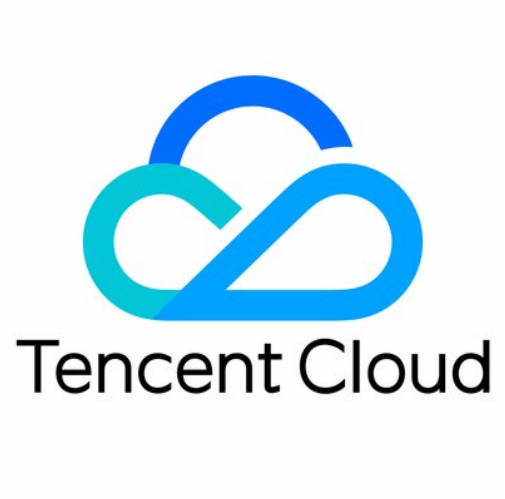
Mobile Live Video Broadcasting

Host Competition

Product Documentation



Host Competition

Last updated：2022-06-14 12:41:46

Overview

In **RTMP-based mic connect**, the MLVB SDK of Tencent Video Cloud Toolkit offers the `MLVBLiveRoom` component to help you quickly implement the host competition feature. To better cater to your needs, Tencent Cloud has launched an RTC-based host competition scheme and offered simpler and more flexible V2 APIs.

MLVB's V2 APIs support publishing/host competition via RTMP as well as RTC. You can choose whichever scheme fits your needs. Below is a comparison of the two schemes.

Item	RTMP	WebRTC
Protocol	Based on TCP	Based on UDP (more suitable for streaming)
QoS	Poor adaptability to bad network connection	Video streaming unaffected with 50% packets loss; audio mic connect unaffected with 70% packets loss
Region	Chinese mainland	Worldwide
Tencent Cloud products used	MLVB, CSS	MLVB, CSS, TRTC
Price	0.0028 USD/min	Tiered pricing. For details, see Purchase Guide.

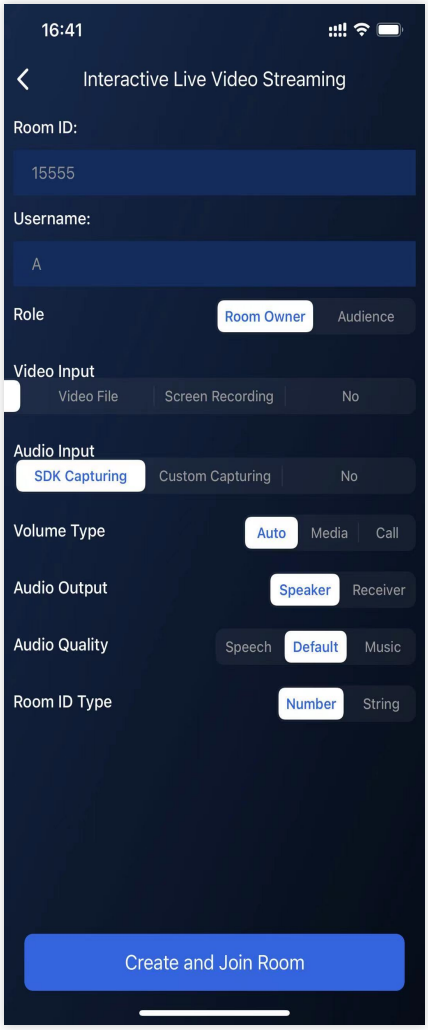
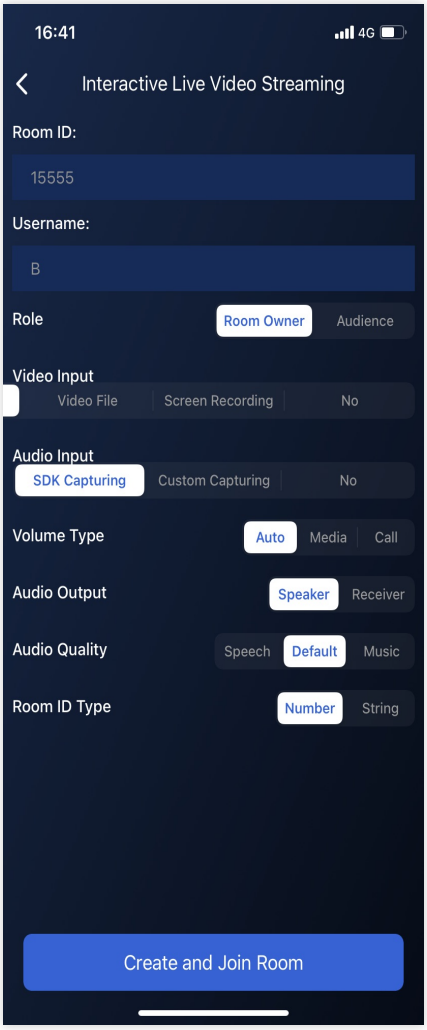
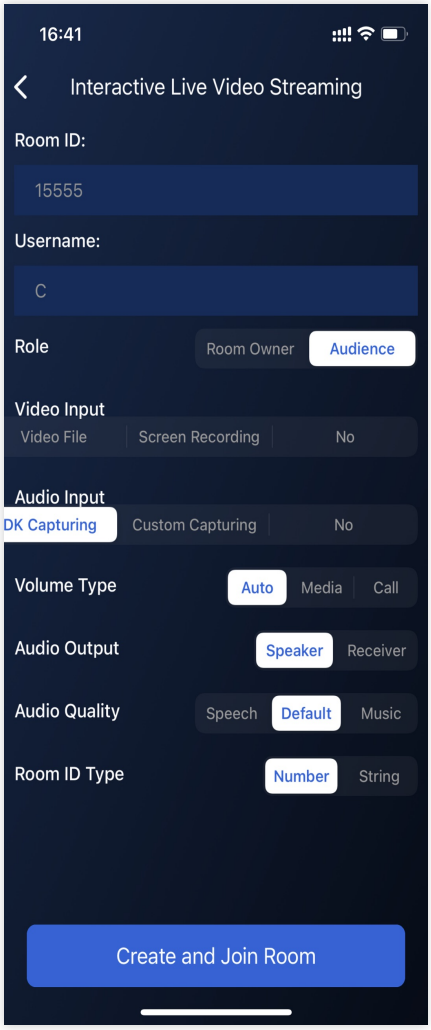
Demonstration

The MLVB SDK provides new V2 APIs via `V2TXLivePusher` (publishing) and `V2TXLivePlayer` (playback) to power larger-scale live streaming scenarios with greater flexibility and lower latency. Hosts can use the capabilities provided by the APIs for RTC-based publishing. Audience, by default, play streams via CDNs, whose cost is relatively low. To compete, hosts only need to play each other's stream. To enable RTC-based host competition, you must activate TRTC.
Below are the UI views of the MLVB-API-Example demo.

UI demonstration

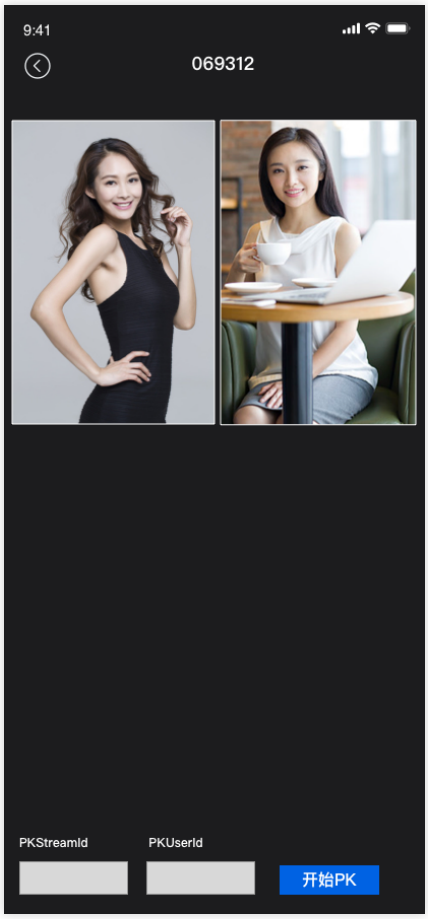
Before streaming

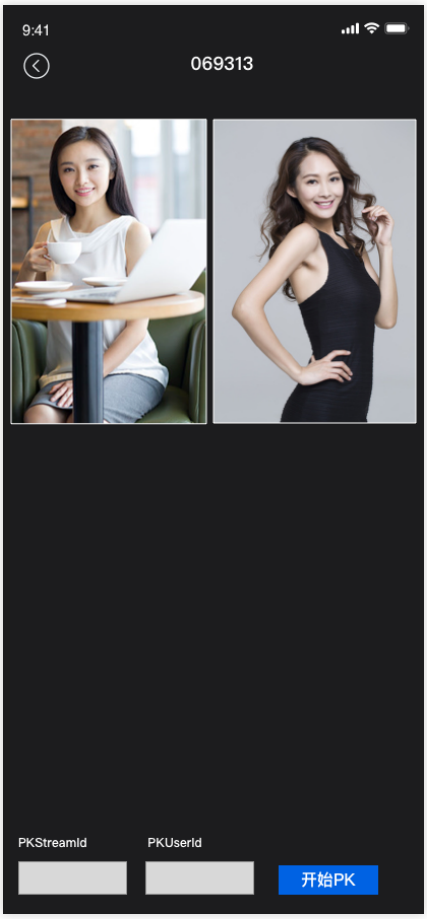
Host A (Phone A)	Host B (Phone B)	Audience of Host A (Phone C)

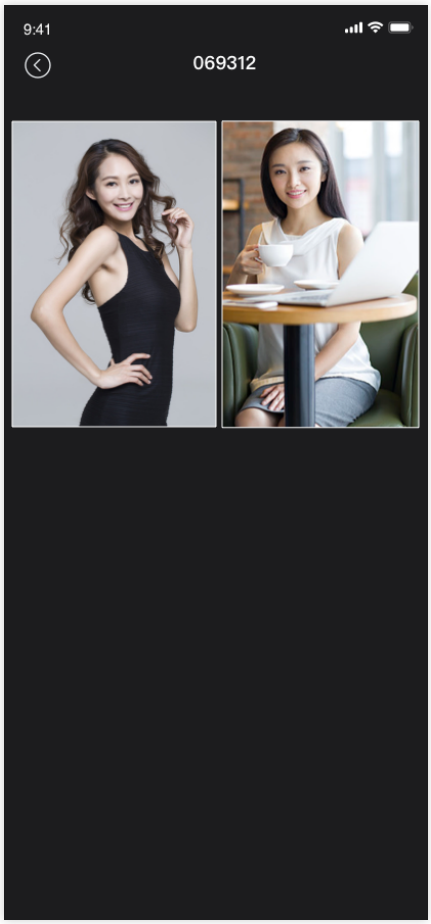
		

Competing

Host A (Phone A)	Host B (Phone B)	Audience of Host A (Phone C)



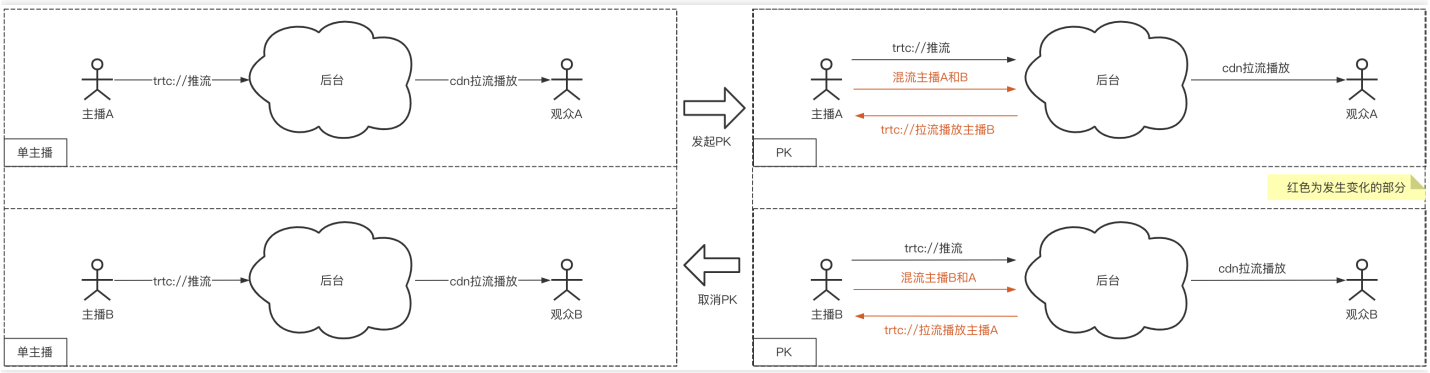




Implementation

As shown in the figure below, both host A and host B have their audience. To compete, they only need to do the following:

- Host A starts playing host B's stream and initiates a stream mixing task to mix his or her stream with host B's so that his or her audience can watch them compete.
- Host B starts playing host A's stream and initiates a stream mixing task to mix his or her stream with host A's so that his or her audience can watch them compete.
- Audience A and B can continue to play streams via CDNs and will see the competing videos of host A and host B after stream mixing.



1. Host A starts publishing

Host A calls `V2TXLivePusher` to publish a stream. For how to splice a publishing URL, please see Publish/Playback URL.

- java java
- Objective-C ObjectiveC

```
V2TXLivePusher pusher = new V2TXLivePusherImpl(this, V2TXLiveMode.TXLiveMode_RTC);
pushURLA= "trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxx";
pusher.startPush(pushURLA);
```

2. Host B starts publishing

Host B calls `V2TXLivePusher` to publish a stream. For how to splice a publishing URL, please see Publish/Playback URL.

- java java
- Objective-C ObjectiveC

```
V2TXLivePusher pusher = new V2TXLivePusherImpl(this, V2TXLiveMode.TXLiveMode_RTC);
pushURLB "trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=B&usersig=xxx";
pusher.startPush(pushURLB);
```

3. Start competition

Host A and host B call `V2TXLivePlayer` to play each other's stream and start RTC-based competition. For how to splice a playback URL, please see Publish/Playback URL.

- java java
- Objective-C ObjectiveC

```
// Host A
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
playURLB = "trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&userId=B&usersig=xxx&appscene=live"
player.startPlay(playURLB);
...

// Host B
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
playURLA= "trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&userId=A&usersig=xxx&appscene=live"
player.startPlay(playURLA);
```

4. Audience watch the hosts compete

After host competition starts, audience can watch via one of two methods.

1. Host A's audience calls `V2TXLivePlayer` to play host B's stream, and host B's audience calls `V2TXLivePlayer` to play host A's stream.
2. Host A and host B mix their streams, and audience use the original URL to play the mixed stream.
   Host A and host B each initiate a stream mixing task to mix their streams so that their audience can watch them complete. To achieve this, the hosts need to call `setMixTranscodingConfig` to start On-Cloud MixTranscoding, specifying audio-related parameters including `audioSampleRate`, `audioBitrate`, and `audioChannels`.

Sample code:

- java java
- Objective-C ObjectiveC

```
// Host A
V2TXLiveDef.V2TXLiveTranscodingConfig config = new V2TXLiveDef.V2TXLiveTranscodingConfig();

// Set the resolution to 720 × 1280 px, bitrate 1500 Kbps, and frame rate 20 fps
config.videoWidth = 720;
```

```
config.videoHeight = 1280;
config.videoBitrate = 1500;
config.videoFramerate = 20;
config.videoGOP = 2;
config.audioSampleRate = 48000;
config.audioBitrate = 64;
config.audioChannels = 2;
config.mixStreams = new ArrayList<>();

// Position of the camera image of host A
V2TXLiveDef.V2TXLiveMixStream local = new V2TXLiveDef.V2TXLiveMixStream();
local.userId = "localUserId";
local.streamId = null; // `streamID` is required for the remote user but not for the local user
local.x = 0;
local.y = 0;
local.width = videoWidth;
local.height = videoHeight;
local.zOrder = 0; // When `zOrder` is set to `0`, it indicates that the host's image is displayed at the bottom
config.mixStreams.add(local);

// Position of the camera image of host B
V2TXLiveDef.V2TXLiveMixStream remoteB = new V2TXLiveDef.V2TXLiveMixStream();
remoteB.userId = "remoteUserIdB";
remoteB.streamId = "remoteStreamIdB"; // `streamID` is required for the remote user but not for the local user
remoteB.x = 400; // For reference only
remoteB.y = 800; // For reference only
remoteB.width = 180; // For reference only
remoteB.height = 240; // For reference only
remoteB.zOrder = 1;
config.mixStreams.add(remoteB);

// Start On-Cloud MixTranscoding
pusher.setMixTranscodingConfig(config);

//Host B
V2TXLiveDef.V2TXLiveTranscodingConfig config = new V2TXLiveDef.V2TXLiveTranscodingConfig();

// Set the resolution to 720 × 1280 px, bitrate 1500 Kbps, and frame rate 20 fps
config.videoWidth = 720;
config.videoHeight = 1280;
config.videoBitrate = 1500;
config.videoFramerate = 20;
config.videoGOP = 2;
```

```
config.audioSampleRate = 48000;
config.audioBitrate = 64;
config.audioChannels = 2;
config.mixStreams = new ArrayList<>();

// Position of the camera image of host B
V2TXLiveDef.V2TXLiveMixStream local = new V2TXLiveDef.V2TXLiveMixStream();
local.userId = "localUserId";
local.streamId = null; // `streamID` is required for the remote user but not for the local user
local.x = 0;
local.y = 0;
local.width = videoWidth;
local.height = videoHeight;
local.zOrder = 0; // When `zOrder` is set to `0`, it indicates that the host's image is displayed at the bottom
config.mixStreams.add(local);

// Position of the camera image of host A
V2TXLiveDef.V2TXLiveMixStream remoteA = new V2TXLiveDef.V2TXLiveMixStream();
remoteA.userId = "remoteUserIdA";
remoteA.streamId = "remoteStreamIdA"; // `streamID` is required for the remote user but not for the local user
remoteA.x = 400; // For reference only
remoteA.y = 800; // For reference only
remoteA.width = 180; // For reference only
remoteA.height = 240; // For reference only
remoteA.zOrder = 1;
config.mixStreams.add(remoteA);

// Start On-Cloud MixTranscoding
pusher.setMixTranscodingConfig(config);
```

Note：

Since you need to maintain room and user status by yourself, the new RTC-based scheme may seem more complicated than the old one. In fact, **there isn't an always better scheme, only one that better suits your needs**.

- You can stick to the old scheme if your application scenarios do not require low latency or high concurrency.
- If you want to use V2 APIs without having to manage a room and users, try using Tencent Cloud's IM SDK to implement the necessary logic.

Billing

For billing details, please see Purchase Guide.

FAQs

1. Why is publishing and playback using the same `streamid` on the same device possible with `TXLivePusher` and `TXLivePlayer` but not with `V2TXLivePusher` and `V2TXLivePlayer`?

`V2TXLivePusher` and `V2TXLivePlayer` are based on Tencent Cloud's TRTC protocol. This is a UDP-based private protocol that features ultra-low latency and does not support **using the same `streamid` for ultra-low-latency publishing and playback on the same device**. We have determined that it's not necessary to support this given the current use cases, but may consider optimizing the protocol in the future.

2. What are the parameters mentioned in Activate TRTC?

`SDKAppID` identifies your application, and `UserID` your user. `UserSig` is a security signature calculated based on the two parameters using the **HMAC SHA256** encryption algorithm. Attackers cannot use your Tencent Cloud traffic without authorization as long as they cannot forge a `UserSig`. `UserSig` calculation involves hashing crucial information such as `SDKAppID`, `UserID`, and `ExpireTime`, as shown below.

```
// UserSig formula, in which `secretkey` is the key used to calculate UserSig
usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
base64(userid + sdkappid + currtime + expire)))
```

3. How can I set audio or video quality using `V2TXLivePusher` and `V2TXLivePlayer`?

We provide APIs for the setting of audio and video quality. For details, please see setAudioQuality() and setVideoQuality:resolutionMode:().

4. What does the error code `-5` mean?

The error code `-5` means failure to call an API due to invalid license. The enumerated value is V2TXLIVE_ERROR_INVALID_LICENSE. For other error codes, please see V2TXLiveCode.

5. What is the typical latency of RTC-based mic connect?

In the new RTC-based mic connect scheme, the mic connect latency is lower than 200 ms, and the latency for hosts and audience is 100-1,000 ms.

6. What should I do if the `404` error occurs when I try to play streams via CDNs after successfully publishing streams over RTC?

Check if you have enabled TRTC's relayed push feature. The feature is needed because, after publishing streams via RTC, to enable CDN playback, you need to relay the streams to CDNs.