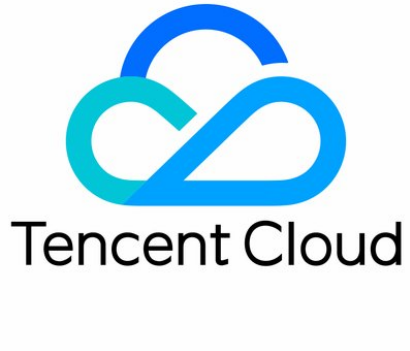
Mobile Live Video Broadcasting

Cloud Communication Integration

Product Documentation



Contents

Cloud Communication Integration
Single live broadcast

Last updated：2018-07-11 12:08:12

Single-session LVB is to broadcast one or several LVB streams related to official PGCs simultaneously. It is often used for live broadcasting of events and leader speeches. A typical example is a popular livestreaming quiz App Chongding Dahui prevailing at the end of 2017.

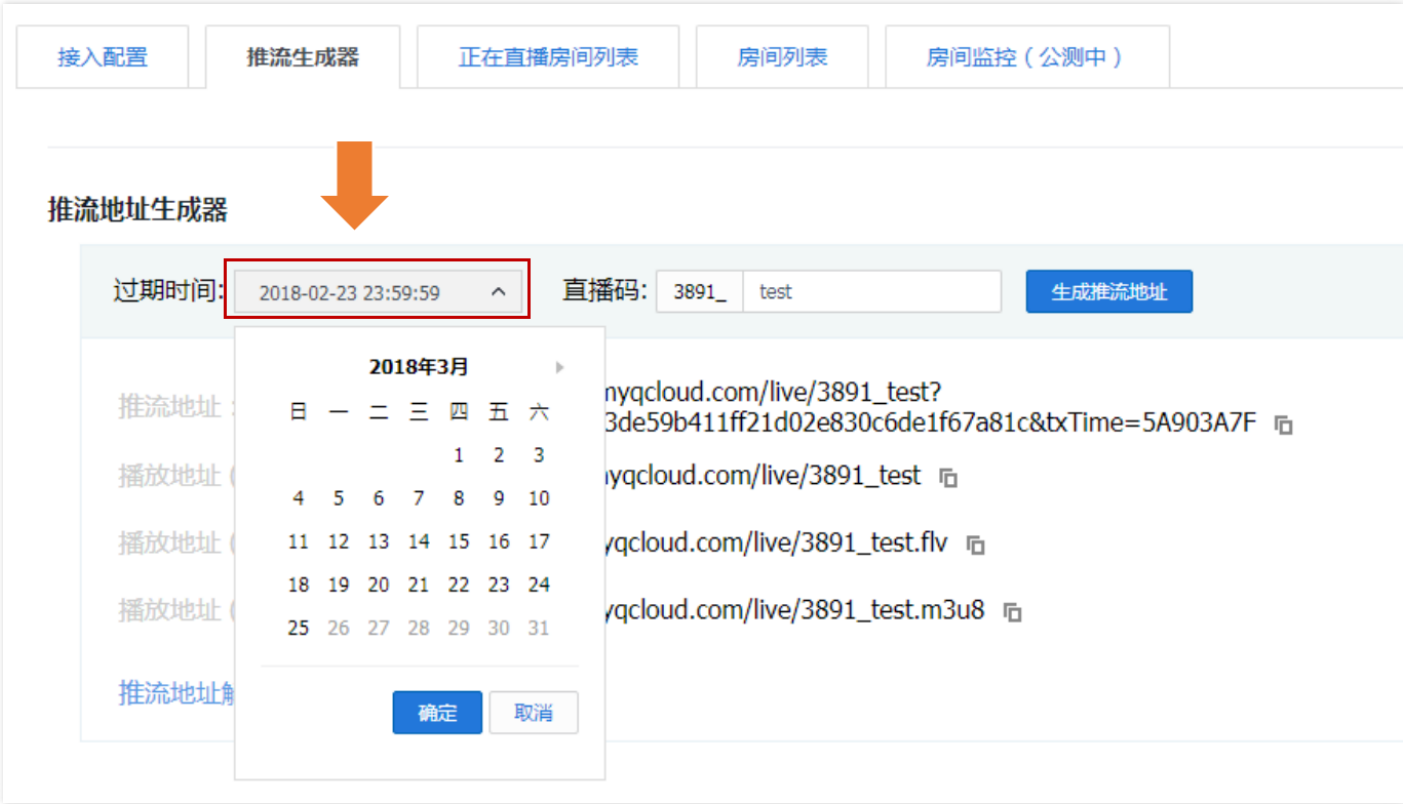
Single-session LVB is easy to integrate. As VJ (push) ends are provided with mature solutions to PGC resources, your R&D resources are used to play live audio/video streams on terminals:

Push and playback URLs

Because only a few LVB streams are broadcasted in single-session LVB, you can manually generate push and playback URLs via **LVB Console -> Access Management -> LVB Code Access -> URL Generator** . For more information, please see DOC.

Tencent Cloud customers frequently asked a question: Why can't I push streams? This is caused by two factors:

- The expiration time is too short. URLs are considered invalid after the set expiration time.



- One push URL cannot be used by two persons at the same time. Otherwise a conflict occurs.

How to push

Case 1: Push by a mobile video studio

Suitable for formal live broadcasting scenarios. Use a professional mobile video studio to interface with a PC, and then push via Obs Studio on the PC.

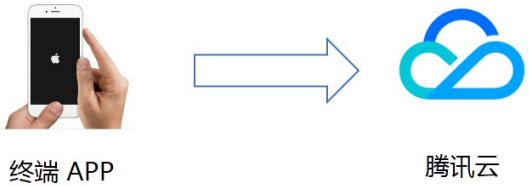
- **Advantage**: A mobile video studio makes it easy to switch to ads or play other videos during in-show time. The popular livestreaming quiz App Chongding Dahui prevailing at the end of 2017 uses this solution.
- **Reference**: For more information on push by Obs, please see DOC .

Case 2: Push by cameras

Suitable for LVB scenarios unable to be implemented in studios, such as **live broadcasting for events** and **live shows**. Connect cameras to a PC via HDMI and push by Obs Studio, or connect cameras to a video encoder and push by the video encoder.

- **Advantage**: Location-insensitive for LVB.
- **Reference**: For more information on push by Obs, please see DOC.
- **Note**: LVB requires high network quality. As networks vary depending on locations for event LVB or live show, be sure to test networks in advance and get both WiFi network and 4G mobile data ready. In addition, wired network is preferred.

Case 3: Push by mobile



With improved mobile performance, LVB on mobile is as good as the above two solutions. You can install the Video Cloud Toolkit on a mobile with good performance and start pushing by using the **RTMP push**.

- **Advantages**: Easy to use and get started, low costs.
- **Reference**: If you want to add your watermark in LVB streams, download a proper SDK and replace the demo watermark with yours. For more information on the push SDK, please see (iOS | Android).
- **Note**: Videos pushed to Tencent Cloud are smoother than those pushed to other clouds. That is because UDP protocol with strong anti-packet loss and bandwidth occupation capabilities is used by Tencent Video Cloud toolkit to push videos to Tencent Cloud while RTMP protocol is used to other clouds.

How to play

iOS player

- **Step 1**: Download Tencent Cloud SDK SDK. If push is not required, download an independent player version.
- **Step 2**: Integrate the SDK into your SDK according to the documentation TXLivePlayer .

Android player

- **Step1**: Download Tencent Cloud SDK SDK. If push is not required, download an independent player version.
- **Step 2**: Integrate the SDK into your SDK according to the documentation TXLivePlayer .

Web player

- **Interfacing guide**: Because javascript components can be directly used in Web pages, you only need to interface with the Web player according to the documentation TCPlayer .
- **High latency**: URLs with HLS (m3u8) protocol can be played by Web players on various terminals with a latency of over 20 seconds, which is greater than the latency of 2 to 5 seconds when URLs with FLV protocol are played.

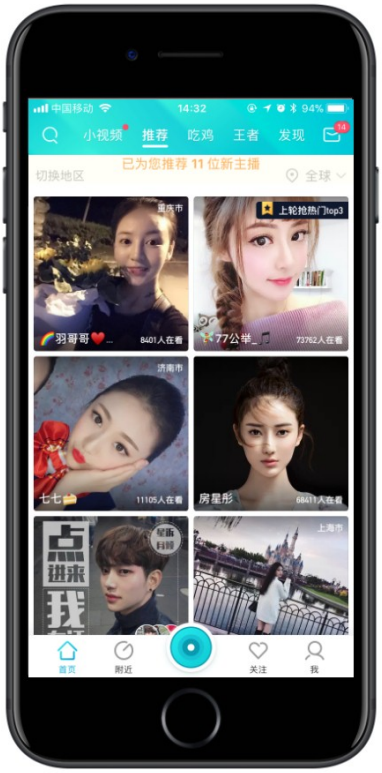
Mini Program Player

- **Specific category**: If the category of your Mini Program meets category requirements, you can use the tag <live-player&gt (generated from the simplified version of the built-in Tencent Cloud SDK) to implement LVB on iOS and Android with low latency according to DOC.
- **Other categories**: If your Mini Program belongs to any other categories, you can only implement LVB with high latency by using the tag <video> plus HLS (m3u8) protocol.

Create Room list（UGC+OGC）

Last updated：2018-07-24 18:45:50

Free-run LVB (UGC + OGC) solution means that a VJ can start live broadcasting using his/her phone at any time. This solution is adopted by a number of live broadcasting platforms such as Inke, Huajiao, Douyu and Now. Compared with the single-session LVB solution that works by manually generating one or two LVB URLs, the free-run LVB solution requires you to focus on the **room management** related logic, that is, to maintain a "room list" visible to all users.

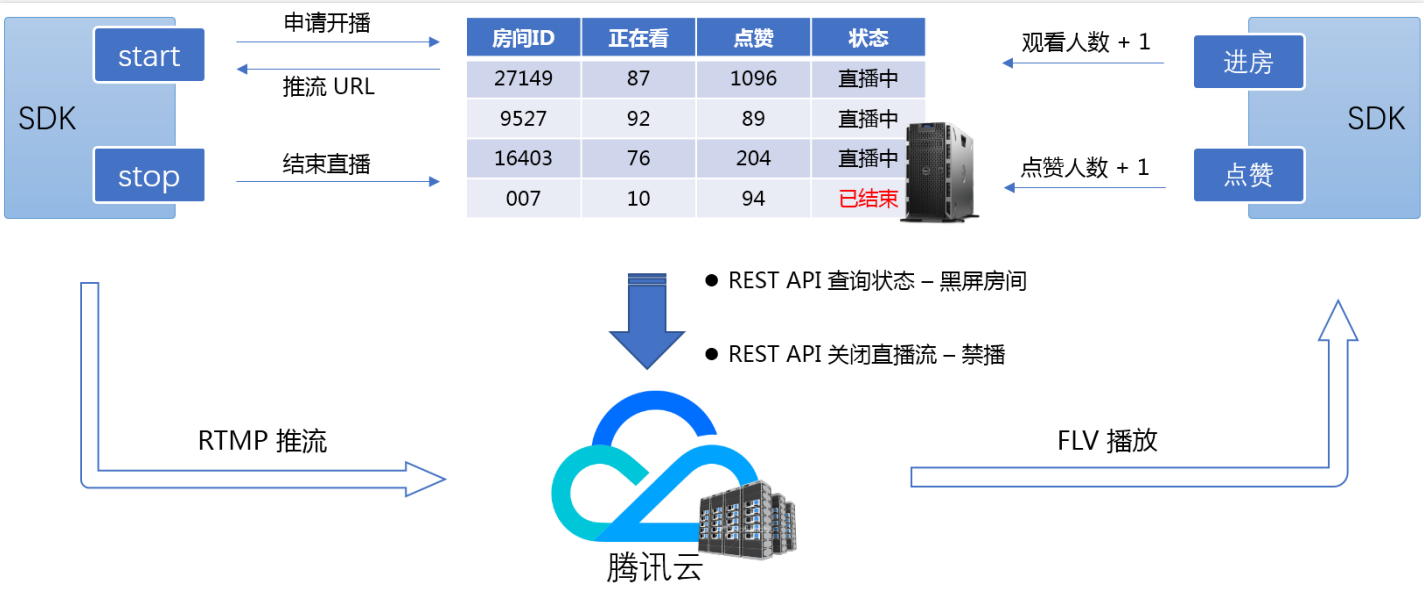


映客直播

花椒直播

斗鱼直播

The management and maintenance of a room list involves adding, deleting, modifying and querying rooms.



ADD: Create a room for broadcasting

Before starting broadcasting, a VJ needs to apply for the creation of a room, which means adding a new data item to the room list on your sever.

- **Step 1: VJ requests broadcasting (Client -> Server)**
  The Client sends the VJ account ID, room title, broadcast cover URL, geographical location (optional) and other information to your Server.

- **Step 2: Server creates a room (Server -> Client)**
  The Server adds a record in the room list and sets its status to "Waiting for broadcasting (**inactive**)", and then returns the push URL, which is required for starting broadcasting, in the response packet to the Client.

- **Step 3: VJ starts push (TXLivePusher)**
  After the Client gets the push URL and informs SDK of the URL, the SDK starts push and then notifies you through TXLivePushListener callback whether the push is successful.

- **Step 4: VJ confirms broadcasting (Client -> Server)**
  A VJ's push may not be successful. The failure of push can be caused by many reasons. For example, port 1935 used for push is disabled by the security firewall of the network, or the VJ accidentally selected "Reject" for camera authorization request when the App was just installed. Therefore, the purpose of Step 4 is to inform the backend to switch the room status from "Waiting for broadcasting (**inactive**)" to "Broadcasting...(**active**)" after the Client receives the push success event (ID: 1003) from SDK.

DELETE: Close a room

After the broadcasting is finished, the Client notifies the backend to modify the room status to "Broadcasting Over (**close**)" or delete the room from the list.

- **Step 1: VJ stops broadcasting (Client -> Server)**
  When a VJ stops broadcasting, the Client notifies the server of the ID of the live stream to be ended. Then, the server modifies the room status to "Broadcasting Over (**close**)" or delete the room from the list.

- **Step 2: Troubleshoot black-screen rooms (Server -> Tencent Cloud)**
  If the VJ is disconnected from network or the App crashes unexpectedly, the Client is unable to notify the server, leaving some dark-screen rooms in the room list (the VJ can no longer push streams and no one closes these rooms, so the viewers can only see the black screen when entering the room).

  The Server can check whether all the rooms are really in the "Broadcasting...(**active**)" status at a regular basis (preferably every 10s) through Tencent Cloud REST API (**Live_Channel_GetStatus**). If a room is "offline" in three consecutive queries, the Server considers it a "black-screen room" and closes it.

  > If the network becomes unavailable for a short while and then is restored, the SDK is reconnected to the network automatically. The room status you queried during the reconnection may not be accurate. Therefore, it is recommended to consider a room "black-screen" only if you get an "offline" result in three queries in a row.

MODIFY: Room information

In many scenarios, you need to modify the room information, for example:

- **Add viewers (Client -> Server)**
  When a new viewer joins the room, the number of viewers in the room needs to increase by 1. In this case, the viewer's App sends the Server a request for an addition to the room members.

- **Add likes (Client -> Server)**
  When a viewer gives a "like" to a VJ, the number of "likes" in the room needs to increase by 1. In this case, the App sends the Server a request for an addition of like in the response function of the "Like" button.

  > Note: A complete implementation of "like" involves broadcasting the "like" message to all the viewers through the chat room message channel.

- **Banned broadcasting due to violations (Server -> Tencent Cloud)**
  When regulators find that the broadcasting content in a room violates relevant regulations, the broadcasting in the room needs to be banned, which means the room status should be changed to "Broadcasting Over (**close**)". At the same time, your server needs to notify Tencent Cloud through REST API Enable/Disable Push to stop push immediately.

  > Note: Tencent Video Cloud's Porn Detection service is to help you identify among a number of rooms the live streams suspected of porn by capturing screenshots regularly, and to notify your backend server of the IDs of the suspected live streams via the address you specify. This service is still in Beta phase and cannot be activated by yourself. Contact us via 400 or submit a ticket if you need this service.

QUERY: Room list

Each viewer would query the current room list from backend after opening the App. Therefore, an API for fetching the list needs to be provided for the App at backend.

- **Paging logic**
  If the list contains a large number of rooms (for example, more than 100), it is recommended to add a paging logic that can help reduce server load and improve the list display speed.

- **Construct playback URL**
  With the LVB Code (or room ID), you can construct the playback URL in a straightforward way. The following shows the RTMP, FLV and HLS playback URLs constructed with the LVB Code **8888_test_12345_test**. After obtaining the playback URL, the App sends it to Tencent Cloud RTMP SDK for playback:

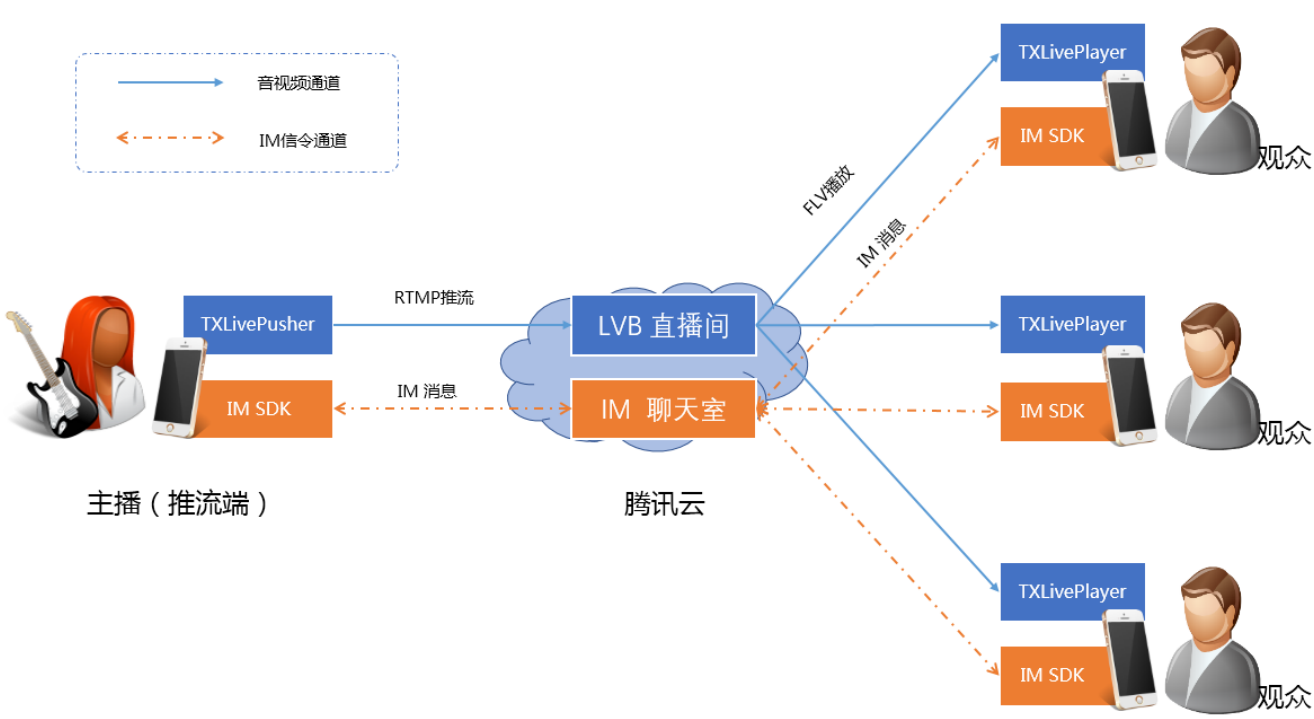
```
rtmp  rtmp://8888.liveplay.myqcloud.com/live/8888_test_12345_test
flv   http://8888.liveplay.myqcloud.com/live/8888_test_12345_test.flv
hls   http://8888.liveplay.myqcloud.com/live/8888_test_12345_test.m3u8
```

- **Don't construct playback URL on the Client**
  Playback URL is issued from the server, instead of being constructed on the Client. This can make your system more flexible. With the growth of your business, you may consider adding playback hotlink protection at the viewer end to prevent your video data from being hacked. However, hotlink protection signature can only be issued from the server, so constructing URL on client makes no sense in this respect.

IM Chatroom

Last updated：2018-07-10 14:49:17

There is a close relationship between online LVB and IM chat rooms. Without the interactive messages in the chat rooms, the LVB will become boring and dull. This document describes how to use **Tencent Cloud Instant Messaging (IM)** service to build simple chat room features:
(1) Broadcast on-screen comments and likes.
(2) Broadcast system notifications, for example "XXX has joined the room" or "The VJ has left".



1. Tencent Cloud IM Service

The predecessor of Tencent Cloud IM is QQ's instant messaging system. We remove the QQ message module and change it to IM SDK which is more suitable for mobile access. The message backend has been transformed so that it was unbound with the QQ number, constituting the current IM backend.

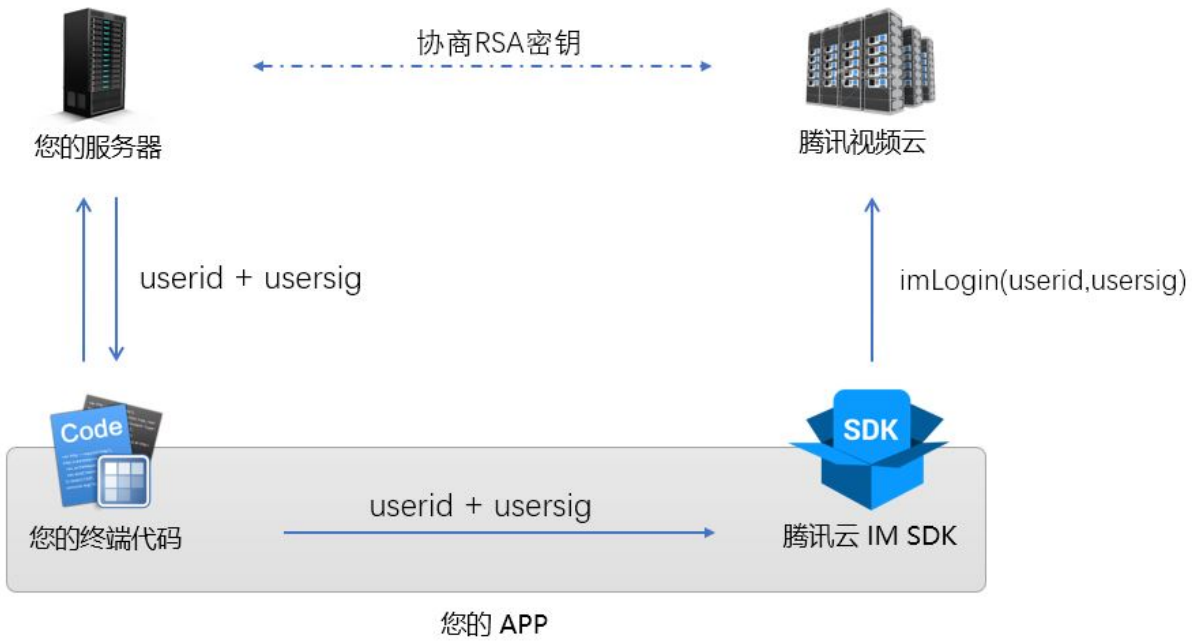
IM SDK can be considered as the QQ without user interaction page. Integrating the IM SDK into your App is like integrating a QQ message kernel.

As we all know, QQ can receive and send messages of private chats and group chats, but you need to log in to it before you can use it. Logging in to QQ requires a QQ account and password. Similarly, logging in to IM SDK needs a user-specified username (userid) and password (usersig).

- **User name (userid)**
  This can be the user ID in your current application. For example, if the account ID of a user is 27149, then he/she can use 27149 as the userid to log in to the IM SDK.

- **Password (usersig)**
  Since you specified that 27149 is your user, how can Tencent Cloud confirm that the user is a valid user you have approved? Usersig is used to solve this problem. Usersig is the asymmetric encryption result of userid, appid, and other information.

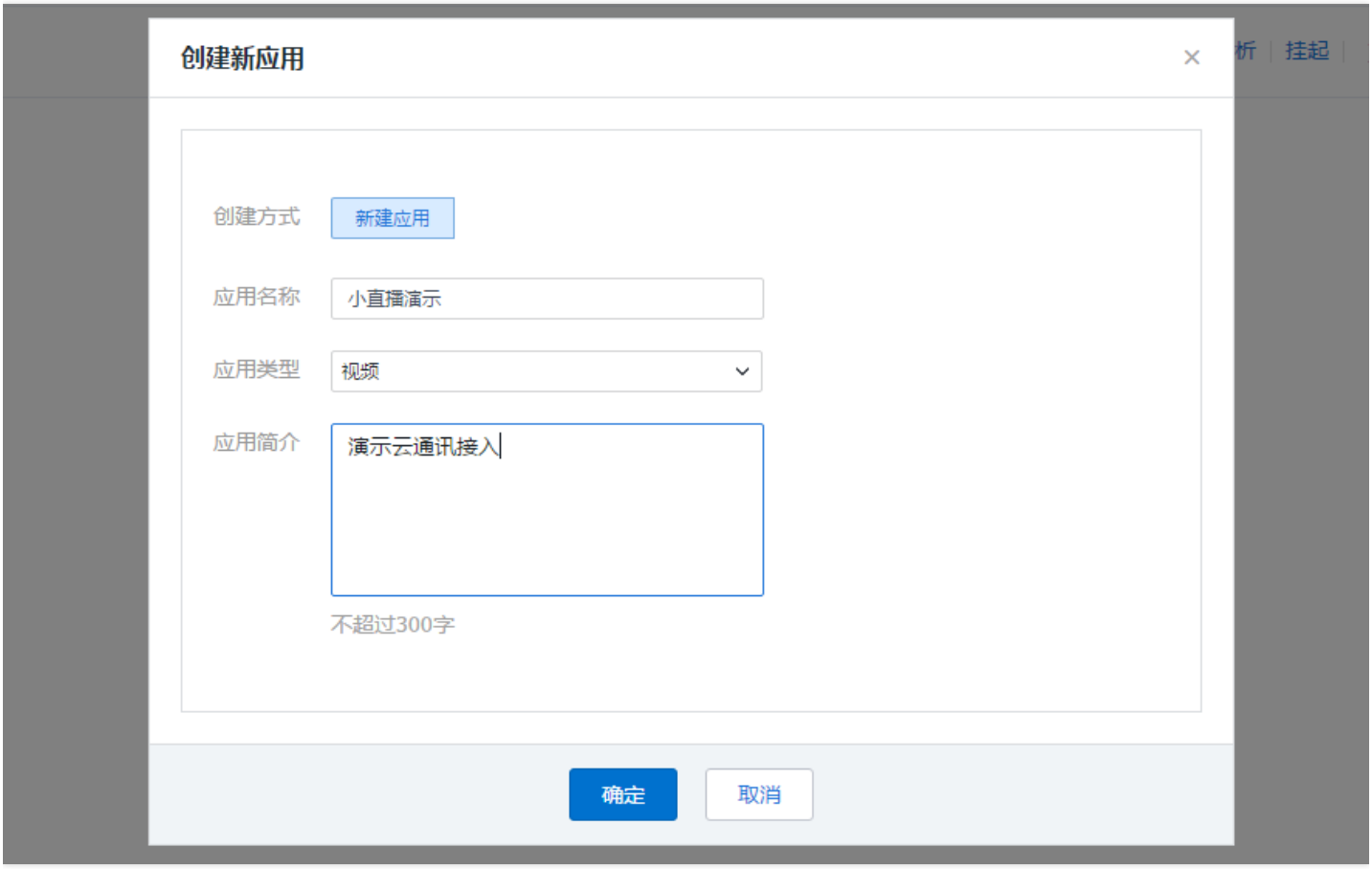
  The encryption key and the decryption key used in asymmetric encryption are different. The private key can be kept on your server to asymmetrically encrypt the userid and appid to generate the usersig. At the same time, Tencent Cloud keeps the corresponding public key to verify whether the usersig is valid and is signed by your server.



2. Service Activation and Configuration

Log in to the IM Console. If you have not activated the service, click the **Activate IM** button. For a new Tencent Cloud account, the IM App list is empty, as shown below:

Click the **Create Application Access** button to create a new application access, that is, the name of the application for which you want to get the access to Tencent Cloud IM service. Our test application is called "Small LVB Demo", as shown below:



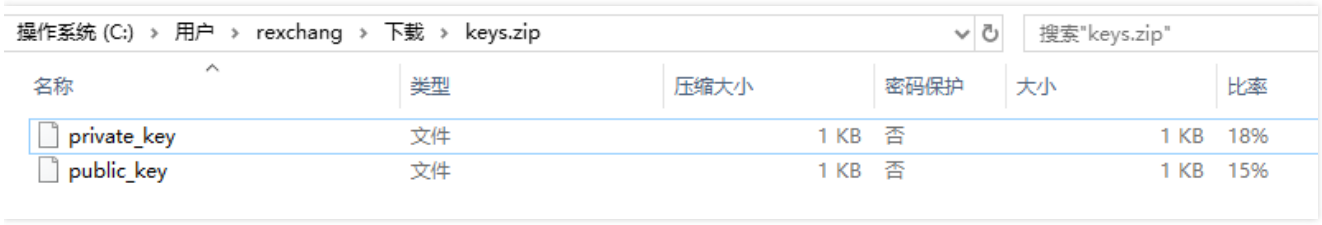
Click the **OK** button, and then you can see in the application list the item you just added, as shown below:

Click the **Application Configuration** link, you will enter the application configuration interface, and then click the **Edit** button on the right side of the **Account System Integration**, and you can configure as shown below (The account name and administrator name are recommended in English. You can enter any account name you like. The administrator account will be used when

calling IM's REST API).

Click the **Save** button, the page will automatically refresh, and then you can see the **Download Public and Private Keys** button.

Click the **Download Public and Private Keys** button to get a zip file called **keys.zip** which has a private_key and a public_key. The **private_key** is the private key used to sign UserSig.



You can test whether private_key can perform the signature normally in the **Development Tools**.

3. Server Access Guidelines

3.1 Distribute the UserSig

UserSig is distributed to the Client in a simple way through the backend server. By clicking a few mouse clicks in the previous step, we have obtained the public and private keys used to sign UserSig.

Next, you can read the DOC to understand the UserSig generation code for language versions (Java, PHP, C++) and then integrate them into your backend system.

The recommended practice is to integrate them into the login process, that is, when the user logs in, your backend server returns UserSig to your application in addition to the original information.

3.2 Call the REST API

You can also perform secondary development by calling REST API, such as:

API	Description	Purpose
v4/group_open_http_svc/create_group	Create groups	The chat room corresponding to an LVB can be created by the VJ on the client side using the IM SDK, or can be created by your server using this API
v4/group_open_http_svc/send_group_msg	Send messages	Messages can be pushed to each member of the group using this API
v4/openim_dirty_words/add	Add dirty words	Used for sensitive words filtering.

4. Client Access Guidelines

4.1. Download SDK

You can download the latest version of IM SDK from the IM official website. We advise you to download the v3 version, which is less difficult to access than v2.

4.2 IM login (imLogin)

The userSig signed by your server can be returned to the application when the application connects to the server. After the application gets the userid and usersig, you can log in to Tencent Cloud IM service to send and receive messages (Similarly, you can log in to the QQ backend to send and receive messages after entering the correct QQ number and password). The example code is as follows:

```
// iOS example code: imLogin
//
#import "ImSDK/ImSDK.h"
TIMLoginParam *param = [[TIMLoginParam alloc] init];
// identifier is the user name, userSig is the user login credential, and the calculation results of the server are as follows
param.identifier = _config.userID;
param.userSig = _config.userSig;
param.appidAt3rd = [NSString stringWithFormat:@"%d", _config.appID];

[[TIMManager sharedInstance] login:param succ:^{
//Logged in successfully
} fail:^(int code, NSString *msg) {
//Failed to log in
}];


// android example code: imLogin
//
TIMUserConfig timUserConfig = new TIMUserConfig();
TIMManager.getInstance().login(identifier, userSig, timUserConfig, new TIMCallBack() {
@Override
```

```
public void onError(int code, String desc) {
//"code" (error code) and "desc" (error description) can be used to locate the reason for the failure
//For the list of "code" (error code), please see the error codes
Log.d(tag, "login failed. code: " + code + " errmsg: " + desc);
}

@Override
public void onSuccess() {
Log.d(tag, "login succ");
}
});
```

4.3 Join a chat room

The IM SDK can only send and receive broadcast messages after joining a specific chat room. The sample code for joining the chat room is as follows:

```
// iOS example code: Join a chat room
//
[[TIMGroupManager sharedInstance] joinGroup:@"TGID1JYSZEAEQ" msg:@"Apply Join Group" succ:^(){
NSLog(@"Join Succ");
}fail:^(int code, NSString * err) {
NSLog(@"code=%d, err=%@", code, err);
}];
```

```
// Android example code: Join a chat room
//
TIMGroupManager.getInstance().applyJoinGroup(groupID, "" , new TIMCallBack() {
@Override
public void onError(int i, String s) {
Log.d(TAG,"Failed to join the group");
}

@Override
public void onSuccess() {
Log.d(TAG,"Joined the group"+groupID+" successfully");
}
});
```

4.4 Send group chats

```
// iOS example code: Send group chats
//
TIMTextElem * text_elem = [[TIMTextElem alloc] init];
[text_elem setText:@"this is a text message"];
TIMMessage * msg = [[TIMMessage alloc] init];
[msg addElem:text_elem];
[conversation sendMessage:msg succ:^(){
NSLog(@"SendMsg Succ");
}fail:^(int code, NSString * err) {
NSLog(@"SendMsg Failed:%d->%@", code, err);
}];
```

```
// Android example code: Send group chats
//
TIMTextElem textElem = new TIMTextElem();
textElem.setText(msgText);
TIMMessage message = new TIMMessage();
message.addElement(textElem);
TIMConversation conversation = TIMManager.getInstance().getConversation(TIMConversationType.Group, groupId);
conversation.sendMessage(message, new TIMValueCallBack<TIMMessage>(){
@Override
public void onError(int i, String s) {
Log.d(TAG, "Failed to send broadcast messages");
}
@Override
public void onSuccess(TIMMessage timMessage) {
Log.d(TAG, "Broadcast messages sent successfully");
}
});
```

4.5 More APIs

Only the most commonly used APIs are introduced here. For more APIs of IM SDK, please see the Documentation of Tencent Cloud IM service.

LiveRoom Module

Last updated：2018-08-06 10:30:40

Feature Description

LVB+Joint Broadcasting is an LVB mode commonly used in the **Live Show** and **Online Education** scenarios. With a good applicability to many scenarios, it supports online live broadcasting featuring both high concurrency and low cost, but also enables video chats between VJs and viewers via joint broadcasting.

LiveRoom

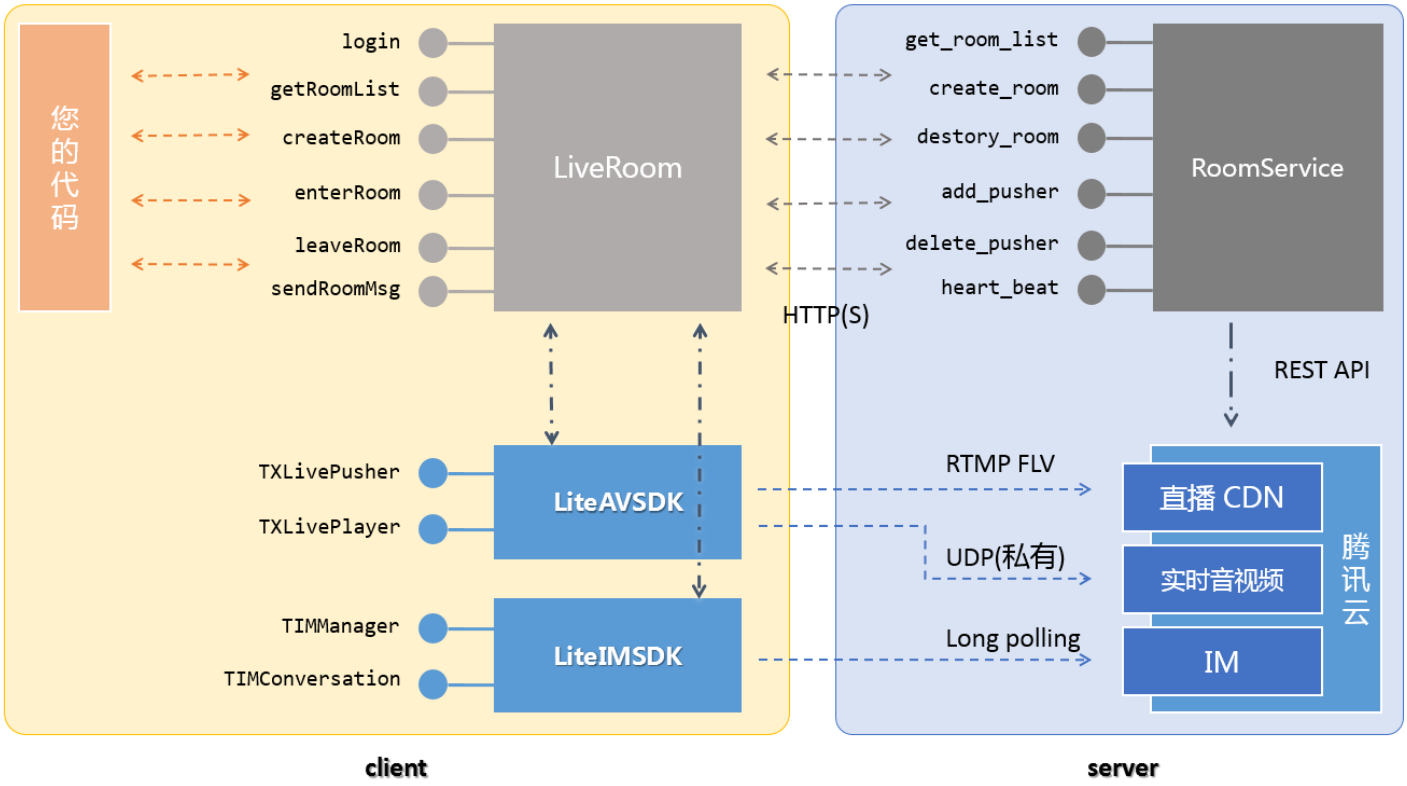
If you only need a simple LVB solution that allows VJs broadcasting, viewers watching the broadcasting, and text interaction, please see Single-session LVB or Free-run LVB.

However, if you want to go with joint broadcasting, it's not easy to implement it by only using TXLivePusher and TXLivePlayer. To solve this problem, **LiveRoom**, a component consisting of Client and Server, is provided:

- **Client (Terminal)**
  LiveRoom component on Client encapsulates Tencent Video Cloud LiteAVSDK (used for audio/video and includes TXLivePusher and TXLivePlayer APIs) and LiteIMSDK (used for sending and receiving messages and includes TIMManager and TIMConversation APIs). It takes too much time and effort to perform LVB and joint broadcasting by directly using LiteAVSDK

and LiteIMSDK. LiveRoom makes it very easy for you to use LVB and joint broadcasting by calling createRoom, enterRoom, and leaveRoom.

- **Server (Backend)**
  RoomService is LiveRoom's background component. It has two responsibilities: The first is to manage rooms (add, delete, modify or query rooms) and viewers (especially when a joint broadcasting with VJ involves multiple viewers); the second is to control the Tencent Cloud LVB, Tencent-RTC and IM services (by calling Tencent Cloud backend REST APIs).



Interfacing with terminal

Step 1: Download the SDK for your platform.

Platform	Programming Language	SDK Download	API Documentation
iOS	Objective-C	DOWNLOAD	API documentation
Android	java	DOWNLOAD	API documentation
IE browser	javascript	DOWNLOAD	API documentation
PC(C++)	C++	DOWNLOAD	API documentation

- LiteAV SDK in the decompressed SDK folder is used to implement audio/video related features.
- LiteIM SDK in the decompressed SDK folder is used to implement IM related features.
- LiveRoom in the decompressed Demo\liveroom folder provides open source code to make it easy for you to debug and customize as needed.

Step 2: Log in to RoomService (login)

LiveRoom cannot run by relying on the component on Client alone. It also needs a background service, which is called **RoomService**, to manage rooms and coordinate status. To use RoomService, LiveRoom must **log in** to RoomService.

See RoomService to learn how to enter the parameters of the login function.

Step 3: Get room lists (getRoomList)

Both VJs and viewers (teachers and students) need room lists. You can get room lists by calling **getRoomList** in LiveRoom. Each room in a list has roomInfo. When you create a room, it is recommended that you define roomInfo in JSON format for high scalability.

> Skip this step if you want to use your own room list. But you need to specify roomID in Step 4, and the roomID must be unique.

Step 4: VJs start broadcasting (createRoom)

To start broadcasting, call the API **startLocalPreview** in LiveRoom to enable local camera preview. A "view" object needs to be specified to display video images in the camera. During the period, LiveRoom requests camera permissions and VJs look at cameras to adjust beautifying and whitening effects.

Then, a room is created in the room list in the background by calling the API **createRoom** of LiveRoom, and the VJ goes into the push mode.

> **Parameter roomID**
> If you do not enter the roomId when calling createRoom, a roomID will be automatically assigned to you at backend via the callback API of createRoom. If you want to manage the room list on your own, the roomID can be assigned by your server. You only need to enter the assigned roomID when calling createRoom.

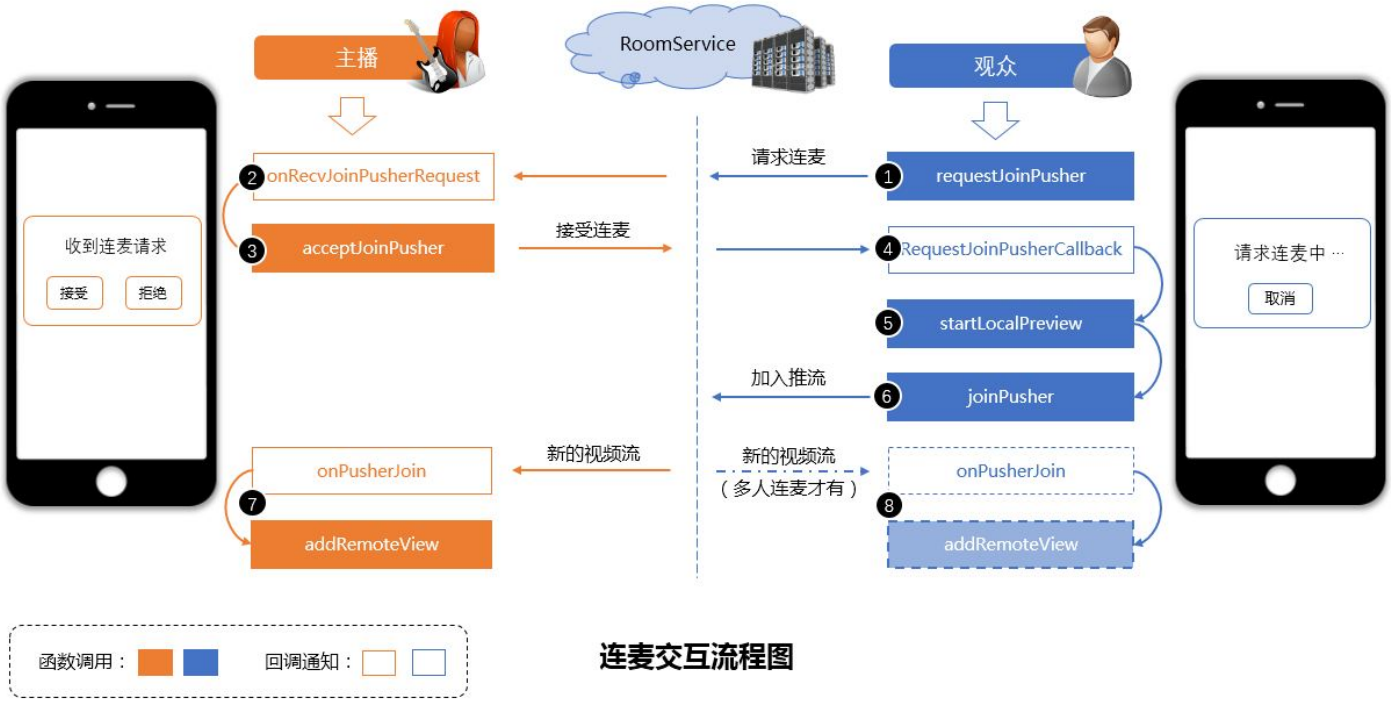
Step 5: View LVB (enterRoom)

Viewers can enter live rooms and view LVB via **enterRoom** in LiveRoom. A "view" object needs to be specified for enterRoom to display video images in the LVB stream.

Besides, after viewers enter a room, you can get the list of viewers by calling **getAudienceList** in LiveRoom. The list may not contain all the viewers. If the number of viewers is less than 30, all of them will display in the list. Otherwise, the last 30 viewers will display in the list. (For the sake of performance, at most 10 profile photos can display in the screen.)

Step 6: Joint broadcasting (joinPusher)

Joint broadcasting is a process that requires the participation of both a VJ and viewers, as shown below:



- 1 (viewer): Initiates a request for joint broadcasting through **requestJoinPusher**.
- 2 (VJ): Receives the callback notification from **onRecvJoinPusherRequest**. A prompt asking whether the VJ to accept the request appears.
- 3 (VJ): Selects **acceptJoinPusher** to accept or **rejectJoinPusher** to reject the request.
- 4 (viewer): Learns whether his request is accepted or not through **RequestJoinPusherCallback**.
- 5 (viewer): Enables his local camera by calling **startLocalPreview** if his request is accepted. The SDK applies permissions for the viewer's camera and microphone as startLocalPreview needs a view object to display the image of the camera.
- 6 (viewer): Goes into the joint broadcasting mode (push on viewer side) by calling **joinPusher**.
- 7 (VJ): Once the viewer is ready for joint broadcasting, the VJ receives a notification of **onPusherJoin**, which encapsulates the viewer's information into the "pusherInfo" object. Next, the VJ displays the remote image of the viewer on the screen through the **addRemoteView** function.
- 8 (viewer): If one or more viewers are under joint broadcasting, the viewer end will also receive the onPusherJoin notification, and the images of other viewers will display on the screen via addRemoteView.

> You can ignore the last step if you want a 1v1 joint broadcasting.

Step 7: On-screen comment (sendMsg)

LiveRoom provides message delivery APIs. You can send ordinary text messages (on-screen comments) via **sendRoomTextMsg** and custom messages (such as giving a like, or sending a flower) via **sendRoomCustomMsg**.

You can receive text and custom messages sent from others in the chatroom via **onRecvRoomTextMsg** and **onRecvRoomCustomMsg** in RoomListenerCallback.

ATTENTION

You can receive at most 40 messages in one second via Tencent Cloud IM. If you refresh all of them to the screen based on the receipt frequency, LVB stuttering will occur. Mind the refresh frequency.

That is the reason why some customers gain very smooth experience in trial but suffer serious stuttering after release.

Interfacing with backend

Why need to log in to LiveRoom?

LiveRoom cannot run by relying on the component on Client alone. It also needs a background service, which is called **RoomService**, to manage rooms and coordinate status. To use RoomService, LiveRoom must **log in** to RoomService.

How do I enter parameters related to login?

The parameters should be entered based on application scenarios, as shown below. Solution 1 is intended for debugging, solution 2 for quick go-live, and solution 3 for customization.

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
serverDomain	Use Tencent Cloud RoomService `https://room.qcloud.com/weapp/live_room`	Configure in advance before using Tencent Cloud RoomService `https://room.qcloud.com/weapp/live_room`	Self-deployed RoomService `https://[yourcompany]/weapp/live_room`
sdkAppID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
accType	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
userID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined, such as 9527	User-defined, such as 9527
userSig	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	Generated by your server. How to generate it?	Generated by your server. How to generate it?
Account owner	Tencent Cloud test account	Your account	Your account

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
Account limits	Available from 10:00 to 22:00	No limit. Customization is not supported.	No limit. Customization is supported.
Application scenarios	Debugging by Terminal Development team before go-live	Early stage after go-live	Growth stage

Solution 1: For testing only

This solution uses the test accounts provided by Tencent Cloud for debugging. These accounts are only available from 10:00 to 22:00 during the debugging period.

Step 1: Configure RoomService

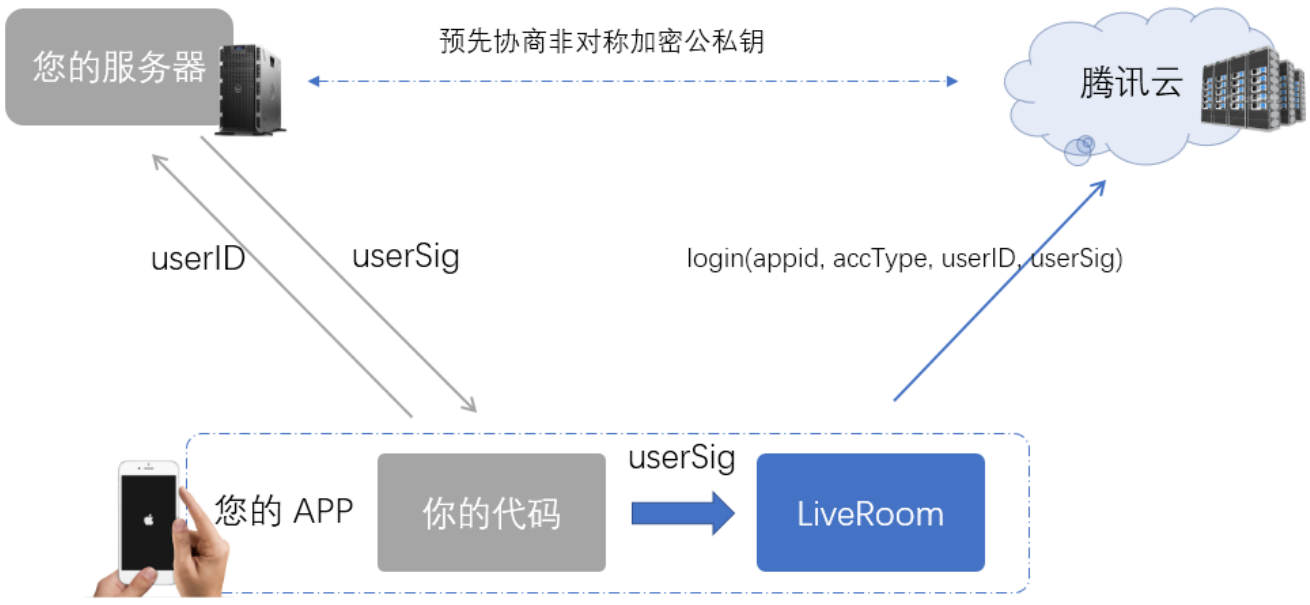
For serverDomain, enter `https://room.qcloud.com/weapp/live_room` .

Step 2: Get parameters required for login

Get relevant parameters from the testing URL (`https://room.qcloud.com/weapp/utils/get_login_info_debug`).

Solution 2: Use Tencent Cloud RoomService

This solution uses your Tencent Cloud account and Tencent Cloud RoomService. You need to configure RoomService before implementing this solution.



Step 1: Configure RoomService

The URL of Tencent Cloud RoomService is:

```
https://room.qcloud.com/weapp/live_room
```

Click RoomTool.zip to download the backend configuration tool for Tencent Cloud RoomService. This tool is based on Node.js and can only be used when Node.js has been installed. The PDF and PPT files in the configuration tool package describe in detail how to configure the RoomService. The following provides an overview of the configuration items.

Configuration Item	Description	Obtained From
LVB appID	Identifies customers in Tencent Cloud LVB	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC
IM sdkAppID	Identifies IM customers in Tencent Cloud IM service	DOC
IM accountType	Used to identify App type and now is retained for compatibility.	DOC
IM administrator	Required for IM REST API in RoomService to send room's system messages.	DOC
IM privateKey	Used by RoomService to issue the administrator's usersig to call IM REST API to send room's system messages.	DOC
IM publicKey	Used by RoomService to verify login user's identity.	DOC

Step 2: Get parameters required for login

RoomService can only be used when `login(serverDomain, sdkAppID, accType, userID, userSig)` is called successfully on terminal. The first four parameters can be hard-coded at Client, but UserSig must be issued by your backend server. If UserSig is calculated at Client, the signature keys need to be written in the terminal code, causing the risk of being hacked.

RoomService and IM use the same UserSig issued by your server, so a UserSig can be used to log in to both IM and RoomService. For more information on how to use it, please see issuing UserSig.

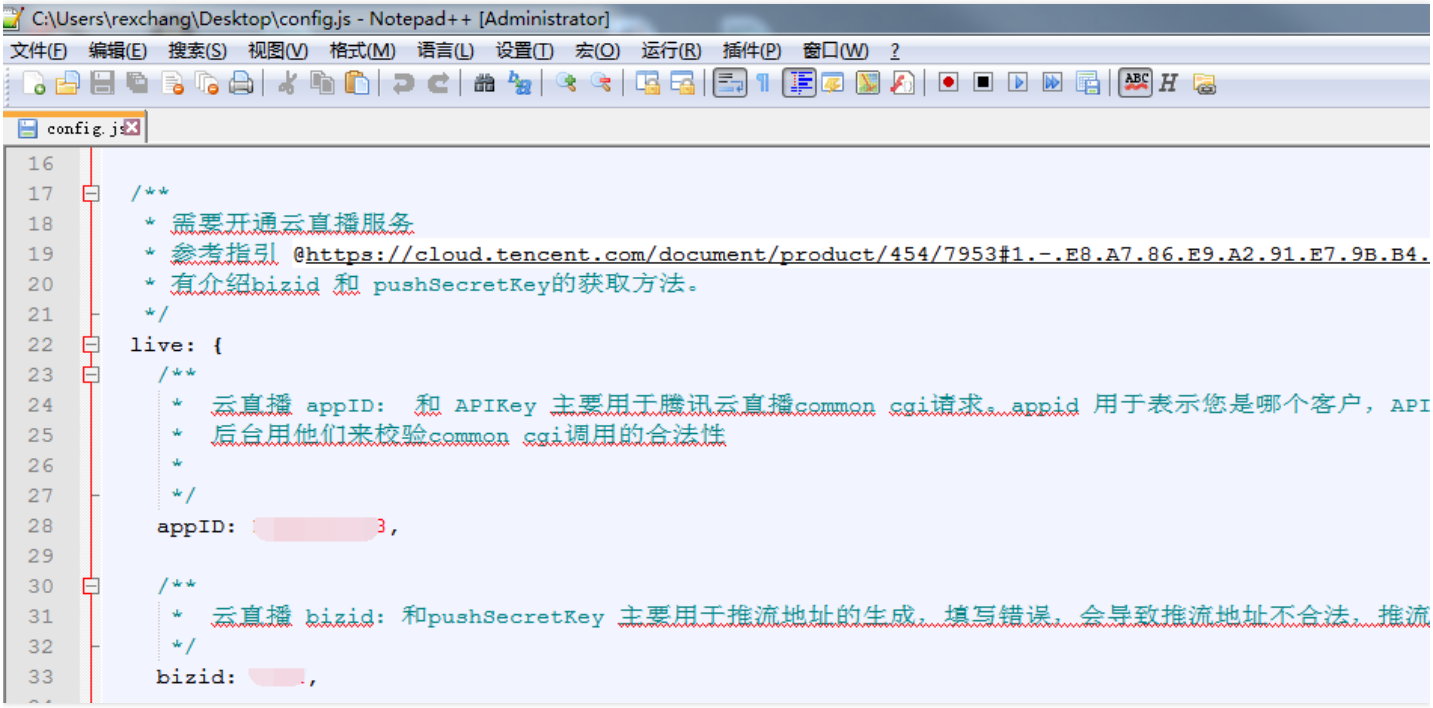
Solution 3: User-deployed RoomService

This solution uses your Tencent Cloud account and the RoomService you deploy, allowing you to modify and customize the internal logic as needed.

Step 1: Download source code, modify configuration, and deploy RoomService

Download RoomService backend source code from CODE. The source code package is composed of three directories, among which live_room contains the source code you need.

After downloading the source code, decompress the package, locate the file config.js under the folder live_room, and then modify some configuration items. The configuration items are similar to those in Solution 1, except that you can modify local source code directly without using the configuration tool for RoomService. You can configure them by referring to Step 1 in Solution 1.



Configuration Item	Description	Obtained From
LVB pushSecretKey	Required for calculating the hotlink protection signature of a push URL	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC

Then, you can deploy RoomService on your backend server and notify your terminal development engineer of the public network URL of your server, so that he/she can specify the backend URL for RoomService when calling the parameter login on terminal. For example:

```
https://[www.yourcompany.com]/weapp/live_room
```

Step 2: Get parameters required for login

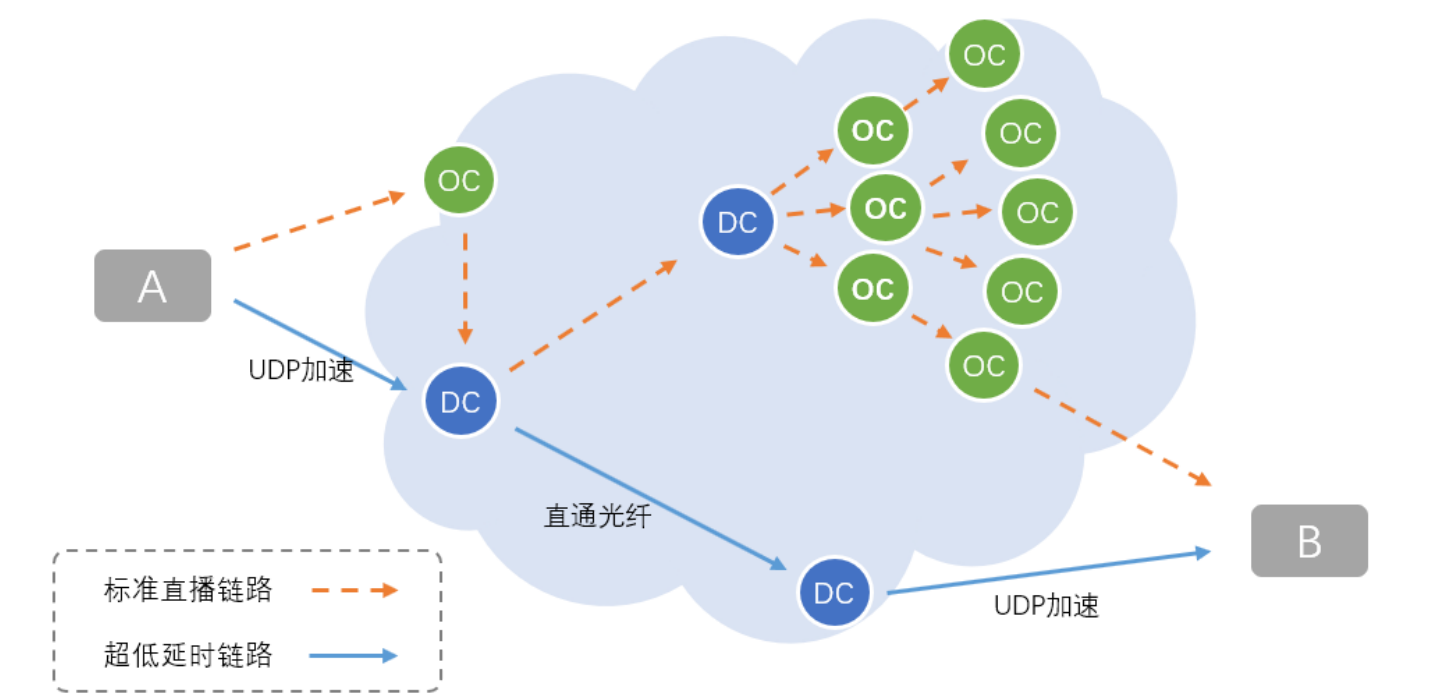
After Step 1, you have created your RoomService, which also authenticates user identity based on `sdkAppID, accType, userID and userSig` (or another authentication method you desire). Similarly, you need to assign a login signature to Client as described in Generating UserSig.

> In the backend source code of RoomService, the function getSig in `logic\im_mgr.js` generates the UserSig in the sample code in node.js. The sample code in Java and PHP will be available soon.

How It Works

1. Two "paths"

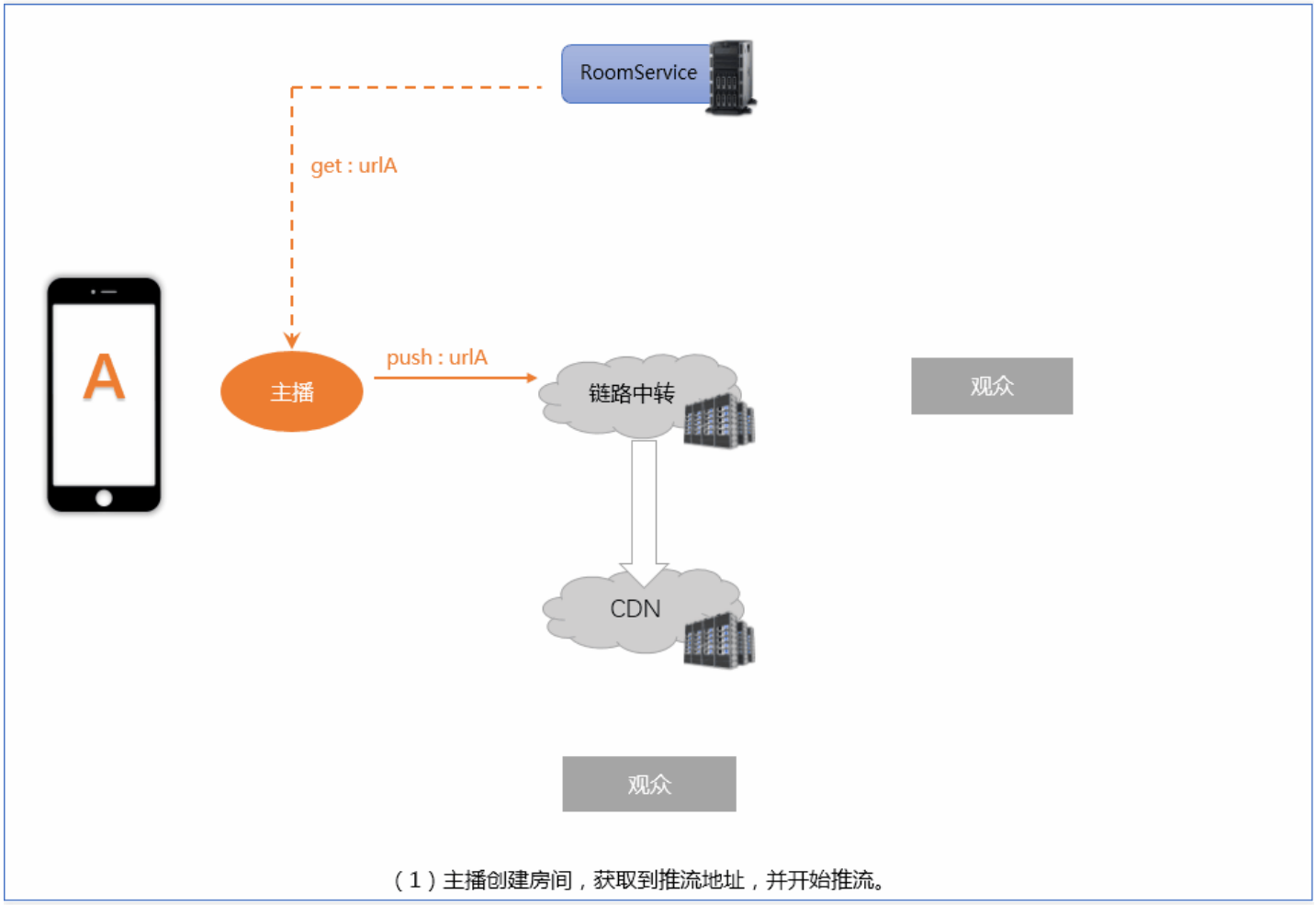
Tencent Cloud uses two paths to implement LVB plus joint broadcasting. LVB is implemented using standard RTMP and FLV protocols over standard CDN, which allows as many viewers as possible to watch the live show with very low bandwidth costs. However, the latency is greater than 2 seconds. Joint broadcasting is implemented using UDP protocol over special Direct Connect. With a latency of around 500 ms, joint broadcasting allows at most 10 people to enable video chat simultaneously, with the cost of each push higher than that of ordinary LVB.



Path	LVB Path	Joint Broadcasting Path
Latency	>=2s	Around 500 ms
Underlying protocol	RTMP/HTTP-FLV	Private UDP protocol
Price/Fee	low	Each push higher than LVB
Maximum number of concurrent viewers	No upper limit	<=10
TXLivePusher	SD, HD, and FHD are available for setVideoQuality	MAIN_PUBLISHER and SUB_PUBLISHER are available for setVideoQuality
TXLivePlayer	PLAY_TYPE_LIVE_FLV	PLAY_TYPE_LIVE_RTMP_ACC
Playback URL	Ordinary FLV URLs	RTMP URLs with hotlink protection signatures

2. Function principle

You can integrate LiveRoom without having to understand the function principle. If you are interested, the following figure provides a general picture on how it works.



（1）主播创建房间，获取到推流地址，并开始推流。

PK

Last updated：2018-08-06 10:31:36

Feature Description

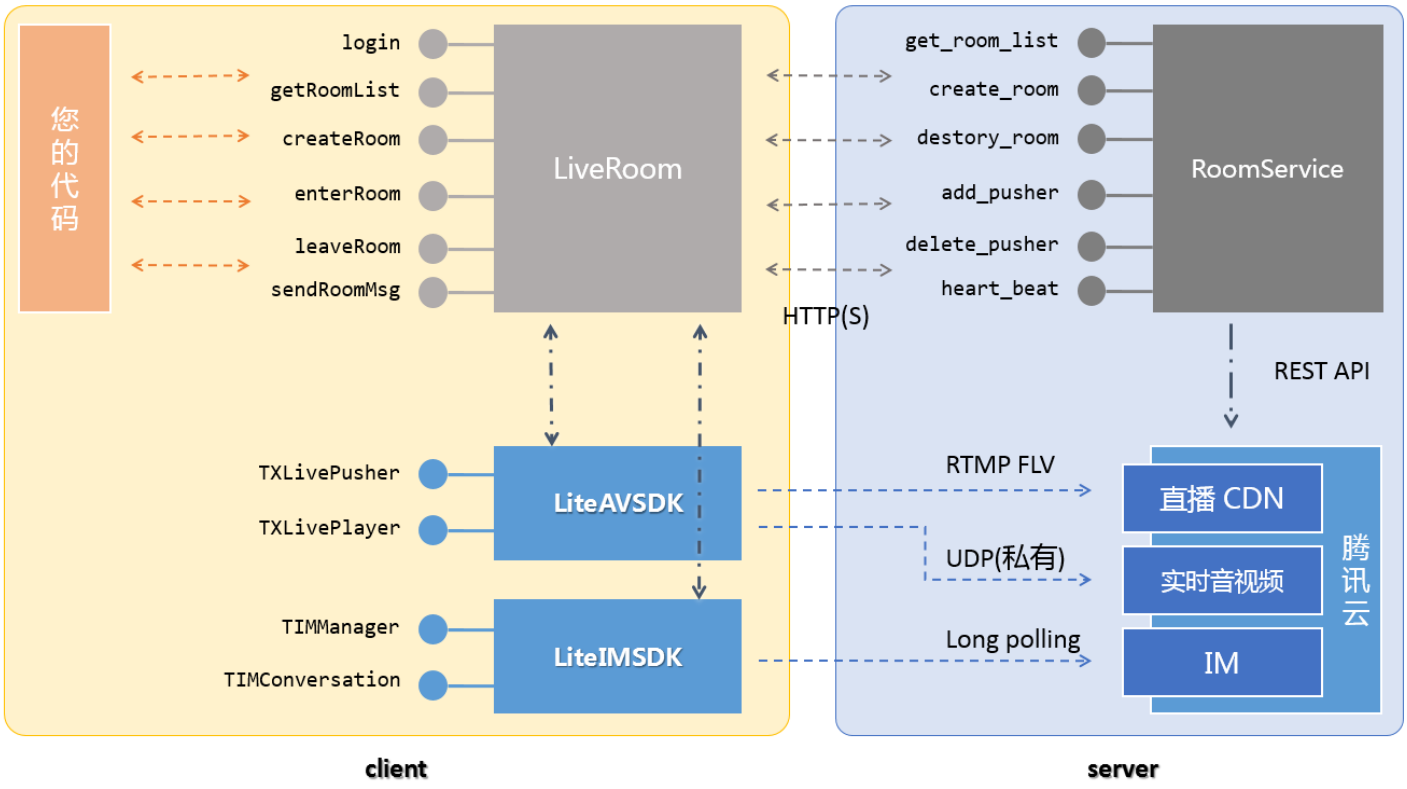
VJ PK is often used in live shows to attract viewers. This feature allows viewers to watch two VJs in different rooms perform joint broadcasting (video chat) in a split screen from the original CDN LVB stream without having to switch to another URL. Besides, the latency between the two VJs is reduced to 500 ms.

LiveRoom

In addition to VJ PK, **LiveRoom** also provides joint broadcasting between viewers and VJs. LiveRoom involves a Client and a Server:

- **Client (Terminal)**
  LiveRoom component on Client encapsulates Tencent Video Cloud LiteAVSDK (used for audio/video and includes TXLivePusher and TXLivePlayer APIs) and LiteIMSDK (used for sending and receiving messages and includes TIMManager and TIMConversation APIs). It takes too much time and effort to perform LVB and VJ PK by directly using LiteAVSDK and LiteIMSDK. LiveRoom makes it very easy for you to use LVB and VJ PK by calling createRoom, enterRoom, and leaveRoom.

- **Server (Backend)**
  RoomService is LiveRoom's background component. It has two responsibilities: The first is to manage rooms (add, delete, modify or query rooms) and viewers (by maintaining the number of users pushing in a room under joint broadcasting, not available under VJ PK); the second is to control Tencent Cloud LVB, Tencent-RTC and IM services (by calling Tencent Cloud backend REST APIs).



Interfacing with terminal

Step 1: Download the SDK for your platform.

Platform	Programming Language	SDK Download	API Documentation
iOS	Objective-C	DOWNLOAD	API documentation
Android	java	DOWNLOAD	API documentation

- LiteAV SDK in the decompressed SDK folder is used to implement audio/video related features.
- LiteIM SDK in the decompressed SDK folder is used to implement IM related features.
- LiveRoom in the decompressed Demo\liveroom folder provides open source code to make it easy for you to debug and customize as needed.

Step 2: Log in to RoomService (login)

LiveRoom cannot run by relying on the component on Client alone. It also needs a background service, which is called

RoomService, to manage rooms and coordinate status. To use RoomService, LiveRoom must **log in** to RoomService.

See RoomService to learn how to enter the parameters of the login function.

Step 3: Get room lists (getRoomList)

Both VJs and viewers need room lists. You can get room lists by calling **getRoomList** in LiveRoom. Each room in a list has roomInfo. When you create a room, it is recommended that you define roomInfo in JSON format for high scalability.

> Skip this step if you want to use your own room list. But you need to specify roomID in Step 4, and the roomID must be unique.

Step 4: VJs start broadcasting (createRoom)

To start broadcasting, call the API **startLocalPreview** in LiveRoom to enable local camera preview. A "view" object needs to be specified to display video images in the camera. During the period, LiveRoom requests camera permissions and VJs look at cameras to adjust beautifying and whitening effects.

Then, a room is created in the room list in the background by calling the API **createRoom** of LiveRoom, and the VJ goes into the push mode.

> **Parameter roomID**
> If you do not enter the roomId when calling createRoom, a roomID will be automatically assigned to you at backend via the callback API of createRoom. If you want to manage the room list on your own, the roomID can be assigned by your server. You only need to enter the assigned roomID when calling createRoom.

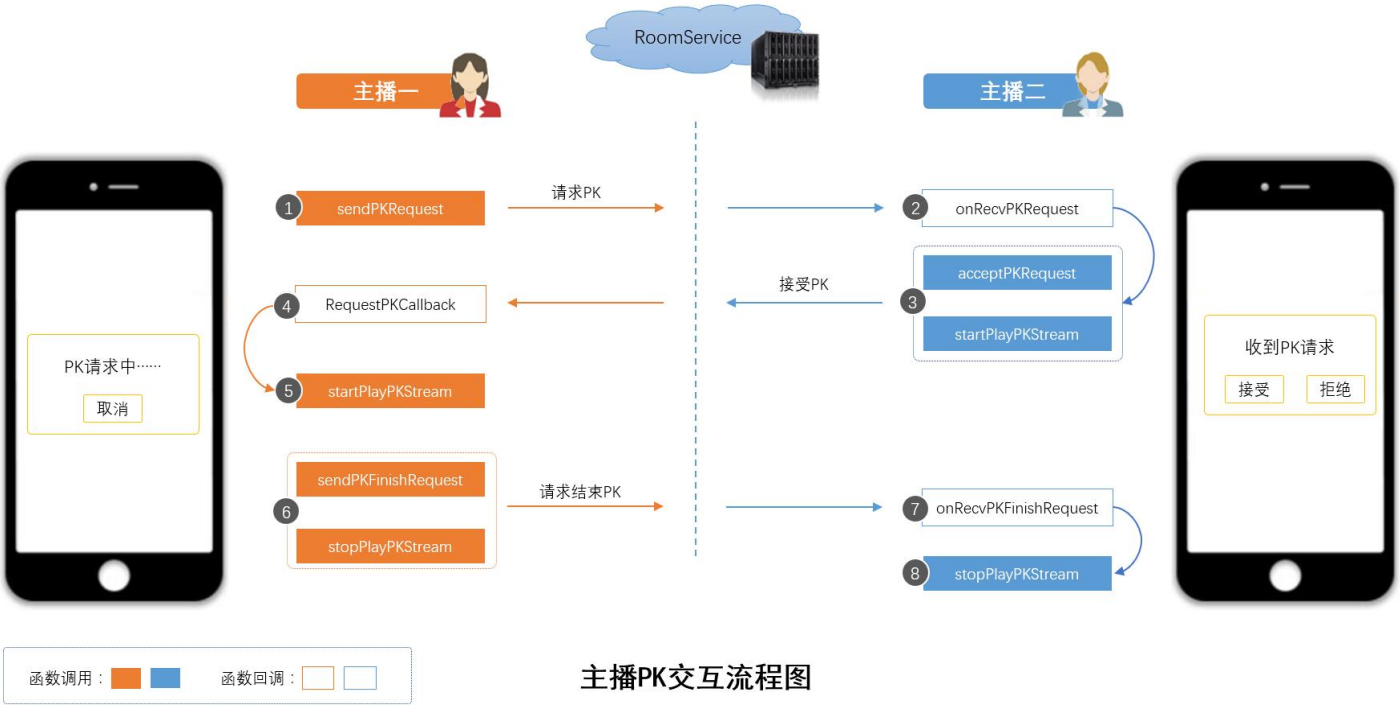
Step 5: View LVB (enterRoom)

Viewers can enter live rooms and view LVB via **enterRoom** in LiveRoom. A "view" object needs to be specified for enterRoom to display video images in the LVB stream.

Besides, after viewers enter a room, you can get the list of viewers by calling **getAudienceList** in LiveRoom. The list may not contain all the viewers. If the number of viewers is less than 30, all of them will display in the list. Otherwise, the last 30 viewers will display in the list. (For the sake of performance, at most 10 profile photos can display in the screen.)

Step 6: Perform VJ PK (sendPKRequest)

VJ PK is performed in such a way that VJs from two rooms establish real-time video chat and interact with each other by pulling each other's video streams during live broadcasting. Viewers in both of the two rooms can see the interaction process, as shown below:



主播PK交互流程图

a. Select a target VJ

Call getOnlinePusherList to get the list of LVB VJs and call back GetOnlinePusherListCallback to return VJ profile, including user name, profile photo, and user ID. Then, a list appears, in which you can select a VJ to perform PK.

b. Launch PK

- Step 1 (VJ 1): Calls **sendPKRequest** to send a PK request to VJ 2.
- Step 2 (VJ 2): Receives the callback notification from **onRecvPKRequest**. A prompt asking whether VJ 2 to accept the PK appears.
- Step 3 (VJ 2): Either calls **acceptPKRequest** to accept the PK or **rejectPKRequest** to reject it. If VJ 2 accepts it, call startPlayPKStream to play VJ 1's video stream.
- Step 4 (VJ 1): Calls **RequestPKCallback** to learn whether the PK is accepted.
- Step 5 (VJ 1): If the PK is accepted, VJ 1 calls **startPlayPKStream** to play VJ 2's video stream.

> Parameter startPlayPKStream can be used to complete pull playback and trigger stream mixing at background (adding another VJ's video stream to yours). Viewers can see the PK between two VJs without having to pull streams again.

c. Terminate PK

Either VJ can terminate the PK. Assume that VJ 1 terminates the PK:

- Step 6 (VJ 1): Calls **sendPKFinishRequest** to send a PK termination request to VJ 2, and calls **stopPlayPKStream** to stop playing VJ 2's video stream.
- Step 7 (VJ 2): Receives the callback notification from **onRecvPKFinishRequest**.
- Step 8 (VJ 2): Calls **stopPlayPKStream** to stop playing VJ 1's video stream.

Parameter stopPlayPKStream is used to stop playing the video stream and cancel stream mixing at background. Viewers are switched to the LVB mode without having to pull streams again.

Step 7: On-screen comment (sendMsg)

LiveRoom provides message delivery APIs. You can send ordinary text messages (on-screen comments) via **sendRoomTextMsg** and custom messages (such as giving a like, or sending a flower) via **sendRoomCustomMsg**.

You can receive text and custom messages sent from others in the chatroom via **onRecvRoomTextMsg** and **onRecvRoomCustomMsg** in RoomListenerCallback.

ATTENTION

You can receive at most 40 messages in one second via Tencent Cloud IM. If you refresh all of them to the screen based on the receipt frequency, LVB stuttering will occur. Mind the refresh frequency.

That is the reason why some customers gain very smooth experience in trial but suffer serious stuttering after release.

Interfacing with backend

Why need to log in to LiveRoom?

LiveRoom cannot run by relying on the component on Client alone. It also needs a background service, which is called **RoomService**, to manage rooms and coordinate status. To use RoomService, LiveRoom must **log in** to RoomService.

How do I enter parameters related to login?

The parameters should be entered based on application scenarios, as shown below. Solution 1 is intended for debugging, solution 2 for quick go-live, and solution 3 for customization.

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
serverDomain	Use Tencent Cloud RoomService `https://room.qcloud.com/weapp/live_room`	Configure in advance before using Tencent Cloud RoomService `https://room.qcloud.com/weapp/live_room`	Self-deployed RoomService `https://[yourcompany]/weapp/live_room`
sdkAppID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
accType	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
userID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined, such as 9527	User-defined, such as 9527

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
userSig	Obtained from the testing URL https://room.qcloud.com/weapp/utils/get_login_info_debug	Generated by your server. How to generate it?	Generated by your server. How to generate it?
Account owner	Tencent Cloud test account	Your account	Your account
Account limits	Available from 10:00 to 22:00	No limit. Customization is not supported.	No limit. Customization is supported.
Application scenarios	Debugging by Terminal Development team before go-live	Early stage after go-live	Growth stage

Solution 1: For testing only

This solution uses the test accounts provided by Tencent Cloud for debugging. These accounts are only available from 10:00 to 22:00 during the debugging period.

Step 1: Configure RoomService

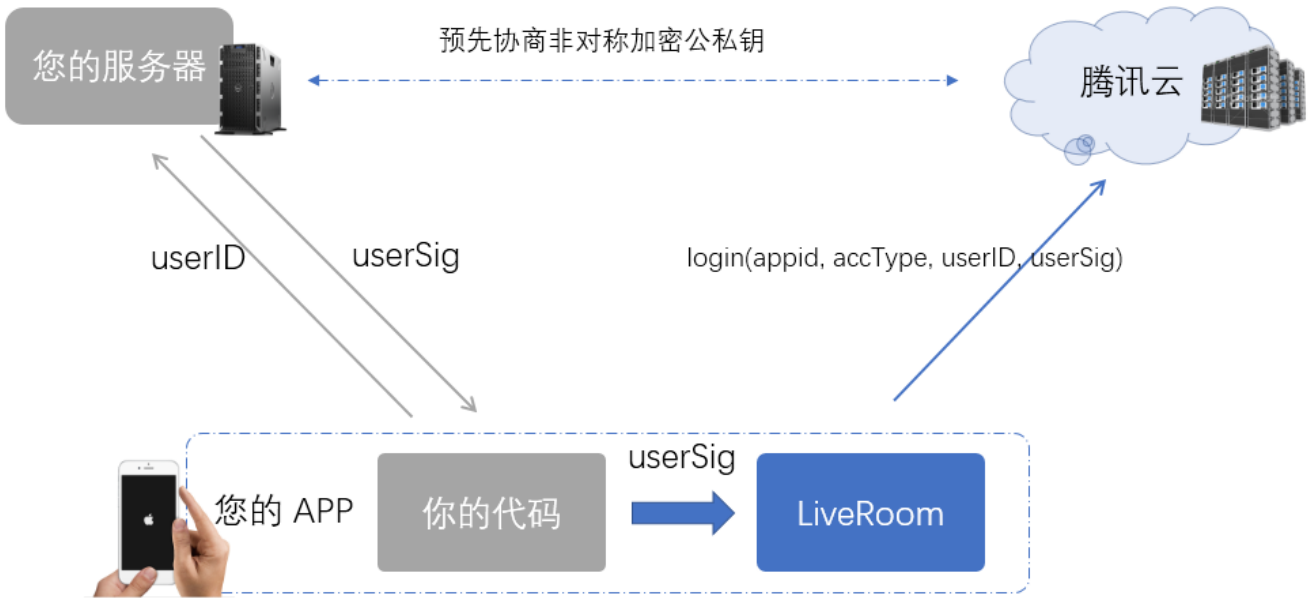
For serverDomain, enter `https://room.qcloud.com/weapp/live_room` .

Step 2: Get parameters required for login

Get relevant parameters from the testing URL (`https://room.qcloud.com/weapp/utils/get_login_info_debug`).

Solution 2: Use Tencent Cloud RoomService

This solution uses your Tencent Cloud account and Tencent Cloud RoomService. You need to configure RoomService before implementing this solution.



Step 1: Configure RoomService

The URL of Tencent Cloud RoomService is:

```
https://room.qcloud.com/weapp/live_room
```

Click RoomTool.zip to download the backend configuration tool for Tencent Cloud RoomService. This tool is based on Node.js and can only be used when Node.js has been installed. The PDF and PPT files in the configuration tool package describe in detail how to configure the RoomService. The following provides an overview of the configuration items.

Configuration Item	Description	Obtained From
LVB appID	Identifies customers in Tencent Cloud LVB	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC
IM sdkAppID	Identifies IM customers in Tencent Cloud IM service	DOC
IM accountType	Used to identify App type and now is retained for compatibility.	DOC
IM administrator	Required for IM REST API in RoomService to send room's system messages.	DOC
IM privateKey	Used by RoomService to issue the administrator's usersig to call IM REST API to send room's system messages.	DOC
IM publicKey	Used by RoomService to verify login user's identity.	DOC

Step 2: Get parameters required for login

RoomService can only be used when `login(serverDomain, sdkAppID, accType, userID, userSig)` is called successfully on terminal. The first four parameters can be hard-coded at Client, but UserSig must be issued by your backend server. If UserSig is calculated at Client, the signature keys need to be written in the terminal code, causing the risk of being hacked.

RoomService and IM use the same UserSig issued by your server, so a UserSig can be used to log in to both IM and RoomService. For more information on how to use it, please see issuing UserSig.

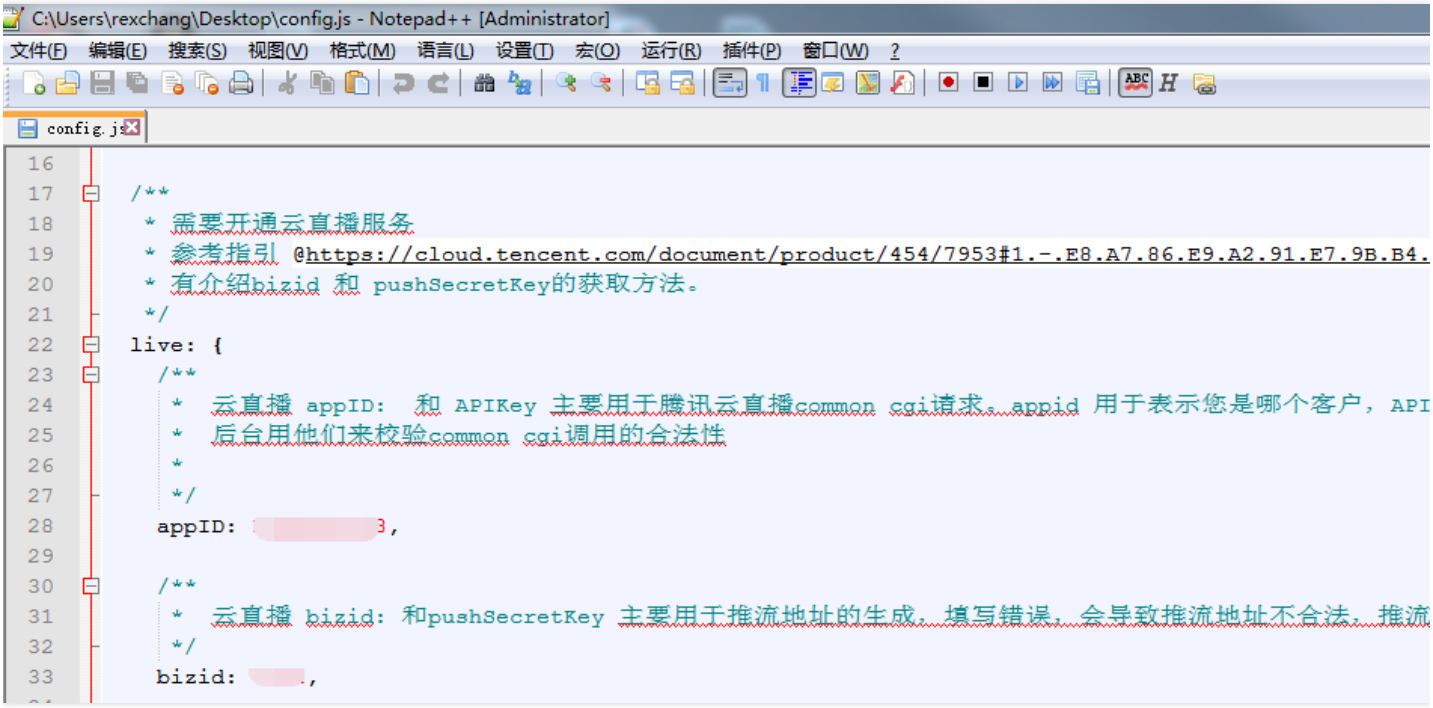
Solution 3: User-deployed RoomService

This solution uses your Tencent Cloud account and the RoomService you deploy, allowing you to modify and customize the internal logic as needed.

Step 1: Download source code, modify configuration, and deploy RoomService

Download RoomService backend source code from CODE. The source code package is composed of three directories, among which live_room contains the source code you need.

After downloading the source code, decompress the package, locate the file config.js under the folder live_room, and then modify some configuration items. The configuration items are similar to those in Solution 1, except that you can modify local source code directly without using the configuration tool for RoomService. You can configure them by referring to Step 1 in Solution 1.



Configuration Item	Description	Obtained From
LVB pushSecretKey	Required for calculating the hotlink protection signature of a push URL	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC

Then, you can deploy RoomService on your backend server and notify your terminal development engineer of the public network URL of your server, so that he/she can specify the backend URL for RoomService when calling the parameter login on terminal. For example:

```
https://[www.yourcompany.com]/weapp/live_room
```

Step 2: Get parameters required for login

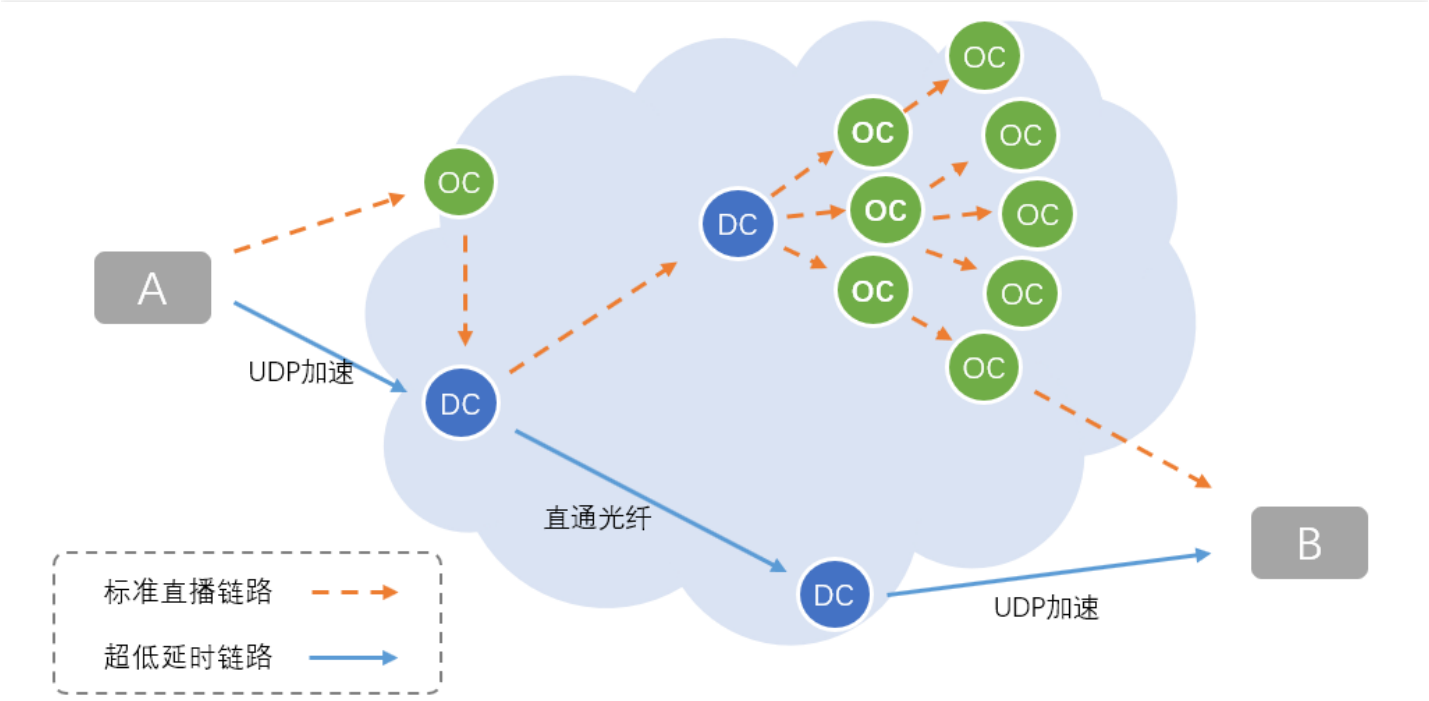
After Step 1, you have created your RoomService, which also authenticates user identity based on `sdkAppID, accType, userID and userSig` (or another authentication method you desire). Similarly, you need to assign a login signature to Client as described in Generating UserSig.

> In the backend source code of RoomService, the function getSig in `logic\im_mgr.js` generates the UserSig in the sample code in node.js. The sample code in Java and PHP will be available soon.

How It Works

1. Two "paths"

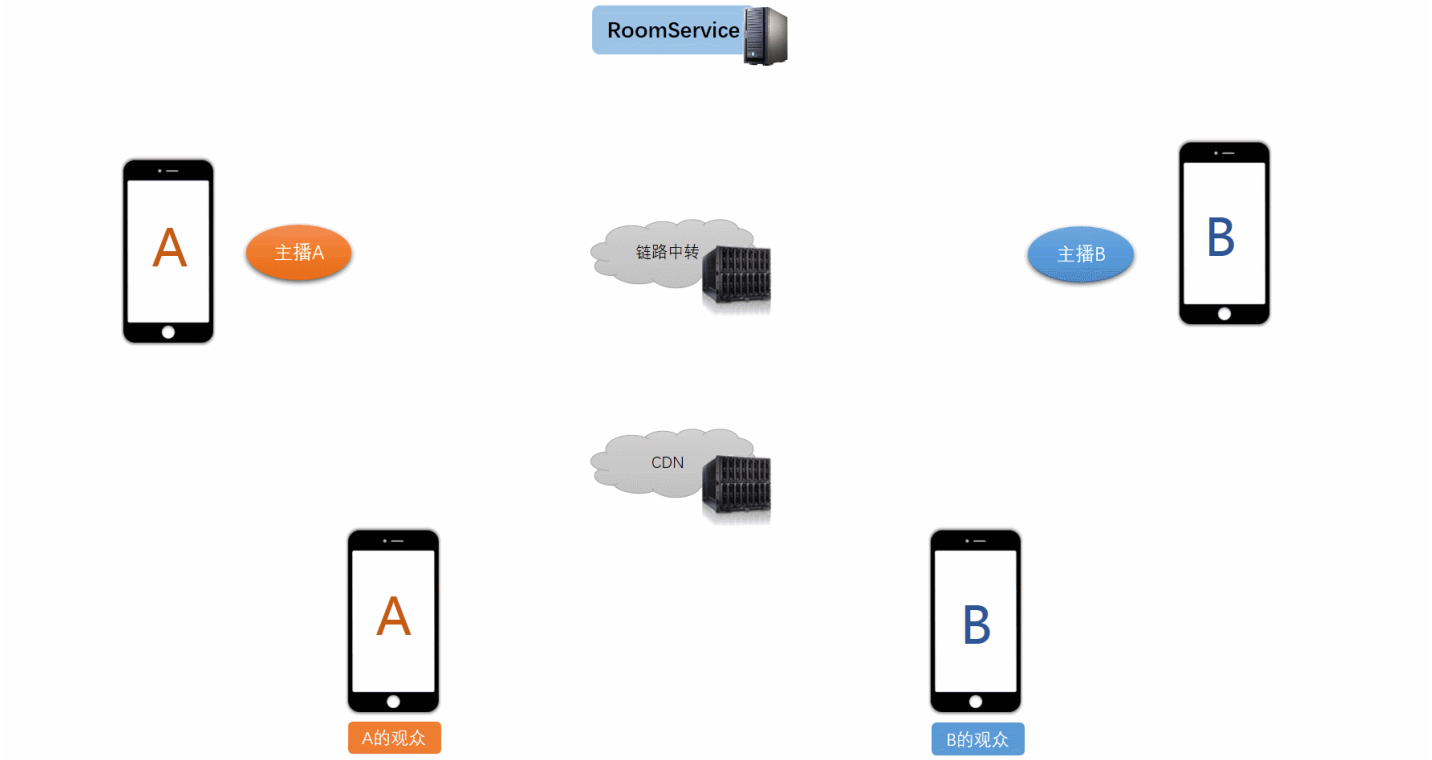
Tencent Cloud uses two paths to implement LVB and VJ PK. LVB is implemented using RTMP and FLV protocols over standard CDN, which allows as many viewers as possible to watch the live show with very low bandwidth costs. However, the latency is greater than 2 seconds. VJ PK is implemented using UDP protocol over special Direct Connect. With a latency of around 500 ms, VJ PK allows at most 10 people to enable video chat simultaneously, with the cost of each push higher than that of ordinary LVB.



Path	LVB Path	VJ PK Path
Latency	>=2s	Around 500 ms
Underlying protocol	RTMP/HTTP-FLV	Private UDP protocol
Price/Fee	low	Each push higher than LVB
Maximum number of concurrent viewers	No upper limit	<=10
TXLivePusher	SD, HD, and FHD are available for setVideoQuality	MAIN_PUBLISHER is available for setVideoQuality
TXLivePlayer	PLAY_TYPE_LIVE_FLV	PLAY_TYPE_LIVE_RTMP_ACC
Playback URL	Ordinary FLV URLs	RTMP URLs with hotlink protection signatures

2. Function principle

You can integrate LiveRoom without having to understand the function principle. If you are interested, the following figure provides a general picture on how it works.

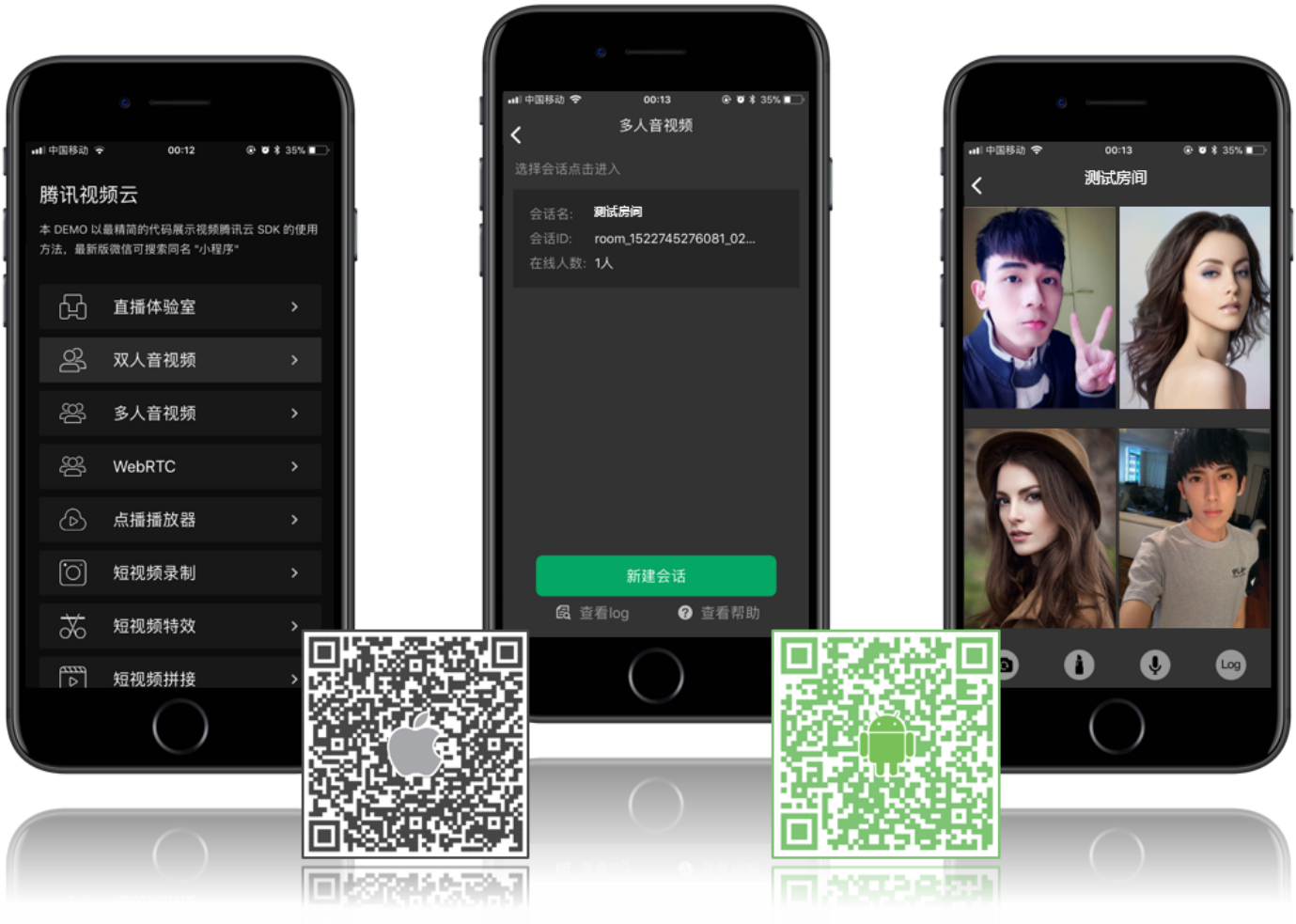


RTCRoom Module

Last updated：2018-07-10 14:50:54

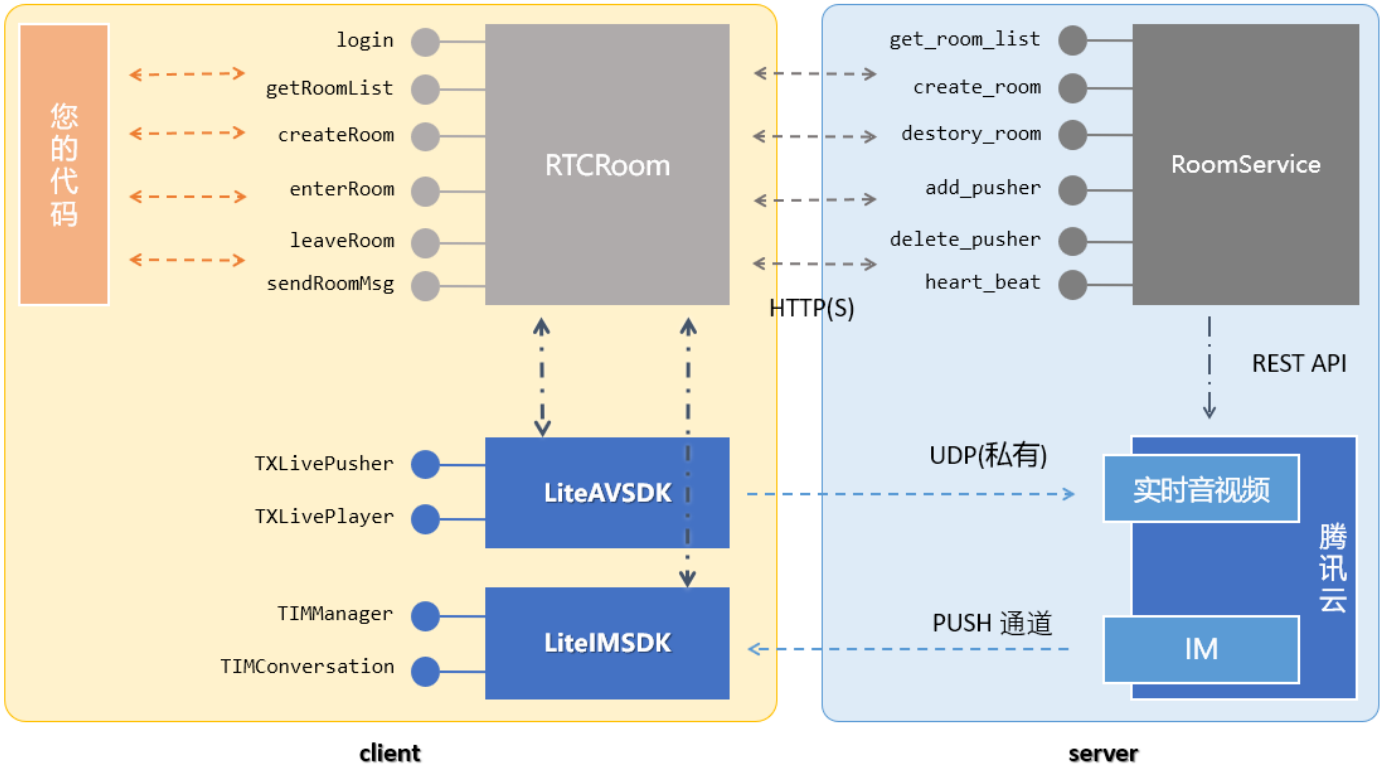
Feature Description

RTCRoom is a video chat solution that is used for real-time video chats between two or more people. It is applied in such scenarios as online customer service, remote loss assessment, remote account opening and online court hearing. This solution is implemented in several versions on various platforms including iOS, Android, mini programs and Windows, and supports interoperability between platforms.



RTCRoom

RTCRoom involves a Client and a Server.



- **Client (Terminal)**
  RTCRoom's terminal component is available at an open source basis. It encapsulates Tencent Video Cloud LiteAVSDK (used for audio/video and includes TXLivePusher and TXLivePlayer APIs) and LiteIMSDK (used for sending and receiving messages and includes TIMManager and TIMConversation APIs).

  You don't need to use LiteAVSDK and LiteIMSDK, which are underlying SDKs. You can implement the video chat feature by simply using RTCRoom's createRoom, enterRoom, leaveRoom and other APIs.

  At the same time, RTCRoom relies on a backend service for room management and status coordination, as described below.

- **Server (Backend)**
  RoomService is RTCRoom's backend component, which is also open-source. Its code logic fulfills two purposes: the first is to manage rooms (add, delete, modify or query rooms) and viewers (maintain the room members who are in the process of a video chat); the second is to control Tencent-RTC and IM services (by calling Tencent Cloud's backend REST APIs).

  You can deploy the provided source code to your business backend, or use the free RoomService service offered by Tencent Cloud.

Interfacing with terminal

Step 1: Download the SDK for your platform.

Platform	Programming Language	SDK Download	API Documentation
iOS	Objective-C	DOWNLOAD	API documentation

Platform	Programming Language	SDK Download	API Documentation
Android	java	DOWNLOAD	API documentation
WeChat Mini Program	javascript	DOWNLOAD	API documentation
IE browser	javascript	DOWNLOAD	API documentation
Windows (C++)	C++	DOWNLOAD	API documentation
Windows (C#)	C#	DOWNLOAD	API documentation

- LiteAV SDK in the decompressed SDK folder is used to implement audio/video related features.
- LiteIM SDK in the decompressed SDK folder is used to implement IM related features.
- RTCRoom in the decompressed Demo\rtcroom folder is provided at an open source basis to make it easier for debugging and customization as needed.

Step 2: Log in to RoomService (login)

RTCRoom cannot run by relying on the terminal component alone. It also needs a backend service, which is called **RoomService**, to manage rooms and coordinate status. **Login** is required for RTCRoom to use RoomService.

See RoomService to learn how to enter the parameters of the login function.

Step 3: Get room list

You can get a room list by calling the API **getRoomList** of RTCRoom. Each room in the list has its roomInfo, which is input when createRoom is called. It is recommended to define roomInfo in JSON format for a high scalability.

> Skip this step if you want to use your own room list. But you need to specify roomID in Step 4, and the roomID must be unique.

Step 4: Create a room

Enable local camera preview by calling the API **startLocalPreview** of RTCRoom. A "view" object needs to be specified to display the video images from the camera. During this period, RTCRoom requests the authorization to use camera and VJ can adjust the beauty filter and whitening effects by facing the camera.

Then, a room is created in the room list at backend by calling the API **createRoom** of RTCRoom, and the creator of the room goes into the push mode.

> **Parameter RoomID**
>
> If you do not enter the roomId when calling createRoom, a roomID will be automatically assigned to you at backend via the callback API of createRoom. If you want to manage the room list on your own, the roomID can be assigned by your server. You only need to enter the assigned roomID when calling createRoom.

Step 5: Join a room

You can join a room identified by roomID by calling the API **enterRoom** of RTCRoom. If there are already members in the room, **onGetPusherList** of RoomListenerCallback returns a list of existing room members (pusherInfoList).

Next, by calling the API **addRemoteView** and specifying the pusherInfo and the "view" object, the remote image specified by the pusherInfo can be displayed on the specified "view".

When you are in the room, RoomListenerCallback notifies you of members who join or leave the room using **onPusherJoin** and **onPusherQuit**.

Step 6: Receive/send text messages

LiveRoom comes with APIs for sending messages. You can send text messages via the API **sendRoomTextMsg**, or send custom messages (such as URLs of images) via the API **sendRoomCustomMsg**.

You can receive text and custom messages sent from others in the chatroom via **onRecvRoomTextMsg** and **onRecvRoomCustomMsg** in RoomListenerCallback.

Interfacing with backend

Why is login required for RTCRoom?

RTCRoom cannot run by relying on the terminal component alone. It also needs a backend service, which is called **RoomService**, to manage rooms and coordinate status. **Login** is required for RTCRoom to use RoomService.

How do I enter parameters related to login?

The parameters should be entered based on application scenarios, as shown below. Solution 1 is intended for debugging, solution 2 for quick go-live, and solution 3 for customization.

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
serverDomain	Tencent Cloud RoomService `https://room.qcloud.com/weapp/rtc_room`	Tencent Cloud RoomService `https://room.qcloud.com/weapp/rtc_room` (Configuration is needed in advance)	User-deployed RoomService `https://[yourcompany]/weapp/rtc_room`
sdkAppID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
accType	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined. How to obtain it?	User-defined. How to obtain it?
userID	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	User-defined, such as 9527	User-defined, such as 9527
userSig	Obtained from the testing URL `https://room.qcloud.com/weapp/utils/get_login_info_debug`	Generated by your server. How to generate it?	Generated by your server. How to generate it?
Account owner	Tencent Cloud test account	Your account	Your account
Account limits	Available from 10:00 to 22:00	No limit. Customization is not supported.	No limit. Customization is supported.

Parameter Name	Solution 1 (for Testing Only)	Solution 2 (Tencent Cloud RoomService)	Solution 3 (User-deployed RoomService)
Application scenarios	Debugging by Terminal Development team before go-live	Early stage after go-live	Growth stage

Solution 1: For testing only

This solution uses the test accounts provided by Tencent Cloud for debugging. These accounts are only available from 10:00 to 22:00 during the debugging period.

Step 1: Configure RoomService

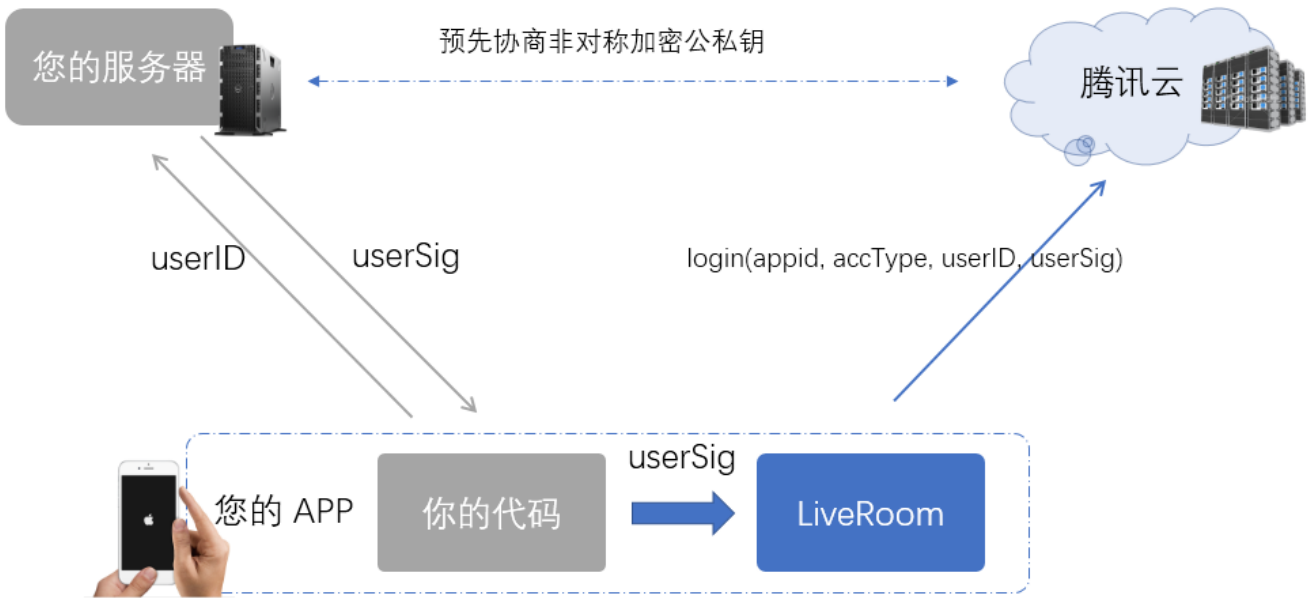
Enter `https://room.qcloud.com/weapp/rtc_room` as the serverDomain.

Step 2: Get parameters required for login

Get relevant parameters from the testing URL (`https://room.qcloud.com/weapp/utils/get_login_info_debug`).

Solution 2: Use Tencent Cloud RoomService

This solution uses your Tencent Cloud account and Tencent Cloud RoomService. You need to configure RoomService before implementing this solution.



Step 1: Configure RoomService

The URL of Tencent Cloud RoomService is:

```
https://room.qcloud.com/weapp/rtc_room
```

Click RoomTool.zip to download the backend configuration tool for Tencent Cloud RoomService. This tool is based on Node.js and can only be used when Node.js has been installed. The PDF and PPT files in the configuration tool package describe in detail how to configure the RoomService. The following provides an overview of the configuration items.

Configuration Item	Description	Obtained From
LVB appID	Identifies customers in Tencent Cloud LVB	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC
IM sdkAppID	Identifies IM customers in Tencent Cloud IM service	DOC
IM accountType	Used to identify App type and now is retained for compatibility.	DOC
IM administrator	Required for IM REST API in RoomService to send room's system messages.	DOC
IM privateKey	Used by RoomService to issue the administrator's usersig to call IM REST API to send room's system messages.	DOC
IM publicKey	Used by RoomService to verify login user's identity.	DOC

Step 2: Get parameters required for login

RoomService can only be used when `login(serverDomain, sdkAppID, accType, userID, userSig)` is called successfully on terminal. The first four parameters can be hard-coded at Client, but UserSig must be issued by your backend server. If UserSig is calculated at Client, the signature keys need to be written in the terminal code, causing the risk of being hacked.

RoomService and IM use the same UserSig issued by your server, so a UserSig can be used to log in to both IM and RoomService. For more information on how to use it, please see issuing UserSig.

Solution 3: User-deployed RoomService

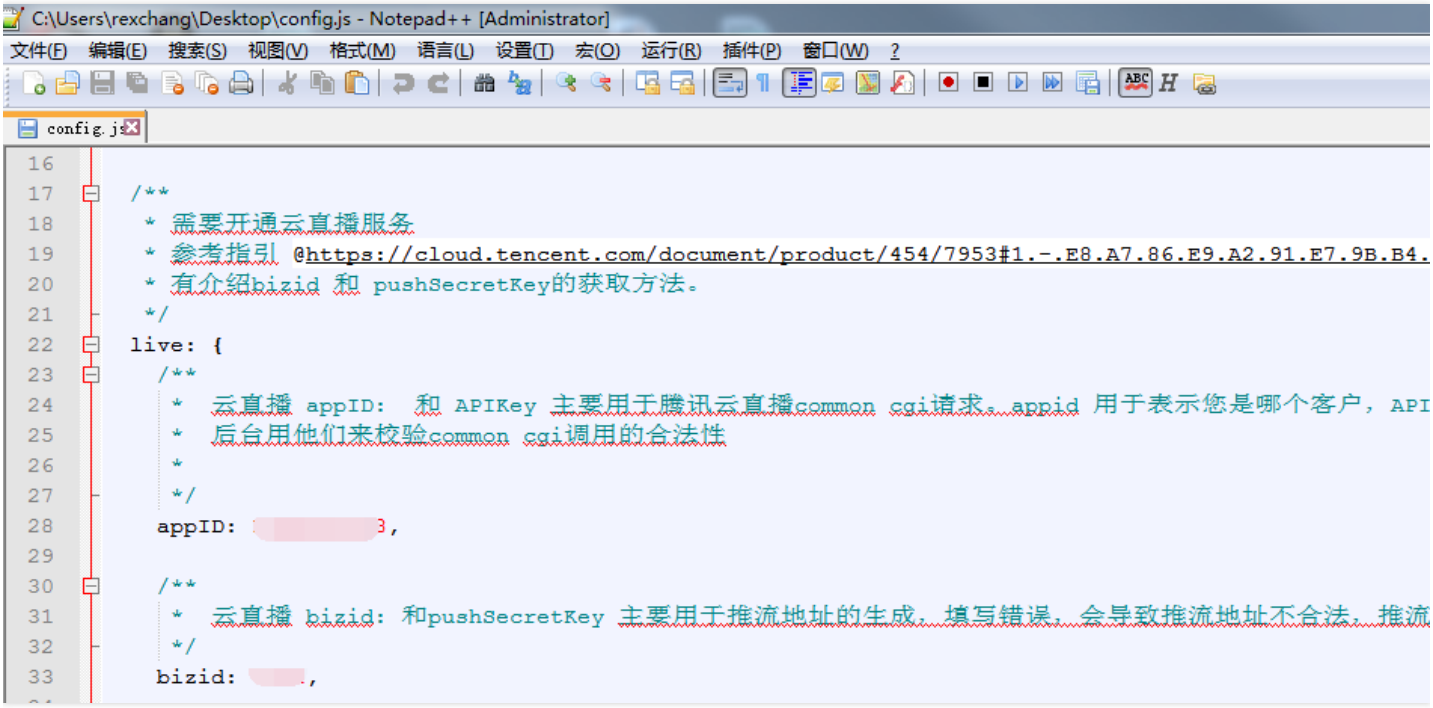
This solution uses your Tencent Cloud account and the RoomService you deploy, allowing you to modify and customize the internal logic as needed.

Step 1: Download source code, modify configuration, and deploy RoomService

Download RoomService backend source code from CODE. The source code package is composed of three directories, among which rtc_room contains the source code you need.

> Only the source code in node.js is available now. The source code in php and java is still under development. Please visit our official website regularly to learn about the latest updates.

After downloading the source code, decompress the package, locate the file config.js under the folder rtc_room, and then modify some configuration items. The configuration items are similar to those in Solution 1, except that you can modify local source code directly without using the configuration tool for RoomService. You can configure them by referring to Step 1 in Solution 1.



Configuration Item	Description	Obtained From
LVB pushSecretKey	Required for calculating the hotlink protection signature of a push URL	DOC
LVB APIKey	Used for security protection when backend REST APIs of Tencent Cloud LVB are called	DOC

Then, you can deploy RoomService on your backend server and notify your terminal development engineer of the public network URL of your server, so that he/she can specify the backend URL for RoomService when calling the parameter login on terminal. For example:

```
https://[www.yourcompany.com]/weapp/rtc_room
```

Step 2: Get parameters required for login

After Step 1, you have created your RoomService, which also authenticates user identity based on `sdkAppID, accType, userID and userSig` (or another authentication method you desire). Similarly, you need to assign a login signature to Client as described in Generating UserSig.

> In the backend source code of RoomService, the function getSig in `logic\im_mgr.js` generates the UserSig in the sample code in node.js. The sample code in Java and PHP will be available soon.

Live Streaming Quiz (Prize Pool Mode)

Last updated：2019-03-01 16:50:27

Effect Experience

Demo installation

Download and install our Demo to experience how online quizzes are presented. Two options are available to experience it, both of which allow for perfect "sound-image-question" synchronization.

iOS platform (ipa)	Android platform (apk)	OBS Studio customized version (exe)
iOS	Android	Obs Studio

Option:

Create a room to view what it looks like at the host end and enter the room to view what it looks like at the viewer end.

- Demos for the iPhone platform are provided in an enterprise-signed manner. Go to **Settings** -> **General** -> **Device Management** to add a trust certificate first.
- Demos are only used to illustrate the technical expertise of Tencent Cloud. They do not represent how the final product looks like. Read this document carefully before you integrate any demo.

ios

android

Demo入口

主持人（演示出题）

观众（演示答题）

SDK Downloading

- **LiteAVSDK (4.1.3173)**
  is used for RTMP push and FLV playback. The Smart version provides both of the two functions, and the LivePlay version only provides FLV playback.

Operating System	Download Link	RTMP Push	RTMP Playback	FLV Playback	Notes
iOS	DOWNLOAD	YES	YES	YES	Both SDK and Demo source codes are in a zip file
	DOWNLOAD	NO	YES	YES	Size increment: 700 KB, both SDK and Demo source codes are in a zip file
Android	DOWNLOAD	YES	YES	YES	Both SDK and Demo source codes are in a zip file
	DOWNLOAD	NO	YES	YES	jar increment: 350 KB, both SDK and Demo source codes are in a zip file

- **LiteIMSDK (1.3.0.130)**
  is used for functions such as chat room and on-screen comment. The download link here points to a simplified version. You may download a full version from the Tencent Instant Messaging (IM) website.

Operating System	Download Link	Notes
iOS	DOWNLOAD	This is a simplified version, with a size increment of 1.74 MB.
Android	DOWNLOAD	This is a simplified version, with a size increment of 670 KB.

Our Advantages

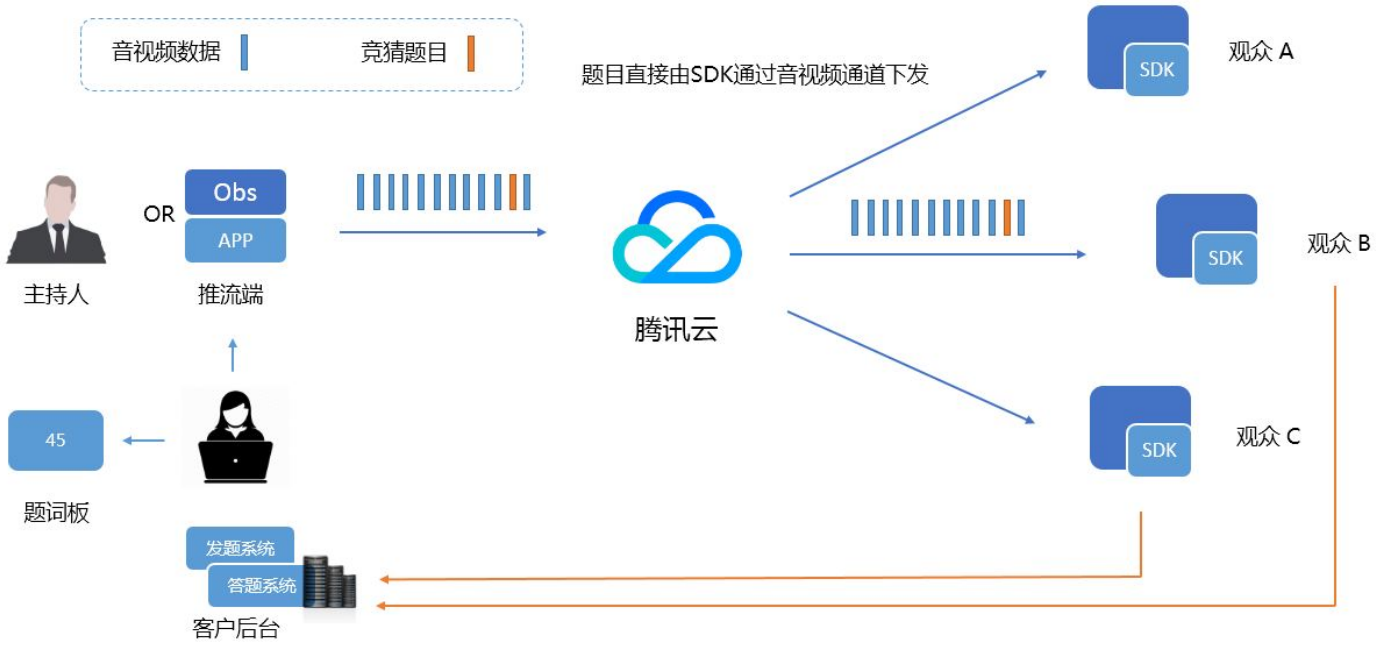
- **Precise "sound-image-question" synchronization**
  Both Tencent Cloud SDK and the cloud support inserting **questions** or **time synchronization signaling** into LVB streams to achieve perfect synchronization of sounds, images and question pop-ups.

- **Ultra-low latency deviation at the viewer end**
  The latency correction technology supported by the **speedy playback mode** in Tencent Cloud SDK can keep the latency deviation between viewers within 1 sec, thus ensuring that viewers answer questions synchronously.

- **Integration with WeChat Mini Programs**
  Tencent Cloud SDK is packed in WeChat by default and made publicly available as a <live-player> tag. Set the mode to live, and also set min-cache and max-cache to 1 to enable ultra-low latency in playback.

Method Details

Method 1: Transparent question transmission



How it works

- **Message delivery (OBS)**:
  If you are using OBS to push streams in studios, you can replace the current OBS software directly with Obs Studio, which was modified by Tencent Cloud. We have added a **Question Assignment** button to the **Tools** menu bar, so that you can insert questions into LVB streams. You can edit questions in an ini file in advance.

- **Message delivery (App)**:
  To simply use your App to push streams, you can use the sendMessage method of the TXLivePusher in Tencent Cloud SDK. This method allows inserting a buffer, the length of which must be limited to 10 KB, to RTMP streams.

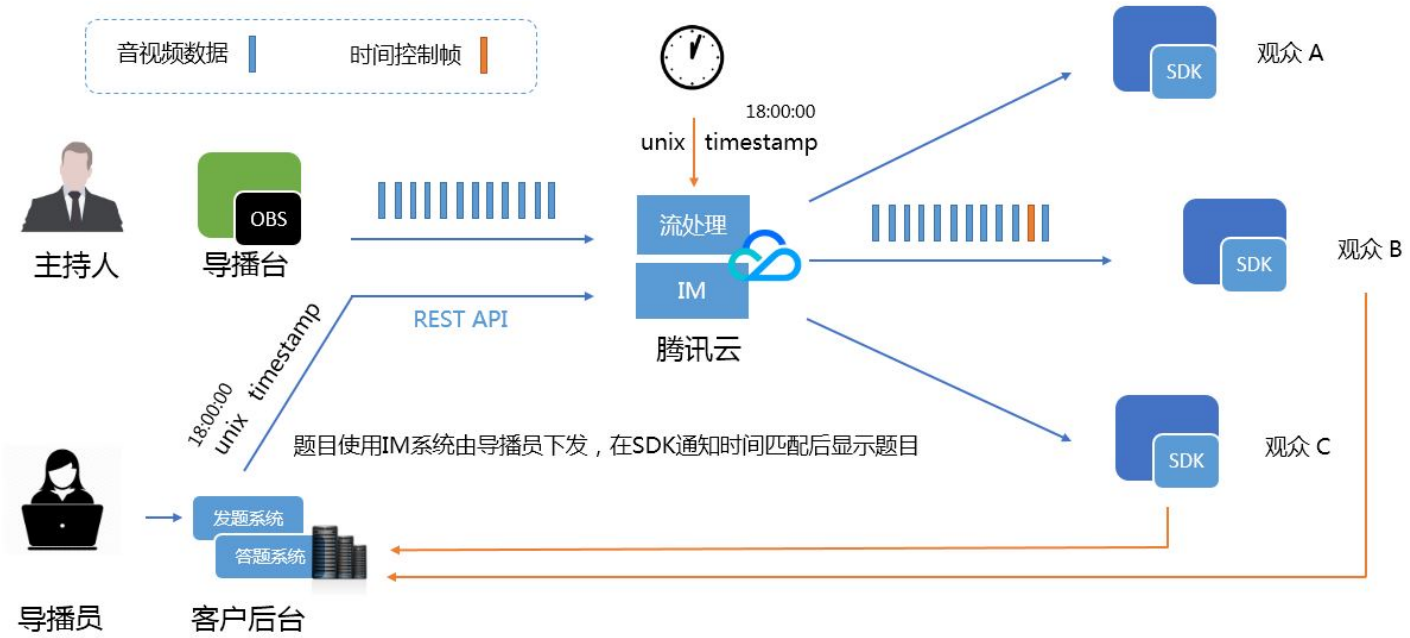
```
//iOS sample code
[_answerPusher sendMessage:[mesg dataUsingEncoding:NSUTF8StringEncoding]];

//Android sample code
mTXLivePusher.sendMessage(questionInfo.getBytes("UTF-8"));
```

- **Message receiving**:
  By using the onPlayEvent (PLAY_EVT_GET_MESSAGE: 2012) function of the TXLivePlayer in Tencent Cloud SDK, you can have the message received by your App and the questions spread to the massive viewer-end Apps at exactly the time when a specific image is played back.

For more information about integration solutions for message receiving, please see our integration document (iOS Platform | Android Platform)).

Method 2: NTP Time Synchronization



How it works

1. Tencent Cloud inserts in real time an international standard timestamp calibrated by NTP into LVB streams every other second.
2. The program director in the studio assigns questions at the appropriate time based on the pace at which the host raises questions. The assignment system inserts the current international standard time into the questions assigned each time.
3. When playing back video streams inserted with such timestamps, SDK notifies your App of the time when the currently played back image was recorded. Considering a fixed latency from the studio to the cloud, make sure you correct the deviation in advance.
4. Your App can display specified questions as needed according to the time notifications saying when the current image was recorded from the SDK.

In summary, the biggest difference between Method 1 and Method 2 is how questions are spread. In Method 2, the questions are quickly sent to the viewer-end App through the IM channel and cached at the viewer end. Then the questions are displayed after the player is notified of the expected NTP timestamp.

Method 3: Mini Program Solutions

While both of the above two methods provide perfect "sound-image-question" synchronization, what matters more than this minor optimization on experience is the spreading capability of an App. Mini Program happens to provide an App with the capability of viral spread.

Tencent Cloud SDK is packed in WeChat by default and made publicly available as a <live-player> tag.

- If an flv playback URL is used,
- set the mode to live,
- min-cache and max-cache to 1,
- and push-end GOP to 1.

This minimizes the playback latency and keeps the latency deviation between viewers within 1 sec. Although exact "sound-image-question" synchronization is unrealizable, the effects can be as good as they are in an App. (Online quizzes were not popularized when the video cloud SDK was packed in WeChat, so inserting messages into audio/video streams is not supported.)

The rest step is to assign questions to the mini program through the websocket channel of the mini program or by using the webim solution.

Integration Guide (Method 1)

Step 1: Activate Tencent Cloud LVB service

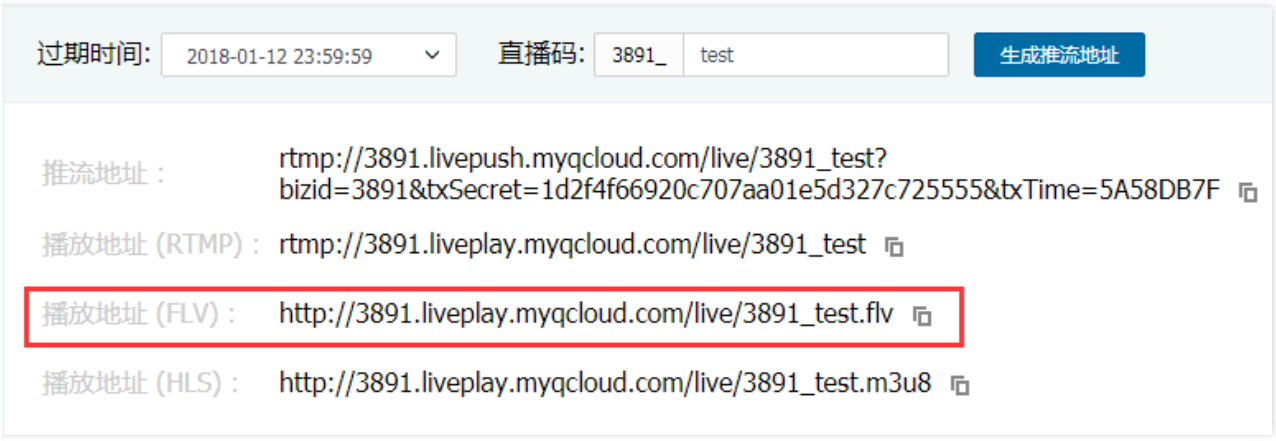
Contact us for activating Tencent Cloud LVB service. You can call our 400 customer service number to rush the approval process if you need it urgently.

Step 2: Obtain the push URL

To simply obtain a push URL, please see the document How to Get URL Quickly..
To understand the relationship between push URL and Live room ID, please see the document How to realize auto construction at the backend.
To protect the push URL from being stolen, please see the document Hotlink protection signature.

Step 3: Obtain the playback URL

There is a one-to-one mapping between the playback URL and the push URL. Please see the following figure for the mapping rule.



Be sure to use the playback URL in **FLV** format, because RTMP tends to stutter in high-concurrency scenarios.

Step 4: Configure the push end

If you are using your App to push streams, please see the document(iOS | Android)..

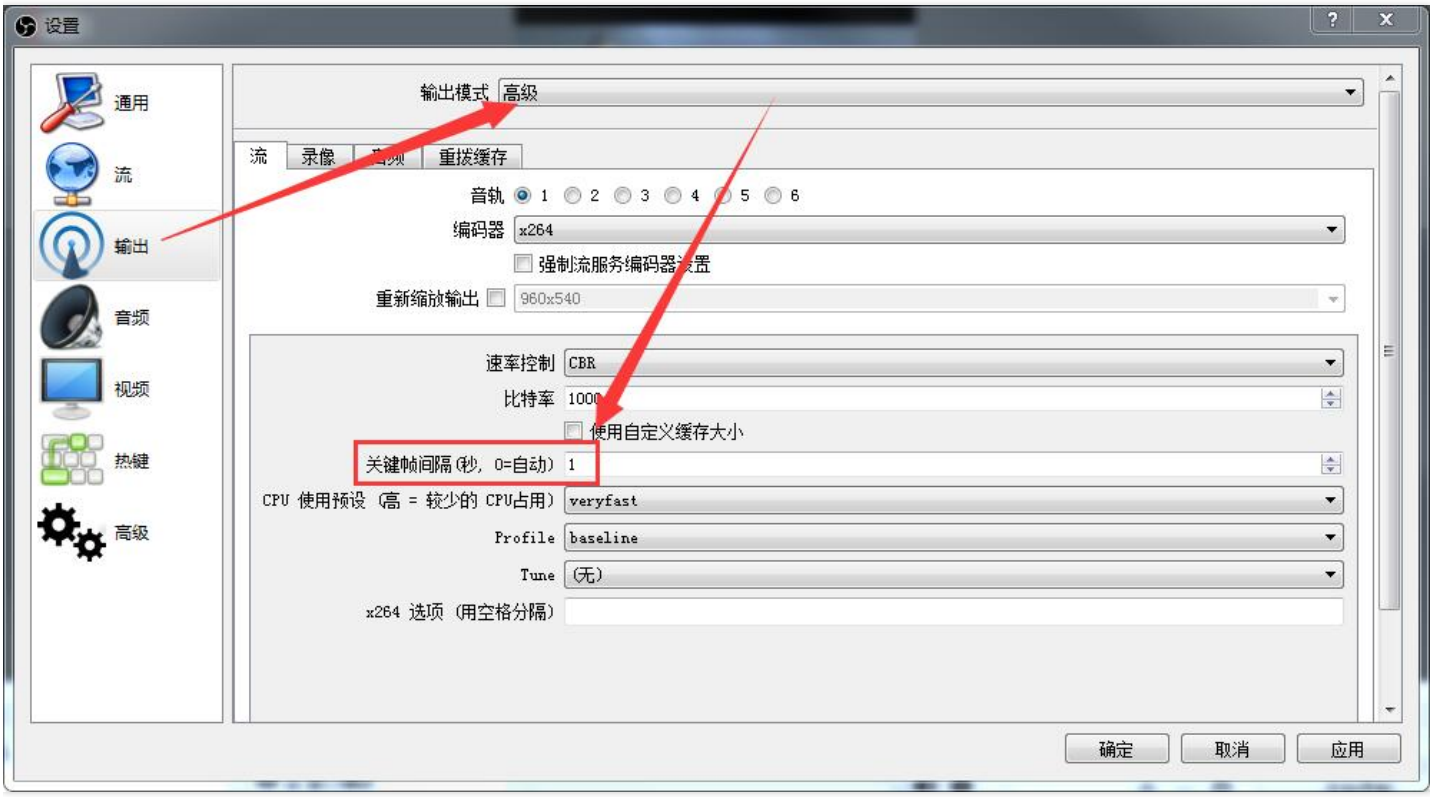
If you are using OBS to push streams, note the following important settings:

I-frame interval (GOP)

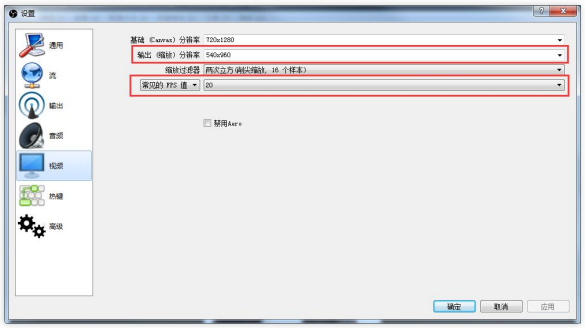
Generally, two methods are available for integration studios: OBS Studio push and encoder push, both having mature configuration APIs. It is recommended to set GOP (also known as "keyframe interval") to 1 sec, which reduces the viewer-end latency deviation to a very low level.

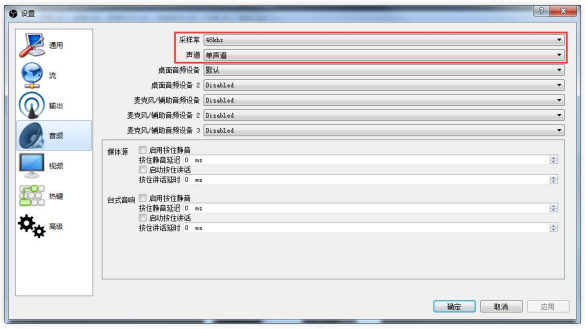
Setting GOP to x264 has little effect on the encoding efficiency, but causes much latency. The higher the GOP is, the more cache there is on the server. Excessively high GOP has huge impact on viewers who just entered, because it takes time for SDK to correct latency.

The following figure shows the keyframe interval settings in OBS Studio:



Encoding parameters

Recommended Configuration	Resolution	Video Bitrate	Frame Rate	Number of Channels	Sampling Rate	Audio Bitrate
Image Quality Priority	540x960	1000 Kbps	25	1	48k	72 Kbps
Cost Priority	360x640	600 Kbps	20	1	48k	72 Kbps
						

Recommended Configuration	Resolution	Video Bitrate	Frame Rate	Number of Channels	Sampling Rate	Audio Bitrate
						

Step 5: Integrate the player

1. Download the SDK version listed in the second section of the document.

2. See the integration document (iOS | Android)) to integrate the player. It takes about half a day to finish the work in the two platforms.

3. **Change the default settings**
   Normal LVB scenarios are set by default in the SDK, so it is necessary to change the settings as follows:

```
//iOS Source Code
TXLivePlayConfig *config = [[TXLivePlayConfig alloc] init];
TXLivePlayer *player = [[TXLivePlayer alloc] init];
//
//Enable message receiving. Failing to receive messages means this function has not been enabled (default: disabled).
config.enableMessage = YES;
//
//Set the break-event point for latency to 2 seconds. (Given the latency due to the transmission from the cloud and the p
ush end, the actual latency is more than 3 seconds: 3 seconds for SDK push latency and 4-5 seconds for obs push latency.)
config.bAutoAdjustCacheTime = YES;
config.maxAutoAdjustCacheTime = 2;
config.minAutoAdjustCacheTime = 2;
config.cacheTime = 2;
config.connectRetryCount = 3;
config.connectRetryInterval = 3;
config.enableAEC = NO;
//First setConfig and then startPlay
[player setConfig:config];
```

```
//Android Source Code
mTXLivePlayConfig = new TXLivePlayConfig();
mTXLivePlayer = new TXLivePlayer(context);
//
//Enable message receiving. Failing to receive messages means this function has not been enabled (default: disabled).
mTXLivePlayConfig.setEnableMessage(true);
//
```

```
//Set the break-event point for latency to 2 seconds. (Given the latency due to the transmission from the cloud and the p
ush end, the actual latency is more than 3 seconds: 3 seconds for SDK push latency and 4-5 seconds for OBS push latenc
y.)
mTXLivePlayConfig.setAutoAdjustCacheTime(true);
mTXLivePlayConfig.setCacheTime(2.0f);
mTXLivePlayConfig.setMaxAutoAdjustCacheTime(2.0f);
mTXLivePlayConfig.setMinAutoAdjustCacheTime(2.0f);
//
//First setConfig and then startPlay
mTXLivePlayer.setConfig(mTXLivePlayConfig);
```

4. Be sure to use the playback URL in **FLV** format, because RTMP tends to stutter in high-concurrency scenarios.

Step 6: Questions spreading

- If you are using your App to assign questions, you can use the sendMessage calling method in TXLivePusher. Please see the document (iOS | Android). for details.

- If you are using the customized Obs Studio to assign questions, you can first edit the questions in a local ini file and then a program director will spread the questions at an appropriate time.

Reliability evaluation

Some customers may worry if the unstable audio/video channels will cause any stutter or video data loss that makes the viewer unable to see the questions.

- Frame loss with LVB audio/video data occurs on a per-GOP basis. If GOP is 1, then 1 second of audio/video data will be lost each time.
- According to the node deployment by Tencent Cloud, 90% of the video stutter is caused by a slow network connection at the viewer end. In this case, the other network connections won't be smooth, either.

The solution to this problem is to send a question message every second (if GOP is set to 1 second) and eliminate the repeated question numbers at the viewer end. This prevents audio/video stutter from affecting the reliability of question arrival.

Step 7: Receiving question messages

In our push APP Demo and customized OBS Studio, questions are organized into a buffer in json format and inserted into audio/video streams before being sent out.

Once you receive this buffer, you can parse it and complete the UI display. If you need to adjust json format to support more customizations, modify the source code or contact us for help.

- Switch the **enableMessage** toggle button in TXLivePlayConfig to **YES**.
- TXLivePlayer listens into messages by **TXLivePlayListener**, message No.: **PLAY_EVT_GET_MESSAGE (2012)**.

```
//iOS code
-(void) onPlayEvent:(int)EvtID withParam:(NSDictionary *)param {
[self asyncRun:^{
if (EvtID == PLAY_EVT_GET_MESSAGE) {
dispatch_async(dispatch_get_main_queue(), ^{ //Throw to the main thread to avoid thread security issues
if ([_delegate respondsToSelector:@selector(onPlayerMessage:)]) {
[_delegate onPlayerMessage:param[@"EVT_GET_MSG"]];
```

```
}
});
}
}];
}
```

```
//Android sample code
mTXLivePlayer.setPlayListener(new ITXLivePlayListener() {
@Override
public void onPlayEvent(int event, Bundle param) {
if (event == TXLiveConstants.PLAY_ERR_NET_DISCONNECT) {
roomListenerCallback.onDebugLog("[AnswerRoom] Pull failed: network disconnected");
roomListenerCallback.onError(-1, "Network disconnected, pull failed");
}
else if (event == TXLiveConstants.PLAY_EVT_GET_MESSAGE) {
String msg = null;
try {
msg = new String(param.getByteArray(TXLiveConstants.EVT_GET_MSG), "UTF-8");
roomListenerCallback.onRecvAnswerMsg(msg);
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
}
}
}
});
```

Step 8: Developing a quiz system

Due to Tencent Cloud's position as a PaaS service provider, quiz and payment systems that are closely associated with your business are not included in our offering package, so you need to develop them by yourself.

A common solution is to collect customers' answers as HTTP(S) requests to the quiz server. You just need to get prepared to deal with surges in highly concurrent requests.

Some customers may ask whether the IM system can be used to do the quizzes. The answer for now is no, because the IM system is good at spreading messages, while the aim of doing quizzes is to collect information.

Step 9: Displaying quiz result

The quiz will be closed after the questions are displayed for a while. Then the quiz system will gather the results and deliver them to the viewers.

If you are using the customized Obs Studio to spread results, prepare a simple server that provides an http API for communication with OBS about questions, answers and the number of people in the agreed json format. In this way, questions and answers are delivered.

Integration Guide (Method 2)

Step 1: Activate Tencent Cloud LVB service

Repeat the corresponding step in Method 1.

Step 2: Obtain the push URL and add NTP timestamp

Follow the corresponding step in Method 1, except that an extra parameter is required to obtain the push URL:

Add NTP timestamp

Add the parameter &txAddTimestamp=2 after obtaining the push URL. (txAddTimestamp was previously set to 1, which caused black screen in mini programs.) The server inserts an international standard SEI timestamp into LVB streams every other second (with a deviation less than 100 ms). If you use our player to playback the video streams, you will receive a message that notifies NTP time of the current image every other second.

Step 3: Obtain the playback URL

Repeat the corresponding step in Method 1.

Step 4: Configure the push end

Repeat the corresponding step in Method 1.

Step 5: Integrate the player

Follow the corresponding step in Method 1, except that the message obtained is not in json format, but a 8-byte (64-bit) timestamp.

```
long timeStamp = byteArrayToInt(param.getByteArray(TXLiveConstants.EVT_GET_MSG));
/**
* Convert a 8-byte array into a long value
*/
public static long byteArrayToInt(byte[] byteArray) {
byte[] a = new byte[8];
int i = a.length - 1, j = byteArray.length - 1;
for (; i >= 0; i--, j--) {// Copy data from the end of b (the lowest bit of an int)
if (j >= 0)
a[i] = byteArray[j];
else
a[i] = 0;// If b.length is less than 4, pad the highest bit with zeros
}
// Note that this is different from converting a byte array into an int in that the conversion requires converting the elements of the array into a long before doing the shift.
// Doing the shift directly will not produce correct results, because Java takes numbers as int by default if not specified other wise.
long v0 = (long) (a[0] & 0xff) << 56;// &0xff converts a byte to an int without difference, preventing Java from keeping the s ign bit of the high bits after the automatic type promotion.
long v1 = (long) (a[1] & 0xff) << 48;
long v2 = (long) (a[2] & 0xff) << 40;
long v3 = (long) (a[3] & 0xff) << 32;
long v4 = (long) (a[4] & 0xff) << 24;
long v5 = (long) (a[5] & 0xff) << 16;
long v6 = (long) (a[6] & 0xff) << 8;
long v7 = (long) (a[7] & 0xff);
return v0 + v1 + v2 + v3 + v4 + v5 + v6 + v7;
}
```

Step 6: Questions spreading

If you are using your own IM system to assign questions, ignore this section. If you are using Tencent Cloud IM service to do this, follow these steps:

- **1. Activate IM service**

  Activate Tencent Cloud Instant Messaging service.

- **2. Configure IM service**

  Complete the initial configuration following the document. Be sure to select **standalone mode** for integration mode.

- **3. Use REST API to create a BChatRoom for question assignment**

  REST The API in Tencent Cloud IM is specially designed for integration at the server end. The action of creating a group is usually triggered by your server, so the use of REST APIs is an appropriate method to perform the integration.

  BChatRoom is very ideal for assigning questions, because it is originally intended for system notifications, which ensures high message arrival rate and reliability.

  Use v4/group_open_http_svc/create_group to create a group. For the test method, please see **How to use REST APIs at the IM background.pdf - Step 3** in the SDK package.

- **4. Use REST API to create an AVChatRoom for sending on-screen comments**

  AVChatRoom is very suitable for sending on-screen comments in a chat room. Featuring strict dirty words filtering and frequency limitation logics, AVChatRoom is specially designed to optimize large chat room scenarios.

  Use v4/group_open_http_svc/create_group to create a group. For the test method, please see **How to use REST APIs at the IM background.pdf - Step 4** in the SDK package.

  The frequency for AVChatroom is limited to 40 comments/sec by default. Contact us if you want to adjust this limitation, because the bandwidth fee will be higher for more comments per sec.

- **5. Use REST API to send question broadcasts in BChatRoom**

  Use v4/group_open_http_svc/send_group_msg to send messages. For the test method, please see **How to use REST APIs at the IM background.pdf - Step 5** in the SDK package.

Step 7: Message receiving and on-screen comment sending and receiving

The client uses IM SDK to receive messages or send and receive on-screen comments. The integration steps are as follows.

- **1. Integrate simplified IMSDK**
  Simplified IMSDK is in SDK. Please see **"Integration Guide - IMSDK.pdf"** in the zip package for more information.

- **2. Perform the integration with the source code**
  The SDK package contains a source code file named **AnswerPlayIMCenter** that encapsulates simple calls to IMSDK (equivalent to sample code the integration document). The following describes the member functions in this class:

Member functions	Role
initIMCenter	For initialization. Your IM service information with Tencent Cloud is required.
loginIMUser	For login. Try comparing IMSKD to a UI-less version of QQ. Just as logging in is required for receiving and sending messages with QQ, you can log in to IMSKD by simply changing the login credentials from your QQ username and login password to your Userid and UserSig issued by your server.
joinIMGroup	For joining BChatRoom and AVChatRoom created by your backend server using a REST API in Step 6.
sendChatMessage	For sending on-screen comments
onRecvChatMessage	For receiving on-screen comments from AVChatRoom. Make sure you set a limit on the rendering frequency. Do not refresh the screen every time an on-screen comment is received, which would otherwise pose a huge challenge to the phone performance. This issue has been ignored by many customers because it is hard to detect owing to the relatively small number of messages during the test.
onRecvIssueMessage	For receiving question messages from BChatRoom. As described in Method 2, every question should be displayed with an international NTP time.

- **3. Timing of question display**
  Do not display a question received by onRecvIssueMessage until the player finishes the GET_MESSAGE callback. If the callback time is not sooner than the NTP time of the question, the question can be displayed.

- **4. How do I get a UserSig?**
  A UserSig is an important information for logging into IMUser, which is like a login password in QQ. UserSigs are issued by your server using RSA asymmetric encryption, so they provide a high level of security.

  For the ways to issue a UserSig, please see TLS Backend APIs. For a quick debugging process, you can use a small tool in Windows, which allows you to start terminal logic debugging before the backend developers get involved.

  > Note: The public and private keys in the small tool are for test purposes only.

Step 8: Developing a quiz system

Due to Tencent Cloud's position as a PaaS service provider, quiz and payment systems that are closely associated with your business are not included in our offering package, so you need to develop them by yourself.

A common solution is to collect customers' answers as HTTP(S) requests to the quiz server. You just need to get prepared to deal with surges in highly concurrent requests.

Some customers may ask whether the IM system can be used to do the quizzes. The answer for now is no, because the IM system is good at spreading messages, while the aim of doing quizzes is to collect information.

Step 9: Displaying quiz result

The quiz will be closed after the questions are displayed for a while. Then the quiz system will gather the results and deliver them to the viewers. To deliver the results, you can continue to use the question spread channel in **Step 6**.