# 直播 SDK

# 常见问题

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【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或95716。

# 文档目录



版权所有：腾讯云计算（北京）有限责任公司

# 常见问题
# 推拉流 URL

最近更新时间：2025-01-21 10:42:32

## 前提条件

- 已注册腾讯云账号，并开通 腾讯云直播服务 。
- 已在 域名注册 申请域名，并备案成功。
- 已在**云直播控制台** > 域名管理 中添加推流/播放域名，具体操作请参见 添加自有域名 。
- 成功 配置域名 CNAME 。

## RTMP 协议推流（不支持连麦）

直播 SDK 支持 RTMP（不支持连麦）和 RTC（支持连麦）两种推流协议，本章节介绍 RTMP 推拉流 URL 的生成。

### 控制台生成推流 URL

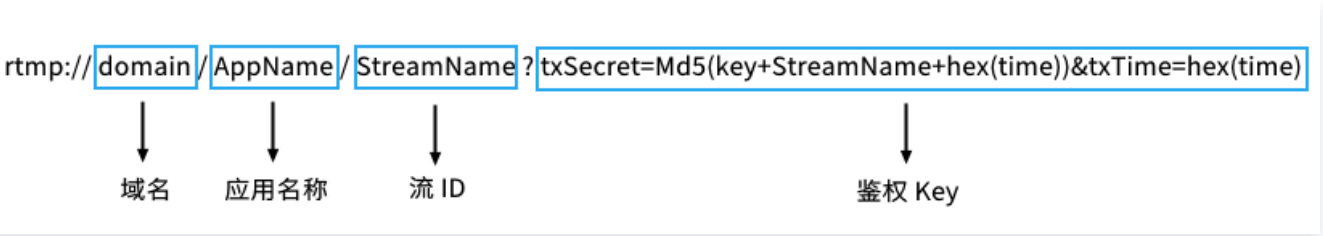
1. 登录云直播控制台。
2. 选择进入**直播工具箱** > 地址生成器，进入如下配置：
   - 按需选择生成类型。
   - 选择您已添加到域名管理里对应的域名。
   - 按需编辑 AppName。AppName 为区分同一个域名下多个 App 的地址路径，默认为 live。
   - 填写自定义的流名称 StreamName。
   - 选择地址过期时间。
3. 单击**生成地址**即可生成您需要的推流/播放地址。


**说明：**

- AppName 可自定义，仅支持英文字母、数字和符号。
- 除上述方法，您还可以在云直播控制台的 域名管理 **中，选择推流域名单击**管理**，选择**推流配置**，输入推流地址的过期时间和自定义的流名称 StreamName，**单击**生成推流地址**即可生成推流地址。

### 推流 URL 拼接规则

实际产品中，当直播间较多时，您不可能为每一个主播手工创建推流和播放 URL，您可通过服务器**自行拼装**推流和播放地址，只要符合腾讯云标准规范的 URL 就可以用来推流，如下是一条标准的推流 URL，它由四个部分组成：




- **Domain**
  推流域名，可使用腾讯云直播提供的默认推流域名，也可以用自有已备案且 CNAME 配置成功的推流域名。
- **AppName**
  直播的应用名称，默认为 live，可自定义。
- **StreamName（流 ID）**
  自定义的流名称，每路直播流的唯一标识符，推荐用随机数字或数字与字母组合。
- **鉴权 Key（非必需）**
  包含 txSecret 和 txTime 两部分：`txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)` 。
  开启推流鉴权后需使用包含鉴权 Key 的 URL 进行推流。若未开启推流鉴权，则推流地址中无需 “?” 及其后内容。
- **txTime（地址有效期）**
  表示何时该 URL 会过期，格式支持十六进制的 UNIX 时间戳。

  > **说明：**
  > 例如 `5867D600` 代表2017年1月1日0时0点0分过期，我们的客户一般会将 txTime 设置为当前时间24小时以后过期，过期时间不要太短也不要太长，当主播在直播过程中遭遇网络闪断时会重新恢复推流，如果过期时间太短，主播会因为推流 URL 过期而无法恢复推流。

- **txSecret（防盗链签名）**
- 用以防止攻击者伪造您的后台生成推流 URL，计算方法参见 防盗链计算 。

## 查看推流地址示例代码

进入**云直播控制台** > **域名管理**，选中事先配置的推流域名，**管理** > **推流配置**页面下半部分有**推流地址示例代码**（PHP 和 Java 两个版本）演示如何生成防盗链地址。更多详情操作请参见 **推流配置** 。

## 播放 URL 拼接规则

播放地址主要由播放前缀、播放域名（domain）、应用名称（AppName）、流名称（StreamName）、播放协议后缀、鉴权参数以及其他自定义参数组成。例如：

```
webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.flv?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.m3u8?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

- **播放前缀**

| 播放协议 | 播放前缀 | 备注 |
| --- | --- | --- |

| | | |
|---|---|---|
| WebRTC | webrtc:// | 强烈推荐，秒开效果最好，支持超高并发。 |
| HTTP-FLV | http:// 或 https:// | 推荐，秒开效果好，支持超高并发。 |
| RTMP | rtmp:// | 不推荐，秒开效果差，不支持高并发 |
| HLS（m3u8） | http:// 或 https:// | 手机端和 Mac safari 浏览器推荐的播放协议。 |

- **Domain播放域名，自有已备案且 CNAME 配置成功的播放域名。**
- **AppName**
  直播的应用名称，用于区分直播流媒体文件存放路径，默认为 live，可自定义。
- **StreamName（流名称）**
  自定义的流名称，每路直播流的唯一标识符，推荐用随机数字或数字与字母组合。
- **鉴权参数（非必需）**
  包含 txSecret 和 txTime 两部分：`txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)` 。
  开启播放鉴权后需使用包含鉴权 Key 的 URL 进行播放。若未开启播放鉴权，则播放地址中无需 “?” 及其后内容。
- **txTime（地址有效期）：** 表示何时该 URL 会过期，格式支持十六进制的 UNIX 时间戳。
- **txSecret（防盗链签名）：** 用以防止攻击者伪造您的后台生成播放 URL，计算方法参见 防盗链计算 。

# RTC 协议推流（支持连麦）

直播 SDK 支持 RTMP（不支持连麦）和 RTC（支持连麦）两种推流协议，本章节介绍 RTC 推拉流 URL 的生成。
如果您有 观众连麦 或者 直播 PK 的需求，需要使用支持更低延迟、更好弱网抗性的 RTC 协议进行推拉流。

## 控制台生成推流 URL

1. 登录云直播控制台。
2. 选择进入**连麦管理** > **地址生成器**，进入如下配置：
   - 按需选择连麦应用。如果您当前没有连麦应用，可以在 连麦应用 > **新建连麦应用**新建一个。
   - 填写**主播 Stream Id** 和**主播 User Id**，这一步骤是为了生成主播的推流和播放地址。
   - 按需填写**连麦观众 Stream Id** 和**连麦观众 User Id**，这一步骤是为了生成连麦观众的推流和播放地址，如果您当前没有连麦观众，暂时可以随意填写。
   - 选择域名，这一步骤是为了生成 CDN 观看地址。
   - 按需编辑 AppName。AppName 为区分同一个域名下多个 App 的地址路径，默认为 live。
   - 选择地址过期时间。
3. 单击**生成地址**即可生成您需要的推流/播放地址。

> **说明：**
> 更多使用手动生成 URL 的介绍请参见 控制台指南 。

## 推流 URL 拼接规则

实际产品中，当直播间较多时，您不可能为每一个主播手工创建推流和播放 URL，您可以按照如下规范在工程代码中自动拼接 URL，如下是一条标准的推流 URL，示例如下：

```
trtc://cloud.tencent.com/push/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx
```

在上述的 URL 中，存在一些关键字段，关于其中关键字段的含义信息，详见下表：

| 字段名称 | 字段含义 |
| --- | --- |
| trtc:// | 互动直播推流 URL 的前缀字段 |
| cloud.tencent.com | 互动直播特定域名，请勿修改 |
| push | 标识位，表示推流 |
| streamid | 流 ID，需要由开发者自定义 |
| sdkappid | 对应 服务开通 一节中生成的 SDKAppID |
| userId | 用户 ID，需要由开发者自定义 |
| usersig | 由 服务开通 一节中获取的密钥计算得出 |

## 播放 URL 拼接规则

- 在连麦过程中，主播与连麦者相互之间的观看都要用 RTC来播放，播放的 URL 字符串与推流 URL 只有一个字段的差别，把 `push` 换成 `play` 即可，示例如下：

```
trtc://cloud.tencent.com/play/streamid?sdkappid=1400188888&userId=A&usersig=xxxxx
```

在上述的 URL 中，存在一些关键字段，关于其中关键字段的含义信息，详见下表：

| 字段名称 | 字段含义 |
| --- | --- |
| trtc:// | 互动直播拉流 URL 的前缀字段 |
| cloud.tencent.com | 互动直播特定域名，请勿修改 |
| play | 标识位，表示拉流 |
| streamid | 流 ID，需要由开发者自定义 |
| sdkappid | 对应 服务开通 一节中生成的 SDKAppID |
| userId | 用户 ID，需要由开发者自定义 |
| usersig | 由 服务开通 一节中获取的密钥计算得出 |

- CDN 的观看地址与前文中的 播放 URL 规则一致。

# Crash 相关问题

最近更新时间：2023-10-11 16:20:23

## 1. 启动屏幕分享出现如下 crash?

```
Caused by: java.lang.SecurityException: Media projections require a foreground service of
type ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION
```

### 解决方案：

在使用 SDK，将 targetSdkVersion 设置为 30，进行屏幕分享时会出现如下崩溃，这主要是因为谷歌隐私策略导致的，需要启动一个前台的 Service，并且 `android:foregroundServiceType="mediaProjection"` 才可以解决，具体步骤如下：

1. 创建一个 Service ，并绑定一个 Notification 使其作为前台 Service。

```
public class TestService extends Service {

    private final static String NOTIFICATION_CHANNEL_ID =
"com.tencent.liteav.demo.TestService";
    private final static String NOTIFICATION_CHANNEL_NAME =
"com.tencent.liteav.demo.channel_name";
    private final static String NOTIFICATION_CHANNEL_DESC =
"com.tencent.liteav.demo.channel_desc";

    @Override
    public void onCreate() {
        super.onCreate();
        startNotification();
    }

    @Override
    public IBinder onBind(Intent intent) {
        throw new UnsupportedOperationException("Not yet implemented");
    }

    private void startNotification() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.O) {
            return;
        }

        Intent notificationIntent = new Intent(this, TestService.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
notificationIntent, 0);
        NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this,
                NOTIFICATION_CHANNEL_ID).setLargeIcon(
                    BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher_foreground))
                        .setSmallIcon(R.drawable.ic_launcher_foreground)
                        .setContentTitle("Starting Service")
                        .setContentText("Starting monitoring service")
```

```
                        .setContentIntent(pendingIntent);
        Notification notification = notificationBuilder.build();
        NotificationChannel channel = new NotificationChannel(NOTIFICATION_CHANNEL_ID,
                NOTIFICATION_CHANNEL_NAME, NotificationManager.IMPORTANCE_DEFAULT);
        channel.setDescription(NOTIFICATION_CHANNEL_DESC);
        NotificationManager notificationManager =
(NotificationManager)getSystemService(
                Context.NOTIFICATION_SERVICE);
        notificationManager.createNotificationChannel(channel);
        //必须使用此方法显示通知，不能使用notificationManager.notify，否则还是会报上面的错误
        startForeground(1, notification);
    }
}
```

2. **在 AndroidManifest.xml 中配置：**

2.1 加入权限

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

2.2 注册 Service

```
<service
    android:name=".TestService"
    android:foregroundServiceType="mediaProjection"
    android:enabled="true"
    android:exported="false" />
```

3. **在启动推流前，请先启动此 Service：**

```
startService(new Intent(this, TestService.class));
```

> ⚠ **注意：**
> 9.0 及之后的系统，应用退后台后 摄像头和麦克风将停止工作，该 Service 也可保证应用退后台摄像头和麦克风依旧可以正常工作。

## 2. 应用启动出现如下 crash?

```
java.lang.UnsatisfiedLinkError: No implementation found for byte[]
com.tencent.liteav.basic.license.LicenceCheck.nativeIvParameterSpec(byte[]) (tried
Java_com_tencent_liteav_basic_license_LicenceCheck_nativeIvParameterSpec and
Java_com_tencent_liteav_basic_license_LicenceCheck_nativeIvParameterSpec___3B);
```

```
java.lang.UnsatisfiedLinkError: No implementation found for boolean
com.tencent.liteav.sdk.common.LicenseChecker.nativeSetLicense
```

```
java.lang.UnsatisfiedLinkError: No implementation found for void
com.tencent.liteav.basic.log.TXCLog.nativeLogInit()
```

```
java.lang.UnsatisfiedLinkError: No implementation found for void
com.tencent.liteav.base.util.LiteavLog.nativeSetConsoleLogEnabled(boolean)
```

出现 java.lang.UnsatisfiedLinkError 异常，一般是 so 加载失败了。

## 解决方案：

- **Android Studio 编译错误**

  可能和 Android Studio 编译有关，so 没有正确编译到 APK 内，此时 Clean Build，然后重新编译即可。
- **设备 CPU 架构和打包到 APK 内的 so 架构不匹配**

  请检查设备 CPU 架构和打包到 APK 内部的 so 是否匹配，如果不匹配，请在 build.gradle 中正确配置：

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

# 开通各项云服务

最近更新时间：2025-06-12 10:30:12

## 云直播

### 如何开通云直播服务?

进入 云直播管理控制台，进入腾讯云直播服务开通页，查看相关协议并勾选**同意**，单击**申请开通** 即可开通云直播服务。


### 如何开启流防盗链 KEY?

推流防盗链 KEY 是为了确保只有您的 App 用户才可以推流的安全保护手段，可随时按您的需要在直播管理控制台 域名管理 中修改，详细请参见 播放鉴权配置 。

### 如何通过 API 访问鉴权 KEY?

API 访问鉴权 KEY 是您的后台服务器在调用云直播相关的 云 API 接口 时需要用到的，目的是帮助腾讯云确认调用的合法性，API 访问鉴权 KEY 的获取请参见 接口鉴权 。

### 如何回调事件通知 URL?

腾讯云可以对一些直播事件配置回调，腾讯云服务会以 HTTP POST 的形式向您配置的地址发送通知，事件回调 URL 修改请参见 回调配置 。

## 云点播

### 如何开通云点播服务?

只需要在 云点播管理控制台 开通即可以使用云点播服务，默认按照日结方式计费，您也可以选购合适的资源包抵扣。

### 如何查询我的点播 APPID?

每个腾讯云账号均有一个唯一的 APPID 与之对应，其位于 账号信息 中。


## 即时通信（IM）

### 如何开通 IM 服务?

进入 即时通信 IM 管理控制台 。
新认证的腾讯云账号，可参照指引，单击**立即开始**，即可创建一个新的应用跑通 Demo：


## 什么是 SDK APPID?

下图中所示的数字即为 SDKAPPID，它表示您在该腾讯云账号下的一款产品，如果您有多个产品，就对应有多个 SDKAPPID。


## 什么是 Administrator?

IM 提供了一套 REST API 用于让您的后台服务器可以直接调用 IM 服务，例如建群、发送系统消息、把某个用户剔除群组等等，但 IM REST API 仅允许管理员进行调用，也就是需要一个管理员用户名（Administrator）和对应的密码（UserSig），具体使用过程请参见 REST API 简介 。


> **说明：**
> UserSig 是用户登录即时通信 IM 的密码，具体获取密钥方法请参见 获取密钥 。

# 小程序相关问题

最近更新时间：2023-09-19 18:43:55

## 小程序资质相关?

出于政策和合规的考虑，微信暂时没有放开所有小程序对 <live-pusher> 和 <live-player> 标签的支持。

- 小程序推拉流标签不支持个人小程序，只支持企业类小程序。
- 小程序推拉流标签使用权限暂时只开放给有限的类目，具体可以参考 <live-pusher> 类目 和 <live-player> 类目 。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 接口设置 中自助开通该组件权限，如下图所示：


## 小程序是否支持 uniapp?

不支持 uniapp 开发环境，请使用原生小程序开发环境。

# 直播基础相关问题

最近更新时间：2023-10-11 16:20:23

## 推流、直播和点播分别是什么？

- **推流**：主播将本地视频源和音频源推送到腾讯视频云服务器，在有些场景中也被称为“RTMP 发布”。
- **直播**：直播的视频源是实时生成的，有人推流直播才有意义，一旦主播停播，直播 URL 也就失效了，而且由于是实时直播，所以播放器在播直播视频的时候是没有进度条的。
- **点播**：点播的视频源是云端的一个文件，文件只要没有被提供方删除就随时可以播放（类似腾讯视频），而且由于整个视频都在服务器上，所以播放的时候是有进度条的。

## 云直播播放域名有什么要求？

控制台进行域名提交管理前，需对域名进行备案，域名的位数限制为45位，暂不支持大写的域名，请输入不超过45位的小写域名地址，详情可查看 域名管理 。

## 直播域名接入播放域名和推流域名可以是同一个吗？能使用二级域名吗？

接入播放域名和推流域名必须是不同的两个域名，但可以通过二级域名来进行区分。
例如：`123.abc.com` 用于推流域名，`456.abc.com` 用于播放域名。

## 支持哪些推流协议？

虽然 RTMP 在直播领域不是特别流行，但是在推流服务，也就是“主播”到“服务器”这个方向上 RTMP 居于主导地位，目前国内的视频云服务都是以 RTMP 为主要推流协议（由于直播 SDK 第一个功能模块就是主播推流，所以也被称为是 RTMP SDK）。

## 支持哪些播放协议？

目前常见的直播协议包括：RTMP、FLV、HLS 和 WebRTC。

- **RTMP**：RTMP 协议比较全能，既可以用来推送又可以用来直播，其核心理念是将大块的视频帧和音频帧拆分，然后以小数据包的形式在互联网上进行传输，而且支持加密，因此隐私性相对比较理想，但拆包组包的过程比较复杂，所以在海量并发时也容易出现一些不可预期的稳定性问题。
- **FLV**：FLV 协议由 Adobe 公司主推，格式极其简单，只是在大块的视频帧和音视频头部加入一些标记头信息，由于这种简洁，在延迟表现和大规模并发方面都很成熟，唯一的不足就是在手机浏览器上的支持非常有限，但是用作手机端 App 直播协议却异常合适。
- **HLS**：苹果推出的解决方案，将视频分成5秒 - 10秒的视频小分片，然后用 m3u8 索引表进行管理，由于客户端下载到的视频都是5秒 - 10秒的完整数据，故视频的流畅性很好，但也同样引入了很大的延迟（HLS 的一般延迟在10秒 - 30秒左右）。相比于 FLV，HLS 在 iPhone 和大部分 Android 手机浏览器上的支持非常给力，所以常用于 QQ 和微信朋友圈的 URL 分享。
- **WebRTC**：名称源自网页即时通信（Web Real-Time Communication）的缩写，是一个支持网页浏览器进行实时语音对话或视频对话的API。它于2011年06月01日开源并在 Google、Mozilla、Opera 支持下被纳入万维网联盟的 W3C 推荐标准。快直播正是用的 WebRTC 协议，它是标准直播在超低延迟播放场景下的延伸，比传统直播协议延迟更低，为观众提供毫秒级的直播观看体验。能够满足一些对延迟性能要求更高的特定场景需求，例如在线教育、体育赛事直播、在线答题等。

| 直播协议 | 优点 | 缺点 | 播放延迟 |
| --- | --- | --- | --- |
| FLV | 成熟度高、高并发无压力 | 需集成 SDK 才能播放 | 2s - 3s |
| RTMP | 延迟较低 | 高并发情况下表现不佳 | 1s - 3s |
| HLS(m3u8) | 手机浏览器支持度高 | 延迟非常高 | 10s - 30s |
| WebRTC | 延迟最低 | 需集成 SDK 才能播放 | < 1s |

版权所有：腾讯云计算（北京）有限责任公司

## 播放地址由什么组成?

腾讯云播放地址主要由播放前缀、播放域名（domain）、应用名称（AppName）、流名称（StreamName）、播放协议后缀、鉴权参数以及其他自定义参数组成，如下：

```
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.m3u8?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.flv?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
https://domain/AppName/StreamName.m3u8?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
https://domain/AppName/StreamName.flv?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

- **播放前缀**
  RTMP 播放协议：**rtmp://** 。
  HTTP-FLV 播放协议：**http://** 或者 **https://** 。
  HLS 播放协议：**http://** 或者 **https://** 。
  WebRTC 播放协议：**webrtc://**。
- **应用名称（AppName）**
  应用名称指的是直播流媒体文件存放路径，默认云直播会分配一个路径：**live**。
- **流名称（StreamName）**
  流名称（StreamName）是指每路直播流唯一的标识符。
- **鉴权参数以及其他自定义参数**
  鉴权参数：txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)。

## 常见的推流方式?

- **移动端 Android/iOS，使用摄像头**：使用第三方软件或 直播 SDK 采集摄像头视频，并推送视频流至直播流推流地址。
- **台式机或笔记本，使用摄像头或桌面录屏**：使用第三方软件采集摄像头视频或桌面图像，将视频或桌面内容推流至直播流推流地址。第三方推流软件包括：OBS（推荐）、XSplit、FMLE 等。
- **视频采集设备**：高清摄像机类设备如果具备 HDMI 或者 SDI 输出接口，可以接入编码器，以 RTMP 推流的方式向直播服务推送直播内容，您需要将直播推流地址配置到编码器的 RTMP 发布地址。
  网络摄像头类设备，如果支持 RTMP 推流，则可将直播推流地址配置到摄像头的 RTMP 发布地址。
- **视频文件转视频流**：读取某个视频文件，并以 RTMP 流方式输出作为视频源来向直播服务的 RTMP 推流地址进行视频发布。可以使用 `ffmpeg` 命令来实现（Windows、Linux 及 Mac 均适用）。

## 断流和禁播的区别?

- **断流功能**：一条直播中的流，如果断流，则本次推流会被中断，观众端将无法观看直播。断流后，主播端可以再次发起推流，继续直播活动。
- **禁播功能**：一条直播中的流，如果禁播，则本次推流会被中断，观众端将无法观看直播。断流后，主播端在禁播时间内无法再次发起推流，禁播功能可通过云直播控制台的流管理页面配置，禁用后的直播流展示于禁播流列表页，单击【启用】后可恢复使用。

## 直播 v1 接口升级 v2 接口怎样无缝对接？是否有接口对应关系?

已有 v1 和 v2 接口对应表，具体请参见 直播 SDK V1 升级到 V2 API 接口对比说明 。

## 关于直播 SDK 隐私合规方面会采集哪些信息（如获取 mac 地址，imei 等）?

直播 SDK 隐私合规符合应用市场要求，具体请参见 隐私协议 。

# 推流播放相关问题

最近更新时间：2023-10-11 16:20:23

## 直播的在线人数是否有上限?

腾讯云直播默认不限制观看直播的在线人数，只要网络等条件允许都可以观看直播。如果用户配置了带宽限制，当观看人数过多、超出了限制带宽时新的用户无法观看，此情况下在线人数是有限制的。

## 如何使用播放转码?

考虑到不同的网络因素，满足您使用不同码率不同分辨率的需求，您可以前往 转码配置 设置不同码率不同分辨率的转码模板，更多转码相关信息请参见 直播转封装及转码 。

### 原始、高清、标清场景

在业务播放场景中，一般会用到三个码率：原始、高清、标清。

- 原始流与推流码率分辨率一致。
- 高清流建议使用码率：2000kbps，分辨率：1080p。
- 标清流建议使用码率：1000kbps，分辨率：720p。

## 如何使用时移回看?

若您想回看过去某段时间的精彩内容，您可以使用时移功能，时移功能目前仅支持 HLS 协议。具体时移的相关介绍以及开通方法请参见 直播时移 。

## 如何使用 HTTPS 播放?

若您的播放域名需要支持 HTTPS，您需要准备好有效的证书内容、有效私钥内容前往 域名管理 ，选择**播放域名管理 > 高级配置 > HTTPS配置**添加配置，添加成功之后会有个生效时间（2小时），在生效后您的直播流就可以支持 HTTPS 协议播放。

## 如何使用海外加速节点播放?

云直播 CDN 节点不仅遍布中国大陆区域，同样在全世界各个大洲也有节点覆盖，覆盖广并且稳定。假如您的用户分布在中国香港、中国澳门、中国台湾或海外其他地区，您可以通过在 域名管理 中配置域名时候加速区域选择**全球加速**或**中国港澳台地区及海外地区**，来获得海外节点覆盖的支持。

> **注意**
> 云直播海外加速目前仅支持 HTTP-FLV + HLS 协议。

## 如何开启播放防盗链?

为防止非法用户盗取您的播放 URL 在别处播放，造成流量损失，强烈建议您给播放地址加上播放防盗链，防止因盗链产生不必要的损失。云直播的播放防盗链主要由四个参数值控制：txTime、key（哈希密钥）、txSecret、有效时间。

| 防盗链参数 | 描述 | 补充说明 |
| --- | --- | --- |
| txTime | 播放 URL 的有效时间 | 格式为16进制 UNIX 时间。<br>如果当前 txTime 的值大于当前请求的时间则可以正常播放，否则播放会被后台拒绝。 |
| key | MD5 计算方式的密钥 | 可以自定义，并可以设置主备两个 key。<br>当您的主 key 意外泄露的时候，您可以使用备用 key 进行拼接播放 URL，并同时更改主 key 的值。 |

| | | |
|---|---|---|
| txSecret | 播放 URL 中的加密参数 | 值是通过将 key，StreamName，txTime 依次拼接的字符串进行 MD5 加密算法得出。<br>xSecret = MD5（key+StreamName+txTime）。 |
| 有效时间 | 地址有效时间 | 有效时间设置必须大于0。<br>假设 txTime 设置为当前时间，有效时间设置为300s，则播放 URL 过期时间为当前时间 + 300s。 |

### 防盗链计算

防盗链计算需要三个参数，key（随机字符串）、StreamName（流名称），txTime（16进制格式）。
假设您设置的 key 为 **somestring**，流名称（StreamName）为 **test**，txTime 为**5c2acacc**（2019-01-01 10:05:00）。高清码率为：**900kbps**，转码模板名称为：**900**。

- 原始流播放地址：

```
txSecret = MD5(somestringtest5c2acacc) = b77e812107e1d8b8f247885a46e1bd34
http://domain/live/test.flv?txTime=5c2acacc&txSecret=b77e812107e1d8b8f247885a46e1bd34
http://domain/live/test.m3u8?txTime=5c2acacc&txSecret=b77e812107e1d8b8f247885a46e1bd34
```

- 高清流播放地址：

```
txSecret = MD5(somestringtest_9005c2acacc) = 4beae959b16c77da6a65c7edda1dfefe
http://domain/live/test_900.flv?
txTime=5c2acacc&txSecret=4beae959b16c77da6a65c7edda1dfefe
http://domain/live/test_900.m3u8?
txTime=5c2acacc&txSecret=4beae959b16c77da6a65c7edda1dfefe
```

### 开启播放防盗链

1. 登录进入 域名管理 。
2. 选择播放域名或单击所在行的**管理**，进入域名详情页。
3. 选择**访问控制**，单击**编辑**。
4. 设置**播放鉴权** 为开启，单击**保存**。

> **注意**
> - 播放鉴权的设置成功后需要**30分钟**后生效。
> - HTTP-FLV：正在播放的 URL 在 txTime 过期后依然能正常播放，在 txTime 过期后重新请求播放则会拒绝。
> - HLS：由于 HLS 是短链接，会不断的请求 m3u8 获取最新的 ts 分片。假设您设置 txTime 的值为当前时间 + 10分钟，则在10分钟之后 HLS 播放 URL 请求会被拒绝。针对这个问题您可在业务端动态更新 HLS 的请求地址，或者将 HLS 的播放地址过期时间设置久一点。

## 播放鉴权配置中主 Key 的格式有什么要求？有效时间时长有没有限制？

鉴权配置中主 Key 值仅支持大写字母，小写字母和数字，最大长度256位。字母数字随机组合搭配即可。
有效时间时长建议设置为一场直播的时间长度。

## 直播录制后，如何获取录制文件？

录制文件生成后自动存储到云点播系统，有以下方式可以获取录制文件：

- 云点播控制台

- 录制事件通知
- 点播 API 查询

## Android 直播推流退后台无法推流?

9.0 及之后的系统，应用退后台后 摄像头和麦克风将停止工作，开启前台 Service 可规避此问题，详情参考 解决方案 。

# License 相关问题

最近更新时间：2024-12-06 09:56:02

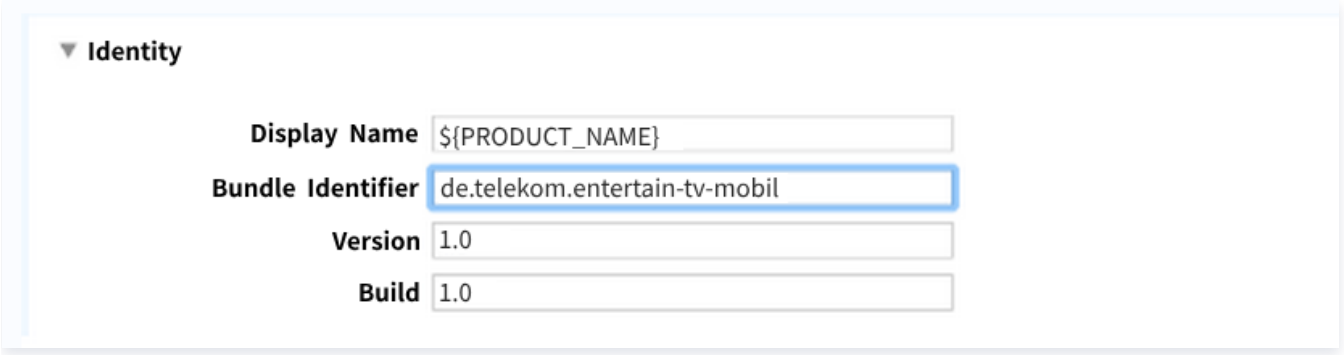
## Android 下如何获取 package name?

您可在 Android 工程下的 `Mainfest.xml` 文件中获取，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.huawei.player"
android:versionCode="20181111"
android:versionName="1.0">
```

## iOS 下如何获取 Bundle ID?

iOS 可在工程配置中的 General > Identity 中获取，如下图所示：




## 测试版 License 到期后是否可以延期?

测试 License 试用期最多28天，初次申请满14天后可延期一次，不支持二次延期，到期后请尽快 购买正式 License 。试用期内申请测试续期，则续期到期时间以申请测试时刻为准；若试用期结束后申请测试续期，则续期到期时间以申请测试续期时刻为准。

- 当申请测试开始时间为 `2022-05-25 11:34:55` ，则14天后到期时间为 `2022-06-09 00:00:00` 。
- 免费续期一次时，若在试用期14天内申请续期，则到期时间为 `2022-06-23 00:00:00` ；若在试用期14天结束后申请续期，申请续期的时间为 `2022-07-03 22:26:20` ，则续期的到期时间为 `2022-07-18 00:00:00` 。

## 测试版 License 能否更改 Android 的 Package Name 和 iOS 的 Bundle ID?

测试版 License 能更改 Android 的 Package Name 和 iOS 的 Bundle ID。具体操作：进入**云直播控制台** > License 管理，单击测试版 License 信息右侧的**编辑**，进入编辑页面即可修改 Android 的 Package Name 和 iOS 的 Bundle ID。

## 正式版 License 能否更改 Android 的 Package Name 和 iOS 的 Bundle ID?

正式版 License 不能更改 Package Name 和 Bundle ID。

## License 可以同时支持多个 App 吗?

一个 License 只能对应一个 Package Name 和一个 Bundle ID，若多个 App 使用 SDK 功能，需要购买多个资源包新增多个 License。

## 腾讯云视立方 · 直播 SDK License 是必须购买的吗?

腾讯云视立方 · 直播 SDK 的直播推流功能，必须通过购买直播 License 进行解锁。

> **注意：**
> 短视频 License 无法解锁直播 SDK 相关功能。

## 腾讯云视立方 · 直播 SDK License 有单独购买入口吗?

**可购买独立直播 License 授权解锁功能模块**，详情请参见 独立 License 价格 。
直播 License（原移动直播基础版 SDK License）需购买云直播10TB及以上 流量资源包 获取一年使用授权。

## 直播 License（原移动直播基础版 SDK License）提供美颜功能吗?

直播 License（原移动直播基础版 SDK License）即可解锁直播推流和播放功能，以及基础的美颜功能（美白、磨皮等）。
高级美颜（大眼、瘦脸）、背景虚化、动效贴纸等增值能力由 腾讯云视立方 · 腾讯特效 SDK 提供。

> **说明：**
> SDK 下载中的3个版本的 SDK 均可用直播 License（原移动直播基础版 SDK License）来解锁直播推流和播放功能。

## 一个账号下能创建多个 License 吗?

同一个账号下创建 License 的数量没有限制。为了方便用户管理，相同包名的 License 建议通过续期的方式延长有效时间。

## 相同包名可以创建多个 License 吗?

可以，多个 License 填写相同的包名不会影响使用，不过同时创建多个 License 有效期会各自计算，一般不建议创建多个相同包名的 License。

## License 可以修改吗?

腾讯云视立方 · 直播 SDK License 可通过续期来延长有效时间，包名信息不支持修改，请您在添加 License 先核对包名在应用商店里是否被占用，提交后不支持修改和替换。

## 为什么我创建了多个 License ，但是 licenseurl 和 key 都是一样的?

同一个账号下腾讯云视立方 · 直播 SDK 的 licenseurl 和 key 默认是相同的。这样保证了测试 License 、正式 License、不同包名的 License 均可以复用相同的接口信息。

> **说明：**
> 不建议使用免费测试的 License 发布到线上运行。您可以通过添加新的正式版 License，即可无需再修改接口中的 licenseurl 和 key，切换到正式版 License。

## 关联 License 的资源包是不是只能这个 License 使用?

该账号下的标准直播播放域名产生的日结流量后付费消耗均可抵扣。资源包关联只是用于同步有效期，里面的流量不限于 License 使用（流量用尽也不影响 License 的使用）。**例如：** 用户甲是日结流量后付费计费，购买了一个10TB标准直播流量包和50TB标准直播流量包，分别创建了 License A 和 License B：

- License A 对应的 App 使用的是 `abc.com` 域名播放，产生了20TB的播放流量。
- License B 对应的 App 使用的是 `def.com` 域名播放，产生了30TB的播放流量。

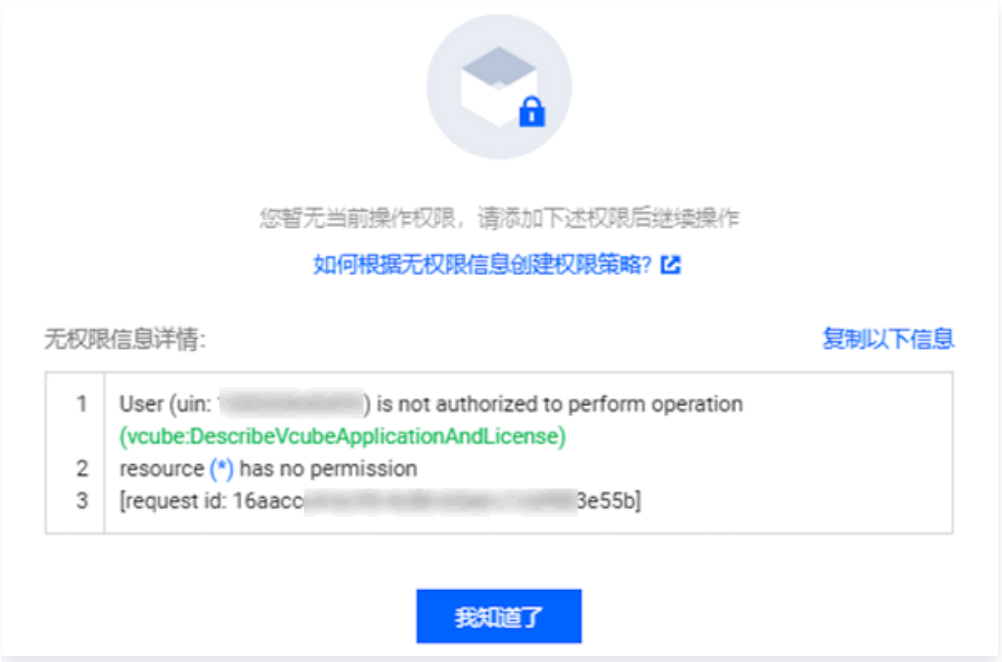
只要 `abc.com` 和 `def.com` 这两个是属于用户甲云直播账号下标准直播的播放域名，即可使用购买的10TB + 50TB来抵扣，抵扣后用户甲的标准直播流量包剩余10TB流量。

## 购买腾讯云视立方 · 直播 SDK License 可以用于小程序直播吗?

不支持，直播 SDK License 仅支持 iOS 和 Android 端的 App 在使用直播 SDK 直播功能时使用。小程序端接入直播功能需要先具备对应的服务类目，详情参见 方案选择 。

## 为什么我的子账户已经授权了直播和点播所有权限，但是还是无法访问 License 控制台相关界面?

**问题截图：**




**问题解析：**

新版 SDK License 本次升级更新了接口（详情请参见 新旧 License 说明 ），需要主账号为子账号独立进行重新授权策略后方可访问 License 控制台页面。

- 若您仅需要提供子账号查询 License 的权限，请授权 QcloudVCUBEReadOnlyAccess 策略。
- 若您需要提供子账号所有 License 操作权限，请授权 QcloudVCUBEFullAccess 策略。

为用户/用户组关联策略以授权相关操作权限的关联指引请参见 策略授权管理 。

> **说明：**
> License 界面所有功能操作已独立于云直播、云点播策略外，即原 QcloudVODFullAccess、QcloudLIVEFullAccess 策略已不包含 License 相关接口，需按照上述说明单独授权。

## 为什么接收不到 License 到期等相关消息通知?

腾讯云视立方 · 直播 SDK 的直播 License 用于音视频终端 SDK（腾讯云视立方）管理功能模块的授权解锁，您可以通过在 消息订阅 中订阅音视频终端 SDK，配置**站内信/邮件/短信/微信/企微**等消息接收渠道，接收正式版 License 到期提醒。直播正式版 License 将在到期时间距离当前时间为30天、15天、7天、1天时各向您发送一次到期提醒，提示您及时续费以免影响正常业务运行。

## 为什么我刚续期了 License，控制台 License 已显示在有效期内，但终端应用提示"许可证无效"?

可能因为受本地缓存影响，License 未能自动更新。请您按以下步骤操作：

1. 确保设备已连接网络。
2. 重启应用程序。
3. 等待许可证自动更新完成。

同时建议您在 License 到期前，提前续期 License，以免影响您的业务。

# 降低延迟相关问题

最近更新时间：2022-10-08 16:33:46

正常情况下，使用 RTMP 协议推流并通过 FLV 协议播放，延迟在2秒 - 3秒左右，如果太长一般是有问题的。如果您发现直播延迟时间特别长，可以按照如下思路来排查。

## Step 1. 检查播放协议

如果您的播放协议采用的是 HLS（m3u8）协议，并感觉延迟较大，这个是正常的。 HLS 协议是苹果主推的基于大颗粒的 TS 分片的流媒体协议，每个分片的时长通常在5秒以上，分片数量一般为3个 - 4个，所以总延迟在10秒 - 30秒左右。
如果您必须要使用 HLS（m3u8）协议，只能通过适当减少分片个数或者缩短每个分片的时长来降低延迟，但需要综合考虑对卡顿指标可能造成的影响，目前您可以通过 提交工单 或者联系腾讯云技术支持工程师进行调整。

## Step 2. 检查播放器设置

腾讯云直播 SDK 的播放器支持极速、流畅和自动三种模式，具体设置请参见 延时调节：

- **极速模式：**能保证绝大多数场景下延迟都在2秒 - 3秒以内，美女秀场适合这个模式。
- **流畅模式：**绝大多数场景下延迟都在5秒以内，适合对延迟不敏感但对流畅度要求高的场景，例如游戏直播。


## Step 3. 尽量在客户端打水印

腾讯云直播支持在云端打水印，但是打水印会引入额外的1秒 - 2秒的延迟，所以如果您使用的是腾讯云直播 SDK，可以选择直接在主播端 App 打上水印，这样就不需要在云端来打，从而减少水印造成的延迟。

## Step 4. 使用第三方推流器

我们只能确保在腾讯云一体化解决方案中保持理想的效果，如果您使用的是第三方推流软件，建议您使用腾讯云直播 SDK 的 推流 Demo 做个对比，排除一下第三方推流器的编码缓存引入大延迟的可能，因为很多第三方的推流器会暴力地采用无限缓冲的方式来解决上行带宽不足的问题。

## Step 5. 检查 OBS 设置

如果您采用的是 OBS 推流，并且发现在播放端延迟比较大。建议您按照 OBS 推流 中的描述配置对应的参数，并注意把关键帧间隔设置为1秒或2秒。

## Step 6. 接入快直播

如果以上建议都不能满足您对延迟的要求，您可以接入腾讯云快直播，快直播比标准直播延迟更低，可以提供毫秒级的极致直播观看体验。具体请参见 快直播 文档介绍。

# 推流失败相关问题

最近更新时间：2025-02-12 16:21:23

如果您按照 最佳实践 - 直播推流 中的范例来操作，发现仍推流不成功。可以依照本文档中罗列的视频推流过程中的常见问题，按照下列思路依次排查。

## 1. 域名是否 CNAME 到了腾讯云地址?

推流域名只有 CNAME 到腾讯云地址才能推流成功，可以在 域名管理 里面查看已经创建的推流域名是否有 CNAME。其中有个 CNAME 标题栏，可以根据此项中的状态来查看推流域名是否有 CNAME。已经 CNAME 的状态如下：


如果还没 CNAME，可以根据 CNAME 配置 来配置。

## 2. 网络是否正常?

RTMP 推流默认使用的端口号是1935。如果您在测试时发现无法连接服务器，可能是因为所在网络的防火墙不允许1935端口通行。此时，您可以尝试切换网络（例如改用4G），以排查是否是此原因导致的问题。

## 3. txTime 是否过期?

有些客户担心自己的直播流量被人盗用，会将 txTime 设置得过于保守，例如从当前时间开始往后推5分钟。其实由于有 txSecret 签名的存在，txTime 的有效期不用设置得太短。相反，如果有效期设置得太短，当主播在直播过程中遭遇网络闪断时会因为推流 URL 过期而无法恢复推流。
txTime 建议设置为当前时间往后推12或者24小时为宜，也就是要长于一场普通直播的直播时间。

## 4. txSecret 是否正确?

腾讯云目前要求推流地址都要加防盗链以确保安全，防盗链计算错误或者已经过了有效期的推流 URL，都会被腾讯云**踢掉**，这种情况下直播 SDK 会抛出 **PUSH_WARNING_SERVER_DISCONNECT** 事件， 直播 SDK DEMO 此时的表现如下：


阅读 功能实践 – 直播推流 了解如何获取可靠的推流 URL。

## 5. 推流 URL 是否被占用?

一个推流 URL 同时只能有一个推流端，第二个尝试去推流的 Client 会被腾讯云拒绝掉。此种情况可以登录直播控制台，在 流管理 的**在线流**中查看此条流是否已经在推，也可以在**禁推流**中查看该条流是否被禁推。

## 6. 使用 V2TXLivePusher 调用 startPush 推流返回-2错误?

目前有以下几个场景会报错-2：

- 使用 LiteAVSDK_Smart 版本 V2TXLivePusher 推流 `trtc://` 协议，因为 smart 版本不支持 TRTC 协议，需要全功能版才支持。
- 调用 startPush 推流传的推流地址缺少必要参数，请参见 推拉流 URL 拼接正确的流地址。
- 播放器初始化模式选择的是 V2TXLiveMode_RTC，但是传的 URL 是 `rtmp://` 协议地址。

# 播放失败相关问题

最近更新时间：2025-05-26 10:54:52

如果您发现直播无法观看，完全搞不懂里面出了什么情况，按照下面的思路进行排查，一般都能在几十秒内确认问题原因。


## 1. 检查播放 URL

在所有检查开始之前，您务必要先检查一下地址是否正确，因为这里出错概率最高，腾讯云的直播地址分推流地址和播放地址两种，我们要首先排除**误拿推流地址来播放**的错误。

```
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.m3u8?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.flv?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
https://domain/AppName/StreamName.m3u8?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
https://domain/AppName/StreamName.flv?
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

**说明**

domain 为推流/播放域名，AppName 自定义、默认为 live，StreamName 自定义；若未开启推流或播放鉴权，则无 “?” 及其后的 txSecret 内容。例如，推流域名为 `www.push.com`，AppName 为 live，StreamName 为 test01，未开启推流鉴权，则推流地址为 `rtmp://www.push.com/live/test01` 。

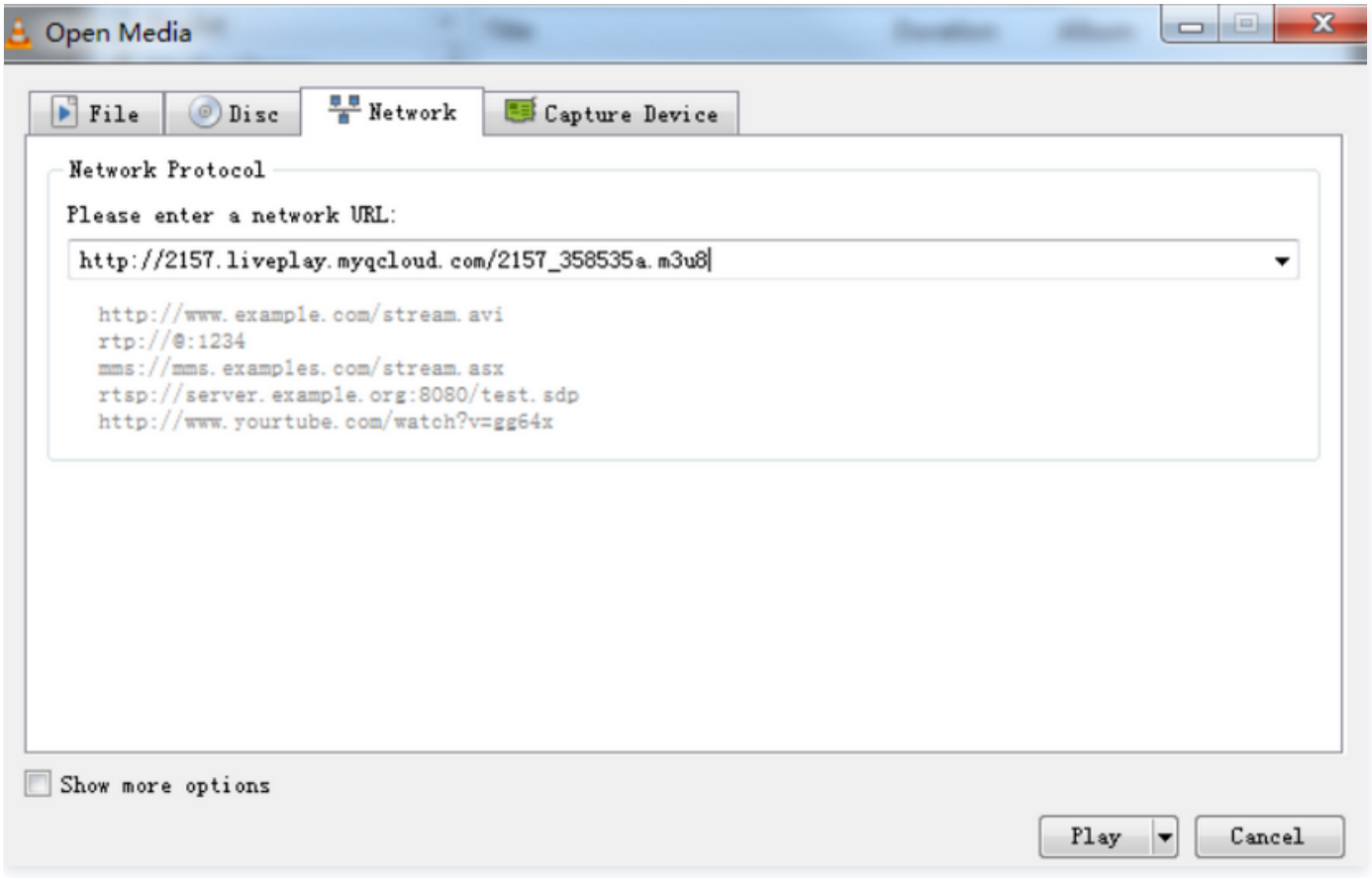
小直播的播放 URL 可以用调试的办法获取，您可以全局搜索代码寻找关键字 **startLivePlay**，然后在此处打下调试断点，这里是小直播对 RTMP SDK 的调用点，startLivePlay 的参数即为播放 URL。

## 2. 检查视频流

播放 URL 正确不代表视频就能播放，所以要检查视频流是否正常：

- 对于**直播**，如果主播已经结束推流，直播 URL 就不能观看。
- 对于**点播**，如果云端的视频文件已经被移除，同样也是不能观看。

常用的解决办法就是用 VLC 检查一下，VLC 是 PC 上的一款开源播放器，支持的协议很多，所以最适合用来做检查。对于 WebRTC 协议，因为是腾讯私有协议，您可以使用 快直播 Demo 验证。




## 3. 检查播放端

如果视频流非常健康，我们就要分情况检查一下播放器是否正常：

- **Web 浏览器**
  - **格式支持**：手机浏览器**只支持 HLS（m3u8）和 MP4** 格式的播放地址。
  - **HLS（m3u8）**：腾讯云 HLS 协议是懒启动的，简言之，只有当有观众请求 HLS 格式的观看地址后，腾讯云才会启动 HLS 格式的转码，这种懒启动策略的目的是规避资源浪费。但也就产生一个问题：**HLS 格式的播放地址要在全球首个用户发起请求后10秒才能观看**。
  - **腾讯云 Web 播放器**：支持同时指定多种协议的播放地址，能够根据所在的平台（PC/Android/iOS）采用最佳的播放策略，同时内部的选择性重试逻辑也能针对性解决 HLS（m3u8）懒启动的问题。
- **RTMP SDK**
  如果 RTMP SDK DEMO 本身播放没有问题，推荐您参考 RTMP SDK 的播放文档（iOS 和 Android）检查一下对接逻辑是否错误。
- **移动端调用 startLivePlay 返回 -5**
  在调用 `startLivePlay` 前，需要通过 `V2TXLivePremier#setLicence` 或者 `TXLiveBase#setLicence` 设置 License 后方可成功播放，否则将播放失败（黑屏），全局仅设置一次即可。直播 License、短视频 License 和播放器 License 均可使用，若您暂未获取上述 License，可 快速免费申请测试版 License 以正常播放，正式版 License 需 购买。
- **onError 抛出 V2TXLIVE_ERROR_NO_AVAILABLE_HEVC_DECODERS(-2304)**
  当抛出此错误码时，说明当前设备不支持 H265 解码，请切换至 H264 数据流进行播放。
- **调用 `setAudioRoute` 设置音频路由不生效**
  1.1 音频路由必须在硬件设备打开后才会生效，所以请在 `onCaptureFirstAudioFrame` 回调收到后再设置；
  1.2 确认是否添加 `<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />` 权限。

- Android 硬解黑屏
  检查一下 `manifest` 文件中 `android:hardwareAccelerated` 属性，确保其值为 true。

## 4. 检查防火墙拦截

这是常见的一种情况，不少客户的公司网络环境会限制视频播放，限制的原理是由防火墙侦测 HTTP 请求的是否是流媒体资源。如果您使用 4G 进行直播观看没有问题，而用公司的 Wi-Fi 网络无法观看，即说明公司的网络策略有所限制，您可以尝试跟网管沟通，让网管给您的 IP 做一下特殊处理。

## 5. 检查推流端

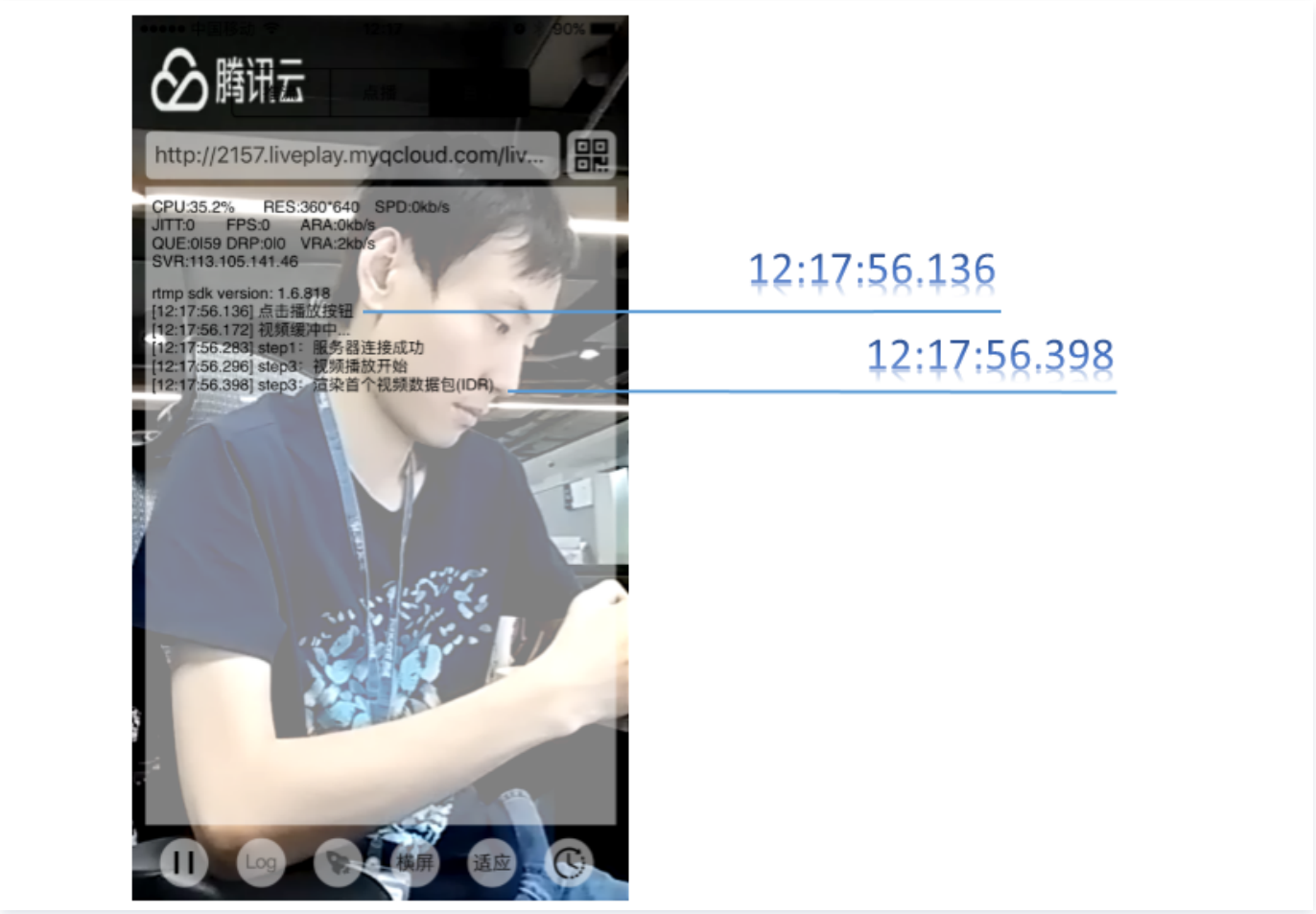
如果是直播 URL 根本不能播放，而且没有步骤4中防火墙限制的可能，那么很大概率是推流不成功，可以到 推流失败问题排查 继续问题的排查。

# 如何实现秒开

最近更新时间：2022-10-08 16:33:47

## 什么叫做“秒开”？

**秒开**即从视频播放开始到真正看到第一帧画面所消耗的时间要尽可能的短（几百毫秒时间），不能让观众有明显的等待时间。




这种能力主要依靠云端服务的优化以及播放器的配合，如果您组合使用腾讯云视立方 · 直播 SDK，配合视频云服务实现直播能力，可以实现200ms左右的首屏打开速度，如果网络下行足够好的话甚至可以更快。

## 如何实现“秒开”？

### App 端

使用 直播 SDK + FLV 播放协议即可实现秒开：

- **HTTP + FLV 播放协议**
  HTTP + FLV 协议是目前直播行业使用最普遍的播放协议，它的数据组织格式比较简单，可以做到一旦连通服务器就能获取到音视频数据。相比之下，RTMP 协议由于连接初期不可避免的几次协商握手过程，导致在首帧速度方面略逊于 FLV 协议。
- **腾讯云视立方 · 直播 SDK**
  秒开的云端实现原理其实非常简单，服务器始终缓存一组 GOP 画面（至少包含一个可以用于解码的关键帧），这样播放器一旦连通服务器就可以获取到一帧关键帧（I 帧），进而可以解码和播放，但这种云端的缓存也会带来负作用：播放器在连通服务器后，通常会一口气被塞过来几秒钟的音视频数据，从而产生不小的播放端延迟，我们称之为 “秒开后遗症”。
  一款好的播放器，除了具备秒开能力，还要具备优秀的**延迟修正**能力，能够在无损观看体验的情况下，自动修正播放端延迟到一个合理的

范围内（例如1秒以内），而腾讯云视立方 · 直播 SDK 在这方面就做的非常优秀，您甚至可以指定播放器的延迟修正模式（ iOS & Android ）。

## PC 浏览器

PC 浏览器的视频播放内核一般都是采用 Flash 控件（目前 Chrome 也支持 MSE，但并不比 Flash 有明显优势），Flash 播放器策略是比较刚性的**强制缓冲**模式，所以视频打开速度没有什么优化空间，一般很难做到1秒以内，这一点可以通过各大视频网站和直播平台的 PC 端表现就能发现。

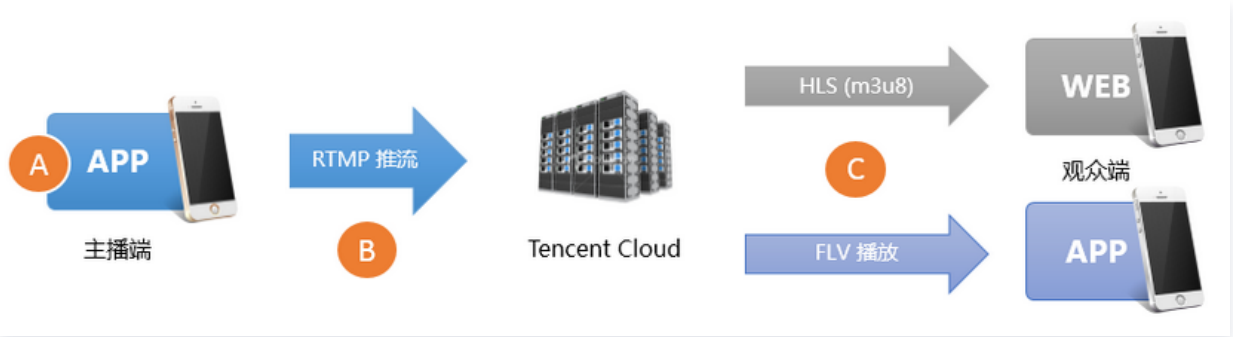
## 手机浏览器

- **iPhone**
  Safari 对 HLS（m3u8）的支持很好，甚至直接使用 iPhone 的硬解芯片协助视频播放，在具备 DNS 缓存的情况下，视频打开速度通常都有保障，但这也仅限于 iOS 平台。
- **Android**
  Android 上的表现就具有比较大的随机性，由于碎片化严重，各个版本和机型的系统浏览器实现都有差异，QQ 和微信内的浏览器甚至采用了腾讯自己的 X5 内核，所以具体表现会有比较大的差异。

版权所有：腾讯云计算（北京）有限责任公司

# 如何优化视频卡顿（V1）

最近更新时间：2023-06-16 11:33:23




造成播放端卡顿的原因主要有三种：

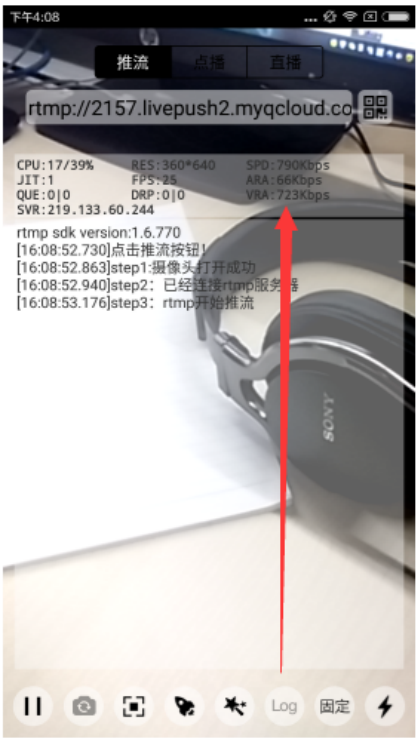
- **原因1：推流帧率太低**
  如果主播端手机性能较差，或者有很占 CPU 的后台程序在运行，可能导致视频的帧率太低。正常情况下 FPS 达到每秒15帧以上的视频流才能保证观看的流畅度，如果 FPS 低于10帧，可以判定为**帧率太低**，这会导致**全部观众**的观看体验都很卡顿。当然如果主播端画面本身变化就很少，如静态画面或 PPT 播放等场景，则不受该原因影响。
- **原因2：上传阻塞**
  主播的手机在推流时会源源不断地产生音视频数据，但如果手机的上传网速太小，那么产生的音视频数据都会被堆积在主播的手机里传不出去，上传阻塞会导致**全部观众**的观看体验都很卡顿。

**国内运营商**提供的宽带上网套餐中，下载网速虽然已经达到了10Mbps、20Mbps甚至是100Mbps、200Mbps，但上传网速却还一直限制的比较小，很多小城市的上行网速最快是512Kbps（也就是每秒最多上传64KB的数据）。
**Wi-Fi 上网**遵循 IEEE 802.11 规定的载波多路侦听和冲突避免标准，简言之就是一个 Wi-Fi 热点同时只能跟一个手机通讯，其它手机在跟热点通讯前都要先探测或询问自己是否能够通讯，所以一个 Wi-Fi 热点使用的人越多就越慢。同时 Wi-Fi 信号受建筑墙体的屏蔽干扰很严重，而一般的中国普通家庭很少在装修时考虑好 Wi-Fi 路由器和各个房间的信号衰减问题，可能主播本人也不清楚自己做直播的房间离家里的路由器究竟穿了几堵墙。

- **原因3：下行不佳**
  即观众的下载带宽跟不上或者网络波动较大，例如直播流的码率是2Mbps的，也就是每秒钟有2M比特的数据流要下载下来，但如果观众端的带宽不够，就会导致观众端播放体验非常卡顿。 下行不佳只会影响当前网络环境下的观众。

## 查看 SDK 状态提示信息

如果您使用的是腾讯云视立方 · 直播 SDK 来推流，该 SDK 提供了一种状态反馈机制，每隔1秒 – 2秒就会将内部各种状态参数反馈出来，您可以通过注册 TXLivePushListener 监听器来获取这些状态。相关状态的说明如下：




RTMP SDK 状态参数表

| 推流状态 | 参数名称 | 含义说明 |
| --- | --- | --- |
| CPU | CPU使用率 | App: 17% Sys: 39% |
| RES | 推流分辨率 | 360 * 640 |
| SPD | 网络上行速度 | 790kbps |
| JIT | 网络抖动 | 不推荐参考 |
| FPS | 视频帧率 | 25帧/秒 |
| ARA | 音频码率 | 66kbps |
| QUE | 缓冲积压 | 0帧 |
| DRP | 主动丢包 | 推流以来尚未丢过 |
| VRA | 视频码率 | 723kpbs |

| 推流状态 | 含义说明 |
| --- | --- |
| NET_STATUS_CPU_USAGE | 当前进程的 CPU 使用率和本机总体的 CPU 使用率。 |
| NET_STATUS_VIDEO_FPS | 当前视频帧率，也就是视频编码器每条生产了多少帧画面。 |
| NET_STATUS_NET_SPEED | 当前的发送速度，单位：kbps。 |
| NET_STATUS_VIDEO_BITRATE | 当前视频编码器输出的比特率，也就是编码器每秒生产了多少视频数据，单位： kbps。 |
| NET_STATUS_AUDIO_BITRATE | 当前音频编码器输出的比特率，也就是编码器每秒生产了多少音频数据，单位：kbps。 |
| NET_STATUS_CACHE_SIZE | 音视频数据堆积情况，这个数字超过个位数，即说明当前上行带宽不足以消费掉已经生产的音视频数据。 |
| NET_STATUS_CODEC_DROP_CNT | 全局丢包次数，为了避免延迟持续恶性堆积，SDK 在数据积压超过警戒线以后会主动丢包，丢包次数越多，说明网络问题越严重。 |
| NET_STATUS_SERVER_IP | 连接的推流服务器的 IP，一般应该是离客户端跳数比较少的就近服务器。 |

## 解决帧率太低问题

### 1. 帧率太低的评判

通过直播 SDK 的 TXLivePushListener 的 **VIDEO_FPS** 的状态数据，我们可以获得当前推流的视频帧率。正常来说每秒15帧以上的视频流才能保证观看的流畅度，常规推流如果 FPS 在10帧以下，观众就会明显的感到画面卡顿。

### 2. 针对性优化方案

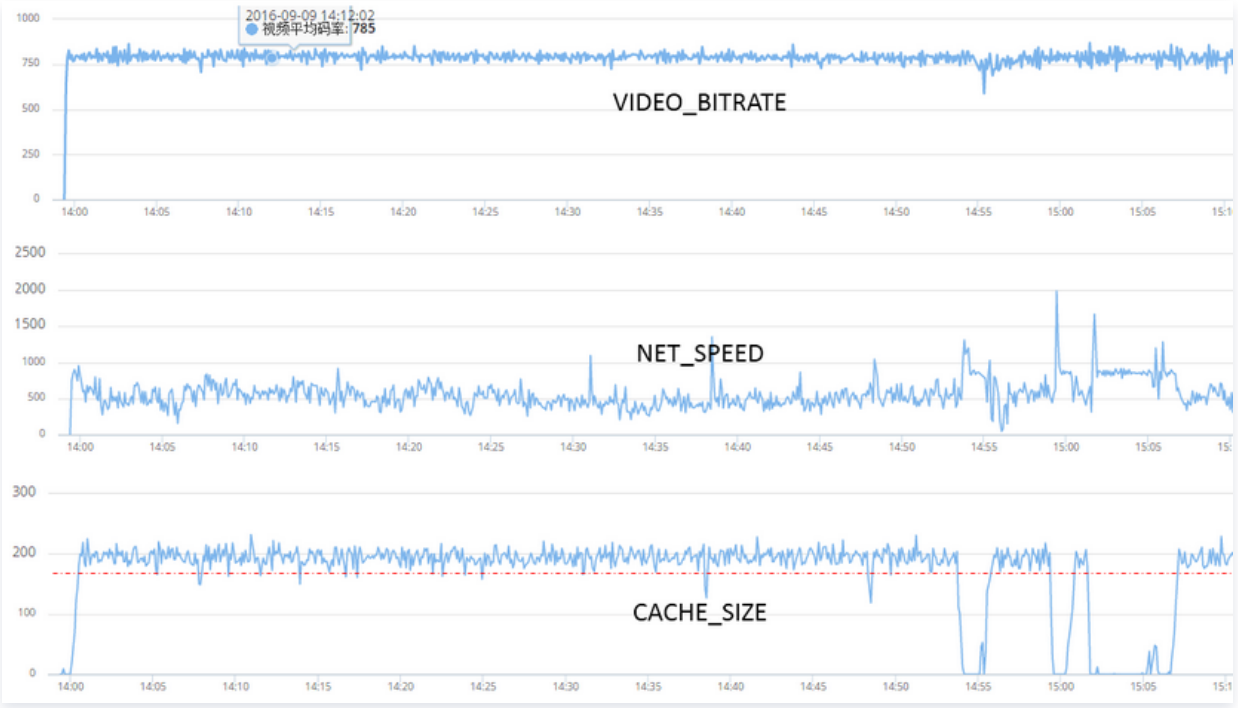
- **2.1 观察 CPU_USAGE 的大小**
  通过直播 SDK 的 TXLivePushListener 的 **CPU_USAGE** 的状态数据，我们可以获得**当前推流 SDK 的 CPU 占用情况**和**当前系统的 CPU 占用情况**。如果当前系统的整体 CPU 使用率超过80%，那么视频的采集和编码都会受到影响，无法正常发挥作用；如果 CPU 使用率达到100%，那么主播端本身就已经很卡，观众端要有流畅的观看体验显然是不可能的。
- **2.2 确认谁在消耗 CPU**
  一款直播 App 中使用 CPU 的不可能只有推流 SDK，弹幕、飘星、文本消息互动等都有可能会消耗一定的 CPU，这些都是不可避免的。如果单纯要测试推流 SDK 的 CPU 占用情况，可以使用我们的 精简版 DEMO 来观察和评估。
- **2.3 不盲目追高分辨率**
  过高的视频分辨率并不一定能带来清晰的画质：首先，较高的分辨率要配合较高的码率才能发挥效果，低码率高分辨的清晰度很多时候比不上高码率低分辨率。其次，像1280 x 720这样的分辨率在平均5寸左右的手机屏幕上并不能看出优势，要想跟960 x 540的分辨率拉开差距，只有在 PC 上全屏观看才能有明显的感官差异。但较高的分辨率会显著提升 SDK 的 CPU 使用率，因此常规情况下推荐使用直播 SDK 中 TXLivePusher 的 setVideoQuality 设置**高清**档即可，盲目追高分辨率有可能达不到预期的目标。
- **2.4 适当使用硬件加速**
  现在的智能手机都支持硬件编码来降低视频编码对 CPU 的依赖，如果您发现您的 App 的 CPU 使用率过高，可以开启硬件编码来降低 CPU 使用率。TXLivePusher 的 setVideoQuality 的**高清档**默认使用的是软件编码（硬件编码在部分 Android 手机上的编码效果不佳，马赛克感很强），如果要使用硬件编码，可以使用 TXLivePushConfig 的 enableHWAcceleration 选项开启。

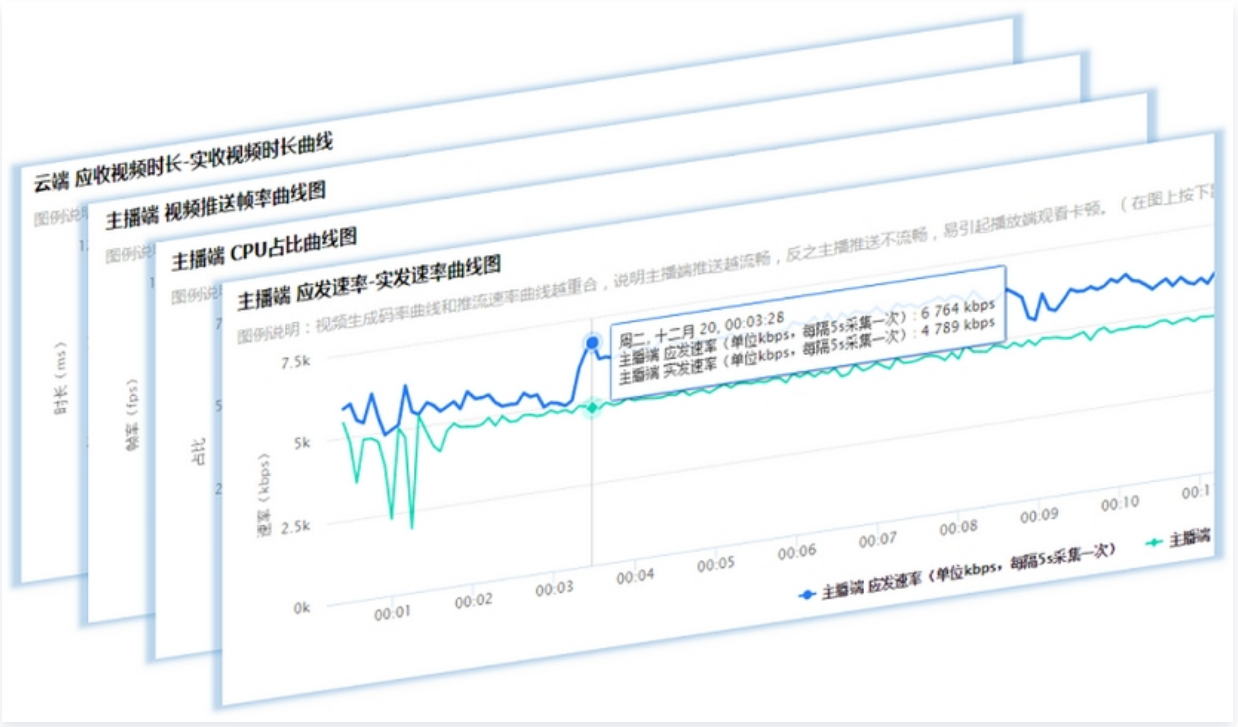
## 解决上传阻塞问题

据统计，视频云客户群80%以上的直播间卡顿问题，均是由于主播端上传阻塞所致。

### 1. 上传阻塞的评判

- **1.1：BITRATE 与 NET_SPEED 的关系**
  BITRATE( = VIDEO_BITRATE + AUDIO_BITRATE ) 指的是编码器每秒产生了多少音视频数据要推出去，NET_SPEED 指的是每秒钟实际推出了多少数据，所以如果 BITRATE == NET_SPEED 的情况是常态，则推流质量会非常良好；而如果 BITRATE >= NET_SPEED 这种情况的持续时间比较长，推流质量就很难有什么保障。
- **1.2：CACHE_SIZE 和 DROP_CNT 的数值**
  BITRATE >= NET_SPEED 的情况一旦出现，编码器产生的音视频数据就会在主播的手机上积压起来，积压的严重程度以 CACHE_SIZE 这个状态值展示出来，如果 CACHE_SIZE 超过警戒线，SDK 会主动丢弃一些音视频数据，从而触发 DROP_CNT 的增长。下图所示就是一个典型的上行阻塞，途中 CACHE_SIZE 始终在**红色警戒线**以上，说明上行网络不足以满足数据的传输需求，也就是上行阻塞严重：




**说明**

您可以在 **直播控制台** > **运营分析** 里看到类似上图的图表。

## 2. 针对性优化方案

- **2.1 主动提示主播**

对于注重清晰度的场景下，通过合适的 UI 交互提示主播**“当前网络质量很糟糕，建议您拉近离路由器的距离，避免 Wi-Fi 穿墙”**是最好的选择。

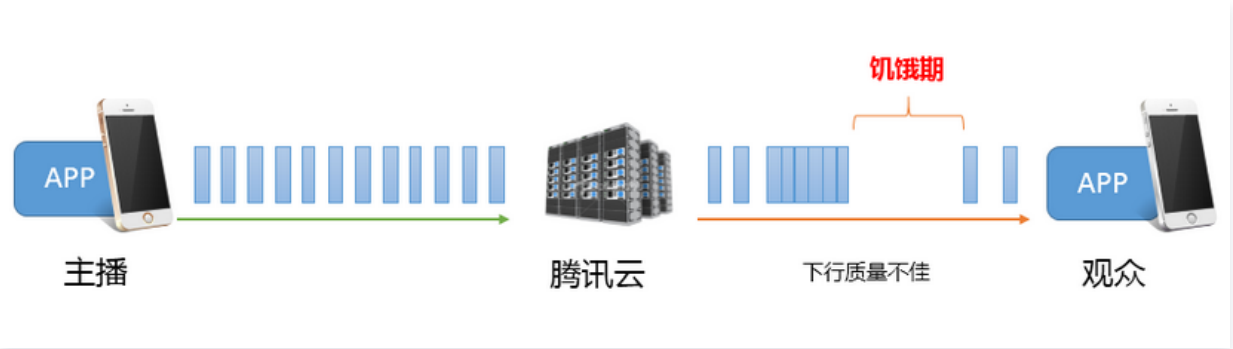
直播 SDK 的推流功能文档中有涉及**事件处理**的介绍，您可以利用它来做到这一点，推荐的做法是：如果 App 在短时间内连续收到直播

SDK 的多个 PUSH_WARNING_NET_BUSY 事件，则提示主播网络关注一下当前网络质量，因为对于上行阻塞这种情况而言，主播本人是没办法通过视频的表现感知到的，只能通过观众的提醒或者 App 的提醒来了解。

- **2.2 合理的编码设置**
  如下是我们推荐的编码设置（适合美女秀场，更多信息请参见 如何实现更好的画质），可以通过 TXLivePusher 里的 setVideoQuality 接口进行相应档位的设置：

| 档位 | 分辨率 | FPS | 码率 | 使用场景 |
| --- | --- | --- | --- | --- |
| 标清 | 360 * 640 | 15 | 400kbps – 800kbps | 如果您比较关注带宽成本，推荐选择该档位，画质会偏模糊，但带宽费用较高清档要低 60%。 |
| 高清（推荐） | 540 * 960 | 15 | 1200kbps | 如果您比较关注画质，推荐选择该档位，能确保绝大多数主流手机都能推出很清晰的画面。 |
| 超清 | 720 * 1280 | 15 | 1800kbps | 慎用：如果您的场景多是小屏观看不推荐使用，如果是大屏幕观看且主播网络质量很好可以考虑。 |

## 优化播放端




### 1. 卡顿与延迟

如上图，下行网络的波动或者下行带宽不够，都会导致在播放过程中出现一段段的**饥饿期**（App 这段时间内拿不到可以播放的音视频数据）。如果想要让观看端的视频卡顿尽量少，就要尽可能地让 App 缓存足够多的视频数据，以保证它能平安度过这些“饥饿期”，但是 App 缓存太多的音视频数据会引入一个新的问题，即**高延迟**，这对互动性要求高的场景是很坏的消息，同时如果不做延迟修正和控制，卡顿引起的延迟会有**累积效应**，就是播放时间越久，延迟越高，延迟修正做得好不好是衡量一款播放器是否足够优秀的关键指标。所以**延迟和流畅是一架天平的两端**，如果过分强调低延迟，就会导致轻微的网络波动即产生明显的播放端卡顿。反之，如果过分强调流畅，就意味着引入大量的延迟（典型的案例就是 HLS（m3u8）通过引入20秒 – 30秒的延迟来实现流畅的播放体验）。

### 2. 针对性优化方案

为了能够让您无需了解过多流控处理知识就能优化出较好的播放体验，腾讯云视立方 · 直播 SDK 经过多个版本的改进，优化出一套自动调节技术，并在其基础上推出了三种比较优秀的 延迟控制方案：

- **自动模式**：如果您不太确定您的主要场景是什么，可以直接选择这个模式。

> **说明**
> 把 TXLivePlayConfig 中的 setAutoAdjustCache 开关打开，即为自动模式。在该模式下播放器会根据当前网络情况，对延迟进行自动调节（默认情况下播放器会在1秒 – 5秒这个区间内自动调节延迟大小，您可以通过 setMinCacheTime 和 setMaxCacheTime 对默认值进行修改），以保证在足够流畅的情况下尽量降低观众跟主播端的延迟，确保良好的互动体验。

- **极速模式**：主要适用于**秀场直播**等互动性高，并且对延迟要求比较苛刻的场景。

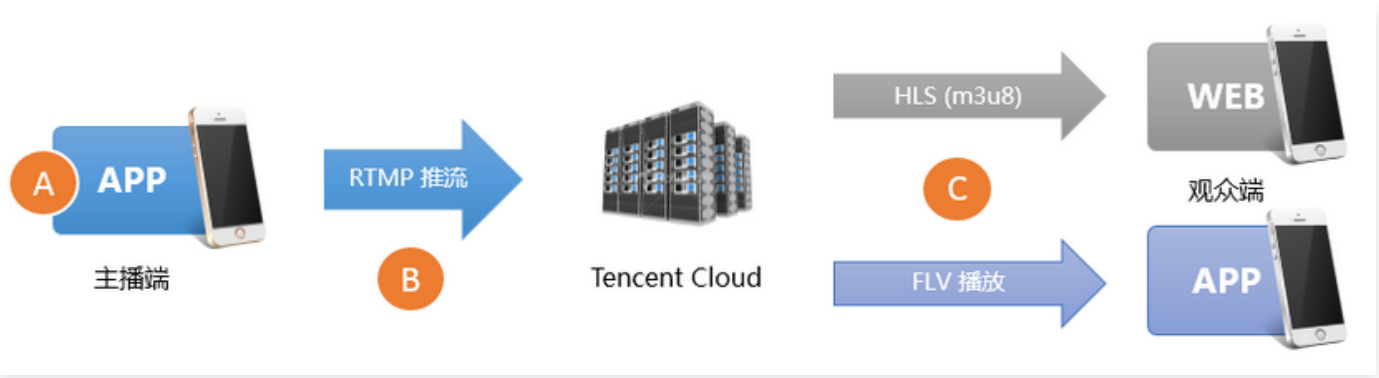
> **说明**
> 极速模式设置方法是 **setMinCacheTime = setMaxCacheTime = 1s** ，自动模式跟极速模式的差异只是 MaxCacheTime 有所不同（极速模式的 MaxCacheTime 一般比较低，而自动模式的 MaxCacheTime 则相对较高），这种灵活性主要得益于 SDK 内部的自动调控技术，可以在不引入卡顿的情况下自动修正延时大小，而 MaxCacheTime 反应的就是调节速度：MaxCacheTime 的值越大，调控速度会越发保守，卡顿概率就会越低。

- **流畅模式**：主要适用于**游戏直播**等大码率高清直播场景。

> **说明**
> - 当把播放器中的 setAutoAdjustCache 开关关闭，即为流畅模式，在该模式下播放器采取的处理策略跟 Adobe Flash 内核的缓存策略如出一辙：当视频出现卡顿后，会进入 loading 状态直到缓冲区蓄满，之后进入 playing 状态，直到下一次遭遇无法抵御的网络波动。默认情况下缓冲大小为5秒，您可以通过 setCacheTime 进行更改。
> - 在延迟要求不高的场景下，这种看似简单的模式会更加可靠，因为该模式本质上就是通过牺牲一点延迟来降低卡顿率。

# 如何优化视频卡顿（V2）

最近更新时间：2025-06-17 18:11:22




造成播放端卡顿的原因主要有三种：

- **原因1：推流帧率太低**
  如果主播端手机性能较差，或者有很占 CPU 的后台程序在运行，可能导致视频的帧率太低。正常情况下 FPS 达到每秒15帧以上的视频流才能保证观看的流畅度，如果 FPS 低于10帧，可以判定为**帧率太低**，这会导致**全部观众**的观看体验都很卡顿。当然如果主播端画面本身变化就很少，如静态画面或 PPT 播放等场景，则不受该原因影响。
- **原因2：上传阻塞**
  主播的手机在推流时会源源不断地产生音视频数据，但如果手机的上传网速太小，那么产生的音视频数据都会被堆积在主播的手机里传不出去，上传阻塞会导致**全部观众**的观看体验都很卡顿。

**国内运营商**提供的宽带上网套餐中，下载网速虽然已经达到了10Mbps、20Mbps甚至是100Mbps、200Mbps，但上传网速却还一直限制的比较小，很多小城市的上行网速最快是512Kbps（也就是每秒最多上传64KB的数据）。
**Wi-Fi 上网**遵循 IEEE 802.11 规定的载波多路侦听和冲突避免标准，简言之就是一个 Wi-Fi 热点同时只能跟一个手机通讯，其它手机在跟热点通讯前都要先探测或询问自己是否能够通讯，所以一个 Wi-Fi 热点使用的人越多就越慢。同时 Wi-Fi 信号受建筑墙体的屏蔽干扰很严重，而一般的中国普通家庭很少在装修时考虑好 Wi-Fi 路由器和各个房间的信号衰减问题，可能主播本人也不清楚自己做直播的房间离家里的路由器究竟穿了几堵墙。

- **原因3：下行不佳**
  即观众的下载带宽跟不上或者网络波动较大，例如直播流的码率是2Mbps的，也就是每秒钟有2M比特的数据流要下载下来，但如果观众端的带宽不够，就会导致观众端播放体验非常卡顿。 下行不佳只会影响当前网络环境下的观众。

## 查看 SDK 状态提示信息

如果您使用的是腾讯云视立方 · 直播 SDK 来推流，该 SDK 提供了一种状态反馈机制，每隔2秒就会将内部各种状态参数反馈出来，您可以通过注册 V2TXLivePusherObserver 监听器，然后通过回调函数 onStatisticsUpdate 来获取这些状态。这里以最新的终端 V2版本为例进行说明，若为终端V1，详情请参见 优化视频卡顿（V1）。
V2TXLivePusherStatistics 相关状态的说明如下：

| 推流状态 | 含义说明 |
| --- | --- |
| appCpu | 当前 App 的 CPU 使用率（%） |
| systemCpu | 当前系统的 CPU 使用率（%） |
| width | 视频宽度 |
| height | 视频高度 |
| fps | 帧率（fps） |

| | |
|---|---|
| audioBitrate | 音频码率（Kbps） |
| videoBitrate | 视频码率（Kbps） |

## 解决帧率太低问题

### 1. 帧率太低的评判

通过直播 SDK 的 V2TXLivePusherObserver 的 onStatisticsUpdate 回调中 **V2TXLivePusherStatistics.fps** 的状态数据，我们可以获得当前推流的视频帧率。正常来说每秒15帧以上的视频流才能保证观看的流畅度，常规推流如果 FPS 在10帧以下，观众就会明显的感到画面卡顿。

### 2. 针对性优化方案

- **2.1 观察 appCpu 和 systemCpu 的大小**
  通过直播 SDK 的 V2TXLivePusherObserver 的 onStatisticsUpdate 回调中 **V2TXLivePusherStatistics.appCpu** 和 **V2TXLivePusherStatistics.systemCpu** 的状态数据，我们可以获得**当前推流 SDK 的 CPU 占用情况**和**当前系统的 CPU 占用情况**。如果当前系统的整体 CPU 使用率超过80%，那么视频的采集和编码都会受到影响，无法正常发挥作用；如果 CPU 使用率达到100%，那么主播端本身就已经很卡，观众端要有流畅的观看体验显然是不可能的。
- **2.2 确认谁在消耗 CPU**
  一款直播 App 中使用 CPU 的不可能只有推流 SDK，弹幕、飘星、文本消息互动等都有可能会消耗一定的 CPU，这些都是不可避免的。如果单纯要测试推流 SDK 的 CPU 占用情况，可以使用我们的 工具包 DEMO 来观察和评估。
- **2.3 不盲目追高分辨率**
  过高的视频分辨率并不一定能带来清晰的画质：首先，较高的分辨率要配合较高的码率才能发挥效果，低码率高分辨的清晰度很多时候比不上高码率低分辨率。其次，像1280 x 720这样的分辨率在平均5寸左右的手机屏幕上并不能看出优势，要想跟960 x 540的分辨率拉开差距，只有在 PC 上全屏观看才能有明显的感官差异。但较高的分辨率会显著提升 SDK 的 CPU 使用率，因此常规情况下推荐使用直播 SDK 中 V2TXLivePusher 的 setVideoQuality 设置**高清**档即可，盲目追高分辨率有可能达不到预期的目标。

## 解决上传阻塞问题

据统计，视频云客户群80%以上的直播间卡顿问题，均是由于主播端上传阻塞所致。

### 1. 主动提示主播

对于注重清晰度的场景下，通过合适的 UI 交互提示主播**当前网络质量很糟糕，建议您拉近离路由器的距离，避免 Wi-Fi 穿墙**是最好的选择。
直播 SDK 的推流功能文档中有涉及**事件处理**的介绍，您可以利用它来做到这一点，推荐的做法是：如果 App 在短时间内连续收到直播 SDK 的多个 V2TXLIVE_WARNING_NETWORK_BUSY 事件，则提示主播网络关注一下当前网络质量，因为对于上行阻塞这种情况而言，主播本人是没办法通过视频的表现感知到的，只能通过观众的提醒或者 App 的提醒来了解。

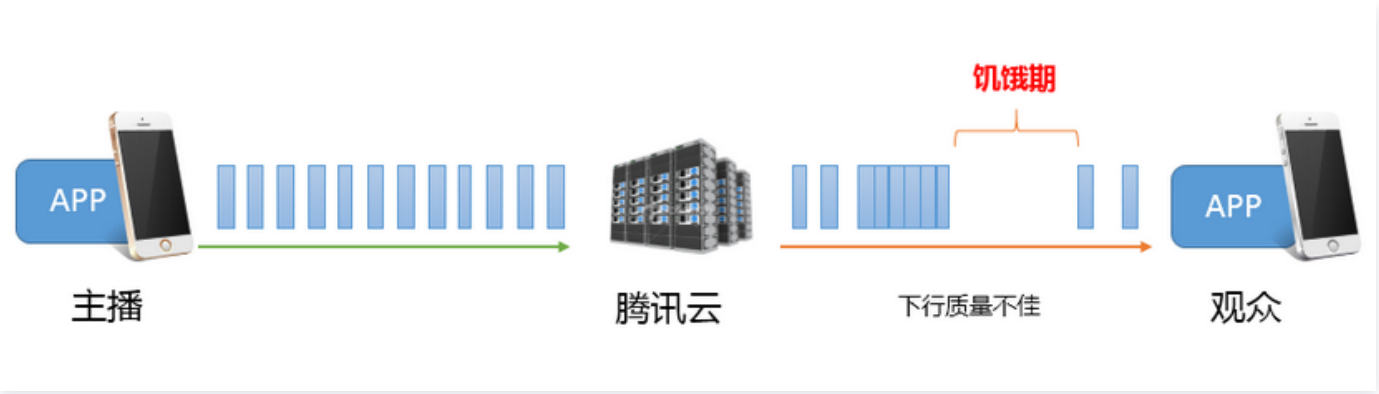
### 2. 合理的编码设置

如下是我们推荐的编码设置（更多信息请参见 设定画面质量），可以通过 V2TXLivePusher 里的 setVideoQuality 接口进行相应档位的设置：

| 应用场景 | resolution | resolutionMode |
|---|---|---|
| 秀场直播 | • V2TXLiveVideoResolution960x540<br>• V2TXLiveVideoResolution1280x720 | 横屏或者竖屏 |
| 手游直播 | V2TXLiveVideoResolution1280x720 | 横屏或者竖屏 |
| 连麦（主画面） | V2TXLiveVideoResolution640x360 | 横屏或者竖屏 |

| | | |
|---|---|---|
| 连麦（小画面） | V2TXLiveVideoResolution480x360 | 横屏或者竖屏 |
| 蓝光直播 | V2TXLiveVideoResolution1920x1080 | 横屏或者竖屏 |

## 优化播放端




### 1. 卡顿与延迟

如上图，下行网络的波动或者下行带宽不够，都会导致在播放过程中出现一段段的**饥饿期**（App 这段时间内拿不到可以播放的音视频数据）。如果想要让观看端的视频卡顿尽量少，就要尽可能地让 App 缓存足够多的视频数据，以保证它能平安度过这些“饥饿期”，但是 App 缓存太多的音视频数据会引入一个新的问题，即**高延迟**，这对互动性要求高的场景是很坏的消息，同时如果不做延迟修正和控制，卡顿引起的延迟会有**累积效应**，就是播放时间越久，延迟越高，延迟修正做得好不好是衡量一款播放器是否足够优秀的关键指标。所以**延迟和流畅是一架天平的两端**，如果过分强调低延迟，就会导致轻微的网络波动即产生明显的播放端卡顿。反之，如果过分强调流畅，就意味着引入大量的延迟（典型的案例就是 HLS（m3u8）通过引入20秒 – 30秒的延迟来实现流畅的播放体验）。

### 2. 针对性优化方案

为了能够让您无需了解过多流控处理知识就能优化出较好的播放体验，腾讯云视立方 · 直播 SDK 经过多个版本的改进，优化出一套自动调节技术，并在其基础上推出了三种比较优秀的 延迟控制方案，可以通过 V2TXLivePlayer 的 setCacheParams 来设置：

- **自动模式**：若您不太确定您的主要场景是什么，可以直接选择这个模式。

> **说明**
> 在该模式下播放器会根据当前网络情况，对延迟进行自动调节（默认情况下播放器会在1秒 – 5秒这个区间内自动调节延迟大小，您也可以通过 setCacheParams 对默认值进行修改），以保证在足够流畅的情况下尽量降低观众跟主播端的延迟，确保良好的互动体验。

- **极速模式**：主要适用于 **秀场直播** 等互动性高，并且对延迟要求比较苛刻的场景。

> **说明**
> 极速模式设置方法是 **minTime = maxTime = 1s** ，自动模式跟极速模式的差异只是 maxTime 有所不同（极速模式的 maxTime 一般比较低，而自动模式的 maxTime 则相对较高），这种灵活性主要得益于 SDK 内部的自动调控技术，可以在不引入卡顿的情况下自动修正延时大小，而 maxTime 反应的就是调节速度：maxTime 的值越大，调控速度会越发保守，卡顿概率就会越低。

- **流畅模式**：主要适用于 **游戏直播** 等大码率高清直播场景。

> **说明**

- 在该模式下播放器采取的处理策略跟 Adobe Flash 内核的缓存策略如出一辙：当视频出现卡顿后，会进入 loading 状态直到缓冲区蓄满，之后进入 playing 状态，直到下一次遭遇无法抵御的网络波动。默认情况下缓冲大小为5秒，您可以通过 setCacheParams 进行更改。
- 在延迟要求不高的场景下，这种看似简单的模式会更加可靠，因为该模式本质上就是通过牺牲一点延迟来降低卡顿率。

版权所有：腾讯云计算（北京）有限责任公司

# 生成 UserSig 签名

最近更新时间：2022-10-08 16:33:47

## UserSig 介绍

UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。
目前，腾讯云视立方 · 直播 SDK、实时音视频（TRTC）以及即时通信（IM）等服务都采用了该套安全保护机制。要使用这些服务，您都需要在相应 SDK 的初始化或登录函数中提供 SDKAppID、UserID 和 UserSig 三个关键信息。
其中 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 的计算原理如下所示，其本质就是对 SDKAppID、UserID 和 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)))
```

**说明**

- `currtime` 为当前系统的时间，`expire` 为签名过期的时间。
- 更多相关详情，请参见 客户端计算 UserSig 和 服务端计算 UserSig 。

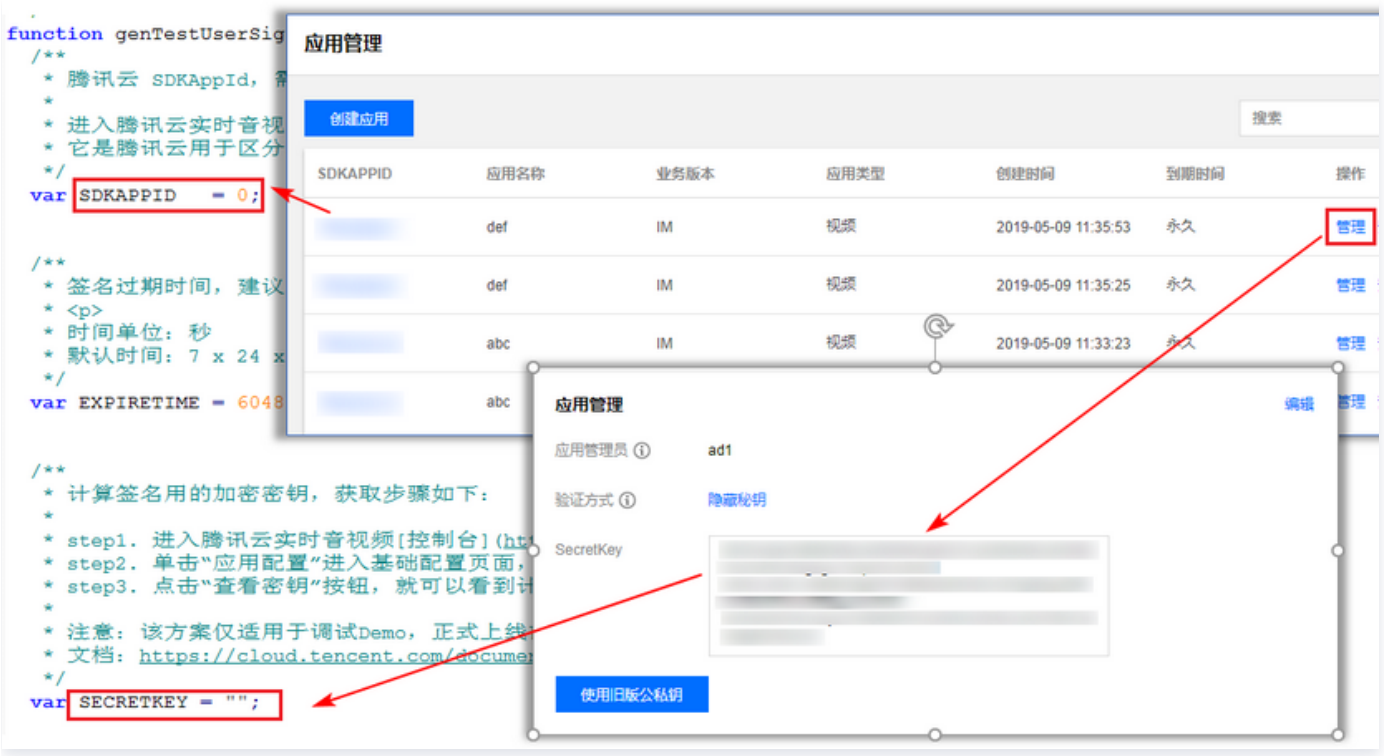
## 密钥获取

访问 **云直播控制台** > **应用管理** 可以查询计算 UserSig 用的密钥，方法如下：
1. 选择一个应用并进入详情页面，如果还没有应用就创建一个。
2. 进入**应用管理**页面，单击**查看密钥**按钮即可获得加密密钥。


## 客户端计算

我们在 IM SDK 的示例代码中提供了一个叫做 `GenerateTestUserSig` 的开源模块，您只需要将其中的 SDKAPPID、EXPIRETIME 和 SECRETKEY 三个成员变量修改成您自己的配置，就可以调用 `genTestUserSig()` 函数获取计算好的 UserSig，从而快速跑通 SDK 的相关功能：

| 语言版本 | 适用平台 | 源码位置 |
| --- | --- | --- |
| Objective-C | iOS | Github |
| Java | Android | Github |
| Javascript | 小程序 | Github |



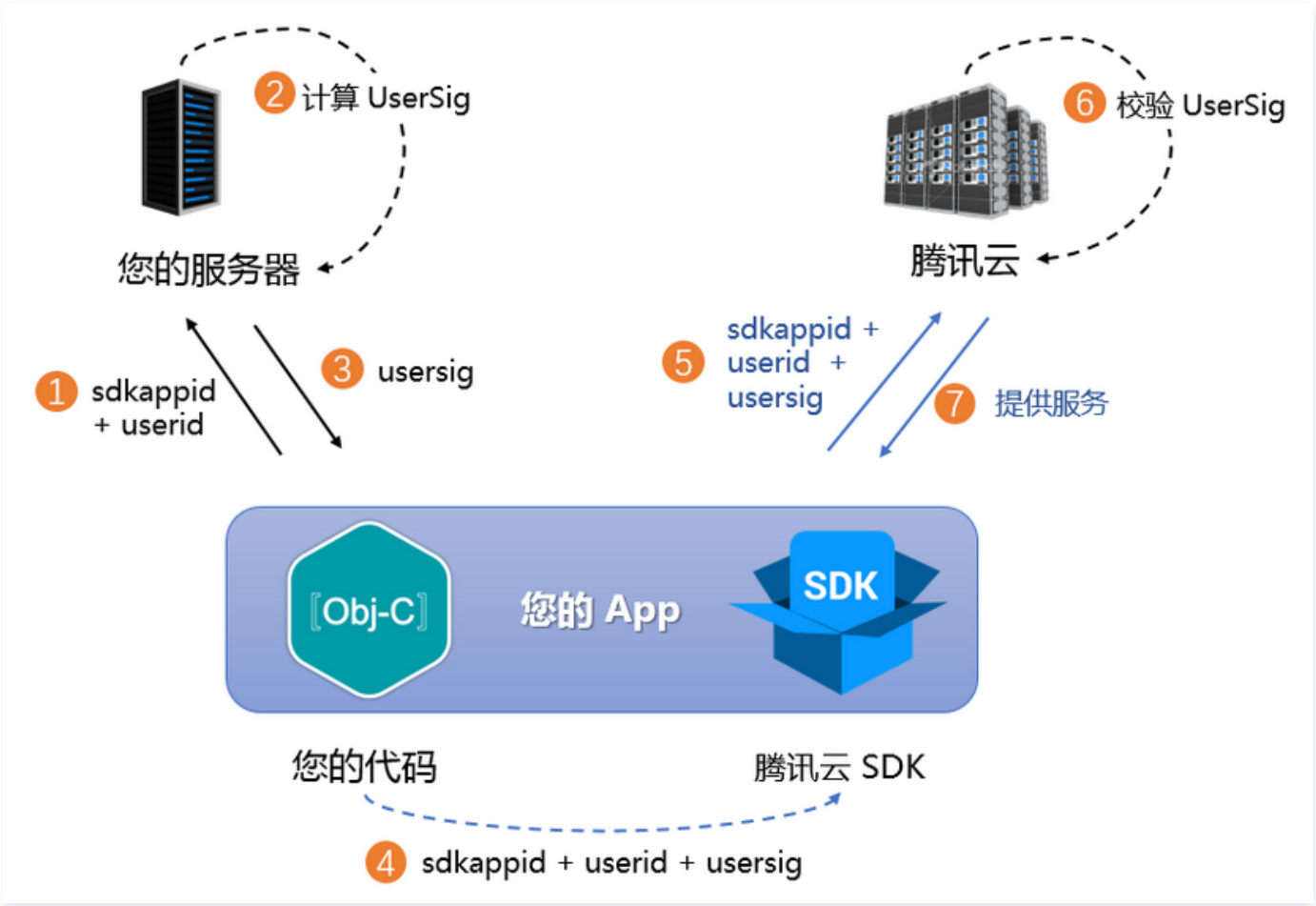

> **注意**
> 该方案仅适用于调试，如果产品要正式上线，**不推荐**采用这种方案，因为客户端代码（尤其是 Web 端）中的 SECRETKEY 很容易被反编译逆向破解。一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。
> 正确的做法是将 UserSig 的计算代码放在您的业务服务器上，然后由您的 App 在需要的时候向您的服务器获取实时算出的 UserSig。

## 服务端计算

采用服务端计算 UserSig 的方案，可以最大限度地保障计算 UserSig 用的密钥不被泄露，因为攻破一台服务器的难度要高于逆向一款 App。具体的做法如下：

1. 您的 App 在调用 SDK 的初始化函数之前，首先要向您的服务器请求 UserSig。
2. 您的服务器根据 SDKAppID 和 UserID 计算 UserSig，计算源码见文档前半部分。
3. 服务器将计算好的 UserSig 返回给您的 App。
4. 您的 App 将获得的 UserSig 通过特定 API 传递给 SDK。
5. SDK 将 SDKAppID + UserID + UserSig 提交给腾讯云服务器进行校验。
6. 腾讯云校验 UserSig，确认合法性。

7. 校验通过后，会向 TRTCSDK 提供实时音视频服务。




为了简化您的实现过程，我们提供了多个语言版本的 UserSig 计算源代码：

| 语言版本 | 签名算法 | 关键函数 | 下载链接 |
| --- | --- | --- | --- |
| Java | HMAC-SHA256 | genUserSig | Github |
| GO | HMAC-SHA256 | genUserSig | Github |
| PHP | HMAC-SHA256 | genUserSig | Github |
| Nodejs | HMAC-SHA256 | genUserSig | Github |
| Python | HMAC-SHA256 | genUserSig | Github |
| C# | HMAC-SHA256 | genUserSig | Github |

## 老版本算法

为了简化签名计算难度，方便客户更快速地使用腾讯云服务，即时通信 IM 服务自2019-08-06开始启用新的签名算法，从之前的 ECDSA-SHA256 升级为 HMAC-SHA256，也就是从2019-08-06之后创建的 SDKAppID 均会采用新的 HMAC-SHA256 算法。
如果您的 SDKAppID 是2019-07-19之前创建的，可以继续使用老版本的签名算法，算法的源码下载链接如下：

| 语言版本 | 签名算法 | 下载链接 |
| --- | --- | --- |
| Java | ECDSA-SHA256 | Github |

| | | |
|---|---|---|
| C++ | ECDSA-SHA256 | Github |
| GO | ECDSA-SHA256 | Github |
| PHP | ECDSA-SHA256 | Github |
| Nodejs | ECDSA-SHA256 | Github |
| C# | ECDSA-SHA256 | Github |
| Python | ECDSA-SHA256 | Github |

版权所有：腾讯云计算（北京）有限责任公司

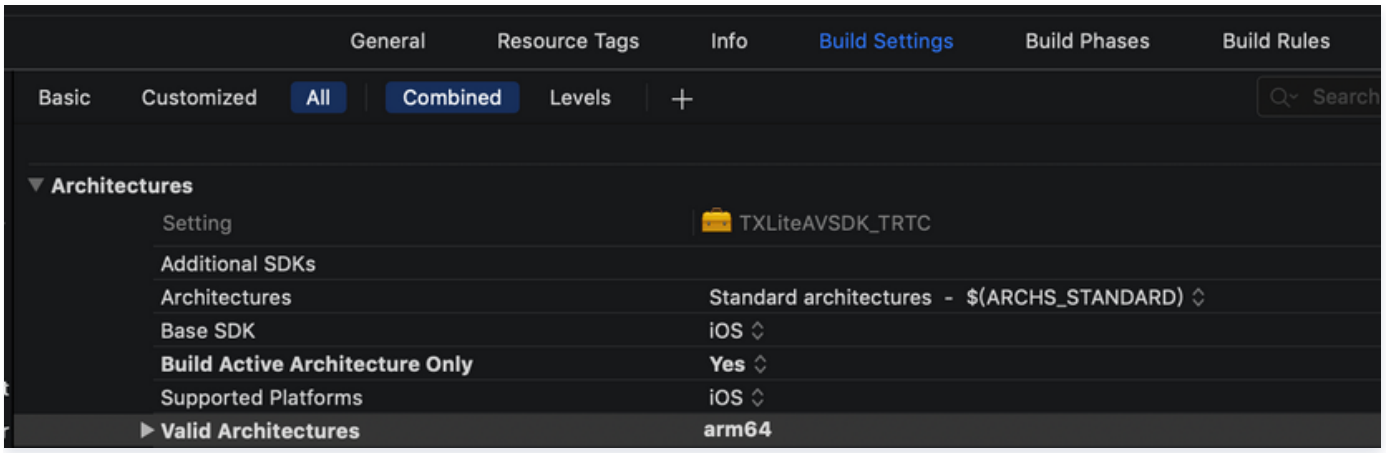
# 如何缩减安装包体积

最近更新时间：2022-11-25 15:07:09

## iOS 平台如何缩减安装包体积?

### 只打包 arm64 架构（荐）

苹果 iPhone 5s 及以后的手机都可以支持只打包 x64 架构，在 XCode 中的 Build Setting 里将 Build Active Architecture Only 设置为 YES，同时将 Valid Architectures 里只写 arm64 即可，TRTC SDK 的单架构 ipa 增量只有1.9M。




## Android 平台如何缩减安装包体积?

### 1. 只打包部分 so

如果您的 App 只定位国内使用，可以只打包 `armeabi-v7a` 架构的 so 文件，这样可以将安装包增量压缩到5M以内；如果您的 App 希望上架 Google Play，可以打包 `armeabi-v7a` 和 `arm64-v8a` 两个架构的 so 文件。
具体操作方法是在当前项目的 build.gradle 里添加 `abiFilters "armeabi-v7a"` 来指定打包单架构的 so 文件，或添加 `abiFilters "armeabi-v7a","arm64-v8a"` 来指定打包双架构的 so 文件。

- 如果不打算上架 Google Play：

```
android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
    defaultConfig {
        applicationId "com.tencent.liteav.demo"
        minSdkVersion rootProject.ext.minSdkVersion
        targetSdkVersion rootProject.ext.targetSdkVersion
        versionCode 1
        versionName "2.0"
        multiDexEnabled true

        ndk {
            abiFilters "armeabi-v7a"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles.add(file('proguard-rules.pro'))
        }
    }
```

- 如果希望上架到 Google Play：

```
android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
    defaultConfig {
        applicationId "com.tencent.liteav.demo"
        minSdkVersion rootProject.ext.minSdkVersion
        targetSdkVersion rootProject.ext.targetSdkVersion
        versionCode 1
        versionName "2.0"
        multiDexEnabled true

        ndk {
            abiFilters "armeabi-v7a","arm64-v8a"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles.add(file('proguard-rules.pro'))
        }
    }
```

## 2. 安装后下载 so（只打包 jar）

Android 版 SDK 的体积主要来自于 so 文件，如果您希望将安装包增量压缩到1M以内，可以考虑采用安装后再下载 so 的方式：

- **Step1：下载指定架构的 so 文件**
  在 GitHub 文件夹下，单击下载类似 `LiteAVSDK_Live_x.x.xxx.zip` 命名的压缩包，将其解压并找到指定架构的 so 文件。
- **Step2：将 so 文件上传到服务器**
  上传 Step1 中下载到的 so 文件，上传到您的服务器（或腾讯云 COS 对象存储服务）上，并记录下载地址，例如 `http://xxx.com/so_files.zip`。
- **Step3：首次启动 SDK 前下载 so 文件**
  在用户启动 SDK 相关功能前，例如开始播放视频之前，先用 loading 动画提示用户“正在加载相关的功能模块”。

在用户等待过程中，App 就可以到 `http://xxx.com/so_files.zip` 下载 so 文件，并存入应用目录下（例如应用根目录下的 files 文件夹），为了确保这个过程不受运营商 DNS 劫持的影响，请在文件下载完成后校验 so 文件的完整性，防止 zip 压缩包被运营商篡改。

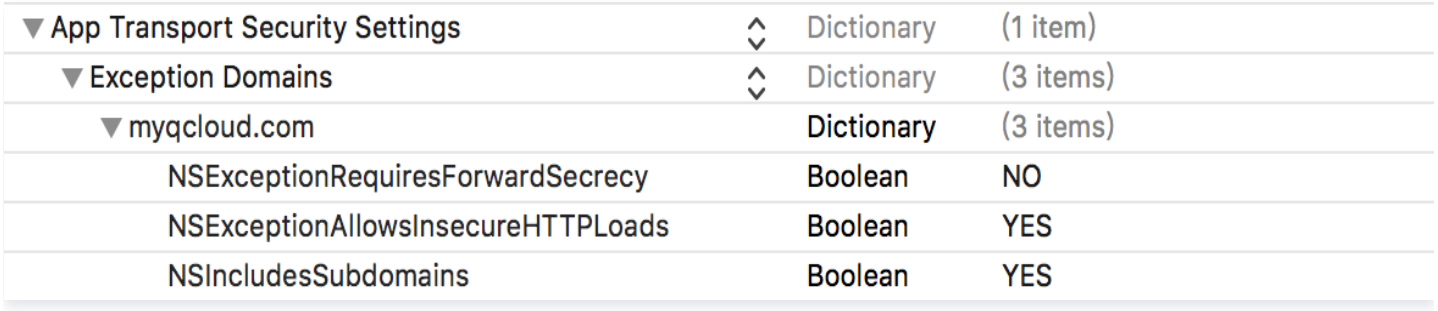
- **Step4：使用 API 加载 SO 文件**
  等待所有 so 文件就位以后，调用 `TXLiveBase` 类（LiteAVSDK 最早的一个基础模块）中的 `setLibraryPath()` 接口，将下载 so 的目标路径设置到 SDK。之后，SDK 会到这些路径下加载需要的 so 文件并启动相关功能。

> ⚠ **注意**
> 如果您的 App 希望上架 Google Play，请不要使用该方案，有可能会无法上架。

# 如何适配苹果 ATS

最近更新时间：2023-09-19 18:43:55

苹果在 WWDC 2016 中表示，从2017年01月01日起，所有新提交的 App 默认不能使用 `NSAllowsArbitraryLoads=YES` 来绕过 ATS 的限制。腾讯云已正式支持 HTTPS，您只需要使用新版 SDK（接口无变化），并且将原来的视频地址的前缀从 `http://` 换成 `https://` 即可，SDK 内部会自动适配。
但需要提醒的是，HTTPS 相比于 HTTP，在引入更多安全性（视频方面并不是特别需要）的同时也牺牲了连接速度和 CPU 使用率。所以如果您 App 在新的上架政策下，还需要继续使用 HTTP，方法是修改 Info.plist，将 `myqcloud.com` 加入到 `NSExceptionDomains` 中。具体的修改如图所示：




针对特定域名禁用 ATS 是可以被苹果审核所接受的，您可能需要在审核时进行说明， `myqcloud.com` 是用于视频播放的域名。