云点播

播放器 SDK 文档

产品文档

腾讯云

文档目录

播放器 SDK 文档

概述

最近更新时间：2021-10-21 16:57:27

产品说明

腾讯云视立方播放器 Player（以下简称播放器 Player）是腾讯云提供给用户，为点播业务提供全面、稳定、流畅的视频播放服务，帮助用户连接云端服务、打造云端一体化能力。播放器 Player 在点播播放场景提供了超级播放器和超级播放器 Adapter 两种播放器类型，同时支持 Web 端、iOS 端、Android 端、Flutter 端等多个平台。此外，云点播为客户提供多种视频播放解决方案，赋能客户的不同场景，满足客户多样化需求。

快速了解

为了保证用户可以快速了解播放器 Player，在正式使用超级播放器和超级播放器 Adapter 前，建议所有客户首先阅读播放器的 基本概念 并通过 如何选择点播播放器 快速选择匹配自身业务的播放器类型。

核心优势

云端一体化服务

超级播放器融合强大的云点播音视频服务能力，打造功能完备的云端一体化能力，为用户业务提供更有价值的业务运营能力。

全方位视频安全

超级播放器支持防盗链、URL 鉴权、HLS 加密、私有协议加密、离线下载等视频安全方案，同时支持动态水印等视频安全能力，全方位保证用户媒体安全，满足业务不同场景的安全需求。

完整的数据支撑

超级播放器支持全链路视频播放质量监控，包含播放性能、用户行为、文件特征等多维度数据指标，助力业务高效运营。

极致的播放体验

依靠腾讯云海量加速节点，提供完备的视频加速能力，毫秒级的延迟让用户无延迟体验极速视频播放，为用户业务保驾护航。

多样化播放能力

提供首屏秒开、边播边缓存、倍速播放、视频打点、媒体弹幕和外挂字幕等多种功能，用户可以根据业务需求选择功能，助力业务生态建设。

常见场景

短视频播放

超级播放器结合云点播平台提供的内容审核、媒资管理、无缝切换、首屏秒开互动浮窗等功能，常用于用户打造短视频相关场景，同时云点播提供 短视频 UGSV Demo 供用户参考。

长视频播放

超级播放器结合云点播的自适应码流、无缝清晰度切换、缩略图、截图、倍速播放等功能，长视频用户打造视频剧集和门户类场景，同时云点播提供 视频播放方案 供用户参考。

视频版权保护

超级播放器支持云点播的视频安全能力，支持私有协议加密、离线下载、跑马灯和防盗链等功能，助力用户保障自己视频安全，同时云点播提供 视频安全方案教程 供用户参考。

直播录制

超级播放器支持直播录制文件播放，同时支持直播时移回看、伪直播观看，在音视频直播点播场景下帮助用户打造一体化观看体验。

SDK下载

您可以体验播放器 Demo，更多信息，请参见 Demo 体验。
您可以下载对应的 SDK 进行集成操作，更多信息，请参见 SDK 下载。
您可以通过 功能列表，查询超级播放器是否满足自己的能力需求。

接入指引

为了帮助您快速接入超级播放器，我们为您提供了超级播放器 接入指引，以示例的方式为您讲解接入步骤。

如遇到播放问题，请参见 常见问题文档。

基本概念

最近更新时间：2022-05-18 14:34:21

本文对腾讯云视立方播放器 Player（以下简称播放器 Player）的点播播放场景中涉及的基本概念进行说明，帮助您快速的理解和使用播放器 Player 下的视频点播能力。

基本概念

播放器 Player 在云点播平台支持超级播放器和超级播放器 Adapter 两种方式接入点播服务。

超级播放器

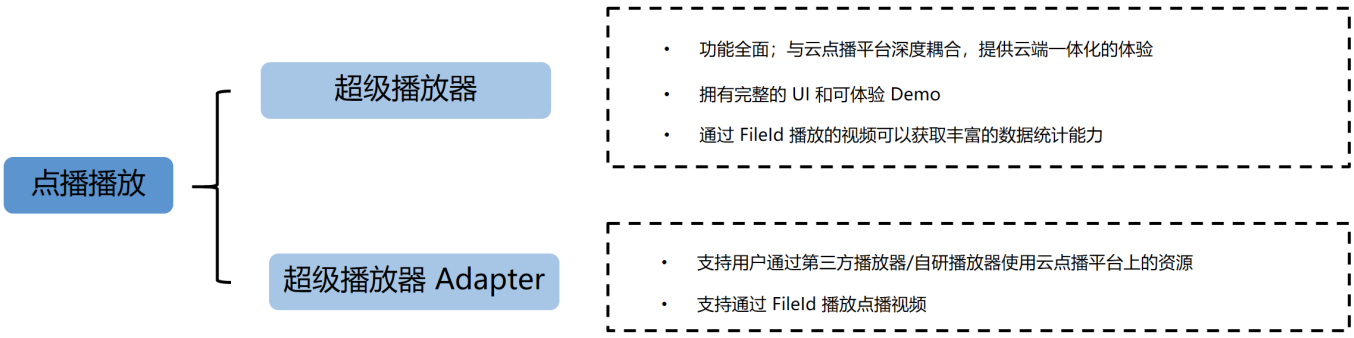
超级播放器是一款独立完整的视频播放器，具备视频加密、缩略图预览、清晰度切换等全面的视频播放功能；该播放器拥有完整 UI 和体验 Demo，同时深度融合腾讯云点播业务，可通过使用点播文件标识 FileID 播放云点播资源，此外，超级播放器还提供云点播全链路视频播放质量数据服务。如果您想要快速接入超级播放器，请参见 超级播放器教程。

超级播放器 Adapter

超级播放器 Adapter 是一款用于连接第三方播放器与腾讯云点播资源的播放器插件，具备视频播放、视频加密等能力。将超级播放器 Adapter 集成在用户第三方播放器中，即可通过点播文件标识 FileID 播放云点播资源。如果您想要快速接入超级播放器 Adapter，请参见各端集成文档。

如何选择播放器

为降低用户的接入难度、匹配用户自身业务场景，云点播建议用户在接入播放器服务时时，选择最适合自己的播放器类型接入：



- **超级播放器**：适用于尚未集成播放器，但需要快速搭建点播视频播放能力的用户。播放器 Player 功能全面，接入便捷，云点播为用户提供了详细的超级播放器接入教程，详情请参见 超级播放器教程。
- **超级播放器 Adapter**：适用于需要使用自研/第三方播放器播放云点播资源的用户。云点播提供超级播放器 Adapter 帮助客户顺利接入和使用云点播平台资源。

各类型播放器支持的平台端类型如下：

播放器类别	超级播放器	超级播放器 Adapter

播放器类别	超级播放器	超级播放器 Adapter
Web 端	✓	✓
iOS 端	✓	✓
Android 端	✓	✓
Flutter 端	✓	–
UI	✓	–
Demo	✓	–

SDK 下载
SDK 下载

最近更新时间：2022-04-02 09:05:45

腾讯云视立方播放器 Player（以下简称播放器 Player）支持 Web、iOS、Android 和 Flutter 四大终端，云点播为客户提供了完整的 Demo 和播放器 Player 下载。

SDK 下载 & Demo

终端类别	播放器类型	SDK & Demo 下载地址	Demo 展示	使用文档
Web 端	超级播放器	SDK	Demo	Web – 超级播放器
	超级播放器 Adapter	SDK	–	Web – 超级播放器 Adapter
iOS 端	超级播放器	SDK + Demo		iOS – 超级播放器
	超级播放器 Adapter	SDK	–	iOS – 超级播放器 Adapter
Android 端	超级播放器	SDK + Demo		Android – 超级播放器
	超级播放器 Adapter	SDK	–	Android– 超级播放器 Adapter
Flutter 端	超级播放器	SDK + Demo	Demo	Flutter – 超级播放器

功能说明

最近更新时间：2022-06-08 10:12:55

超级播放器

超级播放器在各端上都提供丰富的能力接入，各平台支持的能力列表展示如下：

功能点	功能说明	iOS & Android	Web	Flutter
多种格式	支持 RTMP、FLV、HLS、MP4、WebRTC 等丰富的音视频格式	✓	✓	✓
URL 播放	支持网络视频的 URL 方式播放	✓	✓	✓
DASH 协议	支持标准协议的 DASH 格式视频	×	✓	×
FileID 播放	支持使用云点播的 FileID 方式播放	✓	✓	✓
首屏秒开预加载	支持视频内容预加载，视频首屏可达到秒开	✓	✓	×
快速 seek	支持精准快速的 seek 到指定位置播放	✓	✓	✓
H.265 硬解	支持对 H.265 硬解码播放	✓	✓*	✓
软硬解自动切换	当终端不支持硬件解码时自动切换到软解	✓	–	✓
自适应码流	播放 HLS 自适应码流时，支持手动指定或根据网络带宽自动选择清晰度流进行播放	✓	✓	×
清晰度切换	支持用户流畅无卡顿的切换多路清晰度流	✓	✓	✓
清晰度命名	支持为不同清晰度流进行自定义命名	✓	✓	×
播放控制	支持开始、结束、暂停、自动播放、循环播放、断点续播、重播等播放控制功能	✓	✓	✓
倍速播放	支持0.5倍 – 2倍的视频变速播放，可保证音频变速不变调	✓	✓	✓
自定义启播时间	支持自定义视频开启播放的时间	✓	✓	✓
试看功能	支持播放开启试看功能的视频	✓	✓	✓
进度条操作	拖拽进度条切换进度	✓	✓	✓
进度条标记及缩略图预览	支持在进度条上添加标记信息，并支持缩略图（雪碧图）预览	✓	✓	✓

播放器尺寸	支持自定义设置播放器尺寸	✓	✓	✓
屏幕填充适应	支持为视频画面选择不同填充模式，适应屏幕大小	✓	✓	✓
小窗播放	支持切换到小窗播放	✓	✓	✓
视频镜像	支持水平、垂直等方向的镜像	✓	✓	✓
视频旋转	支持对视频画面按角度旋转，同时支持根据视频文件内部 rotate 参数自动旋转视频	✓	–	×
亮度调节	支持播放视频时调节系统亮度	✓	–	✓
音量调节	支持播放视频时调节系统音量和静音操作	✓	✓	✓
双声道音频	支持播放双声道音频	✓	✓	✓
纯音频播放	支持 MP3 等文件纯音频播放	✓	✓	✓
锁定屏幕	支持锁屏功能，包含锁定旋转和隐藏界面元素	✓	–	✓
弹幕	支持在视频上方展示弹幕	✓	✓	✓
图片贴片	支持暂停时，增加图片贴片用于广告展示	✓	✓	✓
视频截图	支持截取播放画面的任意一帧	✓	–	✓
字幕导入	支持导入自定义字幕文件	×	✓	×
设置封面	支持设置播放视频的封面	✓	✓	✓
多实例	支持在一个界面添加多个播放器同时播放	✓	✓	✓
边下边播	支持视频播放的同时缓存下载后面的内容	✓	✓	✓
Referer 防盗链	支持通过播放请求中携带的 Referer 字段识别请求的来源，以黑名单或白名单方式对来源请求进行控制	✓	✓	✓
Key 防盗链	支持在播放链接中加入控制参数，控制链接的有效时间、试看时长、允许播放的 IP 数等	✓	✓	✓
HLS 加密	支持基于 HLS 提供的 AES encryption 方案，使用密钥对视频数据加密	✓	✓	✓
私有协议加密	支持在云端通过私有协议对视频进行加密，且仅能通过播放器 SDK 对加密后的视频进行解密播放	✓	✓	×
离线下载	支持离线下载加密视频后，仅可通过播放器 SDK 对视频进行解密播放	✓	–	×
播放回调	支持对播放状态回调、首帧回调、播放完成或失败回调	✓	✓	✓
支持 HTTPS	支持播放 HTTPS 的视频资源	✓	✓	✓

自定义 HTTP 头部	请求视频资源时，自定义 HTTP Headers 内容	✓	–	×

说明：

- 表中“–”表示该端无需具备相应功能或不存在相关概念。
- Web 播放器支持H.265硬解，但在实际播放场景中，最终效果将依赖浏览器对H.265硬解的支持情况，仅当浏览器也同样支持时，播放时才可支持H.265硬解。

超级播放器 Adapter

超级播放器 Adapter 在各端上都提供丰富的能力接入，各平台支持的能力列表展示如下：

功能点	功能说明	iOS/Android	Web
QUIC 协议	支持 QUIC 协议	×	✓
FileID播放	支持使用云点播的 FileId 方式播放	✓	✓
清晰度切换	支持用户流畅无卡顿的切换多路清晰度流	✓	✓
清晰度命名	支持为不同清晰度流进行自定义命名	✓	✓
进度条标记	支持在进度条上添加标记信息	✓	✓
缩略图预览	支持在进度条上展示缩略图（雪碧图）做预览	✓	✓
设置封面	支持设置播放视频的封面	✓	✓
HLS 加密	支持基于 HLS 提供的 AES encryption 方案，使用密钥对视频数据加密	✓	✓
私有协议加密	支持在云端通过私有协议对视频进行加密，且仅能通过播放器 SDK 对加密后的视频进行解密播放	✓	✓

接口说明

更多操作，请参见各端接口说明文档：

终端	Web端	iOS 端	Android 端

终端	Web端	iOS 端	Android 端
超级播放器	Web 超级播放器 - 接口说明	iOS 超级播放器 - 接口说明	Android 超级播放器 - 接口说明
超级播放器 Adapter	Web 超级播放器 Adapter - 接口说明	iOS 超级播放器 Adapter- 接口说明	Android 超级播放器 - 接口说明

超级播放器教程
阶段1：用超级播放器播放视频

最近更新时间：2022-03-08 14:17:25

学习目标

通过本阶段的教程后，您将掌握上传一个视频到云点播，并通过视频处理后，在超级播放器中播放的技能。

前置条件

在开始本教程之前，请您确保已满足以下前置条件。

开通云点播

您需要开通云点播，步骤如下：

1. 注册 腾讯云账号，并完成 实名认证。
2. 购买云点播服务，具体请参见 计费概述。
3. 选择 **云产品>视频服务>云点播**，进入云点播控制台。

至此，您已经完成了云点播的开通步骤。

保持防盗链未开启

您需要确认自己的默认分发域名，没有开启防盗链：

1. 登录云点播控制台，选择 **分发播放设置** > **域名管理** ，单击“默认分发域名”操作栏下的 **设置**，进入设置页面。

2. 查看“Referer 防盗链”和“Key 防盗链”的启用状态，确认均为“未启用”状态。

步骤1：上传及处理视频

本步骤，我们将指导您如何上传视频，并对视频转自适应码流、截取封面和雪碧图。

1. 登录云点播控制台，选择 **媒资管理**> **视频管理** ，单击 **上传视频**。

2. 在上传界面，选择 **本地上传**，并单击 **选择视频** 上传本地视频，其他设置如下：
   - **视频处理** 选择 **上传后自动进行视频处理**。
   - **处理类型** 选择 **任务流**。
   - **任务流模板** 选择 LongVideoPreset。

> 说明：
> LongVideoPreset 是预置任务流，使用10模板转自适应码流，10模板截图做封面，10模板截雪碧图。

3. 单击 **开始上传**，进入“正在上传”页，等待上传完成。
4. 选择 **媒资管理>视频管理**，找到新上传的视频（FileId 为528xxx3757278095），其中 ID 即为上传视频的 FileId。

> **说明：**
> 等待“视频状态”由 **处理中** 变为 **正常**，表示视频已处理完毕。

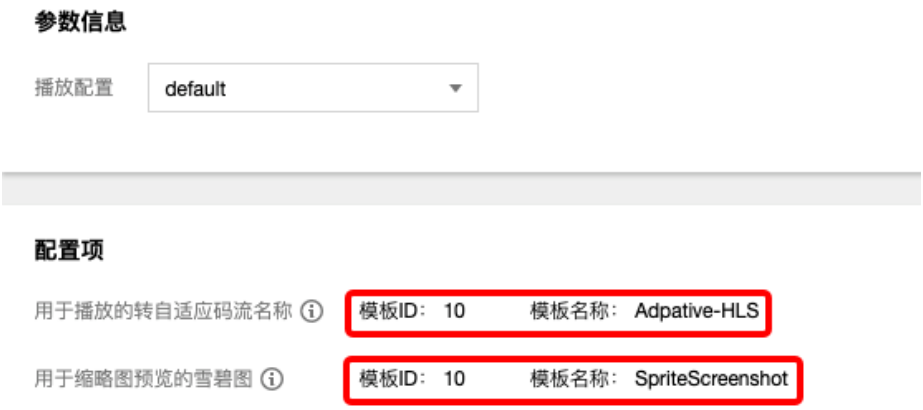
5. 单击新上传视频“操作”栏下的 **管理**，进入管理页面：
   - 选择“基本信息”页签，可以看到生成的封面，以及自适应码流输出（模板 ID 为10）。
   - 选择“截图信息”页签，可以看到生成的雪碧图（模板 ID 为10）。

步骤2：预览播放体验

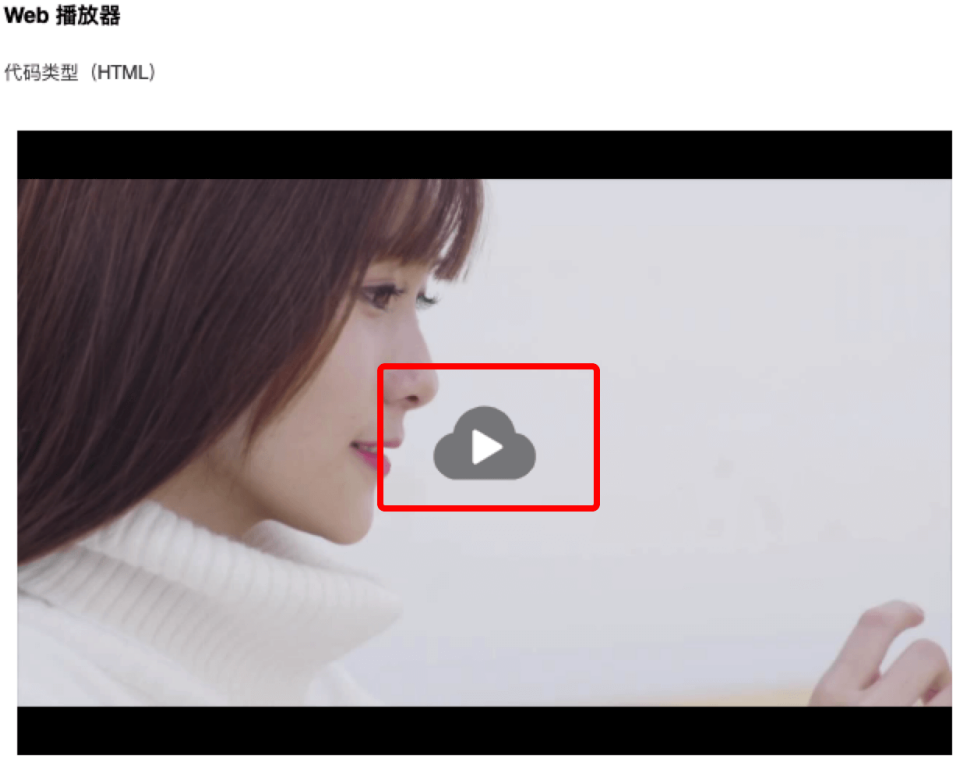
前面的步骤中，您已经上传视频，并对视频进行了处理。现在，将使用三端的超级播放器，快速体验播放效果。

1. 选择 **媒资管理>视频管理** ，找到**步骤1**上传和处理过的视频，单击“操作”栏下的 **管理**，选择 **超级播放器预览**。
2. **超级播放器配置** 选择 default。

> 说明：
> default 是预置超级播放器配置，用于播放10模板转自适应码流输出，10模板截雪碧图输出。



3. 在 **Web 播放器** 中，单击播放器中间的按钮，即可在 Web 端播放体验。



4. 在 **移动端播放器** 中，点击 **扫码下载**，安装“腾讯云工具包”。

5. 手机打开腾讯云工具包，选择 **播放器 > 超级播放器**，然后点击右上角扫码，即可在移动端播放体验。

步骤3：使用 Demo 体验

您可以分别使用 Web、Android 和 iOS 三端的超级播放器 Demo 进行验证，具体请参考 Demo 的源码。

说明：

在控制台的 **媒资管理** > **视频管理** > **超级播放器预览** 中，可以获取预览视频对应的 Web 播放器源码，供您直接参考使用。

总结

学习本教程后，您已经初步了解了，如何上传一个视频到云点播，并通过视频处理后，在超级播放器中播放。

如果您希望：

- 开启 Key 防盗链后播放视频，请参考 阶段2：开启防盗链后的视频播放。
- 自定义视频播放时的内容和样式，请参考 阶段3：自定义播放内容与样式。
- 对视频进行加密，并播放加密后的视频，请参考 阶段4：播放加密视频。

阶段2：开启防盗链后的视频播放

最近更新时间：2022-03-08 14:18:23

学习目标

云点播支持设置防盗链，实现有效时间、播放人数、播放时长等控制。学习本阶段教程后，您将掌握防盗链开启后，使用超级播放器的视频播放方式。

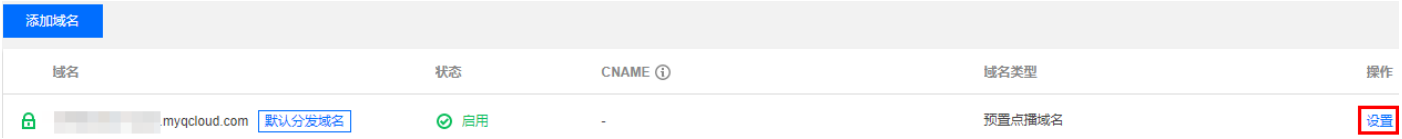
阅读之前，请先确保已经学习超级播放器指引的 阶段1：用超级播放器播放视频 篇部分，本教程使用了 阶段1 篇开通的账号以及上传的视频。

步骤1：开启防盗链

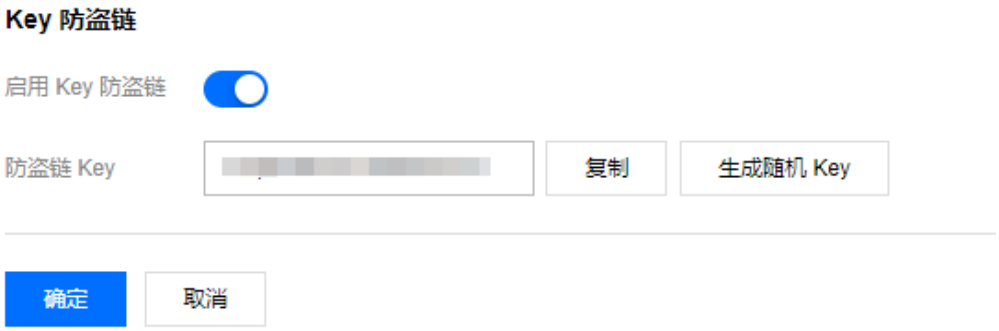
以您账号下的默认分发域名开启 Key 防盗链为例：

> ⚠ 注意：
> 请避免直接对生产环境的现网域名开启防盗链，否则可能造成现网的视频无法播放。

1. 登录云点播控制台，选择【分发播放设置】>【域名管理】，单击“默认分发域名”的【设置】，进入设置页面。



2. 单击“Key 防盗链”右侧的【编辑】，打开【启用 Key 防盗链】，并单击【生成随机 Key】生成一个随机的 Key（2WExxx48eW），将生成好的 Key 复制下来，然后单击【确定】保存生效。



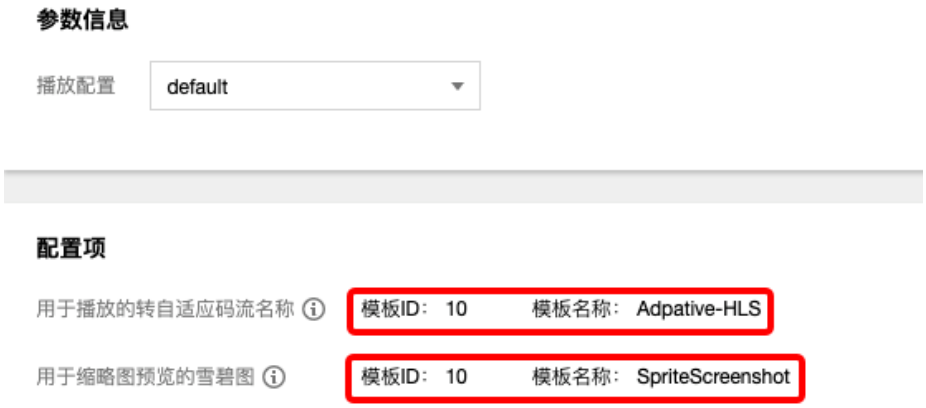
步骤2：预览播放体验

前面的步骤，您已经对默认分发域开启了防盗链。现在，将使用三端的超级播放器，快速体验播放效果。

1. 选择【媒资管理】>【视频管理】，找到**步骤1**上传和处理过的视频，单击“操作”栏下的【管理】，选择【超级播放器预览】。
2. 【超级播放器配置】选择 default。

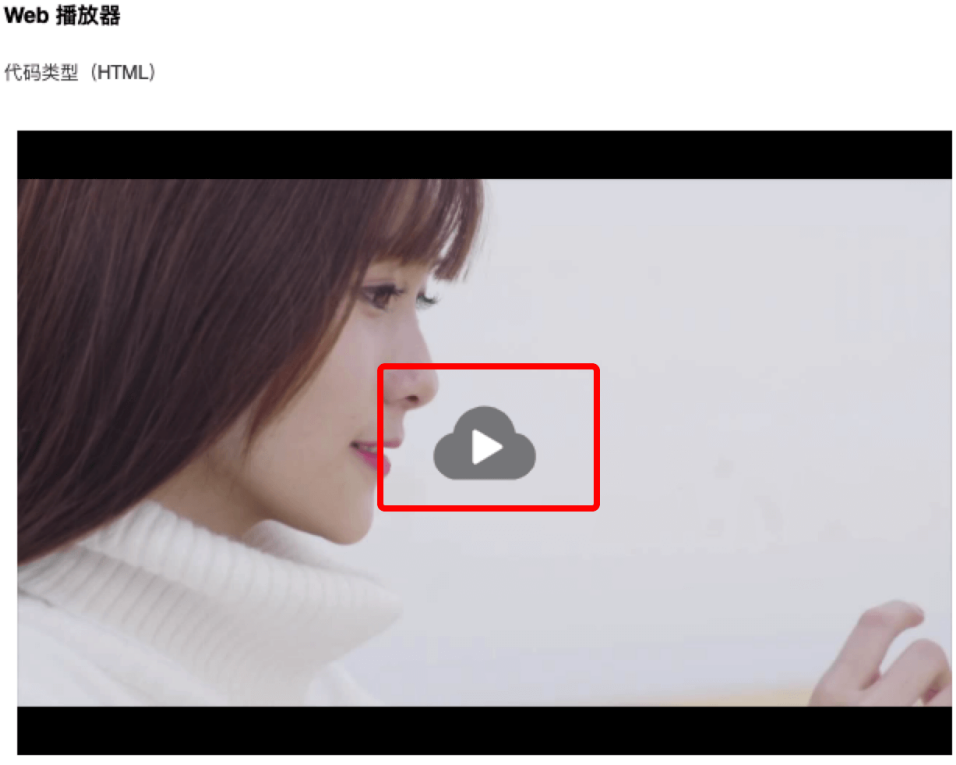
说明：

default 是预置超级播放器配置，用于播放10模板转自适应码流输出，10模板截雪碧图输出。



3. 因为默认分发域名开启了防盗链，【播放控制】选项卡支持预览时选定防盗链的过期时间、试看时长等。此处可维持默认参数（播放防盗链过期时间默认1天，试看时长和最多可播放 IP 个数不填写）。

4. 在【Web 播放器】中，单击播放器中间的按钮，即可在 Web 端播放体验。



5. 在【移动端播放器】中，点击【扫码下载】，安装“腾讯云工具包”。

6. 手机打开腾讯云工具包，选择【播放器】>【超级播放器】，然后点击右上角扫码，即可在移动端播放体验。

步骤3：使用 Demo 验证

开启防盗链后，超级播放器必须使用有效期内的签名，才能播放视频。您可以使用签名工具快速生成签名。

1. 打开 超级播放器 – 签名生成工具 页面，并填写参数：
   - 【用户 appId】：填写视频所属的 appId：1400xxx357（如果使用的是子应用，填写子应用的 appId）。
   - 【视频 fileId】：填写视频的 fileId：528xxx3757278095。
   - 【当前 Unix 时间戳】：工具自动生成出了当前的 Unix 时间（1591516390），无需填写。
   - 【签名过期 Unix 时间戳】：签名本身的过期时间，可以不填写，默认为1天后过期。
   - 【链接过期时间】：Key 防盗链过期时间，可以填6小时后的十六进制 Unix 时间：5edcf146。
   - 【防盗链 Key】：填写上一步获取到的防盗链 Key：2WExxx48eW。
2. 单击【生成签名】，生成出来的签名显示在“生成签名结果”文本框中。

获取超级播放器签名后，您可以分别使用 Web、Android 和 iOS 三端的超级播放器 Demo 进行验证，具体请参考 Demo 的源码。

> 说明：
> 在控制台的【媒资管理】>【视频管理】>【超级播放器预览】中，可以获取预览视频对应的 Web 播放器源码，供您直接参考使用。

复制代码

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="viewport" content="width=device-width, initial-scale=1, maxim
        <title>腾讯云视频点播示例</title>
        <!-- 引入播放器 css 文件 -->
        <link href="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.css" re
        <!-- 如需在IE8、9浏览器中初始化播放器，浏览器需支持Flash并在页面中引入 -->
        <!--[if lt IE 9]>
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/ie8/videojs-ie8
        <![endif]-->
        <!-- 如果需要在 Chrome Firefox 等现代浏览器中通过H5播放hls，需要引入 hls.js -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/libs/hls.min.0.
        <!-- 引入播放器 js 文件 -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.v4.min
        <!-- 示例 CSS 样式可自行删除 -->
    </head>
    <body>
        <!-- 设置播放器容器 -->
        <video id="player-container-id" preload="auto" width="600" height="400" p
        <!--
        注意事项:
        * 播放器容器必须为 video 标签
        * player-container-id 为播放器容器的ID，可自行设置
        * 播放器区域的尺寸请按需设置，建议通过 css 进行设置，通过css可实现容器自适应等效果
        * playsinline webkit-playsinline x5-playsinline 这几个属性是为了在标准移动端浏
        -->
        <script>
            var player = TCPlayer("player-container-id", { /**player-container-id
                fileID: "[redacted]", /**请传入需要播放的视频fileID 必须 */
                appID: "[redacted]", /**请传入点播账号的appID 必须 */
                psign: ""
                /**其他参数请在开发文档中查看 */
                });
        </script>
    </body>
</html>
```

总结

学习本教程后，您已经掌握防盗链开启后，如何使用超级播放器播放视频。

如果您希望：

- 自定义视频播放时的内容和样式，请参考 阶段3：自定义播放内容与样式。
- 对视频进行加密，并播放加密后的视频，请参考 阶段4：播放加密视频。

阶段3：自定义播放内容与样式

最近更新时间：2022-03-08 14:18:30

学习目标

学习本阶段教程，您将了解如何定制超级播放器播放的视频内容与样式，包括：

- 播放子流规格中最小分辨率为480p，最大分辨率1080p。
- 使用视频中间部分的截图作为视频封面。
- 进度条上的缩略图预览，调整为20%的间隔。

阅读之前，请先确保已经学习超级播放器指引的 阶段1：用超级播放器播放视频 和 阶段2：开启防盗链后的视频播放 篇部分，本教程使用了 阶段1 篇开通的账号以及上传的视频，并需要按照 阶段2 开启防盗链。

步骤1：创建自适应码流模板

1. 登录云点播控制台，选择【视频处理设置】>【模板设置】，单击“转自适应码流模板”页签下的【创建转自适应码流模板】。

2. 进入“模板设置”页面后，单击【添加子流】，新建子流1、子流2和子流3，填写参数如下：
   - **基本信息模块：**
     - 【模板名称】：填写 MyTestTemplate。
     - 【加密类型】：选择【不加密】。
     - 【是否允许低分辨率转高分辨率】：不开启。
   - **子流信息模块：**

子流编号	视频码率	分辨率	帧率	音频码率	声道
子流1	512kbps	视频长边0px，视频短边480px	24fps	48kbps	双声道
子流2	512kbps	视频长边0px，视频短边720px	24fps	48kbps	双声道
子流3	1024kbps	视频长边0px，视频短边1080px	24fps	48kbps	双声道

3. 单击【创建】，则生成了一个包含3个子流的自适应码流模板，模板 ID 为125866。

模板名称 / ID	打包类型	子流数	禁止低分辨率转高分辨率
Presetting HLS 10	HLS	6条子流	禁止
Encrypt-SimpleAES HLS 12	HLS	6条子流	禁止
MyTestTemplate 125866	HLS	3条子流	禁止

步骤2：创建雪碧图模板

1. 登录云点播控制台，选择【视频处理设置】>【模板设置】，单击“截图模板”页签下的【创建截图模板】。
2. 进入“模板设置”页面后，填写模板参数：
   - 【模板名称】：填写 MyTestTemplate。
   - 【模板类型】：选择【雪碧图截图】。
   - 【图片尺寸】：726px × 240px。

- 【采样间隔】：20%。
- 【小图行数】：10。
- 【小图列数】：10。

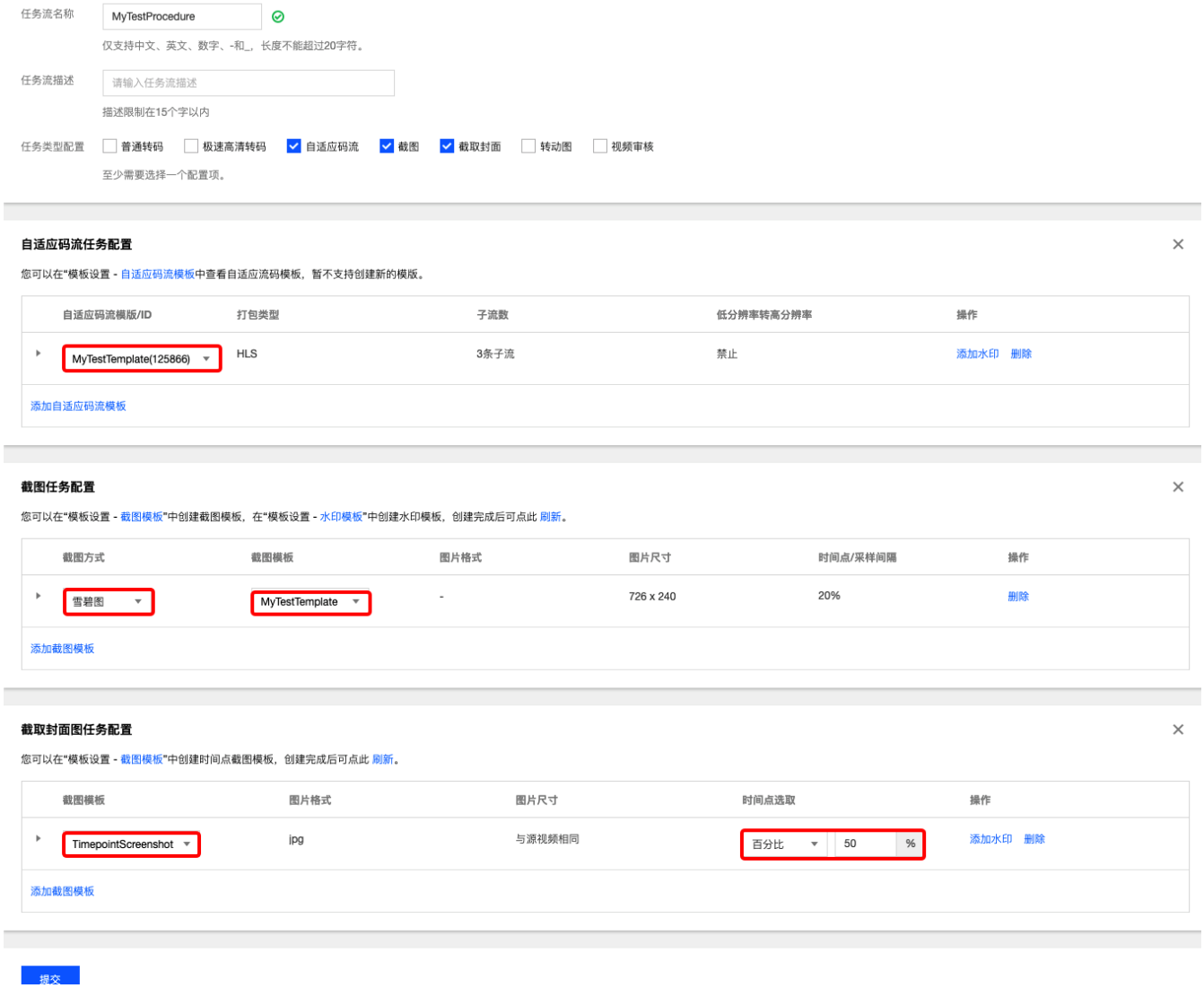
3. 单击【创建】，则生成了一个模板 ID 为41377的雪碧图模板。

步骤3：创建任务流并发起处理

创建新的转自适应码流模板（ID 为125866）和雪碧图模板（ID 为41377）后，还需要创建一个新的任务流。

1. 登录云点播控制台，选择【视频处理设置】>【任务流设置】，单击【创建任务流】：
   - 【任务流名称】：填写 MyTestProcedure。
   - 【任务类型配置】：勾选【自适应码流】、【截图】和【截取封面】：
     - 在【自适应码流任务配置】选项卡，单击【添加自适应码流模板】，在“自适应码流模版/ID”栏选择**步骤1**创建的自定义转自适应码流模板 MyTestTemplate(126866)。

- 在【截图任务配置】选项卡，单击【添加截图模板】，“截图方式”栏选择【雪碧图】，“截图模板”栏选择步骤2创建的自定义雪碧图模板 MyTestTemplate(41377)。
- 在【截取封面图任务配置】选项卡，单击【添加截图模板】，“截图模板”栏选择【TimepointScreenshot】，“时间点选取”栏选择【百分比】，填写50%。



2. 单击【提交】，生成了一个名为 MyTestProcedure 的任务流。

3. 在控制台选择【媒资管理】>【视频管理】，勾选要处理的视频（FileId 为528xxx3757278095），单击【视频处理】。
4. 在视频处理弹框：
   - 【处理类型】选择【任务流】。

- 【任务流模板】选择【MyTestProcedure】。

5. 单击【确定】，等待**视频状态**从“处理中”变为“正常”，表示视频已处理完毕。

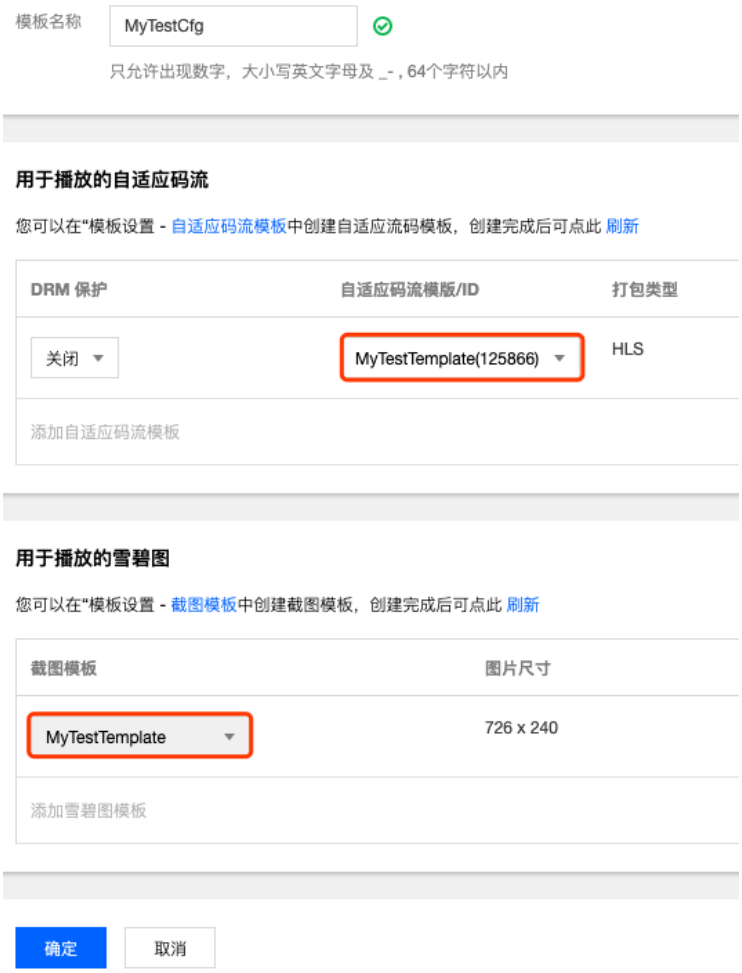
6. 单击视频“操作”栏中的【管理】，进入管理页面：
   - 在【基本信息】栏，可以看到生成的封面，以及自适应码流输出（模板 ID 为125866）。
   - 在【截图信息】栏，可以看到生成的雪碧图（模板 ID 为41377）。

步骤4：创建超级播放器配置

为了播放自定义的自适应码流和雪碧图，您需要使用自定义播放器配置。

1. 登录云点播控制台，选择【分发播放配置】>【超级播放器配置】，单击【新建】。
2. 在超级播放器配置页面中，分别单击【添加自适应码流模板】和【添加雪碧图模板】，然后填写以下参数：
   - 【模板名称】：填写 MyTestCfg。
   - 【用于播放的自适应码流】选项卡的“自适应码流模板/ID”栏选择：MyTestTemplate(125866)。

- 【用于播放的雪碧图】选项卡的“截图模板”栏选择：MyTestTemplate(41377)。



3. 单击【确定】，则生成新的超级播放器配置 MyTestCfg。

步骤5：预览播放体验

经过之前的步骤，您已经对视频进行了处理。现在将使用三端的超级播放器，快速体验播放效果。

1. 选择【视频管理】的“已上传”页签，找到之前步骤上传和处理过的视频，单击“操作”栏中的【管理】，选择“超级播放器预览”页签。
2. 【播放配置】选择 MyTestCfg。

3. 因为默认分发域名开启了防盗链，【播放控制】选项卡支持预览时选定防盗链的过期时间、试看时长等。此处可维持默认参数（播放防盗链过期时间默认1天，试看时长和最多可播放 IP 个数不填写）。

4. 在【Web 播放器】中，单击播放器中间的按钮，即可在 Web 端播放体验。

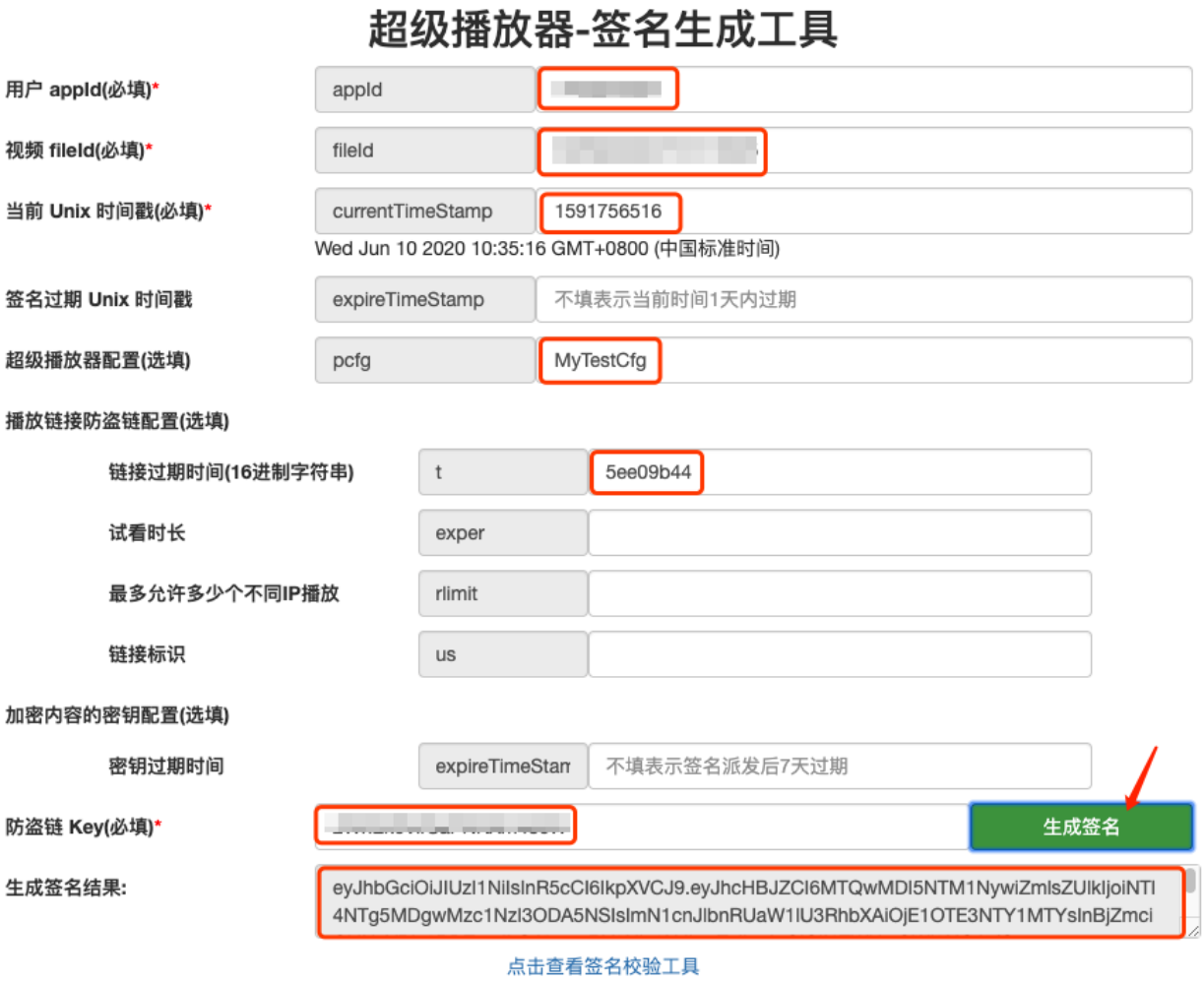
5. 在【移动端播放器】中，点击【扫码下载】，安装“腾讯云工具包”。

6. 手机打开腾讯云工具包，选择【播放器】>【超级播放器】，然后点击右上角扫码，即可在移动端播放体验。

步骤6：使用 Demo 验证

开启防盗链后，超级播放器必须使用有效期内的签名，才能播放视频。下面将介绍如何使用签名工具快速生成签名。

1. 打开 超级播放器 – 签名生成工具 页面，并填写参数：
   - 【用户 appId】：填写视频所属的 appId：1400xxx357（如果使用的是子应用，填写子应用的 appId）。
   - 【视频 fileId】：填写视频的 FileId：528xxx3757278095。
   - 【当前 Unix 时间戳】：工具自动生成出了当前的 Unix 时间（1591756516），无需填写。
   - 【超级播放器配置】：填写自定义的超级播放器配置名 MyTestCfg。
   - 【签名过期 Unix 时间戳】：签名本身的过期时间，可以不填写，默认为1天后过期。
   - 【链接过期时间】：Key 防盗链过期时间，可以填6小时后的十六进制 Unix 时间：5ee09b44。
   - 【防盗链 Key】：填写之前获取到的防盗链 Key：2WExxx48eW。
2. 单击【生成签名】，生成出来的签名显示在“生成签名结果”文本框中。



获取超级播放器签名后，您可以分别使用 Web、Android 和 iOS 三端的超级播放器 Demo 进行验证，具体请参考 Demo 的源码。

> 说明：
> 在控制台的【媒资管理】>【视频管理】>【超级播放器预览】中，可以获取预览视频对应的 Web 播放器源码，供您直接参考使用。

复制代码

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="viewport" content="width=device-width, initial-scale=1, maxim
        <title>腾讯云视频点播示例</title>
        <!-- 引入播放器 css 文件 -->
        <link href="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.css" re
        <!-- 如需在IE8、9浏览器中初始化播放器，浏览器需支持Flash并在页面中引入 -->
        <!--[if lt IE 9]>
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/ie8/videojs-ie8
        <![endif]-->
        <!-- 如果需要在 Chrome Firefox 等现代浏览器中通过H5播放hls，需要引入 hls.js -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/libs/hls.min.0.
        <!-- 引入播放器 js 文件 -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.v4.min
        <!-- 示例 CSS 样式可自行删除 -->
    </head>
    <body>
        <!-- 设置播放器容器 -->
        <video id="player-container-id" preload="auto" width="600" height="400" p
        <!--
        注意事项:
        * 播放器容器必须为 video 标签
        * player-container-id 为播放器容器的ID，可自行设置
        * 播放器区域的尺寸请按需设置，建议通过 css 进行设置，通过css可实现容器自适应等效果
        * playsinline webkit-playsinline x5-playsinline 这几个属性是为了在标准移动端浏
        -->
        <script>
            var player = TCPlayer("player-container-id", { /**player-container-id
                fileID: "[redacted]", /**请传入需要播放的视频fileID 必须 */
                appID: "[redacted]", /**请传入点播账号的appID 必须 */
                psign: ""
                /**其他参数请在开发文档中查看 */
                });
        </script>
    </body>
</html>
```

总结

学习本教程后，您已经掌握如何定制超级播放器播放的视频内容与样式。

如果您希望对视频进行加密，并播放加密后的视频，请参考 阶段4：播放加密视频。

阶段4：播放加密视频

最近更新时间：2022-03-08 14:18:37

学习目标

学习本阶段教程，您将了解并掌握如何对视频加密，并使用超级播放器播放加密后的视频。

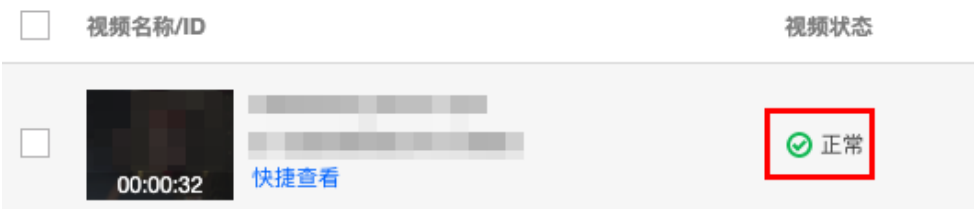
阅读之前，请先确保已经学习超级播放器指引的 阶段1：用超级播放器播放视频 和 阶段2：开启防盗链后的视频播放 篇部分，本教程使用了 阶段1 篇开通的账号以及上传的视频，并需要按照 阶段2 开启防盗链。

步骤1：视频加密

1. 登录云点播控制台，选择 **媒资管理>视频管理**，勾选要处理的视频（FileId 为528xxx3757278095），单击 **视频处理**。
2. 在视频处理界面：
   - **处理类型** 选择 **任务流**。
   - **任务流模板** 选择 **SimpleAesEncryptPreset**。

> 说明：
>
> - SimpleAesEncryptPreset 是预置任务流：使用12模板转自适应码流，10模板截图做封面，10模板截雪碧图。
> - 12模板自适应码流是转出加密的多码率输出。

3. 单击 **确定**，等待“视频状态”栏从“处理中”变为“正常”，表示视频已处理完毕：



4. 单击视频“操作”栏下的 **管理**，进入管理页面：

- 选择“基本信息”页签，可以看到生成的封面，以及加密的自适应码流输出（模板 ID 为12）。
- 选择“截图信息”页签，可以看到生成的雪碧图（模板 ID 为10）。

步骤2：预览播放体验

前面的步骤中，您已经上传视频，并对视频进行了处理。现在，将使用三端的超级播放器，快速体验播放效果。

1. 选择 **媒资管理>视频管理** ，找到**步骤1**上传和处理过的视频，单击“操作”栏下的 **管理**，选择 **超级播放器预览**。
2. **超级播放器配置** 选择 basicDrmPreset。

> 说明：
> basicDrmPreset 是预置超级播放器配置，用于播放12模板转自适应码流输出，10模板截雪碧图输出。

3. 因为默认分发域名开启了防盗链， **播放控制** 选项卡支持预览时选定防盗链的过期时间、试看时长等。此处可维持默认参数（播放防盗链过期时间默认1天，试看时长和最多可播放 IP 个数不填写）。

4. 在 **Web 播放器** 中，单击播放器中间的按钮，即可在 Web 端播放体验。

Web 播放器

代码类型（HTML）

5. 在 **移动端播放器** 中，单击 **扫码下载**，安装“腾讯云工具包”。

终端播放器

1. 下载视频云工具包 App。

扫码下载

2. 使用视频云工具包 App扫描下方二维码可在终端上预览视频。

6. 手机打开腾讯云工具包，选择 **播放器>超级播放器**，然后点击右上角扫码，即可在移动端播放体验。

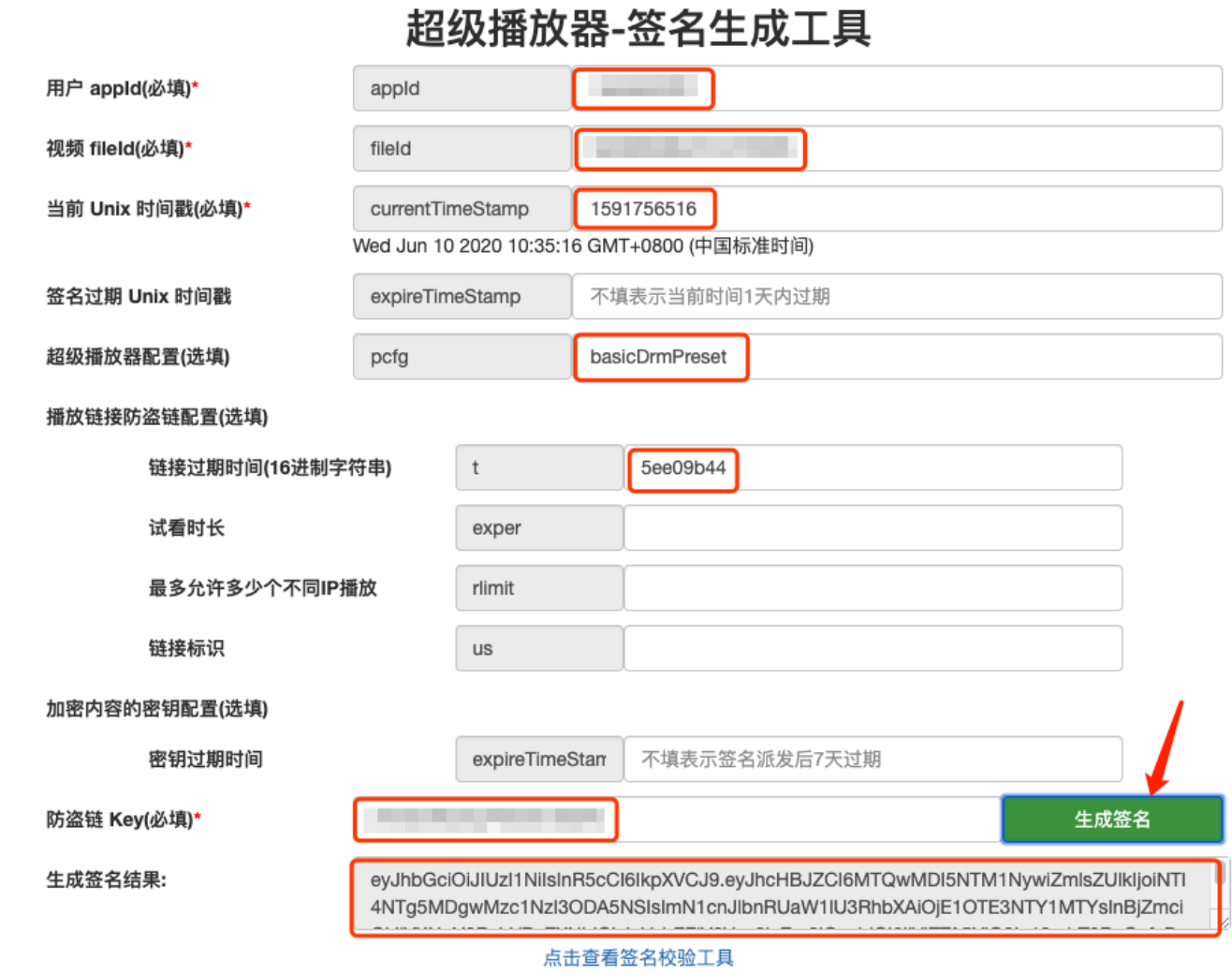
步骤3：使用 Demo 验证

超级播放器必须使用有效期内的签名，才能播放加密视频。下面将介绍如何使用签名工具快速生成签名。

1. 打开 超级播放器 - 签名生成工具 页面，并填写参数：
   - **用户 appId**：填写视频所属的 appId：1400xxx357（如果使用的是子应用，填写子应用的 appId）。
   - **视频 fileId**：填写视频的 FileId：528xxx3757278095。
   - **当前 Unix 时间戳**：工具自动生成出了当前的 Unix 时间（1591756516），无需填写。
   - **超级播放器配置**：填写预置超级播放器配置名 basicDrmPreset。
   - **签名过期 Unix 时间戳**：签名本身的过期时间，可以不填写，默认为1天后过期。
   - **链接过期时间**：Key 防盗链过期时间，可以填6小时后的十六进制 Unix 时间：5ee09b44。
   - **防盗链 Key**：填写之前获取到的防盗链 Key：2WExxx48eW。

> ⚠ 注意：
> basicDrmPreset 是预置超级播放器配置，用于播放12模板转自适应码流输出，10模板截雪碧图输出。

2. 单击 **生成签名**，生成出来的签名显示在 **生成签名结果** 文本框中。



获取超级播放器签名后，您可以分别使用 Web、Android 和 iOS 三端的超级播放器 Demo 进行验证，具体请参考 Demo 的源码。

> 说明：
> 在控制台的 **媒资管理>视频管理 > 超级播放器预览** 中，可以获取预览视频对应的 Web 播放器源码，供您直接参考使用。

复制代码

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="viewport" content="width=device-width, initial-scale=1, maxim
        <title>腾讯云视频点播示例</title>
        <!-- 引入播放器 css 文件 -->
        <link href="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.css" re
        <!-- 如需在IE8、9浏览器中初始化播放器，浏览器需支持Flash并在页面中引入 -->
        <!--[if lt IE 9]>
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/ie8/videojs-ie8
        <![endif]-->
        <!-- 如果需要在 Chrome Firefox 等现代浏览器中通过H5播放hls，需要引入 hls.js -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/libs/hls.min.0.
        <!-- 引入播放器 js 文件 -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/tcplayer.v4.min
        <!-- 示例 CSS 样式可自行删除 -->
    </head>
    <body>
        <!-- 设置播放器容器 -->
        <video id="player-container-id" preload="auto" width="600" height="400" p
        <!--
        注意事项:
        * 播放器容器必须为 video 标签
        * player-container-id 为播放器容器的ID，可自行设置
        * 播放器区域的尺寸请按需设置，建议通过 css 进行设置，通过css可实现容器自适应等效果
        * playsinline webkit-playsinline x5-playsinline 这几个属性是为了在标准移动端浏
        -->
        <script>
            var player = TCPlayer("player-container-id", { /**player-container-id
                fileID: "[redacted]", /**请传入需要播放的视频fileID 必须 */
                appID: "[redacted]", /**请传入点播账号的appID 必须 */
                psign: ""
                /**其他参数请在开发文档中查看 */
                });
        </script>
    </body>
</html>
```

总结

学习本教程后，您已经掌握如何对视频加密，并使用超级播放器播放加密后的视频。

阶段5：播放长视频方案

最近更新时间：2022-03-21 14:31:16

本文针对音视频平台常见的长视频播放场景，推出超级播放器播放长视频教程。用户将掌握如何在 Web 端、iOS 端、Android 端播放器上播放视频，同时自动切换自适应码流、预览视频缩略图、添加视频打点信息。

学习目标

学习本阶段教程，您将掌握：

- 如何在云点播转出自适应码流（播放器能够根据当前带宽，动态选择最合适的码率播放）
- 如何在云点播设置视频打点信息
- 如何在云点播使用雪碧图做缩略图
- 如何使用播放器进行播放

阅读之前，请先确保已经学习超级播放器指引的 阶段1：用超级播放器播放视频 篇部分，了解云点播 fileid 的概念。

操作步骤

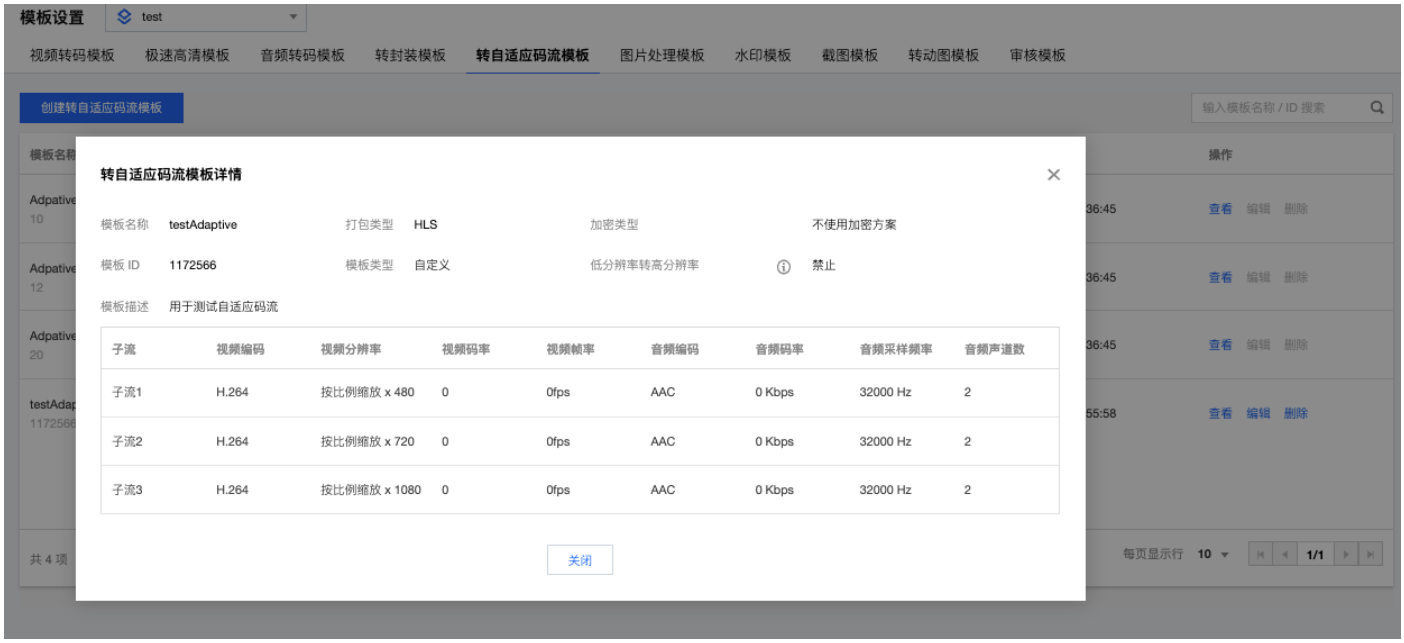
步骤1：转出自适应码流与雪碧图

本步骤，我们将指导您如何对视频转出自适应码流与雪碧图。

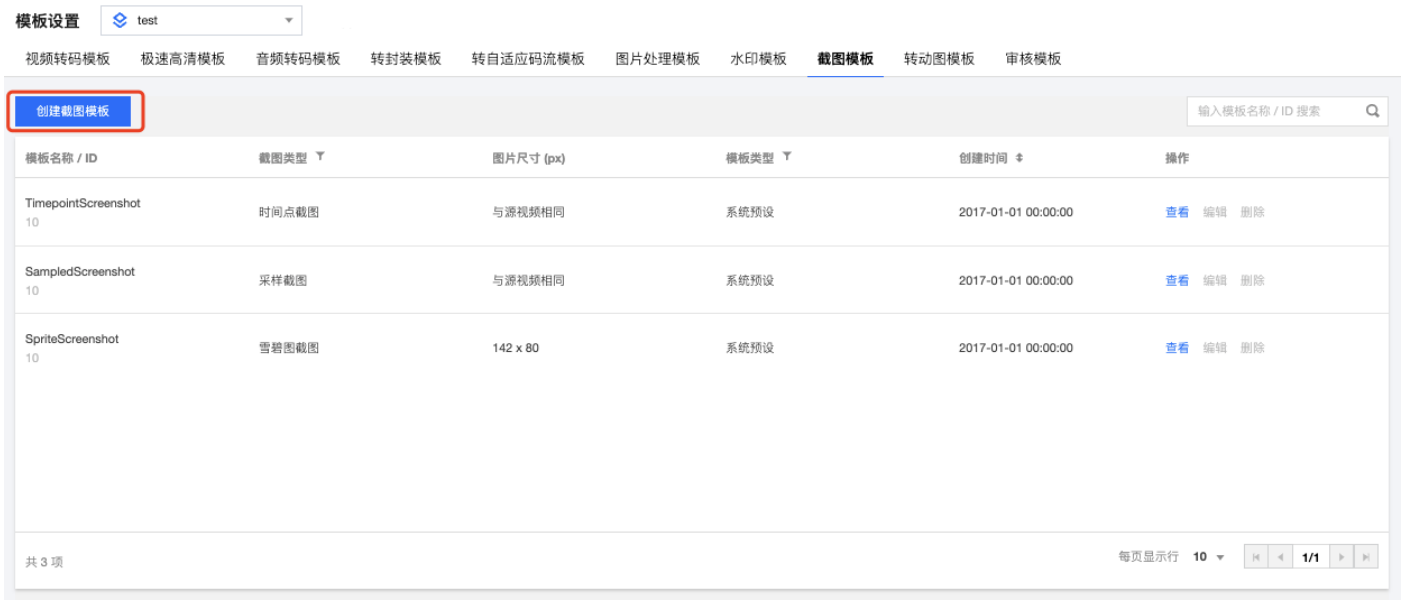
1. 登录 云点播控制台，选择**视频处理设置>模板设置>转自适应码流模板**，单击**创建转自适应码流模板**。

通过模板创建用户需要的自适应码流，本文创建一条名为 testAdaptive 的自适应码流模板，总共包含三条子流，分辨率

分别为480p,720p和1080p。视频码率、视频帧率、音频码率则保持与原视频一致。

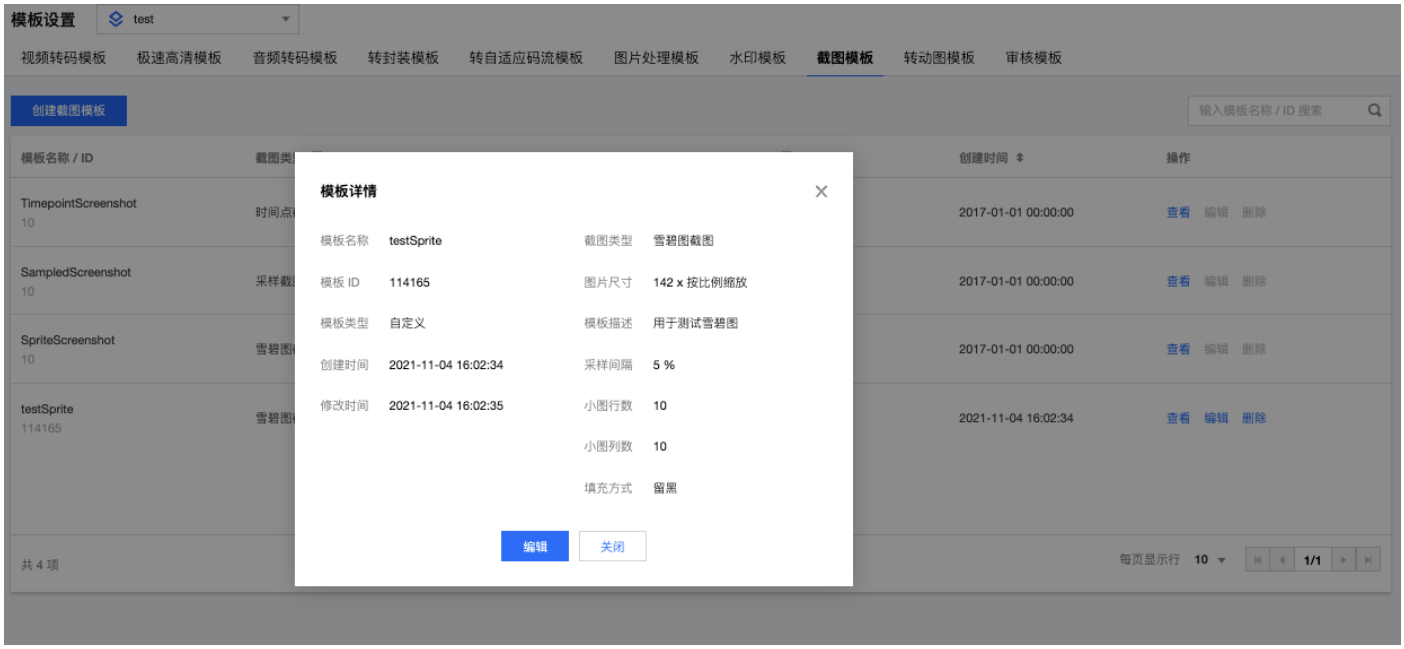


2. 选择**视频处理设置>模板设置>截图模板**，单击**创建截图模板**。



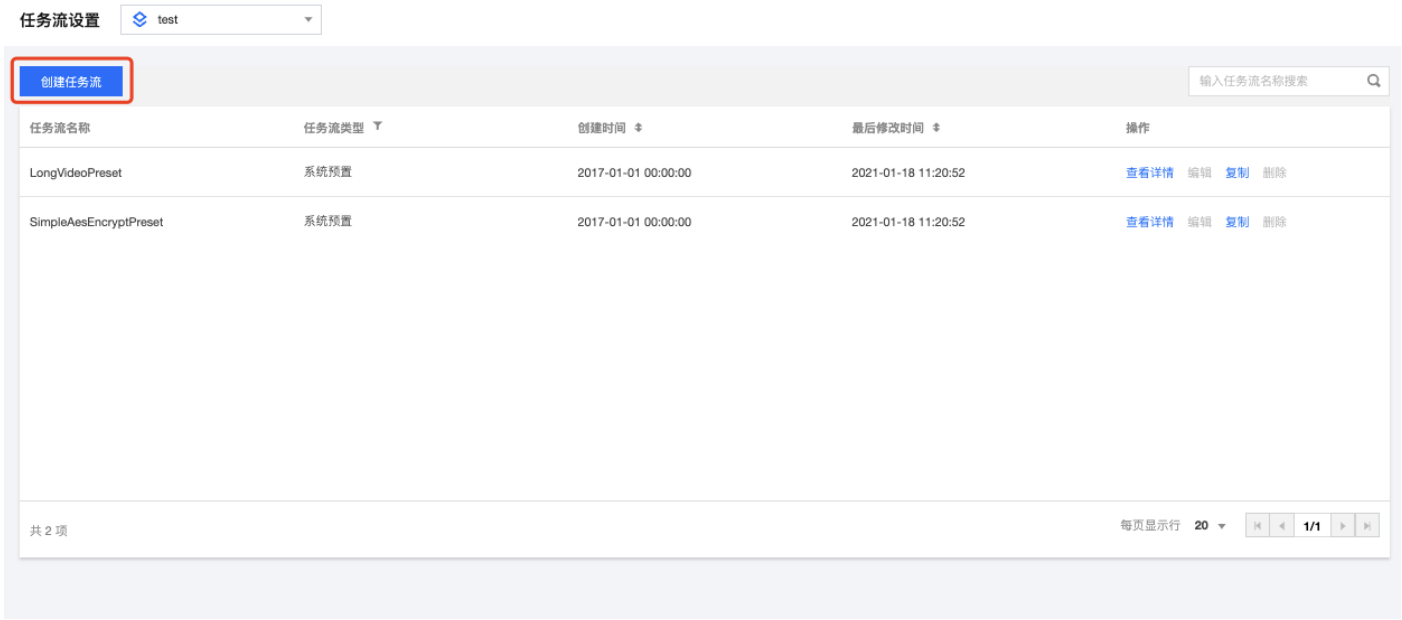
通过模板创建用户需要的雪碧图，本文创建一个名为 testSprite 的雪碧图模板，采样间隔为5%，小图行数：10，小图列

数：10。



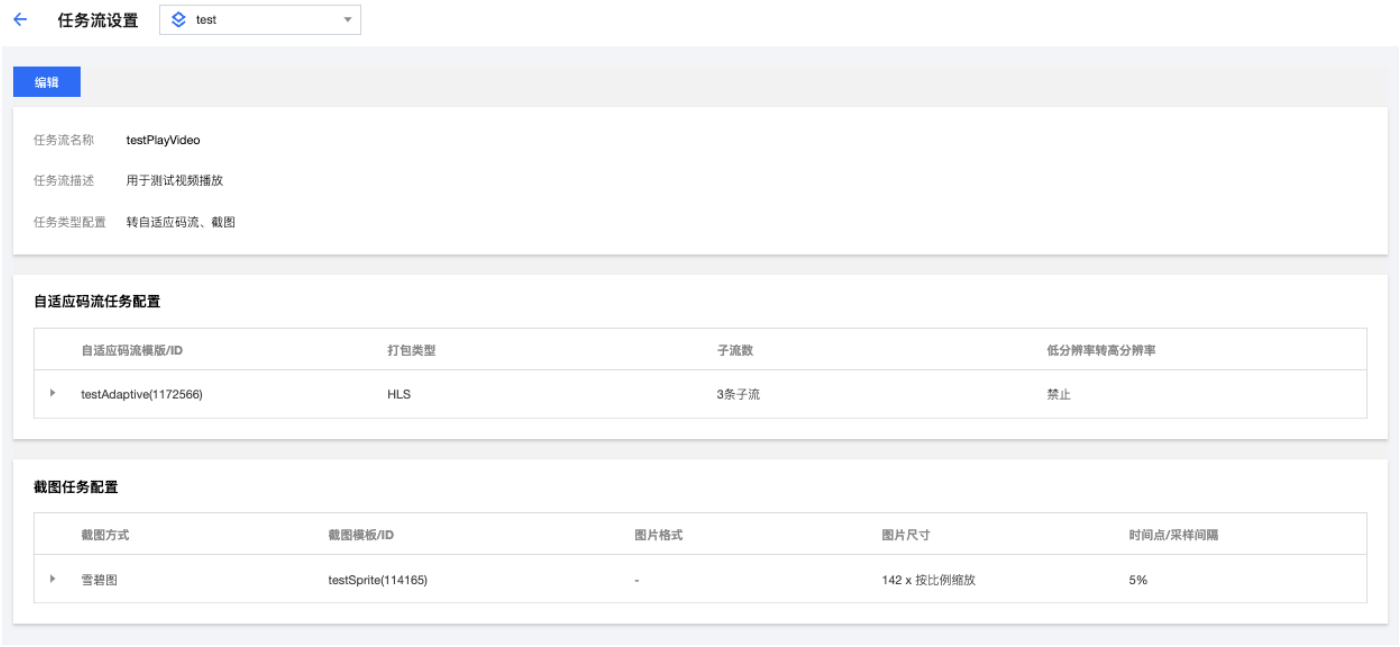
3. 通过任务流，添加自适应码流模板与雪碧图模板。

选择**视频处理设置>任务流设置>创建任务流**，单击**创建任务流**。

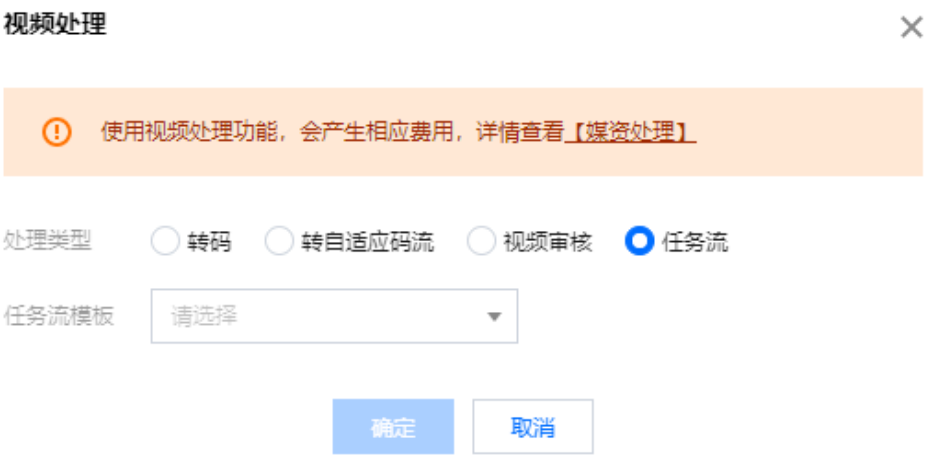


通过任务流，添加用户需要处理的任务，本文为展示播放自适应码流过程，创建了一条 testPlayVideo 的任务流，该任务

流仅增加了自适应码流模板和雪碧图模板。



4. 选择**媒资管理>音视频管理**，勾选需要处理的视频，单击**视频处理>任务流**，选择任务流模板，发起任务。

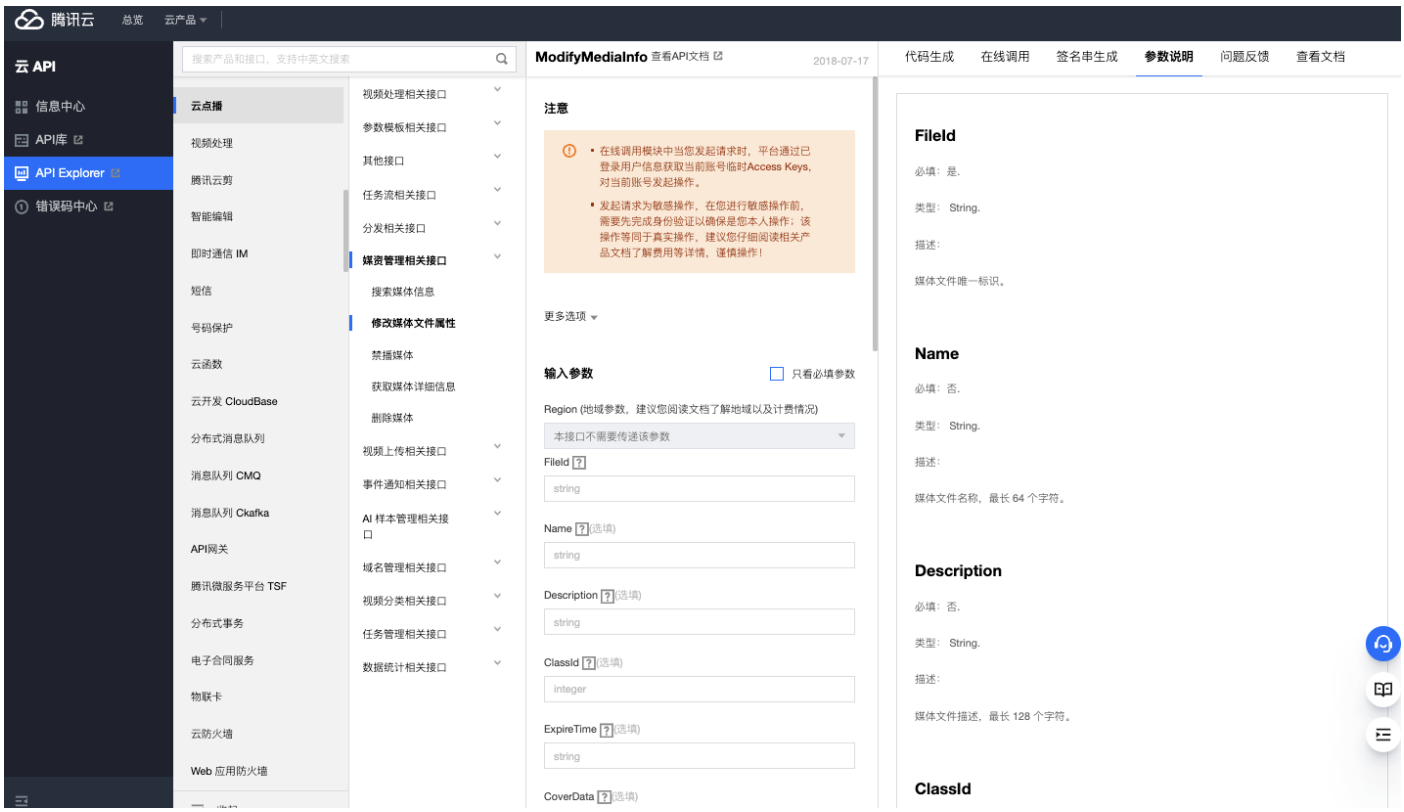


5. 至此，我们可以在**音视频管理>操作>管理**中，能够获取处理后的任务结果。

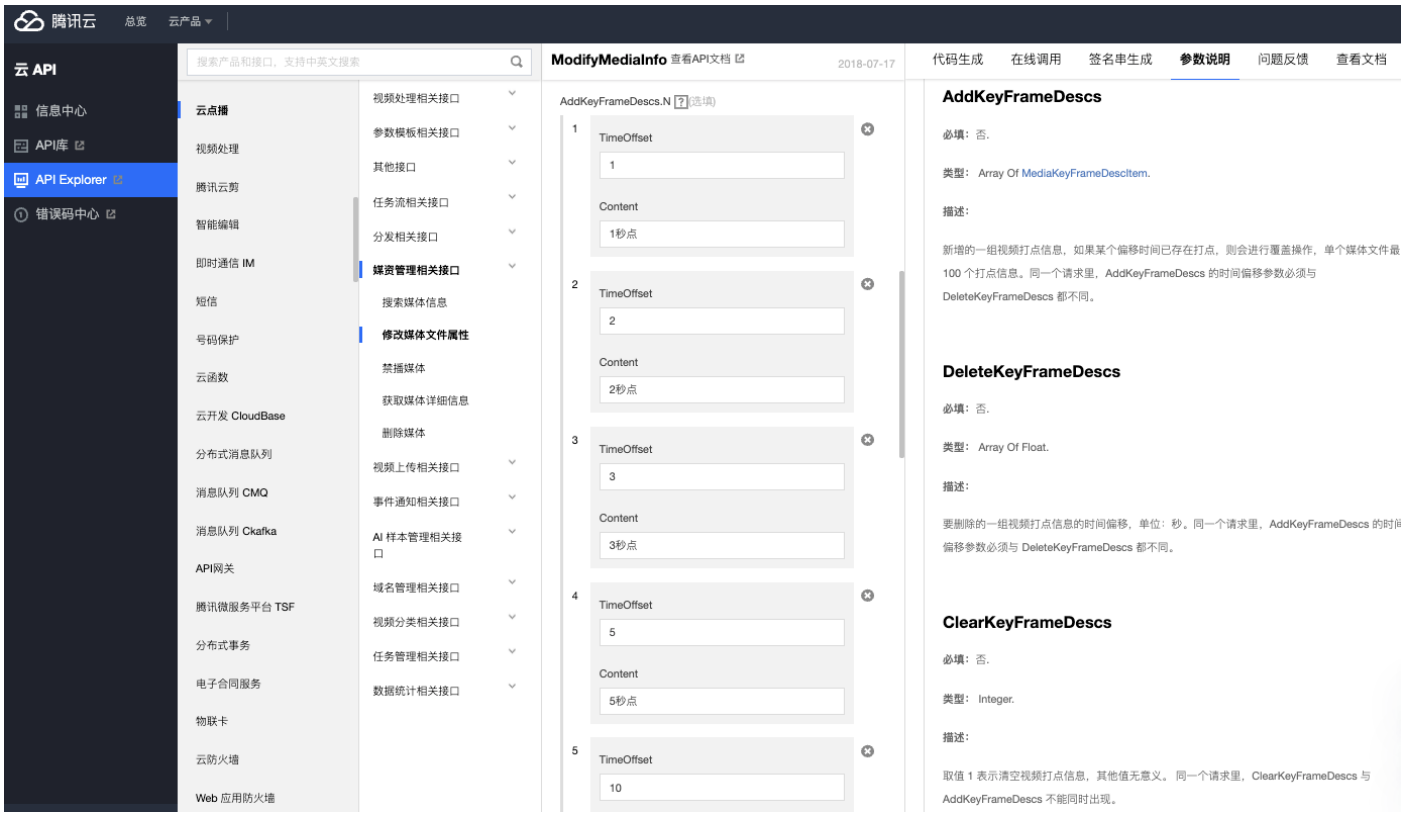
步骤2：增加视频打点信息

本步骤，我们将指导您新增的一组视频打点信息。

1. 进入云点播服务端 API 文档>**媒资管理相关接口**>**修改媒体文件属性**,单击调试，进入云 API 控制台进行调试。



2. 通过参数名称 **AddKeyFrameDescs.N** 添加指定视频打点信息



至此您已经完成了在云端上的操作，此时您在云点播已经转出自适应码流，视频雪碧图和添加了相关视频打点信息。

步骤3：播放器端集成

本步骤，我们将指导您在 Web 端、iOS 端、Android 端播放器播放自适应码流、添加缩略图与打点信息。

Web 端

用户需要集成视立方超级播放器请参见 集成指引，引入播放器 SDK 文件之后，可以**媒资管理**提取上传媒资的 appId 和 fileId 进行播放。

播放器的构建方法为 TCPlayer，通过其创建播放器实例即可播放。

1. 在 html 文件放置播放器容器

在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-playsinlin
e>
</video>
```

2. 使用 fileID 播放

在 index.html 页面初始化的代码中加入以下初始化脚本，传入获取到的 fileID 与 appID 即可播放。

```
var player = TCPlayer('player-container-id', { // player-container-id 为播放器容器 ID，必须与 html 中一致
fileID: '5285890799710670616', // 请传入需要播放的视频 fileID（必须）
appID: '1400329073' // 请传入点播账号的 appID（必须）
});
```

iOS 端

用户需要集成视立方超级播放器请参见 集成指引，集成完后，可以**媒资管理**提取上传媒资的 appId 和 fileId 进行播放。

播放器主类为 SuperPlayerView，创建后即可播放视频：

```
// 引入头文件
#import <SuperPlayer/SuperPlayer.h>

// 创建播放器
_playerView = [[SuperPlayerView alloc] init];
// 设置代理，用于接受事件
_playerView.delegate = self;
// 设置父 View，_playerView 会被自动添加到 holderView 下面
_playerView.fatherView = self.holderView;
```

使用 fileId 播放

```
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1400329073;// 配置 AppId
model.videoId = [[SuperPlayerVideoId alloc] init];
```

```
model.videoId.fileId = @"5285890799710670616"; // 配置 FileId
[_playerView playWithModel:model];
```

Android 端

集成视立方超级播放器请参考集成指引，集成完后，可以**媒资管理**提取上传媒资的 appId 和 fileId 进行播放。

播放器主类为 SuperPlayerView，创建后即可播放视频：

1. 在布局文件创建SuperPlayerView

```
<!-- 超级播放器-->
<com.tencent.liteav.demo.superplayer.SuperPlayerView
android:id="@+id/superVodPlayerView"
android:layout_width="match_parent"
android:layout_height="200dp" />
```

2. 使用 fileId 播放

```
//在布局文件引入SuperPlayerView，然后创建实例
mSuperPlayerView = (SuperPlayerView) findViewById(R.id.superVodPlayerView);
//不开防盗链
SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1400329073;// 配置 AppId
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "5285890799710670616"; // 配置 FileId
mSuperPlayerView.playWithModel(model);
```

总结

至此，您就可以在播放器查看雪碧图预览、视频打点信息和自动切换动态自适应码流。更多功能请参见 功能说明

Demo 体验

最近更新时间：2022-04-06 10:28:01

腾讯云视立方 · 播放器 Player Demo 提供完整的产品级交互界面和业务源码，开发者可基于 Demo 体验播放器各项功能，有助于快速匹配和实现业务需求，节约开发时间和成本。本文将提供超级播放器分别在 Web 端和移动端的体验 Demo，您可以根据当前项目需要进行针对性的调试。

Web端

Web 端播放器支持 PC 端和移动端的浏览器视频播放，丰富灵活的接口可帮助用户快速与自有 Web 应用集成。

Demo 展示

Demo 下载

Web 端超级播放器 – 点播播放 Demo

移动端

腾讯云工具包 App 是腾讯云音视频开发的集多款产品及功能于一身的最佳体验方案。下载腾讯云工具包，选择**腾讯云工具包 > 播放器 Player > 超级播放器**中，您可根据自身需求选择相应功能进行体验。

Demo 展示

Demo 下载

平台	Demo 体验	源码地址
iOS		Github
Android		Github
Flutter	Demo	Github

iOS 端集成
超级播放器
接入指引

最近更新时间：2022-06-01 09:37:48

产品概述

腾讯云视立方 iOS 超级播放器是腾讯云开源的一款播放器组件，简单几行代码即可拥有类似腾讯视频强大的播放功能，包括横竖屏切换、清晰度选择、手势和小窗等基础功能，还支持视频缓存，软硬解切换和倍速播放等特殊功能，相比系统播放器，支持格式更多，兼容性更好，功能更强大，同时还具备首屏秒开、低延迟的优点，以及视频缩略图等高级能力。

若超级播放器组件满足不了您的业务的个性化需求，且您具有一定的开发经验，可以集成 视立方播放器 SDK，自定义开发播放器界面和播放功能。

准备工作

1. 开通 云点播 相关服务，未注册用户可注册账号 试用。
2. 下载 Xcode，如您已下载可略过该步骤，您可以进入 App Store 下载安装。
3. 下载 Cocoapods，如您已下载可略过该步骤，您可以进入 Cocoapods官网 按照指引进行安装。

通过本文您可以学会

- 如何集成腾讯云视立方 iOS 超级播放器
- 如何创建和使用播放器

集成准备

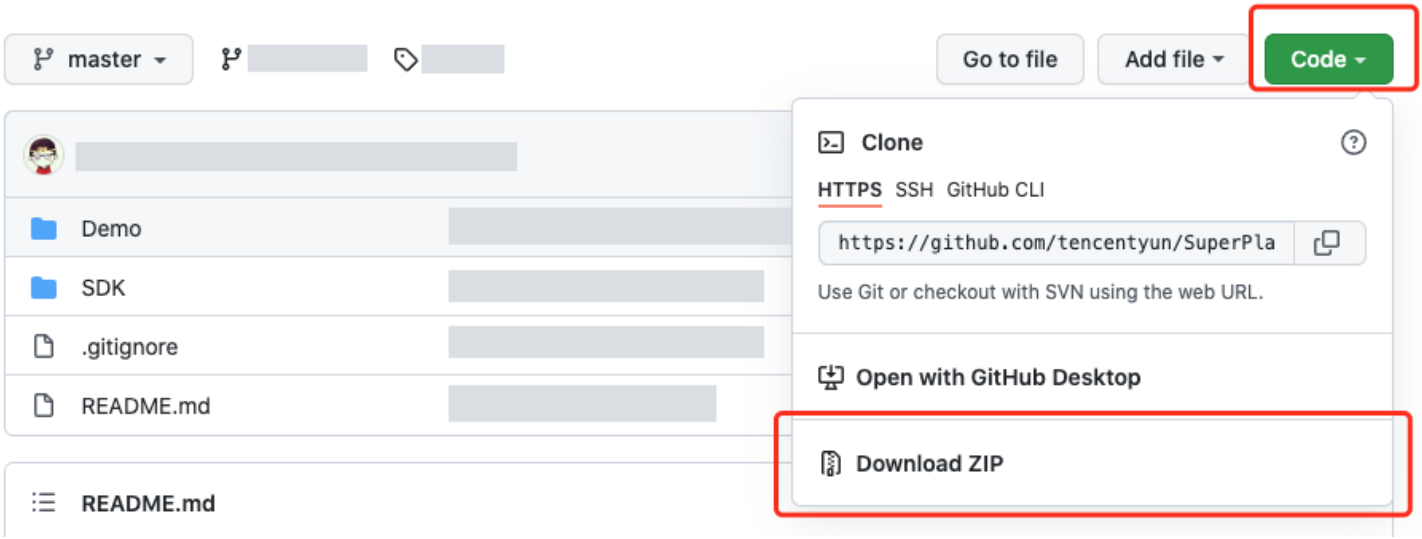
步骤1：项目下载

腾讯云视立方 iOS 超级播放器的项目地址是 SuperPlayer_iOS。

您可通过 **下载播放器组件 ZIP 包** 或 **Git 命令下载** 的方式下载腾讯云视立方 iOS 超级播放器项目工程。

下载播放器组件 ZIP 包

您可以直接下面播放器组件 ZIP包，单击页面的 **Code** > **Download ZIP** 下载。



Git 命令下载

1. 首先确认您的电脑上安装了 Git。如果没有安装，可以参见 Git 安装教程 进行安装。
2. 执行下面的命令把超级播放器组件工程代码 clone 到本地。

```
git clone git@github.com:tencentyun/SuperPlayer_iOS.git
```

提示下面的信息表示成功 clone 工程代码到本地。

```
正克隆到 'SuperPlayer_iOS'...
remote: Enumerating objects: 2637, done.
remote: Counting objects: 100% (644/644), done.
remote: Compressing objects: 100% (333/333), done.
remote: Total 2637 (delta 227), reused 524 (delta 170), pack-reused 1993
接收对象中: 100% (2637/2637), 571.20 MiB | 3.94 MiB/s, 完成.
处理 delta 中: 100% (1019/1019), 完成.
```

下载工程后，工程源码解压后的目录如下：

文件名	作用
SDK	存放播放器的 framework 静态库
Demo	存放超级播放器 Demo
App	程序入口界面
SuperPlayerDemo	超级播放器 Demo
SuperPlayerKit	超级播放器组件

步骤2：集成指引

本步骤，用于指导用户如何集成播放器，推荐用户选择使用 Cocoapods 集成 或者 手动下载 SDK 再将其导入到您当前的工程项目中。

Cocoapods 集成

1. 本项目支持 Cocoapods 安装，只需要将如下代码添加到 Podfile 中：

```
pod 'SuperPlayer'
```

2. 执行 pod install 或 pod update。

手动下载 SDK

1. 下载 SDK + Demo 开发包，腾讯云视立方 iOS 超级播放器项目为 SuperPlayer_iOS。
2. 导入 TXLiteAVSDK_Player.framework 到工程中，并勾选 Do Not Embed。

步骤3：使用播放器功能

本步骤，用于指导用户创建和使用播放器，并使用播放器进行视频播放。

1. 创建播放器：

播放器主类为 SuperPlayerView ，创建后即可播放视频。

```
// 引入头文件
#import <SuperPlayer/SuperPlayer.h>

// 创建播放器
_playerView = [[SuperPlayerView alloc] init];
// 设置代理，用于接受事件
_playerView.delegate = self;
// 设置父 View，_playerView 会被自动添加到 holderView 下面
_playerView.fatherView = self.holderView;
```

2. 配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 - [AppDelegate application:didFinishLaunchingWithOptions:] 中进行如下设置：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
NSString * const licenceURL = @"<获取到的licenseUrl>";
NSString * const licenceKey = @"<获取到的key>";

//TXLiveBase 位于 "TXLiveBase.h" 头文件中
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
```

3. **播放视频：**

本步骤，用于指导用户播放视频，腾讯云视立方 iOS 超级播放器支持 云点播 FileId 或者 使用 URL 进行播放，推荐您选择集成 FileId 使用更完善的能力。

云点播 FileId 播放

视频 FileId 在一般是在视频上传后，由服务器返回：

i. 客户端视频发布后，服务器会返回 FileId 到客户端。

ii. 服务端视频上传时，在 确认上传 的通知中包含对应的 FileId。

如果文件已存在腾讯云，则可以进入 媒资管理 ，找到对应的文件，查看 FileId。如下图所示，ID 即表示 FileId：

注意：

- **通过 FileId 播放时，需要首先使用 Adaptive-HLS(10) 转码模板对视频进行转码，或者使用超级播放器签名 psign 指定播放的视频，否则可能导致视频播放失败。转码教程和说明可参见 用超级播放器播放视频，psign 生成教程可参见 psign 教程。**
- **若您在通过 FileId 播放时出现“no v4 play info”异常，则说明您可能存在上述问题，建议您根据上述教程调整。同时您也可以直接获取源视频播放链接，通过 URL 播放 的方式实现播放。**
- **未经转码的源视频在播放时有可能出现不兼容的情况，建议您使用转码后的视频进行播放。**

```
//在未开启防盗链进行播放的过程中，如果出现了“no v4 play info”异常，建议您使用Adaptive-HLS(10)转码模板对视频进行转码，或直接获取源视频播放链接通过url方式进行播放。

SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1400329071;// 配置 AppId
model.videoId = [[SuperPlayerVideoId alloc] init];
model.videoId.fileId = @"5285890799710173650"; // 配置 FileId
//私有加密播放需填写 psign， psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
//model.videoId.pSign = @"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTQwMDMyOTA3MSwiZmlsZUlkIjoiNTI4NTg5MDc5OTcxMDE3MzY1MCIsImN1cnJlbnRUaW1lU3RhbXAiOjEsImV4cGlyZVRpbWVTdGFtcCI6MjE0NzQ4MzY0NywidXJsQWNjZXNzSW5mbyI6eyJ0IjoiN2ZmZmZmZmYifSwiZHJtTGljZW5zZUluZm8iOnsiZXhwaXJlVGltZVN0YW1wIjoyMTQ3NDgzNjQ3fX0.yJxpnQ2Evp5KZQFfuBBK05BoPpQAzYAWo6liXws-LzU";
[_playerView playWithModel:model];
```

使用 URL 播放

```
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.videoURL = @"http://your_video_url.mp4"; // 配置您的播放视频url
[_playerView playWithModel:model];
```

4. **退出播放：**

当不需要播放器时，调用 resetPlayer 清理播放器内部状态，释放内存。

```
[_playerView resetPlayer];
```

您已完成了腾讯云视立方 iOS 超级播放器的创建、播放视频和退出播放的能力集成。

功能使用

1、全屏播放

超级播放器支持全屏播放，在全屏播放场景内，同时支持锁屏、手势控制音量和亮度、弹幕、截屏、清晰度切换等功能设置。功能效果可在 **腾讯云视立方 App** > **播放器** > **超级播放器** 中体验，单击界面右下角**全屏**即可进入全屏播放界面。

在窗口播放模式下，可通过调用下述接口进入全屏播放模式：

```
- (void)superPlayerFullScreenChanged:(SuperPlayerView *)player {
//用户可在此自定义切换全屏后的逻辑
}
```

全屏播放界面功能介绍

返回窗口模式

通过返回即可返回窗口播放模式，单击后 SDK 处理完全屏切换的逻辑后会触发的代理方法为：

```
// 返回事件
- (void)superPlayerBackAction:(SuperPlayerView *)player;
单击左上角返回按钮触发
// 全屏改变通知
- (void)superPlayerFullScreenChanged:(SuperPlayerView *)player;
```

锁屏

锁屏操作可以让用户进入沉浸式播放状态。单击后由 SDK 自己处理，无回调。

```
// 用户可通过以下接口控制是否锁屏
@property(nonatomic, assign) BOOL isLockScreen;
```

弹幕

打开弹幕功能后屏幕上会有用户发送的文字飘过。

在这里拿到 SPDefaultControlView 对象，播放器 view 初始化的时候去给 SPDefaultControlView 的弹幕按钮设置事件，弹幕内容和弹幕 view 需要用户自己自定义，详细参见 SuperPlayerDemo 下的 CFDanmakuView、CFDanmakuInfo、CFDanmaku。

```
SPDefaultControlView *dv = (SPDefaultControlView *)**self**.playerView.controlView;
[dv.danmakuBtn addTarget:**self** action:**@selector**(danmakuShow:) forControlEvents:UIControlEve
ntTouchUpInside];
```

CFDanmakuView：弹幕的属性在初始化时配置。

```
// 以下属性都是必须配置的--------
// 弹幕动画时间
@property(nonatomic, assign) CGFloat duration;
// 中间上边/下边弹幕动画时间
@property(nonatomic, assign) CGFloat centerDuration;
// 弹幕弹道高度
@property(nonatomic, assign) CGFloat lineHeight;
// 弹幕弹道之间的间距
@property(nonatomic, assign) CGFloat lineMargin;

// 弹幕弹道最大行数
@property(nonatomic, assign) NSInteger maxShowLineCount;

// 弹幕弹道中间上边/下边最大行数
@property(nonatomic, assign) NSInteger maxCenterLineCount;
```

截屏

超级播放器提供播放过程中截取当前视频帧功能，您可以把图片保存起来进行分享。单击截屏按钮后，由 SDK 内部处理，无截屏成功失败的回调，截取到的图片目录为手机相册。

清晰度切换

用户可以根据需求选择不同的视频播放清晰度，如高清、标清或超清等。单击后触发的显示清晰度view以及单击清晰度选项均由 SDK 内部处理，无回调。

2、悬浮窗播放

超级播放器支持悬浮窗小窗口播放，可以在切换到应用内其它页面时，不打断视频播放功能。功能效果可在 **腾讯云视立方 App** > **播放器** > **超级播放器** 中体验，单击界面左上角**返回**，即可体验悬浮窗播放功能。


```
// 如果在竖屏且正在播放的情况下单击返回按钮会触发接口
[SuperPlayerWindowShared setSuperPlayer:self.playerView];
[SuperPlayerWindowShared show];
```

```
// 单击浮窗返回窗口触发的代码接口
SuperPlayerWindowShared.backController = self;
```

3、视频封面

超级播放器支持用户自定义视频封面，用于在视频接收到首帧画面播放回调前展示。功能效果可在 **腾讯云视立方 App** > **播放器** > **超级播放器** > **自定义封面演示** 视频中体验。

- 当超级播放器设置为自动播放模式PLAY_ACTION_AUTO_PLAY时，视频自动播放，此时将在视频首帧加载出来之前展示封面。
- 当超级播放器设置为手动播放模式PLAY_ACTION_MANUAL_PLAY时，需用户单击**播放**后视频才开始播放。在单击**播放**前将展示封面；在单击**播放**后到视频首帧加载出来前也将展示封面。

视频封面支持使用网络 URL 地址或本地 File 地址，使用方式可参见下述指引。若您通过 FileID 的方式播放视频，则可直接在云点播内配置视频封面。

```
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
SuperPlayerVideoId *videoId = [SuperPlayerVideoId new];
videoId.fileId = @"8602268011437356984";
model.appId = 1400329071;
model.videoId = videoId;
//播放模式，可设置自动播放模式：PLAY_ACTION_AUTO_PLAY，手动播放模式：PLAY_ACTION_MANUAL_PLAY
model.action = PLAY_ACTION_MANUAL_PLAY;
//设定封面的地址为网络url地址，如果coverPictureUrl不设定，那么就会自动使用云点播控制台设置的封面
model.customCoverImageUrl = @"http://1500005830.vod2.myqcloud.com/6c9a5118vodcq1500005830/cc1e282086022680110873365l8/MXUW1a5I9TsA.png";
[self.playerView playWithModel:model]
```

4、视频列表轮播

超级播放器支持视频列表轮播，即在给定一个视频列表后：

- **支持按顺序循环播放列表中的视频，播放过程中支持自动播放下一集也支持手动切换到下一个视频。**
- **列表中最后一个视频播放完成后将自动开始播放列表中的第一个视频。**

功能效果可在 腾讯云视立方 App > 播放器 > 超级播放器 > 视频列表轮播演示 视频中体验。


```
//步骤1:构建轮播数据的 NSMutableArray
NSMutableArray *modelArray = [NSMutableArray array];
```

```
SuperPlayerModel *model = [SuperPlayerModel new];
SuperPlayerVideoId *videoId = [SuperPlayerVideoId new];
videoId.fileId = @"8602268011437356984";
model.appId = 1252463788;
model.videoId = videoId;
[modelArray addObject:model];

model = [SuperPlayerModel new];
videoId = [SuperPlayerVideoId new];
videoId.fileId = @"4564972819219071679";
model.appId = 1252463788;
model.videoId = videoId;
[modelArray addObject:model];

//步骤2：调用 SuperPlayerView 的轮播接口
[self.playerView playWithModelList:modelArray isLoopPlayList:YES startIndex:0];
```

```
(void)playWithModelList:(NSArray *)playModelList isLoopPlayList:(BOOL)isLoop startIndex:(NSInteger)ind
ex;
```

接口参数说明

参数名	类型	描述
playModelList	NSArray *	轮播数据列表
isLoop	Boolean	是否循环
index	NSInteger	开始播放的视频索引

5、视频试看

超级播放器支持视频试看功能，可以适用于非 VIP 试看等场景，开发者可以传入不同的参数来控制视频试看时长、提示信息、试看结束界面等。功能效果可在 **腾讯云视立方 App > 播放器 > 超级播放器 > 试看功能演示** 视频中体验。

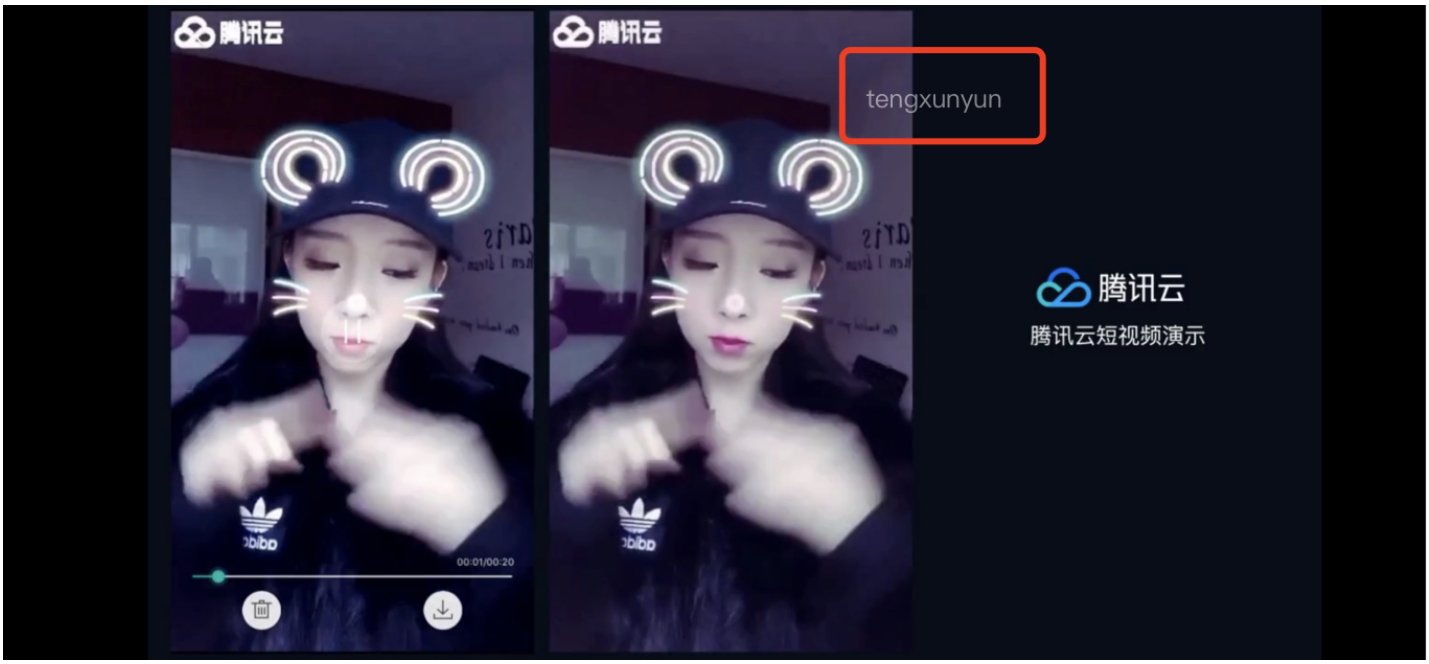

```
//步骤1：创建试看model
TXVipWatchModel *model = [[TXVipWatchModel alloc] init];
model.tipTtitle = @"可试看15秒，开通VIP观看完整视频";
model.canWatchTime = 15;
//步骤2：设置试看model
self.playerView.vipWatchModel = model;
//步骤3：调用方法展示试看功能
[self.playerView showVipTipView];
```

TXVipWatchModel 类参数说明：

参数名	类型	描述
tipTtitle	NSString	试看提示信息
canWatchTime	float	试看时长，单位为妙

6、动态水印

超级播放器支持在播放界面添加不规则跑动的文字水印，有效防盗录。全屏播放模式和窗口播放模式均可展示水印，开发者可修改水印文本、文字大小、颜色。功能效果可在 **腾讯云视立方 App > 播放器 > 超级播放器 > 动态水印演示** 视频中体验。




```
//步骤1：创建视频源信息 model
SuperPlayerModel * playermodel = [SuperPlayerModel new];
//添加视频源其他信息
//步骤2：创建动态水印 model
DynamicWaterModel *model = [[DynamicWaterModel alloc] init];
//步骤3：设置动态水印的数据
model.dynamicWatermarkTip = @"shipinyun";
model.textFont = 30;
model.textColor = [UIColor colorWithRed:255.0/255.0 green:255.0/255.0 blue:255.0/255.0 alpha:0.8];
playermodel.dynamicWaterModel = model;
//步骤4：调用方法展示动态水印
[self.playerView playWithModel:playermodel];
```

DynamicWaterModel 类参数说明：

参数名	类型	描述

参数名	类型	描述
dynamicWatermarkTip	NSString	水印文本信息
textFont	CGFloat	文字大小
textColor	UIColor	文字颜色

Demo 体验

更多完整功能可直接运行工程 Demo，或扫码下载移动端 Demo 腾讯云视立方 App体验。

运行工程 Demo

1. 在 Demo 目录，执行命令行 pod update，重新生成 TXLiteAVDemo.xcworkspace 文件。
2. 双击打开工程，修改证书选择真机运行。
3. 成功运行 Demo 后，进入 **播放器 > 超级播放器**，可体验播放器功能。

腾讯云视立方 App

在 **腾讯云视立方 App > 播放器** 中可体验更多超级播放器功能。

接口说明

最近更新时间：2022-05-18 14:33:03

TXVodPlayer

点播播放器

请参见 TXVodPlayer。

主要负责从指定的点播流地址拉取音视频数据，并进行解码和本地渲染播放。

播放器包含如下能力：

- 支持 FLV、MP4 及 HLS 多种播放格式，支持 基础播放（URL 播放） 和 点播播放（Fileid 播放）两种播放方式 。
- 屏幕截图，可以截取当前播放流的视频画面。
- 通过手势操作，调节亮度、声音、进度等。
- 可以手动切换不同的清晰度，也可根据网络带宽自适应选择清晰度。
- 可以指定不同倍速播放，并开启镜像和硬件加速。
- 完整能力，请参见 点播超级播放器 - 能力清单。

播放器配置接口

API	描述
config	点播配置，配置信息请参见 TXVodPlayConfig。
isAutoPlay	startPlay 后是否立即播放，默认 YES。
token	加密 HLS 的 token。设置此值后，播放器自动在 URL 中的文件名之前增加 voddrm.token.TOKEN TextureView。
loop	是否循环播放 SurfaceView。
enableHWAcceleration	视频渲染回调。（仅硬解支持）

播放基础接口

API	描述
startPlay	播放 HTTP URL 形式地址。
startPlayWithParams	以 fileId 形式播放。
stopPlay	停止播放。
isPlaying	是否正在播放。
pause	暂停播放，停止获取流数据,保留最后一帧画面。
resume	恢复播放，重新获取流数据。

API	描述
seek	跳转到视频流指定时间点，单位秒。
currentPlaybackTime	获取当前播放位置，单位秒。
duration	获取总时长，单位秒。
playableDuration	获取可播放时长，单位秒。
width	获取视频宽度。
height	获取视频高度。
setStartTime	设置播放开始时间。

视频相关接口

API	描述
snapshot	获取当前视频帧图像。 **注意：由于获取当前帧图像是比较耗时的操作，所以截图会通过异步回调出来。**
setMirror	设置镜像。
setRate	设置点播的播放速率，默认1.0。
bitrateIndex	返回当前播放的码率索引。
setBitrateIndex	设置当前正在播放的码率索引，无缝切换清晰度。 清晰度切换可能需要等待一小段时间。
setRenderMode	设置 图像平铺模式。
setRenderRotation	设置 图像渲染角度。

音频相关接口

API	描述
setMute	设置是否静音播放。
setAudioPlayoutVolume	设置音量大小，范围：0 - 100。

事件通知接口

API	描述
delegate	事件回调，建议使用 vodDelegate。

API	描述
vodDelegate	设置播放器的回调。
videoProcessDelegate	视频渲染回调（仅硬解支持）。

TRTC 相关接口

通过以下接口，可以把点播播放器的音视频流通过 TRTC 进行推送，更多 TRTC 服务请参见 TRTC 产品概述。

API	描述
attachTRTC	点播绑定到 TRTC 服务。
detachTRTC	点播解绑 TRTC 服务。
publishVideo	开始推送视频流。
unpublishVideo	取消推送视频流。
publishAudio	开始推送音频流。
unpublishAudio	取消推送音频流。

TXVodPlayListener

腾讯云点播回调通知。

SDK 基础回调

API	描述
onPlayEvent	点播播放事件通知，请参见 播放事件列表、事件参数。
onNetStatus	点播播放器 网络状态通知。

TXVodPlayConfig

点播播放器配置类。

基础配置接口

API	描述
connectRetryCount	设置播放器重连次数。
connectRetryInterval	设置播放器重连间隔，单位秒。
timeout	设置播放器连接超时时间，单位秒。

API	描述
cacheFolderPath	设置点播缓存目录，点播 MP4、HLS 有效。
maxCacheItems	设置缓存文件个数。
playerType	设置播放器类型。
headers	设置自定义 HTTP headers。
enableAccurateSeek	设置是否精确 seek，默认 true。
autoRotate	播放 MP4 文件时，若设为 YES 则根据文件中的旋转角度自动旋转。 旋转角度可在 PLAY_EVT_CHANGE_ROTATION 事件中获得。默认 YES。
smoothSwitchBitrate	平滑切换多码率 HLS，默认 false。
progressInterval	设置进度回调间隔，单位毫秒。
maxBufferSize	最大预加载大小，单位 MB。

错误码表

常规事件

code	事件定义	含义说明
2004	PLAY_EVT_PLAY_BEGIN	视频播放开始（若有转菊花效果，此时将停止）。
2005	PLAY_EVT_PLAY_PROGRESS	视频播放进度，会通知当前播放进度、加载进度和总体时长。
2007	PLAY_EVT_PLAY_LOADING	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件。
2014	PLAY_EVT_VOD_LOADING_END	视频播放 loading 结束，视频继续播放。
2006	PLAY_EVT_PLAY_END	视频播放结束。
2013	PLAY_EVT_VOD_PLAY_PREPARED	播放器已准备完成，可以播放。
2003	PLAY_EVT_RCV_FIRST_I_FRAME	网络接收到首个可渲染的视频数据包（IDR）。
2009	PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2011	PLAY_EVT_CHANGE_ROTATION	MP4 视频旋转角度。

警告事件

code	事件定义	含义说明

code	事件定义	含义说明
-2301	PLAY_ERR_NET_DISCONNECT	网络断连，且经多次重连亦不能恢复,更多重试请自行重启播放。
-2305	PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
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（重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT）。
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-2304	PLAY_ERR_HEVC_DECODE_FAIL	H265 解码失败。
-2303	PLAY_ERR_FILE_NOT_FOUND	播放的文件不存在。

超级播放器 Adapter

最近更新时间：2022-03-09 16:40:33

产品概述

腾讯云视立方 iOS 超级播放器 Adapter 为云点播提供给客户希望使用第三方播放器或自研播放器开放的对接云 PAAS 资源的播放器插件，常用于有自定义播放器功能需求的用户。

SDK下载

腾讯云视立方 iOS 超级播放器 Adapter SDK 和 Demo 项目下载地址是 TXCPlayerAdapterSDK_iOS。

阅读对象

本文档部分内容为腾讯云专属能力，使用前请开通 腾讯云 相关服务，未注册用户可注册账号 免费试用。

集成指引

环境要求

配置支持 HTTP 请求，需要在项目的 info.plist 文件中添加 App Transport Security Settings->Allow Arbitrary Loads 设置为 YES。

组件依赖

添加 GCDWebServer 组件依赖。

```
pod "GCDWebServer", "~> 3.0"
```

GCDWebServer 是一个轻量的 HTTP server，它基于 GCD 并可用于 OS X & iOS，该库还实现了基于 Web 的文件上传以及 WebDAV server 等扩展功能。

使用播放器

变量声明，播放器主类为 TXCPlayerAdapter ，创建后即可播放视频。

fileId 一般是在视频上传后，由服务器返回：

1. 客户端视频发布后，服务器会返回 fileId 到客户端。
2. 服务端视频上传，在 确认上传 的通知中包含对应的 fileId。

如果文件已存在腾讯云，则可以进入 媒资管理 ，找到对应的文件。点开后在右侧视频详情中，可以看到相关参数。

```
NSInteger appId; ////appid 在腾讯云点播申请
NSString *fileId;
//psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
NSString *pSign = self.pSignTextView.text;

TXCPlayerAdapter *adapter = [TXCPlayerAdapter shareAdapterWithAppId:appId];
```

请求视频信息和播放：

```
id<ITXCPlayerAssistorProtocol> assistor = [TXCPlayerAdapter createPlayerAssistorWithFileId:fileId pSign:pSign];
[assistor requestVideoInfo:^(id<ITXCPlayerAssistorProtocol> response, NSError *error) {
if (error) {
NSLog(@"create player assistor error : %@",error);
[self.view makeToast:error.description duration:5.0 position:CSToastPositionBottom];
return;
}
[weakSelf avplayerPlay:response]; //播放视频
}];



- (void)avplayerPlay:(id<ITXCPlayerAssistorProtocol>)response
{
AVPlayerViewController *playerVC = [[AVPlayerViewController alloc] init];
self.playerVC = playerVC;
TXCStreamingInfo *info = response.getStreamingInfo;
AVPlayer *player = [[AVPlayer alloc] initWithURL:[NSURL URLWithString:info.playUrl]];
playerVC.player = player;
playerVC.title = response.getVideoBasicInfo.name;
[self.navigationController pushViewController:playerVC animated:YES];

[player addObserver:self forKeyPath:@"status" options:NSKeyValueObservingOptionNew context:nil];
}
```

使用完后销毁 Player：

```
[TXCPlayerAdapter destroy];
```

SDK 接口说明

初始化 Adatper

初始化 Adapter，单例。

接口

```
+ (instancetype)shareAdapterWithAppId:(NSUInteger)appId;
```

参数说明

appId：填写 appid（如果使用了子应用，则填 subappid）。

销毁 Adatper

销毁 Adapter，当程序退出后调用。

接口

```
+ (void)destroy;
```

创建播放器辅助类

通过播放器辅助类可以获取播放 fileId 相关信息以及处理 DRM 加密接口等。

接口

```
+ (id<ITXCPlayerAssistorProtocol>)createPlayerAssistorWithFileId:(NSString *)fileId
pSign:(NSString *)pSign;
```

参数说明

参数名	类型	描述
fileId	String	要播放的视频 fileId
pSign	String	超级播放器签名

请求视频播放信息

本接口会请求腾讯云点播服务器，获取播放视频的流信息等。

接口

```
- (void)requestVideoInfo:(ITXCRequestVideoInfoCallback)completion;
```

参数说明

参数名	类型	描述
completion	ITXCRequestVideoInfoCallback	异步回调函数

销毁播放器辅助类

销毁辅助类，在退出播放器或者切换了下一个视频播放的时候调用。

接口

```
+ (void)destroyPlayerAssistor:(id<ITXCPlayerAssistorProtocol>)assistor;
```

获取视频的基本信息

获取视频信息， 必须是在 id<ITXCPlayerAssistorProtocol>.requestVideoInfo 回调之后才生效。

接口

```
- (TXCVideoBasicInfo *)getVideoBasicInfo;
```

参数说明

TXCVideoBasicInfo 参数如下：

参数名	类型	描述
name	String	视频名称
size	Int	视频大小，单位：字节
duration	Float	视频时长，单位：秒
description	String	视频描述
coverUrl	String	视频封面

获取视频流信息

获取视频流信息列表，必须是在 id<ITXCPlayerAssistorProtocol>.requestVideoInfo 回调之后才生效。

接口

```
- (TXCStreamingInfo *)getStreamingInfo;
```

参数说明

TXCStreamingInfo 参数如下：

参数名	类型	描述
playUrl	String	播放 URL
subStreams	List	自适应码流子流信息，类型为 TXCSubStreamInfo

TXCSubStreamInfo 参数如下：

参数名	类型	描述
type	String	子流的类型，目前可能的取值仅有 video
width	Int	子流视频的宽，单位：px
height	Int	子流视频的高，单位：px
resolutionName	String	子流视频在播放器中展示的规格名

获取关键帧打点信息

获取视频关键帧打点信息，必须是在 id<ITXCPlayerAssistorProtocol>.requestVideoInfo 回调之后才生效。

接口

```
- (NSArray<TXCKeyFrameDescInfo *> *)getKeyFrameDescInfos;
```

参数说明

TXCKeyFrameDescInfo 参数如下：

参数名	类型	描述
timeOffset	Float	1.1
content	String	"片头开始..."

获取缩略图信息

获取缩略图信息，必须是在 id<ITXCPlayerAssistorProtocol>.requestVideoInfo 回调之后才生效。

接口

```
- (TXCImageSpriteInfo *)getImageSpriteInfo;
```

参数说明

TCXImageSpriteInfo 参数如下：

参数名	类型	描述
imageUrls	List	缩略图下载 URL 数组，类型为 String
webVttUrl	String	缩略图 VTT 文件下载 URL

定制开发
直播场景
接入文档

最近更新时间：2022-06-01 09:38:01

本文档主要介绍使用腾讯云视立方 SDK 实现直播播放的方法。

示例代码

针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

准备工作

1. 为了您更好的产品体验，建议您开通 云直播 相关服务。若您不使用云直播服务，可略过此步骤。
2. 下载 Xcode，如您已下载可略过该步骤，您可以进入 App Store 下载安装。
3. 下载 Cocoapods，如您已下载可略过该步骤，您可以进入 Cocoapods官网 按照指引进行安装。

通过本文你可以学会

- 如何集成腾讯云视立方 iOS 播放器 SDK
- 如何使用播放器 SDK 进行直播播放
- 如何使用播放器 SDK 底层能力实现更多直播播放功能

SDK 集成

步骤1：下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

步骤2：配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 - [AppDelegate application:didFinishLaunchingWithOptions:] 中进行如下设置：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
NSString * const licenceURL = @"<获取到的licenseUrl>";
NSString * const licenceKey = @"<获取到的key>";

//TXLiveBase 位于 "TXLiveBase.h" 头文件中
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
```

步骤3：创建 Player

视频云 SDK 中的 TXLivePlayer 模块负责实现直播播放功能。

```
TXLivePlayer _txLivePlayer = [[TXLivePlayer alloc] init];
```

步骤4：渲染 View

接下来我们要给播放器的视频画面找个地方来显示，iOS 系统中使用 view 作为基本的界面渲染单位，所以您只需要准备一个 view 并调整好布局就可以了。

```
// 用 setupVideoWidget 给播放器绑定决定渲染区域的 view，其首个参数 frame 在 1.5.2 版本后已经被废弃
[_txLivePlayer setupVideoWidget:CGRectMake(0, 0, 0, 0) containView:_myView insertIndex:0];
```

内部原理上讲，播放器并不是直接把画面渲染到您提供的 view（示例代码中的 _myView）上，而是在这个 view 之上创建一个用于 OpenGL 渲染的子视图（subView）。

如果您要调整渲染画面的大小，只需要调整您所常见的 view 的大小和位置即可，SDK 会让视频画面跟着您的 view 的大小和位置进行实时的调整。

如何做动画?

针对 view 做动画是比较自由的，不过请注意此处动画所修改的目标属性应该是 transform 属性而不是 frame 属性。

```
[UIView animateWithDuration:0.5 animations:^{
_myView.transform = CGAffineTransformMakeScale(0.3, 0.3); // 缩小1/3
}];
```

步骤5：启动播放

```
NSString* flvUrl = @"http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
[_txLivePlayer startPlay:flvUrl type:PLAY_TYPE_LIVE_FLV];
```

可选值	枚举值	含义
PLAY_TYPE_LIVE_RTMP	0	传入的 URL 为 RTMP 直播地址

可选值	枚举值	含义
PLAY_TYPE_LIVE_FLV	1	传入的 URL 为 FLV 直播地址
PLAY_TYPE_LIVE_RTMP_ACC	5	低延迟链路地址（仅适合于连麦场景）
PLAY_TYPE_VOD_HLS	3	传入的 URL 为 HLS（m3u8）播放地址

> 关于 HLS（m3u8）
> 在 App 上我们不推荐使用 HLS 这种播放协议播放直播视频源（虽然它很适合用来做点播），因为延迟太高，在 App 上推荐使用 LIVE_FLV 或者 LIVE_RTMP 播放协议。

步骤6：结束播放

结束播放时，如果要推出当前的 UI 界面，要记得用 **removeVideoWidget** 销毁 view 控件，否则会产生内存泄露或闪屏问题。

```
// 停止播放
[_txLivePlayer stopPlay];
[_txLivePlayer removeVideoWidget]; // 记得销毁 view 控件
```

功能使用

本节将介绍常见直播播放功能的使用方式。

1、画面调整

- **view：大小和位置**
  如需修改画面的大小及位置，直接调整 setupVideoWidget 的参数 view 的大小和位置，SDK 会让视频画面跟着您的 view 的大小和位置进行实时的调整。
- **setRenderMode：铺满 or 适应**

可选值	含义
RENDER_MODE_FILL_SCREEN	将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不会留黑边，但可能因为部分区域被裁剪而显示不全。
RENDER_MODE_FILL_EDGE	将图像等比例缩放，适配最长边，缩放后的宽和高都不会超过显示区域，居中显示，画面可能会留有黑边。

- **setRenderRotation：画面旋转**

可选值	含义

可选值	含义
RENDER_ROTATION_PORTRAIT	正常播放（Home 键在画面正下方）
RENDER_ROTATION_LANDSCAPE	画面顺时针旋转270度（Home 键在画面正左方）

最长边填充

完全填充

横屏模式

2、暂停播放

对于直播播放而言，并没有真正意义上的暂停，所谓的直播暂停，只是**画面冻结**和**关闭声音**，而云端的视频源还在不断地更新着，所以当您调用 resume 的时候，会从最新的时间点开始播放，这跟点播是有很大不同的（点播播放器的暂停和继续与播放本地视频文件时的表现相同）。

```
// 暂停
[_txLivePlayer pause];
// 恢复
[_txLivePlayer resume];
```

3、消息接收

此功能可以在推流端将一些自定义 message 随着音视频线路直接下发到观众端，适用场景例如：

（1）冲顶大会：推流端将**题目**下发到观众端，可以做到“音-画-题”完美同步。

（2）秀场直播：推流端将**歌词**下发到观众端，可以在播放端实时绘制出歌词特效，因而不受视频编码的降质影响。

（3）在线教育：推流端将**激光笔**和**涂鸦**操作下发到观众端，可以在播放端实时地划圈划线。

通过如下方案可以使用此功能：

- TXLivePlayConfig 中的 **enableMessage** 开关置为 **YES**。
- TXLivePlayer 通过 **TXLivePlayListener** 监听消息，消息编号：**PLAY_EVT_GET_MESSAGE（2012）**

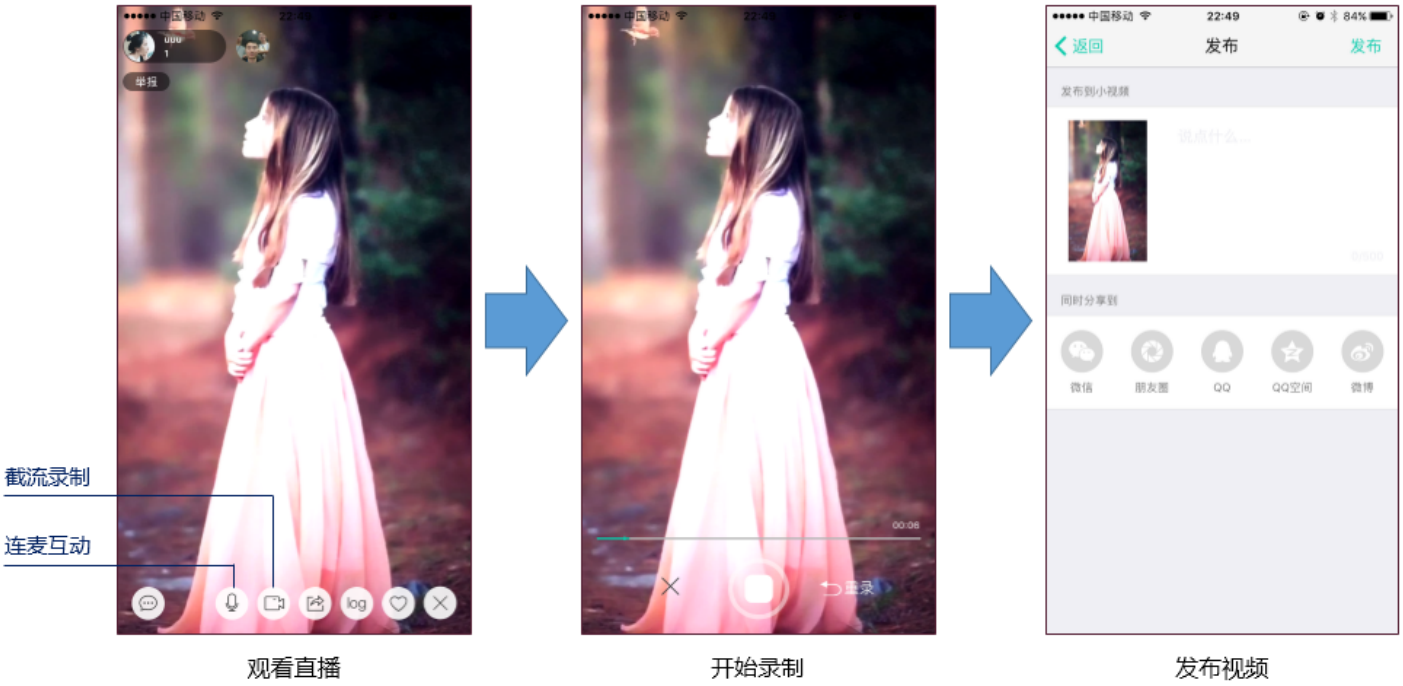
```
-(void) onPlayEvent:(int)EvtID withParam:(NSDictionary *)param {
[self asyncRun:^{
if (EvtID == PLAY_EVT_GET_MESSAGE) {
dispatch_async(dispatch_get_main_queue(), ^{
if ([_delegate respondsToSelector:@selector(onPlayerMessage:)]) {
[_delegate onPlayerMessage:param[@"EVT_GET_MSG"]];
}
});
}
}];
}
```

4、屏幕截图

通过调用 **snapshot** 您可以截取当前直播画面为一帧屏幕，此功能只会截取当前直播流的视频画面，如果您需要截取当前的整个 UI 界面，请调用 iOS 的系统 API 来实现。

5、截流录制

截流录制是直播播放场景下的一种扩展功能：观众在观看直播时，可以通过单击录制按钮把一段直播的内容录制下来，并通过视频分发平台（例如腾讯云的点播系统）发布出去，这样就可以在微信朋友圈等社交平台上以 UGC 消息的形式进行传播。




```
// 如下代码用于展示直播播放场景下的录制功能
//
// 指定一个 TXVideoRecordListener 用于同步录制的进度和结果
_txLivePlayer.recordDelegate = recordListener;
// 启动录制，可放于录制按钮的响应函数里，目前只支持录制视频源，弹幕消息等目前还不支持
[_txLivePlayer startRecord: RECORD_TYPE_STREAM_SOURCE];
// ...
// ...
// 结束录制，可放于结束按钮的响应函数里
[_txLivePlayer stopRecord];
```

- 录制的进度以时间为单位，由 TXVideoRecordListener 的 onRecordProgress 通知出来。
- 录制好的文件以 MP4 文件的形式，由 TXVideoRecordListener 的 onRecordComplete 通知出来。
- 视频的上传和发布由 TXUGCPublish 负责，具体使用方法可以参考 短视频 – 文件发布。

6、清晰度无缝切换

日常使用中，网络情况在不断发生变化。在网络较差的情况下，最好适度降低画质，以减少卡顿；反之，网速比较好，可以观看更高画质。
传统切流方式一般是重新播放，会导致切换前后画面衔接不上、黑屏、卡顿等问题。使用无缝切换方案，在不中断直播的情况下，能直接切到另条流上。

清晰度切换在直播开始后，任意时间都可以调用。调用方式如下

```
// 正在播放的是流http://5815.liveplay.myqcloud.com/live/5815_62fe94d692ab11e791eae435c87f075e.flv，
// 现切换到码率为 900kbps 的新流上
[_txLivePlayer switchStream:@"http://5815.liveplay.myqcloud.com/live/5815_62fe94d692ab11e791eae435c87f075e_900.flv"];
```

> 说明：
> 清晰度无缝切换功能需要在后台配置 PTS 对齐，如您需要可 提交工单 申请使用。

7、直播回看

时移功能是腾讯云推出的特色能力，可以在直播过程中，随时回退到任意直播历史时间点观看，并能在此时间点一直观看直播。非常适合游戏、球赛等互动性不高，但观看连续性较强的场景。

```
// 设置直播回看前，先调用 startPlay
// 开始播放 ...
[TXLiveBase setAppID:@"1253131631"]; // 配置 appId
[_txLivePlayer prepareLiveSeek]; // 后台请求直播起始时间
```

配置正确后，在 PLAY_EVT_PLAY_PROGRESS 事件里，当前进度就不是从0开始，而是根据实际开播时间计算而来。调用 seek 方法，就能从历史事件点重新直播

```
[_txLivePlayer seek:600]; // 从第10分钟开始播放
```

接入时移需要在后台打开2处配置：

1. 录制：配置时移时长、时移储存时长。
2. 播放：时移获取元数据。

> 说明：
> 时移功能处于公测申请阶段，如您需要可 提交工单 申请使用。

8、延时调节

腾讯云 SDK 的直播播放（LVB）功能，并非基于 ffmpeg 做二次开发， 而是采用了自研的播放引擎，所以相比于开源播放器，在直播的延迟控制方面有更好的表现，我们提供了三种延迟调节模式，分别适用于：秀场，游戏以及混合场景。

- 三种模式的特性对比

控制模式	卡顿率	平均延迟	适用场景	原理简述

控制模式	卡顿率	平均延迟	适用场景	原理简述
极速模式	较流畅偏高	2s- 3s	美女秀场（冲顶大会）	在延迟控制上有优势，适用于对延迟大小比较敏感的场景
流畅模式	卡顿率最低	>= 5s	游戏直播（企鹅电竞）	对于超大码率的游戏直播（例如吃鸡）非常适合，卡顿率最低
自动模式	网络自适应	2s-8s	混合场景	观众端的网络越好，延迟就越低；观众端网络越差，延迟就越高

- **三种模式的对接代码**

```
TXLivePlayConfig* _config = [[TXLivePlayConfig alloc] init];
// 自动模式
_config.bAutoAdjustCacheTime = YES;
_config.minAutoAdjustCacheTime = 1;
_config.maxAutoAdjustCacheTime = 5;
// 极速模式
_config.bAutoAdjustCacheTime = YES;
_config.minAutoAdjustCacheTime = 1;
_config.maxAutoAdjustCacheTime = 1;
// 流畅模式
_config.bAutoAdjustCacheTime = NO;
_config.cacheTime = 5;

[_txLivePlayer setConfig:_config];

// 设置完成之后再启动播放
```

> **说明：**
> 更多关于卡顿和延迟优化的技术知识，可以阅读 视频卡顿怎么办？

9、超低延时播放

支持400ms左右的超低延迟播放是腾讯云直播播放器的一个特点，它可以用于一些对时延要求极为苛刻的场景，例如**远程夹娃娃**或者**主播连麦**等，关于这个特性，您需要知道：

- **该功能是不需要开通的**
  该功能并不需要提前开通，但是要求直播流必须位于腾讯云，跨云商实现低延时链路的难度不仅仅是技术层面的。
- **播放地址需要带防盗链**
  播放 URL 不能用普通的 CDN URL， 必须要带防盗链签名，防盗链签名的计算方法见 txTime&txSecret。

- **播放类型需要指定 ACC**
  在调用 startPlay 函数时，需要指定 type 为 **PLAY_TYPE_LIVE_RTMP_ACC**，SDK 会使用 RTMP-UDP 协议拉取直播流。

- **该功能有并发播放限制**
  目前最多同时10路并发播放，设置这个限制的原因并非是技术能力限制，而是希望您只考虑在互动场景中使用（例如连麦时只给主播使用， 或者夹娃娃直播中只给操控娃娃机的玩家使用），避免因为盲目追求低延时而产生不必要的费用损失（低延迟线路的价格要贵于 CDN 线路）。

- **Obs 的延时是不达标的**
  推流端如果是 TXLivePusher，请使用 setVideoQuality 将 quality 设置为 MAIN_PUBLISHER 或者 VIDEO_CHAT。Obs 的推流端积压比较严重，是无法达到低延时效果的。

- **该功能按播放时长收费**
  本功能按照播放时长收费，费用跟拉流的路数有关系，跟音视频流的码率无关，具体价格请参考 价格总览。

10、获取视频信息

播放器 SDK 通过 URL 字符串播放视频，URL 中本身不包含视频信息。为获取相关信息，需要通过访问云端服务器加载到相关视频信息，因此 SDK 只能以事件通知的方式将视频信息发送到您的应用程序中，更多内容参见 事件监听。

例如您可以通过 onNetStatus 的 NET_STATUS_VIDEO_WIDTH 和 NET_STATUS_VIDEO_HEIGHT 获取视频的宽和高。具体使用方法见 状态反馈（onNetStatus）。

事件监听

您可以为 TXLivePlayer 对象绑定一个 TXLivePlayListener，之后 SDK 的内部状态信息均会通过 onPlayEvent（事件通知）和 onNetStatus（状态反馈）通知给您。

事件通知（onPlayEvent）

1. 播放事件

事件 ID	数值	含义说明
PLAY_EVT_CONNECT_SUCC	2001	已经连接服务器
PLAY_EVT_RTMP_STREAM_BEGIN	2002	已经连接服务器，开始拉流（仅播放 RTMP 地址时会抛送）
PLAY_EVT_RCV_FIRST_I_FRAME	2003	网络接收到首个可渲染的视频数据包（IDR）
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始，如果有转菊花什么的这个时候该停了
PLAY_EVT_PLAY_LOADING	2007	视频播放 loading，如果能够恢复，之后会有 BEGIN 事件
PLAY_EVT_GET_MESSAGE	2012	用于接收夹在音视频流中的消息，详情参考 消息接收

- **不要在收到 PLAY_LOADING 后隐藏播放画面**
  因为 PLAY_LOADING -> 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 loading 动画。

2. 结束事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_ERR_NET_DISCONNECT	-2301	网络断连,且经多次重连亦不能恢复,更多重试请自行重启播放

- **如何判断直播已结束?**
  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都要监听，用来作为直播结束的判定事件。

3. 警告事件

如下的这些事件您可以不用关心，我们只是基于白盒化的 SDK 设计理念，将事件信息同步出来

事件 ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连, 已启动自动重连（重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT 了）
PLAY_WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
PLAY_WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解
PLAY_WARNING_VIDEO_DISCONTINUITY	2107	当前视频帧不连续，可能丢帧
PLAY_WARNING_DNS_FAIL	3001	RTMP-DNS 解析失败（仅播放 RTMP 地址时会抛送）
PLAY_WARNING_SEVER_CONN_FAIL	3002	RTMP 服务器连接失败（仅播放 RTMP 地址时会抛送）

事件 ID	数值	含义说明
PLAY_WARNING_SHAKE_FAIL	3003	RTMP 服务器握手失败（仅播放 RTMP 地址时会抛送）

状态反馈（onNetStatus）

通知每秒都会被触发一次，目的是实时反馈当前的推流器状态，它就像汽车的仪表盘，可以告知您目前 SDK 内部的一些具体情况，以便您能对当前网络状况和视频信息等有所了解。

评估参数	含义说明
NET_STATUS_CPU_USAGE	当前瞬时 CPU 使用率
NET_STATUS_VIDEO_WIDTH	视频分辨率 – 宽
NET_STATUS_VIDEO_HEIGHT	视频分辨率 – 高
NET_STATUS_NET_SPEED	当前的网络数据接收速度
NET_STATUS_NET_JITTER	网络抖动情况，抖动越大，网络越不稳定
NET_STATUS_VIDEO_FPS	当前流媒体的视频帧率
NET_STATUS_VIDEO_BITRATE	当前流媒体的视频码率，单位 kbps
NET_STATUS_AUDIO_BITRATE	当前流媒体的音频码率，单位 kbps
NET_STATUS_CACHE_SIZE	缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了
NET_STATUS_SERVER_IP	连接的服务器 IP

API 文档

最近更新时间：2022-05-18 09:46:11

TXLivePlayer

视频播放器

请参见 TXLivePlayer。

主要负责将直播流的音视频画面进行解码和本地渲染，包含如下技术特点：

- 针对腾讯云的拉流地址，可使用低延时拉流，实现直播连麦等相关场景。
- 针对腾讯云的拉流地址，可使用直播时移功能，能够实现直播观看与时移观看的无缝切换。
- 支持自定义的音视频数据处理，让您可以根据项目需要处理直播流中的音视频数据后，进行渲染以及播放。

SDK 基础函数

API	描述
delegate	设置播放回调，见TXLivePlayListener.h文件中的详细定义。
videoProcessDelegate	设置视频处理回调，见TXVideoCustomProcessDelegate.h文件中的详细定义。
audioRawDataDelegate	设置音频处理回调，见TXAudioRawDataDelegate.h文件中的详细定义。
enableHWAcceleration	是否开启硬件加速，默认值：NO。
config	设置 TXLivePlayConfig 播放配置项，见TXLivePlayConfig.h文件中的详细定义。
recordDelegate	设置短视频录制回调，见TXLiveRecordListener.h文件中的详细定义。
isAutoPlay	startPlay 后是否立即播放，默认 YES，只有点播有效。

播放基础接口

API	描述
setupVideoWidget	创建 Video 渲染 View，该控件承载着视频内容的展示。
removeVideoWidget	移除 Video 渲染 Widget。
startPlay	启动从指定 URL 播放 RTMP 音视频流。
stopPlay	停止播放音视频流。
isPlaying	是否正在播放。
pause	暂停播放。

API	描述
resume	继续播放，适用于点播，直播。

视频相关接口

API	描述
setRenderRotation	设置画面的方向。
setRenderMode	设置画面的裁剪模式。
snapshot	截屏。

音频相关接口

API	描述
setMute	设置静音。
setVolume	设置音量。
setAudioRoute	设置声音播放模式（切换扬声器，听筒）。
setAudioVolumeEvaluationListener	设置音量大小回调接口。

直播时移相关接口

API	描述
prepareLiveSeek	直播时移准备，拉取该直播流的起始播放时间。
resumeLive	停止时移播放，返回直播。
seek	–

视频录制相关接口

API	描述
startRecord	开始录制短视频。
stopRecord	结束录制短视频。
setRate	设置播放速率。

更多实用接口

API	描述

API	描述
setLogViewMargin	设置状态浮层 view 在渲染 view 上的边距。
showVideoDebugLog	是否显示播放状态统计及事件消息浮层 view。
switchStream	FLV 直播无缝切换。
callExperimentalAPI	调用实验性 API 接口。

枚举值

枚举	描述
TX_Enum_PlayType	支持的直播和点播类型。

TXLivePlayConfig

腾讯云直播播放器的参数配置模块

请参见 TXLivePlayConfig。

主要负责 TXLivePlayer 对应的参数设置，其中绝大多数设置项在播放开始之后再设置是无效的。

TXLivePlayListener

腾讯云直播播放的回调通知

请参见 TXLivePlayListener。

API	描述
onPlayEvent	直播事件通知。
onNetStatus	网络状态通知。

点播场景
接入文档

最近更新时间：2022-06-01 09:38:11

准备工作

1. 开通 云点播 相关服务，未注册用户可注册账号 试用。
2. 下载 Xcode，如您已下载可略过该步骤，您可以进入 App Store 下载安装。
3. 下载 Cocoapods，如您已下载可略过该步骤，您可以进入 Cocoapods官网 按照指引进行安装。

通过本文你可以学会

- 如何集成腾讯云视立方 iOS 播放器 SDK
- 如何使用播放器 SDK 进行点播播放
- 如何使用播放器 SDK 底层能力实现更多功能

SDK 集成

步骤1：下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

步骤2：配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 -[AppDelegate application:didFinishLaunchingWithOptions:] 中进行如下设置：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchO
ptions {
NSString * const licenceURL = @"<获取到的licenseUrl>";
NSString * const licenceKey = @"<获取到的key>";

//TXLiveBase 位于 "TXLiveBase.h" 头文件中
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
```

步骤3：创建 Player

视频云 SDK 中的 TXVodPlayer 模块负责实现点播播放功能。

```
TXVodPlayer *_txVodPlayer = [[TXVodPlayer alloc] init];
[_txVodPlayer setupVideoWidget:_myView insertIndex:0]
```

步骤4：渲染 View

接下来我们要给播放器的视频画面找个地方来显示，iOS 系统中使用 view 作为基本的界面渲染单位，所以您只需要准备一个 view 并调整好布局就可以了。

```
[_txVodPlayer setupVideoWidget:_myView insertIndex:0]
```

内部原理上讲，播放器并不是直接把画面渲染到您提供的 view（示例代码中的 _myView）上，而是在这个 view 之上创建一个用于 OpenGL 渲染的子视图（subView）。

如果您要调整渲染画面的大小，只需要调整您所常见的 view 的大小和位置即可，SDK 会让视频画面跟着您的 view 的大小和位置进行实时的调整。

如何做动画

针对 view 做动画是比较自由的，不过请注意此处动画所修改的目标属性应该是 transform 属性而不是 frame 属性。

```
[UIView animateWithDuration:0.5 animations:^{
_myView.transform = CGAffineTransformMakeScale(0.3, 0.3); // 缩小1/3
}];
```

步骤5：启动播放

TXVodPlayer 支持两种播放模式，您可以根据需要自行选择：

通过 URL 方式

TXVodPlayer 内部会自动识别播放协议，您只需要将您的播放 URL 传给 startPlay 函数即可。

```
NSString* url = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
[_txVodPlayer startPlay:url];
```

通过 fileId 方式

```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788;
p.fileId = @"4564972819220421305";
[_txVodPlayer startPlayWithParams:p];
```

在 媒资管理 找到对应的文件。点开后在右侧视频详情中，可以看到 fileId。
通过 fileId 方式播放，播放器会向后台请求真实的播放地址。如果此时网络异常或 fileId 不存在，则会收到 PLAY_ERR_GET_PLAYINFO_FAIL事件，反之收到PLAY_EVT_GET_PLAYINFO_SUCC表示请求成功。

步骤6：结束播放

结束播放时，如果要退出当前的 UI 界面，要记得用 **removeVideoWidget** 销毁 view 控件，否则会产生内存泄露或闪屏问题。

```
// 停止播放
[_txVodPlayer stopPlay];
[_txVodPlayer removeVideoWidget]; // 记得销毁 view 控件
```

基础功能使用

1、播放控制

开始播放

```
// 开始播放
[_txVodPlayer startPlay:url];
```

暂停播放

```
// 暂停播放
[_txVodPlayer pause];
```

恢复播放

```
// 恢复播放
[_txVodPlayer resume];
```

结束播放

```
// 结束播放
[_txVodPlayer stopPlay];
```

调整进度（Seek）

当用户拖拽进度条时，可调用 seek 从指定位置开始播放，播放器 SDK 支持精准 seek。

```
int time = 600; // int类型时，单位为 秒
// 调整进度
[_txVodPlayer seek:time];
```

从指定时间开始播放

首次调用 startPlay 之前，支持从指定时间开始播放。

```
float startTimeInMS = 600; // 单位：毫秒
[_txVodPlayer setStartTime:startTimeInMS]; // 设置开始播放时间
[_txVodPlayer startPlay:url];
```

2、画面调整

- **view：大小和位置**
  如需修改画面的大小及位置，直接调整 setupVideoWidget 的参数 view 的大小和位置，SDK 会让视频画面跟着您的 view 的大小和位置进行实时的调整。
- **setRenderMode：铺满或适应**

可选值	含义
RENDER_MODE_FILL_SCREEN	将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不会留黑边，但可能因为部分区域被裁剪而显示不全。
RENDER_MODE_FILL_EDGE	将图像等比例缩放，适配最长边，缩放后的宽和高都不会超过显示区域，居中显示，画面可能会留有黑边。

- **setRenderRotation：画面旋转**

可选值	含义
HOME_ORIENTATION_RIGHT	home 在右边
HOME_ORIENTATION_DOWN	home 在下面
HOME_ORIENTATION_LEFT	home 在左边
HOME_ORIENTATION_UP	home 在上面

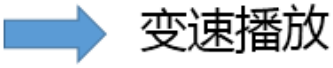
最长边填充

完全填充

横屏模式

3、变速播放

点播播放器支持变速播放，通过接口 setRate 设置点播播放速率来完成，支持快速与慢速播放，如0.5X、1.0X、1.2X、2X等。



支持加速和慢播

```
// 设置1.2倍速播放
[_txVodPlayer setRate:1.2];
// ...
// 开始播放
[_txVodPlayer startPlay:url];
```

4、循环播放

```
// 设置循环播放
[_txVodPlayer setLoop:true];
// 获取当前循环播放状态
[_txVodPlayer loop];
```

5、静音设置

```
// 设置静音，true 表示开启静音， false 表示关闭静音
[_txVodPlayer setMute:true];
```

6、屏幕截图

通过调用 **snapshot** 您可以截取当前视频为一帧画面，此功能只会截取当前直播流的视频画面，如果您需要截取当前的整个 UI 界面，请调用 iOS 的系统 API 来实现。

7、贴片广告

播放器 SDK 支持在界面添加图片贴片，用于广告宣传等。实现方式如下：

- 将autoPlay为 NO，此时播放器会正常加载，但视频不会立刻开始播放。
- 在播放器加载出来后、视频尚未开始时，即可在播放器界面查看图片贴片广告。
- 待达到广告展示结束条件时，使用 resume 接口启动视频播放。

8、HTTP-REF

TXVodPlayConfig 中的 headers 可以用来设置 HTTP 请求头，例如常用的防止 URL 被到处拷贝的 Referer 字段（腾讯云可以提供更加安全的签名防盗链方案），以及用于验证客户端身份信息的 Cookie 字段。

9、硬件加速

对于蓝光级别（1080p）的画质，简单采用软件解码的方式很难获得较为流畅的播放体验，所以如果您的场景是以游戏直播为主，一般都推荐开启硬件加速。

软解和硬解的切换需要在切换之前先 **stopPlay**，切换之后再 **startPlay**，否则会产生比较严重的花屏问题。

```
[_txVodPlayer stopPlay];
_txVodPlayer.enableHWAcceleration = YES;
[_txVodPlayer startPlay:_flvUrl type:_type];
```

10、清晰度设置

SDK 支持 hls 的多码率格式，方便用户切换不同码率的播放流。在收到 PLAY_EVT_PLAY_BEGIN 事件后，可以通过下面方法获取多码率数组

```
NSArray *bitrates = [_txVodPlayer supportedBitrates]; //获取多码率数组
```

在播放过程中，可以随时通过 -[TXVodPlayer setBitrateIndex:] 切换码率。切换过程中，会重新拉取另一条流的数据，因此会有稍许卡顿。SDK 针对腾讯云的多码率文件做过优化，可以做到切换无卡顿。

11、码流自适应

SDK 支持 HLS 的多码流自适应，开启相关能力后播放器能够根据当前带宽，动态选择最合适的码率播放。在收到 PLAY_EVT_PLAY_BEGIN 事件后，可以通过下面方法开启码流自适应：

```
[_txVodPlayer setBitrateIndex:-1]; //index 参数传入-1
```

在播放过程中，可以随时通过 -[TXVodPlayer setBitrateIndex:] 切换其它码率，切换后码流自适应也随之关闭。

12、播放进度监听

点播播放中的进度信息分为2种：**加载进度**和**播放进度**，SDK 目前是以事件通知的方式将这两个进度实时通知出来的。更多事件通知内容参见 事件监听。


```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*)param {
if (EvtID == PLAY_EVT_PLAY_PROGRESS) {
// 加载进度，单位是秒，小数部分为毫秒
float playable = [param[EVT_PLAYABLE_DURATION] floatValue];
[_loadProgressBar setValue:playable];

// 播放进度，单位是秒，小数部分为毫秒
float progress = [param[EVT_PLAY_PROGRESS] floatValue];
[_seekProgressBar setValue:progress];

// 视频总长，单位是秒，小数部分为毫秒
float duration = [param[EVT_PLAY_DURATION] floatValue];
// 可以用于设置时长显示等等
}
}
```

13、播放网速监听

通过 事件监听 方式，可以在视频播放卡顿时在显示当前网速。

- 通过onNetStatus的NET_SPEED获取当前网速。具体使用方法见 状态反馈（onNetStatus）。
- 监听到PLAY_EVT_PLAY_LOADING事件后，显示当前网速。
- 收到PLAY_EVT_VOD_LOADING_END事件后，对显示当前网速的 view 进行隐藏。

14、获取视频分辨率

播放器 SDK 通过 URL 字符串播放视频，URL 中本身不包含视频信息。为获取相关信息，需要通过访问云端服务器加载到相关视频信息，因此 SDK 只能以事件通知的方式将视频信息发送到您的应用程序中，更多内容参见事件监听。

分辨率信息

方法1

通过 onNetStatus 的 VIDEO_WIDTH 和 VIDEO_HEIGHT 获取视频的宽和高。具体使用方法见状态反馈（onNetStatus）。

方法2

直接调用 -[TXVodPlayer width] 和 -[TXVodPlayer height] 获取当前宽高。

15、播放缓冲大小

在视频正常播放时，控制提前从网络缓冲的最大数据大小。如果不配置，则走播放器默认缓冲策略，保证流畅播放。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxBufferSize:10]; // 播放时最大缓冲大小。单位：MB
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

16、视频本地缓存

在短视频播放场景中，视频文件的本地缓存是很刚需的一个特性，对于普通用户而言，一个已经看过的视频再次观看时，不应该再消耗一次流量。

- **格式支持：**SDK 支持 HLS（m3u8）和 MP4 两种常见点播格式的缓存功能。
- **开启时机：**SDK 并不默认开启缓存功能，对于用户回看率不高的场景，也并不推荐您开启此功能。
- **开启方法：**开启此功能需要配置两个参数：本地缓存目录及缓存大小。

```
//设置播放引擎的全局缓存目录
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask,
YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *preloadDataPath = [documentsDirectory stringByAppendingPathComponent:@"/preload"];
if (![[NSFileManager defaultManager] fileExistsAtPath:preloadDataPath]) {
[[NSFileManager defaultManager] createDirectoryAtPath:preloadDataPath
withIntermediateDirectories:NO
attributes:nil
error:&error];
[TXPlayerGlobalSetting setCacheFolderPath:preloadDataPath];
//设置播放引擎缓存大小
[TXPlayerGlobalSetting setMaxCacheSize:200];
// ...
// 开始播放
[_txVodPlayer startPlay:playUrl];
```

> 说明：
> 旧版本通过 TXVodPlayConfig#setMaxCacheItems 接口配置已经废弃，不推荐使用。

高级功能使用

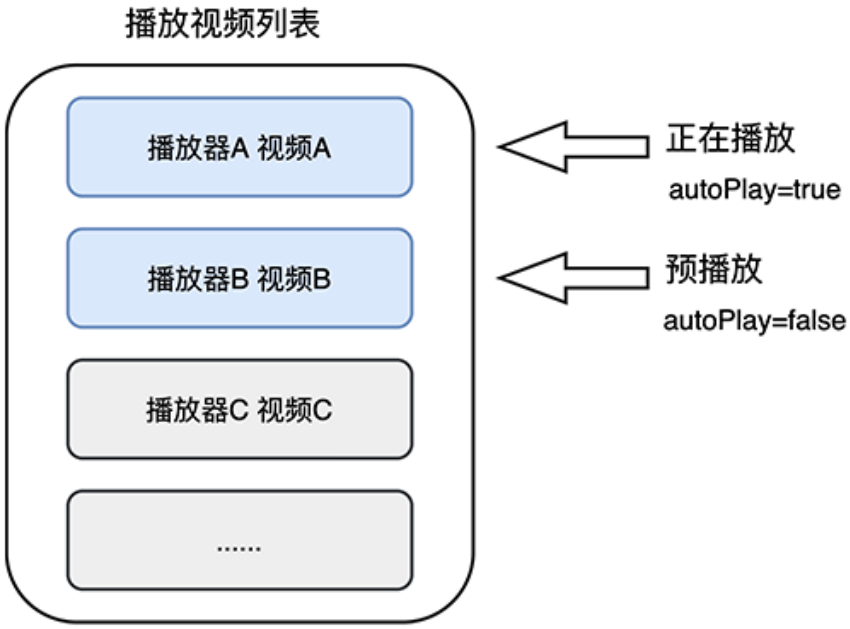
1、视频预播放

步骤1：视频预播放使用

在短视频播放场景中，预加载功能对于流畅的观看体验很有帮助：在观看当前视频的同时，在后台加载即将要播放的下一个视频 URL，这样一来，当用户真正切换到下一个视频时，已经不需要从头开始加载了，而是可以做到立刻播放。

预播放视频会有很好的秒开效果，但有一定的性能开销，如果业务同时有较多的视频预加载需求，建议结合 视频预下载 一起使用。

这就是视频播放中无缝切换的背后技术支撑，您可以使用 TXVodPlayer 中的 isAutoPlay 开关来实现这个功能，具体做法如下：




```
// 播放视频 A: 如果将 isAutoPlay 设置为 YES， 那么 startPlay 调用会立刻开始视频的加载和播放
NSString* url_A = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f10.mp4";
_player_A.isAutoPlay = YES;
[_player_A startPlay:url_A];

// 在播放视频 A 的同时，预加载视频 B，做法是将 isAutoPlay 设置为 NO
NSString* url_B = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
_player_B.isAutoPlay = NO;
[_player_B startPlay:url_B];
```

等到视频 A 播放结束，自动（或者用户手动切换到）视频 B 时，调用 resume 函数即可实现立刻播放。

> ⚠ 注意：
> 设置了 autoPlay 为 false 之后，调用 resume 之前需要保证视频 B 已准备完成，即需要在监听到视频 B 的 PLAY_EVT_VOD_PLAY_PREPARED（2013，播放器已准备完成，可以播放）事件后调用。

```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*)param
{
// 在视频 A 播放结束的时候，直接启动视频 B 的播放，可以做到无缝切换
if (EvtID == PLAY_EVT_PLAY_END) {
[_player_A stopPlay];
[_player_B setupVideoWidget:mVideoContainer insertIndex:0];
[_player_B resume];
}
}
```

步骤2：视频预播放缓冲配置

- 设置较大的缓冲可以更好的应对网络的波动，达到流畅播放的目的。
- 设置较小的缓冲可以帮助节省流量消耗。

预播放缓冲大小

此接口针对预加载场景（即在视频启播前，且设置 player 的 AutoPlay 为 false），用于控制启播前阶段的最大缓冲大小。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxPreloadSize:(2)];; // 预播放最大缓冲大小。单位：MB, 根据业务情况设置去节省流量
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

播放缓冲大小

在视频正常播放时，控制提前从网络缓冲的最大数据大小。如果不配置，则走播放器默认缓冲策略，保证流畅播放。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMaxBufferSize:10]; // 播放时最大缓冲大小。单位：MB
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

2、视频预下载

不需要创建播放器实例，预先下载视频部分内容，使用播放器时，可以加快视频启播速度，提供更好的播放体验。

在使用播放服务前，请确保先设置好视频缓存。

说明：

- TXPlayerGlobalSetting 是全局缓存设置接口，原有 TXVodConfig 的缓存配置接口废弃。
- 全局缓存目录和大小设置的优先级高于播放器 TXVodConfig 配置的缓存设置。

使用示例：

```
//设置播放引擎的全局缓存目录
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *preloadDataPath = [documentsDirectory stringByAppendingPathComponent:@"/preload"];
if (![[NSFileManager defaultManager] fileExistsAtPath:preloadDataPath]) {
[[NSFileManager defaultManager] createDirectoryAtPath:preloadDataPath
withIntermediateDirectories:NO
attributes:nil
error:&error]; //Create folder
}
[TXPlayerGlobalSetting setCacheFolderPath:preloadDataPath];

//设置播放引擎缓存大小
[TXPlayerGlobalSetting setMaxCacheSize:200];
NSString *m3u8url = "http://****";
int taskID = [[TXVodPreloadManager sharedManager] startPreload:m3u8url
preloadSize:10
preferredResolution:1920*1080
delegate:self];


//取消预下载
[[TXVodPreloadManager sharedManager] stopPreload:taskID];
```

3、视频下载

视频下载支持用户在有网络的条件下下载视频，随后在无网络的环境下观看。同时播放器 SDK 提供本地加密能力，下载后的本地视频仍为加密状态，仅可通过指定播放器对视频进行解密播放，可有效防止下载视频的非法传播，保护视频安全。

由于 HLS 流媒体无法直接保存到本地，因此也无法通过播放本地文件的方式实现 HLS 离线播放，对于该问题，您可以通过基于 TXVodDownloadManager 的视频下载方案实现 HLS 的离线播放。

注意：

- TXVodDownloadManager 暂不支持缓存 MP4 和 FLV 格式的文件，仅支持缓存 HLS 格式文件。
- 播放器 SDK 已支持 MP4 和 FLV 格式的本地文件播放。

步骤1：准备工作

TXVodDownloadManager 被设计为单例，因此您不能创建多个下载对象。用法如下：

```
TXVodDownloadManager *downloader = [TXVodDownloadManager shareInstance];
[downloader setDownloadPath:"<指定您的下载目录>"];
```

步骤2：开始下载

开始下载有两种方式：URL 和 fileid。

URL 方式

只需要传入下载地址即可。

```
[downloader startDownloadUrl:@"http://1253131631.vod2.myqcloud.com/26f327f9vodgzp1253131631/f4bdff799031868222924043041/playlist.m3u8"]
```

fileid 方式

fileid 下载至少需要传入 appId 和 fileId。

```
TXPlayerAuthParams *auth = [TXPlayerAuthParams new];
auth.appId = 1252463788;
auth.fileId = @"4564972819220421305";
TXVodDownloadDataSource *dataSource = [TXVodDownloadDataSource new];
dataSource.auth = auth;
[downloader startDownload:dataSource];
```

步骤3：任务信息

在接收任务信息前，需要先设置回调 delegate。

```
downloader.delegate = self;
```

可能收到的任务回调有：

回调信息	含义
-[TXVodDownloadDelegate onDownloadStart:]	任务开始，表示 SDK 已经开始下载

回调信息	含义
-[TXVodDownloadDelegate onDownloadProgress:]	任务进度，下载过程中，SDK 会频繁回调此接口，您可以在这里更新进度显示
-[TXVodDownloadDelegate onDownloadStop:]	任务停止，当您调用是stopDownload停止下载，收到此消息表示停止成功
-[TXVodDownloadDelegate onDownloadFinish:]	下载完成，收到此回调表示已全部下载。此时下载文件可以给 TXVodPlayer 播放
-[TXVodDownloadDelegate onDownloadError:errorMsg:]	下载错误，下载过程中遇到网络断开会回调此接口，同时下载任务停止。所有错误码请参考 TXDownloadError

由于 downloader 可以同时下载多个任务，所以回调接口里带上了 TXVodDownloadMediaInfo 对象，您可以访问 URL 或 dataSource 判断下载源，同时还可以获取到下载进度、文件大小等信息。

步骤4：中断下载

停止下载请调用 -[TXVodDownloadManager stopDownload:] 方法，参数为 -[TXVodDownloadManager sartDownloadUrl:] 返回的对象。**SDK 支持断点续传**，当下载目录没有发生改变时，下次下载同一个文件时会从上次停止的地方重新开始。

如果您不需要重新下载，请调用 -[TXVodDownloadManager deleteDownloadFile:] 方法删除文件，以释放存储空间。

4、加密播放

视频加密方案主要用于在线教育等需要对视频版权进行保护的场景。如果要对您的视频资源进行加密保护，就不仅需要在播放器上做改造，还需要对视频源本身进行加密转码，亦需要您的后台和终端研发工程师都参与其中。在 视频加密解决方案 中您会了解到全部细节内容。

5、播放器配置

在调用 statPlay 之前可以通过 setConfig 对播放器进行参数配置，比如：设置播放器连接超时时间、设置进度回调间隔、设置缓存文件个数等配置，TXVodPlayConfig 支持配置的详细参数请点击基础配置接口了解。使用示例：

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setEnableAccurateSeek:true]; // 设置是否精确 seek，默认 true
[_config setMaxCacheItems:5]; // 设置缓存文件个数为5
[_config setProgressInterval:200]; // 设置进度回调间隔，单位毫秒
[_config setMaxBufferSize:50]; // 最大预加载大小，单位 MB
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

启播时指定分辨率

播放 HLS 的多码率视频源，如果你提前知道视频流的分辨率信息，可以在启播前优先指定播放的视频分辨率。 播放器会查找小于或等于偏好分辨率的流进行启播，启播后没有必要再通过 setBitrateIndex 切换到需要的码流。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
// 传入参数为视频宽和高的乘积(宽 * 高），可以自定义值传入
[_config setPreferredResolution:720*1280];
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

设置播放进度回调时间间隔

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setProgressInterval:200]; // 设置进度回调间隔，单位毫秒
[_txVodPlayer setConfig:_config]; // 把config 传给 _txVodPlayer
```

播放器事件监听

您可以为 TXVodPlayer 对象绑定一个 TXVodPlayListener 监听器，即可通过 onPlayEvent（事件通知）和 onNetStatus（状态反馈）向您的应用程序同步信息。

事件通知（onPlayEvent）

播放事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始
PLAY_EVT_PLAY_PROGRESS	2005	视频播放进度，会通知当前播放进度、加载进度和总体时长
PLAY_EVT_PLAY_LOADING	2007	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件
PLAY_EVT_VOD_LOADING_END	2014	视频播放 loading 结束，视频继续播放

结束事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_ERR_NET_DISCONNECT	-2301	网络断连，且经多次重连亦不能恢复，更多重试请自行重启播放
PLAY_ERR_HLS_KEY	-2305	HLS 解密 key 获取失败

警告事件

如下的这些事件您可以不用关心，它只是用来告知您 SDK 内部的一些事件。

事件 ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连，已启动自动重连（重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT 了）
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解

连接事件

此外还有几个连接服务器的事件，主要用于测定和统计服务器连接时间：

事件 ID	数值	含义说明
PLAY_EVT_VOD_PLAY_PREPARED	2013	播放器已准备完成，可以播放。设置了 autoPlay 为 false 之后，需要在收到此事件后，调用 resume 才会开始播放
PLAY_EVT_RCV_FIRST_I_FRAME	2003	网络接收到首个可渲染的视频数据包（IDR）

画面事件

以下事件用于获取画面变化信息：

事件 ID	数值	含义说明
PLAY_EVT_CHANGE_RESOLUTION	2009	视频分辨率改变
PLAY_EVT_CHANGE_ROATION	2011	MP4 视频旋转角度

视频信息事件

事件 ID	数值	含义说明
PLAY_EVT_GET_PLAYINFO_SUCC	2010	成功获取播放文件信息

如果通过 fileId 方式播放且请求成功，SDK 会将一些请求信息通知到上层。您可以在收到 PLAY_EVT_GET_PLAYINFO_SUCC 事件后，解析 param 获取视频信息。

视频信息	含义说明
EVT_PLAY_COVER_URL	视频封面地址
EVT_PLAY_URL	视频播放地址
EVT_PLAY_DURATION	视频时长

```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary*)param
{
if (EvtID == PLAY_EVT_VOD_PLAY_PREPARED) {
//收到播放器已经准备完成事件，此时可以调用pause、resume、getWidth、getSupportedBitrates 等接口
} else if (EvtID == PLAY_EVT_PLAY_BEGIN) {
// 收到开始播放事件
} else if (EvtID == PLAY_EVT_PLAY_END) {
// 收到开始结束事件
}
}
```

状态反馈（onNetStatus）

状态反馈每0.5秒都会被触发一次，目的是实时反馈当前的推流器状态，它就像汽车的仪表盘，可以告知您目前 SDK 内部的一些具体情况，以便您能对当前视频播放状态等有所了解。

评估参数	含义说明
CPU_USAGE	当前瞬时 CPU 使用率
VIDEO_WIDTH	视频分辨率 – 宽
VIDEO_HEIGHT	视频分辨率 – 高
NET_SPEED	当前的网络数据接收速度
VIDEO_FPS	当前流媒体的视频帧率
VIDEO_BITRATE	当前流媒体的视频码率，单位 kbps
AUDIO_BITRATE	当前流媒体的音频码率，单位 kbps
V_SUM_CACHE_SIZE	缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了
SERVER_IP	连接的服务器 IP

通过 onNetStatus 获取视频播放过程信息示例：

```
- (void)onNetStatus:(TXVodPlayer *)player withParam:(NSDictionary *)param {
//获取当前CPU使用率
float cpuUsage = [[param objectForKey:@"CPU_USAGE"] floatValue];
//获取视频宽度
int videoWidth = [[param objectForKey:@"VIDEO_WIDTH"] intValue];
//获取视频高度
int videoHeight = [[param objectForKey:@"VIDEO_HEIGHT"] intValue];
//获取实时速率
```

```
int speed = [[param objectForKey:@"NET_SPEED"] intValue];
//获取当前流媒体的视频帧率
int fps = [[param objectForKey:@"VIDEO_FPS"] intValue];
//获取当前流媒体的视频码率，单位 kbps
int videoBitRate = [[param objectForKey:@"VIDEO_BITRATE"] intValue];
//获取当前流媒体的音频码率，单位 kbps
int audioBitRate = [[param objectForKey:@"AUDIO_BITRATE"] intValue];
//获取缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了
int jitterbuffer = [[param objectForKey:@"V_SUM_CACHE_SIZE"] intValue];
//获取连接的服务器的IP地址
NSString *ip = [param objectForKey:@"SERVER_IP"];
}
```

场景化功能

1、基于 SDK 的 Demo 组件

基于播放器SDK，腾讯云研发了一款 播放器组件，集质量监控、视频加密、极速高清、清晰度切换、小窗播放等功能于一体，适用于所有点播、直播播放场景。封装了完整功能并提供上层 UI，可帮助您在短时间内，打造一个媲美市面上各种流行视频 App 的播放软件。

2、开源 Github

基于播放器 SDK，腾讯云研发了沉浸式视频播放器组件、视频 Feed 流、多播放器复用组件等，而且随着版本发布，我们会提供跟多的基于用户场景的组件。您可以通过 Player_iOS 下载体验。

API 文档

最近更新时间：2022-05-18 09:46:36

TXVodPlayer

点播播放器

请参见 TXVodPlayer。

主要负责从指定的点播流地址拉取音视频数据，并进行解码和本地渲染播放。

播放器包含如下能力：

- 支持 FLV、MP4 及 HLS 多种播放格式，支持 基础播放（URL 播放） 和 点播播放（Fileid 播放）两种播放方式 。
- 屏幕截图，可以截取当前播放流的视频画面。
- 通过手势操作，调节亮度、声音、进度等。
- 可以手动切换不同的清晰度，也可根据网络带宽自适应选择清晰度。
- 可以指定不同倍速播放，并开启镜像和硬件加速。
- 完整能力，请参见 点播超级播放器 - 能力清单。

播放器配置接口

API	描述
config	点播配置，配置信息请参见 TXVodPlayConfig。
isAutoPlay	startPlay 后是否立即播放，默认 YES。
token	加密 HLS 的 token。设置此值后，播放器自动在 URL 中的文件名之前增加 voddrm.token.TOKEN TextureView。
loop	是否循环播放 SurfaceView。
enableHWAcceleration	视频渲染回调。（仅硬解支持）
setExtentOptionInfo	设置播放器业务参数，参数格式为<nsstring *,="" id="">

播放基础接口

API	描述
startPlay	播放 HTTP URL 形式地址。
startPlayWithParams	以 fileId 形式播放。
stopPlay	停止播放。
isPlaying	是否正在播放。
pause	暂停播放，停止获取流数据,保留最后一帧画面。

API	描述
resume	恢复播放，重新获取流数据。
seek	跳转到视频流指定时间点，单位秒。
currentPlaybackTime	获取当前播放位置，单位秒。
duration	获取总时长，单位秒。
playableDuration	获取可播放时长，单位秒。
width	获取视频宽度。
height	获取视频高度。
setStartTime	设置播放开始时间。

视频相关接口

API	描述
snapshot	获取当前视频帧图像。 **注意：由于获取当前帧图像是比较耗时的操作，所以截图会通过异步回调出来。**
setMirror	设置镜像。
setRate	设置点播的播放速率，默认1.0。
bitrateIndex	返回当前播放的码率索引。
setBitrateIndex	设置当前正在播放的码率索引，无缝切换清晰度。 清晰度切换可能需要等待一小段时间。
setRenderMode	设置 图像平铺模式。
setRenderRotation	设置 图像渲染角度。

音频相关接口

API	描述
setMute	设置是否静音播放。
setAudioPlayoutVolume	设置音量大小，范围：0 - 100。

事件通知接口

API	描述

API	描述
delegate	事件回调，建议使用 vodDelegate。
vodDelegate	设置播放器的回调。
videoProcessDelegate	视频渲染回调（仅硬解支持）。

TRTC 相关接口

通过以下接口，可以把点播播放器的音视频流通过 TRTC 进行推送，更多 TRTC 服务请参见 TRTC 产品概述。

API	描述
attachTRTC	点播绑定到 TRTC 服务。
detachTRTC	点播解绑 TRTC 服务。
publishVideo	开始推送视频流。
unpublishVideo	取消推送视频流。
publishAudio	开始推送音频流。
unpublishAudio	取消推送音频流。

TXVodPlayListener

腾讯云点播回调通知。

SDK 基础回调

API	描述
onPlayEvent	点播播放事件通知，请参见 播放事件列表、事件参数。
onNetStatus	点播播放器 网络状态通知。

TXVodPlayConfig

点播播放器配置类。

基础配置接口

API	描述
connectRetryCount	设置播放器重连次数。
connectRetryInterval	设置播放器重连间隔，单位秒。

API	描述
timeout	设置播放器连接超时时间，单位秒。
cacheFolderPath	设置点播缓存目录，点播 MP4、HLS 有效。
maxCacheItems	设置缓存文件个数。
playerType	设置播放器类型。
headers	设置自定义 HTTP headers。
enableAccurateSeek	设置是否精确 seek，默认 true。
autoRotate	播放 MP4 文件时，若设为 YES 则根据文件中的旋转角度自动旋转。 旋转角度可在 PLAY_EVT_CHANGE_ROTATION 事件中获得。默认 YES。
smoothSwitchBitrate	平滑切换多码率 HLS，默认 false。
progressInterval	设置进度回调间隔，单位毫秒。
maxBufferSize	最大预加载大小，单位 MB。
maxPreloadSize	设置预加载最大缓冲大小，单位：MB。
firstStartPlayBufferTime	设置首缓需要加载的数据时长，单位ms，默认值为100ms。
nextStartPlayBufferTime	缓冲时（缓冲数据不够引起的二次缓冲，或者seek引起的拖动缓冲）最少要缓存多长的数据才能结束缓冲，单位ms，默认值为250ms。
overlayKey	设置 HLS 安全加固加解密 key。
overlayIv	设置 HLS 安全加固加解密 Iv。
extInfoMap	设置拓展信息。
preferredResolution	播放 HLS 有多条码流时，根据设定的 preferredResolution 选最优的码流进行起播 *，preferredResolution 是宽高的乘积（width * height）， 启播前设置才有效。
enableRenderProcess	是否允许加载后渲染后处理服务(如超分插件服务）,默认开启。

TXPlayerGlobalSetting

点播播放器全局配置类

API	描述
setCacheFolderPath	设置播放引擎的cache目录。设置后，预下载，播放器等会优先从此目录读取和存储

API	描述
setMaxCacheSize	设置播放引擎的最大缓存大小。设置后会根据设定值自动清理Cache目录的文件。单位MB。

TXVodPreloadManager

点播播放器预下载接口类

API	描述
sharedManager	获取 TXVodPreloadManager 实例对象，单例模式。
startPreload	启动预下载前，请先设置好播放引擎的缓存目录 TXPlayerGlobalSetting#setCacheFolderPath 和缓存大小 TXPlayerGlobalSetting#setMaxCacheSize。
stopPreload	停止预下载

TXVodDownloadManager

点播播放器视频下载接口类

API	描述
shareInstance	获取 TXVodDownloadManager 实例对象，单例模式。
setDownloadPath	设置下载根目录。
setHeaders	设置下载 HTTP 头。
setListener	设置下载回调方法，下载前必须设好。
startDownloadUrl	以 URL 方式开始下载。
startDownload	以 FileID 方式开始下载。
stopDownload	停止下载，ITXVodDownloadListener.onDownloadStop 回调时停止成功。
deleteDownloadFile	删除下载文件。
deleteDownloadMediaInfo	删除下载信息。
getDownloadMediaInfoList	获取所有用户的下载列表信息。

ITXVodDownloadListener

腾讯云视频下载回调通知。

API	描述
onDownloadStart	下载开始。
onDownloadProgress	下载进度更新。
onDownloadStop	下载停止。
onDownloadFinish	下载结束。
onDownloadError	下载过程中遇到错误。
hlsKeyVerify	下载 HLS，遇到加密的文件，将解密 Key 给外部校验。

错误码表

常规事件

code	事件定义	含义说明
2004	PLAY_EVT_PLAY_BEGIN	视频播放开始（若有转圈圈效果，此时将停止）。
2005	PLAY_EVT_PLAY_PROGRESS	视频播放进度，会通知当前播放进度、加载进度和总体时长。
2007	PLAY_EVT_PLAY_LOADING	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件。
2014	PLAY_EVT_VOD_LOADING_END	视频播放 loading 结束，视频继续播放。
2006	PLAY_EVT_PLAY_END	视频播放结束。
2013	PLAY_EVT_VOD_PLAY_PREPARED	播放器已准备完成，可以播放。
2003	PLAY_EVT_RCV_FIRST_I_FRAME	网络接收到首个可渲染的视频数据包（IDR）。
2009	PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2011	PLAY_EVT_CHANGE_ROTATION	MP4 视频旋转角度。

警告事件

code	事件定义	含义说明
-2301	PLAY_ERR_NET_DISCONNECT	网络断连，且经多次重连亦不能恢复,更多重试请自行重启播放。
-2305	PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
2101	PLAY_WARNING_VIDEO_DECODE_FAIL	当前视频帧解码失败。

code	事件定义	含义说明
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（重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT）。
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-2304	PLAY_ERR_HEVC_DECODE_FAIL	H265 解码失败。
-2303	PLAY_ERR_FILE_NOT_FOUND	播放的文件不存在。

Android 端集成
超级播放器
接入指引

最近更新时间：2022-05-31 15:38:49

产品概述

腾讯云视立方 Android 超级播放器是腾讯云开源的一款播放器组件，集质量监控、视频加密、极速高清、清晰度切换、小窗播放等功能于一体，适用于所有点播、直播播放场景。封装了完整功能并提供上层 UI，可帮助您在短时间内，打造一个媲美市面上各种流行视频 App 的播放软件。

若超级播放器组件满足不了您的业务的个性化需求，且您具有一定的开发经验，可以集成 视立方播放器 SDK，自定义开发播放器界面和播放功能。

准备工作

1. 为了您体验到更完整全面的播放器功能，建议您开通 云点播 相关服务，未注册用户可注册账号 试用。若您不使用云点播服务，可略过此步骤，但集成后仅可使用播放器基础能力。
2. 下载 Android Studio，您可以进入 Android Studio 官网 下载安装，如已下载可略过该步骤。

通过本文您可以学会

1. 如何集成腾讯云视立方 Android 超级播放器
2. 如何创建和使用播放器

集成准备

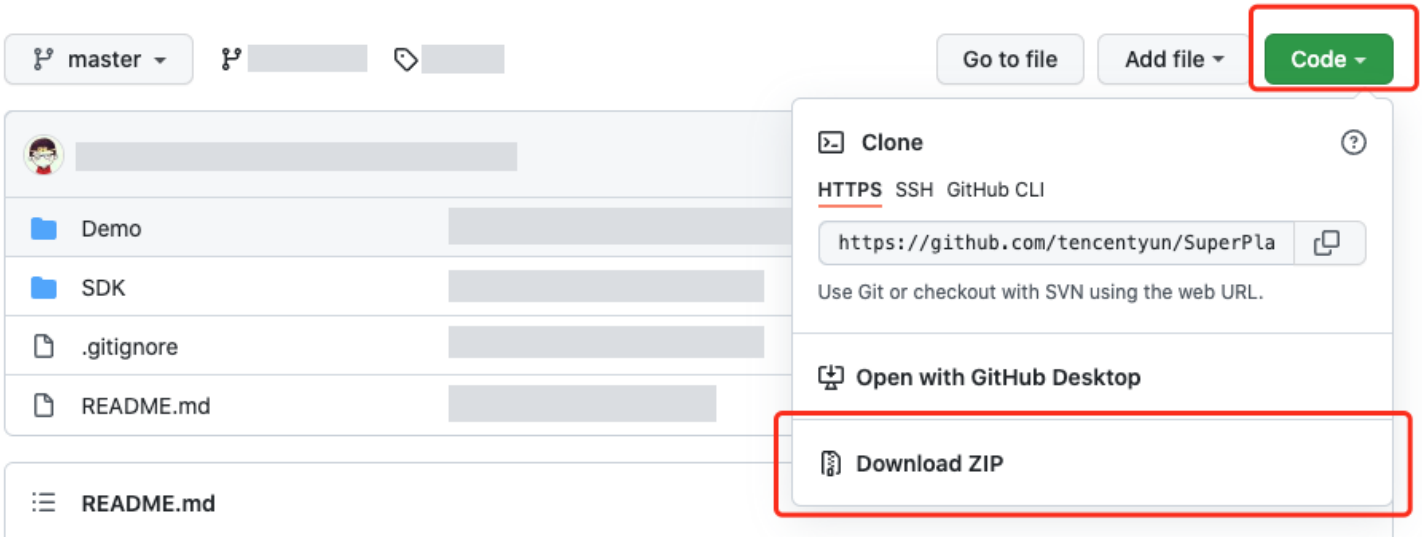
步骤1：项目下载

腾讯云视立方 Android 超级播放器的项目地址是 SuperPlayer_Android。

您可通过 **下载播放器组件 ZIP 包** 或 **Git 命令下载** 的方式下载腾讯云视立方 Android 超级播放器项目工程。

下载播放器组件 ZIP 包

您可以直接下面播放器组件 ZIP 包，单击页面的 **Code** > **Download ZIP** 下载。



Git 命令下载

1. 首先确认您的电脑上安装了 Git，如果没有安装，可以参见 Git 安装教程 进行安装。
2. 执行下面的命令把超级播放器组件工程代码 clone 到本地。

```
git clone git@github.com:tencentyun/SuperPlayer_Android.git
```

提示下面的信息表示成功 clone 工程代码到本地。

```
正克隆到 'SuperPlayer_Android'...
remote: Enumerating objects: 2637, done.
remote: Counting objects: 100% (644/644), done.
remote: Compressing objects: 100% (333/333), done.
remote: Total 2637 (delta 227), reused 524 (delta 170), pack-reused 1993
接收对象中: 100% (2637/2637), 571.20 MiB | 3.94 MiB/s, 完成.
处理 delta 中: 100% (1019/1019), 完成.
```

下载工程后，源码解压后的目录如下：

文件名	作用
LiteAVDemo(Player)	超级播放器 Demo 工程，导入到 Android Studio 后可以直接运行
app	主界面入口
superplayerkit	超级播放器组件（SuperPlayerView），具备播放、暂停、手势控制等常见功能
superplayerdemo	超级播放器 Demo 代码
common	工具类模块

文件名	作用
SDK	视立方播放器 SDK，包括：LiteAVSDK_Player_x.x.x.aar，aar 格式提供的 SDK；LiteAVSDK_Player_x.x.x.zip，lib 和 jar 格式提供的 SDK
Player说明文档(Android).pdf	超级播放器使用文档

步骤2：集成指引

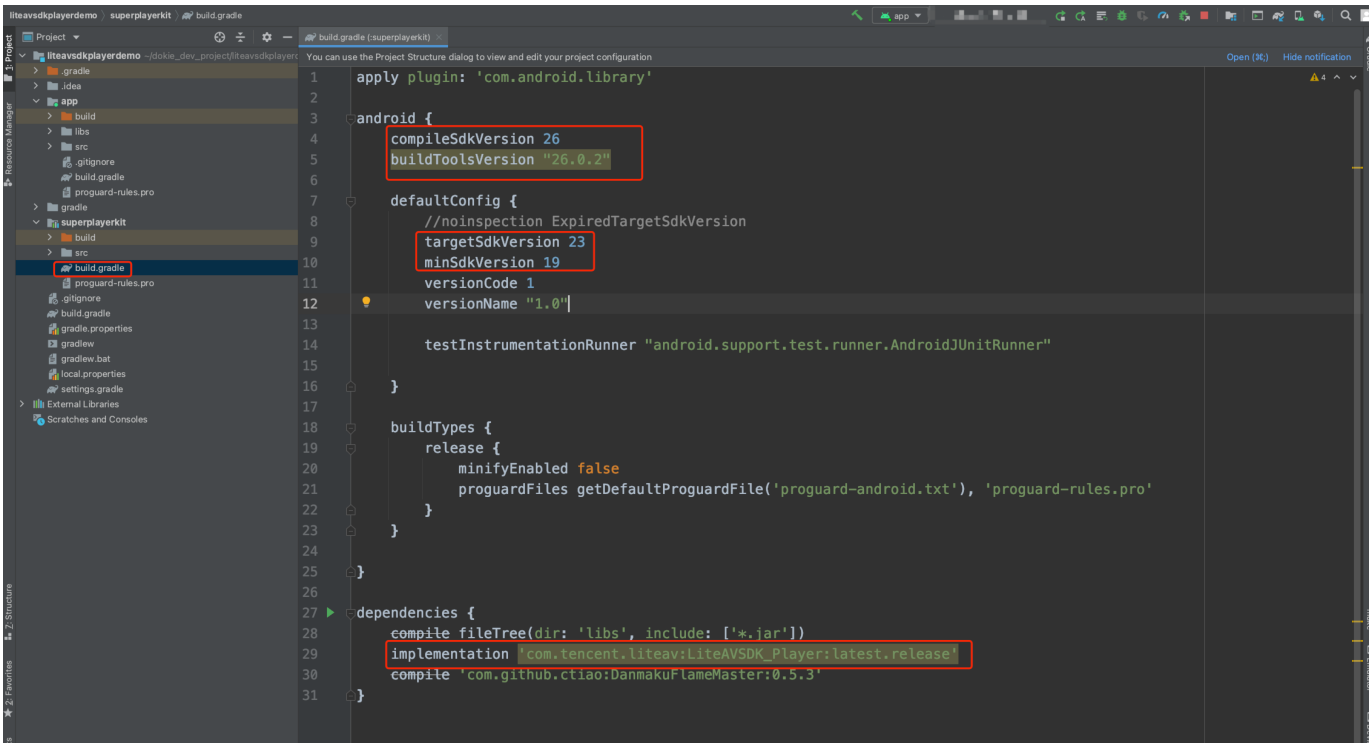
本步骤可指导您如何集成播放器，您可选择使用 Gradle 自动加载的方式，手动下载 aar 再将其导入到您当前的工程或导入 jar 和 so 库的方式集成项目。

Gradle 自动加载（AAR）

1. 下载 SDK + Demo 开发包，项目地址为 Android。
2. 把Demo/superplayerkit这个 module 复制到工程中，然后进行下面的配置：
   - 在工程目录下的setting.gradle 导入superplayerkit。

```
include ':superplayerkit'
```

   - 打开superplayerkit 工程的 build.gradle 文件修改 compileSdkVersion，buildToolsVersion，minSdkVersion，targetSdkVersion 和 rootProject.ext.liteavSdk 的常量值。

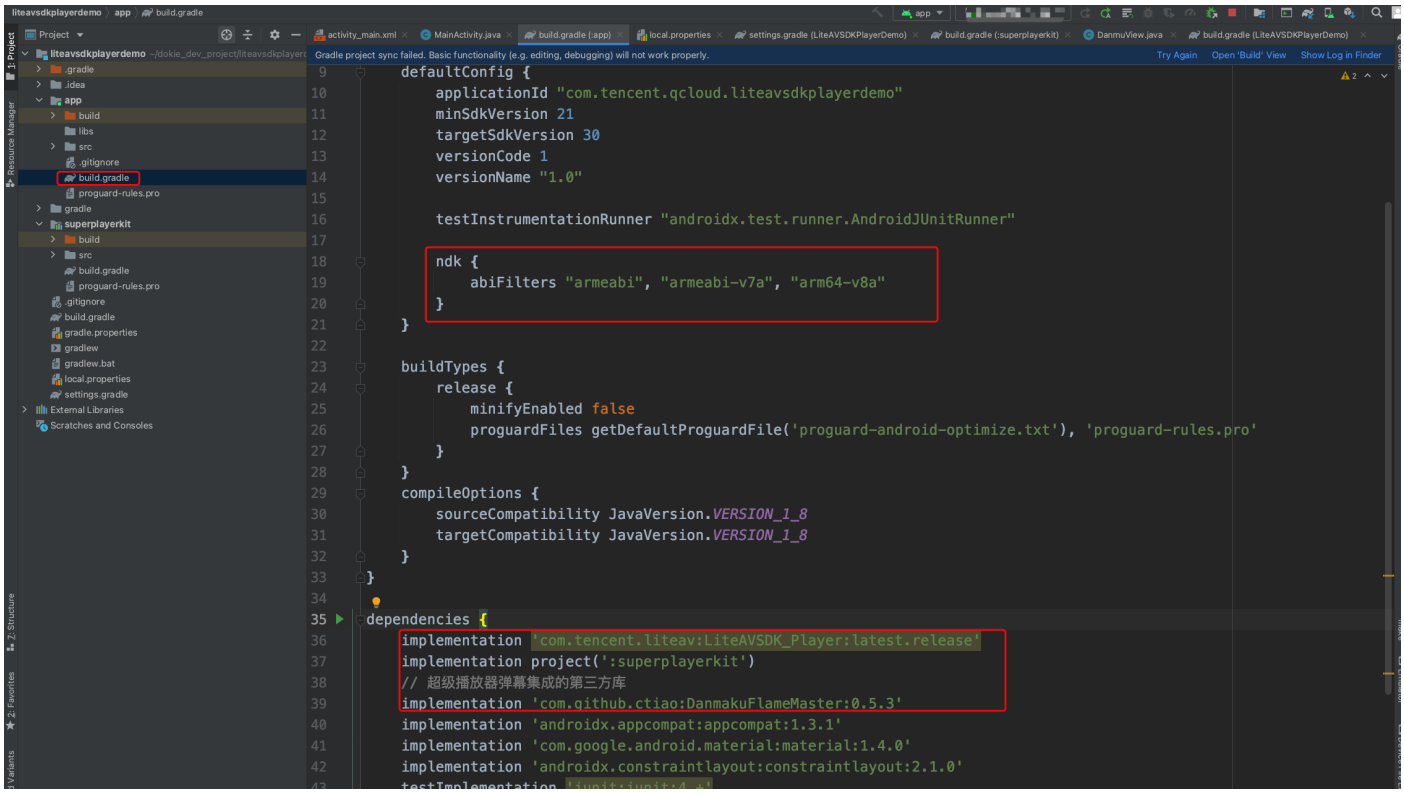



```
compileSdkVersion 26
buildToolsVersion "26.0.2"

defaultConfig {
```

```
targetSdkVersion 23
minSdkVersion 19
}

dependencies {
//如果要集成历史版本，可将 latest.release 修改为对应的版本，例如：8.5.290009
implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
}
```

请参见上面的步骤，把common模块导入到项目，并进行配置。

3. 通过在 gradle 配置 mavenCentral 库，自动下载更新 LiteAVSDK，打开app/build.gradle，进行下面的配置：



i. 在 dependencies 中添加 LiteAVSDK_Player 的依赖。

```
dependencies {
implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
implementation project(':superplayerkit')
// 超级播放器弹幕集成的第三方库
implementation 'com.github.ctiao:DanmakuFlameMaster:0.5.3'
}
```

如果您需要集成历史版本的 LiteAVSDK_Player SDK ，可以在 MavenCentral 查看历史版本，然后通过下面的方式进行集成：

```
dependencies {
// 集成8.5.10033 版本LiteAVSDK_Player SDK
implementation 'com.tencent.liteav:LiteAVSDK_Player:8.5.10033'
}
```

ii. 在 app/build.gradle defaultConfig 中，指定 App 使用的 CPU 架构（目前 LiteAVSDK 支持 armeabi 、armeabi-v7a 和 arm64-v8a，可根据项目需求配置）。

```
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

如果之前没有使用过9.4以及更早版本的 SDK 的 下载缓存功能（TXVodDownloadManager 中的相关接口），并且不需要在9.5及后续 SDK 版本播放9.4及之前缓存的下载文件，可以不需要该功能的 so 文件，达到减少安装包的体积，例如：在9.4及之前版本使用了 TXVodDownloadManager 类的 setDownloadPath 和 startDownloadUrl 函数下载了相应的缓存文件，并且应用内存储了 TXVodDownloadManager 回调的 getPlayPath 路径用于后续播放，这时候需要 libijkhlscache-master.so 播放该 getPlayPath 路径文件，否则不需要。可以在 app/build.gradle 中添加：

```
packagingOptions {
exclude "lib/armeabi/libijkhlscache-master.so"
exclude "lib/armeabi-v7a/libijkhlscache-master.so"
exclude "lib/arm64-v8a/libijkhlscache-master.so"
}
```

iii. 在工程目录的 build.gradle 添加 mavenCentral 库。

```
repositories {
mavenCentral()
}
```

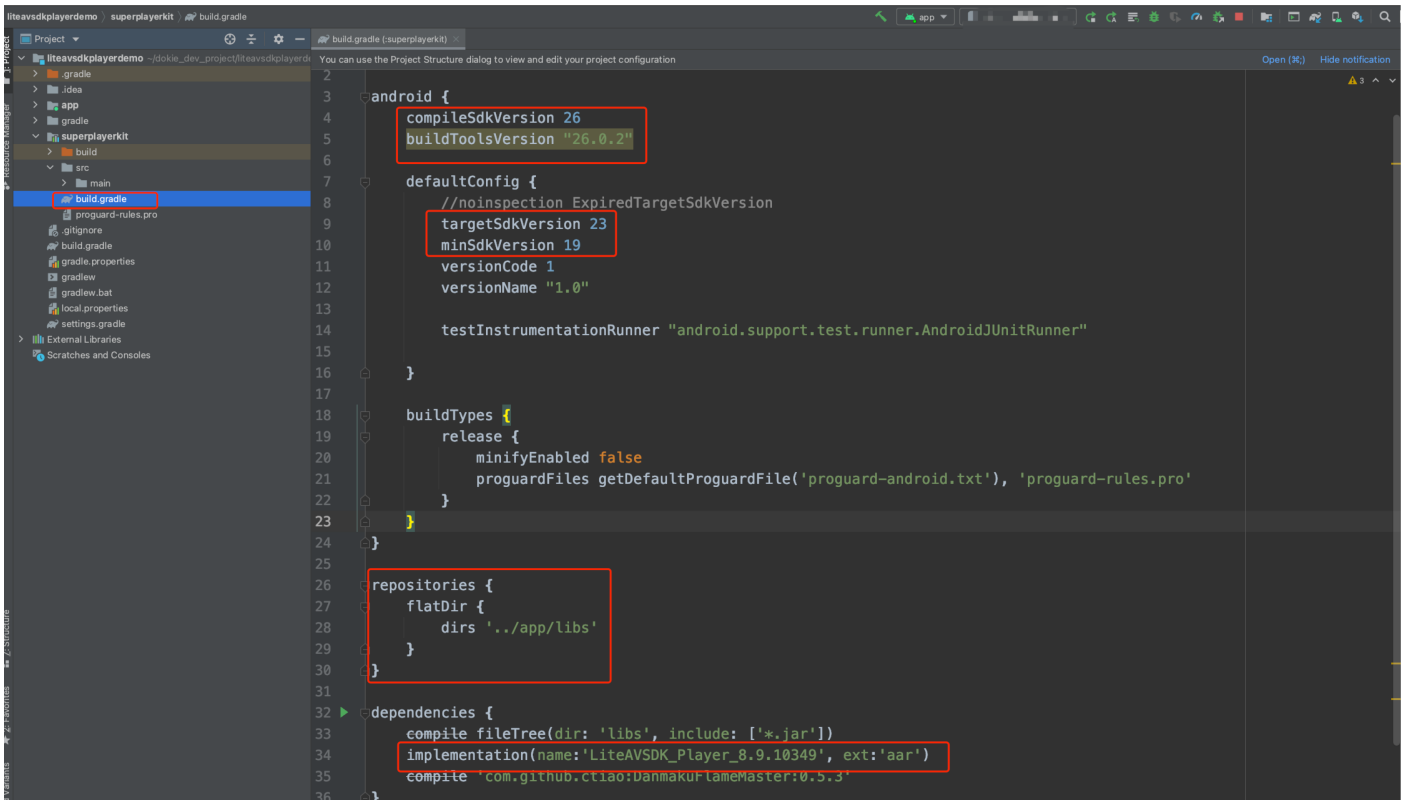
4. 单击  Sync Now 按钮同步 SDK，如果您的网络连接 mavenCentral 没有问题，很快 SDK 就会自动下载集成到工程里。

Gradle 手动下载（AAR）

1. 下载 SDK + Demo 开发包，项目地址为 Android。
2. 导入 SDK/LiteAVSDK_Player_XXX.aar（其中 XXX 为版本号）到 app 下面的 libs 文件夹以及复制 Demo/superplayerkit 这个 module 到工程中。
3. 在工程目录下的 setting.gradle 导入superplayerkit。

```
include ':superplayerkit'
```

4. 打开superplayerkit工程的 build.gradle 文件修改 compileSdkVersion，buildToolsVersion，minSdkVersion，targetSdkVersion 和 rootProject.ext.liteavSdk 的常量值。




```
compileSdkVersion 26
buildToolsVersion "26.0.2"

defaultConfig {
targetSdkVersion 23
minSdkVersion 19
}

dependencies {
implementation(name:'LiteAVSDK_Player_8.9.10349', ext:'aar')
}
```

请参见上面的步骤，把common模块导入到项目，并进行配置。

- 配置 repositories

```
repositories {
flatDir {
dirs '../app/libs'
```

```
}
}
```

5. 在app/build.gradle中添加依赖：

```
compile(name:'LiteAVSDK_Player_8.9.10349', ext:'aar')
implementation project(':superplayerkit')
// 超级播放器弹幕集成的第三方库
implementation 'com.github.ctiao:DanmakuFlameMaster:0.5.3'
```

6. 在项目build.gradle中添加：

```
allprojects {
repositories {
flatDir {
dirs 'libs'
}
}
}
```

7. 在 app/build.gradle defaultConfig 中，指定 App 使用的 CPU 架构（目前 LiteAVSDK 支持 armeabi 、armeabi-v7a 和 arm64-v8a）。

```
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

如果之前没有使用过9.4以及更早版本的 SDK 的 下载缓存功能（TXVodDownloadManager 中的相关接口），并且不需要在9.5及后续 SDK 版本播放9.4及之前缓存的下载文件，可以不需要该功能的 so 文件，达到减少安装包的体积，例如：在9.4及之前版本使用了 TXVodDownloadManager 类的 setDownloadPath 和 startDownloadUrl 函数下载了相应的缓存文件，并且应用内存储了 TXVodDownloadManager 回调的 getPlayPath 路径用于后续播放，这时候需要 libijkhlscache-master.so 播放该 getPlayPath 路径文件，否则不需要。可以在 app/build.gradle 中添加：

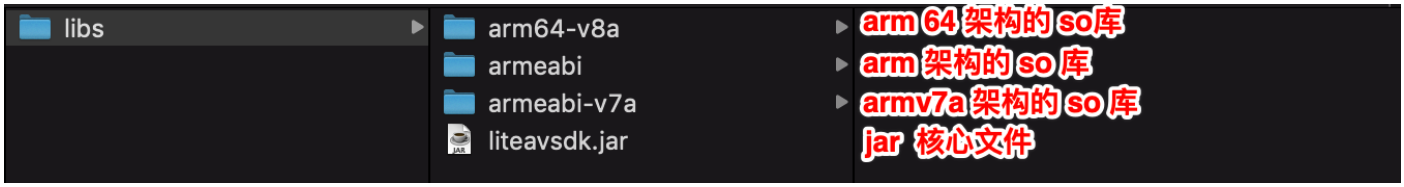
```
packagingOptions {
exclude "lib/armeabi/libijkhlscache-master.so"
exclude "lib/armeabi-v7a/libijkhlscache-master.so"
exclude "lib/arm64-v8a/libijkhlscache-master.so"
}
```

8. 单击 Sync Now 按钮同步 SDK，完成超级播放器的集成工作。

集成 SDK（jar+so）

如果您不想集成 aar 库，也可以通过导入 jar 和 so 库的方式集成 LiteAVSDK：

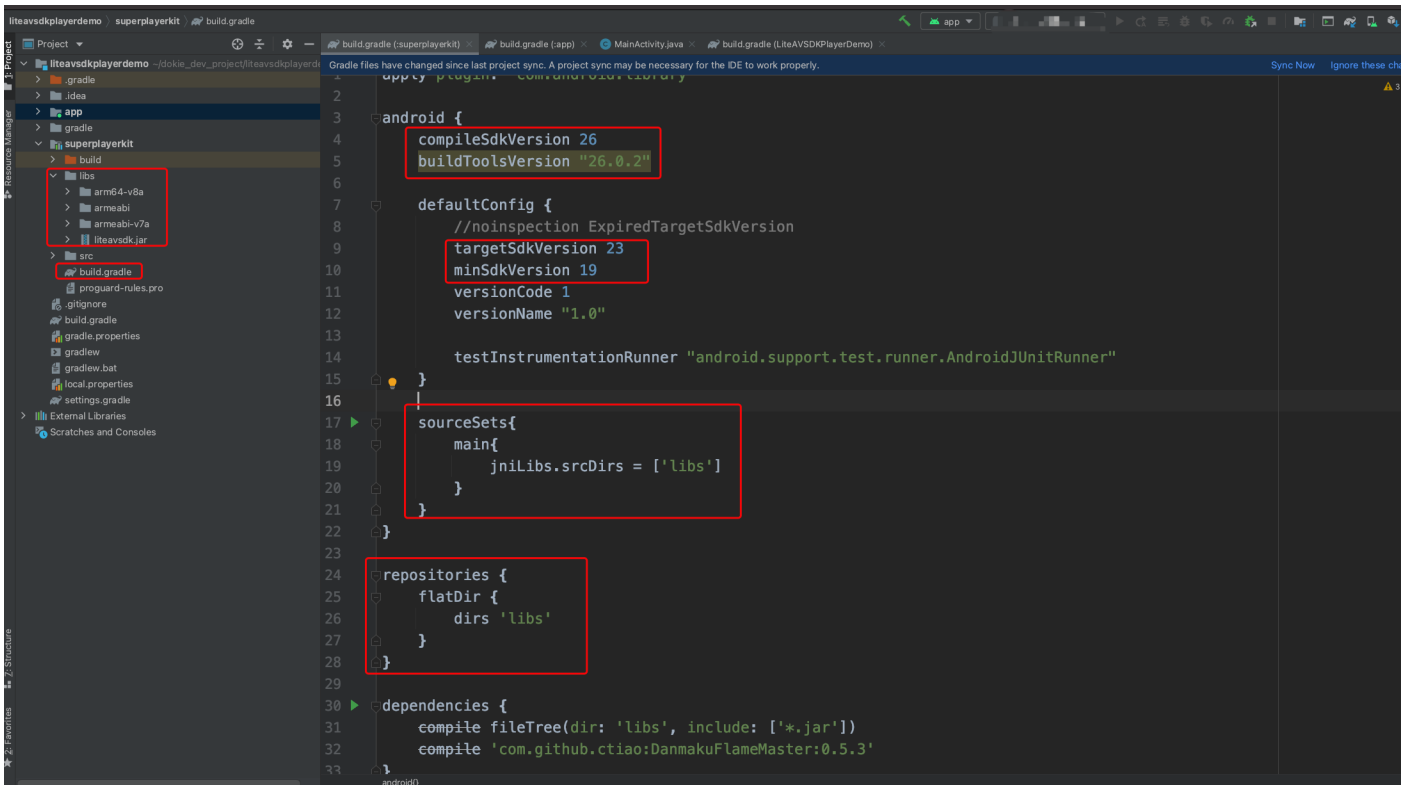
1. 下载 SDK + Demo 开发包，项目地址为 Android，下载完成后进行解压。在 SDK 目录找到 SDK/LiteAVSDK_Player_XXX.zip（其中 XXX 为版本号），解压得到 libs 目录，里面包含 jar 文件和 so 文件夹，文件清单如下：



2. 把Demo/superplayerkit这个 module 复制到工程中，然后在工程目录下的 setting.gradle 导入superplayerkit。

```
include ':superplayerkit'
```

3. 把 步骤1 解压得到的 libs 文件夹复制superplayerkit工程根目录。
4. 修改superplayerkit/build.gradle文件：



```
android {
    compileSdkVersion 26
    buildToolsVersion "26.0.2"

    defaultConfig {
        //noinspection ExpiredTargetSdkVersion
        targetSdkVersion 23
        minSdkVersion 19
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }

    sourceSets{
        main{
            jniLibs.srcDirs = ['libs']
        }
    }
}

repositories {
    flatDir {
        dirs 'libs'
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.github.ctiao:DanmakuFlameMaster:0.5.3'
}
```

compileSdkVersion 26
buildToolsVersion "26.0.2"

defaultConfig {
targetSdkVersion 23

```
minSdkVersion 19
}
```

请参见上面的步骤，把common模块导入到项目，并进行配置。

- 配置 sourceSets，添加 so 库引用代码。

```
sourceSets{
main{
jniLibs.srcDirs = ['libs']
}
}
```

- 配置 repositories，添加 flatDir，指定本地仓库路径。

```
repositories {
flatDir {
dirs 'libs'
}
}
```

5. 在app/build.gradle defaultConfig 中，指定 App 使用的 CPU 架构（目前 LiteAVSDK 支持 armeabi 、armeabi-v7a 和 arm64-v8a）。

```
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
```

6. 单击 Sync Now 按钮同步 SDK，完成 超级播放器的集成工作。

您已经完成了腾讯云视立方 Android 超级播放器项目集成的步骤。

步骤3：配置 App 权限

在 AndroidManifest.xml 中配置 App 的权限，LiteAVSDK 需要以下权限：

```
<!--网络权限-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--点播播放器悬浮窗权限-->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

```
<!--存储-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

步骤4：设置混淆规则

在 proguard-rules.pro 文件，将 TRTC SDK 相关类加入不混淆名单：

```
-keep class com.tencent.** { *; }
```

您已经完成了腾讯云视立方 Android 超级播放器 app 权限配置的步骤。

步骤5：使用播放器功能

本步骤，用于指导用户创建和使用播放器，并使用播放器进行视频播放。

1. 创建播放器

播放器主类为 SuperPlayerView ，创建后即可播放视频， 支持集成 FileID 或者 URL 进行播放。在布局文件创建 SuperPlayerView：

```
<!-- 超级播放器-->
<com.tencent.liteav.demo.superplayer.SuperPlayerView
android:id="@+id/superVodPlayerView"
android:layout_width="match_parent"
android:layout_height="200dp" />
```

2. 配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 Application 类中进行如下设置：

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // 获取到的 licence url
String licenceKey = ""; // 获取到的 licence key
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
TXLiveBase.setListener(new TXLiveBaseListener() {
@Override
public void onLicenceLoaded(int result, String reason) {
Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
}
});
}
}
```

3. 播放视频

本步骤用于指导用户播放视频。腾讯云视立方 Android 超级播放器可用于直播和点播两种播放场景，具体如下：

- 点播播放：超级播放器支持两种点播播放方式，可以 通过 FileID 播放 腾讯云点播媒体资源，也可以直接使用 URL 播放 地址进行播放。
- 直播播放：超级播放器可使用 URL 播放 的方式实现直播播放。通过传入 URL 地址，即可拉取直播音视频流进行直播播放。腾讯云直播URL生成方式可参见 自主拼装直播 URL。**通过 URL 播放（直播、点播）**

  URL可以是点播文件播放地址，也可以是直播拉流地址，传入相应 URL 即可播放相应视频文件。

```
SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1400329073; // 配置 AppId
model.url = "http://your_video_url.mp4"; // 配置您的播放视频url
mSuperPlayerView.playWithModel(model);
```

通过 FileID 播放（点播）

视频 FileId 在一般是在视频上传后，由服务器返回：

a. 客户端视频发布后，服务器会返回 FileId 到客户端。

b. 服务端视频上传时，在 确认上传 的通知中包含对应的 FileId。

如果文件已存在腾讯云，则可以进入 媒资管理 ，找到对应的文件，查看 FileId。如下图所示，ID 即表示 FileId：

注意：

- **通过 FileID 播放时，需要首先使用 Adaptive-HLS(10) 转码模板对视频进行转码，或者使用超级播放器签名 psign 指定播放的视频，否则可能导致视频播放失败。转码教程和说明可参见 用超级播放器播放视频，psign 生成教程可参见 psign 教程。**
- **若您在通过 FileID 播放时出现“no v4 play info”异常，则说明您可能存在上述问题，建议您根据上述教程调整。同时您也可以直接获取源视频播放链接，通过 URL 播放 的方式实现播放。**
- **未经转码的源视频在播放时有可能出现不兼容的情况，建议您使用转码后的视频进行播放。**

```
//在未开启防盗链进行播放的过程中，如果出现了“no v4 play info”异常，建议您使用Adaptive-HLS(10)转码模板对视频进行转码，或直接获取源视频播放链接通过url方式进行播放。

SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1400329071;// 配置 AppId
model.videoId = [[SuperPlayerVideoId alloc] init];
model.videoId.fileId = @"5285890799710173650"; // 配置 FileId
//私有加密播放需填写 psign， psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
//model.videoId.pSign = @"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTQwMDMyOTA3MSwiZmlsZUlkIjoiNTI4NTg5MDc5OTcxMDE3MzY1MCIsImN1cnJlbnRUaW1lU3RhbXAiOjEsImV4cGlyZVRpbWVTdGFtcCI6MjE0NzQ4MzY0NywidXJsQWNjZXNzSW5mbyI6eyJ0IjoiN2ZmZmZmZmYifSwiZHJtTGljZW5zZUluZm8iOnsiZXhwaXJlVGltZVN0YW1wIjoyMTQ3NDgzNjQ3fX0.yJxpnQ2Evp5KZQFfuBBK05BoPpQAzYAWo6liXws-LzU";
[_playerView playWithModel:model];
```

4. **退出播放**

当不需要播放器时，调用 resetPlayer 清理播放器内部状态，释放内存。

```
mSuperPlayerView.resetPlayer();
```

至此，您已经完成了腾讯云视立方 Android 超级播放器创建、播放视频和退出播放的能力即成。

功能使用

本章将为您介绍几种常见的播放器功能使用方式，更为完整的功能使用方式可参见 Demo 体验，超级播放器支持的功能可参见 能力清单。

1、全屏播放

超级播放器支持全屏播放，在全屏播放场景内，同时支持锁屏、手势控制音量和亮度、弹幕、截屏、清晰度切换等功能设置。功能效果可在 **腾讯云视立方 App > 播放器 > 超级播放器** 中体验，单击界面右下角即可进入全屏播放界面。

在窗口播放模式下，可通过调用下述接口进入全屏播放模式：

```
mControllerCallback.onSwitchPlayMode(SuperPlayerDef.PlayerMode.FULLSCREEN);
```

全屏播放界面功能介绍

返回窗口

单击返回，即可返回至窗口播放模式。

```
//单击后触发下面的接口
mControllerCallback.onBackPressed(SuperPlayerDef.PlayerMode.FULLSCREEN);
onSwitchPlayMode(SuperPlayerDef.PlayerMode.WINDOW);
```

锁屏

锁屏操作可以让用户进入沉浸式播放状态。

```
//单击后触发的接口
toggleLockState();
```

弹幕

打开弹幕功能后屏幕上会有用户发送的文字飘过。

```
// 步骤一：向弹幕View中添加一条弹幕
addDanmaku(String content, boolean withBorder);
// 步骤二：打开或者关闭弹幕
toggleBarrage();
```

截屏

超级播放器提供播放过程中截取当前视频帧功能，您可以把图片保存起来进行分享。单击图片4处按钮可以截屏，您可以在 mSuperPlayer.snapshot 接口进行保存截取的图片。

```
mSuperPlayer.snapshot(new TXLivePlayer.ITXSnapshotListener() {
@Override
public void onSnapshot(Bitmap bitmap) {
//在这里可以保存截图
}
});
```

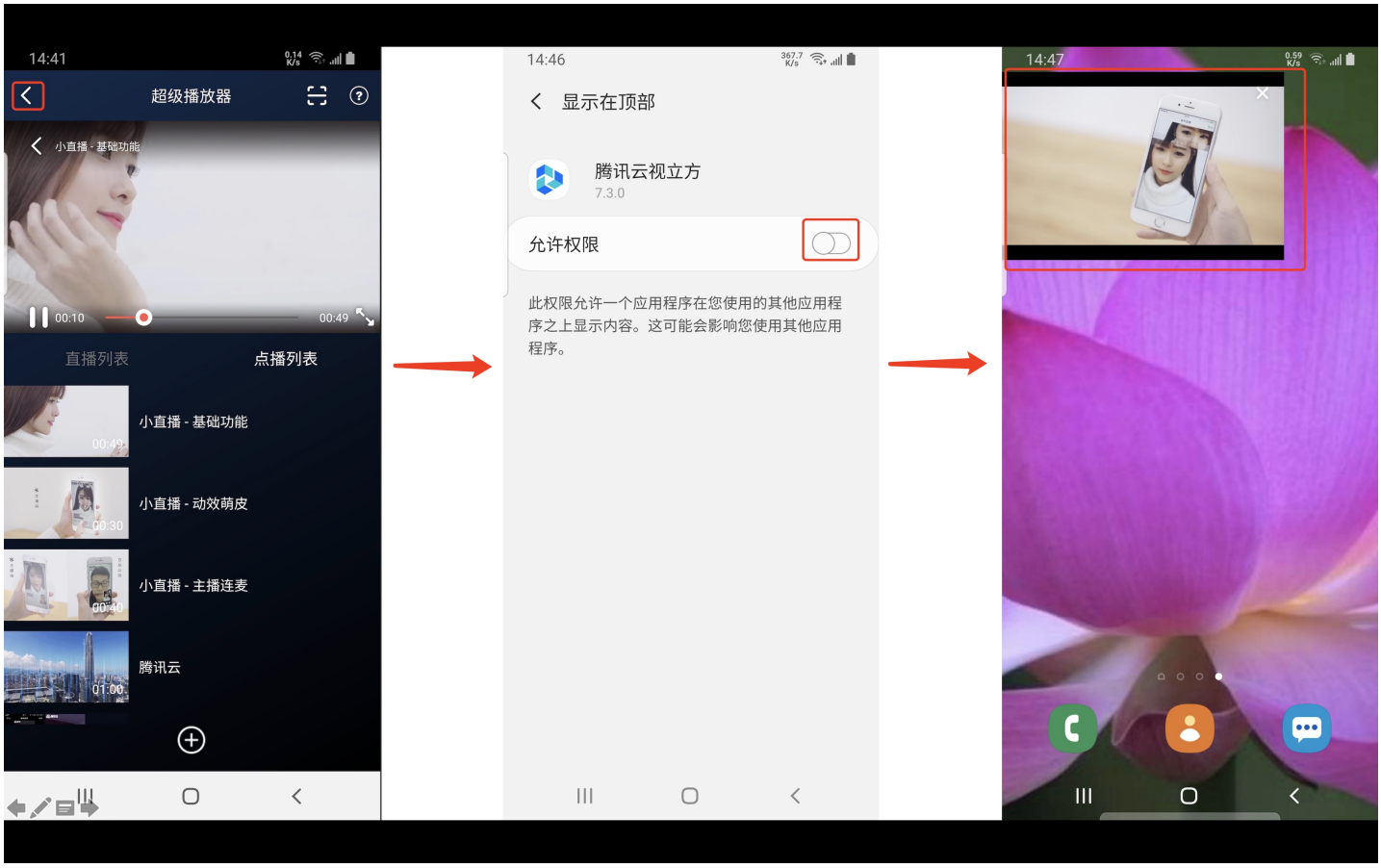
清晰度切换

用户可以根据需求选择不同的视频播放清晰度，如高清、标清或超清等。

```
//单击后触发的显示清晰度view代码接口
showQualityView();
//单击清晰度选项的回调接口为
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
// 清晰度ListView的单击事件
VideoQuality quality = mList.get(position);
mCallback.onQualitySelect(quality);
}
});
//最终改变清晰度的回调
@Override
public void onQualityChange(VideoQuality quality) {
mFullScreenPlayer.updateVideoQuality(quality);
mSuperPlayer.switchStream(quality);
}
```

2、悬浮窗播放

超级播放器支持悬浮窗小窗口播放，可在切换到其它应用时，不打断视频播放功能。功能效果可在 腾讯云视立方 App > 播放器 > 超级播放器 中体验，单击界面左上角返回，即可体验悬浮窗播放功能。



悬浮窗播放依赖于 AndroidManifest 中的以下权限：

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

```
// 切换悬浮窗触发的代码接口
mSuperPlayerView.switchPlayMode(SuperPlayerDef.PlayerMode.FLOAT);
//单击浮窗返回窗口触发的代码接口
mControllerCallback.onSwitchPlayMode(SuperPlayerDef.PlayerMode.WINDOW);
```

3、视频封面

超级播放器支持用户自定义视频封面，用于在视频接收到首帧画面播放回调前展示。功能效果可在 **腾讯云视立方 App > 播放器 > 超级播放器 > 自定义封面演示** 视频中体验。

- 当超级播放器设置为自动播放模式PLAY_ACTION_AUTO_PLAY时，视频自动播放，此时将在视频首帧加载出来之前展示封面；
- 当超级播放器设置为手动播放模式PLAY_ACTION_MANUAL_PLAY时，需用户单击**播放**后视频才开始播放。在单击**播放**前将展示封面；在单击**播放**后到视频首帧加载出来前也将展示封面。

视频封面支持使用网络 URL 地址或本地 File 地址，使用方式可参见下述指引。若您通过 FileID 的方式播放视频，则可直接在云点播内配置视频封面。

```
SuperPlayerModel model = new SuperPlayerModel();
model.appId = "您的appid";
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "您的fileId";
//播放模式，可设置自动播放模式：PLAY_ACTION_AUTO_PLAY，手动播放模式：PLAY_ACTION_MANUAL_PLAY
model.playAction = PLAY_ACTION_MANUAL_PLAY;
//设定封面的地址为网络url地址，如果coverPictureUrl不设定，那么就会自动使用云点播控制台设置的封面
```

```
model.coverPictureUrl = "http://1500005830.vod2.myqcloud.com/6c9a5118vodcq1500005830/cc1e282086022680110873365 18/MXUW1a5I9TsA.png"
mSuperPlayerView.playWithModel(model);
```

4、视频列表轮播

超级播放器支持视频列表轮播，即在给定一个视频列表后：

- 支持按顺序循环播放列表中的视频，播放过程中支持自动播放下一集也支持手动切换到下一个视频。
- 列表中最后一个视频播放完成后将自动开始播放列表中的第一个视频。

功能效果可在 **腾讯云视立方 App> 播放器 > 超级播放器 > 视频列表轮播演示**视频中体验。


```
//步骤1:构建轮播的List<SuperPlayerModel>
ArrayList<SuperPlayerModel> list = new ArrayList<>();
SuperPlayerModel model = new VideoModel();
model = new SuperPlayerModel();
model.videoId = new SuperPlayerVideoId();
model.appid = 1252463788;
model.videoId.fileId = "4564972819219071568";
list.add(model);

model = new SuperPlayerModel();
model.videoId = new SuperPlayerVideoId();
```

```
model.appid = 1252463788;
model.videoId.fileId = "4564972819219071679";
list.add(model);
//步骤2：调用轮播接口
mSuperPlayerView.playWithModelList(list, true, 0);
```

```
public void playWithModelList(List<SuperPlayerModel> models, boolean isLoopPlayList, int index);
```

接口参数说明

参数名	类型	描述
models	List	轮播数据列表
isLoopPlayList	boolean	是否循环
index	int	开始播放的 SuperPlayerModel 索引

5、视频试看

超级播放器支持视频试看功能，可以适用于非 VIP 试看等场景，开发者可以传入不同的参数来控制视频试看时长、提示信息、试看结束界面等。功能效果可在 **腾讯云视立方 App> 播放器 > 超级播放器 > 试看功能演示** 视频中体验。

方法一：

```
//步骤1：创建视频mode
SuperPlayerModel mode = new SuperPlayerModel();
//...添加视频源信息
//步骤2：创建试看信息 mode
VipWatchModel vipWatchModel = new VipWatchModel("可试看%ss，开通 VIP 观看完整视频",15);
mode.vipWatchMode = vipWatchModel;
//步骤3：调用播放视频方法
mSuperPlayerView.playWithModel(mode);
```

方法二：

```
//步骤1：创建试看信息 mode
VipWatchModel vipWatchModel = new VipWatchModel("可试看%ss，开通 VIP 观看完整视频",15);
//步骤2：调用设置试看功能方法
mSuperPlayerView.setVipWatchModel(vipWatchModel);
```

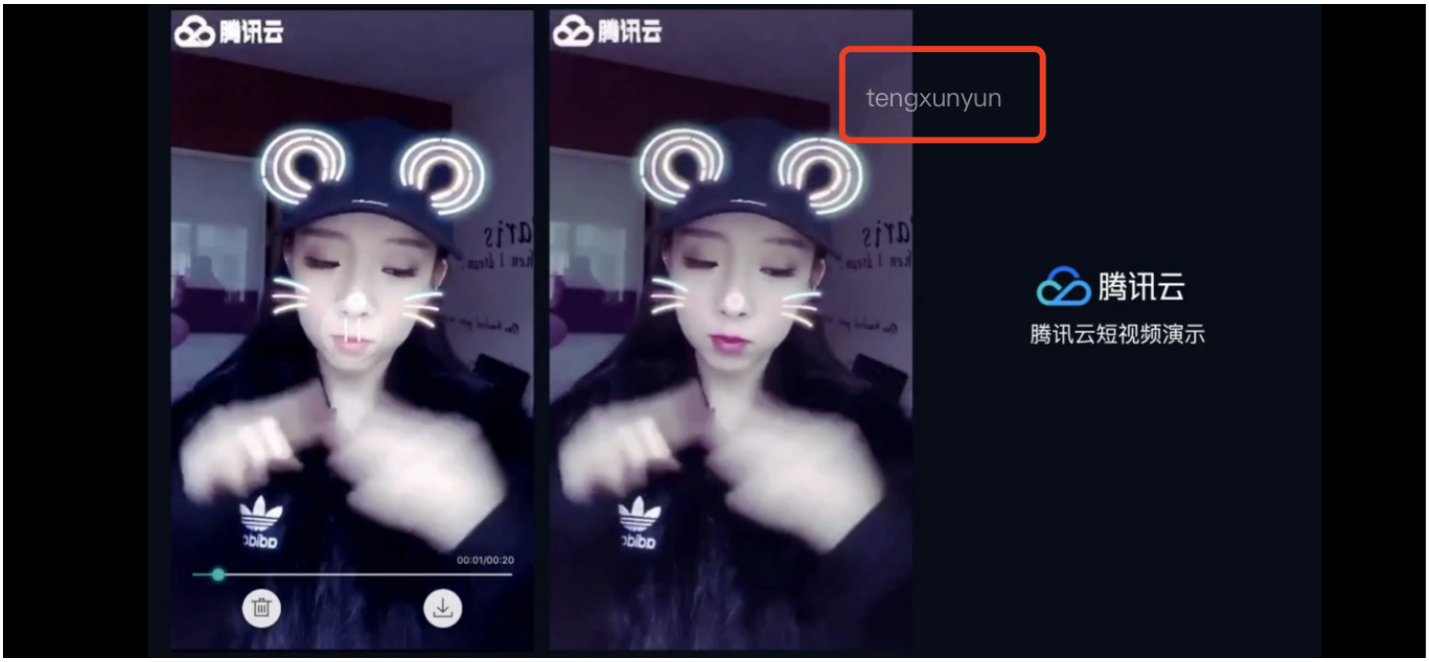
```
public VipWatchModel(String tipStr, long canWatchTime)
```

VipWatchModel 接口参数说明：

参数名	类型	描述
tipStr	String	试看提示信息
canWatchTime	Long	试看时长，单位为妙

6、动态水印

超级播放器支持在播放界面添加不规则跑动的文字水印，有效防盗录。全屏播放模式和窗口播放模式均可展示水印，开发者可修改水印文本、文字大小、颜色。功能效果可在 **腾讯云视立方 App** > **播放器** > **超级播放器** > **动态水印** 演示视频中体验。



方法一：

```
//步骤1：创建视频mode
SuperPlayerModel mode = new SuperPlayerModel();
//...添加视频源信息
//步骤2：创建水印信息mode
DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig("shipinyun", 30, Color.parseColo
r("#80FFFFFF"));
mode.dynamicWaterConfig = dynamicWaterConfig;
//步骤3：调用播放视频方法
mSuperPlayerView.playWithModel(mode);
```

方法二：

```
//步骤1：创建水印信息mode
DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig("shipinyun", 30, Color.parseColo
r("#80FFFFFF"));
//步骤2：调用设置动态水印功能方法
mSuperPlayerView.setDynamicWatermarkConfig(dynamicWaterConfig);
```

```
public DynamicWaterConfig(String dynamicWatermarkTip, int tipTextSize, int tipTextColor)
```

接口参数说明

参数名	类型	描述
dynamicWatermarkTip	String	水印文本信息
tipTextSize	int	文字大小
tipTextColor	int	文字颜色

Demo 体验

更多完整功能可直接运行工程 Demo，或扫码下载移动端 Demo 腾讯云视立方 App 体验。

运行工程 Demo

1. 在 Android Studio 的导航栏选择 **File** > **Open**，在弹框中选择 **Demo** 工程目录： $SuperPlayer_Android/Demo，待成功导入 Demo 工程后，单击 **Run app**，即可成功运行 Demo 。
2. 成功运行 Demo 后如下图，进入**播放器** > **超级播放器**，可体验播放器功能。

腾讯云视立方 App

在 **腾讯云视立方 App** > **播放器** 中可体验更多超级播放器功能。

接口说明

最近更新时间：2022-06-08 10:02:59

TXVodPlayer

点播播放器

请参见 TXVodPlayer。

主要负责从指定的点播流地址拉取音视频数据，并进行解码和本地渲染播放。

播放器包含如下能力：

- 支持 FLV、MP4 及 HLS 多种播放格式，支持 基础播放（URL 播放） 和 点播播放（Fileid 播放）两种播放方式 。
- 屏幕截图，可以截取当前播放流的视频画面。
- 通过手势操作，调节亮度、声音、进度等。
- 可以手动切换不同的清晰度，也可根据网络带宽自适应选择清晰度。
- 可以指定不同倍速播放，并开启镜像和硬件加速。
- 完整能力，请参见 点播超级播放器 - 能力清单。

播放器配置接口

API	描述
setConfig	设置播放器配置信息，配置信息请参见 TXVodPlayConfig
setPlayerView	设置播放器的视频渲染 TXCloudVideoView
setPlayerView	设置播放器的视频渲染 TextureView
setSurface	设置播放器的视频渲染 SurfaceView

播放基础接口

API	描述
startPlay	播放 HTTP URL 形式地址
startPlay	以 fileId 形式播放
stopPlay	停止播放
isPlaying	是否正在播放
pause	暂停播放，停止获取流数据,保留最后一帧画面
resume	恢复播放，重新获取流数据
seek	跳转到视频流指定时间点，单位秒

API	描述
seek	跳转到视频流指定时间点，单位毫秒
getCurrentPlaybackTime	获取当前播放位置，单位秒
getBufferDuration	获取缓存的总时长，单位秒
getDuration	获取总时长，单位秒。
getPlayableDuration	获取可播放时长，单位秒。
getWidth	获取视频宽度。
getHeight	获取视频高度。
setAutoPlay	设置点播是否 startPlay 后自动开始播放，默认自动播放。
setStartTime	设置播放开始时间。
setToken	加密 HLS 的 token。
setLoop	设置是否循环播放。
isLoop	返回是否循环播放状态。

视频相关接口

API	描述
enableHardwareDecode	启用或禁用视频硬解码。
snapshot	获取当前视频帧图像。 **注意：由于获取当前帧图像是比较耗时的操作，所以截图会通过异步回调出来。**
setMirror	设置镜像。
setRate	设置点播的播放速率，默认1.0。
getBitrateIndex	返回当前播放的码率索引。
setBitrateIndex	设置当前正在播放的码率索引，无缝切换清晰度。清晰度切换可能需要等待一小段时间。
setRenderMode	设置 图像平铺模式。
setRenderRotation	设置 图像渲染角度。

音频相关接口

API	描述

API	描述
setMute	设置是否静音播放。
setAudioPlayoutVolume	设置音量大小，范围：0 – 100。
setRequestAudioFocus	设置是否自动获取音频焦点 默认自动获取。

事件通知接口

API	描述
setPlayListener	设置播放器的回调（已弃用，建议使用 setVodListener）。
setVodListener	设置播放器的回调。
onNotifyEvent	点播播放事件通知。
onNetSuccess	点播播放网络状态通知。
onNetFailed	播放 fileId 网络异常通知。

TRTC 相关接口

通过以下接口，可以把点播播放器的音视频流通过 TRTC 进行推送，更多 TRTC 服务请参见 TRTC 产品概述。

API	描述
attachTRTC	点播绑定到 TRTC 服务。
detachTRTC	点播解绑 TRTC 服务。
publishVideo	开始推送视频流。
unpublishVideo	取消推送视频流。
publishAudio	开始推送音频流。
unpublishAudio	取消推送音频流。

ITXVodPlayListener

腾讯云点播回调通知。

SDK 基础回调

API	描述
onPlayEvent	点播播放事件通知，请参见 播放事件列表、事件参数。

API	描述
onNetStatus	点播播放器 网络状态通知。

TXVodPlayConfig

点播播放器配置类。

基础配置接口

API	描述
setConnectRetryCount	设置播放器重连次数。
setConnectRetryInterval	设置播放器重连间隔，单位秒。
setTimeout	设置播放器连接超时时间，单位秒。
setCacheFolderPath	设置点播缓存目录，点播 MP4、HLS 有效。
setMaxCacheItems	设置缓存文件个数。
setPlayerType	设置播放器类型。
setHeaders	设置自定义 HTTP headers。
setEnableAccurateSeek	设置是否精确 seek，默认 true。
setAutoRotate	播放 MP4 文件时，若设为 YES 则根据文件中的旋转角度自动旋转。 旋转角度可在 PLAY_EVT_CHANGE_ROTATION 事件中获得。默认 YES。
setSmoothSwitchBitrate	平滑切换多码率 HLS，默认 false。
setCacheMp4ExtName	缓存 MP4 文件扩展名。
setProgressInterval	设置进度回调间隔，单位毫秒。
setMaxBufferSize	最大预加载大小，单位 MB。

错误码表

常规事件

code	事件定义	含义说明
2004	PLAY_EVT_PLAY_BEGIN	视频播放开始（若有转菊花效果，此时将停止）。
2005	PLAY_EVT_PLAY_PROGRESS	视频播放进度，会通知当前播放进度、加载进度和总体时长。

code	事件定义	含义说明
2007	PLAY_EVT_PLAY_LOADING	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件。
2014	PLAY_EVT_VOD_LOADING_END	视频播放 loading 结束，视频继续播放。
2006	PLAY_EVT_PLAY_END	视频播放结束。
2013	PLAY_EVT_VOD_PLAY_PREPARED	播放器已准备完成，可以播放。
2003	PLAY_EVT_RCV_FIRST_I_FRAME	网络接收到首个可渲染的视频数据包（IDR）。
2009	PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2011	PLAY_EVT_CHANGE_ROTATION	MP4 视频旋转角度。

警告事件

code	事件定义	含义说明
-2301	PLAY_ERR_NET_DISCONNECT	网络断连,且经多次重连亦不能恢复,更多重试请自行重启播放。
-2305	PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
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 (重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT)。
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-2304	PLAY_ERR_HEVC_DECODE_FAIL	H265 解码失败。
-2303	PLAY_ERR_FILE_NOT_FOUND	播放的文件不存在。

超级播放器 Adapter

最近更新时间：2022-03-09 16:40:18

产品概述

腾讯云视立方 Android 超级播放器 Adapter 为云点播提供给客户希望使用第三方播放器或自研播放器开放的对接云 PAAS 资源的播放器插件，常用于有自定义播放器功能需求的用户。

SDK下载

腾讯云视立方 Android 超级播放器 Adapter SDK 和 Demo 项目下载地址是 TXCPlayerAdapterSDK_Android。

阅读对象

本文档部分内容为腾讯云专属能力，使用前请开通 腾讯云 相关服务，未注册用户可注册账号 免费试用。

集成指引

集成 SDK，拷贝 TXCPlayerAdapter-release-1.0.0.aar 到 libs目录，添加依赖项：

```
implementation(name:'TXCPlayerAdapter-release-1.0.0', ext:'aar')
```

添加混淆脚本：

```
-keep class com.tencent.** { *; }
```

使用播放器

变量声明，播放器主类为 ITXCPlayerAssistor ，创建后即可播放视频。

fileId 一般是在视频上传后，由服务器返回：

1. 客户端视频发布后，服务器会返回 fileId 到客户端。
2. 服务端视频上传，在 确认上传 的通知中包含对应的 fileId。

如果文件已存在腾讯云，则可以进入 媒资管理 ，找到对应的文件。单击后在右侧视频详情中，可以看到相关参数。

```
//psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
private String mFileId, mPSign;
ITXCPlayerAssistor mPlayerAssistor = TXCPlayerAdapter.createPlayerAssistor(mFileId, mPSign);
```

初始化

```
// 初始化
TXCPlayerAdapter.init(appId); //appid 在腾讯云点播申请
TXCPlayerAdapter.setLogEnable(true); //开启log

mSuperPlayerView = findViewById(R.id.sv_videoplayer);
mPlayerAssistor = TXCPlayerAdapter.createPlayerAssistor(mFileId, mPSign);
```

请求视频信息和播放

```
mPlayerAssistor.requestVideoInfo(new ITXCRequestVideoInfoCallback() {

@Override
public void onError(int errCode, String msg) {
Log.d(TAG, "onError msg = " + msg);
runOnUiThread(new Runnable() {
@Override
public void run() {
Toast.makeText(VideoActivity.this, "onError msg = " + msg, Toast.LENGTH_SHORT).show();
}
});
}

@Override
public void onSuccess() {
Log.d(TAG, "onSuccess");
TXCStreamingInfo streamingInfo = mPlayerAssistor.getStreamingInfo();
Log.d(TAG, "streamingInfo = " + streamingInfo);
runOnUiThread(new Runnable() {
@Override
public void run() {
if (mPlayerAssistor.getStreamingInfo() != null) {
//播放视频
mSuperPlayerView.play(mPlayerAssistor.getStreamingInfo().playUrl);
} else {
Toast.makeText(VideoActivity.this, "streamInfo = null", Toast.LENGTH_SHORT).show();
}
}
});
```

```
}
});
```

使用完后销毁 Player

```
TXCPlayerAdapter.destroy();
```

SDK 接口列表

初始化 TXCPlayerAdatper

说明

初始化 Adapter，每次

接口

```
TXCPlayerAdapter.init(String appId);
```

参数说明

appId：填写 appid（如果使用了子应用，则填 subappid）

销毁 TXCPlayerAdatper

说明

销毁 Adapter，当程序退出后调用。

接口

```
TXCPlayerAdapter.destroy();
```

创建播放器辅助类

说明

通过播放器辅助类可以获取播放 fileId 相关信息以及处理 DRM 加密接口等。

接口

```
ITXCPlayerAssistor playerAssistor = TXCPlayerAdapter.createPlayerAssistor(String fileId, String pSign);
```

参数说明

参数名	类型	描述
fileId	String	要播放的视频fileId
pSign	String	超级播放器签名

销毁播放器辅助类

说明

销毁辅助类，在退出播放器或者切换了下一个视频播放的时候调用。

接口

```
TXCPlayerAdapter.destroyPlayerAssistor(ITXCPlayerAssistor assistor);
```

请求视频播放信息

说明

本接口会请求腾讯云点播服务器，获取播放视频的流信息等。

接口

```
playerAssistor.requestVideoInfo(ITXCRequestVideoInfoCallback callback);
```

参数说明

参数名	类型	描述
callback	ITXCRequestVideoInfoCallback	异步回调函数

获取视频的基本信息

说明

获取视频信息， 必须是在 playerAssistor.requestPlayInfo 回调之后才生效。

接口

```
TXCVideoBasicInfo playerAssistor.getVideoBasicInfo();
```

参数说明

TXCVideoBasicInfo：参数如下

参数名	类型	描述

参数名	类型	描述
name	String	视频名称。
duration	Float	视频时长，单位：秒。
description	String	视频描述。
coverUrl	String	视频封面。

获取视频流信息

说明

获取视频流信息列表，必须是在 playerAssistor.requestPlayInfo 回调之后才生效。

接口

```
TXCStreamingInfo playerAssistor.getStreamimgInfo();
```

参数说明

TXCStreamingInfo

参数名	类型	描述
playUrl	String	播放 URL。
subStreams	List	自适应码流子流信息，类型为 SubStreamInfo。

SubStreamInfo

参数名	类型	描述
type	String	子流的类型，目前可能的取值仅有 video。
width	Int	子流视频的宽，单位：px。
height	Int	子流视频的高，单位：px。
resolutionName	String	子流视频在播放器中展示的规格名。

获取关键帧打点信息

说明

获取视频关键帧打点信息，必须是在 playerAssistor.requestPlayInfo 回调之后才生效。

接口

```
List<TXCKeyFrameDescInfo> playerAssistor.getKeyFrameDescInfo();
```

参数说明

TXCKeyFrameDescInfo

参数名	类型	描述
timeOffset	Float	1.1
content	String	"片头开始..."

获取缩略图信息

说明

获取缩略图信息，必须是在 playerAssistor.requestPlayInfo 回调之后才生效。

接口

```
TXCImageSpriteInfo playerAssistor.getImageSpriteInfo();
```

参数说明

TCXImageSpriteInfo

参数名	类型	描述
imageUrls	List	缩略图下载 URL 数组，类型为 String 。
webVttUrl	String	缩略图 VTT 文件下载 URL 。

定制开发
直播场景
接入文档

最近更新时间：2022-05-31 15:38:59

本文档主要介绍使用腾讯云视立方 SDK 实现直播播放的方法。

示例代码

针对开发者的接入反馈的高频问题，腾讯云提供有更加简洁的 API-Example 工程，方便开发者可以快速的了解相关 API 的使用，欢迎使用。

所属平台	GitHub 地址
iOS	Github
Android	Github

准备工作

1. 为了您更好的产品体验，建议您开通 云直播 相关服务。若您不使用云直播服务，可略过此步骤。
2. 下载 Android Studio，您可以进入 Android Studio 官网 下载安装，如已下载可略过该步骤。

通过本文你可以学会

- 如何集成腾讯云视立方 Android 播放器 SDK
- 如何使用播放器 SDK 进行直播播放
- 如何使用播放器 SDK 底层能力实现更多直播播放功能

SDK集成

步骤1：下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

步骤2：配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 Application 类中进行如下设置：

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // 获取到的 licence url
String licenceKey = ""; // 获取到的 licence key
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
TXLiveBase.setListener(new TXLiveBaseListener() {
@Override
public void onLicenceLoaded(int result, String reason) {
Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
}
});
}
}
```

步骤3：添加 View

SDK 默认提供 TXCloudVideoView 用于视频渲染，我们第一步要做的就是在布局 xml 文件里加入如下一段代码：

```
<com.tencent.rtmp.ui.TXCloudVideoView
android:id="@+id/video_view"
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:layout_centerInParent="true"
android:visibility="gone"/>
```

步骤4：创建 Player

视频云 SDK 中的 **TXLivePlayer** 模块负责实现直播播放功能，并使用 **setPlayerView** 接口将它与我们刚添加到界面上的 **video_view** 控件进行关联。

```
//mPlayerView 即步骤1中添加的界面 view
TXCloudVideoView mView = (TXCloudVideoView) view.findViewById(R.id.video_view);

//创建 player 对象
TXLivePlayer mLivePlayer = new TXLivePlayer(getActivity());

//关键 player 对象与界面 view
mLivePlayer.setPlayerView(mView);
```

步骤5：启动播放

```
String flvUrl = "http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
mLivePlayer.startPlay(flvUrl, TXLivePlayer.PLAY_TYPE_LIVE_FLV); //推荐 FLV
```

可选值	枚举值	含义
PLAY_TYPE_LIVE_RTMP	0	传入的 URL 为 RTMP 直播地址
PLAY_TYPE_LIVE_FLV	1	传入的 URL 为 FLV 直播地址
PLAY_TYPE_LIVE_RTMP_ACC	5	低延迟链路地址（仅适合于连麦场景）
PLAY_TYPE_VOD_HLS	3	传入的 URL 为 HLS（m3u8）播放地址

> **说明：**
> 在 App 上我们不推荐使用 HLS 这种播放协议播放直播视频源（虽然它很适合用来做点播），因为延迟太高，在 App 上推荐使用 LIVE_FLV 或者 LIVE_RTMP 播放协议。

步骤6：结束播放

结束播放时**需要销毁 view 控件**，尤其是在下次 startPlay 之前，否则**会产生大量的内存泄露以及闪屏问题**。

同时，在退出播放界面时，**需要调用渲染 View 的onDestroy()函数**，否则**可能会产生内存泄露和 Receiver not registered** 报警。

```
@Override
public void onDestroy() {
super.onDestroy();
mLivePlayer.stopPlay(true); // true 代表清除最后一帧画面
mView.onDestroy();
}
```

stopPlay 的布尔型参数含义为：“是否清除最后一帧画面”。早期版本的 RTMP SDK 的直播播放器没有 pause 的概念，所以通过这个布尔值来控制最后一帧画面的清除。

如果是点播播放结束后，也想保留最后一帧画面，您可以在收到播放结束事件后什么也不做，默认停在最后一帧。

功能使用

本节将介绍常见直播播放功能的使用方式。

1、画面调整

- **view：大小和位置**
  如需修改画面的大小及位置，直接调整**步骤1**中添加的 video_view 控件的大小和位置即可。
- **setRenderMode：铺满或适应**

可选值	含义
RENDER_MODE_FULL_FILL_SCREEN	将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不会留黑边，但可能因为部分区域被裁剪而显示不全
RENDER_MODE_ADJUST_RESOLUTION	将图像等比例缩放，适配最长边，缩放后的宽和高都不会超过显示区域，居中显示，画面可能会留有黑边

- **setRenderRotation：画面旋转**

可选值	含义
RENDER_ROTATION_PORTRAIT	正常播放（Home 键在画面正下方）
RENDER_ROTATION_LANDSCAPE	画面顺时针旋转270度（Home 键在画面正左方）

```
// 设置填充模式
mLivePlayer.setRenderMode(TXLiveConstants.RENDER_MODE_ADJUST_RESOLUTION);
// 设置画面渲染方向
mLivePlayer.setRenderRotation(TXLiveConstants.RENDER_ROTATION_LANDSCAPE);
```

最长边填充

完全填充

横屏模式

2、暂停播放

对于直播播放而言，并没有真正意义上的暂停，所谓的直播暂停，只是**画面冻结**和**关闭声音**，而云端的视频源还在不断地更新着，所以当您调用 resume 的时候，会从最新的时间点开始播放，这跟点播是有很大不同的（点播播放器的暂停和继续与播放本地视频文件时的表现相同）。

```
// 暂停
mLivePlayer.pause();
// 继续
mLivePlayer.resume();
```

3、消息接收

此功能可以在推流端将一些自定义 message 随着音视频线路直接下发到观众端，适用场景如下：

- 冲顶大会：推流端将**题目**下发到观众端，可以做到“音-画-题”完美同步。
- 秀场直播：推流端将**歌词**下发到观众端，可以在播放端实时绘制出歌词特效，因而不受视频编码的降质影响。

- 在线教育：推流端将**激光笔**和**涂鸦**操作下发到观众端，可以在播放端实时地划圈划线。

通过如下方案可以使用此功能：

- TXLivePlayConfig 中的 **setEnableMessage** 开关置为 **true**。
- TXLivePlayer 通过 **TXLivePlayListener** 监听消息，消息编号：**PLAY_EVT_GET_MESSAGE（2012）**

```
//Android 示例代码
mTXLivePlayer.setPlayListener(new ITXLivePlayListener() {
@Override
public void onPlayEvent(int event, Bundle param) {
if (event == TXLiveConstants.PLAY_ERR_NET_DISCONNECT) {
roomListenerCallback.onDebugLog("[AnswerRoom] 拉流失败：网络断开");
roomListenerCallback.onError(-1, "网络断开，拉流失败");
}
else if (event == TXLiveConstants.PLAY_EVT_GET_MESSAGE) {
String msg = null;
try {
msg = new String(param.getByteArray(TXLiveConstants.EVT_GET_MSG), "UTF-8");
roomListenerCallback.onRecvAnswerMsg(msg);
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
}
}
}
@Override
public void onNetStatus(Bundle status) {
}
});
```

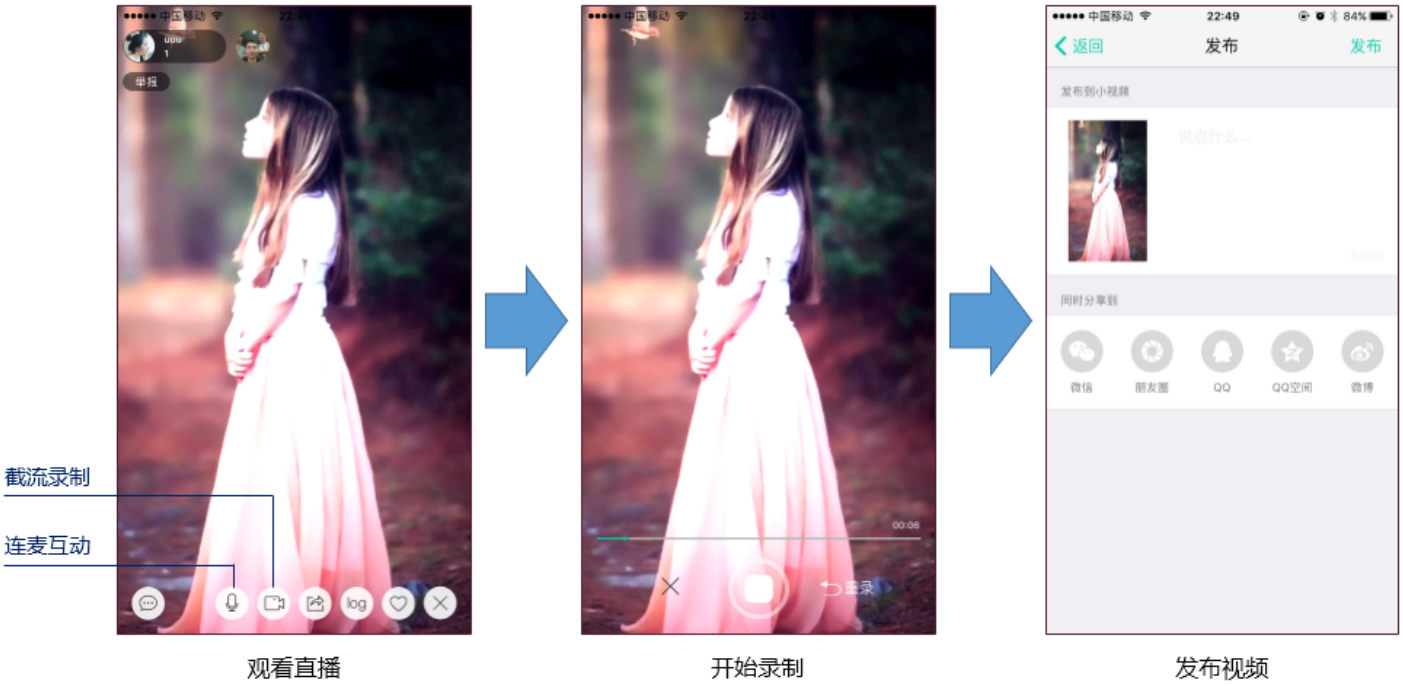
4、屏幕截图

通过调用 **snapshot**，您可以截取当前直播画面为一帧屏幕，此功能只会截取当前直播流的视频画面，如果您需要截取当前的整个 UI 界面，请调用 Android 的系统 API 来实现。


```
mLivePlayer.snapshot(new ITXSnapshotListener() {
@Override
public void onSnapshot(Bitmap bmp) {
if (null != bmp) {
//获取到截图 bitmap
}
}
});
```

5、截流录制

截流录制是直播播放场景下的一种扩展功能：观众在观看直播时，可以通过单击录制按钮把一段直播的内容录制下来，并通过视频分发平台（如腾讯云的点播系统）发布出去，这样就可以在微信朋友圈等社交平台上以 UGC 消息的形式进行传播。




```
//指定一个 ITXVideoRecordListener 用于同步录制的进度和结果
mLivePlayer.setVideoRecordListener(recordListener);
//启动录制，可放于录制按钮的响应函数里，目前只支持录制视频源，弹幕消息等等目前还不支持
mLivePlayer.startRecord(int recordType);
// ...
// ...
//结束录制，可放于结束按钮的响应函数里
mLivePlayer.stopRecord();
```

- 录制的进度以时间为单位，由 ITXVideoRecordListener 的 onRecordProgress 通知出来。
- 录制好的文件以 MP4 文件的形式，由 ITXVideoRecordListener 的 onRecordComplete 通知出来。
- 视频的上传和发布由 TXUGCPublish 负责，具体使用方法可以参考 短视频 – 文件发布。

6、清晰度无缝切换

日常使用中，网络情况在不断发生变化。在网络较差的情况下，最好适度降低画质，以减少卡顿；反之，网速比较好，可以观看更高画质。
传统切流方式一般是重新播放，会导致切换前后画面衔接不上、黑屏、卡顿等问题。使用无缝切换方案，在不中断直播的情况下，能直接切到另条流上。

清晰度切换在直播开始后，任意时间都可以调用。调用方式如下

```
// 正在播放的是流 http://5815.liveplay.myqcloud.com/live/5815_62fe94d692ab11e791eae435c87f075e.flv，
// 现切换到码率为900kbps的新流上
```

```
mLivePlayer.switchStream("http://5815.liveplay.myqcloud.com/live/5815_62fe94d692ab11e791eae435c87f075e_900.flv");
```

> 说明：
> 清晰度无缝切换功能需要在后台配置 PTS 对齐，如您需要可 提交工单 申请使用。

7、直播回看

时移功能是腾讯云推出的特色能力，可以在直播过程中，随时回退到任意直播历史时间点观看，并能在此时间点一直观看直播。非常适合游戏、球赛等互动性不高，但观看连续性较强的场景。

```
// 设置直播回看前，先调用 startPlay
// 开始播放 ...
TXLiveBase.setAppID("1253131631"); // 配置 appId
mLivePlayer.prepareLiveSeek(); // 后台请求直播起始时间
```

配置正确后，在 PLAY_EVT_PLAY_PROGRESS 事件里，当前进度就不是从0开始，而是根据实际开播时间计算而来。调用 seek 方法，就能从历史事件点重新直播

```
mLivePlayer.seek(600); // 从第10分钟开始播放
```

接入时移需要在后台打开以下配置：

- 录制：配置时移时长、时移储存时长。
- 播放：时移获取元数据。

时移功能处于公测申请阶段，如您需要可 提交工单 申请使用。

8、延时调节

腾讯云 SDK 的直播播放（LVB）功能，并非基于 ffmpeg 做二次开发， 而是采用了自研的播放引擎，所以相比于开源播放器，在直播的延迟控制方面有更好的表现，我们提供了三种延迟调节模式，分别适用于：秀场、游戏以及混合场景。

- 三种模式的特性对比

控制模式	卡顿率	平均延迟	适用场景	原理简述
极速模式	较流畅偏高	2s – 3s	美女秀场（冲顶大会）	在延迟控制上有优势，适用于对延迟大小比较敏感的场景
流畅模式	卡顿率最低	>= 5s	游戏直播（企鹅电竞）	对于超大码率的游戏直播（例如吃鸡）非常适合，卡顿率最低

控制模式	卡顿率	平均延迟	适用场景	原理简述
自动模式	网络自适应	2s - 8s	混合场景	观众端的网络越好，延迟就越低；观众端网络越差，延迟就越高

- **三种模式的对接代码**

```
TXLivePlayConfig mPlayConfig = new TXLivePlayConfig();
//
//自动模式
mPlayConfig.setAutoAdjustCacheTime(true);
mPlayConfig.setMinAutoAdjustCacheTime(1);
mPlayConfig.setMaxAutoAdjustCacheTime(5);
//
//极速模式
mPlayConfig.setAutoAdjustCacheTime(true);
mPlayConfig.setMinAutoAdjustCacheTime(1);
mPlayConfig.setMaxAutoAdjustCacheTime(1);
//
//流畅模式
mPlayConfig.setAutoAdjustCacheTime(false);
mPlayConfig.setCacheTime(5);
//
mLivePlayer.setConfig(mPlayConfig);
//设置完成之后再启动播放
```

> **说明：**
> 更多关于卡顿和延迟优化的技术知识，可以阅读 视频卡顿怎么办?

9、超低延时播放

支持400ms左右的超低延迟播放，是腾讯云直播播放器的一个特点，它可以用于一些对时延要求极为苛刻的场景，例如远程夹娃娃或者主播连麦等，关于这个特性，您需要知道：

- **该功能是不需要开通的**
  该功能并不需要提前开通，但是要求直播流必须位于腾讯云，跨云商实现低延时链路的难度不仅仅是技术层面的。
- **播放地址需要带防盗链**
  播放 URL 不能用普通的 CDN URL，必须要带防盗链签名，防盗链签名的计算方法见 txTime 与 txSecret。
- **播放类型需要指定 ACC**
  在调用 startPlay 函数时，需要指定 type 为 **PLAY_TYPE_LIVE_RTMP_ACC**，SDK 会使用 RTMP-UDP 协议

拉取直播流。

- **该功能有并发播放限制**
  目前最多同时10路并发播放，设置这个限制的原因并非是技术能力限制，而是希望您只考虑在互动场景中使用（例如连麦时只给主播使用， 或者夹娃娃直播中只给操控娃娃机的玩家使用），避免因为盲目追求低延时而产生不必要的费用损失（低延迟线路的价格要贵于 CDN 线路）。
- **Obs 的延时是不达标的**
  推流端如果是 TXLivePusher，请使用 setVideoQuality 将 quality 设置为 MAIN_PUBLISHER 或者 VIDEO_CHAT。Obs 的推流端积压比较严重，是无法达到低延时效果的。
- **该功能按播放时长收费**
  本功能按照播放时长收费，费用跟拉流的路数有关系，跟音视频流的码率无关，具体价格请参考 价格总览。

10、获取视频信息

播放器 SDK 通过 URL 字符串播放视频，URL 中本身不包含视频信息。为获取相关信息，需要通过访问云端服务器加载到相关视频信息，因此 SDK 只能以事件通知的方式将视频信息发送到您的应用程序中，更多内容参见 事件监听。

例如您可以通过 onNetStatus 的 NET_STATUS_VIDEO_WIDTH 和 NET_STATUS_VIDEO_HEIGHT 获取视频的宽和高。具体使用方法见 状态反馈（onNetStatus）。

事件监听

您可以为 TXLivePlayer 对象绑定一个 TXLivePlayListener，之后 SDK 的内部状态信息均会通过 onPlayEvent（事件通知）和 onNetStatus（状态反馈）通知给您。

事件通知（onPlayEvent）

1. 播放事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始
PLAY_EVT_PLAY_PROGRESS	2005	视频播放进度，会通知当前播放进度、加载进度 和总体时长
PLAY_EVT_PLAY_LOADING	2007	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件
PLAY_EVT_VOD_LOADING_END	2014	视频播放 loading 结束，视频继续播放

不要在收到 PLAY_LOADING 后隐藏播放画面：因为 PLAY_LOADING -> PLAY_BEGIN 的时间长短是不确定的，可能是5s或5ms，有些用户考虑在 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 loading 动画。**

2. 结束事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_ERR_NET_DISCONNECT	-2301	网络断连,且经多次重连亦不能恢复,更多重试请自行重启播放
PLAY_ERR_HLS_KEY	-2305	HLS 解密 key 获取失败

如何判断直播已结束?

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。

因此，2006和-2301都要监听，用来作为直播结束的判定事件。

3. 警告事件

如下的这些事件您可以不用关心，它只是用来告知您 SDK 内部的一些事件。

事件 ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连, 已启动自动重连 (重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT 了)
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解

状态反馈（onNetStatus）

通知每秒都会被触发一次，目的是实时反馈当前的推流器状态，它就像汽车的仪表盘，可以告知您目前 SDK 内部的一些具体情况，以便您能对当前网络状况和视频信息等有所了解。

评估参数	含义说明
NET_STATUS_CPU_USAGE	当前瞬时 CPU 使用率
NET_STATUS_VIDEO_WIDTH	视频分辨率 - 宽
NET_STATUS_VIDEO_HEIGHT	视频分辨率 - 高
NET_STATUS_NET_SPEED	当前的网络数据接收速度
NET_STATUS_VIDEO_FPS	当前流媒体的视频帧率
NET_STATUS_VIDEO_BITRATE	当前流媒体的视频码率，单位 kbps
NET_STATUS_AUDIO_BITRATE	当前流媒体的音频码率，单位 kbps

NET_STATUS_CACHE_SIZE	缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了
NET_STATUS_SERVER_IP	连接的服务器 IP

API 文档

最近更新时间：2022-05-18 09:47:04

TXLivePlayer

视频播放器

请参见 TXLivePlayer。

主要负责将直播流的音视频画面进行解码和本地渲染，包含如下技术特点：

- 针对腾讯云的拉流地址，可使用低延时拉流，实现直播连麦等相关场景。
- 针对腾讯云的拉流地址，可使用直播时移功能，能够实现直播观看与时移观看的无缝切换。
- 支持自定义的音视频数据处理，让您可以根据项目需要处理直播流中的音视频数据后，进行渲染以及播放。

SDK 基础函数

API	描述
TXLivePlayer	创建 TXLivePlayer 实例。
setConfig	设置 TXLivePlayer 播放配置项。
setPlayListener	设置推流回调接口。

播放基础接口

API	描述
setPlayerView	设置播放器的视频渲染 View。
startPlay	播放器开始播放。
stopPlay	停止播放。
isPlaying	是否正在播放。
pause	暂停播放。
resume	恢复播放。
setSurface	使用 Surface 模式用于本地渲染。
setSurfaceSize	设置渲染 Surface 的大小。

播放配置接口

API	描述

API	描述
setRenderMode	设置播放渲染模式。
setRenderRotation	设置图像渲染角度。
enableHardwareDecode	开启硬件加速。
setMute	设置是否静音播放。
setAudioRoute	设置声音播放模式。
setVolume	设置音量。
switchStream	多清晰度切换。
setAudioVolumeEvaluationListener	设置音量大小回调接口。

本地录制和截图

API	描述
setVideoRecordListener	设置录制回调接口。
startRecord	启动视频录制。
stopRecord	停止视频录制。
snapshot	播放过程中本地截图。

自定义数据处理

API	描述
addVideoRawData	设置软解码数据载体 Buffer。
setVideoRawDataListener	设置软解码视频数据回调。
setAudioRawDataListener	设置音频数据回调。

直播时移接口

API	描述
prepareLiveSeek	直播时移准备。
seek	直播时移跳转。
resumeLive	恢复直播播放。

截图回调接口类

请参见 ITXSnapshotListener。

API	描述
onSnapshot	截图回调。

软解视频数据回调接口类

请参见 ITXVideoRawDataListener。

API	描述
onVideoRawDataAvailable	软解码器解出一帧数据回调一次。

音频原始数据接口类

请参见 ITXAudioRawDataListener。

API	描述
onPcmDataAvailable	音频播放数据回调，数据格式： PCM。
onAudioInfoChanged	音频播放信息回调。

播放器音量大小接口类

请参见 ITXAudioVolumeEvaluationListener。

API	描述
onAudioVolumeEvaluationNotify	播放器音量大小回调, 取值范围 [0，100]。

TXLivePlayConfig

腾讯云直播播放器的参数配置模块

请参见 TXLivePlayConfig。

主要负责 TXLivePlayer 对应的参数设置，其中绝大多数设置项在播放开始之后再设置是无效的。

常用设置项

API	描述
setAutoAdjustCacheTime	设置是否自动调整缓存时间。
setCacheTime	设置播放器缓存时间。

API	描述
setMaxAutoAdjustCacheTime	设置最大的缓存时间。
setMinAutoAdjustCacheTime	设置最小的缓存时间。
setVideoBlockThreshold	设置播放器视频卡顿报警阈值。
setConnectRetryCount	设置播放器重连次数。
setConnectRetryInterval	设置播放器重连间隔。

专业设置项

API	描述
setEnableMessage	开启消息通道。
enableAEC	设置回声消除。

ITXLivePlayListener

腾讯云直播播放的回调通知

请参见 ITXLivePlayListener。

API	描述
onPlayEvent	播放事件通知。
onNetStatus	网络状态通知。

点播场景
接入文档

最近更新时间：2022-06-27 10:40:53

准备工作

1. 为了您体验到更完整全面的播放器功能，建议您开通 云点播 相关服务，未注册用户可注册账号 试用。若您不使用云点播服务，可略过此步骤，但集成后仅可使用播放器基础能力。
2. 下载 Android Studio，您可以进入 Android Studio 官网 下载安装，如已下载可略过该步骤。

通过本文你可以学会

- 如何集成腾讯云视立方 Android 播放器SDK
- 如何使用播放器 SDK 进行点播播放
- 如何使用播放器 SDK 底层能力实现更多功能

SDK集成

步骤1：下载 SDK 开发包

下载 SDK 开发包，并按照 SDK 集成指引 将 SDK 嵌入您的 App 工程中。

步骤2：配置 License 授权

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放 License 获取相关授权。

获取到 License 信息后，在调用 SDK 的相关接口前，通过下面的接口初始化 License，建议在 Application 类中进行如下设置：

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // 获取到的 licence url
String licenceKey = ""; // 获取到的 licence key
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
TXLiveBase.setListener(new TXLiveBaseListener() {
@Override
public void onLicenceLoaded(int result, String reason) {
Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
}
});
}
}
```

步骤3: 添加 View

SDK 默认提供 TXCloudVideoView 用于视频渲染，我们第一步要做的就是在布局 xml 文件里加入如下一段代码：

```
<com.tencent.rtmp.ui.TXCloudVideoView
android:id="@+id/video_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerInParent="true"
android:visibility="gone"/>
```

步骤4：创建 Player

接下来创建一个 TXVodPlayer 的对象，并使用 setPlayerView 接口将它与我们刚添加到界面上的 video_view 控件进行关联。

```
//mPlayerView 即步骤3中添加的视频渲染 view
TXCloudVideoView mPlayerView = findViewById(R.id.video_view);
//创建 player 对象
TXVodPlayer mVodPlayer = new TXVodPlayer(getActivity());
//关联 player 对象与视频渲染 view
mVodPlayer.setPlayerView(mPlayerView);
```

步骤5：启动播放

TXVodPlayer 支持两种播放模式，您可以根据需要自行选择：

通过 URL 方式

TXVodPlayer 内部会自动识别播放协议，您只需要将您的播放 URL 传给 startPlay 函数即可。

```
// 播放 URL 视频资源
String url = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
mVodPlayer.startPlay(url);


// 播放本地视频资源
String localFile = "/sdcard/video.mp4";
mVodPlayer.startPlay(localFile);
```

通过 FileId 方式

```
// 推荐使用下面的新接口
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // 腾讯云账户的appId
"4564972819220421305", // 视频的fileId
"psignxxxxxxx"); // 如果是加密视频，必须填写
mVodPlayer.startPlay(playInfoParam);

// 旧接口，不推荐使用
TXPlayerAuthBuilder authBuilder = new TXPlayerAuthBuilder();
authBuilder.setAppId(1252463788);
authBuilder.setFileId("4564972819220421305");
mVodPlayer.startPlay(authBuilder);
```

在 媒资管理 找到对应的视频文件。在文件名下方可以看到 FileId。

通过 FileId 方式播放，播放器会向后台请求真实的播放地址。如果此时网络异常或 FileId 不存在，则会收到 TXLiveConstants.PLAY_ERR_GET_PLAYINFO_FAIL 事件，反之收到 TXLiveConstants.PLAY_EVT_GET_PLAYINFO_SUCC 表示请求成功。

步骤6：结束播放

结束播放时**记得销毁 view 控件**，尤其是在下次 startPlay 之前，否则会产生大量的内存泄露以及闪屏问题。

同时，在退出播放界面时，记得一定要调用渲染 View 的 onDestroy() 函数，否则可能会产生内存泄露和“Receiver not registered”报警。

```
@Override
public void onDestroy() {
super.onDestroy();
mVodPlayer.stopPlay(true); // true 代表清除最后一帧画面
mPlayerView.onDestroy();
}
```

> 说明：
> stopPlay 的布尔型参数含义为：“是否清除最后一帧画面”。早期版本的 RTMP SDK 的直播播放器没有 pause 的概念，所以通过这个布尔值来控制最后一帧画面的清除。

如果是点播播放结束后，也想保留最后一帧画面，您可以在收到播放结束事件后什么也不做，默认停在最后一帧。

基础功能使用

1、播放控制

开始播放

```
// 开始播放
mVodPlayer.startPlay(url)
```

暂停播放

```
// 暂停播放
mVodPlayer.pause();
```

恢复播放

```
// 恢复播放
mVodPlayer.resume();
```

结束播放

```
// 结束播放
mVodPlayer.stopPlay(true);
```

调整进度（Seek）

当用户拖拽进度条时，可调用 seek 从指定位置开始播放，播放器 SDK 支持精准 seek。

```
int time = 600; // int类型时，单位为 秒
// float time = 600; // float 类型时单位为 秒
// 调整进度
mVodPlayer.seek(time);
```

从指定时间开始播放

首次调用 startPlay 之前，支持从指定时间开始播放。

```
float startTimeInMS = 600; // 单位：毫秒
mVodPlayer.setStartTime(startTimeInMS); // 设置开始播放时间
mVodPlayer.startPlay(url);
```

2、画面调整

- **view：大小和位置**
  如需修改画面的大小及位置，直接调整 SDK 集成时 添加 View 中添加的 “video_view” 控件的大小和位置即可。
- **setRenderMode：铺满或适应**

可选值	含义
RENDER_MODE_FULL_FILL_SCREEN	将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不会留黑边，但可能因为部分区域被裁剪而显示不全
RENDER_MODE_ADJUST_RESOLUTION	将图像等比例缩放，适配最长边，缩放后的宽和高都不会超过显示区域，居中显示，画面可能会留有黑边

- **setRenderRotation：画面旋转**

可选值	含义
RENDER_ROTATION_PORTRAIT	正常播放（Home 键在画面正下方）
RENDER_ROTATION_LANDSCAPE	画面顺时针旋转270度（Home 键在画面正左方）

```
// 将图像等比例铺满整个屏幕
mVodPlayer.setRenderMode(TXLiveConstants.RENDER_MODE_FULL_FILL_SCREEN);
// 正常播放（Home 键在画面正下方）
mVodPlayer.setRenderRotation(TXLiveConstants.RENDER_ROTATION_PORTRAIT);
```

最长边填充

完全填充

横屏模式

3、变速播放

点播播放器支持变速播放，通过接口 setRate 设置点播播放速率来完成，支持快速与慢速播放，如0.5X、1.0X、1.2X、2X等。


```
// 设置1.2倍速播放
mVodPlayer.setRate(1.2);
```

4、循环播放

```
// 设置循环播放
mVodPlayer.setLoop(true);
// 获取当前循环播放状态
mVodPlayer.isLoop();
```

5、静音设置

```
// 设置静音，true 表示开启静音， false 表示关闭静音
mVodPlayer.setMute(true);
```

6、屏幕截图

通过调用 snapshot 您可以截取当前视频为一帧画面，此功能只会截取当前直播流的视频画面，如果您需要截取当前的整个 UI 界面，请调用 Android 的系统 API 来实现。


```
// 屏幕截图
mVodPlayer.snapshot(new ITXSnapshotListener() {
@Override
public void onSnapshot(Bitmap bmp) {
if (null != bmp) {
//获取到截图bitmap
}
}
});
```

7、贴片广告

播放器SDK支持在界面添加图片贴片，用于广告宣传等。实现方式如下：

- 将autoPlay为 NO，此时播放器会正常加载，但视频不会立刻开始播放。
- 在播放器加载出来后、视频尚未开始时，即可在播放器界面查看图片贴片广告。
- 待达到广告展示结束条件时，使用resume接口启动视频播放。

```
mVodPlayer.setAutoPlay(false); // 设置为非自动播放
mVodPlayer.startPlay(url); // 调用startPlay 后会加载视频，加载成功后不会自动播放
// ......
// 在播放器界面上展示广告
// ......
mVodPlayer.resume(); // 广告展示完调用 resume 开始播放视频
```

8、HTTP-REF

TXVodPlayConfig 中的 headers 可以用来设置 HTTP 请求头，例如常用的防止 URL 被到处拷贝的 Referer 字段（腾讯云可以提供更加安全的签名防盗链方案），以及用于验证客户端身份信息的 Cookie 字段。

9、硬件加速

对于蓝光级别（1080p）的画质，简单采用软件解码的方式很难获得较为流畅的播放体验，所以如果您的场景是以游戏直播为主，一般都推荐开启硬件加速。

软解和硬解的切换需要在切换之前先 **stopPlay**，切换之后再 **startPlay**，否则会产生比较严重的花屏问题。

```
mVodPlayer.stopPlay(true);
mVodPlayer.enableHardwareDecode(true);
mVodPlayer.startPlay(flvUrl, type);
```

10、清晰度设置

SDK 支持 HLS 的多码率格式，方便用户切换不同码率的播放流，从而达到播放不同清晰的目标。在收到 PLAY_EVT_PLAY_BEGIN 事件后，可以通过下面方法进行清晰度设置。

```
ArrayList<TXBitrateItem> bitrates = mVodPlayer.getSupportedBitrates(); //获取多码率数组
int index = bitrates.get(i).index; // 指定要播的码率下标
mVodPlayer.setBitrateIndex(index); // 切换码率到想要的清晰度
```

在播放过程中，可以随时通过 mVodPlayer.setBitrateIndex(int) 切换码率。切换过程中，会重新拉取另一条流的数据，SDK 针对腾讯云的多码率文件做过优化，可以做到切换无卡顿。

11、码流自适应

SDK 支持 HLS 的多码流自适应，开启相关能力后播放器能够根据当前带宽，动态选择最合适的码率播放。在收到 PLAY_EVT_PLAY_BEGIN 事件后，可以通过下面方法开启码流自适应。

```
mVodPlayer.setBitrateIndex(-1); //index 参数传入-1
```

在播放过程中，可以随时通过 mVodPlayer.setBitrateIndex(int) 切换其它码率，切换后码流自适应也随之关闭。

12、播放进度监听

点播播放中的进度信息分为2种：**加载进度**和**播放进度**，SDK 目前是以事件通知的方式将这两个进度实时通知出来的。更多事件通知内容参见事件监听。

您可以为 TXVodPlayer 对象绑定一个 **TXVodPlayerListener** 监听器，进度通知会通过 **PLAY_EVT_PLAY_PROGRESS** 事件回调到您的应用程序，该事件的附加信息中即包含上述两个进度指标。


```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
if (event == TXLiveConstants.PLAY_EVT_PLAY_PROGRESS) {
// 加载进度，单位是毫秒
int playable_duration_ms = param.getInt(TXLiveConstants.EVT_PLAYABLE_DURATION_MS);
mLoadBar.setProgress(playable_duration_ms); // 设置loading 进度条

// 播放进度，单位是毫秒
int progress_ms = param.getInt(TXLiveConstants.EVT_PLAY_PROGRESS_MS);
mSeekBar.setProgress(progress_ms); // 设置播放进度条
```

```
// 视频总长，单位是毫秒
int duration_ms = param.getInt(TXLiveConstants.EVT_PLAY_DURATION_MS);
// 可以用于设置时长显示等等
}
}

@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
}
});
```

13、播放网速监听

通过 事件监听 方式，可以在视频播放卡顿时在显示当前网速。

- 通过onNetStatus的NET_STATUS_NET_SPEED获取当前网速。具体使用方法见 状态反馈（onNetStatus）。
- 监听到PLAY_EVT_PLAY_LOADING事件后，显示当前网速。
- 收到PLAY_EVT_VOD_LOADING_END事件后，对显示当前网速的 view 进行隐藏。

```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
if (event == TXLiveConstants.PLAY_EVT_PLAY_LOADING) {
// 显示当前网速

} else if (event == TXLiveConstants.PLAY_EVT_VOD_LOADING_END) {
// 对显示当前网速的 view 进行隐藏
}
}

@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
// 获取实时速率，单位：kbps
int speed = bundle.getInt(TXLiveConstants.NET_STATUS_NET_SPEED);
}
});
```

14、获取视频分辨率

播放器 SDK 通过 URL 字符串播放视频，URL 中本身不包含视频信息。为获取相关信息，需要通过访问云端服务器加载到相关视频信息，因此 SDK 只能以事件通知的方式将视频信息发送到您的应用程序中，更多内容参见 事件监听。

可以通过下面两种方法获取分辨率信息

方法1：通过 onNetStatus **的** NET_STATUS_VIDEO_WIDTH **和** NET_STATUS_VIDEO_HEIGHT **获取视频的宽和高。具体使用方法见** 状态反馈（onNetStatus）。

方法2：在收到播放器的PLAY_EVT_VOD_PLAY_PREPARED **事件回调后，直接调用** TXVodPlayer.getWidth() **和** TXVodPlayer.getHeight() **获取当前宽高。**

```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
}

@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
//获取视频宽度
int videoWidth = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_WIDTH);
//获取视频高度
int videoHeight = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_HEIGHT);
}
});

// 获取视频宽高，需要在收到播放器的PLAY_EVT_VOD_PLAY_PREPARED 事件回调后才返回值
mVodPlayer.getWidth();
mVodPlayer.getHeight();
```

15、播放缓冲大小

在视频正常播放时，控制提前从网络缓冲的最大数据大小。如果不配置，则走播放器默认缓冲策略，保证流畅播放。

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxBufferSize(10); // 播放时最大缓冲大小。单位：MB
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

16、视频本地缓存

在短视频播放场景中，视频文件的本地缓存是很刚需的一个特性，对于普通用户而言，一个已经看过的视频再次观看时，不应该再消耗一次流量。

- **格式支持：**SDK 支持 HLS(m3u8) 和 MP4 两种常见点播格式的缓存功能。
- **开启时机：**SDK 并不默认开启缓存功能，对于用户回看率不高的场景，也并不推荐您开启此功能。
- **开启方式：**全局生效，在使用播放器开启。开启此功能需要配置两个参数：本地缓存目录及缓存大小。

```
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
if (sdcardDir != null) {
//设置播放引擎的全局缓存目录
TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/txcache");
//设置播放引擎的全局缓存目录和缓存大小，//单位MB
TXPlayerGlobalSetting.setMaxCacheSize(200);
}

//使用播放器
```

> 说明：
> 旧版本通过 TXVodPlayConfig#setMaxCacheItems 接口配置已经废弃，不推荐使用。

高级功能使用

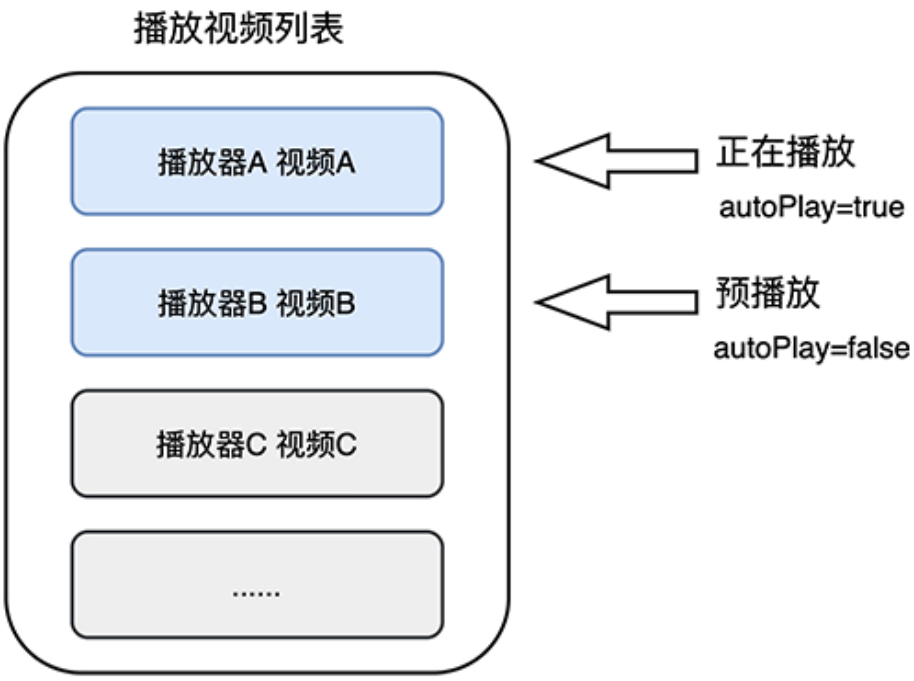
1、视频预播放

步骤1：视频预播放使用

在短视频播放场景中，视频预播放功能对于流畅的观看体验很有帮助：在观看当前视频的同时，在后台加载即将要播放的下一个视频 ，这样一来，当用户真正切换到下一个视频时，已经不需要从头开始加载了，而是可以做到立刻播放。

预播放视频会有很好的秒开效果，但有一定的性能开销，如果业务同时有较多的视频预加载需求，建议结合 视频预下载 一起使用。

这就是视频播放中无缝切换的背后技术支撑，您可以使用 TXVodPlayer 中的 setAutoPlay 开关来实现这个功能，具体做法如下：




```
// 播放视频 A: 如果将 autoPlay 设置为 true， 那么 startPlay 调用会立刻开始视频的加载和播放
String urlA = "http://1252463788.vod2.myqcloud.com/xxxxx/v.f10.mp4";
playerA.setAutoPlay(true);
playerA.startPlay(urlA);

// 在播放视频 A 的同时，预加载视频 B，做法是将 setAutoPlay 设置为 false
String urlB = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
playerB.setAutoPlay(false);
playerB.startPlay(urlB); // 不会立刻开始播放，而只会开始加载视频
```

等到视频 A 播放结束，自动（或者用户手动切换到）视频 B 时，调用 resume 函数即可实现立刻播放。

> **注意：**
> 设置了 autoPlay 为 false 之后，调用 resume 之前需要保证视频 B 已准备完成，即需要在监听到视频 B 的 PLAY_EVT_VOD_PLAY_PREPARED（2013，播放器已准备完成，可以播放）事件后调用。

```
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
// 在视频 A 播放结束的时候，直接启动视频 B 的播放，可以做到无缝切换
if (event == PLAY_EVT_PLAY_END) {
playerA.stop();
playerB.setPlayerView(mPlayerView);
playerB.resume();
```

```
}
}
```

步骤2：视频预播放缓冲配置

- 设置较大的缓冲可以更好的应对网络的波动，达到流畅播放的目的。
- 设置较小的缓冲可以帮助节省流量消耗。

预播放缓冲大小

此接口针对预加载场景（即在视频启播前，且设置 player 的 AutoPlay 为 false），用于控制启播前阶段的最大缓冲大小。

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxPreloadSize(2); // 预播放最大缓冲大小。单位：MB, 根据业务情况设置去节省流量
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

播放缓冲大小

在视频正常播放时，控制提前从网络缓冲的最大数据大小。如果不配置，则走播放器默认缓冲策略，保证流畅播放。

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMaxBufferSize(10); // 播放时最大缓冲大小。单位：MB
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

2、视频预下载

不需要创建播放器实例，预先下载视频部分内容，使用播放器时，可以加快视频启播速度，提供更好的播放体验。

在使用播放服务前，请确保先设置好 视频缓存。

> 说明：
>
> 1. TXPlayerGlobalSetting 是全局缓存设置接口，原有 TXVodConfig 的缓存配置接口废弃。
> 2. 全局缓存目录和大小设置的优先级高于播放器 TXVodConfig 配置的缓存设置。

使用示例：

```
//先设置播放引擎的全局缓存目录和缓存大小
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
//设置播放引擎的全局缓存目录和缓存大小
if (sdcardDir != null) {
TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/PlayerCache");
TXPlayerGlobalSetting.setMaxCacheSize(200); //单位MB
}
```

```
String palyrl = "http://****";
//启动预下载
final TXVodPreloadManager downloadManager = TXVodPreloadManager.getInstance(getApplicationCont
ext());
final int taskID = downloadManager.startPreload(playUrl, 3, 1920*1080, new ITXVodPreloadListener() {
@Override
public void onComplete(int taskID, String url) {
Log.d(TAG, "preload: onComplete: url: " + url);
}

@Override
public void onError(int taskID, String url, int code, String msg) {
Log.d(TAG, "preload: onError: url: " + url + ", code: " + code + ", msg: " + msg);
}

});

//取消预下载
downloadManager.stopPreload(taskID);
```

3、视频下载

视频下载支持用户在有网络的条件下下载视频，随后在无网络的环境下观看。同时播放器 SDK 提供本地加密能力，下载后的本地视频仍为加密状态，仅可通过指定播放器对视频进行解密播放，可有效防止下载后视频的非法传播，保护视频安全。

由于 HLS 流媒体无法直接保存到本地，因此也无法通过播放本地文件的方式实现 HLS 下载到本地后播放，对于该问题，您可以通过基于 TXVodDownloadManager 的视频下载方案实现 HLS 的离线播放。

> ⚠ 注意：
>
> - TXVodDownloadManager 暂不支持缓存 MP4 和 FLV 格式的文件，仅支持缓存非嵌套 HLS 格式文件。
> - 播放器 SDK 已支持 MP4 和 FLV 格式的本地文件播放。

步骤1：准备工作

TXVodDownloadManager 被设计为单例，因此您不能创建多个下载对象。用法如下：

```
TXVodDownloadManager downloader = TXVodDownloadManager.getInstance();
downloader.setDownloadPath("<指定您的下载目录>");
```

步骤2: 开始下载

开始下载有 URL 和 Fileid 两种方式，具体操作如下：

URL 方式

至少需要传入下载地址 URL，不支持嵌套 HLS 格式，仅支持非嵌套 HLS 格式下载。userName 不传入具体值时，默认为"default"。

```
downloader.startDownloadUrl("http://1500005830.vod2.myqcloud.com/43843ec0vodtranscq1500005830/00eb06a886022680114373569 84/video_10_0.m3u8", "");
```

Fileid 方式

Fileid 下载至少需要传入 AppID、 Fileid 和 qualityId。带签名视频需传入 pSign，userName 不传入具体值时，默认为“default”。

```
// QUALITYOD // 原画
// QUALITYFLU // 流畅
// QUALITYSD // 标清
// QUALITYHD // 高清
TXVodDownloadDataSource source = new TXVodDownloadDataSource(1252463788, "4564972819220421305", QUALITYHD, "", "");
downloader.startDownload(source);
```

步骤3：任务信息

在接收任务信息前，需要先设置回调 listener。

```
downloader.setListener(this);
```

可能收到的任务回调有：

回调信息	说明
void onDownloadStart(TXVodDownloadMediaInfo mediaInfo)	任务开始，表示 SDK 已经开始下载
void onDownloadProgress(TXVodDownloadMediaInfo mediaInfo)	任务进度，下载过程中，SDK 会频繁回调此接口，您可以通过mediaInfo.getProgress() 获取当前进度
void onDownloadStop(TXVodDownloadMediaInfo mediaInfo)	任务停止，当您调用 stopDownload 停止下载，收到此消息表示停止成功

回调信息	说明
void onDownloadFinish(TXVodDownloadMediaInfo mediaInfo)	下载完成，收到此回调表示已全部下载。此时下载文件可以给 TXVodPlayer 播放
void onDownloadError(TXVodDownloadMediaInfo mediaInfo, int error, String reason)	下载错误，下载过程中遇到网络断开会回调此接口，同时下载任务停止。错误码位于 TXVodDownloadManager中

由于 downloader 可以同时下载多个任务，所以回调接口里带上了 TXVodDownloadMediaInfo 对象，您可以访问 URL 或 dataSource 判断下载源，同时还可以获取到下载进度、文件大小等信息。

步骤4：中断下载

停止下载请调用 downloader.stopDownload() 方法，参数为 downloader.startDownload() 返回的对象。SDK 支持断点续传，当下载目录没有发生改变时，下次下载同一个文件时会从上次停止的地方重新开始。

步骤5：管理下载

获取所有用户账户的下载列表信息，也可获取指定用户账户的下载列表信息。

```
// 获取所有用户的下载列表信息
// 接入方可根据下载信息中的userName区分不同用户的下载列表信息
List<TXVodDownloadMediaInfo> downloadList = downloader.getDownloadMediaInfoList();
if (downloadInfoList == null || downloadInfoList.size() <= 0) return;
// 获取默认“default”用户的下载列表
List<TXVodDownloadMediaInfo> defaultUserDownloadList = new ArrayList<>();
for(TXVodDownloadMediaInfo downloadMediaInfo : downloadInfoList) {
if ("default".equals(downloadMediaInfo.getUserName())) {
defaultUserDownloadList.add(downloadMediaInfo);
}
}
```

获取某个 Fileid 相关下载信息，包括当前下载状态，获取当前下载进度，判断是否下载完成等，需要传入 AppID、 Fileid 和 qualityId。

```
// 获取某个fileId相关下载信息
TXVodDownloadMediaInfo downloadInfo = downloader.getDownloadMediaInfo(1252463788, "4564972819220421305", QUALITYHD);
int duration = downloadInfo.getDuration(); // 获取总时长
int playableDuration = downloadInfo.getPlayableDuration(); // 获取已下载的可播放时长
float progress = downloadInfo.getProgress(); // 获取下载进度
String playPath = downloadInfo.getPlayPath(); // 获取离线播放路径，传给播放器即可离线播放
```

```
int downloadState = downloadInfo.getDownloadState(); // 获取下载状态，具体参考STATE_xxx常量
boolean isDownloadFinished = downloadInfo.isDownloadFinished(); // 返回true表示下载完成
```

获取某个 URL 相关下载信息，需要传入 URL 信息。

```
// 获取某个url下载信息
TXVodDownloadMediaInfo downloadInfo = downloader.getDownloadMediaInfo("http://1253131631.vod2.myqcloud.com/26f327f9vodgzp1253131631/f4bdff799031868222924043041/playlist.m3u8");
```

删除下载信息和相关文件，需传入 TXVodDownloadMediaInfo 参数。

```
// 删除下载信息
boolean deleteRst = downloader.deleteDownloadMediaInfo(downloadInfo);
```

4、加密播放

视频加密方案主要用于在线教育等需要对视频版权进行保护的场景。如果要对您的视频资源进行加密保护，就不仅需要在播放器上做改造，还需要对视频源本身进行加密转码，亦需要您的后台和终端研发工程师都参与其中。在 视频加密解决方案 中您会了解到全部细节内容。

5、播放器配置

在调用 statPlay 之前可以通过 setConfig 对播放器进行参数配置，例如：设置播放器连接超时时间、设置进度回调间隔、设置缓存文件个数等配置，TXVodPlayConfig 支持配置的详细参数请单击 基础配置接口 了解。使用示例：

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setEnableAccurateSeek(true); // 设置是否精确 seek，默认 true
config.setMaxCacheItems(5); // 设置缓存文件个数为5
config.setProgressInterval(200); // 设置进度回调间隔，单位毫秒
config.setMaxBufferSize(50); // 最大预加载大小，单位 MB
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

启播时指定分辨率

播放 HLS 的多码率视频源，如果你提前知道视频流的分辨率信息，可以在启播前优先指定播放的视频分辨率。 播放器会查找小于或等于偏好分辨率的流进行启播，启播后没有必要再通过 setBitrateIndex 切换到需要的码流。

```
TXVodPlayConfig config = new TXVodPlayConfig();
// 传入参数为视频宽和高的乘积(宽 * 高 )，可以自定义值传入
config.setPreferredResolution(TXLiveConstants.VIDEO_RESOLUTION_720X1280);
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

设置播放进度回调时间间隔

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setProgressInterval(200); // 设置进度回调间隔，单位毫秒
mVodPlayer.setConfig(config); // 把config 传给 mVodPlayer
```

播放器事件监听

您可以为 TXVodPlayer 对象绑定一个 TXVodPlayListener 监听器，即可通过onPlayEvent（事件通知）和 onNetStatus（状态反馈）向您的应用程序同步信息。

播放事件通知（onPlayEvent）

事件 ID	数值	含义说明
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始
PLAY_EVT_PLAY_PROGRESS	2005	视频播放进度，会通知当前播放进度、加载进度 和总体时长
PLAY_EVT_PLAY_LOADING	2007	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件
PLAY_EVT_VOD_LOADING_END	2014	视频播放 loading 结束，视频继续播放

结束事件

事件 ID	数值	含义说明
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_ERR_NET_DISCONNECT	-2301	网络断连,且经多次重连亦不能恢复,更多重试请自行重启播放
PLAY_ERR_HLS_KEY	-2305	HLS 解密 key 获取失败

警告事件

如下的这些事件您可以不用关心，它只是用来告知您 SDK 内部的一些事件。

事件 ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连, 已启动自动重连 (重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT 了)
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解

连接事件

连接服务器的事件，主要用于测定和统计服务器连接时间：

事件 ID	数值	含义说明
PLAY_EVT_VOD_PLAY_PREPARED	2013	播放器已准备完成，可以播放。设置了 autoPlay 为 false 之后，需要在收到此事件后，调用 resume 才会开始播放
PLAY_EVT_RCV_FIRST_I_FRAME	2003	网络接收到首个可渲染的视频数据包（IDR）

画面事件

以下事件用于获取画面变化信息：

事件 ID	数值	含义说明
PLAY_EVT_CHANGE_RESOLUTION	2009	视频分辨率改变
PLAY_EVT_CHANGE_ROTATION	2011	MP4 视频旋转角度

视频信息事件

事件 ID	数值	含义说明
TXLiveConstants.PLAY_EVT_GET_PLAYINFO_SUCC	2010	成功获取播放文件信息

如果通过 fileId 方式播放且请求成功（接口：startPlay(TXPlayerAuthBuilder authBuilder)），SDK 会将一些请求信息通知到上层。您可以在收到 TXLiveConstants.PLAY_EVT_GET_PLAYINFO_SUCC 事件后，解析 param 获取视频信息。

视频信息	含义说明
EVT_PLAY_COVER_URL	视频封面地址
EVT_PLAY_URL	视频播放地址
EVT_PLAY_DURATION	视频时长
EVT_TIME	事件发生时间
EVT_UTC_TIME	UTC 时间
EVT_DESCRIPTION	事件说明
EVT_PLAY_NAME	视频名称
TXVodConstants.EVT_IMAGESPRIT_WEBVTTURL	雪碧图 web vtt 描述文件下载 URL，10.2版本开始支持
TXVodConstants.EVT_IMAGESPRIT_IMAGEURL_LIST	雪碧图图片下载URL，10.2版本开始支持

视频信息	含义说明
TXVodConstants.EVT_DRM_TYPE	加密类型，10.2版本开始支持

通过 onPlayEvent 获取视频播放过程信息示例：

```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
if (event == TXLiveConstants.PLAY_EVT_VOD_PLAY_PREPARED) {
// 收到播放器已经准备完成事件，此时可以调用pause、resume、getWidth、getSupportedBitrates 等接口
} else if (event == TXLiveConstants.PLAY_EVT_PLAY_BEGIN) {
// 收到开始播放事件
} else if (event == TXLiveConstants.PLAY_EVT_PLAY_END) {
// 收到开始结束事件
}
}

@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
}
});
```

播放状态反馈（onNetStatus）

状态反馈每0.5秒都会被触发一次，目的是实时反馈当前的推流器状态，它就像汽车的仪表盘，可以告知您目前 SDK 内部的一些具体情况，以便您能对当前视频播放状态等有所了解。

评估参数	含义说明
NET_STATUS_CPU_USAGE	当前瞬时 CPU 使用率
NET_STATUS_VIDEO_WIDTH	视频分辨率 – 宽
NET_STATUS_VIDEO_HEIGHT	视频分辨率 – 高
NET_STATUS_NET_SPEED	当前的网络数据接收速度
NET_STATUS_VIDEO_FPS	当前流媒体的视频帧率
NET_STATUS_VIDEO_BITRATE	当前流媒体的视频码率，单位 kbps
NET_STATUS_AUDIO_BITRATE	当前流媒体的音频码率，单位 kbps
NET_STATUS_VIDEO_CACHE	缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了

NET_STATUS_SERVER_IP	连接的服务器 IP

通过 onNetStatus 获取视频播放过程信息示例：

```
mVodPlayer.setVodListener(new ITXVodPlayListener() {
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
}

@Override
public void onNetStatus(TXVodPlayer player, Bundle bundle) {
//获取当前CPU使用率
CharSequence cpuUsage = bundle.getCharSequence(TXLiveConstants.NET_STATUS_CPU_USAGE);
//获取视频宽度
int videoWidth = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_WIDTH);
//获取视频高度
int videoHeight = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_HEIGHT);
//获取实时速率，单位：kbps
int speed = bundle.getInt(TXLiveConstants.NET_STATUS_NET_SPEED);
//获取当前流媒体的视频帧率
int fps = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_FPS);
//获取当前流媒体的视频码率，单位 kbps
int videoBitRate = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_BITRATE);
//获取当前流媒体的音频码率，单位 kbps
int audioBitRate = bundle.getInt(TXLiveConstants.NET_STATUS_AUDIO_BITRATE);
//获取缓冲区（jitterbuffer）大小，缓冲区当前长度为0，说明离卡顿就不远了
int jitterbuffer = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_CACHE);
//获取连接的服务器的IP地址
String ip = bundle.getString(TXLiveConstants.NET_STATUS_SERVER_IP);
}
});
```

场景化功能

1、基于 SDK 的 Demo 组件

基于播放器 SDK，腾讯云研发了一款 播放器组件，集质量监控、视频加密、极速高清、清晰度切换、小窗播放等功能于一体，适用于所有点播、直播播放场景。封装了完整功能并提供上层 UI，可帮助您在短时间内，打造一个媲美市面上各种流行视频 App 的播放软件。

2、开源 Github

基于播放器 SDK，腾讯云研发了沉浸式视频播放器组件、视频 Feed 流、多播放器复用组件等，而且随着版本发布，我们会提供跟多的基于用户场景的组件。您可以通过 Player_Android 下载体验。

API 文档

最近更新时间：2022-06-01 10:05:00

TXVodPlayer

点播播放器

请参见 TXVodPlayer。

主要负责从指定的点播流地址拉取音视频数据，并进行解码和本地渲染播放。

播放器包含如下能力：

- 支持 FLV、MP4 及 HLS 多种播放格式，支持 基础播放（URL 播放） 和 点播播放（Fileid 播放）两种播放方式 。
- 屏幕截图，可以截取当前播放流的视频画面。
- 通过手势操作，调节亮度、声音、进度等。
- 可以手动切换不同的清晰度，也可根据网络带宽自适应选择清晰度。
- 可以指定不同倍速播放，并开启镜像和硬件加速。
- 完整能力，请参见 点播超级播放器 - 能力清单。

播放器配置接口

API	描述
setConfig	设置播放器配置信息，配置信息请参见 TXVodPlayConfig 。
setPlayerView	设置播放器的视频渲染 TXCloudVideoView。
setPlayerView	设置播放器的视频渲染 TextureView。
setSurface	设置播放器的视频渲染 SurfaceView。
setStringOption	设置播放器业务参数，参数格式为<String,Object>。

播放基础接口

API	描述
startPlay	播放 HTTP URL 形式地址。
startPlay	以 fileId 形式播放，传入TXPlayInfoParams参数。
stopPlay	停止播放。
isPlaying	是否正在播放。
pause	暂停播放，停止获取流数据,保留最后一帧画面。
resume	恢复播放，重新获取流数据。

API	描述
seek	跳转到视频流指定时间点，单位秒。
seek	跳转到视频流指定时间点，单位毫秒。
getCurrentPlaybackTime	获取当前播放位置，单位秒。
getBufferDuration	获取缓存的总时长，单位秒。
getDuration	获取总时长，单位秒。
getPlayableDuration	获取可播放时长，单位秒。
getWidth	获取视频宽度。
getHeight	获取视频高度。
setAutoPlay	设置点播是否 startPlay 后自动开始播放，默认自动播放。
setStartTime	设置播放开始时间。
setToken	加密 HLS 的 token。
setLoop	设置是否循环播放。
isLoop	返回是否循环播放状态。

视频相关接口

API	描述
enableHardwareDecode	启用或禁用视频硬解码。
snapshot	获取当前视频帧图像。 **注意：由于获取当前帧图像是比较耗时的操作，所以截图会通过异步回调出来。**
setMirror	设置镜像。
setRate	设置点播的播放速率，默认1.0。
getBitrateIndex	返回当前播放的码率索引。
setBitrateIndex	设置当前正在播放的码率索引，无缝切换清晰度。清晰度切换可能需要等待一小段时间。
setRenderMode	设置 图像平铺模式。
setRenderRotation	设置 图像渲染角度。

音频相关接口

API	描述
setMute	设置是否静音播放。
setAudioPlayoutVolume	设置音量大小，范围：0 – 100。
setRequestAudioFocus	设置是否自动获取音频焦点 默认自动获取。

事件通知接口

API	描述
setPlayListener	设置播放器的回调（已弃用，建议使用 setVodListener ）。
setVodListener	设置播放器的回调。
onNotifyEvent	点播播放事件通知。
onNetSuccess	点播播放网络状态通知。
onNetFailed	播放 fileId 网络异常通知。

TRTC 相关接口

通过以下接口，可以把点播播放器的音视频流通过 TRTC 进行推送，更多 TRTC 服务请参见 TRTC 产品概述。

API	描述
attachTRTC	点播绑定到 TRTC 服务。
detachTRTC	点播解绑 TRTC 服务。
publishVideo	开始推送视频流。
unpublishVideo	取消推送视频流。
publishAudio	开始推送音频流。
unpublishAudio	取消推送音频流。

ITXVodPlayListener

腾讯云点播回调通知。

SDK 基础回调

API	描述
onPlayEvent	点播播放事件通知，请参见 播放事件列表、事件参数。

API	描述
onNetStatus	点播播放器 网络状态通知。

TXVodPlayConfig

点播播放器配置类。

基础配置接口

API	描述
setConnectRetryCount	设置播放器重连次数。
setConnectRetryInterval	设置播放器重连间隔，单位秒。
setTimeout	设置播放器连接超时时间，单位秒。
setCacheFolderPath	设置点播缓存目录，点播 MP4、HLS 有效。
setMaxCacheItems	设置缓存文件个数。接口废弃，请使用 TXPlayerGlobalSetting#setMaxCacheSize 进行全局配置。
setPlayerType	设置播放器类型。
setHeaders	设置自定义 HTTP headers。
setEnableAccurateSeek	设置是否精确 seek，默认 true。
setAutoRotate	播放 MP4 文件时，若设为 YES 则根据文件中的旋转角度自动旋转。 旋转角度可在 PLAY_EVT_CHANGE_ROTATION 事件中获得。默认 YES。
setSmoothSwitchBitrate	平滑切换多码率 HLS，默认 false。
setCacheMp4ExtName	缓存 MP4 文件扩展名。
setProgressInterval	设置进度回调间隔，单位毫秒。
setMaxBufferSize	最大预加载大小，单位 MB。
setMaxPreloadSize	设置预加载最大缓冲大小，单位：MB。
setExtInfo	设置拓展信息。
setPreferredResolution	播放 HLS 有多条码流时，根据设定的 preferredResolution 选最优的码流进行起播，preferredResolution 是宽高的乘积（width * height），启播前设置才有效。

TXPlayerGlobalSetting

点播播放器全局配置类

API	描述
setCacheFolderPath	设置播放引擎的cache目录。设置后，离线下载，预下载，播放器等会优先从此目录读取和存储
setMaxCacheSize	设置播放引擎的最大缓存大小。设置后会根据设定值自动清理Cache目录的文件。单位MB。

TXVodPreloadManager

点播播放器预下载接口类

API	描述
getInstance	获取 TXVodPreloadManager 实例对象，单例模式。
startPreload	启动预下载前，请先设置好播放引擎的缓存目录。 TXPlayerGlobalSetting#setCacheFolderPath 和缓存大小。 TXPlayerGlobalSetting#setMaxCacheSize。
stopPreload	停止预下载。

TXVodDownloadManager

点播播放器视频下载接口类。当前只支持下载非嵌套 m3u8 视频源，对 simpleAES 加密视频源将进行腾讯云私有加密算法加密以提升安全性。

API	描述
getInstance	获取 TXVodDownloadManager 实例对象，单例模式。
setHeaders	设置下载 HTTP 头。
setListener	设置下载回调方法，下载前必须设好。
startDownloadUrl	以 URL 方式开始下载。
startDownload	以 fileid 方式开始下载。
stopDownload	停止下载，ITXVodDownloadListener.onDownloadStop 回调时停止成功。
deleteDownloadMediaInfo	删除下载信息。
getDownloadMediaInfoList	获取所有用户的下载列表信息。
getDownloadMediaInfo	获取下载信息。

ITXVodDownloadListener

腾讯云视频下载回调通知。

API	描述
onDownloadStart	下载开始
onDownloadProgress	下载进度更新
onDownloadStop	下载停止
onDownloadFinish	下载结束
onDownloadError	下载过程中遇到错误

TXVodDownloadDataSource

腾讯云视频fildid下载源，下载时作为参数传入。

API	描述
TXVodDownloadDataSource	构造函数，传入 appid，fileid，quality，pSign，username 等参数
getAppId	获取传入的 appid
getFileId	获取传入的 fileid
getPSign	获取传入的 psign
getQuality	获取传入的 quality
getUserName	获取传入的 userName，默认“default”

TXVodDownloadMediaInfo

腾讯云视频下载信息，可获取下载进度，播放链接等等信息

API	描述
getDataSource	fileid下载时获取传入的 fileid 下载源
getDuration	获取下载的总时长
getPlayableDuration	获取已下载的可播放时长
getSize	获取下载文件总大小，只针对 fileid 下载源有效
getDownloadSize	获取已下载文件大小，只针对 fileid 下载源有效
getProgress	获取当前下载进度

API	描述
getPlayPath	获取当前播放路径，可传给 TXVodPlayer 播放
getDownloadState	获取下载状态
isDownloadFinished	判断是否下载完成

错误码表

常规事件

code	事件定义	含义说明
2004	PLAY_EVT_PLAY_BEGIN	视频播放开始（若有转菊花效果，此时将停止）。
2005	PLAY_EVT_PLAY_PROGRESS	视频播放进度，会通知当前播放进度、加载进度和总体时长。
2007	PLAY_EVT_PLAY_LOADING	视频播放 loading，如果能够恢复，之后会有 LOADING_END 事件。
2014	PLAY_EVT_VOD_LOADING_END	视频播放 loading 结束，视频继续播放。
2006	PLAY_EVT_PLAY_END	视频播放结束。
2013	PLAY_EVT_VOD_PLAY_PREPARED	播放器已准备完成，可以播放。
2003	PLAY_EVT_RCV_FIRST_I_FRAME	网络接收到首个可渲染的视频数据包（IDR）。
2009	PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2011	PLAY_EVT_CHANGE_ROTATION	MP4 视频旋转角度。

警告事件

code	事件定义	含义说明
-2301	PLAY_ERR_NET_DISCONNECT	网络断连，且经多次重连亦不能恢复，更多重试请自行重启播放。
-2305	PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
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 (重连超过三次就直接抛送 PLAY_ERR_NET_DISCONNECT)。

code	事件定义	含义说明
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-2304	PLAY_ERR_HEVC_DECODE_FAIL	H265 解码失败。
-2303	PLAY_ERR_FILE_NOT_FOUND	播放的文件不存在。

Web 端集成
超级播放器
接入指引

最近更新时间：2022-06-27 15:36:04

本文档将介绍适用于点播播放和直播播放的 Web 超级播放器（TCPlayer），它可以帮助腾讯云点播客户通过灵活的接口，快速与自有 Web 应用集成，实现视频播放功能。

概述

Web 超级播放器是通过 HTML5 的 <video> 标签以及 Flash 实现视频播放。在浏览器不支持视频播放的情况下，实现了视频播放效果的多平台统一体验，并结合腾讯云点播视频服务，提供防盗链和播放 HLS 普通加密视频等功能。

协议支持

音视频协议	用途	URL 地址格式	PC 浏览器	移动浏览器
MP3	音频	http://xxx.vod.myqcloud.com/xxx.mp3	支持	支持
MP4	点播	http://xxx.vod.myqcloud.com/xxx.mp4	支持	支持
HLS（M3U8）	直播	http://xxx.liveplay.myqcloud.com/xxx.m3u8	支持	支持
	点播	http://xxx.vod.myqcloud.com/xxx.m3u8	支持	支持
FLV	直播	http://xxx.liveplay.myqcloud.com/xxx.flv	支持	部分支持
	点播	http://xxx.vod.myqcloud.com/xxx.flv	支持	部分支持
WebRTC	直播	webrtc://xxx.liveplay.myqcloud.com/live/xxx	支持	支持

说明
- 视频编码格式仅支持 H.264 编码。
- 播放器兼容常见的浏览器，播放器内部会自动区分平台，并使用最优的播放方案。
- HLS、FLV 视频在部分浏览器环境播放需要依赖 Media Source Extensions。
- 在不支持 WebRTC 的浏览器环境，传入播放器的 WebRTC 地址会自动进行协议转换来更好的支持媒体播放。

功能支持

功能	Chrome	Firefox	Edge	QQ	Mac	iOS	微信	Android	IE

\浏览器				浏览器	Safari	Safari		Chrome	11
播放器尺寸设置	✓	✓	✓	✓	✓	✓	✓	✓	✓
续播功能	✓	✓	✓	✓	✓	✓	✓	✓	✓
倍速播放	✓	✓	✓	✓	✓	✓	✓	✓	✓
缩略图预览	✓	✓	✓	✓	–	–	–	–	✓
切换 fileID 播放	✓	✓	✓	✓	✓	✓	✓	✓	✓
镜像功能	✓	✓	✓	✓	✓	✓	✓	✓	✓
进度条标记	✓	✓	✓	✓	✓	–	–	–	✓
HLS 自适应码率	✓	✓	✓	✓	✓	✓	✓	✓	✓
Referer 防盗链	✓	✓	✓	✓	✓	✓	✓	–	✓
清晰度切换提示	✓	✓	✓	✓	–	–	–	✓	✓
试看功能	✓	✓	✓	✓	✓	✓	✓	✓	✓
HLS 标准加密播放	✓	✓	✓	✓	✓	✓	✓	✓	✓
HLS 私有加密播放	✓	✓	✓	–	–	–	• Android: ✓ • iOS: –	✓	✓
视频统计信息	✓	✓	✓	✓	–	–	–	–	–
视频数据监控	✓	✓	✓	✓	–	–	–	–	–
自定义提示文案	✓	✓	✓	✓	✓	✓	✓	✓	✓
自定义 UI	✓	✓	✓	✓	✓	✓	✓	✓	✓

弹幕	✓	✓	✓	✓	✓	✓	✓	✓	✓
水印	✓	✓	✓	✓	✓	✓	✓	✓	✓
视频列表	✓	✓	✓	✓	✓	✓	✓	✓	✓
弱网追帧	✓	✓	✓	✓	✓	✓	✓	✓	✓

说明

- 视频编码格式仅支持 H.264 编码。
- Chrome、Firefox 包括 Windows、macOS 平台。
- Chrome、Firefox、Edge 及 QQ 浏览器播放 HLS 需要加载 hls.js。
- Referer 防盗链功能是基于 HTTP 请求头的 Referer 字段实现的，部分 Android 浏览器发起的 HTTP 请求不会携带 Referer 字段。

播放器兼容常见的浏览器，播放器内部会自动区分平台，并使用最优的播放方案。例如：在 Chrome 等现代浏览器中优先使用 HTML5 技术实现视频播放，而手机浏览器上会使用 HTML5 技术或者浏览器内核能力实现视频播放。

集成指引

通过以下步骤，您就可以在网页上添加一个视频播放器。

步骤1：在页面中引入文件

在本地的项目工程内新建 index.html 文件，html 页面内引入播放器样式文件与脚本文件：

```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/tcplayer.min.css" rel="stylesheet"/>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 Webrtc 视频，需要在 tcplayer.vx.x.x.min.js 之前引入 TXLivePlayer-x.x.x.min.js。-->
<!--有些浏览器环境不支持 Webrtc，播放器会将 Webrtc 流地址自动转换为 HLS 格式地址，因此快直播场景同样需要引入hls.min.x.xx.xm.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/libs/TXLivePlayer-1.2.0.min.js"></script>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 协议的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.xm.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/hls.min.0.13.2m.js"></script>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 FLV 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 flv.min.x.x.x.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.min.1.6.2.js"></script>
<!--播放器脚本文件-->
```

```
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/tcplayer.v4.5.2.min.js"></script
>
```

建议在使用播放器 SDK 的时候自行部署资源，单击下载播放器资源。
部署解压后的文件夹，不能调整文件夹里面的目录，避免资源互相引用异常。
如果您部署的地址为 aaa.xxx.ccc，在合适的地方引入播放器样式文件与脚本文件：

```
<link href="aaa.xxx.ccc/tcplayer.min.css" rel="stylesheet"/>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.m.js。-->
<script src="aaa.xxx.ccc/libs/hls.min.0.13.2m.js"></script>
<!--播放器脚本文件-->
<script src="aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script>
```

步骤2：放置播放器容器

在需要展示播放器的页面位置加入播放器容器。例如，在 index.html 中加入如下代码（容器 ID 以及宽高都可以自定义）。

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-playsinlin
e>
</video>
```

说明：

- 播放器容器必须为 <video> 标签。
- 示例中的 player-container-id 为播放器容器的 ID，可自行设置。
- 播放器容器区域的尺寸，建议通过 CSS 进行设置，通过 CSS 设置比属性设置更灵活，可以实现例如铺满全屏、容器自适应等效果。
- 示例中的 preload 属性规定是否在页面加载后载入视频，通常为了更快的播放视频，会设置为 auto，其他可选值：meta（当页面加载后只载入元数据），none（当页面加载后不载入视频），移动端由于系统限制不会自动加载视频。
- playsinline 和 webkit-playsinline 这几个属性是为了在标准移动端浏览器不劫持视频播放的情况下实现行内播放，此处仅作示例，请按需使用。
- 设置 x5-playsinline 属性在 TBS 内核会使用 X5 UI 的播放器。

步骤3：播放器初始化

页面初始化后，即可播放视频资源。超级播放器同时支持点播和直播两种播放场景，具体播放方式如下：

- 点播播放：播放器可以通过 FileID 播放腾讯云点播媒体资源，云点播具体流程请参见 使用超级播放器播放 > 接入指引 文档。

- 直播播放：播放器通过传入 URL 地址，即可拉取直播音视频流进行直播播放。腾讯云直播 URL 生成方式可参见 自主拼装直播 URL。

通过 URL 播放（直播、点播）

在页面初始化之后，调用播放器实例上的方法，将 URL 地址传入方法。

```
var player = TCPlayer('player-container-id', {}); // player-container-id 为播放器容器 ID，必须与 html 中一致
player.src(url); // url 播放地址
```

通过 FileID 播放（点播）

在 index.html 页面初始化的代码中加入以下初始化脚本，传入在准备工作中获取到的 fileID（媒资管理）中的视频 ID 与 appID（在账号信息>基本信息中查看）。

```
var player = TCPlayer('player-container-id', { // player-container-id 为播放器容器 ID，必须与 html 中一致
fileID: '3701925921299637010', // 请传入需要播放的视频 fileID（必须）
appID: '1500005696' // 请传入点播账号的 appID（必须）
//私有加密播放需填写 psign， psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
//psign:'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwNTY5NiwiZmlsZUlkIjoiMzcwMTkyNTkyMTI5OTYzNzAxMCIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MjY4NjAxNzYsImV4cGlyZVRpbWVTdGFtcCI6MjYyNjg1OTE3OSwicGNmZyI6InByaXZhdGUiLCJ1cmxBY2Nlc3NJbmZvIjp7InQiOiI5YzkyYjBhYiJ9LCJkcm1MaWNlbnNlSW5mbyI6eyJleHBpcmVUaW1lU3RhbXAiOjI2MjY4NTkxNzksInN0cmljdE1vZGUiOjJ9fQ.Bo5K5ThInc4n8AlzIZQ-CP9a49M2mEr9-zQLH9ocQgI',
});
```

> ⚠ 注意：
>
> 要播放的视频建议使用腾讯云转码，原始视频无法保证在浏览器中正常播放。

步骤4: 更多功能

播放器可以结合云点播的服务端能力实现高级功能，比如自动切换自适应码流、预览视频缩略图、添加视频打点信息等。这些功能在 播放长视频方案 中有详细的说明，可以参考文档实现。

此外，播放器还提供更多其他功能，功能列表和使用方法请参见 功能展示 页面。

接口说明

最近更新时间：2022-06-27 15:36:13

本文档是介绍适用于直播和点播播放的 Web 超级播放器（ TCPlayer ）的相关参数以及 API。本文档适合有一定 Javascript 语言基础的开发人员阅读。

初始化参数

播放器初始化需要传入两个参数，第一个为播放器容器 ID，第二个为功能参数对象。

```
var player = TCPlayer('player-container-id', options);
```

options 参数列表

options 对象可配置的参数：

名称	类型	默认值	说明
appID	String	无	必选。
fileID	String	无	必选。
sources	Array	无	播放器播放地址，格式：[{ src: '//path/to/video.mp4', type: 'video/mp4' }]
width	String/Number	无	播放器区域宽度，单位像素，按需设置，可通过 CSS 控制播放器尺寸。
height	String/Number	无	播放器区域高度，单位像素，按需设置，可通过 CSS 控制播放器尺寸。
controls	Boolean	true	是否显示播放器的控制栏。
poster	String	无	设置封面图片完整地址（如果上传的视频已生成封面图，优先使用生成的封面图，详细请参见 云点播 - 管理视频）。
autoplay	Boolean	false	是否自动播放。
playbackRates	Array	[0.5，1，1.25，1.5，2]	设置变速播放倍率选项，仅 HTML5 播放模式有效。
loop	Boolean	false	是否循环播放。
muted	Boolean	false	是否静音播放。
preload	String	auto	是否需要预加载，有3个属性"auto"，"meta"和"none"，移动端由于系统限制，设置 auto 无效。

名称	类型	默认值	说明
swf	String	无	Flash 播放器 swf 文件的 URL。
posterImage	Boolean	true	是否显示封面。
bigPlayButton	Boolean	true	是否显示居中的播放按钮（浏览器劫持嵌入的播放按钮无法去除）。
language	String	"zh-CN"	设置语言，可选值为 "zh-CN"/"en"
languages	Object	无	设置多语言词典。
controlBar	Object	无	设置控制栏属性的参数组合，后面有详细介绍。
reportable	Boolean	true	设置是否开启数据上报。
plugins	Object	无	设置插件功能属性的参数组合，后面有详细介绍。
hlsConfig	Object	无	hls.js 的启动配置，详细内容请参见官方文档 hls.js。
webrtcConfig	Object	无	webrtc 的启动配置，后面有详细介绍。

> 注意：
> controls、playbackRates、loop、preload、posterImage 这些参数在浏览器劫持播放的状态下将无效（什么是劫持播放？）。

controlBar 参数列表

controlBar 参数可以配置播放器控制栏的功能，支持的属性有：

名称	类型	默认值	说明
playToggle	Boolean	true	是否显示播放、暂停切换按钮。
progressControl	Boolean	true	是否显示播放进度条。
volumePanel	Boolean	true	是否显示音量控制。
currentTimeDisplay	Boolean	true	是否显示视频当前时间。
durationDisplay	Boolean	true	是否显示视频时长。
timeDivider	Boolean	true	是否显示时间分割符。
playbackRateMenuButton	Boolean	true	是否显示播放速率选择按钮。
fullscreenToggle	Boolean	true	是否显示全屏按钮。
QualitySwitcherMenuButton	Boolean	true	是否显示清晰度切换菜单。

> ⚠ 注意：
> controlBar 参数在浏览器劫持播放的状态下将无效（什么是劫持播放？）。

plugins 插件参数列表

plugins 参数可以配置播放器插件的功能，支持的属性有：

名称	类型	默认值	说明
ContinuePlay	Object	无	控制续播功能，支持的属性如下： • auto: Boolean 是否在播放时自动续播 • text: String 提示文案 • btnText: String 按钮文案
VttThumbnail	Object	无	控制缩略图显示，支持的属性如下： • vttUrl: String vtt文件绝对地址，必传 • basePath: String 图片路径，非必须，不传时使用 vttUrl 的 path • imgUrl: String 图片绝对地址，非必须
ProgressMarker	Boolean	无	控制进度条显示
DynamicWatermark	Object	无	控制动态水印显示，支持的属性如下： • content: String 文字水印内容，必传 • speed: Number 水印移动速度，取值范围 0-1, 非必须
ContextMenu	Object	无	可选值如下： • mirror: Boolean 控制是否支持镜像显示 • statistic: Boolean 控制是否支持显示数据面板 • levelSwitch: Object 控制切换清晰度时的文案提示 • { • open: Boolean 是否开启提示 • switchingText: String, 开始切换清晰度时的提示文案 • switchedText: String, 切换成功时的提示文案 • switchErrorText: String, 切换失败时的提示文案 • }

webrtcConfig 参数列表

webrtcConfig 参数来控制播放 webrtc 过程中的行为表现，支持的属性有：

名称	类型	默认值	说明
connectRetryCount	Number	3	SDK 与服务器重连次数
connectRetryDelay	Number	1	SDK 与服务器重连延时
receiveVideo	Boolean	true	是否拉取视频流
receiveAudio	Boolean	true	是否拉取音频流

名称	类型	默认值	说明
showLog	Boolean	false	是否在控制台打印日志

对象方法

初始化播放器返回对象的方法列表：

名称	参数及类型	返回值及类型	说明
src()	(String)	无	设置播放地址。
ready(function)	(Function)	无	设置播放器初始化完成后的回调。
play()	无	无	播放以及恢复播放。
pause()	无	无	暂停播放。
currentTime(seconds)	(Number)	(Number)	获取当前播放时间点，或者设置播放时间点，该时间点不能超过视频时长。
duration()	无	(Number)	获取视频时长。
volume(percent)	(Number) [0，1][可选]	(Number)/设置时无返回	获取或设置播放器音量。
poster(src)	(String)	(String)/设置时无返回	获取或设置播放器封面。
requestFullscreen()	无	无	进入全屏模式。
exitFullscreen()	无	无	退出全屏模式。
isFullscreen()	无	Boolean	返回是否进入了全屏模式。
on(type，listener)	(String, Function)	无	监听事件。
one(type，listener)	(String, Function)	无	监听事件，事件处理函数最多只执行1次。
off(type，listener)	(String, Function)	无	解绑事件监听。
buffered()	无	TimeRanges	返回视频缓冲区间。
bufferedPercent()	无	值范围[0，1]	返回缓冲长度占视频时长的百分比。

名称	参数及类型	返回值及类型	说明
width()	(Number) [可选]	(Number)/设置时无返回	获取或设置播放器区域宽度，如果通过 CSS 设置播放器尺寸，该方法将无效。
height()	(Number) [可选]	(Number)/设置时无返回	获取或设置播放器区域高度，如果通过 CSS 设置播放器尺寸，该方法将无效。
videoWidth()	无	(Number)	获取视频分辨率的宽度。
videoHeight()	无	(Number)	获取视频分辨率的高度。
dispose()	无	无	销毁播放器。

> **注意：**
> 对象方法不能同步调用，需要在相应的事件（如 loadedmetadata）触发后才可以调用，除了 ready、on、one 以及 off。

事件

播放器可以通过初始化返回的对象进行事件监听，示例：

```
var player = TCPlayer('player-container-id', options);
// player.on(type, function);
player.on('error', function(error) {
// 做一些处理
});
```

其中 type 为事件类型，支持的事件有：

名称	介绍
play	已经开始播放，调用 play() 方法或者设置了 autoplay 为 true 且生效时触发，这时 paused 属性为 false。
playing	因缓冲而暂停或停止后恢复播放时触发，paused 属性为 false 。通常用这个事件来标记视频真正播放，play 事件只是开始播放，画面并没有开始渲染。
loadstart	开始加载数据时触发。
durationchange	视频的时长数据发生变化时触发。
loadedmetadata	已加载视频的 metadata。
loadeddata	当前帧的数据已加载，但没有足够的数据来播放视频的下一帧时，触发该事件。

名称	介绍
progress	在获取到媒体数据时触发。
canplay	当播放器能够开始播放视频时触发。
canplaythrough	当播放器预计能够在不停下来进行缓冲的情况下持续播放指定的视频时触发。
error	视频播放出现错误时触发。
pause	暂停时触发。
ratechange	播放速率变更时触发。
seeked	搜寻指定播放位置结束时触发。
seeking	搜寻指定播放位置开始时触发。
timeupdate	当前播放位置有变更，可以理解为 currentTime 有变更。
volumechange	设置音量或者 muted 属性值变更时触发。
waiting	播放停止，下一帧内容不可用时触发。
ended	视频播放已结束时触发。此时 currentTime 值等于媒体资源最大值。
resolutionswitching	清晰度切换进行中。
resolutionswitched	清晰度切换完毕。
fullscreenchange	全屏状态切换时触发。
webrtcevent	播放 webrtc 时的事件集合。
webrtcstats	播放 webrtc 时的统计数据。

WebrtcEvent 列表

播放器可以通过 webrtcevent 获取播发 webrtc 过程中的所有事件，示例：

```
var player = TCPlayer('player-container-id', options);
player.on('webrtcevent', function(event) {
// 从回调参数 event 中获取事件状态码及相关数据
});
```

webrtcevent 状态码如下

状态码	回调参数	介绍
1001	无	开始拉流

状态码	回调参数	介绍
1002	无	已经连接服务器
1003	无	视频播放开始
1004	无	停止拉流，结束视频播放
1005	无	连接服务器失败，已启动自动重连恢复
1006	无	获取流数据为空
1007	localSdp	开始请求信令服务器
1008	remoteSdp	请求信令服务器成功
1009	无	拉流卡顿等待缓冲中
1010	无	拉流卡顿结束恢复播放

错误码

当播放器触发 error 事件时，监听函数会返回错误码，其中3位数以上的错误码为媒体数据接口错误码。错误码列表：

名称	描述
-1	播放器没有检测到可用的视频地址。
-2	获取视频数据超时。
1	视频数据加载过程中被中断。 可能原因： • 网络中断。 • 浏览器异常中断。 解决方案： • 查看浏览器控制台网络请求信息，确认网络请求是否正常。 • 重新进行播放流程。
2	由于网络问题造成加载视频失败。 可能原因：网络中断。 解决方案： • 查看浏览器控制台网络请求信息，确认网络请求是否正常。 • 重新进行播放流程。
3	视频解码时发生错误。 可能原因：视频数据异常，解码器解码失败。 解决方案： • 尝试重新转码再进行播放，排除由于转码流程引入的问题。 • 确认原始视频是否正常。 • 请联系技术客服并提供播放参数进行定位排查。

名称	描述
4	视频因格式不支持或者服务器或网络的问题无法加载。 可能原因： • 获取不到视频数据，CDN 资源不存在或者没有返回视频数据。 • 当前播放环境不支持播放该视频格式。 解决方案： • 查看浏览器控制台网络请求信息，确认视频数据请求是否正常。 • 确认是否按照使用文档加载了对应视频格式的播放脚本。 • 确认当前浏览器和页面环境是否支持将要播放的视频格式。 • 请联系技术客服并提供播放参数进行定位排查。
5	视频解密时发生错误。 可能原因： • 解密用的密钥不正确。 • 请求密钥接口返回异常。 • 当前播放环境不支持视频解密功能。 解决方案： • 确认密钥是否正确，以及密钥接口是否返回正常。 • 请联系技术客服并提供播放参数进行定位排查。
10	点播媒体数据接口请求超时。在获取媒体数据时，播放器重试3次后仍没有任何响应，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
11	点播媒体数据接口没有返回数据。在获取媒体数据时，播放器重试3次后仍没有数据返回，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
12	点播媒体数据接口返回异常数据。在获取媒体数据时，播放器重试3次后仍返回无法解析的数据，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 播放参数有误，媒体数据接口无法处理。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
13	播放器没有检测到可以在当前播放器播放的视频数据，请对该视频进行转码操作。

名称	描述
14	HTML5 + hls.js 模式下播放 hls 出现网络异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Network Errors。
15	HTML5 + hls.js 模式下播放 hls 出现多媒体异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Media Errors。
16	HTML5 + hls.js 模式下播放 hls 出现多路复用异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Mux Errors。
17	HTML5 + hls.js 模式下播放 hls 出现其他异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Other Errors。
10008	媒体数据服务没有找到对应播放参数的媒体数据，请确认请求参数 appID fileID 是否正确，以及对应的媒体数据是否已经被删除。

超级播放器 Adapter

最近更新时间：2021-08-09 19:18:07

本文档是介绍腾讯云视立方 Web 超级播放器 Adapter，它可以帮助腾讯云客户通过灵活的接口，快速实现第三方播放器与云点播能力的结合，实现视频播放功能。超级播放器 Adapter 支持获取视频基本信息、视频流信息、关键帧与缩略图信息等，支持私有加密，本文档适合有一定 Javascript 语言基础的开发人员阅读。

集成SDK

超级播放器 Adapter 提供以下两种集成方式：

1. cdn 集成

在需要播放视频的页面中引入初始化脚本，脚本会在全局下暴露 TcAdapter 变量。

```
<script src="https://cloudcache.tencentcs.com/qcloud/video/dist/tcadapter.1.0.0.min.js"></script>
```

2. npm 集成

```
// npm install
npm install --save tcadapter
// import TcAdapter
import TcAdapter from 'tcadapter';
```

放置播放器容器

在需要展示播放器的页面加入容器，TcAdapter 仅需要承载播放视频的容器，播放样式和自定义功能可由第三方播放器或使用者自行实现

```
<video id="player-container-id">
</video>
```

使用 SDK

检测当前环境是否支持TcAdapter

```
TcAdapter.isSupported();
```

初始化Adapter，创建Adapter实例

说明

初始化 Adapter，初始化过程会请求腾讯云点播服务器，获取视频文件信息。

接口

```
const adapter = new TcAdapter('player-container-id', {
fileID: string,
appID: string,
psign: string,
hlsConfig: {}
}, callback);
```

参数说明

参数名	类型	描述
appID	String	点播账号的 appID
fileID	String	要播放的视频fileId
psign	String	超级播放器签名
hlsConfig	HlsConfig	hls相关设置，可使用hls.js支持的任意参数
callback	TcAdapterCallBack	初始化完成回调，可以在此方法之后获取视频基本信息

> **说明：**
> TcAdapter 底层基于 hls.js 实现，可以通过 HlsConfig 接收 hls.js 支持的任意参数，用于对播放行为的精细调整。

获取视频的基本信息

说明

获取视频信息， 必须是在初始化之后才生效。

接口

```
VideoBasicInfo adapter.getVideoBasicInfo();
```

参数说明

VideoBasicInfo：参数如下

参数名	类型	描述
name	String	视频名称。

参数名	类型	描述
duration	Float	视频时长，单位：秒。
description	String	视频描述。
coverUrl	String	视频封面。

获取视频流信息

说明

接口

```
List<StreamingOutput> adapter.getStreamimgOutputList();
```

参数说明

StreamingOutput

参数名	类型	描述
drmType	String	自适应码流保护类型，目前取值有 plain 和 simpleAES。plain 表示不加密，simpleAES 表示 HLS 普通加密。
playUrl	String	播放 URL。
subStreams	List	自适应码流子流信息，类型为 SubStreamInfo。

SubStreamInfo

参数名	类型	描述
type	String	子流的类型，目前可能的取值仅有 video。
width	Int	子流视频的宽，单位：px。
height	Int	子流视频的高，单位：px。
resolutionName	String	子流视频在播放器中展示的规格名。

获取关键帧打点信息

说明

接口

```
List<KeyFrameDescInfo> adapter.getKeyFrameDescInfo();
```

参数说明

KeyFrameDescInfo

参数名	类型	描述
timeOffset	Float	1.1
content	String	"片头开始..."

获取缩略图信息

说明

接口

```
ImageSpriteInfo adapter.getImageSpriteInfo();
```

参数说明

ImageSpriteInfo

参数名	类型	描述
imageUrls	List	缩略图下载 URL 数组，类型为 String 。
webVttUrl	String	缩略图 VTT 文件下载 URL 。

监听事件

说明：播放器可以通过初始化返回的对象进行事件监听，示例：

```
const adapter = TcAdapter('player-container-id', options);
adapter.on(TcAdapter.TcAdapterEvents.Error, function(error) {
// do something
});
```

其中 type 为事件类型，支持的事件包括hls原生的事件以及以下事件，可从 TcAdapter.TcAdapterEvents 中访问到事件名称：

名称	介绍
LOADEDMETADATA	通过 playcgi 获取到了相应的视频信息，在此事件回调中可以获取视频相关信息
HLSREADY	hls实例创建完成，可以在此时机调用 hls 实例对象上的各种属性和方法
ERROR	出现错误时触发，可从回调参数中查看失败具体原因

获取 Hls 实例

说明：adapter 底层基于 hls.js 实现，可以通过 adapter 实例访问到 hls 实例以及实例上的属性和方法，用于实现对播放流程的精细控制。

```
adapter.on('hlsready', () => {
const hls = adapter.hls;
// ...
})
```

> **说明：**
> 参见 hls.js 链接。

示例1：在 React 中使用 TcAdapter

> **说明**
> 参阅更多 示例。

```
import { useEffect, useRef } from 'react';
import TcAdapter from 'tcadapter';

function App() {
if (!TcAdapter.isSupported()) {
throw new Error('current environment can not support TcAdapter');
}

const videoRef = useRef(null);
useEffect(() => {
const adapter = new TcAdapter(videoRef.current, {
appID: '1500002611',
fileID: '5285890813738446783',
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NTg
5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFt
cCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNTY1
MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
}, () => {
console.log('basicInfo', adapter.getVideoBasicInfo());
});
```

```
adapter.on(TcAdapter.TcAdapterEvents.HLSREADY, () => {
const hls = adapter.hls;
// ...
})
}, []);


const play = () => {
videoRef.current.play();
}

return (
<div>
<div>
<video id="player" ref={ videoRef }></video>
</div>
<button onClick={play}>play</button>
</div>
);
}

export default App;
```

示例2: tcadapter 与 videojs 结合

> **说明**
> 参阅更多 示例。

```
// 1. videojs 播放 hls 会使用 @videojs/http-streaming，所以我们开发一套使用 tcadapter 播放的策略覆盖原有逻辑（也可以直接修改 @videojs/http-streaming 内部逻辑）

// src/js/index.js
import videojs from './video';
import '@videojs/http-streaming';
import './tech/tcadapter'; // 新增逻辑
export default videojs;


// src/js/tech/tcadapter.js
import videojs from '../video.js';
```

```
import TcAdapter from 'tcadapter';

class Adapter {
constructor(source, tech, options) {
const el = tech.el();
// 获取参数并初始化实例
const adapter = new TcAdapter(el, {
appID: '1500002611',
fileID: '5285890813738446783',
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NTg
5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFt
cCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNTY1
MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
});
adapter.on(TcAdapter.TcAdapterEvents.LEVEL_LOADED, this.onLevelLoaded.bind(this));
}

dispose() {
this.hls.destroy();
}

onLevelLoaded(event) {
this._duration = event.data.details.live ? Infinity : event.data.details.totalduration;
}

}

let hlsTypeRE = /^application\/(x-mpegURL|vnd\.apple\.mpegURL)$/i;
let hlsExtRE = /\.m3u8/i;

let HlsSourceHandler = {
name: 'hlsSourceHandler',
canHandleSource: function (source) {
// skip hls fairplay, need to use Safari resolve it.
if (source.skipHlsJs || (source.keySystems && source.keySystems['com.apple.fps.1_0'])) {
return '';
} else if (hlsTypeRE.test(source.type)) {
return 'probably';
} else if (hlsExtRE.test(source.src)) {
return 'maybe';
} else {
return '';
```

```
}
},

handleSource: function (source, tech, options) {
if (tech.hlsProvider) {
tech.hlsProvider.dispose();
tech.hlsProvider = null;
} else {
// hls关闭自动加载后，需要手动加载资源
if (options.hlsConfig && options.hlsConfig.autoStartLoad === false) {
tech.on('play', function () {
if (!this.player().hasStarted()) {
this.hlsProvider.hls.startLoad();
}
});
}
}
tech.hlsProvider = new Adapter(source, tech, options);
return tech.hlsProvider;
},
canPlayType: function (type) {
if (hlsTypeRE.test(type)) {
return 'probably';
}
return '';
}
};

function mountHlsProvider(enforce) {
if (TcAdapter && TcAdapter.isSupported() || !!enforce) {
try {
let html5Tech = videojs.getTech && videojs.getTech('Html5');
if (html5Tech) {
html5Tech.registerSourceHandler(HlsSourceHandler, 0);
}
} catch (e) {
console.error('hls.js init failed');
}
} else {
//没有引入tcadapter 或者 MSE 不可用或者x5内核禁用
}
}
```

```
mountHlsProvider();
export default Adapter;
```

Flutter 端集成
超级播放器

最近更新时间：2022-06-30 15:41:45

腾讯云视立方 Flutter 播放器是腾讯云开源的一款播放器组件，简单几行代码即可拥有类似腾讯视频强大的播放功能。包括横竖屏切换、清晰度选择、手势、小窗等基础功能，还支持视频缓存，软硬解切换，倍速播放等特殊功能。相比系统播放器，支持格式更多，兼容性更好，功能更强大。同时还支持直播流（FLV + RTMP）播放，具备首屏秒开、低延迟的优点，清晰度无缝切换、直播时移等高级能力。
本播放器基于是基于点播和直播播放的一个 Flutter 插件，同时支持 Android 和 iOS 两个平台。完全免费开源，不对播放地址来源做限制，可放心使用。

SDK 下载

腾讯云视立方 Flutter 播放器项目的地址是 Player Flutter。

项目简介

腾讯云视立方 · 播放器 SDK 是音视频终端 SDK（腾讯云视立方）的子产品 SDK 之一，基于腾讯云强大的后台能力与 AI 技术，提供视频点播和直播播放能力的强大播放载体。结合腾讯云点播或云直播使用，可以快速体验流畅稳定的播放性能。充分覆盖多类应用场景，满足客户多样需求，让客户轻松聚焦于业务发展本身，畅享极速高清播放新体验。

此项目提供了点播播放和直播播放，您可以基于播放器搭建自己的播放业务：

- 点播播放： TXVodPlayerController 对Android和iOS两个平台的点播播放器SDK进行接口封装， 你可以通过集成 TXVodPlayerController 进行点播播放业务开发。详细使用例子可以参考 DemoTXVodPlayer 。
- 直播播放： TXLivePlayerController 对Android和iOS两个平台的直播播放器SDK进行接口封装， 你可以通过集成 TXLivePlayerController 进行直播播放业务开发。详细使用例子可以参考 DemoTXLivePlayer 。

为了减少接入成本， 在 example 里提供了播放器组件（带 UI 的播放器），基于播放器组件简单的几行代码就可以搭建视频播放业务。您可以根据自己项目的需求， 把播放组件的相关代码应用到项目中去，根据需求进行调整 UI 和交互细节。

- 播放器组件：SuperPlayerController 播放器组件，对点播和直播进行了二次封装，可以方便你快速简单集成。目前是 Beta 版本，功能还在完善中。详细使用例子可以参考DemoSuperplayer。

快速集成

pubspec.yaml 配置

1. 集成 LiteAVSDK_Player 版本，默认情况下也是集成此版本。在pubspec.yaml中增加配置：

```
super_player:
git:
```

```
url: https://github.com/LiteAVSDK/Player_Flutter
path: Flutter
```

2. 如果要集成 LiteAVSDK_Professional 版本，则pubspec.yaml中配置改为：

```
super_player:
git:
url: https://github.com/LiteAVSDK/Player_Flutter
path: Flutter
ref: Professional
```

3. 更新依赖包：

```
flutter packages get
```

添加原生配置

Android 配置

在 Android 的 AndroidManifest.xml 中增加如下配置：

```
<!--网络权限-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--点播播放器悬浮窗权限-->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<!--存储-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

iOS 配置

1. 在 iOS 的Info.plist中增加如下配置：

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```

2. iOS 原生采用pod方式进行依赖，编辑podfile文件，指定你的播放器 SDK 版本，默认集成的是 Player 版SDK。

```
pod 'TXLiteAVSDK_Player' //Player版
```

3. Professional 版 SDK 集成：

```
pod 'TXLiteAVSDK_Professional' //Professional版
```

如果不指定版本，默认会安装最新的 TXLiteAVSDK_Player 最新版本。

集成过程中常见问题

- 执行flutter doctor命令检查运行环境，知道出现”No issues found!“。
- 执行flutter pub get确保所有依赖的组件都已更新成功。

申请视频播放能力 License 和集成

若您已获得相关 License 授权，需在 腾讯云视立方控制台 获取 License URL 和 License Key：

若您暂未获得 License 授权，需先参见 视频播放License 获取相关授权。

集成播放器前，需要 注册腾讯云账户，注册成功后申请视频播放能力 License， 然后通过下面方式集成，建议在应用启动时进行。

如果没有集成 License，播放过程中可能会出现异常。

```
String licenceURL = ""; // 获取到的 licence url
String licenceKey = ""; // 获取到的 licence key
SuperPlayerPlugin.setGlobalLicense(licenceURL, licenceKey);
```

点播播放器使用

点播播放器核心类 TXVodPlayerController **，详细 Demo 可参见** DemoTXVodPlayer 。

```
import 'package:flutter/material.dart';
import 'package:super_player/super_player.dart';

class Test extends StatefulWidget {
@override
State<StatefulWidget> createState() => _TestState();
}

class _TestState extends State<Test> {
late TXVodPlayerController _controller;

double _aspectRatio = 16.0 / 9.0;
String _url =
"http://1400329073.vod2.myqcloud.com/d62d88a7vodtranscq1400329073/59c68fe75285890800381567412/adp.10.m3u8";

@override
void initState() {
super.initState();
String licenceUrl = ""; // 获取到的 licence url
String licenseKey = ""; // 获取到的 licence key
SuperPlayerPlugin.setGlobalLicense(licenceUrl, licenseKey);
_controller = TXVodPlayerController();
initPlayer();
}

Future<void> initPlayer() async {
await _controller.initialize();
await _controller.startPlay(_url);
}

@override
Widget build(BuildContext context) {
return Container(
height: 220,
color: Colors.black,
child: AspectRatio(aspectRatio: _aspectRatio, child: TXPlayerVideo(controller: _controller)));
}
}
```

播放器组件使用

播放器组件核心类 SuperPlayerVideo ，创建后即可播放视频。

```
import 'package:flutter/material.dart';
import 'package:super_player/super_player.dart';

/// flutter superplayer demo
class DemoSuperplayer extends StatefulWidget {
@override
State<StatefulWidget> createState() => _DemoSuperplayerState();
}

class _DemoSuperplayerState extends State<DemoSuperplayer> {
List<SuperPlayerModel> videoModels = [];
bool _isFullScreen = false;
SuperPlayerController _controller;

@override
void initState() {
super.initState();
String licenceUrl = "填入您购买的 license 的 url";
String licenseKey = "填入您购买的 license 的 key";
SuperPlayerPlugin.setGlobalLicense(licenceUrl, licenseKey);
_controller = SuperPlayerController(context);
FTXVodPlayConfig config = FTXVodPlayConfig();
config.preferredResolution = 720 * 1280;
_controller.setPlayConfig(config);
_controller.onSimplePlayerEventBroadcast.listen((event) {
String evtName = event["event"];
if (evtName == SuperPlayerViewEvent.onStartFullScreenPlay) {
setState(() {
_isFullScreen = true;
});
} else if (evtName == SuperPlayerViewEvent.onStopFullScreenPlay) {
setState(() {
_isFullScreen = false;
});
} else {
print(evtName);
}
});
```

```
initData();
}

@override
Widget build(BuildContext context) {
return WillPopScope(
child: Container(
child: Scaffold(
backgroundColor: Colors.transparent,
appBar: _isFullScreen
? null
: AppBar(
backgroundColor: Colors.transparent,
title: const Text('SuperPlayer'),
),
body: SafeArea(
child: Builder(
builder: (context) => getBody(),
),
),
),
),
onWillPop: onWillPop);
}

Future<bool> onWillPop() async {
return !_controller.onBackPress();
}

Widget getBody() {
return Column(
children: [_getPlayArea()],
);
}

Widget _getPlayArea() {
return SuperPlayerView(_controller);
}

Widget _getListArea() {
return Container(
```

```
margin: EdgeInsets.only(top: 10),
child: ListView.builder(
itemCount: videoModels.length,
itemBuilder: (context, i) => _buildVideoItem(videoModels[i]),
),
);
}

Widget _buildVideoItem(SuperPlayerModel playModel) {
return Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
ListTile(
leading: Image.network(playModel.coverUrl),
title: new Text(
playModel.title,
style: TextStyle(color: Colors.white),
),
onTap: () => playCurrentModel(playModel)),
Divider()
],
);
}

void playCurrentModel(SuperPlayerModel model) {
_controller.playWithModel(model);
}

void initData() async {
SuperPlayerModel model = SuperPlayerModel();
model.videoURL = "http://1500005830.vod2.myqcloud.com/6c9a5118vodcq1500005830/48d0f1f93877
02299774251236/gZyqjgaZmnwA.m4v";
model.playAction = SuperPlayerModel.PLAY_ACTION_AUTO_PLAY;
model.title = "腾讯云音视频";
_controller.playWithModel(model);
}

@override
void dispose() {
// must invoke when page exit.
_controller.releasePlayer();
```

```
super.dispose();
}
}
```

深度定制开发指引

腾讯云播放器 SDK Flutter 插件对原生播放器能力进行了封装， 如果您要进行深度定制开发，建议采用如下方法：

- 基于点播播放，接口类为 TXVodPlayerController 或直播播放，接口类为 TXLivePlayerController ，进行定制开发，项目中提供了定制开发 Demo，可参考 example 工程里的 DemoTXVodPlayer 和 DemoTXLivePlayer 。
- 播放器组件 SuperPlayerController 对点播和直播进行了封装，同时提供了简单的 UI 交互， 由于此部分代码在 example 目录。如果您有对播放器组件定制化的需求，您可以进行如下操作：

  把播放器组件相关的代码，代码目录： exmple/lib/superplayer ，复制到您的项目中，进行定制化开发。

更多功能

完整功能可扫码下载视频云工具包体验，或直接运行工程 Demo。