# 移动直播

# 技术支持

# 产品文档

腾讯云

# 文档目录

技术支持
- 如何联系我们
- 快速获取URL
- 直播基础知识
- 视频卡顿怎么办
- 如何实现秒开
- 如何降低延迟
- 为何推流不成功
- 为何直播看不了
- 如何实现好的画质
- 如何开通各项云服务
- 如何录制并回看
- 如何适配苹果 ATS
- 事件通知码

# 技术支持
# 如何联系我们

最近更新时间：2018-09-19 20:26:13

**常见对接问题**

- 直播基础知识！
- 视频卡顿怎么办？
- 如何实现秒开？
- 如何降低延迟？
- 为何推流不成功？
- 为何直播看不了？
- 如何实现好的画质？
- 如何开通各项云服务？
- 如何实现连麦功能？
- 如何录制并回看？
- 腾讯云ATS适配情况？

## 热线电话

当您在使用腾讯云遇到问题时，可以直接致电客服人员，寻求相应的帮助。

- 客服咨询：4009-100-100
- 备案帮助：4009-100-100 转 3

## 工单系统

当您遇到运维或技术类产品使用问题时，可以登录腾讯云官网，根据界面指引提交工单，我们将尽快响应，期待收到您的宝贵意见。
工单相关入口如下：

- 工单提交：提交工单
- 状态查询：工单列表

工单状态说明如下：

- 未处理：新提交的工单，或者技术支持收到工单但评估还未完成。您可以对未处理工单进行补充和关闭操作。

- 处理中：技术支持收到工单并评估，正在实施过程中。您无法对处理中的工单做任何操作。
- 待补充：技术支持收到工单并评估，未提交完整的信息，需要补充。您可以对待补充工单进行关闭操作。

> 说明：当“待补充”的工单补充后重新提交，会再次进入“未处理”状态。

- 已结单：工单实施完成，或者您在技术支持操作前关闭工单。

## 问答社区

问答社区是腾讯云为开发者提供的交流平台。该平台汇聚了云计算、人工智能、小程序、产品文档等热门标签版块。
您可以登录问答社区进行提问，会有产品专家或热心用户对您提出的问题，进行相应解答、探讨和指导。
平台入口：问答

## 技术交流

关注公众号"腾讯云视频"，在聊天窗口发送关键字"技术支持"，将有专人和您联系。

# 快速获取URL

最近更新时间：2018-02-22 19:12:31

## 快速获得URL

如果您是想要生成一组URL用于测试，那您只需要打开直播控制台>>直播码接入>>推流生成器，点击 **生成推流地址** 按钮，即可生成一个推流URL和三种不同播放协议的播放URL。


使用我们的 Demo 视频云工具包 中的 **RTMP 推流** 和 **直播播放器** 功能，可以快速测试推流 URL 和播放 URL 的有效性。

## 后台派发URL

从设计灵活性的角度来讲，后台直接派发 URL 要比把 URL 写死在 APP 里要好得多。相应的，如果您的直播产品是主播可以自由开播的，那么像上面这样手动生成 URL 地址也无法满足需求。

这时，就需要您的后台自动派发 URL 了，文档 DOC 中介绍了腾讯云推流和播放 URL 的组成，以及如何让您的后台业务服务器自己生成 URL。

# 直播基础知识

最近更新时间：2018-07-11 12:01:13

## 1. 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（类似优酷土豆、爱奇艺和腾讯视频），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。

## 2. 常见的直播协议有哪些？

目前常见的直播协议有三种：RTMP、 FLV 和 HLS。

- **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“剁碎”，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，由于这种极致的简洁，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 5-10 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 5-10 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 10-30s 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。

| 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 | 优质线路下理论延迟最低 | 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 |

## 3. 常见的点播协议有哪些？

目前常见的点播格式有三种：MP4、HLS 和 FLV。

- **MP4**：非常经典的文件格式，在移动终端和 PC 浏览器上的支持度都很好（在 iOS 和大部分 Android 设备上，都可以使用系统浏览器进行播放，在 PC 上可以使用 Flash 控件进行播放），但是 MP4 的视频文件格式比较复

杂，所以处理成本高，而且由于索引表复杂度高，导致时长稍大（比如半小时）的 MP4 文件在线播放时加载速度会很慢。

- **HLS**：苹果公司力推的标准，在移动终端的浏览器上的支持度较好，但 IE 的支持情况依赖 Flash 的二次开发工作（建议使用腾讯视频云的 Flash 播放器控件），其精简的 m3u8 的索引结构可以规避 MP4 的索引慢问题，如果是用于点播，是非常不错的选择。
- **FLV**：Adobe 公司所推的标准，目前直播平台最常用的封装格式，在 PC 端有 Flash 的强力支持，但在移动终端只有 App 实现播放器才有可能支持（或者使用本播放器），大部分手机端浏览器均不支持，目前腾讯视频云的直播录制，采用的就是 FLV 视频格式。

| 点播协议 | 优点 | 缺点 |
| --- | --- | --- |
| HLS(m3u8) | 手机浏览器支持度高 | 大量小分片的文件组织形式，错误率和维护成本均高于单一文件 |
| MP4 | 手机浏览器支持度高 | 格式过于复杂和娇贵，容错性很差，对播放器的要求很高 |
| FLV | 格式简单问题少，适合直播转录制场景 | 手机浏览器支持差，需集成SDK才能播放 |

### 4. 常见的推流协议有哪些？

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】>【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，由于腾讯视频云 SDK 第一个功能模块就是主播推流，所以也被称为是 RTMP SDK。

### 5. 腾讯 RTMP SDK 支持哪些功能和协议?

腾讯视频云 RTMP SDK 支持推流、直播和点播三个功能：

- **推流**：支持 **RTMP 发布协议**，并包含硬件加速、美颜滤镜、带宽适应、清晰度调整等强大功能。
- **直播**：支持 FLV 协议和 RTMP 协议，**推荐使用 FLV**，具有秒开优化，延迟自动控制技术以及适应性良好的硬件解码能力。
- **点播**：支持 **MP4\HLS\FLV** 文件**在线**或**本地**点播服务，注意老版本 SDK 是只支持 FLV 点播的。

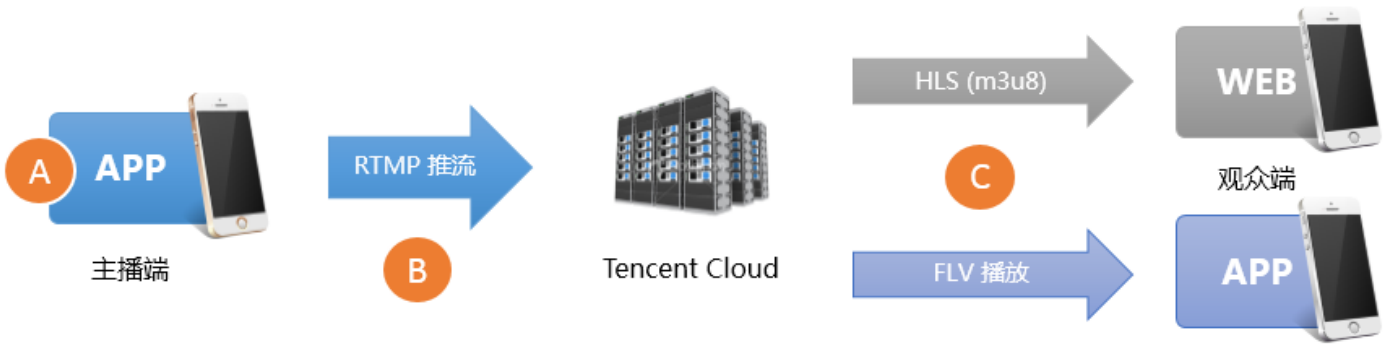
### 6. 直播需要网络视听许可证吗？

通过互联网向用户提供直播等音视频服务的应用程序，需要取得相关部门颁发的《信息网络传播视听节目许可证》。

# 视频卡顿怎么办

最近更新时间：2018-07-11 12:01:44

## 1. 为什么会卡顿

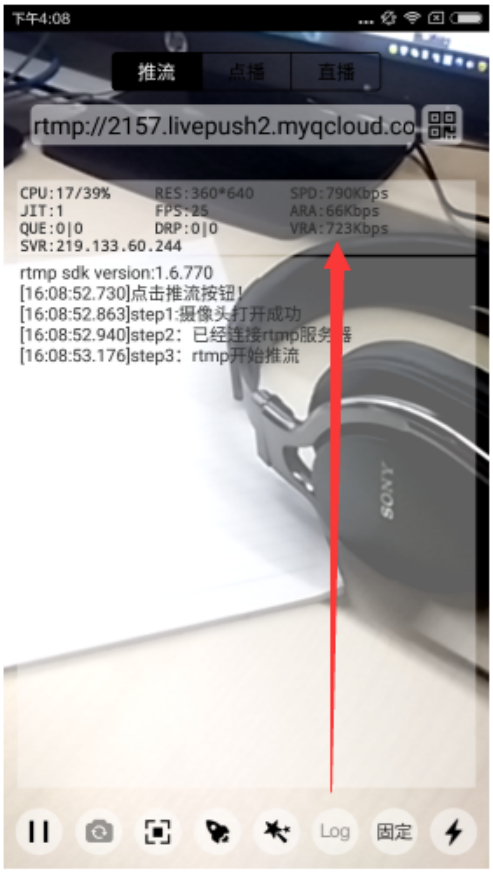



卡顿的原因无外乎三种情况：

- **原因 1：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。正常情况下每秒 15FPS 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。
- **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，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厉害，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**
  就是观众的下载带宽跟不上或者网络很波动，比如直播流的码率是 1Mbps 的，也就是每秒钟有 1M 比特的数据流要下载下来，但如果观众端的带宽不够，就会导致观众端体验非常卡顿。 下行不佳只会影响当前网络环境下的观众。

# 2. 发现问题的“眼睛”

推流 SDK 提供了一种状态反馈机制，每隔 1-2 秒就会将内部各种状态参数反馈出来，我们可以通过注册 **TXLivePushListener** 监听器来获取这些状态。




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 |

| 推流状态 | 含义说明 |
| --- | --- |
| 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 |

# 3. 帧率太低

## 3.1 帧率太低的评判

通过 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 15FPS 以上的视频流才能保证观看的流畅度，如果 FPS 在 10 帧以下，观众就会明显的感到画面卡顿。

## 3.2 针对性优化方案

- **3.2.1 观察 CPU_USAGE 的大小**
  通过 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。
- **3.2.2 确认谁在消耗 CPU**
  一款直播 App 中使用 CPU 的不可能只有 RTMP 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 简单版 DEMO 来观察和评估。
- **3.2.3 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，要像跟 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 TXLivePush 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。
- **3.3.4 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。TXLivePush 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，马赛克感很强是个硬伤），如果要使用硬件编码，可以使用 TXLivePushConfig 的 enableHWAcceleration 选项开启。

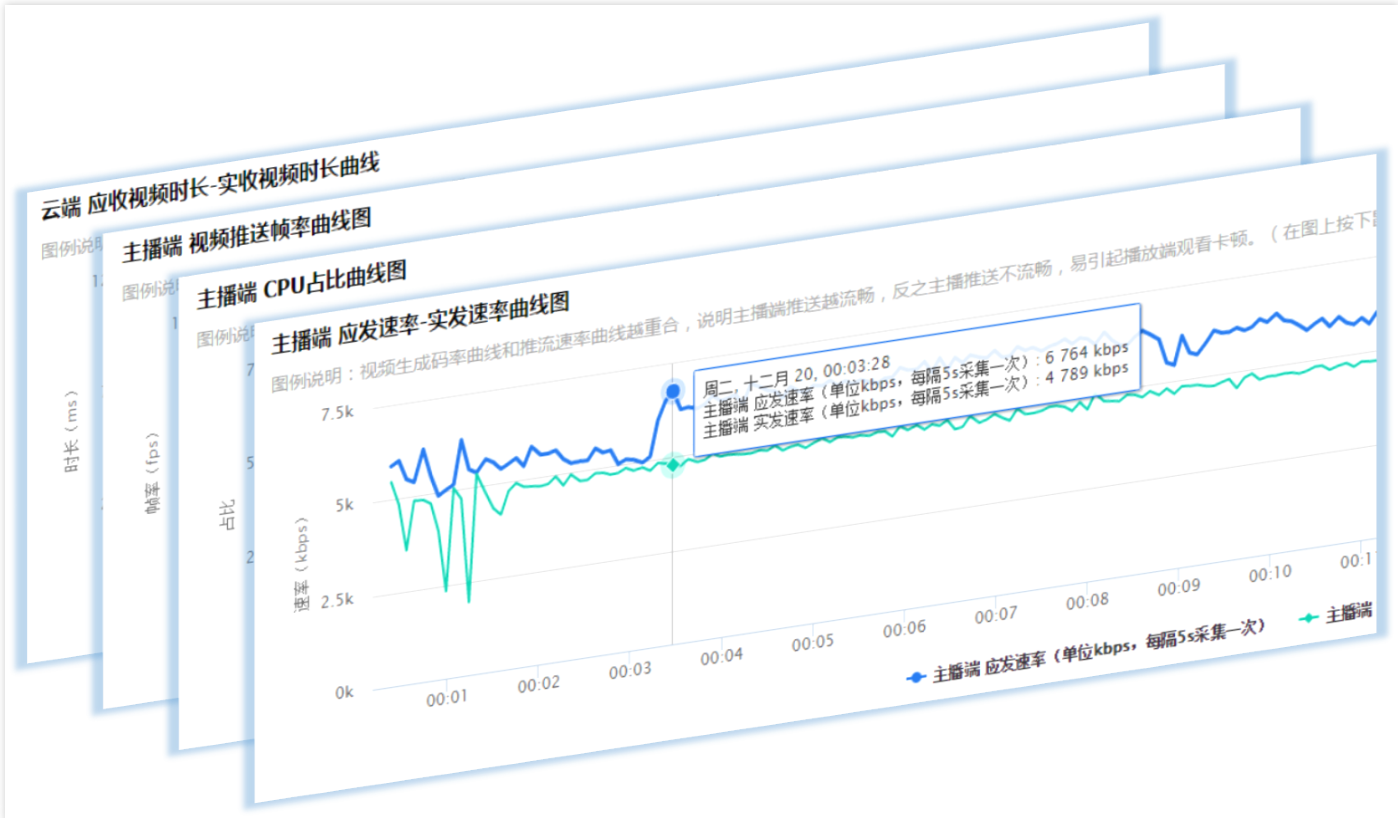
# 4. 上传阻塞

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。

## 4.1 上传阻塞的评判

- **4.1.1：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。
- **4.1.2：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：


2016-09-09 14:12:02
视频平均码率: 785
VIDEO_BITRATE
NET_SPEED
CACHE_SIZE


云端 应收视频时长-实收视频时长曲线
主播端 视频推送帧率曲线图
主播端 CPU占比曲线图
主播端 应发速率-实发速率曲线图
图例说明：视频生成码率曲线和推流速率曲线越重合，说明主播端推送越流畅，反之主播推送不流畅，易引起播放端观看卡顿。（在图上按下
周二, 十二月 20, 00:03:28
主播端 应发速率（单位kbps，每隔5s采集一次）：6 764 kbps
主播端 实发速率（单位kbps，每隔5s采集一次）：4 789 kbps
主播端 应发速率（单位kbps，每隔5s采集一次）
主播端

> 注意：
> 您可以在【直播控制台】>【质量监控】里看到类似上图的图表。

## 4.2 针对性优化方案

- **4.2.1 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。
  RTMP 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 RTMP 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。
- **4.2.2 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，更多信息请参考 如何实现更好的画质？），可以通过 TXLivePush 里的 setVideoQuality 接口进行相应档位的设置：

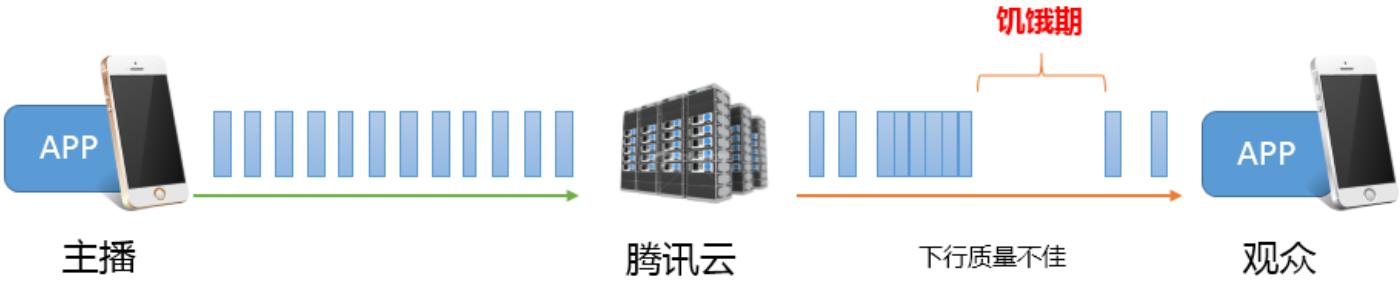
| 档位 | 分辨率 | FPS | 码率 | 使用场景 |
| --- | --- | --- | --- | --- |
| **标清** | 360*640 | 15 | 400kbps - 800kbps | 如果您比较关注带宽成本，推荐选择该档位，画质会偏模糊，但带宽费用较高清档要低 60%。 |
| **高清（推荐）** | 540*960 | 15 | 1200kbps | 如果您比较关注画质，推荐选择该档位，能确保绝大多数主流手机都能推出很清晰的画面。 |
| **超清** | 720*1280 | 15 | 1800kbps | 慎用：如果您的场景多是小屏观看不推荐使用，如果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

- **4.2.3 启用流控辅助**
  有些客户会说：“我们的 App 任何用户都有可能使用，让我去决定他们的网速不现实。”如果主播的上传网速的不确定性确实很大，推荐开启网络自适应，参考文档见 iOS 平台 & Android 平台 。不过我们依然建议优先参考 **4.2.1 主动提示主播** 中的方案，毕竟既要高清流畅，又不能保证上传网速，本身就是 **即要马儿跑得快，又要马儿不吃草** 的事情。

| 设置项 | 中文含义 | 建议值 |
| --- | --- | --- |
| videoBitrateMin | 最小码率 | 400 |
| videoBitrateMax | 最大码率 | 1000（推荐根据具体分辨率而定） |

| 设置项 | 中文含义 | 建议值 |
| --- | --- | --- |
| enableAutoBitrate | 码率自适应 | 开启 |
| autoAdjustStrategy | 码率调整策略 | AUTO_ADJUST_BITRATE_STRATEGY_2 |

# 5. 播放端的优化




## 5.1 卡顿 & 延迟

如上图，下行网络的波动或者下行带宽不沟通，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 —— **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 10-30 秒的延迟来实现流畅的播放体验）。

## 5.2 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 RTMP 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。

> 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 1s - 5s 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，因而对延迟要求比较苛刻的场景。

  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。

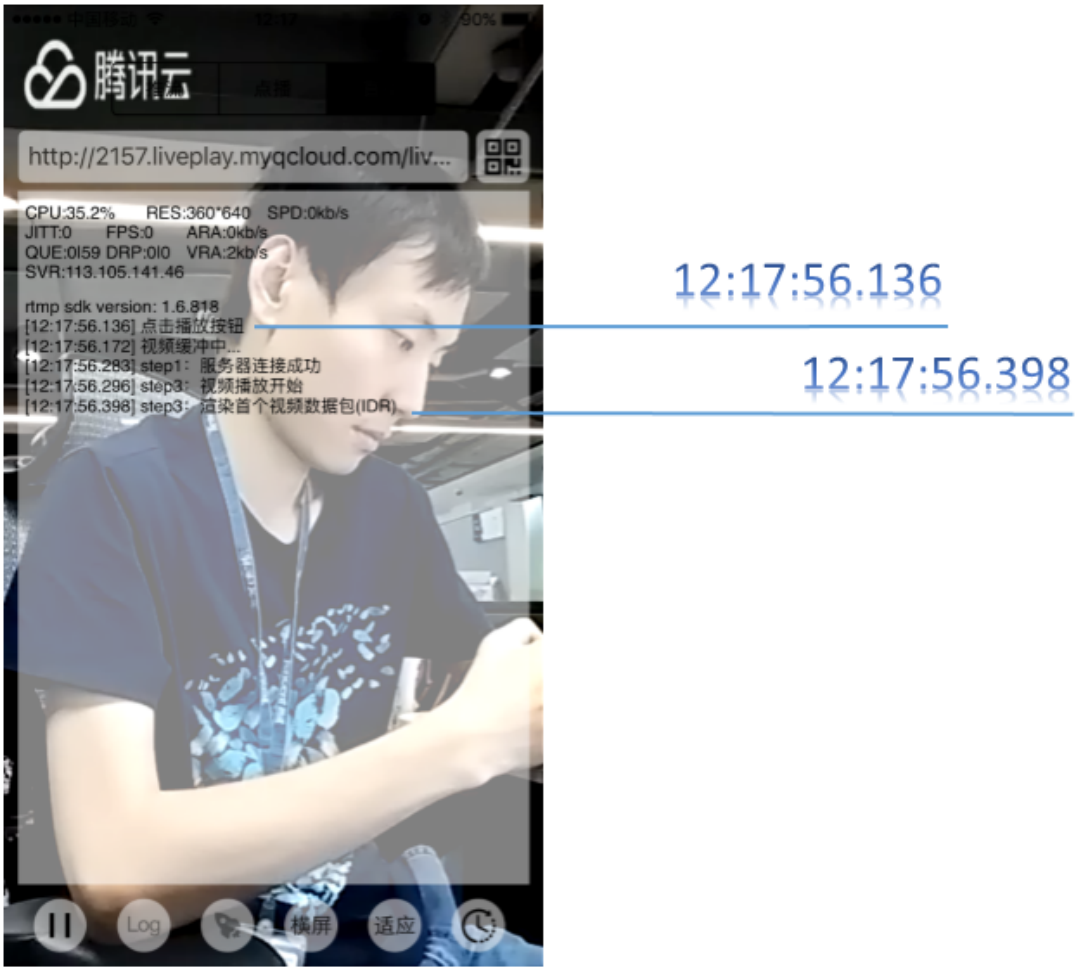
  当把播放器中的 setAutoAdjustCache 开关关闭，即为流畅模式，在该模式下播放器采取的处理策略跟 Adobe Flash 内核的缓存出策略如出一辙：当视频出现卡顿后，会进入 loading 状态直到缓冲区蓄满，之后进入 playing 状态，直到下一次遭遇无法抵御的网络波动。默认情况下缓冲大小为 5s，您可以通过 setCacheTime 进行更改。
  在延迟要求不高的场景下，这种看似简单的模式会更加可靠，因为该模式本质上就是通过牺牲一点延迟来降低卡顿率。

# 如何实现秒开

最近更新时间：2018-07-11 12:02:24

## 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，最快可以实现 **200ms左右** 的首屏打开速度，如果网络下行足够够好的话甚至可以瞬开。

## 如何实现“秒开”？

### APP 端

使用 腾讯云音视频 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（比如 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。

# 如何降低延迟

最近更新时间：2018-07-11 12:03:43

按正常情况 RTMP 推流+FLV 播放的正常延迟在 2-3 秒左右，太长则是有问题的，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

不少客户播放协议采用 HLS(m3u8)，并感觉延迟较大，这个是正常的。苹果主推的 HLS 是基于大颗粒的 TS 分片的流媒体协议，每个分片都有 5s 以上的时长，分片数量一般为 3-4 个，所以总延迟在 20s - 30s 就不足为怪。
换用 FLV 作为播放协议即可解决这个问题，但是要注意，如果您要在手机浏览器上观看直播视频，只有 HLS(m3u8) 这一种播放协议可以选择，其它的直播协议在苹果的 Safari 浏览器上都是不支持的。

## Step 2. 检查播放器设置

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

- **极速模式**：能保证绝大多数场景下延迟都在 2-3 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，比如游戏直播。


## Step 3. 后台不要打水印

腾讯云支持后台打水印，目的是满足一些不能使用腾讯云 RTMP SDK 的推流器（支持直播端加水印）但依然要打水印的客户，但是这种方案会引入额外的三秒延迟，故如果您本身使用的是腾讯云 RTMP SDK 来推流，就把后台水印关闭后在主播端的 App 上加水印。

### Step 4. 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，建议您先换成腾讯云 RTMP 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对会暴力的采用无限缓冲的方式来解决上行带宽不足的问题。

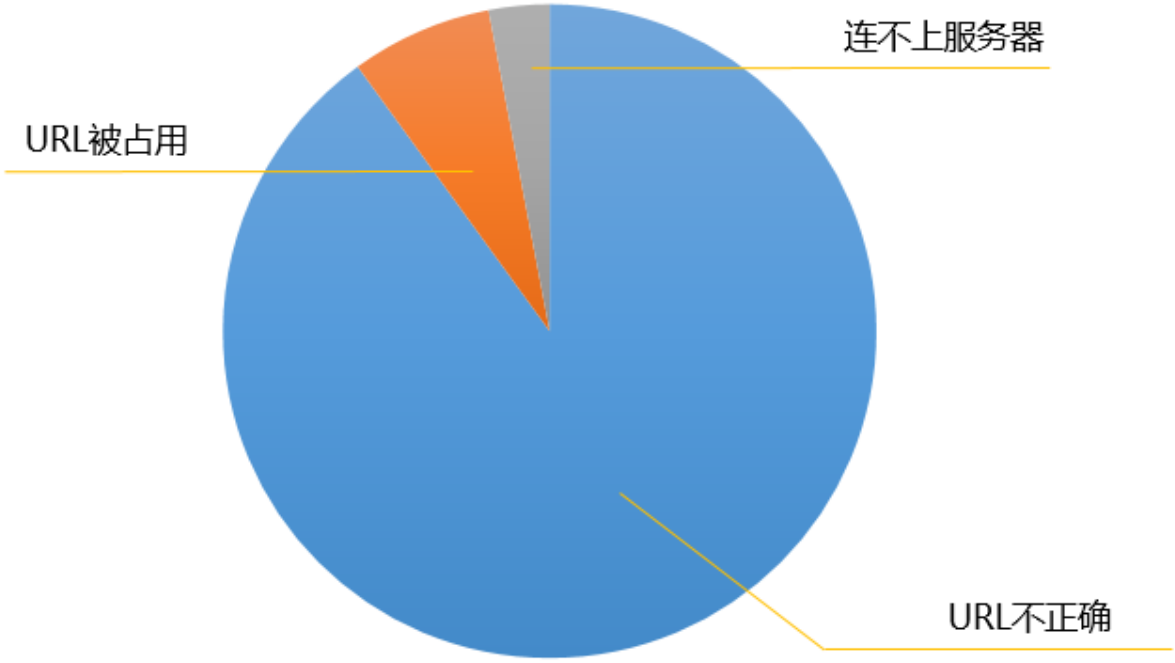
### Step 5. 检查 OBS 设置

不少客户反馈采用 OBS 推流，在播放端延迟比较大。建议按照 PC 推流 文档中配置对应的参数，并注意要把关键帧间隔设置为 1 或者 2。

# 为何推流不成功

最近更新时间：2018-07-11 12:05:48

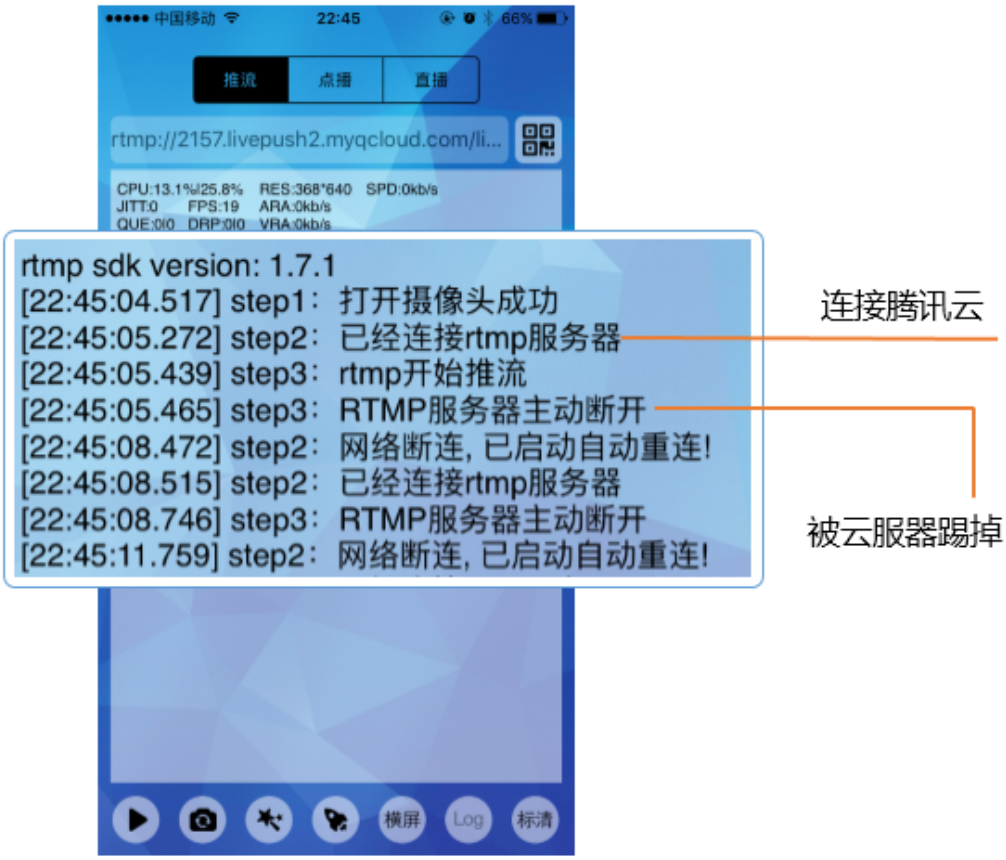
腾讯云统计客户投诉中关于推流不成功的原因，主要集中在如下三个点上：




## txSecret 不正确

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 RTMP SDK 会抛出 **PUSH_WARNING_SERVER_DISCONNECT** 事件， RTMP SDK DEMO 此

时的表现如下：




阅读 如何获取推流URL？ 了解如何获取可靠的推流URL。

## txTime 已过期

有些客户担心自己的直播流量被人盗用，会将 txTime 设置的过于保守，比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 的有效期不用设置的太短。相反，如果有效期设置的太短，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。

## 推流URL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，这种情况下 RTMP SDK 会抛出 **PUSH_WARNING_SERVER_DISCONNECT** 事件。

# 连不上云服务器

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（比如 4G ）来排查是不是这个原因导致的问题。

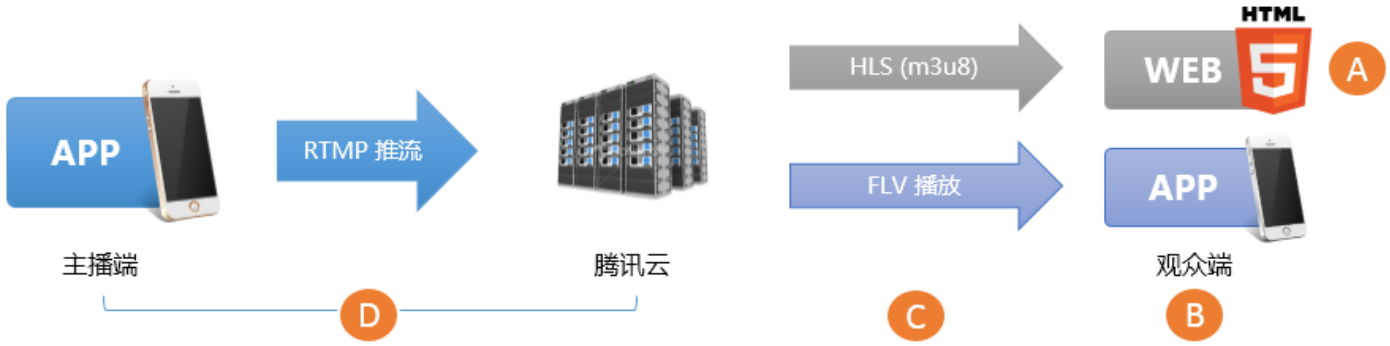
# 小直播推流地址

小直播的推流 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 **startPush**，然后在此处打下调试断点，这里是小直播对 RTMP SDK 的调用点，startPush 的参数即为推流URL。

# 为何直播看不了

最近更新时间：2018-07-11 12:06:03

如果您发现直播无法观看，是不是感觉非常的无奈？是不是觉得腾讯云就是个**黑盒子**，完全搞不懂里面出了什么情况？
不着急，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：




## step 1. 检查播放URL

对，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推流地址 | rtmp://6666.livepush.myqcloud.com/live/6666_xxxxxxxxxxx?bizid=6666 |
| RTMP播放地址 | rtmp://6666.liveplay.myqcloud.com/live/6666_xxxxxxxxxxx |
| FLV播放地址（荐） | http://6666.liveplay.myqcloud.com/live/6666_xxxxxxxxxxx.flv |
| HLS播放地址 | http://6666.liveplay.myqcloud.com/6666_xxxxxxxxxxx.m3u8 |

**小直播的播放URL**

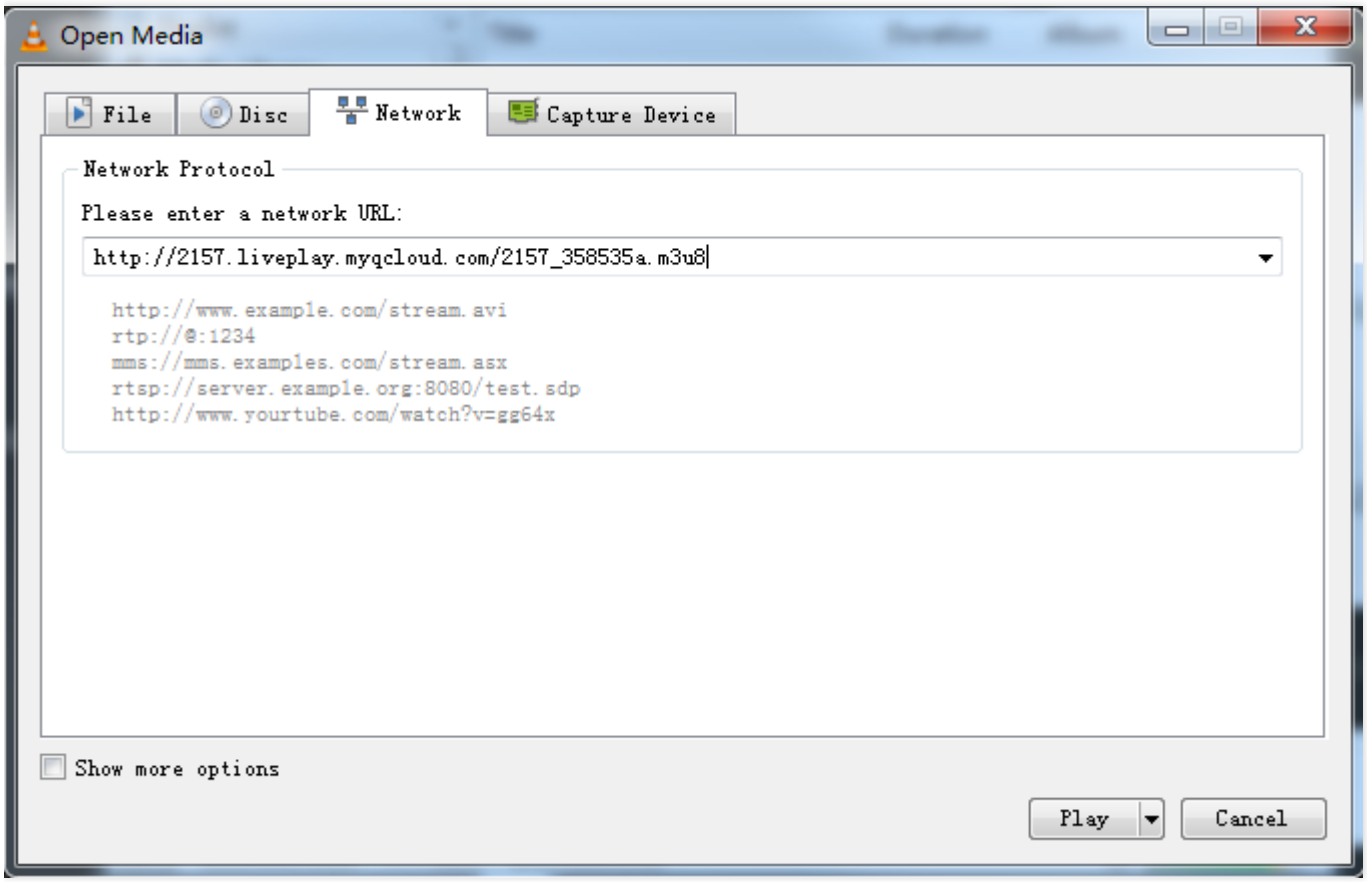
小直播的播放 URL 可以用调试的办法获取，您可以全局搜索代码寻找关键字 **startPlay**，然后在此处打下调试断点，这里是小直播对 RTMP SDK 的调用点，startPlay 的参数即为播放URL。

# step 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就不能观看了。
- 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：




# step 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OK：

### 3.1 Web浏览器（A）

- **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。

### 3.2 RTMP SDK（B）

如果 RTMP SDK DEMO 本身播放没有问题，推荐你参考RTMP SDK 的播放文档（iOS & Android）检查一下对接逻辑是否错误？

## step 4. 防火墙拦截（C）

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（公司老板都不希望员工上班看视频吗）。

如果您使用 4G 进行直播观看没有问题，而是用公司的WiFi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TA给您的IP做一下特殊处理。

## step 5. 检查推流端（D）

如果是直播 URL 根本不能播放，而且没有 Step4 中防火墙限制的可能，那么很大概率是推流不成功，可以到 为何推流不成功 继续问题的排查。

# 如何实现好的画质

最近更新时间：2018-07-11 12:06:18

## 场景一：美女直播

### Step1: 更新最新版SDK

每一版 SDK 我们都或多或少的对美颜效果进行一定的提升，比如：

- 1.9.1 我们更新了新的美颜引擎，在前景侧重、算法优化、曝光改善等方面做了不少改进，同时性能也有很大的提升。
- 1.9.2 我们优化了降噪效果，使夜间的噪点大幅减少，人物的清晰度也有提升
- 2.0.0 我们又为 iOS 增加了一些新的滤镜，来解决肤色过黄的问题
- ... ...

### Step2: 设置画质级别

我们要知道直播端看到的画面跟观众端看到的画面是不一样的：

- **主播 vs 观众**：
  主播看到的画面是经过采集后的视频直接渲染到了手机屏幕上，所以清晰度最高，该画面还要经过 **视频编码 => 网络传输 => 视频解码**，才能到达观众的手机屏幕，由于视频编码会导致画质的损失，所以观众看到的画面在清晰度上要逊于主播端。

  不合理的参数设置会导致画质损失严重，比较典型的一个错误配置就是“高分辨率配低码率”，这样的配置会导致画面模糊，并且画面马赛克严重。怎么办？

- **setVideoQuality**
  我在 1.9.1 版本的 TXLivePusher 中新加入的 setVideoQuality 函数，并给出了几个级别的设置，您选择 **高清** 模式即可达到最佳的美女直播效果。详情请关注 iOS平台 & Android平台。

### Step3: Android 增加手动曝光

同样美颜算法，在不同的 Android 手机上差异可能很大，究其原因，主要是曝光的差异产生了视觉感受的大不同。

在 iOS 平台我们采用了系统的自动曝光，但是 Android 机型差异太大，很多千元机的自动曝光效果实在一般，所以我们还是推荐在您的 UI 界面上提供一个自动曝光的操作滑竿，让主播可以自己调节曝光值大小，以符合他/她的预期。

Android 版 RTMP SDK 中的 TXLivePush::**setExposureCompensation** 接口提供了调节曝光的能力，参数 value 为 -1 到 1 的浮点数： 0 表示不调整， -1 是将曝光降到最低， 1 表示是将曝光加强到最高。


### Step4: 增加色彩滤镜

滤镜也是很关键的一环，不同的色彩滤镜能够营造不同的氛围，主播配合衣服的颜色或者房间灯光选择合适的滤镜，会让整个画面的氛围有更好的效果。


1.9.1 版本的 RTMP SDK 开始支持颜色滤镜， TXLivePusher 中新加入的 setFilter 可以设置滤镜效果，我们设计师团队提供了八种素材，默认打包在了 Demo 中，您可以随意使用，不用担心版权问题。

### Step5: Android马赛克严重

有些客户会发现 Android 版 RTMP SDK 推流出来的画面马赛克特别严重，尤其是在画面有运动时，这是 Android 硬件编码的常规表现，解决方案有两个：

- **我比较关注耗电！**
  如果您比较关注APP耗电情况，那么可以考虑直接调大推流的码率，或者使用 setVideoQuality 的 **高清** 档位（如果设定的码率比较低，Android 的硬编码模块会通过大幅降低画质来确保码率稳定）

- **我比较关注带宽！**
  如果您比较关注带宽成本，那么直接调高码率可能并不适合您的需求，这时您可以通过关闭硬件加速解决问题（关闭方法见 setHardwareAcceleration）。

### Step6: 关闭网络自适应

**TXLivePushConfig** 中的 **AutoAdjustBitrate** 选项是网络自适应开关，开启后，当主播网络变差时，会通过降低画质来保证流畅性，但这一点 并不适合 美女秀场模式。

网络自适应是适合游戏直播场景的技术，因为游戏直播场景中观众对流畅性的追求高于画质，如果在战斗时间主播网络有波动，画质可以渣一点，但绝对不能卡，所以降低画质来保证流畅性（帧率）就是必选项了。

但美女秀场场景中，清晰度更加重要，不少客户反馈有些直播间画质很好，有些直播间画质很差，导致这个现象的一个高概率原因就是开启了网络自适应。

我们推荐关闭网络自适应，对于出现网络波动的情况，改用 主动提示 的方式进行优化，这种方式更加釜底抽薪。

# 场景二：游戏直播

### Option1: 简单应对

在直播开始的界面上提供三种清晰度选项：标清、高清和超清，**让主播自己选择**，游戏主播一般相对比较偏专业，会自己摸索出适合自己当前在玩的游戏的效果，三种档位对应的配置如下：


| 档位 | 分辨率（Resolution） | FPS（FPS） | 码率（Bitrate） |
| --- | --- | --- | --- |
| 标清 | VIDEO_RESOLUTION_TYPE_360_640 | 20 | 800kbps |
| 高清 | VIDEO_RESOLUTION_TYPE_540_960 | 20 | 1000kbps |
| 超清 | VIDEO_RESOLUTION_TYPE_720_1280 | 20 | 1800kbps |

> 注意： 游戏直播场景，FPS 最低是 20 ，不能更低了，否则观众端的表现是卡顿感严重。

### Option2: 专业应对

根据不同的游戏设置不同的分辨率和码率，这就比较耗功夫和时间了，比如：

- **皇室战争** - 这类画面变化幅度不大的游戏，推荐选择 **960 * 540** 的分辨率，800kbps-1000kbps 的码率就可以输出不错的效果。
- **捕鱼达人** - 这类画面变化幅度较大的游戏，推荐选择 **960 * 540** 的分辨率，码率相对要高一点，比如1200kbps - 1500kbps。
- **神庙逃跑** - 这类画面变化幅度超大的游戏，推荐选择 **640 * 360** 的分辨率，码率也要很大，比如 2000kbps，否则妥妥的满屏马赛克。

## 音视频小科普

### Point1：720P一定更清楚吗？

如果限定一个码率，比如 800kbps，那么**分辨率越高就会让编码器越难做**。

可以想象，编码器必须拆东墙补西墙，通过减少色彩信息或者引入马赛克这种"鱼目混珠"的手段来承载足够多的像素点。所以，同样的是2G的一个电影文件，1080p画质的版本可能不如720p画质的版本看起来更清晰。

同时，如果你的观众都是在小手机屏幕上观看视频，那么 **960 * 540 1000kbps** 同 **1280 * 720 1800kbps** 的差距其实也不明显，比如下面两张图片就是基于 airplay 技术的 iOS 录屏直播，您看得出左边和右边的清晰度差距吗？


用 32 寸的 LCD 显示器全屏看，还是有差距的。

### Point2：帧率不要超过24！

如果限定一个码率，比如800kbps，那么帧率越高，编码器就必须加大对单帧画面的压缩比，也就是通过降低画质来承载足够多的帧数。如果视频源来自摄像头，24FPS已经是肉眼极限，所以一般20帧的FPS就已经可以达到很好的用户体验了。

有些玩过3D游戏的朋友可能会说："游戏的帧率越高越流畅吗？比如 60FPS， 120FPS?"

这里要注意一定不要混淆场景：游戏追求高帧率是**渲染帧率**，其目的是为了尽可能让3D模型渲染出来的运动效果更加接近真实运动轨迹，所以帧率越高越好。

但采集帧率不需要这么高，比如手机摄像头，它采集的目标是真实世界的物体，真实世界的物体本来就是连续运动的，而不是用一阵阵画面刷新来模拟的，所以 20FPS 的采集就足以。

对于游戏直播，帧率达到 24FPS 当然最好，但也要考虑到系统编码开销，手机的温度和CPU占用等等，毕竟主播还要用同样的一台手机玩游戏呢。

# 如何开通各项云服务

最近更新时间：2018-07-11 12:02:54

## 1. 视频直播（LVB）

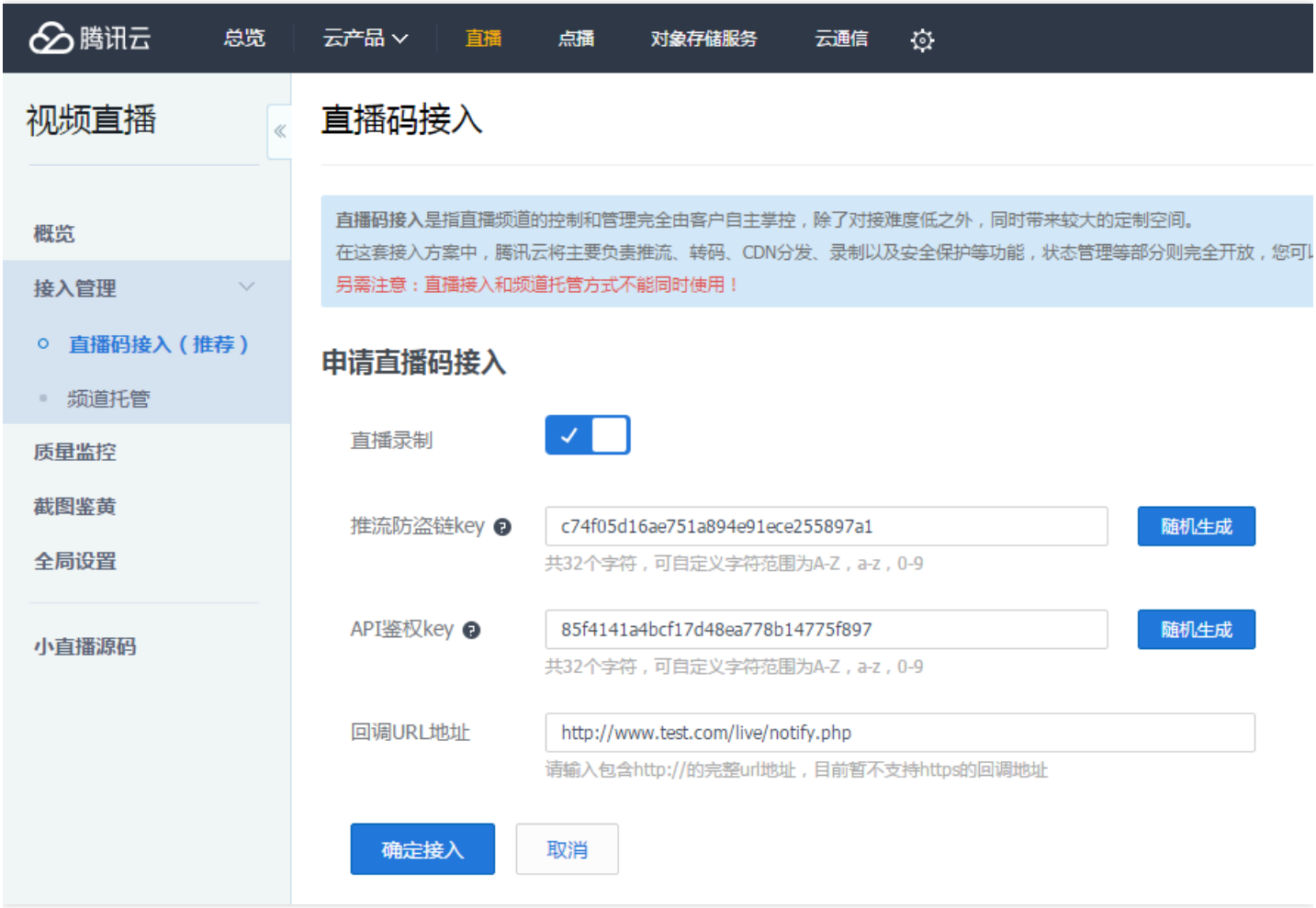
### 1.1 如何开通视频直播服务

进入直播管理控制台，如果服务还没有开通，则会有如下提示:


点击申请开通，之后会进入腾讯云人工审核阶段，审核通过后即可开通。

### 1.2 如何开启直播码接入?

直播服务开通后，进入直播控制台>>直播码接入>>接入配置 完成相关配置，即可开启直播码服务：




| 配置项 | 取值范围 | 具体含义 |
| --- | --- | --- |
| 直播录制 | 开启 OR 关闭 | 开启直播录制后，只要是直播的视频，都会被后台默认的录制下来。DOC |
| 推流防盗链key | 32位小写的字符串 | 出于安全考虑，推流地址需要绑定防盗链以避免被其他人抢占，该KEY用于计算防盗链签名。DOC |
| API鉴权key | 32位小写的字符串 | 您的服务器在调用腾讯云后台API时，需要提供身份鉴权信息，该KEY用于帮助腾讯云确定您的服务器的合法身份。DOC |
| 回调URL | HTTP协议URL地址 | 腾讯云会把视频的推流断流等事件通知通过这个URL投递给您，尚不支持HTTPS协议。DOC |

点击**确定接入**按钮，即可将您的腾讯云直播服务切换到直播码模式了。

### 1.3 直播 APPID

每个腾讯云账号在开通视频直播服务后，均会分配到一个直播APPID，直播APPID在直播管理控制台 的顶部区域可见，如下图：


### 1.4 直播 BIZID

每个腾讯云账号在开通视频直播服务后，均会分配到一个直播BIZID，直播BIZID主要用于直播码里的推流URL和播放URL的拼装，因此在开启直播码模式后，即可在直播管理控制台 的顶部区域可见，如下图：


### 1.5 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，推流防盗链KEY可以在开启直播码时指定，亦可随时按您的需要在直播管理控制台中修改：


## 1.6 API鉴权KEY

API访问鉴权KEY是您的后台服务器在调用腾讯云直播码相关的云端API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可以在开启直播码时指定，亦可随时按您的需要在直播管理控制台中修改：


## 1.7 事件通知URL

事件回调URL是来自您的后台服务器的一个地址，当腾讯云有一些直播相关事件需要通知您的后台服务器时，会以HTTP POST的形式向该地址发送通知，事件回调URL可以在开启直播码时指定，亦可随时按您的需要在直播管理控

制台中修改：


# 2. 视频点播（VOD）

## 2.1 如何开通视频点播服务

每个新认证的腾讯云账号都有7天的免费试用，试用过后，只需在点播管理控制台 购买适合您的套餐即可：


## 2.2 查询我的点播APPID

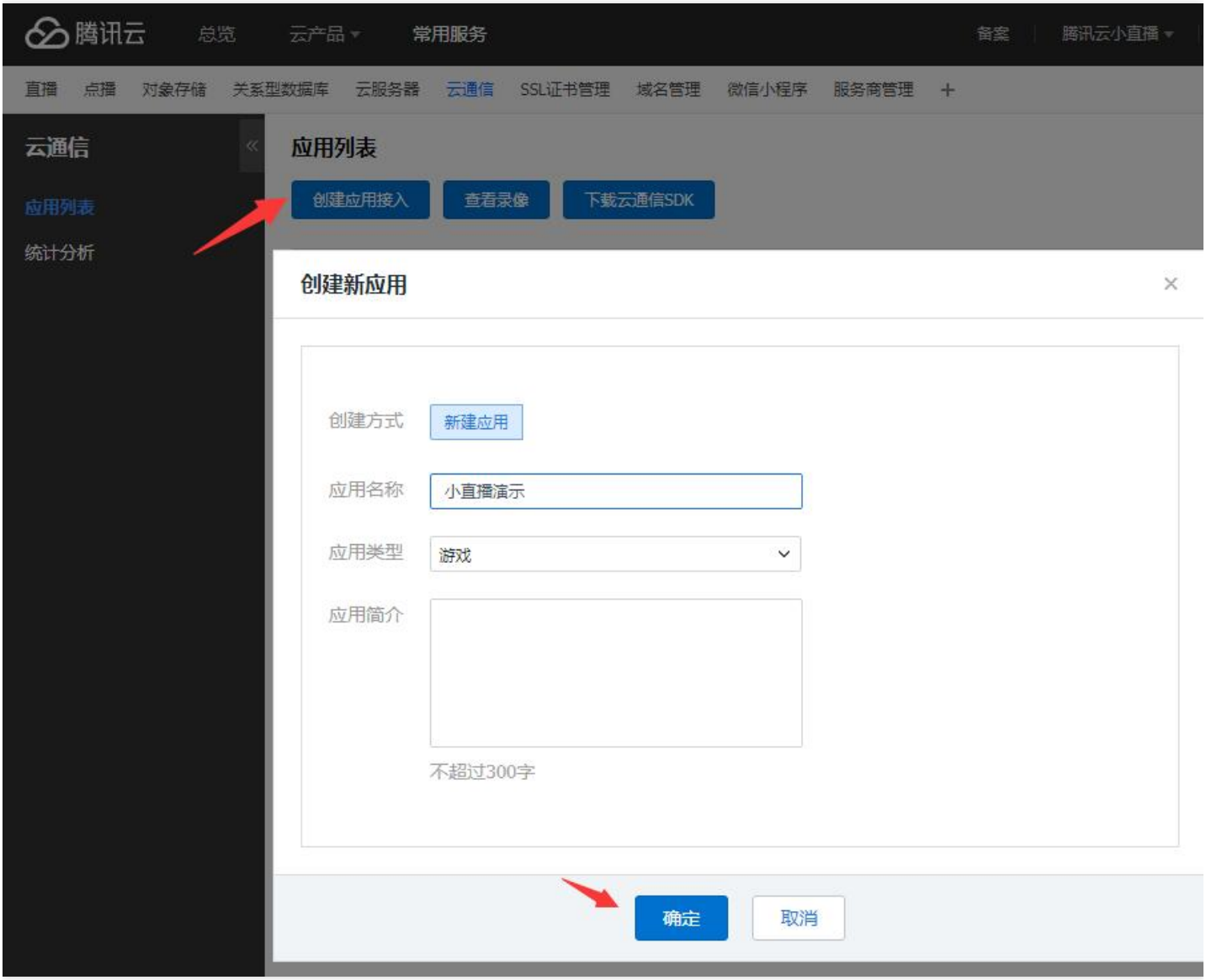
每个腾讯云账号均有一个唯一的点播APPID与之对应，其位于点播管理控制台 里的一个比较隐蔽的位置，需要的您的账号下至少存在一个上传或者录制的视频文件，查看方案如下：


# 3. 云通讯服务（IM）

## 3.1 如何开通 IM 服务

进入云通讯管理控制台，直接点击**开通云通讯**按钮即可。

新认证的腾讯云账号，云通讯的应用列表是空的，点击**创建应用接入**按钮即可创建一个新的应用：




### 3.2 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，可以有多个 sdkappid。


### 3.3 AccountType

上图的列表中，右侧有一个**应用配置**按钮，点击这里进入下一步的配置工作，如下图所示：


**账号名称**，可以随便填写，但请尽量使用英文字符和数字。**账号管理员** 在调用 IM 的 REST API 时才会用到。

点击**保存**按钮之后，即完成了对App的IM消息接入配置，AccountType(账号类型) 也就生成了，如下图：


### 3.4 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，比如建群、发送系统消息、把某个用于剔除群组等等，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)，具体使用过程详见 DOC。


### 3.5 PrivateKey & PublicKey

您可以把 IM SDK 理解为一个没有用户交互界面的 QQ，把 IM SDK 集成到您的 APP 里，就相当于把一个 QQ 的消息内核集成在您的 APP 内部。

我们都知道，QQ 可以用来收发单聊和群聊的消息，但前提是您必须先登录才能使用。我们也都知道，登录 QQ 用的是 QQ 号和密码，登录 IM SDK 也是一样，只是肯定不能再用 QQ 号和密码了，而是使用您指定的用户名（userid）和密码（usersig）。

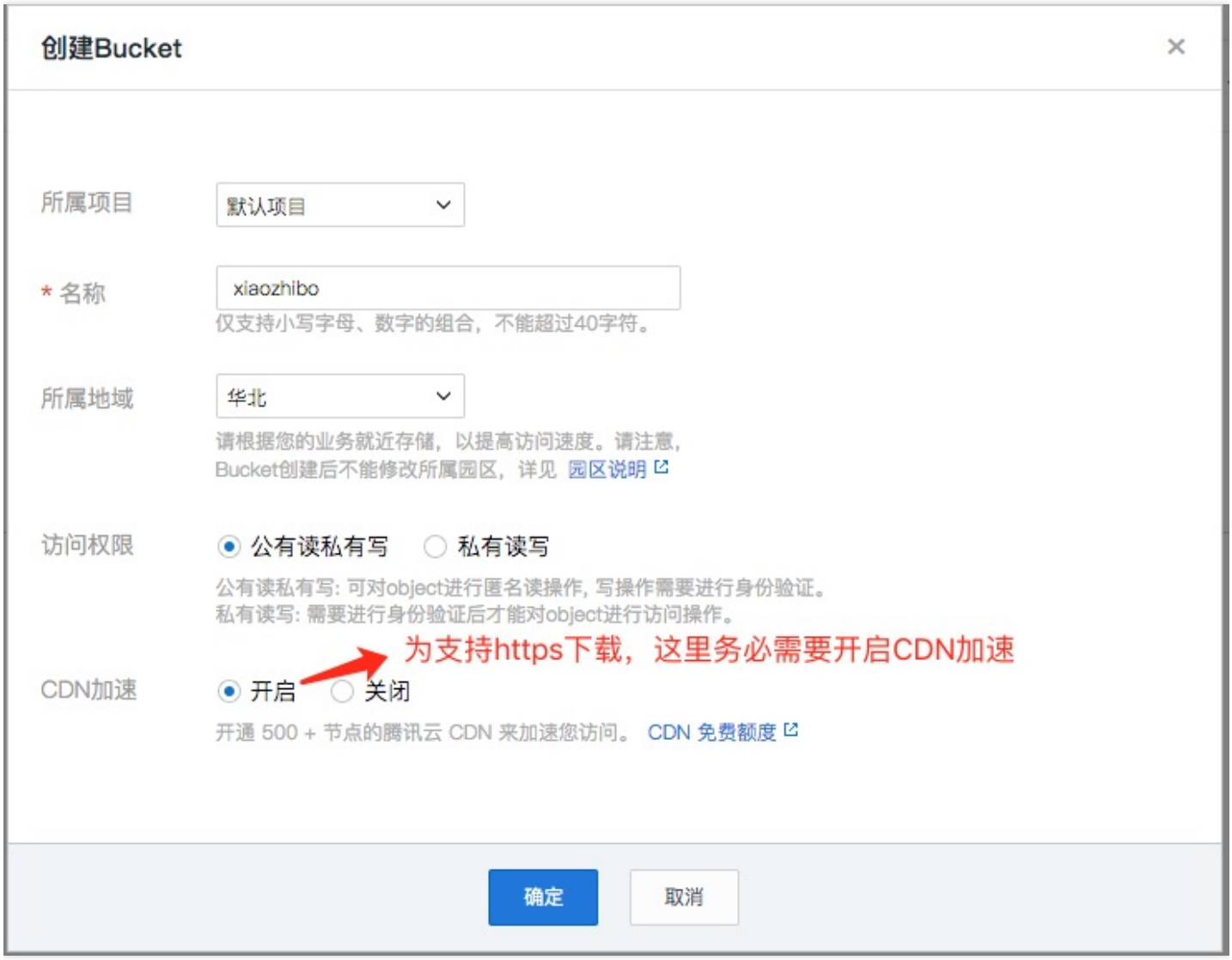
用户名您可以随意指定，但是腾讯云需要确认这个用户名是否合法，这里就需要用到非对称加密技术。非对称加密用的加密密钥和解密密钥是不同的，您的服务器可以持有私钥（privateKey）并对 userid 和 appid 进行非对称加密，加密之后的结果就是 usersig 签名；而腾讯云同步持有对应的公钥（publicKey），这样一来，腾讯云就可以确认 usersig 是否合法，从而可以确认它是否是由您的服务器签发的。

这里计算 usersig 签名用的公私钥，可以在如下位置下载获得：


# 4. 对象存储服务（COS）

## 4.1 如何开通云通讯服务

新认证的腾讯云账号都可以立刻使用对象存储服务，只需要进入COS管理控制台创建爱你一个Bucket即可开始启用，**请注意为支持https下载（适配苹果的ATS），请开启CDN加速。**




### 4.2 什么是Bucket（桶）

Bucket是一个略显技术化的名词，翻译成中文即"桶"的意思，您可以简单将它理解为**磁盘分区**的概念。打个比方，您在腾讯云购买了对象存储服务(COS)，可以比作您在京东商城购买了一块新的硬盘，在您向上面存储数据之前，一般会先对齐进行分区和格式化。那么我们可以这样说，您在真实世界里的硬盘上建一个分区，就好比在腾讯云的COS里建一个Bucket。

### 4.3 查询我的Bucketname

您在创建Bucket时所指定的名称即为您的一个bucketname，比如在4.1的例子中我所指定的xiaozhibo即是一个bucketname。

### 4.4 查询我的COS APPID

点击COS管理控制台的密钥管理标签页，即可获得COS的APPID了，它是跟一对API必要绑定在一起的。


### 4.5 查询我的COS SecretId 和 SecretKey

点击COS管理控制台的密钥管理标签页，即可获得跟COS APPID 绑定的SecretId 和 SecretKey了，他们主要用于访问COS的API，因为COS是一个对安全要求很高的云端服务，所以如果API没有传入正确的密钥，腾讯云会拒绝这些API请求。


# 5. 云服务器（可选）

您可以使用您自己的服务器作为业务服务器部署后台脚本，不过推荐您使用腾讯云的云服务器部署后台脚本，更为专业和稳定，另外如果您选择腾讯云的云数据库作为分布式数据库，则必须搭配腾讯云的云服务器才能访问。

进入云服务器管理控制台，点击"购买云服务器"，将进入服务器购买页面：


点击下一步后，进入镜像选择页面，推荐您从服务器市场中选择带有nginx+php+mysql的linux镜像：


后续操作按照指引即可，镜像按照完毕后即可使用云服务器。

# 6. 云数据库（可选）

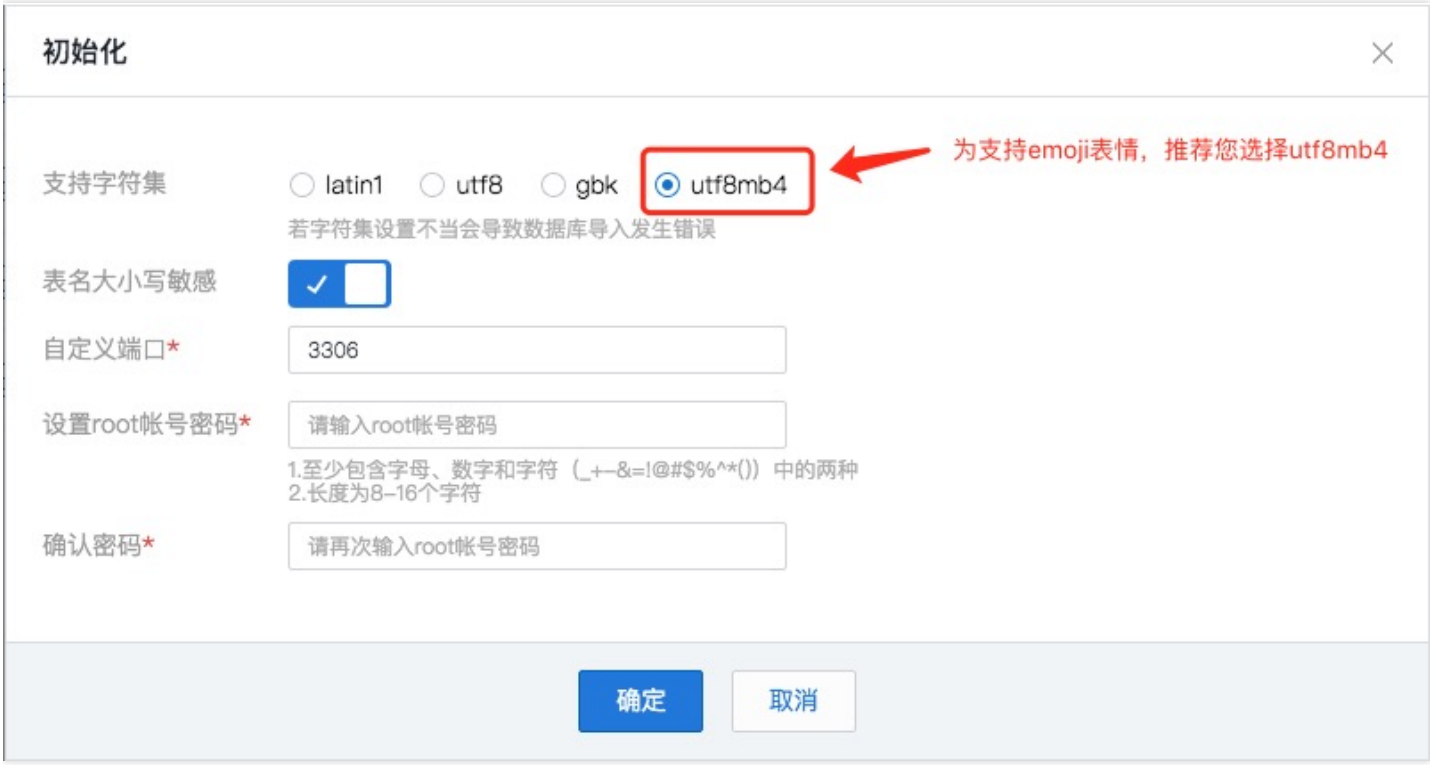
## 6.1 如何开通云数据库

进入云数据库管理控制台，如果您没开通过CDB(MYSQL)服务，可直接点上面的"新建"按钮：


**注意：必须用开通云服务器相同的腾讯云账号开通云数据库服务，并且需要选择与云服务器相同的区域**

购买完成后，在"实例列表"里可以看到该实例，点击右边的"初始化"设置数据库的字符集和密码：


MySQL-实例列表
全部项目
全部地域
云数据库MySQL支持按量计费阶梯价，用得越久越便宜。内存和硬盘可按需搭配。了解详情
+ 新建
对比监控
续费
更多操作
请输入IP(换行分隔)或实例名
ID/实例名
监控
状态
实例类型
所属项目
所属地域
配置类型
操作
cdb-99tcm5zm
cdb125093
运行中
主实例
默认项目
华南地区（广州）
高IO版
登录 管理
cdb-n2rqdle9
cdb125029
未初始化
主实例
默认项目
华南地区（广州）
高IO版
初始化
管理


初始化
支持字符集
latin1
utf8
gbk
utf8mb4
为支持emoji表情，推荐您选择utf8mb4
若字符集设置不当会导致数据库导入发生错误
表名大小写敏感
自定义端口*
3306
设置root帐号密码*
请输入root帐号密码
1.至少包含字母、数字和字符（_+-&=!@#$%^*()）中的两种
2.长度为8-16个字符
确认密码*
请再次输入root帐号密码
确定
取消

## 6.2 如何使用数据库

实例初始化完成后，在实例列表中可以看到该实例的内网地址：


您可以在云服务器上使用mysql命令远程连接该实例，并进行数据库操作，也可以在实例列表中点“管理”进入管理页面操作数据库：

# 如何录制并回看

最近更新时间：2018-07-11 11:54:43

## 功能介绍

所谓录制回看，是指您可以把用户整个直播过程录制下来，然后作为点播视频用于回看。

在APP上线的初期阶段，由于主播数量比较少，所以在直播列表中加入录制回看，能够在一定程度上丰富APP在观众端的信息量。

即使到APP成长起来主播数量形成规模以后，好的直播内容的沉淀依然是必不可少的一个部分，每个主播的个人介绍里除了有名字、照片和个人信息，历史直播的视频回看更是不可或缺的一个重要组成部分。

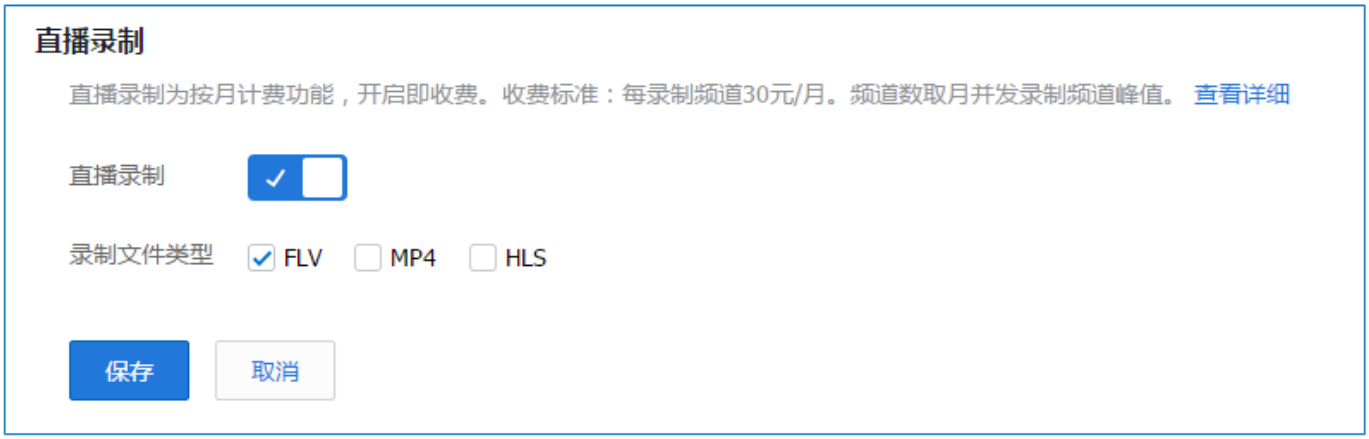

## 开启录制

录制回看功能依托于腾讯云的**点播服务**支撑，如果您想要对接这个功能，首先需要在腾讯云的管理控制台开通点播服务。服务开通之后，新录制的文件就可以在点播控制台的 视频管理 里找到它们。

那么怎么开启录制呢？这里有两个办法：

### 1. 全局开启录制

即指定所有直播的视频流全部开启或者关闭录制，在 直播管理控制台 >> 接入管理 >> 直播码接入 >> 接入配置 中可以对其进行设置，见下图：




注意：全局录制的分片时长默认为30分钟，如果您需要配置全局录制的分片时长，可以提工单申请。

### 2. 指定房间录制

在所有直播的视频流全部关闭录制的情况下，您依然可以对个别重要的视频流开启录制，操作方法是在推流URL后拼接 **&record=mp4** 或者 **&record=hls** 或者 **&record=mp4|hls**，例如：

```
rtmp://2121.livepush.myqcloud.com/live/2121_15919131751?txSecret=aaa&txTime=bbb&record=mp4&record_interval=5400
```

这里有几点要特别说明：

- 目前录制功能支持的视频封装格式有 mp4、 hls 和 flv 三种，关于点播格式的差异，您可以参考这里 DOC。
- record=mp4|hls|flv 分隔符格式用于指定同时录制一种以上的视频格式（只有 MP4 和 HLS 支持手机浏览器播放）。
- mp4 格式的视频是 不支持 直播中途切换分辨率或做横竖屏切换的。

- 如果您指定的录制格式是 flv 或 mp4 ，可以通过 record_interval 参数用于指定单个录制分片的时长，单位是秒，最长支持120分钟（也就是7200），不指定的话默认值是30分钟（也就是1800）。

- hls（m3u8）文件本身就是小分片机制，所以无所谓切断问题，只要直播过程中不断流，您只会拿到一个m3u8文件。但如果直播期间推流出现中断，录制过程依然会出现分段问题（也就是会得到多个m3u8），其中一个常见的问题就是 App 切后台，推荐采用后台推流解决方案进行缓解。

- 如果您想只录制音频，可以通过添加参数 record_type=audio 来指定。

# 获取文件

一个新的录制视频文件生成后，会相应的生成一个观看地址，您可以按照自己的业务需求对其进行处理。在小直播中，我们直接将录制的文件URL和房间列表拼在了一起，以弥补在线主播不足的窘境。

但您可以结合自己的业务场景实现很多的扩展功能，比如：您可以将其追加到主播的资料信息里，作为该主播曾经直播的节目而存在；或者将其放入回放列表中，经过专门的人工筛选，将优质的视频推荐给您的App用户。

那么怎样才能拿到文件的地址呢？有如下两种解决方案：

### 1. 被动监听通知

您可以使用腾讯云的**事件通知服务**：您的服务器注册一个自己的**回调URL**给腾讯云，腾讯云会在一个新的录制文件生成时通过这个URL通知给您。

如下是一个典型的通知消息，它的含义是：一个id为91924872665818215 86的新的flv录制文件已经生成，播放地址为： `http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv` 。

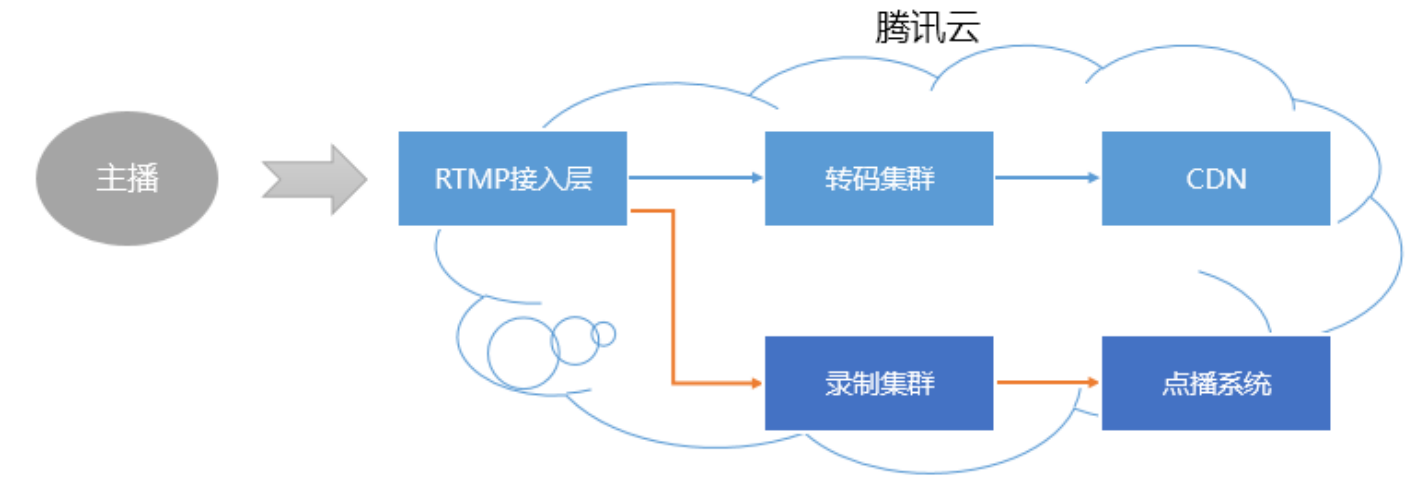
```
{
"channel_id": "2121_15919131751",
"end_time": 1473125627,
"event_type": 100,
"file_format": "flv",
"file_id": "9192487266581821586",
"file_size": 9749353,
"sign": "fef79a097458ed80b5f5574cbc13e1fd",
"start_time": 1473135647,
"stream_id": "2121_15919131751",
"t": 1473126233,
"video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
"video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv"
}
```

### 2. 主动查询文件

您可以通过腾讯云的文件查询接口（**Live_Tape_GetFilelist**）定时地查看是否有新的录制文件生成，不过这种方案在要查询的频道数特别多的时候，响应速度不理想，同时调用频率也不能太快（仅对刚结束的频道进行调用为宜），这种方案的实时性和可靠性不高，并不推荐频繁使用。

## 常见问题

### 1. 直播录制的原理是什么？




对于一条直播流，一旦开启录制，音视频数据就会被旁路到录制系统。主播的手机推上来的每一帧数据，都会被录制系统追加写入到录制文件中。

一旦直播流断中断，接入层会 立刻 通知录制服务器将正在写入的文件落地，将其转存到点播系统中，并为其生成索引，这样您在点播系统中就会看到这个新生成的录制文件了。同时，如果您配置了录制事件通知，录制系统会将该文件的 **索引ID** 和 **在线播放地址** 等信息通知给您之前配置的服务器上。

但是，如果一个文件过大，在云端的转出和处理过程中就很容易出错，所以为了确保成功率，我们的单个录制文件最长不会超过90分钟，您可以通过 record_interval 参数指定更短的分片。

### 2. 一次直播会有几个录制文件？

- 如果一次直播过程非常短暂，比如只有不到 1 秒钟时间，那么可能是没有录制文件的。
- 如果一次直播时间不算长（小于 record_interval），且中途没有推流中断的事情发生，那么通常只有一个文件。
- 如果一次直播时间很长（超过 record_interval ），那么会按照 record_interval 指定的时间长度进行分片，分片的原因是避免过长的文件在分布式系统中流转的时间不确定性。
- 如果一次直播过程中发生推流中断（之后 SDK 会尝试重新推流），那么每次中断均会产生一个新的分片。

### 3. 如何知道哪些文件属于某一次直播？

准确来说，作为 PAAS 的腾讯云并不清楚您的一次直播是怎么定义的，如果您的一次直播持续了20分钟，但中间有一次因为网络切换导致的断流，以及一次手动的停止和重启，那么这算是一次直播还是三次呢？

对于普通的移动直播场景，我们一般定义如下的界面之间的这段时间为一次直播：


所以来自 APP 客户端的时间信息很重要，如果您希望定义这段时间内的录制文件都属于这次直播，那么只需要用直播码和时间信息检索收到的录制通知即可（每一条录制通知事件都会携带stream_id、start_time 和 end_time 等信息）。

### 4. 如何把碎片拼接起来？

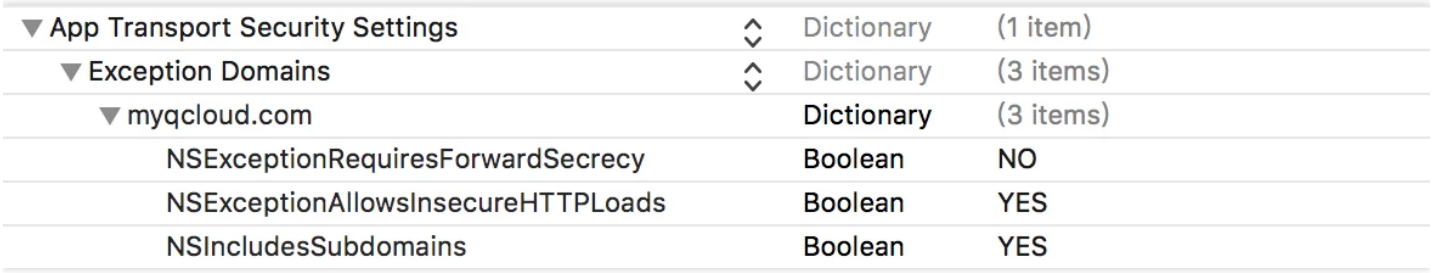
目前腾讯云支持使用云端 API 接口拼接视频分片，API 详细用法可以参考 视频拼接 。

# 如何适配苹果 ATS

最近更新时间：2018-07-11 12:06:40

苹果在WWDC 2016中表示，从2017年1月1日起，所有新提交的app默认不能使用 `NSAllowsArbitraryLoads=YES` 来绕过ATS的限制。腾讯云将于12月12日正式支持HTTPS，届时您只需要使用新版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，还要需要继续使用HTTP，方法修改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。

# 事件通知码

最近更新时间：2018-07-10 14:28:54

## PushEvent

### 常规事件

每一次成功的推流都会有通知事件，比如收到 1003 就意味着摄像头的画面开始渲染。

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| 1001 | PUSH_EVT_CONNECT_SUCC | 已经成功连接到云端服务器 |
| 1002 | PUSH_EVT_PUSH_BEGIN | 与服务器握手完毕,一切正常，准备开始上行推流 |
| 1003 | PUSH_EVT_OPEN_CAMERA_SUCC | 已成功启动摄像头，摄像头被占用或者被限制权限的情况下无法打开 |
| 1004 | PUSH_EVT_SCREEN_CAPTURE_SUCC | 录屏启动成功 （ Android SDK 专用 ） |
| 1005 | PUSH_EVT_CHANGE_RESOLUTION | 推流动态调整分辨率 |
| 1006 | PUSH_EVT_CHANGE_BITRATE | 推流动态调整码率 |
| 1007 | PUSH_EVT_FIRST_FRAME_AVAILABLE | 首帧画面采集完成 |
| 1008 | PUSH_EVT_START_VIDEO_ENCODER | 编码器启动 |
| 1009 | PUSH_EVT_CAMERA_REMOVED | 摄像头设备已被移出（ Windows SDK 专用 ） |
| 1010 | PUSH_EVT_CAMERA_AVAILABLE | 摄像头设备重新可用（ Windows SDK 专用 ） |
| 1011 | PUSH_EVT_CAMERA_CLOSED | 关闭摄像头完成（ Windows SDK 专用 ） |
| 1012 | PUSH_EVT_SNAPSHOT_RESULT | 截图快照返回码（ Windows SDK 专用 ） |
| 1018 | PUSH_EVT_ROOM_IN | 已经在 webrtc 房间里面，进房成功后通知 |
| 1019 | PUSH_EVT_ROOM_OUT | 不在 webrtc 房间里面，进房失败或者中途退出房间时通知 |
| 1020 | PUSH_EVT_ROOM_USERLIST | 下发 webrtc 房间成员列表(不包括自己) |
| 1021 | PUSH_EVT_ROOM_NEED_REENTER | Wi-Fi 切换到 4G 会触发断线重连，此时需要重新进入 webrtc 房间(拉取最优的服务器地址) |

### 严重错误

SDK 发现了一些严重问题，推流无法继续了，比如用户禁用了 App 的 Camera 权限导致摄像头打不开。

> 注：视频编码失败并不会直接影响推流，SDK 会做自行处理以保证后面的视频编码成功。

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| -1301 | PUSH_ERR_OPEN_CAMERA_FAIL | 打开摄像头失败 |
| -1302 | PUSH_ERR_OPEN_MIC_FAIL | 打开麦克风失败 |
| -1303 | PUSH_ERR_VIDEO_ENCODE_FAIL | 视频编码失败 |
| -1304 | PUSH_ERR_AUDIO_ENCODE_FAIL | 音频编码失败 |
| -1305 | PUSH_ERR_UNSUPPORTED_RESOLUTION | 不支持的视频分辨率 |
| -1306 | PUSH_ERR_UNSUPPORTED_SAMPLERATE | 不支持的音频采样率 |
| -1307 | PUSH_ERR_NET_DISCONNECT | 网络断连，且连续三次无法重新连接，需要自行重启推流 |
| -1308 | PUSH_ERR_CAMERA_OCCUPY<br>PUSH_ERR_AUDIO_SYSTEM_NOT_WORK<br>PUSH_ERR_SCREEN_CAPTURE_START_FAILED | 摄像头正在被占用中，可尝试打开其他摄像头（Windows）<br>系统异常，录音失败（iOS）<br>开始录屏失败，可能是被用户拒绝了（Android） |
| -1309 | PUSH_ERR_SCREEN_CAPTURE_UNSURPORT | 录屏失败，不支持的 Android 系统版本，需要 5.0 以上的系统（Android SDK 专用） |

### 警告事件

大部分的警告事件会触发一些重试性的保护逻辑或者恢复逻辑，很大概率能够由SDK自行恢复。部分警告事件需要由开发者处理。

- **WARNING_NET_BUSY：**主播网络不给力，可以通过 UI 向用户进行提示。
- **WARNING_SERVER_DISCONNECT：**推流请求被后台拒绝了，出现这个问题一般是由于推流地址里的 txSecret 计算错了，或者是推流地址被其他人占用了（一个推流 URL 同时只能有一个端推流）。

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| 1101 | PUSH_WARNING_NET_BUSY | 上行网速不够用，建议提示用户改善当前的网络环境 |
| 1102 | PUSH_WARNING_RECONNECT | 网络断连，已启动重连流程（重试失败超过三次会放弃） |
| 1103 | PUSH_WARNING_HW_ACCELERATION_FAIL | 硬编码启动失败，自动切换到软编码 |
| 1104 | PUSH_WARNING_VIDEO_ENCODE_FAIL | 视频编码失败，非致命错，内部会重启编码器 |
| 1107 | PUSH_WARNING_VIDEO_ENCODE_SW_SWITCH_HW | 由于机器性能问题，自动切换到硬件编码（Android SDK 专用） |
| 3001 | PUSH_WARNING_DNS_FAIL | DNS 解析失败，启动重试流程 |
| 3002 | PUSH_WARNING_SEVER_CONN_FAIL | 服务器连接失败，启动重试流程 |
| 3003 | PUSH_WARNING_SHAKE_FAIL | 服务器握手失败，启动重试流程 |
| 3004 | PUSH_WARNING_SERVER_DISCONNECT | RTMP 服务器主动断开，请检查推流地址的合法性或防盗链有效期 |
| 3005 | PUSH_WARNING_READ_WRITE_FAIL | RTMP 读/写失败，将会断开连接 |

# PlayEvent

**关键事件**

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| 2001 | PLAY_EVT_CONNECT_SUCC | 已经连接服务器 |
| 2002 | PLAY_EVT_RTMP_STREAM_BEGIN | 已经连接服务器，开始拉流 |
| 2003 | PLAY_EVT_RCV_FIRST_I_FRAME | 渲染首个视频数据包(IDR) |
| 2004 | PLAY_EVT_PLAY_BEGIN | 视频播放开始 |
| 2005 | PLAY_EVT_PLAY_PROGRESS | 视频播放进度(点播专用) |
| 2006 | PLAY_EVT_PLAY_END | 视频播放结束 |
| 2007 | PLAY_EVT_PLAY_LOADING | 视频播放加载中 |

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| 2008 | PLAY_EVT_START_VIDEO_DECODER | 解码器启动 |
| 2009 | PLAY_EVT_CHANGE_RESOLUTION | 视频分辨率改变 |
| 2010 | PLAY_EVT_GET_PLAYINFO_SUCC<br>PLAY_EVT_SNAPSHOT_RESULT | 获取点播文件信息成功（Android iOS）<br>系截图快照返回码（Windows） |
| 2011 | PLAY_EVT_CHANGE_ROATION | MP4 视频旋转角度（Android、 iOS） |
| 2012 | PLAY_EVT_GET_MESSAGE | 用于接收夹在音视频流中的消息，详情参考 iOS 消息接收、Android 消息接收 |
| 2013 | PLAY_EVT_PREPARED | 视频加载完毕（Android、 iOS） |
| 2014 | PLAY_EVT_VOD_LOADING_END | 加载结束（Android、 iOS） |
| -2301 | PLAY_ERR_NET_DISCONNECT | 网络断连，且连续三次无法重新连接，需要自行重启推流 |
| -2302 | PLAY_ERR_GET_RTMP_ACC_URL_FAIL | 获取加速拉流地址失败 |
| -2303 | PLAY_ERR_FILE_NOT_FOUND | 播放文件不存在（Android、 iOS） |
| -2304 | PLAY_ERR_HEVC_DECODE_FAIL | H265 解码失败（Android、 iOS） |
| -2305 | PLAY_ERR_HLS_KEY | HLS 解码 key 获取失败（Android、 iOS） |
| -2306 | PLAY_ERR_GET_PLAYINFO_FAIL | 获取点播文件信息失败（Android、 iOS） |

**注意：**

- **判断直播是否结束：**基于各种标准的实现原理不同，很多直播流通常没有结束事件（2006）抛出，此时可预期的表现是：主播结束推流后，SDK 会很快发现数据流拉取失败（WARNING_RECONNECT），然后开始重试，直至三次重试失败后抛出 PLAY_ERR_NET_DISCONNECT 事件。所以 2006 和 -2301 都要判断，用来作为直播结束的判定事件。
- **不要在收到 PLAY_LOADING 后隐藏播放画面：**因为 PLAY_LOADING 到 PLAY_BEGIN 的时间长短是不确定的，可能是 5s 也可能是 5ms，有些客户考虑在 LOADING 时隐藏画面， BEGIN 时显示画面，会造成严重的画面闪烁（尤其是直播场景下）。推荐的做法是在视频播放画面上叠加一个半透明的加载动画。

### 警告事件

内部警告并非不可恢复的错误，SDK 会启动相应的恢复措施，警告的目的主要用于提示开发者或者最终用户。

| code | 事件定义 | 含义说明 |
| --- | --- | --- |
| 2101 | PLAY_WARNING_VIDEO_DECODE_FAIL | 当前视频帧解码失败 |
| 2102 | PLAY_WARNING_AUDIO_DECODE_FAIL | 当前音频帧解码失败 |
| 2103 | PLAY_WARNING_RECONNECT | 网络断连，已启动自动重连恢复 (重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT 了) |
| 2104 | PLAY_WARNING_RECV_DATA_LAG | 视频流不太稳定，可能是观看者当前网速不充裕 |
| 2105 | PLAY_WARNING_VIDEO_PLAY_LAG | 当前视频播放出现卡顿 |
| 2106 | PLAY_WARNING_HW_ACCELERATION_FAIL | 硬解启动失败，采用软解 |
| 2107 | PLAY_WARNING_VIDEO_DISCONTINUITY | 当前视频帧不连续，视频源可能有丢帧，可能会导致画面花屏 |
| 2108 | PLAY_WARNING_FIRST_IDR_HW_DECODE_FAIL | 当前流硬解第一个 I 帧失败，SDK 自动切软解 |
| 3001 | PLAY_WARNING_DNS_FAIL | DNS 解析失败（仅播放 RTMP:// 地址时会抛送） |
| 3002 | PLAY_WARNING_SEVER_CONN_FAIL | 服务器连接失败（仅播放 RTMP:// 地址时会抛送） |
| 3003 | PLAY_WARNING_SHAKE_FAIL | 服务器握手失败（仅播放 RTMP:// 地址时会抛送） |
| 3004 | PLAY_WARNING_SERVER_DISCONNECT | RTMP 服务器主动断开 |