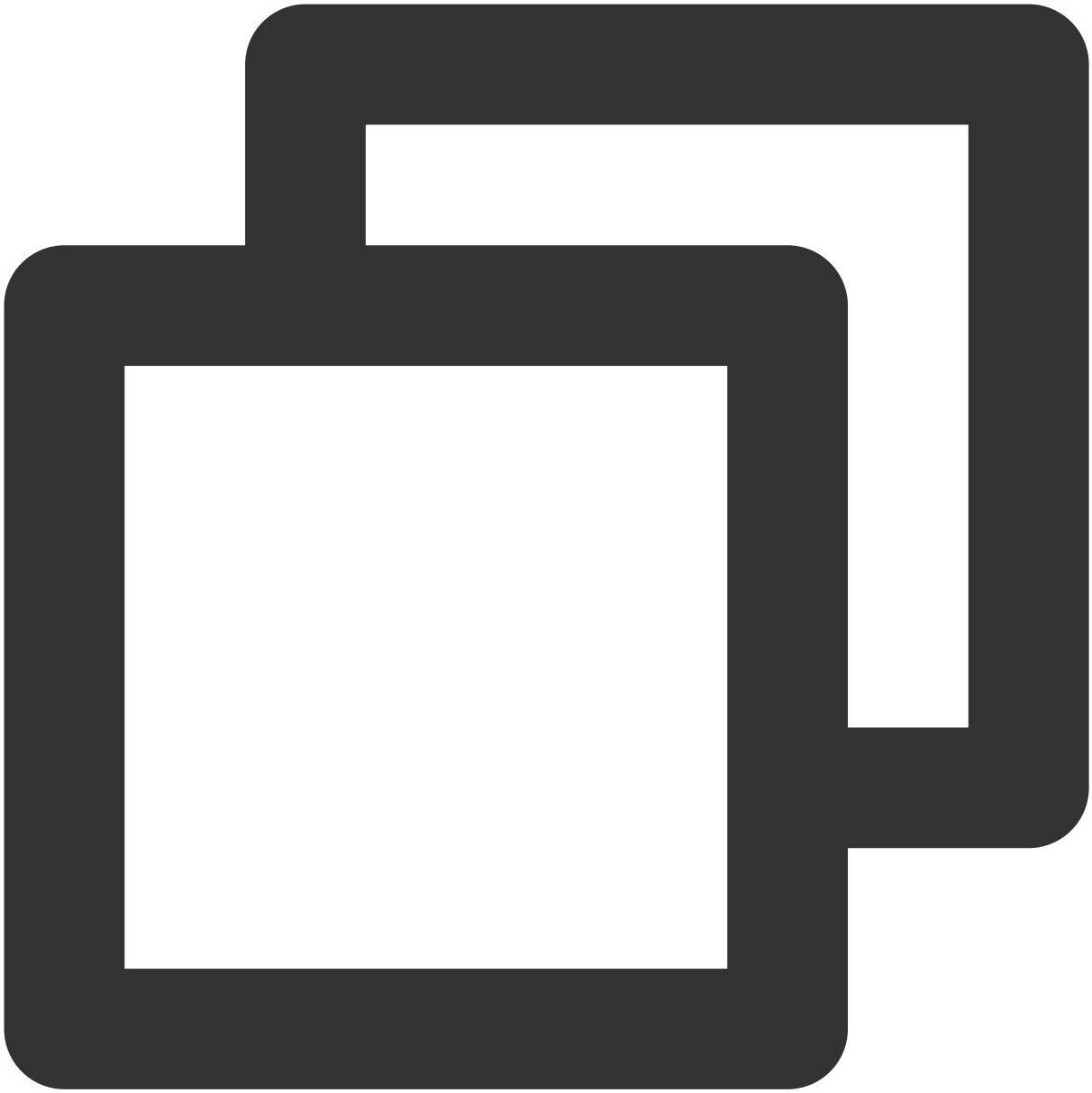
Filter: `../assets/lut` .

Step 3: Project configuration.

1. Configure permissions.

To configure app permissions in AndroidManifest.xml, for live showroom scenarios, both the TRTC SDK and the Tencent Effect SDK require the following permissions:

©2013-2022 Tencent Cloud. All rights reserved.



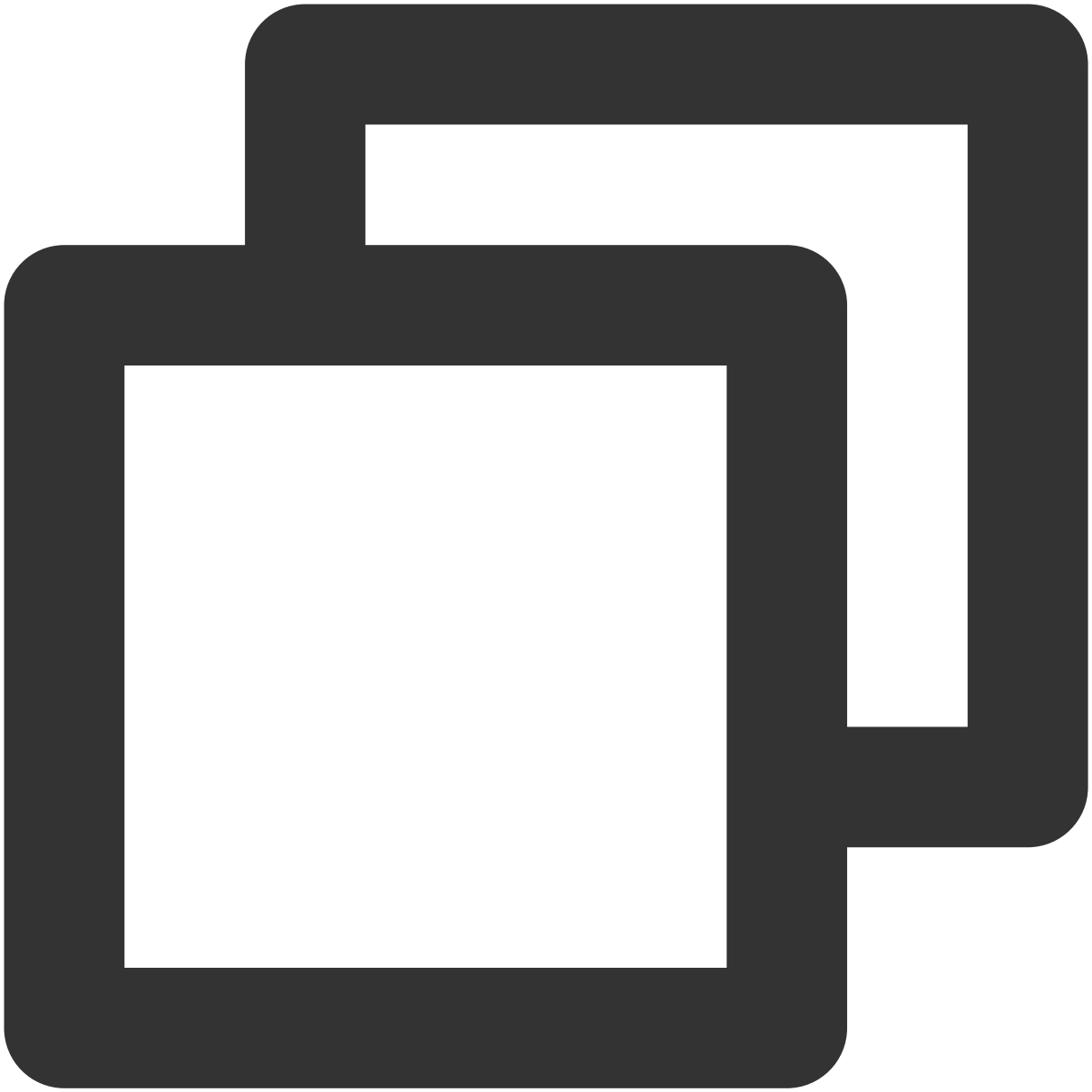
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Do not set `android:hardwareAccelerated="false"` . Disabling hardware acceleration will result in failure to render the other party's video stream.
The TRTC SDK does not have built-in permission request logic. You need to declare the corresponding permissions and features yourself. Some permissions (such as storage, recording and camera), also require runtime dynamic requests.
If the Android project's `targetSdkVersion` is 31 or higher, or if the target device runs Android 12 or a newer version, the official requirement is to dynamically request `android.permission.BLUETOOTH_CONNECT` permission in the code to use the Bluetooth feature properly. For more information, see Bluetooth Permissions.
2. Obfuscation configuration.
Since we use Java's reflection features inside the SDK, you need to add relevant SDK classes to the non-obfuscation list in the proguard-rules.pro file:

©2013-2022 Tencent Cloud. All rights reserved.



```
-keep class com.tencent.** { *; }
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class androidx.exifinterface.** { *;}
```

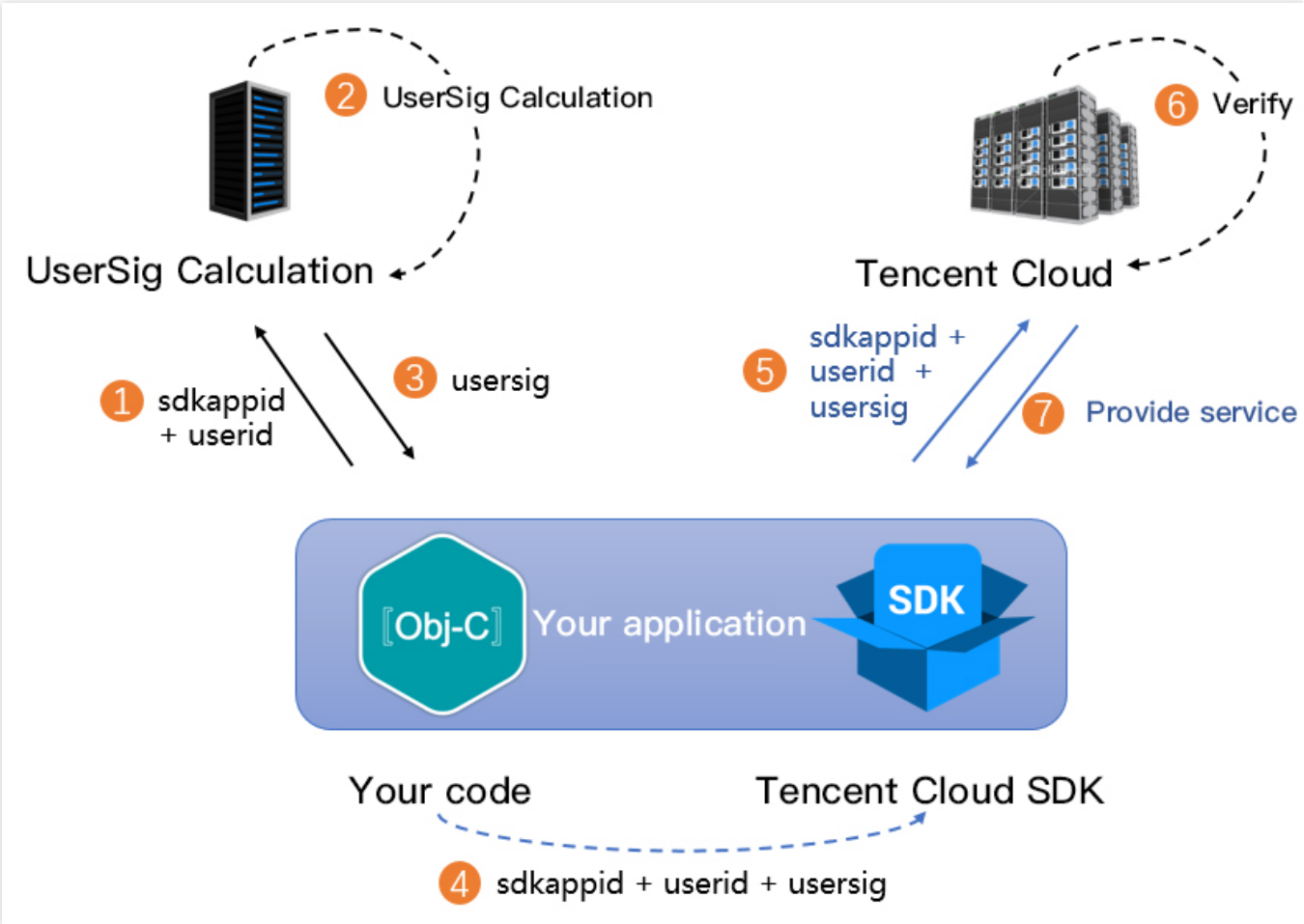
Step 4: Authentication and authorization.

TRTC authentication credential.

©2013-2022 Tencent Cloud. All rights reserved.

Tencent Effect authentication license.
UserSig is a security protection signature designed by Tencent Cloud to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when it enters the room.
Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.
Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.
The specific implementation process is as follows:
1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.
3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.

©2013-2022 Tencent Cloud. All rights reserved.



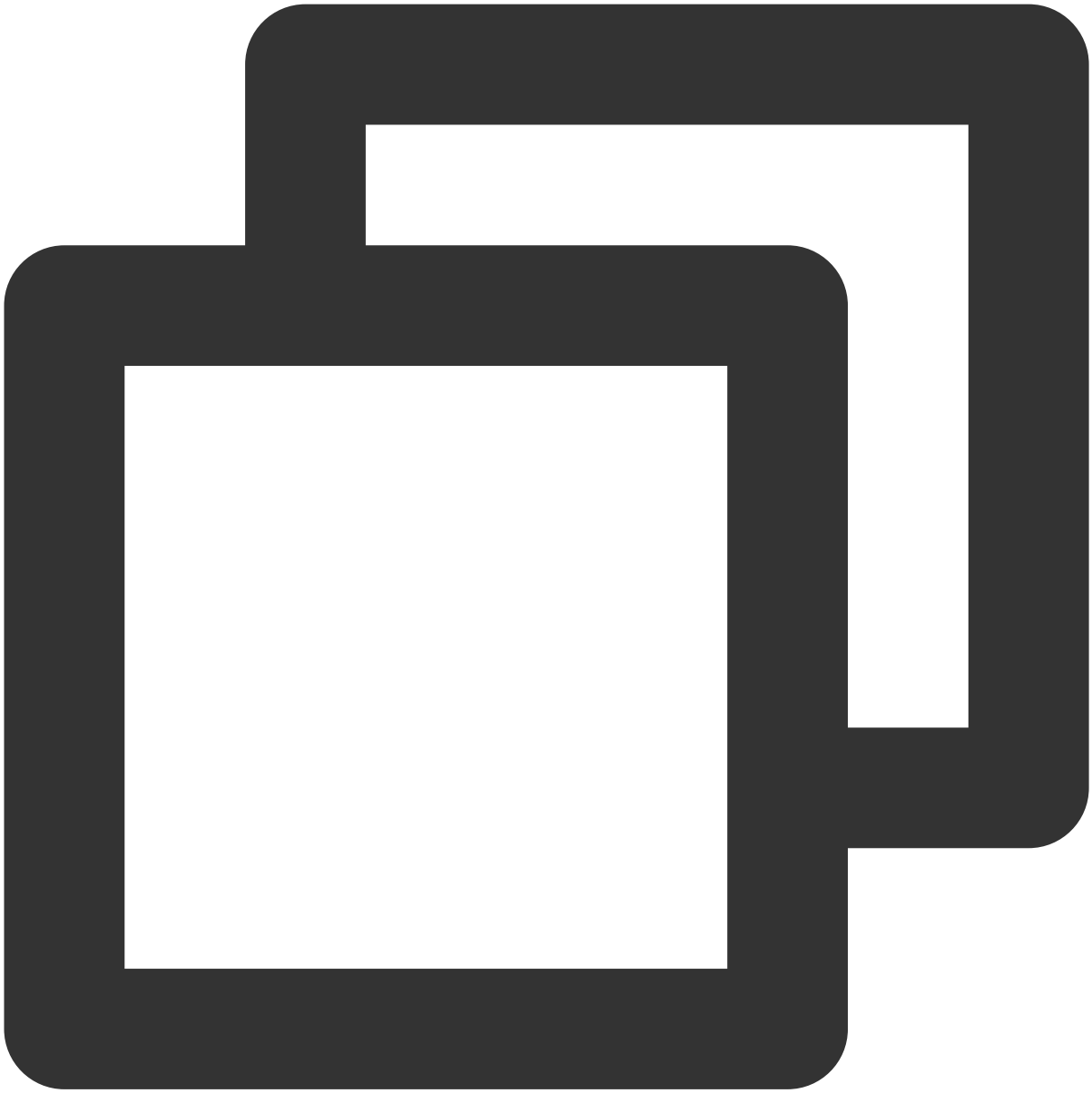
Note:

The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Before using Tencent Effect, you need to verify the license credential with Tencent Cloud. Configuring the License requires License Key and License Url. Sample code is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



```
import com.tencent.xmagic.telicense.TELicenseCheck;

// If the purpose is just to trigger the download or update of the License, and not
TELicenseCheck.getInstance().setTELicense(context, URL, KEY, new TELicenseCheck.TEL
    @Override
    public void onLicenseCheckFinish(int errorCode, String msg) {
        // Note: This callback does not necessarily be called on the calling thread
        if (errorCode == TELicenseCheck.ERROR_OK) {
            // Authentication successful.
        } else {
            // Authentication failed.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            }


        }
    });
```

Note:

It is recommended to trigger the authentication permission in the initialization code of related business modules. Ensure to avoid having to download the License temporarily before use. Additionally, during authentication, network permissions must be ensured.

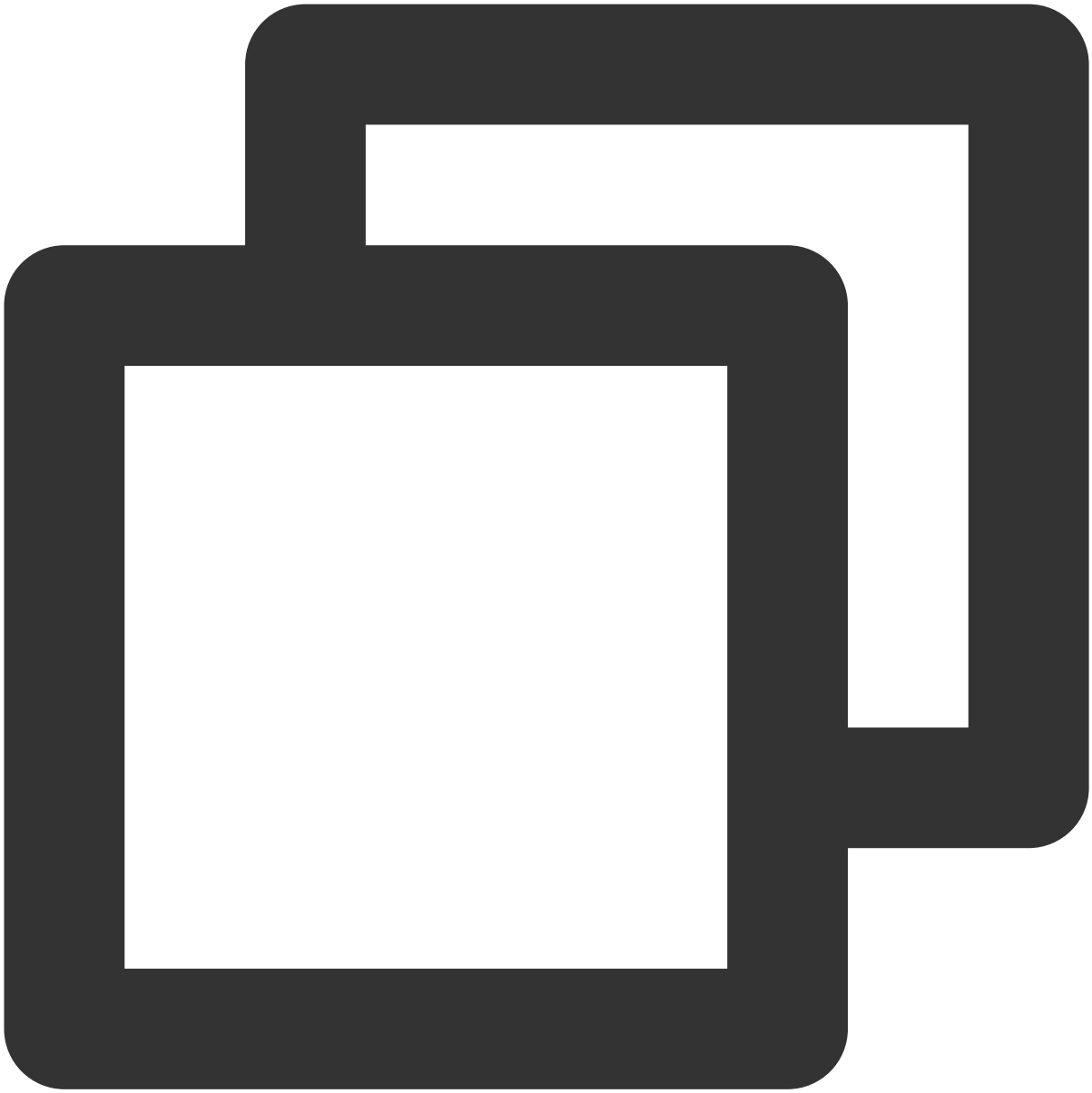
The actual application's Package Name must match exactly with the Package Name associated with the creation of License. Otherwise, it will lead to License verification failure. For details, see Authentication Error Code.

Step 5: Initializing the SDK.

Initialize the TRTC SDK.

Initialize the Tencent Effect SDK.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (Single Instance Pattern).
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Set event listeners.
mTRTCCloud.addListener(trtcSdkListener);

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
private TRTCCloudListener trtcSdkListener = new TRTCCloudListener() {
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        Log.d(TAG, errCode + errMsg);
    }
```

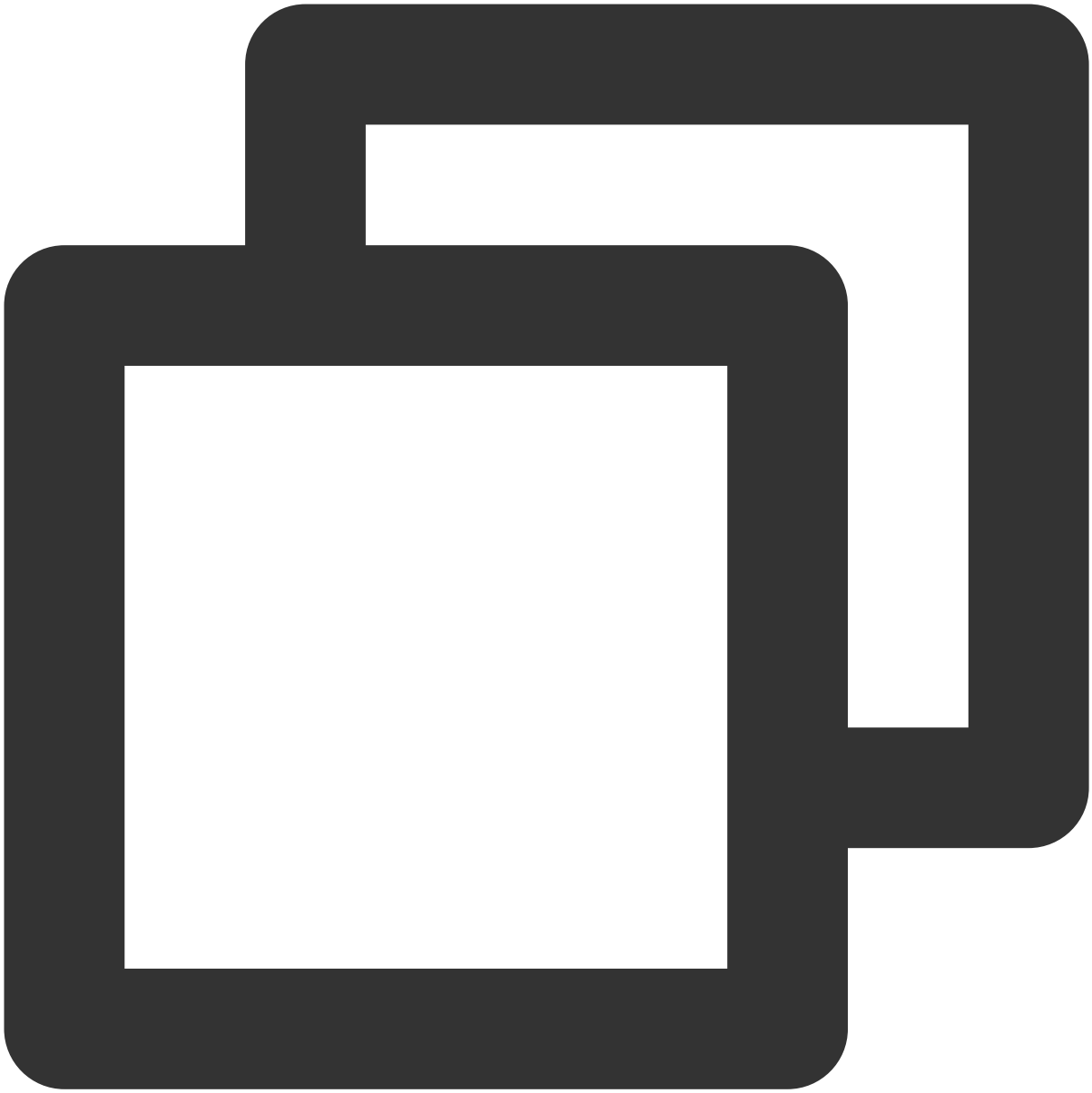
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        Log.d(TAG, warningCode + warningMsg);
    }
};

// Remove event listener.
mTRTCCloud.removeListener(trtcSdkListener);
// Terminate TRTC SDK instance (Singleton Pattern).
TRTCCloud.destroySharedInstance();
```

Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

©2013-2022 Tencent Cloud. All rights reserved.



```
import com.tencent.xmagic.XmagicApi;

// Initialize the beauty SDK.
XmagicApi mXmagicApi = new XmagicApi(context, XmagicResParser.getResPath(), new Xma

// During development and debugging, you can set the log level to DEBUG. For releas
mXmagicApi.setXmagicLogLevel(Log.WARN);

// Release the beauty SDK. This method needs to be called on the GL thread.
mXmagicApi.onDestroy();
```

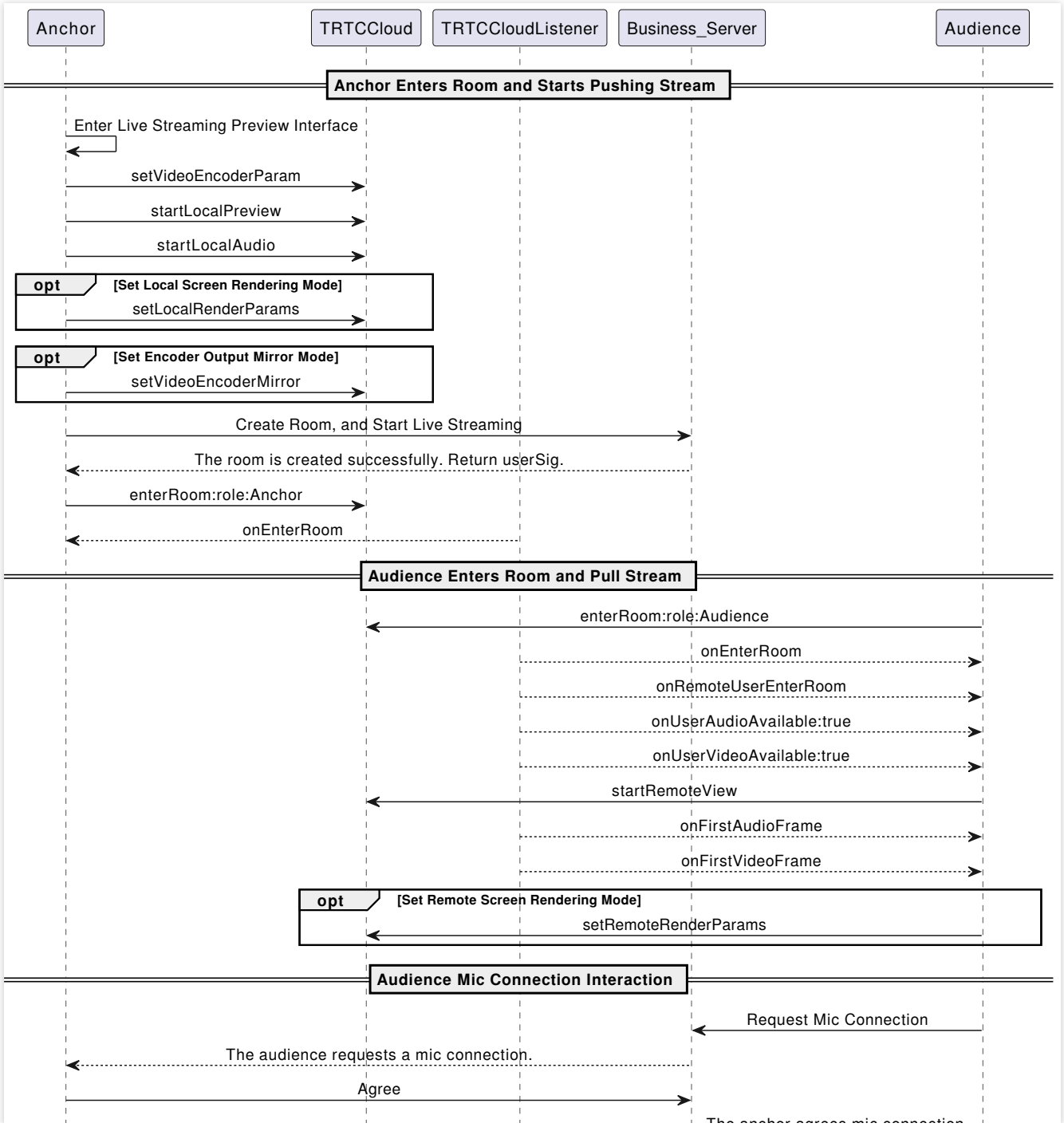
©2013-2022 Tencent Cloud. All rights reserved.

Note:

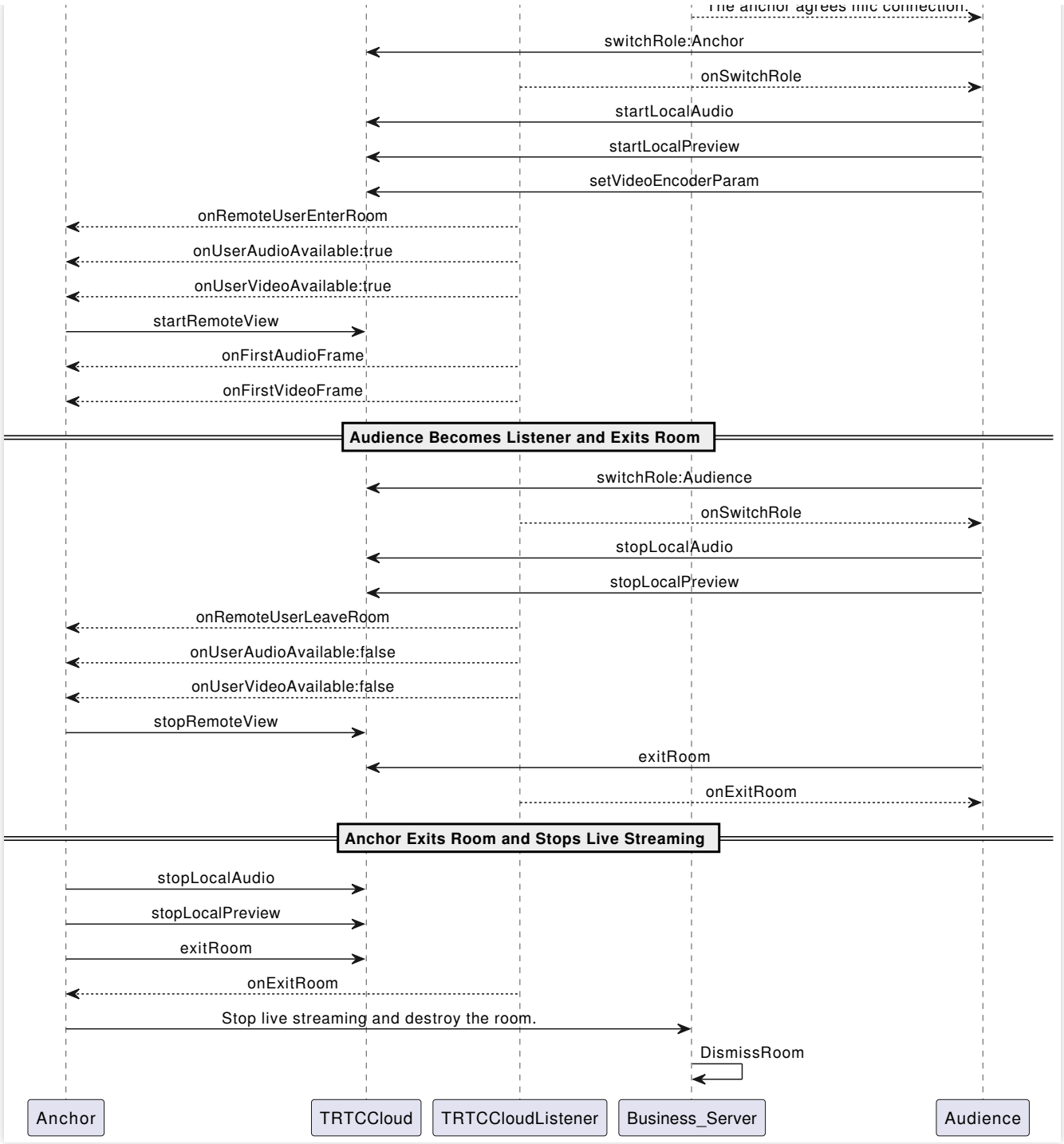
Before the Tencent Effect SDK is initialized, resource copying and other preparatory work are needed. For detailed steps, see Using the Tencent Effect SDK.

Integration Process

API sequence diagram.



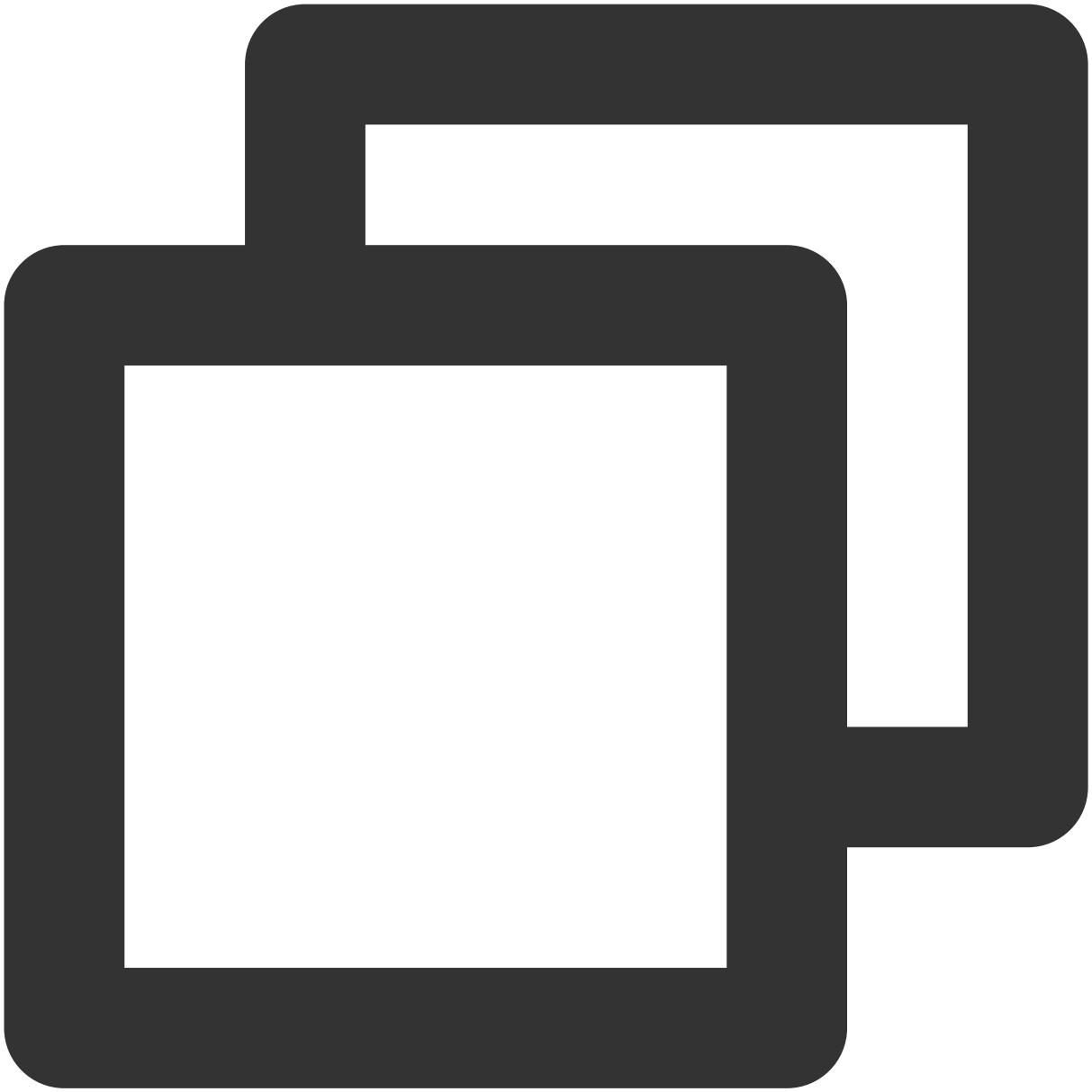
©2013-2022 Tencent Cloud. All rights reserved.



Step 1: The anchor enters the room to push streams.

The control used by the TRTC SDK to display video streams only supports passing in a `TXCloudVideoView` type. Therefore, you need to first define the view rendering control in the layout file.

©2013-2022 Tencent Cloud. All rights reserved.



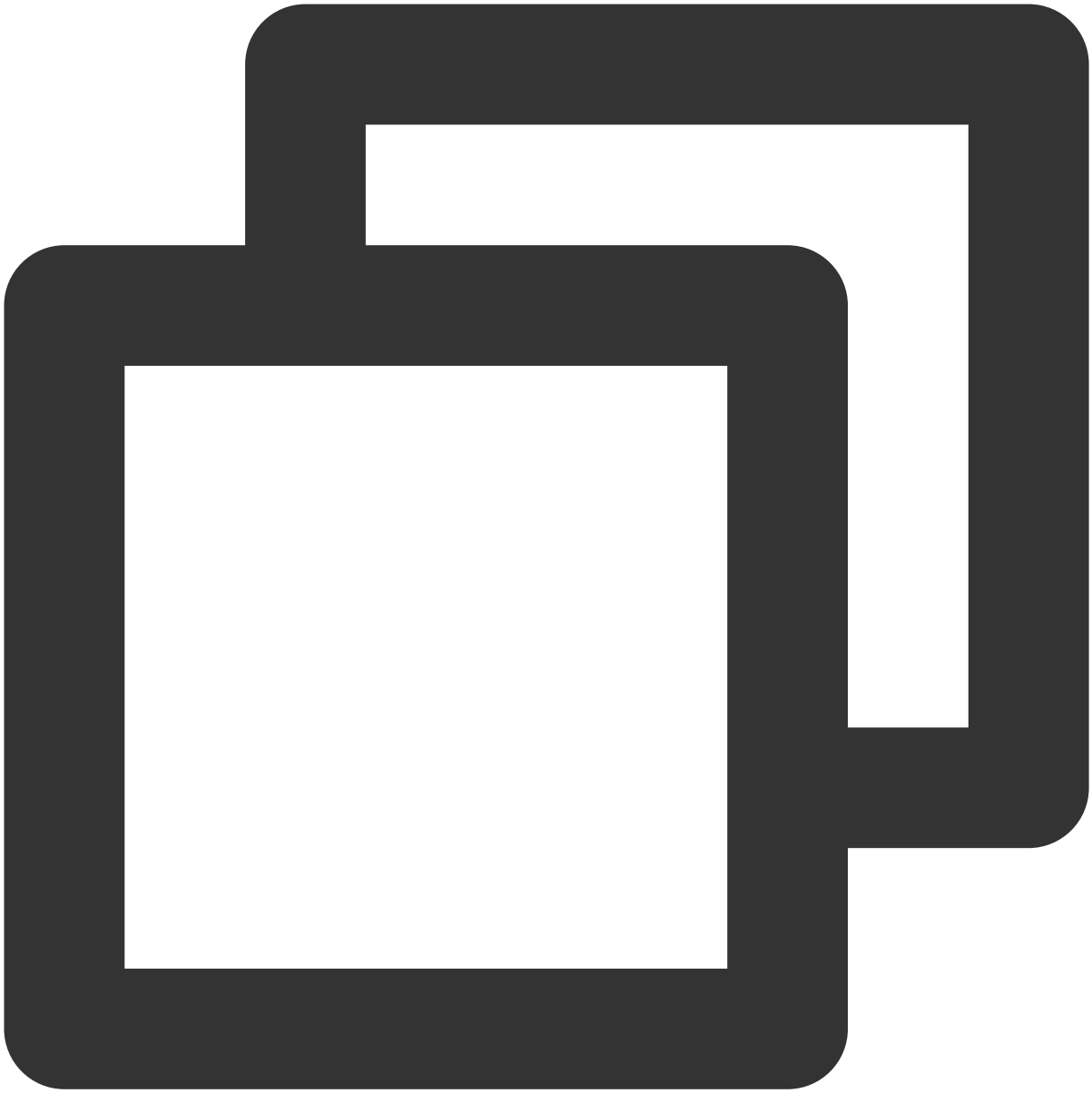
```
<com.tencent.rtmp.ui.TXCloudVideoView
    android:id="@+id/live_cloud_view_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Note:

If you need to specifically use `TextureView` or `SurfaceView` as the view rendering control, see Advanced Features - View Rendering Control.

1. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
TXCloudVideoView mTxcvvAnchorPreviewView = findViewById(R.id.live_cloud_view_main);

// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
encParam.videoFps = 15;
encParam.videoBitrate = 1300;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAnchorPreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

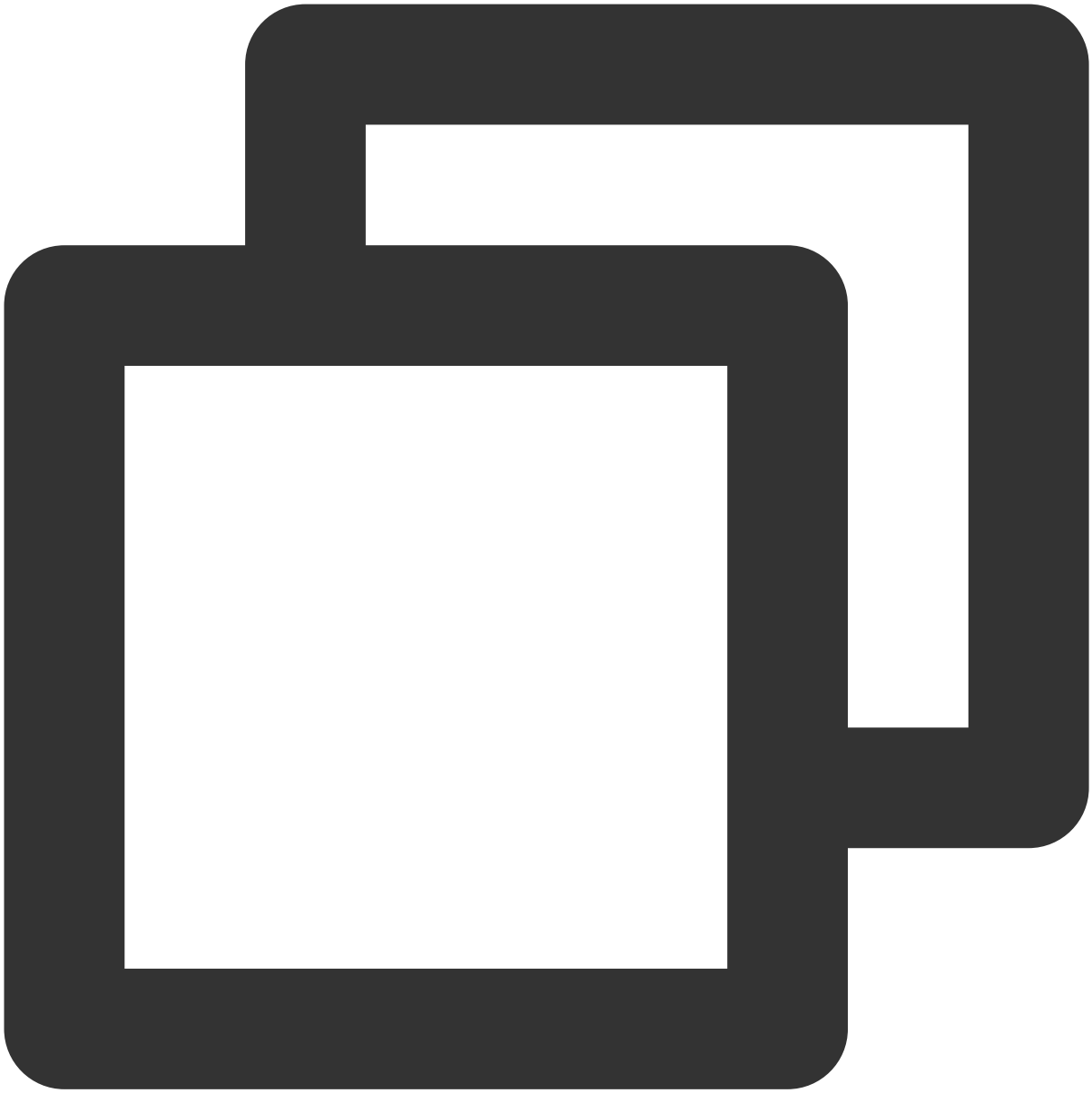
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

2. The anchor sets rendering parameters for the local video, and the encoder output video mode (optional).

©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;     // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering parameters for the local video.
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output.
mTRTCCloud.setVideoEncoderMirror(boolean mirror);
// Set the rotation of the video encoder output.
mTRTCCloud.setVideoEncoderRotation(int rotation);
```

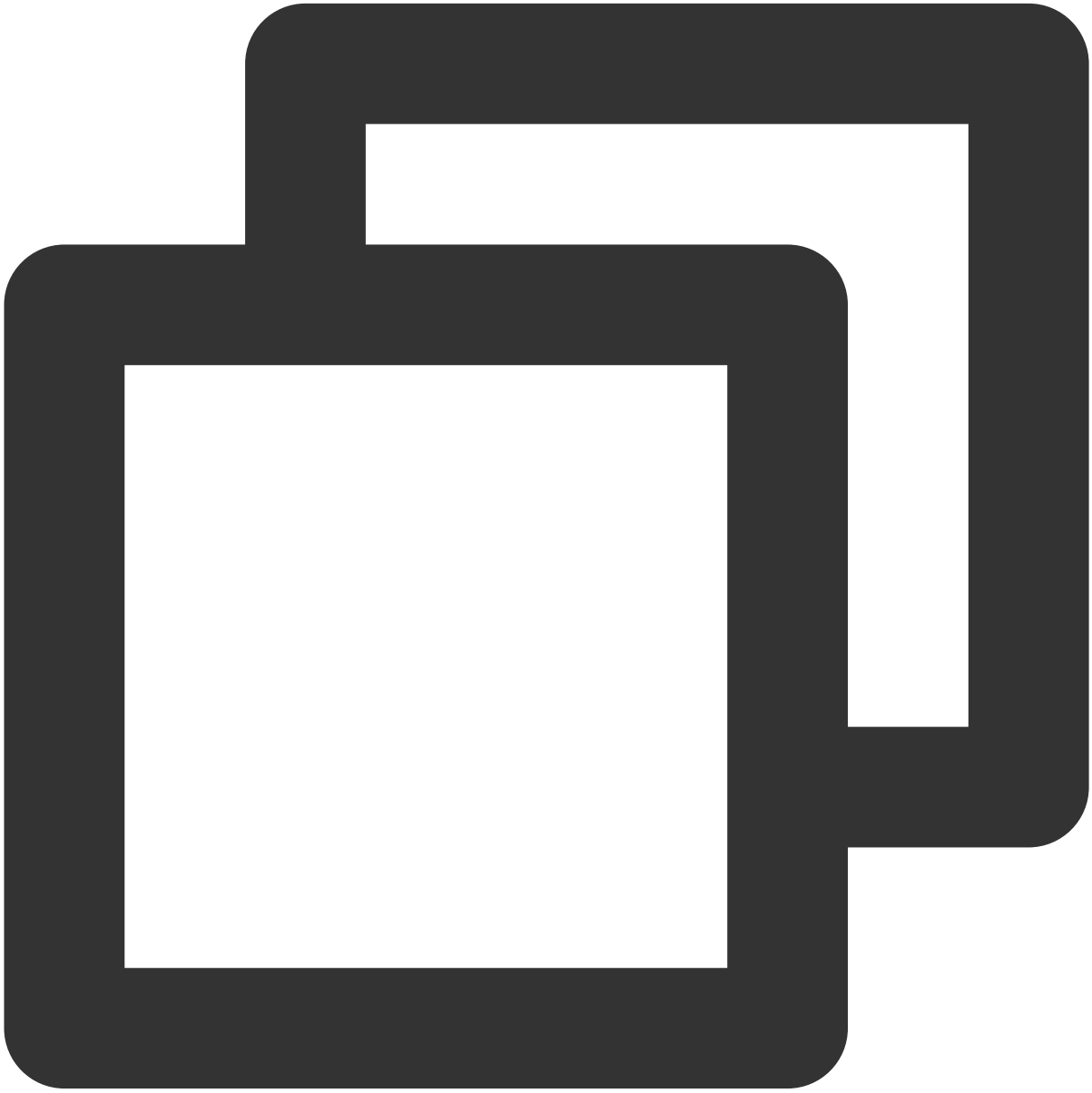
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Setting local video rendering parameters only affects the rendering effect of the local video.

Setting encoder output mode affects the viewing effect for other users in the room (and the cloud recording files).

3. The anchor starts the live streaming, entering the room and start streaming.



```
public void enterRoomByAnchor(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Specify the anchor role.
    params.role = TRTCCloudDef.TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

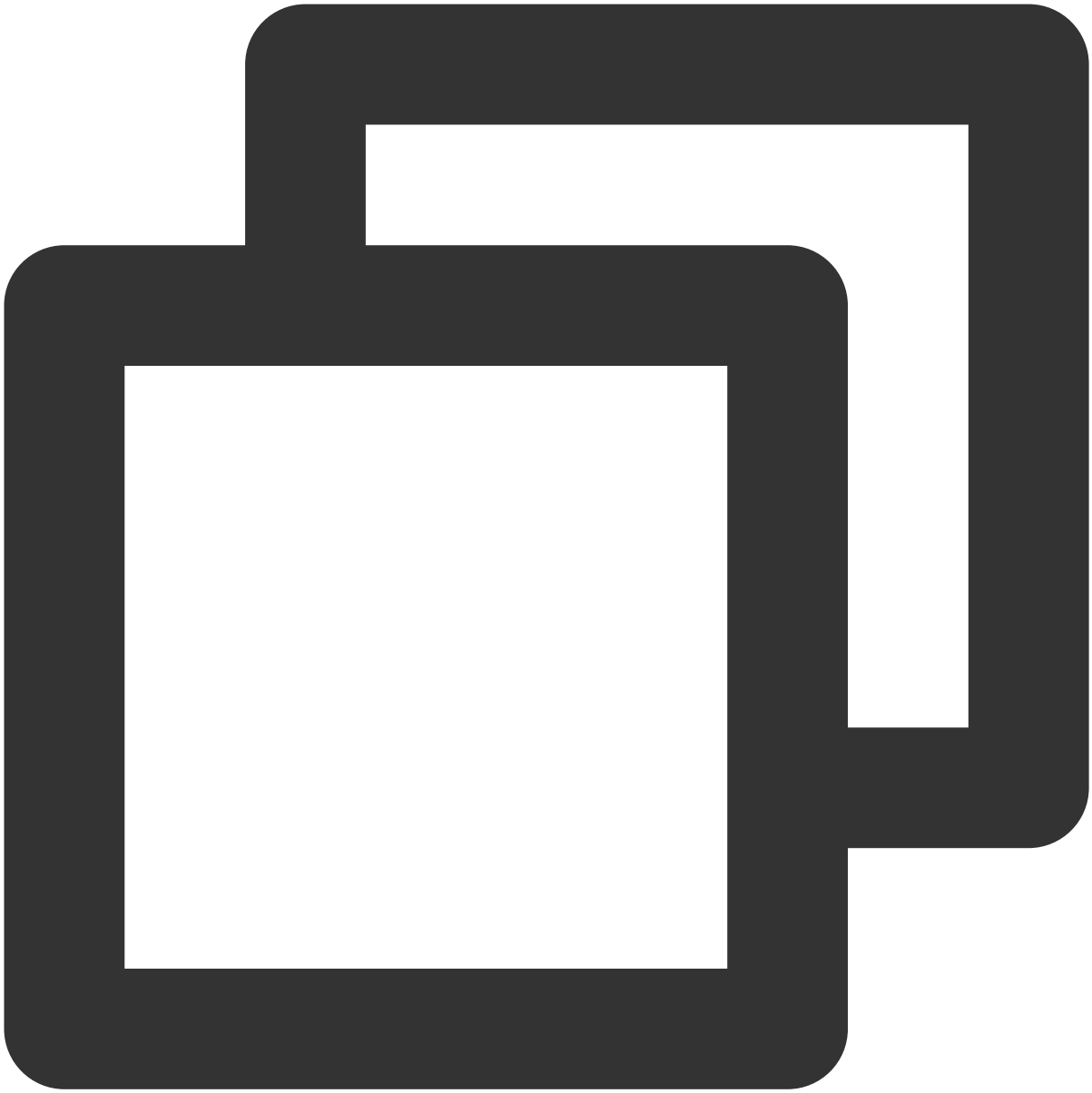
TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In live showroom scenarios, it is recommended to choose `TRTC_APP_SCENE_LIVE` as the room entry mode.

Step 2: The audience enters the room to pull streams.

1. Audience enters the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



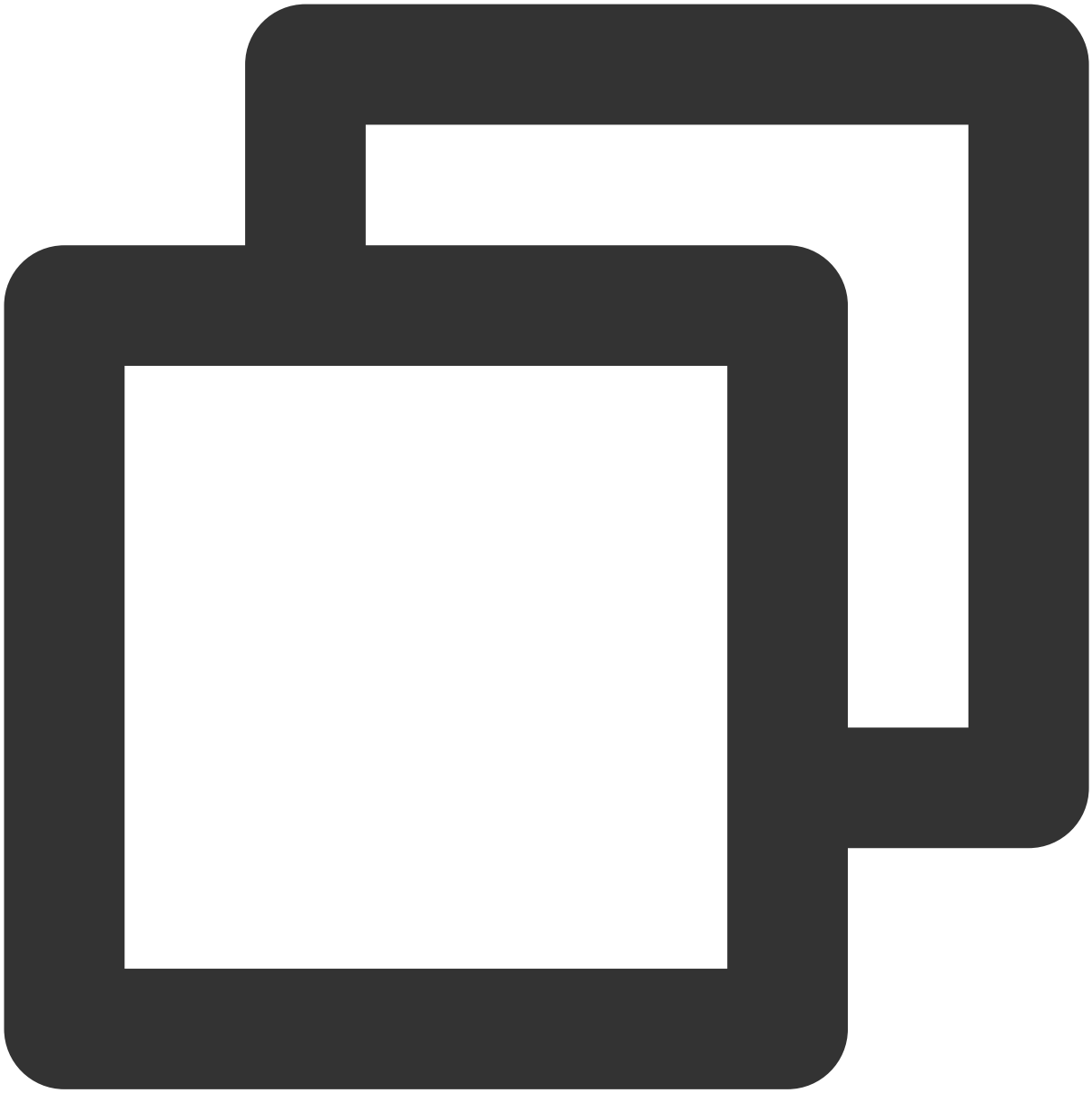
```
public void enterRoomByAudience(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Specify the audience role.
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

2. Audience subscribes to the anchor's audio and video streams.

©2013-2022 Tencent Cloud. All rights reserved.

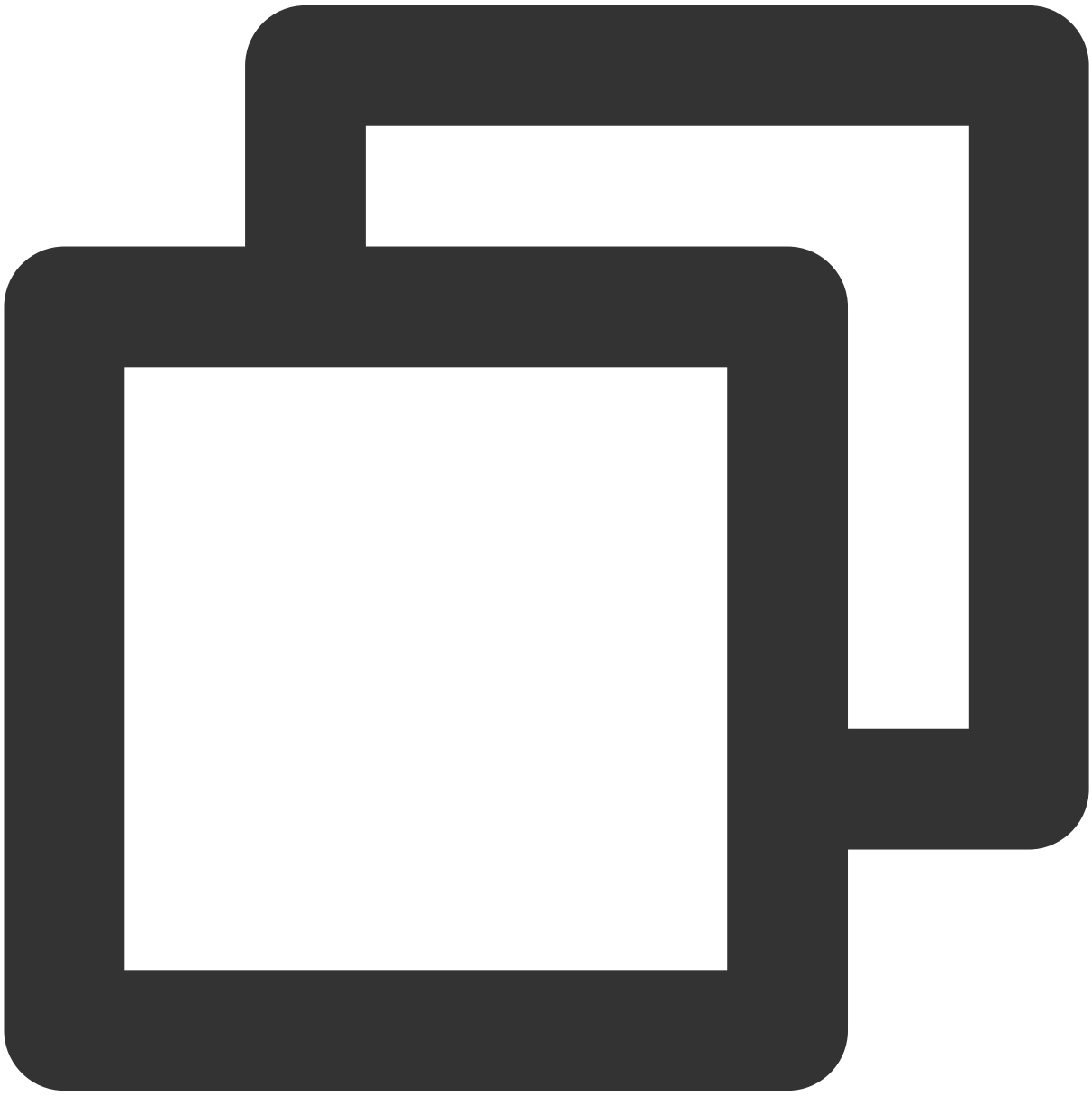


```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }
```

3. Audience sets the rendering mode for the remote video (optional).



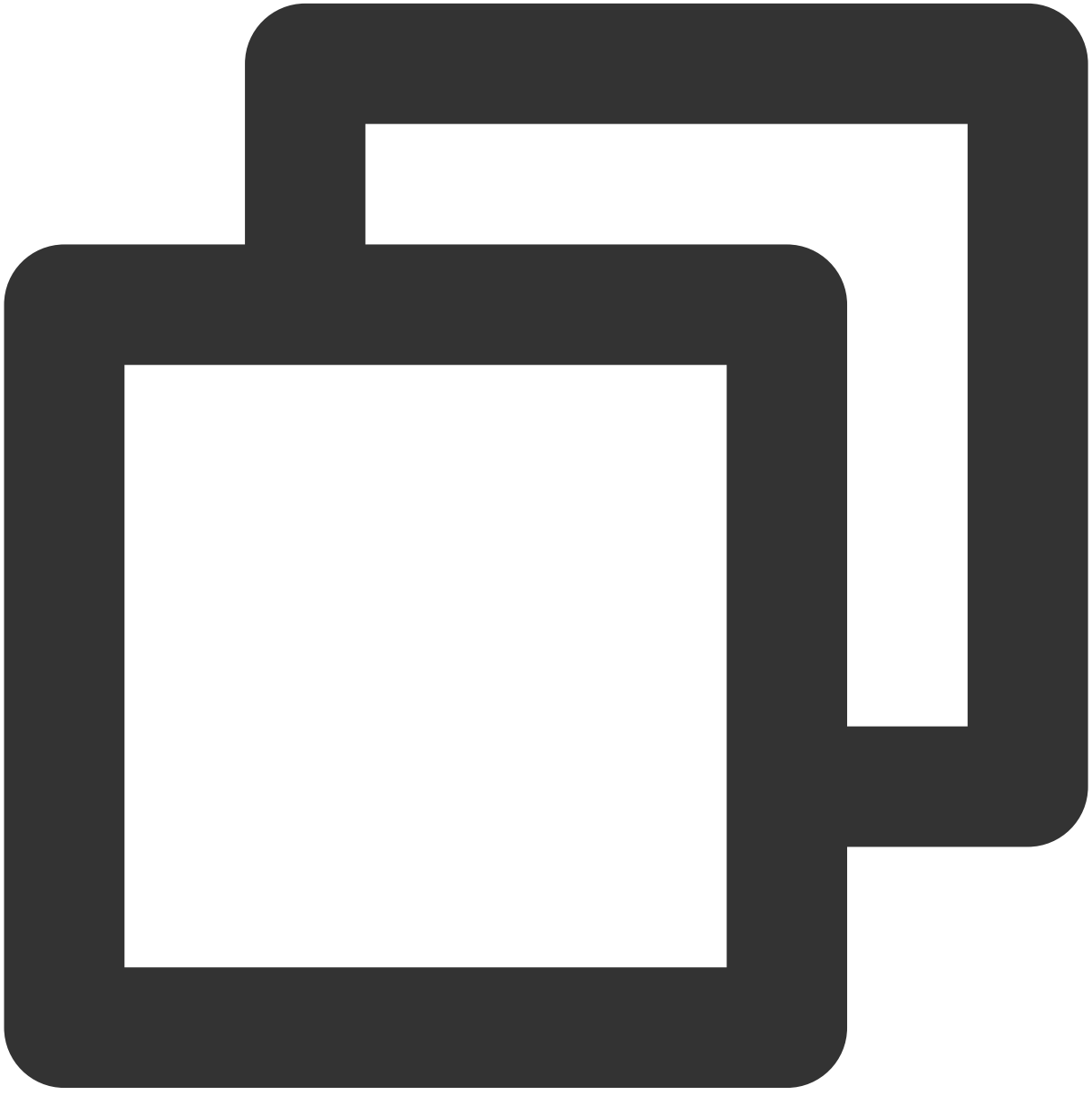
```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;     // Video fill mode
```

©2013-2022 Tencent Cloud. All rights reserved.

```
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering mode for the remote video.
mTRTCCloud.setRemoteRenderParams(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, p
```

Step 3: The audience interacts via mic.

1. The audience is switched to the anchor role.

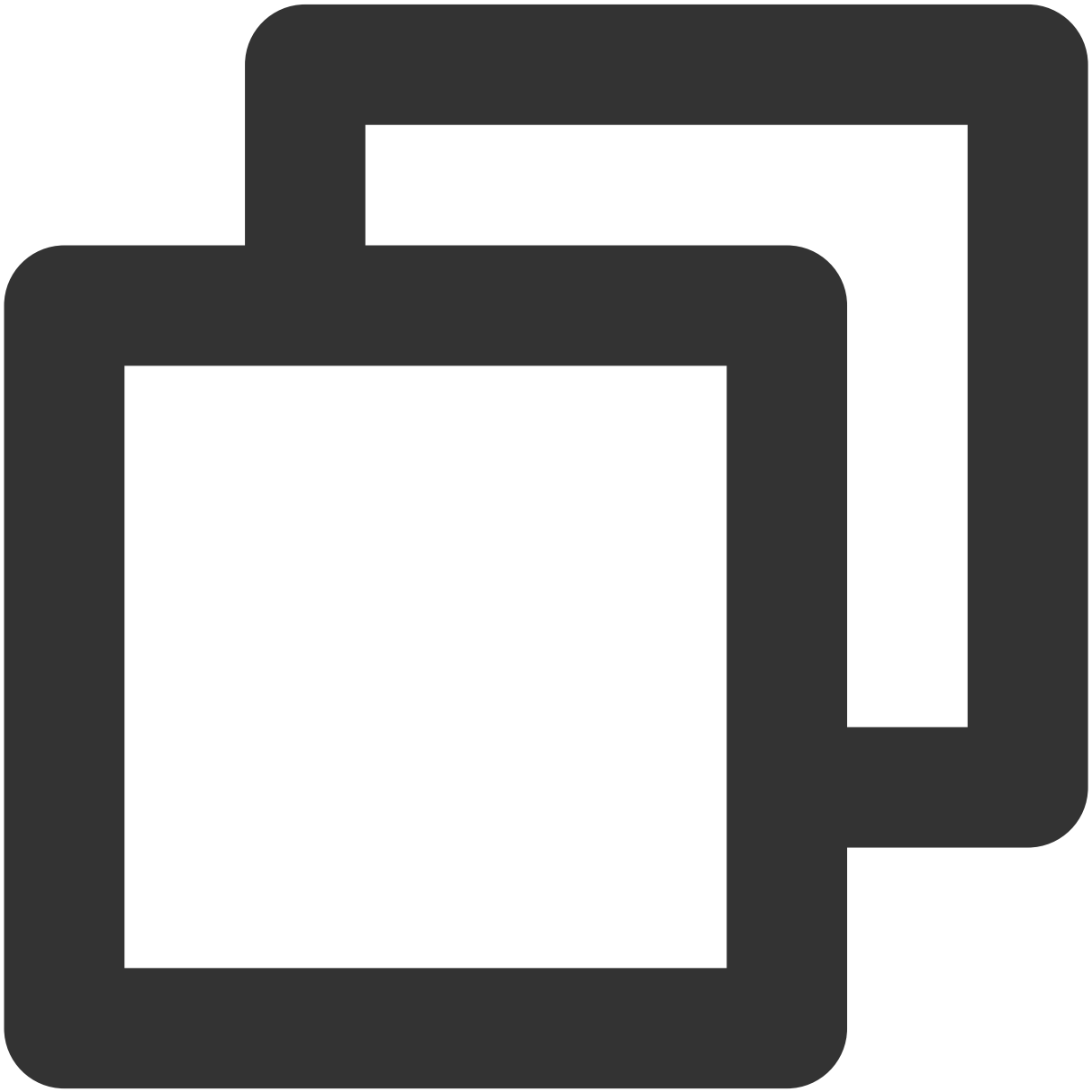


```
// Switched to the anchor role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Event callback for switching the role.
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Role switched successfully.
    }
}
```

2. Audience start local audio and video capture and streaming.



```
// Obtain the video rendering control for displaying the co-broadcasting audience's
TXCloudVideoView mTxcvvAudiencePreviewView = findViewById(R.id.live_cloud_view_sub)
```

©2013-2022 Tencent Cloud. All rights reserved.

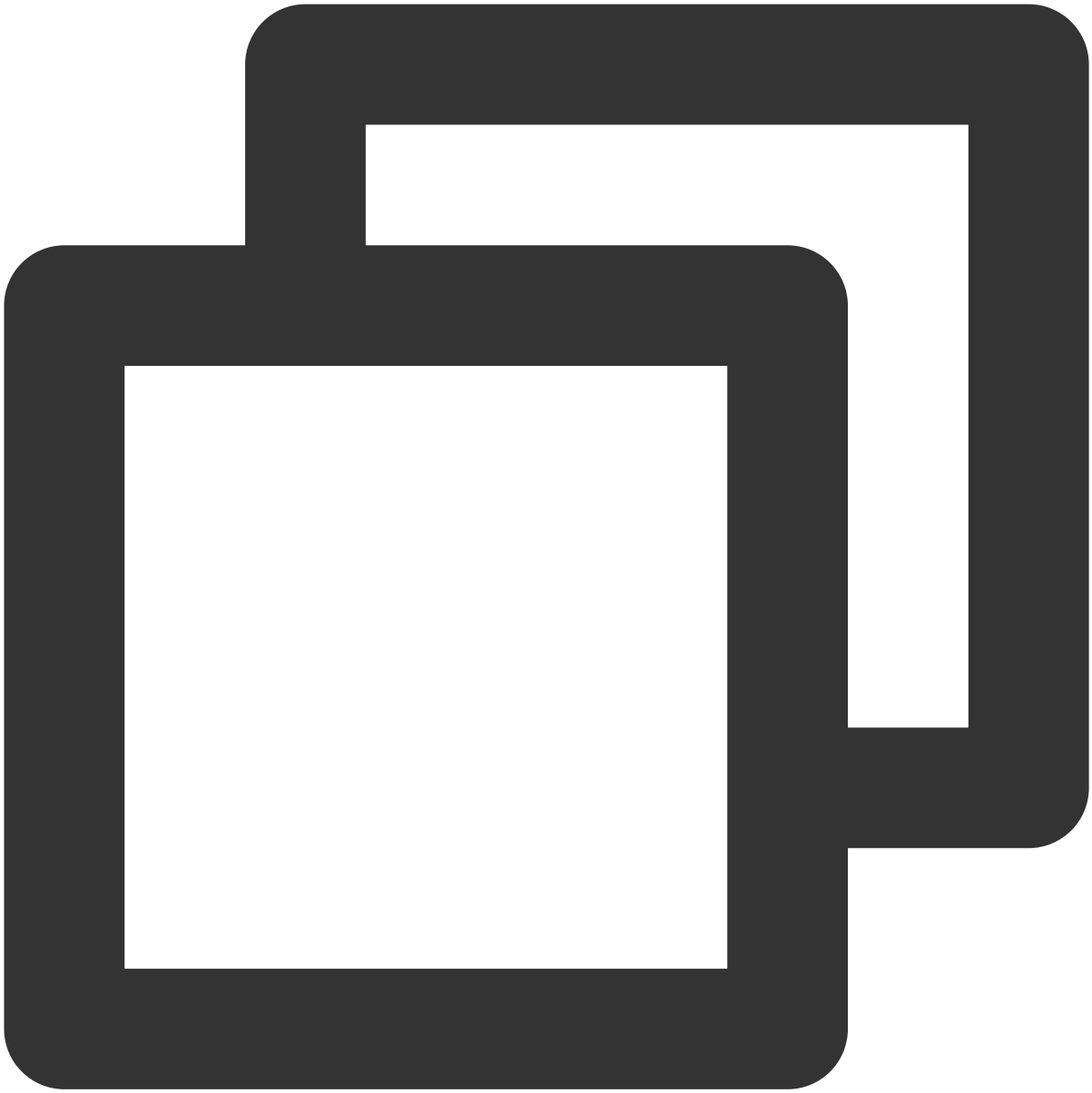
```
// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270;
encParam.videoFps = 15;
encParam.videoBitrate = 550;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);

// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAudiencePreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

3. The audience leaves the seat and stops streaming.

©2013-2022 Tencent Cloud. All rights reserved.



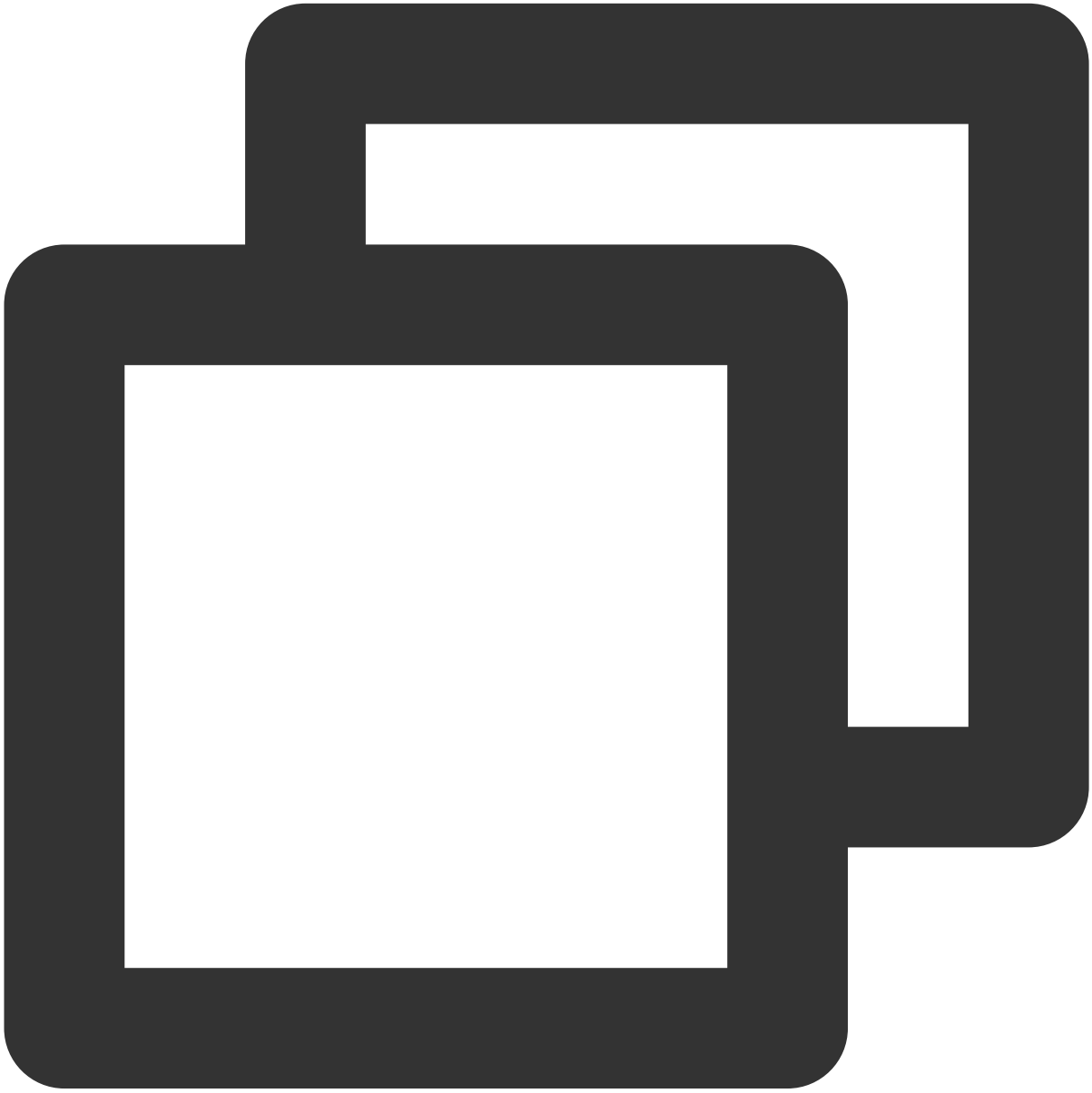
```
// Switched to the audience role.
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);

// Event callback for switching the role.
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Stop camera capture and streaming.
        mTRTCCloud.stopLocalPreview();
        // Stop microphone capture and streaming.
        mTRTCCloud.stopLocalAudio();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
}
```

Step 4: Exiting and dissolving the room.

1. Exit the room.



```
public void exitRoom() {
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.stopLocalPreview();
    mTRTCCloud.exitRoom();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}

// Event callback for exiting the room.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server.");
    } else if (reason == 2) {
        Log.d(TAG, "The current room has been dissolved.");
    }
}
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you want to call `enterRoom` again or switch to another audio/video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter exceptions such as the camera or microphone being forcefully occupied.

2. Dissolve the room.

Server Dissolvement: TRTC provides the Server dissolves the room API `DismissRoom` (differentiating between numeric room ID and string room ID). You can call this API to remove all users from the room and dissolve the room.

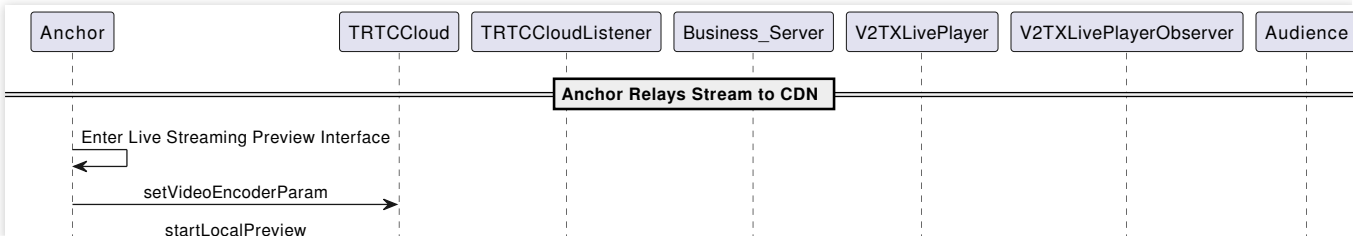
Client Dissolvement: Through the `exitRoom` API of each client, all the anchors and audiences in the room can be completed of room exit. After room exit, according to TRTC room lifecycle rules, the room will automatically be dissolved. For details, see Exit Room.

Warning:

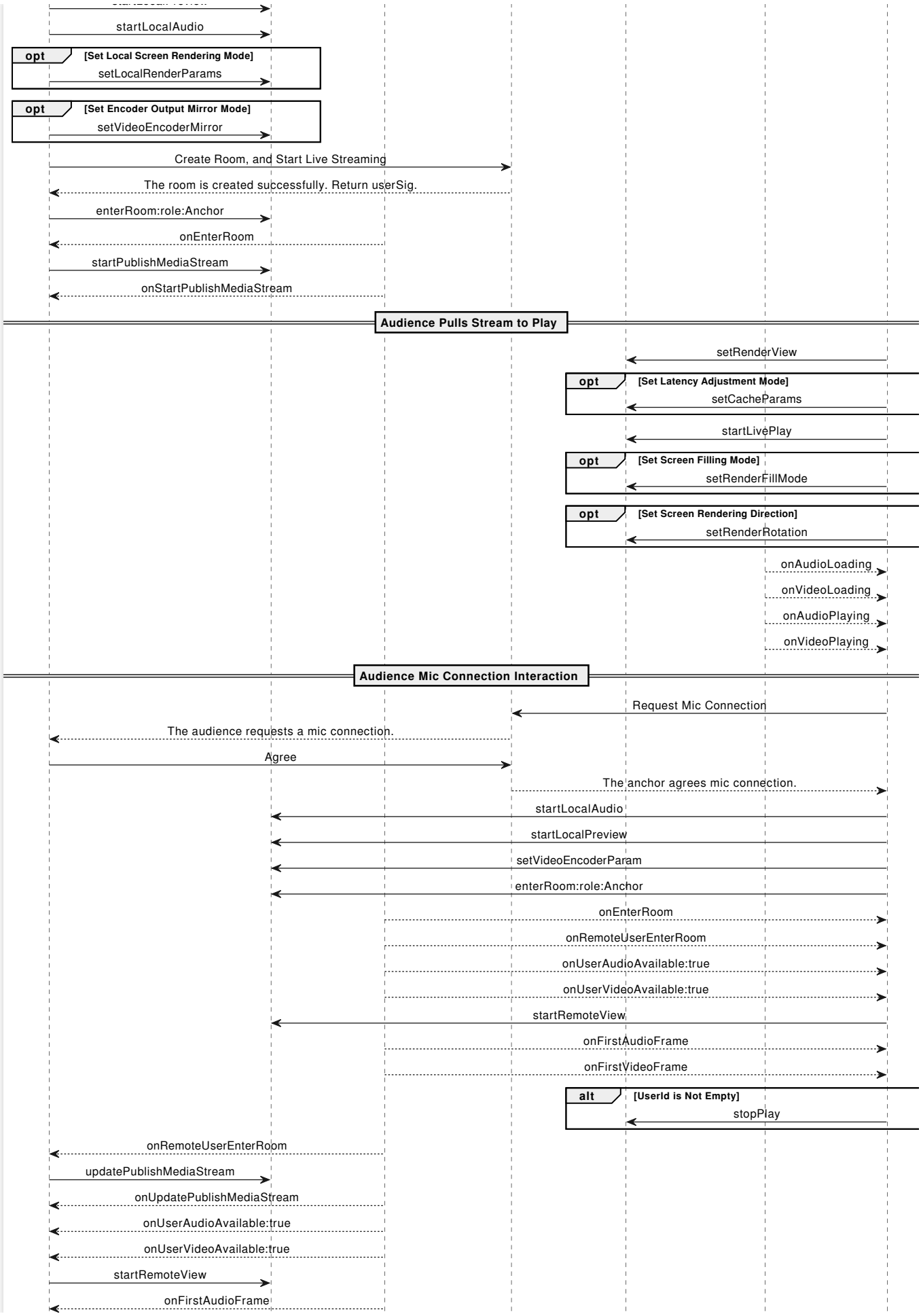
It is recommended that after the end of live streaming, you call the room dissolvement API on the server to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

Alternative solutions

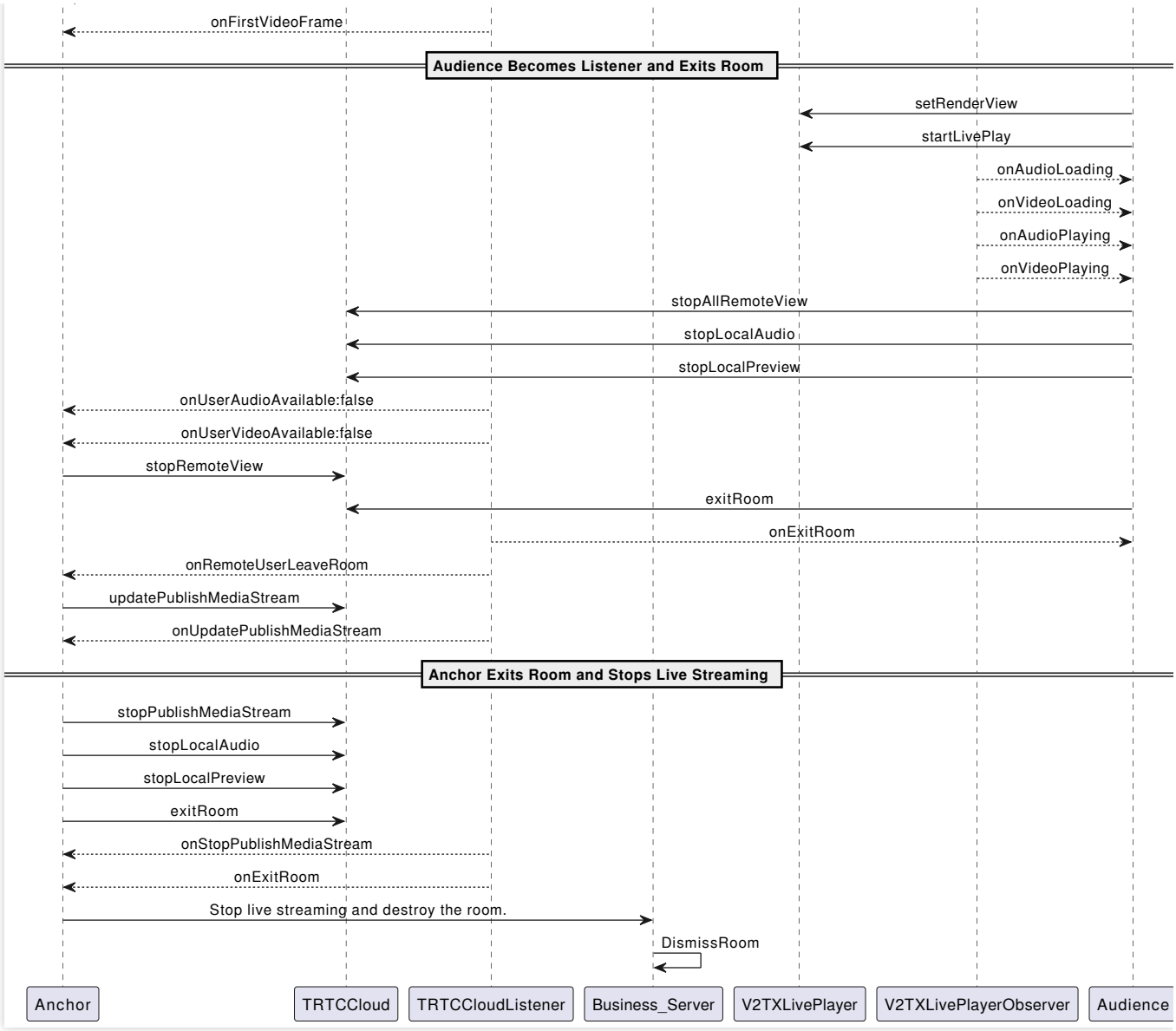
API sequence diagram.



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



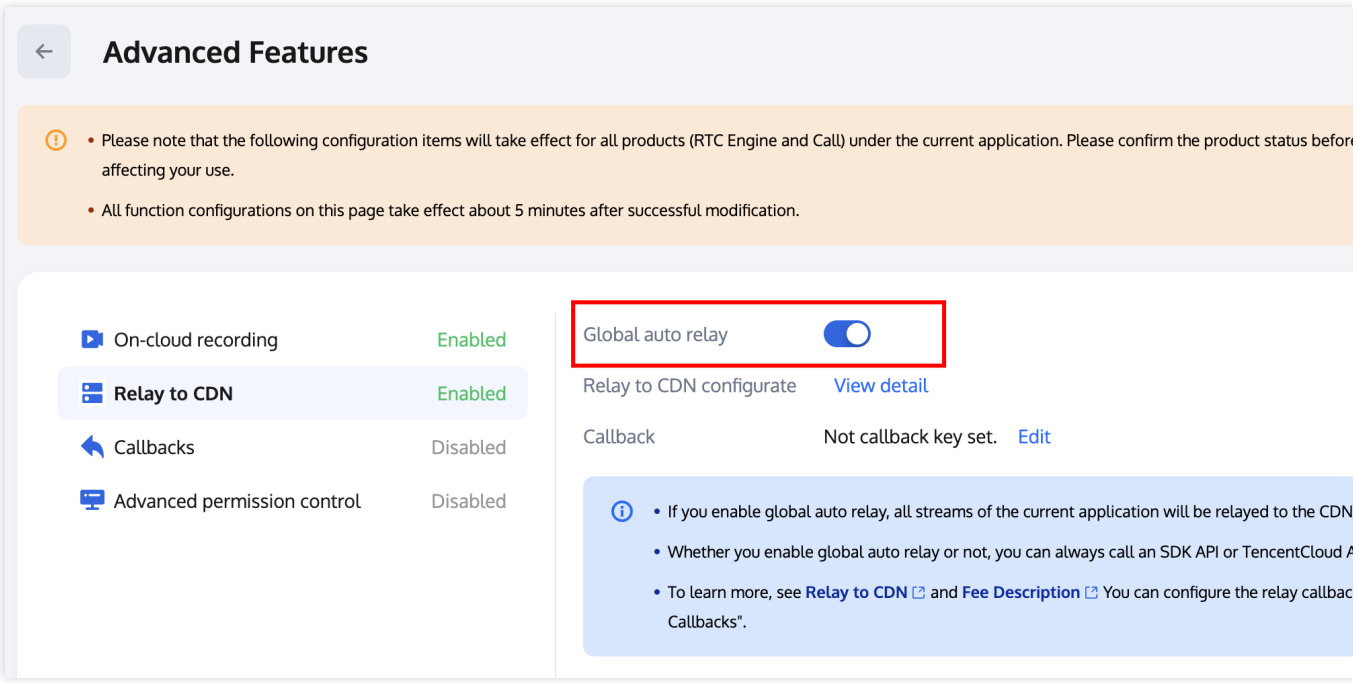
Step 1: The anchor relays the streams to CDN.

1. Related configurations for relaying to live streaming CDN.

Global automatic relayed push

If you need to automatically relay all anchors' audio and video streams in the room to live streaming CDN, you just need to enable **Relay to CDN** in the TRTC console **Advanced Features** page.

©2013-2022 Tencent Cloud. All rights reserved.



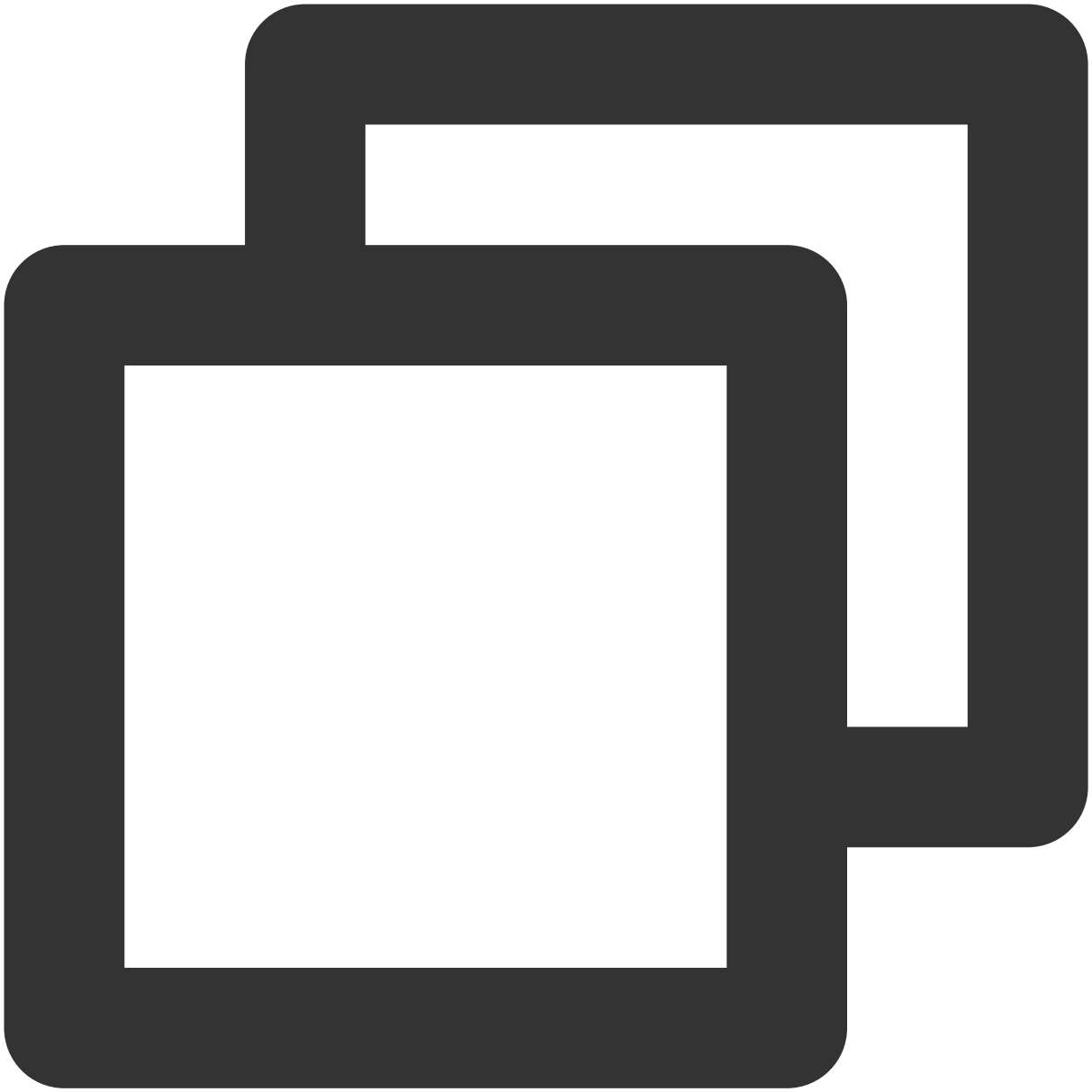
Relayed push of the specified streams

If you need to manually specify the audio and video streams to be published to live streaming CDN, or publish the mixed audio and video streams to live streaming CDN, you can do so by calling the startPublishMediaStream API. In this case, you do not need to activate global automatically relaying to CDN in the console. For detailed introduction, see Publish Audio and Video Streams to Live Streaming CDN.

2. The anchor activates local video preview and audio capture before entering the room.

The control used by the TRTC SDK to display video streams only supports passing in a `TXCloudVideoView` type. Therefore, you need to first define the view rendering control in the layout file.

©2013-2022 Tencent Cloud. All rights reserved.

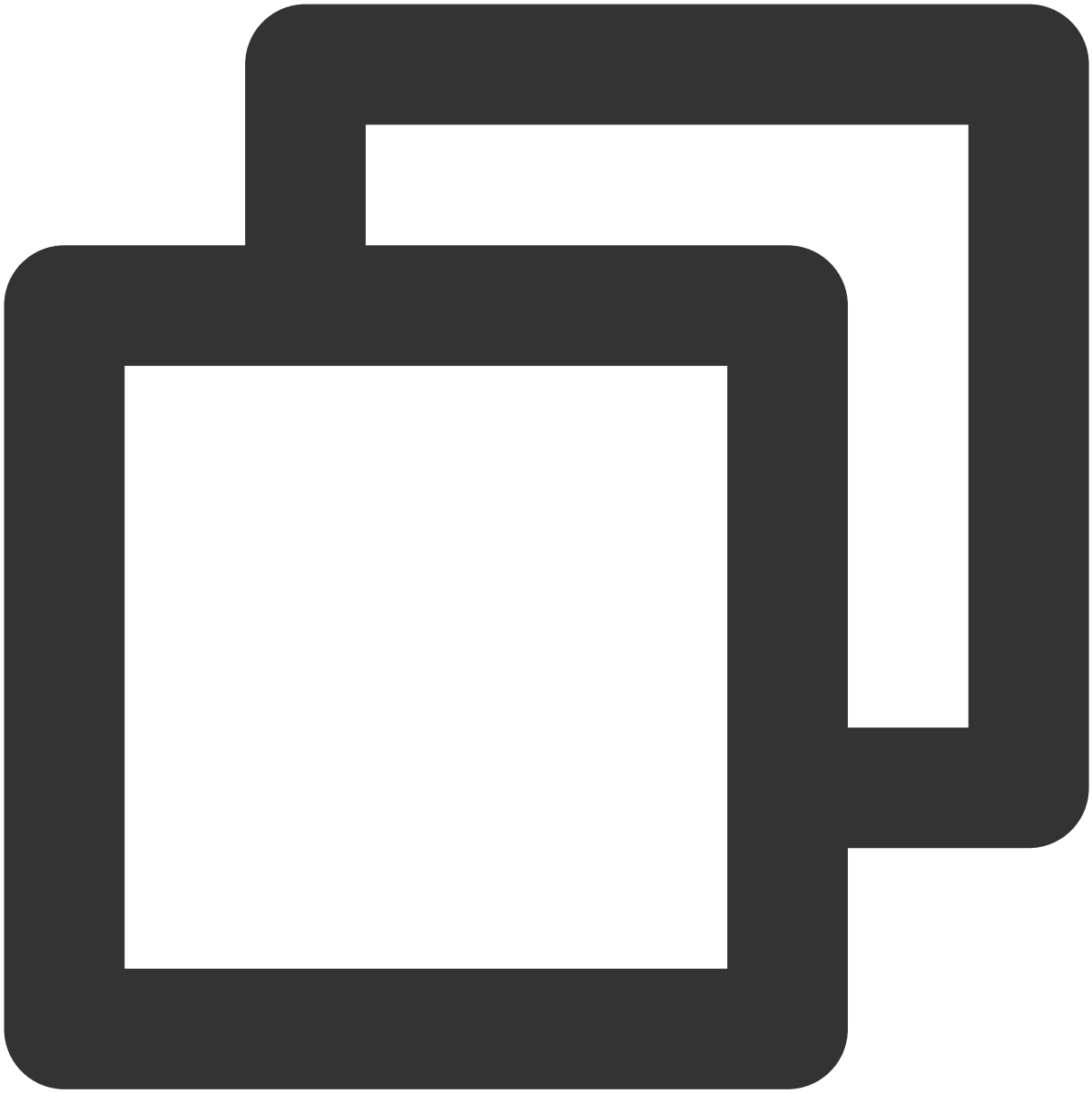


```
<com.tencent.rtmp.ui.TXCloudVideoView
    android:id="@+id/live_cloud_view_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Note:

If you need to specifically use `TextureView` or `SurfaceView` as the view rendering control, see Advanced Features - View Rendering Control.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
TXCloudVideoView mTxcvvAnchorPreviewView = findViewById(R.id.live_cloud_view_main);

// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
encParam.videoFps = 15;
encParam.videoBitrate = 1300;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAnchorPreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

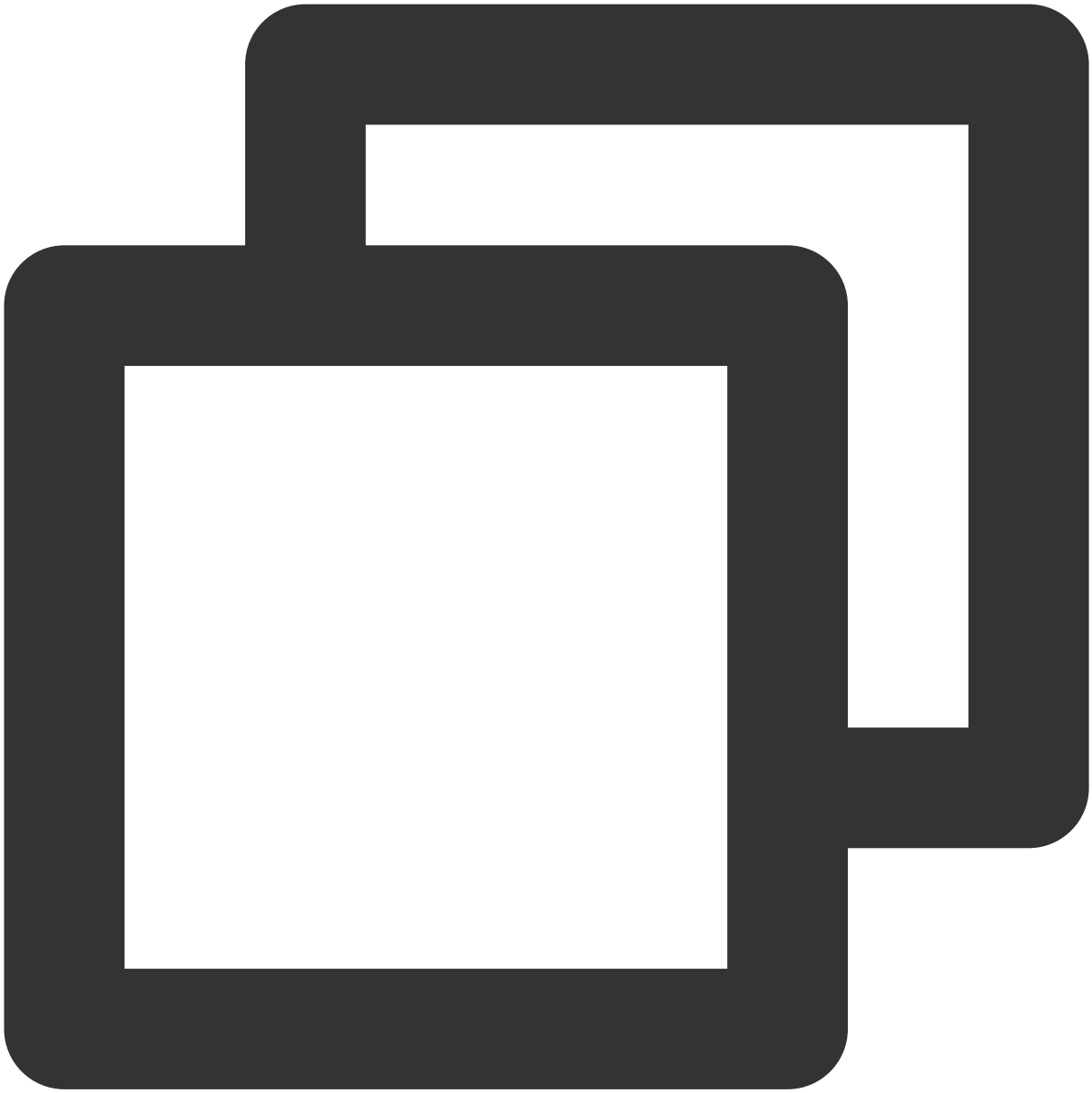
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

3. The anchor sets rendering parameters for the local screen, and the encoder output video mode.

©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;     // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering parameters for the local video.
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output.
mTRTCCloud.setVideoEncoderMirror(boolean mirror);
// Set the rotation of the video encoder output.
mTRTCCloud.setVideoEncoderRotation(int rotation);
```

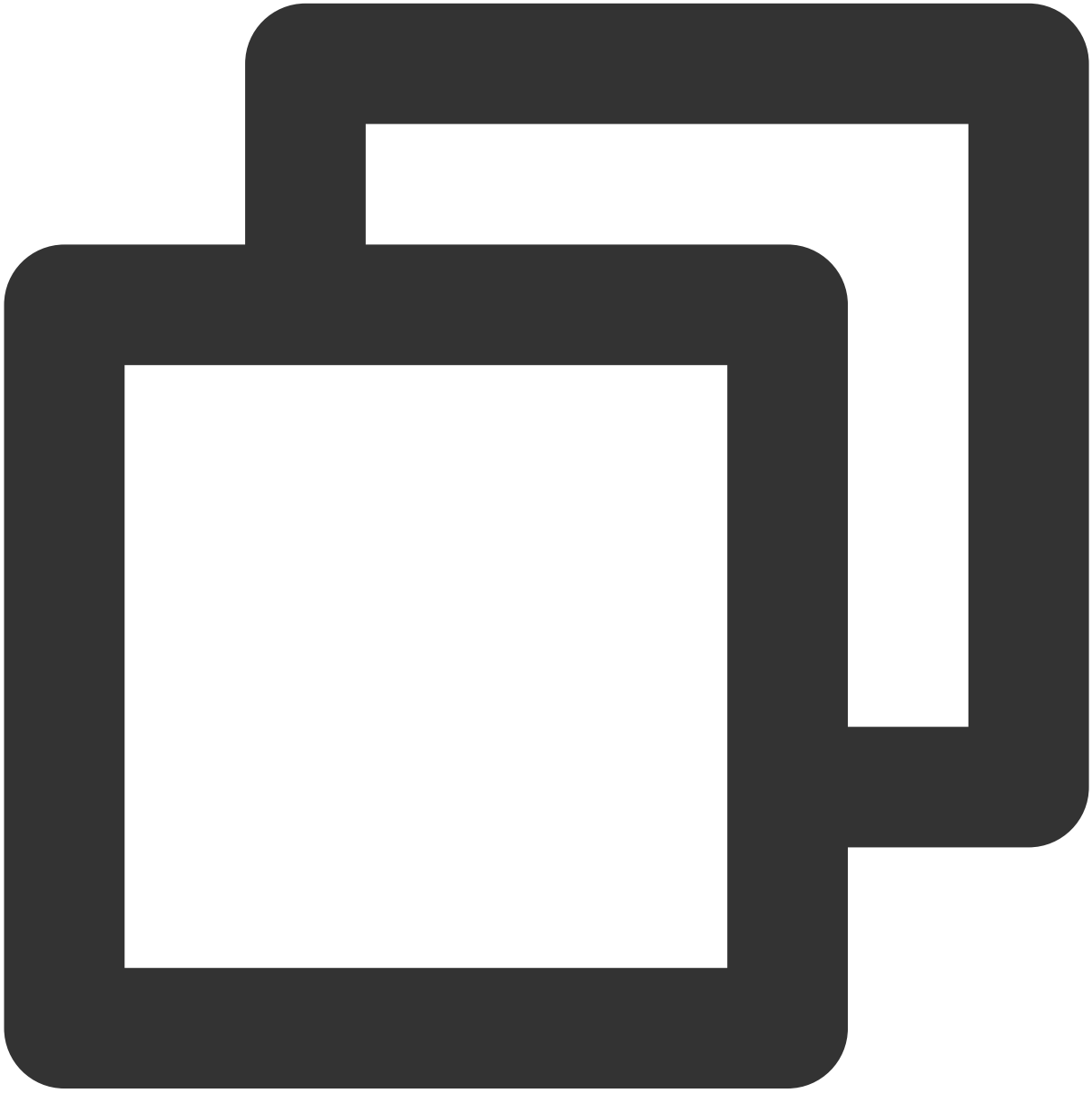
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Setting local video rendering parameters only affects the rendering effect of the local video.

Setting encoder output mode affects the viewing effect for other users in the room (and the cloud recording files).

4. The anchor starts the live streaming, entering the room and start streaming.



```
public void enterRoomByAnchor(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Specify the anchor role.
    params.role = TRTCCloudDef.TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");
    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

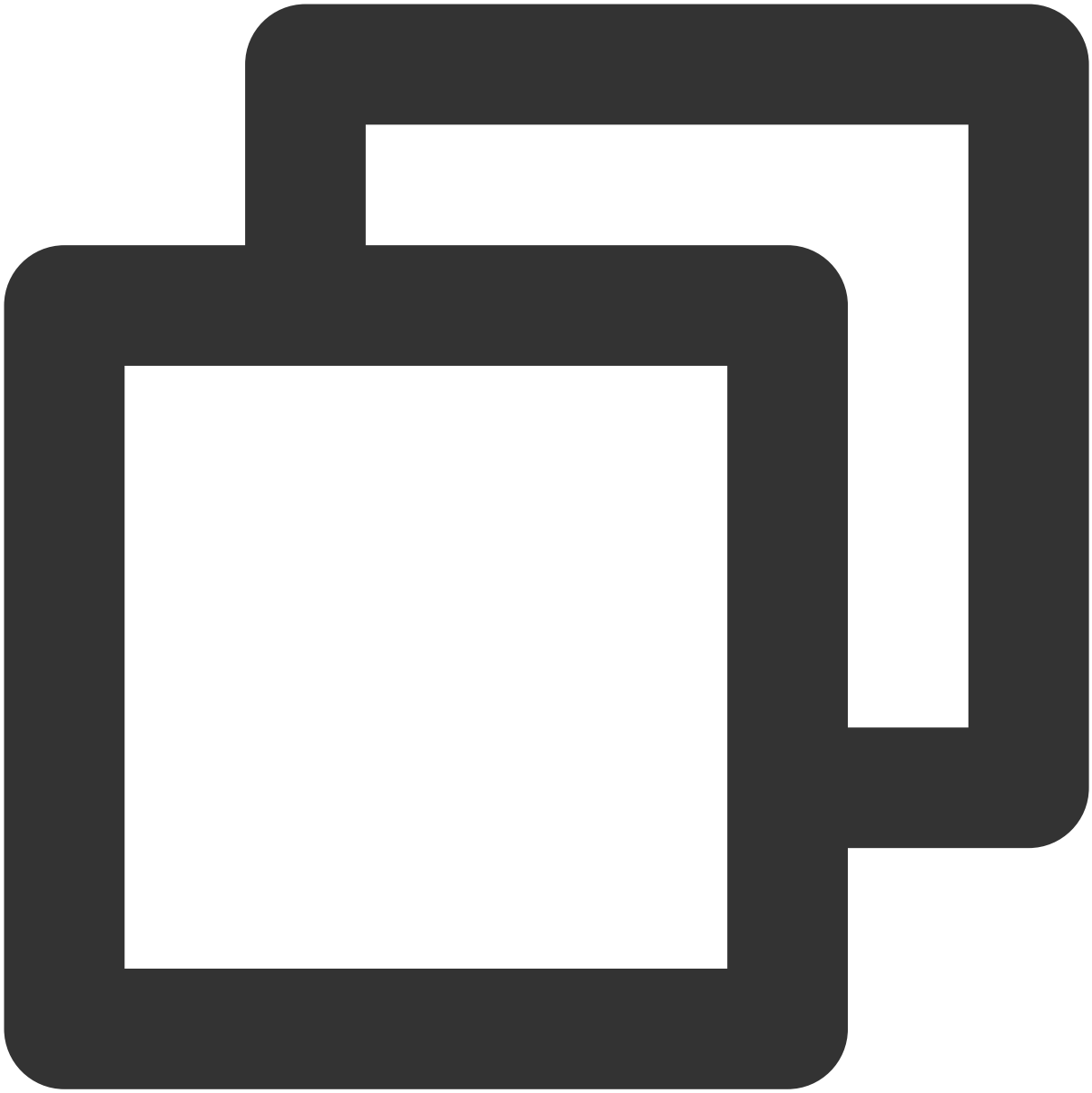
TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In live showroom scenarios, it is recommended to choose `TRTC_APP_SCENE_LIVE` as the room entry mode.

5. The anchor relays the audio and video streams to the live streaming CDN.

©2013-2022 Tencent Cloud. All rights reserved.



```
public void startPublishMediaToCDN(String streamName) {
    // Set the expiration time for the push URLs.
    long txTime = (System.currentTimeMillis() / 1000) + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    String secretParam = UrlHelper.getSafeUrl(LIVE_URL_KEY, streamName, txTime);

    // The target URLs for media stream publication.
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // The target URLs are set for relaying to CDN.
    target.mode = TRTCCloudDef.TRTC_PublishBigStream_ToCdn;
    TRTCCloudDef.TRTCPublishCdnUrl cdnUrl = new TRTCCloudDef.TRTCPublishCdnUrl();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Construct push URLs (in RTMP format) to the live streaming service provider.
    cdnUrl.rtmpUrl = "rtmp://" + PUSH_DOMAIN + "/live/" + streamName + "?" + secret
    // True means Tencent Cloud CSS, and false means third-party live streaming ser
    cdnUrl.isInternalLine = true;
    // Multiple CDN push URLs can be added.
    target.cdnUrlList.add(cdnUrl);

    // Set media stream encoding output parameters (can be defined according to bus
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
    trtcStreamEncoderParam.audioEncodedChannelNum = 1;
    trtcStreamEncoderParam.audioEncodedKbps = 50;
    trtcStreamEncoderParam.audioEncodedCodecType = 0;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 2;
    trtcStreamEncoderParam.videoEncodedKbps = 1300;
    trtcStreamEncoderParam.videoEncodedWidth = 540;
    trtcStreamEncoderParam.videoEncodedHeight = 960;

    // Start publishing media streams.
    mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam,  null);
}
```

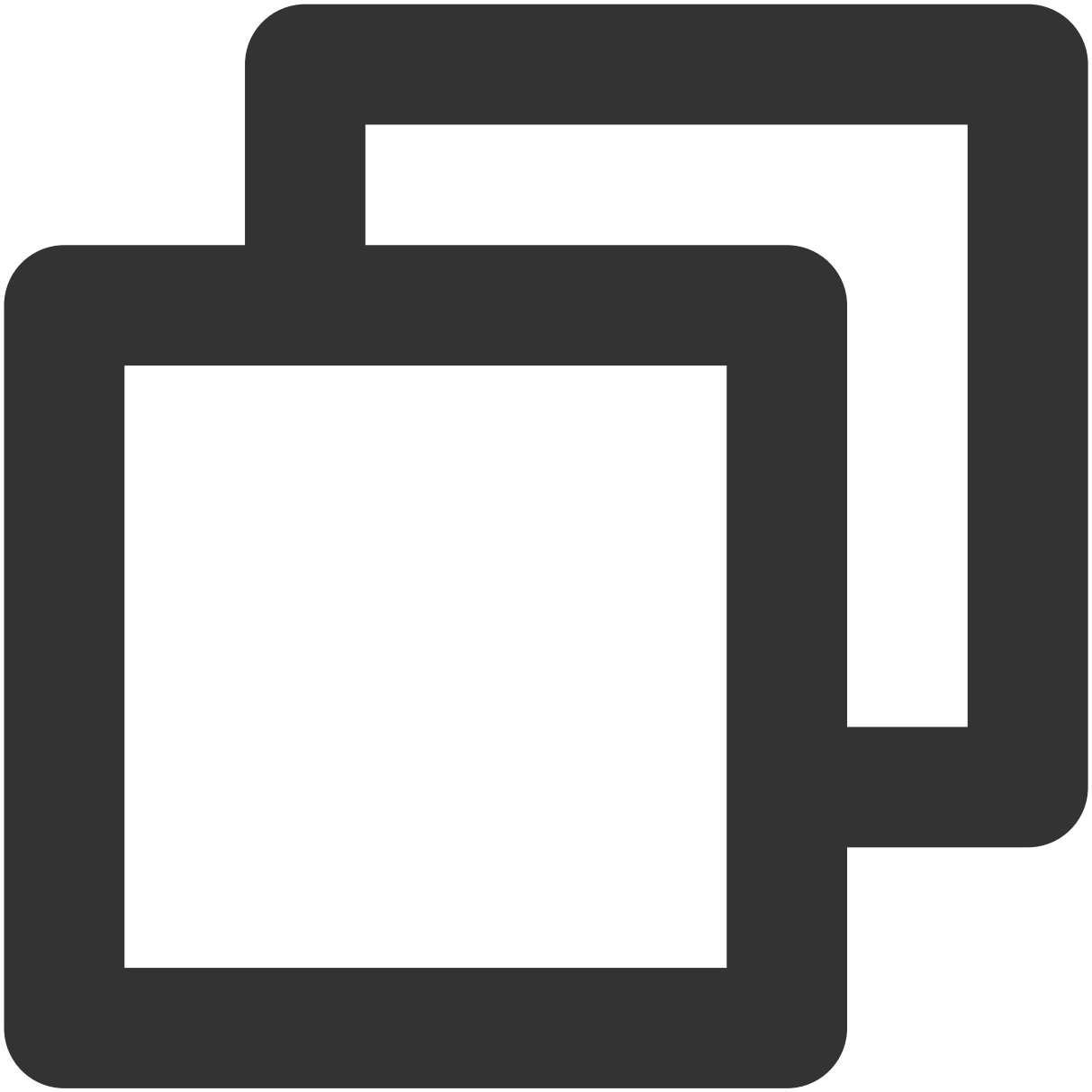
Note:

During single-anchor live streaming, only initiate the relayed push task. When there is an audience co-broadcasting or anchor PK, update this task to a mixed-stream transcoding task.

Information of push authentication KEY `LIVE_URL_KEY` and push domain name `PUSH_DOMAIN` are required to obtain in the CSS console **Domain Management** page.

After the media stream is published, SDK will provide the backend-initiated task identifier (taskId) through the callback onStartPublishMediaStream.

©2013-2022 Tencent Cloud. All rights reserved.

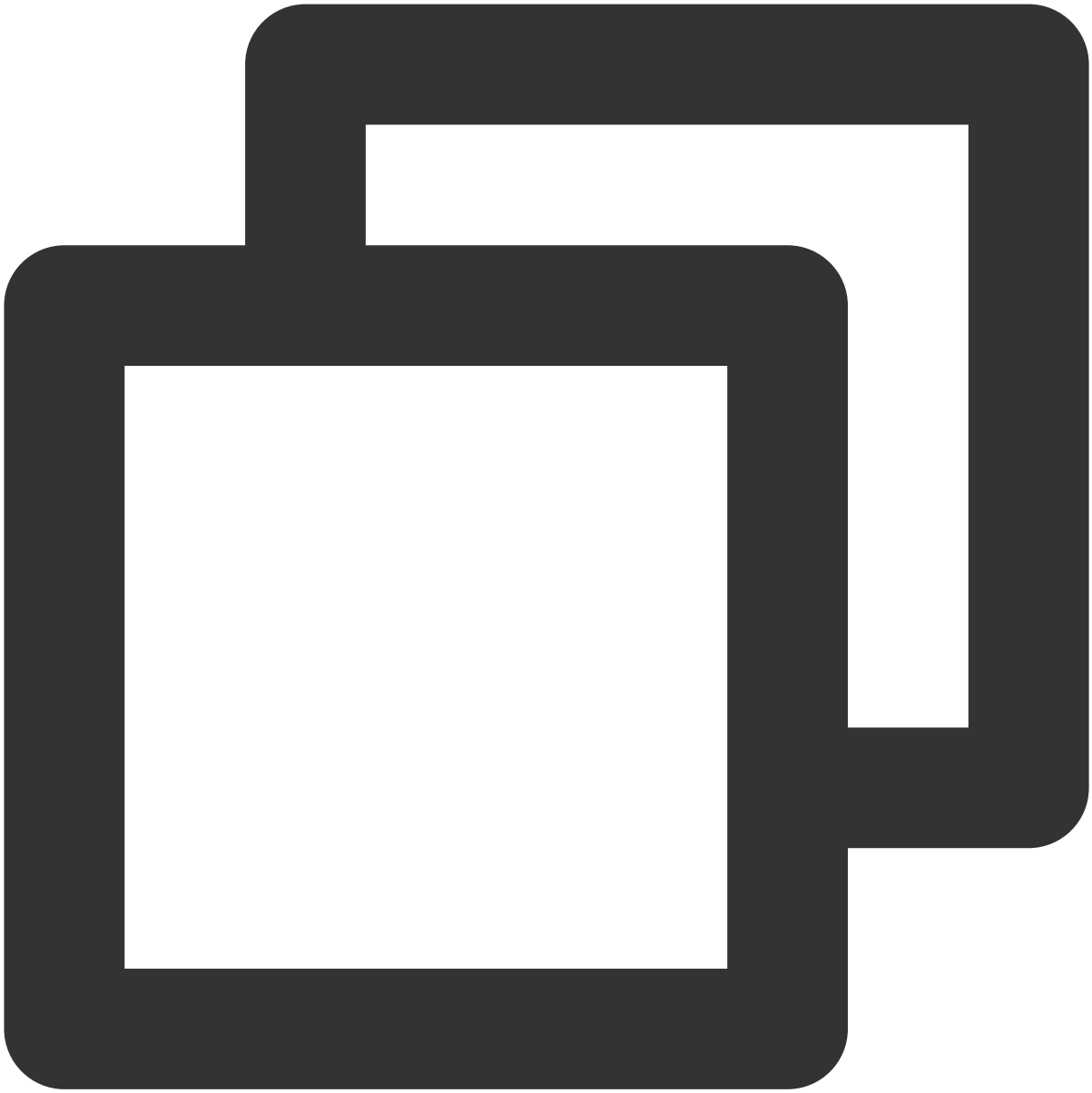


```
@Override
public void onStartPublishMediaStream(String taskId, int code, String message, Bund
    // taskId: When the request is successful, TRTC backend will provide the taskId
    // code: Callback result. 0 means success and other values mean failure.
}
```

Step 2: The audience pulls streams for playback.

CDN audiences do not need to enter the TRTC room. They can directly pull the anchor's CDN stream for playback.

©2013-2022 Tencent Cloud. All rights reserved.



```
import com.tencent.live2.V2TXLivePlayer;
import com.tencent.live2.V2TXLivePlayerObserver;
import com.tencent.live2.impl.V2TXLivePlayerImpl;

// Initialize the player.
V2TXLivePlayer mLivePlayer = new V2TXLivePlayerImpl(context);
// Set the player callback listener.
mLivePlayer.setObserver(mV2TXLivePlayerObserver);

// Set the video rendering control for the player.
mLivePlayer.setRenderView(TXCloudVideoView view);
```

©2013-2022 Tencent Cloud. All rights reserved.

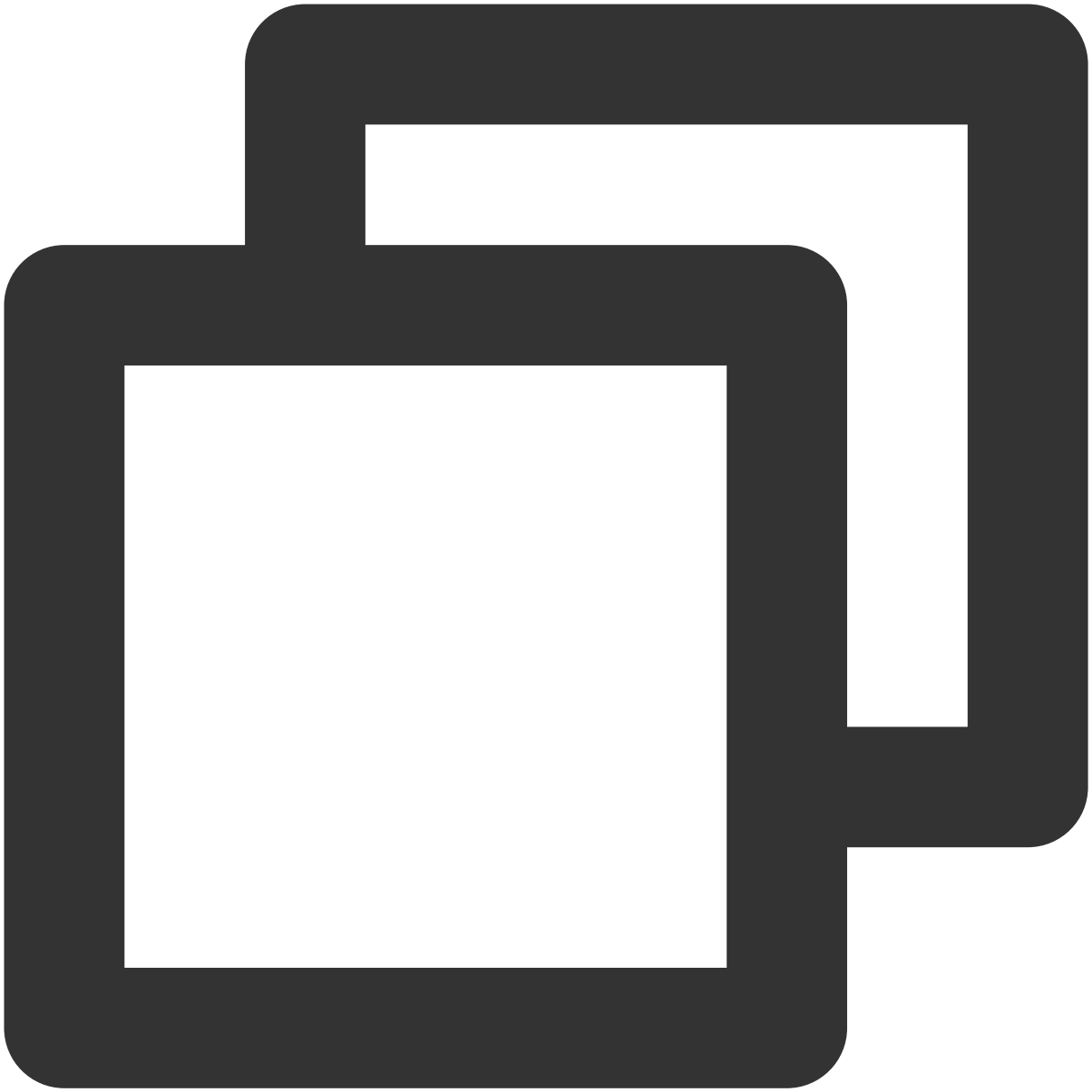
```
mLivePlayer.setRenderView(TextureView view);
mLivePlayer.setRenderView(SurfaceView view);

// Set delay management mode (optional).
mLivePlayer.setCacheParams(1.0f, 5.0f); // Auto mode
mLivePlayer.setCacheParams(1.0f, 1.0f); // Speed mode
mLivePlayer.setCacheParams(5.0f, 5.0f); // Smooth mode

// Concatenate the pull URLs for playback.
String flvURL = "http://" + PLAY_DOMAIN + "/live/" + streamName + ".flv"; // FLV UR
String hlsURL = "http://" + PLAY_DOMAIN + "/live/" + streamName + ".m3u8"; // HLS U
String rtmpURL = "rtmp://" + PLAY_DOMAIN + "/live/" + streamName; // RTMP URL
String webrtcURL = "webrtc://" + PLAY_DOMAIN + "/live/" + streamName; // WebRTC URL

// Start playing.
mLivePlayer.startLivePlay(flvURL);

// Custom set fill mode (optional).
mLivePlayer.setRenderFillMode(V2TXLiveFillModeFit);
// Custom video rendering direction (optional).
mLivePlayer.setRenderRotation(V2TXLiveRotation0);
```

Note:

The playback domain name `PLAY_DOMAIN` requires you to Add Your Own Domain in the CSS console for live streaming playback. You also should configure domain CNAME.

The live streaming feature requires setting the License before success in playback. Otherwise, playback will fail (black screen). It needs to be set globally only once. If you have not obtained the License, you can freely apply for a Trial Version License for normal playback. The Official Version License requires purchase.

©2013-2022 Tencent Cloud. All rights reserved.

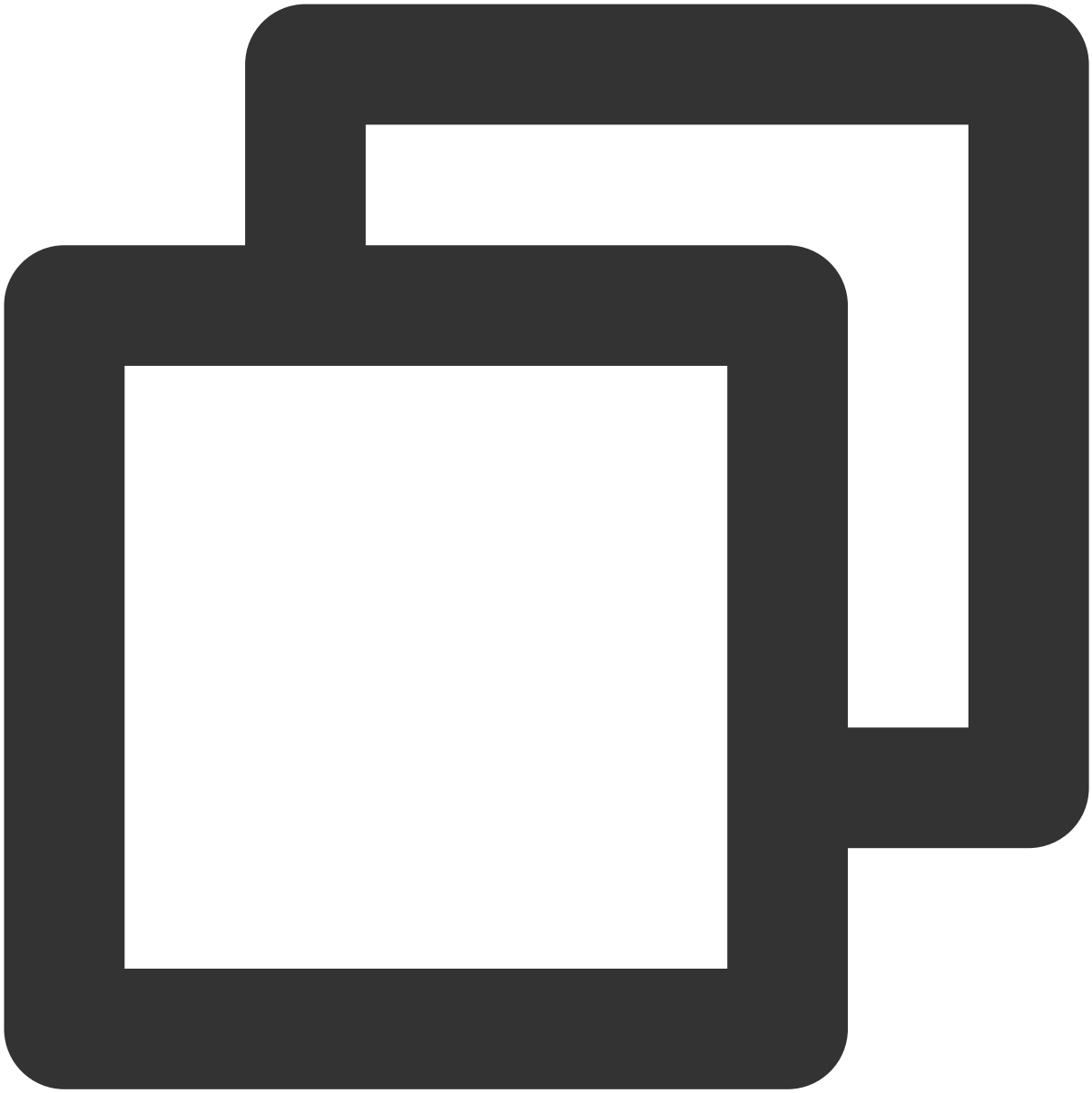


```
import com.tencent.rtmp.TXLiveBase;

// Set Licence.
TXLiveBase.getInstance().setLicence(context, LICENSEURL, LICENSEURLKEY);
```

Step 3: The audience interacts via mic.

1. Viewers who want to co-broadcasting need to enter the TRTC room for real-time interaction with the anchor.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enter the TRTC room and start streaming.
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID.
    params.sdkAppId = SDKAppID;
    // Specify the anchor role.
```

©2013-2022 Tencent Cloud. All rights reserved.

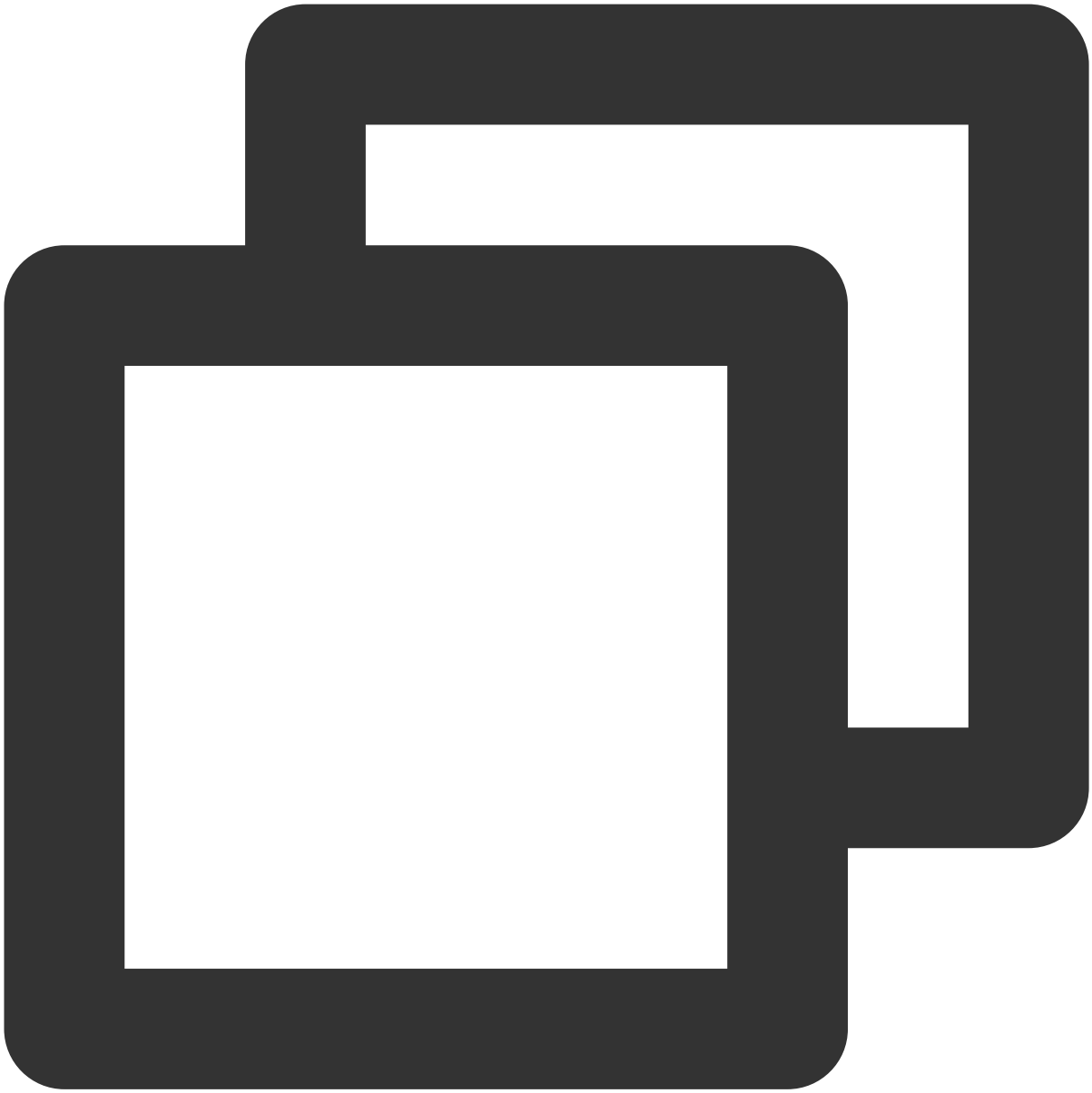
```
    params.role = TRTCCloudDef.TRTCRoleAnchor;

    // Enable local audio and video capture.
    startLocalMedia();
    // In an interactive live streaming scenario, enter the room and push streams.
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Enable local video preview and audio capture.
public void startLocalMedia() {
    // Obtain the video rendering control for displaying the co-broadcasting audien
    TXCloudVideoView mTxcvvAudiencePreviewView = findViewById(R.id.live_cloud_view_

    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
    encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT
    mTRTCCloud.setVideoEncoderParam(encParam);

    // boolean mIsFrontCamera can specify using the front/rear camera for video cap
    mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAudiencePreviewView);
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
}

// Event callback for the result of entering the room.
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        Log.d(TAG, "Enter room succeed");

    } else {
        // result indicates the error code when you fail to enter the room.
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

2. The mic-connection audience start subscribing to the anchor's audio and video streams after they successfully enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }

    @Override
    public void onFirstVideoFrame(String userId, int streamType, int width, int height)
        // The SDK starts rendering the first frame of the local or remote user's video
        if (!userId.isEmpty()) {
            // Stop playing the CDN stream upon receiving the first frame of the anchor
            mLivePlayer.stopPlay();
        }
    }
```

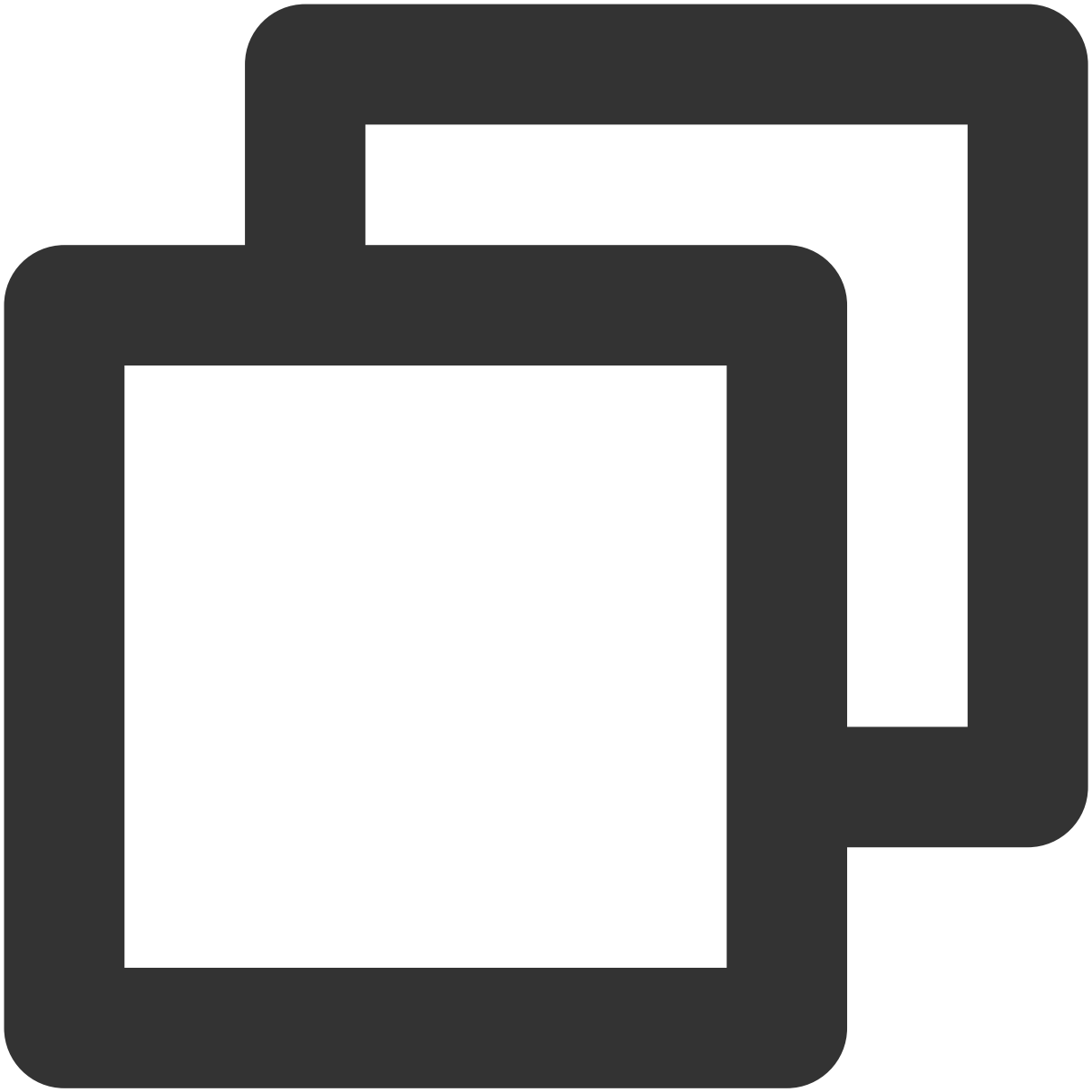
Note:

TRTC stream pulling `startRemoteView` can directly reuse the video rendering control previously used by the CDN stream pulling `setRenderView` .

To avoid video interruptions when switching between stream pullers, it is recommended to wait until the TRTC first frame callback `onFirstVideoFrame` is received before stopping the CDN stream pulling.

3. The anchor updates the publication of mixed media streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the mic-connection audience's room entry.
@Override
public void onRemoteUserEnterRoom(String userId) {
    if (!mixUserList.contains(userId)) {
        mixUserList.add(userId);
    }
    updatePublishMediaToCDN(streamName, mixUserList, taskId);
}

// Event callback for updating the media stream.
@Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
public void onUpdatePublishMediaStream(String taskId, int code, String message, Bun
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure.
}

// Update the publication of mixed media streams to the live streaming CDN.
public void updatePublishMediaToCDN(String streamName, List<String> mixUserList, St
    // Set the expiration time for the push URLs.
    long txTime = (System.currentTimeMillis() / 1000) + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    String secretParam = UrlHelper.getSafeUrl(LIVE_URL_KEY, streamName, txTime);

    // The target URLs for media stream publication.
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // The target URLs are set for relaying the mixed streams to CDN.
    target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToCdn;
    TRTCCloudDef.TRTCPublishCdnUrl cdnUrl = new TRTCCloudDef.TRTCPublishCdnUrl();
    // Construct push URLs (in RTMP format) to the live streaming service provider.
    cdnUrl.rtmpUrl = "rtmp://" + PUSH_DOMAIN + "/live/" + streamName + "?" + secret
    // True means Tencent Cloud CSS, and false means third-party live streaming ser
    cdnUrl.isInternalLine = true;
    // Multiple CDN push URLs can be added.
    target.cdnUrlList.add(cdnUrl);

    // Set media stream encoding output parameters.
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
    trtcStreamEncoderParam.audioEncodedChannelNum = 1;
    trtcStreamEncoderParam.audioEncodedKbps = 50;
    trtcStreamEncoderParam.audioEncodedCodecType = 0;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 2;
    trtcStreamEncoderParam.videoEncodedKbps = 1300;
    trtcStreamEncoderParam.videoEncodedWidth = 540;
    trtcStreamEncoderParam.videoEncodedHeight = 960;

    // Configuration parameters for media stream transcoding.
    TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new TRTCCloudDef.T
    if (mixUserList != null) {
        ArrayList<TRTCCloudDef.TRTCUser> audioMixUserList = new ArrayList<>();
        ArrayList<TRTCCloudDef.TRTCVideoLayout> videoLayoutList = new ArrayList<>()

        for (int i = 0; i < mixUserList.size() && i < 16; i++) {
            TRTCCloudDef.TRTCUser user = new TRTCCloudDef.TRTCUser();
            // The integer room number is intRoomId.
            user.strRoomId = mRoomId;
            user.userId = mixUserList.get(i);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            audioMixUserList.add(user);

            TRTCCloudDef.TRTCVideoLayout videoLayout = new TRTCCloudDef.TRTCVideoLa
            if (mixUserList.get(i).equals(mUserId)) {
                // The layout for the anchor's video.
                videoLayout.x = 0;
                videoLayout.y = 0;
                videoLayout.width = 540;
                videoLayout.height = 960;
                videoLayout.zOrder = 0;
            } else {
                // The layout for the mic-connection audience's video.
                videoLayout.x = 400;
                videoLayout.y = 5 + i * 245;
                videoLayout.width = 135;
                videoLayout.height = 240;
                videoLayout.zOrder = 1;
            }
            videoLayout.fixedVideoUser = user;
            videoLayout.fixedVideoStreamType = TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_
            videoLayoutList.add(videoLayout);
        }

        // Specify the information for each input audio stream in the transcoding s
        trtcStreamMixingConfig.audioMixUserList = audioMixUserList;
        // Specify the information of position, size, layer, and stream type for ea
        trtcStreamMixingConfig.videoLayoutList = videoLayoutList;
    }

    // Update the published media stream.
    mTRTCCloud.updatePublishMediaStream(taskId, target, trtcStreamEncoderParam, trt
}
```

Note:

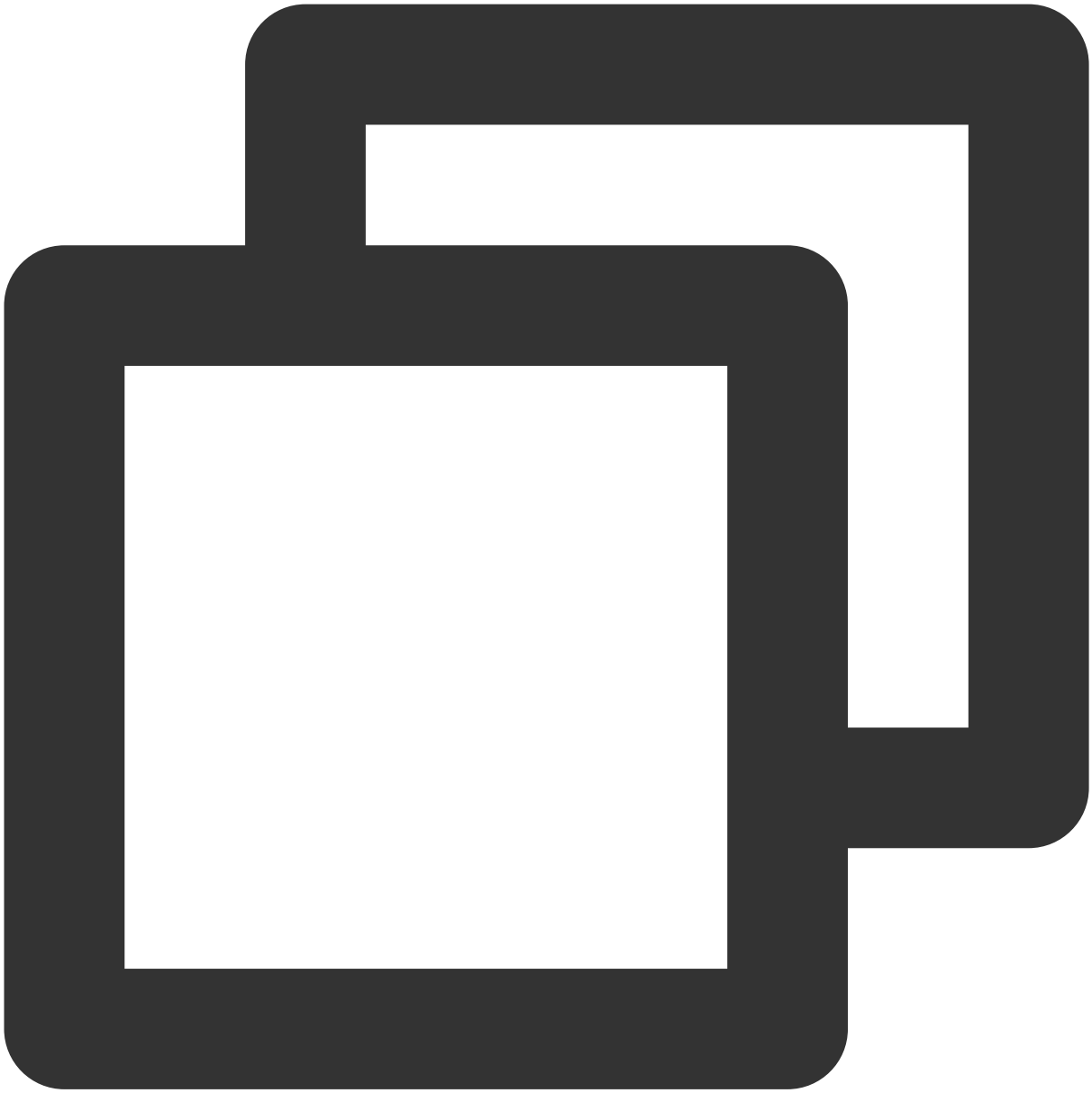
To ensure continuous CDN playback without stream disconnection, you need to keep the media stream encoding output parameter `trtcStreamEncoderParam` and the stream name `streamName` unchanged.

Media stream encoding output parameters and mixed display layout parameters can be customized according to business needs. Currently, up to 16 channels of audio and video input are supported. If a user only provides audio, it will still be counted as one channel.

Switching between audio only, audio and video, and video only is not supported within the same task.

4. The off-streaming audience exit the room, and the anchor updates the mixed stream task.

©2013-2022 Tencent Cloud. All rights reserved.



```
// The reusable TRTC video rendering control.
mLivePlayer.setRenderView(TXCloudVideoView view);
// Restart playing CDN media stream.
mLivePlayer.startLivePlay(URL);

// Player event callback.
private V2TXLivePlayerObserver mV2TXLivePlayerObserver = new V2TXLivePlayerObserver
    @Override
    public void onVideoLoading(V2TXLivePlayer player, Bundle extraInfo) {
        // Video loading event.
    }
```

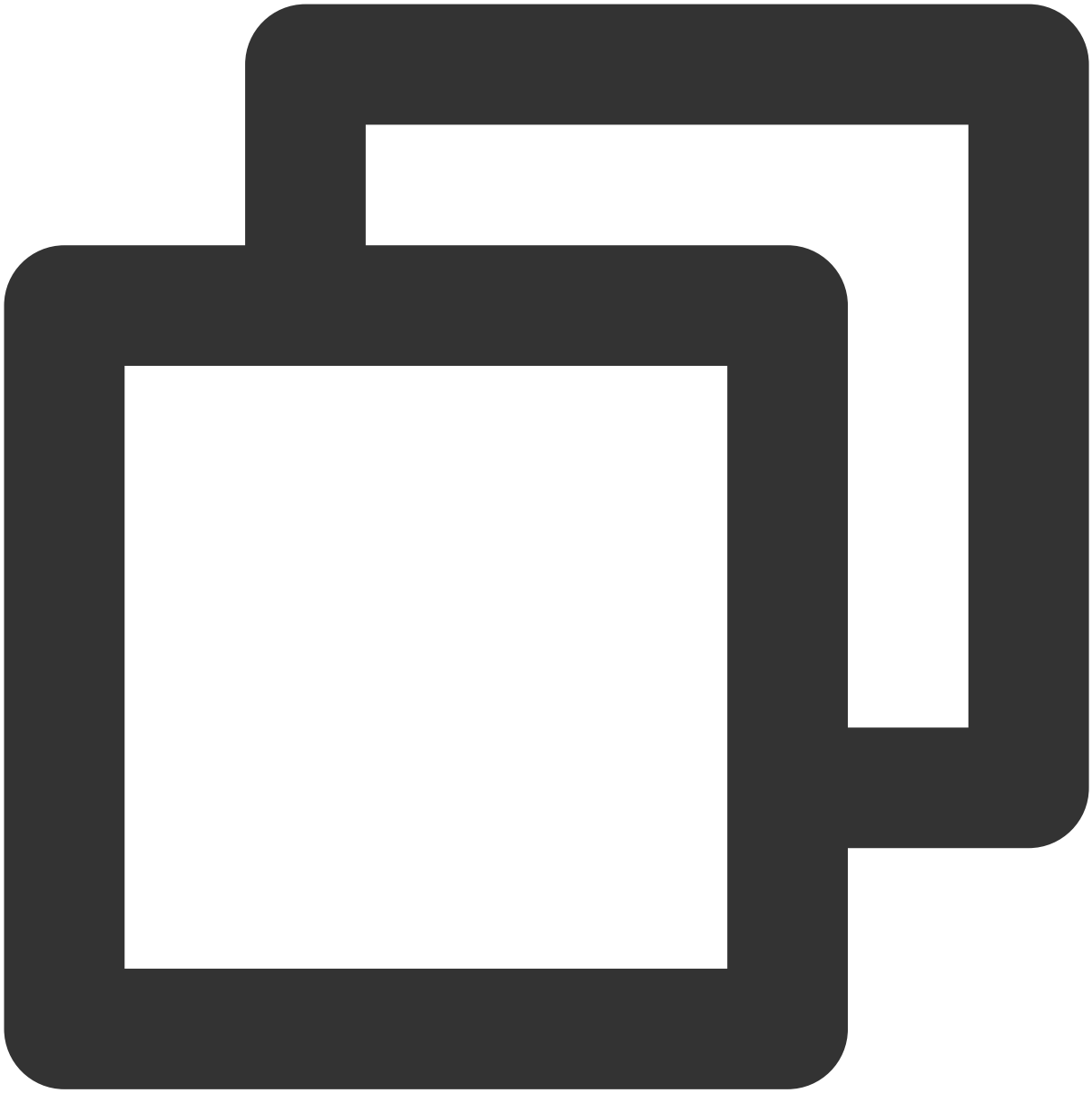
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onVideoPlaying(V2TXLivePlayer player, boolean firstPlay, Bundle ext
        // Video playback event.
        if (firstPlay) {
            mTRTCCloud.stopAllRemoteView();
            mTRTCCloud.stopLocalAudio();
            mTRTCCloud.stopLocalPreview();
            mTRTCCloud.exitRoom();
        }
    }
};
```

Note:

To avoid video interruptions when switching the stream puller, it is recommended to wait for the player's video playback event `onVideoPlaying` before exiting the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.

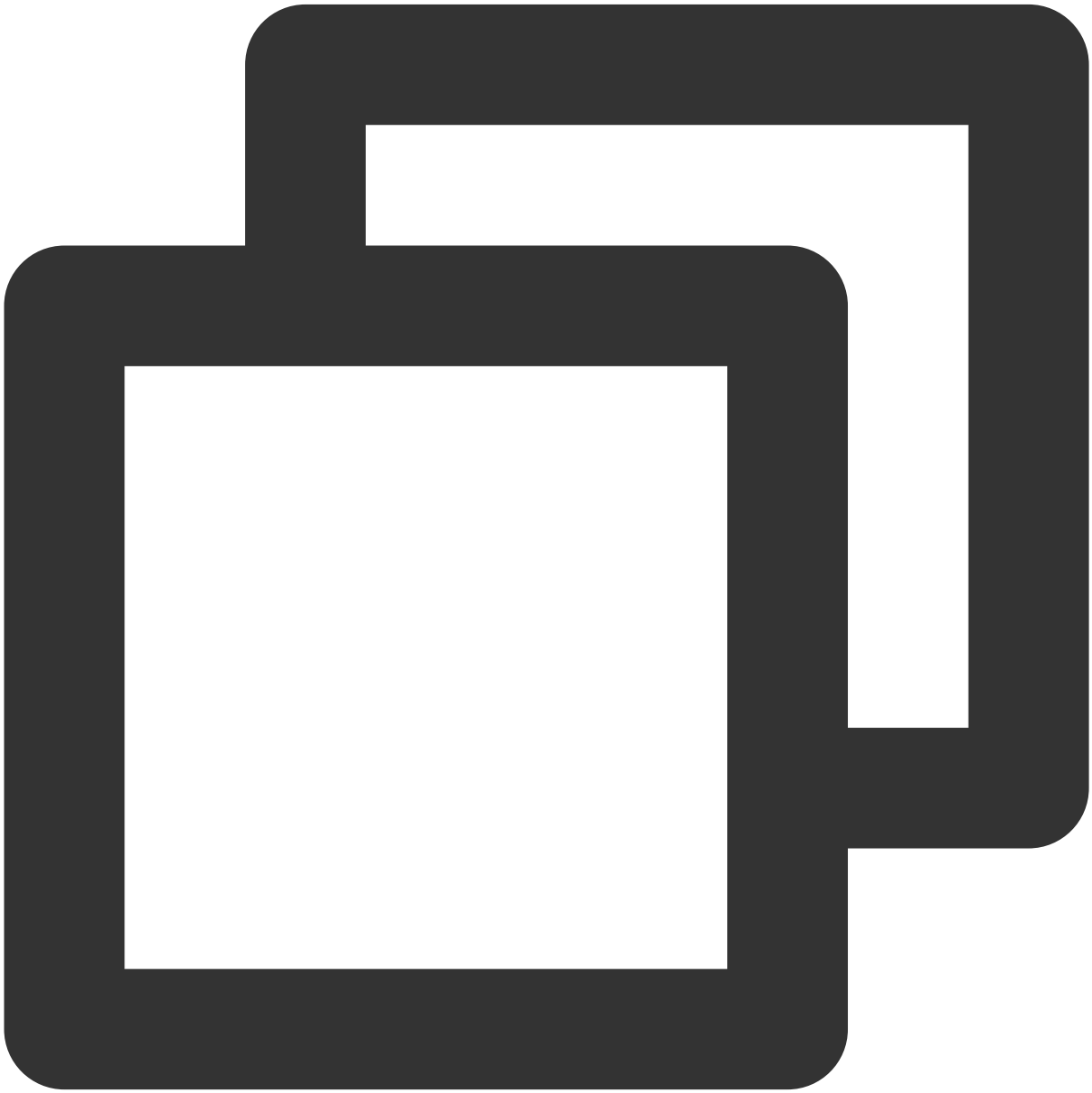


```
// Event callback for the mic-connection audience's room exit.
@Override
public void onRemoteUserLeaveRoom(String userId, int reason) {
    if (mixUserList.contains(userId)) {
        mixUserList.remove(userId);
    }
    // The anchor updates the mixed stream task.
    updatePublishMediaToCDN(streamName, mixUserList, taskId);
}

// Event callback for updating the media stream.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
@Override
public void onUpdatePublishMediaStream(String taskId, int code, String message, Bun
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure.
}
```

Step 4: The anchor stops the live streaming and exits the room.



```
public void exitRoom() {
    // Stop all published media streams.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    mTRTCCloud.stopPublishMediaStream("");
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.stopLocalPreview();
    mTRTCCloud.exitRoom();
}

// Event callback for stopping media streams.
@Override
public void onStopPublishMediaStream(String taskId, int code, String message, Bundl
    // When you call the stop publishing media stream API (stopPublishMediaStream),
    // code: Callback result. 0 means success and other values mean failure.
}

// Event callback for exiting the room.
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room.");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server.");
    } else if (reason == 2) {
        Log.d(TAG, "The current room has been dissolved.");
    }
}
```

Note:

To stop publishing media streams, fill in an empty string for `taskId`. This will stop all the media streams you have published.

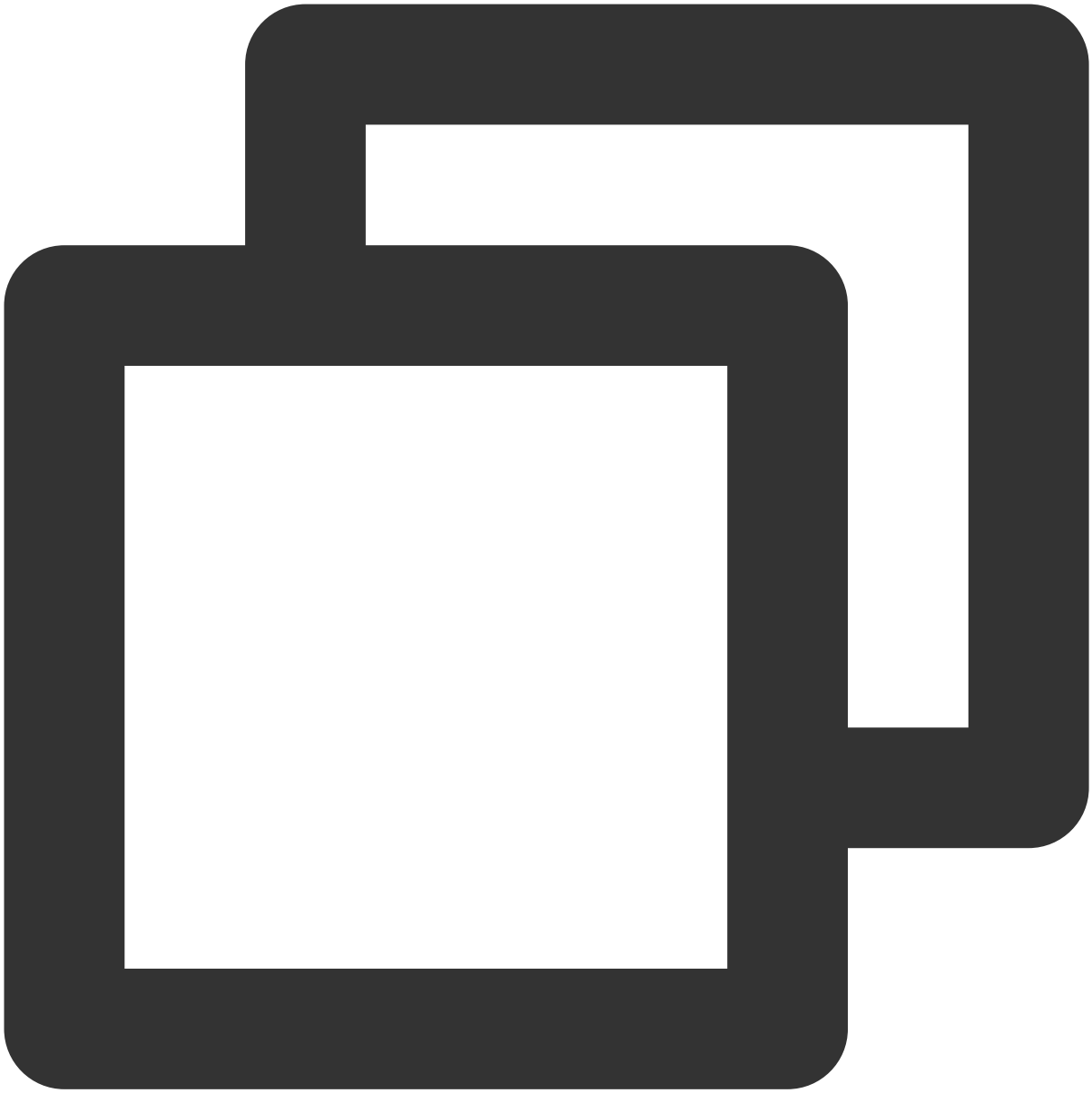
After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

Advanced Features

The anchor initiates the cross-room competition.

1. Either party initiates the cross-room competition.

©2013-2022 Tencent Cloud. All rights reserved.



```
public void connectOtherRoom(String roomId, String userId) {
    try {
        JSONObject jsonObj = new JSONObject();
        ? ? // The digit room ID is roomId.
        jsonObj.put("strRoomId", roomId);
        jsonObj.put("userId", userId);
        mTRTCCloud.ConnectOtherRoom(jsonObj.toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

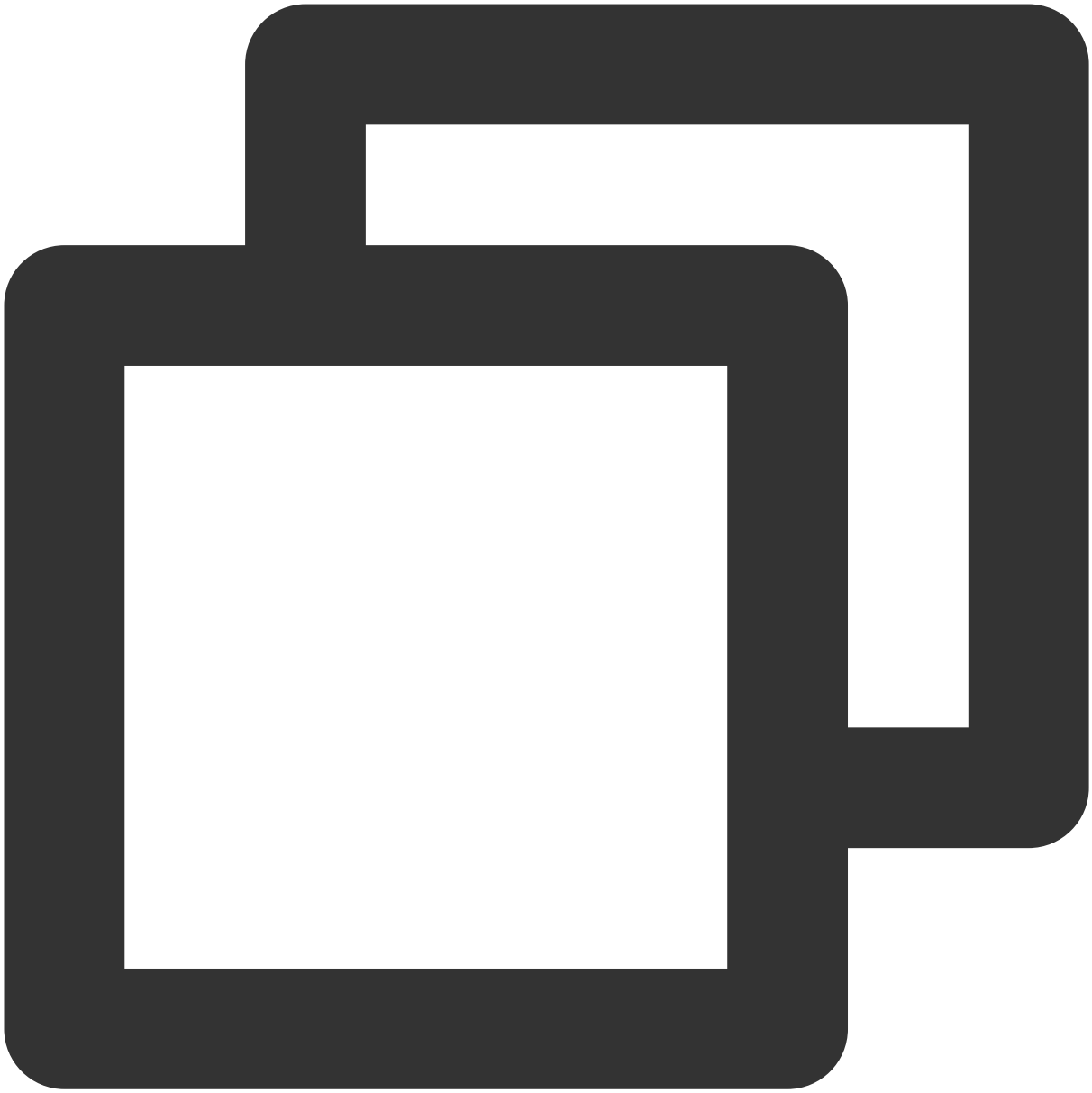
```
// Result callback for requesting cross-room mic-connection.
@Override
public void onConnectOtherRoom(String userId, int errCode, String errMsg) {
    // The user ID of the anchor in the other room you want to initiate the cross-r
    // Error code. ERR_NULL indicates the request is successful.
    // Error message.
}
```

Note:

Both local and remote users participating in the cross-room competition must be in the anchor role and must have audio or video uplink capabilities.

2. All users in both rooms will receive a callback indicating that the audio and video streams from the PK anchor in the other room are available.

©2013-2022 Tencent Cloud. All rights reserved.

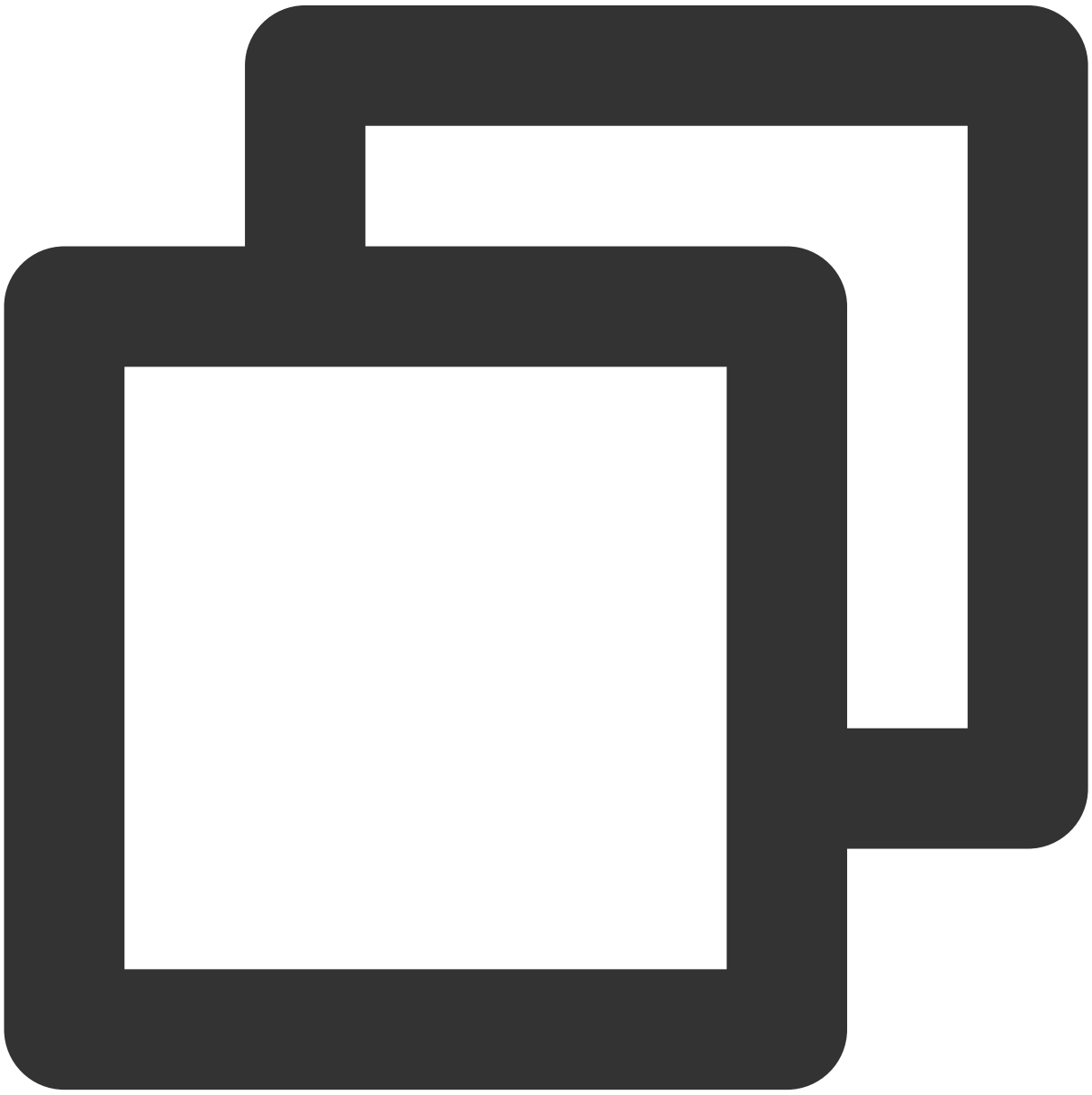


```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }
```

3. Either party exits the cross-room competition.



```
// Exit the cross-room mic-connection.
mTRTCCloud.DisconnectOtherRoom();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Result callback for exiting cross-room mic-connection.
@Override
public void onDisConnectOtherRoom(int errCode, String errMsg) {
    super.onDisConnectOtherRoom(errCode, errMsg);
}
```

Note:

After calling `DisconnectOtherRoom()` , you may exit the cross-room competition with all other room anchors. Either the initiator or the receiver can call `DisconnectOtherRoom()` to exit the cross-room competition.

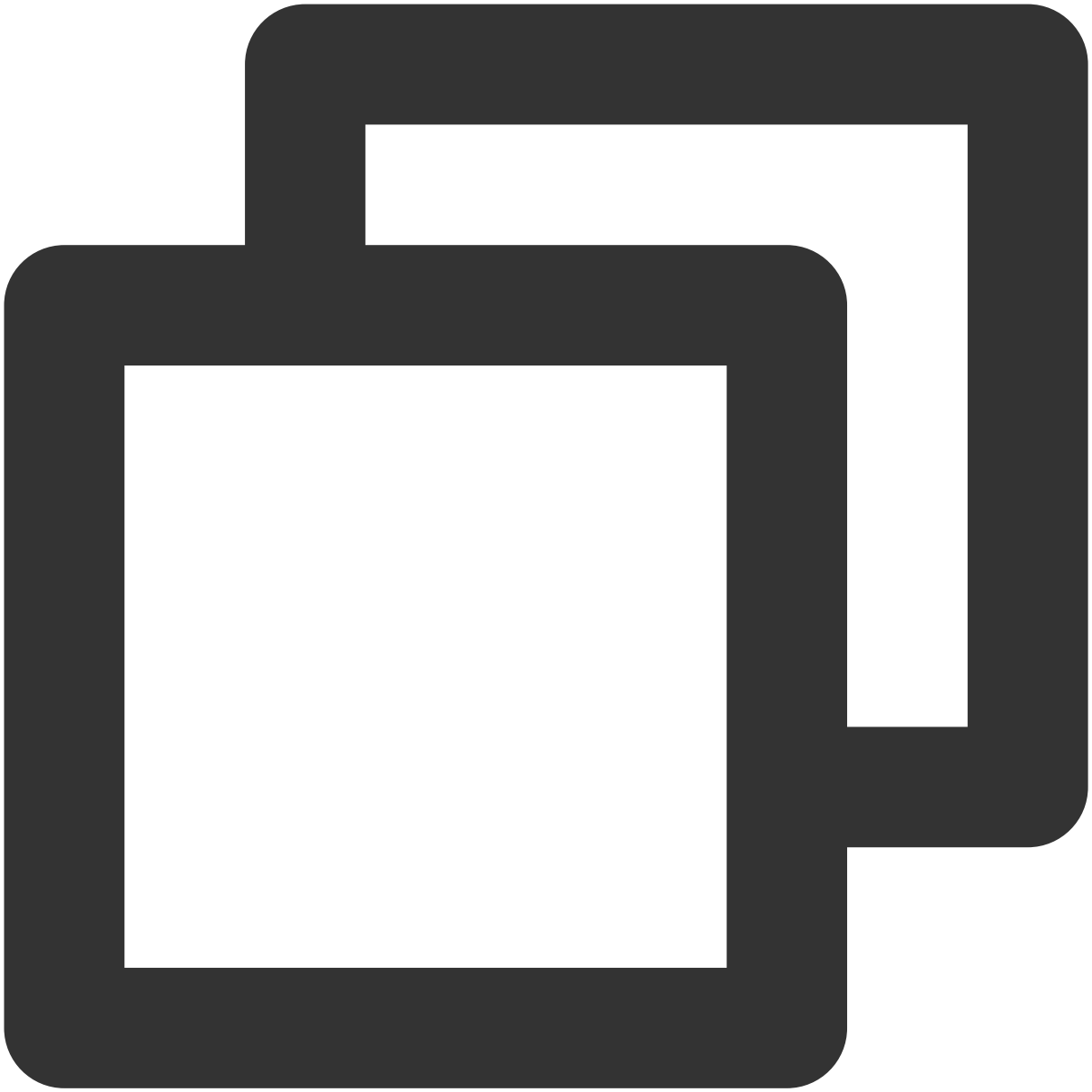
Integrate the third-party beauty features.

TRTC supports integrating third-party beauty effect products. Use the example of Tencent Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate the Tencent Effect SDK, and apply for an authorization license. For details, see Integration Preparation for steps.

2. Resource copying (if any). If your resource files are built in the assets directory, you need to copy them to the App's private directory before use.

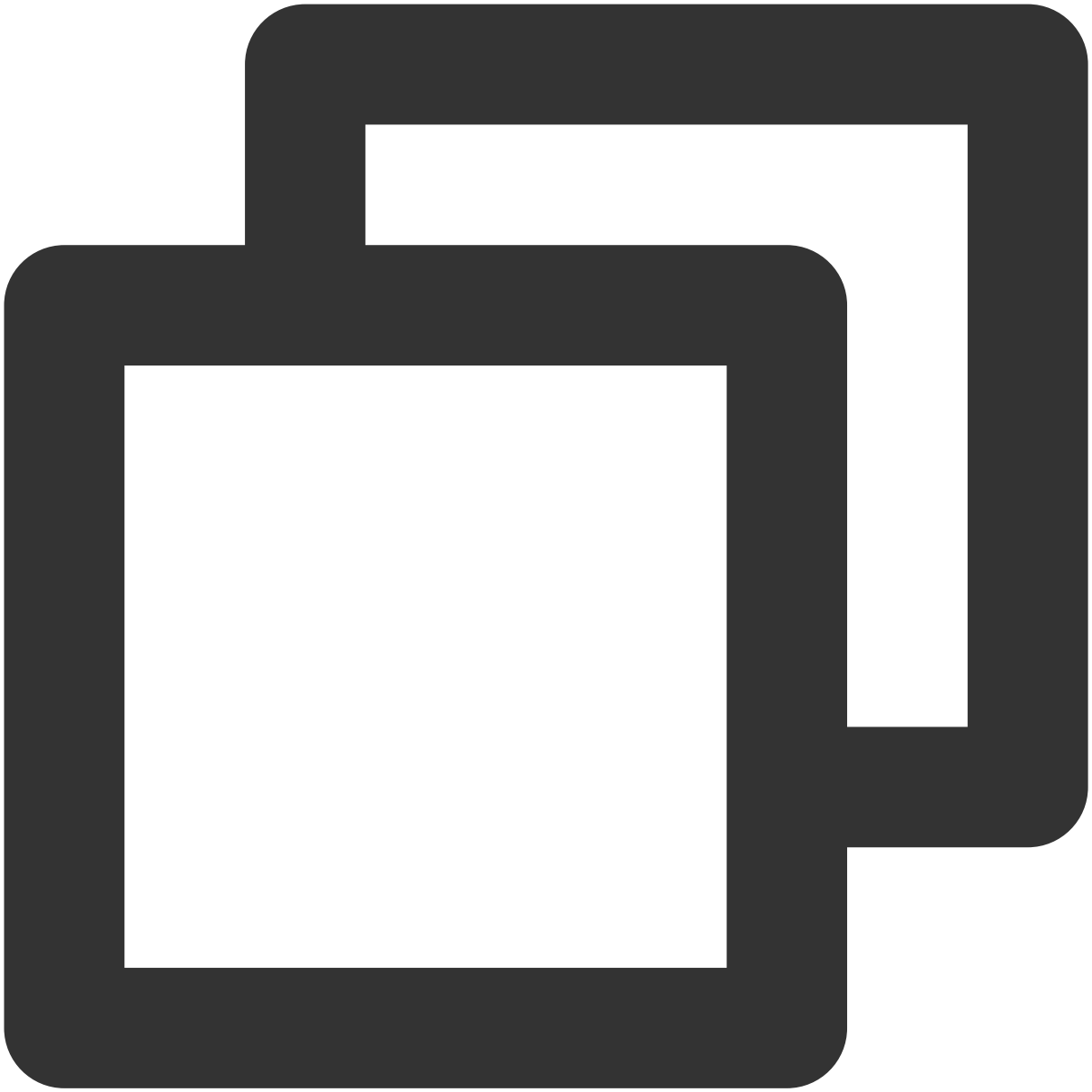
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").getAbsolutePath());
//loading

// Copy resource files to the private directory. Only need to do it once.
XmagicResParser.copyRes(getApplicationContext());
```

If your resource file is dynamically downloaded from the internet, you need to set the resource file path after the download is successful.

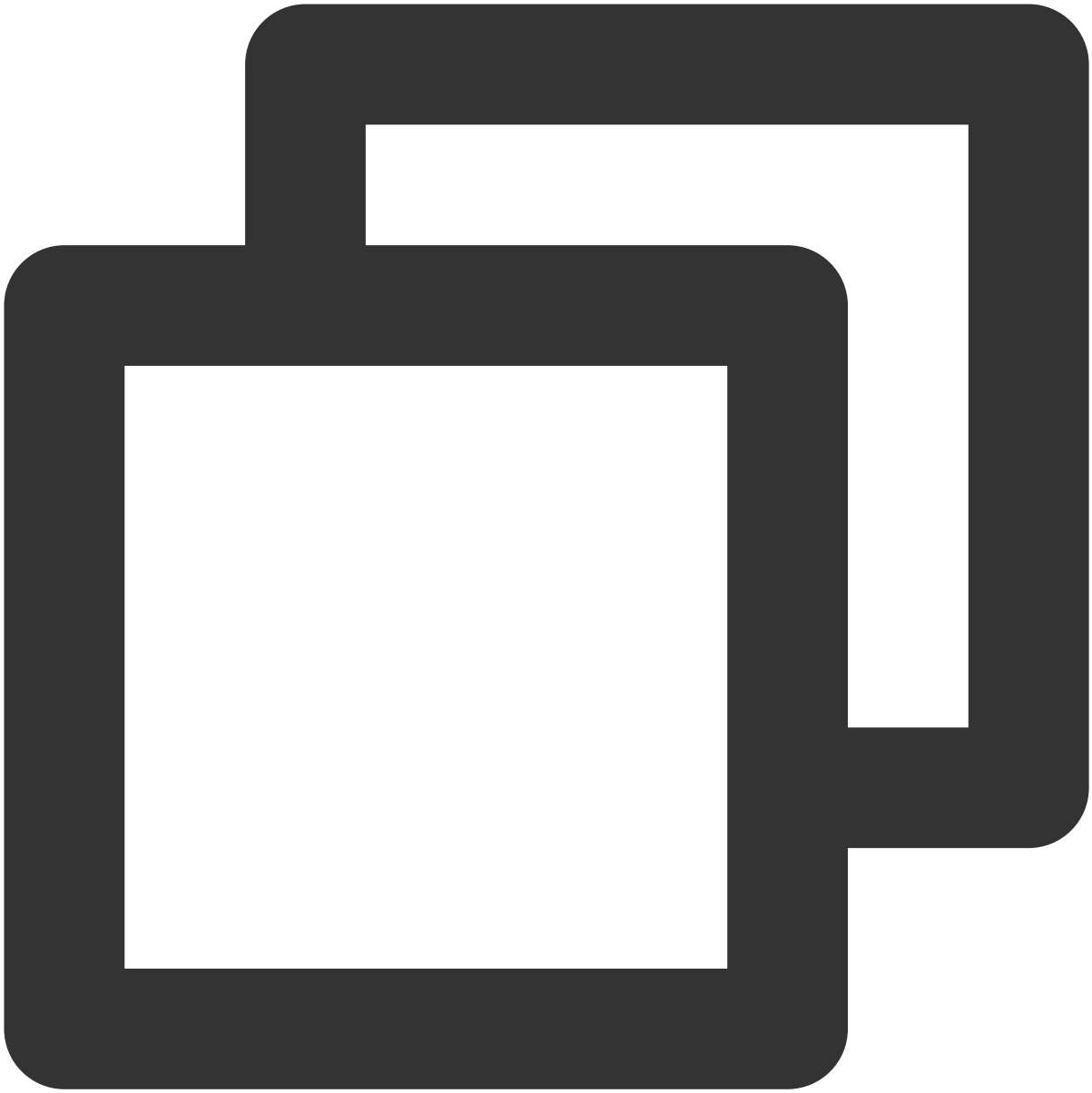
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath(local path of the downloaded resource file);
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
mTRTCCloud.setLocalVideoProcessListener(TRTCCloudDef.TRTC_VIDEO_PIXEL_FORMAT_Textur
    @Override
    public void onGLContextCreated() {
        // The OpenGL environment has already been set up internally within the SDK
        if (mXmagicApi == null) {
            XmagicApi mXmagicApi = new XmagicApi(context, XmagicResParser.getResPat
        } else {
            mXmagicApi.onResume();
        }
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudD
        // Callback for integrating with third-party beauty components for video pr
        if (mXmagicApi != null) {
            dstFrame.texture.textureId = mXmagicApi.process(srcFrame.texture.textur
        }
        return 0;
    }

    @Override
    public void onGLContextDestory() {
        // The internal OpenGL environment within the SDK has been terminated. At t
        mXmagicApi.onDestroy();
    }
});
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products. And **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

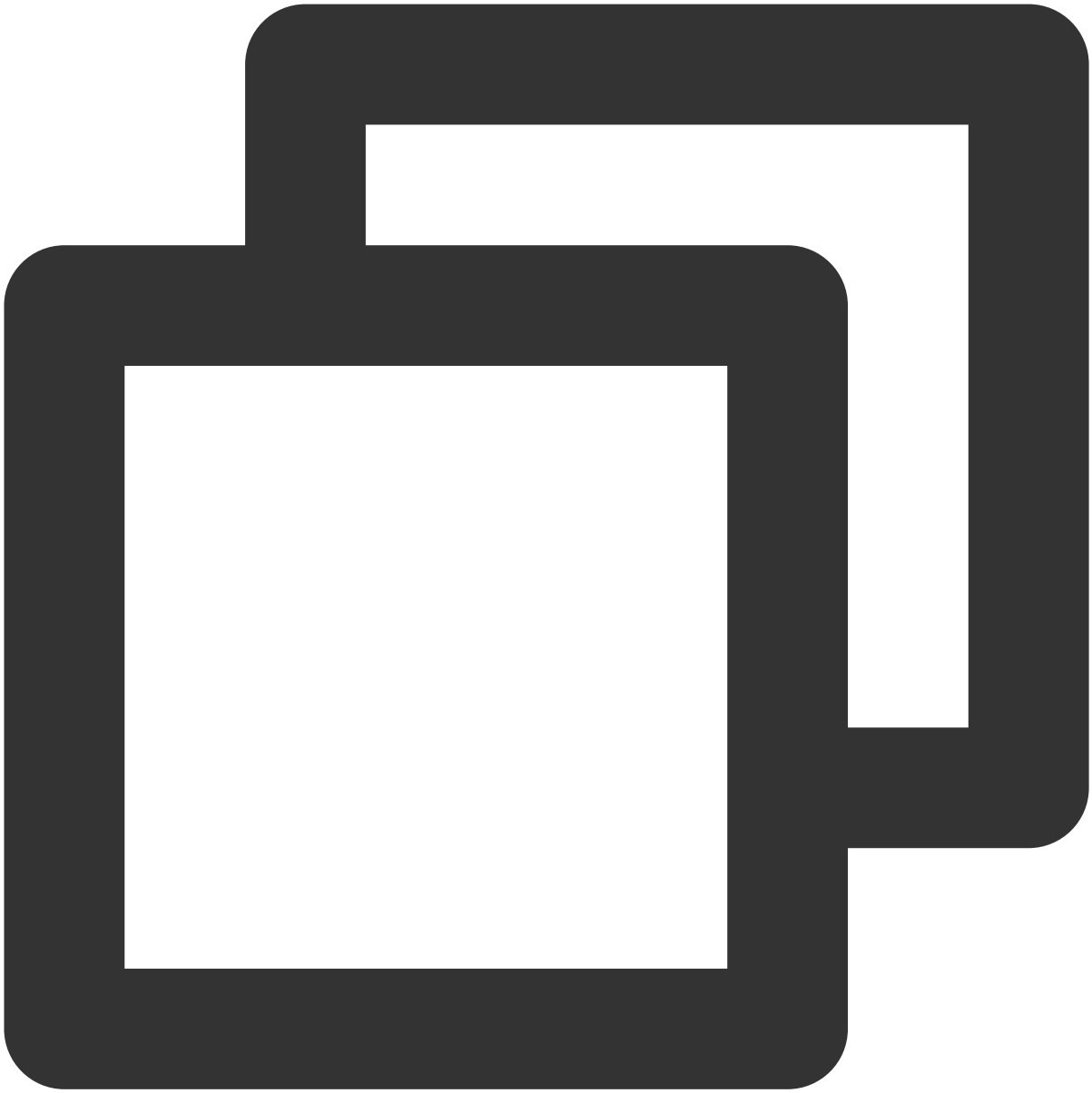
For scenario-specific integration guidelines of Tencent Effect, see Integrating Tencent Effect into TRTC SDK. For guidelines on integrating Tencent Effect independently, see Integrating Tencent Effect SDK.

Dual-stream encoding mode

When the dual-stream encoding mode is enabled, the current user's encoder will output two video streams, a high-definition large screen, and a low-definition small screen, at the same time (but only one audio stream). In this way, other users in the room can choose to subscribe to the high-definition large screen or low-definition small screen based on their network conditions or screen sizes.

1. Enable large-and-small-screen dual-stream encoding mode.

©2013-2022 Tencent Cloud. All rights reserved.

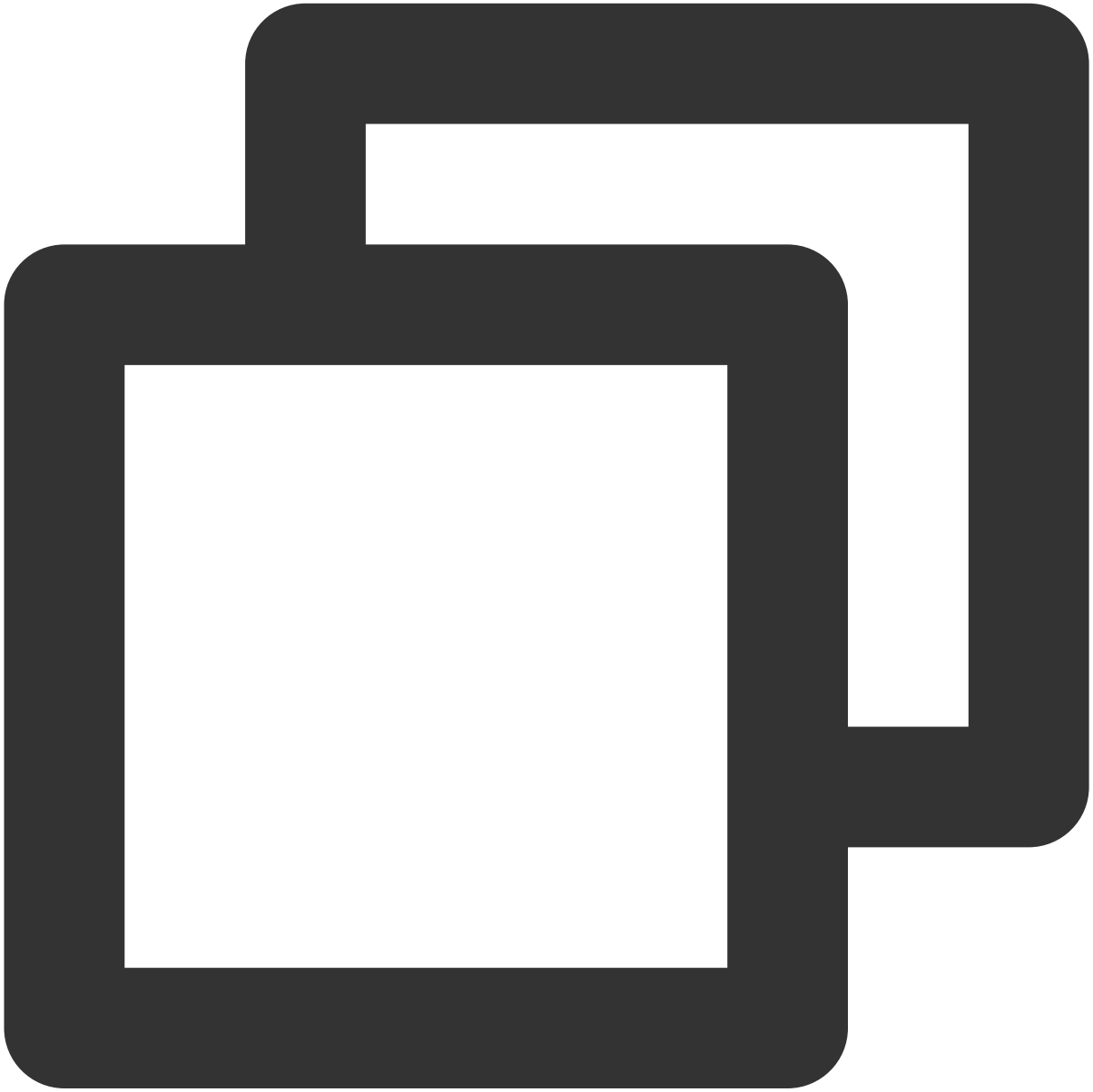


```
public void enableDualStreamMode(boolean enable) {
    // Video encoding parameters for the small-screen stream (customizable).
    TRTCCloudDef.TRTCVideoEncParam smallVideoEncParam = new TRTCCloudDef.TRTCVideoE
    smallVideoEncParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270
    smallVideoEncParam.videoFps = 15;
    smallVideoEncParam.videoBitrate = 550;
    smallVideoEncParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MOD
    mTRTCCloud.enableEncSmallVideoStream(enable, smallVideoEncParam);
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, it will consume more CPU and network bandwidth. Therefore, it may be considered for use on Mac, Windows, or high-performance Pads. It is not recommended for mobile devices.
2. Choose the type of remote user's video stream to pull.



```
// Optional video stream types when subscribing to a remote user's video stream.
mTRTCCloud.startRemoteView(userId, streamType, videoView);

// You can switch the size of the specified remote user's screen at any time.
mTRTCCloud.setRemoteVideoStreamType(userId, streamType);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

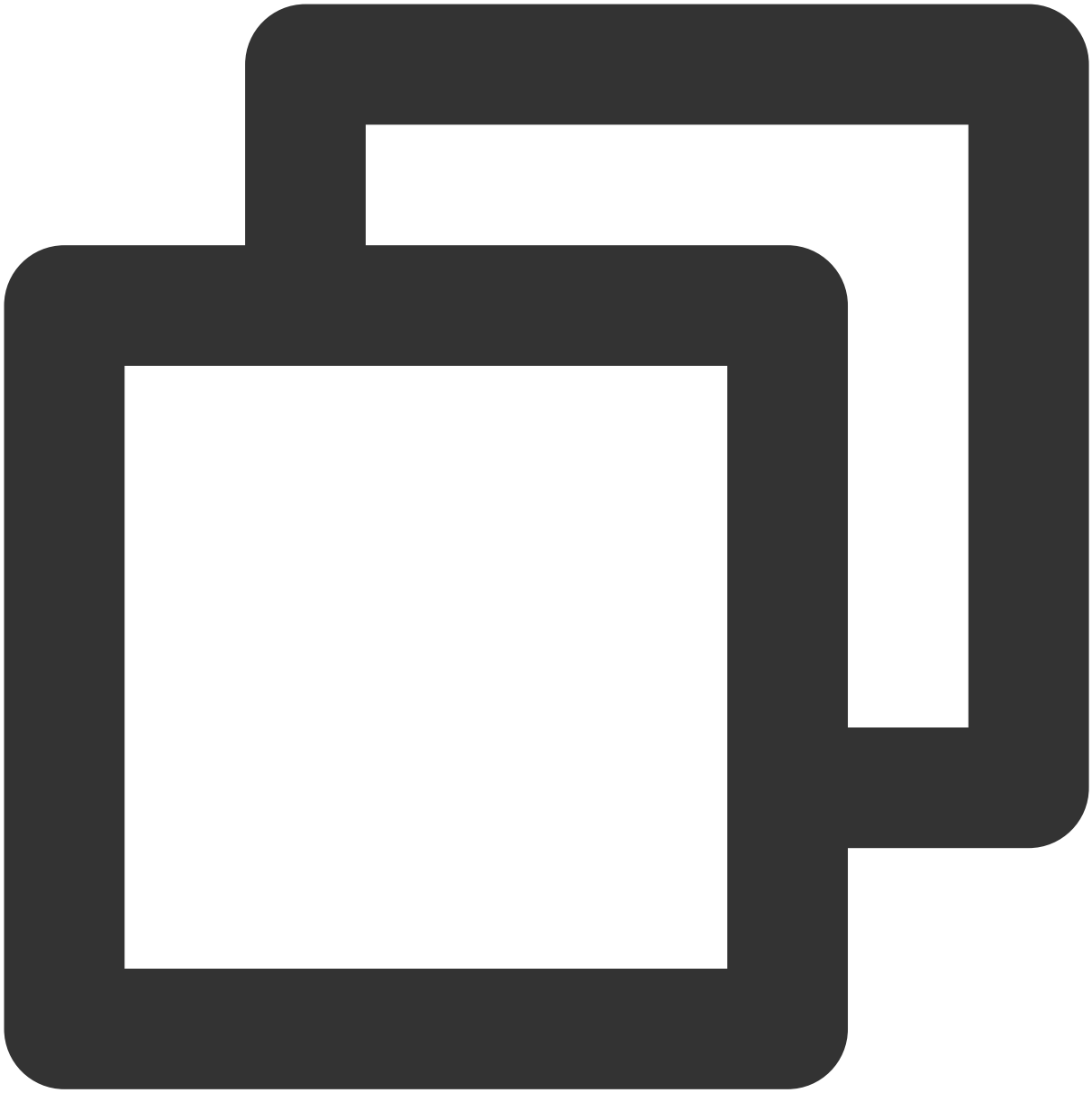
When the dual-stream encoding mode is enabled, you can specify the video stream type as `TRTC_VIDEO_STREAM_TYPE_SMALL` with `streamType` to pull a low-quality small video for viewing.

View rendering control

In TRTC, there are many APIs that require you to control the video screen. All these APIs require you to specify a video rendering control. On the Android platform, `TXCloudVideoView` is used as the video rendering control, and both `SurfaceView` and `TextureView` rendering schemes are supported. Below are the methods for specifying the type of rendering control and updating the video rendering control.

1. If you want mandatory use of a certain scheme, or to convert the local video rendering control to `TXCloudVideoView`, you can code as follows.

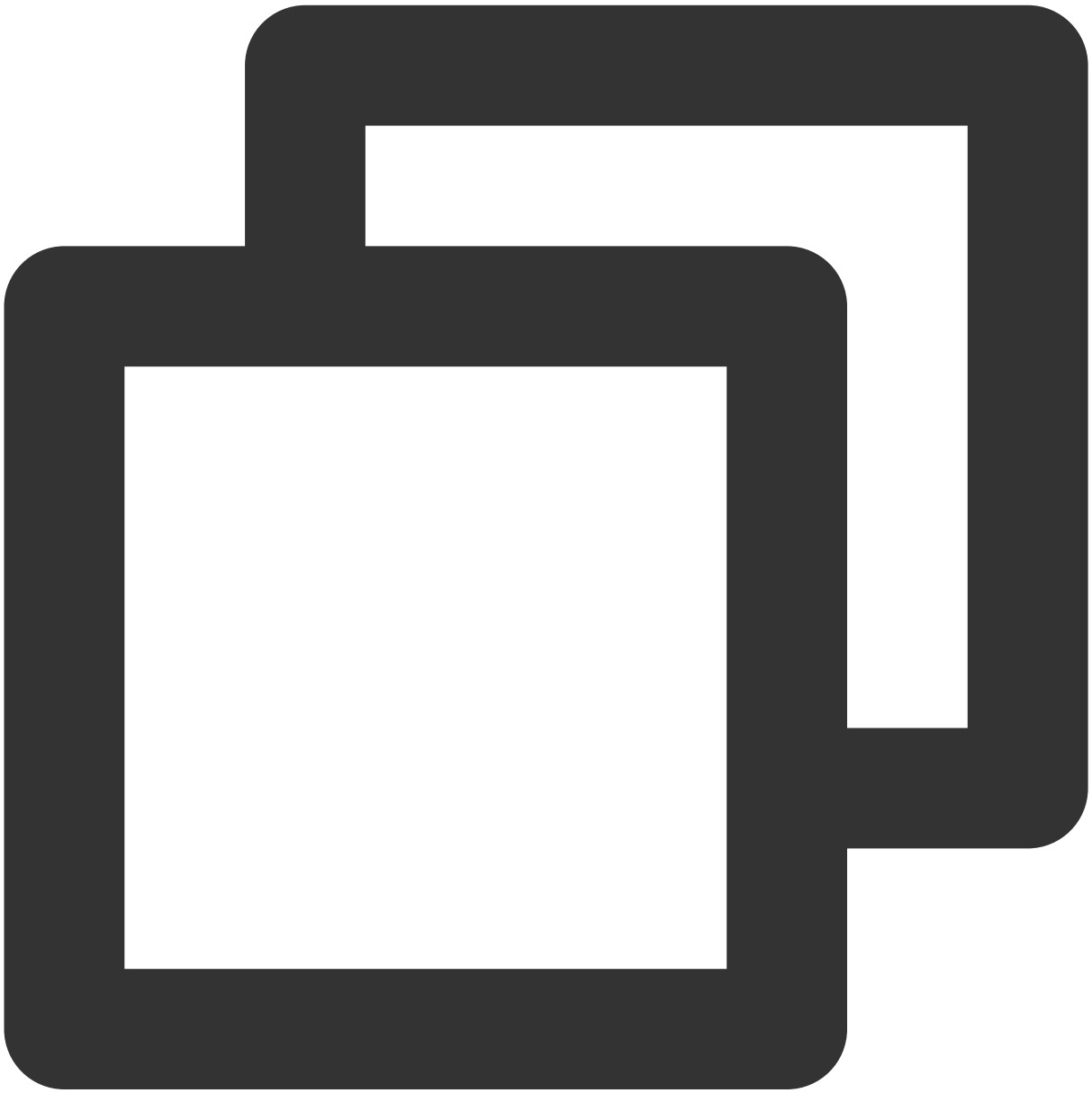
©2013-2022 Tencent Cloud. All rights reserved.



```
// Mandatory use of TextureView.
TextureView textureView = findViewById(R.id.texture_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(context);
cloudVideoView.addVideoView(textureView);

// Mandatory use of SurfaceView.
SurfaceView surfaceView = findViewById(R.id.surface_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(surfaceView);
```

©2013-2022 Tencent Cloud. All rights reserved.

2. If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.



```
// Update local preview screen rendering control.
mTRTCCloud.updateLocalView(videoView);

// Update the remote user's video rendering control.
mTRTCCloud.updateRemoteView(userId, streamType, videoView);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Pass in `videoView` as the target video rendering control. `streamType` only supports `TRTC_VIDEO_STREAM_TYPE_BIG` and `TRTC_VIDEO_STREAM_TYPE_SUB`.

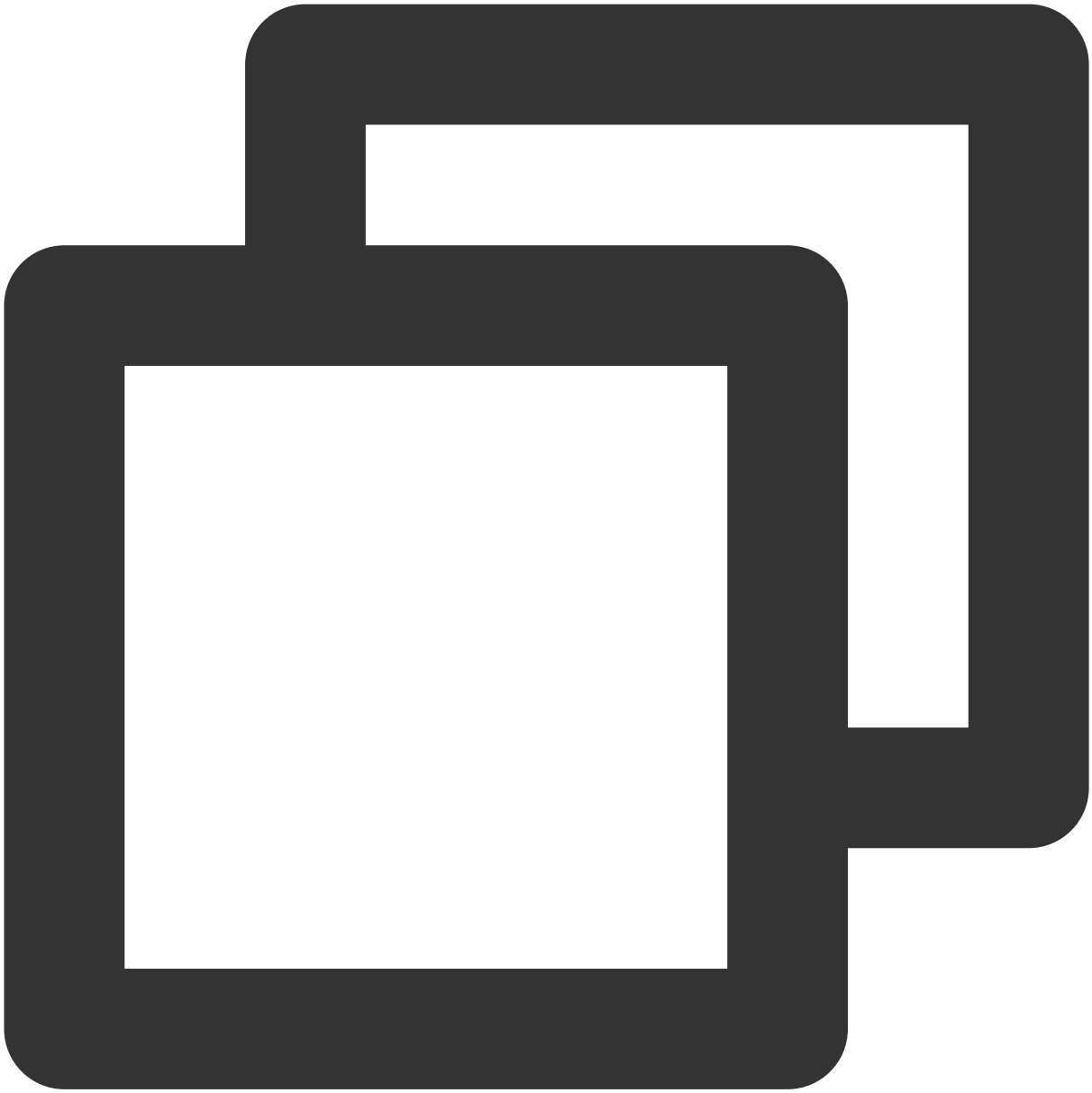
Live Streaming Interactive Messages

Live streaming interaction is particularly important in live streaming scenarios. Users interact with the anchor through likes messages, gift messages, and bullet screen messages. The precondition for implementing the live interaction feature is to activate the Instant Messaging (IM) service and import the IM SDK. For detailed guidelines, see Voice Chat Room Integration Guide - Preparation for Integration.

Likes messages

1. The liker sends custom group messages related to likes through the client. After it is sended successfully, the business party renders the likes effect locally.

©2013-2022 Tencent Cloud. All rights reserved.



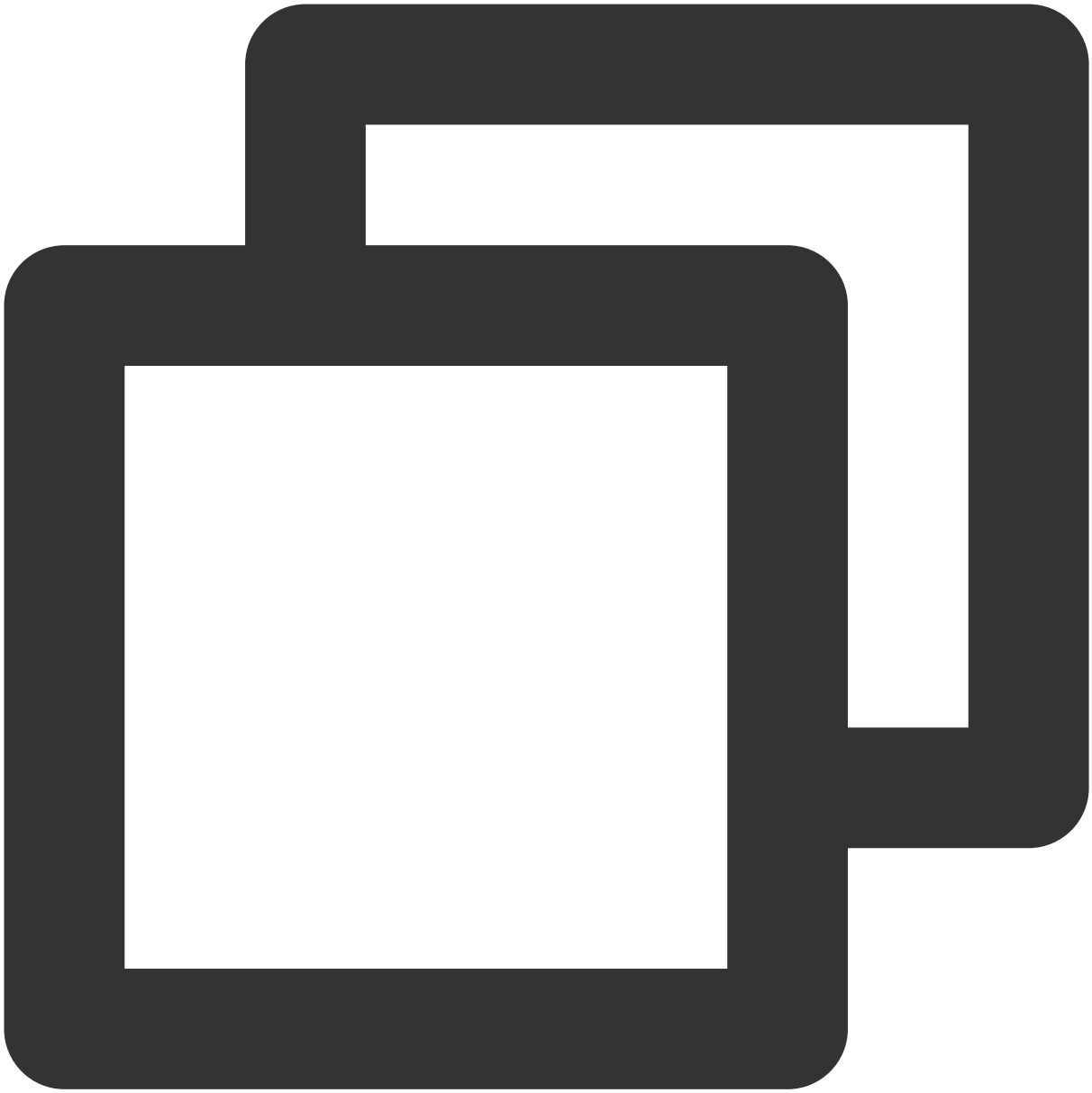
```
// Construct the likes message body.
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("cmd", "like_msg");
    JSONObject msgJsonObject = new JSONObject();
    msgJsonObject.put("type", 1);         // Likes type.
    msgJsonObject.put("likeCount", 10);  // Number of likes.
    jsonObject.put("msg", msgJsonObject);
} catch (JSONException e) {
    e.printStackTrace();
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
String data = jsonObject.toString();

// Send custom group messages (it is recommended that likes messages should be set
V2TIMManager.getInstance().sendGroupCustomMessage(data.getBytes(), mRoomId,
        V2TIMMessage.V2TIM_PRIORITY_LOW, new V2TIMValueCallback<V2TIMMessage>() {
            @Override
            public void onError(int i, String s) {
                // Failed to send likes messages.
            }

            @Override
            public void onSuccess(V2TIMMessage v2TIMMessage) {
                // Likes messages sent successfully.
                // Local rendering of likes effect.
            }
        });
```

2. Other users in the room receive callback for custom group messages. Then proceed with message parsing and likes effect rendering.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Custom group messages received.
V2TIMManager.getInstance().addSimpleMsgListener(new V2TIMSimpleMsgListener() {
    @Override
    public void onRecvGroupCustomMessage(String msgID, String groupID, V2TIMGroupMe
        String customStr = new String(customData);
        if (!customStr.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(customStr);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("like_msg")) {
```

©2013-2022 Tencent Cloud. All rights reserved.

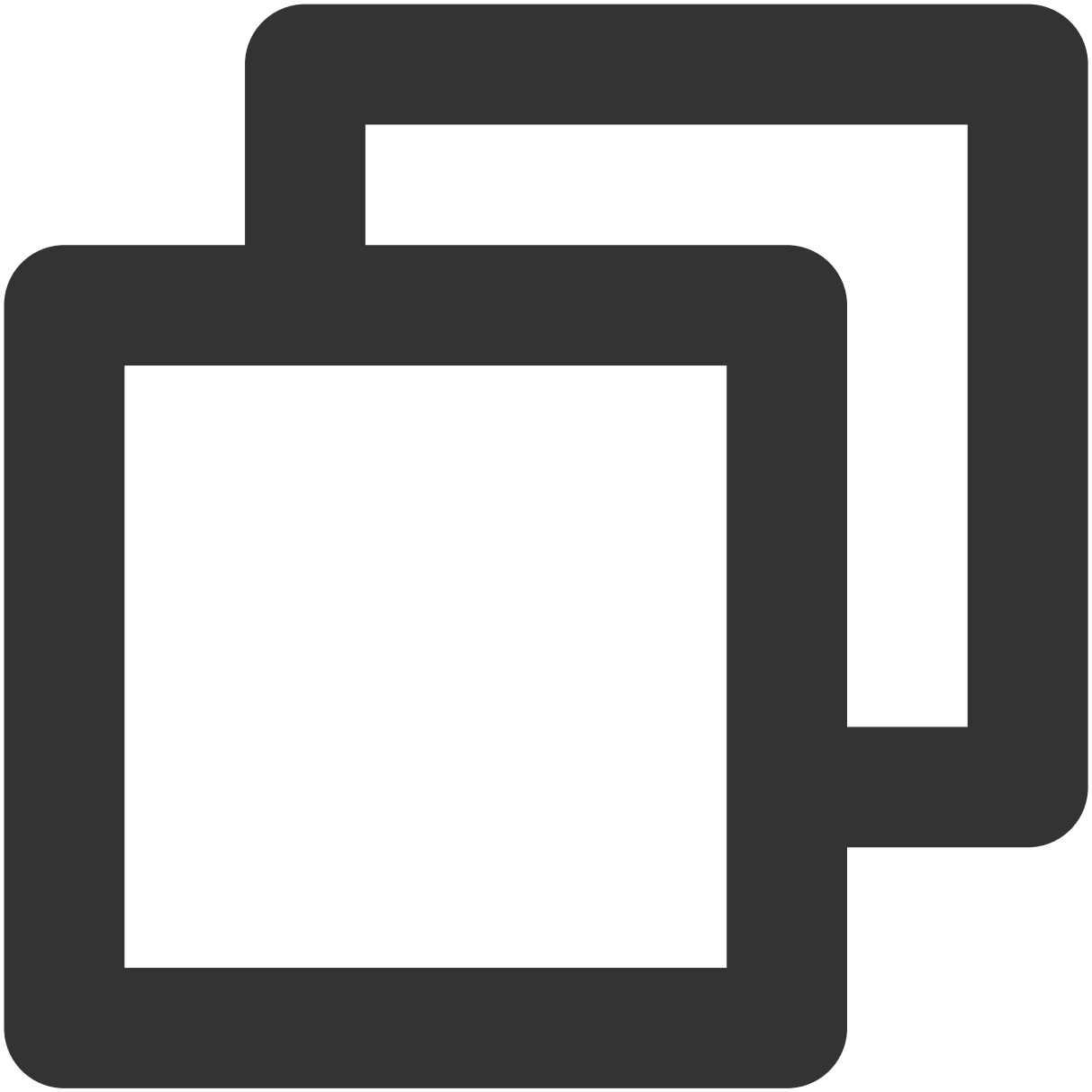
```
                    int type = messageJsonObject.getInt("type");            // Like
                    int likeCount = messageJsonObject.getInt("likeCount");  // Numb
                    // Render likes effect based on likes type and count.
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Gift messages

1. The sender initiates a request to the business server. Upon completing the billing and settlement, the business server calls the REST API to send a custom message to the group.

1.1 Request URL sample:

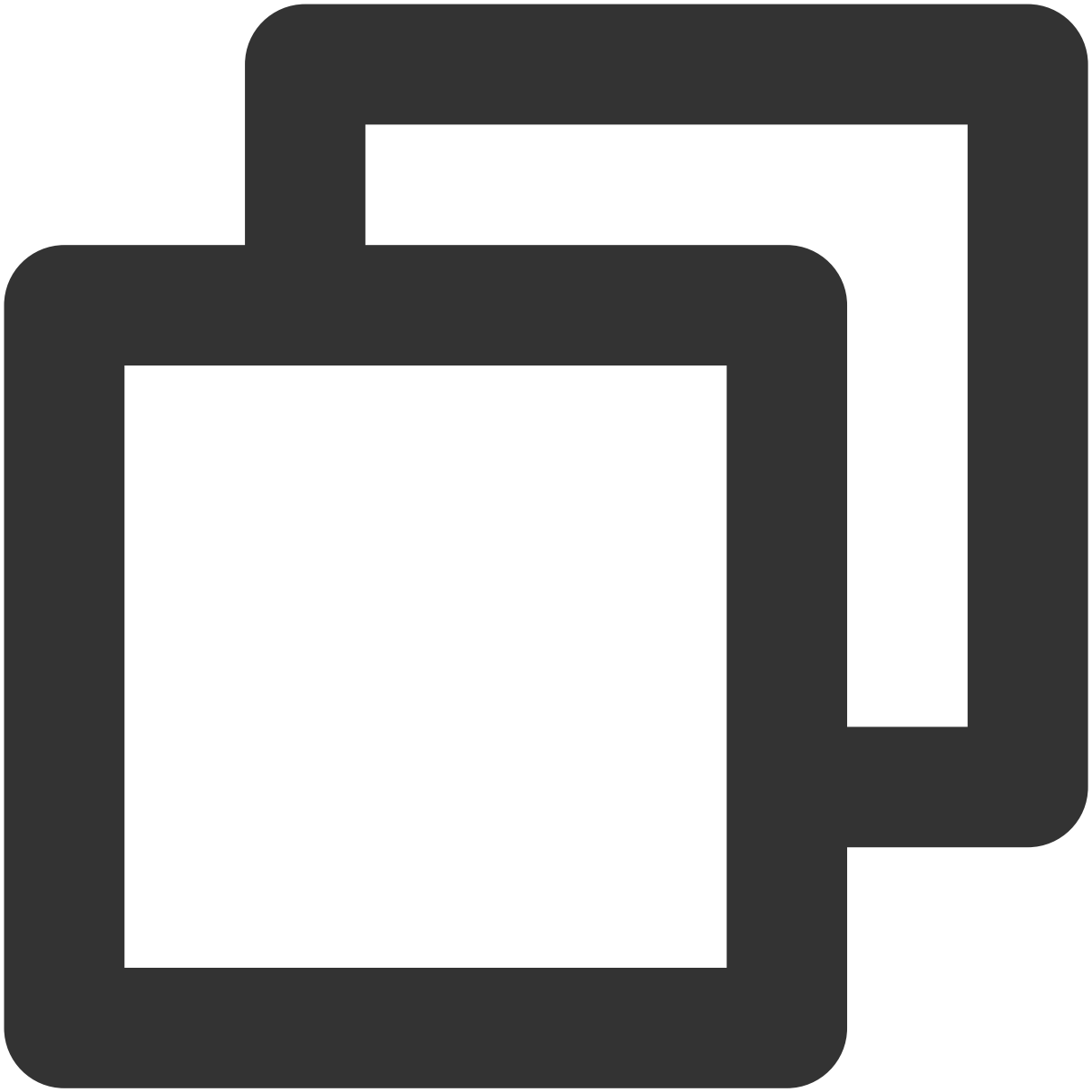
©2013-2022 Tencent Cloud. All rights reserved.



```
https://xxxxxx/v4/group_open_http_svc/send_group_msg?sdkappid=88888888&identifier=a
```

1.2 Request packet body sample:

©2013-2022 Tencent Cloud. All rights reserved.

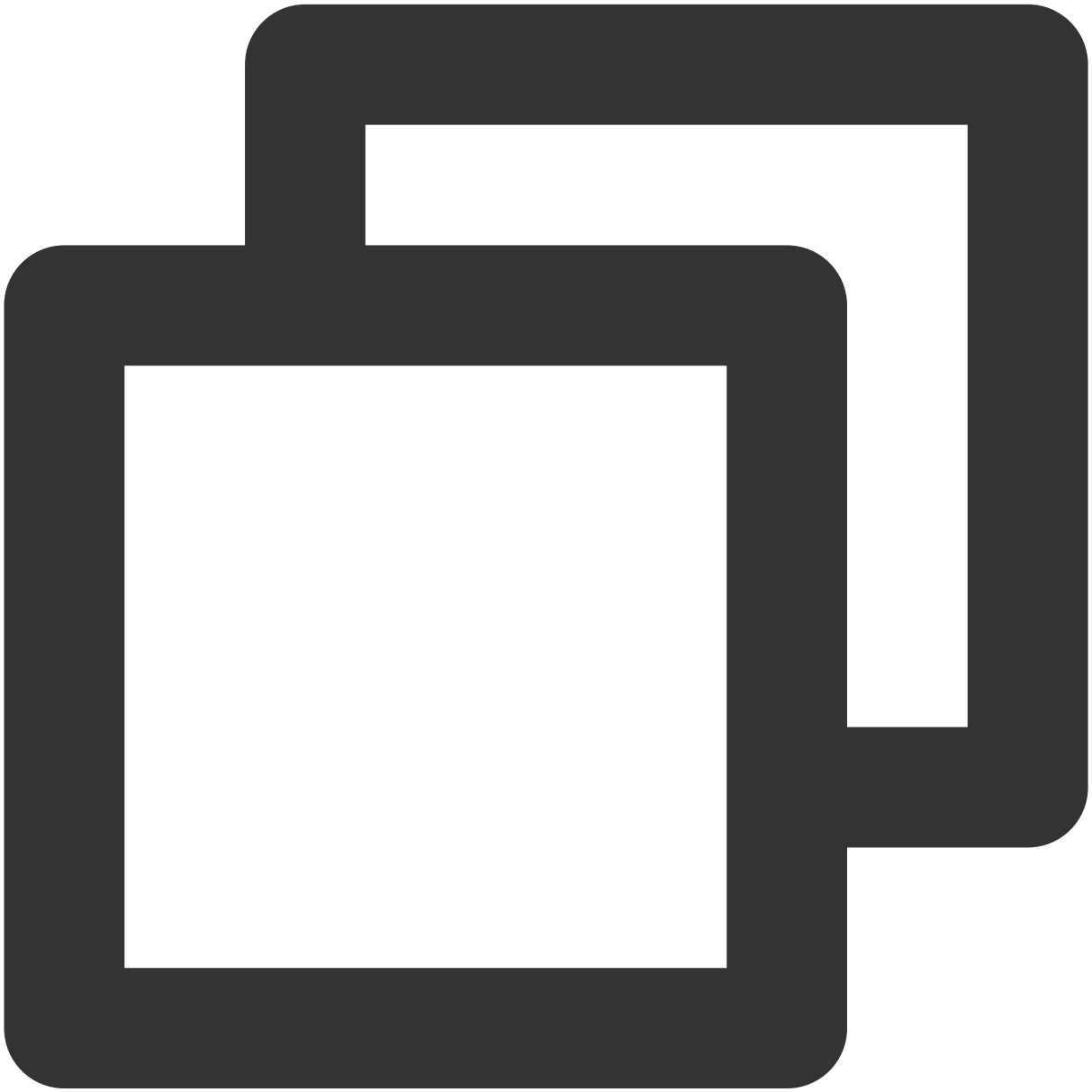


```
{
    "GroupId": "@TGS#12DEVUDHQ",
    "Random": 2784275388,
    "MsgPriority": "High",  // The priority of the message. Gift messages should be
    "MsgBody": [
        {
            "MsgType": "TIMCustomElem",
            "MsgContent": {
                // type: gift type; giftUrl: gift resource URL; giftName: gift name
                "Data": "{\\"cmd\\": \\"gift_msg\\", \\"msg\\": {\\"type\\": 1, \\"
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        ]
    }
```

2. Other users in the room receive a callback for custom group messages. Then proceed with message parsing and gift effect rendering.



```
// Custom group messages received.
V2TIMManager.getInstance().addSimpleMsgListener(new V2TIMSimpleMsgListener() {
    @Override
    public void onRecvGroupCustomMessage(String msgID, String groupID, V2TIMGroupMe
```

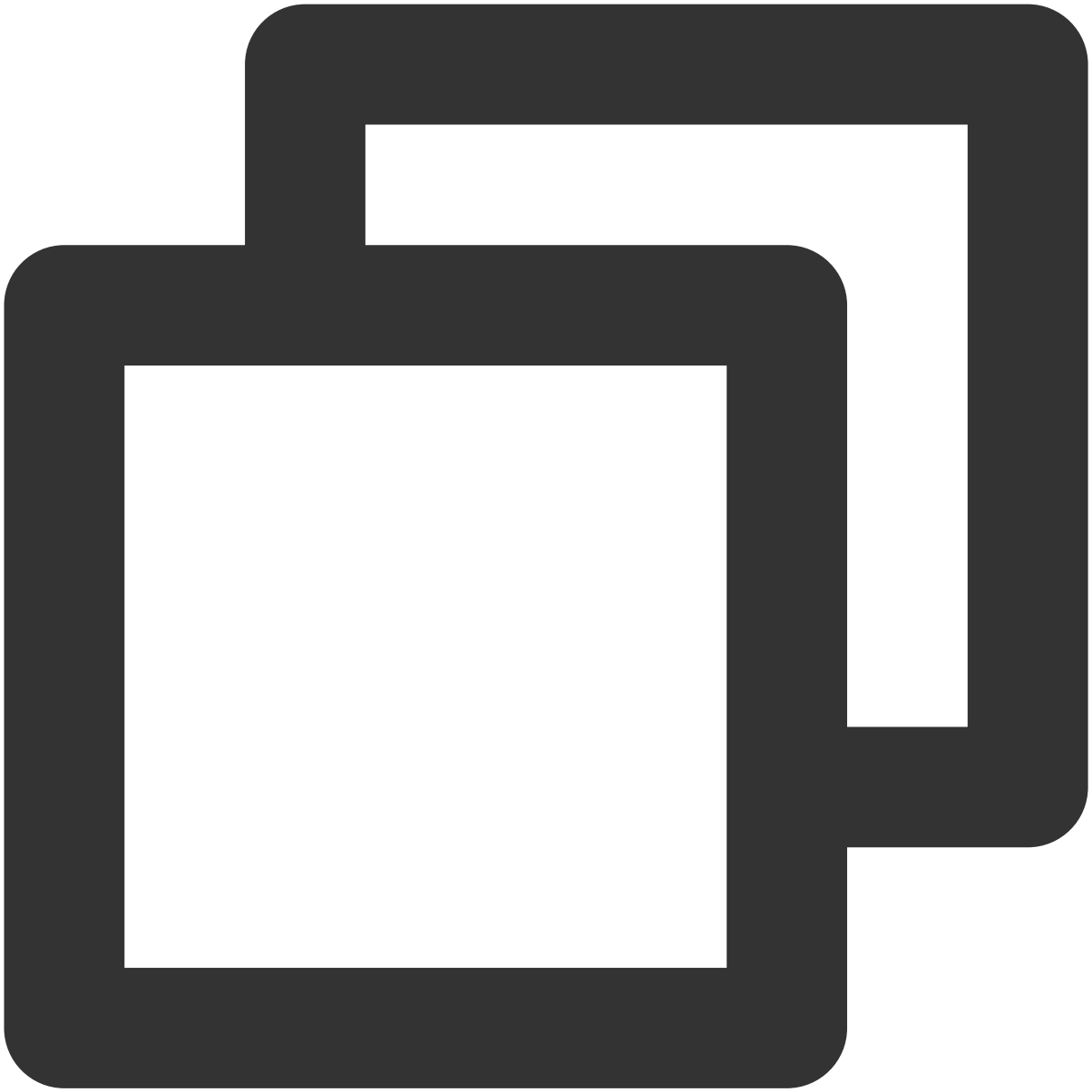
©2013-2022 Tencent Cloud. All rights reserved.

```
        String customStr = new String(customData);
        if (!customStr.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(customStr);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("gift_msg")) {
                    int type = messageJsonObject.getInt("type");                //
                    int giftCount = messageJsonObject.getInt("giftCount");      //
                    String giftUrl = messageJsonObject.getString("giftUrl");    //
                    String giftName = messageJsonObject.getString("giftName");  //
                    // Render gift effects based on gift type, count, resource URL,
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Bullet screen messages

Live showroom usually have text-based bullet screen message interactions. This can be achieved through the sending and receiving of group chat regular text messages via IM.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Send public screen bullet screen messages.
V2TIMManager.getInstance().sendGroupTextMessage(text, groupID, V2TIMMessage.V2TIM_P
    @Override
    public void onError(int i, String s) {
        // Failed to send bullet screen messages.
    }

    @Override
    public void onSuccess(V2TIMMessage v2TIMMessage) {
        // Successfully sent bullet screen messages.
        // Local display of the message text.
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
});

// Receive public screen bullet screen messages.
V2TIMManager.getInstance().addSimpleMsgListener(new V2TIMSimpleMsgListener() {
    @Override
    public void onRecvGroupTextMessage(String msgID, String groupID, V2TIMGroupMemb
        // Render bullet screen messages based on sender and message text.
    }
});
```

Note:

The recommended priority setting is as follows. Gift messages should be set to high priority. Bullet screen messages should be set to medium priority. Like messages should be set to low priority.

Sending group chat messages from the client will not trigger the message reception callback. Only other users within the group can receive them.

Exception Handling

Exception error handling

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your

©2013-2022 Tencent Cloud. All rights reserved.

		internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

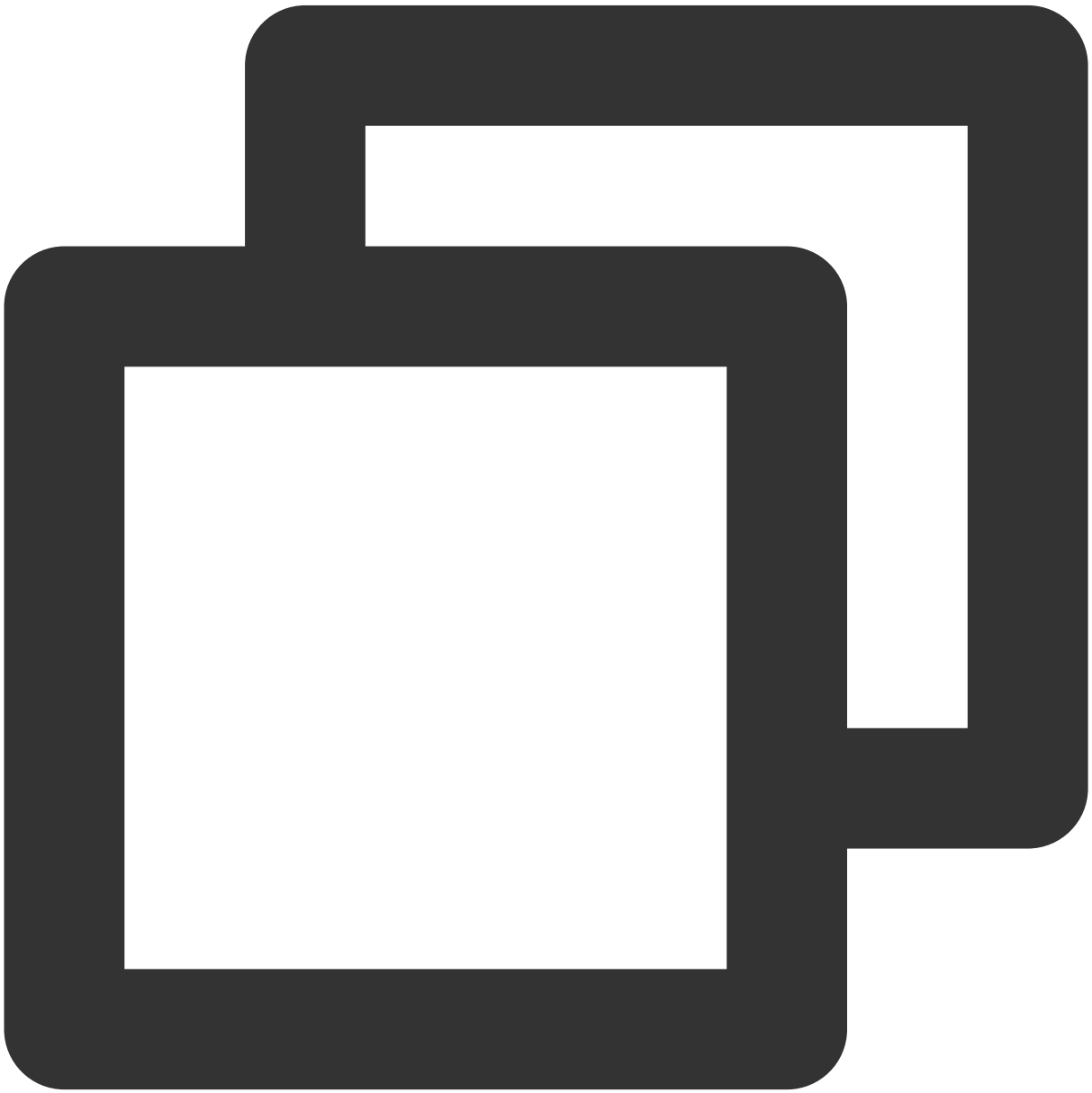
Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to open the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the

©2013-2022 Tencent Cloud. All rights reserved.

		user is currently having a call on the mobile device.

Issues with the remote mirror mode not functioning properly.

In TRTC, video mirror settings are divided into local preview mirror `setLocalRenderParams` and video encoding mirror `setVideoEncoderMirror` . These settings individually affect the mirror effect of the local preview and the output of the video encoding (the mirror mode affects remote viewers and cloud recordings). If you expect the mirror effect seen in the local preview to also take effect on the remote viewer's end, follow these encoding procedures.



```
// Set the rendering parameters for the local video.
```

©2013-2022 Tencent Cloud. All rights reserved.

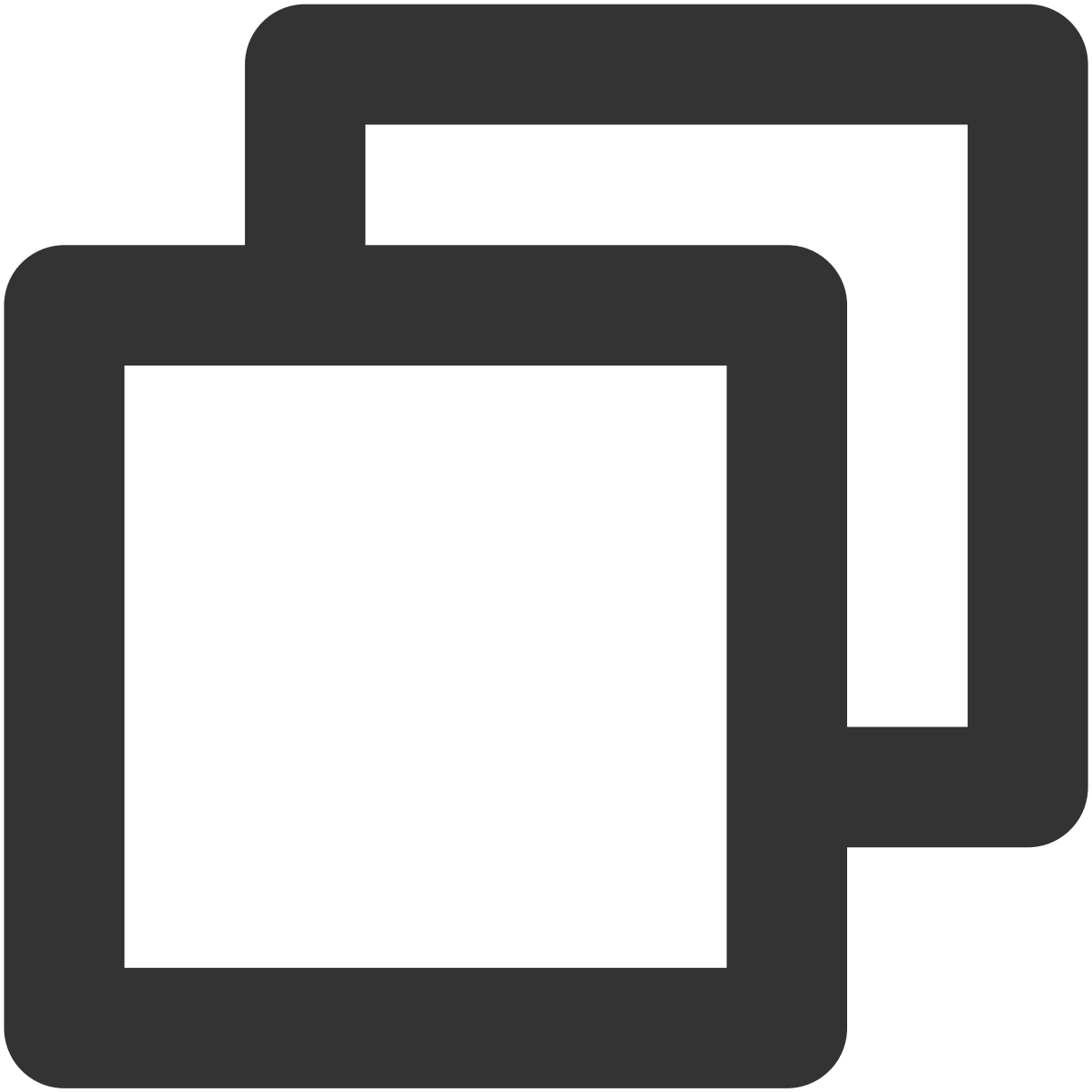
```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_ENABLE; // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;     // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output.
mTRTCCloud.setVideoEncoderMirror(true);
```

Issues with camera scale, focus, and switch.

In live showroom scenarios, the anchor may need to custom adjust the camera settings. The TRTC SDK's device management class provides APIs for these needs.

1. Query and set the zoom factor for the camera.

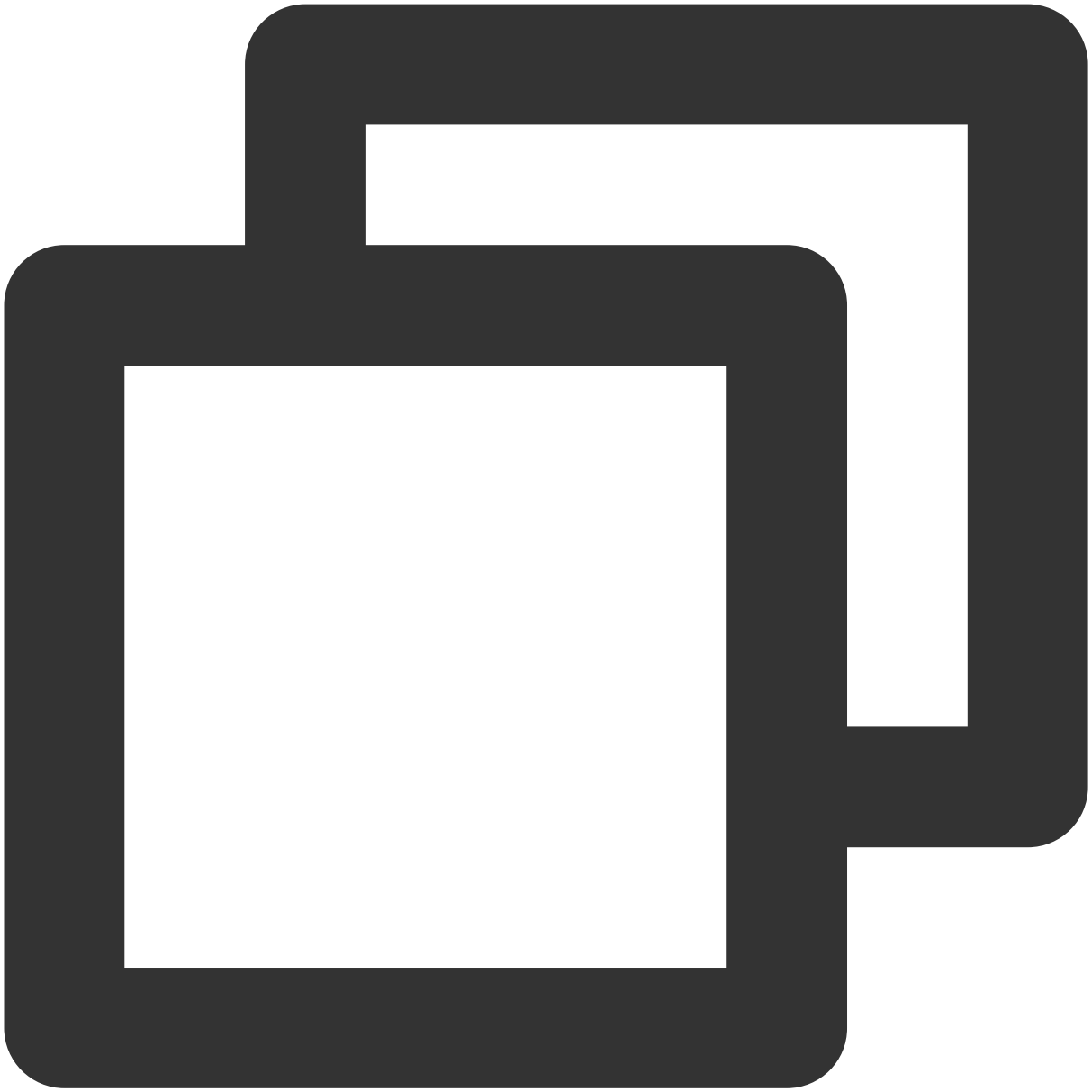
©2013-2022 Tencent Cloud. All rights reserved.



```
// Get the maximum zoom factor for the camera (only for mobile devices).
float zoomRatio = mTRTCCloud.getDeviceManager().getCameraZoomMaxRatio();
// Set the zoom factor for the camera (only for mobile devices).
// Value range is 1 - 5. 1 means the furthest field of view (normal lens), and 5 me
mTRTCCloud.getDeviceManager().setCameraZoomRatio(zoomRatio);
```

2. Set the focus feature and position of the camera.

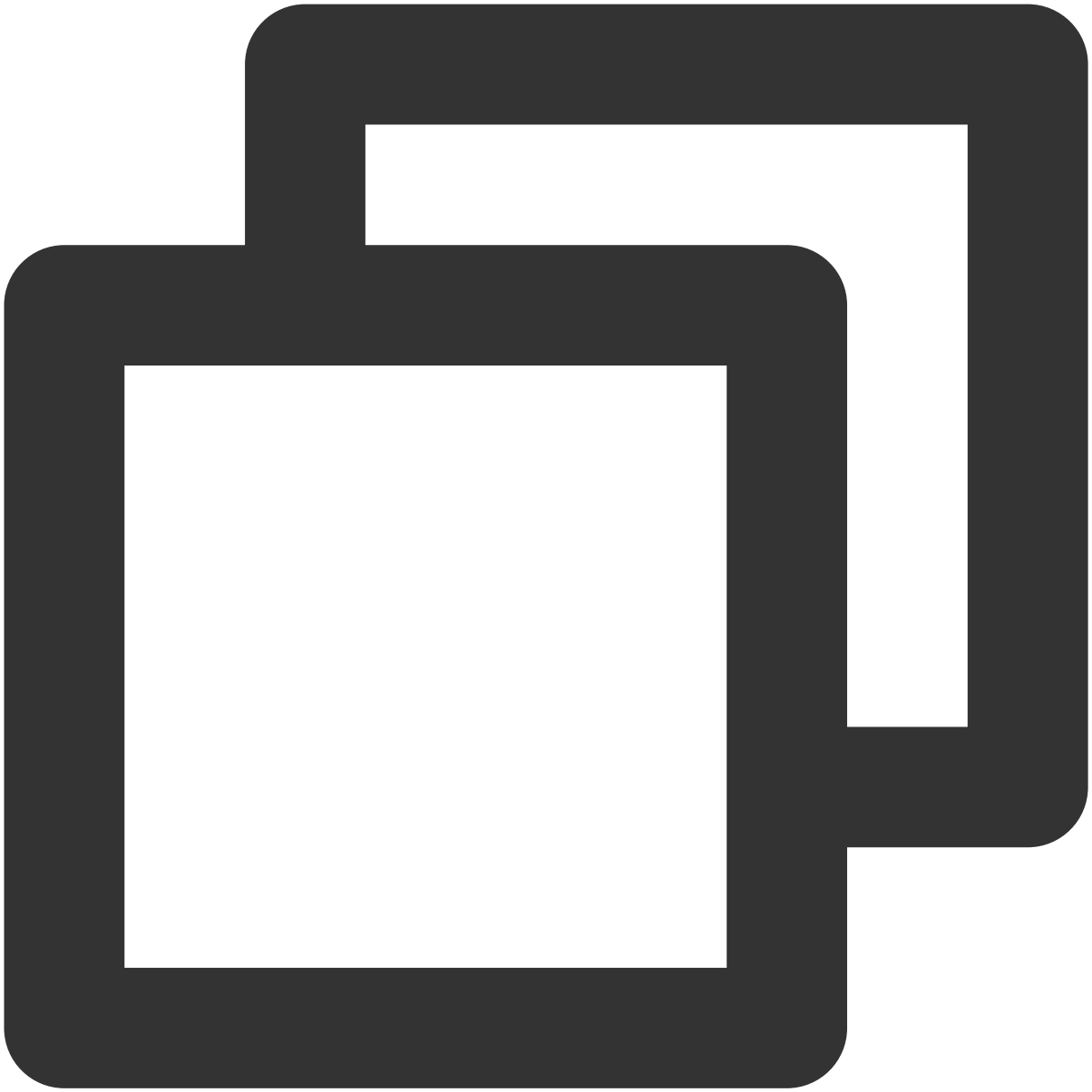
©2013-2022 Tencent Cloud. All rights reserved.



```
// Enable or disable the camera's autofocus feature (only for mobile devices).
mTRTCCloud.getDeviceManager().enableCameraAutoFocus(false);
// Set the focus position of the camera (only for mobile devices).
// The precondition for using this API is to first disable the autofocus feature us
mTRTCCloud.getDeviceManager().setCameraFocusPosition(int x, int y);
```

3. Determine and switch to front or rear cameras.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Determine if the current camera is the front camera (only for mobile devices).
boolean isFrontCamera = mTRTCCloud.getDeviceManager().isFrontCamera();
// Switch to front or rear cameras (only for mobile devices).
// Passing true means switching to front, and passing false means switching to rear
mTRTCCloud.getDeviceManager().switchCamera(!isFrontCamera);
```

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-18 14:27:33

Business Process

This section summarizes some common business processes in live showroom, helping you better understand the implementation process of the entire scenario.

Anchor starts and ends the live streaming.

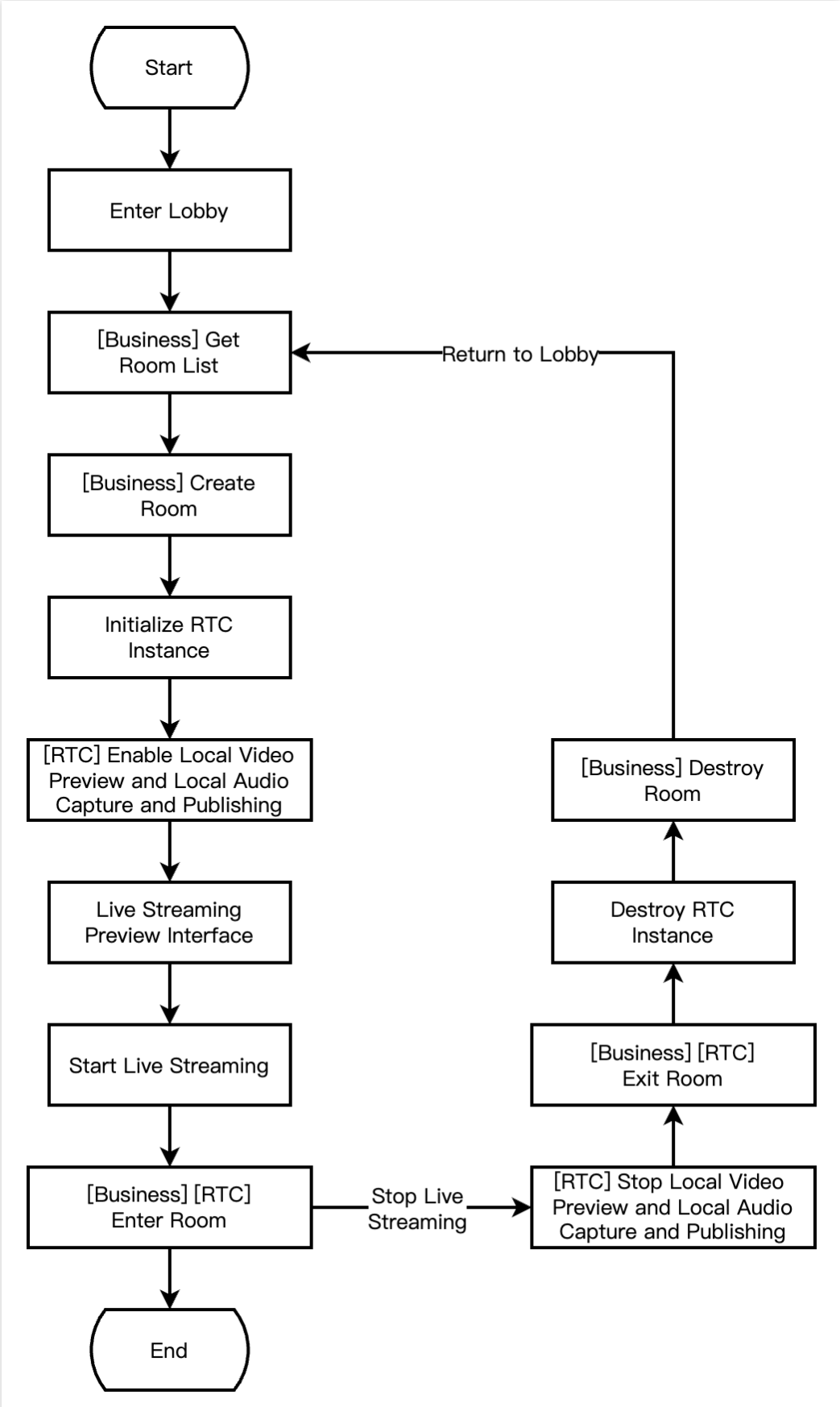
The anchor initiates the cross-room competition.

The RTC audience enters the room for mic-connection.

The CDN audience enters the room for mic-connection.

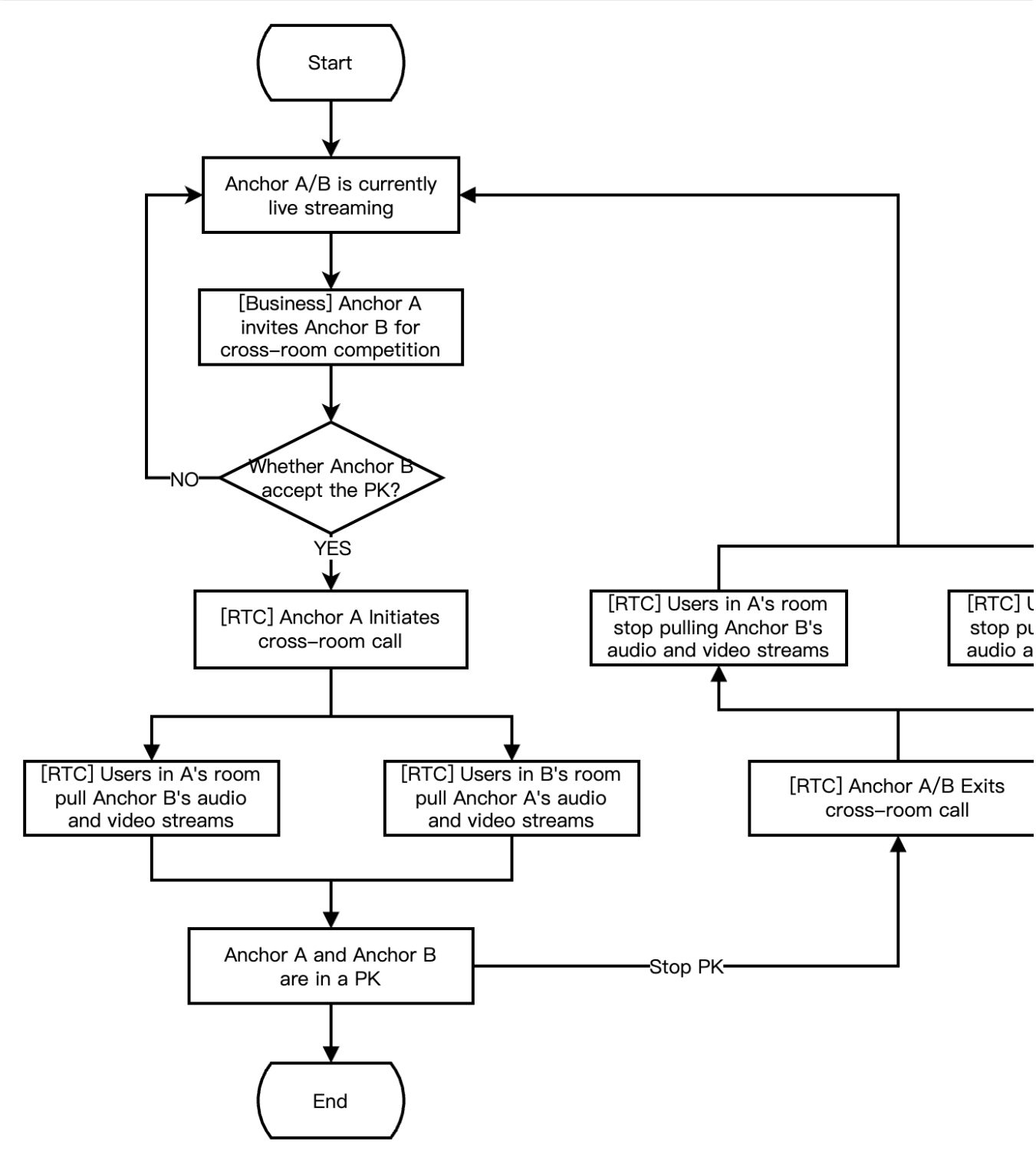
The following figure shows the process of an anchor (room owner) local preview, creating a room, entering the room to start the live streaming, and leaving the room to end the live streaming.

©2013-2022 Tencent Cloud. All rights reserved.



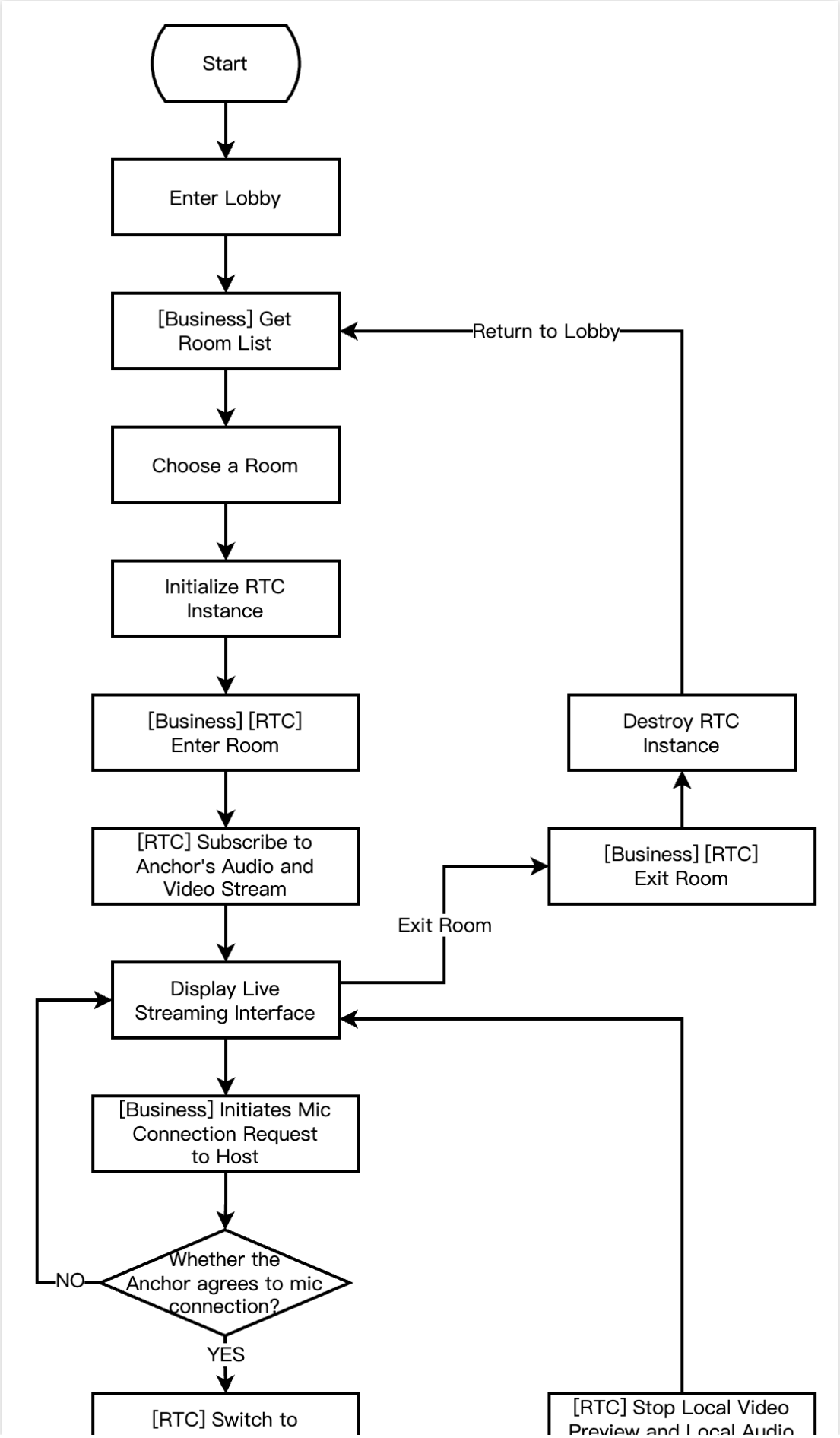
©2013-2022 Tencent Cloud. All rights reserved.

The following figure shows the process of Anchor A inviting Anchor B for a cross-room competition. During the cross-room competition, the audiences in both rooms can see the PK mic-connection live streaming of the two room owners.

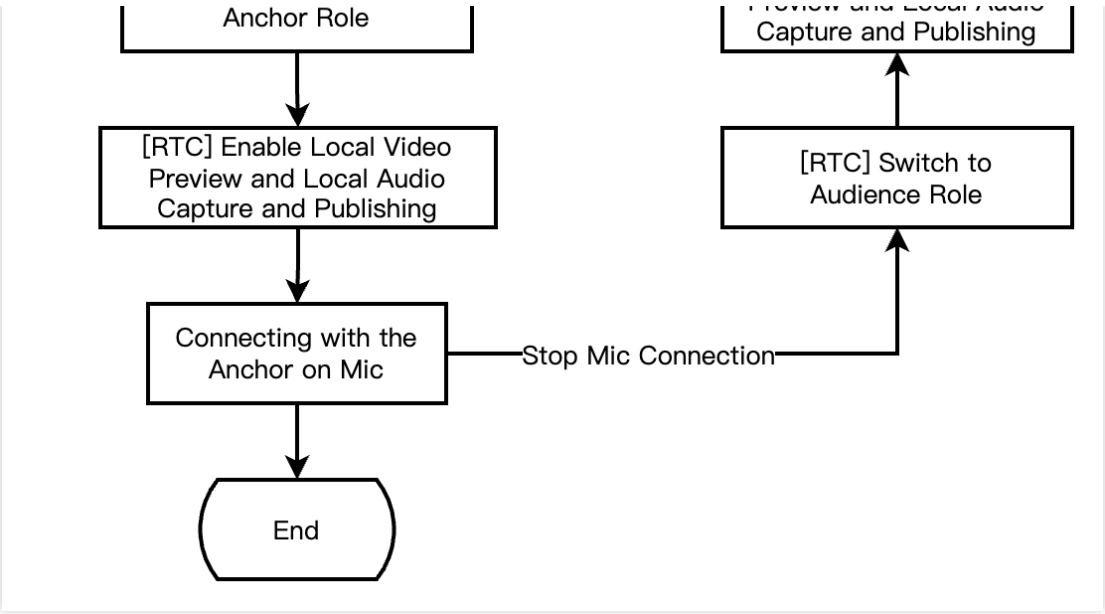


The following figure shows the process for RTC live interactive streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

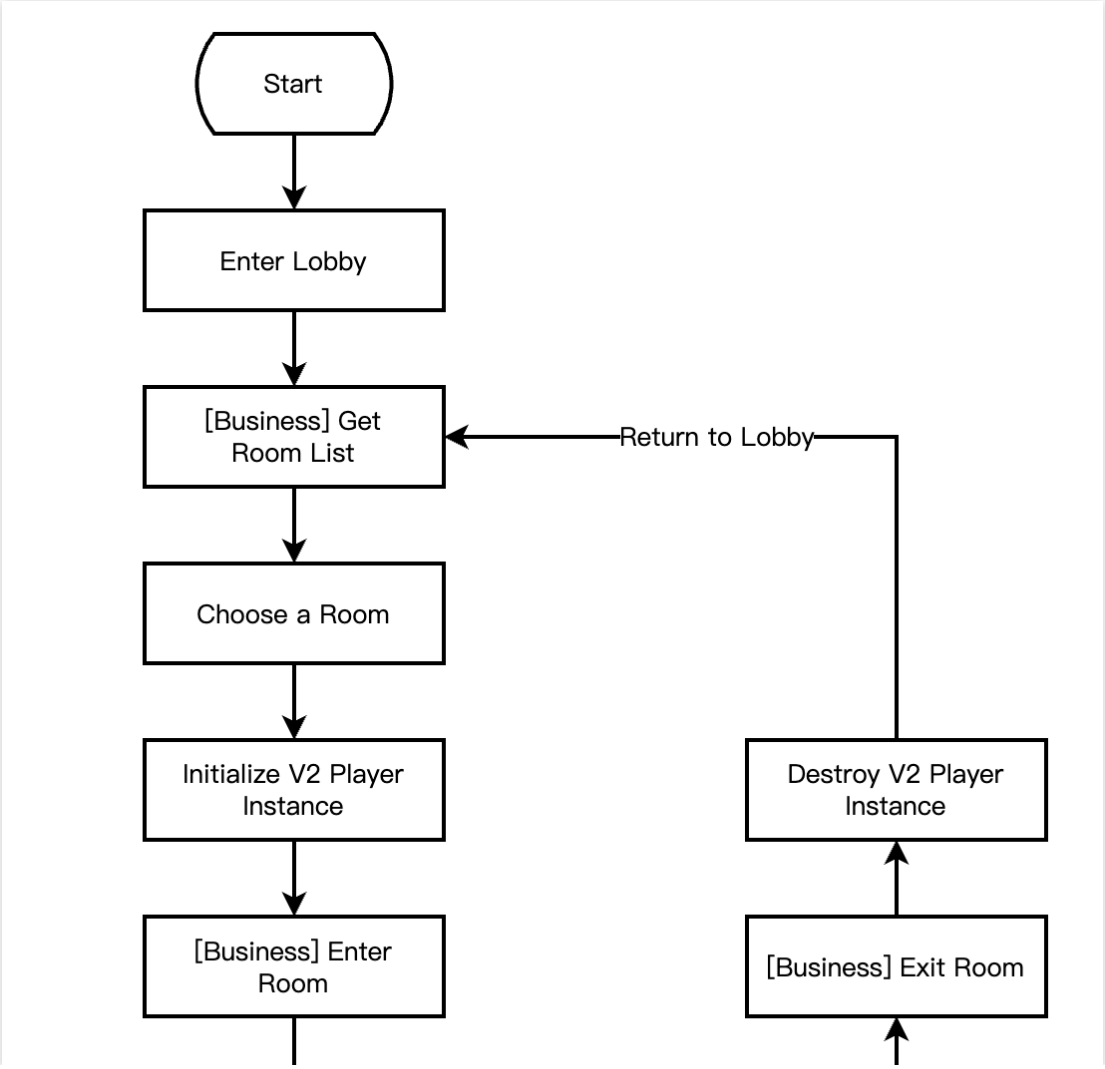
©2013-2022 Tencent Cloud. All rights reserved.



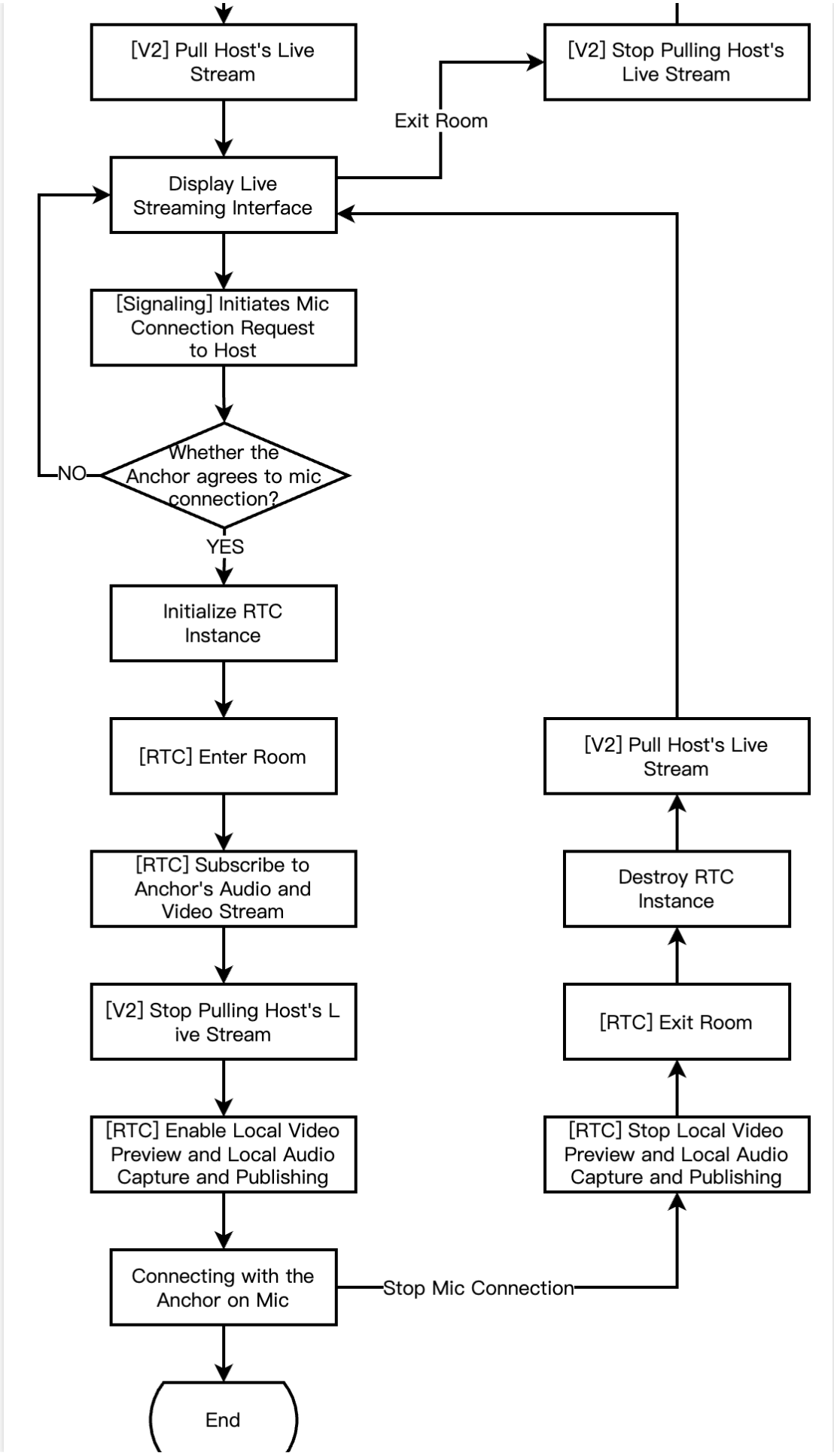
©2013-2022 Tencent Cloud. All rights reserved.



The following figure shows the process for RTC CDN live streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.



©2013-2022 Tencent Cloud. All rights reserved.



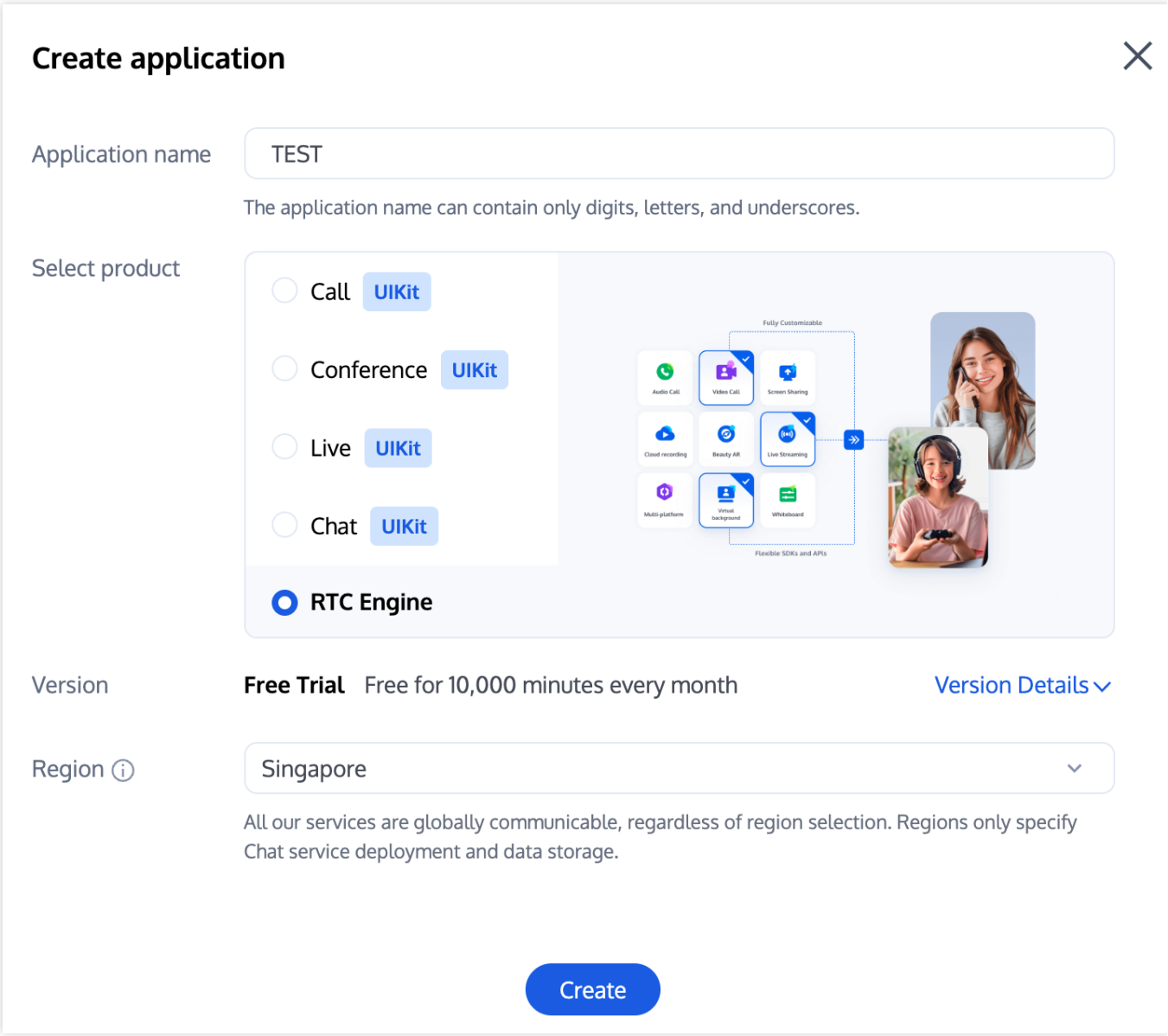
©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparation

Step 1. Activating the service.

Live showroom scenarios usually require two paid PaaS services from Tencent Cloud Tencent Real-Time Communication (TRTC) and Tencent Effect for construction. TRTC is responsible for providing real-time audio and video interaction capabilities. Tencent Effect is responsible for providing beauty effects capabilities. If you use a third-party beauty effect product, you can disregard the Tencent Effect integration part.
Activate TRTC service.
Activate Tencent Effect service.
1. First, you need to log in to the Tencent Real-Time Communication (TRTC) console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



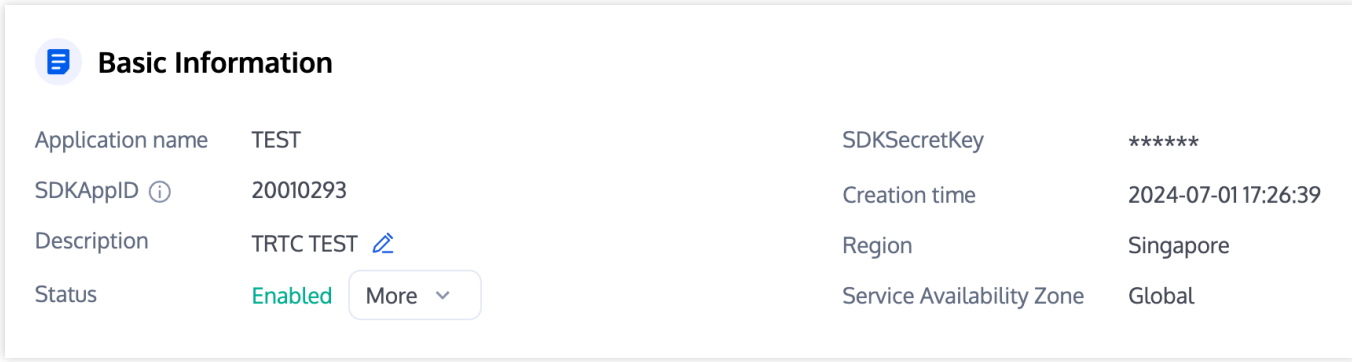
Note:

It is recommended to create two applications for testing and production environments, respectively. Each Tencent Cloud account (UIN) is given 10,000 minutes of free duration every month for one year.

TRTC offers monthly subscription plans including the experience edition (default), basic edition, and professional edition. Different value-added feature services can be unlocked. For details, see Version Features and Monthly Subscription Plan Instructions.

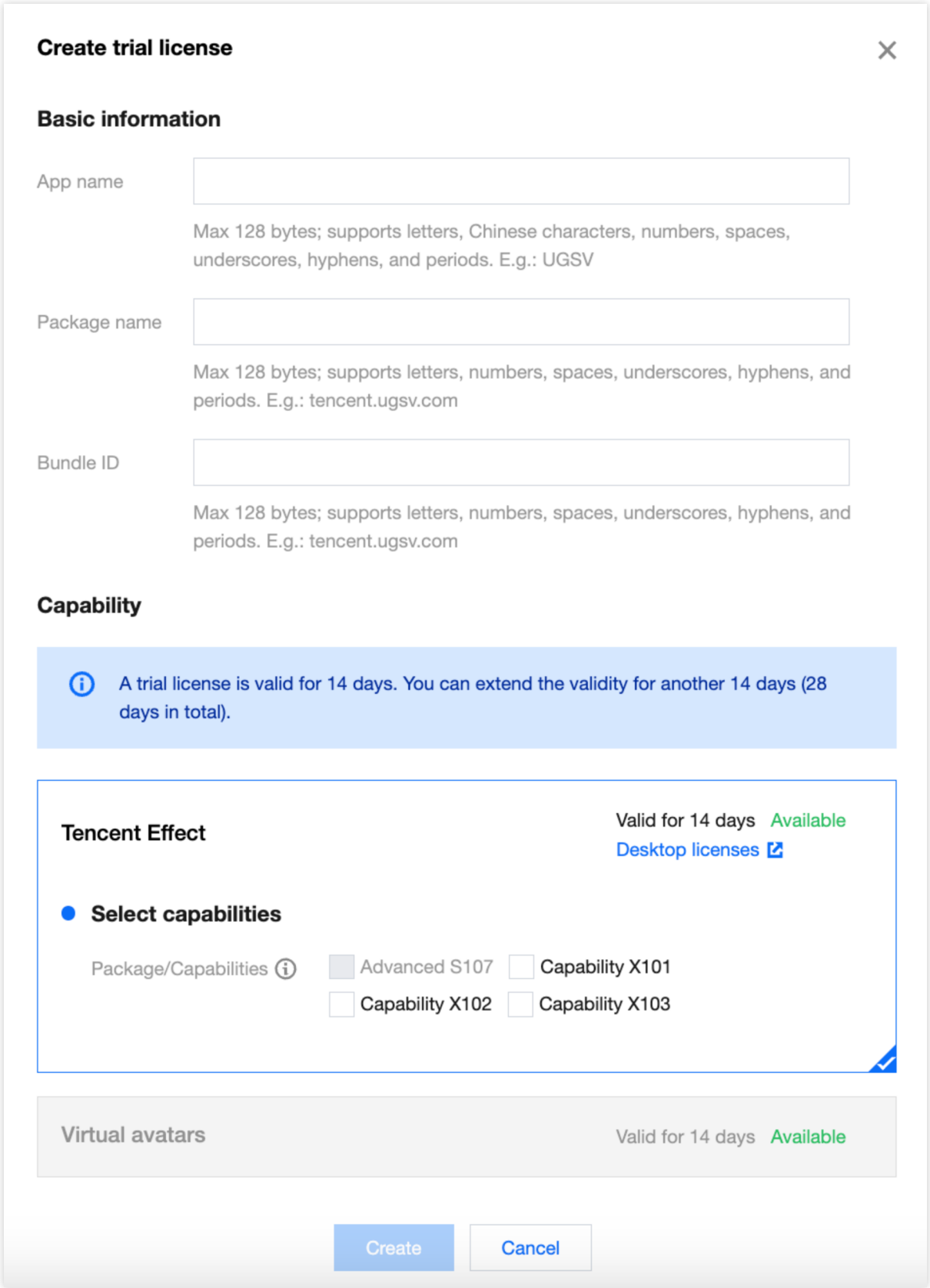
2. After an application is created, you can see the basic information of the application in the Application Management - Application Overview section. It is important to keep the **SDKAppID** and **SDKSecretKey** safe for later use and to avoid key leakage that could lead to traffic theft.

©2013-2022 Tencent Cloud. All rights reserved.



1. Log in to Tencent Cloud Tencent Effect console > Mobile License. Click **Create Trial License** (the free trial validity period for Trial Version License is 14 days. It is extendable once for a total of 28 days). Fill in the actual requirements for `App Name` , `Package Name` and `Bundle ID` . Choose **Tencent Effect**, and choose the capabilities to be tested: Advanced Package S1-07, Atomic Capability X1-01, Atomic Capability X1-02, and Atomic Capability X1-03. **After you check it, accurately fill in the company name, and industry type**. **Upload** company service license, click **OK** to submit the review application, and wait for the manual review process.

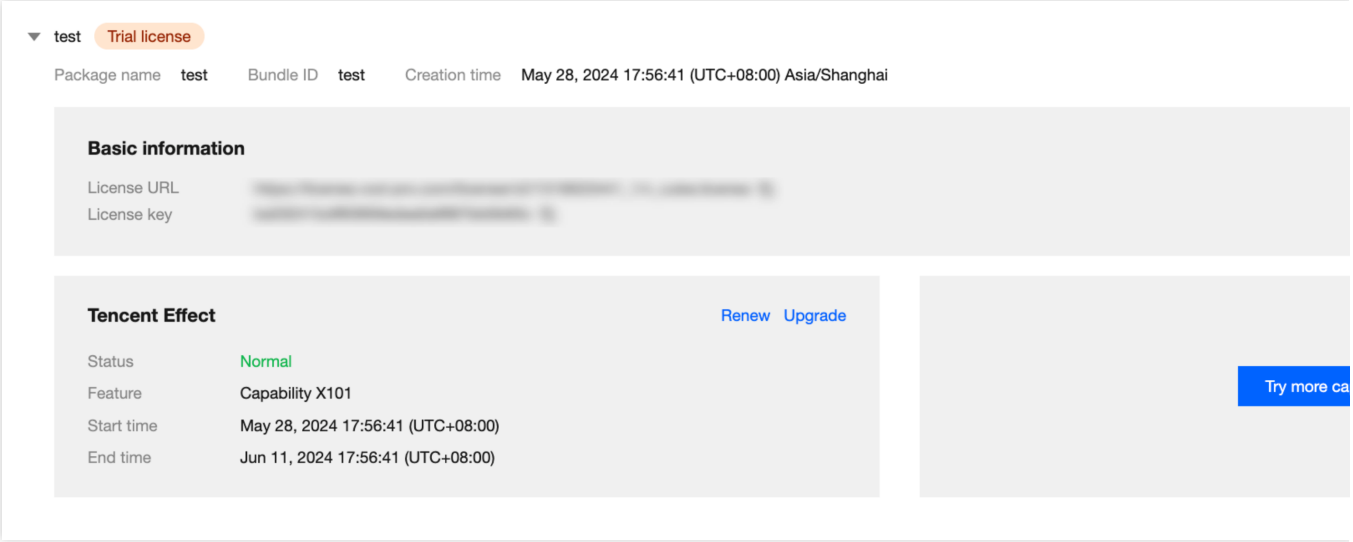
©2013-2022 Tencent Cloud. All rights reserved.



2. After the trial version License is successfully created, the page will display the generated License information. At this time, the License URL and License Key parameters are not yet effective and will only become active after the

©2013-2022 Tencent Cloud. All rights reserved.

submission is approved. **When configuring SDK initialization, you need to input both the License URL and License Key parameters. Keep the following information secure.**



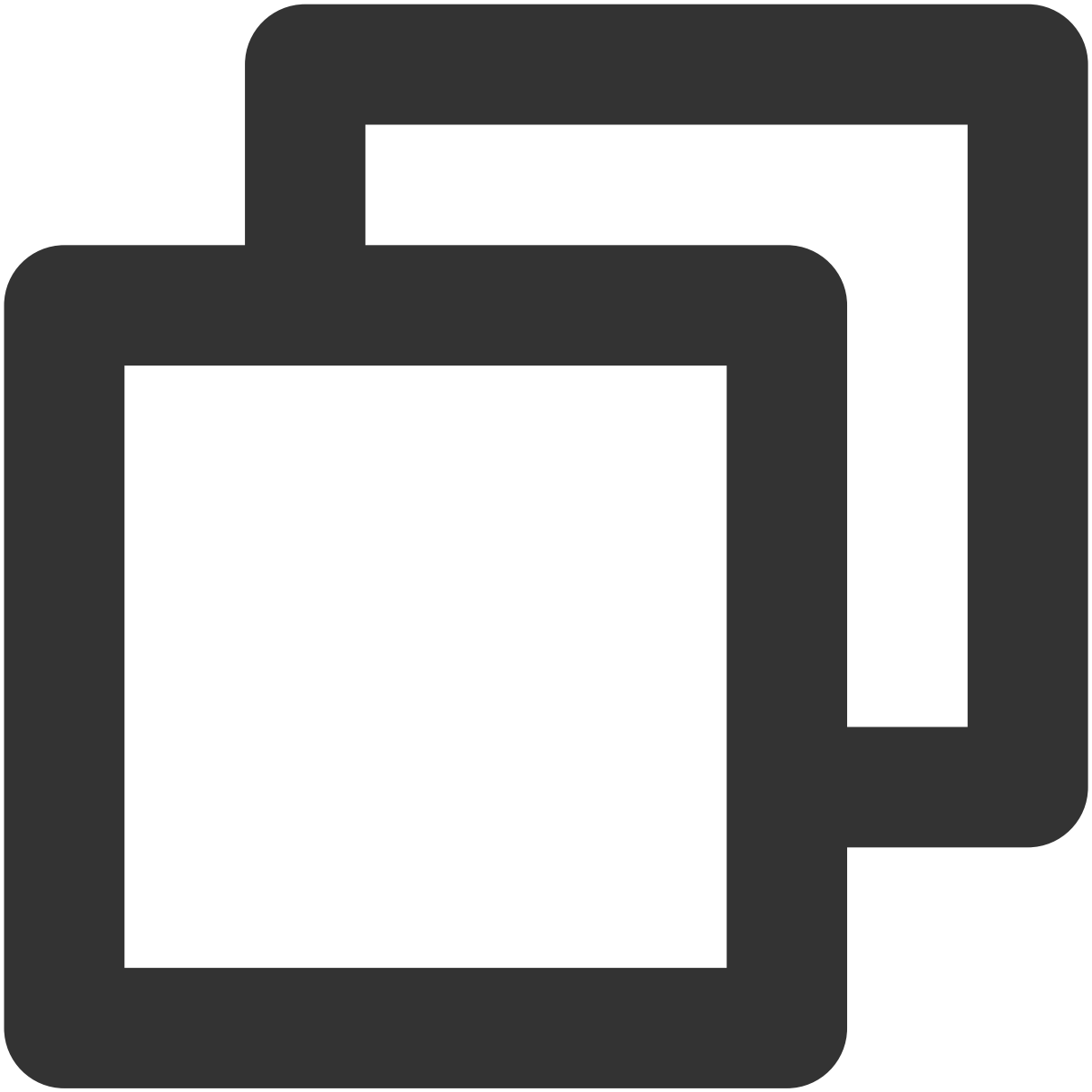
Step 2: Importing SDK.

TRTC SDK and Tencent Effect SDK have been released to the **CocoaPods** repository. You can integrate them via CocoaPods.

1. Install CocoaPods.

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

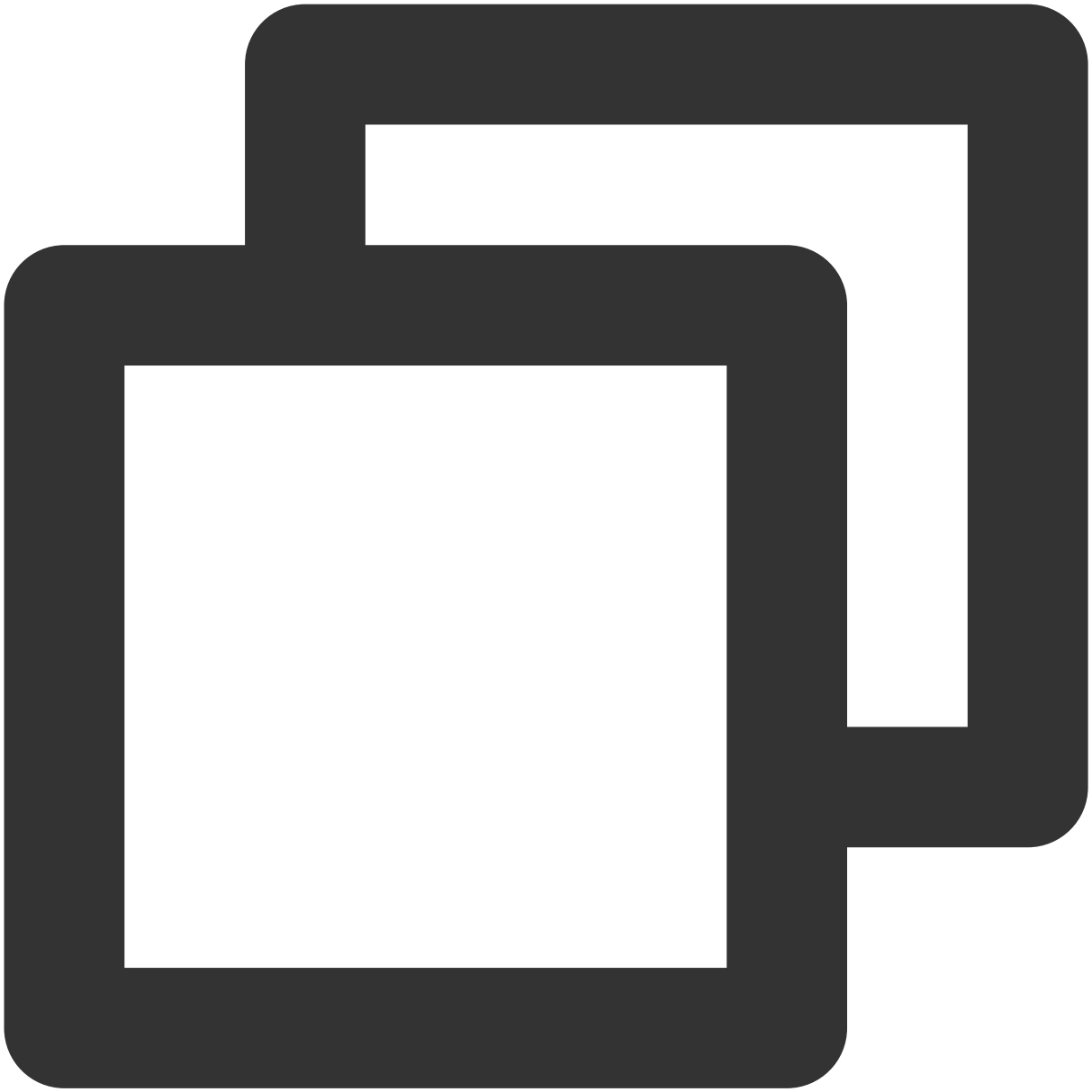


```
sudo gem install cocoapods
```

2. Create a Podfile file.

Go to the project directory, and enter the following command. A Podfile file will then be created in the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

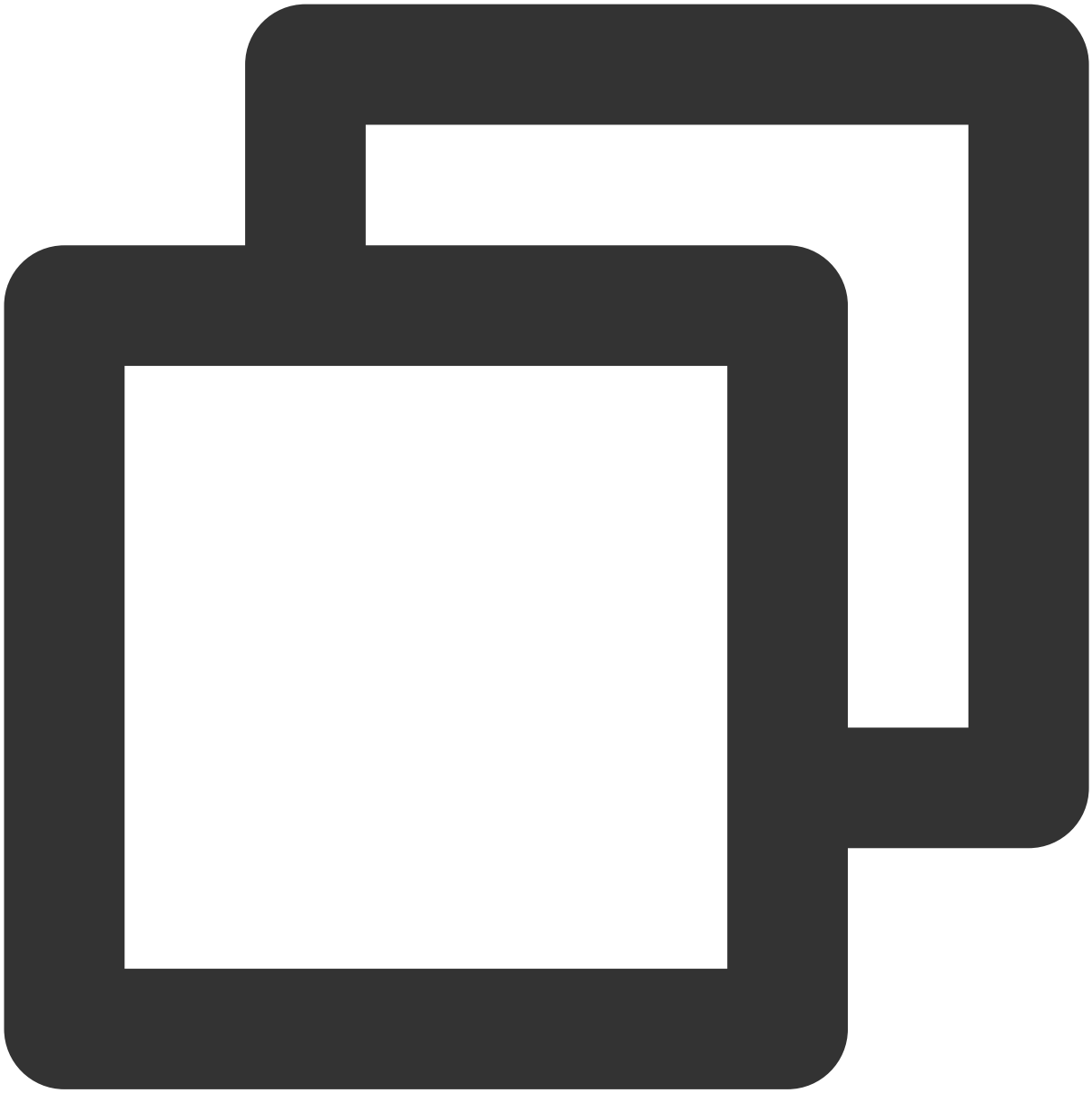


```
pod init
```

3. Edit the Podfile file.

Choose an appropriate version for your project and edit the Podfile file:

©2013-2022 Tencent Cloud. All rights reserved.



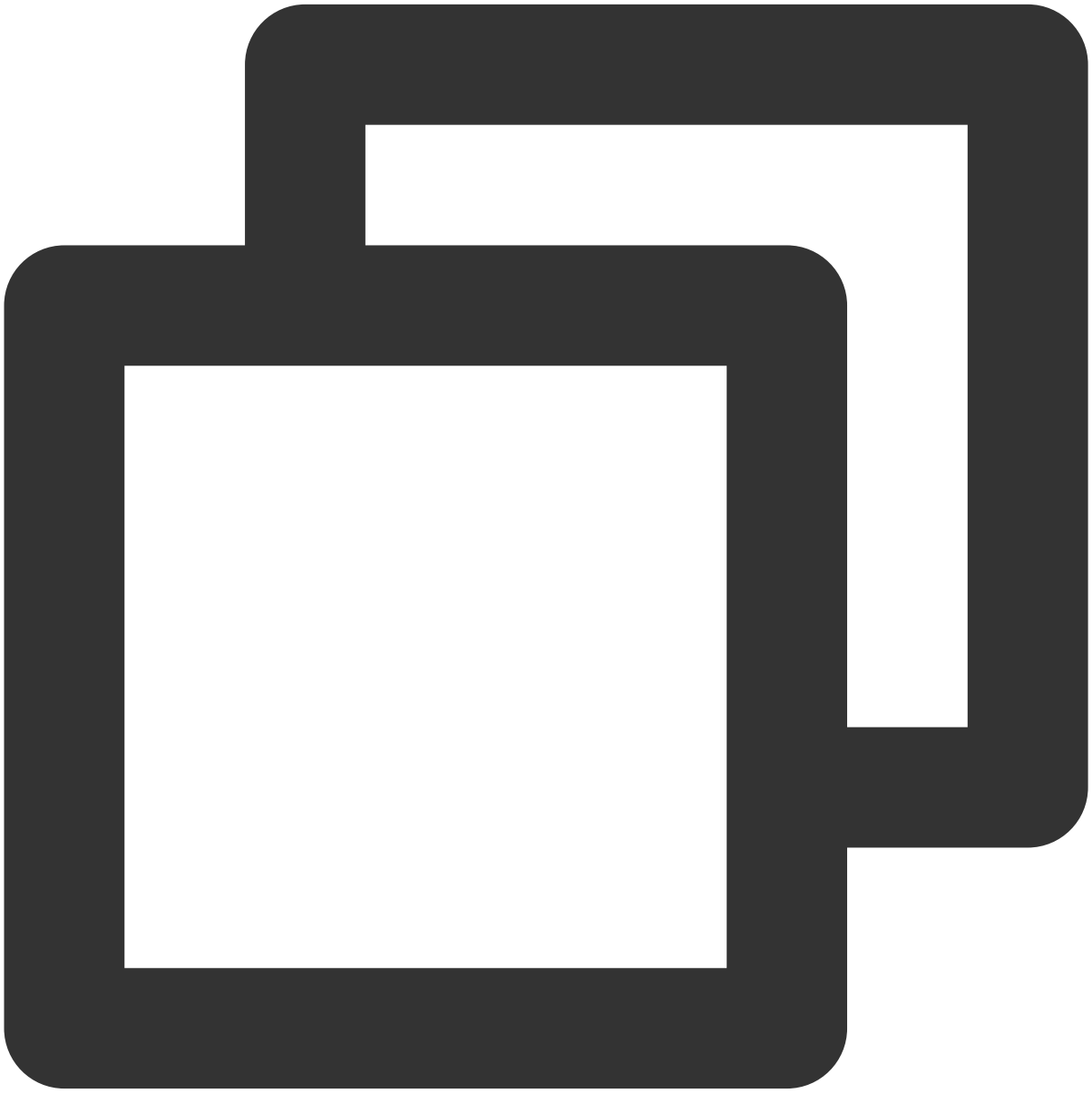
```
platform :ios, '8.0'
    target 'App' do

    # TRTC Lite Edition
    # The installation package has the minimum incremental size. But it only suppor
    pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd

    # Pro Edition
    # Includes a wide range of features such as Real-Time Communication (TRTC), TXL
    # pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/po
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    # Tencent Effect SDK example of S1-07 package is as follows:
    pod 'TencentEffect_S1-07'

end
```

4. Update and install the SDK.

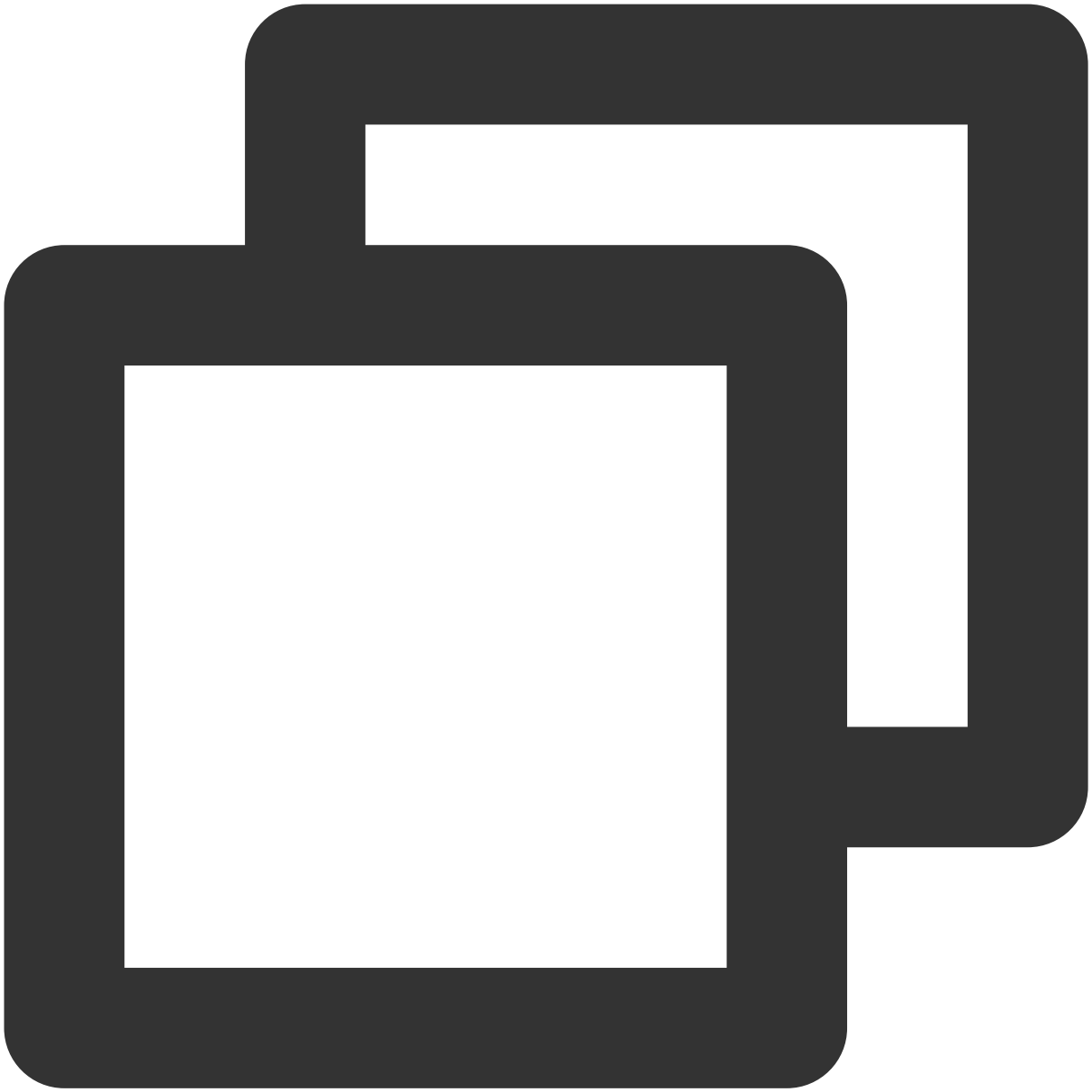
Enter the following command in a terminal window to update the local repository files and install the SDK:



```
pod install
```

Or run this command to update the local repository:

©2013-2022 Tencent Cloud. All rights reserved.



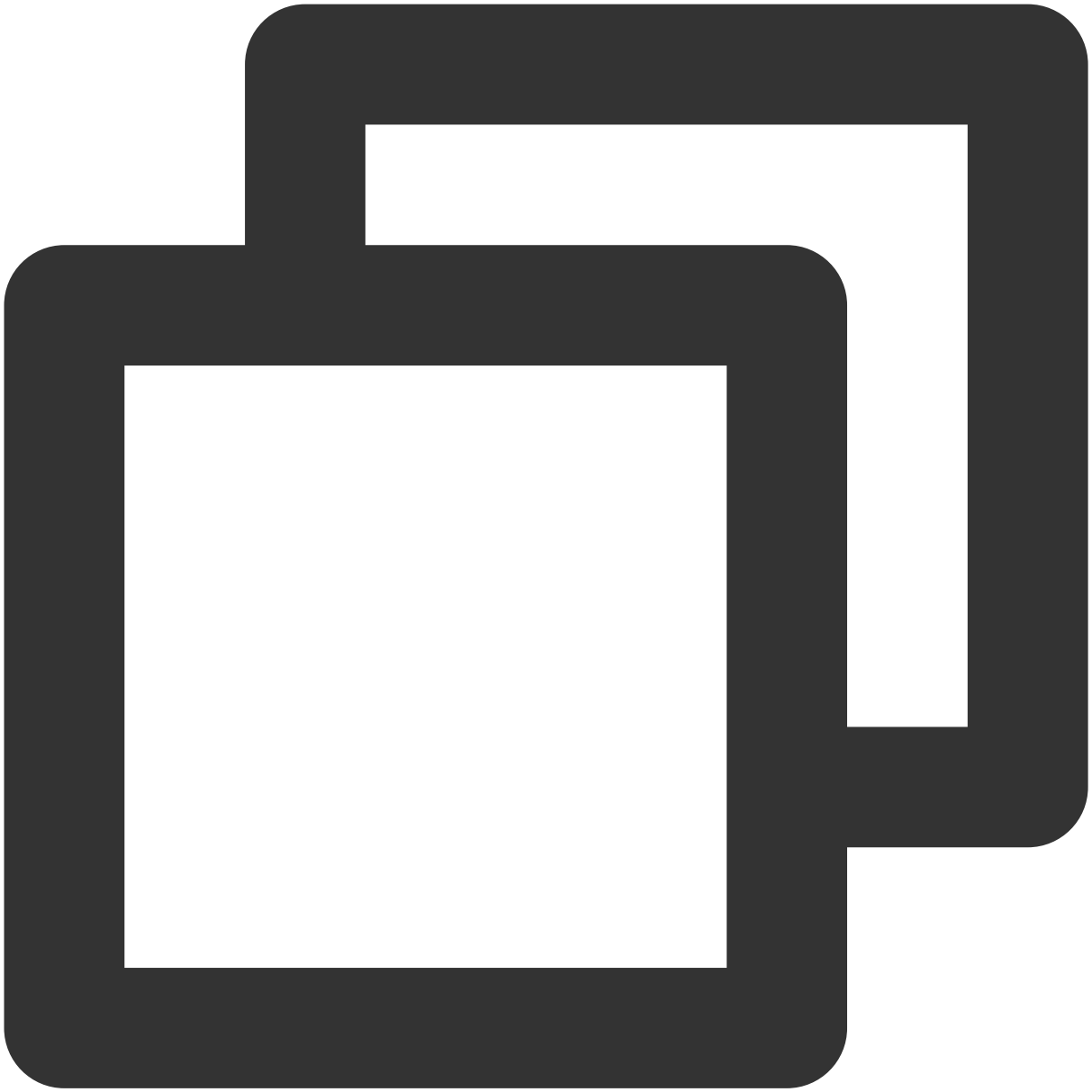
```
pod update
```

Upon the completion of pod command execution, an .xcworkspace project file integrated with the SDK will be generated. Double-click to open it.

Note:

If the pod search fails, it is recommended to try updating the pod's local repo cache. Update command is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
pod setup
pod repo update
rm ~/Library/Caches/CocoaPods/search_index.json
```

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manually Integrating Tencent Effect SDK.

5. Add beauty resources to the actual project.

Download and unzip the corresponding package of SDK and Beauty Resources. Add the bundle resources under the resources/motionRes folder to the actual project.

©2013-2022 Tencent Cloud. All rights reserved.

On the Build Settings, under Other Linker Flags, add `-ObjC` .

6. Modify the Bundle Identifier to match the applied trial authorization.

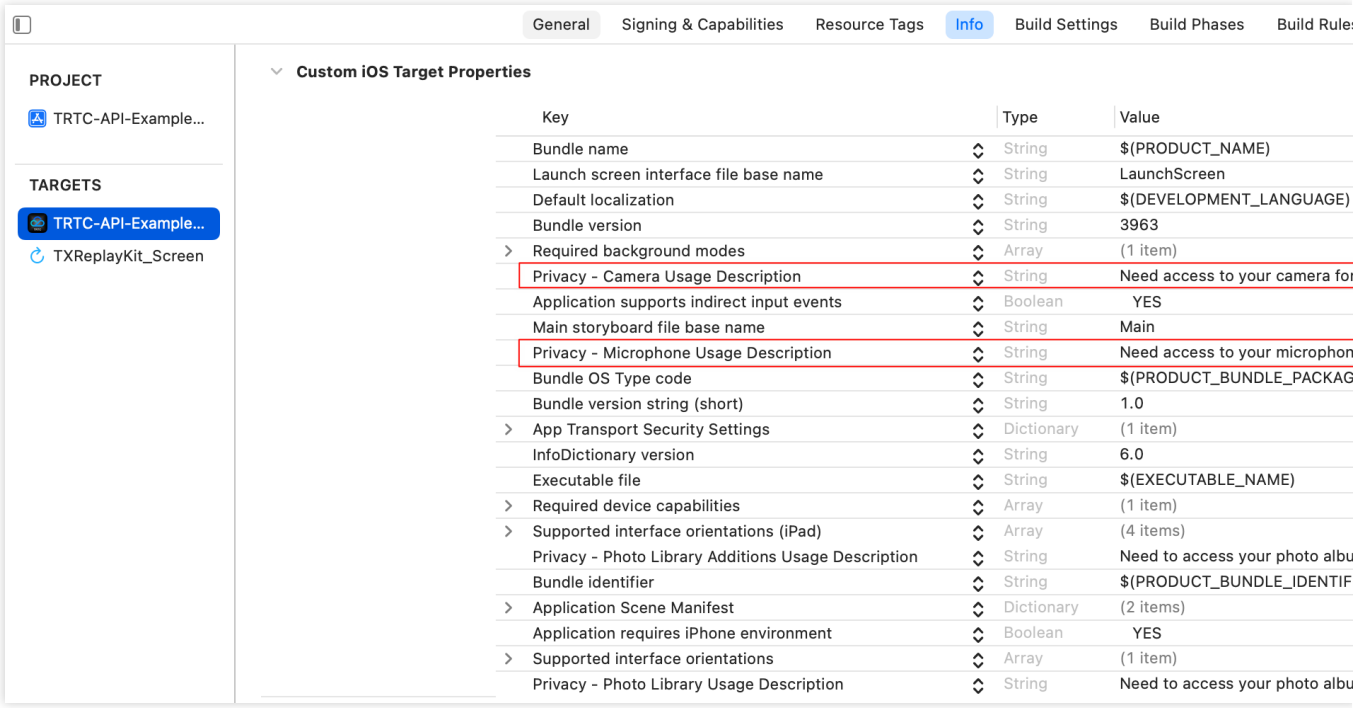
Step 3: Project configuration.

1. Configure permissions.

For live showroom scenarios, TRTC SDK and Tencent Effect SDK require the following permissions. Add the following two items to the App's Info.plist, corresponding to the microphone and camera prompts in the system pop-up authorization dialog box.

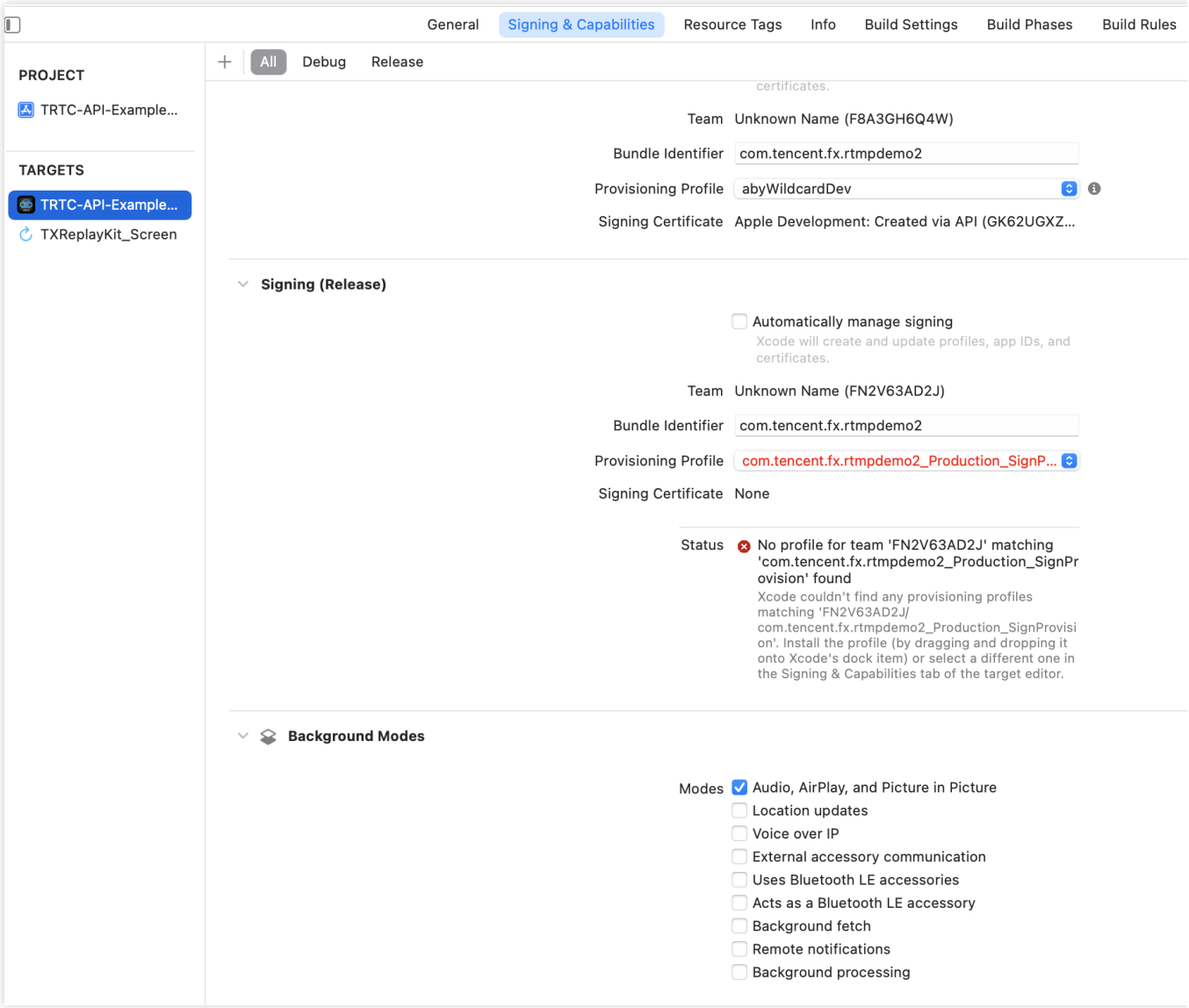
Privacy - Microphone Usage Description. Enter a prompt specifying the purpose of microphone use.

Privacy - Camera Usage Description. Enter a prompt specifying the purpose of camera use.



2. If you need your App to continue running certain features in the background, go to XCode. Choose your current project. Under Capabilities, set the settings for Background Modes to ON, and check Audio, AirPlay, and Picture in Picture, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Step 4: Authentication and authorization.

TRTC authentication credential.

Tencent Effect authentication license.

UserSig is a security protection signature designed by Tencent Cloud to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when it enters the room.

Debugging Stage: UserSig can be generated through two methods for debugging and testing purposes only: client sample code and console access.

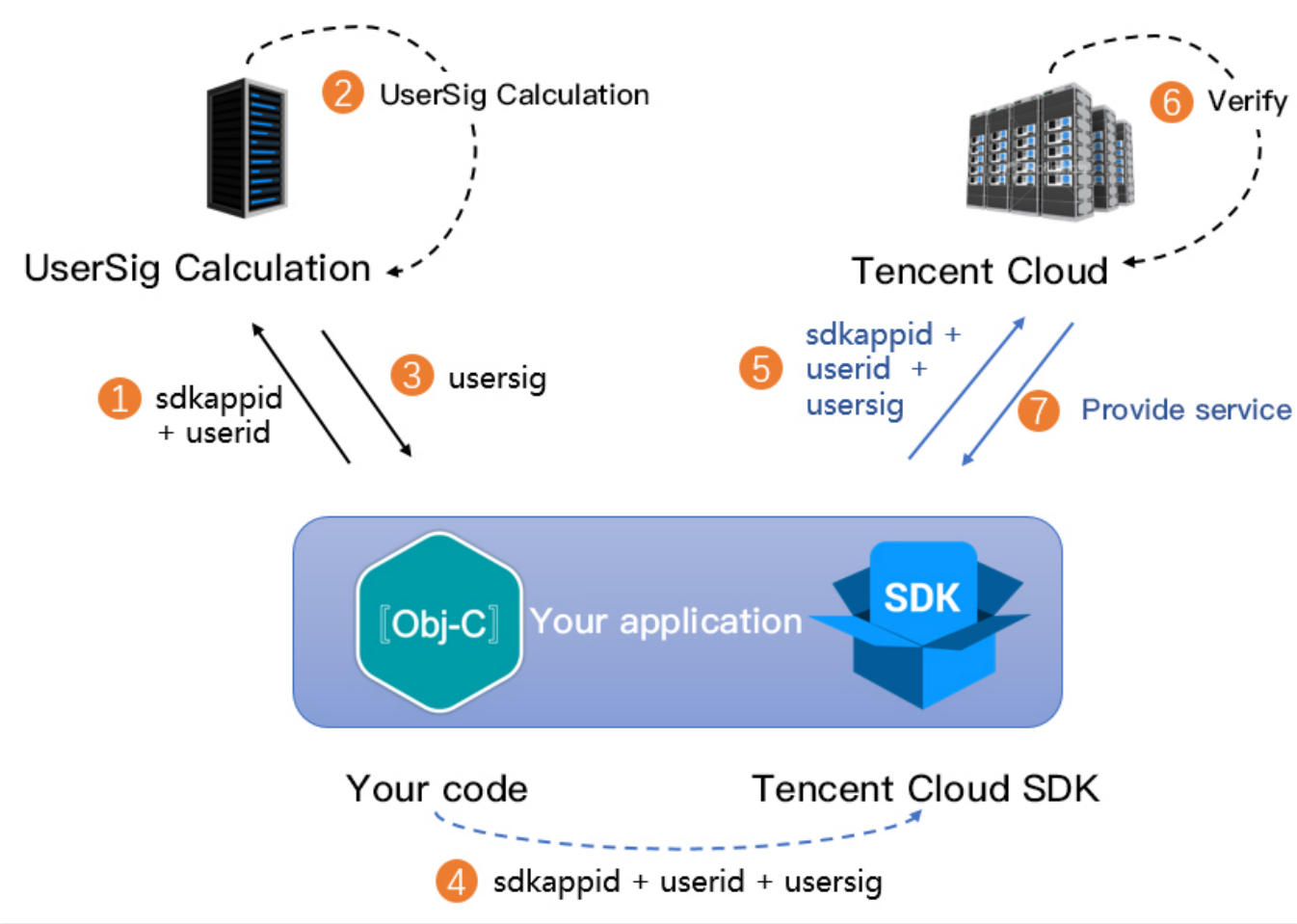
Formal Operation Stage: It is recommended to use a higher security level server computation for generating UserSig. This is to prevent key leakage due to client reverse engineering.

The specific implementation process is as follows:

1. Before calling the SDK's initialization function, your app must first request UserSig from your server.
2. Your server computes the UserSig based on the SDKAppID and UserID.

©2013-2022 Tencent Cloud. All rights reserved.

3. The server returns the computed UserSig to your app.
4. Your app passes the obtained UserSig into the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to Tencent Cloud CVM for verification.
6. Tencent Cloud verifies the UserSig and confirms its validity.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.



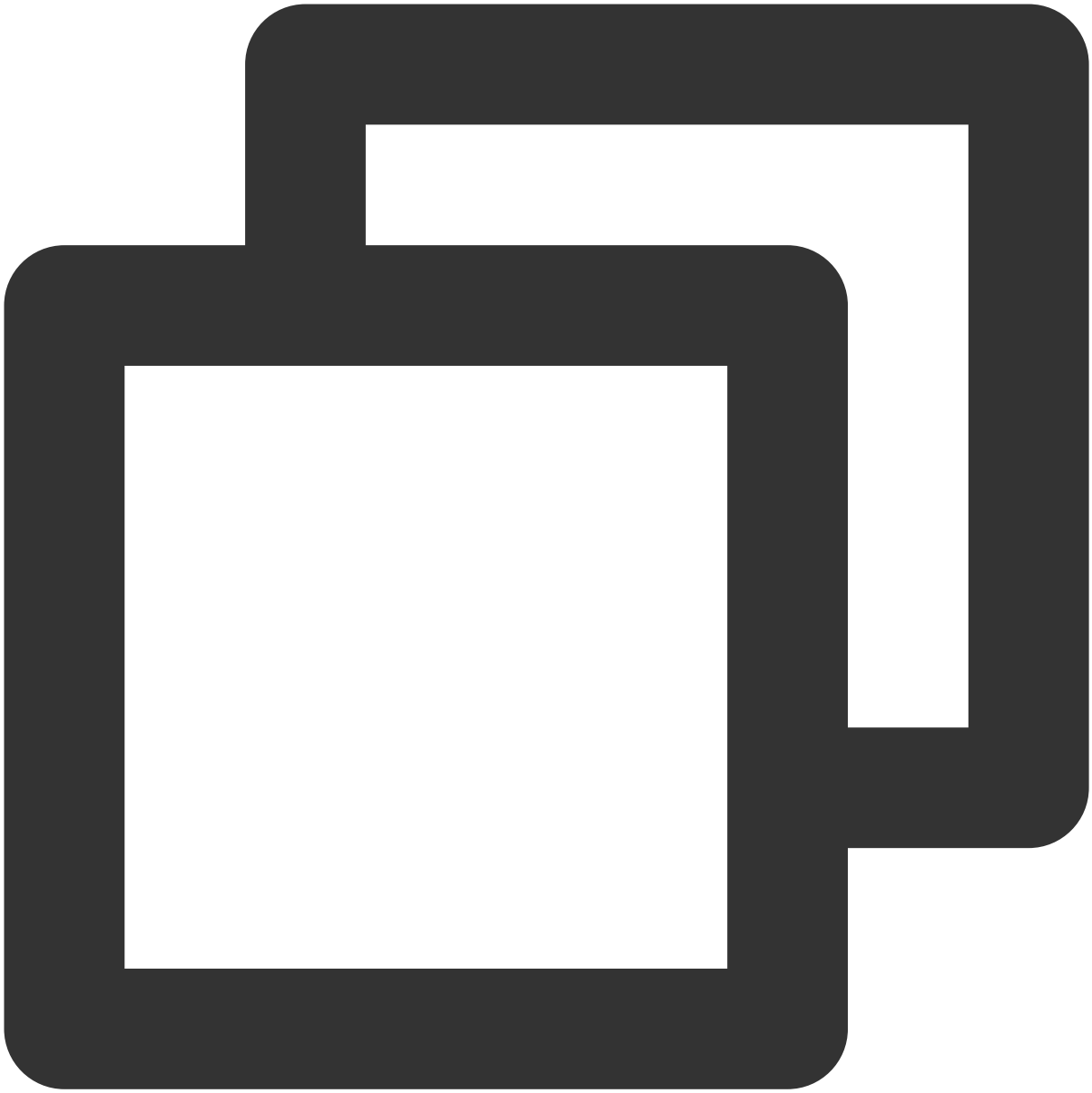
Note:

The local computation method of UserSig during the debugging stage is not recommended for application in an online environment. It is prone to reverse engineering, leading to key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Before using Tencent Effect, you need to verify the license credential with Tencent Cloud. Configuring the License requires License Key and License Url. Sample code is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
 if (authresult == TELicenseCheckOk) {
     NSLog(@"Authentication successful.");
 } else {
     NSLog(@"Authentication failed.");
 }
}];
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

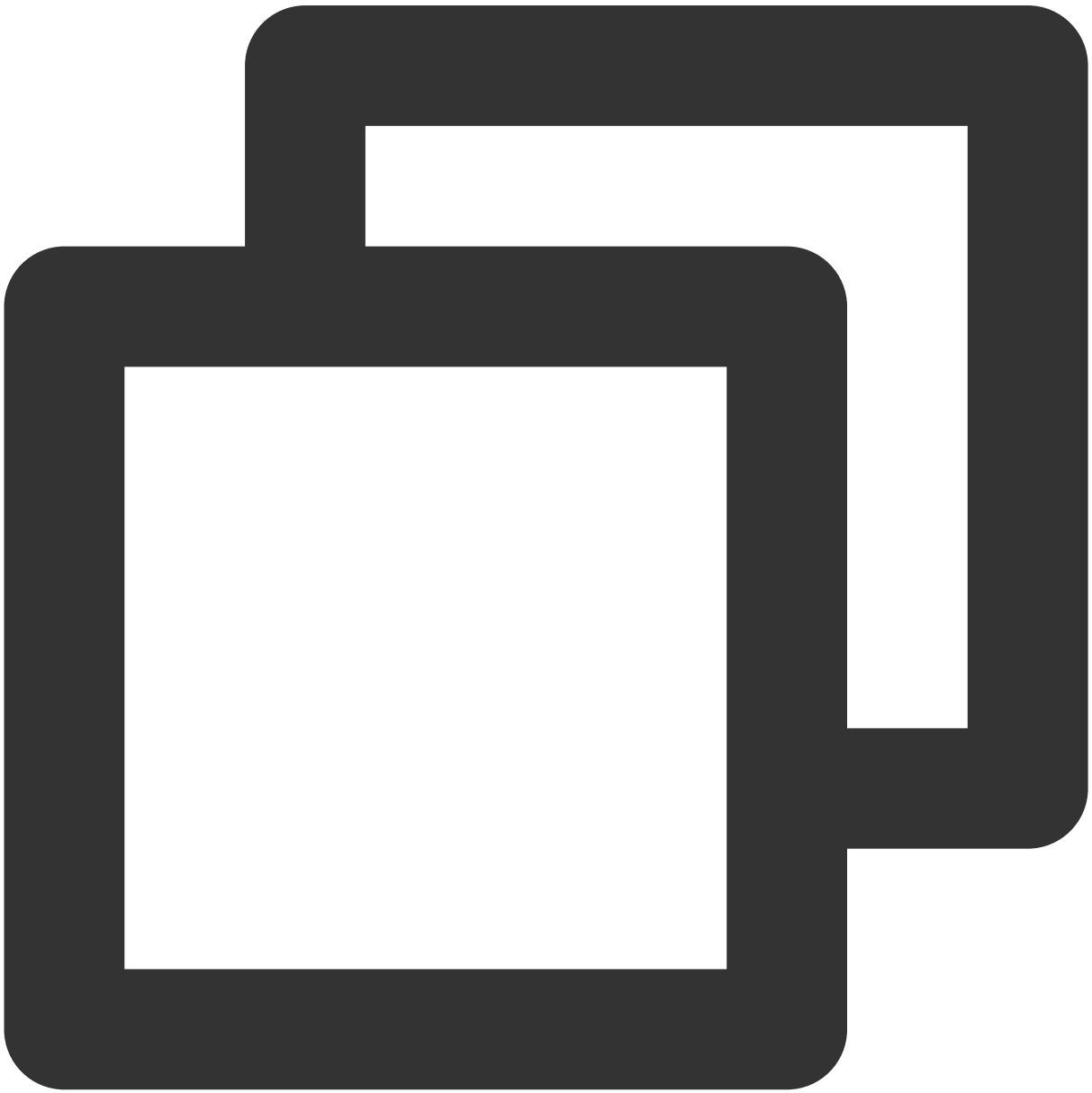
It is recommended to trigger the authentication permission in the initialization code of related business modules. Ensure to avoid having to download the License temporarily before use. Additionally, during authentication, network permissions must be ensured.

The actual application's Bundle ID must match exactly with the Bundle ID associated with the creation of License. Otherwise, it will lead to License verification failure. For details, see Authentication Error Code.

Step 5: Initializing the SDK.

Initialize the TRTC SDK.

Initialize the Tencent Effect SDK.



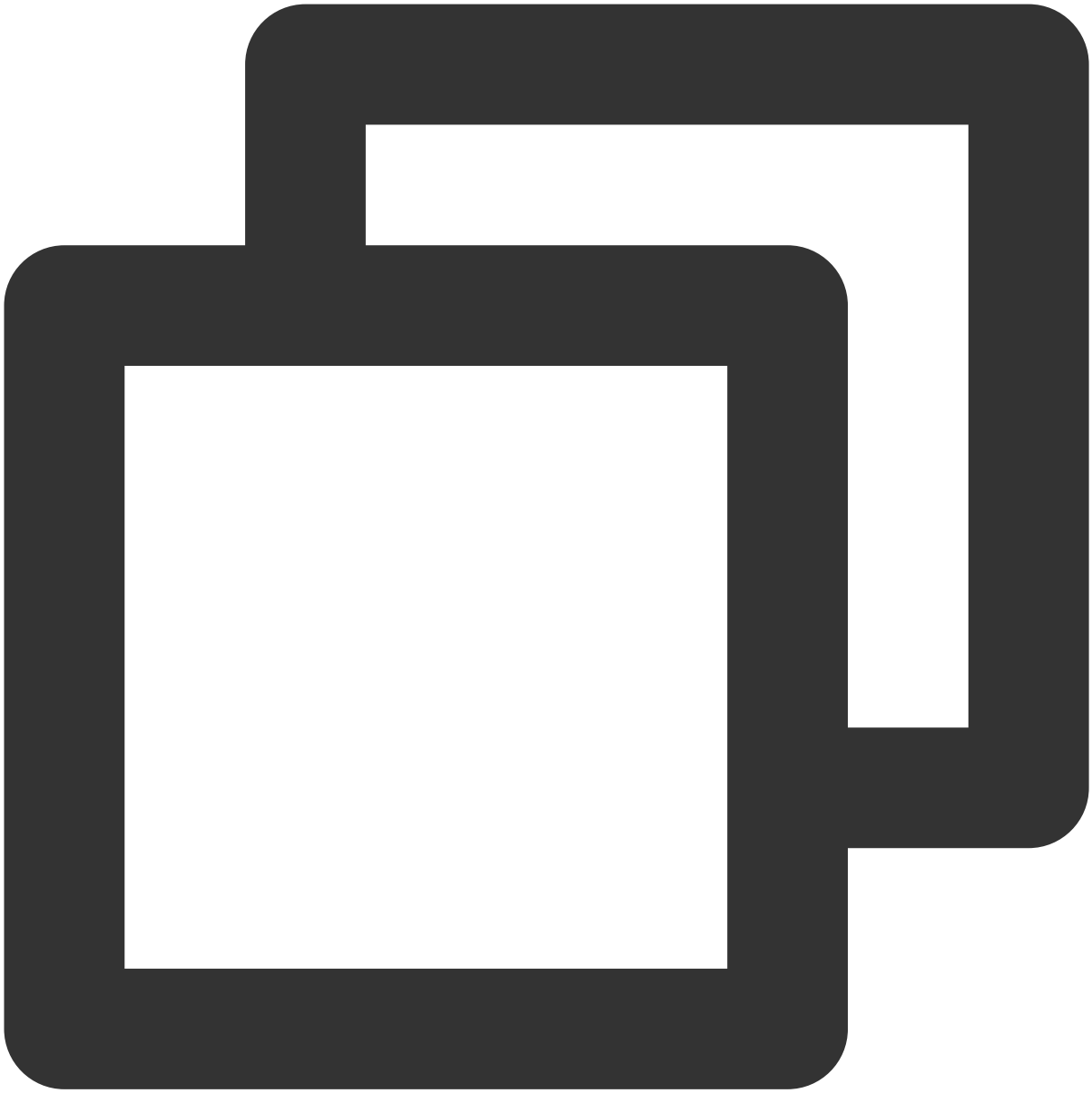
©2013-2022 Tencent Cloud. All rights reserved.

```
// Create TRTC SDK instance (Single Instance Pattern).
self.trtcCloud = [TRTCCloud sharedInstance];
// Set event listeners.
self.trtcCloud.delegate = self;

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
- (void)onError:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg extInfo:(
    NSLog(@"%d: %@", errCode, errMsg);
}

- (void)onWarning:(TXLiteAVWarning)warningCode warningMsg:(nullable NSString *)warn
    NSLog(@"%d: %@", warningCode, warningMsg);
}

// Remove event listener.
self.trtcCloud.delegate = nil;
// Terminate TRTC SDK instance (Singleton Pattern).
[TRTCCloud destroySharedIntance];
```

Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Load beauty-related resources.
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
 @"root_path":[[NSBundle mainBundle] bundlePath]
};

// Initialize the Tencent Effect SDK.
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi

// Release the Tencent Effect SDK.
[self.beautyKit deinit]
```

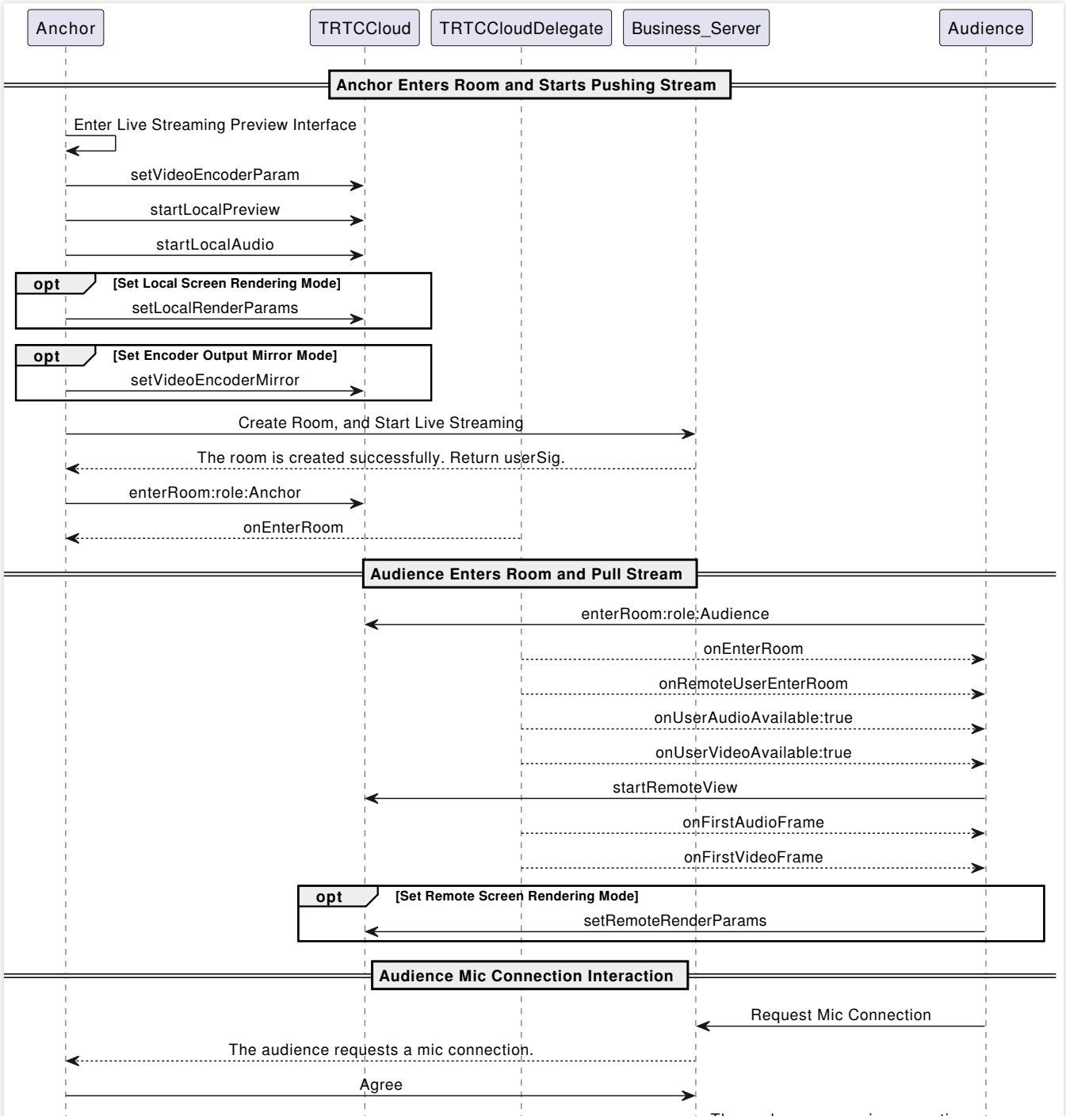
©2013-2022 Tencent Cloud. All rights reserved.

Note:

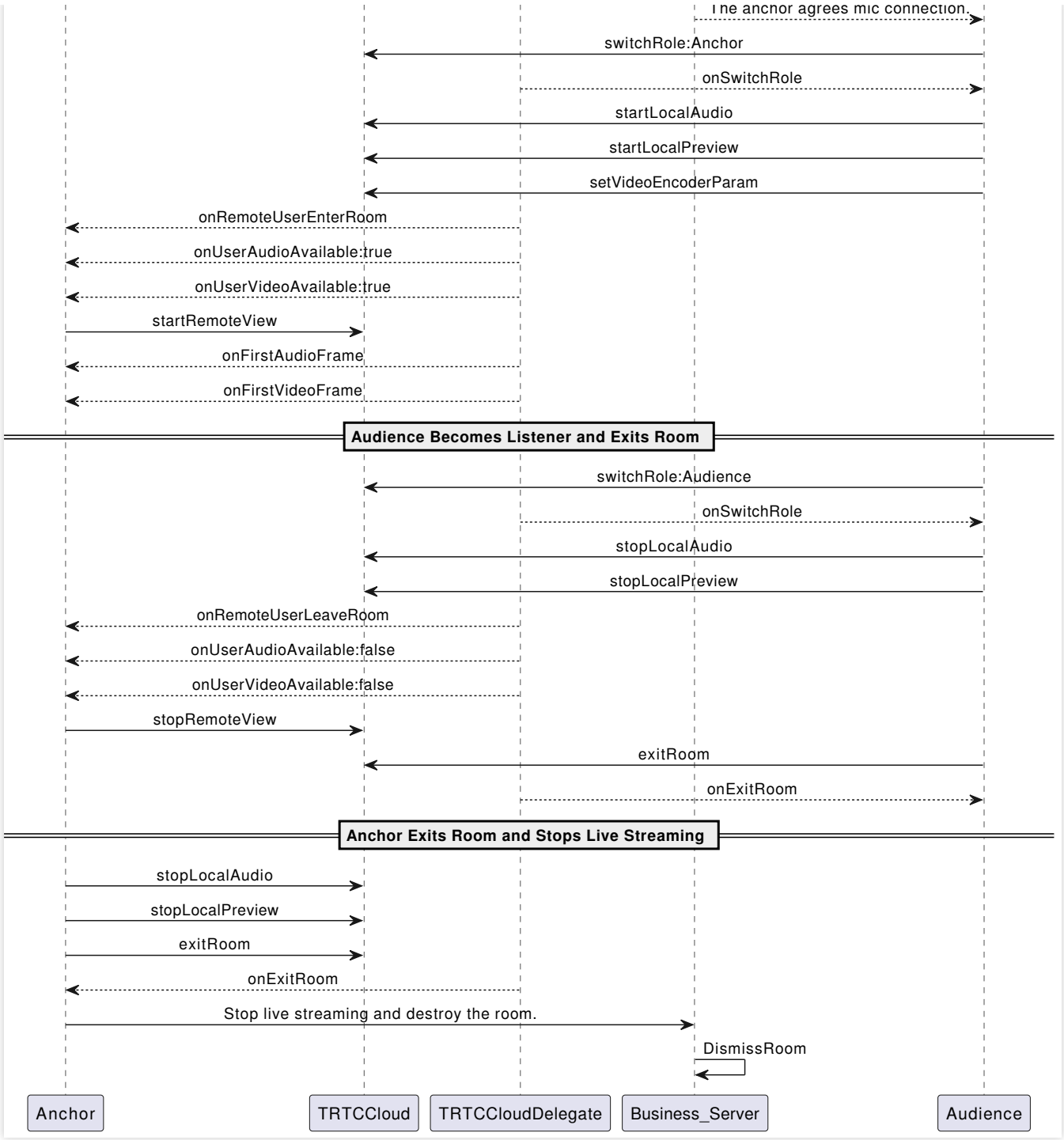
Before initializing the Tencent Effect SDK, resource copying and other preparatory work are needed. For detailed steps, see Tencent Effect SDK integration steps.

Integration Process

API sequence diagram.



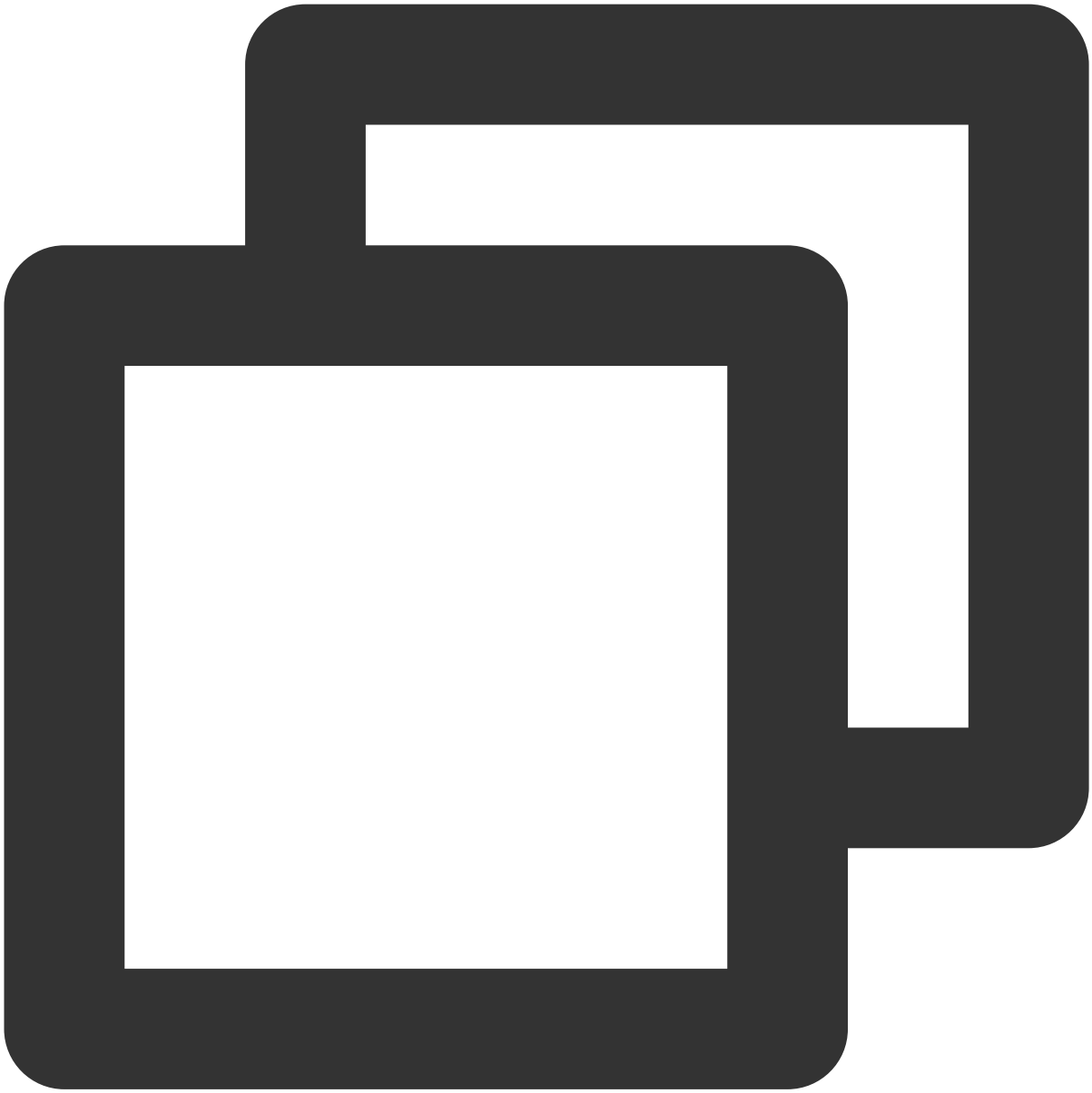
©2013-2022 Tencent Cloud. All rights reserved.



Step 1: The anchor enters the room to push streams.

1. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
@property (nonatomic, strong) UIView *anchorPreviewView;
@property (nonatomic, strong) TRTCCloud *trtcCloud;

- (void)setupTRTC {
    self.trtcCloud = [TRTCCloud sharedInstance];
    self.trtcCloud.delegate = self;
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    encParam.videoBitrate = 1300;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify using the front/rear camera for video capture.
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.anchorPreviewVie

    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

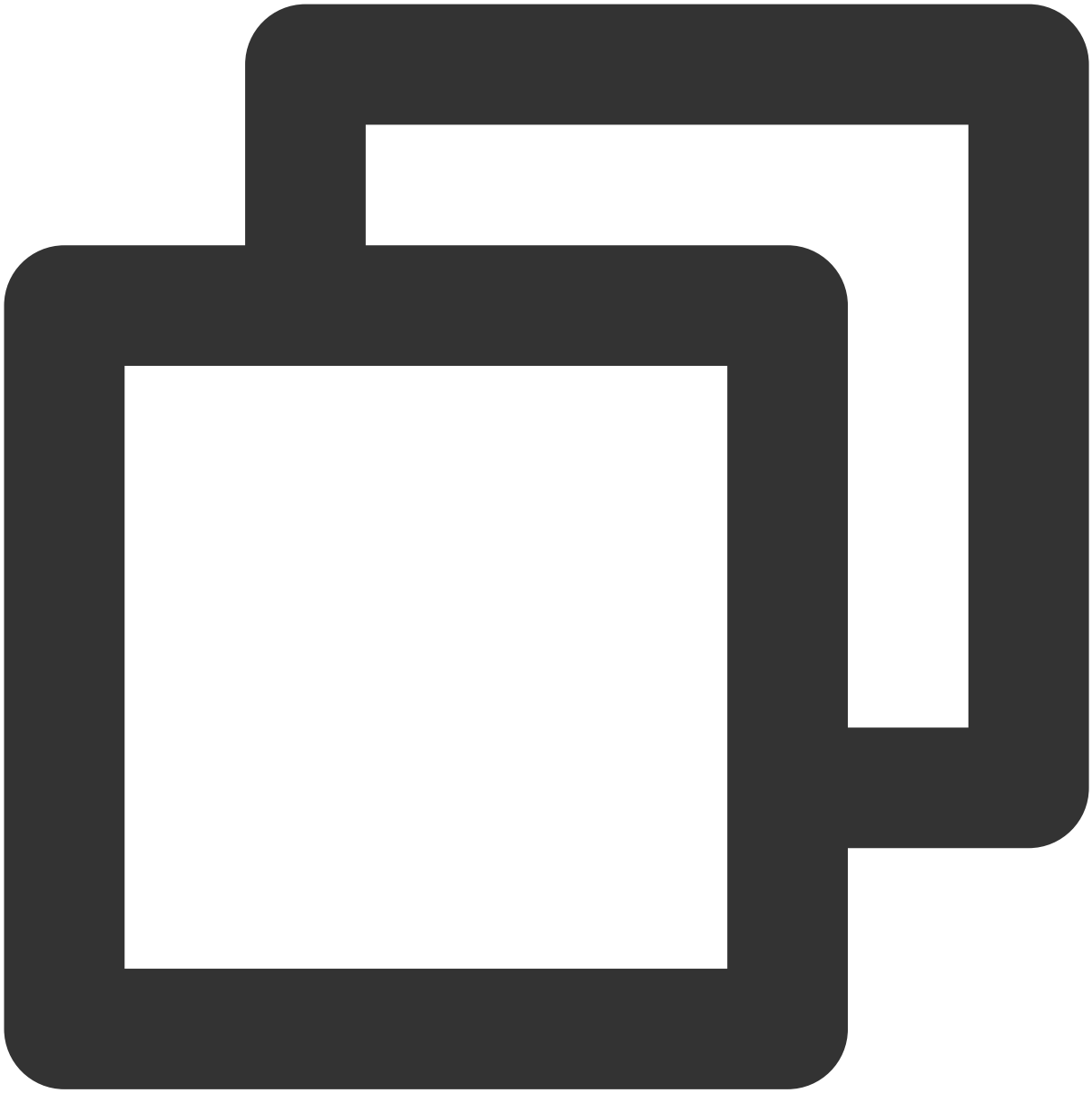
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom` . The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom` . The SDK will start the camera preview and audio capture and automatically start streaming.

2. The anchor sets rendering parameters for the local video, and the encoder output video mode (optional).

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering parameters for the local video.
    [self.trtcCloud setLocalRenderParams:params];
```

©2013-2022 Tencent Cloud. All rights reserved.

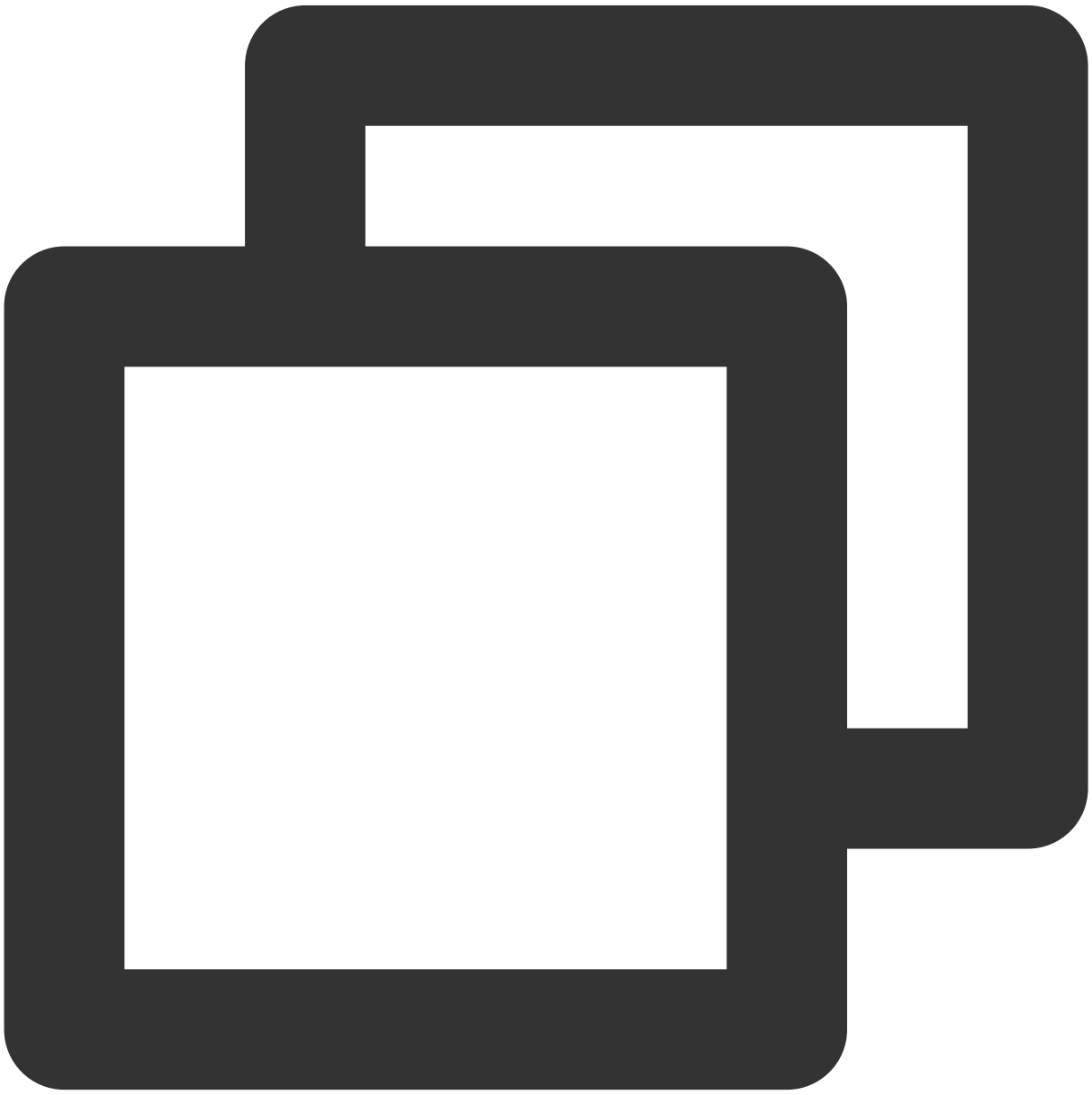
```
    // Set the video mirror mode for the encoder output.
    [self.trtcCloud setVideoEncoderMirror:YES];
    // Set the rotation of the video encoder output.
    [self.trtcCloud setVideoEncoderRotation:TRTCVideoRotation_0];
}
```

Note:

Setting local video rendering parameters only affects the rendering effect of the local video.

Setting encoder output mode affects the viewing effect for other users in the room (and the cloud recording files).

3. The anchor starts the live streaming, entering the room and start streaming.



©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)enterRoomByAnchorWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = @"userSig";
    // Replace with your SDKAppID.
    params.sdkAppId = 0;
    // Specify the anchor role.
    params.role = TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

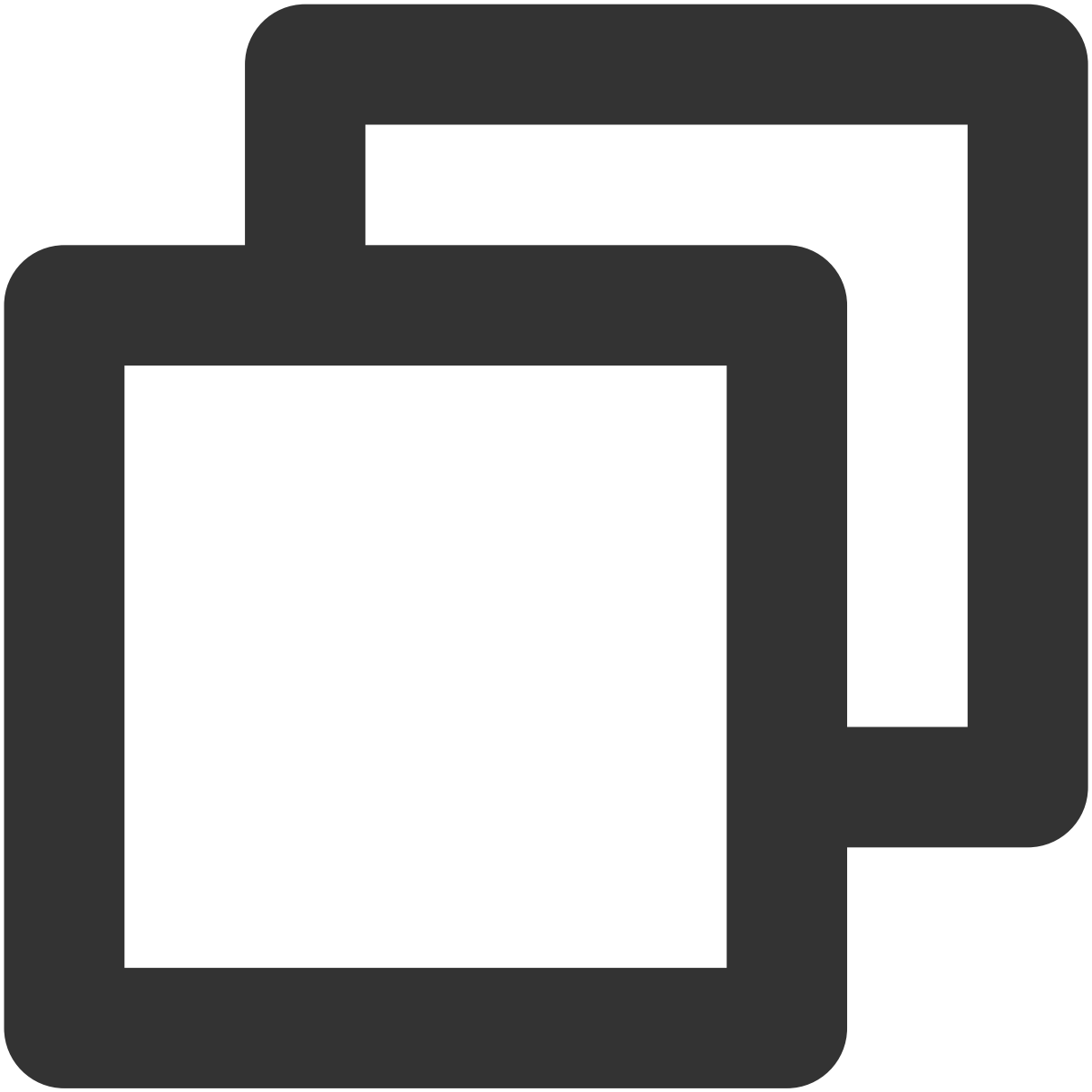
TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In live showroom scenarios, it is recommended to choose `TRTCAppSceneLIVE` as the room entry mode.

Step 2: The audience enters the room to pull streams.

1. Audience enters the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



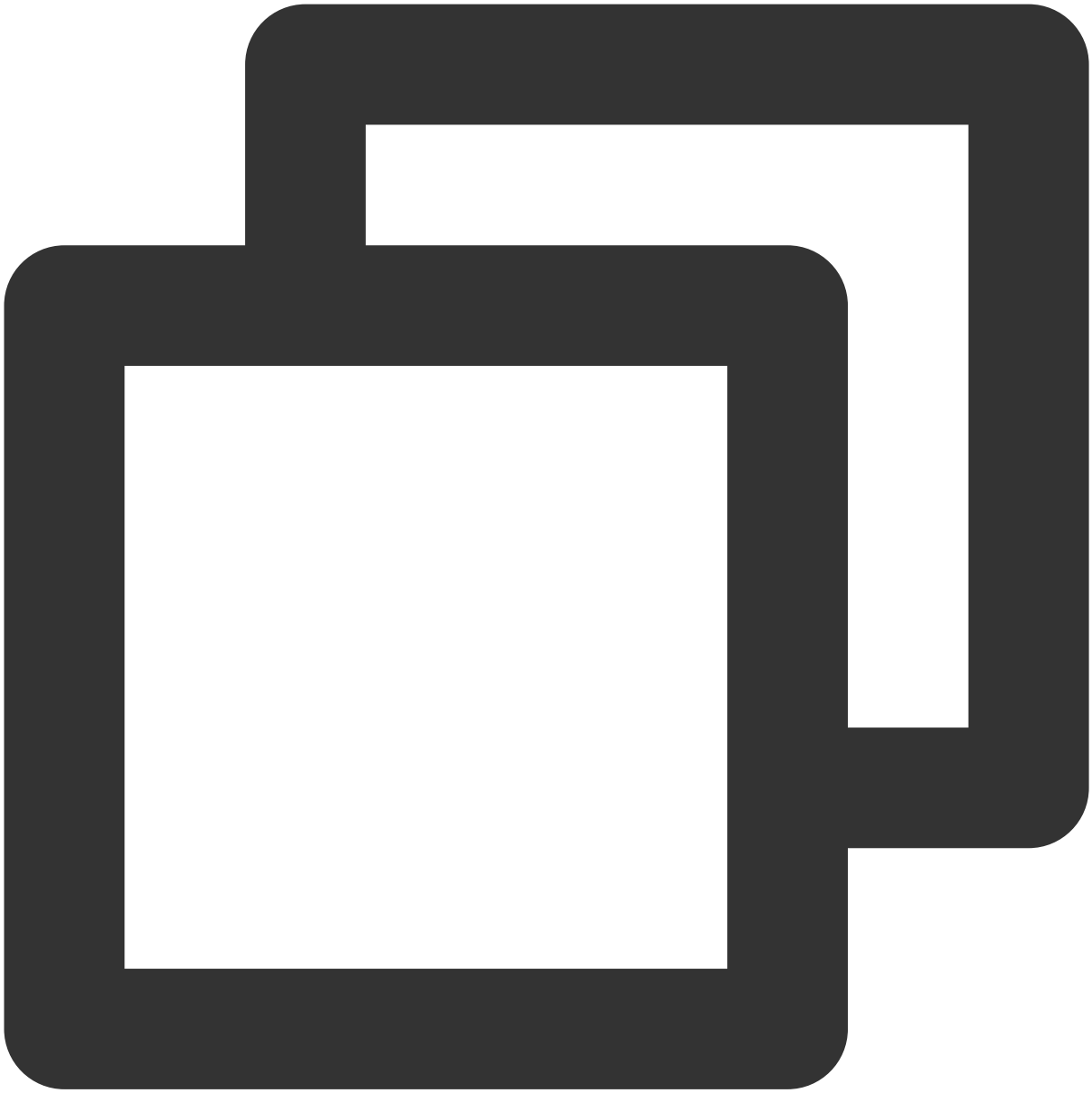
```
- (void)enterRoomByAudienceWithUserId:(NSString *)userId roomId:(NSString *)roomId
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = @"userSig";
    // Replace with your SDKAppID.
    params.sdkAppId = 0;
    // Specify the audience role.
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

2. Audience subscribes to the anchor's audio and video streams.

©2013-2022 Tencent Cloud. All rights reserved.

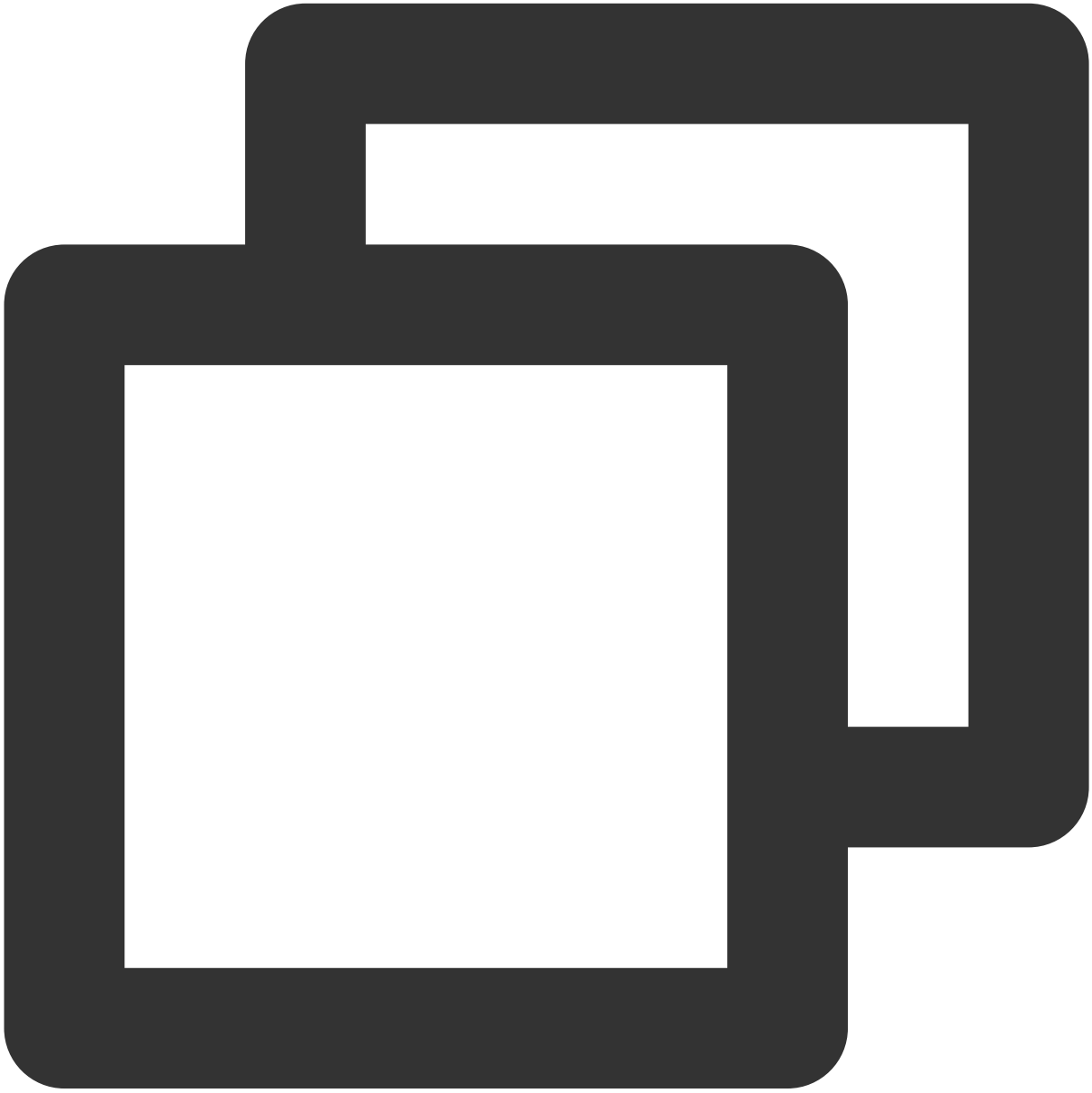


```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}
```

3. Audience sets the rendering mode for the remote video (optional).



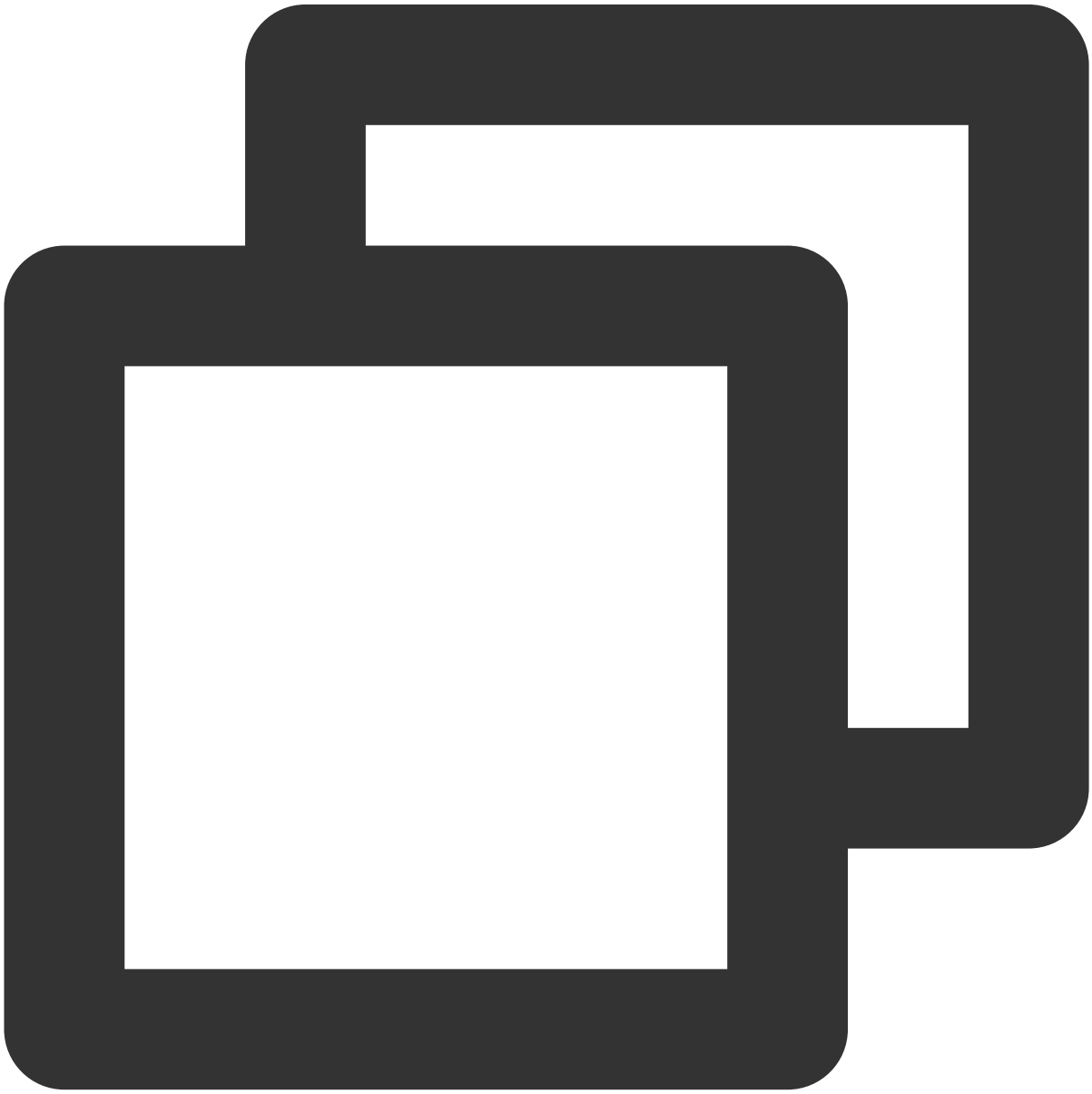
```
- (void)setupRemoteRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering mode for the remote video.
    [self.trtcCloud setRemoteRenderParams:@"userId" streamType:TRTCVideoStreamTypeB
}
```

Step 3: The audience interacts via mic.

1. The audience is switched to the anchor role.

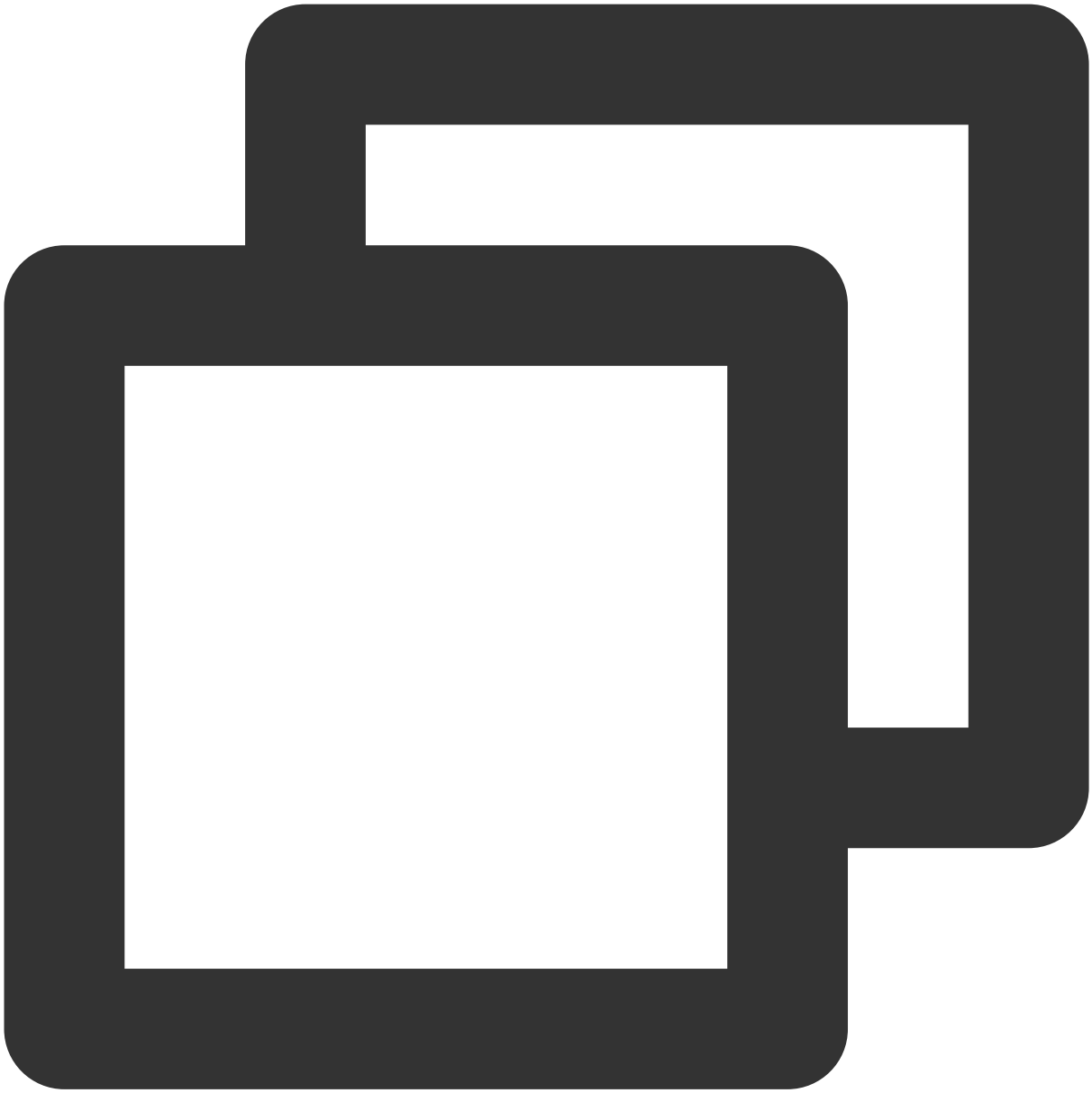


©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)switchToAnchor {
    // Switched to the anchor role.
    [self.trtcCloud switchRole:TRTCRoleAnchor];
}

// Event callback for switching the role.
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Role switched successfully.
    }
}
```

2. Audience start local audio and video capture and streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupTRTC {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify using the front/rear camera for video capture.
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.audiencePreviewV
```

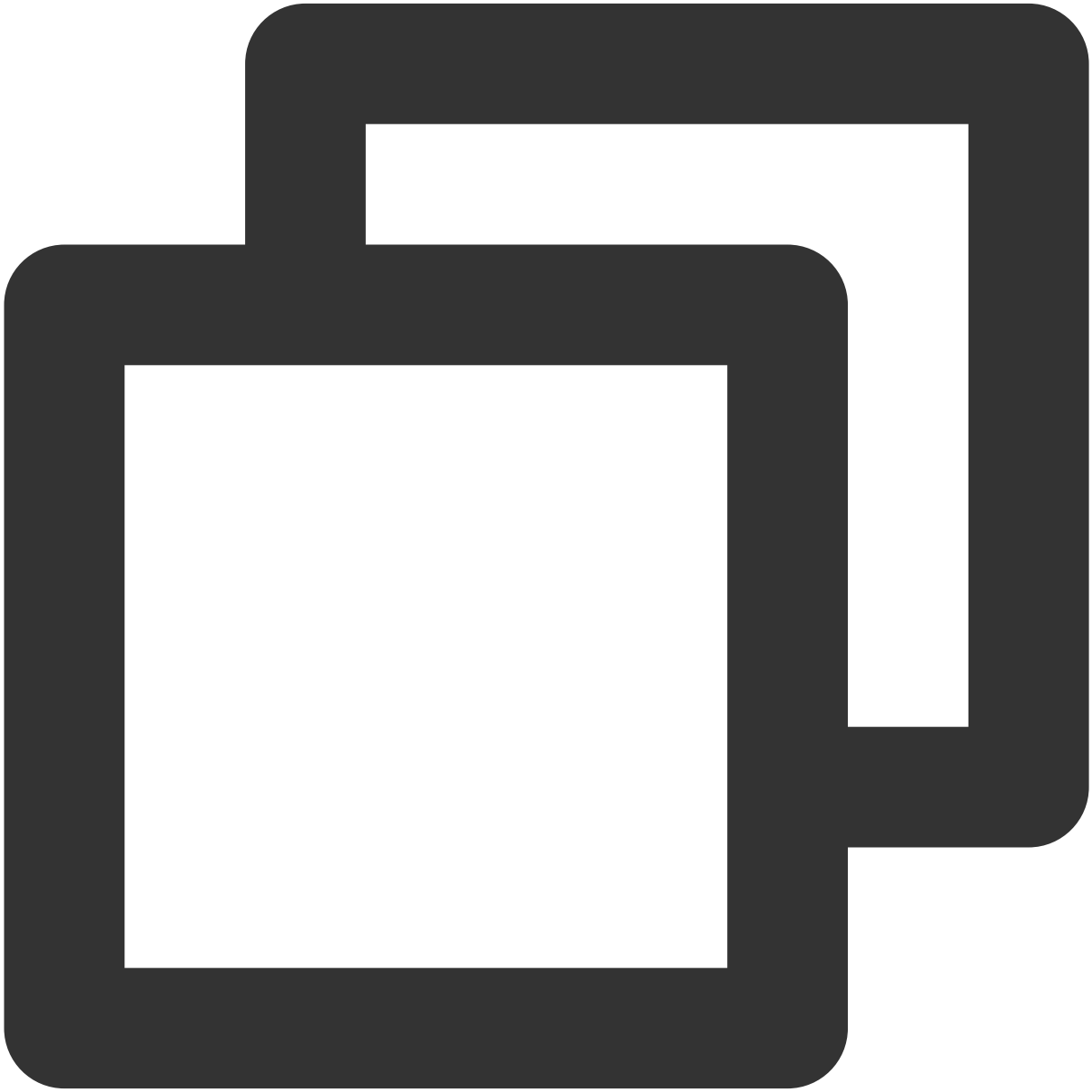
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

3. The audience leaves the seat and stops streaming.



```
- (void)switchToAudience {
    // Switched to the audience role.
```

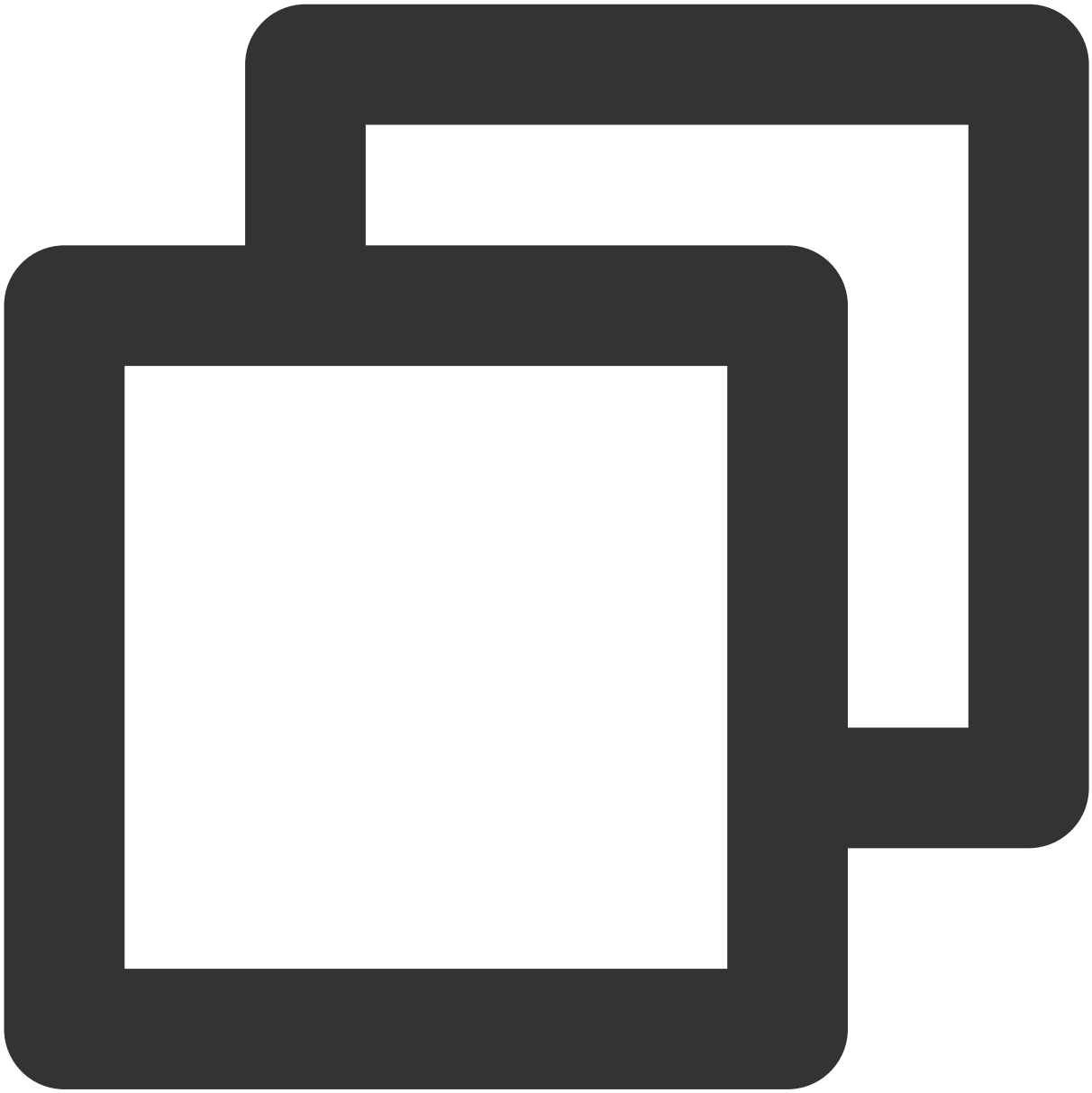
©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud switchRole:TRTCRoleAudience];
}

// Event callback for switching the role.
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Stop camera capture and streaming.
        [self.trtcCloud stopLocalPreview];
        // Stop microphone capture and streaming.
        [self.trtcCloud stopLocalAudio];
    }
}
```

Step 4: Exiting and dissolving the room.

1. Exit the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)exitRoom {
    [self.trtcCloud stopLocalAudio];
    [self.trtcCloud stopLocalPreview];
    [self.trtcCloud exitRoom];
}

// Event callback for exiting the room.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room.");
    } else if (reason == 1) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        NSLog(@"Removed from the current room by the server.");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved.");
    }
}
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you want to call `enterRoom` again or switch to another audio/video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter exceptions such as the camera or microphone being forcefully occupied.

2. Dissolve the room.

Server Dissolvement: TRTC provides the Server dissolves the room API `DismissRoom` (differentiating between numeric room ID and string room ID). You can call this API to remove all users from the room and dissolve the room.

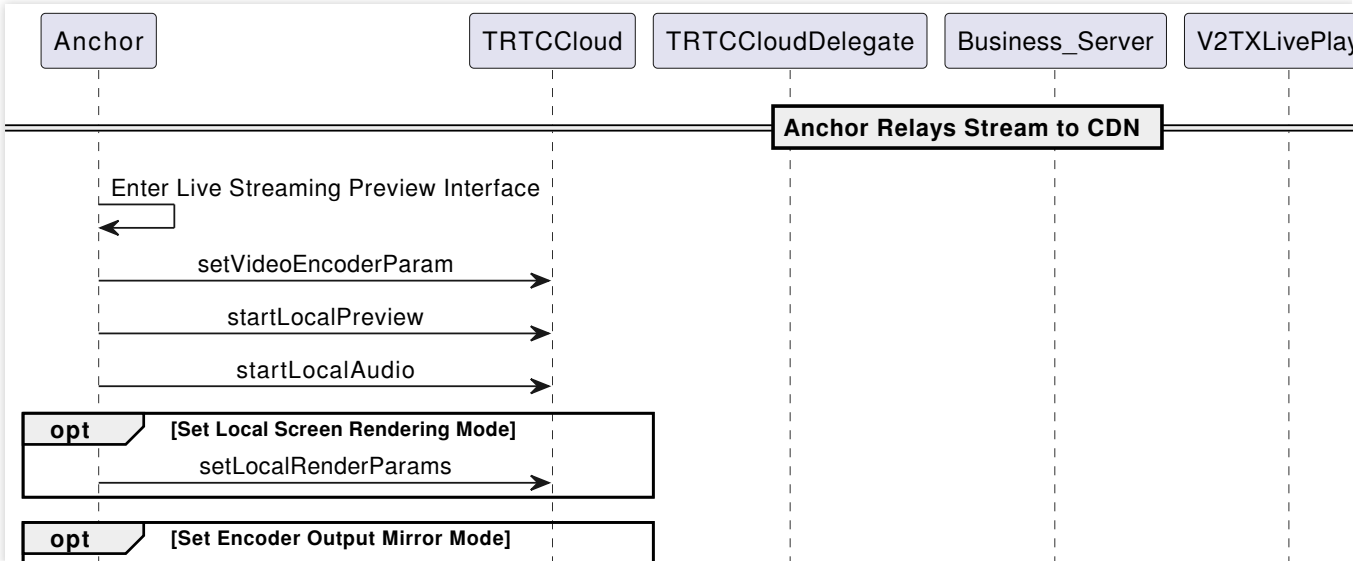
Client Dissolvement: Through the `exitRoom` API of each client, all the anchors and audiences in the room can be completed of room exit. After room exit, according to TRTC room lifecycle rules, the room will automatically be dissolved. For details, see Exit Room.

Warning:

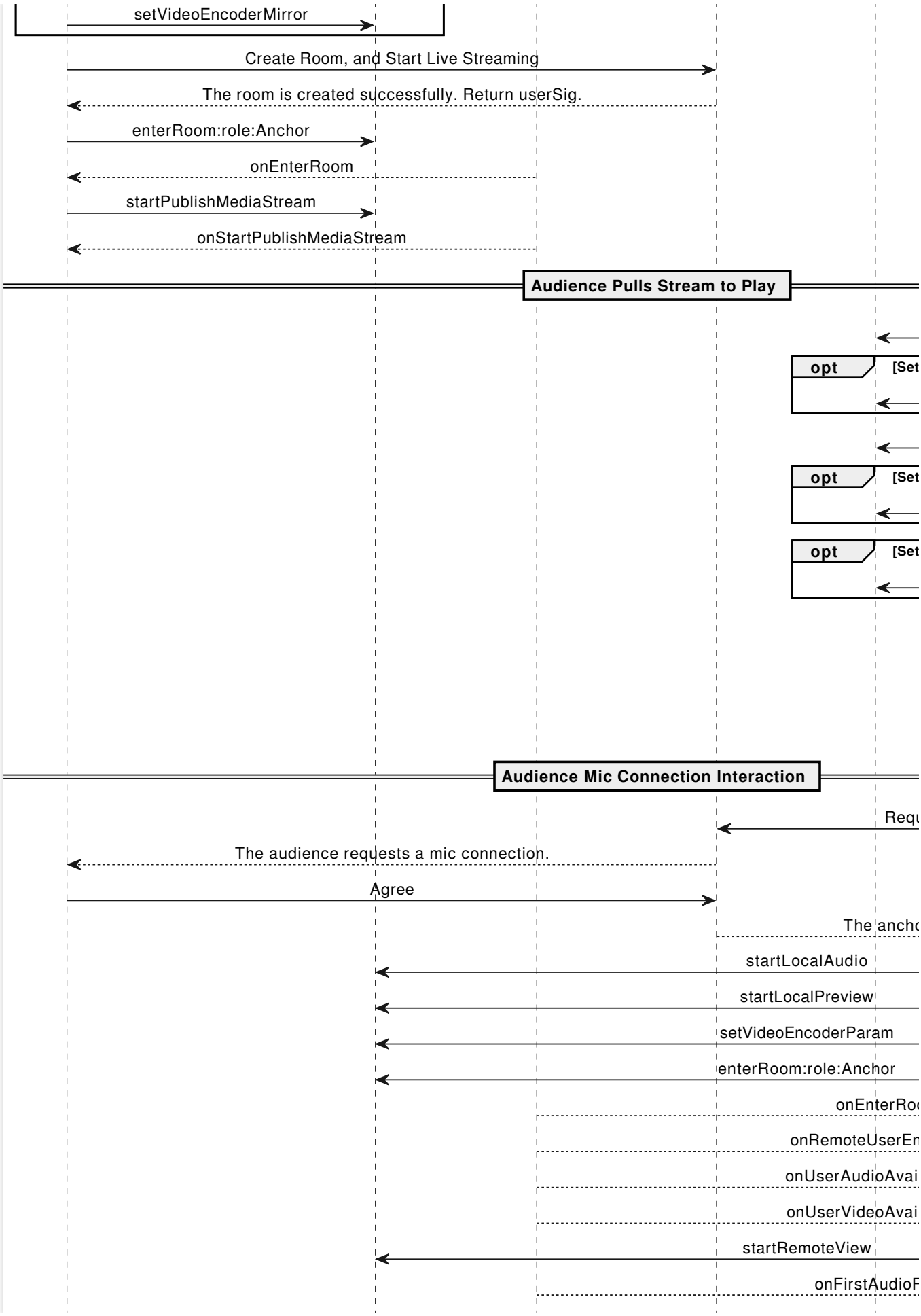
It is recommended that after the end of live streaming, you call the room dissolvement API on the server to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

Alternative solutions

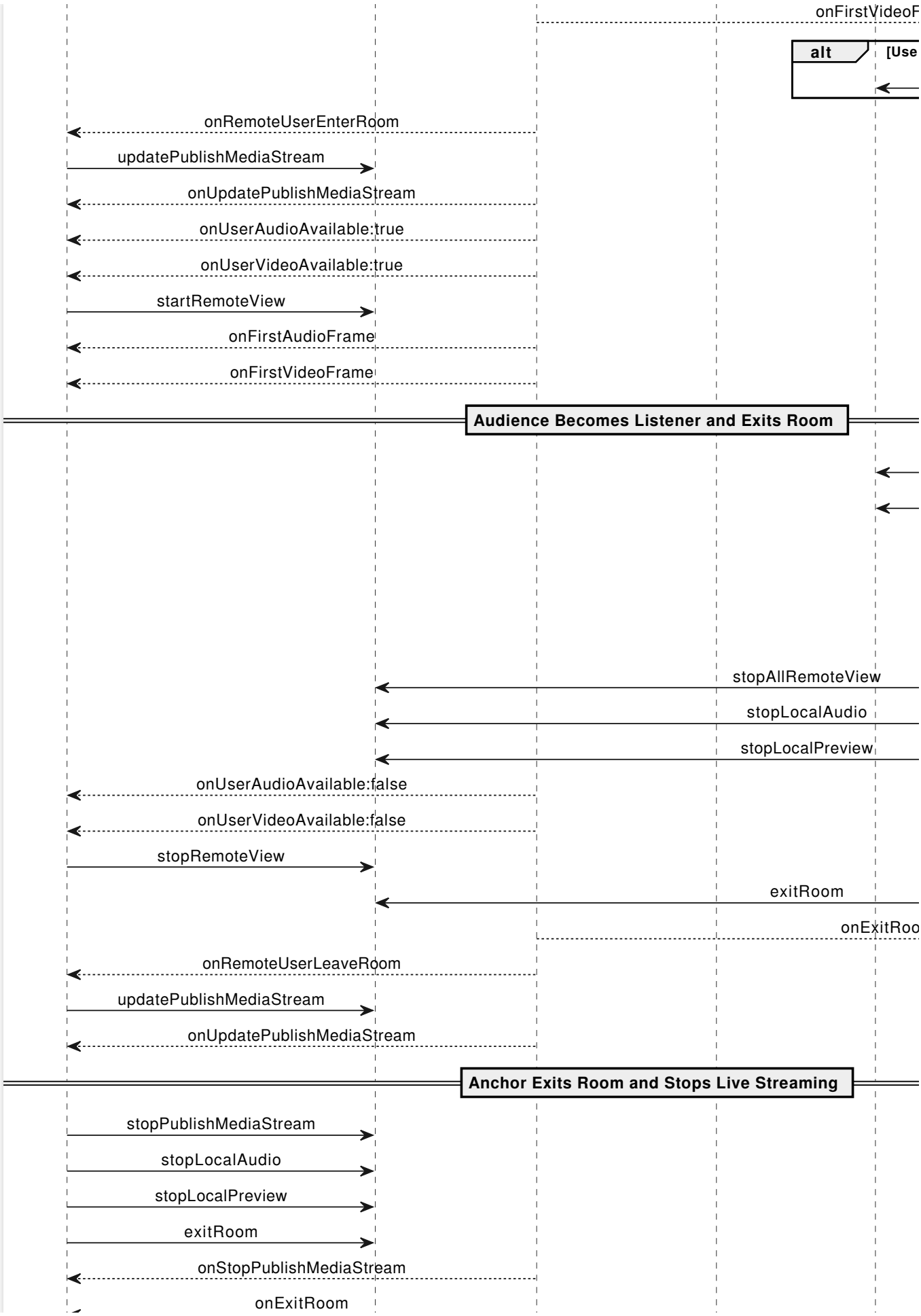
API sequence diagram.



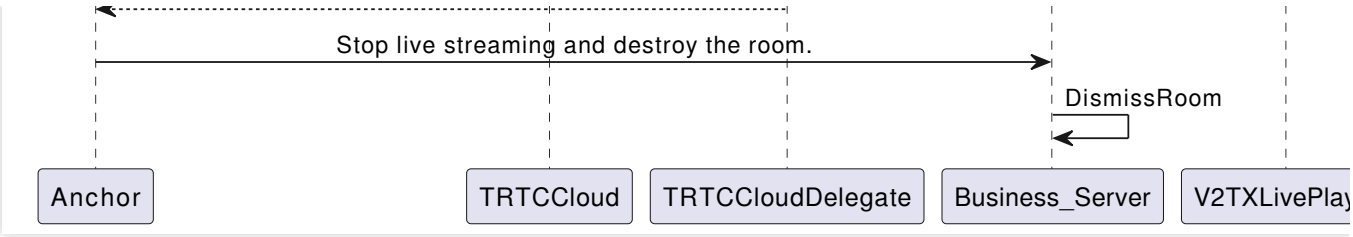
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

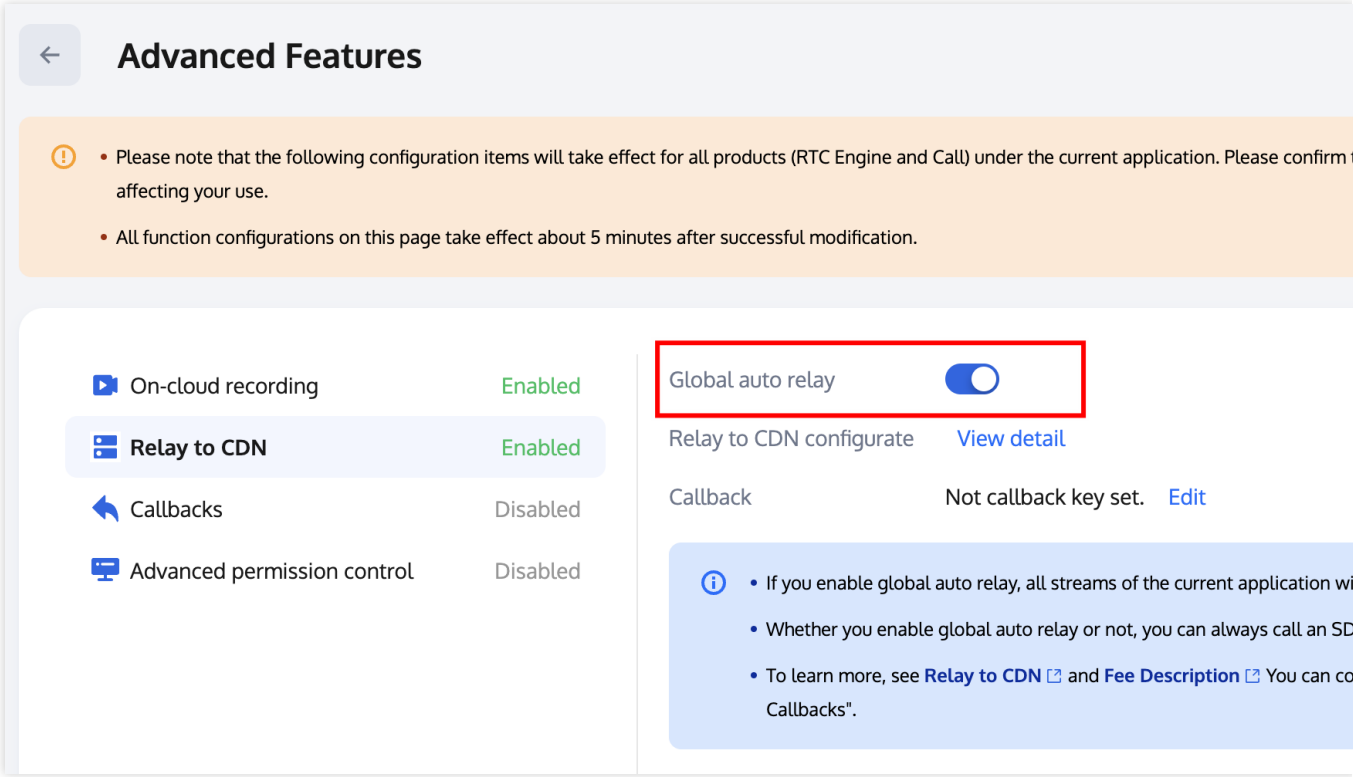


Step 1: The anchor relays the streams to CDN.

1. Related configurations for relaying to live streaming CDN.

Global automatic relayed push

If you need to automatically relay all anchors' audio and video streams in the room to live streaming CDN, you need to enable **Relay to CDN** in the TRTC console **Advanced Features** page.

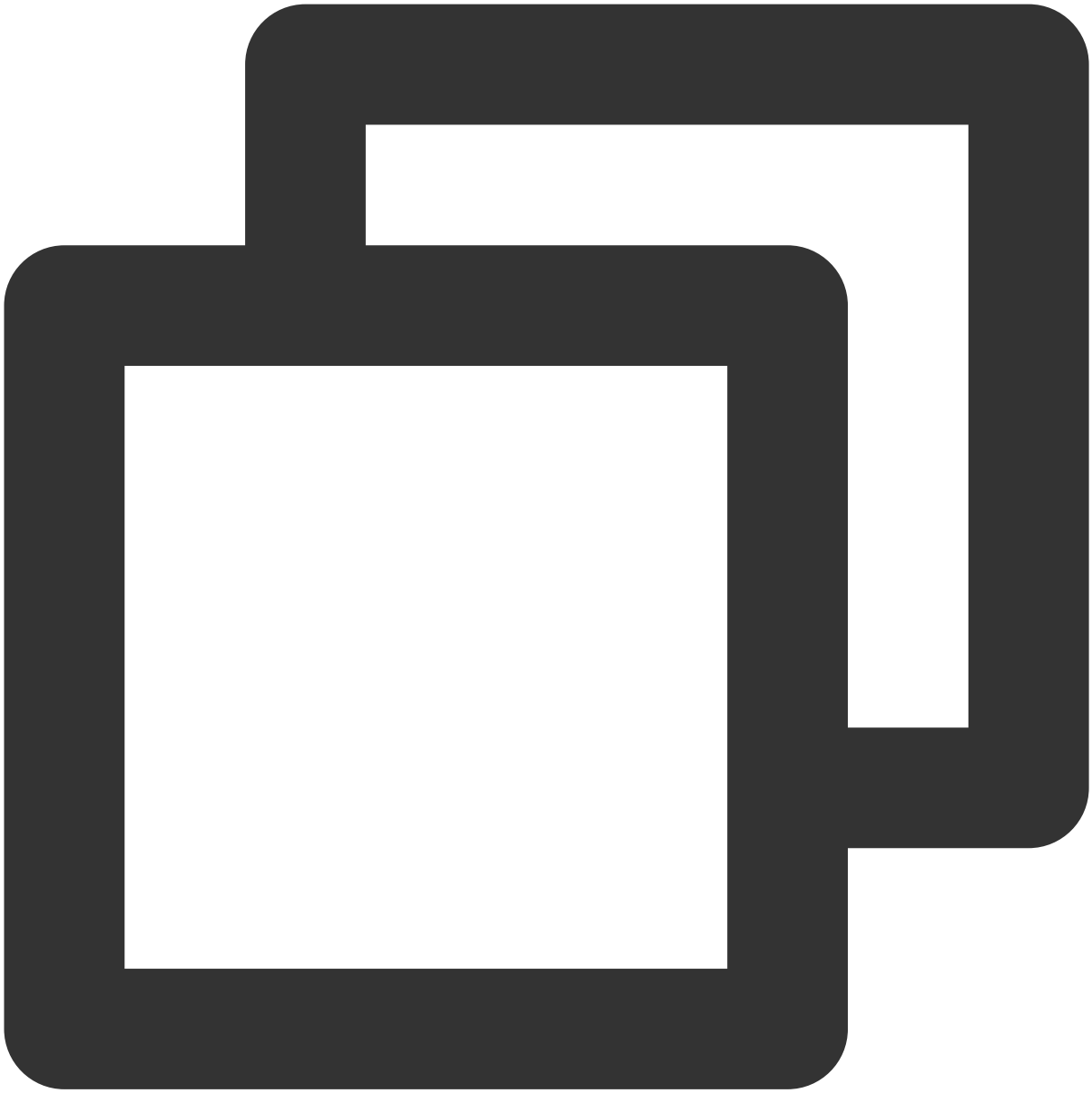


Relayed push of the specified streams

If you need to manually specify the audio and video streams to be published to live streaming CDN, or publish the mixed audio and video streams to live streaming CDN, you can do so by calling the startPublishMediaStream API. In this case, you do not need to activate global automatically relaying to CDN in the console. For detailed introduction, see Publish Audio and Video Streams to Live Streaming CDN.

2. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
@property (nonatomic, strong) UIView *anchorPreviewView;


- (void)setupTRTC {
    self.trtcCloud = [TRTCCloud sharedInstance];
    self.trtcCloud.delegate = self;
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    encParam.videoBitrate = 1300;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify using the front/rear camera for video capture.
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.anchorPreviewVie

    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

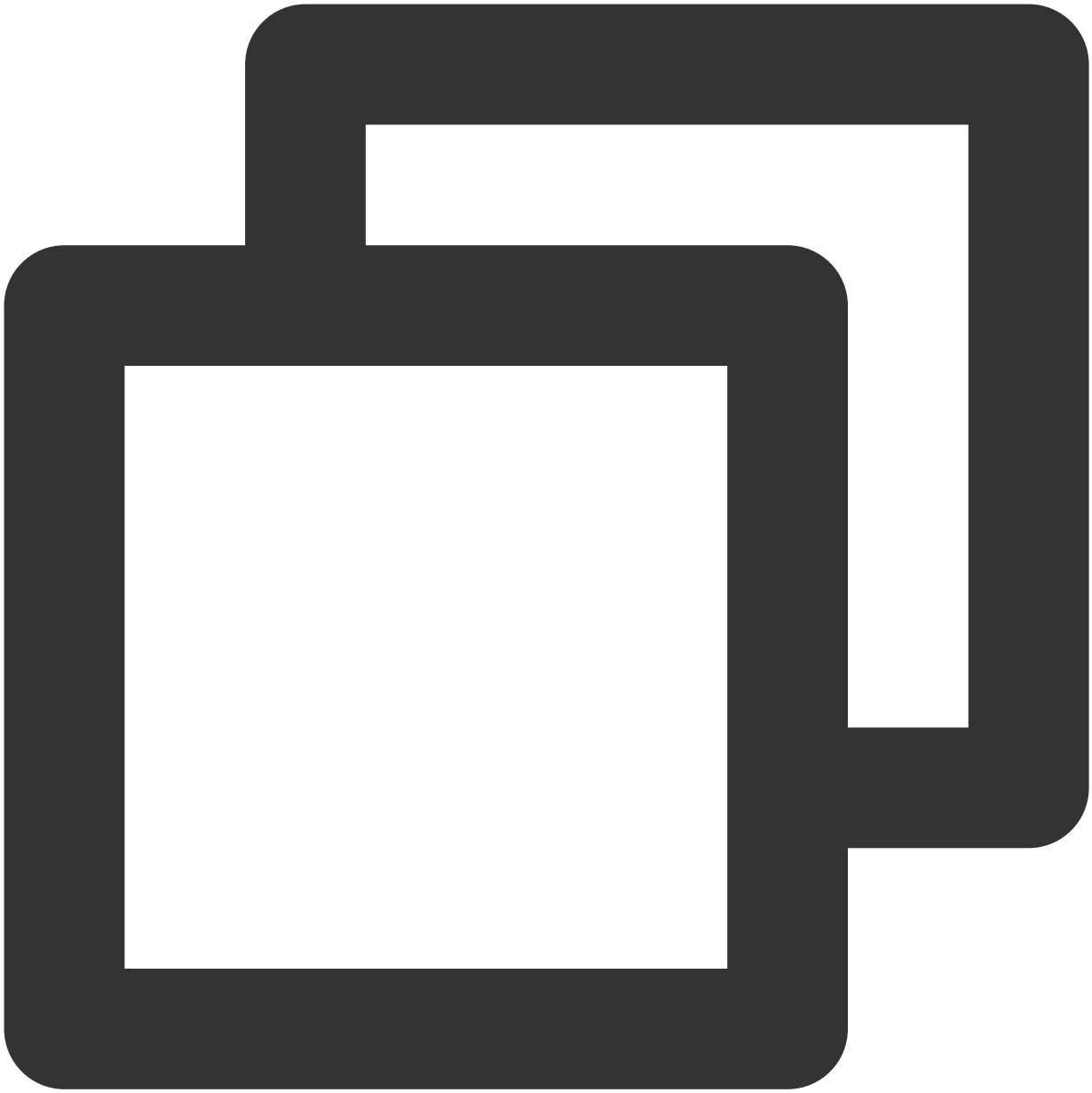
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

3. The anchor sets rendering parameters for the local screen, and the encoder output video mode.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering parameters for the local video.
    [self.trtcCloud setLocalRenderParams:params];
```

©2013-2022 Tencent Cloud. All rights reserved.

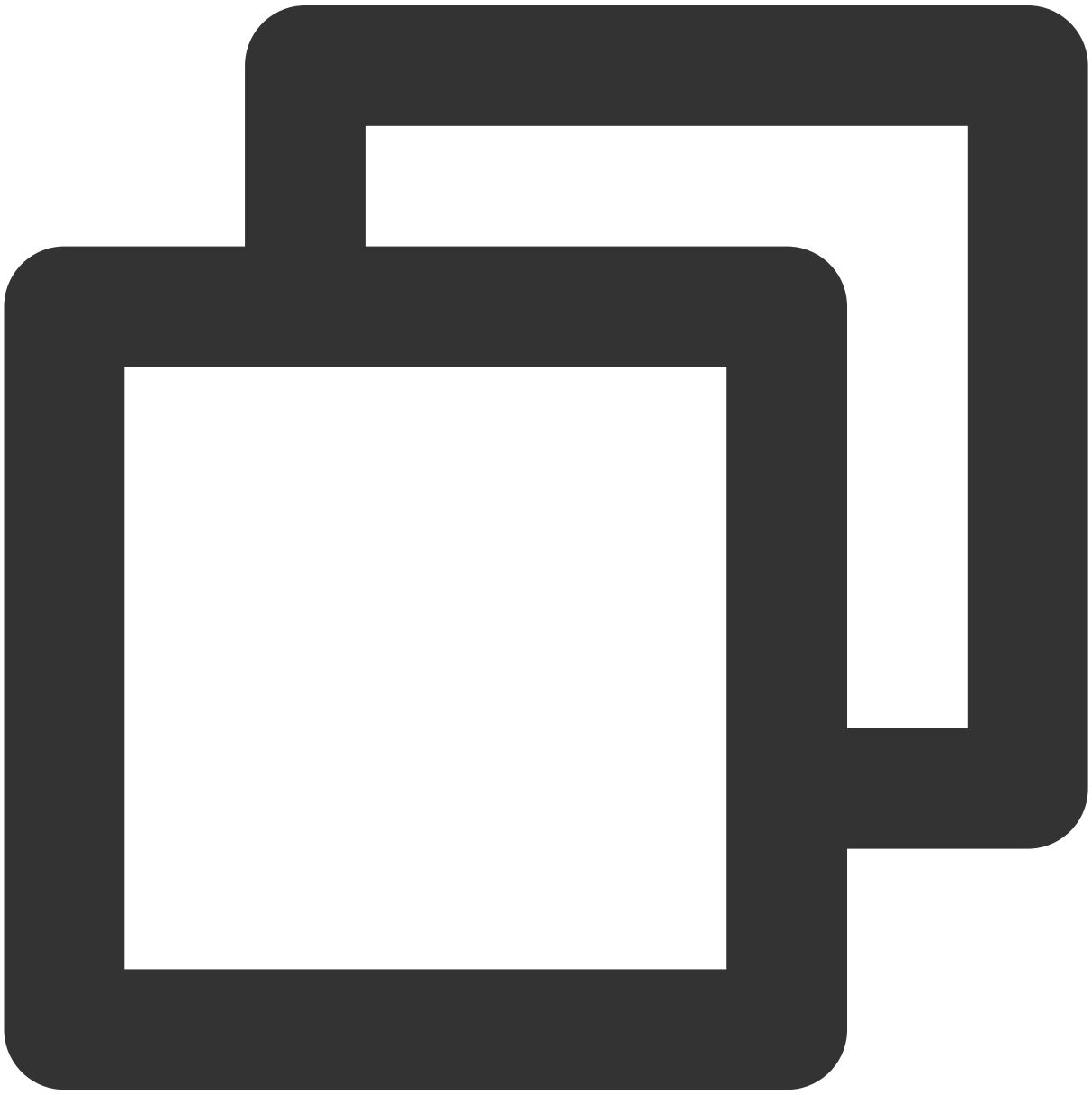
```
    // Set the video mirror mode for the encoder output.
    [self.trtcCloud setVideoEncoderMirror:YES];
    // Set the rotation of the video encoder output.
    [self.trtcCloud setVideoEncoderRotation:TRTCVideoRotation_0];
}
```

Note:

Setting local video rendering parameters only affects the rendering effect of the local video.

Setting encoder output mode affects the viewing effect for other users in the room (and the cloud recording files).

4. The anchor starts the live streaming, entering the room and start streaming.



©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)enterRoomByAnchorWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = @"userSig";
    // Replace with your SDKAppID.
    params.sdkAppId = 0;
    // Specify the anchor role.
    params.role = TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

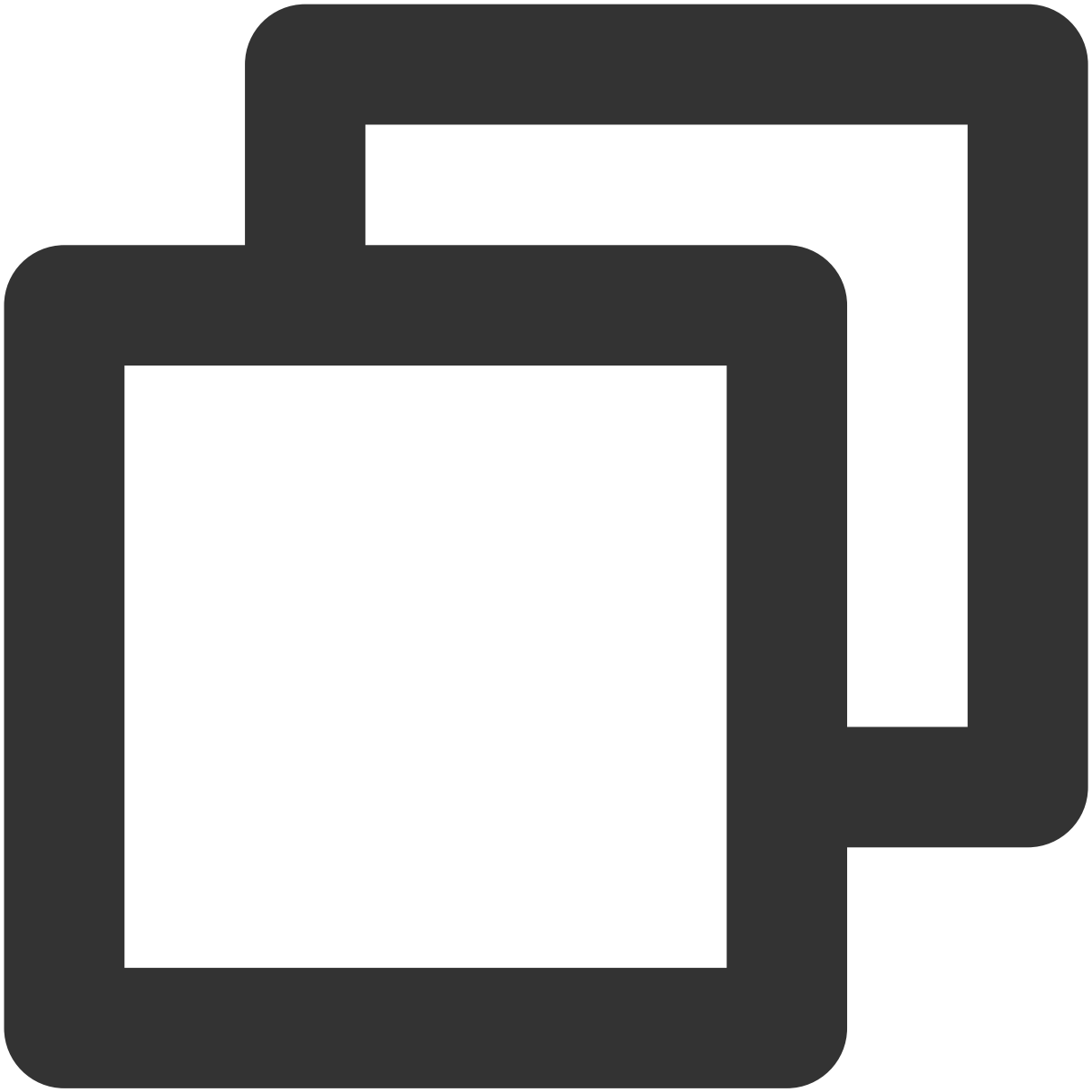
TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In live showroom scenarios, it is recommended to choose `TRTCAppSceneLIVE` as the room entry mode.

5. The anchor relays the audio and video streams to the live streaming CDN.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)startPublishMediaToCDN:(NSString *)streamName {
    NSDate *date = [NSDate dateWithTimeIntervalSinceNow:0];
    // Set the expiration time for the push URLs.
    NSTimeInterval time = [date timeIntervalSince1970] + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    NSString *secretParam = [self getSafeUrl:LIVE_URL_KEY streamName:streamName tim

    // The target URLs for media stream publication.
    TRTCPublishTarget* target = [[TRTCPublishTarget alloc] init];
    // The target URLs are set for relaying to CDN.
    target.mode = TRTCPublishBigStreamToCdn;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    TRTCPublishCdnUrl* cdnUrl = [[TRTCPublishCdnUrl alloc] init];
    // Construct push URLs (in RTMP format) to the live streaming service provider.
    cdnUrl.rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@?%@", PUSH_DOMAI
    // True means Tencent CSS push URLs, and false means third-party services.
    cdnUrl.isInternalLine = YES;
    NSMutableArray* cdnUrlList = [NSMutableArray array];
    // Multiple CDN push URLs can be added.
    [cdnUrlList addObject:cdnUrl];
    target.cdnUrlList = cdnUrlList;

    // Set media stream encoding output parameters (can be defined according to bus
    TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
    encoderParam.audioEncodedSampleRate = 48000;
    encoderParam.audioEncodedChannelNum = 1;
    encoderParam.audioEncodedKbps = 50;
    encoderParam.audioEncodedCodecType = 0;
    encoderParam.videoEncodedWidth = 540;
    encoderParam.videoEncodedHeight = 960;
    encoderParam.videoEncodedFPS = 15;
    encoderParam.videoEncodedGOP = 2;
    encoderParam.videoEncodedKbps = 1300;

    // Start publishing media streams.
    [self.trtcCloud startPublishMediaStream:target encoderParam:encoderParam mixing
}
```

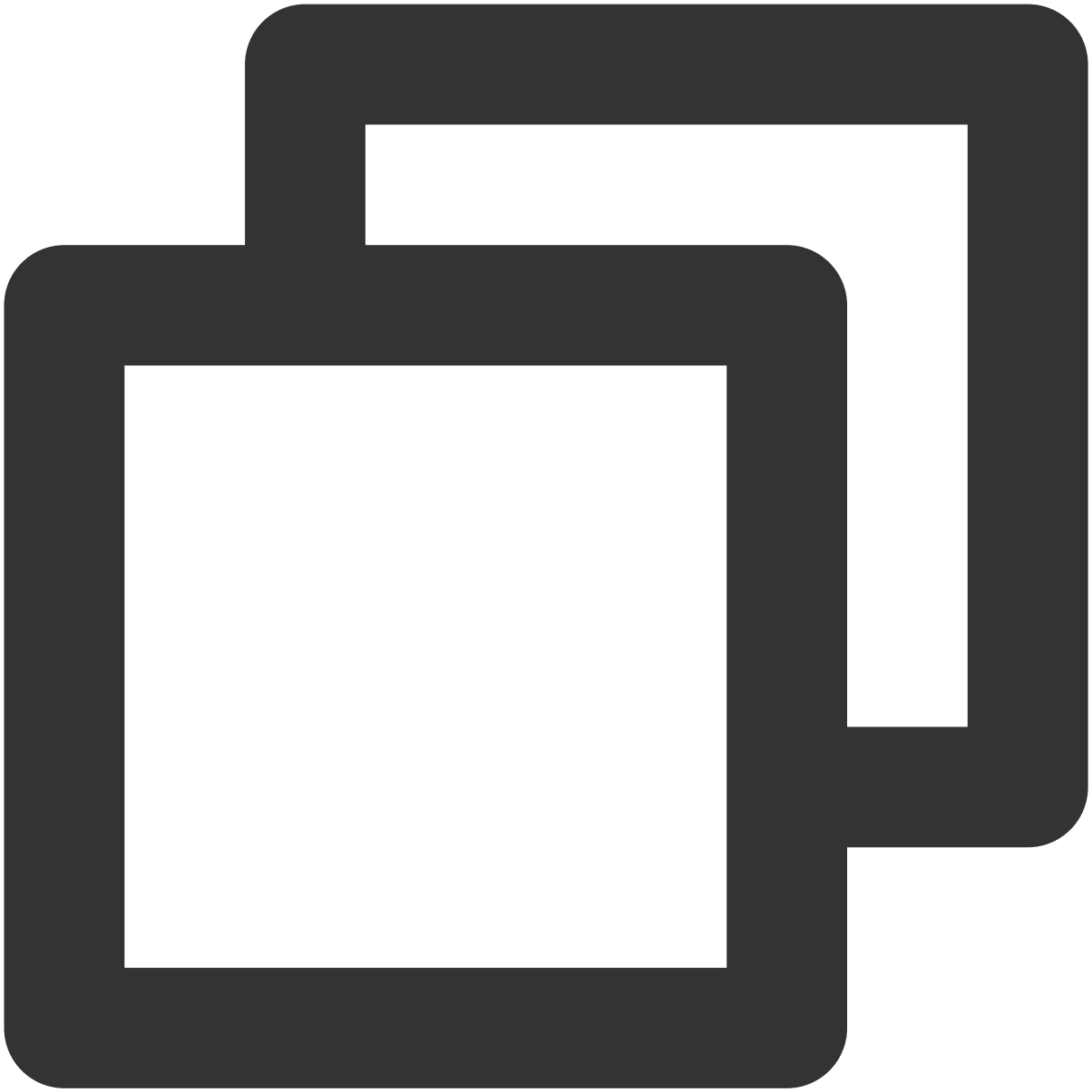
Note:

During single-anchor live streaming, only initiate the relayed push task. When there is an audience co-broadcasting or anchor PK, update this task to a mixed-stream transcoding task.

Information of push authentication KEY LIVE_URL_KEY and push domain name PUSH_DOMAIN are required to obtain in the CSS console Domain Management page.

After the media stream is published, SDK will provide the backend-initiated task identifier (taskId) through the callback onStartPublishMediaStream.

©2013-2022 Tencent Cloud. All rights reserved.

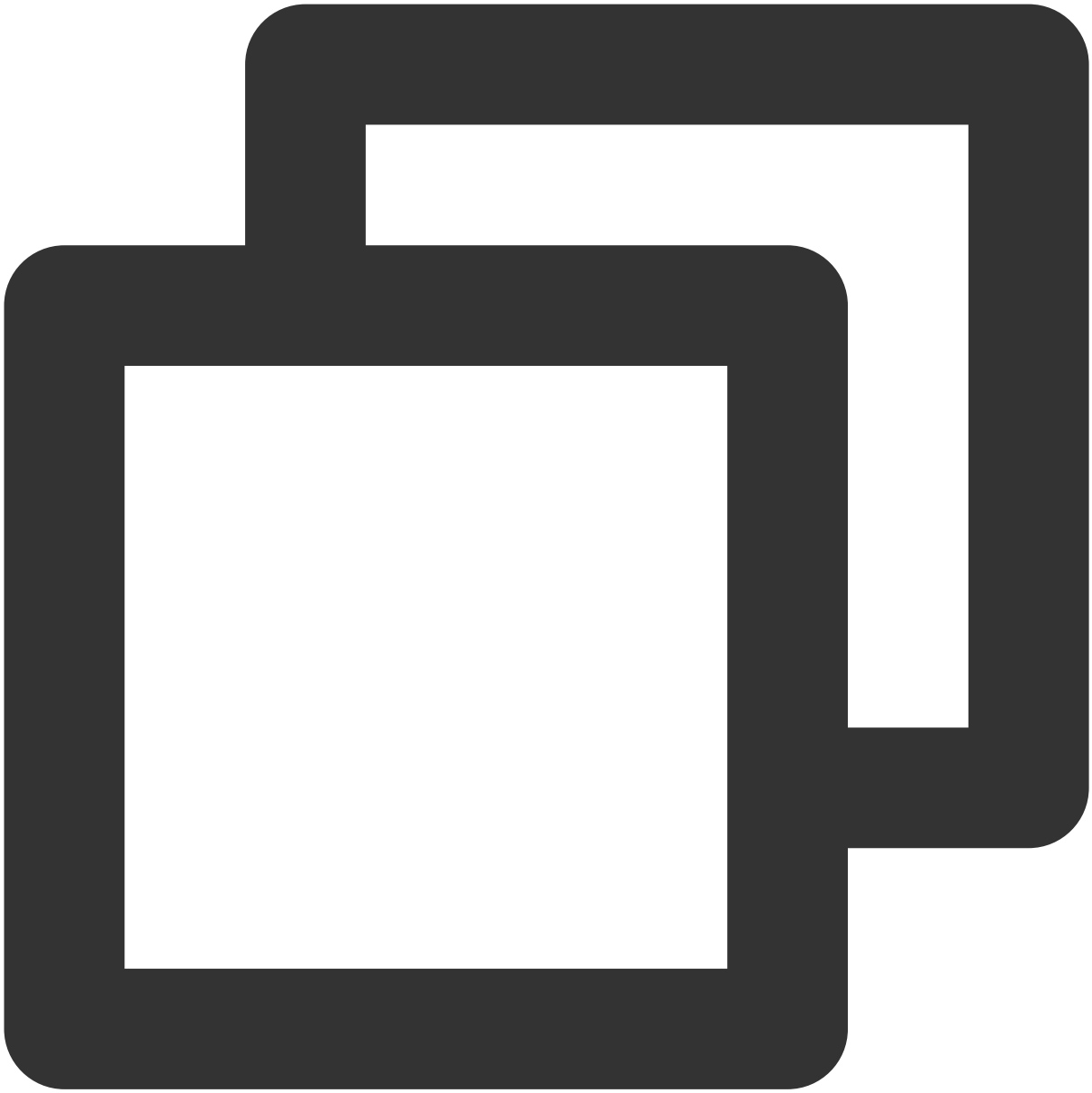


```
- (void)onStartPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStri
    // taskId: When the request is successful, TRTC backend will provide the taskId
    // code: Callback result. 0 means success and other values mean failure.
}
```

Step 2: The audience pulls streams for playback.

CDN audiences do not need to enter the TRTC room. They can directly pull the anchor's CDN stream for playback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Initialize the player.
self.livePlayer = [[V2TXLivePlayer alloc] init];
// Set the player callback listener.
[self.livePlayer setObserver:self];
// Set the video rendering control for the player.
[self.livePlayer setRenderView:self.remoteView];
// Set delay management mode (optional).
[self.livePlayer setCacheParams:1.f maxTime:5.f]; // Auto mode
[self.livePlayer setCacheParams:1.f maxTime:1.f]; // Speed mode
[self.livePlayer setCacheParams:5.f maxTime:5.f]; // Smooth mode
```

©2013-2022 Tencent Cloud. All rights reserved.

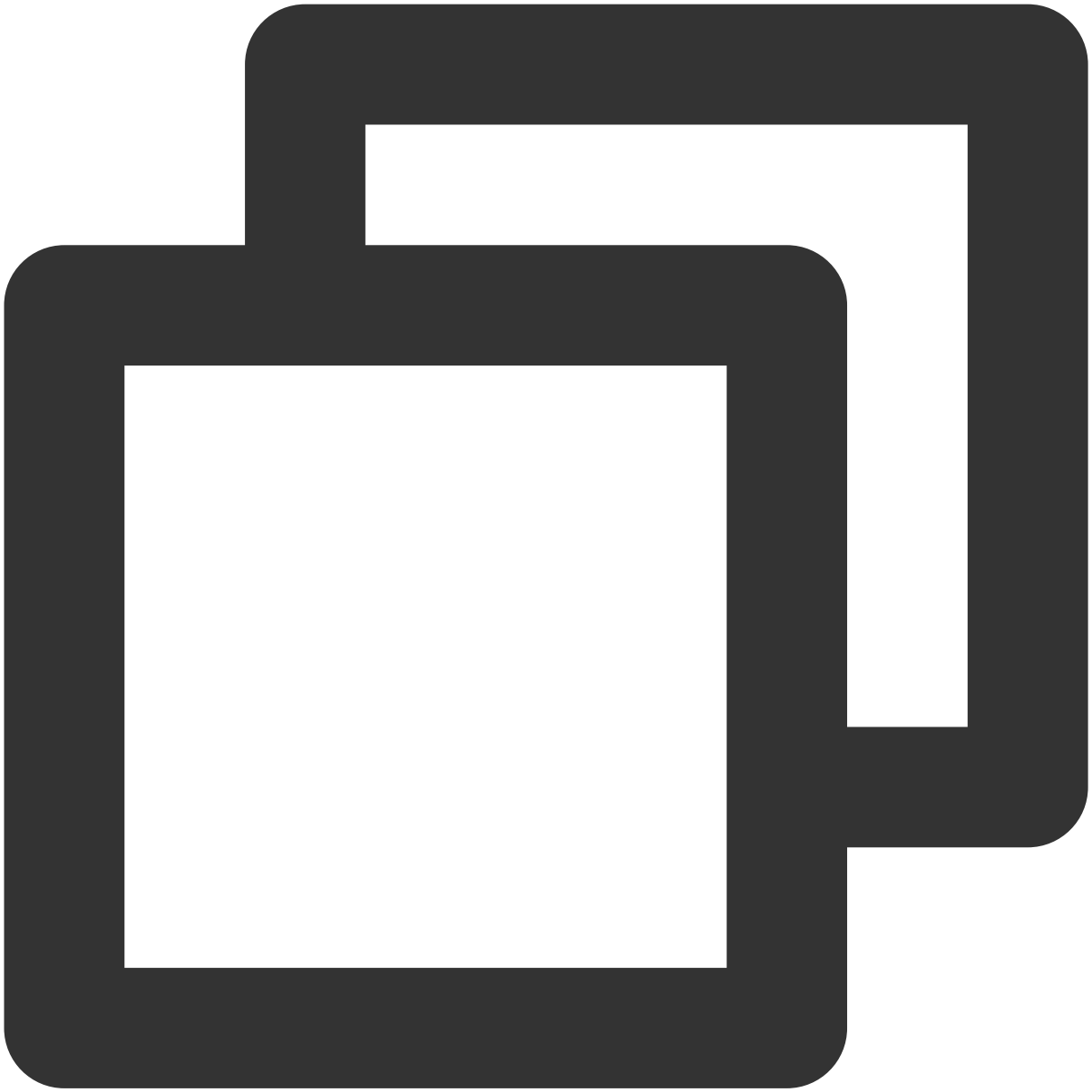
```
// Concatenate the pull URLs for playback.
NSString *flvUrl = [NSString stringWithFormat:@"http://%@/live/%@.flv", PLAY_DOMAIN
NSString *hlsUrl = [NSString stringWithFormat:@"http://%@/live/%@.m3u8", PLAY_DOMAI
NSString *rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@", PLAY_DOMAIN, s
NSString *webrtcUrl = [NSString stringWithFormat:@"webrtc://%@/live/%@", PLAY_DOMAI

// Start playing.
[self.livePlayer startLivePlay:flvUrl];

// Custom set fill mode (optional).
[self.livePlayer setRenderFillMode:V2TXLiveFillModeFit];
// Custom video rendering direction (optional).
[self.livePlayer setRenderRotation:V2TXLiveRotation0];
```

Note:

The playback domain name PLAY_DOMAIN requires you to Add Your Own Domain in the CSS console for live streaming playback. You also should configure domain CNAME.

The live streaming feature requires setting the License before success in playback. Otherwise, playback will fail (black screen). It needs to be set globally only once. If you have not obtained the License, you can freely apply for a Trial Version License for normal playback. The Official Version License requires purchase.

©2013-2022 Tencent Cloud. All rights reserved.

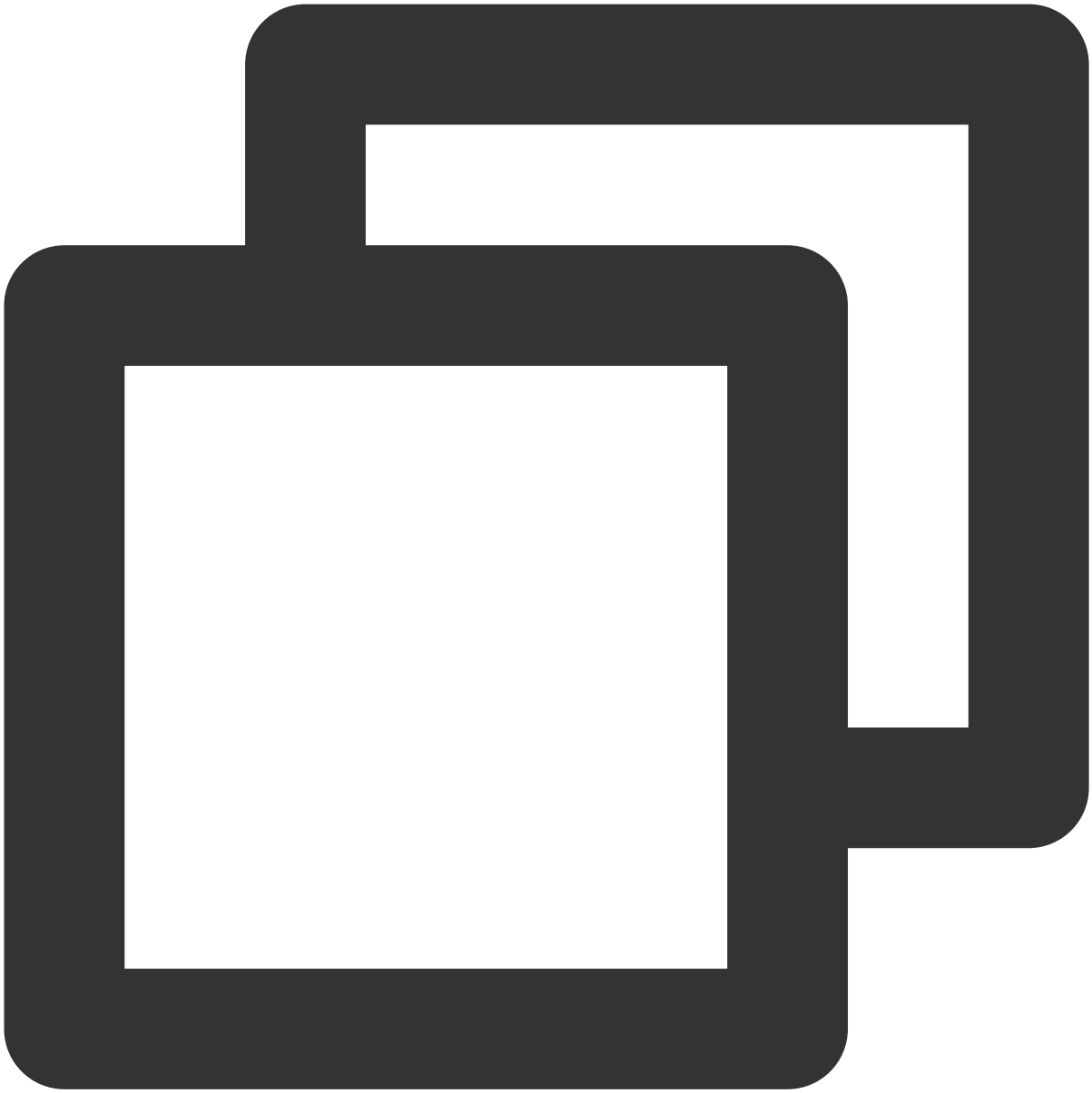


```
[TXLiveBase setLicenceURL:LICENSEURL key:LICENSEURLKEY];
```

Step 3: The audience interacts via mic.

1. Viewers who want to co-broadcasting need to enter the TRTC room for real-time interaction with the anchor.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enter the TRTC room and start streaming.
- (void)enterRoomWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example.
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend.
    params.userSig = @"userSig";
    // Replace with your SDKAppID.
    params.sdkAppId = 0;
    // Specify the anchor role.
```

©2013-2022 Tencent Cloud. All rights reserved.

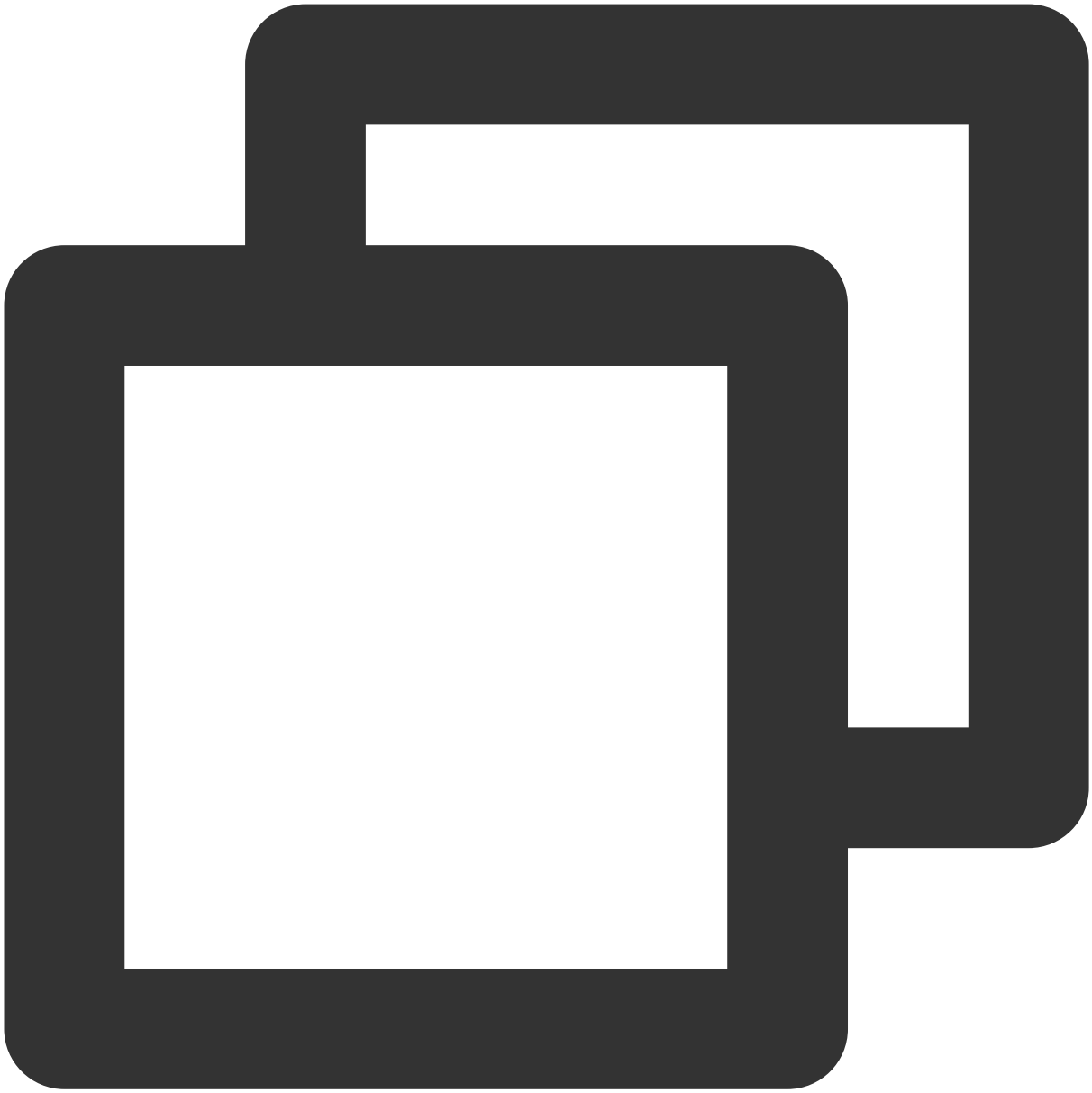
```
    params.role = TRTCRoleAnchor;

    // Enable local audio and video capture.
    [self startLocalMedia];
    // Enter the room in an interactive live streaming scenario.
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Enable local video preview and audio capture.
- (void)startLocalMedia {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify using the front/rear camera for video capture.
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.audiencePreviewV
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}

// Event callback for the result of entering the room.
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // result indicates the time taken (in milliseconds) to join the room.
        NSLog(@"Enter room succeed!");
    } else {
        // result indicates the error code when you fail to enter the room.
        NSLog(@"Enter room failed!");
    }
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

2. The mic-connection audience start subscribing to the anchor's audio and video streams after they successfully enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}

- (void)onFirstVideoFrame:(NSString *)userId streamType:(TRTCVideoStreamType)stream
    // The SDK starts rendering the first frame of the local or remote user's video
    if (![userId isEqualToString:@""]) {
        // Stop playing the CDN stream upon receiving the first frame of the anchor
        [self.livePlayer stopPlay];
    }
}
```

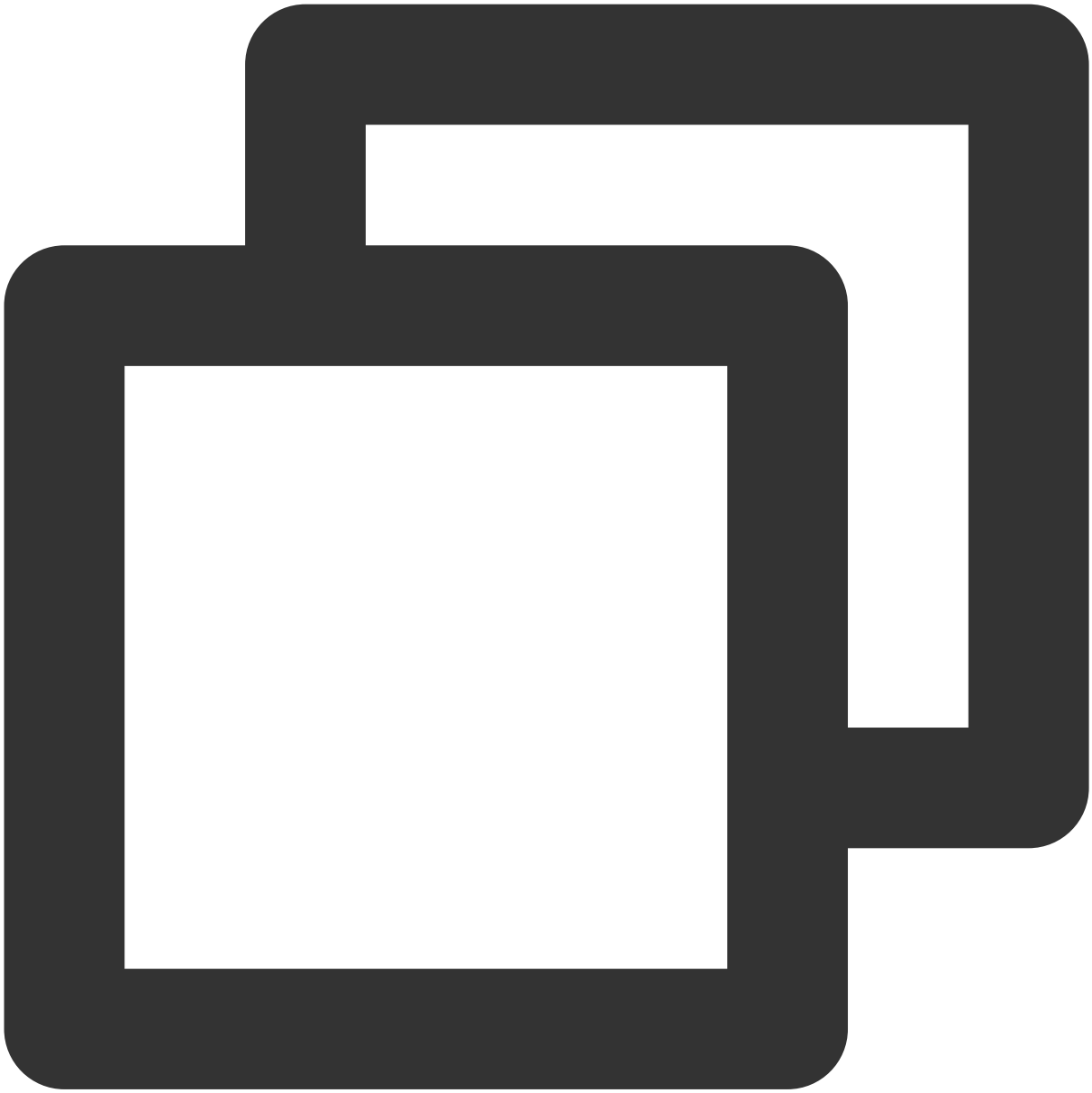
Note:

TRTC stream pulling `startRemoteView` can directly reuse the video rendering control previously used by the CDN stream pulling `setRenderView` .

To avoid video interruptions when switching between stream pullers, it is recommended to wait until the TRTC first frame callback `onFirstVideoFrame` is received before stopping the CDN stream pulling.

3. The anchor updates the publication of mixed media streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the mic-connection audience's room entry.
- (void)onRemoteUserEnterRoom:(NSString *)userId {
    if (![self.mixUserList containsObject:userId]) {
        [self.mixUserList addObject:userId];
    }
    [self updatePublishMediaToCDN];
}

// Update the publication of mixed media streams to the live streaming CDN.
- (void)updatePublishMediaToCDN {
    NSDate *date = [NSDate dateWithTimeIntervalSinceNow:0];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Set the expiration time for the push URLs.
NSTimeInterval time = [date timeIntervalSince1970] + (24 * 60 * 60);
// Generate authentication information. The getSafeUrl method can be obtained i
NSString *secretParam = [self getSafeUrl:LIVE_URL_KEY streamName:self.streamNam

// The target URLs for media stream publication.
TRTCPublishTarget* target = [[TRTCPublishTarget alloc] init];
// The target URLs are set for relaying the mixed streams to CDN.
target.mode = TRTCPublishMixStreamToCdn;
TRTCPublishCdnUrl* cdnUrl = [[TRTCPublishCdnUrl alloc] init];
// Construct push URLs (in RTMP format) to the live streaming service provider.
cdnUrl.rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@?%@", PUSH_DOMAI
// True means Tencent CSS push URLs, and false means third-party services.
cdnUrl.isInternalLine = YES;
NSMutableArray* cdnUrlList = [NSMutableArray array];
// Multiple CDN push URLs can be added.
[cdnUrlList addObject:cdnUrl];
target.cdnUrlList = cdnUrlList;

// Set media stream encoding output parameters.
TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
encoderParam.audioEncodedSampleRate = 48000;
encoderParam.audioEncodedChannelNum = 1;
encoderParam.audioEncodedKbps = 50;
encoderParam.audioEncodedCodecType = 0;
encoderParam.videoEncodedWidth = 540;
encoderParam.videoEncodedHeight = 960;
encoderParam.videoEncodedFPS = 15;
encoderParam.videoEncodedGOP = 2;
encoderParam.videoEncodedKbps = 1300;

TRTCStreamMixingConfig *config = [[TRTCStreamMixingConfig alloc] init];
if (self.mixUserList.count) {
    NSMutableArray<TRTCUser *> *userList = [NSMutableArray array];
    NSMutableArray<TRTCVideoLayout *> *layoutList = [NSMutableArray array];
    for (int i = 1; i < MIN(self.mixUserList.count, 16); i++) {
        TRTCUser *user = [[TRTCUser alloc] init];
        // The integer room number is intRoomId.
        user.strRoomId = self.roomId;
        user.userId = self.mixUserList[i];
        [userList addObject:user];

        TRTCVideoLayout *layout = [[TRTCVideoLayout alloc] init];
        if ([self.mixUserList[i] isEqualToString:self.userId]) {
            // The layout for the anchor's video.
            layout.rect = CGRectMake(0, 0, 540, 960);
            layout.zOrder = 0;
```

```
            } else {
                // The layout for the mic-connection audience's video.
                layout.rect = CGRectMake(400, 5 + i * 245, 135, 240);
                layout.zOrder = 1;
            }
            layout.fixedVideoUser = user;
            layout.fixedVideoStreamType = TRTCVideoStreamTypeBig;
            [layoutList addObject:layout];
        }
        // Specify the information for each input audio stream in the transcoding s
        config.audioMixUserList = [userList copy];
        // Specify the information of position, size, layer, and stream type for ea
        config.videoLayoutList = [layoutList copy];
    }
    // Update the published media stream.
    [self.trtcCloud updatePublishMediaStream:self.taskId publishTarget:target encod
}

// Event callback for updating the media stream.
- (void)onUpdatePublishMediaStream:(NSString *)taskId code:(int)code message:(NSStr
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure.
}
```

Note:

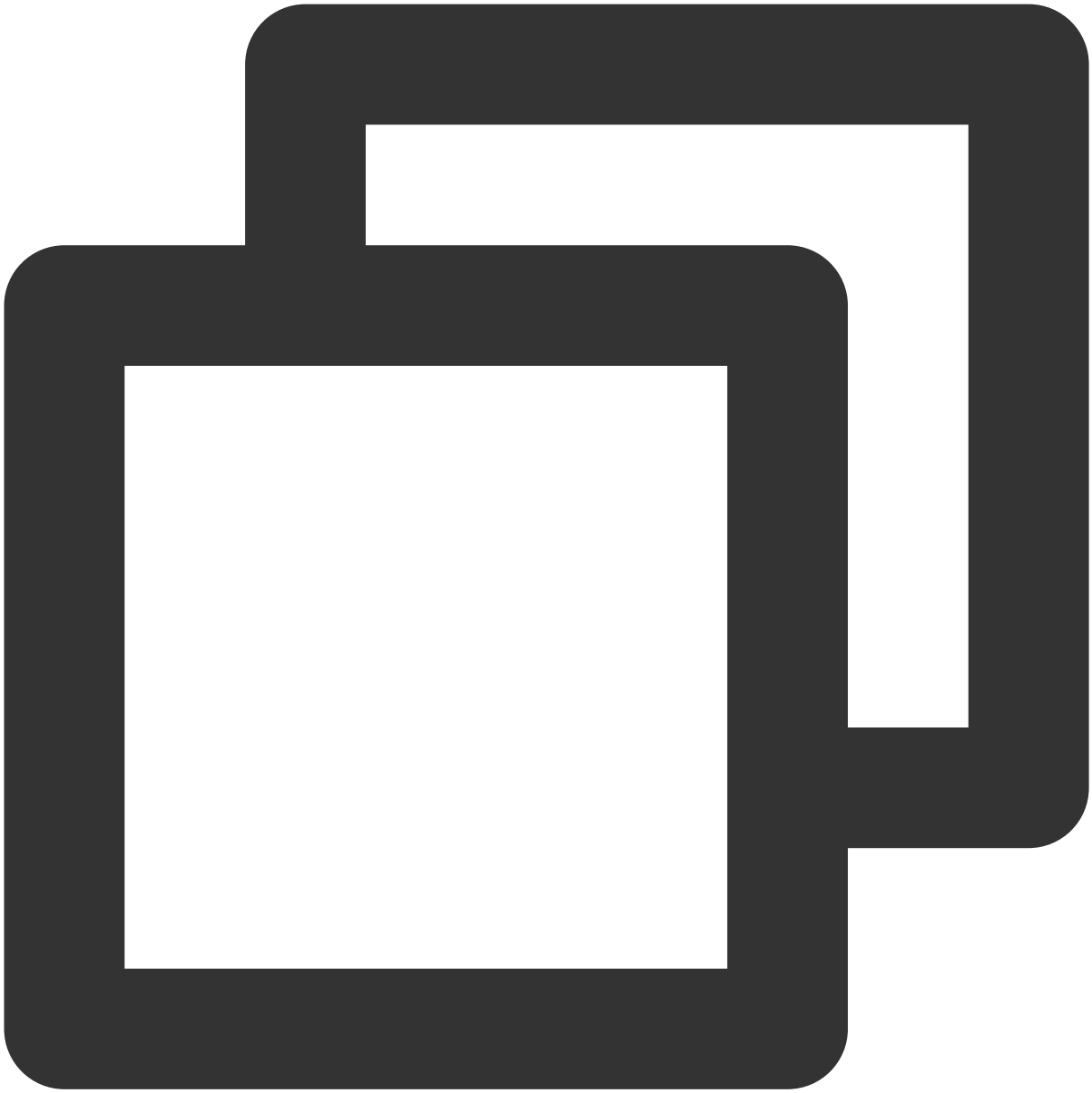
To ensure continuous CDN playback without stream disconnection, you need to keep the media stream encoding output parameter `encoderParam` and the stream name `streamName` unchanged.

Media stream encoding output parameters and mixed display layout parameters can be customized according to business needs. Currently, up to 16 channels of audio and video input are supported. If a user only provides audio, it will still be counted as one channel.

Switching between audio only, audio and video, and video only is not supported within the same task.

4. The off-streaming audience exit the room, and the anchor updates the mixed stream task.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the player callback listener.
[self.livePlayer setObserver:self];
// The reusable TRTC video rendering control.
[self.livePlayer setRenderView:self.remoteView];
// Restart playing CDN media stream.
[self.livePlayer startLivePlay:flvUrl];


- (void)onVideoLoading:(id<V2TXLivePlayer>)player extraInfo:(NSDictionary *)extraIn
    // Video loading event.
```

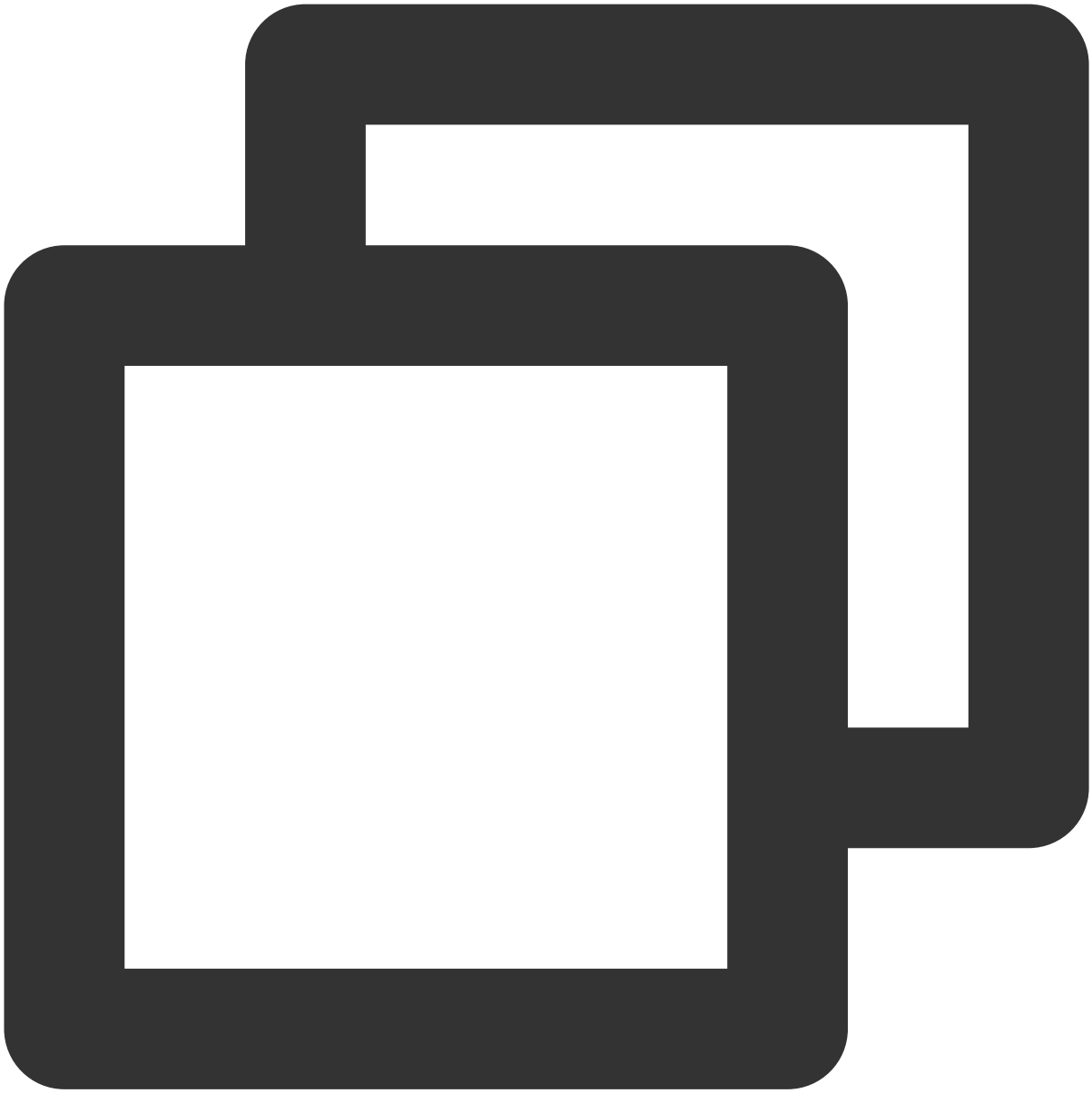
©2013-2022 Tencent Cloud. All rights reserved.

```
}

// Video playback event.
- (void)onVideoPlaying:(id<V2TXLivePlayer>)player firstPlay:(BOOL)firstPlay extraIn
    if (firstPlay) {
        [self.trtcCloud stopAllRemoteView];
        [self.trtcCloud stopLocalAudio];
        [self.trtcCloud stopLocalPreview];
        [self.trtcCloud exitRoom];
    }
}
```

Note:

To avoid video interruptions when switching the stream puller, it is recommended to wait for the player's video playback event `onVideoPlaying` before exiting the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.

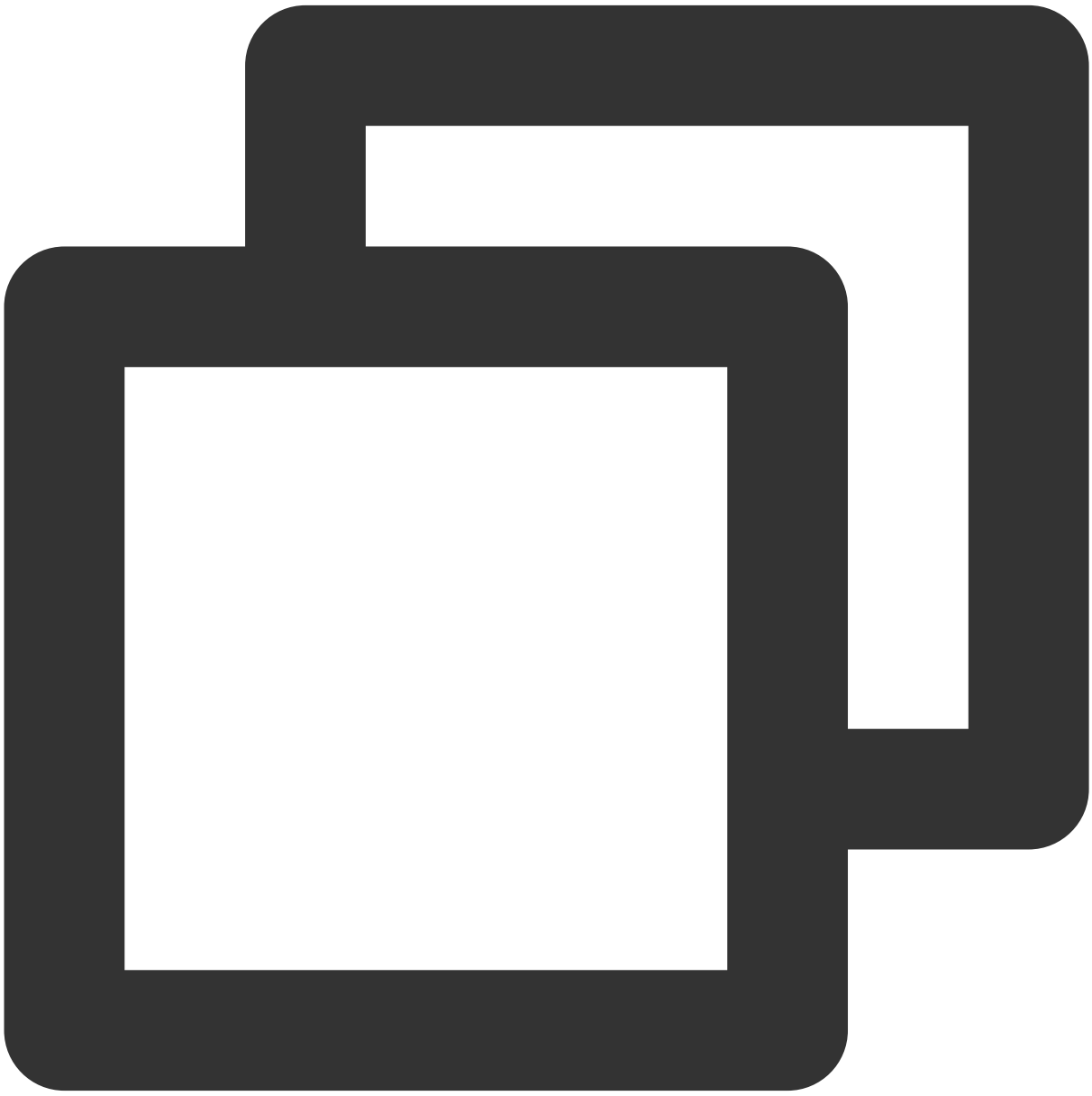


```
// Event callback for the mic-connection audience's room exit.
- (void)onRemoteUserLeaveRoom:(NSString *)userId reason:(NSInteger)reason {
    if ([self.mixUserList containsObject:userId]) {
        [self.mixUserList removeObject:userId];
    }
    // The anchor updates the mixed stream task.
    [self updatePublishMediaToCDN];
}

// Event callback for updating the media stream.
- (void)onUpdatePublishMediaStream:(NSString *)taskId code:(int)code message:(NSStr
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure.
}
```

Step 4: The anchor stops the live streaming and exits the room.



```
- (void)exitRoom {
    // Stop all published media streams.
    [self.trtcCloud stopPublishMediaStream:@""];
    [self.trtcCloud stopLocalAudio];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud stopLocalPreview];
    [self.trtcCloud exitRoom];
}

// Event callback for stopping media streams.
- (void)onStopPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStrin
    // When you call the stop publishing media stream API (stopPublishMediaStream),
    // code: Callback result. 0 means success and other values mean failure.
}

// Event callback for exiting the room.
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room.");
    } else if (reason == 1) {
        NSLog(@"Removed from the current room by the server.");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved.");
    }
}
```

Note:

To stop publishing media streams, fill in an empty string for `taskId`. This will stop all the media streams you have published.

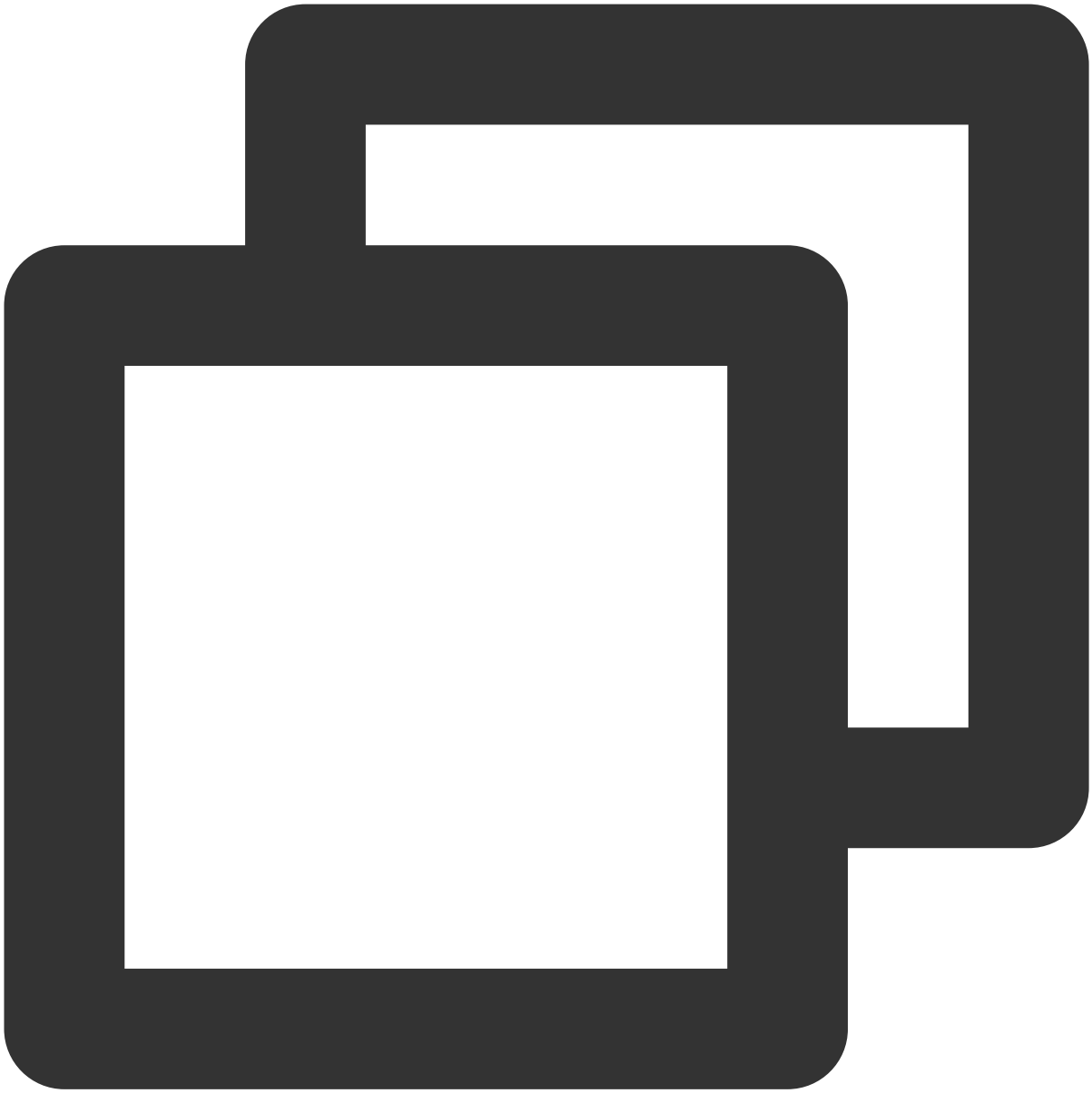
After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

Advanced Features

The anchor initiates the cross-room competition.

1. Either party initiates the cross-room competition.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)connectOtherRoom:(NSString *)roomId {
    NSMutableDictionary *jsonDict = [[NSMutableDictionary alloc] init];
    // The digit room ID is roomId.
    [jsonDict setObject:roomId forKey:@"strRoomId"];
    [jsonDict setObject:self.userId forKey:@"userId"];
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:NSJ
    NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8S
    [self.trtcCloud connectOtherRoom:jsonString];
}

// Result callback for requesting cross-room mic-connection.
```

©2013-2022 Tencent Cloud. All rights reserved.

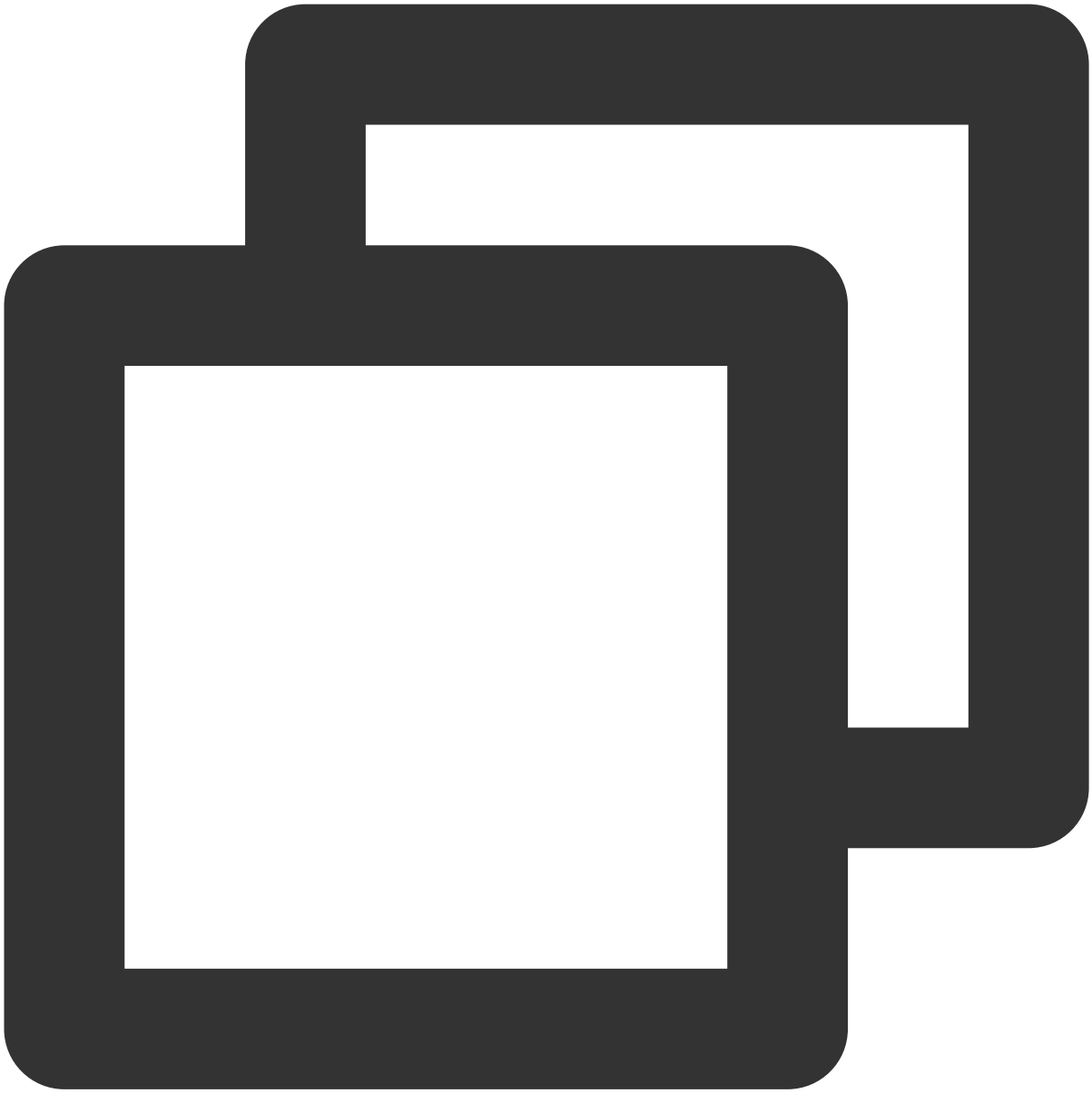
```
- (void)onConnectOtherRoom:(NSString *)userId errCode:(TXLiteAVError)errCode errMsg
    // The user ID of the anchor in the other room you want to initiate the cross-r
    // Error code. ERR_NULL indicates the request is successful.
    // Error message.
}
```

Note:

Both local and remote users participating in the cross-room competition must be in the anchor role and must have audio or video uplink capabilities.

2. All users in both rooms will receive a callback indicating that the audio and video streams from the PK anchor in the other room are available.

©2013-2022 Tencent Cloud. All rights reserved.

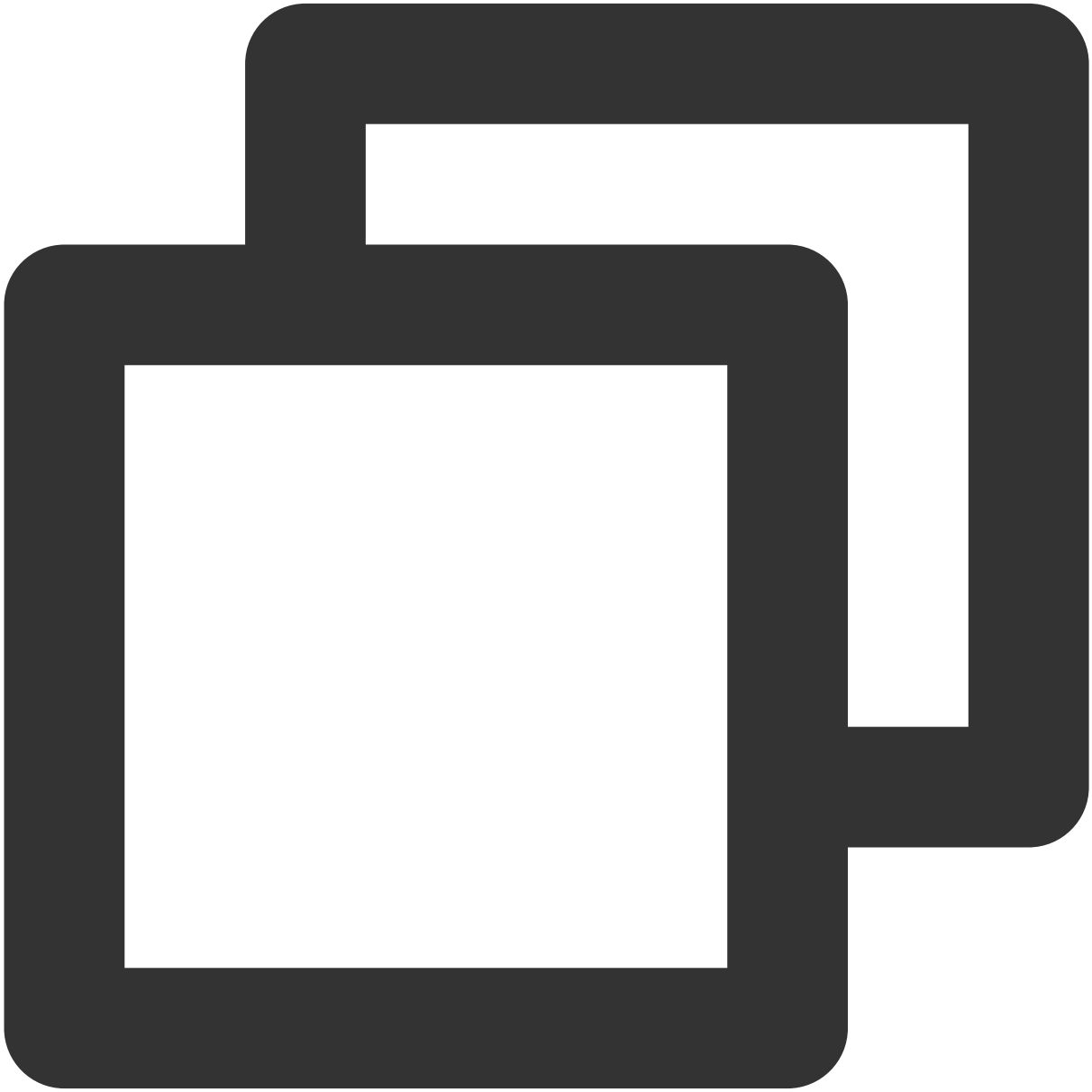


```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio.
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video.
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}
```

3. Either party exits the cross-room competition.



```
// Exit the cross-room mic-connection.
[self.trtcCloud disconnectOtherRoom];

// Result callback for exiting cross-room mic-connection.
- (void)onDisconnectOtherRoom:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Note:

After `DisconnectOtherRoom()` is called, you may exit the cross-room competition with all other room anchors. Either the initiator or the receiver can call `DisconnectOtherRoom()` to exit the cross-room competition.

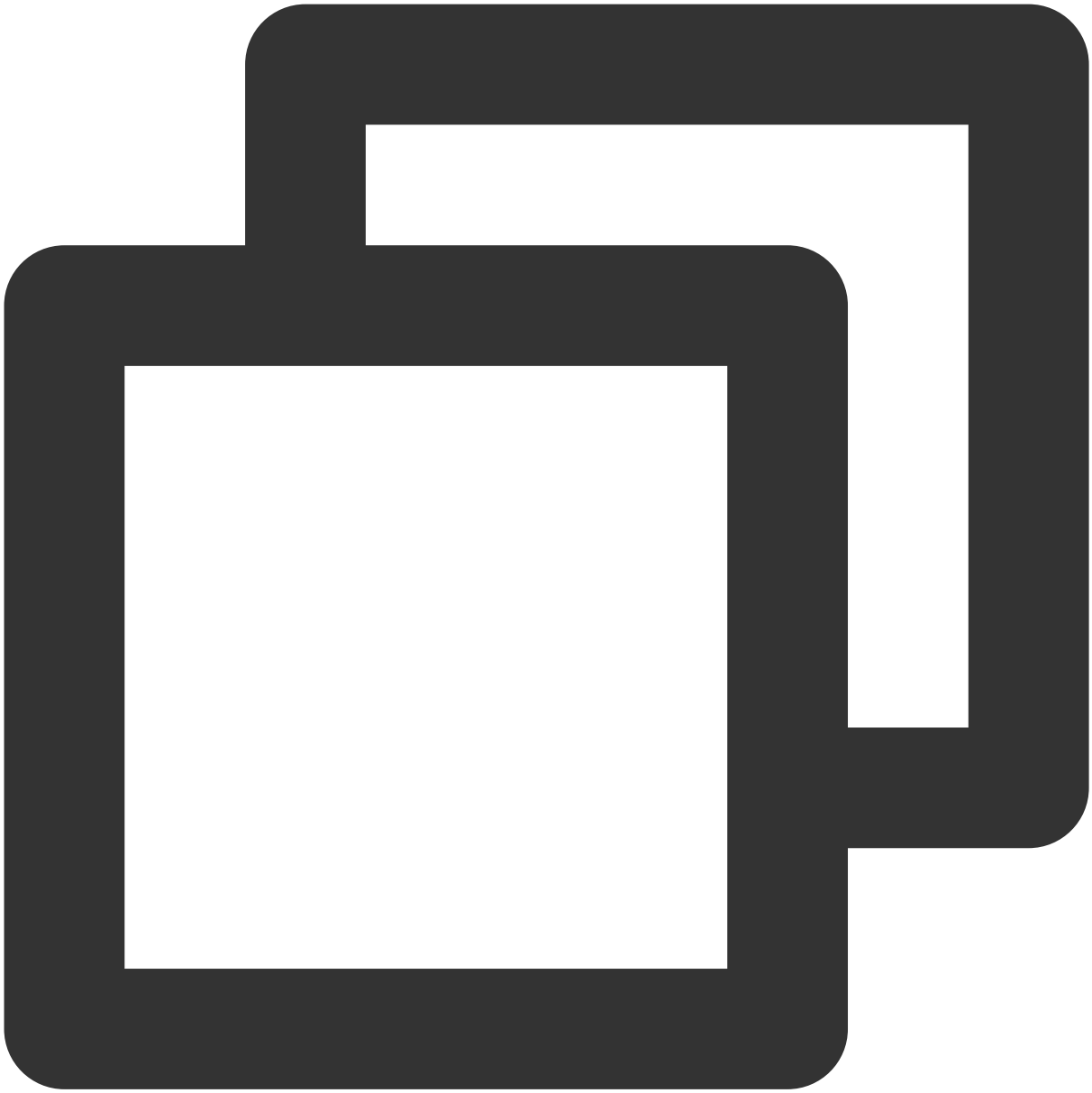
Integrate the third-party beauty features.

TRTC supports integrating third-party beauty effect products. Use the example of Tencent Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate the Tencent Effect SDK, and apply for an authorization license. For details, see Integration Preparation for steps.

2. Set the SDK material resource path (if any).

©2013-2022 Tencent Cloud. All rights reserved.



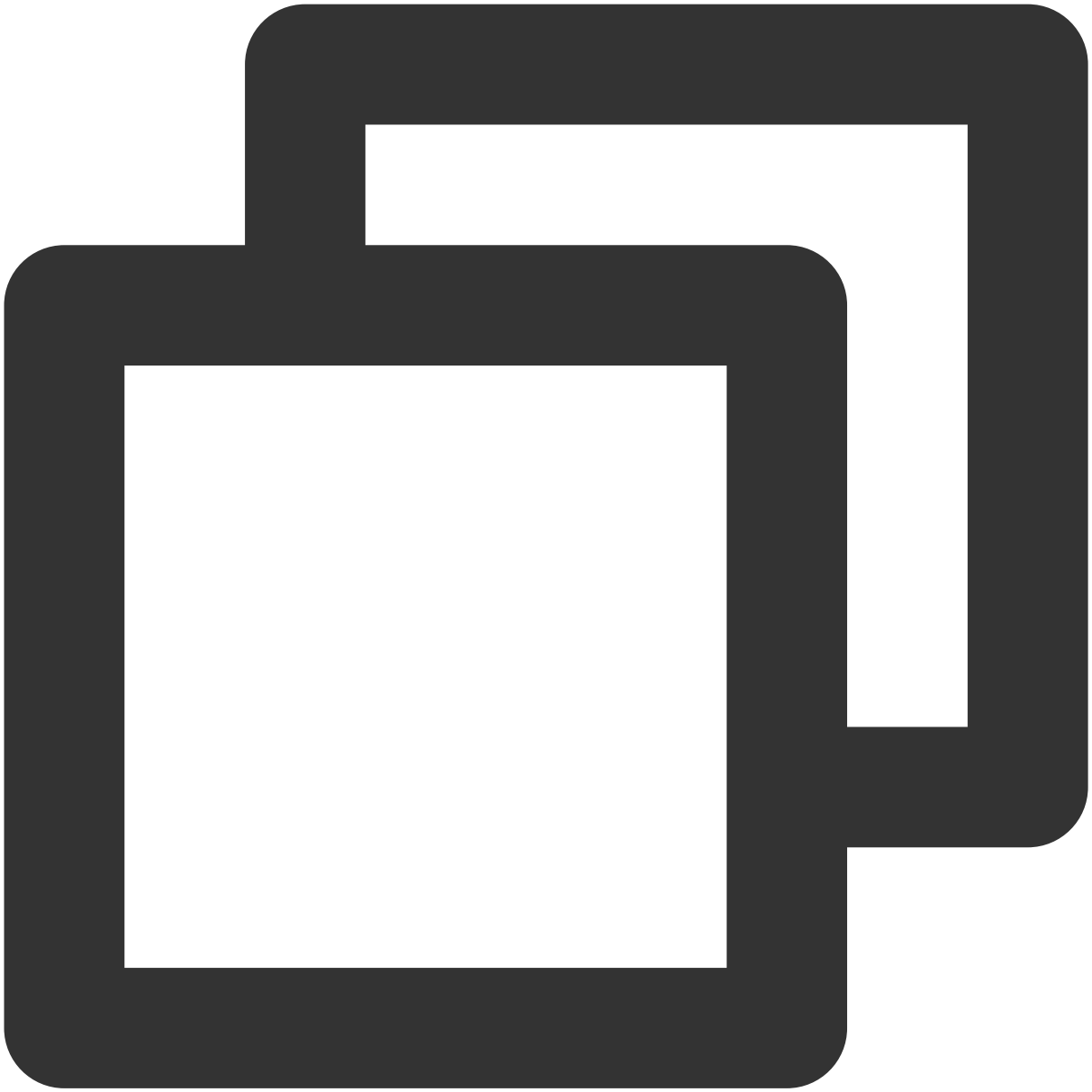
```
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                                       options:NSJSONReadingMutableContainers
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Initialize the SDK: Width and height are the width and height of the texture, re
self.xMagicKit = [[XMagic alloc] initWithRenderSize:CGSizeMake(width,height) assets
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the video data callback for third-party beauty features in the TRTC SDK.
[self.trtcCloud setLocalVideoProcessDelegete:self pixelFormat:TRTCVideoPixelFormat_

#pragma mark - TRTCVideoFrameDelegate

// Construct the YTProcessInput and pass it into the SDK for rendering processing.
- (uint32_t)onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVi
    if (!self.xMagicKit) {
        [self buildBeautySDK:srcFrame.width and:srcFrame.height texture:srcFrame.te
        self.heightF = srcFrame.height;
        self.widthF = srcFrame.width;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    if(self.xMagicKit!=nil && (self.heightF!=srcFrame.height || self.widthF!=srcFra
       self.heightF = srcFrame.height;
       self.widthF = srcFrame.width;
       [self.xMagicKit setRenderSize:CGSizeMake(srcFrame.width, srcFrame.height)];
    }
    YTProcessInput *input = [[YTProcessInput alloc] init];
    input.textureData = [[YTTextureData alloc] init];
    input.textureData.texture = srcFrame.textureId;
    input.textureData.textureWidth = srcFrame.width;
    input.textureData.textureHeight = srcFrame.height;
    input.dataType = kYTTextureData;
    YTProcessOutput *output = [self.xMagicKit process:input withOrigin:YtLightImage
    dstFrame.textureId = output.textureData.texture;
    return 0;
}
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products. And **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

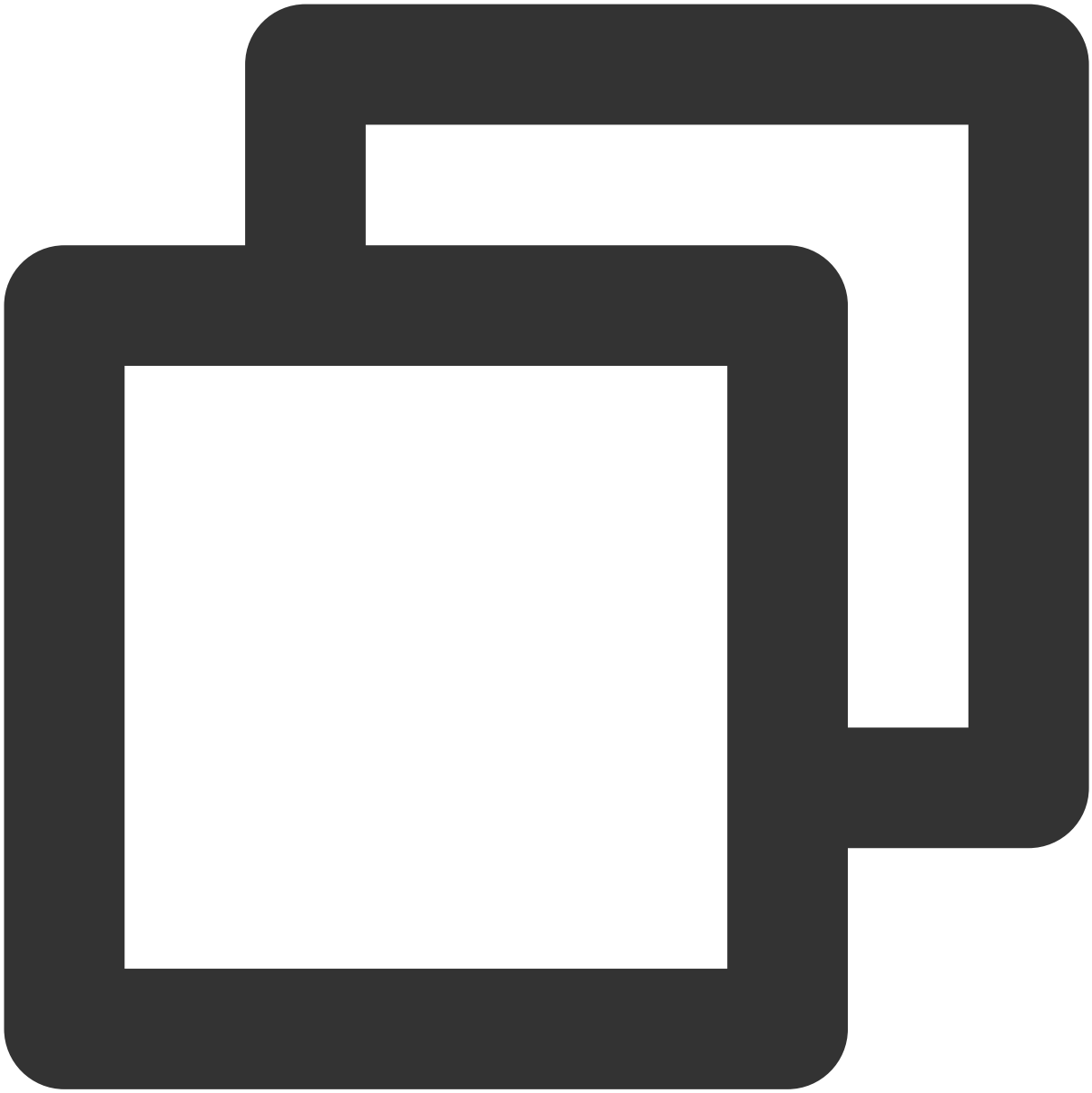
For scenario-specific integration guidelines of Tencent Effect, see Integrating Tencent Effect into TRTC SDK. For guidelines on integrating Tencent Effect independently, see Integrating Tencent Effect SDK.

Dual-stream encoding mode

When the dual-stream encoding mode is enabled, the current user's encoder will output two video streams, a high-definition large screen, and a low-definition small screen, at the same time (but only one audio stream). In this way, other users in the room can choose to subscribe to the high-definition large screen or low-definition small screen based on their network conditions or screen sizes.

1. Enable large-and-small-screen dual-stream encoding mode.

©2013-2022 Tencent Cloud. All rights reserved.

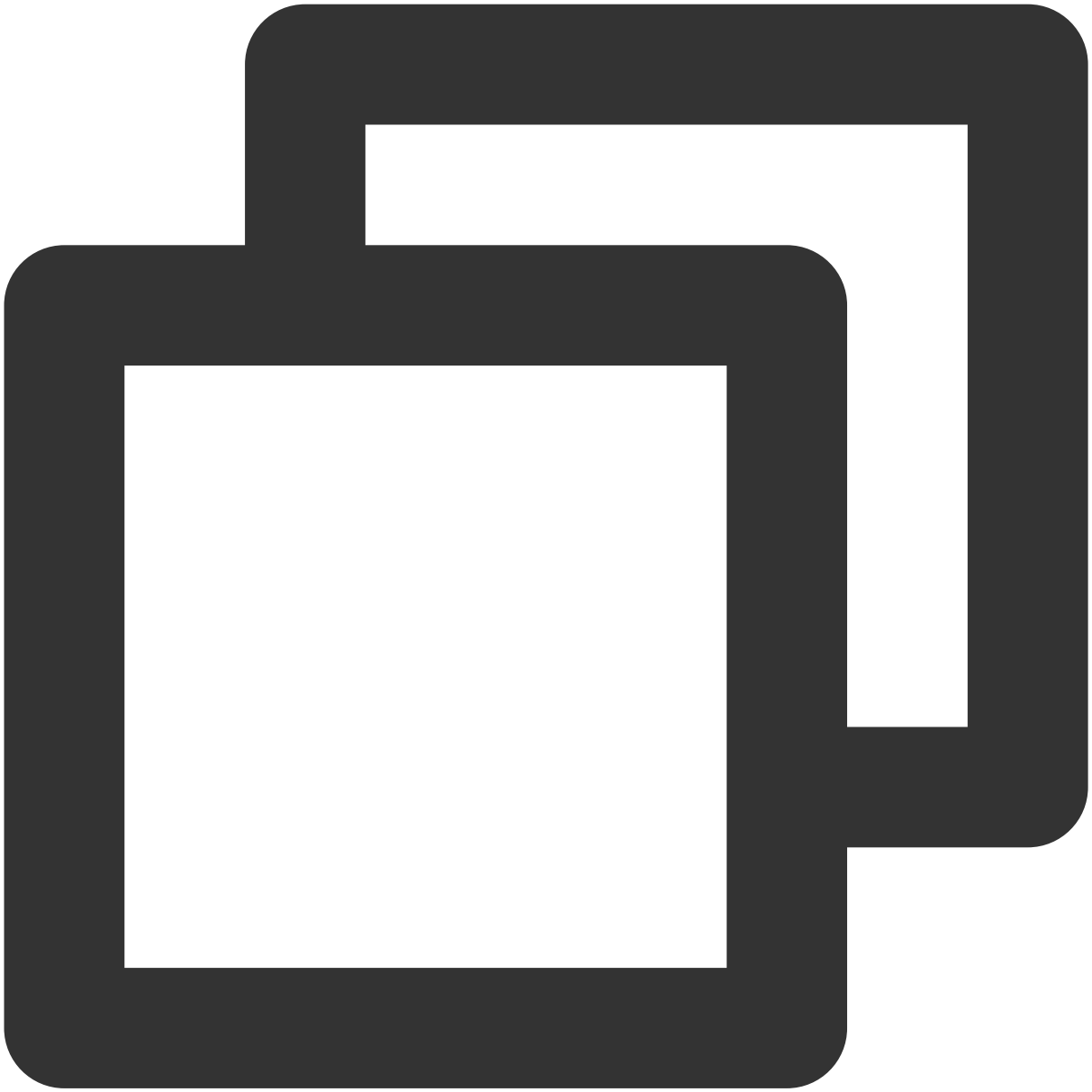


```
- (void)enableDualStreamMode:(BOOL)enable {
    // Video encoding parameters for the small-screen stream (customizable).
    TRTCVideoEncParam *smallVideoEncParam = [[TRTCVideoEncParam alloc] init];
    smallVideoEncParam.videoResolution = TRTCVideoResolution_480_270;
    smallVideoEncParam.videoFps = 15;
    smallVideoEncParam.videoBitrate = 550;
    smallVideoEncParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud enableEncSmallVideoStream:enable withQuality:smallVideoEncParam
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, it will consume more CPU and network bandwidth. Therefore, it may be considered for use on Mac, Windows, or high-performance Pads. It is not recommended for mobile devices.
2. Choose the type of remote user's video stream to pull.



```
// Optional video stream types when subscribing to a remote user's video stream.
[self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig view:view]

// You can switch the size of the specified remote user's screen at any time.
[self.trtcCloud setRemoteVideoStreamType:userId type:TRTCVideoStreamTypeSmall];
```

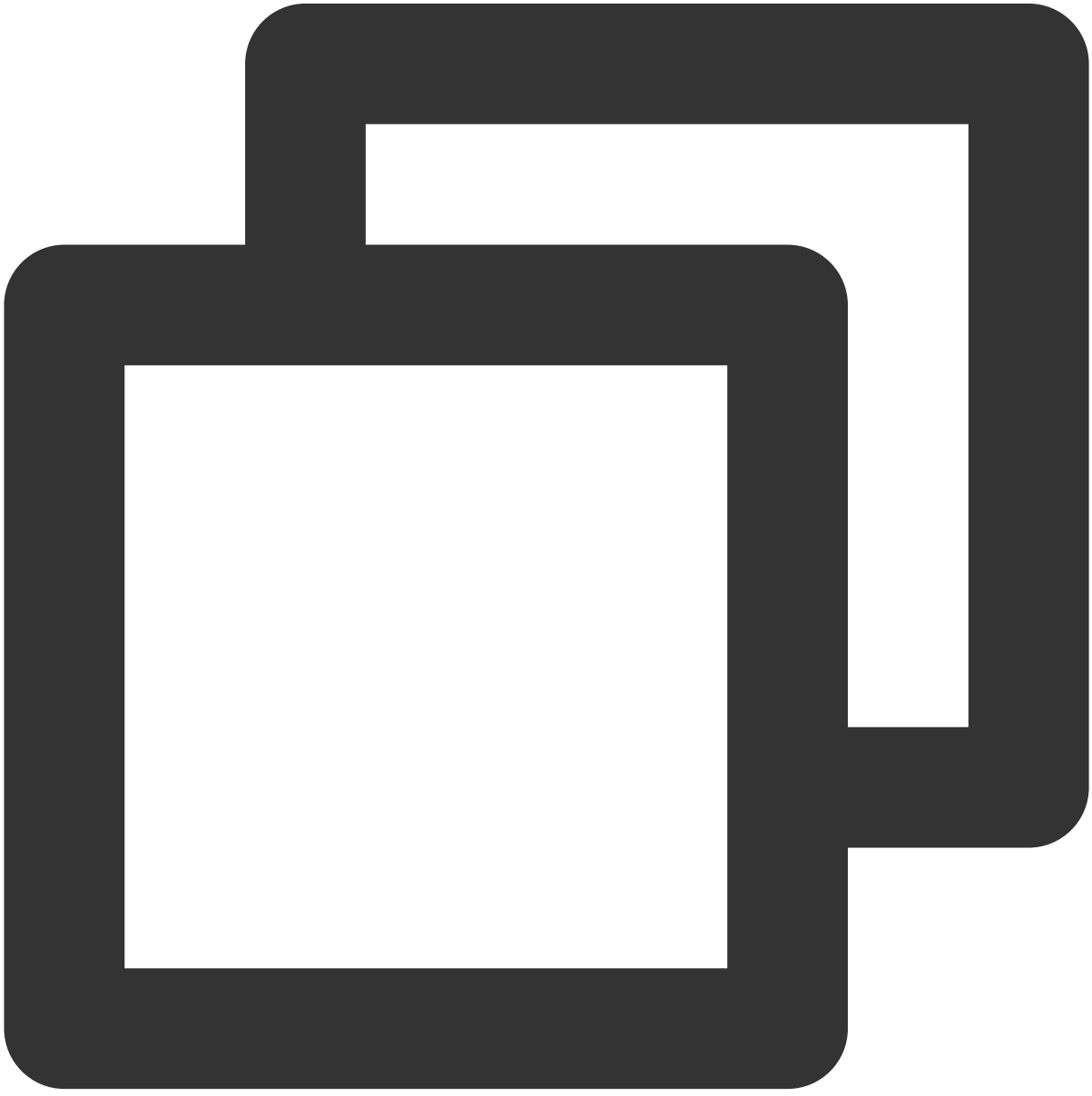
Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, you can specify the video stream type as `TRTCVideoStreamTypeSmall` with `streamType` to pull a low-quality small video for viewing.

View rendering control

If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.



```
// Update local preview screen rendering control.
[self.trtcCloud updateLocalView:view];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Update the remote user's video rendering control.
[self.trtcCloud updateRemoteView:view streamType:TRTCVideoStreamTypeBig forUser:use
```

Note:

The parameter `view` refers to the target video rendering control. And `streamType` only supports `TRTCVideoStreamTypeBig` and `TRTCVideoStreamTypeSub` .

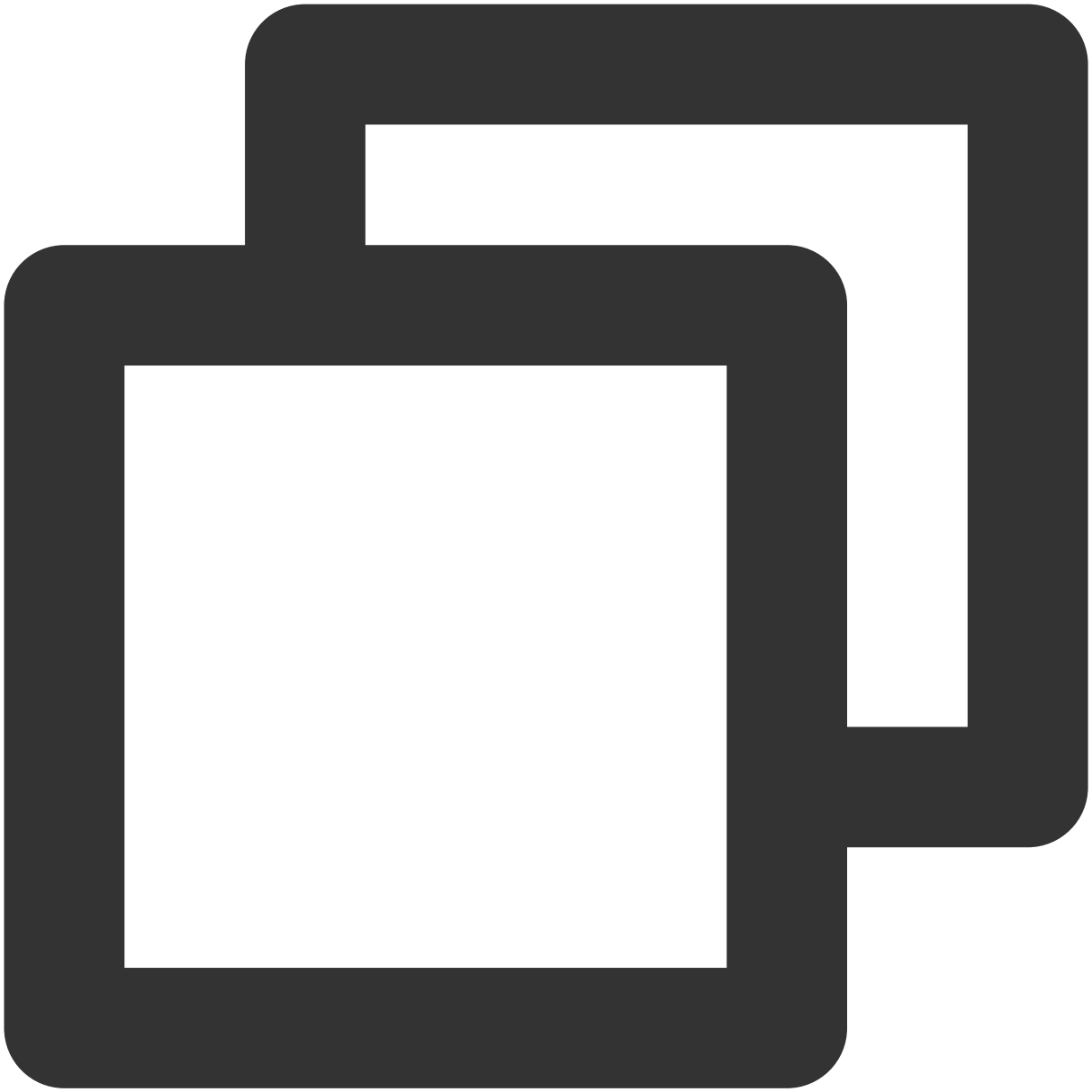
Live Streaming Interactive Messages

Live streaming interaction is particularly important in live streaming scenarios. Users interact with the anchor through like messages, gift messages, and bullet screen messages. The precondition for implementing the live interaction feature is to activate the Instant Messaging (IM) service and import the IM SDK. For detailed guidelines, see Voice Chat Room Integration Guide - Preparation for Integration.

Like message

1. The liker sends custom group messages related to likes through the client. After it is sended successfully, the business party renders the likes effect locally.

©2013-2022 Tencent Cloud. All rights reserved.

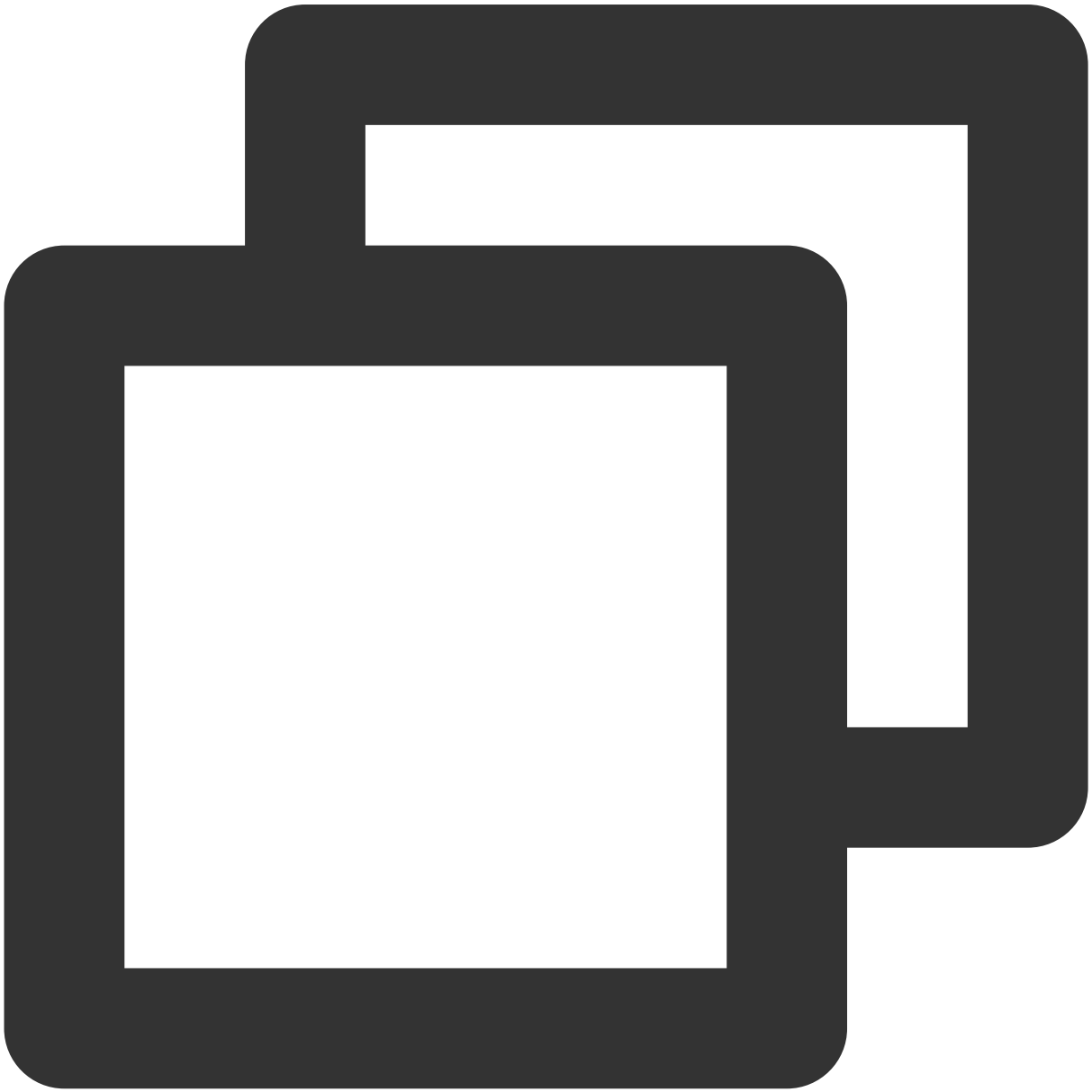


```
// Construct the likes message body.
NSDictionary *msgDict = @{
    @"type": @1,        // Like type
    @"likeCount": @10   // Number of likes
};
NSDictionary *dataDict = @{
    @"cmd": @"like_msg",
    @"msg": msgDict
};
NSError *error;
NSData *data = [NSJSONSerialization dataWithJSONObject:dataDict options:0 error:&er
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Send custom group messages (it is recommended that like messages should be set t
[[V2TIMManager sharedInstance] sendGroupCustomMessage:data to:groupID priority:V2TI
    // Like messages sent successfully.
    // Local rendering of likes effect.
} fail:^(int code, NSString *desc) {
    // Failed to send like messages.
}];
```

2. Other users in the room receive callback for custom group messages. Then proceed with message parsing and likes effect rendering.



©2013-2022 Tencent Cloud. All rights reserved.

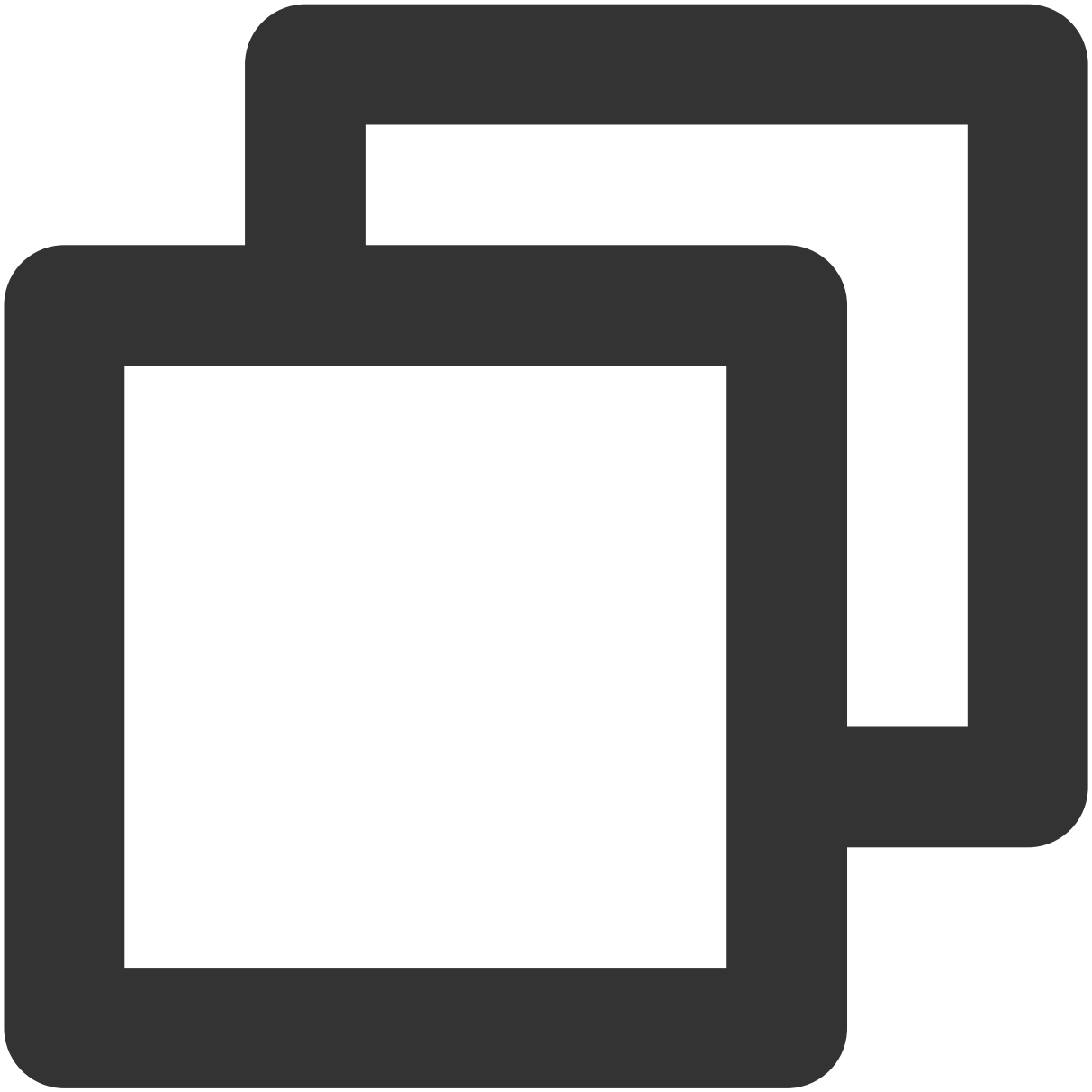
```
// Custom group messages received.
[[V2TIMManager sharedInstance] addSimpleMsgListener:self];
- (void)onRecvGroupCustomMessage:(NSString *)msgID groupID:(NSString *)groupID send
    if (data.length > 0) {
        NSError *error;
        NSDictionary *dataDict = [NSJSONSerialization JSONObjectWithData:data optio
        if (!error) {
            NSString *command = dataDict[@"cmd"];
            NSDictionary *msgDict = dataDict[@"msg"];
            if ([command isEqualToString:@"like_msg"]) {
                NSNumber *type = msgDict[@"type"];            // Likes type.
                NSNumber *likeCount = msgDict[@"likeCount"];            // Number o
                // Render likes effect based on likes type and count.
            }
        } else {
            NSLog(@"Parsing error: %@", error.localizedDescription);
        }
    }
}
```

Gift messages

1. The sender initiates a request to the business server. Upon completing the billing and settlement, the business server calls the REST API to send a custom message to the group.

1.1 Request URL sample:

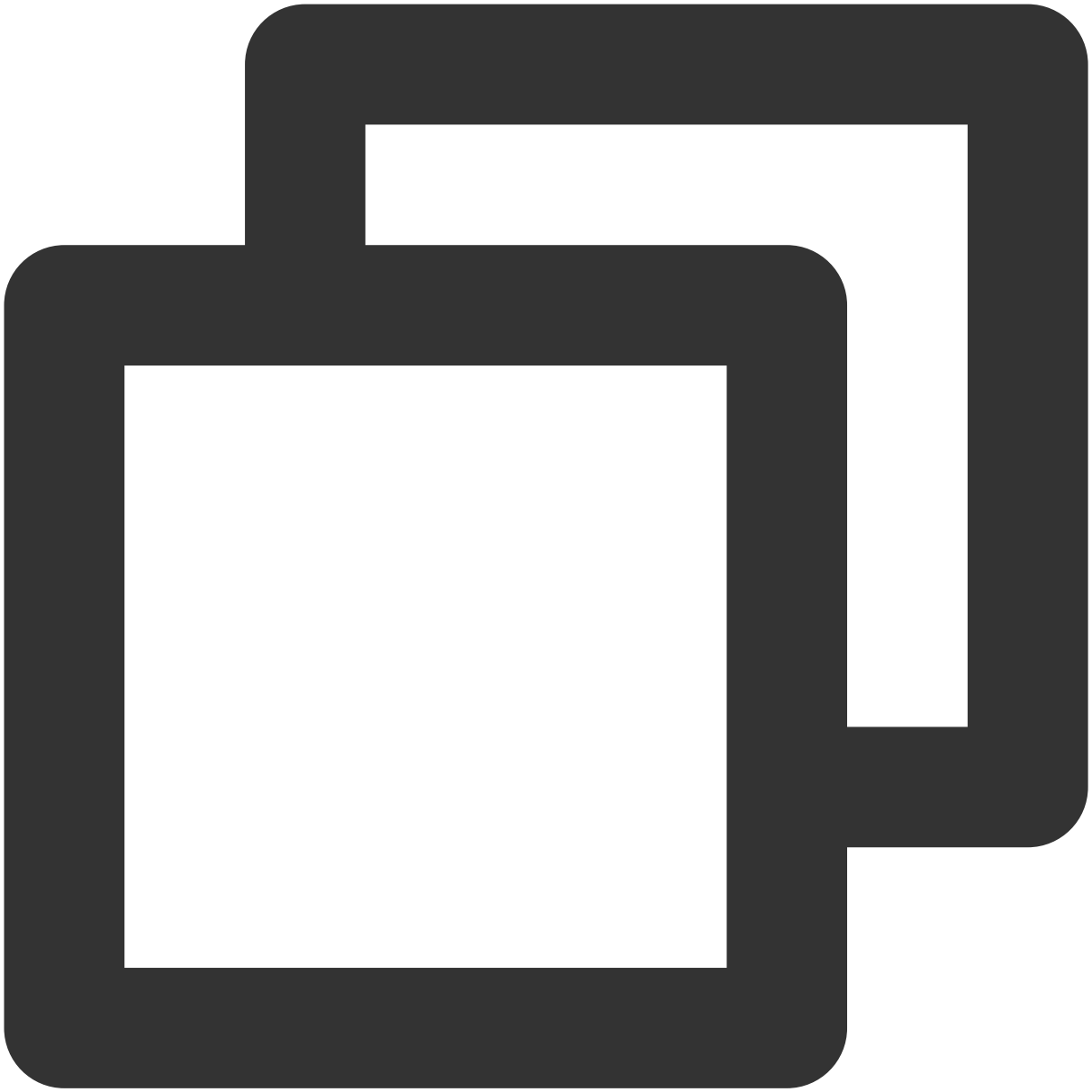
©2013-2022 Tencent Cloud. All rights reserved.



```
https://xxxxxx/v4/group_open_http_svc/send_group_msg?sdkappid=88888888&identifier=a
```

1.2 Request packet body sample:

©2013-2022 Tencent Cloud. All rights reserved.

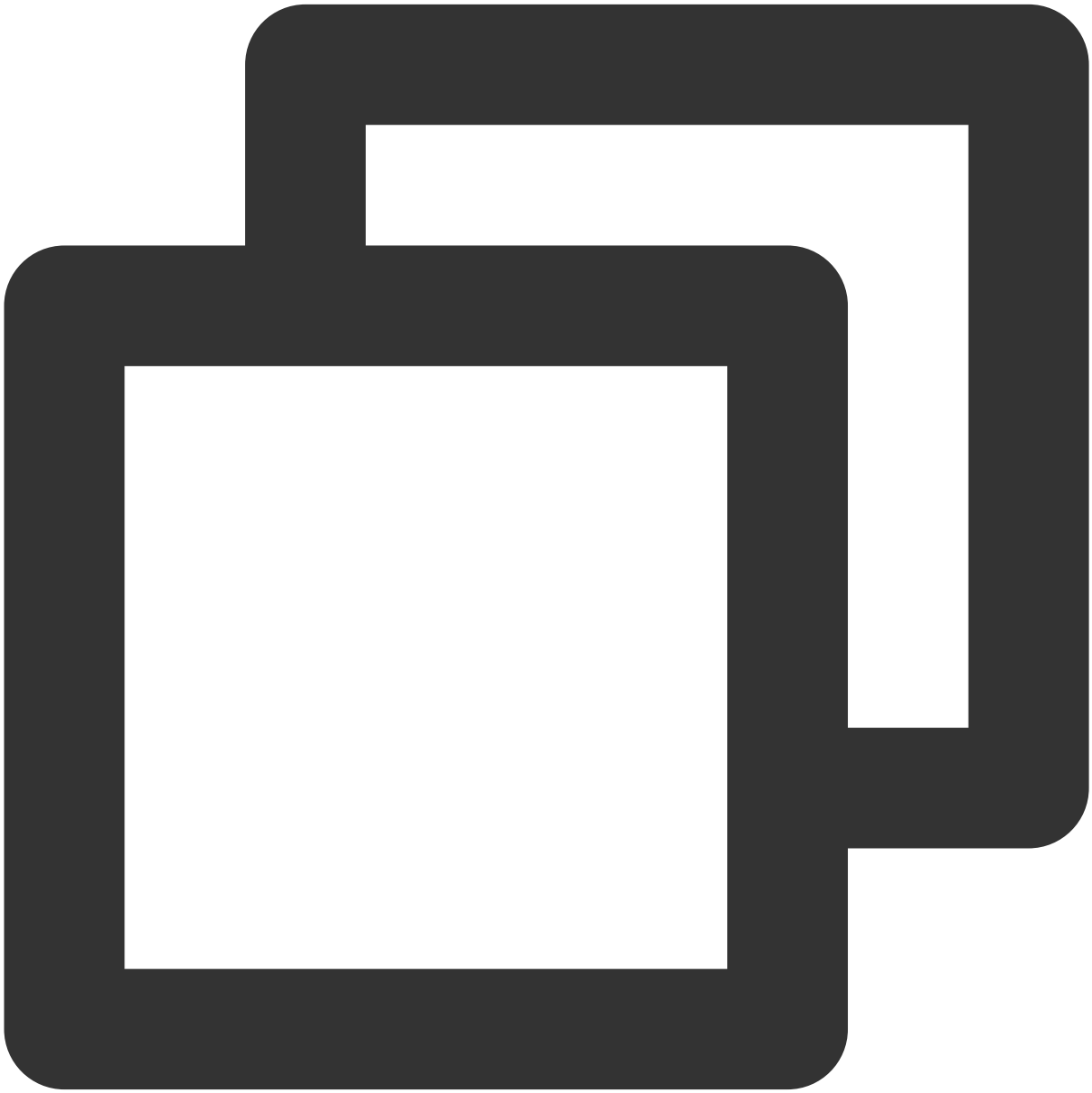


```
{
    "GroupId": "@TGS#12DEVUDHQ",
    "Random": 2784275388,
    "MsgPriority": "High",  // The priority of the message. Gift messages should be
    "MsgBody": [
        {
            "MsgType": "TIMCustomElem",
            "MsgContent": {
                // type: gift type; giftUrl: gift resource URL; giftName: gift name
                "Data": "{\\"cmd\\": \\"gift_msg\\", \\"msg\\": {\\"type\\": 1, \\"
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    ]
}
```

2. Other users in the room receive a callback for custom group messages. Then proceed with message parsing and gift effect rendering.



```
// Custom group messages received.
[[V2TIMManager sharedInstance] addSimpleMsgListener:self];
- (void)onRecvGroupCustomMessage:(NSString *)msgID groupID:(NSString *)groupID send
    if (data.length > 0) {
```

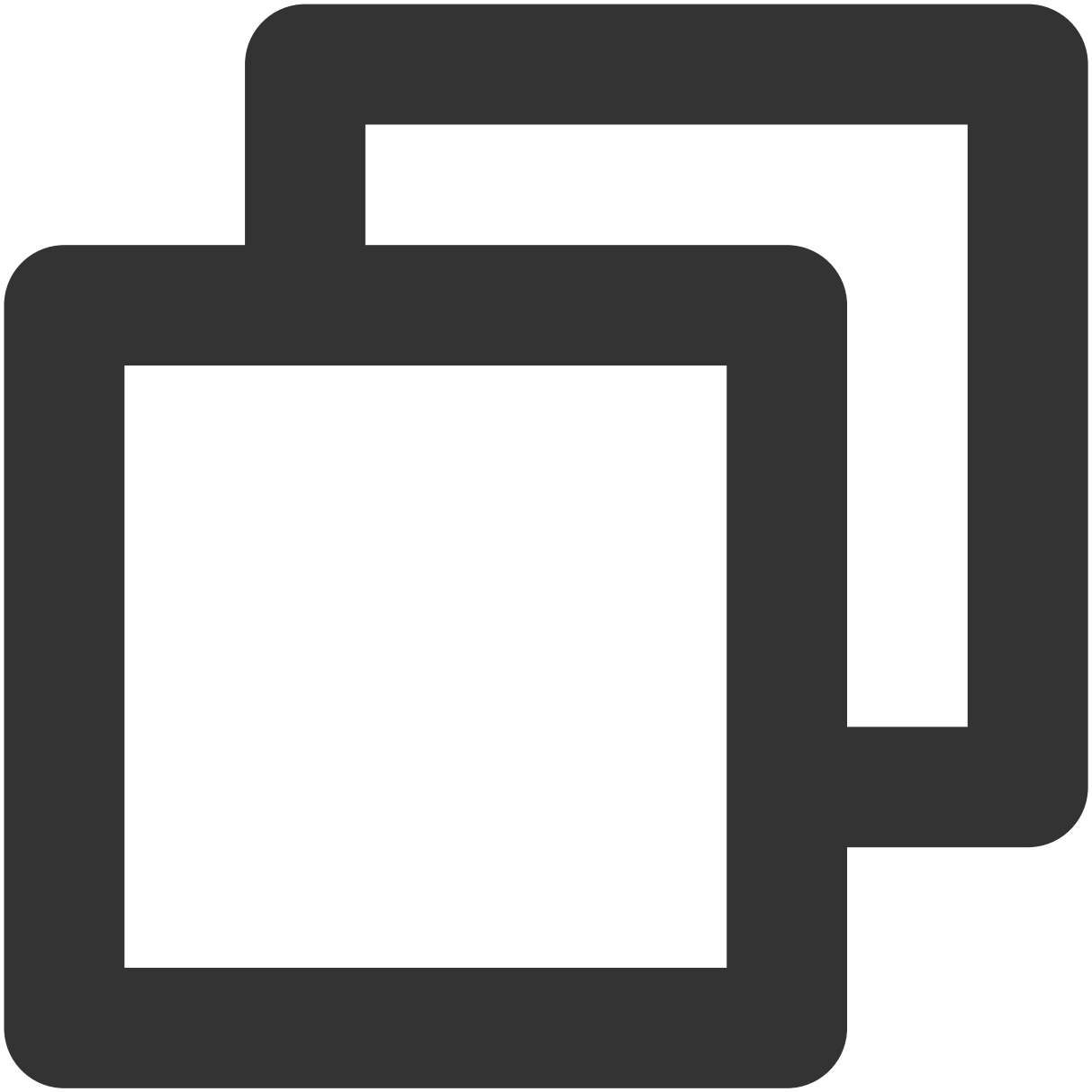
©2013-2022 Tencent Cloud. All rights reserved.

```
        NSError *error;
        NSDictionary *dataDict = [NSJSONSerialization JSONObjectWithData:data optio
        if (!error) {
            NSString *command = dataDict[@"cmd"];
            NSDictionary *msgDict = dataDict[@"msg"];
            if ([command isEqualToString:@"gift_msg"]) {
                NSNumber *type = msgDict[@"type"];            // Gift type.
                NSNumber *giftCount = msgDict[@"giftCount"];  // Number of gifts.
                NSString *giftUrl = msgDict[@"giftUrl"];      // Gift resource URL.
                NSString *giftName = msgDict[@"giftName"];    // Gift name.
                // Render gift effects based on gift type, count, resource URL, and
            }
        } else {
            NSLog(@"Parsing error: %@", error.localizedDescription);
        }
    }
}
```

Bullet screen messages

Live showroom usually have text-based bullet screen message interactions. This can be achieved through the sending and receiving of group chat regular text messages via IM.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Send public screen bullet screen messages.
[[V2TIMManager sharedInstance] sendGroupTextMessage:text to:groupID priority:V2TIM_
    // Successfully sent bullet screen messages.
    // Local display of the message text.
} fail:^(int code, NSString *desc) {
    // Failed to send bullet screen messages.
}];

// Receive public screen bullet screen messages.
[[V2TIMManager sharedInstance] addSimpleMsgListener:self];
- (void)onRecvGroupTextMessage:(NSString *)msgID groupID:(NSString *)groupID sender
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Rendering bullet screen messages based on sender info and message text.
}
```

Note:

The recommended priority setting is as follows. Gift messages should be set to high priority. Bullet screen messages should be set to medium priority. Like messages should be set to low priority.

Sending group chat messages from the client will not trigger the message reception callback. Only other users within the group can receive them.

Exception Handling

Exception error handling

When the TRTC SDK encounters an unrecoverable error, the error will be thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure will lead to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If failed to enter the room, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty.
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect. Check if `TRTCParams.roomId` or

©2013-2022 Tencent Cloud. All rights reserved.

		`TRTCParams.strRoomId` is empty. Note that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request is denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

Errors for relevant monitoring devices. Prompt the user via UI in case of relevant errors.

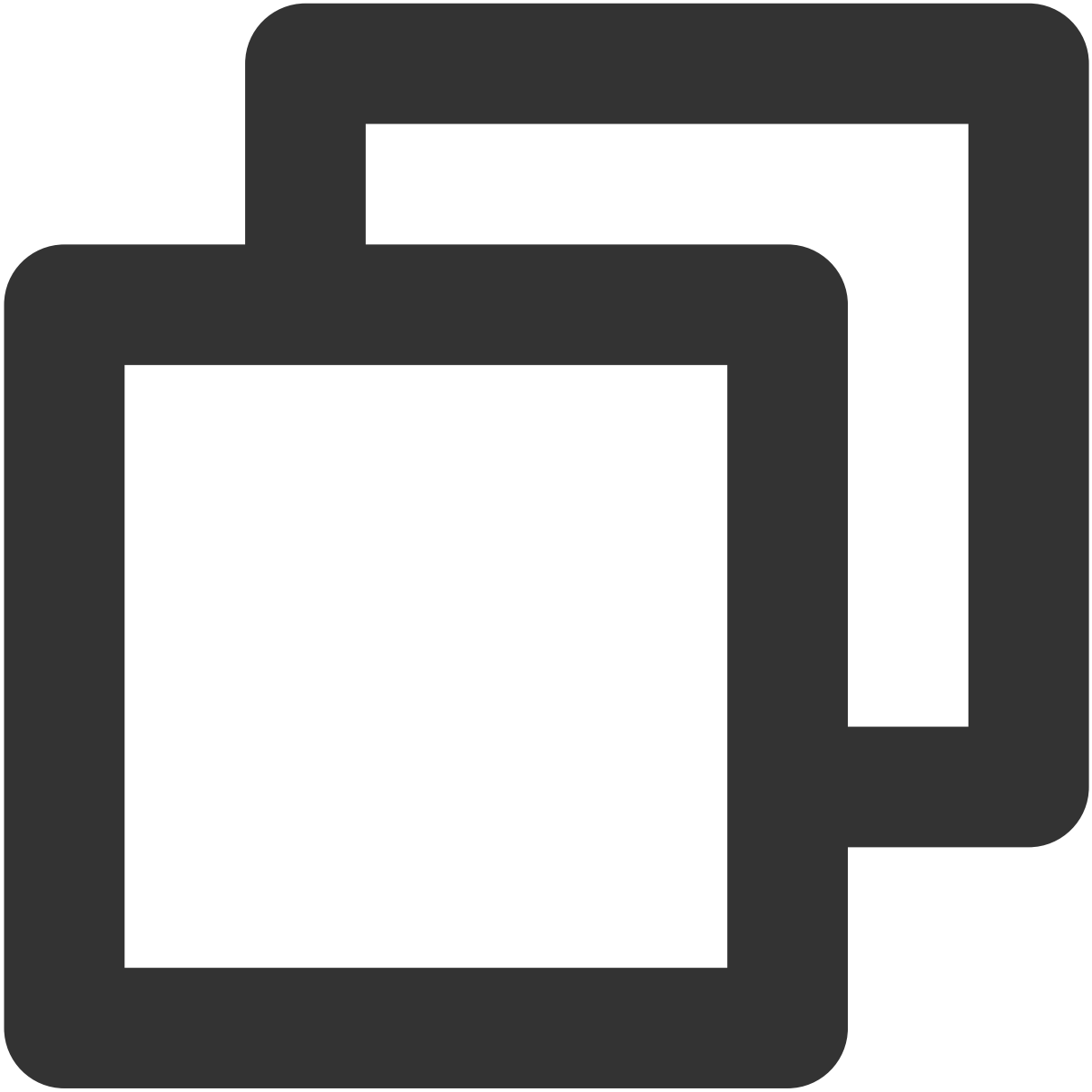
Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to open the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the mic's configuration program (driver) on a Windows or macOS device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Issues with the remote mirror mode not functioning properly.

In TRTC, video mirror settings are divided into local preview mirror `setLocalRenderParams` and video encoding mirror `setVideoEncoderMirror` . These settings individually affect the mirror effect of the local preview and the

©2013-2022 Tencent Cloud. All rights reserved.

output of the video encoding (the mirror mode affects remote viewers and cloud recordings). If you expect the mirror effect seen in the local preview to also take effect on the remote viewer's end, follow these encoding procedures.



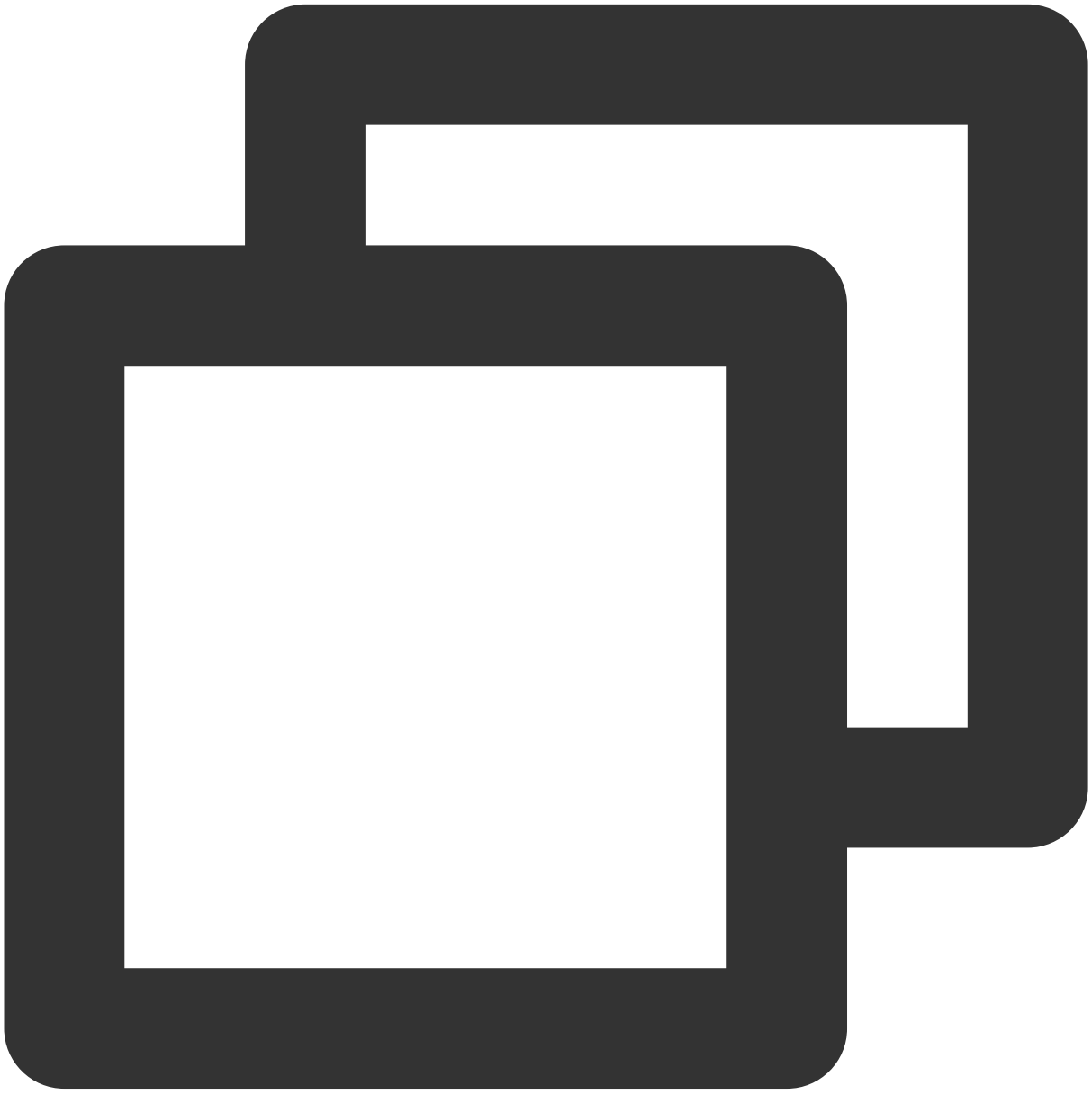
```
// Set the rendering parameters for the local video.
TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
params.mirrorType = TRTCVideoMirrorTypeEnable; // Video mirror mode
params.fillMode = TRTCVideoFillMode_Fill; // Video fill mode
params.rotation = TRTCVideoRotation_0; // Video rotation angle
[self.trtcCloud setLocalRenderParams:params];
// Set the video mirror mode for the encoder output.
[self.trtcCloud setVideoEncoderMirror:YES];
```

©2013-2022 Tencent Cloud. All rights reserved.

Issues with camera scale, focus, and switch.

In live showroom scenarios, the anchor may need to custom adjust the camera settings. The TRTC SDK's device management class provides APIs for these needs.

1. Query and set the zoom factor for the camera.

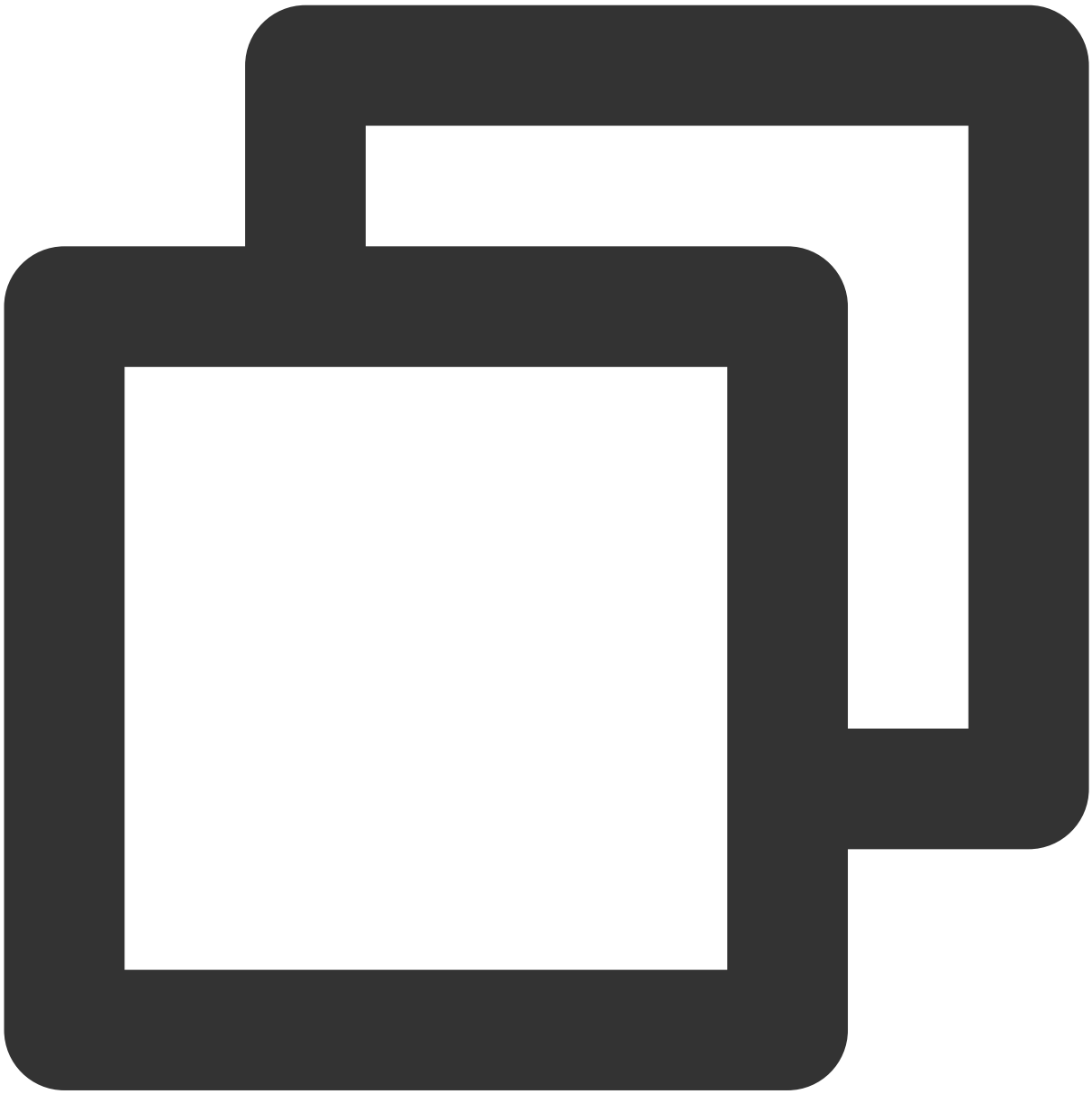


```
// Get the maximum zoom factor for the camera (only for mobile devices).
CGFloat zoomRatio = [[self.trtcCloud getDeviceManager] getCameraZoomMaxRatio];
// Set the zoom factor for the camera (only for mobile devices).
// Value range is 1 - 5. 1 means the furthest field of view (normal lens), and 5 me
```

©2013-2022 Tencent Cloud. All rights reserved.

```
[[self.trtcCloud getDeviceManager] setCameraZoomRatio:zoomRatio];
```

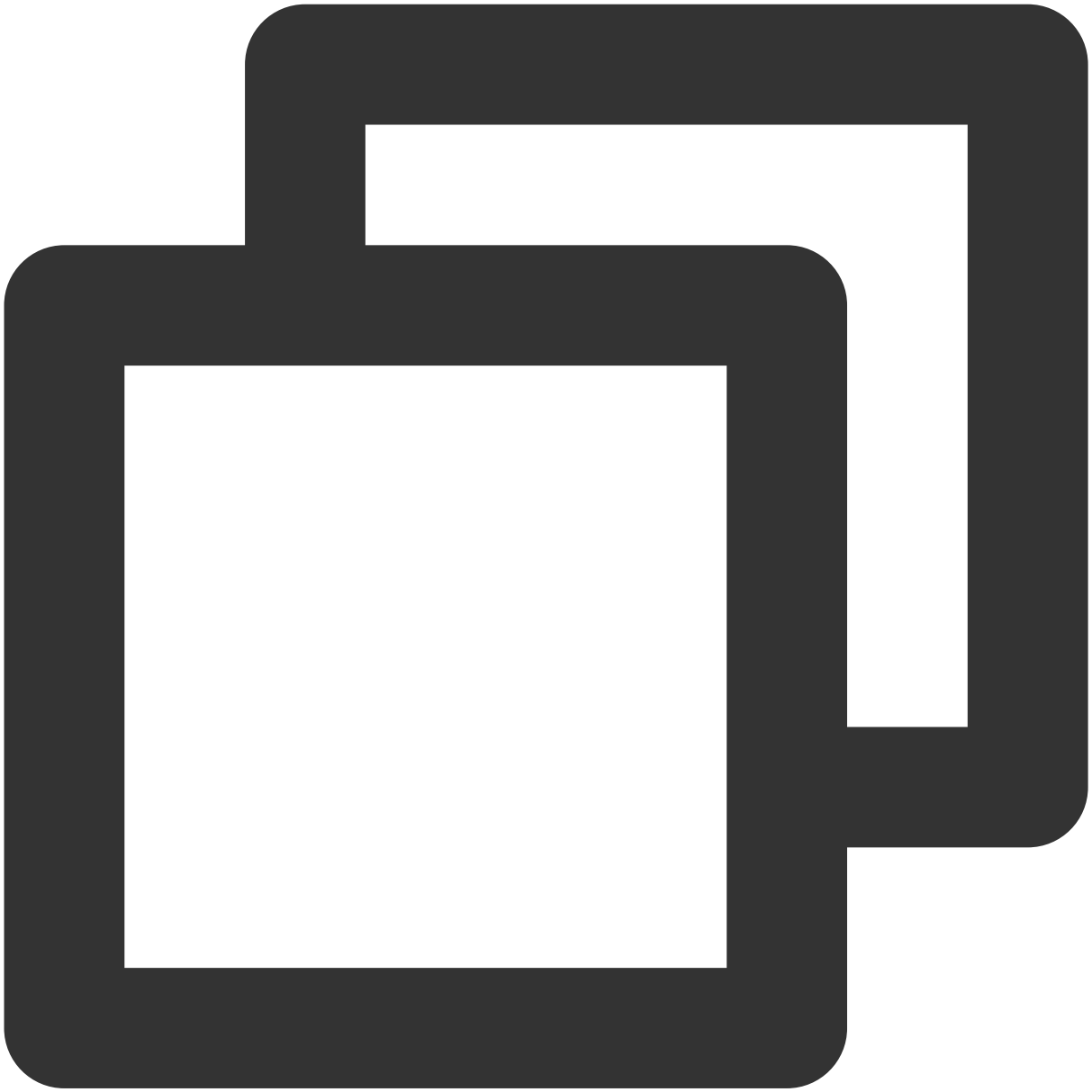
2. Set the focus feature and position of the camera.



```
// Enable or disable the camera's autofocus feature (only for mobile devices).
[[self.trtcCloud getDeviceManager] enableCameraAutoFocus:NO];
// Set the focus position of the camera (only for mobile devices).
// The precondition for using this API is to first disable the autofocus feature us
[[self.trtcCloud getDeviceManager] setCameraFocusPosition:CGPointMake(x, y)];
```

3. Determine and switch to front or rear cameras.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Determine if the current camera is the front camera (only for mobile devices).
BOOL isFrontCamera = [[self.trtcCloud getDeviceManager] isFrontCamera];
// Switch to front or rear cameras (only for mobile devices).
// Incoming true means switching to front, and incoming false means switching to re
[[self.trtcCloud getDeviceManager] switchCamera:!isFrontCamera];
```

©2013-2022 Tencent Cloud. All rights reserved.

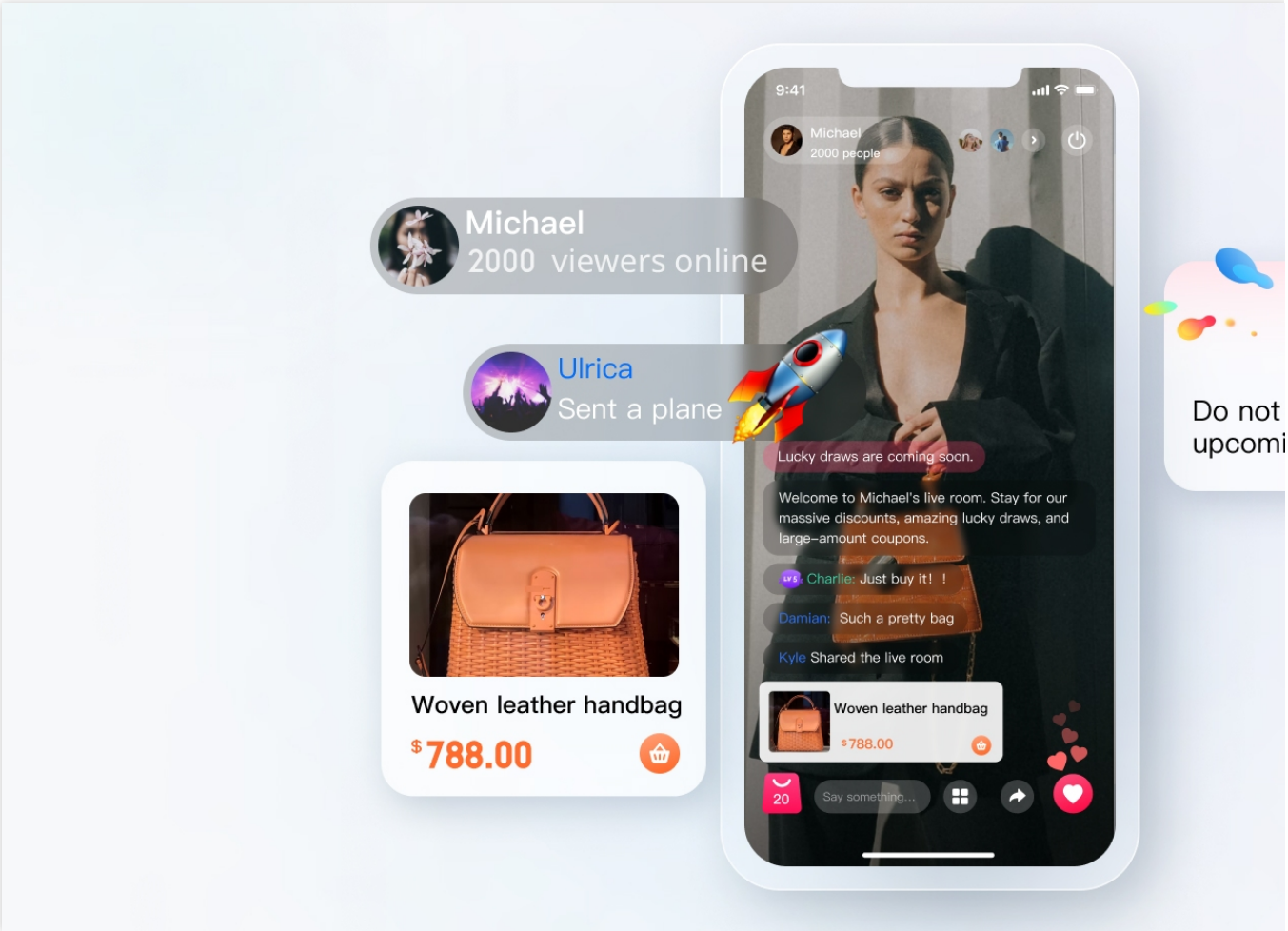
Live Shopping
Live Streaming with Goods
Scenario Solution

Last updated：2024-07-25 17:07:57

Scene Description

Live streaming with goods is an emerging e-commerce model that allows anchors to interact in real time with the audience via live streaming platforms, showcasing and promoting products, thereby selling them. In this scene, anchors typically present various products in the live streaming rooms, including clothing, cosmetics, and household items, and explain product features, discount information, and usage methods to the audience. Audiences can ask questions, comment, and purchase products in the live streaming rooms, achieving instant communication and transactions. By using Tencent Cloud Tencent Real-Time Communication (TRTC) combined with Instant Messaging (IM) and other products, you can easily set up a live shopping streaming room.

©2013-2022 Tencent Cloud. All rights reserved.



Implementation Scheme

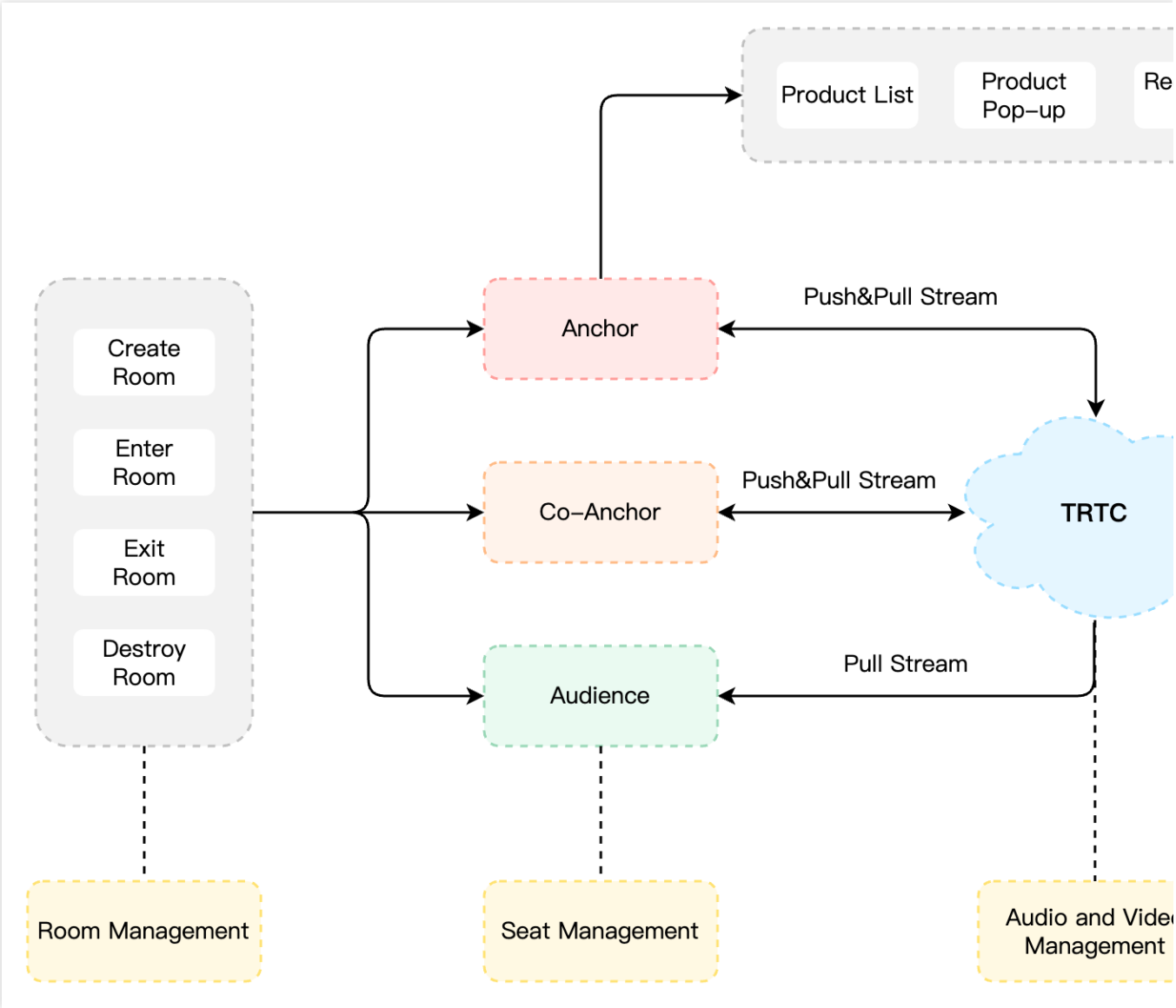
Typically, implementing a complete live streaming with goods scene involves several functional modules: Room Management, Seat Management, Product Management, Audio and Video Management, On-Cloud Recording, etc. The key actions and feature points under each functional module are shown in the table below. Each functional module will be introduced individually to provide a comprehensive understanding of the functionalities required for building a live shopping scene.

Functional Module	Key Actions and Feature Points
Room Management	Create a room, enter a room, exit a room, and destroy a room.
Seat Management	Request to speak, become a listener, invite a listener to speak, remove a speaker, and mute a speaker.
Product Management	Product list management, product pop-up management, and jump and payment.

©2013-2022 Tencent Cloud. All rights reserved.

Audio and Video Management	Local streaming, remote pull-streaming, and audience Mic connection.
On-Cloud Recording	TRTC on-cloud recording.

The overall business architecture of the live streaming with goods scene is shown in the figure below. The speaker creates a room, and other users can join the rooms they are interested in. Upon entering the room, users off the microphone can join the microphone to interact with the speaker via audio and video. The speaker is also responsible for maintaining the product list, and explaining and publishing products. Typically, for compliance requirements, the audio and video content in the live streaming room needs to be recorded and reviewed.



Note:

The solution shown here is the RTC real-time interactive live streaming solution. In actual applications, the RTC CDN live streaming solution may also be used. For details, please refer to Alternative Solutions.

©2013-2022 Tencent Cloud. All rights reserved.

Room Management

The Room Management Module is primarily responsible for maintaining the room list and includes the following features:
Create Room: After users log in to the business system, they can create a room. The room list needs to be updated after a room is created.
Enter a Room: Users can choose to enter an existing room. Upon entering, the current list of room members should be updated.
Exit a Room: Users can choose to exit the current room. Upon exiting, the current list of room members needs to be updated with a delete operation.
Destroy a Room: After all users exit the room, it needs to be destroyed. Upon destruction, the room list needs to be updated with a delete operation.
Note:
Room Management is a necessary module for implementing live shopping, but it is not the main functional module. It can be implemented in conjunction with the business system and IM&TRTC SDK, see Voice Chat Room - Room Management for details.

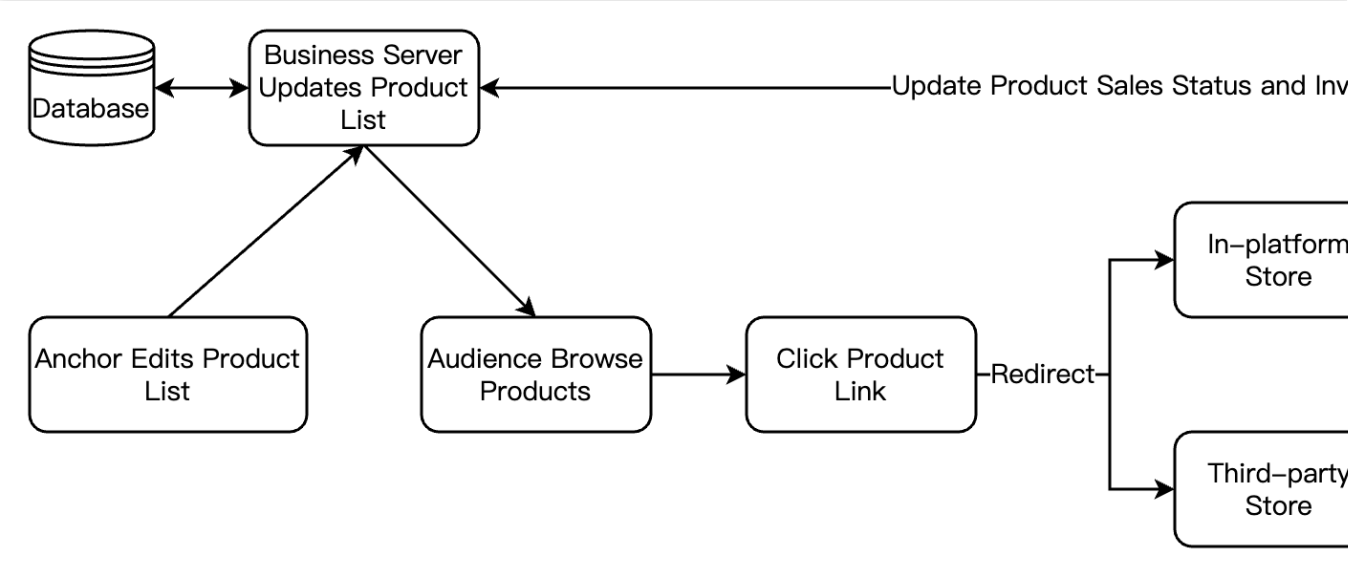
Seat Management

Seats in a live streaming room are generally ordered and limited. Seat Management is primarily responsible for defining the number of seats in a room based on the business scene, as well as managing the status of all seats in the current room. Seat Management mainly includes: request to speak, become a listener, invite a listener to speak, remove a speaker, and mute a speaker.
After users enter a room, only idle seats can be applied for.
After the anchor approves a user to become a speaker, the seat status needs to be changed to non-idle.
After becoming a listener, the co-anchoring user needs to stop local streaming and reset the seat status.
The anchor has the authority to lock the seat, invite a listener to speak, remove a speaker, mute a speaker, etc.
Note:
Seat Management is a necessary module for implementing live shopping, but it is not the main functional module. It can be implemented in conjunction with the business system and IM&TRTC SDK, see Voice Chat Room - Seat Management for details.

Product Management

The Product Management Module is unique to live shopping scenes and generally includes product list management, product pop-up management, product link jump, and payment. The following image displays the basic process of product management:

©2013-2022 Tencent Cloud. All rights reserved.



Product List Management

Product list management is the basic feature of product management, mainly including the addition, deletion, modification, and query of products. Usually, we store various information about products in the backend database, such as product name, description, price, inventory, and images. On the frontend, we can obtain this information through APIs and display it to users in the form of a list.

Product Pop-up Management

During the process of live streaming with goods, as the anchor talks about and lists products, it's often necessary to pop up corresponding product information on the audience's end to prompt them to browse and purchase. The product information pop-up feature can be achieved in the following two ways. You can choose one based on your business needs:

Custom Message

Generally, product information pop-ups can be achieved by sending custom messages to the live streaming room, and parsed and displayed after the audience in the live streaming room receives the custom messages. The sending and receiving of custom messages can be implemented by the business side or through Tencent Cloud Instant Messaging (IM) Group Messages. For specific implementation, see Product Information Pop-up - Custom Messages.

SEI Information

SEI (Supplemental Enhancement Information) provides a method to add additional information to a video stream. You can use Tencent Real-Time Communication (TRTC) Sending SEI Information to insert specified product information into the anchor's video stream. The audience in the live streaming room who are watching the stream can receive SEI messages, which will then be parsed and displayed. Based on the characteristics of SEI, this method can achieve precise synchronization between product information pop-ups and the anchor's live streaming screen. For specific implementation, see Product Information Pop-up - SEI Information.

Product Link Jump and Payment

After selecting products in the live streaming room, the audience needs to click on the product link, and jump to the specific E-commerce shop for order confirmation and payment. The E-commerce shop here can be an in-platform

©2013-2022 Tencent Cloud. All rights reserved.

store or an integrated third-party platform store. After the user completes the payment, we also need to obtain the payment result to update the sales status and inventory information of the product.

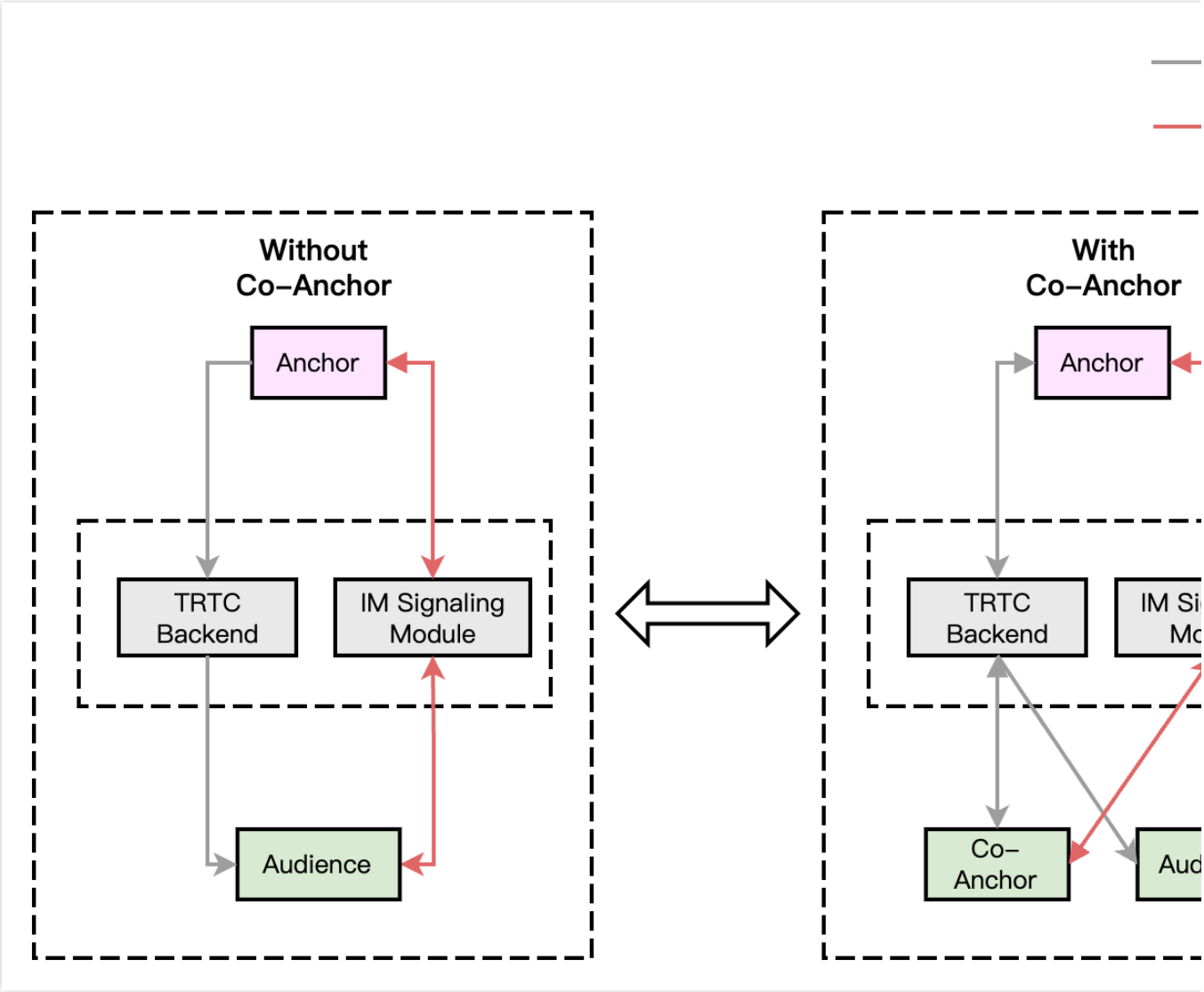
Note:

The above product management module is for reference only. In actual applications, you need to design and deploy according to your business needs.

Audio and Video Management

For standard live shopping scenes (audience size not exceeding 100,000 people), we recommend the **RTC Real-time Interaction Solution**: both the anchor and the audience use the RTC protocol for publishing/playback, minimizing end-to-end delay and ensuring a smoother experience for the audience when joining and leaving the mic, without abrupt changes such as image fast-forwarding or rewinding. Taking the example of multi-person co-anchoring interactive live streaming, the main architecture of pure RTC publishing/playback in a live shopping scene is as shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



The overall process of this solution architecture is as follows:

1. Both the anchor and audience connect through the signaling module, which is mainly responsible for controlling the live streaming process and synchronizing the live streaming status.
2. Regardless of whether there are mic-connecting audiences or not, both the anchor and the audience use the TRTC audio and video cloud service for publishing/playback.
3. After the audience requests to mic connect with the anchor, the signaling module will notify the anchor and synchronize the personal information of the co-speakers.
4. Once the anchor accepts the mic connection request, the mic-connecting audience starts streaming, and all members in the room receive stream update notifications and pull the audio-video stream of the mic-connecting audience.
5. When a mic-connecting audience member requests to disconnect, they stop streaming. All members in the room will receive stream update notifications and stop pulling that audience's audio and video stream.

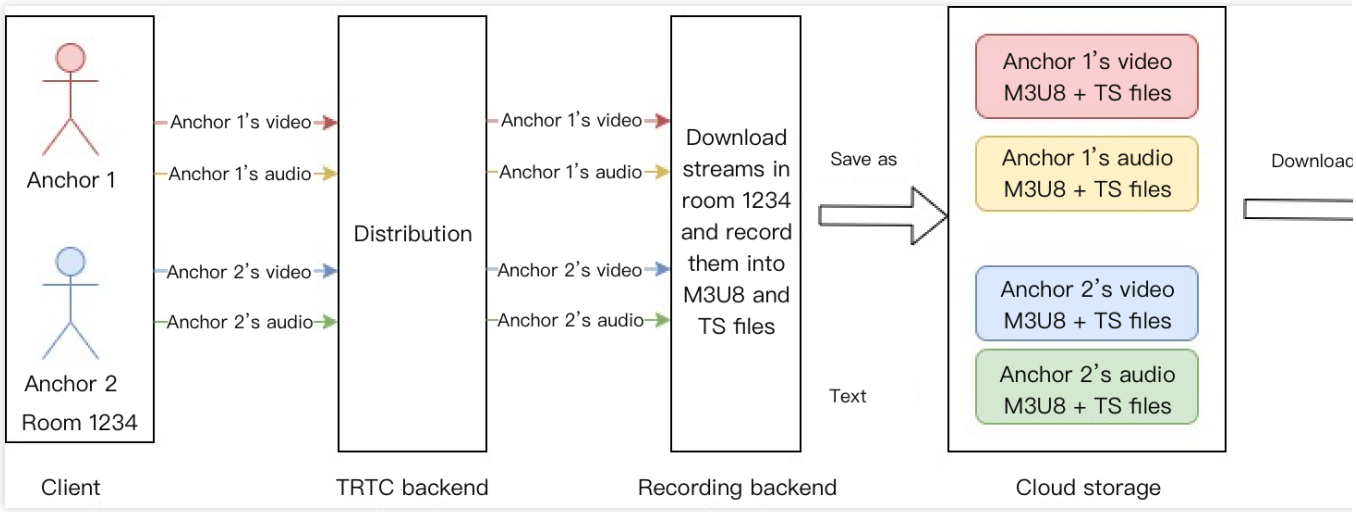
Note:

©2013-2022 Tencent Cloud. All rights reserved.

The signaling module can be a custom-developed signaling channel, and it is also recommended to use Instant Messaging (IM) for signaling interaction.
For large-scale live shopping scenes (with over 100,000 audience members in a single room), it is necessary to use the RTC CDN live streaming solution. For details, see Alternative Solutions.

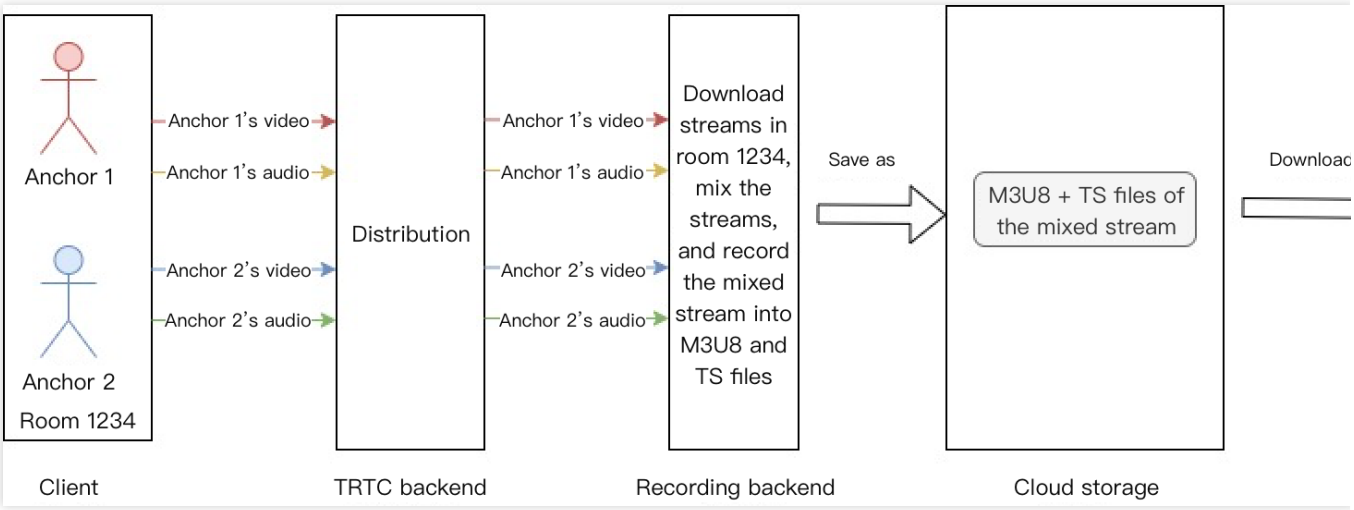
On-Cloud Recording

TRTC's newly upgraded on-cloud recording does not depend on cloud streaming services. It does not require a relayed push for cloud live streaming and uses TRTC's internal real-time recording cluster for audio and video recording, offering a more comprehensive and unified recording experience.
Single Stream Recording: With TRTC's on-cloud recording feature, you can record the audio and video stream of each user in the room into separate files.



Mix Stream Recording: Record the audio-video media streams of the same room as a single file.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

For a detailed introduction and activation guide to TRTC On-Cloud Recording, see On-Cloud Recording.

Key Business Logic

Product Explanation Replay

Product Explanation Replay is an essential feature for live shopping scenes, which is generally divided into In-live Replay and Post-live Replay. Product Explanation Replay helps latecomers and users who missed the live streaming to independently review the product explanation given during the live streaming by the anchor, thus increasing sales volume and improving the conversion rate. The Product Explanation Replay feature can be implemented in the following two ways.

Recorded Replay

Recorded Replay is a common way to implement the Product Explanation Replay feature, which is simple to implement and does not limit the timing of the replay. Below is the basic process for implementing Product Explanation Replay using Recorded Replay.

1. **Record Explanation Material**

Specify Recording Mode (RecordMode)

Single Stream Recording: Record each anchor's audio and video stream in the room into separate audio and video files and upload them to the cloud storage platform.

Mixed Stream Recording: Mix all the audio and video streams of all anchors you subscribe to in the room into one audio and video file and upload it to the cloud storage platform.

Specify Storage Location and Recording Format (StorageParams)

Storage Location: Supports storage to Cloud Video on Demand (VOD) or Cloud Object Storage (COS). You can specify CloudStorage (COS parameters) or CloudVod (VOD parameters) through the StorageParams parameter; it

©2013-2022 Tencent Cloud. All rights reserved.

does not support setting both VOD and COS simultaneously.

Recording Format: When the file is stored in COS, the default recording format is HLS; you can modify the recorded file format through the OutputFormat parameter under RecordParams. When the file is stored in VOD, the default recording format is MP4; you can modify the recorded file format through the MediaType parameter under TencentVod.

Start Recording Task (CreateCloudRecording)

To start on-cloud recording, call the REST API (CreateCloudRecording) through your backend service. Pay special attention to the parameter **Task ID (TaskId)**; this parameter is the unique identifier for this recording task. You need to save this Task ID as it will be required for subsequent operations related to this recording task.

Note:

In the CreateCloudRecording API to initiate on-cloud recording tasks, you need to specify the parameters UserId and UserSig needed for assigning a recording Chatbot to enter the room (How to Obtain UserSig). Please ensure that the UserId is not duplicated with those of the regular anchors or audience in your room and does not match the UserId of a recording Chatbot already assigned to a room in the midst of recording; otherwise, it will lead to the failure of the recording task.

Stop Recording Task (DeleteCloudRecording)

You can timely stop on-cloud recording tasks by calling the REST API (DeleteCloudRecording) through your backend service, requiring the Task ID (TaskId) parameter returned when starting the recording task.

2. **Obtain Playback Address**

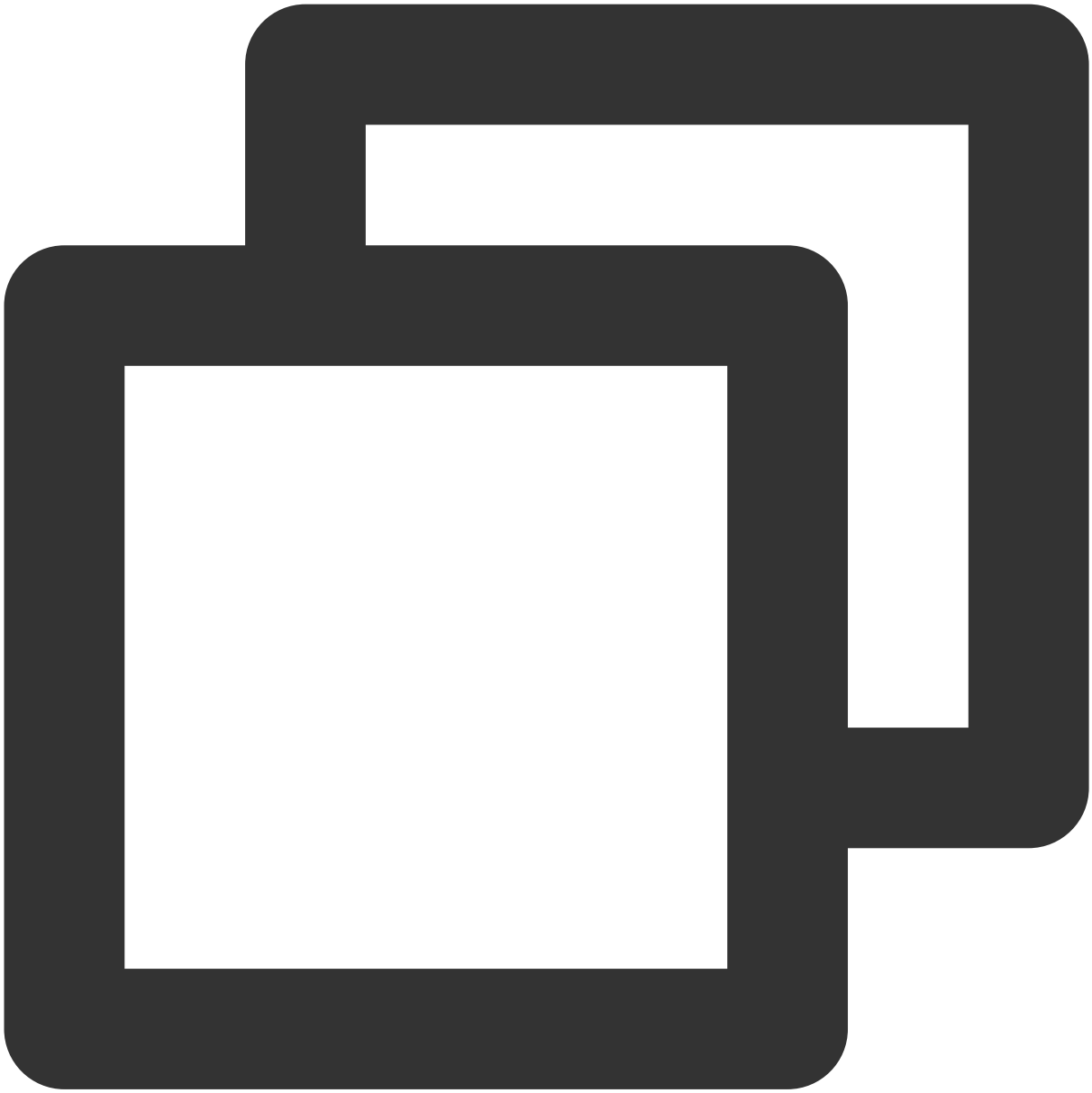
Method 1: Manual Search

After the recording task ends, the files recorded by the TRTC recording system will be uploaded to the cloud storage platform specified by you (Cloud Video on Demand (VOD) or Cloud Object Storage (COS)). You can directly go to the VOD Console or COS Console to locate the desired recorded media files and manually obtain the playback address.

Method 2: Callback Reception

You can also configure the recording callback URL in the console, allowing the cloud platform to proactively push the message of new recording files to your server. After the recording file is successfully transferred, the cloud platform will send a notification to your server through the callback address (HTTP/HTTPS) set in the console. It will push the recording and related events to your server through the callback address you set. You can receive the playback address **VideoUrl** of the recording file by accepting the upload success callback with **an event type of 311**. The callback example information is as follows:

©2013-2022 Tencent Cloud. All rights reserved.




```
{
  "EventGroupId": 3,
  "EventType": 311,
  "CallbackTs": 1622191965320,
  "EventInfo": {
    "RoomId": "20015",
    "EventTs": 1622191965,
    "UserId": "xx",
    "TaskId": "xx",
    "Payload": {
      "Status": 0,
```


©2013-2022 Tencent Cloud. All rights reserved.

```
        "TencentVod": {
          "UserId": "xx",
          "TrackType": "audio_video",
          "MediaId": "main",
          "FileId": "xxxx",
          "VideoUrl": "http://xxxx",
          "CacheFile": "xxxx.mp4",
          "StartTimeStamp": xxxx,
          "EndTimeStamp": xxxx
        }
      }
    }
  }
```

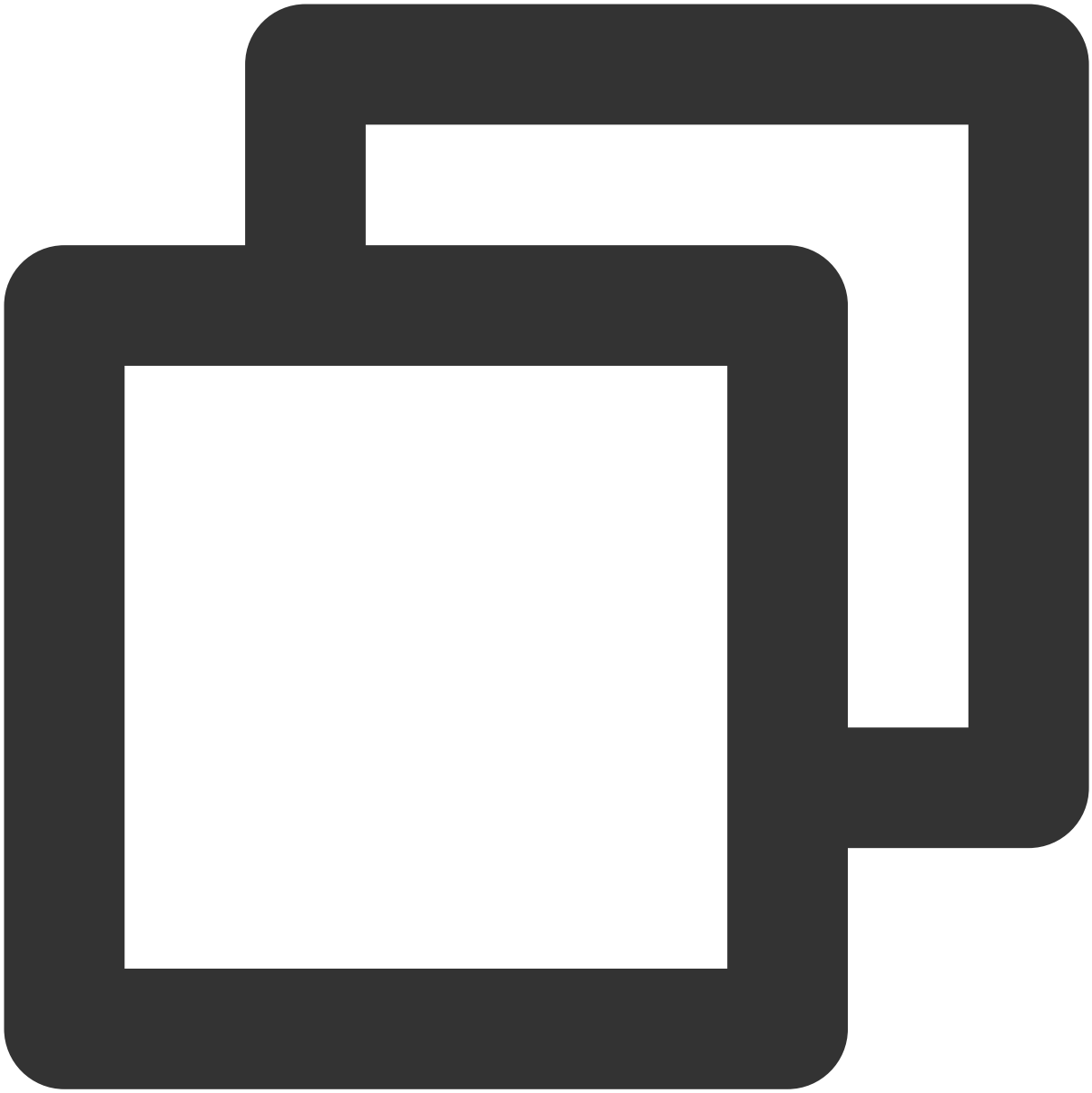
3. **Play Recorded Video**

After completing the preliminary preparations, you can call `startVodPlay` of TXVodPlayer to play the recorded product explanation video. TXVodPlayer will automatically recognize the playback protocol, and you only need to pass the playback URL to the `startVodPlay` function. For more code examples, please see Quick Integration Guide - Product Explanation Replay.

Android

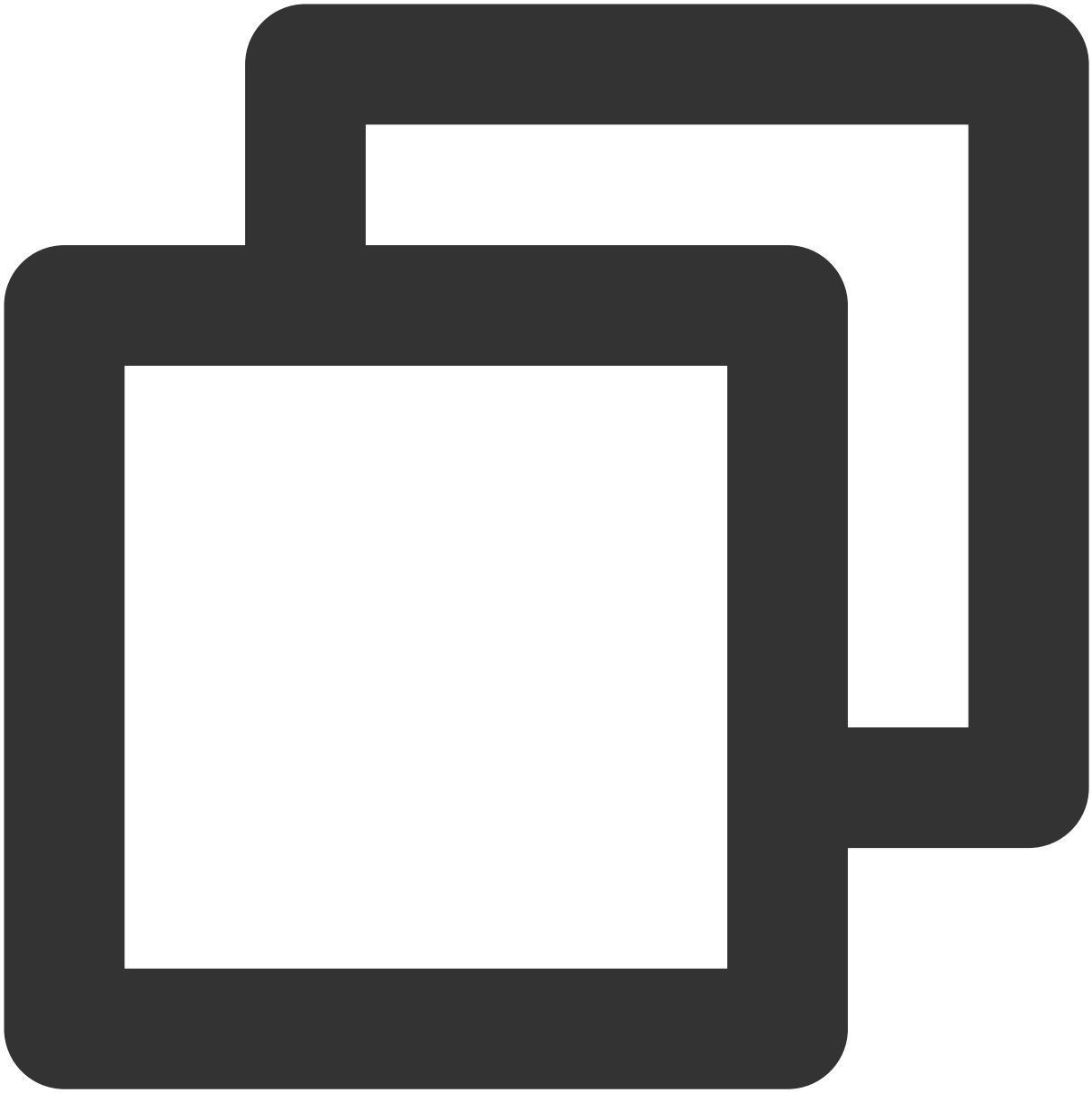
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Play URL video resource
String url = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
mVodPlayer.startVodPlay(url);
```

©2013-2022 Tencent Cloud. All rights reserved.



```
// Play URL video resource
NSString* url = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
[_txVodPlayer startVodPlay:url];
```

Note:

For on-demand playback, the Player SDK is needed, but there's no need for separate integration, and you just need to integrate the full feature version of LiteAVSDK, as detailed in Quick Access Guide - Import SDK.

Live Streaming Time Shift

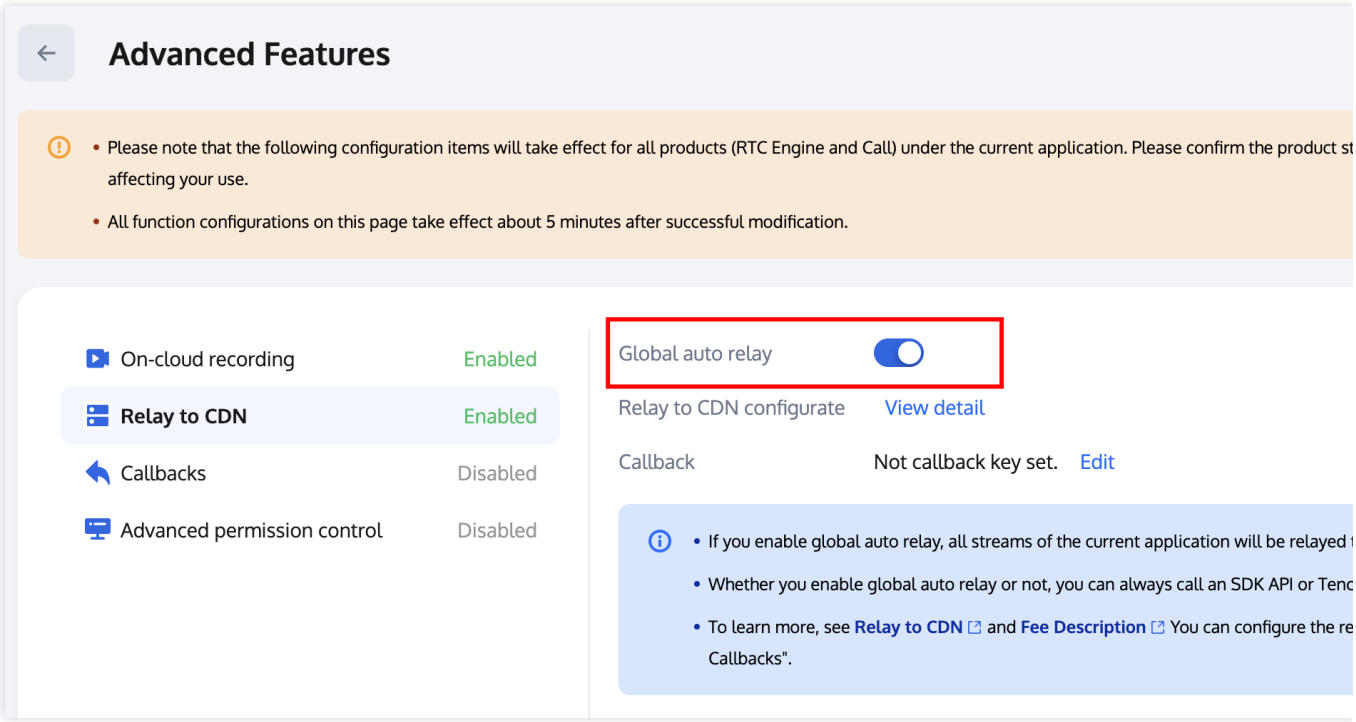
©2013-2022 Tencent Cloud. All rights reserved.

Live Streaming Time Shift can also achieve the product explanation replay feature, but it only supports revisiting the product explanation during the live streaming and does not support replay after the live streaming. Below is the basic process of using Live Streaming Time Shift for product explanation replay.

1. **Enable Relayed Push**

Global Automatic Relayed Push

If you need to automatically relay all anchors' audio and video streams in the room to live streaming CDN, you need to enable **Relay to CDN** in the TRTC console **Advanced Features** page.

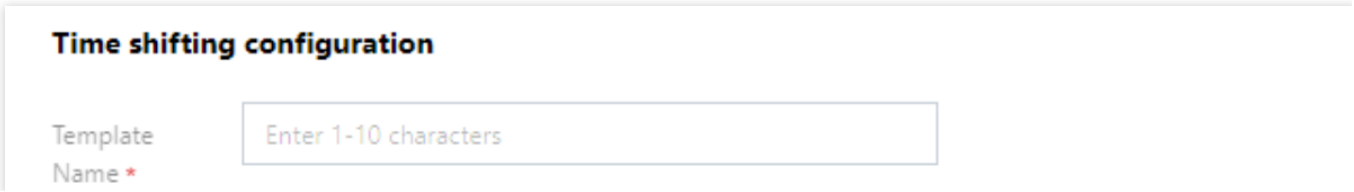


Relayed Push of the Specified Streams

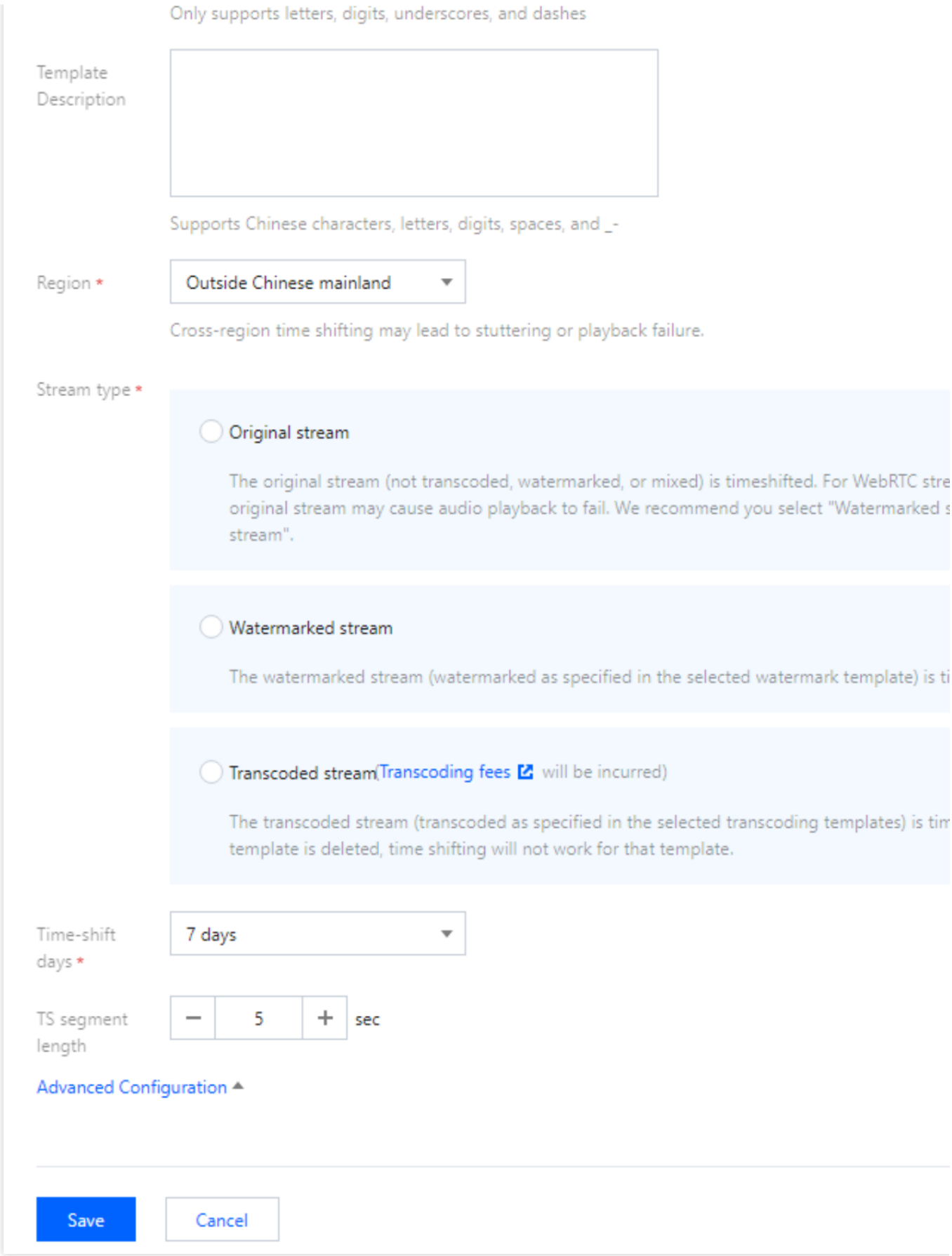
If you need to manually specify the audio and video streams to be published to live streaming CDN, or publish the mixed audio and video streams to live streaming CDN, you can do so by calling the startPublishMediaStream API. In this case, you do not need to activate global automatically relaying to CDN in the console. For a detailed introduction, see Publish Audio and Video Streams to Live Streaming CDN.

2. **Create Time Shift Template**

Go to CSS Console - Feature Configuration - Live Streaming Time Shift, click **Create Template**, fill in the relevant parameters and click **Save**. Finally, follow the prompts to **Bind Push Domain Name**.



©2013-2022 Tencent Cloud. All rights reserved.



3. **Construct Play Request**

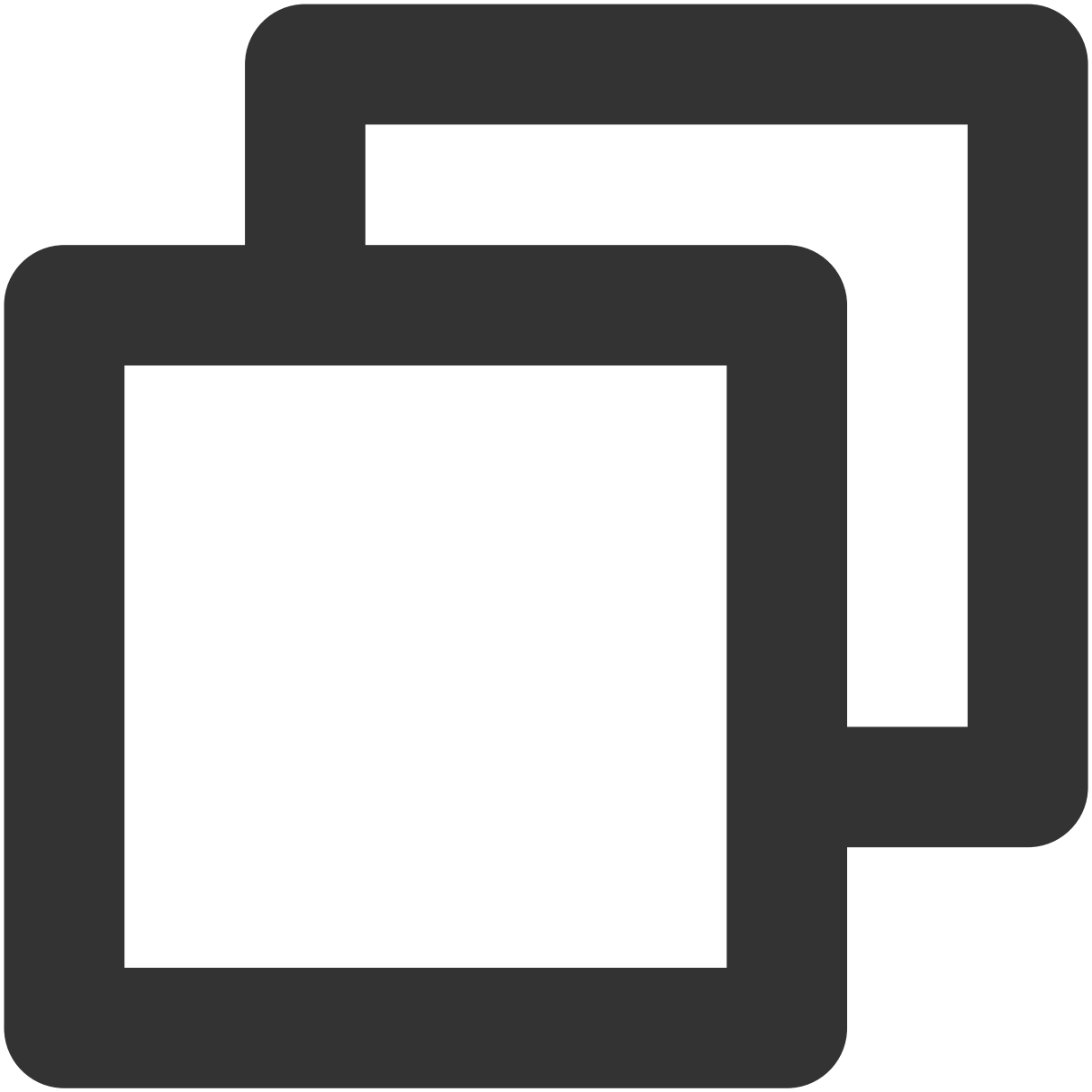
©2013-2022 Tencent Cloud. All rights reserved.

Live Streaming Time Shift can only be distributed through the HLS protocol. Specify the time to revisit in the M3U8 request parameters, which is the time segment when the anchor explains the product.
The general HLS live streaming address format is `http://domain/appname/stream.m3u8`. To support the time shift playback, time shift parameters need to be appended to this address, as shown in the table below:

Field Name	Meaning	Required or Not	Example
txTimeshift	Value on: enable new live streaming time shift.	Yes	txTimeshift=on
tsStart	Time shift start time, the interval between tsStart and tsEnd cannot be less than the duration of one Ts shard and cannot be more than 6 hours.	Yes	tsStart=20121010010101
tsEnd	Time shift end time, the interval between tsStart and tsEnd cannot be less than the duration of one Ts shard and cannot be more than 6 hours.	Yes	tsEnd=20121010010102
tsFormat	The value format for tsStart and tsEnd is `{timeformat}_{unit}_{zone}`. timeformat value: unix: unix timestamp. If 'unix' is selected, the 'zone' part can be ignored. human: human-readable time, 20121010010101 unit: s/ms, unit in seconds or milliseconds zone: Time zones are divided into East and West Zones: The range for the East Zone is 1 ~ 12. The range for the West Zone is -12 ~ -1.	Yes	tsFormat=unix_s tsFormat=human_s_8
tsCodecname	For transcoding streams, it's necessary to specify the template name. Original streams and watermark streams do not carry this field.	No	tsCodecname=hd

Below are examples of playback requests for both time formats:

©2013-2022 Tencent Cloud. All rights reserved.



```
// Playback request with unix-format time
http://example.domain.com/live/stream.m3u8?txTimeshift=on&tsFormat=unix_s&tsStart=1

// Playback request with human-format time
http://example.domain.com/live/stream.m3u8?txTimeshift=on&tsFormat=human_s_8&tsStar
```

Note:

The Live Streaming Time Shift feature is a paid value-added service. Using the Live Streaming Time Shift feature will generate a time-shift bill, see Billing Documentation for billing rules.

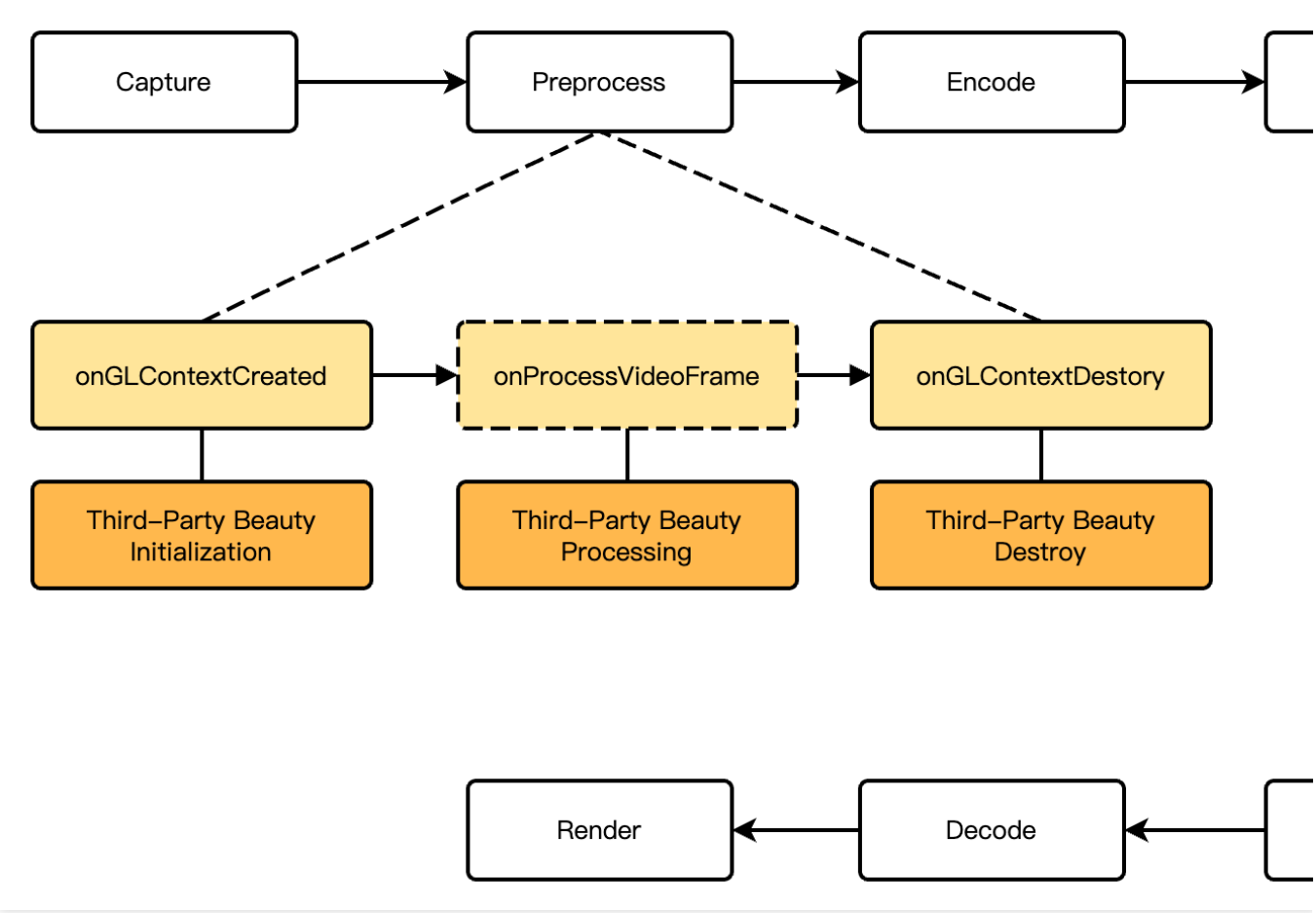
©2013-2022 Tencent Cloud. All rights reserved.

The Live Streaming Time Shift feature is only applicable to RTC CDN live streaming solutions. For more details on implementing the Live Streaming Time Shift feature, see Live Streaming Time Shift.

Integrating Beauty Effect

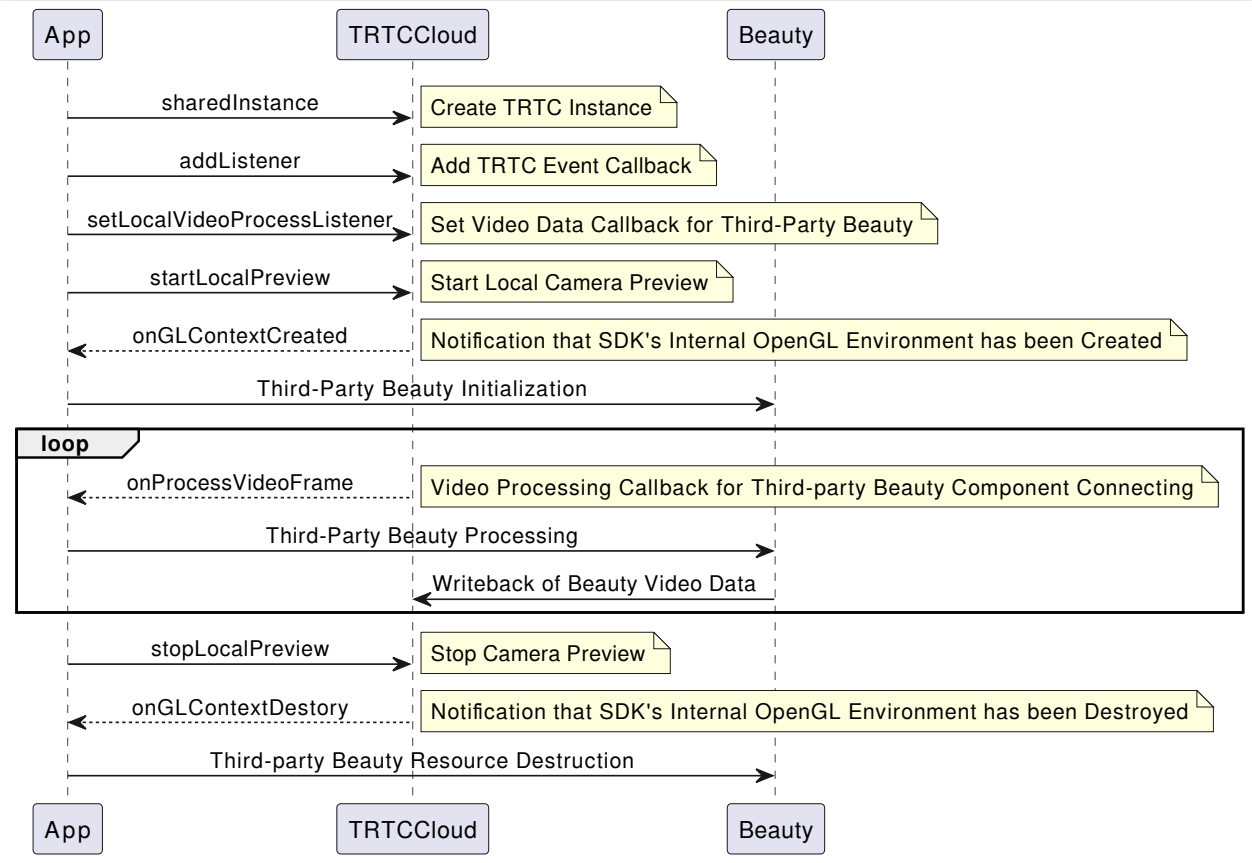
In the live shopping scene, the beauty effect is also a frequently used feature. It not only improves the beauty of the anchor but also adds fun to live interaction through various sticker effects. TRTC supports the integration of Tencent Effect SDK as well as the integration of mainstream third-party beauty effect products in the market, such as Volcano Beauty, and FaceUnity.

Beauty Effect Integration Process



API Call Sequence

©2013-2022 Tencent Cloud. All rights reserved.



Comparison of Beauty Products

Beauty Type	Beauty Effect	Integration Cost	Fees	Virtual AI Digital Human	Support Terminal
Tencent Effect SDK	The basic effect is good, advanced effect for big eyes/slim faces is significant.	Moderately Low	Moderate	Supported	Android/iOS/PC/Flutter/Web/Mini Program
FaceUnity Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are average.	Moderately High	Moderate	Supported	Android/iOS/PC/Untiy

©2013-2022 Tencent Cloud. All rights reserved.

Volcano Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are relatively good.	Moderately High	Relatively High	Supported	Android/iOS/PC/Linux

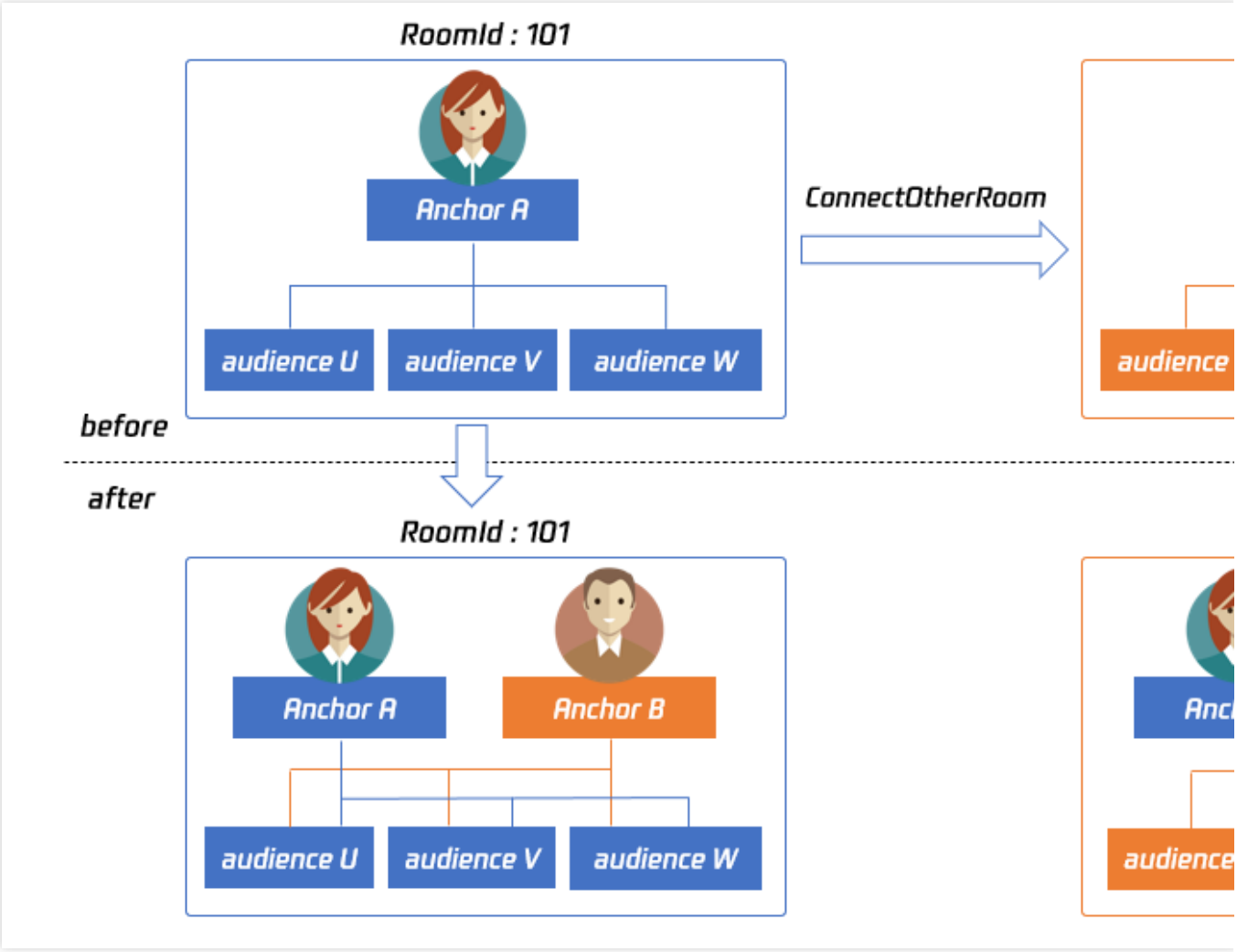
Cross-room Co-hosting Competition

Connecting anchors across rooms for cross-room competition is a novel approach in the live shopping scene. Interactive competition can enhance the entertainment value of live streaming and to some extent, and stimulate the audience's desire to shop. TRTC supports cross-room competition across multiple rooms and among multiple anchors. Below, we introduce specific implementation methods.

1. **How It Works**

By default, only users in the same room can have audio and video calls, and the audio and video streams between different rooms are isolated. Through the cross-room competition, the audio and video streams of an anchor in another room can be published in the current room, while the audio and video streams of the current anchor will also be published in the target anchor's room. This allows anchors in different rooms to share audio and video streams across rooms, enabling audiences in each room to watch the audio and video of both anchors.

©2013-2022 Tencent Cloud. All rights reserved.



The figure above shows the main process of cross-room competition. For example: After anchor A in room 101 establishes a cross-room call with anchor B in room 102 using `ConnectOtherRoom()` :

Users in room 101 will receive two event callbacks from anchor B: `onRemoteUserEnterRoom(B)` and `onUserVideoAvailable(B,true)` . Therefore, users in room 101 can subscribe to the audio and video of anchor B.

Users in room 102 will receive two event callbacks from anchor A: `onRemoteUserEnterRoom(A)` and `onUserVideoAvailable(A,true)` . Therefore, users in room 102 can subscribe to the audio and video of anchor A.

Note:

Both local and peer users participating in cross-room competition must be in the anchor role and must have audio/video uplink capabilities.

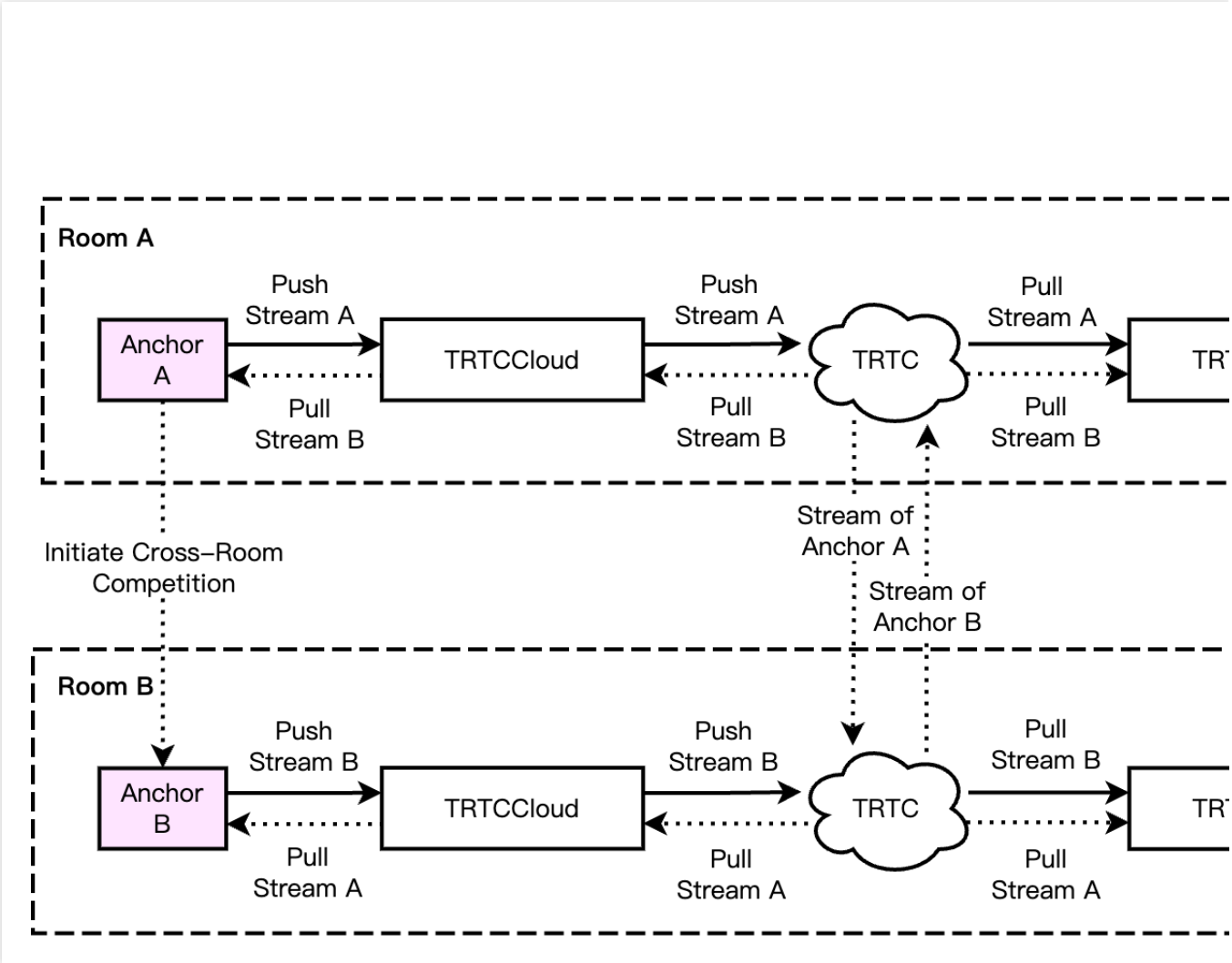
Cross-room competition with multiple room anchors can be achieved by calling `ConnectOtherRoom()` multiple times. Currently, a room can connect with up to three other room anchors at most, and up to 10 anchors in a room can conduct cross-room competition with anchors in other rooms.

©2013-2022 Tencent Cloud. All rights reserved.

TRTC cross-room competition can also be achieved by `createSubCloud()` to create a sub-instance and join the publishing/playback of another room. Currently, there is no limit to the number of sub-instances, facilitating the future expansion of business scenes involving PK between multiple rooms or anchors.

2. **Real-Time Interactive Cross-Room Competition Process**

RTC real-time interaction solution The cross-room competition process is straightforward. Anchors and cross-room competition anchors mutually pull RTC single streams, and the audience simultaneously pulls the RTC single streams of both anchors and cross-room competition anchors. The audience can independently control the subscription logic of the media streams of anchors and cross-room connecting anchors. The RTC real-time interactive cross-room competition process is shown in the figure below:



Note:

In real-time interactive cross-room competition scenes, audiences in the room can independently control the logic of subscribing to the media streams of cross-room connecting anchors, or it can be changed by the room owner to <1>change the uplink capability of a cross-room anchor in their rooms<1>.

©2013-2022 Tencent Cloud. All rights reserved.

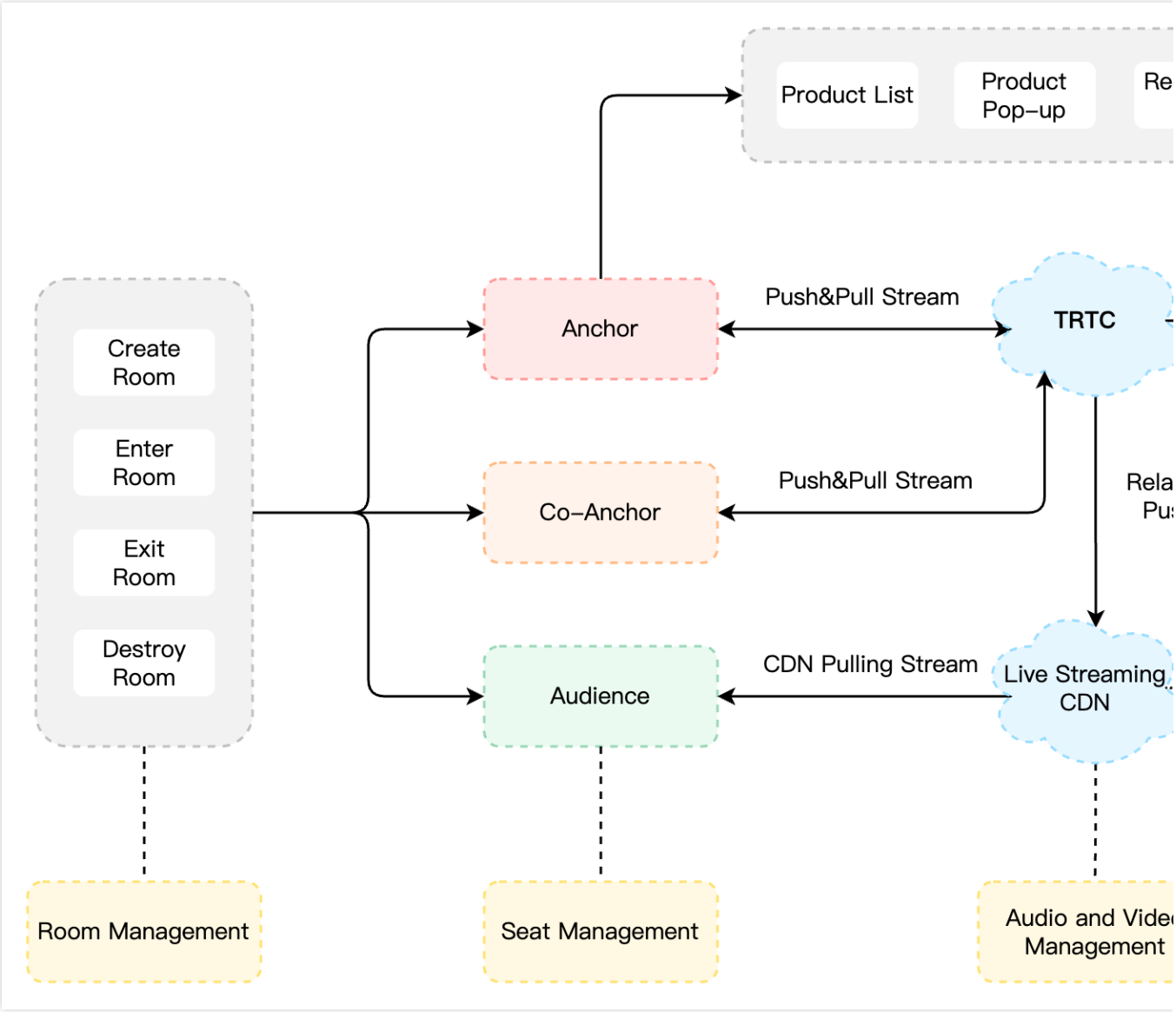
Alternative Solutions

Apart from the recommended RTC Real-time Interaction Solution in the scene implementation scheme, another alternative solution for the live shopping scene without limiting the audience size usually exists: RTC CDN Live Streaming Solution.

RTC CDN Live Streaming Solution

The anchor uses the TRTC protocol for publishing/playback and relayed push of Tencent Cloud Streaming Services or a third-party live streaming platform. General audiences pull the CDN stream to watch while mic-connecting audiences engage in interactive co-anchoring by switching to the TRTC protocol for streaming. This solution is a commonly used compromise, with a higher delay for both watching and mic connecting on the audience side. However, it offers advantages in terms of cost-effectiveness and audience scalability. The whole architecture is shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



The overall process of this solution is as follows:

1. The anchor enters the TRTC room, streams via the RTC protocol, and relays to Cloud Streaming Services.
2. Ordinary off-mic audiences pull CDN accelerated streams and watch through live streaming players.
3. Audiences can request to speak and become the mic-connecting audiences. They stop the CDN streaming and switch to the RTC protocol for publishing/playback.
4. Audiences off the mic become general audiences. They stop the RTC protocol publishing/playback and switch to the CDN streaming.

CDN stream playback addresses support multiple protocols such as RTMP/FLV/HLS/WebRTC, with splicing rules detailed in Splicing Playback URL.

Different live streaming playback protocols have varying compatible platforms, play delays, and billing rules. See the table below for details:

Live Streaming Protocol	Advantages	Disadvantages	PlayDelay

©2013-2022 Tencent Cloud. All rights reserved.

FLV	Mature, suited for high-concurrency scenes	Requires integration of SDK for playback	2s - 3s
RTMP	Relatively low delay	Poor performance in high-concurrency scenes	1s - 3s
HLS(M3U8)	High support on mobile browsers	Very high delay	10s - 30s
WebRTC	Lowest delay	Requires integration of SDK for playback	< 1s

Note:

Tencent Cloud Streaming Services supports multiple playback protocols. You can choose the appropriate pull stream solution based on your business needs. For example, for delay-sensitive businesses, LEB pulling stream is recommended.

HLS and WebRTC playback protocols support the adaptive bitrate feature, which allows smooth switching of playback bitrate under different network conditions. See Adaptive Bitrate for details.

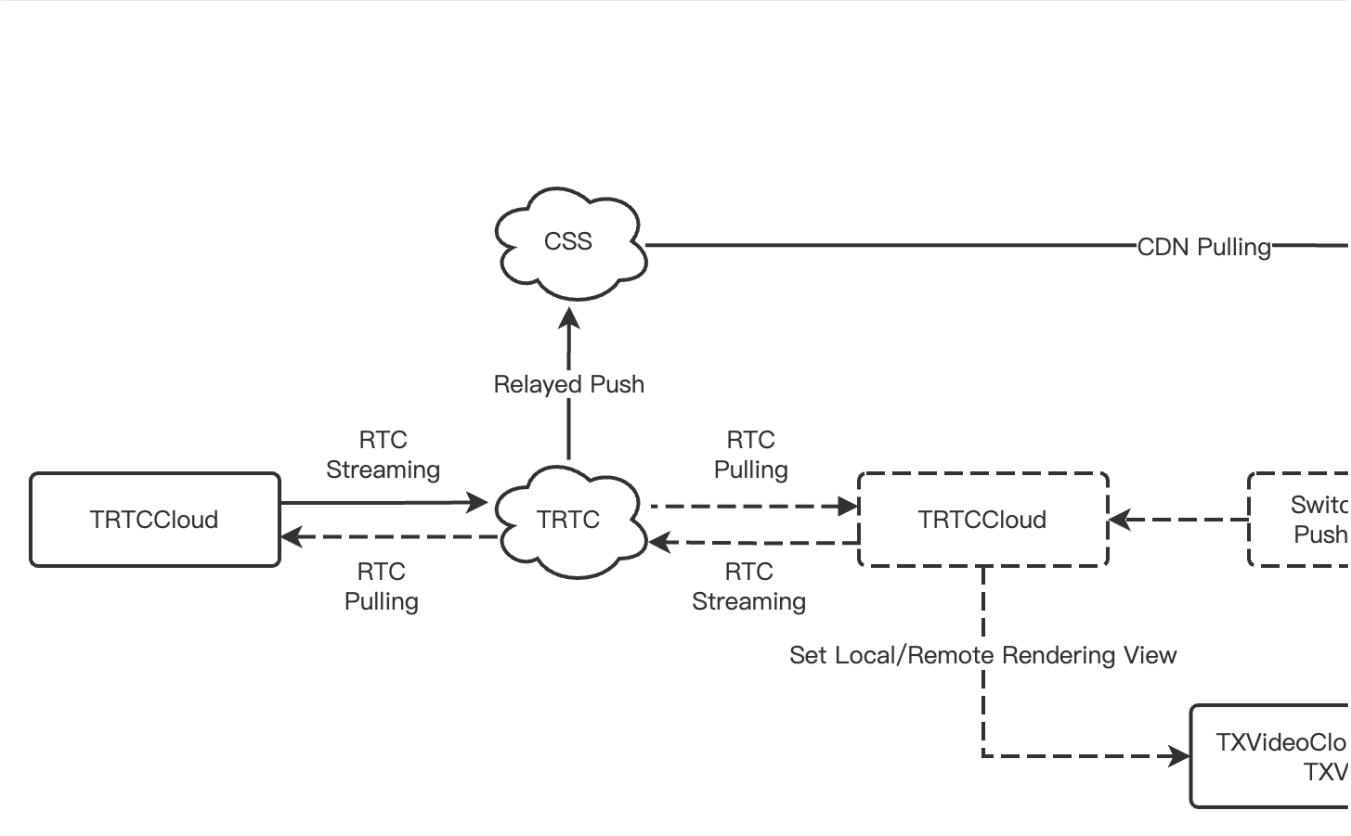
Smooth Mic On/Off Handling

In single-anchor low-frequency mic connection live streaming scenes, due to cost considerations, RTC CDN live streaming solutions or third-party live streaming publishing/playback solutions are often used. In single-anchor streaming, the anchor streams via RTC or third-party live streaming, while audiences pull streams via CDN. In interactive co-anchoring scenes, both the anchor and audiences stream via RTC. This involves switching between publishing/playback tools while maintaining a seamless experience for users. Below are the specific methods for smooth mic on/off handling in both the RTC CDN live streaming and third-party live streaming publishing/playback solutions.

1. RTC CDN Live Streaming Solution

Under the RTC CDN live streaming solution, the anchor always uses the TRTC SDK for publishing/playback. During mic connects, only the mic-connecting audience needs to switch the publishing/playback tools, and the rendering control can be reused.

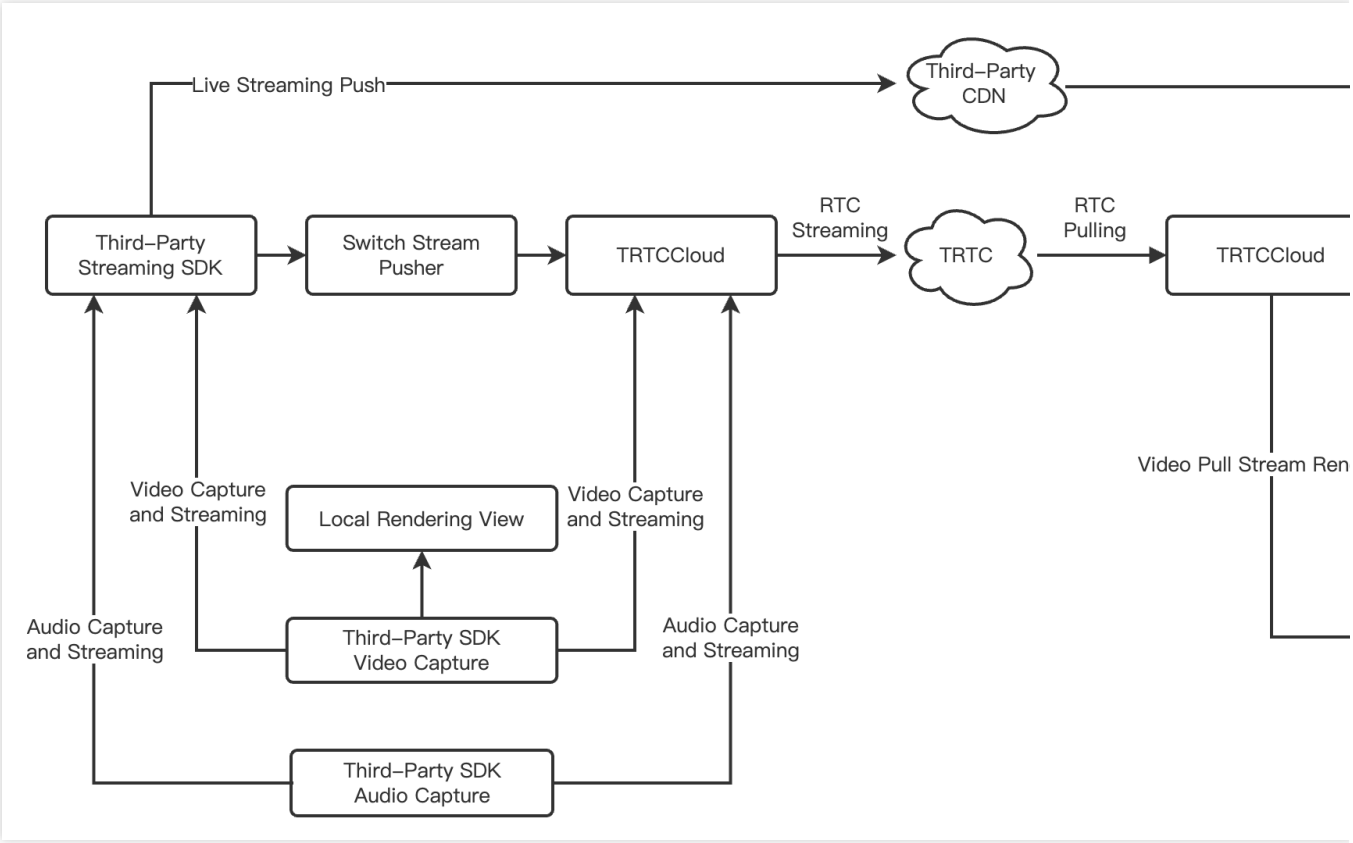
©2013-2022 Tencent Cloud. All rights reserved.



2. Third-Party Live Streaming Publishing/Playback Solution

In the third-party live streaming publishing/playback solution, during mic connects, both the anchor and the mic-connecting audience need to switch the publishing/playback control. It is recommended to use TRTC's custom capture and rendering features.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

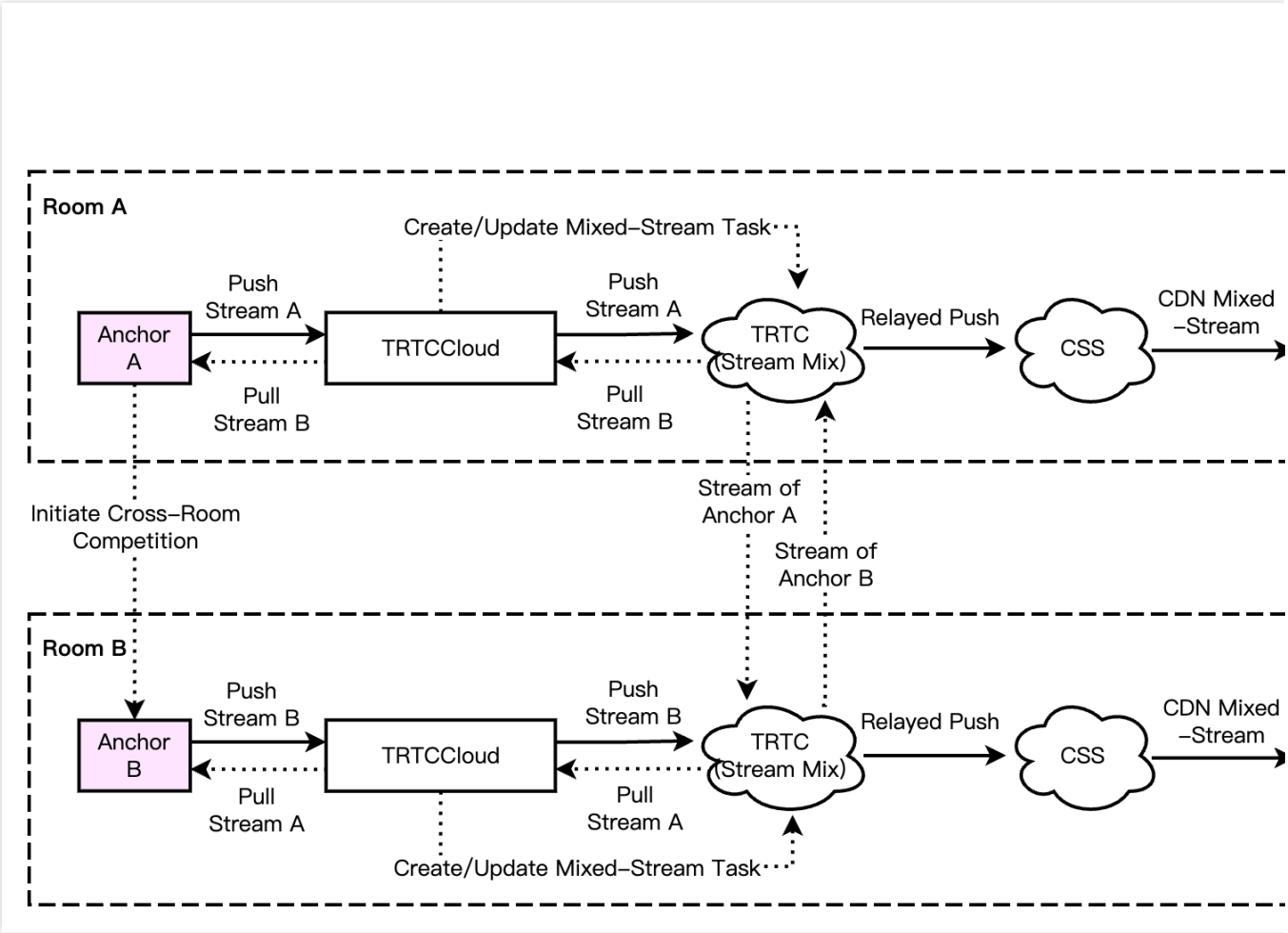
Smooth Mic on: To avoid screen interrupts when switching the stream puller, it is suggested to wait for the TRTC's first frame callback `onFirstVideoFrame` before stopping the CDN streaming.

Smooth Mic off: To avoid screen interrupts when switching the stream puller, it is suggested to wait for the video play event `onVideoPlaying` before stopping the RTC streaming.

CDN Live Streaming in Cross-Room Competition

In the RTC CDN live streaming scene, the process of cross-room competition is relatively complex. Anchors pull RTC single streams from each other and CDN audiences pull the mixed streams from both the anchor and the cross-room competition anchor. Audiences cannot independently control the subscription logic of the anchor and cross-room competition anchor's media streams. The process for cross-room competition in a CDN live streaming scene is shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



Note:

In the CDN live streaming in cross-room competition scene, CDN audiences cannot independently control the subscription logic of the cross-room connected anchor's media stream. It needs to be uniformly controlled by the anchor through updating and publishing the media stream.

Scene Approach

Single-anchor Live Streaming with Goods

Single-anchor live streaming with goods is the most common and basic approach in live shopping scenes. In such a scene, there is only one anchor in the live streaming room, and roles such as co-anchor do not exist. Audiences can enter the live streaming room to watch the live streaming, interact, and shop. The single-anchor live streaming with goods scene is recommended to use the RTC CDN Live Streaming Solution.

Multi-person Co-anchoring Interactive Live Streaming with Goods

Based on the single-anchor live streaming with goods scene, the multi-person co-anchoring interactive live streaming with goods scene provides the approach of inviting the audience to have a mic on for interaction with the anchor.

©2013-2022 Tencent Cloud. All rights reserved.

Anchors can invite the audience to take the mic and control the seats; audiences can also actively request to speak for interaction with the anchor. This approach enhances the audience's sense of participation and motivates the audience. This scene is recommended to use the RTC Real-time Interaction Solution.

Cross-room Competition and Mic Connection Live Streaming with Goods

Besides the traditional single-room anchor live streaming with goods, anchors can also engage in cross-room competition with another live streaming room's anchor to showcase their products, ignite the audience's purchasing enthusiasm, and increase the live streaming's entertainment value. Live streaming with goods in cross-room competition is a popular novel approach. In this scene, the RTC Real-time Interaction Solution or RTC CDN Live Streaming Solution can be selected based on business needs.

Supporting Products for the Solution

System Level	Product Name	Application Scenes
Access Layer	Tencent Real-Time Communication (TRTC)	Provides a low-delay, high-quality multi-person audio and video real-time interaction live streaming solution, which is a foundational capability for live shopping scenes.
Access Layer	Instant Messaging (IM)	Provides room management and seat management capabilities based on group features, enables the sending and receiving of rich media messages such as live streaming room-wide messaging, public screen messages, as well as custom signaling and other communication needs.
Access Layer	Tencent Effect SDK	Provides real-time effects processing capabilities such as beauty, filtering, makeup, fun stickers, emojis, and virtual avatars.
Cloud Services	Cloud Streaming Services (CSS)	Provides real-time audio and video relayed push, along with accelerated media stream distribution services, as well as additional capabilities such as recording and pornography detection.
Cloud Services	Video on Demand (VOD)	Catering to media such as audio, video, and images, it offers an integrated high-quality media service that includes creation, upload, storage, transcoding, media processing service, media AI, accelerated distribution and playback, and copyright protection.
Data Storage	Cloud Object Storage (COS)	Provides storage services for audio and video recording files, as well as audio and video slicing files.

©2013-2022 Tencent Cloud. All rights reserved.

Quick Integration Guide
Android

Last updated：2024-07-18 14:26:14

Business Process

This section summarizes some common business processes in the e-commerce live streaming scenario, helping you better understand the implementation process of the entire scenario.

Anchor starts and ends live broadcast

Anchor initiates the cross-room mic-connection PK

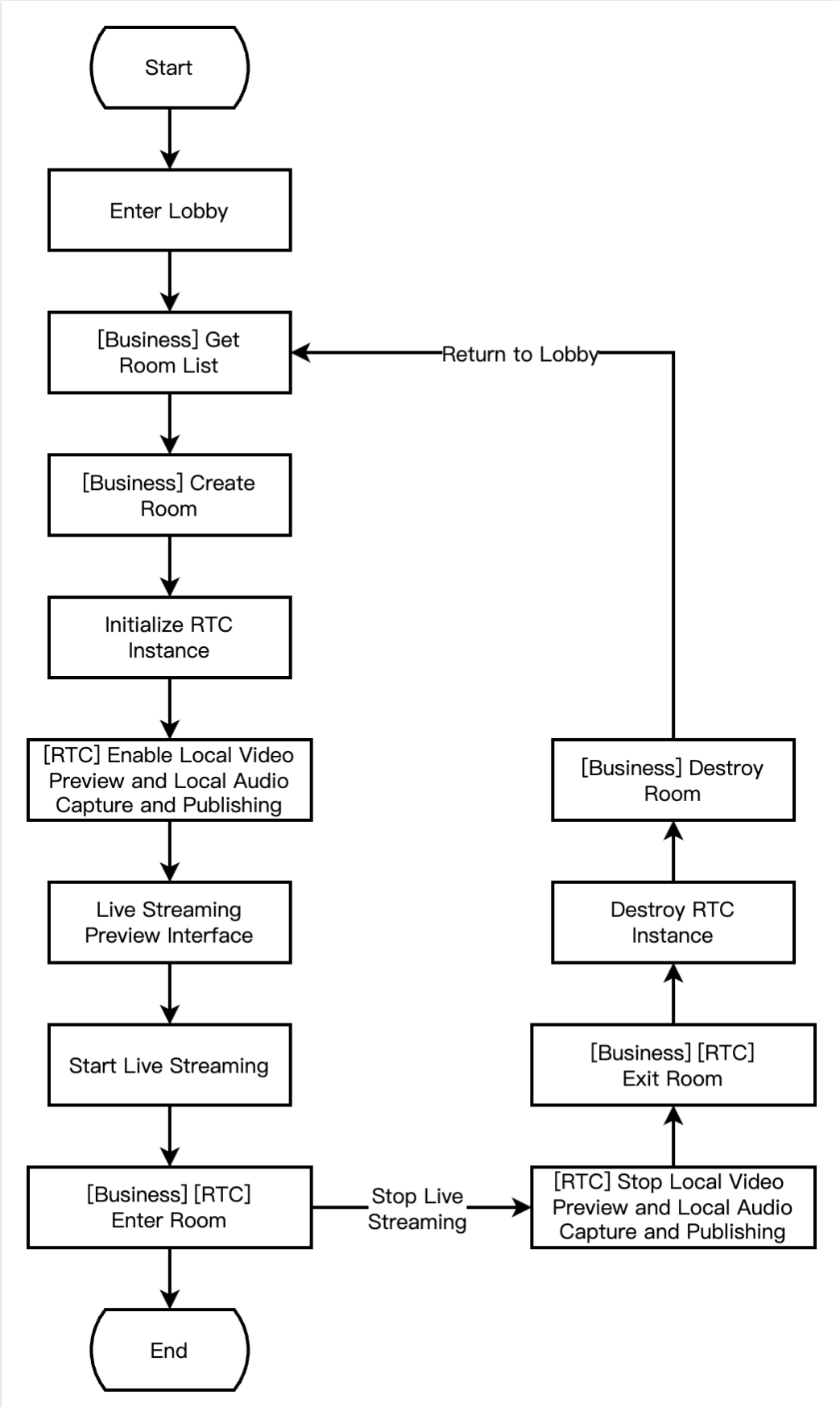
The RTC audience enters the room for mic-connection

The CDN audience enters the room for mic-connection

Product Management for Merchandising

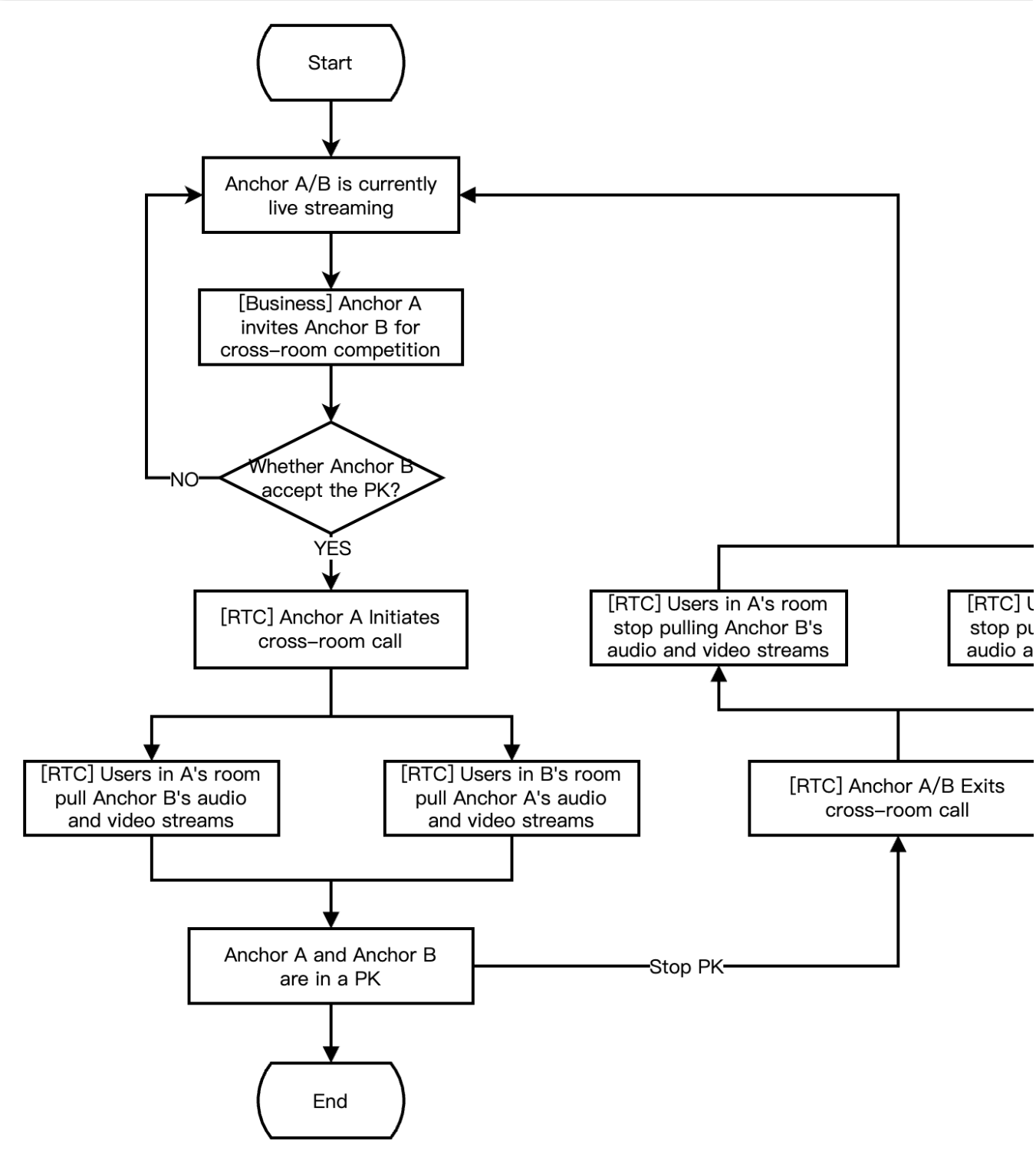
The following diagram shows the process of an anchor (room owner) local preview, creating a room, entering a room to start live streaming, and leaving the room to end the live streaming.

©2013-2022 Tencent Cloud. All rights reserved.



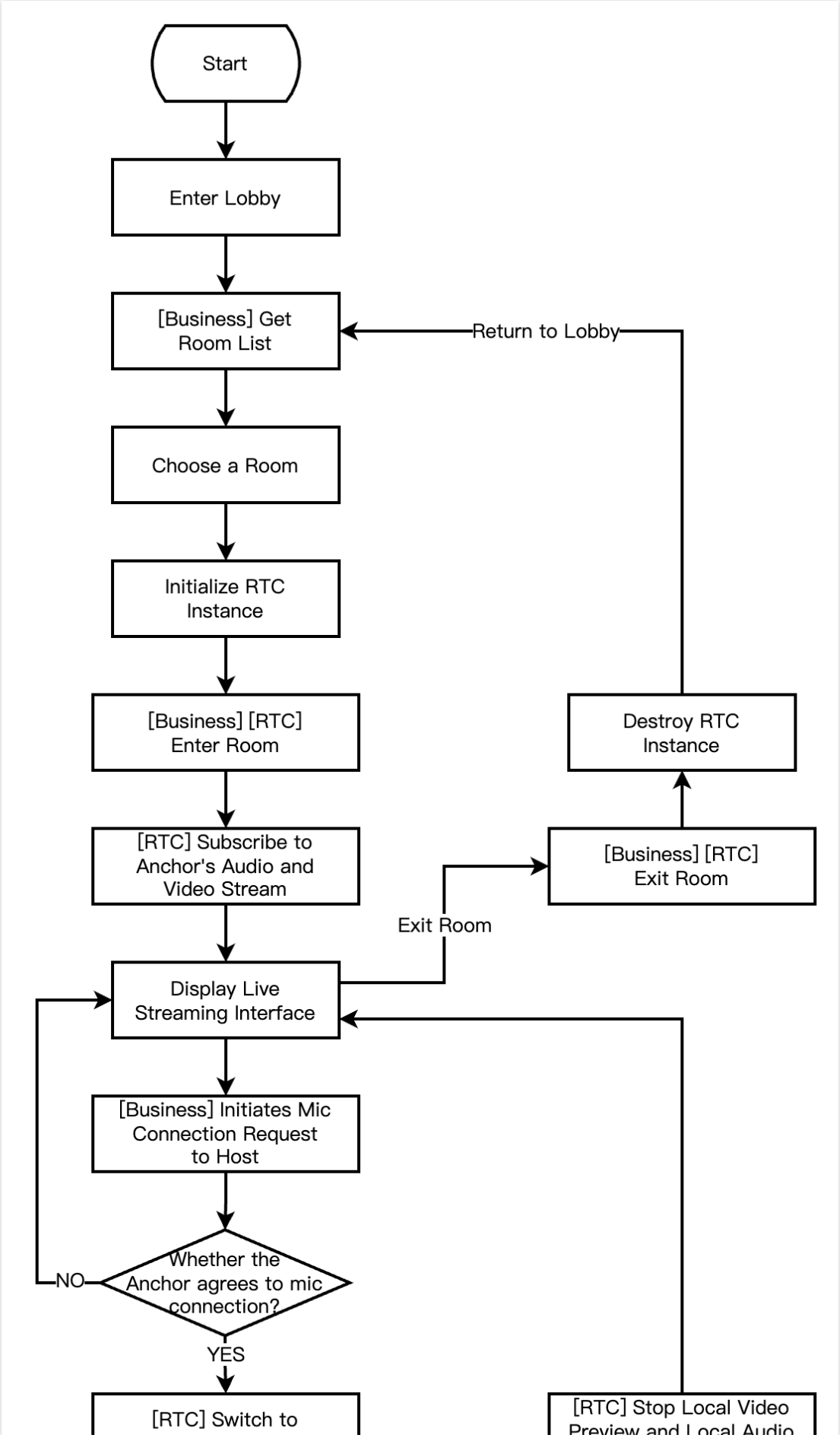
©2013-2022 Tencent Cloud. All rights reserved.

The following diagram shows the process of Anchor A inviting Anchor B for a cross-room PK. During the cross-room PK, the audiences in both rooms can see the PK mic-connection live streaming of the two room owners.

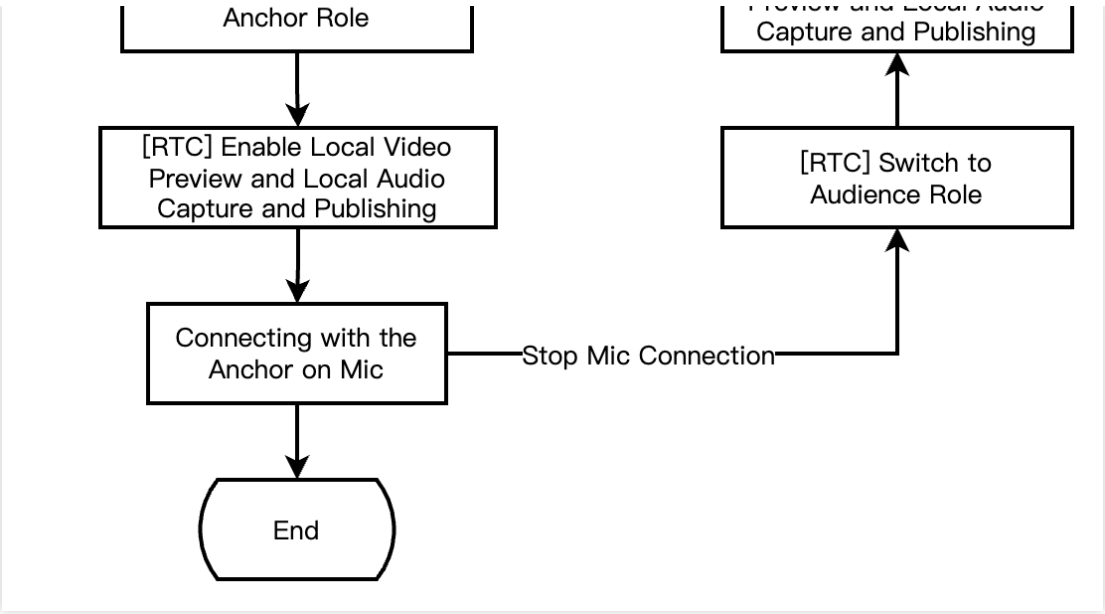


The following diagram shows the process for RTC live interactive streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

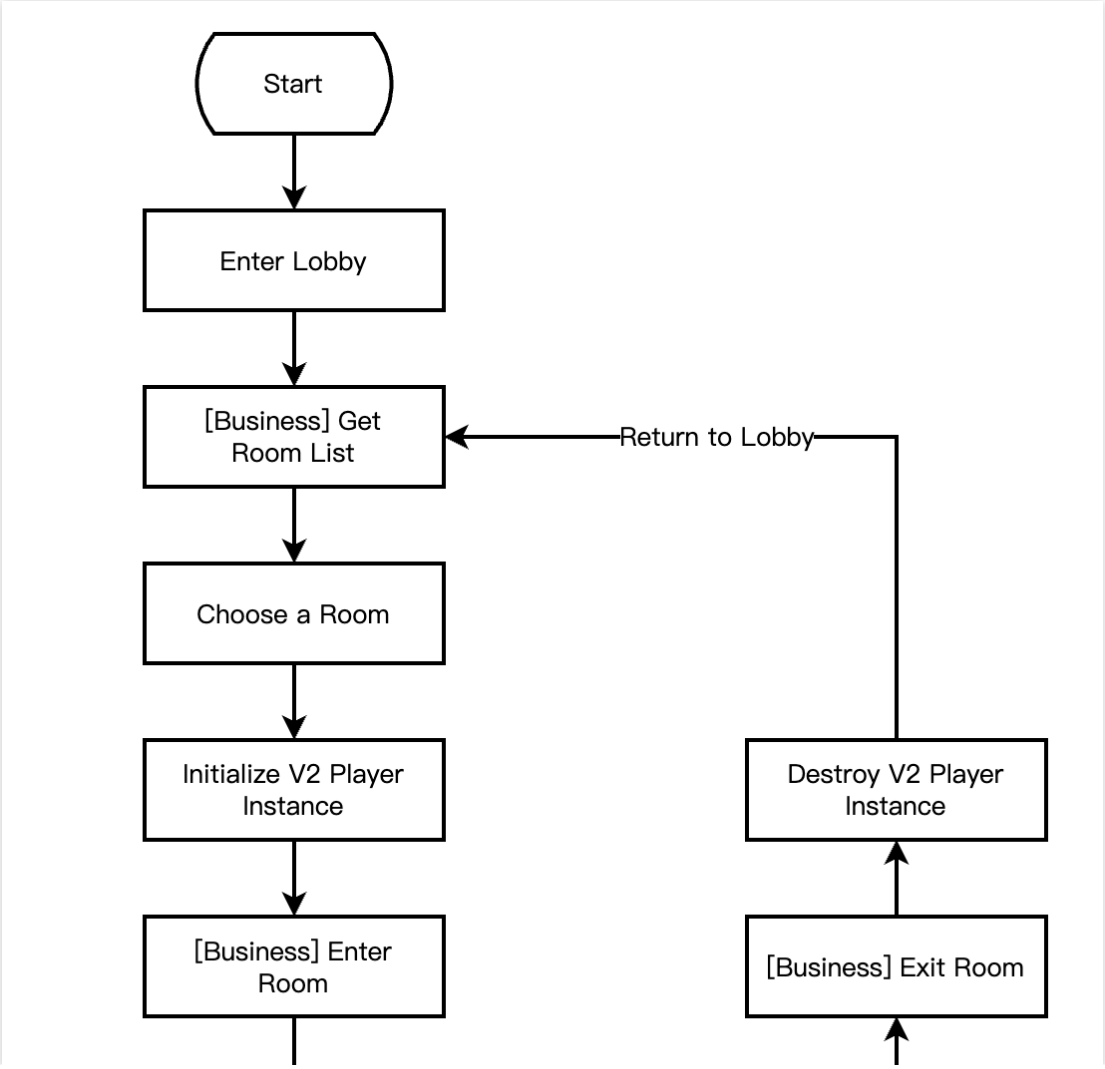
©2013-2022 Tencent Cloud. All rights reserved.



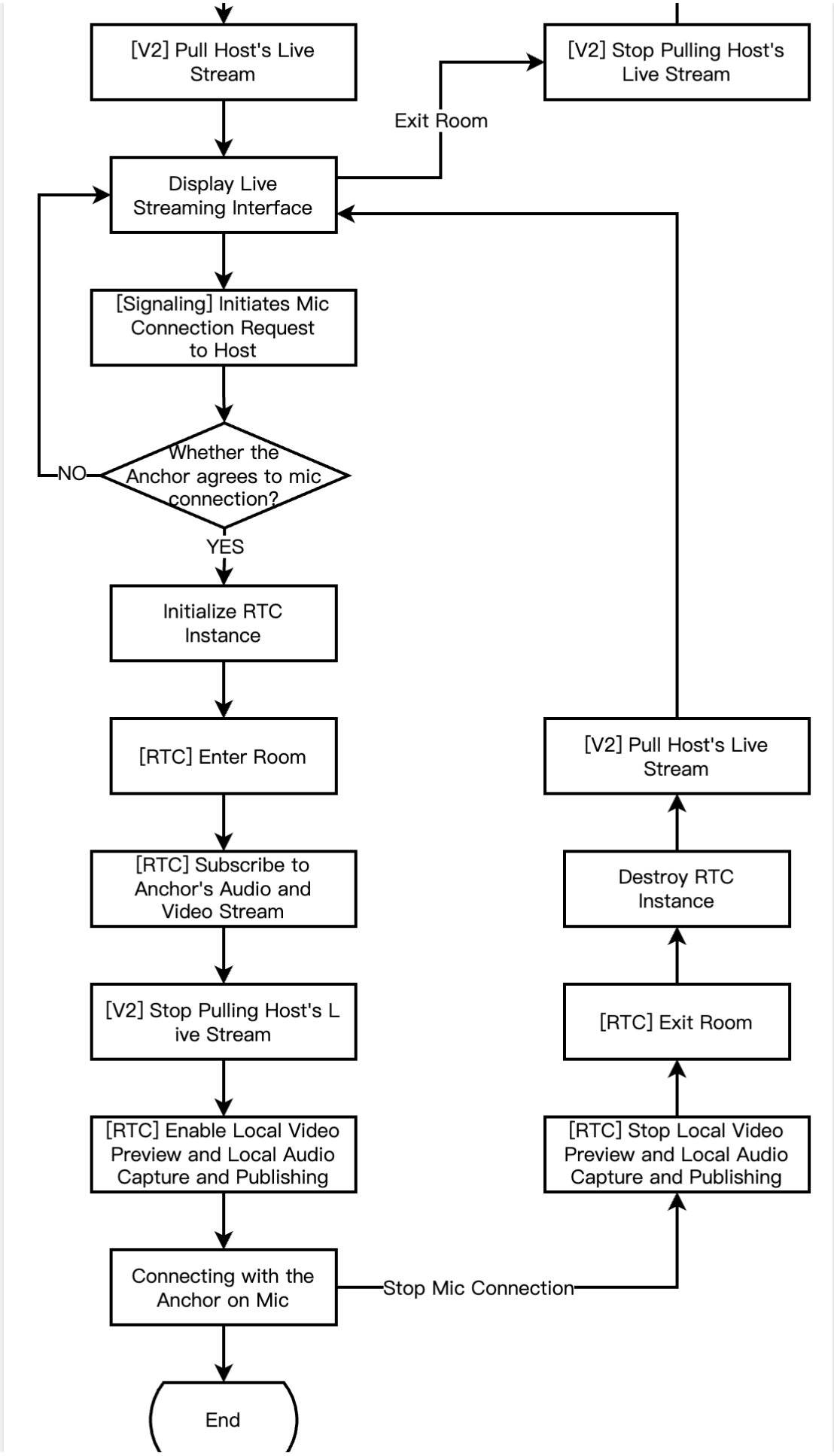
©2013-2022 Tencent Cloud. All rights reserved.



The following diagram shows the process for RTC CDN live streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

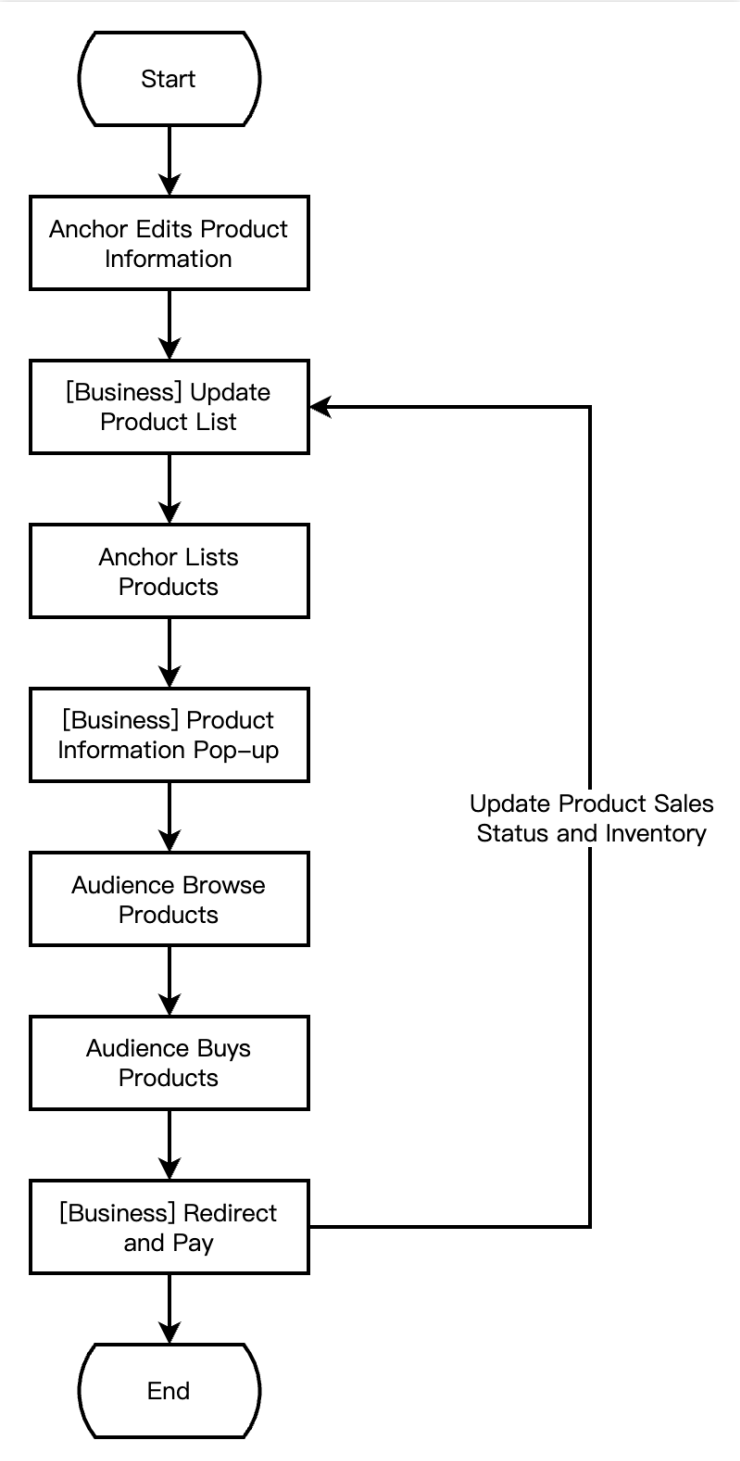


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

The diagram below shows the process in live streaming merchandising scenarios, where the anchor edits and lists products, while audience browses and purchases products.



Integration Preparations

©2013-2022 Tencent Cloud. All rights reserved.

Step 1: activate the service

E-commerce live streaming scenarios usually rely on paid PaaS services such as Real-Time Communication (TRTC), Beauty Special Effect, Player SDK. Among them, TRTC provides real-time audio and video interactive capabilities, Special Effect provides beauty special effects, and the player is responsible for live and on-demand playback. You can freely choose to activate the above services according to your actual business needs.

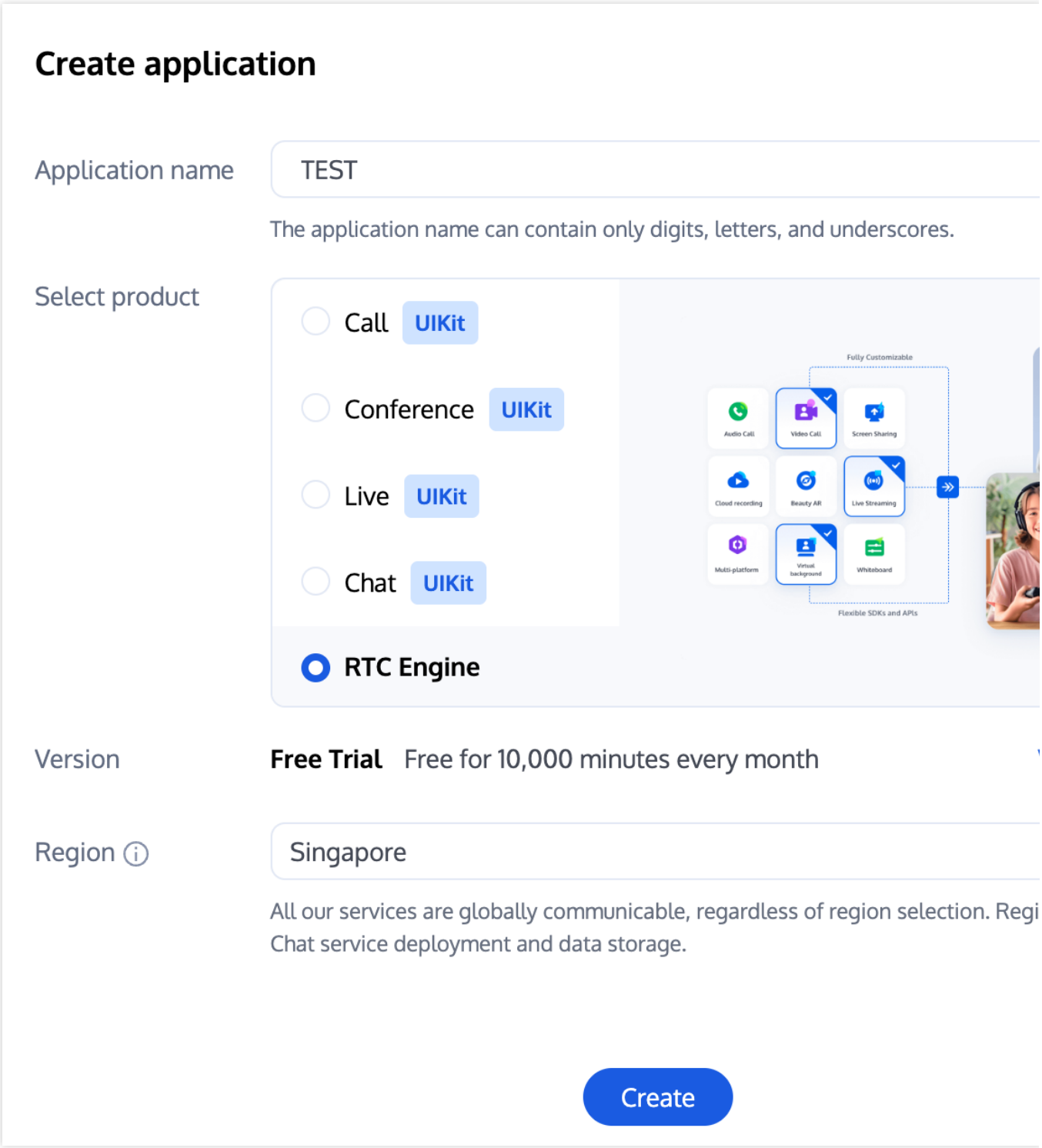
Activate TRTC Service

Activate Special Effect Service

Activate Player Service

1. First, you need to log in to the TRTC Console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



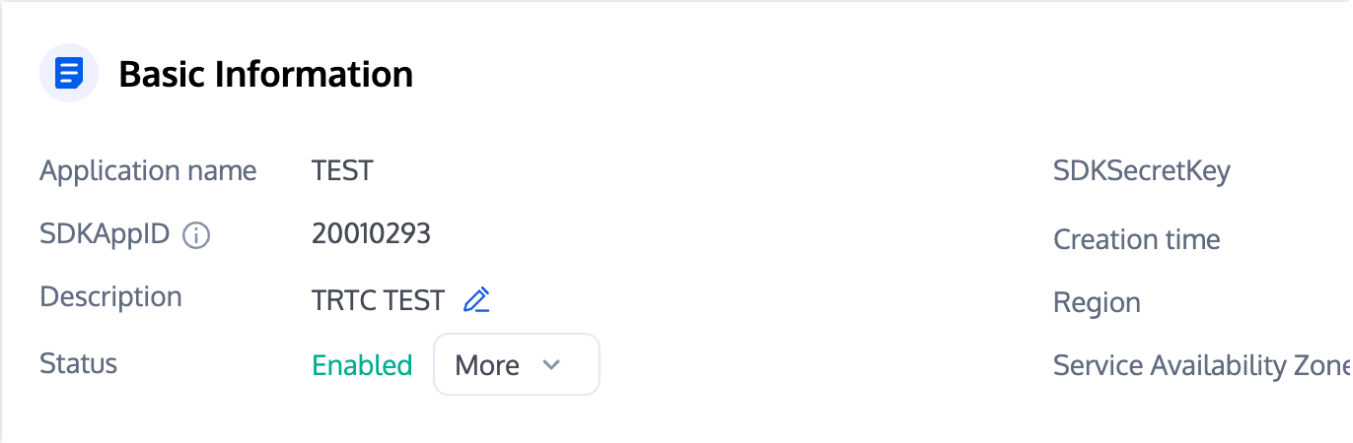
Note:

It is recommended to create two separate applications for testing and production environments. Each account (UIN) is provided with 10,000 minutes of free usage per month within one year.

The TRTC monthly package is divided into Trial Version (by default), Basic Version, and Professional Version, which can unlock different value-added features and services. For details, see Version Features and Monthly Package Description.

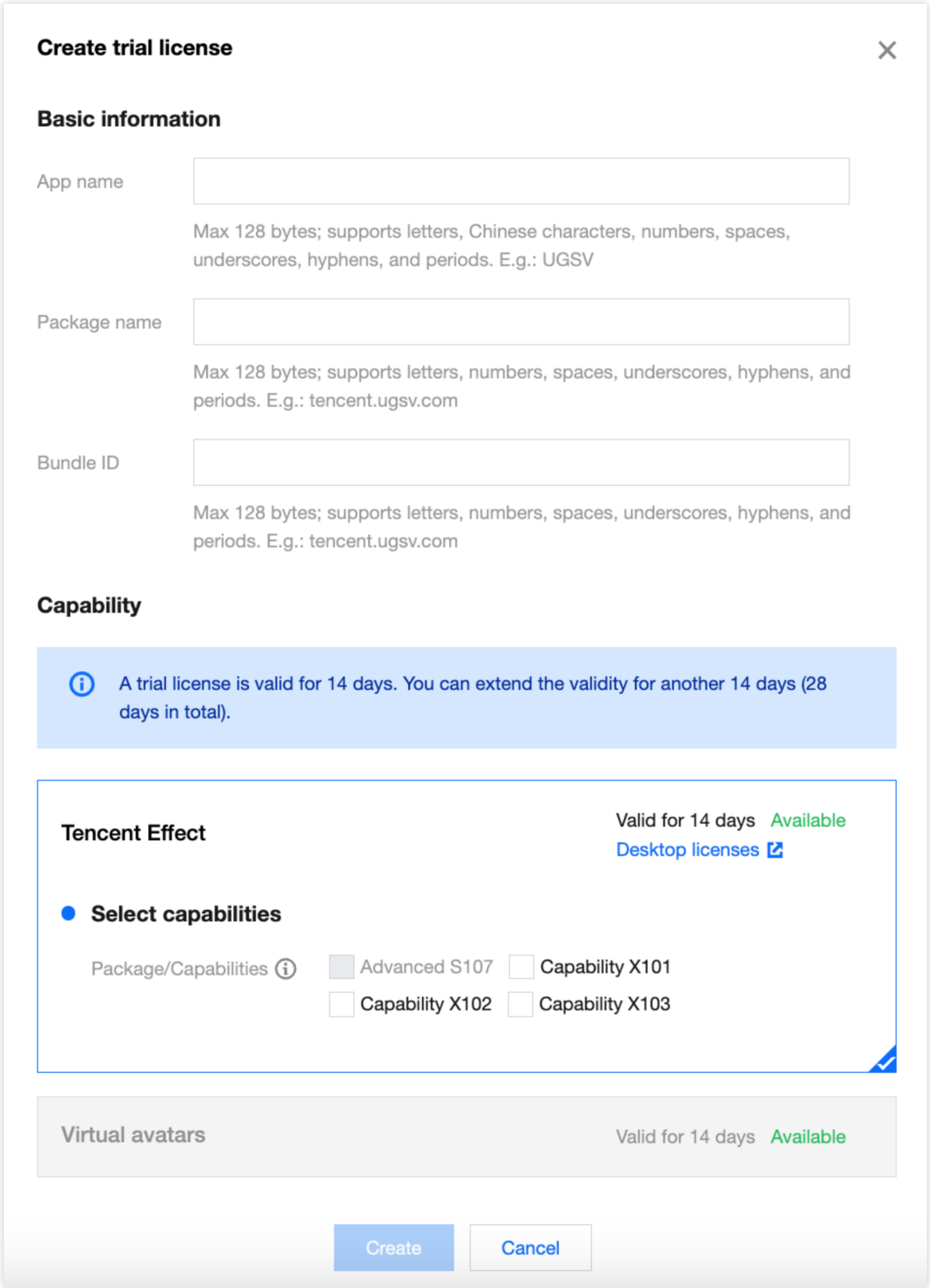
©2013-2022 Tencent Cloud. All rights reserved.

2. Once the application is created, you can find basic information about it under the Application Management - Application Overview section. It is important to store the **SDKAppID** and **SDKSecretKey** for later use and to avoid key leakage to prevent unauthorized traffic usage.



1. Log in to Cloud Special Effect Console > Mobile License. Click **Create Trial License** (the free trial validity period for Trial Version License is 14 days. It is extendable once for a total of 28 days). Fill in the actual requirements for `App Name` , `Package Name` and `Bundle ID` . Select **Special Effect**, and choose the capabilities to be tested: Advanced Package S1-07, Atomic Capability X1-01, Atomic Capability X1-02, and Atomic Capability X1-03. **After you check it, accurately fill in the company name, and industry type. Upload** Company Service License, click **OK** to submit the review application, and wait for the manual review process.

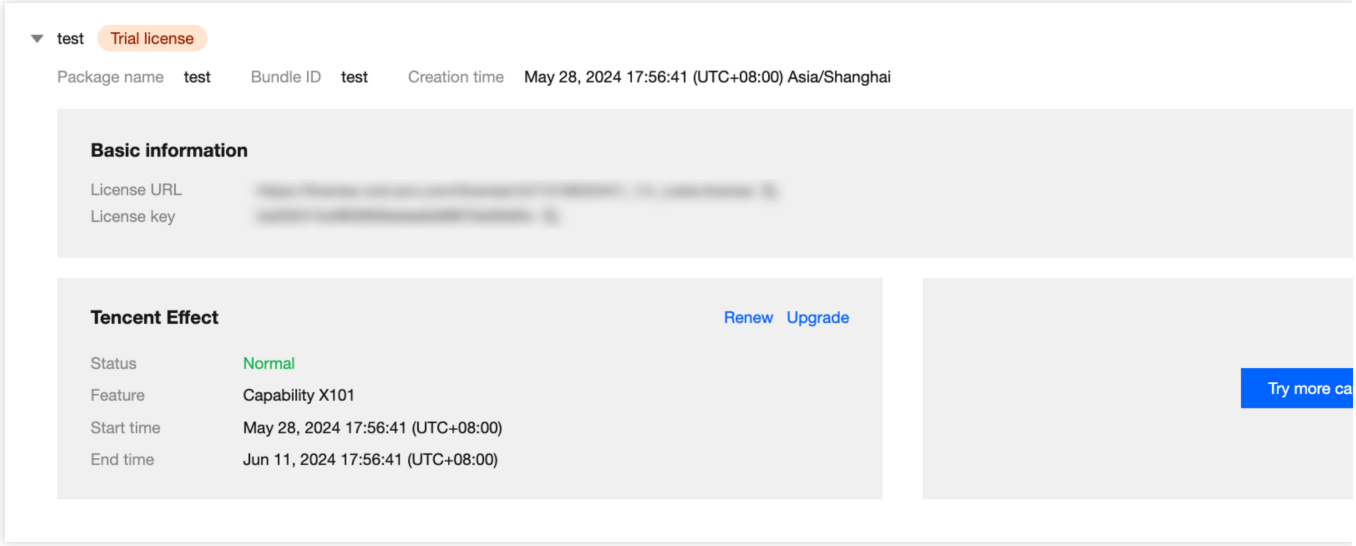
©2013-2022 Tencent Cloud. All rights reserved.



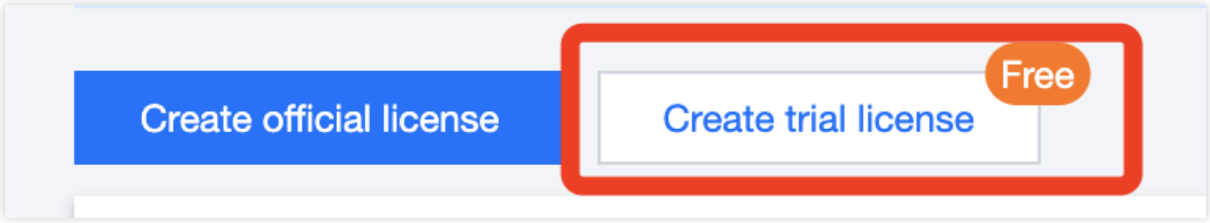
2. After the Trial License is successfully created, the page will display the generated License information. At this time, the License URL and License Key parameters are not yet effective and will only become active after the submission is

©2013-2022 Tencent Cloud. All rights reserved.

approved. **When configuring SDK initialization, you need to input both the License URL and License Key parameters. Keep the following information secure.**

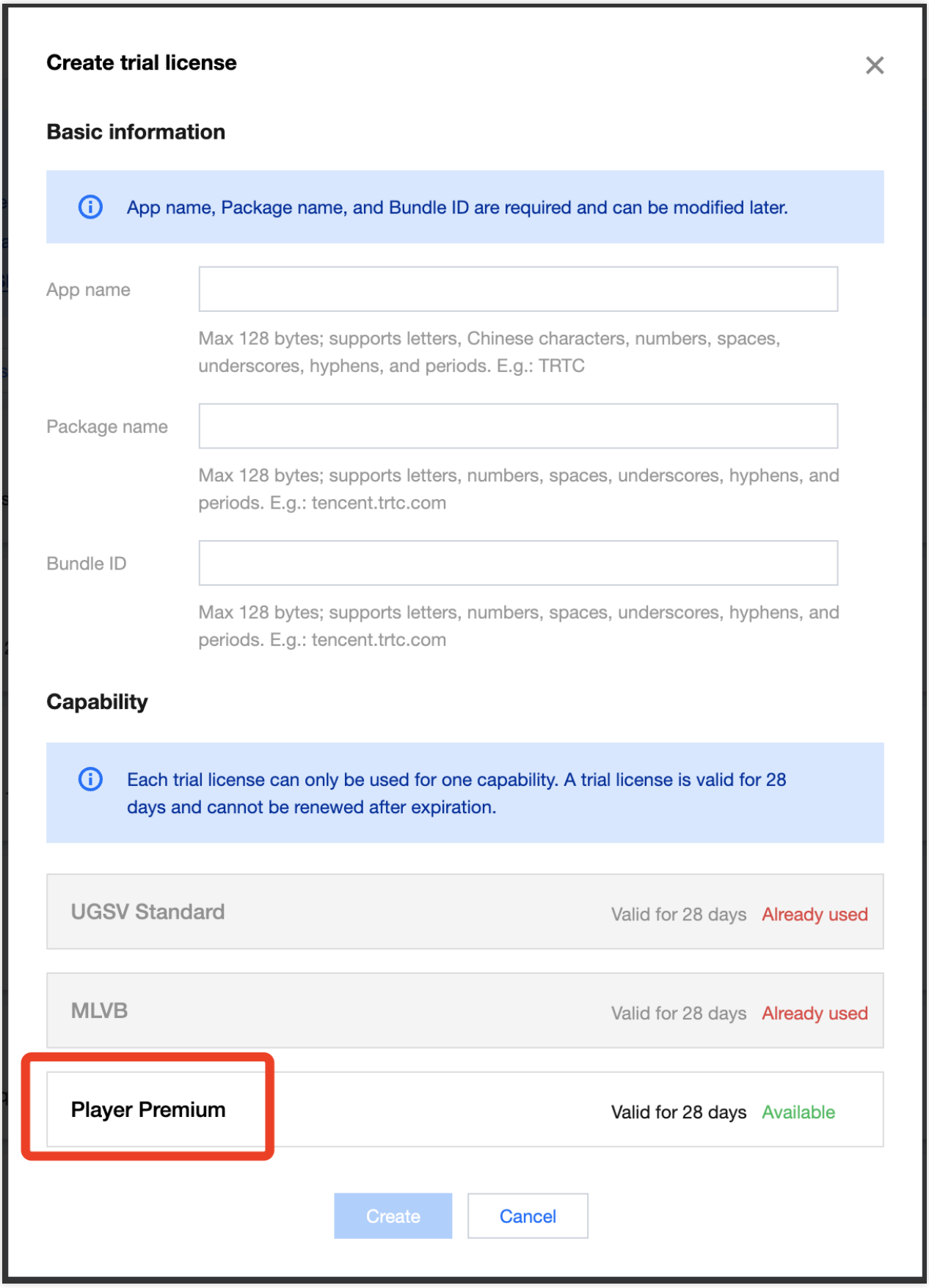


1. Log in to VOD Console or CSS Console > **License Management** > **Mobile License**, and click **Create Trial License**.



2. Enter the `App Name` , `Package Name` , and `Bundle ID` according to your actual needs, select **Player Premium**, and click **OK**.

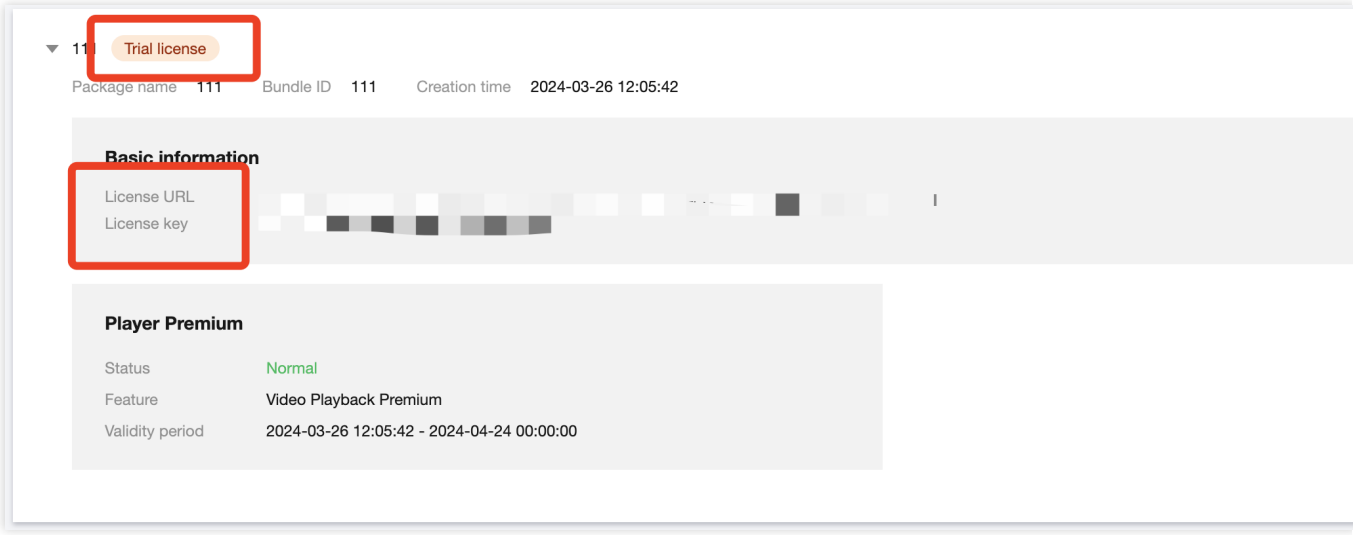
©2013-2022 Tencent Cloud. All rights reserved.



3. After the Trial License is successfully created, the page will display the generated License information. **When initializing the SDK configuration, you need to enter two parameters: License Key and License URL, so**

©2013-2022 Tencent Cloud. All rights reserved.

carefully save the following information.



Note:

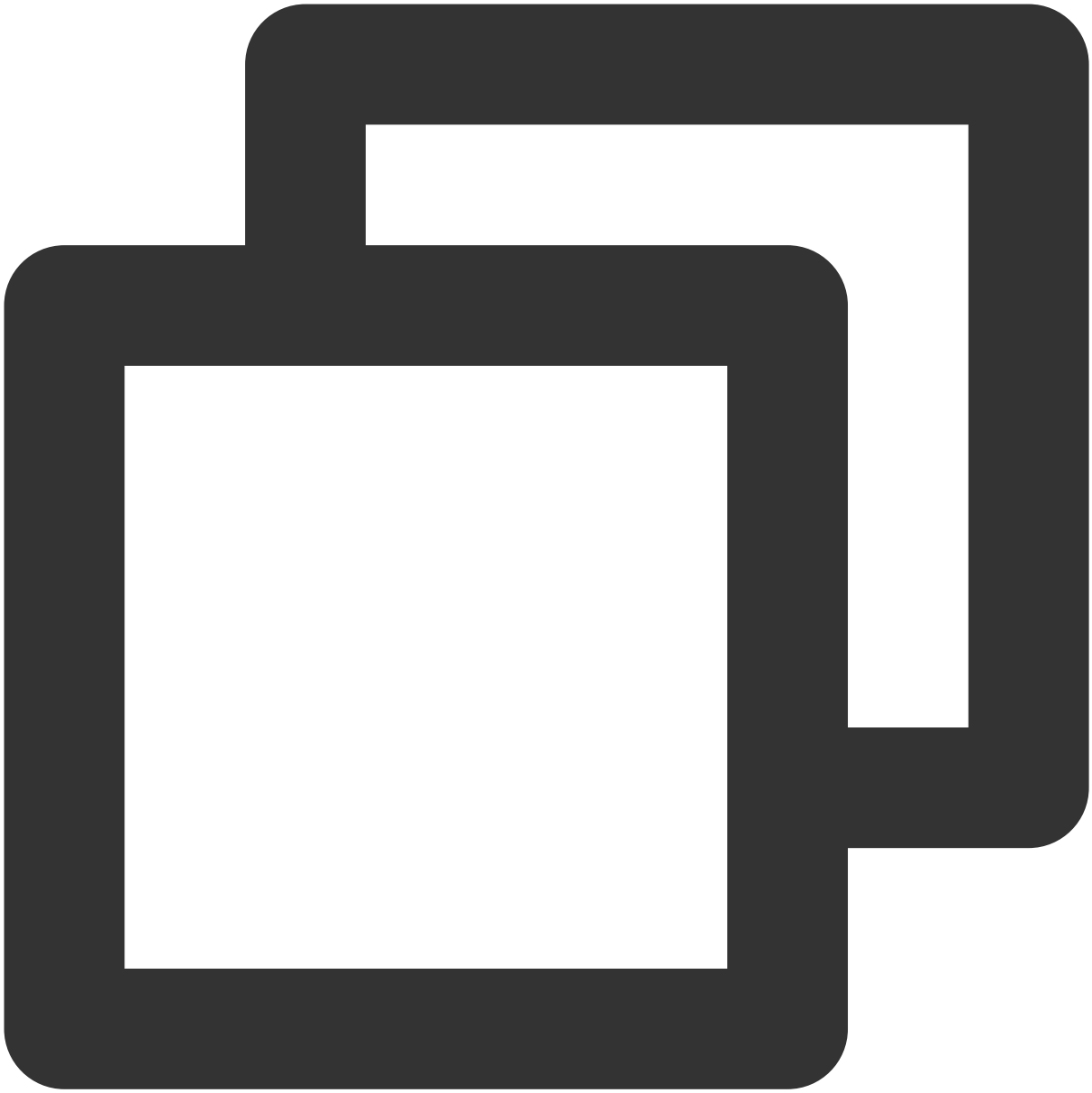
The License URL and Key for the same application are unique; after the Trial License is upgraded to the official version, the License URL and Key remain unchanged.

Step 2: import SDK

TRTC SDK, Special Effect SDK, and Player SDK have all been released on the **mavenCentral** repository. You can configure gradle to download and update automatically.

1. Add the dependency for the appropriate version of the SDK in dependencies.

©2013-2022 Tencent Cloud. All rights reserved.

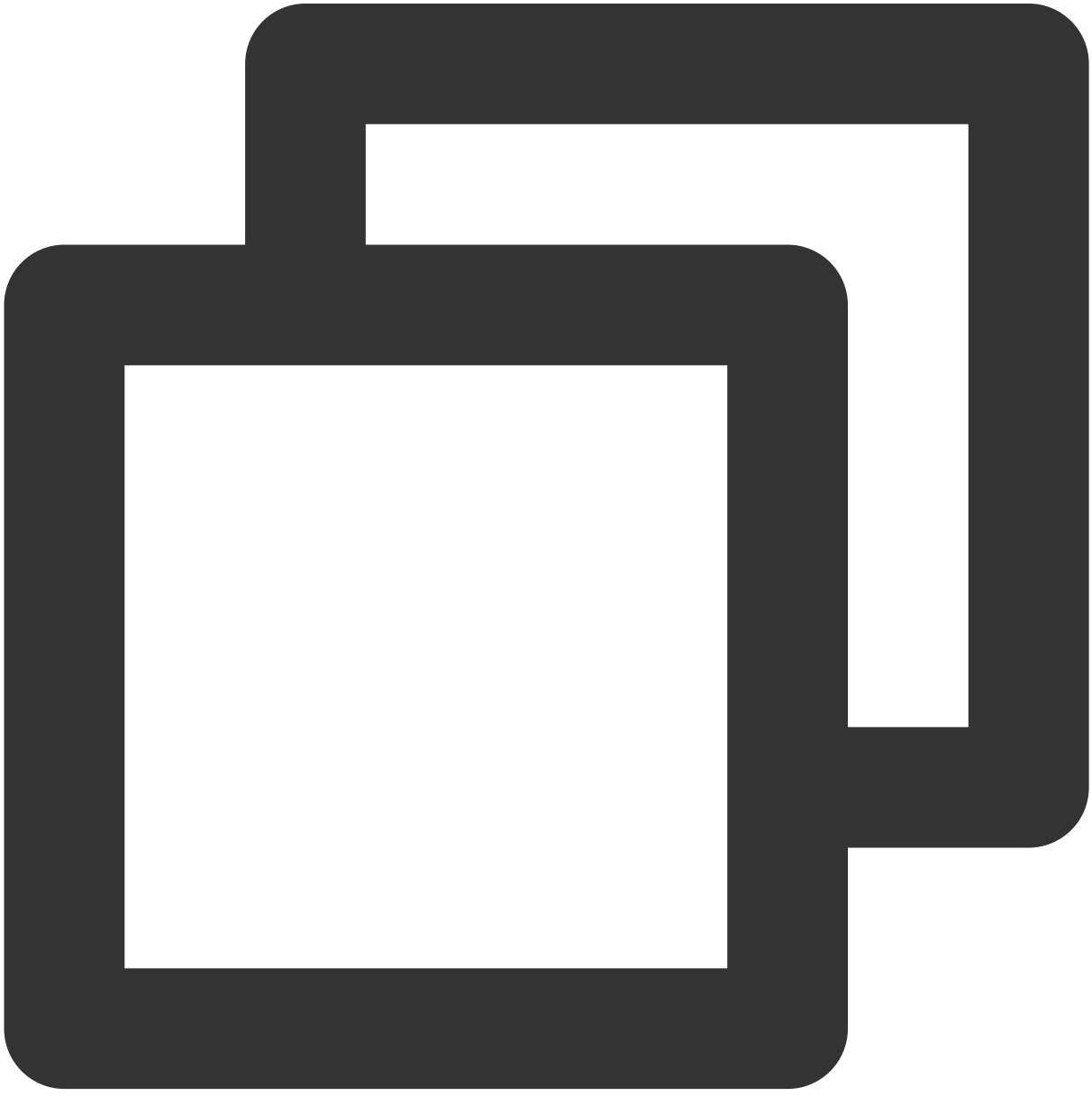


```
dependencies {
    // The full feature version of SDK, including TRTC, live streaming, short video
    implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'

    // Special Effect SDK example of S1-07 package is as follows
    implementation 'com.tencent.mediacloud:TencentEffect_S1-07:latest.release'
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrate the TRTC SDK and Manually Integrate Special Effect SDK.
The implementation of the e-commerce live streaming scenario usually relies on a combination of multiple capabilities such as TRTC and player. **To avoid the symbol conflict issues that arise from single integrations, it is recommended to integrate the full feature version of the SDK**.
2. Specify the CPU architecture used by the app in defaultConfig.



```
defaultConfig {
     ndk {
             abiFilters "armeabi-v7a", "arm64-v8a"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    }
```

Note:

The full feature version of LiteAVSDK supports armeabi/armeabi-v7a/arm64-v8a/x86/x86_64 architectures, while Special Effect SDK only supports armeabi-v7a/arm64-v8a architectures.

3. Click **Sync Now** to automatically download the SDK and integrate it into your project. If your special effect package includes dynamic effect and filter features, then you need to download the corresponding package from the SDK Download Page, unzip the free filter materials (./assets/lut) and animated stickers (./MotionRes) from the package and place them in the following directories in your project:

Dynamic Effect: `../assets/MotionRes`

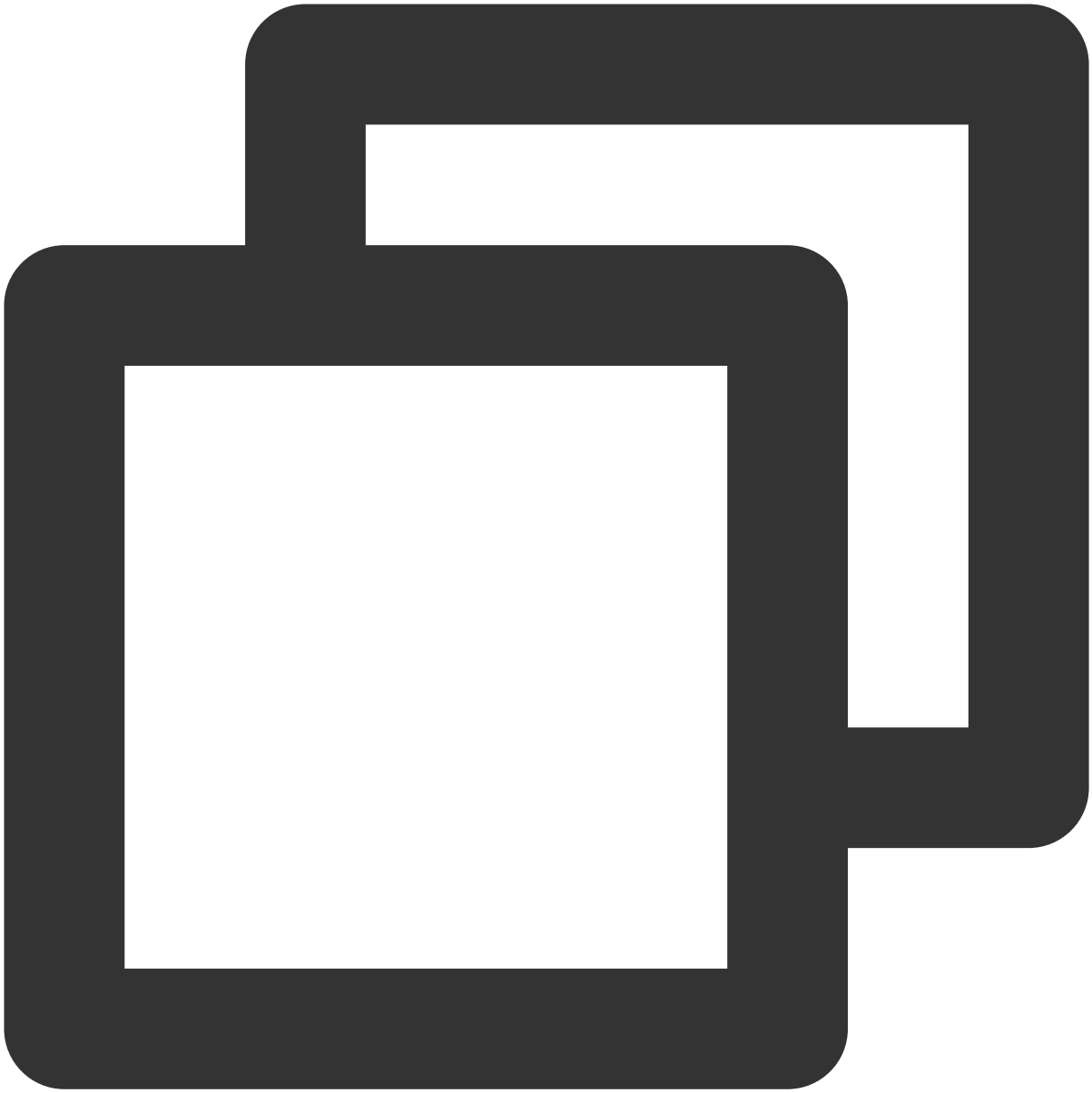
Filter: `../assets/lut`

Step 3: project configuration

1. Configure permissions

To configure App permissions in AndroidManifest.xml, for an e-commerce live streaming scenario, both LiteAVSDK and Special Effect SDK require the following permissions:

©2013-2022 Tencent Cloud. All rights reserved.



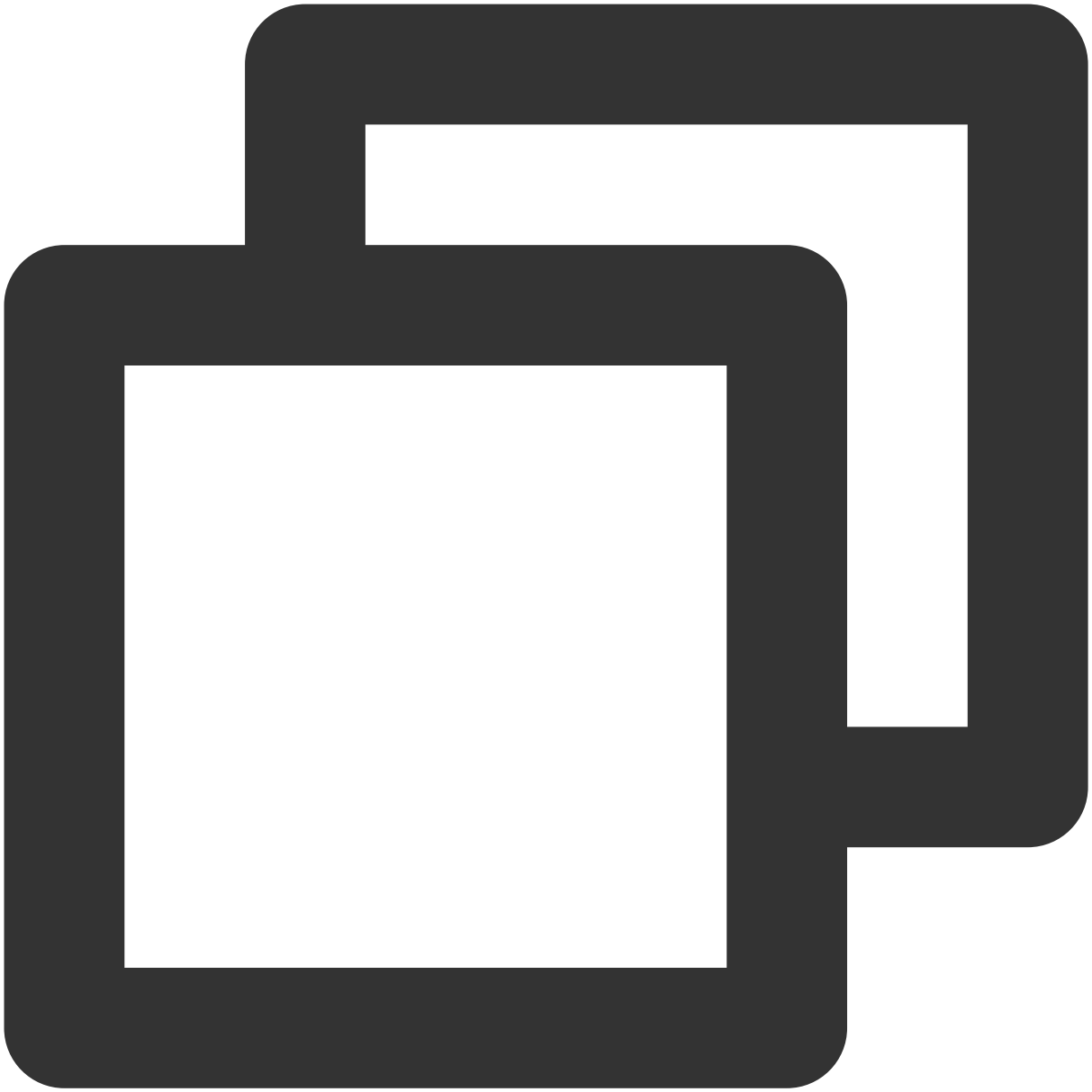
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Do not set `android:hardwareAccelerated="false"` . Disabling hardware acceleration will result in failure to render the other party's video stream.
LiteAVSDK does not have built-in permission request logic, so you need to declare the corresponding permissions yourself. Some permissions (such as storage, recording and camera) also require runtime dynamic requests.
If the Android project's `targetSdkVersion` is 31 or higher, or if the target device runs Android 12 or a newer version, the official requirement is to dynamically request `android.permission.BLUETOOTH_CONNECT` permission in the code to use the Bluetooth feature properly. For more information, see Bluetooth Permissions.
2. Obfuscation configuration
Since we use Java's reflection features inside the SDK, you need to add relevant SDK classes to the non-obfuscation list in the proguard-rules.pro file:

©2013-2022 Tencent Cloud. All rights reserved.



```
-keep class com.tencent.** { *; }
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class androidx.exifinterface.** { *;}
```

Step 4: authentication and authorization

TRTC Authentication Credential

©2013-2022 Tencent Cloud. All rights reserved.

Special Effect Authentication License

Player Authentication License

UserSig is a security protection signature designed by the cloud platform to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when entering a room.

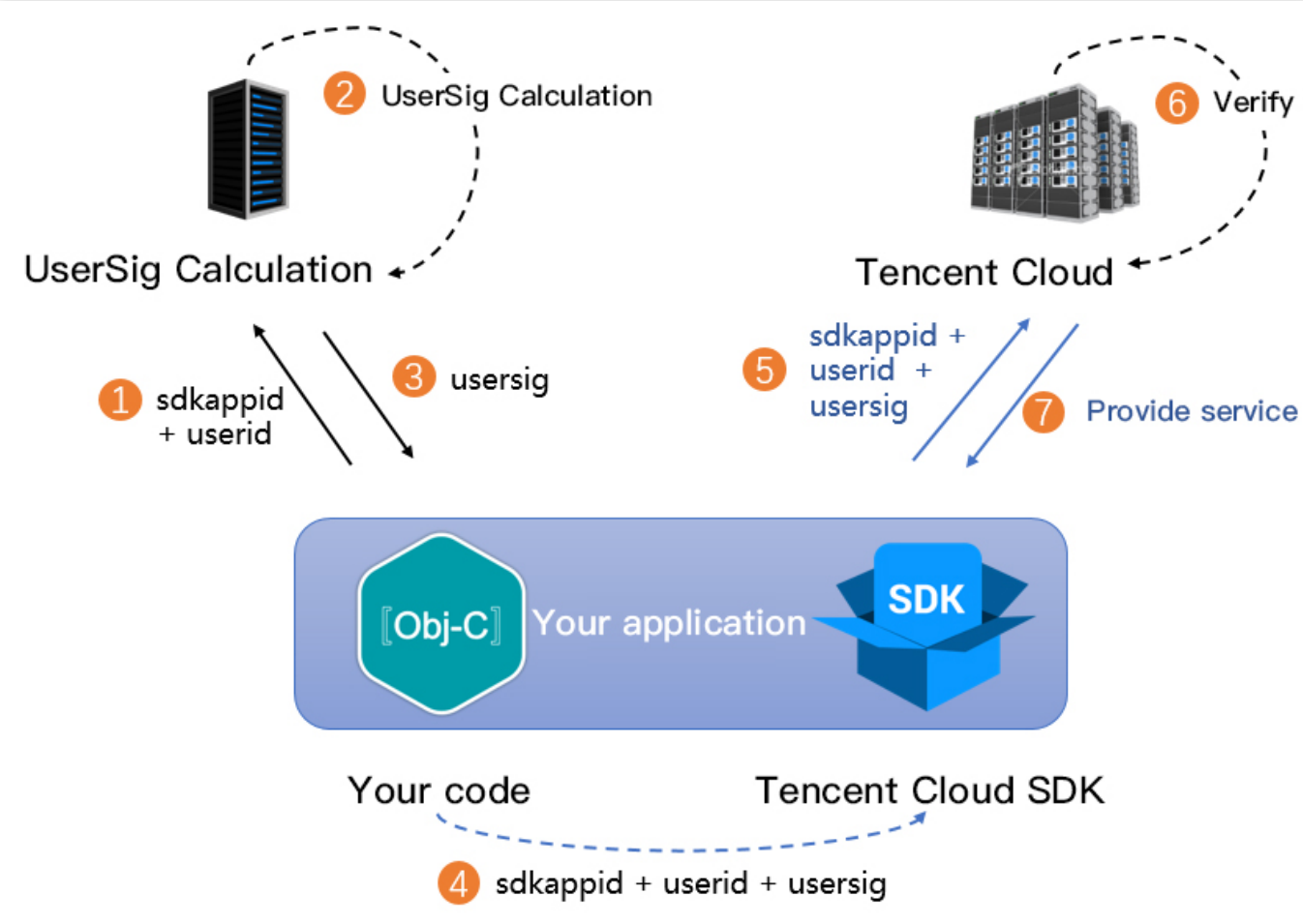
Debugging and testing stage: UserSig can be generated through Client Sample Code and Console Access, which are only used for debugging and testing.

Production stage: It is recommended to use the server computing UserSig solution, which has a higher security level and helps prevent the client from being decompiled and reversed, to avoid the risk of key leakage.

The specific implementation process is as follows:

1. Before calling the initialization API of the SDK, your app must first request UserSig from your server.
2. Your server generates the UserSig based on the SDKAppID and UserID.
3. The server returns the generated UserSig to your app.
4. Your app sends the obtained UserSig to the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to the cloud server for verification.
6. The cloud platform verifies the validity of the UserSig.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.

©2013-2022 Tencent Cloud. All rights reserved.



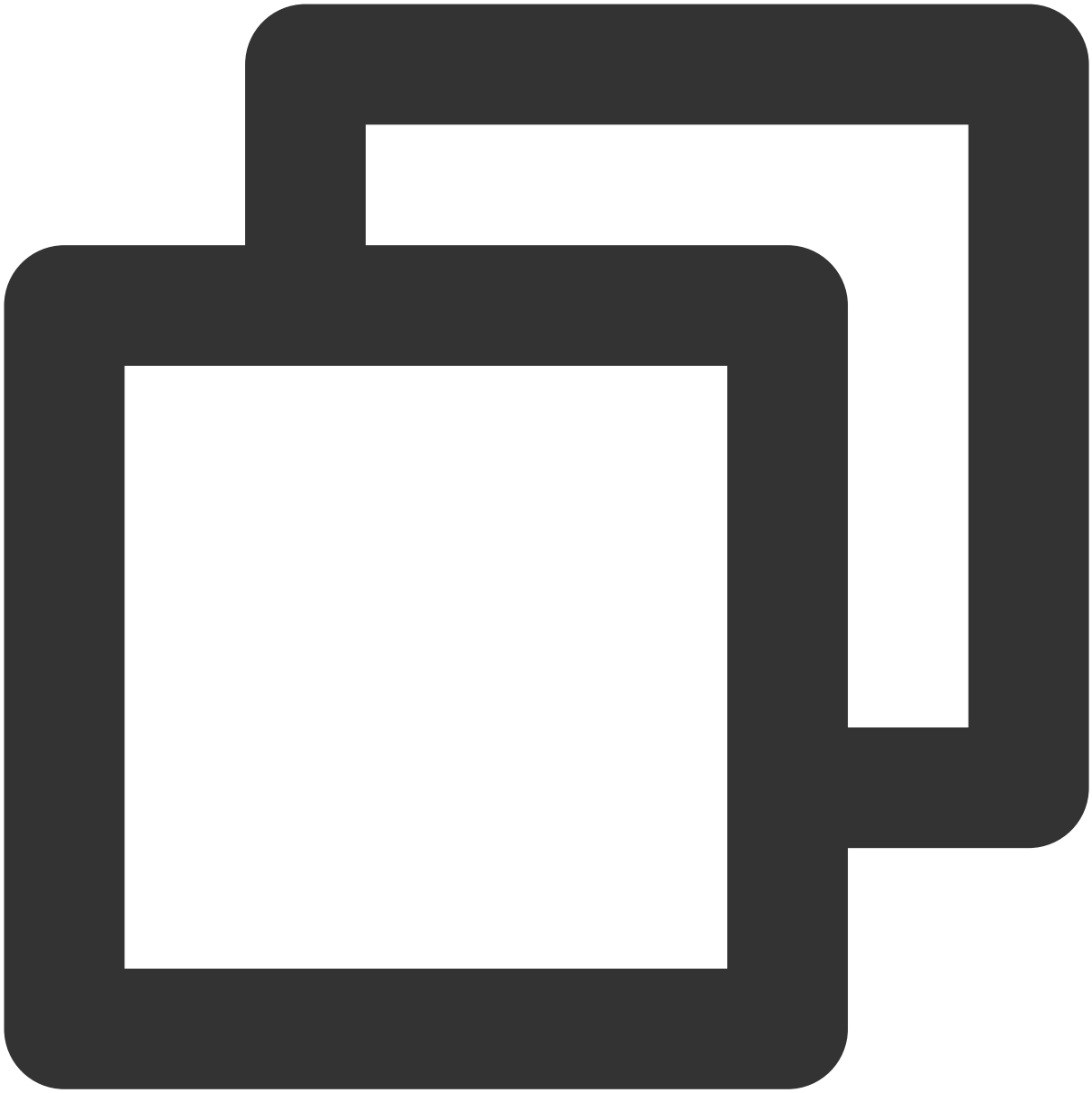
Note:

The method of generating UserSig locally during the debugging and testing stage is not recommended for the online environment because it may be easily decompiled and reversed, causing key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Before using Beauty Special Effect, you need to verify the license credential with the cloud platform. Configuring the License requires License Key and License Url. Sample code is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



```
import com.tencent.xmagic.telicense.TELicenseCheck;

// If the purpose is just to trigger the download or update of the License, and not
TELicenseCheck.getInstance().setTELicense(context, URL, KEY, new TELicenseCheck.TEL
    @Override
    public void onLicenseCheckFinish(int errorCode, String msg) {
        // Note: This callback does not necessarily be called on the calling thread
        if (errorCode == TELicenseCheck.ERROR_OK) {
            // Authentication successful
        } else {
            // Authentication failed
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            }


        }
    });
```

Note:

It is recommended to trigger the authentication permission in the initialization code of related business modules, to avoid having to download the License temporarily before use. Additionally, during authentication, network permissions must be ensured.

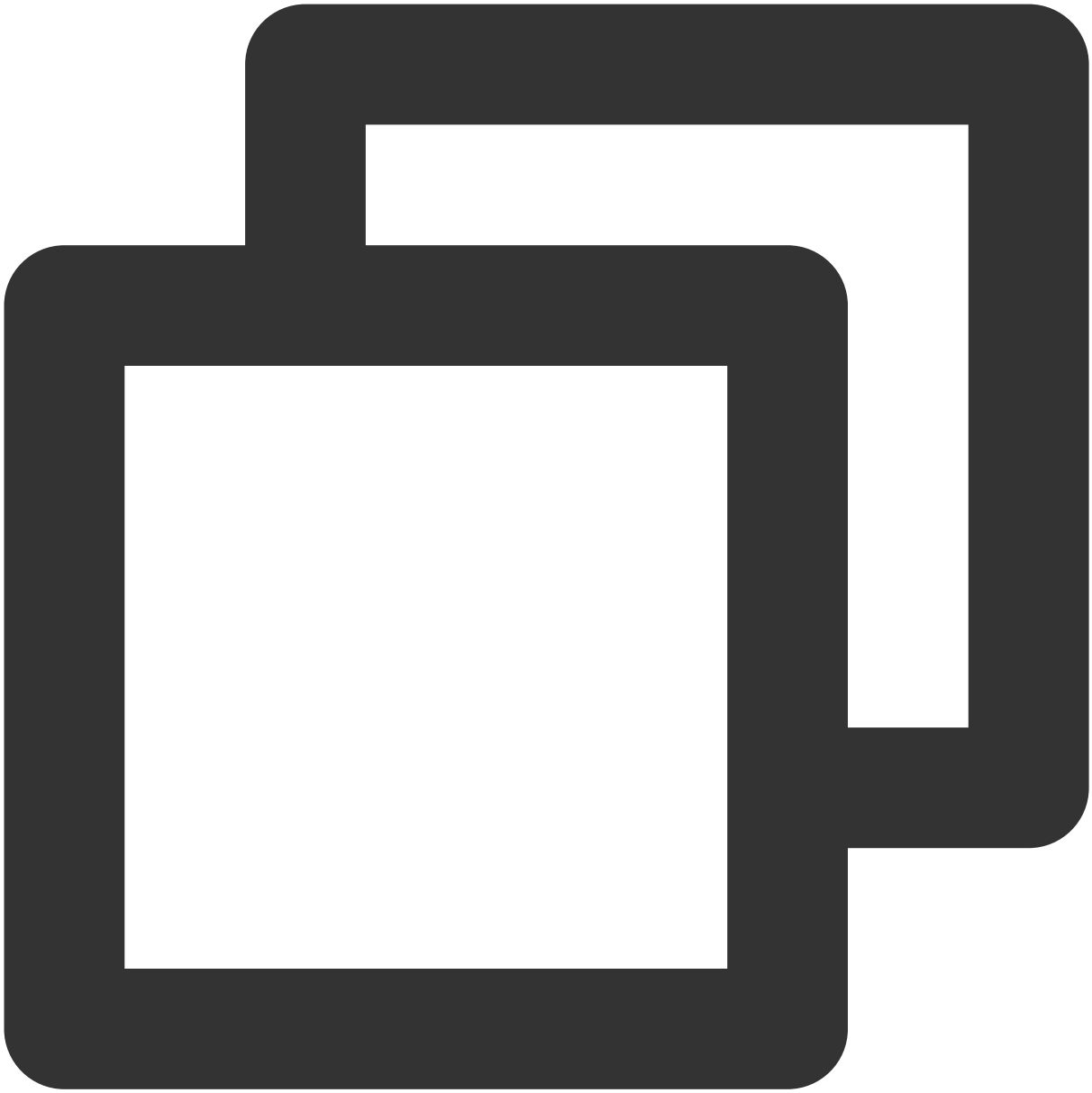
The actual application's Package Name must exactly match the Package Name associated with the License creation. Otherwise, it will lead to License verification failure. For details, see Authentication Error Codes.

The live streaming and on-demand playback features require setting the License before success in playback. Otherwise, playback will fail (black screen). It needs to be set globally only once. If you have not obtained the License, you can freely apply for a Trial Version License for normal playback. The Official Version License requires purchase.

After successfully applying for License, you will receive two strings: **License URL** and **License Key**.

Before your App calls the SDK-related features, you need to configure as follows (recommended to configure in the Application class):

©2013-2022 Tencent Cloud. All rights reserved.

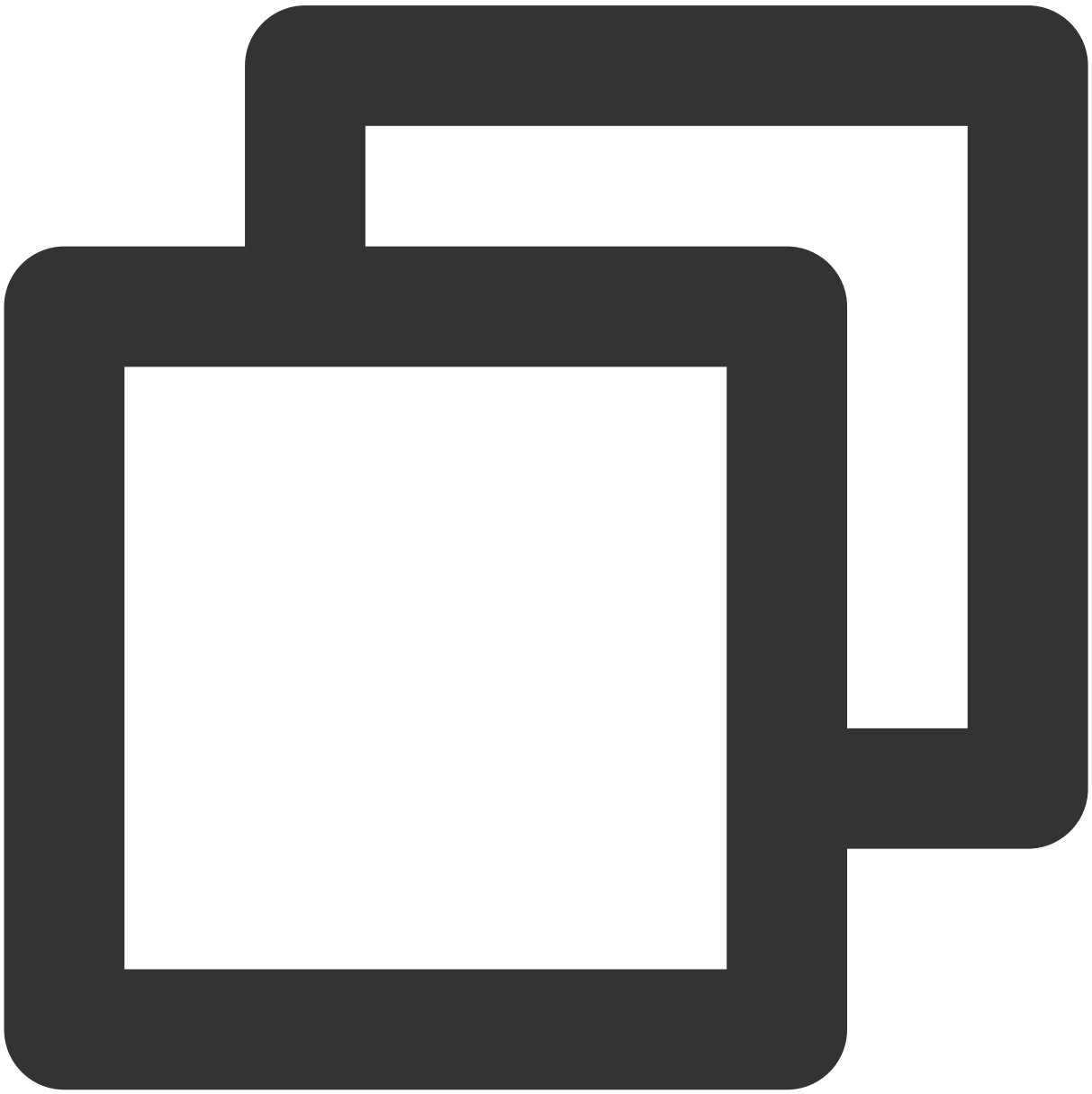


```
public class MApplication extends Application {
    public void onCreate() {
        super.onCreate();
        String licenceURL = ""; // The obtained licence URL
        String licenceKey = ""; // The obtained licence key
        TXLiveBase.getInstance().setLicence(appContext, licenceURL, licenceKey);
        TXLiveBase.setListener(new TXLiveBaseListener() {
            @Override
            public void onLicenceLoaded(int result, String reason) {
                Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reas
                if (result != 0) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                    // If the result is not 0, it means the setting has failed, and
                    TXLiveBase.getInstance().setLicence(appContext, licenceURL, lic
                }
            }
        });
    }
}
```

After the License is successfully set (you need to wait for a while, the specific time depends on the network conditions), you can use the following method to view the License information:



©2013-2022 Tencent Cloud. All rights reserved.

```
TXLiveBase.getInstance().getLicenceInfo();
```

Note:

The actual application's Package Name must exactly match the Package Name associated with the License creation. Otherwise, it will lead to License verification failure.

The License is a strong online verification logic. When the TXLiveBase#setLicence is called after the application is started for the first time, the network must be available. At the first launch of the App, if the network permission is not yet authorized, you need to wait until the permission is granted before calling TXLiveBase#setLicence again.

Listen to the loading result of TXLiveBase#setLicence: For onLicenceLoaded API, if it fails, you should retry and guide according to the actual situation. If it fails multiple times, you can limit the frequency and supplement with product pop-ups and other guides to allow users to check the network conditions.

TXLiveBase#setLicence can be called multiple times. It is recommended to call TXLiveBase#setLicence when entering the main interface of the App to ensure successful loading.

For multi-process Apps, ensure that every process using the player calls TXLiveBase#setLicence when it starts. For example, for Apps on the Android side that use a separate process for video playback, when the process is killed and restarted by the system during background playback, TXLiveBase#setLicence should also be called.

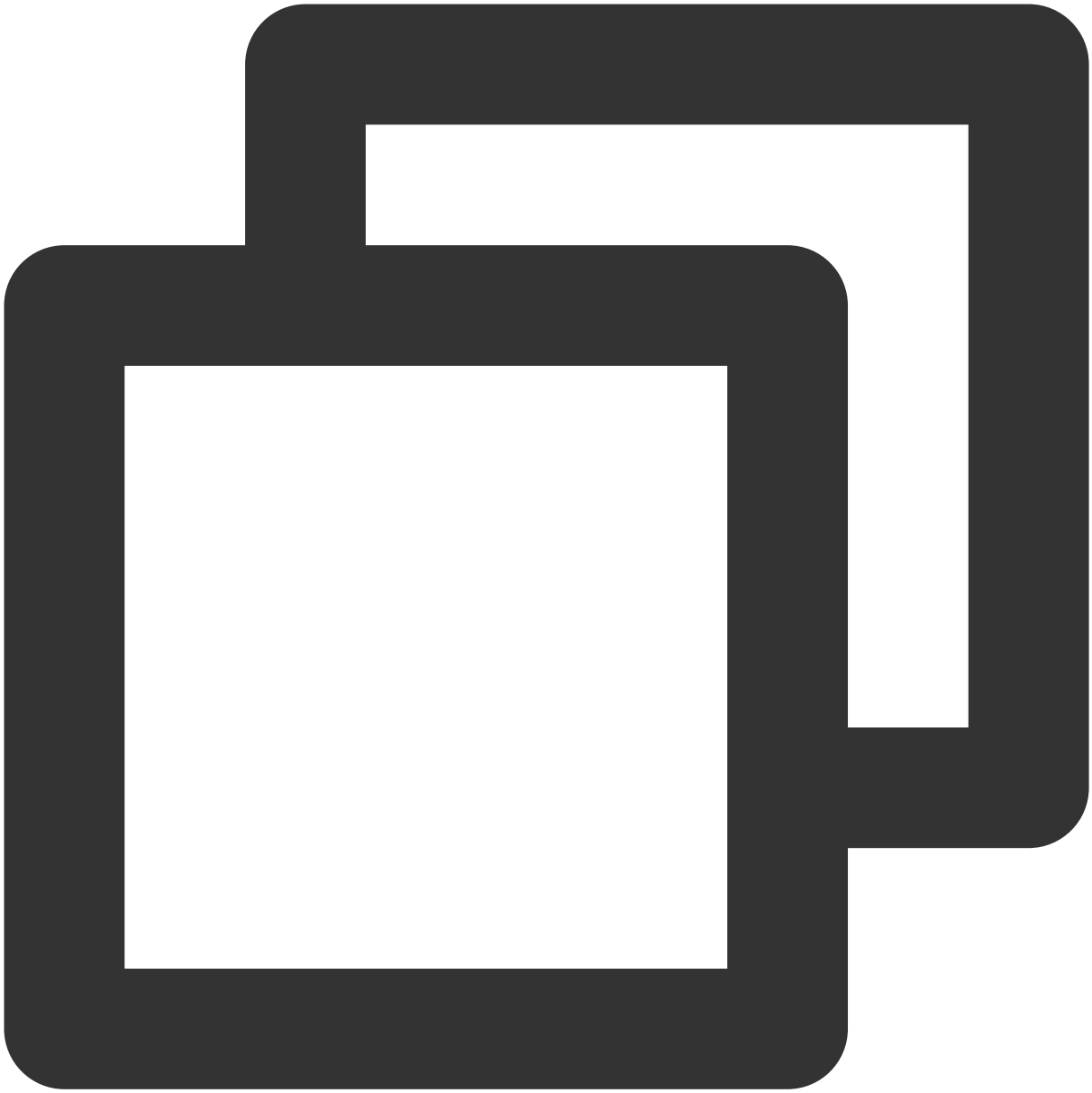
Step 5: initialize the SDK

Initialize the TRTC SDK

Initialize the Special Effect SDK

Initialize Player SDK

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (single instance pattern)
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Set event listeners
mTRTCCloud.addListener(trtcSdkListener);

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
private TRTCCloudListener trtcSdkListener = new TRTCCloudListener() {
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        Log.d(TAG, errCode + errMsg);
    }
```

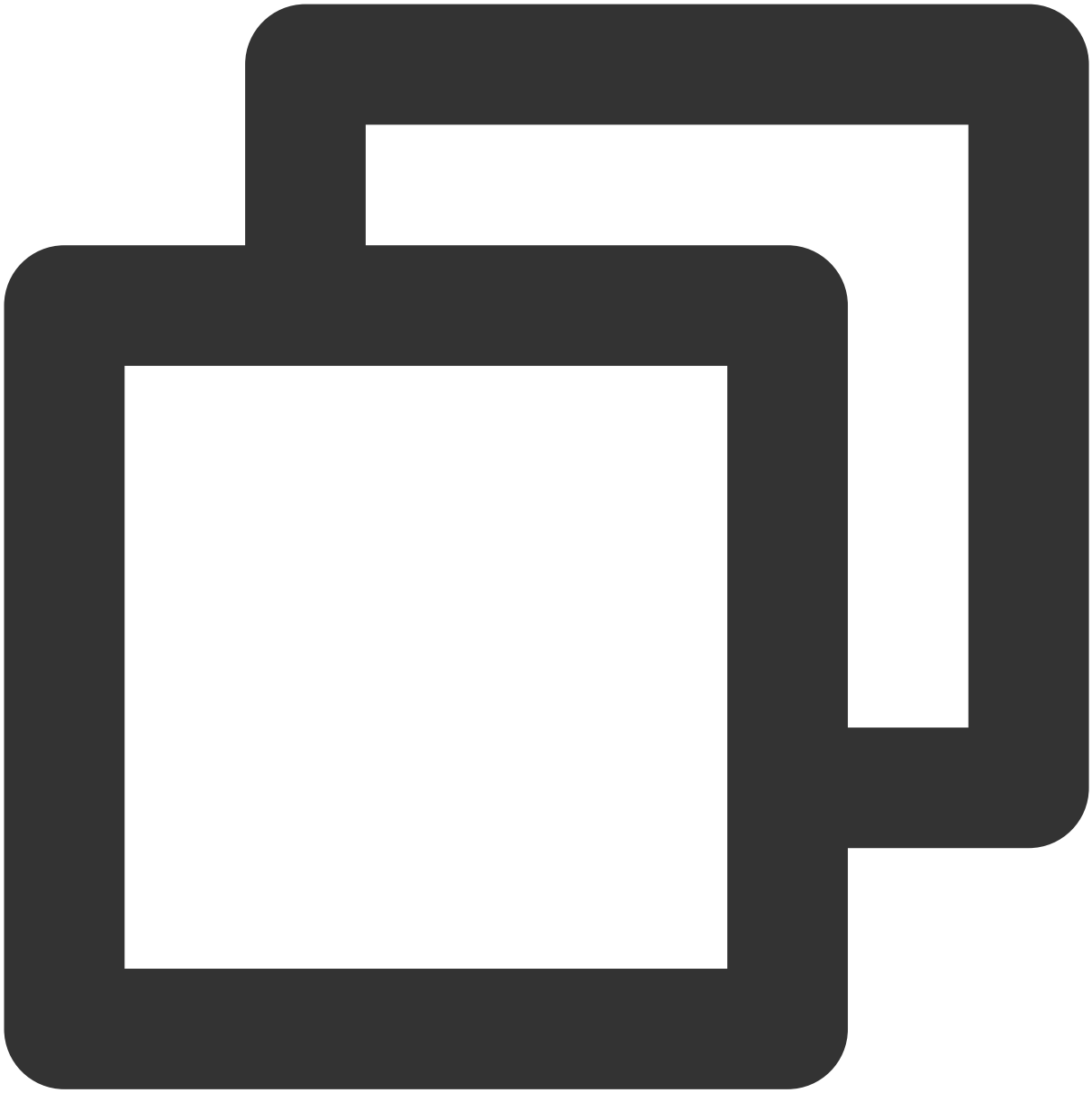
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        Log.d(TAG, warningCode + warningMsg);
    }
};

// Remove event listener
mTRTCCloud.removeListener(trtcSdkListener);
// Destroy TRTC SDK instance (single instance pattern)
TRTCCloud.destroySharedInstance();
```

Note:

It is recommended to listen to SDK events notification. Perform log printing and handling for some common errors. For details, see Error Code Table.

©2013-2022 Tencent Cloud. All rights reserved.



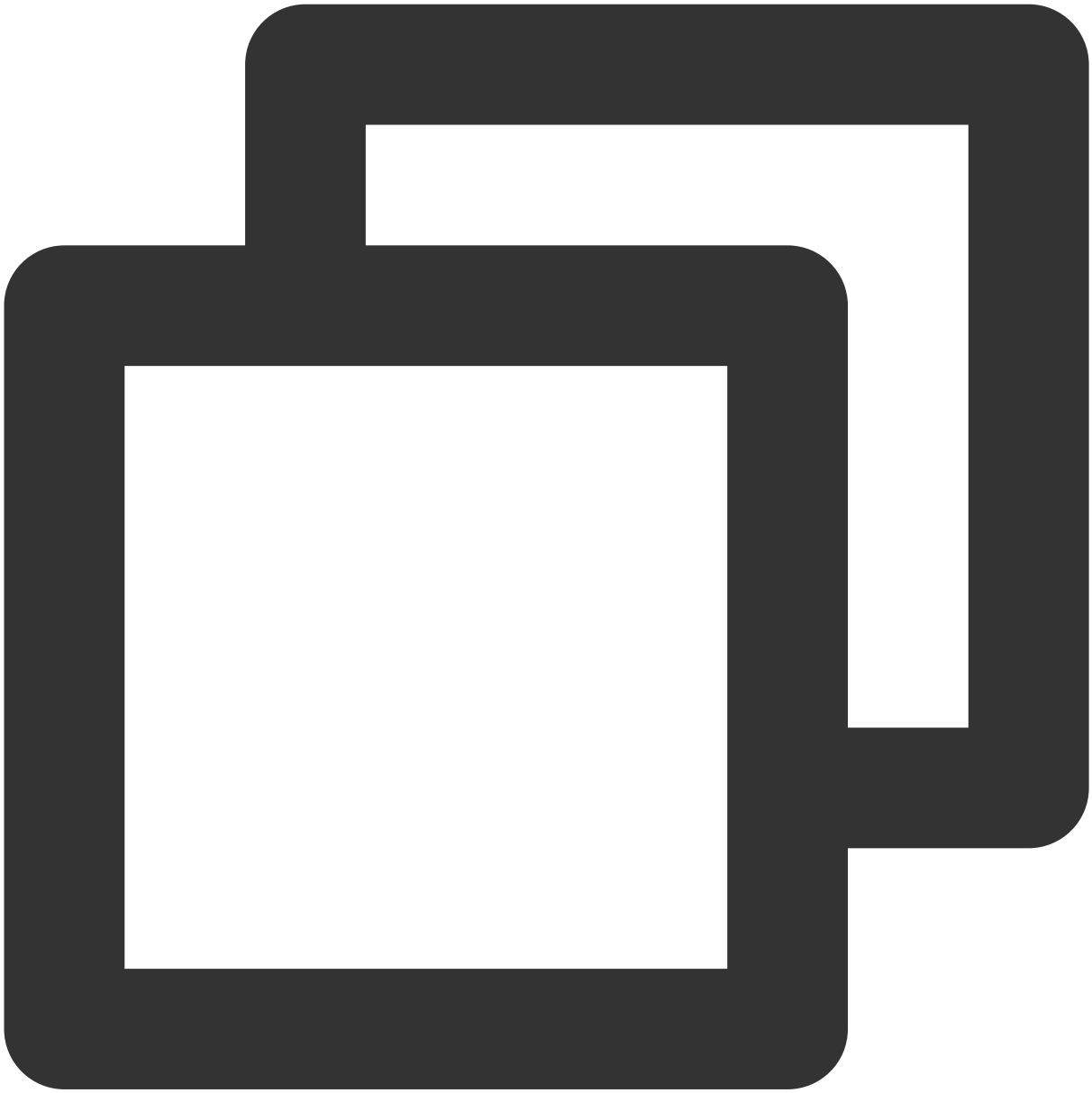
```
import com.tencent.xmagic.XmagicApi;

// Initialize the beauty SDK
XmagicApi mXmagicApi = new XmagicApi(context, XmagicResParser.getResPath(), new Xma

// During development and debugging, you can set the log level to DEBUG. For releas
mXmagicApi.setXmagicLogLevel(Log.WARN);

// Release the beauty SDK. This method needs to be called in the GL thread
mXmagicApi.onDestroy();
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

Before the Special Effect SDK is initialized, resource copying and other preparatory work are needed. For detailed steps, see Using the Special Effect SDK.

On-demand Playback Scenario SDK Initialization.



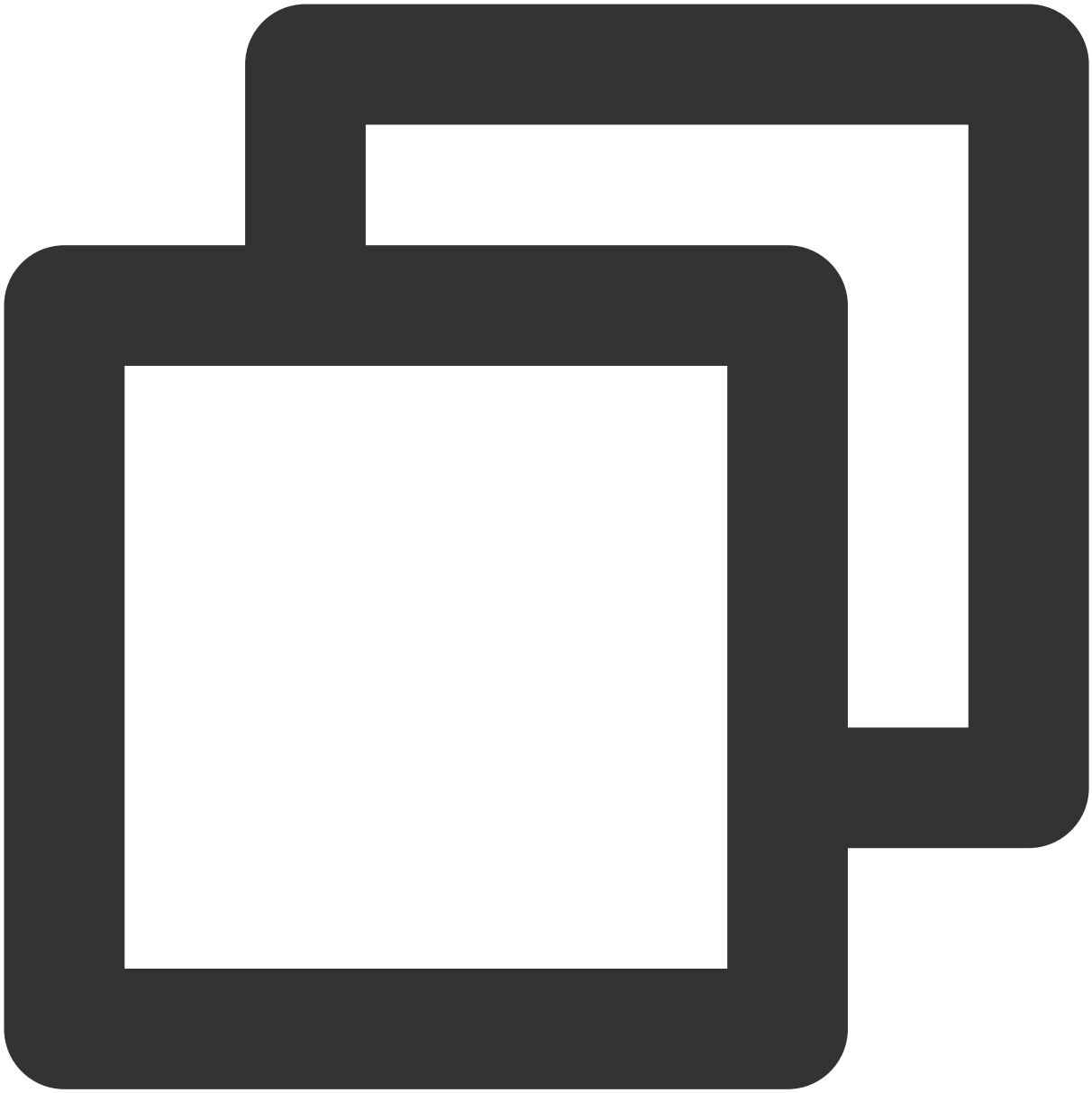
```
// Set the SDK connection environment (if you serve global users, configure the SDK
TXLiveBase.setGlobalEnv("GDPR");

// Create a Player object
TXVodPlayer mVodPlayer = new TXVodPlayer(mContext);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Add a View control for video rendering
TXCloudVideoView mPlayerView = findViewById(R.id.video_view);
// Associate the Player object with the View control
mVodPlayer.setPlayerView(mPlayerView);

// Player parameter configuration
TXVodPlayConfig config = new TXVodPlayConfig();
config.setEnableAccurateSeek(true);  // Set whether to seek accurately. The default
config.setMaxCacheItems(5);          // Set the number of cache files to 5
config.setProgressInterval(200); // Set the interval for progress callbacks, in mil
config.setMaxBufferSize(50); // The maximum pre-load size, in MB
mVodPlayer.setConfig(config);// Pass config to mVodPlayer

// Player event listener
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        // Event notification
    }

    @Override
    public void onNetStatus(TXVodPlayer player, Bundle bundle) {
        // Status feedback
    }
});
```

Live Streaming Scenarios SDK initialization.

©2013-2022 Tencent Cloud. All rights reserved.



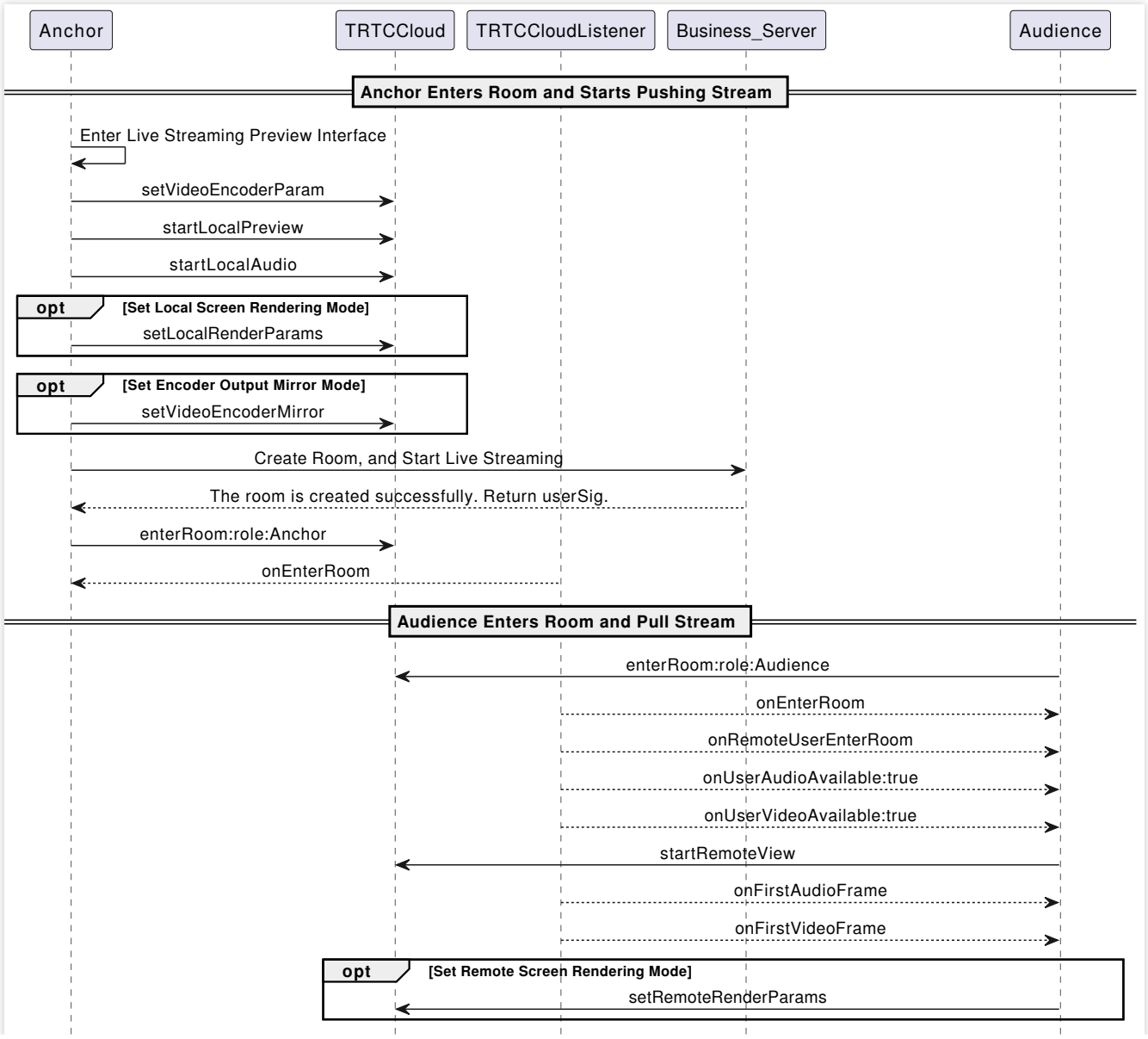
```
// The TXCloudVideoView for video rendering needs to be added in advance
TXCloudVideoView mRenderView = findViewById(R.id.video_view);
// Create a Player object
V2TXLivePlayer mLivePlayer = new V2TXLivePlayerImpl(mContext);
// Associate the Player object with the video rendering view
mLivePlayer.setRenderView(mRenderView);

// Player event listener
mLivePlayer.setObserver(new V2TXLivePlayerObserver() {
    @Override
    public void onVideoLoading(V2TXLivePlayer player, Bundle extraInfo) {
```

©2013-2022 Tencent Cloud. All rights reserved.

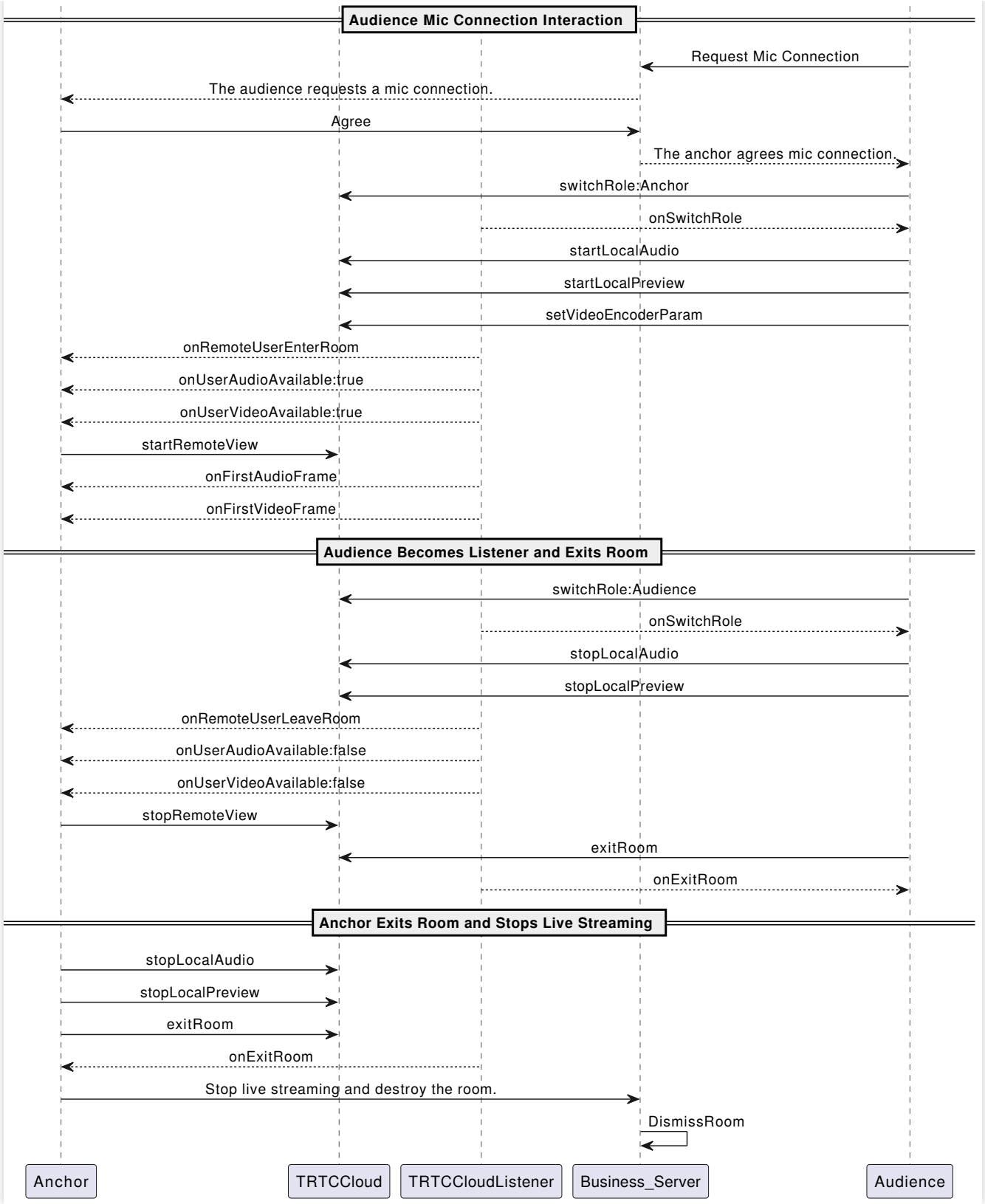
```
            // Video loading event
        }

        @Override
        public void onVideoPlaying(V2TXLivePlayer player, boolean firstPlay, Bundle ext
            // Video playback event
        }
    });
```

Integration Process

API Sequence Diagram



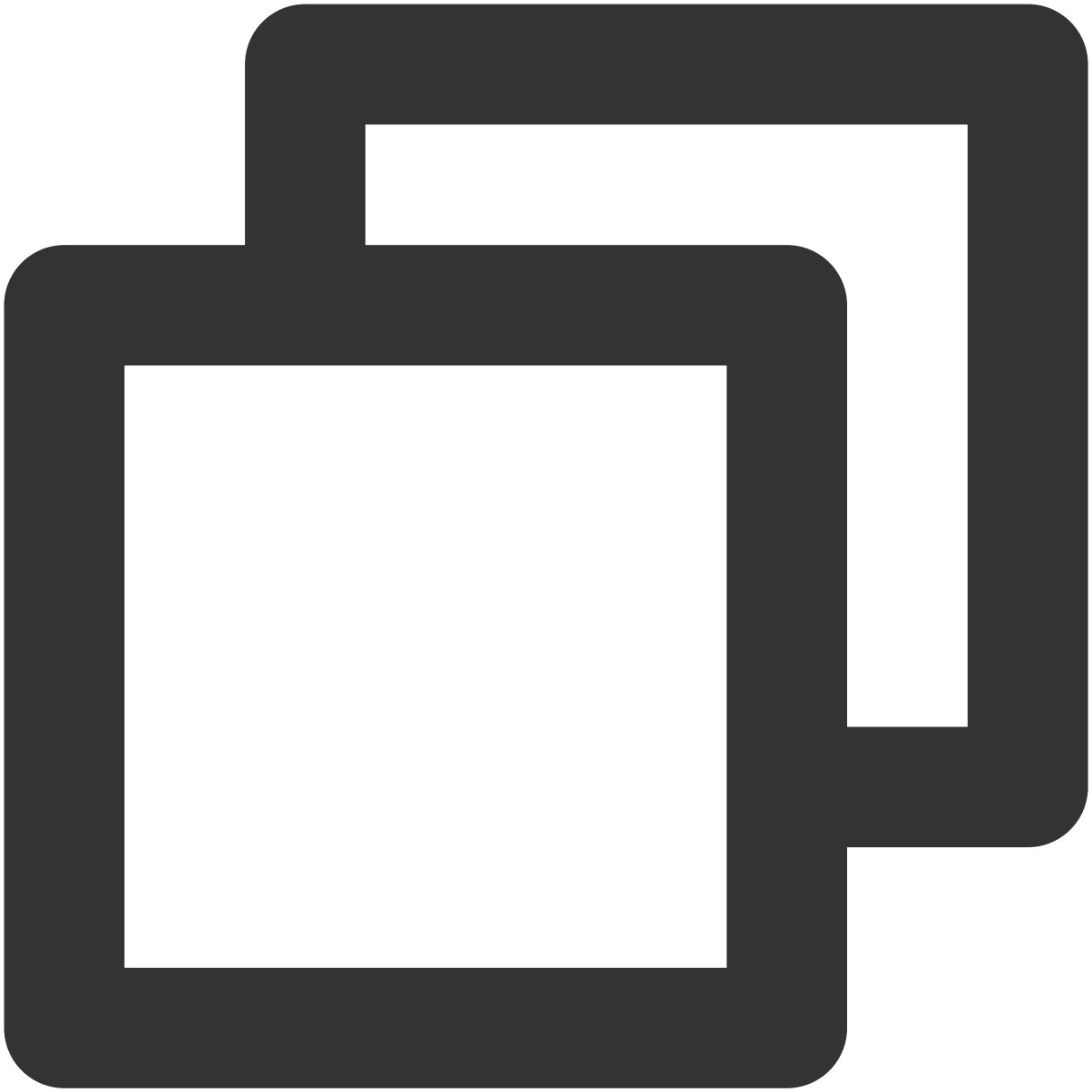
©2013-2022 Tencent Cloud. All rights reserved.



Step 1: The anchor enters the room to push streams

The control used by the TRTC SDK to display video streams only supports passing in a `TXCloudVideoView` type. Therefore, you need to first define the view rendering control in the layout file.

©2013-2022 Tencent Cloud. All rights reserved.



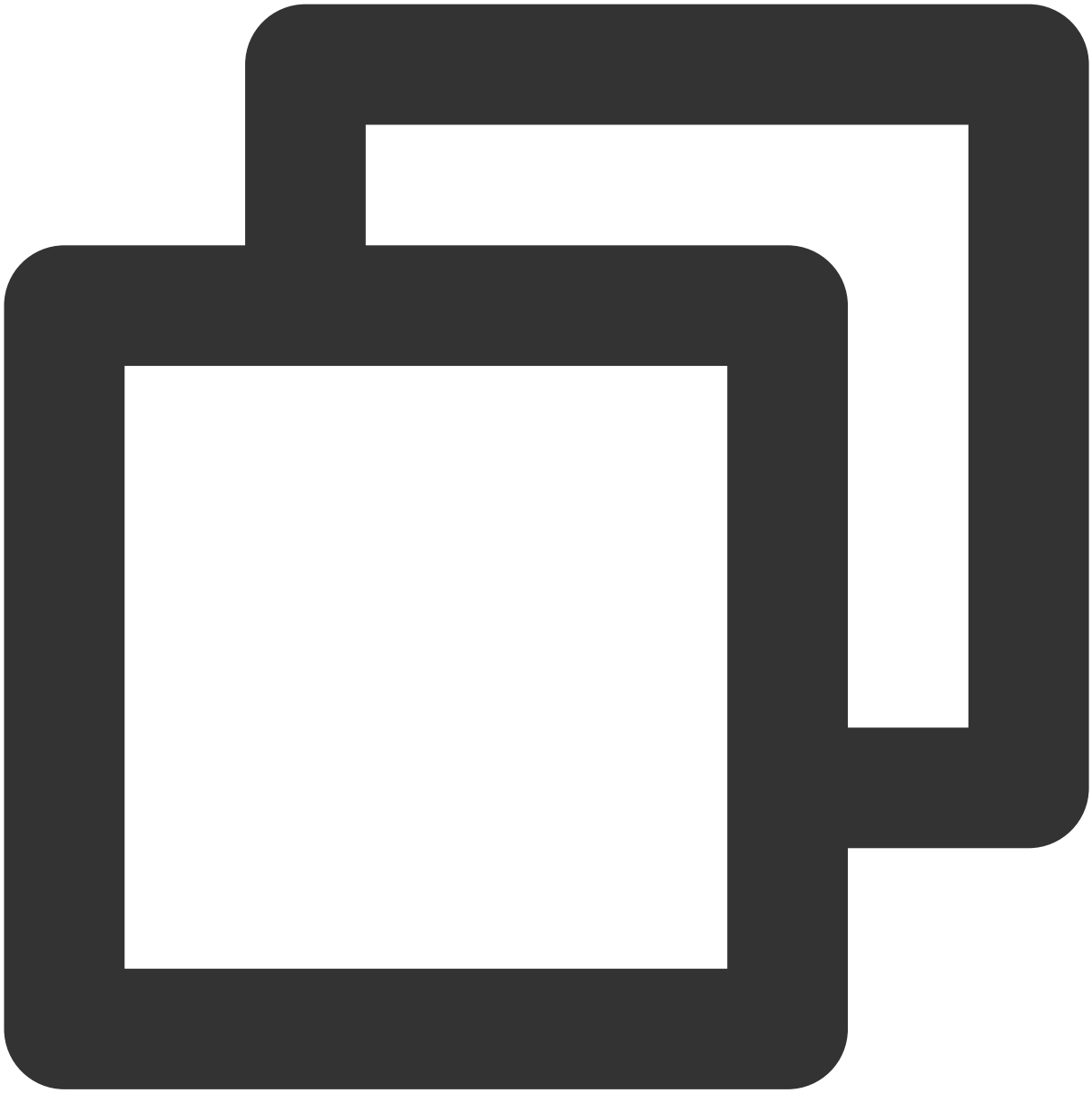
```
<com.tencent.rtmp.ui.TXCloudVideoView
    android:id="@+id/live_cloud_view_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Note:

If you need to specifically use `TextureView` or `SurfaceView` as the view rendering control, see Advanced Features - View Rendering Control.

1. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
TXCloudVideoView mTxcvvAnchorPreviewView = findViewById(R.id.live_cloud_view_main);

// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
encParam.videoFps = 15;
encParam.videoBitrate = 1300;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAnchorPreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

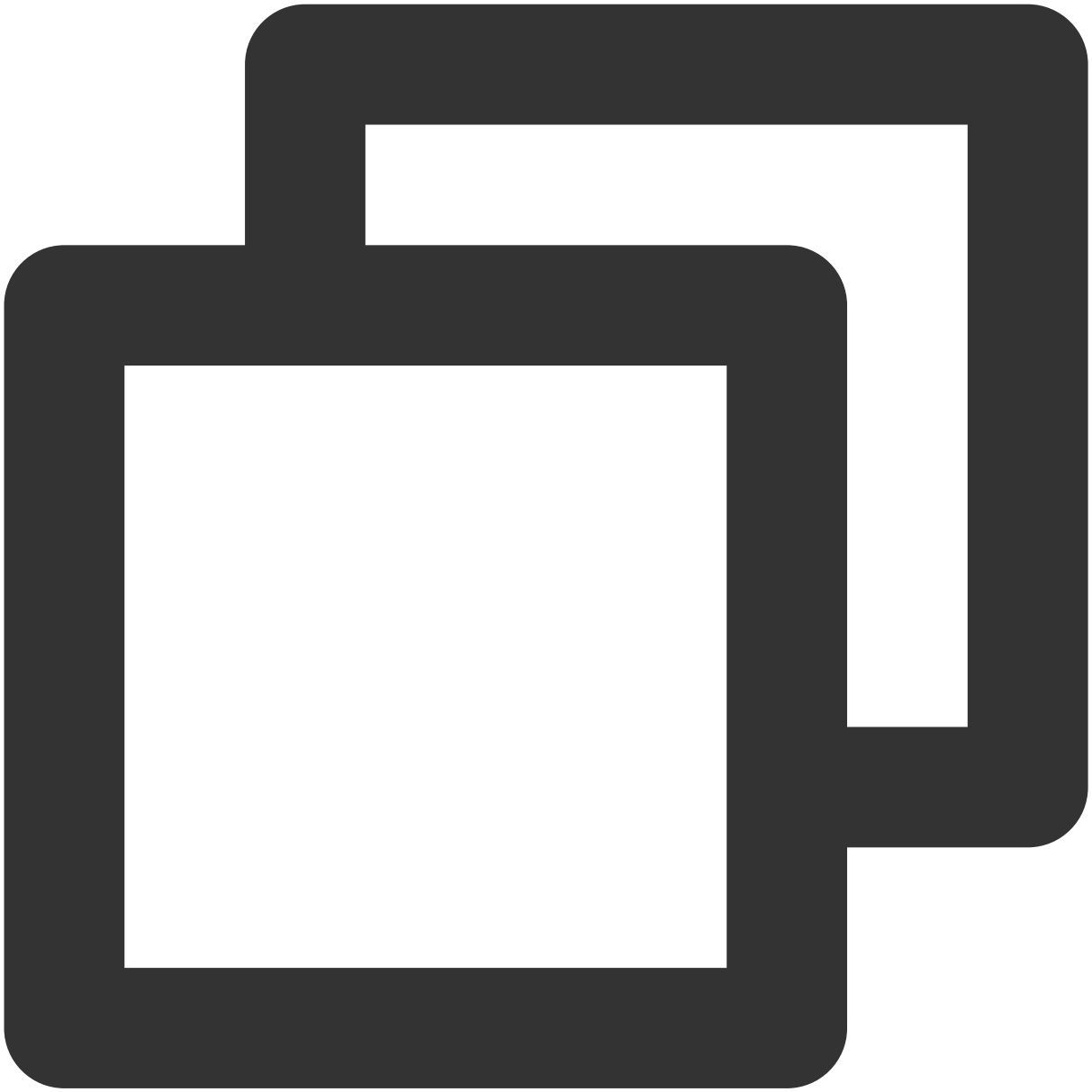
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

2. The anchor sets rendering parameters for the local video, and the encoder output video mode (optional).

©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL; // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering parameters for the local video
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output
mTRTCCloud.setVideoEncoderMirror(boolean mirror);
// Set the rotation of the video encoder output
mTRTCCloud.setVideoEncoderRotation(int rotation);
```

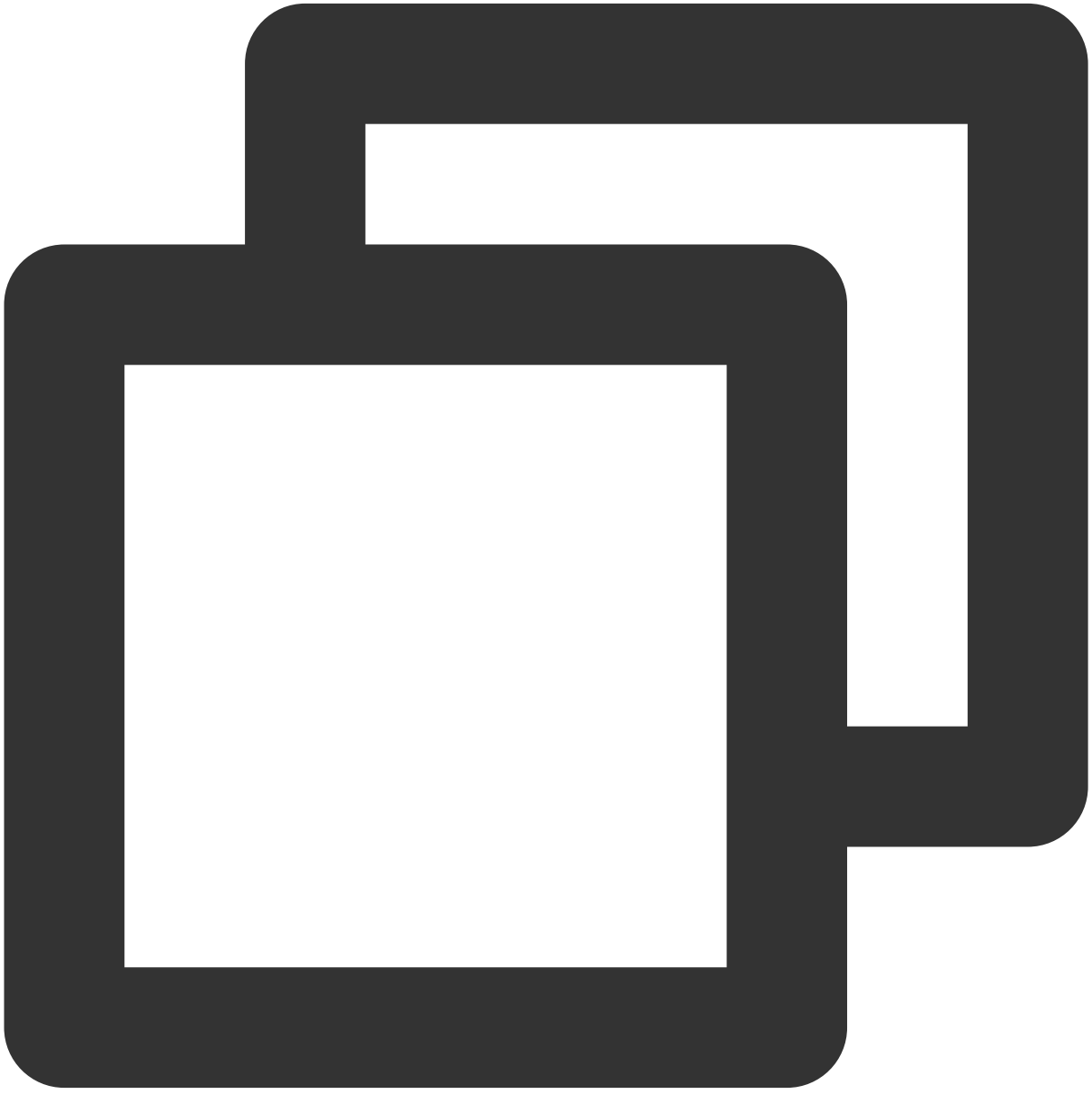
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Setting local screen rendering parameters only affects the rendering effect of the local screen.

Setting encoder output pattern affects the viewing effect for other users in the room (as well as the cloud recording files).

3. The anchor starts the live streaming, entering the room and start streaming.



```
public void enterRoomByAnchor(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example
    params.strRoomId = roomId;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID
    params.sdkAppId = SDKAppID;
    // Specify the anchor role
    params.role = TRTCCloudDef.TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        Log.d(TAG, "Enter room succeed");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

TRTC room IDs are divided into digit type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

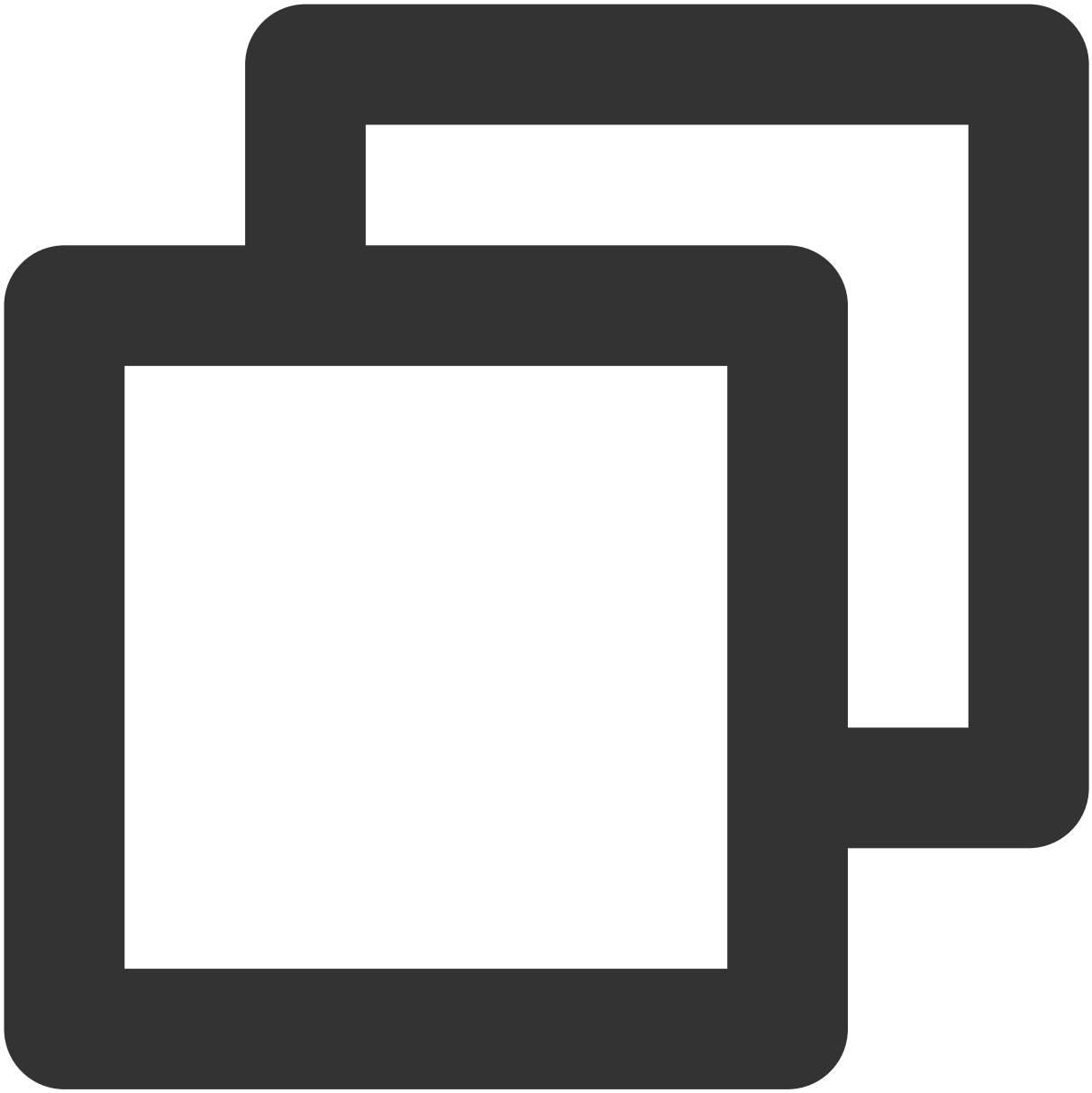
TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In e-commerce live streaming scenarios, it is recommended to choose `TRTC_APP_SCENE_LIVE` as the room entry mode.

Step 2: The audience enters the room to pull streams

1. Audience enters the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



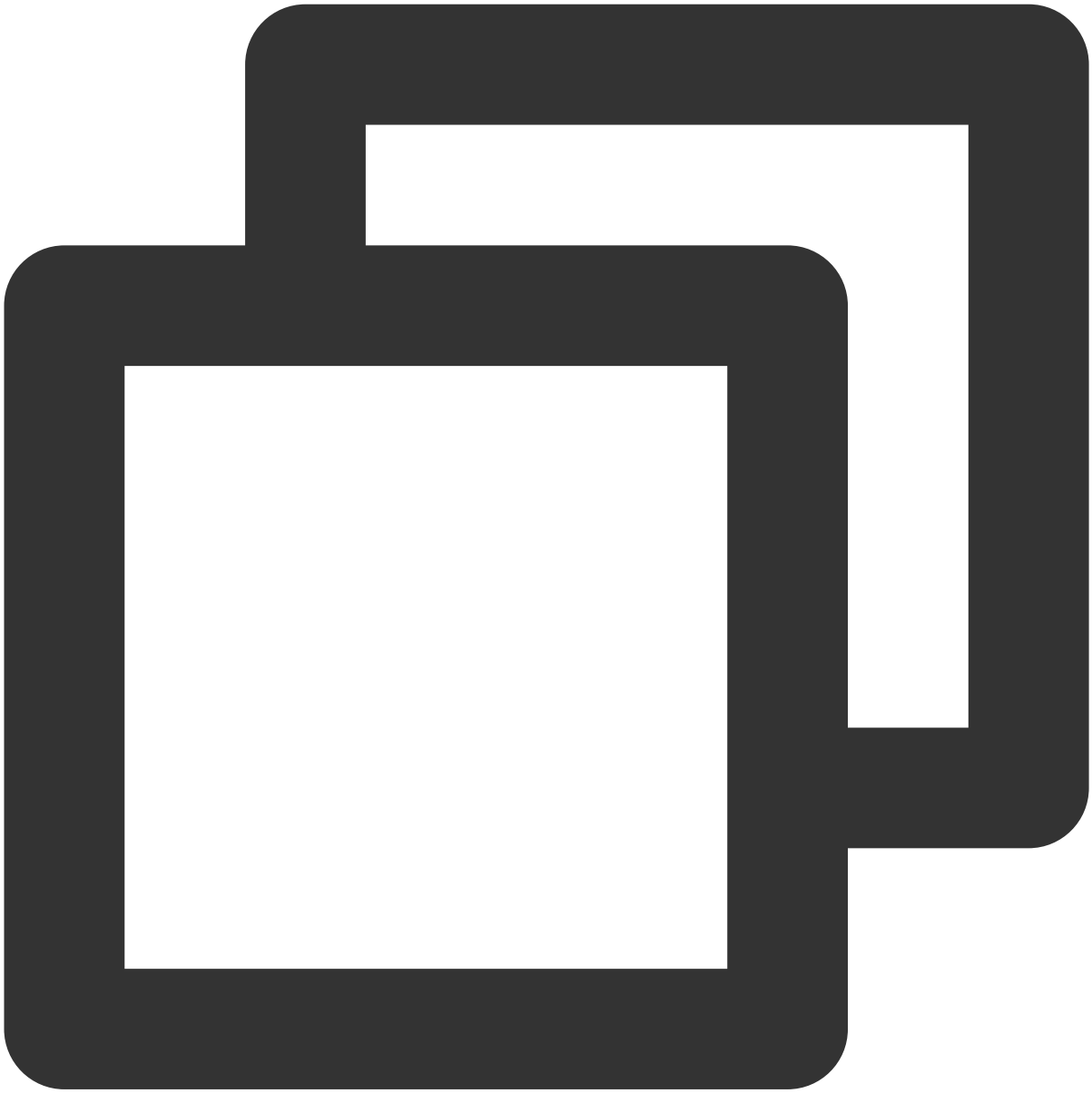
```
public void enterRoomByAudience(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID
    params.sdkAppId = SDKAppID;
    // Specify the audience role
    params.role = TRTCCloudDef.TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        Log.d(TAG, "Enter room succeed");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        Log.d(TAG, "Enter room failed");
    }
}
```

2. Audience subscribes to the anchor's audio and video streams.

©2013-2022 Tencent Cloud. All rights reserved.

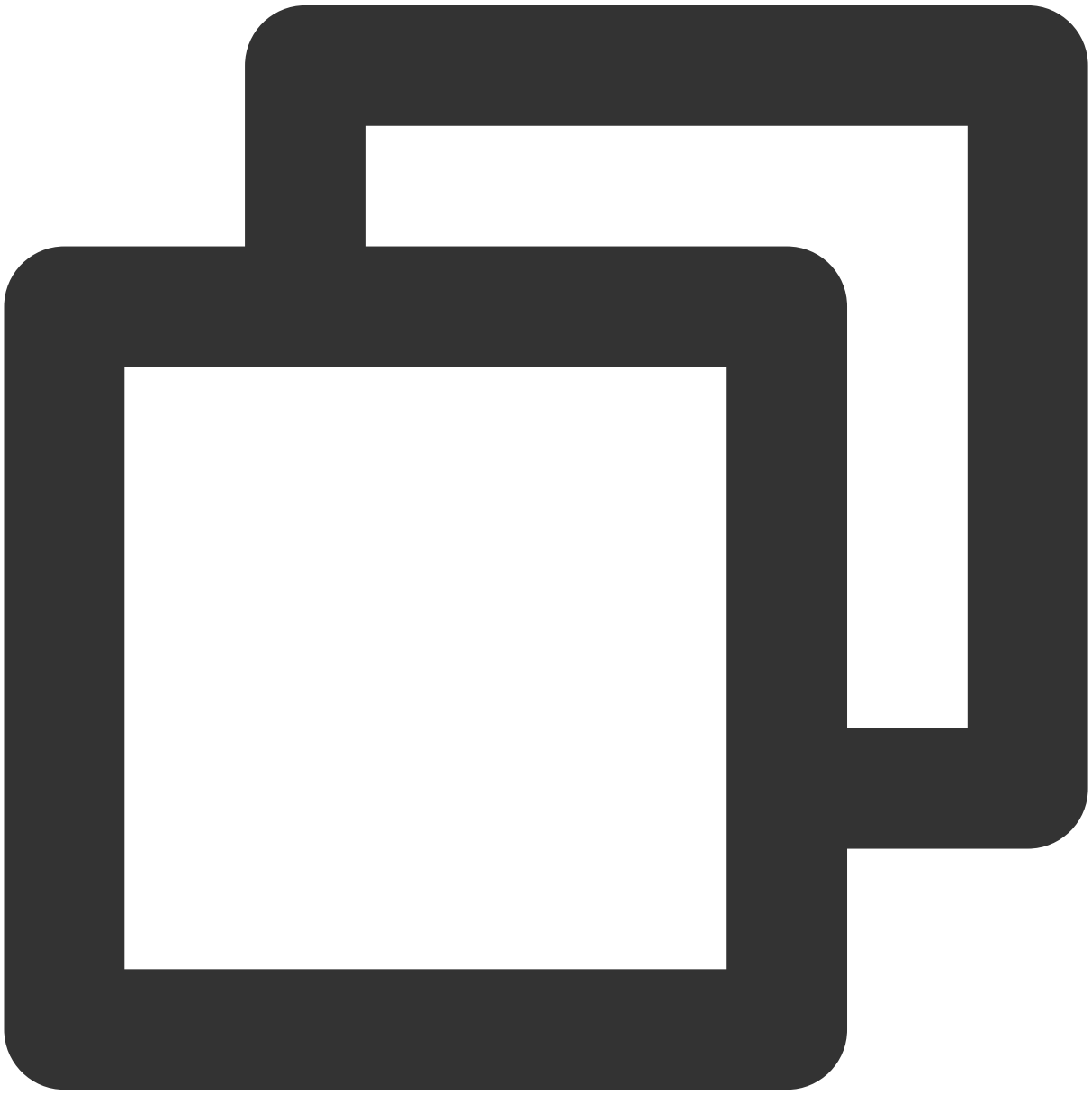


```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }
```

3. Audience sets the rendering mode for the remote video (optional).



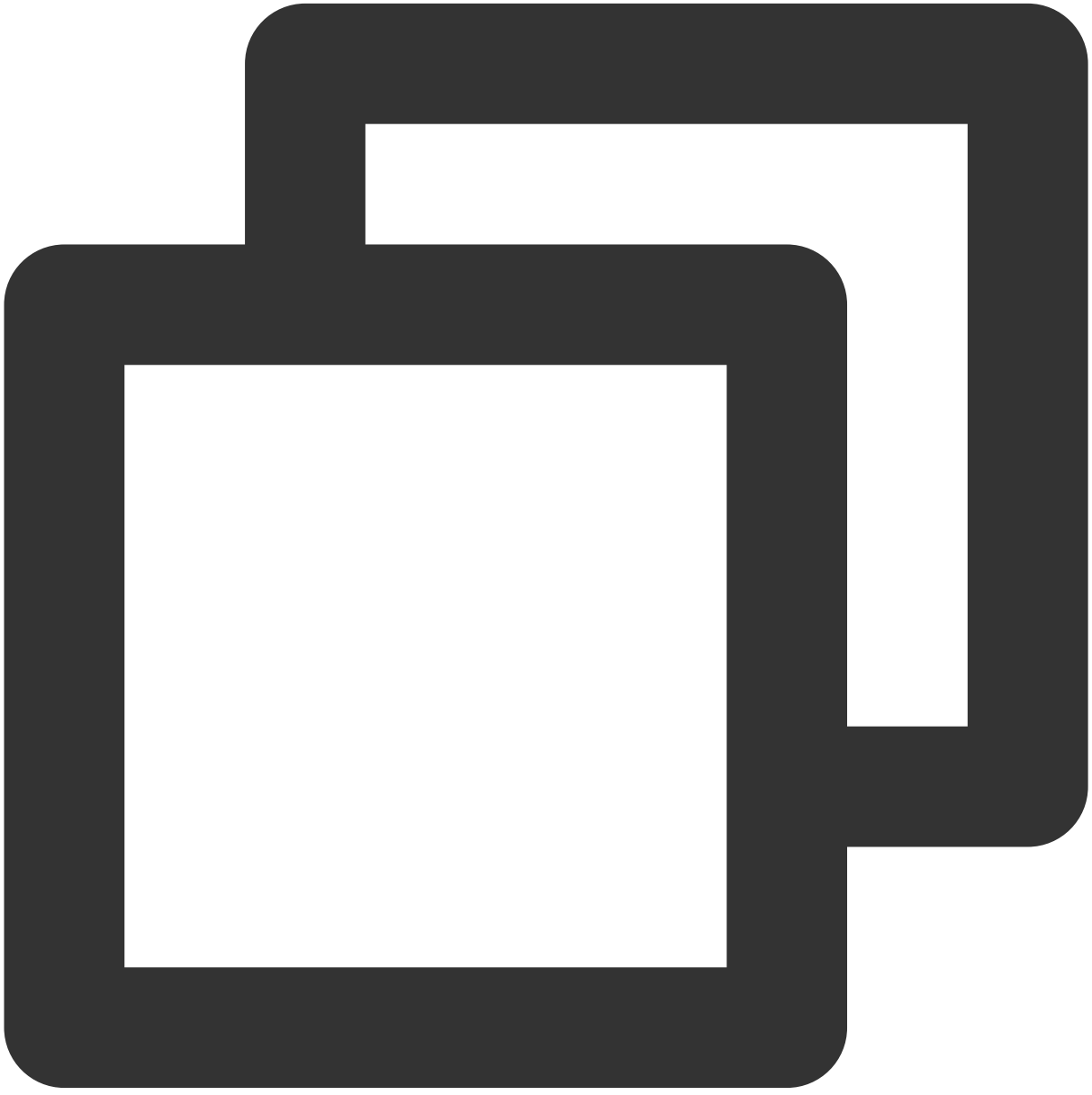
```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL; // Video fill mode
```

©2013-2022 Tencent Cloud. All rights reserved.

```
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering mode for the remote video
mTRTCCloud.setRemoteRenderParams(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, p
```

Step 3: The audience interacts via mic-connection

1. The audience is switched to the anchor role.

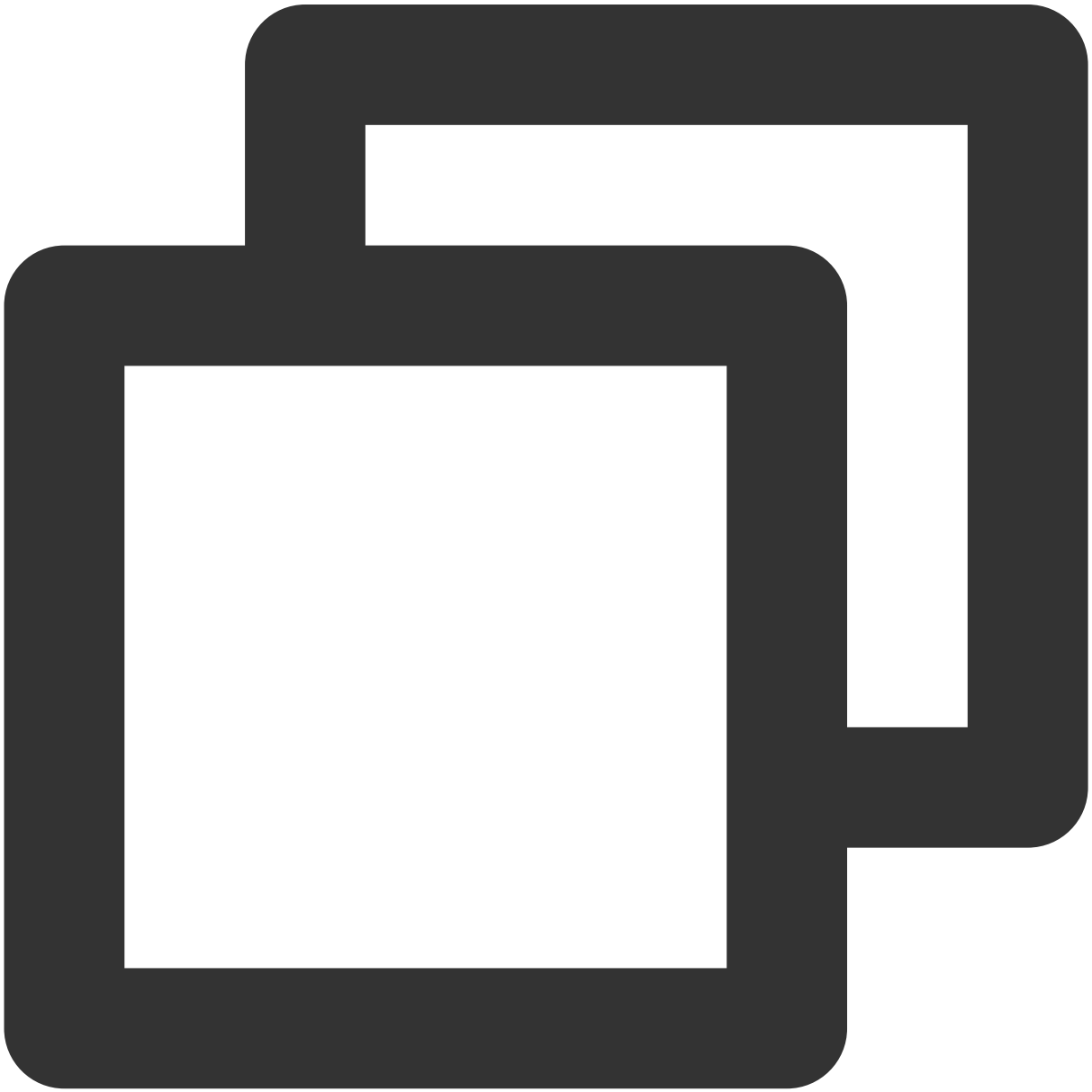


```
// Switch to the anchor role
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Event callback for switching role
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Role switched successfully
    }
}
```

2. The audience starts local audio and video capture and streaming.



```
// Obtain the video rendering control for displaying the mic-connection audience's
TXCloudVideoView mTxcvvAudiencePreviewView = findViewById(R.id.live_cloud_view_sub)
```

©2013-2022 Tencent Cloud. All rights reserved.

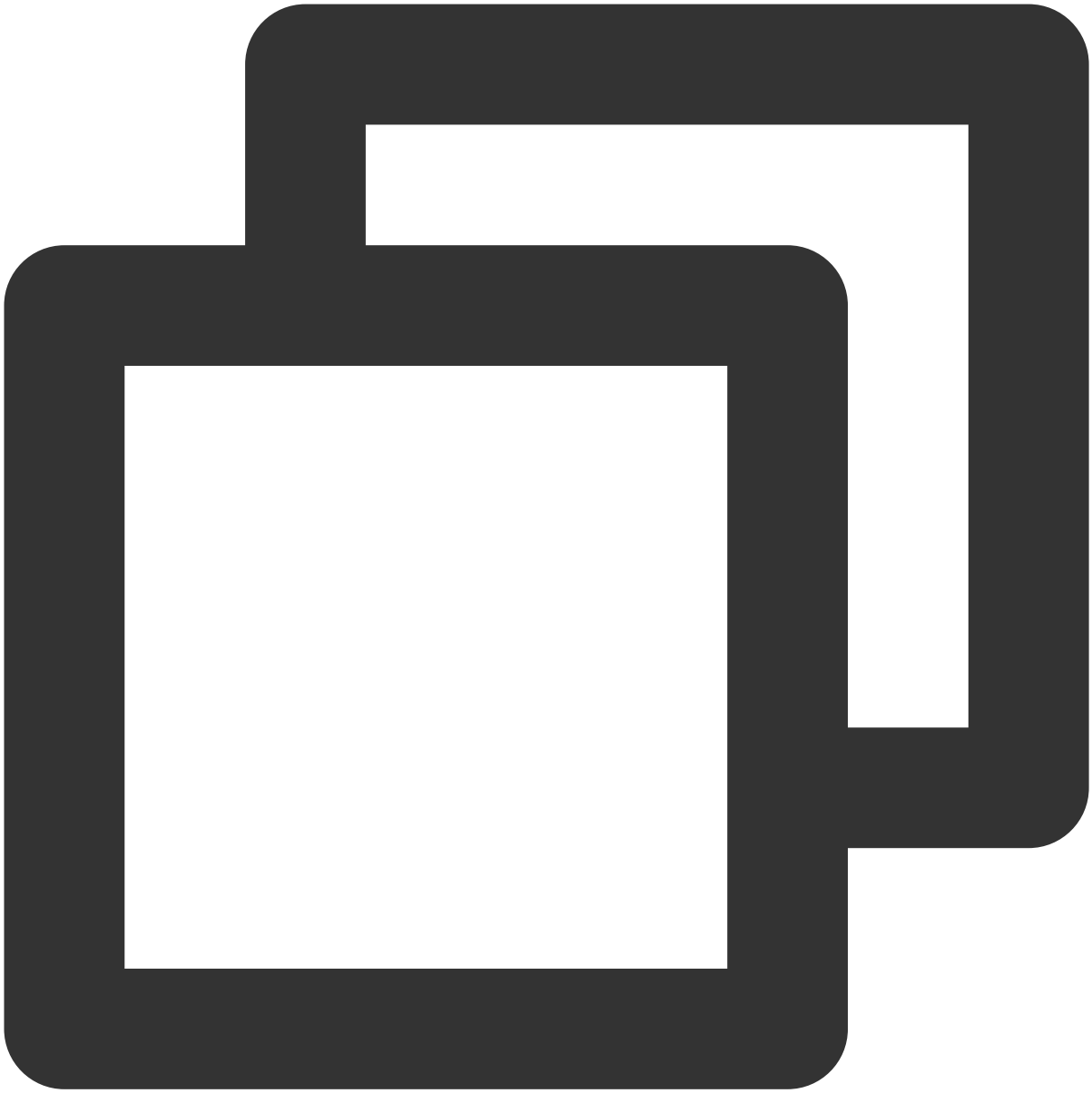
```
// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270;
encParam.videoFps = 15;
encParam.videoBitrate = 550;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);

// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAudiencePreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

3. The audience drops the mic and stops streaming.

©2013-2022 Tencent Cloud. All rights reserved.



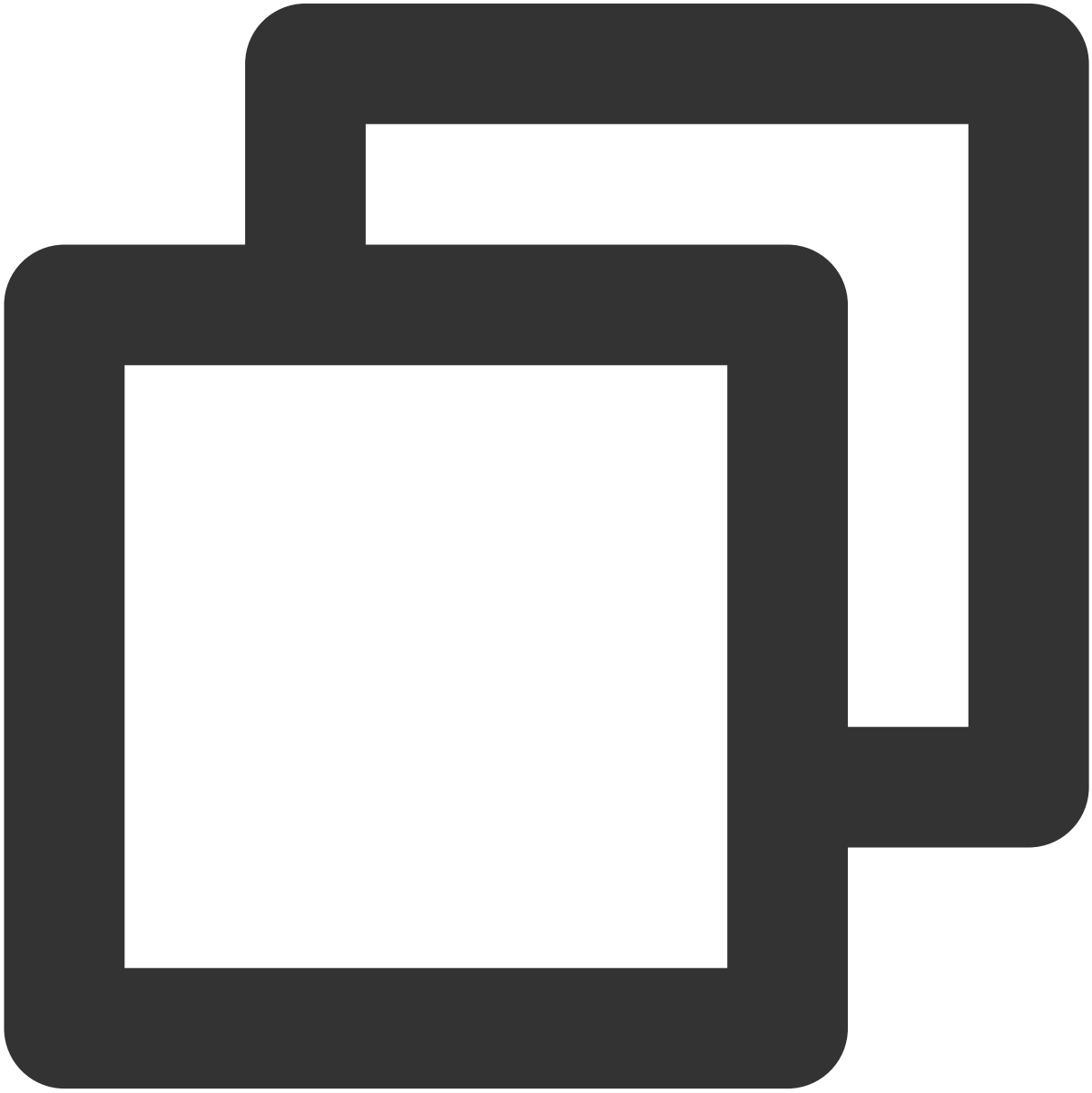
```
// Switch to the audience role
mTRTCCloud.switchRole(TRTCCloudDef.TRTCRoleAudience);

// Event callback for switching role
@Override
public void onSwitchRole(int errCode, String errMsg) {
    if (errCode == TXLiteAVCode.ERR_NULL) {
        // Stop camera capture and streaming
        mTRTCCloud.stopLocalPreview();
        // Stop microphone capture and streaming
        mTRTCCloud.stopLocalAudio();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
}
```

Step 4: Exit and dissolve the room

1. Exit Room



```
public void exitRoom() {
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.stopLocalPreview();
    mTRTCCloud.exitRoom();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}

// Event callback for exiting the room
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server");
    } else if (reason == 2) {
        Log.d(TAG, "The current room has been dissolved");
    }
}
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you wish to call `enterRoom` again or switch to another audio and video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter various exceptional issues such as the camera, microphone device being forcibly occupied.

2. Dissolve Room

Server dissolves the room

TRTC provides the server dissolves digit type room API `DismissRoom` , as well as server dissolves string type room API DismissRoomByStrRoomId. You can call the server dissolves the room API to remove all users from the room and dissolve the room.

Client dissolves the room

The client does not have a API to directly dissolve the room. Each client needs to call exitRoom to exit the room. Once all anchors and audience have exited, the room will automatically be dissolved according to TRTC's room lifecycle rules. For more details, see TRTC Exits Room.

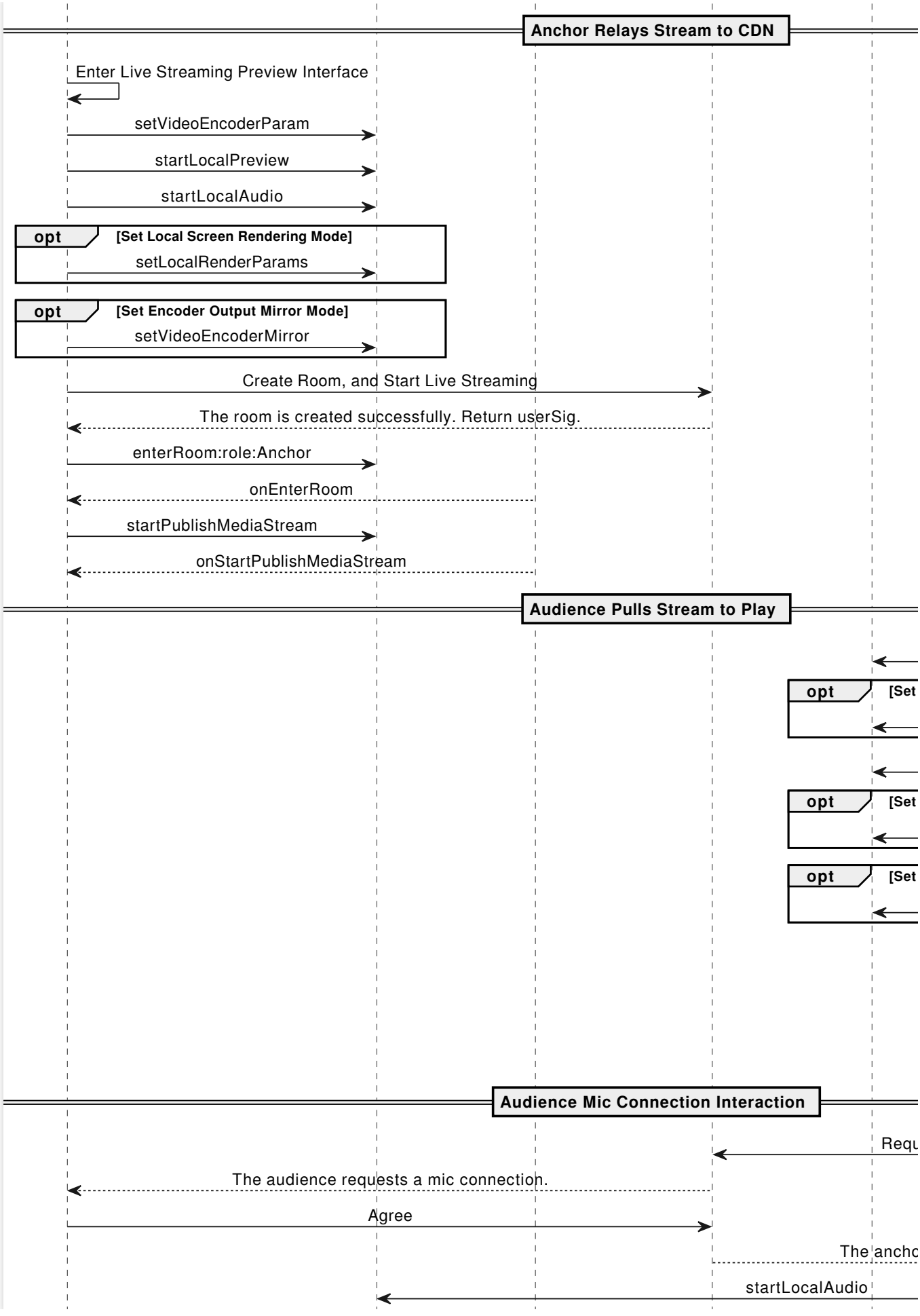
Warning:

It is recommended that after the end of live streaming, you call the room dissolvement API on the server to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

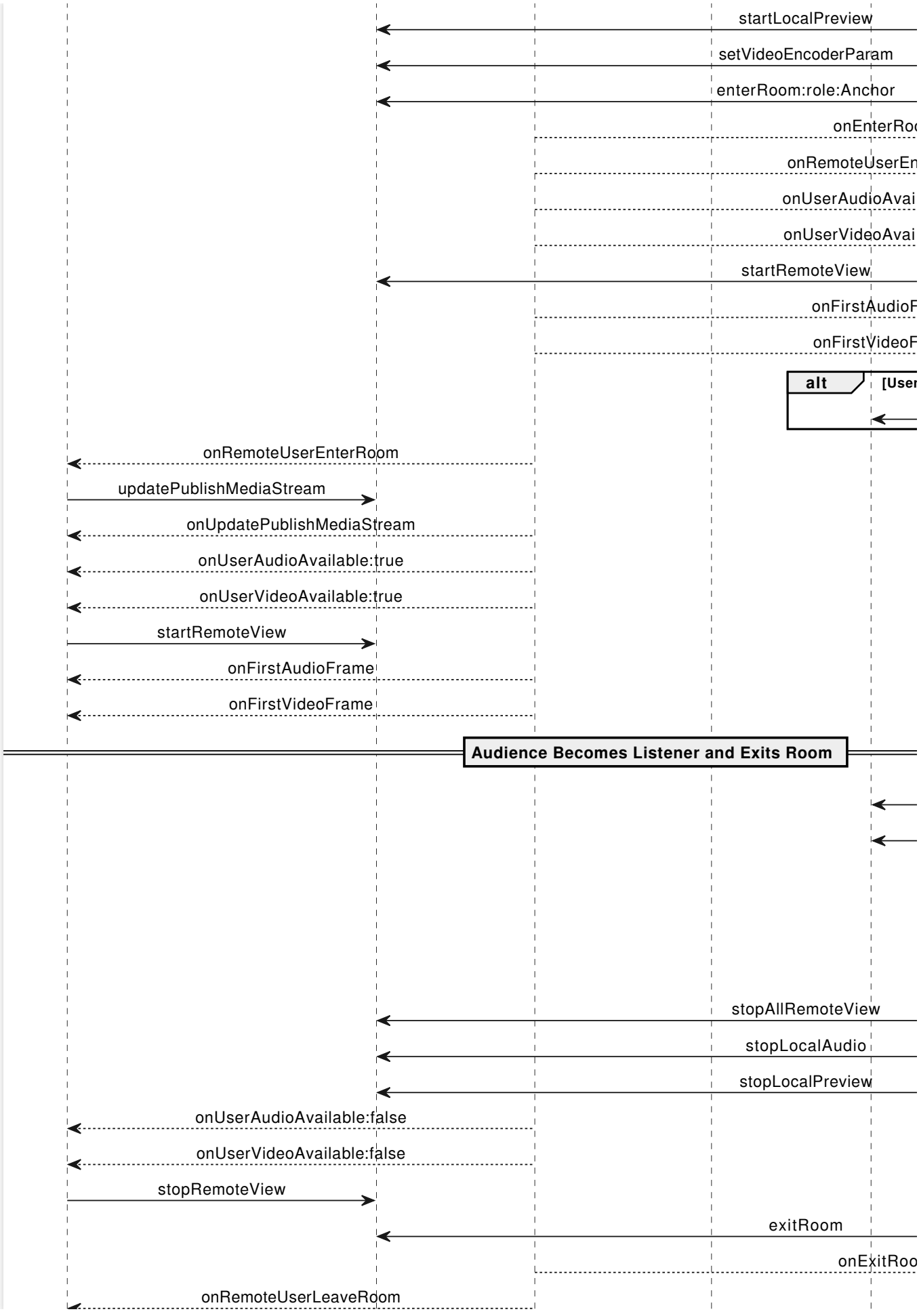
Alternative Solutions

API Sequence Diagram

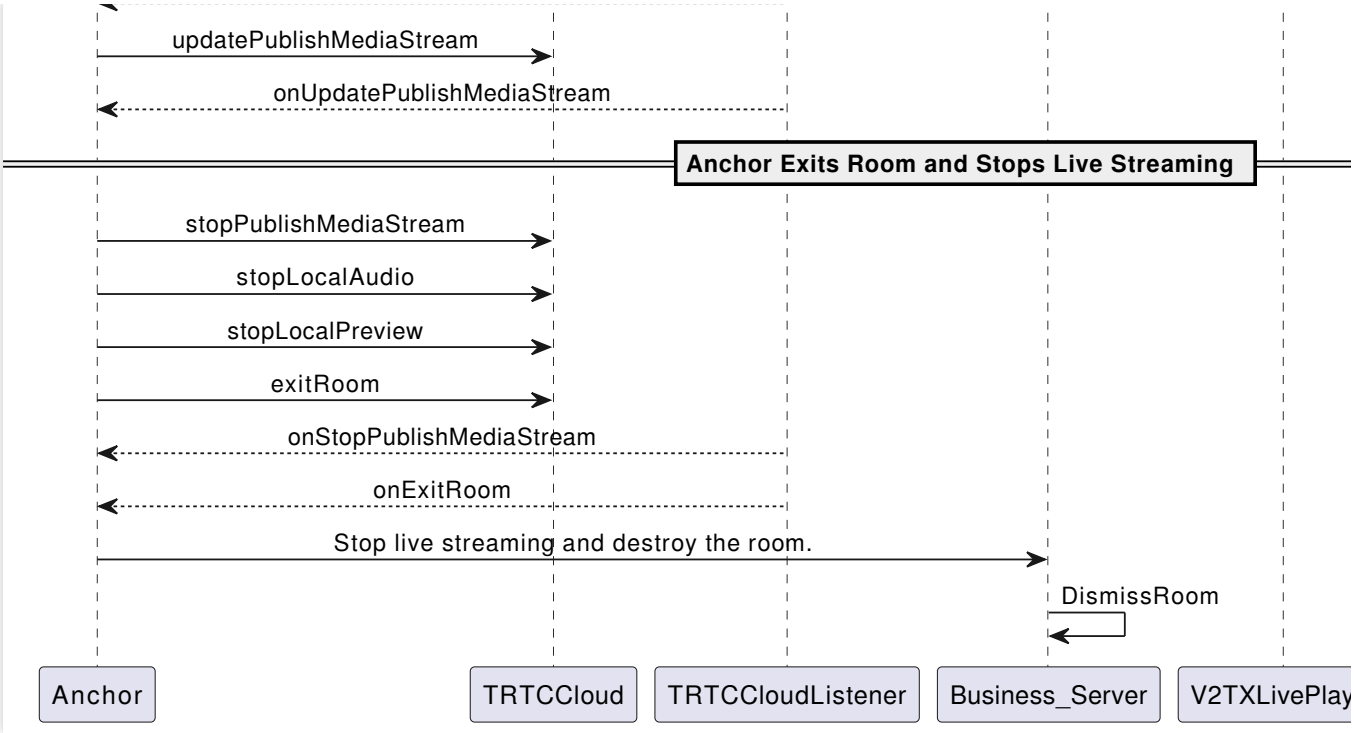
©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



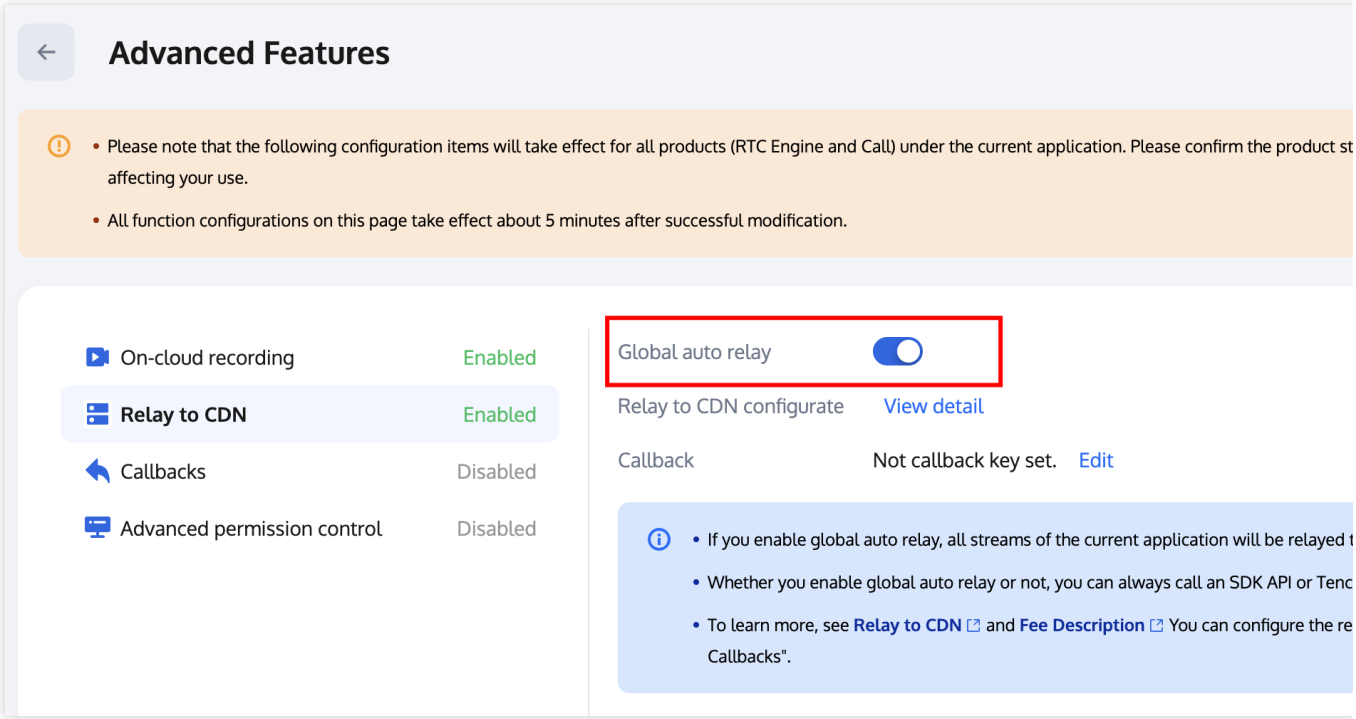
Step 1: The anchor relays stream pushing

1. Related configurations for relaying to live streaming CDN.

Global Automatic Relayed Push

If you need to automatically relay all anchors' audio and video streams in the room to live streaming CDN, you just need to enable **Relay to CDN** on the **Advanced Features** page in the TRTC Console.

©2013-2022 Tencent Cloud. All rights reserved.



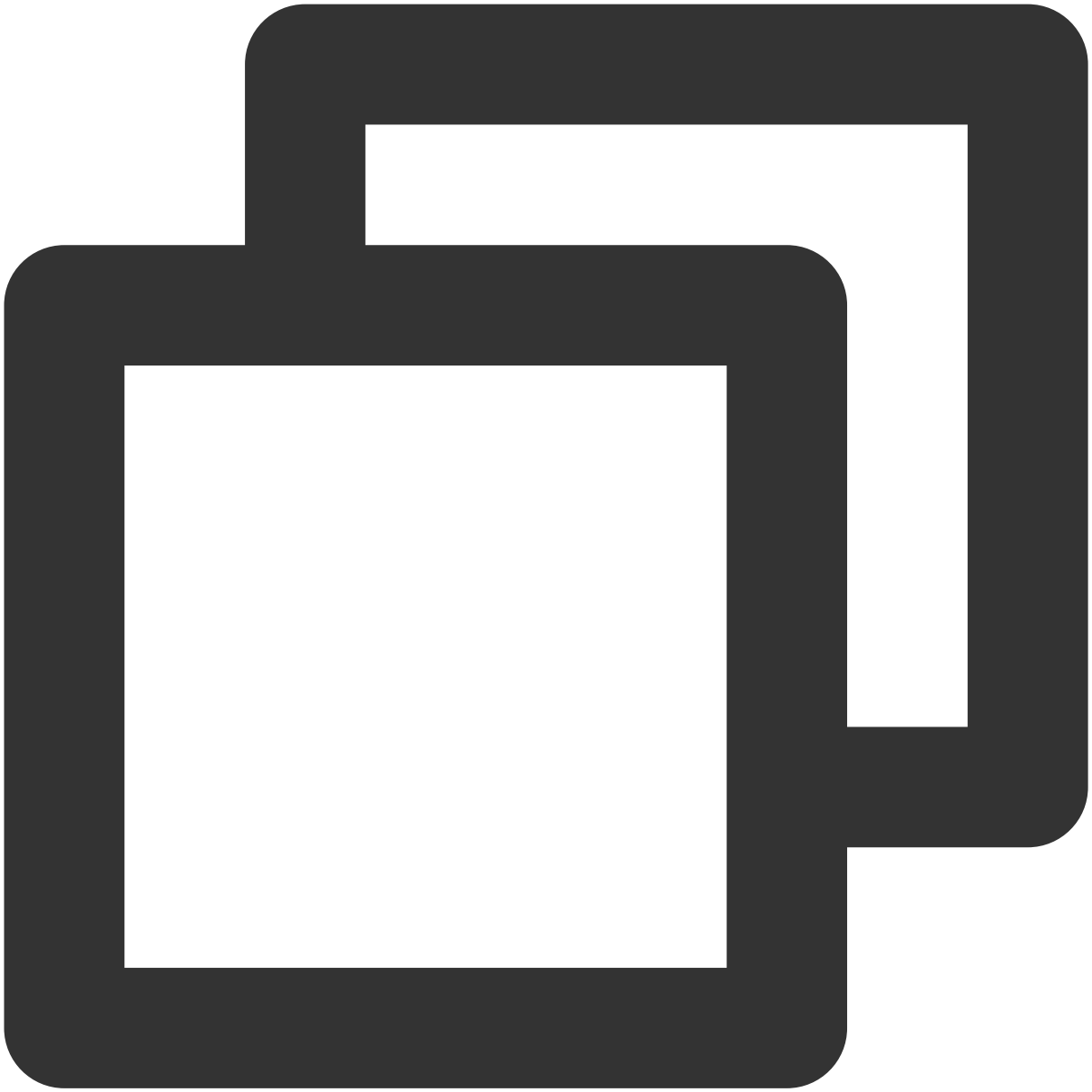
Relayed Push of the Specified Streams

If you need to manually specify the audio and video streams to be published to live streaming CDN, or publish the mixed audio and video streams to live streaming CDN, you can do so by calling the startPublishMediaStream API. In this case, you do not need to activate global automatically relaying to CDN in the console. For a detailed introduction, see Publish Audio and Video Streams to Live Streaming CDN.

2. The anchor activates local video preview and audio capture before entering the room.

The control used by the TRTC SDK to display video streams only supports passing in a `TXCloudVideoView` type. Therefore, you need to first define the view rendering control in the layout file.

©2013-2022 Tencent Cloud. All rights reserved.

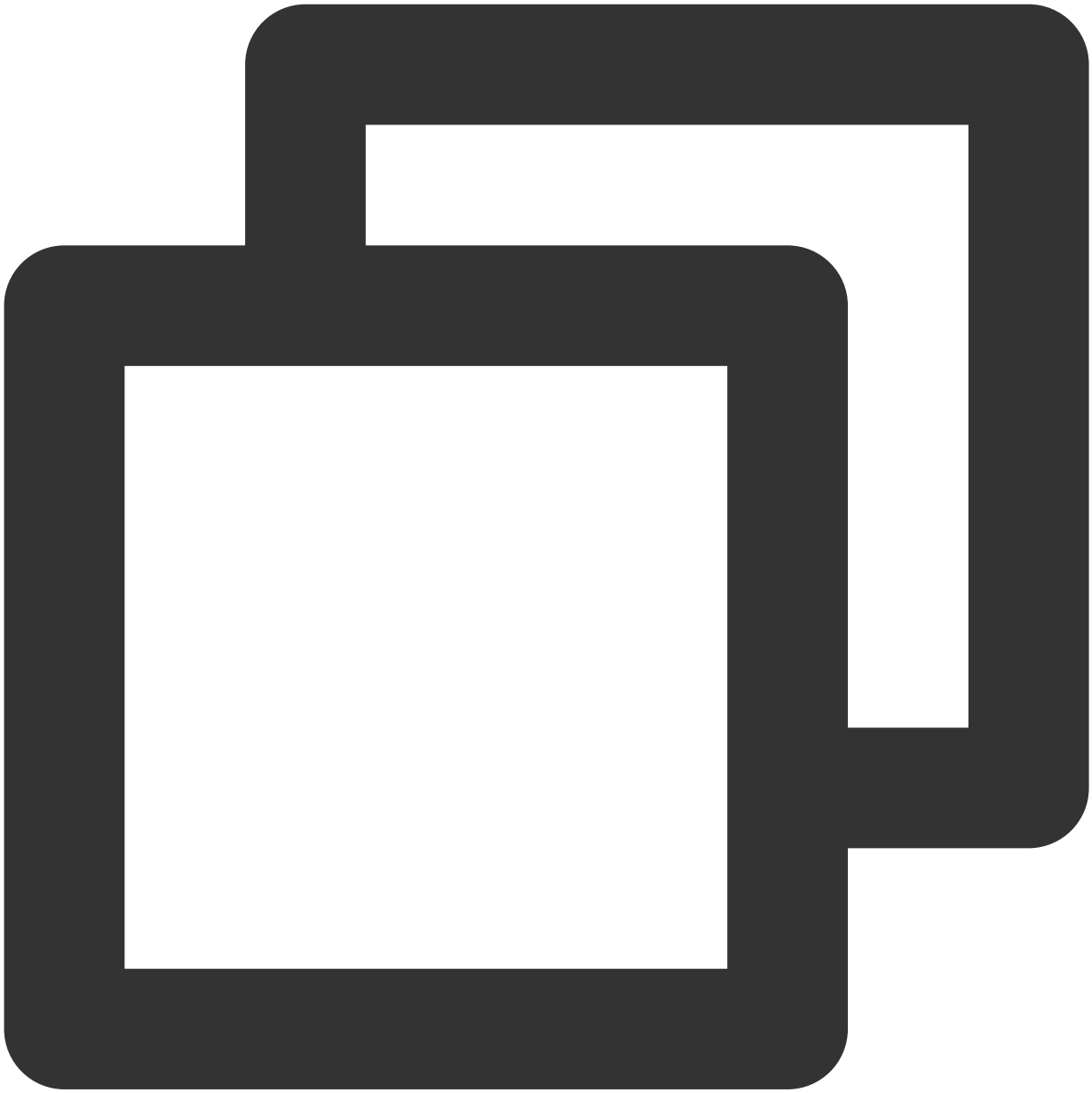


```
<com.tencent.rtmp.ui.TXCloudVideoView
    android:id="@+id/live_cloud_view_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Note:

If you need to specifically use `TextureView` or `SurfaceView` as the view rendering control, see Advanced Features - View Rendering Control.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
TXCloudVideoView mTxcvvAnchorPreviewView = findViewById(R.id.live_cloud_view_main);

// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
encParam.videoFps = 15;
encParam.videoBitrate = 1300;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// boolean mIsFrontCamera can specify using the front/rear camera for video capture
mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAnchorPreviewView);

// Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/MUSIC
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
```

Note:

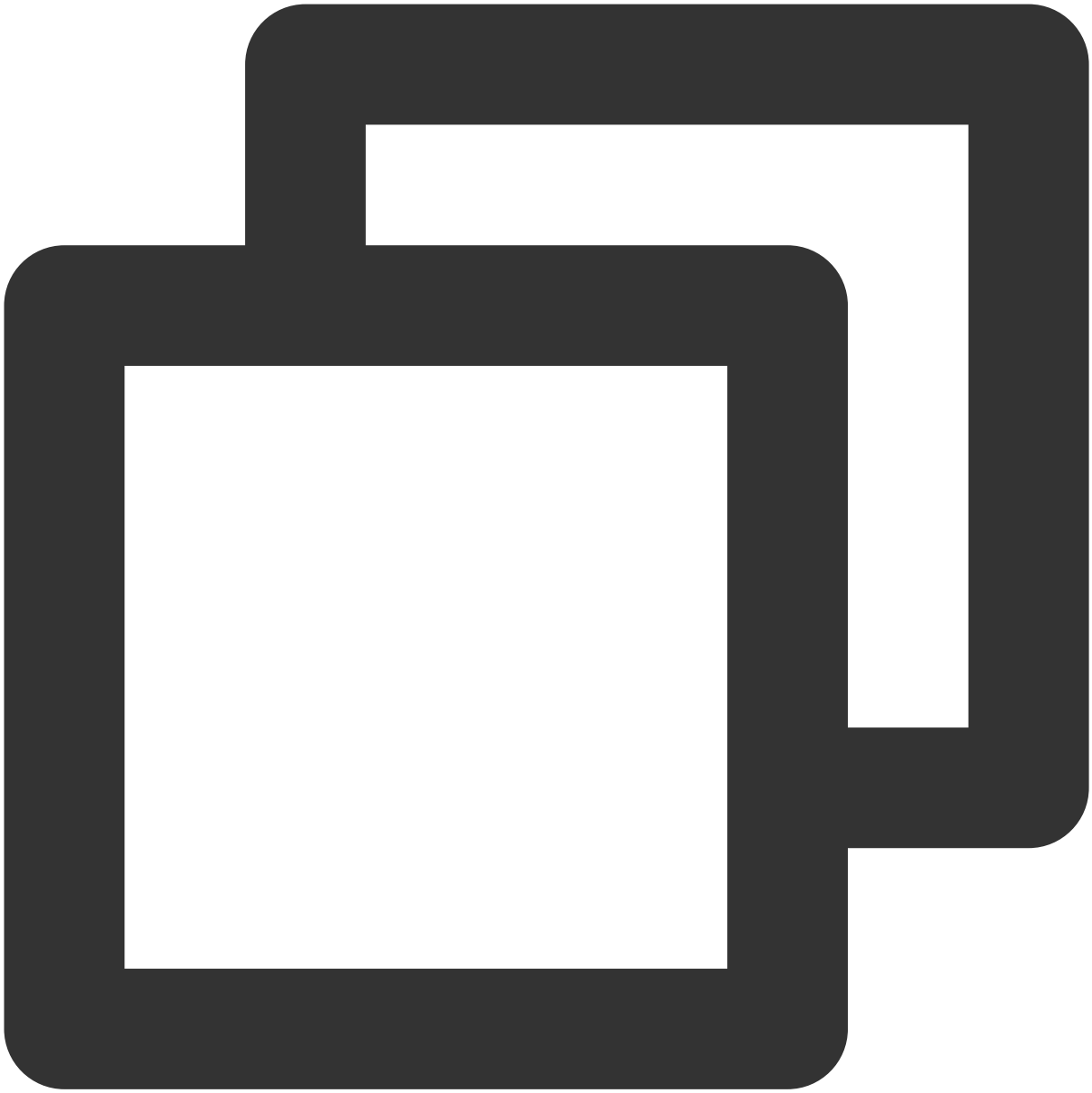
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

3. The anchor sets rendering parameters for the local screen, and the encoder output video mode.

©2013-2022 Tencent Cloud. All rights reserved.



```
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;   // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL; // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
// Set the rendering parameters for the local video
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output
mTRTCCloud.setVideoEncoderMirror(boolean mirror);
// Set the rotation of the video encoder output
mTRTCCloud.setVideoEncoderRotation(int rotation);
```

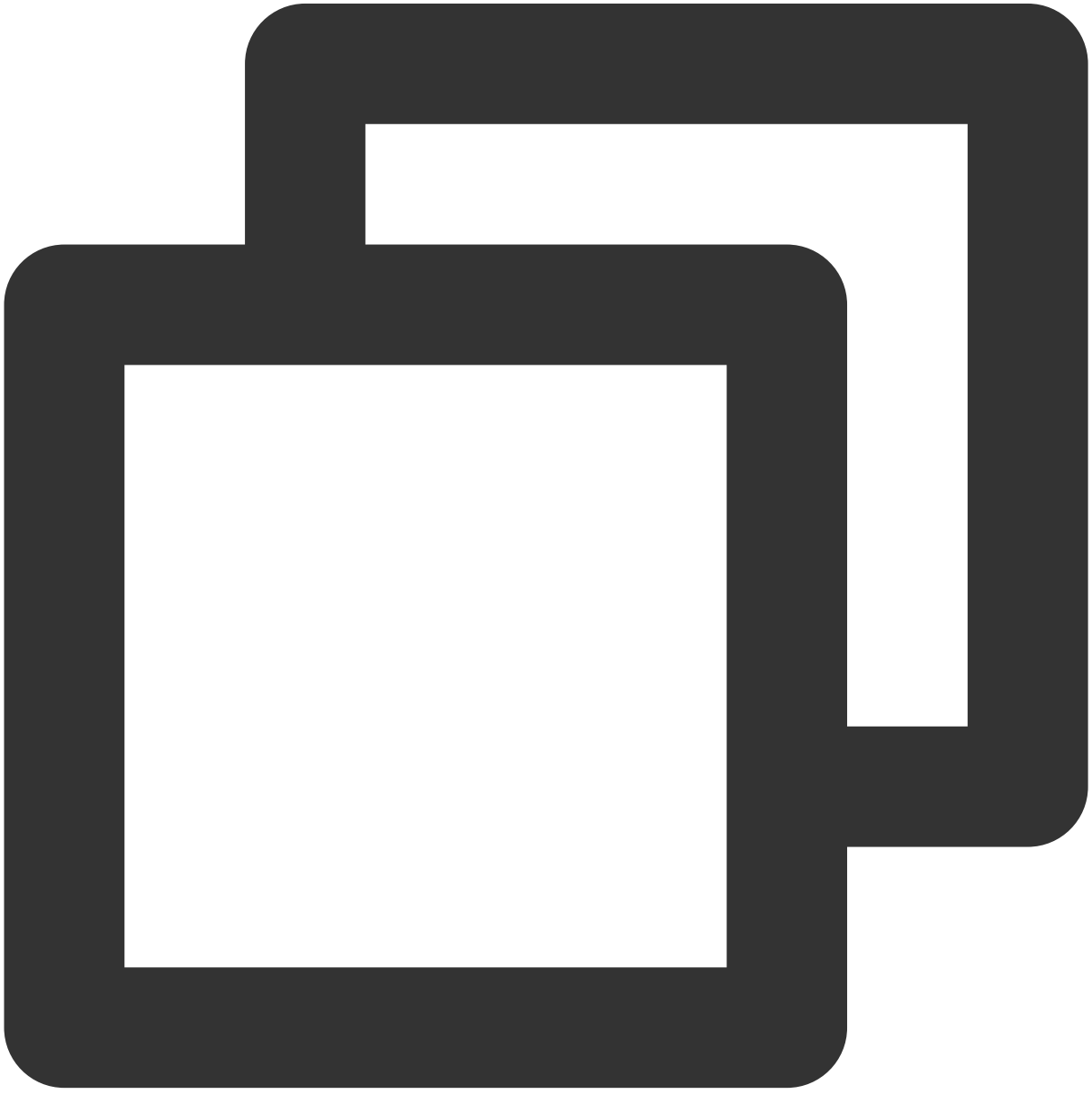
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Setting local screen rendering parameters only affects the rendering effect of the local screen.

Setting encoder output pattern affects the viewing effect for other users in the room (as well as the cloud recording files).

4. The anchor starts the live streaming, entering the room and start streaming.



```
public void enterRoomByAnchor(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example
    params.strRoomId = roomId;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID
    params.sdkAppId = SDKAppID;
    // Specify the anchor role
    params.role = TRTCCloudDef.TRTCRoleAnchor;
    // Enter the room in an interactive live streaming scenario
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        Log.d(TAG, "Enter room succeed");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

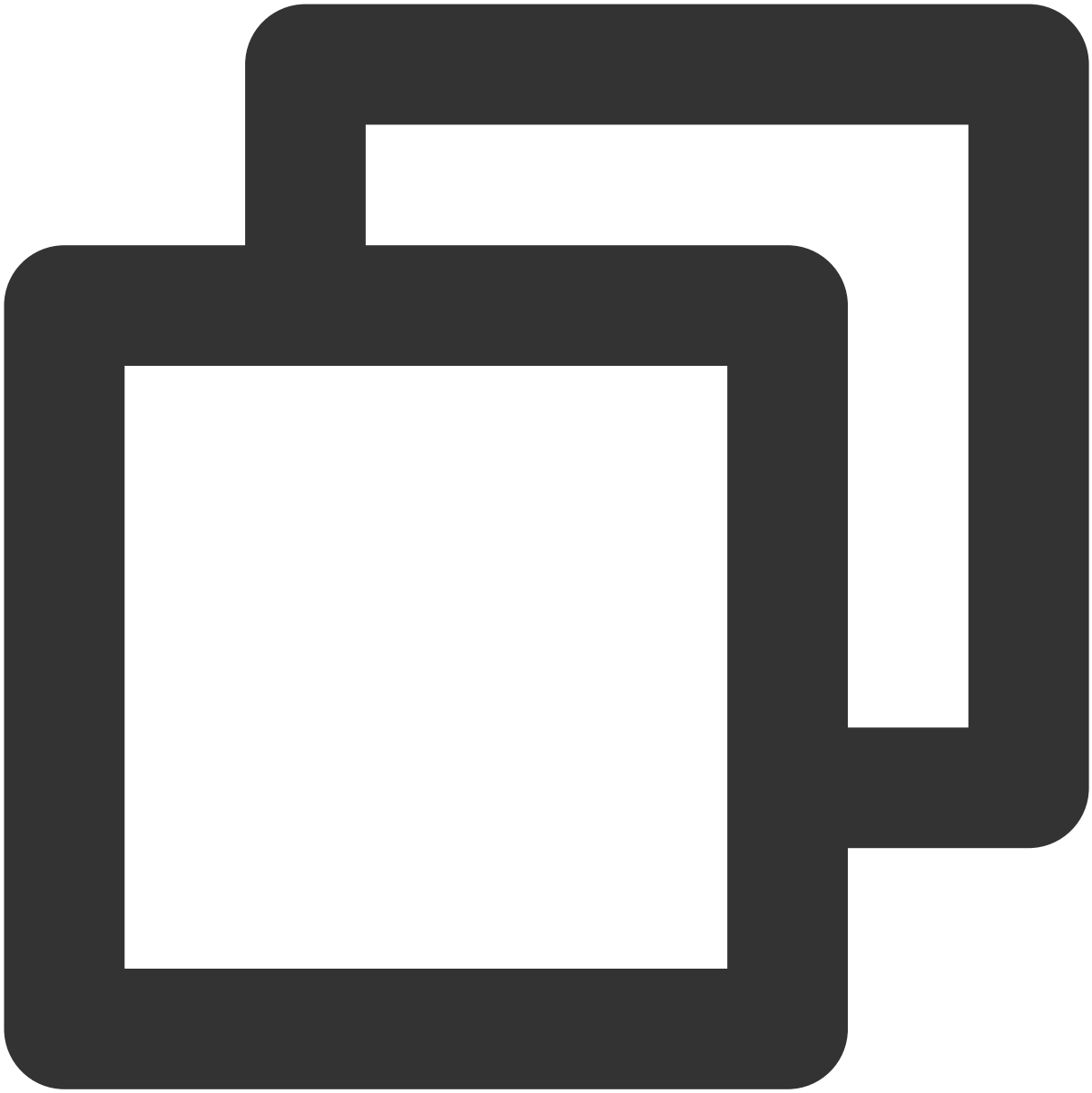
TRTC room IDs are divided into digit type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In e-commerce live streaming scenarios, it is recommended to choose `TRTC_APP_SCENE_LIVE` as the room entry mode.

5. The anchor relays the audio and video streams to the live streaming CDN.

©2013-2022 Tencent Cloud. All rights reserved.



```
public void startPublishMediaToCDN(String streamName) {
    // Set the expiration time for the push URLs
    long txTime = (System.currentTimeMillis() / 1000) + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    String secretParam = UrlHelper.getSafeUrl(LIVE_URL_KEY, streamName, txTime);

    // The target URLs for media stream publication
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // The target URLs are set for relaying to CDN
    target.mode = TRTCCloudDef.TRTC_PublishBigStream_ToCdn;
    TRTCCloudDef.TRTCPublishCdnUrl cdnUrl = new TRTCCloudDef.TRTCPublishCdnUrl();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Construct push URLs (in RTMP format) to the live streaming service provider
    cdnUrl.rtmpUrl = "rtmp://" + PUSH_DOMAIN + "/live/" + streamName + "?" + secret
    // True means the cloud platform CSS, and false means third-party live streamin
    cdnUrl.isInternalLine = true;
    // Multiple CDN push URLs can be added
    target.cdnUrlList.add(cdnUrl);

    // Set media stream encoding output parameters (can be defined according to bus
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
    trtcStreamEncoderParam.audioEncodedChannelNum = 1;
    trtcStreamEncoderParam.audioEncodedKbps = 50;
    trtcStreamEncoderParam.audioEncodedCodecType = 0;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 2;
    trtcStreamEncoderParam.videoEncodedKbps = 1300;
    trtcStreamEncoderParam.videoEncodedWidth = 540;
    trtcStreamEncoderParam.videoEncodedHeight = 960;

    // Start publishing media stream
    mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam,  null);
}
```

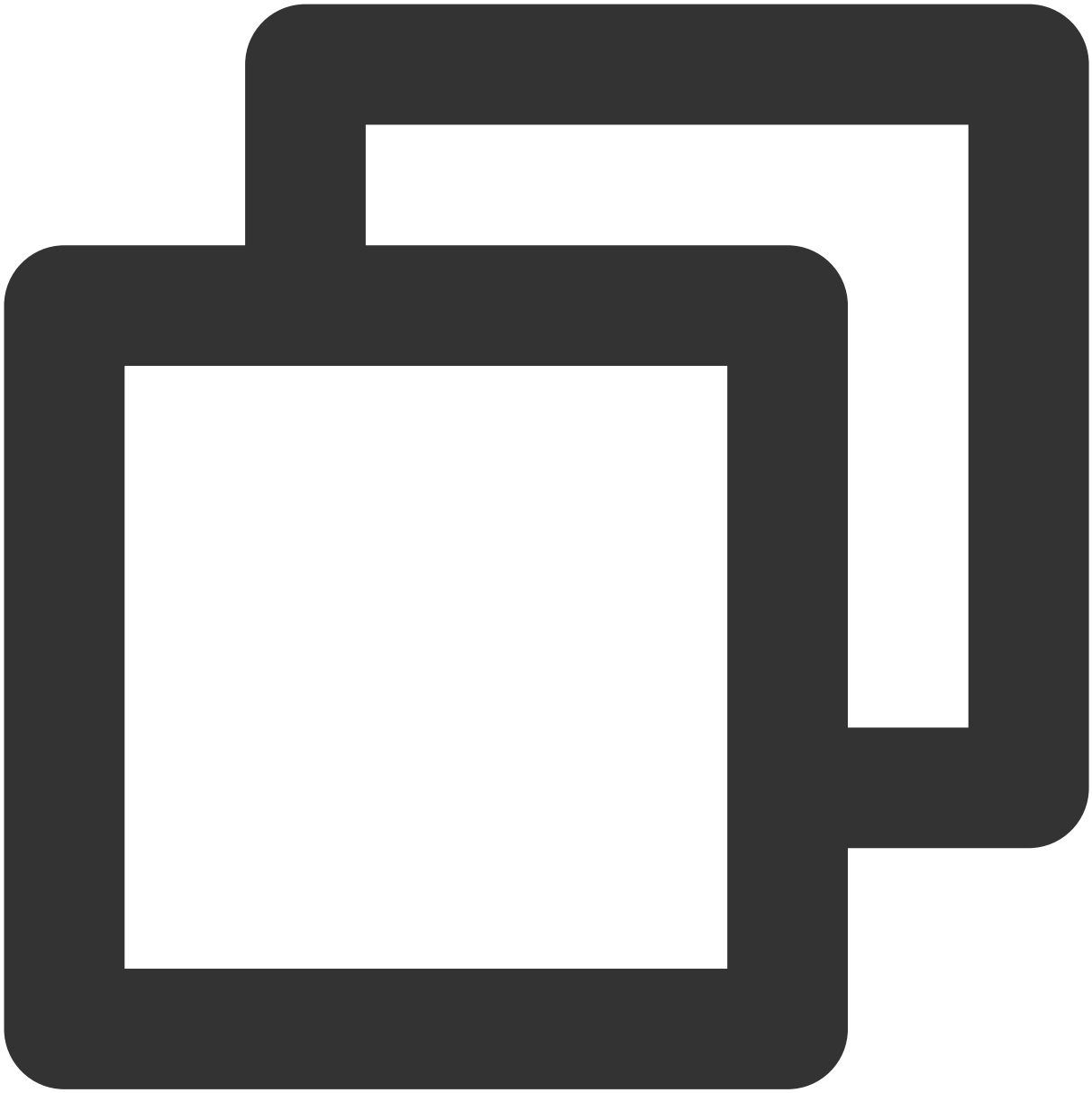
Note:

During single-anchor live streaming, only initiate the relayed push task. When there is an audience mic-connection or anchor PK, update this task to a mixed-stream transcoding task.

Information of push authentication `KEY LIVE_URL_KEY` and push domain name `PUSH_DOMAIN` can be obtained on the **Domain Name Management** page in the CSS Console.

After the media stream is published, SDK will provide the backend-initiated task identifier (taskId) through the callback onStartPublishMediaStream.

©2013-2022 Tencent Cloud. All rights reserved.

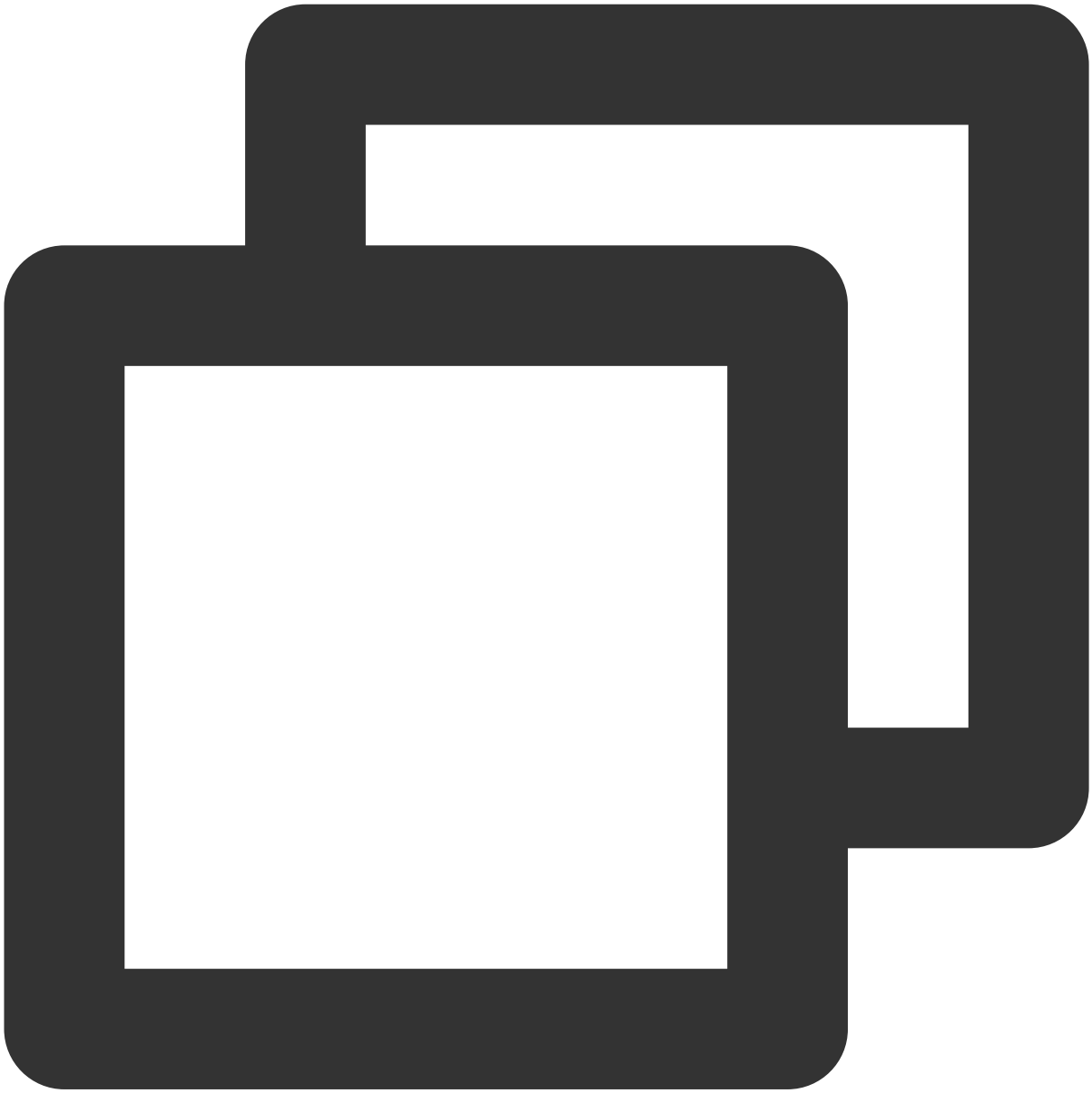


```
@Override
public void onStartPublishMediaStream(String taskId, int code, String message, Bund
    // taskId: When the request is successful, TRTC backend will provide the taskId
    // code: Callback result. 0 means success and other values mean failure
}
```

Step 2: The audience pulls streams for playback

CDN audience do not need to enter the TRTC room; they can directly pull the anchor's CDN stream for playback. In the live streaming playback scenario, see Initialize SDK for player initialization steps.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set delay management mode (optional)
mLivePlayer.setCacheParams(1.0f, 5.0f); // Auto mode
mLivePlayer.setCacheParams(1.0f, 1.0f); // Speed mode
mLivePlayer.setCacheParams(5.0f, 5.0f); // Smooth mode

// Concatenate the pull URLs for playback
String flvURL = "http://" + PLAY_DOMAIN + "/live/" + streamName + ".flv"; // FLV UR
String hlsURL = "http://" + PLAY_DOMAIN + "/live/" + streamName + ".m3u8"; // HLS U
String rtmpURL = "rtmp://" + PLAY_DOMAIN + "/live/" + streamName; // RTMP URL
String webrtcURL = "webrtc://" + PLAY_DOMAIN + "/live/" + streamName; // WebRTC URL
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Start playing
mLivePlayer.startLivePlay(flvURL);

// Custom set fill mode (optional)
mLivePlayer.setRenderFillMode(V2TXLiveFillModeFit);
// Customize video rendering direction (optional)
mLivePlayer.setRenderRotation(V2TXLiveRotation0);
```

Note:

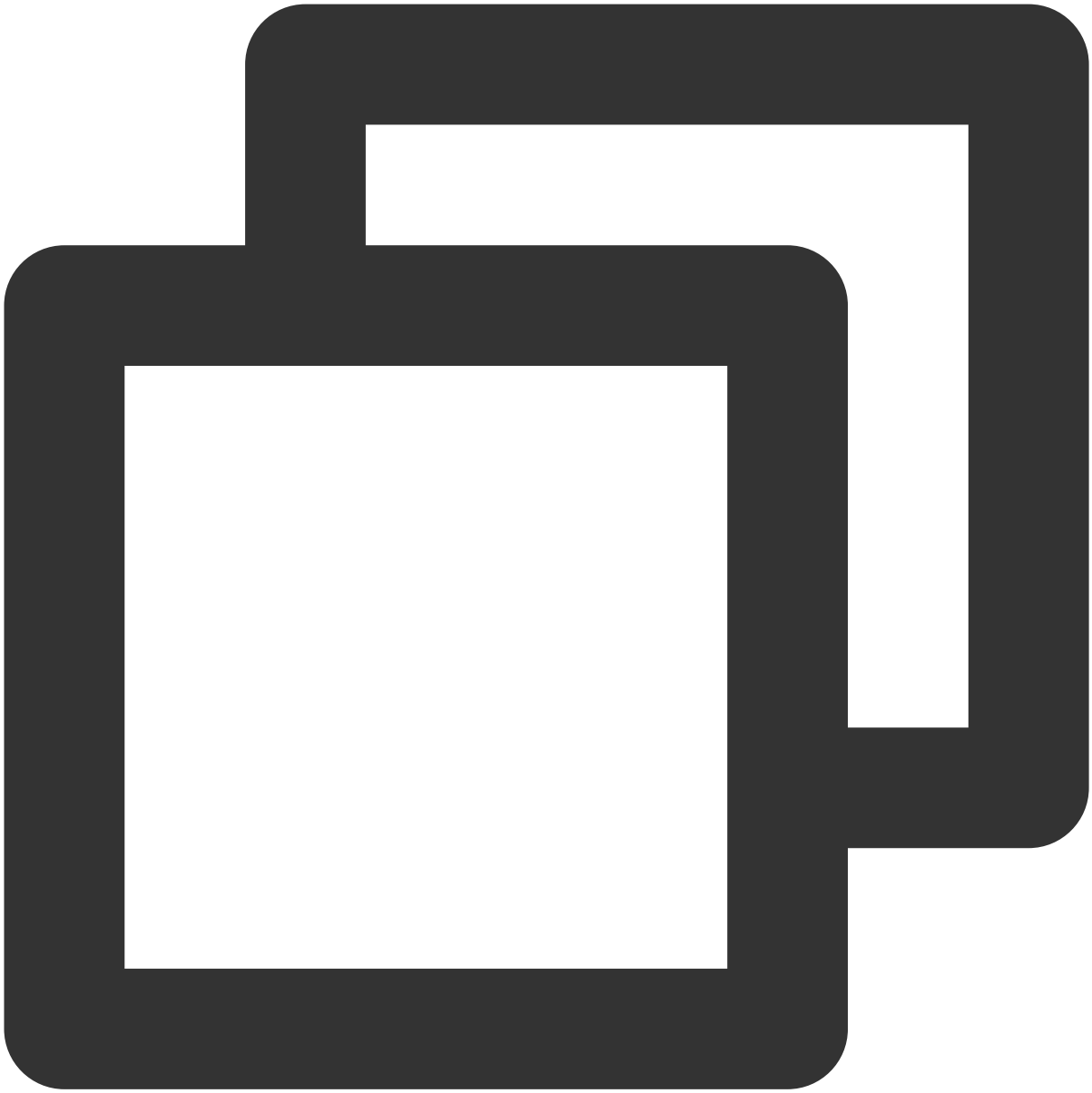
The playback domain name PLAY_DOMAIN requires you to Add Your Own Domain in the CSS console for live streaming playback. You also should configure domain CNAME.

To use the live streaming, you need to configure the player's Licence authorization in advance, or the playback will fail (black screen). For details, see Authentication and Authorization.

Step 3: The audience interacts via mic-connection

1. The mic-connection audiences need to enter the TRTC room for real-time interaction with the anchor.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enter the TRTC room and start streaming
public void enterRoom(String roomId, String userId) {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = getUserSig(userId);
    // Replace with your SDKAppID
    params.sdkAppId = SDKAppID;
    // Specify the anchor role
```

©2013-2022 Tencent Cloud. All rights reserved.

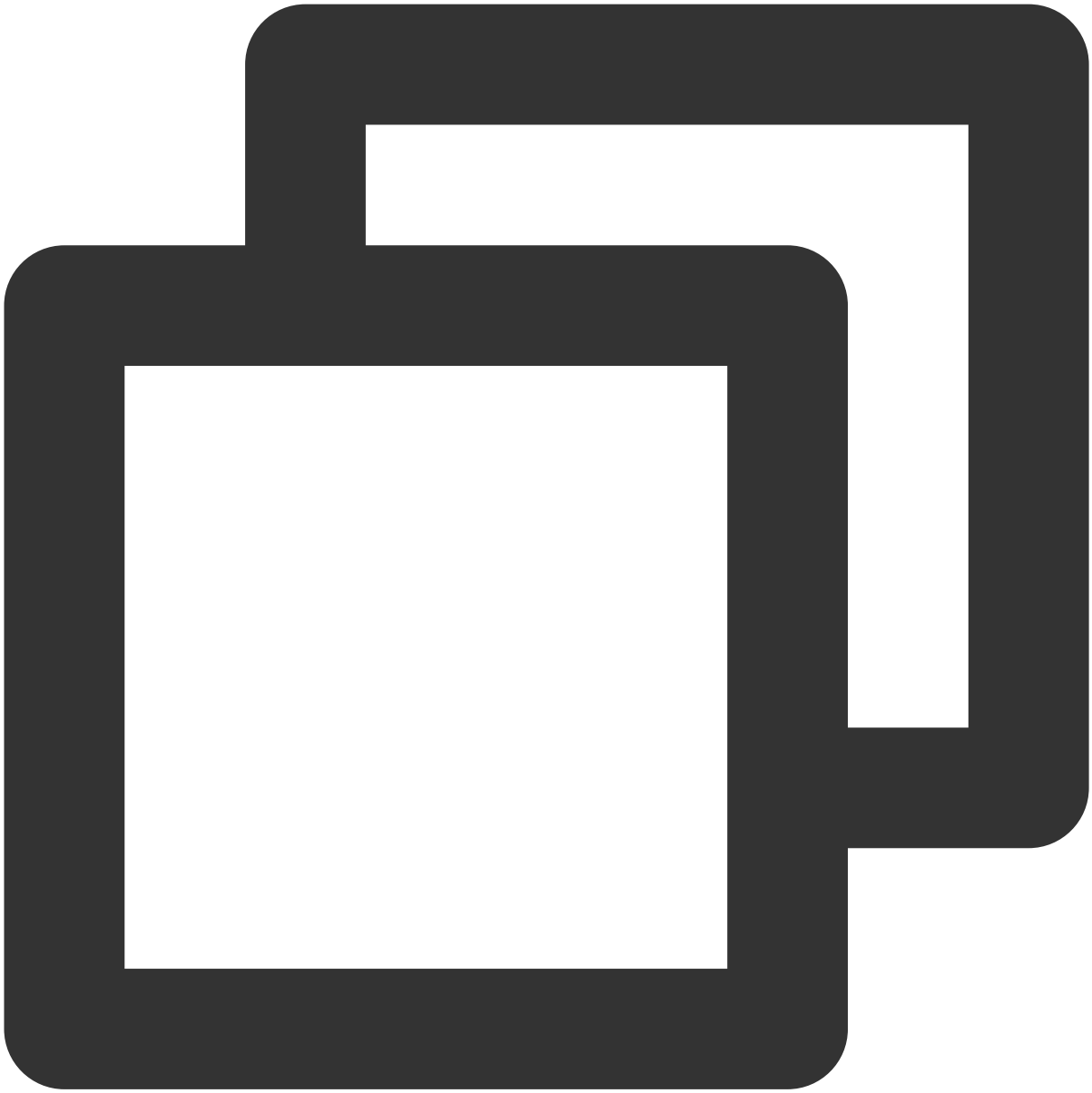
```
    params.role = TRTCCloudDef.TRTCRoleAnchor;

    // Enable local audio and video capture
    startLocalMedia();
    // In an interactive live streaming scenario, enter the room and push streams
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}

// Enable local video preview and audio capture
public void startLocalMedia() {
    // Obtain the video rendering control for displaying the mic-connection audienc
    TXCloudVideoView mTxcvvAudiencePreviewView = findViewById(R.id.live_cloud_view_

    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
    encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT
    mTRTCCloud.setVideoEncoderParam(encParam);

    // boolean mIsFrontCamera can specify using the front/rear camera for video cap
    mTRTCCloud.startLocalPreview(mIsFrontCamera, mTxcvvAudiencePreviewView);
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
}

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        Log.d(TAG, "Enter room succeed");

    } else {
        // The result represents the error code fwhen you fail to enter the room
        Log.d(TAG, "Enter room failed");
    }
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

2. The mic-connection audience start subscribing to the anchor's audio and video streams after they successfully enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }

    @Override
    public void onFirstVideoFrame(String userId, int streamType, int width, int height)
        // The SDK starts rendering the first frame of the local or remote user's video
        if (!userId.isEmpty()) {
            // Stop playing the CDN stream upon receiving the first frame of the anchor
            mLivePlayer.stopPlay();
        }
    }
```

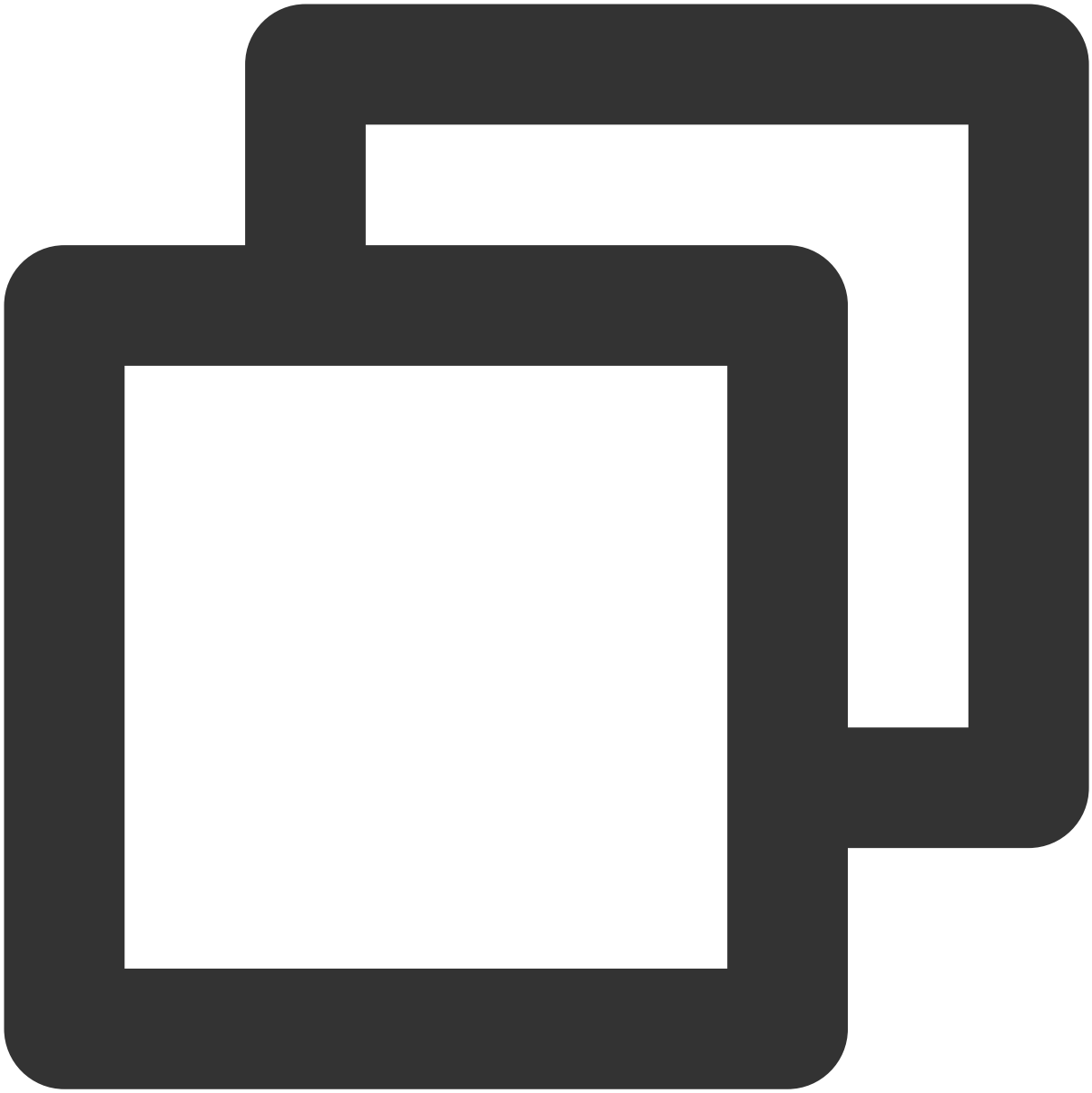
Note:

TRTC stream pulling `startRemoteView` can directly reuse the video rendering control previously used by the CDN stream pulling `setRenderView` .

To avoid video interruptions when switching between stream pullers, it is recommended to wait until the TRTC first frame callback `onFirstVideoFrame` is received before stopping the CDN stream pulling.

3. The anchor updates the publication of mixed media streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the mic-connection audience's room entry
@Override
public void onRemoteUserEnterRoom(String userId) {
    if (!mixUserList.contains(userId)) {
        mixUserList.add(userId);
    }
    updatePublishMediaToCDN(streamName, mixUserList, taskId);
}

// Event callback for updating the media stream
@Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
public void onUpdatePublishMediaStream(String taskId, int code, String message, Bun
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure
}

// Update the publication of mixed media streams to the live streaming CDN
public void updatePublishMediaToCDN(String streamName, List<String> mixUserList, St
    // Set the expiration time for the push URLs
    long txTime = (System.currentTimeMillis() / 1000) + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    String secretParam = UrlHelper.getSafeUrl(LIVE_URL_KEY, streamName, txTime);

    // The target URLs for media stream publication
    TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
    // The target URLs are set for relaying the mixed streams to CDN
    target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToCdn;
    TRTCCloudDef.TRTCPublishCdnUrl cdnUrl = new TRTCCloudDef.TRTCPublishCdnUrl();
    // Construct push URLs (in RTMP format) to the live streaming service provider
    cdnUrl.rtmpUrl = "rtmp://" + PUSH_DOMAIN + "/live/" + streamName + "?" + secret
    // True means the cloud platform CSS, and false means third-party live streamin
    cdnUrl.isInternalLine = true;
    // Multiple CDN push URLs can be added
    target.cdnUrlList.add(cdnUrl);

    // Set media stream encoding output parameters
    TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new TRTCCloudDef.T
    trtcStreamEncoderParam.audioEncodedChannelNum = 1;
    trtcStreamEncoderParam.audioEncodedKbps = 50;
    trtcStreamEncoderParam.audioEncodedCodecType = 0;
    trtcStreamEncoderParam.audioEncodedSampleRate = 48000;
    trtcStreamEncoderParam.videoEncodedFPS = 15;
    trtcStreamEncoderParam.videoEncodedGOP = 2;
    trtcStreamEncoderParam.videoEncodedKbps = 1300;
    trtcStreamEncoderParam.videoEncodedWidth = 540;
    trtcStreamEncoderParam.videoEncodedHeight = 960;

    // Configuration parameters for media stream transcoding
    TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new TRTCCloudDef.T
    if (mixUserList != null) {
        ArrayList<TRTCCloudDef.TRTCUser> audioMixUserList = new ArrayList<>();
        ArrayList<TRTCCloudDef.TRTCVideoLayout> videoLayoutList = new ArrayList<>()

        for (int i = 0; i < mixUserList.size() && i < 16; i++) {
            TRTCCloudDef.TRTCUser user = new TRTCCloudDef.TRTCUser();
            // The integer room number is intRoomId
            user.strRoomId = mRoomId;
            user.userId = mixUserList.get(i);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            audioMixUserList.add(user);

            TRTCCloudDef.TRTCVideoLayout videoLayout = new TRTCCloudDef.TRTCVideoLa
            if (mixUserList.get(i).equals(mUserId)) {
                // The layout for the anchor's video
                videoLayout.x = 0;
                videoLayout.y = 0;
                videoLayout.width = 540;
                videoLayout.height = 960;
                videoLayout.zOrder = 0;
            } else {
                // The layout for the mic-connection audience's video
                videoLayout.x = 400;
                videoLayout.y = 5 + i * 245;
                videoLayout.width = 135;
                videoLayout.height = 240;
                videoLayout.zOrder = 1;
            }
            videoLayout.fixedVideoUser = user;
            videoLayout.fixedVideoStreamType = TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_
            videoLayoutList.add(videoLayout);
        }

        // Specify the information for each input audio stream in the transcoding s
        trtcStreamMixingConfig.audioMixUserList = audioMixUserList;
        // Specify the information of position, size, layer, and stream type for ea
        trtcStreamMixingConfig.videoLayoutList = videoLayoutList;
    }

    // Update the published media stream
    mTRTCCloud.updatePublishMediaStream(taskId, target, trtcStreamEncoderParam,  tr
}
```

Note:

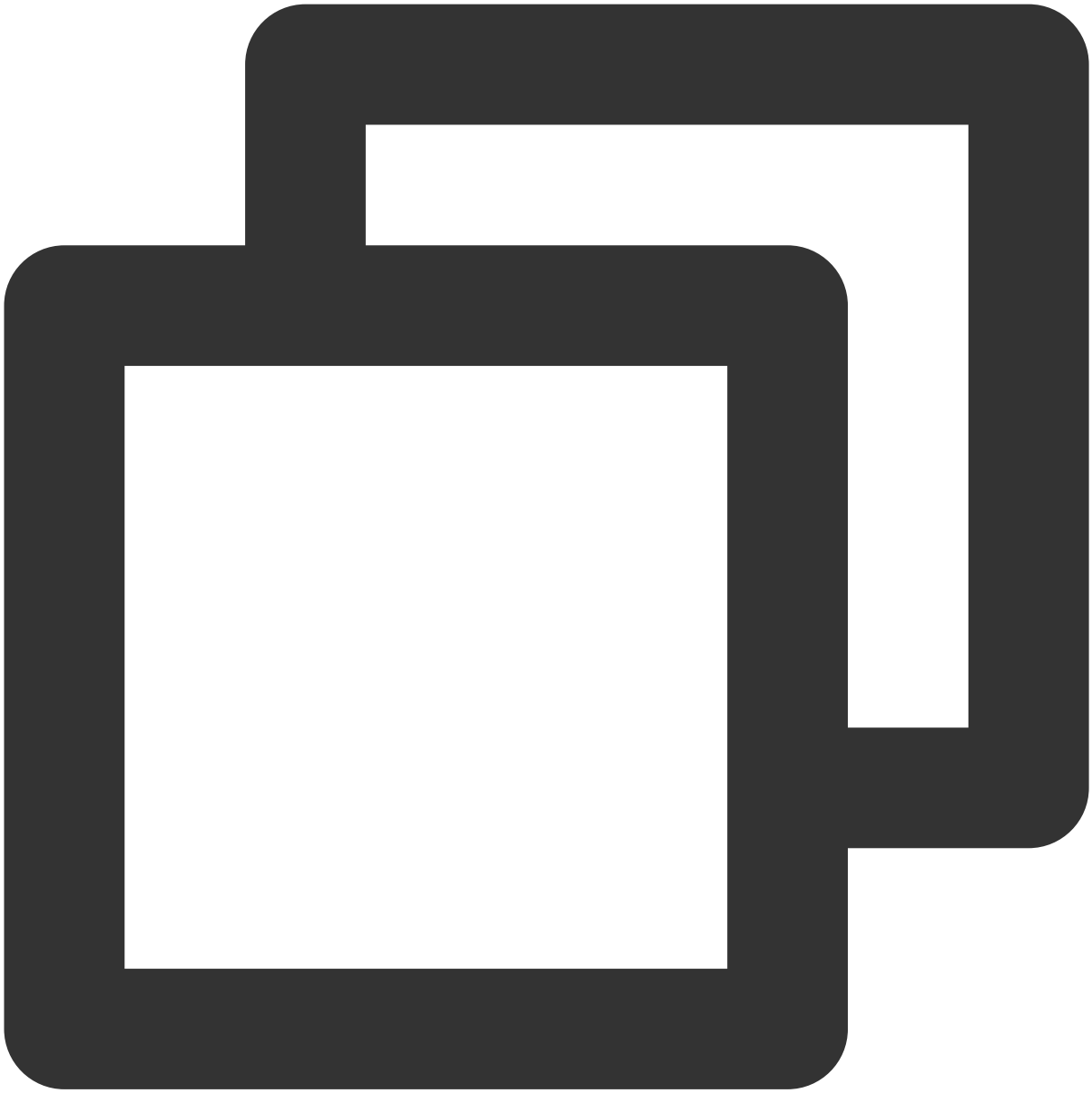
To ensure continuous CDN playback without stream disconnection, you need to keep the media stream encoding output parameter `trtcStreamEncoderParam` and the stream name `streamName` unchanged.

Media stream encoding output parameters and mixed display layout parameters can be customized according to business needs. Currently, up to 16 channels of audio and video input are supported. If a user only provides audio, it will still be counted as one channel.

Switching between audio only, audio and video, and video only is not supported within the same task.

4. The off-streaming audience exit the room, and the anchor updates the mixed stream task.

©2013-2022 Tencent Cloud. All rights reserved.



```
// The reusable TRTC video rendering control
mLivePlayer.setRenderView(TXCloudVideoView view);
// Restart playing CDN media stream
mLivePlayer.startLivePlay(URL);

// Callback for player event listener
mLivePlayer.setObserver(new V2TXLivePlayerObserver() {
    @Override
    public void onVideoLoading(V2TXLivePlayer player, Bundle extraInfo) {
        // Video loading event
    }
```

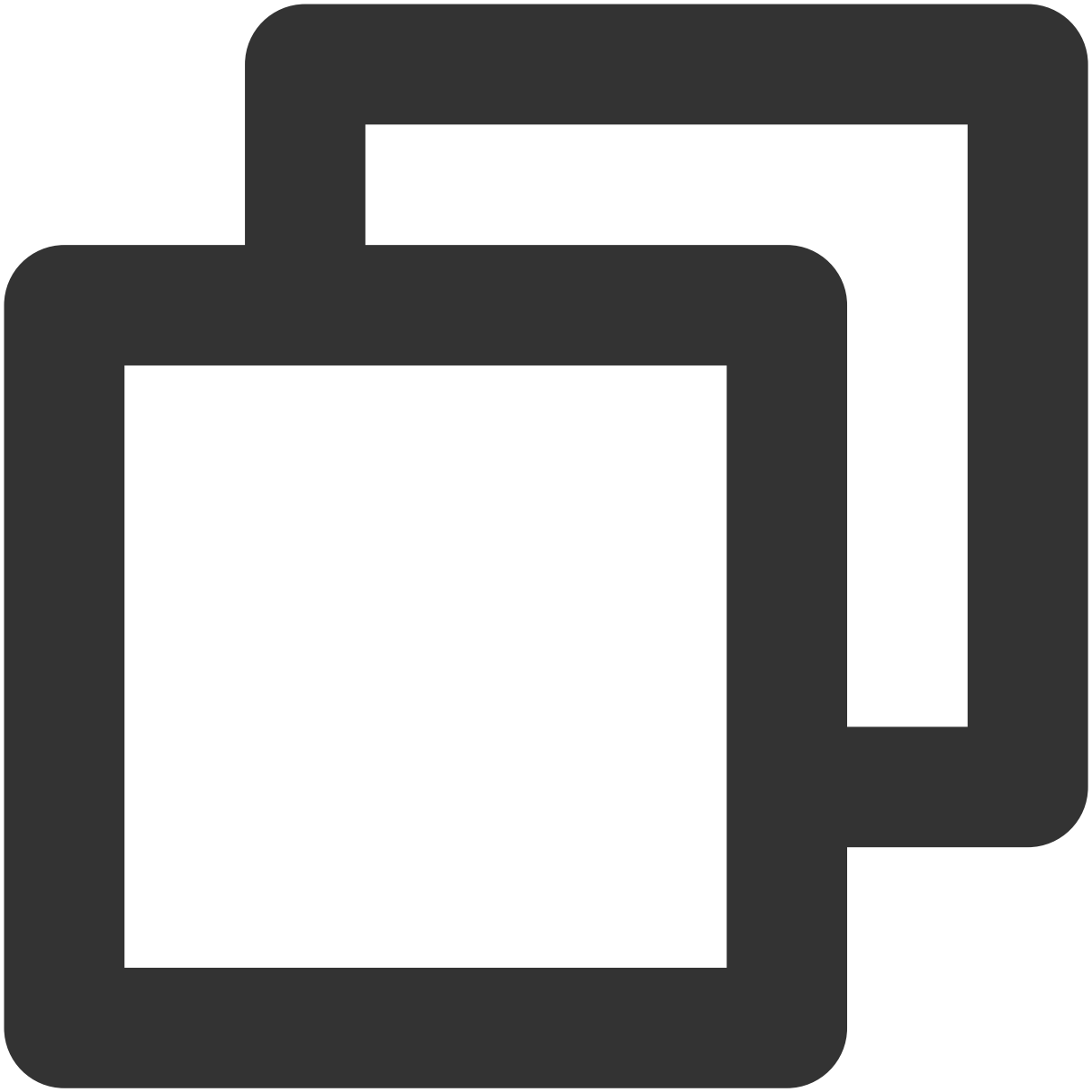
©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onVideoPlaying(V2TXLivePlayer player, boolean firstPlay, Bundle ext
        // Video playback event
        if (firstPlay) {
            mTRTCCloud.stopAllRemoteView();
            mTRTCCloud.stopLocalAudio();
            mTRTCCloud.stopLocalPreview();
            mTRTCCloud.exitRoom();
        }
    }
});
```

Note:

To avoid video interruptions when switching the stream puller, it is recommended to wait for the player's video playback event `onVideoPlaying` before exiting the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.

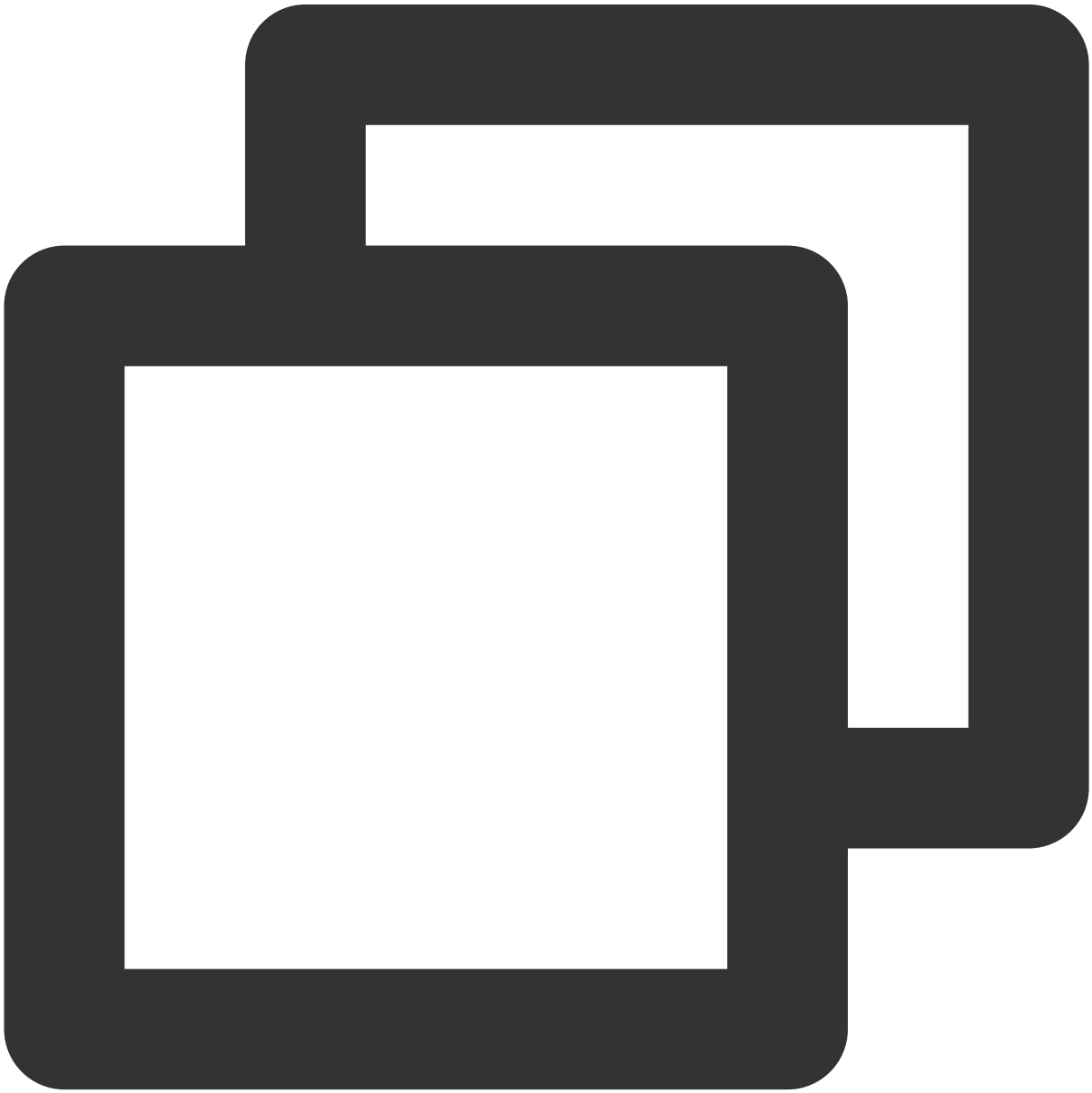


```
// Event callback for the mic-connection audience's room exit
@Override
public void onRemoteUserLeaveRoom(String userId, int reason) {
    if (mixUserList.contains(userId)) {
        mixUserList.remove(userId);
    }
    // The anchor updates the mixed stream task
    updatePublishMediaToCDN(streamName, mixUserList, taskId);
}

// Event callback for updating the media stream
```

©2013-2022 Tencent Cloud. All rights reserved.

```
@Override
public void onUpdatePublishMediaStream(String taskId, int code, String message, Bun
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure
}
```

Step 4: The anchor stops the live streaming and exits the room



```
public void exitRoom() {
    // Stop all published media streams
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    mTRTCCloud.stopPublishMediaStream("");
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.stopLocalPreview();
    mTRTCCloud.exitRoom();
}

// Event callback for stopping media streams
@Override
public void onStopPublishMediaStream(String taskId, int code, String message, Bundl
    // When you call stopPublishMediaStream, the taskId you provide will be returne
    // code: Callback result. 0 means success and other values mean failure
}

// Event callback for exiting the room
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Actively call exitRoom to exit the room");
    } else if (reason == 1) {
        Log.d(TAG, "Removed from the current room by the server");
    } else if (reason == 2) {
        Log.d(TAG, "The current room has been dissolved");
    }
}
```

Note:

To stop publishing media streams, enter an empty string for `taskId` . This will stop all the media streams you have published.

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

Advanced Features

Product Information Pop-up

The Product Information Pop-up feature can be implemented through IM Custom Message or SEI Information. Below are the specific information of the two implementation methods.

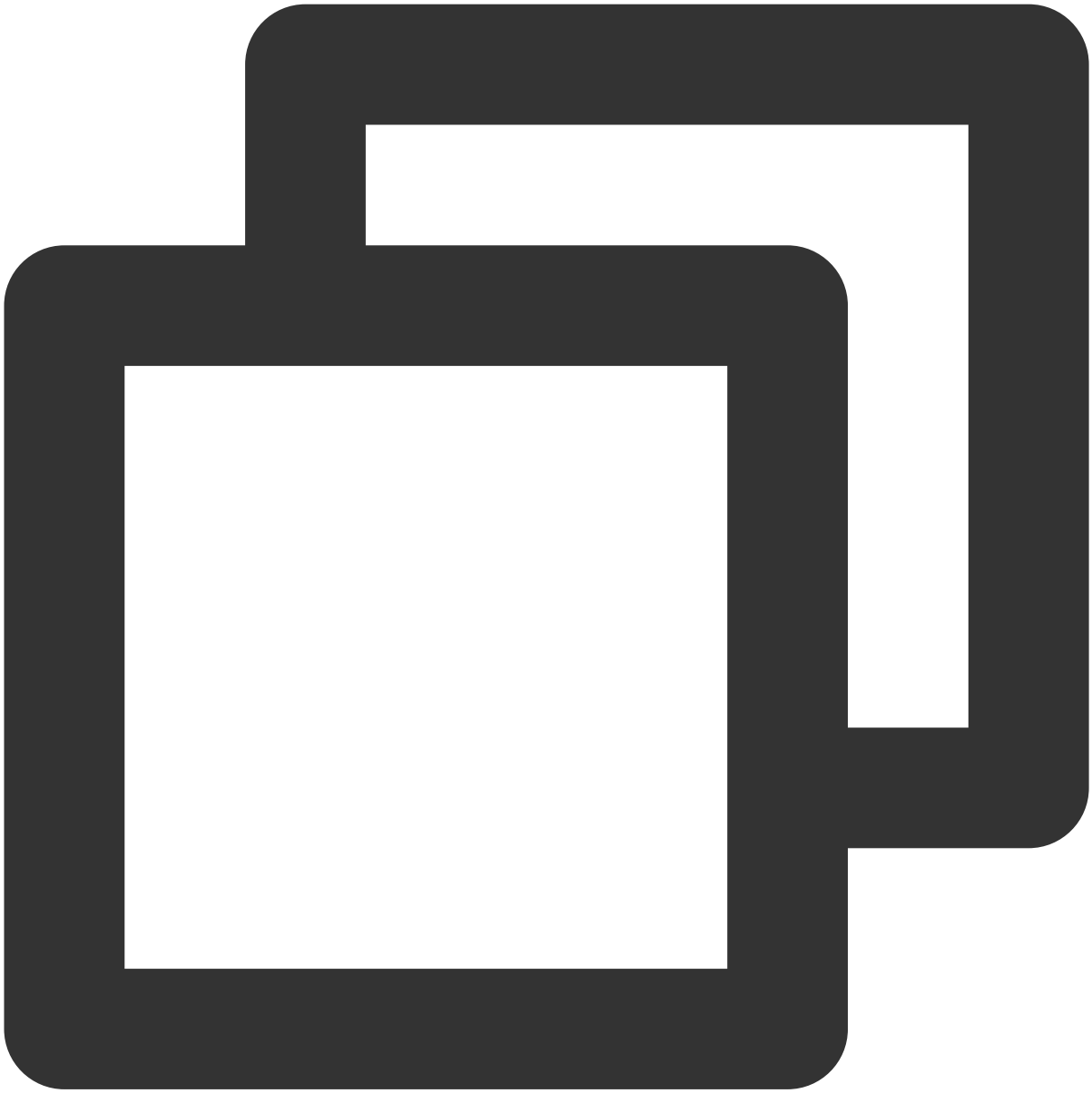
Custom Message

Custom messages depend on Instant Messaging (IM). You need to activate the service and import the IM SDK in advance. For detailed guidelines, see Voice Chat Room Connection Guide - Connection Preparation.

1. Send Custom Messages

Method 1: The anchor sends product pop-up related custom group messages on the client.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Construct product pop-up message body
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("cmd", "item_popup_msg");
    JSONObject msgJsonObject = new JSONObject();
    msgJsonObject.put("itemNumber", 1); // Item number
    msgJsonObject.put("itemPrice", 199.0);  // Item price
    msgJsonObject.put("itemTitle", "xxx");  // Item title
    msgJsonObject.put("itemUrl", "xxx");// Item URL
    jsonObject.put("msg", msgJsonObject);
} catch (JSONException e) {
```

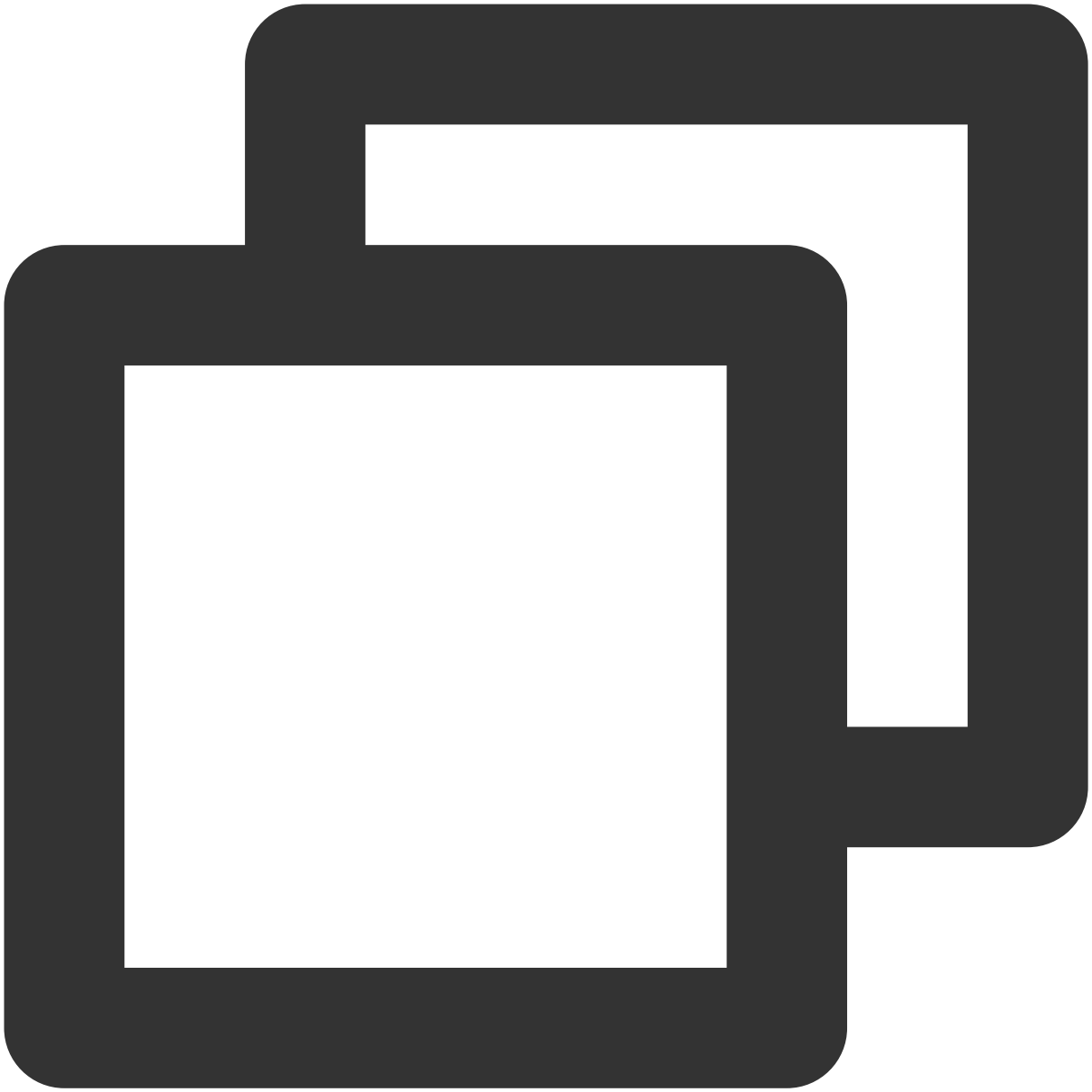
©2013-2022 Tencent Cloud. All rights reserved.

```
    e.printStackTrace();
}
String data = jsonObject.toString();

// Send custom group messages (it is recommended that product pop-up messages shoul
V2TIMManager.getInstance().sendGroupCustomMessage(data.getBytes(), mRoomId,
        V2TIMMessage.V2TIM_PRIORITY_HIGH, new V2TIMValueCallback<V2TIMMessage>() {
            @Override
            public void onError(int i, String s) {
                // Failed to send product pop-up message
            }

            @Override
            public void onSuccess(V2TIMMessage v2TIMMessage) {
                // Successfully sent product pop-up message
                // Locally rendering of product pop-up effect
            }
        });
```

Method 2: The backend operators sends product pop-up related custom group messages on the server.

Request URL sample:

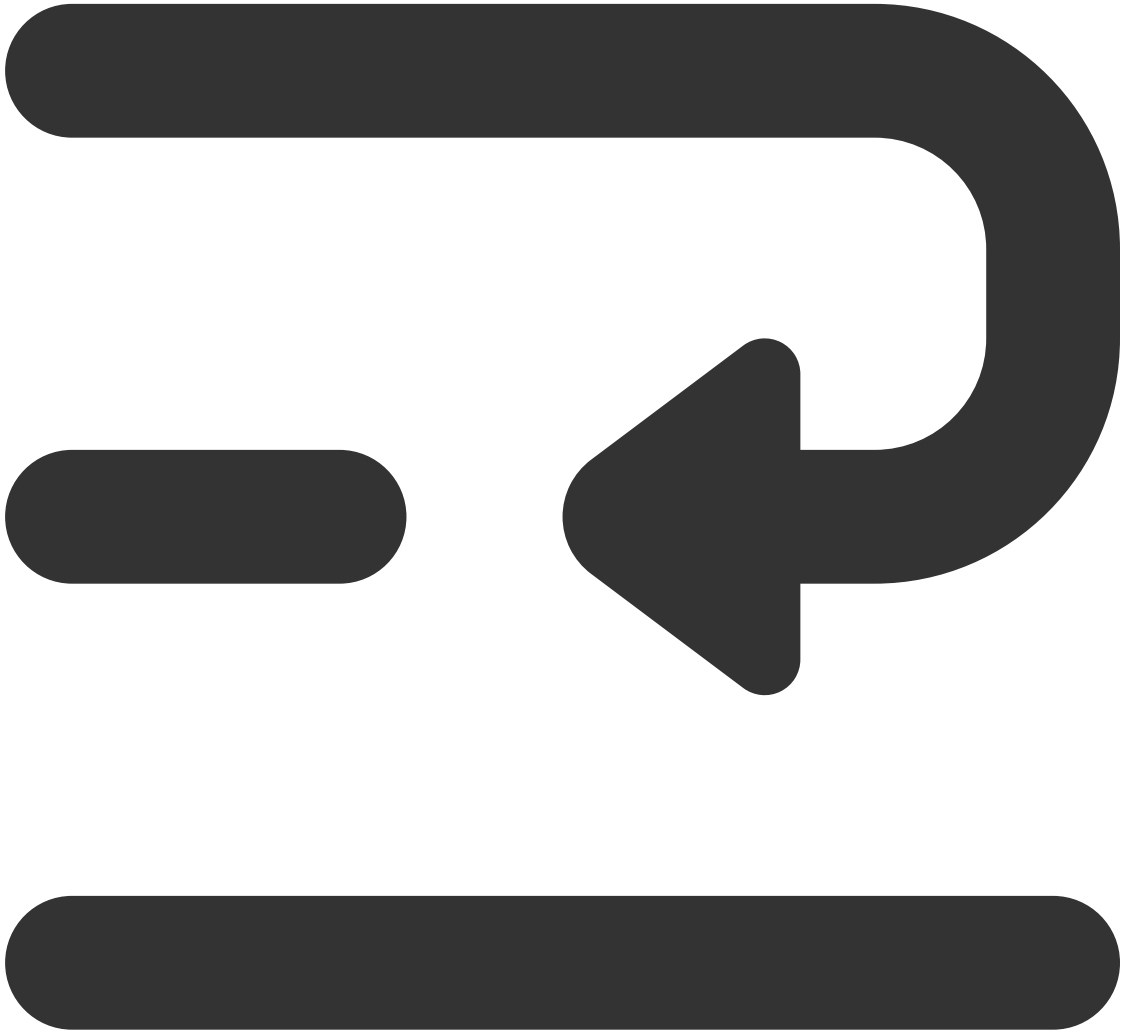
©2013-2022 Tencent Cloud. All rights reserved.



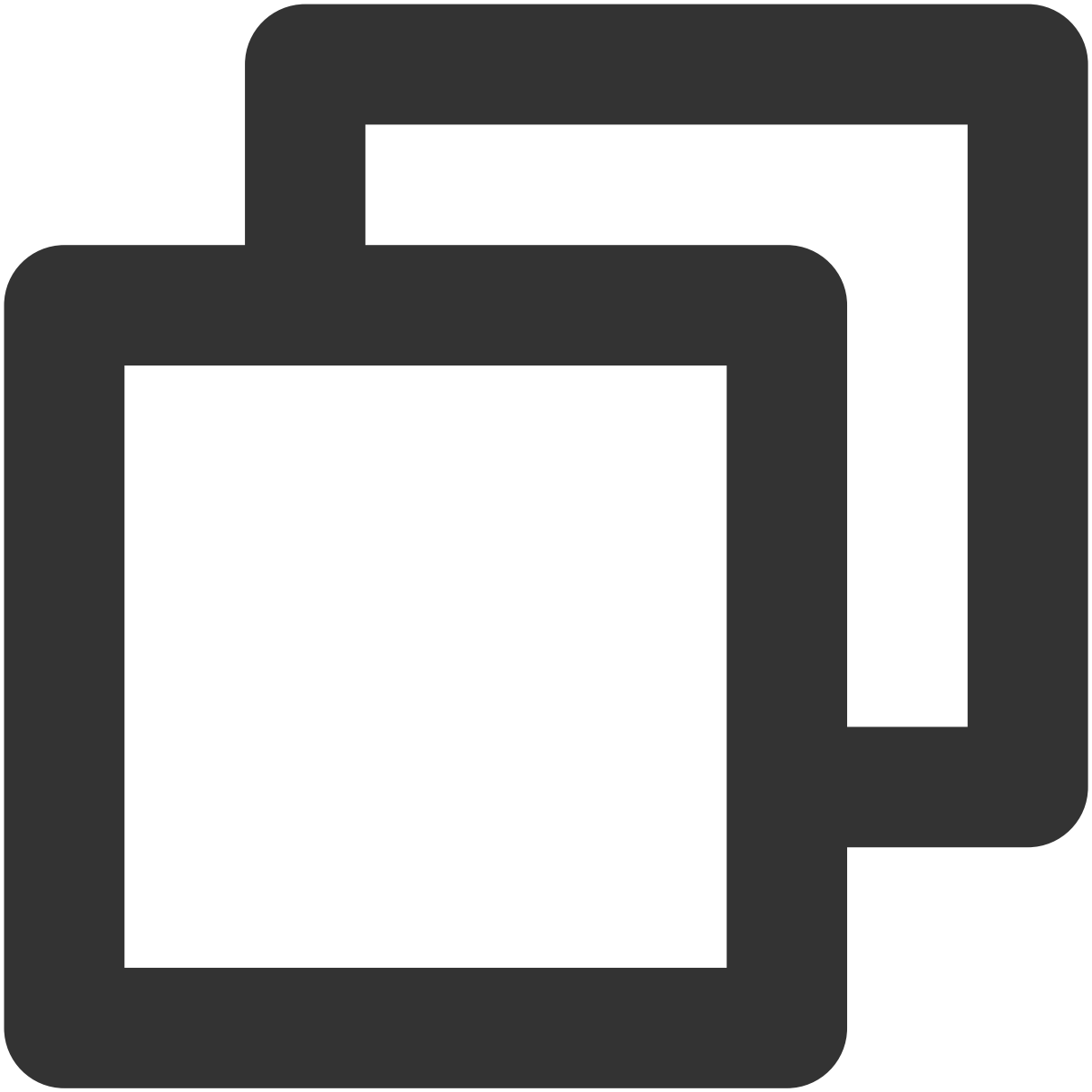
```
https://xxxxxx/v4/group_open_http_svc/send_group_msg?sdkappid=88888888&identifier=a
```

Request packet body sample:

©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.



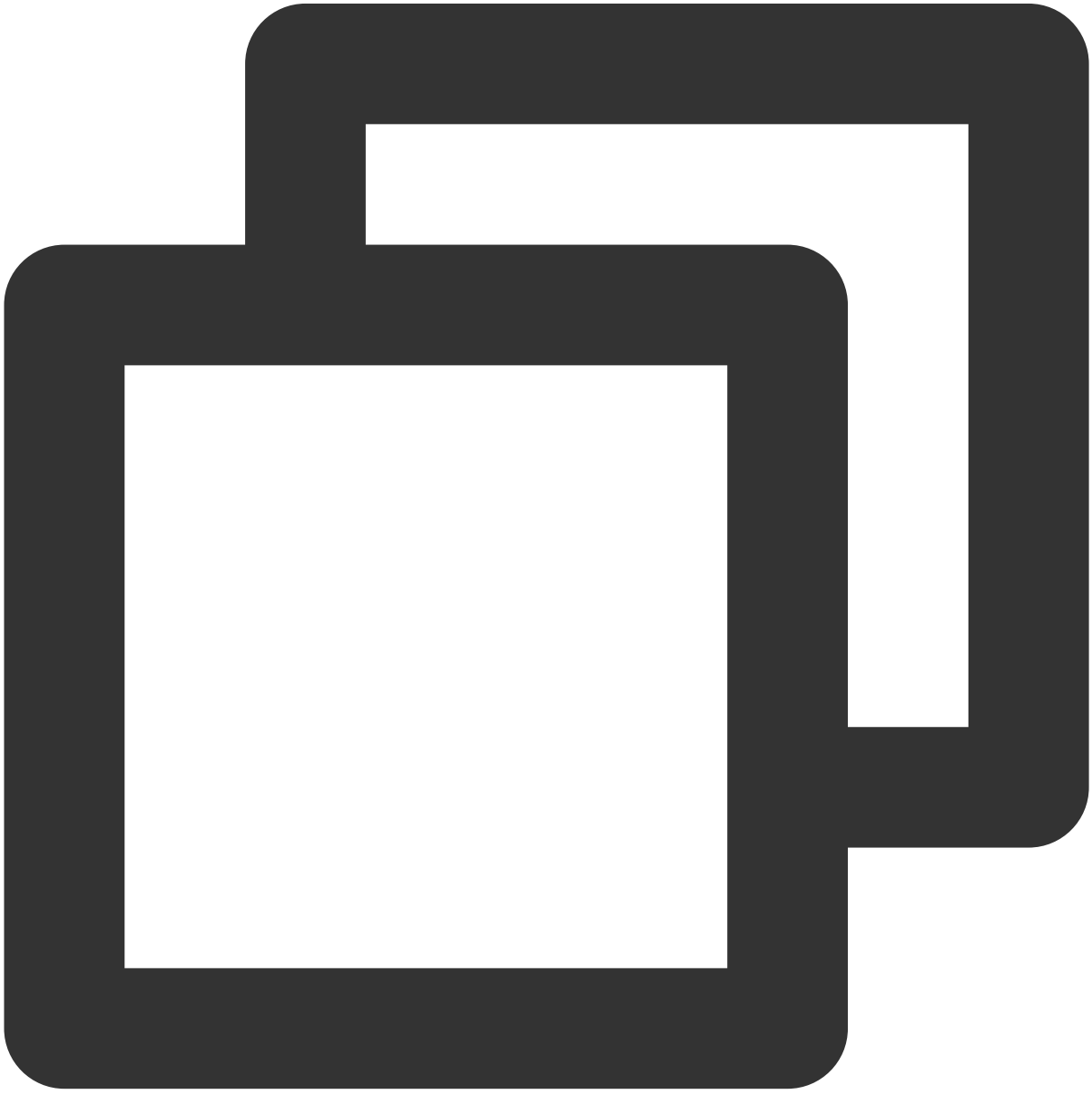
```
{
    "GroupId": "@TGS#12DEVUDHQ",
    "Random": 2784275388,
    "MsgPriority": "High",  // The priority of the message. It is recommended to se
    "MsgBody": [
        {
            "MsgType": "TIMCustomElem",
            "MsgContent": {
                // itemNumber: item number; itemPrice: item price; itemTitle: item
                "Data": "{\\"cmd\\": \\"item_popup_msg\\", \\"msg\\": {\\"itemNumbe
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        ]
    }
```

2. Receive Custom Messages

Other users in the room receive callback for custom group messages, then proceed with message parsing and product pop-up effect rendering.



```
// Custom group messages received
V2TIMManager.getInstance().addSimpleMsgListener(new V2TIMSimpleMsgListener() {
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    public void onRecvGroupCustomMessage(String msgID, String groupID, V2TIMGroupMe
        String customStr = new String(customData);
        if (!customStr.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(customStr);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("item_popup_msg")) {
                    int itemNumber = messageJsonObject.getInt("itemNumber");  // It
                    double itemPrice = messageJsonObject.getDouble("itemPrice");  /
                    String itemTitle = messageJsonObject.getString("itemTitle");  /
                    String itemUrl = messageJsonObject.getString("itemUrl");  // It
                    // Render product pop-up effect based on item number, item pric
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

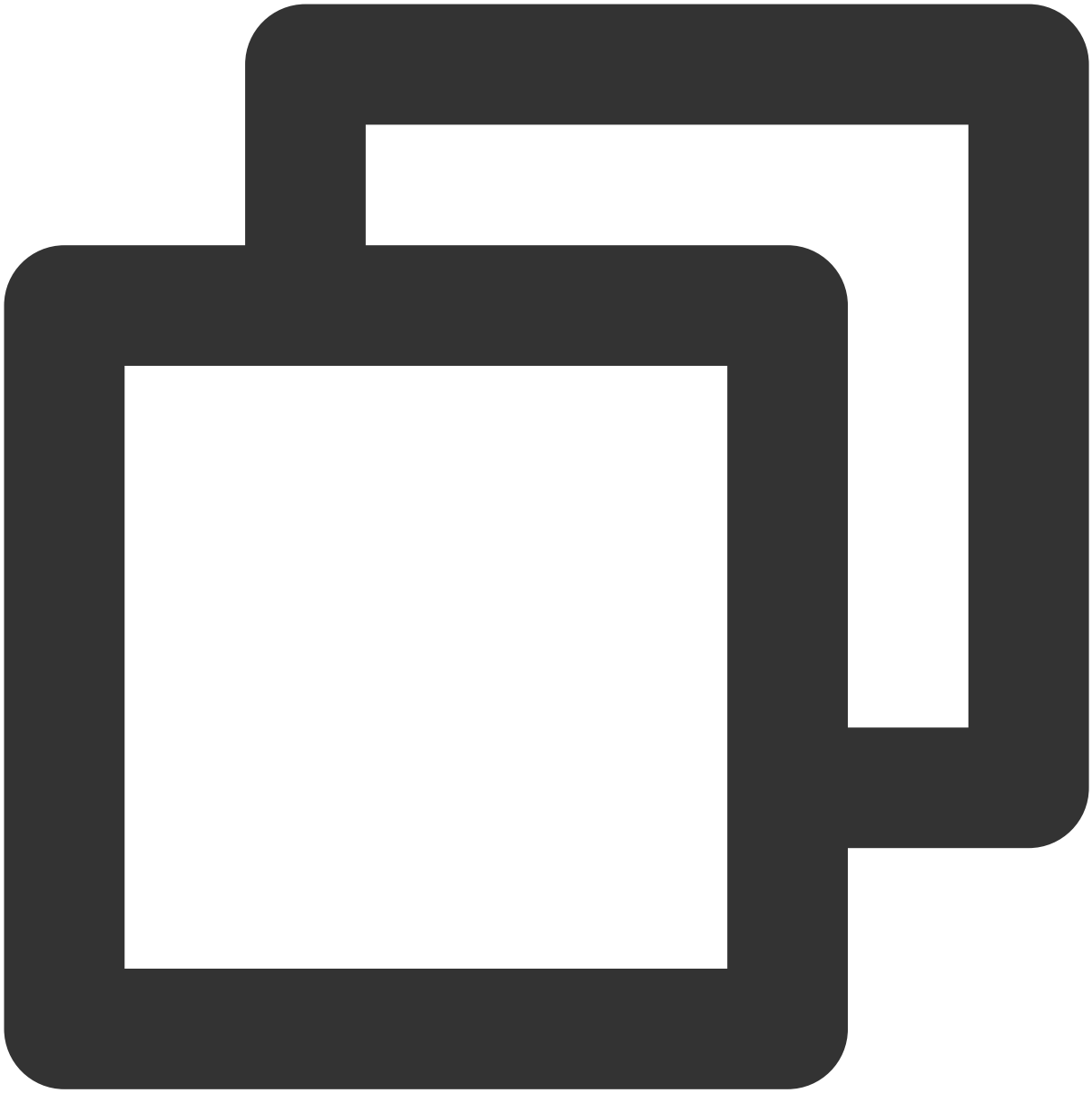
SEI Information

SEI information will be inserted into the anchor's video stream for transmission, achieving precise sync between the product information pop-up and the anchor's live streaming.

1. Send SEI Information

The anchor sends SEI messages related to product pop-up on the TRTC client.

©2013-2022 Tencent Cloud. All rights reserved.



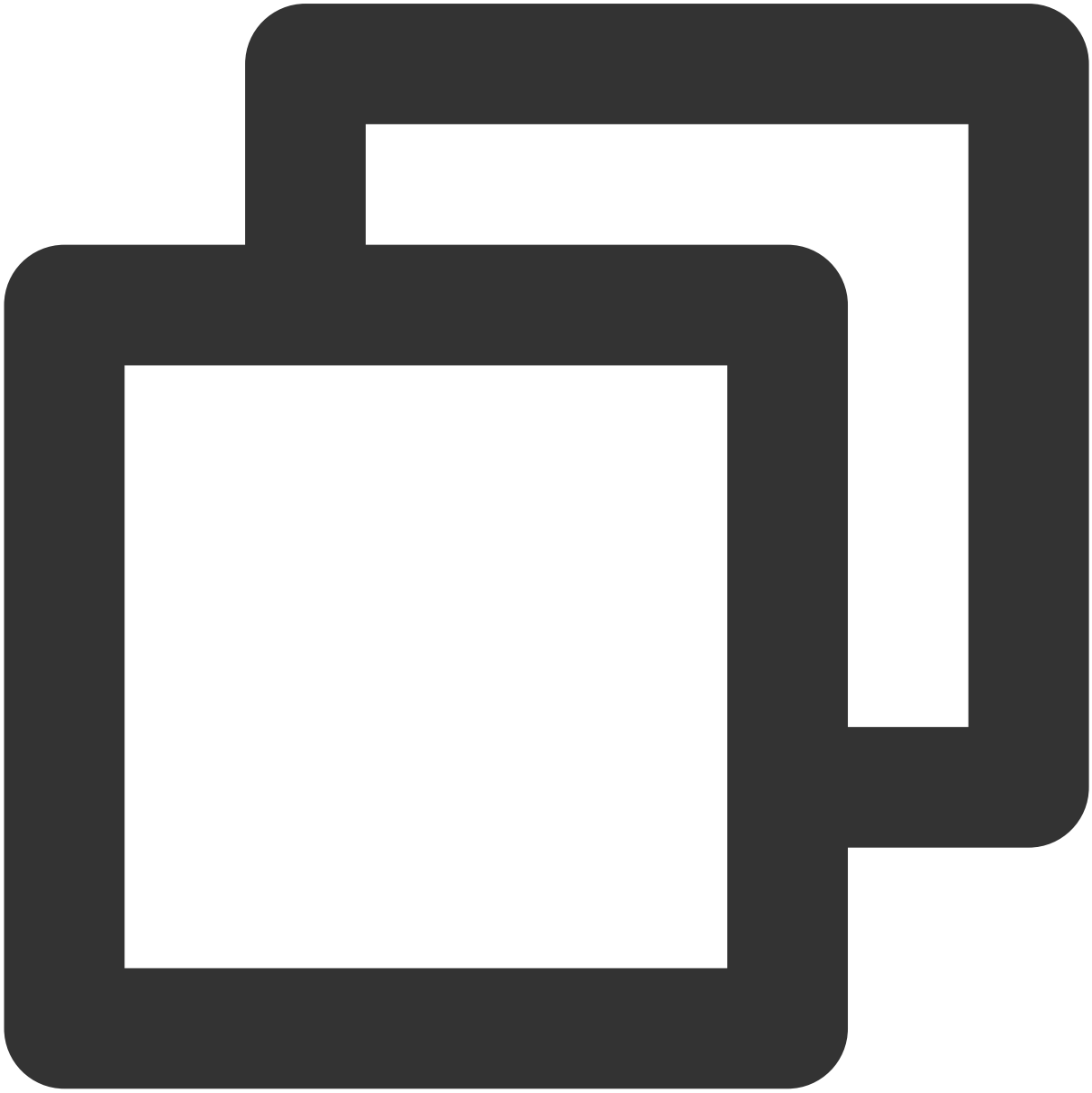
```
// Construct product pop-up message body
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("cmd", "item_popup_msg");
    JSONObject msgJsonObject = new JSONObject();
    msgJsonObject.put("itemNumber", 1); // Item number
    msgJsonObject.put("itemPrice", 199.0);  // Item price
    msgJsonObject.put("itemTitle", "xxx");  // Item title
    msgJsonObject.put("itemUrl", "xxx");// Item URL
    jsonObject.put("msg", msgJsonObject);
} catch (JSONException e) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    e.printStackTrace();
}
String data = jsonObject.toString();

// Send SEI information
mTRTCCloud.sendSEIMsg(data.getBytes(), 1);
```

2. Receive SEI Information

Method 1: The audience receives SEI messages on the TRTC client, then proceeds with message parsing and product pop-up effect rendering.

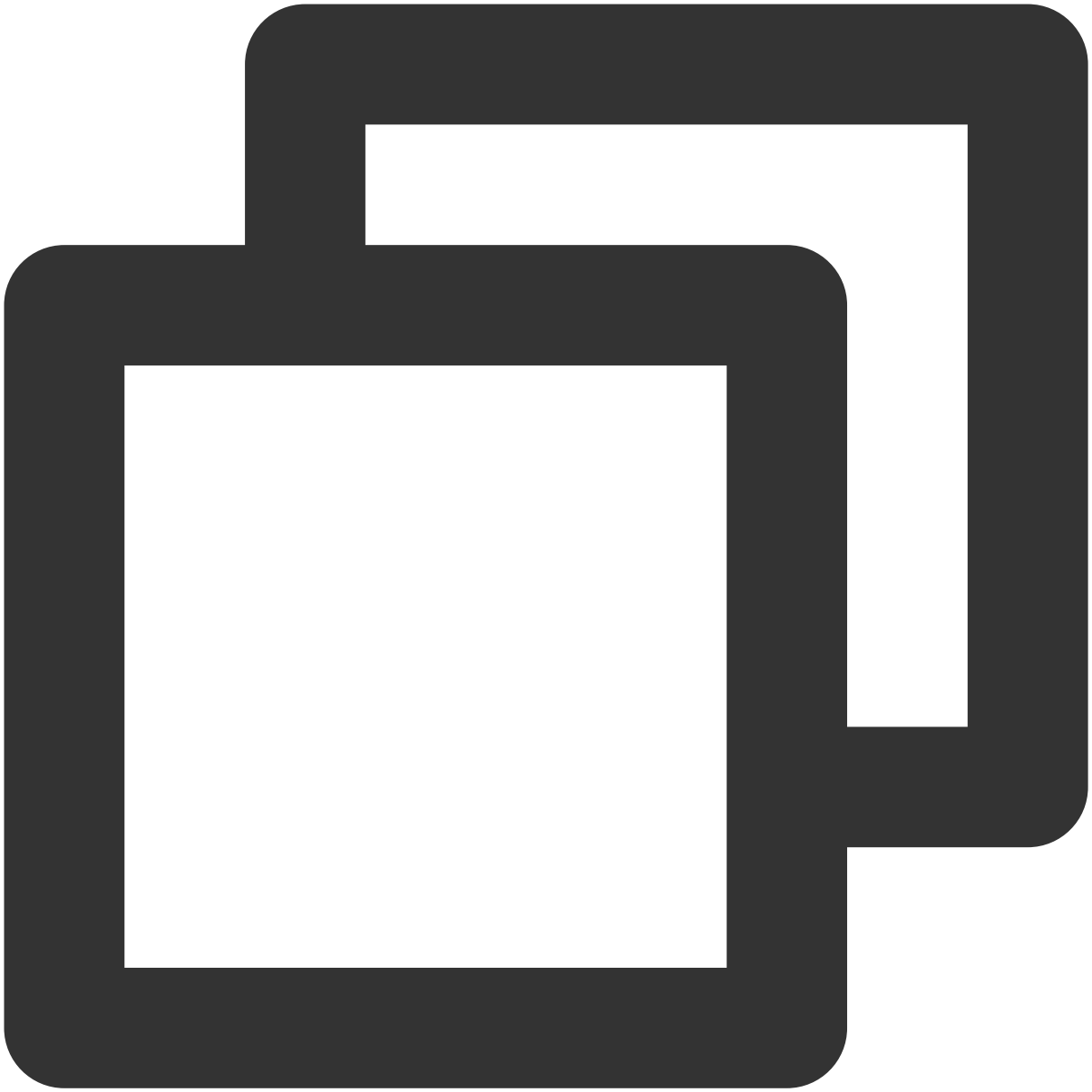


©2013-2022 Tencent Cloud. All rights reserved.

```
mTRTCCloud.setListener(new TRTCCloudListener() {
    @Override
    public void onRecvSEIMsg(String userId, byte[] data) {
        String dataStr = new String(data);
        if (!dataStr.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(dataStr);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("item_popup_msg")) {
                    int itemNumber = messageJsonObject.getInt("itemNumber");  // It
                    double itemPrice = messageJsonObject.getDouble("itemPrice");  /
                    String itemTitle = messageJsonObject.getString("itemTitle");  /
                    String itemUrl = messageJsonObject.getString("itemUrl");  // It
                    // Render product pop-up effect based on item number, item pric
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Method 2: The audience receives SEI messages on the CDN stream player, then proceeds with message parsing and product pop-up effect rendering.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the PayloadType for sending SEI messages in TRTC
mTRTCCloud.callExperimentalAPI("{\\"api\\":\\"setSEIPayloadType\\",\\"params\\":{\\

// Enable receiving SEI messages on the player and set the PayloadType
mLivePlayer.enableReceiveSeiMessage(true, 5);

// SEI message callback and parsing
mLivePlayer.setObserver(new V2TXLivePlayerObserver() {
    @Override
    public void onReceiveSeiMessage(V2TXLivePlayer player, int payloadType, byte[]
        String dataStr = new String(data);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        if (!dataStr.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(dataStr);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("item_popup_msg")) {
                    int itemNumber = messageJsonObject.getInt("itemNumber");  // It
                    double itemPrice = messageJsonObject.getDouble("itemPrice");  /
                    String itemTitle = messageJsonObject.getString("itemTitle");  /
                    String itemUrl = messageJsonObject.getString("itemUrl");  // It
                    // Render product pop-up effect based on item number, item pric
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Note:

It is necessary to ensure that the SEI `PayloadType` of the TRTC sender and the player receiver are consistent, so that the audience can successfully receive the SEI messages relayed via TRTC.

Product Explanation Replay

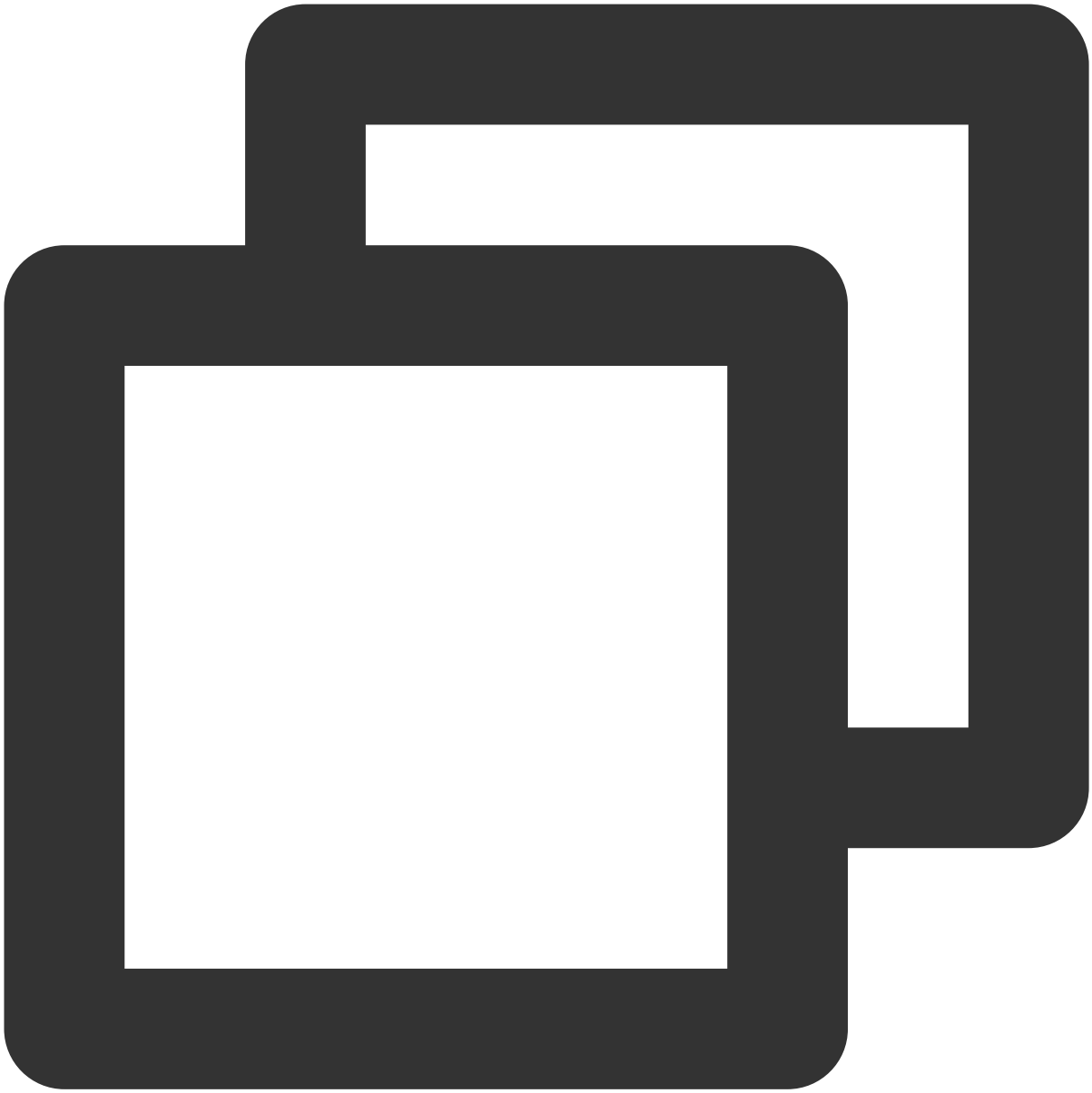
By playing pre-recorded product explanation videos, the product explanation replay feature is implemented.

First, it is necessary to initialize the player, then start playing the recorded video. TXVodPlayer supports two playback modes, which you can choose according to your needs:

Using the URL method

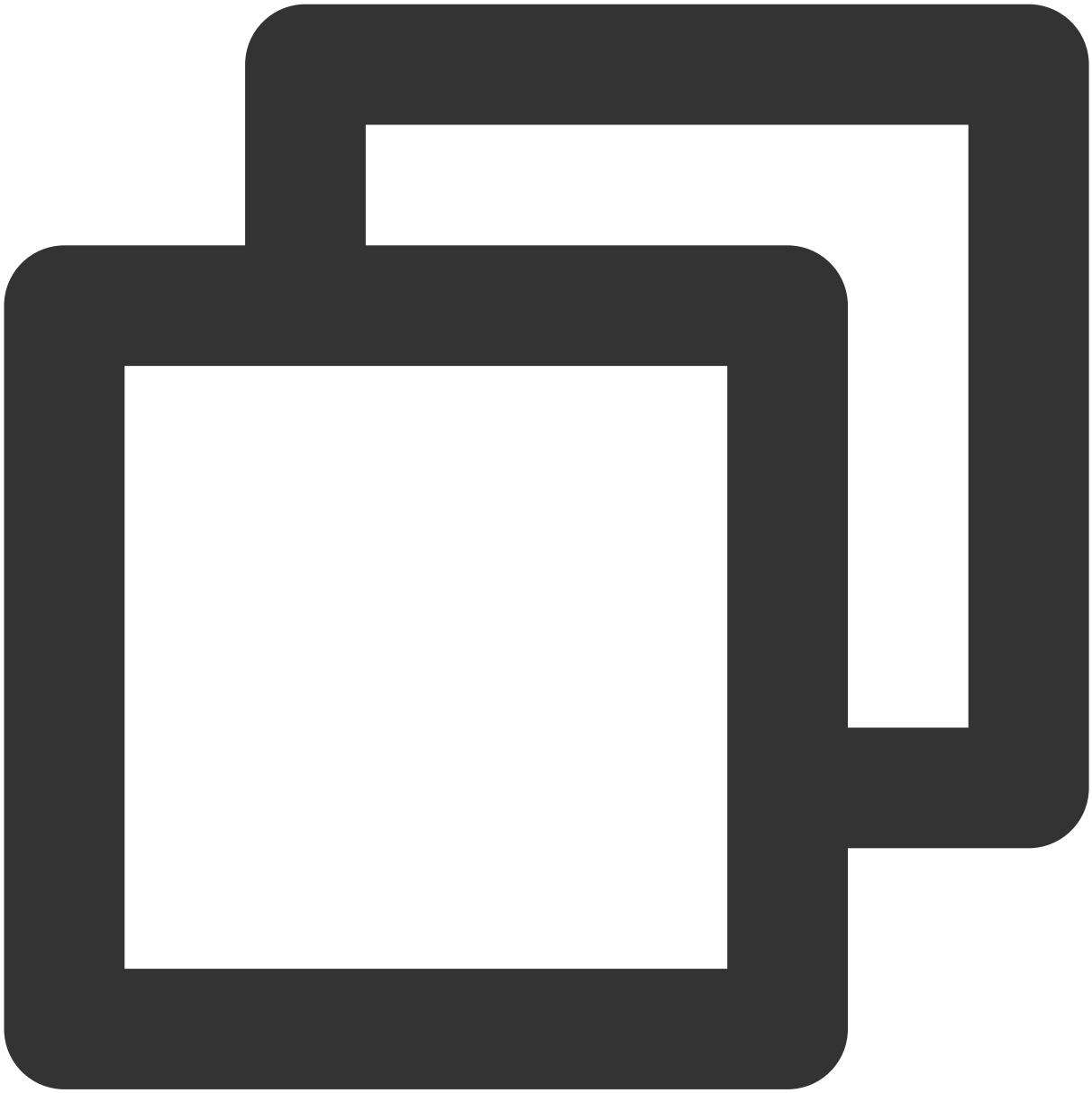
Using the FileId method

©2013-2022 Tencent Cloud. All rights reserved.



```
// Play URL video resource
String url = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
mVodPlayer.startVodPlay(url);

// Play local video resources
String localFile = "/sdcard/video.mp4";
mVodPlayer.startVodPlay(localFile);
```

©2013-2022 Tencent Cloud. All rights reserved.

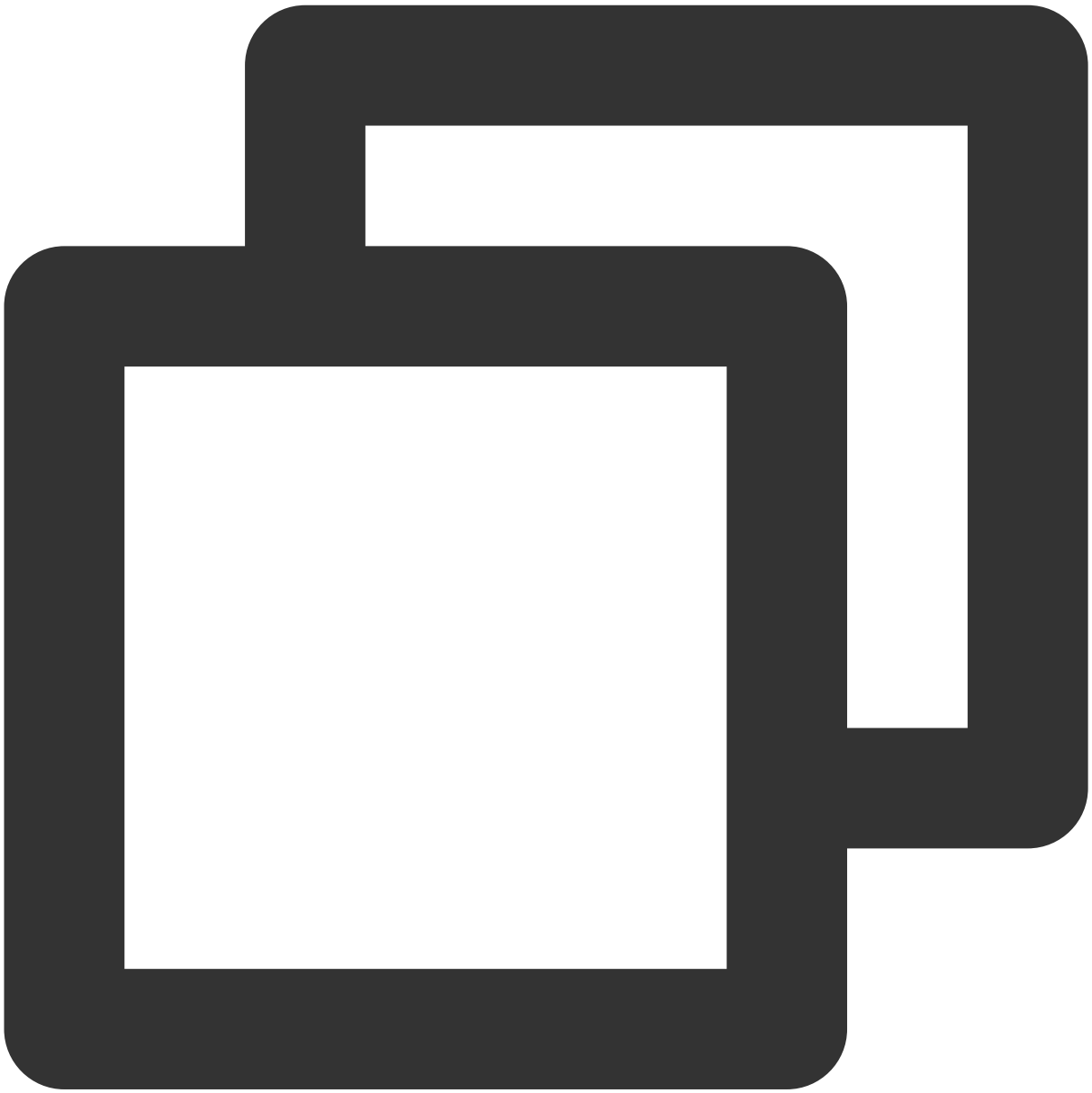


```
// Recommended to use the new API below
// The psign means player signature. For more information about the signature and h
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // The appId of t
    "4564972819220421305", // The fileId of video
    "psignxxxxxxx"); // Player signature
mVodPlayer.startVodPlay(playInfoParam);

// Old API, not recommended
TXPlayerAuthBuilder authBuilder = new TXPlayerAuthBuilder();
authBuilder.setAppId(1252463788);
authBuilder.setFileId("4564972819220421305");
```

©2013-2022 Tencent Cloud. All rights reserved.

```
mVodPlayer.startVodPlay(authBuilder);
```

Playback control: adjust the progress, pause playback, resume playback, and end playback.



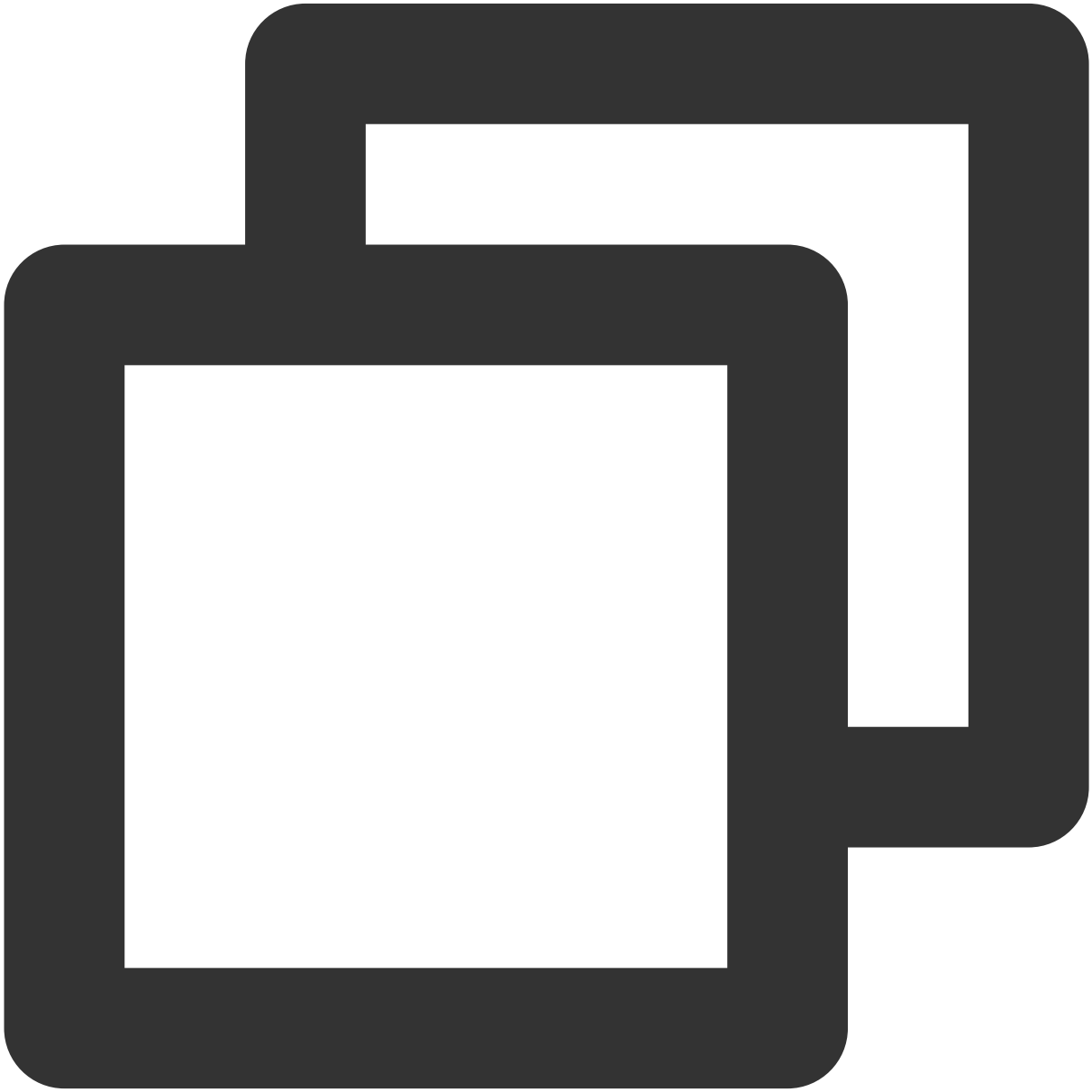
```
// Adjust the progress (seconds)
mVodPlayer.seek(time);

// Pause playback
mVodPlayer.pause();

// Resume playback
mVodPlayer.resume();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// End playback (clear the last frame)
mVodPlayer.stopPlay(true);
```

Note:

When stopping playback, remember to destroy the View control, especially before the next `startVodPlay`.
Otherwise, it will cause a large amount of memory leak and screen flash.
Also, when exiting the playback interface, remember to call the rendering View's `onDestroy()` function.
Otherwise, it may cause memory leaks and a "Receiver not registered" warning.

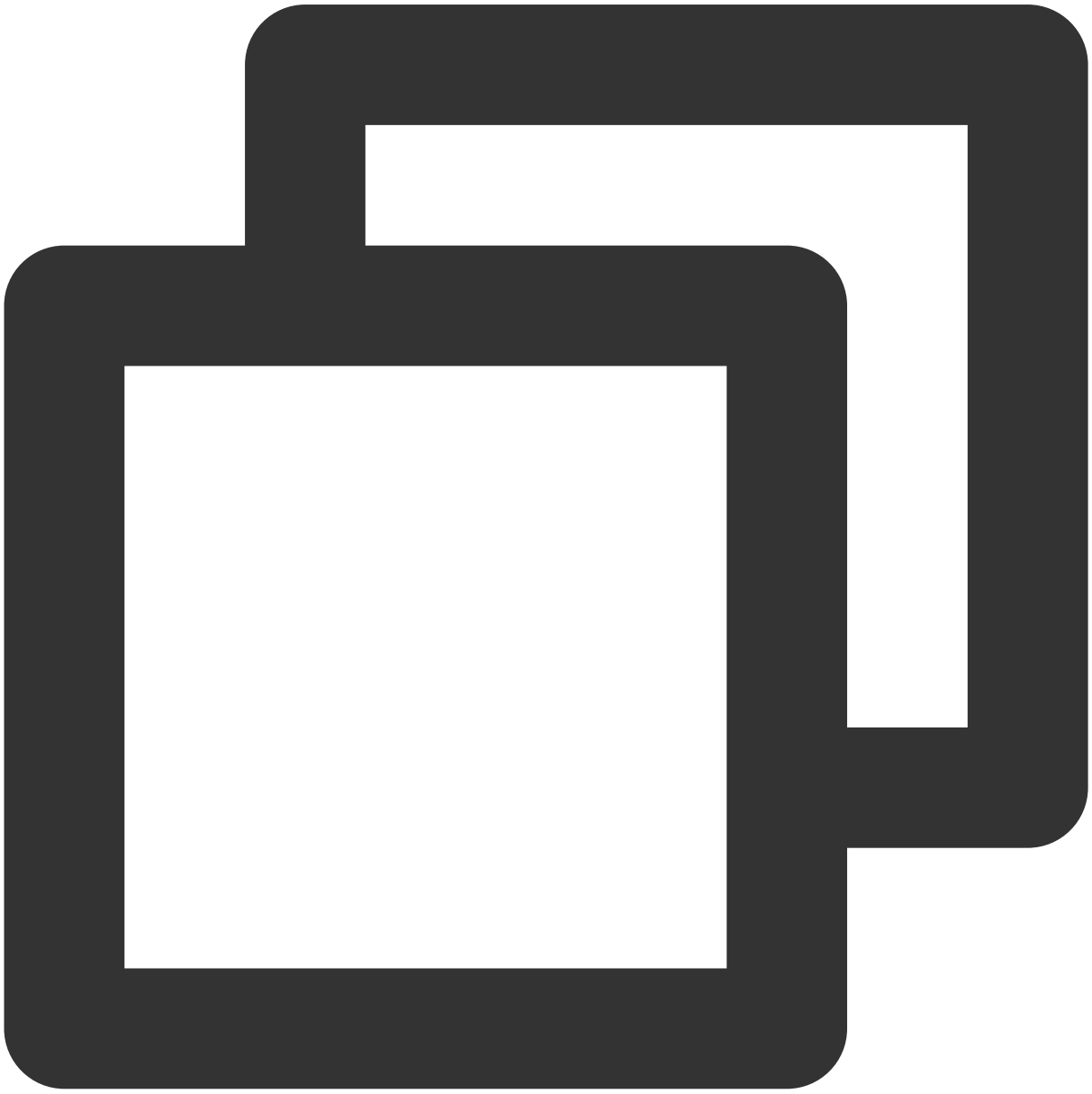


©2013-2022 Tencent Cloud. All rights reserved.

```
@Override
public void onDestroy() {
    super.onDestroy();
    mVodPlayer.stopPlay(true); // True means clearing the last frame
    mPlayerView.onDestroy();
}
```

Cross-room Mic-connection PK

1. Either party initiates the cross-room mic-connection PK.



©2013-2022 Tencent Cloud. All rights reserved.

```
public void connectOtherRoom(String roomId, String userId) {
    try {
        JSONObject jsonObj = new JSONObject();
        // The digit room number is roomId
        jsonObj.put("strRoomId", roomId);
        jsonObj.put("userId", userId);
        mTRTCCloud.ConnectOtherRoom(jsonObj.toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

// Result callback for requesting cross-room mic-connection
@Override
public void onConnectOtherRoom(String userId, int errCode, String errMsg) {
    // The user ID of the anchor in the other room you want to initiate the cross-r
    // Error code. ERR_NULL indicates the request is successful
    // Error message
}
```

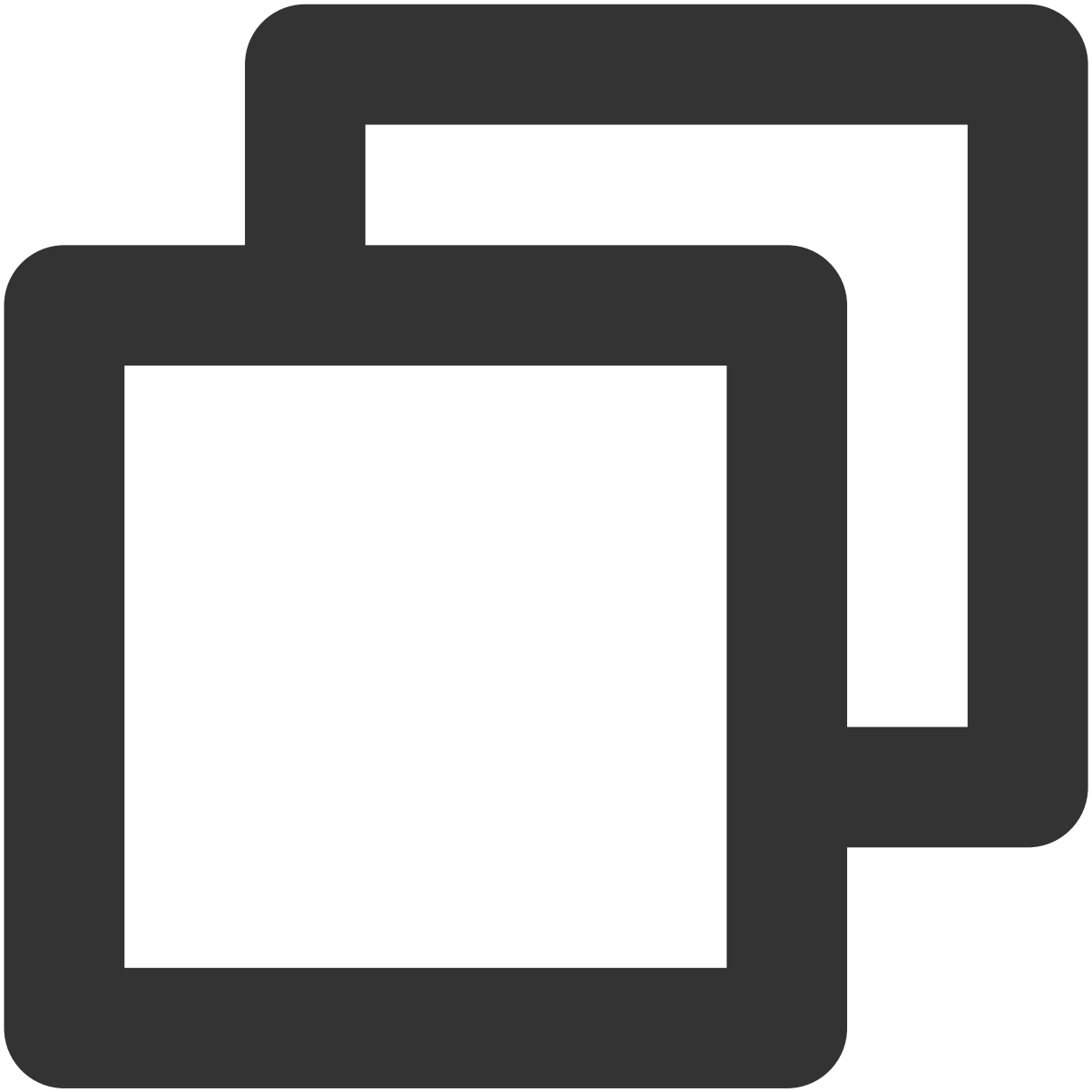
Note:

Both local and remote users participating in the cross-room mic-connection must be in the anchor role and must have audio/video uplink capabilities.

Cross-room mic-connection PK with multiple room anchors can be achieved by calling `ConnectOtherRoom()` multiple times. Currently, a room can connect with up to three other room anchors at most, and up to 10 anchors in a room can conduct cross-room mic-connection competition with anchors in other rooms.

2. All users in both rooms will receive a callback indicating that the audio and video streams from the PK anchor in the other room are available.

©2013-2022 Tencent Cloud. All rights reserved.

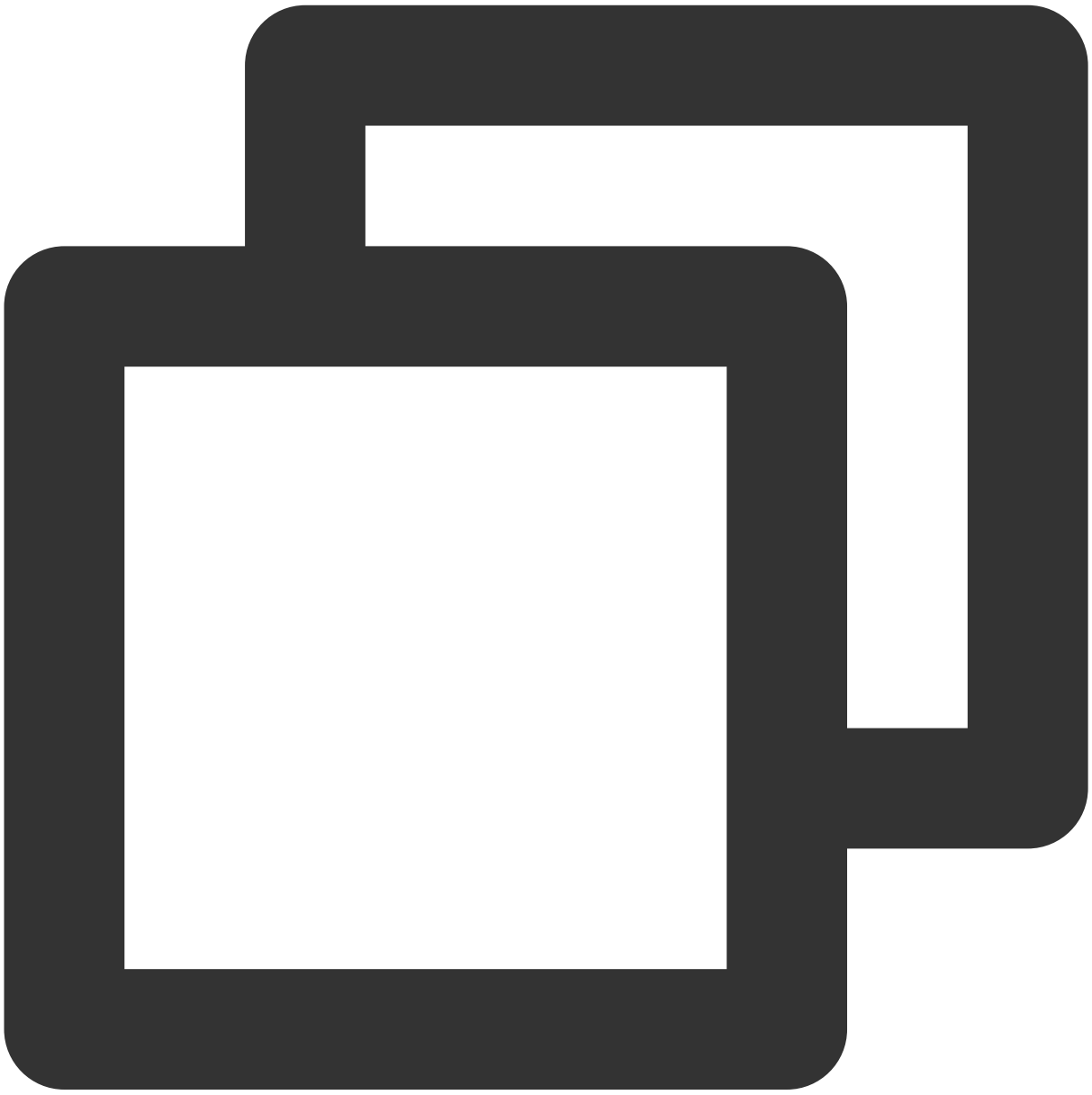


```
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

@Override
public void onUserVideoAvailable(String userId, boolean available) {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }
```

3. Either party exits the cross-room mic-connection PK.



```
// Exiting cross-room mic-connection
mTRTCCloud.DisconnectOtherRoom();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Result callback for exiting cross-room mic-connection
@Override
public void onDisConnectOtherRoom(int errCode, String errMsg) {
    super.onDisConnectOtherRoom(errCode, errMsg);
}
```

Note:

After calling `DisconnectOtherRoom()` , you may exit the cross-room mic-connection PK with all other room anchors.

Either the initiator or the receiver can call `DisconnectOtherRoom()` to exit the cross-room mic-connection PK.

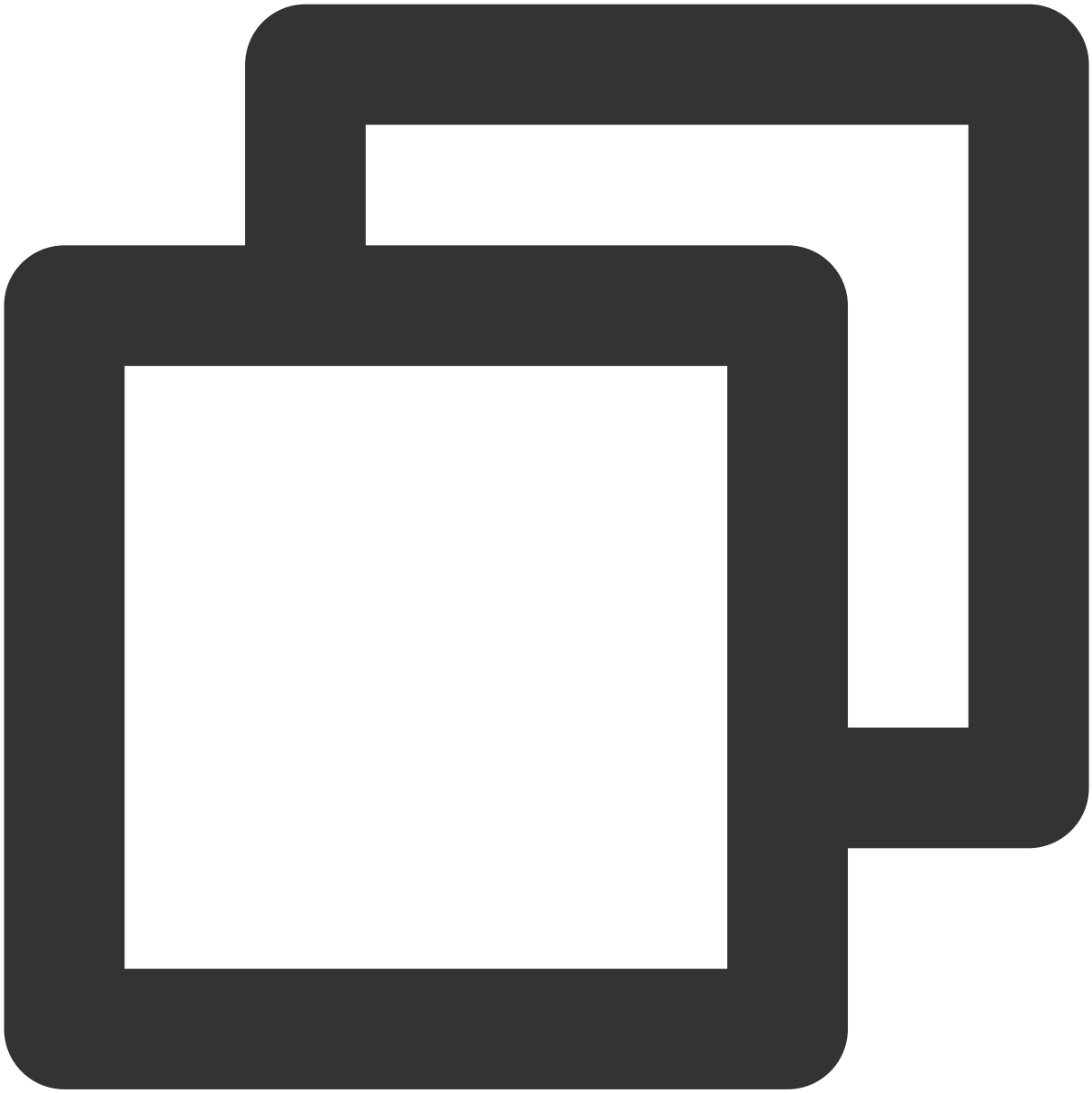
Third-Party Beauty Feature Integration

TRTC supports integrating third-party beauty effect products. Use the example of Special Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate the Special Effect SDK, and apply for an authorization license. For details, see Integration Preparation for steps.

2. Resource copying (if any). If your resource files are built into the assets directory, you need to copy them to the App's private directory before use.

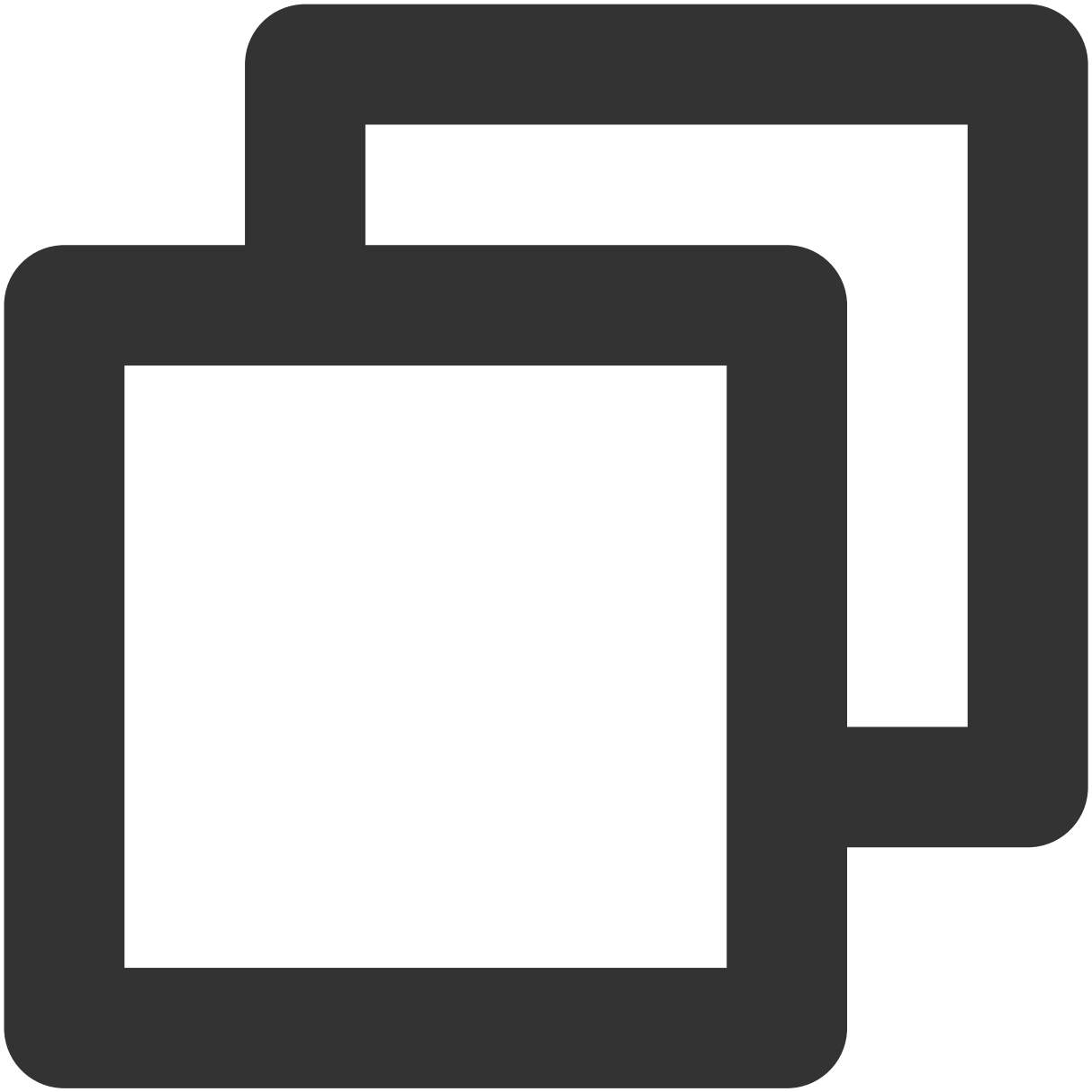
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").getAbsolutePath());
//loading

// Copy resource files to the private directory. Only need to do it once
XmagicResParser.copyRes(getApplicationContext());
```

If your resource file is dynamically downloaded from the internet, you need to set the resource file path after the download is successful.

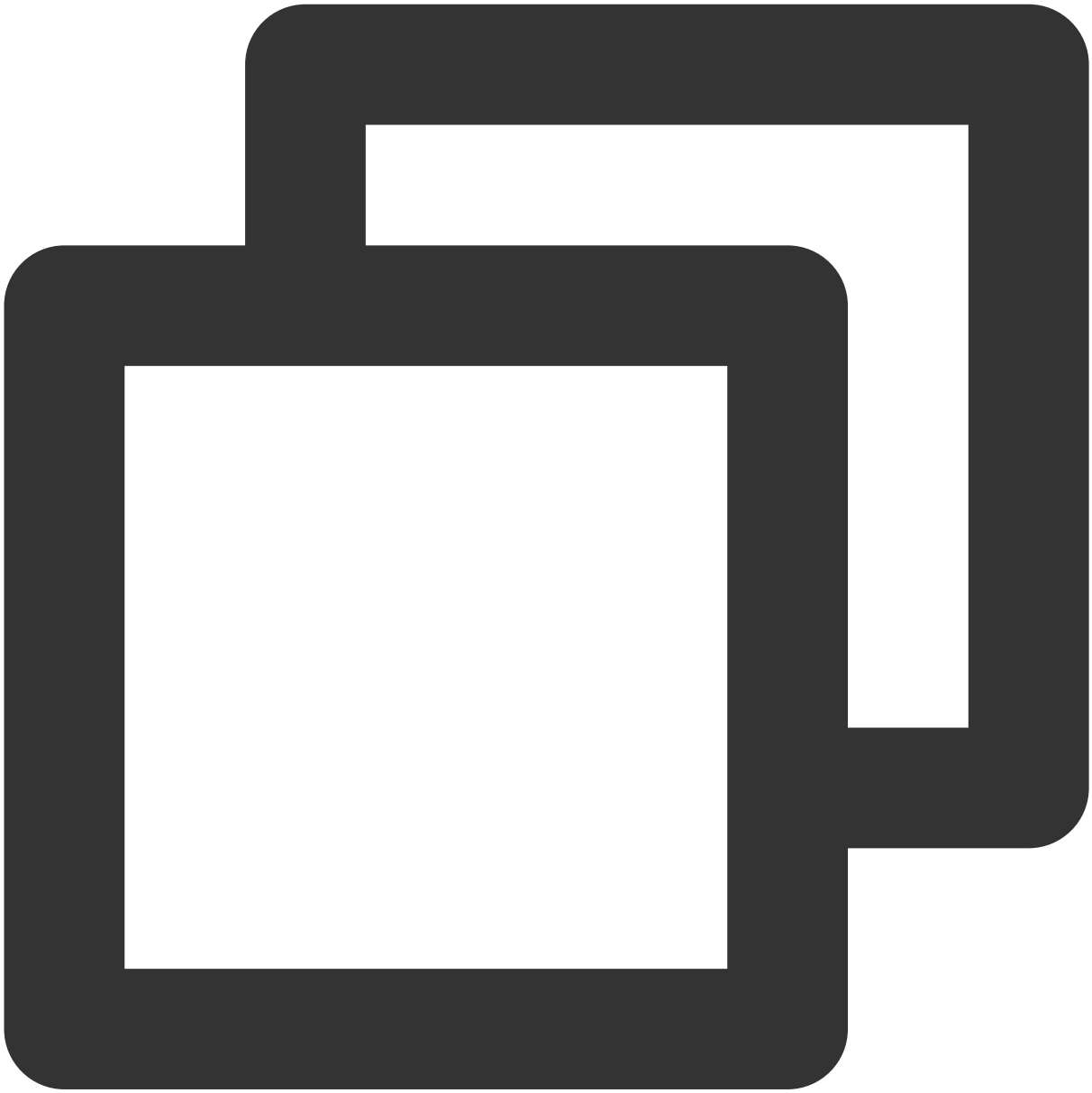
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath (local path of the downloaded resource file);
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
mTRTCCloud.setLocalVideoProcessListener(TRTCCloudDef.TRTC_VIDEO_PIXEL_FORMAT_Textur
    @Override
    public void onGLContextCreated() {
        // The OpenGL environment has already been set up internally within the SDK
        if (mXmagicApi == null) {
            XmagicApi mXmagicApi = new XmagicApi(context, XmagicResParser.getResPat
        } else {
            mXmagicApi.onResume();
        }
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudD
        // Callback for integrating with third-party beauty components for video pr
        if (mXmagicApi != null) {
            dstFrame.texture.textureId = mXmagicApi.process(srcFrame.texture.textur
        }
        return 0;
    }

    @Override
    public void onGLContextDestory() {
        // The internal OpenGL environment within the SDK has been terminated. At t
        mXmagicApi.onDestroy();
    }
});
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products, while **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

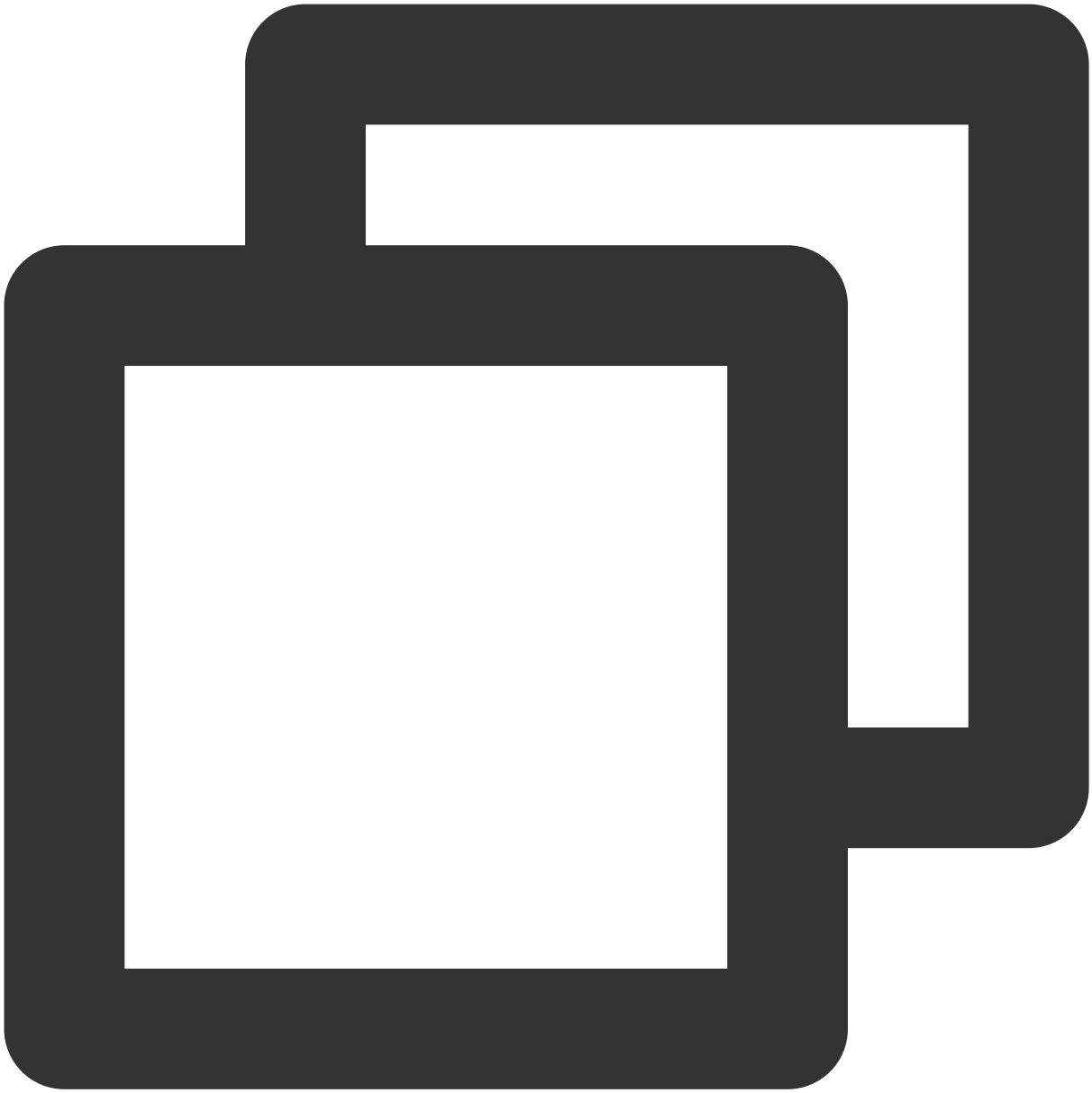
For scenario-specific integration guidelines of beauty effects, see Integrating Special Effect into TRTC SDK. For guidelines on integrating beauty effects independently, see Integrating Special Effect SDK.

Dual-Stream Encoding Mode

When the dual-stream encoding mode is enabled, the current user's encoder outputs two video streams, a high-definition large screen and a low-definition small screen, at the same time (but only one audio stream). In this way, other users in the room can choose to subscribe to the high-definition large screen or low-definition small screen based on their network conditions or screen sizes.

1. Enable large-and-small-screen dual-stream encoding mode.

©2013-2022 Tencent Cloud. All rights reserved.



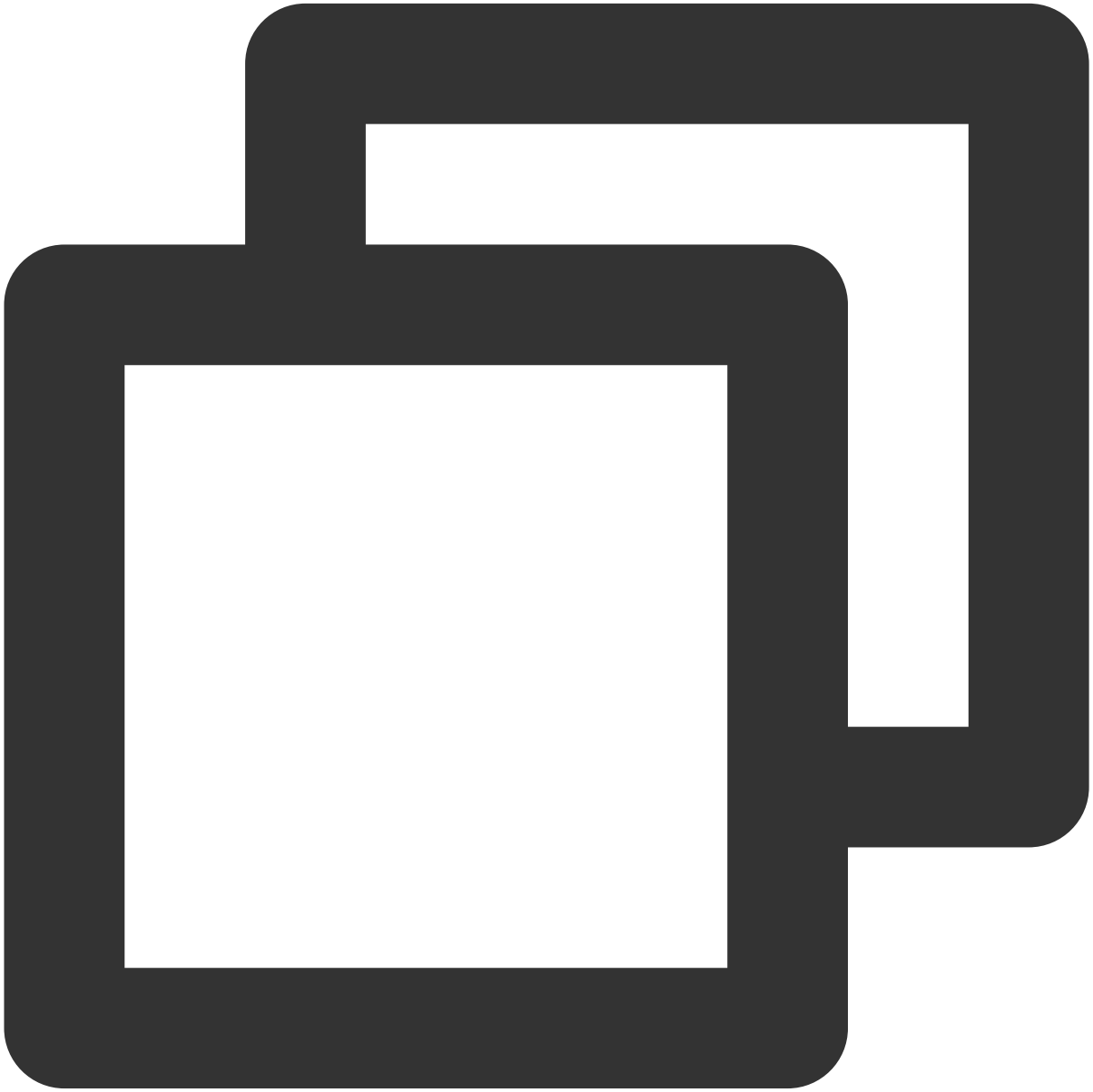
```
public void enableDualStreamMode(boolean enable) {
    // Video encoding parameters for the small stream (customizable).
    TRTCCloudDef.TRTCVideoEncParam smallVideoEncParam = new TRTCCloudDef.TRTCVideoE
    smallVideoEncParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_480_270
    smallVideoEncParam.videoFps = 15;
    smallVideoEncParam.videoBitrate = 550;
    smallVideoEncParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MOD
    mTRTCCloud.enableEncSmallVideoStream(enable, smallVideoEncParam);
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, it consumes more CPU and network bandwidth. Therefore, it may be considered for use on Mac, Windows, or high-performance Pads. It is not recommended for mobile devices.

2. Select the type of remote user's video stream to pull.



```
// Optional video stream types when you subscribe to a remote user's video stream
mTRTCCloud.startRemoteView(userId, streamType, videoView);

// You can switch the size of the specified remote user's screen at any time
mTRTCCloud.setRemoteVideoStreamType(userId, streamType);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

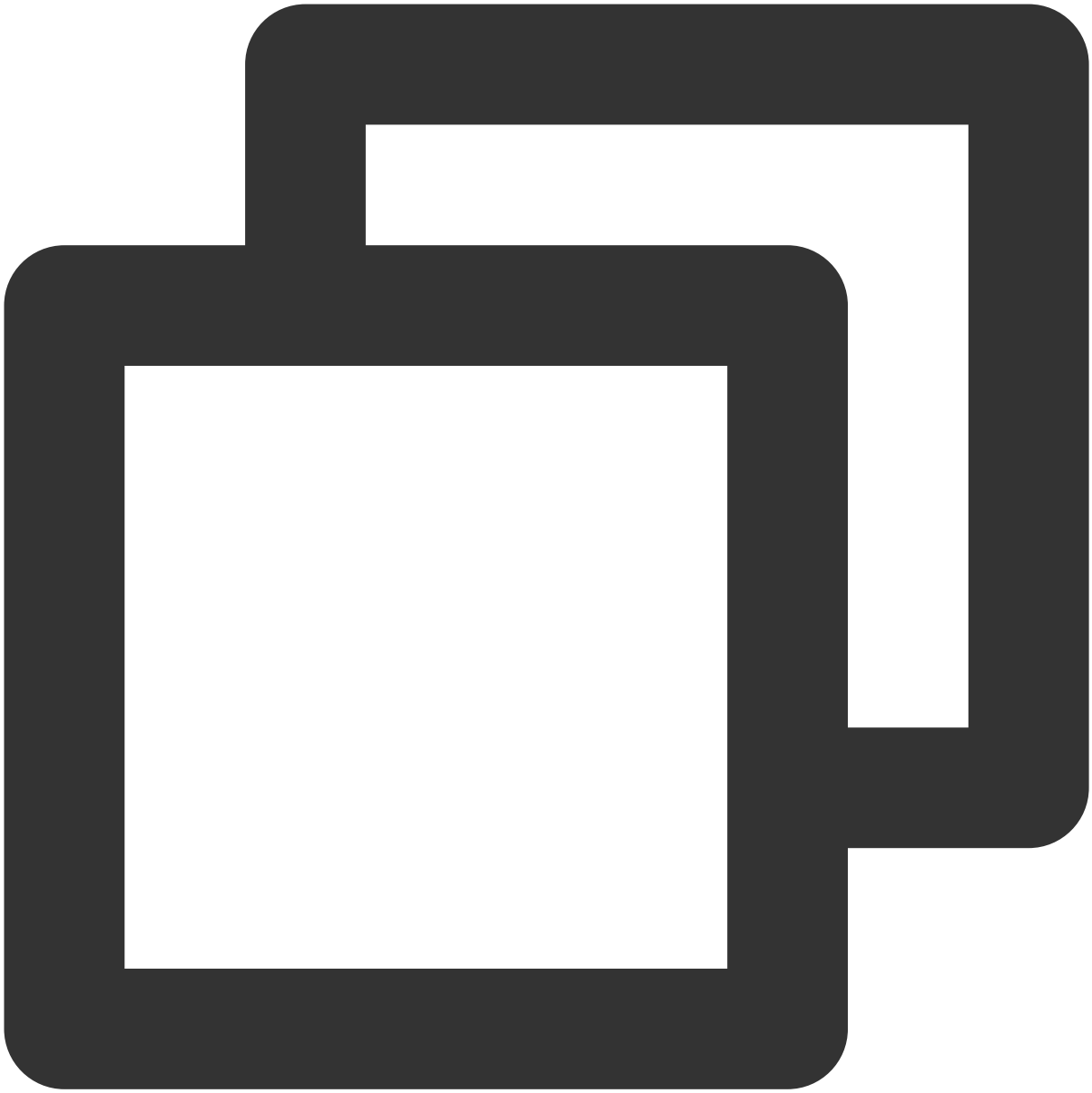
When the dual-stream encoding mode is enabled, you can specify the video stream type as `TRTC_VIDEO_STREAM_TYPE_SMALL` with `streamType` to pull a low-quality small video for viewing.

View rendering control

In TRTC, there are many APIs that require you to control the video screen. All these APIs require you to specify a video rendering control. On the Android platform, `TXCloudVideoView` is used as the video rendering control, and both `SurfaceView` and `TextureView` rendering schemes are supported. Below are the methods for specifying the type of rendering control and updating the video rendering control.

1. If you want mandatory use of a certain scheme, or to convert the local video rendering control to `TXCloudVideoView`, you can code as follows.

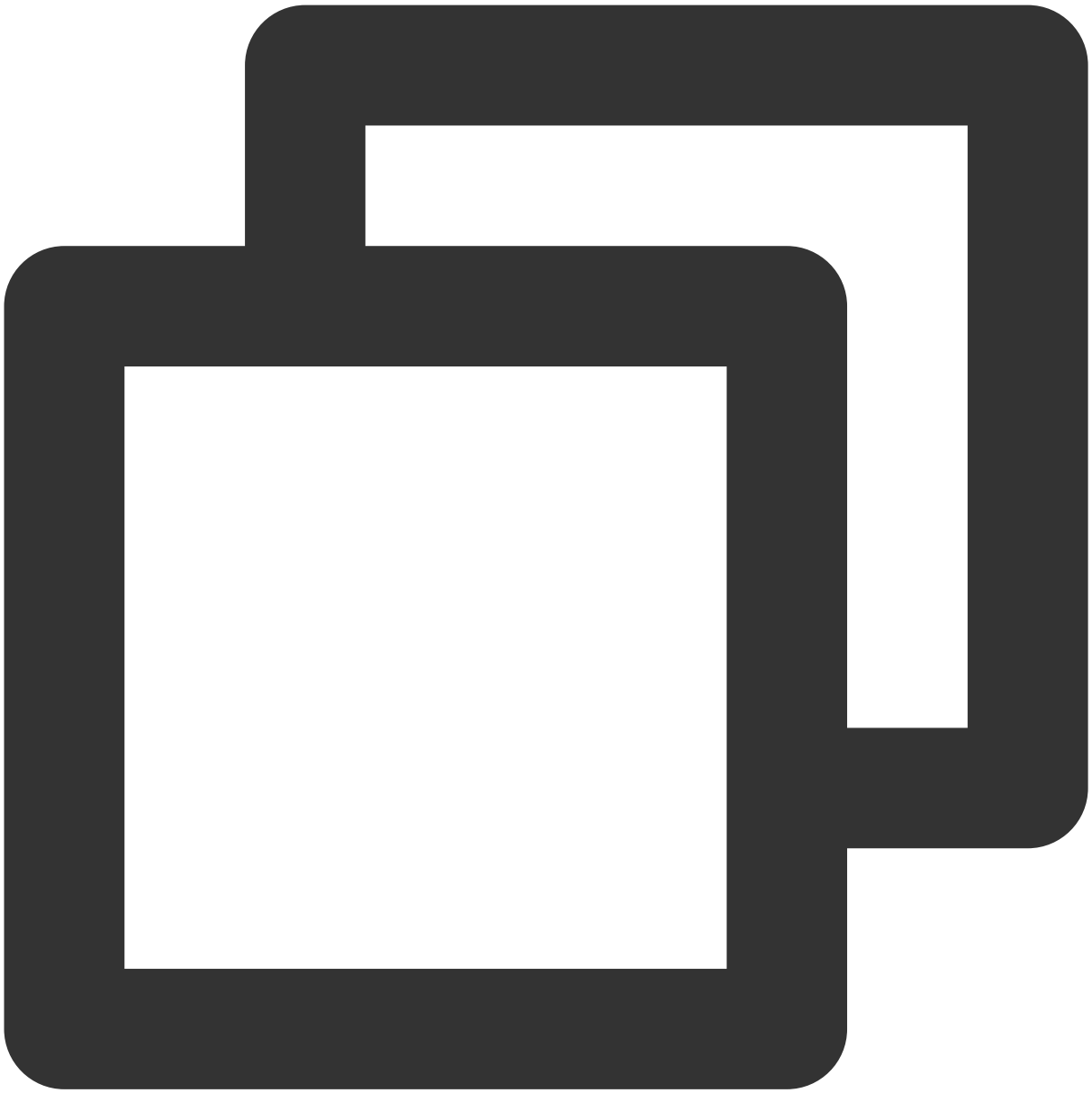
©2013-2022 Tencent Cloud. All rights reserved.



```
// Mandatory use of TextureView
TextureView textureView = findViewById(R.id.texture_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(context);
cloudVideoView.addVideoView(textureView);

// Mandatory use of SurfaceView
SurfaceView surfaceView = findViewById(R.id.surface_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(surfaceView);
```

©2013-2022 Tencent Cloud. All rights reserved.

2. If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.



```
// Update local preview screen rendering control
mTRTCCloud.updateLocalView(videoView);

// Update the remote user's video rendering control
mTRTCCloud.updateRemoteView(userId, streamType, videoView);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

The pass-through parameter `videoView` refers to the target video rendering control. And `streamType` only supports `TRTC_VIDEO_STREAM_TYPE_BIG` and `TRTC_VIDEO_STREAM_TYPE_SUB`.

Exception Handling

Exception error handling

When the TRTC SDK encounters an unrecoverable error, the error is thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure leads to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If room entry is failed, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect.Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Nnote that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.

©2013-2022 Tencent Cloud. All rights reserved.

ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

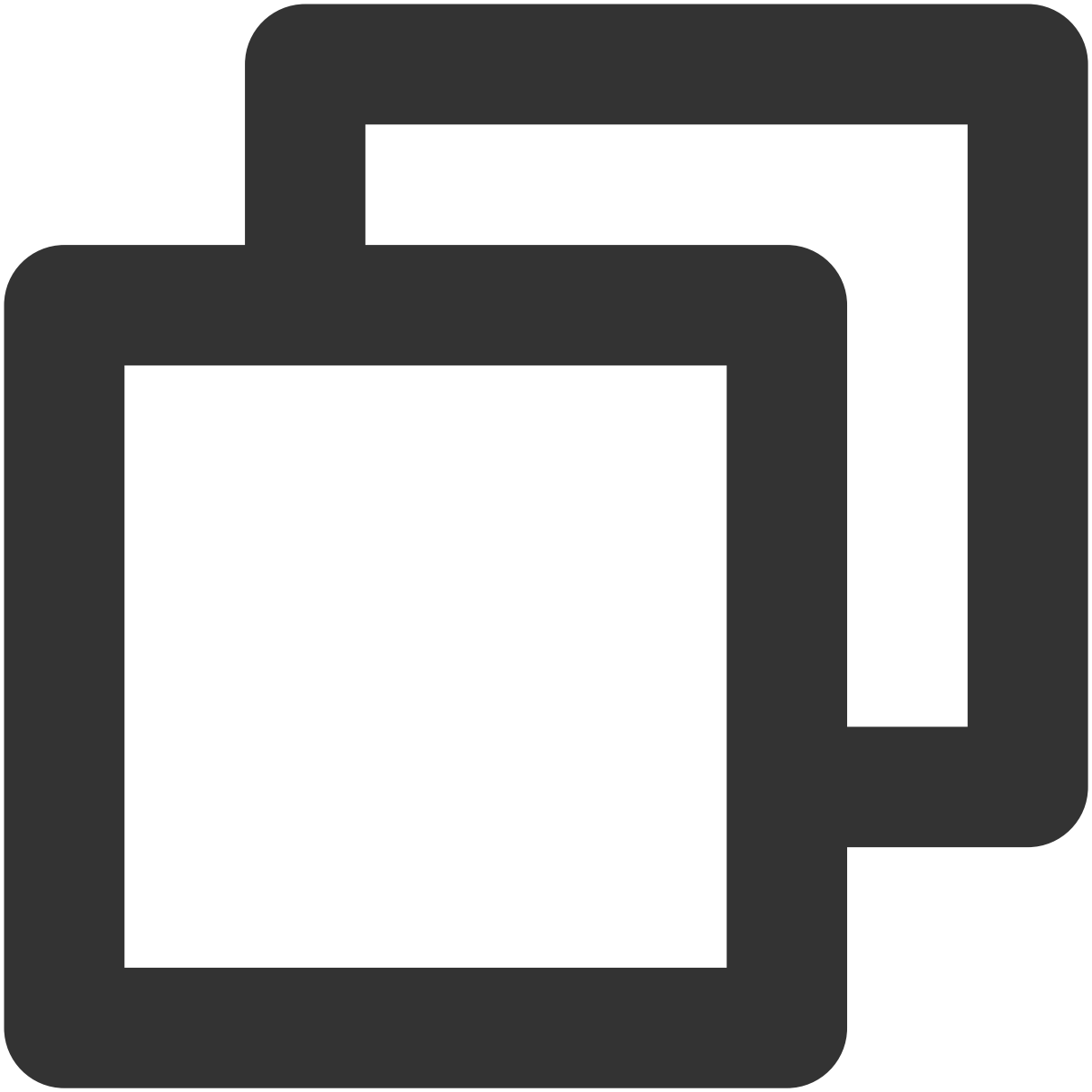
Errors for related monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to enable the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Issues with the remote mirror mode not functioning properly

In TRTC, video mirror settings are divided into local preview mirror `setLocalRenderParams` and video encoding mirror `setVideoEncoderMirror` . These settings separately affect the mirror effect of the local preview and the video encoding output (the mirror mode for remote viewers and cloud recordings). If you expect the mirror effect seen in the local preview to also take effect on the remote viewer's end, follow these encoding procedures.

©2013-2022 Tencent Cloud. All rights reserved.



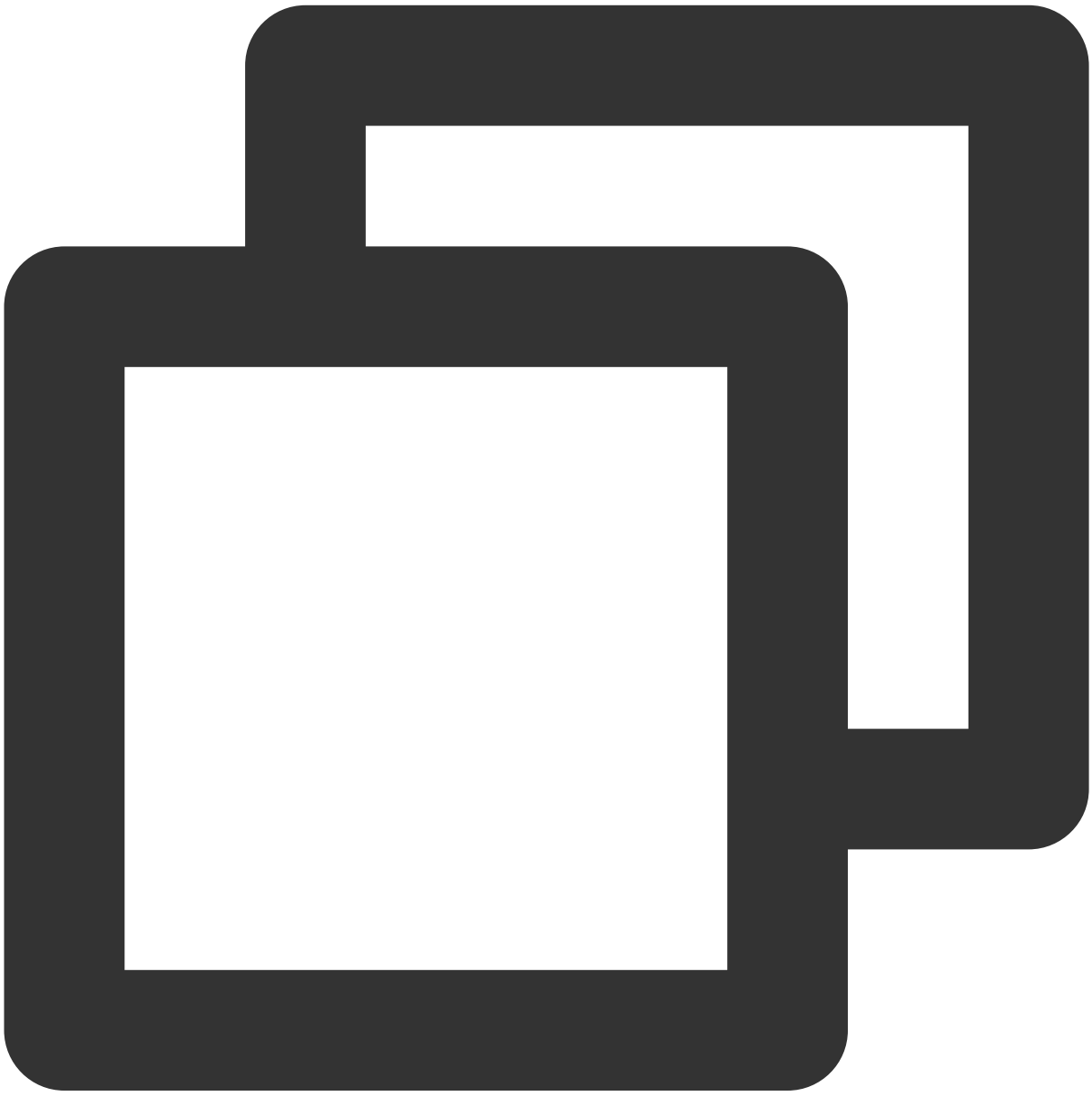
```
// Set the rendering parameters for the local video
TRTCCloudDef.TRTCRenderParams params = new TRTCCloudDef.TRTCRenderParams();
params.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_ENABLE; // Video mirror mod
params.fillMode = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL; // Video fill mode
params.rotation = TRTCCloudDef.TRTC_VIDEO_ROTATION_0;           // Video rotation a
mTRTCCloud.setLocalRenderParams(params);

// Set the video mirror mode for the encoder output
mTRTCCloud.setVideoEncoderMirror(true);
```

©2013-2022 Tencent Cloud. All rights reserved.

Issues with camera scale, focus, and switch

In e-commerce live streaming scenarios, the anchor may need to custom adjust the camera settings. The TRTC SDK's device management class provides APIs for these needs.

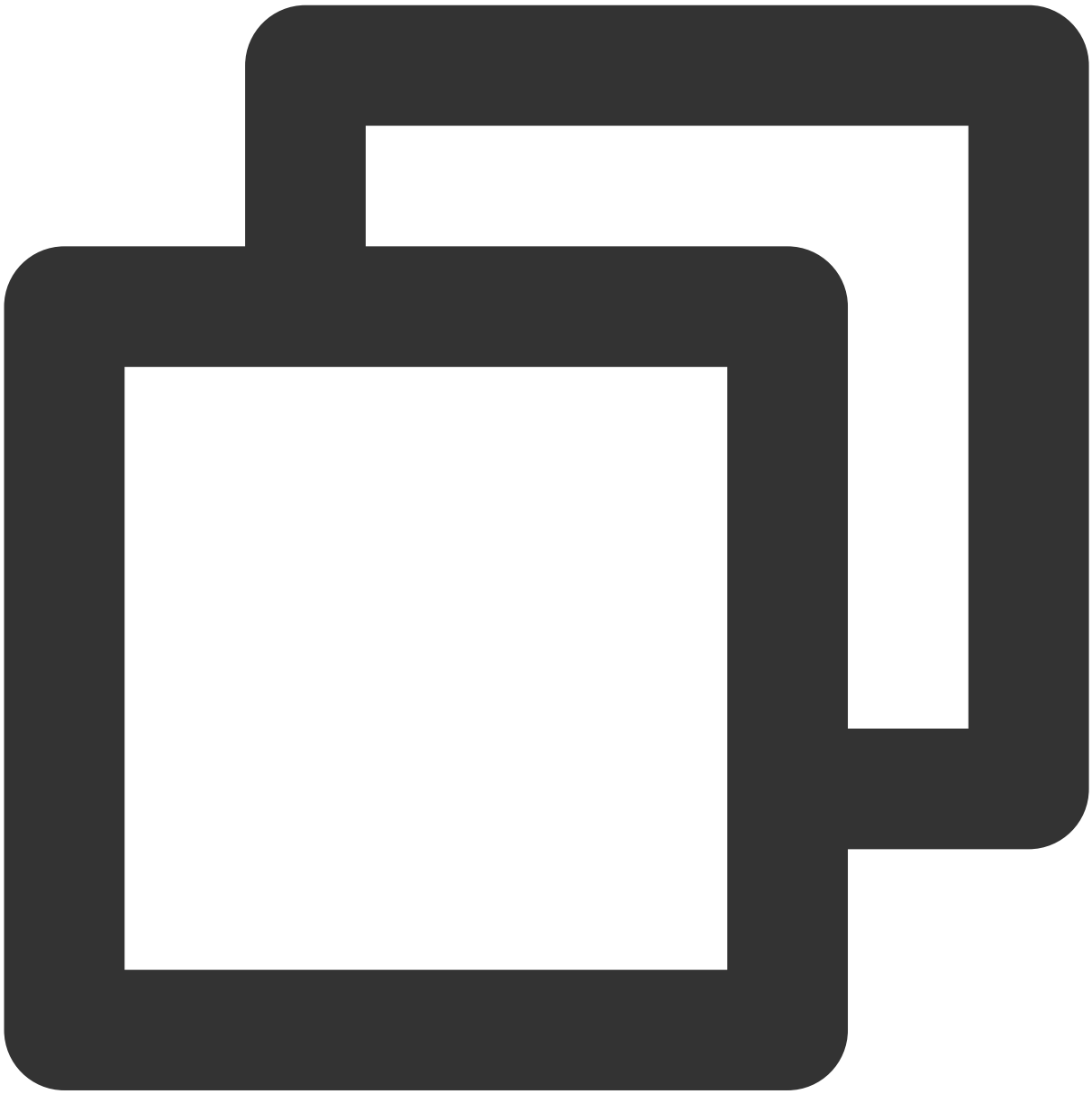
1. Query and set the zoom factor for the camera.



```
// Get the maximum zoom factor for the camera (only for mobile devices)
float zoomRatio = mTRTCCloud.getDeviceManager().getCameraZoomMaxRatio();
// Set the zoom factor for the camera (only for mobile devices)
// Value range is 1-5. 1 means the furthest field of view (normal lens), and 5 mean
mTRTCCloud.getDeviceManager().setCameraZoomRatio(zoomRatio);
```

©2013-2022 Tencent Cloud. All rights reserved.

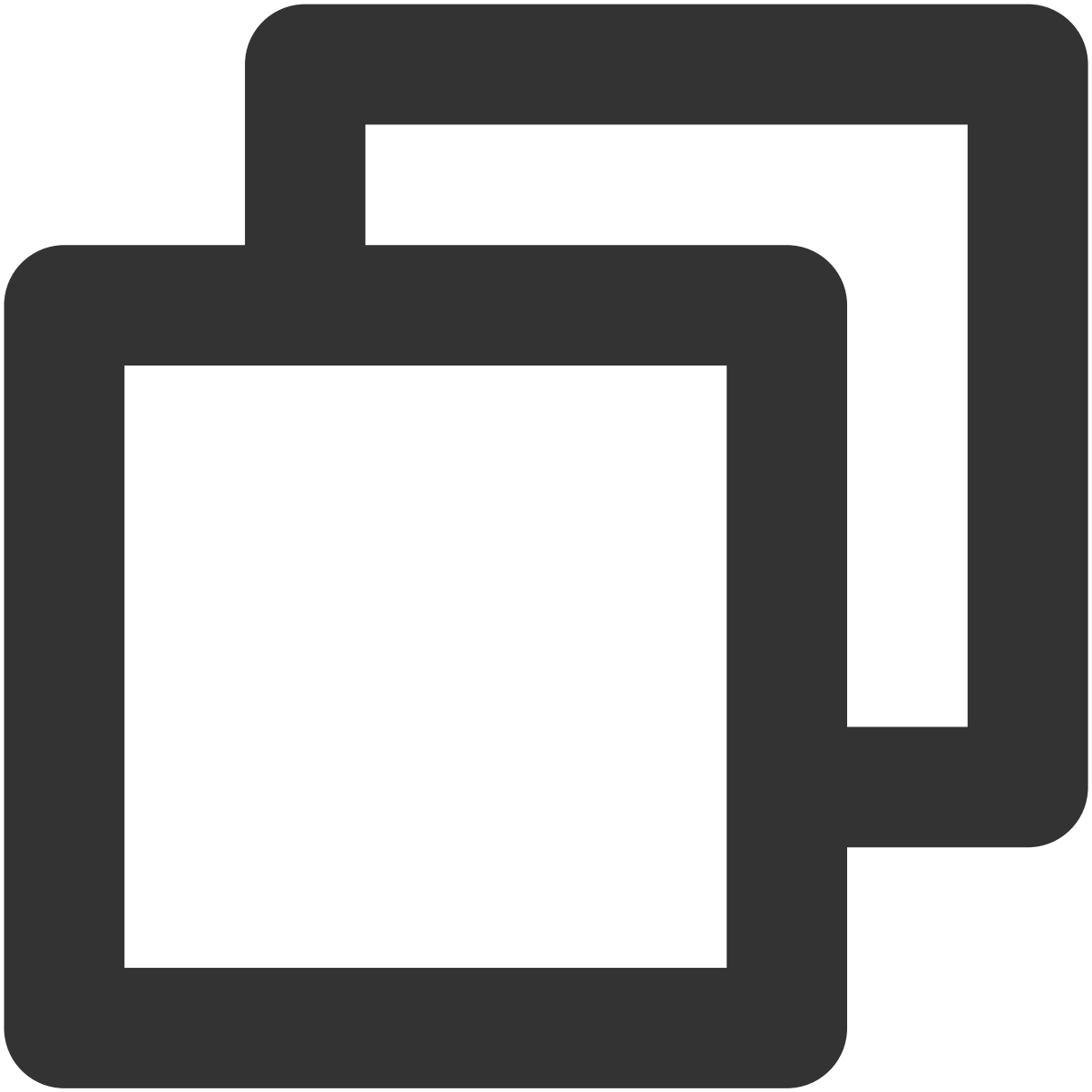
2. Set the focus feature and position of the camera.



```
// Enable or disable the camera's autofocus feature (only for mobile devices)
mTRTCCloud.getDeviceManager().enableCameraAutoFocus(false);
// Set the focus position of the camera (only for mobile devices)
// The precondition for using this API is to first disable the autofocus feature us
mTRTCCloud.getDeviceManager().setCameraFocusPosition(int x, int y);
```

3. Determine and switch to front or rear cameras.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Determine if the current camera is the front camera (only for mobile devices)
boolean isFrontCamera = mTRTCCloud.getDeviceManager().isFrontCamera();
// Switch to front or rear cameras (only for mobile devices)
// Passing true means switching to front, and passing false means switching to rear
mTRTCCloud.getDeviceManager().switchCamera(!isFrontCamera);
```

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-18 14:26:15

Business Process

This document summarizes some common business processes in the e-commerce live streaming scenario, helping you better understand the implementation process of the entire scenario.

Anchor starts and ends live broadcast

Anchor initiates the cross-room mic-connection PK

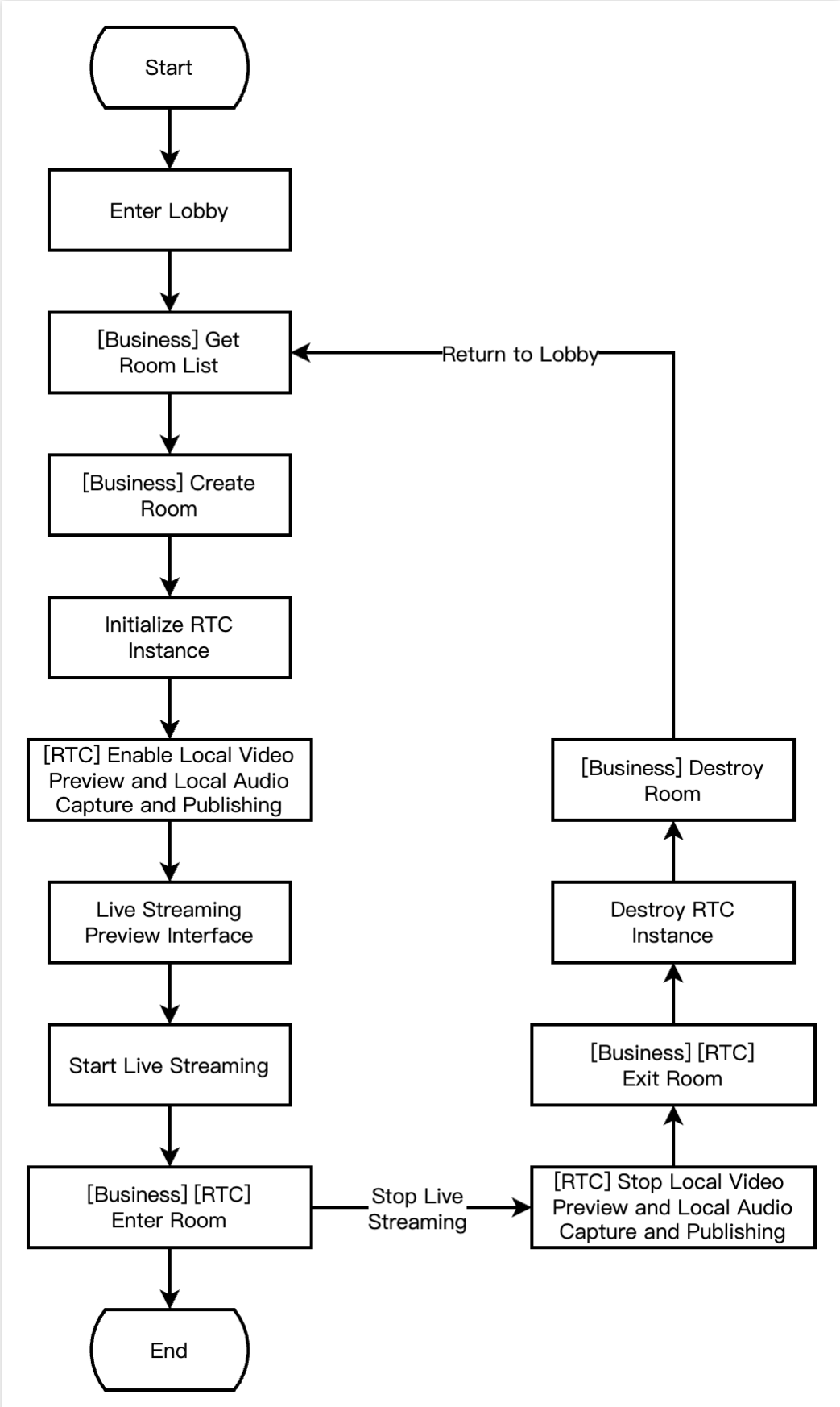
The RTC audience enters the room for mic-connection

The CDN audience enters the room for mic-connection

Product Management for Merchandising

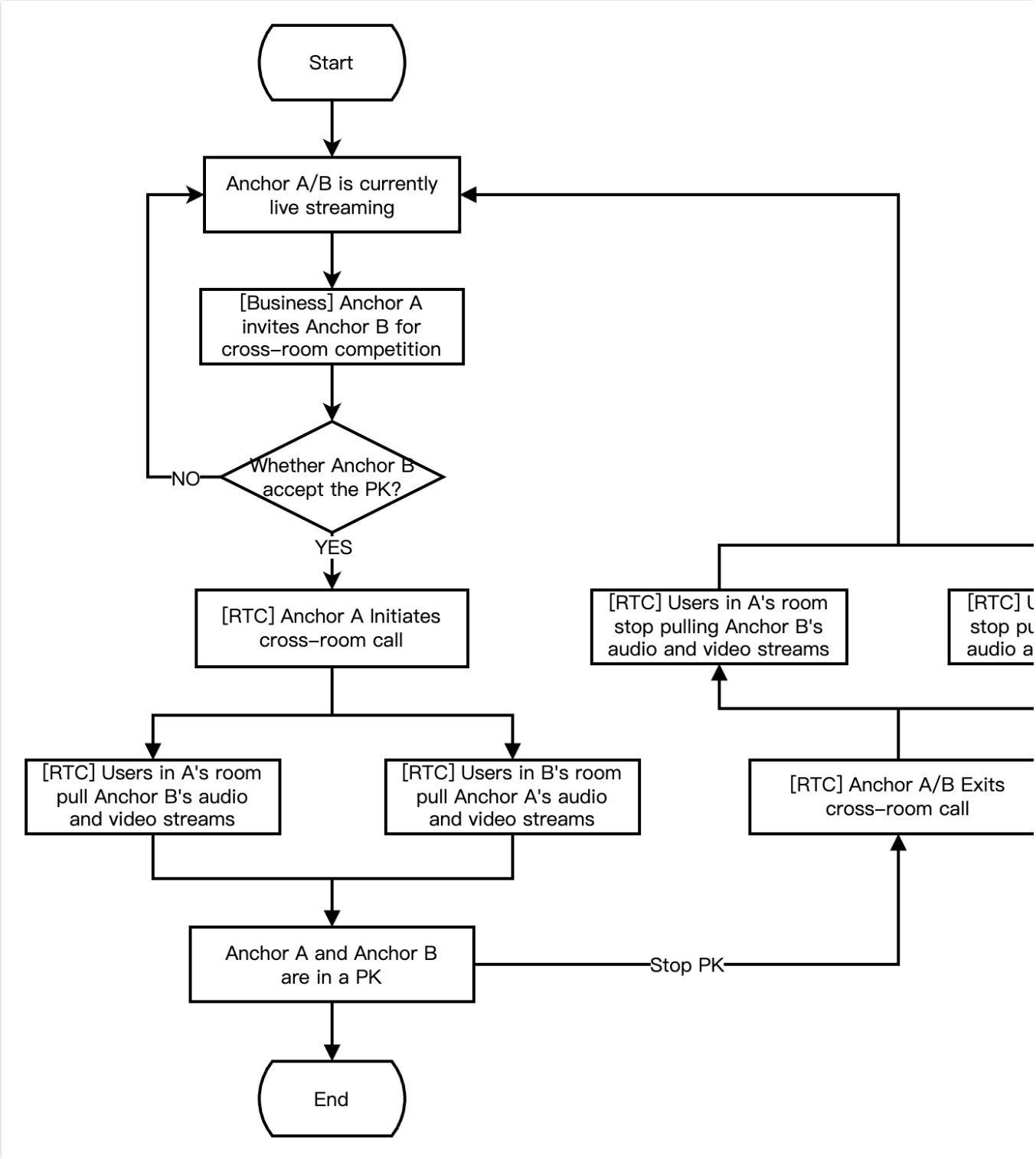
The following diagram shows the process of an anchor (room owner) local preview, creating a room, entering a room to start live streaming, and leaving the room to end the live streaming.

©2013-2022 Tencent Cloud. All rights reserved.



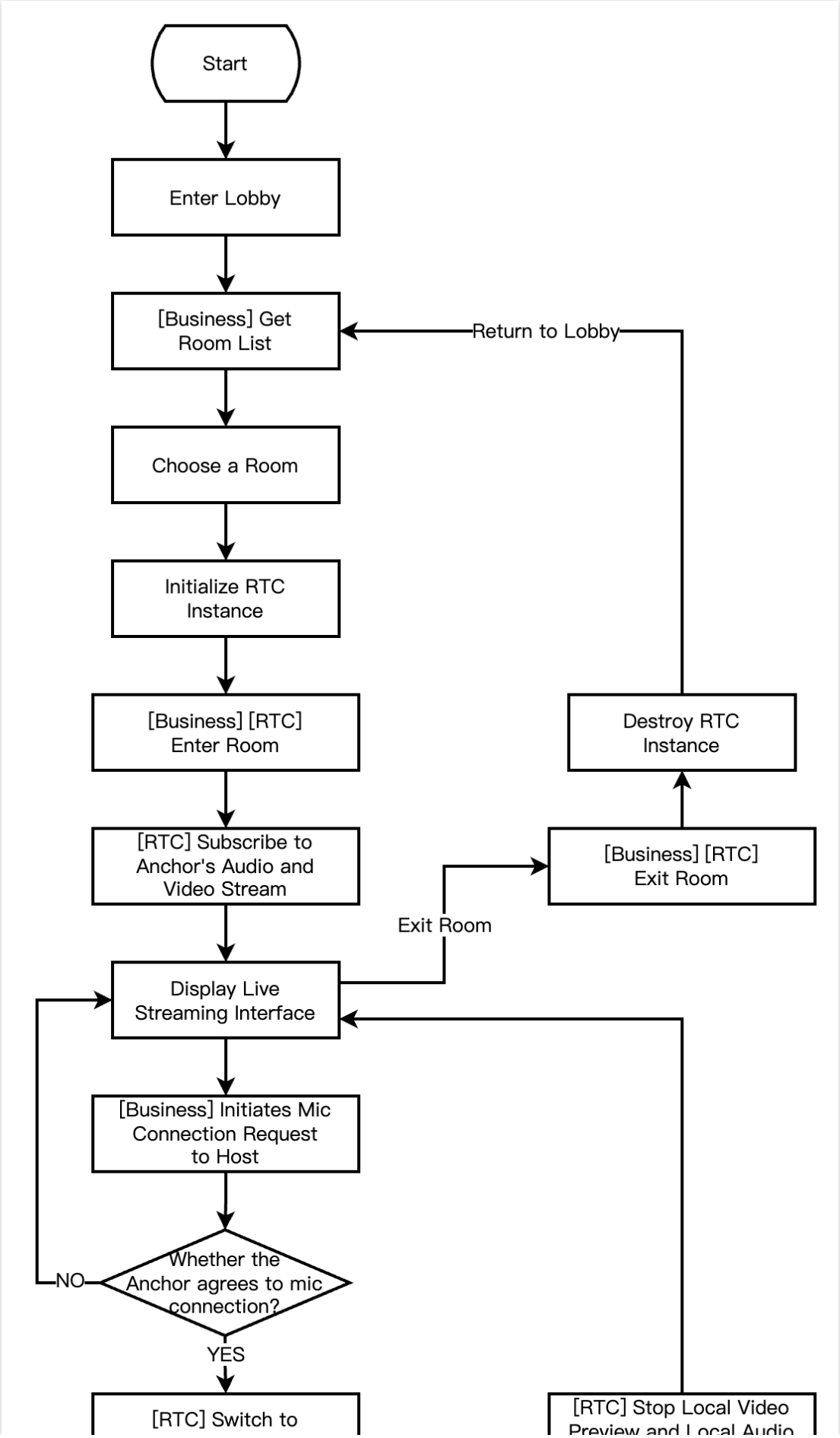
©2013-2022 Tencent Cloud. All rights reserved.

The following diagram shows the process of Anchor A inviting Anchor B for a cross-room PK. During the cross-room PK, the audiences in both rooms can see the PK mic-connection live streaming of the two room owners.

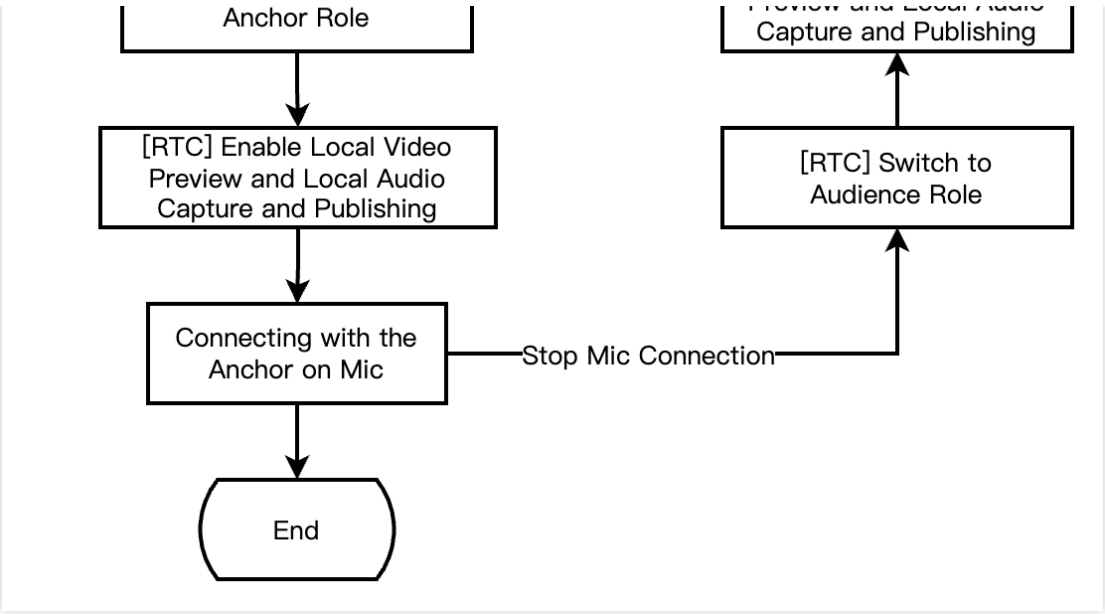


The following diagram shows the process for RTC live interactive streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

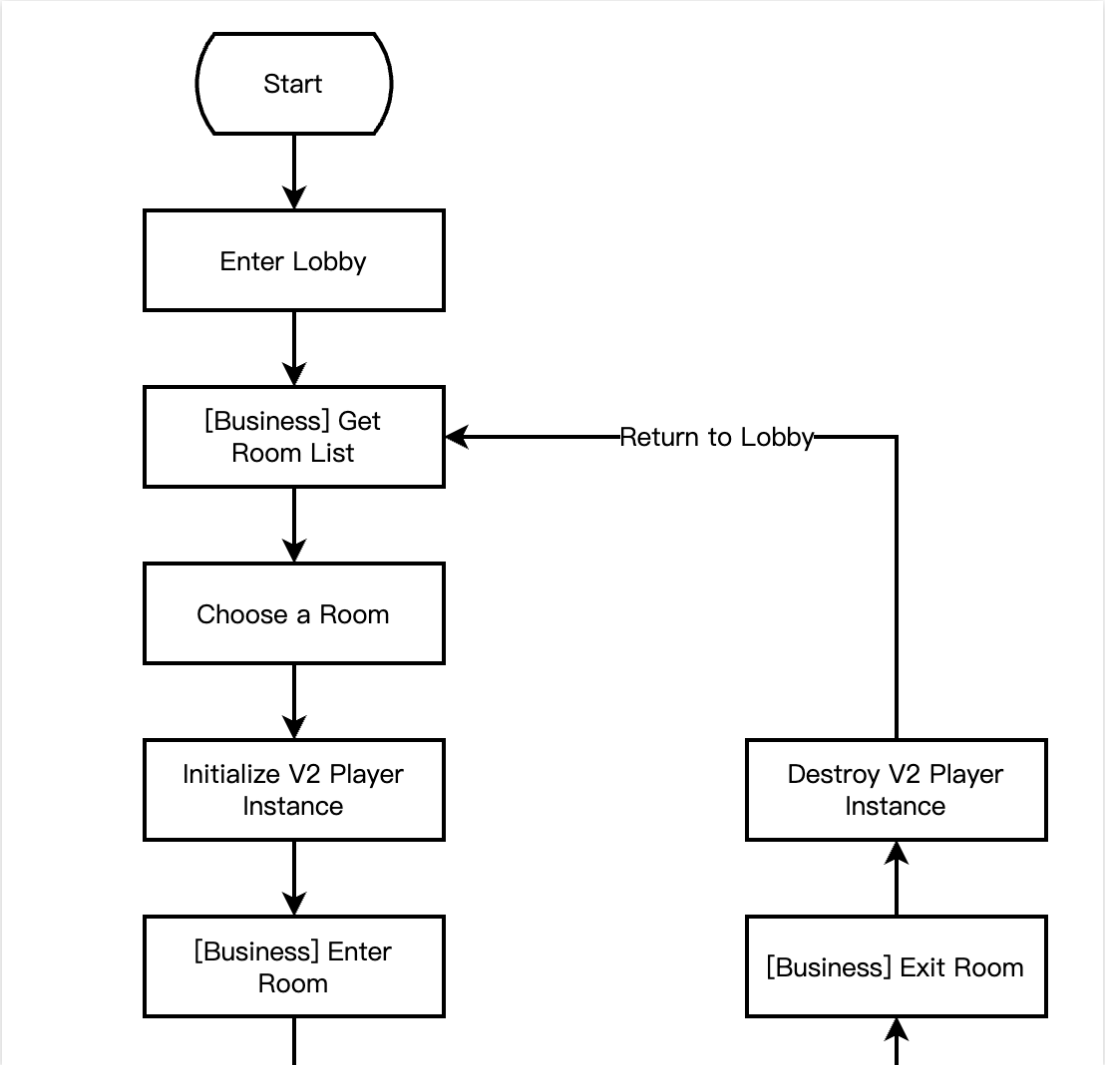
©2013-2022 Tencent Cloud. All rights reserved.



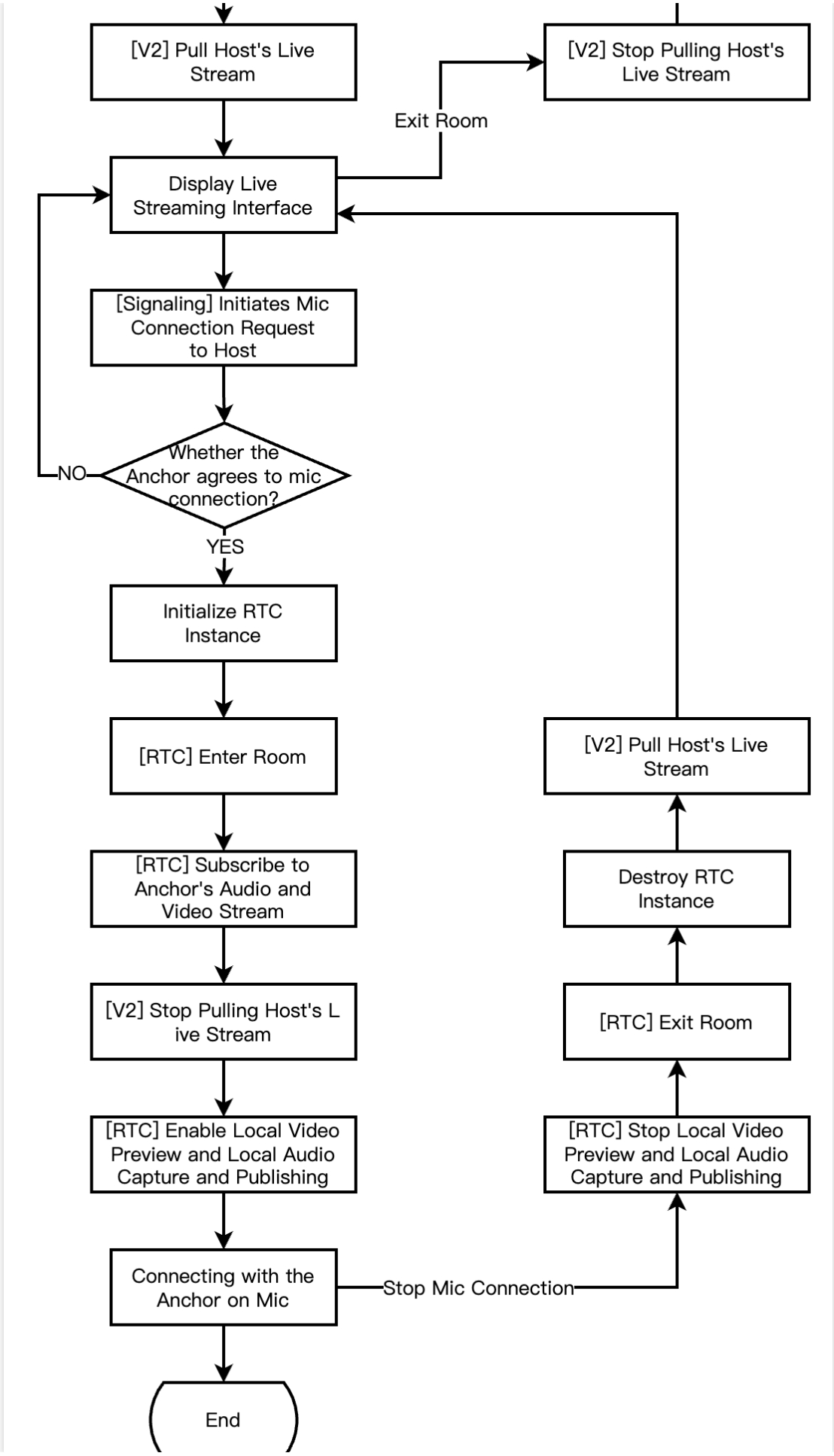
©2013-2022 Tencent Cloud. All rights reserved.



The following diagram shows the process for RTC CDN live streaming audience to enter the room, apply for the mic-connection, end the mic-connection, and exit the room.

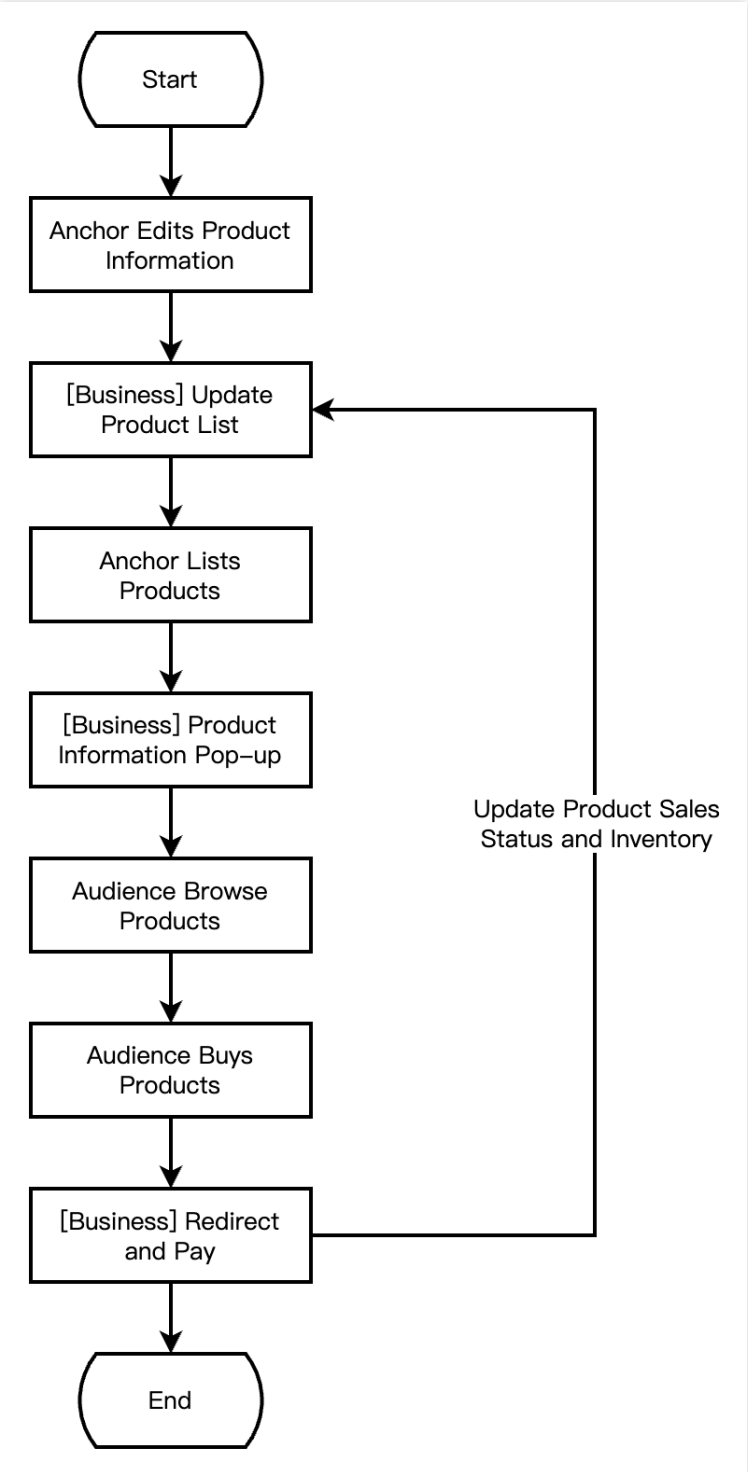


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

The diagram below shows the process in live streaming merchandising scenarios, where the anchor edits and lists products, while audience browses and purchases products.



©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparations

Step 1: activate the service

E-commerce live streaming scenarios usually rely on paid PaaS services such as Real-Time Communication (TRTC), Beauty Special Effect, Player SDK. Among them, TRTC provides real-time audio and video interactive capabilities, Special Effect provides beauty special effects, and the player is responsible for live and on-demand playback. You can freely choose to activate the above services according to your actual business needs.

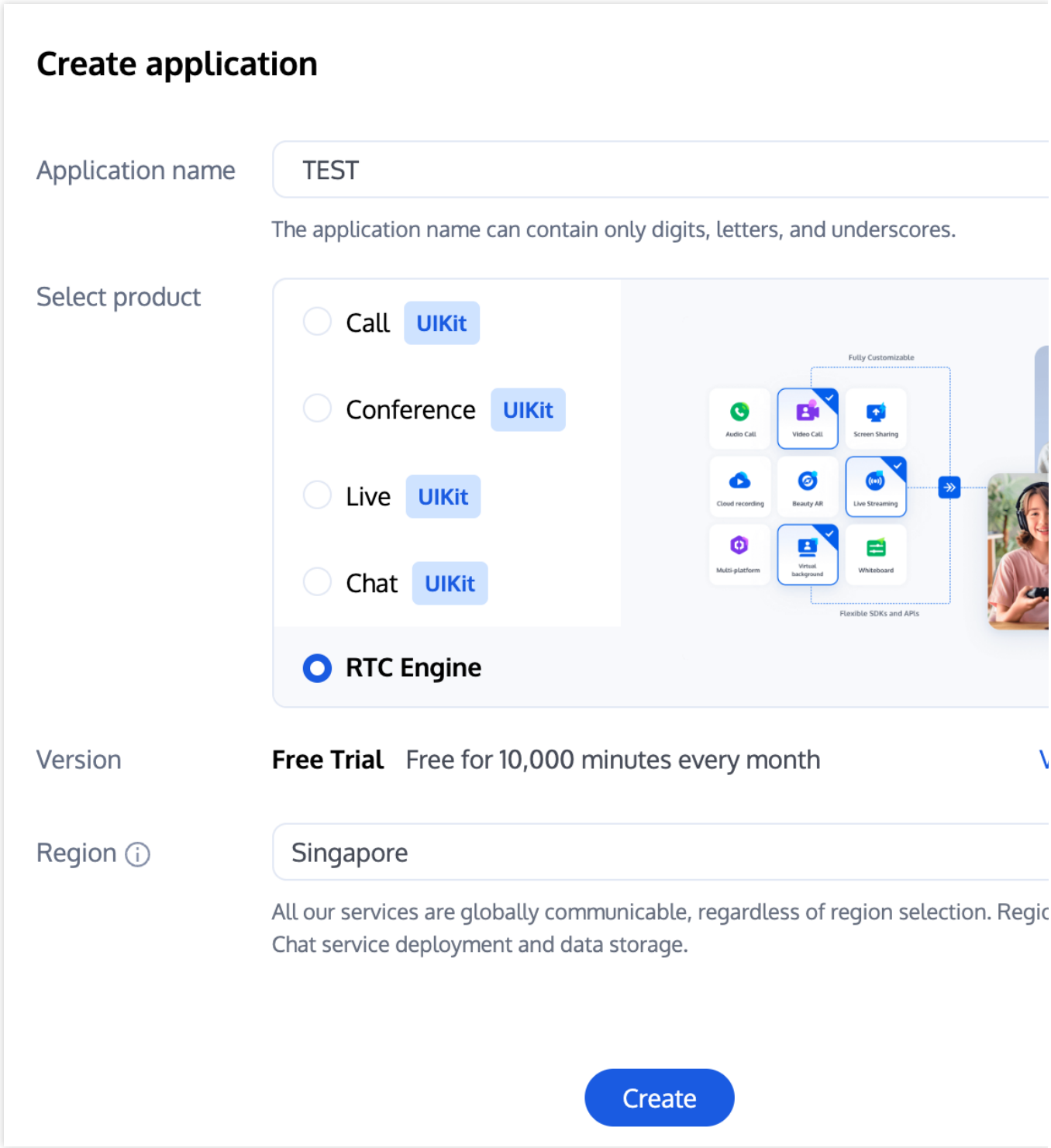
Activate TRTC Service

Activate Special Effect Service

Activate Player Service

1. First, you need to log in to the TRTC Console to create an application. You can choose to upgrade the TRTC application version according to your needs. For example, the professional edition unlocks more value-added feature services.

©2013-2022 Tencent Cloud. All rights reserved.



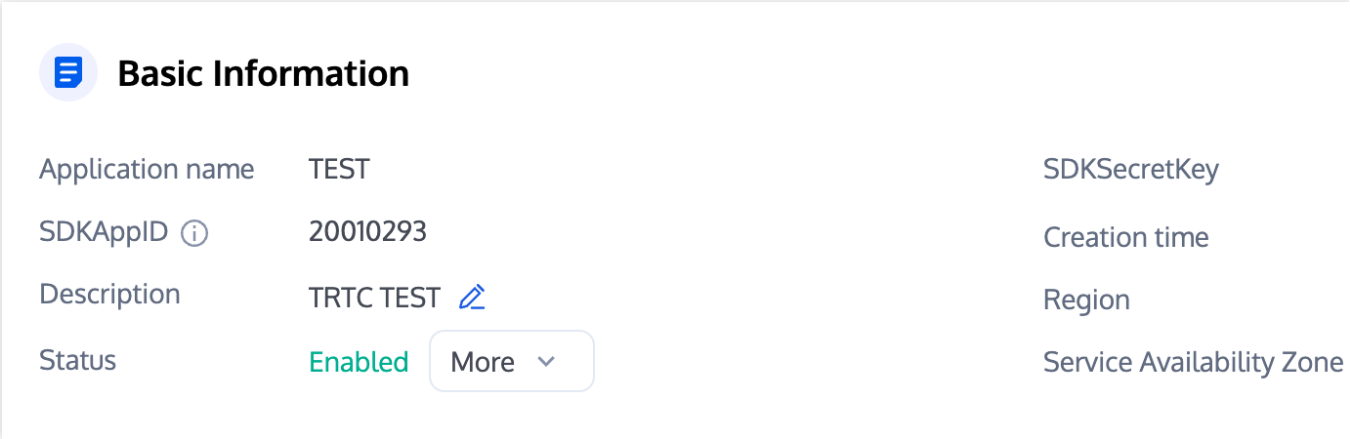
Note:

It is recommended to create two separate applications for testing and production environments. Each account (UIN) is provided with 10,000 minutes of free usage per month within one year.

The TRTC monthly package is divided into Trial Version (by default), Basic Version, and Professional Version, which can unlock different value-added features and services. For details, see Version Features and Monthly Package Description.

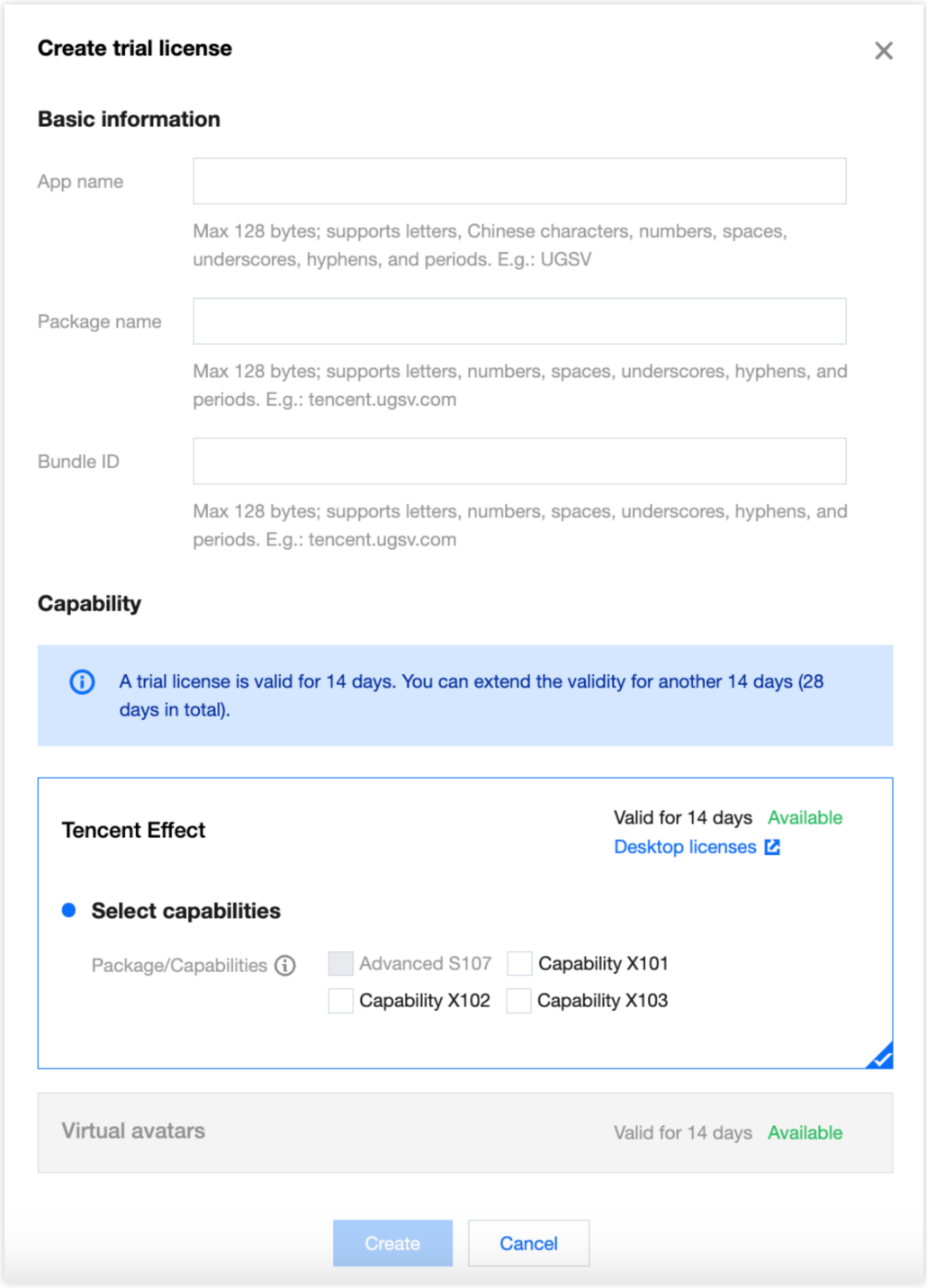
©2013-2022 Tencent Cloud. All rights reserved.

2. Once the application is created, you can find basic information about it under the Application Management - Application Overview section. It is important to store the **SDKAppID** and **SDKSecretKey** for later use and to avoid key leakage to prevent unauthorized traffic usage.



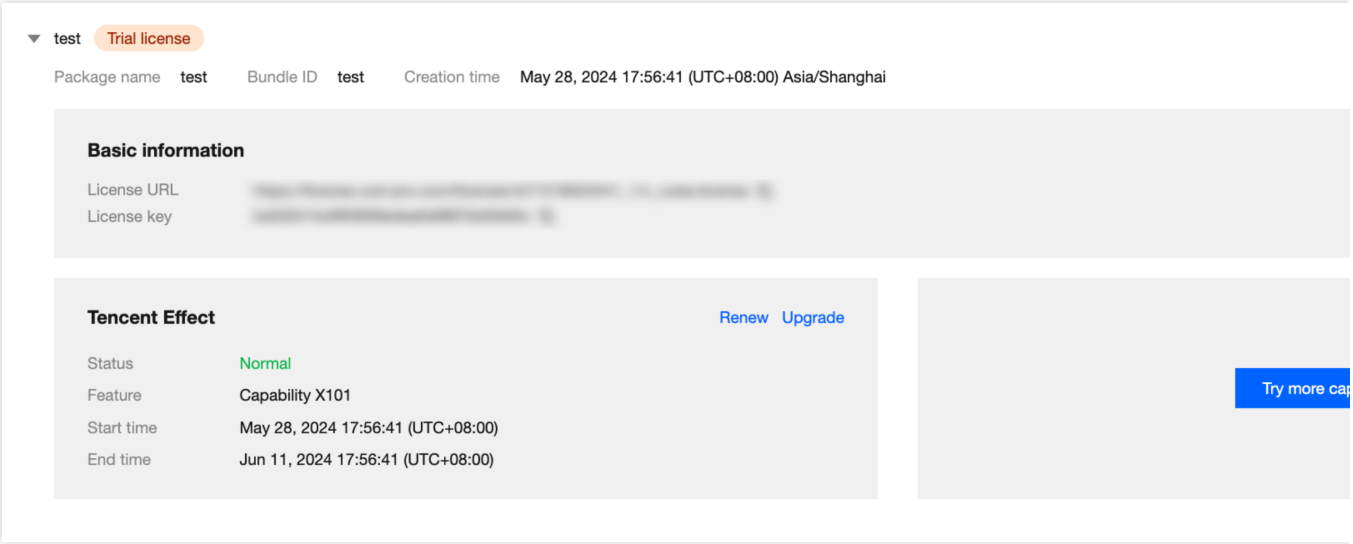
1. Log in to Cloud Special Effect Console > Mobile License. Click **Create Trial License** (the free trial validity period for Trial Version License is 14 days. It is extendable once for a total of 28 days). Fill in the actual requirements for `App Name` , `Package Name` and `Bundle ID` . Select **Special Effect**, and choose the capabilities to be tested: Advanced Package S1-07, Atomic Capability X1-01, Atomic Capability X1-02, and Atomic Capability X1-03. **After you check it, accurately fill in the company name, and industry type. Upload** Company Service License, click **OK** to submit the review application, and wait for the manual review process.

©2013-2022 Tencent Cloud. All rights reserved.

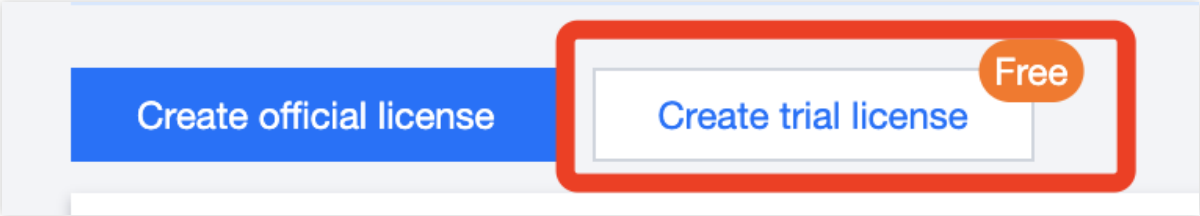


2. After the Trial License is successfully created, the page will display the generated License information. At this time, the License URL and License Key parameters are not yet effective and will only become active after the submission is approved. **When configuring SDK initialization, you need to input both the License URL and License Key parameters. Keep the following information secure.**

©2013-2022 Tencent Cloud. All rights reserved.

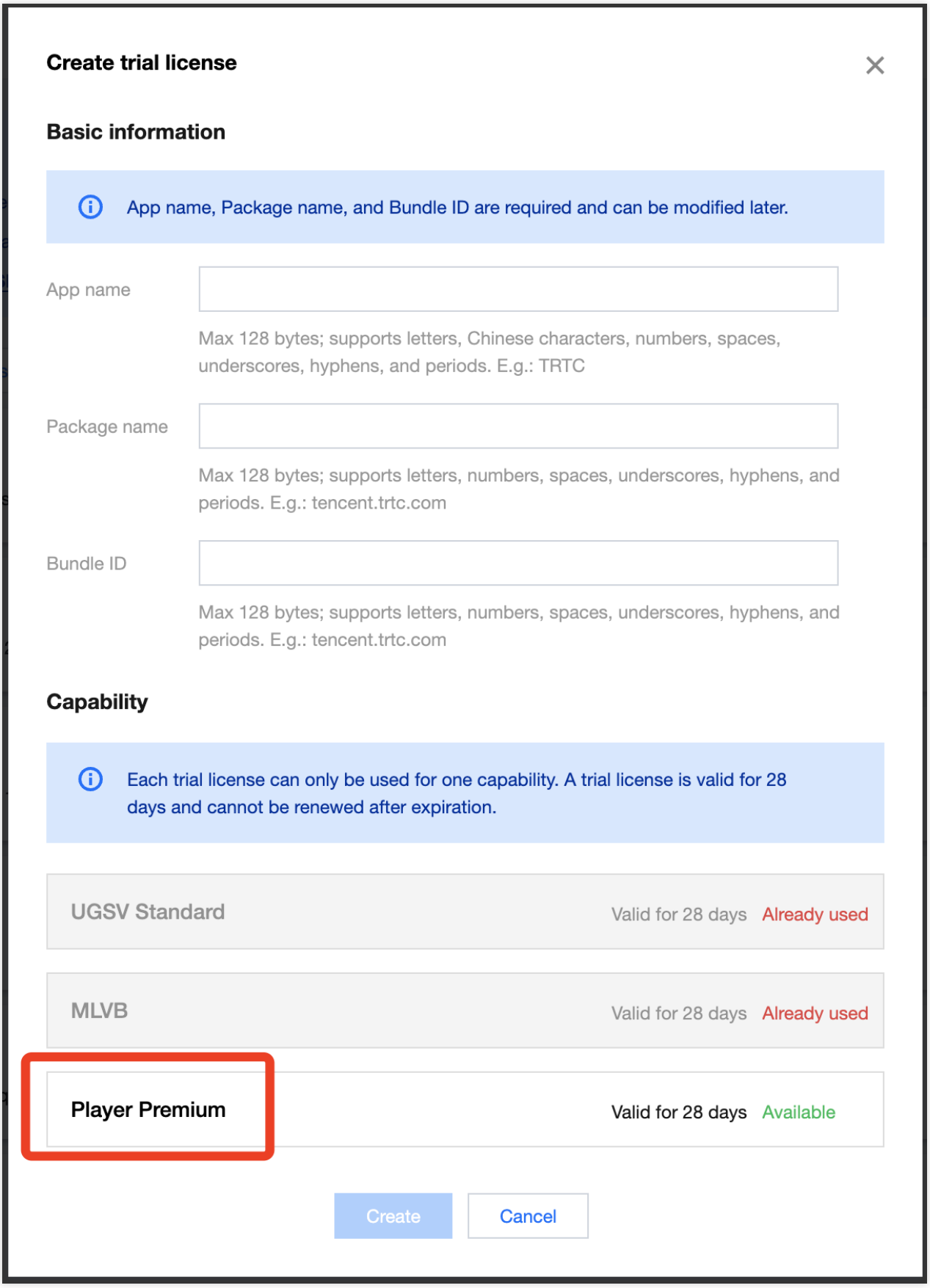


1. Log in to VOD Console or CSS Console > **License Management** > **Mobile License**, and click **Create Trial License**.



2. Enter the `App Name` , `Package Name` , and `Bundle ID` according to your actual needs, select **Player Premium**, and click **OK**.

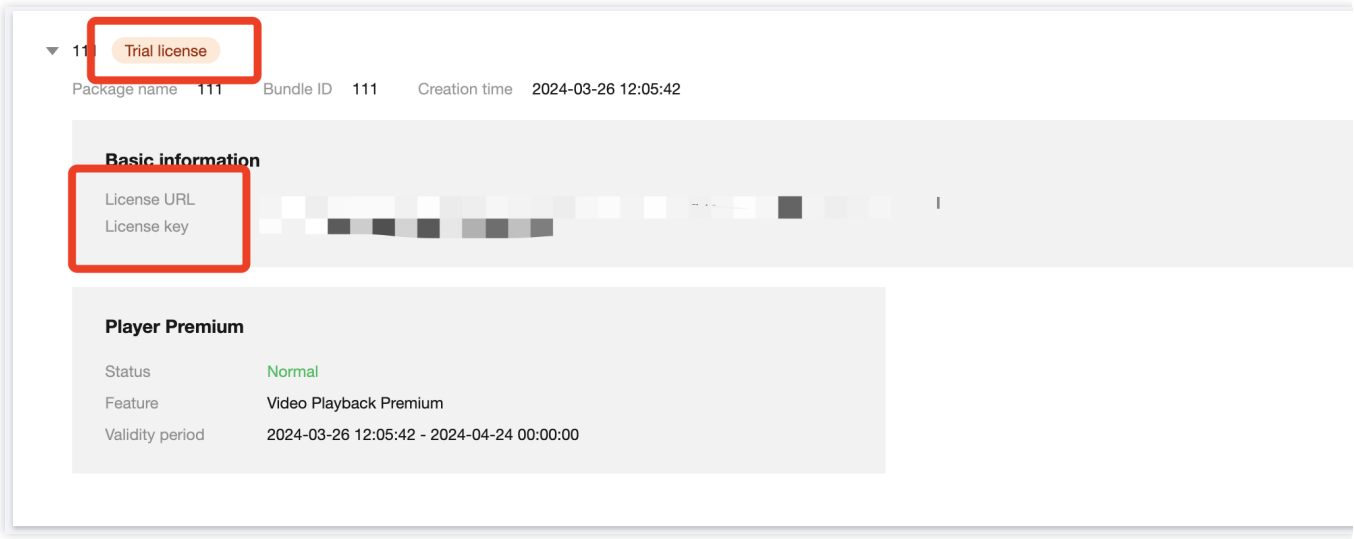
©2013-2022 Tencent Cloud. All rights reserved.



3. After the Trial License is successfully created, the page will display the generated License information. **When initializing the SDK configuration, you need to enter two parameters: License Key and License URL, so**

©2013-2022 Tencent Cloud. All rights reserved.

carefully save the following information.



Note:

The License URL and Key for the same application are unique; after the Trial License is upgraded to the official version, the License URL and Key remain unchanged.

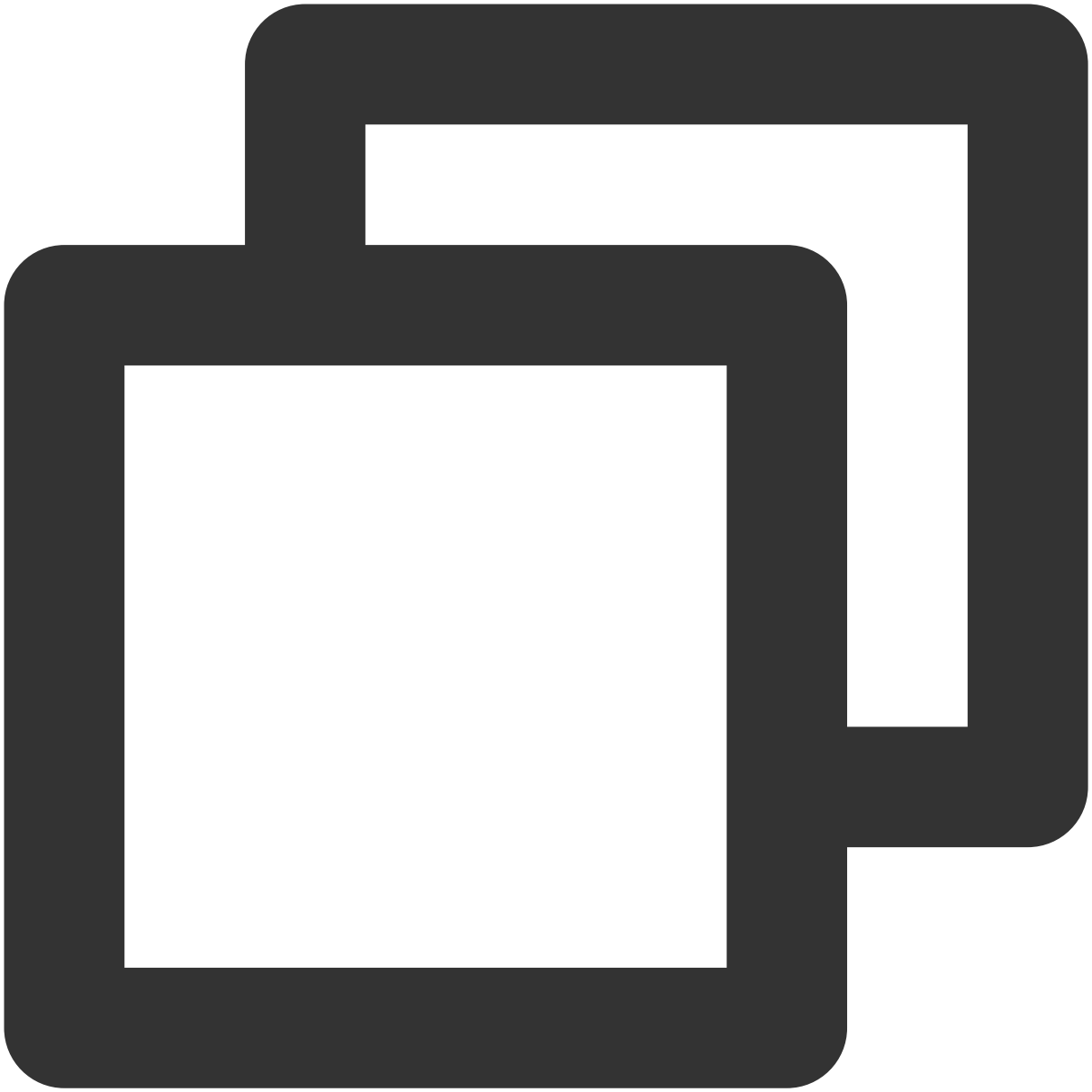
Step 2: import SDK

TRTC SDK and Special Effect SDK have been released to the **CocoaPods** repository, and you can integrate them via CocoaPods.

1. Install CocoaPods

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

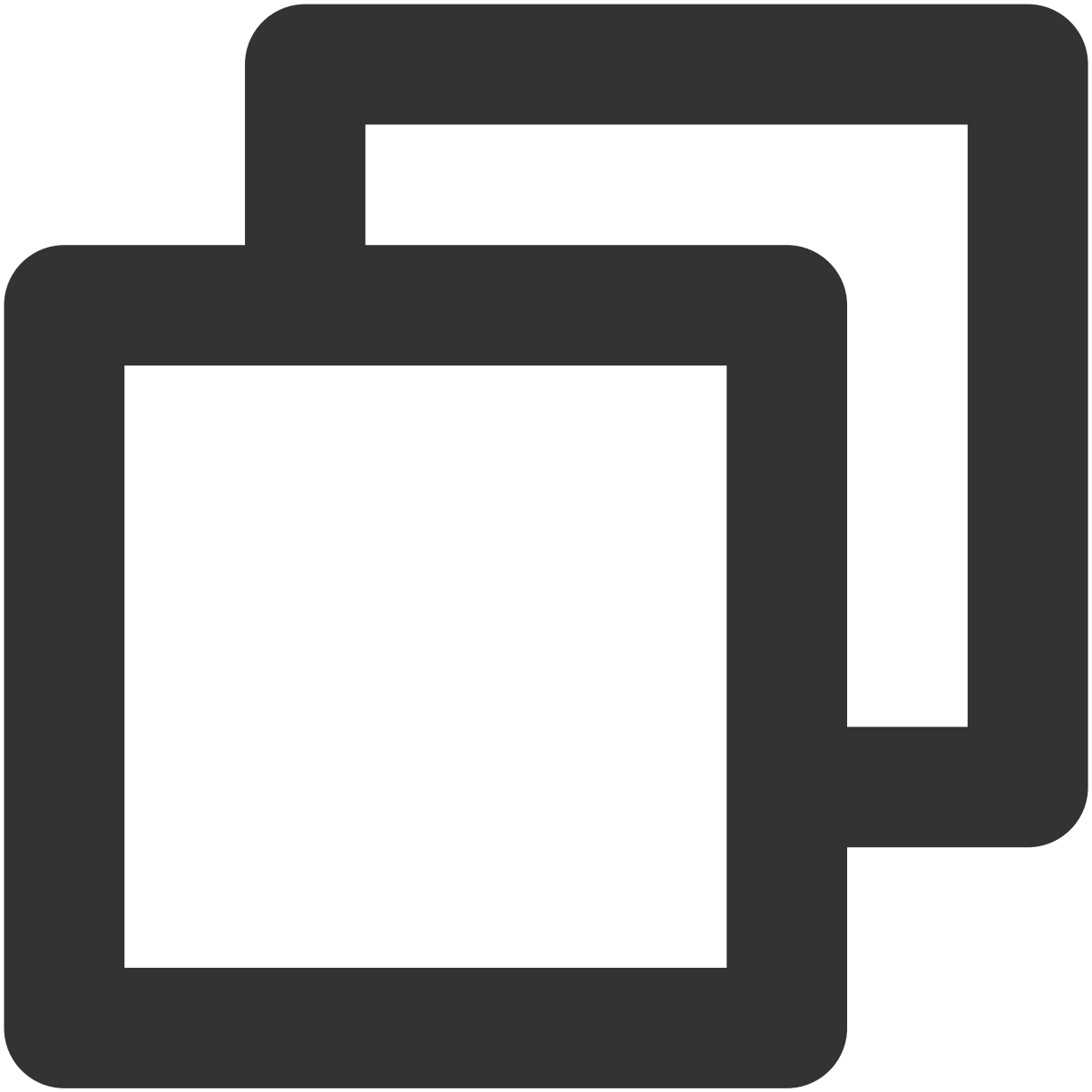


```
sudo gem install cocoapods
```

2. Create Podfile File

Go to the project directory, and enter the following command. A Podfile file will then be created in the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

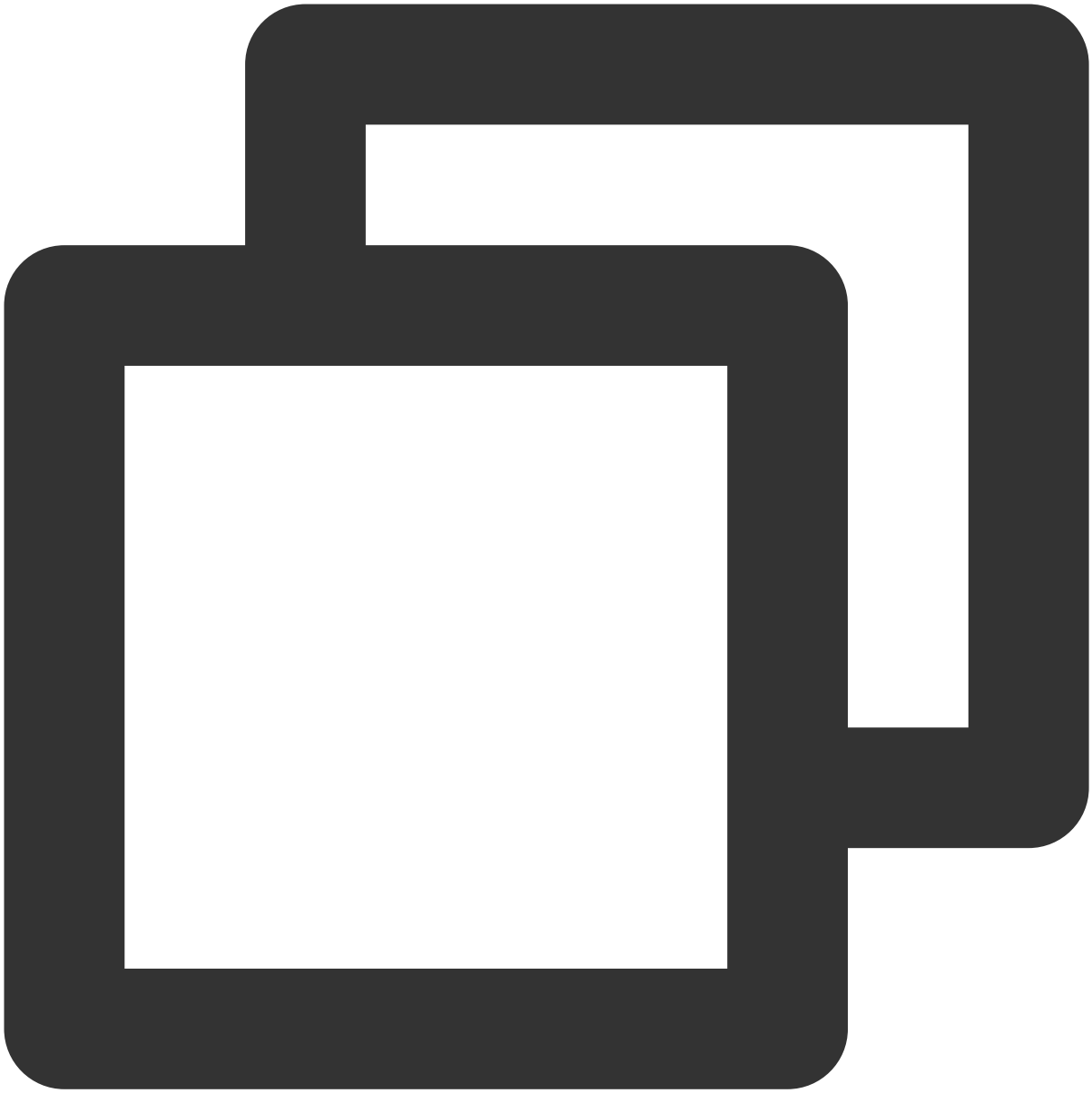


```
pod init
```

3. Edit Podfile File

Choose an appropriate version for your project and edit the Podfile file:

©2013-2022 Tencent Cloud. All rights reserved.



```
platform :ios, '8.0'
    target 'App' do

    # The full feature version of SDK
    # Includes a wide range of features such as Real-Time Communication (TRTC), TXL
    pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/pod/

    # Special Effect SDK example of S1-07 package is as follows
    pod 'TencentEffect_S1-07'

end
```

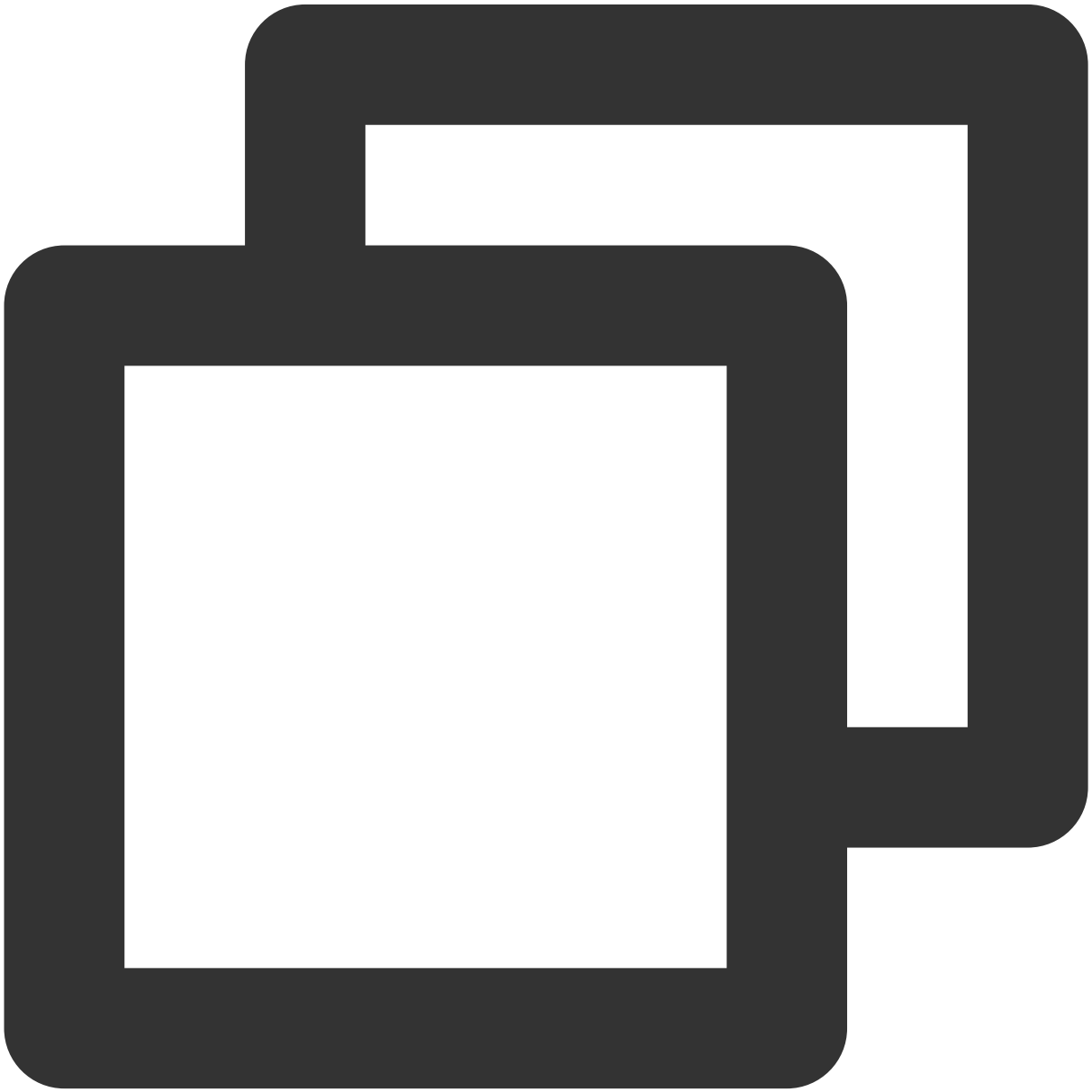
©2013-2022 Tencent Cloud. All rights reserved.

Note:

The implementation of e-commerce live streaming scenarios usually depends on the combination of several capabilities such as TRTC and players. **To avoid the symbol conflict issue that arises from single integration, it is recommended to integrate the full feature version of the SDK.**

4. Update and install the SDK

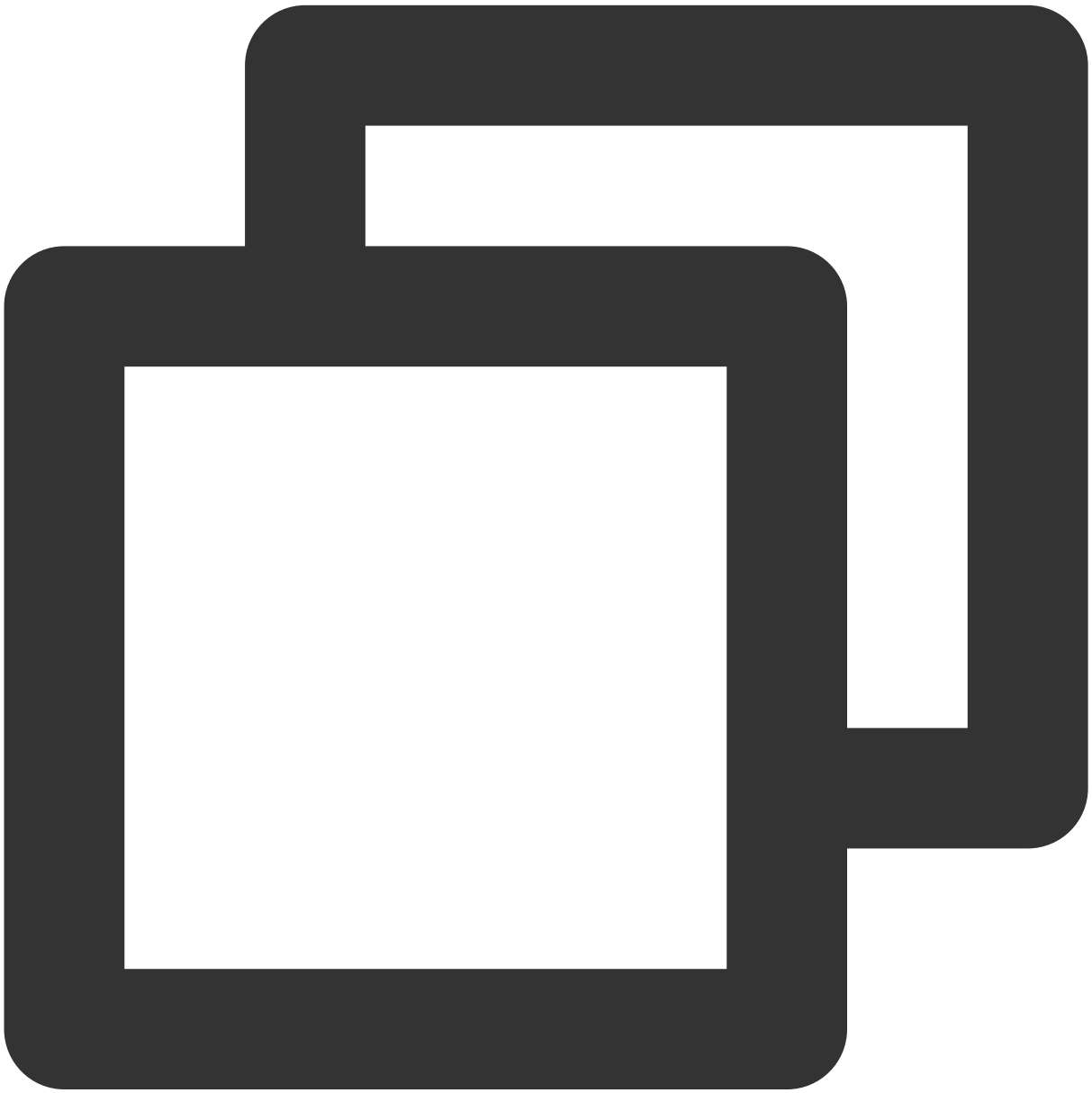
Enter the following command in a terminal window to update the local repository files and install the SDK:



```
pod install
```

©2013-2022 Tencent Cloud. All rights reserved.

Or run this command to update the local repository:



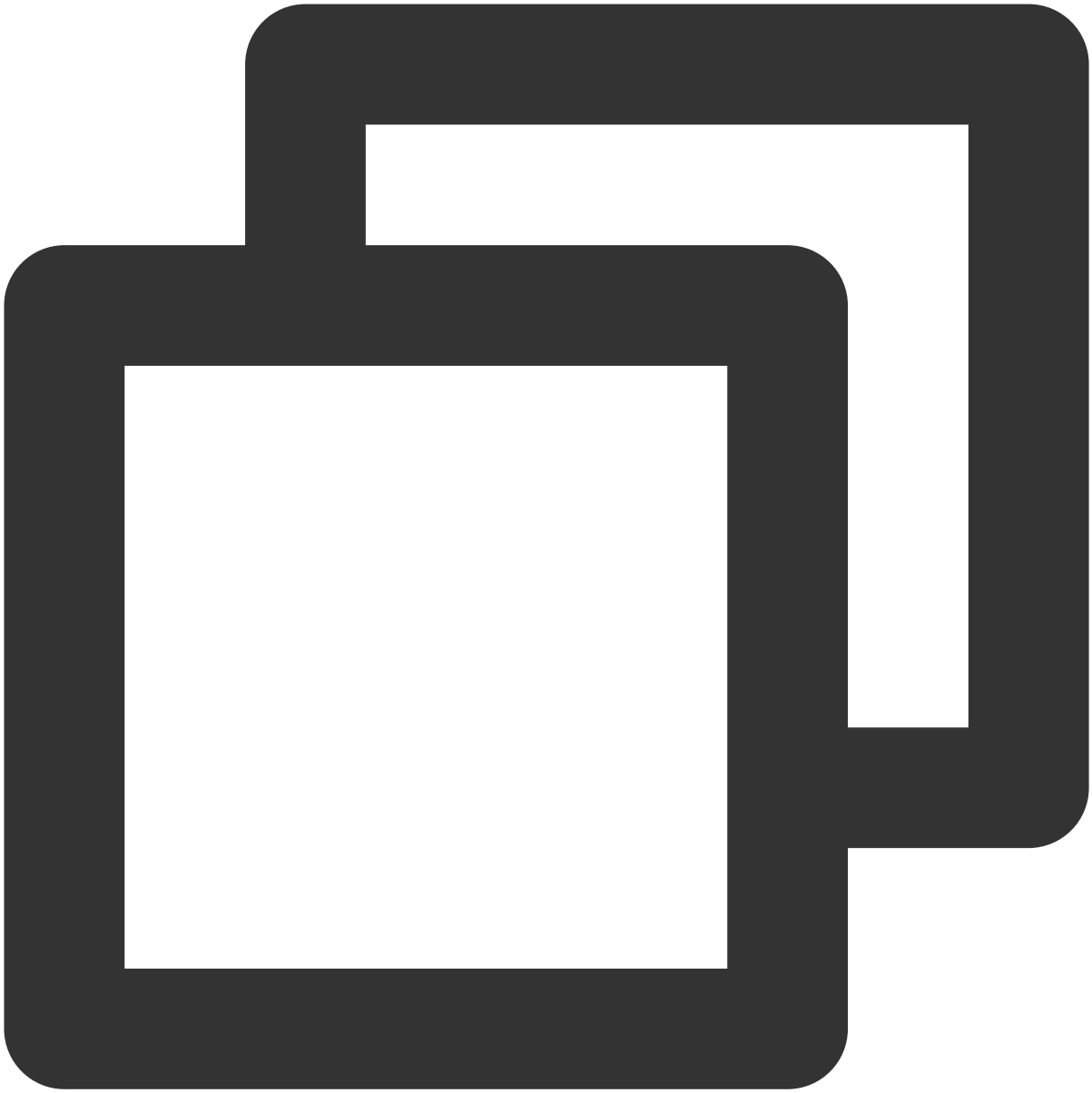
```
pod update
```

Upon the completion of pod command execution, a project file suffixed with .xcworkspace and integrated with the SDK will be generated. Double-click to open it.

Note:

If the pod search fails, it is recommended to try updating the local repo cache of pod. The update command is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
pod setup
pod repo update
rm ~/Library/Caches/CocoaPods/search_index.json
```

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrate the TRTC SDK and Manually Integrate Special Effect SDK.

5. Add beauty resources to the actual project

Download and unzip the corresponding package of SDK and Beauty Resources. Add the bundle resources under the resources/motionRes folder to the actual project.

©2013-2022 Tencent Cloud. All rights reserved.

Add `-ObjC` in Other Linker Flags of Build Settings.

6. Modify the Bundle Identifier to match the applied trial authorization.

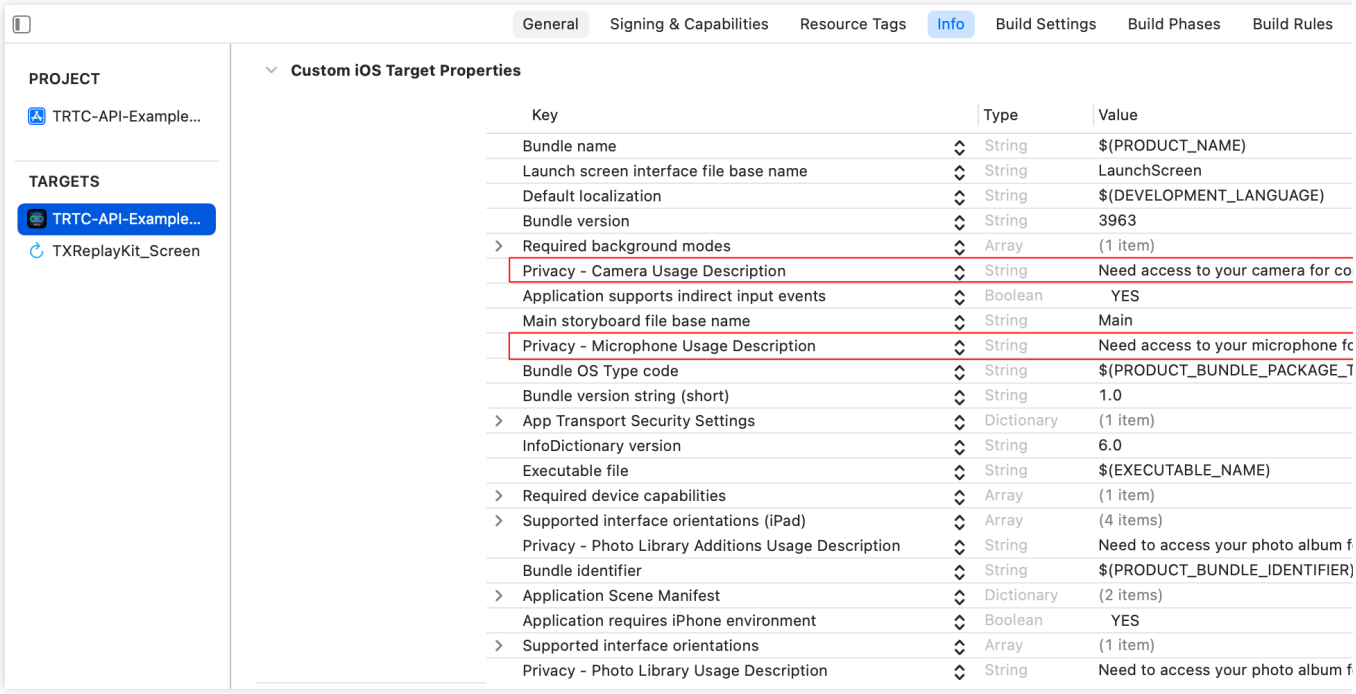
Step 3: project configuration

1. Configure permissions

For e-commerce live streaming scenarios, LiteAVSDK and Special Effect SDK require the following permissions. Add the following two items to the App's Info.plist, corresponding to the microphone and camera prompts in the system pop-up authorization dialog box.

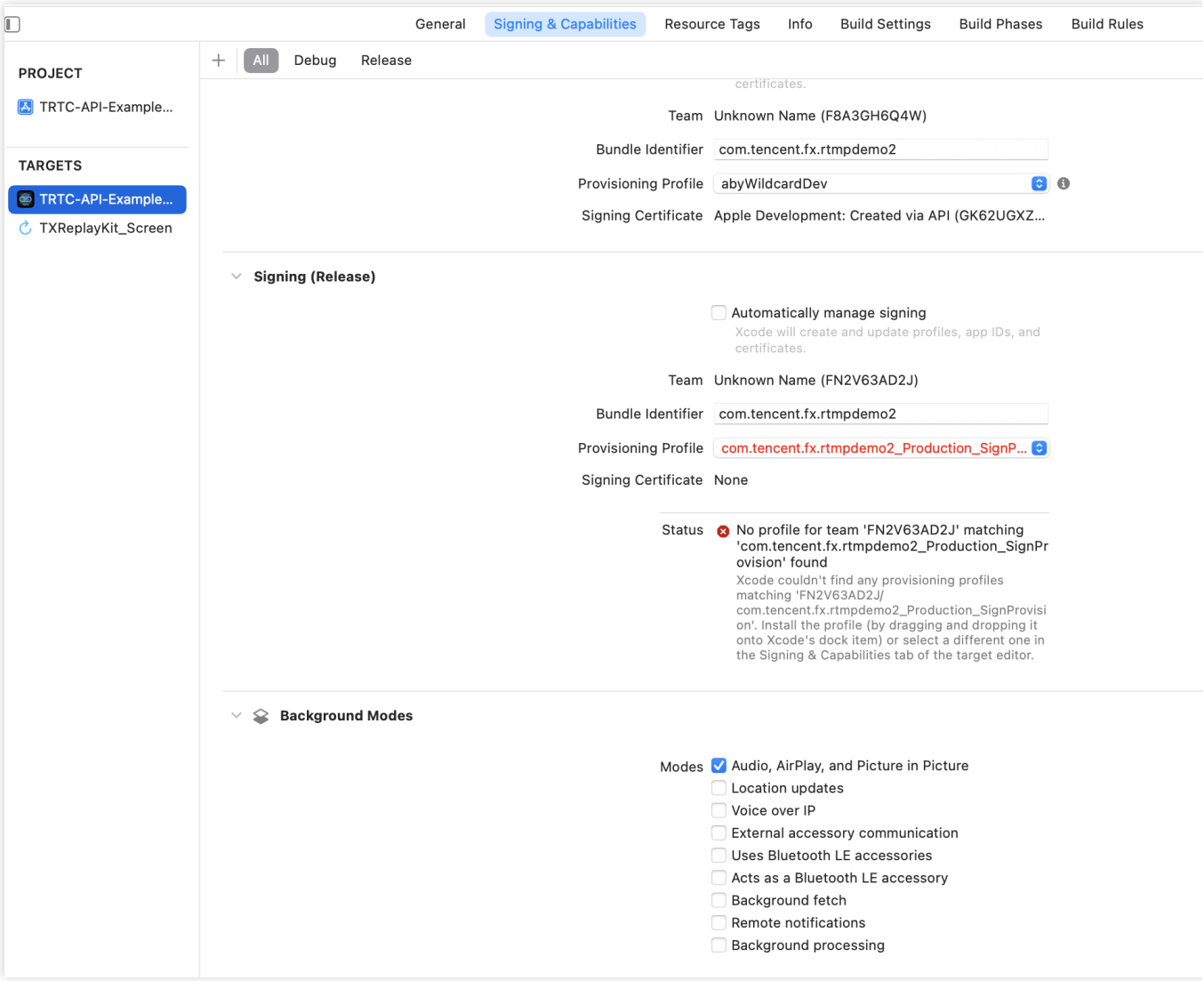
Privacy - Microphone Usage Description. Enter a prompt specifying the purpose of microphone use.

Privacy - Camera Usage Description. Enter a prompt specifying the purpose of camera use.



2. If you need your App to continue running certain features in the background, go to XCode, select your current project, and under Capabilities, set the setting for Background Modes to ON, and check Audio, AirPlay, and Picture in Picture, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Step 4: authentication and authorization

TRTC Authentication Credential

Special Effect Authentication License

Player Authentication License

UserSig is a security protection signature designed by the cloud platform to prevent malicious attackers from misappropriating your cloud service usage rights. TRTC validates this authentication credential when entering a room.

Debugging and testing stage: UserSig can be generated through Client Sample Code and Console Access, which are only used for debugging and testing.

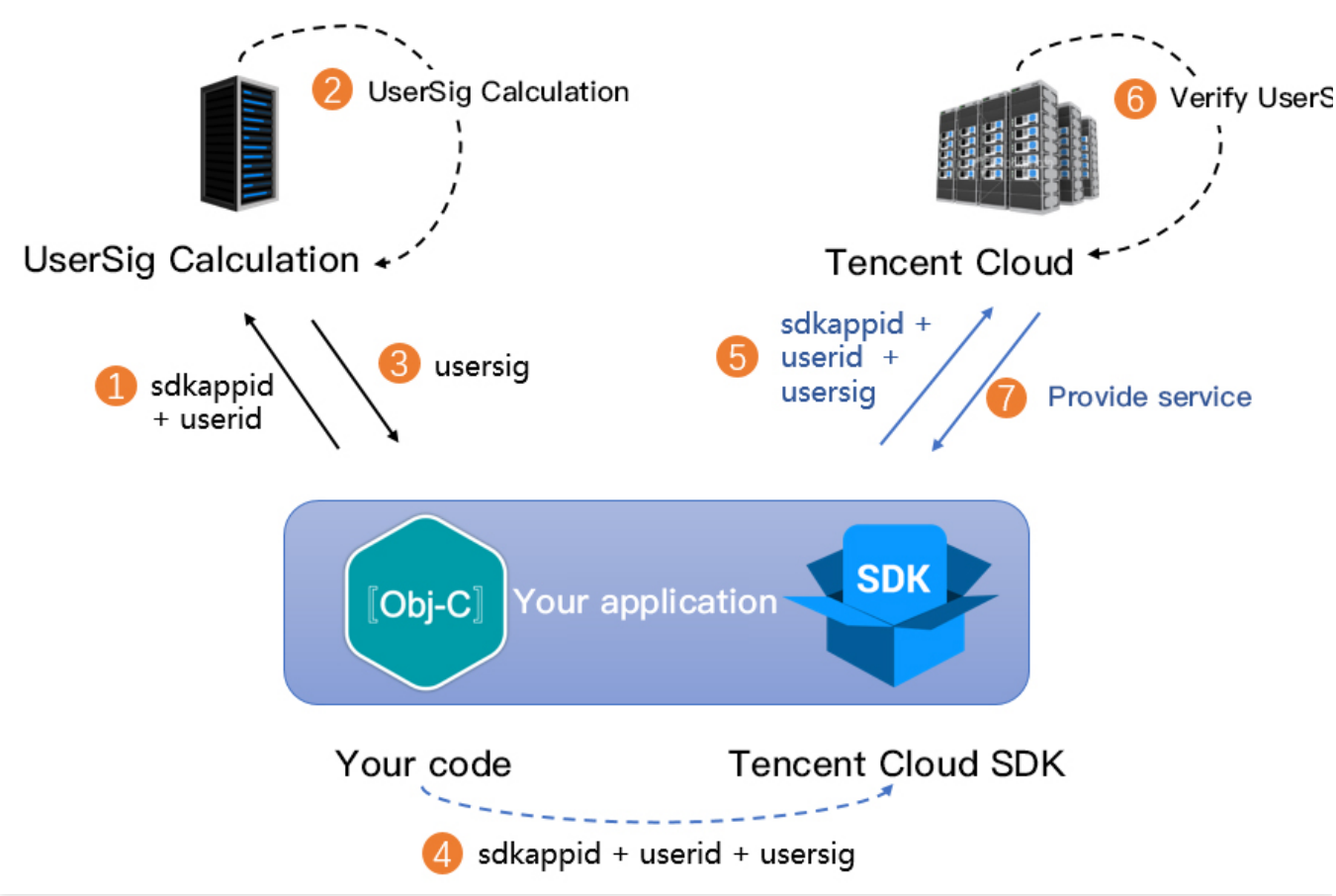
Production stage: It is recommended to use the server computing UserSig solution, which has a higher security level and helps prevent the client from being decompiled and reversed, to avoid the risk of key leakage.

The specific implementation process is as follows:

1. Before calling the initialization API of the SDK, your app must first request UserSig from your server.
2. Your server generates the UserSig based on the SDKAppID and UserID.
3. The server returns the generated UserSig to your app.

©2013-2022 Tencent Cloud. All rights reserved.

4. Your app sends the obtained UserSig to the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to the cloud server for verification.
6. The cloud platform verifies the validity of the UserSig.
7. After the verification is passed, real-time audio and video services will be provided to the TRTC SDK.



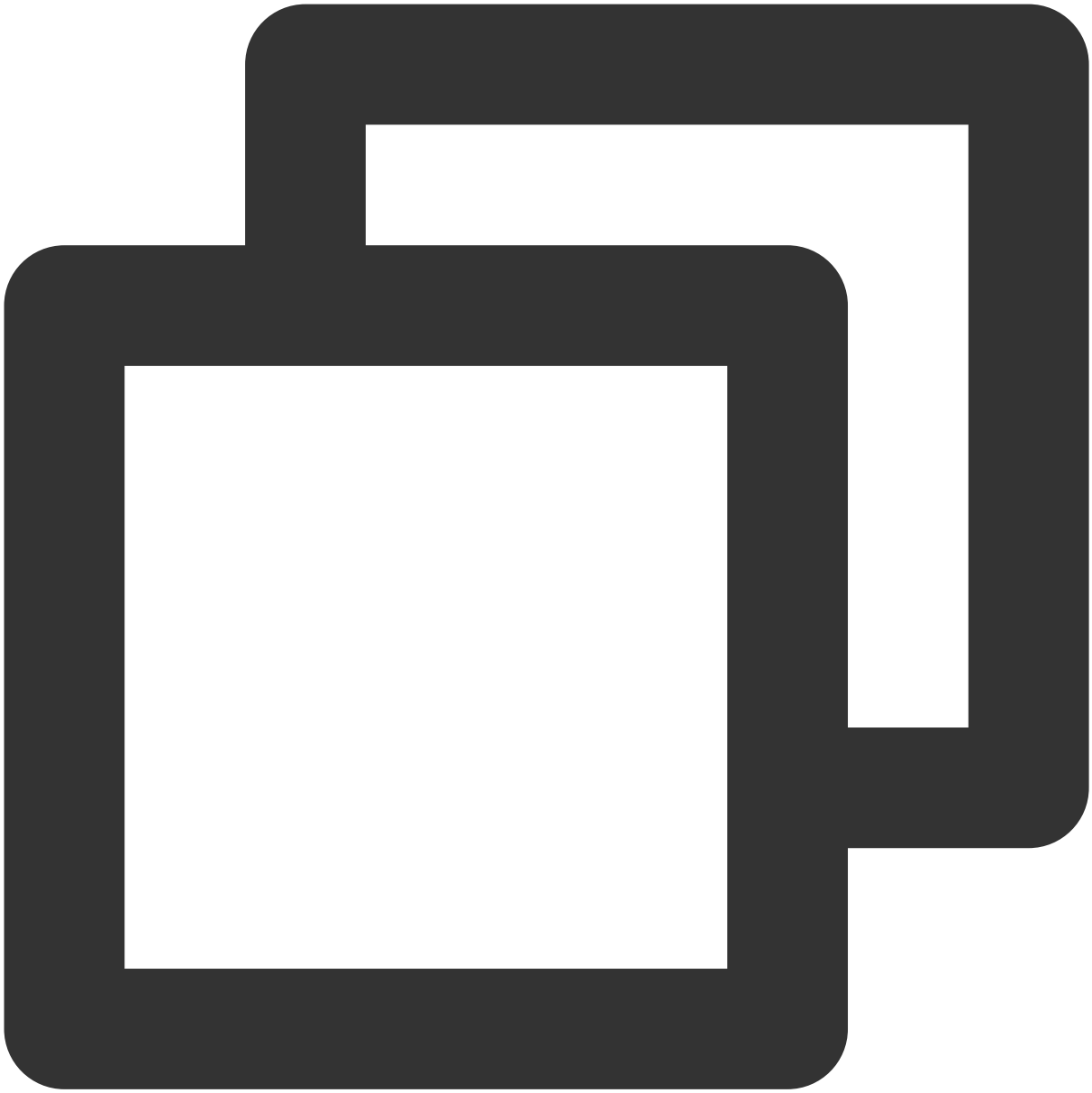
Note:

The method of generating UserSig locally during the debugging and testing stage is not recommended for the online environment because it may be easily decompiled and reversed, causing key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Before using Beauty Special Effect, you need to verify the license credential with the cloud platform. Configuring the License requires License Key and License Url. Sample code is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



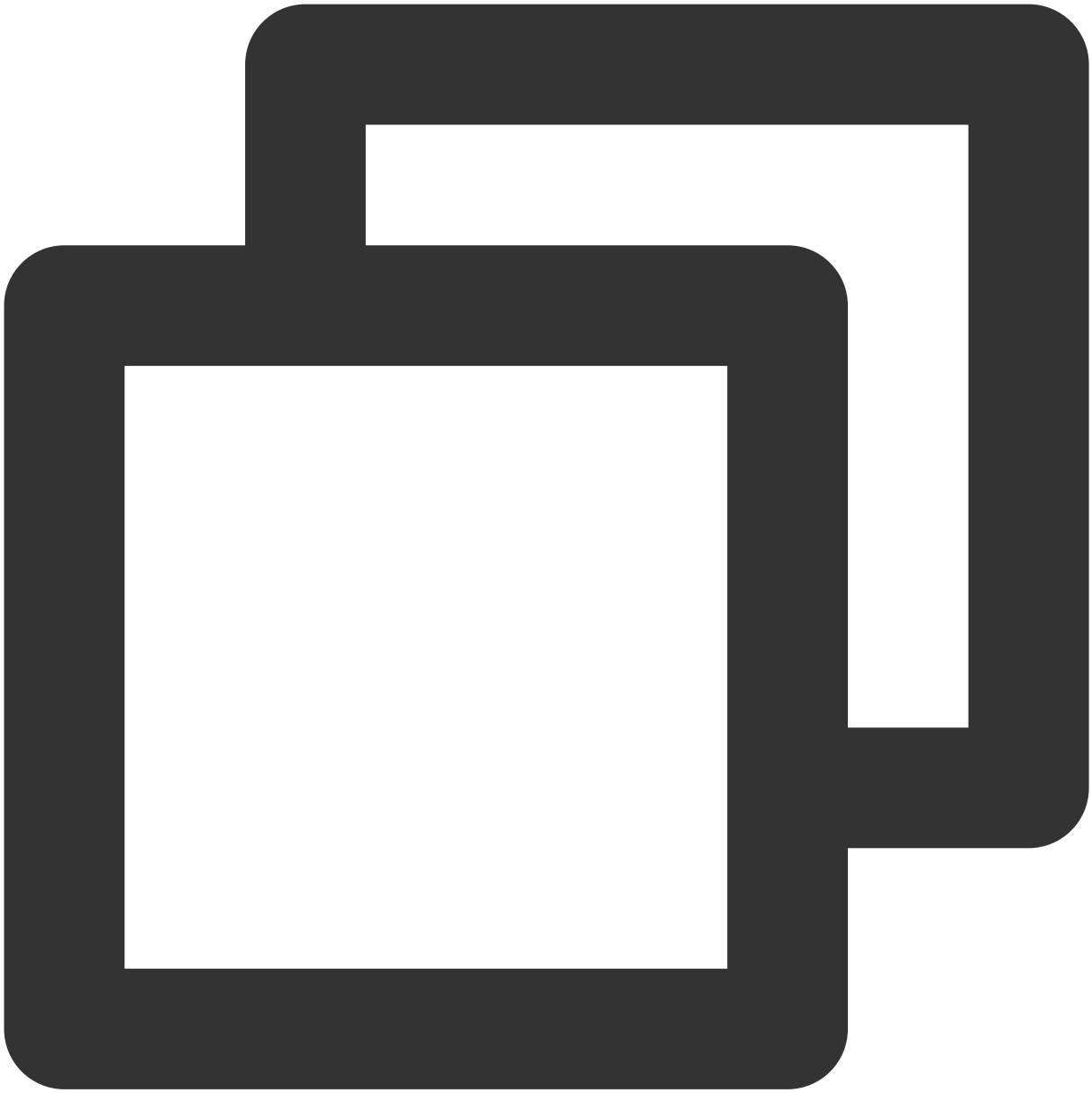
```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
 if (authresult == TELicenseCheckOk) {
     NSLog(@"Authentication successful");
 } else {
     NSLog(@"Authentication failed");
 }
}];
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

It is recommended to trigger the authentication permission in the initialization code of related business modules, to avoid having to download the License temporarily before use. Additionally, during authentication, network permissions must be ensured.
The actual application's Bundle ID must match exactly with the Bundle ID associated with the License creation. Otherwise, it will lead to License verification failure. For details, see Authentication Error Code.
The live streaming and on-demand playback features require setting the License before success in playback. Otherwise, playback will fail (black screen). It needs to be set globally only once. If you have not obtained the License, you can freely apply for a Trial Version License for normal playback. The Official Version License requires purchase.
After successfully applying for License, you will receive two strings: **License URL** and **License Key**.
Before your App calls the SDK-related features (recommended in `- [AppDelegate application:didFinishLaunchingWithOptions:]`), set the following settings:

©2013-2022 Tencent Cloud. All rights reserved.



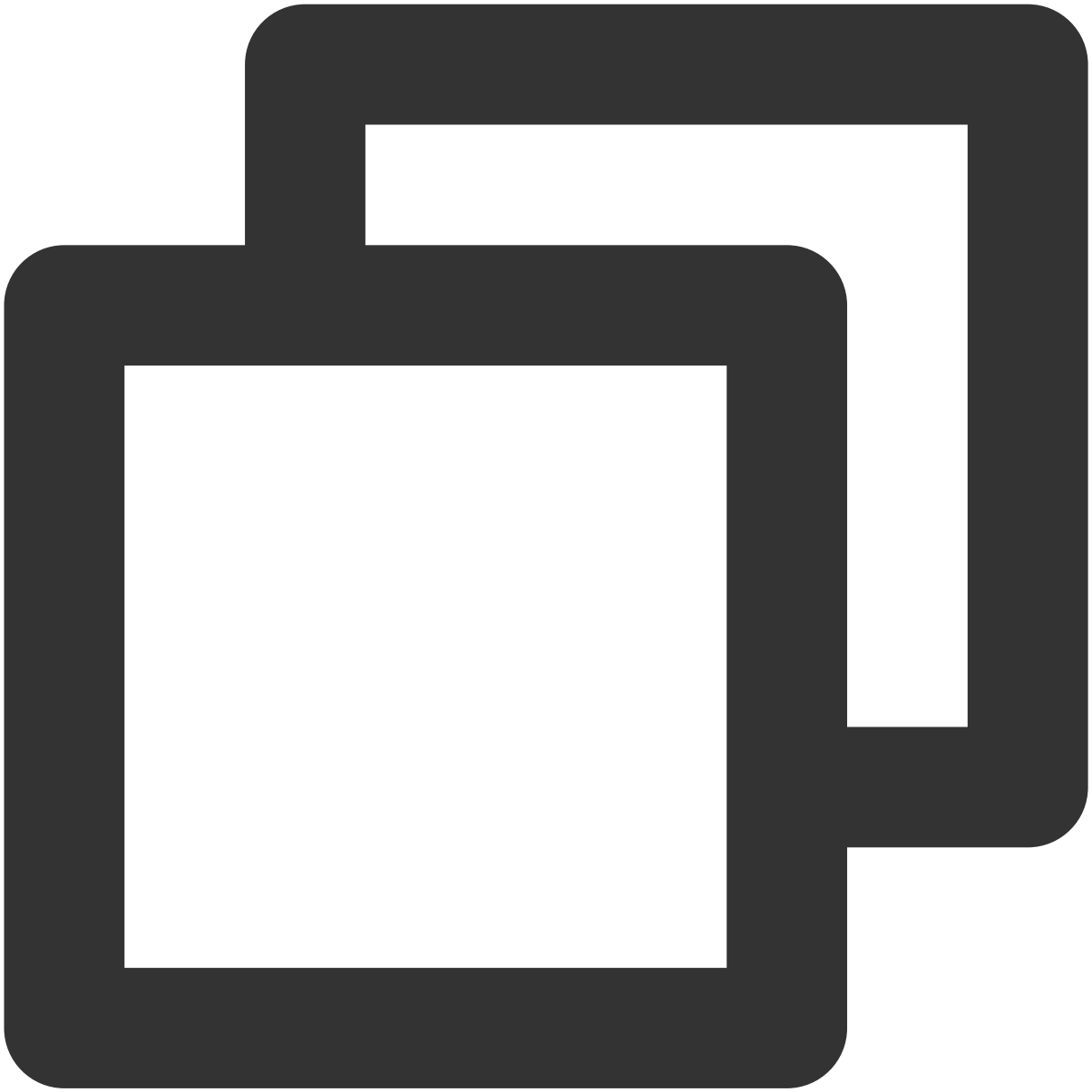
```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<the obtained licenseUrl>";
    NSString * const licenceKey = @"<the obtained key>";

    // TXLiveBase is located in the "TXLiveBase.h" header file
    [TXLiveBase setLicence:licenceURL key:licenceKey];
    [TXLiveBase setObserver:self];
    NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
    return YES;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
#pragma mark - TXLiveBaseDelegate
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
    // If the result is not 0, it means the setting has failed, and you need to ret
    if (result != 0) {
       [TXLiveBase setLicence:licenceURL key:licenceKey];
    }
}
@end
```

After the License is successfully set (you need to wait for a while, the specific time depends on the network conditions), you can use the following method to view the License information:

©2013-2022 Tencent Cloud. All rights reserved.



```
NSLog(@"%@", [TXLiveBase getLicenceInfo]);
```

Note:

The actual application's Bundle ID must match exactly with the Bundle ID associated with the License creation. Otherwise, it will lead to License verification failure.

The License is a strong online verification logic. When the TXLiveBase#setLicence is called after the application is started for the first time, the network must be available. At the first launch of the App, if the network permission is not yet authorized, you need to wait until the permission is granted before calling TXLiveBase#setLicence again.

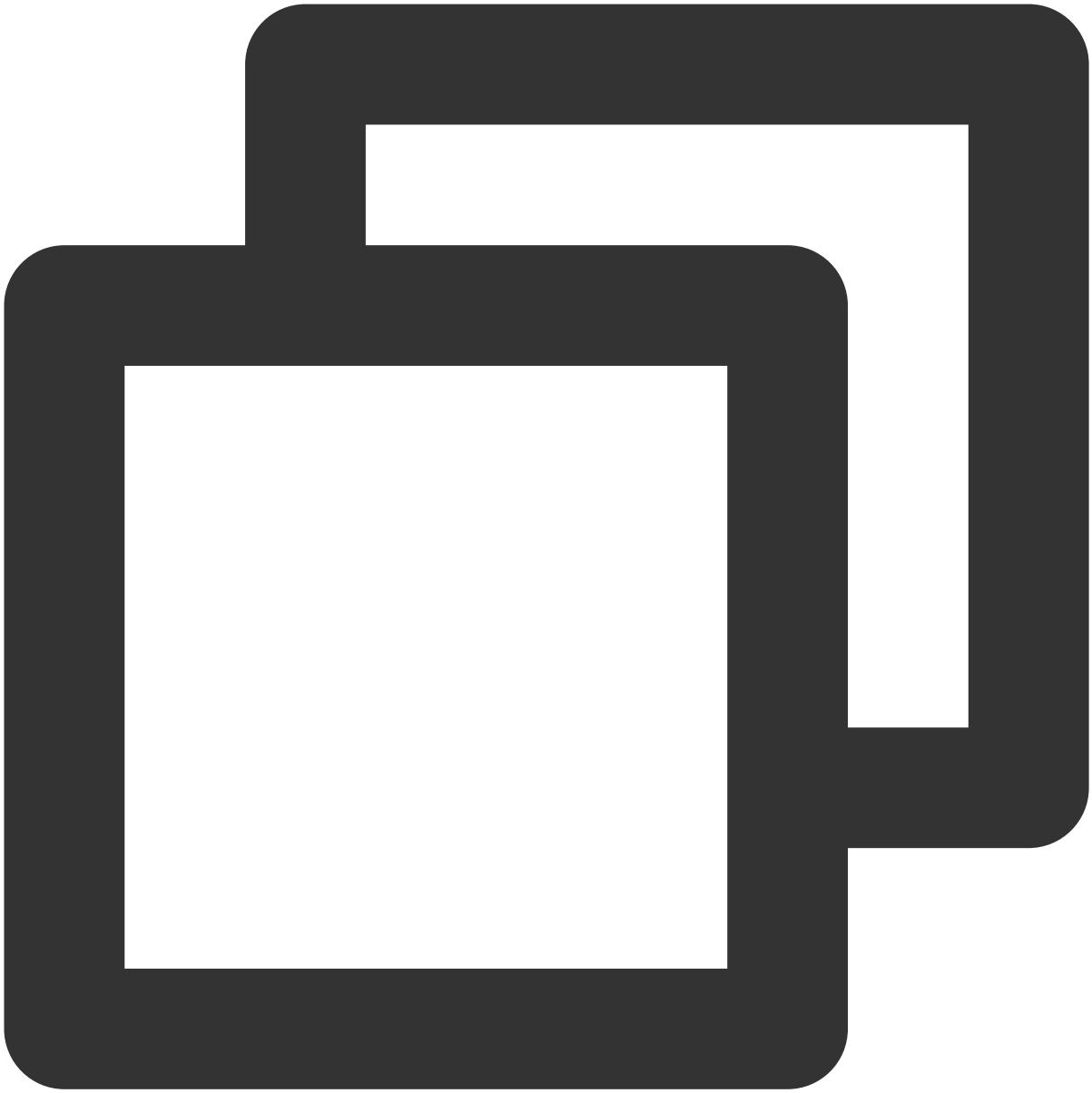
©2013-2022 Tencent Cloud. All rights reserved.

Listen to the loading result of TXLiveBase#setLicence: For onLicenceLoaded API, if it fails, you should retry and guide according to the actual situation. If it fails multiple times, you can limit the frequency and supplement with product pop-ups and other guides to allow users to check the network conditions.
TXLiveBase#setLicence can be called multiple times. It is recommended to call TXLiveBase#setLicence when entering the main interface of the App to ensure successful loading.
For multi-process Apps, ensure that every process using the player calls TXLiveBase#setLicence when it starts. For example, for Apps on the Android side that use a separate process for video playback, when the process is killed and restarted by the system during background playback, TXLiveBase#setLicence should also be called.

Step 5: initialize the SDK

Initialize the TRTC SDK
Initialize the Special Effect SDK
Initialize Player SDK

©2013-2022 Tencent Cloud. All rights reserved.



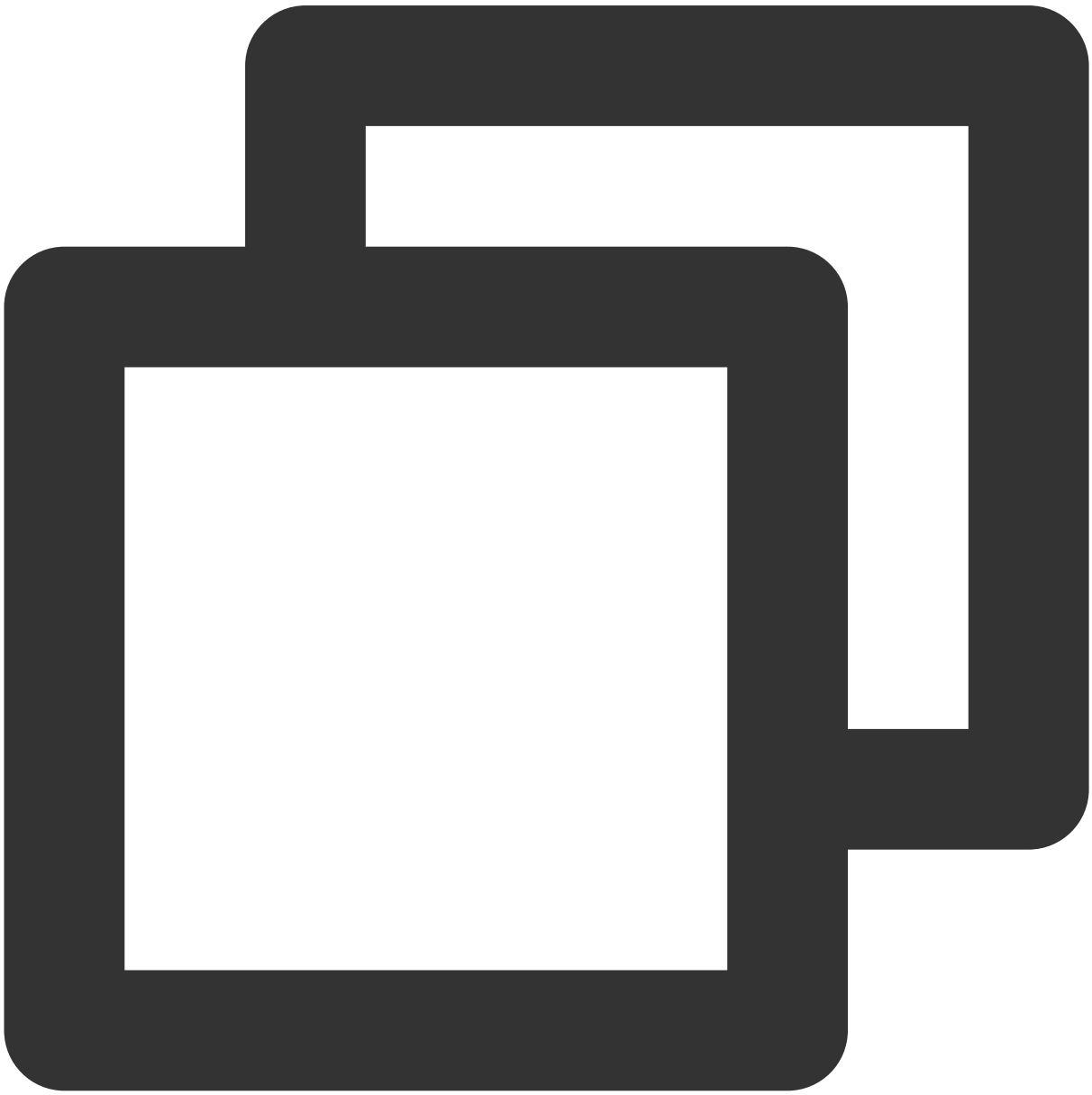
```
// Create TRTC SDK instance (single instance pattern)
self.trtcCloud = [TRTCCloud sharedInstance];
// Set event listeners
self.trtcCloud.delegate = self;

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
- (void)onError:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg extInfo:(
    NSLog(@"%d: %@", errCode, errMsg);
}

- (void)onWarning:(TXLiteAVWarning)warningCode warningMsg:(nullable NSString *)warn
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    NSLog(@"%d: %@", warningCode, warningMsg);
}

// Remove event listener
self.trtcCloud.delegate = nil;
// Destroy TRTC SDK instance (single instance pattern)
[TRTCCloud destroySharedIntance];
```

Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.



©2013-2022 Tencent Cloud. All rights reserved.

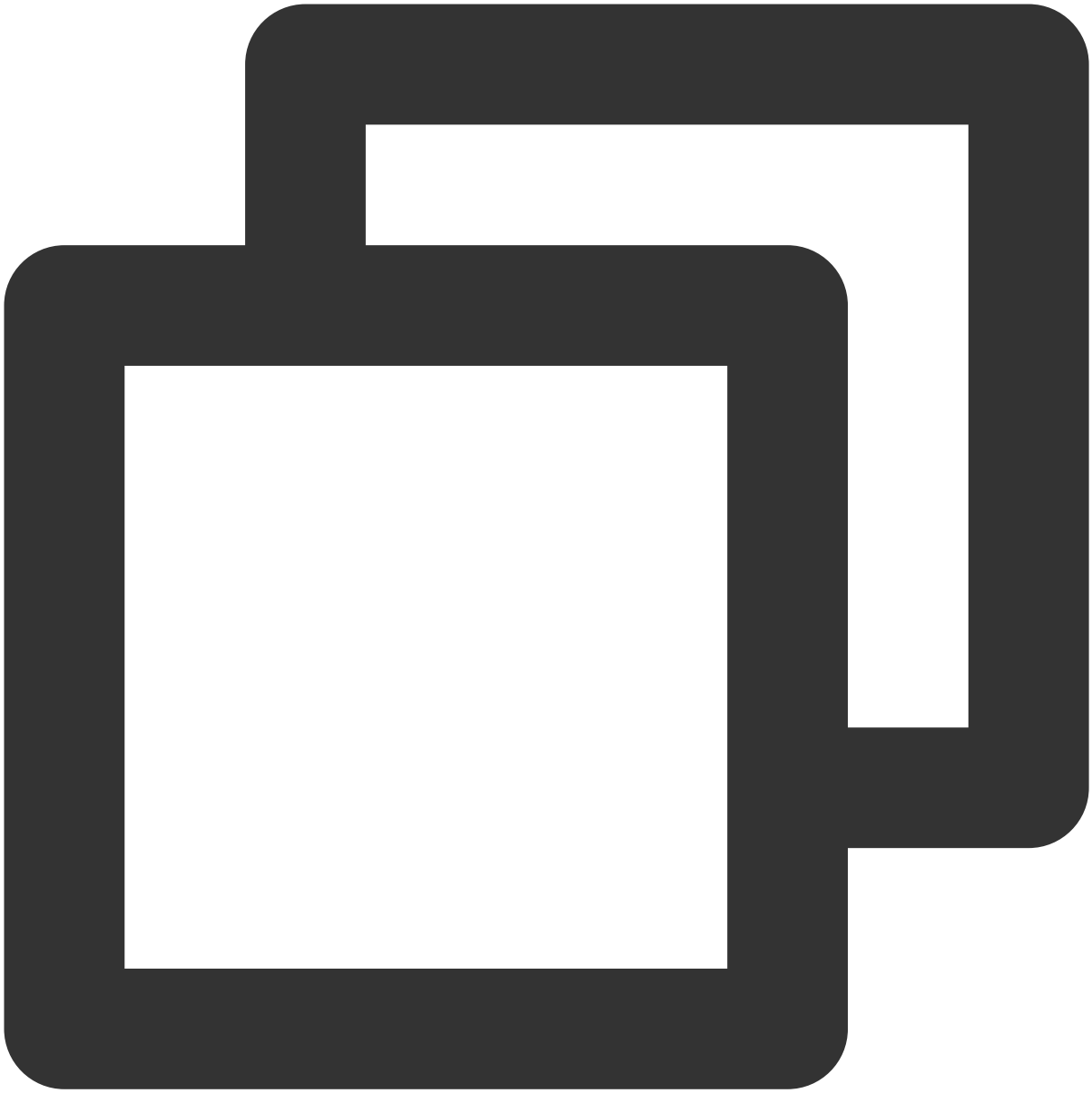
```
// Load beauty-related resources
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
 @"root_path":[[NSBundle mainBundle] bundlePath]
};

// Initialize the Special Effect SDK
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi

// Release the Special Effect SDK
[self.beautyKit deinit];
```

Note:

Before initializing the Special Effect SDK, resource copying and other preparatory work are needed. For detailed steps, see Special Effect SDK Integration Steps.

On-demand Playback Scenario SDK Initialization.

©2013-2022 Tencent Cloud. All rights reserved.



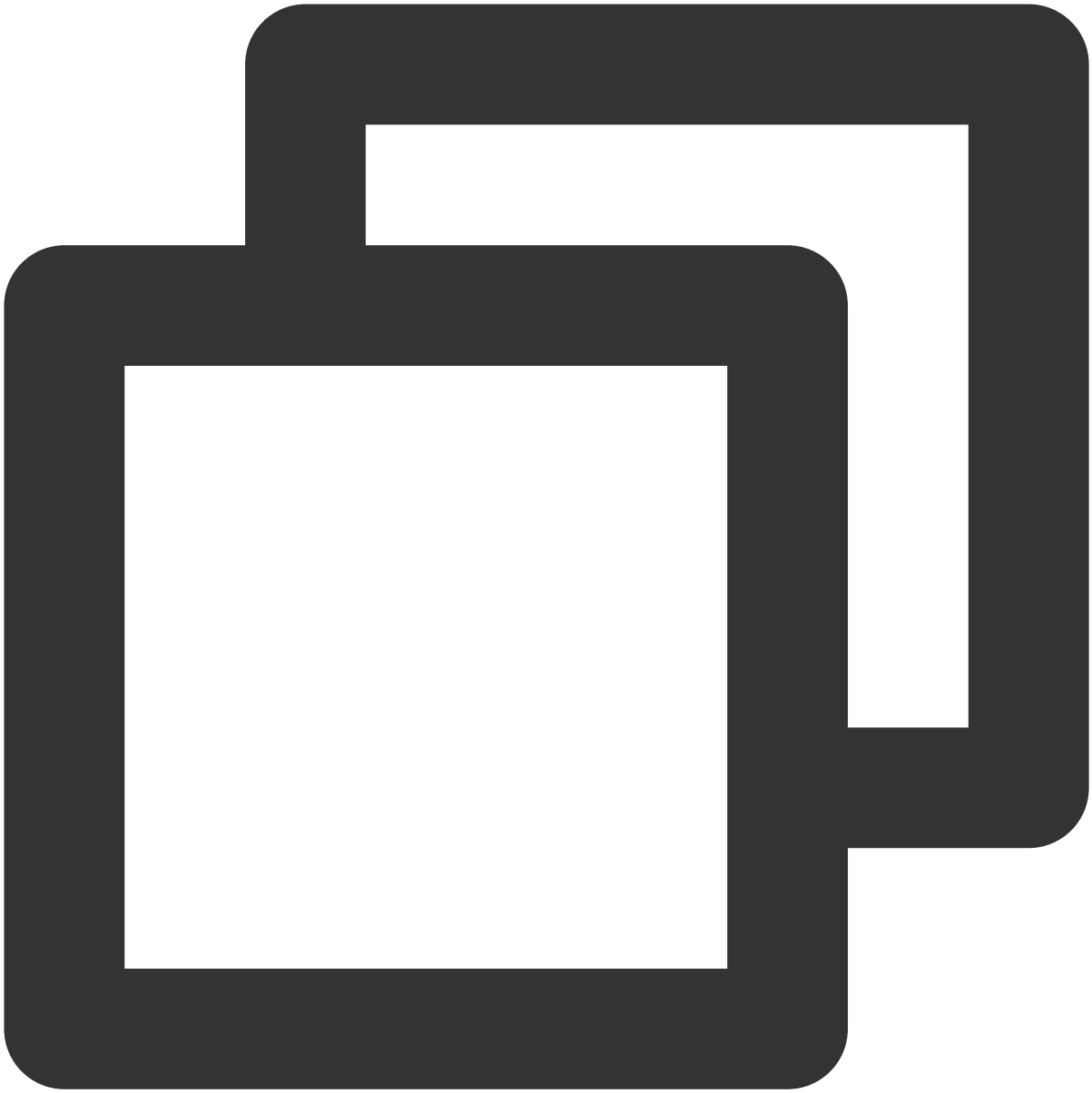
```
// 1. Set the SDK Connect Environment
// If you serve global users, configure the SDK connect environment for global conn
[TXLiveBase setGlobalEnv:"GDPR"];

// 2. Create Player
TXVodPlayer *_txVodPlayer = [[TXVodPlayer alloc] init];

// 3. Associate Rendering View
[_txVodPlayer setupVideoWidget:_myView insertIndex:0];

// 4. Player Parameter Configuration
```

©2013-2022 Tencent Cloud. All rights reserved.

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setEnableAccurateSeek:true];// Set whether to seek accurately. The default
[_config setMaxCacheItems:5];            // Set the number of cache files to 5
[_config setProgressInterval:200];   // Set the interval for progress callbacks, in
[_config setMaxBufferSize:50];   // The maximum pre-load size, in MB
[_txVodPlayer setConfig:_config];        // Pass config to _txVodPlayer

// 5. Player Event Listener
- (void)onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*
// Received event that the player is ready, now you can call pause, resume, getWidt
// Received the start playback event    } else if (EvtID == PLAY_EVT_PLAY_END) {
// Received the playback end event
    }
}
```

Live Streaming Scenarios SDK initialization.

©2013-2022 Tencent Cloud. All rights reserved.



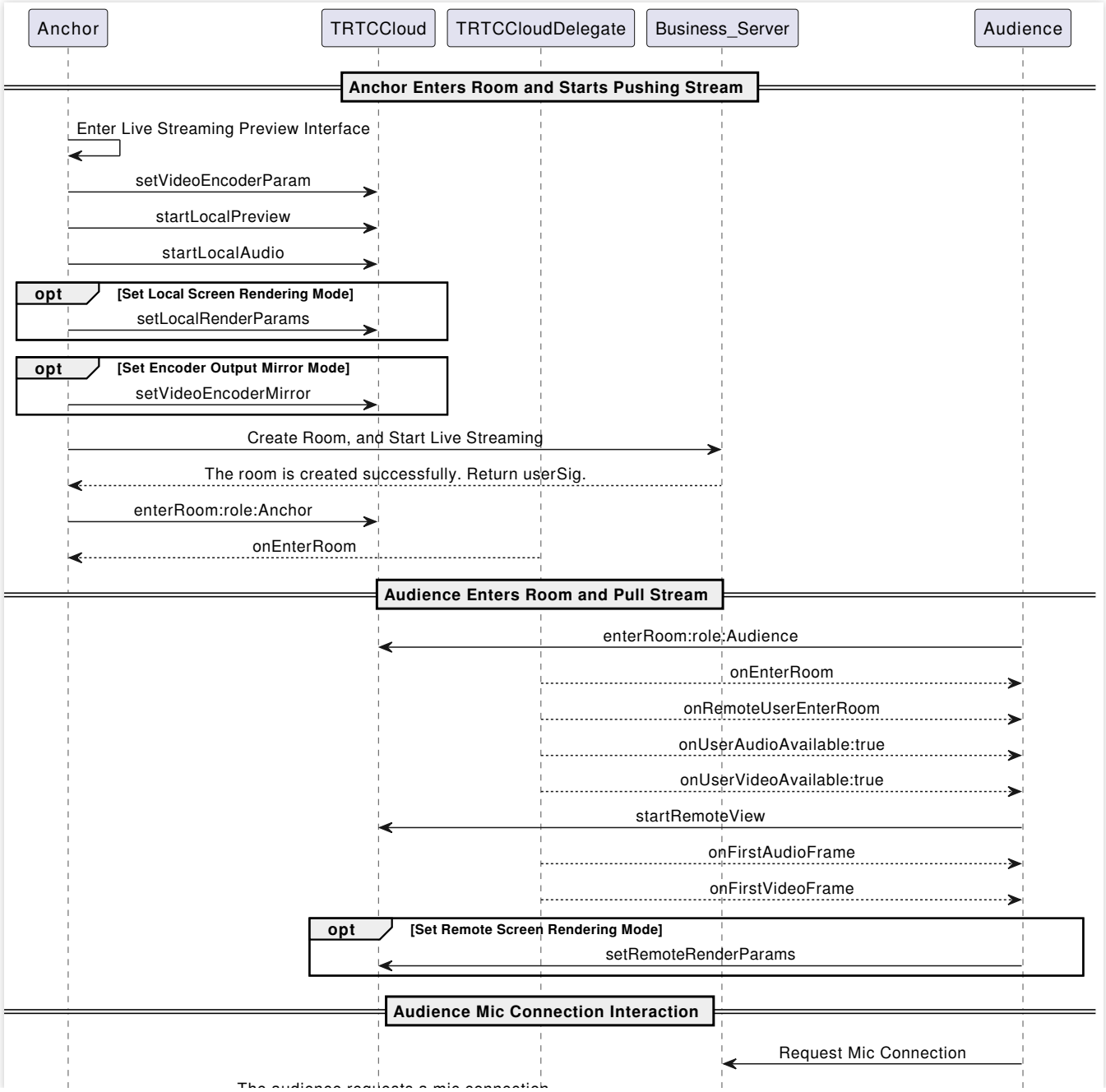
```
// 1. Create Player
V2TXLivePlayer *_txLivePlayer = [[V2TXLivePlayer alloc] init];

// 2. Associate Rendering View
[_txLivePlayer setRenderView:_myView];

// 3. Player Event Listener
[_txLivePlayer setObserver:self];

- (void)onVideoLoading:(id<V2TXLivePlayer>)player extraInfo:(NSDictionary *)extraIn
    // Video loading event
```

©2013-2022 Tencent Cloud. All rights reserved.

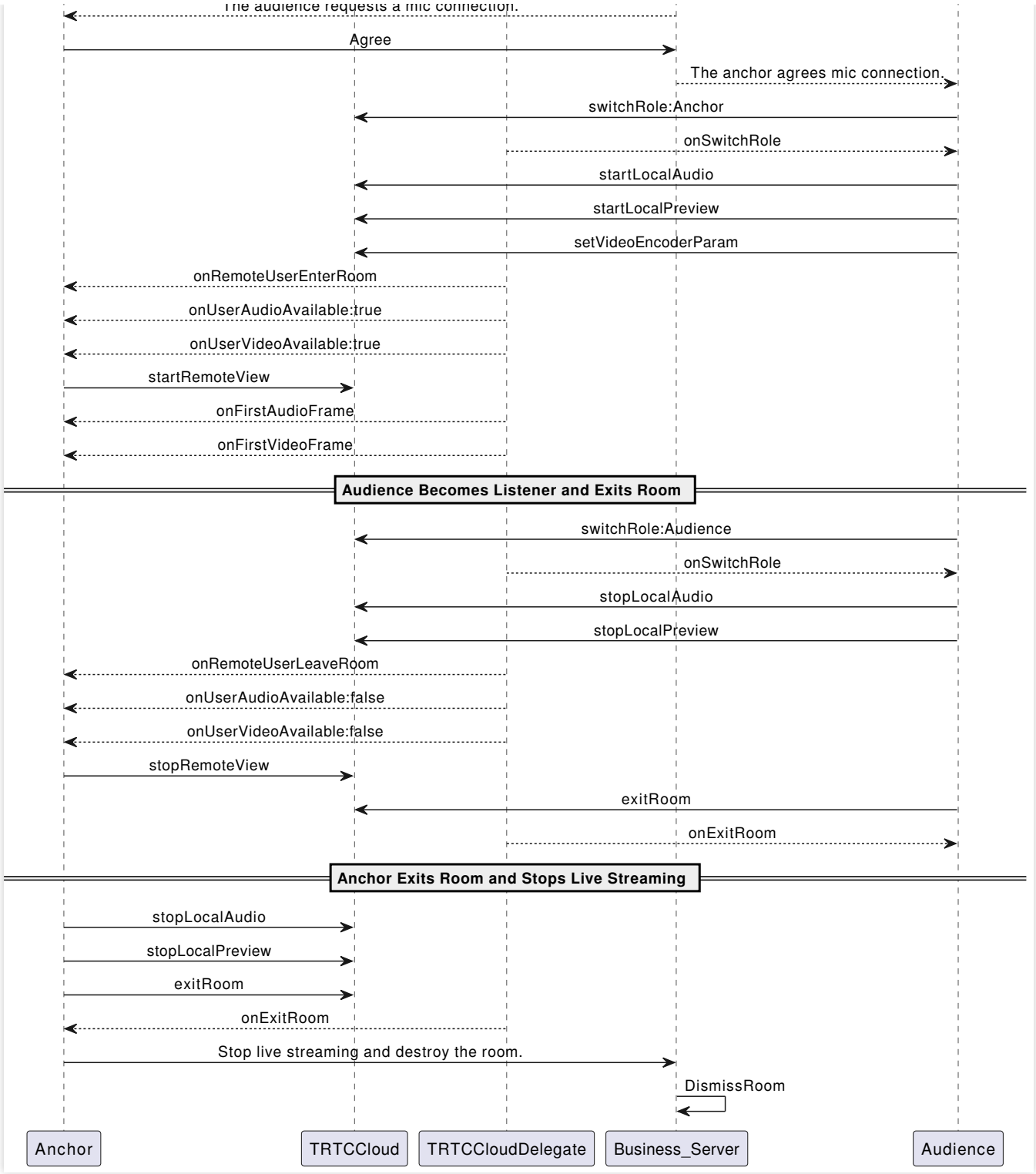
```
}

- (void)onVideoPlaying:(id<V2TXLivePlayer>)player firstPlay:(BOOL)firstPlay extraIn
    // Video playback event
}
```

Integration Process

API Sequence Diagram



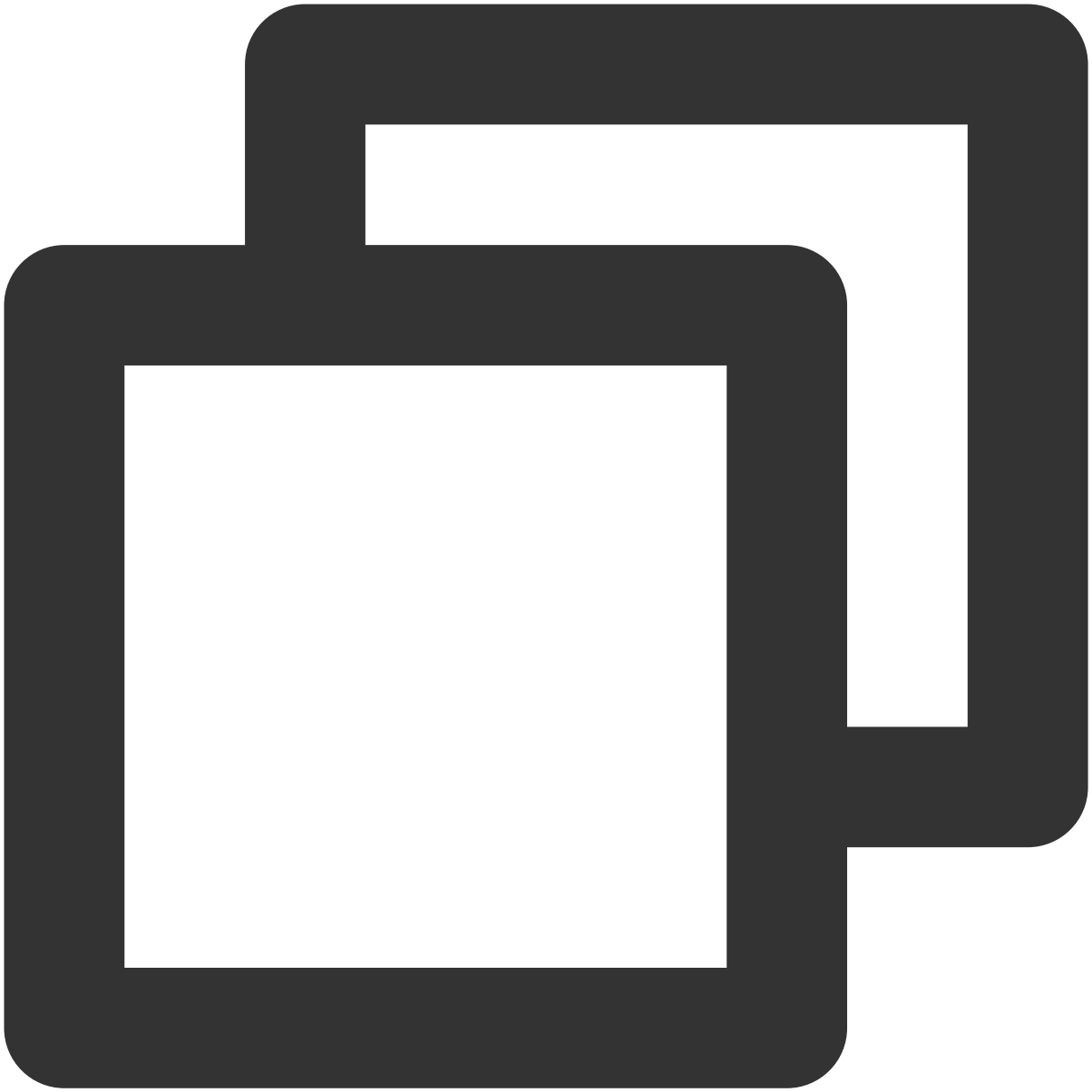
©2013-2022 Tencent Cloud. All rights reserved.



Step 1: The anchor enters the room to push streams

1. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
@property (nonatomic, strong) UIView *anchorPreviewView;
@property (nonatomic, strong) TRTCCloud *trtcCloud;

- (void)setupTRTC {
    self.trtcCloud = [TRTCCloud sharedInstance];
    self.trtcCloud.delegate = self;
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    encParam.videoBitrate = 1300;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify the use of front/rear camera for video capture
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.anchorPreviewVie

    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

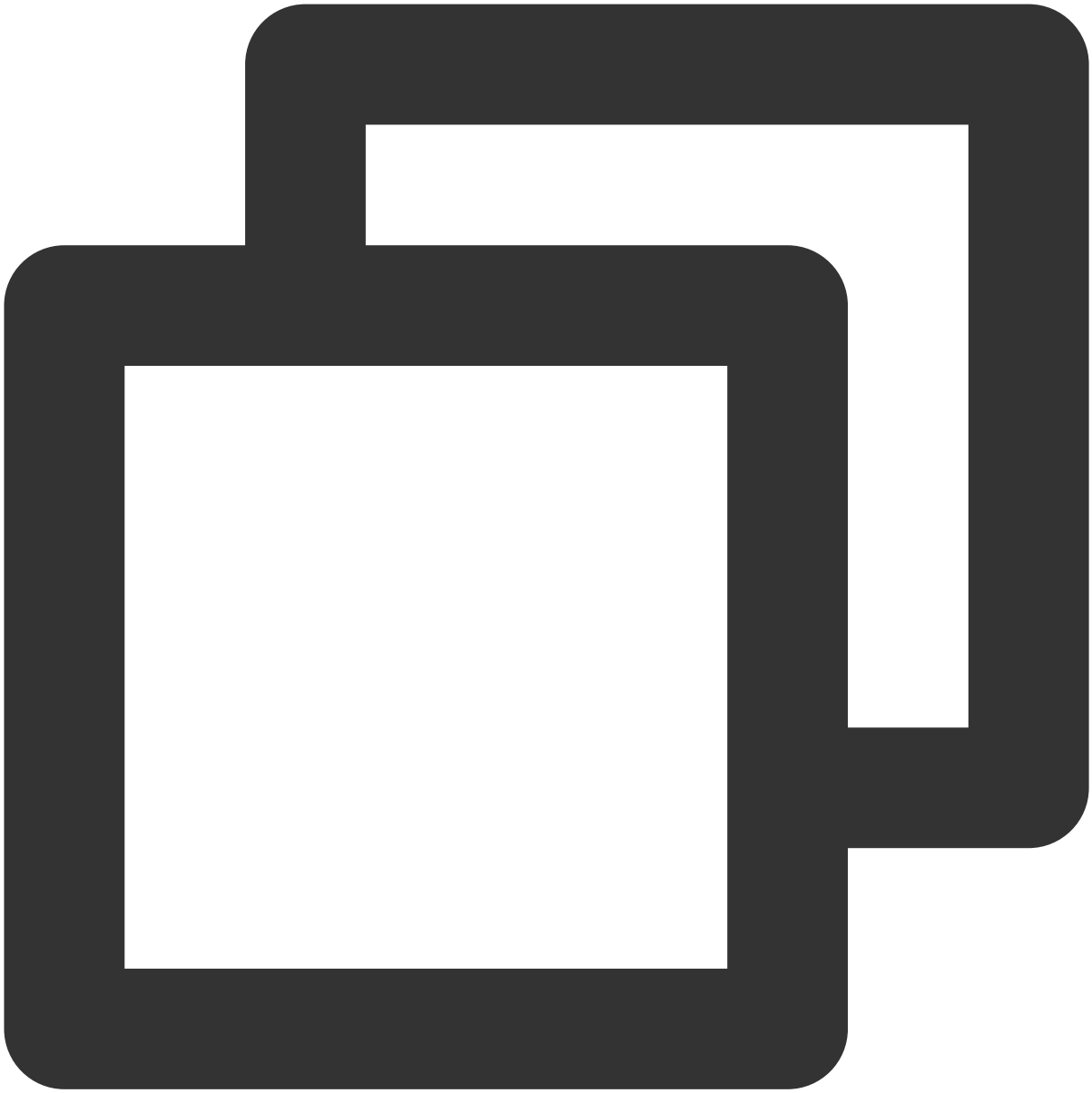
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom` . The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom` . The SDK will start the camera preview and audio capture and automatically start streaming.

2. The anchor sets rendering parameters for the local video, and the encoder output video mode (optional).

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering parameters for the local video
    [self.trtcCloud setLocalRenderParams:params];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Set the video mirror mode for the encoder output
    [self.trtcCloud setVideoEncoderMirror:YES];
    // Set the rotation of the video encoder output
    [self.trtcCloud setVideoEncoderRotation:TRTCVideoRotation_0];
}
```

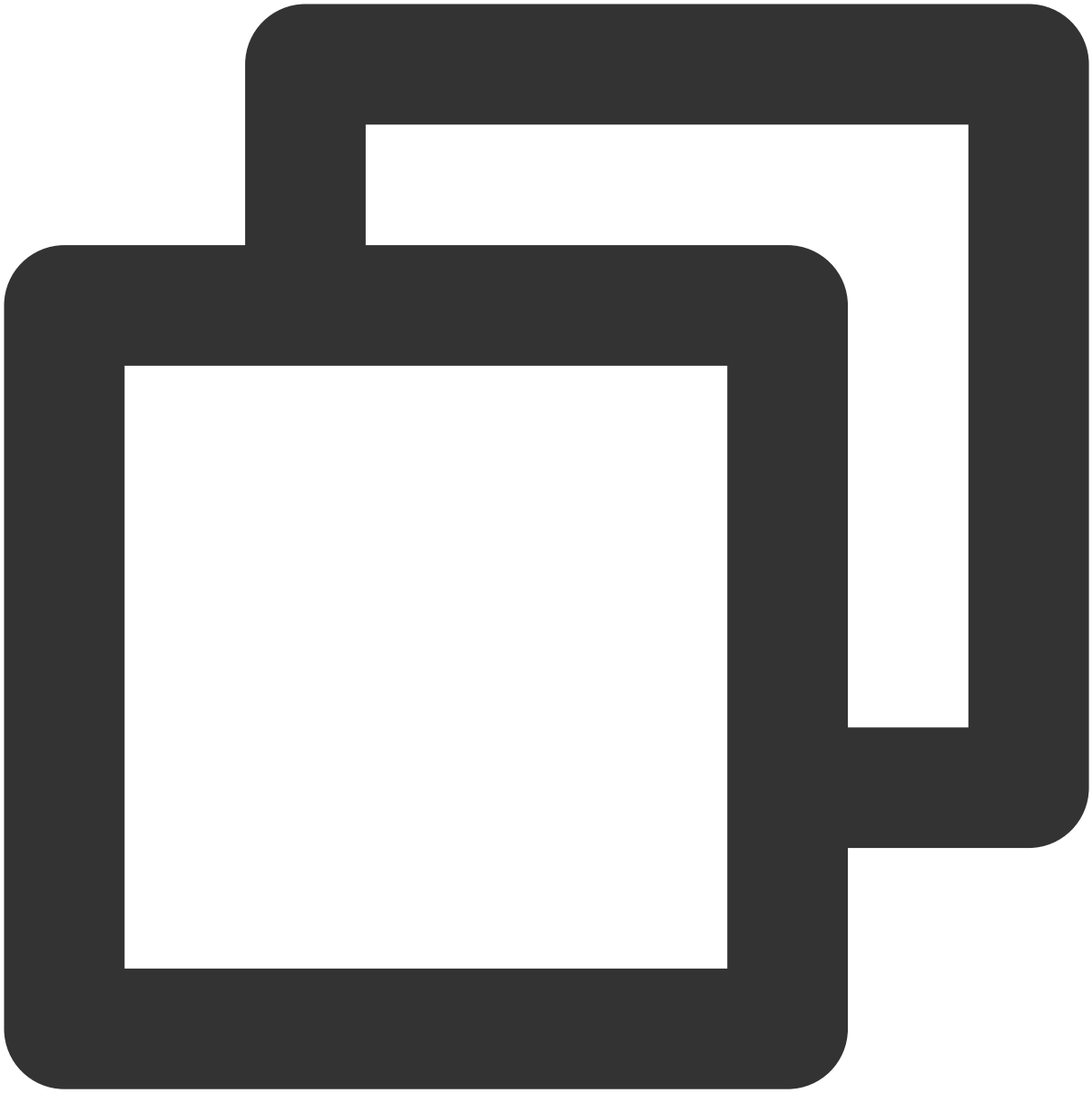
Note:

Setting local screen rendering parameters only affects the rendering effect of the local screen.

Setting encoder output pattern affects the viewing effect for other users in the room (as well as the cloud recording files).

3. The anchor starts the live streaming, entering the room and start streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)enterRoomByAnchorWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = @"userSig";
    // Replace with your SDKAppID
    params.sdkAppId = 0;
    // Specify the anchor role
    params.role = TRTCRoleAnchor;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        NSLog(@"Enter room succeed!");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        NSLog(@"Enter room failed!");
    }
}
```

Note:

TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

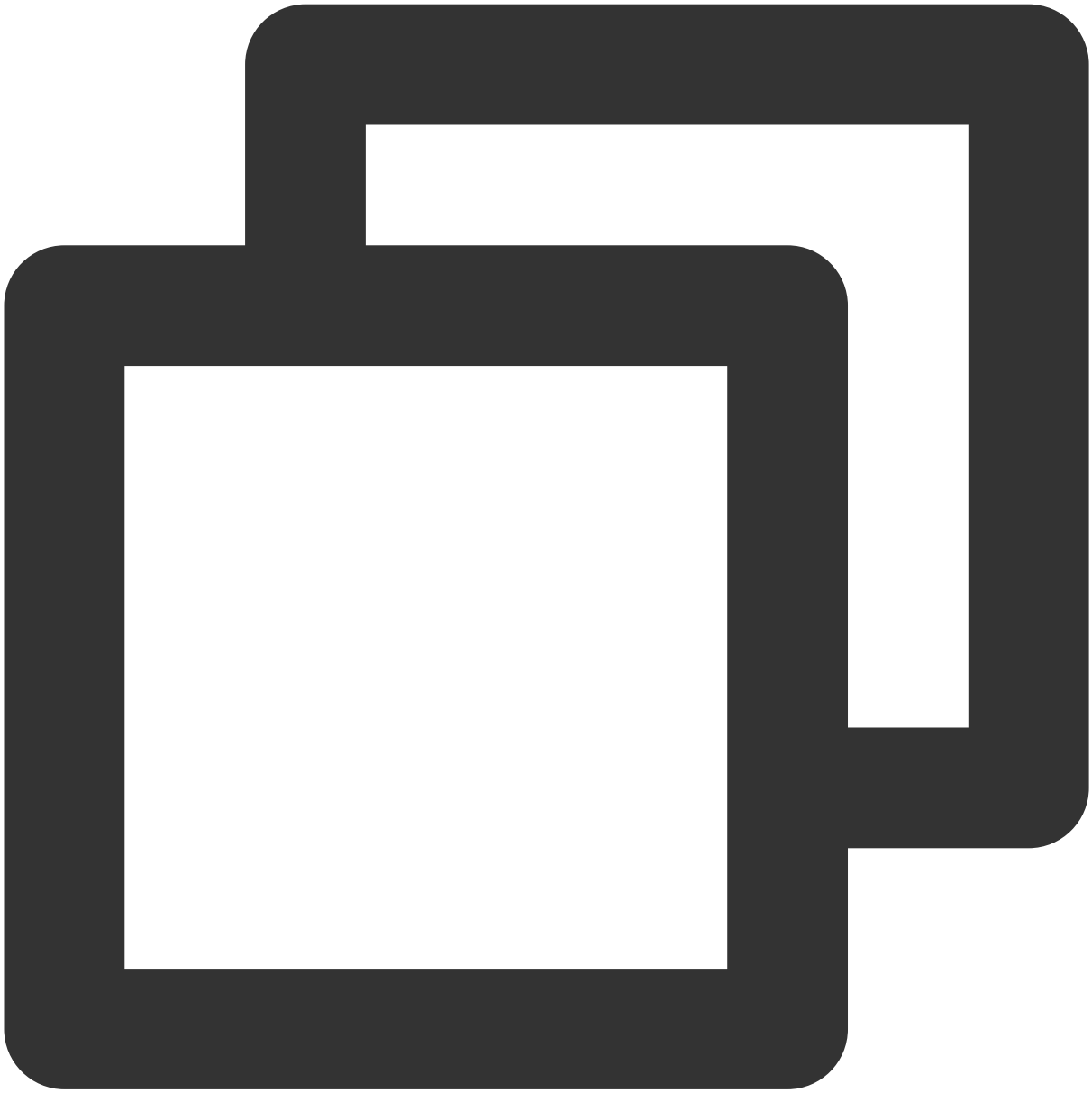
TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In the e-commerce live streaming scenario, it is recommended to choose `TRTCAppSceneLIVE` as the room entry mode.

Step 2: The audience enters the room to pull streams

1. Audience enters the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.



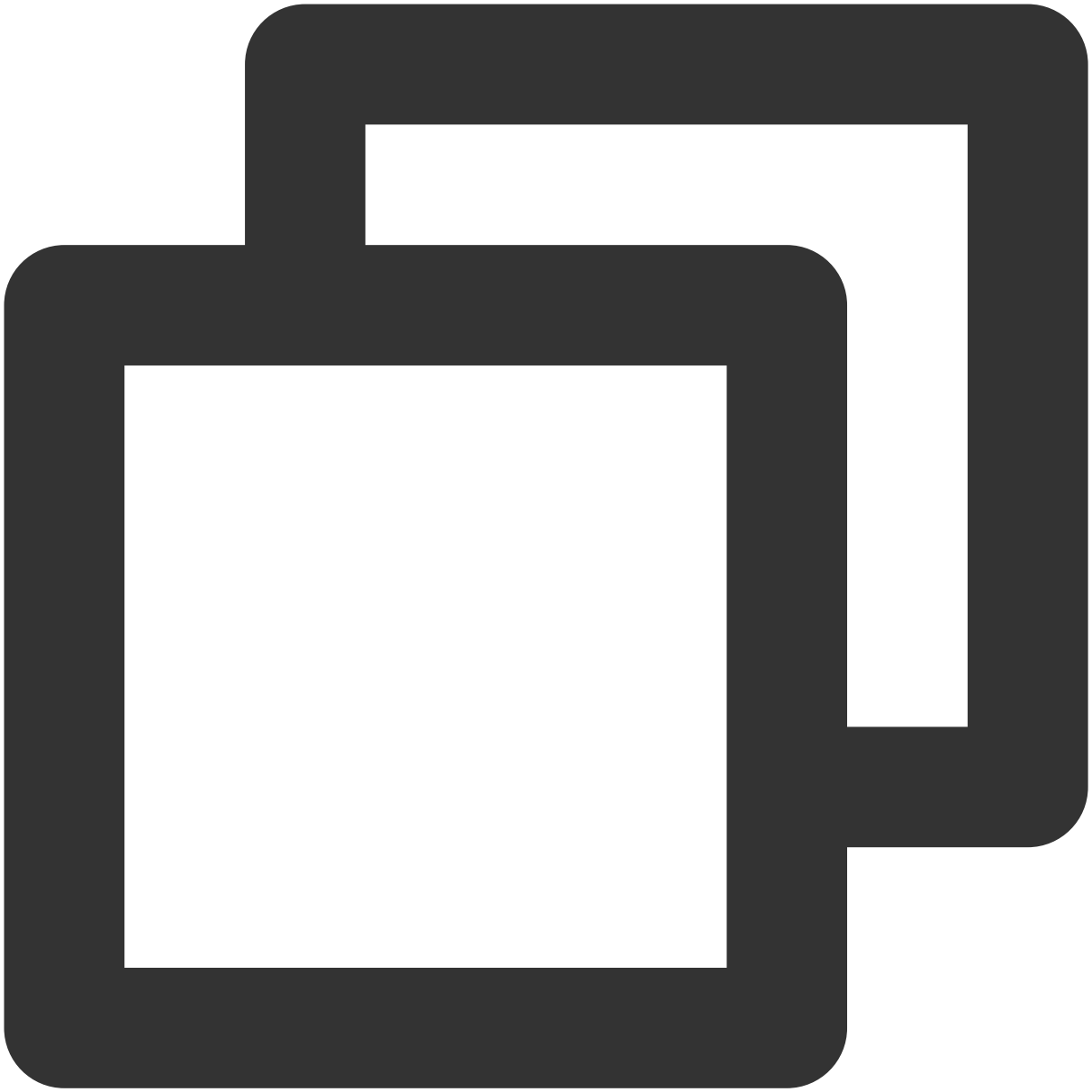
```
- (void)enterRoomByAudienceWithUserId:(NSString *)userId roomId:(NSString *)roomId
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = @"userSig";
    // Replace with your SDKAppID
    params.sdkAppId = 0;
    // Specify the audience role
    params.role = TRTCRoleAudience;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        NSLog(@"Enter room succeed!");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        NSLog(@"Enter room failed!");
    }
}
```

2. Audience subscribes to the anchor's audio and video streams.

©2013-2022 Tencent Cloud. All rights reserved.

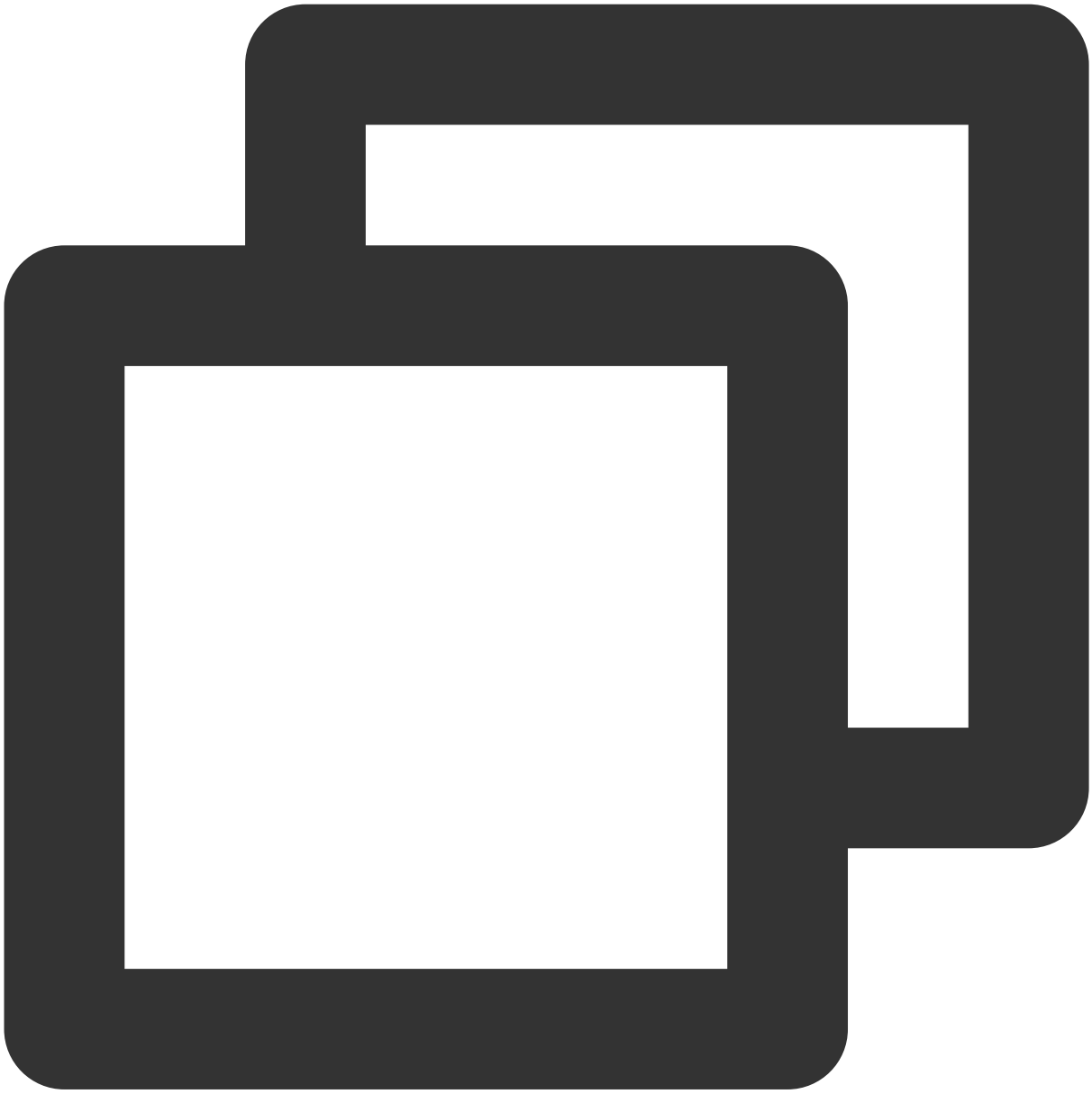


```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}
```

3. Audience sets the rendering mode for the remote video (optional).



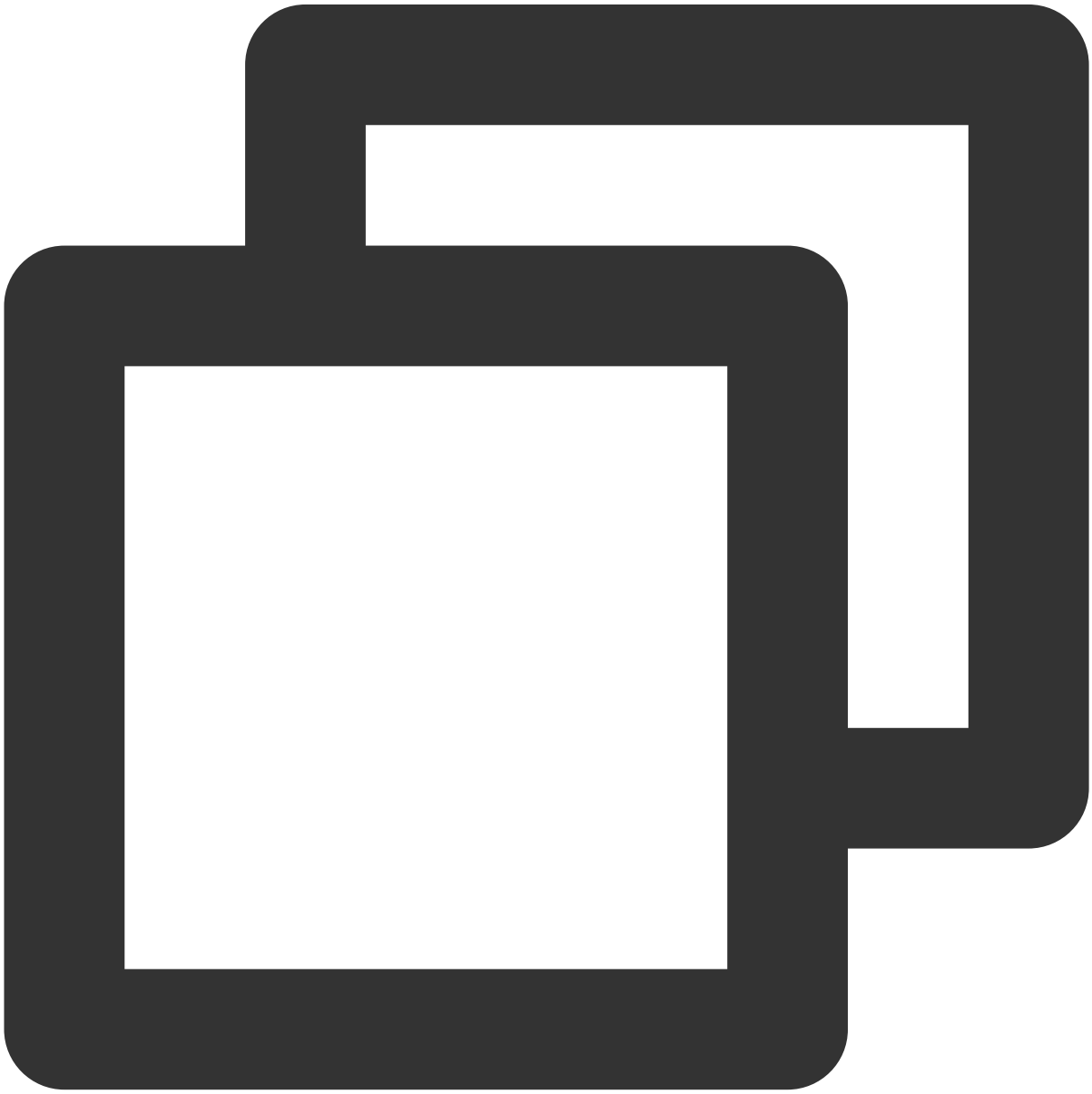
```
- (void)setupRemoteRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering mode for the remote video
    [self.trtcCloud setRemoteRenderParams:@"userId" streamType:TRTCVideoStreamTypeB
}
```

Step 3: The audience interacts via mic-connection

1. The audience is switched to the anchor role.

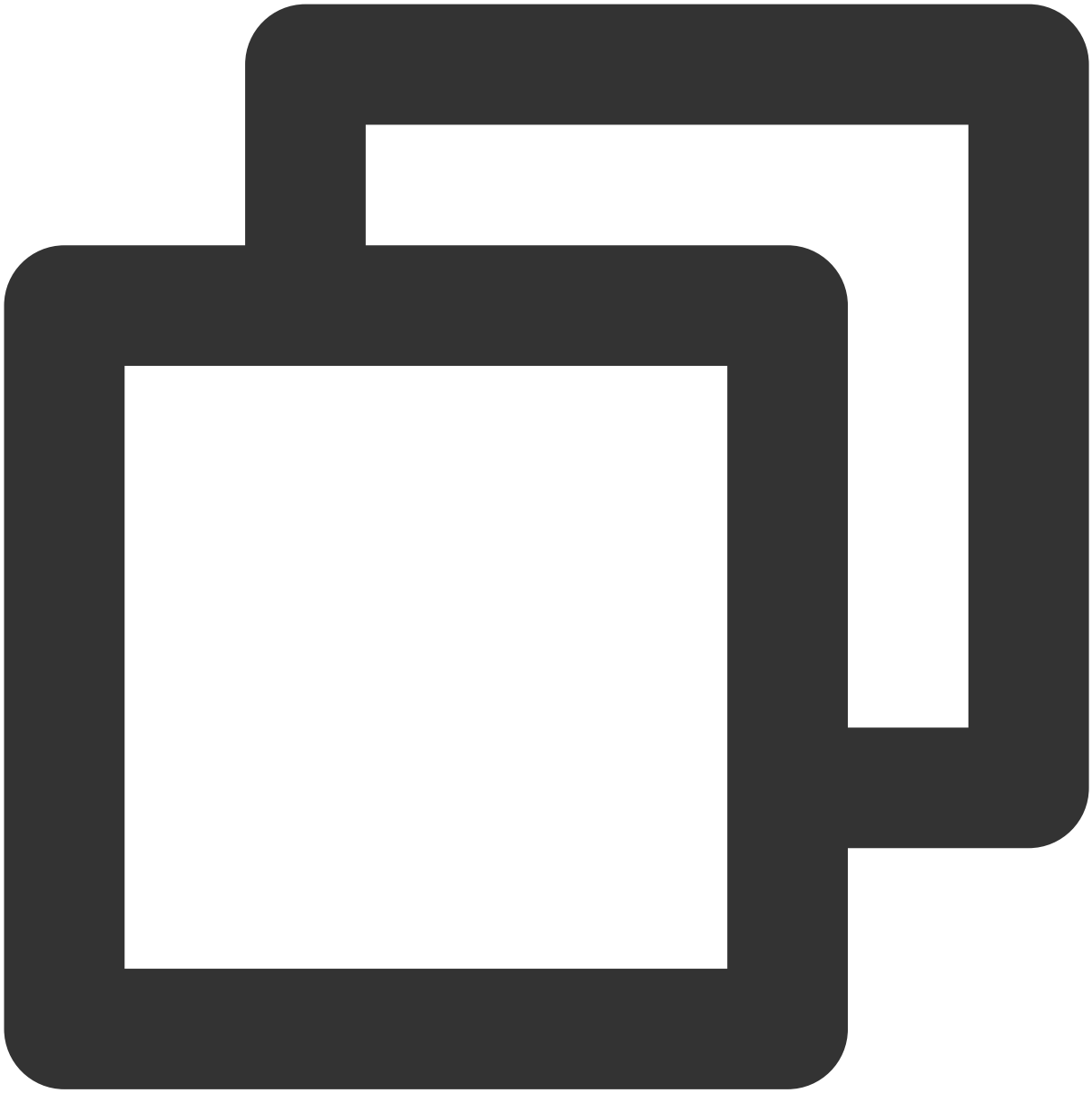


©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)switchToAnchor {
    // Switch to the anchor role
    [self.trtcCloud switchRole:TRTCRoleAnchor];
}

// Event callback for switching role
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Role switched successfully
    }
}
```

2. The audience starts local audio and video capture and streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupTRTC {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify the use of front/rear camera for video capture
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.audiencePreviewV
```

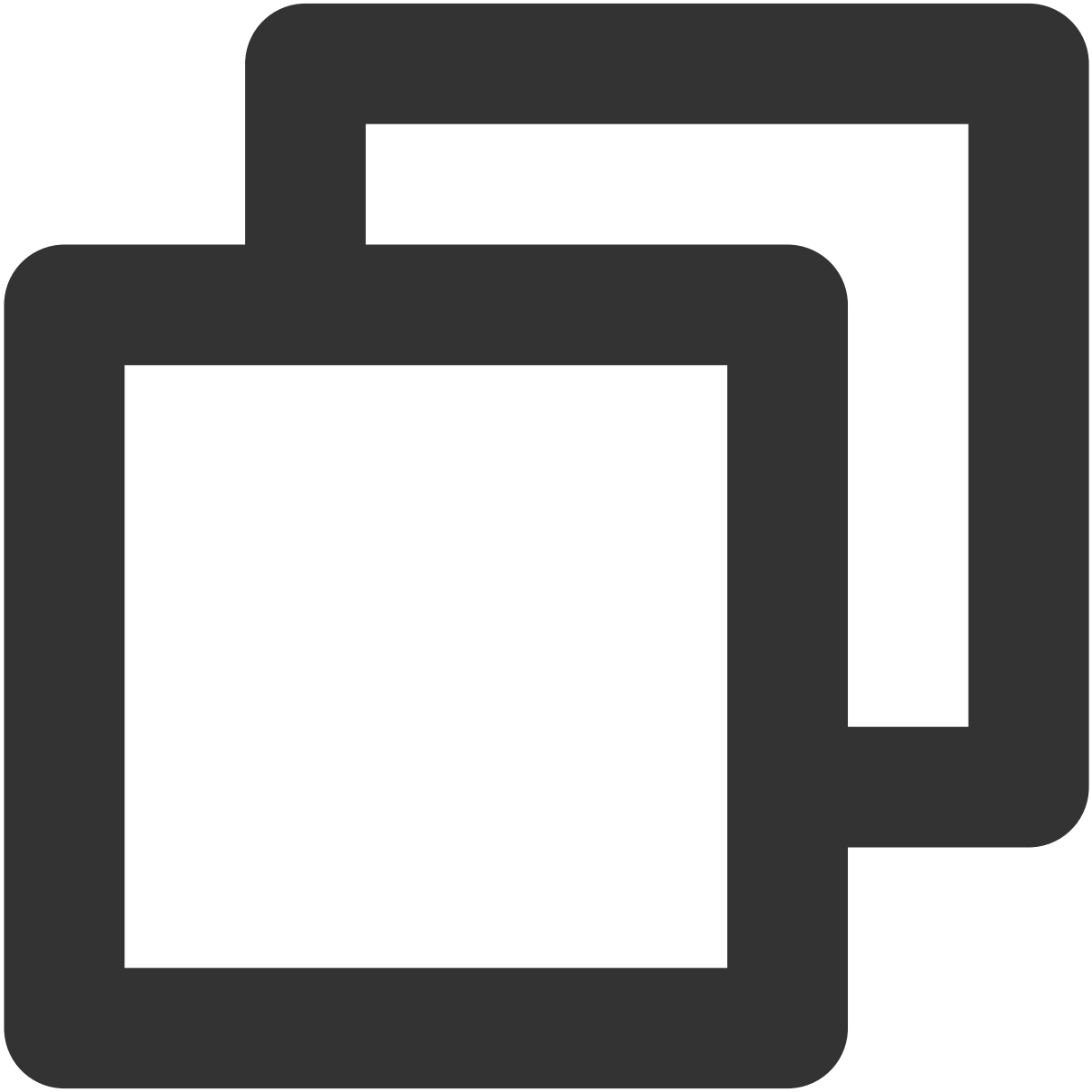
©2013-2022 Tencent Cloud. All rights reserved.

```
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

3. The audience drops the mic and stops streaming.



```
- (void)switchToAudience {
    // Switch to the audience role
```

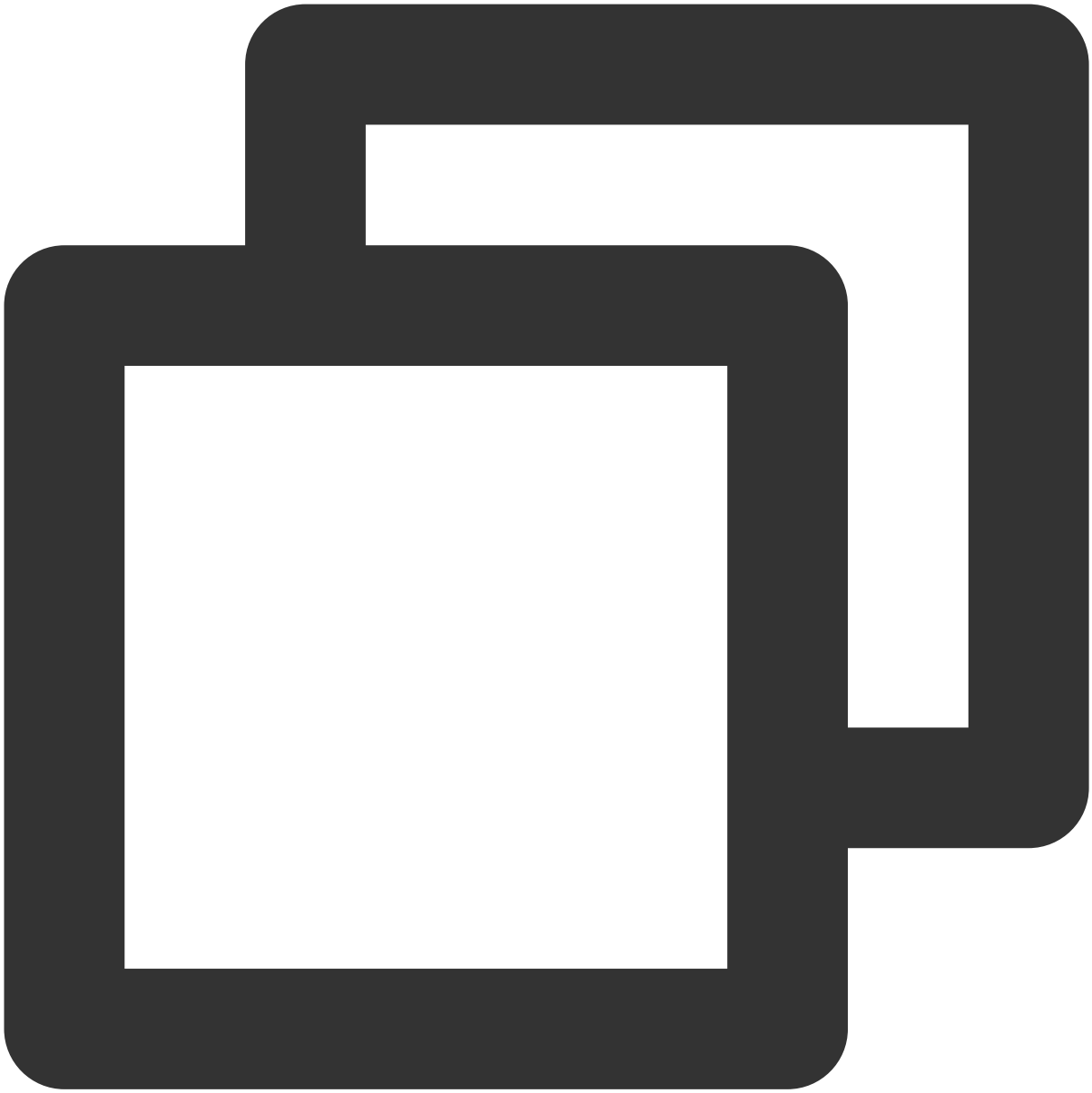
©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud switchRole:TRTCRoleAudience];
}

// Event callback for switching role
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
    if (errCode == ERR_NULL) {
        // Stop camera capture and streaming
        [self.trtcCloud stopLocalPreview];
        // Stop microphone capture and streaming
        [self.trtcCloud stopLocalAudio];
    }
}
```

Step 4: Exit and dissolve the room

1. Exit Room

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)exitRoom {
    [self.trtcCloud stopLocalAudio];
    [self.trtcCloud stopLocalPreview];
    [self.trtcCloud exitRoom];
}

// Event callback for exiting the room
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room");
    } else if (reason == 1) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        NSLog(@"Removed from the current room by the server");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved");
    }
}
```

Note:

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

If you wish to call `enterRoom` again or switch to another audio and video SDK, wait for the `onExitRoom` callback before proceeding. Otherwise, you may encounter various exceptional issues such as the camera, microphone device being forcibly occupied.

2. Dissolve Room

Server dissolves the room

TRTC provides the server dissolves digit type room API `DismissRoom` , as well as server dissolves string type room API DismissRoomByStrRoomId. You can call the server dissolves the room API to remove all users from the room and dissolve the room.

Client dissolves the room

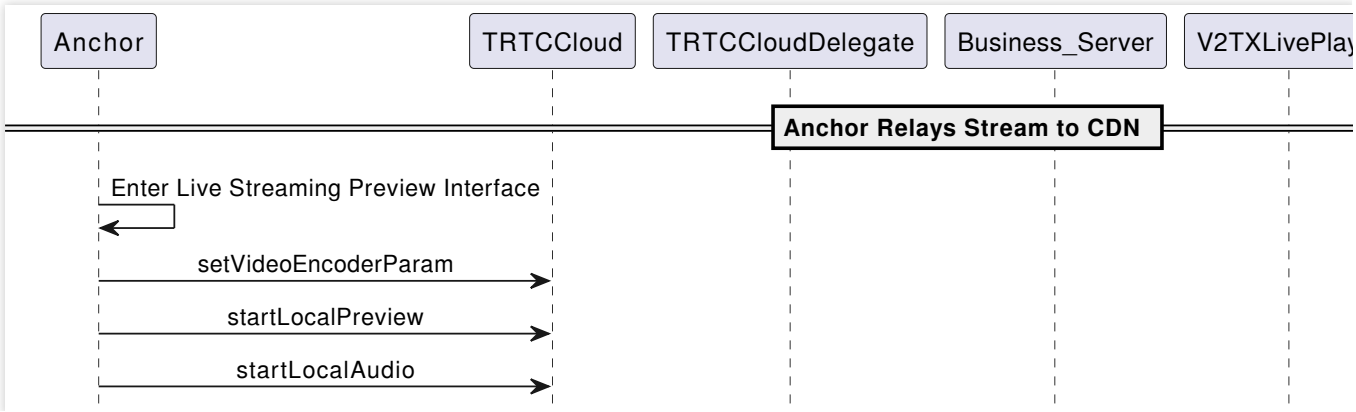
The client does not have a API to directly dissolve the room. Each client needs to call exitRoom to exit the room. Once all anchors and audience have exited, the room will automatically be dissolved according to TRTC's room lifecycle rules. For more details, see TRTC Exits Room.

Warning:

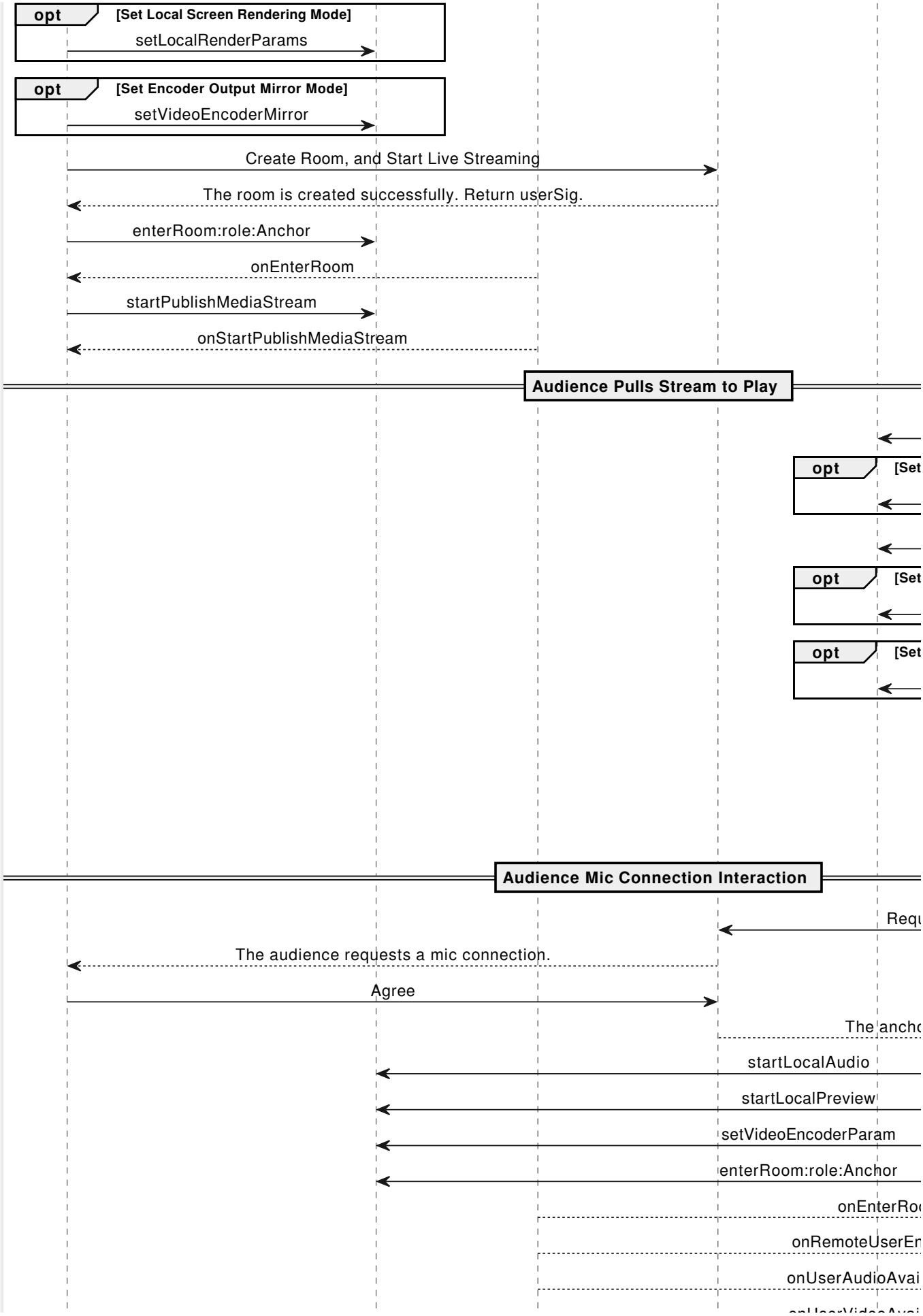
It is recommended that after the end of live streaming, you call the room dissolvement API on the server to ensure the room is dissolved. This will prevent audiences from accidentally entering the room and incurring unexpected charges.

Alternative Solutions

API Sequence Diagram

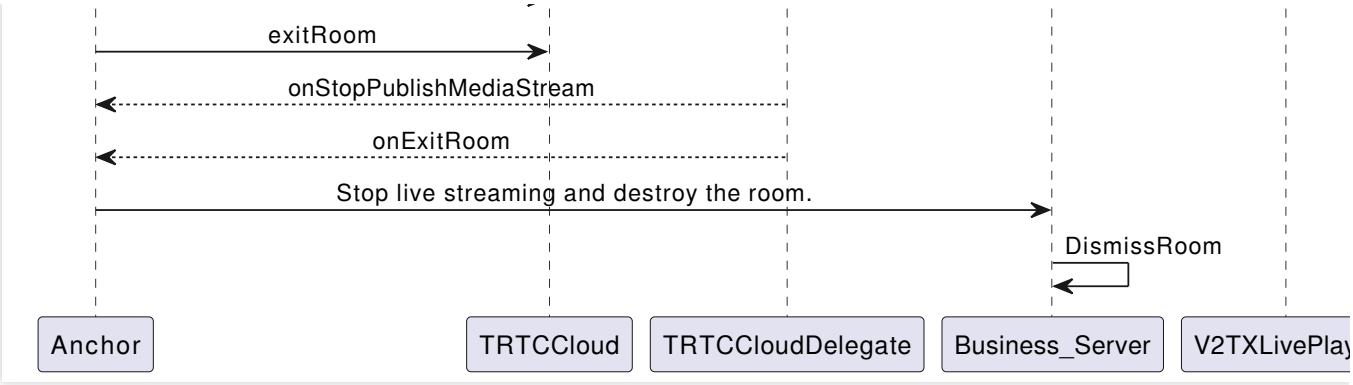


©2013-2022 Tencent Cloud. All rights reserved.



©2013-2022 Tencent Cloud. All rights reserved.

©2013-2022 Tencent Cloud. All rights reserved.

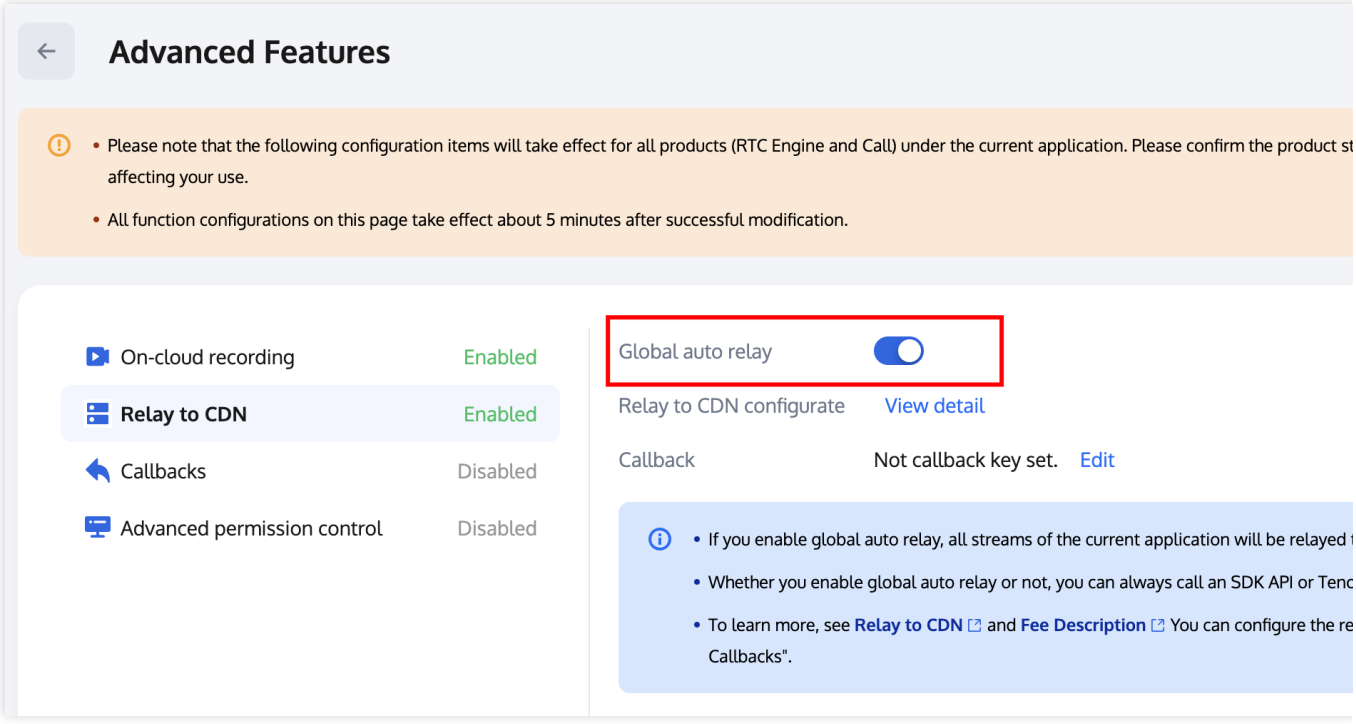


Step 1: The anchor relays stream pushing

1. Related configurations for relaying to live streaming CDN.

Global Automatic Relayed Push

If you need to automatically relay all anchors' audio and video streams in the room to live streaming CDN, you need to enable **Relay to CDN** in the TRTC Console **Advanced Features** page.

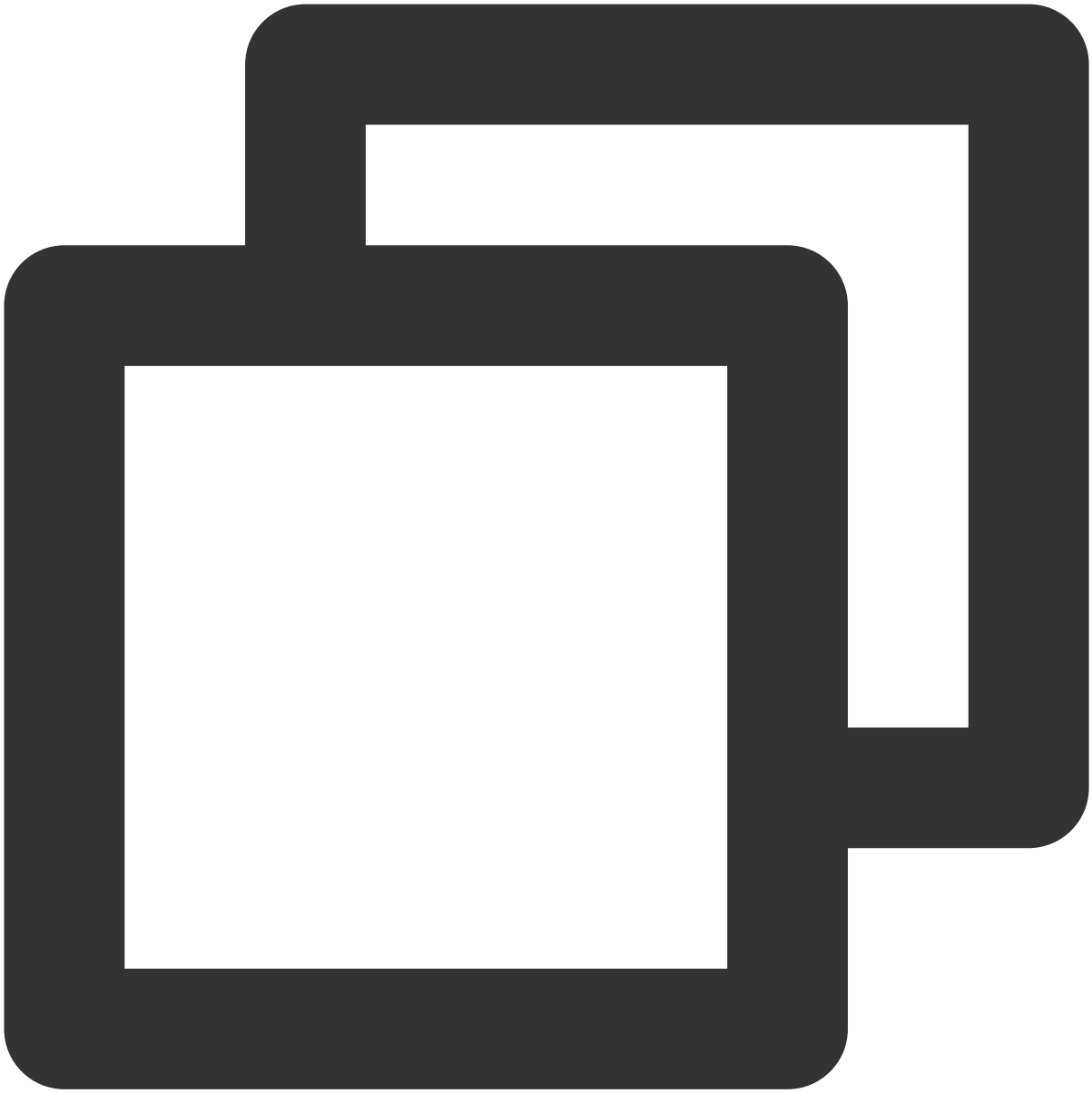


Relayed Push of the Specified Streams

If you need to manually specify the audio and video streams to be published to live streaming CDN, or publish the mixed audio and video streams to live streaming CDN, you can do so by calling the startPublishMediaStream API. In this case, you do not need to activate global automatically relaying to CDN in the console. For a detailed introduction, see Publish Audio and Video Streams to Live Streaming CDN.

2. The anchor activates local video preview and audio capture before entering the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the video rendering control for displaying the anchor's local video previ
@property (nonatomic, strong) UIView *anchorPreviewView;


- (void)setupTRTC {
    self.trtcCloud = [TRTCCloud sharedInstance];
    self.trtcCloud.delegate = self;
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    encParam.videoBitrate = 1300;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify the use of front/rear camera for video capture
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.anchorPreviewVie

    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}
```

Note:

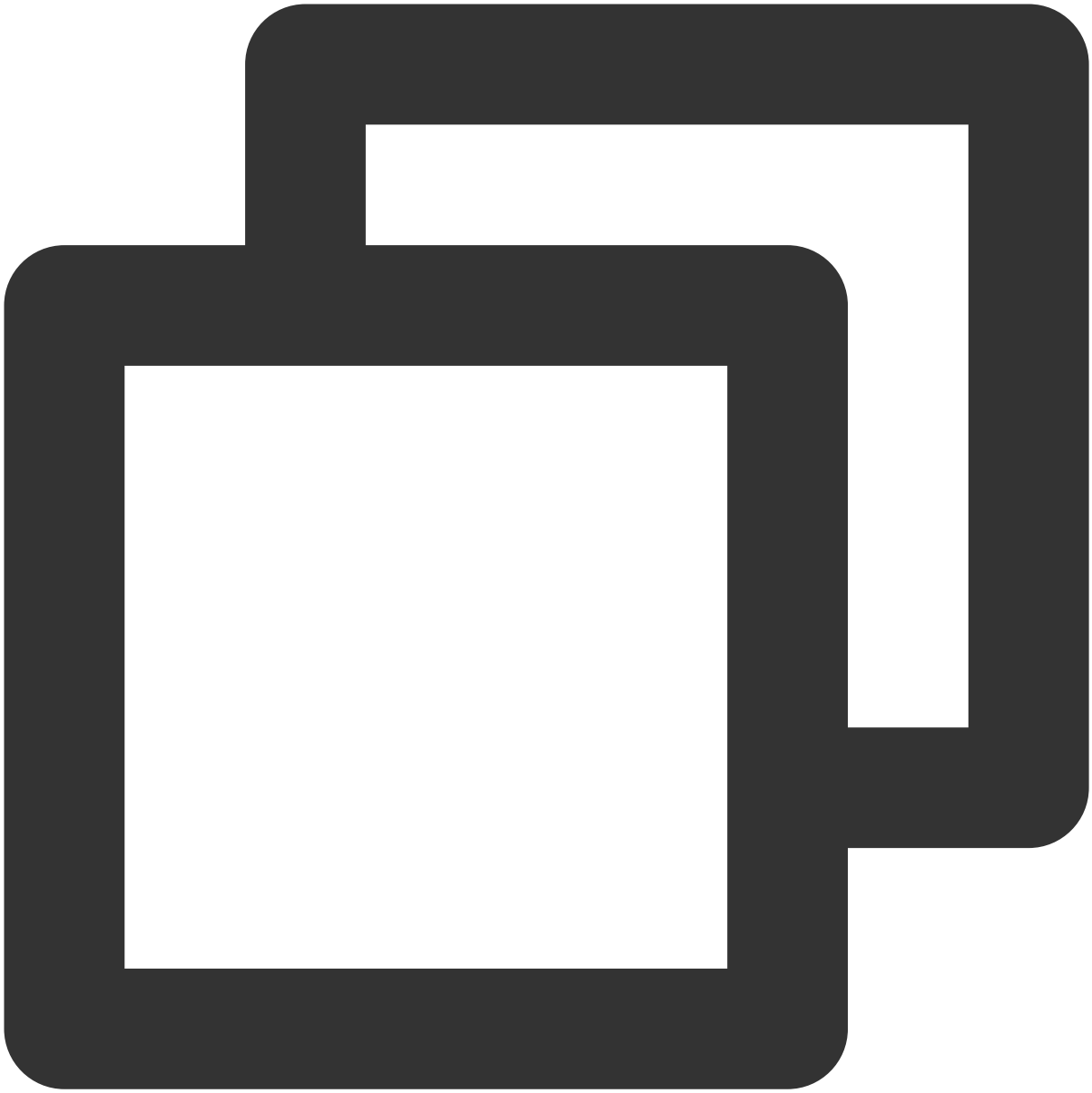
You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above API before `enterRoom`. The SDK will only start the camera preview and audio capture, and wait until you call `enterRoom` to start streaming.

Call the above API after `enterRoom`. The SDK will start the camera preview and audio capture and automatically start streaming.

3. The anchor sets rendering parameters for the local screen, and the encoder output video mode.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupRenderParams {
    TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
    // Video mirror mode
    params.mirrorType = TRTCVideoMirrorTypeAuto;
    // Video fill mode
    params.fillMode = TRTCVideoFillMode_Fill;
    // Video rotation angle
    params.rotation = TRTCVideoRotation_0;
    // Set the rendering parameters for the local video
    [self.trtcCloud setLocalRenderParams:params];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Set the video mirror mode for the encoder output
    [self.trtcCloud setVideoEncoderMirror:YES];
    // Set the rotation of the video encoder output
    [self.trtcCloud setVideoEncoderRotation:TRTCVideoRotation_0];
}
```

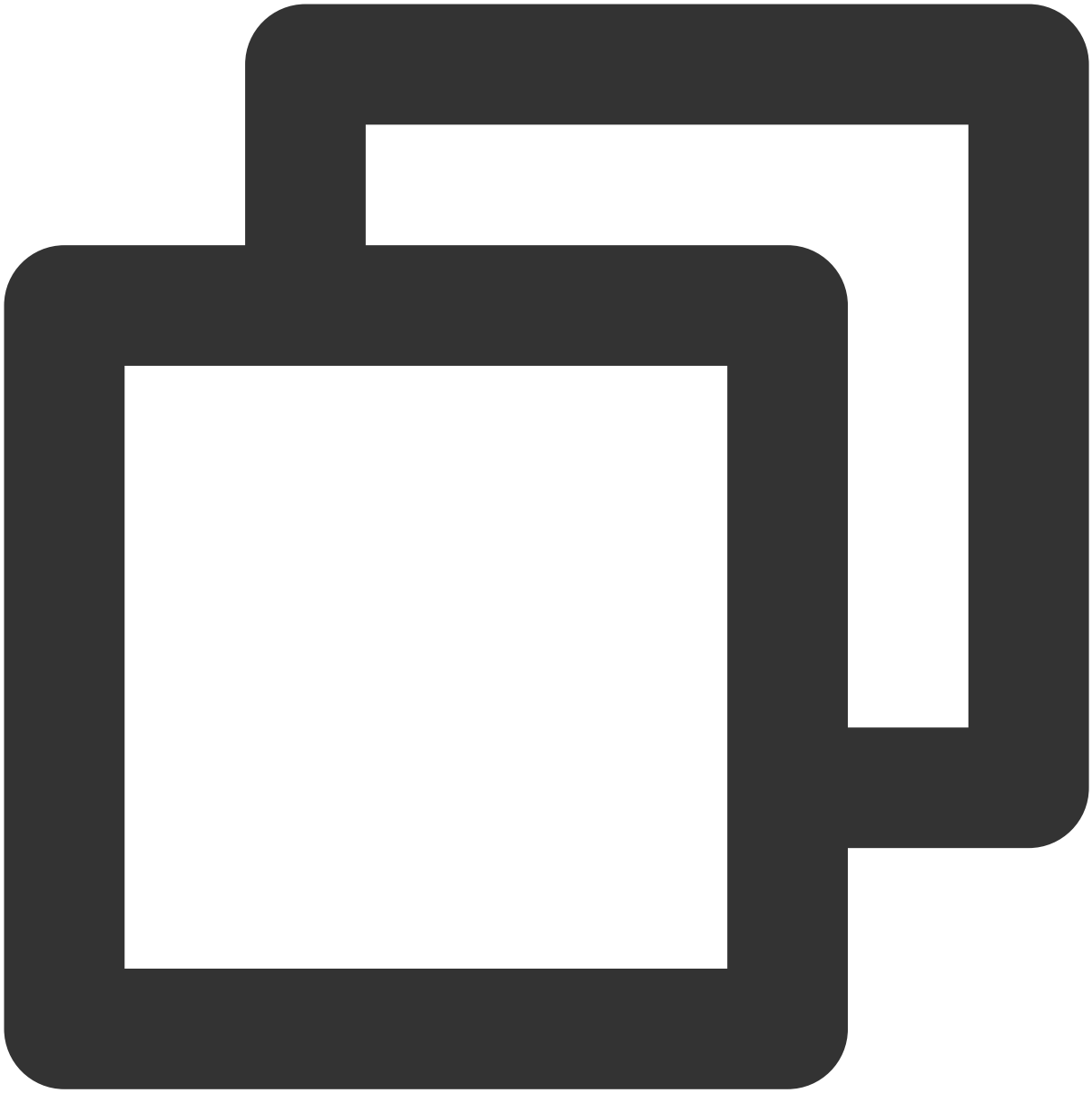
Note:

Setting local screen rendering parameters only affects the rendering effect of the local screen.

Setting encoder output pattern affects the viewing effect for other users in the room (as well as the cloud recording files).

4. The anchor starts the live streaming, entering the room and start streaming.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)enterRoomByAnchorWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = @"userSig";
    // Replace with your SDKAppID
    params.sdkAppId = 0;
    // Specify the anchor role
    params.role = TRTCRoleAnchor;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Enter the room in an interactive live streaming scenario
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        NSLog(@"Enter room succeed!");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        NSLog(@"Enter room failed!");
    }
}
```

Note:

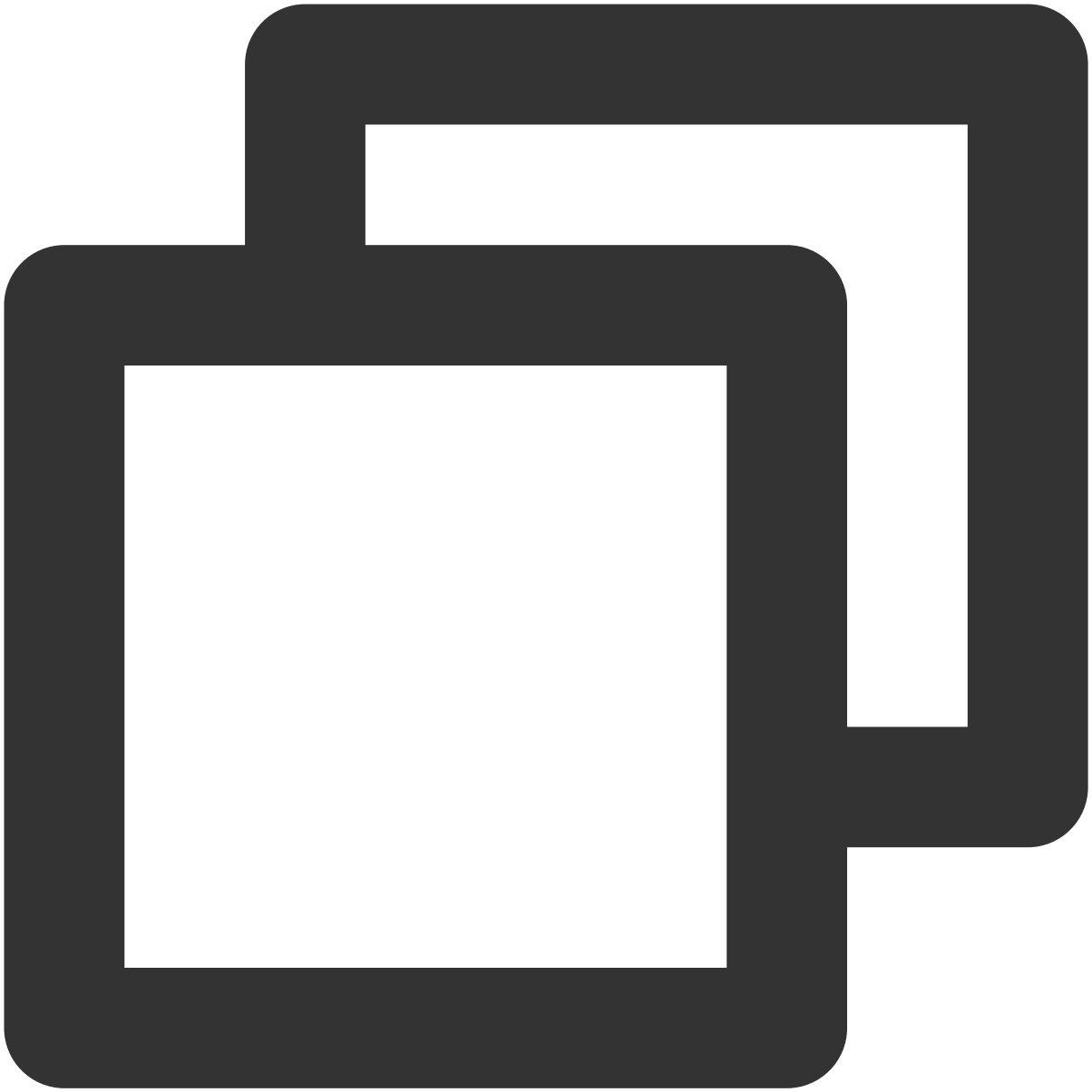
TRTC room IDs are divided into integer type `roomId` and string type `strRoomId`. The rooms of these two types are not interconnected. It is recommended to unify the room ID type.

TRTC user roles are divided into anchors and audiences. Only anchors have streaming permissions. It is necessary to specify the user role when entering the room. If not specified, the default will be the anchor role.

In the e-commerce live streaming scenario, it is recommended to choose `TRTCAppSceneLIVE` as the room entry mode.

5. The anchor relays the audio and video streams to the live streaming CDN.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)startPublishMediaToCDN:(NSString *)streamName {
    NSDate *date = [NSDate dateWithTimeIntervalSinceNow:0];
    // Set the expiration time for the push URLs
    NSTimeInterval time = [date timeIntervalSince1970] + (24 * 60 * 60);
    // Generate authentication information. The getSafeUrl method can be obtained i
    NSString *secretParam = [self getSafeUrl:LIVE_URL_KEY streamName:streamName tim

    // The target URLs for media stream publication
    TRTCPublishTarget* target = [[TRTCPublishTarget alloc] init];
    // The target URLs are set for relaying to CDN
    target.mode = TRTCPublishBigStreamToCdn;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    TRTCPublishCdnUrl* cdnUrl = [[TRTCPublishCdnUrl alloc] init];
    // Construct push URLs (in RTMP format) to the live streaming service provider
    cdnUrl.rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@?%@", PUSH_DOMAI
    // True means CSS push URLs, and false means third-party services
    cdnUrl.isInternalLine = YES;
    NSMutableArray* cdnUrlList = [NSMutableArray array];
    // Multiple CDN push URLs can be added
    [cdnUrlList addObject:cdnUrl];
    target.cdnUrlList = cdnUrlList;

    // Set media stream encoding output parameters (can be defined according to bus
    TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
    encoderParam.audioEncodedSampleRate = 48000;
    encoderParam.audioEncodedChannelNum = 1;
    encoderParam.audioEncodedKbps = 50;
    encoderParam.audioEncodedCodecType = 0;
    encoderParam.videoEncodedWidth = 540;
    encoderParam.videoEncodedHeight = 960;
    encoderParam.videoEncodedFPS = 15;
    encoderParam.videoEncodedGOP = 2;
    encoderParam.videoEncodedKbps = 1300;

    // Start publishing media stream
    [self.trtcCloud startPublishMediaStream:target encoderParam:encoderParam mixing
}
```

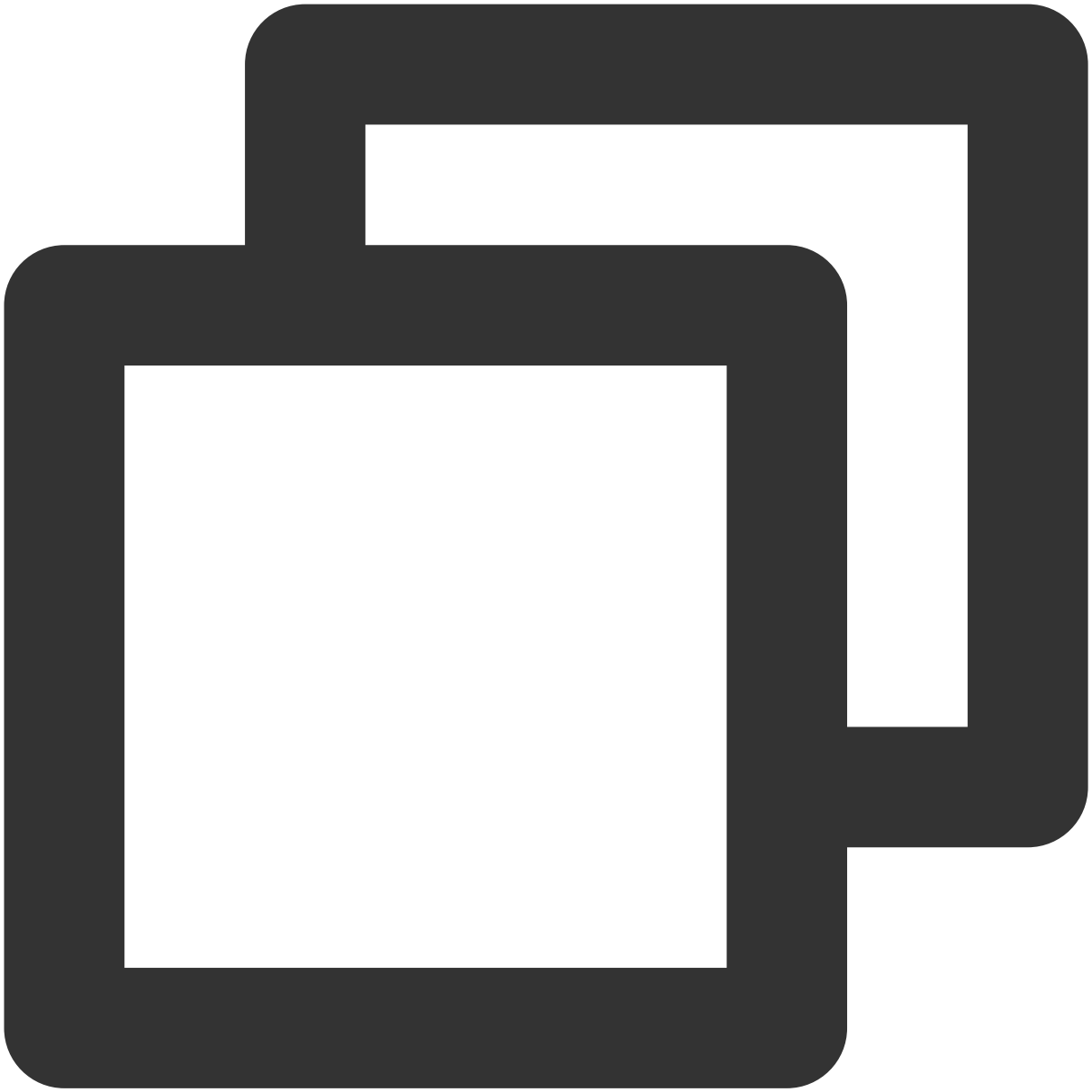
Note:

During single-anchor live streaming, only initiate the relayed push task. When there is an audience mic-connection or anchor PK, update this task to a mixed-stream transcoding task.

Information of push authentication KEY LIVE_URL_KEY and push domain name PUSH_DOMAIN are required to obtain on Domain Management page in the CSS Console.

After the media stream is published, SDK will provide the backend-initiated task identifier (taskId) through the callback onStartPublishMediaStream.

©2013-2022 Tencent Cloud. All rights reserved.

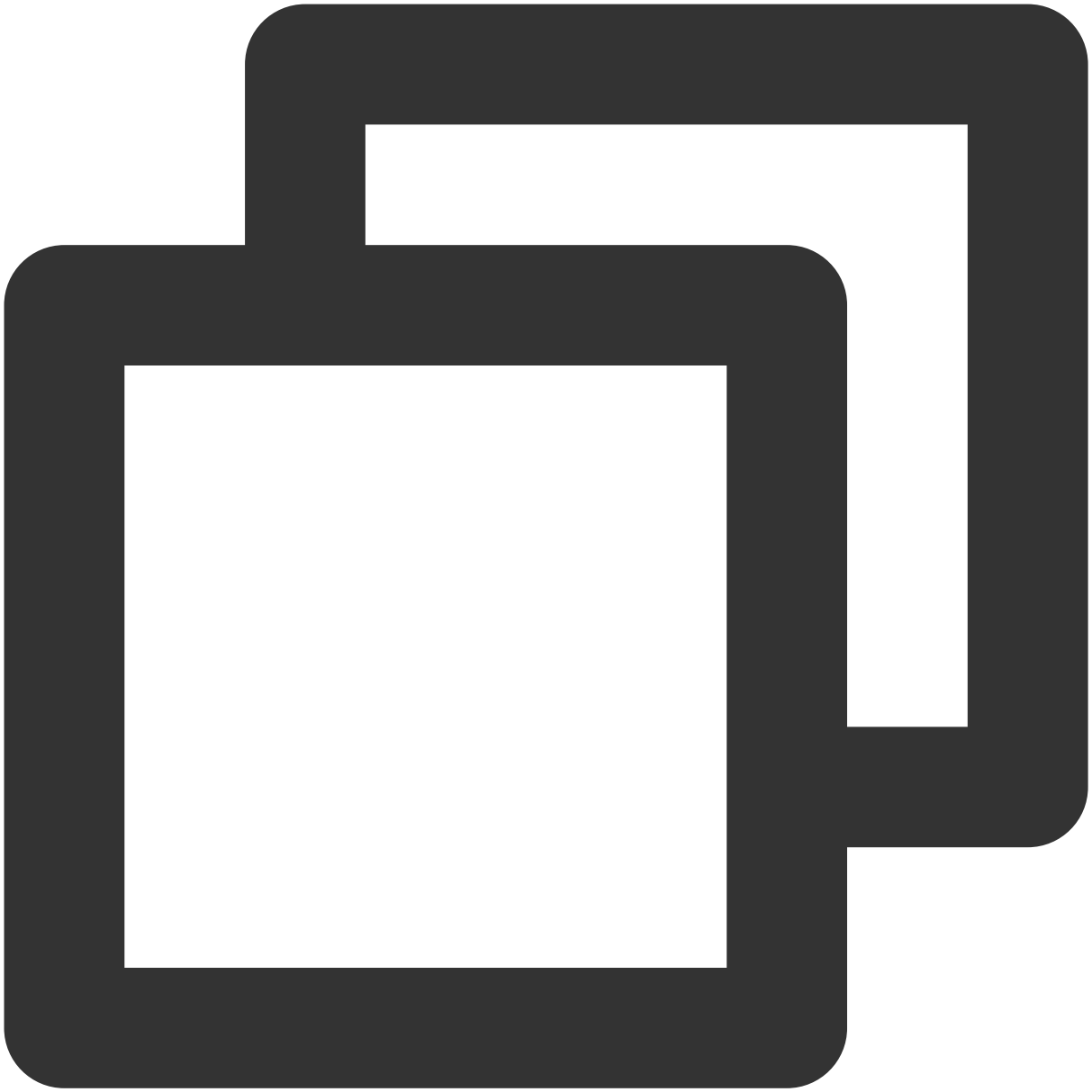


```
- (void)onStartPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStri
    // taskId: When the request is successful, TRTC backend will provide the taskId
    // code: Callback result. 0 means success and other values mean failure
}
```

Step 2: The audience pulls streams for playback

CDN audience do not need to enter the TRTC room; they can directly pull the anchor's CDN stream for playback.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Initialize the player
self.livePlayer = [[V2TXLivePlayer alloc] init];
// Set the player callback listener
[self.livePlayer setObserver:self];
// Set the video rendering control for the player
[self.livePlayer setRenderView:self.remoteView];
// Set delay management mode (optional)
[self.livePlayer setCacheParams:1.f maxTime:5.f]; // Auto mode
[self.livePlayer setCacheParams:1.f maxTime:1.f]; // Speed mode
[self.livePlayer setCacheParams:5.f maxTime:5.f]; // Smooth mode
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Concatenate the pull URLs for playback
NSString *flvUrl = [NSString stringWithFormat:@"http://%@/live/%@.flv", PLAY_DOMAIN
NSString *hlsUrl = [NSString stringWithFormat:@"http://%@/live/%@.m3u8", PLAY_DOMAI
NSString *rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@", PLAY_DOMAIN, s
NSString *webrtcUrl = [NSString stringWithFormat:@"webrtc://%@/live/%@", PLAY_DOMAI

// Start playing
[self.livePlayer startLivePlay:flvUrl];

// Custom set fill mode (optional)
[self.livePlayer setRenderFillMode:V2TXLiveFillModeFit];
// Customize video rendering direction (optional)
[self.livePlayer setRenderRotation:V2TXLiveRotation0];
```

Note:

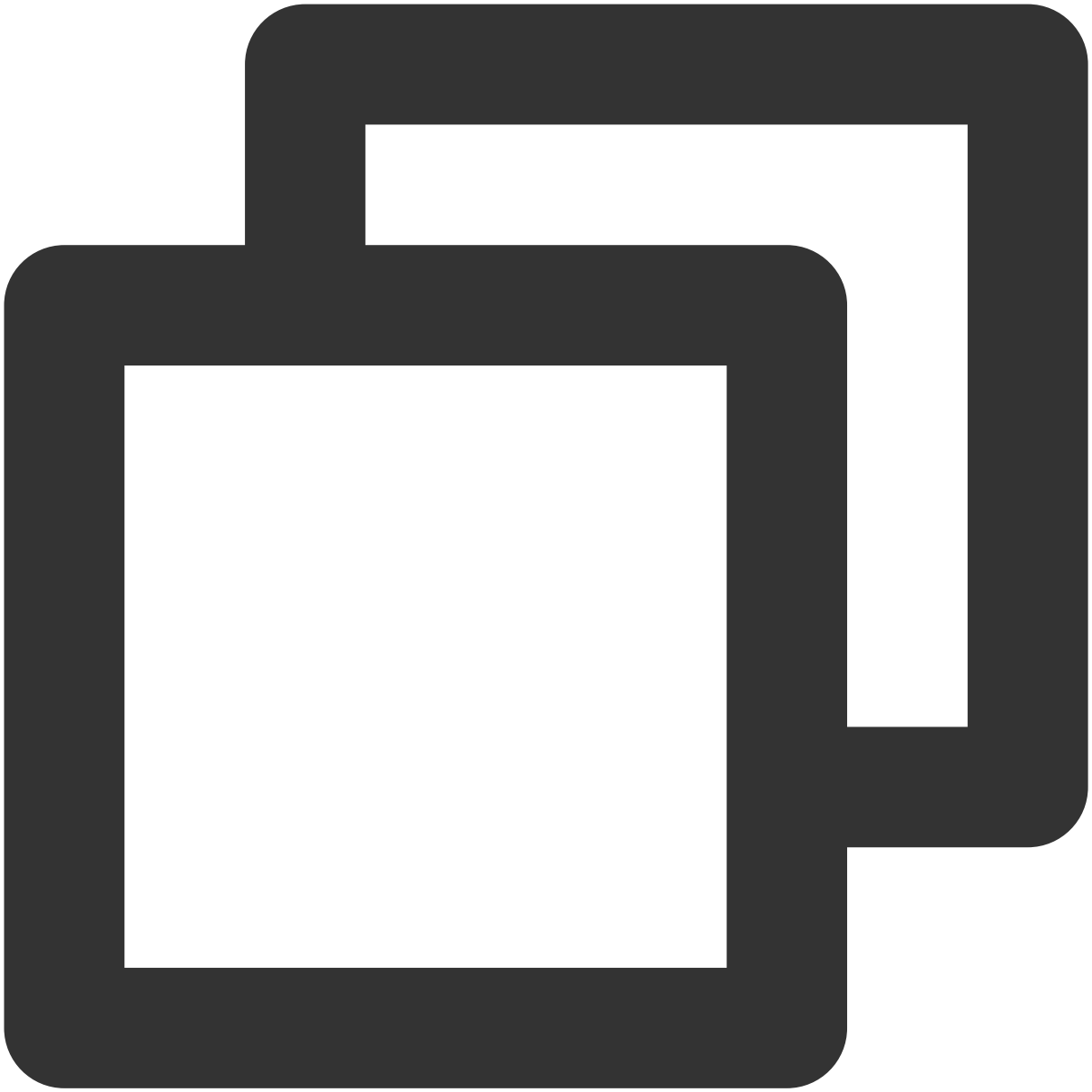
The playback domain name PLAY_DOMAIN requires you to Add Your Own Domain in the CSS console for live streaming playback. You also should configure domain CNAME.

To use the live streaming, you need to configure the player's Licence authorization in advance, or the playback will fail (black screen). For details, see Authentication and Authorization.

Step 3: The audience interacts via mic-connection

1. The mic-connection audiences need to enter the TRTC room for real-time interaction with the anchor.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Enter the TRTC room and start streaming
- (void)enterRoomWithUserId:(NSString *)userId roomId:(NSString *)roomId {
    TRTCParams *params = [[TRTCParams alloc] init];
    // Take the room ID string as an example
    params.strRoomId = roomId;
    params.userId = userId;
    // UserSig obtained from the business backend
    params.userSig = @"userSig";
    // Replace with your SDKAppID
    params.sdkAppId = 0;
    // Specify the anchor role
```

©2013-2022 Tencent Cloud. All rights reserved.

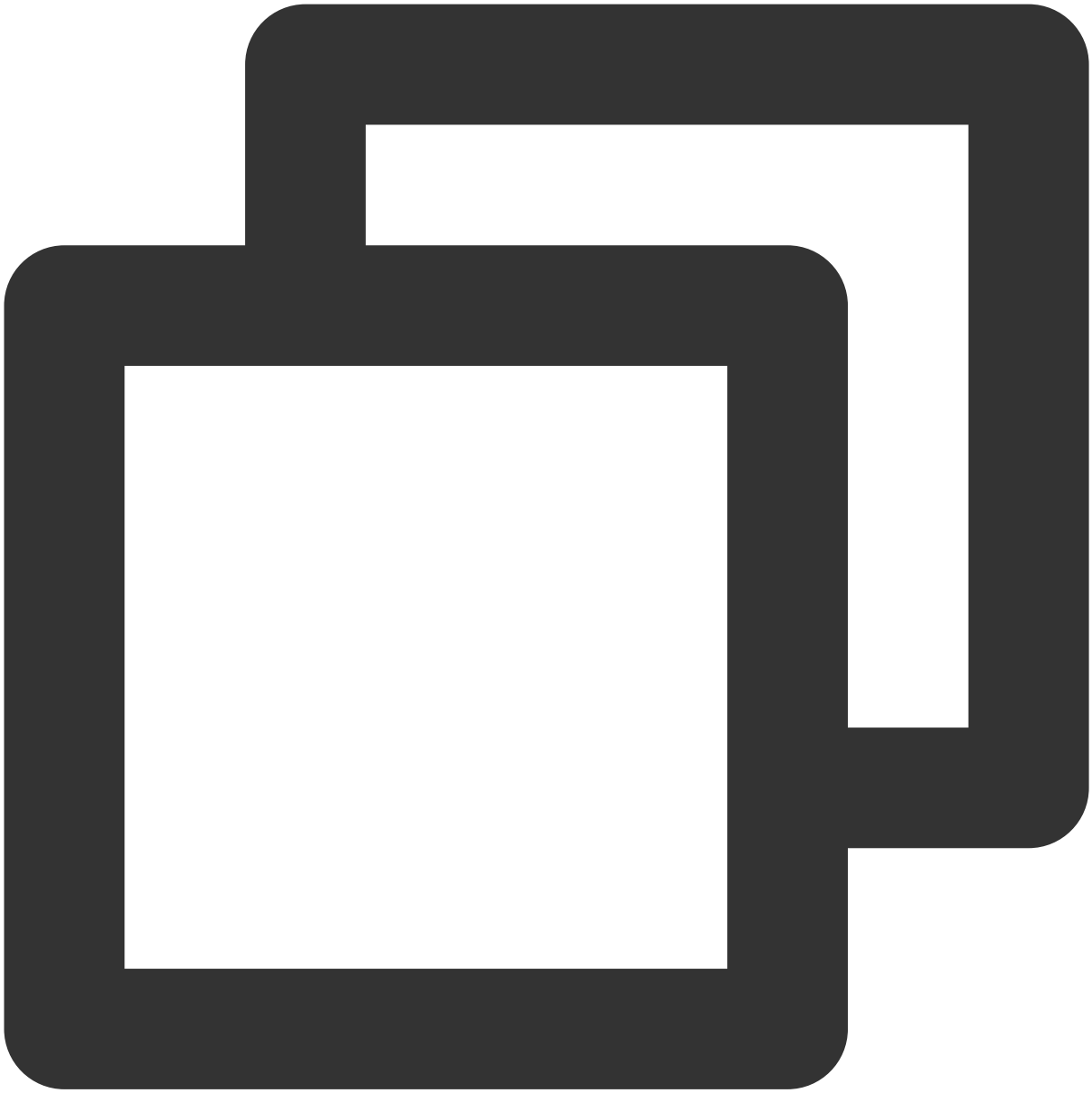
```
    params.role = TRTCRoleAnchor;

    // Enable local audio and video capture
    [self startLocalMedia];
    // Enter the room in an interactive live streaming scenario
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
}

// Enable local video preview and audio capture
- (void)startLocalMedia {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_480_270;
    encParam.videoFps = 15;
    encParam.videoBitrate = 550;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // isFrontCamera can specify the use of front/rear camera for video capture
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.audiencePreviewV
    // Here you can specify the audio quality, from low to high as SPEECH/DEFAULT/M
    [self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
}

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // The result represents the time taken to join the room (in milliseconds)
        NSLog(@"Enter room succeed!");
    } else {
        // The result represents the error code fwhen you fail to enter the room
        NSLog(@"Enter room failed!");
    }
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

2. The mic-connection audience start subscribing to the anchor's audio and video streams after they successfully enter the room.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}

- (void)onFirstVideoFrame:(NSString *)userId streamType:(TRTCVideoStreamType)stream
    // The SDK starts rendering the first frame of the local or remote user's video
    if (![userId isEqualToString:@""]) {
        // Stop playing the CDN stream upon receiving the first frame of the anchor
        [self.livePlayer stopPlay];
    }
}
```

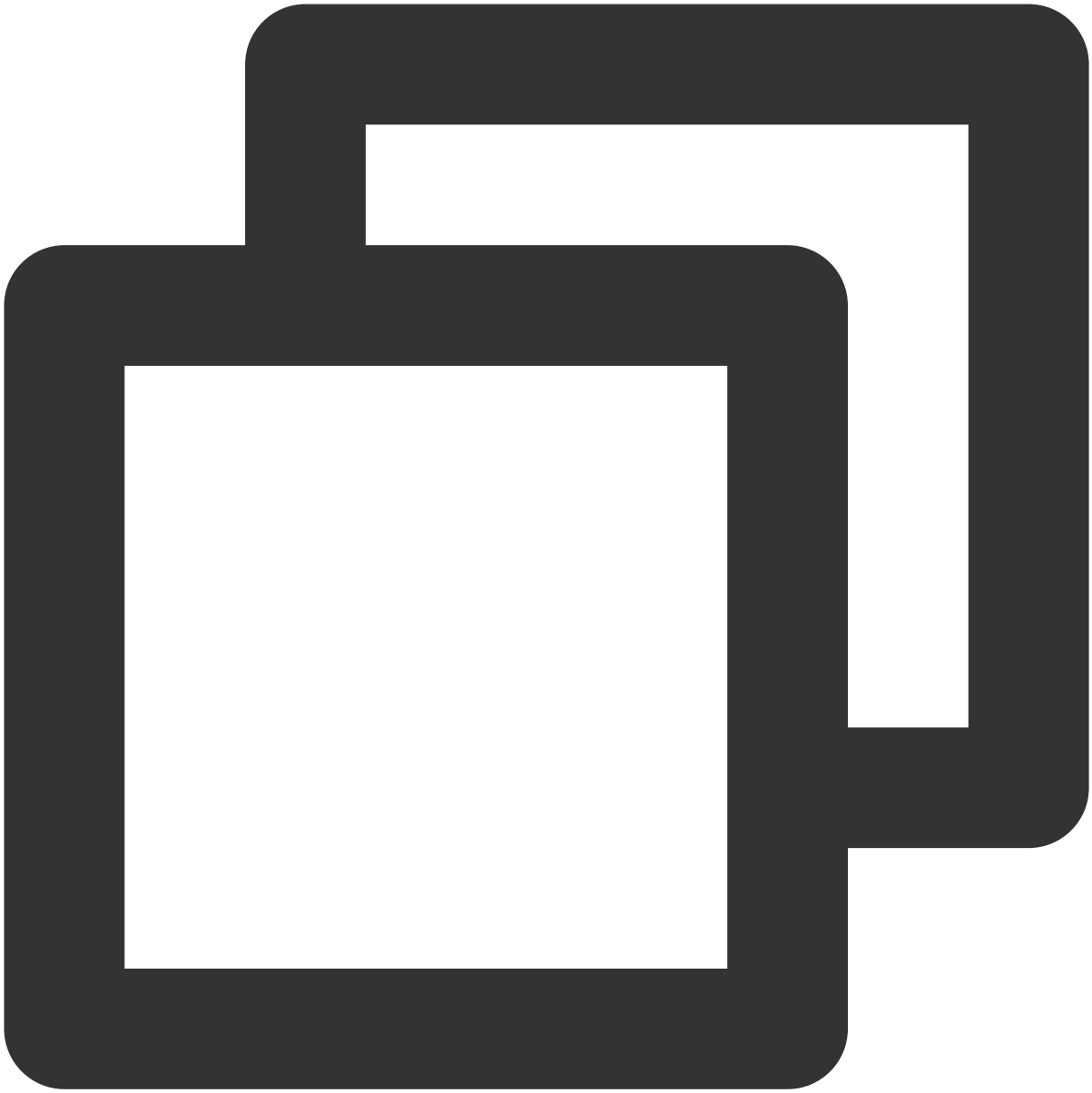
Note:

TRTC stream pulling `startRemoteView` can directly reuse the video rendering control previously used by the CDN stream pulling `setRenderView` .

To avoid video interruptions when switching between stream pullers, it is recommended to wait until the TRTC first frame callback `onFirstVideoFrame` is received before stopping the CDN stream pulling.

3. The anchor updates the publication of mixed media streams.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Event callback for the mic-connection audience's room entry
- (void)onRemoteUserEnterRoom:(NSString *)userId {
    if (![self.mixUserList containsObject:userId]) {
        [self.mixUserList addObject:userId];
    }
    [self updatePublishMediaToCDN];
}

// Update the publication of mixed media streams to the live streaming CDN
- (void)updatePublishMediaToCDN {
    NSDate *date = [NSDate dateWithTimeIntervalSinceNow:0];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Set the expiration time for the push URLs
NSTimeInterval time = [date timeIntervalSince1970] + (24 * 60 * 60);
// Generate authentication information. The getSafeUrl method can be obtained i
NSString *secretParam = [self getSafeUrl:LIVE_URL_KEY streamName:self.streamNam

// The target URLs for media stream publication
TRTCPublishTarget* target = [[TRTCPublishTarget alloc] init];
// The target URLs are set for relaying the mixed streams to CDN
target.mode = TRTCPublishMixStreamToCdn;
TRTCPublishCdnUrl* cdnUrl = [[TRTCPublishCdnUrl alloc] init];
// Construct push URLs (in RTMP format) to the live streaming service provider
cdnUrl.rtmpUrl = [NSString stringWithFormat:@"rtmp://%@/live/%@?%@", PUSH_DOMAI
// True means CSS push URLs, and false means third-party services
cdnUrl.isInternalLine = YES;
NSMutableArray* cdnUrlList = [NSMutableArray array];
// Multiple CDN push URLs can be added
[cdnUrlList addObject:cdnUrl];
target.cdnUrlList = cdnUrlList;

// Set media stream encoding output parameters
TRTCStreamEncoderParam* encoderParam = [[TRTCStreamEncoderParam alloc] init];
encoderParam.audioEncodedSampleRate = 48000;
encoderParam.audioEncodedChannelNum = 1;
encoderParam.audioEncodedKbps = 50;
encoderParam.audioEncodedCodecType = 0;
encoderParam.videoEncodedWidth = 540;
encoderParam.videoEncodedHeight = 960;
encoderParam.videoEncodedFPS = 15;
encoderParam.videoEncodedGOP = 2;
encoderParam.videoEncodedKbps = 1300;

TRTCStreamMixingConfig *config = [[TRTCStreamMixingConfig alloc] init];
if (self.mixUserList.count) {
    NSMutableArray<TRTCUser *> *userList = [NSMutableArray array];
    NSMutableArray<TRTCVideoLayout *> *layoutList = [NSMutableArray array];
    for (int i = 1; i < MIN(self.mixUserList.count, 16); i++) {
        TRTCUser *user = [[TRTCUser alloc] init];
        // The integer room number is intRoomId
        user.strRoomId = self.roomId;
        user.userId = self.mixUserList[i];
        [userList addObject:user];

        TRTCVideoLayout *layout = [[TRTCVideoLayout alloc] init];
        if ([self.mixUserList[i] isEqualToString:self.userId]) {
            // The layout for the anchor's video
            layout.rect = CGRectMake(0, 0, 540, 960);
            layout.zOrder = 0;
```

```
            } else {
                // The layout for the mic-connection audience's video
                layout.rect = CGRectMake(400, 5 + i * 245, 135, 240);
                layout.zOrder = 1;
            }
            layout.fixedVideoUser = user;
            layout.fixedVideoStreamType = TRTCVideoStreamTypeBig;
            [layoutList addObject:layout];
        }
        // Specify the information for each input audio stream in the transcoding s
        config.audioMixUserList = [userList copy];
        // Specify the information of position, size, layer, and stream type for ea
        config.videoLayoutList = [layoutList copy];
    }
    // Update the published media stream
    [self.trtcCloud updatePublishMediaStream:self.taskId publishTarget:target encod
}

// Event callback for updating the media stream
- (void)onUpdatePublishMediaStream:(NSString *)taskId code:(int)code message:(NSStr
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure
}
```

Note:

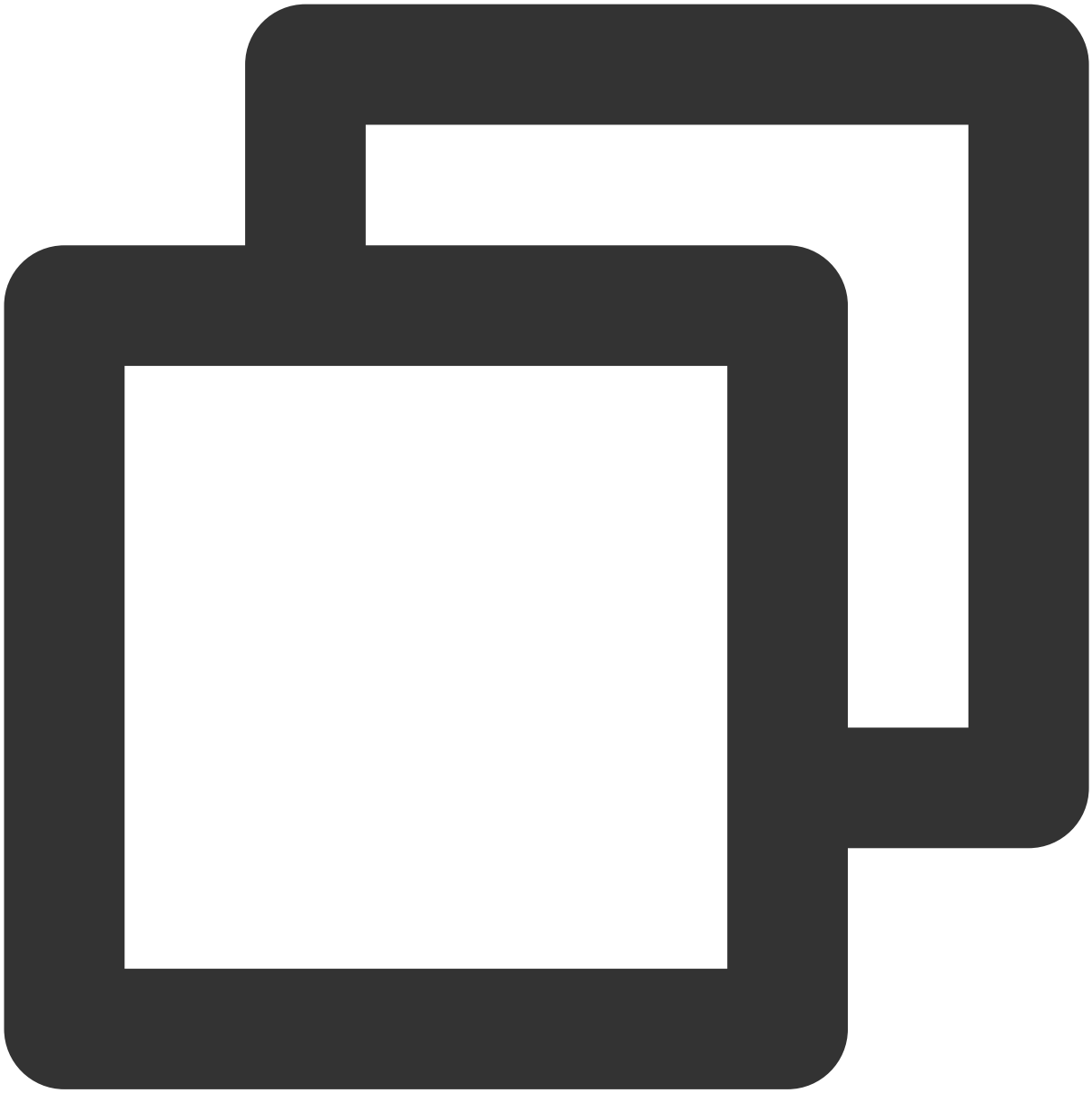
To ensure continuous CDN playback without stream disconnection, you need to keep the media stream encoding output parameter `encoderParam` and the stream name `streamName` unchanged.

Media stream encoding output parameters and mixed display layout parameters can be customized according to business needs. Currently, up to 16 channels of audio and video input are supported. If a user only provides audio, it will still be counted as one channel.

Switching between audio only, audio and video, and video only is not supported within the same task.

4. The off-streaming audience exit the room, and the anchor updates the mixed stream task.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the player callback listener
[self.livePlayer setObserver:self];
// The reusable TRTC video rendering control
[self.livePlayer setRenderView:self.remoteView];
// Restart playing CDN media stream
[self.livePlayer startLivePlay:flvUrl];


- (void)onVideoLoading:(id<V2TXLivePlayer>)player extraInfo:(NSDictionary *)extraIn
    // Video loading event
```

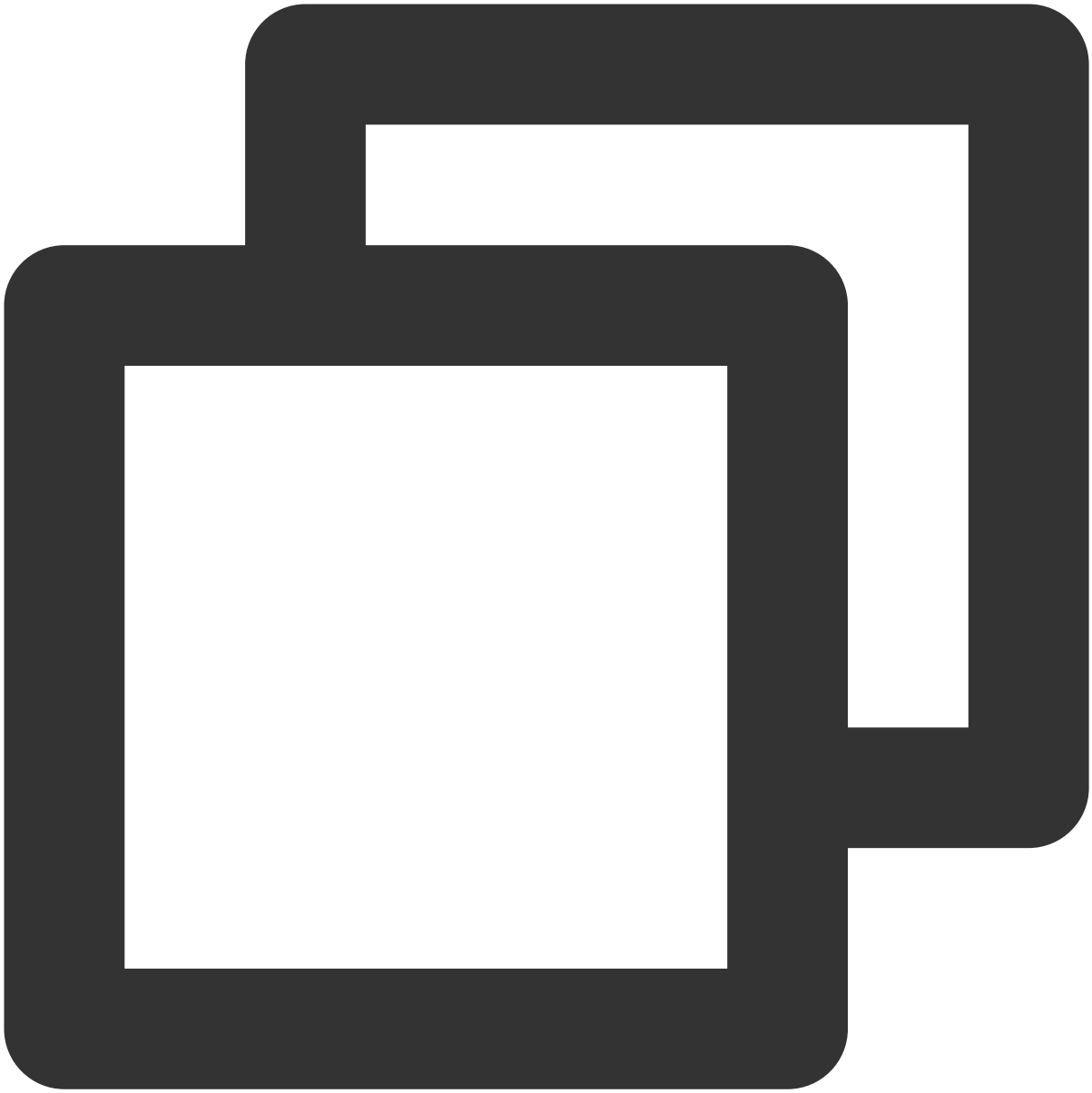
©2013-2022 Tencent Cloud. All rights reserved.

```
}

// Video playback event
- (void)onVideoPlaying:(id<V2TXLivePlayer>)player firstPlay:(BOOL)firstPlay extraIn
    if (firstPlay) {
        [self.trtcCloud stopAllRemoteView];
        [self.trtcCloud stopLocalAudio];
        [self.trtcCloud stopLocalPreview];
        [self.trtcCloud exitRoom];
    }
}
```

Note:

To avoid video interruptions when switching the stream puller, it is recommended to wait for the player's video playback event `onVideoPlaying` before exiting the TRTC room.

©2013-2022 Tencent Cloud. All rights reserved.

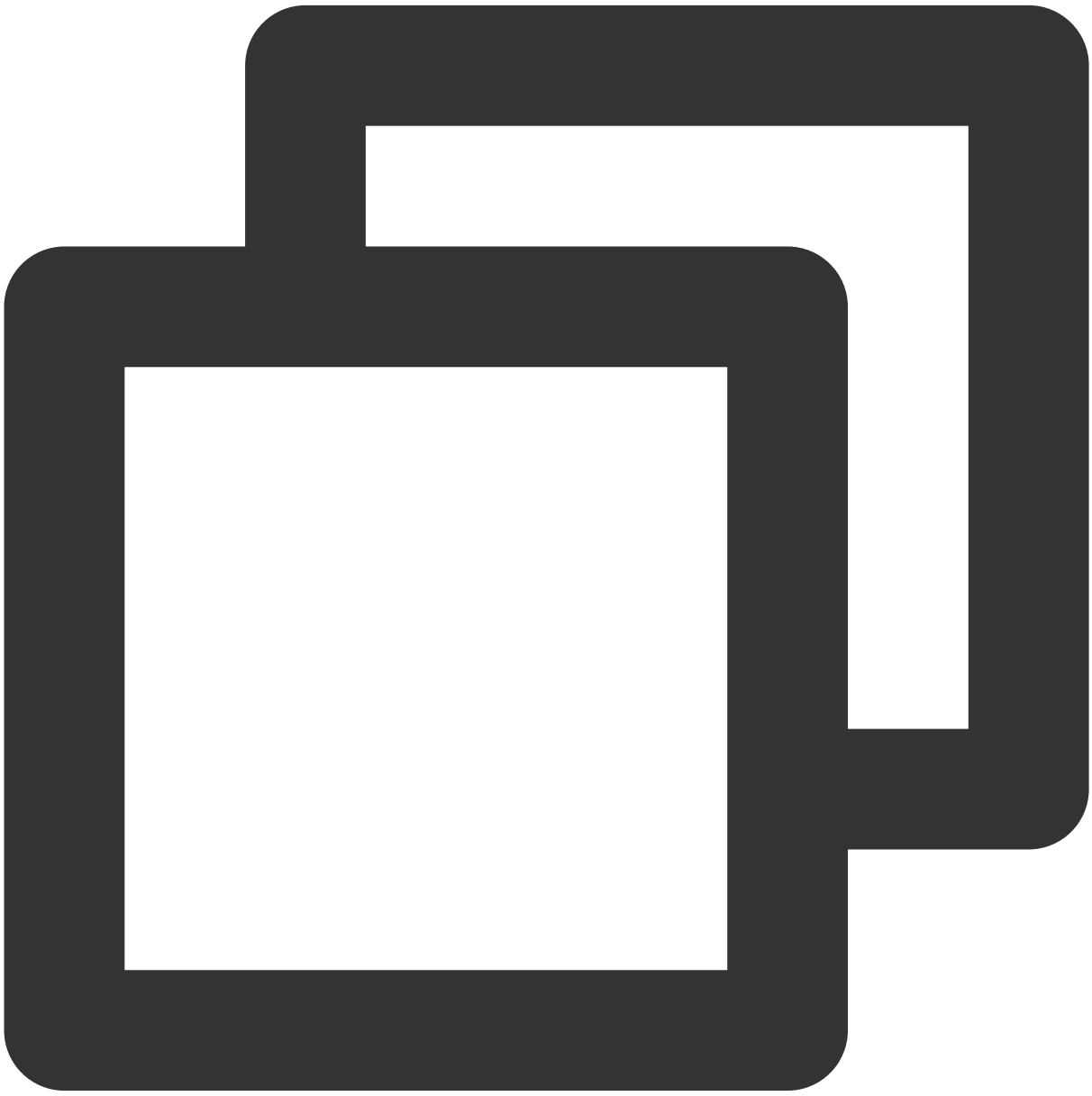


```
// Event callback for the mic-connection audience's room exit
- (void)onRemoteUserLeaveRoom:(NSString *)userId reason:(NSInteger)reason {
    if ([self.mixUserList containsObject:userId]) {
        [self.mixUserList removeObject:userId];
    }
    // The anchor updates the mixed stream task
    [self updatePublishMediaToCDN];
}

// Event callback for updating the media stream
- (void)onUpdatePublishMediaStream:(NSString *)taskId code:(int)code message:(NSStr
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // When you call the publish media stream API (updatePublishMediaStream), the t
    // code: Callback result. 0 means success and other values mean failure
}
```

Step 4: The anchor stops the live streaming and exits the room



```
- (void)exitRoom {
    // Stop all published media streams
    [self.trtcCloud stopPublishMediaStream:@""];
    [self.trtcCloud stopLocalAudio];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    [self.trtcCloud stopLocalPreview];
    [self.trtcCloud exitRoom];
}

// Event callback for stopping media streams
- (void)onStopPublishMediaStream:(NSString *)taskId code:(int)code message:(NSStrin
    // When you call stopPublishMediaStream, the taskId you provide will be returne
    // code: Callback result. 0 means success and other values mean failure
}

// Event callback for exiting the room
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Proactively call exitRoom to exit the room");
    } else if (reason == 1) {
        NSLog(@"Removed from the current room by the server");
    } else if (reason == 2) {
        NSLog(@"The current room is dissolved");
    }
}
```

Note:

To stop publishing media streams, enter an empty string for `taskId` . This will stop all the media streams you have published.

After all resources occupied by the SDK are released, the SDK will throw the `onExitRoom` callback notification to inform you.

Advanced Features

Product Information Pop-up

The Product Information Pop-up feature can be implemented through IM Custom Message or SEI Information. Below are the specific information of the two implementation methods.

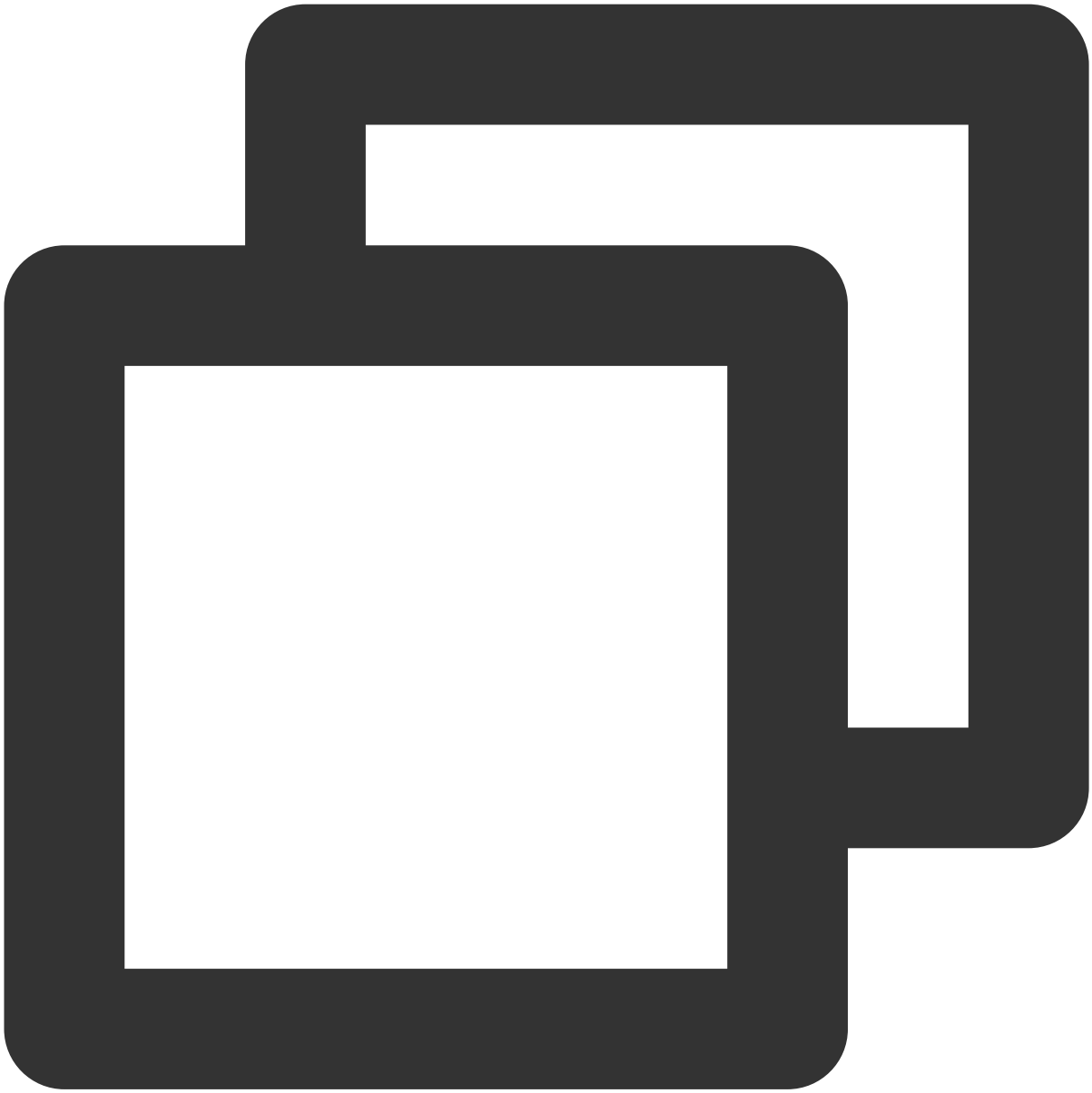
Custom Message

Custom messages depend on Instant Messaging (IM). You need to activate the service and import the IM SDK in advance. For detailed guidelines, see Voice Chat Room Connection Guide - Connection Preparation.

1. Send Custom Messages

Method 1: The anchor sends product pop-up related custom group messages on the client.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Construct product pop-up message body
NSDictionary *msgDict = @{
    @"itemNumber": @1,    // Item number
    @"itemPrice": @199.0,// Item price
    @"itemTitle": @"xxx",// Item title
    @"itemUrl": @"xxx" // Item URL
};
NSDictionary *dataDict = @{
    @"cmd": @"item_popup_msg",
    @"msg": msgDict
};
```

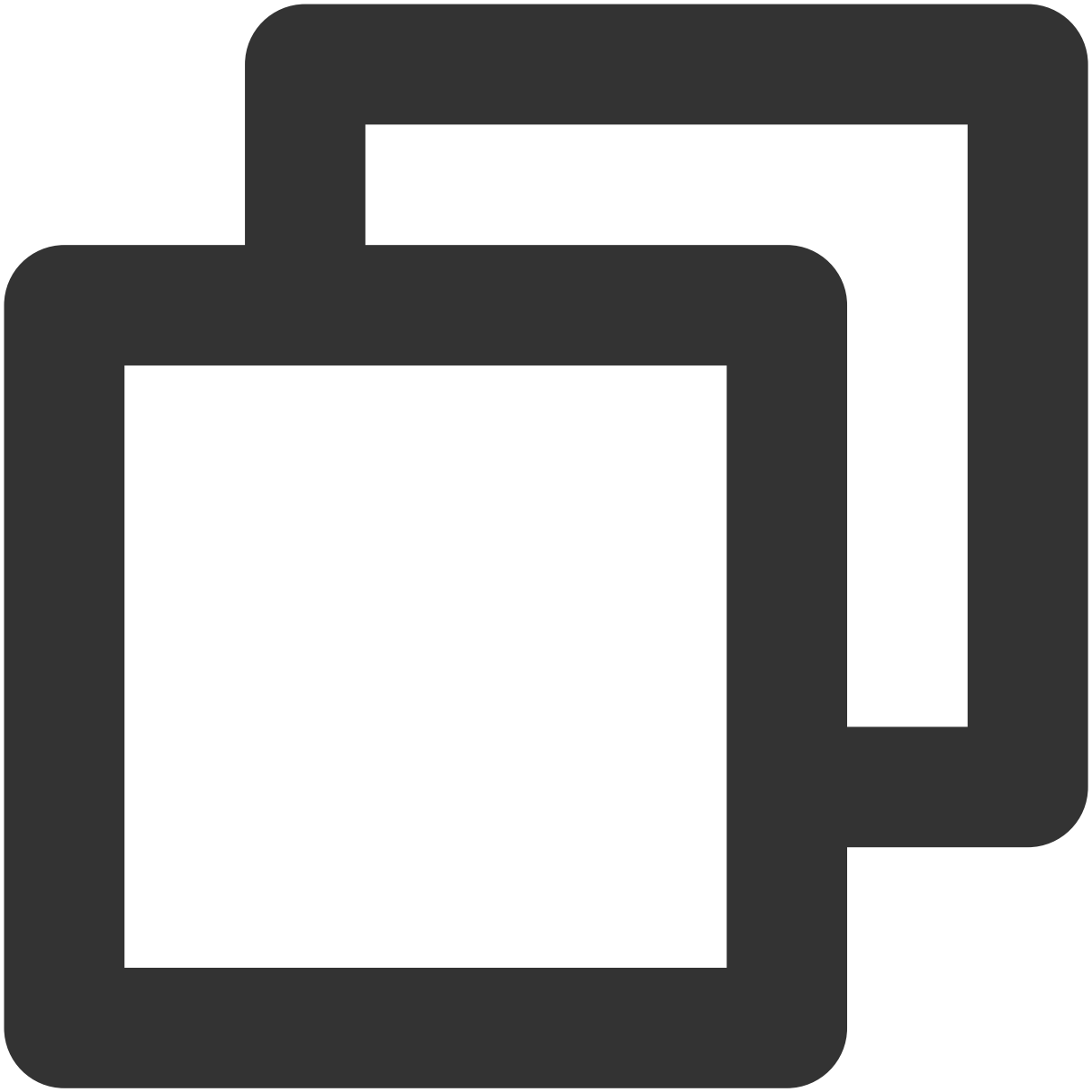
©2013-2022 Tencent Cloud. All rights reserved.

```
NSError *error;
NSData *data = [NSJSONSerialization dataWithJSONObject:dataDict options:0 error:&er

// Send custom group messages (it is recommended that product pop-up messages shoul
[[V2TIMManager sharedInstance] sendGroupCustomMessage:data to:groupID priority:V2TI
    // Successfully sent product pop-up message
    // Locally rendering of product pop-up effect
} fail:^(int code, NSString *desc) {
    // Failed to send product pop-up message
}];
```

Method 2: The backend operators sends product pop-up related custom group messages on the server.

Request URL sample:

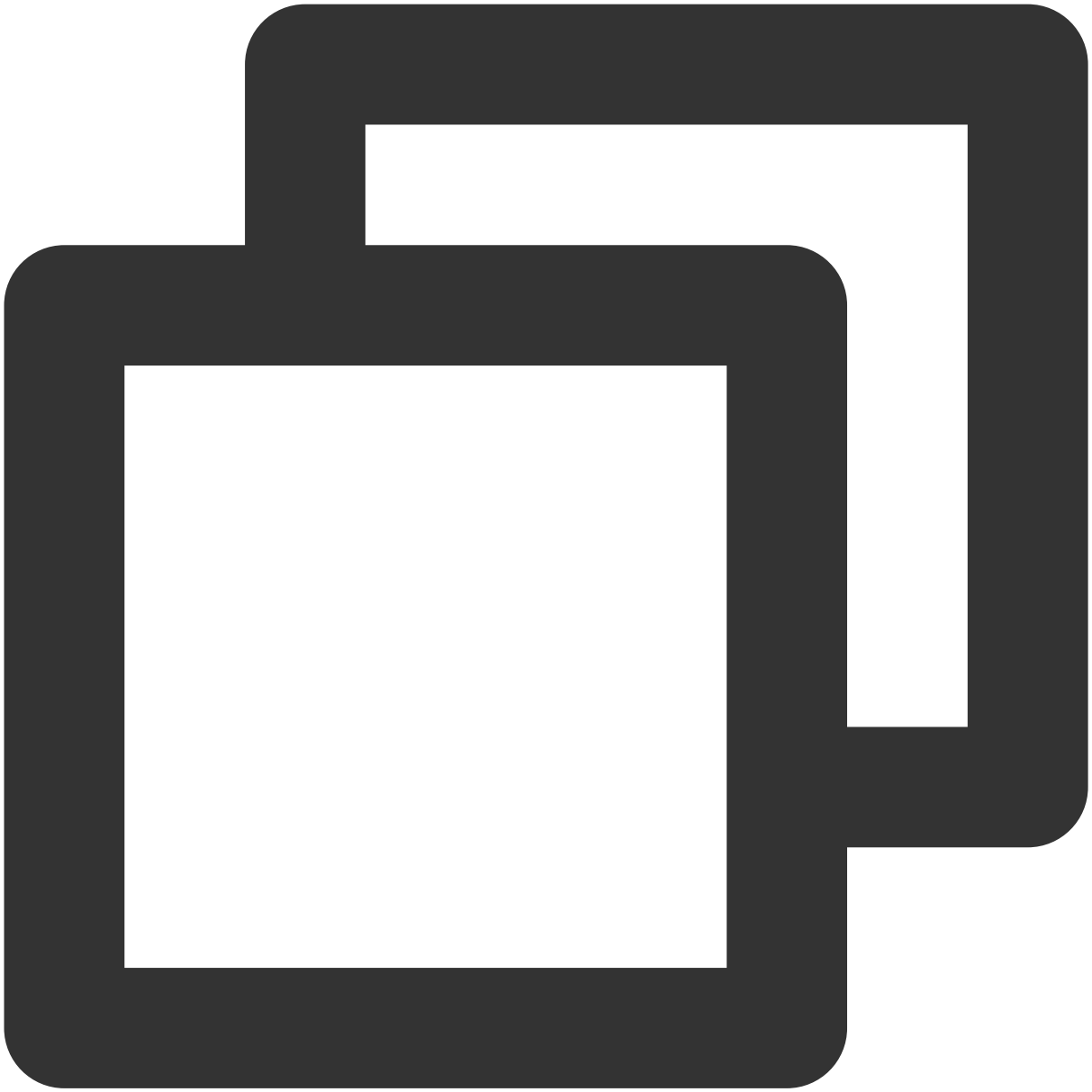
©2013-2022 Tencent Cloud. All rights reserved.



```
https://xxxxxx/v4/group_open_http_svc/send_group_msg?sdkappid=88888888&identifier=a
```

Request packet body sample:

©2013-2022 Tencent Cloud. All rights reserved.



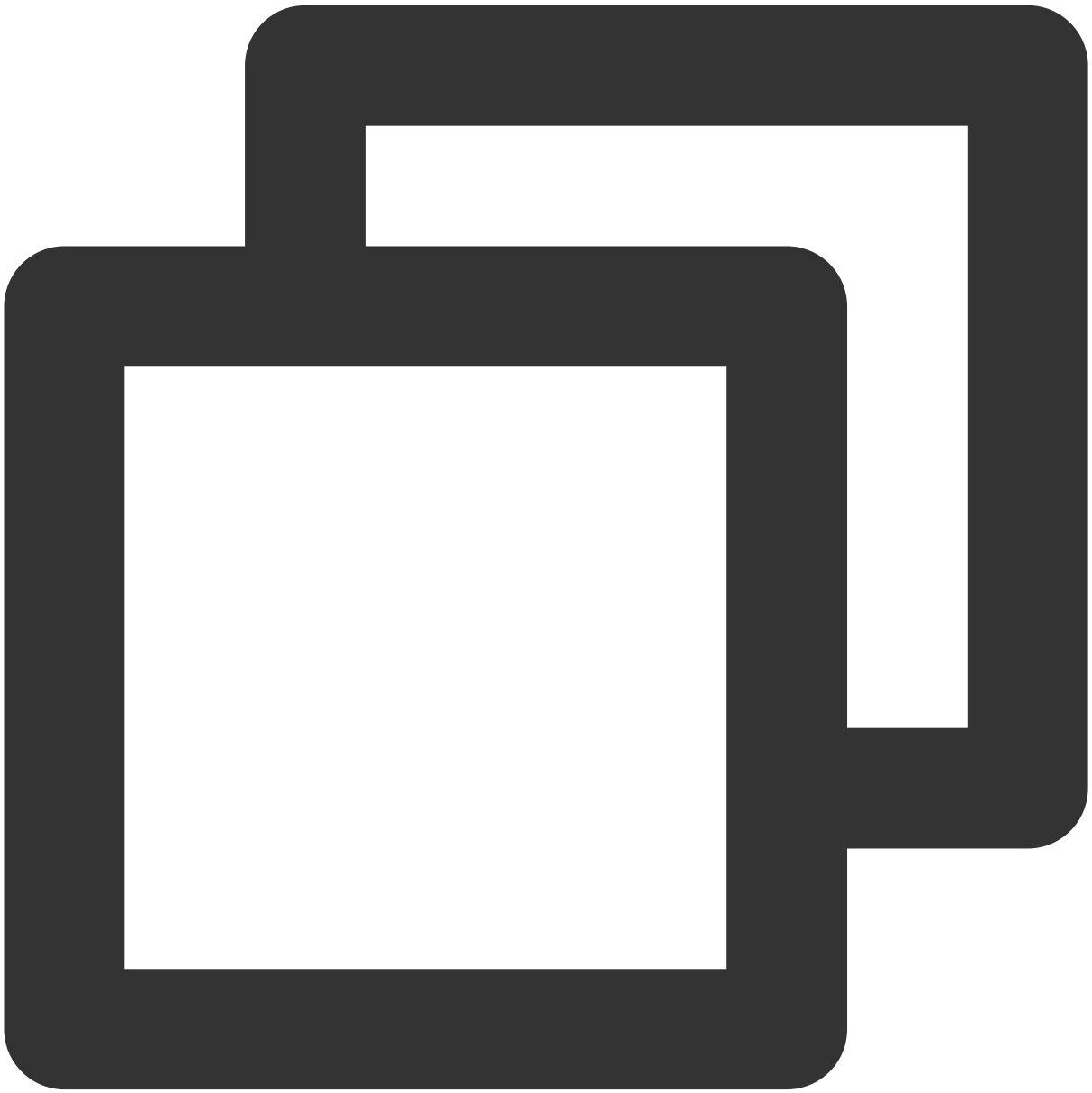
```
{
    "GroupId": "@TGS#12DEVUDHQ",
    "Random": 2784275388,
    "MsgPriority": "High",  // The priority of the message. It is recommended to se
    "MsgBody": [
        {
            "MsgType": "TIMCustomElem",
            "MsgContent": {
                // itemNumber: item number; itemPrice: item price; itemTitle: item
                "Data": "{\\"cmd\\": \\"item_popup_msg\\", \\"msg\\": {\\"itemNumbe
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
    ]
}
```

2. Receive Custom Messages

Other users in the room receive callback for custom group messages, then proceed with message parsing and product pop-up effect rendering.



```
// Custom group messages received
[[V2TIMManager sharedInstance] addSimpleMsgListener:self];
- (void)onRecvGroupCustomMessage:(NSString *)msgID groupID:(NSString *)groupID send
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    if (data.length > 0) {
        NSError *error;
        NSDictionary *dataDict = [NSJSONSerialization JSONObjectWithData:data optio
        if (!error) {
            NSString *command = dataDict[@"cmd"];
            NSDictionary *msgDict = dataDict[@"msg"];
            if ([command isEqualToString:@"item_popup_msg"]) {
                NSNumber *itemNumber = msgDict[@"itemNumber"];// Item number
                NSNumber *itemPrice = msgDict[@"itemPrice"];  // Item price
                NSString *itemTitle = msgDict[@"itemTitle"];  // Item title
                NSString *itemUrl = msgDict[@"itemUrl"];  // Item URL
                // Render product pop-up effect based on item number, item price, i
            }
        } else {
            NSLog(@"Parsing error: %@", error.localizedDescription);
        }
    }
}
```

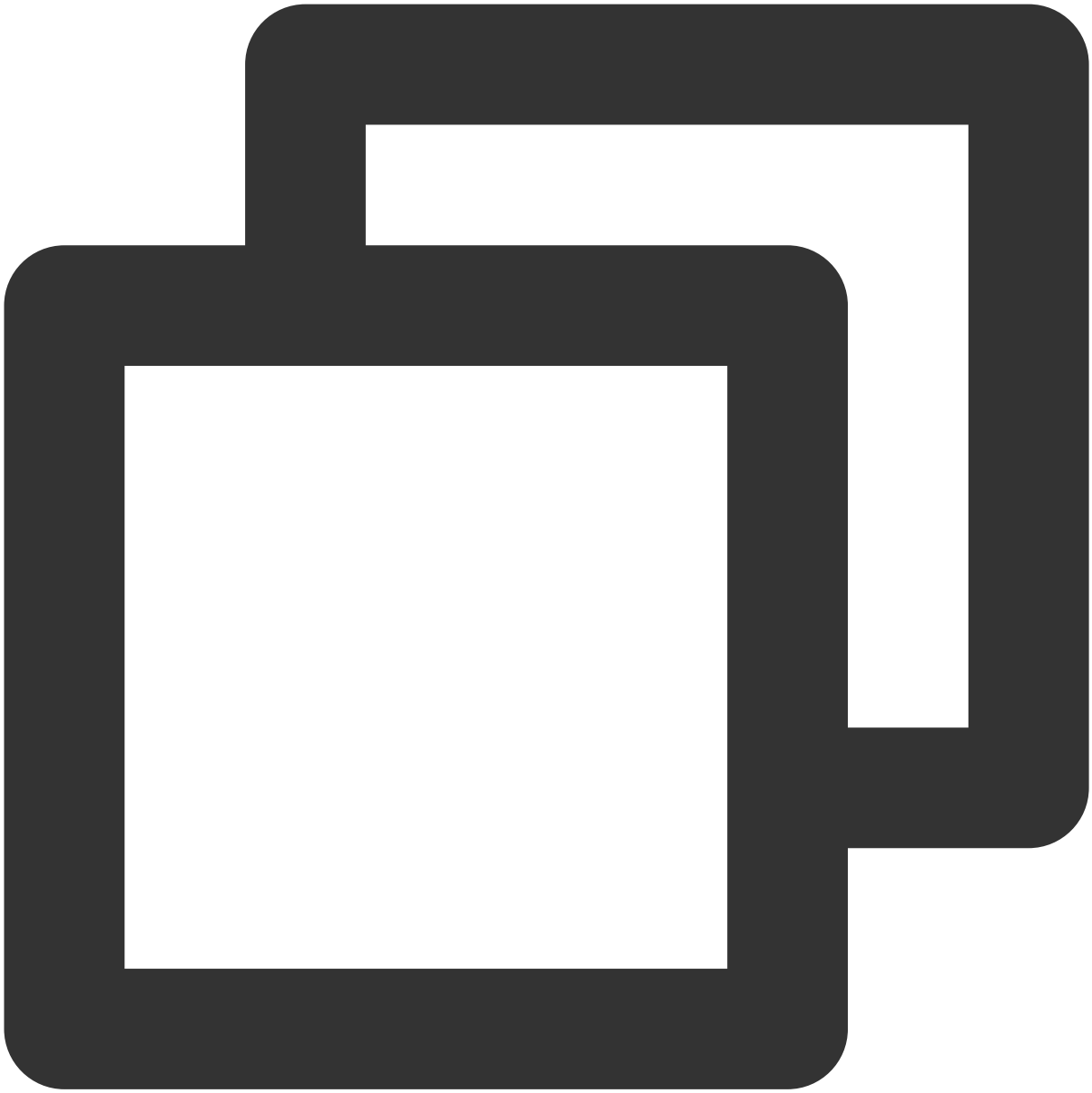
SEI Information

SEI information will be inserted into the anchor's video stream for transmission, achieving precise sync between the product information pop-up and the anchor's live streaming.

1. Send SEI Information

The anchor sends SEI messages related to product pop-up on the TRTC client.

©2013-2022 Tencent Cloud. All rights reserved.



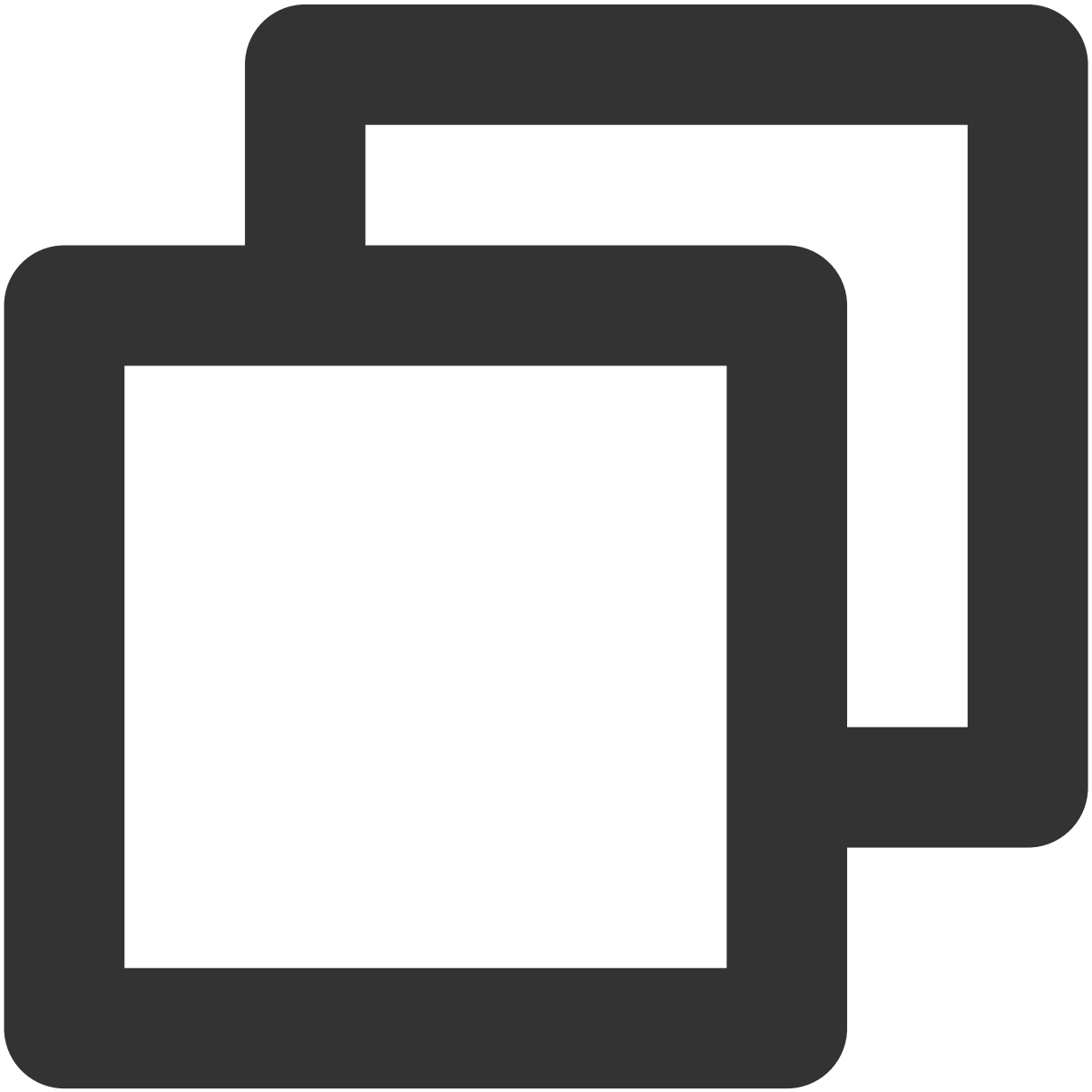
```
// Construct product pop-up message body
NSDictionary *msgDict = @{
    @"itemNumber": @1,    // Item number
    @"itemPrice": @199.0,// Item price
    @"itemTitle": @"xxx",// Item title
    @"itemUrl": @"xxx" // Item URL
};
NSDictionary *dataDict = @{
    @"cmd": @"item_popup_msg",
    @"msg": msgDict
};
```

©2013-2022 Tencent Cloud. All rights reserved.

```
NSError *error;
NSData *data = [NSJSONSerialization dataWithJSONObject:dataDict options:0 error:&er

// Send SEI information
[self.trtcCloud sendSEIMsg:data repeatCount:1];
```

2. Receive SEI Information

Method 1: The audience receives SEI messages on the TRTC client, then proceeds with message parsing and product pop-up effect rendering.



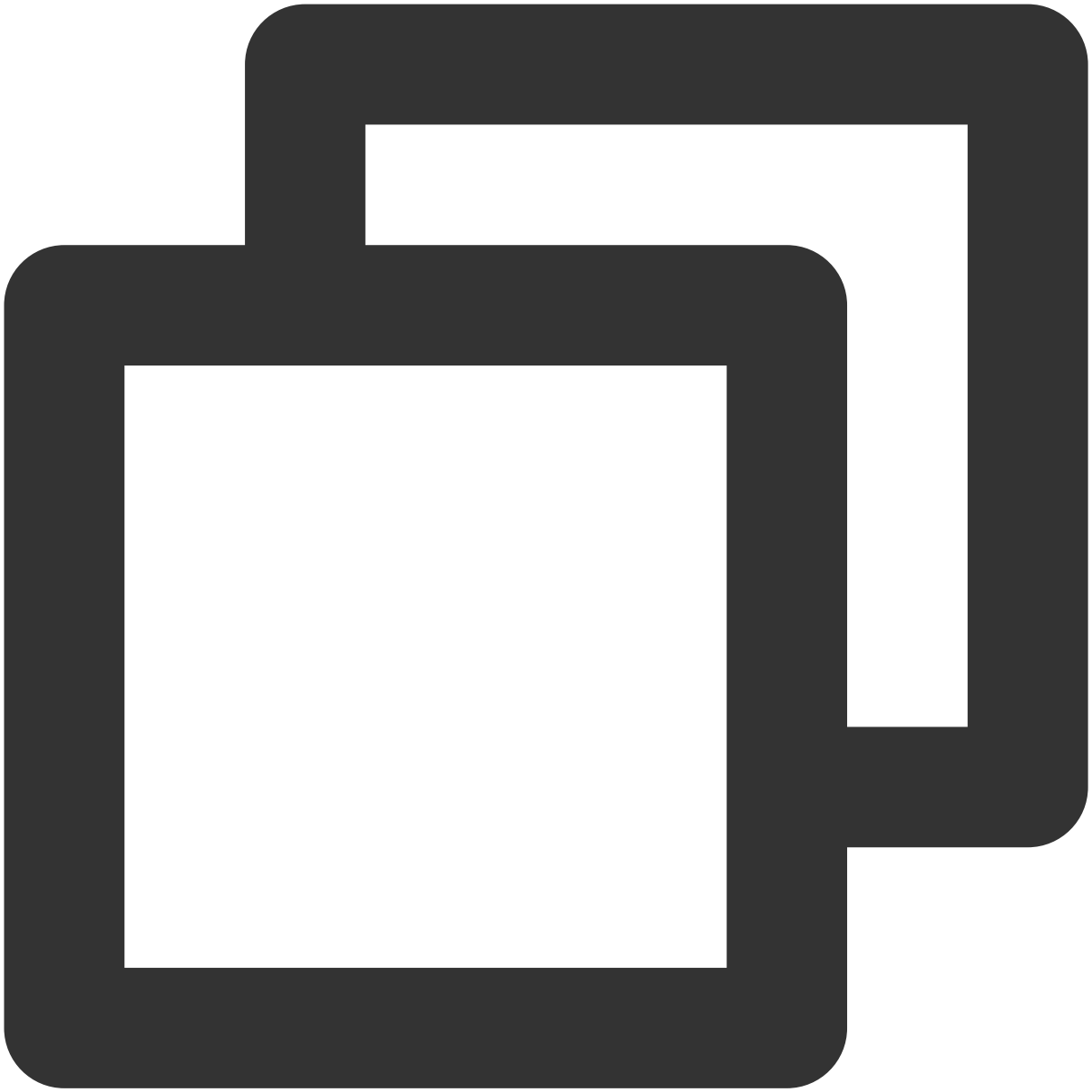
```
// Set TRTC event listener
```

©2013-2022 Tencent Cloud. All rights reserved.

```
self.trtcCloud.delegate = self;

// Receive SEI messages
- (void)onRecvSEIMsg:(NSString *)userId message:(NSData *)message {
    if (message.length > 0) {
        NSError *error;
        NSDictionary *dataDict = [NSJSONSerialization JSONObjectWithData:message op
        if (!error) {
            NSString *command = dataDict[@"cmd"];
            NSDictionary *msgDict = dataDict[@"msg"];
            if ([command isEqualToString:@"item_popup_msg"]) {
                NSNumber *itemNumber = msgDict[@"itemNumber"];// Item number
                NSNumber *itemPrice = msgDict[@"itemPrice"];  // Item price
                NSString *itemTitle = msgDict[@"itemTitle"];  // Item title
                NSString *itemUrl = msgDict[@"itemUrl"];  // Item URL
                // Render product pop-up effect based on item number, item price, i
            }
        } else {
            NSLog(@"Parsing error: %@", error.localizedDescription);
        }
    }
}
```

Method 2: The audience receives SEI messages on the CDN stream player, then proceeds with message parsing and product pop-up effect rendering.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the PayloadType for sending SEI messages in TRTC (must be set before sending
[self.trtcCloud callExperimentalAPI:@"{\\"api\\":\\"setSEIPayloadType\\",\\"params\

// Enable receiving SEI messages on the player and set the PayloadType
[self.livePlayer enableReceiveSeiMessage:YES payloadType:5];

// SEI message callback and parsing
- (void)onReceiveSeiMessage:(id<V2TXLivePlayer>)player payloadType:(int)payloadType
    if (data.length > 0) {
        NSError *error;
        NSDictionary *dataDict = [NSJSONSerialization JSONObjectWithData:data optio
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        if (!error) {
            NSString *command = dataDict[@"cmd"];
            NSDictionary *msgDict = dataDict[@"msg"];
            if ([command isEqualToString:@"item_popup_msg"]) {
                NSNumber *itemNumber = msgDict[@"itemNumber"];// Item number
                NSNumber *itemPrice = msgDict[@"itemPrice"];  // Item price
                NSString *itemTitle = msgDict[@"itemTitle"];  // Item title
                NSString *itemUrl = msgDict[@"itemUrl"];  // Item URL
                // Render product pop-up effect based on item number, item price, i
            }
        } else {
            NSLog(@"Parsing error: %@", error.localizedDescription);
        }
    }
}
```

Note:

It is necessary to ensure that the SEI `PayloadType` of the TRTC sender and the player receiver are consistent, so that the audience can successfully receive the SEI messages relayed via TRTC.

Product Explanation Replay

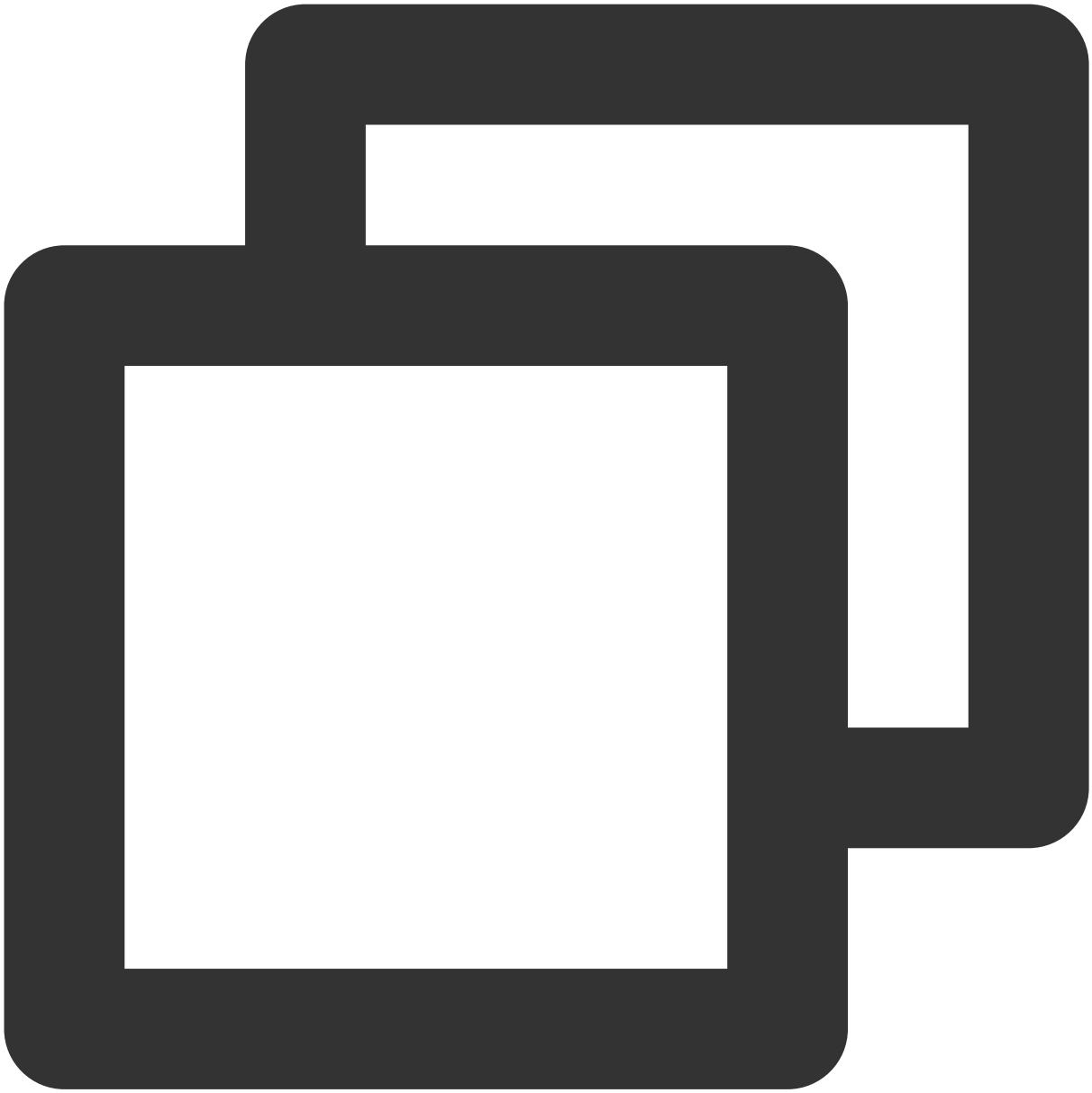
By playing pre-recorded product explanation videos, the product explanation replay feature is implemented.

First, it is necessary to initialize the player, then start playing the recorded video. TXVodPlayer supports two playback modes, which you can choose according to your needs:

Using the URL method

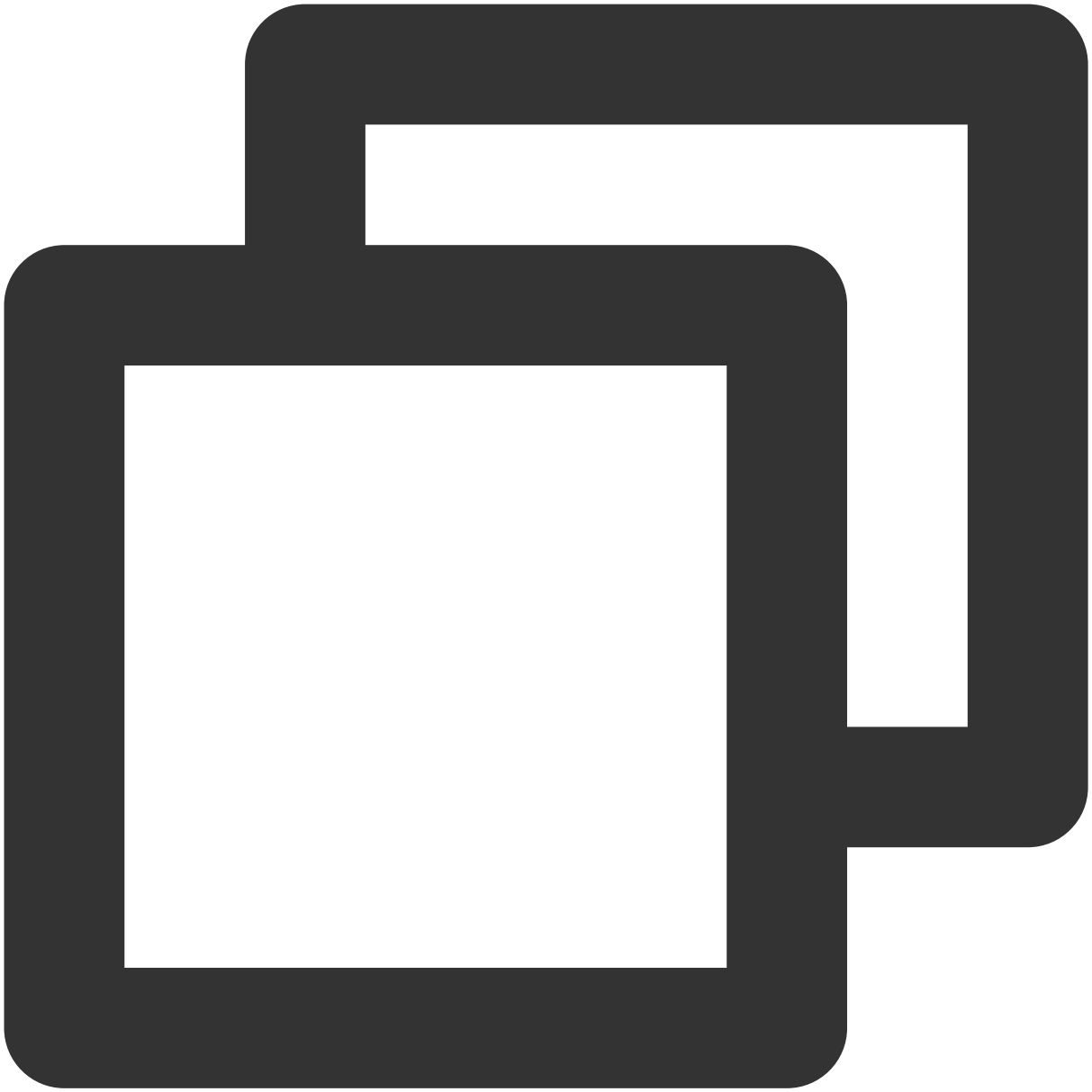
Using the FileId method

©2013-2022 Tencent Cloud. All rights reserved.



```
// Play URL video resource
NSString* url = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
[_txVodPlayer startVodPlay:url];

// Play sandbox local video resources
// Obtain the Documents path
NSString *documentPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
// Obtain the local video path
NSString *videoPath = [NSString stringWithFormat:@"%@/video1.m3u8",documentPath];
[_txVodPlayer startVodPlay:videoPath];
```

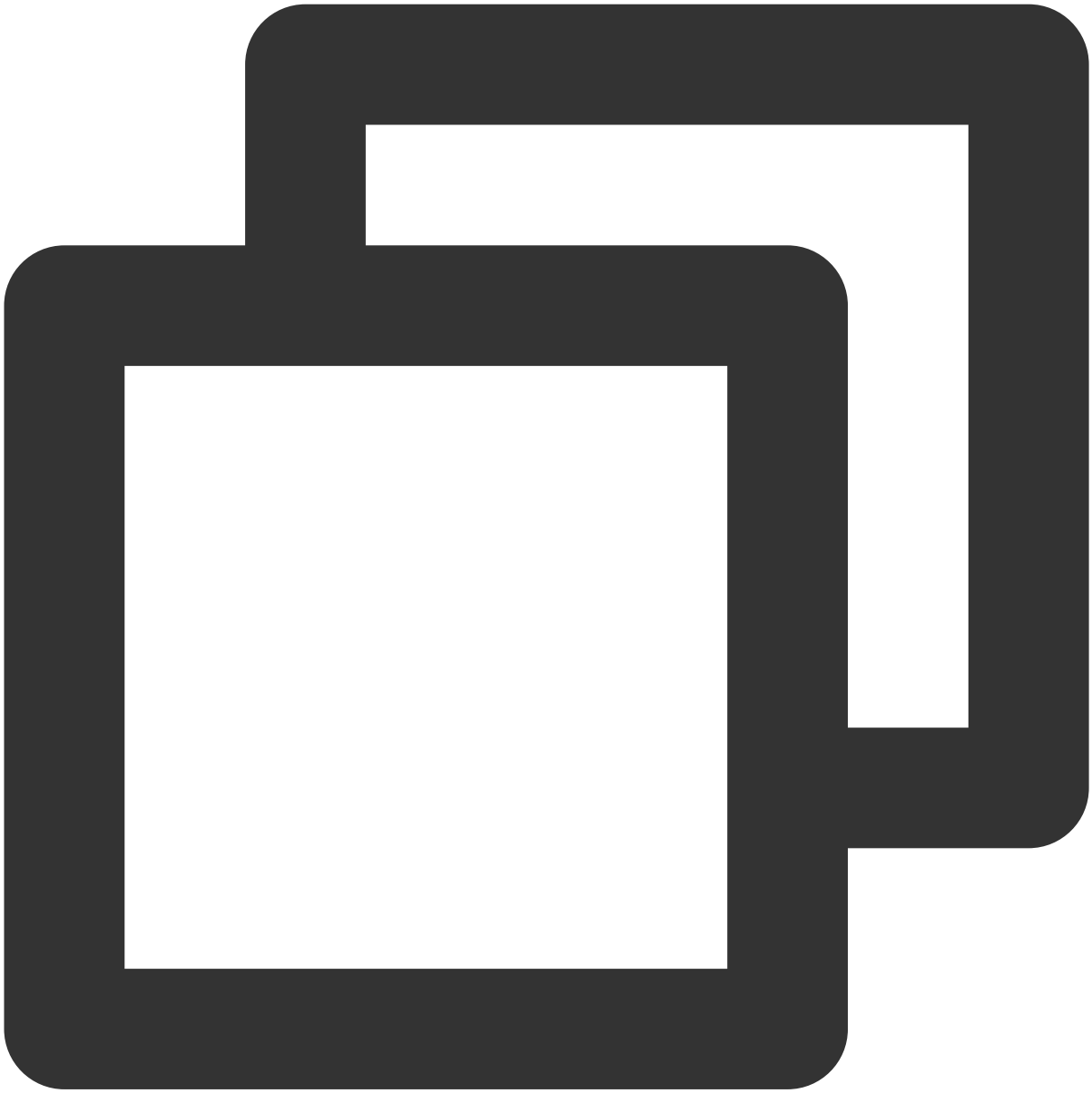
©2013-2022 Tencent Cloud. All rights reserved.



```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788;
p.fileId = @"4564972819220421305";
// The psign means player signature. For more information about the signature and h
p.sign = @"psignxxxx"; // Player signature
[_txVodPlayer startVodPlayWithParams:p];
```

Playback control: adjust the progress, pause playback, resume playback, and end playback.

©2013-2022 Tencent Cloud. All rights reserved.

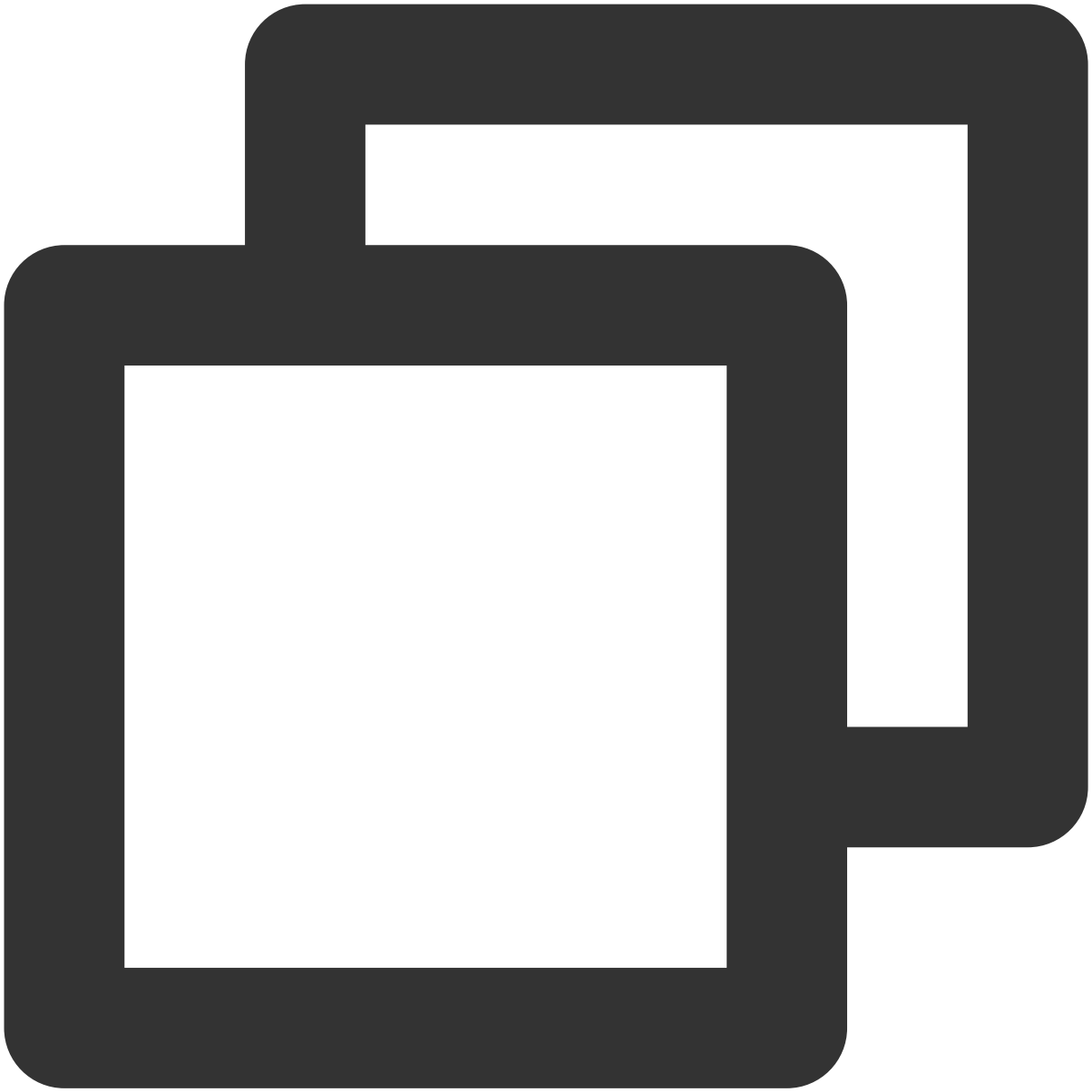


```
// Adjust the progress (seconds)
[_txVodPlayer seek:time];

// Pause playback
[_txVodPlayer pause];

// Resume playback
[_txVodPlayer resume];

// End playback
[_txVodPlayer stopPlay];
```

©2013-2022 Tencent Cloud. All rights reserved.

Note:

When stopping playback, remember to use `removeVideoWidget` to Destroy the view control before exiting the current UI interface. Otherwise, it may cause a memory leak or screen flash.

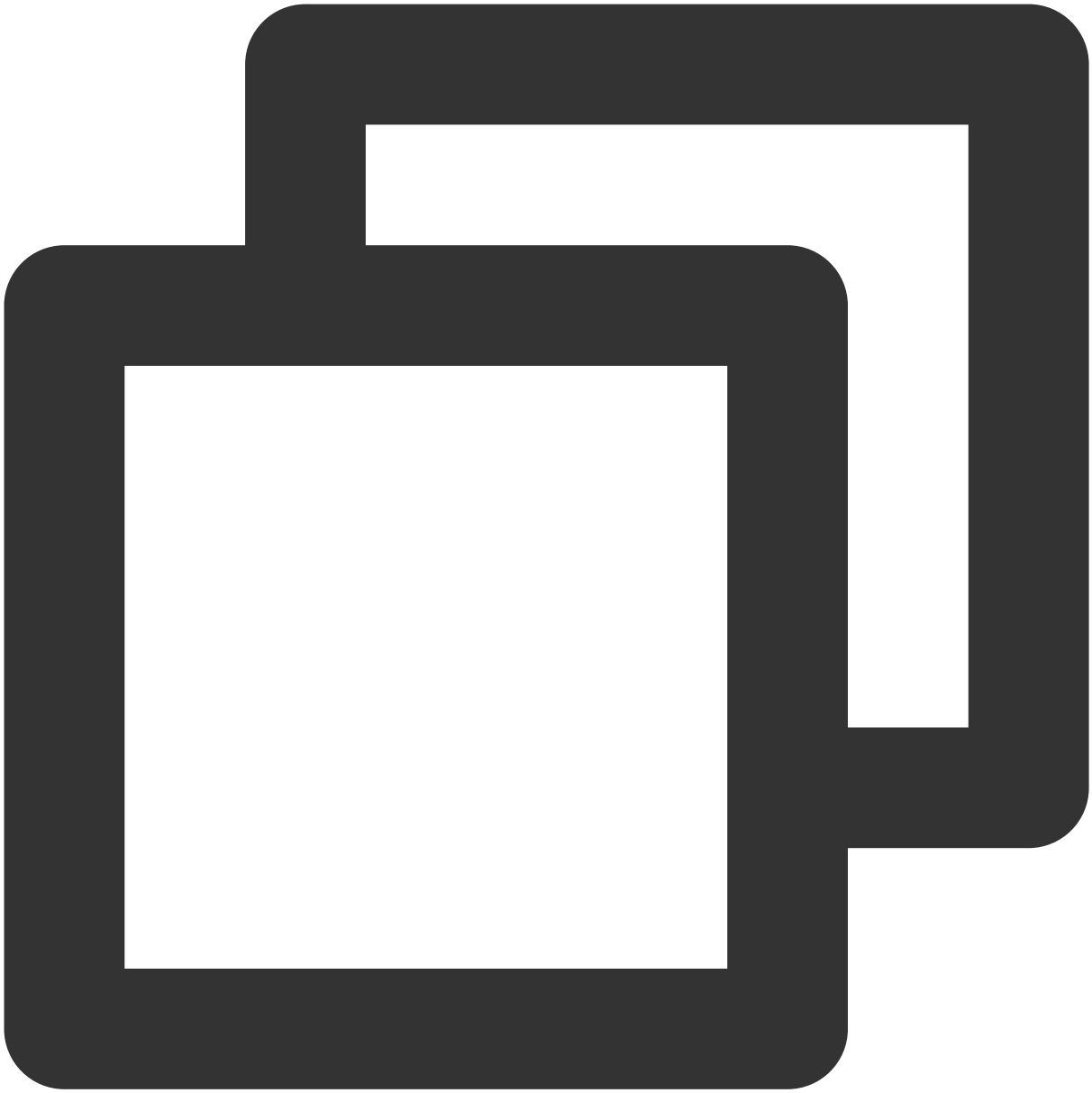


```
// Destroy the view control
[_txVodPlayer removeVideoWidget];
```

Cross-room Mic-connection PK

1. Either party initiates the cross-room mic-connection PK.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)connectOtherRoom:(NSString *)roomId {
    NSMutableDictionary *jsonDict = [[NSMutableDictionary alloc] init];
    // The digit room number is roomId
    [jsonDict setObject:roomId forKey:@"strRoomId"];
    [jsonDict setObject:self.userId forKey:@"userId"];
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:NSJ
    NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8S
    [self.trtcCloud connectOtherRoom:jsonString];
}

// Result callback for requesting cross-room mic-connection
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)onConnectOtherRoom:(NSString *)userId errCode:(TXLiteAVError)errCode errMsg
    // The user ID of the anchor in the other room you want to initiate the cross-r
    // Error code. ERR_NULL indicates the request is successful
    // Error message
}
```

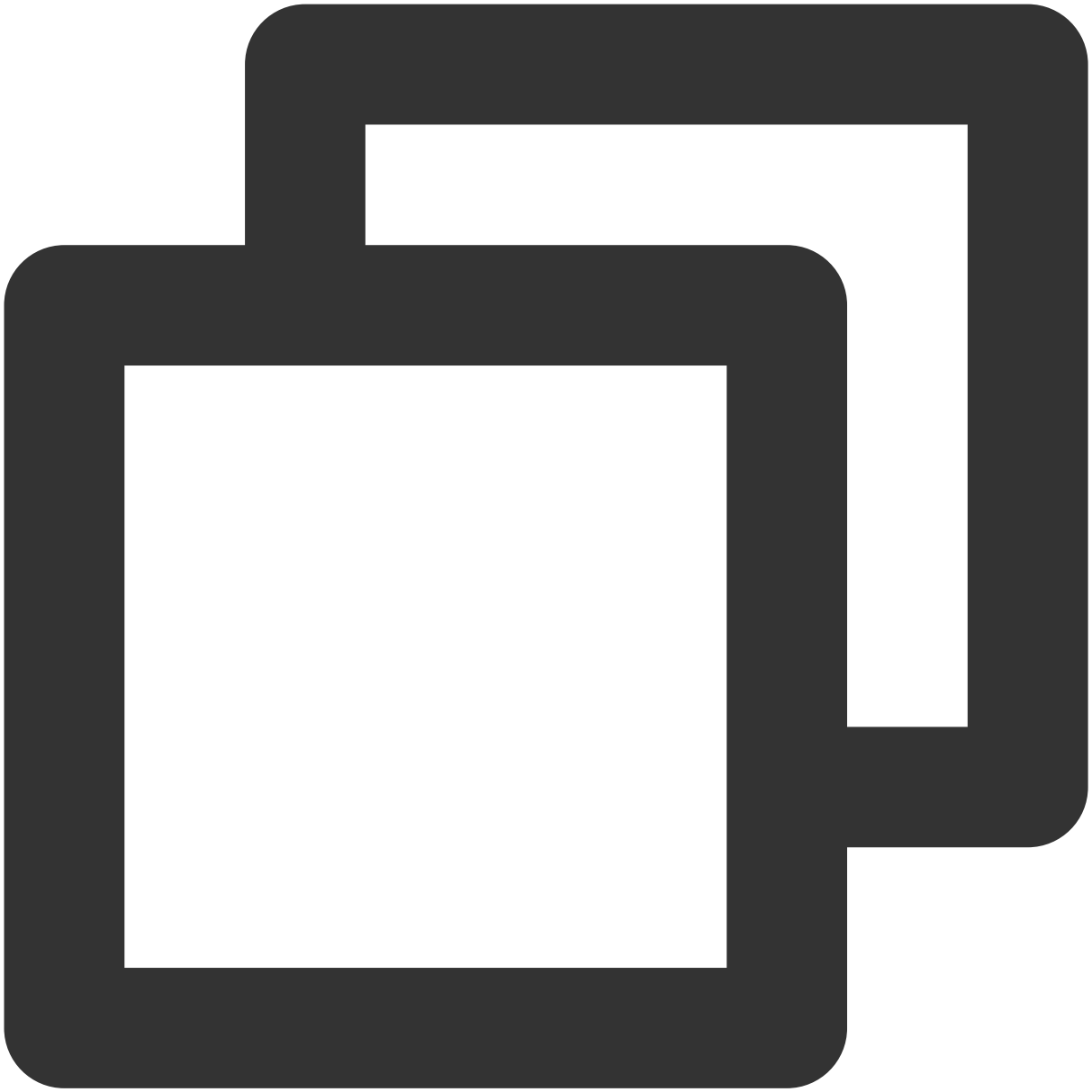
Note:

Both local and remote users participating in the cross-room mic-connection must be in the anchor role and must have audio/video uplink capabilities.

Cross-room mic-connection PK with multiple room anchors can be achieved by calling `ConnectOtherRoom()` multiple times. Currently, a room can connect with up to three other room anchors at most, and up to 10 anchors in a room can conduct cross-room mic-connection competition with anchors in other rooms.

2. All users in both rooms will receive a callback indicating that the audio and video streams from the PK anchor in the other room are available.

©2013-2022 Tencent Cloud. All rights reserved.

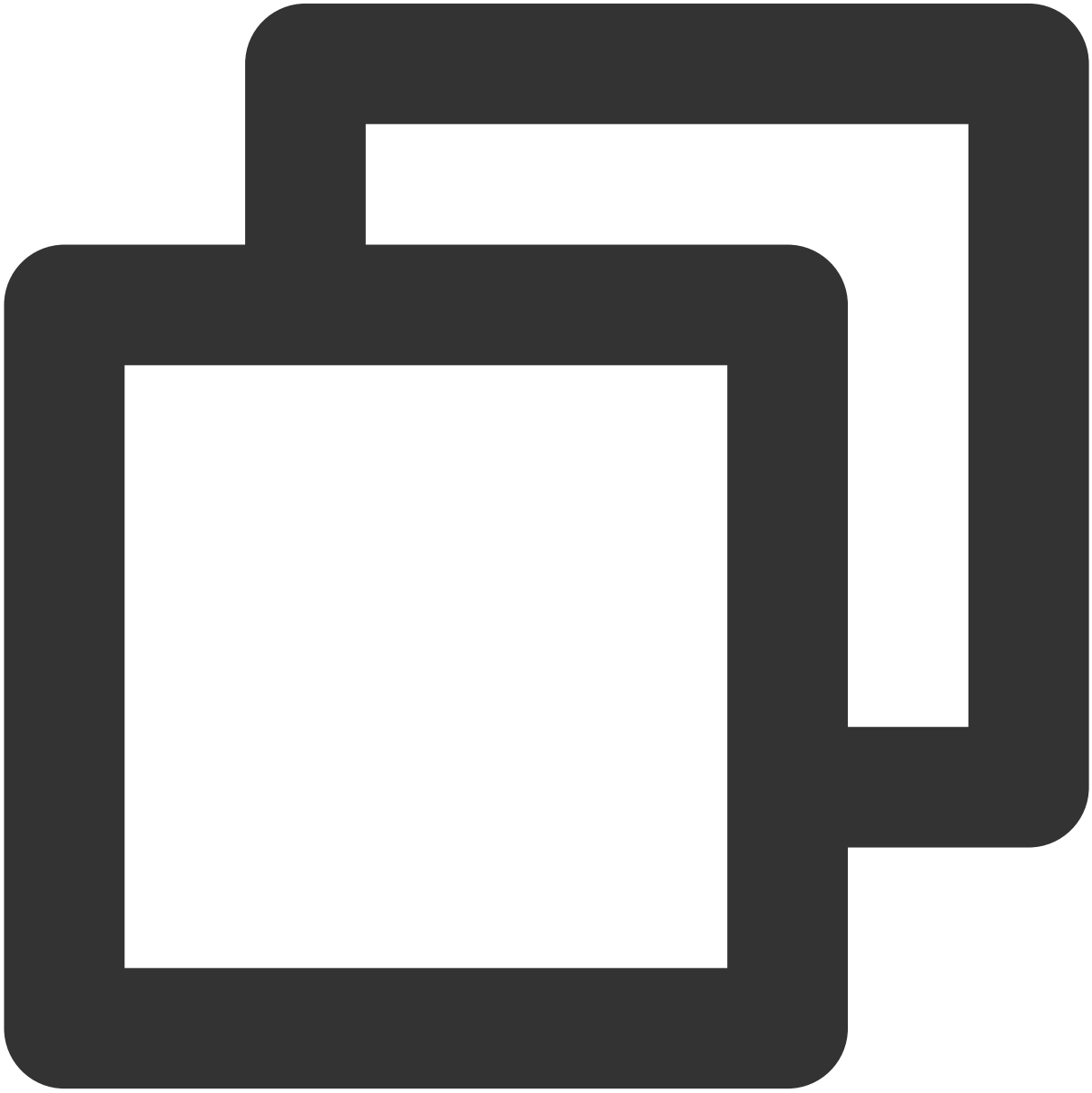


```
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes their audio
    // Under the automatic subscription mode, you do not need to do anything. The S
}

- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}
```

3. Either party exits the cross-room mic-connection PK.



```
// Exiting cross-room mic-connection
[self.trtcCloud disconnectOtherRoom];

// Result callback for exiting cross-room mic-connection
- (void)onDisconnectOtherRoom:(TXLiteAVError)errCode errMsg:(NSString *)errMsg {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Note:

After calling `DisconnectOtherRoom()` , you may exit the cross-room mic-connection PK with all other room anchors.

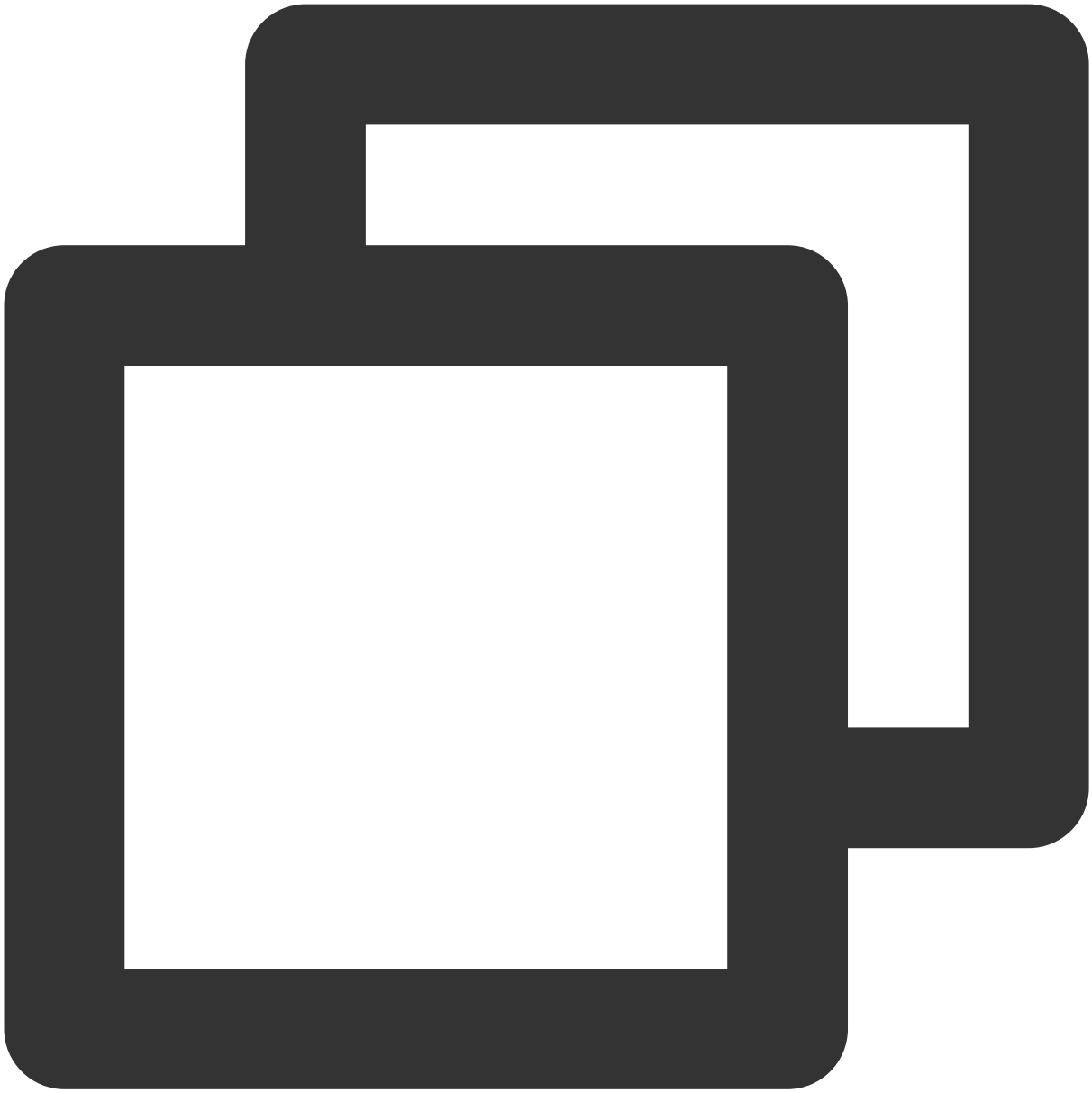
Either the initiator or the receiver can call `DisconnectOtherRoom()` to exit the cross-room mic-connection PK.

Third-Party Beauty Feature Integration

TRTC supports integrating third-party beauty effect products. Use the example of Special Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate the Special Effect SDK, and apply for an authorization license. For details, see Integration Preparation for steps.
2. Set the SDK material resource path (if any).

©2013-2022 Tencent Cloud. All rights reserved.



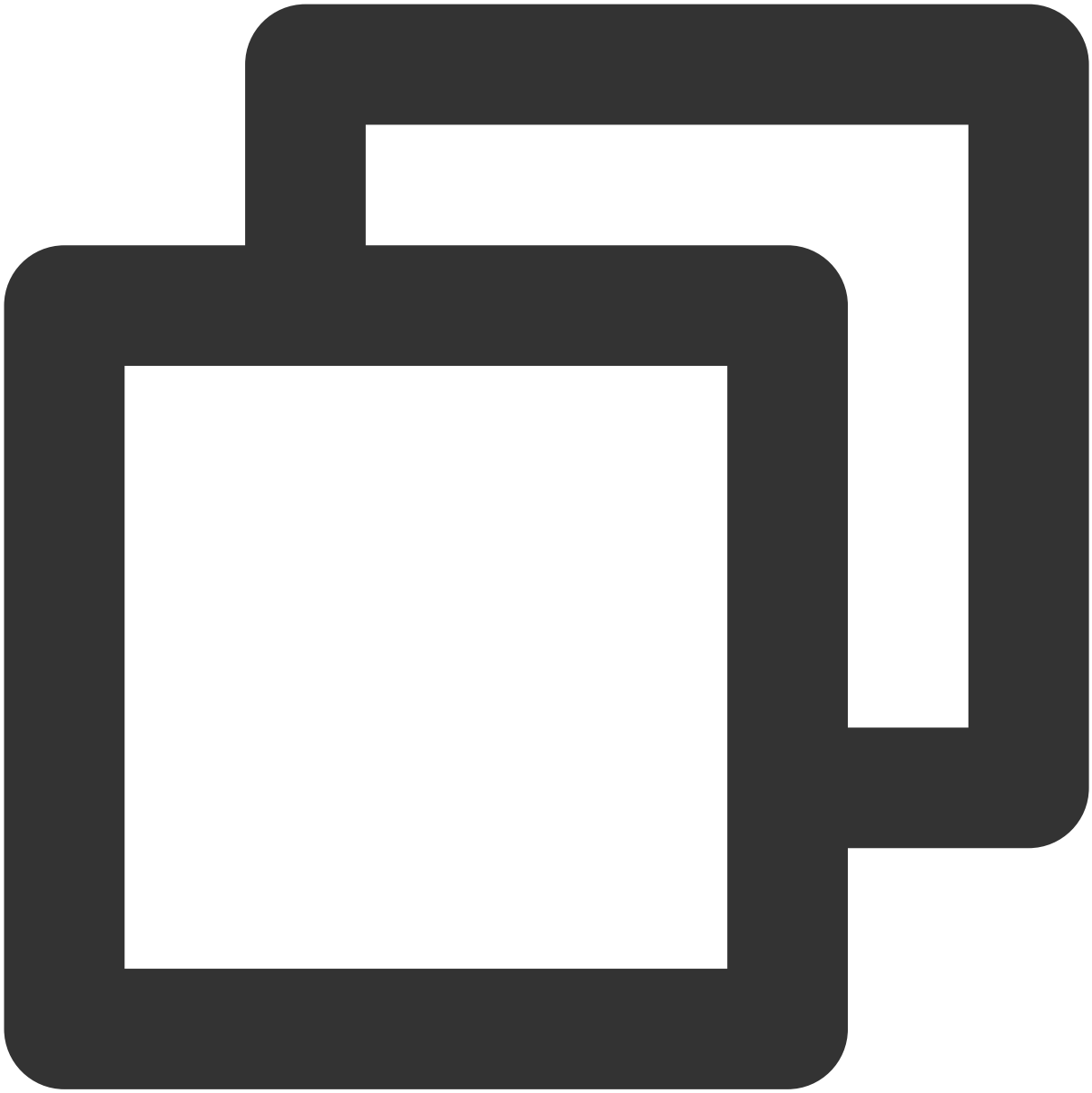
```
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                                       options:NSJSONReadingMutableContainers
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Initialize SDK: Width and height are the width and height of the texture respect
self.xMagicKit = [[XMagic alloc] initWithRenderSize:CGSizeMake(width,height) assets
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the video data callback for third-party beauty features in the TRTC SDK
[self.trtcCloud setLocalVideoProcessDelegete:self pixelFormat:TRTCVideoPixelFormat_

#pragma mark - TRTCVideoFrameDelegate

// Construct the YTProcessInput and pass it into the SDK for rendering processing
- (uint32_t)onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVi
    if (!self.xMagicKit) {
        [self buildBeautySDK:srcFrame.width and:srcFrame.height texture:srcFrame.te
        self.heightF = srcFrame.height;
        self.widthF = srcFrame.width;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    if(self.xMagicKit!=nil && (self.heightF!=srcFrame.height || self.widthF!=srcFra
       self.heightF = srcFrame.height;
       self.widthF = srcFrame.width;
       [self.xMagicKit setRenderSize:CGSizeMake(srcFrame.width, srcFrame.height)];
    }
    YTProcessInput *input = [[YTProcessInput alloc] init];
    input.textureData = [[YTTextureData alloc] init];
    input.textureData.texture = srcFrame.textureId;
    input.textureData.textureWidth = srcFrame.width;
    input.textureData.textureHeight = srcFrame.height;
    input.dataType = kYTTextureData;
    YTProcessOutput *output = [self.xMagicKit process:input withOrigin:YtLightImage
    dstFrame.textureId = output.textureData.texture;
    return 0;
}
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products, while **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

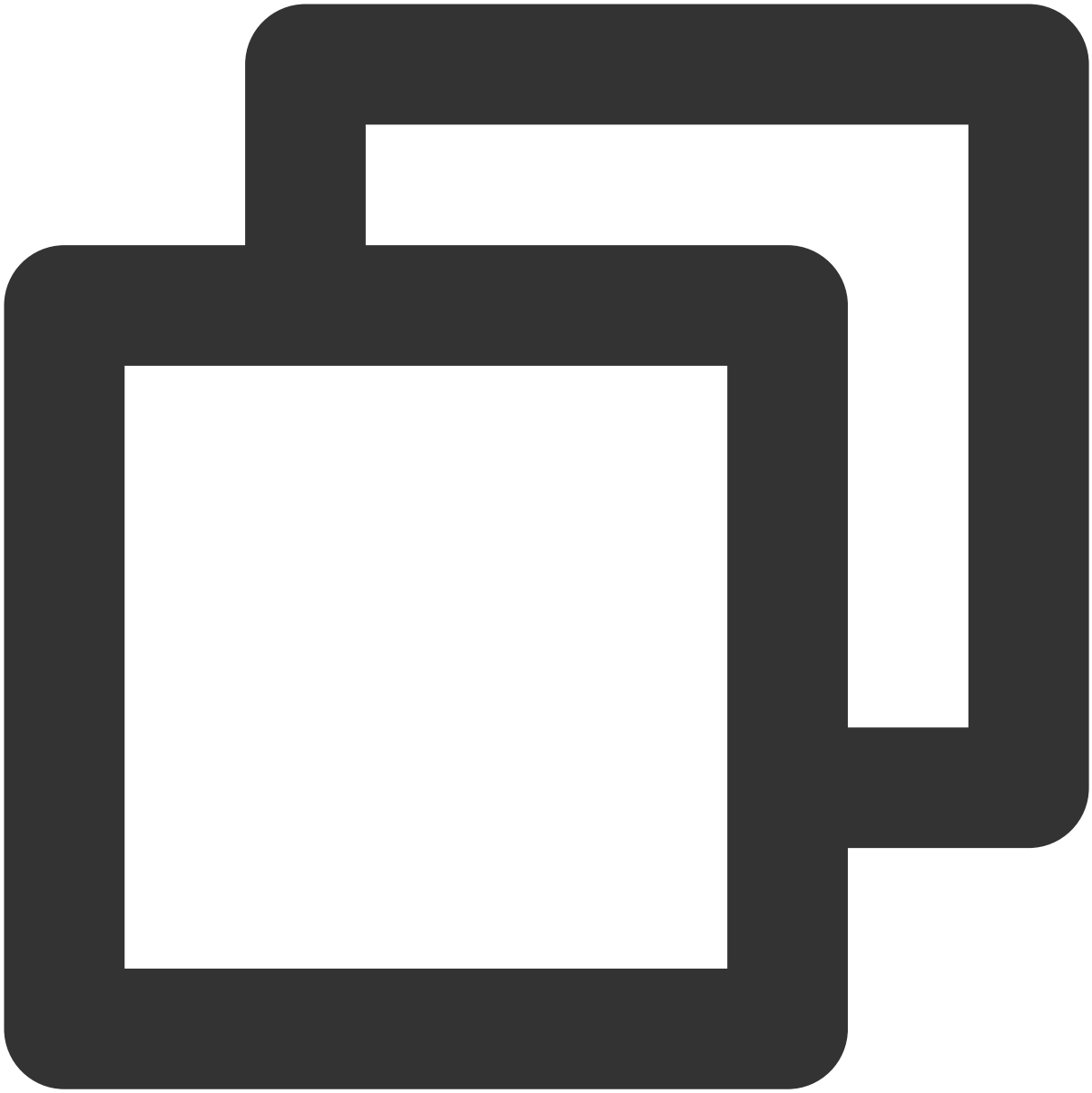
For scenario-specific integration guidelines of beauty effects, see Integrating Special Effect into TRTC SDK. For guidelines on integrating beauty effects independently, see Integrating Special Effect SDK.

Dual-Stream Encoding Mode

When the dual-stream encoding mode is enabled, the current user's encoder outputs two video streams, a high-definition large screen and a low-definition small screen, at the same time (but only one audio stream). In this way, other users in the room can choose to subscribe to the high-definition large screen or low-definition small screen based on their network conditions or screen sizes.

1. Enable large-and-small-screen dual-stream encoding mode.

©2013-2022 Tencent Cloud. All rights reserved.



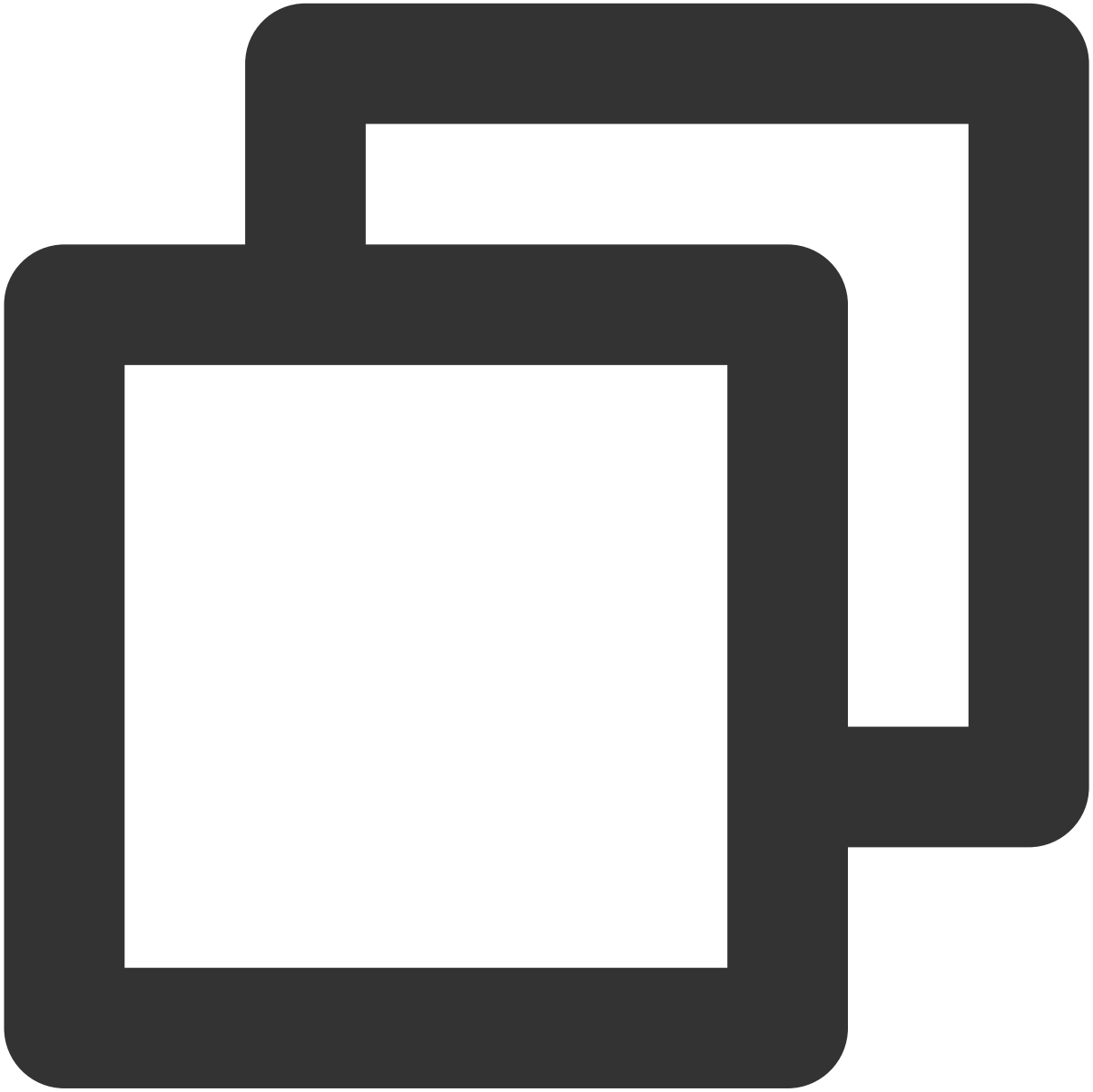
```
- (void)enableDualStreamMode:(BOOL)enable {
    // Video encoding parameters for the small stream (customizable).
    TRTCVideoEncParam *smallVideoEncParam = [[TRTCVideoEncParam alloc] init];
    smallVideoEncParam.videoResolution = TRTCVideoResolution_480_270;
    smallVideoEncParam.videoFps = 15;
    smallVideoEncParam.videoBitrate = 550;
    smallVideoEncParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud enableEncSmallVideoStream:enable withQuality:smallVideoEncParam
}
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, it consumes more CPU and network bandwidth. Therefore, it may be considered for use on Mac, Windows, or high-performance Pads. It is not recommended for mobile devices.

2. Select the type of remote user's video stream to pull.



```
// Optional video stream types when you subscribe to a remote user's video stream
[self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig view:view]

// You can switch the size of the specified remote user's screen at any time
[self.trtcCloud setRemoteVideoStreamType:userId type:TRTCVideoStreamTypeSmall];
```

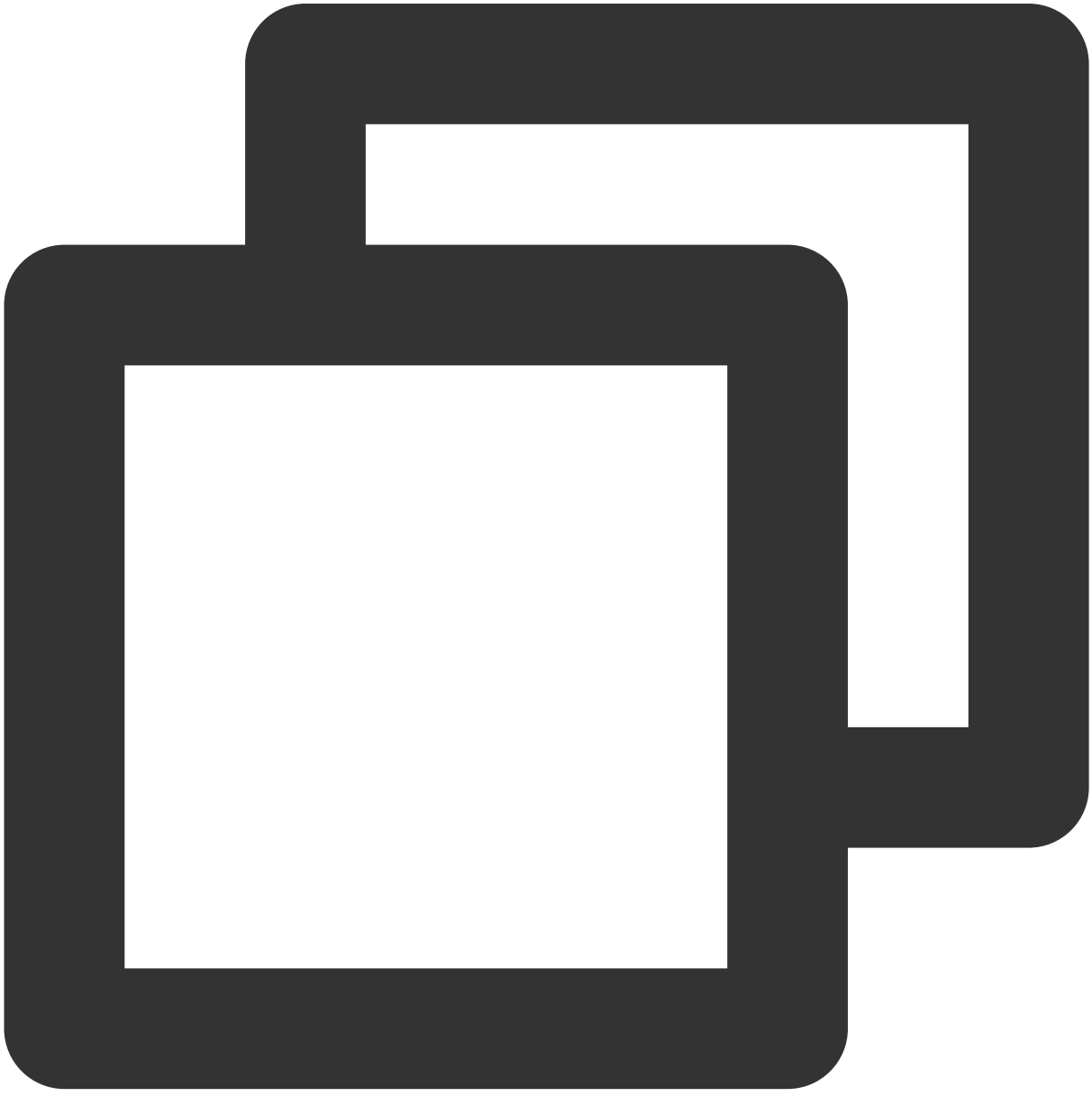
Note:

©2013-2022 Tencent Cloud. All rights reserved.

When the dual-stream encoding mode is enabled, you can specify the video stream type as `TRTCVideoStreamTypeSmall` with `streamType` to pull a low-quality small video for viewing.

View rendering control

If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.



```
// Update local preview screen rendering control
[self.trtcCloud updateLocalView:view];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Update the remote user's video rendering control
[self.trtcCloud updateRemoteView:view streamType:TRTCVideoStreamTypeBig forUser:use
```

Note:

The pass-through parameter `view` refers to the target video rendering control. And `streamType` only supports `TRTCVideoStreamTypeBig` and `TRTCVideoStreamTypeSub` .

Exception Handling

Exception error handling

When the TRTC SDK encounters an unrecoverable error, the error is thrown in the `onError` callback. For details, see Error Code Table.

1. UserSig related

UserSig verification failure leads to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

2. Room entry and exit related

If room entry is failed, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect.Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Nnote that roomId and strRoomId cannot be used interchangeably.

©2013-2022 Tencent Cloud. All rights reserved.

ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

3. Device related

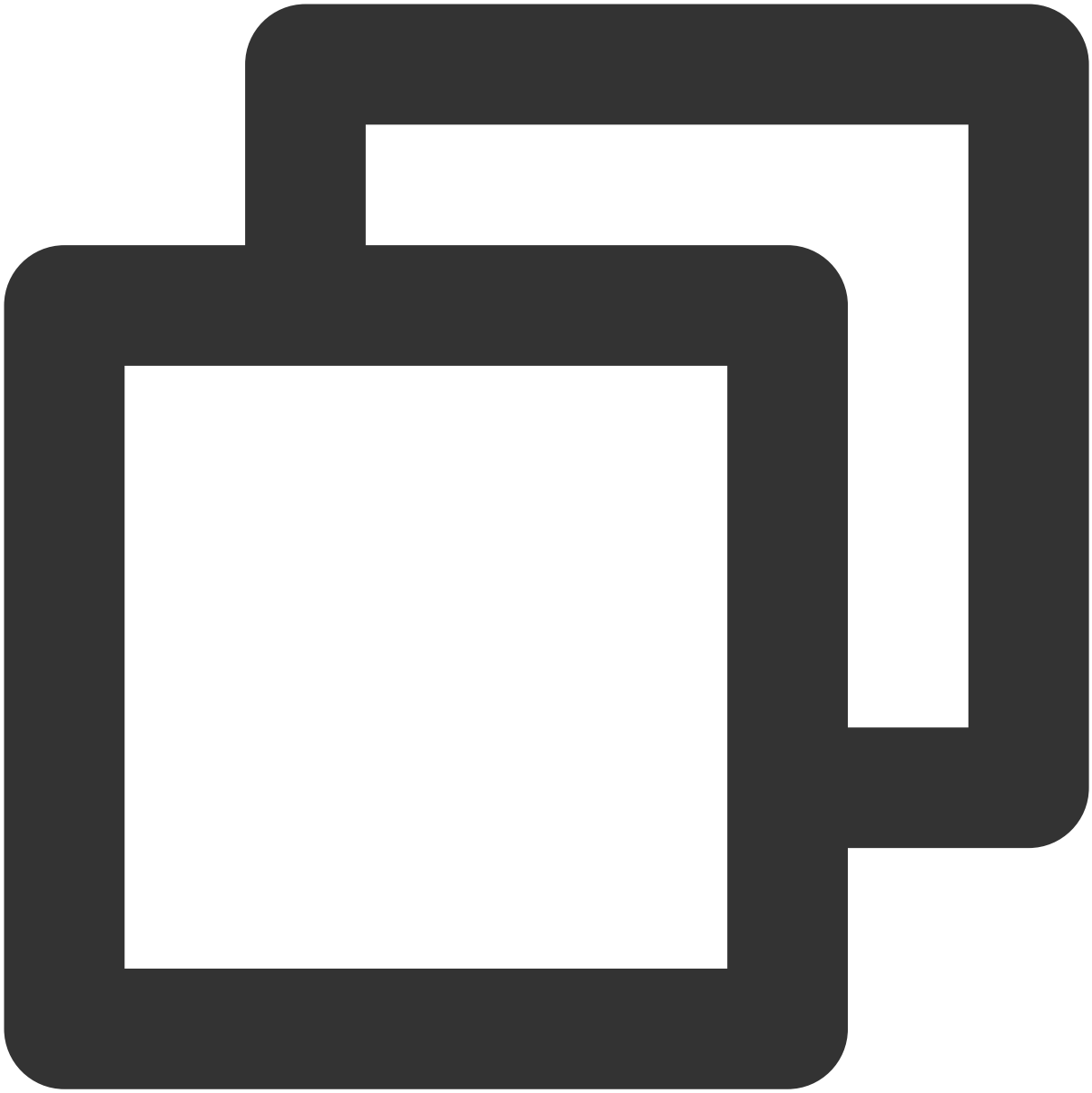
Errors for related monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to enable the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Issues with the remote mirror mode not functioning properly

In TRTC, video mirror settings are divided into local preview mirror `setLocalRenderParams` and video encoding mirror `setVideoEncoderMirror` . These settings separately affect the mirror effect of the local preview and the video encoding output (the mirror mode for remote viewers and cloud recordings). If you expect the mirror effect seen in the local preview to also take effect on the remote viewer's end, follow these encoding procedures.

©2013-2022 Tencent Cloud. All rights reserved.



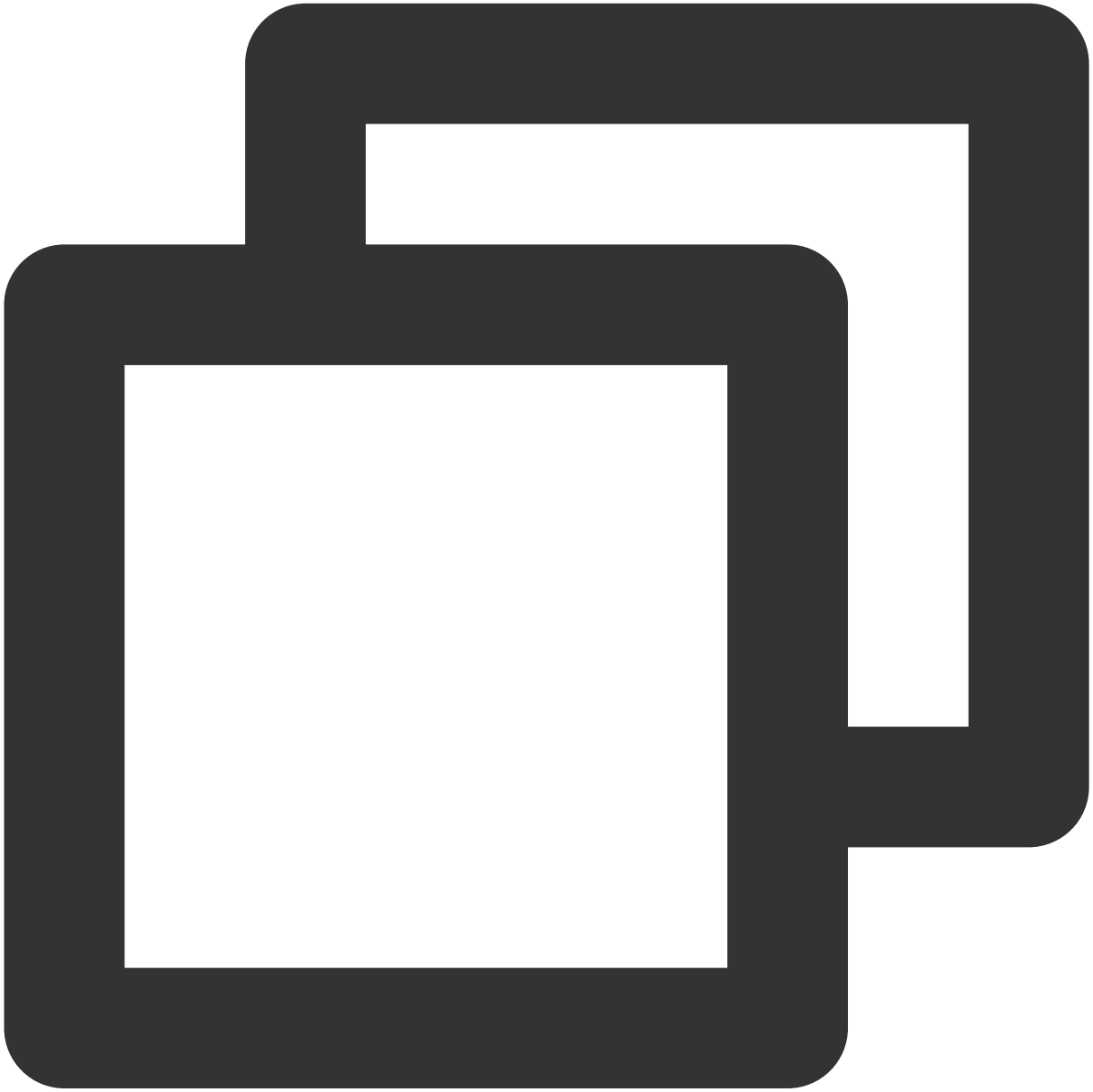
```
// Set the rendering parameters for the local video
TRTCRenderParams *params = [[TRTCRenderParams alloc] init];
params.mirrorType = TRTCVideoMirrorTypeEnable; // Video mirror mode
params.fillMode = TRTCVideoFillMode_Fill; // Video fill mode
params.rotation = TRTCVideoRotation_0; // Video rotation angle
[self.trtcCloud setLocalRenderParams:params];
// Set the video mirror mode for the encoder output
[self.trtcCloud setVideoEncoderMirror:YES];
```

Issues with camera scale, focus, and switch

©2013-2022 Tencent Cloud. All rights reserved.

In e-commerce live streaming scenarios, the anchor may need to custom adjust the camera settings. The TRTC SDK's device management class provides APIs for these needs.

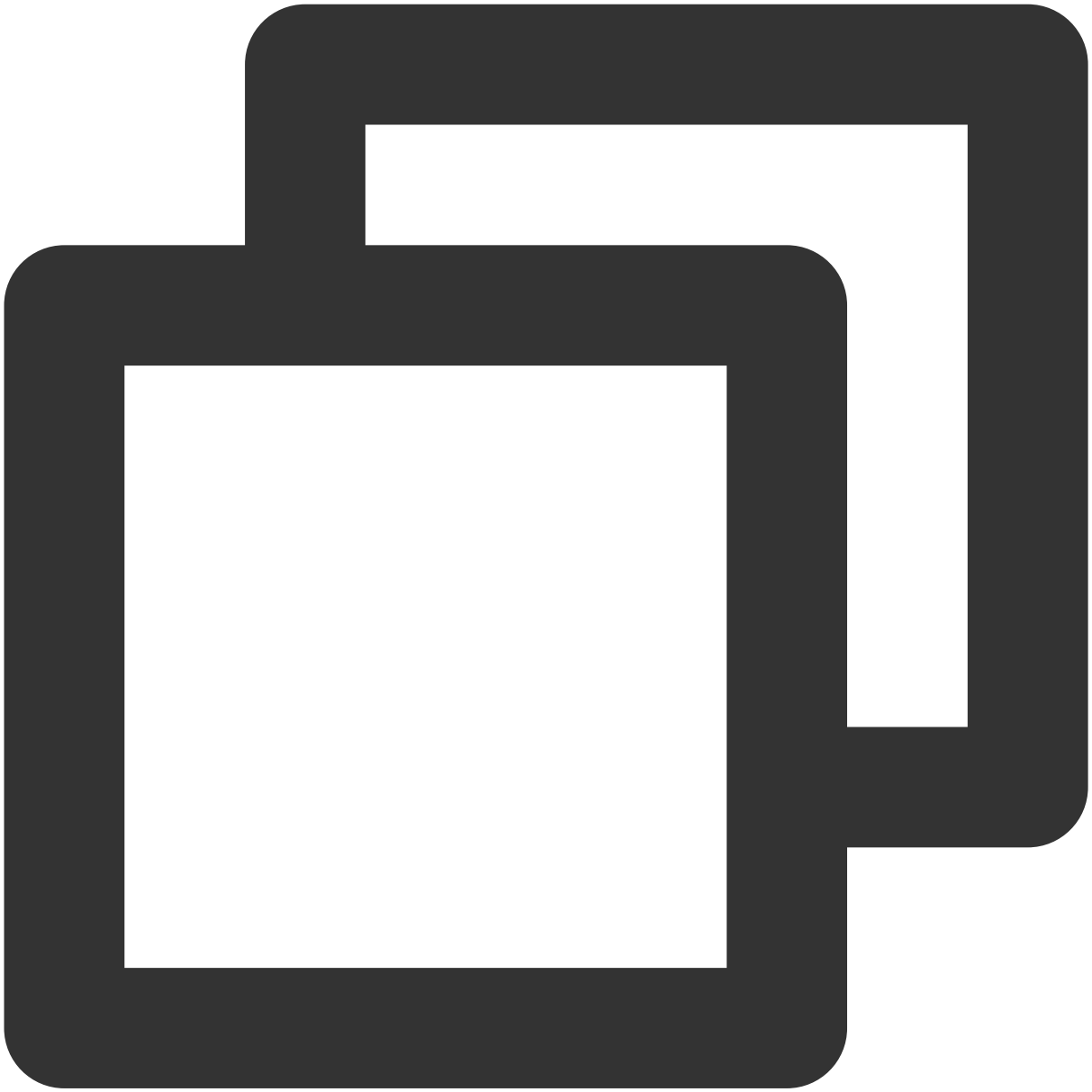
1. Query and set the zoom factor for the camera.



```
// Get the maximum zoom factor for the camera (only for mobile devices)
CGFloat zoomRatio = [[self.trtcCloud getDeviceManager] getCameraZoomMaxRatio];
// Set the zoom factor for the camera (only for mobile devices)
// Value range is 1-5. 1 means the furthest field of view (normal lens), and 5 mean
[[self.trtcCloud getDeviceManager] setCameraZoomRatio:zoomRatio];
```

2. Set the focus feature and position of the camera.

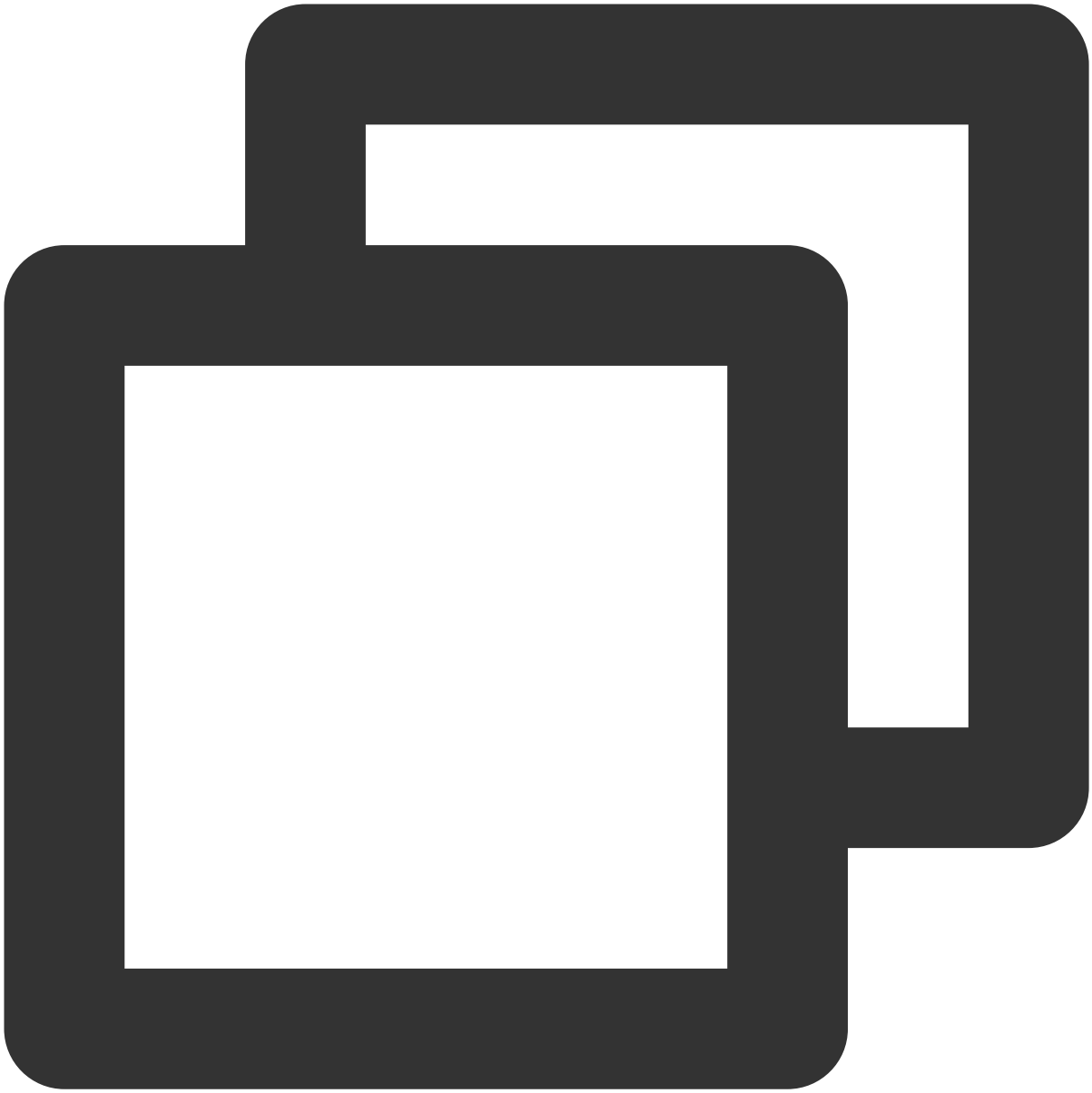
©2013-2022 Tencent Cloud. All rights reserved.



```
// Enable or disable the camera's autofocus feature (only for mobile devices)
[[self.trtcCloud getDeviceManager] enableCameraAutoFocus:NO];
// Set the focus position of the camera (only for mobile devices)
// The precondition for using this API is to first disable the autofocus feature us
[[self.trtcCloud getDeviceManager] setCameraFocusPosition:CGPointMake(x, y)];
```

3. Determine and switch to front or rear cameras.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Determine if the current camera is the front camera (only for mobile devices)
BOOL isFrontCamera = [[self.trtcCloud getDeviceManager] isFrontCamera];
// Switch to front or rear cameras (only for mobile devices)
// Passing true means switching to front, and passing false means switching to rear
[[self.trtcCloud getDeviceManager] switchCamera:!isFrontCamera];
```

©2013-2022 Tencent Cloud. All rights reserved.

Audio/Video Call
1V1 Audio and Video Call Scenario Solution

Last updated：2024-07-25 17:09:36

Scene Overview

Scene Introduction

1V1 Audio and Video Call is a high-frequency usage scene similar to WeChat calls. TRTC (Tencent Real-Time Communication) has an audio call latency of less than 300 ms, a packet loss resistance rate of over 80%, and can resist network jitter of over 1000 ms, ensuring smooth and stable audio calls even in weak network environments. Video calls support high-definition quality of 720 p, 1080 p, 2 K, and 2 K+ (specific devices), providing high-quality video call services. Combined with the rich call signaling management APIs provided by Chat, it easily adapts to various use cases. In addition, we also offer Audio/Video Call scene-based components that can be directly reused, significantly reducing development costs. For details, see Component Introduction.

©2013-2022 Tencent Cloud. All rights reserved.

Scene Approach

The 1V1 Audio and Video Call feature not only incorporates the basic functionality of a WeChat-like calling application, but it also has the potential to transform into a wide range of diverse use cases. Below are a few common scenes briefly introduced.

Game Socializing

In the gaming field, Audio/Video Call facilitates real-time interactions among players, enhancing the overall gaming experience. Players can engage in voice or video chats with friends within the game, share gaming experiences, techniques, or collaborate on policy. Nowadays, audio and video calls are extensively utilized in game socializing features, such as team voice chats.

Online Customer Service

1V1 Audio and Video Call enables customers to communicate with customer service representatives in real-time, resulting in more effective problem-solving. Compared to traditional text-based customer service, audio and video

©2013-2022 Tencent Cloud. All rights reserved.

calls allow customers to describe their issues more vividly and enable service personnel to understand customer needs more clearly, thus improving the efficiency of problem resolution. For instance, dispute resolution and insurance consulting are excellent use cases for this type of communication.

Online Consultation

In the healthcare field, 1V1 Audio and Video Call enables patients to consult with doctors remotely. Patients can describe their symptoms via Audio/Video Call, and doctors can make preliminary diagnoses based on the descriptions. This method not only saves time and energy for patients but also allows doctors to serve more patients, improving the usage of medical resources.

Financial Review

In the financial field, 1V1 Audio and Video Call can be utilized for identity verification and risk assessment. When performing online financial management, account opening, or face-to-face signing, in accordance with national regulatory requirements, audio and video recording services must be provided to create transaction record videos for archiving and reference. Audio/Video Calls are extensively used in the financial review sector, not only enhancing the efficiency of reviews but also mitigating the risk of fraud.

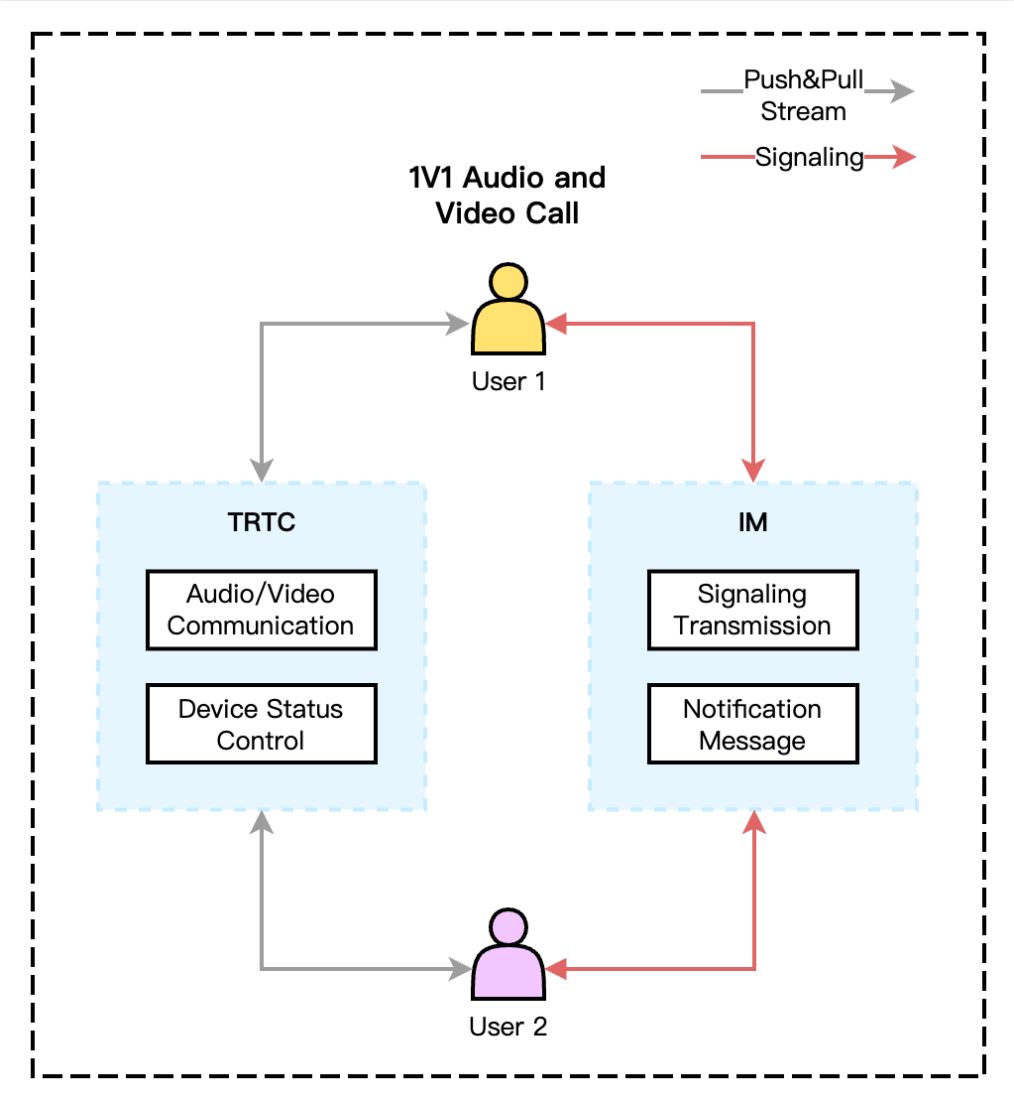
Implementation Scheme

Typically, implementing a basic 1V1 Audio and Video Call scene involves multiple feature modules. We can divide the implementation scheme into three parts: Call Signaling Control, Audio/Video Call, Call Feature Control. The key actions and features of each part are shown in the table below:

Functional Module	Key Actions and Feature Points
Call Signaling Control	Call, Answer, Decline, Hang up
Audio/Video Call	Voice Call, Video Call
Call Feature Control	Enable/Disable Microphone/Camera/Speaker, Earpiece/Hands-free Switching, Camera Switching, Window Size Switching, Network Status Prompt, Call Duration Statistics

The complete implementation of Audio/Video Call scenes often relies on the combined capabilities of real-time audio and video and instant messaging. The real-time audio and video module is responsible for audio and video communication and device status control, while the instant messaging module handles signaling transmission and message push. The main architecture of Audio/Video Call scene is shown below:

©2013-2022 Tencent Cloud. All rights reserved.



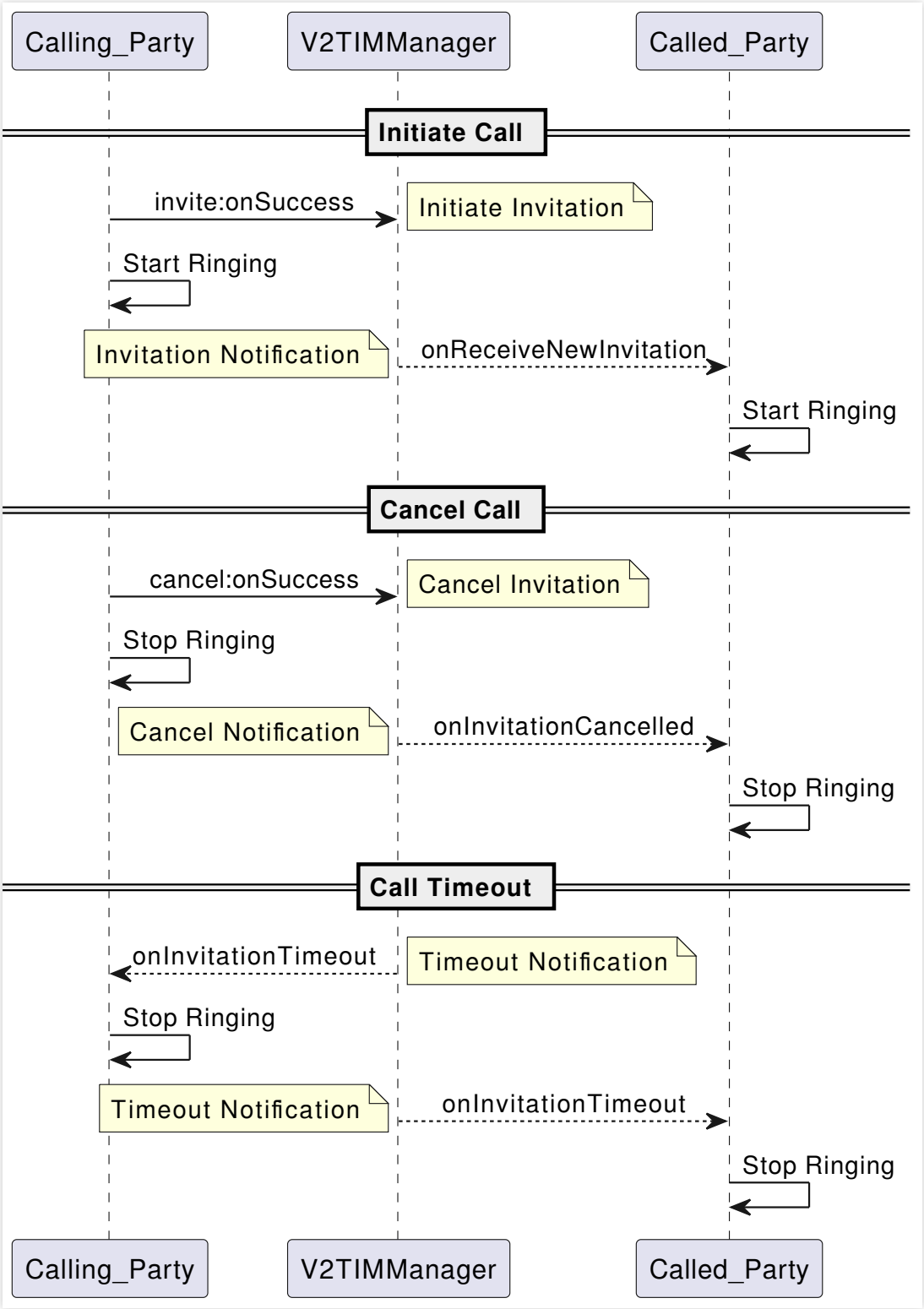
Call Signaling Control

Based on a complete call process, call signaling can be divided into Call,Answer,Decline,Hang up. Taking Chat as an example, the following describes the specific implementation logic of the call signaling control after completing the Log-in Operation.

Call

Call signaling can be subdivided into initiating a call, canceling a call, and call timeout, and their invocation sequence is shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Initiating a call: The caller sends a call invitation to the callee, displays the call page, and plays the ringtone; the callee receives the invitation, displays the call page, and plays the ringtone.

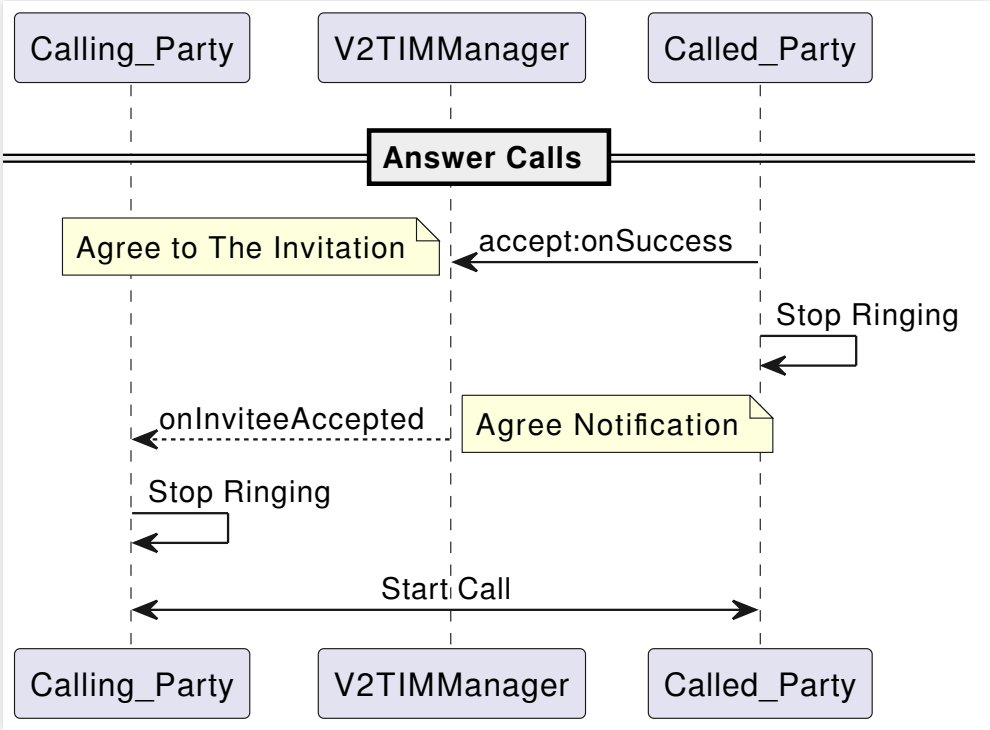
Canceling a call: The caller can cancel the call invitation midway, destroy the call page, and stop the ringtone; the callee receives the cancellation notification, terminates the call page, and stops the ringtone.

©2013-2022 Tencent Cloud. All rights reserved.

Call Timeout: If there is no response beyond the invite's predefined timeout period, both the caller and callee will receive a timeout notification, terminate the call page, and stop the ringtone.

Answer

Upon receiving a call invitation from the caller, the callee can choose to answer the call, initiating the Audio/Video Call.

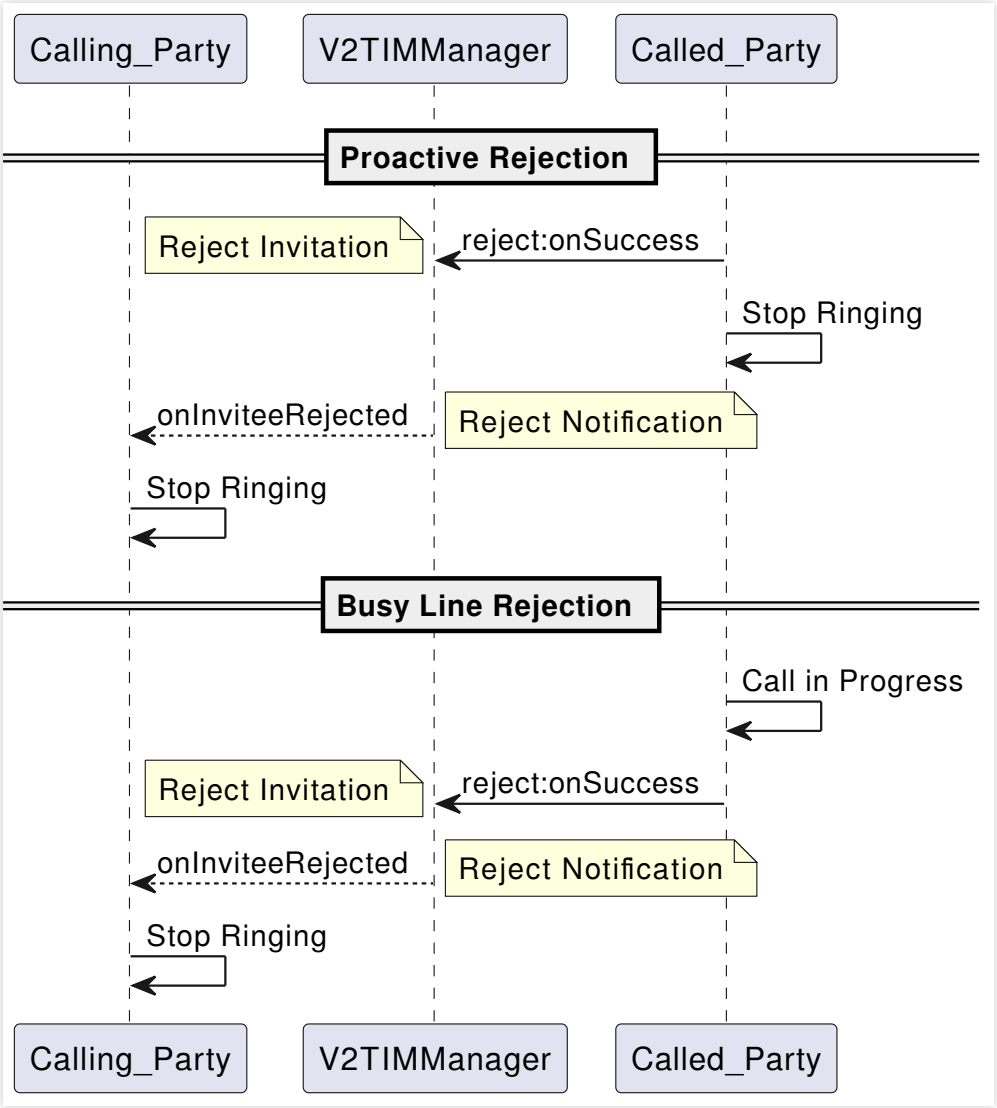


After answering the call, both parties start interactive audio and video communication. For more details on implementation logic, see Audio/Video Call.

Decline

The decline signaling can be subdivided into active decline and busy decline, and their call sequence is shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Proactive Rejection: The callee rejects the call invitation upon receipt, also terminates the call page and stops the ringtone; the caller receives the rejection notice, also terminates the call page and stops the ringtone.
Busy Line Rejection: Upon receiving the call invitation, the callee directly rejects the invitation if a call is already in progress; the caller receives the rejection notice, also terminates the call page and stopping the ringtone.

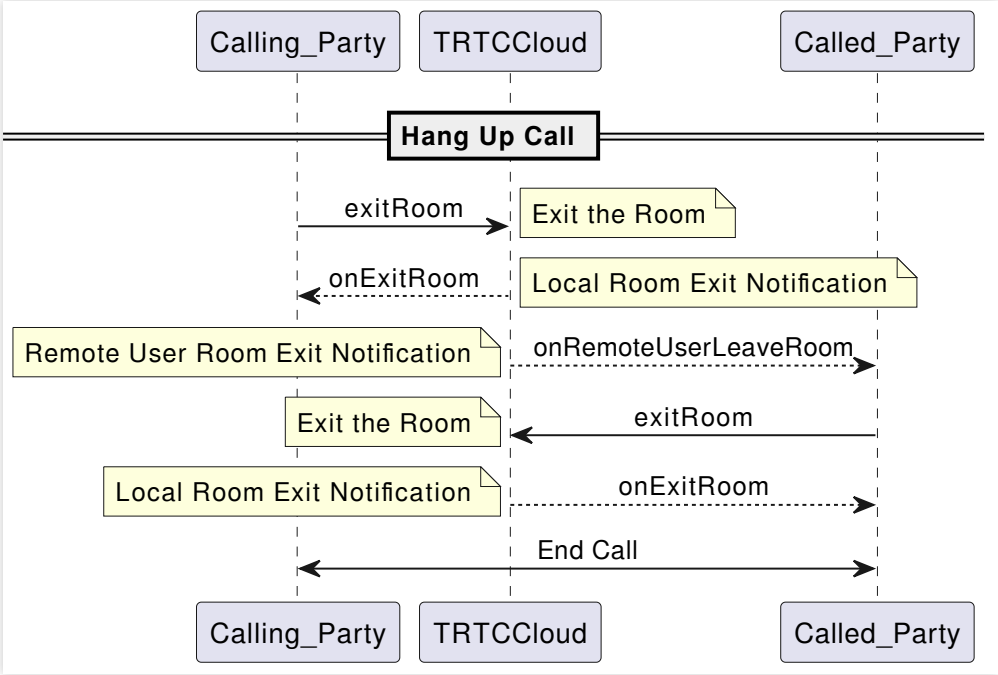
Note:

Both proactive and busy line rejections use the reject signal for implementation, but it's important to distinguish them through the custom data field in the signaling.

Hang up

During a call, either the caller or the callee can opt to hang up at any time, thus ending the audio or video call.

©2013-2022 Tencent Cloud. All rights reserved.



Taking the caller hanging up as an example: The caller performs the exit operation, the callee receives a remote exit notification, also performs the exit operation, and the call between both parties ends.

Note:

The hangup operation does not use the IM signaling notification but is implemented through the TRTC (Tencent Real-Time Communication) remote user exit callback notification.

Audio/Video Call

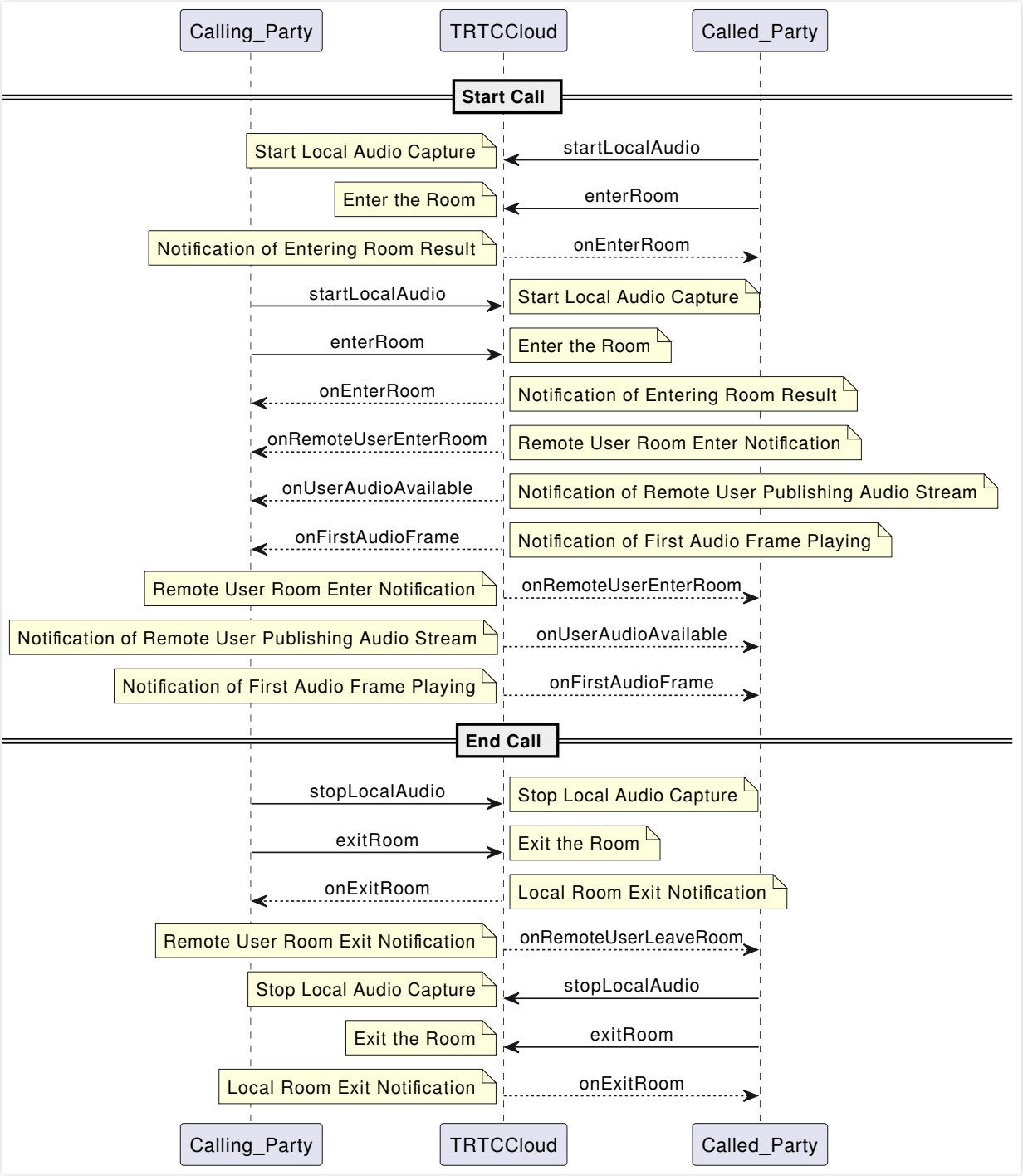
The Audio/Video Call mainly relies on the capabilities of TRTC (Tencent Real-Time Communication), which can be divided into Voice Call and Video Call. Below, we'll detail the specific implementation logic of these two parts.

Audio Call

After connecting, both parties need to enter **the same TRTC (Tencent Real-Time Communication) room**, start local audio capture and streaming, and mutually pull each other's audio stream to achieve a voice call.

The calling sequence for starting and ending a call's audio and video-related APIs is shown in the figure below:

©2013-2022 Tencent Cloud. All rights reserved.



Note:

In voice call mode, the TRTC (Tencent Real-Time Communication) room scene should use `TRTC_APP_SCENE_AUDIOCALL`, and the joining role `TRTCRoleType` should not be specified.

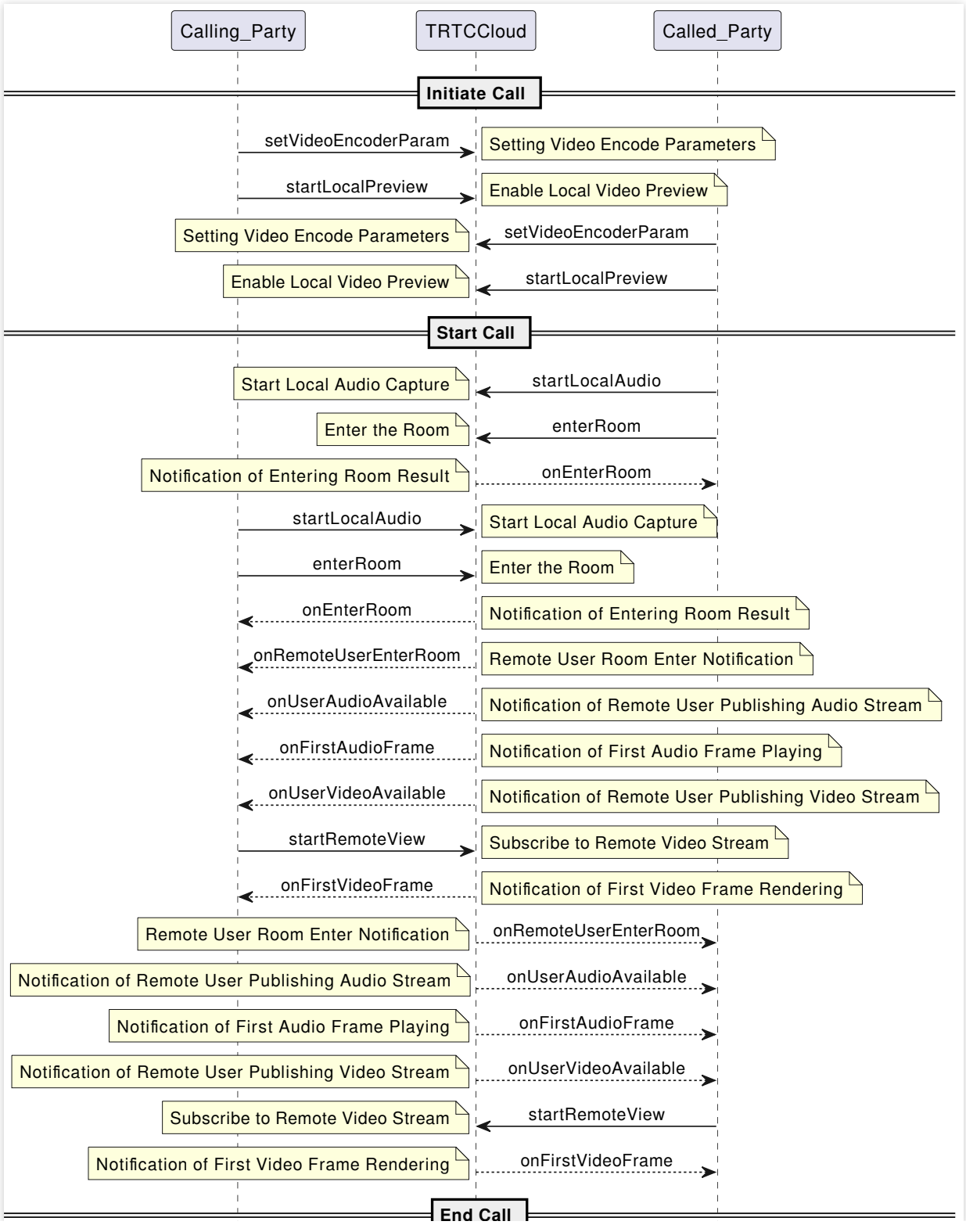
Starting local audio capture `startLocalAudio` allows you to set audio quality parameters at the same time. For voice calls, it's recommended to set `TRTC_AUDIO_QUALITY_SPEECH`.

Under the SDK's default automatic subscription mode, after a user enters a room, they will immediately receive the audio stream from that room, which will be automatically decoded and played without manual pulling.

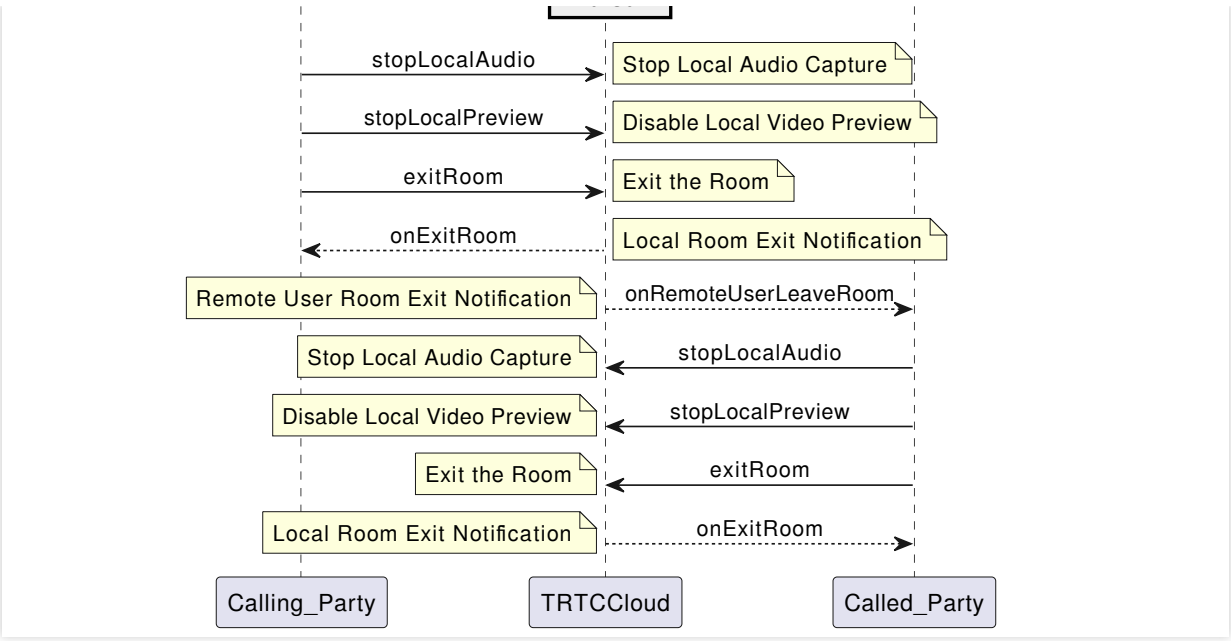
Video Call

©2013-2022 Tencent Cloud. All rights reserved.

During the calling phase, both parties must set video encode parameters and start local video preview. After connecting, both parties need to enter **the same TRTC (Tencent Real-Time Communication) room**, start local audio capture and streaming, and mutually pull each other's audio and video streams to achieve a video call.
The calling sequence for initiating a call, starting a call, and ending a call's audio and video-related APIs is shown in the figure below:



©2013-2022 Tencent Cloud. All rights reserved.



Note:

In video call mode, the TRTC (Tencent Real-Time Communication) room scene should use `TRTC_APP_SCENE_VIDEOCALL`, and the joining role `TRTCRoleType` should not be specified.

Before entering the room, call `startLocalPreview`, and the SDK will only start the camera preview, waiting until you call `enterRoom` to start streaming.

Start local audio capture with `startLocalAudio`, where you can also set the audio parameter. For video calls, it is recommended to set to `TRTC_AUDIO_QUALITY_SPEECH`.

In the SDK's default automatic subscription mode, audio is automatically decoded and played back, while video requires manual invocation of `startRemoteView` to pull and render the remote video stream.

Call Feature Control

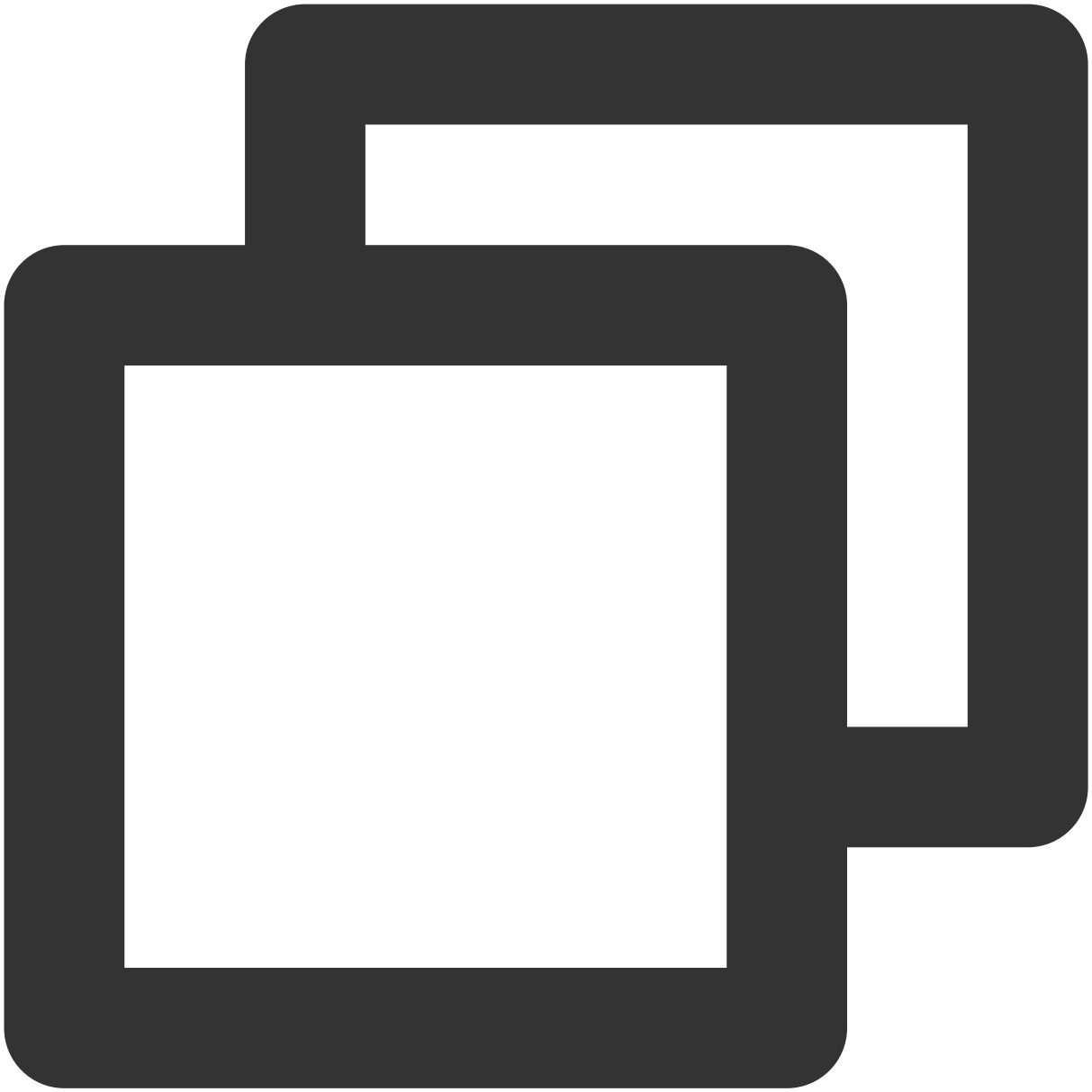
During Audio/Video Call, various feature controls might be involved, such as: turning on/off the microphone, turning on/off the speaker, turning on/off the camera, hands-free/earpiece switching, camera switching, window size switching, network status prompt, call duration statistics. Most of these feature controls and status prompts are facilitated through the TRTC (Tencent Real-Time Communication) SDK. Below, we will introduce their implementations one by one.

Turn on/off Microphone

Android

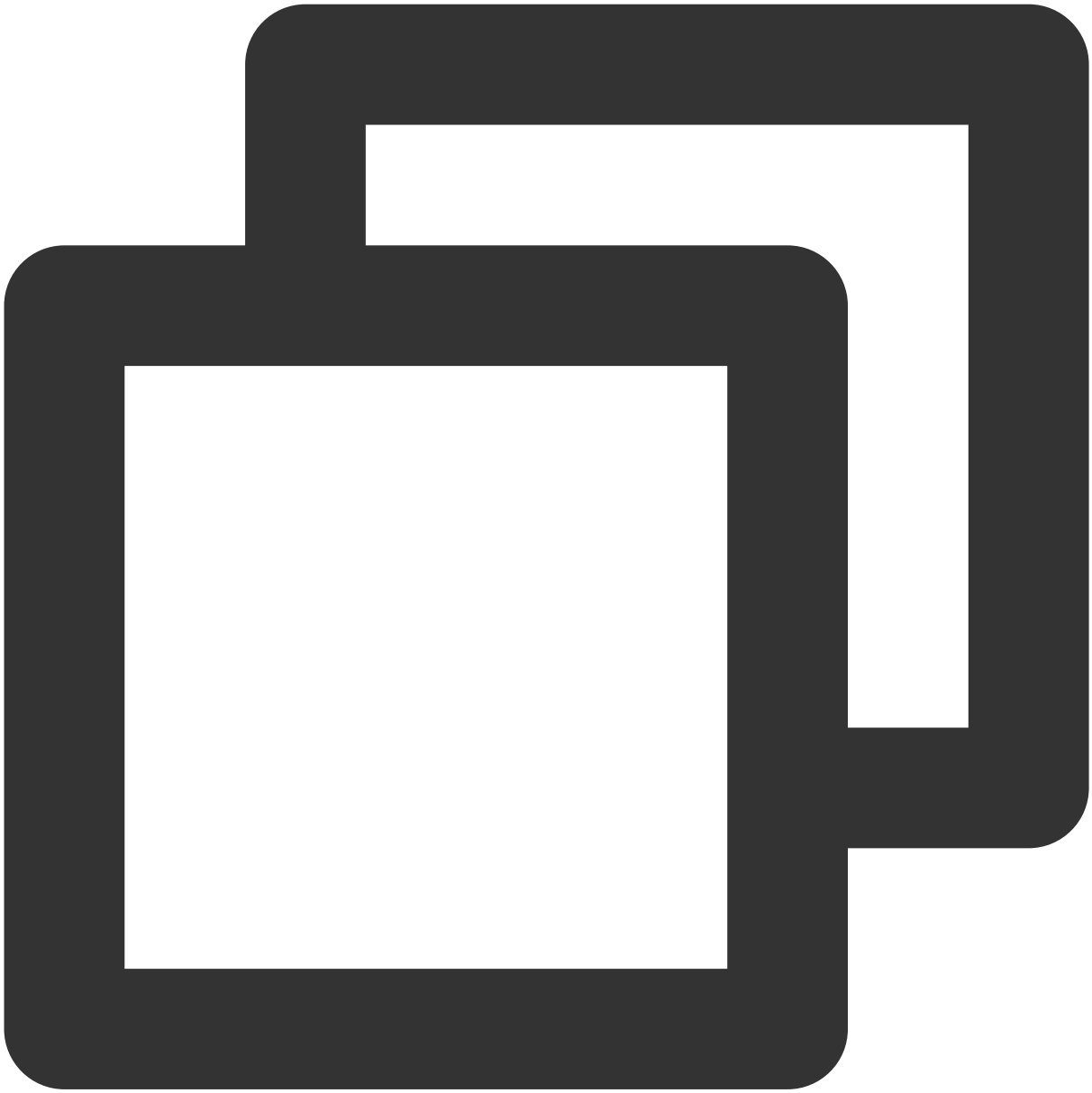
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Turn the mic on
mTRTCCloud.muteLocalAudio(false);
// Turn the mic off
mTRTCCloud.muteLocalAudio(true);
```

©2013-2022 Tencent Cloud. All rights reserved.



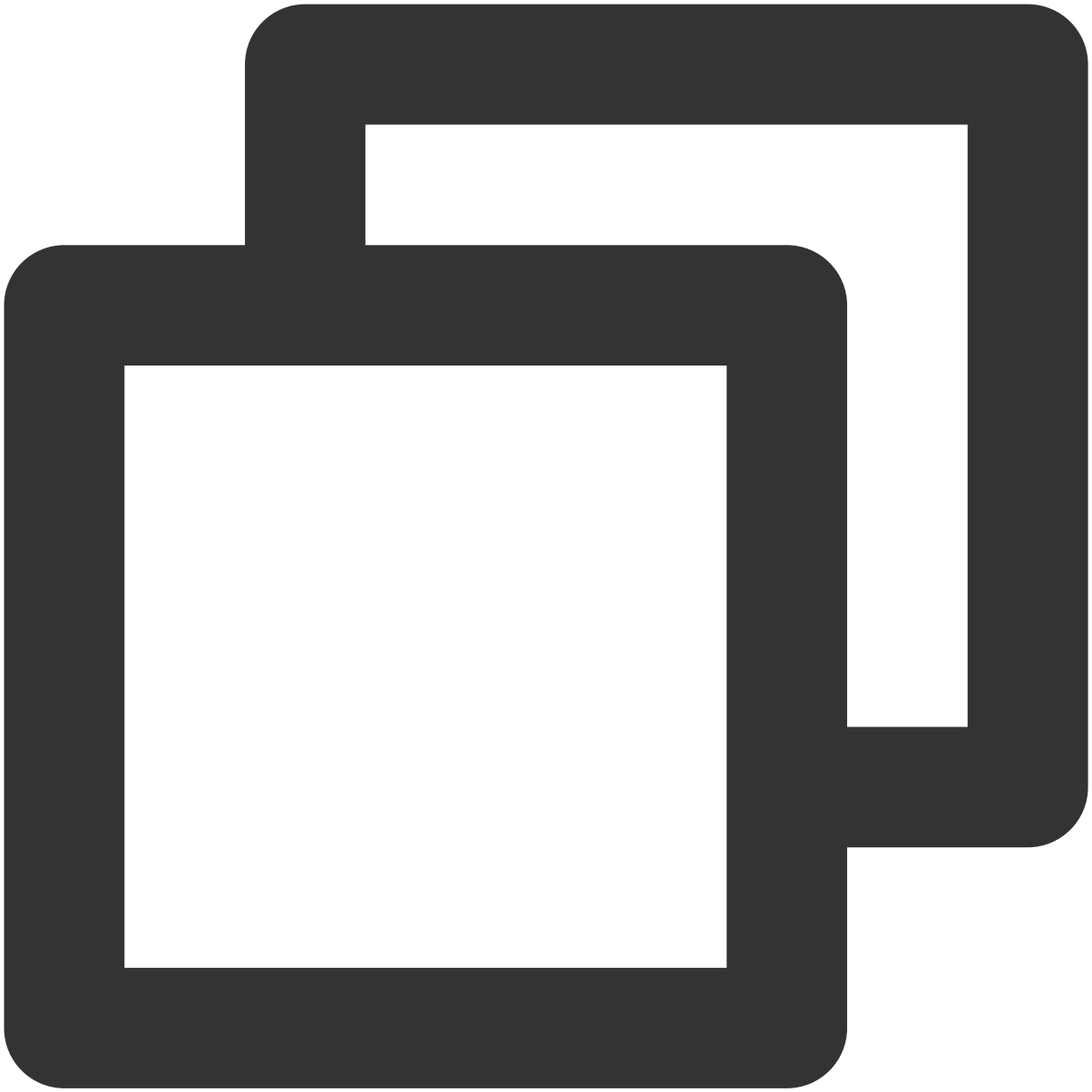
```
// Turn the mic on
[self.trtcCloud muteLocalAudio:NO];
// Turn the mic off
[self.trtcCloud muteLocalAudio:YES];
```

Turn on/off speaker

Android

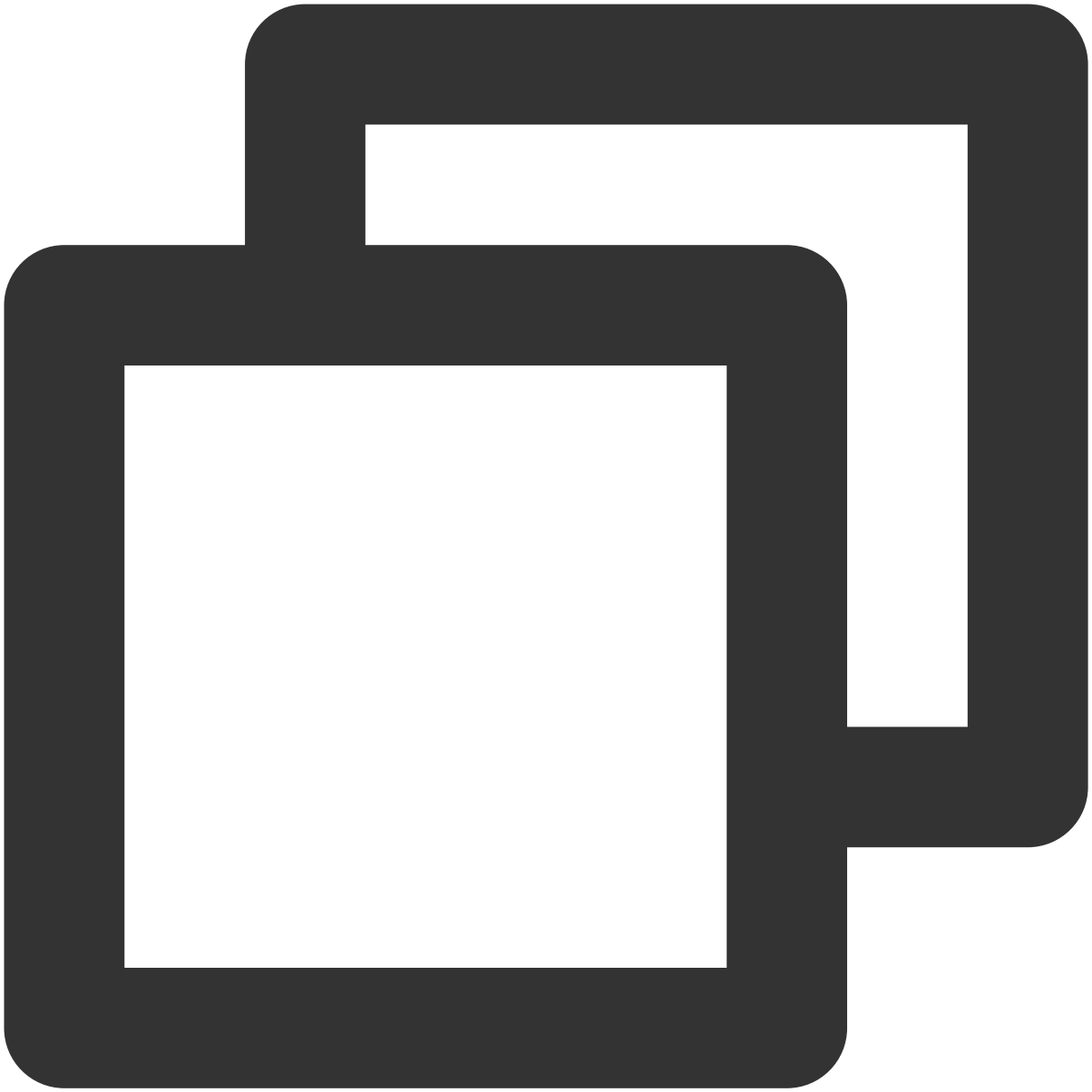
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Turn the speaker on
mTRTCCloud.muteAllRemoteAudio(false);
// Turn the speaker off
mTRTCCloud.muteAllRemoteAudio(true);
```

©2013-2022 Tencent Cloud. All rights reserved.



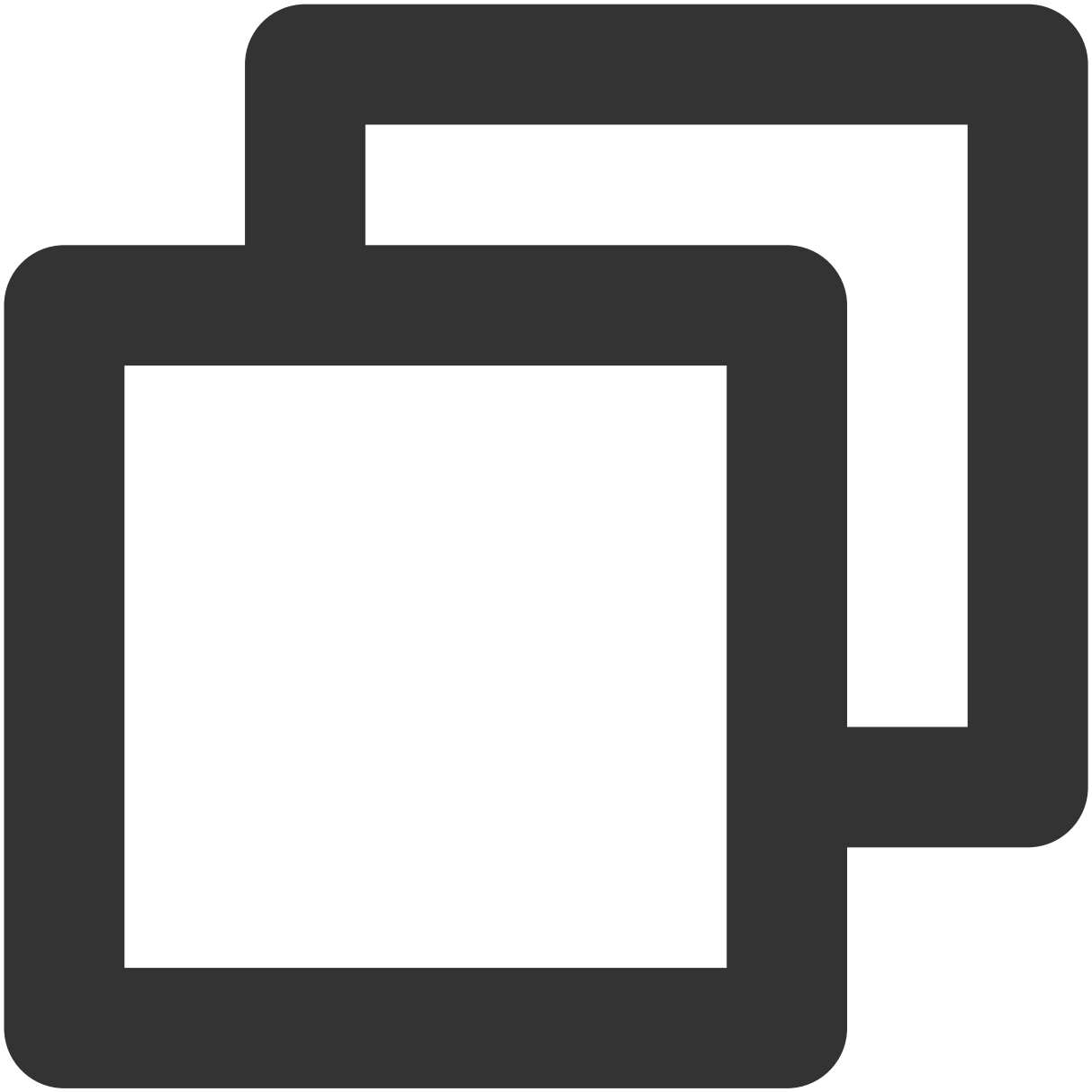
```
// Turn the speaker on
[self.trtcCloud muteAllRemoteAudio:NO];
// Turn the speaker off
[self.trtcCloud muteAllRemoteAudio:YES];
```

Turn on/off camera

Android

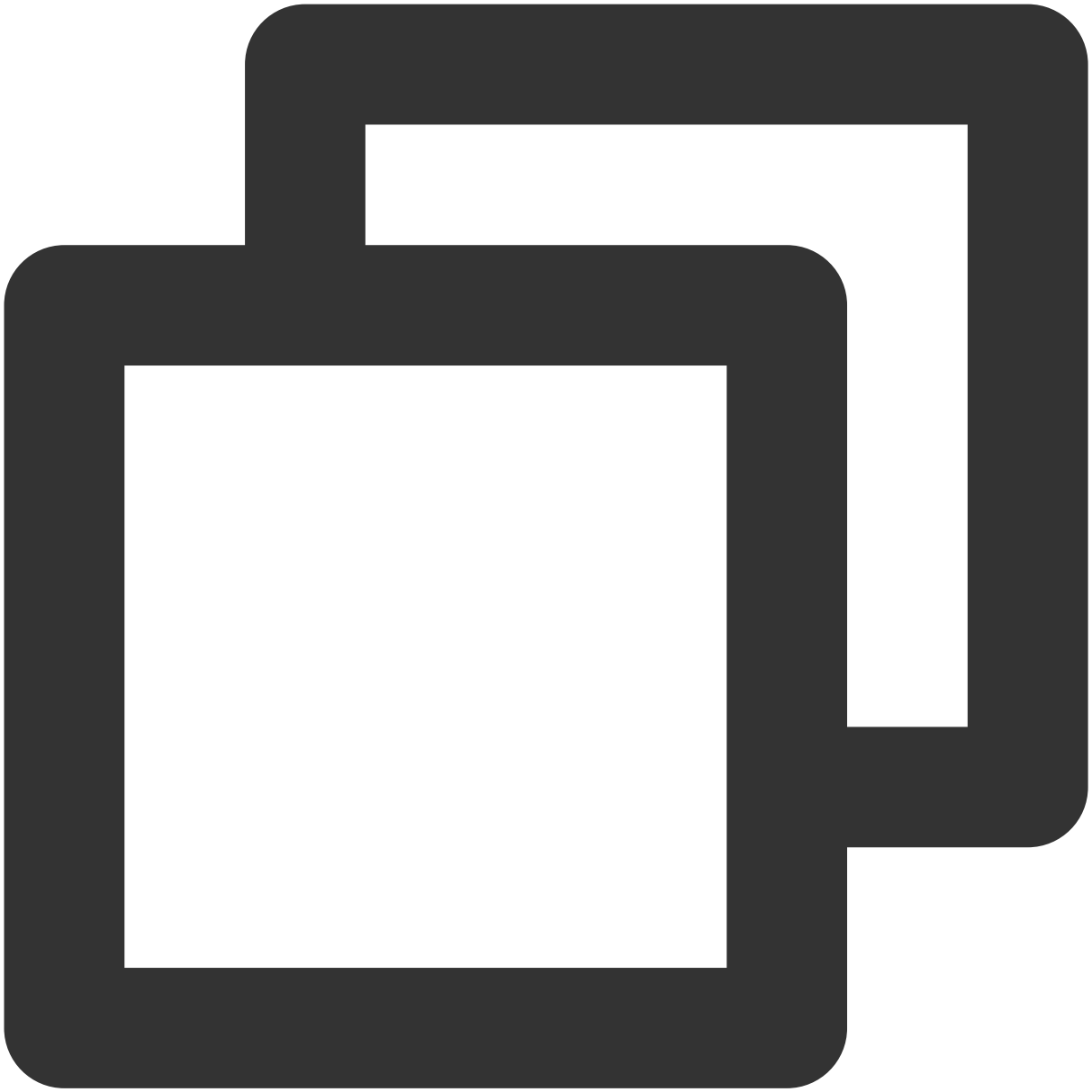
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Turn the camera on, specifying front or rear camera and the rendering widget
mTRTCCloud.startLocalPreview(isFrontCamera, videoView);
// Turn the camera off
mTRTCCloud.stopLocalPreview();
```

©2013-2022 Tencent Cloud. All rights reserved.



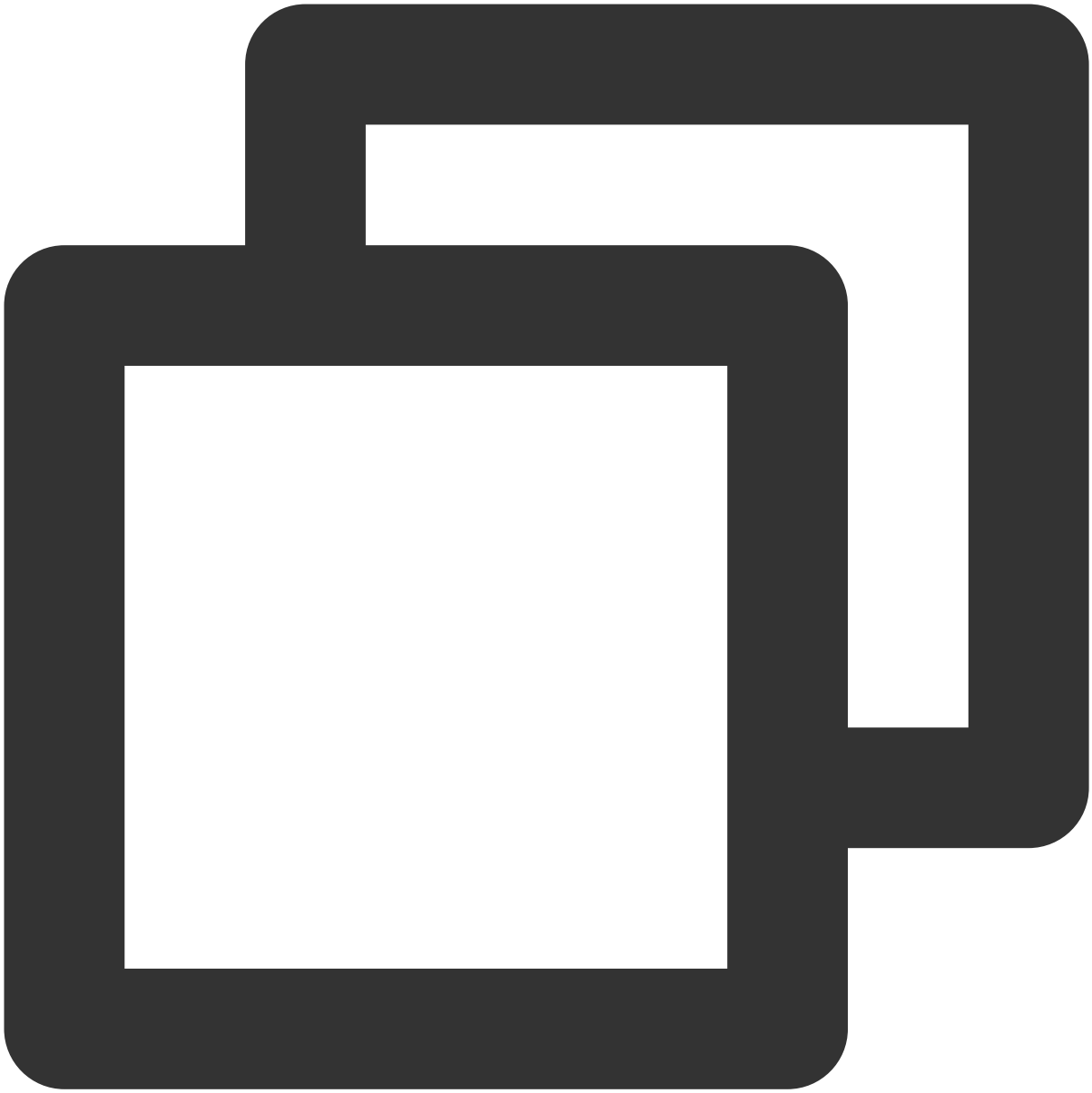
```
// Turn the camera on, specifying front or rear camera and the rendering widget
[self.trtcCloud startLocalPreview:self.isFrontCamera view:self.videoView];
// Turn the camera off
[self.trtcCloud stopLocalPreview];
```

Hands-free/Earpiece Switching

Android

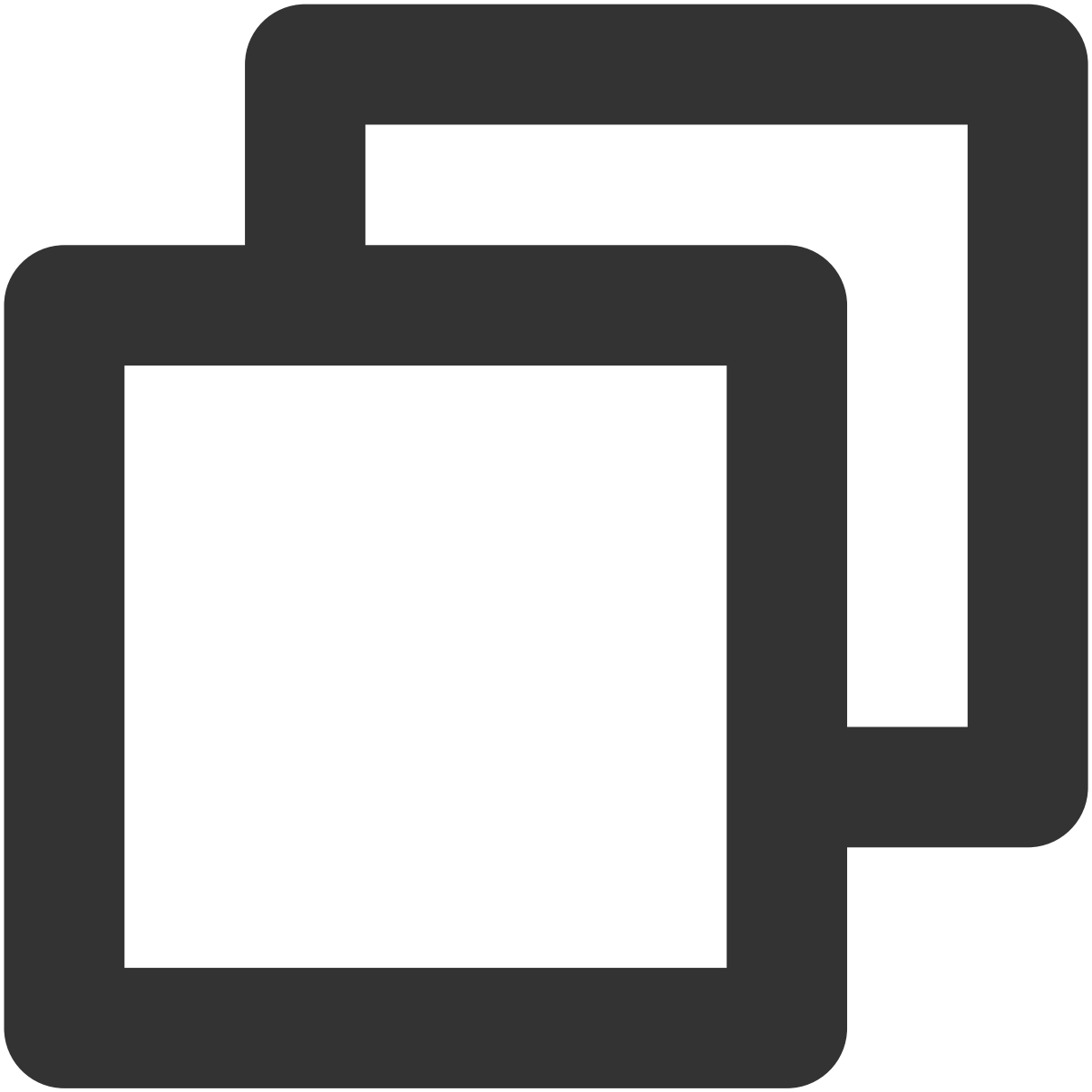
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Switch to earpiece
mTRTCCloud.getDeviceManager().setAudioRoute(TXDeviceManager.TXAudioRoute.TXAudioRou
// Switch to speakerphone
mTRTCCloud.getDeviceManager().setAudioRoute(TXDeviceManager.TXAudioRoute.TXAudioRou
```

©2013-2022 Tencent Cloud. All rights reserved.



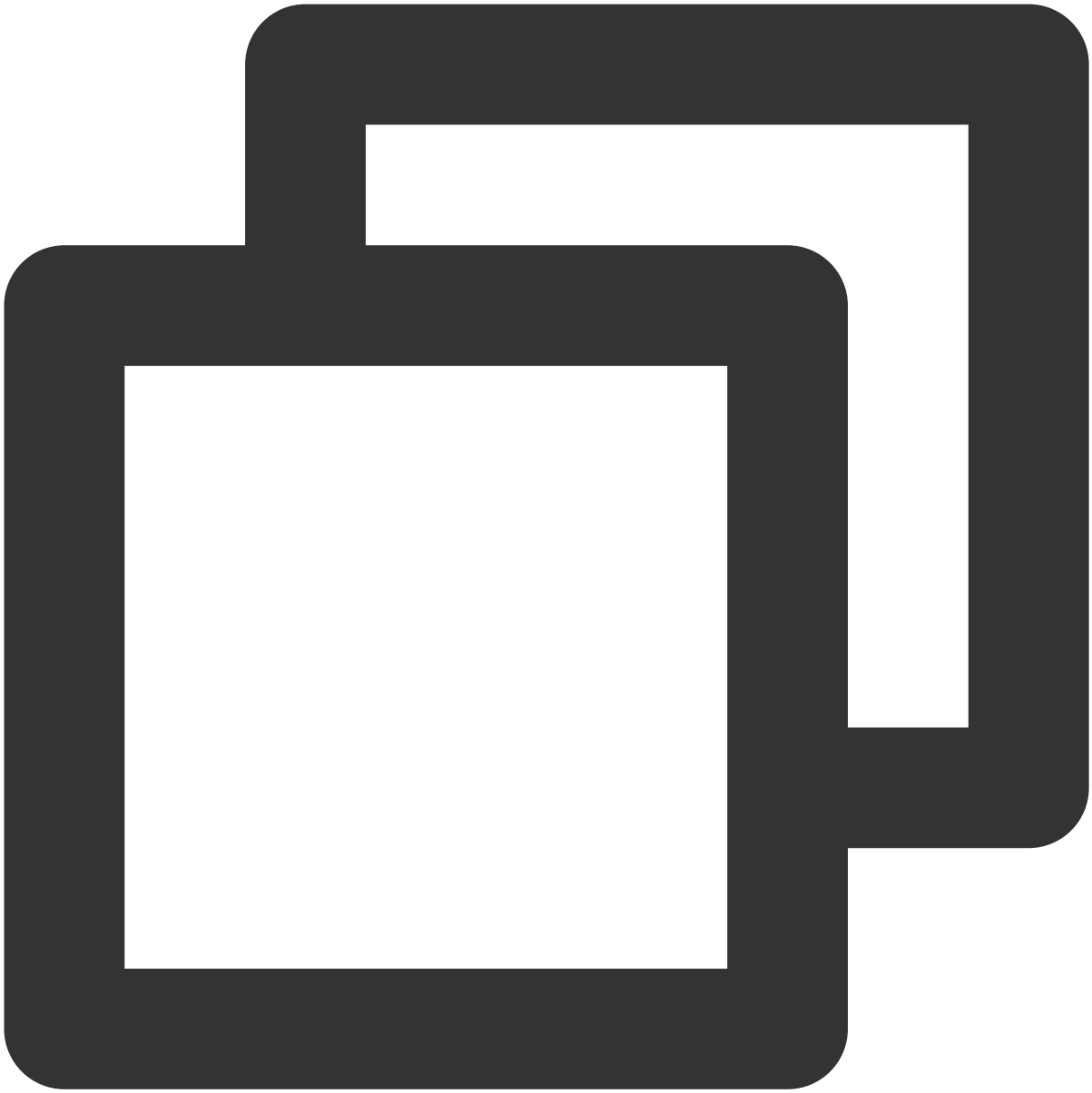
```
// Switch to earpiece
[[self.trtcCloud getDeviceManager] setAudioRoute:TXAudioRouteEarpiece];
// Switch to speakerphone
[[self.trtcCloud getDeviceManager] setAudioRoute:TXAudioRouteSpeakerphone];
```

Camera Switching

Android

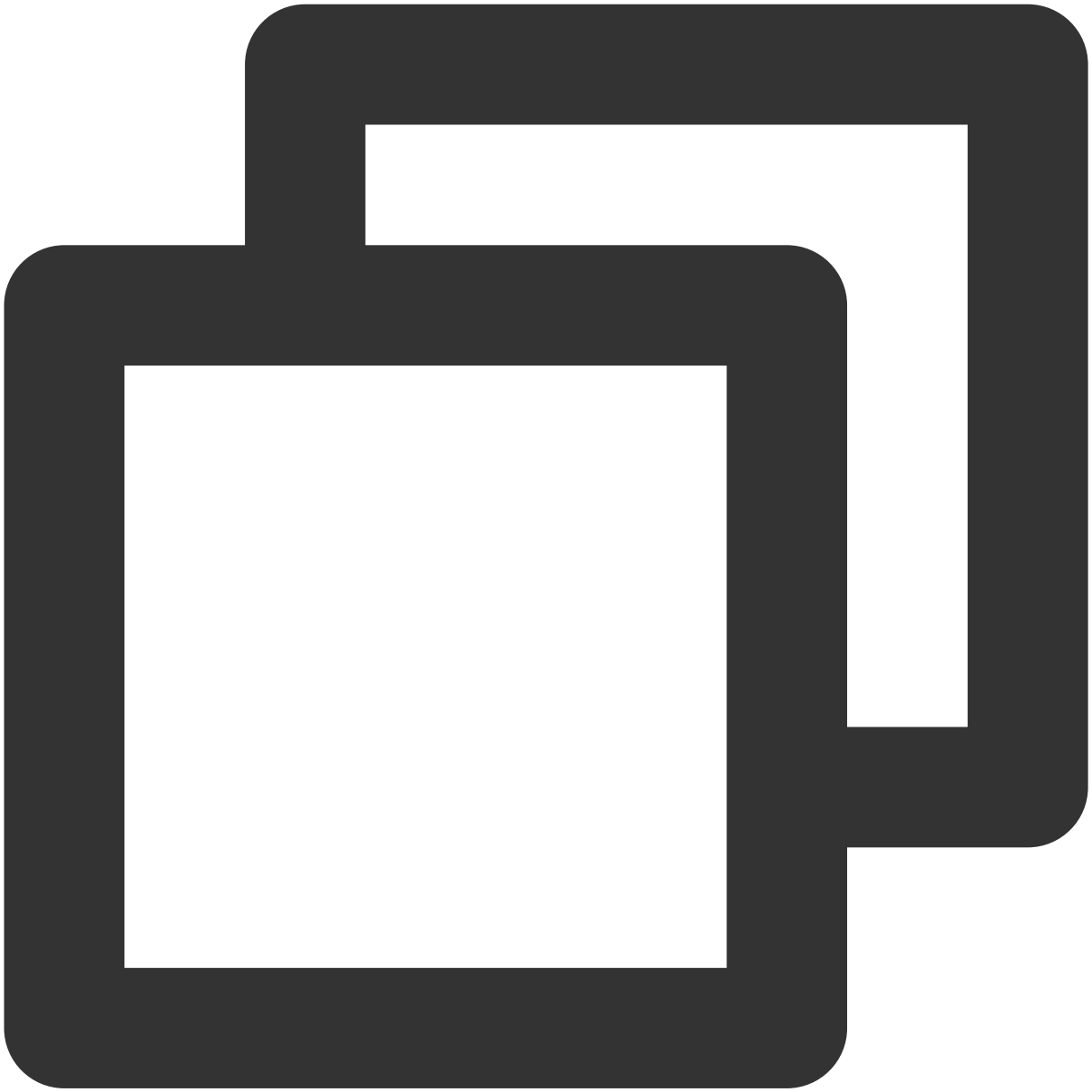
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Determine if the current camera is front-facing
boolean isFrontCamera = mTRTCCloud.getDeviceManager().isFrontCamera();
// Switch between front and rear cameras, true: switch to front-facing; false: swit
mTRTCCloud.getDeviceManager().switchCamera(!isFrontCamera);
```

©2013-2022 Tencent Cloud. All rights reserved.



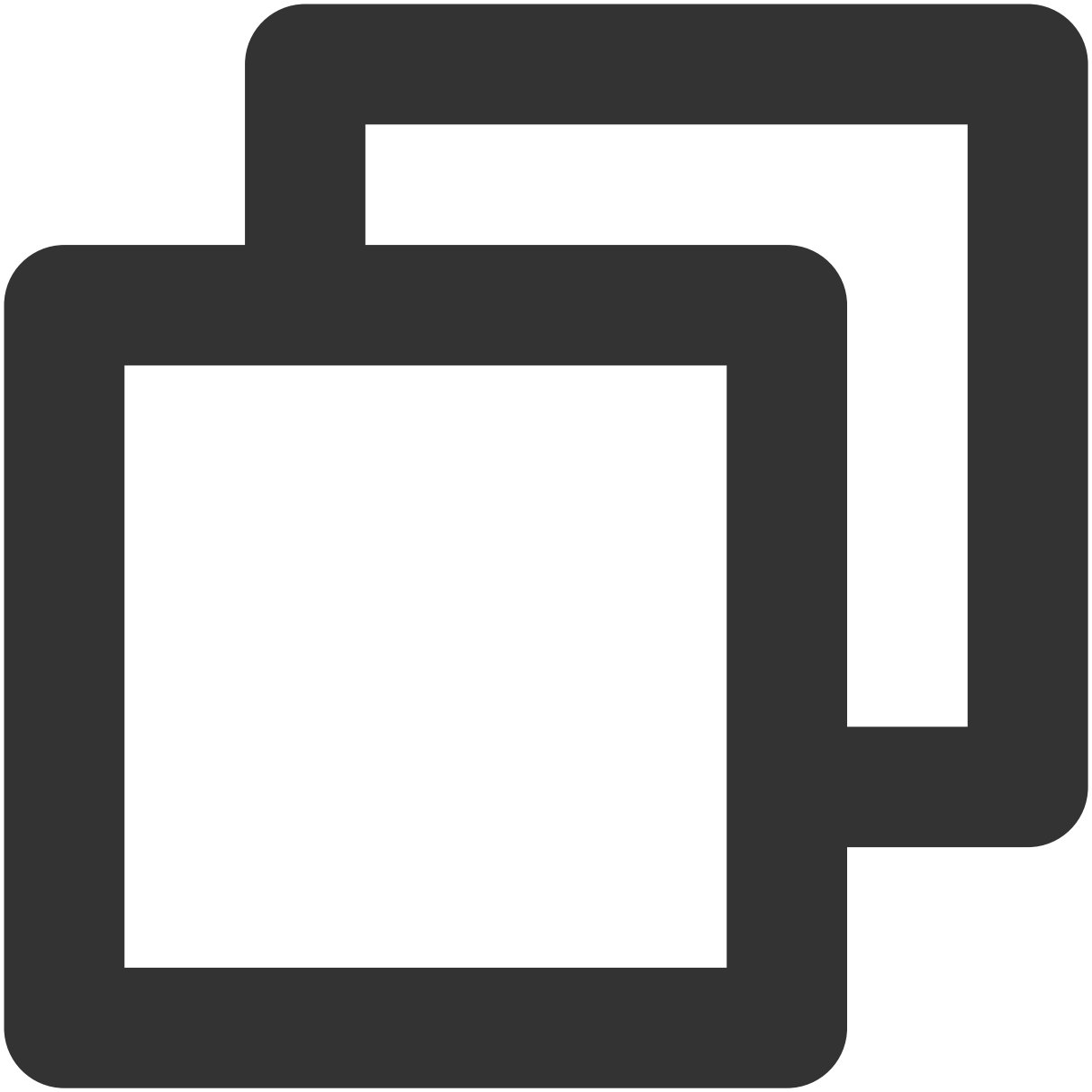
```
// Determine if the current camera is front-facing
BOOL isFrontCamera = [[self.trtcCloud getDeviceManager] isFrontCamera];
// Switch between front and rear cameras, true: switch to front-facing; false: swit
[[self.trtcCloud getDeviceManager] switchCamera:!isFrontCamera];
```

Window Size Switching

Android

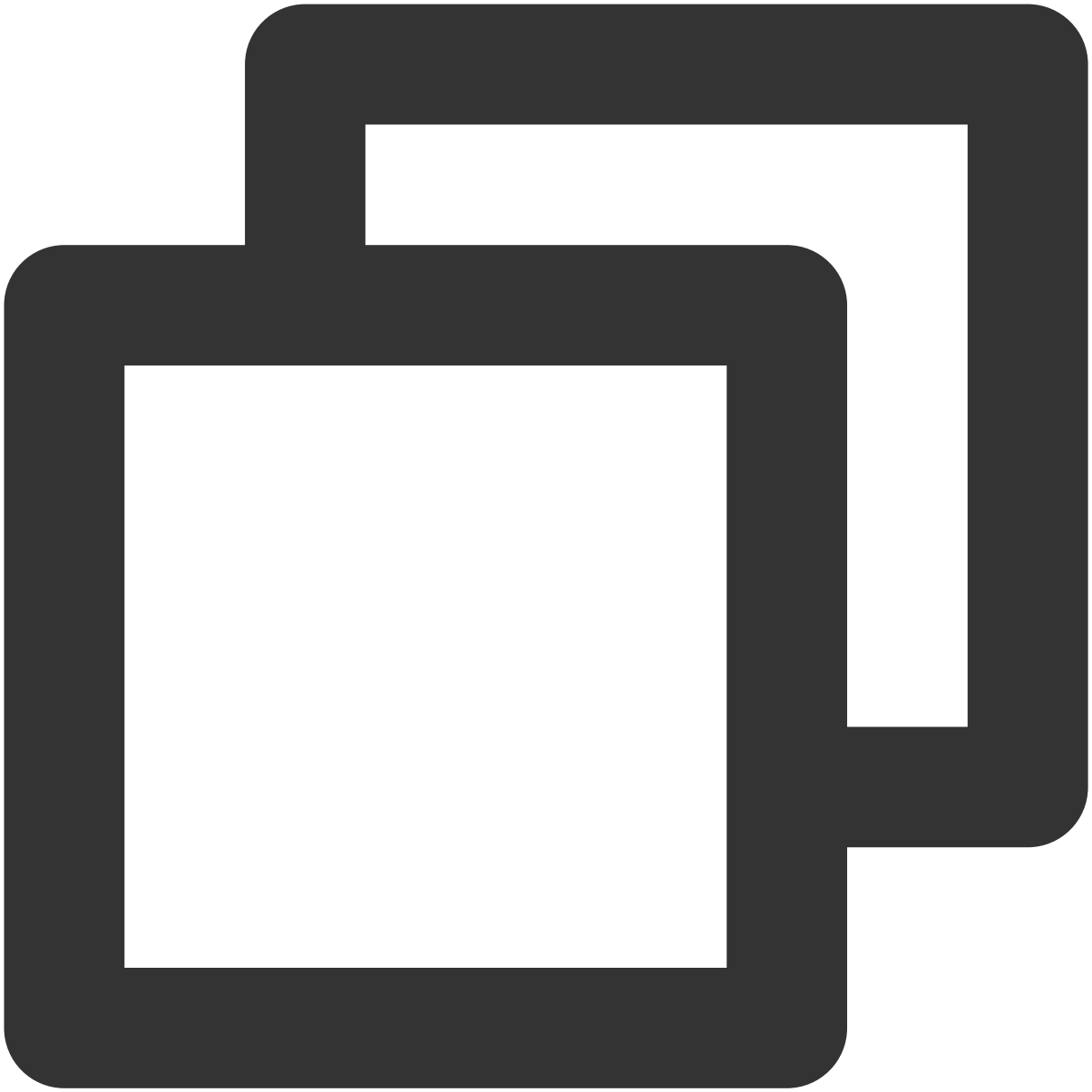
iOS

©2013-2022 Tencent Cloud. All rights reserved.



```
// Update local preview screen rendering control
mTRTCCloud.updateLocalView(previewView);
// Update remote user video rendering control
mTRTCCloud.updateRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, previe
```

©2013-2022 Tencent Cloud. All rights reserved.



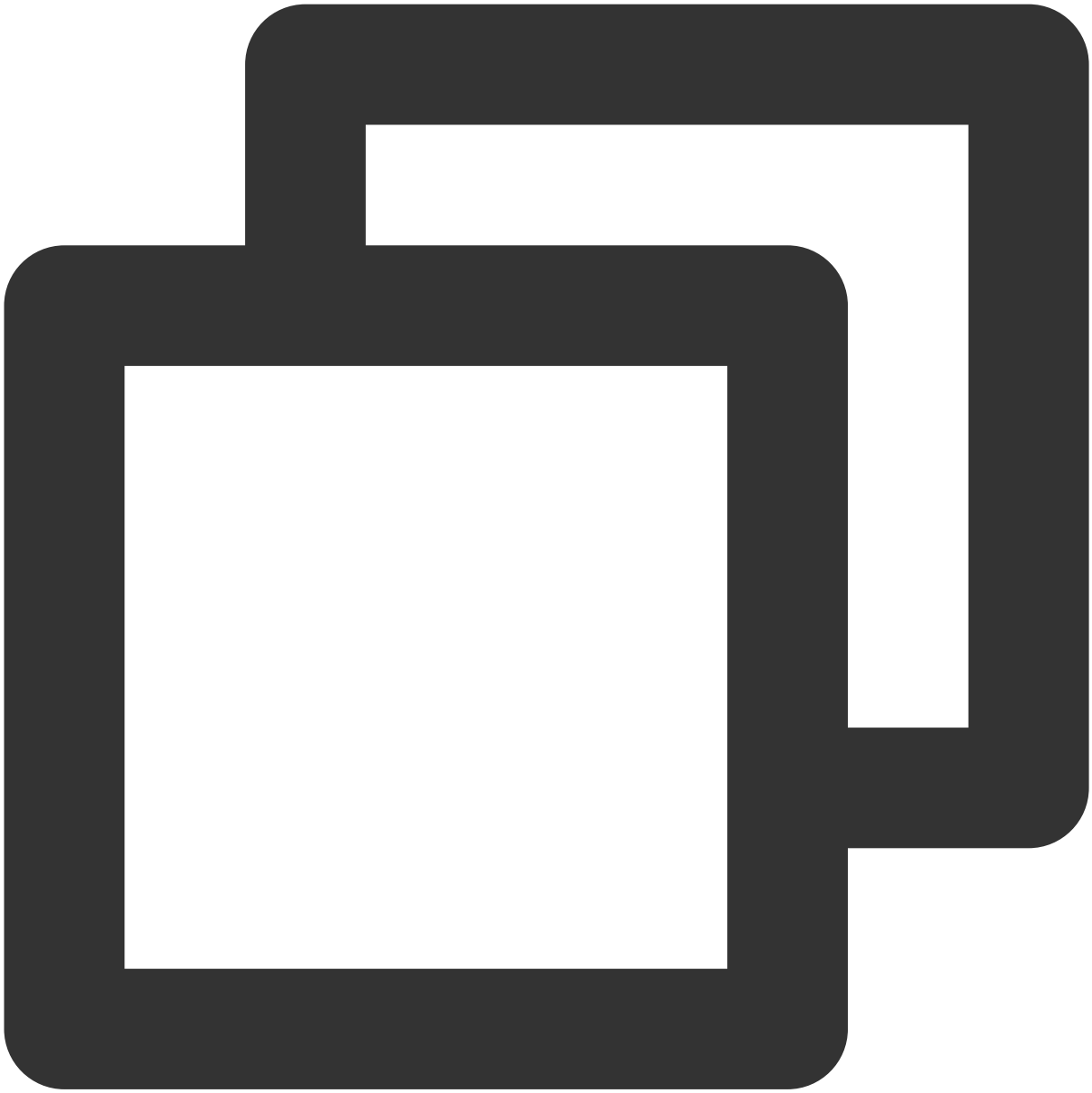
```
// Update local preview screen rendering control
[self.trtcCloud updateLocalView:self.previewView];
// Update remote user video rendering control
[self.trtcCloud updateRemoteView:self.previewView streamType:TRTCVideoStreamTypeBig
```

Network Status Prompt

Android

iOS

©2013-2022 Tencent Cloud. All rights reserved.

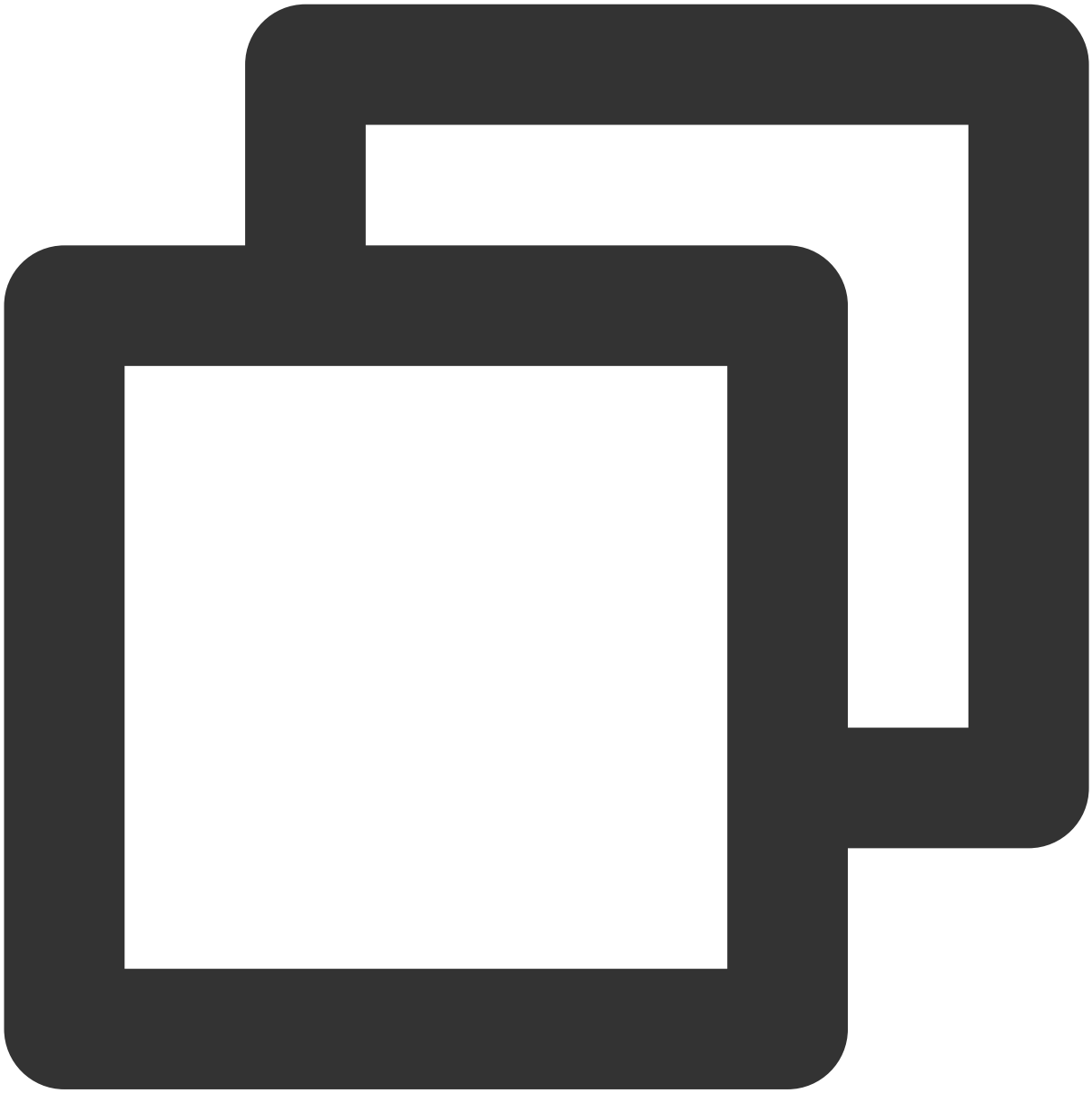


```
@Override
public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality, ArrayList<TRTCC
    if (remoteQuality.size() > 0) {
        switch (remoteQuality.get(0).quality) {
            case TRTCCloudDef.TRTC_QUALITY_Excellent:
                Log.i(TAG, "The other party's network is very good");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Good:
                Log.i(TAG, "The other party's network is quite good");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Poor:
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                Log.i(TAG, "The other party's network is average");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Bad:
                Log.i(TAG, "The other party's network is poor");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Vbad:
                Log.i(TAG, "The other party's network is very poor");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Down:
                Log.i(TAG, "The other party's network is extremely poor");
                break;
            default:
                Log.i(TAG, "Undefined");
                break;
        }
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.



```
#pragma mark - TRTCCloudDelegate

- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<TRT
    if (remoteQuality.count > 0) {
        switch(remoteQuality[0].quality) {
            case TRTCQuality_Unknown:
                NSLog(@"Undefined ");
                break;
            case TRTCQuality_Excellent:
                NSLog(@"The other party's network is very good");
                break;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            case TRTCQuality_Good:
                NSLog(@"The other party's network is quite good");
                break;
            case TRTCQuality_Poor:
                NSLog(@"The other party's network is average");
                break;
            case TRTCQuality_Bad:
                NSLog(@"The other party's network is relatively poor");
                break;
            case TRTCQuality_Vbad:
                NSLog(@"The other party's network is very poor");
                break;
            case TRTCQuality_Down:
                NSLog(@"The other party's network is extremely poor");
                break;
            default:
                break;
        }
    }
}
```

Note:

`localQuality` 's userId field is empty, indicating the local user network quality assessment result.

`remoteQuality` represents the assessment result of the remote user's network quality, which is influenced by factors on both the remote and local sides.

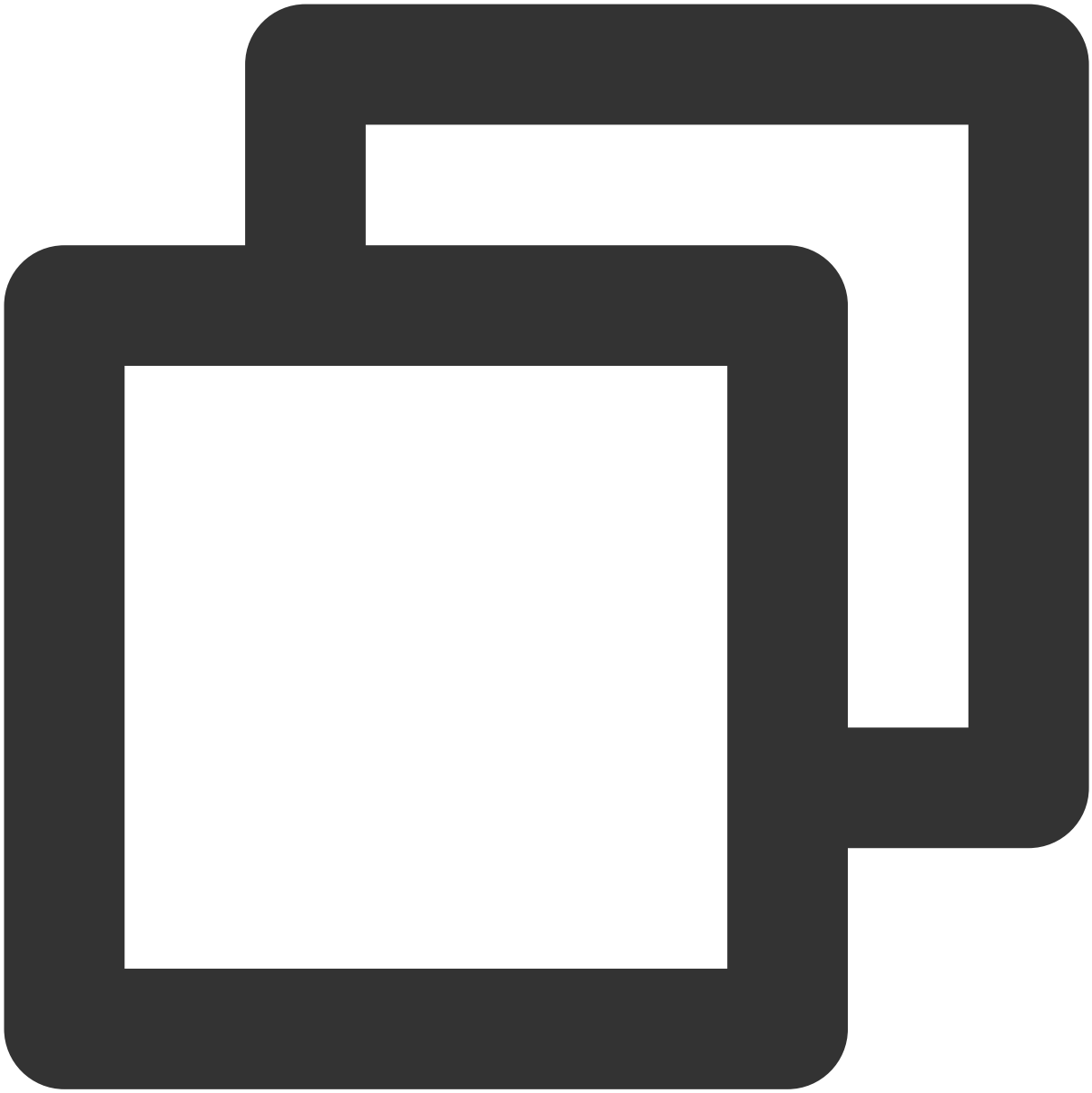
Call duration statistics

It is recommended to use the time when a remote user joins the TRTC (Tencent Real-Time Communication) room as the start time for calculating call duration, and the time when the local user exits the room as the end time for calculating call duration.

Android

iOS

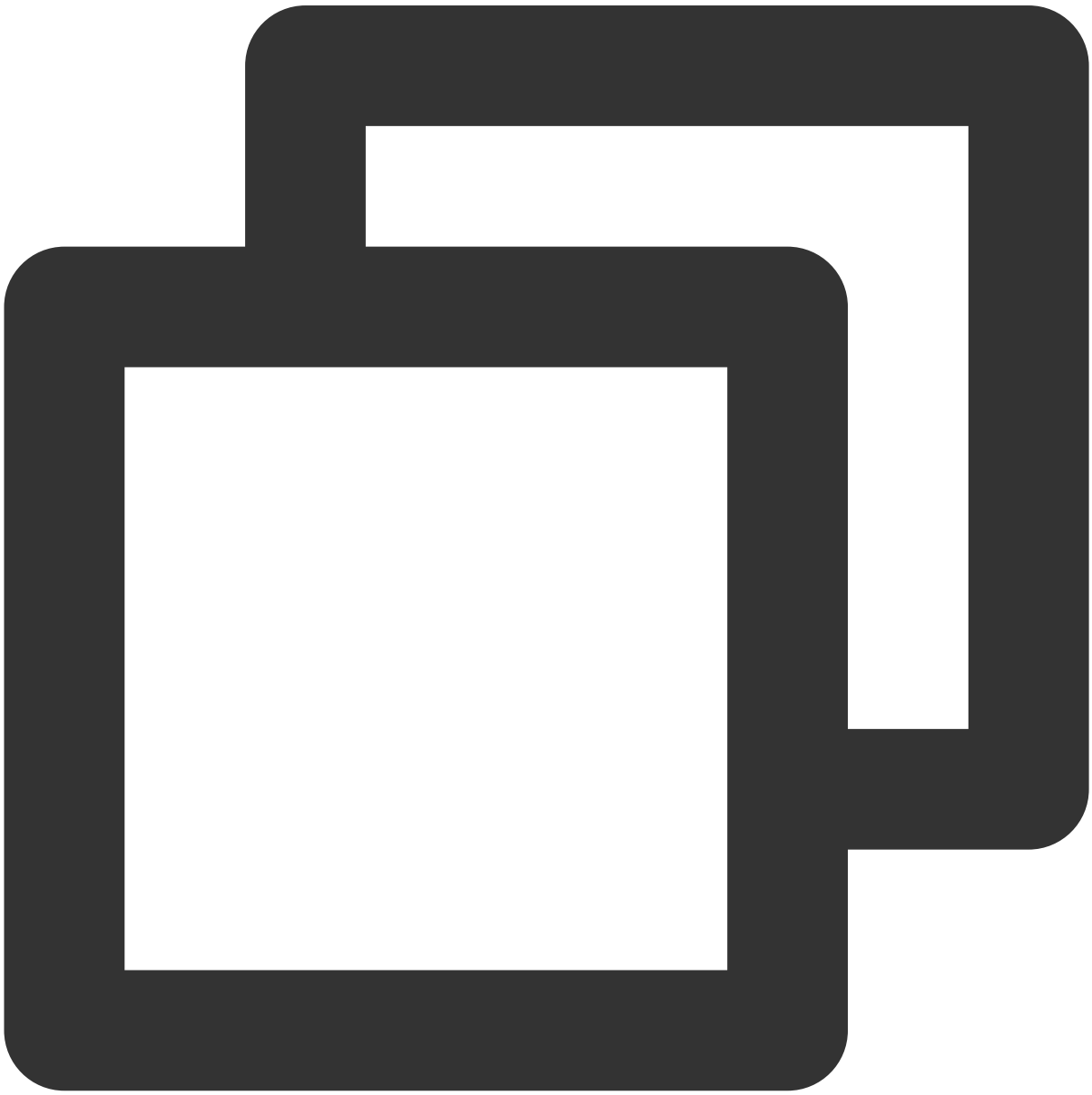
©2013-2022 Tencent Cloud. All rights reserved.



```
// Start call time
long callStartTime = 0;
// End Call Time
long callFinishTime = 0;
// Call Duration (seconds)
long callDuration = 0;

// Remote User Join Callback
@Override
public void onRemoteUserEnterRoom(String userId) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    callStartTime = System.currentTimeMillis();
}

// Local User Leave Callback
@Override
public void onExitRoom(int reason) {
    callFinishTime = System.currentTimeMillis();
    callDuration = (callFinishTime - callStartTime) / 1000;
}
```



```
// Start call time
```

©2013-2022 Tencent Cloud. All rights reserved.

```
@property (nonatomic, assign) NSTimeInterval callStartTime;
// End Call Time
@property (nonatomic, assign) NSTimeInterval callFinishTime;
// Call Duration (seconds)
@property (nonatomic, assign) NSInteger callDuration;

// Remote User Join Callback
- (void)onRemoteUserEnterRoom:(NSString *)userId {
    self.callStartTime = [[NSDate date] timeIntervalSince1970];
}

// Local User Leave Callback
- (void)onExitRoom:(NSInteger)reason {
    self.callFinishTime = [[NSDate date] timeIntervalSince1970];
    self.callDuration = (NSInteger)(self.callFinishTime - self.callStartTime);
}
```

Note:

In cases of exceptions such as forced closure or network disconnection, the client may not be able to log the relevant times. These can be monitored through Server-side Event Callback to track events of entering and exiting the room and calculate the duration of the call.

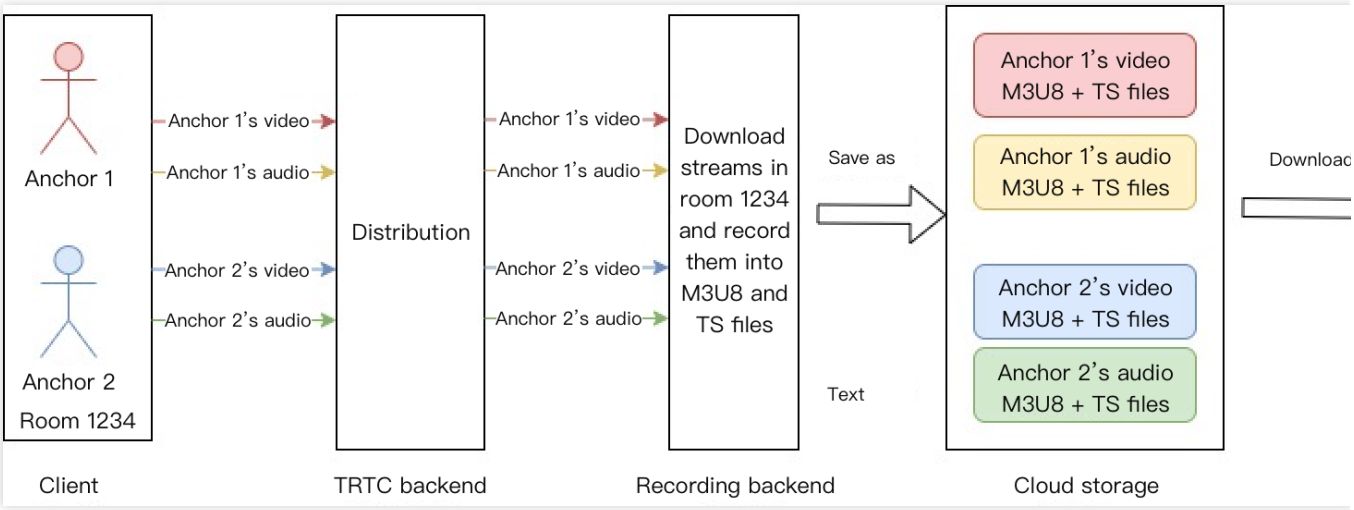
Advanced Features

On-Cloud Recording

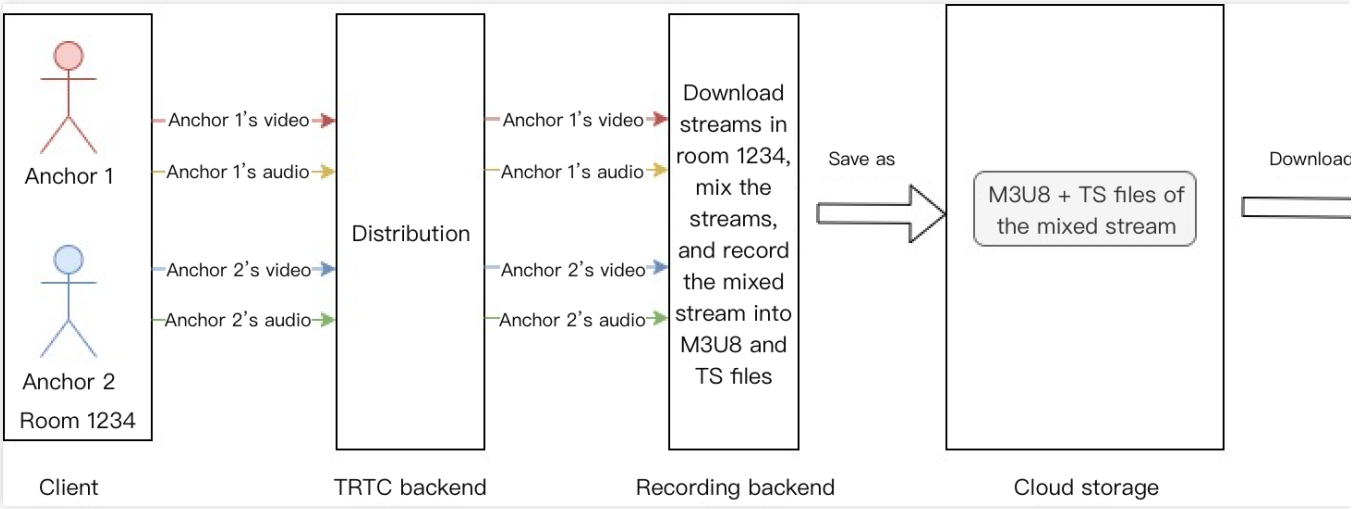
In many scenes of 1V1 Audio and Video Call, it is necessary to record and store the content of the call for filing and post-event analysis. TRTC (Tencent Real-Time Communication)'s latest upgrade to on-cloud recording, which doesn't rely on CSS (Cloud Streaming Services) capabilities and doesn't require rerouting to CSS, uses TRTC (Tencent Real-Time Communication)'s internal real-time recording cluster for audio and video recording, offering a more complete and unified recording experience.

Single Stream Recording:Through TRTC (Tencent Real-Time Communication)'s on-cloud recording feature, you can record the audio and video streams of both parties in the room into separate files.

©2013-2022 Tencent Cloud. All rights reserved.



Mixed Stream Recording: Record all the audio and video media streams in the same room into one file.



Note:

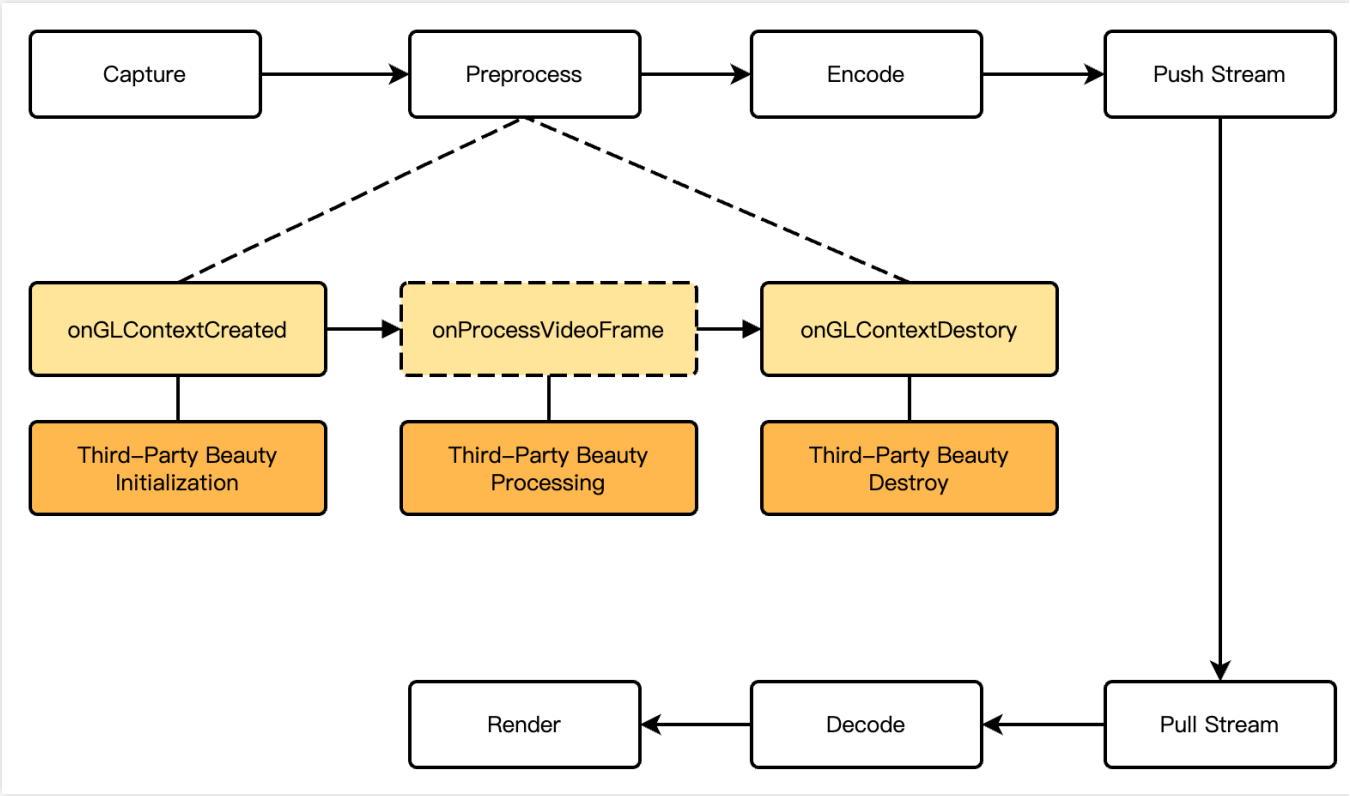
For a detailed introduction and activation guide to TRTC On-Cloud Recording, see On-Cloud Recording.

Video Beauty Effects

In video call scenes, beauty effects are a frequently used feature. Not only can beauty effects enhance the user's appearance, but they also add interest to the call interaction through various sticker effects. TRTC (Tencent Real-Time Communication) supports the integration of Tencent Beauty Special Effects and also supports the connect to mainstream third-party beauty products in the market, such as Volcano Beauty, Xiangxin Beauty, etc.

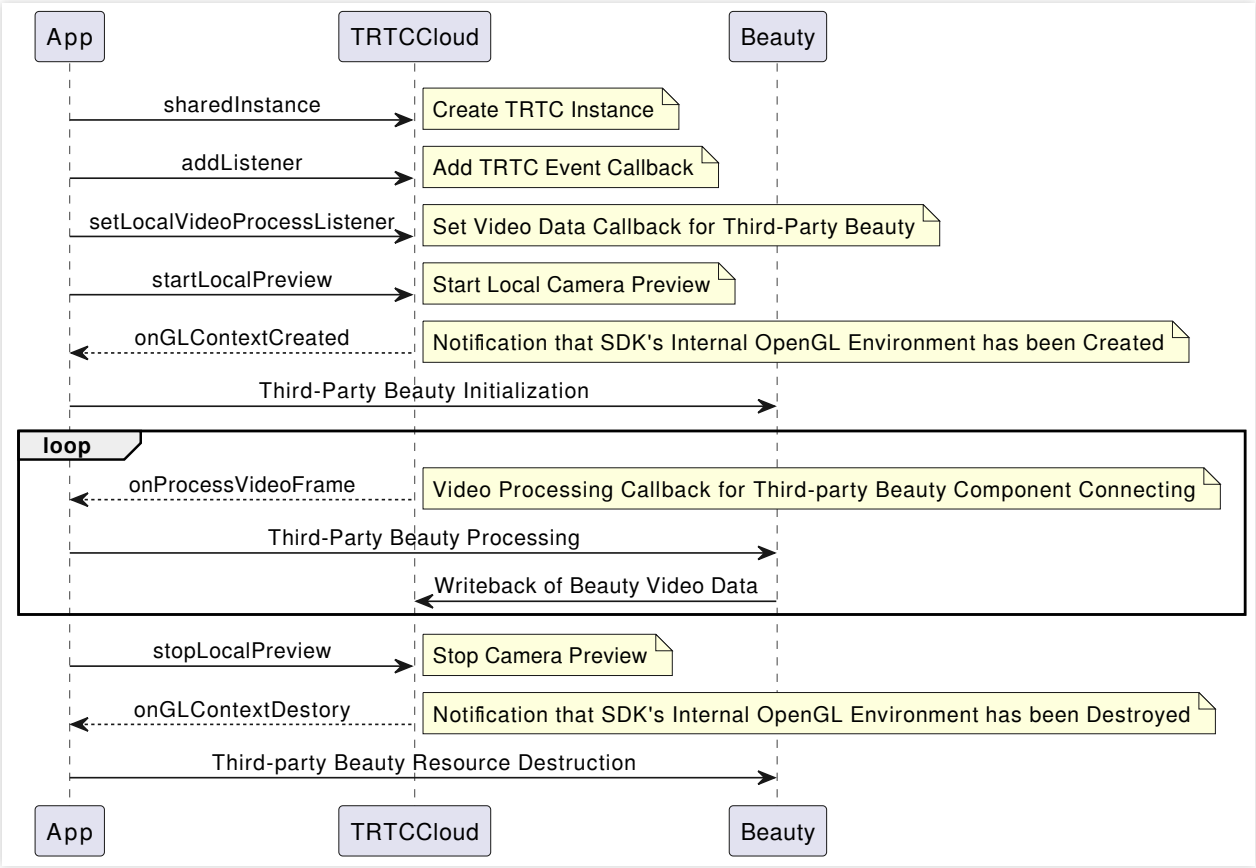
Beauty Enhancement Connect Process

©2013-2022 Tencent Cloud. All rights reserved.



API Call Sequence

©2013-2022 Tencent Cloud. All rights reserved.



Comparison of Beauty Enhancement Products

Beauty Type	Beauty Effect	Integration Cost	Fees	Virtual AI Digital Human	Support Terminal
Tencent Effect SDK	The basic effect is good, advanced effect for big eyes/slim faces is significant.	Moderately Low	Moderate	Supported	Android/iOS/PC/Flutter/Web/Mini Program
FaceUnity Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are average.	Moderately High	Moderate	Supported	Android/iOS/PC/Untiy

©2013-2022 Tencent Cloud. All rights reserved.

Volcano Effect SDK	The basic effect is good, advanced effects like big eyes/slim faces are relatively good.	Moderately High	Relatively High	Supported	Android/iOS/PC/Linux

Offline Message Push

In Audio/Video Call scenes, the offline message push feature is usually necessary, allowing the called user's App to receive new incoming call messages even when it's not online.
Chat provides a complete Android Offline Push Integration Guide, iOS Offline Push Integration Guide, with the main steps as follows:
Android Offline Push
iOS Offline Push
1. Register your application with vendor push platforms.
2. Configure the IM console.
3. Configure the redirected-to page for offline push.
4. Configure vendor push rules.
5. Integrate the vendor push SDK.
6. Sync frontend and backend status.
7. Send offline push messages.
8. Parse offline push messages.
1. Apply for an APNs/VoIP Push certificate.
2. Upload the certificate to the IM console.
3. The app requests a token from Apple's backend.
4. Log in to the IM SDK and then upload the token to Tencent Cloud.
5. Send offline push messages.
6. Parse offline push messages.

Supporting Products for the Solution

System Level	Product Name	Application Scenes
Access	Tencent Real-	Provides low-latency, high-quality real-time audio and video interaction

©2013-2022 Tencent Cloud. All rights reserved.

Layer	Time Communication (TRTC)	solutions, which are the basic infrastructure capabilities for Audio/Video Call scenes.
Access Layer	Instant Messaging (IM)	Provides reliable and stable signaling transmission, custom message sending and receiving, to implement call signaling control in Audio/Video Call scenes.
Access Layer	Tencent Effect SDK	Provides real-time effects processing capabilities such as beauty, filtering, makeup, fun stickers, emojis, and virtual avatars.
Cloud Services	Video on Demand (VOD)	Aimed at audio, video, and images, it provides an all-in-one high-quality media service including production upload, storage, transcoding, MPS (Media Processing Service), media AI, accelerated distribution and playback, and copyright protection.
Data Storage	Cloud Object Storage (COS)	Provides storage services for audio and video recording files, as well as audio and video slicing files.

©2013-2022 Tencent Cloud. All rights reserved.

Quick Integration Guide
Android

Last updated：2024-07-18 14:26:14

Business Process

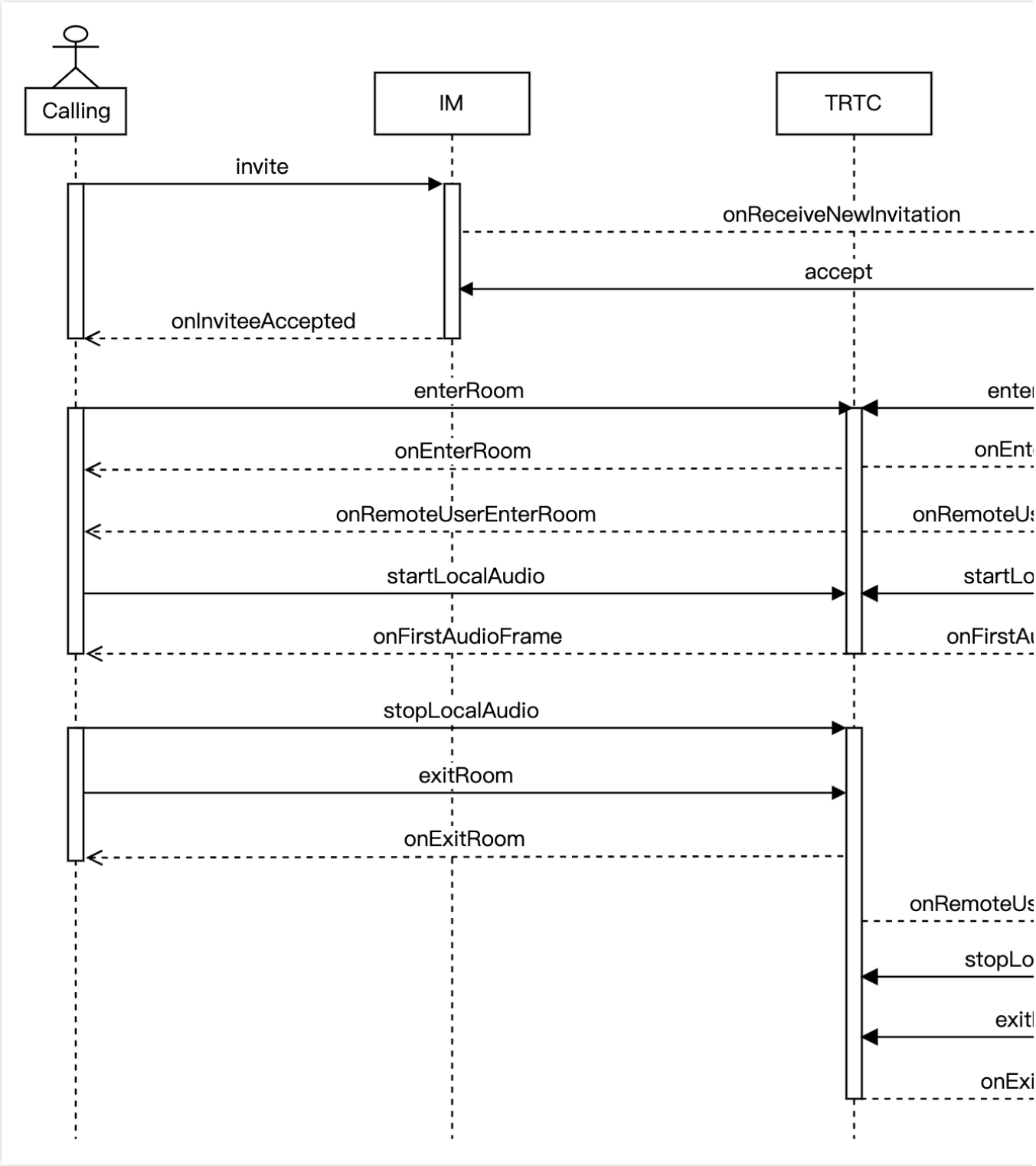
This document summarizes some common business processes in one-to-one audio and video calls, helping you better understand the implementation process of the entire scenario.

Audio Call Process

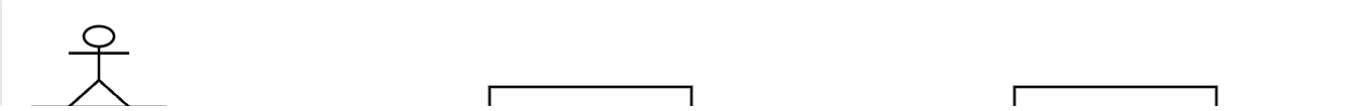
Video Call Process

The following diagram shows the sequence of one-to-one audio call, including processes such as calling, answering, talking, and hanging up.

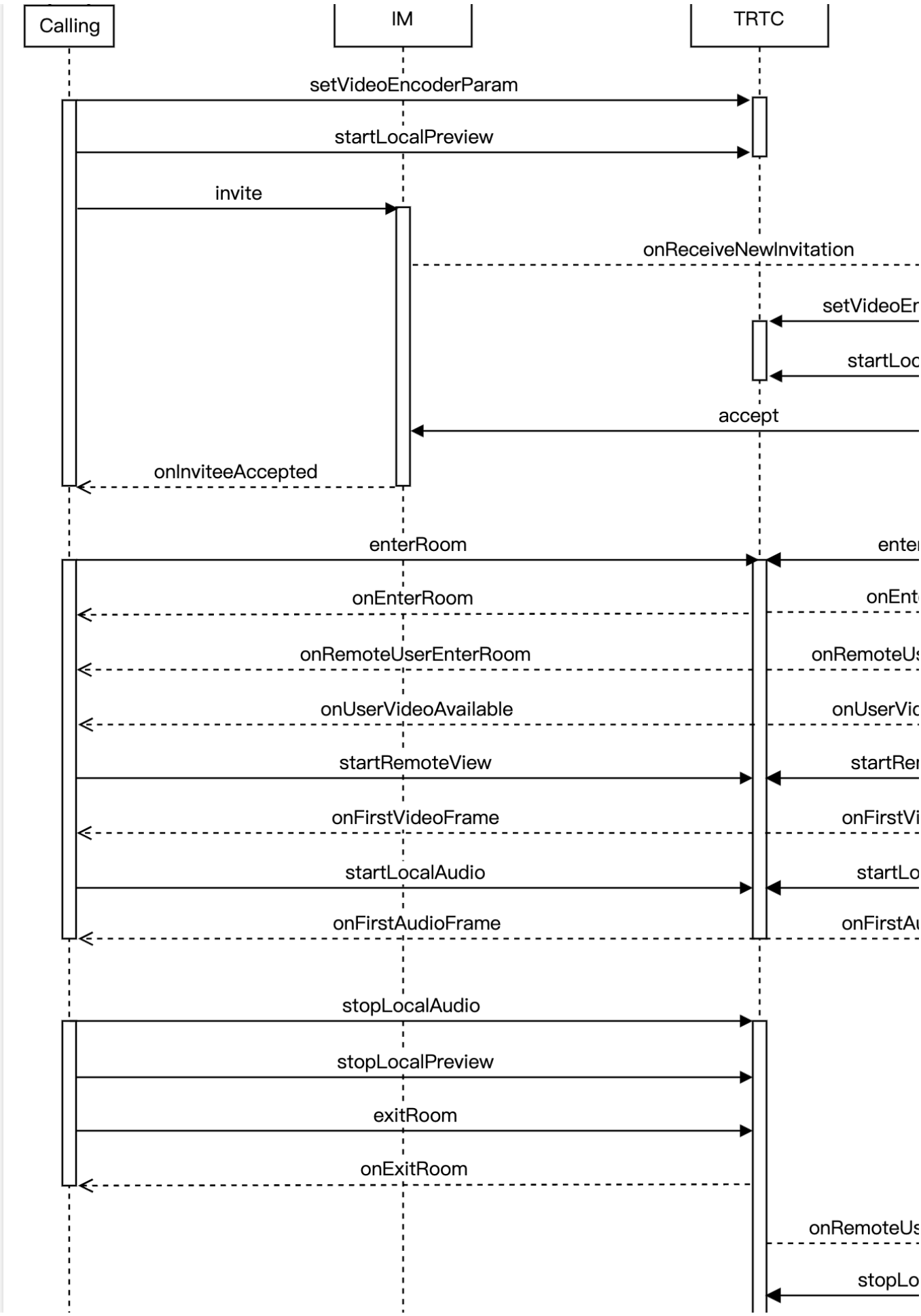
©2013-2022 Tencent Cloud. All rights reserved.



The following diagram shows the sequence of one-to-one video call, including processes such as calling, answering, talking, and hanging up.



©2013-2022 Tencent Cloud. All rights reserved.



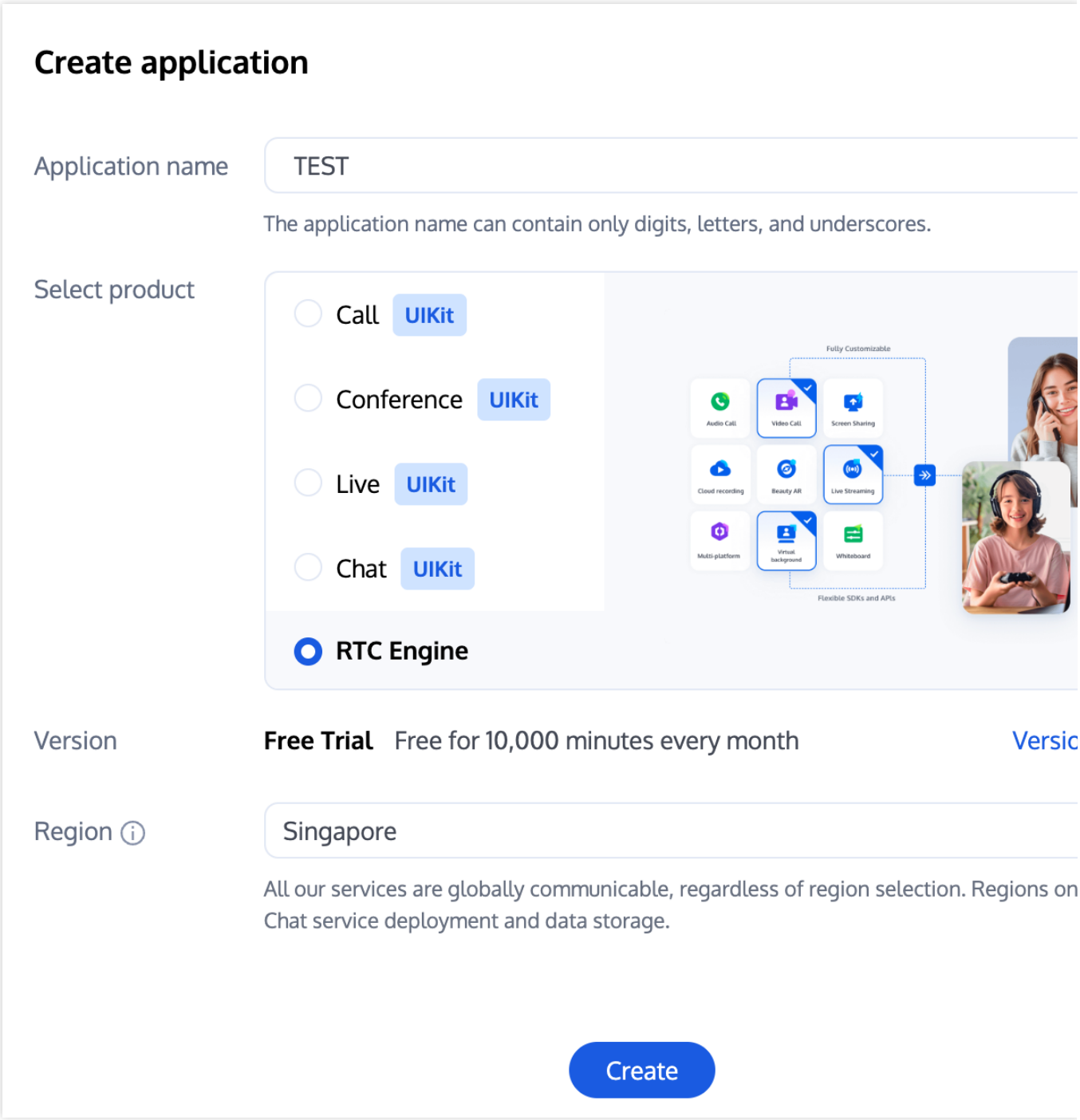
©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparations

Step 1: activate the service

One-to-one audio and video call scenarios usually require dependencies on two paid PaaS services from the cloud platform, Instant Messaging (IM) and Real-Time Communication (TRTC) for construction.
1. First, you need to log in to the TRTC Console to create an application. At this time, an IM trial application with the same SDKAppID as the current TRTC application will be automatically created in the Instant Messaging (IM) Console. The accounts and authentication systems for both can be reused. Subsequently, you can choose to upgrade the TRTC or IM application version as needed. For example, the advanced versions can unlock more value-added features and services.

©2013-2022 Tencent Cloud. All rights reserved.



Note:

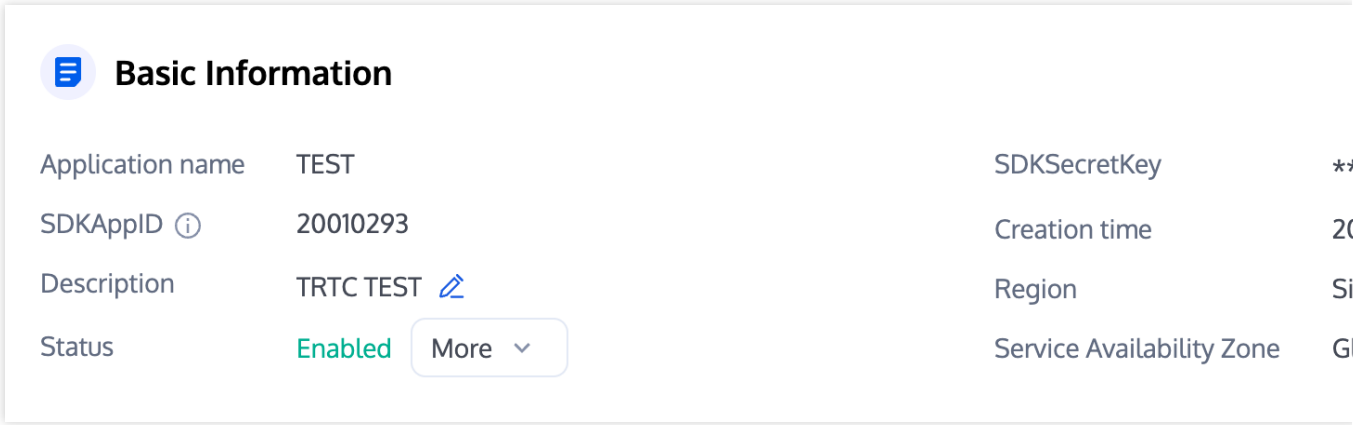
It is recommended to create two separate applications for testing and production environments. Each account (UIN) is provided with 10,000 minutes of free usage per month within one year.

The TRTC monthly package is divided into Trial Version (by default), Basic Version, and Professional Version, which can unlock different value-added features and services. For details, see Version Features and Monthly Package Description.

2. Once the application is created, you can find basic information about it under the **Application Management - Application Overview** section. It is important to store the **SDKAppID** and **SDKSecretKey** for later use and to avoid

©2013-2022 Tencent Cloud. All rights reserved.

key leakage to prevent unauthorized traffic usage.

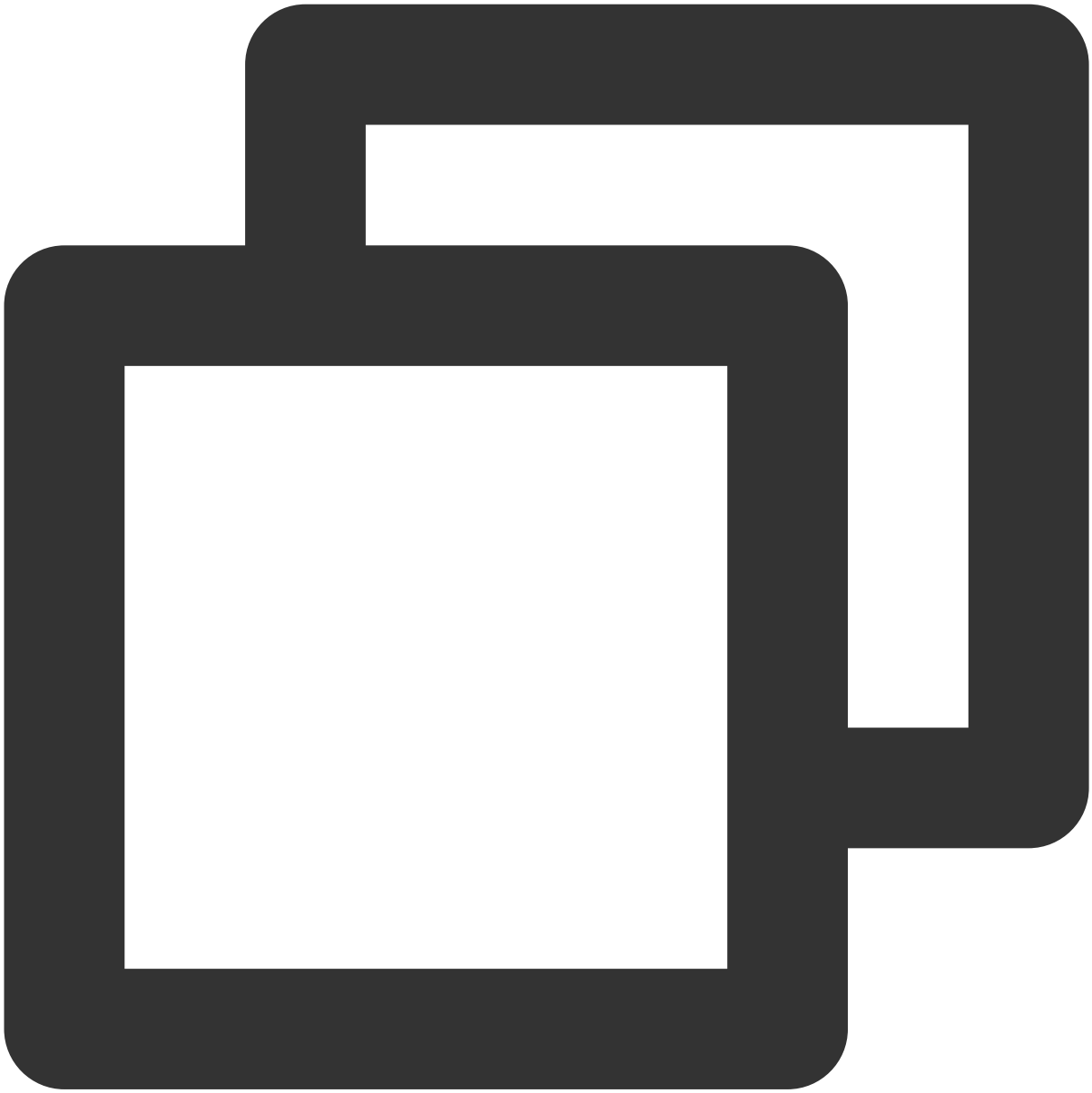


Step 2: import SDK

The TRTC SDK and IM SDK have been released to the **mavenCentral** repository. You can configure gradle to download and update automatically.

1. Add the dependency for the appropriate version of the SDK in dependencies.

©2013-2022 Tencent Cloud. All rights reserved.



```
dependencies {
    // TRTC SDK lite version, includes two features: TRTC and live streaming playba
    implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'

    // Add IM SDK. It is recommended to use the latest version number
    implementation 'com.tencent.imsdk:imsdk-plus:Version number'

    // If you need to add Quic plugin, uncomment the next line (Note: Version numbe
    // implementation 'com.tencent.imsdk:timquic-plugin:Version number'
}
```

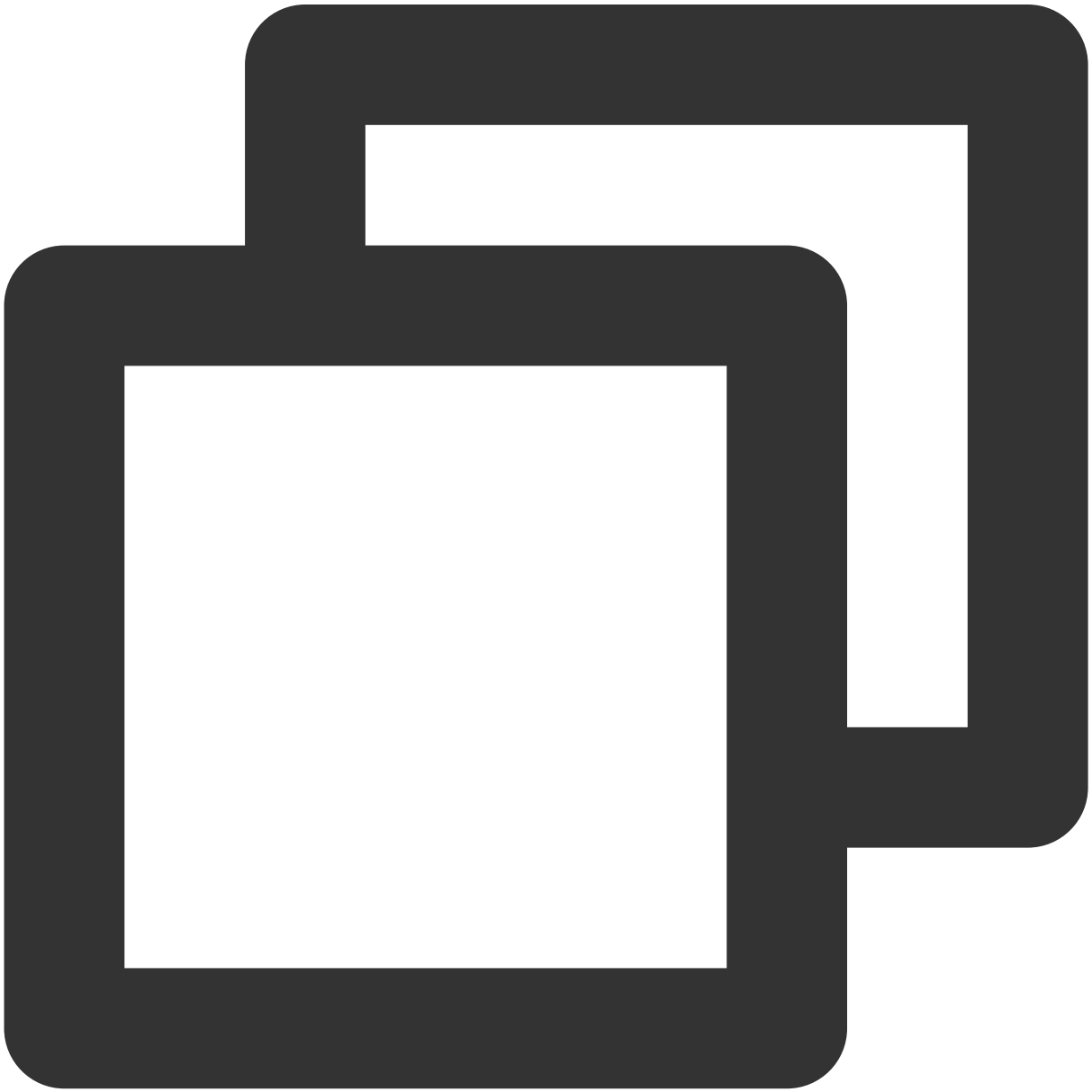
©2013-2022 Tencent Cloud. All rights reserved.

Note:

Besides the recommended automatic loading method, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manually Integrating the IM SDK.

Quic plugin offers axp-quic Multiplexing Transmission Protocol, providing better resistance to poor networks. Even with a packet loss rate of 70%, it still can offer services. **Available only for Flagship users**. For non-Flagship users, purchase the Flagship package before use, and see Pricing Instructions. To ensure proper functionality, update **Terminal SDK** to **version 7.7.5282 or above**.

2. Specify the CPU architecture used by the app in defaultConfig.



```
defaultConfig {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        ndk {
                abiFilters "armeabi-v7a", "arm64-v8a"
        }
    }
```

Note:

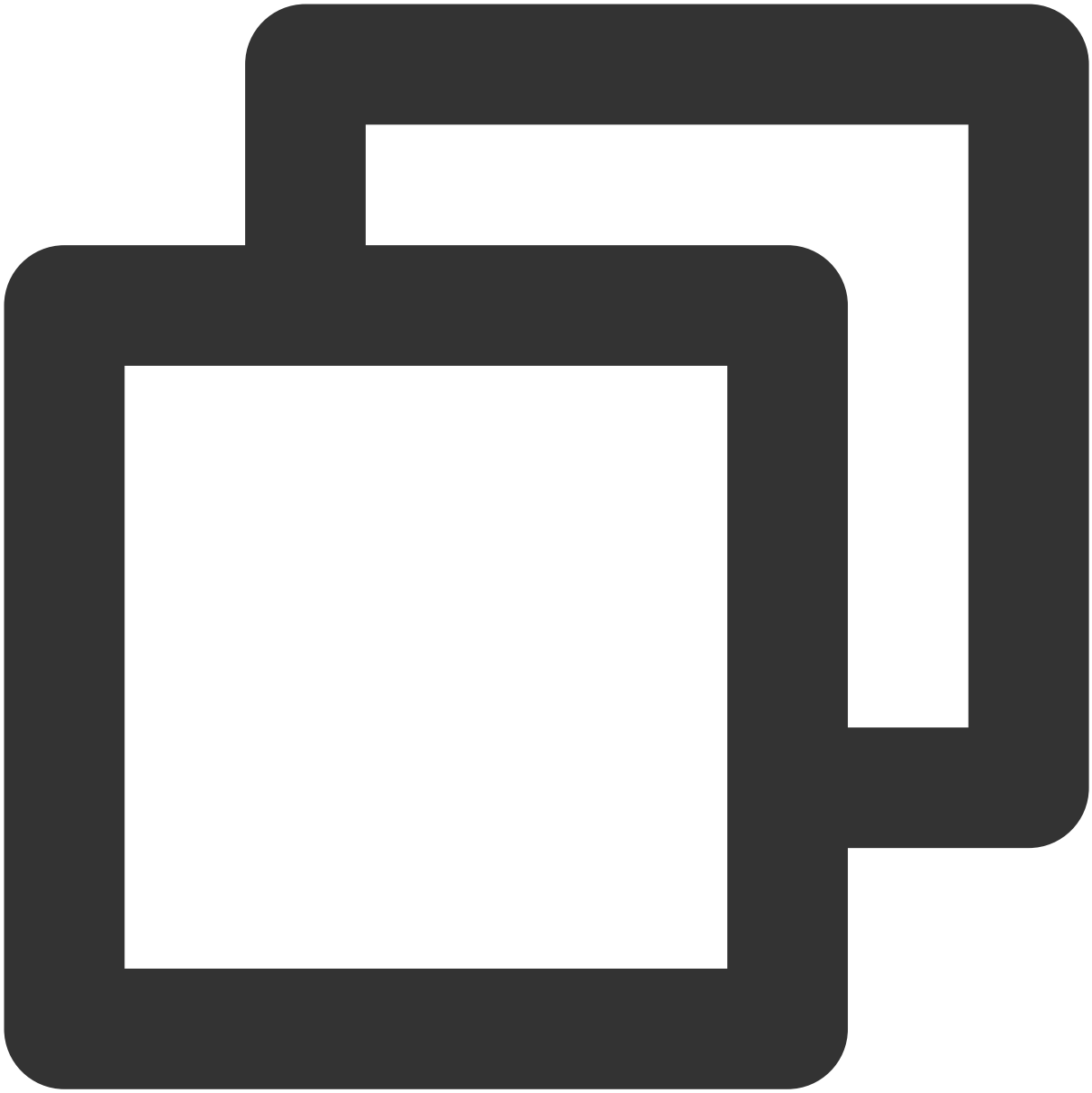
The TRTC SDK supports architectures including armeabi, armeabi-v7a and arm64-v8a. Additionally, it supports architectures for simulators including x86 and x86_64.

The IM SDK supports architectures including armeabi-v7a, arm64-v8a, x86, and x86_64. To reduce the size of the installer package, you can choose to package SO files for only a subset of these architectures.

Step 3: project configuration

1. To configure app permissions in AndroidManifest.xml, for audio/video call scenarios, both the TRTC SDK and IM SDK require the following permissions:

©2013-2022 Tencent Cloud. All rights reserved.



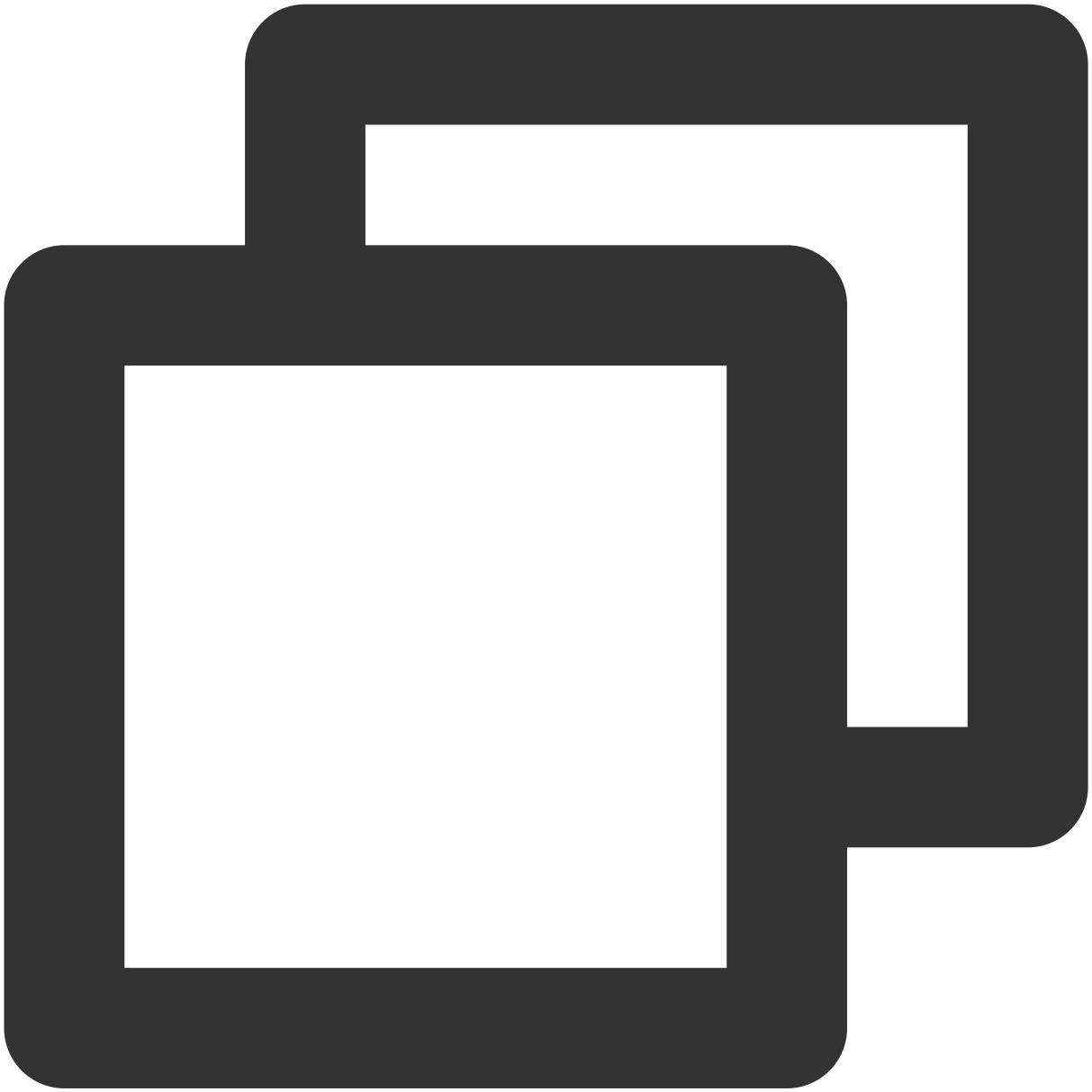
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

Do not set `android:hardwareAccelerated="false"` . Disabling hardware acceleration will result in failure to render the other party's video stream.
The TRTC SDK does not have built-in permission request logic. You need to declare the corresponding permissions yourself. Some permissions (such as storage, recording and camera), also require runtime dynamic requests.
If the Android project's `targetSdkVersion` is 31 or higher, or if the target device runs Android 12 or a newer version, the official requirement is to dynamically request `android.permission.BLUETOOTH_CONNECT` permission in the code to use the Bluetooth feature properly. For more information, see Bluetooth Permissions.
2. Since we use Java's reflection features inside the SDK, you need to add relevant SDK classes to the non-obfuscation list in the proguard-rules.pro file:

©2013-2022 Tencent Cloud. All rights reserved.



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

Step 4: authentication credential

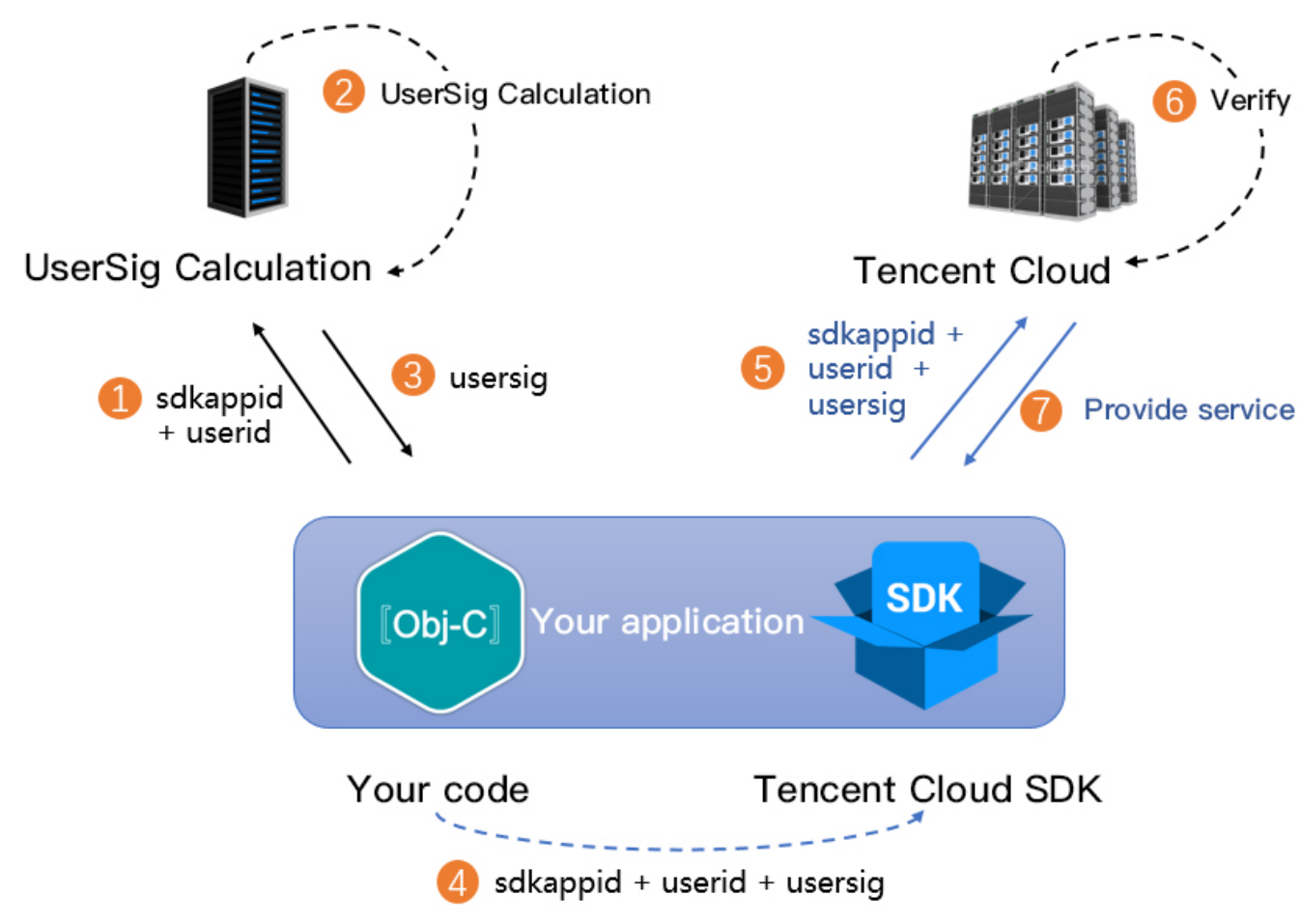
UserSig is a security signature designed by the cloud platform to prevent attackers from accessing your cloud account. Cloud services such as Real-Time Communication (TRTC) and Instant Messaging (IM) adopt this security protection mechanism. Authentication is required for TRTC upon entering a room, and for IM during login.
Debugging and testing stage: UserSig can be generated through Client Example Code and Console Access, which are only used for debugging and testing.

©2013-2022 Tencent Cloud. All rights reserved.

Production stage: It is recommended to use the server computing UserSig solution, which has a higher security level and helps prevent the client from being decompiled and reversed, to avoid the risk of key leakage.

The specific implementation process is as follows:

1. Before calling the initialization API of the SDK, your app must first request UserSig from your server.
2. Your server generates the UserSig based on the SDKAppID and UserID.
3. The server returns the generated UserSig to your app.
4. Your app sends the obtained UserSig to the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to the cloud server for verification.
6. The cloud platform verifies the validity of the UserSig.
7. Once the verification is passed, it will provide instant communication services to the IM SDK and real-time audio and video services to the TRTC SDK.



Note:

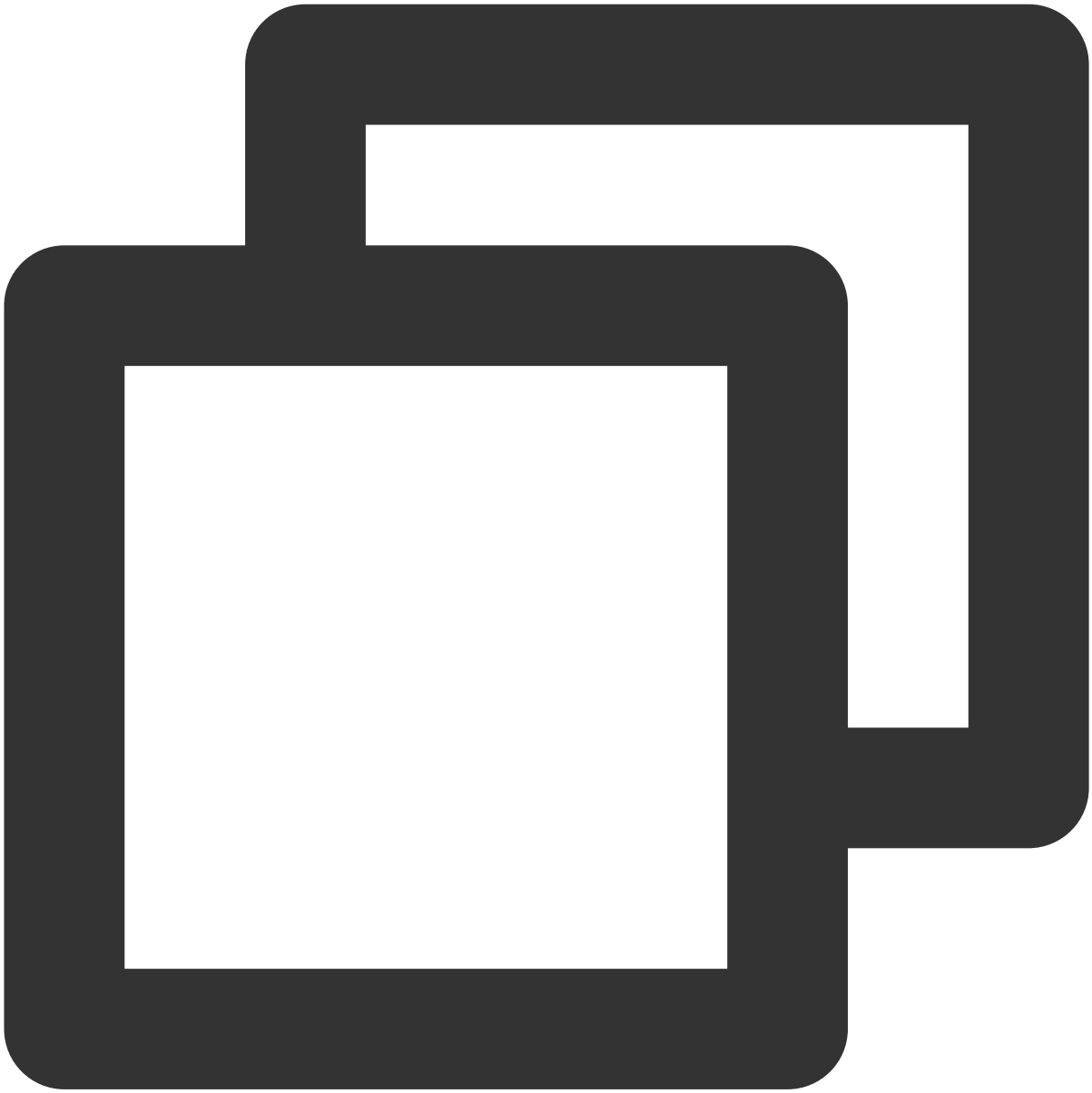
The method of generating UserSig locally during the debugging and testing stage is not recommended for the online environment because it may be easily decompiled and reversed, causing key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

©2013-2022 Tencent Cloud. All rights reserved.

Step 5: initialize the SDK

1. Initialize IM SDK and Add Event Listeners



```
// Add event listener
V2TIMManager.getInstance().addIMSDKListener(imSdkListener);
// Initialize the IM SDK. After calling this API, you can immediately call the log-
V2TIMManager.getInstance().initSDK(context, sdkAppID, null);

// After the SDK is initialized, it will trigger various events, such as connection
private V2TIMSDKListener imSdkListener = new V2TIMSDKListener() {
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

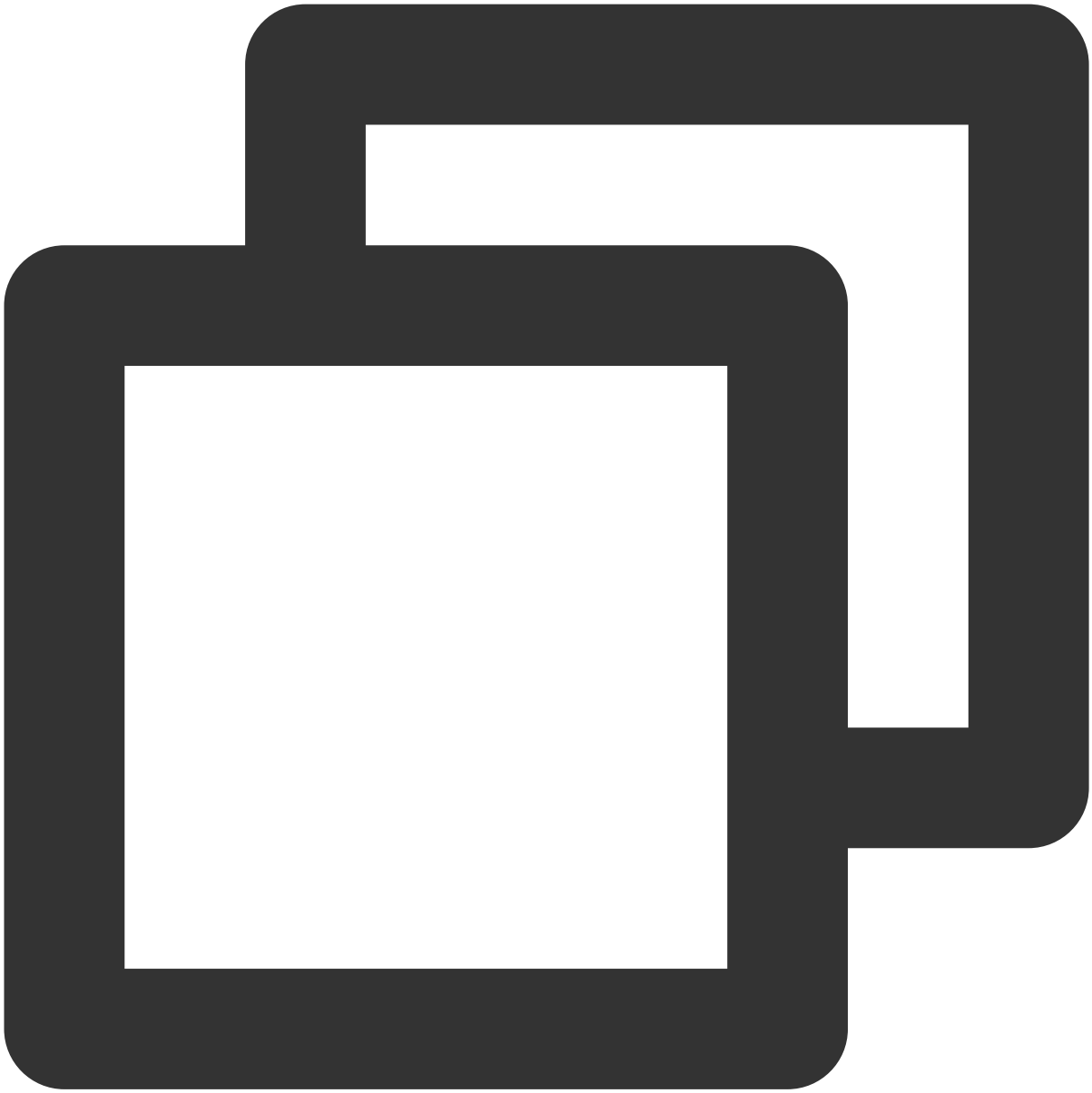
```
    public void onConnecting() {
        Log.d(TAG, "IM SDK is connecting to the Cloud Virtual Machine");
    }

    @Override
    public void onConnectSuccess() {
        Log.d(TAG, "IM SDK has successfully connected to the Cloud Virtual Machine"
    }
};

// Remove event listener
V2TIMManager.getInstance().removeIMSDKListener(imSdkListener);
// Deinitialize the IM SDK
V2TIMManager.getInstance().unInitSDK();
```

Note:

If the lifecycle of your application is consistent with the SDK lifecycle, you do not need to deinitialize before exiting the application. However, if you only initialize the SDK when entering a specific interface and no longer use it after exiting that interface, you can deinitialize the SDK.

2. Create TRTC SDK Instances and Set Event Listeners

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (single instance pattern)
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(context);
// Add TRTC event listener
mTRTCCloud.addListener(trtcSdkListener);

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
private TRTCCloudListener trtcSdkListener = new TRTCCloudListener() {
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        Log.d(TAG, errCode + errMsg);
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        Log.d(TAG, warningCode + warningMsg);
    }
};

// Remove TRTC event listener
mTRTCCloud.removeListener(trtcSdkListener);
// Destroy TRTC SDK instance (single instance pattern)
TRTCCloud.destroySharedInstance();
```

Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

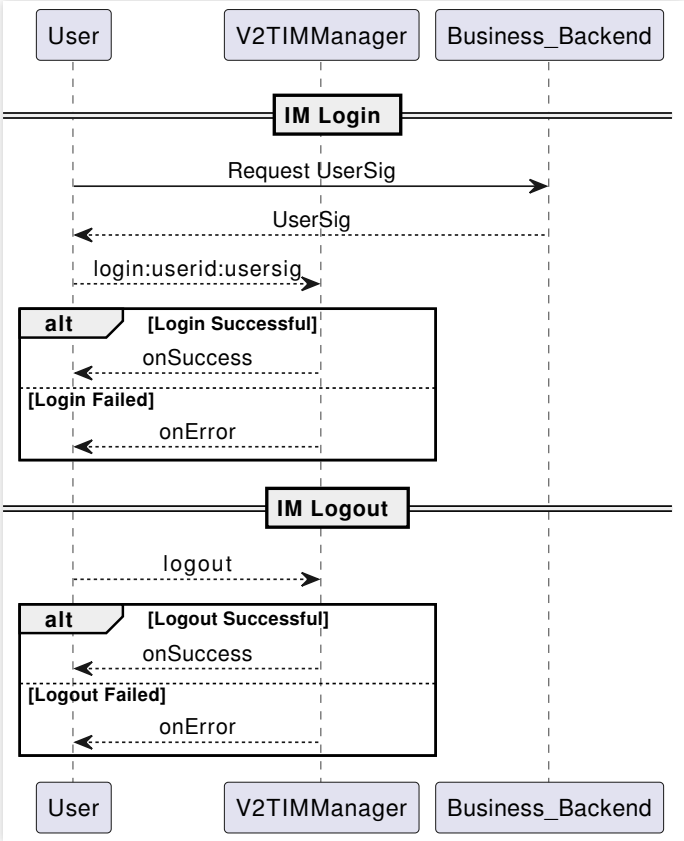
Integration Process

Step 1: log in

After the IM SDK is initialized, you need to call the SDK log in API to authenticate your account identity and gain permissions to use features. **Before using any other features, ensure you are successfully logged in**, or you might encounter feature malfunctions or unavailability. If you only need to use TRTC's audio and video services, you can skip this step.

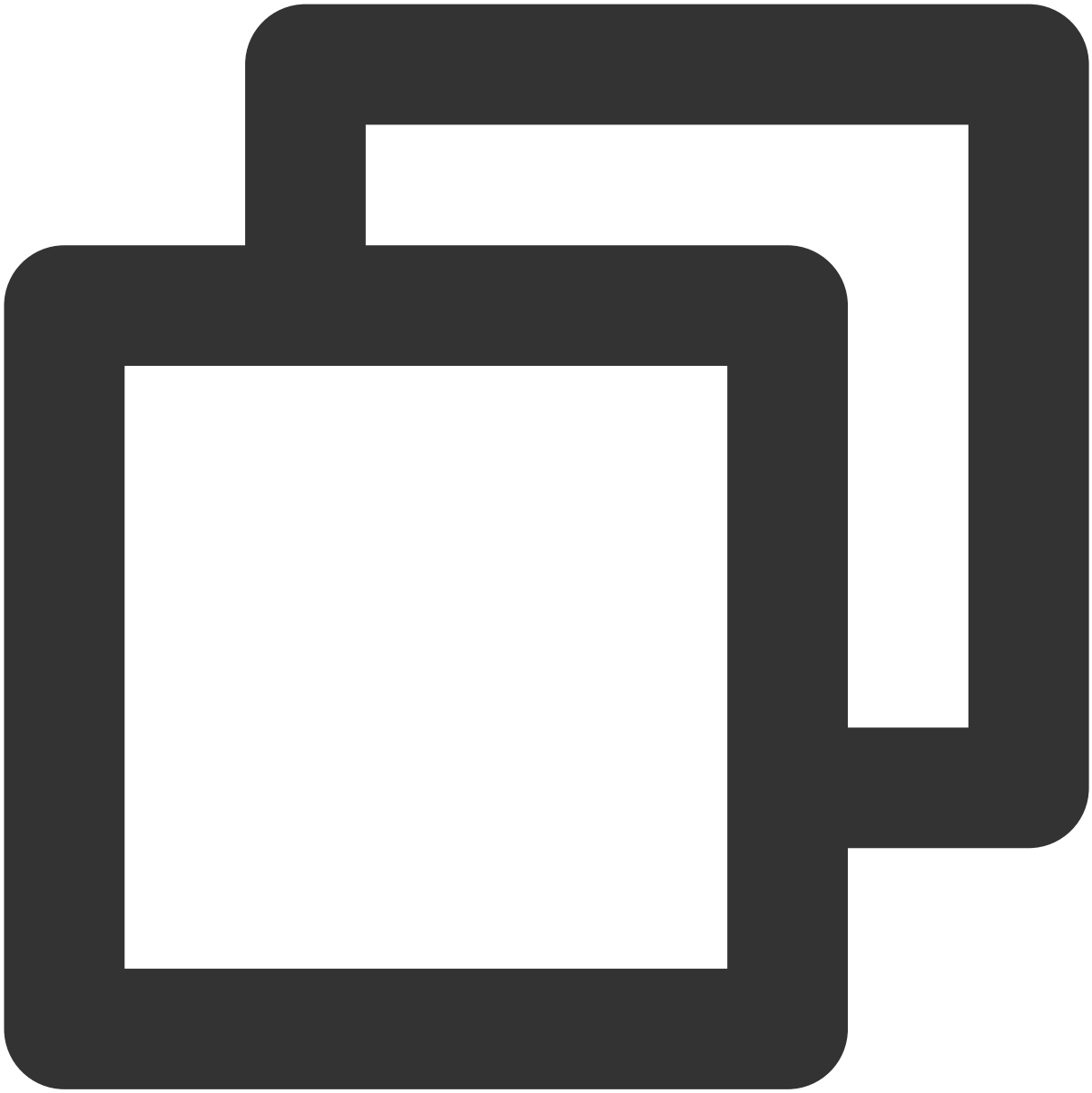
Sequence Diagram

©2013-2022 Tencent Cloud. All rights reserved.



Log in operation

©2013-2022 Tencent Cloud. All rights reserved.

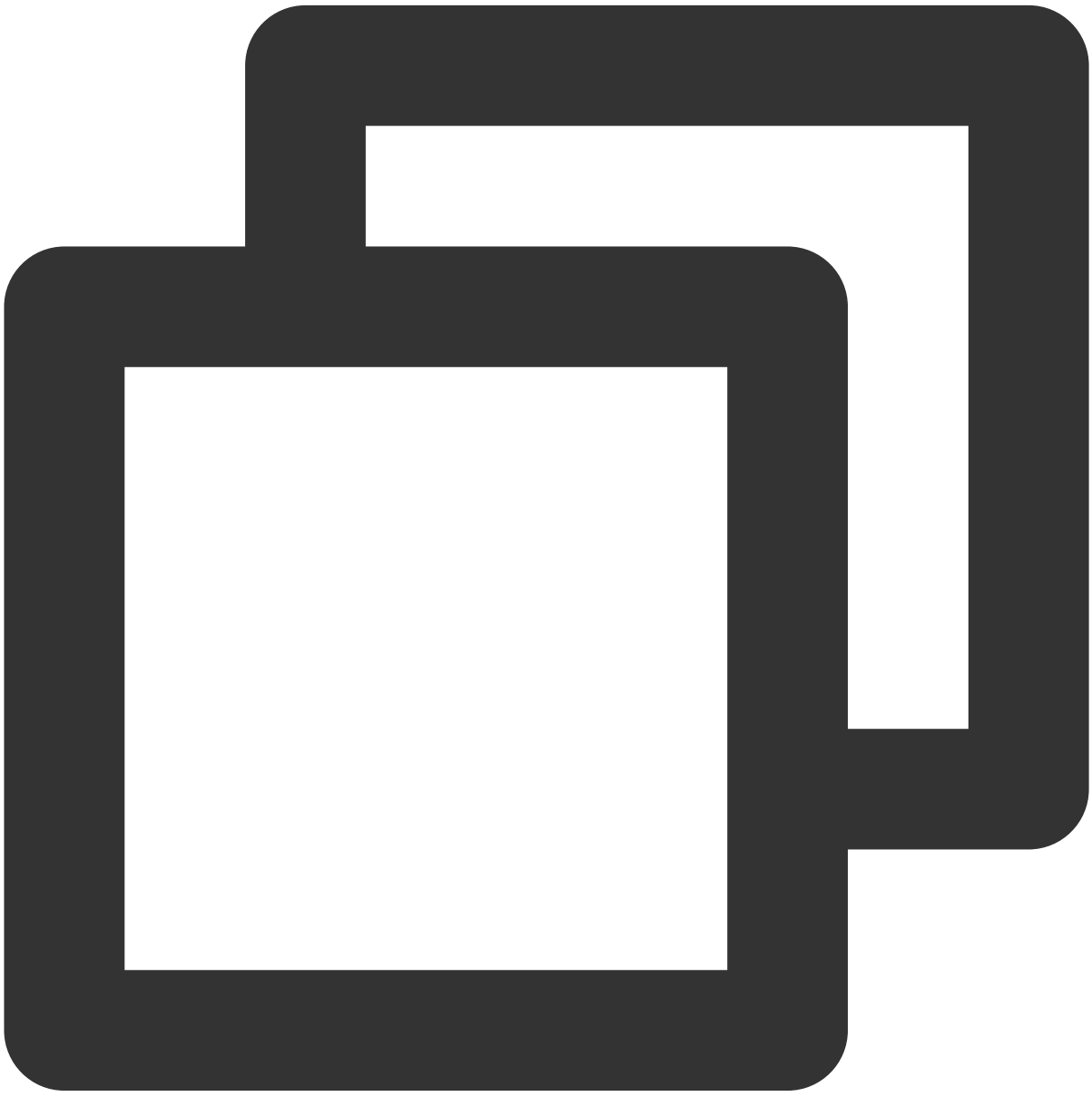


```
// Log in: userID can be defined by the user and userSig can be generated as per st
V2TIMManager.getInstance().login(userID, userSig, new V2TIMCallback() {
    @Override
    public void onSuccess() {
        Log.i("imsdk", "success");
    }

    @Override
    public void onError(int code, String desc) {
        // The following error codes indicate an expired userSig, and you need to g
        // 1. ERR_USER_SIG_EXPIRED(6206).
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // 2. ERR_SVR_ACCOUNT_USERSIG_EXPIRED(70001).
        // Note: Do not call the log-in API in case of other error codes. Otherwise
        Log.i("imsdk", "failure, code:" + code + ", desc:" + desc);
    }
});
```

Log out operation



```
// Log out
V2TIMManager.getInstance().logout(new V2TIMCallback() {
    @Override
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    public void onSuccess() {
        Log.i("imsdk", "success");
    }

    @Override
    public void onError(int code, String desc) {
        Log.i("imsdk", "failure, code:" + code + ", desc:" + desc);
    }
});
```

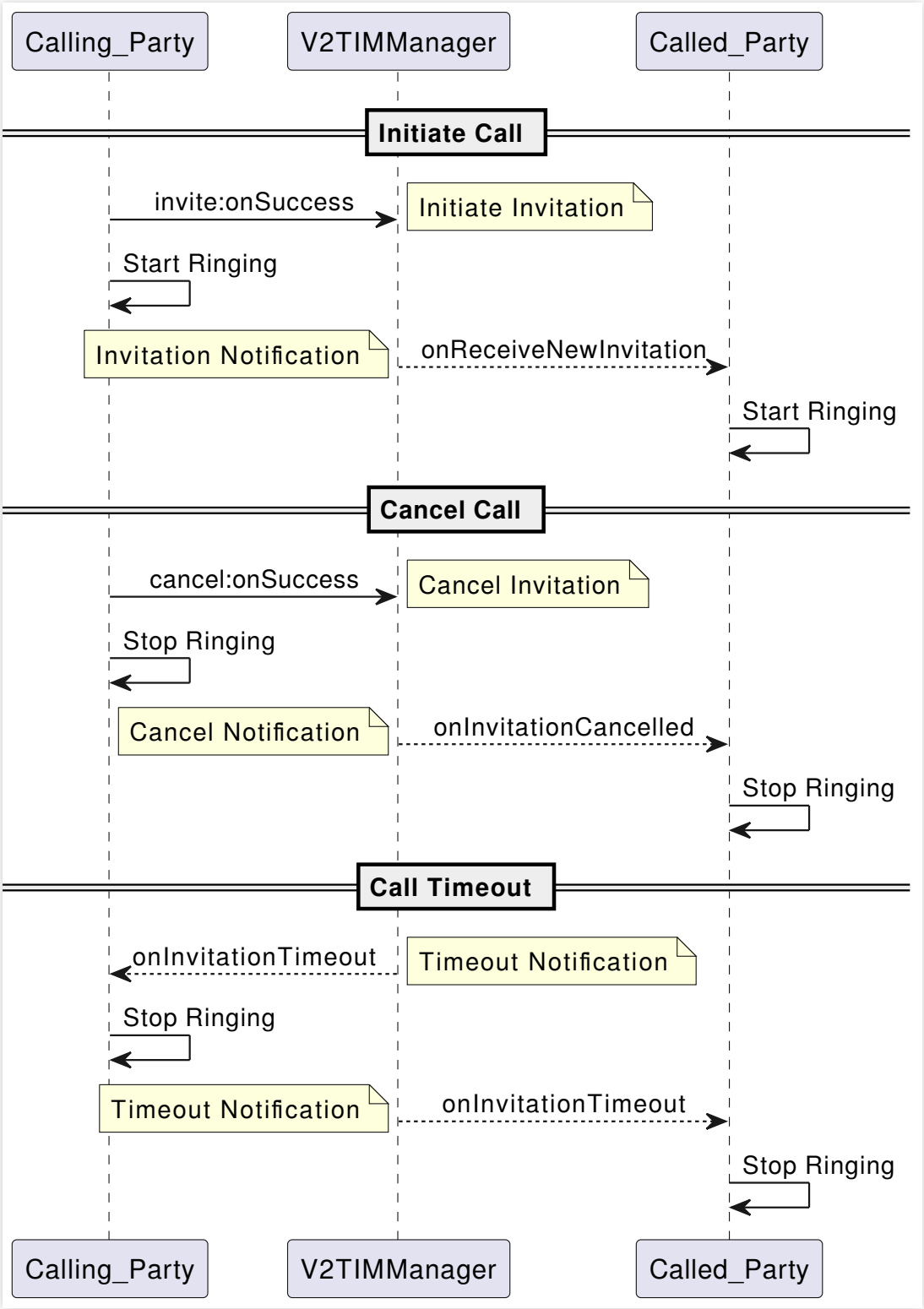
Note:

If the lifecycle of your application matches that of the IM SDK, you do not need to log out before exiting the application. However, if you only use the IM SDK after entering specific interfaces and no longer use it after exiting those interfaces, you can log out and deinitialize the IM SDK.

Step 2: call

Sequence Diagram

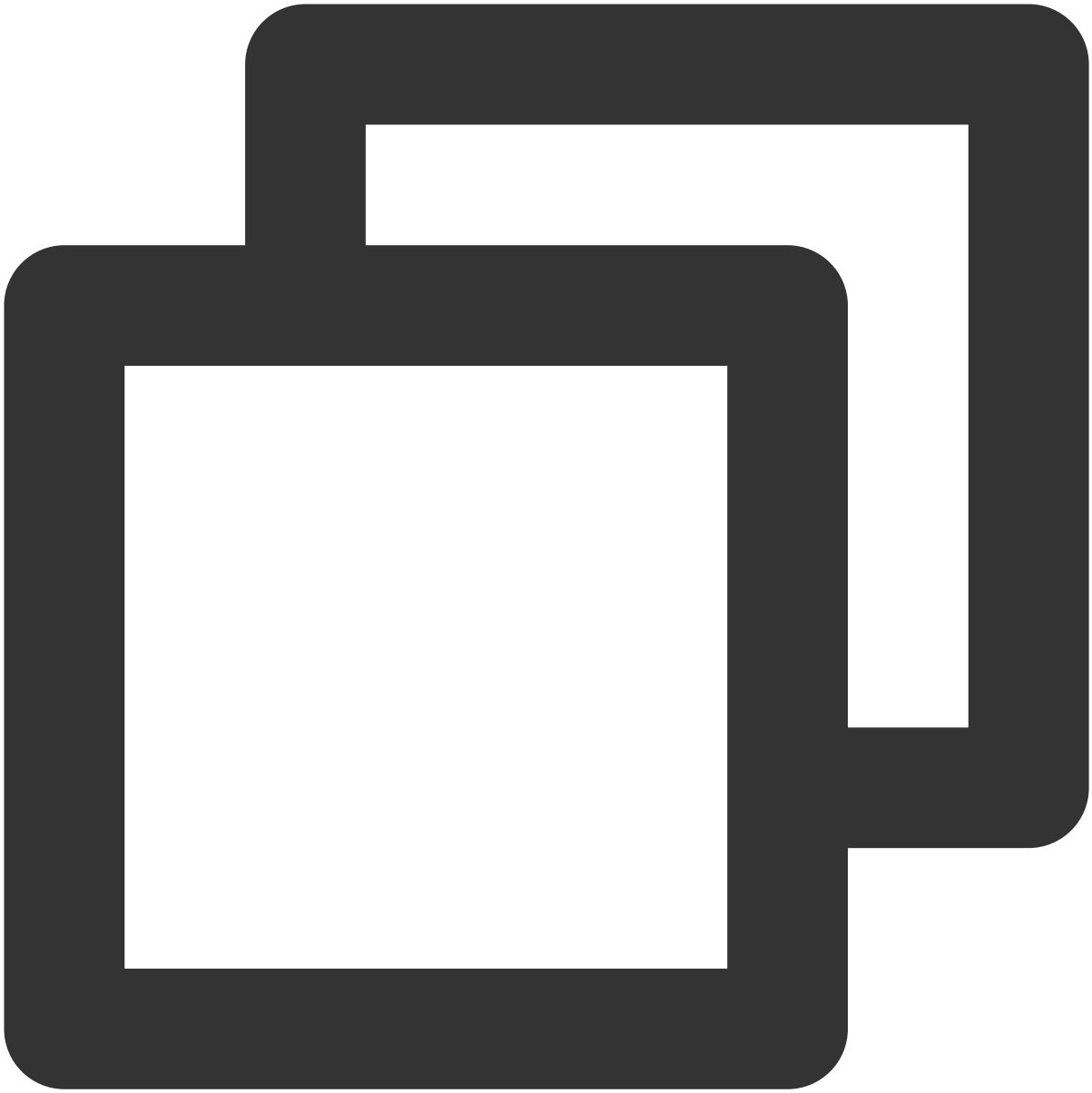
©2013-2022 Tencent Cloud. All rights reserved.



Initiate Call

1. Caller's local video preview (only for video calls; ignore this step for audio calls)

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set video encoding parameters to determine the picture quality seen by remote us
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
encParam.videoFps = 15;
encParam.videoBitrate = 850;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT;
mTRTCCloud.setVideoEncoderParam(encParam);

// Enable local camera preview (you can specify to use the front/rear camera for vi
mTRTCCloud.startLocalPreview(isFrontCamera, previewView);
```

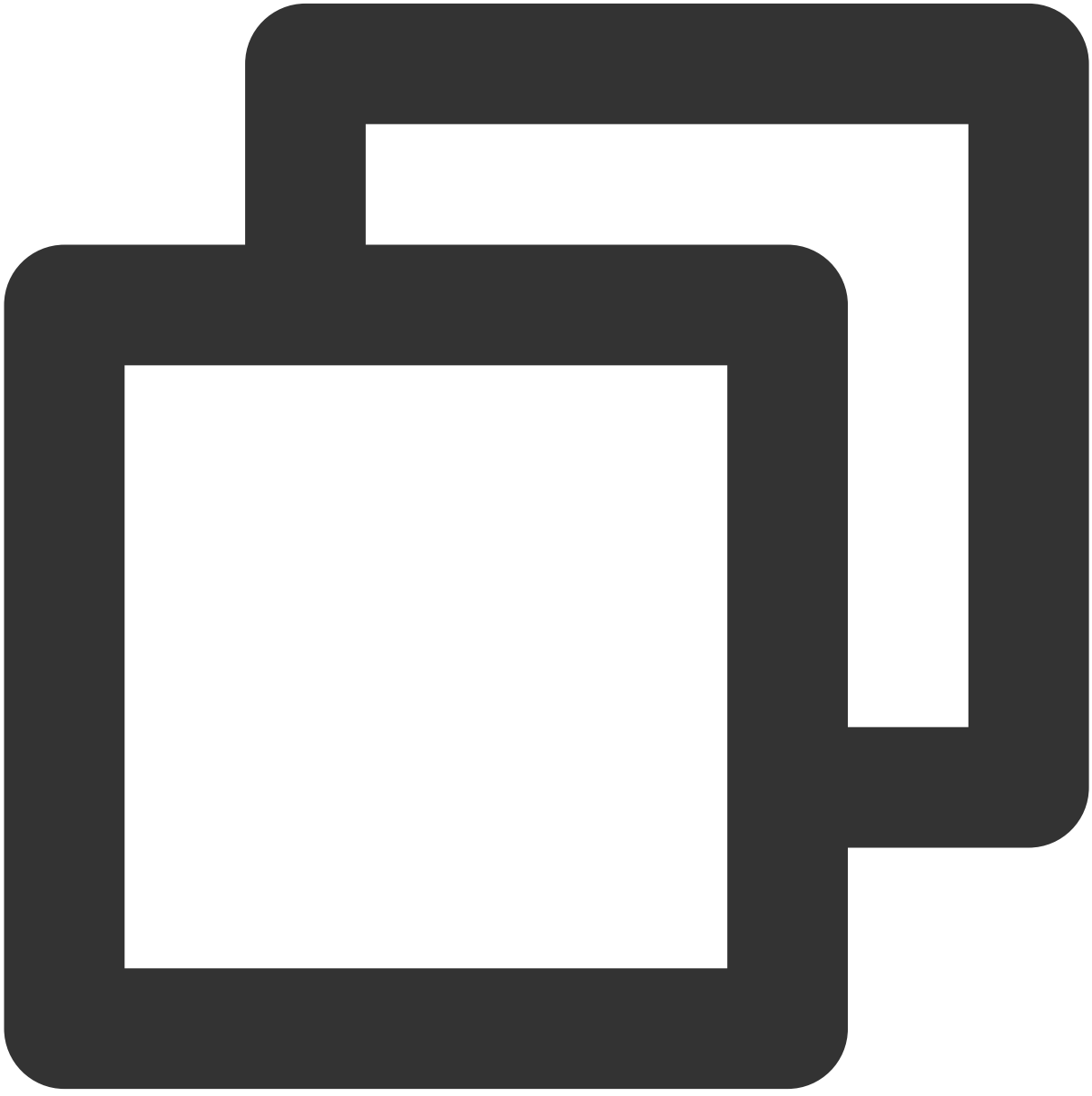
©2013-2022 Tencent Cloud. All rights reserved.

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above APIs before enterRoom. The SDK will only enable the camera preview and will wait until you call enterRoom to start local video streaming.

2. Caller sends call invitation signaling



```
// Construct custom data
JSONObject jsonObject = new JSONObject();
try {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    jsonObject.put("cmd", "av_call");
    JSONObject msgJsonObject = new JSONObject();
    msgJsonObject.put("callType", "videoCall"); // Specify the call type (video cal
    msgJsonObject.put("roomId", generateRoomId());  // Specify the TRTC room ID (ca
    jsonObject.put("msg", msgJsonObject);
} catch (JSONException e) {
    e.printStackTrace();
}
String data = jsonObject.toString();

// Send call invitation signaling
V2TIMManager.getSignalingManager().invite(receiver, data, false, v2TIMOfflinePushIn
    @Override
    public void onError(int code, String desc) {
        // Failed to send call invitation signaling
        // Prompt call failure, you can try to retry
    }

    @Override
    public void onSuccess() {
        // Successfully send call invitation signaling
        // Render call page, play call ringtone
    }
});
```

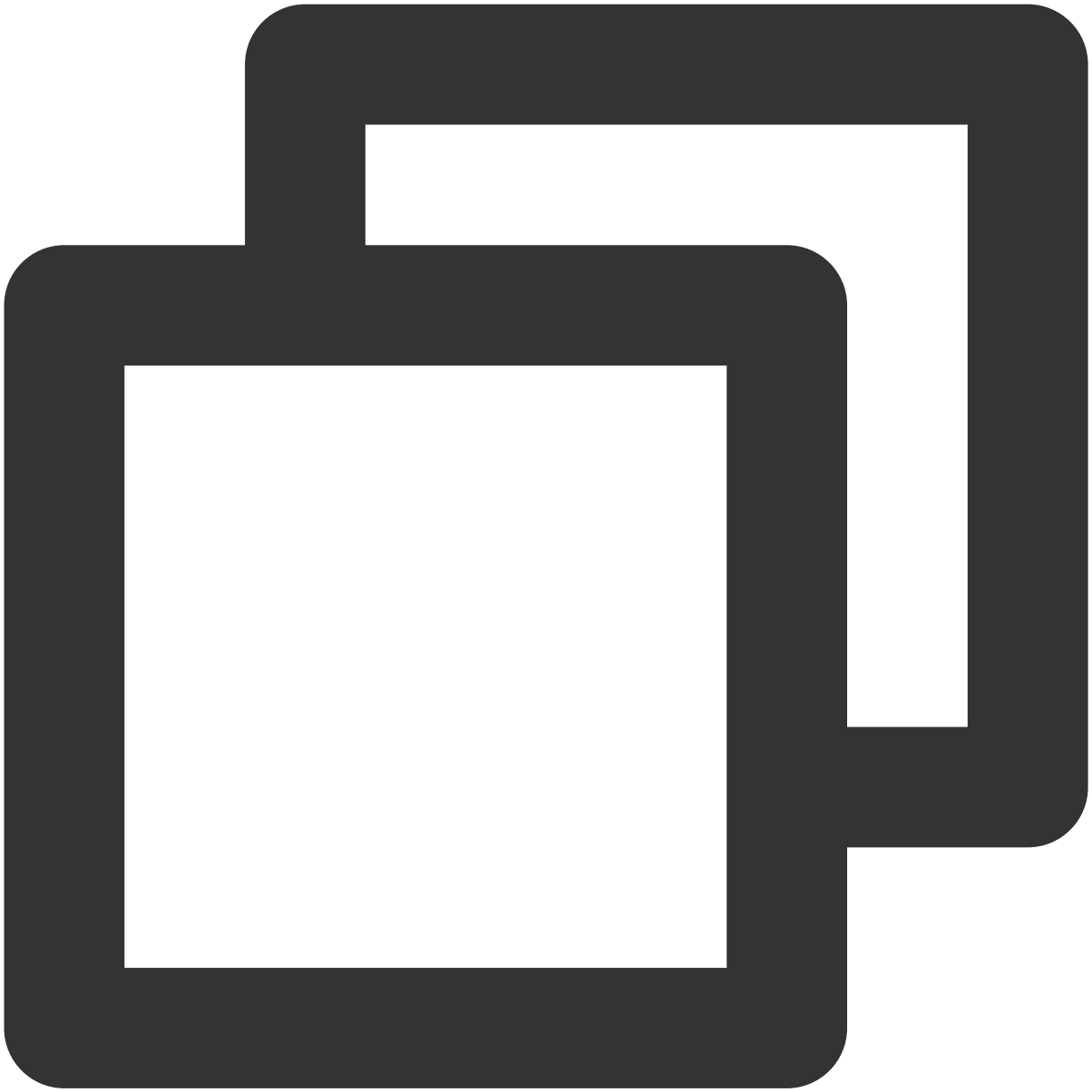
Note:

In audio and video call scenarios, it is usually necessary to configure offline push information `v2TIMOfflinePushInfo` in the invitation signaling. For details, see Offline Push Message.

It is recommended to set a reasonable timeout parameter `timeout` in the invitation signaling, in seconds. The SDK will perform timeout detection to realize auto hang up after call timeout.

3. Callee receives the call invitation notification

©2013-2022 Tencent Cloud. All rights reserved.



```
// Callee receives the call request. The inviteID is the request ID, and inviter is
V2TIMManager.getSignalingManager().addSignalingListener(new V2TIMSignalingListener(
    @Override
    public void onReceiveNewInvitation(String inviteID, String inviter,
                                       String groupId, List<String> inviteeList, St
        if (!data.isEmpty()) {
            try {
                JSONObject jsonObject = new JSONObject(data);
                String command = jsonObject.getString("cmd");
                JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
                if (command.equals("av_call")) {
```

©2013-2022 Tencent Cloud. All rights reserved.

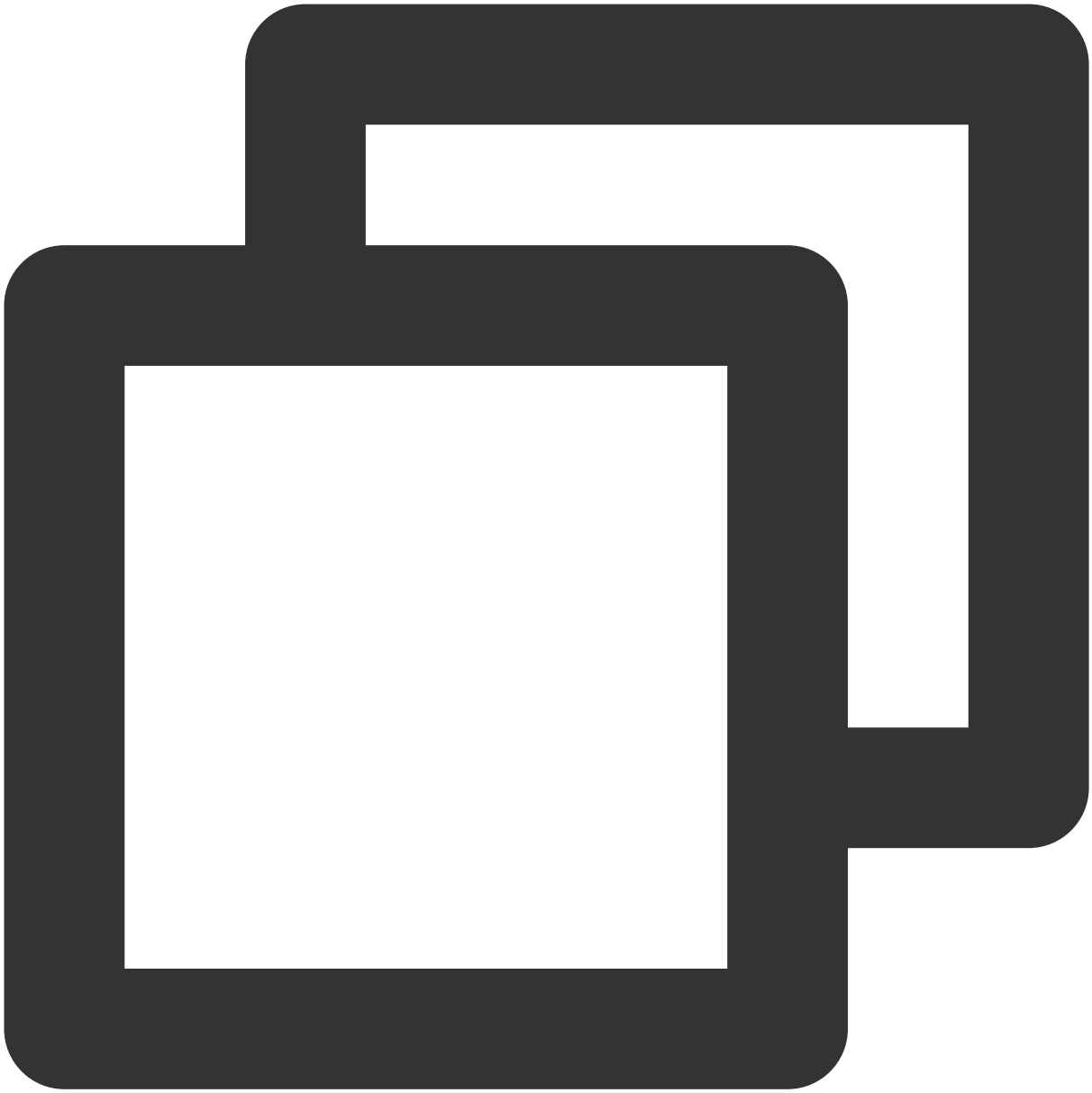
```
                    String callType = messageJsonObject.getString("callType");
                    String roomId = messageJsonObject.getString("roomId");

                    // Render call page, play call ringtone
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
});
```

Note:

Caller initiates a call request. When the callee receives the call request, the business side needs to implement the rendering of the call page and the playing of the call ringtone on its own.

4. Callee's local video preview (only for video calls; ignore this step for audio calls)

©2013-2022 Tencent Cloud. All rights reserved.



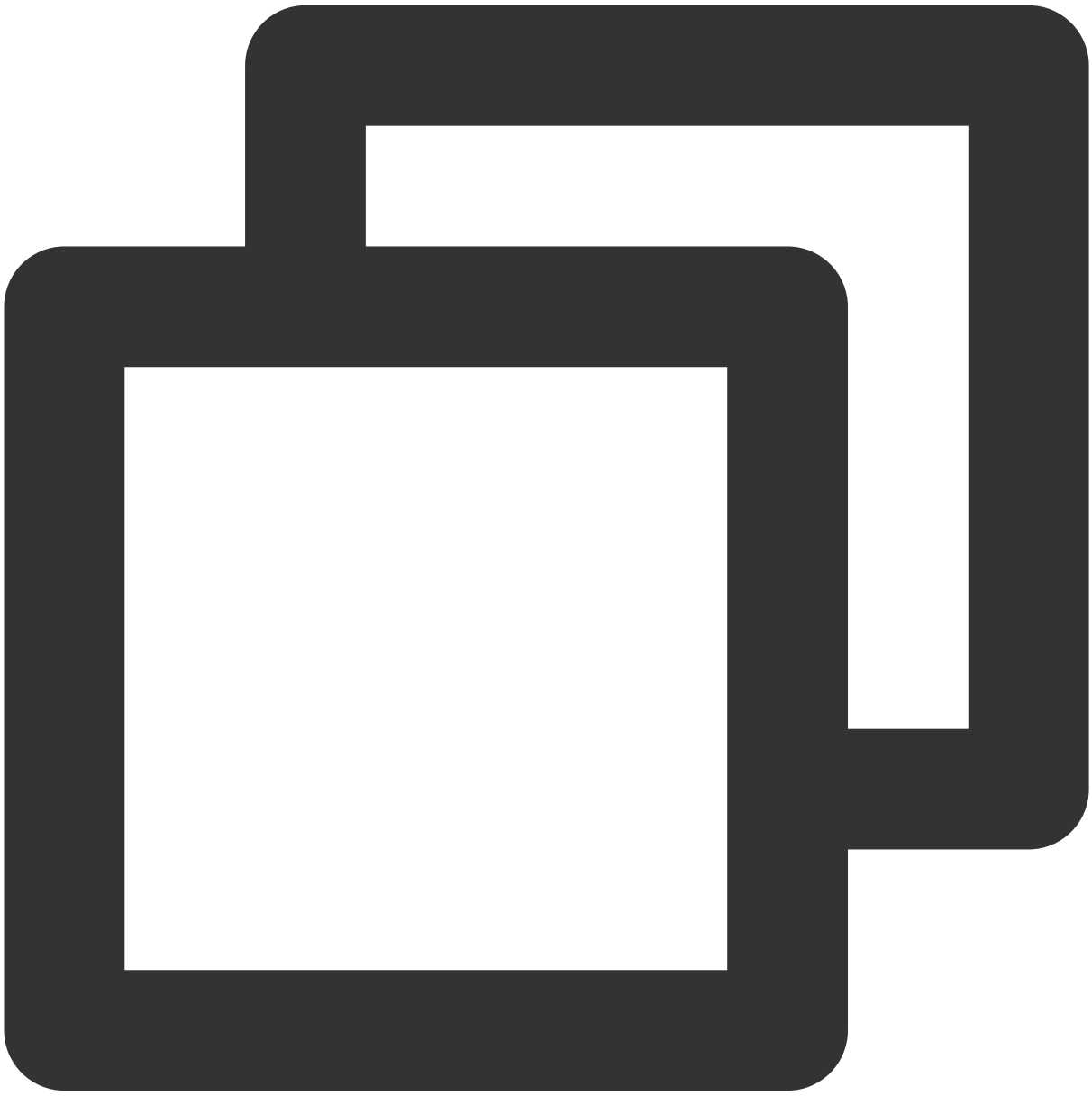
```
if (callType.equals("videoCall")) {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
    encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_960_540;
    encParam.videoFps = 15;
    encParam.videoBitrate = 850;
    encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT
    mTRTCCloud.setVideoEncoderParam(encParam);

    // Enable local camera preview (you can specify to use the front/rear camera fo
    mTRTCCloud.startLocalPreview(isFrontCamera, previewView);
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Cancel Call

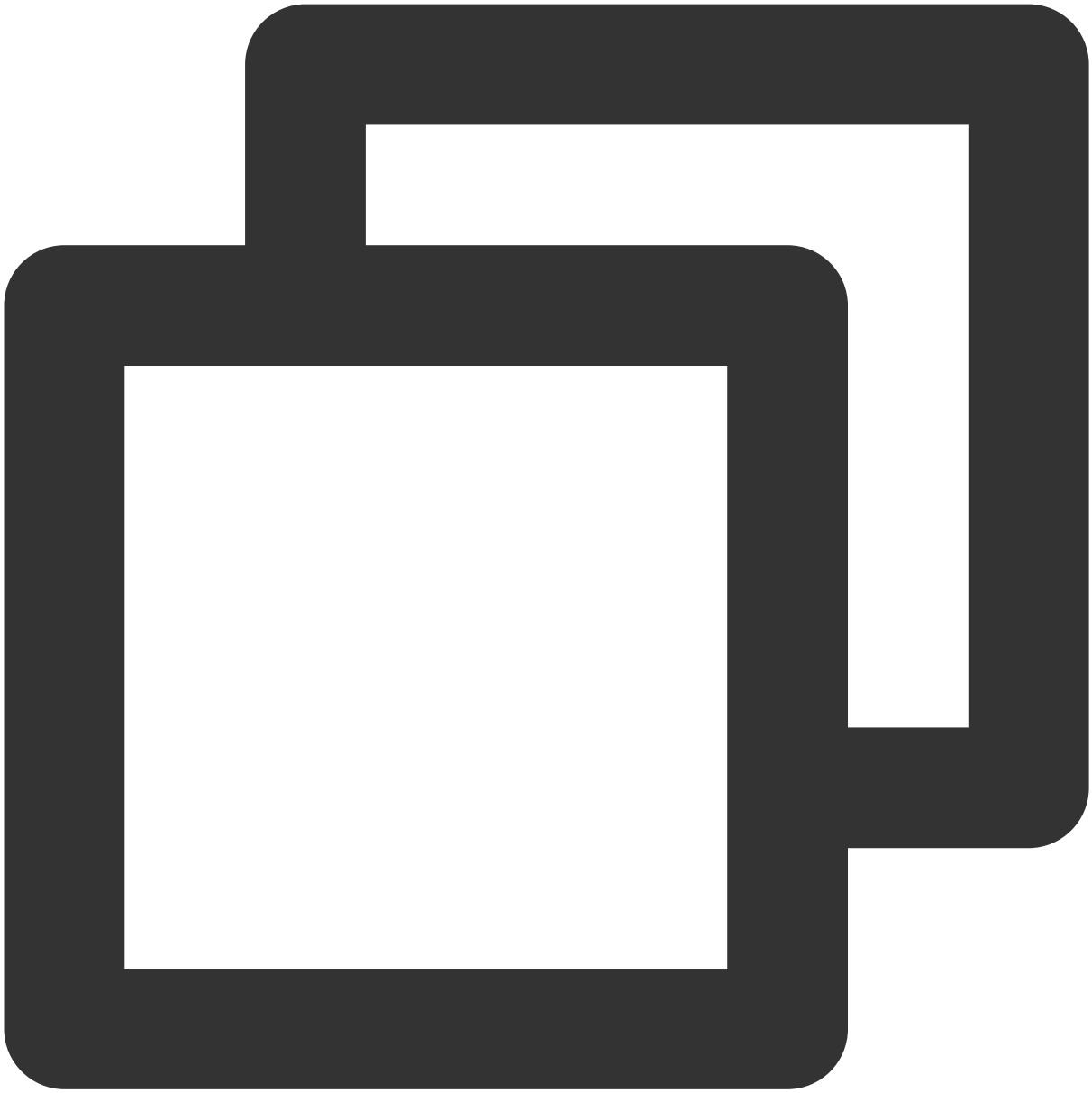
1. Caller cancels the call request



```
V2TIMManager.getSignalingManager().cancel(inviteId, data, new V2TIMCallback() {
        // Prompt cancel failed, you can try to retry    }    @Override    public v
        // Terminate the call page, and stop the call ringtone    }});
```

2. Callee receives the cancellation notification

©2013-2022 Tencent Cloud. All rights reserved.

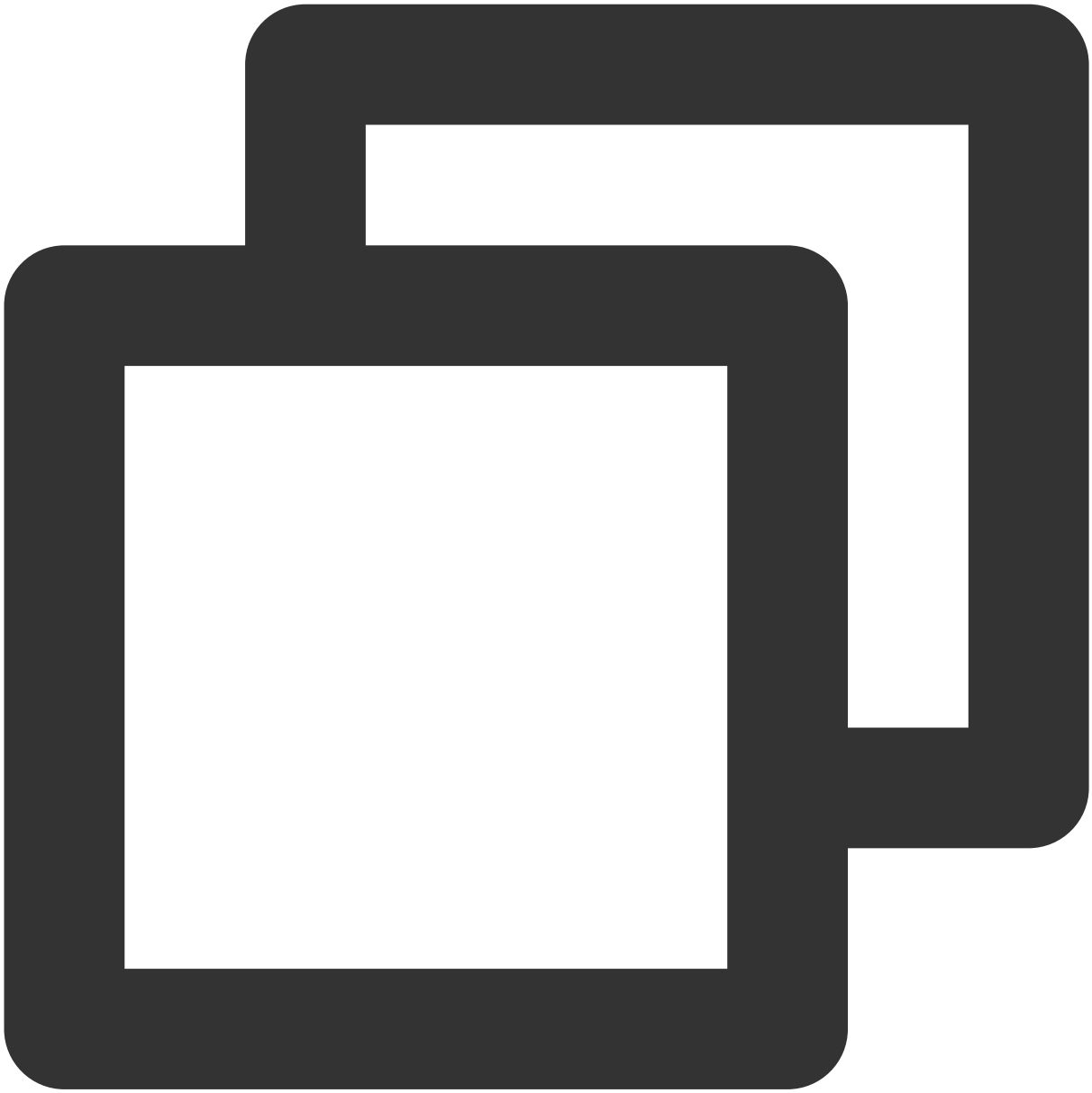


```
@Override
public void onInvitationCancelled(String inviteID, String inviter, String data) {
    // Terminate the call page, and stop the call ringtone}
```

Call Timeout

Both caller and callee will receive a timeout notification. They also terminate the call page, and stop the call ringtone.

©2013-2022 Tencent Cloud. All rights reserved.



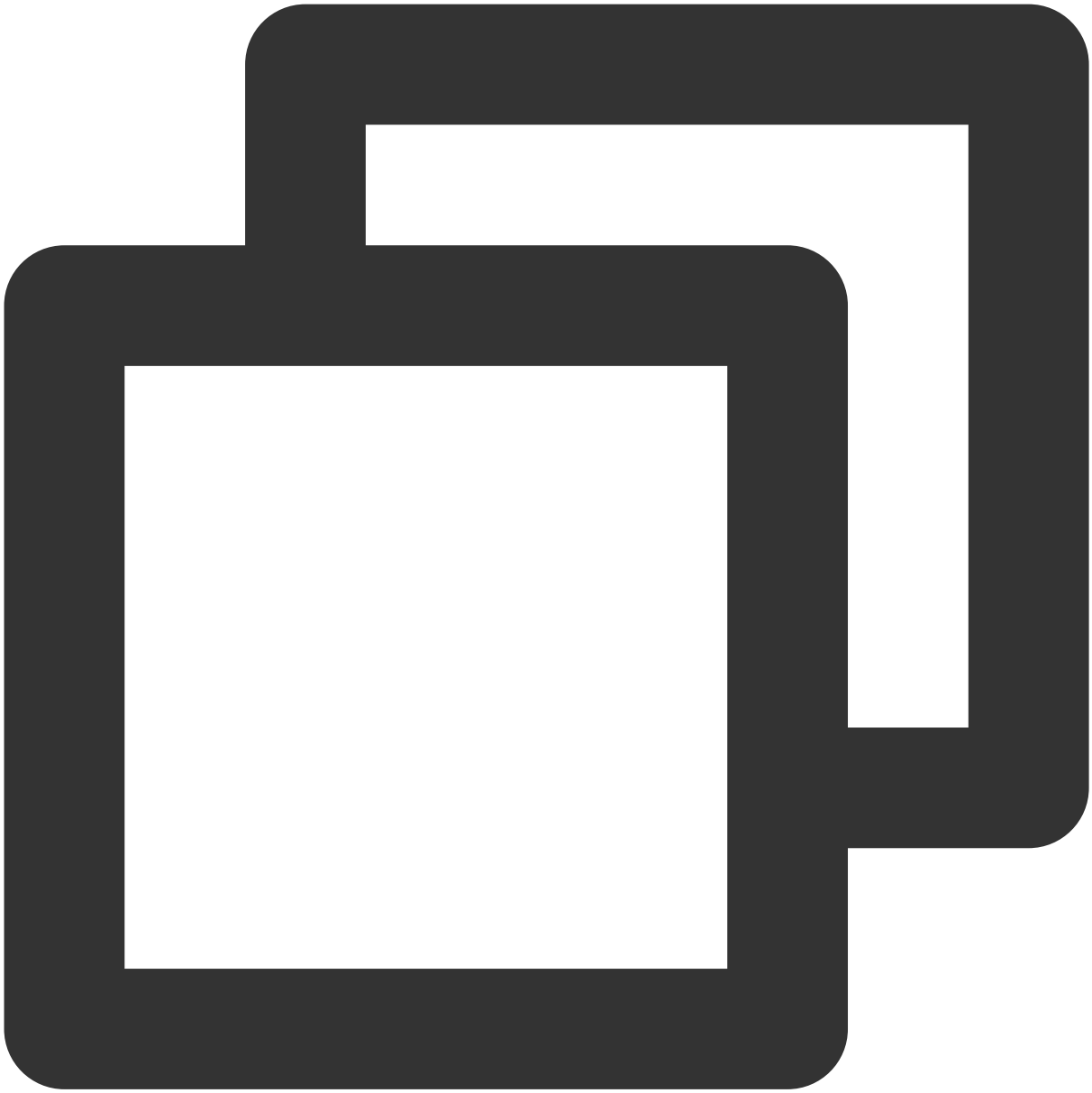
```
@Override
public void onInvitationTimeout(String inviteID, List<String> inviteeList) {
        // Prompt call timeout. Terminate the call page, and stop the call ringtone
}
```

Step 3: answer

Answer signaling

1. Callee sends answer signaling

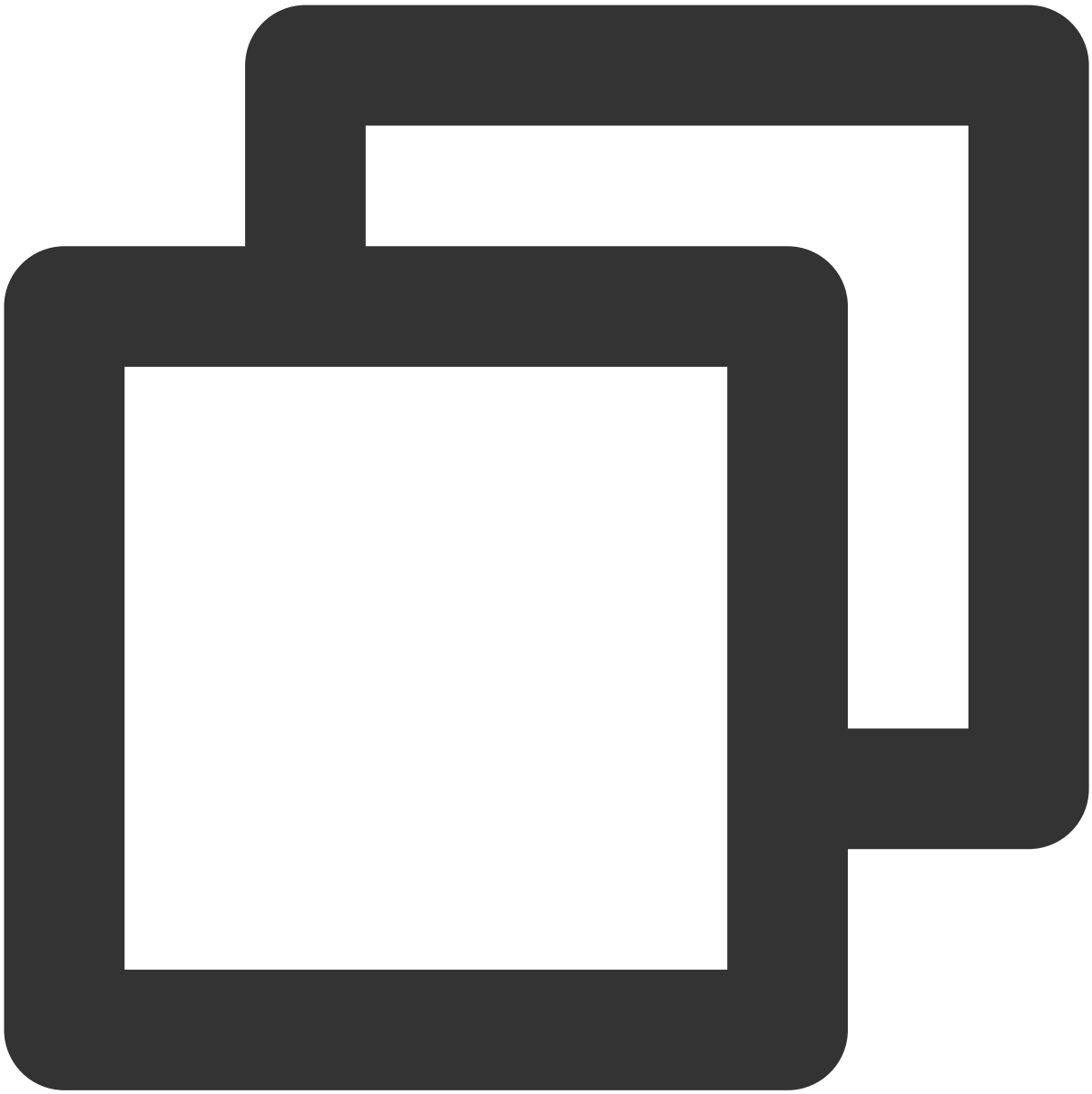
©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMManager.getSignalingManager().accept(inviteId, data, new V2TIMCallback() {
        if (callType.equals("videoCall")) {
            // Start video call
            startVideoCall();
        } else {
            // Start audio call
            startAudioCall();
        }
    }
});
```

©2013-2022 Tencent Cloud. All rights reserved.

2. Caller receives answer notification



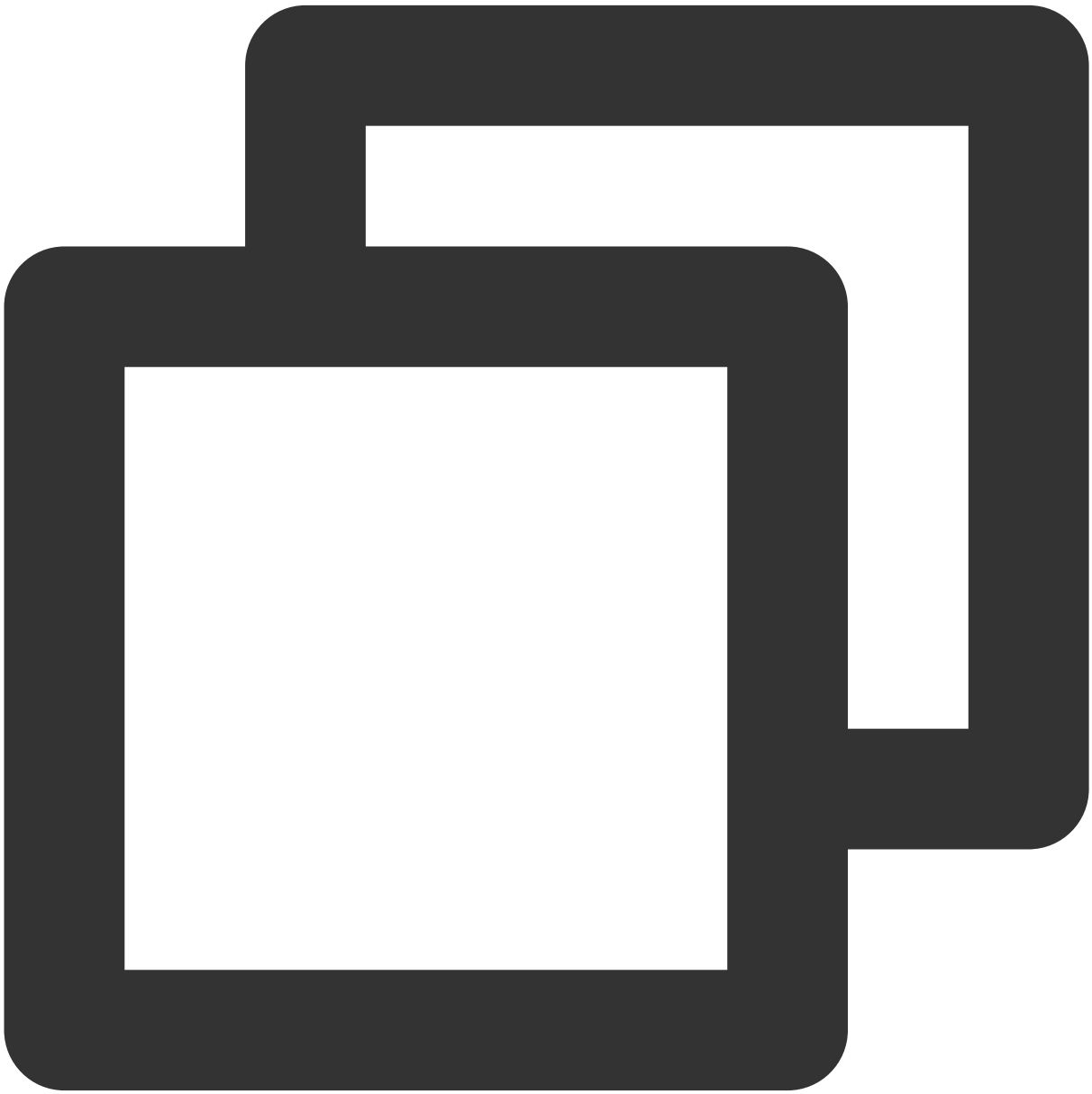
```
@Override
public void onInviteeAccepted(String inviteID, String invitee, String data) {
    if (callType.equals("videoCall")) {
        // Start video call
        startVideoCall();
    } else {
        // Start audio call
        startAudioCall();
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Audio Call

1. Both caller and callee enter **the same TRTC room** to start an audio call.



```
private void startAudioCall() {
    TRTCCloudDef.TRTCParams params = new
TRTCCloudDef.TRTCParams();
params.sdkAppId = SDKAPPID;// TRTC application ID, obtained from the console
params.userSig = USERSIG;// TRTC authentication credential, generated on the server
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.strRoomId = roomId; // Room ID, take the room ID string as an example
params.userId = userId;// Username, it is recommended to stay sync with IM

    mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH);
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_AUDIOCALL);
}
```

Note:

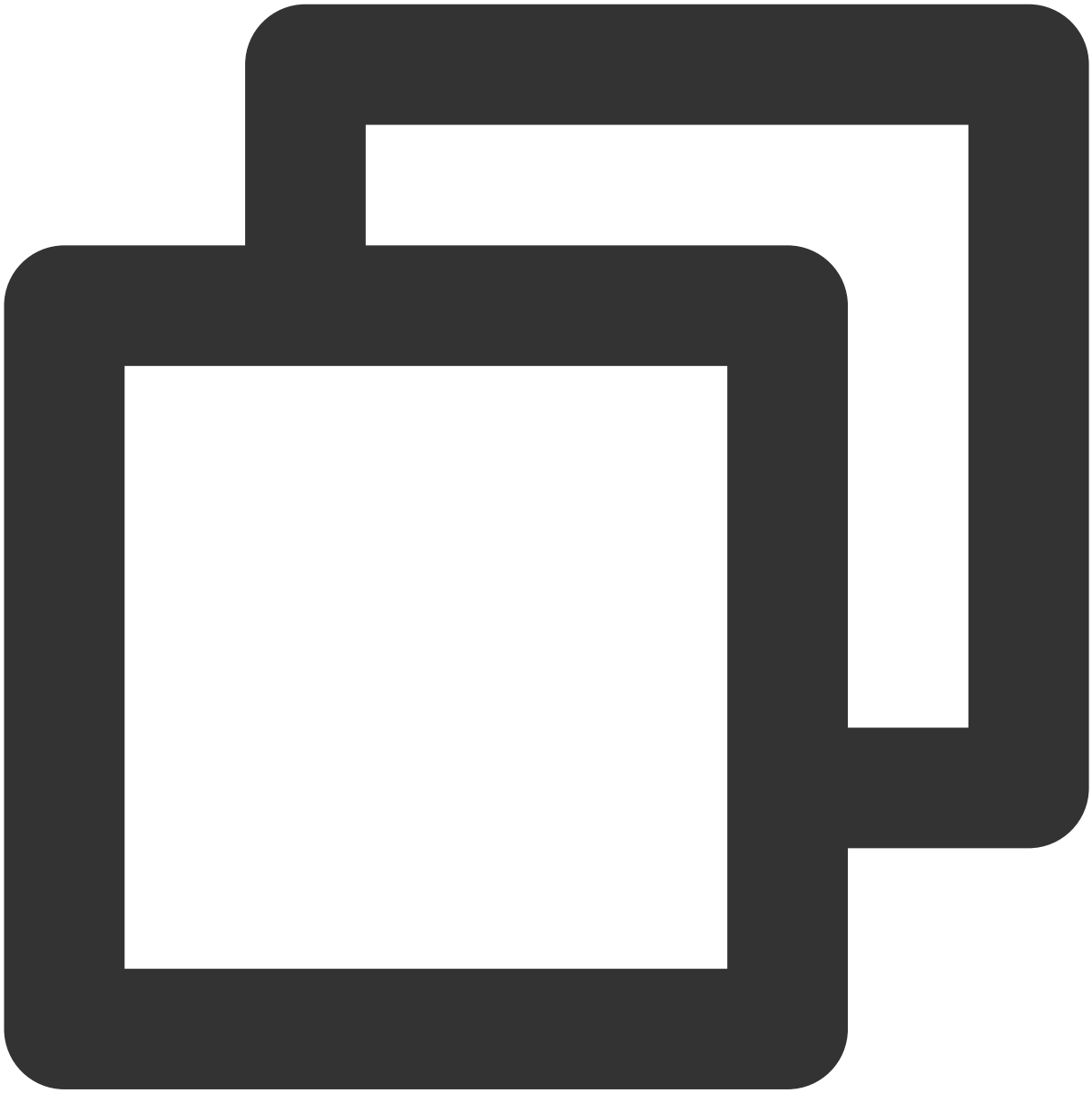
In audio call mode, the TRTC room entry scenario should use `TRTC_APP_SCENE_AUDIOCALL`, and the room entry role `TRTCRoleType` should not be specified.

Starting local audio capture `startLocalAudio` allows you to set audio quality parameters at the same time. For audio calls, it is recommended to set `TRTC_AUDIO_QUALITY_SPEECH`.

Under the SDK's default auto subscription mode, after a user enters a room, they will immediately receive the audio stream from that room, which will be automatically decoded and played without manual pulling.

2. Notification of room entry result, indicates call status.

©2013-2022 Tencent Cloud. All rights reserved.



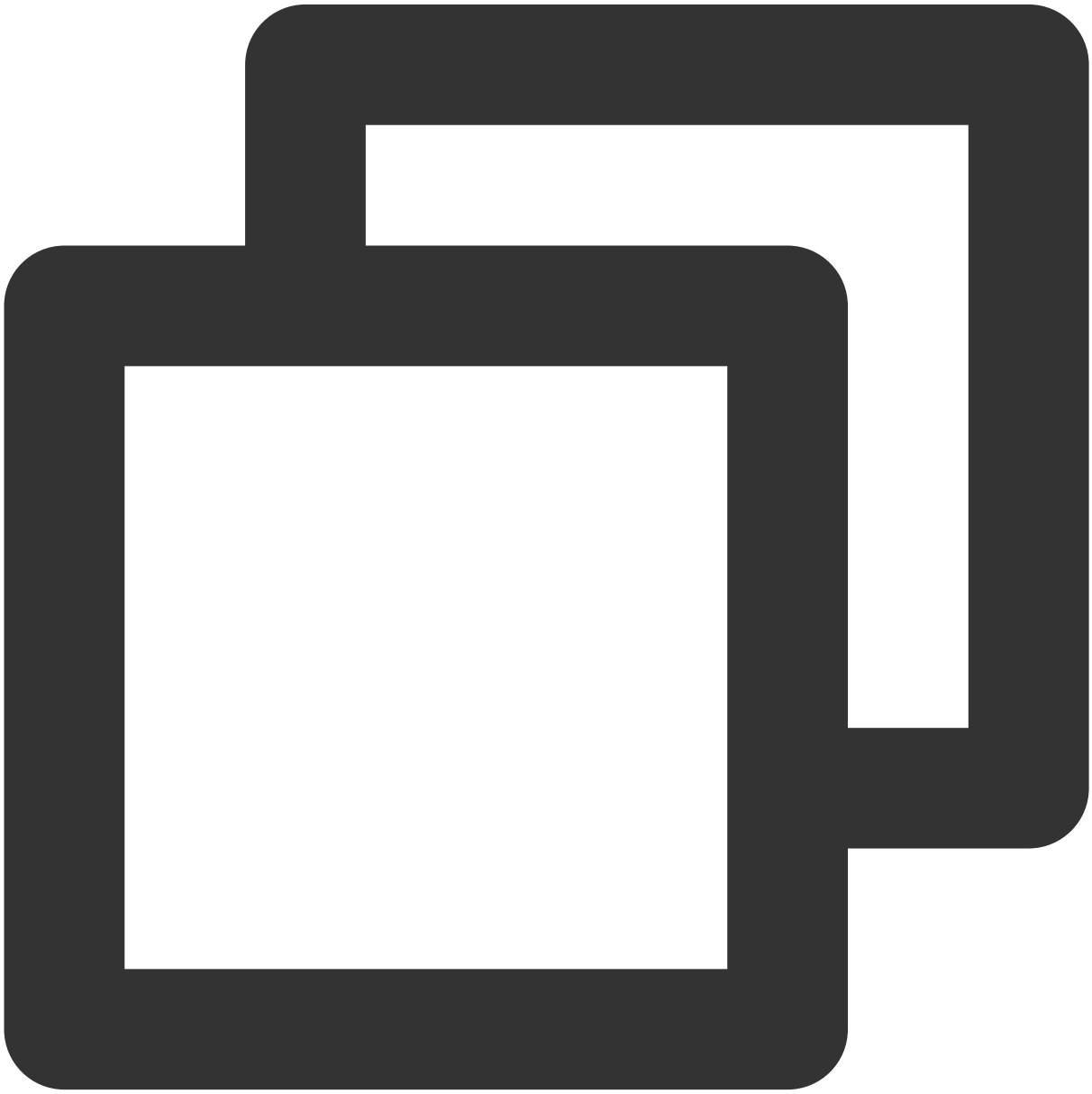
```
// Mark whether the call is in progress
boolean isOnCalling = false;

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // Room entry successful, indicates that the call is in progress
        isOnCalling = true;
    } else {
        // Failed to enter the room, prompt for call exception
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            isOnCalling = false;
        }
    }
```

Video Call

1. Both caller and callee enter **the same TRTC room** to start a video call.



```
private void startVideoCall() {
    TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
    params.sdkAppId = SDKAPPID;// TRTC application ID, obtained from the console
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    params.userSig = USERSIG;  // TRTC authentication credential, generated on the
    params.strRoomId = roomId; // Room ID, take the room ID string as an example
    params.userId = userId;// Username, it is recommended to stay sync with IM

    mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH);
    mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_VIDEOCALL);
}
```

Note:

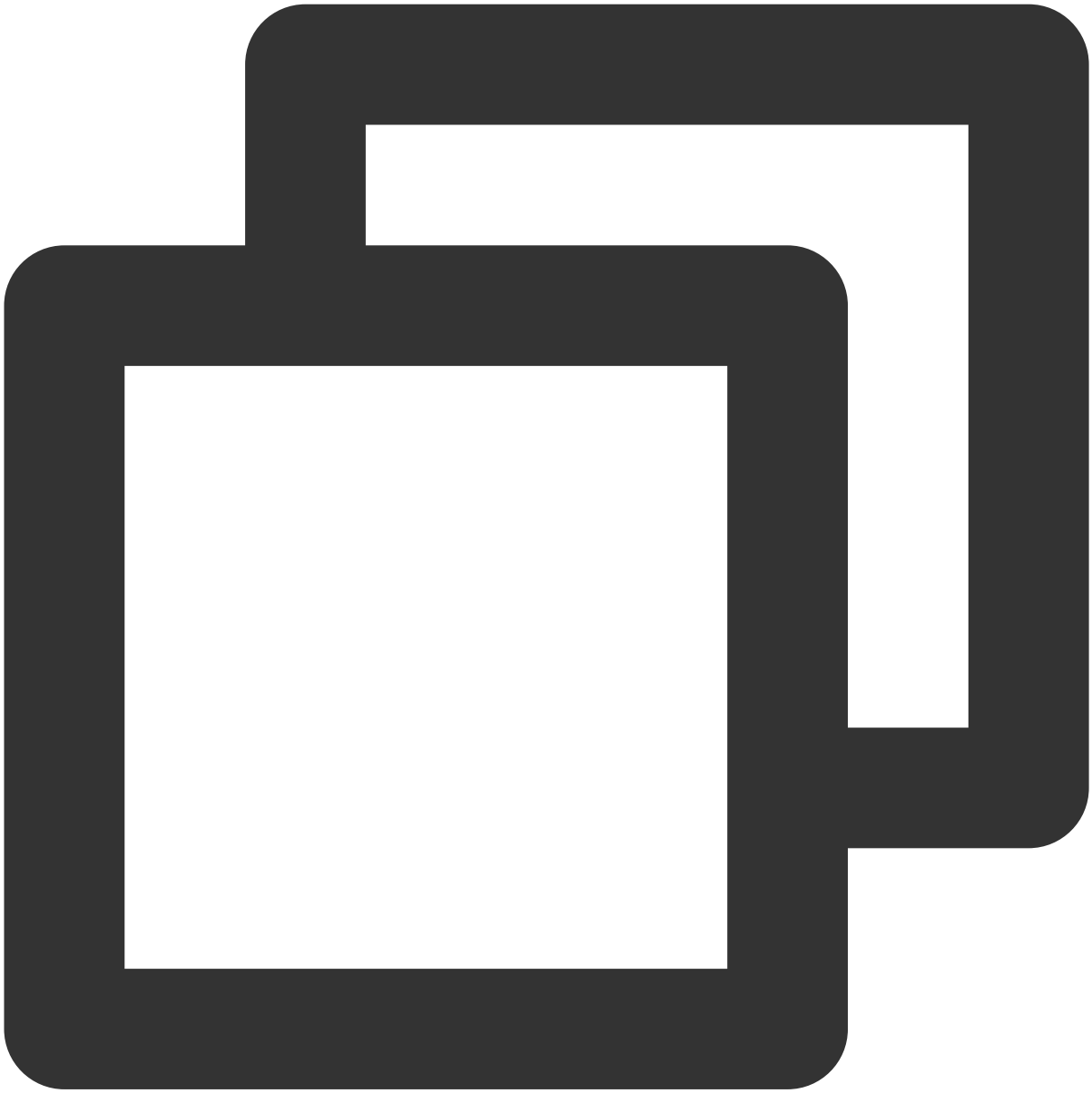
In video call mode, the TRTC room entry scenario should use `TRTC_APP_SCENE_VIDEOCALL`, and the room entry role `TRTCRoleType` should not be specified.

Starting local audio capture `startLocalAudio` allows you to set audio quality parameters at the same time. For video calls, it is recommended to set `TRTC_AUDIO_QUALITY_SPEECH`.

In the SDK's default automatic subscription mode, audio is automatically decoded and played back, while video requires manual invocation of `startRemoteView` to pull and render the remote video stream.

2. Notification of room entry result, indicates call status. Pull remote video stream.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Mark whether the call is in progress
boolean isOnCalling = false;

// Event callback for the result of entering the room
@Override
public void onEnterRoom(long result) {
    if (result > 0) {
        // Room entry successful, indicates that the call is in progress
        isOnCalling = true;
    } else {
        // Failed to enter the room, prompt for call exception
```

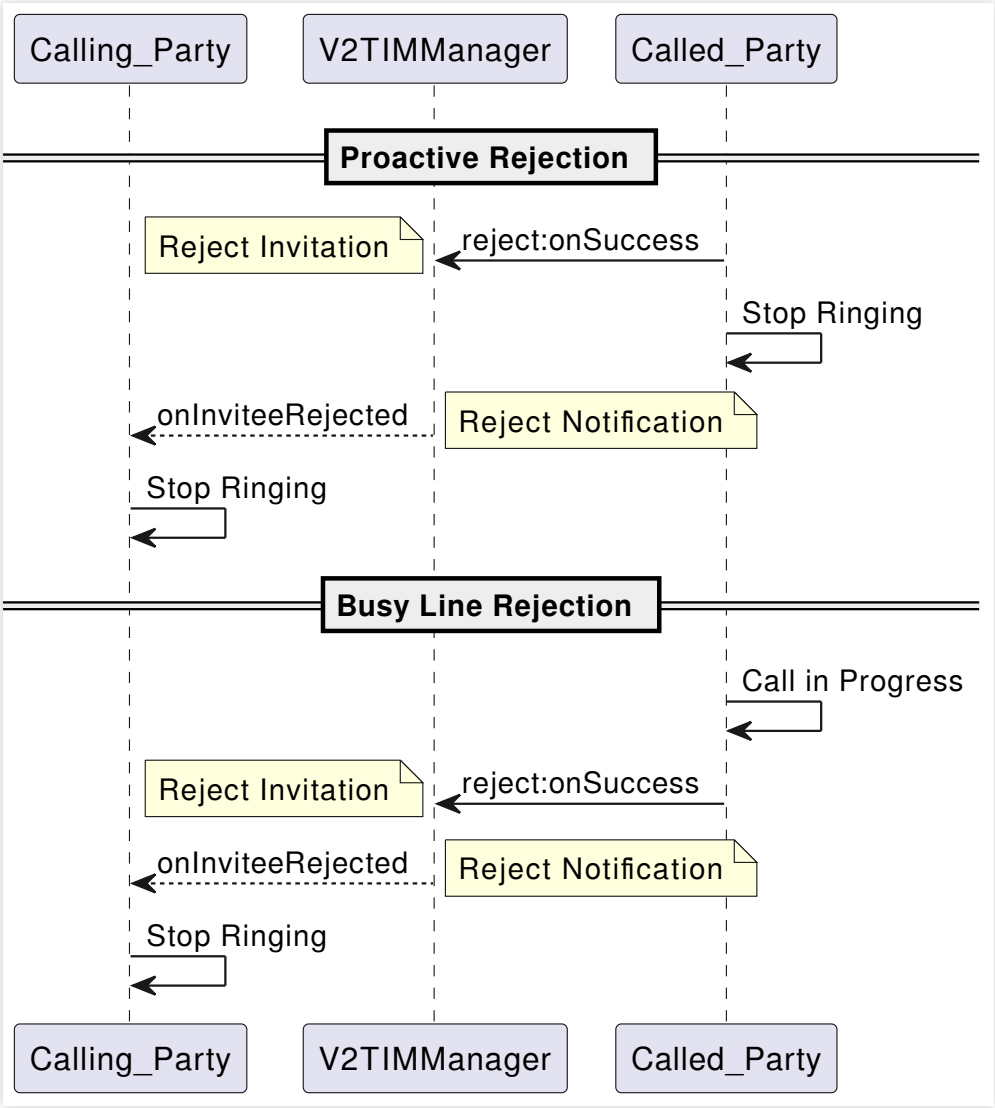
©2013-2022 Tencent Cloud. All rights reserved.

```
            isOnCalling = false;
        }
    }

    // Pull remote video stream
    @Override
    public void onUserVideoAvailable(String userId, boolean available) {
        // The remote user publishes/unpublishes the primary video
        if (available) {
            // Subscribe to the remote user's video stream and bind the video rendering
            mTRTCCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG,
        } else {
            // Unsubscribe to the remote user's video stream and release the rendering
            mTRTCCloud.stopRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG);
        }
    }
```

Step 4: reject call

Sequence Diagram

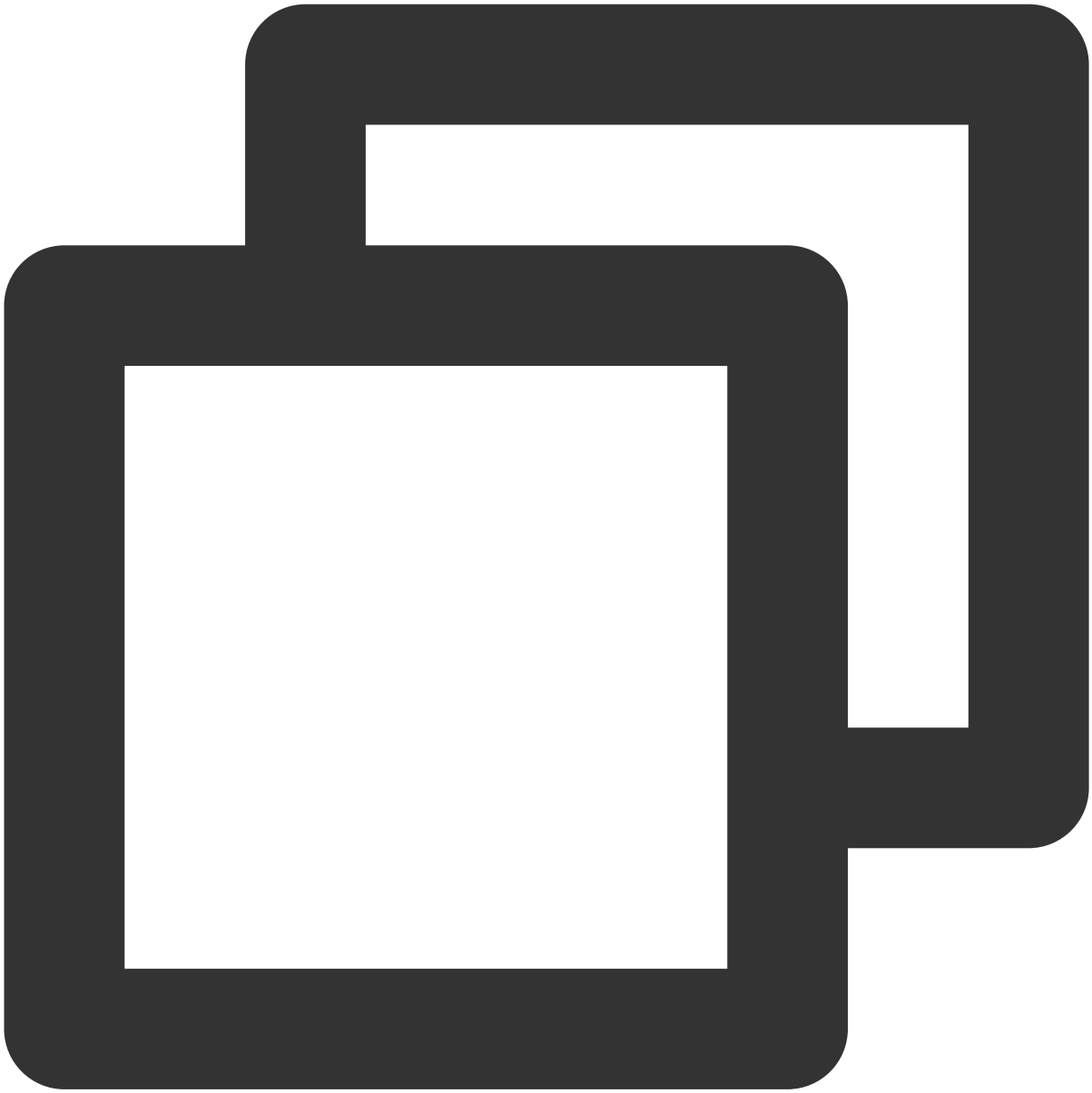
©2013-2022 Tencent Cloud. All rights reserved.



Proactive Rejection

1. Callee sends rejection signal

©2013-2022 Tencent Cloud. All rights reserved.



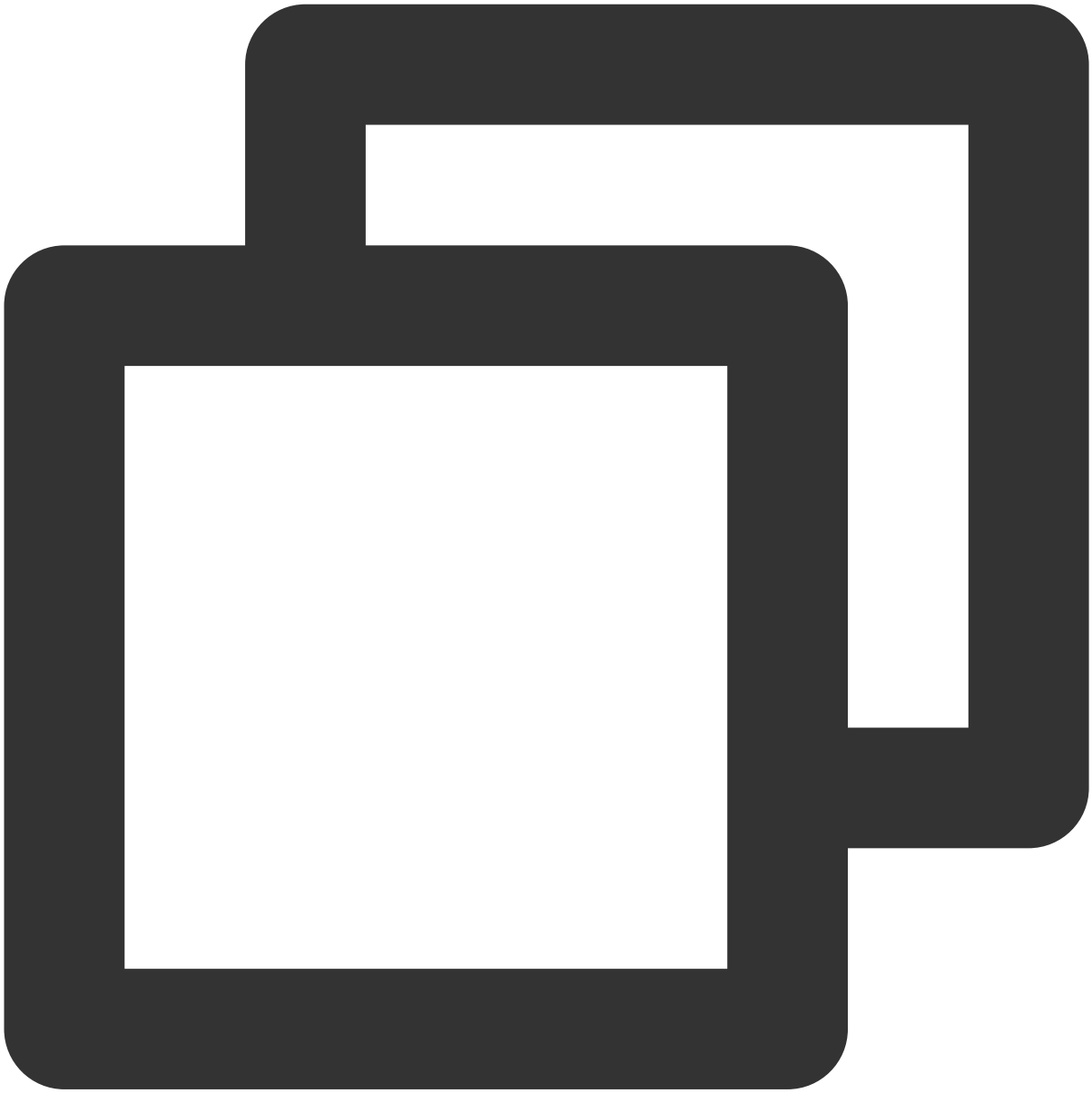
```
private void rejectInvite(String inviteId, String data, final V2TIMCallback callbac
    V2TIMManager.getSignalingManager().reject(inviteId, data, new V2TIMCallback() {
        @Override
        public void onError(int code, String desc) {
            if (callback != null) {
                callback.onError(code, desc);
            }
        }

        @Override
        public void onSuccess() {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            if (callback != null) {
                callback.onSuccess();
            }
        }
    });
}

JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("cmd", "av_call");
    JSONObject msgJsonObject = new JSONObject();
    msgJsonObject.put("callType", "videoCall"); // Specify the call type (video cal
    msgJsonObject.put("reason", "active"); // Specify rejection type (Proactive Rej
    jsonObject.put("msg", msgJsonObject);
} catch (JSONException e) {
    e.printStackTrace();
}

rejectInvite(inviteId, jsonObject.toString(), new V2TIMCallback() {    @Override
```

2. Caller receives rejection notification

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onInviteeRejected(String inviteID, String invitee, String data) {
    if (!data.isEmpty()) {
        try {
            JSONObject jsonObject = new JSONObject(data);
            String command = jsonObject.getString("cmd");
            JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
            if (command.equals("av_call")) {
                String reason = messageJsonObject.getString("reason");
                if (reason.equals("active")) {
                    // Prompt that the other party rejects call
```

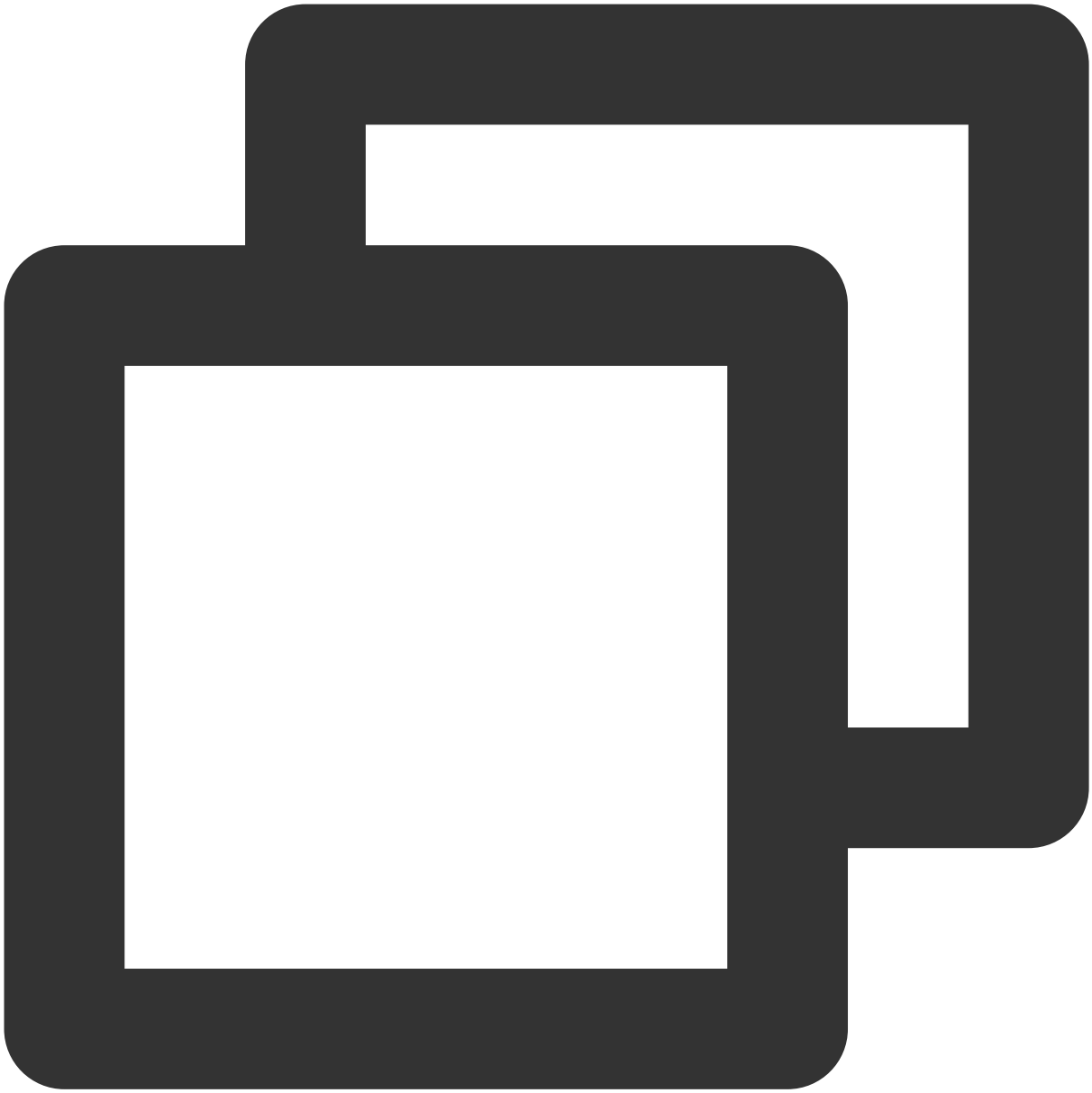
©2013-2022 Tencent Cloud. All rights reserved.

```
                } else if (reason.equals("busy")) {
                    // Prompt that the other party is busy
                }

                // Terminate the call page, and stop the call ringtone
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Busy Line Rejection

Callee receives a new call invitation, if the local call status is in a call, the caller automatically rejects the call.

©2013-2022 Tencent Cloud. All rights reserved.



```
@Override
public void onReceiveNewInvitation(String inviteID, String inviter,
                                   String groupId, List<String> inviteeList, String
    if (!data.isEmpty()) {
        try {
            JSONObject jsonObject = new JSONObject(data);
            String command = jsonObject.getString("cmd");
            JSONObject messageJsonObject = jsonObject.getJSONObject("msg");
            if (command.equals("av_call") && isOnCalling) {
                JSONObject jsonObject = new JSONObject();
                try {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                    jsonObject.put("cmd", "av_call");
                    JSONObject msgJsonObject = new JSONObject();
                    msgJsonObject.put("callType", "videoCall"); // Specify the call
                    msgJsonObject.put("reason", "busy");   // Specify rejection typ
                    jsonObject.put("msg", msgJsonObject);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
                // Local call is in progress, and sends busy line rejection signal
                rejectInvite(inviteId, jsonObject.toString(), new V2TIMCallback() {
                    @Override
                    public void onError(int code, String desc) {
                        // Busy line rejection failed
                    }

                    @Override
                    public void onSuccess() {
                        // Busy line rejection successful
                    }
                });
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

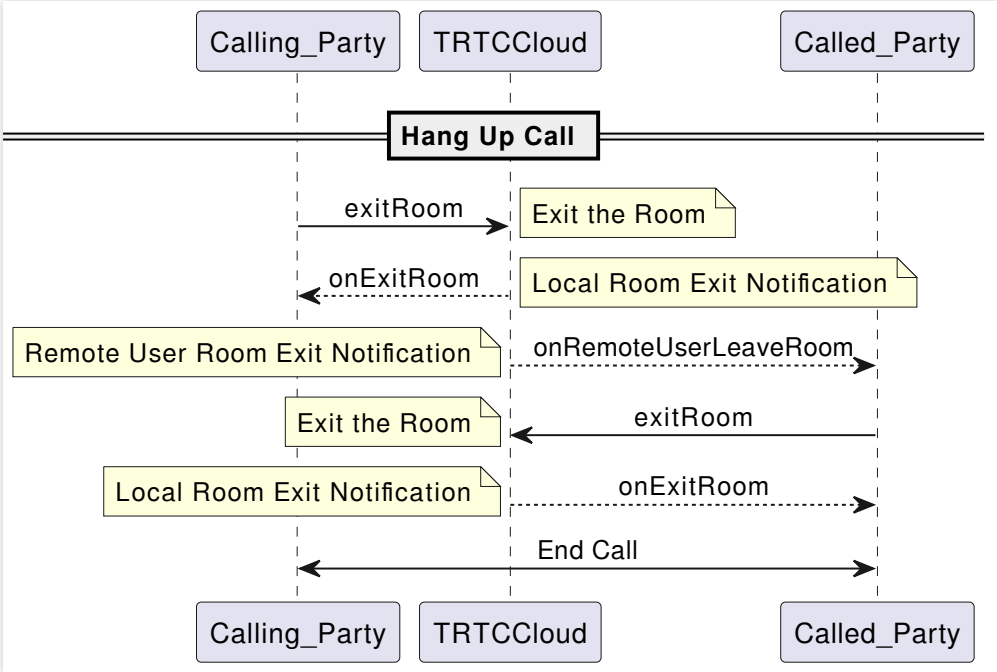
Note:

Both proactive rejection and busy line rejection use the `reject` signaling for implementation, but it's important to distinguish them through the `reason` field of the custom `data` in the signaling.

Step 5: hang up

Sequence Diagram

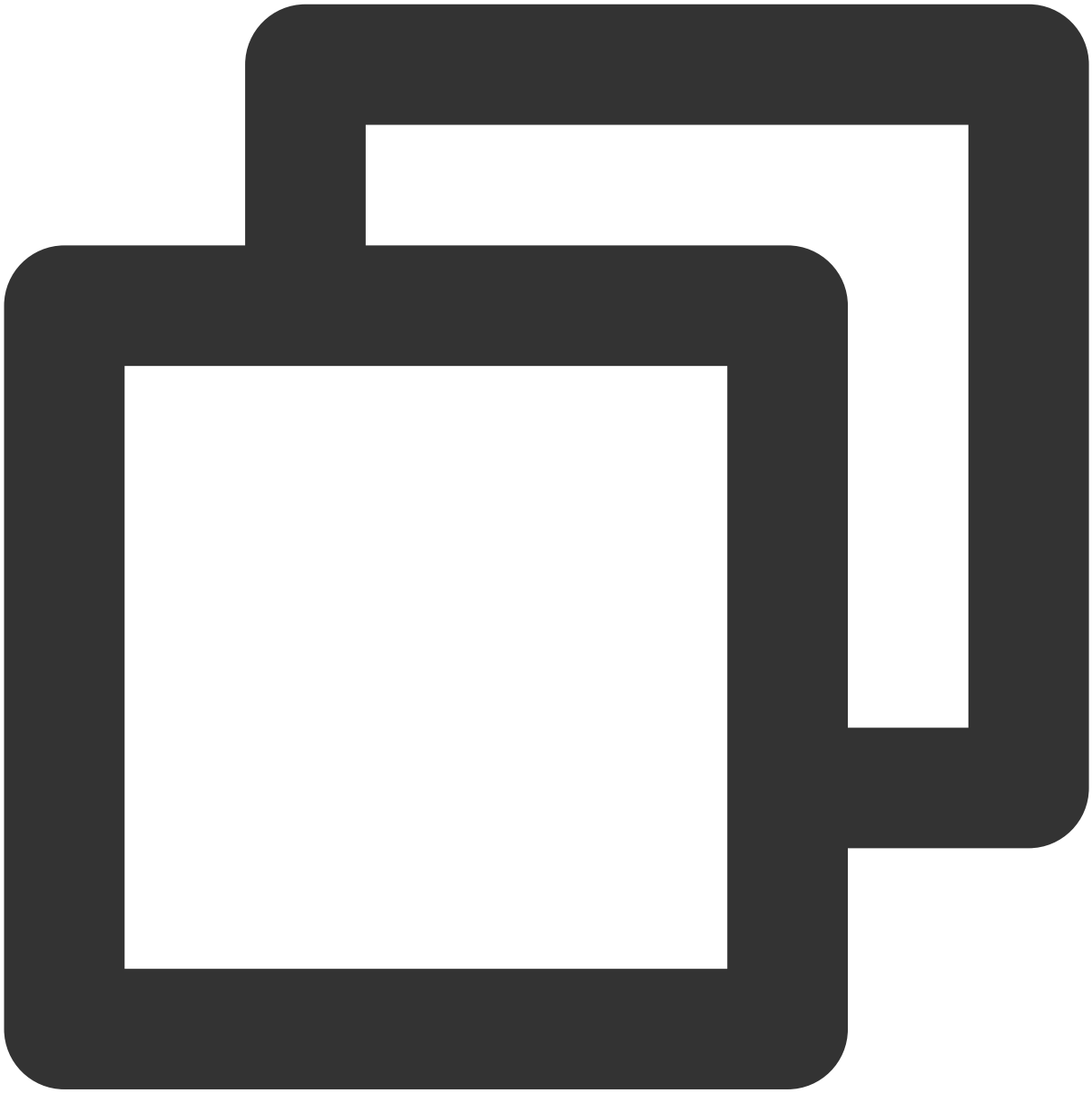
©2013-2022 Tencent Cloud. All rights reserved.



Hang Up Call

1. Either party exits the room, and reset the local call status.

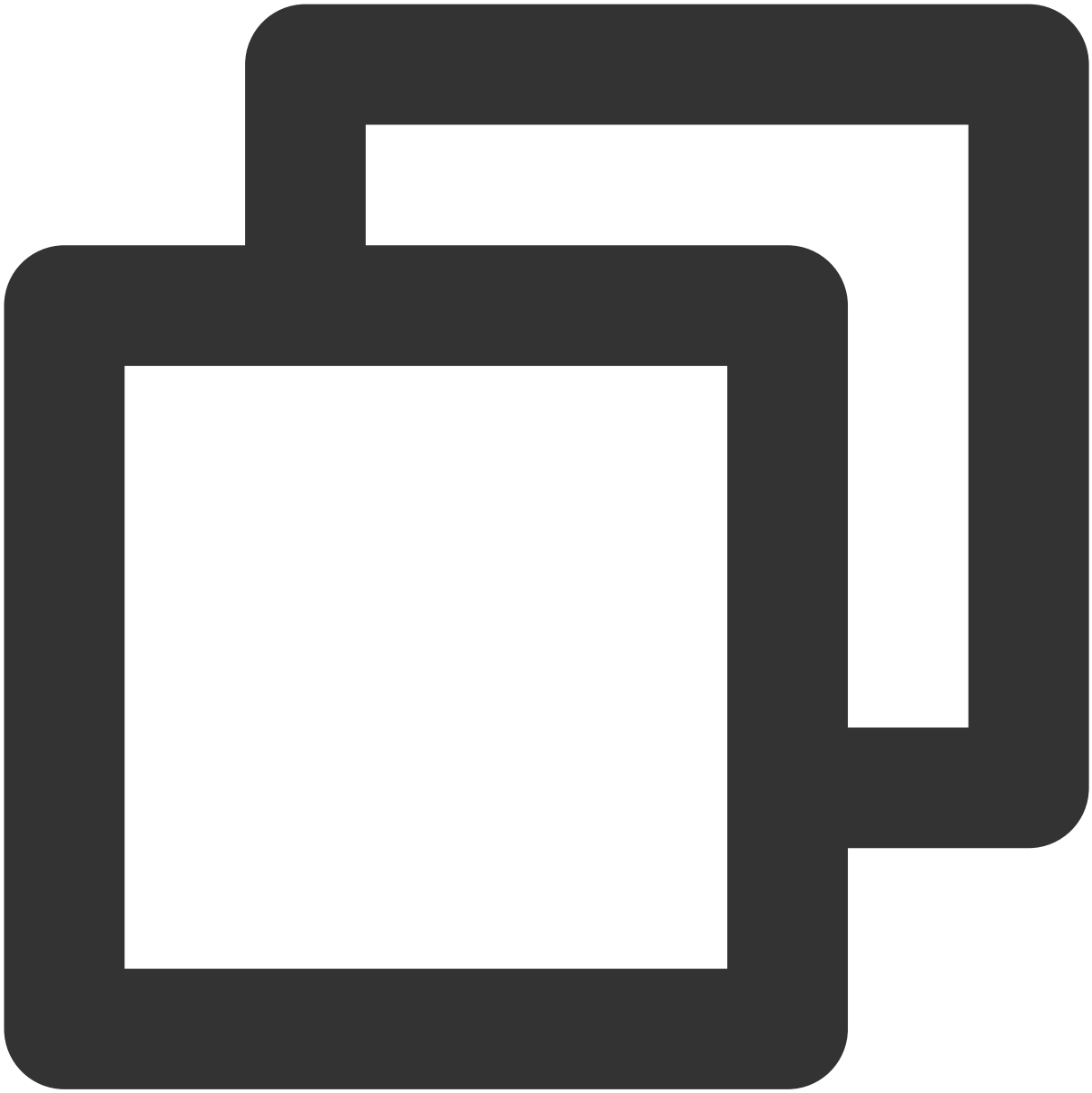
©2013-2022 Tencent Cloud. All rights reserved.



```
private void hangup() {
    mTRTCCloud.stopLocalAudio();
    mTRTCCloud.stopLocalPreview();
    mTRTCCloud.exitRoom();
}

@Override
public void onExitRoom(int reason) {
    // Successfully exited the room and hung up the call
    isOnCalling = false;
}
```

©2013-2022 Tencent Cloud. All rights reserved.

2. The other party receives a notification that the remote side has exited the room, locally executes to exit room and resets the call status.



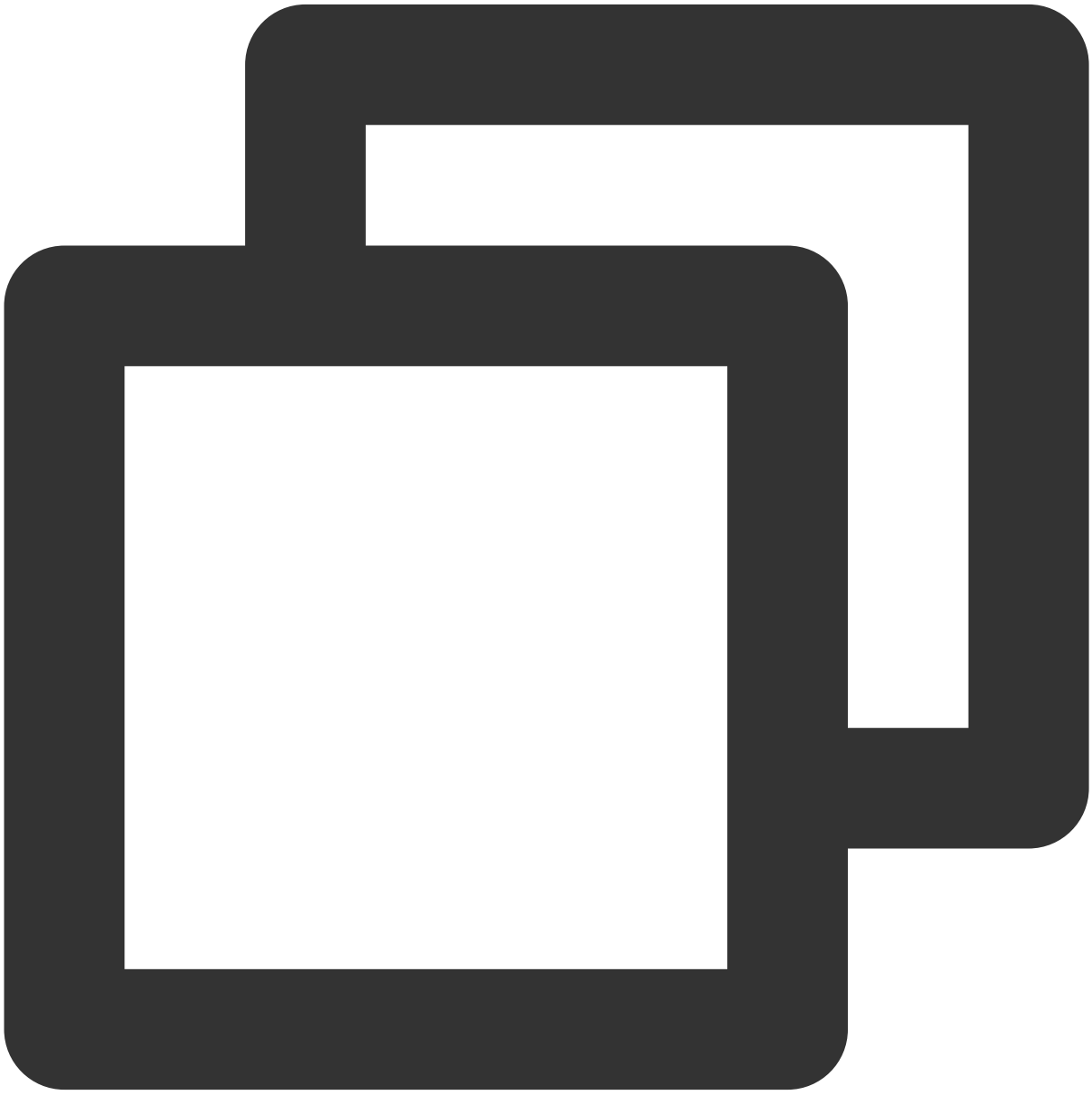
```
@Override
public void onRemoteUserLeaveRoom(String userId, int reason) {
    hangup();
}

@Override
public void onExitRoom(int reason) {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Successfully exited the room and hung up the call
    isOnCalling = false;
}
```

Step 6: feature control

Turn on/off microphone

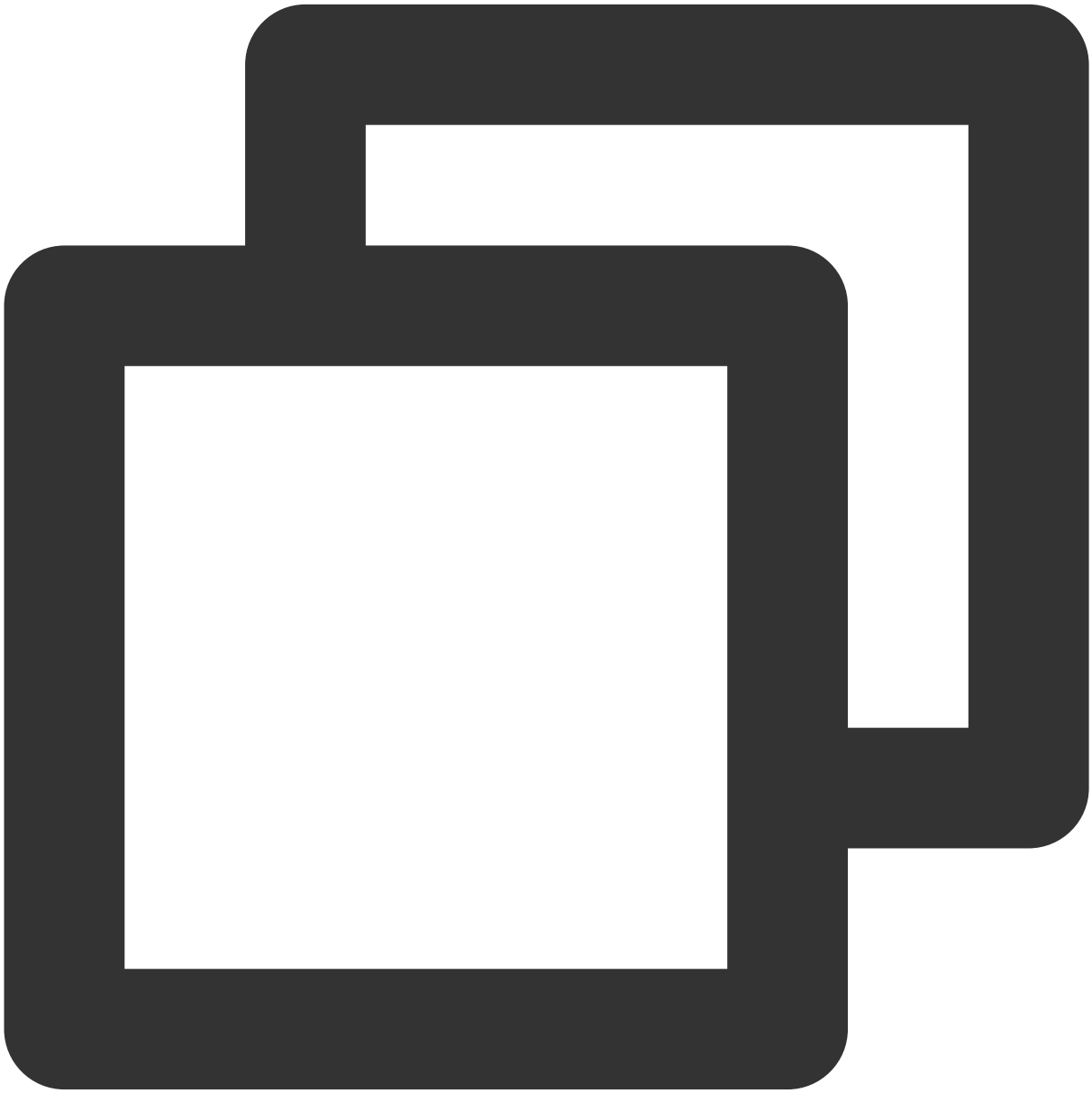


```
// Turn the mic on
mTRTCCloud.muteLocalAudio(false);
// Turn the mic off
```

©2013-2022 Tencent Cloud. All rights reserved.

```
mTRTCCloud.muteLocalAudio(true);
```

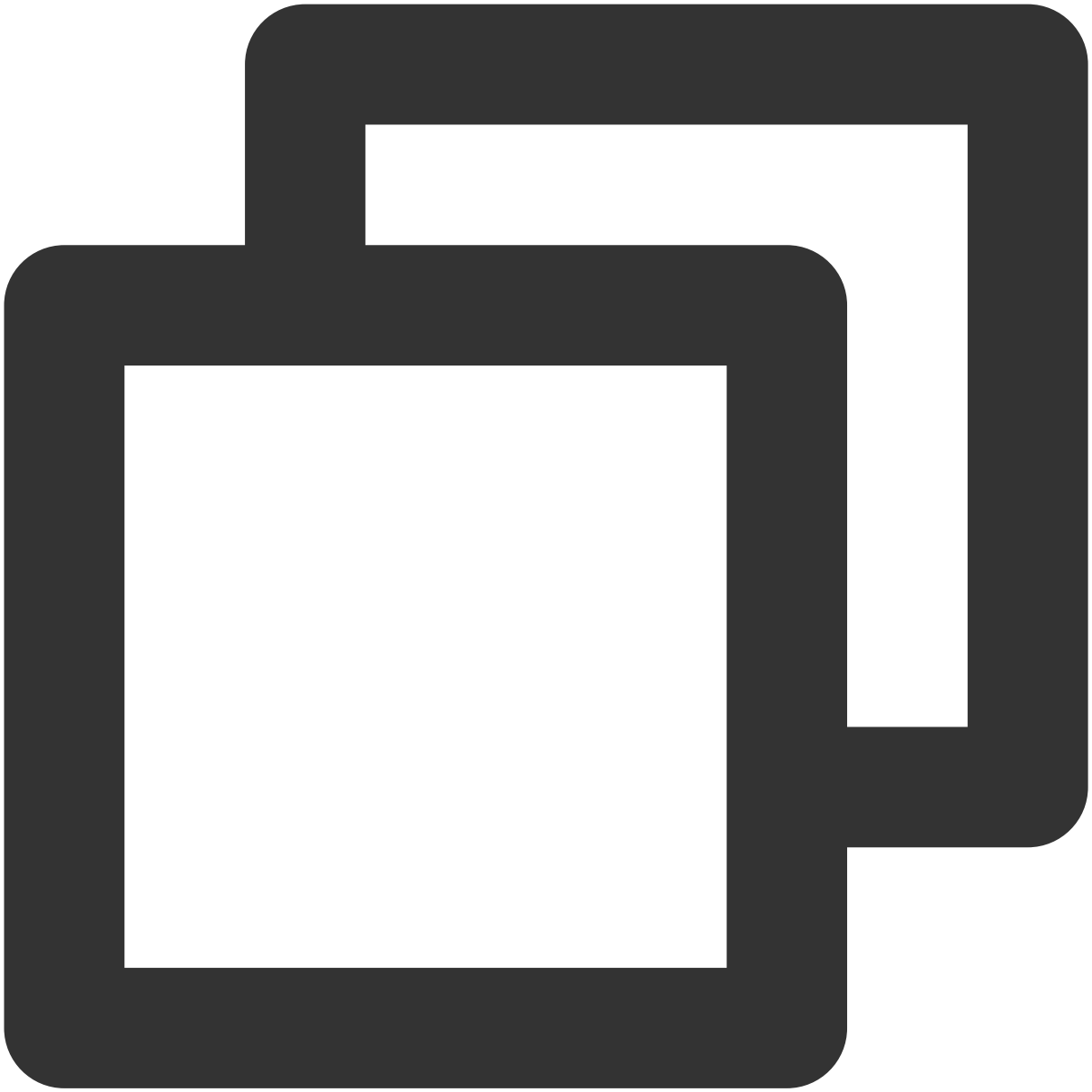
Turn on/off speaker



```
// Turn the speaker on
mTRTCCloud.muteAllRemoteAudio(false);
// Turn the speaker off
mTRTCCloud.muteAllRemoteAudio(true);
```

Turn on/off camera

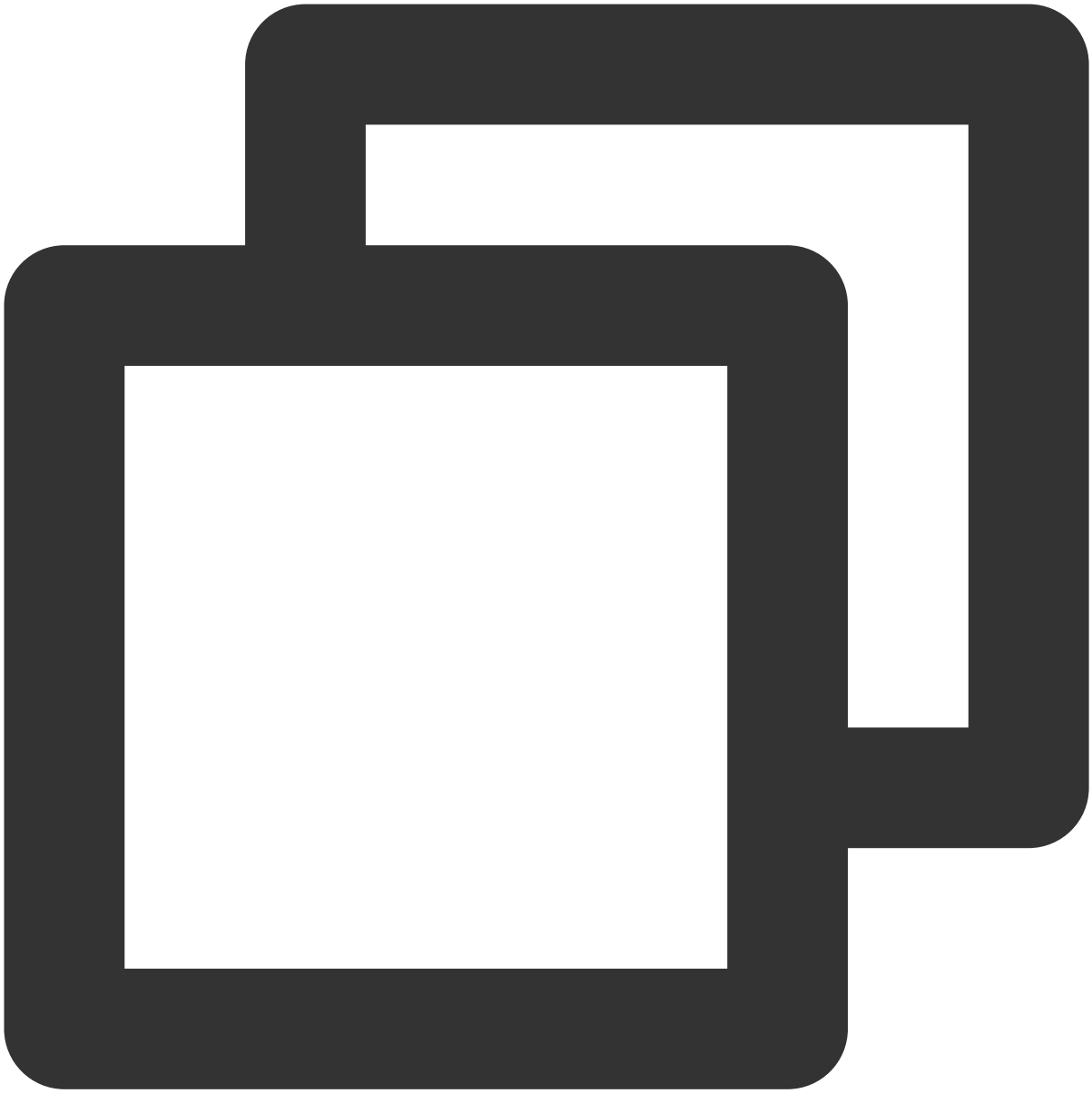
©2013-2022 Tencent Cloud. All rights reserved.



```
// Turn the camera on, specifying front or rear camera and the rendering control
mTRTCCloud.startLocalPreview(isFrontCamera, videoView);
// Turn the camera off
mTRTCCloud.stopLocalPreview();
```

Hands-free/Earpiece Switching

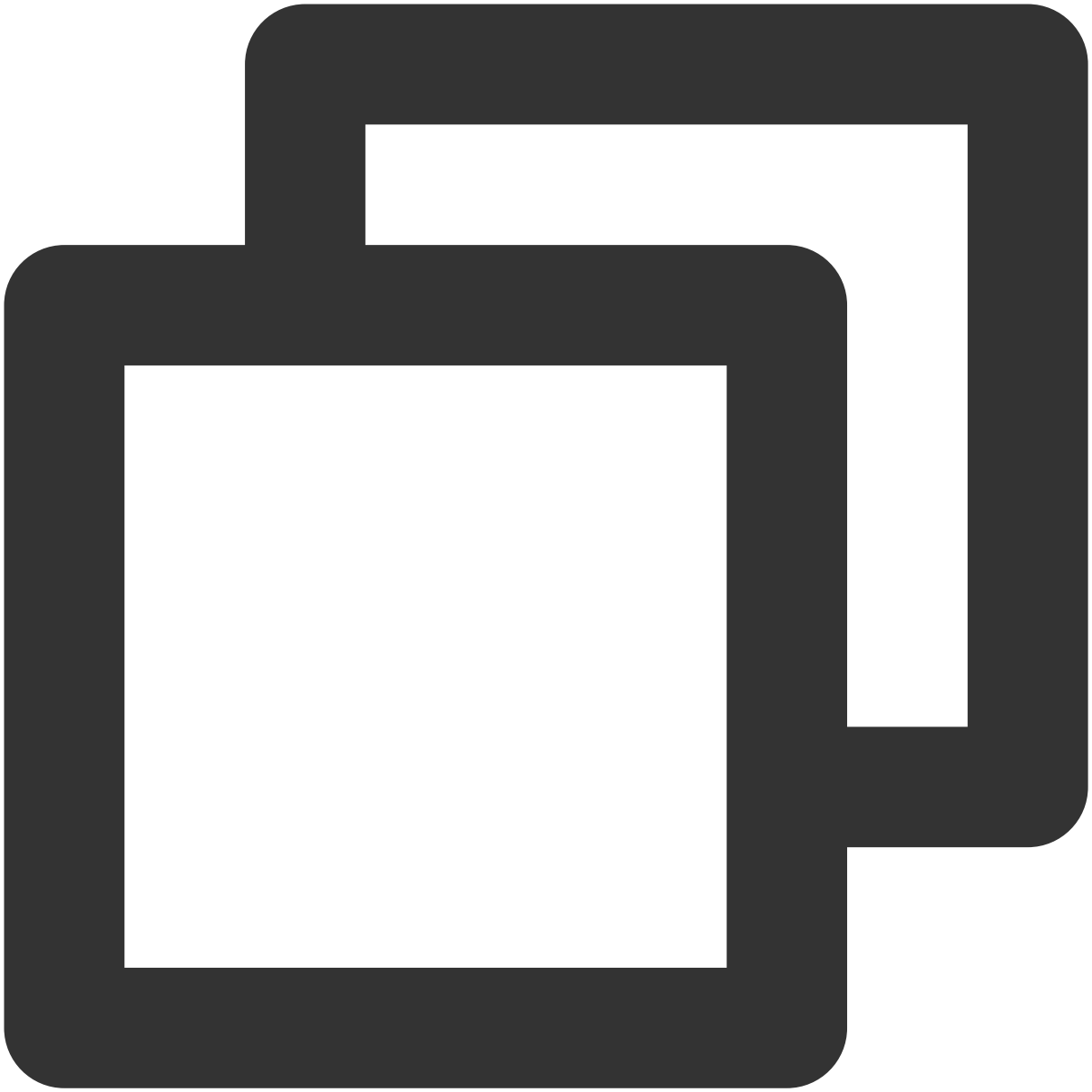
©2013-2022 Tencent Cloud. All rights reserved.



```
// Switch to earpiece
mTRTCCloud.getDeviceManager().setAudioRoute(TXDeviceManager.TXAudioRoute.TXAudioRou
// Switch to speakerphone
mTRTCCloud.getDeviceManager().setAudioRoute(TXDeviceManager.TXAudioRoute.TXAudioRou
```

Camera Switching

©2013-2022 Tencent Cloud. All rights reserved.



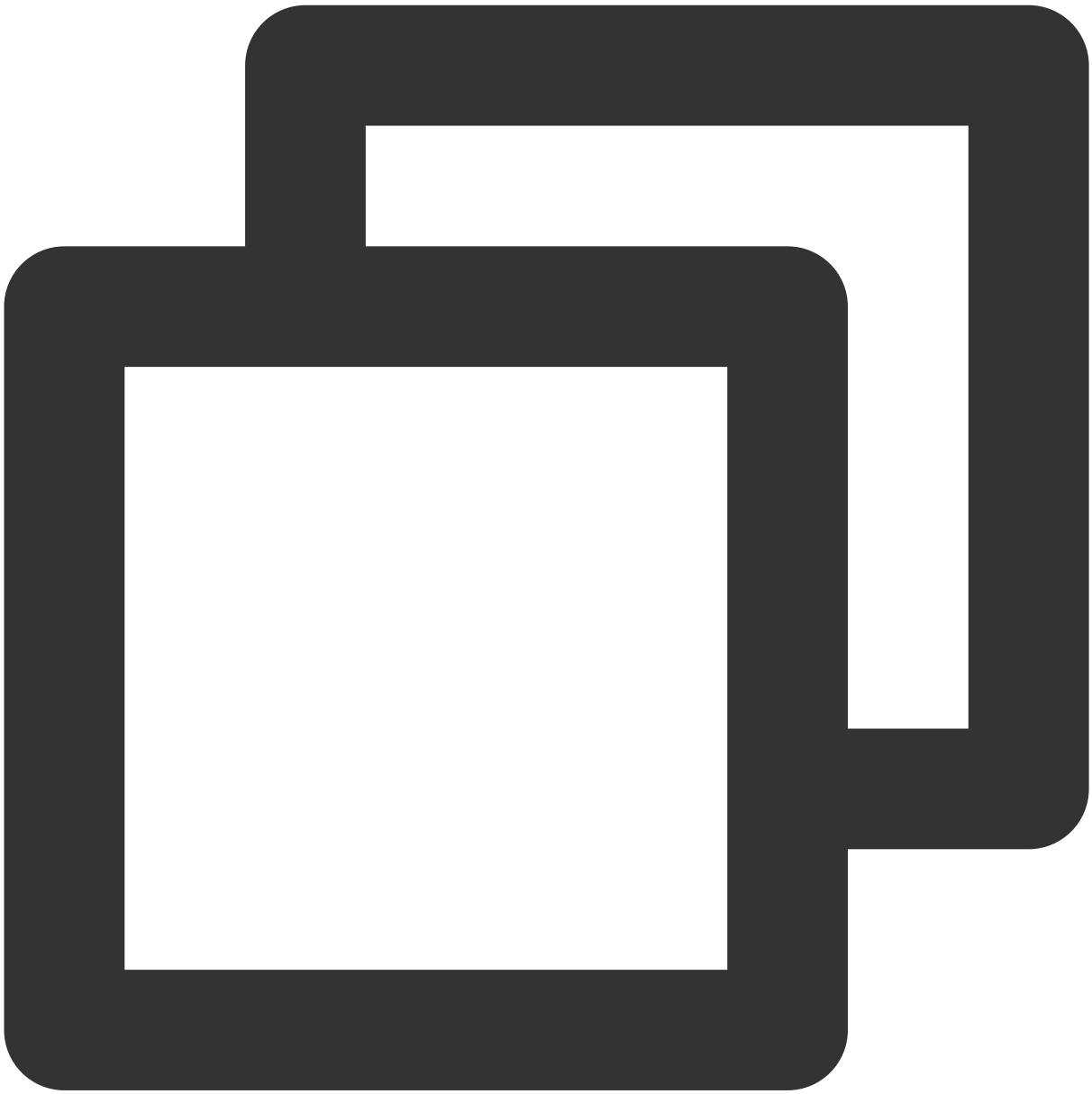
```
// Determine if the current camera is front-facing
boolean isFrontCamera = mTRTCCloud.getDeviceManager().isFrontCamera();
// Switch between front and rear cameras, true: switch to front-facing; false: swit
mTRTCCloud.getDeviceManager().switchCamera(!isFrontCamera);
```

Advanced Features

Network Status Prompt

©2013-2022 Tencent Cloud. All rights reserved.

During audio and video calls, it is often necessary to prompt when the other party's network status is poor, thereby creating an expectation of call lag.



```
@Override
public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality, ArrayList<TRTCC
    if (remoteQuality.size() > 0) {
        switch (remoteQuality.get(0).quality) {
            case TRTCCloudDef.TRTC_QUALITY_Excellent:
                Log.i(TAG, "The other party's network is very good");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Good:
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                Log.i(TAG, "The other party's network is quite good");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Poor:
                Log.i(TAG, "The other party's network is average");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Bad:
                Log.i(TAG, "The other party's network is poor");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Vbad:
                Log.i(TAG, "The other party's network is very poor");
                break;
            case TRTCCloudDef.TRTC_QUALITY_Down:
                Log.i(TAG, "The other party's network is extremely poor");
                break;
            default:
                Log.i(TAG, "Undefined");
                break;
        }
    }
}
```

Note:

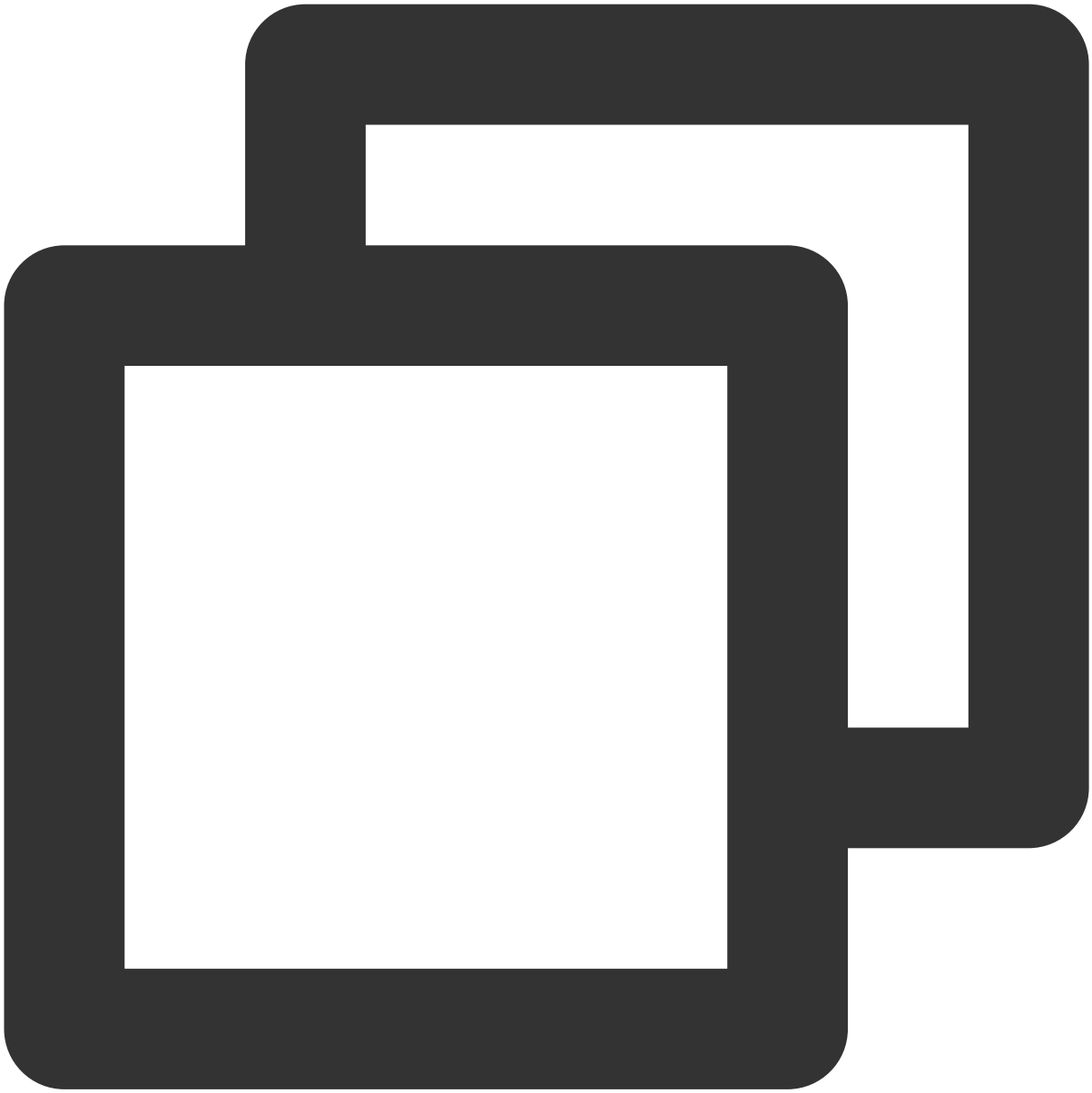
`localQuality` represents the local user network quality assessment result, and its userId field is empty.

`remoteQuality` represents the remote user network quality assessment result, which is influenced by factors on both the remote and local sides.

Call Duration Statistics

It is recommended to use the time when a remote user joins the TRTC room as the start time for calculating call duration, and the time when the local user exits the room as the end time for calculating call duration.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Start call time
long callStartTime = 0;
// End call time
long callFinishTime = 0;
// Call duration (seconds)
long callDuration = 0;

// Callback for remote user entering room
@Override
public void onRemoteUserEnterRoom(String userId) {
    callStartTime = System.currentTimeMillis();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}

// Callback for local user exiting room
@Override
public void onExitRoom(int reason) {
    callFinishTime = System.currentTimeMillis();
    callDuration = (callFinishTime - callStartTime) / 1000;
}
```

Note:

In cases of exceptions such as forced closure or network disconnection, the client may not be able to log the relevant times. These can be monitored through Server Event Callback to track events of entering and exiting the room and calculate the duration of the call.

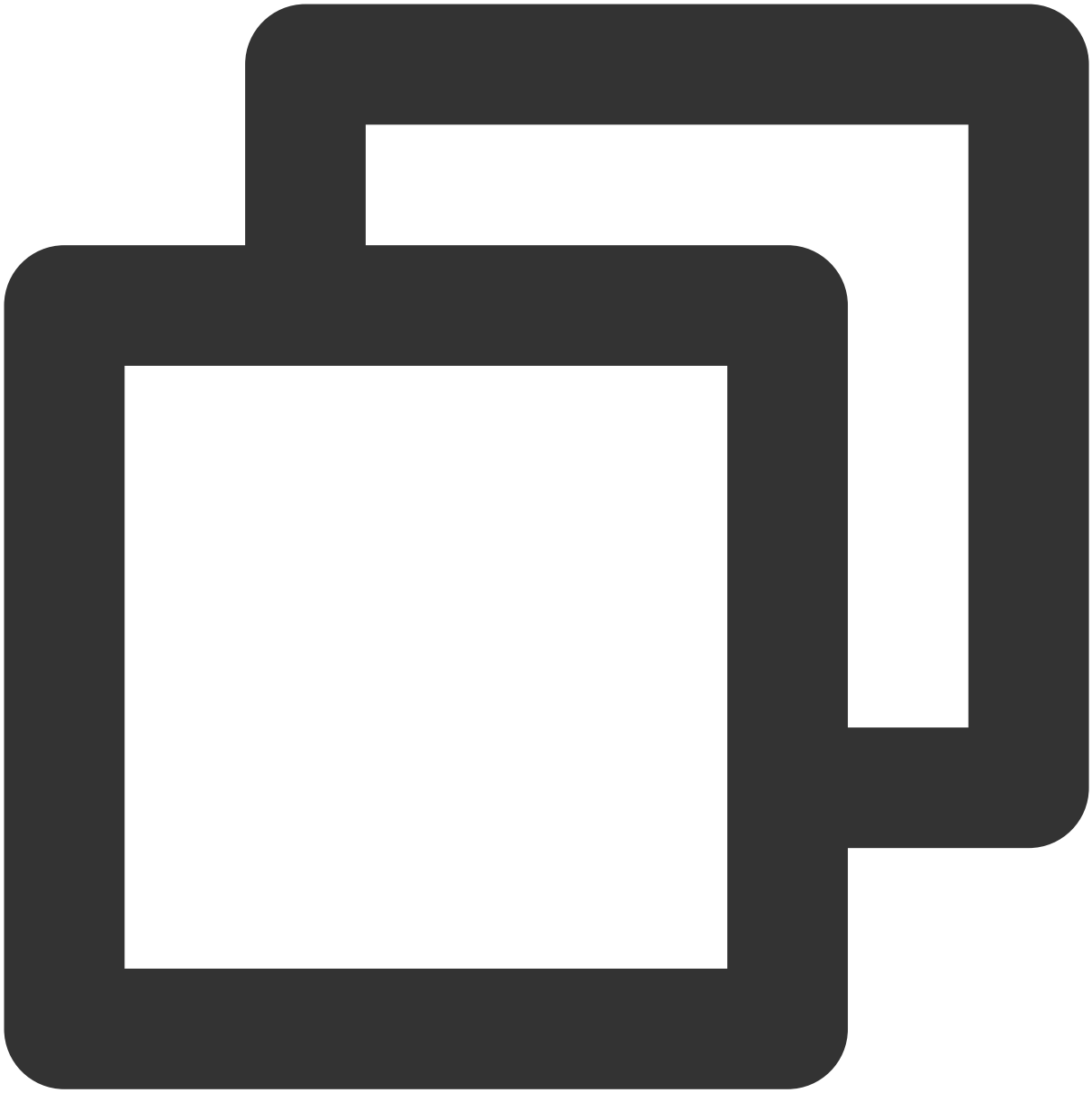
Video Beauty Effects

TRTC supports integrating third-party beauty effect products. Use the example of Special Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate Special Effect SDK, and apply for an authorization license. For details, see Live Show Streaming - Integration Preparation for steps.

2. Resource copying (if any). If your resource files are built into the assets directory, you need to copy them to the App's private directory before use.

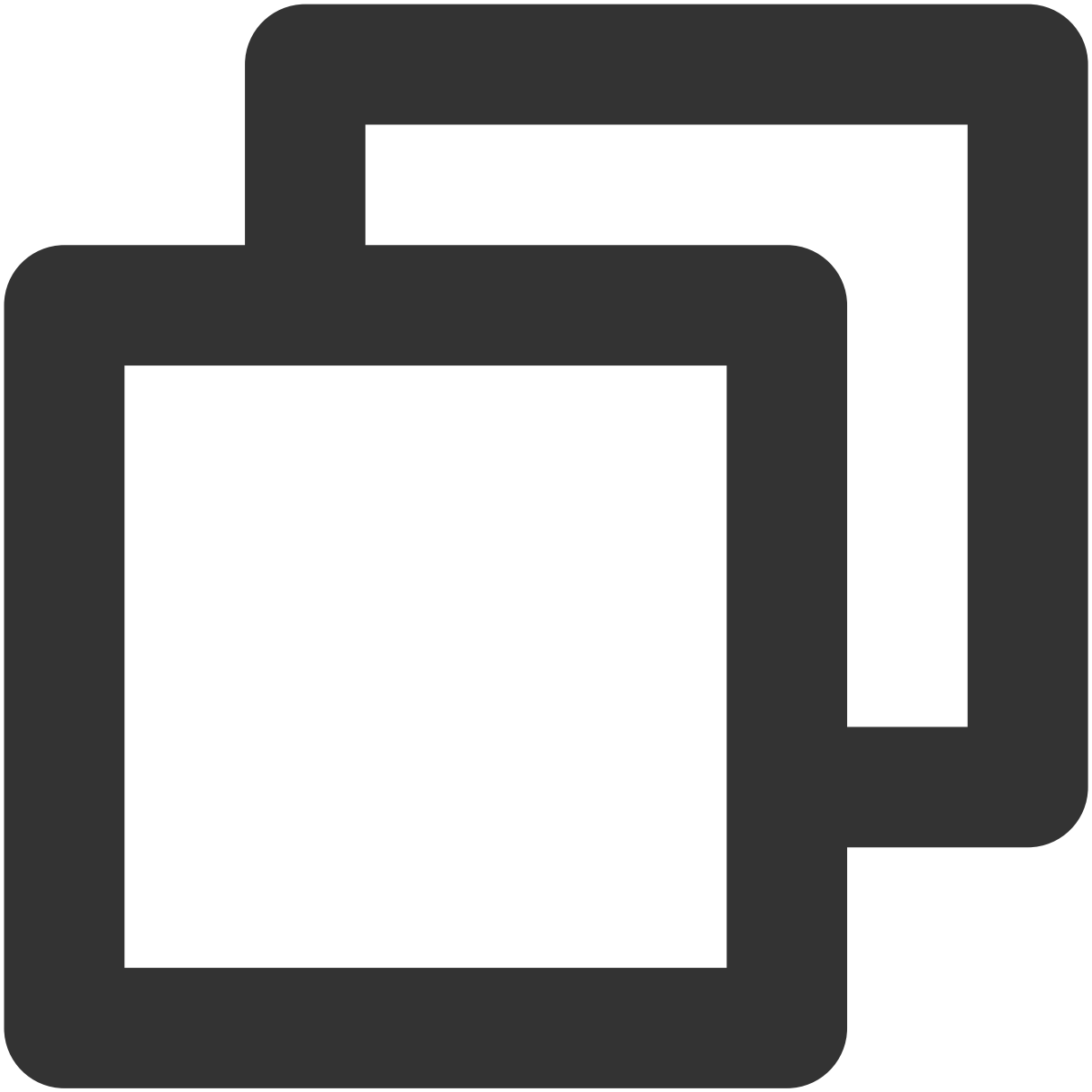
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").getAbsolutePath());
//loading

// Copy resource files to the private directory. Only need to do it once
XmagicResParser.copyRes(getApplicationContext());
```

If your resource file is dynamically downloaded from the internet, you need to set the resource file path after the download is successful.

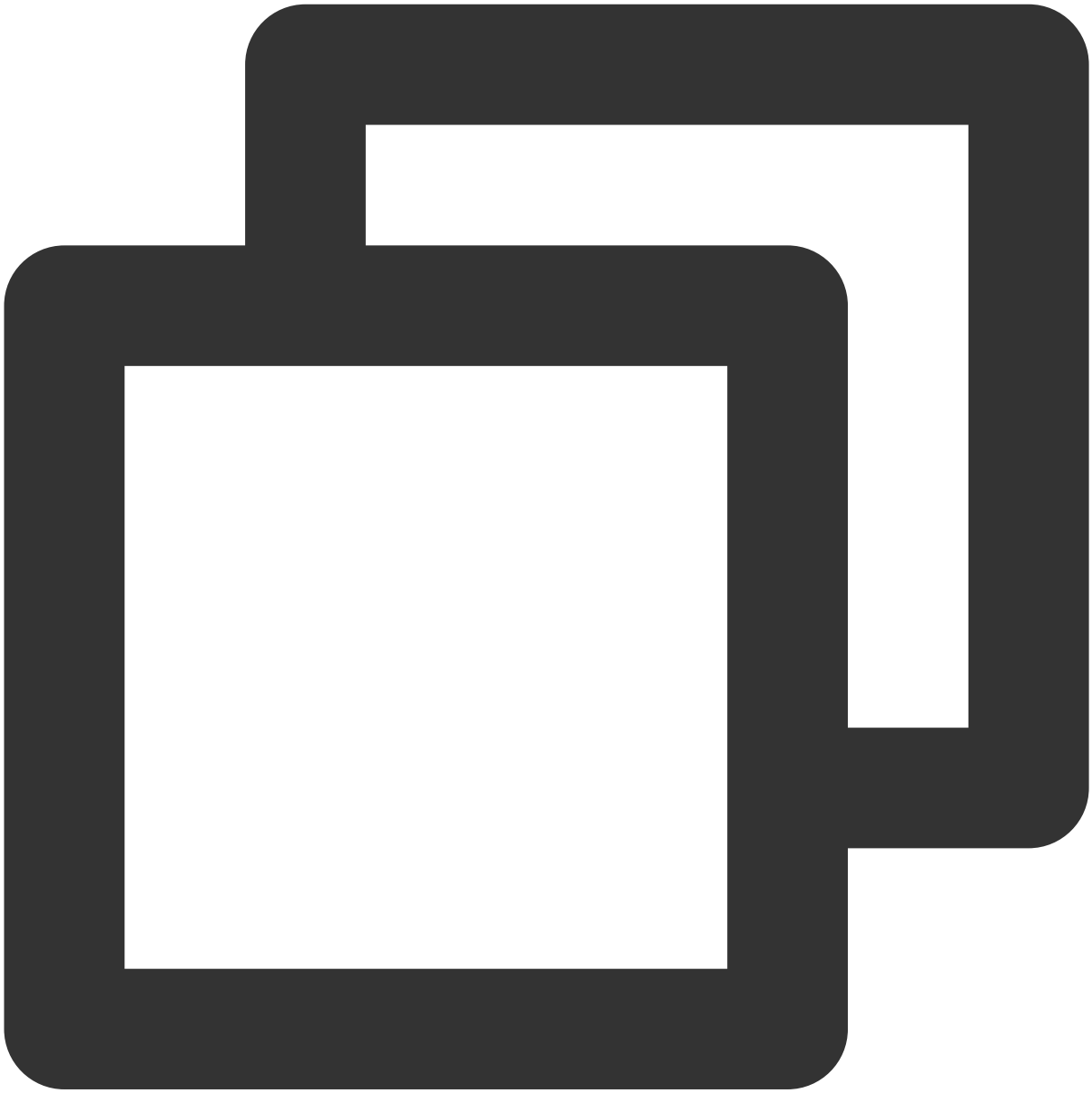
©2013-2022 Tencent Cloud. All rights reserved.



```
XmagicResParser.setResPath (local path of the downloaded resource file);
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
mTRTCCloud.setLocalVideoProcessListener(TRTCCloudDef.TRTC_VIDEO_PIXEL_FORMAT_Textur
    @Override
    public void onGLContextCreated() {
        // The OpenGL environment has already been set up internally within the SDK
        if (mXmagicApi == null) {
            XmagicApi mXmagicApi = new XmagicApi(context, XmagicResParser.getResPat
        } else {
            mXmagicApi.onResume();
        }
    }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    @Override
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudD
        // Callback for integrating with third-party beauty components for video pr
        if (mXmagicApi != null) {
            dstFrame.texture.textureId = mXmagicApi.process(srcFrame.texture.textur
        }
        return 0;
    }

    @Override
    public void onGLContextDestory() {
        // The internal OpenGL environment within the SDK has been terminated. At t
        mXmagicApi.onDestroy();
    }
});
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products. **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

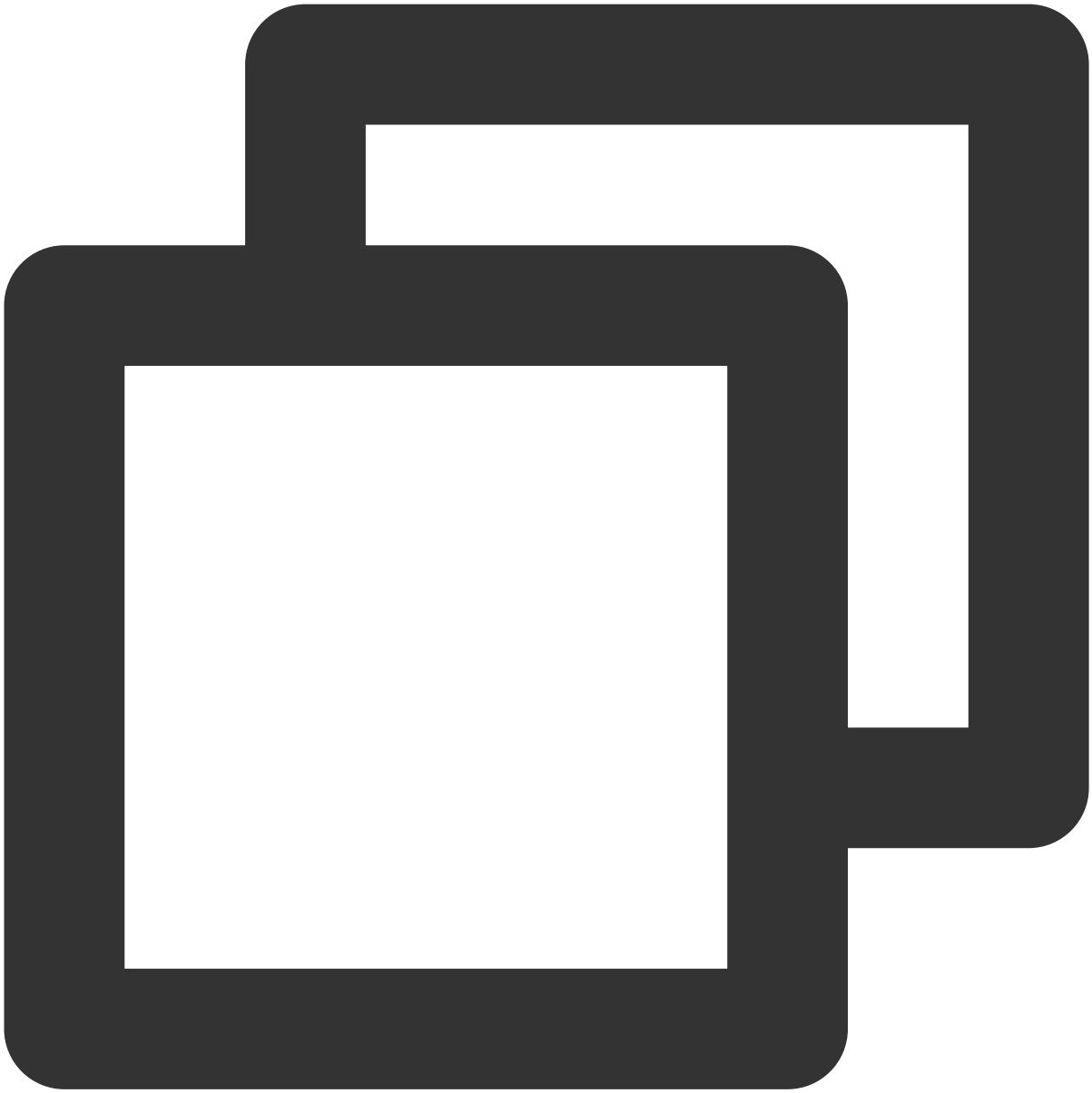
For scenario-specific integration guidelines of beauty effects, see Integrating Special Effect into TRTC SDK. For guidelines on integrating beauty effects independently, see Integrating Special Effect SDK.

Window Size Switching

In TRTC, there are many APIs that require you to control the video screen. All these APIs require you to specify a video rendering control. On the Android platform, `TXCloudVideoView` is used as the video rendering control, and both `SurfaceView` and `TextureView` rendering schemes are supported. Below are the methods for specifying the type of rendering control and updating the video rendering control.

1. If you want mandatory use of a certain scheme, or to convert the local video rendering control to `TXCloudVideoView`, you can code as follows.

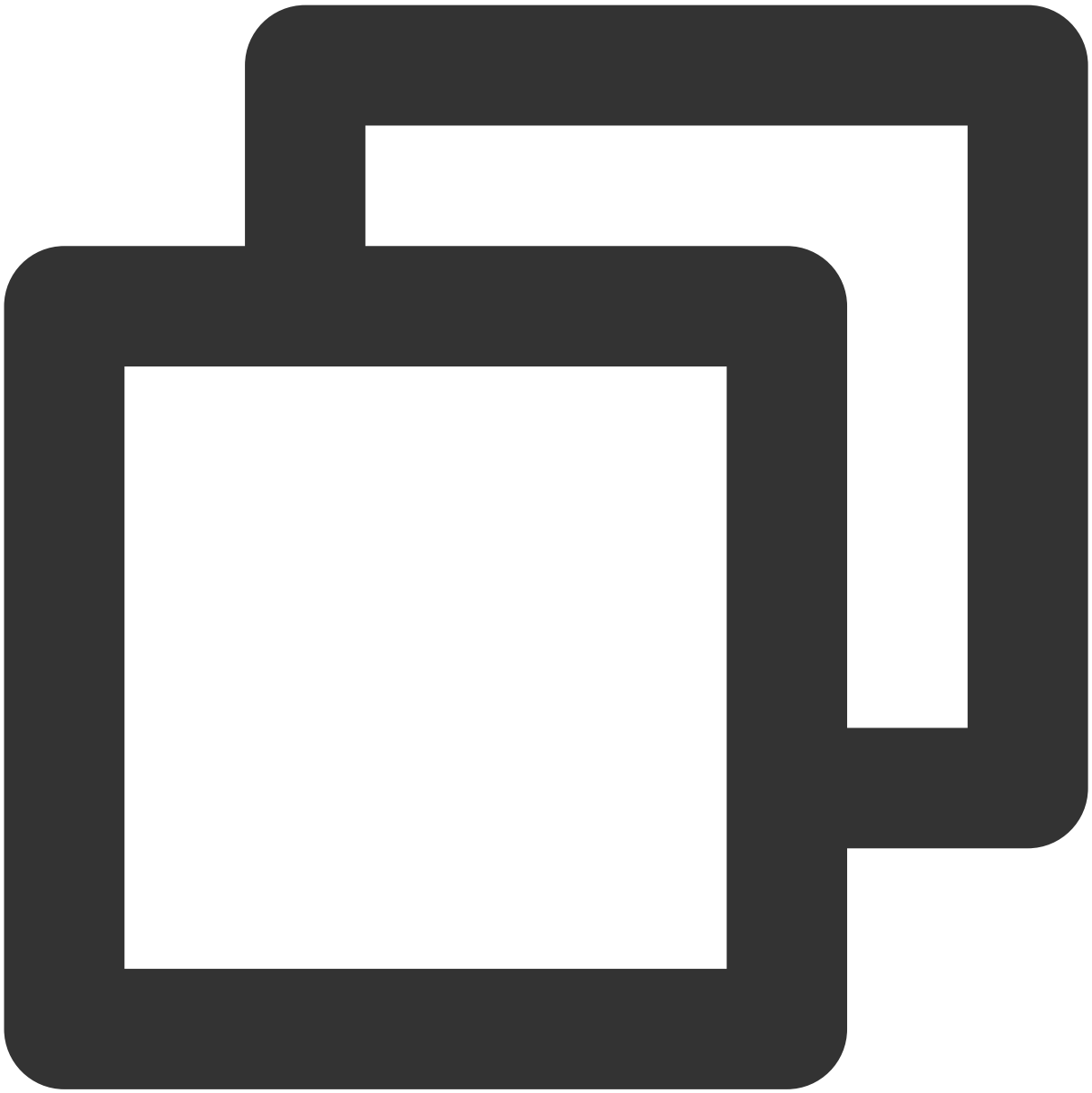
©2013-2022 Tencent Cloud. All rights reserved.



```
// Mandatory use of TextureView
TextureView textureView = findViewById(R.id.texture_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(context);
cloudVideoView.addVideoView(textureView);

// Mandatory use of SurfaceView
SurfaceView surfaceView = findViewById(R.id.surface_view);
TXCloudVideoView cloudVideoView = new TXCloudVideoView(surfaceView);
```

©2013-2022 Tencent Cloud. All rights reserved.

2. If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.



```
// Update local preview screen rendering control
mTRTCCloud.updateLocalView(videoView);

// Update the remote user's video rendering control
mTRTCCloud.updateRemoteView(userId, streamType, videoView);
```

Note:

©2013-2022 Tencent Cloud. All rights reserved.

`videoView` is a target video rendering control of type `TXCloudVideoView`. `streamType` only supports `TRTC_VIDEO_STREAM_TYPE_BIG` and `TRTC_VIDEO_STREAM_TYPE_SUB`.

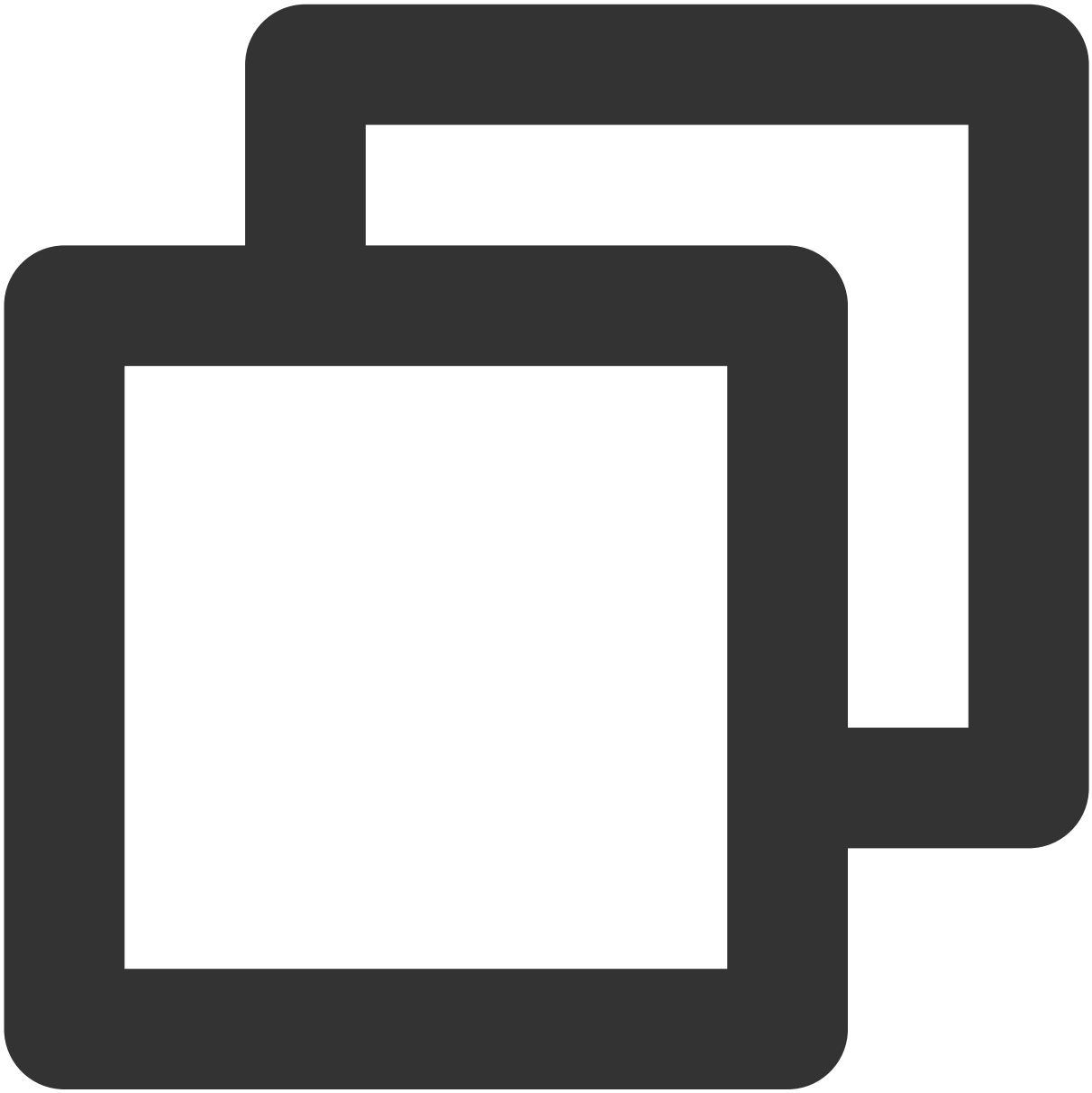
Offline Push Message

In audio/video call scenarios, the offline push message feature is usually necessary, allowing the called user's App to receive new incoming call messages even when it's not online. For detailed guidance on integrating offline push, see Offline Message Push. Below, we will focus on explaining the implementation of step 7: Send Offline Push Message, and step 8: Parse Offline Push Messages.

Send Offline Push Message

When sending a call invitation using invite, you can set offline push parameters through V2TIMOfflinePushInfo. By calling ext of V2TIMOfflinePushInfo to set custom ext data, when the user receives an offline push message to start the App, they can obtain the ext field in the callback of clicking the notification, and then redirect to the specified UI interface based on the content of the ext field.

©2013-2022 Tencent Cloud. All rights reserved.



```
JSONObject contentObj = new JSONObject();
try {
    contentObj.put("cmd", "av_call");
    JSONObject contentDetailObj = new JSONObject();
    contentDetailObj.put("callType", "videoCall");
    contentDetailObj.put("roomId", generateRoomId());
    contentObj.put("cmdInfo", contentDetailObj);
} catch (JSONException e) {
    e.printStackTrace();
}
String data = contentObj.toString();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// OfflineMessageContainerBean is the Javabean corresponding to the pass-through pa
OfflineMessageContainerBean containerBean = new OfflineMessageContainerBean();
OfflineMessageBean entity = new OfflineMessageBean();
entity.content = data;
entity.sender = TUILogin.getLoginUser();
entity.action = OfflineMessageBean.REDIRECT_ACTION_CALL;entity.sendTime = System.cu
v2TIMOfflinePushInfo.setAndroidHuaWeiCategory("IM");
v2TIMOfflinePushInfo.setAndroidVIVOCategory("IM");
v2TIMOfflinePushInfo.setTitle(mNickname);v2TIMOfflinePushInfo.setDesc("You have a n
v2TIMOfflinePushInfo.setAndroidSound("phone_ringing");

V2TIMManager.getSignalingManager().invite(receiver, data, false, v2TIMOfflinePushIn
    @Override
    public void onError(int code, String desc) {
        // Failed to send call invitation
    }

    @Override
    public void onSuccess() {
        // Successfully send call invitation
    }
});
```

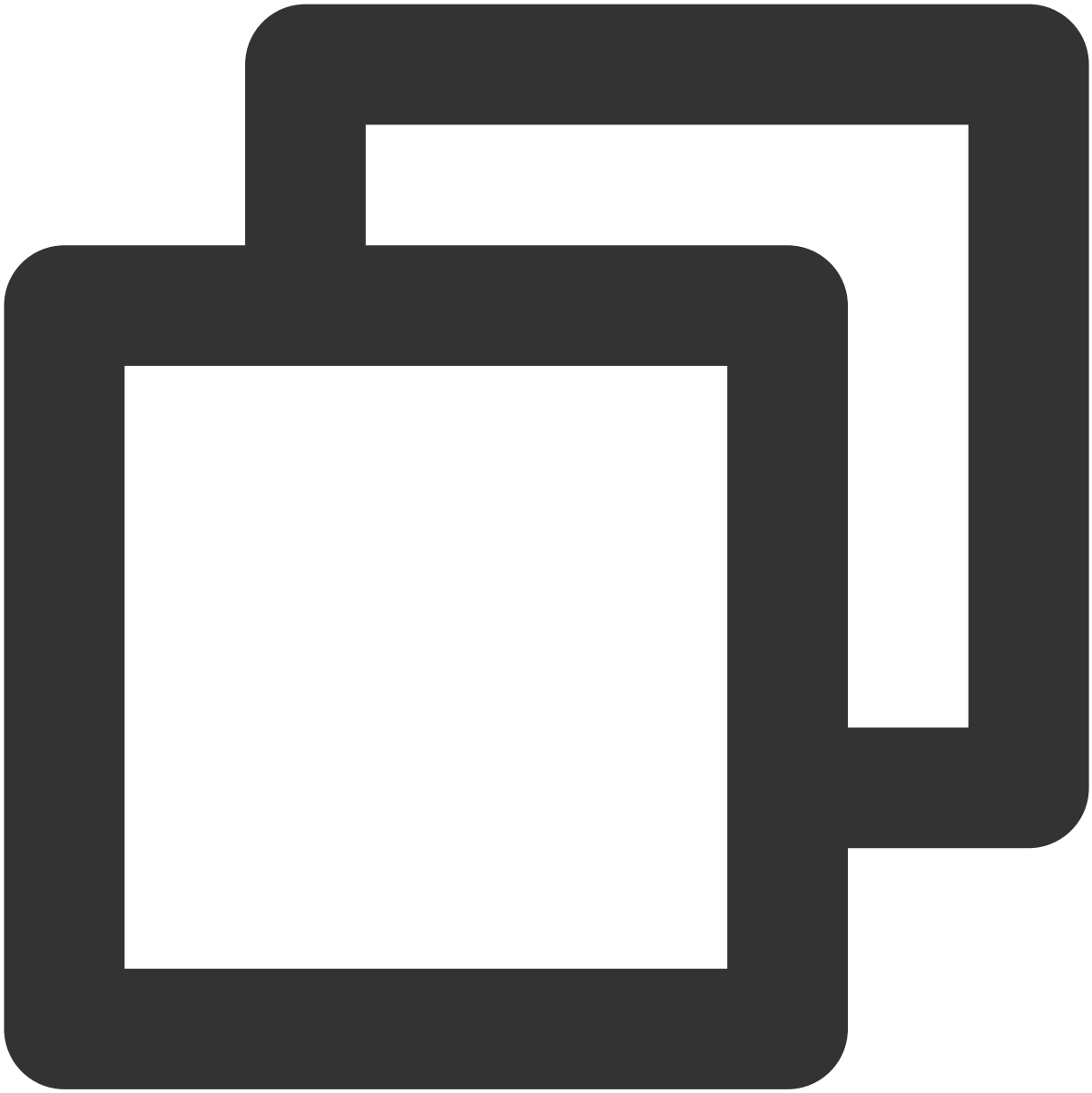
Note:

To be compatible with the parsing format of offline push messages on different platforms, it is recommended to use the wrapper class <1>OfflineMessageContainerBean to set the pass-through parameter ext.

Parse Offline Push Messages

When an offline push message in the notification column is received and clicked, it will automatically redirect to the interface you configured earlier. You can retrieve the passed offline push parameters in the onResume() method of the interface startup by calling getIntent().getExtras(), and then custom the redirection. For details, see the handleOfflinePush() method in TUIKitDemo.

©2013-2022 Tencent Cloud. All rights reserved.



```
private void handleOfflinePush() {
    // Determine whether to log in to IM again based on the log-in status
    // 1. If the log-in status is V2TIMManager.V2TIM_STATUS_LOGOUT, you will redire
    if (V2TIMManager.getInstance().getLoginStatus() == V2TIMManager.V2TIM_STATUS_LO
        Intent intent = new Intent(MainActivity.this, SplashActivity.class);
        if (getIntent() != null) {
            intent.putExtras(getIntent());
        }
        startActivity(intent);
        finish();
        return;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }

    // 2. Otherwise, it means the app is just in the background, directly parse the
    final OfflineMessageBean bean = OfflineMessageDispatcher.parseOfflineMessage(ge
        if (bean != null) {
            setIntent(null);
            NotificationManager manager = (NotificationManager) getSystemService(Co
            if (manager != null) {
                manager.cancelAll();
            }

            if (bean.action == OfflineMessageBean.REDIRECT_ACTION_CALL) {
                Intent startActivityIntent = new Intent(context, MyCustomActivity.c
                startActivityIntent.putExtras(getIntent());
                startActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(startActivityIntent);
            }
        }
}
```

Note:

By clicking the message in the Notification colunm in FCM, you will by default redirect to the application's default Launcher interface. You can retrieve the passed offline push parameters by calling getIntent().getExtras() in the onResume() method of the interface startup, and then custom the redirection.

Exception Handling

TRTC exception error handling

When the TRTC SDK encounters an unrecoverable error, the error is thrown in the `onError` callback. For details, see Error Code Table.

UserSig related

UserSig verification failure leads to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

©2013-2022 Tencent Cloud. All rights reserved.

Room entry and exit related

If room entry is failed, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect.Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Nnote that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

Device related

Errors for related monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to enable the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having

©2013-2022 Tencent Cloud. All rights reserved.

		denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Troubleshooting for not receiving offline push

1. **OPPO Phone**

General reasons for not receiving push notifications on OPPO phones include:

As required by OPPO push official website, ChannelID must be configured on OPPO phones running Android 8.0 or higher. Otherwise, the push message cannot be displayed. For the configuration method, see OPPO Push Configuration.

In the message, if the </1> custom content for offline push pass-through custom content for offline push pass-through <1> is not in JSON format, the OPPO mobile phone will not receive the push.

The notification column display feature is disabled by default for OPPO installation application. You need to check the switch status.

2. **Send a message as a custom message**

The offline push for custom messages is different from that for normal messages. As we cannot parse the content of custom messages, we cannot determine the push's content. Therefore, by default, there is no offline push. If you need an offline push, you need to set the desc field in offlinePushInfo when using sendMessage, and the desc information will by default be displayed during the push.

3. **Device notification column settings impact**

The direct manifestation of offline push is notification column alerts. Thus, like other notifications, it is subject to device notification settings. Take Huawei as an example:

"Settings - Notifications - Lock Screen Notifications - Hide or Do Not Show Notifications" will affect the display of offline push notifications when the screen is locked.

"Settings - Notifications - More Notification Settings - Show Notification Icons (Status Column)" will affect the display of offline push notification icon in the status column.

"Settings - Notifications - Application Notifications Management - Allow Notifications" will directly affect the display of offline push notifications.

"Settings - Notifications - Application Notifications Management - Notification Sound" and "Settings - Notifications - Application Notifications Management - Notification Mute" will affect the offline push notification ringtone.

4. **After the process is completed, the offline push cannot be received**

©2013-2022 Tencent Cloud. All rights reserved.

First, verify whether normal push is possible using the Offline Testing Tool in the IM Console. In cases of push failure with an abnormal device status, check if the parameters configured in the IM console are correct. Additionally, verify the code initiation and registration logic, including whether the manufacturer push service registration and the IM offline push configuration setup are correctly set. For push failures with a normal device status, check if the Channel ID is correctly filled in or if the backend service operates normally.
The offline push feature relies on the vendor's capabilities. Some simple characters may be filtered by the vendor and cannot be passed through and pushed.
If offline push messages are not pushed timely or cannot be received, you need to check the vendor's push restrictions.

Failed to redirect to the page

Click the notification column of an offline push message to redirect to the specified interface. The backend delivers the redirection modes and page parameters that you configure for various vendors in the console to vendor servers based on vendor API rules. When you click the notification column for offline push messages, the system opens and redirects to the corresponding page. The opening of the corresponding page also depends on the manifest file. Only when the configuration in the manifest file is consistent with that in the console, the corresponding page can be opened and redirected properly.
1. First, you need to check whether the configuration in the console and that in the manifest file are correct and consistent with each other. For more information, see the TUIKitDemo configuration. Note that the API modes may vary by vendors.
2. If the system redirects to the configuration page, you need to check whether the parsing of offline messages on the configuration page and the page redirection are proper.

©2013-2022 Tencent Cloud. All rights reserved.

iOS

Last updated：2024-07-18 14:26:14

Business Process

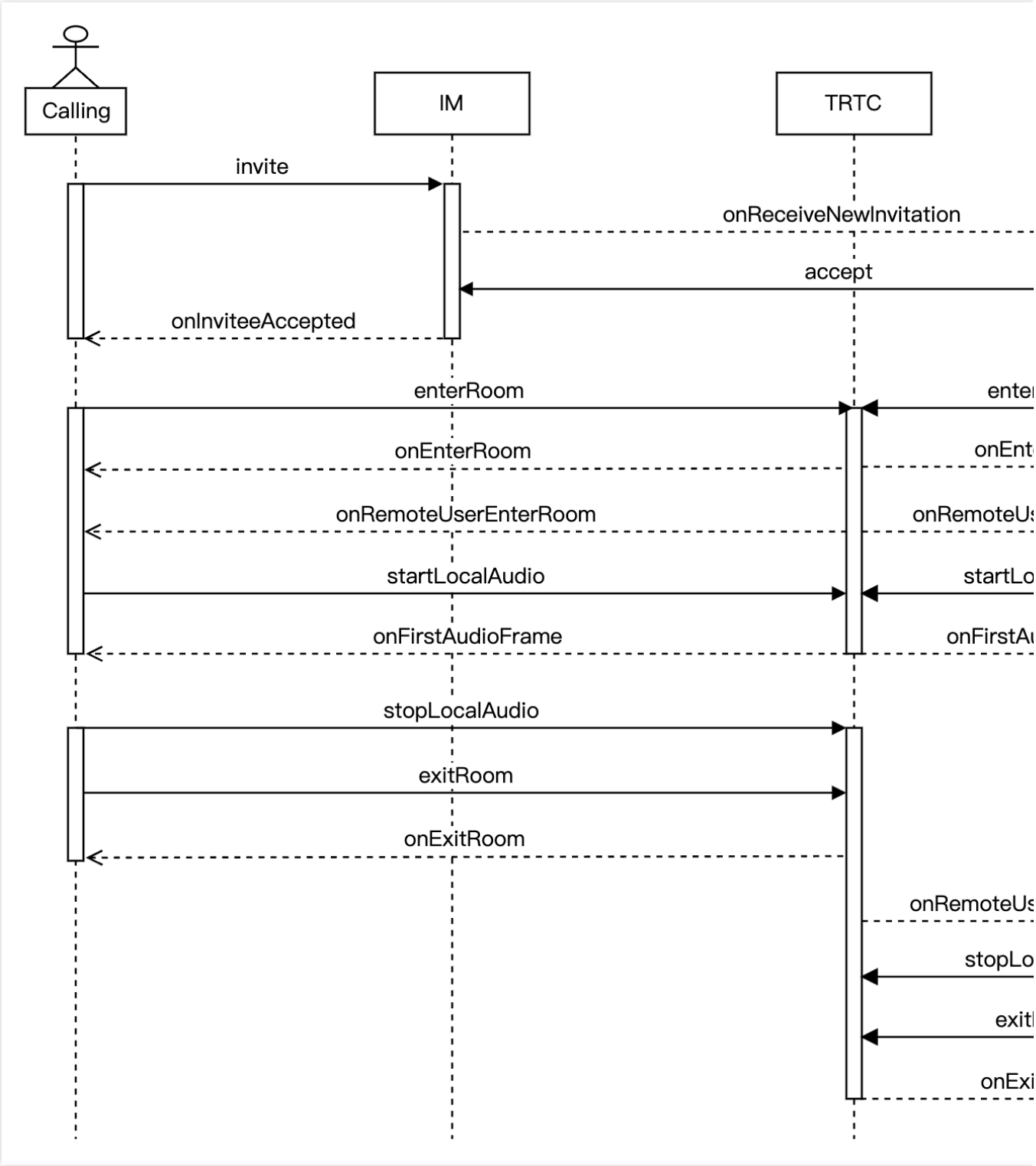
This document summarizes some common business processes in one-to-one audio and video calls, helping you better understand the implementation process of the entire scenario.

Audio Call Process

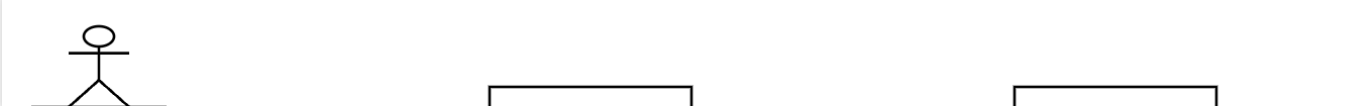
Video Call Process

The following diagram shows the sequence of one-to-one audio call, including processes such as calling, answering, talking, and hanging up.

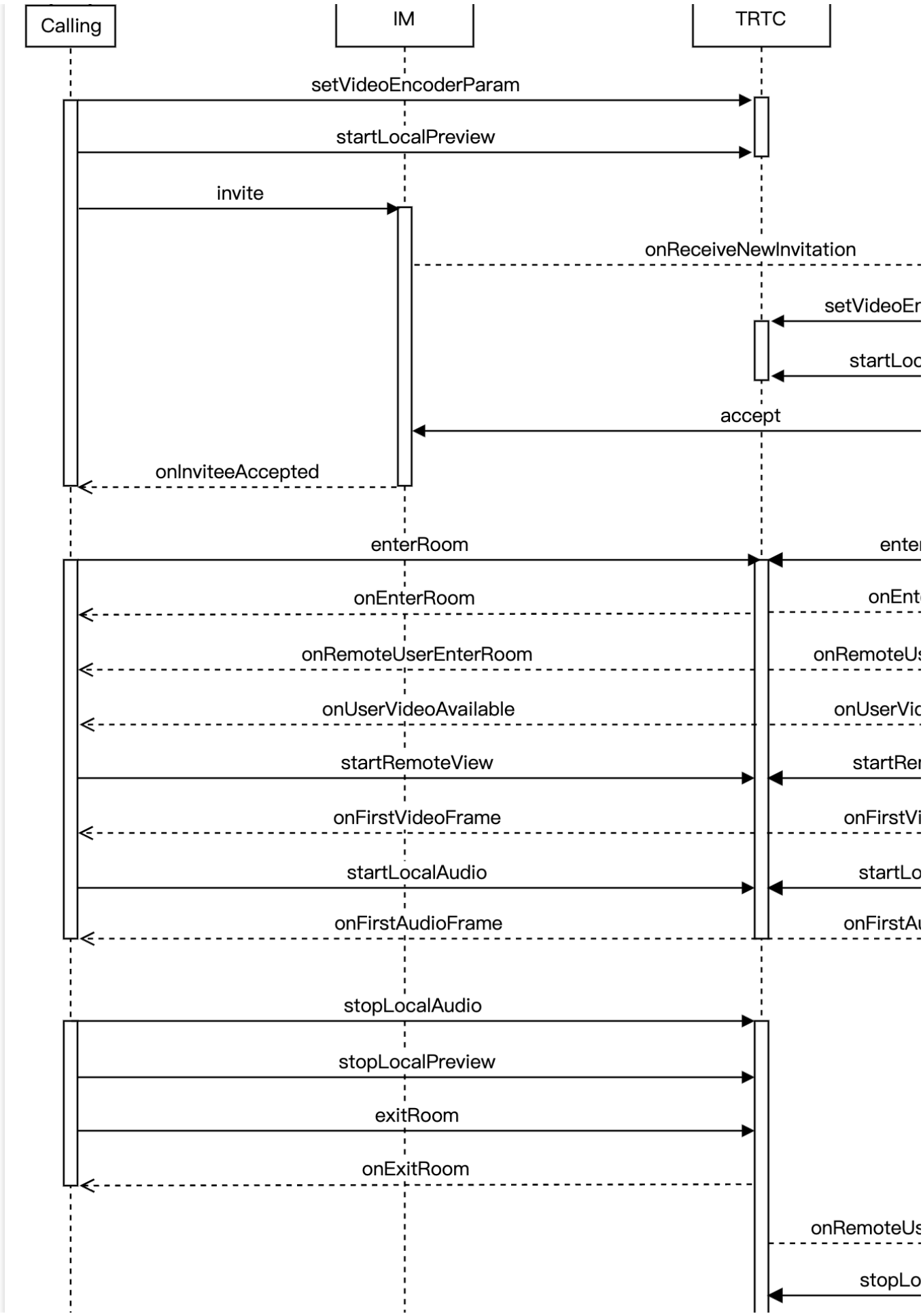
©2013-2022 Tencent Cloud. All rights reserved.



The following diagram shows the sequence of one-to-one video call, including processes such as calling, answering, talking, and hanging up.



©2013-2022 Tencent Cloud. All rights reserved.



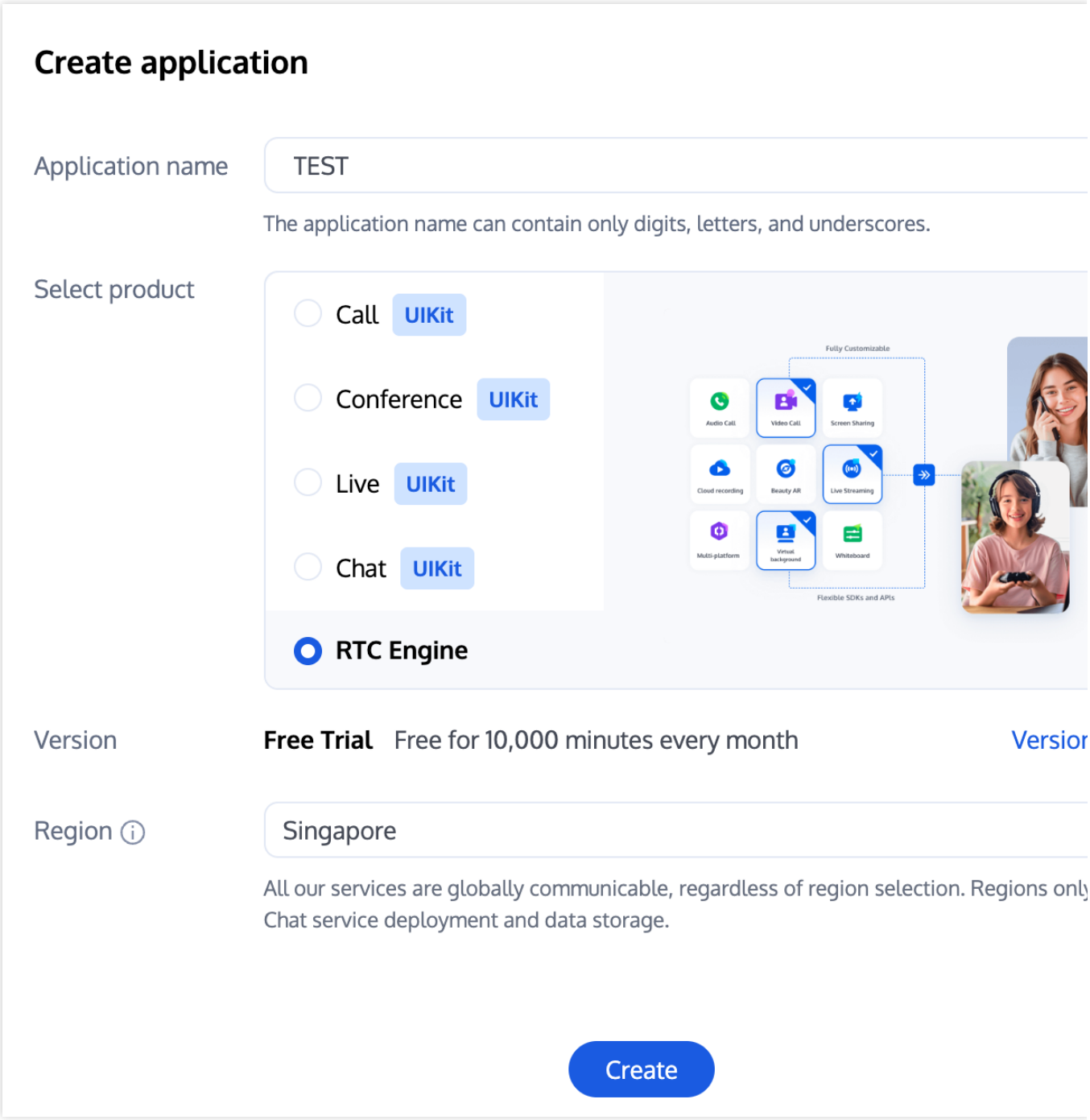
©2013-2022 Tencent Cloud. All rights reserved.

Integration Preparations

Step 1: activate the service

One-to-one audio and video call scenarios usually require dependencies on two paid PaaS services from the cloud platform, Instant Messaging (IM) and Real-Time Communication (TRTC) for construction.
1. First, you need to log in to the TRTC Console to create an application. At this time, an IM trial application with the same SDKAppID as the current TRTC application will be automatically created in the Instant Messaging (IM) Console. The accounts and authentication systems for both can be reused. Subsequently, you can choose to upgrade the TRTC or IM application version as needed. For example, the advanced versions can unlock more value-added features and services.

©2013-2022 Tencent Cloud. All rights reserved.



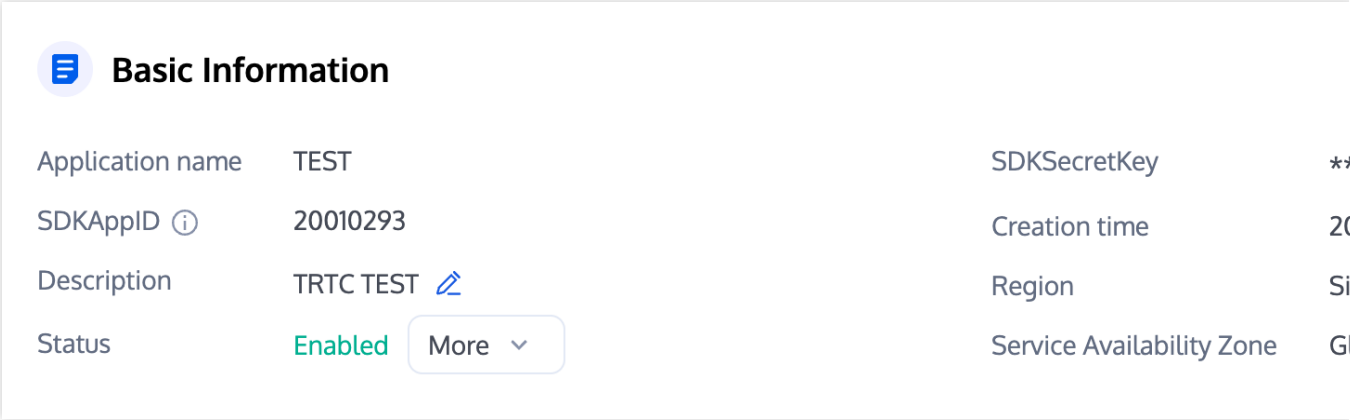
Note:

It is recommended to create two separate applications for testing and production environments. Each account (UIN) is provided with 10,000 minutes of free usage per month within one year.

The TRTC monthly package is divided into Trial Version (by default), Basic Version, and Professional Version, which can unlock different value-added features and services. For details, see Version Features and Monthly Package Description.

2. Once the application is created, you can find the basic information about it under the **Application Management** > **Application Overview** section. It is important to store the **SDKAppID** and **SDKSecretKey** for later use and to avoid key leakage to prevent unauthorized traffic usage.

©2013-2022 Tencent Cloud. All rights reserved.



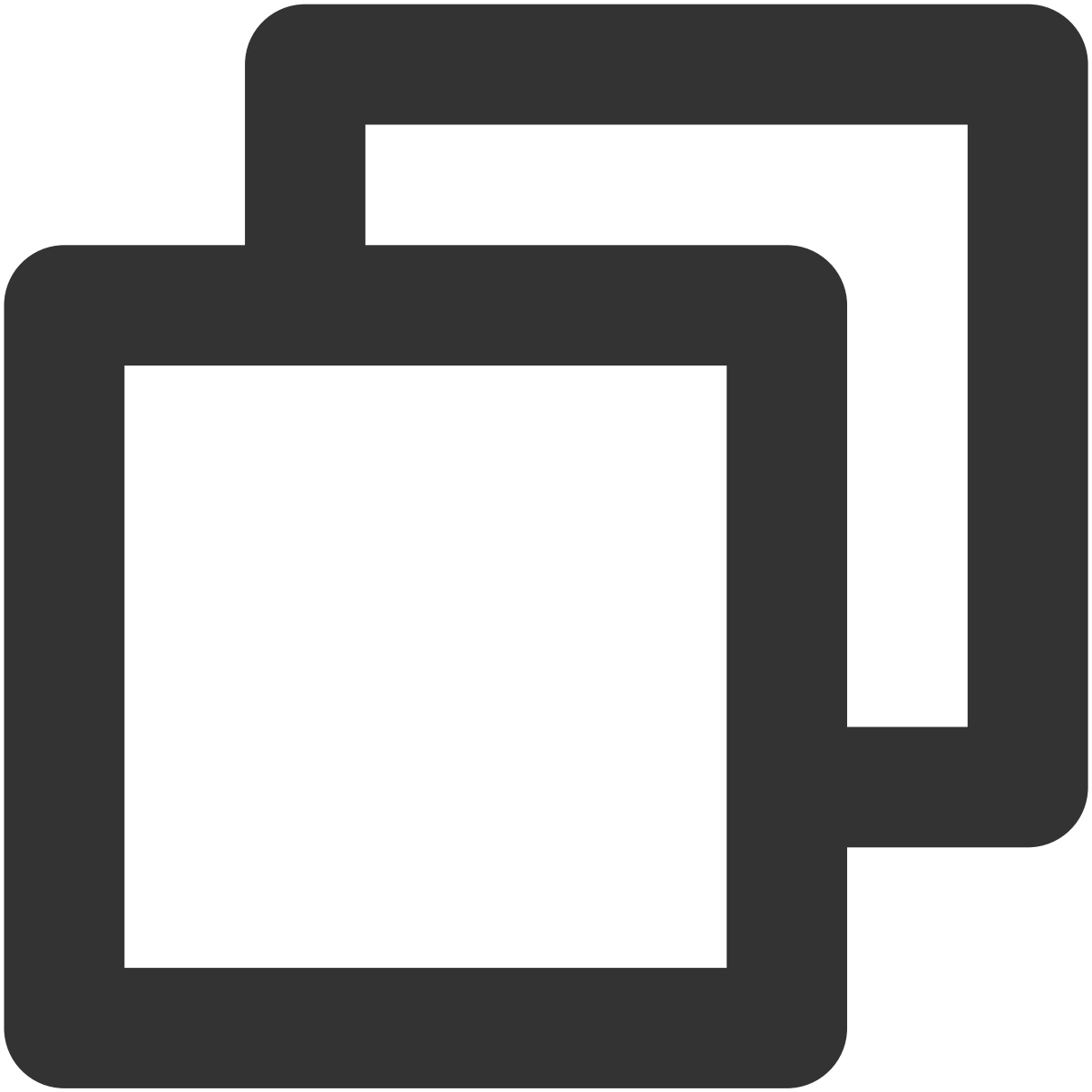
Step 2: import SDK

TRTC SDK and IM SDK are now available on **CocoaPods**. It is recommended to integrate SDKs through CocoaPods.

1. Install CocoaPods

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

©2013-2022 Tencent Cloud. All rights reserved.

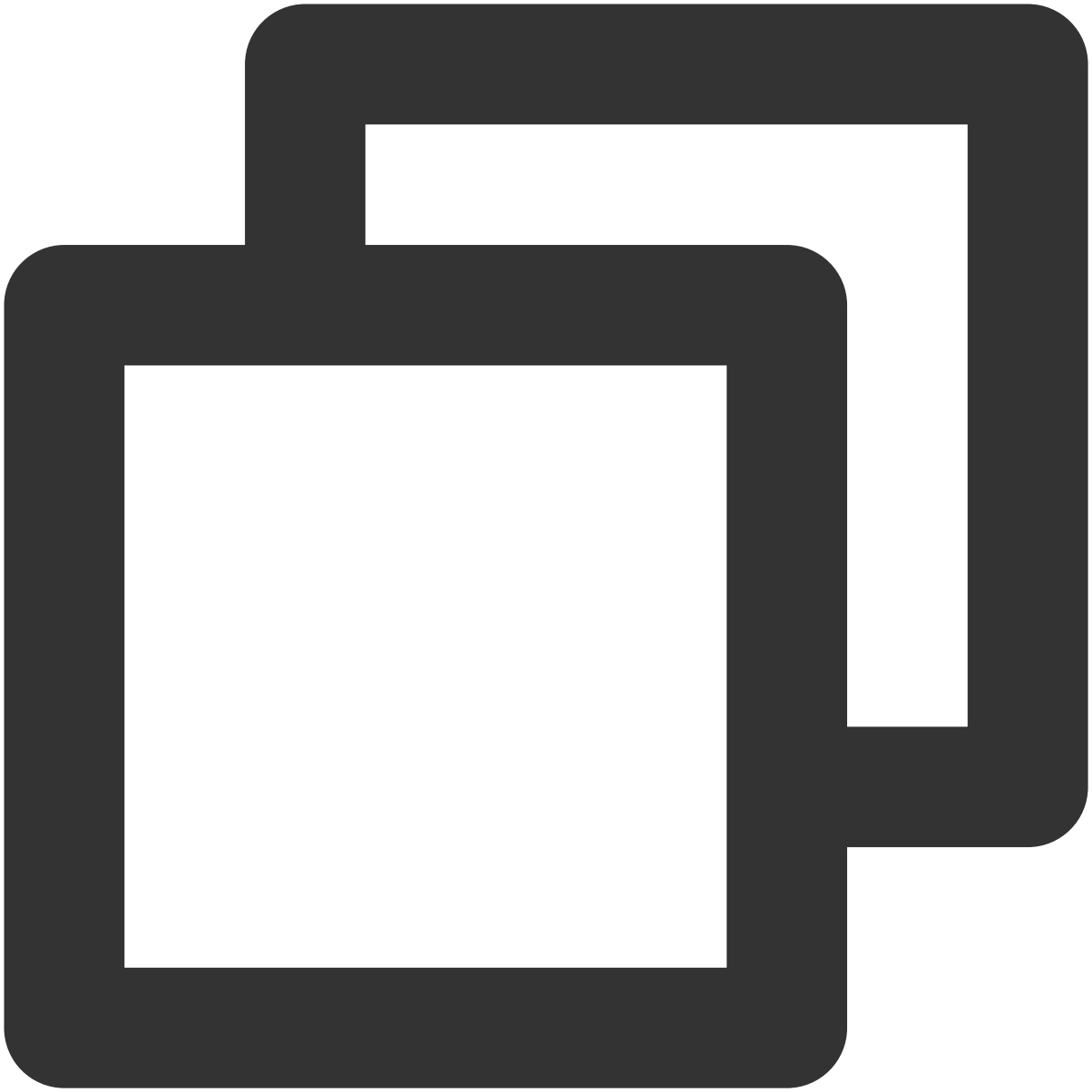


```
sudo gem install cocoapods
```

2. Create Podfile File

Go to the project directory, and enter the following command. A Podfile file will then be created in the project directory.

©2013-2022 Tencent Cloud. All rights reserved.

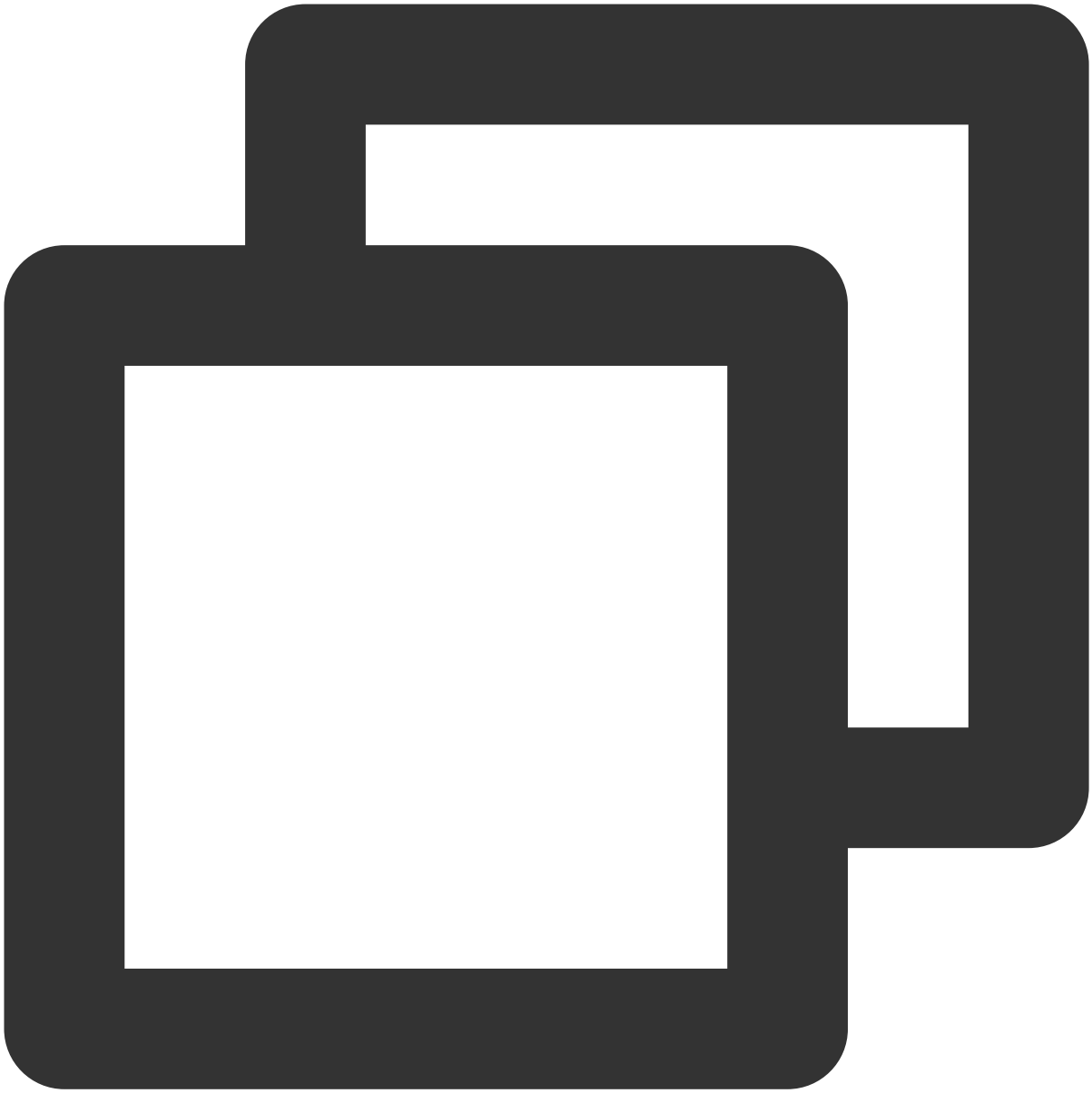


```
pod init
```

3. Edit Podfile File

Choose the appropriate version for your project and edit the Podfile.

©2013-2022 Tencent Cloud. All rights reserved.



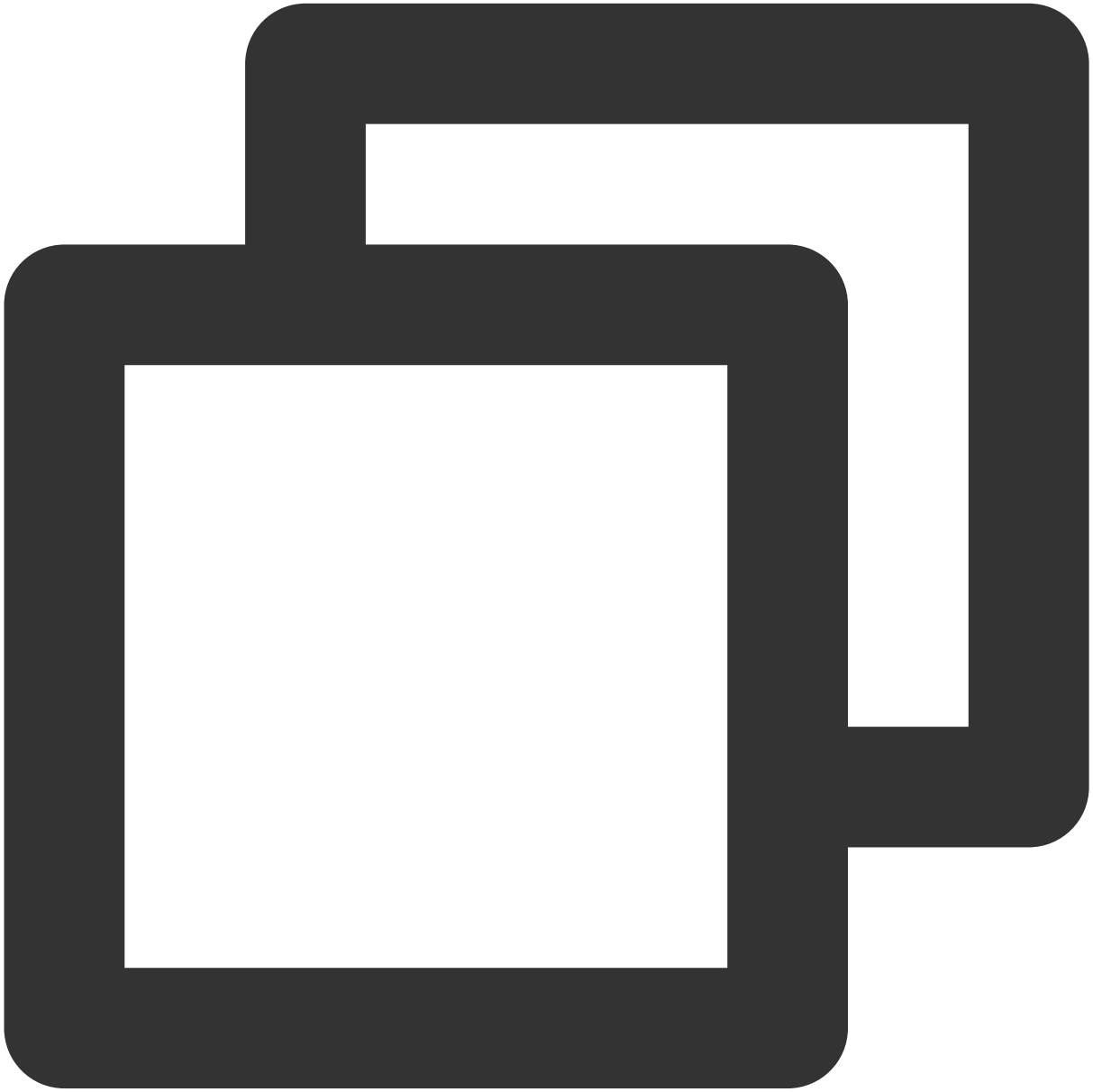
```
platform :ios, '8.0'
target 'App' do

    # TRTC Lite Version
    # The installation package has the minimum incremental size. It only supports t
    pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd

    # Add the IM SDK
    pod 'TXIMSDK_Plus_iOS'
    # pod 'TXIMSDK_Plus_iOS_XCFramework'
    # pod 'TXIMSDK_Plus_Swift_iOS_XCFramework'
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    # If you need to add the Quic plugin, please uncomment the next line.
    # Note: This plugin must be used with the Objective-C edition or XCFramework ed
    # pod 'TXIMSDK_Plus_QuicPlugin'

end
```

4. Update and install the SDK

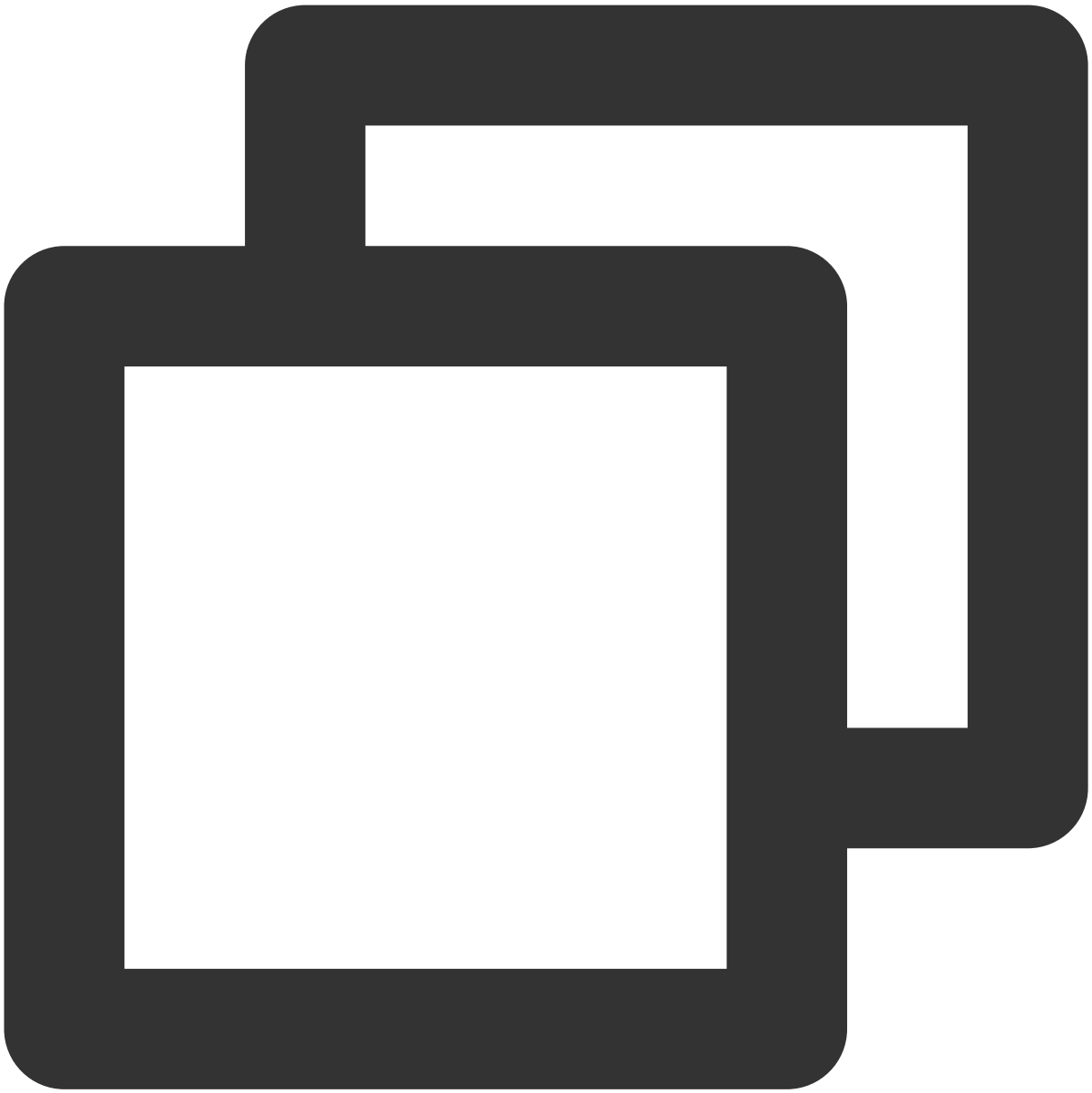
Enter the following command in the terminal window to update the local repository files and install the SDK.



```
pod install
```

©2013-2022 Tencent Cloud. All rights reserved.

Or use the following command to update the local repository.



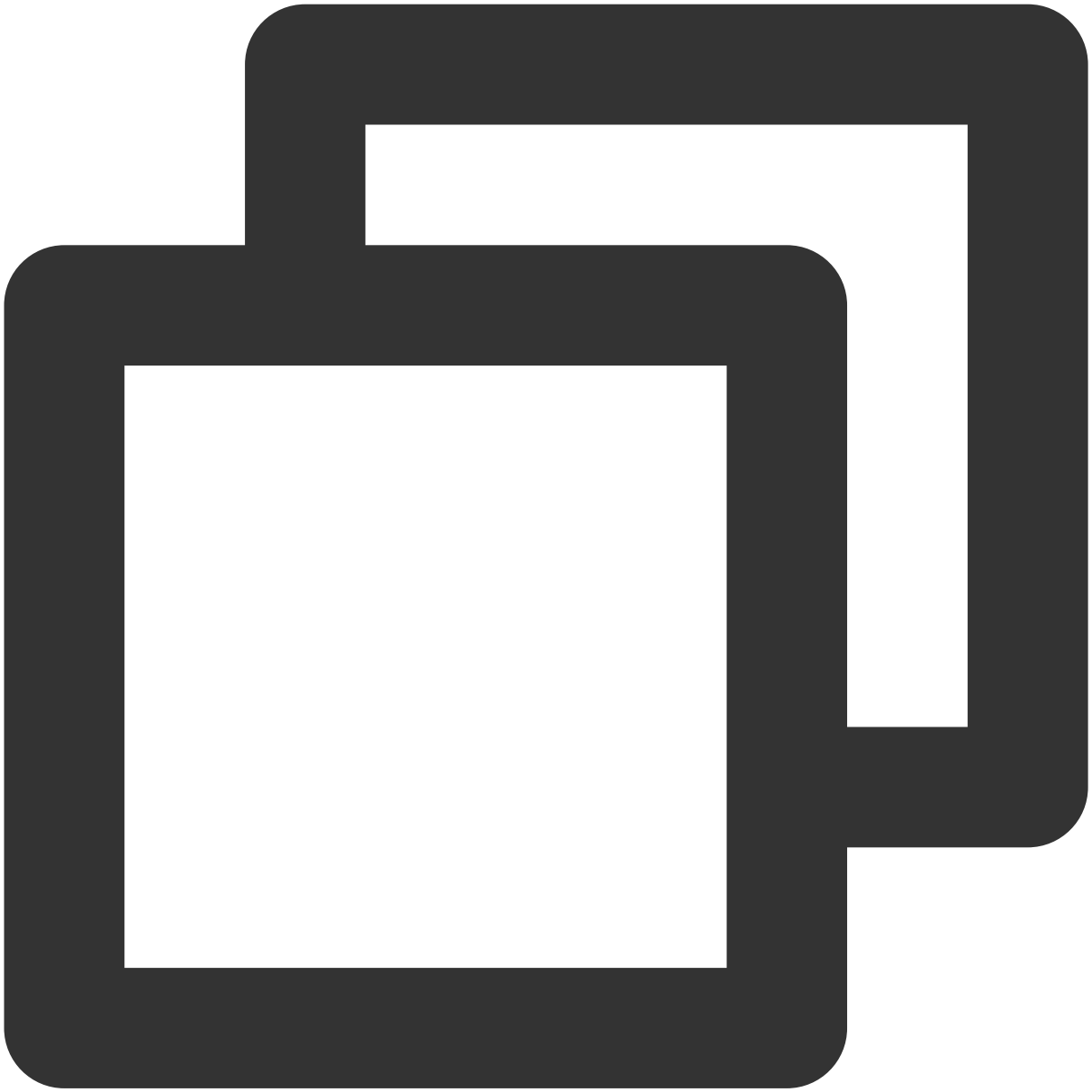
```
pod update
```

Upon the completion of pod command execution, a project file suffixed with .xcworkspace and integrated with the SDK will be generated. Double-click to open it.

Note:

If the pod search fails, it is recommended to try updating the local repo cache of pod. The update command is as follows.

©2013-2022 Tencent Cloud. All rights reserved.



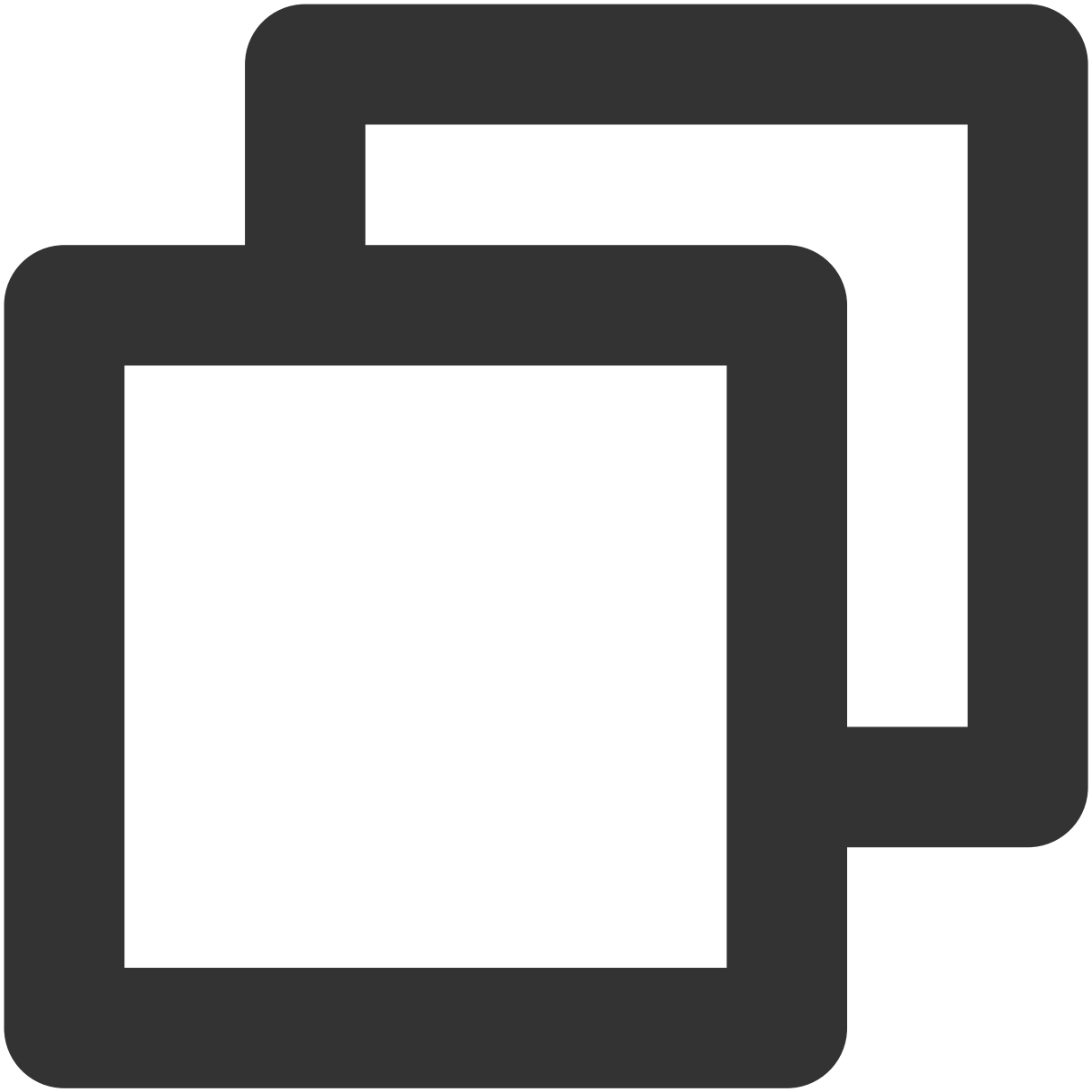
```
pod setup
pod repo update
rm ~/Library/Caches/CocoaPods/search_index.json
```

Besides CocoaPods integration, you can also choose to download the SDK and manually import it. For details, see Manually Integrating the TRTC SDK and Manually Integrating the IM SDK.

Step 3: project configuration

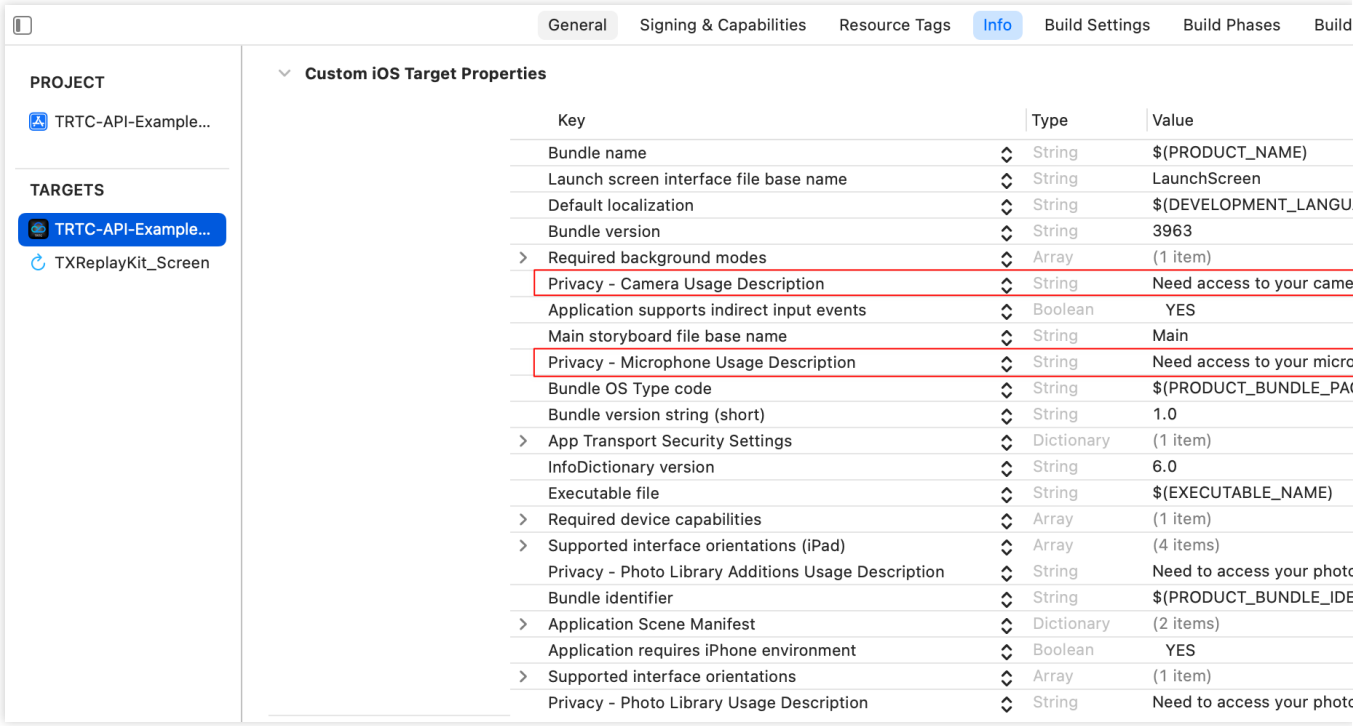
©2013-2022 Tencent Cloud. All rights reserved.

1. In one-to-one audio and video call scenarios, the TRTC SDK and IM SDK need to be authorized for microphone and camera permissions. Add the following content to your app's Info.plist. It corresponds to the system's prompt message in the dialog box when microphone and camera permissions are requested:



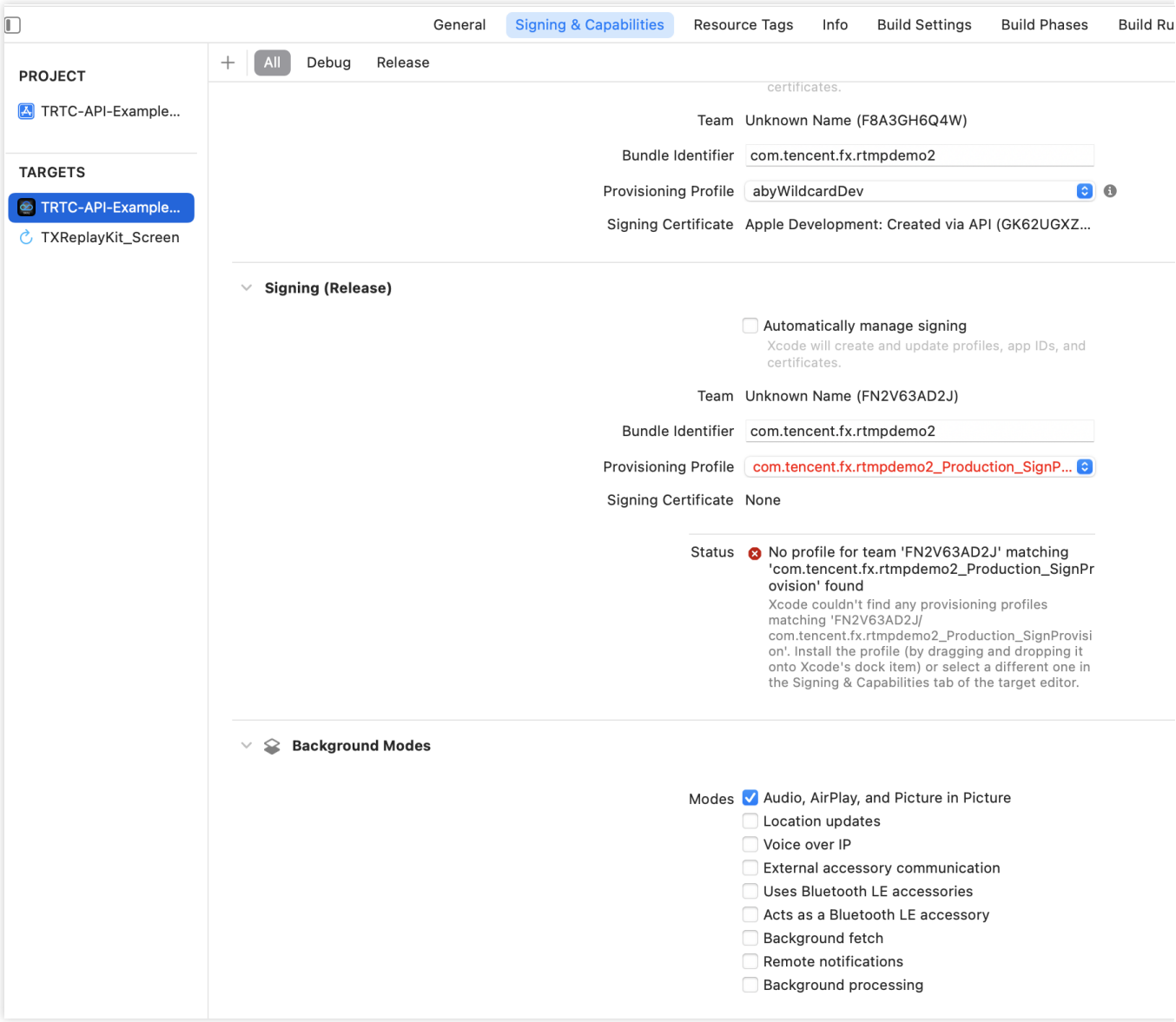
```
Privacy - Microphone Usage Description, and also enter a prompt specifying the purp
Privacy - Camera Usage Description, and enter a prompt specifying the purpose of ca
```

©2013-2022 Tencent Cloud. All rights reserved.



2. If you need your App to continue running certain features in the background, go to XCode, select your current project, and under Capabilities, set the setting for Background Modes to ON, and check Audio, AirPlay, and Picture in Picture, as shown below:

©2013-2022 Tencent Cloud. All rights reserved.



Step 4: authentication credential

UserSig is a security signature designed by the cloud platform to prevent attackers from accessing your cloud account. Cloud services such as Real-Time Communication (TRTC) and Instant Messaging (IM) adopt this security protection mechanism. Authentication is required for TRTC upon entering a room, and for IM during login.

Debugging and testing stage: UserSig can be generated through Client Example Code and Console Access, which are only used for debugging and testing.

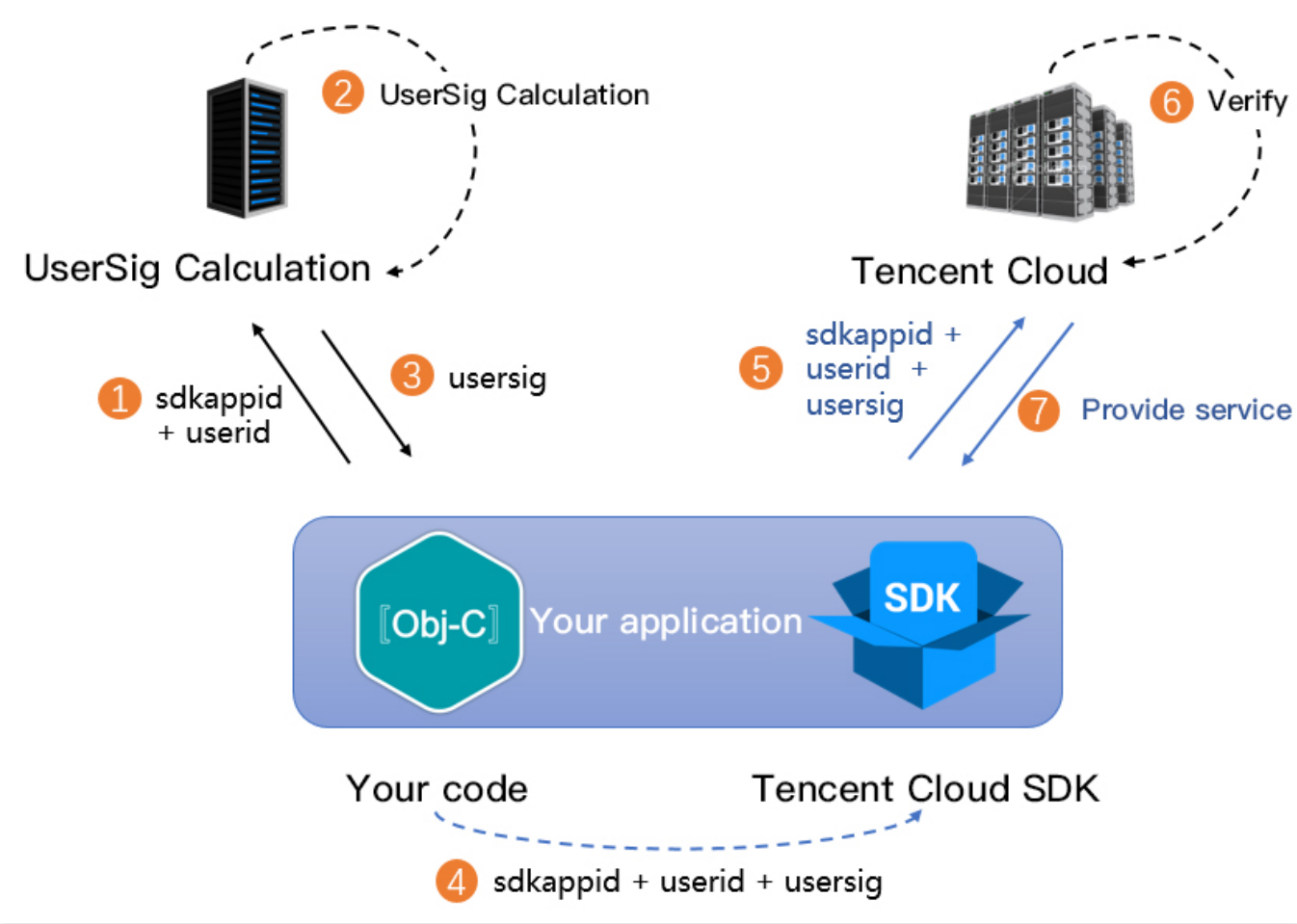
Production stage: It is recommended to use the server computing UserSig solution, which has a higher security level and helps prevent the client from being decompiled and reversed, to avoid the risk of key leakage.

The specific implementation process is as follows:

1. Before calling the initialization API of the SDK, your app must first request UserSig from your server.
2. Your server generates the UserSig based on the SDKAppID and UserID.
3. The server returns the generated UserSig to your app.

©2013-2022 Tencent Cloud. All rights reserved.

4. Your app sends the obtained UserSig to the SDK through a specific API.
5. The SDK submits the SDKAppID + UserID + UserSig to the cloud server for verification.
6. The cloud platform verifies the validity of the UserSig.
7. Once the verification is passed, it will provide instant communication services to the IM SDK and real-time audio and video services to the TRTC SDK.



Note:

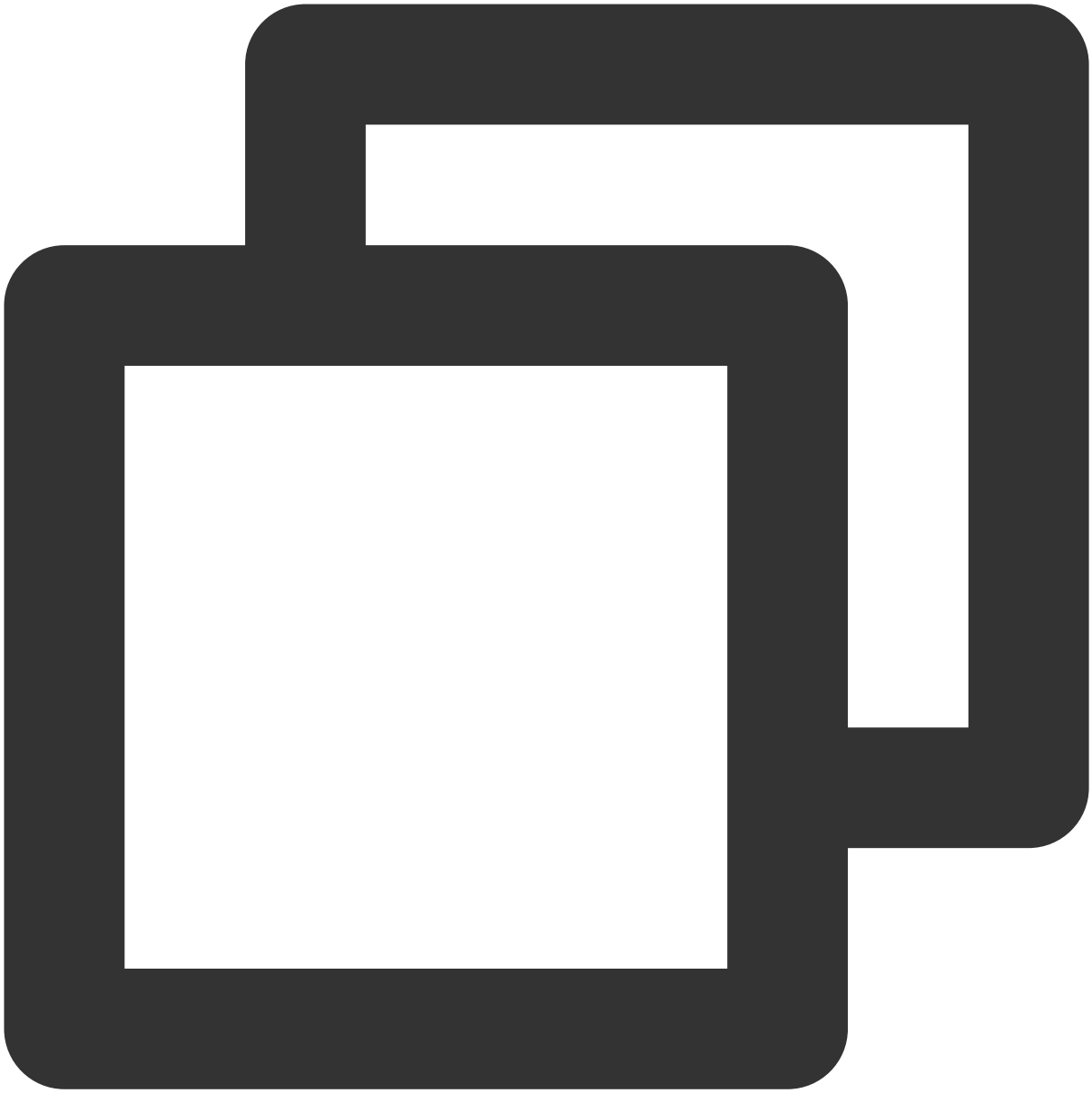
The method of generating UserSig locally during the debugging and testing stage is not recommended for the online environment because it may be easily decompiled and reversed, causing key leakage.

We provide server computation source code for UserSig in multiple programming languages (Java/GO/PHP/Nodejs/Python/C#/C++). For details, see Server Computation of UserSig.

Step 5: initialize the SDK

1. Initialize IM SDK and Add Event Listeners

©2013-2022 Tencent Cloud. All rights reserved.



```
// Obtain the SDKAppID from the Instant Messaging (IM) console.
// Add a V2TIMSDKListener event listener. The self is the implementation class of i
[[V2TIMManager sharedInstance] addIMSDKListener:self];
// Initialize the IM SDK. After calling this API, you can immediately call the log-
[[V2TIMManager sharedInstance] initSDK:sdkAppID config:config];

// After the SDK is initialized, it will trigger various events, such as connection
- (void)onConnecting {
    NSLog(@"The IM SDK is connecting to the Cloud Virtual Machine");
}
```

©2013-2022 Tencent Cloud. All rights reserved.

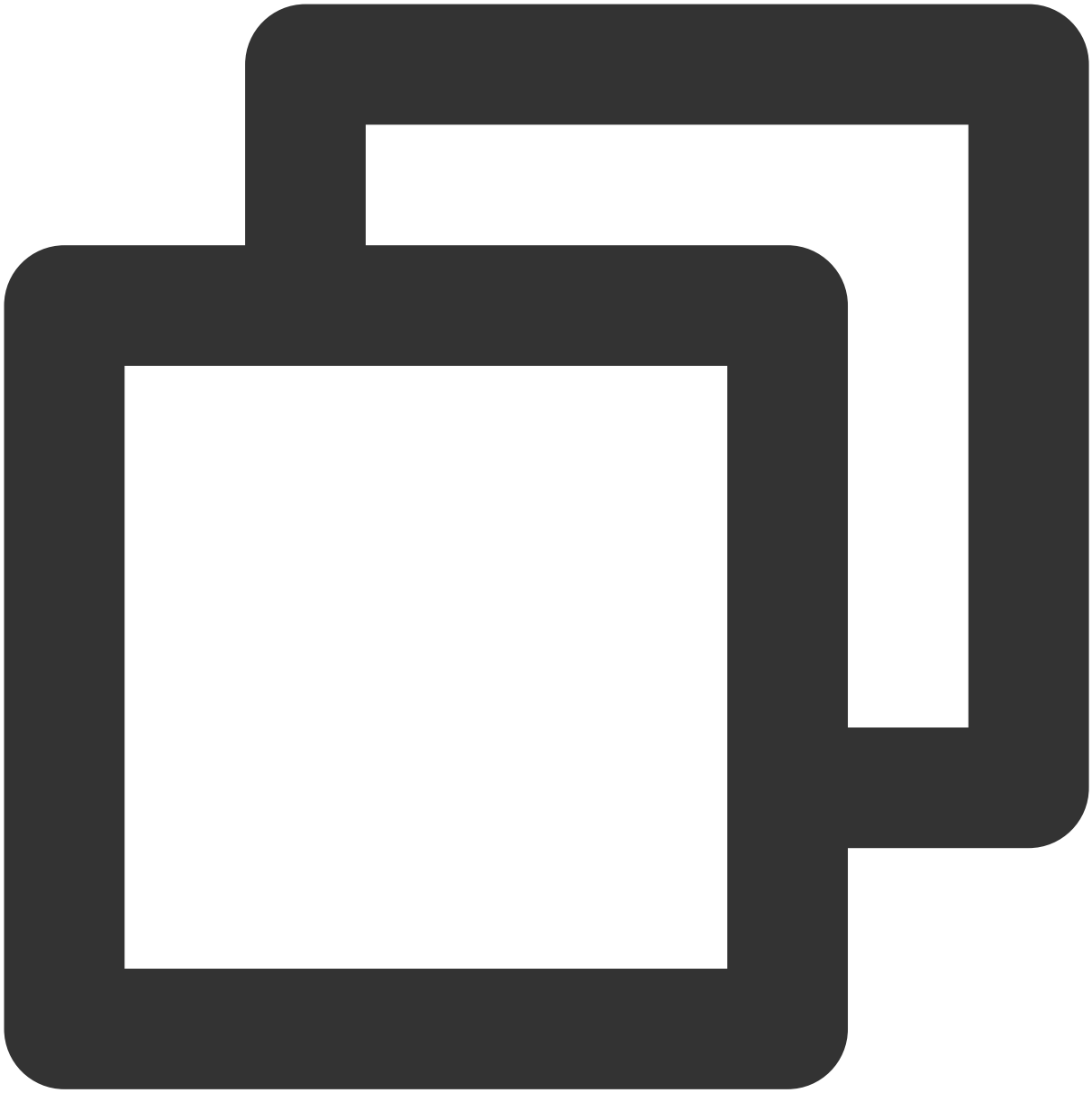
```
- (void)onConnectSuccess {
    NSLog(@"The IM SDK has successfully connected to the Cloud Virtual Machine");
}

// Remove event listener
// The self is the implementation class of id<V2TIMSDKListener>
[[V2TIMManager sharedInstance] removeIMSDKListener:self];
// Deinitialize the SDK
[[V2TIMManager sharedInstance] unInitSDK];
```

Note:

If the lifecycle of your application is consistent with the SDK lifecycle, you do not need to deinitialize before exiting the application. However, if you only initialize the SDK when entering a specific interface and no longer use it after exiting that interface, you can deinitialize the SDK.

2. Create TRTC SDK Instances and Set Event Listeners

©2013-2022 Tencent Cloud. All rights reserved.



```
// Create TRTC SDK instance (single instance pattern)
self.trtcCloud = [TRTCCloud sharedInstance];
// Set event listeners
self.trtcCloud.delegate = self;

// Notifications from various SDK events (e.g., error codes, warning codes, audio a
- (void)onError:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg extInfo:(
    NSLog(@"%d: %@", errCode, errMsg);
}

- (void)onWarning:(TXLiteAVWarning)warningCode warningMsg:(nullable NSString *)warn
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    NSLog(@"%d: %@", warningCode, warningMsg);
}

// Remove event listener
self.trtcCloud.delegate = nil;
// Destroy TRTC SDK instance (single instance pattern)
[TRTCCloud destroySharedIntance];
```

Note:

It is recommended to listen to SDK event notifications. Perform log printing and handling for some common errors. For details, see Error Code Table.

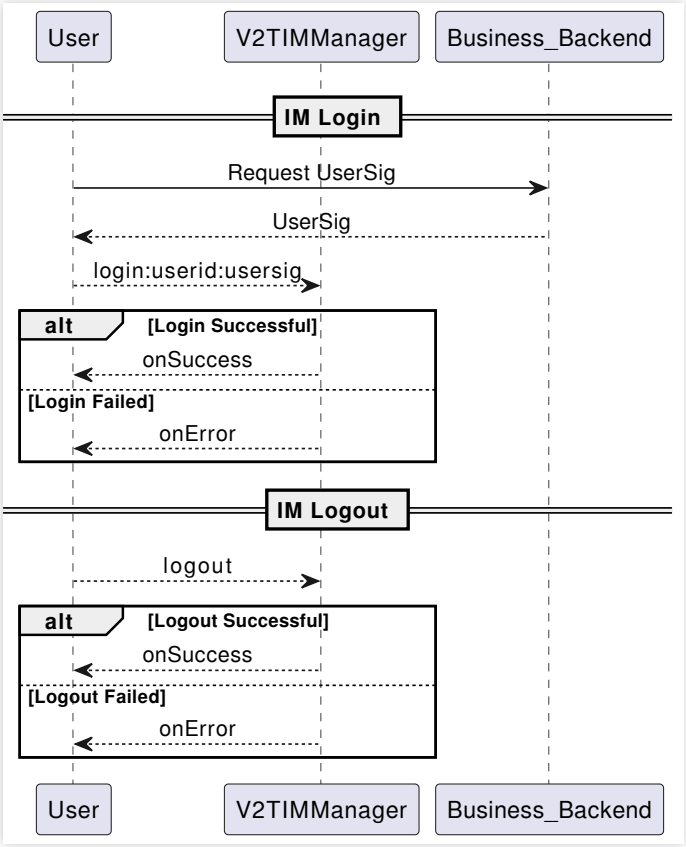
Integration Process

Step 1: log in

After the IM SDK is initialized, you need to call the SDK log in API to authenticate your account identity and gain permissions to use features. **Before using any other features, ensure you are successfully logged in**, or you might encounter feature malfunctions or unavailability. If you only need to use TRTC's audio and video services, you can skip this step.

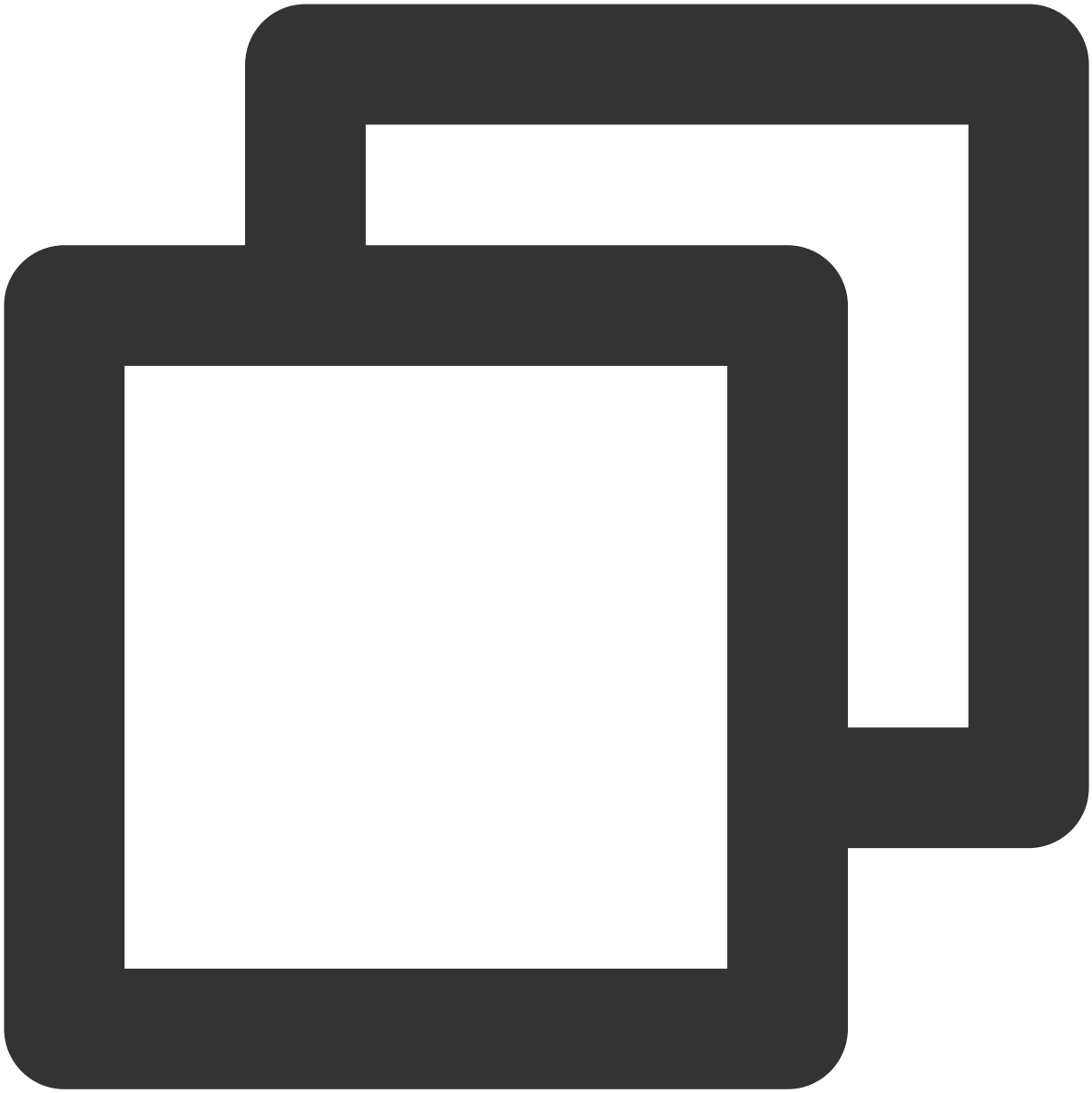
Sequence Diagram

©2013-2022 Tencent Cloud. All rights reserved.



Log in operation

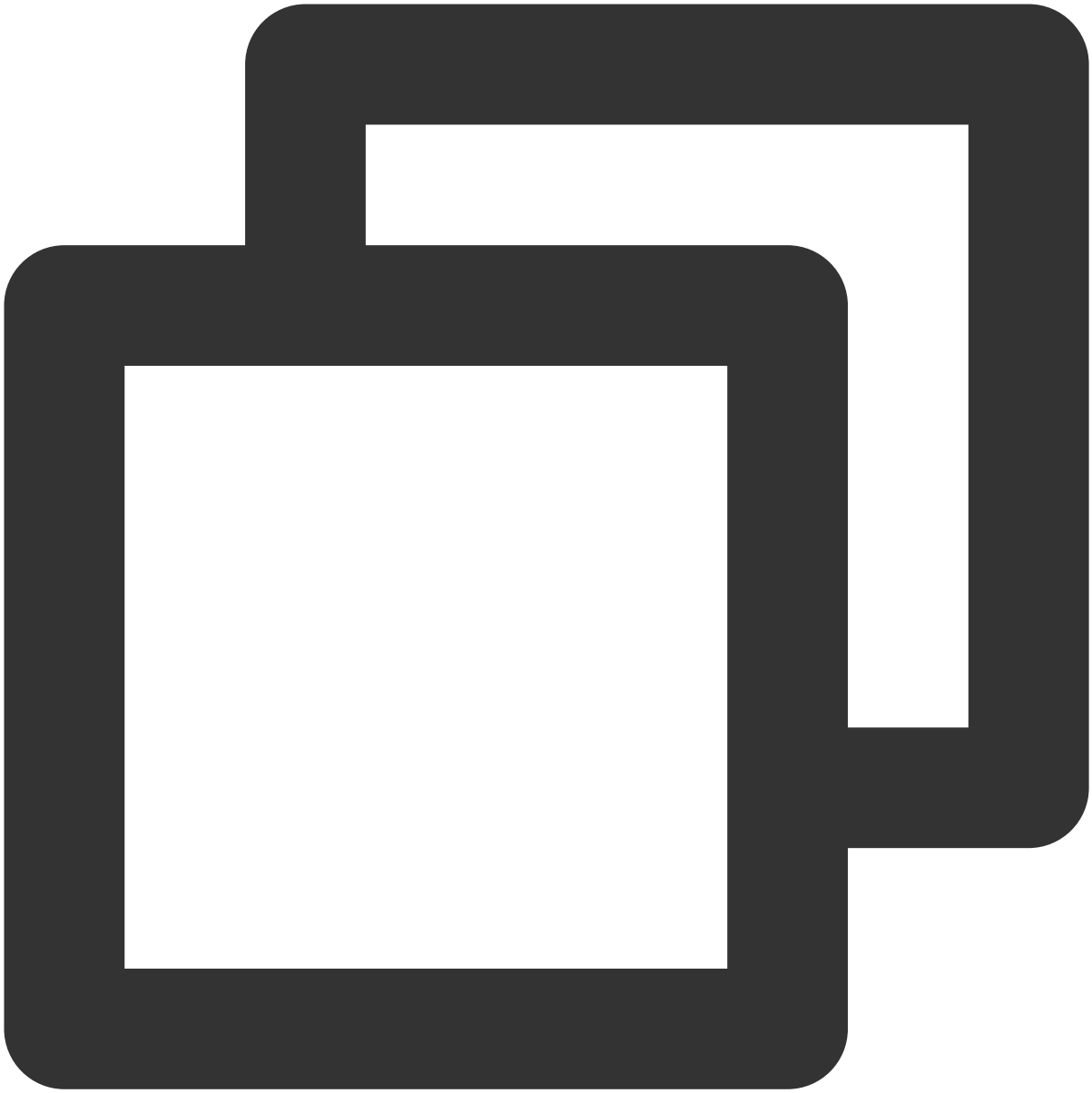
©2013-2022 Tencent Cloud. All rights reserved.



```
// Log in: userID can be defined by the user and userSig can be generated as per st
[[V2TIMManager sharedInstance] login:userID userSig:userSig succ:^{
    NSLog(@"success");
} fail:^(int code, NSString *desc) {
    // The following error codes indicate an expired userSig, and you need to gener
    // 1. ERR_USER_SIG_EXPIRED(6206).
    // 2. ERR_SVR_ACCOUNT_USERSIG_EXPIRED(70001).
    // Note: Do not call the log-in API in case of other error codes. Otherwise, th
    NSLog(@"failure, code:%d, desc:%@", code, desc);
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

Log out operation



```
// Log out
[[V2TIMManager sharedInstance] logout:^{
    NSLog(@"success");
} fail:^(int code, NSString *desc) {
    NSLog(@"failure, code:%d, desc:%@", code, desc);
}];
```

Note:

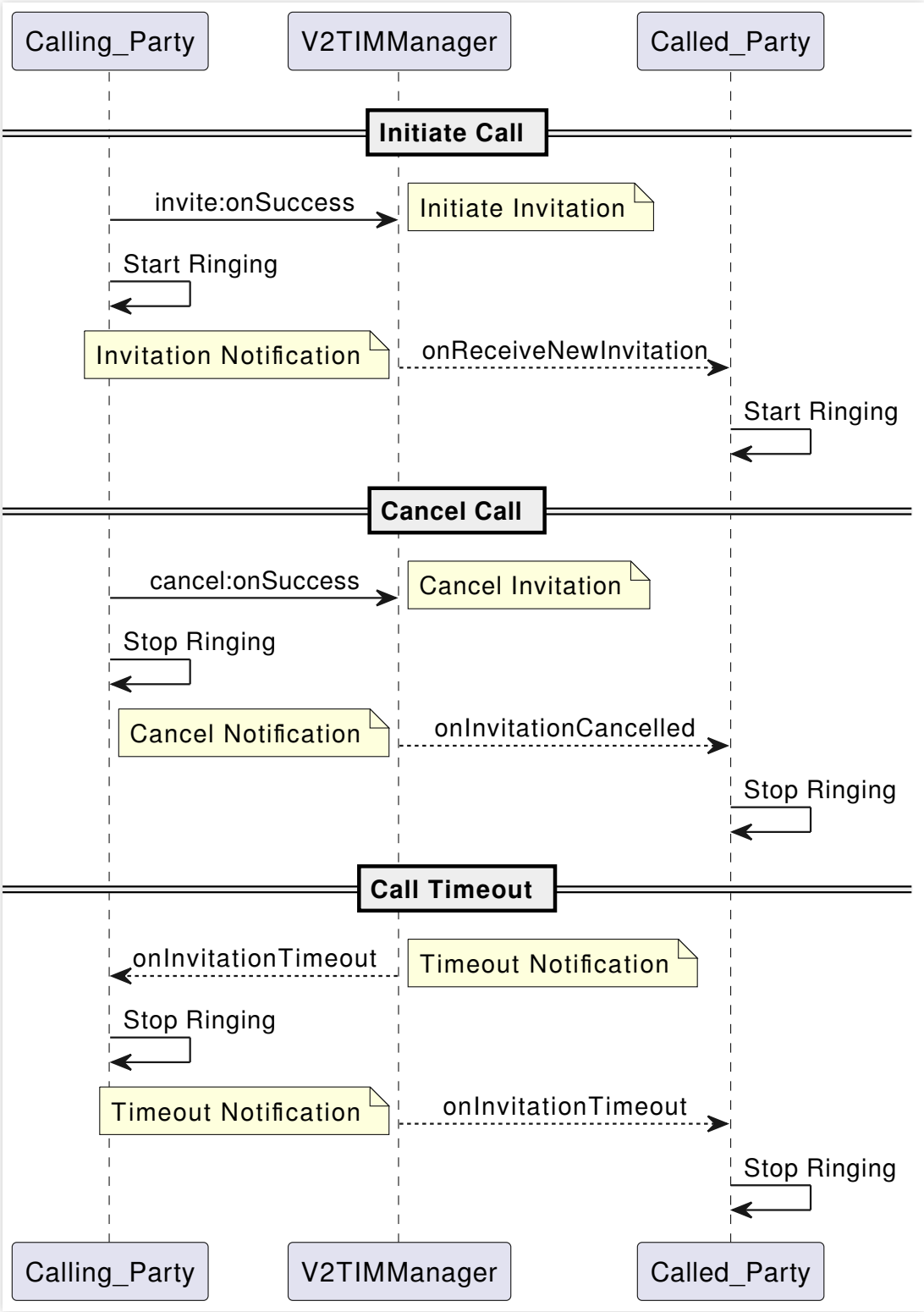
©2013-2022 Tencent Cloud. All rights reserved.

If the lifecycle of your application matches that of the IM SDK, you do not need to log out before exiting the application. However, if you only use the IM SDK after entering specific interfaces and no longer use it after exiting those interfaces, you can log out and deinitialize the IM SDK.

Step 2: call

Sequence Diagram

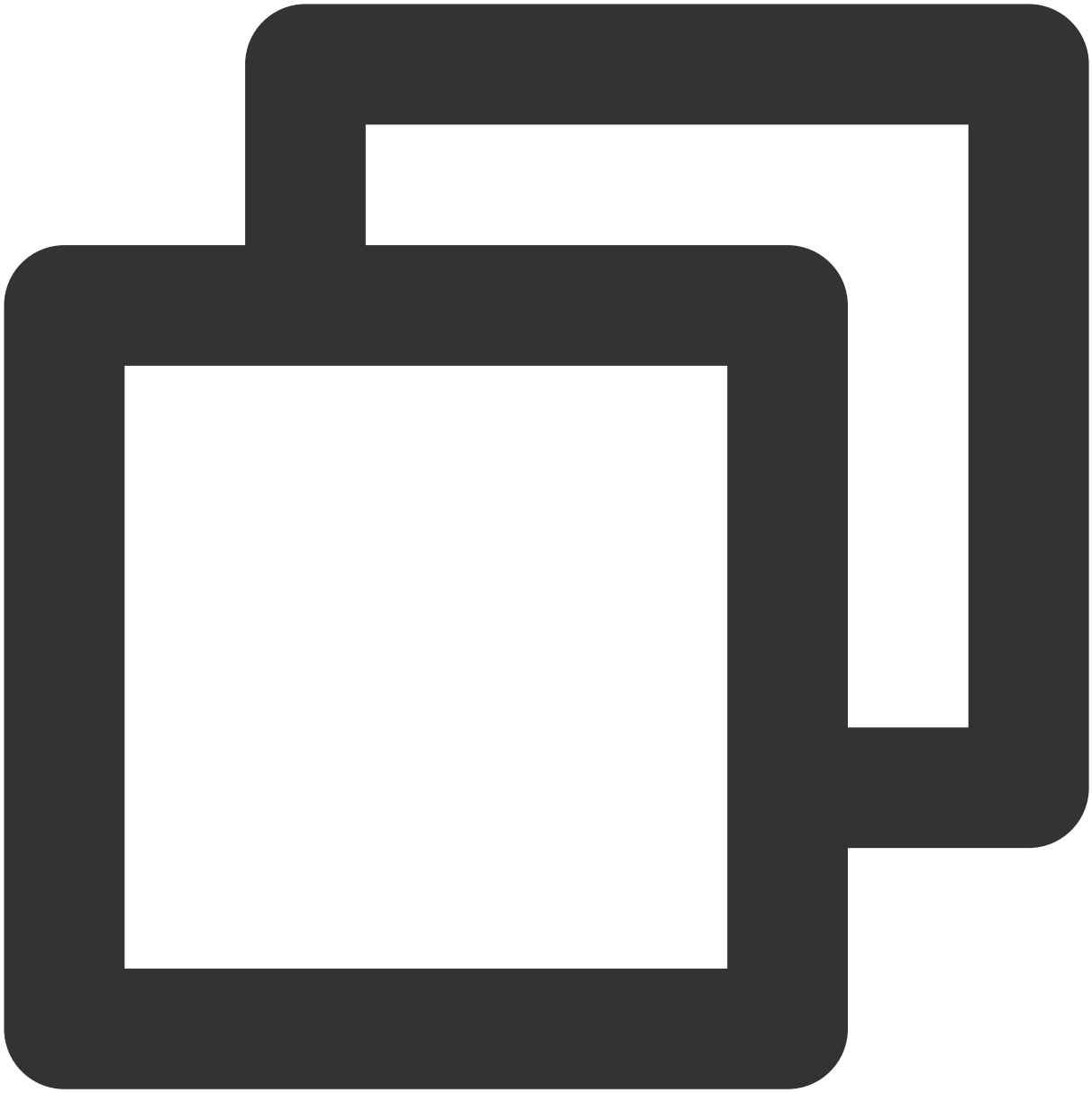
©2013-2022 Tencent Cloud. All rights reserved.



Initiate Call

1. Caller's local video preview (only for video calls; ignore this step for audio calls)

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)setupTRTC {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
    encParam.videoBitrate = 850;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // Enable local camera preview (you can specify to use the front/rear camera fo
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.previewView];
```

©2013-2022 Tencent Cloud. All rights reserved.

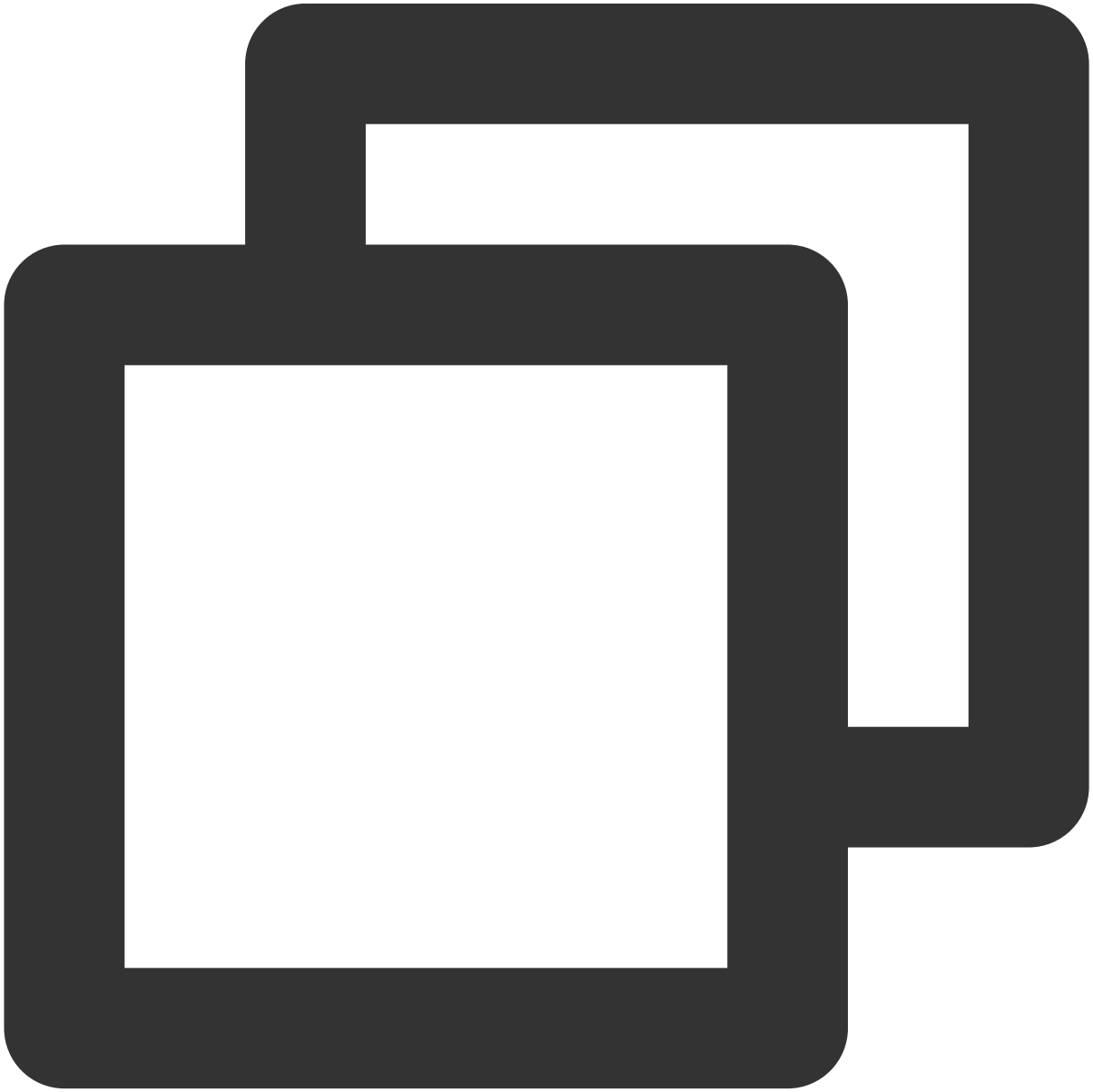
```
}
```

Note:

You can set the video encoding parameters TRTCVideoEncParam according to business needs. For the best combinations of resolutions and bitrates for each tier, see Resolution and Bitrate Reference Table.

Call the above APIs before enterRoom. The SDK will only enable the camera preview and will wait until you call enterRoom to start local video streaming.

2. Caller sends call invitation signaling



```
// Construct custom data
```

©2013-2022 Tencent Cloud. All rights reserved.

```
NSDictionary *dic = @{
    @"cmd": @"av_call",
    @"msg": @{
        // Specify the call type (video call, audio call)
        @"callType": @"videoCall",
        // Specify the TRTC room ID (caller can generate it randomly)
        @"roomId": @"xxxRoomId",
    },
};
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dic
                                                   options:NSJSONWritingPrettyPrint
                                                     error:nil];
if (jsonData) {
    NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8S
    // Send call invitation signaling
    [[V2TIMManager sharedInstance] invite:self.receiverId data:jsonString onlineUse
        // Successfully send call invitation signaling
        // Render call page, play call ringtone
    } fail:^(int code, NSString *desc) {
        // Failed to send call invitation signaling
        // Prompt call failure, you can try to retry
    }];
}
```

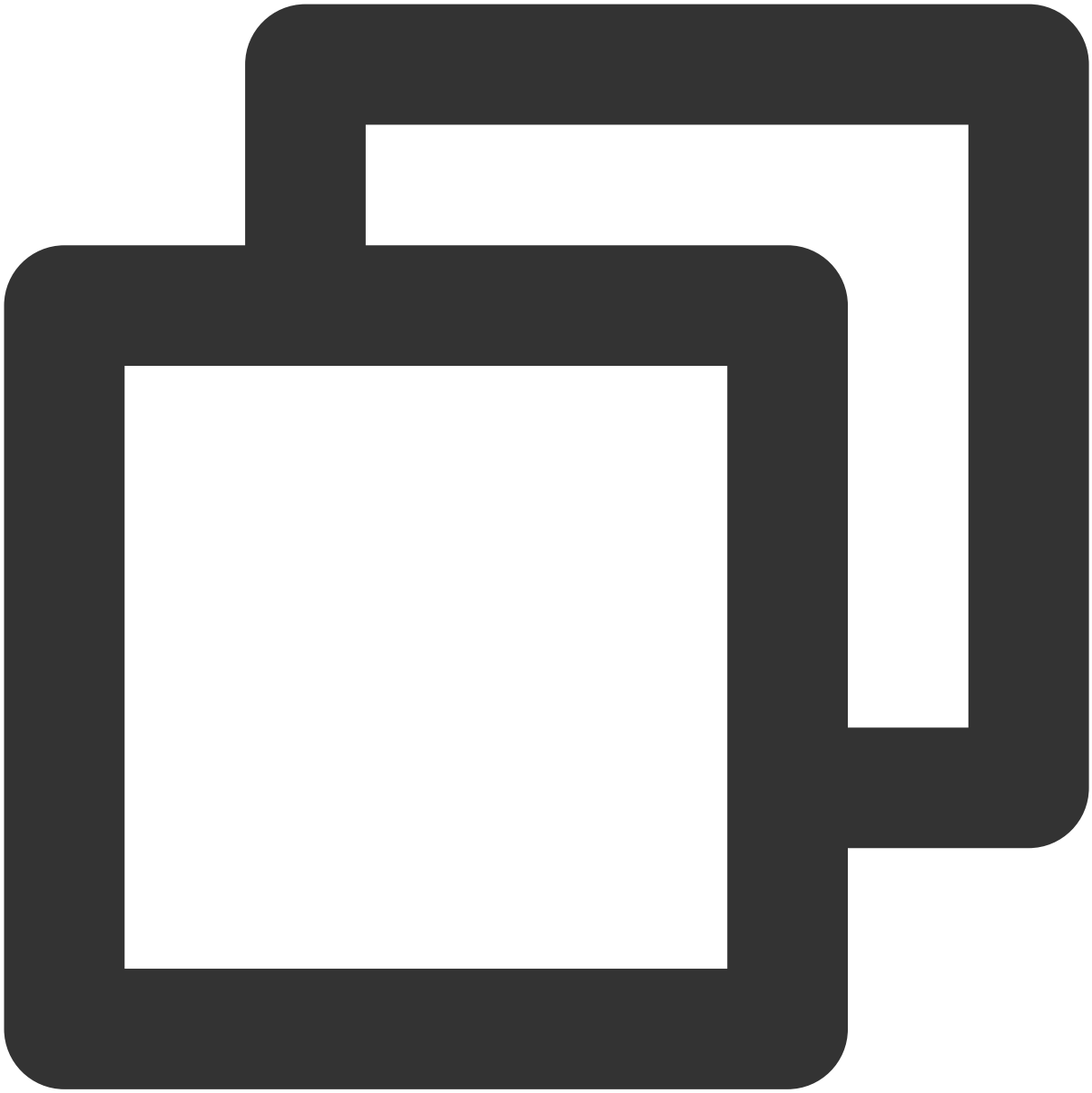
Note:

In audio and video call scenarios, it is usually necessary to configure offline push information `offlinePushInfo` in the invitation signaling. For more details, see Offline Push Message.

It is recommended to set a reasonable timeout parameter `timeout` in the invitation signaling, in seconds. The SDK will perform timeout detection to realize auto hang up after call timeout.

3. Callee receives the call invitation notification

©2013-2022 Tencent Cloud. All rights reserved.



```
[[V2TIMManager sharedInstance] addSignalingListener:self];

#pragma mark - V2TIMSignalingListener

// Callee receives the call request. The inviteID is the request ID, and inviter is
- (void)onReceiveNewInvitation:(NSString *)inviteID inviter:(NSString *)inviter gro
    if (data && ![data isEqualToString:@""]) {
        NSData *jsonData = [data dataUsingEncoding:NSUTF8StringEncoding];
        NSDictionary *dictionary = [NSJSONSerialization JSONObjectWithData:jsonData
                                                                   options:NSJSONRe
                                                                     error:nil];
```

©2013-2022 Tencent Cloud. All rights reserved.

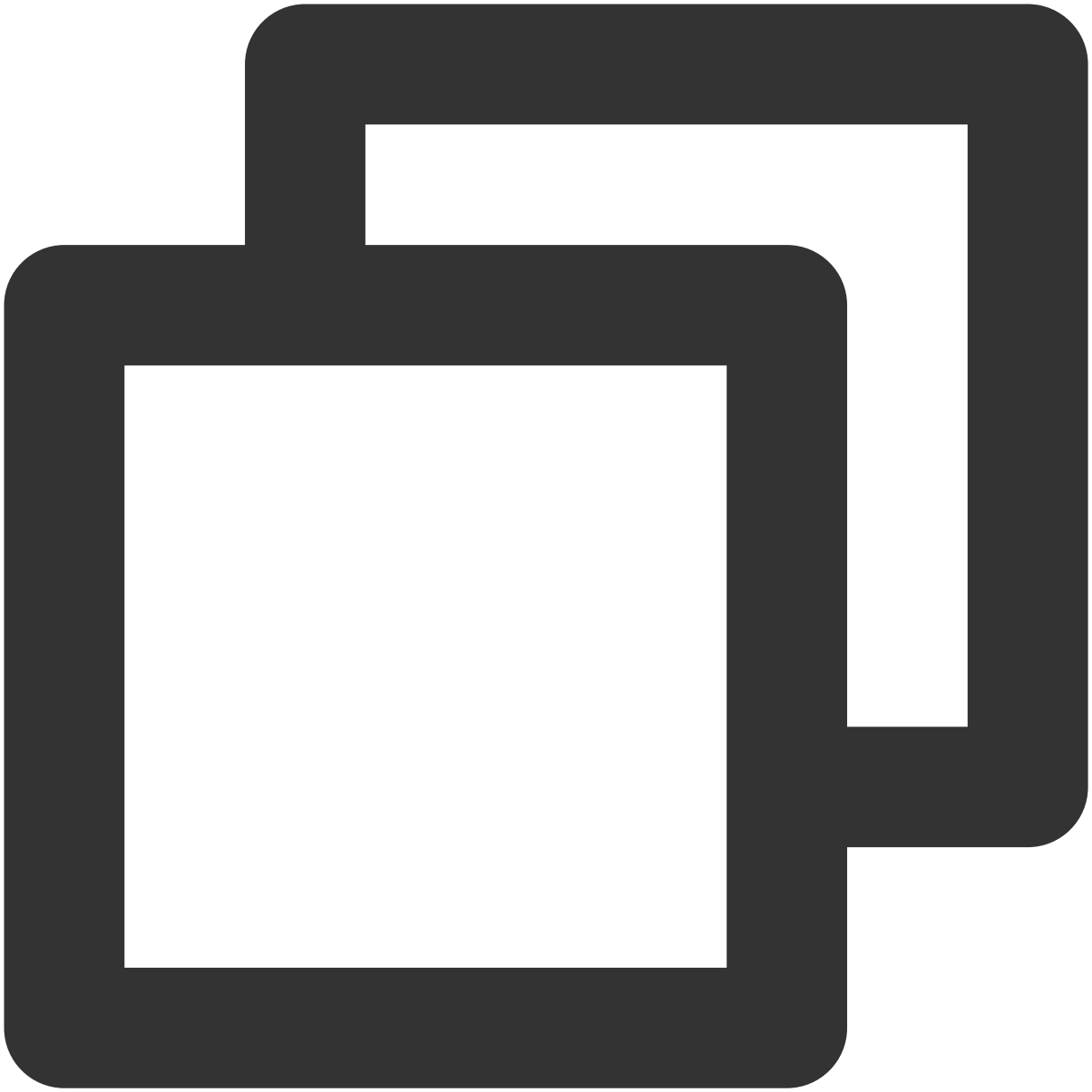
```
        if (dictionary) {
            NSString *command = dictionary[@"cmd"];
            NSDictionary *msg = dictionary[@"msg"];
            if ([command isEqualToString:@"av_call"]) {
                NSString *callType = msg[@"callType"];
                NSString *roomId = msg[@"roomId"];

                // Render call page, play call ringtone
            }
        }
    }
}
```

Note:

Caller initiates a call request. When the callee receives the call request, the business side needs to implement the rendering of the call page and the playing of the call ringtone on its own.

4. Callee's local video preview (only for video calls; ignore this step for audio calls)

©2013-2022 Tencent Cloud. All rights reserved.



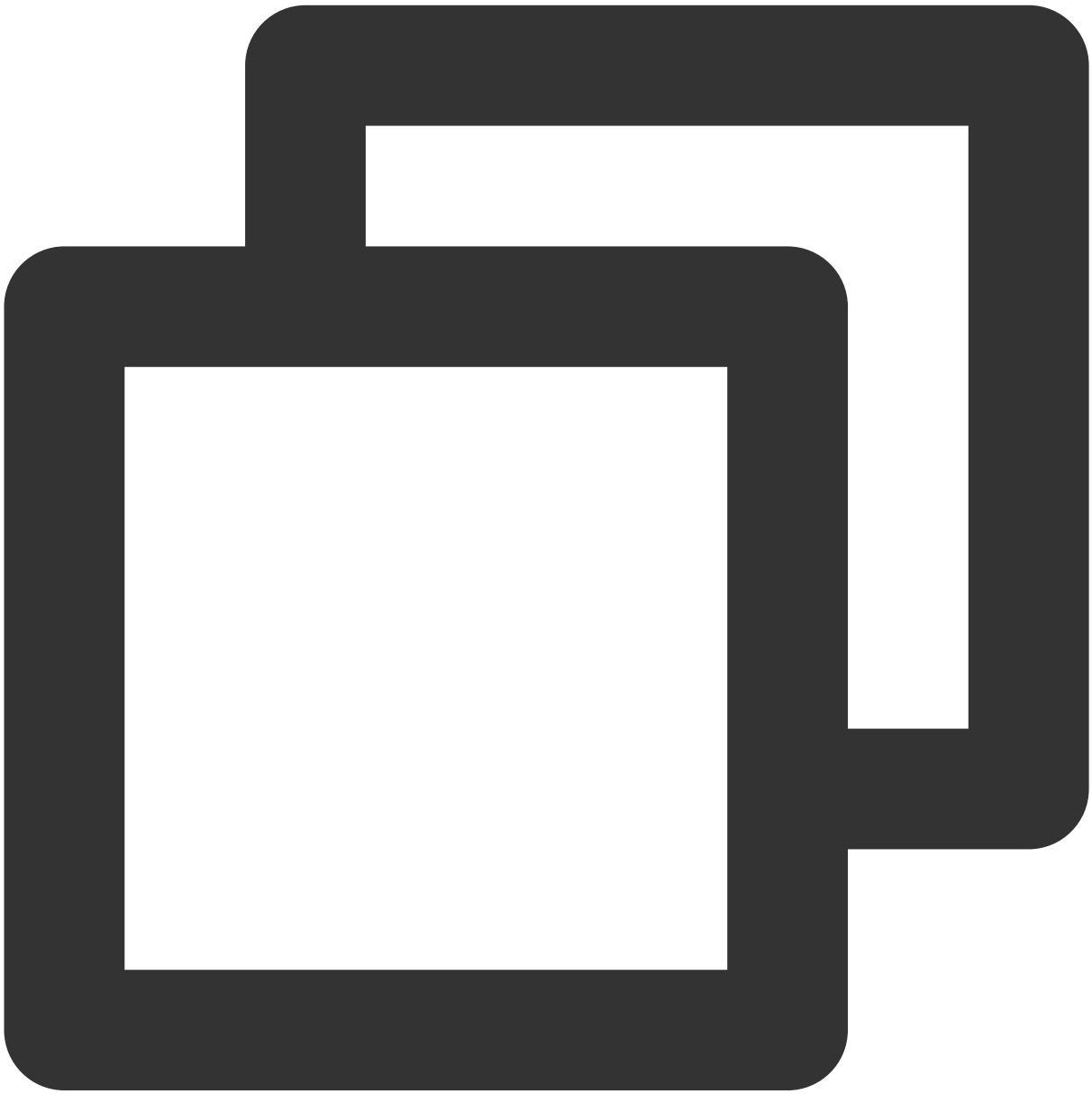
```
if ([callType isEqualToString:@"videoCall"]) {
    // Set video encoding parameters to determine the picture quality seen by remot
    TRTCVideoEncParam *encParam = [[TRTCVideoEncParam alloc] init];
    encParam.videoResolution = TRTCVideoResolution_960_540;
    encParam.videoFps = 15;
    encParam.videoBitrate = 850;
    encParam.resMode = TRTCVideoResolutionModePortrait;
    [self.trtcCloud setVideoEncoderParam:encParam];

    // Enable local camera preview (you can specify to use the front/rear camera fo
    [self.trtcCloud startLocalPreview:self.isFrontCamera view:self.previewView];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

Cancel Call

1. Caller cancels the call request

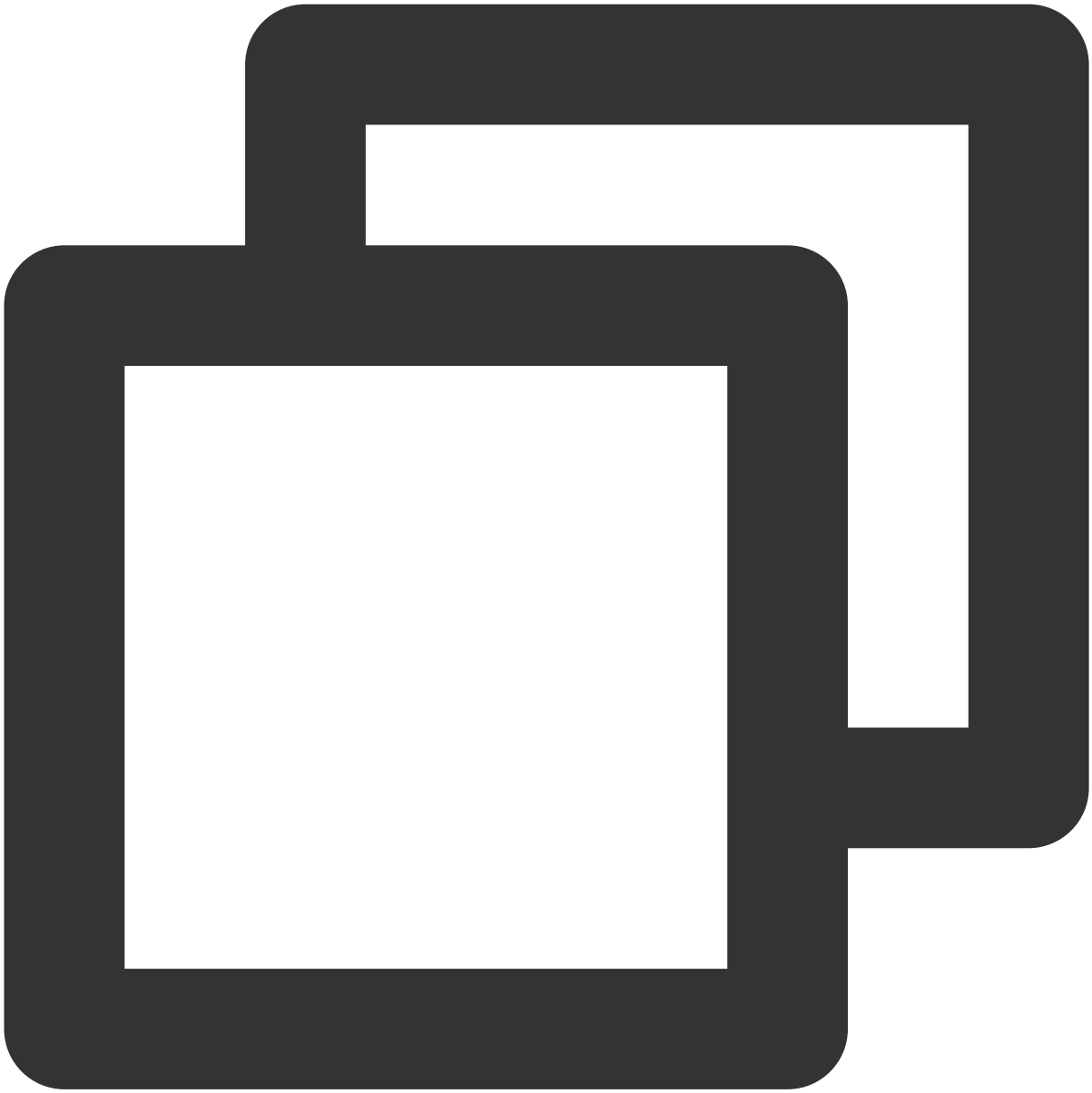


```
[[V2TIMManager sharedInstance] cancel:inviteId data:data succ:^{
    // Successfully cancel the call request
    // Terminate the call page, and stop the call ringtone
} fail:^(int code, NSString *desc) {
    // Failed to cancel call request
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // Prompt cancel failure, and you can retry
}];
```

2. Callee receives the cancellation notification

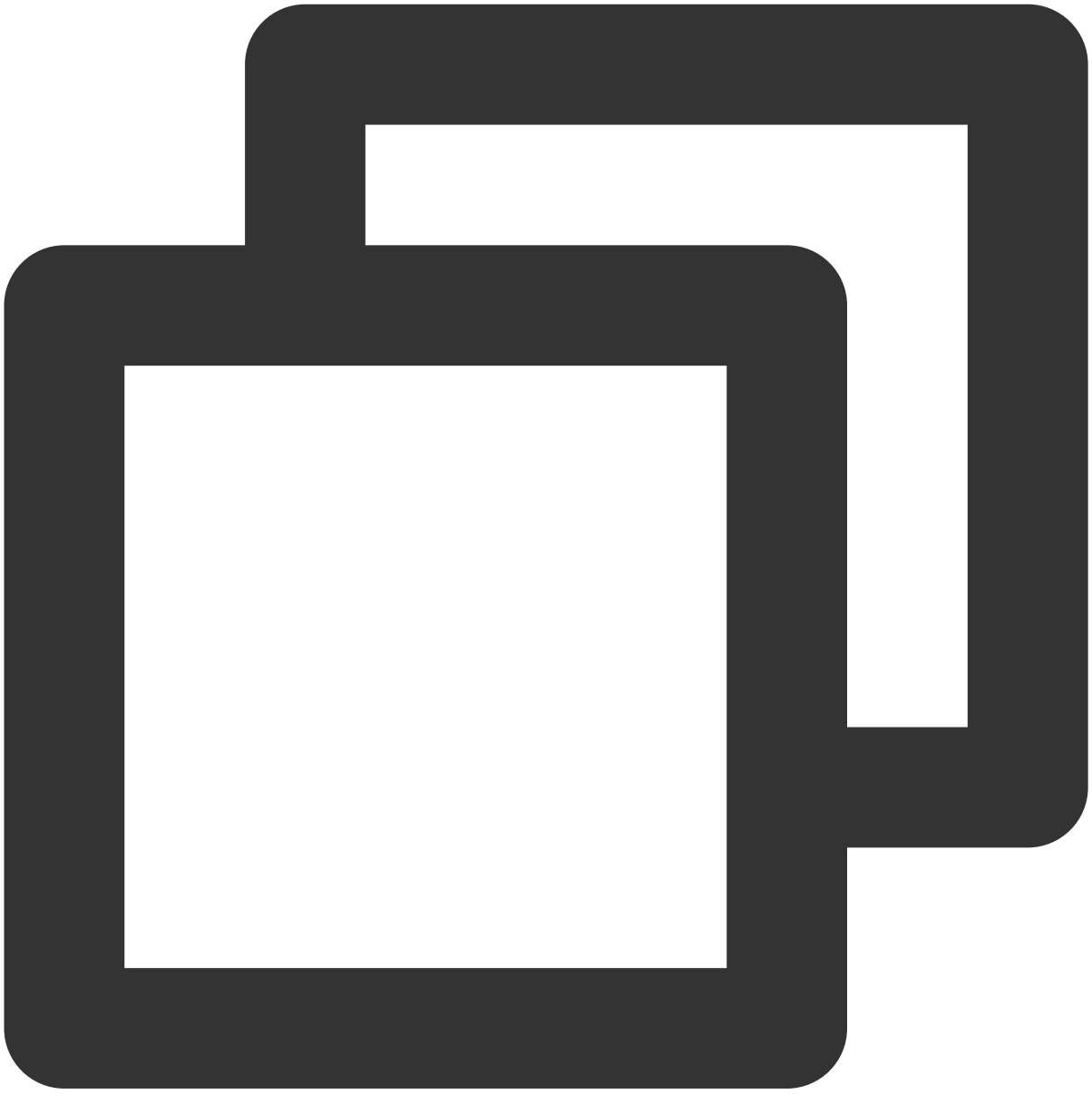


```
#pragma mark - V2TIMSignalingListener

- (void)onInvitationCancelled:(NSString *)inviteID inviter:(NSString *)inviter data
    // Terminate the call page, and stop the call ringtone
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Call Timeout

Both caller and callee will receive a timeout notification. They also terminate the call page, and stop the call ringtone.



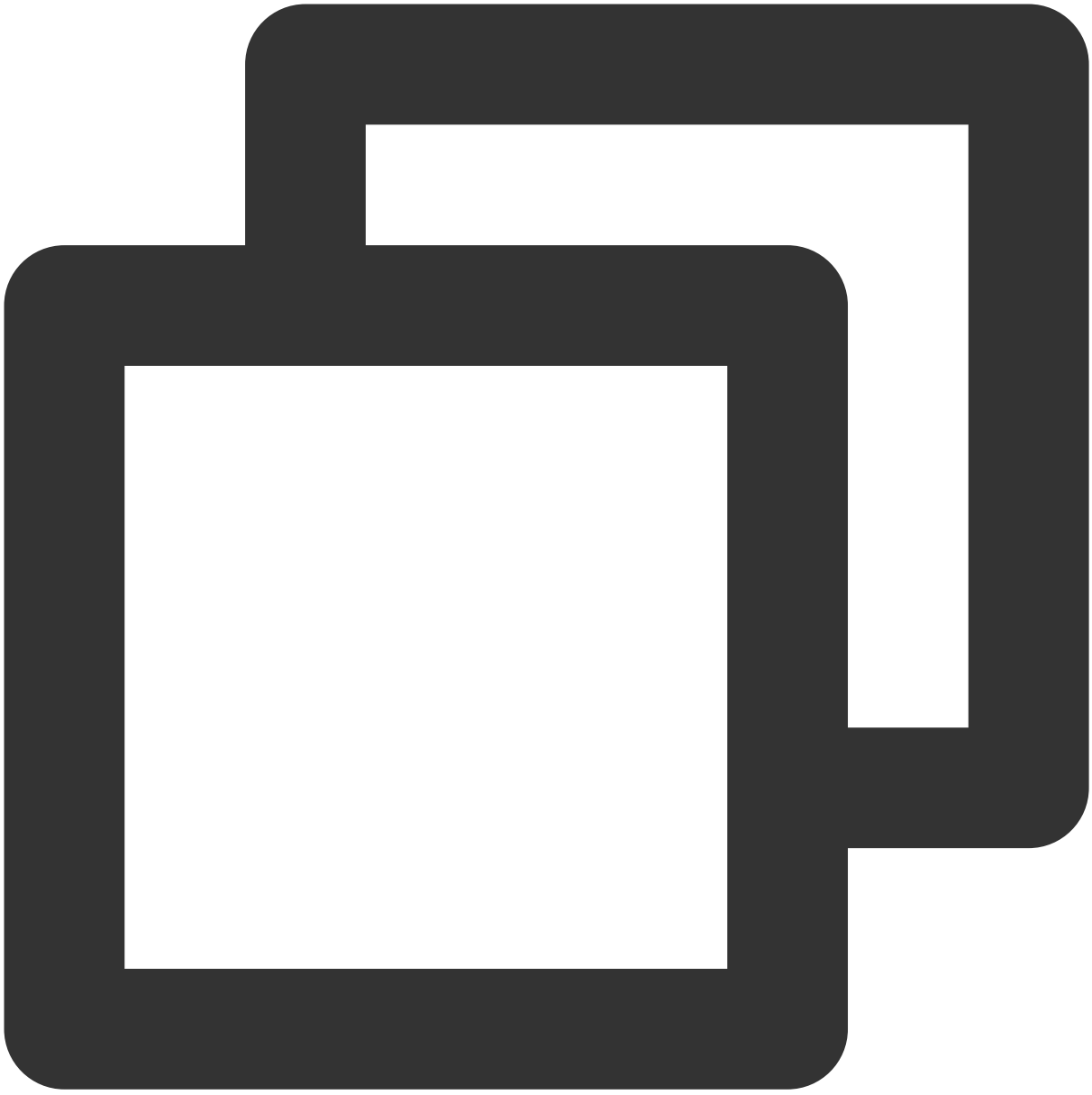
```
#pragma mark - V2TIMSignalingListener

- (void)onInvitationTimeout:(NSString *)inviteID inviteeList:(NSArray<NSString *> *
    // Prompt call timeout. Terminate the call page, and stop the call ringtone
}
```

Step 3: answer

©2013-2022 Tencent Cloud. All rights reserved.

Answer signaling

1. Callee sends answer signaling

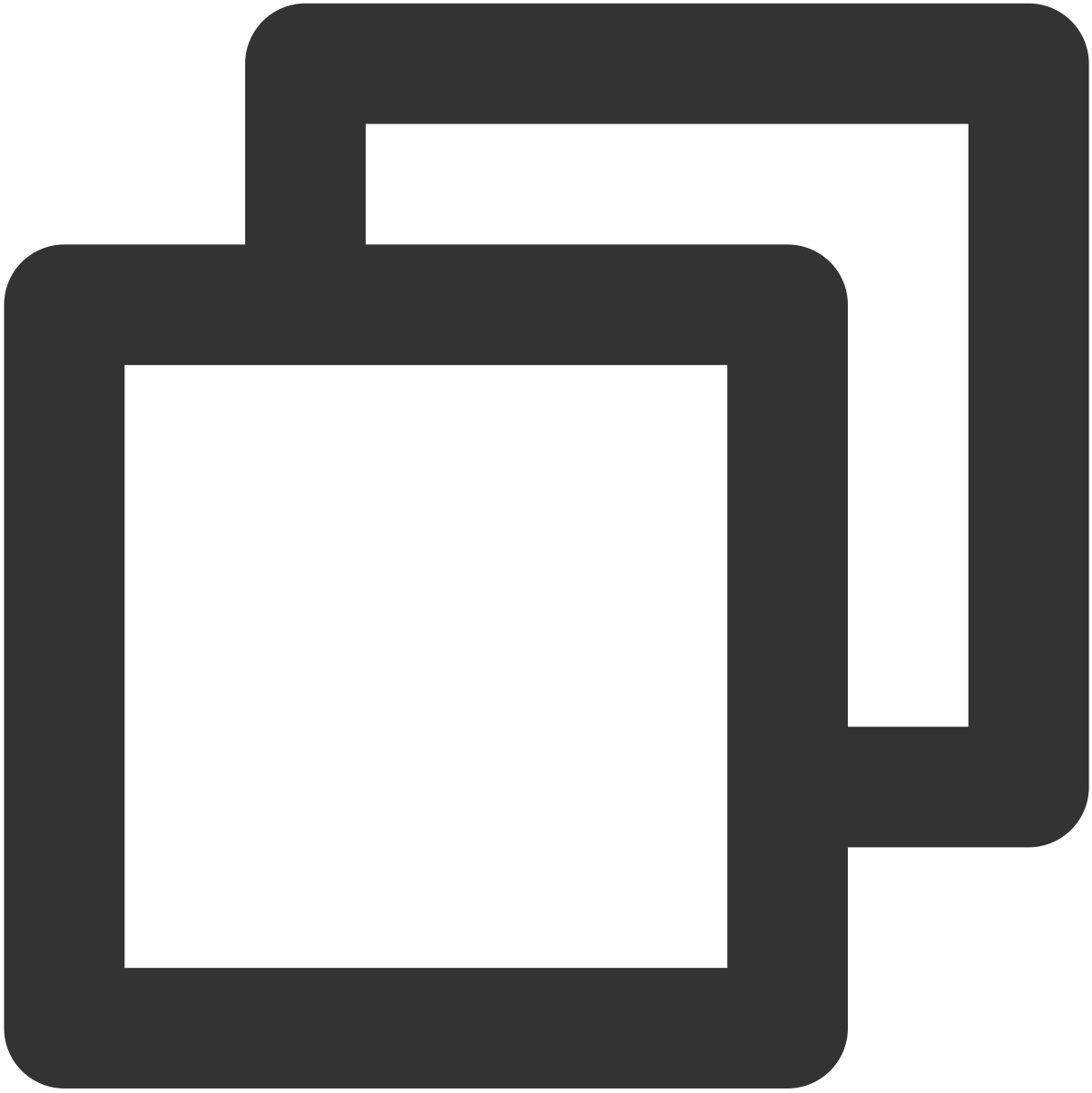


```
[[V2TIMManager sharedInstance] accept:inviteId data:data succ:^{
    // Answer successfully, render the call page, and stop the call ringtone
    if ([callType isEqualToString:@"videoCall"]) {
        // Start video call
        [self startVideoCall];
    } else {
        // Start audio call
        [self startAudioCall];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
} fail:^(int code, NSString *desc) {
    // Answer failed, prompt for exception or retry
}];
```

2. Caller receives answer notification



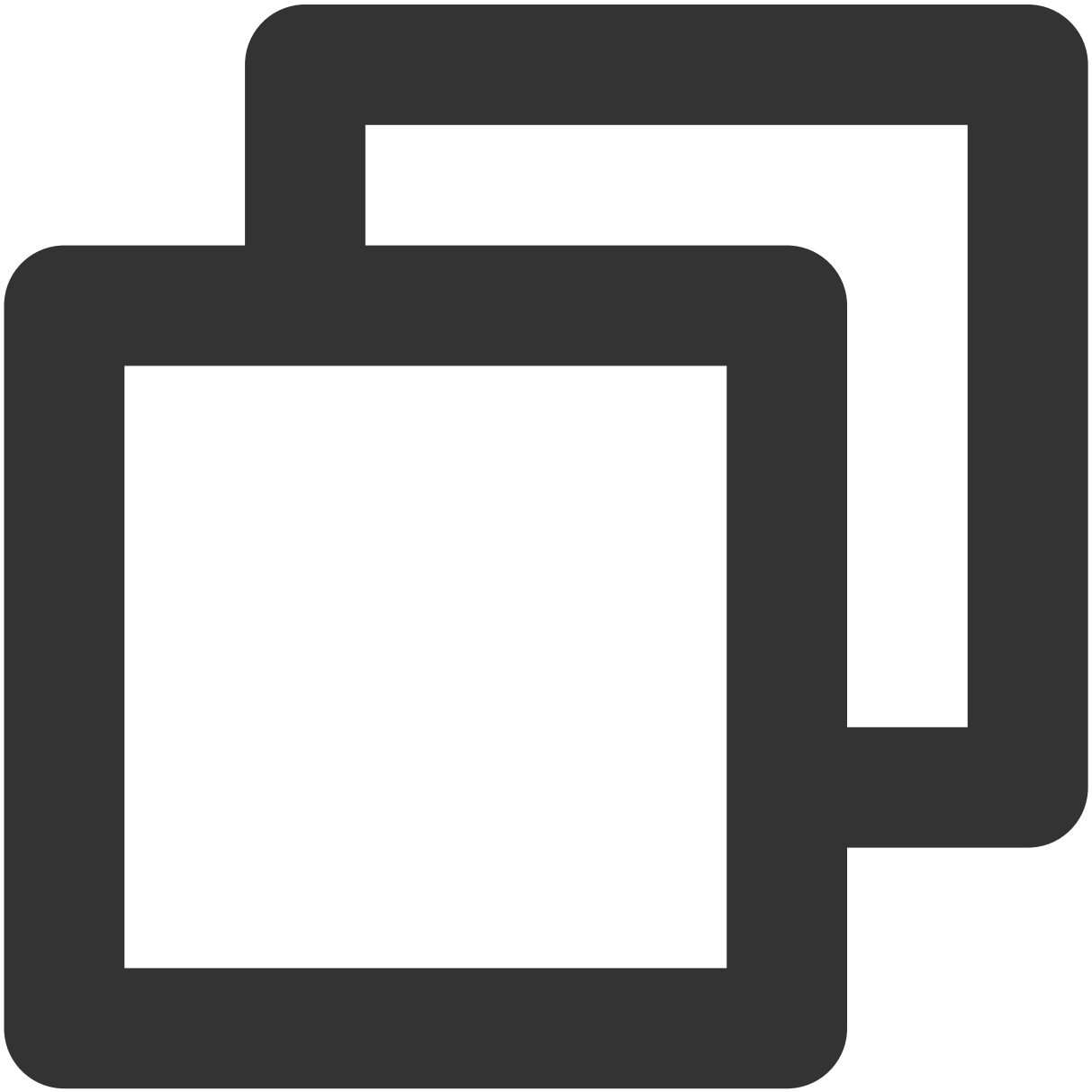
```
#pragma mark - V2TIMSignalingListener

- (void)onInviteeAccepted:(NSString *)inviteID invitee:(NSString *)invitee data:(NS
    if ([self.callType isEqualToString:@"videoCall"]) {
        // Start video call
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            [self startVideoCall];
        } else {
            // Start audio call
            [self startAudioCall];
        }
    }
```

Audio Call

1. Both caller and callee enter **the same TRTC room** to start an audio call.



©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)startAudioCall {
    TRTCParams *params = [[TRTCParams alloc] init];
    // TRTC authentication credential, generated on the server
    params.sdkAppId = SDKAPPID;
    // TRTC application ID, obtained from the console
    params.userSig = USERSIG;
    // Take the room ID string as an example
    params.strRoomId = self.roomId;
    // Username, it is recommended to stay sync with IM
    params.userId = self.userId;

    [self.trtcCloud startLocalAudio:TRTCAudioQualitySpeech];
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneAudioCall];
}
```

Note:

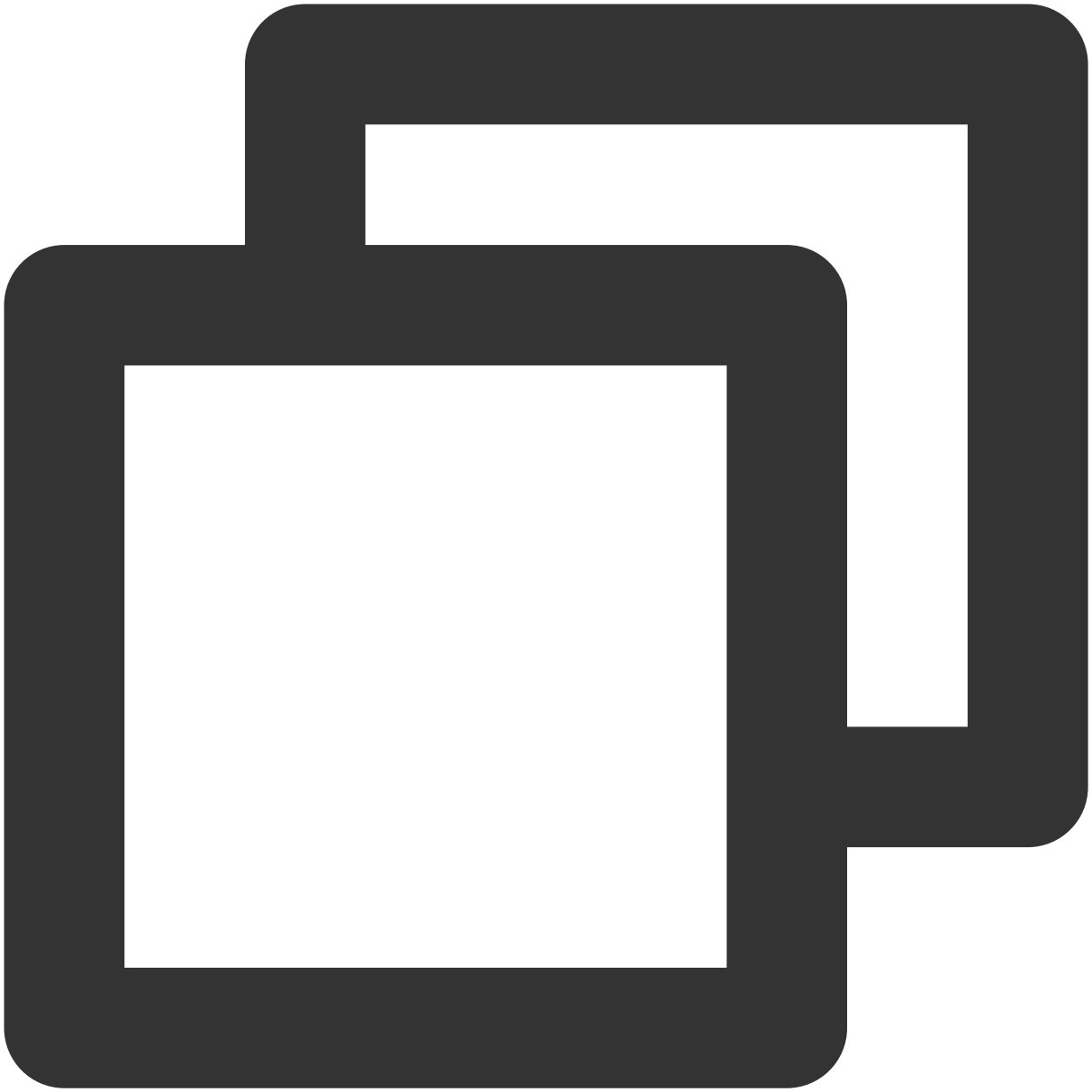
In audio call mode, the TRTC room entry scenario should select `TRTCAppSceneAudioCall`, and there is no need to specify a room entry role `TRTCRoleType`.

Starting local audio capture `startLocalAudio` allows you to set audio quality parameters at the same time. For audio call modes, it is recommended to select `TRTCAudioQualitySpeech`.

Under the SDK's default auto subscription mode, after a user enters a room, they will immediately receive the audio stream from that room, which will be automatically decoded and played without manual pulling.

2. Notification of room entry result, indicates call status.

©2013-2022 Tencent Cloud. All rights reserved.



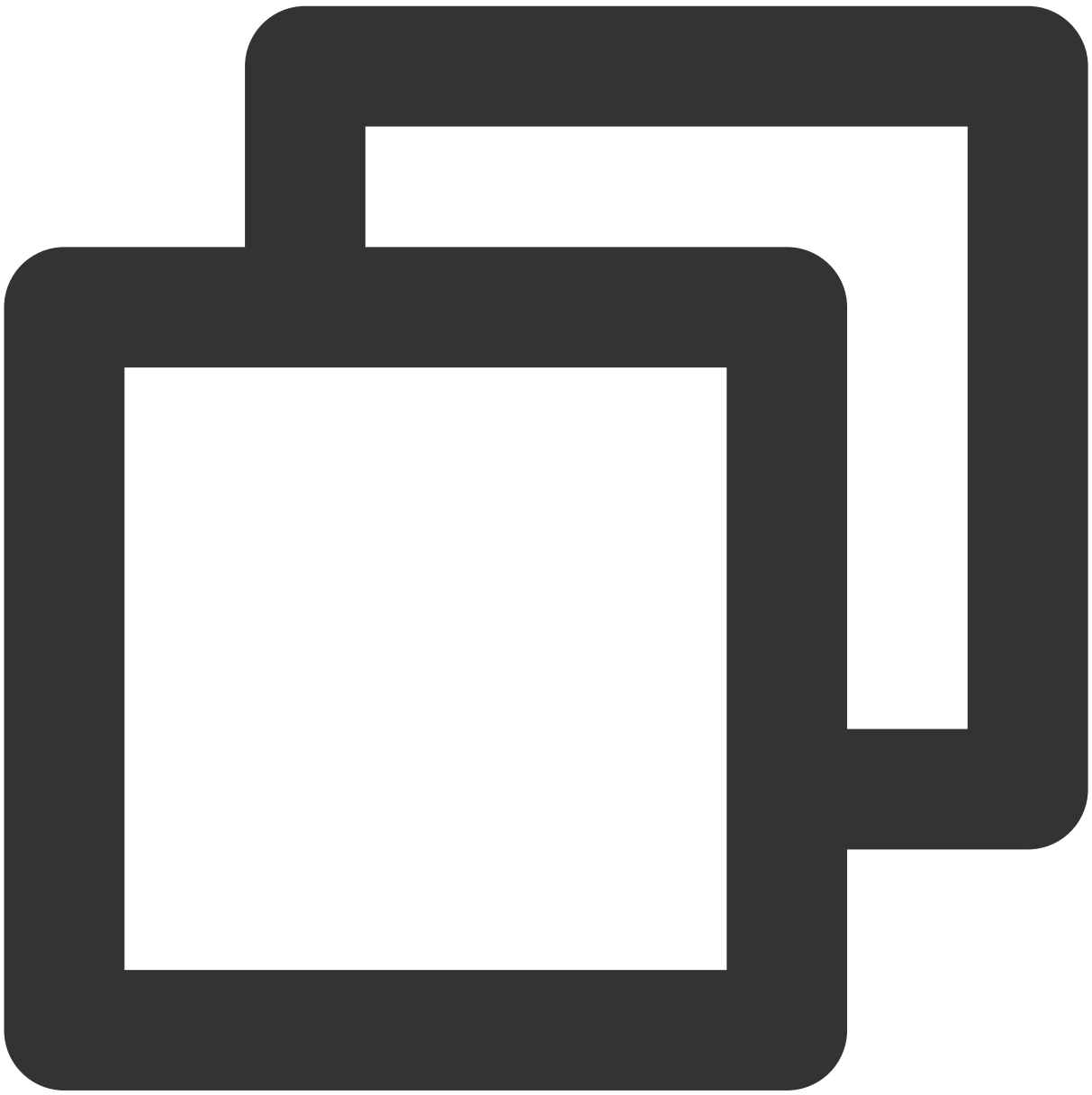
```
// Mark whether the call is in progress
@property (nonatomic, assign) BOOL isOnCalling;

#pragma mark - TRTCCloudDelegate

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // Room entry successful, indicates that the call is in progress
        self.isOnCalling = YES;
    } else {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Failed to enter the room, prompt for call exception
        self.isOnCalling = NO;
    }
}
```

Video Call

1. Both caller and callee enter **the same TRTC room** to start a video call.



```
- (void)startVideoCall {
    TRTCParams *params = [[TRTCParams alloc] init];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    // TRTC authentication credential, generated on the server
    params.sdkAppId = SDKAPPID;
    // TRTC application ID, obtained from the console
    params.userSig = USERSIG;
    // Take the room ID string as an example
    params.strRoomId = self.roomId;
    // Username, it is recommended to stay sync with IM
    params.userId = self.userId;

    [self.trtcCloud startLocalAudio:TRTCAudioQualitySpeech];
    [self.trtcCloud enterRoom:params appScene:TRTCAppSceneVideoCall];
}
```

Note:

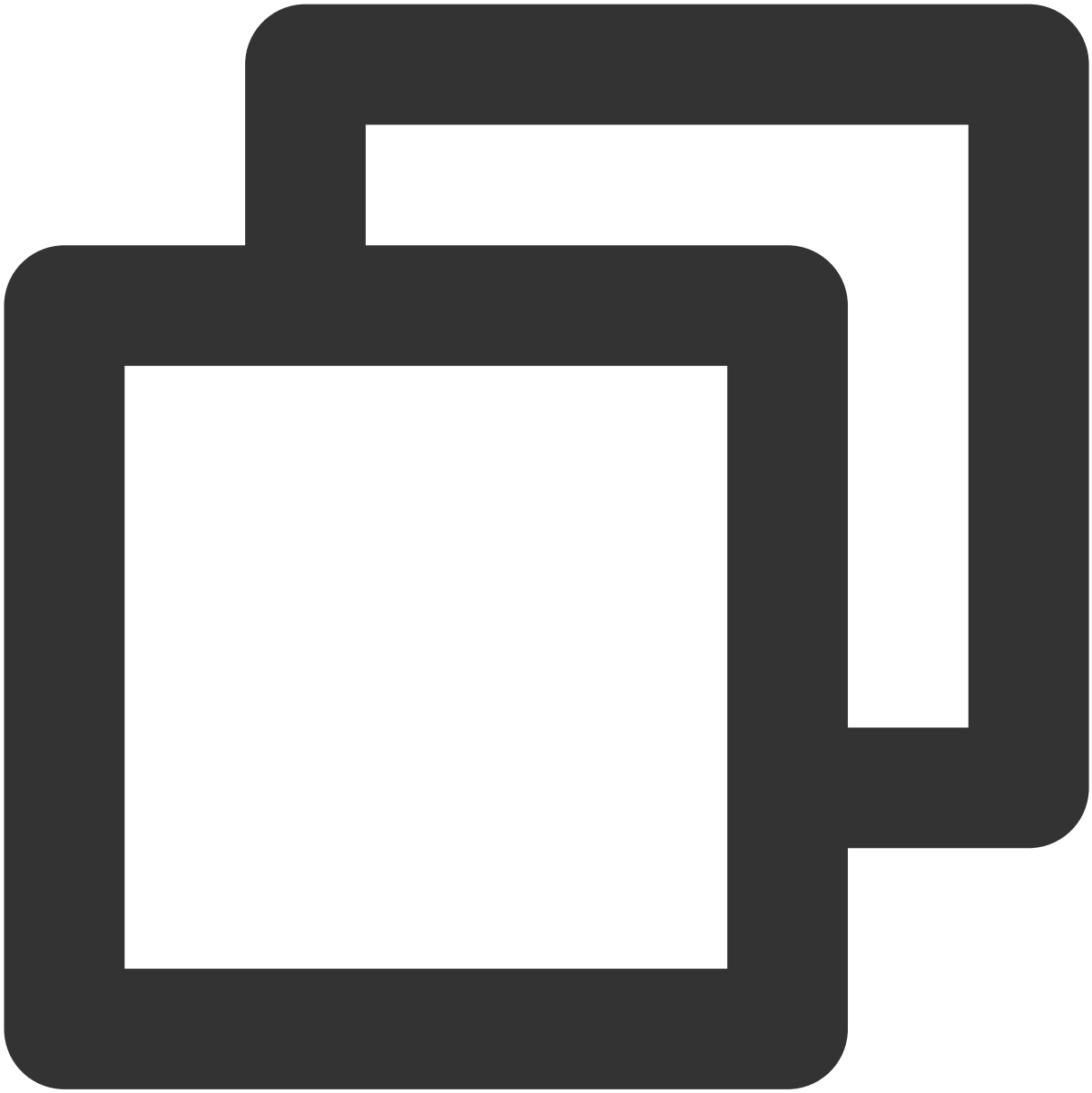
In video call mode, the TRTC room entry scenario should use `TRTCAppSceneVideoCall`, and there's no need to specify the room entry role `TRTCRoleType`.

Starting local audio capture `startLocalAudio` allows you to set audio quality parameters at the same time. For video call modes, it is recommended to select `TRTCAudioQualitySpeech`.

In the SDK's default automatic subscription mode, audio is automatically decoded and played back, while video requires manual invocation of `startRemoteView` to pull and render the remote video stream.

2. Notification of room entry result, indicates call status. Pull remote video stream.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Mark whether the call is in progress
@property (nonatomic, assign) BOOL isOnCalling;

#pragma mark - TRTCCloudDelegate

// Event callback for the result of entering the room
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        // Room entry successful, indicates that the call is in progress
        self.isOnCalling = YES;
    } else {
```

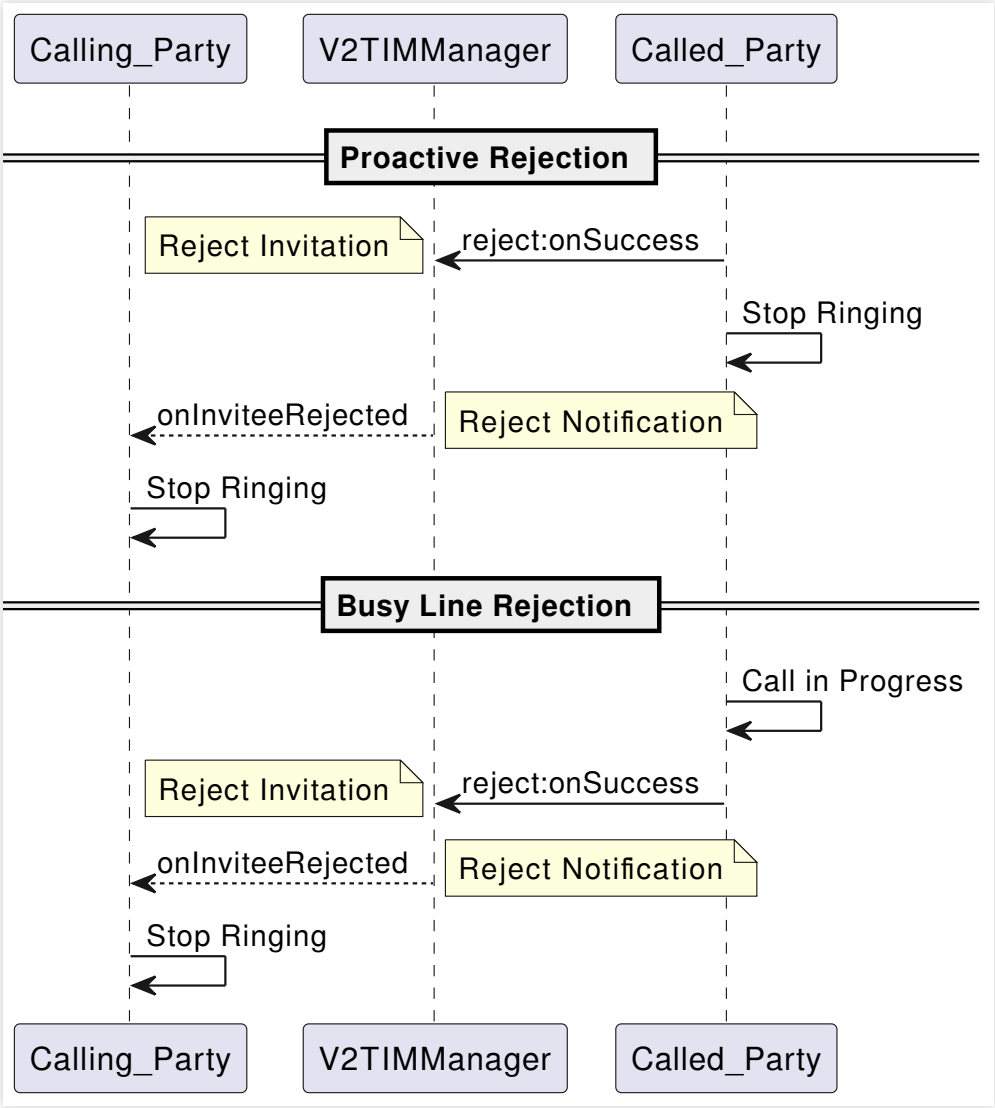
©2013-2022 Tencent Cloud. All rights reserved.

```
        // Failed to enter the room, prompt for call exception
        self.isOnCalling = NO;
    }
}

// Pull remote video stream
- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available {
    // The remote user publishes/unpublishes the primary video
    if (available) {
        // Subscribe to the remote user's video stream and bind the video rendering
        [self.trtcCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig vi
    } else {
        // Unsubscribe to the remote user's video stream and release the rendering
        [self.trtcCloud stopRemoteView:userId streamType:TRTCVideoStreamTypeBig];
    }
}
```

Step 4: reject call

Sequence Diagram

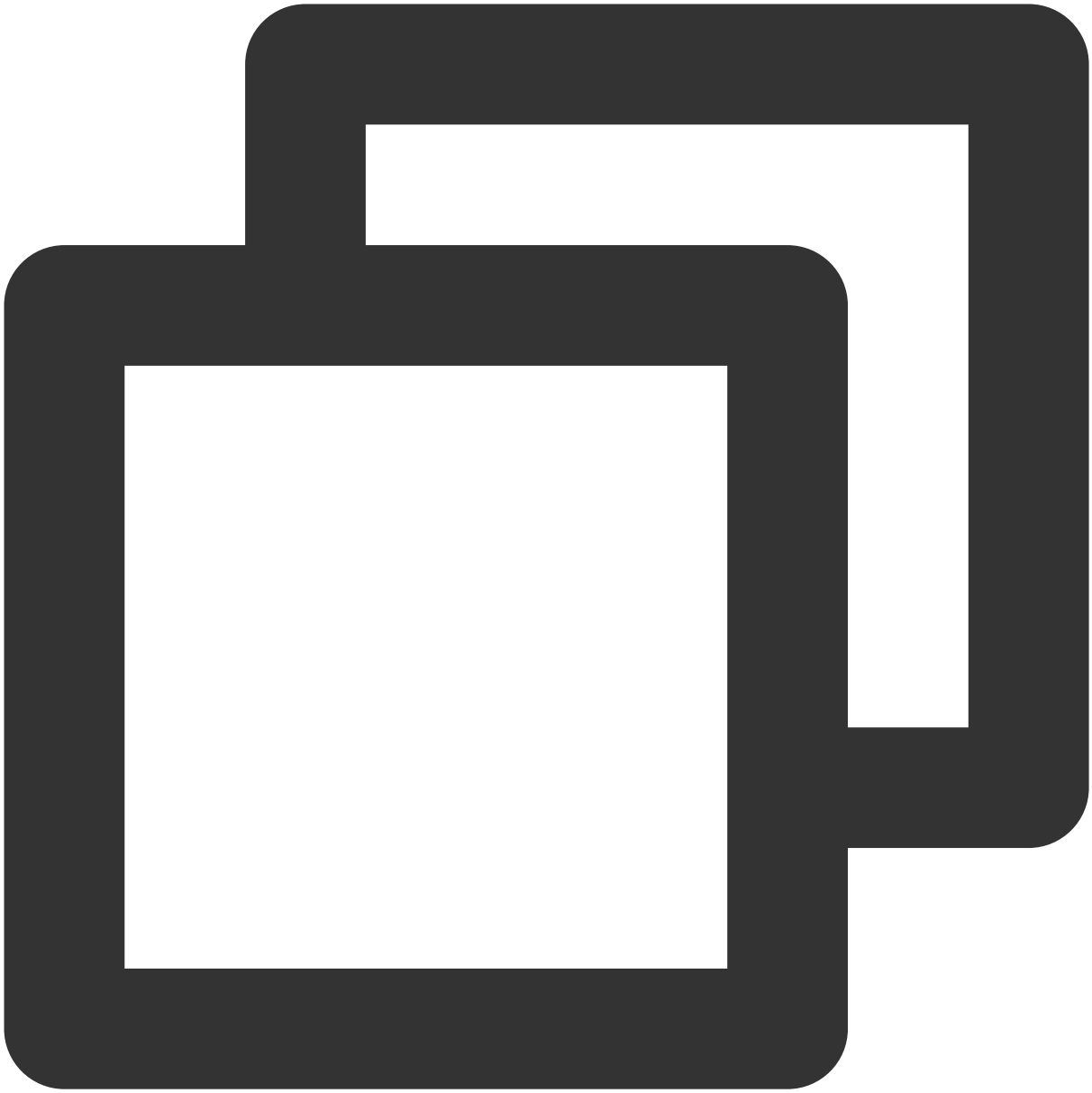
©2013-2022 Tencent Cloud. All rights reserved.



Proactive Rejection

1. Callee sends rejection signal

©2013-2022 Tencent Cloud. All rights reserved.

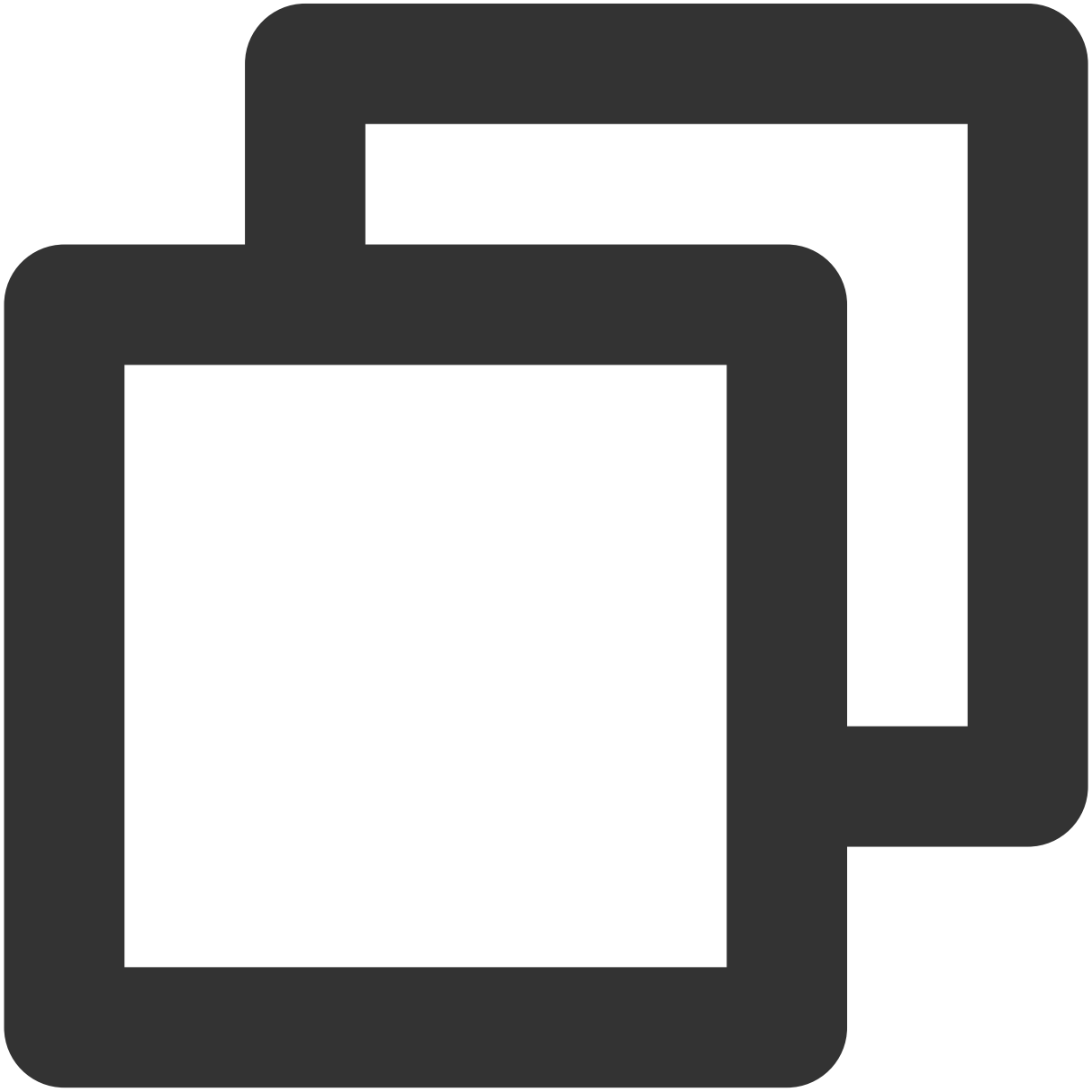


```
NSDictionary *dic = @{
    @"cmd": @"av_call",
    @"msg": @{
        // Specify the call type (video call, audio call)
        @"callType": @"videoCall",
        // Specify rejection type (Proactive Rejection, Busy Line Rejection)
        @"reason": @"active",
    },
};
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dic
                                                   options:NSJSONWritingPrettyPrint
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                    error:nil];
if (jsonData) {
    NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8S
    [[V2TIMManager sharedInstance] reject:self.inviteId data:jsonString succ:^{
        // Rejection successful. Terminate the call page, and stop the call ringtone
    } fail:^(int code, NSString *desc) {
        // Rejection failed, prompt for exception or retry
    }];
}
```

2. Caller receives rejection notification



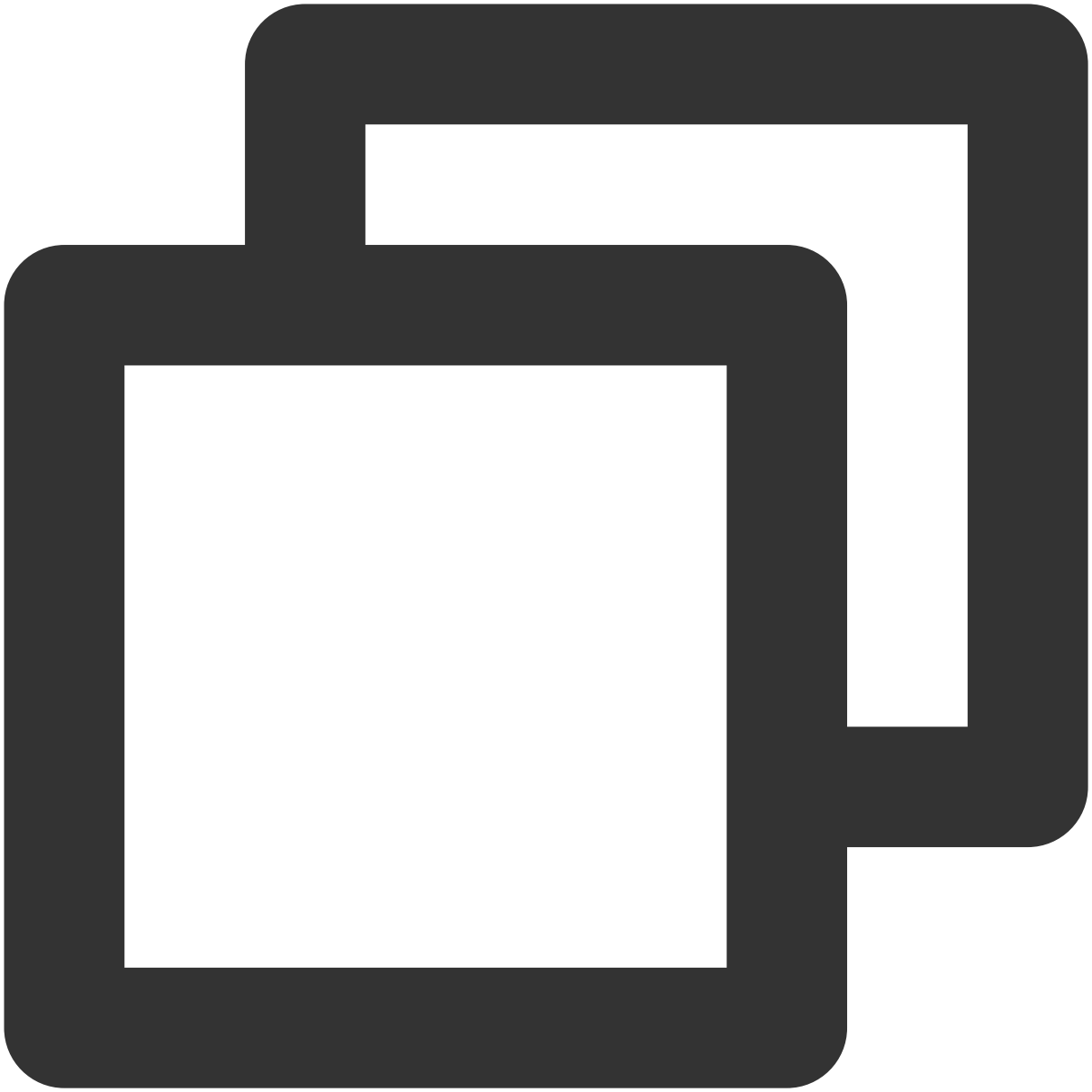
©2013-2022 Tencent Cloud. All rights reserved.

```
#pragma mark - V2TIMSignalingListener

- (void)onInviteeRejected:(NSString *)inviteID invitee:(NSString *)invitee data:(NS
    if (data && ![data isEqualToString:@""]) {
        NSData *jsonData = [data dataUsingEncoding:NSUTF8StringEncoding];
        NSDictionary *dictionary = [NSJSONSerialization JSONObjectWithData:jsonData
                                                                   options:NSJSONRe
                                                                     error:nil];
        if (dictionary) {
            NSString *command = dictionary[@"cmd"];
            NSDictionary *msg = dictionary[@"msg"];
            if ([command isEqualToString:@"av_call"]) {
                NSString *reason = msg[@"reason"];
                if ([reason isEqualToString:@"active"]) {
                    // Prompt that the other party rejects call
                } else if ([reason isEqualToString:@"busy"]) {
                    // Prompt that the other party is busy
                }

                // Terminate the call page, and stop the call ringtone
            }
        }
    }
}
```

Busy Line Rejection

Callee receives a new call invitation, if the local call status is in a call, the caller automatically rejects the call.

©2013-2022 Tencent Cloud. All rights reserved.



```
- (void)onReceiveNewInvitation:(NSString *)inviteID inviter:(NSString *)inviter gro
    if (data && ![data isEqualToString:@""]) {
        NSData *jsonData = [data dataUsingEncoding:NSUTF8StringEncoding];
        NSDictionary *dictionary = [NSJSONSerialization JSONObjectWithData:jsonData
                                                                   options:NSJSONRe
                                                                     error:nil];
        if (dictionary) {
            NSString *command = dictionary[@"cmd"];
            NSDictionary *msg = dictionary[@"msg"];
            if ([command isEqualToString:@"av_call"] && self.isOnCalling) {
                NSDictionary *dic = @{
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                    @"cmd": @"av_call",
                    @"msg": @{
                        // Specify the call type (video call, audio call)
                        @"callType": @"videoCall",
                        // Specify rejection type (Proactive Rejection, Busy Line R
                        @"reason": @"busy",
                    },
                };
                NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dic
                                                                   options:NSJSONWr
                                                                     error:nil];
                if (jsonData) {
                    NSString *jsonString = [[NSString alloc] initWithData:jsonData
                    // Local call is in progress, and sends busy line rejection sig
                    [[V2TIMManager sharedInstance] reject:inviteID data:jsonString
                        // Busy line rejection successful
                    } fail:^(int code, NSString *desc) {
                        // Busy line rejection failed
                    }];
                }
            }
        }
    }
}
```

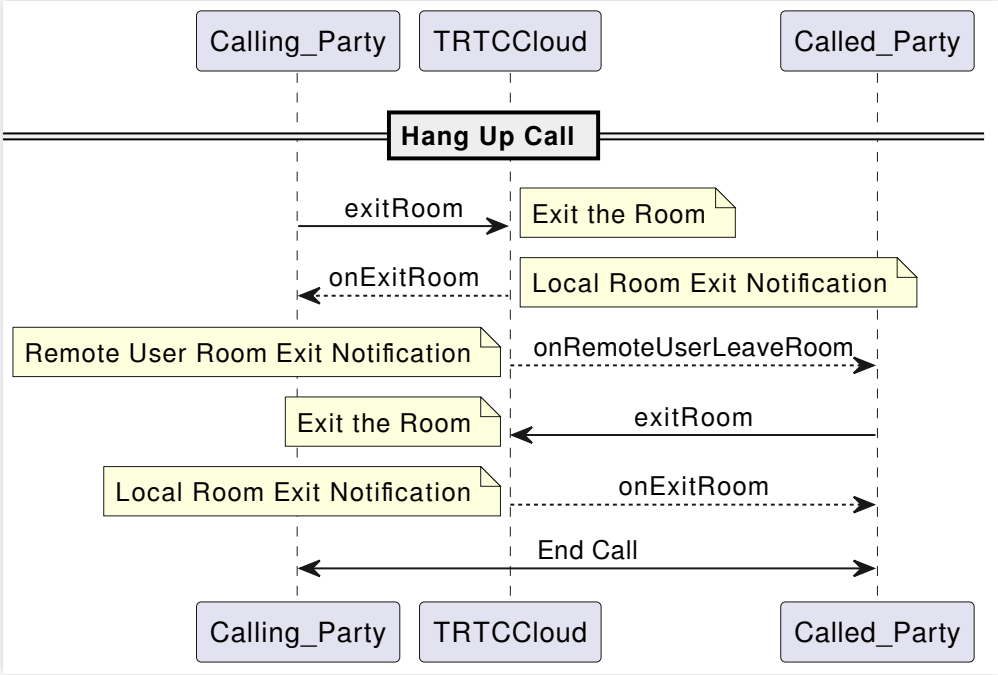
Note:

Both proactive rejection and busy line rejection use the `reject` signaling for implementation, but it's important to distinguish them through the `reason` field of the custom `data` in the signaling.

Step 5: hang up

Sequence Diagram

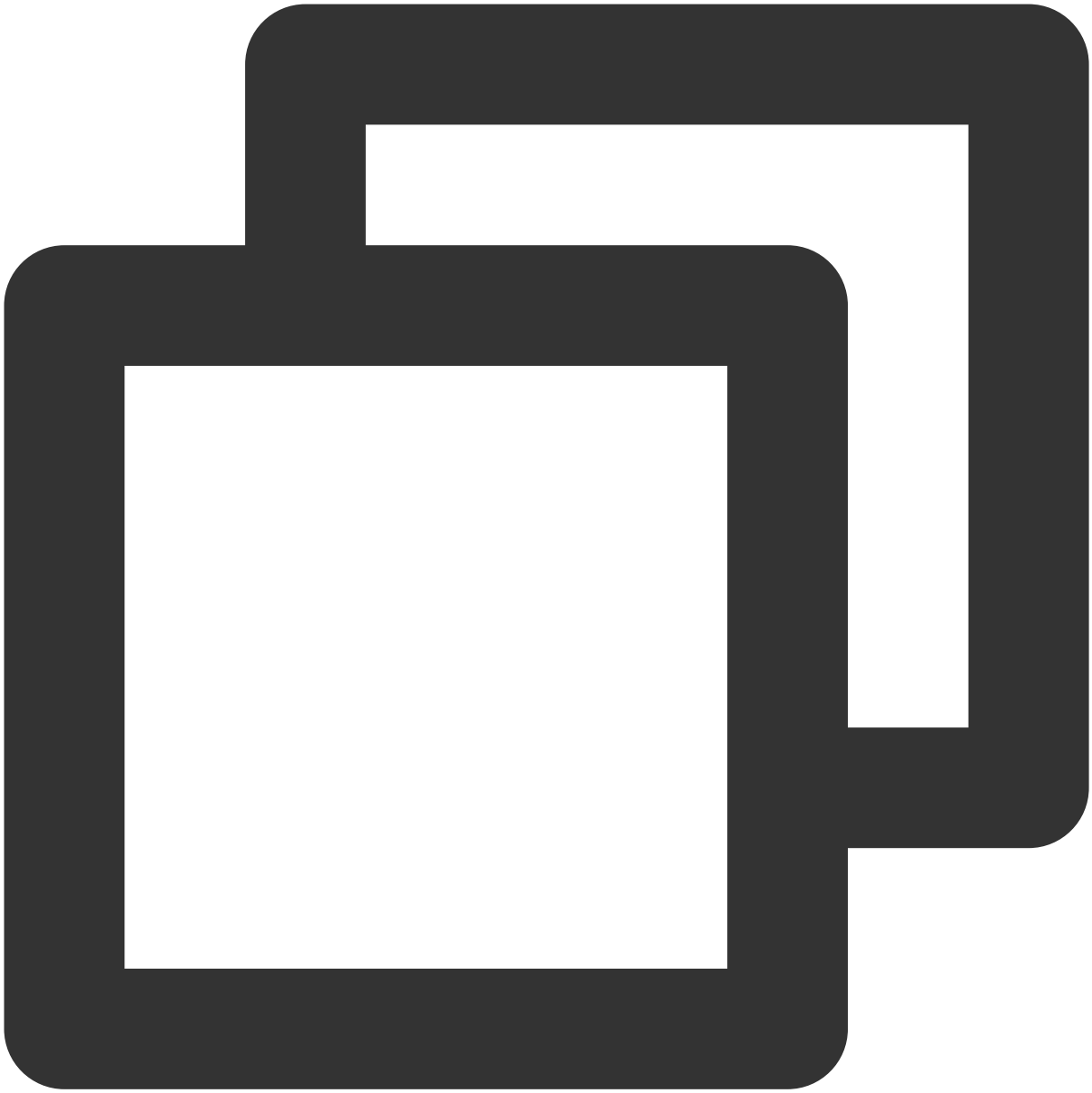
©2013-2022 Tencent Cloud. All rights reserved.



Hang Up Call

1. Either party exits the room, and reset the local call status.

©2013-2022 Tencent Cloud. All rights reserved.

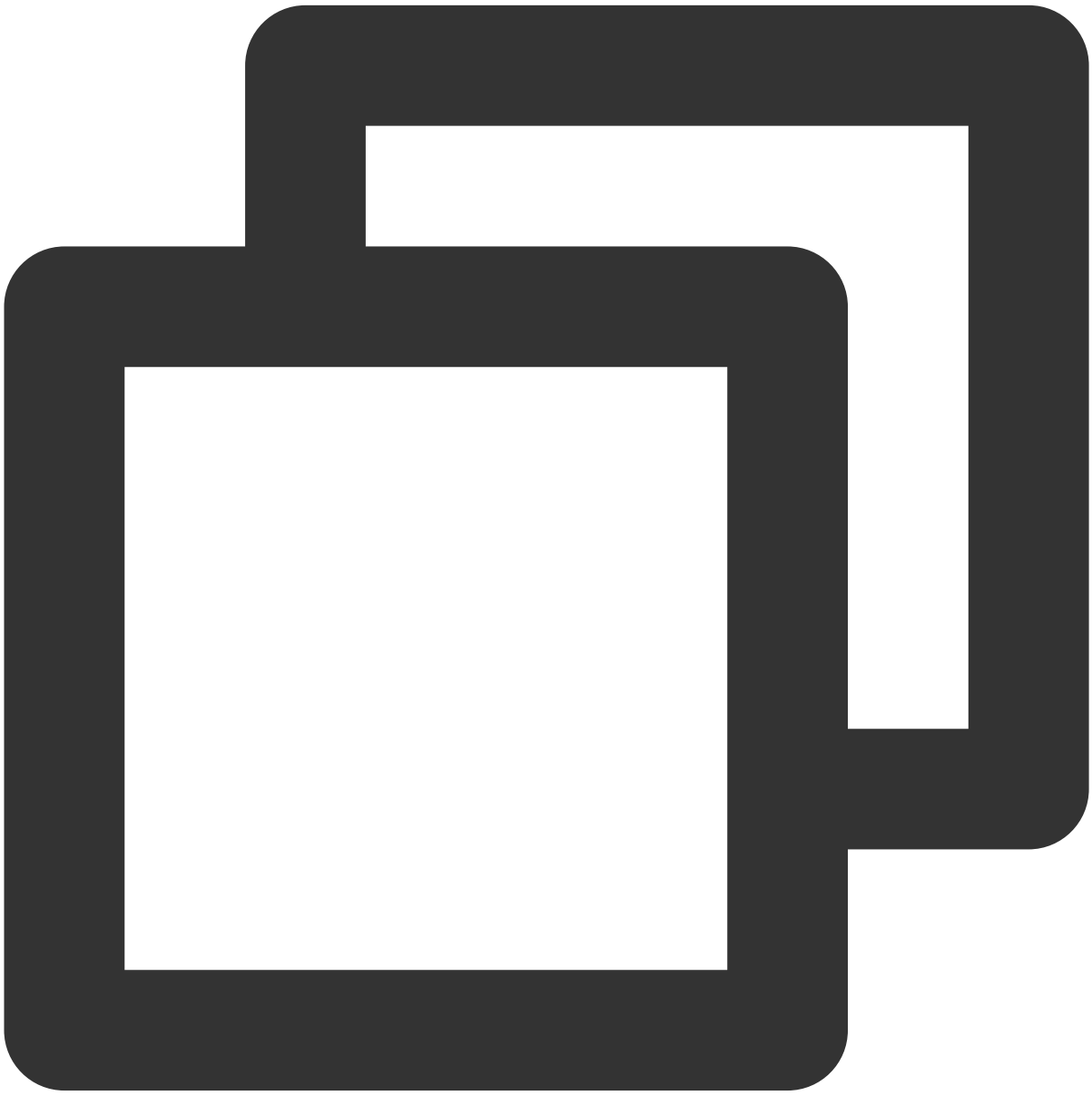


```
- (void)hangup {
    [self.trtcCloud stopLocalAudio];
    [self.trtcCloud stopLocalPreview];
    [self.trtcCloud exitRoom];
}

#pragma mark - TRTCCloudDelegate

- (void)onExitRoom:(NSInteger)reason {
    // Successfully exited the room and hung up the call
    self.isOnCalling = NO;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}
```

2. The other party receives a notification that the remote side has exited the room, locally executes to exit room and resets the call status.



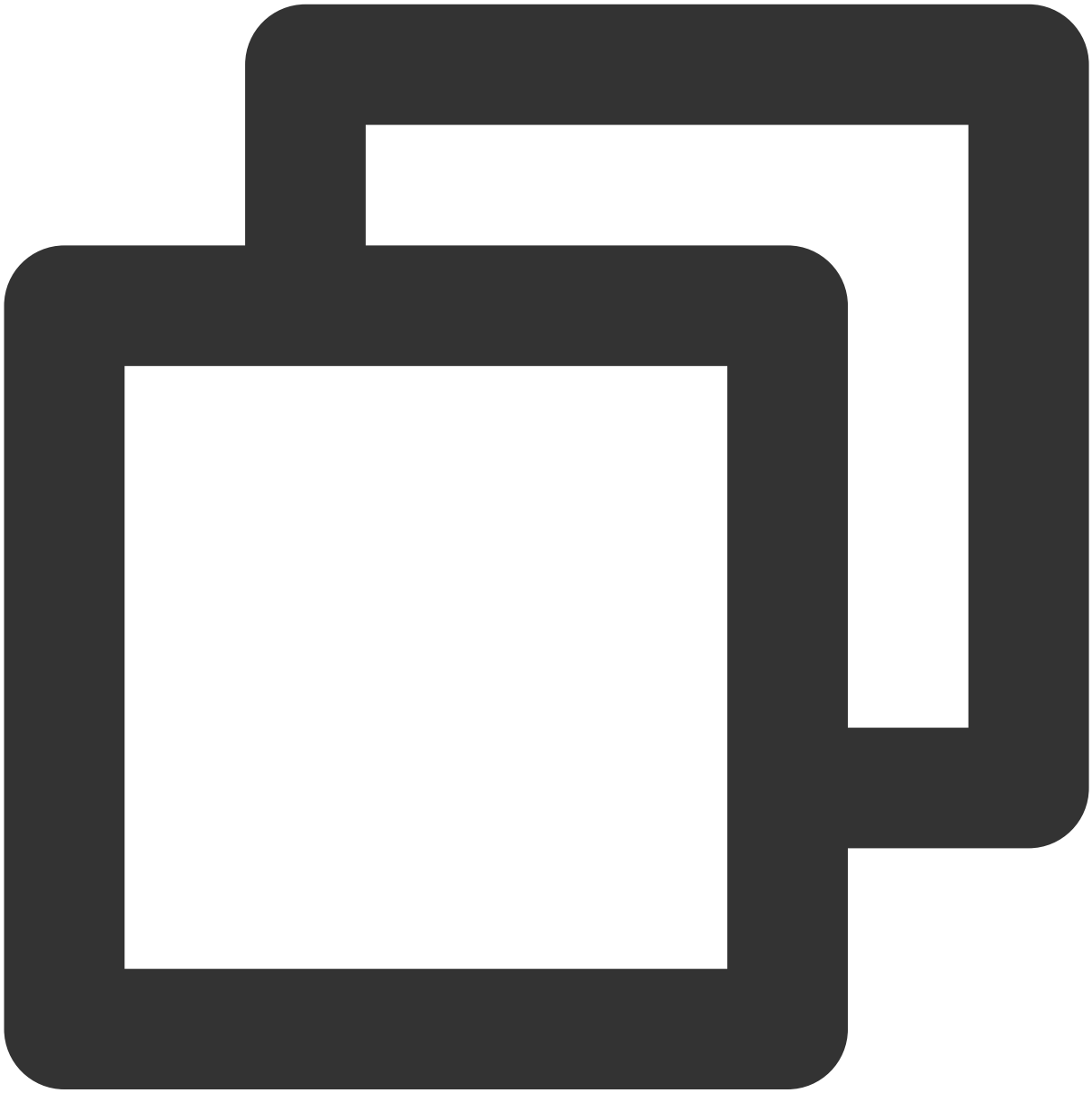
```
#pragma mark - TRTCCloudDelegate

- (void)onRemoteUserLeaveRoom:(NSString *)userId reason:(NSInteger)reason {
    [self hangup];
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- (void)onExitRoom:(NSInteger)reason {
    // Successfully exited the room and hung up the call
    self.isOnCalling = NO;
}
```

Step 6: feature control

Turn on/off microphone

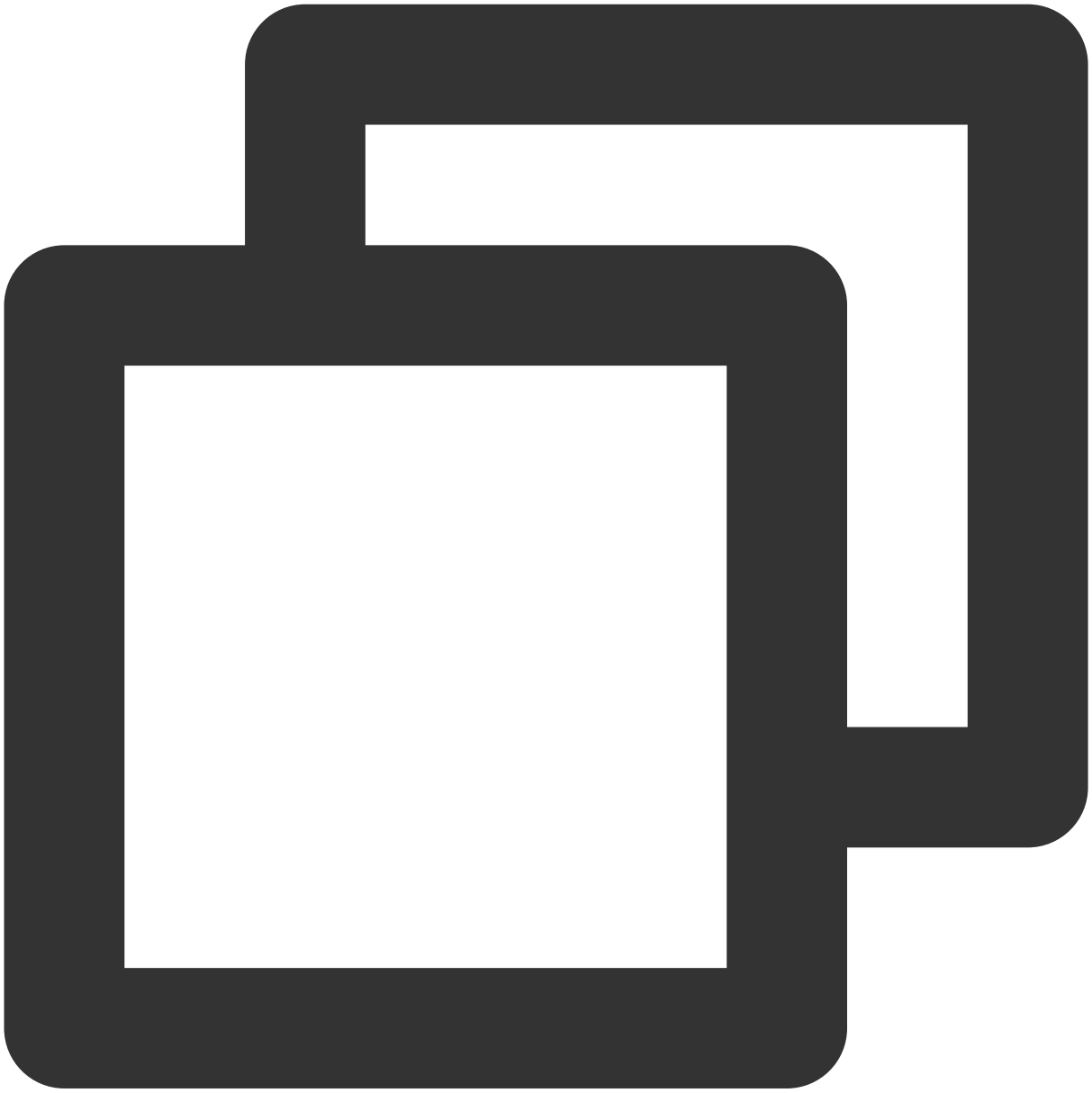


```
// Turn the mic on
[self.trtcCloud muteLocalAudio:NO];
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Turn the mic off
[self.trtcCloud muteLocalAudio:YES];
```

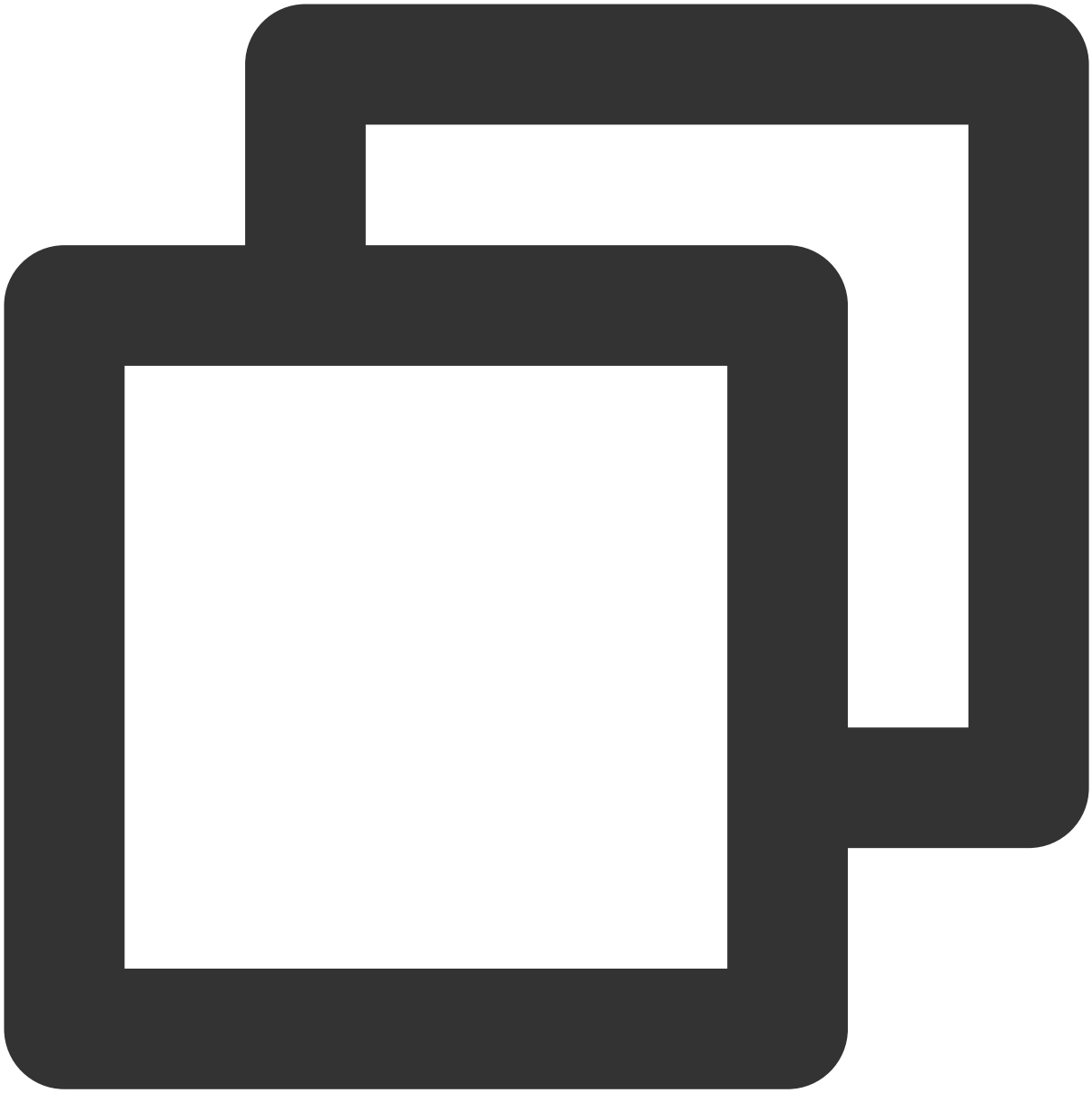
Turn on/off speaker



```
// Turn the speaker on
[self.trtcCloud muteAllRemoteAudio:NO];
// Turn the speaker off
[self.trtcCloud muteAllRemoteAudio:YES];
```

©2013-2022 Tencent Cloud. All rights reserved.

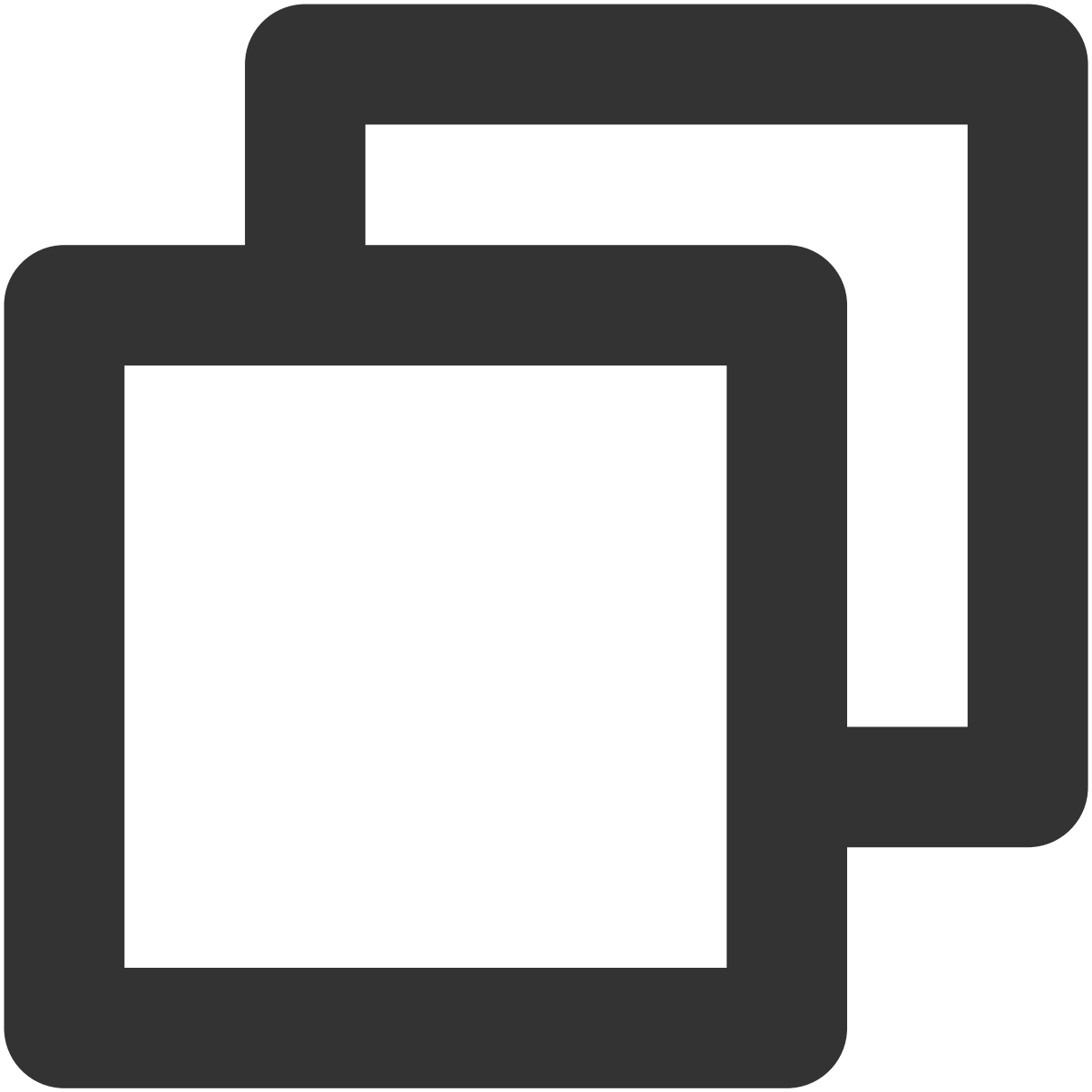
Turn on/off camera



```
// Turn the camera on, specifying front or rear camera and the rendering control
[self.trtcCloud startLocalPreview:self.isFrontCamera view:self.previewView];
// Turn the camera off
[self.trtcCloud stopLocalPreview];
```

Hands-free/Earpiece Switching

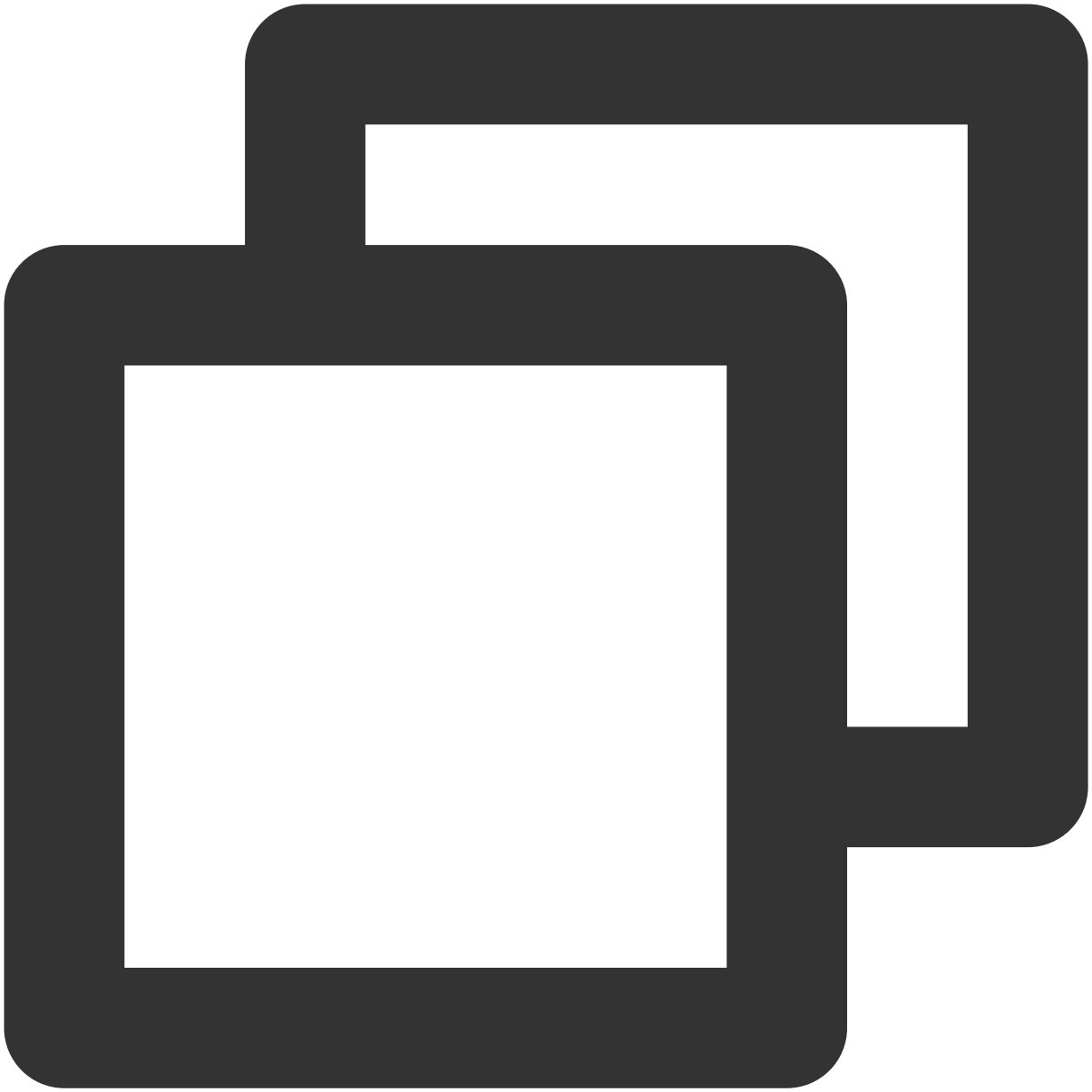
©2013-2022 Tencent Cloud. All rights reserved.



```
// Switch to earpiece
[[self.trtcCloud getDeviceManager] setAudioRoute:TXAudioRouteEarpiece];
// Switch to speakerphone
[[self.trtcCloud getDeviceManager] setAudioRoute:TXAudioRouteSpeakerphone];
```

Camera Switching

©2013-2022 Tencent Cloud. All rights reserved.



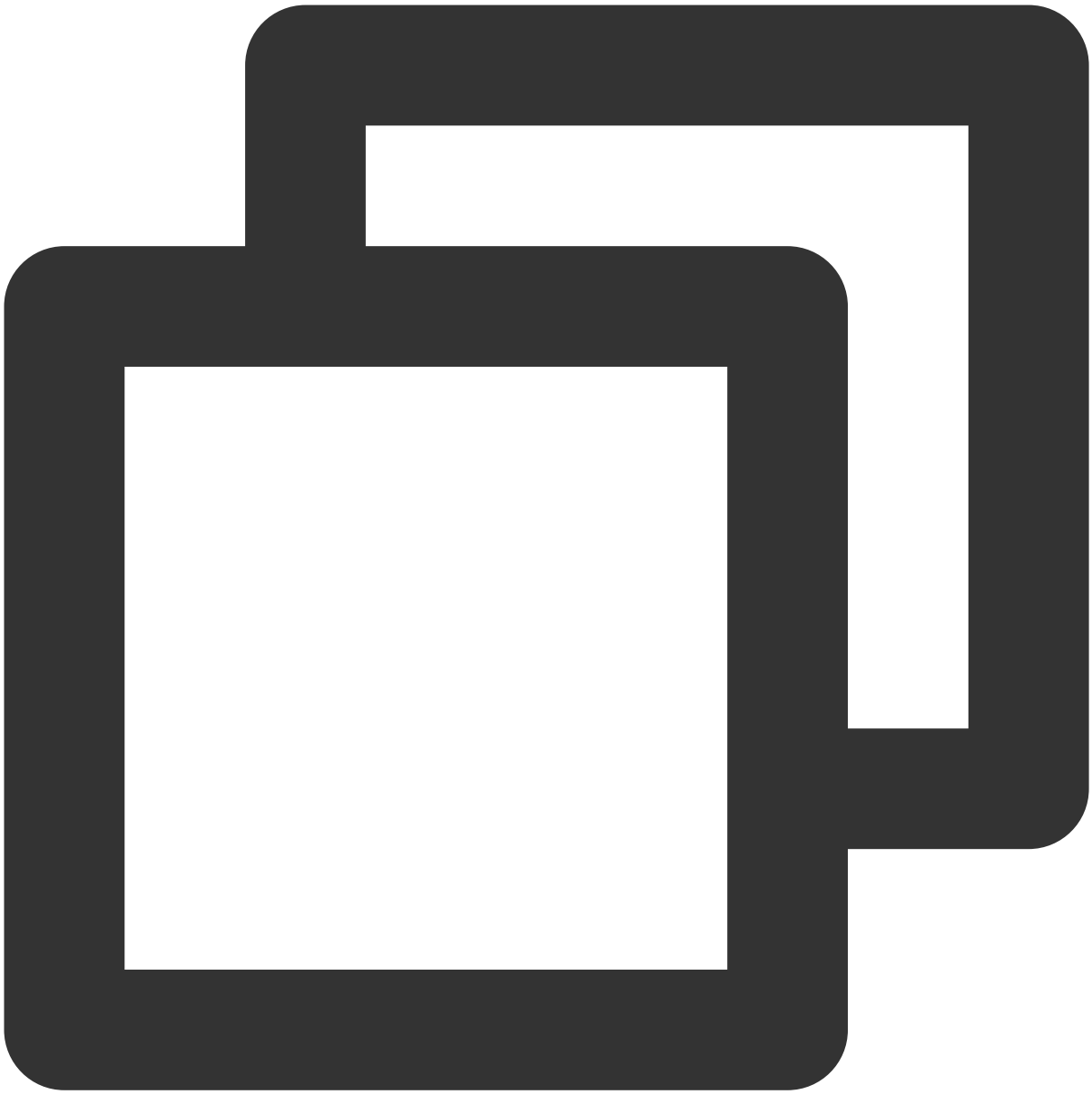
```
// Determine if the current camera is front-facing
BOOL isFrontCamera = [[self.trtcCloud getDeviceManager] isFrontCamera];
// Switch between front and rear cameras, true: switch to front-facing; false: swit
[[self.trtcCloud getDeviceManager] switchCamera:!isFrontCamera];
```

Advanced Features

Network Status Prompt

©2013-2022 Tencent Cloud. All rights reserved.

During audio and video calls, it is often necessary to prompt when the other party's network status is poor, thereby creating an expectation of call lag.



```
#pragma mark - TRTCCloudDelegate

- (void)onNetworkQuality:(TRTCQualityInfo *)localQuality remoteQuality:(NSArray<TRT
    if (remoteQuality.count > 0) {
        switch(remoteQuality[0].quality) {
            case TRTCQuality_Excellent:
                NSLog(@"The other party's network is very good");
                break;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
            case TRTCQuality_Good:
                NSLog(@"The other party's network is quite good");
                break;
            case TRTCQuality_Poor:
                NSLog(@"The other party's network is average");
                break;
            case TRTCQuality_Bad:
                NSLog(@"The other party's network is relatively poor");
                break;
            case TRTCQuality_Vbad:
                NSLog(@"The other party's network is very poor");
                break;
            case TRTCQuality_Down:
                NSLog(@"The other party's network is extremely poor");
                break;
            default:
                NSLog(@"Undefined ");
                break;
        }
    }
}
```

Note:

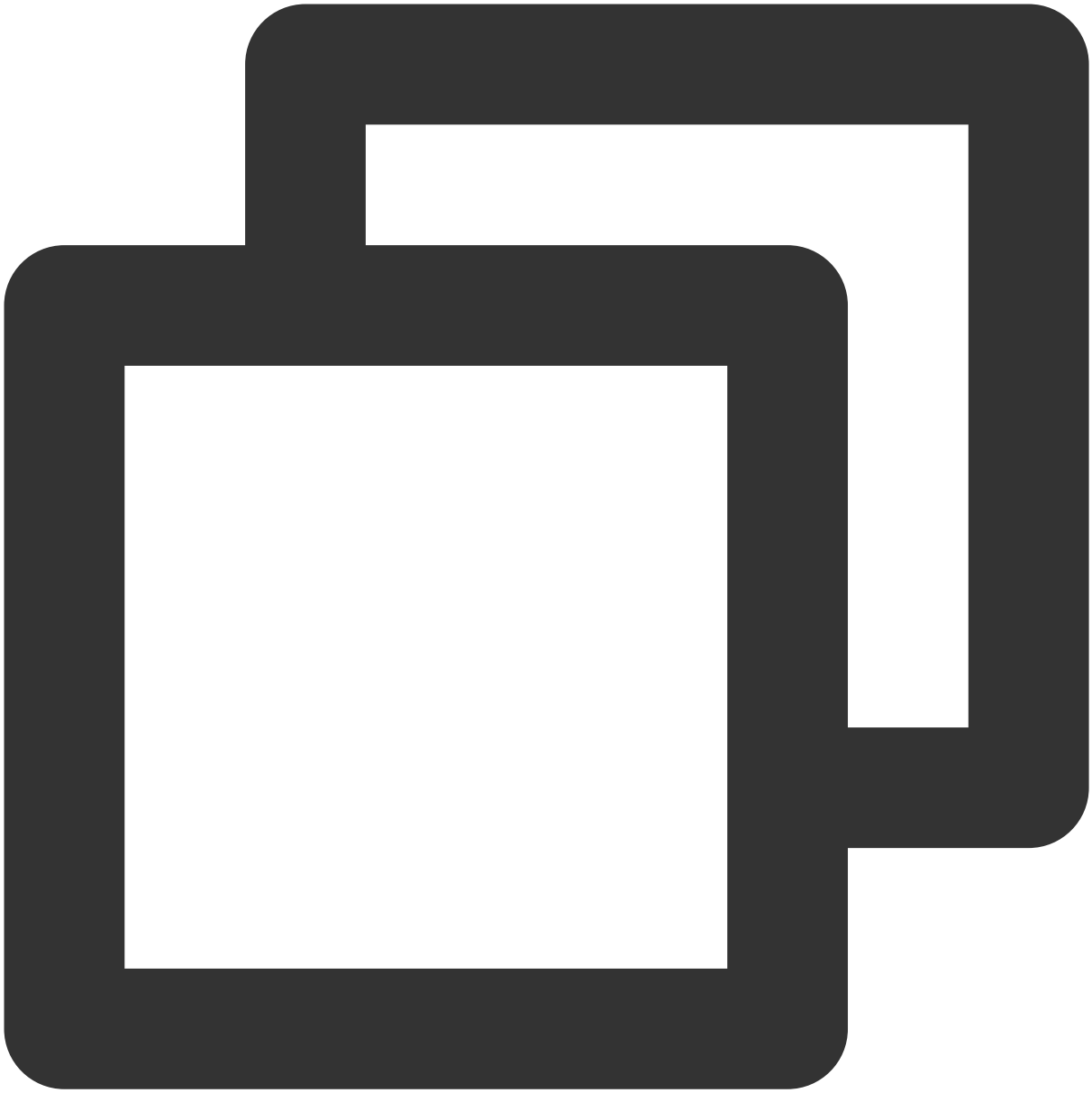
`localQuality` represents the local user network quality assessment result, and its userId field is empty.

`remoteQuality` represents the remote user network quality assessment result, which is influenced by factors on both the remote and local sides.

Call Duration Statistics

It is recommended to use the time when a remote user joins the TRTC room as the start time for calculating call duration, and the time when the local user exits the room as the end time for calculating call duration.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Start call time
@property (nonatomic, assign) NSTimeInterval callStartTime;
// End call time
@property (nonatomic, assign) NSTimeInterval callFinishTime;
// Call duration (seconds)
@property (nonatomic, assign) NSInteger callDuration;

// Callback for remote user entering room
- (void)onRemoteUserEnterRoom:(NSString *)userId {
    self.callStartTime = [[NSDate date] timeIntervalSince1970];
}
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// Callback for local user exiting room
- (void)onExitRoom:(NSInteger)reason {
    self.callFinishTime = [[NSDate date] timeIntervalSince1970];
    self.callDuration = (NSInteger)(self.callFinishTime - self.callStartTime);
}
```

Note:

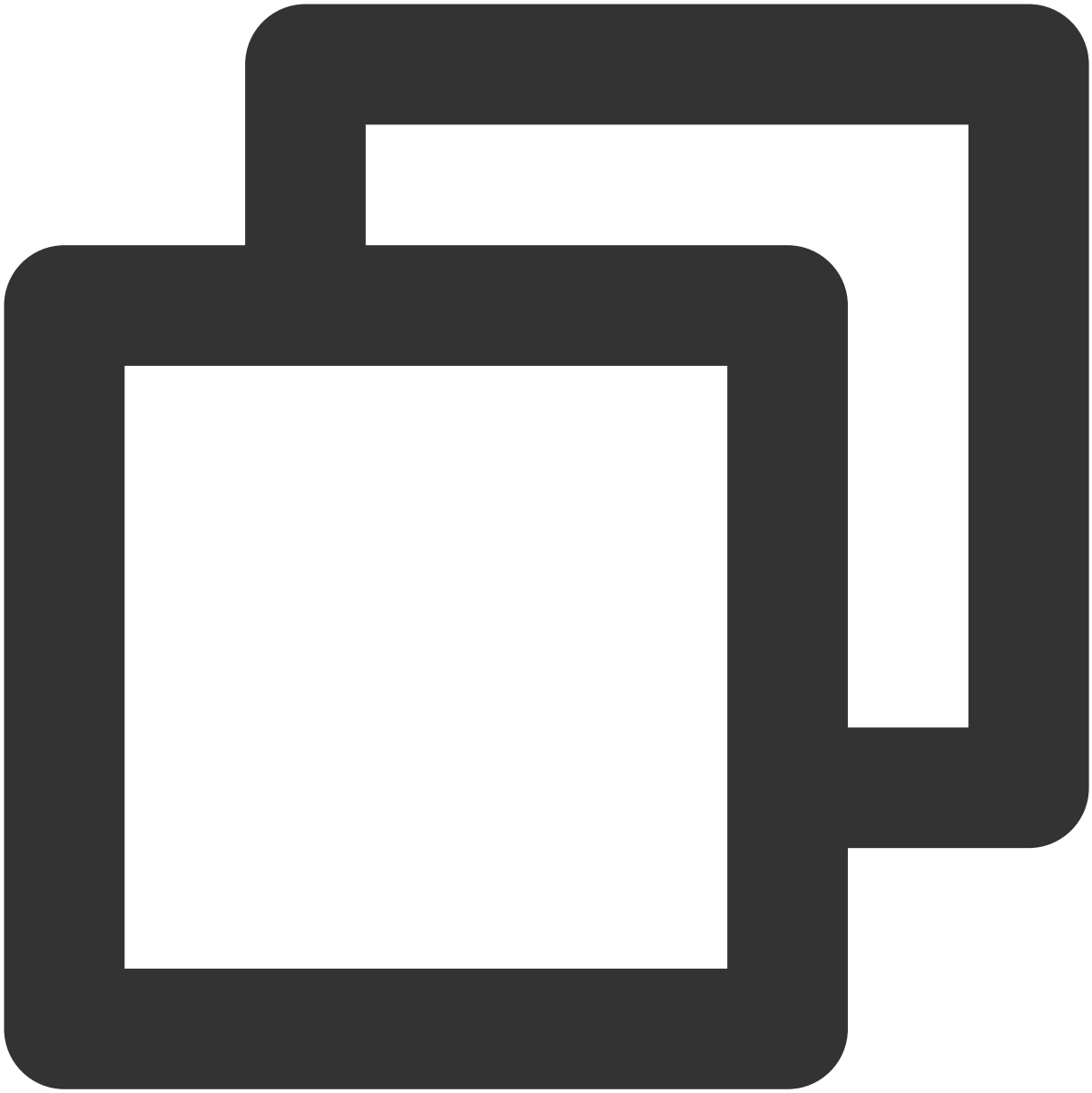
In cases of exceptions such as forced closure or network disconnection, the client may not be able to log the relevant times. These can be monitored through Server Event Callback to track events of entering and exiting the room and calculate the duration of the call.

Video Beauty Effects

TRTC supports integrating third-party beauty effect products. Use the example of Special Effect to demonstrate the process of integrating the third-party beauty features.

1. Integrate Special Effect SDK, and apply for an authorization license. For details, see Live Show Streaming - Integration Preparation for steps.
2. Set the SDK material resource path (if any).

©2013-2022 Tencent Cloud. All rights reserved.



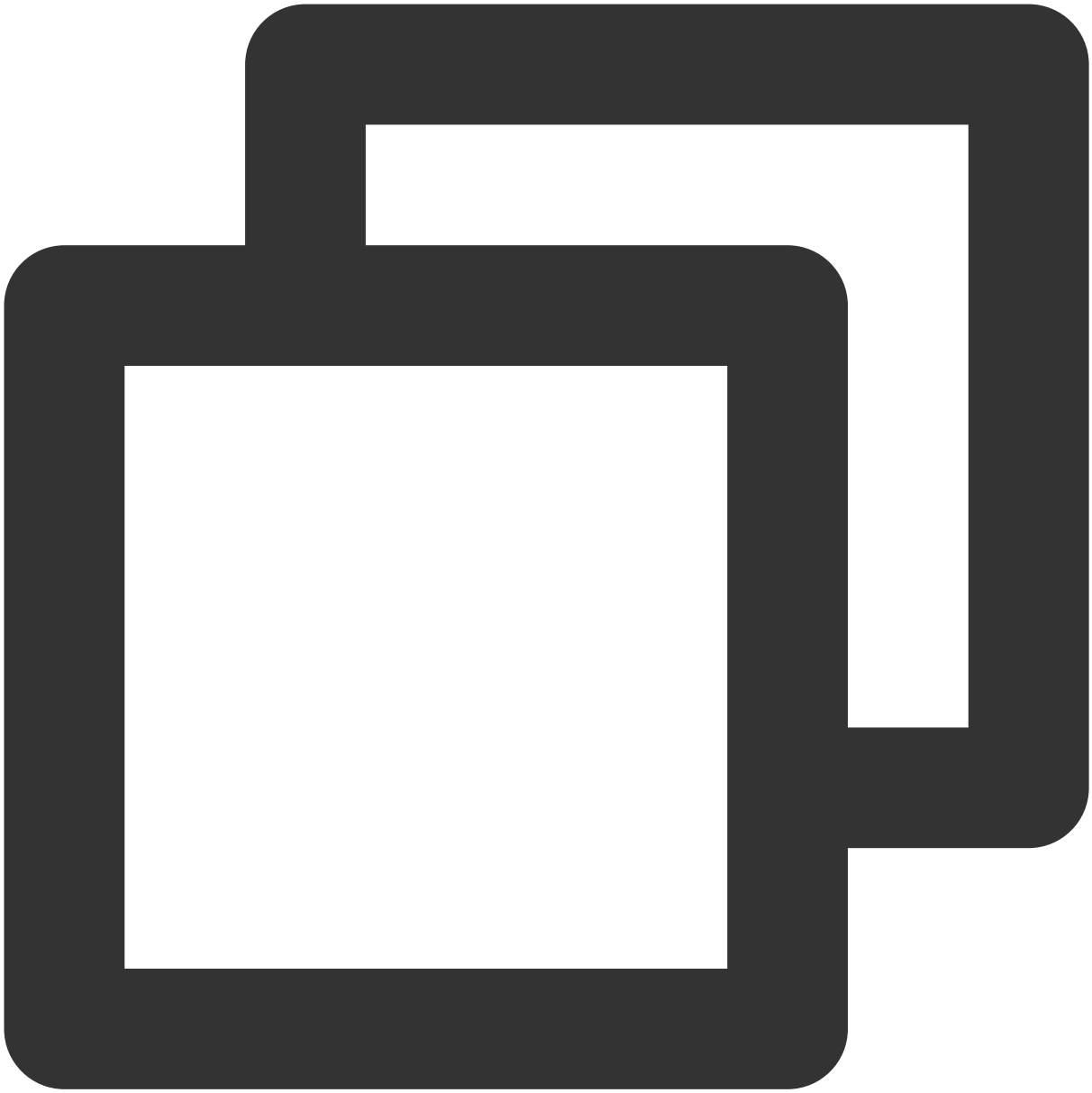
```
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                                       options:NSJSONReadingMutableContainers
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                                error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Initialize SDK: Width and height are the width and height of the texture respect
self.xMagicKit = [[XMagic alloc] initWithRenderSize:CGSizeMake(width,height) assets
```

3. Set the video data callback for third-party beauty features. Pass the results of the beauty SDK processing each frame of data into the TRTC SDK for rendering processing.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Set the video data callback for third-party beauty features in the TRTC SDK
[self.trtcCloud setLocalVideoProcessDelegete:self pixelFormat:TRTCVideoPixelFormat_

#pragma mark - TRTCVideoFrameDelegate

// Construct the YTProcessInput and pass it into the SDK for rendering processing
- (uint32_t)onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVi
    if (!self.xMagicKit) {
        [self buildBeautySDK:srcFrame.width and:srcFrame.height texture:srcFrame.te
        self.heightF = srcFrame.height;
        self.widthF = srcFrame.width;
```

©2013-2022 Tencent Cloud. All rights reserved.

```
    }
    if(self.xMagicKit!=nil && (self.heightF!=srcFrame.height || self.widthF!=srcFra
       self.heightF = srcFrame.height;
       self.widthF = srcFrame.width;
       [self.xMagicKit setRenderSize:CGSizeMake(srcFrame.width, srcFrame.height)];
    }
    YTProcessInput *input = [[YTProcessInput alloc] init];
    input.textureData = [[YTTextureData alloc] init];
    input.textureData.texture = srcFrame.textureId;
    input.textureData.textureWidth = srcFrame.width;
    input.textureData.textureHeight = srcFrame.height;
    input.dataType = kYTTextureData;
    YTProcessOutput *output = [self.xMagicKit process:input withOrigin:YtLightImage
    dstFrame.textureId = output.textureData.texture;
    return 0;
}
```

Note:

Steps 1 and 2 vary depending on the different third-party beauty products, while **Step 3** is a **general and important step** for integrating third-party beauty features into TRTC.

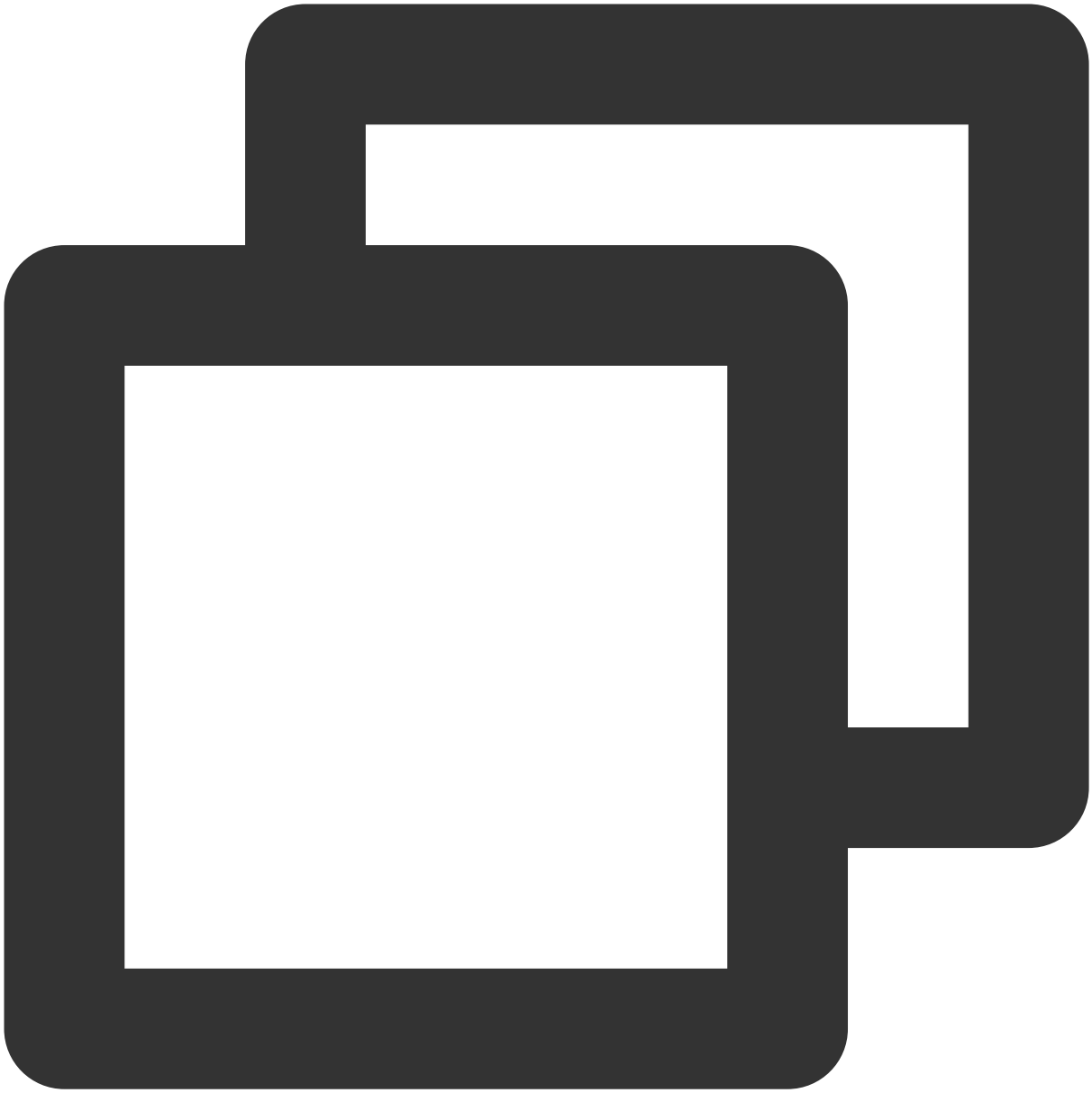
For scenario-specific integration guidelines of beauty effects, see Integrating Special Effect into TRTC SDK. For guidelines on integrating beauty effects independently, see Integrating Special Effect SDK.

Window Size Switching

In TRTC, there are many APIs that require you to control the video screen. All these APIs require you to specify a video rendering control.

1. If your business involves scenarios of switching display zones, you can use the TRTC SDK to update the local preview screen and update the remote user's video rendering control feature.

©2013-2022 Tencent Cloud. All rights reserved.



```
// Update local preview screen rendering control
[self.trtcCloud updateLocalView:self.previewView];

// Update the remote user's video rendering control
[self.trtcCloud updateRemoteView:self.previewView streamType:TRTCVideoStreamTypeBig
```

Note:

`streamType` only supports `TRTCVideoStreamTypeBig` and `TRTCVideoStreamTypeSub` .

Offline Push Message

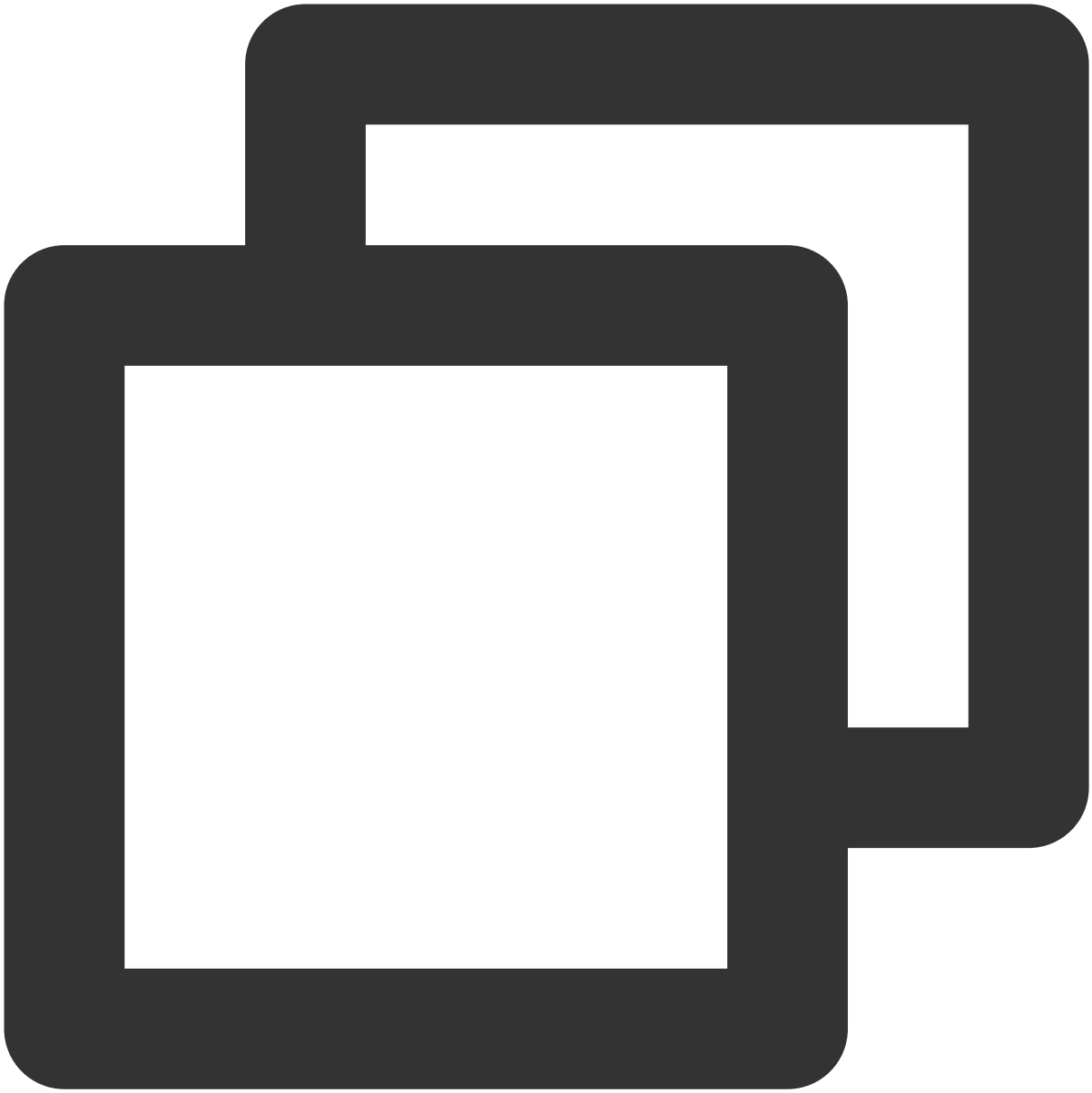
©2013-2022 Tencent Cloud. All rights reserved.

In audio/video call scenarios, the offline push message feature is usually necessary, allowing the called user's App to receive new incoming call messages even when it's not online. For detailed guidance on integrating offline push, see Offline Message Push. Below, we will focus on explaining the implementation of step 5: Send Offline Push Message, and step 6: Parse Offline Push Messages.

Send Offline Push Message

When sending a call invitation using invite, you can set offline push parameters through V2TIMOfflinePushInfo. By calling ext of V2TIMOfflinePushInfo to set custom ext data, when the user receives an offline push message to start the App, they can obtain the ext field in the system callback, and then redirect to the specified UI interface based on the content of the ext field.

©2013-2022 Tencent Cloud. All rights reserved.



```
NSDictionary *dic = @{
    @"cmd": @"av_call",
    @"msg": @{
        // Specify the call type (video call, audio call)
        @"callType": @"videoCall",
        // Specify the TRTC room ID (caller can generate it randomly)
        @"roomId": @"xxxRoomId",
    },
};
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:dic options:NSJSONWritin
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8Strin
```

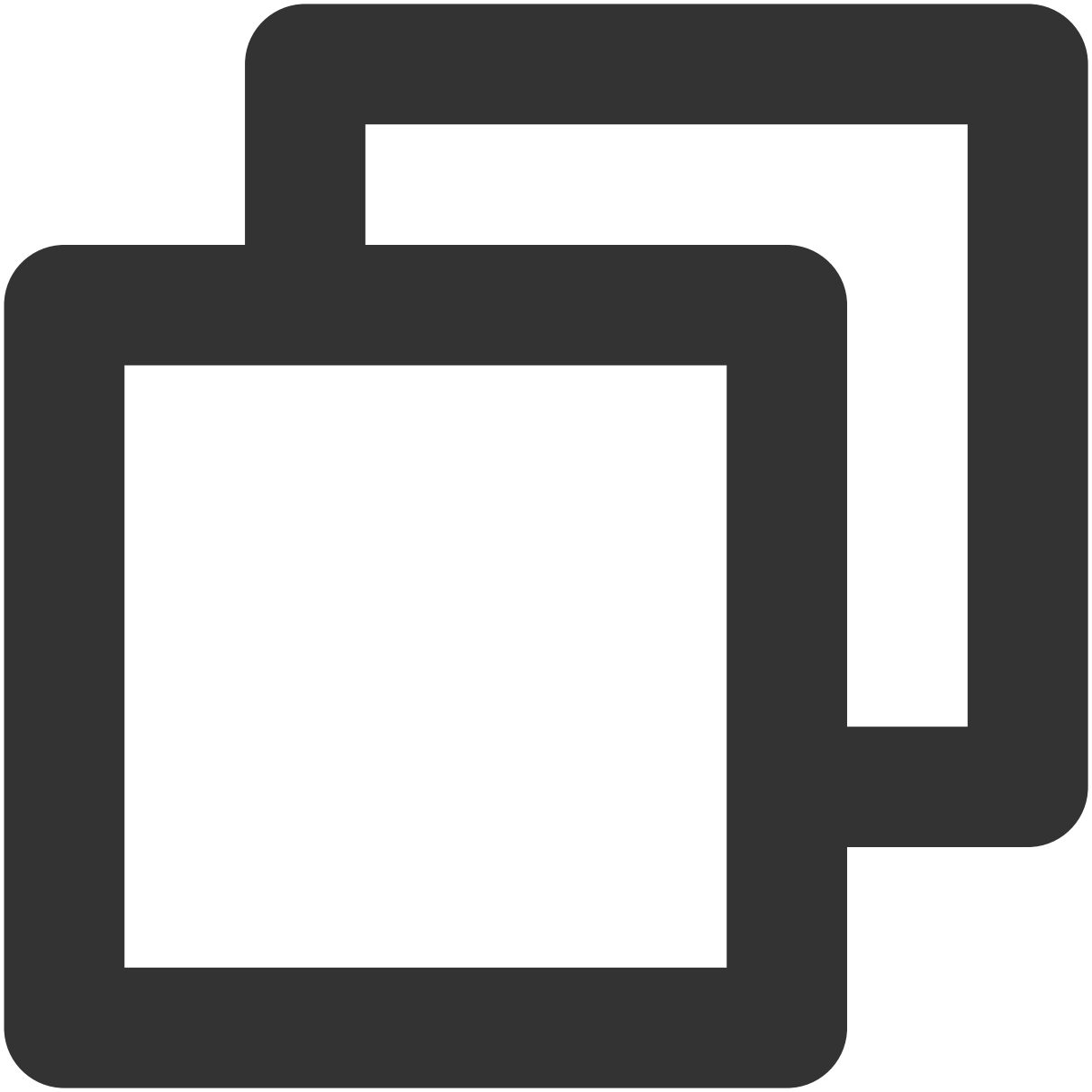
©2013-2022 Tencent Cloud. All rights reserved.

```
V2TIMOfflinePushInfo *pushInfo = [[V2TIMOfflinePushInfo alloc] init];
pushInfo.title = self.nickName;
pushInfo.desc  = @"You have a new call invitation";
NSDictionary *ext = @{
            @"entity" : @{
            @"action" : @1,
            @"content" : jsonString,
            @"sender" : self.senderId,
            @"nickname" : self.nickName,
            @"faceUrl" : faceUrl,
            }
        };
NSData *data = [NSJSONSerialization dataWithJSONObject:ext options:NSJSONWritingPre
pushInfo.ext = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
pushInfo.iOSSound = @"phone_ringing.mp3";
// Below are fields compatible with Android, and need to be filled in
pushInfo.AndroidOPPOChannelID = @"tuikit";
pushInfo.AndroidSound = @"phone_ringing";
pushInfo.AndroidHuaWeiCategory = @"IM";
pushInfo.AndroidVIVOCategory = @"IM";

[[V2TIMManager sharedInstance] invite:@"receiverId" data:jsonString onlineUserOnly:
    // Successfully send call invitation signaling
} fail:^(int code, NSString *desc) {
    // Failed to send call invitation signaling
}];
```

Parse Offline Push Messages

When a user receives an offline push message to start the App, they can obtain the `ext` field in the `AppDelegate -> didReceiveRemoteNotification` system callback, and then redirect to the specified UI interface based on the content of the `ext` field.

©2013-2022 Tencent Cloud. All rights reserved.



```
// After starting the APP, you will receive the following callbacks
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDi
fetchCompletionHandler:(void (^)(UIBackgroundFetchResult result))completionHandler
    // Parse push extension fields
    if ([userInfo[@"ext"]) {
        // Redirect to the specified UI interface
    }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Exception Handling

TRTC exception error handling

When the TRTC SDK encounters an unrecoverable error, the error is thrown in the `onError` callback. For details, see Error Code Table.

UserSig related

UserSig verification failure leads to room-entering failure. You can use the UserSig tool for verification.

Enumeration	Value	Description
ERR_TRTC_INVALID_USER_SIG	-3320	Room entry parameter userSig is incorrect. Check if `TRTCParams.userSig` is empty.
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig verification failed. Check if the parameter `TRTCParams.userSig` is filled in correctly or has expired.

Room entry and exit related

If room entry is failed, you should first verify the correctness of the room entry parameters. It is essential that the room entry and exit APIs are called in a paired manner. This means that, even in the event of a failed room entry, the room exit API must still be called.

Enumeration	Value	Description
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	Room entry request timed out. Check if your internet connection is lost or if a VPN is enabled. You may also attempt to switch to 4G for testing.
ERR_TRTC_INVALID_SDK_APPID	-3317	Room entry parameter sdkAppId is incorrect. Check if `TRTCParams.sdkAppId` is empty
ERR_TRTC_INVALID_ROOM_ID	-3318	Room entry parameter roomId is incorrect.Check if `TRTCParams.roomId` or `TRTCParams.strRoomId` is empty. Nnote that roomId and strRoomId cannot be used interchangeably.
ERR_TRTC_INVALID_USER_ID	-3319	Room entry parameter userId is incorrect. Check if `TRTCParams.userId` is empty.
ERR_TRTC_ENTER_ROOM_REFUSED	-3340	Room entry request was denied. Check if `enterRoom` is called consecutively to enter rooms with the same ID.

©2013-2022 Tencent Cloud. All rights reserved.

Device related

Errors for related monitoring devices. Prompt the user via UI in case of relevant errors.

Enumeration	Value	Description
ERR_CAMERA_START_FAIL	-1301	Failed to enable the camera. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_MIC_START_FAIL	-1302	Failed to open the mic. For example, if there is an exception for the camera's configuration program (driver) on a Windows or Mac device, you should try disabling then re-enabling the device, restarting the machine, or updating the configuration program.
ERR_CAMERA_NOT_AUTHORIZED	-1314	The device of camera is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_MIC_NOT_AUTHORIZED	-1317	The device of mic is unauthorized. This typically occurs on mobile devices and may be due to the user having denied the permission.
ERR_CAMERA_OCCUPY	-1316	The camera is occupied. Try a different camera.
ERR_MIC_OCCUPY	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.

Offline push cannot be received for normal messages

First, check whether the app runtime environment is the same as the certificate environment. Otherwise, offline push messages will not be received.
Secondly, check if the app and certificate environment are set to production. If it is a development environment, applying for a `deviceToken` from Apple may fail. This problem is not found in the production environment, you can switch to the production environment for testing.

Offline push cannot be received for custom messages

The offline push for custom messages is different from that for normal messages. As we cannot parse the content of custom messages, we cannot determine the push's content. Therefore, by default, there is no offline push. If you need an offline push, you need to set the desc field in offlinePushInfo when using sendMessage, and the desc information will by default be displayed during the push.

©2013-2022 Tencent Cloud. All rights reserved.

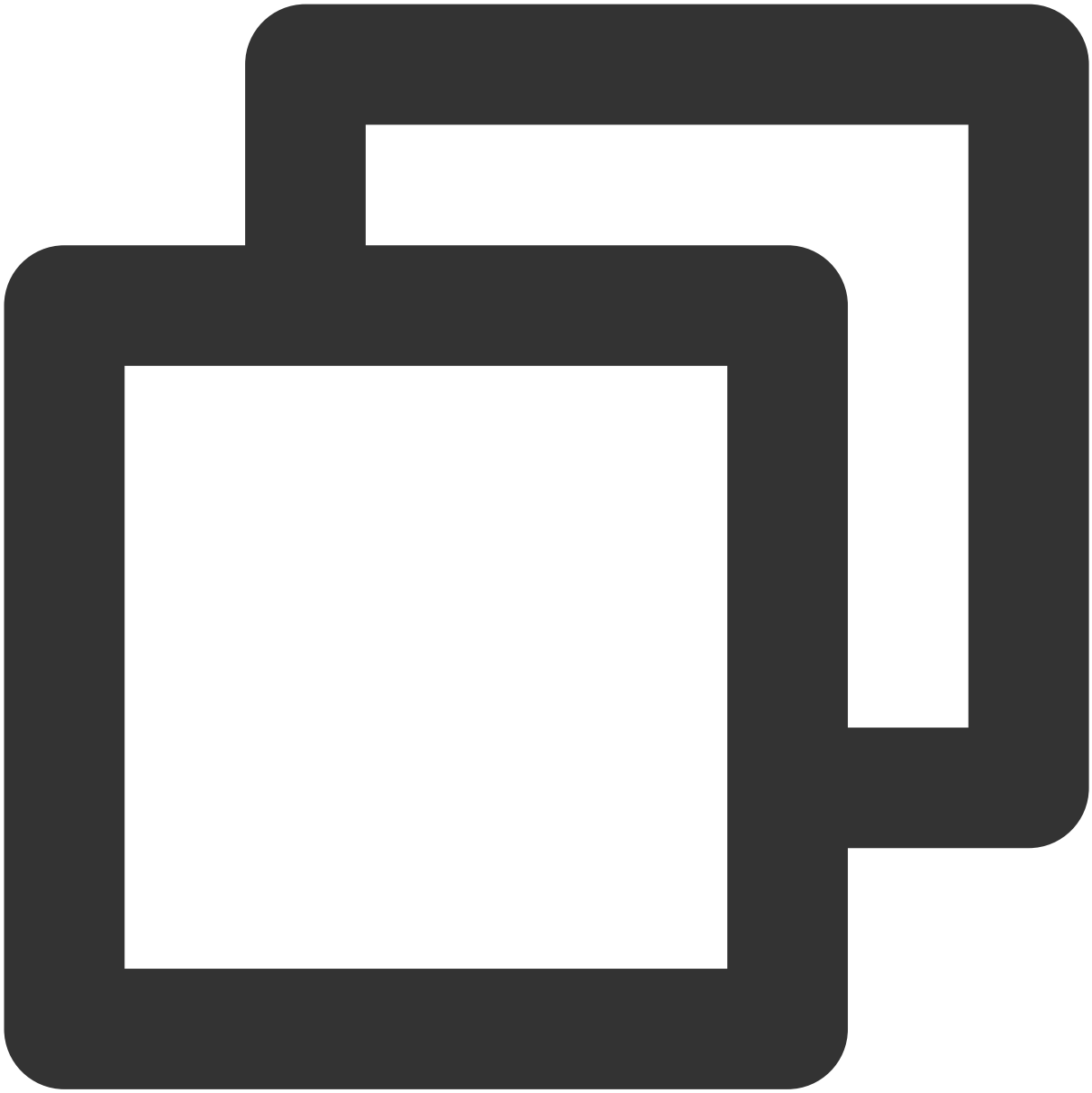
Disable the reception of offline push messages

To disable the reception of offline push messages, you can set the `config` parameter of setAPNS API to `nil`. This feature is supported starting from version 5.6.1200.

Failed to receive push and a bad devicetoken error was reported in the background

The Apple deviceToken depends on the current compiling environment. If the certificate ID and token used to log in to IMSDK and upload the deviceToken to the cloud platform are inconsistent, it will result in an error.
If you use the Release environment to compile, the `-application:didRegisterForRemoteNotificationsWithDeviceToken:` callback will return a production environment token. At this time, the businessID needs to be set to the Certificate ID of the production environment.
If you use the Debug environment to compile, the `-application:didRegisterForRemoteNotificationsWithDeviceToken:` callback will return a development environment token. At this time, the businessID needs to be set to the Certificate ID of the development environment.

©2013-2022 Tencent Cloud. All rights reserved.



```
V2TIMAPNSConfig *confg = [[V2TIMAPNSConfig alloc] init];
/* You need to register a developer certificate with Apple, download and generate t
// Push certificate ID
confg.businessID = sdkBusiId;
confg.token = self.deviceToken;
[[V2TIMManager sharedInstance] setAPNS:confg succ:^{
     NSLog(@"%s, succ, %@", __func__, supportTPNS ? @"TPNS": @"APNS");
} fail:^(int code, NSString *msg) {
     NSLog(@"%s, fail, %d, %@", __func__, code, msg);
}];
```

©2013-2022 Tencent Cloud. All rights reserved.

In the iOS development environment, deviceToken is occasionally not returned for registration or APNs fails to request a token

This problem is caused by instability of APNs. You can resolve the problem in the following ways:

1. Insert a SIM card into the phone and use the 4G network to test.
2. Uninstall and reinstall the application, restart the application, or shut down and restart the phone.
3. Use a package for the production environment.
4. Use another phone with iOS system.

©2013-2022 Tencent Cloud. All rights reserved.