移动直播 SDK

观众连麦

产品文档

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

观众连麦

最近更新时间：2022-03-31 11:31:05

连麦方案介绍

目前，在 **连麦互动 - RTMP方案** 中，腾讯云视立方 · 移动直播 SDK 提供连麦互动组件 MLVBLiveRoom 用来帮助开发者快速实现连麦需求，为了更好的满足开发者对连麦功能的需求，腾讯云新增了**基于实时音视频 TRTC 能力实现的新版连麦方案**，同时提供了更加简单灵活的全新接口：V2TXLivePusher （推流）、V2TXLivePlayer （拉流）。

移动直播全新接口同时支持通过 RTMP 协议及 RTC 协议进行推流/连麦，开发者可根据自身需求选择适合的方案，对比如下：

对比项	旧版连麦方案	新版连麦方案
协议	RTMP 基于 TCP 协议	RTC 基于 UDP 协议（更适合流媒体传输）
QoS	弱网抗性能力弱	50%丢包率可正常视频观看，70%丢包率可正常语音连麦
支持区域	仅支持中国内地（大陆）地区	全球覆盖
使用产品	需开通移动直播、云直播服务	需开通移动直播、云直播、实时音视频服务
价格	0.016元/分钟	阶梯价格，详情请参见 RTC 连麦方案怎么计算费用

新版连麦方案演示

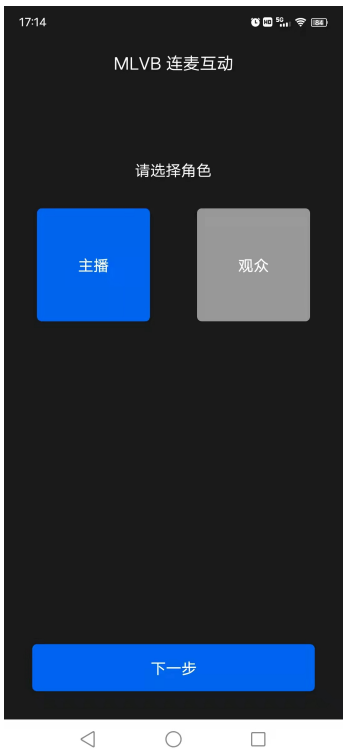
新版连麦方案用来帮助客户实现更加灵活、更低延时、更多人数的直播互动场景。开播端可以利用全新接口提供的 RTC 推流能力，默认情况下，观众端观看则可使用 CDN 方式进行拉流。 CDN 观看费用较低。如果观众端有连麦需求，连麦观众上麦后，可以从 CDN 切换到 RTC 进行观看，这样延时更低，互动效果更好。新版连麦方案需要开通相关云服务和参数配置，详情请参见 配置连麦或 PK 能力。

为了方便用户快速接入，我们在音视频终端 SDK 控制台整理推出连麦管理功能，通过简单配置即可快速跑通 MLVB-API-Example Demo，体验新版连麦方案，全方位管理连麦应用，实时查看连麦应用相关用量统计，以及连麦地址生成器。若您是首次接入，推荐您前往 音视频终端 SDK 控制台 - 连麦管理 体验，再自行接入移动直播 SDK。

下面是 MLVB-API-Example Demo 的演示效果。

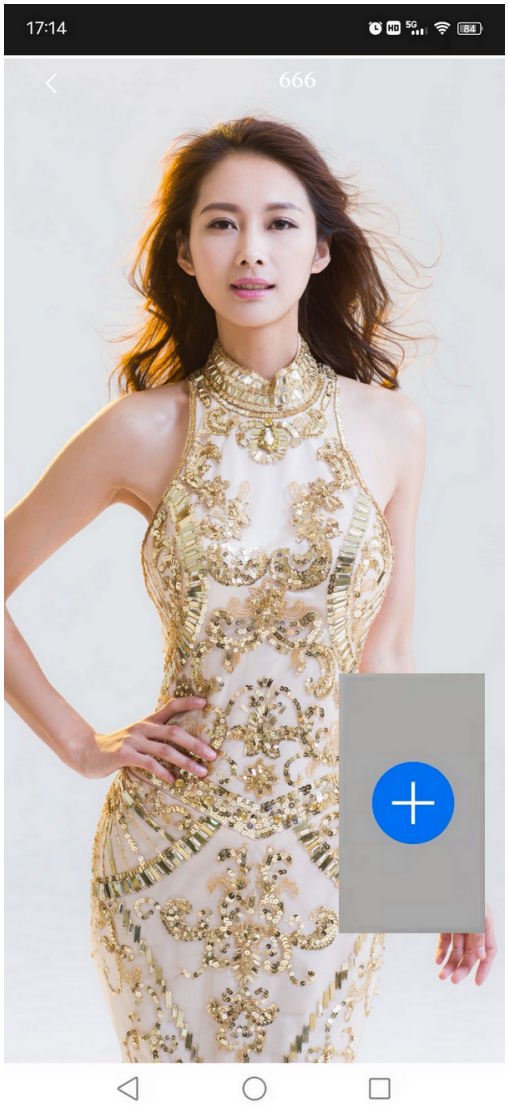
演示图示

直播前

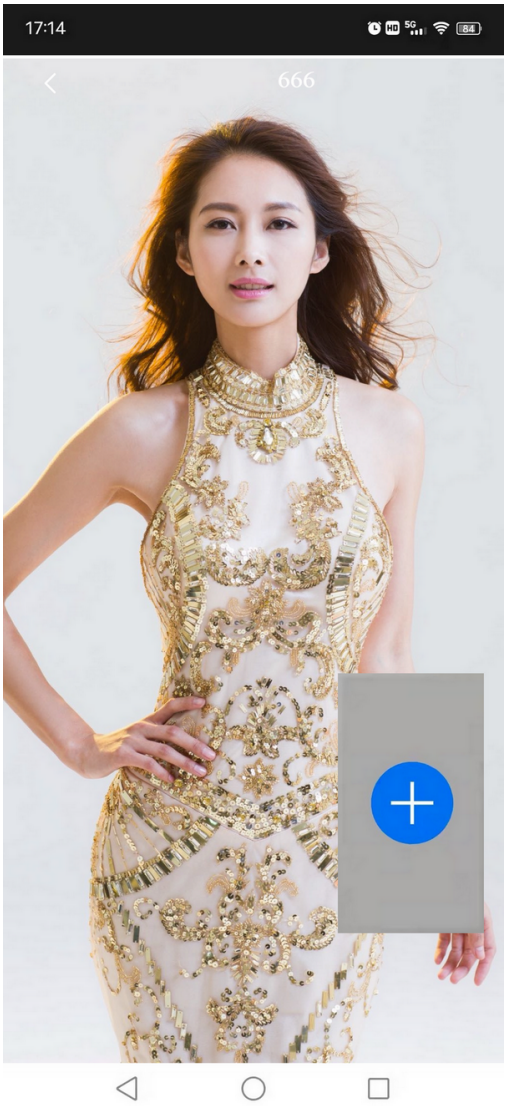
主播			
连麦观众和观众			

连麦操作

主播		



连麦观众



连麦中

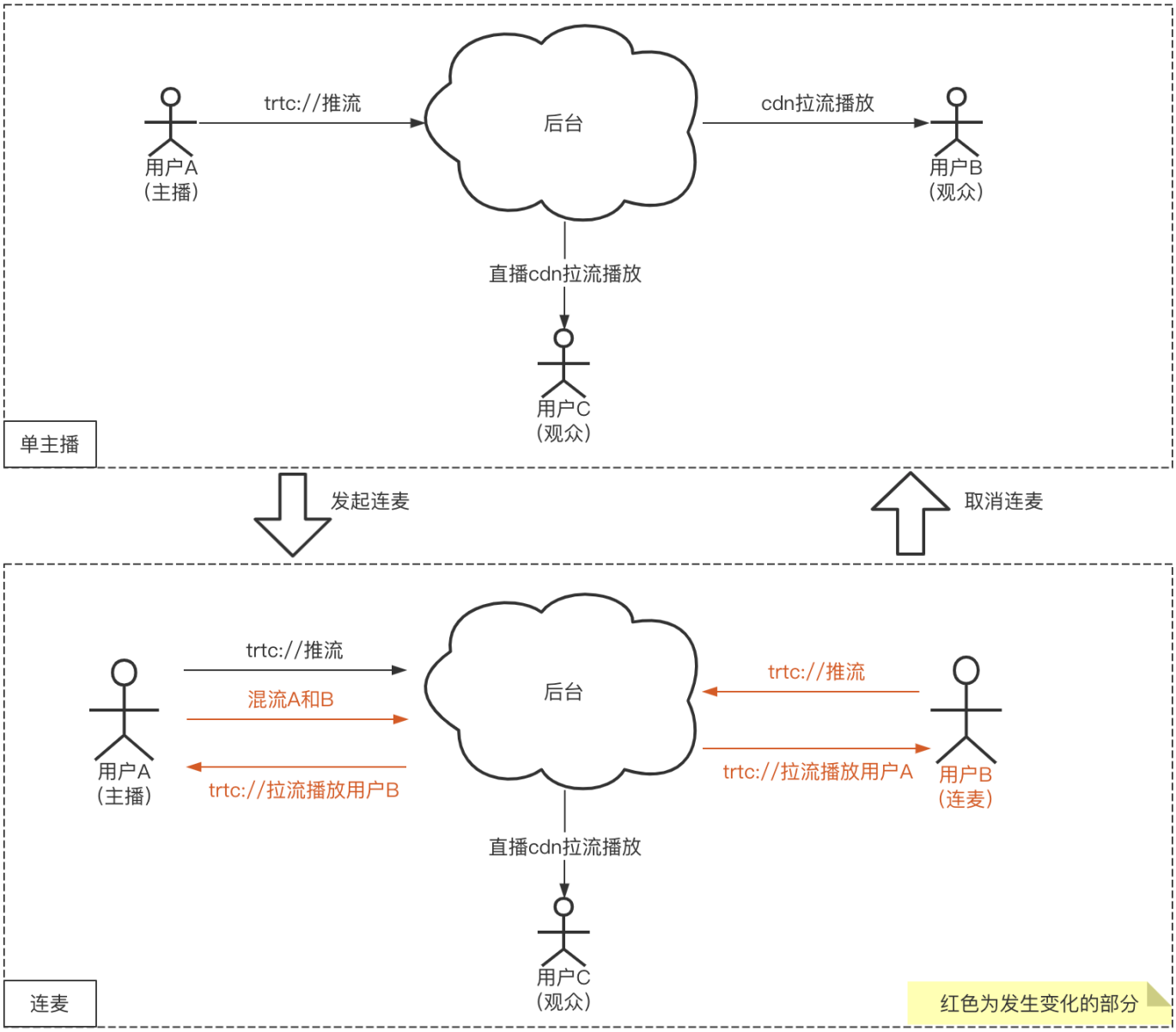
主播端（手机 A）	连麦观众（手机 B）	普通观众（手机 C）

RTC 连麦功能实现

如下示意图，用户 A 是主播，B 和 C 都是观众，如果观众 B 要与 主播 A 连麦，需要做的事情非常简单：

- 观众 B：启动 trtc:// 协议推流，播放流也从 CDN 切到超低延迟的 trtc:// 协议。
- 主播 A：开始播放观众 B 的流，同时发起混流指令，把 A 和 B 的内容合成一路。

- 观众 C：无需变化，继续 CDN 播放即可，只不过会看到混流后的连麦画面。



1. 主播 RTC 推流

主播 A 开始推流，调用 V2TXLivePusher 组件开始主播 A 的推流。URL 拼装方案请参见 如何拼装 URL。

java

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

Objective-C

```
V2TXLivePusher *pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTC];
NSString *pushURLA = @"trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxx";
[pusher startPush:pushURLA];
```

2. 观众 CDN 拉流

所有观众观看主播 A 直播，调用 V2TXLivePlayer 开始播放主播 A 的流。URL 拼装方案请参见 如何拼装 URL。

java

```
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
/**

* 这里使用 CDN 拉流，支持 FLV、HLS、WebRTC 协议，任选一种协议。FLV、HLS 等标准协议价格更合理，WebRTC 快直播能够提供更低延迟的互动体验。
* playURLA= "http://3891.liveplay.myqcloud.com/live/streamidA.flv";
* playURLA= "http://3891.liveplay.myqcloud.com/live/streamidA.hls";
* playURLA= "webrtc://3891.liveplay.myqcloud.com/live/streamidA"
*/
player.startPlay(playURLA);
```

Objective-C

```
V2TXLivePlayer * player = [[V2TXLivePlayer alloc] init];
/**
* 这里使用 CDN 拉流，支持 FLV、HLS、WebRTC 协议，任选一种协议。FLV、HLS 等标准协议价格更合理，WebRTC 快直播能够提供更低延迟的互动体验。
* NSString *playURLA= @"http://3891.liveplay.myqcloud.com/live/streamidA.flv";
* NSString *playURLA= @"http://3891.liveplay.myqcloud.com/live/streamidA.hls";
* NSString *playURLA= @"webrtc://3891.liveplay.myqcloud.com/live/streamidA"
*/
[player setRenderView:view];
[player startPlay:playURLA];
```

3. 观众发起连麦

其中观众 B 调用 V2TXLivePusher 发起推流（后续会称呼为连麦观众 B）。

java

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

Objective-C

```
V2TXLivePusher *pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTC];
NSString *pushURLB = @"trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=B&usersig=xxx";
[pusher startPush:pushURLB];
```

4. 进入连麦状态

主播 A 调用 V2TXLivePlayer **使用 RTC 协议拉取放连麦观众 B 的流。**

java

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

Objective-C

```
V2TXLivePlayer * player = [[V2TXLivePlayer alloc] init];
NSString* playURLB = @"trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&userId=B&usersig=xxx&appscene=live";
[player setRenderView:view];
[player startPlay:playURLB];
```

同时，连麦观众 B 调用 V2TXLivePlayer **切换至 RTC 协议，开始播放主播 A 的流。**

java

```
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
playURLA= "trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&userId=A&user
```

```
sig=xxx&appscene=live";
player.startPlay(playURLA);
```

Objective-C

```
V2TXLivePlayer * player = [[V2TXLivePlayer alloc] init];
NSString* playURLA = @"trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&us
erId=A&usersig=xxx&appscene=live";
[player setRenderView:view];
[player startPlay:playURLA];
```

此时主播 A 和 连麦观众 B 即可进入超低延时的实时互动场景中。

5. 连麦成功后，进行混流

为了保证观众可以看到连麦观众 B 的画面，这里主播 A 需要发起一次混流操作。也就是将主播 A 自己和连麦观众 B，混合成一路流。观众可以在一路流上看到主播和连麦观众进行互动。A 调用 setMixTranscodingConfig 接口启动云端混流，调用时需要设置音频相关的参数，例如 音频采样率 audioSampleRate **、音频码率** audioBitrate **和 声道数** audioChannels **等。**
如果您的业务场景中也包含视频，需同时设置视频相关的参数，例如 视频宽度 videoWidth **、视频高度** videoHeight **、视频码率** videoBitrate **、视频帧率** videoFramerate **等。**

示例代码：

java

```
V2TXLiveDef.V2TXLiveTranscodingConfig config = new V2TXLiveDef.V2TXLiveTranscodingCo
nfig();
// 设置分辨率为 720 × 1280, 码率为 1500kbps，帧率为 20FPS
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<>();

// 主播摄像头的画面位置
V2TXLiveDef.V2TXLiveMixStream local = new V2TXLiveDef.V2TXLiveMixStream();
```

```
local.userId = "localUserId";
local.streamId = null; // 本地画面不用填写 streamID，远程需要
local.x = 0;
local.y = 0;
local.width = videoWidth;
local.height = videoHeight;
local.zOrder = 0; // zOrder 为 0 代表主播画面位于最底层
config.mixStreams.add(local);

// 连麦者的画面位置
V2TXLiveDef.V2TXLiveMixStream remoteA = new V2TXLiveDef.V2TXLiveMixStream();
remoteA.userId = "remoteUserIdA";
remoteA.streamId = "remoteStreamIdA"; // 本地画面不用填写 streamID，远程需要
remoteA.x = 400; //仅供参考
remoteA.y = 800; //仅供参考
remoteA.width = 180; //仅供参考
remoteA.height = 240; //仅供参考
remoteA.zOrder = 1;
config.mixStreams.add(remoteA);

// 连麦者的画面位置
V2TXLiveDef.V2TXLiveMixStream remoteB = new V2TXLiveDef.V2TXLiveMixStream();
remoteB.userId = "remoteUserIdB";
remoteB.streamId = "remoteStreamIdB"; // 本地画面不用填写 streamID，远程需要
remoteB.x = 400; //仅供参考
remoteB.y = 800; //仅供参考
remoteB.width = 180; //仅供参考
remoteB.height = 240; //仅供参考
remoteB.zOrder = 1;
config.mixStreams.add(remoteB);

// 发起云端混流
pusher.setMixTranscodingConfig(config);
```

Objective-C

```
V2TXLiveTranscodingConfig *config = [[V2TXLiveTranscodingConfig alloc] init];
// 设置分辨率为 720 × 1280, 码率为 1500kbps，帧率为 20FPS
config.videoWidth = 720;
```

```
config.videoHeight = 1280;
config.videoBitrate = 1500;
config.videoFramerate = 20;
config.videoGOP = 2;
config.audioSampleRate = 48000;
config.audioBitrate = 64;
config.audioChannels = 2;

// 主播摄像头的画面位置
V2TXLiveMixStream *local = [[V2TXLiveMixStream alloc] init];
local.userId = @"localUserId";
local.streamId = nil; // 本地画面不用填写 streamID，远程需要
local.x = 0;
local.y = 0;
local.width = videoWidth;
local.height = videoHeight;
local.zOrder = 0; // zOrder 为 0 代表主播画面位于最底层

// 连麦者的画面位置
V2TXLiveMixStream *remoteA = [[V2TXLiveMixStream alloc] init];
remoteA.userId = @"remoteUserIdA";
remoteA.streamId = @"remoteStreamIdA"; // 本地画面不用填写 streamID，远程需要
remoteA.x = 400; //仅供参考
remoteA.y = 800; //仅供参考
remoteA.width = 180; //仅供参考
remoteA.height = 240; //仅供参考
remoteA.zOrder = 1;

// 连麦者的画面位置
V2TXLiveMixStream *remoteB = [[V2TXLiveMixStream alloc] init];
remoteB.userId = @"remoteUserIdB";
remoteB.streamId = @"remoteStreamIdB"; // 本地画面不用填写 streamID，远程需要
remoteB.x = 400; //仅供参考
remoteB.y = 800; //仅供参考
remoteB.width = 180; //仅供参考
remoteB.height = 240; //仅供参考
remoteB.zOrder = 1;

//设置混流 streams
config.mixStreams = @[local,remoteA,remoteB];
```

```
// 发起云端混流
pusher.setMixTranscodingConfig(config);
```

> ⚠ **注意：**
> 发起云端混流后，默认混流 ID，是发起混流者的 ID，如果需要指定流 ID，需要进行传入。

这样其他观众在观看时，就可以看到 A，B 两个主播的连麦互动。

> ⓘ **说明：**
> 此处开发者可能会有疑问：貌似新的 RTC 连麦方案还需要我们自己维护一套房间和用户状态，这样不是更麻烦吗？是的，**没有更好的方案，只有更适合自己的方案**，我们也有考虑到这样的场景：
>
> - 如果对时延和并发要求并不高的场景，可以继续使用连麦互动的旧方案。
> - 如果既想用到新版连麦方案相关的接口，但是又不想维护一套单独的房间状态，可以尝试搭配 腾讯云 IM SDK，快速实现相关逻辑。

RTC 连麦方案怎么计算费用

具体请参见 连麦互动费用-新方案（RTC 连麦）。

常见问题

1. 为什么使用 V2TXLivePusher&V2TXLivePlayer 接口时，同一台设备不支持使用相同 streamid 同时推流和拉流，而 TXLivePusher&TXLivePlayer 可以支持?

是的，目前 V2TXLivePusher&V2TXLivePlayer 是 腾讯云 TRTC 协议实现，其基于 UDP 的超低延时的私有协议暂时还不支持**同一台设备，使用相同的 streamid，一边推超低延时流，一边拉超低延时的流**，同时考虑到用户的使用场景，所以暂时并未支持，后续会酌情考虑此问题的优化。

2. 服务开通 章节中生成参数都是什么意思呢?

SDKAppID 用于标识您的应用，UserID 用于标识您的用户，而 UserSig 则是基于前两者计算出的安全签名，它由 HMAC SHA256 加密算法计算得出。只要攻击者不能伪造 UserSig，就无法盗用您的云服务流量。
UserSig 的计算原理如下图所示，其本质就是对 SDKAppID、UserID、ExpireTime 等关键信息进行了一次哈希加密：

```
//UserSig 计算公式，其中 secretkey 为计算 usersig 用的加密密钥
```

```
usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
base64(userid + sdkappid + currtime + expire)))
```

3. V2TXLivePusher&V2TXLivePlayer 如何设置音质或者画质呢?

我们有提供对应的音质和画质的设置接口，详情见 API 文件：设置推流音频质量 和 设置推流视频参数。

4. 收到一个错误码：-5，代表什么意思?

-5表示由于许可证无效，因此无法调用API，对应的枚举值为：V2TXLIVE_ERROR_INVALID_LICENSE，更多错误码请参见 API 状态码。

5. RTC连麦方案的时延性有可以参考的数据吗?

新的 RTC 连麦方案中，主播连麦的延时 < 200ms，主播和观众的延时在 100ms - 1000ms。

6. RTC 推流成功后，使用 CDN 拉流一直提示404?

检查一下是否有开启实时音视频服务的旁路直播功能，基本原理是 RTC 协议推流后，如果需要使用 CDN 播放，RTC 会在后台服务中旁路流信息到 CDN 上。