云点播

播放器 SDK 文档

腾讯云

【版权声明】

【商标声明】

【服务声明】

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

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

- iOS
  - TXVodPlayer
  - TXVodPlayListener
  - 播放配置(Config)
    - TXPlayerGlobalSetting
    - TXVodPlayConfig
  - 视频下载(Download)
    - TXVodPreloadManagerDelegate
    - TXVodDownloadDataSource
    - TXVodDownloadMediaInfo
    - TXVodDownloadManager
    - TXVodDownloadDelegate
    - TXVodPreloadManagerDelegate
    - TXVodPreloadManager
  - 类型定义
    - TXVodSDKEventDef
    - TXPlayerSubtitleRenderModel
    - TXTrackInfo
    - TXVodDef
    - TXBitrateItem
    - TXPlayerAuthParams
    - TXImageSprite
    - TXPlayerDrmBuilder
- Android
  - TXVodPlayer
  - ITXVodPlayListener
  - ITXVodSubtitleDataListener
  - 播放配置(Config)
    - TXPlayerGlobalSetting
    - TXVodPlayConfig
  - 视频下载(Download)
    - TXVodDownloadManager
    - ITXVodDownloadListener
    - TXVodDownloadDataSource
    - TXVodDownloadMediaInfo
    - TXVodPreloadManager
    - ITXVodPreloadListener
    - ITXVodFilePreloadListener
  - 类型定义
    - TXVodConstants
    - TXPlayInfoParams
    - TXVodDef
    - TXTrackInfo

播放器 SDK 文档
概述

最近更新时间：2024-11-25 17:16:22

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播放器和第三方播放器插件，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，在正式使用播放器和第三方播放器插件前，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

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

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

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

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

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

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

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播放器，我们为您提供了播放器接入指引，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：2024-09-25 17:15:01

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 在云点播平台支持播放器和第三方播放器插件两种方式接入点播服务。

播放器

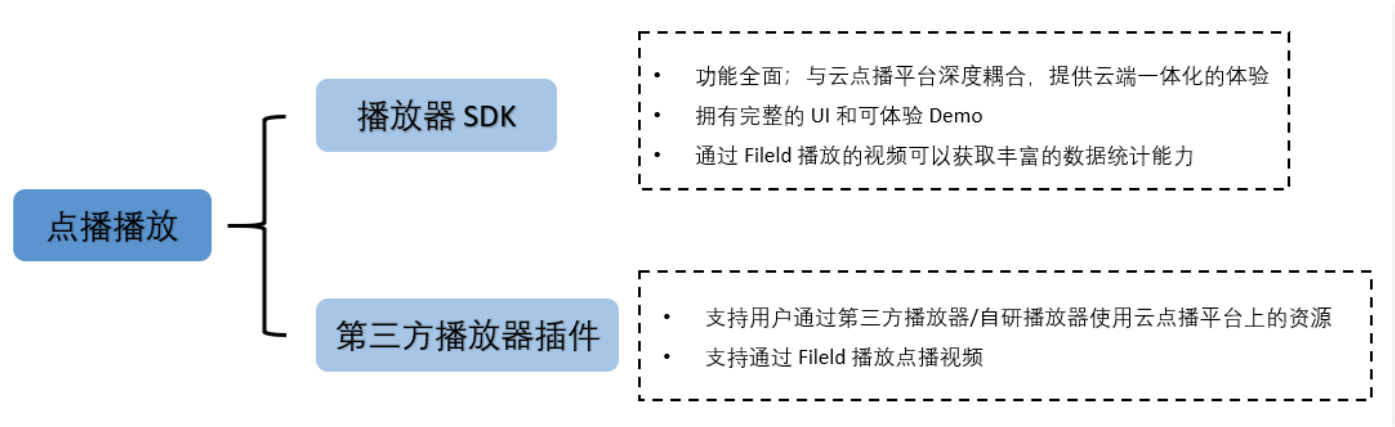
播放器是一款独立完整的视频播放器，具备视频加密、缩略图预览、清晰度切换等全面的视频播放功能；该播放器拥有完整 UI 和体验 Demo，同时深度融合腾讯云点播业务，可通过使用点播文件标识 FileID 播放云点播资源，此外，播放器还提供云点播全链路视频播放质量数据服务。如果您想要快速接入播放器，请参见 播放器教程 。

第三方播放器插件

第三方播放器插件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。将第三方播放器插件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第三方播放器插件，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，选择最适合自己的类型接入：



- **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播放器接入教程，详情请参见 播放器教程 。
- **第三方播放器插件**：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。云点播提供第三方播放器插件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	播放器	第三方播放器插件
Web 端	✓	✓
iOS 端	✓	✓

Android 端	✓	✓
Flutter 端	✓	-
UI	✓	-
Demo	✓	-

Demo 体验

最近更新时间：2024-08-26 11:17:11

腾讯云视立方 · 播放器 SDK Demo 提供完整的产品级交互界面和业务源码，开发者可按需取用。

功能体验 Demo

您可通过下述地址/二维码获得 Demo 进行功能体验。移动端扫码下载腾讯云音视频 App 后，在**视频播放**卡片中体验。

Web

功能演示 · 示例代码

立即体验

集成指引

iOS

影视剧集 · 短视频· Feed 流

Demo 源码　集成指引

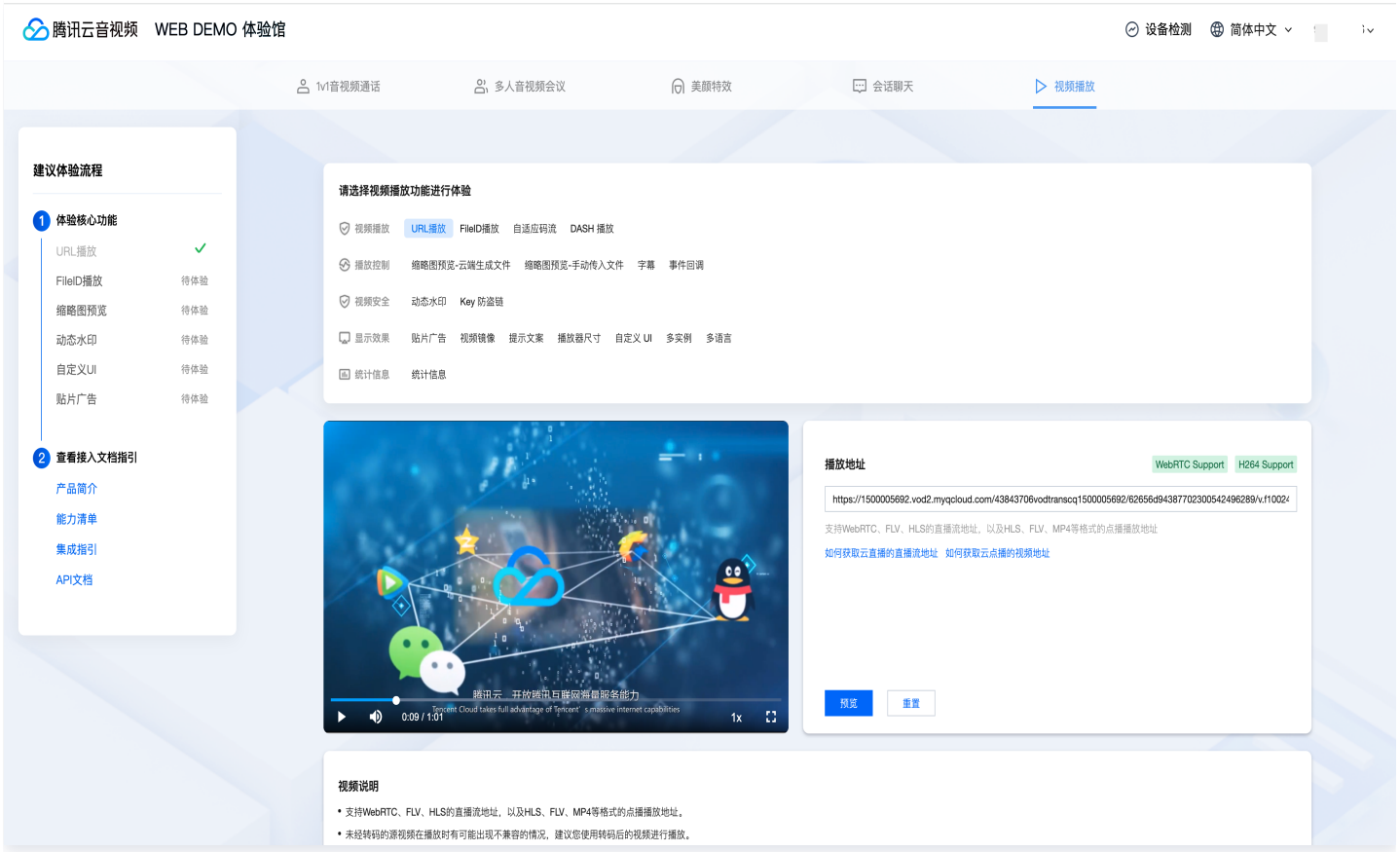
Android

影视剧集 · 短视频· Feed流

Demo 源码　集成指引

Web端（TCPlayer）

Web 端播放器支持 PC 端和移动端的浏览器视频播放，Web 播放器 Demo 提供了可对比查看视频播放功能效果及其配套代码的 Demo 体验页面，您可以通过修改示例代码，即时的在播放区域内查看修改后的功能效果。

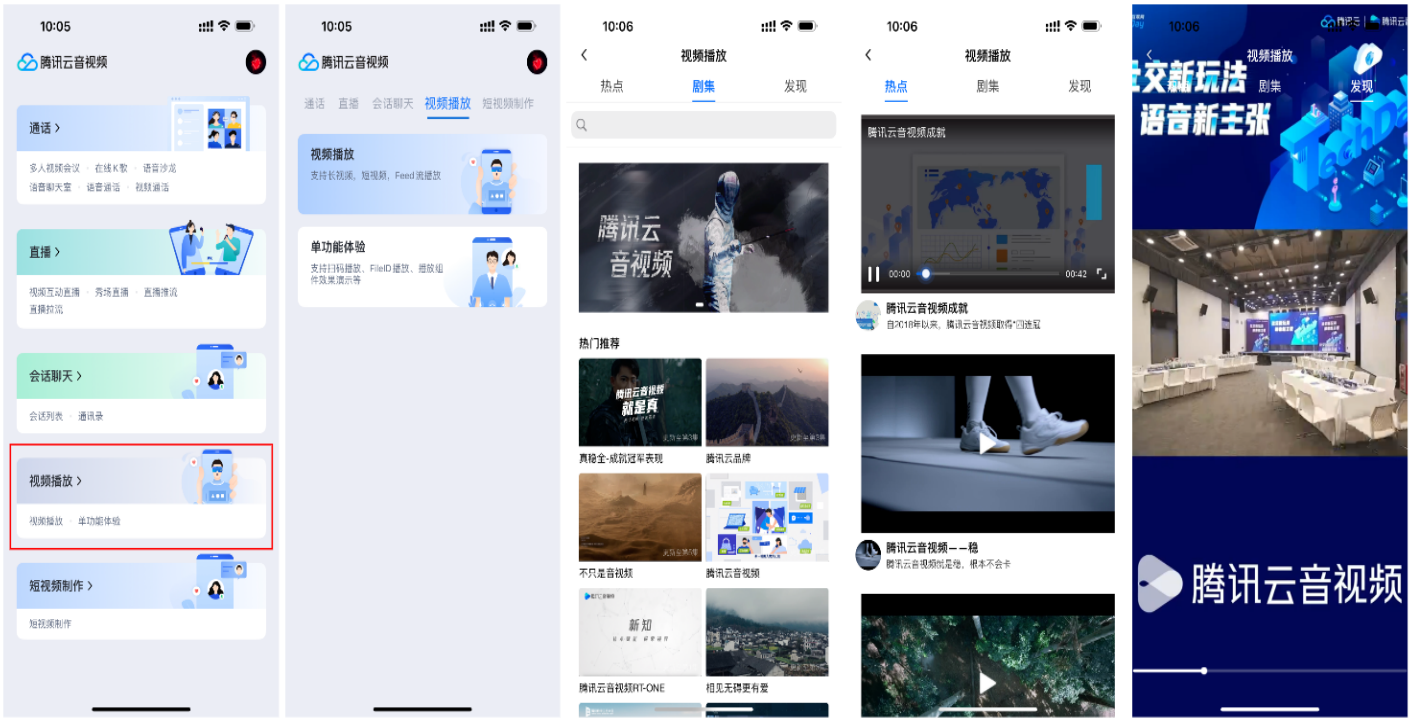


说明：
通过腾讯云账号/手机/邮箱登录后即可体验。

移动端

腾讯云音视频 App 是腾讯云音视频开发的集多款产品及功能于一身的体验方案，您可根据自身需求选择相应功能进行体验。
体验路径： 扫码 下载腾讯云音视频 App > 视频播放卡片。

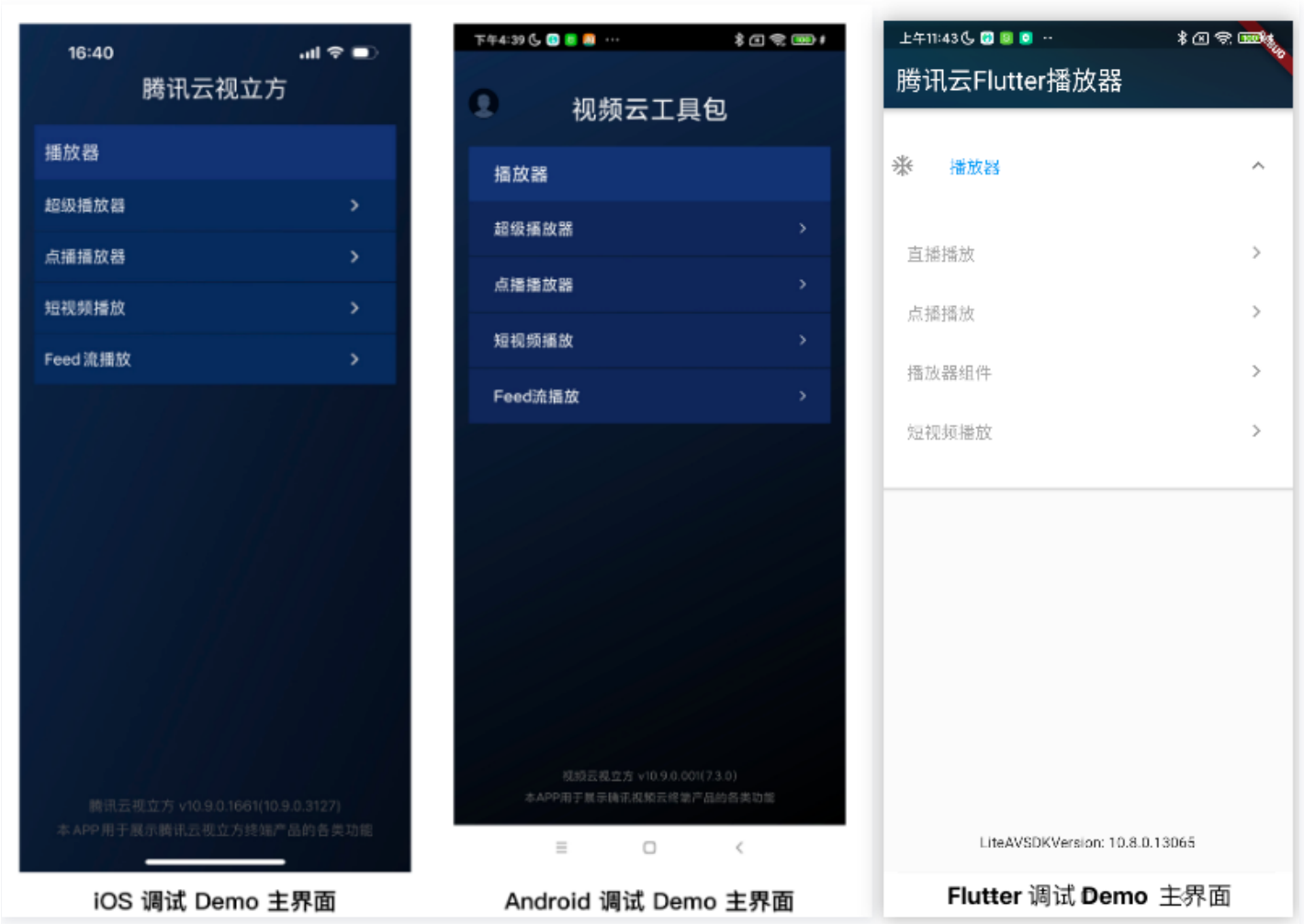
- 在 **剧集** 中，您可以体验到常见的影视剧集视频播放场景样式，类似“腾讯视频”的剧集选择及播放场景。
- 在 **热点** 中，您可以体验到类似“腾讯新闻”的 Feed 流播放场景。
- 在 **发现** 中，您可以体验到类似“腾讯微视”的沉浸式短视频播放场景。



视频播放　视频播放　影视剧集　Feed 流播放　短视频播放

开发调试 Demo

为了帮助开发者更好的理解播放器 SDK 的使用方式，播放器 SDK 移动端提供了可供开发调试的 Demo 源代码及接口使用说明，您可参考下述步骤使用。



iOS 调试 Demo 主界面　Android 调试 Demo 主界面　**Flutter 调试 Demo 主界面**

步骤一：获取 Demo 工程源码

您可访问下述 GitHub 地址获取调试 Demo 源代码，或者下载对应的 ZIP 包。

平台	源码地址	ZIP 包下载
iOS	GitHub	ZIP 包
Android	GitHub	ZIP 包
Flutter	GitHub	-

步骤二：配置 License

播放器 SDK 移动端（ iOS & Android & Flutter ）需获取 License 后方可使用。

1. 登录 腾讯云视立方控制台，在左侧菜单中选择 **License 管理** > **移动端 License**，单击**新建测试 License**。

2. 根据实际需求填写 `App Name`、`Package Name` 和 `Bundle ID`，勾选功能模块 **视频播放**，单击**确定**。

- **Package Name**：请在 App 目录下的 **build.gradle** 文件查看 **applicationId** 。
- **Bundle ID**：请在 **Xcode** 中查看项目的 **Bundle Identifier** 。

> ⚠ **注意：**
> 如果在腾讯云控制台申请 Package Name 或 Bundle ID ，和工程中实际的 Package Name 或 Bundle ID 不一致，将会播放失败。

3. 测试版 License 成功创建后，页面会显示生成的 License 信息。**在 SDK 初始化配置时需要传入 License URL 和 License Key 两个参数，请妥善保存以下信息。**

4. 获取到 License URL 和 License Key 后，请参考下面的教程把它们配置到 Demo 工程。

Android 端配置 License

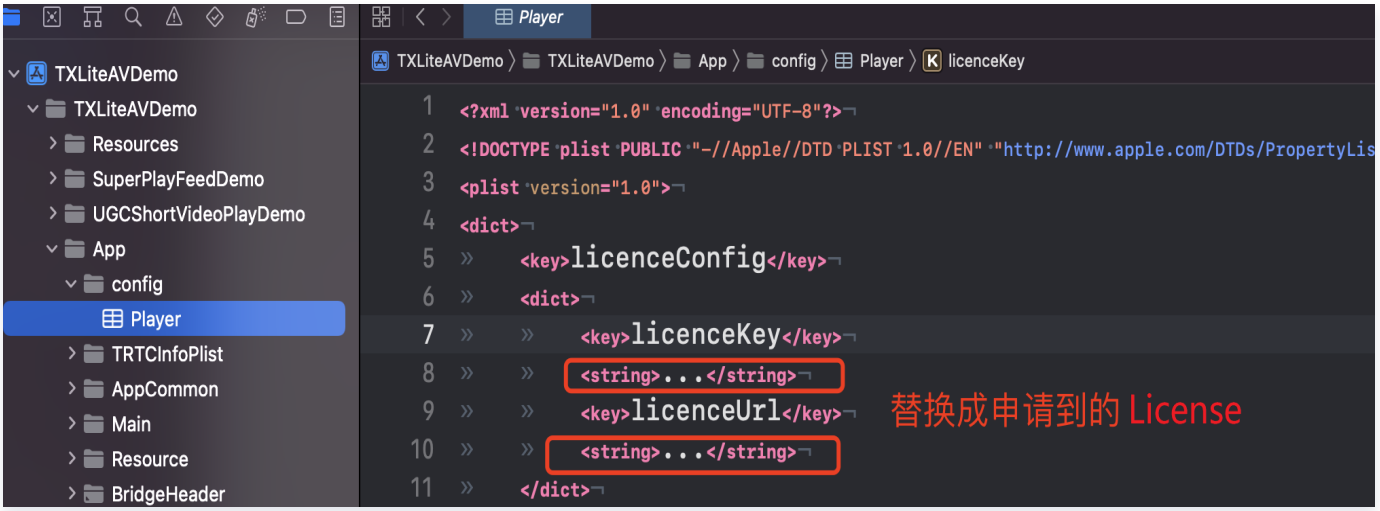
打开 Demo/app/src/main/java/com/tencent/liteav/demo/TXCSDKService.java 文件， 把 License URL 和 License Key 替换成申请到的 License 内容。

TXCSDKService.java

```
8
9   public class TXCSDKService {
10      private static final String TAG        = "TXCSDKService";
11      // 如何获取License? 请参考官网指引 https://cloud.tencent.com/document/product/454/34750
12      private static final String licenseUrl =
13              "请替换成您的licenseUrl";
14      private static final String licenseKey = "请替换成您的licenseKey";
```

iOS 端配置 License

打开 Demo/TXLiteAVDemo/App/config/Player.plist 文件， 把 License URL 和 License Key 替换成申请到的 License 内容。



Flutter 端配置 License

打开 Flutter/example/lib/main.dart 文件， 把 License URL 和 License Key 替换成申请到的 License 内容。

SDK 下载
SDK 下载

最近更新时间：2025-06-20 15:36:21

播放器 SDK 是腾讯云视立方产品家族的子产品之一，提供直播、点播场景的视频播放能力。您可以在 产品功能 中查看 SDK 支持的功能清单，在 Demo 体验 中获取各端 Demo 进行功能体验，在本页面中下载各端 SDK 并获取 Demo 源码。

播放器 SDK 下载

播放器 SDK 提供方为深圳市腾讯计算机系统有限公司，其个人信息处理规则见 腾讯云视立方 · 播放器 SDK 个人信息保护规则，当前可下载版本为 12.6，发布日志见 发布日志（iOS 和 Android），合规使用说明见 腾讯云视立方 · 播放器 SDK 合规使用指南。

Web SDK

Web 播放器 SDK

可用于直播播放和点播播放，适用于 PC 端浏览器和移动端浏览器。

ZIP 下载　集成指引　Demo 示例

iOS SDK 与 Android SDK

iOS 播放器 SDK（基础版）

包含点播播放和直播播放功能，提供常见组件及场景化的 Demo 源码帮助快速搭建应用。

ZIP 下载　集成指引　Demo 源码

Android 播放器 SDK（基础版）

包含点播播放和直播播放功能，提供常见组件及场景化的 Demo 源码帮助快速搭建应用。

ZIP 下载　集成指引　Demo 源码

iOS 播放器 SDK（高级版）

包含点播播放和直播播放功能，提供常见组件及场景化的 Demo 源码帮助快速搭建应用。

ZIP 下载　集成指引　Demo 源码

Android 播放器 SDK（高级版）

包含点播播放和直播播放功能，提供常见组件及场景化的 Demo 源码帮助快速搭建应用。

ZIP 下载　集成指引　Demo 源码

跨平台 SDK

Flutter 播放器 SDK

基于 Flutter 框架封装的播放器 SDK，让您用一套代码快速构建出能够运行于各平台的 App。

ZIP 下载　集成指引　GitHub

SDK 能力清单

功能模块	功能项	功能简介	Web	iOS & Android	Flutter
播放协议/格式	点播或直播支持	同时支持点播播放和直播播放能力。	✓	✓	✓
	支持的直播播放格式	支持 RTMP、FLV、HLS、DASH 和 WebRTC 等直播视频格式。	WebRTC、FLV、HLS、DASH	RTMP、FLV、HLS、WebRTC	RTMP、FLV、HLS、WebRTC
	支持的点播播放格式	支持 HLS、DASH、MP4 和 MP3 等点播音视频格式。	HLS、MP4、MP3、	MP4、MP3、HLS、	MP4、MP3、HLS

		FLV、DASH	DASH（DASH 仅高级版支持）	
URL 播放	支持网络视频的 URL 方式播放，URL 可以为点播播放地址也可以为直播拉流地址	✓	✓	✓
FileID 播放	支持通过云点播文件标识 FileID 播放视频，包含多个清晰度的视频、缩略图、打点等信息	✓	✓	✓
本地视频播放	支持播放存储在本地的视频文件	–	✓	✓
快直播	支持腾讯云毫秒级超低时延快直播播放。	✓	✓	✓
DASH 协议支持	支持标准协议的 DASH 视频播放	✓	✓（仅高级版支持）	×
全景 VR 视频	支持播放全景VR视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，PC端设备支持鼠标在界面上拖动画面查看	✓（仅高级版支持）	✓（仅高级版支持）	×
Quic 加速	支持 Quic 传输协议，有效提升视频传输效率。	–	✓（仅高级版支持）	✓
SDR/HDR 视频	支持播放 SDR 视频和 HDR 10/HLG 标准的 HDR 视频。	–	✓	✓
H.264 播放及软硬解	支持播放 H.264 视频源，并支持软硬解。	✓	✓	✓
H.265 硬解	支持对 H.265 视频源的硬解码播放。	–	✓	✓
纯音频播放	支持 MP3 等文件纯音频播放。	✓	✓	✓
双声道音频	支持播放双声道音频。	×	✓	✓
多音轨	支持播放含多音轨的视频文件，播放时可切换音轨，如英文切换中文。	✓	✓（仅高级版支持）	×
设置 HTTP Header	请求视频资源时，自定义 HTTP Headers 内容。	×	✓	✓
支持 HTTPS	支持播放 HTTPS 的视频资源。	✓	✓	✓
HTTP 2.0	支持 HTTP 2.0协议。	✓	✓	✓

播放性能	预下载	支持提前下载指定视频文件内容，并支持配置预下载视频文件的大小及分辨率；可大幅减少首帧耗时，另外包含对低能耗的针对性优化，性能更佳。	✓	✓	✓
	短视频播放组件	以极低的接入成本，实现极速首帧、无感启播、丝滑切换的短视频播放体验。结合预播放、预下载、播放器复用、精准流量控制、加载策略等技术，在保证低能耗的前提下实现更好的播放效果。	–	✓（仅高级版支持）	×
	边播边缓存	支持播放的同时缓存下载后面的内容，降低网络占用，可设置缓存策略。	✓	✓	✓
	精准 seek	支持在进度条上跳转到指定位置进行播放，移动端可精确到帧级别，Web 端精准到毫秒级。	✓	✓	✓
	自适应码率	支持播放 HLS、DASH 和 WebRTC 的自适应码流，可根据网络带宽自动选择合适的码率进行播放	✓	✓（DASH 仅高级版支持）	✓（不支持 DASH）
	实时下载网速	支持获取实时下载网速，既可根据业务需要给 C 端用户在卡顿时展示下载网速，也是使用自适应码率带宽预测模块的必要前提。	✓	✓	✓
	多实例	支持在一个界面添加多个播放器同时播放。	✓	✓	✓
	动态追帧	发生卡顿时，通过类似“快进”的方式追赶上当前直播进度，保证直播画面实时性。	✓	×	×
播放控制	基础控制	支持开始、结束、暂停和恢复等播放控制功能	✓	✓	✓
	基础画中画	支持切换到画中画以小窗形式播放，移动端同时支持在集成APP内或APP外的画中画播放。	✓	✓	✓
	高级画中画组件	相对基础画中画，新增支持加密视频画中画、离线播放画中画和“秒切”效果。	–	✓（仅高级版支持）	×
	缓存内 seek	支持已经缓存的视频内容在 seek 时不清除缓存内容并快速 seek	✓	✓	✓
	直播时移	支持直播时移视频流播放，可设置开始、结束和当前支持时间，支持拖动	✓	×	×
	进度条标记及	支持在进度条上添加标记信息，并支持	✓	✓	×

	缩略图预览	缩略图（雪碧图）预览			
	设置封面	支持设置播放视频的封面	✓	✓	✓
	重播、循环播放、列表播放	支持视频播放结束后自动或手动重播；同时支持依次播放视频列表中的视频，并支持轮播，即在视频列表最后一个视频播放完成后，播放列表的第一个视频。	✓	✓	✓
	断点续播	支持从上次播放结束位置开始播放	✓	✓	✓
	自定义启播时间	支持自定义视频开启播放的时间	✓	✓	✓
	倍速播放	支持0.5~3倍的变速播放，音频实现变速不变调	✓	✓	✓
	后台播放	支持界面切到后台后继续播放音频和视频	–	✓	✓
	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	✓	✓	✓
	播放失败重试	播放失败时自动重试，支持直播的自动重连功能	✓	✓	✓
	音量设置	支持实时调节系统音量和静音操作	✓	✓	✓
	清晰度切换和命名	支持用户流畅无卡顿的切换 HLS 视频的多路清晰度流，并支持为不同清晰度流自定义命名	✓	✓	✓
	截图功能	支持截取播放画面的任意一帧	–	✓	×
	试看功能	支持播放开启试看功能的视频	✓	✓	×
	弹幕	支持在视频上方展示弹幕	✓	✓	×
	外挂字幕	支持导入自定义字幕文件，Web 端支持 WebVTT 格式，移动端支持 VTT、SRT 格式	✓	✓（仅高级版支持）	x
视频安全	referer 黑白名单	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。	✓	✓	✓

	HLS 私有加密	支持在云点播的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，可有效防范多种浏览器插件和灰产工具的破解。	✓	✓	✓
	商业 DRM	提供苹果 Fairplay、谷歌 Widevine 原生加密方案。	✓	✓（仅高级版支持）	×
	安全下载	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。	–	✓	✓
	动态水印	支持在播放界面添加不规则跑动的文字水印，有效防盗录。	✓	✓	×
	溯源水印	支持以极低的成本实现对视频盗录者的追溯。	✓	✓	✓
	幽灵水印	随机时间、随机位置、短暂的出现在播放界面上，同时一旦检测到水印被异常去除，将自动停止视频播放；可在几乎不影响观看体验的同时保证视频安全。	✓	×	×
	Web 安全插件	检测Web端播放环境和播放状态是否正常，异常环境下将中断视频播放，保护视频安全。插件包含MSE环境检测、安全结构检查和接口响应完整性校验。	✓（仅高级版）	–	–
显示效果	自定义 UI	SDK 提供包含 UI 集成方案，提供包含 UI 界面的常用播放组件，可以根据自身需求选用。	✓	✓	✓
	屏幕填充	支持为视频画面选择不同填充模式，适配屏幕大小。	✓	✓	×
	设置播放器尺寸	支持自定义设置播放器的宽高。	✓	✓	✓
	图片贴片	支持暂停时，增加图片贴片用于广告展示。	✓	✓	×
	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。	×	✓	×
	锁定屏幕	支持锁屏功能，包含锁定旋转和隐藏界面元素。	–	✓	×
	亮度调节	支持播放视频时调节系统亮度。	–	✓	✓
增值功能	终端极速高清	基于超分辨率技术提供终端的极速高清方案，针对在线视频在播放端进行实时	×	✓	×

		后处理超分，在保证画质的前提下，帮助节省带宽；或在相同带宽条件下，提升视频播放清晰度与主观质量，详情参见 终端极速高清 。			
	播放质量监控	基于播放数据上报，结合云点播和云直播服务，提供播放全链路的数据统计、质量监控及可视化分析服务，点播参见 点播播放数据，直播参见 SDK 质量监控 。	✓	✓	✓

注意：
表中“–”表示该端无需具备相应功能或不存在相关概念。
未注明“仅高级版支持”的“✓”代表基础版即支持。

安装包增量

平台	基础版安装包增量	高级版安装包增量
Android	• arm64：4.1 M • armv7：3.9 M • dex：573 KB	• arm64：4.4 M • armv7：4.2 M • dex：573 KB
iOS	arm64：5M	arm64：5.3M

功能说明

最近更新时间：2023-09-26 09:44:12

播放器 SDK 提供直播、点播场景的视频播放能力，支持 Web/H5、iOS、Android、Flutter 等平台，支持的功能详情如下：

功能模块	功能项	功能简介	Web	iOS & Android	Flutter
播放协议/格式	点播或直播支持	同时支持点播播放和直播播放能力	✓	✓	✓
	支持的直播播放格式	支持 RTMP、FLV、HLS、DASH 和 WebRTC 等直播视频格式	WebRTC、FLV、HLS、DASH	RTMP、FLV、HLS、WebRTC	RTMP、FLV、HLS、WebRTC
	支持的点播播放格式	支持 HLS、DASH、MP4 和 MP3 等点播音视频格式	HLS、MP4、MP3、FLV、DASH	MP4、MP3、HLS、DASH（DASH 仅高级版支持）	MP4、MP3、HLS
	URL 播放	支持网络视频的 URL 方式播放，URL 可以为点播播放地址也可以为直播拉流地址	✓	✓	✓
	FileID 播放	支持通过云点播文件标识 FileID 播放视频，包含多个清晰度的视频、缩略图、打点等信息	✓	✓	✓
	本地视频播放	支持播放存储在本地的视频文件	-	✓	✓
	快直播	支持腾讯云毫秒级超低时延快直播播放	✓	✓	✓
	DASH 协议支持	支持标准协议的 DASH 视频播放	✓	✓（仅高级版支持）	×
	全景VR视频	支持播放全景VR视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，PC端设备支持鼠标在界面上拖动画面查看	✓（仅高级版支持）	✓（仅高级版支持）	×
	Quic 加速	支持 Quic 传输协议，有效提升视频传输效率	-	✓（仅高级版支持）	✓
	SDR/HDR 视频	支持播放 SDR 视频和 HDR 10/HLG 标准的 HDR 视频	-	✓	✓

	H.264播放及软硬解	支持播放H.264视频源，并支持软硬解	✓	✓	✓
	H.265 硬解	支持对 H.265 视频源的硬解码播放	–	✓	✓
	AV1	支持播放 AV1 编码格式的视频	部分支持	部分支持（仅高级版支持）	部分支持
	纯音频播放	支持 MP3 等文件纯音频播放	✓	✓	✓
	双声道音频	支持播放双声道音频	×	✓	✓
	多音轨	支持播放含多音轨的视频文件，播放时可切换音轨，如英文切换中文。	✓	✓（仅高级版支持）	×
	设置 Http Header	请求视频资源时，自定义 HTTP Headers 内容	×	✓	✓
	支持 HTTPS	支持播放 HTTPS 的视频资源	✓	✓	✓
	HTTP 2.0	支持 HTTP 2.0协议	✓	✓	✓
播放性能	短视频播放组件	以极低的接入成本，实现极速首帧、无感启播、丝滑切换的短视频播放体验。结合预播放、预下载、播放器复用、精准流量控制、加载策略等技术，在保证低能耗的前提下实现极致流畅的播放效果。	–	✓（仅高级版支持）	×
	预下载	支持提前下载指定视频文件内容，并支持配置预下载视频文件的大小及分辨率。	✓	✓	✓
	边播边缓存	支持播放的同时缓存下载后面的内容，降低网络占用，可设置缓存策略	✓	✓	✓
	精准 seek	支持在进度条上跳转到指定位置进行播放，移动端可精确到帧级别，Web 端精准到毫秒级	✓	✓	✓
	自适应码率	支持播放 HLS、DASH 和 WebRTC 的自适应码流，可根据网络带宽自动选择合适的码率进行播放	✓	✓（DASH 仅高级版支持）	✓（不支持 DASH）
	实时下载网速	支持获取实时下载网速，既可根据业务需要给 C 端用户在卡顿时展示下载网速，也是使用自适应码率带宽预测模块的必要前提	✓	✓	✓
	多实例	支持在一个界面添加多个播放器同时播放	✓	✓	✓

	动态追帧	发生卡顿时，通过类似“快进”的方式追赶上当前直播进度，保证直播画面实时性	✓	×	×
播放控制	基础控制	支持开始、结束、暂停和恢复等播放控制功能	✓	✓	✓
	基础画中画	支持切换到画中画以小窗形式播放，移动端同时支持在集成 APP 内或 APP 外的画中画播放。	✓	✓	✓
	高级画中画组件	相对基础画中画，新增支持加密视频画中画、离线播放画中画和“秒切”效果。	–	✓（仅高级版支持）	×
	缓存内 seek	支持已经缓存的视频内容在 seek 时不清除缓存内容并快速 seek	✓	✓	✓
	直播时移	支持直播时移视频流播放，可设置开始、结束和当前支持时间，支持拖动	✓	×	×
	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	✓	✓	✓
	设置封面	支持设置播放视频的封面	✓	✓	✓
	重播、循环播放、列表播放	支持视频播放结束后自动或手动重播；同时支持依次播放视频列表中的视频，并支持轮播，即在视频列表最后一个视频播放完成后，播放列表的第一个视频。	✓	✓	✓
	断点续播	支持从上次播放结束位置开始播放	✓	✓	✓
	自定义启播时间	支持自定义视频开启播放的时间	✓	✓	✓
	倍速播放	支持0.5~3倍的变速播放，音频实现变速不变调	✓	✓	✓
	后台播放	支持界面切到后台后继续播放音频和视频	–	✓	✓
	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	✓	✓	✓
	播放失败重试	播放失败时自动重试，支持直播的自动重连功能	✓	✓	✓
	音量设置	支持实时调节系统音量和静音操作	✓	✓	✓
	清晰度切换和命名	支持用户流畅无卡顿的切换 HLS 视频的多路清晰度流，并支持为不同清晰度流自定义命名	✓	✓	✓

	截图功能	支持截取播放画面的任意一帧	–	✓	×
	试看功能	支持播放开启试看功能的视频	✓	✓	×
	弹幕	支持在视频上方展示弹幕	✓	✓	×
	外挂字幕	支持导入自定义字幕文件，Web 端支持WebVTT 格式，移动端支持 VTT、SRT 格式	✓	✓（仅高级版支持）	x
视频安全	referer 黑白名单	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	✓	✓	✓
	HLS 私有加密	支持在云点播的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，可有效防范多种浏览器插件和灰产工具的破解	✓	✓	✓
	商业 DRM	提供苹果 Fairplay、谷歌 Widevine 原生加密方案	✓	✓（仅高级版支持）	×
	安全下载	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	–	✓	✓
	动态水印	支持在播放界面添加不规则跑动的文字水印，有效防盗录	✓	✓	×
	溯源水印	支持以极低的成本实现对视频盗录者的追溯	✓	✓	✓
	幽灵水印	随机时间、随机位置、短暂的出现在播放界面上，同时一旦检测到水印被异常去除，将自动停止视频播放；可在几乎不影响观看体验的同时保证视频安全。	✓	×	×
	Web 安全插件	检测 Web 端播放环境和播放状态是否正常，异常环境下将中断视频播放，保护视频安全。插件包含 MSE 环境检测、安全结构检查和接口响应完整性校验。	✓（仅高级版）	–	–

显示效果	自定义 UI	SDK 提供含 UI 集成方案，提供包含 UI 界面的常用播放组件，可以根据自身需求选用	✓	✓	✓
	屏幕填充	支持为视频画面选择不同填充模式，适应屏幕大小	✓	✓	×
	设置播放器尺寸	支持自定义设置播放器的宽高	✓	✓	✓
	图片贴片	支持暂停时，增加图片贴片用于广告展示	✓	✓	×
	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	×	✓	×
	锁定屏幕	支持锁屏功能，包含锁定旋转和隐藏界面元素	–	✓	×
	亮度调节	支持播放视频时调节系统亮度	–	✓	✓
增值功能	终端极速高清	基于超分辨率技术提供终端的极速高清方案，针对在线视频在播放端进行实时后处理超分，在保证画质的前提下，帮助节省带宽；或在相同带宽条件下，提升视频播放清晰度与主观质量，详情参见 终端极速高清	×	✓	×
	播放质量监控	基于播放数据上报，结合云点播和云直播服务，提供播放全链路的数据统计、质量监控及可视化分析服务，点播参见 点播播放数据，直播参见 SDK 质量监控	✓	✓	✓

注意：

表中“–”表示该端无需具备相应功能或不存在相关概念。

未注明“仅高级版支持”的“✓”代表基础版即支持。

播放器教程
阶段1：播放原始视频

最近更新时间：2023-07-07 17:10:01

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，并在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。

步骤1：上传视频

本步骤，我们将指导您如何上传视频。

1. 登录**云点播控制台** > 应用管理，单击目标应用名称进入**媒资管理** > **音视频管理**页面，单击**上传音视频**。

2. 在上传界面，选择“**本地上传**”，并单击**选择文件** 上传本地视频，其他设置如下：

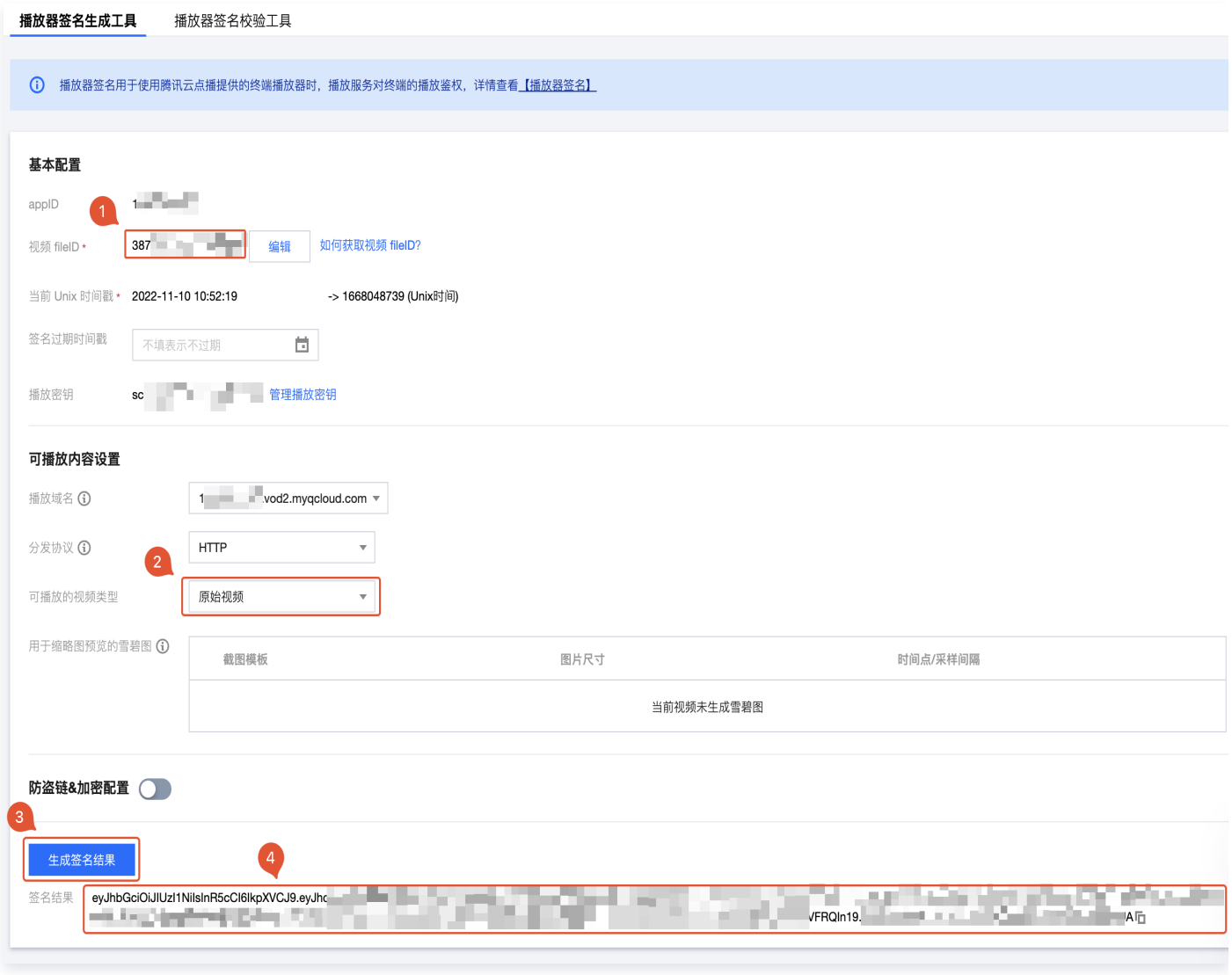
视频处理选择“**只上传，暂不进行视频处理**”。

3. 单击**开始上传**，进入“正在上传”页，当**状态**变为“**上传成功**”即表示上传完成，**文件 ID** 即为上传音视频的 FileId（这里为 387xxxxx8142975036）。

步骤2：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

1. 在应用管理页，选择菜单栏的**分发播放设置** > **播放器签名工具**，填写如下信息：
   - **视频 fileId** 填写 步骤1 的 FileId（387xxxxx8142975036）。
   - **签名过期时间戳**播放器签名过期时间，不填表示签名不过期。
   - **可播放的视频类型**选择“**原始视频**”。
2. 单击**生成签名结果**，得到签名结果字符串。

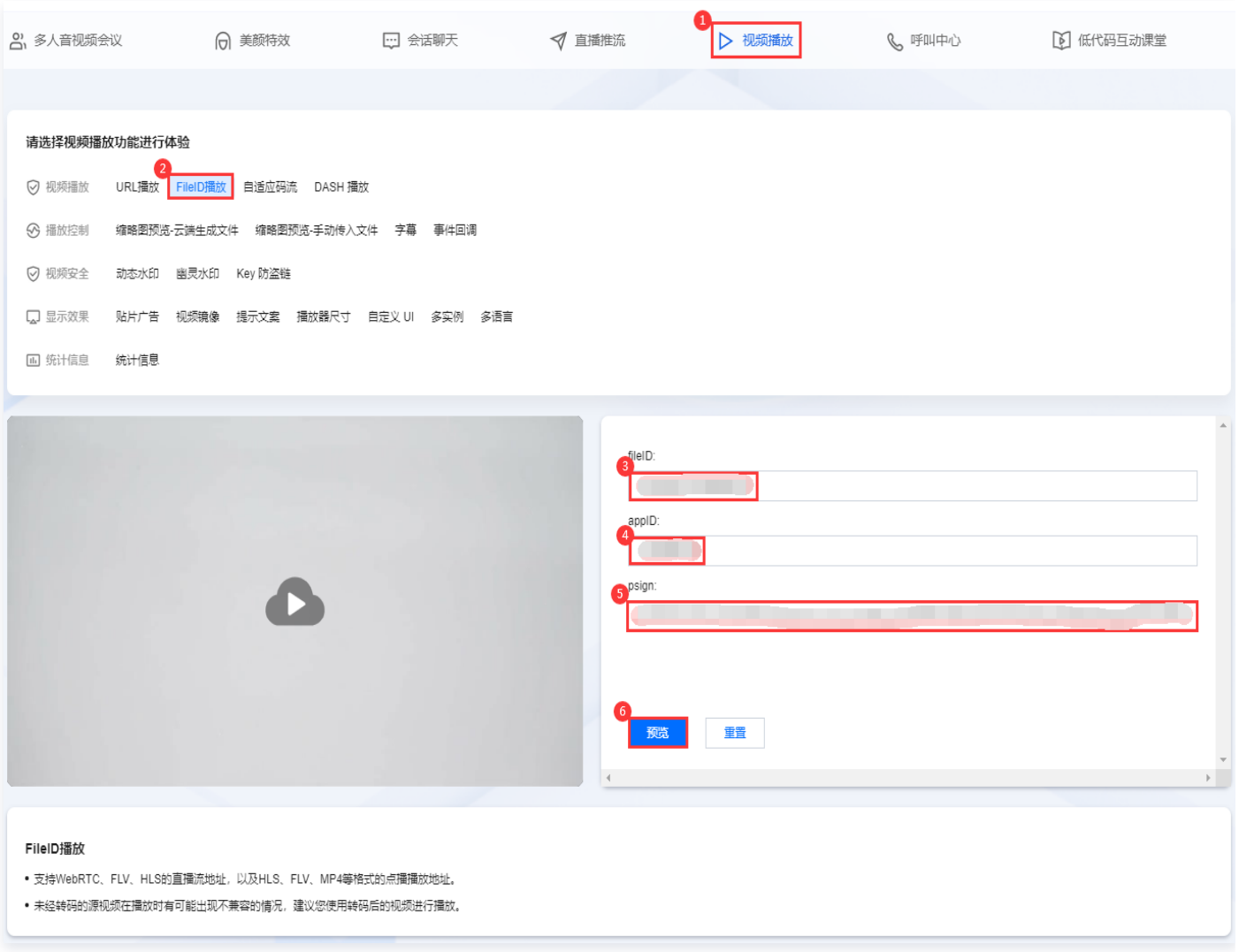


步骤3：播放视频

经过步骤2，我们得到播放视频所需的三个参数：`appId`、`fileId`以及播放器签名（`psign`），下面将展示 Web 端播放视频。

Web 端播放示例

1. 打开 Web端播放器体验，配置如下：
   - **播放器功能**选择“**视频播放**”。
   - 单击 **FileID 播放**标签页。
   - **fileID** 填写上一步的 FileId（387xxxxx8142975036）。
   - **appID** 填写文件所属的 appId（即上一步生成播放器签名页面的 appID）。
   - **psign** 填写上一步生成的签名结果字符串。
2. 单击**预览**即可播放视频。



多端播放器 Demo

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

总结

学习本教程后，您已经掌握如何上传一个视频到云点播，并在播放器中播放。
如果您希望：

- 播放转码视频，请参考 阶段2：播放转码视频 。
- 播放自适应码流视频，请参考 阶段3：播放自适应码流视频 。
- 播放加密视频，请参考 阶段4：播放加密视频 。
- 播放多分辨率的加密视频，请参考 阶段5：播放多分辨率的加密视频 。
- 播放长视频方案，请参考 阶段6：播放长视频方案 。

阶段2：播放转码视频

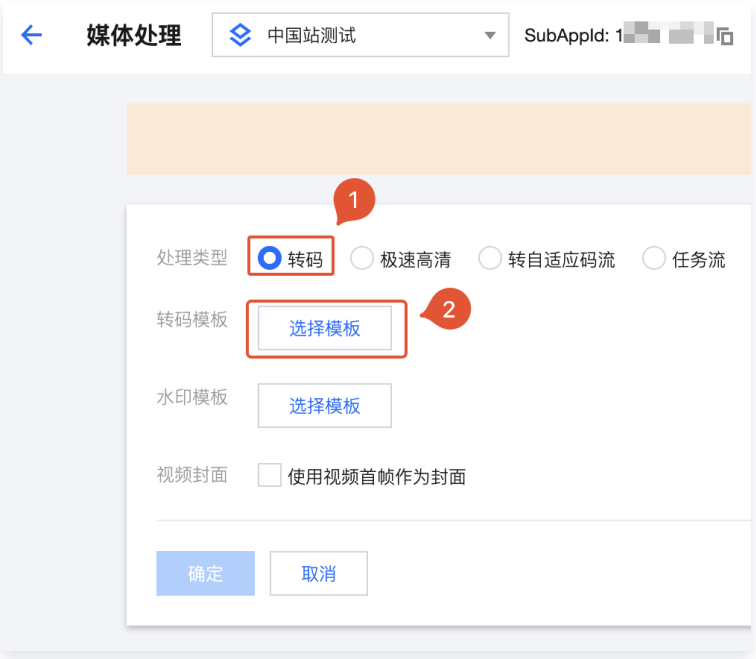
最近更新时间：2024-09-25 17:15:01

学习目标

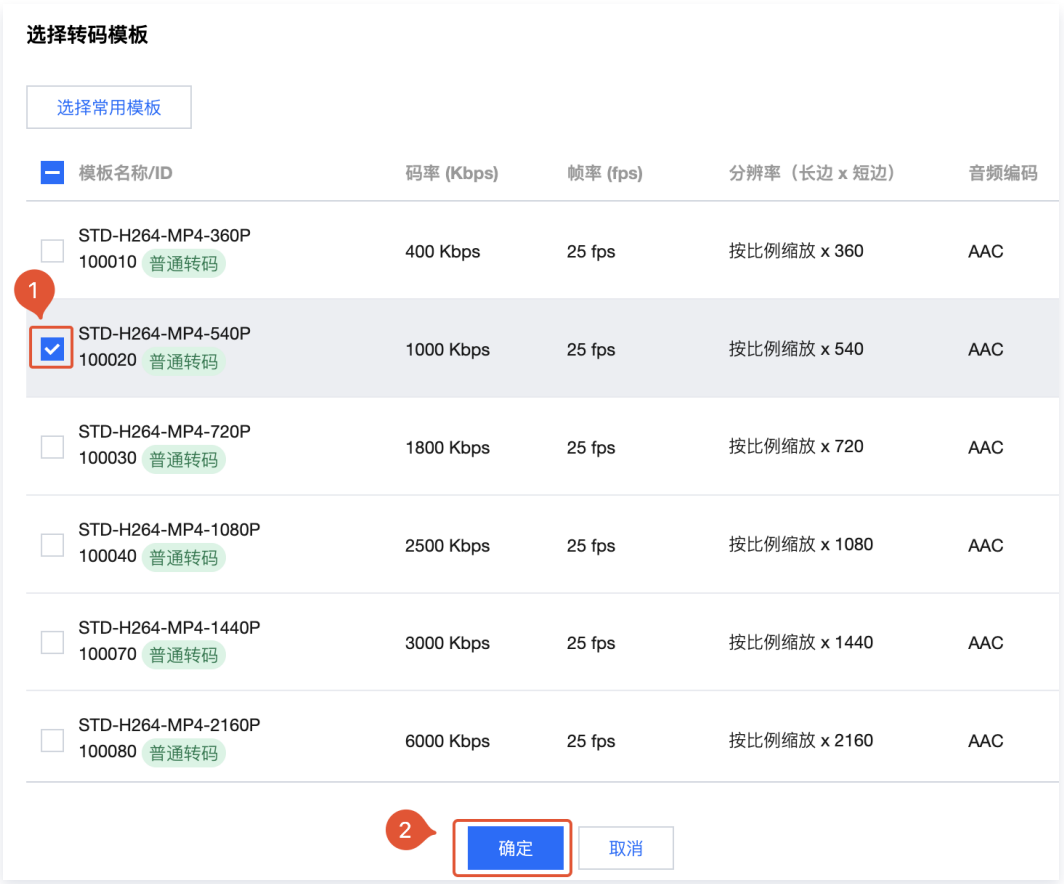
学习本阶段教程，您将了解并掌握如何对视频转码，并使用播放器播放转码视频。
阅读之前，请先确保已经学习播放器指引的 阶段1：播放原始视频 篇部分，本教程使用了阶段1开通的账号以及上传的视频。

步骤1：视频转码

1. 登录**云点播控制台** > **应用管理**，单击目标应用名称后默认进入**媒资管理** > **音视频管理**页面，勾选要处理的视频（FileId 为 387xxxxx8142975036），单击 **转码**。
2. 在**媒体处理**界面：
   2.1 **处理类型**选择“**转码**”。
   2.2 **转码模板**单击**选择模板**：



2.3 在弹出的**选择转码模板**页面选择模板（模板名称为 `STD-H264-MP4-540P`，ID 为 `100020`）：



2.4 单击**确定**发起转码任务。

3. 进入左侧导航栏**任务中心**，列表中的“任务状态”从“处理中”变为“已完成”，表示视频已处理完毕。

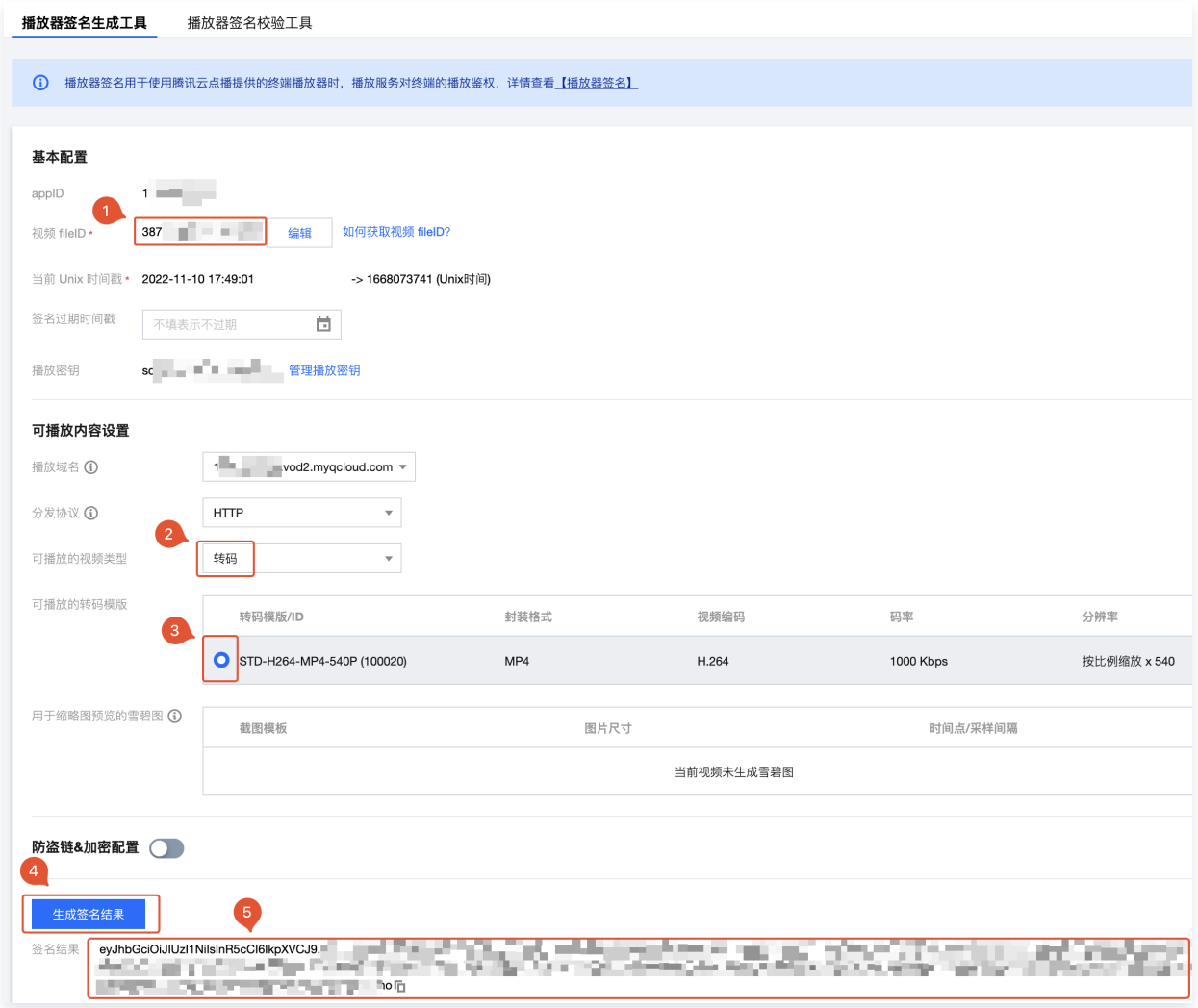
4. 进入**媒资管理** > **音视频管理**，单击发起转码视频条目右侧的**管理**，进入管理页面。

转码信息页面可以看到转码成功的转码模板列表。

步骤2：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

1. 选择**分发播放设置** > **播放器签名工具**，填写如下信息：
   - **视频 fileId** 填写 步骤1 的 FileId（387xxxxx8142975036）。
   - **签名过期时间戳**播放器签名过期时间，不填表示签名不过期。
   - **可播放的视频类型**选择“**转码**”。
   - **可播放的转码模板 ID** 选择 **100020(STD-H264-MP4-540P)**。
2. 单击**生成签名结果**，得到签名结果字符串。



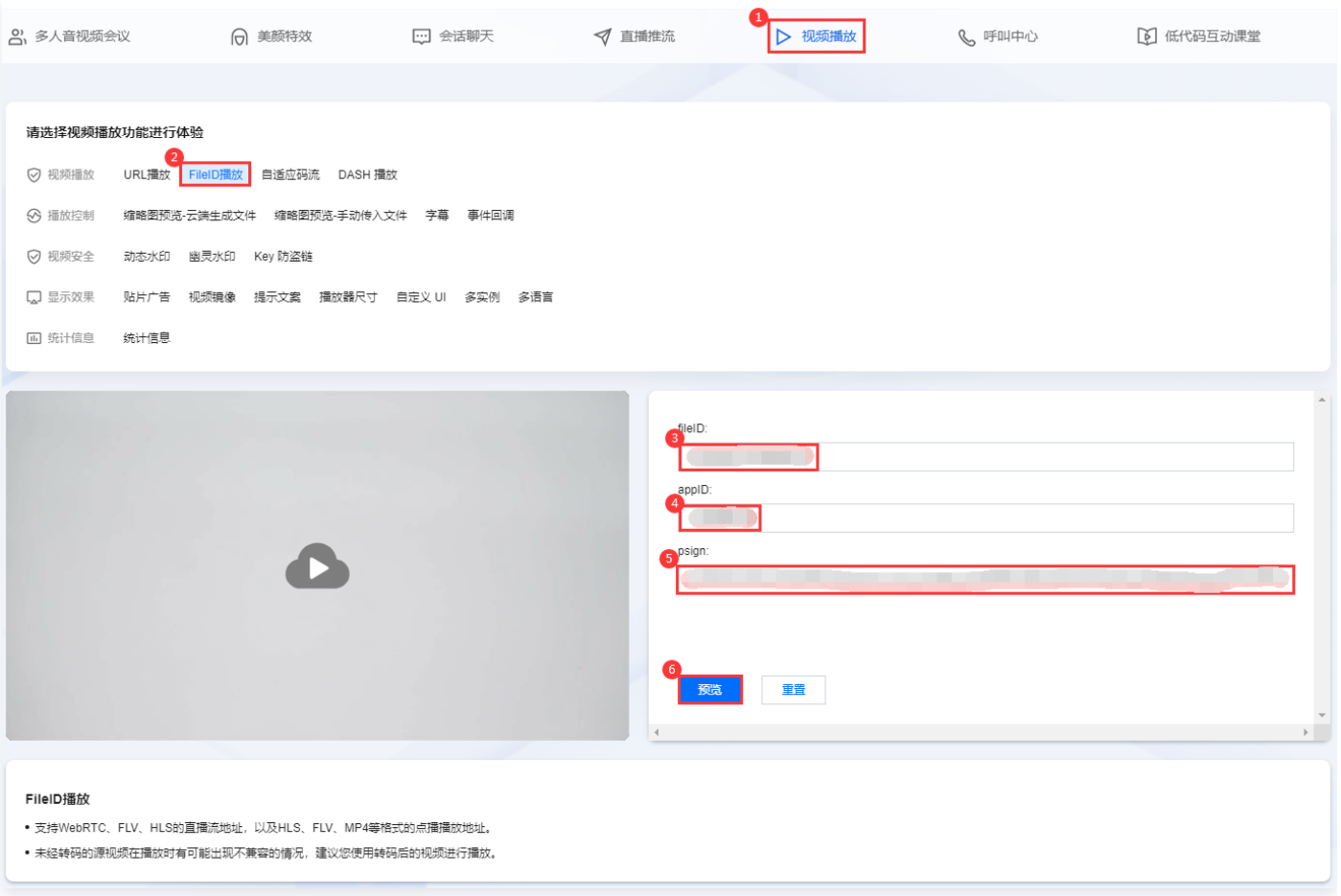
步骤3：播放视频

经过步骤2，我们得到播放视频所需的三个参数：`appId`、`fileId`以及播放器签名（`psign`），下面将展示 Web 端播放视频。

Web 端播放示例

1. 打开 Web 端播放器体验，配置如下：
   - **播放器功能**选择“**视频播放**”。
   - 单击 **FileID 播放** 标签页。
   - **fileID** 填写上一步的 FileId（387xxxxx8142975036）。
   - **appID** 填写文件所属的 appId（即上一步生成播放器签名页面的 appID）。
   - **psign** 填写上一步生成的签名结果字符串。

2. 单击**预览**即可播放视频。



多端播放器 Demo

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

总结

学习本教程后，您已经掌握如何对视频转码，并在播放器中播放。

阶段3：播放自适应码流视频

最近更新时间：2024-09-25 17:15:01

学习目标

学习本阶段教程，您将了解如何播放自适应码流视频，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，请先确保已经学习播放器指引的 阶段1：播放原始视频 篇部分，本教程使用了阶段1开通的账号以及上传的视频。

步骤1：创建自适应码流模板

1. 登录**云点播控制台** > 应用管理，单击目标应用名称进入应用管理页，选择**媒体处理设置** > **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。
     - 打包格式：选择 “HLS”。
     - 加密类型：选择“不加密”。
     - 低分辨率转高分辨率：不开启。
     - 转码方式：选择“普通转码”。

- **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 `1429212` 。

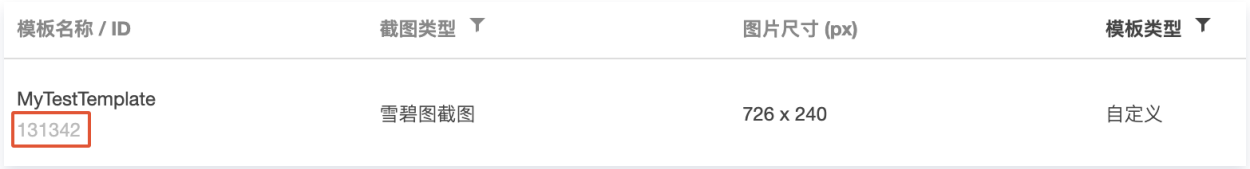
模板名称 / ID	打包类型	加密类型	子流数	低分辨率...
MyTestTemplate 1429212	HLS	不加密	3条子流	禁止

步骤2：创建雪碧图模板

1. 在**媒体处理设置 > 模板设置**页，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。
   - 截图类型：选择“雪碧图截图”。
   - 小图尺寸：726px × 240px。
   - 采样间隔：20%。
   - 小图行数：10。
   - 小图列数：10。

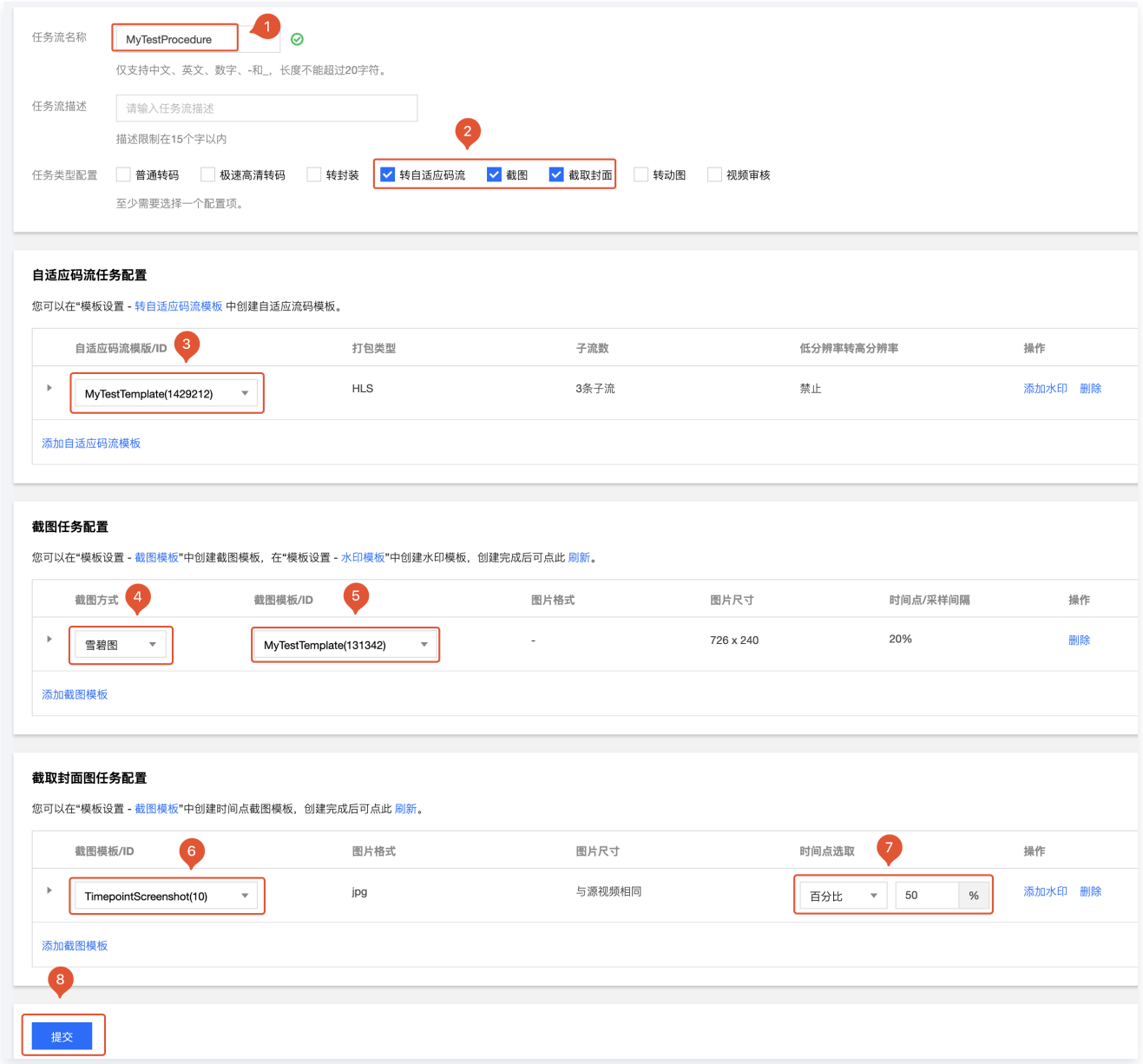
3. 单击**创建**，则生成了一个模板 ID 为 131342 的雪碧图模板。



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 1429212）和雪碧图模板（ID 为 131342）后，还需要创建一个新的任务流。

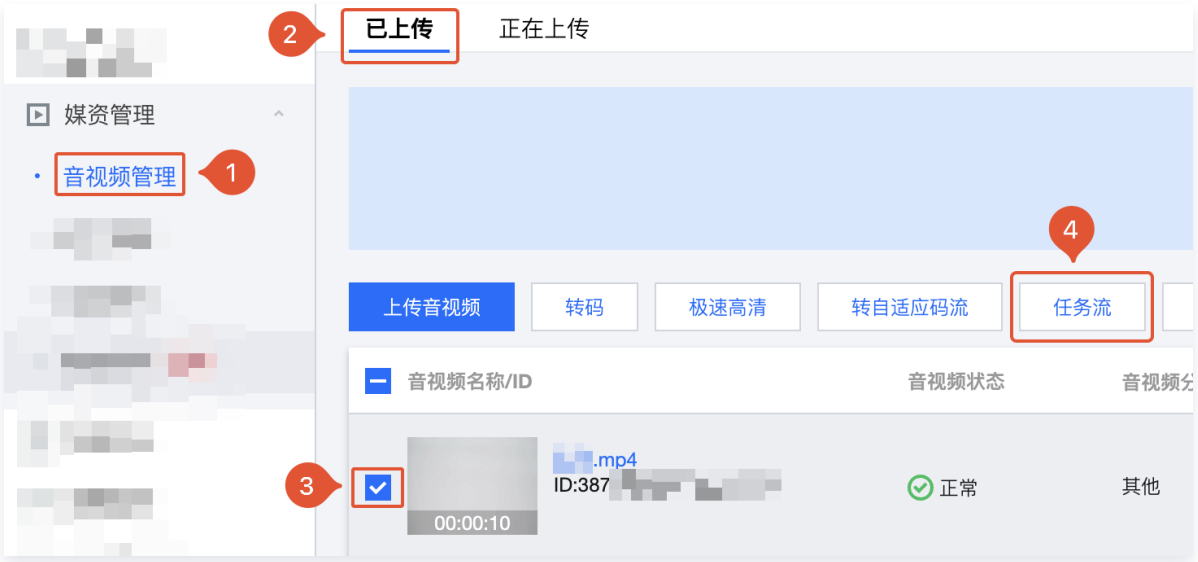
1. 在**媒体处理设置** > **任务流设置**页，单击**创建任务流**：
   - 任务流名称：填写 MyTestProcedure。
   - 任务类型配置：勾选“转自适应码流”、“截图”和“截取封面”：
   - 在**自适应码流任务配置**选项卡，单击**添加自适应码流模板**，在“自适应码流模板/ID”栏选择**步骤1**创建的自定义转自适应码流模板 ID 1429212 (MyTestTemplate)。
   - 在**截图任务配置**选项卡，单击**添加截图模板**，“截图方式”栏选择“雪碧图”，“截图模板”栏选择**步骤2**创建的自定义雪碧图模板 ID 131342 (MyTestTemplate)。
   - 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。

任务流名称	任务流类型	创建时间	最后修改时间
MyTestProcedure	自定义	2022-11-10 21:03:38	2022-11-10 21:03:38

3. 在控制台选择**媒资管理 > 音视频管理**，勾选要处理的视频（FileId 为 387xxxxx8142975036），单击任务流。



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**，等待列表中的“任务状态”从“处理中”变为“已完成”，表示视频已处理完毕。

6. 进入**媒资管理 > 音视频管理**，单击发起转自适应码流的视频条目右侧**管理**，进入管理页面。

   6.1 在**基本信息**模块可以查看：

   可以看到生成的封面，以及自适应码流输出（模板 ID 为 1429212）。

6.2 选择**截图信息**页签：

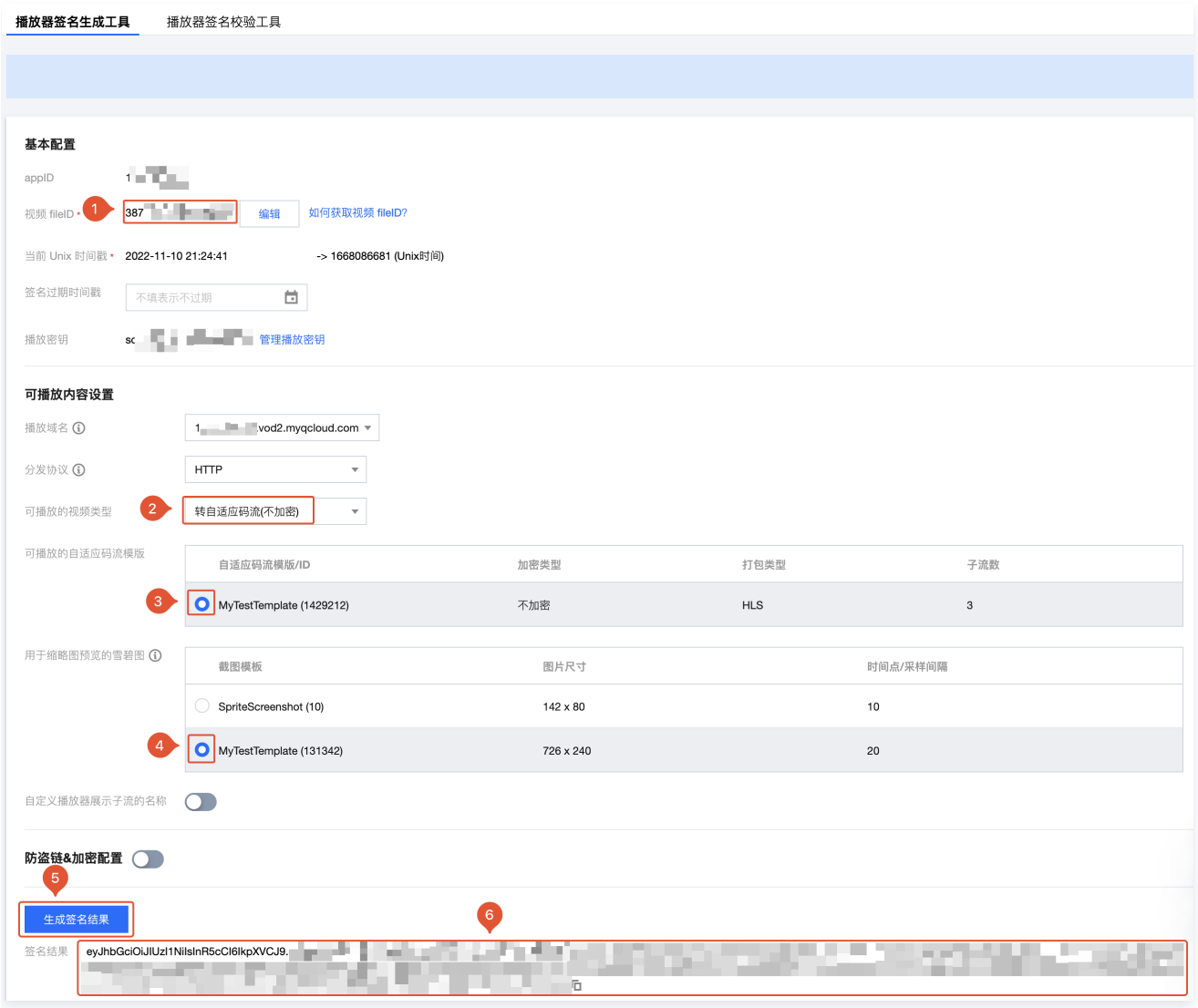
可以看到生成的雪碧图（模板 ID 为 131342）。

步骤4：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

选择 **分发播放设置** > **播放器签名工具**，填写如下信息：

- **视频 fileId** 填写**步骤3**的 FileId（387xxxxx8142975036）。
- **签名过期时间戳**播放器签名过期时间，不填表示签名不过期。
- **可播放的视频类型**选择 **转自适应码流(不加密)**。
- **可播放的自适应码流模板 ID** 选择 **1429212（MyTestTemplate）**。
- **用于缩略图预览的雪碧图 ID** 选择 **131342（MyTestTemplate)**。
- **单击生成签名结果**，得到签名结果字符串。

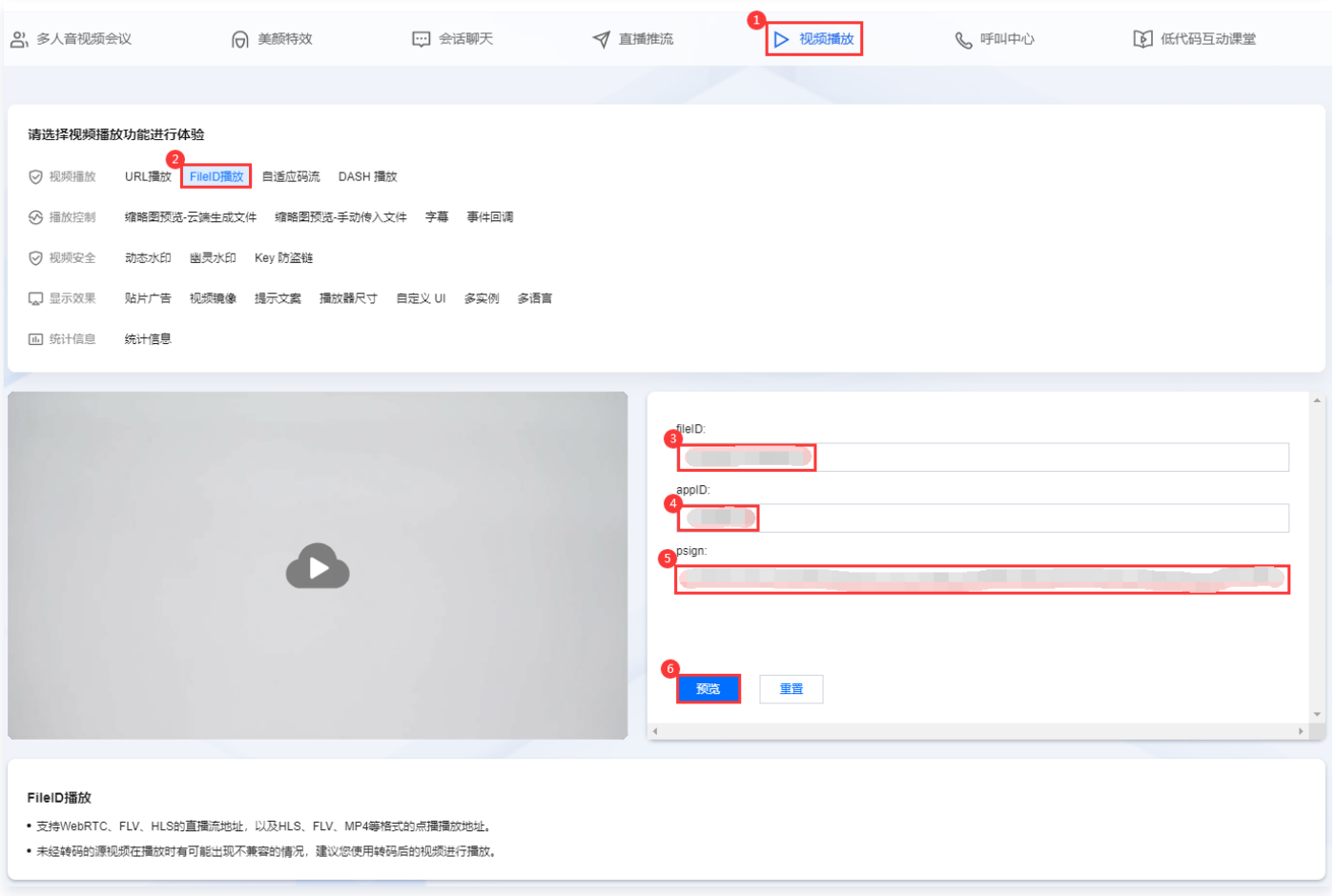


步骤5：播放视频

经过步骤4，我们得到播放视频所需的三个参数：`appId`、`fileId` 以及播放器签名（`psign`），下面将展示 Web 端播放视频。

Web 端播放示例

1. 打开 Web 端播放器体验，配置如下：
   - **播放器功能**选择**视频播放**
   - 单击 **FileID 播放** 标签页
   - **fileID** 填写上一步的 FileId（387xxxxx8142975036）
   - **appID** 填写文件所属的 appId（即上一步生成播放器签名页面的 appID）
   - **psign** 填写上一步生成的签名结果字符串
2. 单击**预览**即可播放视频。



多端播放器 Demo

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

总结

学习本教程后，您已经掌握如何播放自适应码流视频。
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：2025-03-18 10:47:02

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，并使用播放器播放加密视频。
阅读之前，请先确保已经学习播放器指引的 阶段1：播放原始视频 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：视频加密

1. 登录**云点播控制台** > 应用管理，单击目标应用名称进入**媒资管理** > **音视频管理**页，勾选要处理的视频（FileId 为 2437xx90668057），单击**转自适应码流**。

2. 在**媒体处理**界面：
   - **处理类型**：选择**转自适应码流**。
   - **转自适应码流模板 ID** 选择15（OriRes-HLS-Encrypt）。

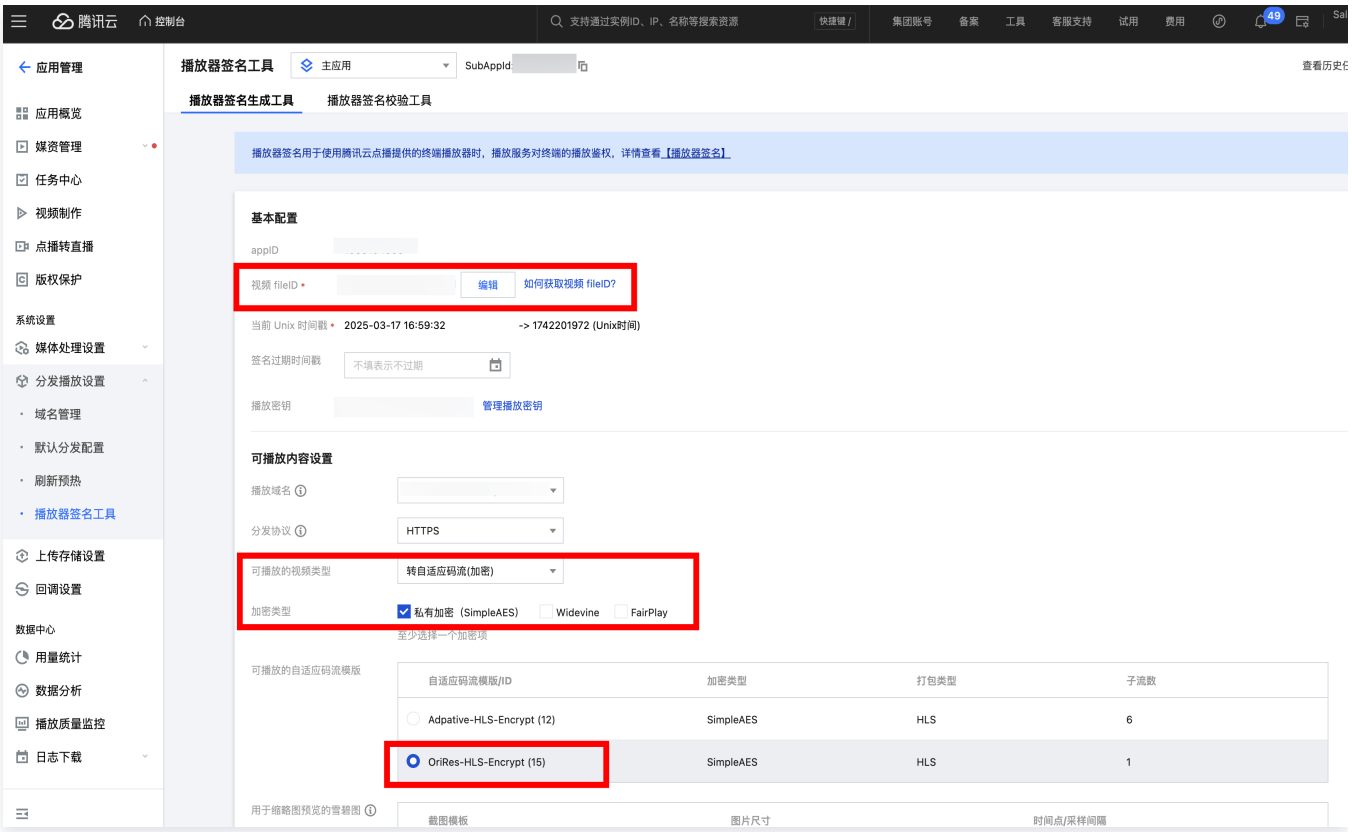
3. 单击**确定**，等待列表中的“任务状态”从“处理中”变为“已完成”，表示视频已处理完毕。
4. 进入**媒资管理** > **音视频管理**，单击发起加密的视频条目右侧的**管理**，进入管理页面。

在**转码信息**模块可以查看加密的自适应码流输出（模板 ID 为 15）。

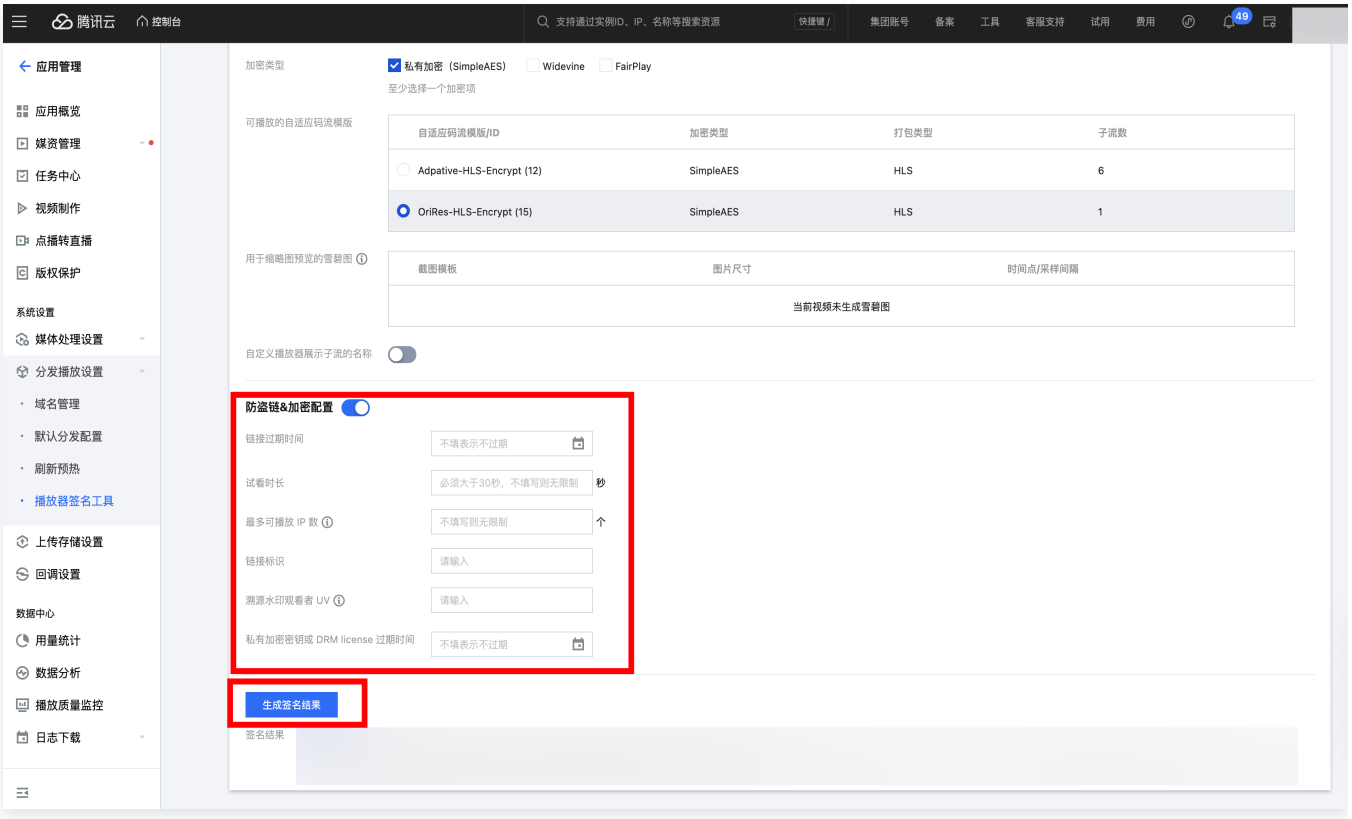
步骤2：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

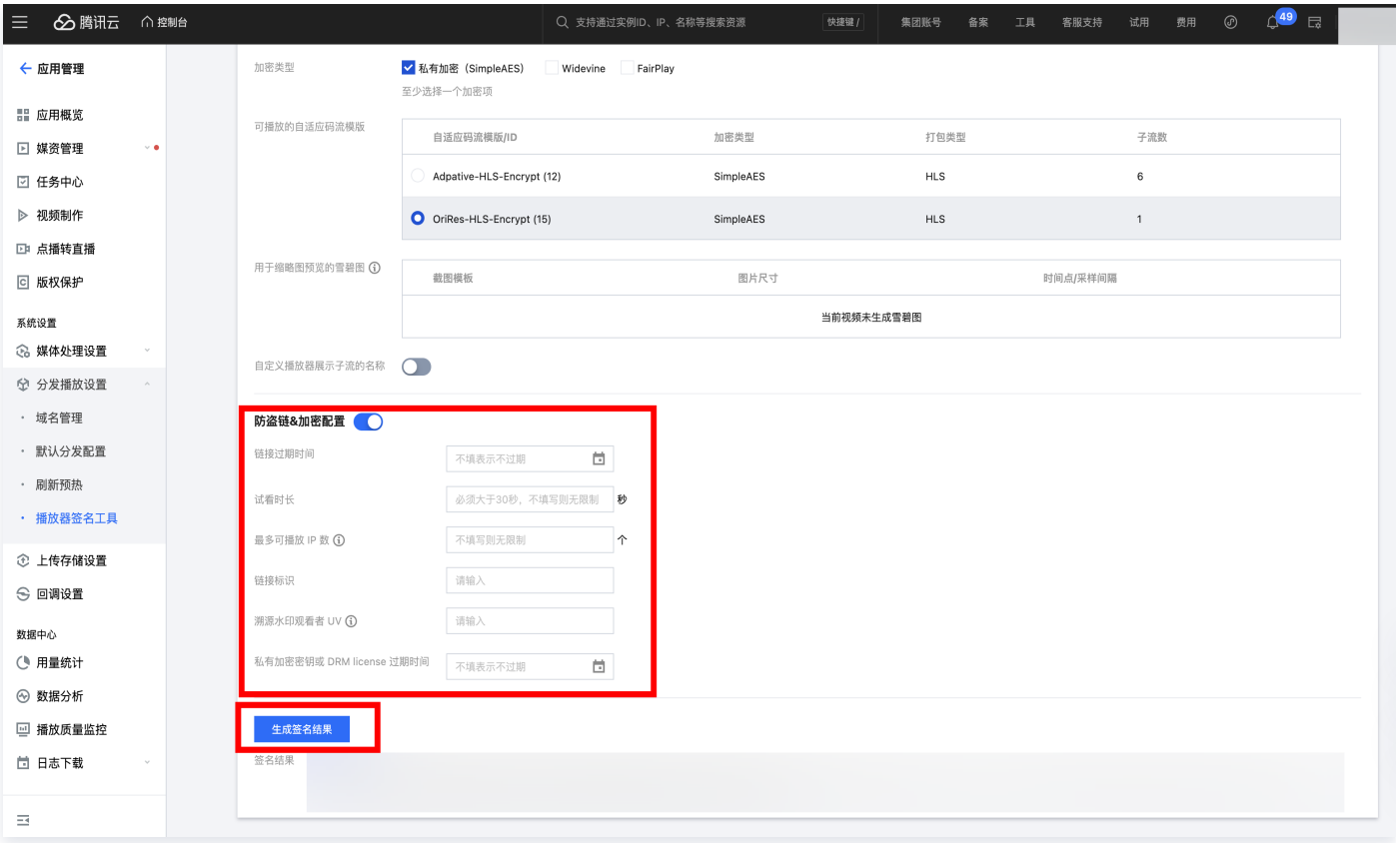
1. 登录**云点播控制台** > **应用管理**，单击目标应用名称进入应用管理页，选择**分发播放设置** > **播放器签名工具**，填写如下信息：
   - **视频 fileId** 填写 步骤1 的 FileId（2437xx90668057）。
   - **签名过期时间戳** 播放器签名过期时间，不填表示签名不过期。
   - **可播放的视频类型** 选择 **转自适应码流(加密)**。
   - **加密类型** 选择 **私有加密（SimpleAES）**。
   - **可播放的自适应码流模板 ID** 选择 **15（OriRes-HLS-Encrypt）**。



- **打开防盗链&加密配置开关。**

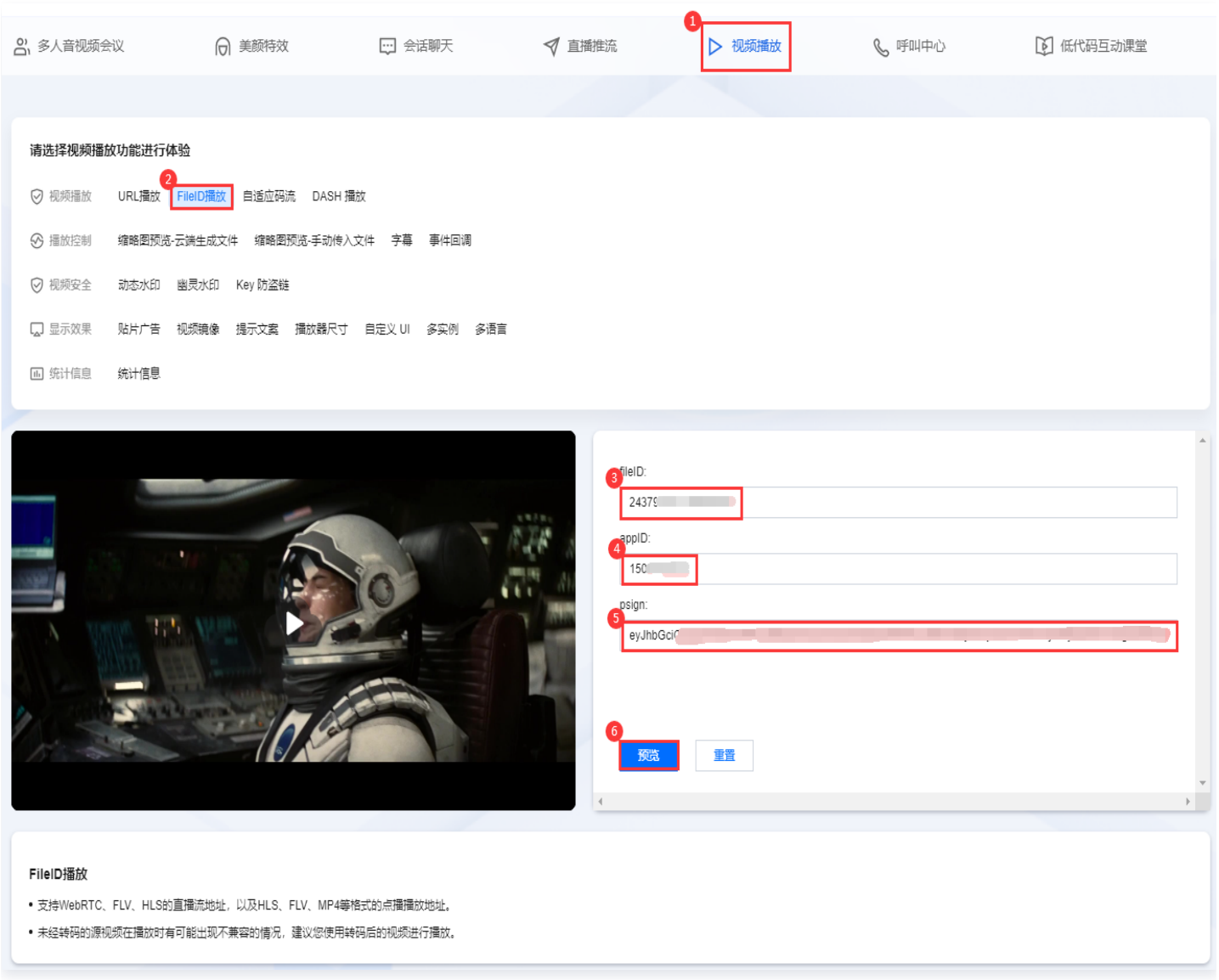


2. **单击生成签名结果**，得到签名结果字符串。



步骤3：播放视频

经过步骤2，我们得到播放视频所需的三个参数：`appId`、`fileId`以及播放器签名（`psign`），下面将展示 Web 端播放视频。



多端播放器 Demo

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

总结

学习本教程后，您已经掌握如何对视频加密，并在播放器中播放。

阶段5：播放多分辨率的加密视频

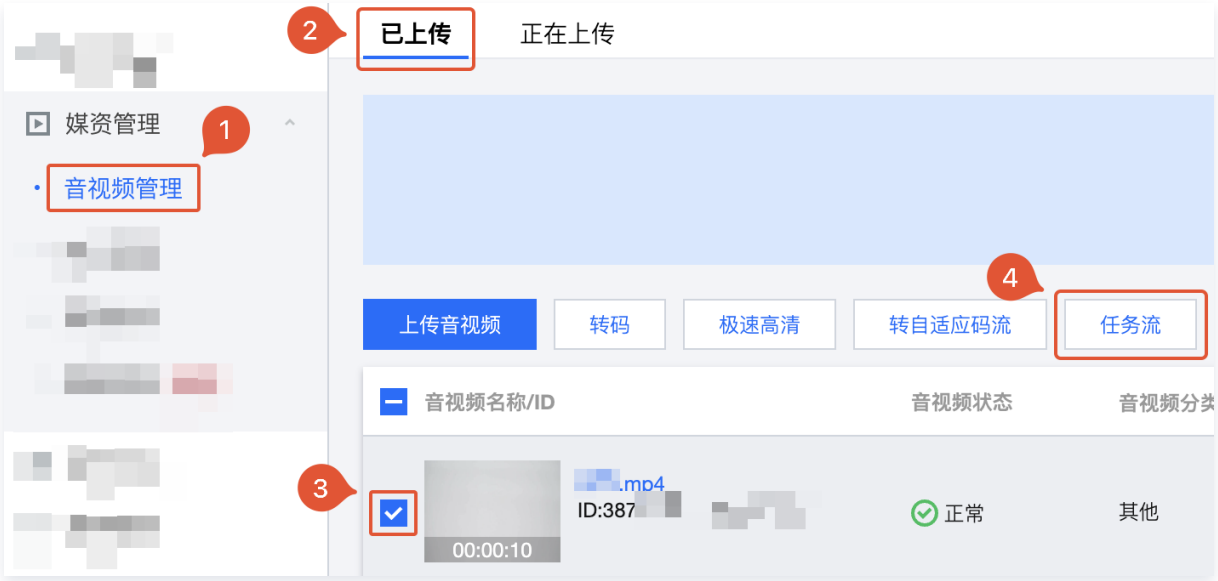
最近更新时间：2024-08-26 10:20:51

学习目标

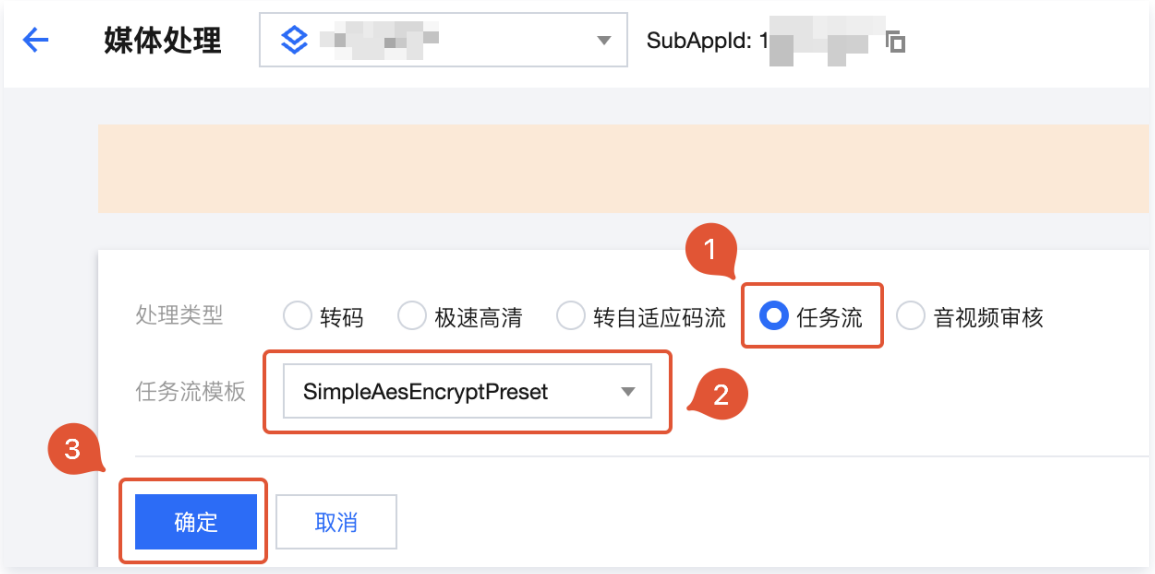
学习 阶段4：播放加密视频 教程，仅掌握播放与原视频同分辨率的加密视频。通过学习本阶段教程，您将了解并掌握如何对视频进行多分辨率转码加密，并使用播放器播放多条分辨率加密视频，有助于在复杂的网络环境下保持视频播放的流畅性。
阅读之前，请先确保已经学习播放器指引的 阶段1：播放原始视频 篇部分，本教程使用了 阶段1 篇开通的账号以及上传的视频。

步骤1：视频加密

1. 登录**云点播控制台** > 应用管理，单击目标应用名称进入**媒资管理** > **音视频管理**页，勾选要处理的视频（FileId 为 387xxxxx8142975036），单击**任务流**。



2. 在**媒体处理**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**，等待列表中的“任务状态”从“处理中”变为“已完成”，表示视频已处理完毕。
4. 进入**媒资管理** > **音视频管理**，单击发起加密的视频条目右侧的**管理**，进入管理页面。

4.1 在**基本信息**模块可以查看：

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）。

4.2 选择**截图信息**页签：

可以看到生成的雪碧图（模板 ID 为 10）。

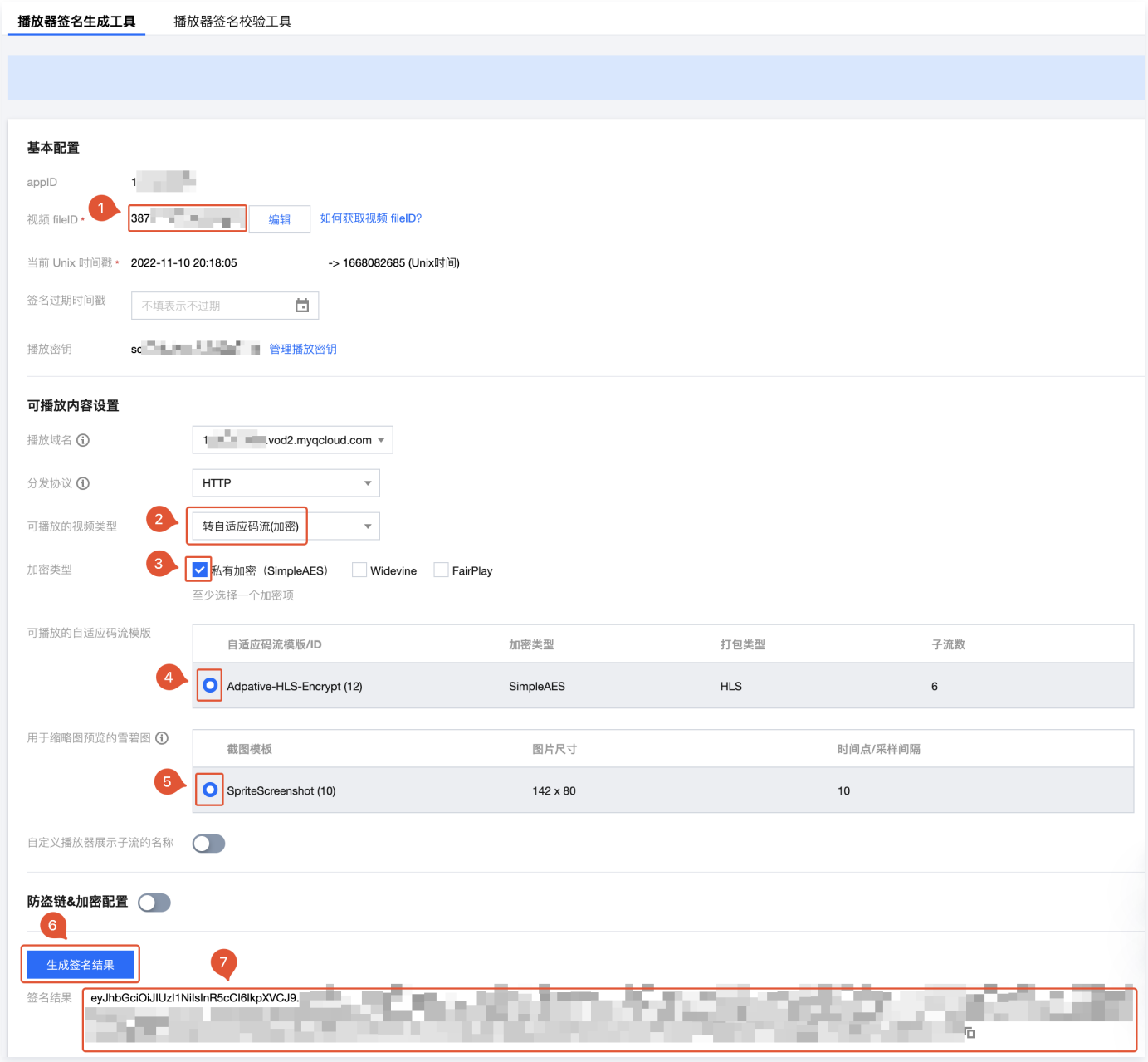

雪碧图截图列表

模板号	小图尺寸	小图行数	小图列数	采样方式	采样间隔	操作
10	142 x 80	10	10	Time	10	复制地址 预览 删除

步骤2：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

1. 登录**云点播控制台** > 应用管理，单击目标应用名称进入应用管理页，选择**分发播放设置** > **播放器签名工具**，填写如下信息：
   - **视频 fileId** 填写 步骤1 的 FileId（387xxxxx8142975036）。
   - **签名过期时间戳** 播放器签名过期时间，不填表示签名不过期。
   - **可播放的视频类型** 选择 **转自适应码流(加密)。**
   - **加密类型** 选择 **私有加密（SimpleAES）。**
   - **可播放的自适应码流模板 ID** 选择 **12（Adaptive-HLS-Encrypt）。**
   - **用于缩略图预览的雪碧图模板 ID** 选择 **10（SpriteScreenshot）。**
2. 单击**生成签名结果**，得到签名结果字符串。

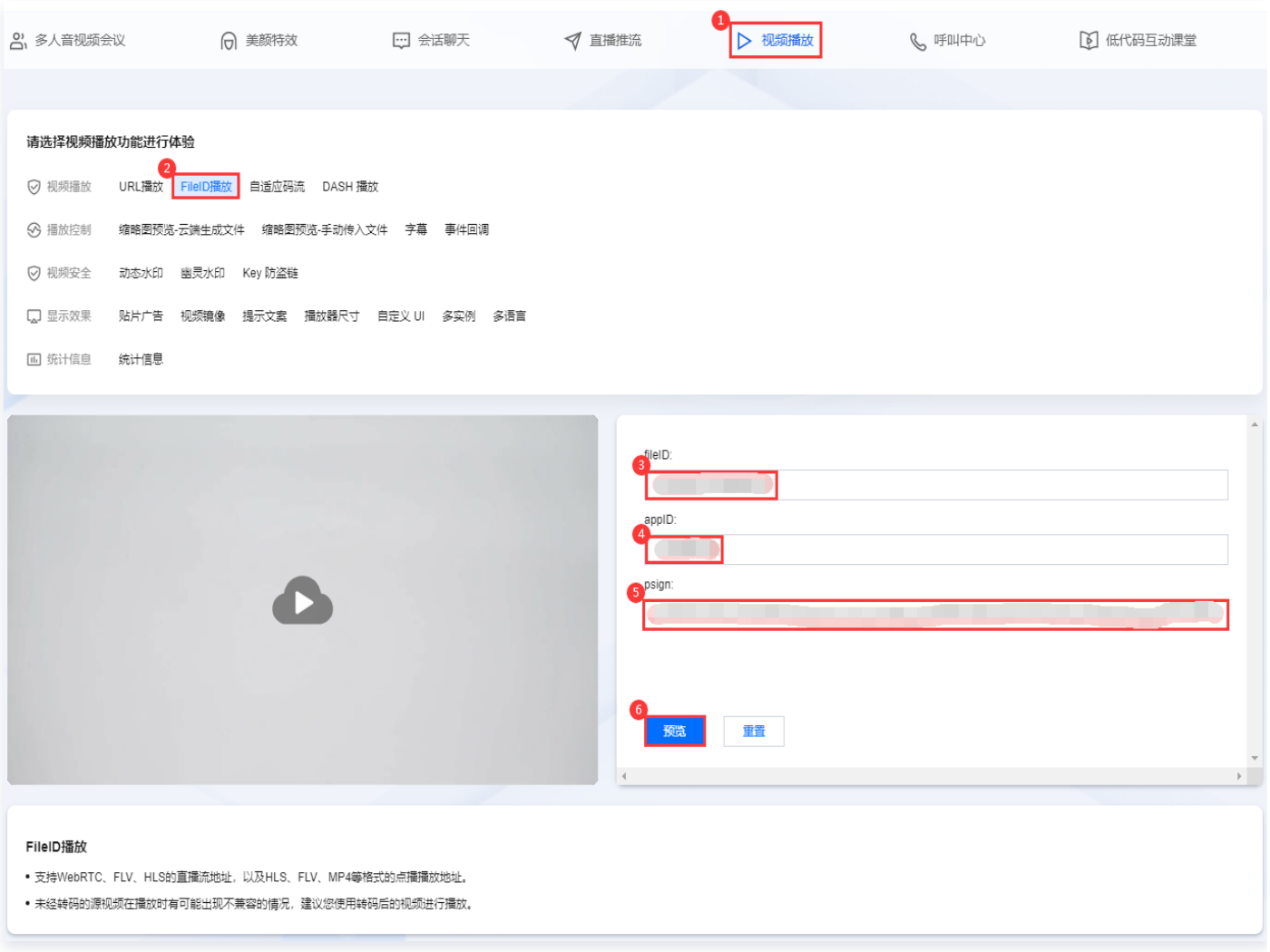


步骤3：播放视频

经过步骤2，我们得到播放视频所需的三个参数：`appId`、`fileId`以及播放器签名（`psign`），下面将展示 Web 端播放视频。

Web 端播放示例

1. 打开 Web端播放器体验，配置如下：
   - **播放器功能**选择**视频播放**。
   - 单击 **FileID 播放** 标签页。
   - **fileID** 填写上一步的 FileId（387xxxxx8142975036）。
   - **appID** 填写文件所属的 appId（即上一步生成播放器签名页面的 appID）。
   - **psign** 填写上一步生成的签名结果字符串。
2. 单击**预览**即可播放视频。



多端播放器 Demo

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

总结

学习本教程后，您已经掌握如何对视频加密，并在播放器中播放。

阶段6：播放长视频方案

最近更新时间：2024-09-29 11:36:42

本文针对音视频平台常见的长视频播放场景，推出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，同时开启 KEY 防盗链、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：

- 如何在云点播设置 KEY 防盗链，实现有效时间、播放人数、播放时长等控制。
- 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，请先确保已经学习播放器指引的 阶段1：用播放器播放视频 篇部分，了解云点播 FileId 的概念。

操作步骤

步骤1：开启 KEY 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** > **应用管理**，单击目标应用名称进入应用管理页，在左侧导航栏选择**分发播放设置** > **域名管理**，进入设置页面。

2. 单击“**访问控制**”页签，找到**Key 防盗链**，单击灰色按钮开启，在弹出的设置页面单击**随机生成**来生成一个随机的 Key，然后单击**确定**保存生效。

步骤2：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。

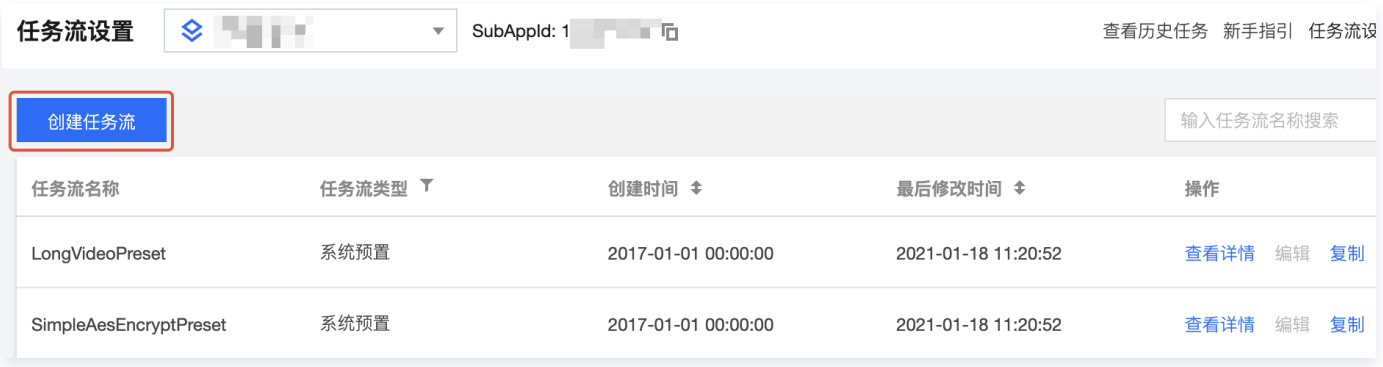
1. 进入目标应用管理页后，在左侧导航栏选择**媒体处理设置 > 模板设置**，进入“**转自适应码流模板**”页单击**创建转自适应码流模板**。

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。

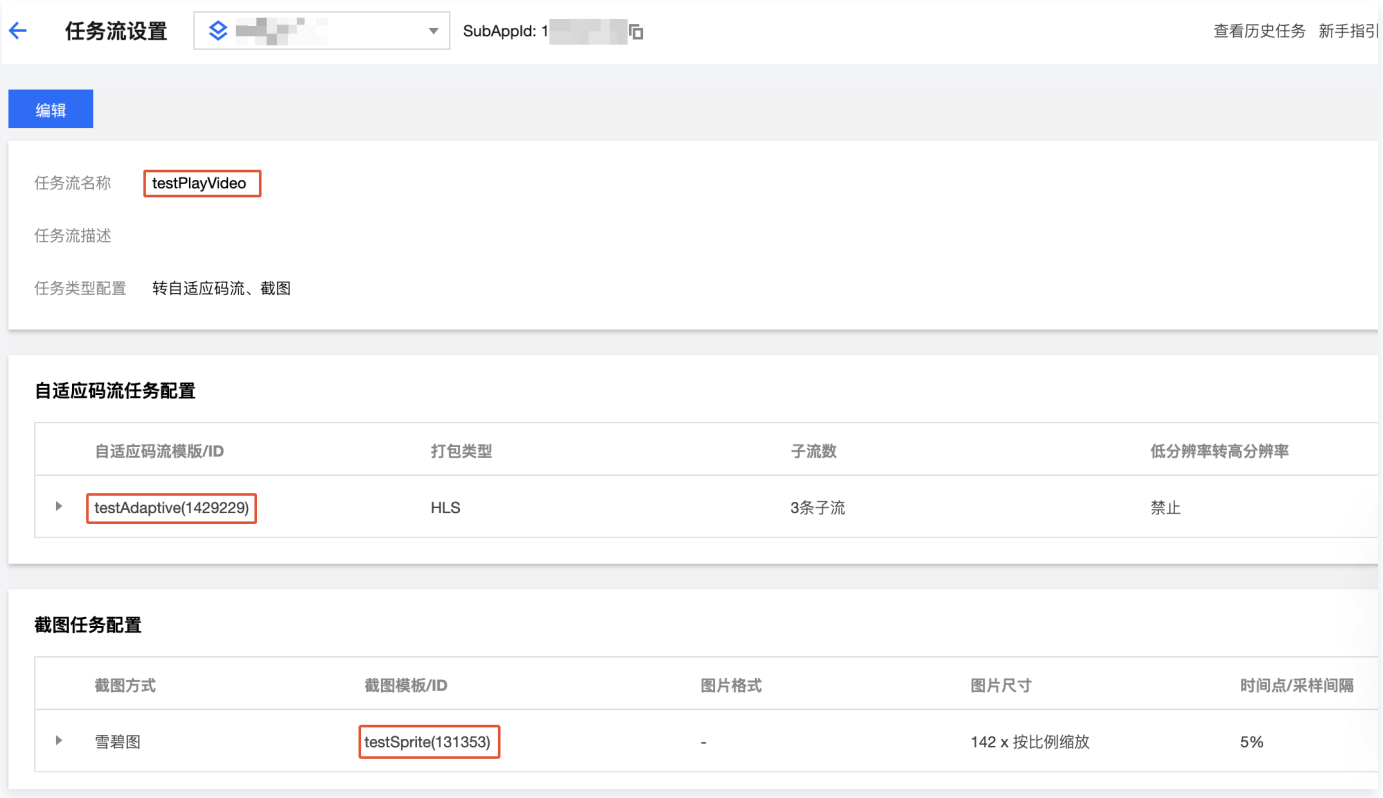
2. 选择**媒体处理设置 > 模板设置 > 截图模板**，单击**创建截图模板**。

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。

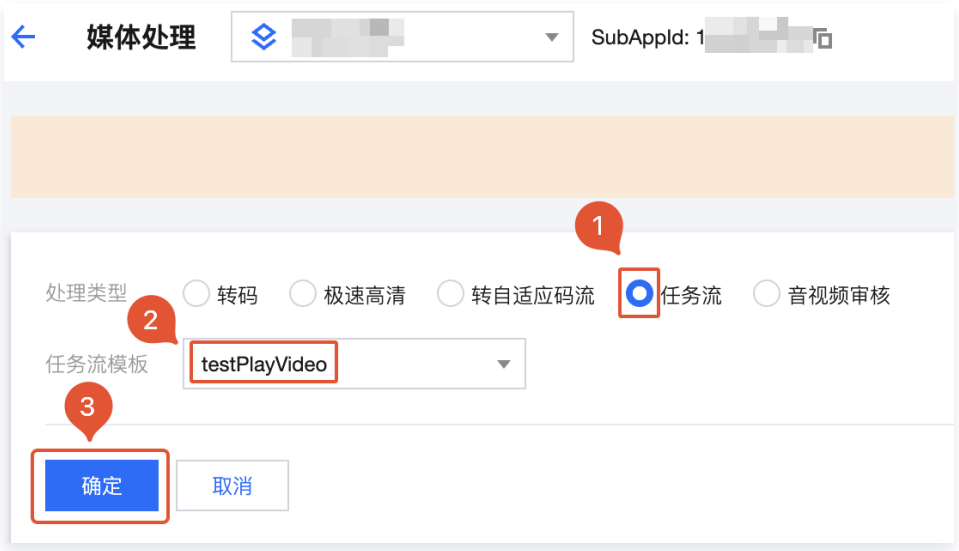
3. 通过任务流，添加自适应码流模板与雪碧图模板。
选择**媒体处理设置 > 任务流设置**，单击**创建任务流**。



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（FileId 为 387xxxxx8142975036），单击**任务流**，选择任务流模板，发起任务。

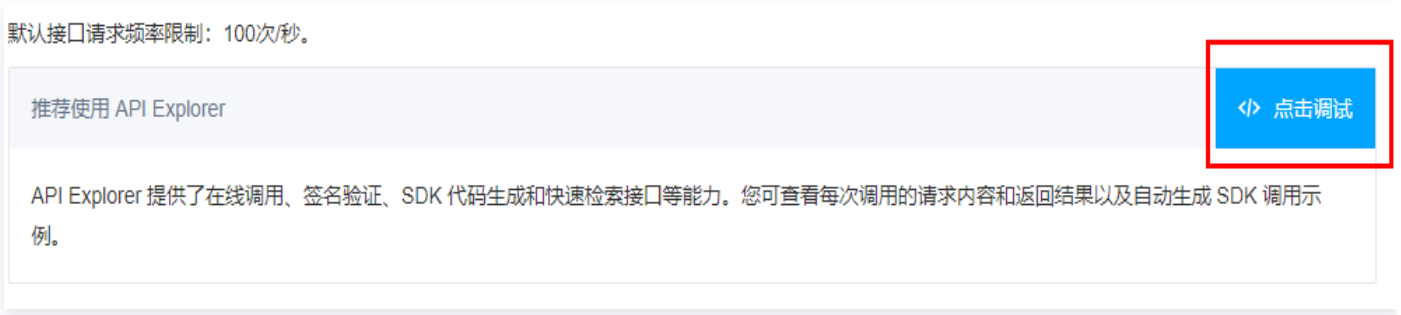


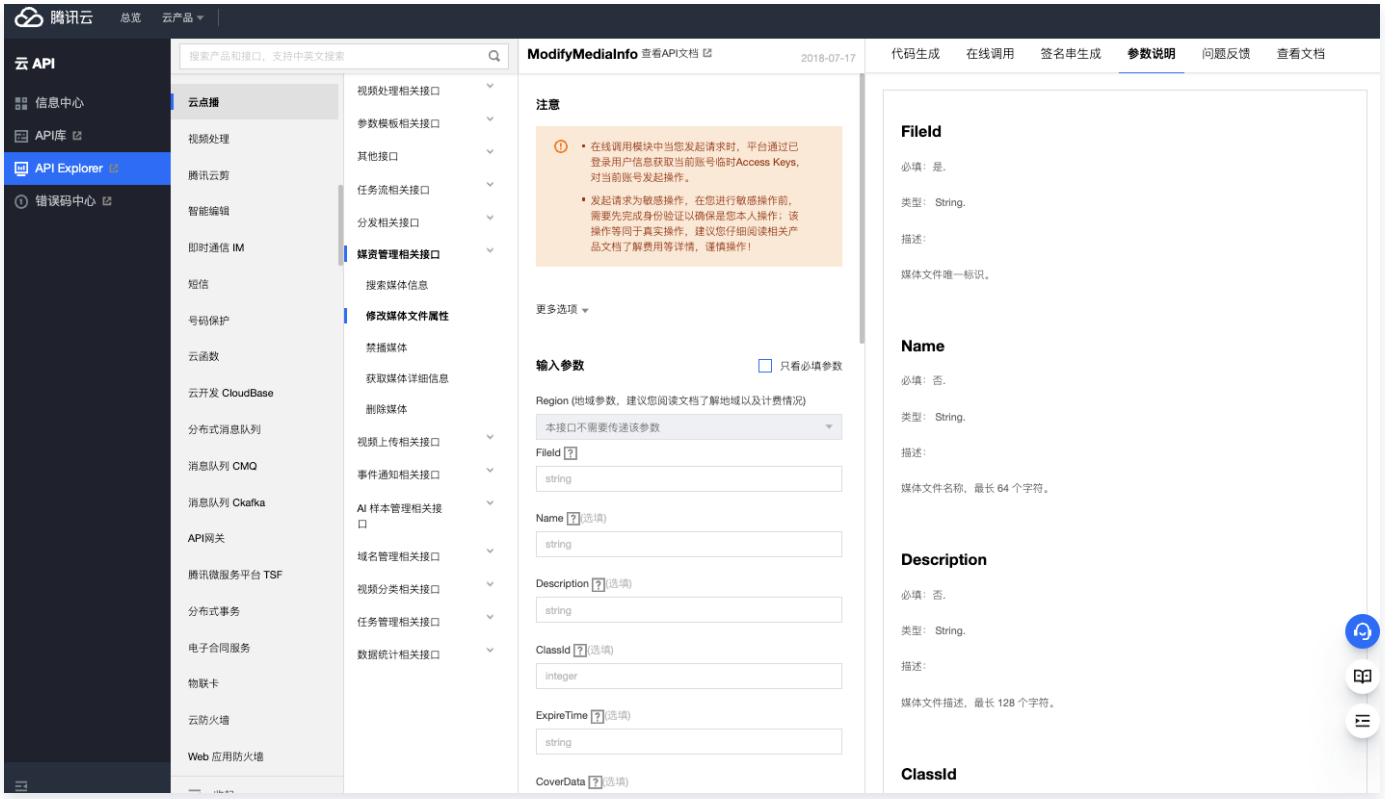
5. 至此，我们可以在**任务中心**中，查看任务执行情况，完成后获取任务结果。

步骤3：增加视频打点信息

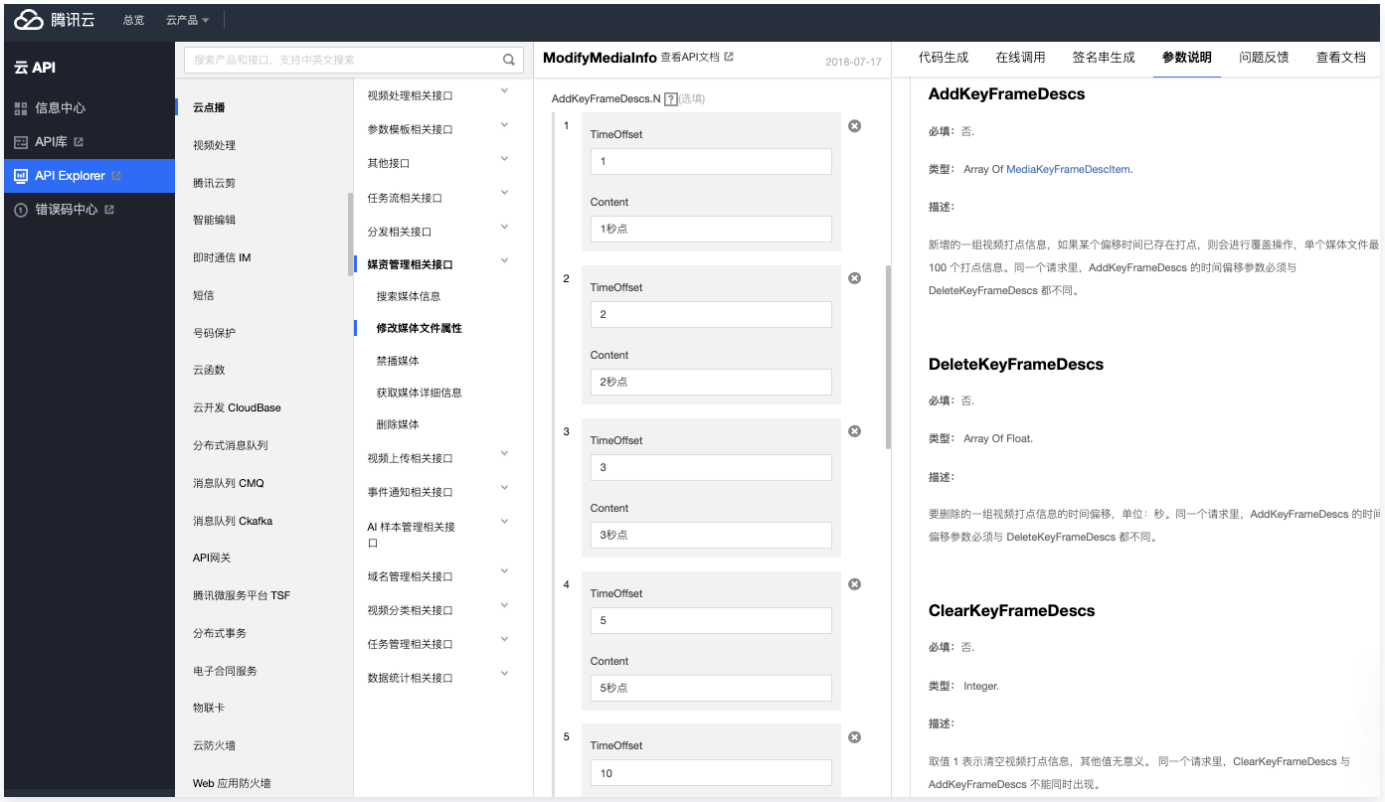
本步骤，我们将指导您新增的一组视频打点信息。

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**点击调试**，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。

注意：

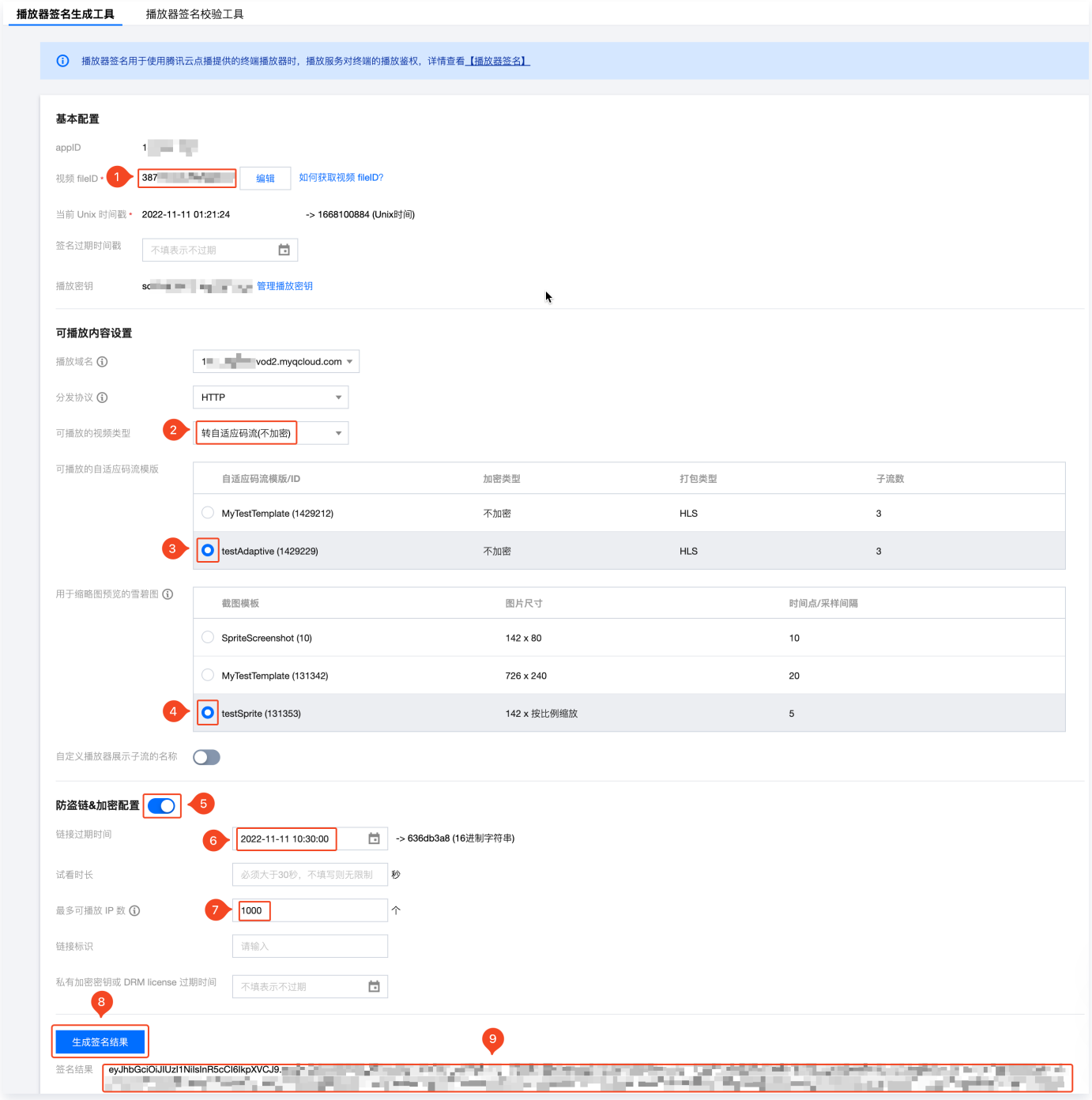
- 该功能仅支持桌面端浏览器。

- 在浏览器劫持视频播放的情况下，该功能无法使用。

步骤4：生成播放器签名

本步骤，我们使用签名工具快速生成播放器签名，用于播放器播放视频。

1. 进入目标应用管理页后，在左侧导航栏选择**分发播放设置** > **播放器签名工具**，填写如下信息：
   - **视频 fileId：**填写 步骤2 使用的 FileId（387xxxxx8142975036）。
   - **签名过期时间戳：**播放器签名过期时间，不填表示签名不过期。
   - **可播放的视频类型：**选择“**转自适应码流(不加密)**”。
   - **可播放的自适应码流模板：**选择 `testAdaptive (1429229)`。
   - **用于缩略图预览的雪碧图：**选择 `testSprite (131353)`。
   - **防盗链&加密配置**开关打开，配置如下：
     - **链接过期时间：**设置获取的播放链接的防盗链签名过期时间。
     - **最多可播放 IP 数：**设置最多允许多少个 IP 不同的终端播放。
2. 单击**生成签名结果**，得到签名结果字符串。



步骤5：播放器端集成

经过步骤4，我们得到播放视频所需的三个参数：`appId`、`fileId` 以及播放器签名（`psign`）
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播放器请参见 集成指引，引入播放器 SDK 文件之后，使用 `appId`、`fileId` 以及播放器签名（`psign`）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-playsinline>
</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: '387xxxxx8142975036', // 要播放的视频 fileID
    appID: '1400329073', // 要播放视频的点播账号 appID
    psign:'psignxxxx'    // psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
});
```

iOS 端

用户需要集成视立方播放器请参见 集成指引，集成完后，使用 `appId` 、 `fileId` 以及播放器签名（ `psign` ）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 = @"387xxxxx8142975036"; // 配置 FileId
// pSign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
model.videoId.pSign = @"psignxxxx";
[_playerView playWithModelNeedLicence:model];
```

```
[_playerView playWithModel:model];
```

Android 端

集成视立方播放器请参考集成指引，集成完后，使用 `appId` 、`fileId` 以及播放器签名（`psign`）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

```
<!-- 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);

SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1400329073;// 配置 AppId
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "387xxxxx8142975036"; // 配置 FileId
// pSign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
model.videoId.pSign = "psignxxxx";

mSuperPlayerView.playWithModel(model);
```

总结

至此，您就可以在播放器播放开启了 KEY 防盗链的点播账号下的媒体文件，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。
更多播放器功能请参见 功能说明 。

含 UI 集成方案
Web 接入指引
TCPlayer 集成指引

最近更新时间：2025-05-09 11:28:12

本文档将介绍适用于点播播放和直播播放的 Web 播放器 SDK（TCPlayer），它可快速与自有 Web 应用集成，实现视频播放功能。Web 播放器 SDK（TCPlayer）内默认包含部分 UI，您可按需取用。

概述

Web 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地支持媒体播放。

功能支持

功能\浏览器	Chrome	Firefox	Edge	QQ 浏览器	Mac Safari	iOS Safari	微信	Android Chrome	IE 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	✓	✓	✓	✓	✓	✓	✓	✓	✓
幽灵水印	✓	✓	✓	✓	✓	✓	✓	✓	✓
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。

准备工作

播放器 SDK Web 端（TCPlayer）自 5.0.0 版本起需获取 License 授权后方可使用。若您无需使用新增的高级功能，可直接申请基础版 License 以**继续免费使用 TCPlayer**；若您需要使用新增的高级功能则需购买高级版 License。具体信息如下：

TCPlayer 功能	功能范围	所需 License	定价	授权单位
基础版功能	包含 5.0.0 以前版本提供的全部功能，详情见 产品功能	播放器 Web 端基础版 License	0元 免费申请	精准域名（1个 License 最多可关联10个精准域名）
高级版功能	基础版功能、VR 播放、安全检查	播放器 Web 端高级版 License	399元/月 立即购买	泛域名（1个 License 最多可授权1个泛域名）

说明：

1. 播放器 Web 端基础版 License 可免费申请，申请后有效期默认1年；在有效期剩余时间小于30天时，可免费续期。
2. 为方便本地开发，播放器不会校验 localhost 或者 127.0.0.1，因此申请 License 时不需要申请这类本地服务域名。

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

播放器 SDK 支持 CDN 和 NPM 两种集成方式：

1. 通过 CDN 集成

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://tcsdk.com/player/tcplayer/release/v5.3.3/tcplayer.min.css"
rel="stylesheet"/>
 <!--播放器脚本文件-->
 <script
src="https://tcsdk.com/player/tcplayer/release/v5.3.3/tcplayer.v5.3.3.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，自行部署情况下，需要手动引用资源包文件夹 libs 下面的依赖文件，否则会默认请求腾讯云 cdn 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.x.xx.m.js"></script>
<!--播放器脚本文件-->
<script src="aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script>
```

2. 通过 npm 集成

首先安装 tcplayer 的 npm 包：

```
npm install tcplayer.js
```

导入 SDK 和样式文件：

```
import TCPlayer from 'tcplayer.js';
import 'tcplayer.js/dist/tcplayer.min.css';
```

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-playsinline>
</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。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 使用播放器播放 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。

```
// player-container-id 为播放器容器 ID，必须与 html 中一致
var player = TCPlayer('player-container-id', {
    sources: [{
      src: 'path/to/video', // 播放地址
    }],
    licenseUrl: 'license/url', // license 地址，必传。参考准备工作部分，在视立方控制台申请 license 后可获得 licenseUrl
});

// 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）
    //必传 psign， psign 即播放器签名，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',
    licenseUrl: 'license/url', // 参考准备工作部分，在视立方控制台申请 license 后可获得 licenseUrl
});
```

> **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

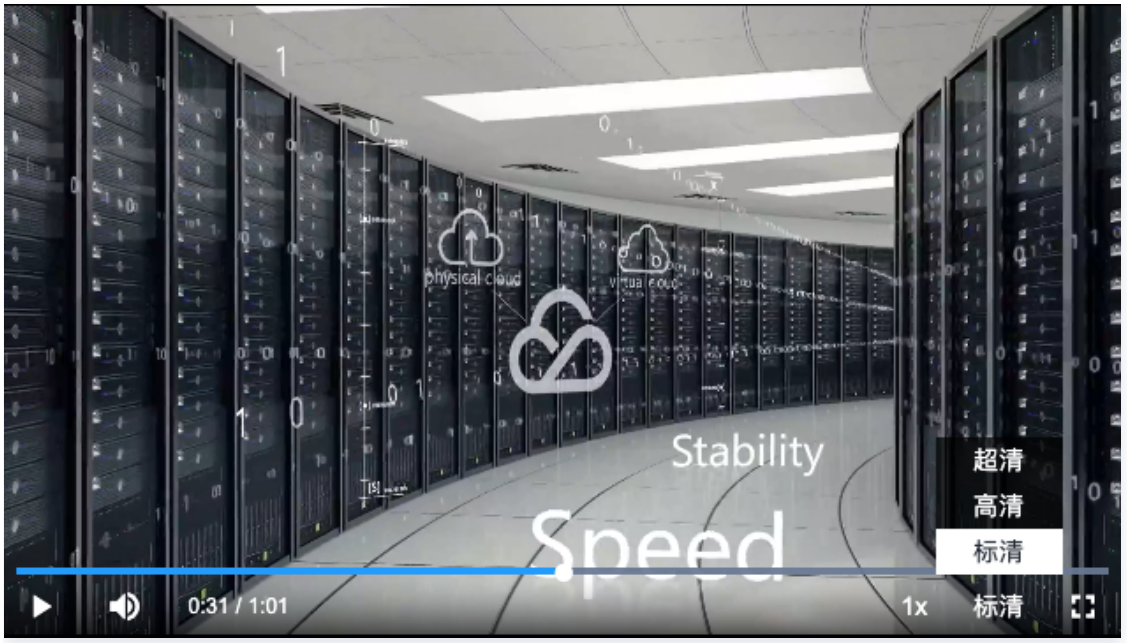
播放器可以结合云点播的服务端能力实现高级功能，例如自动切换自适应码流、预览视频缩略图、添加视频打点信息等。这些功能在播放长视频方案 中有详细的说明，可以参考文档实现。

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。

TCPlayer 清晰度配置说明

最近更新时间：2023-10-11 15:05:21

概述



在播放过程中，您可以通过自动或手动切换视频清晰度，以适应不同尺寸的播放设备和网络环境，从而提高观看体验。本文将从以下几个场景进行说明。

直播场景

直播场景以 URL 的形式来播放视频，初始化播放器时，可以通过 sources 字段指定所要播放的 URL。或者在初始化播放器之后，调用播放器实例上的 src 方法进行播放。

1. 自适应码率（ABR）

自适应码率地址在切换时可以做到无缝衔接，切换过程不会出现中断或跳变，实现了观感和听感的平滑过渡。使用该技术也比较简单，仅需将播放地址传给播放器，播放器会自动解析子流，并将清晰度切换组件渲染到控制条上。

示例1: 播放 HLS 自适应码率地址

在初始化播放器时，传入自适应码率地址，播放器将自动生成清晰度切换组件，并会根据网络状况进行自动切换。

```
const player = TCPlayer('player-container-id', { // player-container-id 为播放器容器ID，必须与html中一致
  sources: [{
    src: 'https://hls-abr-url', // hls 自适应码率地址
  }],
});
```

> **注意：**

> 解析 HLS 自适应码率的子流，需要依赖播放环境的 MSE API。在不支持 MSE 的浏览器环境（例如 iOS 的 Safari 浏览器），会由浏览器内部处理，根据网络情况自动切换清晰度，但无法解析出多种清晰度来供您手动切换。

示例2：播放 WebRTC 自适应码率地址

在 WebRTC 自适应码率场景，传入地址后，播放器会根据地址中的 ABR 模板自动拆解子流地址。

```
const player = TCPlayer('player-container-id',{
  sources: [{
    src: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c&tabr_bitrates=d1080p,d540p
,d360p&tabr_start_bitrate=d1080p&tabr_control=auto'
  }],

  webrtcConfig: {
    // 是否渲染多清晰度的开关，默认开启，可选
    enableAbr: true,
    // 模板名对应的label名，可选
    abrLabels: {
      d1080p: 'FHD',
      d540p: 'HD',
      d360p: 'SD',
      auto: 'AUTO',
    },
  },
});
```

这里对 WebRTC 地址中的参数做以下说明：

1. tabr_bitrates 指定了 ABR 模板，有几个模板就会渲染出几个清晰度。如果没有单独设置清晰度的 label，模板名称（如 d1080p）将被设为清晰度名称。
2. tabr_start_bitrate 指定了起播的清晰度规格。
3. tabr_control 设置是否开启自动切换清晰度。开启后，播放器会单独渲染出自动清晰度选项。

2. 手动设置清晰度

如果播放地址不是自适应码率地址，也可以手动设置清晰度。参见如下代码：

```
const player = TCPlayer('player-container-id', { // player-container-id 为播放器容器
ID，必须与html中一致
  multiResolution:{
    // 配置多个清晰度地址
    sources:{
      'SD':[{
        src: 'http://video-sd-url',
      }],
      'HD':[{
        src: 'http://video-hd-url',
      }],
```

```
        'FHD':[{
          src: 'http://video-fhd-url',
        }]
      },
      // 配置每个清晰度标签
      labels:{
        'SD':'标清','HD':'高清','FHD':'超清'
      },
      // 配置各清晰度在播放器组件上的顺序
      showOrder:['SD','HD','FHD'],
      // 配置默认选中的清晰度
      defaultRes: 'SD',
    },
});
```

点播场景

在点播场景下，如果通过 fileID 播放，播放哪种规格的文件（原始文件、转码文件、自适应码率文件）以及自适应码率文件子流的清晰度，都是在播放器签名中设置的。您可以参见指引 播放自适应码流视频，以便于您了解点播场景下播放视频的整个流程。
计算播放器签名时，可以通过 contentInfo 字段中的 resolutionNames 来设定不同分辨率的子流的展示名字。不填或者填空数组则使用默认配置。

```
resolutionNames: [{
  MinEdgeLength: 240,
  Name: '240P',
}, {
  MinEdgeLength: 480,
  Name: '480P',
}, {
  MinEdgeLength: 720,
  Name: '720P',
}, {
  MinEdgeLength: 1080,
  Name: '1080P',
}, {
  MinEdgeLength: 1440,
  Name: '2K',
}, {
  MinEdgeLength: 2160,
  Name: '4K',
}, {
  MinEdgeLength: 4320,
  Name: '8K',
}]
```

播放时的子流数量取决于转码时根据不同的自适应码率模板转换出的子流数。这些子流会依据短边长度落在由 resolutionNames 设定的哪个 MinEdgeLength 范围，再以对应的 Name 作为清晰度名称进行展示。
若您需要快速体验生成播放器签名，可以使用腾讯云点播控制台的 播放器签名生成工具 。

TCPlayer 快直播降级说明

最近更新时间：2023-10-11 15:05:21

降级场景

快直播基于 WebRTC 实现，依赖于操作系统和浏览器对于 WebRTC 的支持。
目前，SDK 对以下操作系统和浏览器进行了测试，测试结果如下：

操作系统	操作系统版本	浏览器类型	浏览器版本	是否支持拉流
Windows	win 10	Chrome	86+	✓
		Firefox	88+	✓
		Microsoft Edge	86+	✓
macOS	10.5+	Safari	13.1+	✓
		Chrome	86+	✓
		Firefox	88+	✓
		Microsoft Edge	86+	✓
iOS	13.1.1+	Safari	13.7+	✓
		Chrome	86+	✓
		Firefox	33+	✓
		Microsoft Edge	89	✓
		微信内嵌	-	✓
Android	-	Chrome	86+	✓
		Firefox	88+	✓
		微信内嵌	X5 内核	✓
		微信内嵌	XWeb 内核	✓

此外，在部分支持 WebRTC 的浏览器，也会出现解码失败或者服务端问题，这些情况下，播放器都会将 WebRTC 地址转换为兼容性较好的 HLS 地址来播放，这个行为称为降级处理。
总结一下，会触发降级的场景有以下几个：

- 浏览器环境不支持 WebRTC 。
- 连接服务器失败，并且连接重试次数已超过设定值（内部状态码 -2004）。
- 播放过程解码失败（内部状态码 -2005）。
- 其他 WebRTC 相关错误（内部状态码 -2001）。

降级方式

1. 自动降级

初始化播放器时，通过 sources 字段传入了快直播地址，在需要降级处理的环境，播放器自动会进行协议的转换，将快直播地址转换为 HLS 协议地址。
例如，快直播地址：

```
webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c
```

会自动转换为：

```
https://global-lebtest-play.myqcloud.com/live/lebtest.m3u8?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c
```

2. 指定降级

在播放自适应码率（ABR）场景，如果需要降级，并不能直接通过格式转换得到自适应码率的 HLS 地址，需要手动指定。又或者是在用户希望手动指定的其他场景，都可以通过如下方式指定降级地址，这里的地址并不局限于 HLS 协议，也可以是其他协议地址：

```
var player = TCPlayer('player-container-id',{
    sources: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c&tabr_bitrates=d1080p,d540p
,d360p&tabr_start_bitrate=d1080p',
    webrtcConfig: {
      fallbackUrl: 'https://global-lebtest-
play.myqcloud.com/live/lebtest_HLSABR.m3u8',
    },
});
```

降级回调

当触发降级时，播放器会触发回调：

```
player.on('webrtcfallback', function(event) {
    console.log(event);
});
```

iOS 接入指引

最近更新时间：2025-06-04 15:55:52

产品概述

腾讯云视立方 iOS 播放器组件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 播放器组件 。
- 如何创建和使用播放器 。
- 播放器推出短视频组件、画中画2.0、VR 播放等高级组件，功能介绍和使用指引请参见 移动端高级功能 。

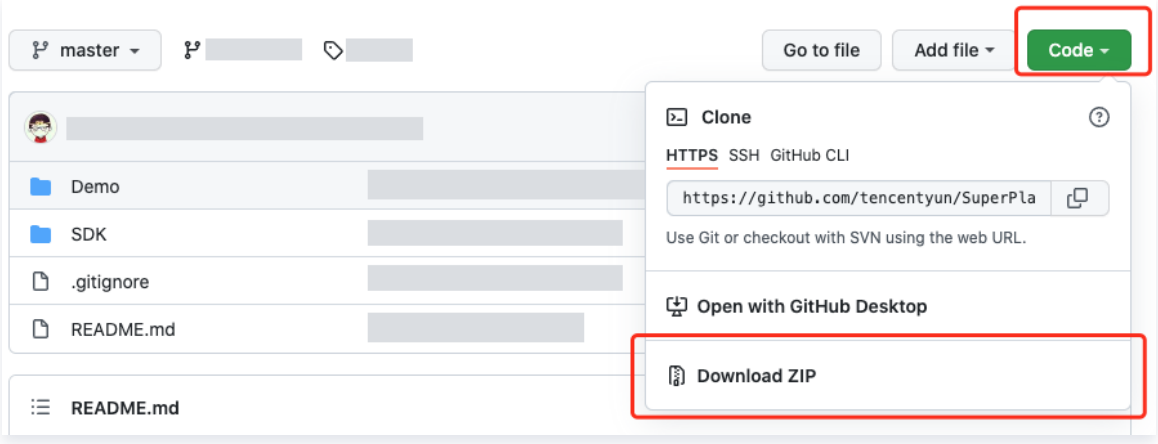
集成准备

步骤1：项目下载

腾讯云视立方 iOS 播放器的项目地址是 LiteAVSDK/Player_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 播放器组件项目工程。

下载播放器组件 ZIP 包

您可以直接下载播放器组件 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 方式直接集成 SuperPlayer。

```
pod 'SuperPlayer'
```

   - 如果您需要依赖 Player 版，可以在 podfile 文件中添加如下依赖：

```
pod 'SuperPlayer/Player'
```

- 如果您需要依赖专业版，可以在 podfile 文件中添加如下依赖：

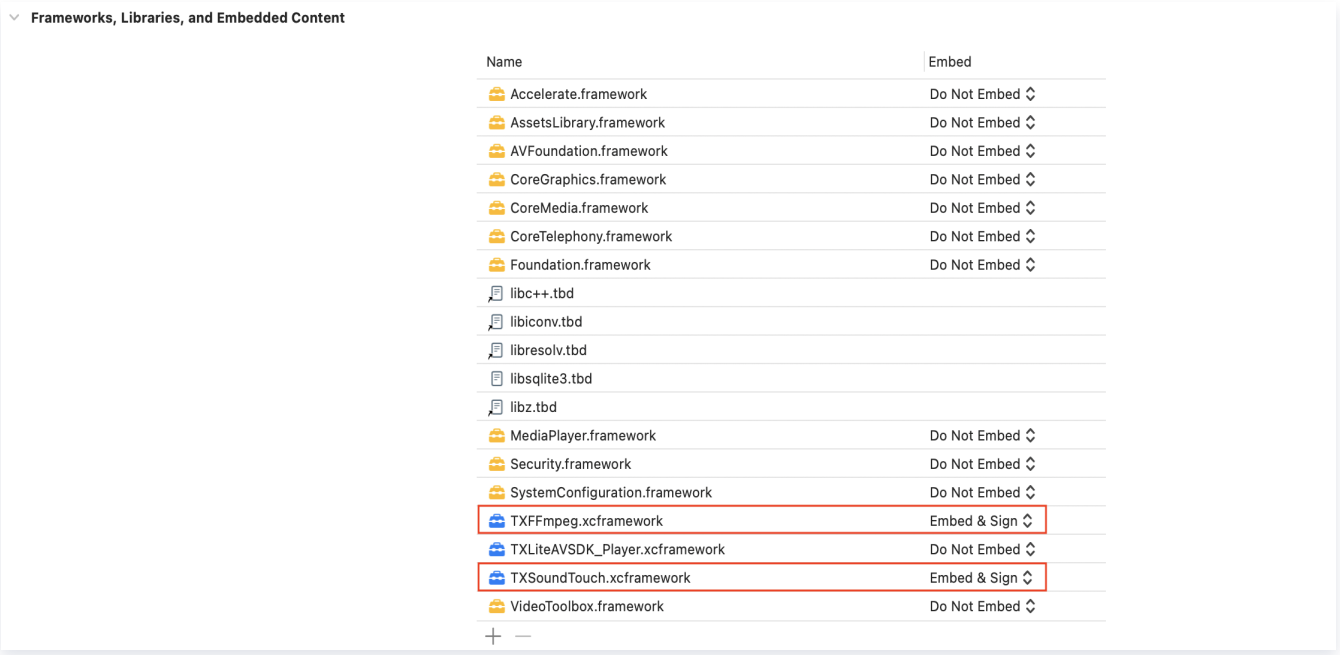
```
pod 'SuperPlayer/Professional'
```

2. 执行 `pod install` 或 `pod update` 。

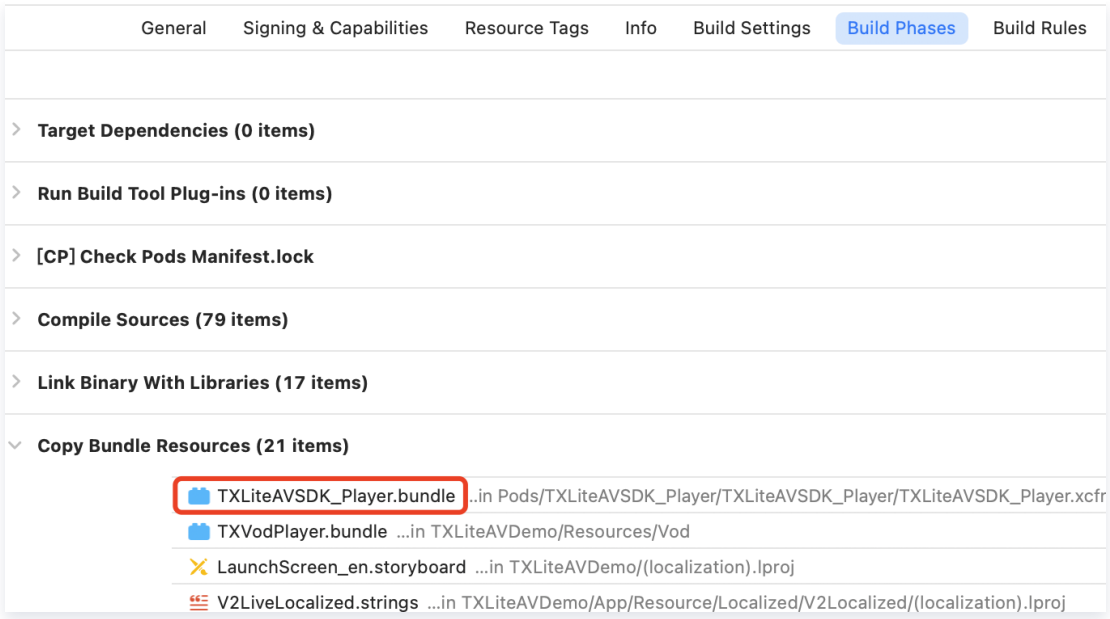
手动下载 SDK

1. 下载 SDK + Demo 开发包，腾讯云视立方 iOS 播放器项目为 LiteAVSDK/Player_iOS 。
2. 导入 `TXLiteAVSDK_Player.xcframework` 到工程中，并勾选 `Do Not Embed` 。
3. 将 Demo/TXLiteAVDemo/SuperPlayerKit/SuperPlayer 拷贝到自己的工程目录下。
4. SuperPlayer 依赖第三方库包括：AFNetworking、SDWebImage、Masonry、TXLiteAVSDK_Player。
   如果是手动集成 TXLiteAVSDK_Player，需要添加所需要的系统库和 library：
   - **系统 Framework 库**：MetalKit、ReplayKit、SystemConfiguration、CoreTelephony、VideoToolbox、CoreGraphics、AVFoundation、Accelerate、MobileCoreServices。
   - **系统 Library 库**：libz、libresolv、libiconv、libc++、libsqlite3。

   具体操作步骤可以 参考：定制开发 > 点播场景 > 接入文档 > SDK 集成 步骤1 > 手动集成 SDK。
   此外还需要把 TXLiteAVSDK_Player 文件下的 TXFFmpeg.xcframework 和 TXSoundTouch.xcframework 以动态库的方式加进来如下图所示：



5. 如果是用 Pod 的方式集成 TXLiteAVSDK_Player，不需要添加任何库。
6. 从 **11.7.15343版本** 以后，Player SDK 适配了苹果的隐私清单，下载对应 SDK 并将 SDK 内 **TXLiteAVSDK_Player.bundle** 添加到项目工程里：



如果是 pods 引入 SDK 的话，可以忽略上述步骤。

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 播放器 License 获取相关授权。
- 获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

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 播放器组件支持云点播 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 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，或者使用播放器组件签名 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 用播放器组件播放视频，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
// psign 即播放器签名，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 playWithModelNeedLicence: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 playWithModelNeedLicence:model];
```

本地视频播放

```
SuperPlayerModel *model = [[SuperPlayerModel alloc]init];
//把您的视频文件添加到工程，然后通过NSBundle获取视频的文件路径
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"your_video_name"
ofType:@"mp4"];
model.videoURL = [filePath stringByReplacingOccurrencesOfString:@"file://"
withString:@""];
[_playerView playWithModelNeedLicence: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 播放器组件的创建、播放视频和退出播放的能力集成。

功能使用

1. 全屏播放

播放器组件支持全屏播放，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 > 播放器组件** 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:UIControlEventTouchUpInside];
```

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

播放器组件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

播放器组件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

播放器组件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。

- 当播放器组件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。
- 当播放器组件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 = @"https://qcloudimg.tencent-
cloud.cn/raw/3d895b8d2c37b447cdd2691fb8d9d58c.png";
[self.playerView playWithModelNeedLicence:model];
```

4. 视频列表轮播

播放器组件支持视频列表轮播，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
playWithModelListNeedLicence:modelArray isLoopPlayList:YES startIndex:0];
```

```
(void)playWithModelListNeedLicence:(NSArray *)playModelList isLoopPlayList:
(BOOL)isLoop startIndex:(NSInteger)index;
```

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. 画中画功能

> **注意：**
> 在现有基础画中画的方案上，现已经升级推出 高级画中画版本，主要支持加密视频画中画、离线播放画中画、从前台无缝切换到画中画的场景，优化了实现方式和逻辑，无需长时间等待，实现真正意义的“秒切”效果。

画中画（PictureInPicture）在 iOS 9 就已经推出了，不过之前都只能在 iPad 上使用，iPhone 要使用画中画需更新到 iOS 14 才能使用。 目前腾讯云播放器可以支持应用内和应用外画中画能力，极大的满足用户的诉求。使用前需要开通后台模式。
步骤为：XCode 选择对应的 Target > Signing & Capabilities > Background Modes，勾选 “Audio, AirPlay, and Picture in Picture”。

使用画中画能力代码示例：

```
// 进入画中画
if (![TXVodPlayer isSupportPictureInPicture]) {
    return;
}
[_vodPlayer enterPictureInPicture];

// 退出画中画
[_vodPlayer exitPictureInPicture];
```

6. 视频试看

播放器组件支持视频试看功能，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。

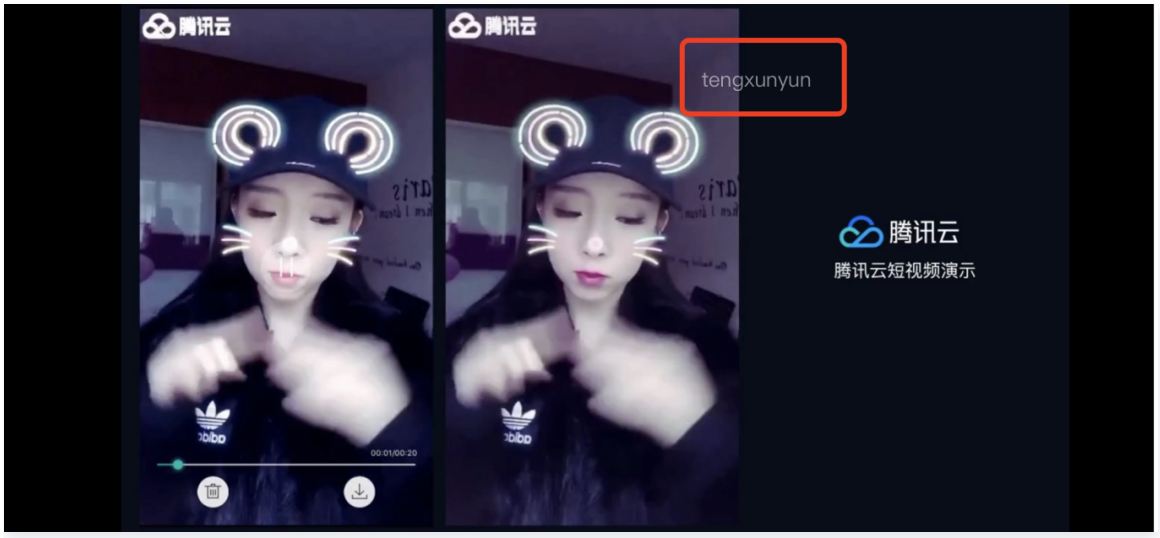

```
//步骤1：创建试看 model
TXVipWatchModel *model = [[TXVipWatchModel alloc] init];
model.tipTtitle = @" 可试看15秒，开通 VIP 观看完整视频";
model.canWatchTime = 15;
//步骤2：设置试看 model
self.playerView.vipWatchModel = model;
//步骤3：调用方法展示试看功能
[self.playerView showVipTipView];
```

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，单位为秒。

7. 动态水印

播放器组件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 playWithModelNeedLicence:playermodel];
```

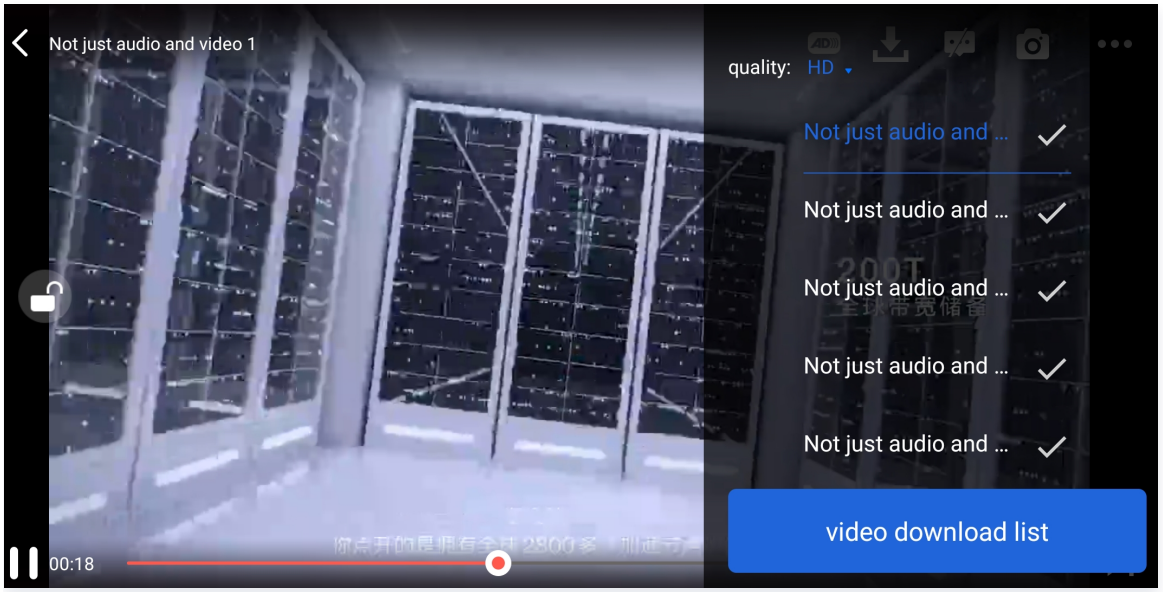
DynamicWaterModel 类参数说明：

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

8. 视频下载

支持用户在有网络的条件下缓存视频，随后在无网络的环境下观看；同时离线缓存的视频仅可在客户端内观看，不可被下载至本地，可有效防止下载视频的非法传播，保护视频安全。 您可在**腾讯云视立方 App > 播放器 > 超级播放器** > 离线缓存（全屏）演示视频

中，使用全屏观看模式后体验。



- **VideoCacheView（缓存选择列表视图）**
  用于选择下载对应清晰度的视频。左上角选择清晰度后，再单击要下载的视频选项，出现对勾后，代表开始了下载。单击下方的 **video download list** 按钮后会跳转到 VideoDownloadListView 所在的 Activity。

```
// 步骤1：初始化缓存选择列表视图//@property (nonatomic, strong) VideoCacheView
*cacheView;_cacheView = [[VideoCacheView alloc]
initWithFrame:CGRectZero];_cacheView.hidden = YES;[self.playerView
addSubview:_cacheView];
// 步骤2：设置正在播放的视频选项[_cacheView setVideoModels:_currentPlayVideoArray
currentPlayingModel:player.playerModel];
// video download list 按钮的单击事件- (UIButton *)viewCacheListBtn;
```

```
- (void)setVideoModels:(NSArray *)models currentPlayingModel:(SuperPlayerModel
*)currentModel;
```

接口参数说明：

参数名	类型	描述
models	NSArray	下载列表的视频数据模型
SuperPlayerModel	currentModel	当前在播放的视频数据模型

- **VideoCacheListView（视频下载列表）**
  显示所有正在下载的和下载完成视频的列表 View。
  单击时：
  - 如果正在下载，会暂停下载。
  - 如果暂停下载，会继续下载。
  - 如果下载完成，会跳转播放。

```
// 添加数据，数据从 TXVodDownloadManager#getDownloadMediaInfoList 接口获取到
NSArray<TXVodDownloadMediaInfo *> *array = [[[TXVodDownloadManager
shareInstance] getDownloadMediaInfoList] mutableCopy];
for (TXVodDownloadMediaInfo *info in array) {
    VideoCacheListModel *model = [[VideoCacheListModel alloc] init];
    model.mediaInfo = info;
    [self.videoCacheArray addObject:model];
}

// 列表项支持单击播放、长按删除等操作
- (void)longPress:(UILongPressGestureRecognizer *)longPress;  // 长按
```

- 下载后的视频支持无网络情况下进行播放，播放时请参考如下代码：

```
NSArray<TXVodDownloadMediaInfo *> *mediaInfoList = [[TXVodDownloadManager
shareInstance] getDownloadMediaInfoList];
TXVodDownloadMediaInfo *mediaInfo = [mediaInfoList firstObject];
SuperPlayerUrl *superPlayerUrl = [[SuperPlayerUrl alloc] init];
superPlayerUrl.title = @"*********";
superPlayerUrl.url = mediaInfo.playpath;
NSArray<SuperPlayerUrl *> *multiVideoURLs = @[superPlayerUrl];
SuperPlayerModel *playerModel = [[SuperPlayerModel alloc] init];
playerModel.multiVideoURLs = multiVideoURLs;
[self.playerView playWithModelNeedLicence:playerModel];
```

注意：
视频文件下载无网络播放时，一定要通过获取下载列表并通过下载列表视频对象 `TXVodDownloadMediaInfo` 的 `PlayPath` 进行播放，切勿直接保存 PlayPath 对象。

9. 雪碧图和打点信息

打点信息

支持在进度条关键位置添加文字介绍，用户单击后可显示打点位置的文字信息，以快速了解当前位置的视频信息。单击视频信息后，可以 seek 到打点信息位置。

您可在**腾讯云视立方 App > 播放器 > 超级播放器** > 腾讯云视频中，使用全屏观看模式后体验。

雪碧图

支持用户在拖拽进度条或执行快进操作时查看视频缩略图，以快速了解指定进度的视频内容。缩略图预览基于视频雪碧图实现，您可以在云点播控制台中生成视频文件雪碧图，或直接生成雪碧图文件。 您可在**腾讯云视立方 App > 播放器 > 超级播放器** > 腾讯云视频中，使用全屏观看模式后体验。


```
// 步骤1：通过 playWithModelNeedLicence 播放器视频，才能在 onPlayEvent 回调中获取到雪碧图和打点信息数据[self.playerView playWithModelNeedLicence:playerModel];

// 步骤2: playWithModelNeedLicence 在 VOD_PLAY_EVT_GET_PLAYINFO_SUCC 回调事件中取得关键帧和雪碧图信息NSString *imageSpriteVtt =
[paramobjectForKey:VOD_PLAY_EVENT_IMAGESPRIT_WEBVTTURL]?:@"";
NSArray<NSString *> *imageSpriteList =
[paramobjectForKey:VOD_PLAY_EVENT_IMAGESPRIT_IMAGEURL_LIST];
NSArray<NSURL *> *imageURLs = [self convertImageSpriteList:imageSpriteList];
[self.imageSprite setVTTUrl:[NSURL URLWithString:imageSpriteVtt]
imageUrls:imageURLs];
```

```
// 步骤3: 将拿到的打点信息和雪碧图，并显示到界面上
if (self.isFullScreen) {
    thumbnail = [self.imageSprite getThumbnail:draggedTime];
}if (thumbnail) {
    [self.fastView showThumbnail:thumbnail withText:timeStr];
}
```

10. 外挂字幕

> **注意：**
> 外挂字幕依赖播放器高级版本 SDK 且 SDK 需要11.3版本以上才支持。

目前支持 SRT 和 VTT 这两种格式的字幕。用法如下：

步骤1：添加外挂字幕。

往 `SuperPlayerModel#subtitlesArray` 传入外挂字幕类别字段。

```
// 传入 字幕url， 字幕名称， 字幕类型
SuperPlayerSubtitles *subtitleModel = [[SuperPlayerSubtitles alloc] init];
subtitleModel.subtitlesUrl = @"https://mediacloud-76607.gzc.vod.tencent-
cloud.com/DemoResource/TED-CN.srt";
subtitleModel.subtitlesName = @"ex-cn-srt";
subtitleModel.subtitlesType = 0;
[subtitlesArray addObject:subtitleModel];

// 播放
[self.playerView playWithModelNeedLicence:model];
```

步骤2：播放后切换字幕。

```
// 开始播放视频后，选中添加的外挂字幕
- (void)controlViewSwitch:(UIView *)controlView withSubtitlesInfo:(TXTrackInfo
*)info preSubtitlesInfo:(TXTrackInfo *)preInfo {
```

```
    if (info.trackIndex == -1) {
        [self.vodPlayer deselectTrack:preInfo.trackIndex];
        self->_lastSubtitleIndex = -1;
    } else {
        if (preInfo.trackIndex != -1) {
            // 其它字幕不需要的话， 进行deselectTrack
            [self.vodPlayer deselectTrack:preInfo.trackIndex];
        }
        // 选中字幕
        [self.vodPlayer selectTrack:info.trackIndex];
        self->_lastSubtitleIndex = info.trackIndex;
    }
}
```

步骤3：配置字幕样式。

字幕样式支持在播放前或者播放过程中配置。

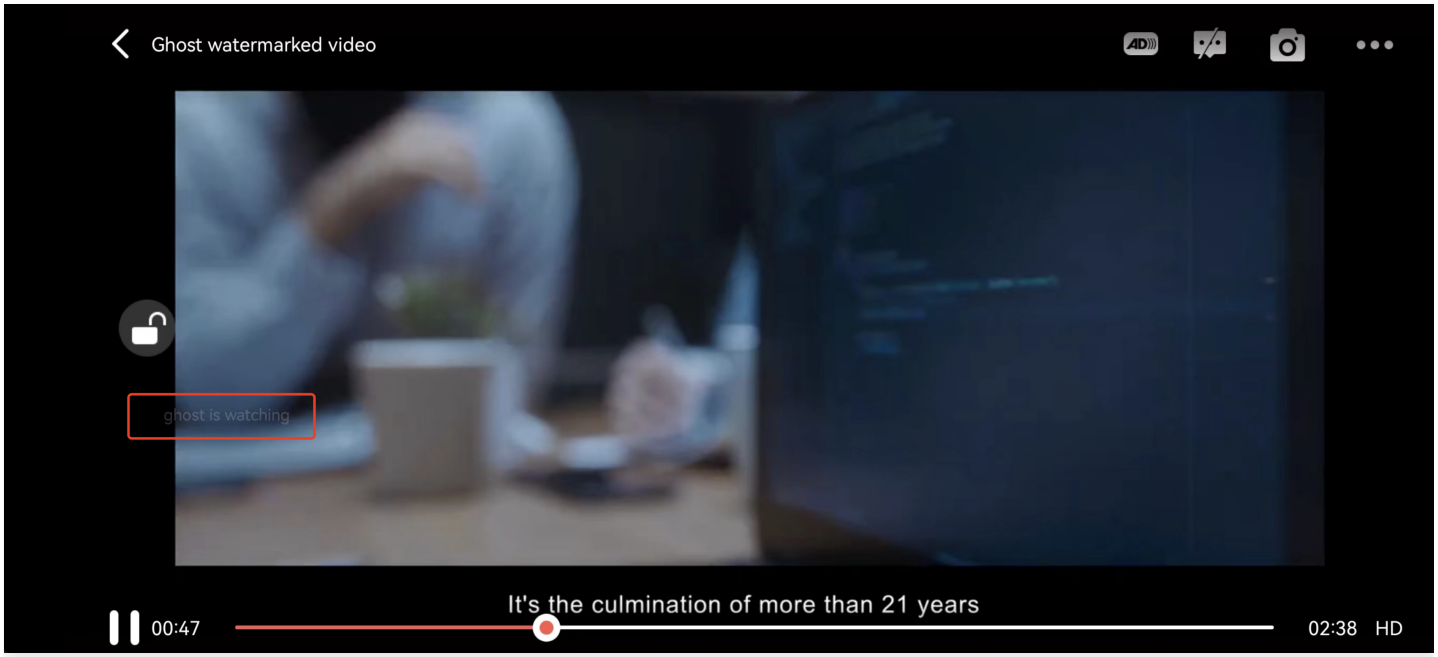
```
TXPlayerSubtitleRenderModel *model = [[TXPlayerSubtitleRenderModel alloc] init];
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.isBondFontStyle = NO;  // 设置字幕字体是否为粗体
model.fontColor = 0xFF000000; // 设置字幕字体颜色，默认白色不透明
[_txVodPlayer setSubtitleStyle:model];
```

11. 幽灵水印

幽灵水印内容在播放器签名中填写，经云点播后台，最终展示到播放端上，整个传输链路过程由云端和播放端共同协作，确保水印的安全。在播放器签名中 配置幽灵水印教程 。幽灵水印仅在视频上出现一段很短的时间，这种闪现对视频的观看影响很微小。每次水印出现的画面位置都不固定，杜绝了他人遮挡水印的企图。效果如下图所示，在视频开始播放时，就会出现一次水印，然后消失。等到下一次再出现，再消失。

幽灵水印的内容在收到播放器的 `VOD_PLAY_EVT_GET_PLAYINFO_SUCC` 事件后， 通过 `[param objectForKey:@"EVT_KEY_WATER_MARK_TEXT"]` 获取。

注意： 播放器 11.6 版本开始支持。




```
// 步骤 1： 配置支持幽灵水印的 FileId 播放视频
SuperPlayerModel *model = [[SuperPlayerModel alloc] init];
model.appId = 1500006438;
model.videoId = [[SuperPlayerVideoId alloc] init];
model.videoId.fileId = @"387702307847129127";
model.videoId.pSign =
@"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwNjQzOCwiZmlsZUlkIjoiMzg3N
zAyMzA3ODQ3MTI5MTI3IiwiY29udGVudEluZm8iOnsiYXVkaW9WaWRlb1R5cGUiOiJSYXdBZGFwdGl2ZSIsI
nJhd0FkYXB0aXZlRGVmaW5pdGlvbiI6MTB9LCJjdXJyZW50VGltZVN0YW1wIjoxNjg2ODgzMzYwLCJnaG9zd
FdhdGVybWFya0luZm8iOnsidGV4dCI6Imdob3N0IGlzIHdhdGNoaW5nIn19.0G2o4P5xVZ7zFlFUgBLntfX0
3iGxK9ntD_AONClUUno";
[_playerView playWithModelNeedLicence:model];

// 步骤 2： 在 SuperPlayerView 收到 PLAY_EVT_GET_PLAYINFO_SUCC事件后，获取幽灵水印内容展示
- (void)onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary
*)param {
    dispatch_async(dispatch_get_main_queue(), ^{
        if (EvtID == PLAY_EVT_PLAY_EVT_GET_PLAYINFO_SUCCPLAY_PROGRESS) {
            NSString *ghostWaterText = [param
objectForKey:@"EVT_KEY_WATER_MARK_TEXT"];
            if (ghostWaterText && ghostWaterText.length > 0) {
                DynamicWaterModel *model = [[DynamicWaterModel alloc] init];
                model.showType = ghost;
                model.duration = self.playerModel.duration;
                model.dynamicWatermarkTip = ghostWaterText;
                model.textFont = 30;
                model.textColor = [UIColor redColor];
                if (![self.subviews containsObject:self.watermarkView]) {
```

```
                    [self addSubview:self.watermarkView];
                    [self.watermarkView mas_makeConstraints:^(MASConstraintMaker
*make) {
                        make.edges.equalTo(self);
                    }];
                }
                [self.watermarkView setDynamicWaterModel:model];
            }
        }
    };
}
```

Demo 体验

更多功能和调试 Demo 体验，请 单击这里 。

Android 接入指引

最近更新时间：2025-06-03 14:16:51

产品概述

腾讯云视立方 Android 播放器组件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 播放器组件。
2. 如何创建和使用播放器。
3. 播放器推出短视频组件、画中画2.0、VR 播放等高级组件，功能介绍和使用指引请参见 移动端高级功能 。

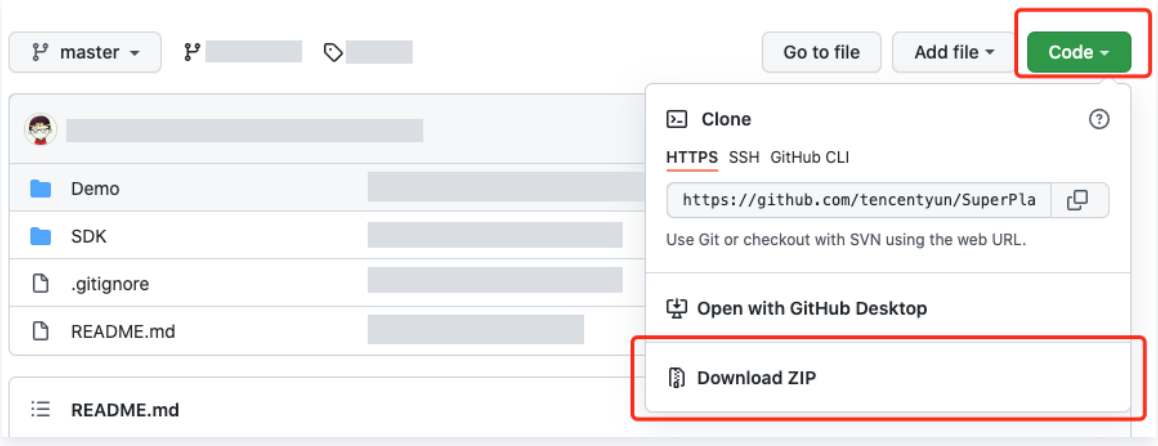
集成准备

步骤1：项目下载

腾讯云视立方 Android 播放器组件的项目地址是 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 播放器组件项目工程。

下载播放器组件 ZIP 包

您可以直接下载播放器组件 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. 执行下面的命令把播放器组件的组件工程代码 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)	播放器组件 Demo 工程，导入到 Android Studio 后可以直接运行
app	主界面入口
superplayerkit	播放器组件（SuperPlayerView），具备播放、暂停、手势控制等常见功能
superplayerdemo	播放器组件 Demo 代码
common	工具类模块
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	播放器组件使用文档

步骤2：集成指引

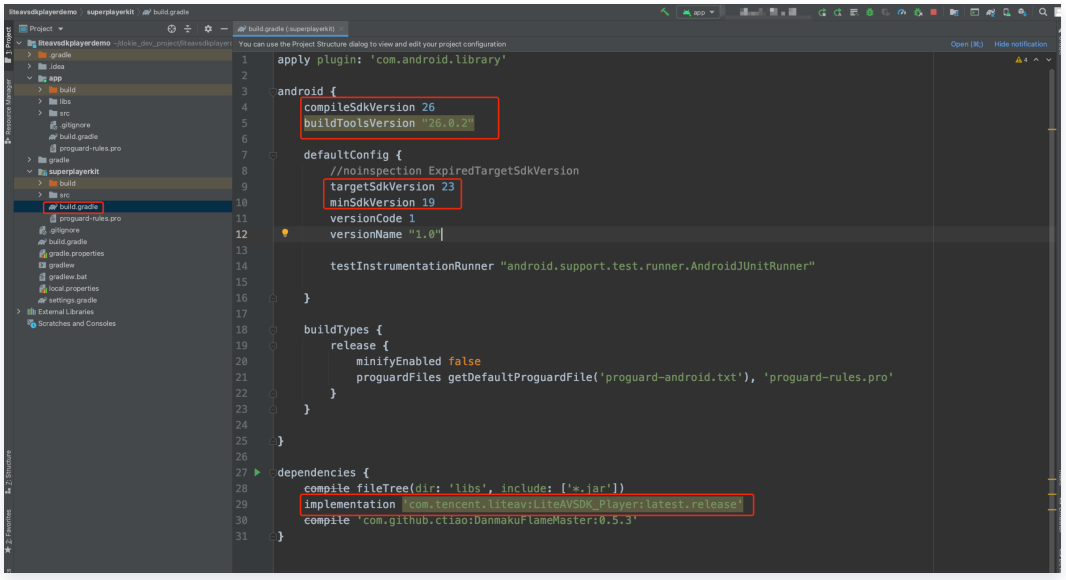
本步骤可指导您如何集成播放器，您可选择使用 Gradle 自动加载的方式，手动下载 aar 再将其导入到您当前的工程或导入 jar 和 so 库的方式集成项目。

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` 。

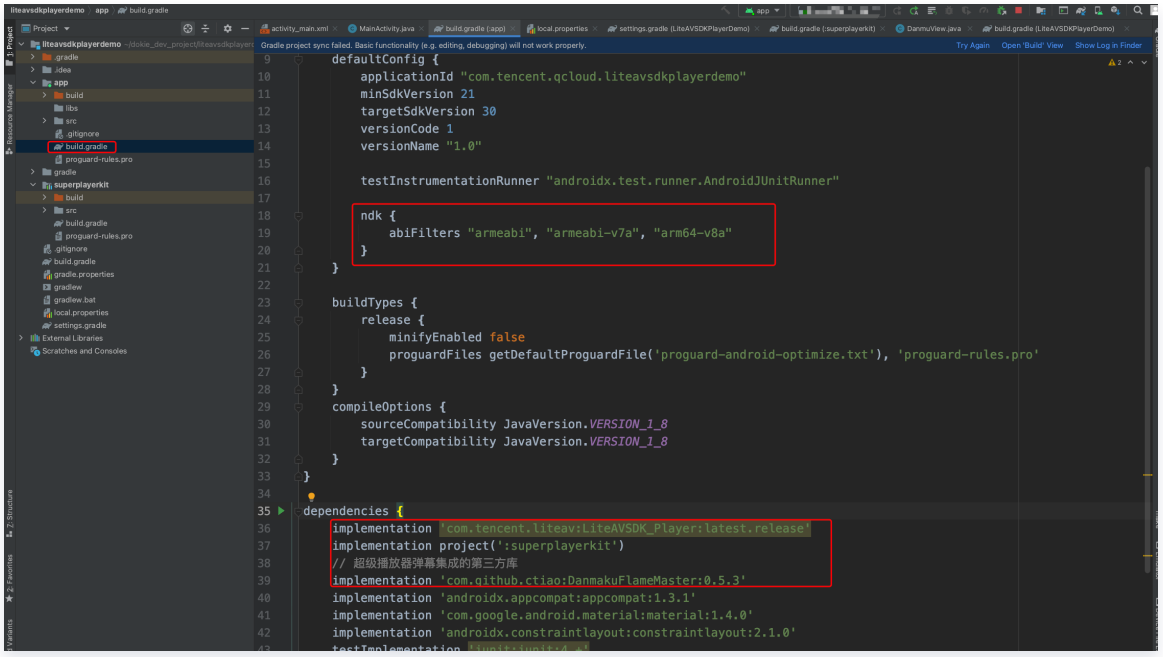
```
include ':superplayerkit'
```

   - 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：



3.1 在 dependencies 中添加 LiteAVSDK_Player 的依赖。

```
dependencies {
  implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
  implementation project(':superplayerkit')
}
```

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'
}
```

3.2 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**同时满足**以下两点，可以不需要该功能的 so 文件，达到减少安装包的体积。

- 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）。
- 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。

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"
}
```

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

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

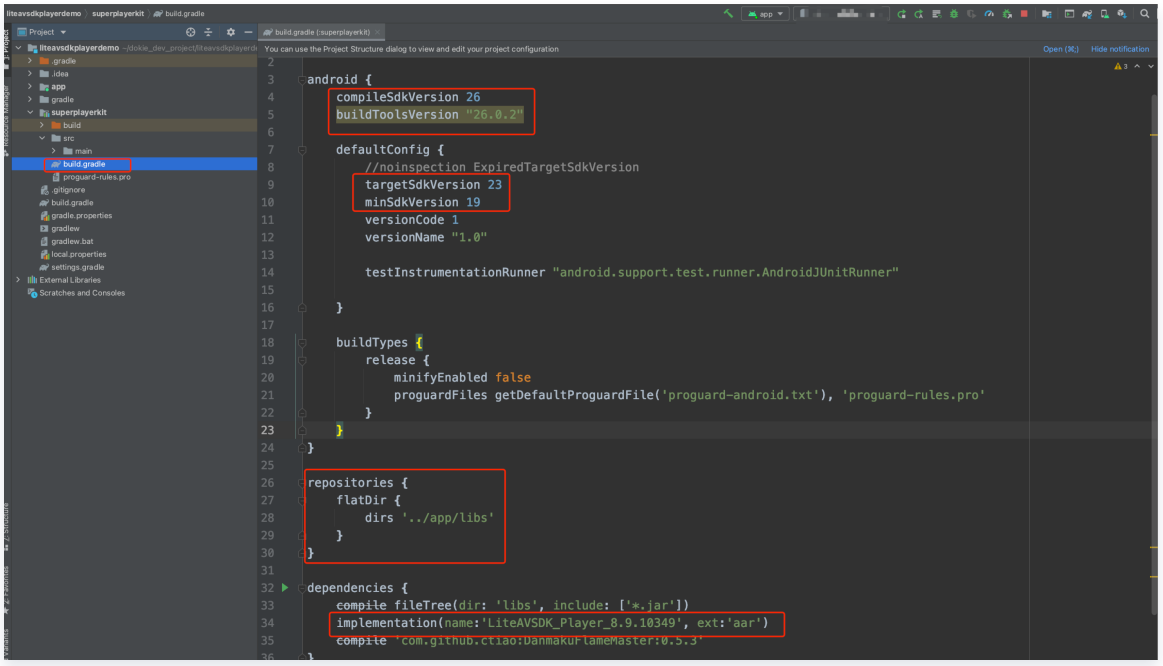
4. 单击  Sync Now 按钮同步 SDK，如果您的网络连接 mavenCentral 没有问题，很快 SDK 就会自动下载集成到工程里。

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')
// 播放器组件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"
}
```

如果同时**满足**以下两点，可以不需要该功能的 so 文件，达到减少安装包的体积。

- 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）。
- 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。

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：

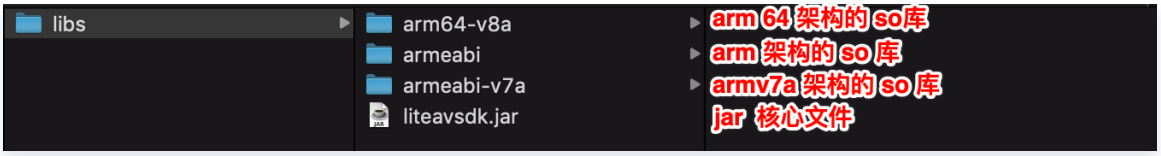
```
packagingOptions {
 exclude "lib/armeabi/libijkhlscache-master.so"
 exclude "lib/armeabi-v7a/libijkhlscache-master.so"
 exclude "lib/arm64-v8a/libijkhlscache-master.so"
}
```

8. 单击 Sync Now 按钮同步 SDK，完成播放器组件的集成工作。

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：

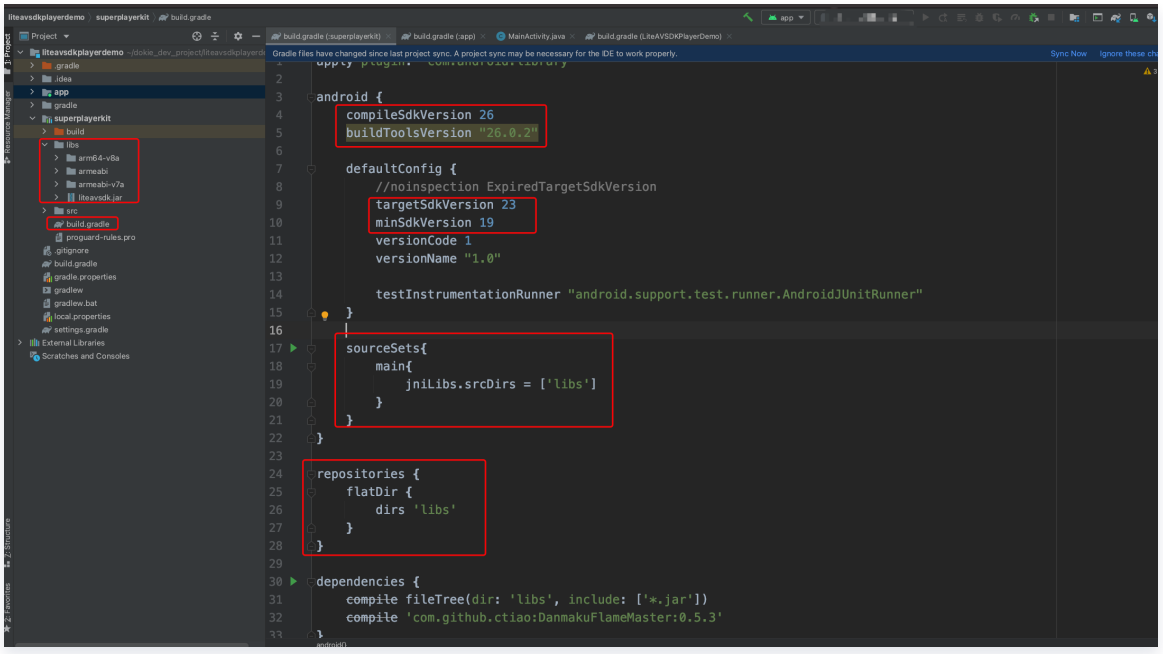
1. 下载 SDK + Demo 开发包，项目地址为 Android，下载完成后进行解压。在 SDK 目录找到 SDK/LiteAVSDK_Player_XXX.zip（其中 XXX 为版本号），解压得到 libs 目录，里面包含 jar 文件和 so 文件夹，文件清单如下：



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. 把上面步骤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：




```
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 播放器组件的集成工作。

您已经完成了腾讯云视立方 Android 播放器组件项目集成的步骤。

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" />
```

网络安全配置允许 App 发送 http 请求

出于安全考虑，从 Android P 开始，Google 要求 App 的请求都使用加密链接。播放器 SDK 会启动一个 localsever 代理 http 请求，如果您的应用 **targetSdkVersion 大于或等于28**，可以通过 网络安全配置 来开启允许向127.0.0.1发送 http 请求。否则播放时将出现 "java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted" 错误，导致无法播放视频。配置步骤如下：

1. 在项目新建 res/xml/network_security_config.xml 文件，设置网络安全性配置。

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
</network-security-config>
```

2. 在 AndroidManifest.xml 文件下的 application 标签增加以下属性。

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
                    ... >
        ...
    </application>
</manifest>
```

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 播放器组件 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：

   ```
   <!-- 播放器组件-->
   <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 播放器 License 获取相关授权。获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

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 播放器组件可用于直播和点播两种播放场景，具体如下：

- 点播播放：播放器组件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：播放器组件可使用 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.playWithModelNeedLicence(model);
```

通过 FileID 播放（点播）

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，或者使用播放器组件签名 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 用播放器组件播放视频，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](#url) 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 = new SuperPlayerModel();
model.appId = 1400329071;// 配置 AppId
```

```
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "5285890799710173650"; // 配置 FileId
// psign 即播放器签名，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.eyJhcHBJZCI6MTQwMDMyOTA3MSwiZmlsZ
UlkIjoiNTI4NTg5MDc5OTcxMDE3MzY1MCIsImN1cnJlbnRUaW1lU3RhbXAiOjEsImV4cGly
ZVRpbWVTdGFtcCI6MjE0NzQ4MzY0NywidXJsQWNjZXNzSW5mbyI6eyJ0IjoiN2ZmZmZmZmY
ifSwiZHJtTGljZW5zZUluZm8iOnsiZXhwaXJlVGltZVN0YW1wIjoyMTQ3NDgzNjQ3fX0.yJ
xpnQ2Evp5KZQFfuBBK05BoPpQAzYAWo6liXws-LzU";
mSuperPlayerView.playWithModelNeedLicence(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 播放器组件创建、播放视频和退出播放的能力集成。

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，播放器组件支持的功能可参见 能力清单。

1、全屏播放

播放器组件支持全屏播放，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

播放器组件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
  }
});
```

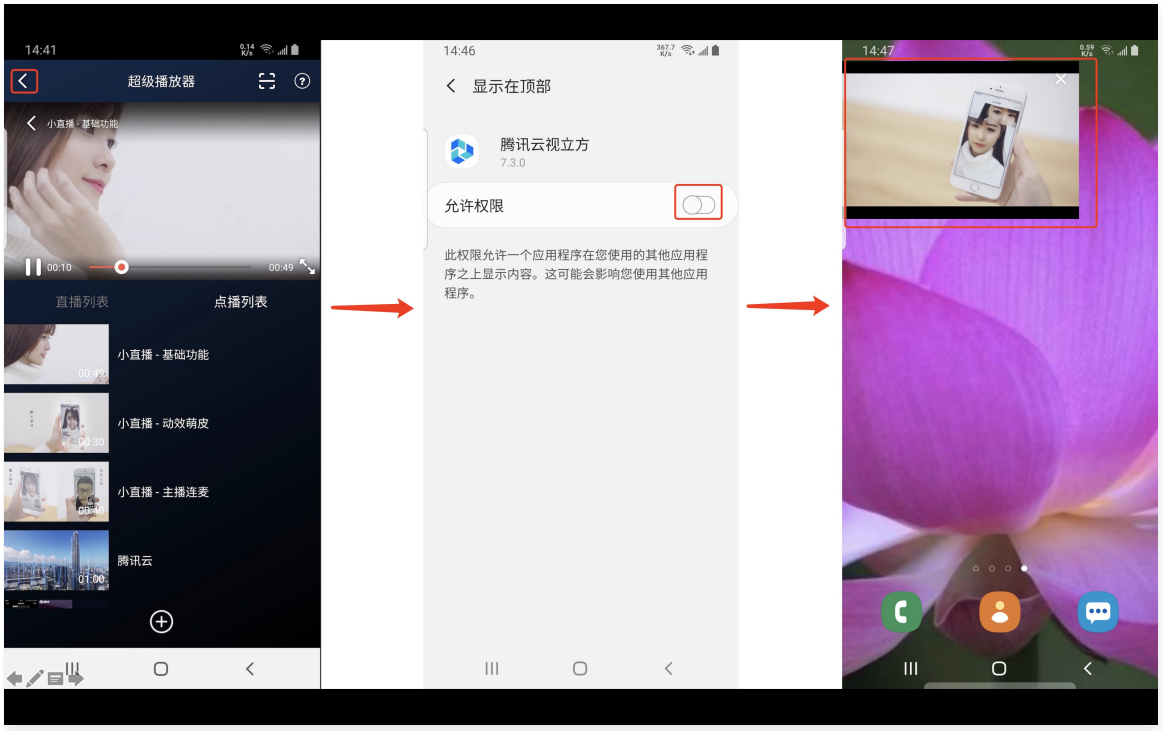
清晰度切换

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

播放器组件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

播放器组件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。

- 当播放器组件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；
- 当播放器组件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 = "https://qcloudimg.tencent-
cloud.cn/raw/946152ef79a6034786eb868f425b5f85.png"
mSuperPlayerView.playWithModelNeedLicence(model);
```

4、视频列表轮播

播放器组件支持视频列表轮播，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.playWithModelListNeedLicence(list, true, 0);
```

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

接口参数说明

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

5、画中画功能

从 Android 8.0（API 级别 26）开始，Android 允许以画中画 (PiP) 模式启动 Activity。

如果您需要启用或者禁用画中画，只需更改 SuperPlayerGlobalConfig 中 enablePIP 的值。
要将画中画添加到您的应用中，需要对支持画中画的 Activity 在 AndroidManifest 中加上以下属性。

```
<activity>
    android:name=".demo.SuperPlayerActivity"
    android:resizeableActivity="true"
    android:supportsPictureInPicture="true"
    android:documentLaunchMode="intoExisting"
    android:excludeFromRecents="true"

android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize|scre
enLayout">
```

```
</activity>
```

同时对支持画中画的 Activity 的生命周期需要参照 SuperPlayerActivity 做下特殊处理。开启画中画，在 SuperPlayerView 中使用 PictureInPictureHelper。

```
PictureInPictureHelper mPictureInPictureHelper = new
PictureInPictureHelper(mContext);
mPictureInPictureHelper.setListener(this);
mPictureInPictureHelper.enterPictureInPictureMode(getPlayerState(),mTXCloudVideoView
);
```

在退出时需要在 SuperPlayerView 中释放。

```
mPictureInPictureHelper.release();
```

如果需要对画中画自定义按钮前移后移的时间间隔修改，只需要修改 PictureInPictureHelper 中 PIP_TIME_SHIFT_INTERVAL 的值。

6、视频试看

播放器组件支持视频试看功能，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.playWithModelNeedLicence(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);
```

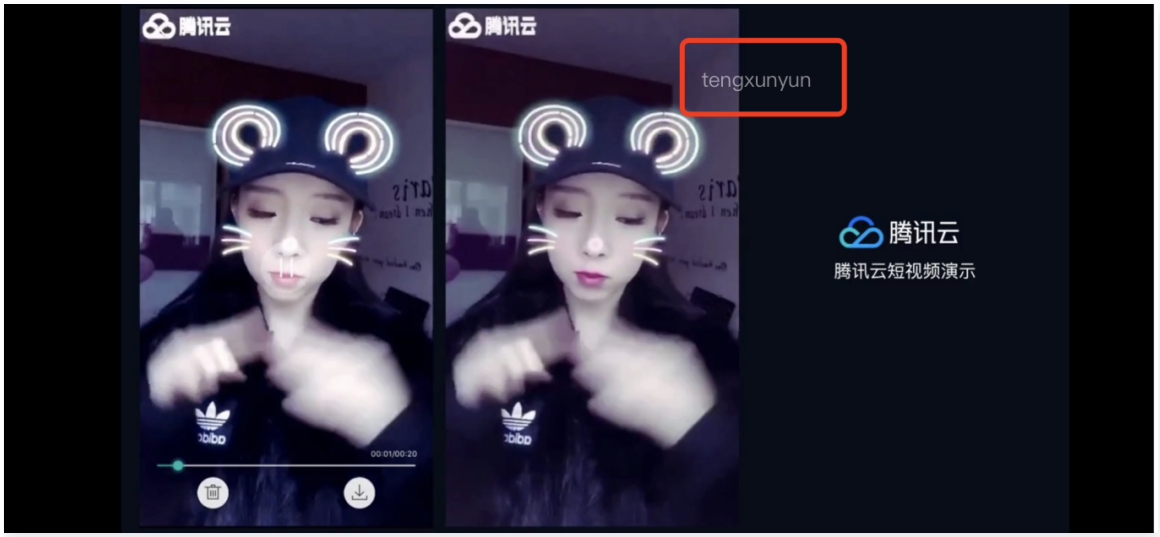
```
public VipWatchModel(String tipStr, long canWatchTime)
```

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，单位为秒。

7、动态水印

播放器组件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.parseColor("#80FFFFFF"));
mode.dynamicWaterConfig = dynamicWaterConfig;
//步骤3：调用播放视频方法
mSuperPlayerView.playWithModelNeedLicence(mode);
```

方法二：

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

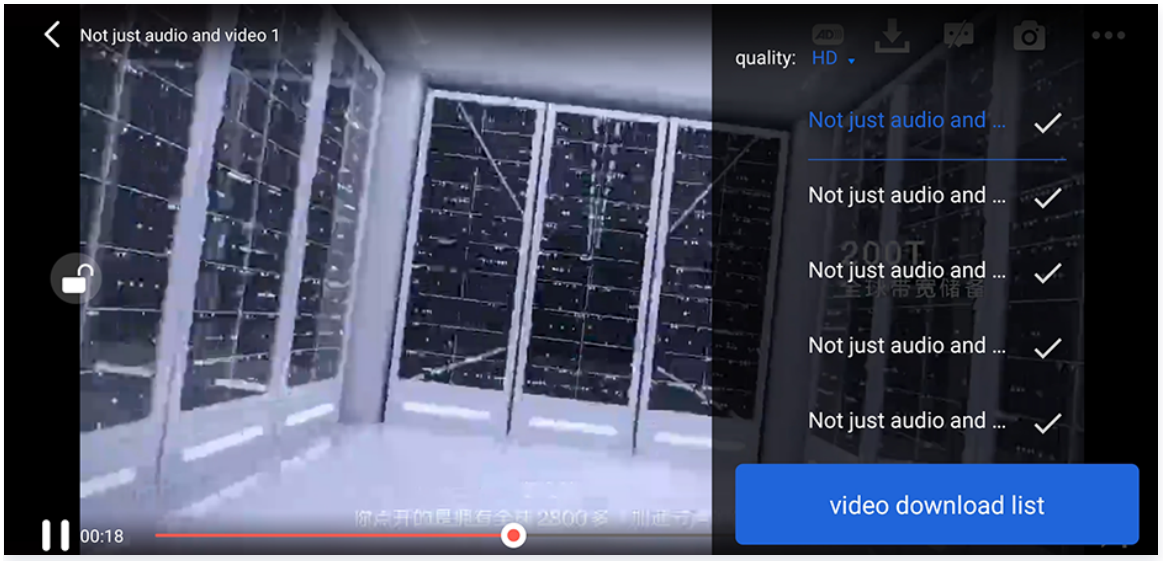
```
public DynamicWaterConfig(String dynamicWatermarkTip, int tipTextSize, int
tipTextColor)
```

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。

8、视频下载

支持用户在有网络的条件下缓存视频，随后在无网络的环境下观看；同时离线缓存的视频仅可在客户端内观看，不可被下载至本地，可有效防止下载视频的非法传播，保护视频安全。
您可在**腾讯云视立方 App > 播放器 > 超级播放器 > 离线缓存（全屏）**演示视频中，使用全屏观看模式后体验。



DownloadMenuListView（缓存选择列表视图），用于选择下载对应清晰度的视频。左上角选择清晰度后，再单击要下载的视频选项，出现对勾后，代表开始了下载。单击下方的 video download list 按钮后会跳转到 VideoDownloadListView 所在的 Activity。

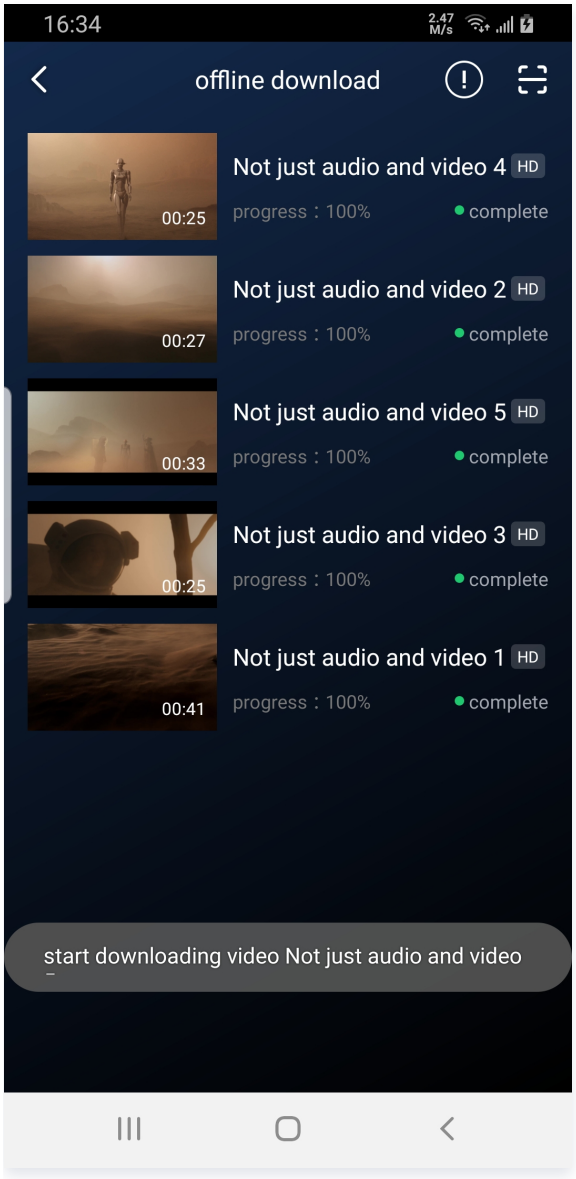
```
// 步骤1：初始化下载数据 参数见下方列表
DownloadMenuListView mDownloadMenuView =
findViewById(R.id.superplayer_cml_cache_menu);
mDownloadMenuView.initDownloadData(superPlayerModelList, mVideoQualityList,
mDefaultVideoQuality, "default");

// 步骤2：设置正在播放的视频选项
mDownloadMenuView.setCurrentPlayVideo(mSuperplayerModel);

// 步骤3：设置 video download list 按钮的点击事件
mDownloadMenuView.setOnCacheListClick(new OnClickListener() {
     @Override
     public void onClick(View v) {
       // 跳转到 VideoDownloadListView 所在的 Activity
       startActivity(DownloadMeduListActivity.this,VideoDownloadListActivity.class);
     }
});

// 步骤4:通过动画展示 view
mDownloadMenuView.show();
```

```
public void initDownloadData(List<SuperPlayerModel> superPlayerModelList,
                             List<VideoQuality> qualityList,
                             VideoQuality currentQuality,
                             String userName)
```

接口参数说明

参数名	类型	描述
superPlayerModelList	List<SuperPlayerModel>	下载的视频数据。
qualityList	List<VideoQuality>	视频清晰度数据。
currentQuality	VideoQuality	当前的视频清晰度。
userName	String	用户名。

VideoDownloadListView（视频下载列表），显示所有正在下载的和下载完成视频的列表 View。点击时，如果正在下载，会暂停下载；如果暂停下载，会继续下载；如果下载完成，会跳转播放。




```
// 步骤1：绑定控件
VideoDownloadListView mVideoDownloadListView =
findViewById(R.id.video_download_list_view);
```

```
//步骤2：添加数据
mVideoDownloadListView.addCacheVideo(mDataList, true);
```

接口参数说明

```
public void addCacheVideo(List<TXVodDownloadMediaInfo> mediaInfoList, boolean
isNeedClean);
```

参数名	类型	描述
mediaInfoList	List<TXVodDownloadMediaInfo>	添加的视频数据类型。
isNeedClean	boolean	是否清除之前的数据

9、雪碧图和打点信息

打点信息

支持在进度条关键位置添加文字介绍，用户点击后可显示打点位置的文字信息，以快速了解当前位置的视频信息。点击视频信息后，可以 seek 到打点信息位置。
您可在腾讯云视立方 App > 播放器 > 超级播放器 > 腾讯云视频中，使用全屏观看模式后体验。

雪碧图

支持用户在拖拽进度条或执行快进操作时查看视频缩略图，以快速了解指定进度的视频内容。缩略图预览基于视频雪碧图实现，您可以在云点播控制台中生成视频文件雪碧图，或直接生成雪碧图文件。

您可在**腾讯云视立方 App > 播放器 > 超级播放器 > 腾讯云**视频中，使用全屏观看模式后体验。


```
// 步骤1：播放视频 superplayerModel的url 变量需要为空，且 videoId 不为空，这样才会通过 PlayWithField 播放，才能在 onPlayEvent 回调中获取到打点信息和雪碧图数据
mSuperplayerView.play(superplayerModel);

// 步骤2: PlayWithFileId 播放时候 在 VOD_PLAY_EVT_GET_PLAYINFO_SUCC 回调事件中取得打点信息和雪碧图信息
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    switch (event) {
        case TXVodConstants.VOD_PLAY_EVT_GET_PLAYINFO_SUCC:

            // 获取 雪碧图 图片链接 URL
            playImageSpriteInfo.imageUrls =
param.getStringArrayList(TXVodConstants.EVT_IMAGESPRIT_IMAGEURL_LIST);
            // 获取 雪碧图 web vtt 描述文件下载 URL
            playImageSpriteInfo.webVttUrl =
param.getString(TXVodConstants.EVT_IMAGESPRIT_WEBVTTURL);
            // 获取 打点信息
           ArrayList<String> keyFrameContentList =

param.getStringArrayList(TXVodConstants.EVT_KEY_FRAME_CONTENT_LIST);
            // 获取 打点信息时间信息
            float[] keyFrameTimeArray =
param.getFloatArray(TXVodConstants.EVT_KEY_FRAME_TIME_LIST);

            // 构建 打点信息数据列表
            if (keyFrameContentList != null && keyFrameTimeArray != null
                    && keyFrameContentList.size() == keyFrameTimeArray.length) {
                for (int i = 0; i < keyFrameContentList.size(); i++) {
                    PlayKeyFrameDescInfo frameDescInfo = new PlayKeyFrameDescInfo();
                    frameDescInfo.content = keyFrameContentList.get(i);
                    frameDescInfo.time = keyFrameTimeArray[i];
                    mKeyFrameDescInfoList.add(frameDescInfo);
```

```
                }
            }
            break;
        default:
            break;
    }
}

// 步骤3：将拿到的打点信息和雪碧图信息通过 updateVideoImageSpriteAndKeyFrame 方法赋值给对应的
view。
// 雪碧图的 view 对应 VideoProgressLayout 组件中 mIvThumbnail。
// 打点信息的 view 对应 PointSeekBar 组件中的 TCPointView。
updateVideoImageSpriteAndKeyFrame(playImageSpriteInfo,keyFrameDescInfoList);
```

10、外挂字幕

> **注意：**
> 外挂字幕依赖播放器高级版本SDK 且 SDK 需要11.3 版本以上才支持。

目前支持 SRT 和 VTT 这两种格式的字幕。用法如下：

步骤1：添加外挂字幕。

往 `SuperPlayerModel#subtitleSourceModelList` 传入外挂字幕类别字段。

```
// 传入 字幕url， 字幕名称， 字幕类型
SubtitleSourceModel subtitleSourceModel = new SubtitleSourceModel();
subtitleSourceModel.name = "ex-cn-srt";
subtitleSourceModel.url = "https://mediacloud-76607.gzc.vod.tencent-
cloud.com/DemoResource/TED-CN.srt";
subtitleSourceModel.mimeType = TXVodConstants.VOD_PLAY_MIMETYPE_TEXT_SRT;
model.subtitleSourceModelList.add(subtitleSourceModel);


// 播放
```

```
mSuperPlayerView.playWithModelNeedLicence(model);
```

步骤2：播放后切换字幕。

```
// 开始播放视频后，选中添加的外挂字幕
public void onClickSubTitleItem(TXTrackInfo clickInfo) {
    List<TXTrackInfo> subtitleTrackInfoList = mVodPlayer.getSubtitleTrackInfo();
    for (TXTrackInfo trackInfo : subtitleTrackInfoList) {
        if (trackInfo.trackIndex == clickInfo.trackIndex) {
            // 选中字幕
            mVodPlayer.selectTrack(trackInfo.trackIndex);
            mSelectedSubtitleTrackInfo = trackInfo;
        } else {
            // 其它字幕不需要的话， 进行deselectTrack
            mVodPlayer.deselectTrack(trackInfo.trackIndex);
        }
    }
}
```

步骤3：配置字幕样式。

字幕样式支持在播放前或者播放过程中配置。

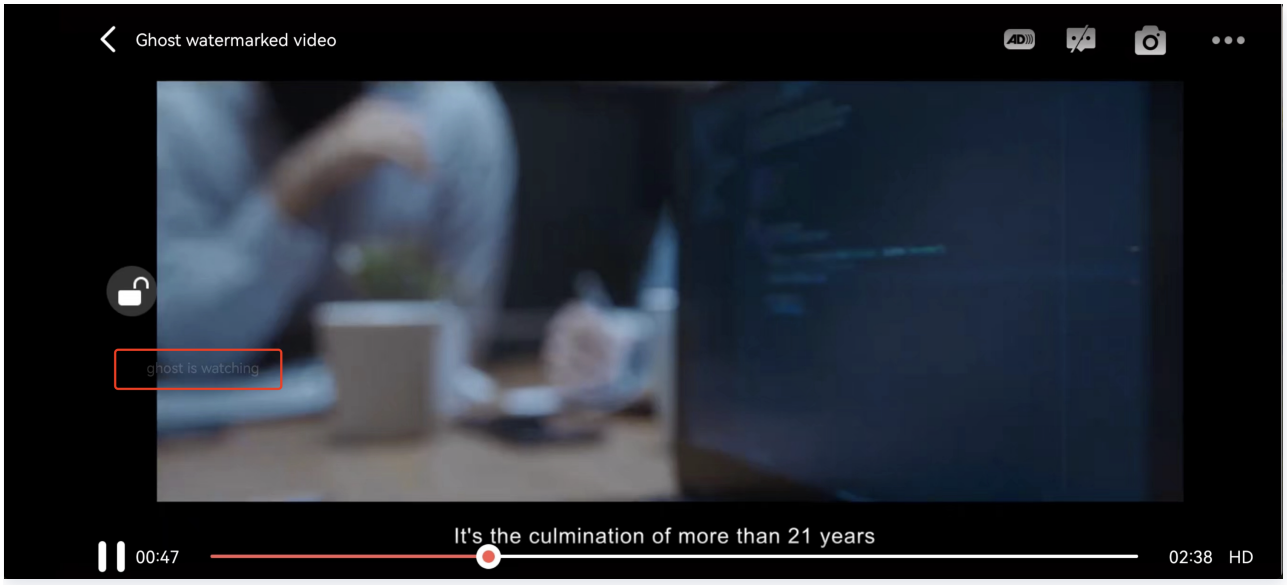
```
TXSubtitleRenderModel model = new TXSubtitleRenderModel();
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.fontColor = 0xFFFFFFFF; // 设置字幕字体颜色，默认白色不透明
model.isBondFontStyle = false;  // 设置字幕字体是否为粗体
mVodPlayer.setSubtitleStyle(model);
```

11、幽灵水印

幽灵水印内容在播放器签名中填写，经云点播后台，最终展示到播放端上，整个传输链路过程由云端和播放端共同协作，确保水印的安全。在播放器签名中 配置幽灵水印教程 。幽灵水印仅在视频上出现一段很短的时间，这种闪现对视频的观看影响很微小。每次水印出现的画面位置都不固定，杜绝了他人遮挡水印的企图。效果如下图所示，在视频开始播放时，就会出现一次水印，然后消失。等到下一次再出现，再消失。

幽灵水印的内容在收到播放器的 `TXVodConstants#VOD_PLAY_EVT_GET_PLAYINFO_SUCC` **事件后， 通过** `param.getString(TXVodConstants.EVT_KEY_WATER_MARK_TEXT)` **获取。**

注意： 播放器 11.6 版本开始支持。




```
// 步骤 1： 配置支持幽灵水印的 FileId 播放视频
SuperPlayerModel model = new SuperPlayerModel();
model.appId = 1500006438;
model.videoId.fileId = "387702307847129127";
model.videoId.pSign = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDA"
                    + "wNjQzOCwiZmlsZUlkIjoiMzg3NzAyMzA3ODQ3MTI5MTI3IiwiY29udG"
                    + "VudEluZm8iOnsiYXVkaW9WaWRlb1R5cGUiOiJSYXdBZGFwdGl2ZSIsIn"
                    + "Jhd0FkYXB0aXZlRGVmaW5pdGlvbiI6MTB9LCJjdXJyZW50VGltZVN0YW1w"
                    + "IjoxNjg2ODgzMzYwLCJnaG9zdFdhdGVybWFya0luZm8iOnsidGV4dCI6I"
                    + "mdob3N0IGlzIHdhdGNoaW5nIn19.0G2o4P5xVZ7zF"
                    + "lFUgBLntfX03iGxK9ntD_AONClUUno";
mSuperPlayerView.playWithModelNeedLicence(model);

// 步骤 2： 在 SuperPlayerView#onRcvWaterMark 中收到幽灵水印内容回调后，展示幽灵水印
public void onRcvWaterMark(String text, long duration) {
    if (!TextUtils.isEmpty(text)) {
        DynamicWaterConfig dynamicWaterConfig = new DynamicWaterConfig(text, 30,
Color.parseColor("#30FFFFFF"));
        dynamicWaterConfig.durationInSecond = duration;
        dynamicWaterConfig.setShowType(DynamicWaterConfig.GHOST_RUNNING);
        setDynamicWatermarkConfig(dynamicWaterConfig);
    }
}
```

Demo 体验

更多功能和调试 Demo 体验，请 单击这里 。

Flutter 接入指引

最近更新时间：2025-04-01 15:13:32

SDK 下载

腾讯云视立方 Flutter 播放器 SDK 的地址是 Player Flutter 。

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 。

通过本文您可以学会

- 如何集成腾讯云视立方 Flutter 播放器 SDK。
- 如何使用播放器组件进行点播播放。

播放器组件简介

Flutter 播放器组件是基于 Flutter 播放器 SDK 的扩展，播放器组件对于点播播放器，集成了更多的功能，包括全屏切换、清晰度切换、
进度条、播放控制、封面标题展示等常用功能，并且相对于点播播放器使用起来更加方便，如果想更加方便快捷的集成 Flutter 视频播放能力，可以选择 Flutter 播放器组件使用。
支持功能列表：

- 全屏播放
- 播放过程中屏幕旋转自适应
- 自定义视频封面
- 清晰度切换
- 声音和亮度调节
- 倍速播放
- 硬件加速开启\关闭
- 画中画（PIP）（支持 Android 和 iOS 平台）
- 雪碧图和关键帧打点信息

更多功能正在逐步开发中。

集成指引

1. 将项目中 `superplayer_widget` 目录复制到自己的 flutter 工程下。
2. 在自己项目的配置文件 `pubspec.yaml` 下添加依赖。

分支集成

```
superplayer_widget:
    # 该路径根据superplayer_widget存放路径改变
    path: ../superplayer_widget
super_player:
  git:
```

```
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: main
# ref 可以根据自身项目需要，替换为对应的版本或分支。
```

pub 集成

```
superplayer_widget:
    # 该路径根据superplayer_widget存放路径改变
    path: ../superplayer_widget
# pub集成默认为 professional 版本，如果有其余版本需求，请使用分支集成方式
super_player: ^12.3.0
```

3. 修改 superplayer_widget 的 `superPlayer` 依赖。
   进入修改 superplayer_widget 的 `pubspec.yaml` 。
   将如下配置进行替换：

```
super_player:
  path: ../
```

   替换为：

```
super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: main
```

   ref 可以根据自身项目需要，替换为对应的版本或分支。
4. 由于目前播放器组件接入了国际化，需要在入口函数中添加国际化组件，如下示例：

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    localizationsDelegates: [
      SuperPlayerWidgetLocals.delegate,
      // ...... your app other delegate
    ],
    supportedLocales: [
      Locale.fromSubtags(languageCode: 'en'),
      Locale.fromSubtags(languageCode: 'zh'),
      // ....... other language
    ],
```

```
    // ..... your app other code
  );
}
```

5. 在需要使用到的页面，导入 `superplayer_widget` 的依赖包，如下所示：

```
import 'package:superplayer_widget/demo_superplayer_lib.dart';
```

6. 其余原生相关配置，可以参考 集成指引 。

SDK 集成

步骤1：申请播放器 License 和集成

集成播放器前，需要 注册腾讯云账户 ，注册成功后申请播放器 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);
```

步骤2：创建 controller

```
SuperPlayerController _controller = SuperPlayerController(context);
```

步骤3：配置播放器

```
FTXVodPlayConfig config = FTXVodPlayConfig();
// 如果不配置preferredResolution，则在播放多码率视频的时候优先播放720 * 1280分辨率的码率
config.preferredResolution = 720 * 1280;
_controller.setPlayConfig(config);
```

FTXVodPlayConfig 中的详细配置可参考 Flutter 点播播放器的配置播放器接口。

步骤4：设置监听事件

```
_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);
  }
});
```

步骤5：添加布局

```
Widget _getPlayArea() {
    return Container(
    height: 220,
    child: SuperPlayerView(_controller),
  );
}
```

步骤6：添加返回事件监听

添加返回事件监听，确保用户在触发返回事件的时候，如果播放器处于全屏等状态，可以优先退出全屏，再次触发才会退出页面。如果全屏播放状态下需要直接退出页面，可以不实现该监听。

```
  @override
Widget build(BuildContext context) {
  return WillPopScope(
      child: Container(
        decoration: BoxDecoration(
            image: DecorationImage(
              image: AssetImage("images/ic_new_vod_bg.png"),
              fit: BoxFit.cover,
            )),
        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();
```

```
}
```

步骤7：启动播放

通过 url 方式

```
SuperPlayerModel model = SuperPlayerModel();
model.videoURL =
"http://1400329073.vod2.myqcloud.com/d62d88a7vodtranscq1400329073/59c68fe7528589
0800381567412/adp.10.m3u8";
_controller.playWithModelNeedLicence(model);
```

通过 fileId 方式

```
SuperPlayerModel model = SuperPlayerModel();
model.appId = 1500005830;
model.videoId = new SuperPlayerVideoId();
model.videoId.fileId = "8602268011437356984";
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
model.videoId.pSign = "psignXXX"
_controller.playWithModelNeedLicence(model);
```

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 SuperPlayerViewEvent.onSuperPlayerError 事件。

步骤8：结束播放

结束播放时**记得调用 controller 的销毁方法**，尤其是在下次 startVodPlay 之前，否则可能会产生大量的内存泄露以及闪屏问题。也确保在退出页面的时候，能够结束视频播放。

```
  @override
void dispose() {
  // must invoke when page exit.
  _controller.releasePlayer();
  super.dispose();
}
```

播放器组件接口列表

1、视频播放

> **注意：**
> 10.7版本开始，startPlay 变更为 startVodPlay，需要通过 {@link SuperPlayerPlugin#setGlobalLicense} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和播放器 Licence 均可使用，若您暂未获取上述 Licence ，可 免费申请测试版 License 以正常播放，正式版 License 需 购买 。

说明

开始播放视频。

接口

```
_controller.playWithModelNeedLicence(model);
```

参数说明

- SuperPlayerModel

参数名	类型	描述
appId	int	应用 appId。fileId 播放必填。
videoURL	String	视频 url，url 播放必填。
multiVideoURLs	List<String>	多码率 url，多码率 url 播放必填。
defaultPlayIndex	int	默认播放码率序号，配合 multiVideoURLs 使用。
videoId	SuperPlayerVideoId	fileId 存储对象，以下会有详细介绍。
title	String	视频标题，用户可设置该字段来自定义标题，从而覆盖播放器内部从服务器请求的标题。
coverUrl	String	从腾讯服务器拉取的封面图片,该值会在 SuperVodDataLoader 中被自动赋值。
customeCoverUrl	String	自定义视频封面，该字段会被优先判断，可以通过定义该参数来实现自定义封面。
duration	int	视频时长，单位 秒。
videoDescription	String	视频描述。
videoMoreDescription	String	视频详细描述。
playAction	int	action 包括 PLAY_ACTION_AUTO_PLAY、PLAY_ACTION_MANUAL_PLAY和PLAY_ACTION_PRELOAD，以下对参数含义会有详细介绍。

- SuperPlayerVideoId

参数名	类型	描述
fileId	String	文件 id，必填。
psign	String	播放器签名，签名介绍和生成方式参见 自定义监控相关文档 。

- playAction

参数名	说明
PLAY_ACTION_AUTO_PLAY	调用 playWithModel 之后，会自动开始播放视频。
PLAY_ACTION_MANUAL_PLAY	调用 playWithModel 之后，需要手动播放，并且播放器实质上并未加载视频，只会显示封面图，相对于 PLAY_ACTION_PRELOAD 没有任何视频播放资源消耗。
PLAY_ACTION_PRELOAD	调用 playWithModel 之后，会显示封面图，不会开始播放视频，不过播放器实质上已经加载了视频，相对于 PLAY_ACTION_MANUAL_PLAY，起播速度会更快。

2、暂停播放

说明

暂停播放视频。

接口

```
_controller.pause();
```

3、继续播放

说明

继续播放视频。

接口

```
_controller.resume();
```

4、重新开始播放

说明

重新开始播放视频。

接口

```
_controller.reStart();
```

5、重置播放器

说明

重置播放器状态，并停止播放视频。

接口

```
_controller.resetPlayer();
```

6、释放播放器

说明

释放播放器资源，并停止播放，调用该方法之后，controller 将不可再复用。

接口

```
_controller.releasePlayer();
```

7、播放器返回事件

说明

触发播放器返回事件，该方法主要用于全屏播放模式下的返回判断和处理。

返回 true：执行了退出全屏等操作，消耗了返回事件；返回 false：未消耗事件。

接口

```
_controller.onBackPress();
```

8、切换清晰度

说明

实时切换当前正在播放的视频的清晰度。

接口

```
_controller.switchStream(videoQuality);
```

参数说明

videoQuality 在开始播放之后，一般可通过 _controller.currentQualiyList 和 _controller.currentQuality 来获取，前者为清晰度列表，后者为默认清晰度。**清晰度切换能力在播放器组件中已经集成，切换到全屏之后可点击右下角清晰度进行切换。**

参数名	类型	描述
index	int	清晰度序号。
bitrate	int	清晰度码率。
width	int	该清晰度下视频的宽度。
height	int	该清晰度下视频的高度。
name	String	清晰度简称。
title	String	用于显示的清晰度名称。

url	String	清晰度 url，用于多码率下的清晰度 url，非必填。

9、调整进度(seek)

说明

调整当前视频的播放进度。

接口

```
_controller.seek(progress);
```

参数说明

参数名	类型	描述
progress	double	需要调整到的时间，单位秒。

10、配置播放器组件

说明

配置播放器组件。

接口

```
_controller.setPlayConfig(config);
```

参数说明

参数名	类型	描述
connectRetryCount	int	播放器重连次数，当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过该值设置 SDK 重连次数。
connectRetryInterval	int	播放器重连间隔，当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过该值设置两次重连间隔时间。
timeout	int	播放器连接超时时间。
playerType	int	播放器类型。0：点播；1：直播；2：直播回看。
headers	Map	自定义 http headers。
enableAccurateSeek	bool	是否精确 seek，默认 true。
autoRotate	bool	播放 mp4 文件时，若设为 true 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 true。
smoothSwitchBitrate	bool	平滑切换多码率 HLS，默认 false。设为 false 时，可提高多码率地址打开速度；设为 true，在 IDR 对齐时可平滑切换码率。
cacheMp4ExtName	String	缓存 mp4 文件扩展名，默认 mp4。

progressInterval	int	设置进度回调间隔,若不设置，SDK 默认间隔0.5秒回调一次，单位毫秒。
maxBufferSize	int	最大播放缓冲大小，单位 MB。此设置会影响 playableDuration，设置越大，提前缓存的越多。
maxPreloadSize	int	预加载最大缓冲大小，单位：MB。
firstStartPlayBufferTime	int	首缓需要加载的数据时长，单位 ms，默认值为100ms。
nextStartPlayBufferTime	int	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	String	HLS 安全加固加解密 key。
overlayIv	String	HLS 安全加固加解密 Iv。
extInfoMap	Map	一些不必周知的特殊配置。
enableRenderProcess	bool	是否允许加载后渲染后处理服务,默认开启，开启后超分插件如果存在，默认加载。
preferredResolution	int	优先播放的分辨率，preferredResolution = width * height。

11、开关硬解

说明

开启或关闭硬解播放能力。

接口

```
_controller.enableHardwareDecode(enable);
```

12、获得播放状态

说明

获得当前播放器的播放状态。

接口

```
SuperPlayerState superPlayerState = _controller.getPlayerState();
```

参数说明

参数名	类型	描述
INIT	SuperPlayerState	初始状态
PLAYING	SuperPlayerState	播放中
PAUSE	SuperPlayerState	暂停中
LOADING	SuperPlayerState	缓冲中

END	SuperPlayerState	播放结束

13、进入画中画模式

说明

调动该方法之后，视频将会进入画中画模式，该模式只支持 Android 7.0 以上，并且支持画中画模式的机型。其中 iOS 直播画中画需要使用 premium 权限，并使用 12.1 以上版本的 SDK。

接口

```
_controller.enterPictureInPictureMode(
backIcon: "images/ic_pip_play_replay.png",
playIcon: "images/ic_pip_play_normal.png",
pauseIcon: "images/ic_pip_play_pause.png",
forwardIcon: "images/ic_pip_play_forward.png");
```

参数说明

该参数只适用于 Android 平台，iOS 平台使用默认图片。

参数名	类型	描述
backIcon	String	回退按钮图标，由于 Android 平台限制，图标大小不得超过1M，不传则使用系统自带图标，传空字符串会隐藏图标。
playIcon	String	播放按钮图标，由于 Android 平台限制，图标大小不得超过1M，不传则使用系统自带图标，传空字符串会隐藏图标。
pauseIcon	String	暂停按钮图标，由于 Android 平台限制，图标大小不得超过1M，不传则使用系统自带图标，传空字符串会隐藏图标。
forwardIcon	String	快进按钮图标，由于 Android 平台限制，图标大小不得超过1M，不传则使用系统自带图标，传空字符串会隐藏图标。

14、设置平铺模式

目前播放器组件提供了两种平铺模式，适应视频分辨率模式 `SuperPlayerRenderMode.ADJUST_RESOLUTION` 和 铺满播放器模式 `SuperPlayerRenderMode.FILL_VIEW` ，默认 `SuperPlayerRenderMode.ADJUST_RESOLUTION` 模式，其中使用 `FILL_VIEW` 模式，必须要给播放器设置一个确定的高度，否则播放器的高度可能会撑开到全屏，该模式由组件构造方法传入，示例如下：

```
Widget _getPlayArea() {
  return Container(
    decoration: BoxDecoration(color: Colors.black),
    height: playerHeight,
    child: SuperPlayerView(_controller, SuperPlayerRenderMode.ADJUST_RESOLUTION),
  );
}
```

事件通知

播放事件监听

说明

监听播放器的操作事件。

代码

```
_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);
    }
});
```

事件说明

状态	含义
onStartFullScreenPlay	进入全屏播放
onStopFullScreenPlay	退出全屏播放
onSuperPlayerDidStart	播放开始通知
onSuperPlayerDidEnd	播放结束通知
onSuperPlayerError	播放错误通知
onSuperPlayerBackAction	返回事件

高级功能

1、通过 fileId 提前请求视频数据

可通过 SuperVodDataLoader 提前将视频数据请求下来，提高起播速度。

代码示例

```
SuperPlayerModel model = SuperPlayerModel();
model.appId = 1500005830;
model.videoId = new SuperPlayerVideoId();
```

```
model.videoId.fileId = "8602268011437356984";
model.title = "云点播";
SuperVodDataLoader loader = SuperVodDataLoader();
// model中的必要参数会在SuperVodDataLoader中直接赋值
loader.getVideoData(model, (resultModel) {
  _controller.playWithModelNeedLicence(resultModel);
})
```

2、画中画模式的使用

1. 平台配置

Android

在自己项目 android 包下，找到 build.gradle，确保 compileSdkVersion 和 targetSdkVersion 的版本为31或以上。

iOS

在自己项目的 target 下选择 **Signing & Capabilities** 添加 **Background Modes**，勾选 **“Audio,AirPlay,and Picture in Picture”** 。

2. 复制 superPlayer 示例代码

将 github 项目中 superplayer_widget 导入到自己的 lib 目录下，仿照示例代码中的 demo_superplayer.dart 集成播放器组件。然后就可以在播放器组件的播放界面右边中间看到画中画模式按钮，点击即可进入画中画模式。

3. 监听画中画模式生命周期

使用 SuperPlayerPlugin 中的 onExtraEventBroadcast 可监听到画中画模式的生命周期，示例代码如下：

```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
  int eventCode = event["event"];
  if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_ALREADY_EXIT) {
    // exit pip mode
  } else if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_REQUEST_START) {
    // enter pip mode
  } else if (eventCode == TXVodPlayEvent.EVENT_PIP_MODE_ALREADY_ENTER) {
    // already enter pip mode
  } else if (eventCode == TXVodPlayEvent.EVENT_IOS_PIP_MODE_WILL_EXIT) {
    // will exit pip mode
  } else if (eventCode == TXVodPlayEvent.EVENT_IOS_PIP_MODE_RESTORE_UI) {
    // restore UI only support iOS
  }
});
```

4. 画中画模式进入错误码

进入画中画模式失败的时候，除了有 log 提示以外，还会有 toast 提示，可在 superplayer_widget.dart 的 _onEnterPipMode 方法内修改错误情况处理。错误码含义如下：

参数名	值	描述
NO_ERROR	0	启动成功，没有错误。
ERROR_PIP_LOWER_VERSION	-101	Android 版本过低，不支持画中画模式。
ERROR_PIP_DENIED_PERMISSION	-102	画中画模式权限未打开，或者当前设备不支持画中画。
ERROR_PIP_ACTIVITY_DESTROYED	-103	当前界面已经销毁。
ERROR_IOS_PIP_DEVICE_NOT_SUPPORT	-104	设备或系统版本不支持（iPad iOS9+ 才支持 PIP）。
ERROR_IOS_PIP_PLAYER_NOT_SUPPORT	-105	播放器不支持。
ERROR_IOS_PIP_VIDEO_NOT_SUPPORT	-106	视频不支持。
ERROR_IOS_PIP_IS_NOT_POSSIBLE	-107	PIP 控制器不可用。
ERROR_IOS_PIP_FROM_SYSTEM	-108	PIP 控制器报错。
ERROR_IOS_PIP_PLAYER_NOT_EXIST	-109	播放器对象不存在。
ERROR_IOS_PIP_IS_RUNNING	-110	PIP 功能已经运行。
ERROR_IOS_PIP_NOT_RUNNING	-111	PIP 功能没有启动。

5. **判断当前设备是否支持画中画**

使用 SuperPlayerPlugin 中的 isDeviceSupportPip 可判断当前是否能够开启画中画，代码示例如下：

```
int result = await SuperPlayerPlugin.isDeviceSupportPip();
if(result == TXVodPlayEvent.NO_ERROR) {
  // pip support
}
```

result 的返回结果的含义和画中画模式错误码一致。

6. **使用画中画控制器管理画中画**

画中画控制器 TXPipController 为 superplayer_widget 中封装的画中画工具，**必须与 SuperPlayerView 搭配起来使用**。

进入画中画会自动关闭当前界面，并回调提前设置的监听方法，在回调的方法中可以保存播放器当前界面的必要参数。画中画还原之后，会重新将之前的界面 push 回来，并传递之前保存的参数。
使用该控制器的时候，画中画和播放器只能存在一个实例，当重新进入播放器界面的时候，画中画会自动关闭。

6.1 在自己的项目的入口处，如 main.dart，调用 TXPipController 设置画中画控制跳转，跳转的页面为用于进入画中画的播放器页面。
可根据自身项目情况设置不同的界面，代码实例如下：

```
TXPipController.instance.setNavigatorHandle((params) {
  navigatorKey.currentState?.push(MaterialPageRoute(builder: (_) =>
DemoSuperPlayer(initParams: params)));
});
```

6.2 设置画中画的播放页面监听，需要实现 `TXPipPlayerRestorePage` 方法，设置之后，当即将进入画中画时，控制器会回调 `void onNeedSavePipPageState(Map<String, dynamic> params)` 方法，此时可以在 params 中存入当前页面需要的参数。

```
TXPipController.instance.setPipPlayerPage(this);
```

随后，当用户点击 SuperPlayerView 上的进入画中画按钮的时候，会调用 `SuperPlayerView` 的 `_onEnterPipMode` 内部方法进入画中画，也可以自行调用 `SuperPlayerController` 的 `enterPictureInPictureMode` 方法进入。

3、视频下载

下载视频

1. 使用播放器组件的视频下载，首先需要把SuperPlayerModel中的 `isEnableDownload` 打开，该字段默认关闭。

```
SuperPlayerModel model = SuperPlayerModel();
// 打开视频下载能力
model.isEnableDownload = true;
```

播放器组件目前只会在点播播放模式下启用下载。

2. 使用 `SuperPlayerController` 的 `startDownload` 方法，可以直接下载当前播放器正在播放的视频，对应的是当前播放视频的清晰度。也可是使用 `DownloadHelper` 下载指定视频，如下：

```
DownloadHelper.instance.startDownloadBySize(videoModel, videoWidth, videoHeight);
```

使用 `DownloadHelper` 的 startDownloadBySize，可下载指定分辨率的视频，如果没有该分辨率，会下载相近分辨率的视频。
除了以上接口以外，也可选择传入画质 ID 或者 mediaInfo 直接下载。

```
// QUALITY_240P  240p
// QUALITY_360P  360P
// QUALITY_480P  480p
// QUALITY_540P  540p
```

```
// QUALITY_720P  720p
// QUALITY_1080P 1080p
// QUALITY_2K    2k
// QUALITY_4K    4k
// quality参数可以自定义，取分辨率宽高最小值(如分辨率为1280*720，期望下载此分辨率的流，
quality传入 QUALITY_720P)
// 播放器 SDK 会选择小于或等于传入分辨率的流进行下载
// 使用画质ID下载
DownloadHelper.instance.startDownload(videoModel, qualityId);
// 使用mediaInfo下载
DownloadHelper.instance.startDownloadOrg(mediaInfo);
```

3. 画质 ID 转换

点播的 `CommonUtils` 提供了 `getDownloadQualityBySize` 方法，用于将分辨率转为对应的画质 ID。

```
CommonUtils.getDownloadQualityBySize(width, height);
```

停止下载视频

使用 `DownloadHelper` 的 `stopDownload` 方法可以停止对应的视频下载，示例如下：

```
DownloadHelper.instance.stopDownload(mediaInfo);
```

mediaInfo 可通过 `DownloadHelper` 的 `getMediaInfoByCurrent` 方法获取，或者使用 `TXVodDownloadController` 的 `getDownloadList` 获得下载信息。

删除下载视频

使用 `DownloadHelper` 的 `deleteDownload` 方法，可以删除对应的视频。

```
bool deleteResult = await
DownloadHelper.instance.deleteDownload(downloadModel.mediaInfo);
```

deleteDownload 会返回删除的结果，来判断是否删除成功。

下载状态

`DownloadHelper` 提供了基本的 `isDownloaded` 方法判断视频是否已经下载。也可以注册监听来实时判断下载状态。
`DownloadHelper` 对下载事件进行了分发，可通过如下代码进行事件注册。

```
// 注册下载事件监听
DownloadHelper.instance.addDownloadListener(FTXDownloadListener((event, info) {
      // 下载状态变化
    }, (errorCode, errorMsg, info) {
      // 下载错误回调
    }));
// 移除下载事件监听
```

```
DownloadHelper.instance.removeDownloadListener(listener);
```

此外，还可以通过 `TXVodDownloadController.instance.getDownloadInfo(mediaInfo)` 方法或者 `TXVodDownloadController.instance.getDownloadList()` 方法直接查询 mediaInfo 中的 downloadState 来判断下载状态。

播放下载的视频

`TXVodDownloadController.instance.getDownloadInfo(mediaInfo)` 和 `TXVodDownloadController.instance.getDownloadList()` 获得到的视频信息中有个 playPath 字段，使用 TXVodPlayerController 直接播放即可。

```
controller.startVodPlay(mediaInfo.playPath);
```

4、横竖屏的使用

横竖屏切换配置

播放器组件横竖屏的切换，iOS 需要使用 Xcode 打开，打开项目配置，General 分页下的 Deployment 标签下，勾选上 `Landscape left` 和 `Landscape right` 。确保 iOS 设备能够支持横屏。
如果希望自己的 App 其他页面稳定保持竖屏，不受横竖屏自动旋转影响，需要在自己项目下的入口处，配置竖屏。代码如下：

```
SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
```

根据 sensor 配置自动全屏

Android 端需要调用如下方法，来开启对 sensor 的监听。

```
SuperPlayerPlugin.startVideoOrientationService();
```

调用之后，在 Android 设备上，将会对 Android sensor 进行监听，会通过 `SuperPlayerPlugin.instance.onEventBroadcast` 对 flutter 侧发送旋转事件。播放器组件内部也会自动根据该事件旋转播放器。监听使用范例如下：

```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
    int eventCode = event["event"];
    if (eventCode == TXVodPlayEvent.EVENT_ORIENTATION_CHANGED ) {
      int orientation = event[TXVodPlayEvent.EXTRA_NAME_ORIENTATION];
      // do orientation
    }
  });
```

常见问题

1. 集成后，播放经常出现有声音，却没有画面的现象，怎么处理？

   由于含 UI 组件和 Flutter 播放器插件会随着版本迭代而更新调用方式，需要保持两者版本一致。播放器组件版本可以使用 `PlayerConstants.PLAYER_WIDGET_VERSION` 来确认，Flutter 播放器插件版本，除了集成时候的版本以外，也可以使用

`FPlayerPckInfo.PLAYER_VERSION` 版本确认，确保两者的版本一致。

2. 需要去掉屏幕自动旋转的能力，怎么处理?

播放器组件采用的开源方式，为了满足不同客户的业务定制化需求，建议客户采用直接修改组件代码的方式来完成业务定制化需求。如果需要跟进组件版本，可以建立私有仓库，fork 组件代码，有更新之后可以 pick 变动。

Demo 体验

更多功能和调试 Demo 体验，请 单击这里，运行该 demo 的时候，需要在 demo_config 中设置自己的播放器 license，并在 Android 和 iOS 配置中，将包名和 bundleId 修改为自己签名的包名和 bundleId。

无 UI 集成方案
Web 端集成
TCPlayer 集成指引

最近更新时间：2025-05-09 11:28:12

本文档将介绍适用于点播播放和直播播放的 Web 播放器 SDK（TCPlayer），它可快速与自有 Web 应用集成，实现视频播放功能。Web 播放器 SDK（TCPlayer）内默认包含部分 UI，您可按需取用。

概述

Web 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	https://xxx.vod.myqcloud.com/xxx.mp3	支持	支持
MP4	点播	https://xxx.vod.myqcloud.com/xxx.mp4	支持	支持
HLS（M3U8）	直播	https://xxx.liveplay.myqcloud.com/xxx.m3u8	支持	支持
	点播	https://xxx.vod.myqcloud.com/xxx.m3u8	支持	支持
FLV	直播	https://xxx.liveplay.myqcloud.com/xxx.flv	支持	部分支持
	点播	https://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\浏览器	Chrome	Firefox	Edge	QQ 浏览器	Mac Safari	iOS Safari	微信	Android Chrome	IE 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	✓	✓	✓	✓	✓	✓	✓	✓	✓
幽灵水印	✓	✓	✓	✓	✓	✓	✓	✓	✓
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。

准备工作

播放器 SDK Web 端（TCPlayer）自 5.0.0 版本起需获取 License 授权后方可使用。若您无需使用新增的高级功能，可直接申请基础版 License 以**继续免费使用 TCPlayer**；若您需要使用新增的高级功能则需购买高级版 License。具体信息如下：

TCPlayer 功能	功能范围	所需 License	定价	授权单位
基础版功能	包含 5.0.0 以前版本提供的全部功能，详情见 产品功能	播放器 Web 端基础版 License	0元 免费申请	精准域名（1个 License 最多可授权10个精准域名）
高级版功能	基础版功能、VR 播放 、安全检查	播放器 Web 端高级版 License	399元/月 立即购买	泛域名（1个 License 最多可授权1个泛域名）

说明：
播放器 Web 端基础版 License 可免费申请，申请后有效期默认1年；在有效期剩余时间小于30天时，可免费续期。

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

播放器 SDK 支持 cdn 和 npm 两种集成方式：

1. 通过 cdn 集成

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，自行部署情况下，需要手动引用资源包文件夹 libs 下面的依赖文件，否则会默认请求腾讯云 cdn 文件。在 html 页面内引入播放器样式文件与脚本文件：

```
<link href="https://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="https://aaa.xxx.ccc/libs/hls.min.x.xx.m.js"></script>
<!--播放器脚本文件-->
<script src="https://aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script>
```

2. 通过 npm 集成

首先安装 tcplayer 的 npm 包：

```
npm install tcplayer.js
```

导入 SDK 和样式文件：

```
import TCPlayer from 'tcplayer.js';
import 'tcplayer.js/dist/tcplayer.min.css';
```

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-playsinline>
</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。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 使用播放器播放 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。

```
// player-container-id 为播放器容器 ID，必须与 html 中一致
var player = TCPlayer('player-container-id', {
    sources: [{
      src: 'path/to/video', // 播放地址
```

```
    }],
    licenseUrl: 'license/url', // license 地址，必传。参考准备工作部分，在视立方控制台申请 license 后可获得 licenseUrl
});

// 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）
    //必传 psign， psign 即播放器签名，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',
    licenseUrl: 'license/url', // 参考准备工作部分，在视立方控制台申请 license 后可获得 licenseUrl
});
```

> **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。

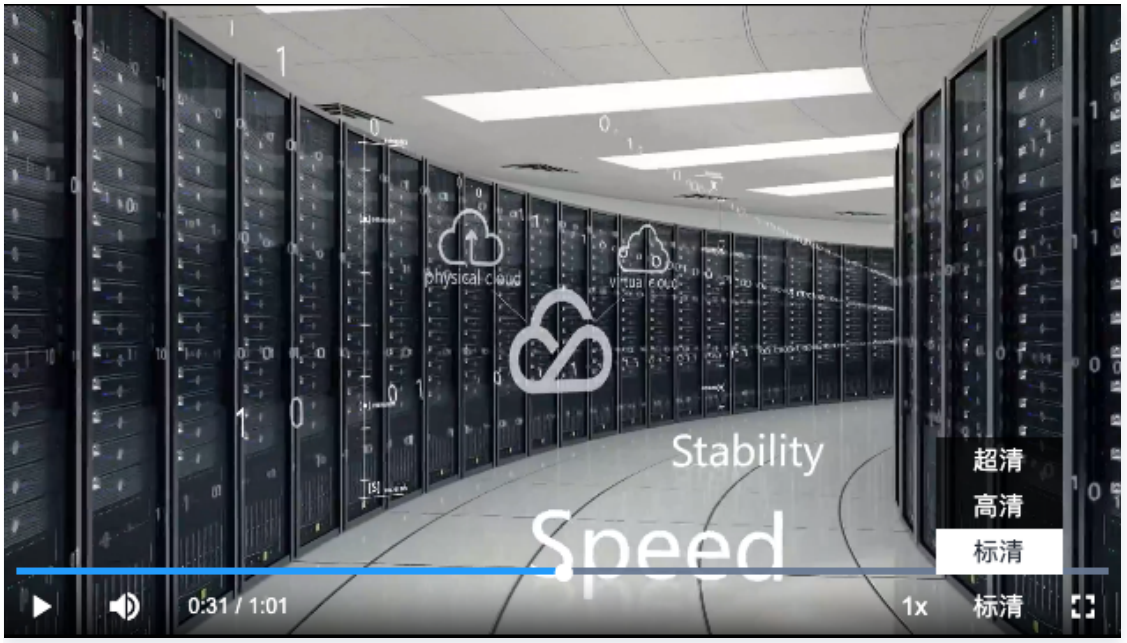
步骤4: 更多功能

播放器可以结合云点播的服务端能力实现高级功能，例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。

TCPlayer 清晰度配置说明

最近更新时间：2023-10-11 15:05:21

概述



在播放过程中，您可以通过自动或手动切换视频清晰度，以适应不同尺寸的播放设备和网络环境，从而提高观看体验。本文将从以下几个场景进行说明。

直播场景

直播场景以 URL 的形式来播放视频，初始化播放器时，可以通过 sources 字段指定所要播放的 URL。或者在初始化播放器之后，调用播放器实例上的 src 方法进行播放。

1. 自适应码率（ABR）

自适应码率地址在切换时可以做到无缝衔接，切换过程不会出现中断或跳变，实现了观感和听感的平滑过渡。使用该技术也比较简单，仅需将播放地址传给播放器，播放器会自动解析子流，并将清晰度切换组件渲染到控制条上。

示例1: 播放 HLS 自适应码率地址

在初始化播放器时，传入自适应码率地址，播放器将自动生成清晰度切换组件，并会根据网络状况进行自动切换。

```
const player = TCPlayer('player-container-id', { // player-container-id 为播放器容器ID，必须与html中一致
  sources: [{
    src: 'https://hls-abr-url', // hls 自适应码率地址
  }],
});
```

> **注意：**

> 解析 HLS 自适应码率的子流，需要依赖播放环境的 MSE API。在不支持 MSE 的浏览器环境（例如 iOS 的 Safari 浏览器），会由浏览器内部处理，根据网络情况自动切换清晰度，但无法解析出多种清晰度来供您手动切换。

示例2：播放 WebRTC 自适应码率地址

在 WebRTC 自适应码率场景，传入地址后，播放器会根据地址中的 ABR 模板自动拆解子流地址。

```
const player = TCPlayer('player-container-id',{
  sources: [{
    src: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c&tabr_bitrates=d1080p,d540p
,d360p&tabr_start_bitrate=d1080p&tabr_control=auto'
  }],

  webrtcConfig: {
    // 是否渲染多清晰度的开关，默认开启，可选
    enableAbr: true,
    // 模板名对应的label名，可选
    abrLabels: {
      d1080p: 'FHD',
      d540p: 'HD',
      d360p: 'SD',
      auto: 'AUTO',
    },
  },
});
```

这里对 WebRTC 地址中的参数做以下说明：

1. tabr_bitrates 指定了 ABR 模板，有几个模板就会渲染出几个清晰度。如果没有单独设置清晰度的 label，模板名称（如 d1080p）将被设为清晰度名称。
2. tabr_start_bitrate 指定了起播的清晰度规格。
3. tabr_control 设置是否开启自动切换清晰度。开启后，播放器会单独渲染出自动清晰度选项。

2. 手动设置清晰度

如果播放地址不是自适应码率地址，也可以手动设置清晰度。参见如下代码：

```
const player = TCPlayer('player-container-id', { // player-container-id 为播放器容器
ID，必须与html中一致
  multiResolution:{
    // 配置多个清晰度地址
    sources:{
      'SD':[{
        src: 'http://video-sd-url',
      }],
      'HD':[{
        src: 'http://video-hd-url',
      }],
```

```
        'FHD':[{
          src: 'http://video-fhd-url',
        }]
      },
      // 配置每个清晰度标签
      labels:{
        'SD':'标清','HD':'高清','FHD':'超清'
      },
      // 配置各清晰度在播放器组件上的顺序
      showOrder:['SD','HD','FHD'],
      // 配置默认选中的清晰度
      defaultRes: 'SD',
    },
});
```

点播场景

在点播场景下，如果通过 fileID 播放，播放哪种规格的文件（原始文件、转码文件、自适应码率文件）以及自适应码率文件子流的清晰度，都是在播放器签名中设置的。您可以参见指引 播放自适应码流视频，以便于您了解点播场景下播放视频的整个流程。
计算播放器签名时，可以通过 contentInfo 字段中的 resolutionNames 来设定不同分辨率的子流的展示名字。不填或者填空数组则使用默认配置。

```
resolutionNames: [{
  MinEdgeLength: 240,
  Name: '240P',
}, {
  MinEdgeLength: 480,
  Name: '480P',
}, {
  MinEdgeLength: 720,
  Name: '720P',
}, {
  MinEdgeLength: 1080,
  Name: '1080P',
}, {
  MinEdgeLength: 1440,
  Name: '2K',
}, {
  MinEdgeLength: 2160,
  Name: '4K',
}, {
  MinEdgeLength: 4320,
  Name: '8K',
}]
```

播放时的子流数量取决于转码时根据不同的自适应码率模板转换出的子流数。这些子流会依据短边长度落在由 resolutionNames 设定的哪个 MinEdgeLength 范围，再以对应的 Name 作为清晰度名称进行展示。
若您需要快速体验生成播放器签名，可以使用腾讯云点播控制台的 播放器签名生成工具 。

TCPlayer 快直播降级说明

最近更新时间：2023-10-17 11:31:42

降级场景

快直播基于 WebRTC 实现，依赖于操作系统和浏览器对于 WebRTC 的支持。
目前，SDK 对以下操作系统和浏览器进行了测试，测试结果如下：

操作系统	操作系统版本	浏览器类型	浏览器版本	是否支持拉流
Windows	win 10	Chrome	86+	✓
		Firefox	88+	✓
		Microsoft Edge	86+	✓
macOS	10.5+	Safari	13.1+	✓
		Chrome	86+	✓
		Firefox	88+	✓
		Microsoft Edge	86+	✓
iOS	13.1.1+	Safari	13.7+	✓
		Chrome	86+	✓
		Firefox	33+	✓
		Microsoft Edge	89	✓
		微信内嵌	-	✓
Android	-	Chrome	86+	✓
		Firefox	88+	✓
		微信内嵌	X5 内核	✓
		微信内嵌	XWeb 内核	✓

此外，在部分支持 WebRTC 的浏览器，也会出现解码失败或者服务端问题，这些情况下，播放器都会将 WebRTC 地址转换为兼容性较好的 HLS 地址来播放，这个行为称为降级处理。
总结
会触发降级的场景有以下几个：

- 浏览器环境不支持 WebRTC。
- 连接服务器失败，并且连接重试次数已超过设定值（内部状态码 -2004）。
- 播放过程解码失败（内部状态码 -2005）。
- 其他 WebRTC 相关错误（内部状态码 -2001）。

降级方式

1. 自动降级

初始化播放器时，通过 sources 字段传入了快直播地址，在需要降级处理的环境，播放器自动会进行协议的转换，将快直播地址转换为 HLS 协议地址。
例如，快直播地址：

```
webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c
```

会自动转换为：

```
https://global-lebtest-play.myqcloud.com/live/lebtest.m3u8?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c
```

2. 指定降级

在播放自适应码率（ABR）场景，如果需要降级，并不能直接通过格式转换得到自适应码率的 HLS 地址，需要手动指定。又或者是在用户希望手动指定的其他场景，都可以通过如下方式指定降级地址，这里的地址并不局限于 HLS 协议，也可以是其他协议地址。

```
var player = TCPlayer('player-container-id',{
    sources: 'webrtc://global-lebtest-play.myqcloud.com/live/lebtest?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=69f1eb8c&tabr_bitrates=d1080p,d540p
,d360p&tabr_start_bitrate=d1080p',
    webrtcConfig: {
      fallbackUrl: 'https://global-lebtest-
play.myqcloud.com/live/lebtest_HLSABR.m3u8',
    },
});
```

降级回调

当触发降级时，播放器会触发回调：

```
player.on('webrtcfallback', function(event) {
    console.log(event);
});
```

iOS 端集成
集成指引

最近更新时间：2024-11-26 11:02:11

本文主要介绍如何快速地将腾讯云视立方 · 播放器 LiteAVSDK_Player（iOS）集成到您的项目中，按照如下步骤进行配置，就可以完成 SDK 的集成工作。

开发环境要求

- Xcode 9.0+。
- iOS 9.0 以上的 iPhone 或者 iPad 真机。
- 项目已配置有效的开发者签名。

集成 LiteAVSDK

您可以选择使用 CocoaPods 自动加载的方式，或者先下载 SDK，再将其导入到您当前的工程项目中。

CocoaPods集成

1. **安装 CocoaPods**
   在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：

```
sudo gem install cocoapods
```

2. **创建 Podfile 文件**
   进入项目所在路径，输入以下命令行之后项目路径下会出现一个 Podfile 文件。

```
pod init
```

3. **编辑 Podfile 文件**
   使用 CocoaPod 官方源，支持选择版本号。编辑 Podfile 文件：
   Pod 方式直接集成最新版本 TXLiteAVSDK_Player：

```
platform :ios, '9.0'
source 'https://github.com/CocoaPods/Specs.git'

target 'App' do
pod 'TXLiteAVSDK_Player'
end
```

   如果您需要指定某一个特定版本，可以在 podfile 文件中添加如下依赖：

```
pod 'TXLiteAVSDK_Player', '~> 10.3.11513'
```

   如果您需要集成播放高级版本（Premium）SDK，在 podfile 文件中添加如下依赖：

```
platform :ios, '9.0'
source 'https://github.com/CocoaPods/Specs.git'

target 'App' do
pod 'TXLiteAVSDK_Player_Premium'
end
```

4. **更新并安装 SDK**
   - 在终端窗口中输入如下命令以更新本地库文件，并安装 LiteAVSDK：

     ```
     pod install
     ```

   - 或使用以下命令更新本地库版本：

     ```
     pod update
     ```

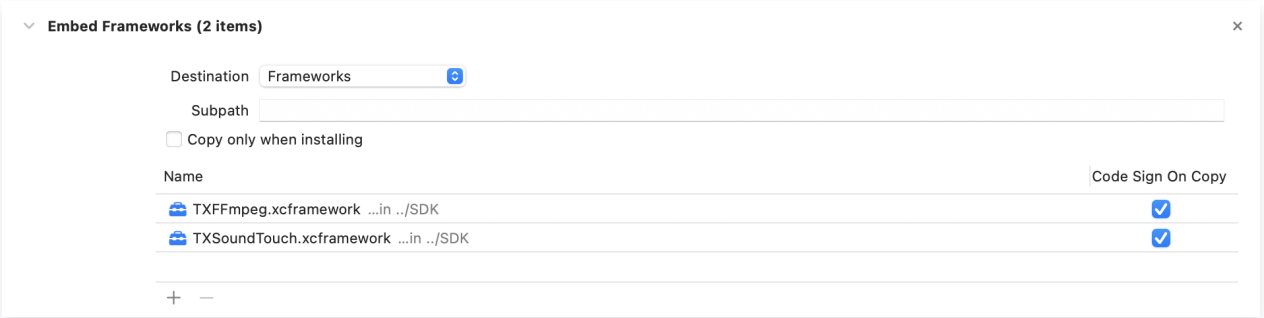
     pod 命令执行完后，会生成集成了 SDK 的 `.xcworkspace` 后缀的工程文件，双击打开即可。

手动集成 SDK

1. 下载 最新版本 TXLiteAVSDK_Player 的 SDK + Demo 开发包。
   如果您需要集成播放高级版本（Premium）SDK，请 单击此处 下载。
2. 将 SDK/TXLiteAVSDK_Player.xcframework 添加到待集成的工程中，并勾选 `Do Not Embed` 。
3. 需要配置项目 Target 的 -ObjC，否则会因为加载不到 SDK 的类别而导致 Crash。

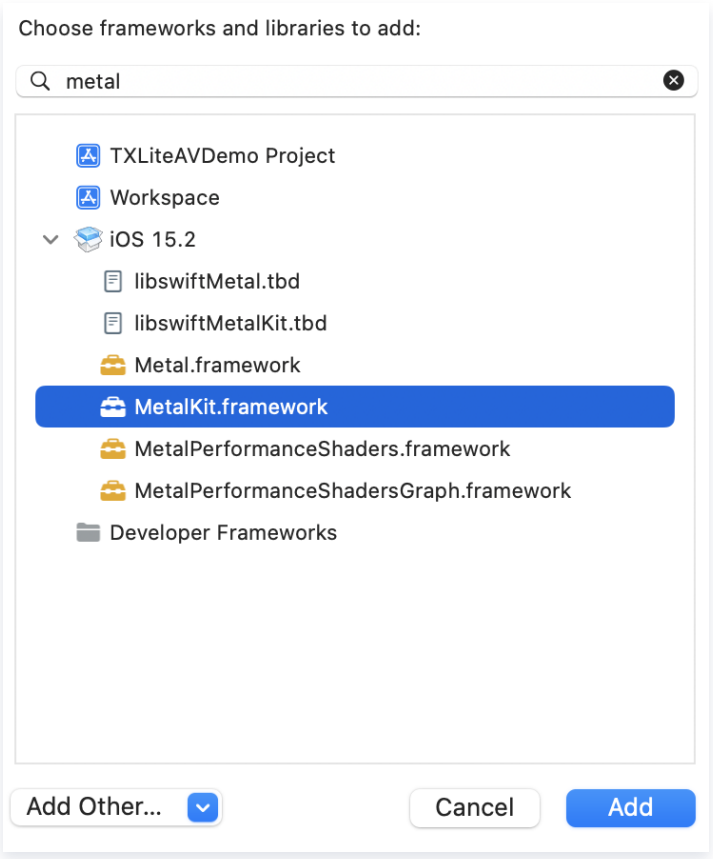
   ```
   打开 Xcode -> 选择对应的 Target -> 选择"Build Setting" Tab -> 搜索"Other Link Flag" -
   > 输入"-ObjC"
   ```

4. 添加相应的库文件（SDK 目录里）
   - **TXFFmpeg.xcframework**：将.xcframework 文件添加到项目工程中，并在“General > Frameworks, Libraries, and Embedded Content”中将其设置为“Embed&Sign”，并在“Project Setting > Build Phases > Embed Frameworks”中进行检查，设置"Code Sign On Copy"选项为勾选状态，
   - **TXSoundTouch.xcframework**：将.xcframework 文件添加到项目工程中，并在“General > Frameworks, Libraries, and Embedded Content”中将其设置为“Embed&Sign”，并在“Project Setting > Build Phases > Embed Frameworks”中进行检查，设置 Code Sign On Copy 选项为勾选状态。

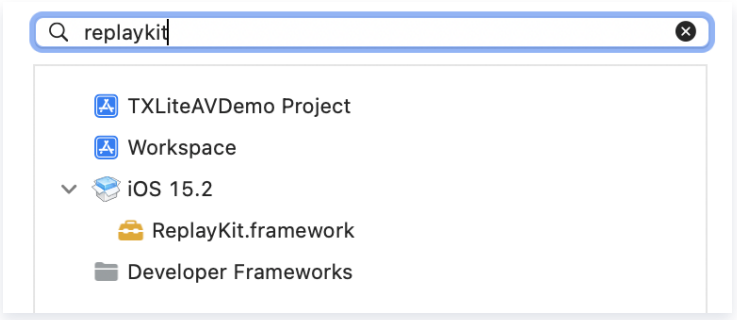


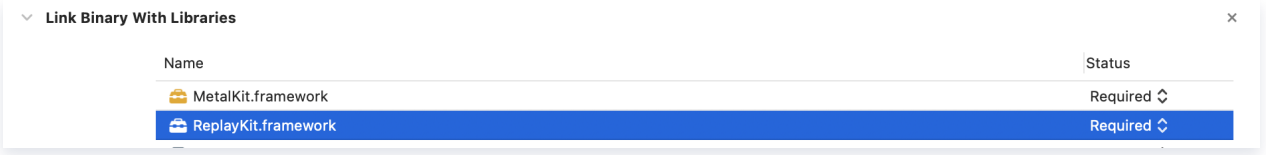
同时，切换到 Xcode 的 “Build Settings > Search Paths”，在“Framework Search Paths”中添加上述 Framework 所在的路径。

- **MetalKit.framework**：打开 Xcode，切换到“project setting > Build Phases > Link Binary With Libraries”，选择左下角的“+”号，并输入“MetalKit”，并加入项目工程中，如下图所示：



- **ReplayKit.framework**：打开 Xcode，切换到“project setting > Build Phases > Link Binary With Libraries”，选择左下角的“+”号，并输入“ReplayKit”，并加入项目工程中，如下图所示：

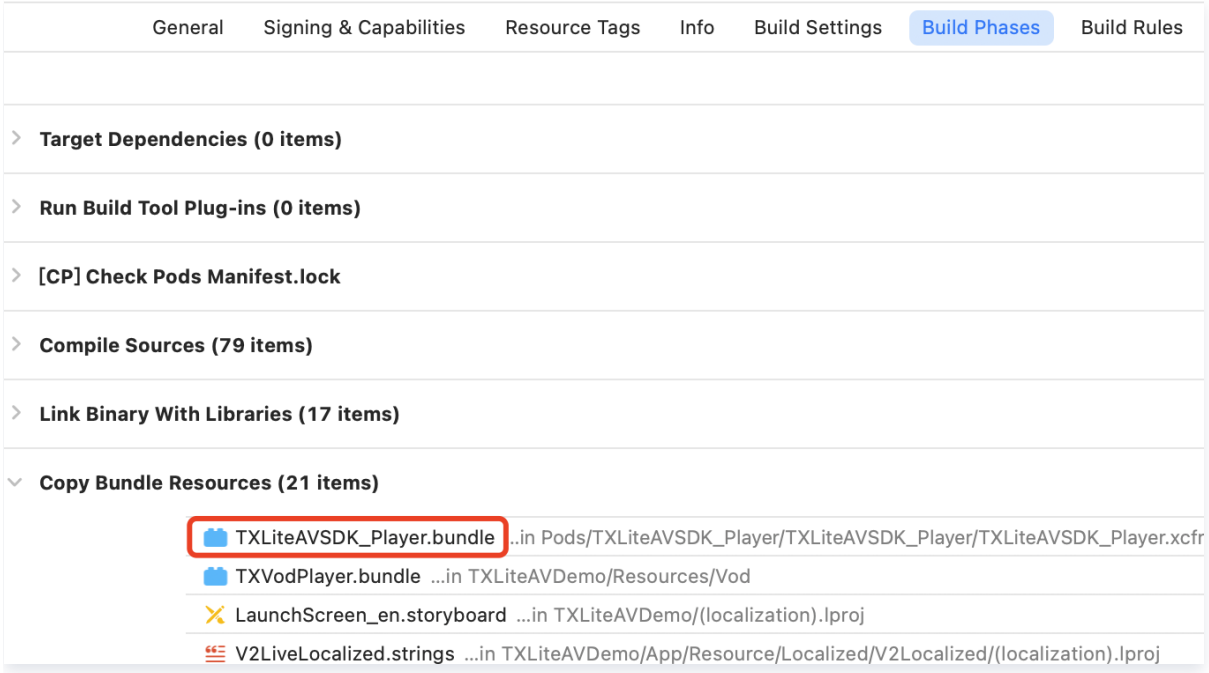




使用同样的方式添加如下系统库：

- **系统 Framework 库**：SystemConfiguration, CoreTelephony, VideoToolbox, CoreGraphics, AVFoundation, Accelerate, MobileCoreServices。
- **系统 Library 库**：libz, libresolv, libiconv, libc++, libsqlite3。

5. 从 **11.7.15343** 版本 以后，Player SDK适配了苹果的隐私清单，下载对应SDK并将SDK内 **TXLiteAVSDK_Player.bundle** 添加到项目工程里：

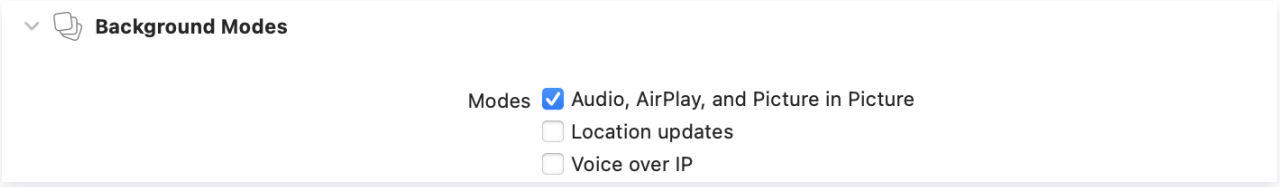


如果是 pods 引入 SDK 的话，可以忽略上述步骤。

画中画功能

如果需要使用画中画能力，请按如下图的方式进行配置，若无此部分需求可以忽略。

1. 为了使用 iOS 的画中画（Picture-In-Picture），请将 SDK 升级到10.3版本及以上。
2. 使用画中画能力时，需要开通后台模式。XCode 选择对应的Target > Signing & Capabilities > Background Modes，勾选“Audio, AirPlay, and Picture in Picture”，如图所示：



在工程中引入 SDK

项目代码中使用 SDK 有两种方式：

- **方式一：** 在项目需要使用 SDK API 的文件里，添加模块引用。

```
@import TXLiteAVSDK_Player;
// 如果您使用的的 Premium 版本，请用： @import TXLiteAVSDK_Player_Premium;
```

- **方式二：**在项目需要使用 SDK API 的文件里，引入具体的头文件。

```
#import "TXLiteAVSDK_Player/TXLiteAVSDK.h"
// 如果您使用的的 Premium 版本，请用： #import
"TXLiteAVSDK_Player_Premium/TXLiteAVSDK.h"
```

给 SDK 配置 License 授权

1. 单击 License 申请 获取测试用 License，不配置 License 将会播放时视频失败，具体操作请参见 测试版 License 。您会获得两个字符串：一个字符串是 licenseURL，另一个字符串是解密 key。
2. 获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

设置 SDK 接入环境

> **注意：**
> 如果您的应用要出海， 服务海外用户， 则此步骤是必须的。

为服务客户更高质量、更安全合规地开展业务，符合各国家和地区的法律法规要求，腾讯云提供两套 SDK 接入环境。若您服务中国地区用户，则无需此配置，若您服务全球用户，推荐您使用以下接口配置全球接入环境。

```
// 若您服务全球用户， 配置 SDK 接入环境为全球接入环境
[TXLiveBase setGlobalEnv:"GDPR"]
```

常见问题

项目里面同时集成了直播 SDK/实时音视频/播放器等 LiteAVSDK 系列的多个 SDK 报符号冲突问题怎么解决?

如果集成了2个或以上产品（直播、播放器、TRTC、短视频）的 LiteAVSDK 版本，编译时会出现库冲突问题，因为有些 SDK 底层库有相同符号文件，这里建议只集成一个全功能版 SDK 可以解决，直播、播放器、TRTC、短视频这些都包含在一个 SDK 里面。具体请参见 SDK 下载 。

Swift 项目工程里怎么调用 SDK 的 API 方法?

如果在 Swift 的项目工程里想调用 SDK 的 API 接口，有下面两种方式：

方式一：使用桥接头文件

1. 创建桥接头文件。例如 `***-Bridging-Header.h` ，并添加如下代码 `#import <TXLiteAVSDK_Player/TXLiteAVSDK.h>` 。
2. 配置工程 `BuildSetting` 的 `Objective-c Bridging header` 选项。设置桥接文件的路径并添加到 `Objective-c Bridging header` 中（ 如：$(SRCROOT)/SwiftCallOC/***-Bridging-Header.h，根据项目具体路径确定），编译运行即可。

方式二：使用SDK内的 module.modulemap 文件

1. 检查 `TXLiteAVSDK_Player.framework` 里是否有包含 `Modules - module.modulemap` 文件（Player SDK 默认都提供）。
2. 配置工程 `BuildSetting` 的 `Swift Compiler - Search Paths` 选项。添加 `module.modulemap` 文件所在的目录路径或其上层目录路径，此处可为：

${PODS_ROOT}/TXLiteAVSDK_Player/TXLiteAVSDK_Player/TXLiteAVSDK_Player.framework/Modules（根据项目具体路径确定）。

3. 在需要调用的类顶部，使用 `import TXLiteAVSDK_Player` 来进行引入并调用相关的方法。

以上集成的方式及 Demo，可以具体参见 GitHub Demo 。

点播场景

最近更新时间：2025-06-23 11:57:12

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 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。
- 播放器推出短视频组件、画中画2.0、VR 播放等高级组件，功能介绍和使用指引请参见 移动端高级功能 。

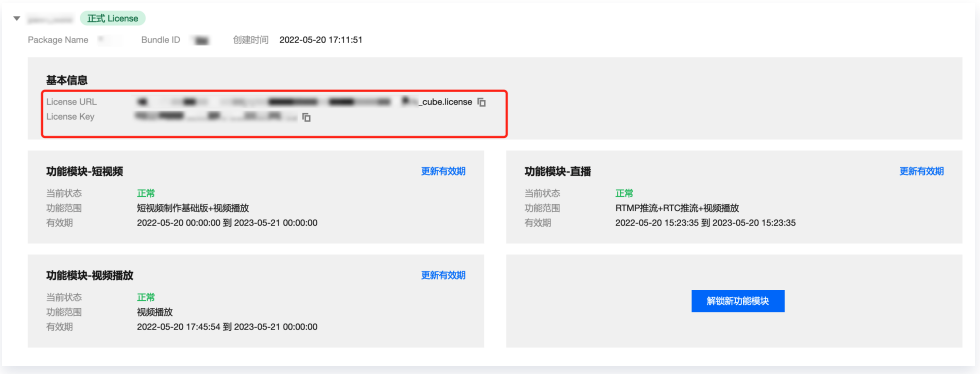
SDK 集成

步骤1：集成 SDK 开发包

下载和集成 SDK 开发包，请参见同目录下的 SDK 集成指引 。

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 播放器 License 获取相关授权。
- 获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

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。

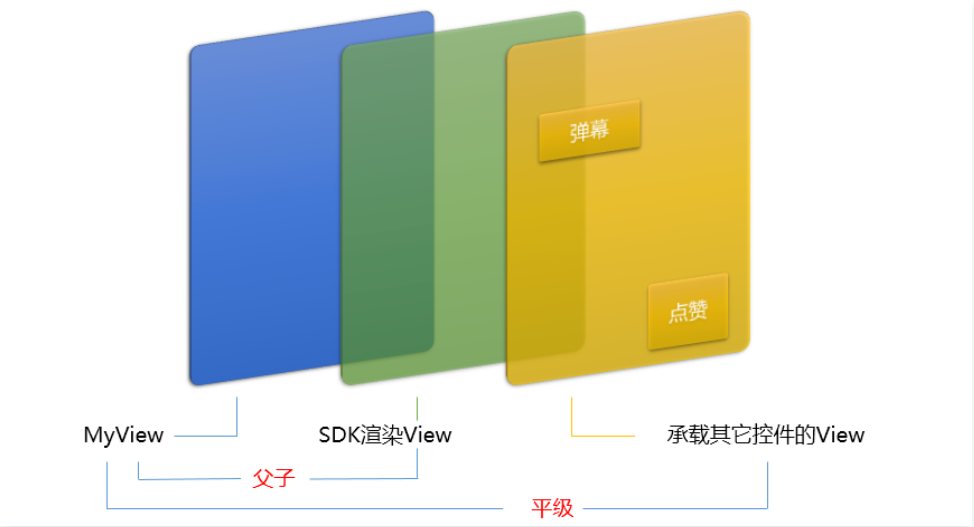
```
TXVodPlayer *_txVodPlayer = [[TXVodPlayer alloc] init];
[_txVodPlayer setupVideoWidget:_myView insertIndex:0]
```

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，只需要调整您所创建的 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。

```
// 播放 URL 视频资源
NSString* url = @"http://1252463788.vod2.myqcloud.com/xxxxx/v.f20.mp4";
[_txVodPlayer startVodPlay:url];

// 播放沙盒本地视频资源
// 获取 Documents 路径
NSString *documentPath =
[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)
```

```
firstObject];
// 获取本地视频路径
NSString *videoPath = [NSString
stringWithFormat:@"%@/video1.m3u8",documentPath];
[_txVodPlayer startVodPlay:videoPath];
```

通过 fileId 方式

```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788;
p.fileId = @"4564972819220421305";
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
p.sign = @"psignxxxxx"; // 播放器签名
[_txVodPlayer startVodPlayWithParams: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 startVodPlay: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 默认支持精准 seek，可以在播放器前通过 `TXVodPlayConfig#setEnableAccurateSeek` 进行配置。

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

精准和非精准 Seek

播放器 SDK 11.8 版本开始，支持调用 seek 接口时，指定精准或非精准 seek。

```
float time = 600; // float 类型时单位为 秒
// 调整进度
[_txVodPlayer seek:time accurateSeek:YES];  // 精准 seek
[_txVodPlayer seek:time accurateSeek:NO];  // 非精准 seek
```

Seek 到视频流指定 PDT 时间点

跳转到视频流指定 PDT（Program Date Time ）时间点，可实现视频快进、快退、进度条跳转等功能，目前只支持 HLS 视频格式。
注意：播放器高级版 11.6 版本开始支持。

```
long long pdtTimeMs = 600; // 单位为 毫秒
[_txVodPlayer seekToPdtTime:time];
```

从指定时间开始播放

首次调用 startVodPlay之前，支持从指定时间开始播放。

```
float startTimeInSecond = 60; // 单位：秒
[_txVodPlayer setStartTime:startTimeInSecond];  // 设置开始播放时间
[_txVodPlayer startVodPlay: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

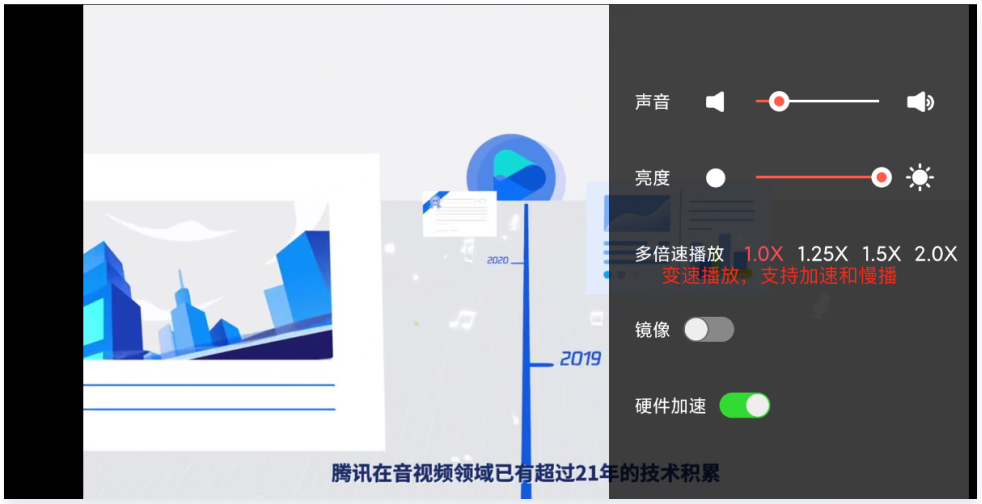
最长边填充

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。




```
// 设置1.2倍速播放
[_txVodPlayer setRate:1.2];
// 开始播放
[_txVodPlayer startVodPlay: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。

```
NSMutableDictionary<NSString *, NSString *> *httpHeader = [[NSMutableDictionary
alloc] init];
[httpHeader setObject:@"${Referer Content}" forKey:@"Referer"];
[_config setHeaders:httpHeader];
[_txVodPlayer setConfig:_config];
```

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 **startVodPlay**，否则会产生比较严重的花屏问题。

```
[_txVodPlayer stopPlay];
_txVodPlayer.enableHWAcceleration = YES;
[_txVodPlayer startVodPlay:_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。可以通过下面方法获取多码率数组。

```
// 在收到播放器 PLAY_EVT_VOD_PLAY_PREPARED 事件调用 getSupportedBitrates 才会有值返回
NSArray *bitrates = [_txVodPlayer supportedBitrates]; //获取多码率数组
// TXBitrateItem 类字段含义：index-码率下标；width-视频宽；height-视频高；birate-视频码率
```

```
TXBitrateItem *item = [bitrates objectAtIndex:i];
[_txVodPlayer setBitrateIndex:item.index]; // 切换码率到想要的清晰度

// 获取当前播放的码率下标，返回值 -1000 为默认值，表示没有设置过码率标；返回值 -1 表示开启了自适应码流
int index = [_txVodPlayer bitrateIndex];
```

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。
如果您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，从而避免播放后切换码流。详细方法参考 播放器配置#启播前指定分辨率 。

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。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. 开启平滑切换码率

在启动播放前，通过开启平滑切换码率，在播放过程中切换码率，可以达到无缝平滑切换不同清晰度。对比关闭平滑切换码率，切换过程比较耗时、体验更好，可以根据需求进行设置。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
// 设为 YES，在 IDR 对齐时可平滑切换码率，设为 NO 时，可提高多码率地址打开速度
[_config setSmoothSwitchBitrate:YES];
[_txVodPlayer setConfig:_config];
```

13.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

             // 获取 PDT 时间，播放器高级版 11.6 版本开始支持
            long long pdt_time_ms = [param[VOD_PLAY_EVENT_PLAY_PDT_TIME_MS]
longLongValue];
    }
}
```

14. 播放网速监听

通过 事件监听 方式，可以在视频播放卡顿时显示当前网速。

- 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。

15. 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。

16. 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
```

17. 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 startVodPlay:url];
```

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

18. 屏幕控制(亮屏和灭屏)

由于手机个性化设置中经常会设置屏幕锁定的时间，在视频播放场景中可能会出现屏幕灭屏（或被锁定）的情况，这极大影响了用户体验。因此，为了解决这一情况需要在播放过程中的相关时机添加如下代码，以便使屏幕一直处于亮屏状态。

```
（1）亮屏（禁止灭屏）
// 启动播放（startVodPlay / startPlayDrm / startVodPlayWithParams）
```

```
// 恢复播放（resume）
 [[UIApplication sharedApplication] setIdleTimerDisabled:YES];

（2）灭屏（恢复灭屏）
// 停止（stopPlay）
// 暂停（pause）
[[UIApplication sharedApplication] setIdleTimerDisabled:NO];
```

注意：
以上接口的使用请注意在主线程调用。

19. DRM 加密视频播放

注意：
此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持播放商业级 DRM 加密视频，目前支持 WideVine 和 Fairplay 两种 DRM 方案。更多的商业级 DRM 信息，请参考 商业级 DRM 综述 。
可通过下面两种方式播放 DRM 加密视频：

通过 FileId 播放

```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = ${appId};  // 腾讯云账户的 appId
p.fileId = @"${fieId}";  // DRM 加密视频的 fileId
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
p.sign = @"${psgin}"; // 加密视频的播放器签名
[_txVodPlayer startVodPlayWithParams:p];
```

通过 FileId 播放适用于接入云点播后台。这种方式和播放普通 FileId 文件没有差别，需要在云点播先配置资源为 DRM 类型，SDK 会在内部识别并处理。

自定义配置播放

```
// 通过 TXVodPlayer#startPlayDrm 接口播放
// @param certificateUrl 证书提供商url
// @param licenseUrl 解密的key url
// @param videoUrl 待播放视频的Url地址
TXPlayerDrmBuilder *builder = [[TXPlayerDrmBuilder alloc]
initWithDeviceCertificateUrl:@"${certificateUrl}" licenseUrl:@"${licenseUrl}"
videoUrl:@"${videoUrl}"];
```

```
[_txVodPlayer startPlayDrm:builder];
```

20. 外挂字幕

> **注意：**
> 此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持添加和切换外挂字幕，现已支持 SRT 和 VTT 这两种格式的字幕。
建议在 `startVodPlay` 之前添加字幕和配置字幕样式，在收到 `PLAY_EVT_VOD_PLAY_PREPARED` 事件后，调用 selectTrack 选择字幕。添加字幕后，并不会主动加载字幕， 调用 `selectTrack` 后，才会加载字幕，字幕选择成功会有 `VOD_PLAY_EVT_SELECT_TRACK_COMPLETE` 事件回调。
用法如下：
步骤1：添加外挂字幕。

```
// 传入 字幕url， 字幕名称， 字幕类型，建议在启动播放器前添加
[_txVodPlayer addSubtitleSource:@"https://mediacloud-76607.gzc.vod.tencent-cloud.com/DemoResource/subtitleVTT.vtt" name:@"subtitleName"
mimeType:TX_VOD_PLAYER_MIMETYPE_TEXT_VTT];
```

步骤2：播放后切换字幕。

```
// 开始播放视频后，选中添加的外挂字幕，请在收到 VOD_PLAY_EVT_SELECT_TRACK_COMPLETE 事件后调用
NSArray<TXTrackInfo *> *subtitlesArray = [_txVodPlayer getSubtitleTrackInfo];
for (int i = 0; i < subtitlesArray.count; i++) {
    TXTrackInfo *info = subtitlesArray[i];
    if (info.trackIndex == 0) {
        [_txVodPlayer selectTrack:info.trackIndex];  // 选中字幕
    } else {
        // 其它字幕不需要的话， 进行deselectTrack
        [_txVodPlayer deselectTrack:info.trackIndex];
    }
}

// 监听轨道切换消息
- (void)onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary *)param {
    if (EvtID == VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
        int trackIndex = [(NSNumber *)[param valueForKey:EVT_KEY_SELECT_TRACK_INDEX] intValue];
        int errorCode = [(NSNumber *)[param valueForKey:EVT_KEY_SELECT_TRACK_ERROR_CODE] intValue];
        NSLog(@"receive VOD_PLAY_EVT_SELECT_TRACK_COMPLETE, trackIndex=%d , errorCode=%d", trackIndex, errorCode);
    }
}
```

步骤3：配置字幕样式。
字幕样式支持在播放前或者播放过程中配置。

```
// 详细参数配置请参考 API 文档
TXPlayerSubtitleRenderModel *model = [[TXPlayerSubtitleRenderModel alloc] init];
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.isBondFontStyle = NO;  // 设置字幕字体是否为粗体
model.fontColor = 0xFF000000; // 设置字幕字体颜色，默认白色不透明
[_txVodPlayer setSubtitleStyle:model];
```

21、字幕文本回调

> **注意：**
> 此功能播放器高级版 12.3 版本开始支持。

播放器高级版 SDK 在默认配置下是通过内置引擎渲染字幕和展示， 可通过修改配置支持回调文本，业务可以在获取到字幕文本后自行渲染和展示。现已支持 SRT 和 VTT 这两种格式的字幕。
详细用法如下：

步骤1：设置字幕文本回调

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
NSMutableDictionary<NSString *, id> *extInfoMap = [NSMutableDictionary dictionary];
[extInfoMap setObject:@(0) forKey:@"450"];
[_config setExtInfoMap:extInfoMap];
[_txVodPlayer setConfig:_config];
```

步骤2：添加和选择字幕

添加和选择字幕文档，请参见 外挂字幕 章节。

步骤3：注册监听字幕文本回调

选中字幕后可注册下面接口监听字幕文本内容。相关字段含义说明：

- `TXVodSubtitleData#trackIndex`，当前字幕的轨道 index；
- `TXVodSubtitleData#subtitleData`，实际字幕文本内容，当回调的 `subtitleData` 为空表示字幕为空，业务上封装展示即可；
- `TXVodSubtitleData` 类的其它字段暂时没有实际意义，不用关注。

```
// protocol TXVodPlayListener
- (void)onPlayer:(TXVodPlayer *)player subtitleData:(TXVodSubtitleData
*)subtitleData {
    long trackIndex = subtitleData.trackIndex; // 当前字幕的轨道 index
    NSString *data = subtitleData.subtitleData;    // 实际字幕文本内容
    // 根据需要展示 data 字幕文本内容
```

```
}
```

22. 多音轨切换

> **注意：**
> 此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持切换视频内置的多音轨。用法如下：

```
NSArray<TXTrackInfo *> *soundTrackArray = [_txVodPlayer getAudioTrackInfo];
for (int i = 0; i < soundTrackArray.count; i++) {
    TXTrackInfo *info = soundTrackArray[i];
    if (info.trackIndex == 0) {
        // 通过判断 trackIndex 或者 name 切换到需要的音轨
        [_txVodPlayer selectTrack:info.trackIndex];
    } else {
        // 其它字幕不需要的话， 进行 deselectTrack
        [_txVodPlayer deselectTrack:info.trackIndex];
    }
}
```

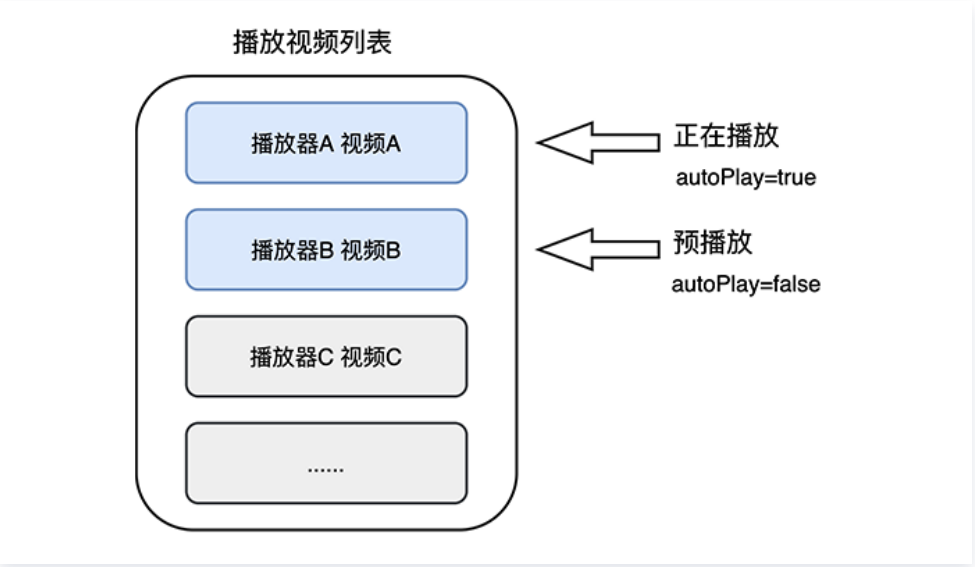
进阶功能使用

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，会占用下载带宽和线程资源。建议视频预播放并发个数控制在3个以内。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 startVodPlay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 startVodPlay: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 startVodPlay: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地应对网络的波动，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：**
> - **视频预下载会占用下载带宽和线程资源，建议进行队列控制，并发个数控制在3个以内。**
> - 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。

通过媒资 URL 预下载

通过媒资 URL 预下载视频代码示例如下：

```
//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];


- (void)onComplete:(int)taskID
               url:(NSString *)url {
    /**
      * 下载完成回调
      * taskID 下载任务ID
      * url 下载任务地址
      */
}
- (void)onError:(int)taskID
            url:(NSString *)url
          error:(NSError *)error {
    /**
      * 下载错误回调
      * taskID 下载任务ID
      * url 下载任务地址
      * error 下载失败的错误信息
      */
}
```

通过媒资 FileId 预下载

> **注意：**
> 通过 fileId 预下载从 11.3 版本开始支持。

通过 fileId 预下载是耗时操作， 请不要在主线程调用，否则会抛出非法调用异常。startPreload 时传入的 preferredResolution 要和启播时设置的优先启播分辨率保持一致，否则将达不到预期的效果。 `onStart` 回调的 URL

可以保存起来，传给播放器播放。使用示例如下：

```
//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];

TXPlayerAuthParams *params = [[TXPlayerAuthParams alloc] init];
params.appId = ${appId};
params.fileId = @"${fileId}";
params.sign = @"${psign}";
// 注意：  耗时操作，请不要在主线程调用！在主线程调用将会抛出非法调用异常。
int taskID = [[TXVodPreloadManager sharedManager] startPreload:params
                                                  preloadSize:10
                                            preferredResolution:1920*1080
                                                       delegate:self]; //
TXVodPreloadManagerDelegate

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

//预下载TXVodPreloadManagerDelegate代理方法实现
- (void)onStart:(int)taskID
         fileId:(NSString *)fileId
            url:(NSString *)url
          param:(NSDictionary *)param {
          /**
            * 启动下载（此方法在换链成功以后，启动下载之前回调）
            * taskID 下载任务ID
            * fileId 下载视频的 fileId。URL方式缓存时，此参数为nil
            * url 下载任务地址
            * param 附加参数
            */
}
- (void)onComplete:(int)taskID
               url:(NSString *)url {
           /**
```

```
             * 下载完成回调
             * taskID 下载任务ID
             * url 下载任务地址
             */
}
- (void)onError:(int)taskID
            url:(NSString *)url
          error:(NSError *)error {
          /**
             * 下载错误回调
             * taskID 下载任务ID
             * url 下载任务地址
             * error 下载失败的错误信息
             */
}
```

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。如果是加密视频，通过播放器 SDK 下载后的视频在本地保持为加密状态，仅可通过腾讯云播放器 SDK 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 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：**
> 视频下载支持下载 MP4 和 HLS 视频，对应嵌套 HLS 视频，需要指定偏好清晰度（preferredResolution）。

步骤1：准备工作

SDK 初始化时，设置全局存储路径，用于视频下载，预加载，和缓存等功能。用法如下：

```
NSString *cachesDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES)            firstObject];
NSString downloadPath = [NSString stringWithFormat:@"%@/txdownload",cachesDir];
[TXPlayerGlobalSetting setCacheFolderPath:downloadPath];
```

`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];
```

设置下载使用的 httpHeader

根据业务需求进行配置，播放器启动下载时将发送给服务端， 播放器 12.2 版本开始支持。

```
NSDictionary *httpHeader = [[NSMutableDictionary alloc] init];
[downloader setHeaders:httpHeader];  // 设置下载httpHeader
```

步骤2：开始下载

开始下载有两种方式：fileid 和 URL。

fileid 方式

fileid 下载至少需要传入 appId 和 fileId，userName 不传入具体值时，默认为 “default”。**注意：加密视频只能通过 Fileid 下载。**

```
TXVodDownloadDataSource *source = [[TXVodDownloadDataSource alloc] init];
source.appId = 1252463788;
source.fileId = @"4564972819220421305";
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
source.pSign = @"xxxxxxxxxx";

// 指定下载清晰度
// 可用枚举值有：240p:TXVodQuality240P, 360p:TXVodQuality360P,
480p:TXVodQuality480P, 540p:TXVodQuality540P, 720p:TXVodQuality720P,
// 1080p:TXVodQuality1080P,2K:TXVodQuality2K,4K:TXVodQuality4K
// quality参数可以自定义，取分辨率宽高最小值(如分辨率为1280*720, 期望下载此分辨率的流，
quality传入 TXVodQuality720P)
// 播放器 SDK 会选择小于或等于传入分辨率的流进行下载
source.quality = TXVodQuality720P;  // 720p

// ** 注意如果是使用旧的 v2 协议下载，请通过 TXVodDownloadDataSource中的 auth 属性来设置
appId 和 fileId 这些参数 **
// source.auth = auth;  ** 默认无需设置 **

[downloader startDownload:dataSource];
```

URL 方式

至少需要传入下载地址 URL。preferredResolution 取值为视频分辨率宽和高的乘积：preferredResolution=width * height。如果是嵌套 HLS 格式，preferredResolution 不传入具体值时，默认值为921600。userName 不传入具体值时，默认为"default"。私有加密请使用 fileid 形式。

```
[downloader startDownloadUrl:@"" resolution:@921600 userName:@""];
```

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。
-[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 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。

下载错误码

错误码	数值	含义说明
TXDownloadSuccess	0	下载成功。
TXDownloadAuthFaild	-5001	向云点播控制台请求视频信息失败，建议检查fileId、psign参数是否正确。
TXDownloadNoFile	-5003	无此清晰度文件。
TXDownloadFormatError	-5004	下载文件格式不支持。
TXDownloadDisconnet	-5005	网络断开，建议检查网络是否正常。
TXDownloadHlsKeyError	-5006	获取 HLS 解密 Key 失败。
TXDownloadPathError	-5007	下载目录访问失败，建议检查是否有访问下载目录的权限。

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 startDownloadUrl:]` 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

1. 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。

```
// getDownloadMediaInfoList 是耗时接口，请不要在主线程调用
NSArray<TXVodDownloadMediaInfo *> *array = [[[TXVodDownloadManager shareInstance]
getDownloadMediaInfoList] mutableCopy];
// 获取默认 “default” 用户的下载列表
for (TXVodDownloadMediaInfo *info in array) {
    if ([info.userName isEqualToString:@"default"]) {
        // 保存 “default” 用户的下载列表
    }
}
```

2. 获取 FileId 或 URL 相关的下载信息：

2.1 通过接口 `-[TXVodDownloadManager getDownloadMediaInfo:]` 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 *sourceMediaInfo = [[TXVodDownloadMediaInfo alloc]
init];
TXVodDownloadDataSource *dataSource = [[TXVodDownloadDataSource alloc] init];
dataSource.appId = 1252463788;
dataSource.fileId = @"4564972819220421305";
dataSource.pSign = @"psignxxxx";
dataSource.quality = TXVodQualityHD;
sourceMediaInfo.dataSource = dataSource;
// getDownloadMediaInfo 是耗时接口，请不要在主线程调用
TXVodDownloadMediaInfo *downloadMediaInfo =  [[TXVodDownloadManager
shareInstance] getDownloadMediaInfo:sourceMediaInfo];

// 获取下载文件总大小，单位：Byte，只针对 fileid 下载源有效。
// 备注：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法
获取。
downloadMediaInfo.size; //  获取下载文件总大小
downloadMediaInfo.duration;   // 获取总时长
downloadMediaInfo.playableDuration; // 获取已下载的可播放时长
downloadMediaInfo.progress;   // 获取下载进度
downloadMediaInfo.playPath;   // 获取离线播放路径，传给播放器即可离线播放
downloadMediaInfo.downloadState;  // 获取下载状态，具体参考 STATE_xxx 常量
[downloadMediaInfo isDownloadFinished];  // 返回 YES 表示下载完成
```

2.2 获取某个 URL 相关下载信息，需要传入 URL 信息即可。

```
// 获取某个 fileId 相关下载信息
TXVodDownloadMediaInfo *sourceMediaInfo = [[TXVodDownloadMediaInfo alloc]
init];
mediaInfo.url = @"videoURL";
TXVodDownloadMediaInfo *downloadMediaInfo =  [[TXVodDownloadManager
shareInstance] getDownloadMediaInfo:sourceMediaInfo];
```

3. 删除下载信息和相关文件：
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。

步骤6：下载后离线播放

下载后的视频支持无网络的情况下进行播放，无需进行联网。下载完成后，即可进行播放。

```
// getDownloadMediaInfoList 是耗时接口，请不要在主线程调用
NSArray<TXVodDownloadMediaInfo *> *mediaInfoList = [[TXVodDownloadManager
shareInstance] getDownloadMediaInfoList];
TXVodDownloadMediaInfo *mediaInfo = [mediaInfoList firstObject]; // 根据情况找到当前的
media 对象
if (mediaInfo.downloadState == TXVodDownloadMediaInfoStateFinish) { // 判断是否下载完成
    [self.player startVodPlay:mediaInfo.playPath];
}
```

> **注意：**
> 离线下载播放时，一定要通过获取下载列表并通过下载列表视频对象 `TXVodDownloadMediaInfo` 的 `PlayPath` 进行播放，切勿直接保存 PlayPath 对象。

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。
在腾讯云控制台提取到 appId， 加密视频的 fileId 和 psign 后，可以通过下面的方式进行播放：

```
TXPlayerAuthParams *p = [TXPlayerAuthParams new];
p.appId = 1252463788; // 腾讯云账户的 appId
p.fileId = @"4564972819220421305"; // 视频的 fileId
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
p.sign = @"psignxxxxx"; // 播放器签名
[_txVodPlayer startVodPlayWithParams:p];
```

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 = [[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
```

- **启播前指定分辨率**

播放 HLS 的多码率视频源，如果您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
```

- **启播前指定媒资类型**

当提前知道播放的媒资类型时，可以通过配置 `TXVodPlayConfig#setMediaType` 减少播放器 SDK 内部播放类型探测，提升启播速度。

> **注意：**
> `TXVodPlayConfig#setMediaType` 11.2 版本开始支持。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMediaType:MEDIA_TYPE_FILE_VOD];    // 用于提升MP4启播速度
// [_config setMediaType:MEDIA_TYPE_HLS_VOD];    // 用于提升HLS启播速度
[_txVodPlayer setConfig:_config];
```

- **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
```

- **启播前指定优先启播音轨**

> **注意：**
> 播放器高级版 12.3 版本开始支持。

当提前知道播放的音轨名称时，可以通过配置 `TXVodPlayConfig#setPreferredAudioTrack` 在启播前指定优先启播音轨。

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setPreferredAudioTrack:@"audioTrackName"];  // audioTrackName为实际音轨名称
[_txVodPlayer setConfig:_config];  // 把config 传给 _txVodPlayer
```

6. HttpDNS 解析服务

移动解析（HTTPDNS）基于 HTTP 协议向 DNS 服务器发送域名解析请求，替代了基于 DNS 协议向运营商 Local DNS 发起解析请求的传统方式，可避免 Local DNS 造成域名劫持和跨网访问问题，解决移动互联网服务中域名解析异常带来的视频播放失败困扰。

> ⚠ **注意：**
> HttpDNS 解析服务从 10.9 版本开始支持。

1. 开通 HTTPDNS 解析服务。
   您可以选择腾讯云或其它云提供商，开通 HTTPDNS 解析服务，确保开通成功后， 再集成到播放 SDK。
2. 在播放 SDK 接入 HTTPDNS 解析服务。
   下面以接入 腾讯云 HTTPDNS 为例子，展示如何在播放器 SDK 接入：

```
// 步骤1： 打开 HttpDNS 解析开关
[TXLiveBase enableCustomHttpDNS:YES];
// 步骤2： 实现 HttpDNS 解析代理：TXLiveBaseDelegate#onCustomHttpDNS
- (void)onCustomHttpDNS:(NSString *)hostName ipList:(NSMutableArray<NSString *>
*)list {
    // 播放器 SDK 会把 hostName 回调给业务， 业务可以根据实际需要决定是否把 hostName 解析 ip，
    // 如果返回空 ip，则 SDK 内部不会对此次网络请求使用 httpdns
    // 把 hostName 解析到 ip 地址后，保存到 iPList，返回给 SDK 内部。注意：这里不要进行耗时的
异步操作。
    // MSDKDnsResolver 是腾讯云提供的 HTTPDNS SDK 解析接口
    // NSArray *result = [[MSDKDns sharedInstance] WGGetHostByName:hostName];
    NSString *ip = nil;
    if (result && result.count > 1) {
        if (![result[1] isEqualToString:@"0"]) {
            ip = result[1];
        } else {
            ip = result[0];
        }
    }
        [list addObject:ip];
}

// 步骤3： 设置 HttpDNS 解析代理
[TXLiveBase sharedInstance].delegate = self;
```

7. HEVC 自适应降级播放

播放器支持同时传入 HEVC 和其它视频编码格式比如： H.264 的播放链接， 当播放机型不支持 HEVC 格式时，将自动降级为配置的其它编码格式（如： H.264 ）的视频播放。
注意：播放器高级版 11.7 版本开始支持。

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

NSDictionary *dic = @{
    VOD_KEY_VIDEO_CODEC_TYPE:@(kCMVideoCodecType_HEVC),  // 指定原始 HEVC 视频编码类型
    VOD_KEY_BACKUP_URL:@"${backupPlayUrl}"  // 设置 H.264 格式等备选播放链接地址
    VOD_KEY_BACKUP_URL_MEDIA_TYPE: @(TUI_MEDIA_TYPE_AUTO)};  // 非必选，播放器 SDK
12.0 版本开始支持。设置备选播放资源的 MediaType，默认为TXVCubePlayerParams.MEDIA_TYPE_AUTO
[_txVodPlayer setExtentOptionInfo:dic];
```

```
// 设置原始 HEVC 播放链接
[_txVodPlayer startVodPlay:@"${hevcPlayUrl}"];
```

8、音量均衡

播放器支持在播放音频时自动调整音量，使得所有音频的音量保持一致。这可以避免某些音频过于响亮或过于安静的问题，提供更好的听觉体验。通过 `TXVodPlayer#setAudioNormalization` 设置音量均衡，响度范围：-70～0 (LUFS)，同时支持自定义数值。
注意：播放器高级版 11.7 版本开始支持。

```
/**
 可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）
 关：AUDIO_NORMALIZATION_OFF
 开：AUDIO_NORMALIZATION_STANDARD（标准）
    AUDIO_NORMALIZATION_LOW （低）
    AUDIO_NORMALIZATION_HIGH（高）
 可填自定义数值：从低到高，范围-70 - 0 LUFS
*/
[_txVodPlayer setAudioNormalization:AUDIO_NORMALIZATION_STANDARD];       //启动音量均衡

[_txVodPlayer setAudioNormalization:AUDIO_NORMALIZATION_OFF];                //关闭音
量均衡
```

9、MP4 视频本地加密

开启 MP4 本地加密后，播放器在缓存文件时，将对数据进行加密存储，加密视频文件将只能由播放器来解密播放，无法用第三方播放器播放。

> **注意：**
> - 开启 MP4本地加密并开始播放后，就无法再更改加密设置了，除非文件被清理并重新缓存。
> - 此功能播放器高级版 12.2 版本开始支持。

```
//前置条件：设置播放器的全局缓存目录，此配置在项目中设置一次即可
NSString *cachesDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES)             firstObject];
NSString downloadPath = [NSString stringWithFormat:@"%@/txdownload",cachesDir];
[TXPlayerGlobalSetting setCacheFolderPath:downloadPath];

TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
_config.encryptedMp4Level = MP4_ENCRYPTION_LEVEL_L2;      // 设置使用mp4本地加密播放和存
储
[_txVodPlayer setConfig:_config];
```

10、画中画功能

播放器 SDK 提供 基础画中画 和 高级画中画 两种能力。基础画中画支持常规视频播放，而高级画中画（仅限播放器高级版）在此基础上全面升级，额外支持加密视频播放、离线播放，并优化切换逻辑，实现前台秒切画中画，无需等待加载，体验更流畅。更多功能详情，请参考 高级画中画使用指引 。

> **注意：**
> 高级画中画功能需使用播放器高级版。

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_END	2006	视频播放结束
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。
VOD_PLAY_EVT_SEEK_COMPLETE	2019	Seek 完成，10.3版本开始支持。
VOD_PLAY_EVT_LOOP_ONCE_COMPLETE	6001	循环播放，一轮播放结束（10.8 版本开始支持）。
VOD_PLAY_EVT_HIT_CACHE	2002	启播时命中缓存事件（11.2 版本开始支持）。
VOD_PLAY_EVT_VIDEO_SEI	2030	收到 SEI 帧事件（ 播放器高级版11.6 版本开始支持）。
VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK	2031	发生 HEVC 降级播放（播放器高级版12.0版本开始支持）。
VOD_PLAY_EVT_VOD_PLAY_FIRST_VIDEO_PACKET	2017	播放器收到首帧数据包事件（12.0 版本开始支持）。

SEI 帧

SEI（Supplemental Enhancement Information）帧是一种用于传递附加信息的帧类型，播放器高级版会解析视频流中的 SEI 帧，通过 VOD_PLAY_EVT_VIDEO_SEI 事件回调， 注意：播放器高级版 11.6 版本开始支持。

```
-(void) onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:
(NSDictionary*)param {
    if (EvtID == VOD_PLAY_EVT_VIDEO_SEI) {
        int seiType = [param objectForKey:EVT_KEY_SEI_TYPE]; //  the type of video
```

```
SEI
        int seiSize = [param objectForKey:EVT_KEY_SEI_SIZE]; // the data size of
video SEI
        NSData *seiData = [param objectForKey:EVT_KEY_SEI_DATA]; // the byte array
data of video SEI
    }
}
```

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
EVT_KEY_WATER_MARK_TEXT	幽灵水印文本内容（11.6 版本开始支持）

```
-(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
    }
}
```

幽灵水印

幽灵水印内容在播放器签名中填写，经云点播后台，最终展示到播放端上，整个传输链路过程由云端和播放端共同协作，确保水印的安全。在播放器签名中 配置幽灵水印教程 。幽灵水印的内容在收到播放器的 `VOD_PLAY_EVT_GET_PLAYINFO_SUCC` 事件后， 通过 `[param objectForKey:@"EVT_KEY_WATER_MARK_TEXT"]` 获取。详细使用教程参见 超级播放器组件 > 幽灵水印 。
注意： 播放器 11.6 版本开始支持。

播放错误事件

> **说明：**
> [-6004 , -6010] 错误事件 11.0 版本开始支持。

事件 ID	数值	含义说明
PLAY_ERR_NET_DISCONNECT	-2301	错误导致重试亦不能恢复正常播放。例如：网络异常或下载数据错误，导致解封装超时或失败。
PLAY_ERR_HLS_KEY	-2305	HLS 解密 key 获取失败
VOD_PLAY_ERR_SYSTEM_PLAY	-6004	系统播放器播放错误

_FAIL		
VOD_PLAY_ERR_DECODE_VIDEO_FAIL	-6006	视频解码错误，视频格式不支持。
VOD_PLAY_ERR_DECODE_AUDIO_FAIL	-6007	音频解码错误，音频格式不支持。
VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL	-6008	字幕解码错误
VOD_PLAY_ERR_RENDER_FAIL	-6009	视频渲染错误
VOD_PLAY_ERR_PROCESS_VIDEO_FAIL	-6010	视频后处理错误
VOD_PLAY_ERR_GET_PLAYINFO_FAIL	-2306	获取点播文件信息失败，建议检查AppId、FileId或Psign填写是否正确。

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，单位 KBps。
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 bps。
AUDIO_BITRATE	当前流媒体的音频码率，单位 bps。
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 bps
    int videoBitRate = [[param objectForKey:@"VIDEO_BITRATE"] intValue];
    //获取当前流媒体的音频码率，单位 bps
    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"];
}
```

其它功能使用

HLS 直播视频源播放

播放器高级版本支持播放 HLS 直播视频源，从 11.8 版本开始支持带 HLS EVENT 直播视频源。用法如下：

```
TXVodPlayConfig *_config = [[TXVodPlayConfig alloc]init];
[_config setMediaType:MEDIA_TYPE_HLS_LIVE];    // 指定HLS直播媒资类型
[_txVodPlayer setConfig:_config];
[_txVodPlayer startVodPlay:${YOUR_HSL_LIVE_URL}];
```

场景化功能

1. 动态设置 AudioSession

有时候需要根据场景来动态设置播放音频输出方式，特别是对于 iPhone 来说，天然支持多音频播放及后台模式。因此，我们根据用户的场景支持了以下三种主要的模式：

- AVAudioSessionCategoryPlayback：后台独占播放。
- AVAudioSessionCategoryPlayAndRecord：后台独占播放。
- AVAudioSessionCategoryAmbient：混合播放。

可以根据所处的场景，利用上面的模式来设置 AudioSession 的 Category 和 Option 来达到自己的目的。下面罗列了两种场景的设置(以下设置可以根据自己的场景进行动态调整和设置)：

场景一：播放列表场景（视频播放需要支持列表里静音播放，并且不打断外部音频播放）。

```
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback
withOptions:AVAudioSessionCategoryOptionInterruptSpokenAudioAndMixWithOthers
error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];
```

场景二：播放详情场景（视频详情有声音，并且暂时打断外部音频，当视频播放完成以后，就恢复外部音频）。

```
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryAmbient
withOptions:AVAudioSessionCategoryOptionMixWithOthers error:nil];
```

```
[[AVAudioSession sharedInstance] setActive:NO
withOptions:AVAudioSessionSetActiveOptionNotifyOthersOnDeactivation error:nil];
```

2. 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。

3. 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，我们会提供更多的基于用户场景的组件。您可以通过 Player_iOS 下载体验。

Android 端集成
集成指引

最近更新时间：2024-04-26 11:02:11

本文主要介绍如何快速地将腾讯云视立方 · 播放器 SDK 集成到您的项目中，不同版本的 SDK 集成方式都通用，按照如下步骤进行配置，就可以完成 SDK 的集成工作。

开发环境要求

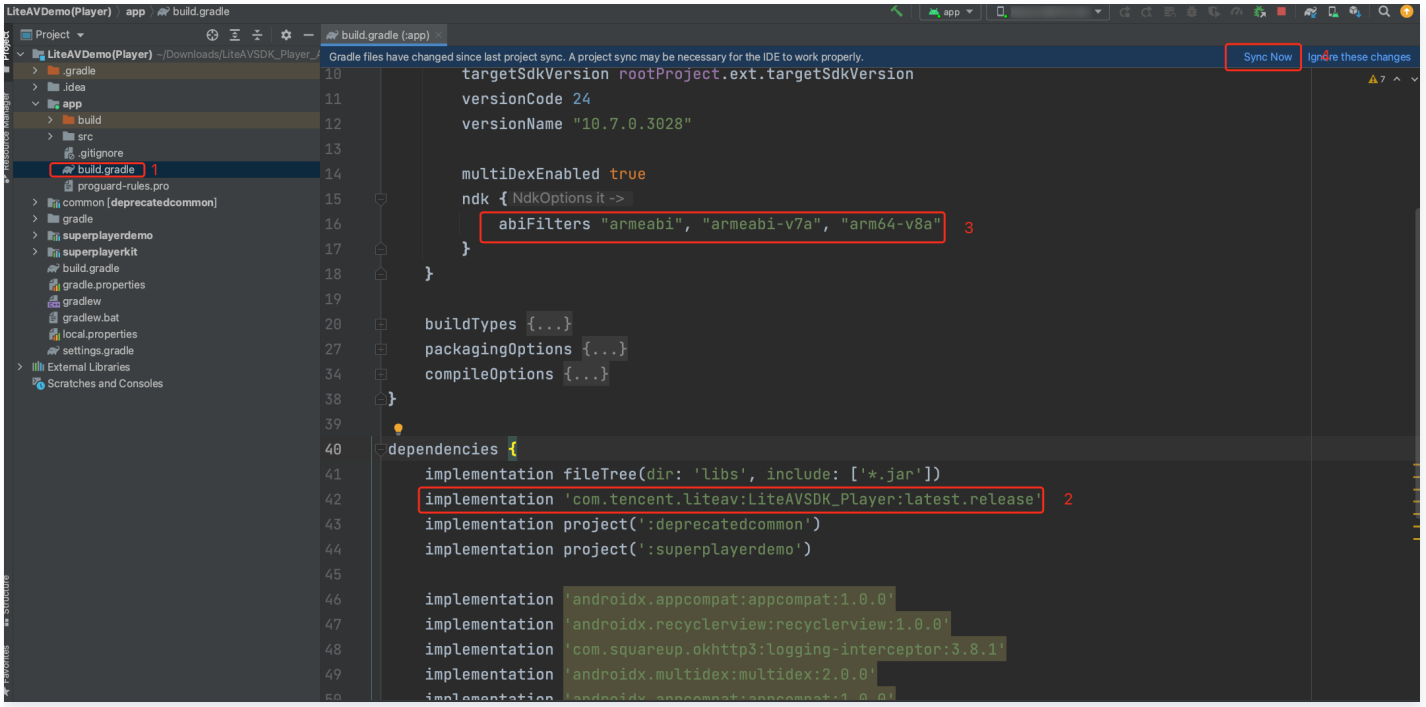
- Android Studio 2.0+。
- Android 4.1（SDK API 16）及以上系统。

集成 SDK（aar）

您可以选择使用 Gradle 自动加载的方式，或者手动下载 aar 再将其导入到您当前的工程项目中。

方法一：自动加载（aar）

播放器 SDK 已经发布到 mavenCentral 库（播放高级版本 mavenCentral 库），您可以通过在 gradle 配置 mavenCentral 库，自动下载更新 LiteAVSDK_Player。
只需要用 Android Studio 打开需要集成 SDK 的工程，然后通过简单的四个步骤修改 `build.gradle` 文件，就可以完成 SDK 集成：



1. 打开工程根目录下的 build.gradle，添加mavenCentral库。

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

2. 打开 app 下的 build.gradle，在 dependencies 中添加 LiteAVSDK_Player 的依赖。

```
dependencies {
  // 此配置默认集成 LiteAVSDK_Player 最新版本
 implementation 'com.tencent.liteav:LiteAVSDK_Player:latest.release'
  // 集成历史版，如：10.7.0.13038 版本，可通过下面方式集成
  // implementation 'com.tencent.liteav:LiteAVSDK_Player:10.7.0.13038'
}
```

如果您需要集成播放高级版本（Premium）SDK，在 dependencies 中添加如下依赖：

```
 dependencies {
  // 此配置默认集成 LiteAVSDK_Player_Premium 最新版本
 implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:latest.release'
  // 集成历史版，如：10.7.0.13038 版本，可通过下面方式集成
  // implementation 'com.tencent.liteav:LiteAVSDK_Player_Premium:10.7.0.13038'
}
```

3. 在 defaultConfig 中，指定 App 使用的 CPU 架构（目前 LiteAVSDK_Player 支持 armeabi 、 armeabi-v7a 和 arm64-v8a）。

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

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。

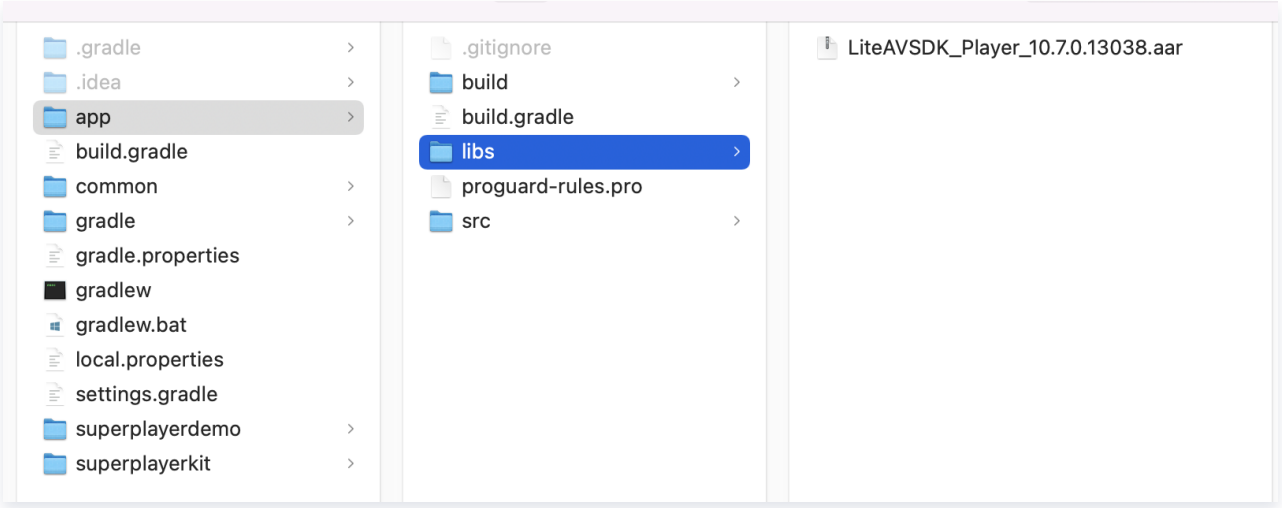
方法二：手动下载（aar）

如果您的网络连接 mavenCentral 有问题，也可以手动下载 SDK 集成到工程里：

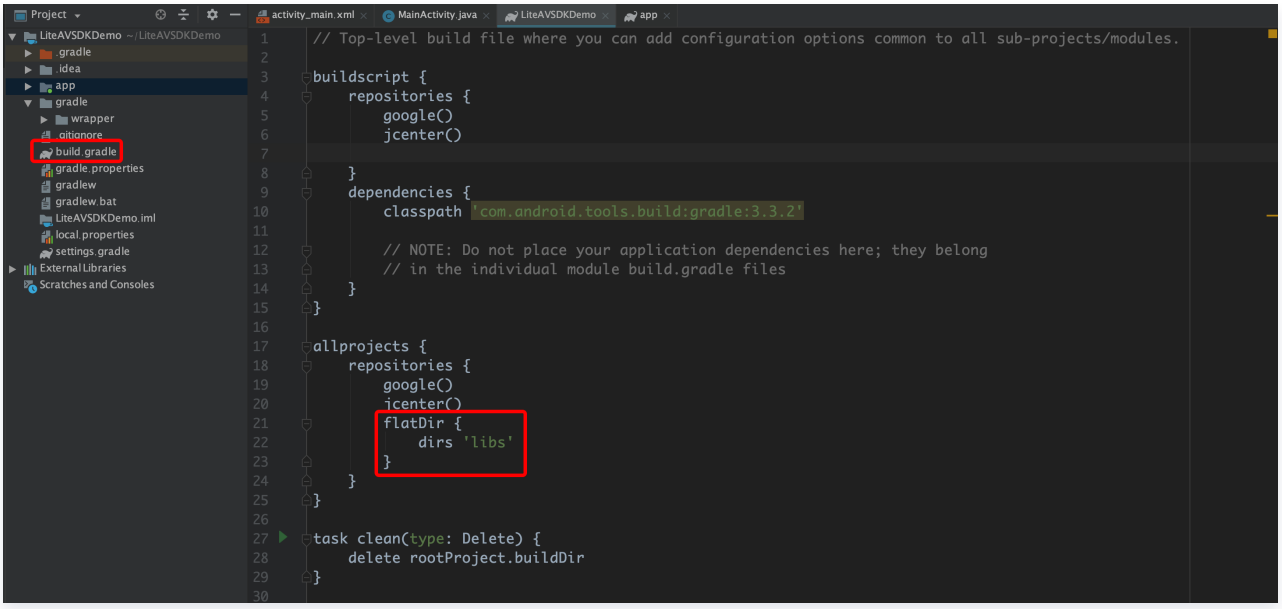
1. 下载 LiteAVSDK_Player ，下载完成后进行解压。

   如果您需要集成播放高级版本（Premium）SDK，请 单击此处 下载。

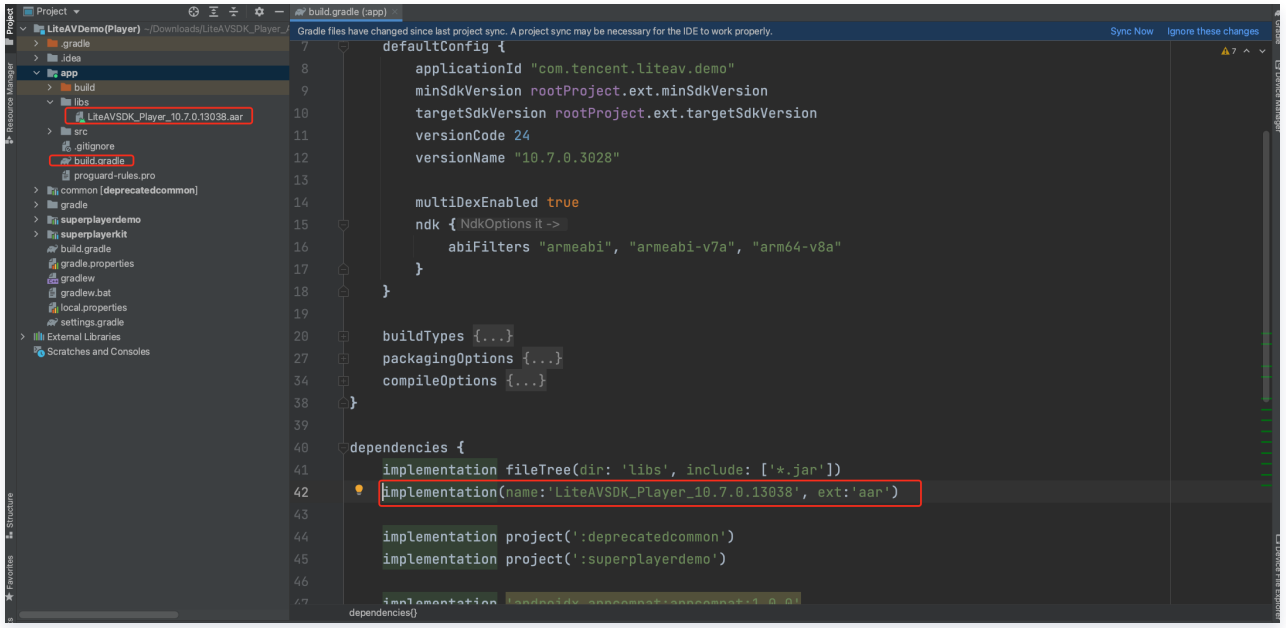
2. 将下载文件解压之后 SDK 目录下的 aar 文件拷贝到工程的 **app/libs** 目录下：



3. 在工程根目录下的 build.gradle 中，添加 **flatDir**，指定本地仓库路径。



4. 添加 LiteAVSDK_Player 依赖，在 app/build.gradle 中，添加引用 aar 包的代码。




```
implementation(name:'LiteAVSDK_Player_10.7.0.13038', ext:'aar')
```

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

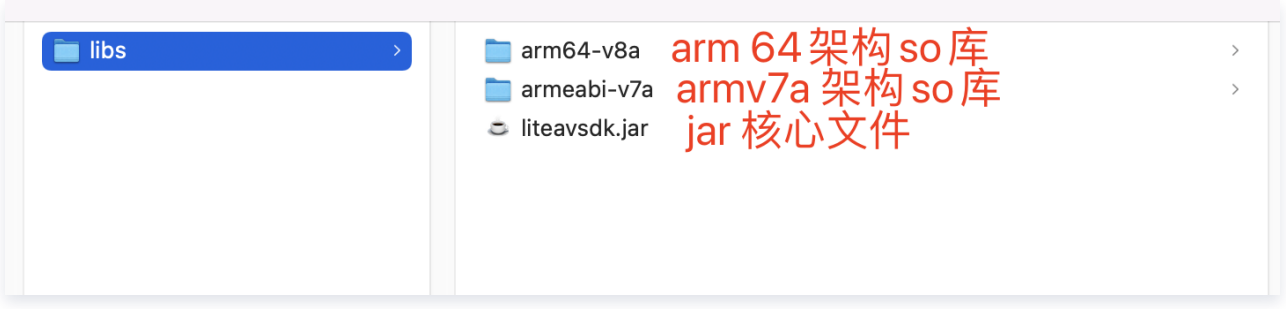
```
defaultConfig {
 ndk {
     abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
 }
}
```

6. 单击 Sync Now 按钮同步 SDK，完成 LiteAVSDK 的集成工作。

集成 SDK（jar）

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：

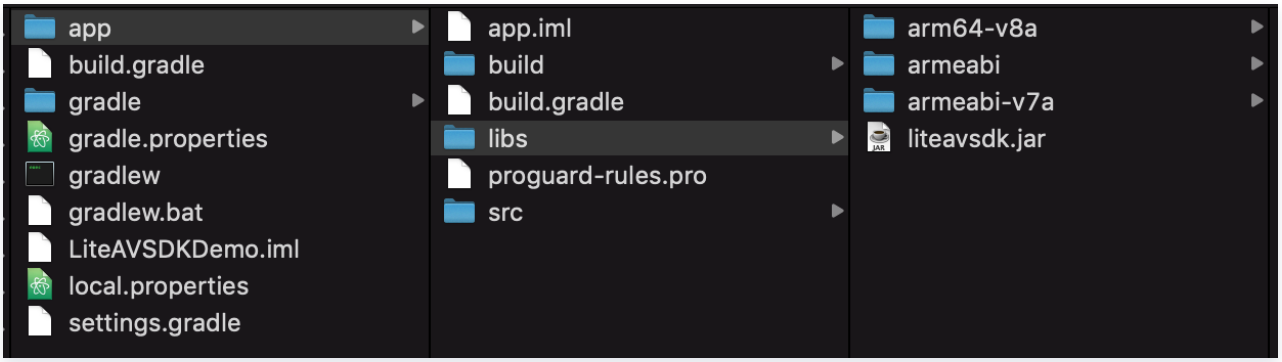
1. 下载 LiteAVSDK_Player ，下载完成后进行解压。在 SDK 目录下找到 `LiteAVSDK_Player_xxx.zip`（其中 `xxx` 为 LiteAVSDK 的版本号），解压后得到 libs 目录，里面主要包含 jar 文件和 so 文件夹，文件清单如下：



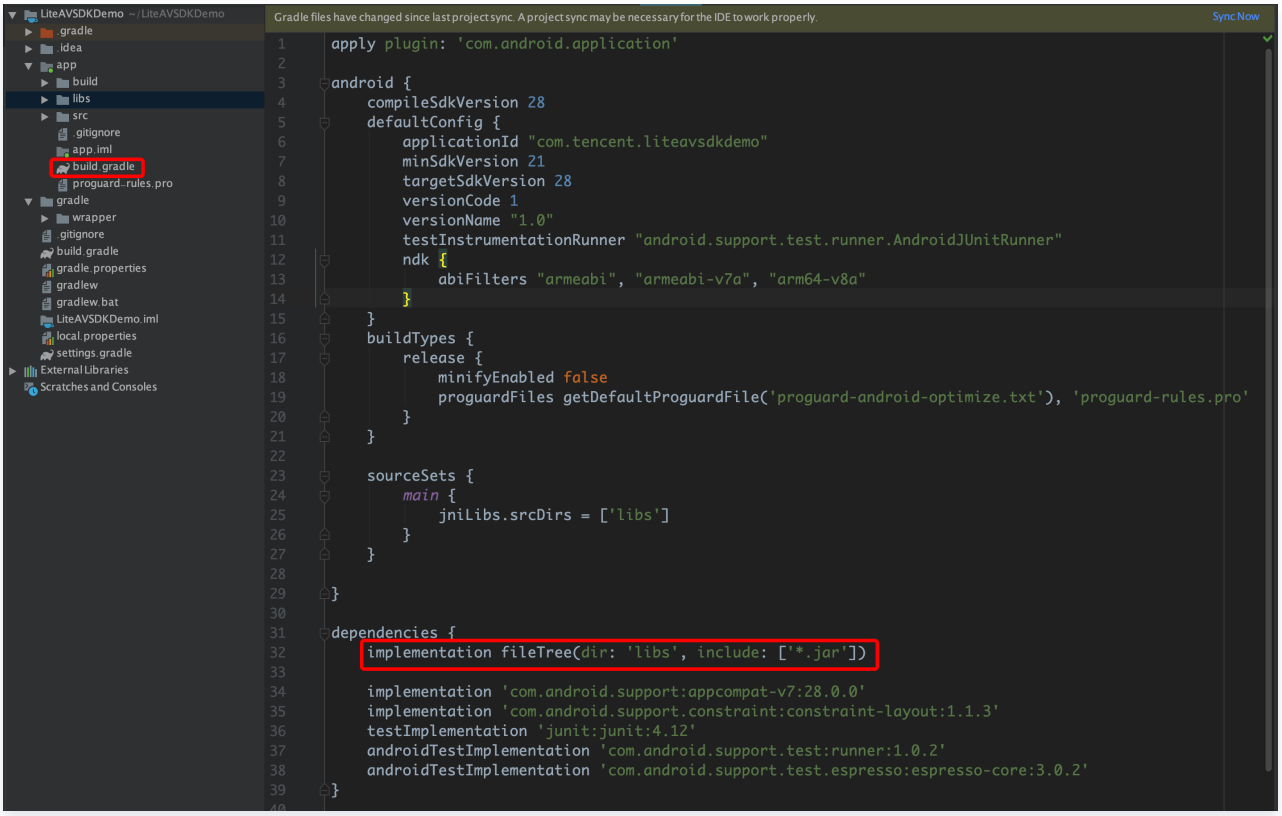
如果您还需要 armeabi 架构 so，复制一份 armeabi-v7a 目录，重命名为 armeabi 即可。

如果您需要集成播放高级版本（Premium）SDK，请 单击此处 下载。

2. 将解压得到的 jar文件和 armeabi、armeabi-v7a、arm64-v8a 文件夹拷贝到 `app/libs` 目录下。

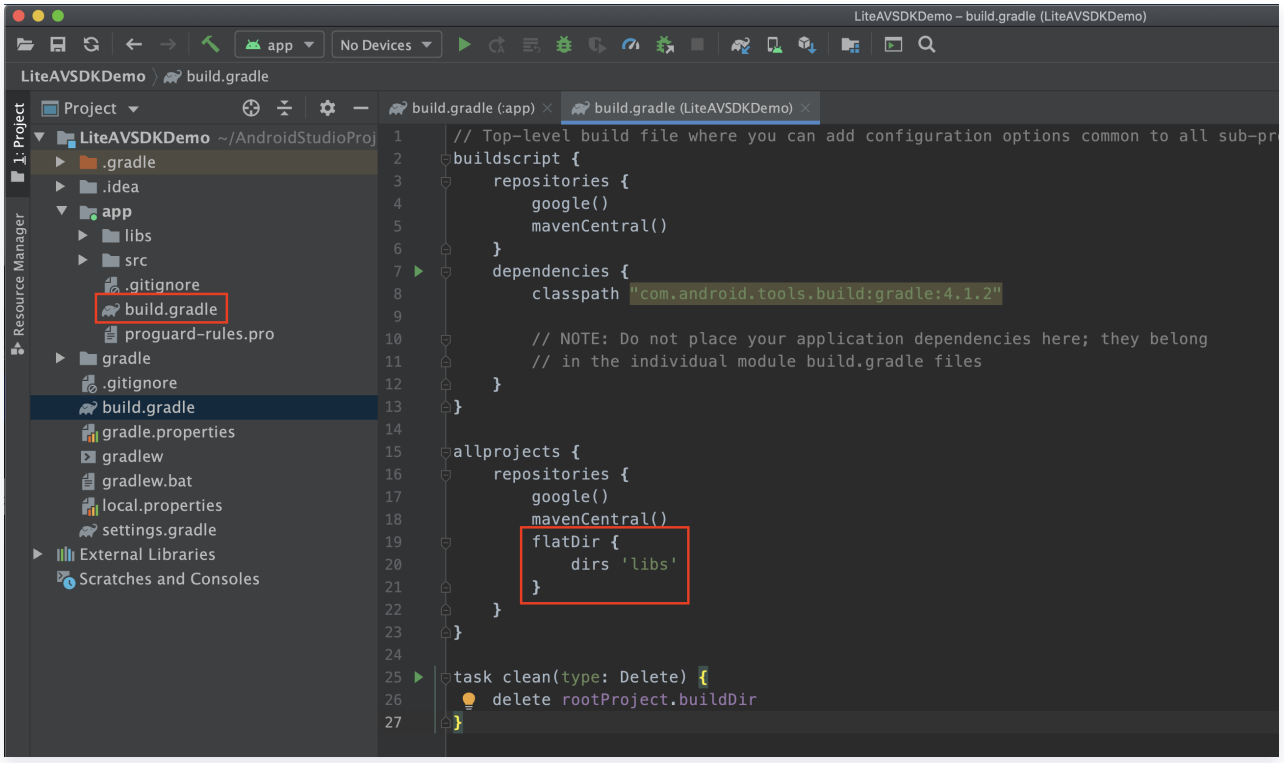


3. 在 `app/build.gradle` 中，添加引用 jar 库的代码。

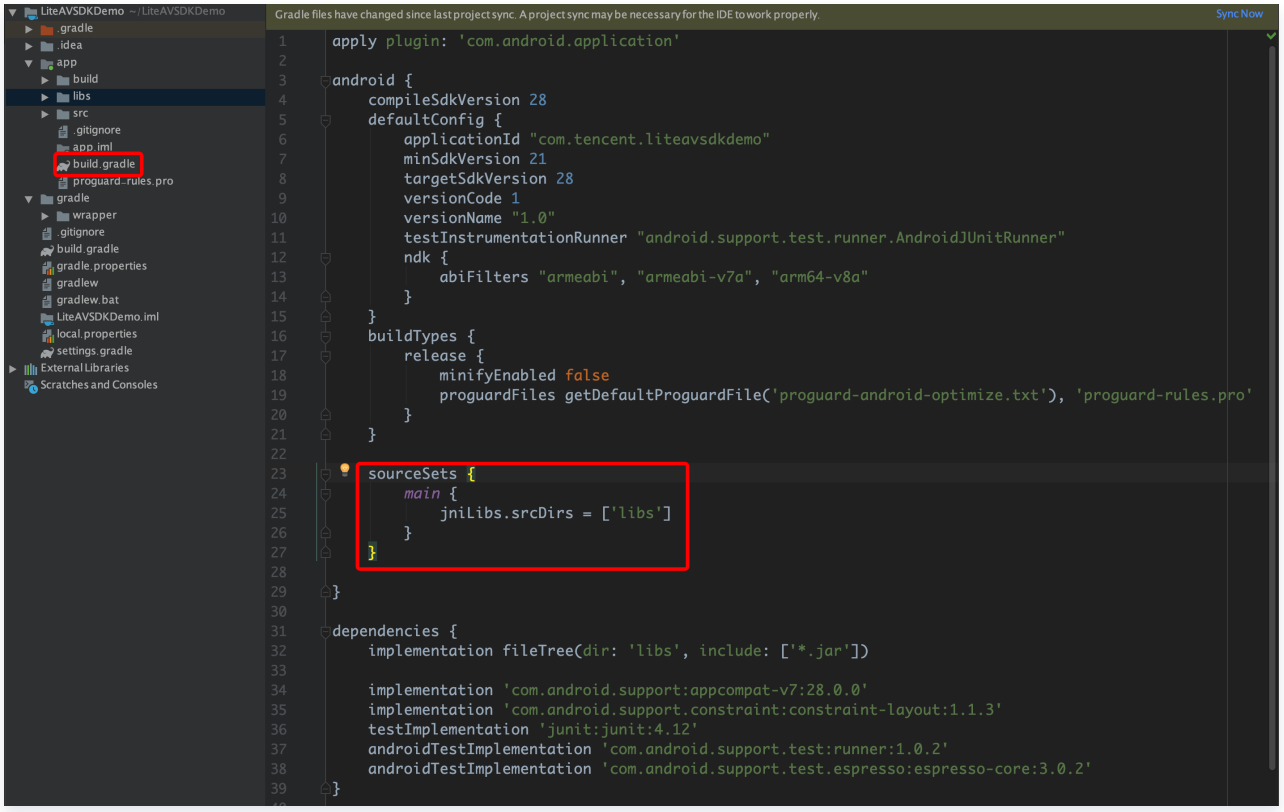



```
dependencies{
 implementation fileTree(dir:'libs',include:['*.jar'])
}
```

4. 在工程根目录下的 build.gradle 中，添加 **flatDir**，指定本地仓库路径。



5. 在 `app/build.gradle` 中，添加引用 so 库的代码。



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

```
defaultConfig {
```

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

7. 单击 Sync Now 按钮同步 SDK，完成 LiteAVSDK 的集成工作。

配置 App 打包参数

> **注意：**
> 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：

```
// 新客户集成 SDK 建议把下面的脚本加上，可以优化包体积
packagingOptions {
    exclude "lib/armeabi/libijkhlscache-master.so"
    exclude "lib/armeabi-v7a/libijkhlscache-master.so"
    exclude "lib/arm64-v8a/libijkhlscache-master.so"
}
```

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

网络安全配置允许 App 发送 http 请求

出于安全考虑，从 Android P 开始，Google 要求 App 的请求都使用加密链接。播放器 SDK 会启动一个 localsever 代理 http 请求，如果您的应用 **targetSdkVersion 大于或等于28**，可以通过 网络安全配置 来开启允许向127.0.0.1发送 http 请求。否则播放时将出现 "java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted" 错误，导致无法播放视频。配置步骤如下：

1. 在项目新建 res/xml/network_security_config.xml 文件，设置网络安全性配置。

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
```

```
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
</network-security-config>
```

2. 在 AndroidManifest.xml 文件下的 application 标签增加以下属性。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
                    ... >
        ...
    </application>
</manifest>
```

设置混淆规则

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

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

配置 License 授权

单击 License 申请 获取测试用 License，不配置 License 将会播放时视频失败，具体操作请参见 测试版 License 。您会获得两个字符串：一个字符串是 licenseURL，另一个字符串是解密 key。
获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

常见问题

项目里面同时集成了直播 SDK/实时音视频/播放器等 LiteAVSDK 系列的多个 SDK 报符号冲突问题怎么解决?

如果集成了2个或以上产品（直播、播放器、TRTC、短视频）的 LiteAVSDK 版本，编译时会出现库冲突问题，因为有些 SDK 底层库有相同符号文件，这里建议只集成一个全功能版 SDK 可以解决，直播、播放器、TRTC、短视频这些都包含在一个 SDK 里面。具体请参见 SDK 下载 。

点播场景

最近更新时间：2025-06-04 15:55:52

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

- 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。
- 播放器推出短视频组件、画中画2.0、VR 播放等高级组件，功能介绍和使用指引请参见 移动端高级功能 。

SDK 集成

步骤1：集成 SDK 开发包

下载和集成 SDK 开发包，请参见同目录下的 SDK 集成指引 。

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 播放器 License 获取相关授权。
- 获取到 License 信息后，在调用 SDK 的相关接口前，需要初始化配置 License，详细教程请参见 配置查看 License 。

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 `startVodPlay` 函数即可。

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


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

// 从 11.8 版本开始支持播放器 content:// URI 视频资源和 asset 目录视频资源
// 播放 content:// URI 视频资源
String localFile = "content://xxx/xxx/video.mp4";
mVodPlayer.startVodPlay(localFile);

// 播放 asset 目录视频资源， 传入的地址必须以 asset:// 开头
String localFile = "asset://video.mp4";
mVodPlayer.startVodPlay(localFile);
```

通过 FileId 方式

```
// 推荐使用下面的新接口
```

```
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // 腾讯云账户的
appId
    "4564972819220421305", // 视频的 fileId
    "psignxxxxxxx"); // 播放器签名
mVodPlayer.startVodPlay(playInfoParam);

// 旧接口，不推荐使用
TXPlayerAuthBuilder authBuilder = new TXPlayerAuthBuilder();
authBuilder.setAppId(1252463788);
authBuilder.setFileId("4564972819220421305");
mVodPlayer.startVodPlay(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 startVodPlay 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.startVodPlay(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 默认支持精准 seek，可以在播放器前通过 `TXVodPlayConfig#setEnableAccurateSeek` 进行配置。

```
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);
```

精准和非精准 Seek

播放器 SDK 11.8 版本开始，支持调用 seek 接口时，指定精准或非精准 seek。

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

Seek 到视频流指定 PDT 时间点

跳转到视频流指定 PDT（Program Date Time ）时间点，可实现视频快进、快退、进度条跳转等功能，目前只支持 HLS 视频格式。
注意：播放器高级版 11.6 版本开始支持。

```
long pdtTimeMs = 600; // 单位为 毫秒
mVodPlayer.seekToPdtTime(time);
```

从指定时间开始播放

首次调用 startVodPlay 之前，支持从指定时间开始播放。

```
float startTimeInSecond = 60; // 单位：秒
mVodPlayer.setStartTime(startTimeInSecond);    // 设置开始播放时间
mVodPlayer.startVodPlay(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.startVodPlay(url);    // startVodPlay 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。

```
TXVodPlayConfig mPlayConfig = new TXVodPlayConfig();
Map<String, String> headers = new HashMap<>();
headers.put("Referer", "${Refer Content}");
mPlayConfig.setHeaders(headers);
```

```
mVodPlayer.setConfig(mPlayConfig);
```

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 startVodPlay，否则会产生比较严重的花屏问题。

```
mVodPlayer.stopPlay(true);
mVodPlayer.enableHardwareDecode(true);
mVodPlayer.startVodPlay(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。可以通过下面方法进行清晰度设置。

```
// 获取多码率数组，TXBitrateItem 类字段含义：index-码率下标；width-视频宽；height-视频高；
bitrate-视频码率
// 在收到播放器 PLAY_EVT_VOD_PLAY_PREPARED 事件调用 getSupportedBitrates 才会有值返回
ArrayList<TXBitrateItem> bitrates = mVodPlayer.getSupportedBitrates();
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

// 获取当前播放的码率下标，返回值 -1000 为默认值，表示没有设置过码率标；返回值 -1 表示开启了自适应
码流
int index = mVodPlayer.getBitrateIndex();
```

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。
如果您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，从而避免播放后切换码流。详细方法参考 播放器配置#启播前指定分辨率 。

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。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. 开启平滑切换码率

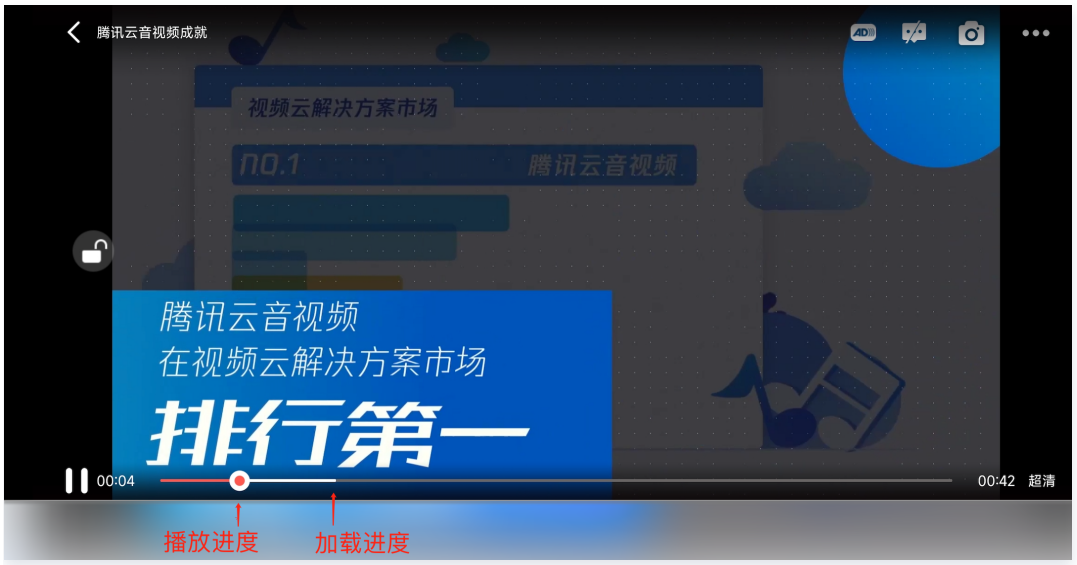
在启动播放前，通过开启平滑切换码率，在播放过程中切换码率，可以达到无缝平滑切换不同清晰度。对比关闭平滑切换码率，切换过程比较耗时、体验更好，可以根据需求进行设置。

```
TXVodPlayConfig mPlayConfig = new TXVodPlayConfig();
```

```
// 设为 true，在IDR对齐时可平滑切换码率，设为 false 时，可提高多码率地址打开速度
mPlayConfig.setSmoothSwitchBitrate(true);
mVodPlayer.setConfig(mPlayConfig);
```

13. 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

            // 获取 PDT 时间，播放器高级版 11.6 版本开始支持
            long pdt_time_ms = param.getLong(TXVodConstants.EVT_PLAY_PDT_TIME_MS);
        }
    }
```

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

14. 播放网速监听

通过 事件监听 方式，可以在视频播放卡顿时显示当前网速。

- 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);
    }
});
```

15. 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();
```

16. 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
```

17. 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。

18. DRM 加密视频播放

> **注意：**
> 此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持播放商业级 DRM 加密视频，目前支持 WideVine 和 Fairplay 两种 DRM 方案。更多的商业级 DRM 信息，请参见 商业级 DRM 综述 。
注意：DRM 播放请通过 `TXVodPlayer#setSurface` 配置后播放，否则会出现播放异常。
可通过下面两种方式播放 DRM 加密视频：

通过 FileId 播放

```
// DRM 播放请通过 TXVodPlayer#setSurface配置后播放，否则会出现播放异常
mVodPlayer.setSurface(mPlayerView.getHolder().getSurface());

// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(${appId}, // 腾讯云账户的
appId
    ${fieId}, // DRM 加密视频的 fileId
    ${psgin}); // 加密视频的播放器签名
mVodPlayer.startVodPlay(playInfoParam);
```

通过 FileId 播放适用于接入云点播后台。这种方式和播放普通 FileId 文件没有差别，需要在云点播先配置资源为 DRM 类型，SDK 会在内部识别并处理。

自定义配置播放

```
// DRM 播放请通过 TXVodPlayer#setSurface配置后播放，否则会出现播放异常
mVodPlayer.setSurface(mPlayerView.getHolder().getSurface());

// 步骤一： 设置 Drm 证书提供商环境，此步骤默认情况下不需要配置
// Google Drm 证书提供商环境默认使用 googleapis.com 域名，可通过下面接口设置为
googleapis.cn 域名
TXPlayerGlobalSetting.setDrmProvisionEnv(TXPlayerGlobalSetting.DrmProvisionEnv.D
RM_PROVISION_ENV_CN);  // googleapis.cn 域名证书地址

// 步骤二：通过 TXVodPlayer#startPlayDrm 接口播放
TXPlayerDrmBuilder builder = new TXPlayerDrmBuilder();
builder.setPlayUrl(${url});  // 设置播放视频的 url
builder.setKeyLicenseUrl(${keyLicenseUrl});   /// 设置解密 key url
mVodPlayer.startPlayDrm(builder);
```

19. 外挂字幕

> ⚠ 注意：

> 此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持添加和切换外挂字幕，现已支持 SRT 和 VTT 这两种格式的字幕。
实践教程：建议在 `startVodPlay` 之前添加字幕和配置字幕样式，在收到 `VOD_PLAY_EVT_VOD_PLAY_PREPARED` 事件后，调用 selectTrack 选择字幕。添加字幕后，并不会主动加载字幕，调用 `selectTrack` 后，才会加载字幕，字幕选择成功会有 `VOD_PLAY_EVT_SELECT_TRACK_COMPLETE` 事件回调。
详细用法如下：

步骤1：设置字幕渲染目标对象 View。

```
// 把TXSubtitleView添加到播放器所在布局xml
<com.tencent.rtmp.ui.TXSubtitleView
            android:id="@+id/subtitle_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

mSubtitleView = findViewById(R.id.subtitle_view);
// 设置字幕渲染目标对象
mVodPlayer.setSubtitleView(mSubtitleView);
```

步骤2：添加外挂字幕。

```
// 传入 字幕url， 字幕名称， 字幕类型， 建议在启动播放器前添加
mVodPlayer.addSubtitleSource("https://mediacloud-76607.gzc.vod.tencent-
cloud.com/DemoResource/subtitleVTT.vtt", "subtitleName",
TXVodConstants.VOD_PLAY_MIMETYPE_TEXT_VTT);
```

步骤3：播放后切换字幕。

```
// 开始播放视频后，选中添加的外挂字幕， 在收到 VOD_PLAY_EVT_VOD_PLAY_PREPARED 事件后调用
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    if (event == TXVodConstants.VOD_PLAY_EVT_VOD_PLAY_PREPARED) {
        List<TXTrackInfo> subtitleTrackInfoList = mVodPlayer.getSubtitleTrackInfo();
        for (TXTrackInfo track : subtitleTrackInfoList) {
            Log.d(TAG, "TrackIndex= " + track.getTrackIndex() + " ,name= " +
track.getName());
            if (TextUtils.equals(track.getName(), "subtitleName")) {
                // 选中字幕
                mVodPlayer.selectTrack(track.trackIndex);
            } else {
                // 其它字幕不需要的话， 进行deselectTrack
                mVodPlayer.deselectTrack(track.trackIndex);
            }
        }
    }
}
```

```
// 如果需要，可以监听轨道切换消息
mVodPlayer.setVodListener(new ITXVodPlayListener() {
    @Override
    public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
        if (event == TXVodConstants.VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
            int trackIndex =
param.getInt(TXVodConstants.EVT_KEY_SELECT_TRACK_INDEX);
            int errorCode =
param.getInt(TXVodConstants.EVT_KEY_SELECT_TRACK_ERROR_CODE);
            Log.d(TAG, "receive VOD_PLAY_EVT_SELECT_TRACK_COMPLETE, trackIndex=" +
trackIndex + " ,errorCode=" + errorCode);
        }
    }

    @Override
    public void onNetStatus(TXVodPlayer player, Bundle status) {

    }
});
```

步骤4：配置字幕样式。

字幕样式支持在播放前或者播放过程中配置。

```
// 详细参数配置请参考 API 文档
TXSubtitleRenderModel model = new TXSubtitleRenderModel();
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.fontColor = 0xFFFFFFFF; // 设置字幕字体颜色，默认白色不透明
model.isBondFontStyle = false;  // 设置字幕字体是否为粗体
mVodPlayer.setSubtitleStyle(model);
```

20、字幕文本回调

> **注意：**
> 此功能播放器高级版 12.3 版本开始支持。

播放器高级版 SDK 在默认配置下是通过内置引擎渲染字幕和展示， 可通过修改配置支持回调文本，业务可以在获取到字幕文本后自行渲染和展示。现已支持 SRT 和 VTT 这两种格式的字幕。
详细用法如下：

步骤1：设置字幕文本回调

```
TXVodPlayConfig config = new TXVodPlayConfig();
Map<String, Object> extInfoMap = new HashMap<>();
```

```
extInfoMap.put("450", new Integer(0));
config.setExtInfo(extInfoMap);
mVodPlayer.setConfig(config);
```

步骤2：添加和选择字幕

添加和选择字幕文档，请参见 外挂字幕 章节。

步骤3：注册监听字幕文本回调

选中字幕后可注册下面接口监听字幕文本内容。相关字段含义说明：

- `TXVodSubtitleData#trackIndex`，当前字幕的轨道 index；
- `TXVodSubtitleData#subtitleData`，实际字幕文本内容，当回调的 `subtitleData` 为空表示字幕为空，业务上封装展示即可；
- `TXVodSubtitleData` 类的其它字段暂时没有实际意义，不用关注。

```
mVodPlayer.setVodSubtitleDataListener(new ITXVodSubtitleDataListener() {
    @Override
    public void onSubtitleData(TXVodDef.TXVodSubtitleData subtitleData) {
        long trackIndex = subtitleData.trackIndex; // 当前字幕的轨道 index
        String data = subtitleData.subtitleData;   // 实际字幕文本内容
        // 根据需要展示 data 字幕文本内容
    }
});
```

21. 多音轨切换

> **注意：**
> 此功能需要播放器高级版本才支持。

播放器高级版 SDK 支持切换视频内置的多音轨。用法参见如下代码：

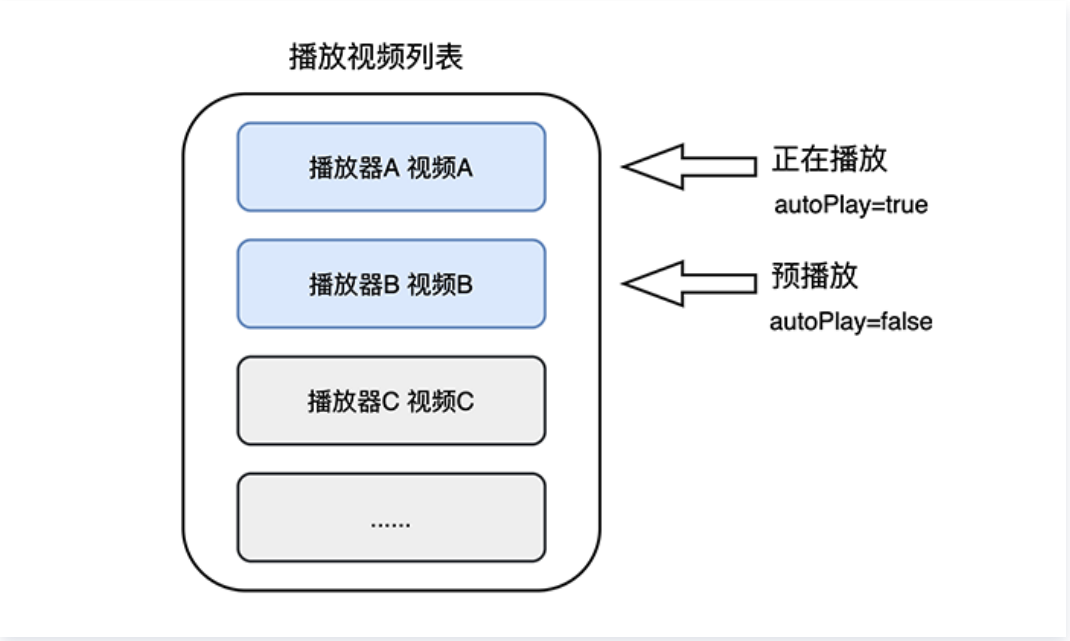
```
// 返回音频轨道信息列表
List<TXTrackInfo> soundTrackInfoList = mVodPlayer.getAudioTrackInfo();
for (TXTrackInfo trackInfo : soundTrackInfoList) {
    if (trackInfo.trackIndex ==  0) {
        // 通过判断 trackIndex 或者 name 切换到需要的音轨
        mVodPlayer.selectTrack(trackInfo.trackIndex);
    } else {
        // 不需要的音轨进行 deselectTrack
        mVodPlayer.deselectTrack(trackInfo.trackIndex);
    }
}
```

进阶功能使用

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，会占用下载带宽和线程资源。建议视频预播放并发个数控制在3个以内。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 startVodPlay 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.startVodPlay(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.startVodPlay(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地应对网络的波动，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：**
> - **视频预下载会占用下载带宽和线程资源，建议进行队列控制，并发个数控制在3个以内。**
> - 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。

通过媒资 URL 预下载

通过媒资 URL 预下载视频代码示例如下：

```
//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(getApplicationContext());
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)
```

通过媒资 FileId 预下载

> **注意：**
> 通过 fileId 预下载从 11.3 版本开始支持。

通过 fileId 预下载是耗时操作， 请不要在主线程调用，否则会抛出非法调用异常。startPreload 时传入的 preferredResolution 要和启播时设置的优先启播分辨率保持一致，否则将达不到预期的效果。`onStart` 回调的 URL 可以保存起来，传给播放器播放。使用示例如下：

```
//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
}

// 启动预下载
Runnable task = new Runnable() {
    @Override
```

```
    public void run() {
        TXPlayInfoParams playInfoParams = new TXPlayInfoParams(${appId},
"${fileId}",
                "${psign}");
        // 注意：  耗时操作，请不要在主线程调用！在主线程调用将会抛出非法调用异常。
        mPreLoadManager.startPreload(playInfoParams, 3, 1920 * 1080, new
ITXVodFilePreloadListener() {

            @Override
            public void onStart(int taskID, String fileId, String url, Bundle
bundle) {
                // 通过 fileId 换链成功后会回调 onStart， url 可以保存起来，传给播放器播
放
                Log.d(TAG, "preload: onStart: taskID: " + taskID + ", fileId: "
+ fileId + ", url: " + url + ",bundle: " + bundle);
            }

            @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);
            }
        });
    }
};
new Thread(task).start();


//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。如果是加密视频，通过播放器 SDK 下载后的视频在本地保持为加密状态，仅可通过腾讯云播放器 SDK 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：**
> 视频下载支持下载 MP4 和 HLS 视频，对应嵌套 HLS 视频，需要指定偏好清晰度（preferredResolution）。

步骤1：准备工作

SDK 初始化时，设置全局存储路径，用于视频下载，预加载，和缓存等功能。用法如下：

```
File sdcardDir = context.getExternalFilesDir(null);
TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/txcache");
```

`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();
```

设置下载使用的 httpHeader

根据业务需求进行配置，播放器启动下载时将发送给服务端， 播放器 12.2 版本开始支持。

```
Map<String, String> httpHeaders = new HashMap<>();
downloader.setHeaders(httpHeaders);  // 设置下载httpHeader
```

步骤2：开始下载

开始下载有 Fileid 和 URL 两种方式，具体操作如下：

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”。

注意：加密视频只能通过Fileid下载，psign参数必须填写。

```
// QUALITY_240P  240p
// QUALITY_360P  360P
// QUALITY_480P  480p
// QUALITY_540P  540p
// QUALITY_720P  720p
// QUALITY_1080P 1080p
// QUALITY_2K    2k
// QUALITY_4K    4k
// quality参数可以自定义，取分辨率宽高最小值(如分辨率为1280*720, 期望下载此分辨率的流，
quality传入 QUALITY_720P)
// 播放器 SDK 会选择小于或等于传入分辨率的流进行下载
TXVodDownloadDataSource source = new TXVodDownloadDataSource(1252463788,
"4564972819220421305", QUALITY_720P, "pSignxxxx", "");    //pSign 加密视频签名， 通
过fileId下载必填
downloader.startDownload(source);
```

URL 方式

至少需要传入下载地址 URL。preferredResolution 取值为视频分辨率宽和高的乘积：preferredResolution=width * height。 如果是嵌套 HLS 格式，preferredResolution 不传入具体值时，默认值为921600。userName 不传入具体值时，默认为"default"。

```
downloader.startDownloadUrl("", 921600 , "");
```

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
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。

下载错误码

错误码	数值	含义说明
DOWNLOAD_SUCCESS	0	下载成功。
DOWNLOAD_AUTH_FAILED	-5001	向云点播控制台请求视频信息失败，建议检查 fileId、psign参数是否正确。
DOWNLOAD_NO_FILE	-5003	无此清晰度文件。
DOWNLOAD_FORMAT_ERRO	-5004	下载文件格式不支持。

R		
DOWNLOAD_DISCONNECT	-5005	网络断开，建议检查网络是否正常。
DOWNLOAD_HLS_KEY_ERROR	-5006	获取 HLS 解密 Key 失败。
DOWNLOAD_PATH_ERROR	-5007	下载目录访问失败，建议检查是否有访问下载目录的权限。
DOWNLOAD_403FORBIDDEN	-5008	请求下载时，鉴权信息不通过，建议检查签名（psign）是否已过期。

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

1. 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
// getDownloadMediaInfoList 是耗时接口，请不要在主线程调用
List<TXVodDownloadMediaInfo> downloadInfoList =
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);
  }
}
```

2. 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
// getDownloadMediaInfo 是耗时接口，请不要在主线程调用
TXVodDownloadMediaInfo downloadInfo = downloader.getDownloadMediaInfo(1252463788,
"4564972819220421305", QUALITYHD);
// 获取下载文件总大小，单位：Byte，只针对 fileid 下载源有效。
// 备注：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法获取。
long size = downloadInfo.getSize();  // 获取下载文件总大小
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
```

3. 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， 耗时接口，请不要在主线程调用
TXVodDownloadMediaInfo downloadInfo =
downloader.getDownloadMediaInfo("http://1253131631.vod2.myqcloud.com/26f327f9vodgzp1253131631/f4bdff799031868222924043041/playlist.m3u8”);
```

4. 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);
```

步骤6：下载后离线播放

下载后的视频支持无网络的情况下进行播放，无需进行联网。下载完成后，通过 `TXVodDownloadMediaInfo#getPlayPath` 获取到下载地址即可进行播放。

```
// getDownloadMediaInfoList 是耗时接口，请不要在主线程调用
List<TXVodDownloadMediaInfo> mediaInfoList =
TXVodDownloadManager.getInstance().getDownloadMediaInfoList();
// 业务侧根据实际需求查找到需要播放的 media 对象
for (TXVodDownloadMediaInfo mediaInfo : mediaInfoList) {
    if (mediaInfo.getDownloadState() == TXVodDownloadMediaInfo.STATE_FINISH) {  // 判断是否下载完成
        mVodPlayer.startVodPlay(mediaInfo.getPlayPath());  // 播放已经下载完成的视频
    }
}
```

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。
在腾讯云控制台提取到 appId， 加密视频的 fileId 和 psign 后，可以通过下面的方式进行播放：

```
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
TXPlayInfoParams playInfoParam = new TXPlayInfoParams(1252463788, // 腾讯云账户的appId
    "4564972819220421305", // 视频的 fileId
```

```
    "psignxxxxxxx"); // 播放器签名
mVodPlayer.startVodPlay(playInfoParam);
```

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
```

- **启播前指定分辨率**

  播放 HLS 的多码率视频源，如果您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(TXVodConstants.VIDEO_RESOLUTION_720X1280);
  mVodPlayer.setConfig(config);  // 把 config 传给 mVodPlayer
  ```

- **启播前指定媒资类型**

  当提前知道播放的媒资类型时，可以通过配置 `TXVodPlayConfig#setMediaType` 减少播放器 SDK 内部播放类型探测，提升启播速度。

  > **注意：**
  > `TXVodPlayConfig#setMediaType` 11.2 版本开始支持。

  ```
  TXVodPlayConfig config = new TXVodPlayConfig();
  config.setMediaType(TXVodConstants.MEDIA_TYPE_FILE_VOD);  // 用于提升MP4启播速度
  // config.setMediaType(TXVodConstants.MEDIA_TYPE_HLS_VOD);  // 用于提升HLS启播速度
  mVodPlayer.setConfig(config);
  ```

- **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
  ```

- **启播前指定优先启播音轨**

  > **注意：**

播放器高级版 12.3 版本开始支持。

当提前知道播放的音轨名称时，可以通过配置 `TXVodPlayConfig#setPreferredAudioTrack` 在启播前指定优先启播音轨。

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setPreferredAudioTrack("audioTrackName");  // audioTrackName为实际音轨名称
mVodPlayer.setConfig(config);  // 把config 传给 mVodPlayer
```

6. HttpDNS 解析服务

移动解析（HTTPDNS）基于 HTTP 协议向 DNS 服务器发送域名解析请求，替代了基于 DNS 协议向运营商 Local DNS 发起解析请求的传统方式，可避免 Local DNS 造成域名劫持和跨网访问问题，解决移动互联网服务中域名解析异常带来的视频播放失败困扰。

注意：
HttpDNS 解析服务从 10.9 版本开始支持。

1. 开通 HTTPDNS 解析服务。
   您可以选择腾讯云或其它云提供商，开通 HTTPDNS 解析服务，确保开通成功后， 再集成到播放 SDK。
2. 在播放 SDK 接入 HTTPDNS 解析服务。
   下面以接入 腾讯云 HTTPDNS 为例子，展示如何在播放器 SDK 接入：

```
// 步骤1： 打开 HttpDNS 解析开关
TXLiveBase.enableCustomHttpDNS(true);
// 步骤2： 设置 HttpDNS 解析回调， TXLiveBaseListener#onCustomHttpDNS
TXLiveBase.setListener(new TXLiveBaseListener() {
    @Override
    public void onCustomHttpDNS(String hostName, List<String> ipList) {
        // 播放器 SDK 会把 hostName 回调给业务， 业务可以根据实际需要决定是否把 hostName 解析
ip，
        // 如果返回空 ip，则 SDK 内部不会对此次网络请求使用 httpdns
        // 把 hostName 解析到 ip 地址后，保存到 iPList，返回给 SDK 内部。注意：这里不要进行异
步操作。
        // 使用示例：MSDKDnsResolver 是腾讯云提供的 HTTPDNS SDK 解析接口
        // 注意：
        // String ips = MSDKDnsResolver.getInstance().getAddrByName(hostname);
        String[] ipArr = ips.split(";");
        if (0 != ipArr.length) {
            for (String ip : ipArr) {
                if ("0".equals(ip)) {
                    continue;
                }
                ipList.add(ip);
            }
        }
    }
```

```
});
```

7. HEVC 自适应降级播放

播放器支持同时传入 HEVC 和其它视频编码格式例如： H.264 的播放链接， 当播放机型不支持 HEVC 格式时，将自动降级为配置的其它编码格式（如： H.264 ）的视频播放。

> **注意：**
> 播放器高级版 11.7 版本开始支持。

```
//设置备选播放链接
String backupPlayUrl = "${backupPlayUrl}";  // 备选播放链接
mVodPlayer.setStringOption(TXVodConstants.VOD_KEY_BACKUP_URL, backupPlayUrl); // 设置
H.264 格式等备选播放链接地址
// 非必选，播放器 SDK 12.0 版本开始支持。设置备选播放资源的 MediaType，默认为
TXVCubePlayerParams.MEDIA_TYPE_AUTO
//mVodPlayer.setStringOption(TXVodConstants.VOD_KEY_BACKUP_URL_MEDIA_TYPE,
TXVodConstants.MEDIA_TYPE_AUTO);

// 设置原始 HEVC 播放链接
String hevcPlayUrl = "${hevcPlayUrl}";
mVodPlayer.setStringOption(TXVodConstants.VOD_KEY_MIMETYPE,
TXVodConstants.VOD_PLAY_MIMETYPE_H265); // 指定原始 HEVC 视频编码类型
mVodPlayer.startVodPlay(hevcPlayUrl);
```

8. 音量均衡

播放器支持在播放音频时自动调整音量，使得所有音频的音量保持一致。这可以避免某些音频过于响亮或过于安静的问题，提供更好的听觉体验。通过 `TXVodPlayer#setAudioNormalization` 设置音量均衡，响度范围：-70～0 (LUFS)，同时支持自定义数值。
注意：播放器高级版 11.7 版本开始支持。

```
/**
 可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）
 关：AUDIO_NORMALIZATION_OFF
 开：AUDIO_NORMALIZATION_STANDARD（标准）
    AUDIO_NORMALIZATION_LOW （低）
    AUDIO_NORMALIZATION_HIGH（高）
 可填自定义数值：从低到高，范围-70 - 0 LUFS
*/
mVodPlayer.setAudioNormalization(TXVodConstants.AUDIO_NORMALIZATION_STANDARD);  //启
动音量均衡

mVodPlayer.setAudioNormalization(TXVodConstants.AUDIO_NORMALIZATION_OFF);       //关
闭音量均衡
```

9、MP4 视频本地加密

开启 MP4 本地加密后，播放器在缓存文件时，将对数据进行加密存储，加密视频文件将只能由播放器来解密播放，无法用第三方播放器播放。

> **注意：**
> 开启 MP4本地加密并开始播放后，就无法再更改加密设置了，除非文件被清理并重新缓存。

此功能播放器高级版 12.2 版本开始支持。

```
//前置条件：设置播放器的全局缓存目录，此配置在项目中设置一次即可
File sdcardDir = getApplicationContext().getExternalFilesDir(null);
if (sdcardDir != null) {
    TXPlayerGlobalSetting.setCacheFolderPath(sdcardDir.getPath() + "/PlayerCache");
}

TXVodPlayConfig config = new TXVodPlayConfig();
config.setEncryptedMp4Level(TXVodConstants.MP4_ENCRYPTION_LEVEL_L2); // 设置使用mp4本地加密播放和存储
mVodPlayer.setConfig(config);
```

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_END	2006	视频播放结束。
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。
TXVodConstants.VOD_PLAY_EVT_SEEK_COMPLETE	2019	Seek 完成，10.3版本开始支持。
VOD_PLAY_EVT_LOOP_ONCE_COMPLETE	6001	循环播放，一轮播放结束（10.8 版本开始支持）。
TXVodConstants.VOD_PLAY_EVT_HIT_CACHE	2002	启播时命中缓存事件（11.2 版本开始支持）。

TXVodConstants.VOD_PLAY_EVT_VIDEO_SEI	2030	收到 SEI 帧事件（ 播放器高级版11.6 版本开始支持）。
VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK	2031	发生 HEVC 降级播放（播放器高级版12.0版本开始支持）。
VOD_PLAY_EVT_FIRST_VIDEO_PACKET	2017	播放器收到首帧数据包事件（12.0 版本开始支持）。

SEI 帧

SEI（Supplemental Enhancement Information）帧是一种用于传递附加信息的帧类型，播放器高级版会解析视频流中的 SEI 帧，通过 VOD_PLAY_EVT_VIDEO_SEI 事件回调， 注意：播放器高级版 11.6 版本开始支持。

```
@Override
public void onPlayEvent(TXVodPlayer player, int event, Bundle param) {
    if (event == TXVodConstants.VOD_PLAY_EVT_VIDEO_SEI) {
        int seiType = param.getInt(TXVodConstants.EVT_KEY_SEI_TYPE);  //  the type
of video SEI
        int seiSize = param.getInt(TXVodConstants.EVT_KEY_SEI_SIZE);  // the data
size of video SEI
        byte[] seiData = param.getByteArray(TXVodConstants.EVT_KEY_SEI_DATA);  //
the byte array data of video SEI
    }
}
```

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：startVodPlay(TXPlayInfoParams playInfoParams)），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_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。
TXVodConstants.EVT_DRM_TYPE	加密类型，10.2版本开始支持。
TXVodConstants.EVT_KEY_WATER_MARK_TEXT	幽灵水印文本内容（11.6 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) {
    }
});
```

幽灵水印

幽灵水印内容在播放器签名中填写，经云点播后台，最终展示到播放端上，整个传输链路过程由云端和播放端共同协作，确保水印的安全。在播放器签名中 配置幽灵水印教程 。幽灵水印的内容在收到播放器的 `TXVodConstants#VOD_PLAY_EVT_GET_PLAYINFO_SUCC` 事件后， 通过 `param.getString(TXVodConstants.EVT_KEY_WATER_MARK_TEXT)` 获取。详细使用教程参见 超级播放器组件 > 幽灵水印 。

注意： 播放器 11.6 版本开始支持。

播放错误事件

> **说明：**
> [-6004 , -6010] 错误事件 11.0 版本开始支持。

事件 ID	数值	含义说明
PLAY_ERR_NET_DISCONNECT	-2301	视频数据错误导致重试亦不能恢复正常播放。例如：网络异常或下载数据错误，导致解封装超时或失败。
PLAY_ERR_HLS_KEY	-2305	HLS 解密 key 获取失败。
VOD_PLAY_ERR_SYSTEM_PLAY_FAIL	-6004	系统播放器播放错误。
VOD_PLAY_ERR_DECODE_VIDEO_FAIL	-6006	视频解码错误，视频格式不支持。
VOD_PLAY_ERR_DECODE_AUDIO_FAIL	-6007	音频解码错误，音频格式不支持。
VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL	-6008	字幕解码错误。
VOD_PLAY_ERR_RENDER_FAIL	-6009	视频渲染错误。
VOD_PLAY_ERR_PROCESS_VIDEO_FAIL	-6010	视频后处理错误。

VOD_PLAY_ERR_GET_PLAYINFO_FAIL	-2306	获取点播文件信息失败，建议检查 AppId、FileId 或 Psign 填写是否正确。

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，单位 KBps。
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 bps。
NET_STATUS_AUDIO_BITRATE	当前流媒体的音频码率，单位 bps。
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，单位：KBps。
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 bps
        int videoBitRate = bundle.getInt(TXLiveConstants.NET_STATUS_VIDEO_BITRATE);
        //获取当前流媒体的音频码率，单位 bps
        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);
    }
});
```

其它功能使用

HLS 直播视频源播放

播放器高级版本支持播放 HLS 直播视频源，从 11.8 版本开始支持带 HLS EVENT 直播视频源。用法如下：

```
TXVodPlayConfig config = new TXVodPlayConfig();
config.setMediaType(TXVodConstants.MEDIA_TYPE_HLS_LIVE); // 指定HLS直播媒资类型
mVodPlayer.setConfig(config);
mVodPlayer.startVodPlay(${YOUR_HSL_LIVE_URL});
```

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，我们会提供更多的基于用户场景的组件。您可以通过 Player_Android 下载体验。

Flutter 端集成
集成指引

最近更新时间：2025-04-01 14:11:52

环境准备

- Flutter 3.0 及以上版本。
- Android 端开发：
  - Android Studio 3.5及以上版本。
  - App 要求 Android 4.1及以上版本设备。
- iOS 端开发：
  - Xcode 11.0及以上版本。
  - OSX 系统版本要求 10.11 及以上版本。
  - 请确保您的项目已设置有效的开发者签名。

SDK 下载

腾讯云视立方 Flutter 播放器项目的地址是 Player Flutter 。

> **注意：**
> 运行该 demo 的时候，需要在 demo_config 中设置自己的播放器 license，并在 Android 和 iOS 配置中，将包名和 bundleId 修改为自己签名的包名和 bundleId。

快速集成

在项目的 pubspec.yaml 中添加依赖

支持基于 LiteAVSDK Player 或 Professional 版本集成，您可以根据项目需要进行集成。

1. 集成 LiteAVSDK_Player 版本最新版本，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
```

集成 LiteAVSDK_Player_Premium (播放器高级版) 最新版本，则 `pubspec.yaml` 中配置改为：

```
super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: Player_Premium
```

集成 LiteAVSDK_Professional 最新版本，则 `pubspec.yaml` 中配置改为：

```
super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: Professional
```

如果需要集成指定播放器版本的 SDK，可以指定通过 ref 依赖的 tag 来指定到对应版本，如下所示：

```
super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: release_pro_v12.0.0

# release_pro_v12.0.0 表示将集成Android端TXLiteAVSDK_Professional_12.0.0.14681 版
本，
# 和iOS 端集成 TXLiteAVSDK_Professional_12.0.16292 版本
```

更多归档的 tag 请参考 release 列表 。

pub 集成方式

```
# pub集成默认为 professional 版本，如果有其余版本需求，请使用分支集成方式
super_player: ^12.3.0
```

2. 集成之后，可以通过代码编辑器自带的 UI 界面来获取 Flutter 依赖，也可以直接使用如下命令获取：

```
flutter pub get
```

3. 使用过程中，可以通过以下命令来更新现有 Flutter 依赖：

```
flutter pub upgrade
```

添加原生配置

Android 端配置

1. 在 Android 的 `AndroidManifest.xml` 中增加如下配置：

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

网络安全配置允许 App 发送 http 请求

出于安全考虑，从 Android P 开始，Google 要求 App 的请求都使用加密链接。播放器 SDK 会启动一个 localsever 代理 http 请求，如果您的应用 **targetSdkVersion 大于或等于28**，可以通过 网络安全配置 来开启允许向127.0.0.1发送 http 请求。否则播放时将出现 "java.io.IOException: Cleartext HTTP traffic to 127.0.0.1 not permitted" 错误，导致无法播放视频。配置步骤如下：

1.1 在项目新建 res/xml/network_security_config.xml 文件，设置网络安全性配置。

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
</network-security-config>
```

1.2 在 AndroidManifest.xml 文件下的 application 标签增加以下属性。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
                    ... >
        ...
    </application>
</manifest>
```

2. 确保 Android 目录下的 `build.gradle` 使用了 mavenCenter，能够成功下载到依赖。

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

3. 配置 Android 最小 SDK 版本，由于 flutter 默认配置的 Android 最小版本过低，需要手动更改为至少19，如果需要使用画中画能力，compileSdkVersion 和targetSdkVersion 则需要修改为至少31。

```
compileSdkVersion 31
defaultConfig {
    applicationId "com.tencent.liteav.demo"
    minSdkVersion 19
    targetSdkVersion 31
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
}
```

4. `AndroidManifest.xml` 根节点 manifest 标签内增加如下配置 `xmlns:tools="http://schemas.android.com/tools"`，示例如下：

```
<manifest
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.player">
 <!-- your config...... -->
</manifest>
```

application 节点下增加 `tools:replace="android:label"`，示例如下：

```
<application
    android:label="super_player_example"
    android:icon="@mipmap/ic_launcher"
    android:requestLegacyExternalStorage="true"
    tools:replace="android:label">
<!-- your config...... -->
</application>
```

5. 如果需要更新原生 SDK 依赖版本，可手动删除 Android 目录下的 `build` 文件夹，也可以使用如下命令强制刷新。

```
./gradlew build
```

iOS 端配置

> **注意：**
> iOS 端目前暂不支持模拟器运行调试，建议在真机下进行开发调试。

1. 在 iOS 的 `Info.plist` 中增加如下配置：

```
<key>NSAppTransportSecurity</key>
<dict>
 <key>NSAllowsArbitraryLoads</key>
 <true/>
</dict>
```

2. iOS 原生业务侧可自行编辑 `podfile` 文件，来指定您的播放器 SDK 版本，默认集成的是 Player 版 SDK。

```
# pod 'TXLiteAVSDK_Player_Premium'      // premium版
pod 'TXLiteAVSDK_Player'             //Player版
```

Professional 版 SDK 集成：

```
pod 'TXLiteAVSDK_Professional'  //Professional版
```

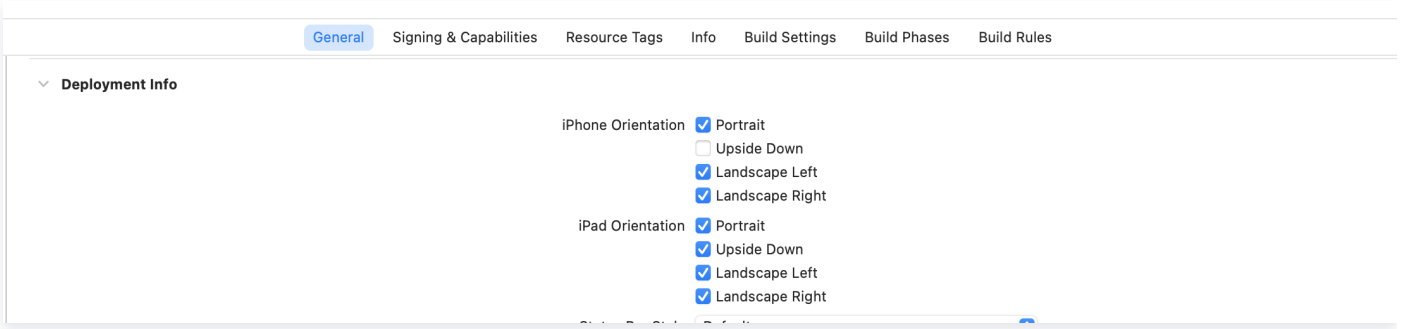
如果不指定版本，默认会安装最新的 `TXLiteAVSDK_Player` 最新版本。

3. 部分情况下（如：发布了新版本），需要强制更新 iOS 播放器依赖，可以在 iOS 目录下使用如下命令进行更新：

```
rm -rf Pods
rm -rf Podfile.lock
pod update
```

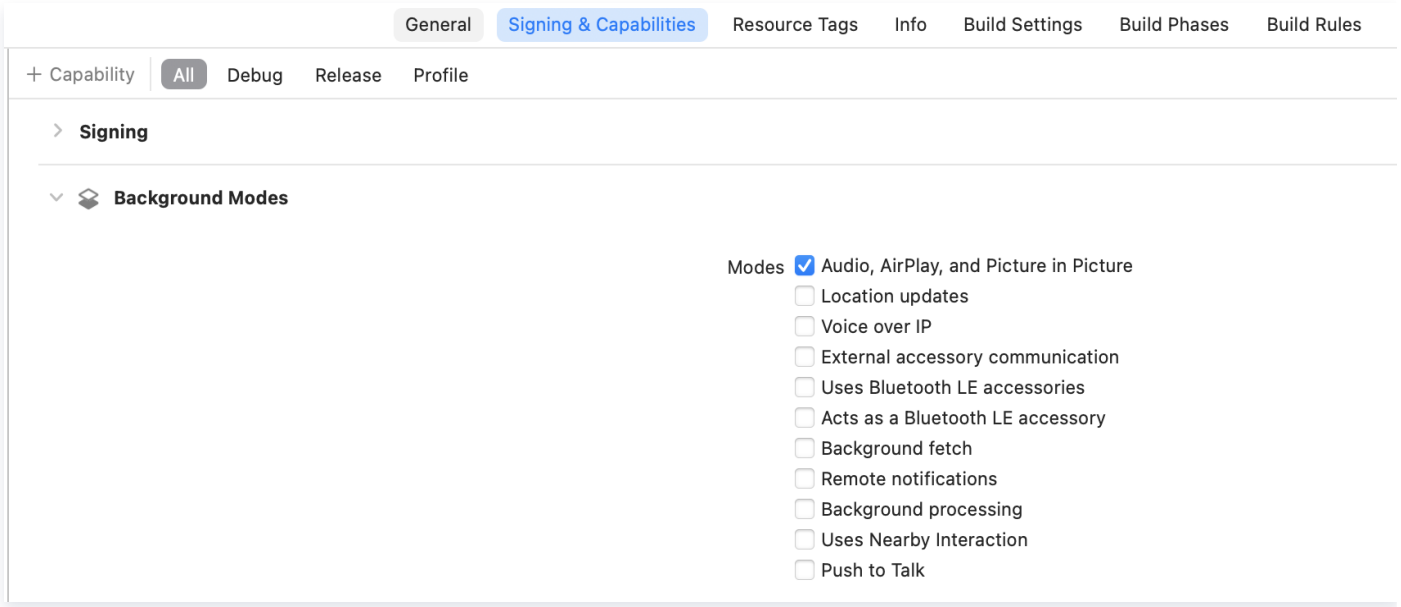
4. 横屏配置

如果需要应用支持横屏，需要在IOS项目配置 `General` 页面的 `Deployment Info` 标签下设置横竖屏的支持方向，可以全部进行勾选，如下图所示：



5. 画中画配置

如果需要项目支持画中画，需要在IOS项目配置 `Signing & Capabilities` 页面的 `Background Modes` 标签下，勾选 "Audio, AirPlay, and Picture in Picture" 来让项目支持画中画能力，如下图所示：



集成播放器 License

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 播放器License 获取相关授权。
集成播放器前，需要 注册腾讯云账户 ，注册成功后申请播放器 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);
```

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： `Flutter/superplayer_widget` ，导入到您的项目中，进行定制化开发。

常见问题

1. iOS 端运行，出现

   `No visible @interface for 'TXLivePlayer' declares the selector 'startLivePlay:type:'` 等类似找不到接口错误。
   可以使用如下命令，更新 iOS SDK：

   ```
   rm -rf Pods
   rm -rf Podfile.lock
   pod update
   ```

2. 同时集成 tencent_trtc_cloud 和 Flutter 播放器出现 SDK 或 符号冲突。

   常见异常日志：

   ```
   java.lang.RuntimeException: Duplicate class com.tencent.liteav.TXLiteAVCode found in modules classes.jar
   ```

此时需要集成flutter 播放器的 Professional 版本，让tencent_trtc_cloud 和 flutter 播放器共同依赖于同一个版的 LiteAVSDK_Professional。注意确保依赖的LiteAVSDK_Professional 的版本必须一样。
如：依赖 Android 端 TXLiteAVSDK_Professional_10.3.0.11196 和 iOS 端 TXLiteAVSDK_Professional to 10.3.12231 版本，依赖声明如下：

```
tencent_trtc_cloud: 2.3.8

super_player:
  git:
    url: https://github.com/LiteAVSDK/Player_Flutter
    path: Flutter
    ref: release_pro_v1.0.3.11196_12231
```

3. 需要同时使用多个播放器实例的时候，频繁切换播放视频，画面呈现模糊。
   在每个播放器组件容器销毁的时候，调用播放器的 `dispose` 方法，将播放器释放。
4. 其余通用 Flutter 依赖问题：
   - 执行 `flutter doctor` 命令检查运行环境，直到出现“No issues found!”。
   - 执行 `flutter pub get` 确保所有依赖的组件都已更新成功。
5. 集成 superPlayer 之后，出现如下 manifest 错误：

```
Attribute application@label value=(super_player_example) from
AndroidManifest.xml:9:9-45
is also present at [com.tencent.liteav:LiteAVSDK_Player:10.8.0.13065]
AndroidManifest.xml:22:9-41 value=(@string/app_name).
Suggestion: add 'tools:replace="android:label"' to <application> element at
AndroidManifest.xml:8:4-51:19 to override.
```

解决方法：由于播放器 Android SDK 的 AndroidManifest 已经定义过 label，而 flutter 新建项目之后，在 Android 目录的 AndroidManifest 也会定义 label，此处建议根据错误提示，进入您的 Android 项目目录，在 AndroidManifest 的根节点 `manifest` 节点下增加 `xmlns:tools="http://schemas.android.com/tools"`，并在 application 节点下增加 `tools:replace="android:label"`。

6. 集成 superPlayer 之后，出现如下版本错误：

```
uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library
[:super_player]
```

解决方法：目前播放器 Android SDK 最小支持版本为 android 19，flutter 部分版本默认 Android 最小支持版本为 android 16。建议您将最小支持版本提高到 android 19。具体修改方法为：进入您的 Android 项目的主 module 下，一般为 `app` 目录，将该目录下的 `build.gradle` 中的 `minSdkVersion` 修改为19。

7. 如何提取播放器 SDK 的运行 Log？

解决方法：播放器 SDK 默认把运行的 log 输出到本地文件，腾讯云技术支持 在帮忙定位问题时，需要这些运行 log 分析问题。Andorid 平台 log 保存在目录：`/sdcard/Android/data/packagename/files/log/tencent/liteav`，iOS 平台 log 保存在目录：`sandbox的Documents/log`。更详细的 log 提取可参考 此教程 。

8. 如何减少控制台 log 输出?

解决方法：可以通过下面的接口设置 log 输出级别：
SuperPlayerPlugin.setLogLevel(TXLogLevel.LOG_LEVEL_NULL)，支持以下 log 级别：

```
class TXLogLevel {
  static const LOG_LEVEL_VERBOSE = 0; // 输出所有级别的log
  static const LOG_LEVEL_DEBUG = 1; // 输出 DEBUG,INFO,WARNING,ERROR 和 FATAL 级别
的log
  static const LOG_LEVEL_INFO = 2; // 输出 INFO,WARNNING,ERROR 和 FATAL 级别的log
  static const LOG_LEVEL_WARN = 3; // 输出WARNNING,ERROR 和 FATAL 级别的log
  static const LOG_LEVEL_ERROR = 4; // 输出ERROR 和 FATAL 级别的log
  static const LOG_LEVEL_FATAL = 5; // 只输出FATAL 级别的log
  static const LOG_LEVEL_NULL = 6; // 不输出任何sdk log
}
```

9. 项目使用过程中，出现原生相关报错，例如
   错误：不兼容的类型`、`error: initializing 'BOOL' (aka 'bool') with an expression of incompatible type 'void'
   等错误，是由于SDK 更新，导致 SDK 与 flutter 端原生代码不兼容。此时只需要更新 SDK 版本即可。
   解决方法：在项目目录下，打开终端，依次输入如下命令：

```
flutter pub cache clean
flutter clean
flutter pub upgrade
flutter pub get
```

   确保命令执行成功，更新本地 flutter 依赖。
   然后在 ios 目录下，打开终端，输入如下命令，更新 iOS 依赖：

```
rm -rf Pods
rm -rf Podfile.lock
pod update
```

   如果问题依然存在，可以尝试删除项目 build 文件夹，并且手动删除您电脑中的 flutter 依赖缓存文件夹 `.pubcache` 。然后重新刷新 flutter pub 依赖再进行编译运行。
10. 安卓点播播放器播放视频，播放器边缘出现平铺拉伸现象。
    该问题是flutter端sdk的纹理渲染问题，可以将flutter版本升级到 flutter 3.7.0以上。
11. flutter调试和测试包运行没问题，但是打正式包会闪退。
    flutter打正式包默认是开启混淆的，播放器SDK需要配置如下混淆规则：

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

12. 播放本地视频无法播放
    flutter播放器支持本地视频播放，需要将正确的本地视频地址传入到视频播放接口中，出现无法播放现象，首先需要检查本地视频地址是否可用，文件是否损坏，如果本地视频没有问题，需要检查应用是否具有存储或者图片/视频读取权限。
13. 运行 iOS 项目出现 `CocoaPods could not find compatible versions for pod "Flutter"` 等类似报错

该问题是由于在高 flutter 开发环境中，已经不再支持 iOS 低版本，可以检查项目中 Minimum Deployments 配置的 iOS 版本是否过小，或者是否继承了只支持低 iOS 版本的依赖。

14. 播放器在切换绑定的纹理的时候，画面经常会消失。

由于 flutter 界面经常会全局刷新 UI，导致 TXPlayerVideo 与 controller 被动发生重复绑定，发生纹理绑定抢夺问题 。建议使用 TXPlayerVideo 的 onRenderViewCreatedListener 回调，获取到 viewId 后，使用 controller 的 setPlayView 进行绑定。

更多功能

您可以通过运行项目中的 example 体验完整功能，example 运行指引。
播放器 SDK 官网提供了 iOS、Android 和 Web 端的 Demo 体验，请单击这里 。

点播场景

最近更新时间：2025-06-04 15:55:52

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 。

通过本文您可以学会

- 如何集成腾讯云视立方 Flutter 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。

基础知识

本文主要介绍视频云 SDK 的点播播放功能，在此之前，先了解如下一些基本知识会大有裨益：

- **直播和点播**
  - 直播（LIVE）的视频源是主播实时推送的。因此，主播停止推送后，播放端的画面也会随即停止，而且由于是实时直播，所以播放器在播直播 URL 的时候是没有进度条的。
  - 点播（VOD）的视频源是云端的一个视频文件，只要未被从云端移除，视频就可以随时播放， 播放中您可以通过进度条控制播放位置，腾讯视频和优酷土豆等视频网站上的视频观看就是典型的点播场景。
- **协议的支持**
  通常使用的点播协议如下，现在比较流行的是 HLS（以“http”打头，以“.m3u8”结尾）的点播地址：

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

特别说明

视频云 SDK **不会对播放地址的来源做限制**，即您可以用它来播放腾讯云或非腾讯云的播放地址。但视频云 SDK 中的播放器只支持 FLV 、RTMP 和 HLS（m3u8）三种格式的直播地址，以及 MP4、HLS（m3u8）和 FLV 三种格式的点播地址。

SDK 集成

步骤1：集成 SDK 开发包

下载和集成 SDK 开发包，请参考 集成指引 。

步骤2：创建 controller

```
TXVodPlayerController _controller = TXVodPlayerController();
```

步骤3：设置监听事件

```
// 监听视频宽高变化，设置合适的宽高比例，也可自行设置宽高比例，视频纹理也会根据比例进行相应拉伸
playEventSubscription = _controller.onPlayerEventBroadcast.listen((event) async {
  // Subscribe to event distribution
  final int code = event["event"];
  if (code == TXVodPlayEvent.PLAY_EVT_CHANGE_RESOLUTION) {
    int? videoWidth = event[TXVodPlayEvent.EVT_PARAM1];
    int? videoHeight = event[TXVodPlayEvent.EVT_PARAM2];
  }
});
```

步骤4：添加布局

```
@override
Widget build(BuildContext context) {
return Container(
  decoration: BoxDecoration(
      image: DecorationImage(
        image: AssetImage("images/ic_new_vod_bg.png"),
        fit: BoxFit.cover,
      )),
  child: Scaffold(
      backgroundColor: Colors.transparent,
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        title: const Text('点播'),
      ),
      body: SafeArea(
          child: Container(
            height: 150,
            color: Colors.black,
            child: Center(
              child: _aspectRatio > 0
                  ? AspectRatio(
                aspectRatio: _aspectRatio,
                child: TXPlayerVideo(
                  androidRenderType: _renderType,
                  onRenderViewCreatedListener: (viewId) {
                    _controller.setPlayerView(viewId);
                  },
                ),
              ) : Container(),
            ),
          ))));
}
```

步骤5：播放器初始化（12.3.1 以及之后版本不再需要调用）

```
// 初始化播放器，分配共享纹理
await _controller.initialize();
```

步骤6：启动播放

通过 url 方式

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

```
// 播放视频资源
String _url =

"http://1400329073.vod2.myqcloud.com/d62d88a7vodtranscq1400329073/59c68fe7528589
0800381567412/adp.10.m3u8";
await _controller.startVodPlay(_url);
```

通过 field 方式

```
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
TXPlayInfoParams params = TXPlayInfoParams(appId: 1252463788,
        fileId: "4564972819220421305", psign: "psignxxxxxxx");
await _controller.startVodPlayWithParams(params);
```

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7：结束播放

结束播放时**记得调用 controller 的销毁方法**，尤其是在下次 startVodPlay 之前，否则可能会产生大量的内存泄露以及闪屏问题。

```
@override
void dispose() {
  _controller.dispose();
  super.dispose();
}
```

基础功能使用

1、播放控制

- **开始播放**

```
// 开始播放
_controller.startVodPlay(url)
```

- **暂停播放**

```
// 暂停播放
_controller.pause();
```

- **恢复播放**

```
// 恢复播放
_controller.resume();
```

- **结束播放**

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

- **结束播放器**

```
// 释放controller
_controller.dispose();
```

- **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。

```
double time = 600; // double，单位为 秒
// 调整进度
_controller.seek(time);
```

- **Seek 到视频流指定 PDT 时间点**

  跳转到视频流指定 PDT（Program Date Time ）时间点，可实现视频快进、快退、进度条跳转等功能，目前只支持 HLS 视频格式。
  注意：播放器高级版 11.7 版本开始支持。

```
int pdtTimeMs = 600; // 单位为 毫秒
_controller.seekToPdtTime(time);
```

- **从指定时间开始播放**

  首次调用 startVodPlay 之前，支持从指定时间开始播放。

```
double startTimeInSecond = 60; // 单位：秒
_controller.setStartTime(startTimeInSecond);    // 设置开始播放时间
_controller.startVodPlay(url);
```

2、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
_controller.setRate(1.2);
```

3、循环播放

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

4、静音设置

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

5 、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。

```
_controller.setAutoPlay(false);  // 设置为非自动播放
_controller.startVodPlay(url);    // startVodPlay 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
// ......
_controller.resume();  // 广告展示完调用 resume 开始播放视频
```

6、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。

```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
Map<String, String> httpHeaders = {'Referer': 'Referer Content'};
```

```
playConfig.headers = httpHeaders;
_controller.setConfig(playConfig);
```

7、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 **startVodPlay**，否则会产生比较严重的花屏问题。

```
_controller.stopPlay(true);
_controller.enableHardwareDecode(true);
_controller.startVodPlay(url);
```

8、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。可以通过下面方法进行清晰度设置。

```
// 在收到播放器 PLAY_EVT_VOD_PLAY_PREPARED 事件调用 getSupportedBitrates 才会有值返回
List _supportedBitrates = (await _controller.getSupportedBitrates())!;; //获取多码率数组
int index = _supportedBitrates[i];  // 指定要播的码率下标
_controller.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 `_controller.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。
如果您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，从而避免播放后切换码流。详细方法参见 播放器配置#启播前指定分辨率 。

9、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。可以通过下面方法开启码流自适应。

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

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

10、开启平滑切换码率

在启动播放前，通过开启平滑切换码率，在播放过程中切换码率，可以达到无缝平滑切换不同清晰度。对比关闭平滑切换码率，切换过程比较耗时、体验更好，可以根据需求进行设置。

```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
/// 设为true，可平滑切换码率，设为false时，可提高多码率地址打开速度
playConfig.smoothSwitchBitrate = true;
_controller.setConfig(playConfig);
```

11、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。
通过 `onPlayerEventBroadcast` 接口监听播放器事件，进度通知会通过 PLAY_EVT_PLAY_PROGRESS 事件回调到您的应用程序。

```
_controller.onPlayerEventBroadcast.listen((event) async {
  if(event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_PROGRESS) {// 更多详细请查看iOS或者Android原生SDK状态码
    // 可播放时长，即加载进度，单位是毫秒
    double playableDuration = event[TXVodPlayEvent.EVT_PLAYABLE_DURATION_MS].toDouble();
    // 播放进度，单位是秒
    int progress = event[TXVodPlayEvent.EVT_PLAY_PROGRESS].toInt();
    // 视频总长，单位是秒
    int duration = event[TXVodPlayEvent.EVT_PLAY_DURATION].toInt();
  }
});
```

12、播放网速监听

通过 `onPlayerNetStatusBroadcast` 接口监听播放器的网络状态，如： `NET_STATUS_NET_SPEED` 。

```
_controller.onPlayerNetStatusBroadcast.listen((event) {
    (event[TXVodNetEvent.NET_STATUS_NET_SPEED]).toDouble();
});
```

13、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.
可以通过下面两种方法获取分辨率信息
方法1：通过 `onPlayerNetStatusBroadcast` 的 `NET_STATUS_VIDEO_WIDTH` 和 `NET_STATUS_VIDEO_HEIGHT` 获取视频的宽和高。
方法2：在收到播放器的PLAY_EVT_VOD_PLAY_PREPARED 事件回调后，直接调用 `getWidth()` 和 `getHeight()` 获取当前宽高。

```
 _controller.onPlayerNetStatusBroadcast.listen((event) {
  double w = (event[TXVodNetEvent.NET_STATUS_VIDEO_WIDTH]).toDouble();
  double h = (event[TXVodNetEvent.NET_STATUS_VIDEO_HEIGHT]).toDouble();
});

// 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
_controller.getWidth();
_controller.getHeight();
```

14、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。

```
FTXVodPlayConfig playConfig = FTXVodPlayConfig();
playConfig.maxBufferSize = 10;  ///  播放时最大缓冲大小。单位：MB
_controller.setConfig(playConfig);
```

15、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。

```
//设置播放引擎的全局缓存目录和缓存大小，//单位MB
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
//设置播放引擎的全局缓存目录
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");
```

16、外挂字幕

注意：此功能需要播放器高级版 11.7 版本开始支持。
播放器高级版 SDK 支持添加和切换外挂字幕，现已支持 SRT 和 VTT 这两种格式的字幕。
实践教程：建议在 `startVodPlay` 之前添加字幕和配置字幕样式，在收到 `VOD_PLAY_EVT_VOD_PLAY_PREPARED` 事件后，调用 selectTrack 选择字幕。添加字幕后，并不会主动加载字幕，调用 `selectTrack` 后，才会加载字幕，字幕选择成功会有 `VOD_PLAY_EVT_SELECT_TRACK_COMPLETE` 事件回调。选择字幕后，字幕文本内容会通过 `TXVodPlayEvent.EVENT_SUBTITLE_DATA` 。
事件回调，字幕的显示在需要业务方自行处理。

步骤 1：添加外挂字幕

```
// 添加外挂字幕，传入 字幕url， 字幕名称， 字幕类型， 建议在启动播放器前添加
controller.addSubtitleSource("https://mediacloud-76607.gzc.vod.tencent-
cloud.com/DemoResource/subtitleVTT.vtt", "subtitleName",
TXVodPlayEvent.VOD_PLAY_MIMETYPE_TEXT_SRT)

// 开始播放视频后，监听字幕文本内容回调
_controller.onPlayerEventBroadcast.listen((event) async {
if(event["event"] == TXVodPlayEvent.EVENT_SUBTITLE_DATA) {
  // 字幕文本内容，可用于显示
  String subtitleDataStr = event[TXVodPlayEvent.EXTRA_SUBTITLE_DATA] ?? "";
}
});
```

步骤2：播放后切换字幕

```
// 开始播放视频后，选中添加的外挂字幕， 在收到 VOD_PLAY_EVT_VOD_PLAY_PREPARED 事件后调用
_controller.onPlayerEventBroadcast.listen((event) async {
if(event["event"] == TXVodPlayEvent.PLAY_EVT_VOD_PLAY_PREPARED) {
    List<TXTrackInfo> trackInfoList = await
_vodPlayerController.getSubtitleTrackInfo();
      for (TXTrackInfo tempInfo in trackInfoList) {
        if(tempInfo.name == "subtitleName") {
          // 选中字幕
          _vodPlayerController.selectTrack(tempInfo.trackIndex);
        } else {
          // 其它字幕不需要的话， 进行deselectTrack
          _vodPlayerController.deselectTrack(tempInfo.trackIndex);
        }
      }
    }
});

// 如果需要，可以监听轨道切换消息
_controller.onPlayerEventBroadcast.listen((event) async {
  if(event["event"] == TXVodPlayEvent.VOD_PLAY_EVT_SELECT_TRACK_COMPLETE) {
    int trackIndex = event[TXVodPlayEvent.EVT_KEY_SELECT_TRACK_INDEX];
    int errorCode = event[TXVodPlayEvent.EVT_KEY_SELECT_TRACK_ERROR_CODE];
  }
});
```

步骤3：监听字幕文本内容

```
// 开始播放视频后，监听字幕文本内容回调
_controller.onPlayerEventBroadcast.listen((event) async {
if(event["event"] == TXVodPlayEvent.EVENT_SUBTITLE_DATA) {
  // 字幕文本内容，可用于显示
  String subtitleDataStr = event[TXVodPlayEvent.EXTRA_SUBTITLE_DATA] ?? "";
}
});
```

17、多音轨切换

注意：此功能需要播放器高级版 11.7 版本开始支持。

播放器高级版 SDK 支持切换视频内置的多音轨。用法参见如下代码：

```
// 返回音频轨道信息列表
List<TXTrackInfo> trackInfoList = await _vodPlayerController.getAudioTrackInfo();
for (TXTrackInfo tempInfo in trackInfoList) {
  if(tempInfo.trackIndex == 0) {
    // 通过判断 trackIndex 或者 name 切换到需要的音轨
    _vodPlayerController.selectTrack(tempInfo.trackIndex);
  } else {
```

```
    // 不需要的音轨进行 deselectTrack
    _vodPlayerController.deselectTrack(tempInfo.trackIndex);
  }
}
```

18、进入画中画

目前双端均支持画中画能力，其中 Android 端可以传递自定义窗口图片，大小限制为 1MB，不传则使用默认图标，传空字符串会隐藏图标。iOS 则只能使用系统默认图标，用法如下：

```
_playerController.enterPictureInPictureMode(
    backIconForAndroid: backIconForAndroid,
    playIconForAndroid: playIconForAndroid,
    pauseIconForAndroid: pauseIconForAndroid,
    forwardIconForAndroid: forwardIconForAndroid);
```

如果提升 IOS 画中画体验，或者需要 flutter IOS 端直播画中画能力，可以参考 IOS高级画中画能力，需要高级版支持以及添加资源文件。

19、绑定纹理

build 中布局了 TXPlayerVideo 后，需要与播放器绑定才能将画面显示在 TXPlayerVideo 中，示例如下：

```
  controller.setPlayerView(viewId);
```

20、设置平铺模式

controller 提供了两种平铺模式，一种是按照视频比例优先展示完整画面 `FTXPlayerRenderMode.ADJUST_RESOLUTION`，另外一种是按照视频比例，铺满容器 `FTXPlayerRenderMode.FULL_FILL_CONTAINER` 。示例如下：

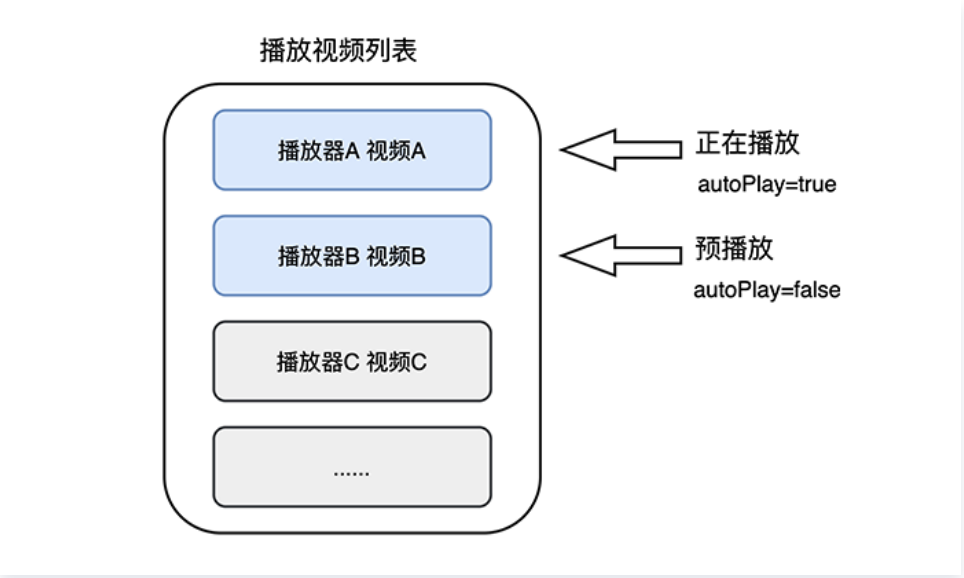
```
  _controller.setRenderMode(FTXPlayerRenderMode.ADJUST_RESOLUTION);
```

进阶功能使用

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，会占用下载带宽和线程资源。建议视频预播放并发个数控制在3个以内。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 TXVodPlayerController 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 startVodPlay 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";
controller.setAutoPlay(isAutoPlay: true);
controller.startVodPlay(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";
controller.setAutoPlay(isAutoPlay: false);
controller.startVodPlay(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。

```
controller.onPlayerEventBroadcast.listen((event) async {//订阅状态变化
  if(event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_END) {
    await _controller_A.stop();
    await _controller_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 = 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。

```
FTXVodPlayConfig config = FTXVodPlayConfig();
config.maxBufferSize = 10; // 播放时最大缓冲大小。单位：MB
_controller.setPlayConfig(config); // 把config 传给 controller
```

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 。

> **注意：**
> 视频预下载会占用下载带宽和线程资源，建议进行队列控制，并发个数控制在3个以内。

使用示例：

通过媒资 URL 预下载

```
//设置播放引擎的全局缓存目录和缓存大小
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
// 该缓存路径默认设置到app沙盒目录下，postfixPath只需要传递相对缓存目录即可，不需要传递整个绝对路径。
// Android 平台：视频将会缓存到sdcard的Android/data/your-pkg-name/files/testCache 目录。
// iOS 平台：视频将会缓存到沙盒的Documents/testCache 目录。
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");

String palyrl = "http://****";
//启动预下载
int taskId = await TXVodDownloadController.instance.startPreLoad(palyrl, 3,
1920*1080,
  onCompleteListener:(int taskId,String url) {
    print('taskID=${taskId} ,url=${url}');
  }, onErrorListener: (int taskId, String url, int code, String msg) {
    print('taskID=${taskId} ,url=${url}, code=${code} , msg=${msg}');
  }
);

//取消预下载
```

```
TXVodDownloadController.instance.stopPreLoad(taskId);
```

通过媒资 FileId 预下载

```
//设置播放引擎的全局缓存目录和缓存大小
SuperPlayerPlugin.setGlobalMaxCacheSize(200);
// 该缓存路径默认设置到app沙盒目录下，postfixPath只需要传递相对缓存目录即可，不需要传递整个绝对路径。
// Android 平台：视频将会缓存到sdcard的Android/data/your-pkg-name/files/testCache 目录。
// iOS 平台：视频将会缓存到沙盒的Documents/testCache 目录。
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");

int retTaskId = -1;
TXVodDownloadController.instance.startPreload(TXPlayInfoParams(appId: 0, fileId: "your fileId"), 1, 720 * 1080,
    onStartListener: (taskId, fileId, url, params) {
      // TXVodDownloadController will call this block for callback taskId and videoInfo
      retTaskId = taskId;
    },
    onCompleteListener: (taskId, url) {
      // preDownload complete
    },
    onErrorListener: (taskId, url, code, msg) {
      // preDownload error
    });

//取消预下载
TXVodDownloadController.instance.stopPreLoad(retTaskId);
```

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 `TXVodDownloadController` 的视频下载方案实现 HLS 的离线播放。

> **说明:**
> 视频下载支持下载 MP4 和 HLS 视频，对应嵌套 HLS 视频，需要指定偏好清晰度（preferredResolution）。

步骤1：准备工作

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

```
// 该缓存路径默认设置到app沙盒目录下，postfixPath只需要传递相对缓存目录即可，不需要传递整个绝对路径。
// Android 平台：视频将会缓存到sdcard的Android/data/your-pkg-name/files/testCache 目录。
// iOS 平台：视频将会缓存到沙盒的Documents/testCache 目录。
SuperPlayerPlugin.setGlobalCacheFolderPath("postfixPath");
```

步骤2：开始下载

开始下载有 Fileid 和 URL 两种方式，具体操作如下：

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”。

> **注意：**
> 加密视频只能通过 Fileid 下载，psign 参数必须填写。

```
// QUALITY_240P  240p
// QUALITY_360P  360P
// QUALITY_480P  480p
// QUALITY_540P  540p
// QUALITY_720P  720p
// QUALITY_1080P 1080p
// QUALITY_2K    2k
// QUALITY_4K    4k
// quality参数可以自定义，取分辨率宽高最小值(如分辨率为1280*720, 期望下载此分辨率的流，quality传入 QUALITY_720P)
// 播放器 SDK 会选择小于或等于传入分辨率的流进行下载
TXVodDownloadMedialnfo medialnfo = TXVodDownloadMedialnfo();
TXVodDownloadDataSource dataSource = TXVodDownloadDataSource();
dataSource.appId = 1252463788;
dataSource.fileId = "4564972819220421305";
dataSource.quality = DownloadQuality.QUALITY_480P;
dataSource.pSign = "pSignxxxx";
medialnfo.dataSource = dataSource;
TXVodDownloadController.instance.startDonwload(medialnfo);
```

URL 方式

至少需要传入下载地址 URL，不支持嵌套 HLS 格式，仅支持单码流的HLS下载。userName 不传入具体值时，默认为"default"。

```
TXVodDownloadMedialnfo medialnfo = TXVodDownloadMedialnfo();
medialnfo.url =
"http://1500005830.vod2.myqcloud.com/43843ec0vodtranscq1500005830/00eb06a8860226
8011437356984/video_10_0.m3u8";
TXVodDownloadController.instance.startDonwload(medialnfo);
```

步骤3：任务信息

在接收任务信息前，需要先设置回调 listener。

```
TXVodDownloadController.instance.setDownloadObserver((event, info) {

}, (errorCode, errorMsg, info) {

});
```

可能收到的任务 event 事件有：

事件	说明
EVENT_DOWNLOAD_START	任务开始，表示 SDK 已经开始下载
EVENT_DOWNLOAD_PROGRESS	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
EVENT_DOWNLOAD_STOP	任务停止，当您调用 `stopDownload` 停止下载，收到此消息表示停止成功
EVENT_DOWNLOAD_FINISH	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

当回调 downlodOnErrorListener 方法的时候，代表下载错误，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。由于 downloader 可以同时下载多个任务，所以回调接口里带上了 `TXVodDownloadMedialnfo` 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 `TXVodDownloadController.instance.stopDownload()` 方法，参数为开始下载传入的 `TXVodDownloadMedialnfo` 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
// 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> downloadInfoList = await
TXVodDownloadController.instance.getDownloadList();
// 获取默认“default”用户已经下载完成的视频
List<TXVodDownloadMediaInfo> defaultUserDownloadList = [];
for(TXVodDownloadMediaInfo mediaInfo in downloadInfoList) {
  if("default" == mediaInfo.userName && mediaInfo.downloadState ==
TXVodPlayEvent.EVENT_DOWNLOAD_FINISH) {
    defaultUserDownloadList.add(mediaInfo);
  }
}
```

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。

```
// 获取某个视频相关下载信息
TXVodDownloadMediaInfo mediaInfo = TXVodDownloadMediaInfo();
mediaInfo.dataSource = TXVodDownloadDataSource();
mediaInfo.dataSource!.appId = 1500005830;
mediaInfo.dataSource!.fileId = "8602268011437356984";
// fileId下载必须传入quality。quality也可以通过
CommonUtils.getDownloadQualityBySize(width, height)来获取
mediaInfo.dataSource!.quality = DownloadQuality.QUALITY_HD;
TXVodDownloadMediaInfo downloadInfo = await
TXVodDownloadController.instance.getDownloadInfo(mediaInfo);
int? duration = downloadInfo.duration;     // 获取总时长
int? playableDuration = downloadInfo.playableDuration; // 获取已下载的可播放时长
double? progress = downloadInfo.progress;     // 获取下载进度
String? playPath = downloadInfo.playPath; // 获取离线播放路径，传给播放器即可离线播放
int? downloadState = downloadInfo.downloadState; // 获取下载状态，具体参考STATE_xxx常量
```

获取某个 URL 相关下载信息，需要传入 URL 信息。目前url下载不支持嵌套m3u8和mp4下载。

```
TXVodDownloadMediaInfo mediaInfo = TXVodDownloadMediaInfo();
mediaInfo.url =
"http://1253131631.vod2.myqcloud.com/26f327f9vodgzp1253131631/f4bdff7990318682229240
43041/playlist.m3u8";
TXVodDownloadController.instance.getDownloadInfo(mediaInfo);
```

```
// 删除下载信息
bool result = await
TXVodDownloadController.instance.deleteDownloadMediaInfo(mediaInfo);
```

步骤6：播放离线视频

通过以上步骤获取到的mediaInfo的downloadState为EVENT_DOWNLOAD_FINISH，并且mediaInfo的playPath有值，则代表获取视频缓存完成，可以直接传给controller进行播放，代码如下：

```
String cacheVideoUrl = cacheMediaInfo.playPath!;
vodPlayerController.startVodPlay(cacheVideoUrl);
```

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。
在腾讯云控制台提取到appId， 加密视频的fileId 和 psign后，可以通过下面的方式进行播放：

```
// psign 即播放器签名，签名介绍和生成方式参见链接：
https://cloud.tencent.com/document/product/266/42436
TXPlayInfoParams params = TXPlayInfoParams(appId: 1252463788,
        fileId: "4564972819220421305", psign: "psignxxxxxxx");
_controller.startVodPlayWithParams(params);
```

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：

```
FTXVodPlayConfig config = FTXVodPlayConfig();
// 如果不配置preferredResolution，则在播放多码率视频的时候优先播放720 * 1280分辨率的码率
config.preferredResolution = 720 * 1280;
config.enableAccurateSeek = true;  // 设置是否精确 seek，默认 true
config.progressInterval = 200;  // 设置进度回调间隔，单位毫秒
config.maxBufferSize = 50;  // 最大预加载大小，单位 MB
_controller.setPlayConfig(config);
```

启播前指定分辨率

播放 HLS 的多码率视频源，如果您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。

```
FTXVodPlayConfig config = FTXVodPlayConfig();
// 传入参数为视频宽和高的乘积(宽 *  高)，可以自定义值传入，默认 720 * 1280
config.preferredResolution = 720 * 1280;
_controller.setPlayConfig(config);
```

启播前指定媒资类型

当提前知道播放的媒资类型时，可以通过配置 `FTXVodPlayConfig#` mediaType减少播放器SDK内部播放类型探测，提升启播速度。

```
FTXVodPlayConfig config = FTXVodPlayConfig();
config.mediaType = TXVodPlayEvent.MEDIA_TYPE_FILE_VOD;    // 用于提升MP4启播速度
config.mediaType = TXVodPlayEvent.MEDIA_TYPE_HLS_VOD;    // // 用于提升HLS启播速度
```

```
_controller.setPlayConfig(config);
```

设置播放进度回调时间间隔

```
FTXVodPlayConfig config = FTXVodPlayConfig();
config.progressInterval = 200;  // 设置进度回调间隔，单位毫秒
_controller.setPlayConfig(config);
```

6、HEVC 自适应降级播放

播放器支持同时传入 HEVC 和其它视频编码格式。例如： H.264 的播放链接， 当播放机型不支持 HEVC 格式时，将自动降级为配置的其它编码格式（如： H.264 ）的视频播放。

> **注意：**
> 播放器高级版 12.0 版本开始支持。

```
//设置备选播放链接
String backupPlayUrl = "${backupPlayUrl}";  // 备选播放链接
await _controller.setStringOption(TXVodPlayEvent.VOD_KEY_BACKUP_URL, backupPlayUrl);
// 设置 H.264 格式等备选播放链接地址 // 设置 H.264 格式等备选播放链接地址
// 备选播放播放资源的类型,比步骤非必须，MEDIA_TYPE 为 backupPlayUrl 对应的媒体类型，如：MP4 对应为 TXVodPlayEvent.MEDIA_TYPE_HLS_VOD， 如果不设置则为 AUTO 类型
// await _controller.setStringOption(TXVodPlayEvent.VOD_KEY_BACKUP_URL_MEDIA_TYPE, TXVodPlayEvent.MEDIA_TYPE_HLS_VOD);

// 设置原始 HEVC 播放链接
await _controller.setStringOption(TXVodPlayEvent.VOD_KEY_VIDEO_CODEC_TYPE, TXVodPlayEvent.VOD_PLAY_MIMETYPE_H265);  // 指定原始 HEVC 视频编码类型
String hevcPlayUrl = "${hevcPlayUrl}";
await _controller.startVodPlay(hevcPlayUrl);
```

播放器事件监听

您可以通过 `TXVodPlayerController` 的 `onPlayerEventBroadcast` 来监听播放器的播放事件，来向您的应用程序同步信息。

播放事件通知（onPlayerEventBroadcast）

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
VOD_PLAY_EVT_SEEK_COMPLETE	2019	Seek 完成，10.3版本开始支持

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）
VOD_PLAY_EVT_VOD_PLAY_FIRST_VIDEO_PACKET	2017	收到首帧数据包事件，12.0版本开始支持。

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
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：startVodPlay(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
EVT_IMAGESPRIT_WEBVTTURL	雪碧图 web vtt 描述文件下载 URL，10.2版本开始支持
EVT_IMAGESPRIT_IMAGEURL_LIST	雪碧图图片下载 URL，10.2版本开始支持
EVT_DRM_TYPE	加密类型，10.2版本开始支持

通过 onPlayerEventBroadcast 获取视频播放过程信息示例：

```
_controller.onPlayerEventBroadcast.listen((event) async {
  if (event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_BEGIN || event["event"] ==
TXVodPlayEvent.PLAY_EVT_RCV_FIRST_I_FRAME) {
  // code ...
  } else if (event["event"] == TXVodPlayEvent.PLAY_EVT_PLAY_PROGRESS) {
  // code ...
  }
});
```

播放状态反馈（onPlayerNetStatusBroadcast）

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，单位 Kbps。
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：

```
_controller.onPlayerNetStatusBroadcast.listen((event) async {
  int videoWidth = event[TXVodNetEvent.NET_STATUS_VIDEO_WIDTH];
});
```

视频播放状态反馈

视频播放状态会在每一次播放状态切换的时候进行通知。
事件通过枚举类 TXPlayerState 来传递事件

状态	含义
paused	暂停播放
failed	播放失败
buffering	缓冲中
playing	播放中
stopped	停止播放
disposed	控件释放了

通过 onPlayerState 获取视频播放状态示例：

```
_controller.onPlayerState.listen((val) { });
```

系统音量监听

为了方便监控视频播放音量，SDK 在 flutter 层做了对原生层的音量变化通知进行了事件封装，可以直接通过 SuperPlayerPlugin 来监听当前设备的音量变化。
通过 onEventBroadcast 获取设备音量状态示例：

```
SuperPlayerPlugin.instance.onEventBroadcast.listen((event) {
  int eventCode = event["event"];
});
```

相关事件含义如下：

状态	值	含义
EVENT_VOLUME_CHANGED	1	音量发生变化
EVENT_AUDIO_FOCUS_PAUSE	2	失去音量输出播放焦点，仅使用 Android
EVENT_AUDIO_FOCUS_PLAY	3	获得音量输出焦点，仅使用 Android

画中画事件监听

由于 SDK 使用的画中画是基于系统提供的画中画能力，进入画中画之后，提供了一系列通知来帮助用户对当前界面做响应的调整。

状态	值	含义
EVENT_PIP_MODE_ALREADY_ENTER	1	已经进入画中画模式
EVENT_PIP_MODE_ALREADY_EXIT	2	已经退出画中画模式
EVENT_PIP_MODE_REQUEST_START	3	开始请求进入画中画模式
EVENT_PIP_MODE_UI_STATE_CHANGED	4	pip UI 状态发生变动，仅在 Android 31以上生效
EVENT_IOS_PIP_MODE_RESTORE_UI	5	重置 UI，仅在 iOS 生效。
EVENT_IOS_PIP_MODE_WILL_EXIT	6	将要退出画中画，仅在 iOS 生效。

使用 onExtraEventBroadcast 监听画中画事件的示例如下：

```
SuperPlayerPlugin.instance.onExtraEventBroadcast.listen((event) {
  int eventCode = event["event"];
});
```

高级版功能
Web 端

最近更新时间：2023-08-22 18:04:02

高级版功能

播放器 SDK Web 端（TCPlayer）的5.0.0版本中在已有功能基础上，新增了两款高级功能，若您需要，可购买高级版 License 使用；若您无需使用，可直接前往控制台申请免费的基础版 License 以继续使用基础版功能。

高级功能	描述
VR 播放插件	支持播放全景 VR 视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，PC 端设备支持鼠标在界面上拖动画面查看。
Web 安全插件	检测 Web 端播放环境和播放状态是否正常，异常环境下将中断视频播放，保护视频安全。插件包含 MSE 环境检测、安全结构检查和接口响应完整性校验。

移动端

最近更新时间：2024-06-14 17:01:41

高级版功能

播放器 SDK 目前分为基础版和高级版，本次新增的高级功能中，一部分可以直接配合基础版 SDK 使用，一部分必须在高级版 SDK 中才支持，但不论是在基础版中直接使用还是在高级版中使用，**只要需要使用高级功能，就需要购买播放器移动端高级版 License** 。功能和支持情况部分如下表，更多能力支持请参见 播放器 SDK 产品功能 文档说明。

功能	描述	支持的SDK
短视频播放组件	以极低的接入成本，实现极速首帧、无感启播、丝滑切换的短视频播放体验。结合预播放、预下载、播放器复用、精准流量控制、加载策略等技术，在保证低能耗的前提下实现极致流畅的播放效果。	11.3 及以上的基础版 SDK、高级版 SDK
高级画中画组件	相对基础画中画，新增支持加密视频画中画、离线播放画中画和“秒切”效果。	11.4 及以上的基础版 SDK、高级版 SDK
VR 播放插件	支持播放全景 VR 视频源，移动端设备支持手指拖动或陀螺仪操作以查看全景视频内容，PC 端设备支持鼠标在界面上拖动画面查看	11.3 及以上的基础版 SDK、高级版 SDK
外挂字幕	支持导入自定义字幕文件，Web 端支持 WebVTT 格式，移动端支持 VTT、SRT 格式	高级版 SDK
DASH 协议支持	支持标准协议的 DASH 视频播放和 DASH 的自适应码流播放，可根据网络带宽自动选择合适的码率进行播放。	高级版 SDK
Quic 加速	支持 Quic 传输协议，有效提升视频传输效率	高级版 SDK
AV1	支持播放 AV1 编码格式的视频（部分支持）	高级版 SDK
多音轨	支持播放含多音轨的视频文件，播放时可切换音轨，如英文切换中文。	高级版 SDK
商业 DRM	提供苹果 Fairplay、谷歌 Widevine 原生加密方案	高级版 SDK

API 文档
Web

最近更新时间：2025-06-10 14:16:01

本文档是介绍适用于直播和点播播放的 Web 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 对象可配置的参数如下表：

名称	类型	默认值	说明
appID	String	无	通过 fileID 播放点播媒体文件时必选，为对应腾讯云账号的 appID
fileID	String	无	通过 fileID 播放点播媒体文件时必选，为点播媒体文件的 ID
psign	String	无	播放器签名，通过 fileID 播放时必传，详情参见 播放器签名
licenseUrl	String	无	播放器 License 地址，查看 播放器 License
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。
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，具体参见 controlBar 参数列表。
reportable	Boolean	true	设置是否开启数据上报。
fakeFullscreen	Boolean	false	设置开启伪全屏，通过样式控制来实现全屏效果。
plugins	Object	无	设置插件功能属性的参数组合，具体参见 plugins 插件参数列表。
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，具体参见 webrtcConfig 参数列表。
xp2pConfig	Object	无	P2P 的启动配置，具体参见 xp2pConfig 参数列表。P2P 功能详情请参见 X-P2P。

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。
浏览器劫持视频播放问题参见 常见问题说明。

controlBar 参数列表

controlBar 参数可以配置播放器控制栏的功能，支持的属性如下表：

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	是否显示时间分隔符。

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。
fullscreenRotate	Boolean	false	是否显示画面旋转按钮。
QualitySwitcherMenuButton	Boolean	true	是否显示清晰度切换菜单。
pictureInPictureToggle	Boolean	false	是否显示画中画按钮。

注意：
controlBar 参数在浏览器劫持播放的状态下将无效。
浏览器劫持视频播放问题参见 常见问题说明 。

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，支持文字和图片，支持的属性为： • type：String 水印类型为文字或图片，取值为 text | image，默认 text，非必传。 • content：String 文字水印内容，必传。 • speed：Number 水印移动速度，取值范围 0-1, 默认值 0.2，非必传。 • opacity：Number 文字水印透明度，取值范围 0-1，非必传。 • fontSize：String 文字字体大小，默认12px，非必传。 • color： String 文字颜色，非必传。 • left：String, 文字位置，支持单位为百分比和 px，该字段设置时，speed 字段无效，非必传。 • top、right、bottom：说明同 left。 • width：String 图片水印宽度，非必传。

			• height：String 图片水印高度，非必传 。
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 }`
PlayList	Object	无	设置播放列表，支持的属性如下： `{ // 要播放的视频信息集合 data: [{ fileID: String, appID: String, duration: Number, // 视频时长 text: String, // 视频名称 psign: String, // 播放器签名 img: String, // 封面图 }], title: String, // 列表标题 loop: Boolean, // 是否循环播放 }`
VR	Object	无	高级版 License 支持，详情参见 Web 高级功能 - VR 播放插件（TCPlayerVRPlugin）
SafeCheck	Object	无	高级版 License 支持，详情参见 Web 高级功能 - 安全检查插件（TCPlayerSafeCheckPlugin）

webrtcConfig 参数列表

webrtcConfig 参数来控制播放 webrtc 过程中的行为表现，支持的属性如下表：

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
showLog	Boolean	false	是否在控制台打印日志
receiveSEI	Boolean	false	是否接收 SEI 信息
fallback	Boolean	true	是否允许降级。
fallbackUrl	String	无	如有降级，会降级到该流地址。

xp2pConfig 参数列表

使用 X-P2P 前，需要申请开通，请移步 X-P2P 单击申请，申请后我们会有专门的产品支持人员联系您。
更多详细资料，请参考 X-P2P 产品文档 。

- **公共参数**

名称	类型	默认值	说明
useXP2P	Boolean	false	是否开启 XP2P
format	String	无	告知 P2P 需要支持的媒体协议，请根据当前播放的视频格式填写，可选值如下： • flv • hls • webrtc
tencentCloudAppId	Number	无	在腾讯云账号的 APPID (控制台查看路径：**账号中心 > 账号信息 > 基本信息 > APPID**)
xp2pAppId	String	无	X-P2P 分配的 ID，由我们邮件提供
xp2pAppKey	String	无	X-P2P 分配的 Key，由我们邮件提供
xp2pPackage	String	无	X-P2P 分配的 Package，由我们邮件提供

- **flv 协议额外参数**

名称	类型	默认值	说明
bizId	String	无	由我们邮件提供
xp2pPlayDomain	String	无	flv 协议的拉流域名，由我们邮件提供
authMode	String	无	鉴权模式，由我们邮件提供
debug	Boolean	false	debug 开关

- **hls 协议额外参数**

名称	类型	默认值	说明

videoType	String	VOD	当前播放的 HLS 是直播还是点播，请准确填写，可选值如下： • LIVE • VOD
channelId	String	自动生成	可以主动为当前资源生成一个 ID，如果不填，则默认内部自动生成。生成规则参见如下 HLS 资源 ID 生成规则
channelIdWithHost	Boolean	true	默认为 true，可选配置，通常不需要修改这个配置。详细解释参见如下 HLS 资源 ID 生成规则
channelIdWithSearch	Boolean	false	默认为 false，可选配置，通常不需要修改这个配置。详细解释参见如下 HLS 资源 ID 生成规则

HLS 资源 ID 生成规则

资源 ID 是 P2P 分享的单位，相同的资源 ID 的节点才能互相 P2P 分享。不同视频必须确保资源 ID 不同，否则会串流。

1.1 **主动传入**

可以通过设置参数 `channelId` 字段，主动为当前视频指定一个资源 ID，必须保证能唯一标识这个文件，避免串流。

1.2 **默认生成**

如果没有传入 `channelId` 字段，sdk 会默认为每一个 url 生成一个 ID，相同 ID 的会互相 P2P 分享，ID 生成规则如下：

(默认) 截取 host 和 path 部分生成 MD5。

> 例如：https://a.b.com/p1/p2/p3.m3u8?m=1&n=2，则 ID = MD5('a.b.com/p1/p2/p3.m3u8')

1.3 **传入参数控制默认生成规则**

- (可选，默认为 true) 通过传入 `channelIdWithHost` 参数，true 表示 ID 包含 host 部分。
- (可选，默认为 false) 通过传入 `channelIdWithSearch` 参数，false 表示包含 search 部分。

> 例如：https://a.b.com/p1/p2/p3.m3u8?m=1&n=2
> 如果传入 { channelIdWithHost: true, channelIdWithSearch: true } 则 ID = MD5('a.b.com/p1/p2/p3.m3u8?m=1&n=2')
> 如果传入 { channelIdWithHost: false, channelIdWithSearch: false } 则 ID = MD5('/p1/p2/p3.m3u8')

> **说明：**
> 可以根据自己业务 url 生成规则，自行选择搭配，目的是确保不能互相 P2P 的视频 url，生成不同的资源 ID，可以互通的 url，生成相同的资源 ID。

1.4 **多码率 M3U8 说明**

如果播放的视频是多码率 M3U8，我们内部会保证播放不同码率的节点不会互相 P2P。

- **X-P2P 协议支持**

音视频协议	用途	PC 浏览器	Android	iOS
FLV	直播	支持	支持	不支持

		• chrome 55+ • firefox 65+ • safari 11+	• chrome 55+ • firefox 65+ • 微信浏览器	
HLS	直播，点播	支持 • chrome 55+ • firefox 65+ • safari 11+	支持 • chrome 55+ • firefox 65+ • 微信浏览器	不支持
WebRTC	直播	部分支持 • Chrome 94+ • edge 94+	部分支持 • Chrome 94+ • edge 94+	不支持

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。
loadVideoByID()	(Object)	无	通过 fileID 播放时，可通过这个方法切换视频，参数为由fileID、appID、psign 组成的对象。
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。
unload()	无	无	停止播放，会断流。(5.0.0版本及以上)
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。
muted()	(Boolean)	(Boolean)	获取或设置播放器是否静音
playbackRate()	(Number)[0, 1]	(Number)	获取或设置播放倍速
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。
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。
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。
blocked	自动播放被浏览器阻止时触发。（原 2005 回调事件统一合并到 blocked 事件中）
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。
webrtcfallback	播放 webrtc 时触发降级

WebrtcEvent 列表

播放器可以通过 webrtcevent 获取播放 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
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。
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。
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 。
50	License 校验失败。
51	License 已过期。
52	License 域名校验失败。
53	License 时间验证失败。
54	License 类型错误。
55	缺少 License url。
56	获取 License 数据失败。
1005	没有找到可以播放的自适应码流
1013	播放器签名缺少 contentInfo 字段
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。
-2002	快直播拉流接口后台返回报错（例如流不存在、鉴权失败等）
-2006	快直播多分辨率平滑切换接口请求失败

iOS
TXVodPlayer

最近更新时间：2025-05-23 17:46:52

TXVodPlayer API 简介

TXVodPlayer 是核心播放类，主要负责播放、暂停和倍速控制等播放控制。播放器的完整能力请参见 播放器 SDK 功能说明 。

接口概览

基础播放接口

API	描述
startVodPlay:	播放 HTTP URL 形式地址。10.7版本开始，startPlay 变更为 startVodPlay，需要通过 {@link TXLiveBase#setLicence} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。 直播 Licence、短视频 Licence 和播放器 Licence 均可使用，若您暂未获取上述 Licence ，可单击 播放器 License 进行申请，正式版 License 需购买。
startVodPlayWithParams:	以 fileId 形式播放，传入 TXPlayInfoParams 参数。10.7版本开始，startPlay 变更为 startVodPlay，需要通过 {@link TXLiveBase#setLicence} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。 直播 Licence、短视频 Licence 和播放器 Licence 均可使用，若您暂未获取上述 Licence 可单击 播放器 License 进行申请，正式版 License 需购买。
startPlayDrm:	播放 DRM 加密视频。
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。
seek:accurateSeek:	跳转到视频流指定时间点，单位秒，小数点后精确到3位。支持精准 seek。
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。
isAutoPlay	设置点播是否 startPlay 后自动开始播放，默认自动播放。

enableHWAcceleration	设置是否开启硬件加速，默认开启。
setStartTime:	设置播放开始时间。
setupVideoWidget:insertIndex:	设置 Video 渲染 View，该控件承载视频内容的展示。
removeVideoWidget	移除 Video 渲染 View。
token	加密 HLS 的 token。
getEncryptedPlayKey:	获取加固加密播放密钥。
loop	是否循环播放。
addSubtitleSource:name:mimeType:	添加外挂字幕（播放器高级版本才支持）。
getSubtitleTrackInfo	返回字幕轨道信息列表（播放器高级版本才支持）。
getAudioTrackInfo	返回音频轨道信息列表（播放器高级版本才支持）。
selectTrack:	选择轨道（播放器高级版本才支持）。
deselectTrack:	取消选择轨道（播放器高级版本才支持）。
seekToPdtTime:	跳转到视频流指定 PDT（Program Date Time ）时间点，可实现视频快进、快退、进度条跳转等功能，目前只支持 HLS 视频格式（播放器高级版 11.6 版本开始支持）。 参数单位毫秒(ms)。

播放器配置接口

API	描述
config	设置播放器配置信息，配置信息请参见 TXVodPlayConfig 。
setExtentOptionInfo:	设置播放器业务参数，参数格式为 `<NSString *, id>` 。
setSubtitleStyle:	设置字幕样式信息，可在播放后对字幕样式进行更新（播放器高级版本才支持）。

setAudioNormalization:	设置音量均衡，响度范围，可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）
	• 关：AUDIO_NORMALIZATION_OFF
	• 开：
	  ○ AUDIO_NORMALIZATION_STANDARD（标准）
	  ○ AUDIO_NORMALIZATION_LOW（低）
	  ○ AUDIO_NORMALIZATION_HIGH（高）
	可填自定义数值：从低到高，范围-70 – 0 LUFS。

视频相关接口

API	描述
enableHWAcceleration	启用或禁用视频硬解码，默认开启。
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。
supportedBitrates	当播放地址为 HSL 时，返回支持的码率（清晰度）列表。
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 。
setAutoMaxBitrate:	设置自适应播放可切换的最高码率。

画中画相关接口

API	描述
isSupportPictureInPicture	是否支持画中画功能。
isSupportSeamlessPictureInPicture	是否支持无缝切换画中画功能（播放器高级版本才支持）。
setAutoPictureInPictureEnabled:	设置是否自动启动画中画（播放器高级版本才支持）。
enterPictureInPicture	进入画中画。
exitPictureInPicture	退出画中画。

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
vodDelegate	设置播放器事件回调对象。
videoProcessDelegate	设置视频渲染回调对象。

TRTC 播片相关接口

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。

接口详情

startVodPlay:

通过 URL 启动播放。
10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

```
- (int)startVodPlay:(NSString *)url;
```

参数说明

参数名	类型	描述
url	NSString	播放地址。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

startVodPlayWithParams:

通过腾讯云 filedID 启动播放。

10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

```
- (int)startVodPlayWithParams:(TXPlayerAuthParams *)params;
```

参数说明

参数名	类型	描述
params	TXPlayerAuthParams	视频 filedID 及信息。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

startPlayDrm:

播放 Drm 加密视频。
10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

> **注意：**
> 播放器高级版本才支持。

```
- (int)startPlayDrm:(TXPlayerDrmBuilder *)drmBuilder;
```

参数说明

参数名	类型	描述
drmBuilder	TXPlayerDrmBuilder	Drm 播放信息。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

stopPlay

停止播放。

```
- (int)stopPlay;
```

返回值

- 0： 停止成功。
- 非 0： 停止失败。

isPlaying

是否正在播放。

```
- (bool)isPlaying;
```

pause

暂停播放。

```
- (void)pause;
```

resume

恢复播放。

```
- (void)resume;
```

seek:

跳转到视频流指定时间点。

```
- (int)seek:(float)time;
```

参数说明

参数名	类型	描述
time	int	视频流时间点，单位为秒。

返回值

- 0：停止成功。
- 非 0：停止失败。

seek:accurateSeek:

跳转到视频流指定时间点。

```
- (void)seek:(float)time accurateSeek:(BOOL)isAccurateSeek;
```

参数说明

参数名	类型	描述
time	float	视频流时间点，单位秒，小数点后精确到3位。
isAccurateSeek	BOOL	是否精准 Seek。

		• YES：表示精确 Seek，必须寻找到当前时间点，这个会比较耗时。 • NO：表示非精准 Seek，也就是寻找前一个I帧。

seekToPdtTime:

跳转到视频流指定时间点。

> **注意：**
> 播放器高级版 11.6 版本开始支持。

```
- (void)seekToPdtTime:(long long)pdtTimeMs;
```

参数说明

参数名	类型	描述
pdtTimeMs	long long	视频流 PDT 时间点，单位毫秒。

currentPlaybackTime

获取当前播放时间点，单位秒。

```
- (float)currentPlaybackTime;
```

duration

获取播放的视频总时长，单位秒。

```
- (float)duration;
```

playableDuration

获取当前可以播放视频时长，单位秒。

```
- (float)playableDuration;
```

width

获取视频宽度。

```
- (int)width;
```

height

获取视频高度。

```
- (int)height;
```

setupVideoWidget:insertIndex:

设置播放器渲染 View，承载视频内容的展示。

```
- (void)setupVideoWidget:(NSView *)view insertIndex:(unsigned int)idx;
```

removeVideoWidget

移除播放器渲染 View。

```
- (void)removeVideoWidget;
```

isAutoPlay

设置点播是否 startPlay 后自动开始播放。默认自动播放。

```
@property BOOL isAutoPlay;
```

setStartTime:

设置播放开始时间，单位秒，需要在启动播放前设置。

```
- (void)setStartTime:(CGFloat)startTime;
```

参数说明

参数名	类型	描述
startTime	CGFloat	视频流时间点，单位秒，小数点后精确到3位。

token

设置加密 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。

```
@property(nonatomic, strong) NSString *token;
```

getEncryptedPlayKey:

获取加固加密播放密钥。

```
+ (NSString *)getEncryptedPlayKey:(NSString *)key;
```

loop

是否循环播放。

```
@property(nonatomic, assign) BOOL loop;
```

addSubtitleSource:name:mimeType:

添加外挂字幕。

> ⚠ **注意：**
> 播放器高级版本才支持。

```
- (void)addSubtitleSource:(NSString *)url name:(NSString *)name mimeType:
(TX_VOD_PLAYER_SUBTITLE_MIME_TYPE)mimeType;
```

参数说明

参数名	类型	描述
url	NSString	字幕地址，支持 Http 链接和本地存储绝对路径。
name	NSString	字幕的名字。如果添加多个字幕，字幕名称请设置为不同的名字，用于区分与其他添加的字幕，否则可能会导致字幕选择错误。
mimeType	TX_VOD_PLAYER_SUBTITLE_MIME_TYPE	字幕类型，仅支持 VVT 和 SRT 格式。具体见 TX_VOD_PLAYER_SUBTITLE_MIME_TYPE 。

getSubtitleTrackInfo

返回字幕轨道信息列表。

> ⚠ **注意：**
> 播放器高级版本才支持。

```
- (NSArray<TXTrackInfo *> *)getSubtitleTrackInfo;
```

返回值
NSArray<TXTrackInfo *> *：字幕轨道信息列表。

getAudioTrackInfo

返回音频轨道信息列表。

> ⚠ **注意：**
> 播放器高级版本才支持。

```
- (NSArray<TXTrackInfo *> *)getAudioTrackInfo;
```

返回值

NSArray<TXTrackInfo *> *：音频轨道信息列表。

selectTrack:

选择轨道。

> **注意：**
> 播放器高级版本才支持。

```
- (void)selectTrack:(NSInteger)trackIndex;
```

参数说明

参数名	类型	描述
trackIndex	NSInteger	轨道 index，通过 -[TXTrackInfo getTrackIndex] 获取。

deselectTrack:

取消选择轨道。

> **注意：**
> 播放器高级版本才支持。

```
- (void)deselectTrack:(NSInteger)trackIndex;
```

参数说明

参数名	类型	描述
trackIndex	NSInteger	轨道 index，通过 -[TXTrackInfo getTrackIndex] 获取。

config

设置播放器配置信息，请参考 TXVodPlayConfig 。

```
@property(nonatomic, copy) TXVodPlayConfig *config;
```

setExtentOptionInfo:

设置播放器业务参数，参数格式为 `<NSString, id>` 。

```
- (void)setExtentOptionInfo:(NSDictionary<NSString *, id> *)extInfo;
```

setSubtitleStyle:

设置字幕样式信息，支持播放前配置，可以支持在播放后对字幕样式进行更新。

> **注意：**
> 播放器高级版本才支持。

```
- (void)setSubtitleStyle:(TXPlayerSubtitleRenderModel *)renderModel;
```

参数说明

参数名	类型	描述
renderModel	TXPlayerSubtitleRenderModel	字幕样式配置参数。

setAudioNormalization:

设置音量均衡，响度范围：-70～0(LUFS)。

> **注意：**
> 播放器高级版本才支持。

```
- (void)setAudioNormalization:(float)value;
```

参数说明

参数名	类型	描述

value	float	可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）
		• 关：AUDIO_NORMALIZATION_OFF
		• 开：
		  ○ AUDIO_NORMALIZATION_STANDARD（标准）
		  ○ AUDIO_NORMALIZATION_LOW（低）
		  ○ AUDIO_NORMALIZATION_HIGH（高）
		可填自定义数值：从低到高，范围-70 – 0 LUFS。

setAutoMaxBitrate:

设置自适应播放可切换的最高码率。

```
- (void)setAutoMaxBitrate:(NSInteger)autoMaxBitrate;
```

enableHWAcceleration

启用或禁用视频硬解码，默认开启硬解码。

```
@property(nonatomic, assign) BOOL enableHWAcceleration;
```

snapshot:

获取当前视频帧图像。

```
- (void)snapshot:(void (^)(UIImage *))snapshotCompletionBlock;
```

参数说明

参数名	类型	描述
snapshotCompletionBlock	void (^)(UIImage *)	截图回调接口类

setMirror:

设置镜像播放。

```
- (void)setMirror:(BOOL)isMirror;
```

setRate:

设置点播的播放速率，默认1.0。

```
- (void)setRate:(float)rate;
```

参数名	类型	描述
rate	float	播放速率 [0.5, 3.0]。

bitrateIndex

返回当前播放的码率索引。

```
- (NSInteger)bitrateIndex;
```

supportedBitrates

当播放地址为 HSL 时，返回支持的码率（清晰度）列表。

```
- (NSArray<TXBitrateItem *> *)supportedBitrates;
```

返回值

NSArray<TXBitrateItem *>： 码率列表。

setBitrateIndex:

设置当前正在播放的码率索引，腾讯云支持多码率 HLS 分片对齐，保证最佳体验。无缝切换清晰度时，清晰度切换可能需要等待一小段时间。

```
- (void)setBitrateIndex:(NSInteger)index;
```

参数说明

参数名	类型	描述
index	NSInteger	码率索引，index == -1，表示开启 HLS 码流自适应。 index > 0 表示手动切换到对应清晰度码率，index 值可以通过接口 -[TXVodPlayer supportedBitrates] 获取。

setRenderMode:

设置播放器图像平铺模式。

```
- (void)setRenderMode:(TX_Enum_Type_RenderMode)renderMode;
```

参数说明

参数名	类型	描述
renderMode	TX_Enum_Type_RenderMode	图像平铺模式，取值有： • RENDER_MODE_FILL_SCREEN：视频画面全屏铺满，将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不留黑边。 • RENDER_MODE_FILL_EDGE：视频画面自适应屏幕，将图像等比例缩放，缩放后的宽和高都不会超过显示区域，居中显示，可能会留有黑边。

setRenderRotation:

设置播放器图像渲染角度。

```
- (void)setRenderRotation:(TX_Enum_Type_HomeOrientation)rotation;
```

参数说明

参数名	类型	描述
rotation	TX_Enum_Type_HomeOrientation	图像渲染角度，取值有： • HOME_ORIENTATION_RIGHT：Home 键在右侧。

		• HOME_ORIENTATION_DOWN：Home 键在下方。 • HOME_ORIENTATION_LEFT：Home 键在左侧。 • HOME_ORIENTATION_UP：Home 键在上方。

setMute:

设置是否静音播放，默认非静音播放。

```
- (void)setMute:(BOOL)bEnable;
```

setAudioPlayoutVolume:

设置音量大小，范围：0 – 150，默认为100。

```
- (void)setAudioPlayoutVolume:(int)volume;
```

vodDelegate

设置播放器事件回调对象。

```
@property(nonatomic, weak) id<TXVodPlayListener> vodDelegate;
```

videoProcessDelegate

设置视频渲染回调对象。

```
@property(nonatomic, weak) id<TXVideoCustomProcessDelegate> videoProcessDelegate;
```

attachTRTC

点播绑定到 TRTC 服务。

```
- (void)attachTRTC:(NSObject *)trtcCloud;
```

detachTRTC

点播解绑 TRTC 服务。

```
- (void)detachTRTC;
```

publishVideo

开始推送视频流。

```
- (void)publishVideo;
```

unpublishVideo

取消推送视频流。

```
- (void)unpublishVideo;
```

publishAudio

开始推送音频流。

```
- (void)publishAudio;
```

unpublishAudio

取消推送音频流。

```
- (void)unpublishAudio;
```

isSupportPictureInPicture

是否支持 Picture In Picture 功能（“画中画”功能）。

```
+ (BOOL)isSupportPictureInPicture;
```

isSupportSeamlessPictureInPicture

是否支持无缝切换 Picture In Picture 功能。需要高级版播放器 SDK。

```
+ (BOOL)isSupportSeamlessPictureInPicture;
```

setAutoPictureInPictureEnabled:

设置是否自动启动 Picture In Picture（自动启动画中画控制开关）。

```
- (void)setAutoPictureInPictureEnabled:(BOOL)enabled;
```

参数说明

参数名	类型	描述
enabled	BOOL	• YES：退后台自动进入画中画。 • NO：不允许自动进入画中画。

enterPictureInPicture

进入画中画功能（此方法需要在 Prepared 后调用）。

```
- (void)enterPictureInPicture;
```

exitPictureInPicture

退出画中画功能。

```
- (void)exitPictureInPicture;
```

TXVodPlayListener

最近更新时间：2025-05-28 11:38:01

TXVodPlayListener API 简介

点播播放器播放事件、网络事件回调监听接口。

回调接口概览

API	描述
onPlayEvent:event:withParam:	播放事件通知。
onNetStatus:withParam:	播放过程中网络状态事件回调。
onPlayer:subtitleData:	字幕数据回调。
onPlayer:pictureInPictureStateDidChange:withParam:	画中画状态回调。
onPlayer:pictureInPictureErrorDidOccur:withParam:	画中画错误信息回调。
onPlayer:airPlayStateDidChange:withParam:	AirPlay 状态回调（仅支持系统播放器）。
onPlayer:airPlayErrorDidOccur:withParam:	AirPlay 错误信息回调（仅支持系统播放器）。

回调接口详情

onPlayEvent:event:withParam:

播放事件通知，包括开始播放、首帧事件、Loading 事件、播放进度、结束播放等事件。

```
- (void)onPlayEvent:(TXVodPlayer *)player event:(int)EvtID withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	当前播放器对象。
EvtID	int	播放器事件，具体见 TXVODEventID 。
param	NSDictionary	播放事件携带的参数，通过(Key,Value)保存，其中 Key 可以参考 TXVodEventDef 中的事件参数。

onNetStatus:withParam:

播放过程中网络状态事件回调。

```
- (void)onNetStatus:(TXVodPlayer *)player withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	当前播放器对象。
param	NSDictionary	播放过程中网络状态参数，格式为：(Key,Value)。具体 Key 见 TXLiveSDKTypeDef 。

onPlayer:subtitleData:

字幕数据回调。

```
- (void)onPlayer:(TXVodPlayer *)player subtitleData:(TXVodSubtitleData
*)subtitleData;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	当前播放器对象。
subtitleData	TXVodSubtitle Data	字幕回调数据。具体见TXVodSubtitleData 。

onPlayer:pictureInPictureStateDidChange:withParam:

画中画状态回调。

```
- (void)onPlayer:(TXVodPlayer *)player pictureInPictureStateDidChange:
(TX_VOD_PLAYER_PIP_STATE)pipState withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	回调的播放器对象。
pipState	TX_VOD_PLAYER_PIP_STATE	画中画控制器状态。
param	NSDictionary	额外参数。

onPlayer:pictureInPictureErrorDidOccur:withParam:

画中画错误信息回调。

```
- (void)onPlayer:(TXVodPlayer *)player pictureInPictureErrorDidOccur:
(TX_VOD_PLAYER_PIP_ERROR_TYPE)errorType withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	回调的播放器对象。
errorType	TX_VOD_PLAYER_PIP_ERROR_TYPE	错误类型。
param	NSDictionary	错误信息。

onPlayer:airPlayStateDidChange:withParam:

AirPlay状态回调（仅支持系统播放器）。

```
- (void)onPlayer:(TXVodPlayer *)player airPlayStateDidChange:
(TX_VOD_PLAYER_AIRPLAY_STATE)airPlayState withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	回调的播放器对象。
airPlayState	TX_VOD_PLAYER_AIRPLAY_STATE	AIRPLAY 状态。
param	NSDictionary	额外参数。

onPlayer:airPlayErrorDidOccur:withParam:

AirPlay 错误信息回调（仅支持系统播放器）。

```
- (void)onPlayer:(TXVodPlayer *)player airPlayErrorDidOccur:
(TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE)errorType withParam:(NSDictionary *)param;
```

参数说明

参数名	类型	描述
player	TXVodPlayer	回调的播放器对象。
errorType	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE	AIRPLAY 状态。

param	NSDictionary	错误信息。

播放配置(Config)
TXPlayerGlobalSetting

最近更新时间：2025-05-23 17:46:52

TXPlayerGlobalSetting API 简介

点播播放器全局配置。

接口概览

API	描述
setCacheFolderPath:	设置播放引擎的 Cache 目录。
cacheFolderPath	获取播放引擎的 Cache 目录。
setMaxCacheSize:	设置播放引擎的最大缓存大小。
maxCacheSize	获取播放引擎的最大缓存大小。
getOptions:	判断播放器特性能力。
setLicenseFlexibleValid:	设置播放器 License 柔性校验。开启后，在播放器首次启动后前3次播放校验将默认通过。
setPlayCGIHosts:	设置腾讯云 PlayCGI 主机域名地址列表。

接口详情

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。

```
+ (void)setCacheFolderPath:(NSString *)cacheFolder;
```

参数说明

参数名	类型	描述
cacheFolder	NSString	缓存目录路径，nil 表示不开启缓存。

cacheFolderPath

获取设置的播放引擎的 Cache 目录。

```
+ (NSString *)cacheFolderPath;
```

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。

```
+ (void)setMaxCacheSize:(NSInteger)maxCacheSizeMB;
```

参数说明

参数名	类型	描述
maxCacheSizeMB	NSInteger	最大缓存大小，单位：MB。

maxCacheSize

获取设置的播放引擎的最大缓存大小。单位 MB。

```
+ (NSInteger)maxCacheSize;
```

getOptions:

判断播放器特性能力。

```
+ (id)getOptions:(NSNumber *)featureId;
```

setLicenseFlexibleValid:

开启播放器 License 柔性校验。开启后，在播放器首次启动后前2次播放校验将默认通过。

```
+ (void)setLicenseFlexibleValid:(BOOL)value;
```

setPlayCGIHosts:

设置腾讯云 PlayCGI 主机域名地址列表，在内置域名请求失败时会启用设置的备用域名。

```
+ (void)setPlayCGIHosts:(NSArray<NSString *> *)hosts;
```

参数说明

参数名	类型	描述
hosts	NSArray<NSString *>	域名地址列表，域名格式为：playvideo.qcloud.com。 发起 PlayCGI 请求时依次使用传入的 hosts 地址，在某个 host 请求失败时自动切换到下个 host 重试请求。

TXVodPlayConfig

最近更新时间：2025-05-23 17:46:52

TXVodPlayConfig API 简介

点播播放器播放配置， 需要在播放前设置。

TX_Enum_MP4EncryptionLevel（MP4加密播放等级）

值	参数名	描述
0	MP4_ENCRYPTION_LEVEL_NONE	不加密。
1	MP4_ENCRYPTION_LEVEL_L1	L1（在线加密）。
2	MP4_ENCRYPTION_LEVEL_L2	L2（本地加密）。

TX_Enum_PlayerType（播放器类型定义）

值	参数名	描述
0	PLAYER_AVPLAYER	系统播放器。
1	PLAYER_THUMB_PLAYER	基于 FFmepg，支持软解，兼容性更好。

TX_Enum_VideoResolution（播放器偏好分辨率）

值	参数名	描述
720 * 1280	VIDEO_RESOLUTION_720X1280	清晰度 720X1280。
1080 * 1920	VIDEO_RESOLUTION_1080X1920	清晰度 1080X1920。
1440 * 2560	VIDEO_RESOLUTION_1440X2560	清晰度 1440X2560。
2160 * 3840	VIDEO_RESOLUTION_2160X3840	清晰度 2160X3840。

TX_Enum_MediaType（媒资类型）

值	参数名	描述
0	MEDIA_TYPE_AUTO	AUTO 类型（默认值，自适应码率播放暂不支持）。

1	MEDIA_TYPE_HLS_VOD	HLS 点播媒资。
2	MEDIA_TYPE_HLS_LIVE	HLS 直播媒资。
3	MEDIA_TYPE_FILE_VOD	MP4等通用文件点播媒资。
4	MEDIA_TYPE_DASH_VOD	DASH 点播媒资。

TX_Enum_Video_Pixel_Format（视频帧输出类型）

值	参数名	描述
0	TX_VIDEO_PIXEL_FORMAT_NONE	无效类型。
1	TX_VIDEO_PIXEL_FORMAT_VideoToolbox	VIDEO TOOL BOX，直接原视频格式输出。
2	TX_VIDEO_PIXEL_FORMAT_RGBA	RGBA 格式（由于苹果不推荐用 RGBA，请使用 BGRA 格式进行替代）。
3	TX_VIDEO_PIXEL_FORMAT_BGRA	BGRA 格式。

接口概览

API	描述
connectRetryCount	设置播放器在异常场景下重连次数。
connectRetryInterval	播放器连接重试间隔，单位秒。最小值为3， 最大值为30。默认值为3。
timeout	设置播放器连接超时时间。
videoFrameFormatType	视频渲染对象回调的视频格式。
keepLastFrameWhenStop	播放停止后是否保留最后一帧画面，默认为 NO。
firstStartPlayBufferTime	首缓需要加载的数据时长，单位 ms， 默认值为 100ms。
nextStartPlayBufferTime	缓冲（缓冲数据不够引起的二次缓冲，或者seek引起的拖动缓冲）最时长，单位ms，默认值为250ms。
playerType	设置播放器类型。
headers	设置 Http header。
enableAccurateSeek	设置是否精确 seek。默认为 YES。
autoRotate	设置播放 MP4 是否自动旋转角度。默认为 YES。
smoothSwitchBitrate	设置是否平滑切换多码率 HLS。默认为 NO。

progressInterval	设置进度回调间隔。默认为500ms。
maxBufferSize	设置播放器最大播放缓冲大小。
maxPreloadSize	设置预加载最大缓冲大小。
overlayKey	设置加密 key。
overlayIv	加密 Iv。
enableRenderProcess	设置播放器是否允许加载渲染后处理服务。默认为 NO。
preferredResolution	设置 HLS 最优的码流进行起播。
encryptedMp4Level	设置 MP4 加密播放。
mediaType	设置播放器播放的媒资类型。
extInfoMap	设置播放器拓展参数。
preferAudioTrack	设置启播时优先使用的音轨。

接口详情

connectRetryCount

设置播放器在异常场景下重连次数。
当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过此函数设置 SDK 重连次数，默认值为3。

```
@property(nonatomic, assign) int connectRetryCount;
```

connectRetryInterval

设置播放器在异常场景下重连间隔时长。
单位秒，最小值为3，最大值为30，默认值为3。

```
@property(nonatomic, assign) int connectRetryInterval;
```

timeout

设置播放器连接超时时间，默认值为10秒。

```
@property(nonatomic, assign) NSTimeInterval timeout;
```

videoFrameFormatType

设置视频渲染对象回调的视频格式。默认值为 TX_VIDEO_PIXEL_FORMAT_NONE。

```
@property(nonatomic, assign) TX_Enum_Video_Pixel_Format videoFrameFormatType;
```

参数说明

参数名	类型	描述
videoFrameFormatType	TX_Enum_Video_Pixel_Format	视频帧回调类型，具体见 TX_Enum_Video_Pixel_Format 。

keepLastFrameWhenStop

播放器 stop 后是否保留最后一帧画面，默认值为NO。

```
@property(nonatomic, assign) BOOL keepLastFrameWhenStop;
```

firstStartPlayBufferTime

首缓需要加载的数据时长，单位 ms。默认值为100ms。

```
@property(nonatomic, assign) int firstStartPlayBufferTime;
```

参数说明

参数名	类型	描述
firstStartPlayBufferTime	int	时长大小。

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。

```
@property(nonatomic, assign) int nextStartPlayBufferTime;
```

playerType

设置播放器类型，默认为腾讯云自研播放器。

```
@property(nonatomic, assign) NSInteger playerType;
```

参数说明

参数名	类型	描述
playerType	int	播放器类型，取值有： • PLAYER_AVPLAYER：iOS 系统播放器。 • PLAYER_THUMB_PLAYER：腾讯云自研播放器，默认值。

headers

自定义配置播放器播放联网过程中携带的 Http header。

```
@property(nonatomic, strong) NSDictionary *headers;
```

参数说明

参数名	类型	描述
headers	NSDictionary <NSString *, NSString *>	自定义的 Http header 内容。

enableAccurateSeek

设置是否精确 seek，默认 true。

```
@property(nonatomic, assign) BOOL enableAccurateSeek;
```

参数说明

参数名	类型	描述
enableAccurateSeek	BOOL	是否精确 seek。

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，默认值为 YES。

```
@property(nonatomic, assign) BOOL autoRotate;
```

参数说明

参数名	类型	描述
autoRotate	BOOL	播放时旋转角度是否自动旋转。

smoothSwitchBitrate

设置是否平滑切换多码率 HLS，默认 NO。

```
@property(nonatomic, assign) BOOL smoothSwitchBitrate;
```

参数说明

参数名	类型	描述

smoothSwitchBitrate	BOOL	是否平滑切换多码率 HLS。

progressInterval

设置进度回调间隔时间，单位毫秒。默认间隔为500毫秒。

```
@property(nonatomic, assign) NSTimeInterval progressInterval;
```

参数说明

参数名	类型	描述
progressInterval	NSTimeInterval	间隔时间，单位毫秒。

maxBufferSize

最大缓存大小，单位 MB。此设置会影响 playableDuration，设置越大，提前缓存的越多。

```
@property(nonatomic, assign) float maxBufferSize;
```

参数说明

参数名	类型	描述
maxBufferSize	float	播放缓冲大小。

maxPreloadSize

设置预加载最大缓冲大小，单位 MB。

```
@property(nonatomic, assign) float maxPreloadSize;
```

参数说明

参数名	类型	描述
maxPreloadSize	float	预加载大小。

overlayKey

设置加密 key。

```
@property(nonatomic, copy) NSString *overlayKey;
```

overlayIv

设置加密 Iv。

```
@property(nonatomic, copy) NSString *overlayIv;
```

encryptedMp4Level

设置 MP4 加密播放。

```
@property(nonatomic, assign) TX_Enum_MP4EncryptionLevel encryptedMp4Level;
```

参数说明

参数名	类型	描述
encryptedMp4Level	TX_Enum_MP4EncryptionLevel	设置 MP4 播放和存储加密等级，从播放器高级版12.2 版本开始支持，具体见 TX_Enum_MP4EncryptionLevel 。

enableRenderProcess

设置 Render 显示后处理标志位，包含超分、VR 播放等功能，使用这些功能需要设置此标志位。默认为 NO。

```
@property(nonatomic, assign) BOOL enableRenderProcess;
```

参数说明

参数名	类型	描述
enableRenderProcess	BOOL	是否允许加载后渲染后处理服务。

preferredResolution

Hls 多 Program 时，根据设定的 preferredResolution 选最优的 Program 进行起播，preferredResolution 为宽高的乘积。配置有效值为 >=-1 的整形数，缺省为-1，播放内核理解为应使用优先级更低的信息进行配置，会从小于该值的program中匹配算数距离最接近的。
优先级为 bitrateIndex > preferredBitrate > preferredResolution。

```
@property(nonatomic, assign) long preferredResolution;
```

参数说明

参数名	类型	描述
preferredResolution	long	视频宽高的乘积（width * height）。

mediaType

设置媒资类型。若自适应码率播放，则须指定具体类型。如自适应播放 HLS 直播资源，须传入 MEDIA_TYPE_HLS_LIVE 类型。

```
@property(nonatomic, assign) TX_Enum_MediaType mediaType;
```

参数说明

参数名	类型	描述
mediaType	TX_Enum_MediaType	媒资类型，具体见 TX_Enum_MediaType 。

extInfoMap

设置播放器特殊配置。

```
@property(nonatomic, strong) NSDictionary *extInfoMap;
```

参数说明

参数名	类型	描述
extInfoMap	NSDictionary	拓展参数。

preferAudioTrack

设置启播时优先加载的音轨名称。仅播放器高级版支持。

```
@property(nonatomic, copy) NSString *preferAudioTrack;
```

参数说明

参数名	类型	描述
preferAudioTrack	NSString	音轨名称。

视频下载(Download) TXVodPreloadManagerDelegate

最近更新时间：2025-05-23 17:46:52

TXVodPreloadManagerDelegate API 简介

预下载 URL 状态回调监听接口。

回调接口概览

API	描述
onStart:fileId:url:param:	视频开始预下载。
onComplete:url:	视频预下载完成。
onError:url:error:	视频预下载出错。

回调接口详情

onStart:fileId:url:param:

预下载结束成功回调。

```
- (void)onStart:(int)taskID fileId:(NSString *)fileId url:(NSString *)url param:
(NSDictionary *)param;
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
fileId	NSString	下载视频的 fileId。URL 方式缓存时，此参数为 nil。
url	NSString	预下载任务 URL。
param	NSDictionary	附加参数。

onComplete:url:

预下载结束成功回调。

```
- (void)onComplete:(int)taskID url:(NSString *)url;
```

参数说明

参数名	类型	描述

taskID	int	预下载任务 ID。
url	NSString	预下载任务 URL。

onError:url:error:

预下载失败回调。

```
- (void)onError:(int)taskID url:(NSString *)url error:(NSError *)error;
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
url	NSString	预下载任务 URL。
error	NSError	错误信息。

TXVodDownloadDataSource

最近更新时间：2025-05-22 16:24:52

TXVodDownloadDataSource 简介

点播下载资源对象。

清晰度 TXVodQuality

参数名	类型	值	描述
TXVodQualityOD	NSInteger	0	原画质。
TXVodQualityFLU	NSInteger	1	流畅。
TXVodQualitySD	NSInteger	2	标清。
TXVodQualityHD	NSInteger	3	高清。
TXVodQualityFHD	NSInteger	4	全高清。
TXVodQuality2K	NSInteger	5	2K。
TXVodQuality4K	NSInteger	6	4K。
TXVodQuality240P	NSInteger	240	流畅240P。
TXVodQuality360P	NSInteger	360	流畅360P。
TXVodQuality480P	NSInteger	480	标清480P。
TXVodQuality540P	NSInteger	540	标清540P。
TXVodQuality720P	NSInteger	720	高清720P 。
TXVodQuality1080p	NSInteger	1080	全高清1080P。

接口详情

auth

点播 fileID 鉴权信息。

```
@property(nonatomic, strong) TXPlayerAuthParams *auth;
```

参数说明

参数名	类型	描述
auth	TXPlayerAuthParams	fileid 信息，具体参见 TXPlayerAuthParams 。

quality

下载清晰度。默认为高清（获取下载信息时，此参数需和下载视频时使用的参数一致）。

```
@property(nonatomic, assign) TXVodQuality quality;
```

参数说明

参数名	类型	描述
quality	TXVodQuality	视频画质 ID ， 具体参见 TXVodQuality 枚举。

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>。

```
@property(nonatomic, copy) NSString *token;
```

templateName

清晰度模板。如果后台转码是自定义模板，请在这里填写模板名。templateName 和 quality 同时设置时，以 templateName 为准。

```
@property(nonatomic, copy) NSString *templateName;
```

fileId

视频文件 ID。

```
@property(nonatomic, copy) NSString *fileId;
```

pSign

签名信息。

```
@property(nonatomic, copy) NSString *pSign;
```

appId

应用 ID。

```
@property(nonatomic, assign) int appId;
```

userName

账户名称，默认“default”。

```
@property(nonatomic, copy) NSString *userName;
```

overlayKey

HLS EXT-X-KEY 加密 key。

```
@property(nonatomic, copy) NSString *overlayKey;
```

overlayIv

HLS EXT-X-KEY 加密 Iv。

```
@property(nonatomic, copy) NSString *overlayKey;
```

TXVodDownloadMediaInfo

最近更新时间：2025-05-26 18:16:22

TXVodDownloadMediaInfo 简介

点播下载媒资描述。

状态枚举

TXVodDownloadMediaInfoState

参数名	类型	值	描述
TXVodDownloadMediaInfoStateInit	NSInteger	0	下载初始态。
TXVodDownloadMediaInfoStateStart	NSInteger	1	下载开始。
TXVodDownloadMediaInfoStateStop	NSInteger	2	下载停止。
TXVodDownloadMediaInfoStateError	NSInteger	3	下载出错。
TXVodDownloadMediaInfoStateFinish	NSInteger	4	下载完成。

接口概览

API	描述
dataSource	用腾讯云视频 fileId 下载时，获取传入的下载源媒资信息。
url	获取实际下载地址。
userName	获取下载账户名称。
duration	获取视频的总时长，单位毫秒。
playableDuration	获取已下载的可播放时长，单位毫秒。
size	获取下载文件总大小，单位：Byte，只针对 fileId 下载源有效。 注意：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法获取。
downloadSize	获取已下载文件大小，单位：Byte，只针对 fileId 下载源有效。
segments	视频分段总数。
downloadSegments	已下载的分段数。

progress	获取当前下载进度。
playPath	获取当前下载资源的播放路径，可传给 TXVodPlayer 播放。
speed	获取下载速度，单位：KByte/秒。（10.9 版本开始支持）
downloadState	获取下载状态。
preferredResolution	获取下载偏好分辨率。
isResourceBroken	判断下载后的视频资源是否损坏，如下载完被删除等情况将返回 true。（11.0 版本开始支持）
isDownloadFinished	判断是否下载完成。

接口详情

dataSource

以腾讯云视频 fileId 下载时获取传入的下载源媒资信息。

```
@property(nonatomic, strong) TXVodDownloadDataSource *dataSource;
```

返回值
下载资源对象信息：TXVodDownloadDataSource

url

获取实际下载地址。

```
@property(nonatomic, copy) NSString *url;
```

userName

获取下载账户名称。默认为 default。

```
@property(nonatomic, copy) NSString *userName;
```

duration

获取视频的总时长，单位毫秒。

```
@property(nonatomic, assign) int duration;
```

playableDuration

获取已下载的可播放时长，单位毫秒。

```
@property(nonatomic, assign) int playableDuration;
```

size

获取下载文件总大小，单位：Byte，只针对腾讯云视频 fileId 下载源有效。
注意：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法获取。

```
@property(nonatomic, assign) long size;
```

downloadSize

获取已下载文件大小，单位：Byte，只针对腾讯云视频 fileId 下载源有效。

```
@property(nonatomic, assign) long downloadSize;
```

segments

视频分段总数。

```
@property(nonatomic, assign) int segments;
```

downloadSegments

视频分段总数。

```
@property(nonatomic, assign) int downloadSegments;
```

progress

获取当前下载进度。

```
@property(nonatomic, assign) float progress;
```

playPath

获取当前下载资源的播放路径，可传给 TXVodPlayer 播放。

```
@property(nonatomic, copy) NSString *playPath;
```

speed

获取下载速度，单位：KByte/秒。（10.9 版本开始支持）

```
@property(nonatomic, assign) int speed;
```

downloadState

获取下载状态。

```
@property(nonatomic, assign) TXVodDownloadMediaInfoState downloadState;
```

返回值
下载状态，具体见 TXVodDownloadMediaInfoState 。

preferredResolution

获取下载偏好分辨率。

```
@property(nonatomic, assign) long preferredResolution;
```

isResourceBroken

判断下载后的视频资源是否损坏，如下载完被删除等情况将返回 true。（11.0 版本开始支持）

```
@property(nonatomic, assign) BOOL isResourceBroken;
```

isDownloadFinished

判断是否下载完成。

```
- (BOOL)isDownloadFinished;
```

TXVodDownloadManager

最近更新时间：2025-05-26 18:16:22

TXDownloadError

下载错误码。

值	参数名	描述
0	TXDownloadSuccess	下载成功。
-5001	TXDownloadAuthFaild	fileId 鉴权失败。
-5003	TXDownloadNoFile	无此清晰度文件。
-5004	TXDownloadFormatError	格式不支持。
-5005	TXDownloadDisconnet	网络断开。
-5006	TXDownloadHlsKeyError	获取 HLS 解密 key 失败。
-5007	TXDownloadPathError	下载目录访问失败。
-5008	TXDownload403Forbidden	鉴权信息不通过，如签名过期或者请求不合法。

TXVodDownloadManager API 简介

点播播放器视频下载接口类。
视频下载支持下载 MP4 和 HLS 视频，对应嵌套 HLS 视频，需要指定偏好清晰度（preferredResolution）

接口概览

API	描述
shareInstance	获取 TXVodDownloadManager 实例对象，单例模式。
setDownloadPath:	设置下载文件的根目录。如不存在，将自动创建。
startDownload:url:	以 URL 方式开始下载。
startDownload:	以 fileId 方式开始下载。
startDownloadUrl:resolution:userName:	以 URL 方式开始下载，可指定偏好清晰度和账户名称。
startDownloadDrm:resolution:userName:	以 Drm 方式开始下载。
stopDownload:	停止下载，-[TXVodDownloadDelegate onDownloadStop:] 回调时停止成功。

deleteDownloadMediaInfo:	删除下载信息。
deleteDownloadFile:	删除下载产生的文件。
getDownloadMediaInfoList	获取所有用户的下载列表信息，耗时接口，请不要在主线程调用。
getDownloadMediaInfo:fileId:qualityId:userName:	获取下载信息。
getDownloadMediaInfo:resolution:userName:	获取下载信息。
encryptHexStringHls:	加密。
headers	设置下载 HTTP 请求头。
delegate	设置下载回调代理对象，下载前必须设好。
supportPrivateEncryptMode	设置是否支持私有加密模式(配置为系统播放器请设置为 NO，自研播放器设置为 YES)。默认设置为 YES。

接口详情

shareInstance

获取 TXVodDownloadManager 实例对象，单例模式。

```
+ (TXVodDownloadManager *)shareInstance;
```

setDownloadPath:

设置下载文件的根目录。此处设置的下载目录优先以 `+[TXPlayerGlobalSetting setCacheFolderPath:]` 设置为准。

```
- (void)setDownloadPath:(NSString *)path;
```

startDownload:

以腾讯云视频 fileId 方式开始下载

```
- (TXVodDownloadMediaInfo *)startDownload:(TXVodDownloadDataSource *)source;
```

参数说明

参数名	类型	描述
source	TXVodDownloadDataSource	下载资源对象。

startDownload:url:

以 URL 方式开始下载。

```
- (TXVodDownloadMediaInfo *)startDownload:(NSString *)username url:(NSString *)url;
```

参数说明

参数名	类型	描述
url	NSString	下载地址，必填。
username	NSString	账户名称，可选参数，不传默认为"default"。

返回值

视频信息和下载状态，具体见 TXVodDownloadMediaInfo 。

startDownloadUrl:resolution:userName:

以 URL 方式开始下载。

```
- (TXVodDownloadMediaInfo *)startDownloadUrl:(NSString *)url resolution:
(long)resolution userName:(NSString *)username;
```

参数说明

参数名	类型	描述
url	NSString	下载地址，必填。
resolution	long	偏好清晰度，多清晰度 url 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600 = 1280 * 720)，单清晰度传入-1。
username	NSString	账户名称，可选参数，不传默认为"default"。

返回值

视频信息和下载状态，具体见 TXVodDownloadMediaInfo 。

startDownloadDrm:resolution:userName:

下载文件 DRM 视频。

```
- (TXVodDownloadMediaInfo *)startDownloadDrm:(TXPlayerDrmBuilder *)drmBuilder
resolution:(long)resolution userName:(NSString *)username;
```

参数说明

参数名	类型	描述
drmBuilder	TXPlayerDrmBuilder	DRM 下载对象，参考 TXPlayerDrmBuilder 。
resolution	long	偏好清晰度，多清晰度 URL 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600=1280*720)，单清晰度传入-1。

username	NSString	账户名称，可选参数，不传默认为"default"。

返回值

视频信息和下载状态，具体见 TXVodDownloadMediaInfo 。

stopDownload:

停止下载， `-[id<TXVodDownloadDelegate> onDownloadStop:]` 回调时停止成功。

```
- (void)stopDownload:(TXVodDownloadMediaInfo *)media;
```

参数说明

参数名	类型	描述
downloadMediaInfo	TXVodDownloadMediaInfo	视频下载信息。

deleteDownloadMediaInfo:

删除下载信息。

```
- (BOOL)deleteDownloadMediaInfo:(TXVodDownloadMediaInfo *)downloadMediaInfo;
```

参数说明

参数名	类型	描述
downloadMediaInfo	TXVodDownloadMediaInfo	视频下载信息。

返回值

是否删除成功。YES：删除成功；NO：删除失败。文件正在下载将无法删除。

getDownloadMediaInfoList

获取所有用户的下载列表信息。

```
- (NSArray<TXVodDownloadMediaInfo *> *)getDownloadMediaInfoList;
```

返回值

视频下载信息列表：NSArray<TXVodDownloadMediaInfo>。

getDownloadMediaInfo:fileId:qualityId:userName:

获取下载信息。

调用此接口要确保之前通过 `startDownload:` 、 `startDownload:url:` 或 `startDownloadDrm:resolution:userName:` 创建过下载任务参数。

```
- (TXVodDownloadMediaInfo *)getDownloadMediaInfo:(int)appId fileId:(NSString
```

```
*)fileId qualityId:(int)qualityId userName:(NSString *)userName;
```

参数说明

参数名	类型	描述
appId	int	腾讯云点播应用 appId。
fileId	NSString	腾讯云点播视频 fileId。
qualityId	int	视频画质 Id ， 具体参考 TXVodQuality 常量。
userName	NSString	账户名称，须与下载时传入的账户名称一致，若下载时未传入，这里传入空字符""。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

getDownloadMediaInfo:resolution:userName:

获取下载信息。

```
- (TXVodDownloadMediaInfo *)getDownloadMediaInfo:(NSString *)url resolution:
(long)preferredResolution userName:(NSString *)userName;
```

参数说明

参数名	类型	描述
url	NSString	下载地址，必选参数，否则下载失败。
preferredResolution	long	下载偏好清晰度，多清晰度 URL 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600=1280*720)，单清晰度传入-1。
userName	userName	账户名称，可选参数，不传默认为"default"。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

encryptHexStringHls:

加密。

```
+ (NSString *)encryptHexStringHls:(NSString *)originHexStr
```

参数名	类型	描述
originHexStr	NSString	初始十六进制字符串。

返回值

加密串：NSString

headers

设置 http 请求头。

```
@property(nonatomic, strong) NSDictionary *headers;
```

参数名	类型	描述
headers	NSDictionary	http 请求头。

delegate

设置下载任务回调对象。

```
@property(nonatomic, weak) id<TXVodDownloadDelegate> delegate;
```

参数名	类型	描述
delegate	id<TXVodDownloadDelegate>	下载任务回调对象。

supportPrivateEncryptMode

是否支持私有加密模式（配置为系统播放器时设置为 NO，自研播放器设置为 YES）。默认为 YES。

```
@property(nonatomic, assign) BOOL supportPrivateEncryptMode;
```

参数名	类型	描述
supportPrivateEncryptMode	BOOL	是否支持私有加密模式。

TXVodDownloadDelegate

最近更新时间：2025-05-26 18:16:22

TXVodDownloadDelegate API 简介

点播播放器下载回调监听接口。

回调接口概览

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:errorCode:errorMsg:	下载过程中遇到错误。

回调接口详情

onDownloadStart:

下载开始。

```
- (void)onDownloadStart:(TXVodDownloadMediaInfo *)mediaInfo;
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadProgress:

下载进度更新。

```
- (void)onDownloadProgress:(TXVodDownloadMediaInfo *)mediaInfo;
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadStop:

下载停止。调 `-[TXVodDownloadManager stopDownload:]` 方法会收到此回调。

```
- (void)onDownloadStop:(TXVodDownloadMediaInfo *)mediaInfo;
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadFinish:

下载结束。

```
- (void)onDownloadFinish:(TXVodDownloadMediaInfo *)mediaInfo;
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadError:errorCode:errorMsg:

下载过程中遇到错误。

```
- (void)onDownloadError:(TXVodDownloadMediaInfo *)mediaInfo errorCode:
(TXDownloadError)code errorMsg:(NSString *)msg;
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。
code	TXDownloadError	下载错误码。
msg	NSString	下载错误信息。

TXVodPreloadManagerDelegate

最近更新时间：2025-05-26 18:16:22

TXVodPreloadManagerDelegate API 简介

腾讯云视频 fileId 和 URL 预下载状态回调监听接口。

回调接口概览

API	描述
onStart:fileId:url:param:	视频预下载开始。 对于 fileId 预下载，在换链成功后，启动预下载前回调。
onComplete:url:	视频预下载完成。
onError:url:error:	视频预下载出错。

回调接口详情

onStart:fileId:url:param:

预下载结束成功回调。

```
- (void)onStart:(int)taskID fileId:(NSString *)fileId url:(NSString *)url param:
(NSDictionary *)param;
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
fileId	NSString	预下载的视频 fileId。
url	NSString	预下载任务 URL，为换链后的视频 URL，可用于后续播放。
param	NSDictionary	预下载携带的额外信息。

onComplete:url:

预下载结束成功回调。

```
- (void)onComplete:(int)taskID url:(NSString *)url;
```

参数说明

参数名	类型	描述

taskID	int	预下载任务 ID。
url	String	预下载任务 URL。

onError:url:error:

预下载失败回调。

```
- (void)onError:(int)taskID url:(NSString *)url error:(NSError *)error;
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
url	NSString	预下载任务 URL。
error	NSError	错误信息。

TXVodPreloadManager

最近更新时间：2025-05-26 18:16:22

TXVodPreloadManager API 简介

点播播放器预下载接口类。
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。

接口概览

API	描述
sharedManager	获取 TXVodPreloadManager 实例对象，单例模式。
startPreload:preloadSize:preferredResolution:delegate:	通过 URL 启动预下载。
startPreloadWithModel:preloadSize:preferredResolution:delegate:	通过 fileId 或 URL 启动预下载，推荐优先使用此接口。
stopPreload:	停止预下载。

接口详情

sharedManager

获取 TXVodPreloadManager 实例对象，单例模式。

```
+ (instancetype)sharedManager;
```

startPreload:preloadSize:preferredResolution:delegate:

通过 URL 启动预下载。
注意：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:]`，这个设置是全局配置需和播放器保持一致，否则会造成播放缓存失效。

```
- (int)startPreload:(NSString *)requestURL preloadSize:(float)preloadSizeMB
preferredResolution:(long)preferredResolution delegate:
(id<TXVodPreloadManagerDelegate>)delegate;
```

参数说明

参数名	类型	描述
requestURL	NSString	预下载的视频 URL。
preloadSizeMB	float	预下载的大小，单位：MB。

preferredResolution	long	期望下载的分辨率，视频宽高的乘积（width * height）。 不支持多分辨率或不需指定时，传-1。
delegate	id<TXVodPreloadManagerDelegate>	预下载监听状态回调对象。

返回值

任务 ID，可用这个任务 ID 停止预下载 `-[TXVodPreloadManager stopPreload:]`。
如果返回-1， 表示此任务 ID 无效。

startPreloadWithModel:preloadSize:preferredResolution:delegate:

启动预下载，支持通过腾讯云 fileId 和视频 URL 预下载。

- 如果 `-[TXPlayerAuthParams url]` 不为空，则优先启动视频 URL 预下载，此时支持在主线调用。
- 如果 `-[TXPlayerAuthParams fileId]` 不为空，则启动视频 fileId 预下载，此时不支持在主线调用。

注意：

1. 预下载是耗时操作，请不要在主线程调用，在主线程调用将会抛出非法调用异常。
2. 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:]`，这个设置是全局配置需和播放器保持一致，否则会造成播放缓存失效。

```
- (int)startPreloadWithModel:(TXPlayerAuthParams *)params preloadSize:
(float)preloadSizeMB preferredResolution:(long)preferredResolution delegate:
(id<TXVodPreloadManagerDelegate>)delegate;
```

参数说明

参数名	类型	描述
params	TXPlayerAuthParams	预下载信息。 可通过 `-[TXPlayerAuthParams headers]` 设置预下载 http 请求头，通过 `-[TXPlayerAuthParams preferAudioTrack]` 设置预下载的音轨名称。
preloadSizeMB	float	预下载的大小，单位：MB。
preferredResolution	long	期望下载的分辨率，视频宽高的乘积（width * height）。 不支持多分辨率或不需指定时，传-1。
delegate	id<TXVodPreloadManagerDelegate>	预下载监听状态回调对象。

返回值

任务 ID，可用这个任务 ID 停止预下载 `-[TXVodPreloadManager stopPreload:]`。
如果返回-1， 表示此任务 ID 无效。

stopPreload:

停止预下载。

```
- (void)stopPreload:(int)taskID;
```

参数说明

参数名	类型	描述
taskID	int	任务 ID。ID 从 startPreload 接口返回值得到。

类型定义

TXVodSDKEventDef

最近更新时间：2025-07-01 14:53:11

TXVodSDKEventDef API 简介

点播播放器用到的常量类。

视频分辨率

TX_Enum_Type_VideoResolution

值	参数名	描述
0	VIDEO_RESOLUTION_TYPE_360_640	建议码率 800kbps。
1	VIDEO_RESOLUTION_TYPE_540_960	建议码率 1200kbps。
2	VIDEO_RESOLUTION_TYPE_720_1280	建议码率 1800kbps。
30	VIDEO_RESOLUTION_TYPE_1080_1920	建议码率 3000kbps。

画面质量档位

TX_Enum_Type_VideoQuality

值	参数名	描述
1	VIDEO_QUALITY_STANDARD_DEFINITION	标清：采用 360 × 640 的分辨率。
2	VIDEO_QUALITY_HIGH_DEFINITION	高清：采用 540 × 960 的分辨率。
3	VIDEO_QUALITY_SUPER_DEFINITION	超清：采用 720 × 1280 的分辨率。
4	VIDEO_QUALITY_LINKMIC_MAIN_PUBLISHER	连麦场景下的大主播使用。
5	VIDEO_QUALITY_LINKMIC_SUB_PUBLISHER	连麦场景下的小主播（连麦的观众）使用。
7	VIDEO_QUALITY_ULTRA_DEFINITION	蓝光：采用 1080 × 1920 的分辨率。

画面旋转方向

TX_Enum_Type_HomeOrientation

值	参数名	描述
0	HOME_ORIENTATION_RIGHT	HOME 键在右边，横屏模式。
1	HOME_ORIENTATION_DOWN	HOME 键在下面，手机直播中最常见的竖屏直播模式。
2	HOME_ORIENTATION_LEFT	HOME 键在左边，横屏模式。
3	HOME_ORIENTATION_UP	HOME 键在上边，竖屏直播。

画面填充模式

TX_Enum_Type_RenderMode

值	参数名	描述
0	RENDER_MODE_FILL_SCREEN	视频画面全屏铺满。
1	RENDER_MODE_FILL_EDGE	视频画面自适应屏幕。

播放事件列表

值	参数名	描述
2002	VOD_PLAY_EVT_HIT_CACHE	启播命中缓存。
2003	VOD_PLAY_EVT_RCV_FIRST_I_FRAME	渲染视频首帧事件。
2004	VOD_PLAY_EVT_PLAY_BEGIN	视频播放开始。
2005	VOD_PLAY_EVT_PLAY_PROGRESS	视频播放进度。
2006	VOD_PLAY_EVT_PLAY_END	视频播放结束。
6001	VOD_PLAY_EVT_LOOP_ONCE_COMPLETE	循环一轮播放结束。
2007	VOD_PLAY_EVT_PLAY_LOADING	视频播放 Loading。
2008	VOD_PLAY_EVT_START_VIDEO_DECODER	解码器启动。
2009	VOD_PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2010	VOD_PLAY_EVT_GET_PLAYINFO_SUCC	获取点播文件信息成功。

2011	VOD_PLAY_EVT_CHANGE_ROTATION	视频旋转信息。
2013	VOD_PLAY_EVT_VOD_PLAY_PREPARED	视频加载完毕。
2014	VOD_PLAY_EVT_VOD_LOADING_END	视频缓冲结束。
2017	VOD_PLAY_EVT_VOD_PLAY_FIRST_VIDEO_PACKET	收到首帧数据（12.0 版本开始支持）。
2019	VOD_PLAY_EVT_VOD_PLAY_SEEK_COMPLETE	Seek 完成（10.3版本开始支持）。
2020	VOD_PLAY_EVT_SELECT_TRACK_COMPLETE	切换轨道完成。
2026	VOD_PLAY_EVT_RCV_FIRST_AUDIO_FRAME	音频首次播放。
2103	PLAY_WARNING_RECONNECT	网络断连，已启动自动重连。
2030	VOD_PLAY_EVT_VIDEO_SEI	视频 SEI 信息事件。
2031	VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK	HEVC 降级播放
-2301	VOD_PLAY_ERR_NET_DISCONNECT	网络断连，且经多次重连抢救无效。
-2303	VOD_PLAY_ERR_FILE_NOT_FOUND	文件不存在。
-2304	PLAY_ERR_HEVC_DECODE_FAIL	HEVC 解码失败。
-2305	VOD_PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
-2306	VOD_PLAY_ERR_GET_PLAYINFO_FAIL	获取点播文件信息失败。
2106	PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-5	VOD_PLAY_ERR_LICENCE_CHECK_FAIL	License 不合法，播放失败。 注：在 startVodPlay 之前，需要通过 TXLiveBase#setLicence 设置 License 后方可成功播放，否则将播放失败（黑屏），全局仅设置一次即可。直播 License、短视频 License 和播放器 License 均可使用，若您暂未获取上述 License，可单击 播放器 License 进行申请，正式版 License 需购买。
-6004	VOD_PLAY_ERR_SYSTEM_PLA	系统播放器播放错误。

	Y_FAIL	
-6005	VOD_PLAY_ERR_DEMUXER_TIMEOUT	解封装超时。
-6006	VOD_PLAY_ERR_DECODE_VIDEO_FAIL	视频解码错误，视频格式不支持。
-6007	VOD_PLAY_ERR_DECODE_AUDIO_FAIL	音频解码错误，音频格式不支持。
-6008	VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL	字幕解码错误。
-6009	VOD_PLAY_ERR_RENDER_FAIL	视频渲染错误。
-6010	VOD_PLAY_ERR_PROCESS_VIDEO_FAIL	视频后处理错误。
-6011	VOD_PLAY_ERR_DOWNLOAD_FAIL	视频下载出错。
-6101	VOD_PLAY_ERR_DRM	DRM 播放失败

画中画控制器状态

值	参数名	描述
0	TX_VOD_PLAYER_PIP_STATE_UNDEFINED	无效状态。
1	TX_VOD_PLAYER_PIP_STATE_WILL_START	画中画即将开始。
2	TX_VOD_PLAYER_PIP_STATE_DID_START	画中画已经开始。
3	TX_VOD_PLAYER_PIP_STATE_WILL_STOP	画中画即将结束。
4	TX_VOD_PLAYER_PIP_STATE_RESTORE_UI	重置 UI。

画中画错误类型

值	参数名	描述
0	TX_VOD_PLAYER_PIP_ERROR_TYPE_NONE	无错误。
1	TX_VOD_PLAYER_PIP_ERROR_TYPE_DEVICE_NOT_SUPPORT	设备或系统版本不支持（iPad iOS9+ 才支持 PIP）。

2	TX_VOD_PLAYER_PIP_ERROR_TYPE_PLAYER_NOT_SUPPORT	播放器不支持。
3	TX_VOD_PLAYER_PIP_ERROR_TYPE_VIDEO_NOT_SUPPORT	视频不支持。
4	TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_IS_NOT_POSSIBLE	PIP 控制器不可用。
5	TX_VOD_PLAYER_PIP_ERROR_TYPE_ERROR_FROM_SYSTEM	PIP 控制器报错。
10	TX_VOD_PLAYER_PIP_ERROR_TYPE_PLAYER_NOT_EXIST	播放器对象不存在。
11	TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_IS_RUNNING	PIP 功能已经运行。
12	TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_NOT_RUNNING	PIP 功能没有启动。
13	TX_VOD_PLAYER_PIP_ERROR_TYPE_PIP_START_TIMEOUT	PIP 启动超时。
20	TX_VOD_PLAYER_PIP_ERROR_TYPE_SEAMLESS_PIP_ERROR	无缝 PIP 功能启动失败。
21	TX_VOD_PLAYER_PIP_ERROR_TYPE_SEAMLESS_PIP_NOT_SUPPORT	不支持无缝切换 PIP。
22	TX_VOD_PLAYER_PIP_ERROR_TYPE_SEAMLESS_PIP_IS_RUNNING	无缝 PIP 功能已经运行。

AirPlay 状态(仅支持系统播放器)

值	参数名	描述
0	TX_VOD_PLAYER_AIRPLAY_STATE_NOT_RUNNING	未运行。
1	TX_VOD_PLAYER_AIRPLAY_STATE_DID_RUNNING	运行中。

AirPlay 错误类型(仅支持系统播放器)

值	参数名	描述
0	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE_NONE	无错误。

1	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE_PLAYER_NOT_SUPPORT	播放器不支持。
2	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE_VIDEO_NOT_SUPPORT	视频不支持。
10	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE_PLAYER_INVALID	播放器对象不可用。
11	TX_VOD_PLAYER_AIRPLAY_ERROR_TYPE_PLAYER_STATE	播放器状态错误。

外挂字幕类型

TX_VOD_PLAYER_SUBTITLE_MIME_TYPE

值	参数名	描述
0	TX_VOD_PLAYER_MIMETYPE_TEXT_SRT	外挂字幕 SRT 格式。
1	TX_VOD_PLAYER_MIMETYPE_TEXT_VTT	外挂字幕 VTT 格式。

播放事件参数

值	参数名	描述
"CPU_USAGE"	NET_STATUS_CPU_USAGE	当前瞬时 CPU 使用率。
"VIDEO_WIDTH"	NET_STATUS_VIDEO_WIDTH	视频分辨率-宽。
"VIDEO_HEIGHT"	NET_STATUS_VIDEO_HEIGHT	视频分辨率-高。
"NET_SPEED"	NET_STATUS_NET_SPEED	当前的网络数据接收速度，单位：KBps。
"VIDEO_FPS"	NET_STATUS_VIDEO_FPS	当前流媒体的视频帧率。
"VIDEO_BITRATE"	NET_STATUS_VIDEO_BITRATE	当前流媒体的视频码率，单位 bps。
"AUDIO_BITRATE";	NET_STATUS_AUDIO_BITRATE	当前流媒体的音频码率，单位 bps。
"VIDEO_CACHE"	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，单位：KBps。

"SERVER_IP"	NET_STATUS_SERVER_IP	连接的服务器 IP。
"EVT_UTC_TIME"	VOD_PLAY_EVENT_UTC_TIME	UTC 时间。
"EVT_BLOCK_DURATION"	VOD_PLAY_EVENT_BLOCK_DURATION	卡顿时间（毫秒）。
"EVT_ERROR_CODE"	VOD_PLAY_EVT_ERROR_CODE	播放器错误码。
"EVT_TIME"	VOD_PLAY_EVENT_TIME	事件发生时间。
"EVT_MSG"	VOD_PLAY_EVENT_MSG	事件说明。
"EVT_PARAM1"	VOD_PLAY_EVENT_PARAM1	事件参数1。
"EVT_PARAM2"	VOD_PLAY_EVENT_PARAM2	事件参数2。
"EVT_GET_MSG"	VOD_PLAY_EVENT_GET_MSG	消息内容。收到 PLAY_EVT_GET_MESSAGE 事件时，通过该字段获取消息内容。
"EVT_PLAY_COVER_URL"	VOD_PLAY_EVENT_PLAY_COVER_URL	视频封面。
"EVT_PLAY_URL"	VOD_PLAY_EVENT_PLAY_URL	视频地址。
"EVT_PLAY_NAME"	VOD_PLAY_EVENT_PLAY_NAME	视频名称。
"EVT_PLAY_DESCRIPTION"	VOD_PLAY_EVENT_PLAY_DESCRIPTION	视频简介。
"EVT_PLAY_PROGRESS"	VOD_PLAY_EVENT_PLAY_PROGRESS	播放进度。
"EVT_PLAY_DURATION"	VOD_PLAY_EVENT_PLAY_DURATION	视频总时长。
"EVT_PLAYABLE_DURATION"	VOD_PLAY_EVENT_PLAYABLE_DURATION	点播可播放时长。
"EVT_IMAGESPRIT_WEBVTTURL"	VOD_PLAY_EVENT_IMAGESPRIT_WEBVTTURL	雪碧图 web vtt 描述文件下载 URL。
"EVT_IMAGESPRIT_IMAGEU	VOD_PLAY_EVENT_IMAGESPRIT_IMAGEURL_LI	雪碧图图片下载 URL。

RL_LIST"	ST	
"VOD_KEY_VIDEO_CODEC_TYPE"	VOD_KEY_VIDEO_CODEC_TYPE	视频编码类型。
"VOD_KEY_BACKUP_URL_MEDIA_TYPE"	VOD_KEY_BACKUP_URL_MEDIA_TYPE	备选播放资源（VOD_KEY_BACKUP_URL）对应的类型（12.0 版本新增）。
"PARAM_MODULE_TYPE"	PLAYER_OPTION_PARAM_MODULE_TYPE	module 类型参数。
"PARAM_MODULE_CONFIG"	PLAYER_OPTION_PARAM_MODULE_CONFIG	module 配置。
"ENABLE_SENSOR"	PLAYER_OPTION_PARAM_MODULE_VR_ENABLE_SENSOR	是否开启传感器，默认 true。
"FOV"	PLAYER_OPTION_PARAM_MODULE_VR_FOV	视场角，默认65.0f度，限制范围30.0f度到110.0f度。
"ANGLE_X"	PLAYER_OPTION_PARAM_MODULE_VR_ANGLE_X	水平旋转角度，正值右转，负值左转。0° 表示正前方，取值范围-180° 到180° 。
"ANGLE_Y"	PLAYER_OPTION_PARAM_MODULE_VR_ANGLE_Y	垂直旋转角度，正值上转，负值下转。0° 表示水平视 ，取值范围-85° 到85° 。
"ANGLE_RATE"	PLAYER_OPTION_PARAM_MODULE_VR_ANGLE_RATE	手势滑动距离与角度比例，比例越大灵敏度越高，默认值为1/3.0f。
"ANGLE_SLOPE_THRESHOLD"	PLAYER_OPTION_PARAM_MODULE_VR_ANGLE_SLOPE_THRESHOLD	旋转 XY 角度斜率阈值，默认值为0.5f，阀值范围内只选取长边旋转。
"EVT_KEY_FRAME_CONTENT_LIST"	VOD_PLAY_EVENT_KEY_FRAME_CONTENT_LIST	视频关键帧描述信息。
"EVT_KEY_FRAME_TIME_LIST"	VOD_PLAY_EVENT_KEY_FRAME_TIME_LIST	关键帧时间。
"EVT_PLAY_PDT_TIME_MS"	VOD_PLAY_EVENT_PLAY_PDT_TIME_MS	播放 PDT 时间（毫秒）。
"VOD_KEY_CUSTOM_DATA"	VOD_KEY_CUSTOM_DATA	自定义透传上报字段 Key（11.7 版本新增）。

"EVT_KEY_VIDEO_ROTATION"	VOD_PLAY_EVENT_KEY_VIDEO_ROTATION	MP4视频旋转角度。
"EVT_KEY_SELECT_TRACK_INDEX"	EVT_KEY_SELECT_TRACK_INDEX	外挂字幕 Event 参数返回：切换的媒体轨道 index。
"EVT_KEY_SELECT_TRACK_ERROR_CODE"	EVT_KEY_SELECT_TRACK_ERROR_CODE	外挂字幕 Event 参数返回：切换媒体轨道的返回错误码。
"VOD_PLAY_BUFFERING_LOADING_TYPE"	VOD_PLAY_BUFFERING_LOADING_TYPE	播放器 Loading 的 Type 类型。
"MONET_AC_DO_ROTATE"	PLAYER_OPTION_PARAM_MODULE_VR_DO_ROTATE	VR 旋转角度。
"EVT_KEY_WATER_MARK_TEXT"	EVT_KEY_WATER_MARK_TEXT	幽灵水印文本（11.5版本开始支持）。
"EVT_KEY_SEI_TYPE"	EVT_KEY_SEI_TYPE	视频 SEI 类型。
"EVT_KEY_SEI_SIZE"	EVT_KEY_SEI_SIZE	视频 SEI 数据 buffer 大小。
"EVT_KEY_SEI_DATA"	EVT_KEY_SEI_DATA	视频 SEI 数据 buffer。

播放器媒资类型

值	参数名	描述
0	MEDIA_TYPE_AUTO	auto 类型。
1	MEDIA_TYPE_HLS_VOD	自适应码率播放 HLS 点播媒资。
2	MEDIA_TYPE_HLS_LIVE	自适应码率播放 HLS 直播媒资。
3	MEDIA_TYPE_FILE_VOD	MP4等通用文件点播媒资。
4	MEDIA_TYPE_DASH_VOD	DASH 点播媒资。

MP4 加密等级

值	参数名	描述
0	MP4_ENCRYPTION_LEVEL_NONE	MP4 不加密播放。

1	MP4_ENCRYPTION_LEVEL_L1	L1，MP4 在线加密播放。
2	MP4_ENCRYPTION_LEVEL_L2	L2，MP4 本地加密播放。

module 类型

TX_VOD_PLAYER_OPTION_PARAM_MODULE_TYPE

值	参数名	描述
0	PLAYER_OPTION_PARAM_MODULE_TYPE_NONE	空类型，即关闭超分和 VR 等。
1	PLAYER_OPTION_PARAM_MODULE_TYPE_SR	超分类型。
11	PLAYER_OPTION_PARAM_MODULE_TYPE_VR_PANORAMA	VR 全景模型，单目。
12	PLAYER_OPTION_PARAM_MODULE_TYPE_VR_BINOCULAR	VR 全景模型，双目。

未分类变量

值	参数名	描述
0	PLAYER_SYSTEM_MEDIA_PLAYER	系统播放器。
1	PLAYER_THUMB_PLAYER	自研播放器，支持软解，兼容性更好。
-1	INDEX_AUTO	自适应码率 index 标识。
"450"	PLAYER_OPTION_KEY_SUBTITLE_OUTPUT_TYPE	外挂字幕输出类型配置 Key。
"VOD_KEY_BACKUP_URL"	VOD_KEY_BACKUP_URL	降级播放备选 URL Key。
"text/x-subrip"	VOD_PLAY_MIMETYPE_TEXT_SRT	外挂字幕 SRT 格式。
"text/vtt"	VOD_PLAY_MIMETYPE_TEXT_VTT	外挂字幕 VTT 格式。

TXPlayerSubtitleRenderModel

最近更新时间：2025-05-22 16:24:52

TXPlayerSubtitleRenderModel 简介

点播播放器播字幕样式渲染参数。

字段详情

canvasWidth

canvasWidth 和 canvasHeight 是字幕渲染画布的大小， canvasWidth 和 canvasHeight 的比例必须和视频的宽高比一致，否则渲染出的字体会变形。 如果不设置，播放器会取当前视频的大小作为渲染画布的大小。

```
@property(nonatomic, assign) int canvasWidth;
```

canvasHeight

canvasWidth 和 canvasHeight 是字幕渲染画布的大小， canvasWidth 和 canvasHeight 的比例必须和视频的宽高比一致，否则渲染出的字体会变形。 如果不设置，播放器会取当前视频的大小作为渲染画布的大小。

```
@property(nonatomic, assign) int canvasHeight;
```

familyName

Font family name。iOS 默认为"Helvetica"，字符串不为空则认为已设置，为空则认为未设置。

```
@property(nonatomic, copy) NSString *familyName;
```

fontSize

字体大小，如果设置了 fontSize，则必须设置 canvasWidth 和 canvasHeight，否则内部不知道以什么大小为参考来渲染字体， 如果不设置 fontSize，内部会使用默认的字体大小。

```
@property(nonatomic, assign) float fontSize;
```

fontScale

字体缩放比例， VTT、CSS 专用。使用 fontScale 乘以 VTT 设定的 font-size: em 值再适应视频宽。如果设置了 fontScale，paramFlags 须置为 TP_SUBTITLE_PARAM_FLAG_FONT_SCALE，最终字体像素为 fontScale × VTT em × 16 × canvas width(video width) / default width(491)，
fontScale 默认1.0，视频宽491像素时，中文字号设定为16像素大小，将 VTT 文件内字体大小设定为1em(font-size: 1.00em;)。

```
@property(nonatomic, assign) float fontScale;
```

fontColor

字体颜色，ARGB 格式。如果不设置，默认为白色不透明(0xFFFFFFFF)。

```
@property(nonatomic, assign) uint32_t fontColor;
```

outlineWidth

描边宽度，如果不设置，内部会使用默认的描边宽度。

```
@property(nonatomic, assign) float outlineWidth;
```

outlineColor

描边颜色，ARGB 格式。如果不设置，默认为黑色不透明(0xFF000000)。

```
@property(nonatomic, assign) uint32_t outlineColor;
```

isBondFontStyle

是否是粗体，默认值为正常字体。

```
@property(nonatomic, assign) BOOL isBondFontStyle;
```

lineSpace

行距，如果设置了 lineSpace，则必须设置 canvasWidth 和 canvasHeight，如果不设置，内部会使用默认的行距。

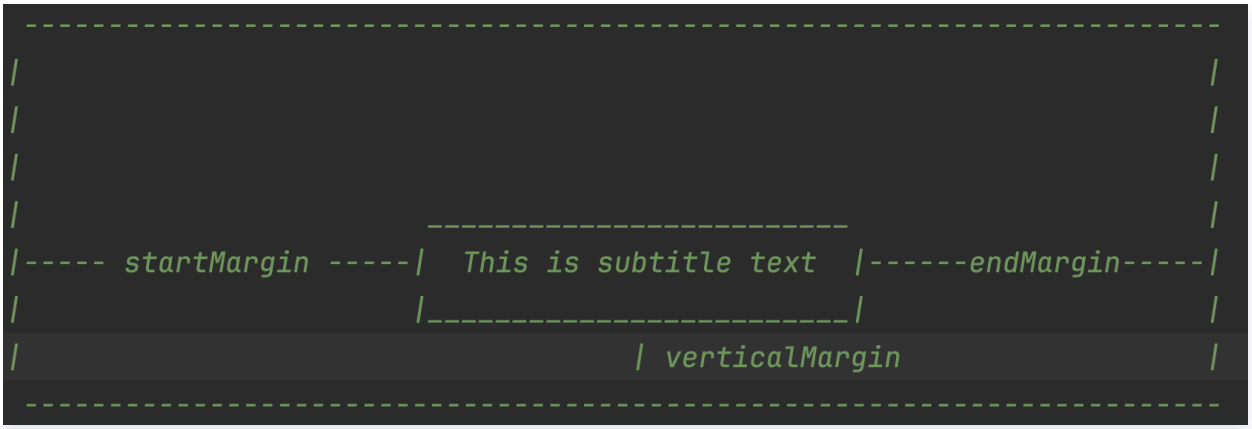
```
@property(nonatomic, assign) float lineSpace;
```

startMargin

startMargin、endMargin 和 verticalMargin 定义字幕的绘制区域，如果不设置，则使用字幕文件中的设置，如果字幕文件也没有定义，则使用默认的。

> **注意：**
> 一旦设置了 startMargin、endMargin 和 yMargin，而字幕文件也定义了这几个参数的一个或多个，则会覆盖字幕文件中相应的参数。

下面示意图描绘了水平书写方向下这几个参数的意义，请借助每个参数的注释来理解。



沿着字幕文本方向的边距，根据不同的书写方向意义不同。 startMargin 是一个比例值，取值范围[0, 1]，即相对于视频画面大小的比例。

- 对于水平书写方向，startMargin 表示字幕左边距离视频画面左边的距离，例如 startMargin=0.05 则边距为视频宽度的0.05倍（5%）。
- 对于垂直书写方向（无论从右到左还是从左到右），startMargin 表示字幕顶部距离视频画面顶部的距离，例如 startMargin=0.05 则边距为视频高度的0.05倍（5%）。

```
@property(nonatomic, assign) float startMargin;
```

endMargin

沿着字幕文本方向的边距，根据不同的书写方向意义不同。endMargin 是一个比例值，取值范围[0, 1]，即相对于视频画面大小的比例。

- 对于水平书写方向，endMargin 表示字幕右边距离视频画面右边的距离，例如 endMargin=0.05 则边距为视频宽度的0.05倍（5%）。
- 对于垂直书写方向（无论从右到左还是从左到右），endMargin 表示字幕底部距离视频画面底部的距离，例如 endMargin=0.05 则边距为视频高度的0.05倍（5%）。

```
@property(nonatomic, assign) float endMargin;
```

verticalMargin

垂直字幕文本方向的边距，不同的书写方向意义不同。verticalMargin 为一个比例值，取值范围[0, 1]，即相对于视频画面大小的比例。

- 对于水平书写方向，yMargin 表示字幕底部距离视频画面底部的距离，例如 yMargin=0.05 则边距为视频高度的0.05倍（5%）。
- 对于垂直、从右至左书写方向，yMargin 表示字幕右边距离视频画面右边的距离，例如 yMargin=0.05 则边距为视频宽度的0.05倍（5%）。
- 对于垂直、从左至右书写方向，yMargin 表示字幕左边距离视频画面左边的距离，例如 yMargin=0.05 则边距为视频宽度的0.05倍（5%）。

```
@property(nonatomic, assign) float verticalMargin;
```

TXTrackInfo

最近更新时间：2025-05-26 18:16:22

TXTrackInfo 简介

点播播放器播轨道的详细信息。

TX_VOD_MEDIA_TRACK_TYPE（轨道类型）

值	参数名	描述
0	TX_VOD_MEDIA_TRACK_TYPE_UNKNOW	未知类型。
1	TX_VOD_MEDIA_TRACK_TYPE_VIDEO	视频轨。
2	TX_VOD_MEDIA_TRACK_TYPE_AUDIO	音频轨。
3	TX_VOD_MEDIA_TRACK_TYPE_SUBTITLE	字幕轨。

接口详情

参数名	类型	描述
trackIndex	int	轨道 index。
trackType	TX_VOD_MEDIA_TRACK_TYPE	track 类型。
name	NSString	轨道名字。
isSelected	bool	当前轨道是否被选中。
isExclusive	bool	如果是 YES，该类型轨道每个时刻只有一条能被选中，如果是 NO，该类型轨道可以同时选中多条。
isInternal	bool	当前的轨道是否是内部原始轨道。
getTrackIndex	int	获取轨道 index。
getTrackType	TX_VOD_MEDIA_TRACK_TYPE	获取轨道类型。
getTrackName	NSString	获取轨道名称。

isEqual:	bool	轨道是否相同。

trackType

获取轨道类型。

```
@property(nonatomic, assign) TX_VOD_MEDIA_TRACK_TYPE trackType;
```

trackIndex

获取轨道 index。

```
@property(nonatomic, assign) int trackIndex;
```

name

获取轨道名称。

```
@property(nonatomic, copy) NSString *name;
```

isSelected

当前轨道是否被选中。

```
@property(nonatomic, assign) bool isSelected;
```

isExclusive

若为 YES，则同一时刻只能选择一个该类型的轨道；若为 NO，则同一时刻可以同时选择多个该类型的轨道。

```
@property(nonatomic, assign) bool isExclusive;
```

isInternal

当前的轨道是否是内部原始轨道。

```
@property(nonatomic, assign) bool isInternal;
```

getTrackIndex

获取轨道 index。

```
- (int)getTrackIndex;
```

返回值

轨道 index，int。

getTrackType

获取轨道类型。

```
- (TX_VOD_MEDIA_TRACK_TYPE)getTrackType;
```

返回值

类型	描述
TX_VOD_MEDIA_TRACK_TYPE	轨道类型。

getTrackName

获取轨道的名称。

```
- (NSString *)getTrackName;
```

返回值

轨道名称，NSString。

isEqual:

轨道是否相同。

```
- (bool)isEqual:(TXTrackInfo *)trackInfo;
```

返回值

轨道是否相同，BOOL。YES 表示相同， NO 表示不同。

TXVodDef

最近更新时间：2025-05-22 16:24:52

常用结构

TXVodSubtitleData

点播播放器回调的字幕文本数据。

参数名	类型	描述
subtitleData	NSString	字幕内容内容，为空时表示没有字幕内容。
trackIndex	int64_t	当前字幕轨道的 trackIndex。
durationMs	int64_t	字幕持续时间，单位毫秒。暂时为空，请勿使用。
startPositionMs	int64_t	字幕开始时间，单位毫秒。暂时为空，请勿使用。

TXBitrateItem

最近更新时间：2025-05-22 16:24:52

TXBitrateItem 简介

视频码率信息。

字段详情

参数名	类型	描述
index	NSInteger	码率下标，m3u8 文件中的序号。
width	NSInteger	此视频流的宽度
height	NSInteger	此视频流的高度。
bitrate	NSInteger	此视频流的码率。
bandwidth	int64_t	此流的带宽。

TXPlayerAuthParams

最近更新时间：2025-05-26 18:16:22

TXPlayerAuthParams API 简介

点播播放器播放媒资参数， 通过 TXPlayInfoParams 可以配置腾讯云 fileId 和 URL 播放。

接口概览

API	描述
appId	应用 appId。
fileId	视频文件 ID。
timeout	加密链接超时时间戳。
exper	试看时长。
us	唯一标识请求。
sign	防盗链签名。
https	是否用 HTTPS 请求。
url	视频文件 URL。
mediaType	媒资类型。
encryptedMp4Level	MP4加密等级。
headers	HTTP 头。
preferAudioTrack	预下载的音轨名称。

接口详情

appId

应用 ID。如果不设置 URL 该字段必填。

```
@property(nonatomic, assign) int appId;
```

fileId

视频文件 ID。如果不设置 URL 该字段必填。

```
@property(nonatomic, copy) NSString *fileId;
```

timeout

加密链接超时时间戳。可选，会转换为十六进制小写字符串，CDN 服务器会根据该时间戳判断链接是否有效。

```
@property(nonatomic, copy) NSString *fileId;
```

exper

预览时长（以秒为单位），可选。

```
@property(nonatomic, assign) int exper;
```

us

唯一标识请求，增加链接唯一性。

```
@property(nonatomic, copy) NSString *us;
```

sign

防盗链。若不使用防盗链，此参数留空。播放器 API 使用的防盗链参数（t、us、exper）与 CDN 一致，区别仅在于 sign 的计算方式。

- 通用防盗链签名：sign = md5(KEY+appId+fileId+t+us)。
- 预览版防盗链签名：sign = md5(KEY+appId+fileId+t+exper+us)。

```
@property(nonatomic, copy) NSString *sign;
```

https

是否使用 HTTPS 请求，默认为 NO。

```
@property(nonatomic, assign) BOOL https;
```

url

视频 URL。

```
@property(nonatomic, copy) NSString *url;
```

mediaType

媒体类型。默认为 MEDIA_TYPE_AUTO。

```
@property(nonatomic, assign) TX_Enum_MediaType mediaType;
```

参数说明

参数名	类型	描述
mediaType	TX_Enum_MediaType	设置媒资类型，默认为 auto 类型。

encryptedMp4Level

设置 MP4 播放的加密等级。

```
@property(nonatomic, assign) TX_Enum_MP4EncryptionLevel encryptedMp4Level;
```

参数说明

参数名	类型	描述
encryptedMp4Level	TX_Enum_MP4EncryptionLevel	设置 MP4 播放和存储加密等级，从播放器高级版12.2 版本开始支持。

headers

自定义配置播放器播放联网过程中携带的 Http header。

```
@property(nonatomic, strong) NSDictionary *headers;
```

preferAudioTrack

设置启播时优先使用的音轨。在预下载场景，则配置优先下载音轨。

> **说明：**
> 播放器高级版本 12.3 版本开始支持。

```
@property(nonatomic, copy) NSString *preferAudioTrack;
```

参数说明

参数名	类型	描述
preferAudioTrack	NSString	音轨名称。

TXImageSprite

最近更新时间：2025-05-26 18:16:22

TXImageSprite 简介

点播雪碧图解析工具类。

接口概览

API	描述
setVTTUrl:imageUrls:	设置雪碧图地址。
getThumbnail:	获取缩略图。

接口详情

setVTTUrl:imageUrls:

设置雪碧图地址， 设置后会启动子线程下载雪碧图并进行解析。

```
- (void)setVTTUrl:(NSURL *)vttUrl imageUrls:(NSArray<NSURL *> *)images;
```

参数说明

参数名	类型	描述
vttUrl	NSURL	雪碧图 web vtt 描述文件下载 URL。
images	NSArray<NSURL *>	雪碧图图片下载 URL。

getThumbnail:

获取缩略图，获取失败返回 nil。

```
- (UIImage *)getThumbnail:(GLfloat)time;
```

参数说明

参数名	类型	描述
time	GLfloat	时间点，单位秒。

返回值

缩略图，UIImage。

TXPlayerDrmBuilder

最近更新时间：2025-05-26 18:16:22

TXPlayerDrmBuilder 简介

Drm 播放信息，配合 `-[TXVodPlayer startPlayDrm:]` 使用。

接口概览

API	描述
initWithDeviceCertificateUrl:licenseUrl:videoUrl:	构造 Drm 播放信息对象。
deviceCertificateUrl	设置证书提供商 URL。
keyLicenseUrl	设置解密 Key URL。
playUrl	设置播放媒体的 URL。

接口详情

initWithDeviceCertificateUrl:licenseUrl:videoUrl:

构造函数。

```
- (instancetype)initWithDeviceCertificateUrl:(NSString *)certificateUrl licenseUrl:(NSString *)licenseUrl videoUrl:(NSString *)videoUrl;
```

参数说明

参数名	类型	描述
certificateUrl	NSString	证书提供商 URL。
licenseUrl	NSString	播放媒体 URL。
videoUrl	NSString	解密 Key URL。

deviceCertificateUrl

设置证书提供商 URL。

```
@property(nonatomic, strong) NSString *deviceCertificateUrl;
```

参数说明

参数名	类型	描述

deviceCertificateUrl	NSString	证书提供商 URL。

keyLicenseUrl

设置解密 Key URL。

```
@property(nonatomic, strong) NSString *keyLicenseUrl;
```

playUrl

设置播放媒体的 URL。

```
@property(nonatomic, strong) NSString *playUrl;
```

Android TXVodPlayer

最近更新时间：2025-06-04 15:55:52

TXVodPlayer API 简介

TXVodPlayer 是核心播放类，主要负责播放、暂停和倍速控制等播放控制。播放器的完整能力请参见 播放器 SDK 功能说明 。

接口概览

基础播放接口

API	描述
startVodPlay	播放 HTTP URL 形式地址。10.7版本开始，startPlay 变更为 startVodPlay，需要通过 {@link TXLiveBase#setLicence} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和播放器 Licence 均可使用，若您暂未获取上述 Licence ，可单击 播放器 License 进行申请，正式版 License 需购买。
startVodPlay	以 fileId 形式播放，传入TXPlayInfoParams 参数。10.7版本开始，startPlay 变更为 startVodPlay，需要通过 {@link TXLiveBase#setLicence} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和播放器 Licence 均可使用，若您暂未获取上述 Licence 可单击 播放器 License 进行申请，正式版 License 需购买。
startPlayDrm	播放 DRM 加密视频。
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。
seek	跳转到视频流指定时间点，单位秒，小数点后精确到3位。
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。
getEncryptedPlayKey	获取加固加密播放密钥。
setLoop	设置是否循环播放。
isLoop	返回是否循环播放状态。
addSubtitleSource	添加外挂字幕（播放器高级版本才支持）。
getSubtitleTrackInfo	返回字幕轨道信息列表（播放器高级版本才支持）。
getAudioTrackInfo	返回音频轨道信息列表（播放器高级版本才支持）。
selectTrack	选择轨道（播放器高级版本才支持）。
deselectTrack	取消选择轨道（播放器高级版本才支持）。
seekToPdtTime	跳转到视频流指定 PDT（Program Date Time ）时间点，可实现视频快进、快退、进度条跳转等功能，目前只支持 HLS 视频格式（播放器高级版 11.6 版本开始支持）。 参数单位毫秒(ms)。

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 Surface。
setStringOption	设置播放器业务参数，参数格式为 `<String,Object>` 。
setSubtitleStyle	设置字幕样式信息，可在播放后对字幕样式进行更新（播放器高级版本才支持）。
setSubtitleView	设置字幕渲染目标对象 View（播放器高级版本才支持）。

setAudioNormalization	设置音量均衡，响度范围，可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）
	• 关：AUDIO_NORMALIZATION_OFF
	• 开：
	  ○ AUDIO_NORMALIZATION_STANDARD（标准）
	  ○ AUDIO_NORMALIZATION_LOW（低）

	○ AUDIO_NORMALIZATION_HIGH（高） 可填自定义数值：从低到高，范围-70 - 0 LUFS。

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。
getSupportedBitrates	当播放地址为 HSL 时，返回支持的码率（清晰度）列表。
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 。
setAutoMaxBitrate	设置自适应播放可切换的最高码率。

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
setVodSubtitleDataListener	设置视频字幕文本数据输出回调。
setVodListener	设置播放器事件回调。

TRTC 播片相关接口

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。

接口详情

TXVodPlayer

创建点播播放器实例。

```
public TXVodPlayer(Context context)
```

参数说明

参数名	类型	描述
context	Context	系统 Context。

startVodPlay

通过 URL 启动播放。
10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

```
public int startVodPlay(String playUrl)
```

参数说明

参数名	类型	描述
playUrl	String	播放地址。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

startVodPlay

通过腾讯云 fileId 启动播放。

10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

```
public void startVodPlay(TXPlayInfoParams playInfoParams)
```

参数说明

参数名	类型	描述
playInfoParams	TXPlayInfoParams	系统 Context。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

startPlayDrm

播放 Drm 加密视频。
10.7 版本开始，需要通过 XLiveBase#setLicence 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。

> **注意：**
> 播放器高级版本才支持。

```
public int startPlayDrm(TXPlayerDrmBuilder playerDrmBuilder)
```

参数说明

参数名	类型	描述
playerDrmBuilder	TXPlayerDrmBuilder	Drm 播放信息。

返回值

- 0： 播放成功。
- 非 0： 播放失败。

stopPlay

停止播放。

```
public int stopPlay(boolean isNeedClearLastImg)
```

参数说明

参数名	类型	描述

isNeedClearLastImg	boolean	是否需要清除最后一帧画面。 • true：清除最后一帧画面，正常停止播放时，推荐清除。 • false：保留最后一帧画面。

isPlaying

是否正在播放。

```
public boolean isPlaying()
```

pause

暂停播放。

```
public void pause()
```

resume

恢复播放。

```
public void resume()
```

seek

跳转到视频流指定时间点。

```
public void seek(int time)
```

参数说明

参数名	类型	描述
time	int	视频流时间点，单位为秒。

返回值

seek

跳转到视频流指定时间点。

```
public void seek(float time)
```

参数说明

参数名	类型	描述
time	float	视频流时间点，单位秒，小数点后精确到3位。

seek

跳转到视频流指定时间点。

```
public void seek(float timeInSeconds, boolean isAccurateSeek)
```

参数说明

参数名	类型	描述
timeInSeconds	float	视频流时间点，单位秒，小数点后精确到3位。
isAccurateSeek	boolean	是否精准 Seek。 • true: 表示精确 Seek，必须寻找到当前时间点，这个会比较耗时。 • false：表示非精准 Seek，也就是寻找前一个I帧。

seekToPdtTime

跳转到视频流指定时间点。

> **注意：**
> 播放器高级版 11.6 版本开始支持。

```
public void seekToPdtTime(long pdtTimeMs)
```

参数说明

参数名	类型	描述
pdtTimeMs	long	视频流 PDT 时间点，单位毫秒。

getCurrentPlaybackTime

获取当前播放时间点，单位秒。

```
public float getCurrentPlaybackTime()
```

getBufferDuration

获取缓存的总时长，单位秒。

```
public float getBufferDuration()
```

getDuration

获取播放的视频总时长，单位秒。

```
public float getDuration()
```

getPlayableDuration

获取当前可以播放视频时长，单位秒。

```
public float getPlayableDuration()
```

getWidth

获取视频宽度。

```
public int getWidth()
```

getHeight

获取视频高度。

```
public int getHeight()
```

setAutoPlay

设置点播是否 startPlay 后自动开始播放。默认自动播放。

```
public void setAutoPlay(boolean autoPlay)
```

setStartTime

设置播放开始时间，单位秒，需要在启动播放前设置。

```
public void setStartTime(float pos)
```

参数说明

参数名	类型	描述
pos	float	视频流时间点，单位秒，小数点后精确到3位。

setToken

设置加密 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。

```
public void setToken(String token)
```

getEncryptedPlayKey

获取加固加密播放密钥。

```
public static String getEncryptedPlayKey(final String key)
```

setLoop

设置是否循环播放，默认非循环播放。

```
public void setLoop(boolean loop)
```

isLoop

是否循环播放。

```
public void setLoop(boolean loop)
```

addSubtitleSource

添加外挂字幕。
注意：播放器高级版本才支持。

```
public void addSubtitleSource(@NonNull String url, @NonNull String name, String
mimeType)
```

参数说明

参数名	类型	描述
url	String	字幕地址，支持 Http 链接和本地存储绝对路径。
name	String	字幕的名字。如果添加多个字幕，字幕名称请设置为不同的名字，用于区分与其他添加的字幕，否则可能会导致字幕选择错误
mimeType	String	字幕类型，仅支持 VVT 和 SRT 格式。 • VTT 格式：TXVodConstants#VOD_PLAY_MIMETYPE_TEXT_SRT • SRT 格式：TXVodConstants#VOD_PLAY_MIMETYPE_TEXT_VTT

getSubtitleTrackInfo

返回字幕轨道信息列表。

> ⚠ **注意：**
> 播放器高级版本才支持。

```
public List<TXTrackInfo> getSubtitleTrackInfo()
```

返回值

List<TXTrackInfo>：字幕轨道信息列表。

getAudioTrackInfo

返回音频轨道信息列表。

> **注意：**
> 播放器高级版本才支持。

```
public List<TXTrackInfo> getAudioTrackInfo()
```

返回值

List<TXTrackInfo>：音频轨道信息列表。

selectTrack

选择轨道。

> **注意：**
> 播放器高级版本才支持。

```
public void selectTrack(int trackIndex)
```

参数说明

参数名	类型	描述
trackIndex	int	轨道index，通过 TXTrackInfo#getTrackIndex 获取。

deselectTrack

取消选择轨道。

> **注意：**
> 播放器高级版本才支持。

```
public void deselectTrack(int trackIndex)
```

参数说明

参数名	类型	描述

trackIndex	int	轨道 index，通过 TXTrackInfo#getTrackIndex 获取。

setConfig

设置播放器配置信息，请参考 TXVodPlayConfig 。

```
public void setConfig(TXVodPlayConfig config)
```

参数说明

setPlayerView

设置播放器的视频渲染 TXCloudVideoView，仅在启动播放之前设置有效。

```
public void setPlayerView(TXCloudVideoView glRootView)
```

参数说明

参数名	类型	描述
glRootView	TXCloudVideoView	视频渲染 View。

setPlayerView

设置播放器的视频渲染 TextureRenderView，仅在启动播放之前设置有效。

```
public void setPlayerView(TextureRenderView glRootView)
```

参数说明

参数名	类型	描述
glRootView	TextureRenderView	视频渲染 View。

setSurface

设置播放器的视频渲染 Surface，仅在启动播放之前设置有效。

```
public void setSurface(Surface surface)
```

setStringOption

设置播放器业务参数，参数格式为 `<String,Object>`

```
public void setStringOption(String key, Object value)
```

setSubtitleStyle

设置字幕样式信息，支持播放前配置，可以支持在播放后对字幕样式进行更新。

> **注意：**
> 播放器高级版本才支持。

```
public void setSubtitleStyle(TXSubtitleRenderModel renderModel)
```

参数说明

参数名	类型	描述
renderModel	TXSubtitleRenderModel	字幕样式配置参数。

setSubtitleView

设置字幕渲染目标对象 View。

> **注意：**
> 播放器高级版本才支持。

```
public void setSubtitleView(TXSubtitleView subtitleView)
```

参数说明

参数名	类型	描述
subtitleView	TXSubtitleView	播放器 SDK 提供的字幕渲染目标对象 View。

setAudioNormalization

设置音量均衡，响度范围：-70～0(LUFS)。

> **注意：**
> 播放器高级版本才支持。

```
public void setAudioNormalization(float value)
```

参数说明

参数名	类型	描述
value	float	可填预设值（相关类或文件：Android：TXVodConstants；iOS：TXVodPlayConfig.h）

		• 关：AUDIO_NORMALIZATION_OFF
		• 开：
		  ○ AUDIO_NORMALIZATION_STANDARD（标准）
		  ○ AUDIO_NORMALIZATION_LOW（低）
		  ○ AUDIO_NORMALIZATION_HIGH（高）
		可填自定义数值：从低到高，范围-70 – 0 LUFS。

setAutoMaxBitrate

设置自适应播放可切换的最高码率。

```
public void setAutoMaxBitrate(int autoMaxBitrate)
```

enableHardwareDecode

启用或禁用视频硬解码，默认开启硬解码。

```
public boolean enableHardwareDecode(boolean enable)
```

snapshot

获取当前视频帧图像。

```
public void snapshot(TXLivePlayer.ITXSnapshotListener listener)
```

参数说明

参数名	类型	描述
listener	public interface ITXSnapshotListener { void onSnapshot(Bitmap bmp); }	截图回调接口类。

setMirror

设置镜像播放。

```
public void setMirror(boolean mirror)
```

setRate

设置点播的播放速率，默认1.0。

```
public void setRate(float rate)
```

getBitrateIndex

返回当前播放的码率索引。

```
public TXVodPlayer(Context context)
```

getSupportedBitrates

当播放地址为 HSL 时，返回支持的码率（清晰度）列表。

```
public ArrayList<TXBitrateItem> getSupportedBitrates
```

返回值

ArrayList<TXBitrateItem>： 码率列表。

setBitrateIndex

设置当前正在播放的码率索引，腾讯云支持多码率HLS分片对齐，保证最佳体验。无缝切换清晰度时，清晰度切换可能需要等待一小段时间。

```
public void setBitrateIndex(int index) {
```

参数说明

参数名	类型	描述
index	int	码率索引。 • index == -1，表示开启HLS码流自适应。 • index > 0 表示手动切换到对应清晰度码率，index 值可以通过接口 TXVodPlayer#getSupportedBitrates 获取。

setRenderMode

设置播放器图像平铺模式。

```
public void setRenderMode(int mode)
```

参数说明

参数名	类型	描述
mode	int	图像平铺模式，取值有： • TXVodConstants#RENDER_MODE_FULL_FILL_SCREEN：视频画面全屏铺满，将图像等比例铺满整个屏幕，多余部分裁剪掉，此模式下画面不留黑边。 • TXVodConstants#RENDER_MODE_ADJUST_RESOLUTION：视频画面自适应屏幕，将图像等比例缩放，缩放后的宽和高都不会超过显示区域，居中显示，可能会留有黑边。

setRenderRotation

设置播放器图像渲染角度。

```
public void setRenderRotation(int rotation)
```

参数说明

参数名	类型	描述
context	Context	图像渲染角度，取值有： • TXVodConstants#RENDER_ROTATION_PORTRAIT：常规竖屏。 • TXVodConstants#RENDER_ROTATION_LANDSCAPE：右旋90度。

setMute

设置是否静音播放，默认非静音播放。

```
public void setMute(boolean mute)
```

setAudioPlayoutVolume

设置音量大小，范围：0 – 100。

```
public void setAudioPlayoutVolume(int volume)
```

setRequestAudioFocus

是否自动获取音频焦点，默认自动获取。
可以传入 false，然后业务自己管理音频焦点。

```
public boolean setRequestAudioFocus(boolean requestFocus)
```

setVodSubtitleDataListener

设置视频字幕文本数据输出回调。

```
public void setVodSubtitleDataListener(ITXVodPlayListener.ITXVodSubtitleDataListener
listener)
```

参数说明

参数名	类型	描述

listener	ITXVodPlayListener.ITXVodSubtitleDataListener	字幕文本数据回调接口。

setVodListener

创建点播播放器实例。

```
public void setVodListener(ITXVodPlayListener listener)
```

参数说明

参数名	类型	描述
listener	ITXVodPlayListener	播放器事件回调接口。

attachTRTC

点播绑定到 TRTC 服务。

```
public void attachTRTC(Object trtcCloud)
```

detachTRTC

点播解绑 TRTC 服务。

```
public void detachTRTC()
```

publishVideo

开始推送视频流。

```
public void publishVideo()
```

unpublishVideo

取消推送视频流。

```
public void unpublishVideo()
```

publishAudio

开始推送音频流。

```
public void publishAudio()
```

unpublishAudio

取消推送音频流。

```
public void unpublishAudio()
```

ITXVodPlayListener

最近更新时间：2025-05-26 18:16:22

ITXVodPlayListener API 简介

点播播放器播放事件、网络事件回调监听接口。

回调接口概览

API	描述
onPlayEvent	播放事件通知。
onNetStatus	播放过程中网络状态事件回调。

回调接口详情

onPlayEvent

播放事件通知，包括开始播放、首帧事件、Loading 事件、播放进度、结束播放等事件。

```
public void onPlayEvent(final TXVodPlayer player, final int event, final Bundle
param)
```

参数说明

参数名	类型	描述
player	TXVodPlayer	当前播放器对象。
event	int	播放器事件。
param	Bundle	播放事件携带的参数，通过(Key,Value)保存，其中 Key 可以参考 TXVodConstants中的事件参数。

onNetStatus

播放过程中网络状态事件回调。

```
public void onNetStatus(final TXVodPlayer player, final Bundle status)
```

参数说明

参数名	类型	描述
player	TXVodPlayer	当前播放器对象。
status	Bundle	播放过程中网络状态参数，格式为：(Key,Value)。

ITXVodSubtitleDataListener

最近更新时间：2025-05-26 18:16:22

ITXVodSubtitleDataListener API 简介

播放器字幕文本数据回调监听。

回调接口概览

API	描述
onSubtitleData	字幕文本数据回调。

回调接口详情

onSubtitleData

播放事件通知，包括开始播放、首帧事件、Loading 事件、播放进度、结束播放等事件。

```
public void onSubtitleData(TXVodDef.TXVodSubtitleData subtitleData)
```

参数说明

参数名	类型	描述
subtitleData	TXVodSubtitleData	字幕文本数据，详细字段值参考：TXVodSubtitleData 。

播放配置(Config)
TXPlayerGlobalSetting

最近更新时间：2025-05-26 18:16:22

TXPlayerGlobalSetting API 简介

点播播放器全局配置。

接口概览

API	描述
setCacheFolderPath	设置播放引擎的 Cache 目录。
setMaxCacheSize	设置播放引擎的最大缓存大小。
setDrmProvisionEnv	设置 Drm 证书提供商环境。
setPlayCGIHosts	设置腾讯云 PlayCGI 主机域名地址列表。

接口详情

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。

```
public static void setCacheFolderPath(String path)
```

参数说明

参数名	类型	描述
path	String	缓存目录， Sdcard 绝对路径，null 表示不开启缓存。

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。

```
public static void setMaxCacheSize(int sizeMB)
```

参数说明

参数名	类型	描述
sizeMB	int	最大缓存大小，单位：MB。

setDrmProvisionEnv

设置 Drm 证书提供商环境（注意：从 11.2 版本开始支持）。

```
public static void setDrmProvisionEnv(DrmProvisionEnv env)
```

参数说明

参数名	类型	描述
env	DrmProvisionEnv	设置 Drm 证书提供商环境（注意：从 11.2 版本开始支持）。可选值有： • TXPlayerGlobalSetting.DrmProvisionEnv#DRM_PROVISION_ENV_COM，代表使用 google COM 域名证书提供商。 • TXPlayerGlobalSetting.DrmProvisionEnv#DRM_PROVISION_ENV_CN，代表使用 google CN 域名证书提供商。

setPlayCGIHosts

设置腾讯云 PlayCGI 主机域名地址列表，在内置域名请求失败时会启用设置的备用域名。

```
public static void setPlayCGIHosts(List<String> hosts)
```

参数说明

参数名	类型	描述
hosts	List<String>	域名地址列表，域名格式为：playvideo.qcloud.com。

TXVodPlayConfig

最近更新时间：2025-06-04 15:55:52

TXVodPlayConfig API 简介

点播播放器播放配置， 需要在播放前设置。

接口概览

API	描述
setConnectRetryCount	设置播放器在异常场景下重连次数。
setTimeout	设置播放器连接超时时间。
setPlayerType	设置播放器类型。
setHeaders	设置Http header。
setEnableAccurateSeek	设置是否精确 seek。
setAutoRotate	设置播放 MP4 是否自动旋转角度。
setSmoothSwitchBitrate	设置是否平滑切换多码率 HLS。
setCacheMp4ExtName	设置缓存 MP4 文件扩展名。
setProgressInterval	设置进度回调间隔。
setMaxBufferSize	设置播放器最大播放缓冲大小。
setMaxPreloadSize	设置预加载最大缓冲大小。
setEnableRenderProcess	设置播放器是否允许加载渲染后处理服务。
setPreferredResolution	设置 HLS最优的码流进行起播。
setEncryptedMp4Level	设置 MP4 加密播放。
setMediaType	设置播放器播放的媒资类型。
setExtInfo	设置播放器拓展参数。
setPreferredAudioTrack	设置启播时优先使用的音轨。

接口详情

setConnectRetryCount

设置播放器在异常场景下重连次数。
当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过此函数设置 SDK 重连次数，默认值为 3。

```
public void setConnectRetryCount(int count)
```

参数说明

参数名	类型	描述
count	int	播放异常场景下重连次数。

setTimeout

设置播放器连接超时时间，默认值为 10 秒。

```
public void setTimeout(int timeout)
```

参数说明

参数名	类型	描述
interval	int	连接超时时间，单位秒， 默认值为 10 秒。

setCacheFolderPath

设置点播缓存目录。
注意：此接口已经废弃，请使用 `TXPlayerGlobalSetting#setCacheFolderPath` 做全局配置。

```
public void setCacheFolderPath(String folderPath)
```

参数说明

参数名	类型	描述
folderPath	String	缓存路径。

setMaxCacheItems

设置缓存文件个数。
注意：此接口已经废弃，请使用 TXPlayerGlobalSetting#setMaxCacheSize 做全局配置。

```
public void setMaxCacheItems(int maxCacheItems)
```

参数说明

参数名	类型	描述
maxCacheItems	int	最大缓存条目。

setPlayerType

设置播放器类型，默认为腾讯云自研播放器。

```
public void setPlayerType(int playerType)
```

参数说明

参数名	类型	描述
playerType	int	播放器类型，取值有： • TXVodConstants#PLAYER_SYSTEM_MEDIA_PLAYER： Android 系统播放器。 • TXVodConstants#PLAYER_THUMB_PLAYER：腾讯云自研播放器，默认值。

setHeaders

自定义配置播放器播放联网过程中携带的 Http header。

```
public void setHeaders(Map<String, String> headers)
```

参数说明

参数名	类型	描述
headers	Map<String, String>	自定义的 Http header 内容。

setEnableAccurateSeek

设置是否精确 seek，默认 true。

```
public void setEnableAccurateSeek(boolean accurateSeek)
```

参数说明

参数名	类型	描述
accurateSeek	boolean	是否精确 seek。

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，默认值为 YES。

```
public void setAutoRotate(boolean autoRotate)
```

参数说明

参数名	类型	描述
autoRotate	boolean	播放时旋转角度是否自动旋转。

setSmoothSwitchBitrate

设置是否平滑切换多码率 HLS，默认 false。

```
public void setSmoothSwitchBitrate(boolean smoothSwitchBitrate)
```

参数说明

参数名	类型	描述
smoothSwitchBitrate	boolean	是否平滑切换多码率 HLS。

setCacheMp4ExtName

设置缓存 mp4 文件扩展名，默认为 mp4 。

```
public void setCacheMp4ExtName(String cacheMp4ExtName)
```

参数说明

参数名	类型	描述
cacheMp4ExtName	boolean	文件扩展名称。

setProgressInterval

设置进度回调间隔，默认为 0.5 秒回调一次。

```
public void setProgressInterval(int intervalMs)
```

参数说明

参数名	类型	描述
intervalMs	int	间隔时间，单位毫秒。

setMaxBufferSize

设置播放器最大播放缓冲大小，单位 MB 。

```
public void setMaxBufferSize(float maxBufferSize)
```

参数说明

参数名	类型	描述
maxBufferSize	float	播放缓冲大小。

setMaxPreloadSize

设置预加载最大缓冲大小，单位：MB。

```
public void setMaxPreloadSize(float maxPreloadSize)
```

参数说明

参数名	类型	描述
maxPreloadSize	float	预加载大小。

setFirstStartPlayBufferTime

设置播放器首缓需要加载的数据时长，单位 ms，默认值为100ms。
注意：此接口已经废弃，请使用 #setMaxBufferSize或 #setMaxPreloadSize 设置缓冲大小。

```
public void setFirstStartPlayBufferTime(int milliseconds)
```

参数说明

参数名	类型	描述
milliseconds	int	时长大小。

setEnableRenderProcess

设置播放器是否允许加载渲染后处理服务，默认关闭。

```
public void setEnableRenderProcess(boolean enableRenderProcess)
```

参数说明

参数名	类型	描述
enableRenderProcess	boolean	是否允许加载后渲染后处理服务。

setPreferredResolution

播放 HLS 有多条码流时，播放器根据设定的 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。

```
public void setPreferredResolution(long preferredResolution)
```

参数说明

参数名	类型	描述
preferredResolution	long	视频宽高的乘积（width * height）。

setEncryptedMp4Level

设置 MP4 加密播放，默认不加密。

```
public void setEncryptedMp4Level(int level)
```

参数说明

参数名	类型	描述
level	int	设置 MP4 播放和存储加密等级，从播放器高级版12.2 版本开始支持，目前支持： • TXVodConstants#MP4_ENCRYPTION_LEVEL_NONE：非加密播放，默认支持。 • TXVodConstants#MP4_ENCRYPTION_LEVEL_L2：MP4 本地加密播放。

setMediaType

设置播放器播放的媒资类型，默认为 AUTO 类型。

```
public void setMediaType(int mediaType)
```

参数说明

参数名	类型	描述
mediaType	int	设置媒资类型，默认为 AUTO 类型。可选值有： • TXVodConstants#MEDIA_TYPE_AUTO，AUTO 类型（默认值，自适应码率播放暂不支持）。 • TXVodConstants#MEDIA_TYPE_HLS_VOD，HLS 点播媒资。 • TXVodConstants#MEDIA_TYPE_HLS_LIVE，HLS 直播媒资。 • TXVodConstants#MEDIA_TYPE_FILE_VOD，MP4 等通用文件点播媒资（从 11.2 版本开始支持）。 • TXVodConstants#MEDIA_TYPE_DASH_VOD，DASH 点播媒资（从 11.2 版本开始支持）

setExtInfo

设置播放器拓展参数。

```
public void setExtInfo(Map<String, Object> map)
```

参数说明

参数名	类型	描述
map	Map	拓展参数。

setPreferredAudioTrack

设置启播时优先使用的音轨，播放器高级版本 12.3 版本开始支持。

```
public void setPreferredAudioTrack(String audioTrackName)
```

参数说明

参数名	类型	描述
audioTrackName	String	音轨名称。

视频下载(Download) TXVodDownloadManager

最近更新时间：2025-05-26 18:16:22

TXVodDownloadManager API 简介

点播播放器视频下载接口类。
视频下载支持下载 MP4 和 HLS 视频，对应嵌套 HLS 视频，需要指定偏好清晰度（preferredResolution）。

类常量

参数名	类型	值	描述
TXVodDownloadManager#DOWNLOAD_SUCCESS	int	0	下载成功。
TXVodDownloadManager#DOWNLOAD_AUTH_FAILED	int	-5001	fileId下载鉴权失败。
TXVodDownloadManager#DOWNLOAD_NO_FILE	int	-5003	下载文件不存在。
TXVodDownloadManager#DOWNLOAD_FORMAT_ERROR	int	-5004	下载格式不支持。
TXVodDownloadManager#DOWNLOAD_DISCONNECT	int	-5005	网络错误。
TXVodDownloadManager#DOWNLOAD_HLS_KEY_ERROR	int	-5006	获取 HLS 解密 Key 失败 。
TXVodDownloadManager#DOWNLOAD_PATH_ERROR	int	-5007	下载目录访问失败。
TXVodDownloadManager#DOWNLOAD_403FORBIDDEN	int	-5008	签名过期等或者请求不合法。

接口概览

API	描述
getInstance	获取 TXVodDownloadManager 实例对象，单例模式。
setHeaders	设置下载 HTTP 请求头。
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。
startDownloadDrm	以 Drm 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，耗时接口，请不要在主线程调用。
getDownloadMediaInfo	获取下载信息，耗时接口，请不要在主线程调用。
getDownloadMediaInfo	获取下载信息，耗时接口，请不要在主线程调用。

接口详情

getInstance

获取 TXVodDownloadManager 实例对象，单例模式。

```
public static TXVodDownloadManager getInstance()
```

setHeaders

设置下载 HTTP 请求头。

```
public void setHeaders(Map<String, String> headers)
```

setListener

设置下载回调方法，下载前必须设好。

```
public void setListener(ITXVodDownloadListener listener)
```

参数说明

参数名	类型	描述
listener	ITXVodDownloadListener	下载监听状态回调。

startDownloadUrl

以 URL 方式开始下载。
注意：启动下载前，请先设置好播放引擎的缓存目录 `TXPlayerGlobalSetting#setCacheFolderPath` 。

```
public TXVodDownloadMediaInfo startDownloadUrl(String url, long preferredResolution,
String userName)
```

参数说明

参数名	类型	描述
url	String	下载地址，必选参数，否则下载失败。
preferredResolution	long	下载偏好清晰度，多清晰度 URL 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600 = 1280*720)，单清晰度传入-1。
userName	String	账户名称，可选参数，不传默认为"default"。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

startDownloadUrl

以 URL 方式开始下载。

已废弃，推荐使用：startDownloadUrl(String, long, String)

```
public TXVodDownloadMediaInfo startDownloadUrl(String url, String userName)
```

startDownloadUrl

以 URL 方式开始下载。

已废弃，推荐使用：startDownloadUrl(String, long, String)

```
public TXVodDownloadMediaInfo startDownloadUrl(String url)
```

startDownload

以腾讯云视频 fileId 方式开始下载

```
public TXVodDownloadMediaInfo startDownload(final TXVodDownloadDataSource
dataSource)
```

参数说明

参数名	类型	描述
dataSource	TXVodDownloadDataSource	下载资源对象。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

startDownloadDrm

以 Drm 方式开始下载

```
public TXVodDownloadMediaInfo startDownloadDrm(final TXPlayerDrmBuilder drmBuilder,
long preferredResolution, String userName)
```

参数说明

参数名	类型	描述
drmBuilder	TXPlayerDrmBuilder	Drm 资源构造。
preferredResolution	long	下载偏好清晰度，多清晰度 URL 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600=1280*720)，单清晰度传入-1。
userName	String	账户名称，可选参数，不传默认为"default"。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

stopDownload

停止下载，ITXVodDownloadListener.onDownloadStop 回调时停止成功。

```
public void stopDownload(TXVodDownloadMediaInfo downloadMediaInfo)
```

参数说明

参数名	类型	描述
downloadMediaInfo	TXVodDownloadMediaInfo	视频下载信息。

deleteDownloadFile

删除下载文件。
已废弃，推荐使用 deleteDownloadMediaInfo 。

```
public boolean deleteDownloadFile(String playPath)
```

参数说明

参数名	类型	描述
playPath	String	文件路径。

deleteDownloadMediaInfo

删除下载信息。

```
public boolean deleteDownloadMediaInfo(TXVodDownloadMediaInfo downloadMediaInfo)
```

参数说明

参数名	类型	描述
downloadMediaInfo	TXVodDownloadMediaInfo	视频下载信息。

getDownloadMediaInfoList

获取所有用户的下载列表信息。

```
public List<TXVodDownloadMediaInfo> getDownloadMediaInfoList()
```

返回值

视频下载信息列表：List<TXVodDownloadMediaInfo>。

getDownloadMediaInfo

获取下载信息。
调用此接口要确保之前通过 `TXVodDownloadDataSource(int, String, int, String, String)` 创建下载任务参数，具体参考 `TXVodDownloadMediaInfo#getDataSource` 和 `TXVodDownloadDataSource#getUserName` 。

```
public TXVodDownloadMediaInfo getDownloadMediaInfo(int appId, String fileId, int
qualityId, String userName)
```

参数说明

参数名	类型	描述
appId	int	腾讯云点播应用 appId。
fileId	String	腾讯云点播视频 fileId。
qualityId	int	视频画质 Id ， 具体参考 TXVodDownloadDataSource#QUALITY_240P 常量。
userName	String	账户名称，须与下载时传入的账户名称一致，若下载时未传入，这里传入空字符""。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

getDownloadMediaInfo

获取下载信息，调用此接口要确保之前调用 startDownloadUrl(String, long, String) 启动下载。

```
public TXVodDownloadMediaInfo getDownloadMediaInfo(String url, long
```

```
preferredResolution, String userName)
```

参数说明

参数名	类型	描述
url	String	下载地址，必选参数，否则下载失败。
preferredResolution	long	下载偏好清晰度，多清晰度 URL 为必选参数，值为偏好清晰度宽 * 高(如720p传入921600=1280*720)，单清晰度传入-1。
userName	String	账户名称，可选参数，不传默认为"default"。

返回值

视频下载信息 TXVodDownloadMediaInfo 。

getDownloadMediaInfo

获取 URL下载信息，已废弃，推荐使用：getDownloadMediaInfo(String, long, String)。

```
public TXVodDownloadMediaInfo getDownloadMediaInfo(String url)
```

getDownloadMediaInfo

获取 URL下载信息，已废弃，推荐使用：getDownloadMediaInfo(String, long, String)。

```
public TXVodDownloadMediaInfo getDownloadMediaInfo(int appId, String fileId, int
qualityId)
```

ITXVodDownloadListener

最近更新时间：2025-05-26 18:16:22

ITXVodDownloadListener API 简介

点播播放器下载回调监听接口。

回调接口概览

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。

回调接口详情

onDownloadStart

下载开始。

```
void onDownloadStart(TXVodDownloadMediaInfo mediaInfo)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadProgress

下载进度更新。

```
void onDownloadProgress(TXVodDownloadMediaInfo mediaInfo)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadStop

下载停止。调 TXVodDownloadManager#stopDownload 方法会收到此回调。

```
void onDownloadStop(TXVodDownloadMediaInfo mediaInfo)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadFinish

下载结束。

```
void onDownloadFinish(TXVodDownloadMediaInfo mediaInfo)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。

onDownloadError

下载过程中遇到错误。

```
void onDownloadError(TXVodDownloadMediaInfo mediaInfo, int error, String reason)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。
error	int	下载错误码，参考 下载错误码 。
reason	String	下载错误信息。

hlsKeyVerify

下载 HLS，遇到加密的文件，将解密 key 给外部校验。TXVodDownloadDataSource。
注意：废弃接口，接入时空实现即可。

```
int hlsKeyVerify(TXVodDownloadMediaInfo mediaInfo, String url, byte[] receive)
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	视频下载信息。
url	String	视频下载 URL。
receive	byte[]	receive 服务器返回值。

返回值：

- 0：校验正确，继续下载。
- 其它值：校验失败，抛出下载错误。

TXVodDownloadDataSource

最近更新时间：2025-05-22 16:24:52

TXVodDownloadDataSource 简介

点播下载资源对象。

类常量

参数名	类型	值	描述
TXVodDownloadDataSource#QUALITY_OD	int	0	原画质。
TXVodDownloadDataSource#QUALITY_240P	int	240	流畅240P。
TXVodDownloadDataSource#QUALITY_360P	int	360	流畅360P。
TXVodDownloadDataSource#QUALITY_480P	int	480	标清480P。
TXVodDownloadDataSource#QUALITY_540P	int	540	标清540P。
TXVodDownloadDataSource#QUALITY_720P	int	720	高清720P 。
TXVodDownloadDataSource#QUALITY_1080P	int	1080	全高清1080P。

接口详情

TXVodDownloadDataSource

构造下载资源对象，用于场景腾讯云视频 fileId 下载。

```
public TXVodDownloadDataSource(int appId, String fileId, int quality, String pSign,
String userName)
```

参数说明

参数名	类型	描述
appId	int	腾讯云点播应用 appId。
fileId	String	腾讯云点播视频 fileId。

qualityId	int	视频画质 Id ， 具体参考 TXVodDownloadDataSource#QUALITY_240P 常量。
pSign	String	视频播放签名。
userName	String	账户名称，须与下载时传入的账户名称一致，若下载时未传入，这里传入空字符""。

TXVodDownloadDataSource

构造下载资源对象，用于场景腾讯云视频 fileId V2 版本下载。
已废弃，推荐使用：TXVodDownloadDataSource(int appId, String fileId, int quality, String pSign, String userName)。

```
public TXVodDownloadDataSource(TXPlayerAuthBuilder authBuilder, int quality)
```

TXVodDownloadDataSource

构造下载资源对象，用于场景腾讯云视频 fileId V2 版本下载。
已废弃，推荐使用：TXVodDownloadDataSource(int appId, String fileId, int quality, String pSign, String userName)。

```
public TXVodDownloadDataSource(TXPlayerAuthBuilder authBuilder, String templateName)
```

setToken

设置此值后，播放器自动在 URL 中的文件名之前增加 voddrm.token.<Token>

```
public void setToken(String token)
```

setQuality

设置画质 Id。

```
public void setQuality(int quality)
```

参数说明

参数名	类型	描述
quality	int	视频画质 Id ， 具体参考 TXVodDownloadDataSource#QUALITY_240P 常量。

getAppId

获取传入的 appId。

```
public int getAppId()
```

getFileId

获取传入的 fileId。

```
public String getFileId()
```

getPSign

获取传入的下载签名。

```
public String getPSign()
```

getQuality

获取传入的 quality。

```
public int getQuality()
```

getUserName

获取传入的 userName，默认“default”。

```
public String getUserName()
```

getToken

获取传入的 token。

```
public String getToken()
```

TXVodDownloadMediaInfo

最近更新时间：2025-05-26 18:16:22

TXVodDownloadMediaInfo 简介

点播下载媒资描述。

类常量

参数名	类型	值	描述
TXVodDownloadMediaInfo#STATE_INIT	int	0	下载初始态。
TXVodDownloadMediaInfo#STATE_START	int	1	下载开始。
TXVodDownloadMediaInfo#STATE_STOP	int	2	下载停止。
TXVodDownloadMediaInfo#STATE_ERROR	int	3	下载出错。
TXVodDownloadMediaInfo#STATE_FINISH	int	4	下载完成。

接口概览

API	描述
getDataSource	用腾讯云视频 fileId 下载时，获取传入的下载源媒资信息。
getDuration	获取视频的总时长，单位毫秒。
getPlayableDuration	获取已下载的可播放时长，单位毫秒。
getSize	获取下载文件总大小，单位：Byte，只针对 fileId 下载源有效。 注意：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法获取。
getDownloadSize	获取已下载文件大小，单位：Byte，只针对 fileId 下载源有效。
getProgress	获取当前下载进度。
getPlayPath	获取当前下载资源的播放路径，可传给 TXVodPlayer 播放。
getDownloadState	获取下载状态。
isDownloadFinished	判断是否下载完成。

getSpeed	获取下载速度，单位：KByte/秒。（10.9 版本开始支持）
isResourceBroken	判断下载后的视频资源是否损坏，如下载完被删除等情况将返回true。（11.0 版本开始支持）
getTaskId	获取任务 id，唯一表示下载任务。
getUrl	获取实际下载地址。
getUserName	获取下载账户名称。
getPreferredResolution	获取下载偏好分辨率。

接口详情

getDataSource

已腾讯云视频 fileId 下载时获取传入的下载源媒资信息。

```
public TXVodDownloadDataSource getDataSource()
```

返回值

下载资源对象信息：TXVodDownloadDataSource

getDuration

获取视频的总时长，单位毫秒。

```
public int getDuration()
```

getPlayableDuration

获取已下载的可播放时长，单位毫秒。

```
public TXVodDownloadDataSource(TXPlayerAuthBuilder authBuilder, String templateName)
```

getSize

获取下载文件总大小，单位：Byte，只针对腾讯云视频 fileId 下载源有效。
注意：总大小是指上传到腾讯云点播控制台的原始文件的大小，转自适应码流后的子流大小，暂时无法获取。

```
public long getSize()
```

getDownloadSize

获取已下载文件大小，单位：Byte，只针对腾讯云视频 fileId 下载源有效。

```
public long getDownloadSize()
```

getProgress

获取当前下载进度。

```
public float getProgress()
```

getPlayPath

获取当前下载资源的播放路径，可传给 TXVodPlayer 播放。

```
public String getPlayPath()
```

getDownloadState

获取下载状态。

```
public int getDownloadState()
```

isDownloadFinished

判断是否下载完成。

```
public boolean isDownloadFinished()
```

getSpeed

获取下载速度，单位：KByte/秒。（10.9 版本开始支持）

```
public int getSpeed()
```

getTaskId

获取任务 id，唯一表示下载任务。

```
public int getTaskId()
```

isResourceBroken

判断下载后的视频资源是否损坏，如下载完被删除等情况将返回true。（11.0 版本开始支持）

```
public boolean isResourceBroken()
```

getUrl

获取实际下载地址。

```
public String getUrl()
```

getUserName

获取下载账户名称。

```
public String getUserName()
```

getPreferredResolution

获取下载偏好分辨率。

```
public long getPreferredResolution()
```

TXVodPreloadManager

最近更新时间：2025-05-26 18:16:22

TXVodPreloadManager API 简介

点播播放器预下载接口类。
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。

接口概览

API	描述
getInstance	获取 TXVodPreloadManager 实例对象，单例模式。
startPreload:URL	通过 URL 启动预下载。
startPreload:TXPlayInfoParams	通过 fileId 或 URL 启动预下载，推荐优先使用此接口。
stopPreload	停止预下载。

接口详情

getInstance

获取 TXVodPreloadManager 实例对象，单例模式。

```
public static TXVodPreloadManager getInstance(Context context)
```

startPreload:URL

通过 URL 启动预下载。
注意：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`，这个设置是全局配置需和播放器保持一致，否则会造成播放缓存失效。

```
public int startPreload(final String url, final float preloadSizeMB, final long
preferredResolution, final ITXVodPreloadListener listener)
```

参数说明

参数名	类型	描述
url	String	预下载的视频 URL。
preloadSizeMB	float	预下载的大小，单位：MB。
preferredResolution	long	期望下载的分辨率，视频宽高的乘积（width * height）。 不支持多分辨率或不需指定时，传-1。

listener	ITXVodPreload Listener	预下载监听状态回调。

返回值

任务 ID，可用这个任务 ID 停止预下载 `TXVodPreloadManager#stopPreload` 。
如果返回-1， 表示此任务 ID 无效。

startPreload:TXPlayInfoParams

启动预下载，支持通过腾讯云 fileId 和 视频 URL 预下载。

- 如果 TXPlayInfoParams#getUrl 不为空，则优先启动视频 URL 预下载，此时支持在主线调用。
- 如果 TXPlayInfoParams#getFileId 不为空，则启动视频 fileId 预下载，此时不支持在主线调用。

注意：

1. 预下载是耗时操作，请不要在主线程调用，在主线程调用将会抛出非法调用异常。
2. 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` ，这个设置是全局配置需和播放器保持一致，否则会造成播放缓存失效。

```
public int startPreload(final TXPlayInfoParams playInfoParams, final float
preloadSizeMB, final long preferredResolution,
        final ITXVodFilePreloadListener listener)
```

参数说明

参数名	类型	描述
playInfoParams	TXPlayInfoParams	预下载信息。 可通过 TXPlayInfoParams#setHeaders 设置预下载 http 请求头，通过TXPlayInfoParams#setPreferAudioTrack 设置预下载的音轨名称。
preloadSizeMB	float	预下载的大小，单位：MB。
preferredResolution	long	期望下载的分辨率，视频宽高的乘积（width * height）。 不支持多分辨率或不需指定时，传-1。
listener	ITXVodFilePreload Listener	预下载监听状态回调。

返回值

任务 ID，可用这个任务 ID 停止预下载 `TXVodPreloadManager#stopPreload` 。
如果返回-1， 表示此任务 ID 无效。

stopPreload

停止预下载。

```
public void stopPreload(int taskID)
```

参数说明

参数名	类型	描述
taskID	int	任务 ID。ID 从 `TXVodPreloadManager#startPreload` 返回值得到。

ITXVodPreloadListener

最近更新时间：2025-05-26 18:16:22

ITXVodPreloadListener API 简介

预下载 URL 状态回调监听接口。

回调接口概览

API	描述
onComplete	视频预下载完成。
onError	视频预下载出错。

回调接口详情

onComplete

预下载结束成功回调。

```
void onComplete(int taskID, String url);
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
url	String	预下载任务 URL。

onError

预下载失败回调。

```
void onError(int taskID, String url, int code, String msg)
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
url	String	预下载任务 URL。
code	int	错误码。
msg	String	错误描述信息。

ITXVodFilePreloadListener

最近更新时间：2025-05-26 18:16:22

ITXVodFilePreloadListener API 简介

腾讯云视频 fileId 和 URL 预下载状态回调监听接口。

回调接口概览

API	描述
onStart	视频预下载开始。 对于 fileId 预下载，在换链成功后，启动预下载前回调。
onComplete	视频预下载完成。
onError	视频预下载出错。

回调接口详情

onStart

预下载结束成功回调。

```
public void onStart(int taskID, String fileId, String url, Bundle bundle)
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
fileId	int	预下载的视频 fileId。
url	String	预下载任务 URL，为换链后的视频 URL，可用于后续播放。
bundle	Bundle	预下载携带的额外信息。

onComplete

预下载结束成功回调。

```
void onComplete(int taskID, String url);
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。

url	String	预下载任务 URL。

onError

预下载失败回调。

```
void onError(int taskID, String url, int code, String msg)
```

参数说明

参数名	类型	描述
taskID	int	预下载任务 ID。
url	String	预下载任务 URL。
code	int	错误码。
msg	String	错误描述信息。

类型定义
TXVodConstants

最近更新时间：2025-05-22 16:24:53

TXVodConstants API 简介

点播播放器用到的常量类。

图像平铺模式

值	参数名	描述
0	RENDER_MODE_FULL_FILL_SCREEN	视频画面全屏铺满。
1	RENDER_MODE_ADJUST_RESOLUTION	视频画面自适应屏幕。

图像渲染角度

值	参数名	描述
0	RENDER_ROTATION_PORTRAIT	常规竖屏。
270	RENDER_ROTATION_LANDSCAPE	右旋90度。

播放事件列表

值	参数名	描述
2002	VOD_PLAY_EVT_HIT_CACHE	启播命中缓存。
2003	VOD_PLAY_EVT_RCV_FIRST_I_FRAME	网络接收到首个视频数据包（IDR）。
2004	VOD_PLAY_EVT_PLAY_BEGIN	视频播放开始。
2005	VOD_PLAY_EVT_PLAY_PROGRESS	视频播放进度。
2006	VOD_PLAY_EVT_PLAY_END	视频播放结束。
6001	VOD_PLAY_EVT_LOOP_ONCE_COMPLETE	循环一轮播放结束
2007	VOD_PLAY_EVT_PLAY_LOADING	视频播放 Loading。

2008	VOD_PLAY_EVT_START_VIDEO_DECODER	解码器启动。
2009	VOD_PLAY_EVT_CHANGE_RESOLUTION	视频分辨率改变。
2010	VOD_PLAY_EVT_GET_PLAYINFO_SUCC	获取点播文件信息成功。
2011	VOD_PLAY_EVT_CHANGE_ROTATION	视频旋转信息。
2013	VOD_PLAY_EVT_VOD_PLAY_PREPARED	视频加载完毕。
2014	VOD_PLAY_EVT_VOD_LOADING_END	loading 结束。
2017	VOD_PLAY_EVT_FIRST_VIDEO_PACKET	收到首帧数据（12.0 版本开始支持）。
2019	VOD_PLAY_EVT_SEEK_COMPLETE	Seek 完成（10.3版本开始支持）。
2020	VOD_PLAY_EVT_SELECT_TRACK_COMPLETE	切换轨道完成。
2026	VOD_PLAY_EVT_RCV_FIRST_AUDIO_FRAME	音频首次播放。
2103	VOD_PLAY_WARNING_RECONNECT	网络断连, 已启动自动重连。
2030	VOD_PLAY_EVT_VIDEO_SEI	视频 sei 信息事件。
2031	VOD_PLAY_EVT_HEVC_DOWNGRADE_PLAYBACK	HEVC 降级播放。
-2301	VOD_PLAY_ERR_NET_DISCONNECT	网络断连，且经多次重连抢救无效。
-2303	VOD_PLAY_ERR_FILE_NOT_FOUND	文件不存在。
-2304	VOD_PLAY_ERR_HEVC_DECODE_FAIL	HEVC 解码失败。
-2305	VOD_PLAY_ERR_HLS_KEY	HLS 解密 key 获取失败。
-2306	VOD_PLAY_ERR_GET_PLAYINFO_FAIL	获取点播文件信息失败。

2106	VOD_PLAY_WARNING_HW_ACCELERATION_FAIL	硬解启动失败，采用软解。
-5	VOD_PLAY_ERR_INVALID_LICENCE	license 不合法，播放失败。 注：在 startVodPlay 之前，需要通过 TXLiveBase#setLicence 设置 License 后方可成功播放，否则将播放失败（黑屏），全局仅设置一次即可。直播 License、短视频 License 和播放器 License 均可使用，若您暂未获取上述 License ，可单击 播放器 License 进行申请，正式版 License 需购买。
-6004	VOD_PLAY_ERR_SYSTEM_PLAY_FAIL	系统播放器播放错误。
-6006	VOD_PLAY_ERR_DECODE_VIDEO_FAIL	视频解码错误，视频格式不支持。
-6007	VOD_PLAY_ERR_DECODE_AUDIO_FAIL	音频解码错误，音频格式不支持。
-6008	VOD_PLAY_ERR_DECODE_SUBTITLE_FAIL	字幕解码错误。
-6009	VOD_PLAY_ERR_RENDER_FAIL	视频渲染错误。
-6010	VOD_PLAY_ERR_PROCESS_VIDEO_FAIL	视频后处理错误。
-6101	VOD_PLAY_ERR_DRM	DRM 播放失败

播放事件参数

值	参数名	描述
"CPU_USAGE"	NET_STATUS_CPU_USAGE	当前瞬时 CPU 使用率。
"VIDEO_WIDTH"	NET_STATUS_VIDEO_WIDTH	视频分辨率 – 宽。
"VIDEO_HEIGHT"	NET_STATUS_VIDEO_HEIGHT	视频分辨率 – 高。
"NET_SPEED"	NET_STATUS_NET_SPEED	当前的网络数据接收速度，单位：KBps。
"VIDEO_FPS"	NET_STATUS_VIDEO_FPS	当前流媒体的视频帧率。
"VIDEO_BITRATE"	NET_STATUS_VIDEO_BITRATE	当前流媒体的视频码率，单位 bps。
"AUDIO_BITRATE"	NET_STATUS_A	当前流媒体的音频码率，单位 bps。

;	UDIO_BITRATE	
"VIDEO_CACHE"	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，单位：KBps。
"SERVER_IP"	NET_STATUS_SERVER_IP	连接的服务器 IP。
"EVT_UTC_TIME"	EVT_UTC_TIME	UTC 时间。
"EVT_TIME"	EVT_TIME	事件发生时间。
"EVT_MSG"	EVT_DESCRIPTION	事件说明。
"EVT_PARAM1"	EVT_PARAM1	事件参数1。
"EVT_PARAM2"	EVT_PARAM2	事件参数2。
"EVT_PLAY_COVER_URL"	EVT_PLAY_COVER_URL	视频封面。
"EVT_PLAY_URL"	EVT_PLAY_URL	视频地址。
"EVT_PLAY_NAME"	EVT_PLAY_NAME	视频名称。
"EVT_PLAY_DESCRIPTION"	EVT_PLAY_DESCRIPTION	视频简介。
"EVT_PLAY_PROGRESS_MS"	EVT_PLAY_PROGRESS_MS	播放进度（毫秒）。
"EVT_PLAY_DURATION_MS"	EVT_PLAY_DURATION_MS	播放总长（毫秒）。
"EVT_PLAY_PROGRESS"	EVT_PLAY_PROGRESS	播放进度。
"EVT_PLAY_DURATION"	EVT_PLAY_DURATION	播放总长。
"EVT_PLAYABLE_DURATION_MS"	EVT_PLAYABLE_DURATION_MS	点播可播放时长（毫秒）。
"EVT_PLAYABLE_RATE"	EVT_PLAYABLE_RATE	播放速率。
"EVT_PLAYABLE_DURATION"	EVT_PLAYABLE_DURATION	点播可播放时长。
"EVT_IMAGESPRIT_WEBVTTURL"	EVT_IMAGESPRIT_WEBVTTURL	雪碧图 web vtt 描述文件下载 URL。
"EVT_IMAGESPRIT_IMAGEURL_LIS	EVT_IMAGESPRIT_IMAGEURL_LI	雪碧图图片下载 URL。

T"	ST	
"EVT_DRM_TYPE"	EVT_DRM_TYPE	加密类型。
"EVT_CODEC_TYPE"	EVT_CODEC_TYPE	视频编码类型。
"EVT_KEY_FRAME_CONTENT_LIST"	EVT_KEY_FRAME_CONTENT_LIST	视频关键帧描述信息。
"EVT_KEY_FRAME_TIME_LIST"	EVT_KEY_FRAME_TIME_LIST	关键帧时间。
"EVT_PLAY_PDT_TIME_MS"	EVT_PLAY_PDT_TIME_MS	播放 PDT 时间（毫秒）。
"EVT_KEY_VIDEO_ROTATION"	EVT_KEY_VIDEO_ROTATION	MP4视频旋转角度。
"EVT_KEY_WATER_MARK_TEXT"	EVT_KEY_WATER_MARK_TEXT	幽灵水印文本（11.5版本开始支持）。

播放器媒资类型

值	参数名	描述
0	MEDIA_TYPE_AUTO	AUTO 类型。
1	MEDIA_TYPE_HLS_VOD	自适应码率播放 HLS 点播媒资。
2	MEDIA_TYPE_HLS_LIVE	自适应码率播放 HLS 直播媒资。
3	MEDIA_TYPE_FILE_VOD	MP4 等通用文件点播媒资。
4	MEDIA_TYPE_DASH_VOD	DASH 点播媒资。

MP4 加密等级

值	参数名	描述
0	MP4_ENCRYPTION_LEVEL_NONE	MP4 不加密播放。
2	MP4_ENCRYPTION_LEVEL_L2	L2， MP4 本地加密播放。

未分类变量

值	参数名	描述
0	PLAYER_SYSTEM_MEDIA_PLAYER	系统播放器。

1	PLAYER_THUMB_PLAYER	自研播放器，支持软解，兼容性更好。
-1	INDEX_AUTO	自适应码率 index 标识。
"450"	PLAYER_OPTION_KEY_SUBTITLE_OUTPUT_TYPE	外挂字幕输出类型配置 Key
"backup_url"	VOD_KEY_BACKUP_URL	降级播放备选 URL Key。
"mimetype"	VOD_KEY_MIMETYPE	播放资源的 Mimetype Key。
"text/x-subrip"	VOD_PLAY_MIMETYPE_TEXT_SRT	外挂字幕 SRT 格式。
"text/vtt"	VOD_PLAY_MIMETYPE_TEXT_VTT	外挂字幕 VTT 格式。
"EVT_KEY_WATER_MARK_TEXT"	EVT_KEY_WATER_MARK_TEXT	幽灵水印文本（11.5版本开始支持）。

TXPlayInfoParams

最近更新时间：2025-05-26 18:16:22

TXPlayInfoParams API 简介

点播播放器播放媒资参数， 通过 TXPlayInfoParams 可以配置腾讯云 fileId 和 url 播放。

接口概览

API	描述
TXPlayInfoParams:fileId	创建通过腾讯云点播 fileId 播放的媒资实例。
TXPlayInfoParams:url	创建通过 URL 播放的媒资实例。
setMediaType	设置播放器播放的媒资类型。
setHeaders	设置 Http header。
setEncryptedMp4Level	设置 MP4 加密播放。
setPreferAudioTrack	设置启播时优先使用的音轨。

接口详情

TXPlayInfoParams:fileId

创建通过腾讯云点播 fileId 播放的媒资实例。

```
public TXPlayInfoParams(int appId, String fileId, String pSign)
```

参数说明

参数名	类型	描述
appId	int	腾讯云点播应用 appId。
fileId	String	腾讯云点播资源 fileId。
pSign	String	播放签名。

TXPlayInfoParams:url

创建通过 url 播放的媒资实例。

```
public TXPlayInfoParams(String url)
```

参数说明

参数名	类型	描述
url	String	播放资源地址。

setMediaType

设置媒体类型。

```
public void setMediaType(int mediaType)
```

参数说明

参数名	类型	描述
mediaType	int	设置媒资类型，默认为 AUTO 类型。可选值有： • TXVodConstants#MEDIA_TYPE_AUTO，AUTO 类型（默认值，自适应码率播放暂不支持）。 • TXVodConstants#MEDIA_TYPE_HLS_VOD，HLS 点播媒资。 • TXVodConstants#MEDIA_TYPE_HLS_LIVE，HLS 直播媒资。 • TXVodConstants#MEDIA_TYPE_FILE_VOD，MP4 等通用文件点播媒资（从 11.2 版本开始支持）。 • TXVodConstants#MEDIA_TYPE_DASH_VOD，DASH 点播媒资（从 11.2 版本开始支持）。

setHeaders

自定义配置播放器播放联网过程中携带的 Http header。

```
public void setHeaders(Map<String, String> headers)
```

参数说明

参数名	类型	描述
headers	Map<String, String>	自定义的 Http header 内容。

setEncryptedMp4Level

设置 MP4 加密播放，默认不加密。

```
public void setEncryptedMp4Level(int level)
```

参数说明

参数名	类型	描述

level	int	设置 MP4 播放和存储加密等级，从播放器高级版12.2 版本开始支持，目前支持： • TXVodConstants#MP4_ENCRYPTION_LEVEL_NONE：非加密播放，默认支持。 • TXVodConstants#MP4_ENCRYPTION_LEVEL_L2：MP4 本地加密播放。

setPreferredAudioTrack

设置启播时优先使用的音轨。播放器高级版本 12.3 版本开始支持。
在预下载场景，则配置优先下载音轨。

```
public void setPreferredAudioTrack(String audioTrackName)
```

参数说明

参数名	类型	描述
audioTrackName	String	音轨名称。

TXVodDef

最近更新时间：2025-05-22 16:24:53

常用结构

TXVodSubtitleData

点播播放器回调的字幕文本数据。

参数名	类型	描述
subtitleData	String	字幕内容内容，为空时表示没有字幕内容。
trackIndex	long	当前字幕轨道的 trackIndex。
durationMs	long	字幕持续时间，单位毫秒。暂时为空，请勿使用。
startPositionMs	long	字幕开始时间，单位毫秒。暂时为空，请勿使用。

TXTrackInfo

最近更新时间：2025-05-22 16:24:53

TXTrackInfo 简介

点播播放器播轨道的详细信息。

接口详情

参数名	类型	描述
trackIndex	int	轨道 index。
trackType	int	track类型。取值有： • TXTrackInfo#TX_VOD_MEDIA_TRACK_TYPE_VIDEO：视频轨。 • TXTrackInfo#TX_VOD_MEDIA_TRACK_TYPE_AUDIO：音频轨。 • TXTrackInfo#TX_VOD_MEDIA_TRACK_TYPE_SUBTITLE：字幕轨。
name	String	轨道名字。
isSelected	boolean	当前轨道是否被选中。
isExclusive	boolean	如果是 true，该类型轨道每个时刻只有一条能被选中，如果是 false，该类型轨道可以同时选中多条。
isInternal	boolean	当前的轨道是否是内部原始轨道。

getTrackIndex

获取轨道 index。

```
public int getTrackIndex()
```

getTrackType

获取轨道类型。

```
public int getTrackType()
```

getName

获取轨道名称。

```
public int getName()
```

TXSubtitleRenderModel

最近更新时间：2025-05-22 16:24:53

TXSubtitleRenderModel 简介

点播播放器播字幕样式渲染参数。

字段详情

fontColor

字体颜色，ARGB 格式 如果不设置，默认为白色不透明(0xFFFFFFFF)。

```
public int fontColor
```

fontSize

字体大小。如果设置了 fontSize，则必须设置 canvasWidth 和 canvasHeight，否则内部不知道以什么大小为参考来渲染字体， 如果不设置 fontSize，内部会使用默认的字体大小。

```
public float fontSize
```

familyName

Font family name，Android 默认为"Roboto"，字符串不为空则认为已设置，为空则认为未设置。

```
public String familyName
```

canvasWidth

canvasWidth 和 canvasHeight 是字幕渲染画布的大小， canvasWidth 和 canvasHeight 的比例必须和视频的宽高比一致，否则渲染出的字体会变形。 如果不设置，播放器会取当前视频的大小作为渲染画布的大小。

```
public int canvasWidth
```

canvasHeight

canvasWidth 和 canvasHeight 是字幕渲染画布的大小， canvasWidth 和 canvasHeight 的比例必须和视频的宽高比一致，否则渲染出的字体会变形。 如果不设置，播放器会取当前视频的大小作为渲染画布的大小。

```
public int canvasHeight
```

isBondFontStyle

是否是粗体，默认值为正常字体。

```
public boolean isBondFontStyle
```

outlineWidth

描边宽度，如果不设置，内部会使用默认的描边宽度。

```
public float outlineWidth
```

outlineColor

描边颜色，ARGB 格式 如果不设置，默认为黑色不透明(0xFF000000)。

```
public int outlineColor
```

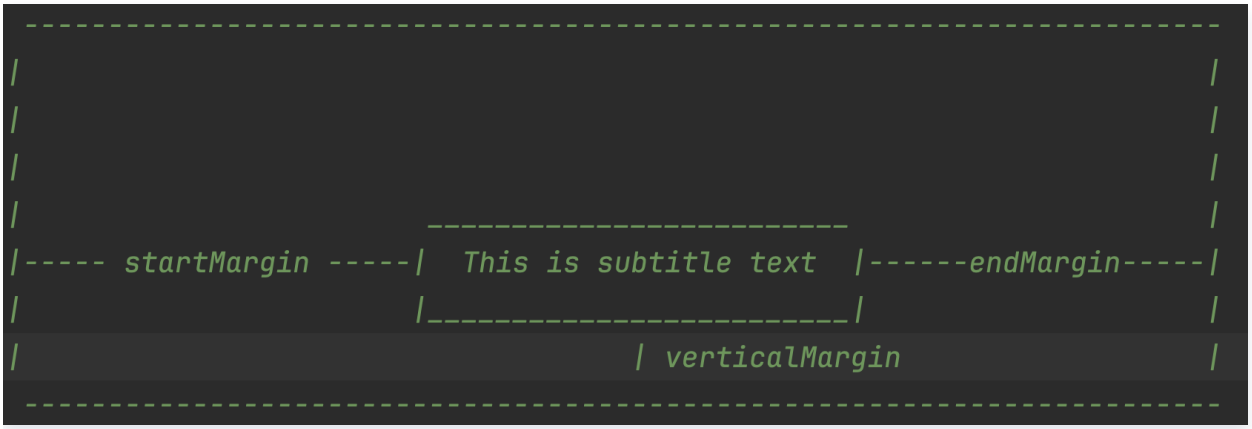
lineSpace

行距，如果设置了 lineSpace，则必须设置 canvasWidth 和 canvasHeight；如果不设置，内部会使用默认的行距。

```
public float lineSpace
```

startMargin

以下 startMargin、endMargin 和 verticalMargin 定义字幕的绘制区域，如果不设置，则使用字幕文件中的设置，如果字幕文件也没有定义，则使用默认的。
注意：一旦设置了 startMargin、endMargin 和 yMargin，而字幕文件也定义了这几个参数的一个或多个，则会覆盖字幕文件中相应的参数。 下面示意图描绘了水平书写方向下这几个参数的意义，请借助每个参数的注释来理解。



沿着字幕文本方向的边距，根据不同的书写方向意义不同。 startMargin 是一个比例值，取值范围[0, 1]，即相对于视频画面大小的比例。

- 对于水平书写方向，startMargin 表示字幕左边距离视频画面左边的距离，比如 startMargin=0.05 则边距为视频宽度的0.05倍（5%）。
- 对于垂直书写方向（无论从右到左还是从左到右），startMargin 表示字幕顶部距离视频画面顶部的距离，比如 startMargin=0.05 则边距为视频高度的0.05倍（5%）

```
public float startMargin
```

endMargin

沿着字幕文本方向的边距。

```
public float endMargin
```

verticalMargin

垂直字幕文本方向的边距。

```
public float verticalMargin
```

TXBitrateItem

最近更新时间：2025-05-22 16:24:53

TXBitrateItem 简介

视频码率信息。

字段详情

参数名	类型	描述
index	int	码率下标，m3u8 文件中的序号。
width	int	此视频流的宽度
height	int	此视频流的高度。
bitrate	int	此视频流的码率。

TXPlayerDrmBuilder

最近更新时间：2025-05-26 18:16:22

TXPlayerDrmBuilder 简介

Drm 播放信息，配合 TXVodPlayer#startPlayDrm 使用。

接口概览

API	描述
TXPlayerDrmBuilder	构造 Drm 播放信息对象。
setDeviceCertificateUrl	设置证书提供商 URL。
setKeyLicenseUrl	设置解密 Key URL。
setPlayUrl	设置播放媒体的 URL。

接口详情

TXPlayerDrmBuilder

构造函数。

```
public TXPlayerDrmBuilder(String licenseUrl, String playUrl)
```

参数说明

参数名	类型	描述
licenseUrl	String	播放媒体 URL。
playUrl	String	解密 Key URL。

setDeviceCertificateUrl

设置证书提供商 URL。

```
public TXPlayerDrmBuilder setDeviceCertificateUrl(String deviceCertificateUrl)
```

参数说明

参数名	类型	描述
deviceCertificateUrl	String	证书提供商 URL。如果是 Widevine，可不填按默认处理。

setKeyLicenseUrl

设置解密 Key URL。

```
public TXPlayerDrmBuilder setKeyLicenseUrl(String keyLicenseUrl)
```

setPlayUrl

设置播放媒体的 URL。

```
public TXPlayerDrmBuilder setPlayUrl(String playUrl)
```

TXImageSprite

最近更新时间：2025-05-26 18:16:22

TXImageSprite 简介

点播雪碧图解析工具类。

接口概览

API	描述
TXImageSprite	构造函数。
setVTTUrlAndImageUrls	设置雪碧图地址。
getThumbnail	获取缩略图。
release	释放资源，使用完毕调用，否则会造成内存泄露。

接口详情

TXImageSprite

构造函数。

```
public TXImageSprite(Context context)
```

setVTTUrlAndImageUrls

设置雪碧图地址， 设置后会启动子线程下载雪碧图并进行解析。

```
public void setVTTUrlAndImageUrls(String vttUrl, List<String> imagesUrl)
```

参数说明

参数名	类型	描述
vttUrl	String	雪碧图 web vtt 描述文件下载 URL。
imagesUrl	List<String>	雪碧图图片下载 URL。

getThumbnail

获取缩略图，获取失败返回 null。

```
public Bitmap getThumbnail(float time)
```

参数说明

参数名	类型	描述
time	float	时间点，单位秒。

release

释放资源，使用完毕调用，否则会造成内存泄露。

```
public void release()
```

Flutter

最近更新时间：2025-06-23 11:53:21

SuperPlayerPlugin类

setGlobalLicense

说明

设置 License

申请到 License 后，通过下面的接口初始化 License，建议在启动的时候进行，如果没有设置 License，将会播放视频失败。

接口

```
static Future<void> setGlobalLicense(String licenceUrl, String licenceKey) async;
```

参数说明

参数名	类型	描述
licenceUrl	String	licence 的 url
licenceKey	String	licence 的 key

返回值

无

createVodPlayer

说明

创建原生层的点播播放器实例，如果使用 `TXVodPlayerController` ，其中已经集成，不需要额外创建。

接口

```
static Future<int?> createVodPlayer() async;
```

参数说明

无

返回值

返回值	类型	描述
playerId	int	播放器 ID

createLivePlayer

说明

创建原生层的点播播放器实例，如果使用 `TXVodPlayerController` ，其中已经集成，不需要额外创建。

接口

```
static Future<int?> createLivePlayer() async;
```

参数说明

无

返回值

返回值	类型	描述
playerId	int	播放器 ID

setConsoleEnabled

说明

打开或关闭播放器原生 log 输出。

接口

```
static Future<int?> setConsoleEnabled() async;
```

参数说明

参数名	类型	描述
enabled	bool	开启或关闭播放器 log

返回值

无

releasePlayer

说明

释放对应播放器的资源。

接口

```
static Future<int?> releasePlayer(int? playerId) async;
```

参数说明

无

返回值

无

setGlobalMaxCacheSize

说明

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。

接口

```
static Future<void> setGlobalMaxCacheSize(int size) async;
```

参数说明

参数名	类型	描述
size	int	最大缓存大小（单位：MB）

返回值

无

setGlobalCacheFolderPath

说明

该缓存路径默认设置到 App 沙盒目录下，参数只需要传递相对缓存目录即可，不需要传递整个绝对路径。

接口

```
static Future<bool> setGlobalCacheFolderPath(String postfixPath) async;
```

参数说明

参数名	类型	描述
postfixPath	String	缓存目录，该缓存路径默认设置到 app 沙盒目录下，postfixPath 只需要传递相对缓存目录即可，不需要传递整个绝对路径。 • Android 平台：视频将会缓存到 sdcard 的 `Android/data/your-pkg-name/files/testCache` 目录。 • iOS 平台视频将会缓存到沙盒的 `Documents/testCache` 目录。

返回值

无

setLogLevel

说明

设置 log 输出级别。

接口

```
static Future<void> setLogLevel(int logLevel) async;
```

参数说明

参数名	类型	描述
logLevel	int	0：输出所有级别的 log 1：输出 DEBUG、INFO、WARNING、ERROR 和 FATAL 级别的 log。 2：输出 INFO、WARNNING、ERROR 和 FATAL 级别的 log。 3：输出 WARNNING、ERROR 和 FATAL 级别的 log。 4：输出 ERROR 和 FATAL 级别的 log。 5：只输出 FATAL 级别的 log 6：不输出任何 SDK log

返回值
无

setBrightness

说明
设置亮度，仅适用于当前 App。
接口

```
static Future<void> setBrightness(double brightness) async;
```

参数说明

参数名	类型	描述
brightness	double	亮度取值范围 0.0~1.0

返回值
无

restorePageBrightness

说明
恢复界面亮度，仅适用于当前 App。
接口

```
static Future<void> restorePageBrightness() async;
```

参数说明
无
返回值
无

getBrightness

说明
获得当前界面的亮度值。
接口

```
static Future<double> getBrightness() async;
```

参数说明
无
返回值

参数名	类型	描述
brightness	double	亮度取值范围 0.0~1.0

setSystemVolume

说明

设置当前系统的音量。

接口

```
static Future<void> setSystemVolume(double volume) async;
```

参数说明

参数名	类型	描述
volume	double	音量取值范围 0.0~1.0

返回值

无

getSystemVolume

说明

设置当前系统的音量。

接口

```
static Future<double> getSystemVolume() async;
```

参数说明

无

返回值说明

参数名	类型	描述
volume	double	音量取值范围 0.0~1.0

abandonAudioFocus

说明

释放音频焦点，仅适用于 Android。

接口

```
static Future<double> abandonAudioFocus() async;
```

参数说明

无

返回值

无

requestAudioFocus

说明
请求获取音频焦点，仅适用于 Android。
接口

```
static Future<void> requestAudioFocus() async ;
```

参数说明
无
返回值
无

isDeviceSupportPip

说明
判断当前设备是否支持画中画模式。
接口

```
static Future<int> isDeviceSupportPip() async;
```

参数说明
无
返回值说明

参数名	类型	描述
isDeviceSupportPip	int	• 0：可开启画中画模式。 • -101：Android 版本过低 。 • -102：画中画权限关闭/设备不支持画中画。 • -103：当前界面已销毁。

getLiteAVSDKVersion

说明
获得当前原生层播放器 SDK 的版本号。
接口

```
static Future<String?> getLiteAVSDKVersion() async;
```

参数说明
无
返回值说明

参数名	类型	描述
sdkVersion	String	当前播放器 SDK 版本

startVideoOrientationService

说明

开始监听设备旋转方向，开启之后，如果设备自动旋转打开，播放器会自动根据当前设备方向来旋转视频方向。

该接口目前只适用 Android 端，iOS 端会自动开启该能力。

注意

在调用该接口前，请务必向用户告知隐私风险。

接口

```
static Future<bool> startVideoOrientationService() async
```

参数说明

无

返回值说明

参数名	类型	描述
result	bool	true 为开启成功，false 为开启失败，如开启过早，还未等到上下文初始化、获取sensor 失败等原因

registerSysBrightness

说明

开启或关闭对于系统亮度的监听，如果开启，当系统亮度发生变化，会改变当前 window 亮度，并回调亮度到 Flutter 层。该接口需配合 setBrightness 和onExtraEventBroadcast 使用。

接口

```
static Future<void>registerSysBrightness(bool isRegister) async
```

参数说明

参数名	类型	描述
isRegister	bool	true：开启监听。 false：关闭监听。

返回值说明

无

setUserId

说明

设置 userId，一般用于控制台数据追踪。

接口

```
static Future<void> setUserId(String userId) async
```

参数说明

参数名	类型	描述
userId	String	用户 userId

返回值说明

无

setSDKListener

说明

设置 sdk 监听，目前支持对 license 设置状态的回调监听。

接口

```
void setSDKListener({FTXLicenceLoadedListener? licenceLoadedListener})
```

参数说明

参数名	类型	描述
licenceLoadedListener	FTXLicenceLoadedListener	参数有 result 和 reason，result 为 0 代表 license 校验成功，result 是校验消息。

返回值说明

无

setLicenseFlexibleValid

说明

开启播放器 License 柔性校验，开启后在播放器首次启动后前 2 次播放校验将默认通过。

接口

```
static Future<void> setLicenseFlexibleValid(bool enabled) async
```

参数说明

参数名	类型	描述
enabled	bool	是否开启柔性校验

返回值说明

无

setDrmProvisionEnv

说明

设置 DRM 证书提供商环境。

接口

```
static Future<void> setDrmProvisionEnv(TXDrmProvisionEnv env) async
```

参数说明

参数名	类型	描述
env	TXDrmProvisionEnv	DRM_PROVISION_ENV_COM: COM 域名证书提供商 DRM_PROVISION_ENV_CN: CN 域名证书提供商

返回值说明

无

TXVodPlayerController类

initialize

说明

初始化 controller，请求分配共享纹理。

> **注意：**
> 12.3.1 以及之后版本不再需要调用。

接口

```
Future<void> initialize({bool? onlyAudio}) async;
```

参数说明

参数名	类型	描述
onlyAudio	bool	选填，是否是纯音频播放器。

返回值说明

无

startVodPlay

注意

10.7版本开始，startPlay 变更为 startVodPlay，需要通过 `{@link SuperPlayerPlugin#setGlobalLicense}` 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和视频播放 Licence 均可使用，若您暂未获取上述 Licence ，可 免费申请测试版 License 以正常播放，正式版 License 需 购买 。

说明

通过播视频 url 进行播放。

接口

```
Future<bool> startVodPlay(String url) async;
```

参数说明

参数名	类型	描述
url	String	要播放的视频 url

返回值说明

参数名	类型	描述
result	bool	创建是否成功

startVodPlayWithParams

注意

10.7版本开始，startPlay 变更为 startVodPlay，需要通过 `{@link SuperPlayerPlugin#setGlobalLicense}` 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和视频播放 Licence 均可使用，若您暂未获取上述 Licence ，可 免费申请测试版 License 以正常播放，正式版 License 需 购买 。

说明

通过视频 field 进行播放。

接口

```
Future<void> startVodPlayWithParams(TXPlayInfoParams params) async;
```

参数说明

参数名	类型	描述
appId	int	应用 appId。必填。
fileId	String	文件 id。必填。
sign	String	防盗链签名，参考 防盗链产品文档 。
url	String	播放链接，该字段与 fileId 填写一个即可。

返回值说明registerSysBrightness

无

pause

说明

暂停当前正在播放的视频。

接口

```
Future<void> pause() async;
```

参数说明

无

返回值说明

无

resume

说明

将当前处于暂停状态的视频恢复播放。

接口

```
Future<void> resume() async;
```

参数说明

无

返回值说明

无

stop

说明

停止当前正在播放的视频。

接口

```
Future<bool> stop({bool isNeedClear = false}) async;
```

参数说明

参数名	类型	描述
isNeedClear	bool	是否清除最后一帧画面

返回值说明

参数名	类型	描述
result	bool	停止是否成功

setIsAutoPlay

说明

设置即将播放的视频，在 startVodPlay 加载视频地址之后，是否直接自动播放。

接口

```
Future<void> setIsAutoPlay({bool? isAutoPlay}) async;
```

参数说明

参数名	类型	描述
isAutoPlay	bool	是否自动播放

返回值说明

无

isPlaying

说明

当前播放器是否正在播放。

接口

```
Future<bool> isPlaying() async;
```

参数说明

无

返回值说明

参数名	类型	描述
isPlaying	bool	是否正在播放

setMute

说明

设置当前播放是否静音。

接口

```
Future<void> setMute(bool mute) async;
```

参数说明

参数名	类型	描述
mute	bool	是否静音

返回值说明

无

setLoop

说明

视频播放完成之后是否循环播放。

接口

```
Future<void> setLoop(bool loop) async;
```

参数说明

参数名	类型	描述
loop	bool	是否循环播放

返回值说明

无

seek

说明

将进度调整到指定位置。

接口

```
_controller.seek(progress);
```

参数说明

参数名	类型	描述
progress	double	需要调整到的播放时间，单位秒。

返回值说明

无

setRate

说明

设置视频播放的速率。

接口

```
Future<void> setRate(double rate) async;
```

参数说明

参数名	类型	描述
rate	double	视频的播放速率。默认1.0。

返回值说明

无

getSupportedBitrates

说明

获得当前正在播放的视频支持的码率。

接口

```
Future<List?> getSupportedBitrates() async;
```

参数说明

无

返回值说明

返回值	类型	描述
index	int	码率序号
width	int	码率对应视频宽度
height	int	码率对应视频高度
bitrate	int	码率值

getBitrateIndex

说明

获得设置过的码率序号。

接口

```
Future<int> getBitrateIndex() async;
```

参数说明

无

返回值说明

返回值	类型	描述
index	int	码率序号

setBitrateIndex

说明

通过码率序号，设置当前码率。

接口

```
Future<void> setBitrateIndex(int index) async;
```

参数说明

返回值	类型	描述
index	int	码率序号。传入-1时，表示开启码流自适应。

返回值说明

无

setStartTime

说明

指定播放开始时间。

接口

```
Future<void> setStartTime(double startTime) async;
```

参数说明

返回值	类型	描述
startTime	double	播放开始时间，单位秒。

返回值说明

无

setAudioPlayoutVolume

说明

设置视频的声音大小。

接口

```
Future<void> setAudioPlayoutVolume(int volume) async;
```

参数说明

参数名	类型	描述
volume	int	视频声音大小，范围0~100。

返回值说明

无

setRequestAudioFocus

说明

请求获得音频焦点，适用于 Android。

接口

```
Future<bool> setRequestAudioFocus(bool focus) async;
```

参数说明

参数名	类型	描述
focus	bool	是否设置焦点

返回值说明

参数名	类型	描述
result	bool	设置焦点是否成功

setConfig

说明
给播放器进行配置。
接口

```
Future<void> setConfig(FTXVodPlayConfig config) async ;
```

参数说明

参数名	类型	描述
config	FTXVodPlayConfig	请参考 `FTXVodPlayConfig`类

返回值说明
无

getCurrentPlaybackTime

说明
获得当前播放时间，单位 秒。
接口

```
Future<double> getCurrentPlaybackTime() async;
```

参数说明
无
返回值说明

参数名	类型	描述
playbackTime	double	当前播放时间，单位秒。

getBufferDuration

说明
获得当前视频已缓存的时间，单位 秒。
接口

```
Future<double> getBufferDuration();
```

参数说明
无
返回值说明

参数名	类型	描述
playbackTime	double	当前视频已缓存的时间，单位秒。

getPlayableDuration

说明

获得当前正在播放视频的可播放时间，单位 秒。

接口

```
Future<double> getPlayableDuration() async;
```

参数说明

无

返回值说明

参数名	类型	描述
playableDuration	double	当前视频可播放时，单位秒。

getWidth

说明

获得当前正在播放视频的宽度。

接口

```
Future<int> getWidth() async;
```

参数说明

无

返回值说明

参数名	类型	描述
width	int	当前视频宽度

getHeight

说明

获得当前正在播放视频的高度。

接口

```
Future<int> getHeight() async;
```

参数说明

无

返回值说明

参数名	类型	描述
height	int	当前视频高度

setToken

说明

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。

接口

```
Future<void> setToken(String? token) async;
```

参数说明

参数名	类型	描述
token	String	播放视频的 token

返回值说明

无

isLoop

说明

获得当前播放器是否循环播放的状态。

接口

```
Future<bool> isLoop() async;
```

参数说明

无

返回值说明

参数名	类型	描述
isLoop	bool	播放器是否处于循环播放状态

enableHardwareDecode

说明

开启/关闭硬解播放，设置后不会立即生效，需要重新播放才可生效。

接口

```
Future<bool> enableHardwareDecode(bool enable);
```

参数说明

参数名	类型	描述
enable	bool	是否开启硬解

返回值说明

参数名	类型	描述

result	bool	硬解/软解设置结果

dispose

说明
销毁 controller，调用该方法会销毁掉所有通知事件，释放掉播放器。
接口

```
Future<void> dispose() async;
```

参数说明
无
返回值说明
无

getDuration

说明
获取视频总时长。
接口

```
Future<double> getDuration() async;
```

参数说明
无
返回值说明

参数名	类型	描述
duration	double	视频总时长，单位 秒

enterPictureInPictureMode

说明
进入画中画模式。
接口

```
Future<int> enterPictureInPictureMode({String? backIconForAndroid, String?
playIconForAndroid, String? pauseIconForAndroid, String? forwardIconForAndroid})
async;
```

参数说明
该参数只适用于 Android 平台。

参数名	类型	描述
backIcon	String	回退按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统

		自带图标。
playIcon	String	播放按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。
pauseIcon	String	暂停按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。
forwardIcon	String	快进按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。

返回值说明

参数名	值	描述
NO_ERROR	0	启动成功，没有错误。
ERROR_PIP_LOWER_VERSION	-101	Android 版本过低，不支持画中画模式。
ERROR_PIP_DENIED_PERMISSION	-102	画中画模式权限未打开，或者当前设备不支持画中画。
ERROR_PIP_ACTIVITY_DESTROYED	-103	当前界面已经销毁。
ERROR_IOS_PIP_DEVICE_NOT_SUPPORT	-104	设备或系统版本不支持（iPad iOS9+ 才支持PIP），只适用于 iOS。
ERROR_IOS_PIP_PLAYER_NOT_SUPPORT	-105	播放器不支持，只适用于 iOS。
ERROR_IOS_PIP_VIDEO_NOT_SUPPORT	-106	视频不支持，只适用于 iOS。
ERROR_IOS_PIP_IS_NOT_POSSIBLE	-107	PIP 控制器不可用，只适用于 iOS。
ERROR_IOS_PIP_FROM_SYSTEM	-108	PIP 控制器报错，只适用于 iOS。
ERROR_IOS_PIP_PLAYER_NOT_EXIST	-109	播放器对象不存在，只适用于 iOS。
ERROR_IOS_PIP_IS_RUNNING	-110	PIP 功能已经运行，只适用于 iOS。
ERROR_IOS_PIP_NOT_RUNNING	-111	PIP 功能没有启动，只适用于 iOS。
ERROR_IOS_PIP_START_TIME_OUT	-112	PIP 启动超时，只适用于 iOS。
ERROR_PIP_AUTH_DENIED	-201	权限不足，目前只出现在直播画中画，适用于 iOS。
ERROR_PIP_CAN_NOT_ENTER	-120	PIP 错误，当前不能进入 PIP 模式。

initImageSprite

说明

初始化视频雪碧图。
接口

```
Future<void> initImageSprite(String? vvtUrl, List<String>? imageUrls) async;
```

参数说明

参数名	类型	描述
vvtUrl	String	雪碧图 web vtt 描述文件下载 URL。
imageUrls	List<String>	雪碧图图片下载 URL。

返回值说明
无

getImageSprite

说明
获取加载的雪碧图。
接口

```
Future<Uint8List?> getImageSprite(double time) async;
```

参数说明

参数名	类型	描述
time	double	时间点，单位秒。

返回值说明

参数名	类型	描述
thumb	Uint8List	雪碧图

exitPictureInPictureMode

说明
退出画中画，如果该播放器处于画中画模式。
接口

```
Future<void> exitPictureInPictureMode() async;
```

参数说明
无
返回值说明
无

addSubtitleSource

说明

添加外挂字幕。

注意：此功能需要播放器高级版 11.7 版本开始支持。

接口

```
Future<void> addSubtitleSource(String url, String name, {String? mimeType}) async;
```

参数说明

参数名	类型	描述
url	String	字幕 url
name	String	字幕名称
mimeType	String	字幕类型，支持 SRT（TXVodPlayEvent.VOD_PLAY_MIMETYPE_TEXT_SRT）和 VTT（TXVodPlayEvent.VOD_PLAY_MIMETYPE_TEXT_VTT）格式。

getSubtitleTrackInfo

说明

返回字幕轨道信息列表。

注意：此功能需要播放器高级版 11.7 版本开始支持。

接口

```
Future<List<TXTrackInfo>> getSubtitleTrackInfo() async;
```

参数说明

TXTrackInfo类：

参数名	类型	描述
trackType	int	轨道类型。取值有： • 视频轨：TX_VOD_MEDIA_TRACK_TYPE_VIDEO = 1 • 音频轨： TX_VOD_MEDIA_TRACK_TYPE_AUDIO = 2 • 字幕轨：TX_VOD_MEDIA_TRACK_TYPE_SUBTITLE = 3
trackIndex	int	轨道 index
name	String	轨道名字
isSelected	bool	当前轨道是否被选中。

isExclusive	bool	如果是 true，该类型轨道每个时刻只有一条能被选中，如果是 false，该类型轨道可以同时选中多条。
isInternal	bool	当前的轨道是否是内部原始轨道。

getAudioTrackInfo

说明

返回字幕轨道信息列表。

注意：此功能需要播放器高级版 11.7 版本开始支持。

接口

```
Future<List<TXTrackInfo>> getAudioTrackInfo() async;
```

参数说明

参考TXTrackInfo类

selectTrack

说明

选择轨道。

注意：此功能需要播放器高级版 11.7 版本开始支持。

接口

```
Future<void> selectTrack(int trackIndex) async;
```

参数说明

参数名	类型	描述
trackIndex	int	轨道 index，trackIndex 轨道 index，通过[TXTrackInfo]的 trackIndex 获取。

deselectTrack

说明

取消选择轨道。

注意：此功能需要播放器高级版 11.7 版本开始支持。

接口

```
Future<void> deselectTrack(int trackIndex) async;
```

参数说明

参数名	类型	描述
trackIndex	int	轨道 index，trackIndex 轨道 index，通过[TXTrackInfo]的 trackIndex 获取。

setStringOption

说明

设置扩展参数

接口

```
Future<void> setStringOption(String key, Object value) async
```

参数说明

参数名	类型	描述
key	String	扩展参数键值。
value	Object	扩展参数值。

setPlayerView

说明

绑定视频渲染纹理

接口

```
Future<void>setPlayerView(int renderViewId) async
```

参数说明

参数名	类型	描述
renderViewId	int	TXPlayerVideo 的 onRenderViewCreatedListener 回调返回的 viewId。

setRenderMode

说明

设置画面平铺模式。

接口

```
Future<void> setRenderMode(FTXPlayerRenderMode renderMode) async
```

参数说明

参数名	类型	描述
renderMode	FTXPlayerRender	设置画面平铺模式，一种是按照视频比例优先展示完整画面 `FTXPlayerRenderMode.ADJUST_RESOLUTION` ，另外一种是按照视频比例，铺满容器 `FTXPlayerRenderMode.FULL_FILL_CONTAINER` 。

	Mode	

reDraw

说明

强制重绘当前画面，仅对 Android 生效。

接口

```
Future<void> reDraw() async
```

FTXVodPlayConfig类

属性配置说明

参数名	类型	描述
connectRetryCount	int	播放器重连次数，当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过该值设置 SDK 重连次数。
connectRetryInterval	int	播放器重连间隔，当 SDK 与服务器异常断开连接时，SDK 会尝试与服务器重连，通过该值设置两次重连间隔时间。
timeout	int	播放器连接超时时间。
playerType	int	播放器类型。0：点播，1：直播，2：直播回看。
headers	Map	自定义 http headers。
enableAccurateSeek	bool	是否精确 seek，默认 true。
autoRotate	bool	播放 mp4 文件时，若设为 true 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 true。
smoothSwitchBitrate	bool	平滑切换多码率 HLS，默认 false。设为 false 时，可提高多码率地址打开速度；设为 true，在 IDR 对齐时可平滑切换码率。
cacheMp4ExtName	String	缓存 mp4 文件扩展名，默认 mp4。
progressInterval	int	设置进度回调间隔，若不设置，SDK 默认间隔0.5秒回调一次，单位毫秒。
maxBufferSize	int	最大播放缓冲大小，单位 MB。此设置会影响 playableDuration，设置越大，提前缓存的越多。
maxPreloadSize	int	预加载最大缓冲大小，单位：MB。
firstStartPlayBufferTime	int	首缓需要加载的数据时长，单位 ms，默认值为100ms。
nextStartPlayBufferTime	int	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	String	HLS 安全加固加解密 key。
overlayIv	String	HLS 安全加固加解密 IV。
extInfoMap	Map	一些不必周知的特殊配置。
enableRenderProcess	bool	是否允许加载后渲染后处理服务，默认开启，开启后超分插件如果存在，默认加载。
preferredResolution	int	优先播放的分辨率，preferredResolution = width * height。
mediaType	int	设置媒资类型, 默认为 auto 类型。可选值有 • TXVodConstants#MEDIA_TYPE_AUTO，AUTO 类型（默认值，自适应码率播放暂不支持）。 • TXVodConstants#MEDIA_TYPE_HLS_VOD，HLS 点播媒资。 • TXVodConstants#MEDIA_TYPE_HLS_LIVE，HLS 直播媒资。 • TXVodConstants#MEDIA_TYPE_FILE_VOD，MP4 等通用文件点播媒资（从 11.7 版本开始支持）。 • TXVodConstants#MEDIA_TYPE_DASH_VOD，DASH 点播媒资（从 11.7 版本开始支持）。

TXLivePlayerController类

initialize

说明

初始化 controller，请求分配共享纹理。

> ⚠ **注意：**
> 12.3.1 以及之后版本不再需要调用。

接口

```
Future<void> initialize({bool? onlyAudio}) async;
```

参数说明

参数名	类型	描述
onlyAudio	bool	选填，是否是纯音频播放器

返回值说明

无

startLivePlay

注意

10.7版本开始，startPlay 变更为 startLivePlay，需要通过 {@link SuperPlayerPlugin#setGlobalLicense} 设置 Licence 后方可成功播放， 否则将播放失败（黑屏），全局仅设置一次即可。直播 Licence、短视频 Licence 和视频播放

Licence 均可使用，若您暂未获取上述 Licence ，可 免费申请测试版 License 以正常播放，正式版 License 需 购买 。

说明

通过播视频 url 进行播放。

接口

```
Future<bool> play(String url) async;
```

参数说明

参数名	类型	描述
url	String	要播放的视频 url

返回值说明

参数名	类型	描述
result	bool	创建是否成功

pause

说明

暂停当前正在播放的视频。

接口

```
Future<void> pause() async;
```

参数说明

无

返回值说明

无

resume

说明

将当前处于暂停状态的视频恢复播放。

接口

```
Future<void> resume() async;
```

参数说明

无

返回值说明

无

stop

说明

停止当前正在播放的视频。

接口

```
Future<bool> stop({bool isNeedClear = false}) async;
```

参数说明

参数名	类型	描述
isNeedClear	bool	是否清除最后一帧画面

返回值说明

参数名	类型	描述
result	bool	停止是否成功

isPlaying

说明

当前播放器是否正在播放。

接口

```
Future<bool> isPlaying() async;
```

参数说明

无

返回值说明

参数名	类型	描述
isPlaying	bool	是否正在播放

setMute

说明

设置当前播放是否静音。

接口

```
Future<void> setMute(bool mute) async;
```

参数说明

参数名	类型	描述
mute	bool	是否静音

返回值说明

无

setVolume

说明

设置视频的声音大小。

接口

```
Future<void> setVolume(int volume);
```

参数说明

参数名	类型	描述
volume	int	视频声音大小，范围0~100

返回值说明

无

setLiveMode

说明

设置直播模式。

接口

```
Future<void> setLiveMode(TXPlayerLiveMode mode) async;
```

参数说明

参数名	类型	描述
mode	int	直播模式。自动模式、极速模式、流畅模式

返回值说明

无

setAppID

说明

设置 appID，云控使用。

接口

```
Future<void> setAppID(int appId) async;
```

参数说明

参数名	类型	描述
appId	int	appId

返回值说明

无

enableHardwareDecode

说明

开启/关闭硬解播放，设置后不会立即生效，需要重新播放才可生效。

接口

```
Future<bool> enableHardwareDecode(bool enable);
```

参数说明

参数名	类型	描述
enable	bool	是否开启硬解

返回值说明

参数名	类型	描述
result	bool	硬解/软解设置结果

enterPictureInPictureMode

说明

进入画中画模式，仅支持 Android 端，iOS 端直播目前暂不支持画中画模式。

接口

```
Future<int> enterPictureInPictureMode({String? backIconForAndroid, String?
playIconForAndroid, String? pauseIconForAndroid, String? forwardIconForAndroid})
async;
```

参数说明

该参数只适用于 Android 平台

参数名	类型	描述
backIcon	String	回退按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。
playIcon	String	播放按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。
pauseIcon	String	暂停按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。
forwardIcon	String	快进按钮图标，由于 Android 平台限制，图标大小不得超过1M，可不传，不传则使用系统自带图标。

返回值说明

参数名	值	描述
NO_ERROR	0	启动成功，没有错误。
ERROR_PIP_LOWER_VERSION	-101	Android 版本过低，不支持画中画模式。
ERROR_PIP_DENIED_PERMISSION	-102	画中画模式权限未打开，或者当前设备不支持画中画。
ERROR_PIP_ACTIVITY_DESTROYED	-103	当前界面已经销毁。
ERROR_IOS_PIP_DEVICE_NOT_SUPPORT	-104	设备或系统版本不支持（iPad iOS9+ 才支持PIP），只适用于 iOS。
ERROR_IOS_PIP_PLAYER_NOT_SUPPORT	-105	播放器不支持，只适用于 iOS。
ERROR_IOS_PIP_VIDEO_NOT_SUPPORT	-106	视频不支持，只适用于 iOS。
ERROR_IOS_PIP_IS_NOT_POSSIBLE	-107	PIP 控制器不可用，只适用于 iOS。
ERROR_IOS_PIP_FROM_SYSTEM	-108	PIP 控制器报错，只适用于 iOS。
ERROR_IOS_PIP_PLAYER_NOT_EXIST	-109	播放器对象不存在，只适用于 iOS。
ERROR_IOS_PIP_IS_RUNNING	-110	PIP 功能已经运行，只适用于 iOS。
ERROR_IOS_PIP_NOT_RUNNING	-111	PIP 功能没有启动，只适用于 iOS。

dispose

说明

销毁 controller，调用该方法会销毁掉所有通知事件，释放掉播放器。

接口

```
Future<void> dispose() async;
```

参数说明

无

返回值说明

无

switchStream

说明

切换播放流。

接口

```
Future<int> switchStream(String url) async;
```

参数说明

参数名	类型	描述
url	String	需要切换的视频源

返回值说明

参数名	类型	描述
result	int	切换结果

enableReceiveSeiMessage

说明

开启接收 SEI 消息。

接口

```
Future<int> enableReceiveSeiMessage(bool isEnabled, int payloadType) async
```

参数说明

参数名	类型	描述
isEnabled	bool	是否开启接收 SEI 消息，默认关闭。
payloadType	int	指定接收 SEI 消息的 payloadType，支持 5、242、243，请与发送端的 payloadType 保持一致。

返回值说明

参数名	类型	描述
result	int	开启结果

showDebugView

说明

显示调试信息图层。

接口

```
Future<void> showDebugView(bool isShow) async
```

参数说明

参数名	类型	描述
isShow	bool	是否显示调试图层

setProperty

说明

调用 V2TXLivePlayer 的高级 API 接口。

接口

```
Future<int> setProperty(String key, Object value) async
```

参数说明

参数名	类型	描述
key	String	扩展参数键值。
value	Object	扩展参数值。

返回值说明

参数名	类型	描述
result	int	设置结果

getSupportedBitrate

说明

获取码流信息。

接口

```
Future<List<FSteamInfo>> getSupportedBitrate() async
```

setCacheParams

说明

设置播放器缓存自动调整的最小和最大时间（单位：秒）。

接口

```
Future<int> setCacheParams(double minTime, double maxTime) async
```

参数说明

参数名	类型	描述
minTime	double	缓存自动调整最小时间。

maxTime	double	缓存自动调整最大时间。

返回值说明

参数名	类型	描述
result	int	设置结果

setPlayerView

说明

绑定视频渲染纹理

接口

```
Future<void>setPlayerView(int renderViewId) async
```

参数说明

参数名	类型	描述
renderViewId	int	TXPlayerVideo 的 onRenderViewCreatedListener 回调返回的 viewId。

setRenderMode

说明

设置画面平铺模式。

接口

```
Future<void> setRenderMode(FTXPlayerRenderMode renderMode) async
```

参数说明

参数名	类型	描述
renderMode	FTXPlayerRenderMode	设置画面平铺模式，一种是按照视频比例优先展示完整画面 `FTXPlayerRenderMode.ADJUST_RESOLUTION` ，另外一种是按照视频比例，铺满容器 `FTXPlayerRenderMode.FULL_FILL_CONTAINER` 。

FTXLivePlayConfig类

属性配置说明

参数名	类型	描述

maxAutoAdjustCacheTime	double	播放器缓存自动调整的最大时间，单位秒，取值需要大于0，默认值为5。
minAutoAdjustCacheTime	double	播放器缓存自动调整的最小时间，单位秒，取值需要大于0，默认值为1。
connectRetryCount	int	播放器遭遇网络连接断开时 SDK 默认重试的次数，取值范围1 – 10，默认值：3。
connectRetryInterval	int	网络重连的时间间隔，单位秒，取值范围3 – 30，默认值：3。

TXVodDownloadController类

startPreLoad

说明

启动预下载。启动预下载前，请先设置好播放引擎的缓存目录`[SuperPlayerPlugin.setGlobalCacheFolderPath]`和缓存大小`[SuperPlayerPlugin.setGlobalMaxCacheSize]`，这个设置是全局配置需和播放器保持一致，否则会造成播放缓存失效。

接口

```
Future<int> startPreLoad(
final String playUrl,
final int preloadSizeMB,
final int preferredResolution, {
FTXPredownlodOnCompleteListener? onCompleteListener,
    FTXPredownlodOnErrorListener? onErrorListener,
}) async
```

```
Future<void> startPreload(TXPlayInfoParams txPlayInfoParams,
    final int preloadSizeMB,
    final int preferredResolution, {
      FTXPredownlodOnCompleteListener? onCompleteListener,
      FTXPredownlodOnErrorListener? onErrorListener,
      FTXPredownlodOnStartListener? onStartListener,
    }) async
```

参数说明

参数名	类型	描述
playUrl	String	要预下载的 url。
preloadSizeMB	int	预下载的大小（单位：MB）。
preferredResolution	int	期望分辨率，值为高x宽。可参考如720*1080。不支持多分辨率或不需指定时，传-1。

onCompleteListener	FTXPredownlodOnCompleteListener?	预下载成功回调，全局。
onErrorListener	FTXPredownlodOnErrorListener	预下载失败回调，全局。

TXPlayInfoParams:

参数名	类型	描述
appId	int	应用appId。必填。
fileId	String	文件id。必填。
url	String	视频 url，与 fileId 只用填写一个，如果都填写，url 优先。
sign	String	防盗链签名，参考 防盗链产品文档 。

返回值说明

参数名	类型	描述
taskId	int	任务 ID

stopPreLoad

说明

停止预下载。

接口

```
Future<void> stopPreLoad(final int taskId) async
```

参数说明

参数名	类型	描述
taskId	int	任务 ID

返回值说明

无

startDownload

说明

开始下载视频。

接口

```
Future<void> startDownload(TXVodDownloadMediaInfo mediaInfo) async
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	下载任务信息

TXVodDownloadMediaInfo

参数名	类型	描述
playPath	String?	缓存地址，获得到的视频缓存会有该值，启动下载可以不赋值。
progress	double?	缓存进度，获得到的视频缓存会有该值，启动下载可以不赋值。
downloadState	int?	缓存状态，获得到的视频缓存会有该值，启动下载可以不赋值。
userName	String?	下载账户名称，用于区分不同账户的下载，传空则为 default。 不建议设置字符串长度过长。
duration	int?	缓存视频总时长，Android 端单位为毫秒，iOS 为秒，获得到的视频缓存会有该值，启动下载可以不赋值。
playableDuration	int?	视频已缓存时长，Android 端单位为毫秒，iOS 为秒，获得到的视频缓存会有该值，启动下载可以不赋值。
size	int?	文件总大小，单位：byte。获得到的视频缓存会有该值，启动下载可以不赋值。
downloadSize	int?	文件已下载的大小，单位：byte。获得到的视频缓存会有该值，启动下载可以不赋值。
url	String?	需要下载的视频 url，url 下载必填，不支持嵌套 m3u8 和 mp4 下载。
dataSource	TXVodDownloadDataSource?	需要下载的视频 fileId 信息，url 与该参数可只使用一个。
speed	int?	下载速度，单位：KByte/秒。
isResourceBroken	bool?	资源是否已损坏，如：资源被删除了。

TXVodDownloadDataSource

参数名	类型	描述
appId	int?	下载文件对应的 appId，必填。
fileId	String?	下载文件 Id，必填。
pSign	String?	加密签名，加密视频必填。
quality	int?	清晰度 ID，必传。

token	String?	加密 token
userName	String?	下载账户名称，用于区分不同账户的下载，传空则为 default。 不建议设置字符串长度过长。

返回值说明

无

stopDownload

说明

停止下载。

接口

```
Future<void> stopDownload(TXVodDownloadMediaInfo mediaInfo) async
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	任务信息

返回值说明

无

setDownloadHeaders

说明

设置下载任务请求头。

接口

```
Future<void> setDownloadHeaders(Map<String, String> headers) async
```

参数说明

参数名	类型	描述
headers	Map<String, String>	请求头信息

返回值说明

无

getDownloadList

说明

获得所有下载任务，包括已下载、正在下载以及下载错误的任务。

接口

```
Future<List<TXVodDownloadMediaInfo>> getDownloadList() async
```

参数说明

无

返回值说明

参数名	类型	描述
mediaInfoList	List<TXVodDownloadMediaInfo>	任务列表，可通过对比 userName 来区分不同用户的下载

getDownloadInfo

说明

获得下载任务信息。

接口

```
Future<TXVodDownloadMediaInfo> getDownloadInfo(TXVodDownloadMediaInfo mediaInfo)
async
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	任务信息

返回值说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	缓存任务详情信息

setDownloadObserver

说明

获得下载任务信息。

接口

```
void setDownloadObserver(FTXDownloadOnStateChangeListener
downloadOnStateChangeListener, FTXDownloadOnErrorListener downloadOnErrorListener)
```

参数说明

参数名	类型	描述
downloadOnStateChangeListener	FTXDownloadOnStateChangeListener	任务下载状态回调

downloadOnErrorListener	FTXDownloadOnErrorListener	任务下载错误回调

返回值说明

无

deleteDownloadMediaInfo

说明

删除下载的视频。

接口

```
Future<bool> deleteDownloadMediaInfo(TXVodDownloadMediaInfo mediaInfo) async
```

参数说明

参数名	类型	描述
mediaInfo	TXVodDownloadMediaInfo	任务下载信息

返回值说明

参数名	类型	描述
result	bool	删除结果

第三方播放器插件
第三方播放器 iOS 插件

最近更新时间：2025-05-27 14:52:42

插件信息

插件名称	第三方播放器 iOS 插件
版本号	V1.4.0
插件介绍	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发的对接云 PaaS 资源的播放器插件，以及常用于有自定义播放器功能需求的用户。
开发者	深圳市腾讯计算机系统有限公司
合规使用说明	第三方播放器插件合规使用指南
个人信息处理规则	第三方播放器插件隐私保护指引
下载 SDK	• 第三方播放器 iOS 插件和 Demo 项目，请参见 TXCPlayerAdapterSDK_iOS 。 • 更新情况可查看 更新日志 。

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：

1.1 客户端视频发布后，服务器会返回 fileId 到客户端。

1.2 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];
```

使用图片解密

经过加密的图片需要经过解密后才能正常使用。

```
//创建TXCPlayerAdapter，使用图片解密时，AppId传入0
```

```
TXCPlayerAdapter *adapter = [TXCPlayerAdapter shareAdapterWithAppId:0];
```

通过 getImageLocalUrl 接口获取 imageUrl 的 LocalUrl。

```
//创建assistor
id<ITXCPlayerAssistorProtocol> assistor = [TXCPlayerAdapter createPlayerAssistor];
//把imageUrl转换成LocalUrl
NSString *localUrl = [assistor getImageLocalUrl:ImageUrl];
```

使用 LocalUrl

```
//显示图片
[_imageView sd_setImageWithURL:[NSURL URLWithString:localUrl]];
```

SDK 接口说明

初始化 Adapter

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，如果是图片解密，则填0）。

销毁 Adapter

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	播放器签名。

创建图片解密辅助类

通过图片辅助类可以获取加密图片的 localUrl。
接口

```
+ (id<ITXCPlayerAssistorProtocol>)createPlayerAssistor;
```

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。

获取图片的 localUrl

接口

```
/// 请求图片的localUrl接口
- (NSString *)getImageLocalUrl:(NSString *)imageURL;
```

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;
```

参数说明

参数名	类型	描述
imageURL	NSString	图片Url

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。

第三方播放器 Android 插件

最近更新时间：2025-05-27 14:52:42

插件信息

插件名称	第三方播放器 Android 插件
版本号	V1.4.1
插件介绍	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发的对接云 PaaS 资源的播放器插件，以及常用于有自定义播放器功能需求的用户。
开发者	深圳市腾讯计算机系统有限公司
合规使用说明	第三方播放器插件合规使用指南
个人信息处理规则	第三方播放器插件隐私保护指引
下载 SDK	• 第三方播放器 Android 插件和 Demo 项目下载地址 TXCPlayerAdapterSDK_Android 。 • 更新情况可查看 更新日志 。

集成指引

SDK 集成

集成 SDK，拷贝 `aar` 到 libs 目录，添加依赖项：

```
implementation(name:'TXCPlayerAdapter-release-1.4.1', 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。点开后在右侧视频详情中，可以看到相关参数。

```
//psign 即播放器签名，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();
```

使用图片解密

经过加密的图片需要经过解密后才能正常使用。

1. 创建图片解密辅助类。

```
ITXCPlayerAssistor assistor = TXCPlayerAdapter.createPlayerAssistor();
```

2. 调用 getImageLocalUrl 方法获取本地代理访问 Url。

```
String imageLocalUrl = assistor.getImageLocalUrl(imgUrl);
```

3. 获取到图片的 localUrl 后，可以直接传给 Glide 等图片加载器加载图片。

```
// 使用Glide加载图片
Glide.with(context).load(imageLocalUrl).into(imageView);
```

SDK 接口说明

初始化 TXCPlayerAdapter

初始化 Adapter（每次）。

接口

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

参数说明

appId：填写 appid（如果使用了新建应用，则填 subappid，如果为图片解密，则无需调用此方法）。

销毁 TXCPlayerAdapter

销毁 Adapter，当程序退出后调用。

接口

```
TXCPlayerAdapter.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

```
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	播放器签名。

创建图片解密辅助类

创建图片解密辅助类可以获取本地代理访问 Url，通过本地代理访问即可实现图片解密。

接口

```
TXCPlayerAdapter.createPlayerAssistor();
```

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

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

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

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

参数说明

参数名	类型	描述
callback	ITXCRequestVideoInfoCallback	异步回调函数。

获取加密图片的本地代理解密 Url

接口

```
String getImageLocalUrl(String imgUrl);
```

参数说明

参数名	类型	描述
imgUrl	String	图片网络 Url

获取视频的基本信息

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
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

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 `playerAssistor.requestPlayInfo` 回调之后才生效。

接口

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

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 `playerAssistor.requestPlayInfo` 回调之后才生效。

接口

```
TXCImageSpriteInfo playerAssistor.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。

第三方播放器 Web 插件

最近更新时间：2025-05-27 14:52:42

本文档是介绍第三方播放器 Web 插件，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。本插件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。

SDK 集成

第三方播放器 Web 插件提供 **CDN 集成**和 **npm 集成**两种集成方式：

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>
```

npm 集成

```
// npm install
npm install tcadapter --save

// 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>
```

SDK 使用说明

检测开发环境

检测当前环境是否支持 TcAdapter。

```
TcAdapter.isSupported();
```

初始化 Adapter

初始化 Adapter，创建 Adapter 实例。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	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。

> **注意：**
> 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。

获取视频基本信息

获取视频的信息， 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。
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

接口

```
List<StreamingOutput> adapter.getStreamimgOutputList();
```

参数说明

StreamingOutput 参数如下：

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

接口

```
List<KeyFrameDescInfo> adapter.getKeyFrameDescInfo();
```

参数说明

KeyFrameDescInfo 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

接口

```
ImageSpriteInfo adapter.getImageSpriteInfo();
```

参数说明

ImageSpriteInfo 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。

监听事件

播放器可以通过初始化返回的对象进行事件监听，示例：

```
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

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：**
> 具体请参见 hls.js 。

示例

例1：在 React 中使用 TcAdapter

具体示例，请参见 GitHub 。

```
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.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NTg5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNTY1MzYyMyJ9fQ.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;
```

例2：TcAdapter 与 videojs 结合

具体示例，请参见 GitHub 。

```
// 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.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NT
g5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFtcC
I6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNT
Y1MzYyMyJ9fQ.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;
```