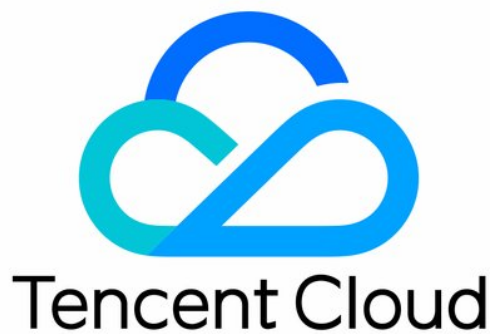


Tencent Real-Time Communication

Voice Chat Room (with UI)

Product Documentation



Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

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

Trademark Notice



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

Service Statement

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

Contents

Voice Chat Room (with UI)

- Overview (TUILiveKit)

- Activating the Service (TUILiveKit)

- Run Demo (TUILiveKit)

 - Android

 - iOS

 - Flutter

- Integration (TUILiveKit)

 - Android

 - iOS

 - Flutter

- Customizable Interface (TUILiveKit)

- Live Streaming and Listening (TUILiveKit)

 - Android

 - iOS

 - Flutter

- Microphone Management (TUILiveKit)

- Room List (TUILiveKit)

- Follow Anchors (TUILiveKit)

- Interactive Bullet Comments (TUILiveKit)

- Interactive Gifts (TUILiveKit)

- Gift Effects (TUILiveKit)

- Client APIs (TUICallKit)

 - Android

 - UIKit API

 - VoiceRoomKit

 - SeatGridView

 - Engine API

 - API Overview

 - TUIRoomEngine

 - TUIRoomObserver

 - TUIRoomDeviceManager

 - TUILiveListManager

 - TUILiveConnectionManager

 - TUICommonDefine

TUIRoomDefine

iOS

UIKit API

VoiceRoomKit

SeatGridView

Engine API

API Overview

TUIRoomEngine

TUIRoomObserver

TUIRoomDeviceManager

TUILiveListManager

TUILiveConnectionManager

TUICommonDefine

TUIRoomDefine

Flutter

UIKit API

SeatGridWidget

Engine API

API Overview

TUIRoomEngine

TUIRoomEngineObserver

TUIRoomDeviceManager

TUILiveListManager

Type Definition

API-Example

Voice Room (SeatGridView)

Integration and Login

Going Live

Listening

Seat Management

Microphone Management

Seat Layout

Customizing a Seat

Server APIs (TUILiveKit)

REST API

RESTful API Overview

RESTful API List

Room Related

Create a Room

Update the Room Information

Get the Room Information

Retrieve the Live Streaming List

Destroy a Room

User-Related Matters

Member Banishment

Retrieve the Ban List

Retrieve the List of Administrators

Lifting a Member's Ban

Modify Administrator

Error Codes (TUILiveKit)

Release Notes (TUILiveKit)

iOS

Android

FAQs (TUILiveKit)

iOS

Android

Voice Chat Room (with UI)

Overview (TUILiveKit)

Last updated : 2025-04-21 09:44:46

Overview

TUILiveKit is a product suitable for video interactive live streaming and voice chat scenarios, such as social entertainment, game interaction, real-time duet, voice radio, etc. By integrating this product, with just three steps and within 30 minutes, you can add features like voice chat rooms, interactive microphone connection, gift sending, room management, microphone position management, etc., to your application and quickly launch your service. The basic feature showcase is as follows:

Anchor

Voice Live Start Page	Voice Live Home Page

Audience

Voice Live Not on Seat	Take the Seat in Voice Live Streaming

Seat Position Management

Invite to Take the Seat Management	Apply to Take the Seat Management	Seat Position Management

Advanced features

Background	Sound Effects & Voice Chan	Music	Interactive Gifts (Fullscreen Gifts)

Supported Platform

Platform	Android	iOS	Flutter
Supported			
Supported Languages/Frameworks	Java Kotlin	Swift Objective-C	Dart

Features

Basic Feature	Advanced Feature	Feature Advantages
HD Live Streaming Voice Chat Room Live Viewing Viewer Mic Connection View Viewer List Member Management Seat Management	Chat Barrage Heartbeat Like Interactive Gifts (Fullscreen Gifts) Sound Effects & Voice Changer	Comprehensive UI Interaction Professional Live Streaming 3A Algorithm, Better Audio Quality Rich REST API

Trying It Online

Platform	Android	iOS	Flutter
Demo Integration	Github: Andorid	Github: iOS	Github: Flutter

Exchange and Feedback

If you have any requirements or feedback, you can contact: info_rtc@tencent.com.

Activating the Service (TUILiveKit)

Last updated : 2025-04-21 10:02:46

Activate Trial Service

To give you a better experience with Interactive Live Streaming features, we provide a 14-day trial version for each SDKAppID for free (the trial version does not include additional call duration). Each SDKAppID can be tried for free twice, with each trial period lasting 14 days. Additionally, the total number of trials for all SDKAppIDs under one account is 10 times.

1. Log in to the [Tencent RTC console](#), and click **Create Application**.

2. In the creation popup, enter **Application Name**, select **Live**, and choose the appropriate **Deployment Region**, then click **Create** to complete the creation of the trial version.

Note :

Live currently only supports Singapore and Silicon Valley deployment region. If you need other deployment region, please [contact us](#).

3. After creation, you can view detailed information about your app on the current page, such as

`SDKAppID` , `SDKSecretKey` , and more. You can go to the [Tencent RTC console](#) to check application version information and refer to [Quick Integration \(TUILiveKit\)](#) for integration.

Purchasing the official version

Before officially going live, you need to purchase the official version of the Live Monthly Subscription Package. For price comparison and feature details of the Live Monthly Subscription Package, please refer to [Tencent RTC Live Monthly Packages](#). The package purchase process is as follows:

1. Visit the [Live Purchase Page](#), select the application (SDKAppID) and version you wish to purchase, **and it is recommended to enable auto-renewal to avoid affecting your business usage**. After confirming the purchase information, check the agreement and click **Subscribe now**.

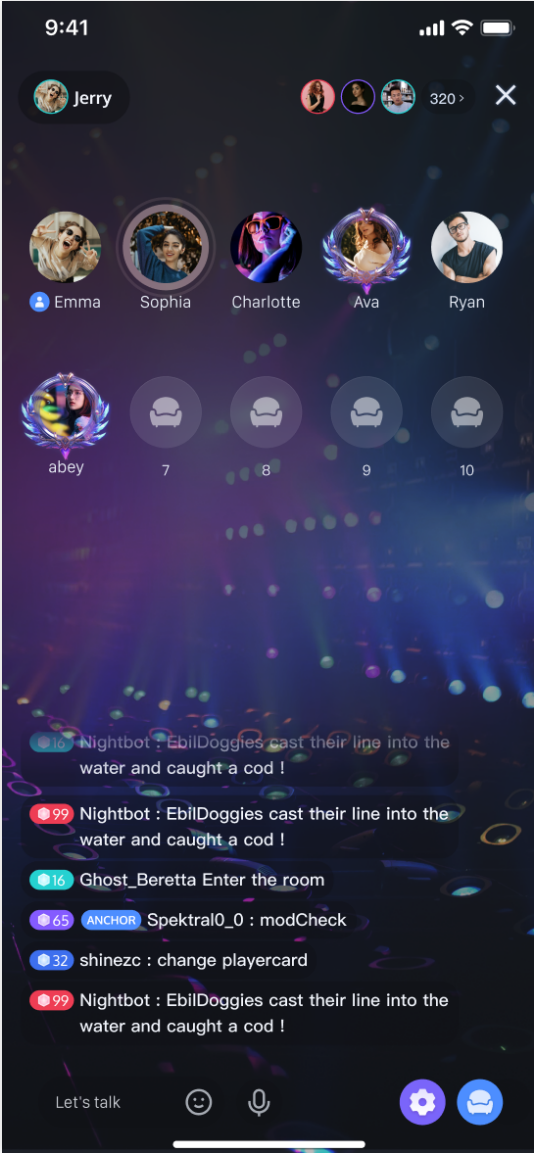
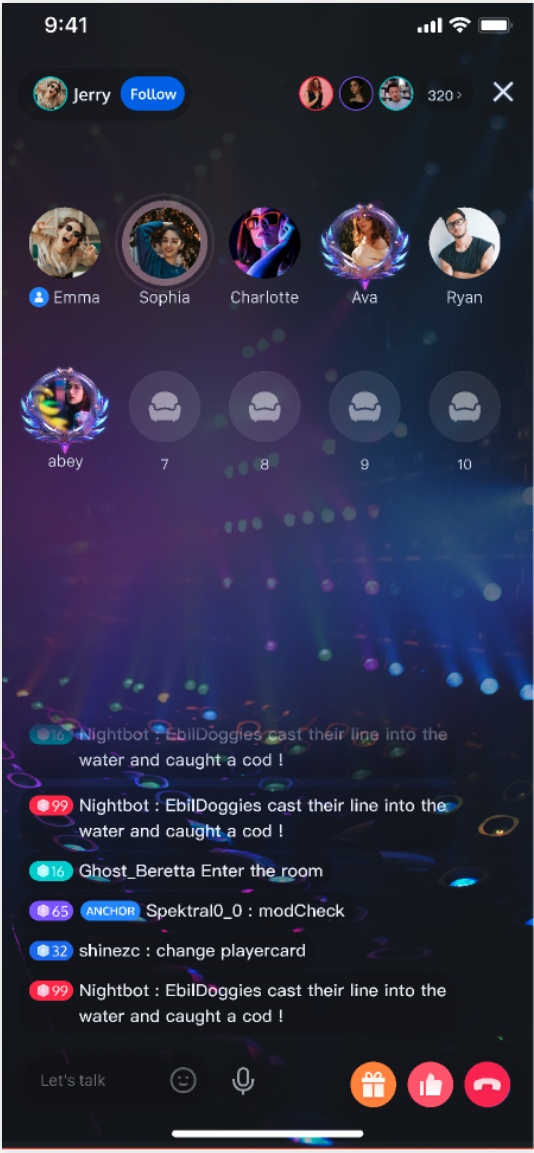
2. Go to the Order Confirmation Page to confirm product information and complete the payment.

Run Demo (TUILiveKit)

Android

Last updated : 2024-09-10 16:32:50

This document will guide you on how to quickly run the Voice Live Streaming Demo. By following this document, you can run the Demo within 10 minutes and eventually experience a complete UI interface for the Voice Live Streaming feature.

Host: During live streaming	Audience: Mic Connect
	

Environment preparations

Android 5.0 (SDK API level 21) or above.

Gradle 4.2.1 or later.

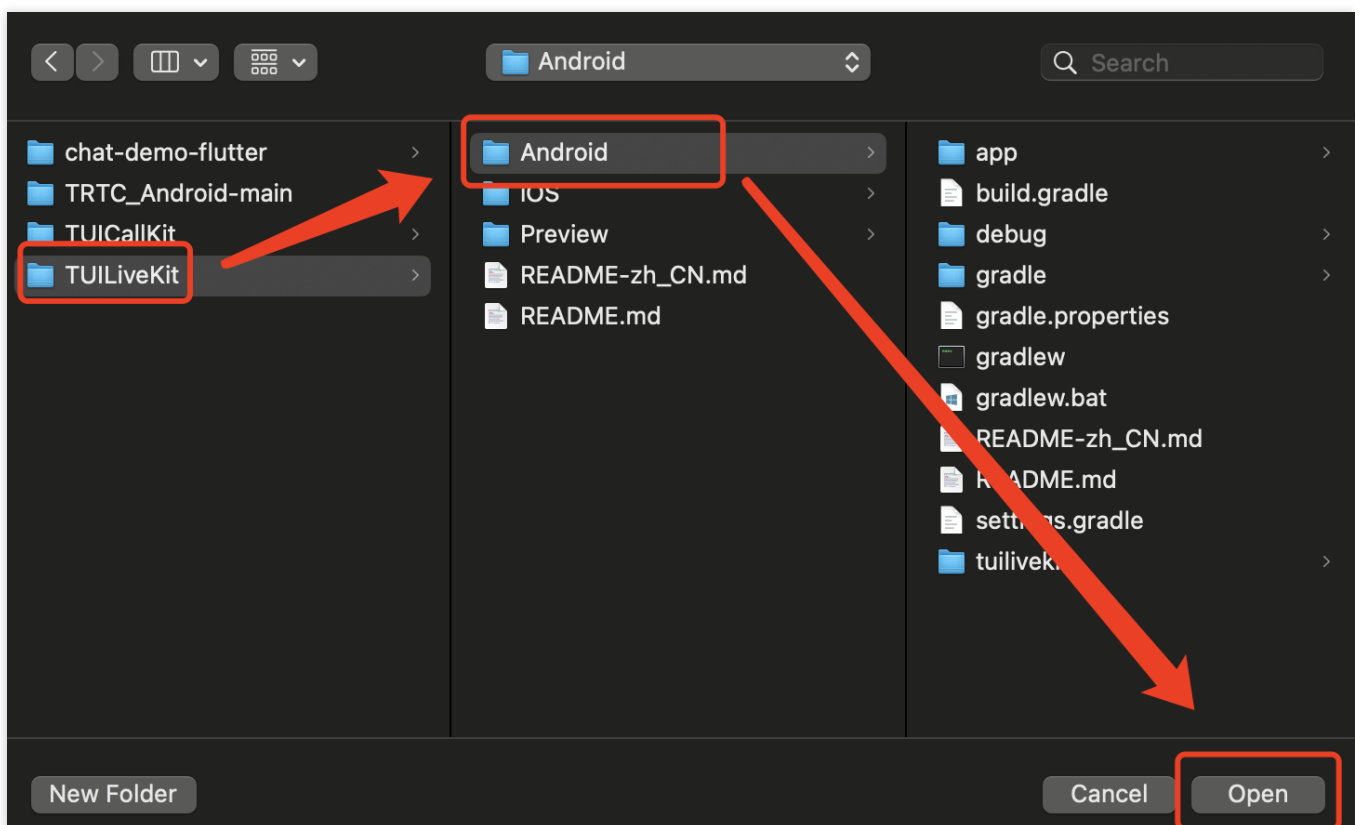
Two Android 5.0 or newer devices.

Step 1: Download the Demo

1. Download the source code of [TUILiveKit Demo](#) from GitHub, or directly run the following command in the terminal:

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

2. Open the TUILiveKit **Android** project through Android Studio:



Step 2: Configure the Demo

1. [Activate the TRTC service](#), obtain the SDKAppID and SDKSecretKey.

Application Overview

- LiveKit

How to Build

Ready to start building?

You can choose to start here or [talk to our experts](#)

Integration Docs

Help you go through, step by step

Run Sample Code

Download and run code within minutes

Try Web Demo

Test crystal-clear video online

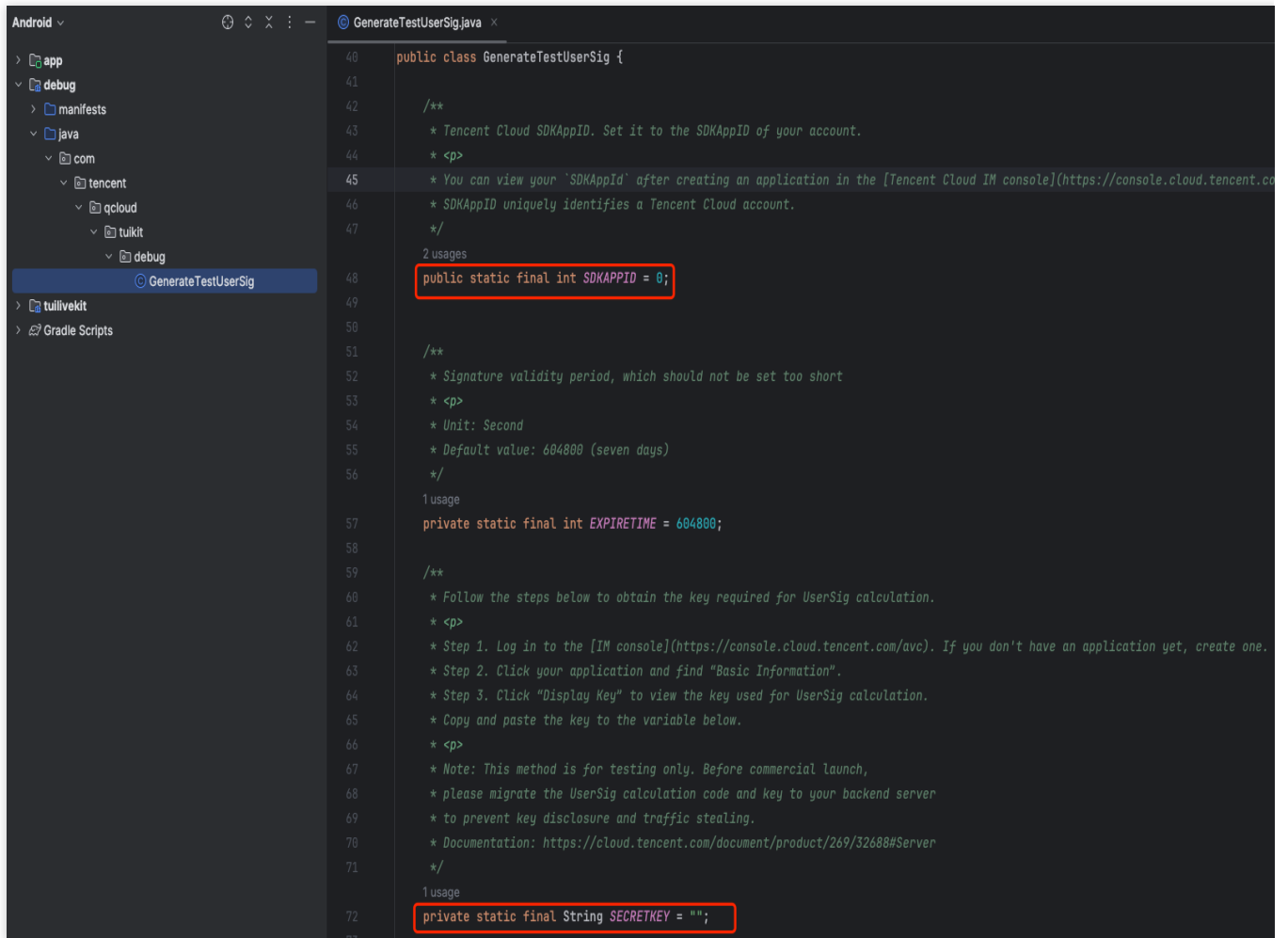
Basic Information

Application name	LiveKit	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-26 16:10:34
Description	--	Region	Singapore
Status	Enabled	Service Availability Zone	Global

Advanced Features

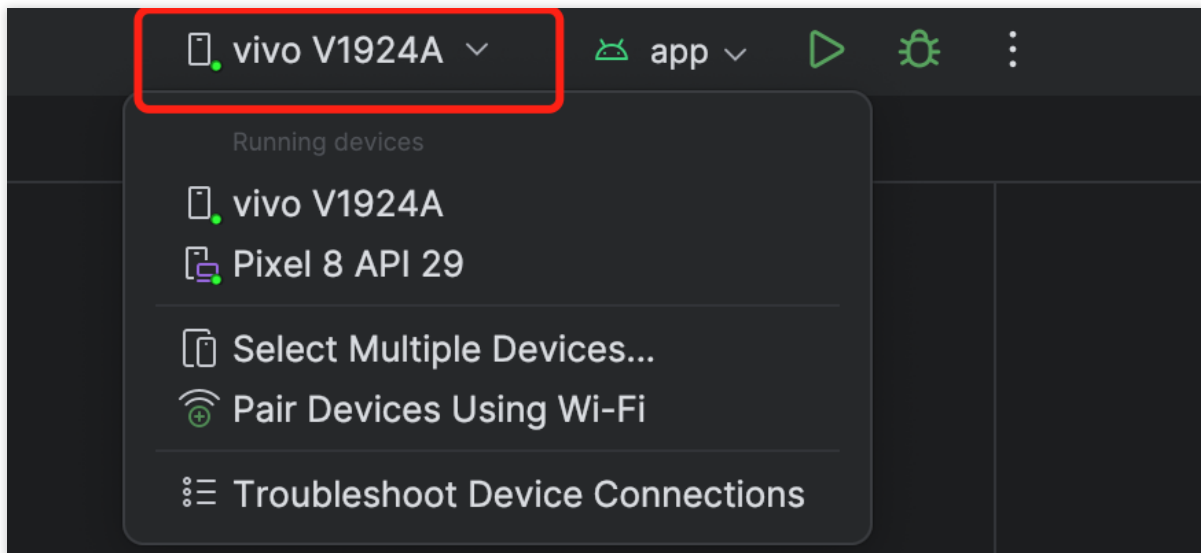
On-cloud recording	Disabled
Relay to CDN	Disabled
Callbacks	Disabled
Advanced permission control	Disabled

2. Open the `Android/debug/src/main/java/com/tencent/qcloud/tuikit/debug/GenerateTestUserSig.java` file, and enter the SDKAppID and SDKSecretKey obtained when [Activate the service](#):



Step 3: Running the Demo

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



2. After selecting, click run to deploy the TUILiveKit Android Demo on the target device.



3. Once the Demo successfully runs on the device, you can initiate and watch the live streaming by following the steps below.

Step 4: Voice Live Streaming

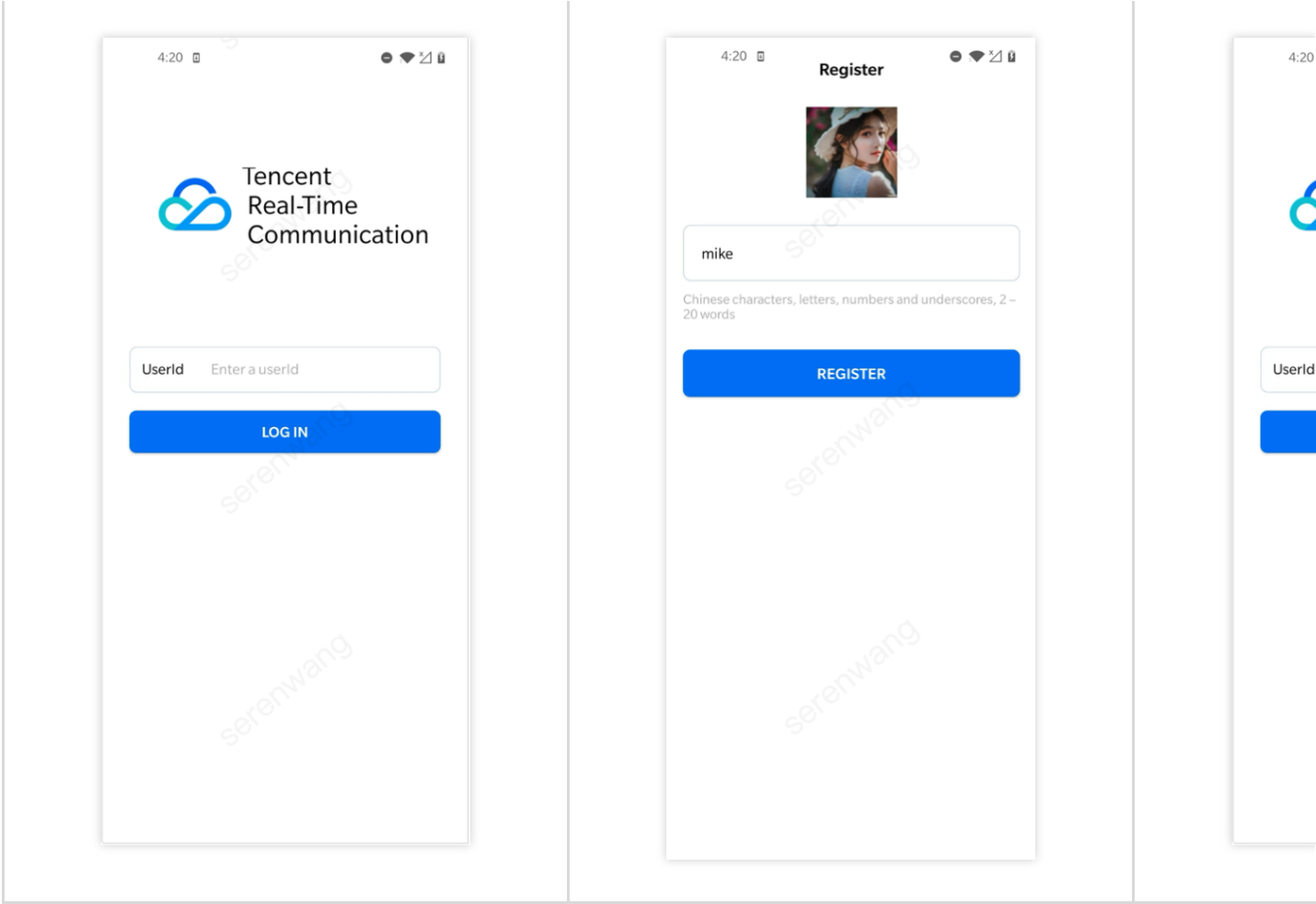
Note :

In order to allow you to experience the complete video live broadcast process, please log in two users on two devices to use the Demo, one as the host and the other as the audience.

1. Log in & Signup

Please enter your `UserId` in the User ID field. If your current User ID has not been used before, you will be taken to the Registration page where you can set an avatar and nickname for yourself.

Anchor (mike)		Audiece (vir



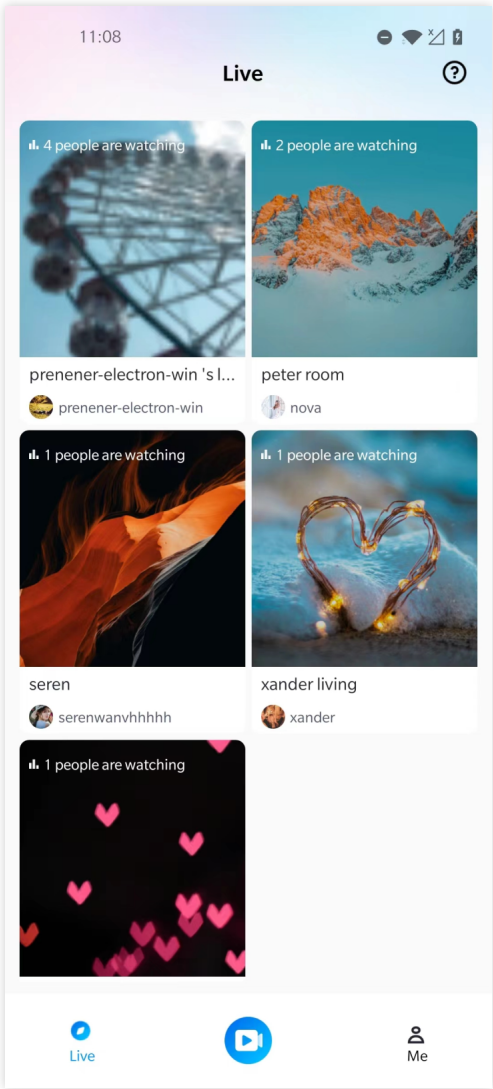
Note :

Try to avoid setting your User ID to simple strings like "1", "123", "111", as TRTC does not support the same User ID being logged into from multiple devices. Such User IDs like "1", "123", "111" are easily occupied by your colleagues during collaborative development, leading to login failures. Therefore, we recommend setting highly recognizable User IDs while debugging.

2. The anchor starts the Voice Live Room

Click the **Start Live** button in the middle of the bottom of the homepage to enter the broadcast preview page, and then click **Start Live** to start the live broadcast.

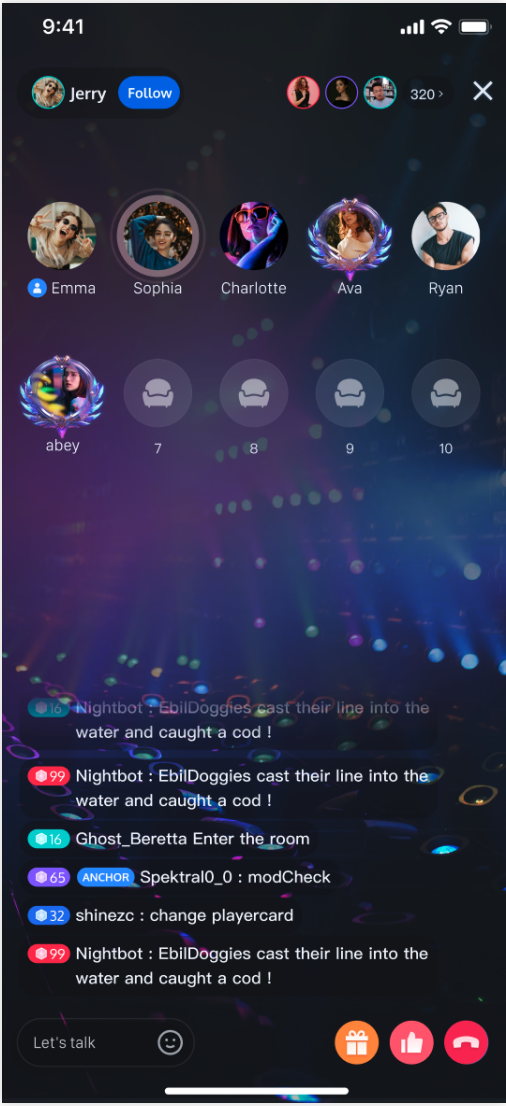
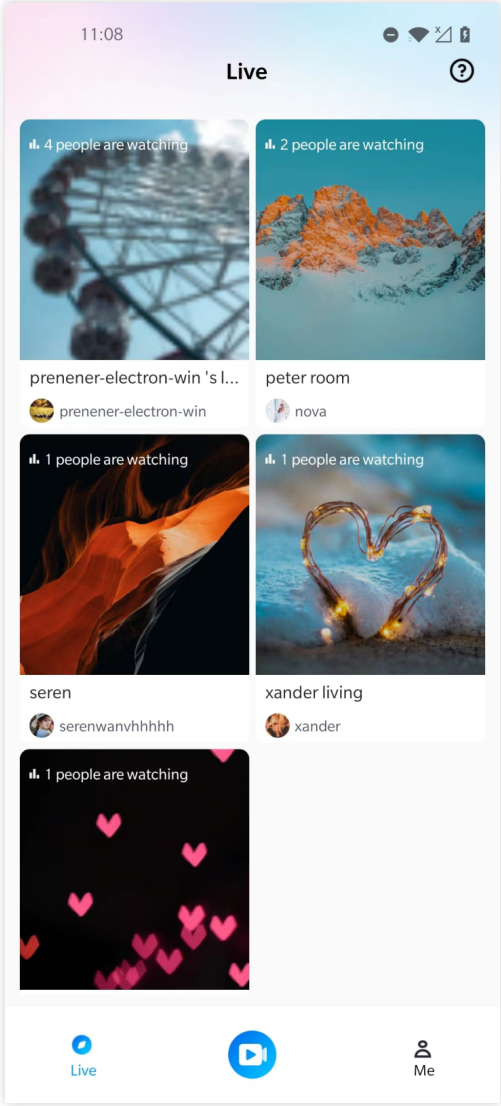
Anchor (Before entering)	Anchor (Preview)



3. Viewers join the Voice Live Room.

Click on any room in the live broadcast list to enter the live broadcast room.

Audience (Before entering)	Audience (After entering)



iOS

Last updated : 2024-09-10 16:32:50

This document will guide you on how to quickly run through the Voice Chat Room Demo. By following this documentation, you can run through the Demo in 10 minutes and eventually experience a Voice Live Streaming feature with a complete UI interface.

Host: During live streaming	Audience: Mic Connect

Environment preparations

Xcode 15 or later.

iOS 13.0 or later.

To install the CocoaPods environment, please click [Viewing](#).

If you encounter any problems with access and use, please refer to [FAQs](#).

Step 1: Download the Demo

1. Download the source code of [TUILiveKit Demo](#) from GitHub, or directly run the following command in the terminal:

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

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

```
cd TUILiveKit/iOS/Example
```

3. Load the dependent libraries:

```
pod install
```

Note:

If you have not installed CocoaPods yet, you can refer to [this link](#) to learn how to install it.

Step 2: Configure Demo

1. [Activate the TRTC service](#), obtain the SDKAppID and SDKSecretKey.

Application Overview

- LiveKit

How to Build

Ready to start building?

You can choose to start here or [talk to our experts](#)

Integration Docs

Help you go through, step by step

Run Sample Code

Download and run code within minutes

Try Web Demo

Test crystal-clear video online

Basic Information

Application name	LiveKit	SDKSecretKey	*****
SDKAppID		Creation time	2024-04-26 16:10:34
Description	--	Region	Singapore
Status	Enabled More	Service Availability Zone	Global

Advanced Features

More Feature

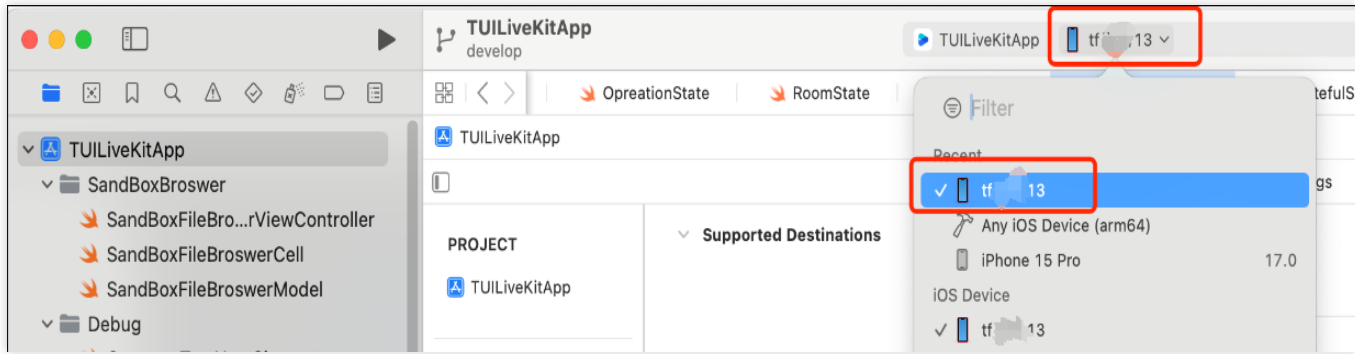
On-cloud recording	Disabled
Relay to CDN	Disabled
Callbacks	Disabled
Advanced permission control	Disabled

2. Open the `/iOS/Example/Debug/GenerateTestUserSig.swift` Document, file, and enter the SDKAppID and SDKSecretKey obtained when [Activate the service](#):

```
TUICallKitApp > Debug > GenerateTestUserSig > No Selection
10 import CommonCrypto
11 import zlib
12
13 /**
14  * Tencent Cloud SDKAppId, which needs to be replaced with the SDKAppId under your own account.
15  *
16  * Enter Tencent Cloud IM to create an application, and you can see the SDKAppId, which is the unique identifier used by Tencent Cloud to
17  */
18 let SDKAPPID: Int = 0
19
20 /**
21  * Signature expiration time, it is recommended not to set it too short
22  *
23  * Time unit: seconds
24  * Default time: 7 x 24 x 60 x 60 = 604800 = 7 days
25  */
26 let EXPIRETIME: Int = 604_800
27
28 /**
29  * Encryption key used for calculating the signature, the steps to obtain it are as follows:
30  *
31  * step1. Enter Tencent Cloud IM, if you do not have an application yet, create one,
32  * step2. Click "Application Configuration" to enter the basic configuration page, and further find the "Account System Integration" section.
33  * step3. Click the "View Key" button, you can see the encryption key used to calculate UserSig, please copy and paste it into the following
34  *
35  * Note: This solution is only applicable to debugging demos.
36  * Before going online officially, please migrate the UserSig calculation code and keys to your backend server to avoid traffic theft caused by
37  */
38 let SECRETKEY = ""
39
```

Step 3: Running the Demo

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



2. After selection, click run to deploy our Demo onto the target device.

3. After the Demo is successfully running on the device, you can start a Voice Live Streaming and join a Voice Live Streaming by following these steps.

Step 4: Voice Live Streaming

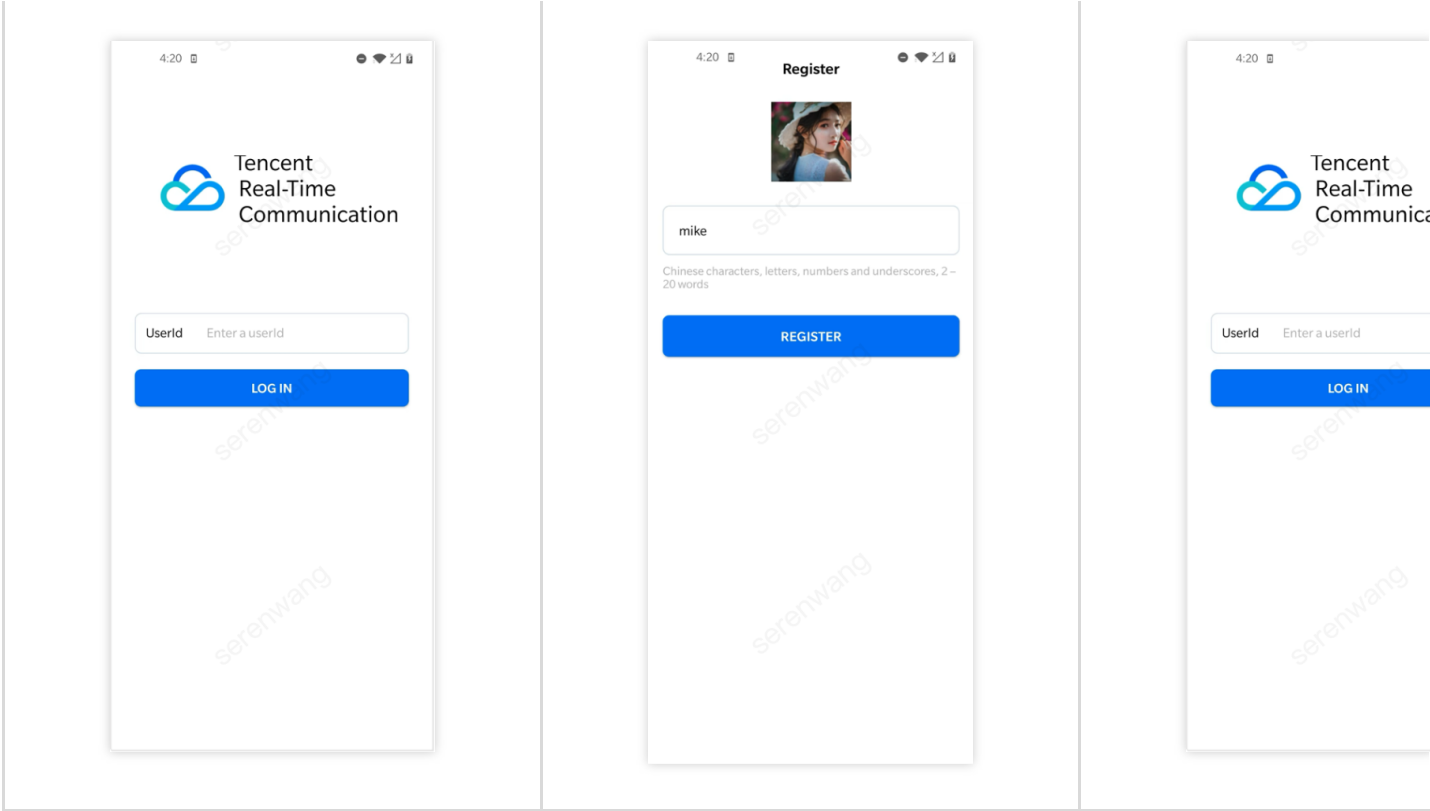
Note :

In order to allow you to experience the complete video live broadcast process, please log in two users on two devices to use the Demo, one as the host and the other as the audience.

1. Log in & Signup

Please enter your `UserId` in the User ID field. If your current User ID has not been used before, you will be taken to the Registration page where you can set an avatar and nickname for yourself.

Anchor (mike)		Audiece (vince)



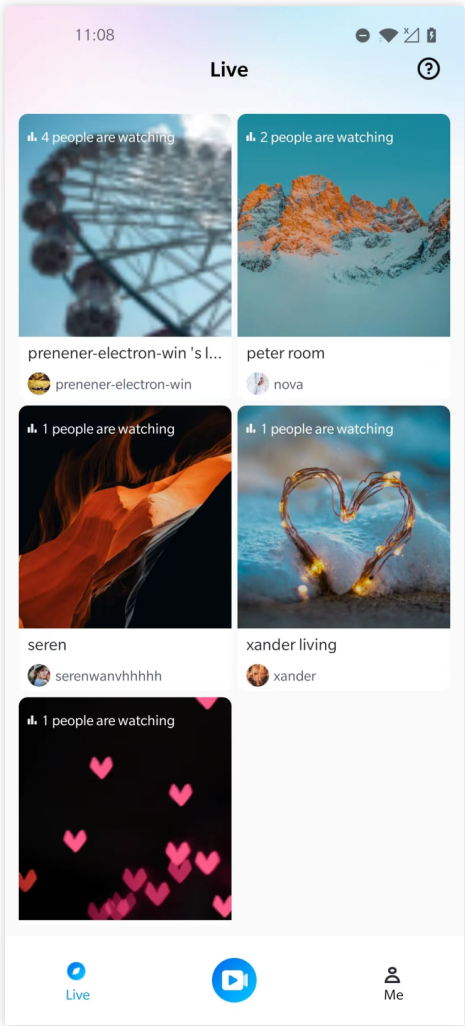
Note :

Try to avoid setting your User ID to simple strings like "1", "123", "111", as TRTC does not support the same User ID being logged into from multiple devices. Such User IDs like "1", "123", "111" are easily occupied by your colleagues during collaborative development, leading to login failures. Therefore, we recommend setting highly recognizable User IDs while debugging.

2. The anchor starts the Voice Live Room

Click the **Start Live** button in the middle of the bottom of the homepage to enter the broadcast preview page, and then click **Start Live** to start the live broadcast.

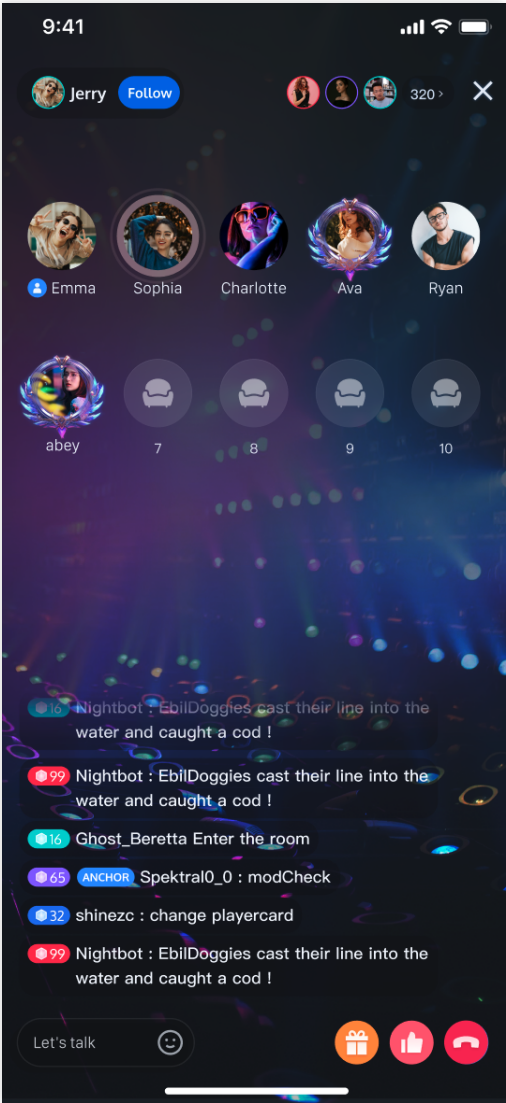
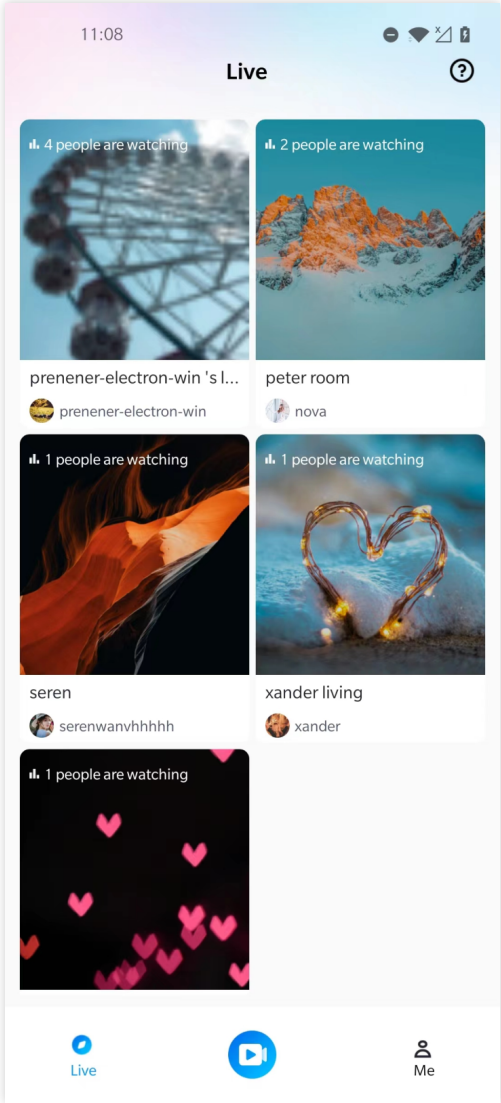
Anchor (Before entering)	Anchor (Preview)	Anc



3. Viewers join the Voice Live Room.

Click on any room in the live broadcast list to enter the live broadcast room.

Audience (Before entering)	Audience (After entering)



Flutter

Last updated : 2025-04-09 18:04:14

This document introduces how to quickly run through the Voice Chat Room Demo. Follow this documentation, and you can run through the Demo within 10 minutes and finally experience a voice chat room feature with a complete UI interface.

Anchor: Voice Room	Audience: Link Mic

Environment Preparation

Platform	Version
Flutter	Flutter 3.27.4 and higher versions. Dart 3.6.2 or higher version.
Android	Android Studio 3.5 and above versions. Android device running Android 5.0 and above versions.
iOS	Xcode 15.0 and above versions. Ensure that your project has a deemed valid developer signature.

Step 1: Download Demo

1. Download the [TUILiveKit Demo](#) source code from github, or run the following commands in command line:

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

2. Open the `TUILiveKit/Flutter/livekit/example` directory through Android Studio:

Step Two: Configure the Demo

1. Please see [Activate Service \(TUILiveKit\)](#), claim the trial version or activate the paid edition. After a successful application, you can obtain a 14-day trial version, try all features of TUILiveKit for free, and obtain `SDKAppID` and `SDKSecretKey`.
2. Open the `Flutter/example/lib/debug/generate_test_user_sig.dart` file and fill in the obtained corresponding `SDKAppID` and `SDKSecretKey`:

Step 3: Running through the Demo

1. Select the device where you want to run the Demo in the upper-right corner of Android Studio as shown below:
2. Click Run after selection to run the TUILiveKit Flutter Demo on the target device.
3. After the Demo has been successfully executed on the device, you can follow these steps to initiate live streaming and view live streams.

Start Voice Room

Notes:

To allow you to fully experience the voice chat room process, please log in two users on two devices respectively for the Demo. One acts as the anchor and the other as the audience.

1. Log in & Sign up

Please enter the ID in `User ID`. If your current UserID has never been used, you will enter the **registration** interface, where you can set your avatar and nickname for yourself.

Anchor (mike)		Audiece (vince)	

Note:

Try to avoid setting your UserID to simple characters such as "1", "123", "111". Since TRTC does not support multi-end login with the same UserID, when multiple people collaborate in development, UserIDs such as "1", "123", "111" are easily occupied by your colleagues, resulting in login failure. Therefore, we recommend that you set some highly recognizable UserIDs when debugging.

2. The anchor starts voice room.

Click the "Broadcast" button in the middle at the bottom of the home page to enter the broadcast preview page. Click **Go Live** again to start voice room.

Anchor: before Room Entry	Anchor: Preview	Anchor: Voice Room

3. Audience Joins Voice Room

Click any one of the rooms in the live list to enter the voice room.

Audience: before Entering Room	Audience: after Entering Room

Integration (TUILiveKit)

Android

Last updated : 2025-04-07 16:52:20

This article will guide you through the process of integrating the TUILiveKit component in a short time. Following this document, you will complete several key steps within an hour and ultimately obtain a voice room feature with a complete UI interface.

Anchor	Audience

Environment Preparations

Android 5.0 (SDK API level 21) or above.

Android Studio 7.0 or above.

Devices with Android 5.0 or above.

Step 1. Activate the service

Before using the Audio and Video Services provided by Tencent Cloud, you need to go to the Console and activate the service for your application. For detailed steps, refer to [Activate the service](#).

Step 2: Download TUILiveKit component

Clone/download the code in [Github](#) , and then copy the tuilivekit subdirectory in the Android directory to the same level directory as the app in your current project, as shown below:

Step 3: Project configuration

1. In the root directory of the project, add the jitpack repository URL to the `settings.gradle.kts` (or `settings.gradle`) file: Add the jitpack repository dependency (to download the SVG animation library for playing gifts, SVGAPlayer):

`settings.gradle.kts`

`settings.gradle`

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()

        // Add jitpack repository address
        maven { url = uri("https://jitpack.io") }
    }
}

dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()

        // Add the jitpack repository URL
        maven { url 'https://jitpack.io' }
    }
}
```

2. In the root directory of the project, add the following code to the `settings.gradle.kts` (or `settings.gradle`) file. It will import the tuilivekit component downloaded in [Step 2](#) into your current project:

`settings.gradle.kts`

`settings.gradle`

```
include(":tuilivekit:livekit")
include(":tuilivekit:component:barrage")

include ':tuilivekit:livekit'
include ':tuilivekit:component:barrage'
```

3. In the app directory, find the `build.gradle.kts` (or `build.gradle`) file and add the following code. It declares the dependency of the current app on the newly added tuilivekit component:

`build.gradle.kts`

`build.gradle`

```
api(project(":tuilivekit:livekit"))

api project(':tuilivekit:livekit')
```

Note:

The TUILiveKit project already has default dependencies on `TRTC SDK` , `IM SDK` , `tuiroomengine` , and the common library `tuicore` . Developers do not need to configure them separately. To upgrade the version, modify the `tuilivekit/build.gradle` file.

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

```
-keep class com.tencent.** { *; }
-keep class com.trtc.uikit.livekit.livestreamcore.** { *; }
-keep class com.trtc.uikit.livekit.component.gift.store.model.** { *; }
-keep class com.squareup.wire.** { *; }
-keep class com.opensource.svgaplayer.proto.** { *; }

-keep class com.tcmediax.** { *; }
-keep class com.tencent.** { *; }
-keep class com.tencent.xmagic.** { *; }
-keep class androidx.exifinterface.** {*; }
-keep class com.google.gson.** { *; }
# Tencent Effect SDK - beauty
-keep class com.tencent.xmagic.** { *; }
-keep class org.light.** { *; }
-keep class org.libpag.** { *; }
-keep class org.extra.** { *; }
-keep class com.gyailib.** { *; }
-keep class com.tencent.cloud.iai.lib.** { *; }
-keep class com.tencent.beacon.** { *; }
-keep class com.tencent.qimei.** { *; }
-keep class androidx.exifinterface.** { *; }
```

5. In the app directory, find the `AndroidManifest.xml` file and add

`tools:replace="android:allowBackup"` to the application node to override internal component settings with your own.

```
// app/src/main/AndroidManifest.xml

<application
    ...

    // Add the following configuration to override the settings in the dependent SD
    android:allowBackup="false"
    tools:replace="android:allowBackup">
```

Step 4. Log in

Add the following code to your project. It calls the relevant interfaces in TUICore to complete the TUI component log in to. This step is crucial; you can only use the features provided by TUILiveKit after successfully logging in.

Kotlin

Java

```
TUILogin.login(applicationContext,
    1400000001, // Please replace with the SDKAppID obtained in step one
    "denny", // Please replace with your UserID
    "xxxxxxxxxxx", // You can calculate a UserSig in the console and fill it in here
    object : TUICallback() {
        override fun onSuccess() {
            Log.i(TAG, "login success")
        }

        override fun onError(errorCode: Int, errorMessage: String) {
            Log.e(TAG, "login failed, errorCode: $errorCode msg:$errorMessage")
        }
    })

TUILogin.login(context,
    1400000001, // Please replace with the SDKAppID obtained in Step 1
    "denny", // Please replace with your UserID
    "xxxxxxxxxxx", // You can calculate a UserSig in the console and fill it in here
    new TUICallback() {
        @Override
        public void onSuccess() {
            Log.i(TAG, "login success");
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
            Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + errorMessage)
        }
    });
```

Parameter Description

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

Parameter	Type	Description
SDKAppID	int	In the final step of step one, you have already obtained it, so it will not be

		repeated here.
UserID	String	The ID of the current user, string type, is only allowed to contain letters (a-z and A-Z), numbers (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in Step One , Step 3 to encrypt information such as SDKAppID and UserID to obtain UserSig, which is a token for authentication used by Tencent Cloud to identify whether the current user can use TRTC services. You can generate a temporarily usable UserSig through the Auxiliary Tools in the console. For more information, see UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate UserSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering. If your key is leaked, attackers can steal your Tencent Cloud traffic.

Production environment: If your project is to be released online, please use the [server-generated UserSig](#) method.

Step 5. Create a voice chat room

After the above log in to method call returns successfully, call the [VoiceRoomKit](#) method [createRoom](#) to create your voice chat live streaming room.

Kotlin

Java

```
import com.trtc.uikit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(applicationContext).createRoom("roomId", VoiceRoomDefin

import com.trtc.uikit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(getApplicationContext()).createRoom("roomId", new Voice
```

Voice chat room preview screen	Voice chat room in-room screen

Step 6: Join the voice chat room

After the above log in to method call returns successfully, call the [VoiceRoomKit](#) method [enterRoom](#), specify the room ID, and enter the Voice Chat Room Live Streaming.

Kotlin

Java

```
import com.trtc.uikit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(applicationContext).enterRoom("roomId")

import com.trtc.uikit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(getApplicationContext()).enterRoom("roomId");
```

Voice Chat Room	Voice Chat Room

More features

- [Room List](#)
- [Follow Anchors](#)
- [Barrage](#)
- [Gift](#)

Q&A

If you encounter problems with access and use, see [Q&A](#).

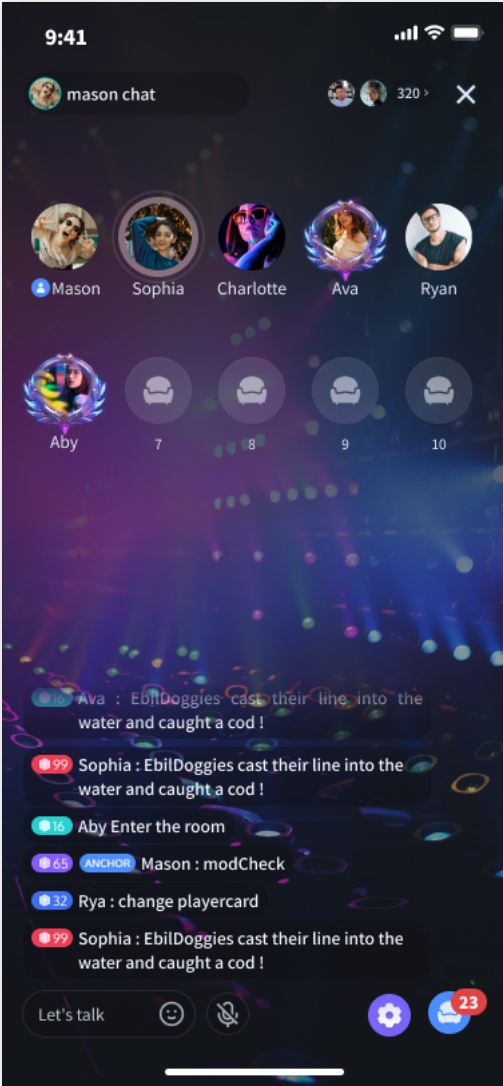
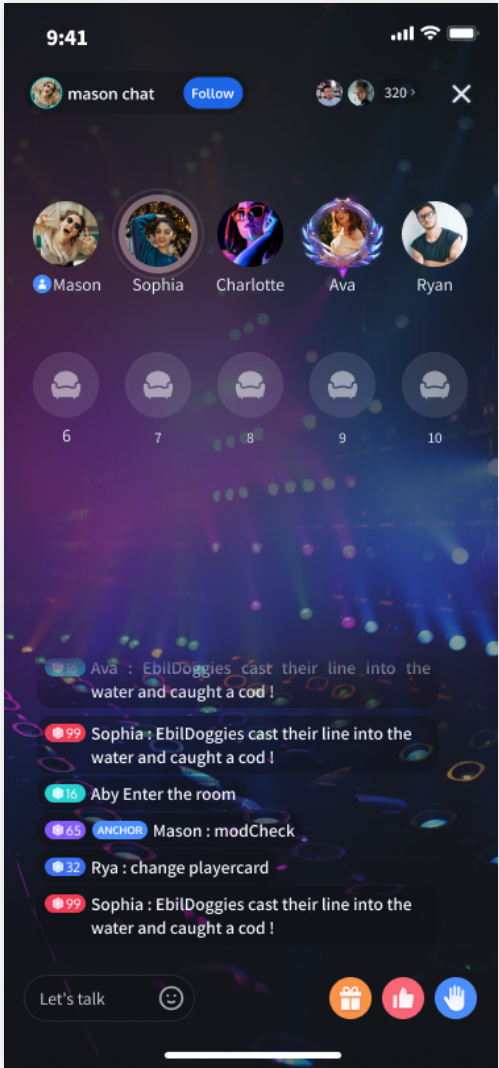
Suggestions and Feedback

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

iOS

Last updated : 2025-01-09 18:08:31

This article will guide you through the process of integrating the TUILiveKit component in a short time. Following this document, you will complete several key steps within an hour and ultimately obtain a voice room feature with a complete UI interface.

Anchor	Audience
	

Environment preparations

Xcode 15 or later

iOS 13.0 or later

CocoaPods environment installation, [click to view](#).

If you encounter problems with access and use, see [Q&A](#).

Step 1. Activate the service

Before using Tencent Cloud's audio and video services, you need to go to the console and activate the audio and video services for your application. For details, see [Activate the Service \(TUILiveKit\)](#).

Step 2. Import the Component

Import components into CocoaPods. If problems exist, Please refer first [Environment Preparation](#). The import components are as follows:

1. Please add the `pod 'TUILiveKit'` dependency to your `Podfile` file and refer to the [Example](#) project if you run into any problems.

```
target 'xxxx' do
  ...
  ...
  pod 'TUILiveKit'
end
```

If you don't have a `Podfile` file, first terminal `cd` into the `xxxx.xcodeproj` directory and then create it with the following command:

```
pod init
```

2. In the terminal, first `cd` to the `Podfile` directory, and then run the following command to install the component.

```
pod install
```

If the latest version of `TUILiveKit` cannot be installed, You can delete **Podfile.lock** and **Pods** first. Then update the CocoaPods repository list locally by executing the following command.

```
pod repo update
```

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

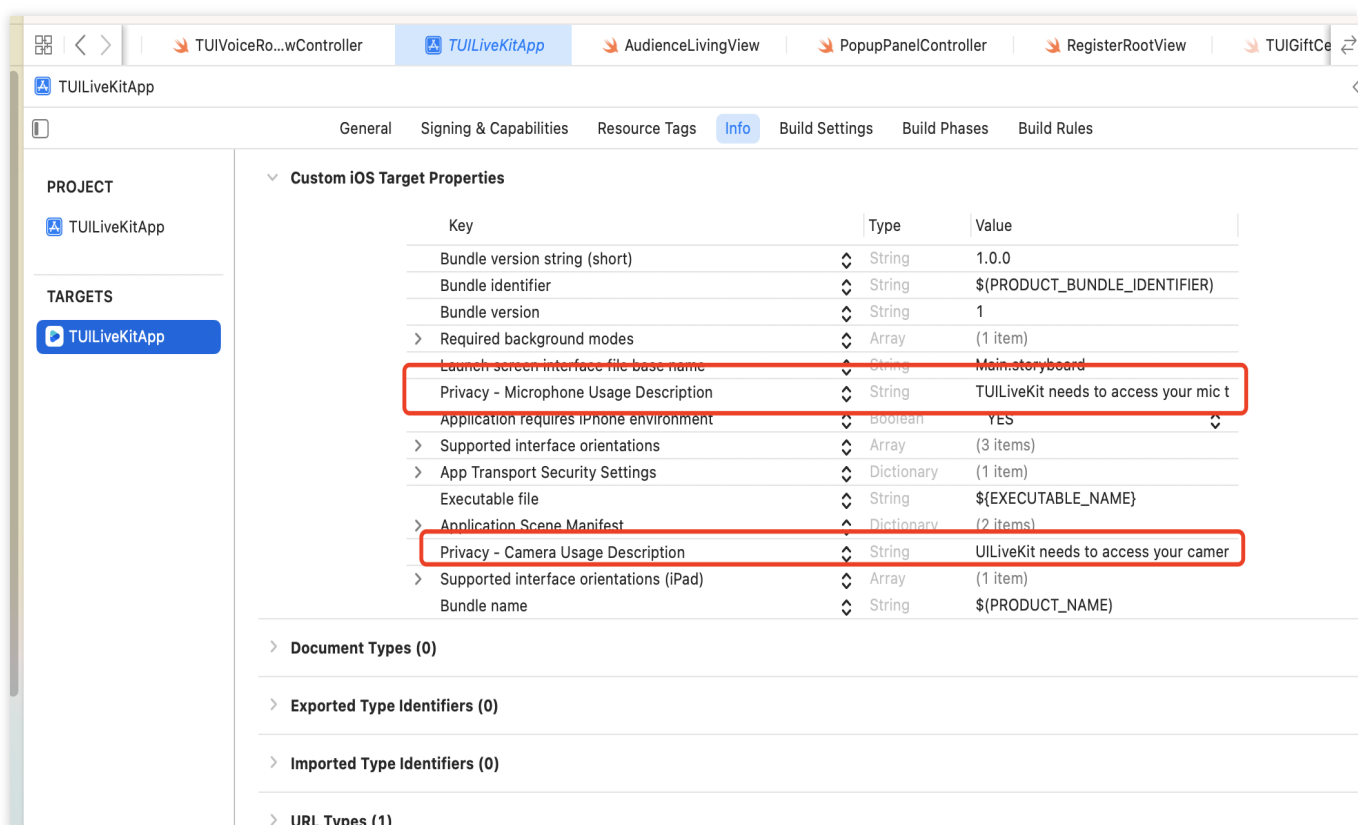
```
pod update
```

3. You can compile and run it first. If you run into problems, see [Q&A](#). If the problem still can't be solved, you can run our [Example](#) project first. Any problems you encounter in the process of access and use, welcome to give us [feedback](#).

Step 3. Configure the Project

To use audio and video features, microphone and camera usage permissions are required. In the App's Info.plist, add the following two items, corresponding to the prompt messages for microphone and camera when the system pops up the authorization dialog.

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



Step 4. Log in

Add the following code to your project, which completes the login of the TUI component by calling the relevant interface in TUICore. This step is very important, because you can use all functions of TUILiveKit only after the login is successful. Therefore, please patiently check whether the relevant parameters are correctly configured.

Swift

Objective-C

```
//
// AppDelegate.swift
//

import TUICore
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) throws -> Bool {
    TUILogin.login(14000000001, // Replace it with the SDKAppID obtained in the console
        userID: "denny", // Please replace it with your UserID
        userSig: "xxxxxxxxxxx") { // You can calculate a UserSig in the console
        print("login success")
    } fail: { (code, message) in
        print("login failed, code: \(code), error: \(message ?? "nil")")
    }

    return true
}

//
// AppDelegate.m
//

#import <TUICore/TUILogin.h>
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [TUILogin login:14000000001 // Replace it with the SDKAppID obtained in the console
        userID:@"denny" // Please replace it with your UserID
        userSig:@"xxxxxxxxxxx" // You can calculate a UserSig in the console
        succ:^(int code, NSString * _Nullable msg) {
            NSLog(@"login success");
        } fail:^(int code, NSString * _Nullable msg) {
            NSLog(@"login failed, code: %d, error: %@", code, msg);
        }];
    return YES;
}
```

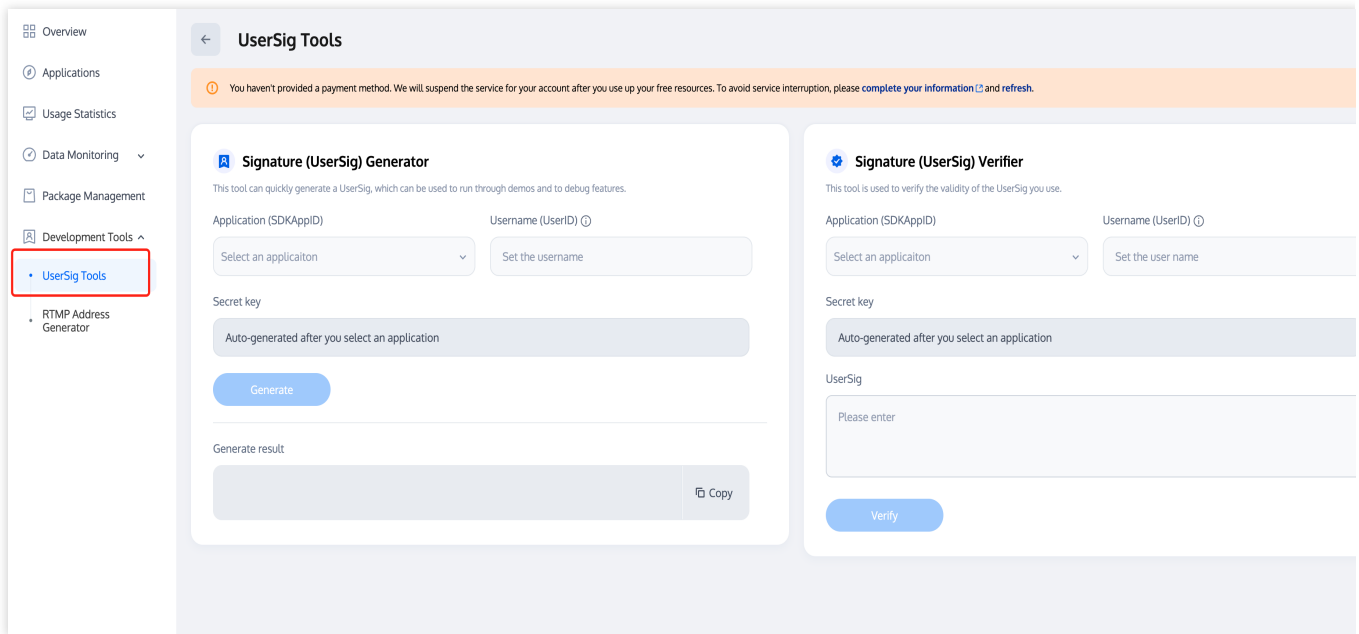
Parameter description

This section details the key parameters required in the login function:

SDKAppID: Obtained in the last step in [Step 1](#) and not detailed here.

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

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



For more information, see [UserSig](#).

Note:

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

SDKAppID is invalid.

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

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

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

Step 5. Create a voice chat room

After the above log in to method call returns successfully, call the [VoiceRoomKit](#) method [createRoom](#) to create your voice chat live streaming room.

Swift

Objective-C

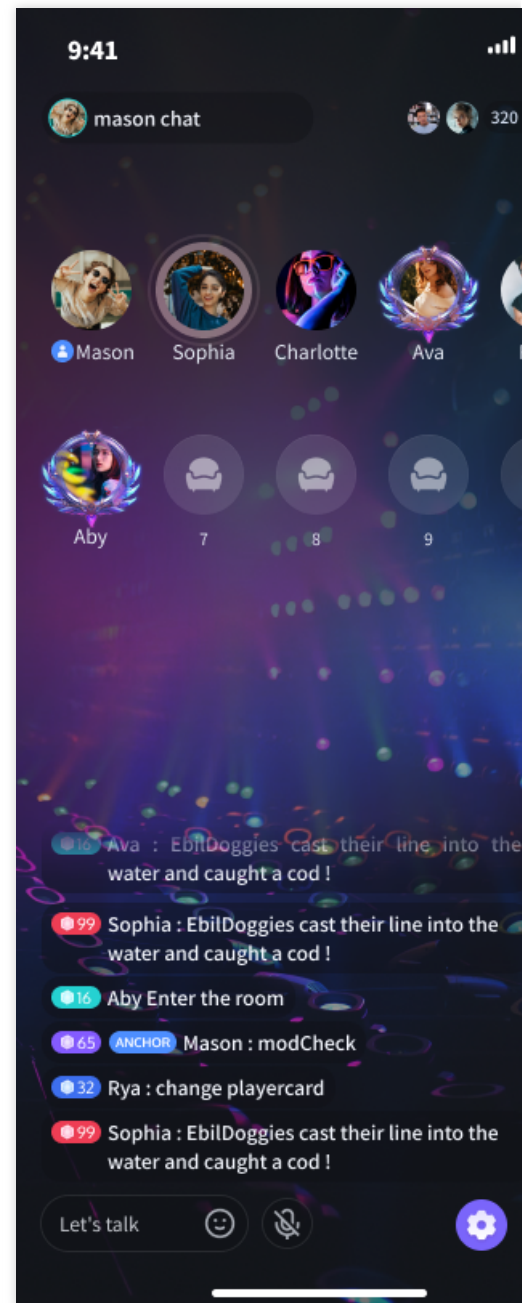
```
let params = CreateRoomParams()
params.maxAnchorCount = VoiceRoomDefine.MAX_CONNECTED_VIEWERS_COUNT
params.seatMode = .applyToTake

VoiceRoomKit.createInstance().createRoom(roomId: "your room id", params: params)

CreateRoomParams *params = [[CreateRoomParams alloc] init];
params.maxAnchorCount = VoiceRoomDefine.MAX_CONNECTED_VIEWERS_COUNT;
params.seatMode = TUISeatModeApplyToTake;

[[VoiceRoomKit sharedInstance] createRoomWithRoomId:@"your room id" params:params];
```

Voice chat room preview screen	Voice chat room in-room screen



Step 6: Join the voice chat room

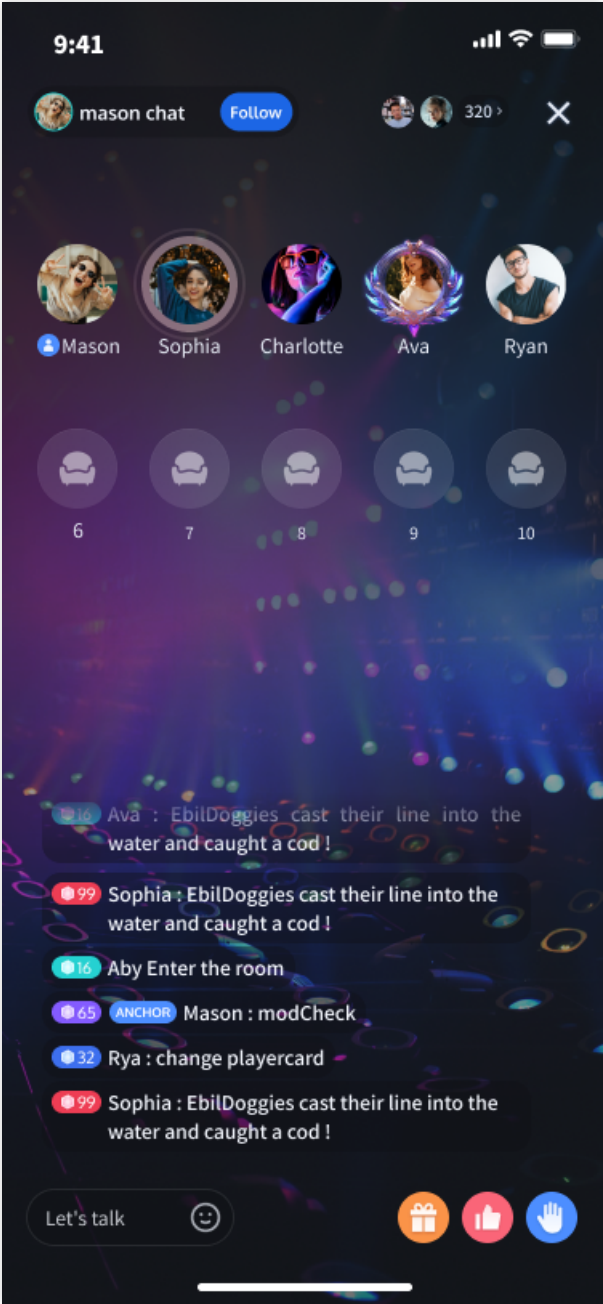
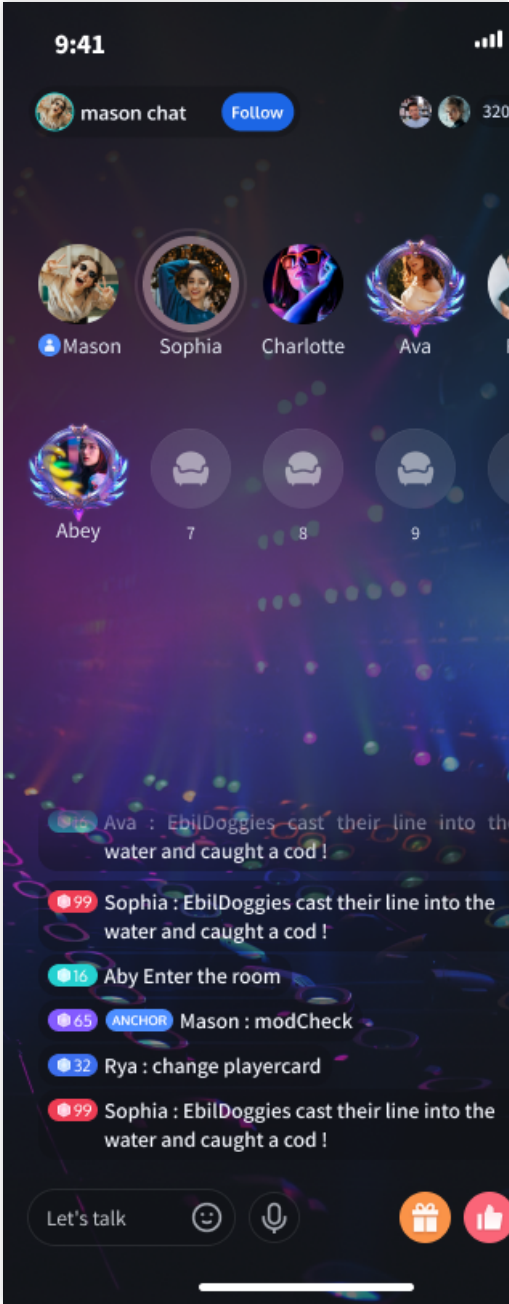
After the above log in to method call returns successfully, call the [VoiceRoomKit](#) method [enterRoom](#), specify the room ID, and enter the Voice Chat Room Live Streaming.

Swift

Objective-C

```
VoiceRoomKit.createInstance().enterRoom(roomId: "your room id")
```

```
[[VoiceRoomKit sharedInstance] enterRoomWithRoomId:@"your room id"];
```

Voice Chat Room	Voice Chat Room
	

More features

[Room List](#)

[Follow Anchors](#)[Barrage](#)[Gift](#)

Q&A

If you encounter any problems with access and use, please refer to [Q&A](#).

Suggestions and Feedback

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

Flutter

Last updated : 2025-04-09 18:04:15

This document will introduce how to complete the access of the TUILiveKit component in a short time. Follow this document, and you can complete the access work within 10 minutes and finally obtain a voice room feature with a complete UI interface.

Environment Preparation

Platform	Version
Flutter	Flutter 3.27.4 and higher versions. Dart 3.6.2 or higher versions.
Android	Android Studio 3.5 and above versions. Android devices running Android 5.0 and above versions.
iOS	Xcode 15.0 and above versions. Ensure that your project has a deemed valid developer signature.

Step 1: Activating Service

See [Activate Service \(TUILiveKit\)](#) to claim the experience version or enable the paid edition.

Step Two: Import the TUILiveKit Component

In the root directory of the project, run the following command to install the `tencent_live_uikit` plug-in via command line.

```
flutter pub add tencent_live_uikit
```

Step Three: Complete Project Configuration

Android

iOS

1. If you need to compile and run on the Android platform, since we use Java's reflection features internally in the SDK, some classes in the SDK need to be added to the non-obfuscation list.

First, need to configure and activate the obfuscation rule in the `android/app/build.gradle` file of the project:

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

Create a `proguard-rules.pro` file under the `android/app` directory of the project, and add the following code in the `proguard-rules.pro` file:

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

2. Configure to enable Multidex support in the `android/app/build.gradle` file of the project.

```
android {
    .....
    defaultConfig {
        .....
        multiDexEnabled true
    }
}
```

1. Use **Xcode** to open your project, select **Project > Building Settings > Deployment**, set the **Strip Style** below to `Non-Global Symbols` to preserve the necessary global symbol information.

2. **Selectable** If you need to debug on the iOS Simulator, and the Mac computer you are using uses an Intel chip, you need to add the following code in the `ios/Podfile` file of the project:

```
target 'xxxx' do
    .....
end
.....

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

```
    end
  end
end
```

3. Since TUILiveKit will use iOS's audio and video features, authorization needed for permission to use microphone and camera.

Authorization operation method: Add the following two items at level 1 `<dict>` under the directory of `Info.plist` in your iOS project, which correspond to the prompt messages when the system pops up the authorization dialog for the microphone and camera respectively.

```
<key>NSCameraUsageDescription</key>
<string>The CallingApp requires camera permission. Video recording with picture onl
<key>NSMicrophoneUsageDescription</key>
<string>The CallingApp needs to access your microphone permission. The recorded vid
```

After completing the above additions, add the following preprocessor definitions in your **ios/Podfile** to enable camera and mic permissions.

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
        '$(inherited)',
        'PERMISSION_MICROPHONE=1',
        'PERMISSION_CAMERA=1',
      ]
    end
  end
end
```

Step 4: Configure Routing and Internationalization

You need to configure routing and proxy in the app. Take configuration in `MaterialApp` as an example. Code as follows:

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

return MaterialApp(
  navigatorObservers: [TUILiveKitNavigatorObserver.instance],
  localizationsDelegates: [
    ...LiveKitLocalizations.localizationsDelegates,
    ...BarrageLocalizations.localizationsDelegates,
```

```
        ...GiftLocalizations.localizationsDelegates,  
    ],  
    supportedLocales: [  
        ...LiveKitLocalizations.supportedLocales,  
        ...BarrageLocalizations.supportedLocales,  
        ...GiftLocalizations.supportedLocales  
    ],  
    //...  
);
```

Step 5: Log In

Before using the Voice Room feature, make sure you have executed the following login code to complete the initialization work.

```
import 'package:tencent_live_uikit/tencent_live_uikit.dart';  
  
void login() async {  
    await TUILogin.instance.login(  
        1400000001,      // Replace with the SDKAppID obtained in step 1  
        "denny",        // Replace with your UserID  
        "xxxxxxxxxx",    // You can calculate a UserSig in the console and fill it in th  
        TUICallback(  
            onError: (code, message) {  
                print("TUILogin login fail, {code:$code, message:$message}");  
            },  
            onSuccess: () async {  
                print("TUILogin login success");  
            },  
        ),  
    );  
};
```

Parameter Description

Here, we will introduce several key parameters needed in the login function:

SDKAppID: You have already obtained it in the last step of step 1. No redundancy here.

User ID: The current user's ID, string type, only allow to include English letters (a-z and A-Z), numbers (0-9), hyphen (-) and underscore (_).

UserSig: Encrypt information such as `SDKAppID` and `UserID` using the `SecretKey` obtained in step 3 of [step 1](#) to obtain `UserSig`. It is a ticket for authentication, used by Tencent Cloud to determine whether the current user can use TRTC services.

You can generate a temporary available UserSig through the [Auxiliary Tool](#) in the console.

For more information, see [UserSig](#).

Notes:

This step is also the one for which we have received the most developer feedback currently. The common issues are as follows:

SDKAppID configuration error. The SDKAppID on the domestic site generally starts with 140 and is a 10-bit integer. UserSig is misconfigured as an encryption key (SecretKey). UserSig is generated by encrypting information such as SDKAppID, UserID, and expiration time with SecretKey, rather than directly configuring SecretKey as UserSig. The UserID is set to simple strings such as "1", "123", "111". Since **TRTC does not support multi-end login with the same UserID**, when multiple people collaborate in development, UserIDs like "1", "123", "111" can be easily occupied by your colleagues, causing login failure. Therefore, we recommend that you set some highly recognizable UserIDs when debugging.

The example code in Github uses the genTestUserSig function to calculate UserSig locally to help you complete the current access process more quickly. However, this solution will expose your SecretKey in the App's code, which is not conducive to your subsequent upgrade and protection of your SecretKey. Therefore, we strongly recommend that you perform the calculation logic of UserSig on the server side, and the app requests the real-time calculated UserSig from your server every time it uses the TUILiveKit component.

Step 6: Integration of the Anchor Broadcast Page

Where you need to enable voice room (specifically determined by your business, executed in its click event), perform the following operations, route to the anchor broadcast page:

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

Navigator.push(context, MaterialPageRoute(
  builder: (context) {
    final String userId = 'replace with your userId';
    final String roomId = LiveIdentityGenerator.instance.generateId(userId, RoomT
    return TUILiveRoomAnchorWidget(roomId: roomId);
  }));
```

Voice chat room preview screen	In-room screen of the voice chat room

Step 7: Integrating the Live List Page

On the live list page, the live voice chat rooms will be shown. You can click any live room and join the live room as an audience to listen and link mic.

You can perform the following operations to route to the live list page:

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

Navigator.push(context, MaterialPageRoute(
  builder: (context) {
    return Scaffold(
      body: SafeArea(child: LiveListWidget()));
  }));
```

You can also directly add the live room list page as a subspace of one of your pages:

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

// Single child widget, taking Container as an example
Container(
  child: LiveListWidget()
)

// Multiple child widget, taking Column as an example
Column(
  children:[LiveListWidget()]
)
```

Live list frame	Audience in-room screen of the voice chat room

Communication and Feedback

If you have any requirements or feedback, you can contact: info_rtc@tencent.com.

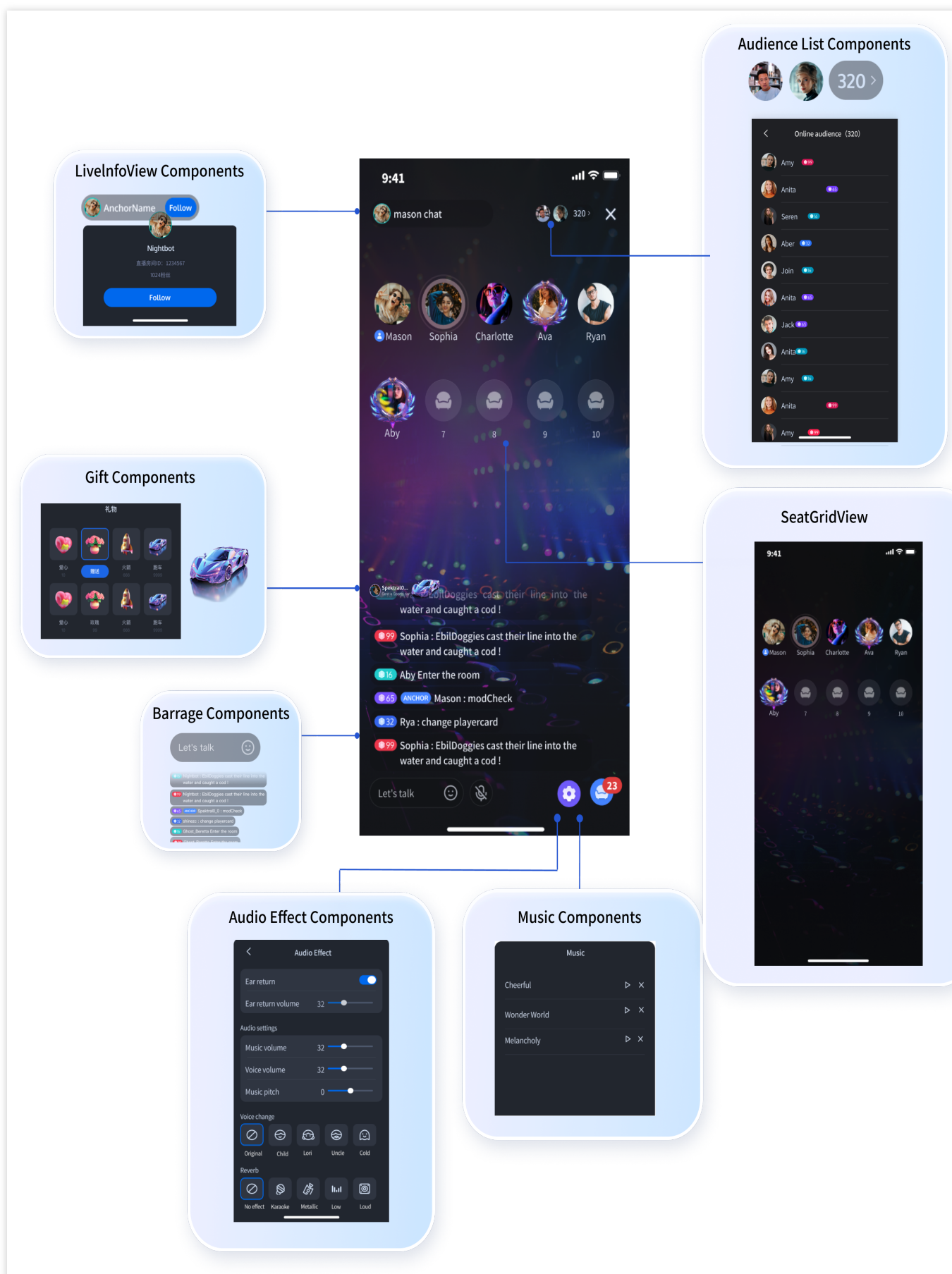
Customizable Interface (TUILiveKit)

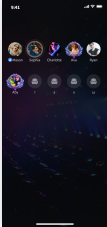
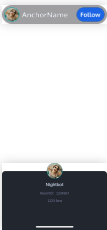
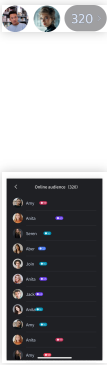
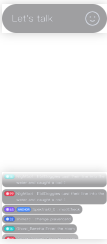
Last updated : 2025-03-18 08:43:00



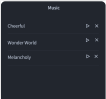

LiveKit Component Collection Introduction

LiveKit is a collection of UI components for live streaming scenarios based on Tencent Cloud RoomEngine SDK. It provides a series of general UI components, including core components, live room information component, audience list component, barrage component, gift component, etc.

By leveraging these UI components, you can quickly build your own voice chat room business logic like building blocks.



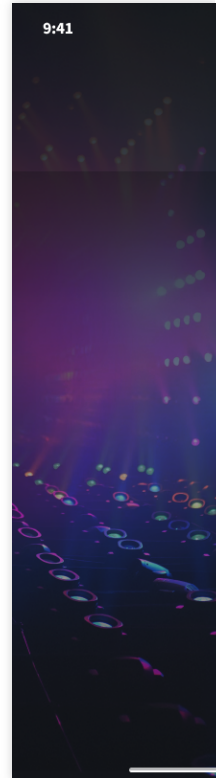
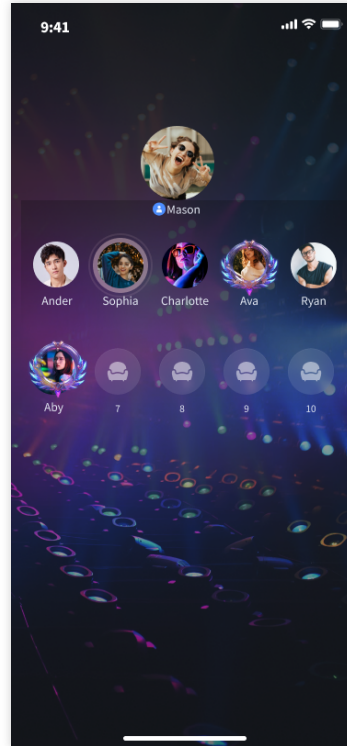
Component Name	Functionality	Thumbnail
SeatGridView	The voice chat room core components provide rich APIs for enabling and disabling the voice chat room, managing seat positions in the live streaming room, applying to join the seat, inviting to take seat, moving seat positions, and kicking someone off the seat.	
LiveInfoView	The live room information component displays anchor information and live room information , allowing viewers to follow or unfollow their favorite anchors, among other features.	
AudienceListView	The viewer list component displays the latest 100 viewers who have entered the live room and the total number of viewers. You can show a simplified view of the viewer list in your live streaming interface, and when the simplified view is clicked, it can bring up the detailed viewer list view.	
BarrageView	The interactive barrage component allows anchors and viewers to send text messages or emoji messages to each other. The interactive barrage component provides BarrageInputView (barrage sending view) and BarrageStreamView (barrage display view).	
GiftView	The interactive gifts component allows live streaming	

	<p>viewers to send virtual gifts to their favorite live anchors, with gift animation effects supported.</p> <p>The interactive gifts component provides GiftButton (gift sending button) and GiftPlayView (gift playback view).</p>	 
MusicView	<p>The music list component allows you to add background music to the live streaming room. You can also pause, replay, or delete background music.</p>	
AudioEffectView	<p>The Audio Effects Setting Component provides settings for your live streaming volume, background music volume, voice changer, and other effects.</p>	

SeatGridView Component Introduction

SeatGridView is a basic control we've developed for the Voice Chat Room UIKit. This core control offers a rich set of APIs, including enabling voice chat rooms, disabling voice chat rooms, seat position management in live streaming rooms, such as applying to join the seat, inviting to take seat, moving seat positions, and kicking someone off the seat. By calling just a few lines of SeatGridView component APIs, you can achieve the following live streaming view. For details on integrating the SeatGridView component, see the Access Documentation for [SeatGridView Access Documentation](#).

Seat List - Grid Layout	Seat List - Element Layout	Seat List - Vertical Layc

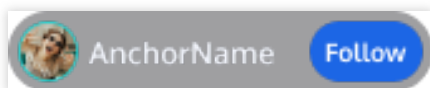


Key Component Introduction

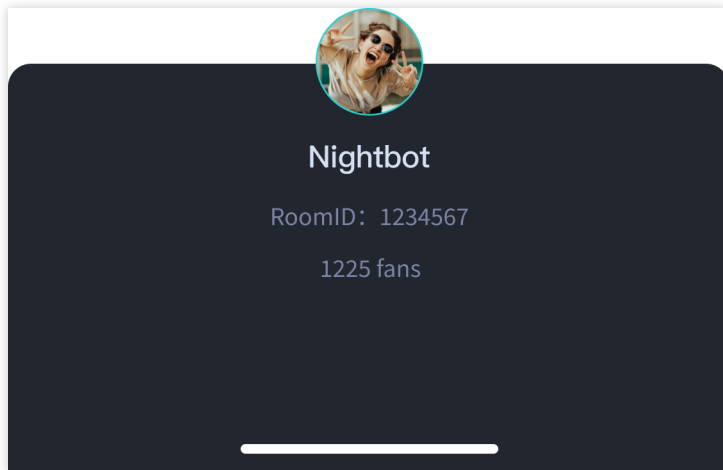
LiveInfoView Component

LiveInfoView (Live Room Information Component) displays the information of the anchor and the live room. By default, it shows the anchor information view. When you click on the anchor information view, a detailed view of the live room information will be displayed.

Anchor Information View:



Live Room Information View:



The Live Room Information Component is coming soon, stay tuned.

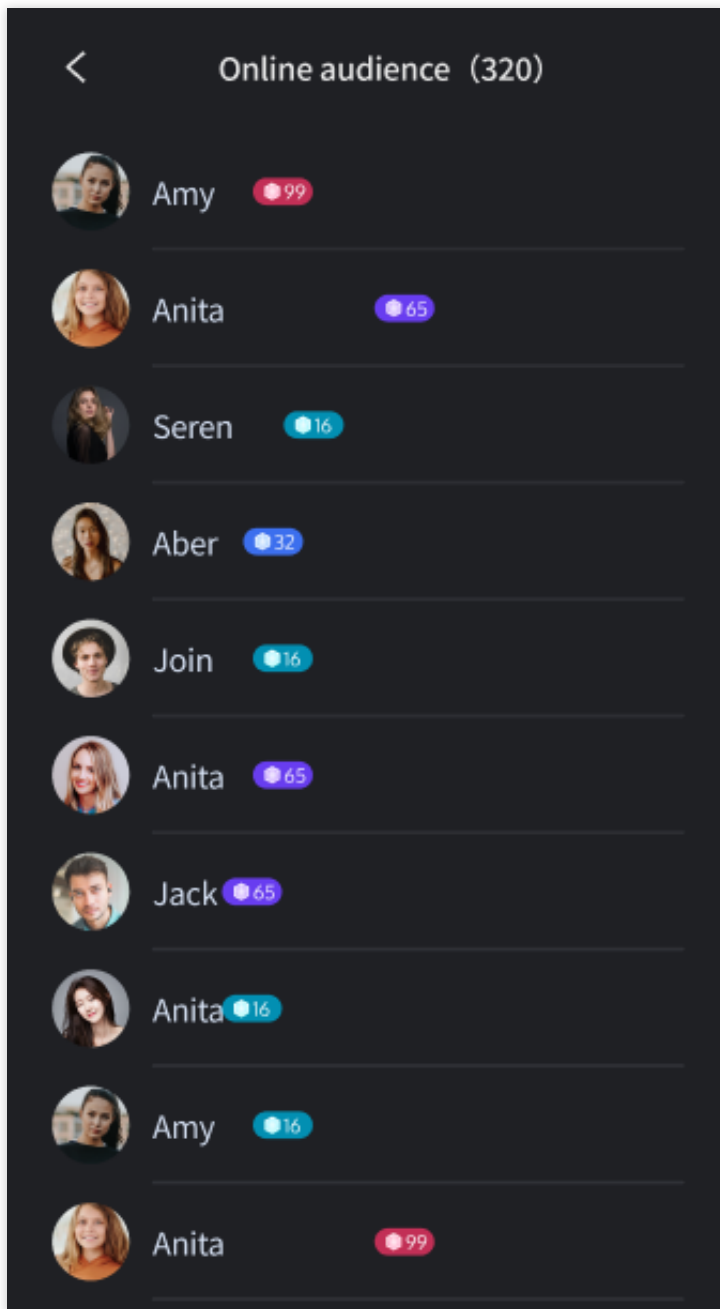
AudienceListView Component

AudienceListView (Audience List Component) displays the first 100 viewers who recently entered the live room and the total number of viewers. By default, it shows the audience list view, and clicking on it will expand to show a detailed view of the audience list.

Audience List View



Audience List Details

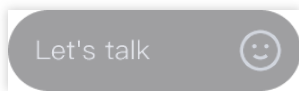


Audience List Component is coming soon, stay tuned.

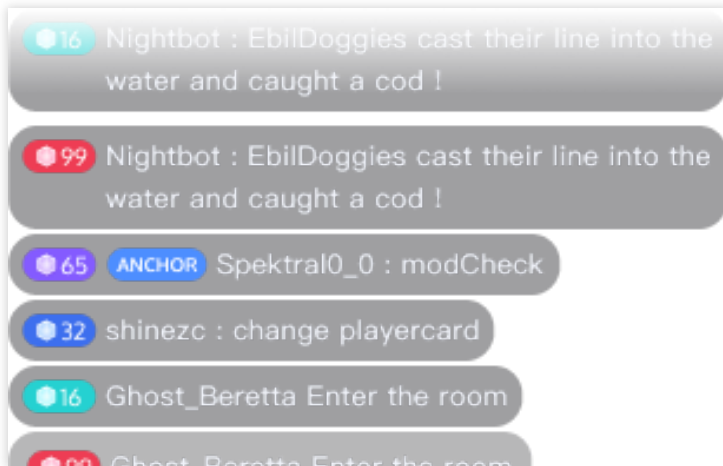
BarrageView Component

BarrageView (Interactive Barrage Component) is an important real-time communication tool that supports various interactive methods. Users can input emojis in the barrage to enhance the entertainment value of messages, making the interaction experience more enjoyable and vivid. Through this feature, viewers can have richer interactions with anchors and other viewers during live streaming, enhancing the overall sense of participation and fun. You can add the barrage sending component to the layout where you want to send barrages, and add the barrage display component to the layout where you want to display barrages.

Barrage Sending Component



Barrage Display Component

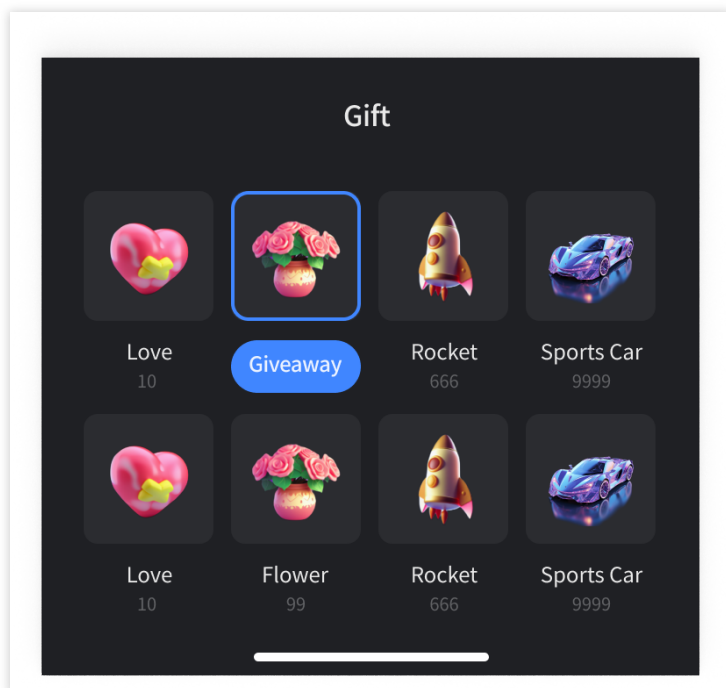


If you want to integrate the barrage component, please refer to the [Interactive Barrage](#) Component Integration Documentation.

GiftView Component

GiftView (Interactive Gifts Component) is a virtual gift interaction platform designed to add more fun to users' social experience. Through this component, users can send virtual gifts to their favorite live streaming anchors to express their appreciation, favorite, and support. The interactive gifts component supports setting **Gift Material**, **Balance Display**, **Normal Gift Playback**, **Full Screen Gift Playback**, and **Recharge Button** among other features.

Gift Sending Component



Gift Display Component

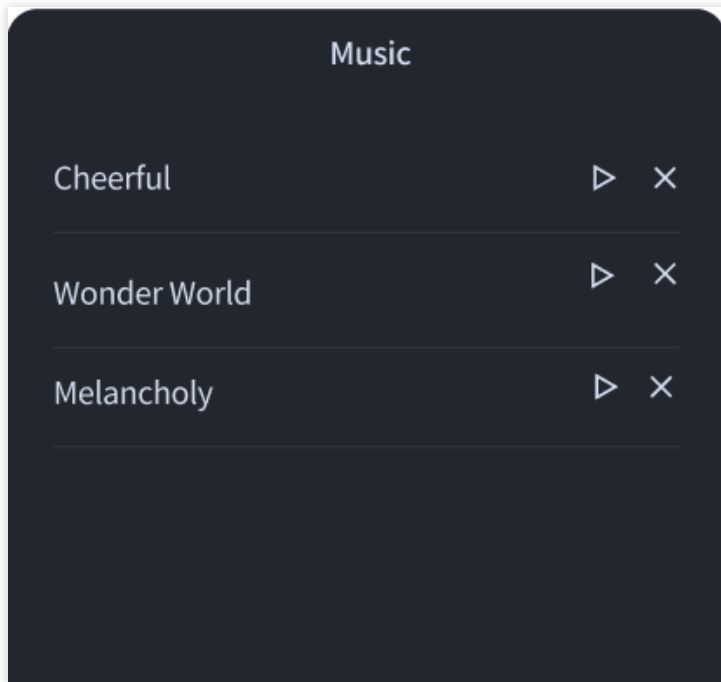


If you want to integrate the gift component, please refer to the [Interactive Gifts](#) Component Integration Documentation.

MusicView Component

MusicView (Music List Component) allows you to add background music to your live streaming room. Simply display our music list component when you need background music, and you can play your favorite music on the music component we provide.

Music List Component

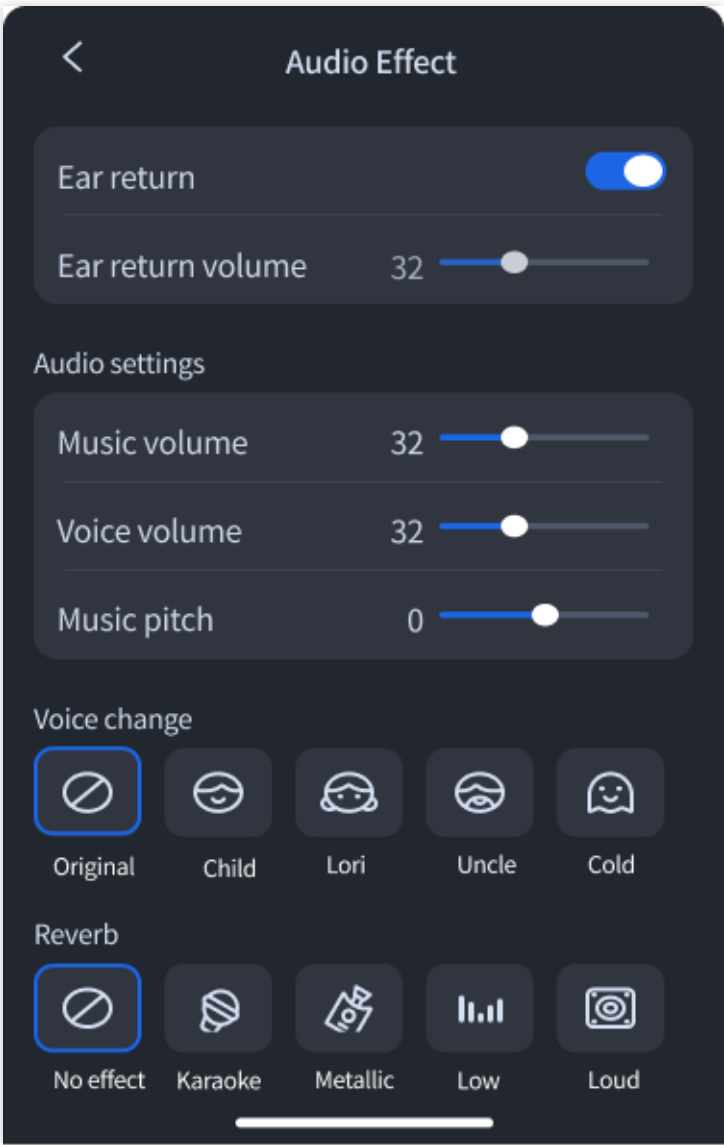


The Background Music Component is coming soon. Stay tuned.

AudioEffectView Component

AudioEffectView (Audio Effects Setting Component) allows you to set live streaming volume, background music volume, and voice changer effects. Simply display the audio effects setting component when you need to adjust these effects, and you can easily adjust to the desired results.

Audio Effects Setting Component



The Audio Effects Setting Component is coming soon. Stay tuned.

Live Streaming and Listening (TUILiveKi Android)

Last updated : 2025-04-07 16:52:20

Applicable Scenario

Our **broadcasters' streaming** and **viewers watching** feature mainly depend on **our core widget of the voice chat room** (SeatGridView), which provides a rich array of APIs for managing voice chat rooms, such as opening and closing them, managing seats within the live room, like applying for a seat, inviting to a seat, moving seats, kicking users off the seat, and more.

After you've integrated the voice chat room UIKit through [Quick Access](#), if there's a discrepancy between the UI style and your ideal UI style, you can use our core widgets to quickly build the main process of the voice chat room within half an hour. Then, add your own business UI views on top of it.

Environment Requirements

Android 5.0 (SDK API Level 21) or later.

Gradle v7.0 or later.

Mobile device on Android 5.0 or later.

For issues during environment configuration or compilation, please refer to [FAQ](#).

Step 1. Activate the service

Please refer to [Activating Services \(TUILiveKit\)](#) to receive the trial version or activate the paid version.

Step 2. Configure the project

1. Find the `build.gradle.kts` (or `build.gradle`) file under the app directory and add the following code to include the dependency for SeatGridView component:

build.gradle.kts

build.gradle

```
api("io.trtc.uikit:live-stream-core:latest.release")
```

```
api 'io.trtc.uikit:live-stream-core:latest.release'
```

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

```
-keep class com.tencent.** { *; }
-keep class com.trtc.uikit.livekit.voiceroomcore.** { *; }
-keep class com.google.gson.** { *; }
```

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

```
// app/src/main/AndroidManifest.xml

<application
    ...

    // Add the following configuration to override the configuration in the depende
    android:allowBackup="false"
    tools:replace="android:allowBackup">
```

Step 3. Log in

Add the following code to your project, which completes the login of TUI Components by calling the relevant interfaces in TUICore. This step is critical; you can only use the features provided by SeatGridView after successful login.

Kotlin

Java

```
TUIRoomEngine.login(applicationContext,
    1400000001,        // Please replace it with the SDKAppID obtained in step 1
    "denny",          // Please replace with your UserID
    "xxxxxxxxxxxx",    // You can calculate a UserSig in the console and fill it in t
    object : TUIRoomDefine.ActionCallback() {
        override fun onSuccess() {
            Log.i(TAG, "login success")
        }

        override fun onError(errorCode: Int, errorMessage: String) {
            Log.e(TAG, "login failed, errorCode: $errorCode msg:$errorMessage")
        }
    })
```

```
TUIRoomEngine.login(context,
    1400000001,        // Please replace it with the SDKAppID obtained in step 1
    "denny",           // Please replace with your UserID
    "xxxxxxxxxxxx",    // You can calculate a UserSig in the console and fill it in th
    new TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.i(TAG, "login success");
        }

        @Override
        public void onError(TUICommonDefine.Error error, String message) {
            Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + erro
        }
    });
```

Parameter Description

Here we detail the key parameters required in the login function:

Parameter	Type	Description
SDKAppID	int	You have already obtained it in the last step of Step 1, so it will not be elaborated here.
UserID	String	The ID of the current user, in string format, only allows letters (a-z and A-Z), digits (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in Step One , Step 3 to encrypt information such as SDKAppID and UserID to obtain UserSig, which is a token for authentication used by Tencent Cloud to identify whether the current user can use TRTC services. You can generate a temporarily usable UserSig through the Auxiliary Tools in the console. For more information, see UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate UserSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering. If your key is leaked, attackers can steal your Tencent Cloud traffic.

Production environment: If your project is to be released online, please use the [server-generated UserSig](#) method.

Step 4: Use core controls to implement the live streaming feature

Create core controls

Create core controls: You can load our core controls in your Activity through Java code or XML. The code example is as follows (XML method is similar):

kotlin
java

```
val seatGridView = SeatGridView(this)

SeatGridView seatGridView = new SeatGridView(this);
```

The anchor starts the live streaming room and the audience joins the live streaming room

Anchor starts the live streaming room: Start a live streaming room and enable local microphone capturing.

kotlin
java

```
val roomInfo = TUIRoomDefine.RoomInfo()
roomInfo.roomId = "roomId_123456"
seatGridView.startVoiceRoom(roomInfo, null)

seatGridView.startMicrophone(null)

TUIRoomDefine.RoomInfo roomInfo = new TUIRoomDefine.RoomInfo();
roomInfo.roomId = "roomId_123456";
seatGridView.startVoiceRoom(roomInfo, null);

seatGridView.startMicrophone(null);
```

Audience joins the live streaming room.

kotlin
java

```
seatGridView.joinVoiceRoom("roomId_123456", null)

seatGridView.joinVoiceRoom("roomId_123456", null);
```

Anchor starts the live streaming room and begins the live stream	Audience joins the live streaming room to watch the live stream

Microphone Position Management

If you need to implement the Seat Management feature, refer to the [Seat Management](#) documentation.

Set seat list layout arrangement

You can quickly set up your seat list layout in the following ways.

kotlin

java

```
// Set grid layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.GRID, null)

// Set element layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FOCUS, null)

// Set vertical layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.VERTICAL, null)

// Set custom layout
val layoutConfig = VoiceRoomDefine.SeatViewLayoutConfig().apply {
    rowConfigs = ArrayList()
    rowSpacing = dp2px(10); //Spacing between each row
}
// First row configuration
val rowConfig1 = VoiceRoomDefine.SeatViewLayoutRowConfig().apply {
    count = 3 //Number of seats displayed in the first row
    seatSize = VoiceRoomDefine.Size(dp2px(50), dp2px(50)) //Size of each seat view
    seatSpacing = dp2px(10) //Horizontal spacing of each seat in the first row
    alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.CENTER //Alignment of s
}
layoutConfig.rowConfigs.add(rowConfig1)
// Second row configuration
val rowConfig2 = VoiceRoomDefine.SeatViewLayoutRowConfig().apply {
    count = 3 //Number of seats displayed in the second row
    seatSize = VoiceRoomDefine.Size(dp2px(50), dp2px(50)) //Size of each seat view
    seatSpacing = dp2px(10) //Horizontal spacing of each seat in the second row
    alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.SPACE_AROUND //Alignmen
}
layoutConfig.rowConfigs.add(rowConfig2)
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FREE, layoutConfig)

// Set grid layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.GRID, null)

// Set element layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FOCUS, null)
```

```
// Set vertical layout
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.VERTICAL, null)

// Set free layout
VoiceRoomDefine.SeatViewLayoutConfig layoutConfig = new VoiceRoomDefine.SeatViewLa
layoutConfig.rowConfigs = new ArrayList<>();
layoutConfig.rowSpacing = dp2px(10); //Spacing between each row
// First row configuration
VoiceRoomDefine.SeatViewLayoutRowConfig rowConfig1 = new VoiceRoomDefine.SeatViewLa
rowConfig1.count = 3; //Number of seats displayed in the first row
rowConfig1.seatSize = new VoiceRoomDefine.Size(dp2px(50),dp2px(50)); //Size of each
rowConfig1.seatSpacing = dp2px(10); //Horizontal spacing of each seat in the first
rowConfig1.alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.CENTER; //Alignme
layoutConfig.rowConfigs.add(rowConfig1);
// Second row configuration
VoiceRoomDefine.SeatViewLayoutRowConfig rowConfig2 = new VoiceRoomDefine.SeatViewLa
rowConfig2.count = 3; //Number of seats displayed in the second row
rowConfig2.seatSize = new VoiceRoomDefine.Size(dp2px(50),dp2px(50)); //Size of each
rowConfig1.seatSpacing = dp2px(10); //Horizontal spacing of each seat in the second
rowConfig2.alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.SPACE_AROUND; //A
layoutConfig.rowConfigs.add(rowConfig2);
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FREE, layoutConfig);
```

Note:

The parameter settings of the custom layout can be viewed in the parameter description of [SeatViewLayoutRowConfig](#). The alignment method can be referred to the [SeatViewLayoutRowAlignment](#) description. The alignment effect can be referred to the [Custom Layout Alignment Illustration](#).

Grid Layout	Element Layout	Vertical Layout	Custom Layout

Custom Layout Alignment Illustration:

Custom Seat View

If you think our default UI does not meet your needs and you want to customize your seat view, you can quickly set up your seat layout in the following ways to fully customize your seat view UI.

kotlin

java

```
val adapter = object : VoiceRoomDefine.SeatViewAdapter {
    override fun createSeatView(seatGridView: SeatGridView, seatInfo: TUIRoomDefine.SeatInfo) {
        return TestSeatInfoView(context, seatGridView, seatInfo)
    }

    override fun updateSeatView(seatGridView: SeatGridView, seatInfo: TUIRoomDefine.SeatInfo) {
        (seatView as TestSeatInfoView).updateSeatView(seatGridView, seatInfo)
    }

    override fun updateUserVolume(seatGridView: SeatGridView, volume: Int, customSeatView: TestSeatInfoView) {
        (customSeatView as TestSeatInfoView).updateUserVolume(seatGridView, volume)
    }
}

seatGridView.setSeatViewAdapter(adapter)

class TestSeatInfoView constructor(context: Context, seatGridView: SeatGridView, seatInfo: TUIRoomDefine.SeatInfo) {
    init {
        initView() //Initialize view
    }

    fun updateSeatView(seatGridView: SeatGridView, seatInfo: TUIRoomDefine.SeatInfo) {
        updateView(seatInfo) //Update custom seat view UI
    }

    fun updateUserVolume(seatGridView: SeatGridView, volume: Int) {
        updateUserVolume(volume) //Update volume change UI
    }
}

VoiceRoomDefine.SeatViewAdapter adapter = new VoiceRoomDefine.SeatViewAdapter() {
    @Override
    public View createSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo) {
        return new TestSeatInfoView(getApplicationContext(), seatGridView, seatInfo)
    }

    @Override
    public void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo, View customSeatView) {
        ((TestSeatInfoView) customSeatView).updateSeatView(seatGridView, seatInfo);
    }
}
```



```
@Override
public void updateUserVolume(SeatGridView seatGridView, int volume, View custom
    ((TestSeatInfoView) customSeatView).updateUserVolume(seatGridView, volume);
}
};
seatGirdView.setSeatViewAdapter(adapter);

public class TestSeatInfoView extends FrameLayout {

    public TestSeatInfoView(@NonNull Context context, SeatGridView seatGirdView, TU
        super(context);
        initView();
    }

    public void updateSeatView(SeatGridView seatGirdView, TUIRoomDefine.SeatInfo se
        updateView(seatInfo);
    }

    public void updateUserVolume(SeatGridView seatGirdView, int volume) {
        updateUserVolume(volume);
    }
}
```

Default Mic View	Custom Mic View Example

iOS

Last updated : 2025-04-07 17:07:21

Applicable Scenario

Our **broadcasters' streaming** and **viewers watching** feature mainly depend on our core control (**LiveCoreView**), which provides a rich array of APIs for managing voice chat rooms, such as opening and closing them, managing seats within the live room, like applying for a seat, inviting to a seat, moving seats, kicking users off the seat, and more.

After you've integrated the voice chat room UIKit through [Quick Access](#), if there's a discrepancy between the UI style and your ideal UI style, you can use our core widgets to quickly build the main process of the voice chat room within half an hour. Then, add your own business UI views on top of it.

Environment preparations

Xcode 15 or later.

iOS 13.0 or later.

CocoaPods environment setup, [Click to view](#).

If you encounter any problems with access and use, please refer to [FAQs](#).

Step 1. Activate the service

Please refer to [Activating Services \(TUILiveKit\)](#) to receive the trial version or activate the paid version.

Step 2: Import the LiveStreamCore component

Use CocoaPods to import the component. If you encounter any issues, please refer to [Environment Preparation](#) first.

Detailed steps for importing the component are as follows:

1. Add the `pod 'LiveStreamCore'` dependency to your `Podfile` file.

Swift

```
target 'xxxx' do
  ...
  ...
  pod 'LiveStreamCore'
```

```
end
```

If you don't have a `Podfile` file, first `cd` in Terminal into the `xxxx.xcodeproj` directory, then create one using the following command:

```
pod init
```

2. In Terminal, first `cd` into the `Podfile` directory and then run the following command to install components.

```
pod install
```

If you can't install the latest version of `SeatGridView`, delete **Podfile.lock** and **Pods** first. Then run the following command to update the local CocoaPods repository list.

```
pod repo update
```

Then, run the following command to update the Pod version of the component library.

```
pod update
```

3. You can start by compiling and running the code. If you encounter any issues, please refer to [FAQ](#). If the problem remains unresolved, consider running our [Example](#) project. We welcome any feedback on issues you encounter during integration and use.

Step 3. Log in

To log in to the `LiveStreamCore` component, add the following code to your project to call the relevant APIs in `RTCRoomEngine`. This step is very important, as the user can use the component features properly only after successfully logging in. Carefully check that the relevant parameters are correctly configured:

swift

```
//  
// AppDelegate.swift  
//  
  
import RTCRoomEngine  
  
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) throws {  
    TUIRoomEngine.login(sdkAppId: 1400000001, // Replace it with the  
                        userId: "denny", // Replace with your User ID  
                        userSig: "xxxxxxxxxxxx") { // Calculate a UserSig in  
        print("login success")  
    } fail: { (code, message) in  
        print("login failed: \(code) \(message)")  
    }  
}
```

```
        print("login failed, code: \\\(code), error: \\\(message ?? "nil")")
    }

    return true
}
```

Parameter Description

Here we detail the key parameters required in the login function:

Parameter	Type	Description
SDKAppID	int	You have already obtained it in the last step of Step 1, so it will not be elaborated here.
UserID	String	The ID of the current user, in string format, only allows letters (a-z and A-Z), digits (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in step 3 of Step One to encrypt information such as SDKAppID and UserID to generate a UserSig. It's a credential used for authentication purposes, allowing Tencent Cloud to identify if the current user is authorized to use the TRTC service. You can generate a temporary UserSig through the Auxiliary Tools in the console. For more information, please refer to How to Calculate and Use UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate userSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering, and once your key is leaked, attackers can steal your Tencent Cloud traffic.

Production Environment: If your project is going to be launched, please adopt the method of [Server-side Generation of UserSig](#).

Step 4: Use core controls to implement the live streaming feature

Create core controls and enable preview

Create Core Controls: In the Activity where you implement streaming, you can load our core controls using Java code or XML. An example of the code method is as follows (the XML method is similar):

swift

```
import LiveStreamCore

let seatGridView = SeatGridView(this)
```

The anchor starts the live streaming room and the audience joins the live streaming room

Anchor opens the live room: Opens a live streaming room and streams data collected from the local microphone.
swift

```
import LiveStreamCore

let roomInfo = TUIRoomInfo()
roomInfo.roomId = "123456"
roomInfo.seatMode = .applyToTake

seatGridView.startVoiceRoom(roomInfo: roomInfo) { roomInfo in
} onError: { code, message in
}

seatGridView.startMicrophone() {
} onError: { code,message in
}
```

Audience joins the live streaming room.

swift

```
import LiveStreamCore

seatGridView.joinVoiceRoom(roomId: "roomId_123456") { roomInfo in
} onError: { code, message in
}
```

Anchor starts the live streaming room and begins the live stream	Audience joins the live streaming room to watch the live stream

Microphone Position Management

If you need to implement the Seat Management feature, refer to the [Seat Management](#) documentation.

Set layout arrangement for the seat list

You can quickly set up your seat list layout in the following ways.

swift

```
import LiveStreamCore

// Set grid layout
seatGridView.setLayoutMode(layoutMode: .grid)

// Set element layout
seatGridView.setLayoutMode(layoutMode: .focus)

// Set vertical layout
seatGridView.setLayoutMode(layoutMode: .vertical)

// Set custom layout
// First row configuration
let rowConfig1 = SGSeatViewLayoutRowConfig(count: 3, //Number of seats displayed in
                                           seatSpacing: 10, //Horizontal spacing be
                                           seatSize: CGSize(width: 50, height: 50),
                                           alignment: .center) //Alignment of seats

// Second row configuration
let rowConfig2 = SGSeatViewLayoutRowConfig(count: 3, //Number of seats displayed in
                                           seatSpacing: 10, //Horizontal spacing be
                                           seatSize: CGSize(width: 50, height: 50),
                                           alignment: .spaceAround) //Alignment of

let layoutConfig = SGSeatViewLayoutConfig(rowConfigs: [rowConfig1, rowConfig2],
                                           rowSpacing: 10)

seatGridView.setLayoutMode(layoutMode: .free, layoutConfig: layoutConfig)
```

Note:
For the parameter settings of the custom layout, refer to the parameter descriptions in [SGSeatViewLayoutRowConfig](#). The alignment method alignment can be found in [SGSeatViewLayoutRowAlignment](#). See the [Schematic diagram](#) for the alignment effect.

Grid Layout	Element Layout	Vertical Layout	Custom Layout

Custom Layout Alignment Illustration:

Custom Seat View

If you think our default UI does not meet your needs and you want to customize your own seat UI, you can quickly set up your seat layout in the following way.

swift

```
import LiveStreamCore

class TestSeatViewDelegate: SGSeatViewDelegate {
    func seatGridView(_ view: SeatGridView, createSeatView seatInfo: TUISeatInfo) -
        return TestSeatInfoView(seatGridView: view, seatInfo: seatInfo)
    }

    func seatGridView(_ view: SeatGridView, updateSeatView seatInfo: TUISeatInfo, s
        if let seatView = seatView as? TestSeatInfoView {
            seatView.updateSeatView(seatGridView: view, seatInfo: seatInfo)
        }
    }

    func seatGridView(_ view: SeatGridView, updateUserVolume volume: Int, seatView:
        if let seatView = seatView as? TestSeatInfoView {
            seatView.updateUserVolume(seatGridView: view, volume: volume)
        }
    }
}

seatGirdView.setSeatViewDelegate(TestSeatViewDelegate())

class TestSeatInfoView: UIView {
    init(seatGridView: SeatGridView, seatInfo: TUISeatInfo) {
        super.init(frame: .zero)
        initView() //Initialize view
    }

    func updateSeatView(seatGridView: SeatGridView, seatInfo: TUISeatInfo) {
        updateView(seatInfo) //Update custom seat view UI
    }

    func updateUserVolume(seatGridView: SeatGridView, volume: Int) {
        updateUserVolume(volume) //Update volume change UI
    }
}
```

Before customizing the seat layout	After customizing the seat layout

Flutter

Last updated : 2025-04-17 11:37:16

Applicable scenario

Anchor starts live streaming and **Listeners listen** features mainly depend on **core component**

(`LiveStreamCore`), which provides various APIs such as enabling/disabling voice chat room, managing microphone positions in the live streaming room, including applying for microphone mode, inviting speakers, moving seats, and kicking someone off the mic.

After you integrate the Voice Chat Room UIKit through [Quick Integration](#), if the UI style does not match your ideal UI style, you can use the core component to rapidly deploy the main process of the voice chat room within half an hour. Then add your own business UI view on top of it.

Environment Preparation

Platform	Version
Flutter	Flutter 3.27.4 and higher versions. Dart 3.6.2 or higher version.
Android	Android Studio 3.5 and above versions. Android device running Android 5.0 and above versions.
iOS	Xcode 15.0 and above versions. Ensure that your project has a deemed valid developer signature.

Step 1: activate the service

See [Activate Service \(TUILiveKit\)](#) to claim the experience version or activate the paid edition.

Step Two: Import the LiveStreamCore Component

In the root directory of the project, run the following command to install the `live_stream_core` plug-in via the command line.

```
flutter pub add live_stream_core
```

If you encounter any problems during integration or use, feel free to [give us feedback](#).

Step Three: Sign In

Add the following code to your project. It enables the TUI component to log in by calling the login-related APIs in `RTCRoomEngine`. This step is crucial. Only after successful login can you properly use the features provided by `LiveStreamCore`.

```
final result = await TUIRoomEngine.login(  
    'Replace with your activated SDKAppID',  
    'Replace with your userId',  
    'Replace with your userSig');
```

Parameter Description

Here we detail the key parameters required in the login function:

Parameter	Type	Description
SDKAppID	int	You have already obtained it in the last step of Step 1, so it will not be elaborated here.
UserID	String	The ID of the current user, in string format, only allows letters (a-z and A-Z), digits (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in step 3 of Step One to encrypt information such as SDKAppID and UserID to generate a UserSig. It's a credential used for authentication purposes, allowing Tencent Cloud to identify if the current user is authorized to use the TRTC service. You can generate a temporary UserSig through the Auxiliary Tools in the console. For more information, please refer to How to Calculate and Use UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate userSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering, and once your key is leaked, attackers can steal your Tencent Cloud traffic.

Production Environment: If your project is going to be launched, please adopt the method of [Server-side Generation of UserSig](#).

Step 4: Use the Core Component to Implement the Live Voice Room Functionality

Note:
During use, if you have any queries about the API call methods, you can refer to [Example](#).

Create a Core Component Instance for Voice Chat Room

You need to create a controller `SeatGridController` first, and then assign it to the voice chat room core component `SeatGridWidget` .
`SeatGridController` is responsible for providing APIs. `SeatGridWidget` is used to display the seat UI.
You can add `SeatGridWidget` anywhere you need to display the seat UI.

```
final controller = SeatGridController();
SeatGridWidget(controller: controller);
```

Host Enables Voice Chat Room and Audience Joins Voice Chat Room

Host Enables Voice Chat Room: Start a voice chat room and stream the data of local microphone capture to the voice chat room.

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

final roomInfo = TUIRoomInfo(roomId: 'replace with your roomId');
roomInfo.name = 'replace with your roomName';
roomInfo.seatMode = TUISeatMode.applyToTake;
roomInfo.isSeatEnabled = true;
roomInfo.roomType = TUIRoomType.livingRoom;
final startVoiceRoomResult = await controller.startVoiceRoom(roomInfo);

final startMicrophoneResult = await controller.startMicrophone();
```

Audience Joins Live Room

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

final result = await controller.joinVoiceRoom('replace with your roomId');
```

The host enables the voice chat room.	Audience joins a voice chat room.

Audience Mic Connection

If you need to implement the audience mic connection feature, see [Seat Management](#) document.

Set Microphone Position List Layout Arrangement

You can set the seat layout through the `setLayoutMode` API.

`setLayoutMode` API has 3 built-in layouts: focus layout (`focus`), grid layout (`grid`), and vertical layout (`vertical`). It also supports custom layout (`free`).

```
// API definition
void setLayoutMode(LayoutMode layoutMode, SeatWidgetLayoutConfig? layoutConfig);

// API utilization
import 'package:live_stream_core/live_stream_core.dart';

// Set grid layout
controller.setLayoutMode(LayoutMode.grid, null);

// Set focus layout
controller.setLayoutMode(LayoutMode.focus, null);

// Set vertical layout
controller.setLayoutMode(LayoutMode.vertical, null);

// Set custom layout
final rowConfig = SeatWidgetLayoutRowConfig(
  count: 2,
  seatSpacing: 20.0,
  seatSize: const Size(80, 80),
  alignment: SeatWidgetLayoutRowAlignment.spaceBetween);
final layoutConfig = SeatWidgetLayoutConfig(
  rowConfigs: [rowConfig, rowConfig]);
controller.setLayoutMode(LayoutMode.free, layoutConfig);
```

Note:

The parameter settings of the custom layout can be viewed in the parameter descriptions in [SeatWidgetLayoutRowConfig](#). Among them, the alignment mode "alignment" can be referred to in the description in [SeatWidgetLayoutRowAlignment](#). The effect of the alignment mode can be seen in the schematic diagram .

Grid layout	Element layout	Vertical layout	Custom layout

--	--	--	--

Custom Layout Alignment Mode Diagram

Custom Seat View

If you consider our default UI does not meet your requirements and you want to customize your own seat UI, you can implement the UI style of the specified seat through the `seatWidgetBuilder` parameter of `SeatGridWidget` .

```
// Definition of seatWidgetBuilder
typedef SeatWidgetBuilder = Widget Function(
  BuildContext context,
  ValueNotifier<TUISeatInfo> seatInfoNotifier,
  ValueNotifier<int> volumeNotifier);

// Usage example
import 'package:live_stream_core/live_stream_core.dart';
import 'package:rtc_room_engine/rtc_room_engine.dart';

SeatGridWidget (
  controller: controller,
  onSeatWidgetTap: (TUISeatInfo seatInfo) {
    // debugPrint('click seatWidget index:${seatInfo.index}');
  },
  seatWidgetBuilder: (
    BuildContext context,
    ValueNotifier<TUISeatInfo> seatInfoNotifier,
    ValueNotifier<int> volumeNotifier) {
    // Return your custom seat component
    return Container();
  }
)
```

Before customizing seat layout	After customizing seat layout

Microphone Management (TUILiveKit)

Last updated : 2025-04-17 11:51:50

Description of the Feature

Seat Management is a real-time interaction and communication method where anchors can interact with the audience on the seat in real-time. Whether it's answering questions, sharing experiences, or entertainment interactions, it greatly enhances the audience's **sense of participation** and **satisfaction**. This direct interaction and communication provide more convenient and efficient channels for commercial operations. Using this feature, you can achieve applying to join the seat, inviting to take seat, moving seat positions, kicking someone off the seat, and locking seats, greatly enriching the playability of the voice chat room.

Seat List	Audience applies to take seat	Anchor Handling Seat Request

Apply To take Seat Process

Audience applies to take seat process

The TUILiveKit audience application for seat feature is mainly implemented through SeatGridView. You can call the following API Function to achieve the audience application for seat feature. The implementation is as follows, with Audience B applying for a seat as an example.

Audience sends a request to join the seat

Audience B sends a request to join the seat to Anchor A. Anchor A will receive the join request from Audience B in the onSeatRequestReceived Callback.

- Android
- iOS
- Flutter
- Kotlin
- Java

```
val seatIndex = 1;
val timeout = 60;
seatGridView.takeSeat(seatIndex, timeout, object : VoiceRoomDefine.RequestCallback
```

```
        override fun onAccepted(userInfo: TUIRoomDefine.UserInfo) {
            Log.i(TAG, "Application for speaking is approved")
        }

        override fun onRejected(userInfo: TUIRoomDefine.UserInfo) {
            Log.i(TAG, "Application for speaking is rejected")
        }

        override fun onCancelled(userInfo: TUIRoomDefine.UserInfo) {
            Log.i(TAG, "Application for speaking is canceled")
        }

        override fun onTimeout(userInfo: TUIRoomDefine.UserInfo) {
            Log.i(TAG, "Application for speaking times out")
        }

        override fun onError(userInfo: TUIRoomDefine.UserInfo, error: TUICommonDefine.E
            Log.i(TAG, "Application for speaking error")
        }
    })

    int seatIndex = 1;
    int timeout = 60;
    seatGridView.takeSeat(seatIndex, timeout, new VoiceRoomDefine.RequestCallback() {
        @Override
        public void onAccepted(TUIRoomDefine.UserInfo userInfo) {
            Log.i(TAG, "Application for speaking is approved");
        }

        @Override
        public void onRejected(TUIRoomDefine.UserInfo userInfo) {
            Log.i(TAG, "Application for speaking is rejected");
        }

        @Override
        public void onCancelled(TUIRoomDefine.UserInfo userInfo) {
            Log.i(TAG, "Application for speaking is canceled");
        }

        @Override
        public void onTimeout(TUIRoomDefine.UserInfo userInfo) {
            Log.i(TAG, "Application for speaking times out");
        }

        @Override
        public void onError(TUIRoomDefine.UserInfo userInfo, TUICommonDefine.Error erro
```

```
        Log.i(TAG, "Application for speaking error");
    }
});
```

swift

```
import RTCRoomEngine
import LiveStreamCore

let seatIndex = 1
let timeout = 60
seatGridView.takeSeat(index: index, timeout: timeout) { userInfo in
    print("Application for speaking is approved")
} onRejected: { userInfo in
    print("Application for speaking is rejected")
} onCancelled: { userInfo in
    print("Application for speaking is canceled")
} onTimeout: { userInfo in
    print("Application for speaking times out")
} onError: { userInfo, code, message in
    print("Application for speaking error")
}
```

Dart

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

const int seatIndex = 1;
const int timeout = 60;
final controller = SeatGridController();
final result = await controller.takeSeat(seatIndex, timeout);
if (result.code == TUIError.success) {
    switch (result.type) {
        case RequestResultType.onAccepted:
            debugPrint('Request to speak approved');
            break;
        case RequestResultType.onRejected:
            debugPrint('Application for speaking is rejected');
            break;
        case RequestResultType.onCancelled:
            debugPrint('Application for speaking is canceled');
            break;
        case RequestResultType.onTimeout:
            debugPrint('Request to speak timeout');
            break;
        default:
```



```

        break;
    }
} else {
    debugPrint('Error in requesting to speak');
}

```

The TUILiveKit audience application for seat feature is mainly implemented through SeatGridView. You can call the following API Function to achieve the audience application for seat feature. The implementation is as follows, with Audience B applying for a seat as an example.

Note:

The room owner will only receive a request to speak when the room mode is APPLY_TO_TAKE, in FREE_TO_TAKE mode, takeSeat will directly succeed in taking the seat.

Host end receives a Request to Speak

Anchor A will receive Audience B's request to join the seat in the onSeatRequestReceived callback method.

Android

iOS

Flutter

Kotlin

Java

```

override fun onSeatRequestReceived(type: VoiceRoomDefine.RequestType, userInfo: TUI
    if (type == VoiceRoomDefine.RequestType.APPLY_TO_TAKE_SEAT) {
        Log.i(TAG, "Received request to join the seat from the audience: ${userInfo.u
    }
}

@Override
public void onSeatRequestReceived(VoiceRoomDefine.RequestType type, TUIRoomDefine.U
    if (type == VoiceRoomDefine.RequestType.APPLY_TO_TAKE_SEAT) {
        Log.i(TAG, "Received request to join the mi from the audience: " + userInfo
    }
}

```

swift

```

import RTCRoomEngine
import LiveStreamCore

func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo) {
    if type == .applyToTakeSeat {
        print("Received audience's request to join the seat: \"\(userInfo.userId)\"")
    }
}

```

Dart

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

// Create a listening Observer instance
final controller = SeatGridController();
final exampleObserver = ExampleObserver();

// Add an observer
controller.addObserver(exampleObserver);

class ExampleObserver extends SeatGridWidgetObserver {
  ExampleObserver() {
    super.onSeatRequestReceived = (type, userInfo) {
      if (type == RequestType.applyToTakeSeat) {
        debugPrint('Receive an audience's request to become a speaker: ${userInfo}');
      }
    };
  }
}
```

Note:

The room owner will only receive a request to take seat when the room mode is APPLY_TO_TAKE, in FREE_TO_TAKE mode, takeSeat will directly succeed in taking the seat.

Anchor responds to the request to take the seat

After Anchor A receives Audience B's request to join the seat, they can call responseRemoteRequest to respond whether they agree to let Audience B take seat. Audience B will receive a callback of Anchor A's acceptance or rejection (onAccepted/onRejected).

Android

iOS

Flutter

Kotlin

Java

```
// Anchor agrees to let the audience take the seat
seatGridView.responseRemoteRequest(userId, true, null);

// Anchor rejects the audience's request to take the seat
seatGridView.responseRemoteRequest(userId, false, null);

// Anchor agrees to let the audience take the seat
```

```
seatGridView.responseRemoteRequest(userId, true, null);

// Anchor rejects the audience's request to take the seat
seatGridView.responseRemoteRequest(userId, false, null);
```

swift

```
import RTCRoomEngine
import LiveStreamCore

// Anchor agrees to let the audience take the seat
seatGridView.responseRemoteRequest(userId, true) {
} onError: { code, message in
}

// Anchor rejects the audience's request to take the seat
seatGridView.responseRemoteRequest(userId, false) {
} onError: { code, message in
}
```

Dart

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

final controller = SeatGridController();

const String userId = 'Replace with the userId of userB';

// Broadcaster agrees to let the audience become a speaker
final result = await controller.responseRemoteRequest(userId, true);

// Broadcaster denies audience to become a speaker
final result = await controller.responseRemoteRequest(userId, false);
```

Callback for seat information changes

Android

iOS

Flutter

If you have already set up the [Custom Seat View](#), you can listen to the `updateSeatView` callback to refresh your custom UI.

Kotlin

Java

```
override fun updateSeatView(seatGridView: SeatGridView, seatInfo: TUIRoomDefine.Sea
```

```

    Log.i(TAG, "Seat information changes");
}

@Override
public void void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo s
    Log.i(TAG, "Seat information changes");
}

```

If you have already set up the [Custom Seat View](#), you can listen to the `updateSeatView` callback to refresh your custom seat UI.

swift

```

import RTCRoomEngine
import LiveStreamCore

func seatGridView(_ view: SeatGridView, updateSeatView seatInfo: TUISeatInfo, seatV
    print("Seat information changes")
}

```

When the seat information changes, the corresponding value of the `seatInfoNotifier` parameter of `SeatWidgetBuilder` will change. You can use your custom seat component as a child of `ValueListenableBuilder` so that your custom seat UI can be updated instantly.

Dart

```

// Definition of seatWidgetBuilder
typedef SeatWidgetBuilder = Widget Function(
    BuildContext context,
    ValueNotifier<TUISeatInfo> seatInfoNotifier,
    ValueNotifier<int> volumeNotifier);

// Usage example
import 'package:live_stream_core/live_stream_core.dart';
import 'package:rtc_room_engine/rtc_room_engine.dart';

SeatGridWidget(
  controller: controller,
  seatWidgetBuilder: (
    BuildContext context,
    ValueNotifier<TUISeatInfo> seatInfoNotifier,
    ValueNotifier<int> volumeNotifier) {
    return ValueListenableBuilder(
      valueListenable: seatInfoNotifier,
      builder: (context, seatInfo, _) {
        // Return your custom seat component
        return Container();
      }
    );
  }
)

```

```

        }
    };
}
)

```

Anchor invites audience to speak flow

Anchor sends invite speaking request

Anchor A sends an invite speaking request to Audience C. Audience C will receive the invite speaking request from Anchor A in the onSeatRequestReceived Callback.

Android

iOS

Flutter

Kotlin

Java

```

val seatIndex = 1;
val userId = "userIdC";
val timeout = 60;
seatGridView.takeUserOnSeatByAdmin(seatIndex, timeout, userId, object : VoiceRoomDe
    override fun onAccepted(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "Invitation to Become a Speaker Accepted")
    }

    override fun onRejected(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "Invitation to Become a Speaker Rejected")
    }

    override fun onCancelled(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "Invitation to Become a Speaker Cancelled")
    }

    override fun onTimeout(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "Invitation to Become a Speaker Timed Out")
    }

    override fun onError(userInfo: TUIRoomDefine.UserInfo, error: TUICommonDefine.E
        Log.i(TAG, "Invitation to Become a Speaker Error")
    }
}))

val seatIndex = 1;
val userId = "userIdC";
val timeout = 60;

```

```
seatGridView.takeUserOnSeatByAdmin(seatIndex, userId, timeout, new VoiceRoomDefine.  
    @Override  
    public void onAccepted(TUIRoomDefine.UserInfo userInfo) {  
        Log.i(TAG, "Invitation to Become a Speaker Accepted");  
    }  
  
    @Override  
    public void onRejected(TUIRoomDefine.UserInfo userInfo) {  
        Log.i(TAG, "Invitation to Become a Speaker Rejected");  
    }  
  
    @Override  
    public void onCancelled(TUIRoomDefine.UserInfo userInfo) {  
        Log.i(TAG, "Invitation to Become a Speaker Cancelled");  
    }  
  
    @Override  
    public void onTimeout(TUIRoomDefine.UserInfo userInfo) {  
        Log.i(TAG, "Invitation to Become a Speaker Timed Out");  
    }  
  
    @Override  
    public void onError(TUIRoomDefine.UserInfo userInfo, TUICommonDefine.Error error)  
        Log.i(TAG, "Invitation to Become a Speaker Error");  
    }  
});
```

swift

```
import RTCRoomEngine  
import LiveStreamCore  
  
let seatIndex = 1  
let userId = "userIdC"  
  
seatGridView.takeUserOnSeatByAdmin(index: seatIndex, timeout: timeout, userId: user  
    print("Invitation to join the microphone is approved")  
} onRejected: { userInfo in  
    print("Invitation to join the microphone is rejected")  
} onCancelled: { userInfo in  
    print("Invitation to join the microphone is canceled")  
} onTimeout: { userInfo in  
    print("Invitation to join the microphone times out")  
} onError: { userInfo, code, message in  
    print("Invitation to join the microphone error")  
}
```

Dart

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

const int seatIndex = 1;
const String userId = 'userIdC'
const int timeout = 60;
final controller = SeatGridController();
final result = await controller.takeUserOnSeatByAdmin(seatIndex, userId, timeout);
if (result.code == TUIError.success) {
  switch (result.type) {
    case RequestResultType.onAccepted:
      debugPrint('Invitation to become a speaker accepted');
      break;
    case RequestResultType.onRejected:
      debugPrint('Invitation to become a speaker rejected');
      break;
    case RequestResultType.onCancelled:
      debugPrint('Invitation to become a speaker cancelled');
      break;
    case RequestResultType.onTimeout:
      debugPrint('Invitation to become a speaker timed out');
      break;
    default:
      break;
  }
} else {
  debugPrint('Error in invitation to become a speaker');
}
```

Audience Side received an Invite Speaking Request

Audience C will receive Anchor A's invite to join the seat in the onSeatRequestReceived callback method.

Android

iOS

Flutter

Kotlin

Java

```
override fun onSeatRequestReceived(type: VoiceRoomDefine.RequestType, userInfo: TUI
    if (type == VoiceRoomDefine.RequestType.INVITE_TO_TAKE_SEAT) {
        Log.i(TAG, "Received invite to join the seat from the anchor: ${userInfo.user
    }
}
```

```

@Override
public void onSeatRequestReceived(VoiceRoomDefine.RequestType type, TUIRoomDefine.U
    if (type == VoiceRoomDefine.RequestType.INVITE_TO_TAKE_SEAT) {
        Log.i(TAG, "Received invite to join the seat from the anchor: " + userInfo.
    }
}

```

swift

```

import RTCRoomEngine
import LiveStreamCore

func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo) {
    if type == .inviteToTakeSeat {
        print("Received invite to join the seat from the anchor: \"\(userInfo.userId)\"
    }
}

```

Dart

```

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

// Create a listening Observer instance
final controller = SeatGridController();
final exampleObserver = ExampleObserver();

// Add an observer
controller.addObserver(exampleObserver);

class ExampleObserver extends SeatGridWidgetObserver {
    ExampleObserver() {
        super.onSeatRequestReceived = (type, userInfo) {
            if (type == RequestType.inviteToTakeSeat) {
                debugPrint('Receive an invitation to become a speaker from the anchor: $
            }
        };
    }
}

```

Audience responds to the invite to speak

After Audience C receives the join request from Anchor A, they can call `responseRemoteRequest` to respond whether they agree to speak. Anchor A will receive Audience C's acceptance or rejection (`onAccepted/onRejected`) callback.

Android

iOS

Flutter

Kotlin

Java

```
// Audience agrees to the anchor's invite
seatGridView.responseRemoteRequest("", true, null);

// Audience rejects the anchor's invite
seatGridView.responseRemoteRequest("", false, null);

// Audience agrees to the anchor's invite
seatGridView.responseRemoteRequest("", true, null);

// Audience rejects the anchor's invite
seatGridView.responseRemoteRequest("", false, null);
```

swift

```
import RTCRoomEngine
import LiveStreamCore

// Audience agrees to the anchor's invite
seatGridView.responseRemoteRequest("userId of anchor", true) {
} onError: { code, message in
}

// Audience rejects the anchor's invite
seatGridView.responseRemoteRequest("userId of anchor", false, null) {
} onError: { code, message in
}
```

Dart

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

final controller = SeatGridController();

const String userId = 'userId of anchor';

// Audience accepts the anchor's invitation
final result = await controller.responseRemoteRequest(userId, true);

// Audience rejects the anchor's invitation
final result = await controller.responseRemoteRequest(userId, false);
```

Note:

If using a custom seat view, both anchors and the audience can refresh the seat UI by listening to the [callback for seat information changes](#).

Leave the Seat Process

After the audience successfully becomes a speaker, they proactively become a listener again.

After Audience B becomes a speaker successfully, they can call `leaveSeat` to leave the mic voluntarily.

Android

iOS

Flutter

Kotlin

Java

```
seatGridView.leaveSeat()

seatGridView.leaveSeat();
```

swift

```
import RTCRoomEngine
import LiveStreamCore

seatGridView.leaveSeat() {
} onError: { code, message in
}
```

Dart

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

final controller = SeatGridController();
final result = await controller.leaveSeat();
```

After the audience successfully joins the stage, the host kicks them off stage

After Audience B becomes a speaker successfully, Anchor A can kick Audience B off.

Android

iOS

Flutter

Kotlin

Java

```
val userId = "userIdB"
```

```
seatGridView.kickUserOffSeatByAdmin(userId, null)

String userId = "userIdB";
seatGridView.kickUserOffSeatByAdmin(userId, null);
```

swift

```
import RTCRoomEngine
import LiveStreamCore

let userId = "userIdB"
seatGridView.kickUserOffSeatByAdmin(userId) {
} onError: { code, message in
}
```

Dart

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

const String userId = "userIdB";
final controller = SeatGridController();
final result = await controller.kickUserOffSeatByAdmin();
```

Audience receives a callback when the host kicks them off stage

After Anchor A kicks Audience B off, Audience B will receive the onKickedOffSeat callback.

Android

iOS

Flutter

Kotlin

Java

```
override fun onKickedOffSeat(inviterUser: UserInfo) {
    Log.i(TAG, "Anchor kicked off the seat")
}

@Override
public void onKickedOffSeat(TUIRoomDefine.UserInfo userInfo) {
    Log.i(TAG, "Anchor kicked off the seat");
}
```

swift

```
import RTCRoomEngine
import LiveStreamCore
```

```
func onKickedOffSeat(userInfo: TUIUserInfo) {  
    print("Anchor kicked off the seat")  
}
```

Flutter

```
import 'package:live_stream_core/live_stream_core.dart';  
import 'package:rtc_room_engine/rtc_room_engine.dart';  
  
// Create a listening Observer instance  
final controller = SeatGridController();  
final exampleObserver = ExampleObserver();  
  
// Add an observer  
controller.addObserver(exampleObserver);  
  
class ExampleObserver extends SeatGridWidgetObserver {  
    ExampleObserver() {  
        super.onKickedOffSeat = (userInfo) {  
            debugPrint('Anchor kicks off the mic');  
        };  
    }  
}
```

Note:

If using a custom seat view, both anchors and the audience can refresh the seat UI by listening to the [callback for seat information changes](#).

Lock Seat Process

Position Lock

The anchor can lock a specific seat, and it will be blocked, preventing any seat operations.

Android

iOS

Flutter

Kotlin

Java

```
val index = 1;  
val isLockSeat = true  
val params = TUIRoomDefine.SeatLockParams().apply {  
    lockSeat = isLockSeat  
}
```

```
seatGridView.lockSeat(index, params, null)

int index = 1;
bool isLockSeat = true;
TUIRoomDefine.SeatLockParams params = new TUIRoomDefine.SeatLockParams();
params.lockSeat = isLockSeat;
seatGridView.lockSeat(index, params, null);
```

swift

```
import RTCRoomEngine
import LiveStreamCore

let index = 1
let isLockSeat = true

let params = TUISeatLockParams()
params.lockSeat = isLockSeat

seatGridView.lockSeat(index: index, lockMode: params) {
} onError: { code, message in
}
```

Dart

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

const int index = 1;
const bool isLockSeat = true
final params = TUISeatLockParams();
params.lockSeat = isLockSeat;
final controller = SeatGridController();
final result = await controller.lockSeat(index, params);
```

Audio Lock

The anchor can lock the audio for a specific seat position, and the user on that position will be muted.

Android

iOS

Flutter

Kotlin

Java

```
val index = 1;
```

```
bool isAudioLocked = true;
val params = TUIRoomDefine.SeatLockParams().apply {
    isAudioLocked = isLockSeat
}
seatGridView.lockSeat(index, params, null)

int index = 1;
bool isAudioLocked = true;
TUIRoomDefine.SeatLockParams params = new TUIRoomDefine.SeatLockParams();
params.lockAudio = seatInfo.isAudioLocked;
seatGridView.lockSeat(index, params, null);
```

swift

```
import RTCRoomEngine
import LiveStreamCore

let index = 1
let isAudioLocked = true

let params = TUISeatLockParams()
params.lockAudio = isAudioLocked

seatGridView.lockSeat(index: index, lockMode: params) {
} onError: { code, message in
}
```

Dart

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

const int index = 1;
const bool isAudioLocked = true
final params = TUISeatLockParams();
params.lockAudio = isAudioLocked;
final controller = SeatGridController();
final result = await controller.lockSeat(index, params);
```

Note:

If using a custom seat view, both anchors and the audience can refresh the seat UI by listening to the [callback for seat information changes](#).

Other seat management feature extensions

Android

iOS

Flutter

The seat management feature is implemented based on `SeatGridView`. If you need to extend the seat management feature, please refer to the [SeatGridView](#) documentation.

The seat management feature is implemented based on `SeatGridView`. If you need to extend the seat management feature, please refer to the [SeatGridView](#) documentation.

The seat management feature is implemented based on `SeatGridView`. If you need to extend the seat management feature, please refer to the [SeatGridWidget](#) documentation.

Room List (TUILiveKit)

Last updated : 2025-04-15 15:10:40

Description of the Feature

TUILiveKit already supports the room list UI component. The room list component helps display all the online live streaming and voice chat rooms in your application. After integrating the room list UI component, you can simply click on a live streaming room in the list to watch the current broadcast in real-time. Once in the live streaming room, you can interact with the host in real-time through danmaku, gifts, mic connection, and other features.

Room list component	Watch live streaming	Mic connection with the host

Feature Integration

Android

iOS

Flutter

Note:

Please make sure to complete the log in to operation as per [Quick Integration \(TUILiveKit\)](#) . You can only successfully enter the live preview screen after `TUILogin.login` log in to is successful.

1. Add a `FrameLayout` layout to the XML where the Room List UI Components need to be displayed.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_live_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

2. By loading the page of TUILiveKit's `TUILiveListFragment` onto the XML definition layout, you can display the Room List.

Java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_main);

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        TUILiveListFragment listFragment = new TUILiveListFragment();
        fragmentTransaction.add(R.id.fl_live_list, listFragment);
        fragmentTransaction.commit();
    }
}
```

Note:

Please make sure that you have completed the [Quick Access](#) log in to operation. You can only enter the live preview screen normally after the `TUILogin.login` log in to is successful.

Swift

Objective-C

```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    // Enter Room List
    let liveListViewController = TUILiveListViewController()
    self.navigationController?.pushViewController(viewController, animated: true)
}

//
//  MainViewController.m
//

#import <TUILiveKit/TUILiveKit-Swift.h>

- (void)buttonTapped:(UIButton *)sender {
    // Enter Room List
    TUILiveListViewController *liveListViewController = [[TUILiveListViewController
    [self.navigationController pushViewController:liveListViewController animated:t
```

```
}
```

Note:

Please make sure you have completed the log in operation as described in [Quick Integration \(TUILiveKit\)](#) . Only after a successful log in can you normally enter the live preview screen.

In your widget, you can display the room list by loading the `LiveListWidget` component of TUILiveKit.

Dart

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

return Scaffold(
  body: SizedBox(
    width: _screenWidth,
    height: double.infinity,
    child: LiveListWidget(), // Adding the room list component LiveListWidget of
  ),
);
```

Feature Customization

If the current UI does not meet your needs, you can modify the source code to achieve the UI effect you are satisfied with. To facilitate your customization of the UI, here is an introduction to the files related to the room list.

Android**iOS****Flutter**

```
// File Location: Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/ui

roomlist // Directory for the implementation o
├── service // Service layer directory for the Li
│   └── RoomListService.java // Specific implementation of the ser
├── store // Data layer directory for the Live
│   └── RoomListState.java // Data encapsulation class for the L
└── view // View layer directory for the Live
    ├── adapter // Adapter directory for the view lay
    │   ├── LoadMoreAdapterWrapper.java // Adapter for adding pull-to-load-mo
    │   └── RoomListAdapter.java // Live room list adapter for the Liv
    ├── ListAudienceActivity.java // Streaming page triggered by clicki
    └── RoomListView.java // Implementation of live room list v
```

```
// File Location: iOS/TUILiveKit/Sources/Component/LiveList/

├─ LiveList                                // Directory for implementation of the
│   └─ Service                            // Service Layer Directory for the Live
│       └─ LiveListService.swift          // Concrete implementation of the Servi
│   └─ Store                              // Data Layer Directory for the Live Ro
│       └─ LiveListActions.swift          // Event Definition Class for the Live
│       └─ LiveListReducer.swift          // Event Response Class for the Live Ro
│       └─ LiveListSelectors.swift        // Data Selector Class for the Live Roo
│       └─ LiveListState.swift            // Data Definition Class for the Live R
│       └─ LiveListStore.swift            // Data Driver and Event Dispatch Proto
│   └─ View                               // View Layer Directory for the Live Ro
│       └─ LiveListCell.swift              // Custom Cell for the Live Room List C
│       └─ LiveListRootView.swift         // Root View for the Live Room List Com

// File Location: TUILiveKit/Flutter/livekit/lib/component/

room_list                                // Directory for implementation of
├─ service                               // Service Layer Directory for the
│   └─ room_list_service.java             // Concrete implementation of the S
├─ store                                 // Data Layer Directory for the Liv
│   └─ room_list_state.java               // Specific encapsulation class for
└─ view                                 // View Layer Directory for the Liv
    └─ room_list_view.java                // Implementation of the Live Room
```

Key code

Get Live Room List Plugin

Android

iOS

Flutter

Java

```
// File Location: Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/ui

private final TUILiveListManager mTUILiveListManager;
mTUILiveListManager = (TUILiveListManager) TUIRoomEngine.sharedInstance().getExtens
```

Swift

```
// File Location: iOS/TUILiveKit/Source/Component/LiveList/Service/LiveListService.
```

```
let listManager = roomEngine.getExtension(extensionType: .liveListManager) as? TUIL
```

Dart

```
// File Location: TUILiveKit/Flutter/livekit/lib/component/room_list/service/room_l

late final TUILiveListManager _liveListManager = TUIRoomEngine.sharedInstance().get
```

Get Live Room List

Android

iOS

Flutter

Java

```
// File Location: Android/tuilivekit/src/Main/java/com/trtc/uikit/livekit/common/ui

private static final int FETCH_LIST_COUNT = 20;
public String cursor = "";
mTUILiveListManager.fetchLiveList(cursor, FETCH_LIST_COUNT, new LiveInfoListCallbac
    @Override
    public void onSuccess(LiveInfoListResult result) {
    }

    @Override
    public void onError(TUICommonDefine.Error error, String s) {
    }
});
```

Swift

```
// File Location: iOS/TUILiveKit/Source/Component/LiveList/Service/LiveListService.

func getLiveList(cursor: String, count: Int = 20) -> AnyPublisher<(String, [TUILive
return Future<(String, [TUILiveInfo]), InternalError> { [weak self] promise in
    guard let self = self else { return }
    guard let listManager = roomEngine.getExtension(extensionType: .liveListManag
    promise(.failure(InternalError(error:TUIError.failed, message: "getRoomList
return
    }
    listManager.fetchLiveList(cursor: cursor, count: count) { cursor, liveInfoLis
    promise(.success((cursor, liveInfoList)))
    } onError: { error, message in
    promise(.failure(InternalError(error: error, message: message)))
    }
}.eraseToAnyPublisher()
```

```
}
```

Dart

```
// File Location: TUILiveKit/Flutter/livekit/lib/component/room_list/service/room_l

Future<void> _fetchLiveList() async {
  final String cursor = roomListState.cursor;
  TUIValueCallBack<TUILiveListResult> result = await _liveListManager.fetchLiveList
  if (result.code != TUIError.success) {
    ErrorHandler.onError(result.code);
    roomListState.loadStatus.value = false;
    roomListState.refreshStatus.value = false;
    roomListState.isHaveMoreData.value = false;
  } else {
    final liveListResult = result.data as TUILiveListResult;
    roomListState.liveInfoList.value = liveListResult.liveInfoList;
    roomListState.cursor = liveListResult.cursor;
    roomListState.loadStatus.value = false;
    roomListState.refreshStatus.value = false;
    roomListState.isHaveMoreData.value = liveListResult.cursor.isNotEmpty;
  }
}
```

Follow Anchors (TUILiveKit)

Last updated : 2024-12-20 10:55:37

Description of the Feature

In live streaming scenarios, the Follow the host feature is an important interaction method. It allows viewers to choose to follow specific hosts, thus establishing long-term interaction and follow relationships on the live streaming platform. The TUILiveKit has implemented the Follow the host feature through Chat.

Follow feature allows users to choose other users they are interested in so that they can get the latest updates, posts, or event information promptly. You can provide personalized content recommendations based on the user's Follow list. **Fans** feature refers to the status of a user being followed by others. When User A follows User B, A becomes a Fan of B.

Through the Follow and Fans features, social applications and websites can create an active and interconnected user network, promoting the dissemination of information and the building of communities.

Use Instructions

Note:

TUILiveKit's **Follow the host feature** relies on the **Chat Follow & Fan APIs**, so you need to activate the related Package bundles for Chat. For details, please refer to the [Follow&Fan](#) documentation.

Unfollowed Status	Followed Status	Broadcast



Follow the host: You can click the



button in the live room information area at the top left corner of the live streaming interface to follow the host

Unfollow: You can click the



button in the live room information area at the top left corner of the live streaming interface to unfollow the host

View the host's fan count: You can click the live room information area at the top left corner of the live streaming interface to pop up the live room details panel, which will display the host's **fan count**.

Feature customization

If the current UI does not meet your needs, you can achieve your desired UI effects by modifying the source code. For easier UI customization, an introduction to the files related to the follow feature is provided here.

Android

iOS

```
// File location: Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/ui
roominfo                                // Directory for implementing the live room informa
├── RoomInfoDetailView.java             // Specific implementation of the Live Room Informa
└── RoomInfoView.java                  // Specific implementation of the Live Room Informa

// File location: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/

LiveInfo                                // Implementation directory of Live Room Inform
├── LiveInfoView.swift                 // Detailed implementation of Live Room Informa
└── View
    ├── LiveInfoPanelView.swift       // Detailed implementation of Live Room Informa
```

Key code

Follow anchor

Android

iOS

Java

```
// File location: Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/service/i

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().followUser(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowOperationResult>>() {
        @Override
        public void onSuccess(List<V2TIMFollowOperationResult> results) {
        }

        @Override
        public void onError(int code, String message) {
        }
    });
```

Swift

```
// File location: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoServ

public func followUser(userId: String) {
    let userInfo = TUIUserInfo()
    userInfo.userId = userId
```



```

V2TIMManager.sharedInstance().followUser([userId]) { [weak self] followResultLi
    guard let self = self, let result = followResultList?.first else { return }
    if result.resultCode == 0, !self.state.followingList.contains(where: { $0.u
        self.state.followingList.insert(userInfo)
        self.getFansNumber()
    }
} fail: { code, message in
    debugPrint("[RoomInfo] followUser failed, error:\\(code), message:\\(String
}
}

```

Unfollow anchor

Android

iOS

Java

```

// File location: Android/tuiliblivekit/src/Main/java/com/trtc/uikit/livekit/service/i

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().unfollowUser(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowOperationResult>>() {
        @Override
        public void onSuccess(List<V2TIMFollowOperationResult> results) {
        }

        @Override
        public void onError(int code, String message) {
        }
    });

```

Swift

```

// File location: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoServ

public func unfollowUser(userId: String) {
    V2TIMManager.sharedInstance().unfollowUser([userId]) { [weak self] followResult
        guard let self = self, let result = followResultList?.first else { return }
        if result.resultCode == 0 {
            self.state.followingList = state.followingList.filter { $0.userId != us
        }
    } fail: { code, message in
        debugPrint("[RoomInfo] unfollowUser failed, error:\\(code), message:\\(Stri
    }
}

```

View anchor's follow status

Android

IOS

Java

```
// File location: Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/service/i

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().checkFollowType(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowTypeCheckResult>>() {
        @Override
        public void onSuccess(List<V2TIMFollowTypeCheckResult> results) {
        }

        @Override
        public void onError(int code, String message) {
        }
    });

// File location: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoServ

public func isFollow(userId: String) {
    let userInfo = TUIUserInfo()
    userInfo.userId = userId
    V2TIMManager.sharedInstance().checkFollowType([state.ownerId]) { [weak self] ch
        guard let self = self, let result = checkResultList?.first else { return }
        if result.followType == .FOLLOW_TYPE_IN_BOTH_FOLLOWERS_LIST || result.follo
            if !self.state.followingList.contains(where: { $0.userId == userId }) {
                self.state.followingList.insert(userInfo)
            }
        } else {
            self.state.followingList = state.followingList.filter { $0.userId != us
        }
    } fail: { code, message in
        debugPrint("[RoomInfo] isFollow failed, error:\\(code), message:\\(String(d
    }
}
```

Get number of followers

Android

iOS

Java

```
// File location: Android/tuiliblivekit/src/Main/java/com/trtc/uikit/livekit/service/i

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().getUserFollowInfo(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowInfo>>() {
        @Override
        public void onSuccess(List<V2TIMFollowInfo> results) {

        }

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

        }
    });

// File location: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoServ

public func getFansNumber() {
    V2TIMManager.sharedInstance().getUserFollowInfo([state.ownerId]) { [weak self]
        guard let self = self, let followInfo = followInfoList?.first else { return
        self.state.fansNumber = Int(followInfo.followersCount)
    } fail: { code, message in
        debugPrint("[RoomInfo] getFansNumber failed, error:\\(code), message:\\(Str
    }
}
```

Interactive Bullet Comments (TUILiveKit)

Last updated : 2025-04-15 15:10:40

Feature Introduction

The interactive barrage feature is an important real-time communication tool that supports various interactive methods. Users can input emoji and text messages in the barrage, enhancing the entertainment value of the messages and making the interaction experience more enjoyable and lively. Through this feature, the audience can engage in richer communication with anchors and other viewers during live broadcasts, enhancing the overall sense of participation and fun. The interactive barrage feature has been implemented in TUILiveKit through Chat.

- iOS
- Flutter
- Android

The barrage component provides 2 view components:

- `BarrageInputView` : A view for sending barrage messages, which can pull up the input interface when clicked.
- `BarrageStreamView` : A view for receiving barrage messages, which will display barrage messages on this view.

The effect is shown below:

Display	Input

The barrage component mainly provides 2 Widget components, a Widget for sending barrage messages (`BarrageSendWidget`) and a Widget for displaying barrage messages (`BarrageDisplayWidget`):

- `BarrageSendWidget` : A widget for sending barrage messages, which can pull up the input interface when clicked.

- `BarrageDisplayWidget` : After receiving barrage messages, it will display them on this widget.

The effect is shown below:

Display	Input

The barrage component provides 2 view components:

`BarrageInputView` : A view for sending barrage messages, which can pull up the input interface when clicked.

`BarrageStreamView` : A view for receiving barrage messages, which will display barrage messages on this view.

The effect is shown below:

Display	Input

Note:

1. Supports **system keyboard** and **emoji keyboard** switching.
2. To respect the copyright of emoji designs, TUILiveKit project does not include large emoji elements. Before official commercial launch, please replace them with other emoji packs that you have designed or own the copyright for. The default **yellow face emoji pack shown below is copyrighted by Tencent Cloud** and can be licensed for a fee. To obtain authorization, please [Submit a Ticket](#) to contact us.

Quick Connection

Note:

If you have already integrated the LiveKit component, the barrage component is integrated by default in LiveKit, and you can directly experience the barrage feature in LiveKit.

Adding the Barrage Component to the Project

iOS

Flutter

Android

1. Download the Barrage component from [GitHub](#) to your local machine, and copy the Barrage (TUILiveKit/Sources/Component/Barrage) directory to your project.
2. Copy the dependency component Common (TUILiveKit/Sources/Common) to your project.
3. Copy resource files TUIBarrage.xcassets (TUILiveKit/Resources/TUIBarrage.xcassets) to your project.
4. Copy the language file Localized (TUILiveKit/Resources/Localized) to your project.

The directory is shown below:

5. Add the following dependency to your Podfile:

```
pod 'RTCCCommon'  
pod 'TUICore'  
pod 'SnapKit'  
pod 'RTCRoomEngine'
```

See the figure below:

6. Run pod install

1. In the dependencies node of the pubspec.yaml file of the project engineering, add a dependency on **barrage**.

```
dependencies:  
  flutter:  
    sdk: flutter  
  flutter_localizations:  
    sdk: flutter  
  intl: ^0.19.0  
  # Add barrage dependency  
  live_uikit_barrage: ^1.0.0
```

2. Execute the `flutter pub get` command.

3. Configure multilingual support. Add multilingual support for the **gift** component to the

`localizationsDelegates` and `supportedLocal`

```
MaterialApp(localizationsDelegates: const [  
  ...BarrageLocalizations.localizationsDelegates,  
], supportedLocales: const [  
  ...BarrageLocalizations.supportedLocales,  
],  
// ...  
);
```

1. Download TUILiveKit from [GitHub](#) to your local machine.

2. Copy the `Android/tuillivekit/component/barrage` directory to your project.

3. Copy the `Android/tuillivekit/component/common` directory to your project.

4. And add the configuration to the `settings.gradle` file in your project:

```
include ':common'  
include ':barrage'
```

5. Add the dependency configuration to the `build.gradle` file of the project you need to import:

```
api project(':common')
api project(':barrage')
```

Access the send barrage messages component.

iOS

Flutter

Android

Create `BarrageInputView` at the position where you need to send barrage messages, which can pull up the input interface when clicked:

```
let barrageInputView = BarrageInputView(roomId: roomId)
addSubview(barrageInputView)
```

Construct the `BarrageSendController` and `BarrageSendWidget` objects where you need to access and send barrage messages, and add the constructed `BarrageSendWidget` object to your Widget tree. Sample code is as follows:

```
BarrageSendController _sendController = BarrageSendController(
    roomId: "liveRoomId",          /// liveRoomId   Replace with you
    ownerId: "liveOwnerId",        /// liveOwnerId Replace with you
    selfUserId: "selfUserId",      /// selfUserId  Replace with you
    selfName: "selfUserName";      /// selfUserName Replace with you
BarrageSendWidget(controller: _sendController);
```

Create `BarrageInputView` at the position where you need to send barrage messages, which can pull up the input interface when clicked:

```
BarrageInputView barrageInputView = new BarrageInputView(mContext);
barrageInputView.init(roomId);
mBarrageInputContainer.addView(barrageInputView);
```

Access the display barrage messages component.

iOS

Flutter

Android

Use `BarrageStreamView` to display barrage messages where needed:

```
let barrageDisplayView = BarrageStreamView(roomId: roomId)
addSubview(barrageDisplayView)
```

After obtaining the anchor information, set `ownerId` to differentiate the display effects between the anchor and the audience.

```
barrageDisplayView.setOwnerId(ownerId)
```

Construct the `BarrageDisplayController` and `BarrageDisplayWidget` objects where you need to access and display barrage messages, and add the constructed `BarrageDisplayWidget` object to your `Widget` tree. Sample code is as follows:

```
BarrageDisplayController _displayController = BarrageDisplayController(  
    roomId: "liveRoomId",          /// liveRoomId   Replace with you  
    ownerId: "liveOwnerId",        /// liveOwnerId  Replace with you  
    selfUserId: "selfUserId",      /// selfUserId   Replace with you  
    selfName: "selfUserName";      /// selfUserName Replace with you  
BarrageDisplayWidget(controller: _displayController);
```

Use `BarrageStreamView` to display barrage messages where needed:

```
BarrageStreamView barrageStreamView = new BarrageStreamView(mContext);  
// ownerId represents the ID of the host, which is used to differentiate the display  
barrageStreamView.init(roomId, ownerId);  
mLayoutBarrageContainer.addView(barrageStreamView);
```

Insert local barrage messages.

iOS

Flutter

Android

The barrage display component `BarrageStreamView` provides the `insertBarrages` API method for (batch) insertion of custom messages (such as gift messages, live room announcements). Custom messages with custom styles can achieve different display effects.

Sample code:

```
// Example 1: Insert a gift message in the barrage area  
let barrage = TUIBarrage()  
barrage.content = "gift"  
barrage.user.userId = sender.userId  
barrage.user.userName = sender.userName  
barrage.user.avatarUrl = sender.avatarUrl  
barrage.user.level = sender.level  
  
barrage.extInfo["TYPE"] = AnyCodable("GIFTMESSAGE")  
barrage.extInfo["gift_name"] = AnyCodable(gift.giftName)  
barrage.extInfo["gift_count"] = AnyCodable(giftCount)  
barrage.extInfo["gift_icon_url"] = AnyCodable(gift.imageUrl)  
barrage.extInfo["gift_receiver_username"] = AnyCodable(receiver.userName)
```



```
barrageDisplayView.insertBarrages([barrage])

// Example 2: Echo the message after sending a barrage
// Implement the delegate of BarrageInputView, BarrageInputViewDelegate
extension MyViewController: BarrageInputViewDelegate {
    func barrageInputViewOnSendBarrage(_ barrage: TUIBarrage) {
        barrageDisplayView.insertBarrages([barrage])
    }
}
```

Note:

`TUIBarrage` 's `extInfo` is a `Map` used to store custom data.

When you need to insert local barrage messages, you can call the `insertMessage` method of `BarrageDisplayWidget` to insert local messages. For example, after LiveKit detects an audience member entering the room, you can insert a barrage message indicating the audience member's entry. The sample code is as follows:

```
BarrageUser barrageUser = BarrageUser();
barrageUser.userId = "enterRoomUserId";           /// UserId information displayed
barrageUser.userName = "enterRoomUserName";       /// Nickname information display
barrageUser.avatarUrl = "enterRoomUserAvatar";    /// Avatar information displayed
barrageUser.level = "66";                         /// Level information displayed

Barrage barrage = Barrage();
barrage.user = barrageUser;
barrage.content = "enter the room";                /// Text content displayed on th
_displayController.insertMessage(barrage);
```

The barrage display component `BarrageStreamView` provides the `insertBarrages` API method for (batch) insertion of custom messages (such as gift messages, live room announcements). Custom messages with custom styles can achieve different display effects.

Sample code:

```
// Insert a gift message in the barrage area
Barrage barrage = new Barrage();
barrage.content = "gift";
barrage.user.userId = sender.userId;
barrage.user.userName = sender.userName;
barrage.user.avatarUrl = sender.avatarUrl;
barrage.user.level = sender.level;
barrage.extInfo.put(Constants.GIFT_VIEW_TYPE, GIFT_VIEW_TYPE_1);
barrage.extInfo.put(GIFT_NAME, barrage.giftName);
barrage.extInfo.put(GIFT_COUNT, giftCount);
barrage.extInfo.put(GIFT_ICON_URL, barrage.imageUrl);
barrage.extInfo.put(GIFT_RECEIVER_USERNAME, receiver.userName);
```

```
mBarrageStreamView.insertBarrages(barrage);
```

Note:

`Barrage` 's `extInfo` is a `Map` used to store custom data.

Custom barrage message

iOS

Flutter

Android

There are two default styles for barrage messages: normal barrage message style and custom message style.

If you need more message styles (such as **gift send echo**), you can implement the delegate method of `BarrageStreamView`, `BarrageStreamViewDelegate`:

```
class MyViewController: BarrageStreamViewDelegate {
    func barrageDisplayView(_ barrageDisplayView: BarrageStreamView, createCustomCe
        // Whether to use custom UI, return nil if not needed
        guard let type = barrage.extInfo["TYPE"], type.value as? String == "GIFTMES
            return nil
        }
        // Return custom message style UI (gift display)
        return CustomBarrageCell(barrage: barrage)
    }
}
```

When you need to display a custom barrage item for specific barrage messages, you can use the `setCustomBarrageBuilder` method of `BarrageDisplayWidget`. For example, the following code shows a custom barrage displaying red text:

```
/// 1. Define a custom barrage item builder
class GiftBarrageItemBuilder extends CustomBarrageBuilder {
    @override
    Widget buildWidget(BuildContext context, Barrage barrage) { /// Customize the Wid
        return const Text(
            barrage.content,
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.w700, color: Colors.red
        );
    }

    @override
    bool shouldCustomizeBarrageItem(Barrage barrage) { /// Determine if the curren
        if (barrage.extInfo.keys.isNotEmpty) {
            return true;
        }
        return false;
    }
}
```

```

    }
}

/// 2. Set `setCustomBarrageBuilder` for `BarrageDisplayWidget`
_displayController.setCustomBarrageBuilder(GiftBarrageItemBuilder());

```

There are two default styles for barrage messages: normal barrage message style and custom message style. The specific style is represented by an integer, and the style of a normal barrage message is 0.

If you need more message styles (such as gift send echo), you can implement the `BarrageItemTypeDelegate` of `BarrageStreamView`, and implement the new style `BarrageItemAdapter`.

The `BarrageItemTypeDelegate` proxy has been overridden to support the new style `GIFT_VIEW_TYPE_1`.

```

public static final int    GIFT_VIEW_TYPE_1        = 1;
public class BarrageViewTypeDelegate implements BarrageItemTypeDelegate {

    @Override
    public int getItemViewType(int position, Barrage barrage) {
        if (barrage.extInfo != null && barrage.extInfo.containsKey(GIFT_VIEW_TYPE))
            String viewTypeString = String.valueOf(barrage.extInfo.get(GIFT_VIEW_TY
            if (String.valueOf(GIFT_VIEW_TYPE_1).equals(viewTypeString)) {
                return GIFT_VIEW_TYPE_1;
            }
        }
        return 0;
    }
}

mBarrageStreamView.setItemTypeDelegate(new BarrageViewTypeDelegate());

```

Implement an adapter for a custom style and set it to a `GIFT_VIEW_TYPE_1` for the style.

```

public class GiftBarrageAdapter implements BarrageItemAdapter {
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent) {
        LinearLayout ll = new LinearLayout(mContext);
        ll.addView(new TextView(mContext));
        return new GiftViewHolder(ll, mDefaultGiftIcon);
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position, Barr
        GiftViewHolder viewHolder = (GiftViewHolder) holder;
        viewHolder.bind(barrage);
    }
    // GiftViewHolder
    ...
}

```

```
}  
  
mBarrageStreamView.setItemAdapter(GIFT_VIEW_TYPE_1, new GiftBarrageAdapter(mContext
```

FAQs

Timing of bullet screen component initialization

Since the controller of the barrage component needs to integrate some live room information parameters, it is necessary to load the barrage component after the **viewer enters the live room** or the **anchor creates the live room**.

Interactive Gifts (TUILiveKit)

Last updated : 2025-04-15 15:10:40

Feature Introduction

The Interactive Gifts component is a virtual gift interaction platform designed to enhance the social experience of users. With this component, users can send virtual gifts to their favorite live stream hosts to show their appreciation, favoritism, and support.

The Interactive Gift Component supports setting **like, gift sending panel, send gifts, play normal gift animation, play SVGA gift animation**, etc.

Flutter

Android

iOS

The interactive gift component provides 2 view components:

- `GiftSendWidget` : send gift message view, clicking it can bring up the gift panel.
- `GiftDisplayWidget` : receive gift message view, it will display gift messages on this view.

The effect is shown below:

Gift Display Panel	Normal Gift Playback Effect	Full-Screen Gift Playback Effect

The gift component mainly provides 2 APIs:

- `GiftListPanelView` : gift panel, presents the gift list, send gifts, and top-up.
- `GiftPlayView` : Panel for playing gifts, automatically listens to gift messages.

The effect is shown below:

Gift Display Panel	Normal Gift Playback Effect	Full-Screen Gift Playback Effect

The gift component mainly provides 2 APIs:

- `GiftListView` : gift panel, presenting the gift list and send gifts.
- `GiftPlayView` : A panel for playing gifts, automatically listens for gift messages.

The effect is shown below:

Gift Display Panel	Normal Gift Playback Effect	Full-Screen Gift Playback Effect

Quick Connection

Note:
If you have already integrated the LiveKit component, we have by default included the gift component in LiveKit. You can directly experience the interactive gift feature in LiveKit.

Adding the Gift Component To the Project

- Flutter
- Android
- iOS

1. Add **gift** dependency in the `dependencies` node of the `pubspec.yaml` file in the project engineering.

```
dependencies:  
  flutter:  
    sdk: flutter  
  flutter_localizations:  
    sdk: flutter  
  intl: ^0.19.0  
  # Add gift dependency  
  live_uikit_gift: ^1.0.0
```

2. Execute the `flutter pub get` command.

3. Configure multilingual support. Add multilingual support for the **gift** component to the `localizationsDelegates` and `supportedLocales` attributes of your application's `MaterialApp` .

```
MaterialApp(localizationsDelegates: const [  
  ...GiftLocalizations.localizationsDelegates,  
], supportedLocales: const [  
  ...GiftLocalizations.supportedLocales,  
],  
// ...
```

```
);
```

Integrate the LiveKit component for normal use, [see details on GitHub](#).

Integrate the LiveKit component for normal use, [see details on GitHub](#).

Integrating the Gift Sending Component

Flutter

Android

iOS

Construct `GiftSendController` and `GiftSendWidget` objects where you need to integrate the gift sending message, and add the constructed `GiftSendWidget` object to your `Widget` tree. Example code is as follows:

```
GiftSendController _giftSendController = GiftSendController(
  roomId: "liveRoomId",          /// liveRoomId   Replace with your live room ID
  owner: ownerInfo,              /// ownerInfo   Replace with your live room ho
  self: selfInfo,                /// selfInfo    Replace with your login user i
);

GiftSendWidget(controller: _giftSendController);
```

Implement the `GiftListPanelView` `OnSendGiftListener` callback `onSendGift` to get the number of gifts and gift information. After preprocessing, you can call the `GiftListPanelView` function `sendGift` for the actual sending of gifts.

```
public void onSendGift(GiftListPanelView giftListView, Gift gift, int giftCount) {
  //...Here is the pre-processing, such as verifying the current user's balance, e
  GiftUser receiver = new GiftUser();
  //...Here set the gift recipient information
  giftListView.sendGift(gift, giftCount, receiver);
}
```

Create a `GiftListView` object and add it to your view. You also need to implement the

`GiftListViewDelegate` callback method `onSendGift` .

When you click the send button on the `GiftListView` panel, you will receive this callback. You can get the number of gifts to be sent and the gift information in this method. After preprocessing, you can call the

`GiftListView` function `sendGift` for the actual sending of gifts.

```
lazy var giftListView = GiftListView(roomId: roomId, delegate: self)
// ...Add giftListView to your parent view and adjust the layout here

func onSendGift(gift model: TUIGift, giftCount: Int) {
  //... Preprocessing here, such as verifying the current user's balance
  let receiver = TUIGiftUser()
  //... Set gift recipient information here
```

```
giftListView.sendGift(model: giftModel, giftCount: giftCount, receiver: receive
}
```

Integrating the Gift Display Component

Flutter

Android

iOS

Construct `GiftDisplayController` and `GiftDisplayWidget` objects where you need to integrate the gift display message, and add the constructed `GiftDisplayWidget` object to your `Widget` tree. Example code is as follows:

```
GiftDisplayController _giftDisplayController = GiftDisplayController(
  roomId: "liveRoomId",          /// liveRoomId   Replace with your live room I
  owner: ownerInfo,             /// ownerInfo   Replace with your live room h
  self: selfInfo,               /// selfInfo    Replace with your login user
);

GiftDisplayWidget(controller: _giftDisplayController!);
```

The gift display component `GiftPlayView` itself will receive and play gift messages.

```
GiftPlayView giftPlayView = new GiftPlayView(mContext, roomId);
```

The gift display component `GiftPlayView` will receive and play gift messages.

You only need to create a `GiftPlayView` object and add it to your view.

```
let giftPlayView = GiftPlayView(groupId: roomId)
// ...Add giftPlayView to your parent view and adjust the layout here
```

Listening To Gift Sending and Receiving Messages

Flutter

Android

iOS

If you need to get callback information for sending and receiving gifts, you can call the

`GiftDisplayController` 's `setGiftCallback` method. Example code is as follows:

```
_giftDisplayController?.setGiftCallback(
  onReceiveGiftCallback: _onReceiveGiftCallback,    /// _onReceiveGiftCallback can
  onSendGiftCallback: _onSendGiftCallback,          /// _onSendGiftCallback can be
);
```


If you need to receive callback information for receiving gifts, implement the `GiftPlayView`

`TUIGiftPlayViewListener` function `onReceiveGift` .

```
public interface TUIGiftPlayViewListener {
    void onReceiveGift(Gift gift, int giftCount, GiftUser sender, GiftUser receiver
    //...
}
```

If you need to get callback information for receiving gifts, you can implement the `GiftPlayView` 's delegate

`GiftPlayViewDelegate` 's `giftPlayView:onReceiveGift:gift:giftCount:sender:receiver` function.

```
func giftPlayView(_ giftPlayView: GiftPlayView, onReceiveGift gift: TUIGift, giftCo
    // Custom Processing
}
```

Custom Gift Panel

Flutter

Android

iOS

If you need to modify the gift list of the gift panel, you can call the `GiftSendController` 's `setGiftList` method, as shown below:

```
List<GiftModel> giftList = [    /// Customize your gift list data source
    GiftModel(giftId: "1", giftName: "Egg", imageUrl: "giftImageUrl", price: 10),
    GiftModel(giftId: "2", giftName: "Star", imageUrl: "giftImageUrl", price: 10),
];

_giftSendController.setGiftList(giftList);
```

If you need to modify the gift list of the gift panel:

```
mGiftCloudServer.queryGiftInfoList((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setGiftList(result);
    } else {
        ToastUtil.toastLongMessage("query gift list error, code = " + error);
    }
}));
```

Note:

Customers implement the logic of the `queryGiftInfoList` method themselves to obtain a custom gift list

`List< Gift >` , and set the gift list through `GiftListView.setGiftList` .

The `animationUrl` of the gift requires an SVGA animation.

If you need to modify the gift list of the gift panel:

```
// File location: iOS/TUILiveKit/Sources/Component/Gift/Store/TUIGiftStore.swift

class TUIGiftStore {
    static let shared = TUIGiftStore()

    private init() {
        giftCloudServer.queryGiftInfoList { [weak self] error, giftList in
            guard let self = self else { return }
            if error == .noError {
                self.giftList = giftList
            }
        }
    }

    var giftList: [TUIGift] = []
    let giftCloudServer: IGiftCloudServer = GiftCloudServer()
}

// File location: iOS/TUILiveKit/Sources/LiveStream/Main/Manager/LSRouterControlCenter.swift
private func getRouteDefaultView(route: LSRoute) -> UIView?
{
    switch route {
    case .giftView:
        let giftPanel = GiftListPanel(roomId: manager.roomState.roomId, dataSource: manager.giftList)
        giftPanel.setGiftList(TUIGiftStore.shared.giftList)
        view = giftPanel
    }
}
```

Note:

Customers implement the logic of the `queryGiftInfoList` method themselves to obtain a custom gift list

`List<TUIGift>`, and set the gift list through `GiftListView.setGiftList`.

The `animationUrl` of the gift requires an SVGA animation.

FAQs

Gift Component Initialization Timing

Since the gift component's `controller` needs to integrate some live room information parameters and user information, it should be loaded after the **audience enters the live broadcast room** or the **anchor creates a live streaming room**.

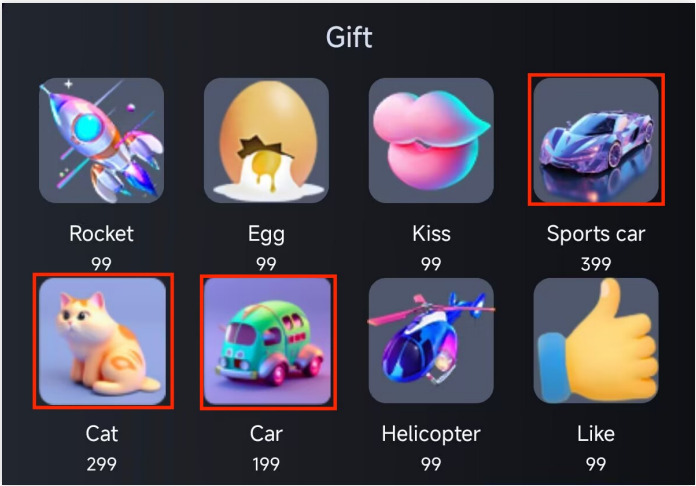
Gift Effects (TUILiveKit)

Last updated : 2025-01-22 15:11:44

TUILiveKit provides two types of gift effect players: the basic effect player and the advanced effect player. By default, the basic effect player is integrated. If you have higher performance requirements for the player or expect support for more animation file formats, we also provide an advanced effect player for your use.

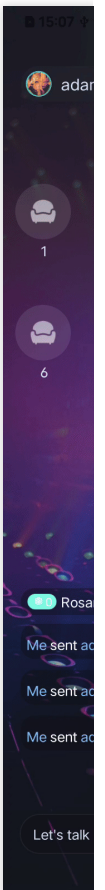
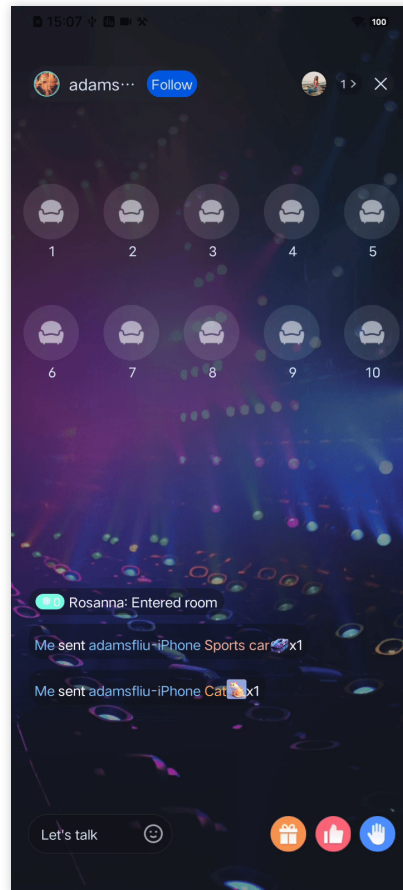
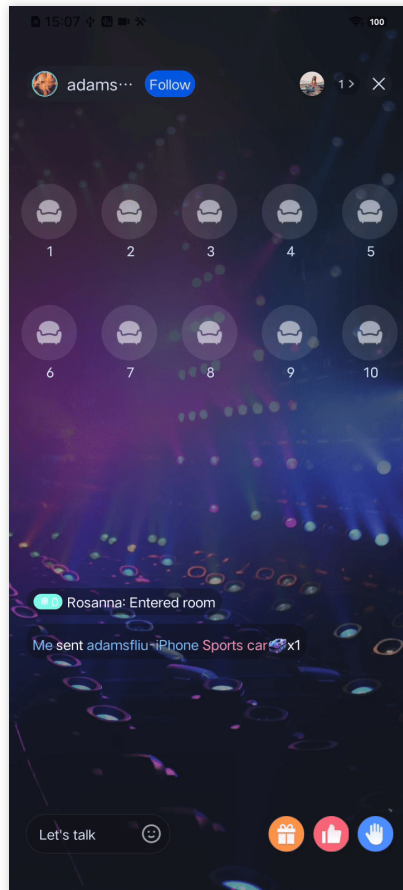
Basic effect player

The basic effect player is based on `SVGA` and supports only `SVGA` format files for special effect animations. When using the basic effect player, it comes with the following three default special effect animations:



Effect Showcase

Sports car	Cat	Car



Advanced Effect Player

The TUILiveKit advanced effect player adopts the **Tencent Effect Player** and supports various formats of special effect animations. The advanced effect player supports various formats of effect animations, such as vap, Lottie, mp4, svga, and more.

Note :

The advanced special effects player requires a separate fee. For more details, please send an email to: TRTC_helper@tencent.com.

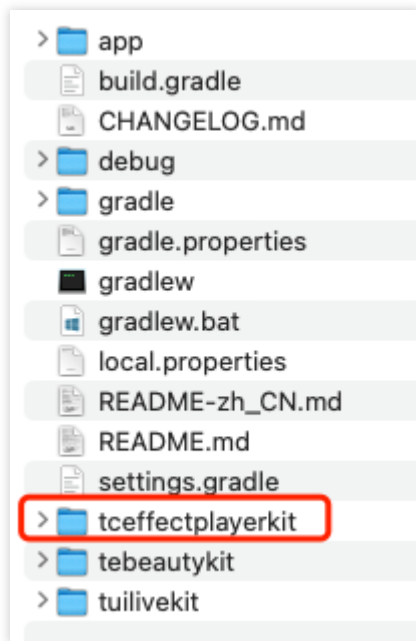
Integration Guide

Step 1: Integrating the gift effects component

Android

iOS

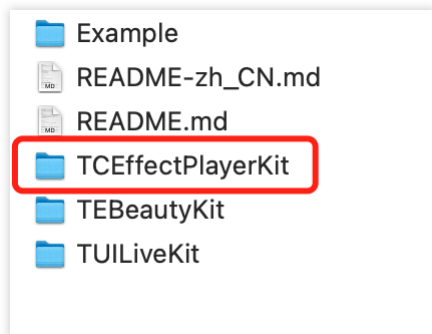
1. Download and extract [TUILiveKit](#). Copy the `Android/tceffectplayerkit` folder to your project, at the same level as the app folder.



2. Please edit your project's `settings.gradle` file and add the following code:

```
include ':tceffectplayerkit'
```

1. Download and extract [TUILiveKit](#). Copy the `ios/TCEffectPlayerKit` folder to your project, at the same level as the `Podfile` folder.



2. Please edit the `Podfile` and add the following code:

```
pod 'TCEffectPlayerKit', :podspec =>
  './TCEffectPlayerKit/TCEffectPlayerKit.podspec'
```

3. Save the changes and run `pod install` in the terminal to install the TCEffectPlayerKit dependency.

Step 2: Authorization

1. Apply for authorization and obtain `LicenseUrl` and `LicenseKey` , For more details, please send an email to: TRTC_helper@tencent.com.

2. In the initialization section of your business logic (typically in the same location as the [login](#) process), add the following authorization code and replace it with the `LicenseUrl` and `LicenseKey` you have obtained:

Android

iOS

```
TCMediaXBase.getInstance().setLicense(context,
    "LicenseUrl", // Replace with your LicenseUrl
    "LicenseKey", // Replace with your LicenseKey
    new ILicenseCallback() {
        @Override
        public void onResult(int error, String message) {
            Log.i("TCMediaXBase", "setLicense result: " + error + " " + message)
        }
    });
```

in iOS, you can set these in the `didFinishLaunchingWithOptions` method of the `AppDelegate` .

```
//
// AppDelegate.swift
//

import TCMediaX

func application(_ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?)
    Bool {
    TCMediaXBase.getInstance().setDelegate(self)
    TCMediaXBase.getInstance().setLicenseURL("LicenseURL", key: "LicenseKEY")
    return true
}

func onLicenseCheckCallback(_ errcode: Int32, withParam param: [AnyHashable : Any])
    Bool {
    debugPrint("[TCMediaXBase] setLicense result: errcode:\\(errcode), param:\\(param)")
    return true
}
```

Client APIs (TUICallKit)

Android

UIKit API

VoiceRoomKit

Last updated : 2024-12-05 17:51:26

API Introduction

VoiceRoomKit is a voice chat room component **with UI interface**, using the VoiceRoomKit API, you can quickly implement a voice chat room through simple interfaces. If you want to experience and debug the voice chat room, please read [Demo Rapid Deployment](#). If you want to integrate our feature directly into your project, please read [Quick Integration \(TUILiveKit\)](#).

API Overview

API	Description
createInstance	Obtain a VoiceRoomKit object instance.
createRoom	Create a voice chat room live streaming room.
enterRoom	Enter a voice chat room live streaming room using roomId.

API Details

createInstance

Obtain a VoiceRoomKit object instance.

```
static VoiceRoomKit createInstance(Context context)
```

Parameter:

Parameter	Type	Note	Default value	Meaning

context	Context	Mandatory	-	Android context object
---------	---------	-----------	---	------------------------

Return Value:VoiceRoomKit

createRoom

Create a voice chat room live streaming room.

```
void createRoom(String roomId, VoiceRoomDefine.CreateRoomParams params);
```

Parameter:

Parameter	Type	Note	Default value	Meaning
roomId	String	Mandatory	-	Live Streaming Room ID
params	CreateRoomParams	Mandatory	-	Create Live Room Parameters

Return Value:void

enterRoom

Enter a voice chat room live streaming room using roomId.

```
void enterRoom(String roomId);
```

Parameter:

Parameter	Type	Note	Default value	Meaning
roomId	String	Mandatory	-	Live Streaming Room ID

Return Value:void

VoiceRoomDefine Introduction

VoiceRoomKit is the UIKit layer data model class for voice chat rooms, mainly including the following data structures:

CreateRoomParams

Parameters object when creating a voice chat room live streaming, mainly including the following configuration parameters:

--	--	--	--	--

Parameter	Type	Note	Default value	Meaning
roomName	String		""	Voice chat room name
maxAnchorCount	int		10	Maximum number of users on stage
seatMode	String	Mandatory		<p>Mic On Mode is divided into the following two types:</p> <p>FREE_TO_TAKE(1): Free to Join the Podium mode, where the audience can join the seat freely without applying.</p> <p>APPLY_TO_TAKE(2): Apply to Join the Mic mode, where the audience needs the host's or administrator's approval to join the seat.</p>

SeatGridView

Last updated : 2024-12-25 18:11:07

API Introduction

SeatGridView is a basic control we developed for the Voice Chat Room UIKit. This core control provides rich APIs such as opening/closing the voice chat room, managing seat positions within the live room like applying to join the seat, inviting to take seat, moving seat positions, and kicking someone off the seat.

API Overview

API	Description
SeatGridView	Create a SeatGridView object, supporting code creation and XML loading .
startMicrophone	Enable local microphone
stopMicrophone	Disable local microphone
muteMicrophone	Pause publishing local audio stream
unmuteMicrophone	Resume publishing local audio stream
startVoiceRoom	Anchor creates live room and starts streaming
stopVoiceRoom	Anchor stops streaming and destroys live room
joinVoiceRoom	Audience joins an anchor's live room
leaveVoiceRoom	Audience leaves an anchor's live room
updateRoomSeatMode	Update Room Mic Mode
responseRemoteRequest	Anchor Responds to Microphone Application/Audience Responds to Microphone Invitation
cancelRequest	Anchor Cancels Microphone Invitation/Audience Cancels Microphone Application
takeSeat	Connect Mic

moveToSeat	Disconnect Mic
leaveSeat	Disconnect Mic
takeUserOnSeatByAdmin	Anchor Invites User to Connect Mic
kickUserOffSeatByAdmin	Anchor Kicks User off Mic
lockSeat	Anchor Locks Mic Position (including position lock, audio status lock, and video status lock)
setLayoutMode	Anchor Sets Layout Mode for Mic List
setSeatViewAdapter	Set the adapter for the seat view
addObserver	Set event callbacks
removeObserver	Remove event callbacks

API Details

SeatGridView

Create a SeatGridView object instance. Supports **code creation** and **XML loading**.

```
public SeatGridView(Context context)
```

Parameter:

Parameter	Type	Meaning
context	Context	Android context object

Return Value:SeatGridView

startMicrophone

Enable the local mic.

```
void startMicrophone(ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
callback	ActionCallback	Callback of the operation

Return Value:void

stopMicrophone

Disable the local mic.

```
void stopMicrophone()
```

Return Value:void

muteMicrophone

Pause publishing local audio stream.

```
void muteMicrophone()
```

Return Value:void

unmuteMicrophone

Resume publishing local audio stream.

```
void unmuteMicrophone(ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
callback	ActionCallback	Callback of the operation

Return Value:void

startVoiceRoom

Anchor creates live room and starts streaming.

```
void startVoiceRoom(RoomInfo roomInfo, GetRoomInfoCallback callback)
```

Parameter:

Parameter	Type	Meaning
roomInfo	RoomInfo	Information for creating a live streaming room
callback	ActionCallback	Callback of the operation

Return Value:void

stopVoiceRoom

Anchor stops streaming and destroys live room.

```
void stopVoiceRoom(ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
callback	ActionCallback	Callback of the operation

Return Value:void

joinVoiceRoom

Audience joins an anchor's live room.

```
void joinVoiceRoom(String roomId, GetRoomInfoCallback callback)
```

Parameter:

Parameter	Type	Meaning
roomId	String	Live Streaming Room ID
callback	ActionCallback	Callback of the operation

Return Value:void

leaveVoiceRoom

Audience leaves an anchor's live room.

```
void leaveVoiceRoom(ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
callback	ActionCallback	Callback of the operation

Return Value:void

updateRoomSeatMode

Update Room Mic Mode.

```
void updateRoomSeatMode(SeatMode seatMode, ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
seatMode	SeatMode	FREE_TO_TAKE: Free seat mode, audience can join the seat freely without application; APPLY_TO_TAKE: Apply to join the seat mode, audience needs the anchor's consent to join the seat.
callback	ActionCallback	Callback of the operation.

Return Value:void**responseRemoteRequest**

Anchor Responds to Microphone Application/Audience Responds to Microphone Invitation.

```
void responseRemoteRequest(String userId, boolean agree, ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
userId	String	Responding user's user ID, if the current role is an audience member, the ID can be left blank
agree	boolean	Whether to accept the request, true: accept the request, false: reject the request
callback	ActionCallback	Callback of the operation

Return Value:void**cancelRequest**

Anchor Cancels Microphone Invitation/Audience Cancels Microphone Application

```
void cancelRequest(String userId, ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
userId	String	Cancelled user's ID, if the current role is an audience member, the ID can be left blank

callback	ActionCallback	Callback of the operation
----------	----------------	---------------------------

Return Value:void

takeSeat

Request to Speak (In speaking mode, application is required)

```
void takeSeat(int index, int timeout, VoiceRoomDefine.RequestCallback callback)
```

Parameter:

Parameter	Type	Meaning
index	int	Mic position number for connecting
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not perform a timeout check or trigger a timeout callback
callback	ActionCallback	Callback of the operation

Return Value:void

moveToSeat

Move Mic (only users who are already on the seat can call this function)

```
void moveToSeat(int index, ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
index	int	Mic position number to move to
callback	ActionCallback	Callback of the operation

Return Value:void

leaveSeat

Proactively leave seat

```
void leaveSeat(ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
-----------	------	---------

callback	ActionCallback	Callback of the operation
----------	----------------	---------------------------

Return Value:void

takeUserOnSeatByAdmin

Anchor Invites User to Connect Mic

```
void takeUserOnSeatByAdmin(int index, String userId, int timeout, VoiceRoomDefine.R
```

Parameter:

Parameter	Type	Meaning
index	int	Invited Mic position number
userId	String	Invited user ID
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not perform a timeout check or trigger a timeout callback
callback	VoiceRoomDefine.RequestCallback	Callback of the operation

Return Value:void

kickUserOffSeatByAdmin

Anchor Kicks User off Mic

```
void kickUserOffSeatByAdmin(String userId, ActionCallback callback)
```

Parameter:

Parameter	Type	Meaning
userId	String	User ID kicked off the Mic
callback	ActionCallback	Callback of the operation

Return Value:void

lockSeat

Microphone Mute, Anchor locks Mic Position (including position lock, audio status lock, and video status lock)

```
void lockSeat(int seatIndex, TUIRoomDefine.SeatLockParams params, ActionCallback ca
```


Parameter:

Parameter	Type	Meaning
seatIndex	int	Mic position number to be locked
params	TUIRoomDefine.SeatLockParams	Microphone Mute Parameters. See details: TUIRoomDefine.SeatLockParams
callback	ActionCallback	Callback of the operation

Return Value:void**setLayoutMode**

Set the layout mode for the mic list.

```
void setLayoutMode(LayoutMode layoutModel, SeatViewLayoutConfig layoutConfig)
```

Parameter:

Parameter	Type	Meaning
layoutModel	LayoutMode	Mic list layout modes, support element layout, grid layout, vertical layout, custom layout.
layoutConfig	SeatViewLayoutConfig	Layout configuration information, effective only in custom layout mode

Return Value:void**setSeatViewAdapter**

Set the adapter for the mic view.

```
void setSeatViewAdapter(VoiceRoomDefine.SeatViewAdapter adapter)
```

Parameter:

Parameter	Type	Meaning
adapter	SeatViewAdapter	Seat view adapter

Return Value:void**addObserver**

Set event callbacks.

```
void addObserver(SeatGridViewObserver observer)
```

Parameter:

Parameter	Type	Meaning
observer	SeatGridViewObserver	Callback object of core component

Return Value:void

removeObserver

Remove event callbacks.

```
void removeObserver(SeatGridViewObserver observer)
```

Parameter:

Parameter	Type	Meaning
observer	SeatGridViewObserver	Callback object of core component

Return Value:void

Type Definition

Type	Description
LayoutMode	Layout modes of the seat position list support element layout, grid layout, vertical layout, and custom layout
SeatViewLayoutRowAlignment	Alignment of seat position layout
RequestType	Request type (apply to speak and invite to speak)
Size	Size of the seat position layout
SeatViewLayoutConfig	Microphone position layout configuration information
SeatViewLayoutRowConfig	Seating Layout Row Configuration Information
RequestCallback	Request Callback
SeatViewAdapter	Seat view adapter

LayoutMode

Layout modes of the seat position list

Type	Description
FOCUS	Element Layout
GRID	Grid Layout
VERTICAL	Vertical layout
FREE	Customized Layout

SeatViewLayoutRowAlignment

Alignment of seat position layout

Type	Description
START	Seat position near the start
END	Seat position near the end
CENTER	Seat position near the middle
SPACE_BETWEEN	No space before the first and after the last seat positions, evenly distribute the remaining space between other seat positions
SPACE_AROUND	Distribute half of the space before the first and after the last seat positions, evenly distribute the remaining space between other seat positions
SPACE_EVENLY	Evenly distribute the remaining space between all seat positions

RequestType

Request type

Type	Description
APPLY_TO_TAKE_SEAT	Apply to speak
INVITE_TO_TAKE_SEAT	Invite to speak

Size

Size of the seat position layout

--	--

Type	Description
width	Layout width
height	Layout height

SeatViewLayoutConfig

Microphone position layout configuration information

Type	Description
rowConfigs	List of all row configuration information in the seat layout, refer to SeatViewLayoutRowConfig for content.
rowSpacing	Seat row spacing

SeatViewLayoutRowConfig

Seating Layout Row Configuration Information

Type	Description
count	Number of seats displayed in this row
seatSpacing	Horizontal spacing of each seat in this row (effective only when the alignment is START, END, or CENTER)
seatSize	Size of the seat layout in this row
alignment	Alignment of the layout in this row (SeatViewLayoutRowAlignment)

RequestCallback

Request to speak/Invite to speak callback

API	Description
onAccepted	Request accepted
onRejected	Request rejected
onCancelled	Request canceled
onTimeout	Request timeout
onError	Request Exception

SeatViewAdapter

Seat View Adapter Interface, you can customize the display UI of each seat by implementing this interface.

API	Description
createSeatView	Callback when creating a single seat layout.
updateSeatView	Callback when updating the seat view.
updateUserVolume	Callback when updating user volume.

Event Callback Details

onAccepted

Application to speak/invite to speak request accepted.

```
void onAccepted(TUIRoomDefine.UserInfo userInfo);
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	Response to the current request's user information

Returned value: void

onRejected

Application to speak/invite to speak request rejected.

```
void onRejected(TUIRoomDefine.UserInfo userInfo);
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	Response to the current request's user information

Returned value: void

onCancelled

Application to speak/invite to speak request canceled.

```
void onCancelled(TUIRoomDefine.UserInfo userInfo);
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	User information for the canceled request

Returned value: void**onTimeout**

Application to speak/invite to speak request timed out.

```
void onTimeout(TUIRoomDefine.UserInfo userInfo);
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	User information for the initiated request

Returned value: void**onError**

Application to speak/invite to speak request error.

```
void onError(TUIRoomDefine.UserInfo userInfo, TUICommonDefine.Error error, String message);
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	User information for the initiated request
error	TUICommonDefine.Error	Error code
message	String	Error message

Returned value: void**createSeatView**

Callback when creating a single seat layout, you need to return your custom view, and the core view will help you create the view.

```
View createSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo);
```

Parameter:

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
seatInfo	SeatInfo	Seat information

Returned value: View**updateSeatView**

Callback when updating the seat view, you can update your seat view based on the seatInfo information returned by the callback.

```
void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo, View seatView);
```

Parameter:

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
seatInfo	SeatInfo	Seat information
seatView	View	Current updated seat view

Return Value: void**updateUserVolume**

Callback when updating user volume, you can update your seat view based on the returned volume.

```
void updateUserVolume(SeatGridView seatGridView, int volume, View seatView);
```

Parameter:

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
volume	int	Volume level
seatView	View	Current seat layout view with volume changes

Return Value: void

SeatGridViewObserver Overview

Function List	Description
onRoomDismissed	Event of Room Termination Received
onKickedOutOfRoom	Event of Being Kicked Out of the Room Received
onSeatRequestReceived	Event of Request for Speaking/Invitation to Speak Received
onSeatRequestCancelled	Event of Request for Speaking/Invitation to Speak Canceled
onKickedOffSeat	Event of User Kicked Off Seat Received
onUserAudioStateChanged	Event of User Audio Status Changed
onSeatViewClicked	Seat View Click Event

SeatGridViewObserver Details

onRoomDismissed

Event of Live Room Destroyed

```
void onRoomDismissed(String roomId);
```

Parameter:

Parameter	Type	Description
roomId	String	Room ID

Return Value:void

onKickedOutOfRoom

Event of Being Kicked Out of the Room

```
void onKickedOutOfRoom(String roomId, TUIRoomDefine.KickedOutOfRoomReason reason, S
```

Parameter:

Parameter	Type	Description
roomId	String	Room ID

reason	KickedOutOfRoomReason	Reason for Being Kicked Out
message	String	Description of Being Kicked Out

Return Value:void

onSeatRequestReceived

Event of Request for Speaking/Invitation to Speak Received

```
void onSeatRequestReceived(VoiceRoomDefine.RequestType type, TUIRoomDefine.UserInfo
```

Parameter:

Parameter	Type	Description
type	RequestType	Request type (Request for Speaking, Invitation to Speak)
userInfo	UserInfo	Information of the user who sent the request

Return Value:void

onSeatRequestCancelled

Event of Request for Speaking/Invitation to Speak Canceled

```
void onSeatRequestCancelled(VoiceRoomDefine.RequestType type, TUIRoomDefine.UserInfo
```

Parameter:

Parameter	Type	Description
type	RequestType	Request type (Request for Speaking, Invitation to Speak)
userInfo	UserInfo	Information of the user who canceled the request

Return Value:void

onKickedOffSeat

Event of User Kicked Off Seat

```
void onKickedOffSeat(UserInfo userInfo);
```

Parameter:

Parameter	Type	Description

userInfo	UserInfo	Information of the host who kicked the user off the seat
----------	----------	--

Return Value:void

onUserAudioStateChanged

Event of User Audio Status Changed

```
void onUserAudioStateChanged(UserInfo userInfo, boolean hasAudio, TUIRoomDefine.Cha
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	User Information
hasAudio	boolean	Is there an audio stream
reason	ChangeReason	Reason for audio stream change

Return Value:void

onSeatViewClicked

Seat View Click Event

```
void onSeatViewClicked(View seatView, TUIRoomDefine.SeatInfo seatInfo);
```

Parameter:

Parameter	Type	Description
seatView	View	Currently clicked seat view object
seatInfo	SeatInfo	Seat information

Return Value:void

Engine API

API Overview

Last updated : 2025-02-13 16:38:50

API OVERVIEW

Creating instances and event callback.

FuncList	DESC
destroySharedInstance	Destroy the TUIRoomEngine instance (singleton mode)
login	After creating a TUIRoomEngine instance, you should login with sdkAppId, userId and userSig then you can call TUIRoomEngine instance and other function.
logout	Log out of your account. If you are in the room, there will be active leaving room and destroying resource operations.
setSelfInfo	Update user name and avatar for logged-in user.
getSelfInfo	Return the basic information of the logged-in user, including nickname and avatar.
addObserver	Set event observer.
removeObserver	Remove event observer.

Room APIs.

FuncList	DESC
createRoom	Create a room.
destroyRoom	Dismiss the room.
enterRoom	Enter a room.
exitRoom	Exit the room.

fetchRoomInfo	Fetch room information.
updateRoomNameByAdmin	Update room name (only support for administrators or room owner).
updateRoomSeatModeByAdmin	Update room seat mode (only support for administrators or room owner).
updateRoomPasswordByAdmin	Update room password.
getRoomMetadata	Get room metadata.
setRoomMetadataByAdmin	Set room metadata, if the key already exists, update its value, if not, add the key.

Local user view rendering, video management.

FuncList	DESC
setLocalVideoView	Set the local camera to preview the render view.
openLocalCamera	Open the local camera.
closeLocalCamera	Close the local camera.
startPushLocalVideo	Start publishing local video stream, default enabled.
stopPushLocalVideo	Stop publishing local video stream.
updateVideoQuality	Update video encoding quality.
updateVideoQualityEx	Set the video encoding parameters.
setVideoResolutionMode	Set the video resolution mode (horizontal resolution or vertical resolution).
enableGravitySensor	Turn on gravity sensing mode. (only availble on mobile OS and the camera capture scene inside the SDK).
startScreenSharing	Start screen sharing (only available on mobile OS).
stopScreenSharing	Stop screen sharing.

Local User Audio Management.

FuncList	DESC

openLocalMicrophone	Open local microphone.
closeLocalMicrophone	Close the local microphone.
updateAudioQuality	Update audio encoding quality.
muteLocalAudio	Pause publishing the local audio stream.
unmuteLocalAudio	Resume publishing the local audio stream.
enableSystemAudioSharing	Enable system audio sharing

Remote user view rendering and video management.

FuncList	DESC
setRemoteVideoView	Set the render view for remote user.
startPlayRemoteVideo	Start playing the remote user's video stream.
stopPlayRemoteVideo	Stop playing the remote user's video stream.
muteRemoteAudioStream	Mute the remote user's audio stream.

User information in the room.

FuncList	DESC
getUserList	Get the list of user in the room.
getUserInfo	Get user information.

User management in the room.

FuncList	DESC
changeUserRole	Change user role (only support for administrators or room owner).
changeUserNameCard	Change user nickname in the room (only support to change all user for administrators or room owner, user can only change by self).

kickRemoteUserOutOfRoom	Kick the remote user out of the room (only support for administrators or room owner).
addCategoryTagForUsers	Add a tag for the user (only support for administrators or room owner).
removeCategoryTagForUsers	Remove tag for user (only support for room owner).
getUserListByTag	Get user information in the room based on the tag.
setCustomInfoForUser	Set custom information for room users.

User speech management in the room.

FuncList	DESC
disableDeviceForAllUserByAdmin	The owner or administrator control that all users whether can open device. For example: all users are prohibited from opening the microphone(only available in the conference scenario).
openRemoteDeviceByAdmin	Request the remote user to open the media device (only support for administrators or room owner).
closeRemoteDeviceByAdmin	Close remote user media devices (only support for administrators or room owner).
applyToAdminToOpenLocalDevice	Apply to open the local media device (available to general users).

Seat management in the room.

FuncList	DESC
getSeatList	Get seat list.
lockSeatByAdmin	Lock the seat (only support for administrators or room owner).
takeSeat	Take the seat.
leaveSeat	Leave the seat.
moveToSeat	Move to seat.
takeUserOnSeatByAdmin	Invite user to take the seat (only support for administrators or room owner).

kickUserOffSeatByAdmin	Kick off the user from seat (only support for administrators or room owner).
getSeatApplicationList	Get the request list of users who want to take the seat in the room (only support for administrators or room owner).

Message.

FuncList	DESC
disableSendingMessageByAdmin	Disable the ability of remote users to send messages (only support for administrators or room owner).
disableSendingMessageForAllUser	Disable the ability of all users to send messages (only support for administrators or room owner).

Request Management.

FuncList	DESC
cancelRequest	Cancel request.
responseRemoteRequest	Response request.

Advanced Features.

FuncList	DESC
getTRTCCloud	Get the TRTC instance object.
setBeautyLevel	Set the beauty level.
setWhitenessLevel	Set whitening level.
getExtension	Get the extension.
getMediaDeviceManager	Get device management class.
getLiveConnectionManager	Get live-connection management class.
getLiveBattleManager	Get live-battle management class.

Debug related.

FuncList	DESC
callExperimentalAPI	Call experimental APIs.

Error event callback.

FuncList	DESC
onError	Error event callback.

Login status event callback.

FuncList	DESC
onKickedOffLine	The current user was kicked offline.
onUserSigExpired	The current user signature is expired.

Event callback in the room.

FuncList	DESC
onRoomNameChanged	The name of the room has changed.
onAllUserMicrophoneDisableChanged	The status of disabling to open microphone has changed for all users.
onAllUserCameraDisableChanged	The status of disabling to open camera has changed for all users.
onScreenShareForAllUserDisableChanged	The status of disabling to open screen sharing has changed for all users.
onSendMessageForAllUserDisableChanged	The status of disabling to send message has changed for all users.
onRoomDismissed	Room was dismissed.

onKickedOutOfRoom	The current user has been kicked off from the room.
onRoomSeatModeChanged	The room seat mode has changed.
onRoomUserCountChanged	The count of user in the room has changed.
onRoomMetadataChanged	The key-value of room metadata has changed.

User event callback in the room.

FuncList	DESC
onRemoteUserEnterRoom	Remote user entered room.
onRemoteUserLeaveRoom	Remote user left room.
onUserInfoChanged	User information has changed in the room.
onUserVideoStateChanged	The status of the user has video stream changed.
onUserAudioStateChanged	The status of the user has audio stream changed.
onUserVoiceVolumeChanged	User volume changed.
onSendMessageForUserDisableChanged	The status of disabling to send message has changed for user.
onUserNetworkQualityChanged	The User network status changed.
onUserScreenCaptureStopped	Screen sharing stopped.

Seat event callback in the room.

FuncList	DESC
onSeatListChanged	Seat list changed.
onKickedOffSeat	The user was kicked off the seat.

Request event callback.

--	--

FuncList	DESC
onRequestReceived	Receive a request message.
onRequestCancelled	Received a cancelled request.
onRequestProcessed	Receive a request to be processed by other administrator/owner.

Device management APIs.

FuncList	DESC
isFrontCamera	Query whether the front camera is being used (only available for mobile OS).
switchCamera	Switch to the front/rear camera (only available for mobile OS).
isAutoFocusEnabled	Query whether automatic face detection is supported (only available for mobile OS).
enableCameraAutoFocus	Enable auto focus (only available for mobile OS).
enableCameraTorch	Enable/Disable flash, i.e., the torch mode (only available for mobile OS).
setAudioRoute	Set the audio route (only available for mobile OS).

TUIRoomEngine

Last updated : 2025-02-13 16:38:51

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUIRoomEngine @ TUIKitEngine.

Function: TUIRoomEngine Main function APIs.

Version: 2.9.0

TUIRoomEngine

TUIRoomEngine

FuncList	DESC
destroySharedInstance	Destroy the TUIRoomEngine instance (singleton mode)
login	After creating a TUIRoomEngine instance, you should login with sdkAppId, userId and userSig then you can call TUIRoomEngine instance and other function.
logout	Log out of your account. If you are in the room, there will be active leaving room and destroying resource operations.
setSelfInfo	Update user name and avatar for logged-in user.
getSelfInfo	Return the basic information of the logged-in user, including nickname and avatar.
setSelfInfo	Update user basic information for logged-in user.
addObserver	Set event observer.
removeObserver	Remove event observer.
createRoom	Create a room.
destroyRoom	Dismiss the room.
enterRoom	Enter a room.
enterRoom	Enter a room.

<code>enterRoom</code>	Enter a room.
<code>exitRoom</code>	Exit the room.
<code>fetchRoomInfo</code>	Fetch room information.
<code>fetchRoomInfo</code>	Fetch room information.
<code>updateRoomNameByAdmin</code>	Update room name (only support for administrators or room owner).
<code>updateRoomSeatModeByAdmin</code>	Update room seat mode (only support for administrators or room owner).
<code>updateRoomPasswordByAdmin</code>	Update room password.
<code>getRoomMetadata</code>	Get room metadata.
<code>setRoomMetadataByAdmin</code>	Set room metadata, if the key already exists, update its value, if not, add the key.
<code>setLocalVideoView</code>	Set the local camera to preview the render view.
<code>openLocalCamera</code>	Open the local camera.
<code>closeLocalCamera</code>	Close the local camera.
<code>startPushLocalVideo</code>	Start publishing local video stream, default enabled.
<code>stopPushLocalVideo</code>	Stop publishing local video stream.
<code>updateVideoQuality</code>	Update video encoding quality.
<code>updateVideoQualityEx</code>	Set the video encoding parameters.
<code>setVideoResolutionMode</code>	Set the video resolution mode (horizontal resolution or vertical resolution).
<code>enableGravitySensor</code>	Turn on gravity sensing mode. (only available on mobile OS and the camera capture scene inside the SDK).
<code>startScreenSharing</code>	Start screen sharing (only available on mobile OS).
<code>stopScreenSharing</code>	Stop screen sharing.
<code>openLocalMicrophone</code>	Open local microphone.
<code>closeLocalMicrophone</code>	Close the local microphone.

updateAudioQuality	Update audio encoding quality.
muteLocalAudio	Pause publishing the local audio stream.
unmuteLocalAudio	Resume publishing the local audio stream.
enableSystemAudioSharing	Enable system audio sharing
setRemoteVideoView	Set the render view for remote user.
startPlayRemoteVideo	Start playing the remote user's video stream.
stopPlayRemoteVideo	Stop playing the remote user's video stream.
muteRemoteAudioStream	Mute the remote user's audio stream.
getUserList	Get the list of user in the room.
getUserInfo	Get user information.
changeUserRole	Change user role (only support for administrators or room owner).
changeUserNameCard	Change user nickname in the room (only support to change all user for administrators or room owner, user can only change by self).
kickRemoteUserOutOfRoom	Kick the remote user out of the room (only support for administrators or room owner).
addCategoryTagForUsers	Add a tag for the user (only support for administrators or room owner).
removeCategoryTagForUsers	Remove tag for user (only support for room owner).
getUserListByTag	Get user information in the room based on the tag.
setCustomInfoForUser	Set custom information for room users.
disableDeviceForAllUserByAdmin	The owner or administrator control that all users whether can open device. For example: all users are prohibited from opening the microphone(only available in the conference scenario).
openRemoteDeviceByAdmin	Request the remote user to open the media device (only support for administrators or room owner).
closeRemoteDeviceByAdmin	Close remote user media devices (only support for administrators or room owner).
applyToAdminToOpenLocalDevice	Apply to open the local media device (available to general users).

getSeatList	Get seat list.
lockSeatByAdmin	Lock the seat (only support for administrators or room owner).
takeSeat	Take the seat.
leaveSeat	Leave the seat.
moveToSeat	Move to seat.
takeUserOnSeatByAdmin	Invite user to take the seat (only support for administrators or room owner).
kickUserOffSeatByAdmin	Kick off the user from seat (only support for administrators or room owner).
getSeatApplicationList	Get the request list of users who want to take the seat in the room (only support for administrators or room owner).
disableSendingMessageByAdmin	Disable the ability of remote users to send messages (only support for administrators or room owner).
disableSendingMessageForAllUser	Disable the ability of all users to send messages (only support for administrators or room owner).
cancelRequest	Cancel request.
responseRemoteRequest	Response request.
getTRTCCloud	Get the TRTC instance object.
setBeautyLevel	Set the beauty level.
setWhitenessLevel	Set whitening level.
getExtension	Get the extension.
getMediaDeviceManager	Get device management class.
getLiveConnectionManager	Get live-connection management class.
getLiveBattleManager	Get live-battle management class.
callExperimentalAPI	Call experimental APIs.

destroySharedInstance

destroySharedInstance

Destroy the TUIRoomEngine instance (singleton mode)

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types. Use destroySharedInstance to release the singleton object.

login

login

void login	(Context context
	int sdkAppId
	String userId
	String userSig
	TUIRoomDefine.ActionCallback callback)

After creating a TUIRoomEngine instance, you should login with sdkAppId, userId and userSig then you can call TUIRoomEngine instance and other function.

Param	DESC
sdkAppId	It is Application ID. You can see the SDKAppId by creating an application in the TRTC Console .
userId	User ID, it is the unique identifier used by Tencent Cloud to distinguish users.
userSig	The user signature designed by Tencent Cloud based on the UserId, which is used to access Tencent Cloud services. More details, see UserSig

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

If a user is kicked off while online, the SDK will notify you through the \$onKickedOffLine\$ callback in \$TUIRoomObserver\$.

logout

logout

void logout	(TUIRoomDefine.ActionCallback callback)
-------------	---

Log out of your account. If you are in the room, there will be active leaving room and destroying resource operations.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

setSelfInfo

setSelfInfo

void setSelfInfo	(String userName
	String avatarURL
	TUIRoomDefine.ActionCallback callback)

Update user name and avatar for logged-in user.

Param	DESC
avatarURL	User avatar URL.
userName	User name.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getSelfInfo

getSelfInfo

Return the basic information of the logged-in user, including nickname and avatar.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Return Desc:

[LoginUserInfo](#) User login information.

setSelfInfo

setSelfInfo

void setSelfInfo	(TUIRoomDefine. LoginUserInfo userInfo
	TUIRoomDefine.ActionCallback callback)

Update user basic information for logged-in user.

Param	DESC
userInfo	Local user information.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

addObserver

addObserver

void addObserver	(TUIRoomObserver observer)
------------------	---

Set event observer.

Param	DESC
observer	Listening instance.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

You can use TUIRoomObserver to receive roomEngine events. More details, see [TUIRoomObserver](#).

removeObserver

removeObserver

void removeObserver	(TUIRoomObserver observer)
---------------------	---

Remove event observer.

Param	DESC
observer	The event observer to be removed.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

createRoom

createRoom

void createRoom	(TUIRoomDefine. RoomInfo roomInfo
	TUIRoomDefine.ActionCallback callback)

Create a room.

Param	DESC
roomInfo	Room information. More details, see TUIRoomInfo .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

You can create a conference room or live room through this API.

destroyRoom

destroyRoom

void destroyRoom	(TUIRoomDefine.ActionCallback callback)
------------------	---

Dismiss the room.**Note**

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the room was dismissed, the SDK notifies the users in the room through the [onRoomDismissed](#) callback in \$TUIRoomObserver\$.

enterRoom

enterRoom

void enterRoom	(String roomId
	TUIRoomDefine.GetRoomInfoCallback callback)

Enter a room.

Param	DESC
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entered the room, the SDK will notify the user in the room through the \$onRemoteUserEnterRoom\$ callback in \$TUIRoomObserver\$.

enterRoom

enterRoom

void enterRoom	(String roomId
	TUIRoomDefine. RoomType roomType
	TUIRoomDefine.GetRoomInfoCallback callback)

Enter a room.

Param	DESC
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entered the room, the SDK will notify the user in the room through the `$onRemoteUserEnterRoom$` callback in `$TUIRoomObserver$`.

enterRoom

enterRoom

void enterRoom	(String roomId
	TUIRoomDefine. RoomType roomType
	TUIRoomDefine. EnterRoomOptions options
	TUIRoomDefine.GetRoomInfoCallback callback)

Enter a room.

Param	DESC
options	Room options. More details, see TUIEnterRoomOptions .
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entering the room, the SDK will notify the user in the room through the `$onRemoteUserEnterRoom$` callback in `$TUIRoomObserver$`.

exitRoom

exitRoom

void exitRoom	(boolean syncWaiting
	TUIRoomDefine.ActionCallback callback)

Exit the room.

Param	DESC
syncWaiting	true: wait for exit request finished, false: exit immediately.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After left the room, the SDK will notify the user in the room through the [onRemoteUserLeaveRoom](#) callback in \$TUIRoomObserver\$.

fetchRoomInfo

fetchRoomInfo

void fetchRoomInfo	(TUIRoomDefine.GetRoomInfoCallback callback)
--------------------	--

Fetch room information.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

You can get room information through this API.

fetchRoomInfo

fetchRoomInfo

void fetchRoomInfo	(String roomId
	TUIRoomDefine. RoomType roomType
	TUIRoomDefine.GetRoomInfoCallback callback)

Fetch room information.

Param	DESC
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Return Desc:

Successfully fetched the room information callback, which will include room information of \$TUIRoomInfo.

updateRoomNameByAdmin

updateRoomNameByAdmin

void updateRoomNameByAdmin	(String roomName
	TUIRoomDefine.ActionCallback callback)

Update room name (only support for administrators or room owner).

Param	DESC
roomName	Room name.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the room name was updated, the SDK notifies users in the room through the [onRoomNameChanged](#) callback in \$TUIRoomObserver\$.

updateRoomSeatModeByAdmin

updateRoomSeatModeByAdmin

void updateRoomSeatModeByAdmin	(TUIRoomDefine. SeatMode seatMode
	TUIRoomDefine.ActionCallback callback)

Update room seat mode (only support for administrators or room owner).

Param	DESC
seatMode	TUISeatModeFreeToTake: Free to take seat mode, users can take the seat without application; TUISeatModeApplyToTake: Apply to take seat mode, users can only take the seat after the owner or administrator approved.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the room seat mode was updated, the SDK will notify users in the room through the [onRoomSeatModeChanged](#) callback in `$TUIRoomObserver$`.

updateRoomPasswordByAdmin

updateRoomPasswordByAdmin

<code>void updateRoomPasswordByAdmin</code>	(String password
	<code>TUIRoomDefine.ActionCallback</code> callback)

Update room password.

Param	DESC
password	Room password.

Note

The function only supports the [CONFERENCE](#) room type.

getRoomMetadata

getRoomMetadata

<code>void getRoomMetadata</code>	(List<String> keys
	<code>TUIRoomDefine.GetRoomMetadataCallback</code> callback)

Get room metadata.

--	--

Param	DESC
keys	Room metadata key list, if keys are passed as empty, all metadata will be retrieved.

Note

The function only supports the [LIVE](#) room type.

setRoomMetadataByAdmin

setRoomMetadataByAdmin

void setRoomMetadataByAdmin	(HashMap<String, String> metadata
	TUIRoomDefine.ActionCallback callback)

Set room metadata, if the key already exists, update its value, if not, add the key.

Param	DESC
metadata	Key-Value in room metadata.

Note

The function only supports the [LIVE](#) room type.

setLocalVideoView

setLocalVideoView

void setLocalVideoView	(TUIVideoView view)
------------------------	---------------------

Set the local camera to preview the render view.

Param	DESC
view	Render view.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

openLocalCamera

openLocalCamera

void openLocalCamera	(boolean isFront
	TUIRoomDefine. VideoQuality quality
	TUIRoomDefine.ActionCallback callback)

Open the local camera.

Param	DESC
isFront	true: front false: rear (only available on mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After opened the local camera in the room, the local video stream is published by default, and the SDK notifies the users in the room through the `$onUserVideoStateChanged$` callback in `$TUIRoomObserver$`.

closeLocalCamera

closeLocalCamera

Close the local camera.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After closed the local camera in the room, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

startPushLocalVideo

startPushLocalVideo

Start publishing local video stream, default enabled.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After published the local video, if your local camera is opening, the SDK will notify users in the room through the [onUserVideoStateChanged](#) callback in \$TUIRoomObserver\$.

stopPushLocalVideo

stopPushLocalVideo

Stop publishing local video stream.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After stopped published local video, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in \$TUIRoomObserver\$.

updateVideoQuality

updateVideoQuality

void updateVideoQuality	(TUIRoomDefine. VideoQuality quality)
-------------------------	---

Update video encoding quality.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

updateVideoQualityEx

updateVideoQualityEx

void updateVideoQualityEx	(TUIRoomDefine. VideoStreamType streamType
	TUIRoomDefine. RoomVideoEncoderParams params)

Set the video encoding parameters.

Param	DESC
params	Encoding parameters of the video. More details, see RoomVideoEncoderParams .
streamType	The type of video stream. More details, see VideoStreamType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

setVideoResolutionMode

setVideoResolutionMode

void setVideoResolutionMode	(TUIRoomDefine. VideoStreamType streamType
	TUIRoomDefine. ResolutionMode resolutionMode)

Set the video resolution mode (horizontal resolution or vertical resolution).

Param	DESC
resolutionMode	Resolution mode. More details, see ResolutionMode .
streamType	The type of video stream. More details, see VideoStreamType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

enableGravitySensor

enableGravitySensor

void enableGravitySensor	(boolean enable)
--------------------------	------------------

Turn on gravity sensing mode. (only availble on mobile OS and the camera capture scene inside the SDK).

Param	DESC
enable	true: Open false: Close.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After turning on gravity sensing, if the device on the collection end rotates, the images on the collection end and the audience will be rendered accordingly to ensure that the image in the field of view is always facing up.

startScreenSharing

startScreenSharing

Start screen sharing (only available on mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After screen sharing started, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

stopScreenSharing

stopScreenSharing

Stop screen sharing.

After screen sharing ended, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$` and also notifies you through the [onUserScreenCaptureStopped](#) callback.

openLocalMicrophone

openLocalMicrophone

<code>void openLocalMicrophone</code>	(<code>TUIRoomDefine.AudioQuality</code> quality
	<code>TUIRoomDefine.ActionCallback</code> callback)

Open local microphone.

Param	DESC
quality	Audio quality.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After opened the local microphone in a room, the SDK notifies users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

closeLocalMicrophone

closeLocalMicrophone

Close the local microphone.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After closed the local microphone in the room, the SDK notifies the users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

updateAudioQuality

updateAudioQuality

void updateAudioQuality	(TUIRoomDefine. AudioQuality quality)
-------------------------	---

Update audio encoding quality.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

muteLocalAudio

muteLocalAudio

Pause publishing the local audio stream.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

If you have opened your microphone and call the API to pause publishing the local audio stream, the SDK will notify the users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

unmuteLocalAudio

unmuteLocalAudio

void unmuteLocalAudio	(TUIRoomDefine.ActionCallback callback)
-----------------------	---

Resume publishing the local audio stream.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

If you have opened your microphone and call the API to resume publishing the local audio stream, the SDK will notify the users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

enableSystemAudioSharing

enableSystemAudioSharing

<code>void enableSystemAudioSharing</code>	(boolean enable)
--	------------------

Enable system audio sharing

This API captures system audio data from your device and mixes it into the current audio stream of the SDK. This ensures that other users in the room hear the audio played back by the another app.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Android: You need to use this interface to enable system sound capture first, and it will take effect only when you call `startScreenCapture` to enable screen sharing.

setRemoteVideoView

setRemoteVideoView

<code>void setRemoteVideoView</code>	(String userId
	<code>TUIRoomDefine.VideoStreamType</code> streamType
	<code>TUIVideoView</code> view)

Set the render view for remote user.

Param	DESC
streamType	The type of video stream. More details, see VideoStreamType .
userId	Remote user ID.
view	Render view.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

startPlayRemoteVideo

startPlayRemoteVideo

void startPlayRemoteVideo	(String userId
	TUIRoomDefine. VideoStreamType streamType
	TUIRoomDefine.PlayCallback callback)

Start playing the remote user's video stream.

Param	DESC
streamType	The type of video stream. More details, see VideoStreamType .
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

stopPlayRemoteVideo

stopPlayRemoteVideo

void stopPlayRemoteVideo	(String userId
	TUIRoomDefine. VideoStreamType streamType)

Stop playing the remote user's video stream.

Param	DESC
streamType	The type of video stream. More details, see VideoStreamType .
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

muteRemoteAudioStream

muteRemoteAudioStream

void muteRemoteAudioStream	(String userId
	boolean isMute)

Mute the remote user's audio stream.

Param	DESC
isMute	true: pause pulling remote user's audio stream, false: resume pulling remote user's audio stream.
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getUserList

getUserList

void getUserList	(long nextSequence
	TUIRoomDefine.GetUserListCallback callback)

Get the list of user in the room.

Param	DESC
nextSequence	Filling in 0 for the first request, if the returned data of the callback is not zero, paging is required, continue the operation until it is 0.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getUserInfo

getUserInfo

void getUserInfo	(String userId
	TUIRoomDefine.GetUserInfoCallback callback)

Get user information.

Param	DESC
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

changeUserRole

changeUserRole

void changeUserRole	(String userId
	TUIRoomDefine. Role role
	TUIRoomDefine.ActionCallback callback)

Change user role (only support for administrators or room owner).

Param	DESC
role	User role. More details, see Role .
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the user role changed, the SDK will notify the users in the room through the [onUserInfoChanged](#) callback in \$TUIRoomObserver\$.

changeUserNameCard

changeUserNameCard

--	--

void changeUserNameCard	(String userId
	String nameCard
	TUIRoomDefine.ActionCallback callback)

Change user nickname in the room (only support to change all user for administrators or room owner, user can only change by self).

Param	DESC
nameCard	User nickname to set, maximum support is 32 bytes
userId	User ID to change.

Note

The function only supports the [CONFERENCE](#) room type.

After the user nickname changed, the SDK will notify the users in the room through the [onUserInfoChanged](#) callback in `$TUIRoomObserver$`.

kickRemoteUserOutOfRoom

kickRemoteUserOutOfRoom

void kickRemoteUserOutOfRoom	(String userId
	TUIRoomDefine.ActionCallback callback)

Kick the remote user out of the room (only support for administrators or room owner).

Param	DESC
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the remote user is kicked off from the room, the SDK notifies the kicked user through the [onKickedOutOfRoom](#) callback in `$TUIRoomObserver$` and notifies users in the room through [onRemoteUserLeaveRoom](#).

addCategoryTagForUsers

addCategoryTagForUsers

void addCategoryTagForUsers	(int tag
	List<String> userList
	TUIRoomDefine.ActionCallback callback)

Add a tag for the user (only support for administrators or room owner).

Param	DESC
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.
userList	User list.

Note

The function only supports the [CONFERENCE](#) room type.

removeCategoryTagForUsers

removeCategoryTagForUsers

void removeCategoryTagForUsers	(int tag
	List<String> userList
	TUIRoomDefine.ActionCallback callback)

Remove tag for user (only support for room owner).

Param	DESC
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.
userList	User list.

Note

The function only supports the [CONFERENCE](#) room type.

getUserListByTag

getUserListByTag

void getUserListByTag	(int tag
	long nextSequence
	TUIRoomDefine.GetUserListCallback callback)

Get user information in the room based on the tag.

Param	DESC
nextSequence	Filling in 0 for the first request, if the returned data of the callback is not zero, paging is required, continue the operation until it is 0.
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.

Note

The function only supports the [CONFERENCE](#) room type.

setCustomInfoForUser

setCustomInfoForUser

void setCustomInfoForUser	(String userId
	HashMap<String
	byte[]> customInfo
	TUIRoomDefine.ActionCallback callback)

Set custom information for room users.

Param	DESC
customInfo	Custom information.
userId	User userId.

Note

The function only supports the [CONFERENCE](#) room type.

disableDeviceForAllUserByAdmin

disableDeviceForAllUserByAdmin

void disableDeviceForAllUserByAdmin	(TUIRoomDefine. MediaDevice device
	boolean isDisable
	TUIRoomDefine.ActionCallback callback)

The owner or administrator control that all users whether can open device. For example: all users are prohibited from opening the microphone(only available in the conference scenario).

Param	DESC
device	Device. More details, see: MediaDevice .
isDisable	true: disable user to open device, false: enable user to open device.

Note

The function only supports the [CONFERENCE](#) room type.

After the API call is successful:

If the device type is [MICROPHONE](#) , the SDK will notify the users in the room through [onAllUserMicrophoneDisableChanged](#) in \$TUIRoomObserver\$.

If the device type is [CAMERA](#) , the SDK will notify the users in the room through [onAllUserCameraDisableChanged](#) in \$TUIRoomObserver\$.

If the device type is [SCREEN_SHARING](#) , the SDK will notify the users in the room through [onScreenShareForAllUserDisableChanged](#) in \$TUIRoomObserver\$.

openRemoteDeviceByAdmin

openRemoteDeviceByAdmin

Request openRemoteDeviceByAdmin	(String userId
	TUIRoomDefine. MediaDevice device

	int timeout
	TUIRoomDefine.RequestCallback callback)

Request the remote user to open the media device (only support for administrators or room owner).

Param	DESC
device	Media device. More details, see: MediaDevice .
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.
userId	User ID.

Note

The function only supports the [CONFERENCE](#) room type.

After calling the API, the SDK will notify the requested user through [onRequestReceived](#) in \$TUIRoomObserver\$.

Return Desc:

TUIRequest Request body.

closeRemoteDeviceByAdmin

closeRemoteDeviceByAdmin

void closeRemoteDeviceByAdmin	(String userId
	TUIRoomDefine. MediaDevice device
	TUIRoomDefine.ActionCallback callback)

Close remote user media devices (only support for administrators or room owner).

Param	DESC
device	Media device. More details, see: MediaDevice .
userId	User ID.

Note

The function only supports the [CONFERENCE](#) room type.

After the API call is successful:

If the device type is [MICROPHONE](#), the SDK will notify the users in the room through [onUserAudioStateChanged](#) in `$TUIRoomObserver$`.

If the device type is [CAMERA](#) or [SCREEN_SHARING](#), the SDK will notify the users in the room through [onUserVideoStateChanged](#) in `$TUIRoomObserver$`.

applyToAdminToOpenLocalDevice

applyToAdminToOpenLocalDevice

Request applyToAdminToOpenLocalDevice	(TUIRoomDefine. MediaDevice device
	int timeout
	TUIRoomDefine.RequestCallback callback)

Apply to open the local media device (available to general users).

Param	DESC
device	Media device. More details, see: MediaDevice .
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

Note

The function only supports the [CONFERENCE](#) room type.

After the API call is successful, the SDK will notify the requested user through [onRequestReceived](#) in `$TUIRoomObserver$`.

Return Desc:

TUIRequest: Request body.

getSeatList

getSeatList

void getSeatList	(TUIRoomDefine.GetSeatListCallback callback)
------------------	--

Get seat list.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

lockSeatByAdmin

lockSeatByAdmin

void lockSeatByAdmin	(int seatIndex
	TUIRoomDefine. SeatLockParams lockParams
	TUIRoomDefine.ActionCallback callback)

Lock the seat (only support for administrators or room owner).

Param	DESC
lockParams	Seat lock parameter. More details, see: \$TUISeatLockParam\$.
seatIndex	Seat index.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

If the lockParams is lockSeat, it means that the current seat can not be taken by all users and the user will be kicked off if the seat was taken.

If the lockParams is lockVideo/lockAudio, it means that the current seat can not publish video/audio stream.

takeSeat

takeSeat

Request takeSeat	(int seatIndex
	int timeout
	TUIRoomDefine.RequestCallback callback)

Take the seat.

Param	DESC

seatIndex	Seat index. If the seat is not enabled or the sequence of seats is not concerned, just fill in -1.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

The user can publish audio/video stream after taking the seat when isSeatEnable is true.

After taking the seat successfully, the SDK will notify the users in the room through `$onSeatListChanged$` in `$TUIRoomObserver$`.

When the [TUISeatMode](#) is `ApplyToTake`, the operation to take seat need approval from the owner or administrator.

When the [TUISeatMode](#) is `FreeToTake`, you can take seat freely.

Return Desc:

TUIRequest Request body.

leaveSeat

leaveSeat

void leaveSeat	(TUIRoomDefine.ActionCallback callback)
----------------	---

Leave the seat.**Note**

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

The user can not publish audio/video stream after leaving the seat when isSeatEnable is true.

After leaving seat successfully, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

moveToSeat

moveToSeat

void moveToSeat	(int targetSeatIndex
	TUIRoomDefine.ActionCallback callback)

Move to seat.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After moving seat successfully, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

takeUserOnSeatByAdmin

takeUserOnSeatByAdmin

Request takeUserOnSeatByAdmin	(int seatIndex
	String userId
	int timeout
	TUIRoomDefine.RequestCallback callback)

Invite user to take the seat (only support for administrators or room owner).

Param	DESC
seatIndex	Seat index.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the API call is successful, the SDK will notify the invited user through [onRequestReceived](#) in `$TUIRoomObserver$`.

Return Desc:

TUIRequest: Request body.

kickUserOffSeatByAdmin

kickUserOffSeatByAdmin

--	--

void kickUserOffSeatByAdmin	(int seatIndex
	String userId
	TUIRoomDefine.ActionCallback callback)

Kick off the user from seat (only support for administrators or room owner).

Param	DESC
seatIndex	Seat index. If the seat is not enabled and the sequence of seats is not concerned, just fill in -1.
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After the API call is successful, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

getSeatApplicationList

getSeatApplicationList

void getSeatApplicationList	(TUIRoomDefine.RequestListCallback callback)
-----------------------------	--

Get the request list of users who want to take the seat in the room (only support for administrators or room owner).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

disableSendingMessageByAdmin

disableSendingMessageByAdmin

void disableSendingMessageByAdmin	(String userId
	boolean isDisable
	TUIRoomDefine.ActionCallback callback)

Disable the ability of remote users to send messages (only support for administrators or room owner).

Param	DESC
isDisable	true: disable user to send message, false: enable user to send message.
userId	User ID.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After disabling the ability of remote users to send messages, the SDK notifies the disabled user through [onSendMessageForUserDisableChanged](#) in `$TUIRoomObserver$`.

disableSendingMessageForAllUser

disableSendingMessageForAllUser

void disableSendingMessageForAllUser	(boolean isDisable
	TUIRoomDefine.ActionCallback callback)

Disable the ability of all users to send messages (only support for administrators or room owner).

Param	DESC
isDisable	true: disable all users to send message, false: enable all users to send message.

Note

The function only supports the [CONFERENCE](#) room type.

After disabling the ability of all users to send messages, the SDK notifies users in the room through [onSendMessageForAllUserDisableChanged](#) in `$TUIRoomObserver$`.

cancelRequest

cancelRequest

void cancelRequest	(String requestId
	TUIRoomDefine.ActionCallback callback)

Cancel request.

Param	DESC
requestId	Request ID (get from the sent request).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After cancelling a request, the SDK will notify the requested user through [onRequestCancelled](#) in `$TUIRoomObserver$`.

The API can be used to cancel a request that has been sent.

responseRemoteRequest

responseRemoteRequest

void responseRemoteRequest	(String requestId
	boolean agree
	TUIRoomDefine.ActionCallback callback)

Response request.

Param	DESC
agree	YES: Agree the request, NO: Reject the request.
requestId	Request ID (get from the sent request or notification of the OnRequestReceived event).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

When received a request, you can use this API to reply the received request.

getTRTCCloud

getTRTCCloud

Get the TRTC instance object.

setBeautyLevel

setBeautyLevel

void setBeautyLevel	(int beautyStyle
	float beautyLevel)

Set the beauty level.

Param	DESC
beautyLevel	Beauty level, the value range is 0 - 9; 0 indicates to disable the filter, and 9 indicates the most obvious effect.
beautyStyle	Beauty style: TXBeautyStyleSmooth: Smooth; TXBeautyStyleNature: Natural; TXBeautyStylePitu: Pitu style.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

setWhitenessLevel

setWhitenessLevel

void setWhitenessLevel	(float whitenessLevel)
------------------------	------------------------

Set whitening level.

Param	DESC
whitenessLevel	Whitening level, ranging from 0 - 9; 0 indicates to disable the filter, and 9 indicates the most obvious effect.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getExtension

getExtension

Object getExtension	(TUICommonDefine. ExtensionType extensionType)
---------------------	--

Get the extension.

Param	DESC
extensionType	Extension type. More deatils, see TUIExtensionType .

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getMediaDeviceManager

getMediaDeviceManager

Get device management class.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

getLiveConnectionManager

getLiveConnectionManager

Get live-connection management class.

Note

The function supports the [LIVE](#) room type.

getLiveBattleManager

getLiveBattleManager

Get live-battle management class.

Note

The function supports the [LIVE](#) room type.

callExperimentalAPI

callExperimentalAPI

--	--

Object callExperimentalAPI	(String jsonStr)
----------------------------	------------------

Call experimental APIs.

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

TUIRoomObserver

Last updated : 2025-02-13 16:38:50

Copyright (c) 2024 Tencent. All rights reserved.
Module: TUIRoomObserver @ TUIKitEngine.
Function: TUIRoomEngine event callback APIs.

TUIRoomObserver

TUIRoomObserver

FuncList	DESC
onError	Error event callback.
onKickedOffLine	The current user was kicked offline.
onUserSigExpired	The current user signature is expired.
onRoomNameChanged	The name of the room has changed.
onAllUserMicrophoneDisableChanged	The status of disabling to open microphone has changed for all users.
onAllUserCameraDisableChanged	The status of disabling to open camera has changed for all users.
onScreenShareForAllUserDisableChanged	The status of disabling to open screen sharing has changed for all users.
onSendMessageForAllUserDisableChanged	The status of disabling to send message has changed for all users.
onRoomDismissed	Room was dismissed.
onKickedOutOfRoom	The current user has been kicked off from the room.
onRoomSeatModeChanged	The room seat mode has changed.
onRoomUserCountChanged	The count of user in the room has changed.
onRoomMetadataChanged	The key-value of room metadata has changed.

onRemoteUserEnterRoom	Remote user entered room.
onRemoteUserLeaveRoom	Remote user left room.
onUserInfoChanged	User information has changed in the room.
onUserVideoStateChanged	The status of the user has video stream changed.
onUserAudioStateChanged	The status of the user has audio stream changed.
onUserVoiceVolumeChanged	User volume changed.
onSendMessageForUserDisableChanged	The status of disabling to send message has changed for user.
onUserNetworkQualityChanged	The User network status changed.
onUserScreenCaptureStopped	Screen sharing stopped.
onSeatListChanged	Seat list changed.
onKickedOffSeat	The user was kicked off the seat.
onRequestReceived	Receive a request message.
onRequestCancelled	Received a cancelled request.
onRequestProcessed	Receive a request to be processed by other administrator/owner.

onError

onError

void onError	(TUICommonDefine. Error errorCode
	String message)

Error event callback.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Callback error events when entering a room or opening a device.

Param	DESC
errorCode	Error code. More details, see: Error .

message	Error message.
---------	----------------

onKickedOffLine

onKickedOffLine

void onKickedOffLine	(String message)
----------------------	------------------

The current user was kicked offline.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user is kicked offline.

Param	DESC
message	Description of being kicked off.

onUserSigExpired

onUserSigExpired

The current user signature is expired.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's signature expires.

onRoomNameChanged

onRoomNameChanged

void onRoomNameChanged	(String roomId
	String roomName)

The name of the room has changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when the room name changes.

Param	DESC
roomId	Room ID.

roomName	Room name.
----------	------------

onAllUserMicrophoneDisableChanged

onAllUserMicrophoneDisableChanged

void onAllUserMicrophoneDisableChanged	(String roomId
	boolean isDisable)

The status of disabling to open microphone has changed for all users.

Param	DESC
isDisable	true: disable user to open microphone false: enable user to open microphone.
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type..

Called when the microphone disable status changes for all users.

onAllUserCameraDisableChanged

onAllUserCameraDisableChanged

void onAllUserCameraDisableChanged	(String roomId
	boolean isDisable)

The status of disabling to open camera has changed for all users.

Param	DESC
isDisable	true: disable user to open camera false: enable user to open camera.
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type..

Called when the camera disable status changes for all users.

onScreenShareForAllUserDisableChanged

onScreenShareForAllUserDisableChanged

void onScreenShareForAllUserDisableChanged	(String roomId
	boolean isDisable)

The status of disabling to open screen sharing has changed for all users.

Param	DESC
isDisable	true: disable user to open screen sharing false: enable user to open screen sharing.
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type..

Called when the screen sharing permissions change for all users.

onSendMessageForAllUserDisableChanged

onSendMessageForAllUserDisableChanged

void onSendMessageForAllUserDisableChanged	(String roomId
	boolean isDisable)

The status of disabling to send message has changed for all users.

Param	DESC
isDisable	true: disable user to send message false: enable user to send message.
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type..

Called when the message sending permissions change for all users.

onRoomDismissed

onRoomDismissed

void onRoomDismissed	(String roomId
	TUIRoomDefine. RoomDismissedReason reason)

Room was dismissed.

Param	DESC
reason	The reason why the room was dismissed. More details, see: RoomDismissedReason .
roomId	Room ID.

Note

The function only supports the [CONFERENCE](#) room type..

Called when the room is dismissed.

onKickedOutOfRoom

onKickedOutOfRoom

void onKickedOutOfRoom	(String roomId
	TUIRoomDefine. KickedOutOfRoomReason reason
	String message)

The current user has been kicked off from the room.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user is kicked out of the room.

Param	DESC
message	Description of being kicked off.
reason	Reason for being kicked off.
roomId	Room ID.

onRoomSeatModeChanged

onRoomSeatModeChanged

void onRoomSeatModeChanged	(String roomId
	TUIRoomDefine. SeatMode seatMode)

The room seat mode has changed.

Param	DESC
roomId	: Room ID.
seatMode	: Seat mode. More details, see TUISeatMode . #### [REMARK] The function supports the CONFERENCE and LIVE room types. Called when the seat mode of the room changes.

onRoomUserCountChanged

onRoomUserCountChanged

void onRoomUserCountChanged	(String roomId
	int userCount)

The count of user in the room has changed.

Param	DESC
roomId	Room ID.
userCount	Count of user. #### [REMARK] The function supports the CONFERENCE and LIVE room types. Called when the user count of the room changes.

onRoomMetadataChanged

onRoomMetadataChanged

--	--

void onRoomMetadataChanged	(String key
	String value)

The key-value of room metadata has changed.

Param	DESC
key	The key of room metadata.
value	The value of room metadata. #### [REMARK] The function supports the CONFERENCE and LIVE room types. Called when the custom information of the room changes.

onRemoteUserEnterRoom

onRemoteUserEnterRoom

void onRemoteUserEnterRoom	(String roomId
	TUIRoomDefine. UserInfo userInfo)

Remote user entered room.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a remote user enters the room.

Param	DESC
roomId	Room ID.
userInfo	User information. More details, see TUIUserInfo .

onRemoteUserLeaveRoom

onRemoteUserLeaveRoom

void onRemoteUserLeaveRoom	(String roomId
	TUIRoomDefine. UserInfo userInfo)

Remote user left room.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a remote user leaves the room.

Param	DESC
roomId	Room ID.
userInfo	User information. More details, see TUIUserInfo .

onUserInfoChanged

onUserInfoChanged

void onUserInfoChanged	(TUIRoomDefine. UserInfo userInfo
	List<TUIRoomDefine. UserInfoModifyFlag > modifyFlag)

User information has changed in the room.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's information changes.

Param	DESC
modifyFlag	Modifiable parameter. More details, see UserInfoModifyFlag .
userInfo	User information. More details, see TUIUserInfo .

onUserVideoStateChanged

onUserVideoStateChanged

void onUserVideoStateChanged	(String userId
	TUIRoomDefine. VideoStreamType streamType
	boolean hasVideo
	TUIRoomDefine. ChangeReason reason)

The status of the user has video stream changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's video state changes.

Param	DESC
hasVideo	The current user whether has video stream.
reason	The reason why the video stream changed: BY_SELF : Changed by self BY_ADMIN : Changed by administrator.
streamType	Video stream type. More details, see TUIVideoStreamType .
userId	User ID.

onUserAudioStateChanged

onUserAudioStateChanged

void onUserAudioStateChanged	(String userId
	boolean hasAudio
	TUIRoomDefine. ChangeReason reason)

The status of the user has audio stream changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's audio state changes.

Param	DESC
hasAudio	The current user whether has audio stream.
reason	The reason why the video stream changed: BY_SELF : Changed by self BY_ADMIN : Changed by administrator.
userId	User ID.

onUserVoiceVolumeChanged

onUserVoiceVolumeChanged

void onUserVoiceVolumeChanged	(Map<String, Integer> volumeMap)
-------------------------------	----------------------------------

User volume changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's voice volume changes.

Param	DESC
volumeMap	: User volume dictionary key: userId, value: the volume of all speaking users, with a value range of 0 - 100.

onSendMessageForUserDisableChanged

onSendMessageForUserDisableChanged

void onSendMessageForUserDisableChanged	(String roomId
	String userId
	boolean isDisable)

The status of disabling to send message has changed for user.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's message sending permissions change.

Param	DESC
isDisable	true: disable user to send message false: enable user to send message.
userId	User ID.

onUserNetworkQualityChanged

onUserNetworkQualityChanged

void onUserNetworkQualityChanged	(Map<String, TUICommonDefine. NetworkInfo > networkMap)
----------------------------------	---

The User network status changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's network quality changes.

Param	DESC
networkMap	User network status Map. More details, see NetworkInfo .

onUserScreenCaptureStopped

onUserScreenCaptureStopped

void onUserScreenCaptureStopped	(int reason)
---------------------------------	--------------

Screen sharing stopped.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user's screen capture stops.

Param	DESC
reason	Stop reason, 0: user actively stops; 1: the screen or the window is closed ; 2: the status of the screen or the window has changed (such as device disconnect).

onSeatListChanged

onSeatListChanged

void onSeatListChanged	(List<TUIRoomDefine. SeatInfo > seatList
	List<TUIRoomDefine. SeatInfo > seatedList
	List<TUIRoomDefine. SeatInfo > leftList)

Seat list changed.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when the seat list changes.

Param	DESC
leftList	List of newly leave-seat users.
seatList	The latest user list on seat, including new users.
seatedList	List of newly take-seat users.

onKickedOffSeat

onKickedOffSeat

void onKickedOffSeat	(int seatIndex
----------------------	----------------

	TUIRoomDefine. UserInfo operateUser)
--	--

The user was kicked off the seat.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a user is kicked off a seat.

Param	DESC
operateUser	User information of the owner/administrator who kicked the user.
seatIndex	Seat index.

onRequestReceived

onRequestReceived

void onRequestReceived	(TUIRoomDefine. Request request)
------------------------	--

Receive a request message.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a request is received.

Param	DESC
request	Request content. More details, see Request .

onRequestCancelled

onRequestCancelled

void onRequestCancelled	(TUIRoomDefine. Request request
	TUIRoomDefine. UserInfo operateUser)

Received a cancelled request.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a request is cancelled.

Param	DESC

operateUser	Operator information.
request	Request content. More details, see Request .

onRequestProcessed

onRequestProcessed

void onRequestProcessed	(TUIRoomDefine. Request request
	TUIRoomDefine. UserInfo operateUser)

Receive a request to be processed by other administrator/owner.

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

Called when a request is processed.

Param	DESC
operateUser	Operator information.
request	Request content. More details, see Request .

TUIRoomDeviceManager

Last updated : 2025-02-13 16:38:50

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUIRoomDeviceManager @ TUIKitEngine.

Function: Device testing and management APIs.

TUIRoomDeviceManager

TUIRoomDeviceManager

FuncList	DESC
isFrontCamera	Query whether the front camera is being used (only available for mobile OS).
switchCamera	Switch to the front/rear camera (only available for mobile OS).
isAutoFocusEnabled	Query whether automatic face detection is supported (only available for mobile OS).
enableCameraAutoFocus	Enable auto focus (only available for mobile OS).
enableCameraTorch	Enable/Disable flash, i.e., the torch mode (only available for mobile OS).
setAudioRoute	Set the audio route (only available for mobile OS).

EnumType

EnumType	DESC
AudioRoute	Audio routing (the route via which audio is played).

isFrontCamera

isFrontCamera

Query whether the front camera is being used (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

switchCamera

switchCamera

void switchCamera	(boolean front Camera)
-------------------	---

Switch to the front/rear camera (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

isAutoFocusEnabled

isAutoFocusEnabled

Query whether automatic face detection is supported (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

enableCameraAutoFocus

enableCameraAutoFocus

void enableCameraAutoFocus	(boolean enabled)
----------------------------	-------------------

Enable auto focus (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

After auto focus is enabled, the camera will automatically detect and always focus on faces.

enableCameraTorch

enableCameraTorch

void enableCameraTorch	(boolean enabled)
------------------------	-------------------

Enable/Disable flash, i.e., the torch mode (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

setAudioRoute

setAudioRoute

void setAudioRoute	(AudioRoute route)
--------------------	-------------------------------------

Set the audio route (only available for mobile OS).

Note

The function supports the [CONFERENCE](#) and [LIVE](#) room types.

A mobile phone has two audio playback devices: the receiver at the top and the speaker at the bottom.

If the audio route is set to the receiver, the volume is relatively low, and audio can be heard only when the phone is put near the ear. This mode has a high level of privacy and is suitable for answering calls.

If the audio route is set to the speaker, the volume is relatively high, and there is no need to put the phone near the ear.

TUIAudioRoute

TUIAudioRoute

Audio routing (the route via which audio is played).

Enum	Value	DESC
SPEAKERPHONE	0	Speakerphone: The speaker at the bottom is used for playback. With relatively high volume, it is used to play music out loud.
EARPIECE	1	Earpiece: The speaker at the top is used for playback. With relatively low volume, it is suitable for call scenarios that require privacy.

TUILiveListManager

Last updated : 2025-02-13 16:38:51

Copyright (c) 2024 Tencent. All rights reserved.
Module: TUILiveListManager @ TUIKitEngine.
Function: Live Room list APIS, the functions on this webpage only support to **LIVE** room type.

TUILiveListManager

TUILiveListManager

FuncList	DESC
onLiveInfoChanged	Live information changed callback.
setObserver	Set the event observer.
setLiveInfo	Modify live information.
getLiveInfo	Get live information.
fetchLiveList	Get a list of live rooms, with a maximum of 50 returned per fetch.

StructType

FuncList	DESC
LiveInfo	Live information.

EnumType

EnumType	DESC
LiveModifyFlag	The TUILiveInfo modifiable parameter key.

onLiveInfoChanged

onLiveInfoChanged

void onLiveInfoChanged	(LiveInfo liveInfo
	List< LiveModifyFlag > modifyFlagList)

Live information changed callback.

Param	DESC
liveInfo	Live information.
modifyFlag	Modifiable parameter. More details, see TUILiveModifyFlag .

setObserver

setObserver

void setObserver	(Observer observer)
------------------	---------------------

Set the event observer.

You can use `TUILiveListManagerObserver` to receive live room event.

Param	DESC
observer	Listening instance.

setLiveInfo

setLiveInfo

void setLiveInfo	(LiveInfo liveInfo
	List< LiveModifyFlag > modifyFlagList
	TUIRoomDefine.ActionCallback callback)

Modify live information.

Param	DESC
-------	------

liveInfo	Live information.
modifyFlag	Modifiable parameter. More details, see TUILiveModifyFlag .

getLiveInfo

getLiveInfo

void getLiveInfo	(String roomId
	LiveInfoCallback callback)

Get live information.

Param	DESC
roomId	Room ID.

fetchLiveList

fetchLiveList

void fetchLiveList	(String cursor
	int count
	LiveInfoListCallback callback)

Get a list of live rooms, with a maximum of 50 returned per fetch.

Param	DESC
count	Set the number of live lists to be obtained in each request.
cursor	Set the index of live lists in each request.

TUILiveModifyFlag

TUILiveModifyFlag

The TUILiveInfo modifiable parameter key.

Enum	Value	DESC
ACTIVITY_STATUS	0x0100	ActivityStatus: Live room active status, support for custom settings.
COVER_URL	0x0200	CoverUrl: Live room cover.
CATEGORY	0x0400	Category: Live room category.
PUBLISH	0x2000	Publish: Live room public flag.
BACKGROUND_URL	0x40000	BackgroundUrl: Live room background.

TUILiveInfo

TUILiveInfo**Live information.**

EnumType	DESC
activityStatus	Live room active status: User-defined tag.
backgroundUrl	CoverUrl: Live room background, maximum support is 200 bytes.
categoryList	Live room category tags, a single room supports up to 3 tags.
coverUrl	CoverUrl: Live room cover, maximum support is 200 bytes.
isPublicVisible	The public parameter can control whether the live room can be seen by others.
roomInfo	Room information (Read only). More details, see TUIRoomInfo .
viewCount	Total views (the count of user entering live room).

TUILiveConnectionManager

Last updated : 2025-02-13 16:38:51

Copyright (c) 2024 Tencent. All rights reserved.
Module: TUILiveConnectionManager @ TUIKitEngine
Function: Live-Connection APIS, the functions on this webpage only support to **LIVE** room type.

TUILiveConnectionManager

TUILiveConnectionManager

FuncList	DESC
onConnectionUserListChanged	Callback for connected users changed.
onConnectionRequestReceived	Callback for received the connection invitation
onConnectionRequestCancelled	Callback for canceled the connection invitation
onConnectionRequestAccept	Callback for accepted the connection invitation
onConnectionRequestReject	Callback for rejected the connection invitation
onConnectionRequestTimeout	Callback for timeout the connection invitation
addObserver	Add event callback
removeObserver	Remove event callback
requestConnection	Request connection invitation
cancelConnectionRequest	Cancel request about connection invitation
acceptConnection	Accept the connection invitation
rejectConnection	Reject the connection invitation
disconnect	Exit the connection.

StructType

--	--

FuncList	DESC
ConnectionUser	Connection User Info

EnumType

EnumType	DESC
ConnectionCode	Connection Request Status

onConnectionUserListChanged

onConnectionUserListChanged

void onConnectionUserListChanged	(List< ConnectionUser > connectedList
	List< ConnectionUser > joinedList
	List< ConnectionUser > leavedList)

Callback for connected users changed.

Param	DESC
connectedList	List of connected users.
joinedList	List of joined connected users.
leavedList	List of leaved connected users.

onConnectionRequestReceived

onConnectionRequestReceived

void onConnectionRequestReceived	(ConnectionUser inviter
	List< ConnectionUser > inviteeList
	String extensionInfo)

Callback for received the connection invitation

Param	DESC
extensionInfo	Extension info.
inviteeList	The list of ConnectionUser about the invitee in current connection.
inviter	The ConnectionUser about the inviter.

onConnectionRequestCancelled

onConnectionRequestCancelled

void onConnectionRequestCancelled	(ConnectionUser inviter)
-----------------------------------	---

Callback for canceled the connection invitation

Param	DESC
inviter	The ConnectionUser about the inviter.

onConnectionRequestAccept

onConnectionRequestAccept

void onConnectionRequestAccept	(ConnectionUser invitee)
--------------------------------	---

Callback for accepted the connection invitation

Param	DESC
invitee	The ConnectionUser about the invitee.

onConnectionRequestReject

onConnectionRequestReject

void onConnectionRequestReject	(ConnectionUser invitee)
--------------------------------	---

Callback for rejected the connection invitation

--	--

Param	DESC
invitee	The ConnectionUser about the invitee.

onConnectionRequestTimeout

onConnectionRequestTimeout

void onConnectionRequestTimeout	(ConnectionUser inviter
	ConnectionUser invitee)

Callback for timeout the connection invitation

Param	DESC
invitee	The ConnectionUser about the invitee.
inviter	The ConnectionUser about the inviter.

addObserver

addObserver

void addObserver	(Observer observer)
------------------	---------------------

Add event callback

Param	DESC
observer	The instance being listened to.

removeObserver

removeObserver

void removeObserver	(Observer observer)
---------------------	---------------------

Remove event callback

Param	DESC
-------	------

observer	The instance being listened to.
----------	---------------------------------

requestConnection

requestConnection

void requestConnection	(List<String> roomIdList
	int timeout
	String extensionInfo
	ConnectionRequestCallback callback)

Request connection invitation

Param	DESC
extensionInfo	Extension info.
roomIdList	The list of room IDs will be invited.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

cancelConnectionRequest

cancelConnectionRequest

void cancelConnectionRequest	(List<String> roomIdList
	TUIRoomDefine.ActionCallback callback)

Cancel request about connection invitation

Param	DESC
roomIdList	The list of room IDs whose connection requests will be canceled.

acceptConnection

acceptConnection

void acceptConnection	(String roomId
	TUIRoomDefine.ActionCallback callback)

Accept the connection invitation

Param	DESC
roomId	The room ID of the inviter about connection invitation.

rejectConnection**rejectConnection**

void rejectConnection	(String roomId
	TUIRoomDefine.ActionCallback callback)

Reject the connection invitation

Param	DESC
roomId	The room ID of the inviter about connection invitation.

disconnect**disconnect**

void disconnect	(TUIRoomDefine.ActionCallback callback)
-----------------	---

Exit the connection.

Calling this interface will exit the room connection state, and can only be called in the connected state.

TUIConnectionCode**TUIConnectionCode****Connection Request Status**

Enum	Value	DESC
UNKNOWN	-1	default.
SUCCESS	0	Request success.
ROOM_NOT_EXISTS	1	Request room not exist.
CONNECTING	2	The room you are invited to connect to is already in the invitation list or is already connected.
CONNECTING_OTHER_ROOM	3	The room you are invited to connect to is connecting with other rooms.
CONNECTION_FULL	4	The current number of connections has reached the maximum limit.
RETRY	5	Please try again.

TUIConnectionUser

TUIConnectionUser

Connection User Info

EnumType	DESC
avatarUrl	The user avatar url of the connection user.
joinConnectionTime	The Timestamp of when the user joined the connection.
roomId	The room id of the connection user.
userId	The user id of the connection user.
userName	The user name of the connection user.

TUICommonDefine

Last updated : 2025-02-13 16:38:51

TUICommonDefine

StructType

FuncList	DESC
NetworkInfo	Network quality information.
NetworkQualityInfo	

EnumType

EnumType	DESC
Error	Error code definition.
NetworkQuality	Network quality.
ExtensionType	Extension type.
TUIAudioPlaybackDevice	
Camera	

TUIError

TUIError

Error code definition.

Enum	Value	DESC
SUCCESS	0	Operate successfully.
FAILED	-1	Unclassified error.

FREQ_LIMIT	-2	The operation was frequency-limited. Try again later.
REPEAT_OPERATION	-3	Operation is repeated.
SDKAPPID_NOT_FOUND	-1000	SDKAppID not found. Please confirm application information in the TRT
INVALID_PARAMETER	-1001	An invalid parameter was passed when the API was called.
SDK_NOT_INITIALIZED	-1002	Not logged in, please login first.
PERMISSION_DENIED	-1003	Failed to get permission. Audio/video permissions are currently denied. Please check whether the device permissions are enabled. You can use the following codes to handle in the Room scenario: The camera does not have system authorization: ERR_CAMERA_NOT_AUTHORIZED The microphone does not have system authorization: ERR_MICROPHONE_NOT_AUTHORIZED
REQUIRE_PAYMENT	-1004	This function requires additional payment. Please activate the corresponding payment in the TRTC console .
CAMERA_START_FAIL	-1100	Failed to turn the camera on.
CAMERA_NOT_AUTHORIZED	-1101	No permission to access the camera. Please check the system authorization.
CAMERA_OCCUPIED	-1102	The camera is occupied. Please check whether there are other processes using the camera.
CAMERA_DEVICE_EMPTY	-1103	There is currently no camera available.
MICROPHONE_START_FAIL	-1104	Failed to turn the microphone on.
MICROPHONE_NOT_AUTHORIZED	-1105	No permission to access the microphone. Please check the system authorization.
MICROPHONE_OCCUPIED	-1106	The microphone is occupied. Please check whether there are other processes using the microphone.

MICROPHONE_DEVICE_EMPTY	-1107	There is currently no microphone
GET_SCREEN_SHARING_TARGET_FAILED	-1108	Failed to obtain the screen sharin (screen or window). Please check recording permission.
START_SCREEN_SHARING_FAILED	-1109	Failed to start screen sharing. Ple whether anyone in the room is sh: screen.
ROOM_ID_NOT_EXIST	-2100	The room does not exist when en: room.
OPERATION_INVALID_BEFORE_ENTER_ROOM	-2101	This operation is valid after enteri room.
EXIT_NOT_SUPPORTED_FOR_ROOM_OWNER	-2102	The owner can not leave room. Co room type: You can transfer the o and then leaving room. Live room can only dismiss the room.
OPERATION_NOT_SUPPORTED_IN_CURRENT_ROOM_TYPE	-2103	The current room type does not s: operation.
ROOM_ID_INVALID	-2105	The roomId is invalid. It's must be ASCII characters (0x20-0x7e) and maximum length is 48 bytes.
ROOM_ID_OCCUPIED	-2106	The room ID is occupied, please : another room ID.
ROOM_NAME_INVALID	-2107	The room name is invalid. The m: length of the name is 30 bytes. If i Chinese, the character encoding i UTF-8.
ALREADY_IN_OTHER_ROOM	-2108	The current user is already in othe needs to leave room before joinin: room: A single roomEngine instance onl user entering one room. If you wa a different room, please leave roo use a new roomEngine instance.
NEED_PASSWORD	-2109	The current room needs a passwo: enter.

WRONG_PASSWORD	-2110	Error password for entering the rc
ROOM_USER_FULL	-2111	The room is full of users.
ROOM_METADATA_EXCEED_KEY_COUNT_LIMIT	-2112	The number of room metadata ke the limit.
ROOM_METADATA_EXCEED_VALUE_SIZE_LIMIT	-2113	The size of room metadata value the limit.
USER_NOT_EXIST	-2200	The user does not exist.
USER_NOT_ENTERED	-2201	The user is not in the current room
NEED_OWNER_PERMISSION	-2300	The opeation needs owner permis
NEED_ADMIN_PERMISSION	-2301	The opeation needs owner or adn permission.
REQUEST_NO_PERMISSION	-2310	The request does not have permis as canceling an request that was requested by oneself.
REQUEST_ID_INVALID	-2311	The request ID is invalid or has be processed.
REQUEST_ID_REPEAT	-2312	The request is repeated.
REQUEST_ID_CONFLICT	-2313	The request conflict.
MAX_SEAT_COUNT_LIMIT	-2340	The number of seats exceeds the number of seats in your package.
ALREADY_IN_SEAT	-2341	The current user is already on the
SEAT_OCCUPIED	-2342	The current seat is already occup
SEAT_LOCKED	-2343	The current seat is locked.
SEAT_INDEX_NOT_EXIST	-2344	The seat number does not exist.
USER_NOT_IN_SEAT	-2345	The current user is not on the sea
ALL_SEAT_OCCUPIED	-2346	The number of people on the seat
SEAT_NOT_SUPPORT_LINK_MIC	-2347	The current room does not suppo microphone before the seat is enz
OPEN_MICROPHONE_NEED_SEAT_UNLOCK	-2360	The current seat audio is locked,

		can't push audio stream when using microphone.
OPEN_MICROPHONE_NEED_PERMISSION_FROM_ADMIN	-2361	You need to apply the owner or administrator to open the microphone.
OPEN_CAMERA_NEED_SEAT_UNLOCK	-2370	The current seat video is locked, so you can't push video stream when using the seat.
OPEN_CAMERA_NEED_PERMISSION_FROM_ADMIN	-2371	You need to apply to the owner or administrator to open the camera.
OPEN_SCREEN_SHARE_NEED_SEAT_UNLOCK	-2372	The current seat video is locked. You need to unlock the seat before screen sharing can be enabled.
OPEN_SCREEN_SHARE_NEED_PERMISSION_FROM_ADMIN	-2373	You need to apply to the owner or administrator to enable screen sharing.
SEND_MESSAGE_DISABLED_FOR_ALL	-2380	All users can't send message in the room.
SEND_MESSAGE_DISABLED_FOR_CURRENT	-2381	You can't send message in the current room.
ROOM_ALREADY_CONNECTED	-3001	The room has been connected with another room.
ROOM_CONNECTED_IN_OTHER_ROOM	-3002	The room has been connected with another room.
MAX_CONNECTED_COUNT_LIMIT	-3003	The current room connection exceeds the maximum limit.

TUINetworkQuality

TUINetworkQuality

Network quality.

Enum	Value	DESC
UNKNOWN	0	Undefined.
EXCELLENT	1	The current network is excellent.
GOOD	2	The current network is good.

POOR	3	The current network is poor.
BAD	4	The current network is bad.
VERY_BAD	5	The current network is very bad.
DOWN	6	The current network does not meet TRTC's minimum requirements.

TUIExtensionType

TUIExtensionType

Extension type.

Enum	Value	DESC
DEVICE_MANAGER	1	Device management extension.
LIVE_LIST_MANAGER	2	Live management extension.
CONFERENCE_LIST_MANAGER	3	Conference list extensions.
CONFERENCE_INVITATION_MANAGER	4	Conference invitation extensions.
LIVE_LAYOUT_MANAGER	5	Live layout extensions.

TUINetworkInfo

TUINetworkInfo

Network quality information.

EnumType	DESC
delay	Network delay(ms).
downLoss	Downlink packet loss rate, unit (%). The smaller the value, the better the network. If downLoss is 0%, it means that the network quality of the downlink is very good and the data packets received from the cloud are basically not lost. If downLoss is 30%, it means that 30% of the audio and video data packets transmitted from the cloud to the SDK will be lost in the transmission link.
quality	Network quality.

upLoss	<p>Uplink packet loss rate, unit (%). The smaller the value, the better the network.</p> <p>If upLoss is 0%, it means that the network quality of the uplink is very good and the data packets uploaded to the cloud are basically not lost.</p> <p>If upLoss is 30%, it means that 30% of the audio and video data packets sent by the SDK to the cloud will be lost in the transmission link.</p>
userId	User ID.

TUIRoomDefine

Last updated : 2025-02-13 16:38:51

TUIRoomDefine

StructType

FuncList	DESC
RoomInfo	Room information.
LoginUserInfo	User login information.
UserInfo	User information in the room.
RoomVideoEncoderParams	Video encoding parameters.
SeatInfo	Room seat information.
SeatLockParams	Lock seat parameters.
UserVoiceVolume	User volume in the room.
Request	Request.
EnterRoomOptions	Enter a room parameters.

EnumType

EnumType	DESC
RoomType	Room type.
SeatMode	Seat mode.
MediaDevice	Types of media devices in the room.
Role	Type of user role in the room.
RoomDismissedReason	The reason of dismissing room.
VideoQuality	Video quality.

AudioQuality	Audio quality.
VideoStreamType	Video stream type.
ChangeReason	Reasons for audio and video status changed.
KickedOutOfRoomReason	Reasons for users being kicked off from the room.
ResolutionMode	Resolution mode.
RequestAction	Request type.
UserInfoModifyFlag	The TUIUserInfo modifiable parameter key.

TUIRoomType

TUIRoomType

Room type.

Enum	Value	DESC
CONFERENCE	1	Conference room, suitable for conference and education scenarios.
LIVE	2	Live room, suitable for live scenarios.

TUISeatMode

TUISeatMode

Seat mode.

Enum	Value	DESC
FREE_TO_TAKE	1	Free to take seat mode. Users can take the seat freely without applying.
APPLY_TO_TAKE	2	Apply to take seat mode. Users can take the seat after the owner or administrator approve.

TUIMediaDevice

TUIMediaDevice

Types of media devices in the room.

Enum	Value	DESC
MICROPHONE	1	Microphone.
CAMERA	2	Camera.
SCREEN_SHARING	3	Screen sharing.

TUIRole

TUIRole

Type of user role in the room.

Enum	Value	DESC
ROOM_OWNER	0	Owner, generally refers to the room creator.
MANAGER	1	Administrator.
GENERAL_USER	2	General user.

TUIRoomDismissedReason

TUIRoomDismissedReason

The reason of dismissing room.

Enum	Value	DESC
BY_OWNER	1	Dismissed by owner.
BY_SERVER	2	Dismissed by the room server.

TUIVideoQuality

TUIVideoQuality

Video quality.

Enum	Value	DESC
Q_360P	1	Low-definition 360P.
Q_540P	2	Standard definition 540P.
Q_720P	3	High definition 720P.
Q_1080P	4	Full high definition 1080P.

TUIAudioQuality

TUIAudioQuality**Audio quality.**

Enum	Value	DESC
SPEECH	0	Speech mode.
DEFAULT	1	Default mode.
MUSIC	2	Music mode.

TUIVideoStreamType

TUIVideoStreamType**Video stream type.**

Enum	Value	DESC
CAMERA_STREAM	0	HD camera video stream.
SCREEN_STREAM	1	Screen sharing video stream.
CAMERA_STREAM_LOW	2	Low-definition camera video stream.

TUIChangeReason

TUIChangeReason

Reasons for audio and video status changed.

Enum	Value	DESC
BY_SELF	0	Changed by self.
BY_ADMIN	1	Changed by the owner and administrator.

TUIKickedOutOfRoomReason

TUIKickedOutOfRoomReason

Reasons for users being kicked off from the room.

Enum	Value	DESC
BY_ADMIN	0	Kicked off by owner or administrator.
BY_LOGGED_ON_OTHER_DEVICE	1	Kicked off when other devices enter the room.
BY_SERVER	2	Kicked off by the server.

TUIResolutionMode

TUIResolutionMode

Resolution mode.

Enum	Value	DESC
LANDSCAPE	0	Landscape.
PORTRAIT	1	Portrait.

TUIRequestAction

TUIRequestAction

Request type.

--	--	--

Enum	Value	DESC
INVALID_ACTION	0	Invalid request.
REQUEST_TO_OPEN_REMOTE_CAMERA	1	Request the remote user to open the camera.
REQUEST_TO_OPEN_REMOTE_MICROPHONE	2	Request the remote user to open the microphone.
REQUEST_TO_TAKE_SEAT	4	Request to take seat.
REQUEST_REMOTE_USER_ON_SEAT	5	Request the remote user to take seat.
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_CAMERA	6	Request to open local camera by the administrator.
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_MICROPHONE	7	Request to open local microphone by the administrator.
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_SCREEN_SHARE	8	Request to enable the screen sharing by the administrator.

TUIUserInfoModifyFlag

TUIUserInfoModifyFlag

The TUIUserInfo modifiable parameter key.

Enum	Value	DESC
USER_ROLE	0x01	userRole changed.
NAME_CARD	0x02	nameCard changed.

TUIRoomInfo

TUIRoomInfo**Room information.**

EnumType	DESC
createTime	The room creation time (read-only).
isCameraDisableForAllUser	The status of disabling for all user to open the camera in room (optional parameter for creating a room), default value: false.
isMessageDisableForAllUser	The status of disabling for all user to send message in room (optional parameter for creating a room), default value: false.
isMicrophoneDisableForAllUser	The status of disabling for all user to open the microphone in room (optional parameter for creating a room), default value: false.
isScreenShareDisableForAllUser	The status of disabling for all user to start the screen sharing in room (optional parameter for creating a room), default value: false.
isSeatEnabled	The status of enabling seat.
maxSeatCount	Maximum number of seat.
memberCount	The count of members in the room (read-only).
name	Room name (Optional parameter, default is room ID, maximum support is 100 bytes).
ownerAvatarUrl	Room owner avatar URL: Default is the room creator's avatar URL (read-only).
ownerId	Owner ID: Default is the room creator (read-only).
ownerName	Room owner name: Default is the room creator's name (read-only).
password	Room password.
roomId	Room ID (String type, required parameter for creating room, maximum support is 48 bytes).
roomType	Room type (Optional parameter). More details, see: RoomType .
seatMode	Seat mode (only available after isSeatEnabled is true).

TUILoginUserInfo

TUILoginUserInfo

User login information.

EnumType	DESC
avatarUrl	User avatar URL.
customInfo	Custom information.
userId	User ID.
userName	User name.

TUIUserInfo

TUIUserInfo

User information in the room.

EnumType	DESC
avatarUrl	Use avatar URL.
hasAudioStream	The status of the user has audio stream, default value: false.
hasScreenStream	The status of the user has screen sharing stream, default value: false.
hasVideoStream	The status of the user has video stream, default value: false.
isMessageDisabled	The status of disabling the user to send message, default value: false.
nameCard	User nickname in the room, maximum support is 32 bytes.
roomCustomInfo	Room user custom information.
userId	User ID.
userName	User name.
userRole	User role type. Conference room roles only exist within the room. Once checked out and re-entered, the role becomes a general user. Live room can be set before entering the room, and as long as the room is not disbanded, the role still exists. More details, see: Role .

TUIRoomVideoEncoderParams

TUIRoomVideoEncoderParams

Video encoding parameters.

EnumType	DESC
bitrate	Target video bitrate.
fps	Video capture frame rate.
resolutionMode	Resolution mode. More details, see: ResolutionMode .
videoResolution	Video quality. More details, see: VideoQuality .

TUISeatInfo

TUISeatInfo

Room seat information.

EnumType	DESC
avatarUrl	User avatar URL.
index	Seat index.
isAudioLocked	The status of audio-locked for the seat. Default value: false.
isLocked	The status of locked for the seat. Default value: false.
isVideoLocked	The status of video-locked for the seat. Default value: false.
nameCard	User nickname in the room.
userId	User ID.
userName	User name.

TUISeatLockParams

TUISeatLockParams

Lock seat parameters.

EnumType	DESC
lockAudio	Lock seat audio, default value: false.
lockSeat	Lock seat, default value: false.
lockVideo	Lock seat camera, default value: false.

TUIUserVoiceVolume

TUIUserVoiceVolume

User volume in the room.

EnumType	DESC
userId	User ID.
volume	Volume: The volume of all users who are speaking, the value range is 0 - 100.

TUIRequest

TUIRequest

Request.

EnumType	DESC
avatarUrl	User avatar URL.
content	Request content.
nameCard	User nickname in the room.
requestAction	Request type. More details, see TUIRequestAction .
requestId	Request ID.
timestamp	Timestamp.

userId	User ID.
userName	User name.

TUIEnterRoomOptions

TUIEnterRoomOptions

Enter a room parameters.

EnumType	DESC
password	Room password.

iOS

UIKit API

VoiceRoomKit

Last updated : 2024-12-05 17:06:33

API Introduction

VoiceRoomKit is a component of the voice chat room **including UI interface**, using the VoiceRoomKit API, you can quickly implement a voice chat room through simple interfaces. If you want to experience and debug the voice chat room effect, please read [Run demo](#). If you want to directly integrate our features into your project, please read [Quick Integration \(TUILiveKit\)](#).

Note:

If you have your own UI design and want to develop it yourself in combination with VoiceRoomKit, we also provide a more flexible and feature-complete RTC Room Engine SDK. You can learn more about the features by viewing the [RTC Room Engine SDK API](#) documentation.

API Overview

API	Description
createInstance	Obtain a VoiceRoomKit object instance.
createRoom	Create a voice chat room live streaming room.
enterRoom	Enter a voice chat room live streaming room using roomId.

Note:

If this set of APIs including the UI does not meet your needs, we also provide a more flexible and feature-complete RTC Room Engine SDK. You can learn more about the features by viewing our [API Documentation](#).

API Details

createInstance

Obtain a VoiceRoomKit object instance.

```
static func createInstance() -> VoiceRoomKit
```

Return Value:VoiceRoomKit

createRoom

Create a voice chat room live streaming room.

```
func createRoom(roomId: String, params: CreateRoomParams)
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
roomId	String	Mandatory	-	Live Streaming Room ID
params	CreateRoomParams	Mandatory	-	Create Live Room Parameters

Return Value:void

enterRoom

Enter a voice chat room live streaming room using roomId.

```
func enterRoom(roomId: String)
```

Parameter:

Parameter	Type	Description	Default Value	Meaning
roomId	String	Mandatory	-	Live Streaming Room ID

Return Value:void

VoiceRoomDefine Introduction

VoiceRoomKit is the UIKit layer data model class for voice chat rooms, mainly including the following data structures:

CreateRoomParams

Parameters object when creating a voice chat room live streaming, mainly including the following configuration parameters:

--	--	--	--	--

Parameter	Type	Description	Default Value	Meaning
roomName	String		""	Voice chat room name
maxAnchorCount	Int		10	Maximum number of users on stage
seatMode	TUISeatMode	Mandatory		Mic On Mode is divided into the following two types: TUISeatModeFreeToTake: Free taking seat on mode, audience can freely take seat without applying. TUISeatModeApplyToTake: Apply taking seat on mode, audience needs the approval of the anchor or administrator before getting on the seat.

SeatGridView

Last updated : 2024-12-25 17:25:34

API Introduction

SeatGridView is a basic control we developed for the Voice Chat Room UIKit. This core control provides rich APIs such as opening/closing the voice chat room, managing seats within the live room like applying to join the seat inviting to speak, moving microphone positions, and kicking someone off the seat.

API Overview

API	Description
SeatGridView	Create SeatGridView object
startMicrophone	Enable local microphone
stopMicrophone	Disable local microphone
muteMicrophone	Pause publishing local audio stream
unmuteMicrophone	Resume publishing local audio stream
startVoiceRoom	Anchor creates live room and starts streaming
stopVoiceRoom	Anchor stops streaming and destroys live room
joinVoiceRoom	Audience joins an anchor's live room
leaveVoiceRoom	Audience leaves an anchor's live room
updateRoomSeatMode	Update Room Mic Mode
responseRemoteRequest	Anchor Responds to Microphone Application/Audience Responds to Microphone Invitation
cancelRequest	Anchor Cancels Microphone Invitation/Audience Cancels Microphone Application
takeSeat	Connect Mic
moveToSeat	Disconnect Mic

leaveSeat	Disconnect Mic
takeUserOnSeatByAdmin	Anchor Invites User to Connect Mic
kickUserOffSeatByAdmin	Anchor Kicks User off Mic
lockSeat	Anchor Locks Mic Position (including position lock, audio status lock, and video status lock)
setLayoutMode	Anchor Sets Layout Mode for Mic List
setSeatViewDelegate	Set the proxy for seat view
addObserver	Set event callbacks
removeObserver	Remove event callbacks

API Details

SeatGridView

Create an instance of SeatGridView object.

```
SeatGridView()
```

startMicrophone

Enable the local mic.

```
func startMicrophone(onSuccess: @escaping SGOnSuccess,  
                     onError: @escaping SGOnError)
```

Parameter:

Parameter	Type	Meaning
onSuccess	SGOnSuccess	The success callback
onError	SGOnError	The failure callback

stopMicrophone

Disable the local mic.

```
func stopMicrophone()
```

muteMicrophone

Pause publishing local audio stream.

```
func muteMicrophone()
```

unmuteMicrophone

Resume publishing local audio stream.

```
func unmuteMicrophone(onSuccess: @escaping SGOOnSuccess,  
                      onError: @escaping SGOOnError)
```

Parameter:

Parameter	Type	Meaning
onSuccess	SGOOnSuccess	The success callback
onError	SGOOnError	The failure callback

startVoiceRoom

Anchor creates live room and starts streaming.

```
func startVoiceRoom(roomInfo: TUIRoomInfo,  
                   onSuccess: @escaping SGOOnRoomSuccess,  
                   onError: @escaping SGOOnError)
```

Parameter:

Parameter	Type	Meaning
roomInfo	TUIRoomInfo	Information for creating a live streaming room
onSuccess	SGOOnRoomSuccess	Success callback (with new live room information)
onError	SGOOnError	The failure callback

stopVoiceRoom

Anchor stops streaming and destroys live room.

```
func stopVoiceRoom(onSuccess: @escaping SGOOnSuccess,  
                  onError: @escaping SGOOnError)
```

Parameter:

Parameter	Type	Meaning
-----------	------	---------

onSuccess	SOnSuccess	The success callback
onError	SOnError	The failure callback

joinVoiceRoom

Audience joins an anchor's live room.

```
func joinVoiceRoom(roomId: String,  
                   onSuccess: @escaping SOnRoomSuccess,  
                   onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
roomId	String	Live Streaming Room ID
onSuccess	SOnRoomSuccess	Success callback (with live room information)
onError	SOnError	The failure callback

leaveVoiceRoom

Audience leaves an anchor's live room.

```
func leaveVoiceRoom(onSuccess: @escaping SOnSuccess,  
                   onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
onSuccess	SOnSuccess	The success callback
onError	SOnError	The failure callback

updateRoomSeatMode

Update Room Mic Mode.

```
func updateRoomSeatMode(seatMode: TUISeatMode,  
                       onSuccess: @escaping SOnSuccess,  
                       onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
-----------	------	---------

seatMode	TUISeatMode	freeToTake: Free seat mode, audience can connect to seat freely without application; applyToTake: Apply seat mode, audience needs the anchor's approval to connect to seat.
onSuccess	SOnSuccess	The success callback
onError	SOnError	The failure callback

responseRemoteRequest

Anchor Responds to Microphone Application/Audience Responds to Microphone Invitation.

```
func responseRemoteRequest(userId: String,  
                           agree: Bool,  
                           onSuccess: @escaping SOnSuccess,  
                           onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
userId	String	Responding user's user ID, if the current role is an audience member, the ID can be left blank
agree	Bool	Whether to accept the request, true: accept the request, false: reject the request
onSuccess	SOnSuccess	The success callback
onError	SOnError	The failure callback

cancelRequest

Anchor Cancels Microphone Invitation/Audience Cancels Microphone Application

```
func cancelRequest(userId: String,  
                  onSuccess: @escaping SOnSuccess,  
                  onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
userId	String	Cancelled user's ID, if the current role is an audience member, the ID can be left blank
onSuccess	SOnSuccess	The success callback

onError	SOnError	The failure callback
---------	----------	----------------------

takeSeat

Request to Speak (In speaking mode, application is required)

```
func takeSeat(index: Int,  
             timeout: Int,  
             onAccepted: @escaping SOnRequestAccepted,  
             onRejected: @escaping SOnRequestRejected,  
             onCancelled: @escaping SOnRequestCancelled,  
             onTimeout: @escaping SOnRequestTimeout,  
             onError: @escaping SOnRequestError)
```

Parameter:

Parameter	Type	Meaning
index	Int	Mic position number for connecting
timeout	Int	Timeout period, unit: seconds. If set to 0, the SDK will not perform a timeout check or trigger a timeout callback
onAccepted	SOnRequestAccepted	Application for seat has been approved callback
onRejected	SOnRequestRejected	Application for seat has been rejected callback
onCancelled	SOnRequestCancelled	Application for seat has been canceled callback
onTimeout	SOnRequestTimeout	Application for seat has timed out callback
onError	SOnRequestError	Application for seat error callback

moveToSeat

Move Mic (only users who are already on the seat can call this function)

```
func moveToSeat(index: Int,  
               onSuccess: @escaping SOnSuccess,  
               onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
index	Int	Mic position number to move to
onSuccess	SOnSuccess	The success callback

onError	SOnError	The failure callback
---------	----------	----------------------

leaveSeat

Proactively disconnect mic

```
func leaveSeat(onSuccess: @escaping SOnSuccess, onError: @escaping SOnError)
```

Parameter:

Parameter	Type	Meaning
onSuccess	SOnSuccess	The success callback
onError	SOnError	The failure callback

takeUserOnSeatByAdmin

Anchor Invites User to Connect Mic

```
func takeUserOnSeatByAdmin(index: Int,
                            userId: String,
                            onAccepted: @escaping SOnRequestAccepted,
                            onRejected: @escaping SOnRequestRejected,
                            onCancelled: @escaping SOnRequestCancelled,
                            onTimeout: @escaping SOnRequestTimeout,
                            onError: @escaping SOnRequestError)
```

Parameter:

Parameter	Type	Meaning
index	Int	Invited Mic position number
userId	String	Invited user ID
timeout	Int	Timeout period, unit: seconds. If set to 0, the SDK will not perform a timeout check or trigger a timeout callback
onAccepted	SOnRequestAccepted	Invite to speak request approved callback
onRejected	SOnRequestRejected	Invite to speak request rejected callback
onCancelled	SOnRequestCancelled	Invite to speak request canceled callback
onTimeout	SOnRequestTimeout	Invite to speak request timeout callback
onError	SOnRequestError	Invite to speak request error callback

kickUserOffSeatByAdmin

Anchor Kicks User off Mic

```
func kickUserOffSeatByAdmin(userId: String,
                             onSuccess: @escaping SGOOnSuccess,
                             onError: @escaping SGOOnError)
```

Parameter:

Parameter	Type	Meaning
userId	String	User ID kicked off the Mic
onSuccess	SGOnSuccess	The success callback
onError	SGOnError	The failure callback

lockSeat

Microphone Mute, Anchor locks Mic Position (including position lock, audio status lock, and video status lock)

```
func lockSeat(index: Int,
               lockMode: TUISeatLockParams,
               onSuccess: @escaping SGOOnSuccess,
               onError: @escaping SGOOnError)
```

Parameter:

Parameter	Type	Meaning
seatIndex	int	Mic position number to be locked
params	TUISeatLockParams	Microphone Mute Parameters. See details: TUIRoomDefine.SeatLockParams
onSuccess	SGOnSuccess	The success callback
onError	SGOnError	The failure callback

setLayoutMode

Set the layout mode for the mic list.

```
func setLayoutMode(layoutMode: SGLayoutMode,
                    layoutConfig: SeatViewLayoutConfig? = nil)
```

Parameter:

--	--	--

Parameter	Type	Meaning
layoutModel	SGLayoutMode	Layout mode of the mic list, supports element layout, grid layout, vertical layout, and free layout.
layoutConfig	SGSeatViewLayoutConfig	Layout configuration information, effective only in free layout mode.

setSeatViewDelegate

Set the adapter for the mic view.

```
func setSeatViewDelegate(_ delegate: SGSeatViewDelegate)
```

Parameter:

Parameter	Type	Meaning
delegate	SGSeatViewDelegate	Seat view delegate

addObserver

Set event callbacks.

```
func addObserver(observer: SeatGridViewObserver)
```

Parameter:

Parameter	Type	Meaning
observer	SeatGridViewObserver	Callback object of core component

removeObserver

Remove event callbacks.

```
func removeObserver(observer: SeatGridViewObserver)
```

Parameter:

Parameter	Type	Meaning
observer	SeatGridViewObserver	Callback object of core component

Type Definition

Type	Description
SGLayoutMode	Layout modes of the seat list support element layout, grid layout, vertical layout, and custom layout
SGSeatViewLayoutRowAlignment	Alignment of seat layout
SGRequestType	Request type (apply to speak and invite to take seat)
SGSeatViewLayoutConfig	Microphone position layout configuration information
SGSeatViewLayoutRowConfig	Seating Layout Row Configuration Information
SGSeatViewDelegate	Seat view delegate

SGLayoutMode

Layout modes of the seat list

Type	Description
focus	Element Layout
grid	Grid Layout
vertical	Vertical layout
free	Customized Layout

SGSeatViewLayoutRowAlignment

Alignment of seat layout

Type	Description
start	Microphone position near the start
end	Microphone position near the end
center	Microphone position near the middle
spaceBetween	No space before the first and after the last seats, evenly distribute the remaining space between other seats
spaceAround	Distribute half of the space before the first and after the last seats, evenly distribute the remaining space between other seats

spaceEvenly	Evenly distribute the remaining space between all seats
-------------	---

SGRequestType

Request type

Type	Description
applyToTakeSeat	Apply to take seat
inviteToTakeSeat	Invite to take seat

SGSeatViewLayoutConfig

Microphone position layout configuration information

Type	Description
rowConfigs	List of all row configuration information in the seat layout, refer to SGSeatViewLayoutRowConfig
rowSpacing	Seat row spacing

SGSeatViewLayoutRowConfig

Seating Layout Row Configuration Information

Type	Description
count	Number of seats displayed in this row
seatSpacing	Horizontal spacing of each seat in this row (effective only when the alignment is START, END, or CENTER)
seatSize	Size of the seat layout in this row
alignment	Alignment of the layout in this row (SGSeatViewLayoutRowAlignment)

SGSeatViewDelegate

Seat view adapter interface, you can customize the display UI of each seat by implementing this interface.

API	Description
seatGridView:createSeatView	Callback when creating a single seat layout.
seatGridView:updateSeatView	Callback when updating the seat view.

[seatGridView:updateUserVolume](#)

Callback when updating user volume.

Event Callback Details

onAccepted

Application to speak/invite to speak request accepted.

```
onAccepted: (_ userInfo: TUIUserInfo) -> Void
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	Response to the current request's user information

onRejected

Application to speak/invite to speak request rejected.

```
onRejected: (_ userInfo: TUIUserInfo) -> Void
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	Response to the current request's user information

onCancelled

Application to speak/invite to speak request canceled.

```
onCancelled: (_ userInfo: TUIUserInfo) -> Void
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	User information for the canceled request

onTimeout

Application to speak/invite to speak request timed out.

```
onTimeout: (_ userInfo: TUIUserInfo) -> Void
```

Parameter:

Parameter	Type	Description
userInfo	UserInfo	User information for the initiated request

onError

Application to speak/invite to speak request error.

```
onError: (_ userInfo: TUIUserInfo, _ code: Int, _ message: String) -> Void
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	User information for the initiated request
error	TUICommonDefine.Error	Error code
message	String	Error message

seatGridView:createSeatView

Callback when creating a single seat layout, you need to return your custom view, the core view will help you create the view.

```
func seatGridView(_ view: SeatGridView, createSeatView seatInfo: TUISeatInfo) -> UI
```

Parameter:

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
seatInfo	SeatInfo	Seat information

Return value: UIView? (Use default seat view when the return value is nil)

seatGridView:updateSeatView

Callback when updating the seat view, you can update your seat view based on the seatInfo information returned by the callback.

```
func seatGridView(_ view: SeatGridView, updateSeatView seatInfo: TUISeatInfo, seatV
```

Parameter:

--	--	--

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
seatInfo	TUISeatInfo	Seat information
seatView	UIView	Current updated seat view

seatGridView:updateUserVolume

Callback when updating user volume, you can update your seat view based on the returned volume.

```
func seatGridView(_ view: SeatGridView, updateUserVolume volume: CGFloat, seatView:
```

Parameter:

Parameter	Type	Description
seatGridView	SeatGridView	Core components of voice chat room
volume	CGFloat	Mic position volume information
seatView	UIView	Current seat layout view with volume changes

SeatGridViewObserver Overview

Function List	Description
onRoomDismissed	Event of Room Termination Received
onKickedOutOfRoom	Event of Being Kicked Out of the Room Received
onSeatRequestReceived	Event of Request for Speaking/Invitation to Speak Received
onSeatRequestCancelled	Event of Request for Speaking/Invitation to Speak Canceled
onKickedOffSeat	Event of User Kicked Off Seat Received
onUserAudioStateChanged	Event of User Audio Status Changed
onSeatViewClicked	Seat View Click Event

SeatGridViewObserver Details

onRoomDismissed

Event of Live Room Destroyed

```
func onRoomDismissed(roomId: String)
```

Parameter:

Parameter	Type	Description
roomId	String	Room ID

onKickedOutOfRoom

Event of Being Kicked Out of the Room

```
func onKickedOutOfRoom(roomId: String,  
                        reason: TUIKickedOutOfRoomReason,  
                        message: String)
```

Parameter:

Parameter	Type	Description
roomId	String	Room ID
reason	TUIKickedOutOfRoomReason	Reason for Being Kicked Out
message	String	Description of Being Kicked Out

onSeatRequestReceived

Event of Request for Speaking/Invitation to Speak Received

```
func onSeatRequestReceived(type: SGRequestType,  
                           userInfo: TUIUserInfo)
```

Parameter:

Parameter	Type	Description
type	SGRequestType	Request type (Request for Speaking, Invitation to Speak)
userInfo	TUIUserInfo	Information of the user who sent the request

onSeatRequestCancelled

Event of Request for Speaking/Invitation to Speak Canceled

```
func onSeatRequestCancelled(type: SGRequestType,
                           userInfo: TUIUserInfo)
```

Parameter:

Parameter	Type	Description
type	SGRequestType	Request type (Request for Speaking, Invitation to Speak)
userInfo	TUIUserInfo	Information of the user who canceled the request

onKickedOffSeat

Event of User Kicked Off Seat

```
func onKickedOffSeat(userInfo: TUIUserInfo)
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	Information of the host who kicked the user off the seat

onUserAudioStateChanged

Event of User Audio Status Changed

```
func onUserAudioStateChanged(userInfo: TUIUserInfo,
                              hasAudio: Bool,
                              reason: TUIChangeReason)
```

Parameter:

Parameter	Type	Description
userInfo	TUIUserInfo	User Information
hasAudio	Bool	Is there an audio stream
reason	TUIChangeReason	Reason for audio stream change

onSeatViewClicked

Seat View Click Event

```
func onSeatViewClicked(seatView: UIView, seatInfo: TUISeatInfo)
```

Parameter:

Parameter	Type	Description
seatView	View	Currently clicked seat view object
seatInfo	TUISeatInfo	Seat information

Engine API

API Overview

Last updated : 2025-02-13 16:38:49

API OVERVIEW

Creating instances and event callback.

FuncList	DESC
destroySharedInstance	Destroy the TUIRoomEngine instance (singleton mode)
loginWithSDKAppId:userId:userSig:onSuccess:onError:	After creating a TUIRoomEngine instance, you should login with sdkAppId, userId and userSig then you can call TUIRoomEngine instance and other function.
logout:onError:	Log out of your account. If you are in the room, there will be active leaving room and destroying resource operations.
setSelfInfoWithUserName:avatarUrl:onSuccess:onError:	Update user name and avatar for logged-in user.
getSelfInfo	Return the basic information of the logged-in user, including nickname and avatar.
setSelfInfo:onSuccess:onError:	Update user basic information for logged-in user.
addObserver:	Set event observer.
removeObserver:	Remove event observer.

Room APIs.

FuncList	DESC
createRoom:onSuccess:onError:	Create a room.

<code>destroyRoom:onError:</code>	Dismiss the room.
<code>enterRoom:onSuccess:onError:</code>	Enter a room.
<code>enterRoom:roomType:onSuccess:onError:</code>	Enter a room.
<code>enterRoom:roomType:options:onSuccess:onError:</code>	Enter a room.
<code>exitRoom:onSuccess:onError:</code>	Exit the room.
<code>fetchRoomInfo:onError:</code>	Fetch room information.
<code>fetchRoomInfo:roomType:onSuccess:onError:</code>	Fetch room information.
<code>updateRoomNameByAdmin:onSuccess:onError:</code>	Update room name (only support for administrators or room owner).
<code>updateRoomSeatModeByAdmin:onSuccess:onError:</code>	Update room seat mode (only support for administrators or room owner).
<code>updateRoomPasswordByAdmin:onSuccess:onError:</code>	Update room password.
<code>getRoomMetadata:onSuccess:onError:</code>	Get room metadata.
<code>setRoomMetadataByAdmin</code>	Set room metadata, if the key already exists, update its value, if not, add the key.

Local user view rendering, video management.

FuncList	DESC
<code>setLocalVideoView:</code>	Set the local camera to preview the render view.
<code>openLocalCamera:quality:onSuccess:onError:</code>	Open the local camera.
<code>closeLocalCamera</code>	Close the local camera.
<code>startPushLocalVideo</code>	Start publishing local video stream, default enabled.
<code>stopPushLocalVideo</code>	Stop publishing local video stream.
<code>updateVideoQuality:</code>	Update video encoding quality.
<code>updateVideoQualityEx:params:</code>	Set the video encoding parameters.
<code>setVideoResolutionMode:resolutionMode:</code>	Set the video resolution mode (horizontal resolution or

	vertical resolution).
<code>enableGravitySensor:</code>	Turn on gravity sensing mode. (only available on mobile OS and the camera capture scene inside the SDK).
<code>startScreenCaptureByReplaykit:</code>	Start screen sharing (only available on mobile OS).
<code>startScreenCapture:onSuccess:onError:</code>	Start screen sharing (only available on Mac OS).
<code>stopScreenCapture</code>	Stop screen sharing.
<code>getScreenCaptureSources</code>	Get the sharable screen and windows (only available on Mac OS)
<code>selectScreenCaptureTarget:</code>	Select the screen or windows to share (only available on Mac OS)

Local User Audio Management.

FuncList	DESC
<code>openLocalMicrophone:onSuccess:onError:</code>	Open local microphone.
<code>closeLocalMicrophone</code>	Close the local microphone.
<code>updateAudioQuality:</code>	Update audio encoding quality.
<code>muteLocalAudio</code>	Pause publishing the local audio stream.
<code>unmuteLocalAudio:onError:</code>	Resume publishing the local audio stream.
<code>enableSystemAudioSharing:</code>	Enable system audio sharing

Remote user view rendering and video management.

FuncList	DESC
<code>setRemoteVideoView:streamType:view:</code>	Set the render view for remote user.
<code>startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:</code>	Start playing the remote user's video stream.

stopPlayRemoteVideo:streamType:	Stop playing the remote user's video stream.
muteRemoteAudioStream:isMute:	Mute the remote user's audio stream.

User information in the room.

FuncList	DESC
getUserList:onSuccess:onError:	Get the list of user in the room.
getUserInfo:onSuccess:onError:	Get user information.

User management in the room.

FuncList	DESC
changeUserRole:role:onSuccess:onError:	Change user role (only support for administrators or room owner).
changeUserNameCard:nameCard:onSuccess:onError:	Change user nickname in the room (only support to change all user for administrators or room owner, user can only change by self).
kickRemoteUserOutOfRoom:onSuccess:onError:	Kick the remote user out of the room (only support for administrators or room owner).
addCategoryTagForUsers:userList:onSuccess:onError:	Add a tag for the user (only support for administrators or room owner).
removeCategoryTagForUsers:userList:onSuccess:onError:	Remove tag for user (only support for room owner).
getUserListByTag:nextSequence:onSuccess:onError:	Get user information in the room based on the tag.
setCustomInfoForUser:onSuccess:onError:	Set custom information for room users.

User speech management in the room.

FuncList	DESC
disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:	The owner can administrate and control the users who can open the device. For example: users are prohibited from opening the microphone in available conference scenarios.
openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	Request to open the remote user's device (only supported for administrators or room owners).
closeRemoteDeviceByAdmin:device:onSuccess:onError:	Close the remote user's media devices (only supported for administrators or room owners).
applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	Apply to open the local user's device (available for general users).

Seat management in the room.

FuncList	DESC
getSeatList:onError:	Get seat list.

<code>lockSeatByAdmin:lockMode:onSuccess:onError:</code>	Lock the seat (only support for administrator or room owner).
<code>takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Take the seat.
<code>leaveSeat:onError:</code>	Leave the seat.
<code>moveToSeat:onSuccess:onError:</code>	Move to seat.
<code>takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Invite user to take the seat (only support for administrator or room owner).
<code>kickUserOffSeatByAdmin:userId:onSuccess:onError:</code>	Kick off the user from seat (only support for administrator or room owner).
<code>getSeatApplicationList:onError:</code>	Get the request list of users who want to take the seat in the room (only support for administrator or room owner).

Message.

FuncList	DESC
----------	------

disableSendingMessageByAdmin:isDisable:onSuccess:onError:	Disable the ability of remote users to send messages (only support for administrators or room owner).
disableSendingMessageForAllUser:onSuccess:onError:	Disable the ability of all users to send messages (only support for administrators or room owner).

Request Management.

FuncList	DESC
cancelRequest:onSuccess:onError:	Cancel request.
responseRemoteRequest:agree:onSuccess:onError:	Response request.

Advanced Features.

FuncList	DESC
getTRTCCloud	Get the TRTC instance object.
setBeautyLevel:beautyLevel:	Set the beauty level.
setWhitenessLevel:	Set whitening level.
getExtension:	Get the extension.
getMediaDeviceManager	Get device management class.
getLiveConnectionManager	Get live-connection management class.
getLiveBattleManager	Get live-battle management class.

Debug related.

FuncList	DESC
callExperimentalAPI:	Call experimental APIs.

Error event callback.

FuncList	DESC
onError:message:	Error event callback.

Login status event callback.

FuncList	DESC
onKickedOffLine:	The current user was kicked offline.
onUserSigExpired	The current user signature is expired.

Event callback in the room.

FuncList	DESC
onRoomNameChanged:roomName:	The name of the room has changed.
onAllUserMicrophoneDisableChanged:isDisable:	The status of disabling to open microphone has changed for all users.
onAllUserCameraDisableChanged:isDisable:	The status of disabling to open camera has changed for all users.
onScreenShareForAllUserDisableChanged:isDisable:	The status of disabling to open screen sharing has changed for all users.
onSendMessageForAllUserDisableChanged:isDisable:	The status of disabling to send message has changed for all users.
onRoomDismissed:reason:	Room was dismissed.
onKickedOutOfRoom:reason:message:	The current user has been kicked off from the room.
onRoomSeatModeChanged:seatMode:	The room seat mode has changed.
onRoomUserCountChanged:userCount:	The count of user in the room has changed.
onRoomMetadataChanged:value:	The key-value of room metadata has

changed.

User event callback in the room.

FuncList	DESC
onRemoteUserEnterRoom:userInfo:	Remote user entered room.
onRemoteUserLeaveRoom:userInfo:	Remote user left room.
onUserInfoChanged:modifyFlag:	User information has changed in the room.
onUserVideoStateChanged:streamType:hasVideo:reason:	The status of the user has video stream changed.
onUserAudioStateChanged:hasAudio:reason:	The status of the user has audio stream changed.
onUserVoiceVolumeChanged	User volume changed.
onSendMessageForUserDisableChanged:userId:isDisable:	The status of disabling to send message has changed for user.
onUserNetworkQualityChanged:	The user network status changed.
onUserScreenCaptureStopped:	Screen sharing stopped.

Seat event callback in the room.

FuncList	DESC
onSeatListChanged:seated:left:	Seat list changed.
onKickedOffSeat:operateUser:	The user was kicked off the seat.

Request event callback.

FuncList	DESC
onRequestReceived:	Receive a request message.

<code>onRequestCancelled:operateUser:</code>	Received a cancelled request.
<code>onRequestProcessed:operateUser:</code>	Receive a request to be processed by other administrator/owner.

Deprecated callbacks.

FuncList	DESC
<code>onDeviceChanged:type:state:</code>	Local device added.

Device management APIs.

FuncList	DESC
<code>isFrontCamera</code>	Query whether the front camera is being used (only available for mobile OS).
<code>switchCamera:</code>	Switch to the front/rear camera (only available for mobile OS).
<code>isAutoFocusEnabled</code>	Query whether automatic face detection is supported (only available for mobile OS).
<code>enableCameraAutoFocus:</code>	Enable auto focus (only available for mobile OS).
<code>enableCameraTorch:</code>	Enable/Disable flash, i.e., the torch mode (only available for mobile OS).
<code>setAudioRoute:</code>	Set the audio route (only available for mobile OS).
<code>setObserver:</code>	Set event observer.
<code>startCameraDeviceTest:</code>	Start camera testing (only available for desktop OS).
<code>stopCameraDeviceTest</code>	End camera testing (only available for desktop OS).
<code>startMicDeviceTest:playback:</code>	Start microphone testing (only available for desktop OS).
<code>stopMicDeviceTest</code>	End microphone testing (only available for desktop OS)
<code>startSpeakerDeviceTest:</code>	Start speaker testing (only available for desktop OS).
<code>stopSpeakerDeviceTest</code>	End speaker testing (only available for desktop OS).
<code>getDevicesList:callback:</code>	Get the device list (only available for desktop OS).

setCurrentDevice:deviceId:	Set the device to use (only available for desktop OS).
getCurrentDevice:callback:	Get the device currently in use (only available for desktop OS).

TUIRoomEngine

Last updated : 2025-02-13 16:38:50

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUIRoomEngine @ TUIKitEngine.
Function: TUIRoomEngine Main function APIs.
Version: 2.9.0

TUIRoomEngine

TUIRoomEngine

FuncList	DESC
destroySharedInstance	Destroy the TUIRoom instance (singletons)
loginWithSDKAppld:userId:userSig:onSuccess:onError:	After creating a TUIRoom instance, you should log in with the sdkAppId, userId, and userSig. After logging in, you can call the TUIRoom instance's other functions.
logout:onError:	Log out of the account. If the user is in the room, there will be an active leave room and destroying resource operation.

<code>setSelfInfoWithUserName:avatarUrl:onSuccess:onError:</code>	Update u name and avatar for logged-in
<code>getSelfInfo</code>	Return th informati the logge user, incl nickname avatar.
<code>setSelfInfo:onSuccess:onError:</code>	Update u basic informati logged-in
<code>addObserver:</code>	Set event observer.
<code>removeObserver:</code>	Remove e observer.
<code>createRoom:onSuccess:onError:</code>	Create a
<code>destroyRoom:onError:</code>	Dismiss t room.
<code>enterRoom:onSuccess:onError:</code>	Enter a rc
<code>enterRoom:roomType:onSuccess:onError:</code>	Enter a rc
<code>enterRoom:roomType:options:onSuccess:onError:</code>	Enter a rc
<code>exitRoom:onSuccess:onError:</code>	Exit the rc
<code>fetchRoomInfo:onError:</code>	Fetch roo informati
<code>fetchRoomInfo:roomType:onSuccess:onError:</code>	Fetch roo informati
<code>updateRoomNameByAdmin:onSuccess:onError:</code>	Update rc name (on support fo administr or room o

<code>updateRoomSeatModeByAdmin:onSuccess:onError:</code>	Update room seat mode support for administrator or room owner.
<code>updateRoomPasswordByAdmin:onSuccess:onError:</code>	Update room password.
<code>getRoomMetadata:onSuccess:onError:</code>	Get room metadata.
<code>setRoomMetadataByAdmin</code>	Set room metadata key already exists, update value, if not the key.
<code>setLocalVideoView:</code>	Set the local camera to preview the render video.
<code>openLocalCamera:quality:onSuccess:onError:</code>	Open the camera.
<code>closeLocalCamera</code>	Close the camera.
<code>startPushLocalVideo</code>	Start publishing local video stream, disabled.
<code>stopPushLocalVideo</code>	Stop publishing local video stream.
<code>updateVideoQuality:</code>	Update video encoding quality.
<code>updateVideoQualityEx:params:</code>	Set the video encoding parameters.

setVideoResolutionMode:resolutionMode:	Set the video resolution (horizontal resolution, vertical resolution)
enableGravitySensor:	Turn on gravity sensing (only available on mobile OSes; the camera capture is inside the application)
startScreenCaptureByReplaykit:	Start screen sharing (only available on mobile OSes)
startScreenCapture:onSuccess:onError:	Start screen sharing (only available on Mac OS)
stopScreenCapture	Stop screen sharing.
getScreenCaptureSources	Get the list of screen areas/windows available on Mac OS)
selectScreenCaptureTarget:	Select the screen or windows to share (only available on Mac OS)
openLocalMicrophone:onSuccess:onError:	Open local microphone
closeLocalMicrophone	Close the microphone

<code>updateAudioQuality:</code>	Update a encoding quality.
<code>muteLocalAudio</code>	Pause publishing local audio stream.
<code>unmuteLocalAudio:onError:</code>	Resume publishing local audio stream.
<code>enableSystemAudioSharing:</code>	Enable system audio sharing.
<code>setRemoteVideoView:streamType:view:</code>	Set the remote video view for the user.
<code>startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:</code>	Start playing remote user video stream.
<code>stopPlayRemoteVideo:streamType:</code>	Stop playing remote user video stream.
<code>muteRemoteAudioStream:isMute:</code>	Mute the user's audio stream.
<code>getUserList:onSuccess:onError:</code>	Get the list of users in the room.
<code>getUserInfo:onSuccess:onError:</code>	Get user information.
<code>changeUserRole:role:onSuccess:onError:</code>	Change user role (only support for administrator or room owner).
<code>changeUserNameCard:nameCard:onSuccess:onError:</code>	Change user nickname.

	room (only support Tencent Cloud Real-Time Communication). You can change a room for an administrator or room owner. A user can change a room only if the user is an administrator or room owner.
<code>kickRemoteUserOutOfRoom:onSuccess:onError:</code>	Kick the remote user out of the room (only support Tencent Cloud Real-Time Communication). An administrator or room owner can kick a user out of a room.
<code>addCategoryTagForUsers:userList:onSuccess:onError:</code>	Add a category tag for a user (only support Tencent Cloud Real-Time Communication). An administrator or room owner can add a category tag for a user.
<code>removeCategoryTagForUsers:userList:onSuccess:onError:</code>	Remove a category tag for a user (only support Tencent Cloud Real-Time Communication). An administrator or room owner can remove a category tag for a user.
<code>getUserListByTag:nextSequence:onSuccess:onError:</code>	Get user information in the room on the tag.
<code>setCustomInfoForUser:onSuccess:onError:</code>	Set custom information for a room user.
<code>disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:</code>	The room owner or administrator can control the users who can open a device. For example, users are prohibited from opening a device.

	micropho available conferenc scenario)
<code>openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Request t remote us open the device (o support fr administr or room o
<code>closeRemoteDeviceByAdmin:device:onSuccess:onError:</code>	Close ren user med devices (o support fr administr or room o
<code>applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Apply to c the local r device (available general u
<code>getSeatList:onError:</code>	Get seat l
<code>lockSeatByAdmin:lockMode:onSuccess:onError:</code>	Lock the : (only sup administr or room o
<code>takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Take the
<code>leaveSeat:onError:</code>	Leave the
<code>moveToSeat:onSuccess:onError:</code>	Move to s
<code>takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:</code>	Invite use take the s (only sup administr or room o
<code>kickUserOffSeatByAdmin:userId:onSuccess:onError:</code>	Kick off th from seat

	support for administrators or room owners
<code>getSeatApplicationList:onError:</code>	Get the re- list of users want to take seat in the (only support administrators or room owners)
<code>disableSendingMessageByAdmin:isDisable:onSuccess:onError:</code>	Disable the ability of room users to send message: support for administrators or room owners
<code>disableSendingMessageForAllUser:onSuccess:onError:</code>	Disable the ability of room users to send message: support for administrators or room owners
<code>cancelRequest:onSuccess:onError:</code>	Cancel request.
<code>responseRemoteRequest:agree:onSuccess:onError:</code>	Response to request.
<code>getTRTCCloud</code>	Get the TRT instance ID
<code>setBeautyLevel:beautyLevel:</code>	Set the beauty level.
<code>setWhitenessLevel:</code>	Set white level.
<code>getExtension:</code>	Get the extension ID
<code>getMediaDeviceManager</code>	Get device manager

	class.
<code>getLiveConnectionManager</code>	Get live-connectic managen class.
<code>getLiveBattleManager</code>	Get live-b managen class.
<code>callExperimentalAPI:</code>	Call experime APIs.

destroySharedInstance

destroySharedInstance

Destroy the `TUIRoomEngine` instance (singleton mode)

Note

The function supports the `TUIRoomTypeConference` and `TUIRoomTypeLive` room types. Use `destroySharedInstance` to release the singleton object.

loginWithSDKAppld:userId:userSig:onSuccess:onError:

loginWithSDKAppld:userId:userSig:onSuccess:onError:

+ (void)loginWithSDKAppld:	(NSInteger)sdkAppld
userId:	(NSString *)userId
userSig:	(NSString *)userSig
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

After creating a `TUIRoomEngine` instance, you should login with `sdkAppld`, `userId` and `userSig` then you can call `TUIRoomEngine` instance and other function.

Param	DESC
-------	------

sdkAppId	It is Application ID. You can see the SDKAppId by creating an application in the TRTC Console .
userId	User ID, it is the unique identifier used by Tencent Cloud to distinguish users.
userSig	The user signature designed by Tencent Cloud based on the UserId, which is used to access Tencent Cloud services. More details, see UserSig

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

If a user is kicked off while online, the SDK will notify you through the `$onKickedOffLine$` callback in `$TUIRoomObserver$`.

logout:onError:

logout:onError:

+ (void)logout:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Log out of your account. If you are in the room, there will be active leaving room and destroying resource operations.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

setSelfInfoWithUserName:avatarUrl:onSuccess:onError:

setSelfInfoWithUserName:avatarUrl:onSuccess:onError:

+ (void)setSelfInfoWithUserName:	(NSString *)userName
avatarUrl:	(NSString *)avatarURL
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Update user name and avatar for logged-in user.

Param	DESC
-------	------

avatarURL	User avatar URL.
userName	User name.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getSelfInfo

getSelfInfo

Return the basic information of the logged-in user, including nickname and avatar.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Return Desc:

[TUILoginUserInfo](#) User login information.

setSelfInfo:onSuccess:onError:

setSelfInfo:onSuccess:onError:

+ (void)setSelfInfo:	(TUILoginUserInfo *)userInfo
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Update user basic information for logged-in user.

Param	DESC
userInfo	Local user information.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

addObserver:

addObserver:

- (void)addObserver:	(id<TUIRoomObserver>)observer
----------------------	-------------------------------

Set event observer.

Param	DESC
observer	Listening instance.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
You can use TUIRoomObserver to receive roomEngine events. More details, see [TUIRoomObserver](#).

removeObserver:

removeObserver:

- (void)removeObserver:	(id<TUIRoomObserver>)observer
-------------------------	-------------------------------

Remove event observer.

Param	DESC
observer	The event observer to be removed.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

createRoom:onSuccess:onError:

createRoom:onSuccess:onError:

- (void)createRoom:	(TUIRoomInfo *)roomInfo
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Create a room.

--	--

Param	DESC
roomInfo	Room information. More details, see TUIRoomInfo .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

You can create a conference room or live room through this API.

destroyRoom:onError:

destroyRoom:onError:

- (void)destroyRoom:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Dismiss the room.**Note**

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the room was dismissed, the SDK notifies the users in the room through the [onRoomDismissed](#) callback in `$TUIRoomObserver$`.

enterRoom:onSuccess:onError:

enterRoom:onSuccess:onError:

- (void)enterRoom:	(NSString *)roomId
onSuccess:	(TUIRoomInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Enter a room.

Param	DESC
roomId	Room ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entered the room, the SDK will notify the user in the room through the `$onRemoteUserEnterRoom$` callback in `$TUIRoomObserver$`.

enterRoom:roomType:onSuccess:onError:

enterRoom:roomType:onSuccess:onError:

- (void)enterRoom:	(NSString *)roomId
roomType:	(TUIRoomType)roomType
onSuccess:	(TUIRoomInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Enter a room.

Param	DESC
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entered the room, the SDK will notify the user in the room through the `$onRemoteUserEnterRoom$` callback in `$TUIRoomObserver$`.

enterRoom:roomType:options:onSuccess:onError:

enterRoom:roomType:options:onSuccess:onError:

--	--

- (void)enterRoom:	(NSString *)roomId
roomType:	(TUIRoomType)roomType
options:	(TUIEnterRoomOptions *)options
onSuccess:	(TUIRoomInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Enter a room.

Param	DESC
options	Roosn options. More details, see TUIEnterRoomOptions .
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

The limit on the number of rooms that a single device can enter at the same time is: 1 for conference and 3 for live . If the limit is exceeded, the device will exit the earliest joined room. When multiple devices log in with the same account, only one device is allowed to enter a conference room with the same ID. When other devices try to enter, the earlier entered device will be kicked out.

After entering the room, the SDK will notify the user in the room through the \$onRemoteUserEnterRoom\$ callback in \$TUIRoomObserver\$.

exitRoom:onSuccess:onError:

exitRoom:onSuccess:onError:

- (void)exitRoom:	(BOOL)syncWaiting
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Exit the room.

Param	DESC

syncWaiting	true: wait for exit request finished, false: exit immediately.
-------------	--

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After left the room, the SDK will notify the user in the room through the [onRemoteUserLeaveRoom](#) callback in `$TUIRoomObserver$`.

fetchRoomInfo:onError:

fetchRoomInfo:onError:

- (void)fetchRoomInfo:	(TUIRoomInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Fetch room information.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

You can get room information through this API.

fetchRoomInfo:roomType:onSuccess:onError:

fetchRoomInfo:roomType:onSuccess:onError:

- (void)fetchRoomInfo:	(NSString*)roomId
roomType:	(TUIRoomType)roomType
onSuccess:	(TUIRoomInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Fetch room information.

Param	DESC
roomId	Room ID.
roomType	Room type. More details, see TUIRoomType .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Return Desc:

Successfully fetched the room information callback, which will include room information of \$TUIRoomInfo.

updateRoomNameByAdmin:onSuccess:onError:

updateRoomNameByAdmin:onSuccess:onError:

- (void)updateRoomNameByAdmin:	(NSString *)roomName
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Update room name (only support for administrators or room owner).

Param	DESC
roomName	Room name.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the room name was updated, the SDK notifies users in the room through the [onRoomNameChanged](#) callback in \$TUIRoomObserver\$.

updateRoomSeatModeByAdmin:onSuccess:onError:

updateRoomSeatModeByAdmin:onSuccess:onError:

- (void)updateRoomSeatModeByAdmin:	(TUISeatMode)seatMode
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Update room seat mode (only support for administrators or room owner).

Param	DESC

seatMode	TUISeatModeFreeToTake : Free to take seat mode, users can take the seat without application; TUISeatModeApplyToTake : Apply to take seat mode, users can only take the seat after the owner or administrator approved.
----------	---

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the room seat mode was updated, the SDK will notify users in the room through the [onRoomSeatModeChanged](#) callback in `$TUIRoomObserver$`.

updateRoomPasswordByAdmin:onSuccess:onError:

updateRoomPasswordByAdmin:onSuccess:onError:

- (void)updateRoomPasswordByAdmin:	(NSString *)password
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Update room password.

Param	DESC
password	Room password.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

getRoomMetadata:onSuccess:onError:

getRoomMetadata:onSuccess:onError:

- (void)getRoomMetadata:	(NSArray<NSString *> *)keys
onSuccess:	(TUIRoomMetadataResponseBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get room metadata.

--	--

Param	DESC
keys	Room metadata key list, if keys are passed as empty, all metadata will be retrieved.

Note

The function only supports the [TUIRoomTypeLive](#) room type.

setRoomMetadataByAdmin

setRoomMetadataByAdmin

Set room metadata, if the key already exists, update its value, if not, add the key.

Param	DESC
metadata	Key-Value in room metadata.

Note

The function only supports the [TUIRoomTypeLive](#) room type.

setLocalVideoView:

setLocalVideoView:

- (void)setLocalVideoView:	(TUIVideoView * __nullable)view
----------------------------	---------------------------------

Set the local camera to preview the render view.

Param	DESC
view	Render view.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

openLocalCamera:quality:onSuccess:onError:

openLocalCamera:quality:onSuccess:onError:

--	--

- (void)openLocalCamera:	(BOOL)isFront
quality:	(TUIVideoQuality)quality
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Open the local camera.

Param	DESC
isFront	YES: front NO: rear (only available on mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After opened the local camera in the room, the local video stream is published by default, and the SDK notifies the users in the room through the `$onUserVideoStateChanged$` callback in `$TUIRoomObserver$`.

closeLocalCamera

closeLocalCamera

Close the local camera.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After closed the local camera in the room, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

startPushLocalVideo

startPushLocalVideo

Start publishing local video stream, default enabled.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After published the local video, if your local camera is opening, the SDK will notify users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

stopPushLocalVideo

stopPushLocalVideo

Stop publishing local video stream.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After stopped published local video, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

updateVideoQuality:

updateVideoQuality:

- (void)updateVideoQuality:	(TUIVideoQuality)quality
-----------------------------	--

Update video encoding quality.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

updateVideoQualityEx:params:

updateVideoQualityEx:params:

- (void)updateVideoQualityEx:	(TUIVideoStreamType)streamType
params:	(TUIRoomVideoEncoderParams *)params

Set the video encoding parameters.

Param	DESC
params	Encoding parameters of the video. More details, see TUIRoomVideoEncoderParams .
streamType	The type of video stream. More details, see TUIVideoStreamType .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

setVideoResolutionMode:resolutionMode:

setVideoResolutionMode:resolutionMode:

- (void)setVideoResolutionMode:	(TUIVideoStreamType)streamType
resolutionMode:	(TUIResolutionMode)resolutionMode

Set the video resolution mode (horizontal resolution or vertical resolution).

Param	DESC
resolutionMode	Resolution mode. More details, see TUIResolutionMode .
streamType	The type of video stream. More details, see TUIVideoStreamType .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

enableGravitySensor:

enableGravitySensor:

- (void)enableGravitySensor:	(BOOL)enable
------------------------------	--------------

Turn on gravity sensing mode. (only availble on mobile OS and the camera capture scene inside the SDK).

Param	DESC
enable	YES: Open NO: Close.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After turning on gravity sensing, if the device on the collection end rotates, the images on the collection end and the audience will be rendered accordingly to ensure that the image in the field of view is always facing up.

startScreenCaptureByReplaykit:

startScreenCaptureByReplaykit:

--	--

- (void)startScreenCaptureByReplaykit:

(NSString *)appGroup

Start screen sharing (only available on mobile OS).

After screen sharing started, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

startScreenCapture:onSuccess:onError:

startScreenCapture:onSuccess:onError:

- (void)startScreenCapture:	(TUIVideoView *)view
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Start screen sharing (only available on Mac OS).

Param	DESC
view	The render view can be set a null value, it means not displaying the preview screen locally.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After screen sharing started, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$`.

The API can capture the screen content of the entire Mac OS, or capture and share the window content you select.

stopScreenCapture

stopScreenCapture

Stop screen sharing.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After screen sharing ended, the SDK notifies users in the room through the [onUserVideoStateChanged](#) callback in `$TUIRoomObserver$` and also notifies you through the [onUserScreenCaptureStopped](#) callback.

getScreenCaptureSources

getScreenCaptureSources

Get the sharable screen and windows (only available on Mac OS)

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

The user can use the API to select the screen and window that can be shared. Through the API, you can query the ID, name, and thumbnail of the window that can be shared in the current system.

Return Desc:

The list of windows, including screens.

selectScreenCaptureTarget:

selectScreenCaptureTarget:

- (void)selectScreenCaptureTarget:	(NSString *)targetId
------------------------------------	----------------------

Select the screen or windows to share (only available on Mac OS)

Param	DESC
targetId	Selected sharing source.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After you obtain the screen and window that can be shared through getScreenCaptureSources, you can call the API to select the target screen or window.

During the screen sharing, you can also call the API to change the sharing target.

openLocalMicrophone:onSuccess:onError:

openLocalMicrophone:onSuccess:onError:

- (void)openLocalMicrophone:	(TUIAudioQuality)quality
onSuccess:	(TUISuccessBlock)onSuccess

onError:	(TUIErrorBlock)onError
----------	------------------------

Open local microphone.

Param	DESC
quality	Audio quality.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
After opened the local microphone in a room, the SDK notifies users in the room through the [onUserAudioStateChanged](#) callback in \$TUIRoomObserver\$.

closeLocalMicrophone

closeLocalMicrophone

Close the local microphone.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
After closed the local microphone in the room, the SDK notifies the users in the room through the [onUserAudioStateChanged](#) callback in \$TUIRoomObserver\$.

updateAudioQuality:

updateAudioQuality:

- (void)updateAudioQuality:	(TUIAudioQuality)quality
-----------------------------	--

Update audio encoding quality.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

muteLocalAudio

muteLocalAudio

Pause publishing the local audio stream.**Note**

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

If you have opened your microphone and call the API to pause publishing the local audio stream, the SDK will notify the users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

unmuteLocalAudio: onError:

unmuteLocalAudio: onError:

- (void)unmuteLocalAudio:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Resume publishing the local audio stream.**Note**

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

If you have opened your microphone and call the API to resume publishing the local audio stream, the SDK will notify the users in the room through the [onUserAudioStateChanged](#) callback in `$TUIRoomObserver$`.

enableSystemAudioSharing:

enableSystemAudioSharing:

- (void)enableSystemAudioSharing:	(BOOL)enable
-----------------------------------	--------------

Enable system audio sharing

This API captures system audio data from your device and mixes it into the current audio stream of the SDK. This ensures that other users in the room hear the audio played back by the another app.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Android: You need to use this interface to enable system sound capture first, and it will take effect only when you call `startScreenCapture` to enable screen sharing.

setRemoteVideoView:streamType:view:

setRemoteVideoView:streamType:view:

- (void)setRemoteVideoView:	(NSString *)userId
streamType:	(TUIVideoStreamType)streamType
view:	(TUIVideoView * __nullable)view

Set the render view for remote user.

Param	DESC
streamType	The type of video stream. More details, see TUIVideoStreamType .
userId	Remote user ID.
view	Render view.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:

startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:

- (void)startPlayRemoteVideo:	(NSString *)userId
streamType:	(TUIVideoStreamType)streamType
onPlaying:	(TUIPlayOnPlayingBlock)onPlaying
onLoading:	(TUIPlayOnLoadingBlock)onLoading
onError:	(TUIPlayOnErrorBlock)onError

Start playing the remote user's video stream.

Param	DESC
streamType	The type of video stream. More details, see TUIVideoStreamType .
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

stopPlayRemoteVideo:streamType:

stopPlayRemoteVideo:streamType:

- (void)stopPlayRemoteVideo:	(NSString *)userId
streamType:	(TUIVideoStreamType)streamType

Stop playing the remote user's video stream.

Param	DESC
streamType	The type of video stream. More details, see TUIVideoStreamType .
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

muteRemoteAudioStream:isMute:

muteRemoteAudioStream:isMute:

- (void)muteRemoteAudioStream:	(NSString *)userId
isMute:	(BOOL)isMute

Mute the remote user's audio stream.

Param	DESC
isMute	true: pause pulling remote user's audio stream, false: resume pulling remote user's audio stream.
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getUserList:onSuccess:onError:

getUserList:onSuccess:onError:

- (void)getUserList:	(NSInteger)nextSequence
onSuccess:	(TUIUserListResponseBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get the list of user in the room.

Param	DESC
nextSequence	Filling in 0 for the first request, if the returned data of the callback is not zero, paging is required, continue the operation until it is 0.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getUserInfo:onSuccess:onError:**getUserInfo:onSuccess:onError:**

- (void)getUserInfo:	(NSString *)userId
onSuccess:	(TUIUserInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get user information.

Param	DESC
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

changeUserRole:role:onSuccess:onError:**changeUserRole:role:onSuccess:onError:**

- (void)changeUserRole:	(NSString *)userId
-------------------------	--------------------

role:	(TUIRole)role
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Change user role (only support for administrators or room owner).

Param	DESC
role	User role. More details, see TUIRole .
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the user role changed, the SDK will notify the users in the room through the [onUserInfoChanged](#) callback in `$TUIRoomObserver$`.

changeUserNameCard:nameCard:onSuccess:onError:

changeUserNameCard:nameCard:onSuccess:onError:

- (void)changeUserNameCard:	(NSString *)userId
nameCard:	(NSString *)nameCard
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Change user nickname in the room (only support to change all user for administrators or room owner, user can only change by self).

Param	DESC
nameCard	User nickname to set, maximum support is 32 bytes
userId	User ID to change.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After the user nickname changed, the SDK will notify the users in the room through the [onUserInfoChanged](#) callback in `$TUIRoomObserver$`.

kickRemoteUserOutOfRoom:onSuccess:onError:

kickRemoteUserOutOfRoom:onSuccess:onError:

- (void)kickRemoteUserOutOfRoom:	(NSString *)userId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Kick the remote user out of the room (only support for administrators or room owner).

Param	DESC
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the remote user is kicked off from the room, the SDK notifies the kicked user through the [onKickedOutOfRoom](#) callback in `$TUIRoomObserver$` and notifies users in the room through [onRemoteUserLeaveRoom](#).

addCategoryTagForUsers:userList:onSuccess:onError:

addCategoryTagForUsers:userList:onSuccess:onError:

- (void)addCategoryTagForUsers:	(NSInteger)tag
userList:	(NSArray<NSString *> *)userList
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Add a tag for the user (only support for administrators or room owner).

Param	DESC
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.

userList	User list.
----------	------------

Note

The function only supports the [TUIRoomTypeConference](#) room type.

removeCategoryTagForUsers:userList:onSuccess:onError:

removeCategoryTagForUsers:userList:onSuccess:onError:

- (void)removeCategoryTagForUsers:	(NSInteger)tag
userList:	(NSArray<NSString *>)userList
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Remove tag for user (only support for room owner).

Param	DESC
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.
userList	User list.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

getUserListByTag:nextSequence:onSuccess:onError:

getUserListByTag:nextSequence:onSuccess:onError:

- (void)getUserListByTag:	(NSInteger)tag
nextSequence:	(NSInteger)nextSequence
onSuccess:	(TUIUserListResponseBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get user information in the room based on the tag.

Param	DESC
nextSequence	Filling in 0 for the first request, if the returned data of the callback is not zero, paging is required, continue the operation until it is 0.
tag	Integer type, it is recommended that this value must be greater than or equal to 1000, you can customize it.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

setCustomInfoForUser:onSuccess:onError:

setCustomInfoForUser:onSuccess:onError:

- (void)setCustomInfoForUser:	(NSString *)userId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Set custom information for room users.

Param	DESC
customInfo	Custom information.
userId	User userId.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:

disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:

- (void)disableDeviceForAllUserByAdmin:	(TUIMediaDevice)device
isDisable:	(BOOL)isDisable

onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

The owner or administrator control that all users whether can open device. For example: all users are prohibited from opening the microphone(only available in the conference scenario).

Param	DESC
device	Device. More details, see: TUIMediaDevice .
isDisable	true: disable user to open device, false: enable user to open device.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After the API call is successful:

If the device type is MICROPHONE , the SDK will notify the users in the room through

[onAllUserMicrophoneDisableChanged](#) in \$TUIRoomObserver\$.

If the device type is CAMERA , the SDK will notify the users in the room through [onAllUserCameraDisableChanged](#) in

\$TUIRoomObserver\$.

If the device type is SCREEN_SHARING , the SDK will notify the users in the room through

[onScreenShareForAllUserDisableChanged](#) in \$TUIRoomObserver\$.

openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

- (TUIRequest *)openRemoteDeviceByAdmin:	(NSString *)userId
device:	(TUIMediaDevice)device
timeout:	(NSTimeInterval)timeout
onAccepted:	(nullable TUIRequestAcceptedBlock)onAccepted
onRejected:	(nullable TUIRequestRejectedBlock)onRejected
onCancelled:	(nullable TUIRequestCancelledBlock)onCancelled
onTimeout:	(nullable TUIRequestTimeoutBlock)onTimeout
onError:	(nullable TUIRequestErrorBlock)onError

Request the remote user to open the media device (only support for administrators or room owner).

Param	DESC
device	Media device. More details, see: TUIMediaDevice .
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.
userId	User ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After calling the API, the SDK will notify the requested user through [onRequestReceived](#) in `$TUIRoomObserver$`.

Return Desc:

TUIRequest Request body.

closeRemoteDeviceByAdmin:device:onSuccess:onError:

closeRemoteDeviceByAdmin:device:onSuccess:onError:

- (void)closeRemoteDeviceByAdmin:	(NSString *)userId
device:	(TUIMediaDevice)device
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Close remote user media devices (only support for administrators or room owner).

Param	DESC
device	Media device. More details, see: TUIMediaDevice .
userId	User ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After the API call is successful:

If the device type is MICROPHONE, the SDK will notify the users in the room through [onUserAudioStateChanged](#) in `$TUIRoomObserver$`.

If the device type is CAMERA or SCREEN_SHARING, the SDK will notify the users in the room through [onUserVideoStateChanged](#) in `$TUIRoomObserver$`.

applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

- (TUIRequest *)applyToAdminToOpenLocalDevice:	(TUIMediaDevice)device
timeout:	(NSTimeInterval)timeout
onAccepted:	(nullable TUIRequestAcceptedBlock)onAccepted
onRejected:	(nullable TUIRequestRejectedBlock)onRejected
onCancelled:	(nullable TUIRequestCancelledBlock)onCancelled
onTimeout:	(nullable TUIRequestTimeoutBlock)onTimeout
onError:	(nullable TUIRequestErrorBlock)onError

Apply to open the local media device (available to general users).

Param	DESC
device	Media device. More details, see: TUIMediaDevice .
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After the API call is successful, the SDK will notify the requested user through [onRequestReceived](#) in `$TUIRoomObserver$`.

Return Desc:

TUIRequest: Request body.

getSeatList:onError:

getSeatList:onError:

- (void)getSeatList:	(TUISeatListResponseBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get seat list.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

lockSeatByAdmin:lockMode:onSuccess:onError:

lockSeatByAdmin:lockMode:onSuccess:onError:

- (void)lockSeatByAdmin:	(NSInteger)seatIndex
lockMode:	(TUISeatLockParams *)lockParams
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Lock the seat (only support for administrators or room owner).

Param	DESC
lockParams	Seat lock parameter. More details, see: \$TUISeatLockParam\$.
seatIndex	Seat index.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

If the lockParams is lockSeat, it means that the current seat can not be taken by all users and the user will be kicked off if the seat was taken.

If the lockParams is lockVideo/lockAudio, it means that the current seat can not publish video/audio stream.

takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

- (TUIRequest *)takeSeat:	(NSInteger)seatIndex
timeout:	(NSTimeInterval)timeout
onAccepted:	(TUIRequestAcceptedBlock)onAccepted
onRejected:	(TUIRequestRejectedBlock)onRejected
onCancelled:	(TUIRequestCancelledBlock)onCancelled
onTimeout:	(TUIRequestTimeoutBlock)onTimeout
onError:	(TUIRequestErrorBlock)onError

Take the seat.

Param	DESC
seatIndex	Seat index. If the seat is not enabled or the sequence of seats is not concerned, just fill in -1.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

The user can publish audio/video stream after taking the seat when `isSeatEnable` is true.

After taking the seat successfully, the SDK will notify the users in the room through `$onSeatListChanged$` in `$TUIRoomObserver$`.

When the [TUISeatMode](#) is `ApplyToTake`, the operation to take seat need approval from the owner or administrator.

When the [TUISeatMode](#) is `FreeToTake`, you can take seat freely.

Return Desc:

TUIRequest Request body.

leaveSeat:onError:**leaveSeat:onError:**

- (void)leaveSeat:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Leave the seat.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

The user can not publish audio/video stream after leaving the seat when `isSeatEnable` is true.

After leaving seat successfully, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

moveToSeat:onSuccess:onError:

moveToSeat:onSuccess:onError:

- (void)moveToSeat:	(NSInteger)targetSeatIndex
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Move to seat.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After moving seat successfully, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

- (TUIRequest *)takeUserOnSeatByAdmin:	(NSInteger)seatIndex
userId:	(NSString *)userId
timeout:	(NSTimeInterval)timeout
onAccepted:	(TUIRequestAcceptedBlock)onAccepted
onRejected:	(TUIRequestRejectedBlock)onRejected
onCancelled:	(TUIRequestCancelledBlock)onCancelled
onTimeout:	(TUIRequestTimeoutBlock)onTimeout

onError:	(TUIRequestErrorBlock)onError
----------	-------------------------------

Invite user to take the seat (only support for administrators or room owner).

Param	DESC
seatIndex	Seat index.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the API call is successful, the SDK will notify the invited user through [onRequestReceived](#) in `$TUIRoomObserver$`.

Return Desc:

TUIRequest: Request body.

kickUserOffSeatByAdmin:userId:onSuccess:onError:

kickUserOffSeatByAdmin:userId:onSuccess:onError:

- (void)kickUserOffSeatByAdmin:	(NSInteger)seatIndex
userId:	(NSString *)userId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Kick off the user from seat (only support for administrators or room owner).

Param	DESC
seatIndex	Seat index. If the seat is not enabled and the sequence of seats is not concerned, just fill in -1.
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After the API call is successful, the SDK will notify the users in the room through [onSeatListChanged](#) in `$TUIRoomObserver$`.

getSeatApplicationList:onError:

getSeatApplicationList:onError:

- (void)getSeatApplicationList:	(TUIRequestListResponseBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get the request list of users who want to take the seat in the room (only support for administrators or room owner).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

disableSendingMessageByAdmin:isDisable:onSuccess:onError:

disableSendingMessageByAdmin:isDisable:onSuccess:onError:

- (void)disableSendingMessageByAdmin:	(NSString *)userId
isDisable:	(BOOL)isDisable
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Disable the ability of remote users to send messages (only support for administrators or room owner).

Param	DESC
isDisable	true: disable user to send message, false: enable user to send message.
userId	User ID.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After disabling the ability of remote users to send messages, the SDK notifies the disabled user through [onSendMessageForUserDisableChanged](#) in `$TUIRoomObserver$`.

disableSendingMessageForAllUser:onSuccess:onError:

disableSendingMessageForAllUser:onSuccess:onError:

- (void)disableSendingMessageForAllUser:	(BOOL)isDisable
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Disable the ability of all users to send messages (only support for administrators or room owner).

Param	DESC
isDisable	true: disable all users to send message, false: enable all users to send message.

Note

The function only supports the [TUIRoomTypeConference](#) room type.

After disabling the ability of all users to send messages, the SDK notifies users in the room through [onSendMessageForAllUserDisableChanged](#) in `$TUIRoomObserver$`.

cancelRequest:onSuccess:onError:

cancelRequest:onSuccess:onError:

- (void)cancelRequest:	(NSString *)requestId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Cancel request.

Param	DESC
requestId	Request ID (get from the sent request).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After cancelling a request, the SDK will notify the requested user through [onRequestCancelled](#) in `$TUIRoomObserver$`.

The API can be used to cancel a request that has been sent.

responseRemoteRequest:agree:onSuccess:onError:

responseRemoteRequest:agree:onSuccess:onError:

- (void)responseRemoteRequest:	(NSString *)requestId
agree:	(BOOL)agree
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Response request.

Param	DESC
agree	YES: Agree the request, NO: Reject the request.
requestId	Request ID (get from the sent request or notification of the OnRequestReceived event).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

When received a request, you can use this API to reply the received request.

getTRTCCloud

getTRTCCloud

Get the TRTC instance object.

setBeautyLevel:beautyLevel:

setBeautyLevel:beautyLevel:

- (void)setBeautyLevel:	(NSInteger)beautyStyle
-------------------------	------------------------

beautyLevel:	(float)beautyLevel
--------------	--------------------

Set the beauty level.

Param	DESC
beautyLevel	Beauty level, the value range is 0 - 9; 0 indicates to disable the filter, and 9 indicates the most obvious effect.
beautyStyle	Beauty style: TXBeautyStyleSmooth: Smooth; TXBeautyStyleNature: Natural; TXBeautyStylePitu: Pitu style.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

setWhitenessLevel:

setWhitenessLevel:

- (void)setWhitenessLevel:	(float)whitenessLevel
----------------------------	-----------------------

Set whitening level.

Param	DESC
whitenessLevel	Whitening level, ranging from 0 - 9; 0 indicates to disable the filter, and 9 indicates the most obvious effect.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getExtension:

getExtension:

- (id) getExtension:	(TUIExtensionType)extensionType
----------------------	---

Get the extension.

Param	DESC

extensionType	Extension type. More deatils, see TUIExtensionType .
---------------	--

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getMediaDeviceManager

getMediaDeviceManager

Get device management class.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getLiveConnectionManager

getLiveConnectionManager

Get live-connection management class.

Note

The function supports the [TUIRoomTypeLive](#) room type.

getLiveBattleManager

getLiveBattleManager

Get live-battle management class.

Note

The function supports the [TUIRoomTypeLive](#) room type.

callExperimentalAPI:

callExperimentalAPI:

+ (id)callExperimentalAPI:	(NSString *)jsonStr
----------------------------	---------------------

Call experimental APIs.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

TUIRoomObserver

Last updated : 2025-02-13 16:38:49

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUIRoomObserver @ TUIKitEngine.

Function: TUIRoomEngine event callback APIs.

TUIRoomObserver

TUIRoomObserver

FuncList	DESC
onError:message:	Error event callback.
onKickedOffLine:	The current user was kicked offline.
onUserSigExpired	The current user signature is expired.
onRoomNameChanged:roomName:	The name of the room has changed.
onAllUserMicrophoneDisableChanged:isDisable:	The status of disabling to open microphone has changed for all users.
onAllUserCameraDisableChanged:isDisable:	The status of disabling to open camera has changed for all users.
onScreenShareForAllUserDisableChanged:isDisable:	The status of disabling to open screen sharing has changed for all users.
onSendMessageForAllUserDisableChanged:isDisable:	The status of disabling to send message has changed for all users.
onRoomDismissed:reason:	Room was dismissed.
onKickedOutOfRoom:reason:message:	The current user has been kicked off from the room.
onRoomSeatModeChanged:seatMode:	The room seat mode has changed.
onRoomUserCountChanged:userCount:	The count of user in the room has changed.
onRoomMetadataChanged:value:	The key-value of room metadata has

	changed.
<code>onRemoteUserEnterRoom:userInfo:</code>	Remote user entered room.
<code>onRemoteUserLeaveRoom:userInfo:</code>	Remote user left room.
<code>onUserInfoChanged:modifyFlag:</code>	User information has changed in the room.
<code>onUserVideoStateChanged:streamType:hasVideo:reason:</code>	The status of the user has video stream changed.
<code>onUserAudioStateChanged:hasAudio:reason:</code>	The status of the user has audio stream changed.
<code>onUserVoiceVolumeChanged</code>	User volume changed.
<code>onSendMessageForUserDisableChanged:userId:isDisable:</code>	The status of disabling to send message has changed for user.
<code>onUserNetworkQualityChanged:</code>	The user network status changed.
<code>onUserScreenCaptureStopped:</code>	Screen sharing stopped.
<code>onSeatListChanged:seated:left:</code>	Seat list changed.
<code>onKickedOffSeat:operateUser:</code>	The user was kicked off the seat.
<code>onRequestReceived:</code>	Receive a request message.
<code>onRequestCancelled:operateUser:</code>	Received a cancelled request.
<code>onRequestProcessed:operateUser:</code>	Receive a request to be processed by other administrator/owner.
<code>onDeviceChanged:type:state:</code>	Local device added.

onError:message:

onError:message:

- (void)onError:	(TUIError)errorCode
message:	(NSString *)message

Error event callback.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Callback error events when entering a room or opening a device.

Param	DESC
errorCode	Error code. More details, see: TUIError .
message	Error message.

onKickedOffLine:

onKickedOffLine:

- (void)onKickedOffLine:	(NSString *)message
--------------------------	---------------------

The current user was kicked offline.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user is kicked offline.

Param	DESC
message	Description of being kicked off.

onUserSigExpired

onUserSigExpired

The current user signature is expired.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's signature expires.

onRoomNameChanged:roomName:

onRoomNameChanged:roomName:

- (void)onRoomNameChanged:	(NSString *)roomId
roomName:	(NSString *)roomName

The name of the room has changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when the room name changes.

Param	DESC
roomId	Room ID.
roomName	Room name.

onAllUserMicrophoneDisableChanged:isDisable:

onAllUserMicrophoneDisableChanged:isDisable:

- (void)onAllUserMicrophoneDisableChanged:	(NSString *)roomId
isDisable:	(BOOL)isDisable

The status of disabling to open microphone has changed for all users.

Param	DESC
isDisable	true: disable user to open microphone false: enable user to open microphone.
roomId	Room ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type..

Called when the microphone disable status changes for all users.

onAllUserCameraDisableChanged:isDisable:

onAllUserCameraDisableChanged:isDisable:

- (void)onAllUserCameraDisableChanged:	(NSString *)roomId
isDisable:	(BOOL)isDisable

The status of disabling to open camera has changed for all users.

Param	DESC
isDisable	true: disable user to open camera false: enable user to open camera.

roomId	Room ID.
--------	----------

Note

The function only supports the [TUIRoomTypeConference](#) room type..

Called when the camera disable status changes for all users.

onScreenShareForAllUserDisableChanged:isDisable:

onScreenShareForAllUserDisableChanged:isDisable:

- (void)onScreenShareForAllUserDisableChanged:	(NSString *)roomId
isDisable:	(BOOL)isDisable

The status of disabling to open screen sharing has changed for all users.

Param	DESC
isDisable	true: disable user to open screen sharing false: enable user to open screen sharing.
roomId	Room ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type..

Called when the screen sharing permissions change for all users.

onSendMessageForAllUserDisableChanged:isDisable:

onSendMessageForAllUserDisableChanged:isDisable:

- (void)onSendMessageForAllUserDisableChanged:	(NSString *)roomId
isDisable:	(BOOL)isDisable

The status of disabling to send message has changed for all users.

Param	DESC
isDisable	true: disable user to send message false: enable user to send message.
roomId	Room ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type..
Called when the message sending permissions change for all users.

onRoomDismissed:reason:

onRoomDismissed:reason:

- (void)onRoomDismissed:	(NSString *)roomId
reason:	(TUIRoomDismissedReason)reason

Room was dismissed.

Param	DESC
reason	The reason why the room was dismissed. More details, see: TUIRoomDismissedReason .
roomId	Room ID.

Note

The function only supports the [TUIRoomTypeConference](#) room type..
Called when the room is dismissed.

onKickedOutOfRoom:reason:message:

onKickedOutOfRoom:reason:message:

- (void)onKickedOutOfRoom:	(NSString *)roomId
reason:	(TUIKickedOutOfRoomReason)reason
message:	(NSString *)message

The current user has been kicked off from the room.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
Called when a user is kicked out of the room.

Param	DESC
message	Description of being kicked off.

reason	Reason for being kicked off.
roomId	Room ID.

onRoomSeatModeChanged:seatMode:

onRoomSeatModeChanged:seatMode:

- (void)onRoomSeatModeChanged:	(NSString *)roomId
seatMode:	(TUISeatMode)seatMode

The room seat mode has changed.

Param	DESC
roomId	: Room ID.
seatMode	: Seat mode. More details, see TUISeatMode . #### [REMARK] The function supports the TUIRoomTypeConference and TUIRoomTypeLive room types. Called when the seat mode of the room changes.

onRoomUserCountChanged:userCount:

onRoomUserCountChanged:userCount:

- (void)onRoomUserCountChanged:	(NSString *)roomId
userCount:	(NSInteger)userCount

The count of user in the room has changed.

Param	DESC
roomId	Room ID.
userCount	Count of user. #### [REMARK] The function supports the TUIRoomTypeConference and TUIRoomTypeLive room types. Called when the user count of the room changes.

onRoomMetadataChanged:value:

onRoomMetadataChanged:value:

- (void)onRoomMetadataChanged:	(NSString *)key
value:	(NSString *)value

The key-value of room metadata has changed.

Param	DESC
key	The key of room metadata.
value	The value of room metadata. #### [REMARK] The function supports the TUIRoomTypeConference and TUIRoomTypeLive room types. Called when the custom information of the room changes.

onRemoteUserEnterRoom:userInfo:

onRemoteUserEnterRoom:userInfo:

- (void)onRemoteUserEnterRoom:	(NSString *)roomId
userInfo:	(TUIUserInfo *)userInfo

Remote user entered room.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a remote user enters the room.

Param	DESC
roomId	Room ID.
userInfo	User information. More details, see TUIUserInfo .

onRemoteUserLeaveRoom:userInfo:

onRemoteUserLeaveRoom:userInfo:

- (void)onRemoteUserLeaveRoom:	(NSString *)roomId
--------------------------------	--------------------

userInfo:	(TUIUserInfo *)userInfo
-----------	--

Remote user left room.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a remote user leaves the room.

Param	DESC
roomId	Room ID.
userInfo	User information. More details, see TUIUserInfo .

onUserInfoChanged:modifyFlag:

onUserInfoChanged:modifyFlag:

- (void)onUserInfoChanged:	(TUIUserInfo *)userInfo
modifyFlag:	(TUIUserInfoModifyFlag)modifyFlag

User information has changed in the room.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's information changes.

Param	DESC
modifyFlag	Modifiable parameter. More details, see TUIUserInfoModifyFlag .
userInfo	User information. More details, see TUIUserInfo .

onUserVideoStateChanged:streamType:hasVideo:reason:

onUserVideoStateChanged:streamType:hasVideo:reason:

- (void)onUserVideoStateChanged:	(NSString *)userId
streamType:	(TUIVideoStreamType)streamType
hasVideo:	(BOOL)hasVideo
reason:	(TUIChangeReason)reason

The status of the user has video stream changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's video state changes.

Param	DESC
hasVideo	The current user whether has video stream.
reason	The reason why the video stream changed: TUIChangeReasonBySelf : Changed by self TUIChangeReasonByAdmin : Changed by administrator.
streamType	Video stream type. More details, see TUIVideoStreamType .
userId	User ID.

onUserAudioStateChanged:hasAudio:reason:

onUserAudioStateChanged:hasAudio:reason:

- (void)onUserAudioStateChanged:	(NSString *)userId
hasAudio:	(BOOL)hasAudio
reason:	(TUIChangeReason)reason

The status of the user has audio stream changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's audio state changes.

Param	DESC
hasAudio	The current user whether has audio stream.
reason	The reason why the video stream changed: TUIChangeReasonBySelf : Changed by self TUIChangeReasonByAdmin : Changed by administrator.
userId	User ID.

onUserVoiceVolumeChanged

onUserVoiceVolumeChanged

User volume changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's voice volume changes.

Param	DESC
volumeMap	: User volume dictionary key: userId, value: the volume of all speaking users, with a value range of 0 - 100.

onSendMessageForUserDisableChanged:userId:isDisable:

onSendMessageForUserDisableChanged:userId:isDisable:

- (void)onSendMessageForUserDisableChanged:	(NSString *)roomId
userId:	(NSString *)userId
isDisable:	(BOOL)muted

The status of disabling to send message has changed for user.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's message sending permissions change.

Param	DESC
isDisable	true: disable user to send message false: enable user to send message.
userId	User ID.

onUserNetworkQualityChanged:

onUserNetworkQualityChanged:

- (void)onUserNetworkQualityChanged:	(NSArray< TUINetworkInfo *> *)networkList
--------------------------------------	---

The user network status changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's network quality changes.

Param	DESC
networkList	User network status list. More details, see TUINetworkInfo .

onUserScreenCaptureStopped:

onUserScreenCaptureStopped:

- (void)onUserScreenCaptureStopped:	(NSInteger)reason
-------------------------------------	-------------------

Screen sharing stopped.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user's screen capture stops.

Param	DESC
reason	Stop reason, 0: user actively stops; 1: the screen or the window is closed ; 2: the status of the screen or the window has changed (such as device disconnect).

onSeatListChanged:seated:left:

onSeatListChanged:seated:left:

- (void)onSeatListChanged:	(NSArray< TUISeatInfo *> *)seatList
seated:	(NSArray< TUISeatInfo *> *)seatedList
left:	(NSArray< TUISeatInfo *> *)leftList

Seat list changed.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when the seat list changes.

Param	DESC
leftList	List of newly leave-seat users.
seatList	The latest user list on seat, including new users.
seatedList	List of newly take-seat users.

onKickedOffSeat:operateUser:

onKickedOffSeat:operateUser:

- (void)onKickedOffSeat:	(NSInteger)seatIndex
--------------------------	----------------------

operateUser:	(TUIUserInfo *)operateUser
--------------	---

The user was kicked off the seat.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a user is kicked off a seat.

Param	DESC
operateUser	User information of the owner/administrator who kicked the user.
seatIndex	Seat index.

onRequestReceived:

onRequestReceived:

- (void)onRequestReceived:	(TUIRequest *)request
----------------------------	--

Receive a request message.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a request is received.

Param	DESC
request	Request content. More details, see TUIRequest .

onRequestCancelled:operateUser:

onRequestCancelled:operateUser:

- (void)onRequestCancelled:	(TUIRequest *)request
operateUser:	(TUIUserInfo *)operateUser

Received a cancelled request.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a request is cancelled.

Param	DESC

operateUser	Operator information.
request	Request content. More details, see TUIRequest .

onRequestProcessed:operateUser:

onRequestProcessed:operateUser:

- (void)onRequestProcessed:	(TUIRequest *)request
operateUser:	(TUIUserInfo *)operateUser

Receive a request to be processed by other administrator/owner.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a request is processed.

Param	DESC
operateUser	Operator information.
request	Request content. More details, see TUIRequest .

onDeviceChanged:type:state:

onDeviceChanged:type:state:

- (void)onDeviceChanged:	(NSString *)deviceId
type:	(TUIMediaDeviceType)type
state:	(TUIMediaDeviceState)state

Local device added.

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

Called when a device changes.

@deprecated It is not recommended to use it since version v2.0. It is recommended to use [onDeviceChanged](#) in `{TUIRoomDeviceManager$}` instead.

Param	DESC
deviceId	Device ID.

state	0: the device has been added; 1: the device has been removed; 2: the device has been enabled.
type	Device type. More details, see TUIMediaDeviceType .

Note

When a local device (including camera, microphone, and speaker) is added, the SDK will throw this event callback.

TUIRoomDeviceManager

Last updated : 2025-02-13 16:38:49

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUIRoomDeviceManager @ TUIKitEngine.

Function: Device testing and management APIs.

TUIRoomDeviceManager

TUIRoomDeviceManagerObserver

FuncList	DESC
onDeviceChanged:type:state:	The status of the local device changed (only available for desktop OS).
onTestCameraVideoFrameRendered:height:	Test camera video rendered success callback.
onTestMicVolume:	Test microphone volume callback.
onTestSpeakerVolume:	Test speakerphone volume callback.

TUIRoomDeviceManager

FuncList	DESC
isFrontCamera	Query whether the front camera is being used (only available for mobile OS).
switchCamera:	Switch to the front/rear camera (only available for mobile OS).
isAutoFocusEnabled	Query whether automatic face detection is supported (only available for mobile OS).
enableCameraAutoFocus:	Enable auto focus (only available for mobile OS).
enableCameraTorch:	Enable/Disable flash, i.e., the torch mode (only available for mobile OS).

setAudioRoute:	Set the audio route (only available for mobile OS).
setObserver:	Set event observer.
startCameraDeviceTest:	Start camera testing (only available for desktop OS).
stopCameraDeviceTest	End camera testing (only available for desktop OS).
startMicDeviceTest:playback:	Start microphone testing (only available for desktop OS).
stopMicDeviceTest	End microphone testing (only available for desktop OS)
startSpeakerDeviceTest:	Start speaker testing (only available for desktop OS).
stopSpeakerDeviceTest	End speaker testing (only available for desktop OS).
getDevicesList:callback:	Get the device list (only available for desktop OS).
setCurrentDevice:deviceId:	Set the device to use (only available for desktop OS).
getCurrentDevice:callback:	Get the device currently in use (only available for desktop OS).

StructType

FuncList	DESC
TUIDeviceInfo	Device information.

EnumType

EnumType	DESC
TUIMediaDeviceType	Device type.
TUIMediaDeviceState	Device operation.
TUIAudioRoute	Audio routing (the route via which audio is played).

onDeviceChanged:type:state:

onDeviceChanged:type:state:

- (void)onDeviceChanged:	(NSString *)deviceId
type:	(TUIMediaDeviceType)type
state:	(TUIMediaDeviceState)state

The status of the local device changed (only available for desktop OS).

Param	DESC
deviceId	Device ID.
state	0: Device has been added; 1: Device has been removed; 2: Device has been enabled.
type	Device type.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types. When the status of local device changed, the SDK will trigger this event callback.

onTestCameraVideoFrameRendered:height:

onTestCameraVideoFrameRendered:height:

- (void)onTestCameraVideoFrameRendered:	(NSInteger)width
height:	(NSInteger)height

Test camera video rendered success callback.

Param	DESC
height	Height of the video.
width	Width of the video.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types. When the local test camera video rendered successfully, the SDK will trigger the callback.

onTestMicVolume:

onTestMicVolume:

- (void)onTestMicVolume:	(NSInteger)volume
--------------------------	-------------------

Test microphone volume callback.

Param	DESC
volume	The volume value captured by the microphone, with a range of 0 - 100.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types. When testing the local microphone, the SDK will trigger the callback.

onTestSpeakerVolume:

onTestSpeakerVolume:

- (void)onTestSpeakerVolume:	(NSInteger)volume
------------------------------	-------------------

Test speakerphone volume callback.

Param	DESC
volume	The volume value set by the SDK, with a range of 0 - 100.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types. When testing the local speakerphone, the SDK will trigger the callback.

isFrontCamera

isFrontCamera

Query whether the front camera is being used (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

switchCamera:

switchCamera:

- (void)switchCamera:	(BOOL)frontCamera
-----------------------	-------------------

Switch to the front/rear camera (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

isAutoFocusEnabled

isAutoFocusEnabled

Query whether automatic face detection is supported (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

enableCameraAutoFocus:

enableCameraAutoFocus:

- (void)enableCameraAutoFocus:	(BOOL)enabled
--------------------------------	---------------

Enable auto focus (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

After auto focus is enabled, the camera will automatically detect and always focus on faces.

enableCameraTorch:

enableCameraTorch:

- (void)enableCameraTorch:	(BOOL)enabled
----------------------------	---------------

Enable/Disable flash, i.e., the torch mode (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

setAudioRoute:

setAudioRoute:

- (void)setAudioRoute:	(TUIAudioRoute)route
------------------------	--

Set the audio route (only available for mobile OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

A mobile phone has two audio playback devices: the receiver at the top and the speaker at the bottom.

If the audio route is set to the receiver, the volume is relatively low, and audio can be heard only when the phone is put near the ear. This mode has a high level of privacy and is suitable for answering calls.

If the audio route is set to the speaker, the volume is relatively high, and there is no need to put the phone near the ear.

setObserver:

setObserver:

- (void)setObserver:	(id< TUIRoomDeviceManagerObserver >)observer
----------------------	--

Set event observer.

Param	DESC
observer	Listening instance.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

You can use `TUIRoomDeviceManagerObserver` to receive events.

startCameraDeviceTest:

startCameraDeviceTest:

- (void)startCameraDeviceTest:	(TUIVideoView * __nullable)view
--------------------------------	--

Start camera testing (only available for desktop OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
You can use the `$setCurrentDevice$` API to switch between cameras during testing.

stopCameraDeviceTest

stopCameraDeviceTest

End camera testing (only available for desktop OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

startMicDeviceTest:playback:

startMicDeviceTest:playback:

- (void)startMicDeviceTest:	(NSInteger)interval
playback:	(BOOL)playback

Start microphone testing (only available for desktop OS).

Param	DESC
interval	Interval of volume callbacks.
playback	true: The user can hear his own sound during testing. false: The user can not hear his own sound during testing.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.
This API is used to test whether the mic functions properly. The mic volume detected (value range: 0-100) is returned via a callback.

stopMicDeviceTest

stopMicDeviceTest

End microphone testing (only available for desktop OS)

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

startSpeakerDeviceTest:

startSpeakerDeviceTest:

- (void)startSpeakerDeviceTest:	(NSString *) filePath
---------------------------------	-----------------------

Start speaker testing (only available for desktop OS).

Param	DESC
filePath	Path of the audio file.

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

This API is used to test whether the audio playback device functions properly by playing a specified audio file. If users can hear audio during testing, the device functions properly.

stopSpeakerDeviceTest

stopSpeakerDeviceTest

End speaker testing (only available for desktop OS).

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getDevicesList:callback:

getDevicesList:callback:

- (void)getDevicesList:	(TUIMediaDeviceType)type
callback:	(TUIDeviceListBlock)callback

Get the device list (only available for desktop OS).

Param	DESC
type	Device type. More details, see <code>\$TUIMediaDeviceType\$</code> .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

setCurrentDevice:deviceId:

setCurrentDevice:deviceId:

- (void)setCurrentDevice:	(TUIMediaDeviceType) type
deviceId:	(NSString *) deviceId

Set the device to use (only available for desktop OS).

Param	DESC
deviceId	Device ID, which you can use getDevicesList to get.
type	Device type. More details, see <code>\$TUIMediaDeviceType\$</code> .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

getCurrentDevice:callback:

getCurrentDevice:callback:

- (void)getCurrentDevice:	(TUIMediaDeviceType) type
callback:	(TUIDeviceInfoBlock) callback

Get the device currently in use (only available for desktop OS).

Param	DESC
type	Device type. More details, see <code>\$TUIMediaDeviceType\$</code> .

Note

The function supports the [TUIRoomTypeConference](#) and [TUIRoomTypeLive](#) room types.

TUIMediaDeviceType

TUIMediaDeviceType

Device type.

Enum	Value	DESC
TUIMediaDeviceTypeUnknown	-1	Undefined device type.
TUIMediaDeviceTypeAudioInput	0	Microphone.
TUIMediaDeviceTypeAudioOutput	1	Speaker or earpiece.
TUIMediaDeviceTypeVideoCamera	2	Camera.

TUIMediaDeviceState

TUIMediaDeviceState

Device operation.

Enum	Value	DESC
TUIMediaDeviceStateAdd	0	Device has been added.
TUIMediaDeviceStateRemove	1	Device has been removed.
TUIMediaDeviceStateActive	2	Device has been enabled.

TUIAudioRoute

TUIAudioRoute

Audio routing (the route via which audio is played).

Enum	Value	DESC
TUIAudioRouteSpeakerphone	0	Speakerphone: The speaker at the bottom is used for playback. With relatively high volume, it is used to play

		music out loud.
TUIAudioRouteEarpiece	1	Earpiece: The speaker at the top is used for playback. With relatively low volume, it is suitable for call scenarios that require privacy.

TUIDeviceInfo

TUIDeviceInfo

Device information.

EnumType	DESC
deviceId	Device ID.
deviceName	Device name.
deviceProperties	Device properties.

TUILiveListManager

Last updated : 2025-02-13 16:38:49

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUILiveListManager @ TUIKitEngine.

Function: Live Room list APIS, the functions on this webpage only support to [TUIRoomTypeLive](#) room type.

TUILiveListManager

TUILiveListManagerObserver

FuncList	DESC
onLiveInfoChanged:modifyFlag:	Live information changed callback.

TUILiveListManager

FuncList	DESC
setObserver:	Set the event observer.
setLiveInfo:modifyFlag:onSuccess:onError:	Modify live information.
getLiveInfo:onSuccess:onError:	Get live information.
fetchLiveList:count:onSuccess:onError:	Get a list of live rooms, with a maximum of 50 returned per fetch.

StructType

FuncList	DESC
TUILiveInfo	Live information.

onLiveInfoChanged:modifyFlag:

onLiveInfoChanged:modifyFlag:

- (void)onLiveInfoChanged:	(TUILiveInfo *)liveInfo
modifyFlag:	(TUILiveModifyFlag)modifyFlag

Live information changed callback.

Param	DESC
liveInfo	Live information.
modifyFlag	Modifiable parameter. More details, see TUILiveModifyFlag .

setObserver:

setObserver:

- (void)setObserver:	(id< TUILiveListManagerObserver >)observer
----------------------	--

Set the event observer.

You can use [TUILiveListManagerObserver](#) to receive live room event.

Param	DESC
observer	Listening instance.

setLiveInfo:modifyFlag:onSuccess:onError:

setLiveInfo:modifyFlag:onSuccess:onError:

- (void)setLiveInfo:	(TUILiveInfo *)liveInfo
modifyFlag:	(TUILiveModifyFlag)modifyFlag
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Modify live information.

Param	DESC
liveInfo	Live information.
modifyFlag	Modifiable parameter. More details, see <code>TUILiveModifyFlag</code> .

getLiveInfo:onSuccess:onError:

getLiveInfo:onSuccess:onError:

- (void)getLiveInfo:	(NSString *)roomId
onSuccess:	(TUILiveInfoBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get live information.

Param	DESC
roomId	Room ID.

fetchLiveList:count:onSuccess:onError:

fetchLiveList:count:onSuccess:onError:

- (void)fetchLiveList:	(NSString *)cursor
count:	(NSInteger)count
onSuccess:	(TUILiveInfoListBlock)onSuccess
onError:	(TUIErrorBlock)onError

Get a list of live rooms, with a maximum of 50 returned per fetch.

Param	DESC
count	Set the number of live lists to be obtained in each request.
cursor	Set the index of live lists in each request.

TUILiveInfo

TUILiveInfo

Live information.

EnumType	DESC
activityStatus	Live room active status: User-defined tag.
backgroundUrl	CoverUrl: Live room background, maximum support is 200 bytes.
categoryList	Live room category tags, a single room supports up to 3 tags.
coverUrl	CoverUrl: Live room cover, maximum support is 200 bytes.
isPublicVisible	The public parameter can control whether the live room can be seen by others.
roomInfo	Room information (Read only). More details, see TUIRoomInfo .
viewCount	Total views (the count of user entering live room).

TUILiveConnectionManager

Last updated : 2025-02-13 16:38:50

Copyright (c) 2024 Tencent. All rights reserved.

Module: TUILiveConnectionManager @ TUIKitEngine

Function: Live-Connection APIS, the functions on this webpage only support to [TUIRoomTypeLive](#) room type.

TUILiveConnectionManager

TUILiveConnectionObserver

FuncList	DESC
onConnectionUserListChanged:joinedList:leavedList:	Callback for connected users changed.
onConnectionRequestReceived:inviteeList:extensionInfo:	Callback for received the connection invitation
onConnectionRequestCancelled:	Callback for canceled the connection invitation
onConnectionRequestAccept:	Callback for accepted the connection invitation
onConnectionRequestReject:	Callback for rejected the connection invitation
onConnectionRequestTimeout:invitee:	Callback for timeout the connection invitation

TUILiveConnectionManager

FuncList	DESC
addObserver:	Add event callback
removeObserver:	Remove event callback
requestConnection:timeout:extensionInfo:onSuccess:onError:	Request connection invitation
cancelConnectionRequest:onSuccess:onError:	Cancel request about connection invitation
acceptConnection:onSuccess:onError:	Accept the connection invitation

<code>rejectConnection:onSuccess:onError:</code>	Reject the connection invitation
<code>disconnect:onError:</code>	Exit the connection.

StructType

FuncList	DESC
<code>TUIConnectionUser</code>	Connection User Info

EnumType

EnumType	DESC
<code>TUIConnectionCode</code>	Connection Request Status

onConnectionUserListChanged:joinedList:leavedList:

onConnectionUserListChanged:joinedList:leavedList:

- (void)onConnectionUserListChanged:	(NSArray< <code>TUIConnectionUser</code> *> *)connectedList
joinedList:	(NSArray< <code>TUIConnectionUser</code> *> *)joinedList
leavedList:	(NSArray< <code>TUIConnectionUser</code> *> *)leavedList

Callback for connected users changed.

Param	DESC
connectedList	List of connected users.
joinedList	List of joined connected users.
leavedList	List of leaved connected users.

onConnectionRequestReceived:inviteeList:extensionInfo:

onConnectionRequestReceived:inviteeList:extensionInfo:

- (void)onConnectionRequestReceived:	(TUIConnectionUser *)inviter
inviteeList:	(NSArray< TUIConnectionUser *> *)inviteeList
extensionInfo:	(NSString*)extensionInfo

Callback for received the connection invitation

Param	DESC
extensionInfo	Extension info.
inviteeList	The list of TUIConnectionUser about the invitee in current connection.
inviter	The TUIConnectionUser about the inviter.

onConnectionRequestCancelled:**onConnectionRequestCancelled:**

- (void)onConnectionRequestCancelled:	(TUIConnectionUser *)inviter
---------------------------------------	---

Callback for canceled the connection invitation

Param	DESC
inviter	The TUIConnectionUser about the inviter.

onConnectionRequestAccept:**onConnectionRequestAccept:**

- (void)onConnectionRequestAccept:	(TUIConnectionUser *)invitee
------------------------------------	---

Callback for accepted the connection invitation

Param	DESC
invitee	The TUIConnectionUser about the invitee.

onConnectionRequestReject:

onConnectionRequestReject:

- (void)onConnectionRequestReject:	(TUIConnectionUser *)invitee
------------------------------------	---

Callback for rejected the connection invitation

Param	DESC
invitee	The TUIConnectionUser about the invitee.

onConnectionRequestTimeout:invitee:

onConnectionRequestTimeout:invitee:

- (void)onConnectionRequestTimeout:	(TUIConnectionUser *)inviter
invitee:	(TUIConnectionUser *)invitee

Callback for timeout the connection invitation

Param	DESC
invitee	The TUIConnectionUser about the invitee.
inviter	The TUIConnectionUser about the inviter.

addObserver:

addObserver:

- (void)addObserver:	(id< TUILiveConnectionObserver >)observer
----------------------	---

Add event callback

Param	DESC
observer	The instance being listened to.

removeObserver:

removeObserver:

- (void)removeObserver:	(id<TUILiveConnectionObserver>)observer
-------------------------	---

Remove event callback

Param	DESC
observer	The instance being listened to.

requestConnection:timeout:extensionInfo:onSuccess:onError:

requestConnection:timeout:extensionInfo:onSuccess:onError:

- (void)requestConnection:	(NSArray<NSString *>)roomIdList
timeout:	(NSTimeInterval)timeout
extensionInfo:	(NSString*)extensionInfo
onSuccess:	(TUIConnectionRequestBlock)onSuccess
onError:	(TUIErrorBlock)onError

Request connection invitation

Param	DESC
extensionInfo	Extension info.
roomIdList	The list of room IDs will be invited.
timeout	Timeout time, in seconds. If it is set to 0, the SDK will not execute timeout detection and will not trigger a timeout callback.

cancelConnectionRequest:onSuccess:onError:

cancelConnectionRequest:onSuccess:onError:

- (void)cancelConnectionRequest:	(NSArray<NSString *>)roomIdList

onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Cancel request about connection invitation

Param	DESC
roomIdList	The list of room IDs whose connection requests will be canceled.

acceptConnection:onSuccess:onError:

acceptConnection:onSuccess:onError:

- (void)acceptConnection:	(NSString *)roomId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Accept the connection invitation

Param	DESC
roomId	The room ID of the inviter about connection invitation.

rejectConnection:onSuccess:onError:

rejectConnection:onSuccess:onError:

- (void)rejectConnection:	(NSString *)roomId
onSuccess:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Reject the connection invitation

Param	DESC
roomId	The room ID of the inviter about connection invitation.

disconnect:onError:

disconnect:onError:

- (void)disconnect:	(TUISuccessBlock)onSuccess
onError:	(TUIErrorBlock)onError

Exit the connection.

Calling this interface will exit the room connection state, and can only be called in the connected state.

TUIConnectionCode

TUIConnectionCode

Connection Request Status

Enum	Value	DESC
TUIConnectionCodeUnknown	-1	default.
TUIConnectionCodeSuccess	0	Request success.
TUIConnectionCodeRoomNotExist	1	Request room not exist.
TUIConnectionCodeConnecting	2	The room you are invited to connect to is already in the invitation list or is already connected.
TUIConnectionCodeConnectingOtherRoom	3	The room you are invited to connect to is connecting with other rooms.
TUIConnectionCodeFull	4	The current number of connections has reached the maximum limit.
TUIConnectionCodeRetry	5	Please try again.

TUIConnectionUser

TUIConnectionUser

Connection User Info

--

EnumType	DESC
avatarUrl	The user avatar url of the connection user.
joinConnectionTime	The Timestamp of when the user joined the connection.
roomId	The room id of the connection user.
userId	The user id of the connection user.
userName	The user name of the connection user.

TUICommonDefine

Last updated : 2025-02-13 16:38:50

TUICommonDefine

StructType

FuncList	DESC
TUINetworkInfo	Network quality information.
TUINetworkQualityInfo	

EnumType

EnumType	DESC
TUIError	Error code definition.
TUINetworkQuality	Network quality.
TUIExtensionType	Extension type.

TUIError

TUIError

Error code definition.

Enum	Value	DESC
TUIErrorSuccess	0	Operate successfully.
TUIErrorFailed	-1	Unclassified error.
TUIErrorFreqLimit	-2	The operation was frequency-limited, please try again later.
TUIErrorRepeatOperation	-3	Operation is repeated.

TUIErrorSDKAppIDNotFound	-1000	SDKAppID not found. Please confirm the application information in the TRTC console .
TUIErrorInvalidParameter	-1001	An invalid parameter was passed in when the API was called.
TUIErrorSdkNotInitialized	-1002	Not logged in, please login first.
TUIErrorPermissionDenied	-1003	Failed to get permission. Audio/video permissions are currently denied. Please check whether the device permissions are enabled. You can use the following error codes to handle in the Room scenarios: The camera does not have system authorization: ERR_CAMERA_NOT_AUTHORIZED. The microphone does not have system authorization: ERR_MICROPHONE_NOT_AUTHORIZED.
TUIErrorRequirePayment	-1004	This function requires additional packages. Please activate the corresponding packages in the TRTC console .
TUIErrorCameraStartFail	-1100	Failed to turn the camera on.
TUIErrorCameraNotAuthorized	-1101	No permission to access the camera. Please check the system authorization.
TUIErrorCameraOccupied	-1102	The camera is occupied. Please check whether there are other processes using the camera.
TUIErrorCameraDeviceEmpty	-1103	There is currently no camera available.
TUIErrorMicrophoneStartFail	-1104	Failed to turn the microphone on.
TUIErrorMicrophoneNotAuthorized	-1105	No permission to access the microphone. Please check the system authorization.
TUIErrorMicrophoneOccupied	-1106	The microphone is occupied. Please check whether there are other processes using the microphone.
TUIErrorMicrophoneDeviceEmpty	-1107	There is currently no microphone available.
TUIErrorGetScreenSharingTargetFailed	-1108	Failed to obtain the screen sharing source (screen or window). Please check the screen

		recording permission.
TUIErrorStartScreenSharingFailed	-1109	Failed to start screen sharing. Please check whether anyone in the room is sharing the screen.
TUIErrorRoomIdNotExist	-2100	The room does not exist when entering the room.
TUIErrorOperationInvalidBeforeEnterRoom	-2101	This operation is valid after entering the room.
TUIErrorExitNotSupportedForRoomOwner	-2102	The owner can not leave room. Conference room type: You can transfer the owner first and then leaving room. Live room type: You can only dismiss the room.
TUIErrorOperationNotSupportedInCurrentRoomType	-2103	The current room type does not support the operation.
TUIErrorRoomIdInvalid	-2105	The roomId is invalid. It's must be printable ASCII characters (0x20-0x7e) and the maximum length is 48 bytes.
TUIErrorRoomIdOccupied	-2106	The room ID is occupied, please select another room ID.
TUIErrorRoomNameInvalid	-2107	The room name is invalid. The maximum length of the name is 30 bytes. If it contains Chinese, the character encoding must be UTF-8.
TUIErrorAlreadyInOtherRoom	-2108	The current user is already in other room and needs to leave room before joining a new room: A single roomEngine instance only supports user entering one room. If you want to enter a different room, please leave room first or use a new roomEngine instance.
TUIErrorNeedPassword	-2109	The current room needs a password to enter.
TUIErrorWrongPassword	-2110	Error password for entering the room.
TUIErrorRoomUserFull	-2111	The room is full of users.
TUIErrorRoomMetadataExceedKeyCountLimit	-2112	The number of room metadata keys exceeds

		the limit.
TUIErrorRoomMetadataExceedValueSizeLimit	-2113	The size of room metadata value exceeds the limit.
TUIErrorUserNotExist	-2200	The user does not exist.
TUIErrorUserNotEntered	-2201	The user is not in the current room.
TUIErrorUserNeedOwnerPermission	-2300	The operation needs owner permission.
TUIErrorUserNeedAdminPermission	-2301	The operation needs owner or administrator permission.
TUIErrorRequestNoPermission	-2310	The request does not have permission, such as canceling a request that was not requested by oneself.
TUIErrorRequestIdInvalid	-2311	The request ID is invalid or has been processed.
TUIErrorRequestIdRepeat	-2312	The request is repeated.
TUIErrorRequestIdConflict	-2313	The request conflict.
TUIErrorMaxSeatCountLimit	-2340	The number of seats exceeds the maximum number of seats in your package.
TUIErrorAlreadyInSeat	-2341	The current user is already on the seat.
TUIErrorSeatOccupied	-2342	The current seat is already occupied.
TUIErrorSeatLocked	-2343	The current seat is locked.
TUIErrorSeatIndexNotExist	-2344	The seat number does not exist.
TUIErrorUserNotInSeat	-2345	The current user is not on the seat.
TUIErrorAllSeatOccupied	-2346	The number of people on the seat is full.
TUIErrorSeatNotSupportLinkMic	-2347	The current room does not support connect microphone before the seat is enabled.
TUIErrorOpenMicrophoneNeedSeatUnlock	-2360	The current seat audio is locked, and you can't push audio stream when using the microphone.
TUIErrorOpenMicrophoneNeedPermissionFromAdmin	-2361	You need to apply the owner or administrator to open the microphone.

TUIErrorOpenCameraNeedSeatUnlock	-2370	The current seat video is locked, and you can't push video stream when using camera.
TUIErrorOpenCameraNeedPermissionFromAdmin	-2371	You need to apply to the owner or administrator to open the camera.
TUIErrorOpenScreenShareNeedSeatUnlock	-2372	The current seat video is locked. The owner needs to unlock the seat before screen sharing can be enabled.
TUIErrorOpenScreenShareNeedPermissionFromAdmin	-2373	You need to apply to the owner or administrator to enable screen sharing.
TUIErrorSendMessageDisabledForAll	-2380	All users can't send message in the current room.
TUIErrorSendMessageDisabledForCurrent	-2381	You can't send message in the current room.
TUIErrorRoomAlreadyConnected	-3001	The room has been connected with current.
TUIErrorRoomConnectedInOther	-3002	The room has been connected with other room.
TUIErrorMaxConnectedCountLimit	-3003	The current room connection exceeds the maximum limit.

TUINetworkQuality

TUINetworkQuality

Network quality.

Enum	Value	DESC
TUINetworkQualityUnknown	0	Undefine.
TUINetworkQualityExcellent	1	The current network is excellent.
TUINetworkQualityGood	2	The current network is good.
TUINetworkQualityPoor	3	The current network is poor.
TUINetworkQualityBad	4	The current network is bad.
TUINetworkQualityVeryBad	5	The current network is very bad.

TUINetworkQualityDown	6	The current network does not meet TRTC's minimum requirements.
-----------------------	---	--

TUIExtensionType

TUIExtensionType

Extension type.

Enum	Value	DESC
TUIExtensionTypeDeviceManager	1	Device management extension.
TUIExtensionTypeLiveListManager	2	Live management extension.
TUIExtensionTypeConferenceListManager	3	Conference list extensions.
TUIExtensionTypeConferenceInvitationManager	4	Conference invitation extensions.
TUIExtensionTypeLiveLayoutManager	5	Live layout extensions.

TUINetworkInfo

TUINetworkInfo

Network quality information.

EnumType	DESC
delay	Network delay(ms).
downLoss	Downlink packet loss rate, unit (%). The smaller the value, the better the network. If downLoss is 0%, it means that the network quality of the downlink is very good and the data packets received from the cloud are basically not lost. If downLoss is 30%, it means that 30% of the audio and video data packets transmitted from the cloud to the SDK will be lost in the transmission link.
quality	Network quality.
upLoss	Uplink packet loss rate, unit (%). The smaller the value, the better the network. If upLoss is 0%, it means that the network quality of the uplink is very good and the data packets uploaded to the cloud are basically not lost. If upLoss is 30%, it means that 30% of the audio and video data packets sent by the SDK to the cloud will be lost in the transmission link.

userId	User ID.
--------	----------

TUIRoomDefine

Last updated : 2025-02-13 16:38:50

TUIRoomDefine

StructType

FuncList	DESC
TUIRoomInfo	Room information.
TUILoginUserInfo	User login information.
TUIUserInfo	User information in the room.
TUIRoomVideoEncoderParams	Video encoding parameters.
TUISeatInfo	Room seat information.
TUISeatLockParams	Lock seat parameters.
TUIUserVoiceVolume	User volume in the room.
TUIRequest	Request.
TUIEnterRoomOptions	Enter a room parameters.

EnumType

EnumType	DESC
TUIRoomType	Room type.
TUISeatMode	Seat mode.
TUIMediaDevice	Types of media devices in the room.
TUIRole	Type of user role in the room.
TUIRoomDismissedReason	The reason of dismissing room.
TUIVideoQuality	Video quality.

TUIAudioQuality	Audio quality.
TUIVideoStreamType	Video stream type.
TUIChangeReason	Reasons for audio and video status changed.
TUIKickedOutOfRoomReason	Reasons for users being kicked off from the room.
TUIResolutionMode	Resolution mode.
TUICaptureSourceType	Screen sharing capture source type.
TUIRequestAction	Request type.

TUIRoomType

TUIRoomType

Room type.

Enum	Value	DESC
TUIRoomTypeConference	1	Conference room, suitable for conference and education scenarios.
TUIRoomTypeLive	2	Live room, suitable for live scenarios.

TUISeatMode

TUISeatMode

Seat mode.

Enum	Value	DESC
TUISeatModeFreeToTake	1	Free to take seat mode. Users can take the seat freely without applying.
TUISeatModeApplyToTake	2	Apply to take seat mode. Users can take the seat after the owner or administrator approve.

TUIMediaDevice

TUIMediaDevice

Types of media devices in the room.

Enum	Value	DESC
TUIMediaDeviceMicrophone	1	Microphone.
TUIMediaDeviceCamera	2	Camera.
TUIMediaDeviceScreenSharing	3	Screen sharing.

TUIRole

TUIRole

Type of user role in the room.

Enum	Value	DESC
TUIRoleRoomOwner	0	Owner, generally refers to the room creator.
TUIRoleAdministrator	1	Administrator.
TUIRoleGeneralUser	2	General user.

TUIRoomDismissedReason

TUIRoomDismissedReason

The reason of dismissing room.

Enum	Value	DESC
TUIRoomDismissedReasonByOwner	1	Dismissed by owner.
TUIRoomDismissedReasonByServer	2	Dismissed by the room server.

TUIVideoQuality

TUIVideoQuality

Video quality.

Enum	Value	DESC
TUIVideoQuality360P	1	Low-definition 360P.
TUIVideoQuality540P	2	Standard definition 540P.
TUIVideoQuality720P	3	High definition 720P.
TUIVideoQuality1080P	4	Full high definition 1080P.

TUIAudioQuality

TUIAudioQuality**Audio quality.**

Enum	Value	DESC
TUIAudioQualitySpeech	0	Speech mode.
TUIAudioQualityDefault	1	Default mode.
TUIAudioQualityMusic	2	Music mode.

TUIVideoStreamType

TUIVideoStreamType**Video stream type.**

Enum	Value	DESC
TUIVideoStreamTypeCameraStream	0	HD camera video stream.
TUIVideoStreamTypeScreenStream	1	Screen sharing video stream.
TUIVideoStreamTypeCameraStreamLow	2	Low-definition camera video stream.

TUIChangeReason

TUIChangeReason

Reasons for audio and video status changed.

Enum	Value	DESC
TUIChangeReasonBySelf	0	Changed by self.
TUIChangeReasonByAdmin	1	Changed by the owner and administrator.

TUIKickedOutOfRoomReason

TUIKickedOutOfRoomReason

Reasons for users being kicked off from the room.

Enum	Value	DESC
TUIKickedOutOfRoomReasonByAdmin	0	Kicked off by owner or administrator.
TUIKickedOutOfRoomReasonByLoggedInOtherDevice	1	Kicked off when other devices enter the room.
TUIKickedOutOfRoomReasonByServer	2	Kicked off by the server.

TUIResolutionMode

TUIResolutionMode

Resolution mode.

Enum	Value	DESC
TUIResolutionModeLandscape	0	Landscape.
TUIResolutionModePortrait	1	Portrait.

TUICaptureSourceType

TUICaptureSourceType

Screen sharing capture source type.

Enum	Value	DESC
TUICaptureSourceTypeUnknown	-1	Undefine.
TUICaptureSourceTypeWindow	0	Window.
TUICaptureSourceTypeScreen	1	Screen.

TUIRequestAction

TUIRequestAction**Request type.**

Enum	Value	DESC
TUIRequestActionInvalidAction	0	Invalid request.
TUIRequestActionOpenRemoteCamera	1	Request the remote user to open the camera.
TUIRequestActionOpenRemoteMicrophone	2	Request the remote user to open the microphone.
TUIRequestActionTakeSeat	4	Request to take seat.
TUIRequestActionRemoteUserOnSeat	5	Request the remote user to take seat.
TUIRequestActionApplyToAdminToOpenLocalCamera	6	Request to open local camera by the administrator.
TUIRequestActionApplyToAdminToOpenLocalMicrophone	7	Request to open local microphone by the administrator.
TUIRequestActionApplyToAdminToOpenLocalScreenShare	8	Request to enable the screen sharing by the administrator.

TUIRoomInfo

TUIRoomInfo

Room information.

EnumType	DESC
createTime	The room creation time (read-only).
isCameraDisableForAllUser	The status of disabling for all user to open the camera in room (optional parameter for creating a room), default value: NO.
isMessageDisableForAllUser	The status of disabling for all user to send message in room (optional parameter for creating a room), default value: NO.
isMicrophoneDisableForAllUser	The status of disabling for all user to open the microphone in room (optional parameter for creating a room), default value: NO.
isScreenShareDisableForAllUser	The status of disabling for all user to start the screen sharing in room (optional parameter for creating a room), default value: NO.
isSeatEnabled	The status of enabling seat.
maxSeatCount	Maximum number of seat.
memberCount	The count of members in the room (read-only).
name	Room name (Optional parameter, default is room ID, maximum support is 100 bytes).
ownerAvatarUrl	Room owner avatar URL: Default is the room creator's avatar URL (read-only).
ownerId	Owner ID: Default is the room creator (read-only).
ownerName	Room owner name: Default is the room creator's name (read-only).
password	Room password.
roomId	Room ID (String type, required parameter for creating room, maximum support is 48 bytes).
roomType	Room type (Optional parameter). More details, see: TUIRoomType .
seatMode	Seat mode (only available after isSeatEnabled is true).

TUILoginUserInfo

TUILoginUserInfo

User login information.

EnumType	DESC
avatarUrl	User avatar URL.
customInfo	Custom information.
userId	User ID.
userName	User name.

TUIUserInfo

TUIUserInfo

User information in the room.

EnumType	DESC
avatarUrl	Use avatar URL.
hasAudioStream	The status of the user has audio stream, default value: NO.
hasScreenStream	The status of the user has screen sharing stream, default value: NO.
hasVideoStream	The status of the user has video stream, default value: NO.
isMessageDisabled	The status of disabling the user to send message, default value: NO.
nameCard	User nickname in the room, maximum support is 32 bytes.
roomCustomInfo	Room user custom information.
userId	User ID.
userName	User name.
userRole	User role type. Conference room roles only exist within the room. Once checked out and re-entered, the role becomes a general user. Live room can be set before entering the room, and as long as the room is not disbanded, the role still exists. More details, see: TUIRole .

TUIRoomVideoEncoderParams

TUIRoomVideoEncoderParams

Video encoding parameters.

EnumType	DESC
bitrate	Target video bitrate.
fps	Video capture frame rate.
resolutionMode	Resolution mode. More details, see: TUIResolutionMode .
videoResolution	Video quality. More details, see: TUIVideoQuality .

TUISeatInfo

TUISeatInfo

Room seat information.

EnumType	DESC
avatarUrl	User avatar URL.
index	Seat index.
isAudioLocked	The status of audio-locked for the seat. Default value: NO.
isLocked	The status of locked for the seat. Default value: NO.
isVideoLocked	The status of video-locked for the seat. Default value: NO.
nameCard	User nickname in the room.
userId	User ID.
userName	User name.

TUISeatLockParams

TUISeatLockParams

Lock seat parameters.

EnumType	DESC
lockAudio	Lock seat audio, default value: NO.
lockSeat	Lock seat, default value: NO.
lockVideo	Lock seat camera, default value: NO.

TUIUserVoiceVolume

TUIUserVoiceVolume**User volume in the room.**

EnumType	DESC
userId	User ID.
volume	Volume: The volume of all users who are speaking, the value range is 0 - 100.

TUIRequest

TUIRequest**Request.**

EnumType	DESC
avatarUrl	User avatar URL.
content	Request content.
nameCard	User nickname in the room.
requestAction	Request type. More details, see TUIRequestAction .
requestId	Request ID.
timestamp	Timestamp.
userId	User ID.
userName	User name.

TUIEnterRoomOptions

TUIEnterRoomOptions

Enter a room parameters.

EnumType	DESC
password	Room password.

Flutter

UIKit API

SeatGridWidget

Last updated : 2025-04-17 14:16:15

API Introduction

SeatGridWidget is a basic control we developed for the voice chat room UIKit. This core control provides various APIs such as enabling/disabling the voice chat room, managing microphone positions in the live streaming room, including applying for microphone mode, inviting speakers, moving microphone positions, and kicking someone off the mic.

API Overview

API	Description
SeatGridController	Create a SeatGridController object
SeatGridWidget	Create a SeatGridWidget object
startMicrophone	Open the local microphone
stopMicrophone	Close the local microphone
muteMicrophone	Pause publishing local audio stream
unmuteMicrophone	Resume publishing local audio stream
startVoiceRoom	Anchor creates a live room and starts streaming.
stopVoiceRoom	Anchor stops streaming and destroys the live room.
joinVoiceRoom	Audience joins an anchor's live streaming room.
leaveVoiceRoom	Audience leaves an anchor's live streaming room.
updateRoomSeatMode	Update the seat mode of the room.
responseRemoteRequest	Anchor responds to microphone application / Audience responds to microphone invitation

cancelRequest	Anchor cancels microphone invitation / Audience cancels microphone application
takeSeat	Become a speaker.
moveToSeat	Move the seat.
leaveSeat	Leave the seat.
takeUserOnSeatByAdmin	Anchor invites users to speak.
kickUserOffSeatByAdmin	Anchor removes a user from the seat.
lockSeat	Anchor locks the seat (including position lock, audio status lock and video status lock)
setLayoutMode	Anchor sets the layout mode of the seat list.
addObserver	Set event callback
removeObserver	Remove event callback

API Details

SeatGridController

Create an object instance of `SeatGridController` . `SeatGridController` is responsible for providing APIs for the voice chat room scenario.

```
SeatGridController()
```

Return Value: `SeatGridController`

SeatGridWidget

Create an instance of `SeatGridWidget` . `SeatGridWidget` is responsible for rendering the seat UI.

```
SeatGridWidget(  
    {super.key,  
    required this.controller,  
    this.seatWidgetBuilder,  
    this.onSeatWidgetTap});
```

Parameters:

Parameter	Type	Meaning

key	Key?	How does Flutter control replacing the parameters of an old widget with a new widget
controller	SeatGridController	Controller of SeatGridWidget, responsible for providing APIs in the live voice room scenario
seatWidgetBuilder	SeatWidgetBuilder	Constructor of the custom seat widget
onSeatWidgetTap	OnSeatWidgetTap	Callback for seat click event

Return Value: SeatGridWidget

startMicrophone

Open the local microphone.

```
Future<TUIActionCallback> startMicrophone()
```

Return Value: Future<[TUIActionCallback](#)>

stopMicrophone

Close the local microphone.

```
void stopMicrophone()
```

Return Value: void

muteMicrophone

Pause publishing local audio stream.

```
Future<TUIActionCallback> muteMicrophone()
```

Return Value: Future<[TUIActionCallback](#)>

unmuteMicrophone

Resume publishing local audio stream.

```
Future<TUIActionCallback> unmuteMicrophone()
```

Return Value: Future<[TUIActionCallback](#)>

startVoiceRoom

Anchor creates a live room and starts streaming.

```
Future<TUIValueCallBack<TUIRoomInfo>> startVoiceRoom(TUIRoomInfo roomInfo)
```

Parameters:

Parameter	Type	Meaning
roomInfo	TUIRoomInfo	Information of creating a live streaming room

Return Value: Future<[TUIValueCallback](#)<[TUIRoomInfo](#)>>

stopVoiceRoom

Anchor stops streaming and destroys the live room.

```
Future<TUIActionCallback> stopVoiceRoom()
```

Return Value: Future<[TUIActionCallback](#)>

joinVoiceRoom

Audience joins an anchor's live streaming room.

```
Future<TUIValueCallback<TUIRoomInfo>> joinVoiceRoom(String roomId)
```

Parameters:

Parameter	Type	Meaning
roomId	String	Live Streaming Room ID

Return Value: Future<[TUIValueCallback](#)<[TUIRoomInfo](#)>>

leaveVoiceRoom

Audience leaves an anchor's live streaming room.

```
Future<TUIActionCallback> leaveVoiceRoom()
```

Return Value: Future<[TUIActionCallback](#)>

updateRoomSeatMode

Update the seat mode of the room.

```
Future<TUIActionCallback> updateRoomSeatMode(TUISeatMode seatMode)
```

Parameters:

Parameter	Type	Meaning
seatMode	TUISeatMode	Free to Take: In free-speaking mode, the audience can freely join the

podium without applying.
applyToTake: Audience become speakers only after the broadcaster agrees.

Return Value: Future<[TUIActionCallback](#)>

responseRemoteRequest

Anchor responds to microphone application / Audience responds to microphone invitation.

```
Future<TUIActionCallback> responseRemoteRequest(String userId, bool agree)
```

Parameters:

Parameter	Type	Meaning
userId	String	User ID that responds to the user. If the current identity is an audience, the ID can be left blank.
agree	bool	Whether to accept requests: true for accept, false for deny requests

Return Value: Future<[TUIActionCallback](#)>

cancelRequest

Anchor cancels microphone invitation / Audience cancels microphone application

```
Future<TUIActionCallback> cancelRequest(String userId)
```

Parameters:

Parameter	Type	Meaning
userId	String	User ID to be cancelled. If the current identity is an audience, the ID can be left blank.

Return Value: Future<[TUIActionCallback](#)>

takeSeat

Request to speak (application required in speaking mode)

```
Future<RequestCallback> takeSeat(int seatIndex, int timeout)
```

Parameters:

Parameter	Type	Meaning

index	int	Microphone position ID for speaking
timeout	int	Timeout period, in seconds. If set to 0, the SDK will not perform timeout detection or trigger a timeout callback.

Return Value: Future<[RequestCallback](#)>

moveToSeat

Remove seat (this function can only be invoked by the user already in the seat)

```
Future<TUIActionCallback> moveToSeat (int index)
```

Parameters:

Parameter	Type	Meaning
index	int	Microphone position ID that needs to be moved to

Return Value: Future<[TUIActionCallback](#)>

leaveSeat

Become a listener.

```
Future<TUIActionCallback> leaveSeat ()
```

Return Value: Future<[TUIActionCallback](#)>

takeUserOnSeatByAdmin

Anchor invites users to speak.

```
Future<RequestCallback> takeUserOnSeatByAdmin (int seatIndex, String userId, int tim
```

Parameters:

Parameter	Type	Meaning
index	int	Invited microphone position ID
userId	String	ID of the invited user
timeout	int	Timeout period, in seconds. If set to 0, the SDK will not perform timeout detection or trigger a timeout callback.

Return Value: Future<[RequestCallback](#)>

kickUserOffSeatByAdmin

Anchor removes a user from the seat.

```
Future<TUIActionCallback> kickUserOffSeatByAdmin(String userId)
```

Parameters:

Parameter	Type	Meaning
userId	String	User ID of the kicked-off user

Return Value: Future<[TUIActionCallback](#)>

lockSeat

Mute a speaker. Anchor locks the seat (including position lock, audio status lock and video status lock)

```
Future<TUIActionCallback> lockSeat(int index, TUISeatLockParams lockMode)
```

Parameters:

Parameter	Type	Meaning
seatIndex	int	The microphone position ID that needs to be locked.
lockMode	TUISeatLockParams	Microphone Mute Parameters

Return Value: Future<[TUIActionCallback](#)>

setLayoutMode

Set the layout mode of the seat list.

```
void setLayoutMode(LayoutMode layoutMode, SeatWidgetLayoutConfig? layoutConfig)
```

Parameters:

Parameter	Type	Meaning
layoutMode	LayoutMode	The layout mode of the seat position list supports focus layout, grid layout, vertical layout, and free layout.
layoutConfig	SeatWidgetLayoutConfig	Layout configuration information only takes effect in free layout mode.

Return Value: void

addObserver

Set event callback.

```
void addObserver(SeatGridWidgetObserver observer)
```

Parameters:

Parameter	Type	Meaning
observer	SeatGridWidgetObserver	Callback object of the core component

Return Value: void

removeObserver

Remove event callback.

```
void removeObserver(SeatGridWidgetObserver observer)
```

Parameters:

Parameter	Type	Meaning
observer	SeatGridWidgetObserver	Callback object of the core component

Return Value: void

Type Definition

Type	Meaning
SeatWidgetBuilder	Custom seat widget constructor
OnSeatWidgetTap	Seat click event
OnRoomDismissed	Received room destruction event
OnKickedOutOfRoom	Received event of being removed from room
OnSeatRequestReceived	Received request event to speak / Received invitation to speak event
OnSeatRequestCancelled	Event of request to speak / invitation to speak being canceled
OnKickedOffSeat	Received event of user removed from microphone

OnUserAudioStateChanged	User audio status change event.
LayoutMode	The layout mode of the seat position list supports focus layout, grid layout, vertical layout, and custom layout.
SeatWidgetLayoutRowAlignment	Alignment mode of seat layout
RequestType	Request type (apply for microphone mode and invitation to speak)
RequestResultType	Request result type
RequestCallback	Request result callback
SeatWidgetLayoutConfig	Seat layout configuration information
SeatWidgetLayoutRowConfig	Seating layout row layout configuration information

SeatWidgetBuilder

Constructor of the custom seat `widget`

```
typedef SeatWidgetBuilder = Widget Function(  
    BuildContext context,  
    ValueNotifier<TUISeatInfo> seatInfoNotifier,  
    ValueNotifier<int> volumeNotifier);
```

Parameters:

Parameter	Type	Meaning
context	BuildContext	Context
seatInfoNotifier	ValueNotifier< TUISeatInfo >	Seat information notifier
volumeNotifier	ValueNotifier<int>	Volume information notifier

OnSeatWidgetTap

Seat click event

```
typedef OnSeatWidgetTap = void Function(TUISeatInfo seatInfo);
```

Parameters:

Parameter	Type	Meaning
seatInfo	TUISeatInfo	Microphone position information

OnRoomDismissed

Received room destruction event

```
typedef OnRoomDismissed = void Function(String roomId);
```

Parameters:

Parameter	Type	Meaning
roomId	String	room ID

OnKickedOutOfRoom

Received event of being removed from room

```
typedef OnKickedOutOfRoom = void Function(String roomId, TUIKickedOutOfRoomReason reason, String message);
```

Parameters:

Parameter	Type	Meaning
roomId	String	room ID
reason	TUIKickedOutOfRoomReason	Reason for being kicked out
message	String	Kicked-out info

OnSeatRequestReceived

Received request event to speak / Received invitation to speak event

```
typedef OnSeatRequestReceived = void Function(RequestType type, TUIUserInfo userInfo, String message);
```

Parameters:

Parameter	Type	Meaning
type	RequestType	Request Type
userInfo	TUIUserInfo	Requester Information

OnSeatRequestCancelled

Event of request to speak / invitation to speak being canceled

```
typedef OnSeatRequestCancelled = void Function(RequestType type, TUIUserInfo userInfo, String message);
```

Parameters:

--	--	--

Parameter	Type	Meaning
type	RequestType	Request Type
userInfo	TUIUserInfo	Operator Information

OnKickedOffSeat

Received event of user removed from microphone

```
typedef OnKickedOffSeat = void Function(TUIUserInfo userInfo);
```

Parameters:

Parameter	Type	Meaning
userInfo	TUIUserInfo	Operator Information

OnUserAudioStateChanged

User audio status change event

```
typedef OnUserAudioStateChanged = void Function(TUIUserInfo userInfo, bool hasAudio
```

Parameters:

Parameter	Type	Meaning
userInfo	TUIUserInfo	Operator Information
hasAudio	bool	Whether there is an audio stream
reason	TUIChangeReason	Reason for audio stream change

LayoutMode

Layout mode of the seat position list

Enumeration Value	Meaning
focus	Focus on layout
grid	Grid layout
vertical	Vertical layout
free	Custom Layout

SeatWidgetLayoutRowAlignment

Alignment mode of seat layout

Enumeration Value	Meaning
start	Move the seat closer to the starting position
end	Move the seat closer to the ending position
center	Move the seat closer to the intermediate position
spaceBetween	Do not leave space before the first seat and after the last seat. Evenly distribute the remaining space between other seats.
spaceAround	Distribute half of the space before the first seat and after the last seat. Evenly distribute the remaining space between other seats.
spaceEvenly	Evenly distribute the remaining space between all seats.

RequestType

Request Type

Enumeration Value	Meaning
applyToTakeSeat	Request to speak
inviteToTakeSeat	Invitation to speak

RequestResultType

Request to speak / Invitation to speak callback

Enumeration Value	Meaning
onAccepted	The request is accepted.
onRejected	Request rejected.
onCancelled	The request is canceled.
onTimeout	Request timeout
onError	Request anomaly

RequestCallback

Request result callback

--	--

Attribute	Type	Meaning
code	TUIError	Error Code Enumeration
message	String	Error code information
type	RequestResultType	Request result type
userInfo	TUIUserInfo	Request Processor

SeatWidgetLayoutRowConfig

Seating layout row layout configuration information

Attribute	Type	Meaning
count	int	Microphone Quantity Displayed in This Row
seatSpacing	double	Seat Horizontal Spacing in This Row (This Parameter Is Valid Only When the Alignment Mode Is START, END, and CENTER)
seatSize	Size	Seat Layout Size in This Row
alignment	SeatWidgetLayoutRowAlignment	Alignment mode of layout in this row

SeatWidgetLayoutConfig

Seat layout configuration information

Required	Type	Description
rowConfigs	List< SeatWidgetLayoutRowConfig >	Configuration information list of all rows in seat layout
rowSpacing	double	Line Spacing in Seat Layout

Event Definition

SeatGridWidgetObserver

Event List	Type	Meaning
onRoomDismissed	OnRoomDismissed	Received room destruction event
onKickedOutOfRoom	OnKickedOutOfRoom	Received event of being removed from room

onSeatRequestReceived	OnSeatRequestReceived	Received request event to speak / Received invitation to speak event
onSeatRequestCancelled	OnSeatRequestCancelled	Event of request to speak / invitation to speak being canceled
onKickedOffSeat	OnKickedOffSeat	Received event of user removed from microphone
onUserAudioStateChanged	OnUserAudioStateChanged	User audio status change event

Engine API

API Overview

Last updated : 2025-04-17 10:26:36

Overview

TUIRoomEngine ([rtc_room_engine](#)) is a component designed for scenarios involving **corporate meetings, webinars, online education**, etc., supporting multi-person audio and video conversations. It offers room management, multi-person TRTC interactions, member management, screen sharing, and other meeting control features, and supports various video qualities including standard definition, high definition, and ultra-high definition. By integrating this component, you can add multi-person audio and video conversation features to your application.

Integration method:

In your project's `pubspec.yaml` , add the following code to integrate TUIRoomEngine:

```
dependencies:
  rtc_room_engine: latest version
```

Run the following command to install the component:

```
flutter pub get
```

TUIRoomEngine API List

TUIRoomEngine API is a no UI interface for multi-person audio and video rooms, you can use these APIs to perform custom encapsulation based on your business needs.

TUIRoomEngine

TUIRoomEngine Core Methods

API	Description
sharedInstance	Create TUIRoomEngine Instance.
destroyInstance	Destroy TUIRoomEngine Instance.
login	Login Interface, you need to initialize user information first to enter the room and perform a series of operations.

logout	Logout Interface, which includes proactively leaving the room and destroying resources.
setSelfInfo	Set Local Username and Avatar.
setLoginUserInfo	Set Login User Information.
getSelfInfo	Obtain Basic Information of Local User Login.
addObserver	Set event callbacks.
removeObserver	Remove event callbacks.

Room-related proactive interface

API	Description
createRoom	Create Room.
destroyRoom	Dissolve Room.
enterRoom	Enter the room.
exitRoom	Leave the room.
connectOtherRoom	Connect to Other Rooms.
disconnectOtherRoom	Disconnect from Other Rooms.
fetchRoomInfo	Obtain room information.
updateRoomNameByAdmin	Update Room Name (only administrators or group owners can invoke).
updateRoomSeatModeByAdmin	Set Room Management Mode (only administrators or group owners can invoke).

Local User View Rendering, Video Management

API	Description
setLocalVideoView	Set control for local user video rendering.
openLocalCamera	Turns the local camera on.
closeLocalCamera	Turns the local camera off.
updateVideoQuality	Update local video encoding quality settings.

updateVideoQualityEx	Set encoding parameters for the video encoder
setVideoResolutionMode	Set the resolution mode for the video encoder
enableGravitySensor	Enable gravity sensing mode
startPushLocalVideo	Start pushing local video.
stopPushLocalVideo	Stop pushing local video.
startScreenSharing	Starting Screen Sharing
stopScreenSharing	End Screen Sharing

Local User Audio Management

API	Description
openLocalMicrophone	Enable the local mic.
closeLocalMicrophone	Disable the local mic.
updateAudioQuality	Update local audio encoding quality settings.
muteLocalAudio	Stop pushing local audio.
unMuteLocalAudio	Start pushing local audio.

Remote user view rendering, Video Management

API	Description
setRemoteVideoView	Set control for remote user video rendering.
startPlayRemoteVideo	Start playing remote user video.
stopPlayRemoteVideo	Stop playing remote user video.
muteRemoteAudioStream	Mute remote user.

Inside the Room User Information

API	Description
getUserList	Obtain the member list inside the room.
getUserInfo	Obtain member information.

In-room User Management

API	Description
changeUserRole	Modify User Role (only administrators or group owners can invoke).
changeUserNameCard	Modify User Nickname.
kickRemoteUserOutOfRoom	Remove Remote User from Room (only administrators or group owners can invoke).
addCategoryTagForUsers	Add a label to a user (only homeowners can call)
removeCategoryTagForUsers	Remove a label from a user (only homeowners can call)
getUserListByTag	Get user information in the room based on labels
setCustomInfoForUser	Set customized information for members in the room

User speaking management in the room

API	Description
disableDeviceForAllUserByAdmin	Control whether all users in the current room can enable audio stream, video stream capture devices. For example: prohibit everyone from turning on the microphone, camera, or screen sharing (currently available only in meeting scenarios, and only administrators or group owners can invoke).
openRemoteDeviceByAdmin	Request Remote User to Enable Media Devices (only administrators or group owners can invoke).
closeRemoteDeviceByAdmin	Turn Off Remote User Media Devices (only administrators or group owners can invoke).
applyToAdminToOpenLocalDevice	Request to Enable Local Media Devices (available for regular users).

Microphone Position Management within the Rooms

API	Description
setMaxSeatCount	Set Maximum Number of Microphones (can only be set before entering or creating a room).
getSeatList	Get the microphone position list.
getSeatApplicationList	Host/Administrator gets the request list of users applying for the

	microphone in the room.
lockSeatByAdmin	Lock Microphone Position (only administrators or group owners can invoke, including position lock, audio status lock, and video status lock).
takeSeat	Apply to join the microphone (no need to apply in free speaking mode).
leaveSeat	Apply to leave the microphone (no need to apply in free speaking mode).
moveToSeat	Disconnect Mic
takeUserOnSeatByAdmin	Host/Administrator invites users to go on stage.
kickUserOffSeatByAdmin	Host/Administrator removes users from the microphone.

Signaling Management

API	Description
cancelRequest	Cancel Request.
responseRemoteRequest	Reply Request.

Sending Message

API	Description
disableSendingMessageByAdmin	Disable the remote user's ability to send text messages (only administrators or group owners can call this).
disableSendingMessageForAllUser	Disable all users' ability to send text messages (only administrators or group owners can call this).

Advanced Features

API	Description
setBeautyLevel	Set beauty filter effect level
setWhitenessLevel	Set brightening filter effect level
getExtension	Get plugins
getMediaDeviceManager	Get device management class

Debugging related

API	Description
callExperimentalAPI	Calls an experimental API.

TUIRoomObserver Callback Event

TUIRoomObserver is the callback event class corresponding to TUIRoomEngine. You can use this callback to listen to the events you need.

TUIRoomObserver

TUIRoomObserver

Error Callback

API	Description
onError	Error event callback

Callback for login status event

API	Description
onKickedOffLine	Kicked offline by another client during terminal login.
onUserSigExpired	User credentials timeout event.

Room Event Callback

API	Description
onRoomNameChanged	Room name change event.
onAllUserMicrophoneDisableChanged	All users' microphones in the room have been disabled event.
onAllUserCameraDisableChanged	All users' cameras in the room have been disabled event.
onScreenShareForAllUserDisableChanged	All users' screen sharing in the room has been disabled event.
onSendMessageForAllUserDisableChanged	All users' ability to send text messages in the room has been disabled event.

onKickedOutOfRoom	Removed from the room event.
onRoomDismissed	Room dissolved event.
onRoomSeatModeChanged	Room Microphone Mode Change
onRoomUserCountChanged	Room population changed

User Event Callback Inside the Room

API	Description
onRemoteUserEnterRoom	Remote user entered room event.
onRemoteUserLeaveRoom	Remote user left room event.
onUserRoleChanged	User role changed event.
onUserVideoStateChanged	User video status changed event.
onUserAudioStateChanged	Event of User Audio Status Changed
onUserVoiceVolumeChanged	User volume changed event.
onSendMessageForUserDisableChanged	User text message sending capability changed event.
onUserNetworkQualityChanged	User network status changed event.
onUserScreenCaptureStopped	Screen sharing ended.

Room microphone position event callback

API	Description
onRoomMaxSeatCountChanged	Maximum number of microphones in room changed event (only in meeting type rooms).
onSeatListChanged	Microphone position list changed event.
onKickedOffSeat	User removed from microphone event received.

Request signaling event callback

API	Description
onRequestReceived	Request message event received.

<code>onRequestCancelled</code>	Request cancellation event received.
<code>onRequestProcessed</code>	Received request handled by another administrator/homeowner event

TUIRoomEngine

Last updated : 2025-04-17 10:27:23

TUIRoomEngine APIs

TUIRoomEngine API is a no UI interface for multi-person audio and video rooms, you can use these APIs to perform custom encapsulation based on your business needs.

sharedInstance

Create TUIRoomEngine Instance.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static TUIRoomEngine sharedInstance()
```

return: TUIRoomEngine instance.

destroySharedInstance

Destroy TUIRoomEngine instance.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void destroySharedInstance()
```

login

Log in to roomEngine interface, you need to initialize user information first, then you can enter the room and perform a series of operations.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static Future<TUIActionCallback> login(int sdkAppId,  
                                       String userId,  
                                       String userSig)
```

Parameter:

Parameters	Type	Meaning
sdkAppId	int	Get sdkAppId information from Application Information

userId	String	User ID
userSig	String	userSig signature. For the calculation method of userSig, please refer to UserSig related

logout

Logout Interface, which includes proactively leaving the room and destroying resources.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static Future<TUIActionCallback> logout()
```

setSelfInfo

Set Local Username and Avatar.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static Future<TUIActionCallback> setSelfInfo(String userName, String avatarURL)
```

Parameter:

Parameters	Type	Meaning
userName	String	Username
avatarUrl	String	User's profile photo

setLoginUserInfo

Set Login User Information.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static Future<TUIActionCallback> setLoginUserInfo(TUILoginUserInfo userInfo)
```

Parameter:

Parameters	Type	Meaning
userInfo	TUILoginUserInfo	User Information

getSelfInfo

Obtain Basic Information of Local User Login.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
static TUILoginUserInfo getSelfInfo()
```

return: User log in to information.

addObserver

Add TUIRoomEngine event callback.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void addObserver(TUIRoomObserver observer)
```

Parameter:

Parameters	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine event callback

removeObserver

Remove TUIRoomEngine event callback.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void removeObserver(TUIRoomObserver observer)
```

Parameters	Type	Meaning
observer	TUIRoomObserver	TUIRoomEngine event callback

createRoom

The host creates a room, and the user who calls createRoom is the owner of the room. When creating a room, you can set the room ID, room name, and whether the room allows users to initiate audio and video, send messages, and other features.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> createRoom(TUIRoomInfo roomInfo)
```

Parameter:

Parameters	Type	Meaning
------------	------	---------

roomInfo	TUIRoomInfo	Room Basic Information
----------	-----------------------------	------------------------

destroyRoom

Destroy room interface. The room must be destroyed by the owner. After destroying the room, it cannot be entered.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> destroyRoom()
```

enterRoom

Enter room interface.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<TUIRoomInfo>> enterRoom(String roomId, {TUIRoomType roomType
```

Parameter:

Parameters	Type	Meaning
roomId	String	Room number, string type
roomType	TUIRoomType	Room type
options	TUIEnterRoomOptions	Optional parameters for entering the room

exitRoom

Leave room interface, users can leave the room via exitRoom after executing enterRoom.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> exitRoom(bool syncWaiting)
```

Parameter:

Parameters	Type	Meaning
syncWaiting	bool	Whether to exit the room synchronously

connectOtherRoom

Connect to Other Rooms.

Note:

Used to apply for a cross-room mic connection in live streaming scenarios.

```
TUIRequest connectOtherRoom(String roomId,
                             String userId,
                             int timeout,
                             TUIRequestCallback? requestCallback)
```

Parameter:

Parameters	Type	Meaning
roomId	String	Room ID
userId	String	User ID
timeout	int	Time
callback	TUIRequestCallback	Callback for connecting to other rooms request

Return:Request Body

disconnectOtherRoom

Disconnect from Other Rooms.

Note:

Used to disconnect cross-room mic connections in live streaming scenarios.

```
Future<TUIActionCallback> disconnectOtherRoom()
```

fetchRoomInfo

Obtain room information.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<TUIRoomInfo>> fetchRoomInfo()
```

updateRoomNameByAdmin

Update Room Name.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> updateRoomNameByAdmin(String roomName)
```

Parameter:

Parameters	Type	Meaning
roomName	String	Room Name

updateRoomSeatModeByAdmin

Set room microphone mode (only administrators or group owners can call this).

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> updateRoomSeatModeByAdmin(TUISeatMode mode)
```

Parameters	Type	Meaning
mode	TUISeatMode	Room mode

setLocalVideoView

Set the view controls for local user video rendering.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void setLocalVideoView(int viewId)
```

Parameter:

Parameters	Type	Meaning
viewId	int	int64 type value of the pending view pointer, this viewId can be converted to the corresponding native platform's view, and the video footage will be rendered on this view

openLocalCamera

Turn on the local camera and start video stream acquisition.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> openLocalCamera(bool isFront,
```

```
TUIVideoQuality quality)
```

Parameter:

Parameters	Type	Meaning
isFront	bool	Whether to use the front-facing camera
quality	TUIVideoQuality	Video Quality

closeLocalCamera

Turns the local camera off.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void closeLocalCamera()
```

updateVideoQuality

Set local video parameters.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void updateVideoQuality(TUIVideoQuality quality)
```

Parameter:

Parameters	Type	Meaning
quality	TUIVideoQuality	Video Quality

updateVideoQualityEx

Set local video encoder parameters.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void updateVideoQualityEx(  
    TUIVideoStreamType streamType, TUIRoomVideoEncoderParams params);
```

Parameters	Type	Meaning
streamType	TUIVideoStreamType	Video stream type

params	TUIRoomVideoEncoderParams	Video Encoder Parameters
--------	---------------------------	--------------------------

setVideoResolutionMode

Set video encoder resolution mode (landscape resolution or portrait resolution).

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void setVideoResolutionMode(
    TUIVideoStreamType streamType, TUIResolutionMode resolutionMode);
```

Parameters	Type	Meaning
streamType	TUIVideoStreamType	Video stream type
resolutionMode	TUIResolutionMode	Resolution Mode

enableGravitySensor

Enable gravity sensing mode.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void enableGravitySensor(bool enable);
```

Parameters	Type	Meaning
enable	bool	Whether Enabled

startPushLocalVideo

Start pushing the local video stream to the remote.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void startPushLocalVideo()
```

stopPushLocalVideo

Stop pushing local video streams to the remote.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void stopPushLocalVideo()
```

startScreenSharing

Starting Screen Sharing

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<void> startScreenSharing({String appGroup = ''})
```

stopScreenSharing

End Screen Sharing

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<void> stopScreenSharing()
```

openLocalMicrophone

Enable the local mic.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> openLocalMicrophone(TUIAudioQuality quality)
```

Parameters	Type	Meaning
quality	TUIAudioQuality	Audio Quality

closeLocalMicrophone

Disable the local mic.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void closeLocalMicrophone()
```

updateAudioQuality

Update local audio encoding quality settings.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void updateAudioQuality(TUIAudioQuality quality)
```

Parameter:

Parameters	Type	Meaning
quality	TUIAudioQuality	Audio Quality

muteLocalAudio

Stop pushing local audio streams to the remote.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> muteLocalAudio()
```

unMuteLocalAudio

Start pushing the local audio stream to the remote.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> unMuteLocalAudio()
```

setRemoteVideoView

Set the view controls for remote user video rendering.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void setRemoteVideoView(String userId,
                        TUIVideoStreamType streamType,
                        int viewId)
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
streamType	TUIVideoStreamType	User stream type
viewId	int	int64 type value of the pending view pointer, this viewId can be converted to the corresponding native platform's view, and the video

		footage will be rendered on this view
--	--	---------------------------------------

startPlayRemoteVideo

Start playing the remote user's video stream.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void startPlayRemoteVideo(String userId,
                          TUIVideoStreamType streamType,
                          TUIPlayCallback? callback)
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
streamType	TUIVideoStreamType	User stream type
callback	TUIPlayCallback?	Playback result callback

stopPlayRemoteVideo

Stop playing the remote user's video stream.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void stopPlayRemoteVideo(String userId,
                          TUIVideoStreamType streamType)
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
streamType	TUIVideoStreamType	User stream type

muteRemoteAudioStream

Mute remote user.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void muteRemoteAudioStream(String userId, boolean isMute);
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
isMute	bool	Whether to mute

getUserList

Get the current room user list. Note that the maximum number of users that can be pulled at one time is 100.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<TUIUserListResult>> getUserList(int nextSequence)
```

Parameter:

Parameters	Type	Meaning
nextSequence	int	Pagination pull flag. Fill in 0 for the first pull. If nextSeq is not zero in the callback, pagination is needed. Pass in nextSeq to pull again until nextSeq is zero in the callback

getUserInfo

Get the detailed information of users.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<TUIUserInfo>> getUserInfo(String userId)
```

Parameter:

Parameters	Type	Meaning
userId	String	Get detailed information of this user based on userId

changeUserRole

Change the user's role. Only administrators or group owners can call this.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> changeUserRole(String userId,  
                                         TUIRole role)
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
role	TUIRole	User Role

changeUserNameCard

Change the nickname of a user in the room.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> changeUserNameCard(String userId, String nameCard);
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID
nameCard	String	User Nickname

kickRemoteUserOutOfRoom

Remove a user from the room. Only administrators or group owners can call this.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> kickRemoteUserOutOfRoom(String userId)
```

Parameter:

Parameters	Type	Meaning
userId	String	User ID

addCategoryTagForUsers

Add tags for users. Only the room owner can call this.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> addCategoryTagForUsers(int tag, List<String> userList);
```

Parameter:

Parameters	Type	Meaning
tag	int	Mark type. Numeric type, greater than or equal to 1000. You can define it yourself.
userList	List<String>	User list

removeCategoryTagForUsers

Remove tags for users. Only the room owner can call this.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> removeCategoryTagForUsers(int tag, List<String> userList)
```

Parameter:

Parameters	Type	Meaning
tag	int	Type. Numeric type, greater than or equal to 1000. You can define it yourself.
userList	List<String>	User list

getUserListByTag

Get user information in the room based on labels

Remove tags for users. Only the room owner can call this.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIValueCallBack<TUIUserListResult>> getUserListByTag(int tag, int nextSeque
```

Parameter:

Parameters	Type	Meaning
tag	int	Type. Numeric type, greater than or equal to 1000. You can define it yourself.
nextSequence	int	Pagination pull flag. Fill in 0 for the first pull. If nextSequence is not zero in the callback, pagination is needed. Pass it in to pull again until it is zero

setCustomInfoForUser

Set customized information for members in the room

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> setCustomInfoForUser(String userId, HashMap<String, String
```

Parameter:

Parameters	Type	Meaning
userId	String	userId
customInfo	HashMap<String, String>	Customized Information

disableDeviceForAllUserByAdmin

All Users Media Device Management, only administrators or group owners can invoke.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> disableDeviceForAllUserByAdmin(TUIMediaDevice device,  
                                                         bool isDisable)
```

Parameters	Type	Meaning
device	TUIMediaDevice	Device
isDisable	bool	Whether to disable

openRemoteDeviceByAdmin

Request Remote User to Open Media Device, only administrators or group owners can invoke.

Note:

This function is only applicable to conference room type ([conference](#)).

```
TUIRequest openRemoteDeviceByAdmin(String userId,  
                                   TUIMediaDevice device,  
                                   int timeout,  
                                   TUIRequestCallback? requestCallback)
```

Parameters	Type	Meaning
userId	String	User ID
device	TUIMediaDevice	Device
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not

		perform timeout detection and will not trigger a timeout callback
requestCallback	TUIRequestCallback?	Operation result callback

closeRemoteDeviceByAdmin

Close Remote User Media Device, only administrators or group owners can invoke.

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> closeRemoteDeviceByAdmin(String userId,
                                                    TUIMediaDevice device)
```

Parameters	Type	Meaning
userId	String	User ID
device	TUIMediaDevice	Device

applyToAdminToOpenLocalDevice

All Users Media Device Management lock.

Note:

This function is only applicable to conference room type ([conference](#)).

```
TUIRequest applyToAdminToOpenLocalDevice(TUIMediaDevice device,
                                           int timeout,
                                           TUIRequestCallback? requestCallback)
```

Parameters	Type	Meaning
device	TUIMediaDevice	Device
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not perform timeout detection and will not trigger a timeout callback
callback	TUIRequestCallback?	Operation result callback

setMaxSeatCount

Set the maximum number of microphone slots. It can only be set before entering the room or when creating the room.

```
Future<TUIActionCallback> setMaxSeatCount(int maxSeatCount)
```

Parameters	Type	Meaning
maxSeatCount	int	Maximum Number of Microphones

getSeatApplicationList

Host/Administrator retrieves the list of requests from users applying to speak in the room.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<List<TUIRequest>>> getSeatApplicationList ();
```

lockSeatByAdmin

Lock microphone position (includes location lock, audio status lock, video status lock).

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> lockSeatByAdmin(int seatIndex,  
                                          TUISeatLockParams lockParams)
```

Parameters	Type	Meaning
seatIndex	int	Microphone slot number
lockParams	TUISeatLockParams	Mute Microphone parameter

getSeatList

Get the microphone position list.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIValueCallBack<List<TUISeatInfo>>> getSeatList ();
```

takeSeat

Local Microphone On.

Note:

Meeting scenario:[applyToSpeak](#) mode requires an application to the host or administrator before taking the mic. Other modes do not support taking the mic.

Live streaming scenario: [freeToSpeak](#) mode allows free mic access. After taking the mic, open the microphone to speak; [applySpeakAfterTakingSeat](#) mode requires an application to the host or administrator before taking the mic. Other modes do not support taking the mic.

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
TUIRequest takeSeat(int seatIndex,
                    int timeout,
                    TUIRequestCallback? requestCallback)
```

Parameter:

Parameters	Type	Meaning
seatIndex	int	Microphone slot number
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not perform timeout detection and will not trigger a timeout callback
requestCallback	TUIRequestCallback?	Invoke interface callback to notify the request status

Return: Request Body

leaveSeat

Local Microphone Off.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> leaveSeat()
```

moveToSeat

Move Microphone.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> moveToSeat(int targetSeatIndex);
```

Parameters	Type	Meaning
seatIndex	int	Microphone slot number

takeUserOnSeatByAdmin

Host/Administrator invites users to go on stage.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
TUIRequest takeUserOnSeatByAdmin(int seatIndex,  
                                String userId,  
                                int timeout,  
                                TUIRequestCallback? requestCallback)
```

Parameters	Type	Meaning
seatIndex	int	Microphone slot number
userId	String	User ID
timeout	int	Timeout period, unit: seconds. If set to 0, the SDK will not perform timeout detection and will not trigger a timeout callback
requestCallback	TUIRequestCallback?	Invoke interface callback to notify the request status

Return: Request Body

kickUserOffSeatByAdmin

Host/Administrator removes users from the mic.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> kickUserOffSeatByAdmin(int seatIndex,  
                                                String userId)
```

Parameters	Type	Meaning
seatIndex	int	Microphone slot number
userId	String	User ID

disableSendingMessageByAdmin

Disable the remote user's ability to send text messages (only administrators or group owners can call this).

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> disableSendingMessageByAdmin(String userId,
```

```
bool isDisable)
```

Parameters	Type	Meaning
userId	String	User ID
isDisable	bool	Whether to disable

disableSendingMessageForAllUser

Disable all users' ability to send text messages (only administrators or group owners can call this).

Note:

This function is only applicable to conference room type ([conference](#)).

```
Future<TUIActionCallback> disableSendingMessageForAllUser(bool isDisable)
```

Parameters	Type	Meaning
isDisable	bool	Whether to disable

cancelRequest

Cancel the already sent request.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> cancelRequest(String requestId)
```

Parameter:

Parameters	Type	Meaning
requestId	String	Request ID

responseRemoteRequest

Respond to the remote user's request.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
Future<TUIActionCallback> responseRemoteRequest(String requestId,  
                                                bool agree)
```

Parameter:

Parameters	Type	Meaning
------------	------	---------

requestId	String	Request ID
agree	bool	Do you agree?

callExperimentalAPI

Calls an experimental API.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void callExperimentalAPI(String jsonStr);
```

Parameters	Type	Meaning
jsonStr	String	Interface Information

setBeautyLevel

Set beauty filter effect level

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void setBeautyLevel(int beautyStyle, int beautyLevel);
```

Parameters	Type	Meaning
beautyStyle	int	Beauty filter style
beautyLevel	int	Beauty filter effect level

setWhitenessLevel

Set brightening filter effect level

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
void setWhitenessLevel(int whitenessLevel);
```

Parameters	Type	Meaning
whitenessLevel	int	Brightening filter effect level

getExtension

Get plugins

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
dynamic getExtension(TUIExtensionType extensionType);
```

Parameters	Type	Meaning
extensionType	TUIExtensionType	Plugin Type

getMediaDeviceManager

Get device management class

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
TUIRoomDeviceManager getMediaDeviceManager();
```

TUIRoomEngineObserver

Last updated : 2025-04-17 10:28:55

TUIRoomEngine event callback

onError

Triggered when an error event occurs, indicating that the SDK encounters an irrecoverable error, such as a failure of entering a room or a device startup failure.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnError onError = (TUIError errorCode, String message) {}
```

Parameters	Type	Description
errorCode	TUIError	Error Codes onLiveRoomInfoChanged
message	String	Error Message

onKickedOffLine

Triggered when the user is kicked offline.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnKickedOffLine onKickedOffLine = (String message) {}
```

Parameters	Type	Description
message	String	Description of removed from service

onUserSigExpired

`userSig` expiration event, triggered when the user's credential expires.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserSigExpired onUserSigExpired = () {}
```

onRoomNameChanged

Triggered when the room name changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRoomNameChanged onRoomNameChanged = (String roomId, String roomName) {}
```

Parameters	Type	Description
roomId	String	Room ID
roomName	String	Room Name

onAllUserMicrophoneDisableChanged

Triggered when the microphone disable status of all users changes.

Note:

This function is only applicable to conference room type ([conference](#)).

```
OnAllUserMicrophoneDisableChanged onAllUserMicrophoneDisableChanged = (String roomId
```

Parameters	Type	Description
roomId	String	Room ID
isDisable	bool	Whether disabled or not

onAllUserCameraDisableChanged

Triggered when the camera disable status of all users changes.

Note:

This function is only applicable to conference room type ([conference](#)).

```
OnAllUserCameraDisableChanged onAllUserCameraDisableChanged = (String roomId, bool
```

Parameters	Type	Description
roomId	String	Room ID
isDisable	bool	Whether disabled or not

onSendMessageForAllUserDisableChanged

Triggered when the message sending permission of all users changes.

Note:

This function is only applicable to conference room type ([conference](#)).

```
OnSendMessageForAllUserDisableChanged onSendMessageForAllUserDisableChanged = (String roomId, Boolean isDisable)
```

Parameters	Type	Description
roomId	String	Room ID
isDisable	bool	Whether disabled or not

onScreenShareForAllUserDisableChanged

Triggered when the screen sharing permissions of all users change.

Note:

This function is only applicable to conference room type ([conference](#)).

```
OnScreenShareForAllUserDisableChanged onScreenShareForAllUserDisableChanged = (String roomId, Boolean isDisable)
```

Parameters	Type	Description
roomId	String	Room ID
isDisable	bool	Whether disabled or not

onRoomDismissed

Triggered when the room is dissolved.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRoomDismissed onRoomDismissed = (String roomId, TUIRoomDismissedReason reason) {}
```

Parameters	Type	Description
roomId	String	Room ID
reason	TUIRoomDismissedReason	Dissolution reason

onKickedOutOfRoom

Triggered when the user is kicked out of the room by the host/administrator.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnKickedOutOfRoom onKickedOutOfRoom = (String roomId, String message) {}
```

Parameters	Type	Description
roomId	String	Room ID
message	String	Description of removed from room

onRoomSeatModeChanged

Triggered when the microphone mode of the room changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRoomSeatModeChanged onRoomSeatModeChanged =(String roomId, TUISeatMode seatMode)
```

Parameters	Type	Description
roomId	String	Room ID
seatMode	TUISeatMode	Microphone Mode

onRoomUserCountChanged

Triggered when the number of people in the room changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRoomUserCountChanged onRoomUserCountChanged =(String roomId, int userCount) {};
```

Parameters	Type	Description
roomId	String	Room ID
userCount	int	Room Occupancy

onRemoteUserEnterRoom

Triggered when a remote user enters the room.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRemoteUserEnterRoom onRemoteUserEnterRoom = (String roomId, TUIUserInfo userInfo)
```

Parameters	Type	Description
roomId	String	Room ID
userInfo	TUIUserInfo	User Information

onRemoteUserLeaveRoom

Triggered when a remote user leaves the room.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRemoteUserLeaveRoom onRemoteUserLeaveRoom = (String roomId, TUIUserInfo userInfo)
```

Parameters	Type	Description
roomId	String	Room ID
userInfo	TUIUserInfo	User Information

onUserInfoChanged

Triggered when user information in the room changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserInfoChanged onUserInfoChanged = (TUIUserInfo userInfo, List<TUIUserInfoModify
```

Parameters	Type	Description
userInfo	TUIUserInfo	User Information
modifyFlagList	List<TUIUserInfoModifyFlag>	TUIUserInfo change flag list

onUserVideoStateChanged

Triggered when the user's video status changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserVideoStateChanged onUserVideoStateChanged = (String userId, TUIVideoStreamType
```

Parameters	Type	Description
userId	String	User ID
streamType	TUIVideoStreamType	Video stream type
hasVideo	bool	Has video stream
reason	TUIChangeReason	Reason for video stream change

onUserAudioStateChanged

Triggered when the user's audio status changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserAudioStateChanged onUserAudioStateChanged = (String userId, bool hasAudio, TU
```

Parameters	Type	Description
userId	String	User ID
hasAudio	bool	Is there an audio stream
reason	TUIChangeReason	Reason for video stream change

onUserVoiceVolumeChanged

Triggered when the user's volume changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserVoiceVolumeChanged onUserVoiceVolumeChanged = (Map<String, int> volumeMap) {}
```

Parameters	Type	Description
volumeMap	Map	User volume map key: userId value: Used to carry the volume levels of all speaking users. Range: 0 - 100

onSendMessageForUserDisableChanged

Triggered when the user's message sending permissions change.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnSendMessageForUserDisableChanged onSendMessageForUserDisableChanged = (String roomId, String userId, bool isDisable)
```

Parameters	Type	Description
roomId	String	Room ID
userId	String	User ID
isDisable	bool	Whether sending text messages is forbidden

onUserNetworkQualityChanged

Triggered when the user's network quality changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserNetworkQualityChanged onUserNetworkQualityChanged = (Map<String, TUINetwork> networkMap)
```

Parameters	Type	Description
networkMap	Map	User network status map key : userId value: Network condition

onUserScreenCaptureStopped

Triggered when the user's screen sharing stops.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnUserScreenCaptureStopped onUserScreenCaptureStopped = (int reason) {}
```

Parameters	Type	Description
reason	int	Stop reason 0: User stopped voluntarily 1: Stop due to screen window being closed 2: Indicates screen sharing display state change (e.g., interface unplugged, projection mode change, etc.)

onRoomMaxSeatCountChanged

Maximum number of microphones in room changed event (only in meeting type rooms).

```
OnRoomMaxSeatCountChanged onRoomMaxSeatCountChanged = (String roomId, int maxSeatCo
```

Parameters	Type	Description
roomId	String	Room ID
maxSeatCount	int	Maximum number of microphone positions in the room

onSeatListChanged

Triggered when the microphone list changes.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnSeatListChanged onSeatListChanged = (List<TUISeatInfo> seatList, List<TUISeatInfo
```

Parameters	Type	Description
seatList	List< TUISeatInfo >	The latest list of users currently on the microphone, including newly joined users
seatedList	List< TUISeatInfo >	List of newly joined users on the microphone
leftList	List< TUISeatInfo >	List of users who left the microphone

onKickedOffSeat

Triggered when the user is kicked off the microphone.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnKickedOffSeat onKickedOffSeat = (int seatIndex, TUIUserInfo operateUser) {}
```

Parameters	Type	Description
seatIndex	int	Microphone slot number
operateUser	TUIUserInfo	Operator's information

onRequestReceived

Triggered when a request from another user is received.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRequestReceived onRequestReceived = (TUIRequest request) {}
```

Parameters	Type	Description
request	TUIRequest	Request content

onRequestCancelled

Triggered when another user cancels the request.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRequestCancelled onRequestCancelled = (TUIRequest request, TUIUserInfo operateUser)
```

Parameters	Type	Description
request	TUIRequest	Request Information
operateUser	TUIUserInfo	User information of the canceled signal

onRequestProcessed

Triggered when the request is handled by another **admin/host**.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

```
OnRequestProcessed onRequestProcessed = (TUIRequest request, TUIUserInfo operateUser)
```

Parameters	Type	Description
request	TUIRequest	Request Information
operateUser	TUIUserInfo	User information of the canceled signal

TUIRoomDeviceManager

Last updated : 2025-04-18 16:42:52

TUIRoomDeviceManager

TUIRoomDeviceManager

Function List	Description
isFrontCamera	Determine whether the current camera is front-facing (only for mobile terminal)
switchCamera	Switch between front and rear cameras (only for mobile terminal)
isAutoFocusEnabled	Check if automatic face position recognition is supported (only for mobile terminal)
enableCameraAutoFocus	Enable autofocus feature (only for mobile terminal)
enableCameraTorch	Turn the flash on/off, also known as flashlight mode (only for mobile terminal)
setAudioRoute	Set audio routing (only for mobile terminal)

Enumeration Types

Enumeration Types	Description
AudioRoute	Audio routing (i.e., sound playback mode)

isFrontCamera

isFrontCamera

Determine whether the current camera is front-facing (only for mobile terminal)

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

switchCamera

switchCamera

void switchCamera	(boolean frontCamera)
-------------------	-----------------------

Switch between front and rear cameras (only for mobile terminal)

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

isAutoFocusEnabled

isAutoFocusEnabled

Check if automatic face position recognition is supported (only for mobile terminal)

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

enableCameraAutoFocus

enableCameraAutoFocus

void enableCameraAutoFocus	(boolean enabled)
----------------------------	-------------------

Enable autofocus feature (only for mobile terminal)

Once enabled, the SDK will automatically detect the face position in the frame and keep the camera focus on the face position.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

enableCameraTorch

enableCameraTorch

void enableCameraTorch	(boolean enable)
------------------------	------------------

Turn the flash on/off, also known as flashlight mode (only for mobile terminal)

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

setAudioRoute

setAudioRoute

void setAudioRoute	(TUIAudioRoute route)
--------------------	--

Set audio routing (only for mobile terminal)

A phone has two audio playback devices: one is the earpiece at the top of the phone, and the other is the stereo speaker at the bottom of the phone.

When the audio routing is set to the earpiece, the sound is relatively low, and you need to bring your ear close to hear it clearly, ensuring better privacy. It is suitable for phone calls.

When the audio routing is set to the speaker, the sound is relatively loud. You can hear it clearly without holding the phone to your face, thus enabling the "hands-free" feature.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

TUIAudioRoute

TUIAudioRoute

Audio routing (i.e., sound playback mode)

Audio routing, that is, whether the sound is played through the phone's speaker or earpiece. Therefore, this interface is only applicable to mobile devices.

A mobile phone has two speakers: one is the earpiece located at the top of the phone, and the other is the stereo speaker located at the bottom.

When the audio routing is set to the earpiece, the sound is relatively low, and you need to bring your ear close to hear it clearly, ensuring better privacy. It is suitable for phone calls.

When the audio routing is set to the speaker, the sound is relatively loud. You can hear it clearly without holding the phone to your face, thus enabling the "hands-free" feature.

Enumeration	Value	Description
speakerphone	0	Speakerphone: Play sound through the speaker (i.e., "hands-free"), which is located at the bottom of the phone. The volume tends to be loud, suitable for playing music out loud.
earpiece	1	Earpiece: Plays through the earpiece, the earpiece is located at the top of the phone, the sound is small, suitable for privacy-protected call scenarios.

Note:

This function applies to conference room type and live room type ([conference & livingRoom](#)).

TUILiveListManager

Last updated : 2025-04-18 16:43:41

Copyright (c) 2024 Tencent. All rights reserved.
Module: TUILiveListManager @ TUIKitEngine
Functions: Relevant APIs for live room list. The functions on this page only support live room type ([TUIRoomTypeLive](#)).

TUILiveListManager

TUILiveListManager

Function List	Description
onLiveInfoChanged	Live information change callback
addObserver	Add event callback
removeObserver	Remove event callback
setLiveInfo	Modify live information
getLiveInfo	Retrieve live information
fetchLiveList	Search live stream list

Structure Type

Function List	Description
TUILiveInfo	Live information
TUILiveListResult	Live stream list pull results

Enumeration Types

Enumeration Types	Description

TUILiveModifyFlag	Modify the flag bit in the live streaming room
-------------------	--

onLiveInfoChanged

Live information change callback

```
OnLiveInfoChanged onLiveInfoChanged = (TUILiveInfo liveInfo, List<TUILiveModifyFlag
```

Parameter	Description
liveInfo	Live room information
modifyFlagList	Change value flag bit list

addObserver

Add event callback

```
void addObserver(TUILiveListObserver observer);
```

Parameter	Description
observer	Listening instances

removeObserver

Remove event callback

```
void removeObserver(TUILiveListObserver observer);
```

Parameter	Description
observer	Listening instances

setLiveInfo

Modify live information

```
Future<TUIActionCallback> setLiveInfo(String roomId,
    {String? coverUrl,
    String? backgroundUrl,
    List<int>? categoryList,
    bool? isPublicVisible,
    int? activityStatus});
```

Parameter	Description
roomId	room ID
coverUrl	Profile photo URL
categoryList	Live room category tag
isPublicVisible	Is public
activityStatus	Live streaming room active status: user-customized tag

getLiveInfo

Retrieve live information

```
Future<TUIValueCallBack<TUILiveInfo>> getLiveInfo(String roomId);
```

Parameter	Description
roomId	room ID

fetchLiveList

Search live stream list

```
Future<TUIValueCallBack<TUILiveListResult>> fetchLiveList(String cursor, int count)
```

Parameter	Description
cursor	list index
count	The number of pulls each time

TUILiveInfo

Live information

Enumeration Types	Description
roomInfo	Read-only Room Information
coverUrl	Live room cover
backgroundUrl	Live room background
categoryList	Live room category tag
isPublicVisible	Whether the live room is public
activityStatus	Live streaming room active status: user-customized tag
viewCount	Cumulative Viewing Count

TUILiveListResult

Enumeration Types	Description
cursor	list index
liveInfoList	Live room information list

TUILiveModifyFlag

Modify the flag bit in the live streaming room

Error Example	Value	Description
activityStatus	0x0100	ActivityStatus: Live streaming room active status, support customizing
coverUrl	0x0200	CoverUrl: Live room cover
category	0x0400	Category: Live room category
publish	0x2000	Publish: Live room public marking
backgroundUrl	0x40000	BackgroundUrl: Live room background

Type Definition

Last updated : 2025-04-17 10:35:09

Enumeration definitions

TUIRoomDefine

Type	Description
TUIRoomType	Room type
TUISeatMode	Microphone Mode
TUIMediaDevice	Media Device Type Inside the Room
TUIRole	Role Type Inside the Room
TUIVideoQuality	Video Quality
TUIAudioQuality	Audio Quality
TUIVideoStreamType	Video stream type
TUIChangeReason	Change Reason (Reason for user audio/video status change: self-initiated or modified by room owner or admin)
TUICaptureSourceType	Screen Sharing Capture Source Type
TUIRequestAction	Request Type
TUIResolutionMode	Resolution Mode (Landscape or Portrait)
TUIUserInfoModifyFlag	User Information Modification Type

TUICommonDefine

Type	Description
TUIError	Error Codes
TUINetworkQuality	Network quality
TUIExtensionType	Plugin Type

Common Structure

TUIRoomDefine

Type	Description
TUIRoomInfo	Room Information
TUILoginUserInfo	User log in to Information
TUIUserInfo	Inside the Room User Information
TUISeatInfo	Seat Information Inside the Room
TUISeatLockParams	Parameters for Locking the Mic Position
TUIUserVoiceVolume	User Volume Inside the Room
TUIRequest	Signaling Request
TUIActionCallback	User Operation Callback
TUIPlayCallback	Video Playback Callback
TUIRequestCallback	User Request Callback
TUIUserListResult	User List Information
TUIUserVoiceVolume	User Volume Information
TUIValueCallBack<T>	Callback with Return Value (T)
TUIRoomVideoEncoderParams	Video Encoder Parameters
TUIEnterRoomOptions	Room entry parameters

TUICommonDefine

Type	Description
TUINetwork	Network quality information
TUIMessage	Message
TUIImageBuffer	Image information

TUIRoomType

Room type

Enumeration	Value	Description
-------------	-------	-------------

conference	1	Meeting Type Room, suitable for meetings and educational contexts. In this room, users can choose different modes such as free speech mode, apply for speech, and speaking on the microphone
livingRoom	2	Live Streaming Type Room, suitable for live streaming scenarios. In this room, users can speak freely, and the room uses a microphone position control mode, where each microphone position has a number

TUISeatMode

Microphone Mode

Enumeration	Value	Description
freeToTake	1	Freely Join Microphone Mode, the audience can freely join the mic without application
applyToTake	2	Apply for Microphone Mode, the audience needs approval from the homeowner or administrator to join the mic

TUIMediaDevice

Media Device Type Inside the Room

Enumeration	Value	Description
microphone	1	Microphone
camera	2	Camera
screen	3	Screen Sharing

TUIRole

Role Type Inside the Room

Enumeration	Value	Description
roomOwner	0	Homeowner. Generally refers to the creator of the room and the owner of the highest permissions in the room
administrator	1	Room administrator
generalUser	2	Ordinary member in the room

TUIVideoQuality

Video Quality

Enumeration	Value	Description
videoQuality_360P	1	Low-definition 360p
videoQuality_540P	2	Standard Definition 540p
videoQuality_720P	3	HD 720p
videoQuality_1080P	4	Ultra HD 1080p

TUIAudioQuality

Audio Quality

Enumeration	Value	Description
audioProfileSpeech	0	Voice Mode
audioProfileDefault	1	Default Mode
audioProfileMusic	2	Music Mode

TUIVideoStreamType

Video stream type

Enumeration	Value	Description
cameraStream	0	HD Camera Video Stream
screenStream	1	Screen Share Video Stream
cameraStreamLow	2	Low-definition Camera Video Stream

TUIChangeReason

Change Reason (Reason for user audio/video status change: self-initiated or modified by room owner or admin)

Enumeration	Value	Description
changedBySelf	0	User Operation
changedByAdmin	1	Operation by Homeowner or Administrator

TUICaptureSourceType

Screen Sharing Capture Source Type

Enumeration	Value	Description
unknown	-1	Undefined
window	0	Window
screen	1	Screen

TUIRequestAction

Request Type

Enumeration	Value	Description
invalidAction	0	Invalid request
requestToOpenRemoteCamera	1	Request remote user to turn on the camera
requestToOpenRemoteMicrophone	2	Request remote user to turn on the microphone
requestToConnectOtherRoom	3	Request to connect to another room
requestToTakeSeat	4	Request to take the mic
requestRemoteUserOnSeat	5	Request remote user to take the mic
applyToAdminToOpenLocalCamera	6	Request the administrator to turn on the local camera
applyToAdminToOpenLocalMicrophone	7	Request the administrator to turn on the local microphone

TUIResolutionMode

Enumeration	Value	Description
landscape	0	Landscape
portrait	1	Portrait

TUIUserInfoModifyFlag

Enumeration	Value	Description
none	0x00	Empty, indicating no change
userRole	0x01 <<	User Role

	0	
nameCard	0x01 << 1	Nickname in the Room

TUIError

Error Codes

Enumeration	Value	Description
success	0	Operation successful
errFailed	-1	A common error not yet classified
errFreqLimit	-2	Request was frequency limited, please try again later
errRepeatOperation	-3	Duplicate operation, please check if your API call is repeated
errSDKAppIDNotFound	-1000	SDKAppID not found, please confirm application information in the Console of TRTC
errInvalidParameter	-1001	An invalid parameter was passed in during API calling
errSdkNotInitialized	-1002	SDK not initialized
errPermissionDenied	-1003	No operational permissions
errRequirePayment	-1004	This feature requires an additional package
errCameraStartFailed	-1100	Failed to turn on the camera
errCameraNotAuthorized	-1101	Camera not authorized
errCameraOccupy	-1102	Camera is being used
errCameraDeviceEmpty	-1103	No camera device available
errMicrophoneStartFailed	-1104	Failed to open microphone
errMicrophoneNotAuthorized	-1105	Microphone not authorized
errMicrophoneOccupy	-1106	Microphone is occupied
errMicrophoneDeviceEmpty	-1107	No microphone device available

errGetScreenSharingTargetFailed	-1108	Failed to get screen sharing objects
errStartScreenSharingFailed	-1109	Failed to enable screen sharing
errRoomIdNotExist	-2100	The room does not exist when entering, it may have been dissolved
errOperationInvalidBeforeEnterRoom	-2101	You need to enter the room to use this feature
errExitNotSupportedForRoomOwner	-2102	Homeowner cannot perform check-out procedure, Conference room type: transfer homeowner first, then check out. LivingRoom room type: homeowner can only dissolve the room
errOperationNotSupportedInCurrentRoomType	-2103	This operation is not supported under the current room type
errRoomIdInvalid	-2105	Illegal Room ID creation, Custom ID must be printable ASCII characters (0x20-0x7e), up to 48 bytes
errRoomIdOccupied	-2106	Room ID is already in use, please choose another room ID
errRoomNameInvalid	-2107	Invalid room name, the maximum length is 30 bytes, character encoding must be UTF-8, if it includes Chinese characters
errAlreadyInOtherRoom	-2108	The current user is already in another room and needs to check out before joining a new room A single RoomEngine instance only supports a user entering one room. To enter a different room, please check out or use a new RoomEngine instance
errUserNotExist	-2200	The user does not exist.
errUserNotEntered	-2201	User is not in the current room
errUserNeedOwnerPermission	-2300	Homeowner privileges are required to operate
errUserNeedAdminPermission	-2301	Homeowner or administrator privileges are required to operate
errRequestNoPermission	-2310	No permission for signaling request, such as canceling an invitation not initiated by yourself

errRequestIdInvalid	-2311	Signaling request ID is invalid or has already been processed
errMaxSeatCountLimit	-2340	Maximum microphone position limit exceeds the package quota
errAlreadyInSeat	-2341	The current user is already on the microphone position
errSeatOccupied	-2342	The current microphone position is already occupied
errSeatLocked	-2343	The current microphone position is locked
errSeatIndexNotExist	-2344	Microphone position number does not exist
errUserNotInSeat	-2345	The current user is not on the microphone
errAllSeatOccupied	-2346	The number of users on the microphone is full
errOpenMicrophoneNeedSeatUnlock	-2360	The current microphone position is locked for audio
errOpenMicrophoneNeedPermissionFromAdmin	-2361	You need to request permission from the homeowner or administrator to use the microphone
errOpenCameraNeedSeatUnlock	-2370	The current microphone position is locked for video, it needs to be unlocked by the homeowner to use the camera
errOpenCameraNeedPermissionFromAdmin	-2371	You need to request permission from the homeowner or administrator to use the camera
errSendMessageDisabledForAll	-2380	All members are muted in the current room
errSendMessageDisabledForCurrent	-2381	You are muted in the current room

TUINetworkQuality

Network quality

Enumeration	Value	Description
qualityUnknown	0	Undefined
qualityExcellent	1	The present network is exceedingly good
qualityGood	2	The current network is fairly good

qualityPoor	3	Current network is average
qualityBad	4	Current network is poor
qualityVeryBad	5	Current network is very poor
qualityDown	6	Current network does not meet the minimum requirements of TRTC

TUIRoomInfo

Room Information

Field	Type	Description
roomId	String	Room ID
roomType	TUIRoomType	Room type
ownerId	String	Host ID. Defaults to the room creator (read-only)
name	String	Room Name. Defaults to Room ID
isSeatEnabled	bool	Whether to enable microphone position control
seatMode	TUISeatMode	Microphone Mode (effective only when microphone position control is enabled)
createTime	int	Room Creation Time (read-only)
memberCount	int	Number of members in the room (read-only)
maxSeatCount	int	Maximum number of microphones (can only be set before entering the room or when creating the room)
isCameraDisableForAllUser	bool	Whether to disable the camera (optional parameter when creating a room). Default value: false
isMicrophoneDisableForAllUser	bool	Whether to disable the microphone (optional parameter when creating a room). Default value: false
isMessageDisableForAllUser	bool	Whether to disable sending messages (optional parameter when creating a room). Default value: false
enableCDNStreaming	bool	Whether to enable CDN live streaming (optional parameter when creating a room, used in live rooms). Default value: false
cdnStreamDomain	String	Live streaming push domain name (optional parameter

		when creating a room, used in live rooms). Default value: empty
password	String	Room password

TUILoginUserInfo

User log in to Information

Field	Type	Meaning
userId	String	User ID
userName	String	Username
avatarUrl	String	User profile photo URL
customInfo	Map<String, String>	Customized Information

TUIUserInfo

Inside the Room User Information

Field	Type	Description
userId	String	User ID
userName	String	Username
nameCard	String	Nickname in the Room
avatarUrl	String	User profile photo URL
userRole	TUIRole	User Role Type
hasAudioStream	bool	Whether there is an audio stream. Default value: false
hasVideoStream	bool	Whether there is a video stream. Default value: false
hasScreenStream	bool	Whether there is a screen sharing stream. Default value: false
customInfo	Map<String, String>?	Room member custom information
isMessageDisabled	bool	Whether muted

TUISeatInfo

Seat Information Inside the Room

Field	Type	Description
index	int	Mic seat serial number
userId	String	User ID
userName	String	Username
nameCard	String	Nickname in the Room
avatarUrl	String	User profile photo URL
isLocked	bool	Whether the microphone position is locked. Default value: false
isVideoLocked	bool	Whether the microphone position is prohibited from opening the camera. Default value: false
isAudioLocked	bool	Whether the microphone position is prohibited from opening the microphone. Default value: false
userName	String	User Nickname
avatarUrl	String	User profile photo URL

TUISeatLockParams

Parameters for Locking the Mic Position

Field	Type	Meaning
lockSeat	bool	Lock microphone position. Default is false
lockVideo	bool	Lock microphone position camera. Default is false
lockAudio	bool	Lock microphone position microphone. Default is false

TUIUserVoiceVolume

User Volume Inside the Room

Field	Type	Description
userId	String	User ID
volume	int	Volume. Used to carry the volume size of all speaking users, value range 0-100

TUIRequest

Signaling Request

Field	Type	Description
requestId	String	Request ID
requestAction	TUIRequestAction	Request Type
userId	String	User ID
userName	String	Username
nameCard	String	Nickname in the Room
avatarUrl	String	User profile photo URL
content	String	Signaling content
timestamp	int	Timestamp
userName	String	User Nickname
avatarUrl	String	User profile photo URL

TUIActionCallback

Field	Type	Description
code	TUIError	Error Codes
message	String?	Error Message

TUIPlayCallback

Field	Type	Description
onPlaying	(String userId) {}	Playing callback
onLoading	(String userId) {}	Loading callback
onPlayError	(String userId, TUIError code, String message) {}	Playback error callback

TUIRequestCallback

--	--	--

Field	Type	Description
onAccepted	(String requestId, String userId) {}	Request accepted callback
onRejected	(String requestId, String userId, String message) {}	Request rejected callback
onCancelled	(String requestId, String userId) {}	Request canceled callback
onTimeout	(String requestId, String userId) {}	Request timeout callback
onError	(String requestId, String userId, TUIError error, String message) {}	Request error callback

TUIUserListResult

Field	Type	Description
nextSequence	int	Pagination retrieval flag. If the returned nextSequence is not zero, use the returned nextSequence to retrieve again until it returns zero
userInfoList	List<TUIUserInfo>	List of users returned by this call

TUIUserVoiceVolume

Field	Type	Description
userId	String	User ID
volume	int	User Volume Level

TUIValueCallBack<T>

Field	Type	Description
code	TUIError	Error Codes
message	String?	Error Message
data	T?	Return data, example: If T is TUIUserInfo, the data field type of TUIValueCallBack<TUIUserInfo> is TUIUserInfo

TUIRoomVideoEncoderParams

Field	Type	Description

videoResolution	TUIVideoQuality	Video Quality
resolutionMode	TUIResolutionMode	Resolution Mode
fps	int	Video Capture Frame Rate
bitrate	int	Target Video Bitrate

TUIEnterRoomOptions

Field	Type	Description
password	String	Room password

TUINetwork

Network quality information

Field	Type	Description
userId	String	User ID
quality	TUINetworkQuality	Network quality
upLoss	int	Upstream packet loss rate
downLoss	int	Downstream packet loss rate
delay	int	Network Latency

TUIExtensionType

Plugin Type

Enumeration	Value	Description
deviceManager	1	TUIRoomDeviceManager type
liveListManger	2	TUIRLiveListManager type
conferenceListManager	3	TUIConferenceListManager type

TUIMessage

Message

Field	Type	Description

messageId	String	Message ID
message	String	Message text
timestamp	int	Message timestamp
userId	String	Message sender
userName	String	Nickname of the message sender
avatarUrl	String	Profile photo of the message sender

TUImageBuffer

Image information

Field	Type	Description
buffer	String	Image data cache address
length	int	Length
width	int	Width
height	int	Height

API-Example

Voice Room (SeatGridView)

Integration and Login

Last updated : 2024-12-24 17:36:08

This document will introduce how to integrate and log in to the core component of the voice chat room, `SeatGridView`.

Integration

iOS

Android

Environment Preparation

Xcode 15 or later.

iOS 13.0 or later.

CocoaPods environment setup, [Click to view](#).

If you encounter any problems with access and use, please refer to [FAQs](#).

Activate Service

Please refer to [Activating Services \(TUILiveKit\)](#) to receive the trial version or activate the paid version.

Importing the SeatGridView

Use CocoaPods to import the component. If you encounter any issues, please refer to [Environment Preparation](#) first.

Detailed steps for importing the component are as follows:

1. Add the `pod 'SeatGridView'` dependency to your `Podfile` file.

Swift

```
target 'xxxx' do
  ...
  ...
  pod 'SeatGridView'
end
```

If you don't have a `Podfile` file, first `cd` in Terminal into the `xxxx.xcodeproj` directory, then create one using the following command:


```
pod init
```

2. In Terminal, first `cd` into the `Podfile` directory and then run the following command to install components.

```
pod install
```

If you can't install the latest version of SeatGridView, delete **Podfile.lock** and **Pods** first. Then run the following command to update the local CocoaPods repository list.

```
pod repo update
```

Then, run the following command to update the Pod version of the component library.

```
pod update
```

3. You can compile and run it first. If you encounter any issues, please refer to [FAQs](#). If you have any questions during the integration and usage, feel free to [give us feedback](#).

Environment Preparation

Android 5.0 (SDK API Level 21) or later.

Gradle v7.0 or later.

Mobile device on Android 5.0 or later.

For issues during environment configuration or compilation, please refer to [FAQ](#).

Activate Service

Please refer to [Activating Services \(TUILiveKit\)](#) to receive the trial version or activate the paid version.

Importing the SeatGridView

1. Find the `build.gradle.kts` (or `build.gradle`) file under the app directory and add the following code to include the dependency for SeatGridView component:

`build.gradle.kts`

`build.gradle`

```
api("io.trtc.uikit:voice-room-core:latest.release")
```

```
api 'io.trtc.uikit:voice-room-core:latest.release'
```

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

```
-keep class com.tencent.** { *; }  
-keep class com.trtc.uikit.livekit.voiceroomcore.** { *; }
```

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

```
// app/src/main/AndroidManifest.xml

<application
    ...

    // Add the following configuration to override the configuration in the depende
    android:allowBackup="false"
    tools:replace="android:allowBackup">
```

Log-in

iOS

Android

Add the following code to your project, which completes the login of TUI Components by calling the relevant interfaces in TUICore. This step is critical; you can only use the features provided by SeatGridView after successful login.

swift

```
//
// AppDelegate.swift
//

import TUICore
func application(_ application: UIApplication, didFinishLaunchingWithOptions launch

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

    return true
}
```

Parameter Description

Here we detail the key parameters required in the login function:

Parameters	Type	Description
------------	------	-------------

SDKAppID	int	You have already obtained it in the last step of Step 1, so it will not be elaborated here.
UserID	String	The ID of the current user, in string format, only allows letters (a-z and A-Z), digits (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in step 3 of Activate Service to encrypt information such as SDKAppID and UserID to generate a UserSig. It's a credential used for authentication purposes, allowing Tencent Cloud to identify if the current user is authorized to use the TRTC service. You can generate a temporary UserSig through the Auxiliary Tools in the console. For more information, please refer to How to Calculate and Use UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate userSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering, and once your key is leaked, attackers can steal your Tencent Cloud traffic.

Production Environment: If your project is going to be launched, please adopt the method of [Server-side Generation of UserSig](#).

Add the following code to your project, which completes the login of TUI Components by calling the relevant interfaces in TUICore. This step is critical; you can only use the features provided by SeatGridView after successful login.

Kotlin

Java

```
// Log in
TUILogin.login(applicationContext,
    1400000001, // Please replace with the SDKAppID obtained in Step 1
    "denny", // Please replace with your UserID
    "xxxxxxxxxxxx", // You can calculate a UserSig in the Console and fill it in the
    object : TUICallback() {
        override fun onSuccess() {
            Log.i(TAG, "login success")
        }

        override fun onError(errorCode: Int, errorMessage: String) {
            Log.e(TAG, "login failed, errorCode: $errorCode msg:$errorMessage")
        }
    })

// Log in
TUILogin.login(context,
    1400000001, // Please replace with the SDKAppID obtained in Step 1
```

```
"denny",          // Please replace with your UserID
"xxxxxxxxxxxx",   // You can calculate a UserSig in the Console and fill it in here
new TUICallback() {
    @Override
    public void onSuccess() {
        Log.i(TAG, "login success");
    }

    @Override
    public void onError(int errorCode, String errorMessage) {
        Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + errorMessage);
    }
});
```

Parameter Description

Here we detail the key parameters required in the login function:

Parameters	Type	Description
SDKAppID	int	You have already obtained it in the last step of Step 1, so it will not be elaborated here.
UserID	String	The ID of the current user, in string format, only allows letters (a-z and A-Z), digits (0-9), hyphens, and underscores.
userSig	String	Use the SecretKey obtained in step 3 of Activate Service to encrypt information such as SDKAppID and UserID to generate a UserSig. It's a credential used for authentication purposes, allowing Tencent Cloud to identify if the current user is authorized to use the TRTC service. You can generate a temporary UserSig through the Auxiliary Tools in the console. For more information, please refer to How to Calculate and Use UserSig .

Note:

Development Environment: If you are in the local development and debugging stage, you can use the local `GenerateTestUserSig.genTestSig` function to generate userSig. In this method, the SDKSecretKey is vulnerable to decompilation and reverse engineering, and once your key is leaked, attackers can steal your Tencent Cloud traffic.

Production Environment: If your project is going to be launched, please adopt the method of [Server-side Generation of UserSig](#).

Going Live

Last updated : 2025-05-06 11:37:37

This document mainly introduces how to use `SeatGridView` to go live. Only after going live can other audience members enter the voice chat room to listen to the host's broadcast.

Prerequisites

Before using `SeatGridView`, you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

```
import RTCRoomEngine
import SeatGridView

class BroadcastController: UIViewController {
    private let seatGridView: SeatGridView = {
        let view = SeatGridView()
        return view
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Add seatGridView to the view and set layout constraints
    }
}

import com.trtc.uikit.livekit.seatGridView.SeatGridView;

public class BroadcastActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

```
SeatGridView seatGridView = new SeatGridView(this);
addContentView(seatGridView,
    new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, Vie

}

}
```

Step 2: Preparing Live Streaming Parameters

When calling the `startVoiceRoom` API, you need to fill in the key parameters `TUIRoomInfo` . The following is a detailed introduction:

Parameters: TUIRoomInfo

`TUIRoomInfo` consists of many fields, but usually, you only need to focus on filling in the following fields:

Parameter Name	Field Description	Additional Notes	Data Type	Example
roomId	Room ID	Only letters (a-z, A-Z), digits (0-9), underscores, and hyphens are allowed.	New Character String	"room_100001"
name	Room Name	Room name of string type.	New Character String	"denny`s room"
seatMode	Microphone Mode	This field is effective only when microphone control is enabled. It is divided into "freeToTake" and "applyToTake" modes. In freeToTake mode, audience members can take the microphone freely without applying. In applyToTake mode, audience members need the room owner's approval to take the microphone.	Enumeration Value	TUISeatMode.applyToTake
maxSeatCount	Maximum	The maximum	Digits	10

	Number of Microphones	number of mic positions of numeric type, referring to the maximum number of people on mic simultaneously, maxSeatCount. The maximum number of mic positions is consistent with the upper limit of the number of 'Multi-guests' in the purchased package .		
isSeatEnabled	Whether to enable microphone position control	Boolean type mic control enable flag.	Boolean value	true
roomType	Room type	Divided into two room types: "conference" and "live". Voice chat belongs to live, so use live here.	Enumeration Value	TUIRoomType.live

Step 3: Starting Live Streaming

After preparing the parameters in Step 2 `TUIRoomInfo` , you can call the `startVoiceRoom` API function to start live streaming.

After starting the live stream, you also need to call the `startMicrophone` API to enable the local microphone, ensuring that the audience can hear your voice.

iOS

Android

```
// Assemble TRTC room entry parameters. Replace the field values in `TRTCParams` with
let roomInfo = TUIRoomInfo()
roomInfo.roomId = "voice_100001" // Please replace it with your own room ID.
roomInfo.name = "denny`s room" // Please replace it with your own room name.
roomInfo.seatMode = .applyToTake // In typical video live streaming scenarios, the
roomInfo.maxSeatCount = 10 // Please replace it with the maximum number of s
roomInfo.isSeatEnabled = true // If you do not need seat control, you can set i
roomInfo.roomType = .live // Ensure the room type is live.

self.seatGridView.startVoiceRoom(roomInfo: roomInfo) { [weak self] roomInfo in
```

```
// Going live successfully
guard let self = self else { return }
self.seatGridView.startMicrophone {
    // Successfully turned the local mic on
} onError: { code, message in
    // Failed to turn the local mic on
}

} onError: { code, message in
    // Failed to go live
}

TUIRoomDefine.RoomInfo roomInfo = new TUIRoomDefine.RoomInfo();
roomInfo.roomId = "voice_100001"; // Please replace it with your own room ID
roomInfo.name = "denny's room"; // Please replace it with your own room name
roomInfo.maxSeatCount = 10; // Please replace it with the maximum number of seats i
roomInfo.isSeatEnabled = true; // If you do not need seat control, you can set isSe
roomInfo.roomType = TUIRoomDefine.RoomType.LIVE; // Ensure the room type is live.
roomInfo.seatMode = TUIRoomDefine.SeatMode.APPLY_TO_TAKE; // If you don't need to

seatGridView.startVoiceRoom(roomInfo, new TUIRoomDefine.GetRoomInfoCallback() {
    @Override
    public void onSuccess(TUIRoomDefine.RoomInfo roomInfo) {
        // Going live successfully
        seatGridView.startMicrophone( new TUIRoomDefine.ActionCallback() {
            @Override
            public void onSuccess() {
                // Successfully turned the local mic on
            }

            @Override
            public void onError(TUICommonDefine.Error error, String message) {
                // Failed to turn the local mic on
            }
        });
    }
    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // Failed to go live
    }
});
```


Listening

Last updated : 2024-12-24 17:45:36

This document mainly introduces how to use `SeatGridView` to listen to a certain host's live broadcast.

Prerequisites

Before using `SeatGridView`, you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

```
import SeatGridView
import RTCRoomEngine

class ListenController: UIViewController {
    private let seatGridView: SeatGridView = {
        let view = SeatGridView()
        return view
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.seatGridView.addObserver(observer: self)
        // Add seatGridView to the view and set layout constraints
    }

    deinit {
        self.seatGridView.removeObserver(observer: self)
    }
}

import com.trtc.uitkit.livekit.seatGridView.SeatGridView;
```

```
public class ListenActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SeatGridView seatGridView = new SeatGridView(this);
        addContentView(seatGridView,
            new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, Vie
        }
    }
}
```

Step 2: Listening

Call `joinVoiceRoom` to enter a host's live room for listening.

iOS

Android

```
let roomId = "voice_100001"    // Please replace it with the room ID of the host yo
self.seatGridView.joinVoiceRoom(roomId: roomId) { roomInfo in
    // Listening succeeded
} onError: { code, message in
    // Listening failed
}

String roomId = "live_100001";    // Please replace it with the room ID of the host
seatGridView.joinVoiceRoom(roomId, new TUIRoomDefine.GetRoomInfoCallback() {
    @Override
    public void onSuccess(TUIRoomDefine.RoomInfo roomInfo) {
        // Successfully joined the live streaming room
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // Failed to join the live streaming room
    }
});
```

When you want to leave the live room, you can call the `leaveVoiceRoom` API.

iOS

Android

```
self.seatGridView.leaveVoiceRoom {
    // Successfully exited the live streaming room
} onError: { code, message in
    // Failed to exit the live streaming room
}
```

```
}

seatGridView.leaveVoiceRoom( new , new TUIRoomDefine.GetRoomInfoCallback() {
    @Override
    public void onSuccess(TUIRoomDefine.RoomInfo roomInfo) {
        // Successfully exited the live streaming room
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // Failed to exit the live streaming room
    }
});
```

When the host dismisses the room, you will receive the `onRoomDismissed` callback.

iOS

Android

```
func onRoomDismissed(roomId: String) {
    // The room has been dissolved
}

void onRoomDismissed(String roomId) {
    // The room has been dissolved
}
```

Seat Management

Last updated : 2024-12-24 17:53:06

This document mainly introduces the seat management capabilities of `SeatGridView`.

`SeatGridView` supports the following seat management capabilities:

Take seat	Leave Seat	Move seat
Invite to take seat	Kick off seat	Lock seat

Prerequisites

Before using `SeatGridView`, you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

Step 2: Managing Seats

Note:

Note: When using seat management, you need to ensure that you have started broadcasting or entered the live room.

Take seat

The behavior of applying to speak is related to the speaking mode.

When the speaking mode of the live room is "apply to speak mode (`applyToTake`)", applying to speak requires the room owner's approval.

When the speaking mode of the live room is "free to speak mode (`freeToTake`)", applying to speak does not require the room owner's approval and can be done directly.

When you want to speak, call the `takeSeat` API and pass in two parameters: seat index and timeout duration.

iOS

Android

```
let index = 1    // Please replace this with the seat number you want to apply for
let timeout = 30 // Please replace this with the timeout duration for your speaking
self.seatGridView.takeSeat(index: index, timeout: timeout) { userInfo in
    // Speaking request accepted
} onRejected: { userInfo in
    // Speaking request rejected
} onCancelled: { userInfo in
    // Speaking request canceled
} onTimeout: { userInfo in
    // Speaking request timed out
} onError: { userInfo, code, message in
    // Speaking request error
}

int index = 1;    // Please replace this with the seat number you want to apply for
int timeout = 30; // Replace this with the timeout duration for requesting to speak
seatGridView.takeSeat(index, timeout, new TUIRoomDefine.RequestCallback() {
    @Override
    public void onAccepted(String requestId, String userId) {
        // Speaking request accepted
    }

    @Override
    public void onRejected(String requestId, String userId, String message) {
        // Speaking request rejected
    }

    @Override
    public void onCancelled(String requestId, String userId) {
        // Speaking request canceled
    }

    @Override
    public void onTimeout(String requestId, String userId) {
        // Speaking request timed out
    }

    @Override
    public void onError(String requestId, String userId, TUICommonDefine.Error error) {
        // Speaking request error
    }
});
```

When you are the host and someone requests to speak, you will receive the `onSeatRequestReceived` callback. You can call `responseRemoteRequest` to accept/reject the speaking request.

iOS

Android

```
func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo) {
    if type == .applyToTakeSeat {
        let agree = true // true means
        self.seatGridView.responseRemoteRequest(userId: userInfo.userId, agree: true)
        // Agreed to speak successfully
    } onError: { code, message in
        // Failed to agree to speak
    }
}

void onSeatRequestReceived(VoiceRoomDefine.RequestType type, TUIRoomDefine.UserInfo userInfo) {
    if (type == APPLY_TO_TAKE_SEAT) {
        boolean agree = true; // true: agree to connect, false: refuse to connect
        seatGridView.responseRemoteRequest(userInfo.userId, agree, new RequestResponse() {
            @Override
            public void onSuccess () {
                // Agreed to speak successfully
            }

            @Override
            public void onError (TUICommonDefine.Error error, String message) {
                // Failed to agree to speak
            }
        });
    }
}
```

Leave Seat

When you are speaking and want to leave, you can call the `leaveSeat` API.

iOS

Android

```
self.seatGridView.leaveSeat {
    // Left the seat successfully
} onError: { code, message in
    // Failed to leave the seat
}
```

```
seatGridView.leaveSeat(new TUIRoomDefine.ActionCallback {  
    @Override  
    public void onSuccess () {  
        // Left the seat successfully  
    }  
  
    @Override  
    public void onError (TUICommonDefine.Error error, String message) {  
        // Failed to leave the seat  
    }  
});
```

Move Seat

When you are on seat 1 and want to switch to seat 2, you can call the `moveToSeat` API and pass in the index of the seat to switch to.

iOS

Android

```
let index = 2  
self.seatGridView.moveToSeat(index: destinationIndex) {  
    // Successfully moved the seat  
} onError: { code, message in  
    // Failed to move the seat  
}  
  
int index = 2;  
seatGridView.moveToSeat(index, new TUIRoomDefine.ActionCallback {  
    @Override  
    public void onSuccess () {  
        // Successfully moved the seat  
    }  
  
    @Override  
    public void onError (TUICommonDefine.Error error, String message) {  
        // Failed to move the seat  
    }  
});
```

Invite to take seat

When you are the host and want to invite someone not on the mic to seat 4, you can call the

`takeUserOnSeatByAdmin` API, passing three parameters: the seat index to invite to, the timeout period, and the user ID of the person being invited.

iOS

Android

```
let targetIndex = 4
let timeout = 30
let targetUserId = "100002" // Please replace this with the user ID of the host you
self.takeUserOnSeatByAdmin(index: targetIndex, timeout: timeout, userId: targetUserId)
// Speaking invitation accepted
} onRejected: { userInfo in
    // Speaking invitation rejected
} onTimeout: { userInfo in
    // Speaking invitation timed out
} onCancelled: { userInfo in
    // Speaking invitation canceled
} onError: { userInfo, code, message in
    // Speaking invitation error
}
```

```
int targetIndex = 4;
int timeout = 30;
String targetUserId = "100002"; // Please replace this with the user ID of the host
seatGridView.takeUserOnSeatByAdmin(targetIndex, timeout, targetUserId, new new TUIR
@Override
public void onAccepted(String requestId, String userId) {
    // Speaking invitation accepted
}

@Override
public void onRejected(String requestId, String userId, String message) {
    // Speaking invitation rejected
}

@Override
public void onCancelled(String requestId, String userId) {
    // Speaking invitation canceled
}

@Override
public void onTimeout(String requestId, String userId) {
    // Speaking invitation timed out
}

@Override
public void onError(String requestId, String userId, TUICommonDefine.Error error) {
    // Speaking invitation error
}
```



```
});
```

When someone invites you to speak, you will receive the `onSeatRequestReceived` callback. You can call

`responseRemoteRequest` to accept/reject the speaking invitation.

iOS

Android

```
func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo) {
    if type == .inviteToTakeSeat {
        let isAccepted = true // true means accepting the invitation, false means r
        self.seatGridView.responseRemoteRequest(userId: userInfo.userId, agree: isA
            // Agreed to speak invitation successfully
        } onError: { code, message in
            // Failed to agree to speak invitation
        }
    }
}

void onSeatRequestReceived(VoiceRoomDefine.RequestType type, TUIRoomDefine.UserInfo
    if (type == INVITE_TO_TAKE_SEAT) {
        boolean isAccepted = true; // true means accepting the invitation, false me
        seatGridView.responseRemoteRequest(userInfo.userId, isAccepted, new TUIRoo
            @Override
            public void onSuccess () {
                // Agreed to speak invitation successfully
            }

            @Override
            public void onError (TUICommonDefine.Error error, String message) {
                // Failed to agree to speak invitation
            }
        });
    }
}
```

Kick off seat

When you are the host and want to remove the speaker on seat 3, you can call the `kickUserOffSeatByAdmin` API and pass in the user ID of the speaker you want to remove.

iOS

Android

```
String targetUserId = "100002"; // Please replace this with the user ID of the host
seatGridView.kickUserOffSeatByAdmin(targetUserId, new TUIRoomDefine.ActionCallback
    @Override
```

```
public void onSuccess () {
    // Successfully removed a speaker
}

@Override
public void onError (TUICommonDefine.Error error, String message) {
    // Failed to remove a speaker
}

});

String targetUserId = "100002"; // Please replace this with the user ID of the host
self.seatGridView.kickUserOffSeatByAdmin(targetUserId,  new TUIRoomDefine.ActionCal
@Override
public void onSuccess () {
    // Successfully removed a speaker
}

@Override
public void onError (TUICommonDefine.Error error, String message) {
    // Failed to remove a speaker
}

});
```

Lock Seat

When you are the host and want to lock the empty seat at position 5 to prevent others from taking it, or mute the host on seat 6, you can call the `lockSeat` API, passing in two parameters: the index of the seat to be locked and the lock mode.

The structure of the lock mode (`TUISeatLockParams`) is as follows:

Property Name	Field Description	Additional Notes	Data Type	Example
lockSeat	Lock Microphone Position	Locking the corresponding seat prevents applications to take that seat.	Boolean value	True when the seat is locked.
lockVideo	Lock the seat camera.	Locking the corresponding seat camera will stop the seat from publishing video streams.	Boolean value	false

lockAudio	Lock the seat microphone.	Locking the corresponding seat camera will stop the seat from publishing audio streams.	Boolean value	True when the seat microphone is locked.
-----------	---------------------------	---	---------------	--

iOS

Android

```
// Lock a seat.
let lockSeatIndex = 5
let lockSeatParam = TUISeatLockParams()
lockSeatParam.lockSeat = true
self.seatGridView.lockSeat(index: lockSeatIndex, lockMode: lockSeatParam) {
    // Lock the seat successfully
} onError: { code, message in
    // Failed to lock the seat
}

// Lock the seat microphone
let lockSeatAudioIndex = 6
let lockSeatAudioParam = TUISeatLockParams()
lockSeatAudioParam.lockAudio = true
self.seatGridView.lockSeat(index: lockSeatAudioIndex, lockMode: lockSeatAudioParam)
    // Lock the seat microphone successfully
} onError: { code, message in
    // Failed to lock the seat microphone
}

// Lock a seat.
int lockSeatIndex = 5;
TUIRoomDefine.SeatLockParams lockSeatParam = new TUIRoomDefine.SeatLockParams();
lockSeatParam.lockSeat = true;
seatGridView.lockSeat(lockSeatIndex, lockSeatParam, new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
        // Lock the seat successfully
    }

    @Override
    public void onError(TUICommonDefine.Error error, String message) {
        // Failed to lock the seat
    }
})
```

```
    }  
});  
  
// Lock the seat microphone  
int lockSeatIndex = 6;  
TUIRoomDefine.SeatLockParams lockSeatAudioParam = new TUIRoomDefine.SeatLockParams(  
    lockSeatAudioParam.lockAudio = true;  
    seatGridView.lockSeat(lockSeatIndex, lockSeatAudioParam, new TUIRoomDefine.ActionCa  
        @Override  
        public void onSuccess() {  
            // Lock the seat microphone successfully  
        }  
  
        @Override  
        public void onError(TUICommonDefine.Error error, String message) {  
            // Failed to lock the seat microphone  
        }  
});
```

Microphone Management

Last updated : 2024-12-24 18:00:32

This document mainly introduces the microphone management capabilities of `SeatGridView` .

`SeatGridView` supports the following microphone management capabilities:

Start Microphone	Stop Microphone
Mute Microphone	Unmute Microphone

Prerequisites

Before using `SeatGridView` , you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

```
import UIKit
import RTCRoomEngine
import SeatGridView

class SeatManagementController: UIViewController {
    private let seatGridView: SeatGridView = {
        let view = SeatGridView()
        return view
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.seatGridView.addObserver(observer: self)
        // Add seatGridView to the view and set layout constraints
    }

    deinit {
        self.seatGridView.removeObserver(observer: self)
    }
}
```

```
    }  
}  
  
import com.trtc.uikit.livekit.seatGridView.SeatGridView;  
  
public class MicrophoneManagementActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(@Nullable Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        SeatGridView seatGridView = new SeatGridView(this);  
        addContentView(seatGridView,  
            new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, Vie  
  
        seatGridView.addObserver(this);  
    }  
  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        seatGridView.removeObserver(this);  
    }  
}
```

Step 2: Managing the Microphone

Start Microphone

Call `startMicrophone` to turn on the microphone.

iOS

Android

```
self.seatGridView.startMicrophone {  
    // Successfully turned on the mic  
} onError: { code, message in  
    // Failed to turn on the mic  
}  
  
seatGridView.startMicrophone(new TUIRoomDefine.ActionCallback() {  
    @Override  
    public void onSuccess () {  
        // Successfully turned on the mic  
    }  
  
    @Override  
    public void onError (TUICommonDefine.Error error, String message) {
```

```
        // Failed to turn on the mic
    }
});
```

Stop Microphone

Call `stopMicrophone` to turn off the microphone.

iOS

Android

```
self.seatGridView.stopMicrophone()

seatGridView.stopMicrophone();
```

Mute Microphone

Call `muteMicrophone` to mute.

iOS

Android

```
self.seatGridView.muteMicrophone()

seatGridView.muteMicrophone();
```

Unmute Microphone

Call `unmuteMicrophone` to unmute.

iOS

Android

```
self.seatGridView.unmuteMicrophone {
    print("Unmuted successfully")
} onError: { code, message in
    print("Failed to unmute")
}

seatGridView.unMuteLocalAudio(new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
        // Successfully unmuted
    }

    @Override
```

```
public void onError(TUICommonDefine.Error error, String message) {  
    // // Failed to unmute  
}  
});
```

When someone's microphone status changes, you will receive a callback `onUserAudioStateChanged` .

iOS

Android

```
func onUserAudioStateChanged(userInfo: TUIUserInfo, hasAudio: Bool, reason: TUICan  
    if hasAudio {  
        print("\(userInfo.userId) has audio")  
    } else {  
        print("\(userInfo.userId) has no audio")  
    }  
}  
  
void onUserAudioStateChanged(TUIRoomDefine.UserInfo userInfo, boolean hasAudio,  
    TUIRoomDefine.ChangeReason reason) {  
    // userInfo.userName's audio status changed  
}
```


Seat Layout

Last updated : 2024-12-24 18:07:20

This document mainly introduces how to customize the seat layout for `SeatGridView`.

Prerequisites

Before using `SeatGridView`, you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

```
import UIKit
import RTCRoomEngine
import SeatGridView

class CustomizeSeatLayoutController: UIViewController {
    private let seatGridView: SeatGridView = {
        let view = SeatGridView()
        return view
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Add seatGridView to the view and set layout constraints
    }
}

import com.trtc.uikit.livekit.seatGridView.SeatGridView;

public class CustomizeSeatLayoutActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

```
SeatGridView seatGridView = new SeatGridView(this);
addContentView(seatGridView,
               new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, Vie
    }
}
```

Step 2: Preparing Seat Layout Parameters

When calling the `setLayoutMode` API, you need to fill in the key parameters `layoutMode` and `layoutConfig`. The following is a detailed introduction:

Parameter 1: layoutMode

`layoutMode` is an enumeration of type `SGLayoutMode`.

Enumeration value types	Meaning	Additional Notes
focus	Centralized layout (e.g., 1-3-3 layout, 1 element in the first row, 3 elements in the second and third rows)	Built-in layout styles
grid	Grid Layout	Built-in layout styles (this is the default style)
vertical	Vertical layout	Built-in layout styles
free	Free layout	Custom layout styles

Notes:

When using the built-in layout styles of `seatGridView`, there is no need to provide additional `SGSeatViewLayoutConfig` parameters. You only need to pass the `SGLayoutMode` parameter to `setLayoutMode`.

Parameter Two: layoutConfig

`layoutConfig` is a structure of type `SGSeatViewLayoutConfig`, consisting of the fields `rowConfigs` and `rowSpacing`.

Parameter Name	Field Description	Data Type
rowConfigs	Used to store the configuration of each row's seat layout.	An array of <code>SGSeatViewLayoutRowConfig</code> type
rowSpacing	Used to store the row spacing of the seat layout	Floating point number

`SGSeatViewLayoutRowConfig` consists of four fields: `count` , `seatSpacing` , `seatSize` , and `alignment` .

Parameter Name	Field Description	Data Type
<code>count</code>	Number of seats per row	Integer
<code>seatSpacing</code>	Vertical spacing of seats in the same row	Floating point number
<code>seatSize</code>	Size of the seat	Size type
<code>alignment</code>	Arrangement of seats in the same row	<code>SGSeatViewLayoutRowAlignment</code>

`SGSeatViewLayoutRowAlignment` is an enumeration.

Enumeration value types	Meaning
<code>spaceAround</code>	Dispersed alignment, not against the container wall, with remaining space evenly distributed on both sides of each item
<code>spaceBetween</code>	Spaced alignment, with the leftmost and rightmost items against the left or right border, and equal spacing between items.
<code>spaceEvenly</code>	Average alignment, not against the container wall, with remaining space evenly distributed.
<code>start</code>	Left alignment
<code>end</code>	Right alignment
<code>center</code>	Center alignment

Step 3: Setting the Seat Layout

After preparing the parameters `layoutMode` and `layoutConfig` in Step 2, you can call the `setLayoutMode` API function to set the seat layout.

Built-In Layout

When using the built-in layout, you only need to pass the `layoutMode` parameter.

iOS

Android

```
// Centralized layout
self.seatGridView.setLayoutMode(layoutMode: .focus)
// Grid Layout
```

```
self.seatGridView.setLayoutMode(layoutMode: .grid)
// Vertical
self.seatGridView.setLayoutMode(layoutMode: .vertical)

// Centralized layout
seatGridView.setLayoutMode(VoiceRoomDefine.LayoutMode.FOCUS, null);
// Grid Layout
seatGridView.setLayoutMode(VoiceRoomDefine.LayoutMode.GRID, null);
// Vertical layout
seatGridView.setLayoutMode(VoiceRoomDefine.LayoutMode.VERTICAL, null);
```

Custom Layout

When using a custom layout, the value of `layoutMode` should be `free` and the corresponding seat layout configuration parameter `layoutConfig` needs to be passed.

For example, to customize a 2, 1, 2 layout, with each row arranged as `spaceBetween`, `center`, and `spaceBetween`:

iOS

Android

```
// Create seat configuration for each row
let firstRowConfig = SGSeatViewLayoutRowConfig(count: 2,
    seatSpacing: 10.0, seatSize: CGSize(width: 50, height: 50), alignment: .spaceBe
let secondRowConfig = SGSeatViewLayoutRowConfig(count: 1,
    seatSpacing: 10.0, seatSize: CGSize(width: 72, height: 72), alignment: .center)
let thirdRowConfig = SGSeatViewLayoutRowConfig(count: 2,
    seatSpacing: 10.0, seatSize: CGSize(width: 50, height: 50), alignment: .spaceBe

let rowConfigs: [SGSeatViewLayoutRowConfig] = [firstRowConfig, secondRowConfig, thi
let layoutConfig = SGSeatViewLayoutConfig(rowConfigs: rowConfigs, rowSpacing: 20.0)

// Set layout mode
self.seatGridView.setLayoutMode(layoutMode: .free, layoutConfig: layoutConfig)

// Create seat configuration for each row
VoiceRoomDefine.SeatViewLayoutRowConfig firstRowConfig = new VoiceRoomDefine.SeatVi
firstRowConfig.count = 2;
firstRowConfig.seatSpacing = 10;
firstRowConfig.seatSize = new VoiceRoomDefine.Size(50,50);
firstRowConfig.alignment = SPACE_BETWEEN;

VoiceRoomDefine.SeatViewLayoutRowConfig secondRowConfig = new VoiceRoomDefine.SeatV
secondRowConfig.count = 1;
secondRowConfig.seatSpacing = 10;
secondRowConfig.seatSize = new VoiceRoomDefine.Size(72,72);
secondRowConfig.alignment = CENTER;
```

```
VoiceRoomDefine.SeatViewLayoutRowConfig thirdRowConfig = new VoiceRoomDefine.SeatVi
thirdRowConfig.count = 2;
thirdRowConfig.seatSpacing = 10;
thirdRowConfig.seatSize = new VoiceRoomDefine.Size(50,50);
thirdRowConfig.alignment = SPACE_BETWEEN;

List<VoiceRoomDefine.SeatViewLayoutRowConfig> rowConfigs = new ArrayList<>();
rowConfigs.add(firstRowConfig);
rowConfigs.add(secondRowConfig);
rowConfigs.add(thirdRowConfig);
VoiceRoomDefine.SeatViewLayoutConfig config = new VoiceRoomDefine.SeatViewLayoutCon
config.rowConfigs = rowConfigs;
config.rowSpacing = 20;

// Set layout mode
seatGridView.setLayoutMode(LayoutMode.FREE, config);
```

Customizing a Seat

Last updated : 2024-12-24 18:09:24

This document mainly introduces how to customize a `SeatGridView` .

Prerequisites

Before using `SeatGridView` , you need to [integrate and log in](#) to `SeatGridView` to ensure the subsequent features work properly.

Usage guide

Step 1: Adding SeatGridView to the View

You need to import the `SeatGridView` module first, then create a `SeatGridView` object and add it to your view.

iOS

Android

```
import UIKit
import RTCRoomEngine
import SeatGridView

class CustomizeSeatViewController: UIViewController {
    private let seatGridView: SeatGridView = {
        let view = SeatGridView()
        self.seatGridView.setSeatViewDelegate(self)
        return view
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.seatGridView.setSeatViewDelegate(self)
        // Add seatGridView to the view and set layout constraints
    }
}

import com.trtc.uitkit.livekit.seatGridView.SeatGridView;

public class CustomizeSeatViewActivity extends AppCompatActivity {
    @Override
```

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    SeatGridView seatGridView = new SeatGridView(this);
    addContentView(seatGridView,
        new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, Vie
    }
}
```

Step 2: Customizing a Seat

iOS

Android

If you want to customize the seat view, you need to implement the `SeatGridView` delegate

`SGSeatViewDelegate`.

```
protocol SGSeatViewDelegate{
    func seatGridView(_ view: SeatGridView, createSeatView seatInfo: TUISeatInfo) -
    func seatGridView(_ view: SeatGridView, updateSeatView seatInfo: TUISeatInfo, s
    func seatGridView(_ view: SeatGridView, updateUserVolume volume: Int, seatView:
}
```

The following is a usage example:

```
extension CustomizeSeatViewController: SGSeatViewDelegate {
    func seatGridView(_ view: SeatGridView, createSeatView seatInfo: TUISeatInfo) -
        return CustomSeatView(seatInfo: seatInfo)
    }

    func seatGridView(_ view: SeatGridView, updateSeatView seatInfo: TUISeatInfo, s
        if let seatView = seatView as? CustomSeatView {
            seatView.updateSeatView(withSeatInfo: seatInfo)
        }
    }

    func seatGridView(_ view: SeatGridView, updateUserVolume volume: Int, seatView:
        if let seatView = seatView as? CustomSeatView {
            seatView.updateSeatView(withVolume: volume)
        }
    }
}

// Please replace with your SeatView
class CustomSeatView: UIView {
    var seatInfo: TUISeatInfo = TUISeatInfo()
    var volume: Int = 0
}
```

```
// ... UI

init(seatInfo: TUISeatInfo) {
    self.seatInfo = seatInfo
}

required init?(coder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
}

func updateSeatView(withSeatInfo seatInfo: TUISeatInfo) {
    self.seatInfo = seatInfo
    // ...
}

func updateSeatView(withVolume volume: Int) {
    self.volume = volume
    // ...
}
}
```

If you want to customize the seat view, you need to implement the `SeatGridView` adapter `SeatViewAdapter`.

```
public interface SeatViewAdapter {
    View createSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo)
    void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo,
    void updateUserVolume(SeatGridView seatGridView, int volume, View seatView);
}
```

The following is a usage example:

```
seatGridView.setSeatViewAdapter(new VoiceRoomDefine.SeatViewAdapter() {
    @Override
    View createSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo)
        TextView seatUserName = new TextView(CustomizeSeatViewActivity.this);
        seatUserName.setText(seatInfo.userName);
        return seatUserName;
}

@Override
void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo,
```



```
@Override
void updateUserVolume(SeatGridView seatGridView, int volume, View seatView) {

}
});
```

Server APIs (TUILiveKit)

REST API

RESTful API Overview

Last updated : 2024-08-21 18:04:59

RESTful API is the backend HTTP management interface for the Live Streaming Room SDK. Its main purpose is to provide developers with a simple management entry point. For security, RESTful API only offers HTTPS interfaces.

Prerequisites

To call the RESTful API, you must purchase or acquire the necessary Package bundles for LiveSdk.

Calling method

Request URL

The URL format of the RESTful API is as follows:

```
https://console.tim.qq.com/$ver/$servicename/$command?
sdkappid=$SDKAppID&identifier=$identifier&usersig=$usersig&random=99999999&cont
enttype=json
```

The meanings and values of each parameter are as follows (both parameter names and their values are case-sensitive):

Parameter	Meaning	Fetching Value
https	Request protocol	The request protocol is HTTPS, and the request method is POST
console.tim.qq.com	Request Domain	Fixed as console.tim.qq.com
ver	Protocol Version Number	Fixed as v4
servicename	Internal Service Name, different servicenames correspond to different service types	Example: v4/live_engine_http_srv/create_room , where room_engine_http_srv is the servicename . For more details, refer to the RESTful API List

command	Command Word, used in combination with the servicename to identify a specific business feature	Example: v4/live_engine_http_srv/create_room , where create_room is the command . For more details, refer to the RESTful API List
sdkappid	Application Identifier obtained from the Chat console	Obtained when applying for integration
identifier	username, must be an App Administrator Account when calling RESTful APIs	Refer to App Administrator
usersig	password corresponding to username	Refer to Generating UserSig
random	Identifies the random number parameter for the current request	A 32-bit unsigned integer random number, ranging from 0 to 4294967295
contenttype	Request Format	Fixed value is json

Note:

When calling REST APIs, the App Server's identifier must be an App Administrator Account.

An App can generate a UserSig for the Administrator Account each time it calls a REST API, or it can generate a fixed UserSig for repeated use, but please pay special attention to the validity period of the UserSig.

HTTP Request Body Format

REST APIs only support the POST method, and the request body is in JSON format. For the specific body format, refer to the detailed description of each API.

It's particularly important that the POST body cannot be empty. Even if a protocol doesn't require any information to be carried, an empty JSON object, namely {}, must be included.

HTTP return code

Unless a network error occurs (e.g., a 502 error), the invocation result of REST APIs is always 200. The real API invocation error codes and error messages are returned in the HTTP response body.

HTTP Response Body Format

The response body of the RESTful API is also in JSON format, and its format conforms to the following characteristics:

```
{  
  "ActionStatus": "OK",
```

```
"ErrorInfo": "",
"ErrorCode": 0,
"RequestId": "Id-70e312f1de024af5a36714b7b71da224-O-Seq-63504"
// Other response content of REST API
}
```

The response payload must contain the properties `ActionStatus`, `ErrorInfo`, `ErrorCode`, `RequestId`. Their meanings are as follows:

Field	Type	Description
ActionStatus	String	The result of the request processing. OK means success, FAIL means failure. If it's FAIL, ErrorInfo will provide the reason for failure
ErrorInfo	String	Failure
ErrorCode	Integer	Error code. 0 means success, others mean failure. You can refer to the Error Code Table for specific reasons
RequestId	String	Error code. 0 means success, others mean failure. You can refer to the Error Code Table for specific reasons

Common Error Codes for RESTful APIs

Error code	Description
60002	HTTP parsing error, please check the HTTP request URL format
60003	HTTP request JSON parsing error, please check the JSON format
60004	Request URL or JSON payload contains an incorrect account or signature
60005	Request URL or JSON payload contains an incorrect account or signature
60006	SDKAppID invalid, please verify the validity of SDKAppID
60007	RESTful API invocation frequency exceeds its limit, please reduce the request frequency
60008	Service request timed out or HTTP request format is incorrect, please check and retry
60009	Request Resource Error, please check the request URL
60010	The request requires App administrator privileges
60011	SDKAppID request frequency exceeds the limit, please reduce the request frequency

60012	RESTful API requires SDKAppID, please check the SDKAppID in the request URL
60013	HTTP response package JSON parsing error
60014	Account Replacement timed out
60015	The request body has the wrong account type. Please ensure the account is in string format
60016	SDKAppID is disabled.
60017	The request is disabled.
60018	The request is too frequent, please try again later.
60019	The request is too frequent, please try again later.
60020	Your Professional Edition package has expired and been deactivated. Please log in to [Chat purchase page](https://buy.tencentcloud.com/avc) to repurchase the package. Once purchased, it will take effect 5 minutes later.
60021	The Call Source IP for RestAPI is illegal.

FAQ

Is there a chance that REST API requests timeout, receiving no response?

1. The timeout set for the Live backend REST interface is 3 seconds. The caller's timeout setting should be longer than 3 seconds.
2. `telnet console.tim.qq.com 443` to confirm if the service port can be connected.
3. Use `curl -I https://console.tim.qq.com` to simply test if the status code is 200.
4. Confirm whether the machine's DNS server configuration is an internal DNS server or a public DNS server. If it is an internal DNS server, ensure that the DNS server's network egress and the region ISP of the machine's network egress IP match.
5. It is recommended for business callers to use the "Long Connection + Connection Pool" model.

Note:

Due to the high time consumption of establishing HTTPS short connections, every request incurs TCP and TLS handshake overhead. Therefore, it is recommended to access REST APIs with long connections.

Using the standard HTTP library scenarios: HTTP 1.0 requires specifying the request header `Connection: keep-alive`, HTTP 1.1 supports long connections by default. In scenarios where HTTPS requests are encapsulated based on TCP, you can reuse TCP connections to send and receive requests.

RESTful API List

Last updated : 2024-08-21 18:04:59

Room management

Feature	API
Create Room	live_engine_http_srv/create_room
Dismiss room	live_engine_http_srv/destroy_room
Update Room Information	live_engine_http_srv/update_room_info
Retrieve Room Information	live_engine_http_srv/get_room_info
Getting the Live Stream List	live_engine_http_srv/get_room_list

User management

Feature	API
Fetch Administrator List	live_engine_http_srv/get_admin_list
Member Ban	live_engine_http_srv/ban_member
Unblocking Members	live_engine_http_srv/unban_member
Obtaining the Block List	live_engine_http_srv/get_banned_member_list
Modify Administrator	live_engine_http_srv/modify_admin

Room Related

Create a Room

Last updated : 2025-04-09 16:06:20

Feature Overview

App admin can create a room through this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/create_room?sdkappid=88888888&identifier=adm
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/create_room	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App admin account. For more details, see App Admin
usersig	The signature generated by the App admin account. For specific operations, see Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomInfo":{
    "RoomId":"live-room",
    "RoomType":"Live",
    "RoomName":"live name",
    "Owner_Account":"administrator",
    "MaxMemberCount":400,
    "CoverURL":"cover url",
    "IsMessageDisabled":true,
    "Category":[1, 2, 3],
    "IsPublicVisible": true,
    "IsSeatEnabled": true,
    "TakeSeatMode":"ApplyToTake",
    "MaxSeatCount":8,
    "ActivityStatus":1,
    "IsUnlimitedRoomEnabled":true
  }
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID, up to 48 bytes
RoomName	String	Optional	Room Name, defaults to Room ID, up to 100 bytes
RoomType	String	Mandatory	Room Type: Live (Live Room)
Owner_Account	String	Optional	Host ID (must be an imported account), defaults to the User ID of the API caller
MaxMemberCount	Integer	Optional	Maximum number of room members, Default Setting: Upper limit of paid package, for example, the Trial Version is 20, if you Upgrade Package, you need to Modify room information to modify this field
CoverURL	String	Optional	Room cover, Maximum 200 bytes

IsMessageDisabled	Bool	Optional	Mute all audio, Default false
Category	Array	Optional	Room Type Indicator, can be used as a room type identifier by services, such as Game, Music, etc, Supports up to three.
IsPublicVisible	Bool	Optional	Room Public Indicator
IsSeatEnabled	Bool	Optional	Support for microphone positions, default is not supported
TakeSeatMode	String	Optional	When using microphone position capability, you need to fill in the microphone mode, FreeToTake (free to take the mic), ApplyToTake (apply to take the mic)
MaxSeatCount	Integer	Optional	Maximum number of microphone positions, when using the microphone position capability, defaults to package limit
ActivityStatus	Integer	Optional	Live room active status: user-defined Definition tag
IsUnlimitedRoomEnabled	Bool	Optional	Enable the live room stream mixing capability to support high-concurrency scenarios. Default: false. When set to true, it means using the SDK TUILiveKit .

Note :

After the live streaming, users need to manually call the [Dissolve Room](#) API to end the meeting.

If the Dissolve Room interface is not manually invoked, the backend will attempt to recycle the room 6 hours after the meeting ends, provided there are no members left in the room.

Sample response packets**Basic Form**

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "RoomId": "live-room"
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues
RoomId	String	Room ID

Error codes

Unless a network error occurs (e.g., 502 error), the HTTP status code for this interface will always be 200. The actual error codes and messages are conveyed through ErrorCode and ErrorInfo in the response body.

Common error codes (60000 to 79999) see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100003	Room ID already exists, please choose another Room ID
100007	No payment information, a package bundle needs to be purchased from the Console
100010	Room ID has been used, and the Operator is the Homeowner, it can be used directly
100011	Room ID is occupied by IM, you can change to another Room ID or dissolve the group first through the IM interface
100012	Creating rooms exceeds frequency limit; the same Room ID can only be created once per second

Possible callbacks

[Callback after creating a room](#)

Update the Room Information

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrators can update room information through this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/update_room_info?sdkappid=88888888&identifie
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/update_room_info	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomInfo" : {
    "RoomId": "live-room",
    "RoomName" : "rname-123",
    "MaxMemberCount" : 300,
    "IsMessageDisabled" : false,
    "CoverURL": "https://xxxx.png",
    "Category": [1,2,3],
    "ActivityStatus":1,
    "IsPublicVisible": false,
    "TakeSeatMode": "ApplyToTake"
  }
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
RoomName	String	Optional	Room Name
MaxMemberCount	Integer	Optional	Max Room Capacity
IsMessageDisabled	Bool	Optional	Room-wide Mute Switch, except for the Administrator and Homeowner
CoverURL	String	Optional	Room cover
Category	Array	Optional	Room Category Tag, maximum array length is 3
ActivityStatus	Integer	Optional	Room self-definition status
IsPublicVisible	Bool	Optional	Room Open Switch, mainly used for retrieving the room list
TakeSeatMode	String	Optional	Seat Mode, Free to Join the Podium FreeToTake, Apply to join the microphone ApplyToTake

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved
100006	No permission to dissolve, for example, the operator is neither the homeowner nor the root administrator, or the room being dissolved is not a live room, etc

Get the Room Information

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can retrieve room information through this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/get_room_info?sdkappid=88888888&identifier=a
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/get_room_info	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "live-room",
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": { // Response body
    "RoomInfo" : {
      "RoomId" : "rid-123",
      "RoomName" : "rname-123",
      "RoomType" : "Live",
      "Owner_Account" : 144115233775727695,
      "CreateTime": 1693271354,
      "IsSeatEnabled" : true,
      "TakeSeatMode":1,
      "MaxSeatCount" : 8,
      "MaxMemberCount" : 300,
      "IsMessageDisabled" : false,
      "CoverURL": "https://xxxx.png",
      "Category": [1,2,3],
      "ActivityStatus":1,
      "ViewCount":10,
      "IsPublicVisible": true
    }
  }
}
```


Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues
RoomInfo	Object	Room Details
RoomId	String	Room ID
RoomName	String	Room Name
RoomType	String	Room Type: Live (Live Room)
Owner_Account	String	Room Owner ID
IsSeatEnabled	Bool	Microphone Position Toggle
TakeSeatMode	Integer	Seat Mode: 1: Free to Join the Podium 2: Apply to join the microphone
MaxSeatCount	Integer	Number of Microphone Positions, limited by package bundles
MaxMemberCount	Integer	Maximum Room Capacity
IsMessageDisabled	String	Group-wide Mute Toggle
CoverURL	String	Room cover
Category	Array	Live Room Category Tags, maximum array size is 3
ActivityStatus	Integer	Live room active status: user-defined Definition tag
IsPublicVisible	Bool	Whether the room is publicly visible, used for retrieving the Live Room List
CreateTime	Integer	Room Creation Time
ViewCount	Integer	Total number of times the user has entered the room

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by `ErrorCode` and `ErrorInfo`. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved

Retrieve the Live Streaming List

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can use this interface to get the live stream list.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/get_room_list?sdkappid=88888888&identifier=a
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: console.tim.qq.com Singapore : adminapisgp.im.qcloud.com
v4/live_engine_http_srv/get_room_list	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: json

Maximum calling frequency

10 times per second.

Sample request packets

Basic Form

```
{
  "Next": "",
  "Count": 20
}
```

Note: The list is sorted in descending order by the ViewCount field, with pagination fetching

Request packet fields

Field	Type	Attribute	Description
Next	String	Mandatory	Pagination parameters for pagination request. The first time it is an empty string. Subsequently, use the Next field in the response body. When the Next field in the response body is an empty string, it indicates that the pull is completed
Count	Integer	Mandatory	Number of rooms for a single pull. The default is 20, and the upper limit is 20

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response" : {
    "Next": "1693271355-123123123123",
    "RoomList": [
      {
        "RoomName": "name",
        "RoomId": "room_id0",
        "Owner_Account": "bren",
        "CoverURL": "cover_url",
        "Category": [1,2,3],
        "CreateTime": 1693271354,
        "ActivityStatus": 1,
        "ViewCount": 1000
      },
    ],
  },
}
```

```
...
{
  "RoomName": "name",
  "RoomId": "room_id19",
  "Owner_Account": "bren",
  "CoverURL": "cover_url",
  "Category": [1, 2, 3],
  "CreateTime": 1693271354,
  "ActivityStatus": 1,
  "ViewCount": 900
}
]
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	Results of request processing. OK indicates success, FAIL indicates failure
ErrorCode	Integer	Error code. 0 indicates success, non-zero indicates failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID, each request will return a Request ID, which is required when locating issues
RoomList	Array	Room ID List, sorted by view count in descending order by default
RoomName	String	Room Name
RoomId	String	Room ID
Owner_Account	String	Room Owner ID
CoverURL	String	Room cover
Category	Array	Live Room Category Tags, maximum array size is 3
CreateTime	Integer	Room Creation Time
ActivityStatus	Integer	Live room active status: user-defined Definition tag
ViewCount	Integer	Total number of times the user has entered the room

Next	String	When the room list is fully loaded, it will return an empty string; otherwise, it returns the Next for the next request
------	--------	---

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description

Destroy a Room

Last updated : 2024-08-21 18:04:59

Feature Overview

App admin can dissolve a room using this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/destroy_room?sdkappid=88888888&identifier=ad
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/destroy_room	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App admin account. For more details, see App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "live-room"
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error

Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved
100006	No permission to dissolve, for example, the operator is neither the homeowner nor the root administrator, or the room being dissolved is not a live room, etc

User-Related Matters

Member Banishment

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can block members through this interface. Blocked users cannot enter the room during the ban validity period.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/ban_member?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/ban_member	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295

contenttype

Request format fixed value: json

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "test",
  "MemberList_Account": [
    "user1",
    "user2"
  ],
  "Reason": "",
  "Duration": 3600
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
MemberList_Account	Array	Mandatory	List of Blocked Members, Up to 20 at a time.
Reason	String	Optional	Ban Reason
Duration	Integer	Mandatory	Ban Time

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

--	--	--

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved
100006	No permission to transfer homeowner
100016	The new homeowner is banned, please unban first

Retrieve the Ban List

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can use this interface to unblock members.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/get_banned_member_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: console.tim.qq.com Singapore : adminapisgp.im.qcloud.com
v4/live_engine_http_srv/get_banned_member_list	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: json

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "test",
  "Offset": 0,
  "Limit": 10
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
Offset	Integer	Optional	Offset, initially starting from 0 for the first pull, subsequent requests are based on the NextOffset value in the response packet
Limit	Integer	Optional	Quantity of single-page pull, maximum value: 100

Sample response packets

Basic Form

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-138c94122acc4ca485c8c7bf270686ac-O-Seq-135511",
  "Response": { //Response Body
    "BannedAccountList": [
      {
        "Member_Account": "user2",
        "BannedUntil": 1719300627
      },
      {
        "Member_Account": "user1",
        "BannedUntil": 1719300627
      }
    ]
  }
}
```

```
      "NextOffset": 0 // Pagination Identifier
    }
  }
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	Results of request processing. OK indicates success, FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID, each request will return a Request ID, which is required when locating issues
BannedAccountList	Array	Blocked Member List
Member_Account	String	Blocked Member Account ID
BannedUntil	Integer	Ban end time

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved

Retrieve the List of Administrators

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can retrieve the room administrator list through this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/get_admin_list?sdkappid=88888888&identifier=
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/get_admin_list	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "test"
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": { //Response Body
    "Admin_Account": [
      "user2",
      "user1"
    ]
  }
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Admin_Account	Array	Administrator List
---------------	-------	--------------------

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved

Lifting a Member's Ban

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can use this interface to unblock members.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/unban_member?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qqcloud.com</code>
v4/live_engine_http_srv/unban_member	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "test",
  "MemberList_Account": [
    "user1",
    "user2"
  ]
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
MemberList_Account	Array	Mandatory	List of Unbanned Members, Up to 20 at a time.

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure
ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by `ErrorCode` and `ErrorInfo`. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved

Modify Administrator

Last updated : 2024-08-21 18:04:59

Feature Overview

App Administrator can set and cancel room administrators through this interface.

API Calling Description

Sample request URL

```
https://xxxxxx/v4/live_engine_http_srv/modify_admin?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

Request parameters

The table below only lists the parameters modified when this interface is called. For more details on other parameters, please refer to [RESTful API Overview](#).

Parameter	Description
xxxxxx	The reserved domain for the country/region where the SDKAppID is located: Others in China: <code>console.tim.qq.com</code> Singapore : <code>adminapisgp.im.qcloud.com</code>
v4/live_engine_http_srv/modify_admin	Request API
sdkappid	SDKAppID assigned by the Chat console when an app is created
identifier	Must be an App Administrator account. For more details, please refer to App Admin
usersig	The Signature generated by the App Administrator account, for detailed operations, please refer to Generating UserSig
random	A random 32-bit unsigned integer ranging from 0 to 4294967295
contenttype	Request format fixed value: <code>json</code>

Maximum calling frequency

200 queries/sec.

Sample request packets

Basic Form

```
{
  "RoomId": "test",
  "Admin_Account": [
    "user1",
    "user2"
  ],
  "OpType": "add"
}
```

Request packet fields

Field	Type	Attribute	Description
RoomId	String	Mandatory	Room ID
Admin_Account	Array	Mandatory	The modified administrator list can support up to 5 administrators
OpType	String	Mandatory	Operation type, add means to add an administrator, delete means to remove an administrator

Sample response packets

Basic Form

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

Response Packet Field Description

Field	Type	Description
ActionStatus	String	The result of the request process: OK indicates success; FAIL indicates failure
ErrorCode	Integer	Error code. 0: success; other values: failure

ErrorInfo	String	Error message
RequestId	String	Unique Request ID is returned with each request and required to provide this RequestId when locating issues

Error codes

Unless a network error occurs (e.g., 502 Error), the HTTP return code for this interface will always be 200. The actual error code and error information are indicated in the response payload by ErrorCode and ErrorInfo. Common Error Codes (60000 to 79999) can be found in the [Error Code](#) document. The private error codes for this API are as follows:

Error code	Description
100001	Internal server error, please retry
100002	Invalid parameter, please check the request for correctness based on the error description
100004	The room does not exist, possibly because it was never created or has already been dissolved
100020	The number of room administrators exceeds the limit. The maximum limit is 5

Error Codes (TUILiveKit)

Last updated : 2025-04-08 14:49:18

General Error Code

Error Code	Description
0	Operation Successful.
-1	Temporarily Unclassified General Error.
-2	Request Rate Limited, Please Try Again Later.
-3	Repeat Operation.
-1000	Not Found SDKAppID, Please Confirm Application Info in TRTC Console .
-1001	Passing illegal parameters when calling API, check if the parameters are legal.
-1002	Not Logged In, Please Call Login API.
-1003	Failed to Obtain Permission, Unauthorized Audio/Video Permission, Please Check if Device Permission is Enabled.
-1004	This feature requires an additional package. Please activate the corresponding package as needed in the TRTC Console .

Local User Rendering, Video Management, Audio Management API Callback Error Definition

Error Code	Description
-1100	System Issue, Failed to Open Camera. Check if Camera Device is Normal.
-1101	Camera has No System Authorization, Check System Authorization.
-1102	Camera is Occupied, Check if Other Process is Using Camera.
-1103	No Camera Device Currently, Please Insert Camera Device to Solve the Problem.
-1104	System Issue, Failed to Open Mic. Check if Mic Device is Normal.
-1105	Mic has No System Authorization, Check System Authorization.
-1106	Mic is Occupied.

-1107	No Mic Device Currently.
-1108	Failed to Obtain Screen Sharing Object, Check Screen Recording Permission.
-1109	Failed to Enable Screen Sharing, Check if Someone is Already Screen Sharing in the Room.

Room Management Related API Callback Error Definition

Error Code	Description
-2101	This Feature Can Only Be Used After Entering the Room.
-2102	Room Owner Does Not Support Leaving the Room, Conference Room Type: Transfer Room Ownership First, Then Leave the Room. Living Room Type: Room Owner Can Only Close the Room.
-2103	This Operation is Not Supported in the Current Room Type.
-2105	Illegal Custom Room ID, Must Be Printable ASCII Characters (0x20-0x7e), Up to 48 Bytes Long.
-2107	Illegal Room Name, Maximum 30 Bytes, Must Be UTF-8 Encoding if Contains Chinese Characters.
-2108	User is Already in Another Room, Single RoomEngine Instance Only Supports User Entering One Room, To Enter Different Room, Please Leave the Room or Use New RoomEngine Instance.

Room User Information API Callback Error Definition

Error Code	Description
-2200	User Not Found.

Room User Speech Management API Callback Error Definition & Room Mic Seat Management API Callback Error Definition

Error Code	Description
-2300	Room Owner Permission Required for Operation.
-2301	Room Owner or Administrator Permission Required for Operation.
-2310	No Permission for Signaling Request, e.g. Canceling an Invite Not Initiated by Yourself.
-2311	Signaling Request ID is Invalid or Has Been Processed.

-2312	Signal request repetition.
-2340	Maximum Mic Seat Exceeds Package Quantity Limit.
-2344	Mic Seat Serial Number Does Not Exist.
-2360	Current Mic Seat Audio is Locked.
-2361	Need to Apply to Room Owner or Administrator to Open Mic.
-2370	Current Mic Seat Video is Locked, Need Room Owner to Unlock Mic Seat Before Opening Camera.
-2371	Need to Apply to Room Owner or Administrator to Open Camera.
-2372	The current microphone position video is locked and needs to be unlocked by the room owner before screen sharing can be enabled.
-2373	Screen sharing needs to be enabled after applying to the room owner or administrator.
-2380	All Members Muted in the Current Room.
-2381	You Have Been Muted in the Current Room.

Definition Of Room Preload API Callback Error

Error code	Description
-4001	The current room does not support preloading.

Common Error Codes for RESTful APIs

Error code	Description
60002	HTTP parsing error, please check the HTTP request URL format.
60003	HTTP request JSON parsing error, please check the JSON format.
60004	Request URL or JSON payload contains an incorrect account or signature.
60005	Request URL or JSON payload contains an incorrect account or signature.
60006	SDKAppID invalid, please verify the validity of SDKAppID.

60007	RESTful API invocation frequency exceeds its limit, please reduce the request frequency.
60008	Service request timed out or HTTP request format is incorrect, please check and retry.
60009	Request Resource Error, please check the request URL.
60010	The request requires App administrator privileges.
60011	SDKAppID request frequency exceeds the limit, please reduce the request frequency.
60012	RESTful API requires SDKAppID, please check the SDKAppID in the request URL.
60013	HTTP response package JSON parsing error.
60014	Account Replacement timed out.
60015	The request body has the wrong account type. Please ensure the account is in string format.
60016	SDKAppID is disabled.
60017	The request is disabled.
60018	The request is too frequent, please try again later.
60019	The request is too frequent, please try again later.
60020	Your Professional Edition package has expired and been deactivated. Please log in to [Chat purchase page](https://buy.tencentcloud.com/avc) to repurchase the package. Once purchased, it will take effect 5 minutes later.
60021	The Call Source IP for RestAPI is illegal.

Error codes

Common error codes (60000 to 79999) see [Error Code](#) documentation.

The private error codes for this API are as follows:

Error Code	Explanation
100001	Server internal error, please retry.
100002	The parameter is illegal. Check whether the request is correct according to the error description.
100003	The room ID already exists. Please select another room ID.
100004	The room does not exist, or it once existed but has now been dissolved.

100005	Not a room member.
100006	Insufficient operation permissions.
100007	No payment information; you need to purchase a package in the console.
100008	The room is full.
100009	Tag quantity Exceeds Upper limit.
100010	The room ID has been used, and the operator is the room owner; it can be used directly.
100011	The room ID has been occupied by Chat. You can use a different room ID or dissolve the group first.
100012	Creating rooms exceeds the frequency limit; the same room ID can only be created once within 1 second.
100013	Exceeds the upper limit, for example, the number of microphone seats, the number of PK match rooms, etc., exceeds the payment limit.
100015	Invalid room type.
100016	This member has been banned.
100017	This member has been muted.
100018	The current room requires a password for entry.
100019	Room Entry Password Error.
100020	The admin quantity exceeds the upper limit.
100102	Signal request conflict.
100200	The mic seat is locked. You can try another mic seat.
100201	The current seat is already occupied.
100202	Already on the mic queue.
100203	Already on the mic.
100204	Not on the mic queue.
100205	The seats are all taken.
100206	Not on the mic seat.

100210	The user is already on the mic seat.
100211	The room does not support mic ability.
100251	The seat list is empty.
100400	The current connection does not exist or has ended.
100401	The room is already in connection.
100402	There is a pending connection request for this room.
100403	The current room is connecting with other rooms.
100404	The room number has exceeded the limit in connection or battle.
100405	creating connections too frequent in a short time. Wait a moment and try again.
100411	The battle does not exist or has ended.
100412	None of the rooms in the battle is valid.
100413	creating battles too frequently. Wait a moment and try again.
100414	The room isn't in the battle.
100415	The room is already in other battle.
100416	There is a pending battle request for this room.
100419	It's not allowed to cancel battle for room in battle.
100420	The battle has not started yet.
100421	The battle session has ended.
100500	The number of keys in the room's Metadata exceeds the limit.
100501	The size of value in the room's Metadata exceeds the maximum byte limit.
100502	The total size of all value in the room's Metadata exceeds the maximum byte limit.
100503	There is no valid keys when delete metadata.
100504	The size of key in the room's Metadata exceeds the maximum byte limit.

Release Notes （TUILiveKit）

iOS

Last updated : 2024-05-17 11:20:40

May 2024

Post updates	Description	Release Date
Version 1.0.0	Support custom Definition gifts Support custom Definition bullet comments Support sound effects & reverb Support likes Support seat controller	2024.05.17

Android

Last updated : 2024-05-17 11:20:40

May 2024

Post updates	Describe	Release time
Version 1.0.0	Support custom Definition gifts Support custom Definition bullet comments Support sound effects & reverb Support likes Support seat controller	2024.05.17

FAQs (TUILiveKit)

iOS

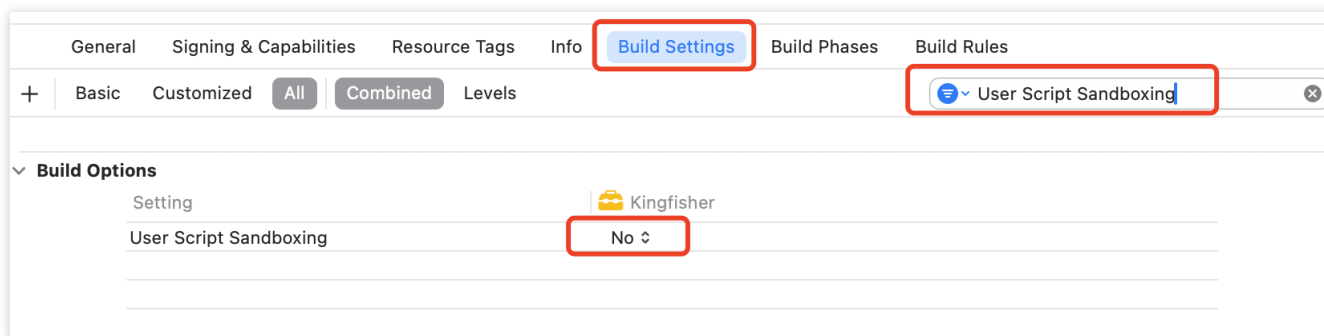
Last updated : 2024-05-17 11:48:33

Xcode 15 compiler error?

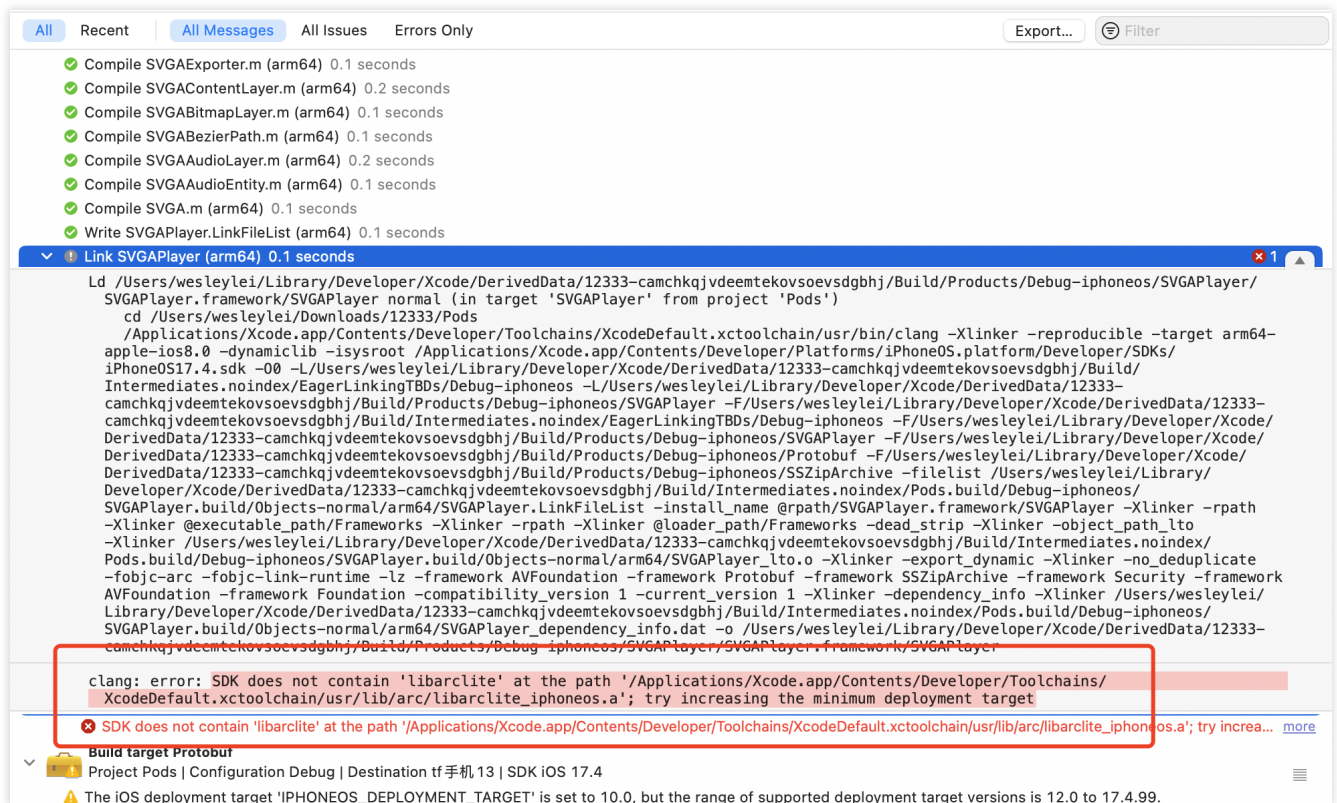
1. Sandbox: rsync is displayed.



You can set User Script Sandboxing to **NO** in **Build Settings**:



2. If SDK does not contain, compile error screenshot:



Add the following code to the Podfile:

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '13.0'
    end
  end
end
```

3. If you run the emulator on an M-series computer, **Linker command failed with exit code 1 (use-v to see invocation)** may appear.

```

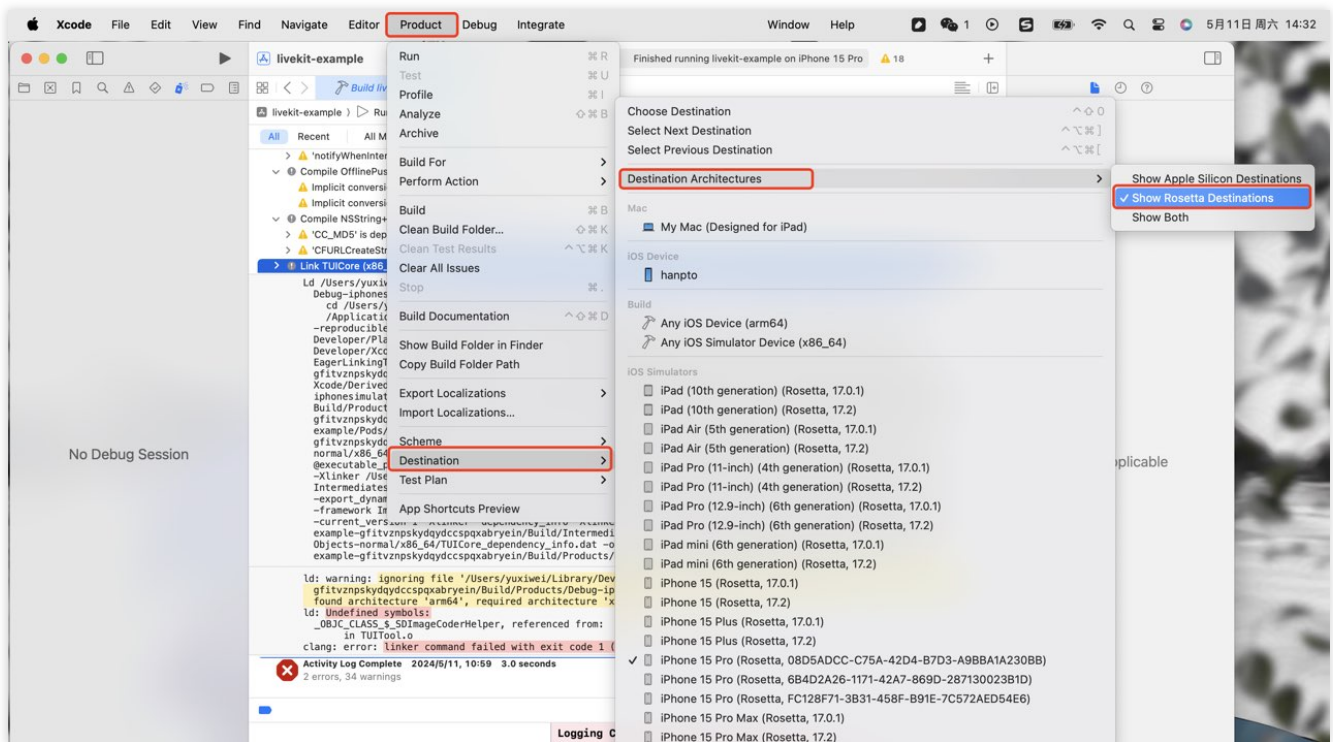
> ⚠️ 'notifyWhenInteractionEndsUsingBlock:' is deprecated: first deprecated in iOS 10.0
✓ ⓘ Compile OfflinePushExtConfigInfo.m (x86_64) 0.2 seconds ⚠️ 2
  ⚠️ Implicit conversion loses integer precision: 'NSInteger' (aka 'long') to 'int'
  ⚠️ Implicit conversion loses integer precision: 'NSInteger' (aka 'long') to 'int'
✓ ⓘ Compile NSString+TUIUtil.m (x86_64) 0.2 seconds ⚠️ 2
  > ⚠️ 'CC_MD5' is deprecated: first deprecated in iOS 13.0 - This function is cryptographically broken and should not be used in security contexts. Clie... more
  > ⚠️ 'CFURLCreateStringByAddingPercentEscapes' is deprecated: first deprecated in iOS 9.0 - Use [NSString stringByAddingPercentEncodingWithAll... more
> ⓘ Link TUICore (x86_64) 0.4 seconds ⚠️ 1 ✖️ 2
  Ld /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfityznpskydqdccspqxabyein/Build/Products/
  Debug-iphonesimulator/TUICore/TUICore.framework/TUICore normal (in target 'TUICore' from project 'Pods')
  cd /Users/yuxiwei/Downloads/livekit-example/Pods
  /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang -Xlinker
  -reproducible -target x86_64-apple-ios13.0-simulator -dynamiclib -isysroot /Applications/Xcode.app/Contents/
  Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator17.0.sdk -O0 -L/Users/yuxiwei/Library/
  Developer/Xcode/DerivedData/livekit-example-gfityznpskydqdccspqxabyein/Build/Intermediates.noindex/
  EagerLinkingTBDs/Debug-iphonesimulator -L/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfityznpskydqdccspqxabyein/Build/Products/Debug-iphonesimulator/TUICore -F/Users/yuxiwei/Library/Developer/
  Xcode/DerivedData/livekit-example-gfityznpskydqdccspqxabyein/Build/Intermediates.noindex/EagerLinkingTBDs/Debug-
  iphonesimulator -F/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfityznpskydqdccspqxabyein/
  Build/Products/Debug-iphonesimulator/TUICore -F/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfityznpskydqdccspqxabyein/Build/Products/Debug-iphonesimulator/SDWebImage -F/Users/yuxiwei/Downloads/livekit-
  example/Pods/TXIMSDK_Plus_iOS -filelist /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfityznpskydqdccspqxabyein/Build/Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/Objects-
  normal/x86_64/TUICore.LinkFileList -install_name @rpath/TUICore.framework/TUICore -Xlinker -rpath -Xlinker
  @executable_path/Frameworks -Xlinker -rpath -Xlinker @loader_path/Frameworks -dead_strip -Xlinker -object_path_lto
  -Xlinker /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfityznpskydqdccspqxabyein/Build/
  Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/Objects-normal/x86_64/TUICore.lto.o -Xlinker
  -export_dynamic -Xlinker -no_deduplicate -Xlinker -objc_abi_version -Xlinker 2 -fobjc-arc -fobjc-link-runtime
  -framework ImSDK_Plus -framework ImageIO -framework SDWebImage -framework Foundation -compatibility_version 1
  -current_version 1 -Xlinker -dependency_info -Xlinker /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-
  example-gfityznpskydqdccspqxabyein/Build/Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/
  Objects-normal/x86_64/TUICore_dependency_info.dat -o /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-
  example-gfityznpskydqdccspqxabyein/Build/Products/Debug-iphonesimulator/TUICore/TUICore.framework/TUICore

ld: warning: ignoring file '/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
gfityznpskydqdccspqxabyein/Build/Products/Debug-iphonesimulator/SDWebImage/SDWebImage.framework/SDWebImage':
found architecture 'arm64', required architecture 'x86_64'
ld: Undefined symbols:
  _OBJC_CLASS_$_SDImageCoderHelper, referenced from:
    in TUITool.o
clang: error: linker command failed with exit code 1 (use -v to see invocation)

```

✖️ Activity Log Complete 2024/5/11, 10:59 3.0 seconds
2 errors, 34 warnings

The xcode configuration needs to be modified. xcode open projects > **Product** > **Destination** > **Destination Architectures** can choose which mode of emulator to open with, and need to select the ending emulator (**Rosetta**).



Is there a conflict between TUILiveKit and the integrated audio and video library?

Tencent Cloud's audio and video libraries cannot be integrated at the same time, and there may be symbol conflicts. You can handle it according to the following scenarios.

1. If you are using the `TXLiteAVSDK_TRTC` library, there will be no symbol conflicts. You can directly add dependencies in the Podfile file.

```
pod 'TUILiveKit'
```

2. If you are using the `TXLiteAVSDK_Professional` library, there will be symbol conflicts. You can add dependencies in the Podfile file.

```
pod 'TUILiveKit/Professional'
```

3. If you are using the `TXLiteAVSDK_Enterprise` library, there will be symbol conflicts. It is recommended to upgrade to `TXLiteAVSDK_Professional` and then use TUILiveKit/Professional.

How to view TRTC logs?

TRTC logs are compressed and encrypted by default, with the extension `.xlog`. Whether the log is encrypted can be controlled by `setLogCompressEnabled`. The file name containing C(compressed) is encrypted and compressed, and the file name containing R(raw) is plaintext.

iOS : Sandbox's `Documents/log`

Note:

To view the .xlog file, you need to download the [decryption tool](#) and run it directly in the Python 2.7 environment with the xlog file in the same directory using `python decode_mars_log_file.py`.

To view the .clog file (new log format after version 9.6), you need to download the [decryption tool](#) and run it directly in the Python 2.7 environment with the clog file in the same directory using `python decompress_clog.py`.

Android

Last updated : 2024-05-17 11:48:33

Can TUILiveKit use TRTC without introducing IM SDK?

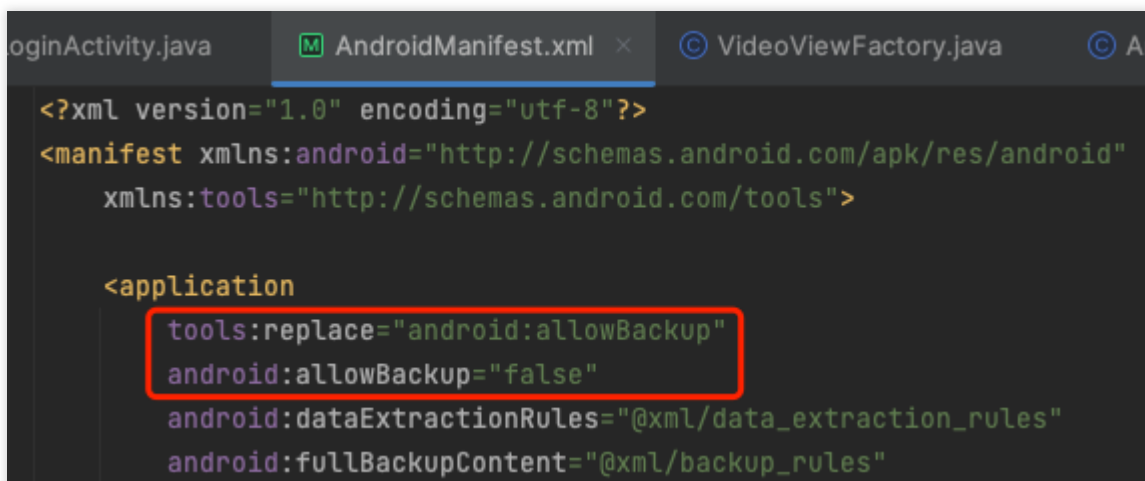
No, all the components of TUIKit use Tencent Cloud IM SDK as the basic service for communication, such as the core logic of creating room signaling, Lian-mic signaling, etc., all use IM services. If you have purchased other IM products, you can also refer to TUILiveKit logic to adapt.

Exception: allowBackup exception, How to Handle?

```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-36
is also present at [com.github.yyued:SVGAPlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(true).
Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.xml:5:5-53:19 to overr
```

Reasons : The `allowBackup` property is configured in the `AndroidManifest.xml` of several modules, causing conflicts.

Solution : You can remove the `allowBackup` attribute from your project's `AndroidManifest.xml` file or change it to false to turn off backup and restore, And add `tools:replace="android:allowBackup"` in the application node of the `AndroidManifest.xml` file; Indicates to override the settings of other modules, using your own Settings.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        tools:replace="android:allowBackup"
        android:allowBackup="false"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

Activity theme issue: activity need to use a Theme.AppCompat theme, How to Handle?


```
FATAL EXCEPTION: main
Process: com.trtc.vikit.livekit.example, PID: 15190
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.trtc.vikit.livekit.example/com.trtc.vikit.livekit.example.LoginActivity}: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3730)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3885)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:101)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2332)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:230)
    at android.app.ActivityThread.main(ActivityThread.java:8115) <1 internal line>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:526)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1034)
```

Reasons: Since `LoginActivity` inherited from `AppCompatActivity`, a `Theme.AppCompat` was to be given to `LoginActivity`.

Solution : You can add a `Theme.AppCompat` theme to the `LoginActivity` configuration in your project's `AndroidManifest.xml` file. You can also use your own `Theme.AppCompat` theme. An example of a fix is shown in the image:

```
<activity
    android:name=".login.LoginActivity"
    android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
</activity>
```