音视频终端 SDK（腾讯云视立方）

运维指南

腾讯云

【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

运维指南
生成 UserSig 签名

最近更新时间：2023-05-26 11:12:08

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，集成腾讯云直播 SDK、实时音视频（TRTC）以及即时通信（IM）等服务的腾讯云视立方也采用了该套安全保护机制。要使用这些服务，您需要 SDK 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。
- 上述原理图仅做 UserSig 计算原理说明，如需了解具体的 UserSig 拼接代码实现方式，请参见 客户端计算 UserSig 和 服务端计算 UserSig 。

密钥获取

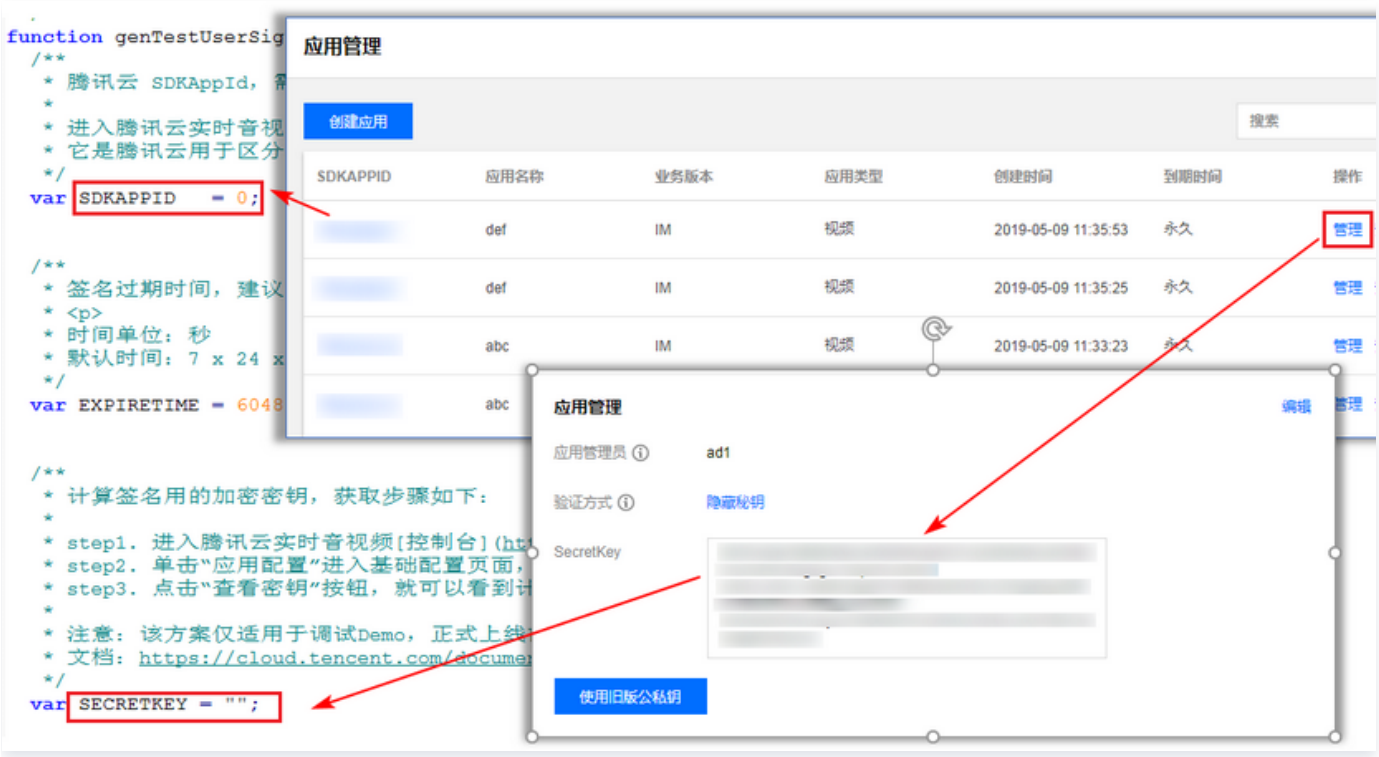
访问 **云直播控制台** > **应用管理** 可以查询计算 UserSig 用的密钥，方法如下：

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

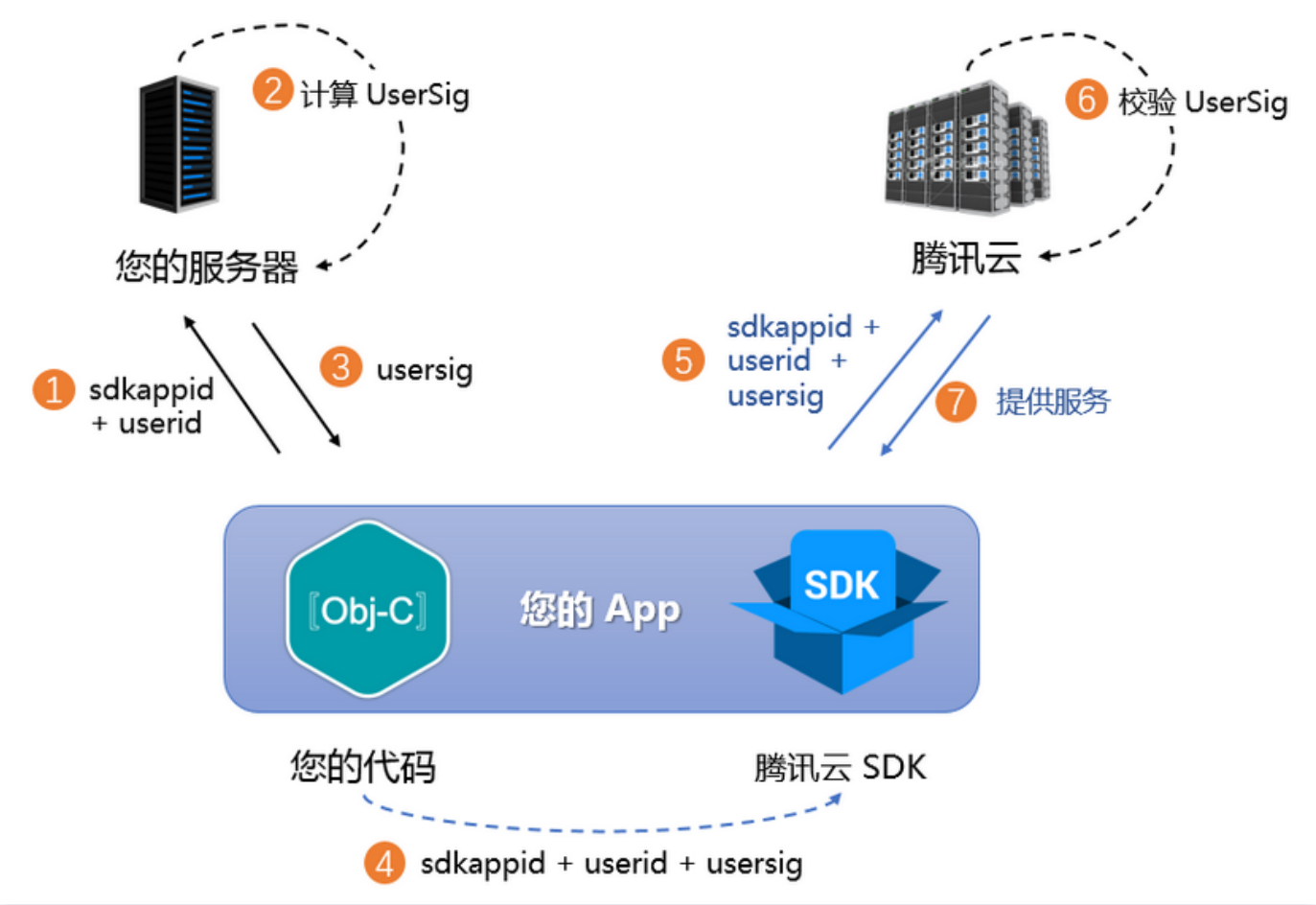
该方案仅适用于调试，如果产品要正式上线，**不推荐**采用这种方案，因为客户端代码（尤其是 Web 端）中的 SECRETKEY 很容易被反编译逆向破解。一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。

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
Node.js	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
Node.js	ECDSA-SHA256	Github
C#	ECDSA-SHA256	Github
Python	ECDSA-SHA256	Github

自主拼装推拉流 URL

最近更新时间：2023-09-20 17:52:23

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

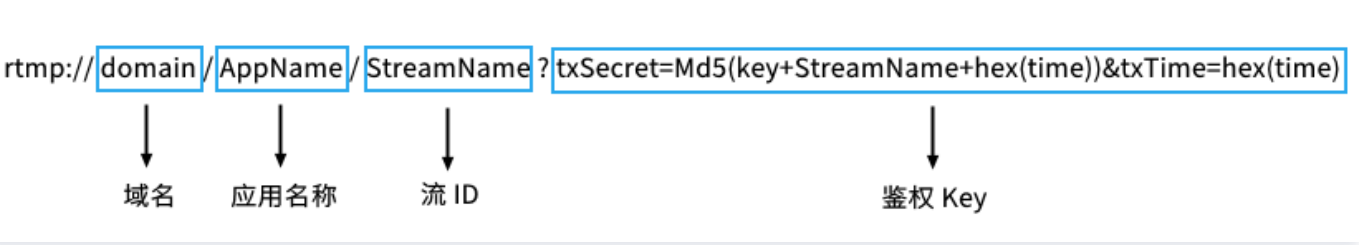
1. 登录云直播控制台。
2. 选择进入**直播工具箱** > **地址生成器**，进入如下配置：
   - 按需选择生成类型。
   - 选择您已添加到域名管理里对应的域名。
   - 按需编辑 AppName。AppName 为区分同一个域名下多个 App 的地址路径，默认为 live。
   - 填写自定义的流名称 StreamName。
   - 选择地址过期时间。
3. 单击**生成地址**即可生成您需要的推流/播放地址。

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))&amp;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 最佳实践-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))&amp;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 最佳实践-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	主播 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	主播 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。

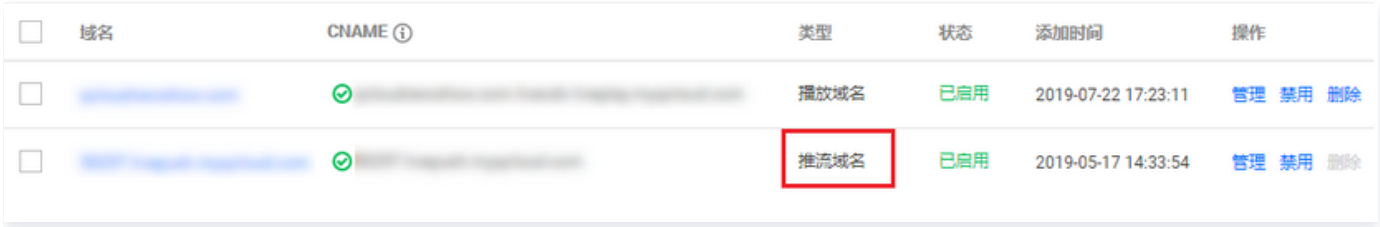
推流失败问题排查

最近更新时间：2023-09-19 21:58:53

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 推流所使用的默认端口号是**1935**，如果您测试时所在网络的防火墙不允许1935端口通行，就会遇到连不上与服务器的问题。此时您可以通过切换网络（例如4G）来排查是不是这个原因导致的问题。

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 txSercet 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 最佳实践 – 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，需要专业版和企业版才支持。
- 调用 startPush 推流传的推流地址缺少必要参数，请参考 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。

播放失败问题排查

最近更新时间：2023-08-18 09:55:34

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` 。

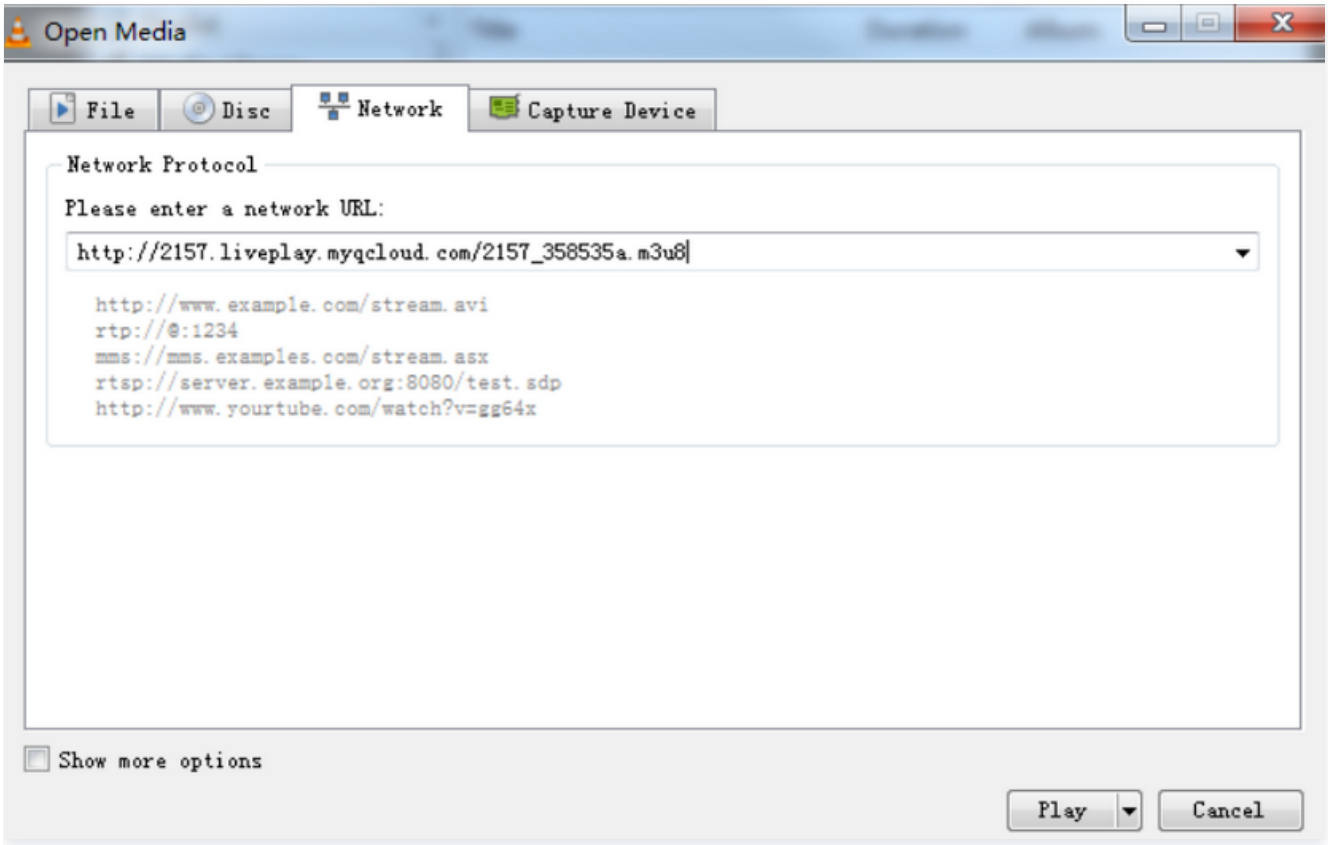
小直播的播放 URL 可以用调试的办法获取，您可以全局搜索代码寻找关键字 **startLivePlay**，然后在此处打下调试断点，这里是小直播对 RTMP SDK 的调用点，startLivePlay 的参数即为播放 URL。

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30秒才能观看**。
  - **腾讯云视立方 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、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；
  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。

降低直播延迟

最近更新时间：2023-03-17 14:15:36

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

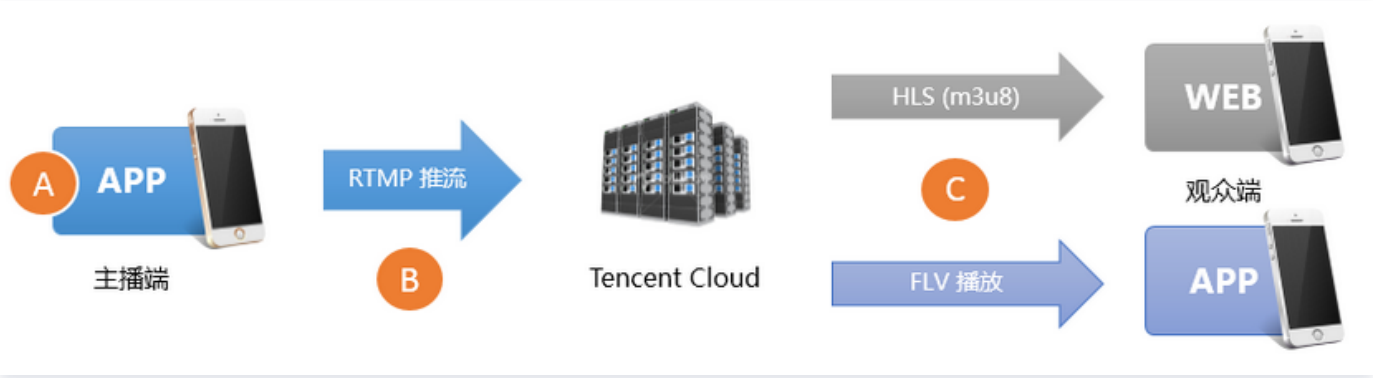
如果您采用的是 OBS 推流，并且发现在播放端延迟比较大。建议您按照 OBS 推流 中的描述配置对应的参数，并注意把关键帧间隔设置为1秒或2秒。

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。

优化视频卡顿

最近更新时间：2023-03-17 14:15:36



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。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。

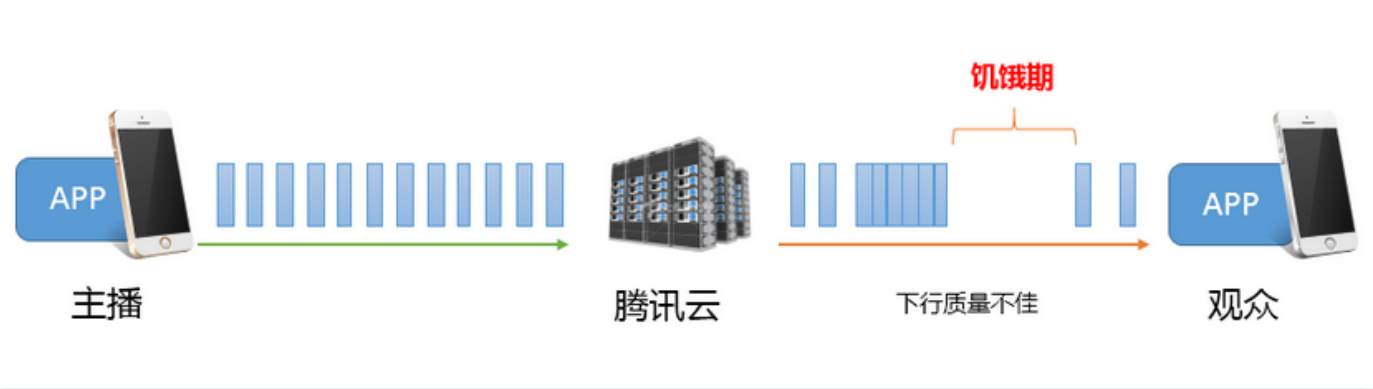
2. 合理的编码设置

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 • 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。

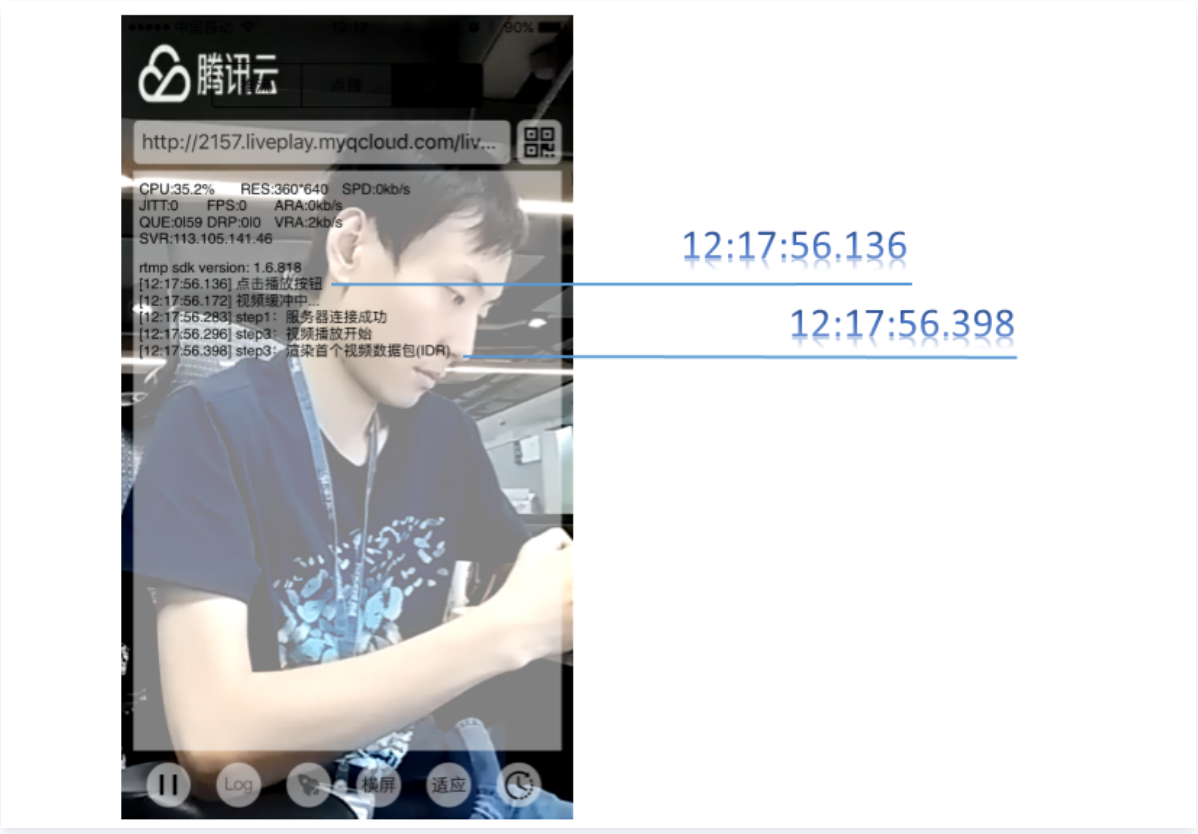
- 在延迟要求不高的场景下，这种看似简单的模式会更加可靠，因为该模式本质上就是通过牺牲一点延迟来降低卡顿率。

实现播放秒开

最近更新时间：2023-09-20 17:52:23

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音视频 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 基础直播Smart版本功能 + 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。
- **腾讯云音视频 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），而腾讯云音视频 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。

Web 播放器播放失败问题排查

最近更新时间：2023-09-20 17:52:23

本文主要介绍 Web 端视频播放的几类常见问题及相应解决方案。

视频播放失败

视频播放失败有多种原因，定位问题的基本思路是：

1. 配置网络抓包，查看网络请求情况。
2. 查看浏览器控制台报错情况。
3. 检查视频格式，使用的浏览器是否支持播放。

以下是视频播放失败的几种原因，以及对应的解决方案：

网络

跨协议拦截

- **问题表现：**在 HTTPS 协议的页面播放 HTTP 协议的视频时，浏览器会出于安全考虑进行拦截。
- **解决方案：**HTTP 协议的页面播放 HTTP 的视频，HTTPS 协议的页面播放 HTTPS 的视频。

CDN 无视频

- **问题表现：**访问视频地址返回404。
- **解决方案：**请 联系我们 定位并修复 CDN 资源。

CDN 鉴权失败

- **问题表现：**访问视频地址返回403，无法加载视频。
- **解决方案：**需确认是否开启 referer 防盗链或者 key 防盗链，视频播放时是否具备校验参数。

微信浏览器拦截

- **问题表现：**在微信无法播放视频，非微信情况下可以播放。
- **解决方案：**需要通过微信申诉解除拦截。

跨域问题

- **问题表现：**在 PC 端无法播放视频，浏览器控制台报跨域相关的错误。
- **问题背景：**在 PC 端使用 Flash 播放视频需要检查视频服务器的 `crossdomain.xml` 文件。

> **crossdomain.xml 的作用：**
> - 位于 `www.a.com` 域中的 SWF 文件要访问 `www.b.com` 的文件时，SWF 首先会检查 `www.b.com` 服务器根目录下是否有 `crossdomain.xml` 文件，如果没有，则访问不成功；如果 `crossdomain.xml` 文件存在，且文件内设置了允许 `www.a.com` 域访问，则通信正常。
> - `crossdomain.xml` 中配置的是 SWF 文件的域名。

在 PC 端的现代浏览器使用 HTML5 播放 HLS 和 FLV 时，视频服务器需要配置跨域资源共享 CORS 。
正常情况下，腾讯云服务会自动配置这两项跨域策略，如遇到异常情况请 联系我们 。

- **解决方案：**视频存储服务器需要部署 `crossdomain.xml` 文件并配置正确的访问策略，以及开启 CORS 支持。

视频未转码

- **问题表现：**在腾讯云控制台上传视频后，播放器提示视频未转码。
- **解决方案：**对视频进行转码操作，具体操作请参见 处理视频 ，确保视频编码格式为 H.264，视频封装格式为 MP4 或者 HLS。

异常视频

- **问题表现：**转码后的视频出现花屏、黑屏、卡顿和无法播放等现象，可能是原始视频有问题或者视频转码失败。
- **解决方案：**需要定位原始视频是否有问题，如果是转码问题请 联系我们 。

浏览器环境不支持播放

通常情况下在 Web 端播放视频依赖浏览器自带的解码器，或者 Flash 解码器，不支持播放会出现 error code 为3或4的错误。浏览器不支持播放的常见问题如下：

浏览器不支持 Flash

- **问题表现：**无法播放 RTMP 和 FLV 格式的视频，或者无法在 IE 浏览器中播放视频。
- **解决方案：**播放 RTMP、FLV 格式的视频以及在 IE 中播放视频都依赖 Flash 插件，请安装并启用 Flash 插件。

浏览器不支持 MSE

- **问题表现：**在 PC 浏览器不支持 Flash 的情况下，使用 H5 方式无法播放 HLS、FLV 格式的视频。
  解决方案：不支持 Flash 的情况下，播放器将使用 MSE 播放 HLS、FLV 格式的视频，如浏览器不支持，只能更换或升级浏览器，目前支持通过 MSE 播放 HLS、FLV 格式视频的浏览器有 Edge、Chrome、Firefox 和 Safari11+。

浏览器不支持解码 H264 或者不支持播放 MP4、HLS 格式的视频

- **问题表现：**排除其他情况后仍无法播放 MP4、HLS 格式的视频，通常出现在部分 PC 软件或者 App 集成精简版本的浏览器内核中，没有对应的视频解码器，会出现无法播放 MP4、HLS 格式视频的情况。
- **解决方案：**在 PC 软件或 App 中升级浏览器内核，或者集成 Flash 插件，并允许调用 Flash 插件。

HLS 加密视频播放失败

HLS 加密视频的播放流程有别于常规视频，通常需要确保获取 key 这个步骤是正常的，常见问题如下：

获取 key 失败

- **问题表现：**获取密钥接口无返回，或者返回非密钥数据。
- **解决方案：**检查 m3u8 文件格式是否符合规范，获取密钥的地址是否正确，密钥接口服务端鉴权是否正常，密钥接口是否正常返回。

解密失败

- **问题表现：**获取密钥接口正常返回，仍无法播放。
- **解决方案：**检查密钥长度，确保密钥长度为16字节，并且是能正确解密的密钥。