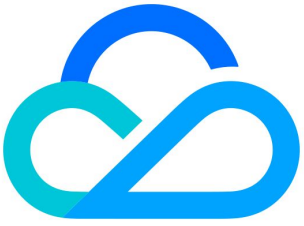
播放器 SDK

含 UI 的集成方案



腾讯云

文档目录

含 UI 的集成方案
Web
TCPlayer 集成指引

最近更新时间：2025-05-09 11:27:31

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：

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 127.0.01，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，比如自动切换自适应码流、预览视频缩略图、添加视频打点信息等。这些功能在 播放长视频方案 中有详细的说明，可以参考文档实现。
此外，播放器还提供更多其他功能，功能列表和使用方法请参见 功能展示 页面。

TCPlayer 清晰度配置说明

最近更新时间：2024-10-10 15:06:31

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：2024-10-10 15:06:31

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（原超级播放器）

最近更新时间：2025-06-04 17:40:01

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 。

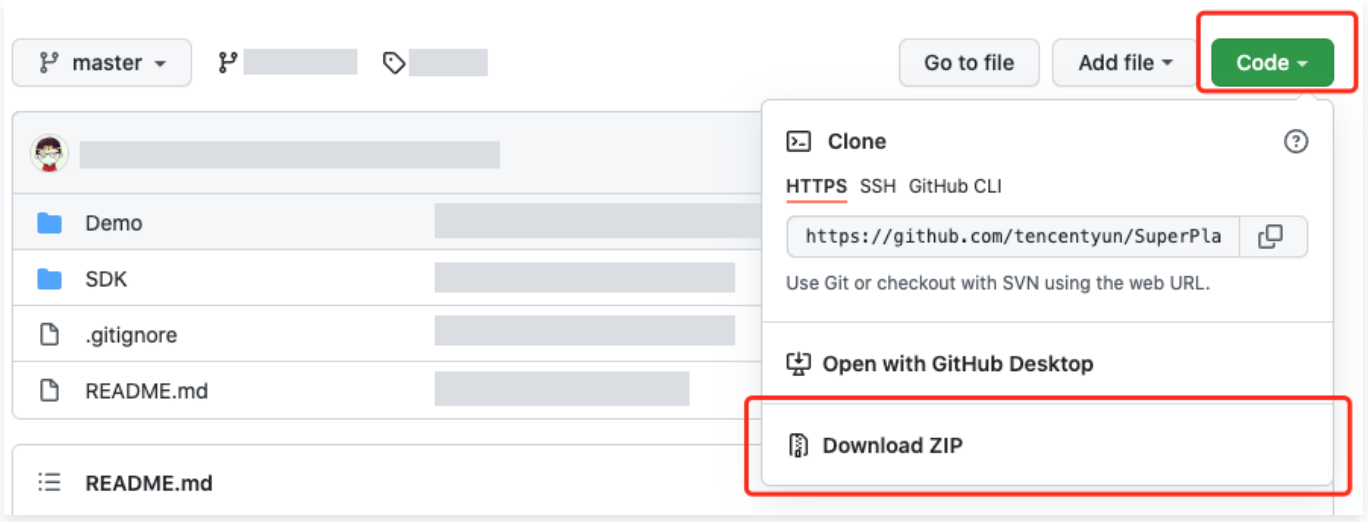
集成准备

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'
```

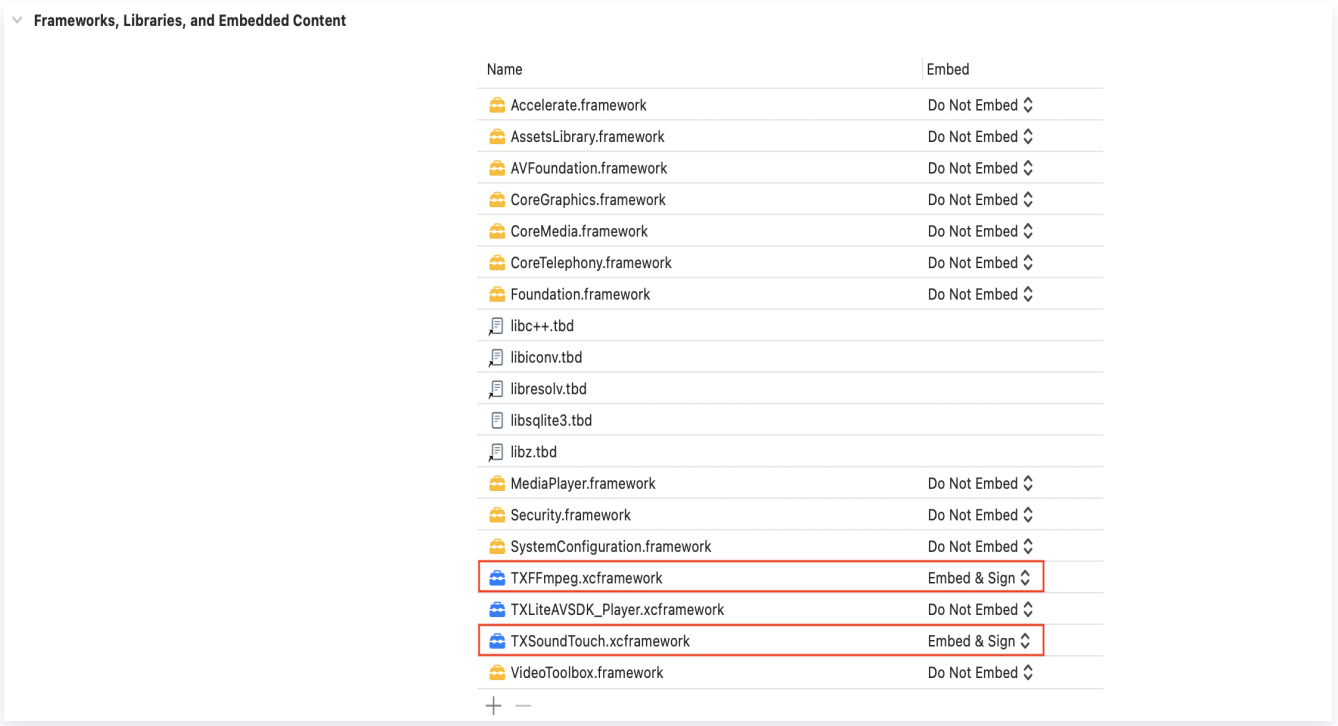
2. 执行 `pod install` 或 `pod update` 。

手动下载 SDK

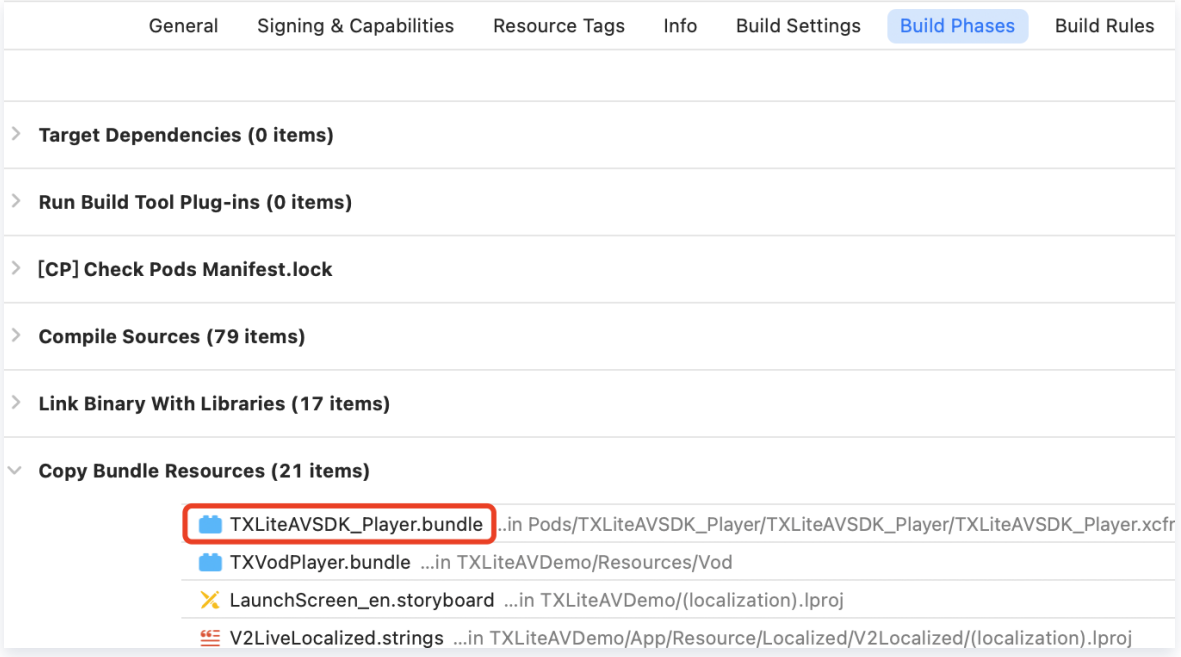
1. 下载 SDK + Demo 开发包，腾讯云视立方 iOS 播放器项目为 LiteAVSDK/Player_iOS 。
2. 导入 `TXLiteAVSDK_Player.xcframework` 到工程中，并勾选 `Do Not Embed` 。
3. 将 Demo/TXLiteAVDemo/SuperPlayerKit/SuperPlayer 拷贝到自己的工程目录下。
4. SuperPlayer 依赖第三方库包括：AFNetworking、SDWebImage、Masonry、TXLiteAVSDK_Player。
   如果是手动集成 TXLiteAVSDK_Player，需要添加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

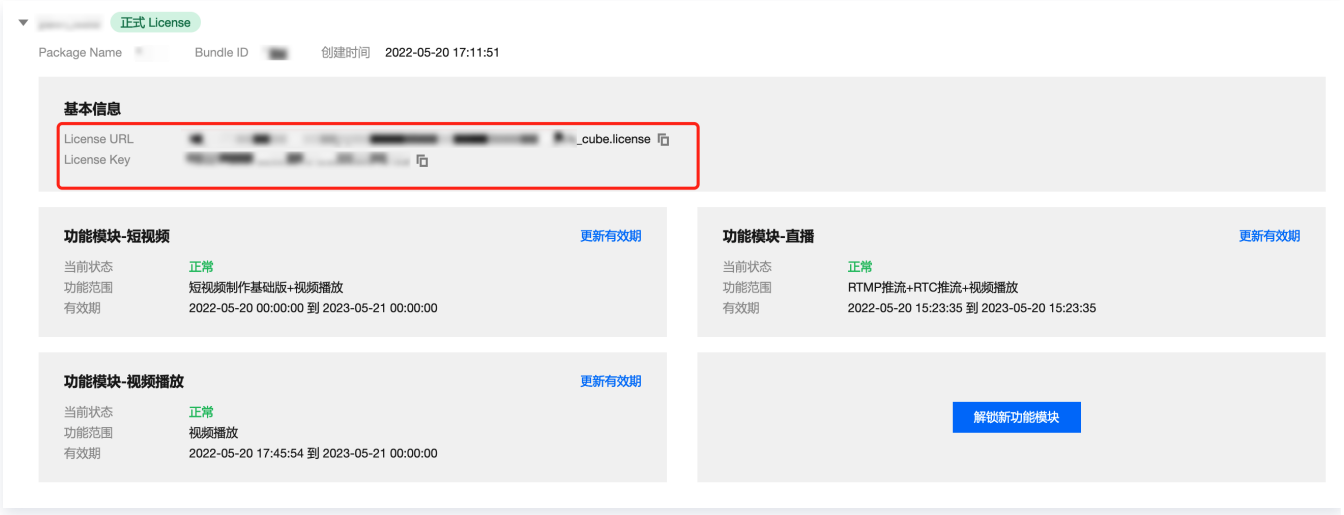
1. **创建播放器：**

   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 > 超级播放器**中体验，单击界面右下角**全屏**即可进入全屏播放界面。

在窗口播放模式下，可通过调用下述接口进入全屏播放模式：

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

全屏播放界面功能介绍

返回窗口模式

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

```
// 返回事件
- (void)superPlayerBackAction:(SuperPlayerView *)player;
```

单击左上角返回按钮触发

```
// 全屏改变通知
- (void)superPlayerFullScreenChanged:(SuperPlayerView *)player;
```

锁屏

锁屏操作可以让用户进入沉浸式播放状态。单击后由 SDK 自己处理，无回调。

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

弹幕

打开弹幕功能后屏幕上会有用户发送的文字飘过。
在这里拿到 SPDefaultControlView 对象，播放器 view 初始化的时候去给 SPDefaultControlView 的弹幕按钮设置事件，弹幕内容和弹幕 view 需要用户自己自定义，详细参见 SuperPlayerDemo 下的

CFDanmakuView、CFDanmakuInfo、CFDanmaku。

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

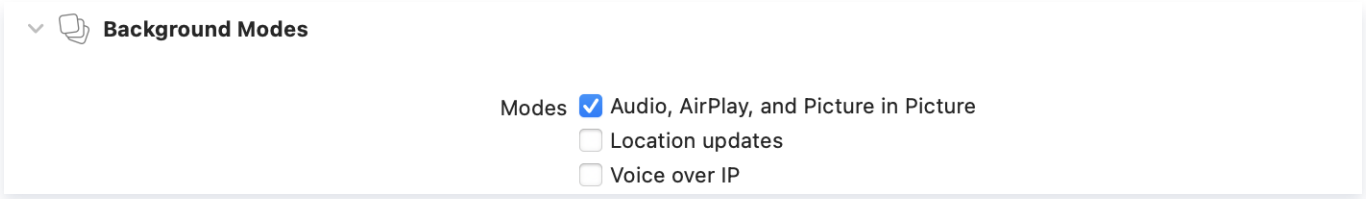
接口参数说明：

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];
```

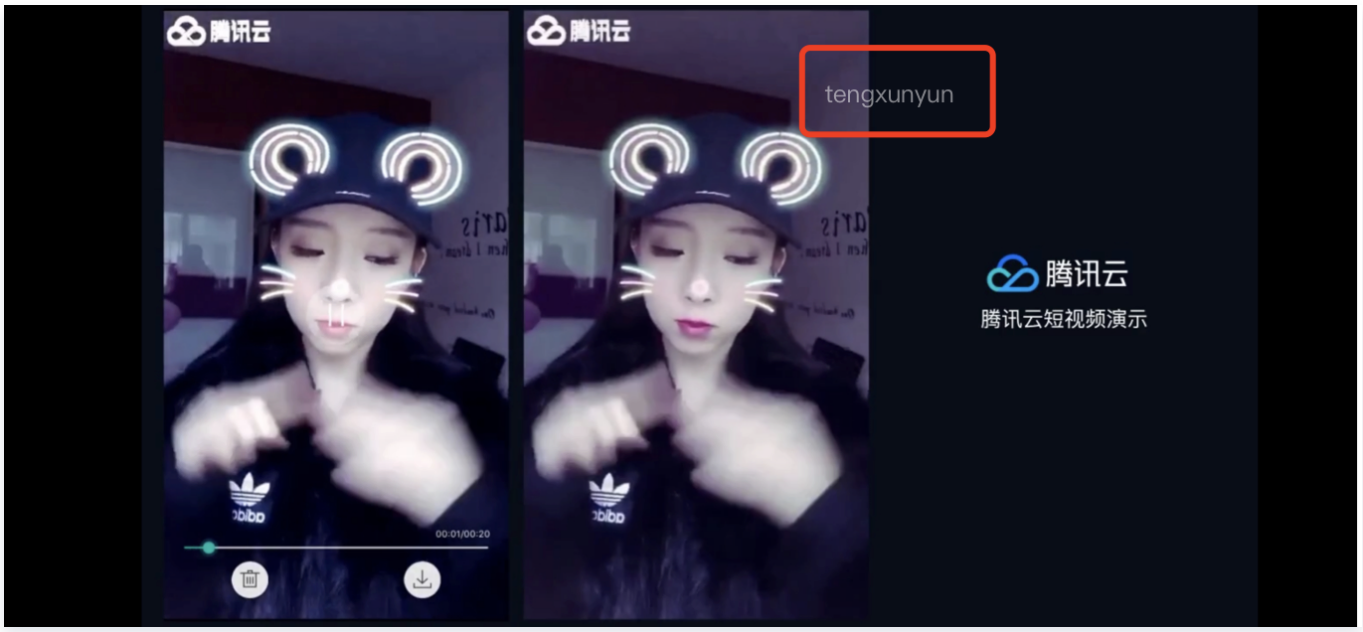
TXVipWatchModel 类参数说明：

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];
```

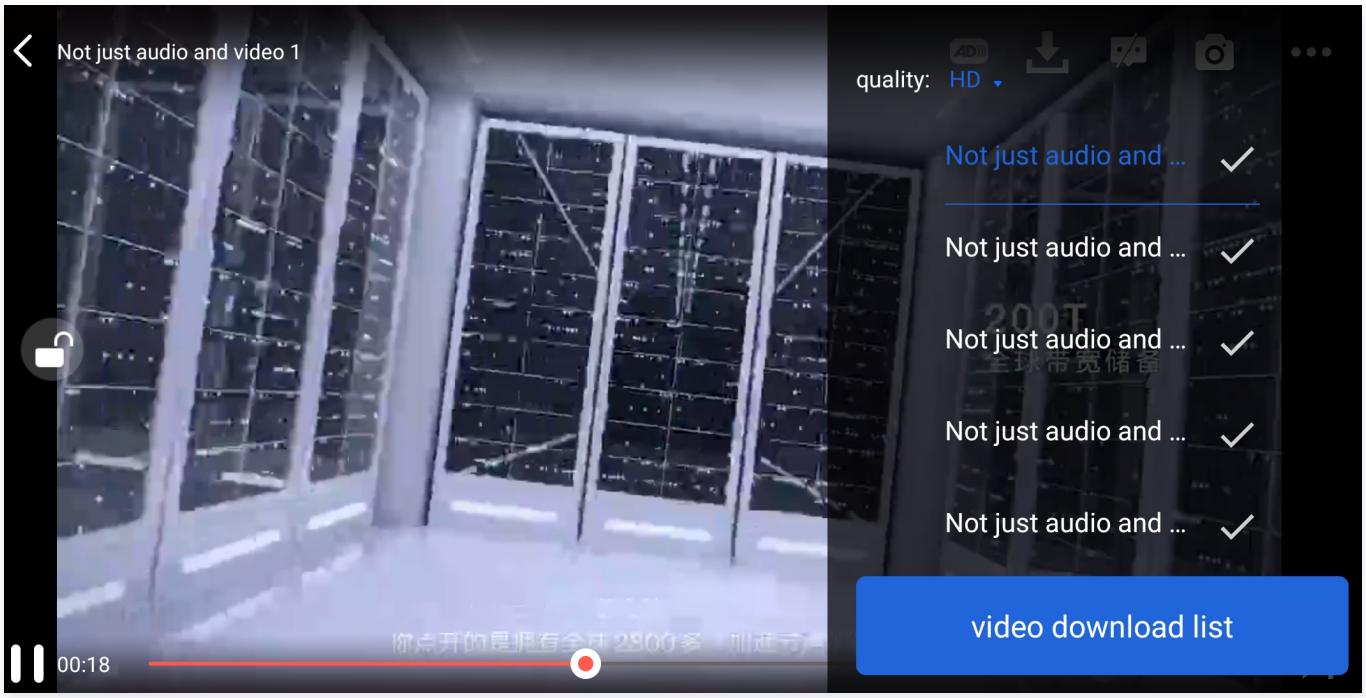
DynamicWaterModel 类参数说明：

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 = [param objectForKey:VOD_PLAY_EVENT_IMAGESPRIT_WEBVTTURL]?:@"";
NSArray<NSString *> *imageSpriteList = [param objectForKey: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。

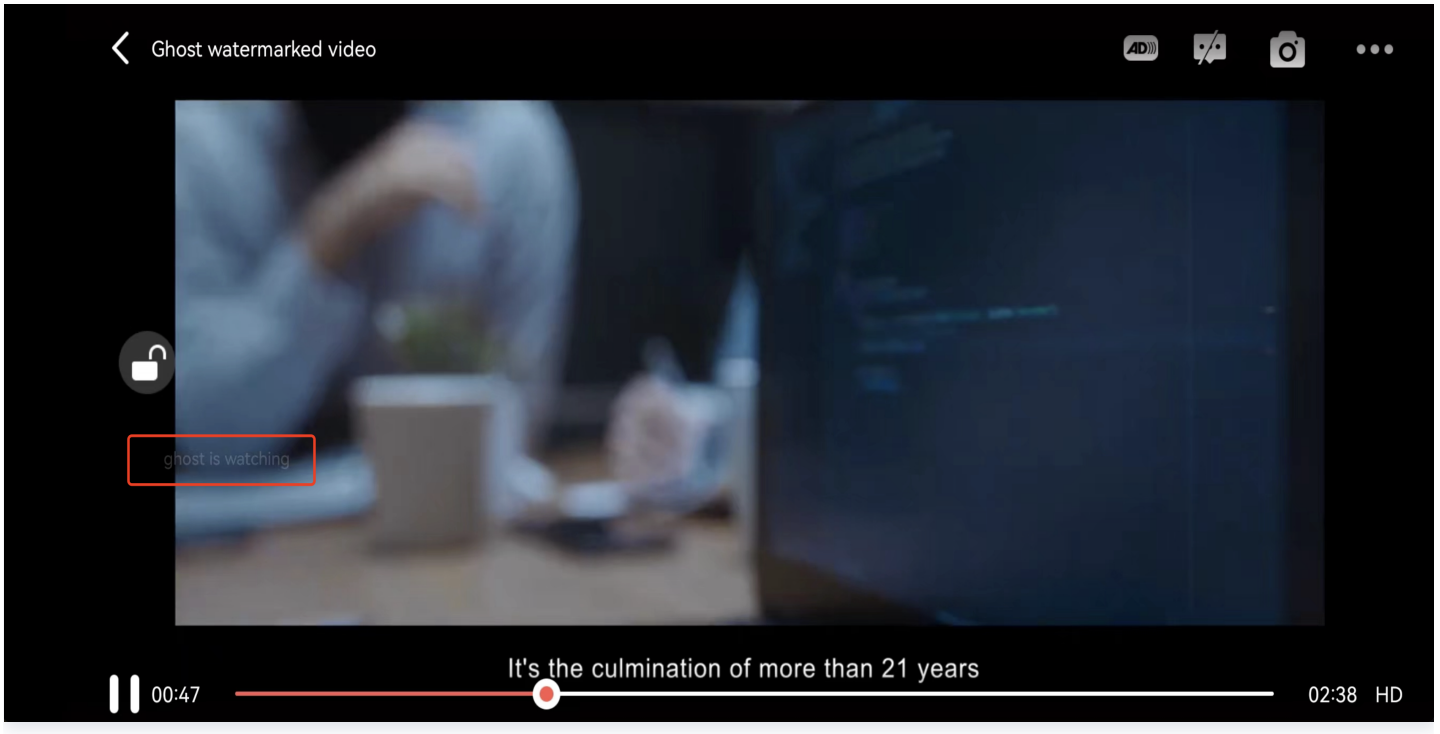
```
TXPlayerSubtitleRenderModel *model = [[TXPlayerSubtitleRenderModel alloc]
init];
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.isBondFontStyle = NO;  // 设置字幕字体是否为粗体
model.fontColor = 0xFF000000; // 设置字幕字体颜色，默认白色不透明
[_txVodPlayer setSubtitleStyle:model];
```

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.eyJhcHBJZCI6MTUwMDAwNjQzOCwiZmlsZUlkIjo
iMzg3NzAyMzA3ODQ3MTI5MTI3IiwiY29udGVudEluZm8iOnsiYXVkaW9WaWRlb1R5cGUiOiJSYXdBZ
GFwdGl2ZSIsInJhd0FkYXB0aXZlRGVmaW5pdGlvbiI6MTB9LCJjdXJyZW50VGltZVN0YW1wIjoxNjg
2ODgzMzYwLCJnaG9zdFdhdGVybWFya0luZm8iOnsidGV4dCI6Imdob3N0IGlzIHdhdGNoaW5nIn19.
0G2o4P5xVZ7zFlFUgBLntfX03iGxK9ntD_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（原超级播放器）

最近更新时间：2024-11-25 15:02:02

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 。

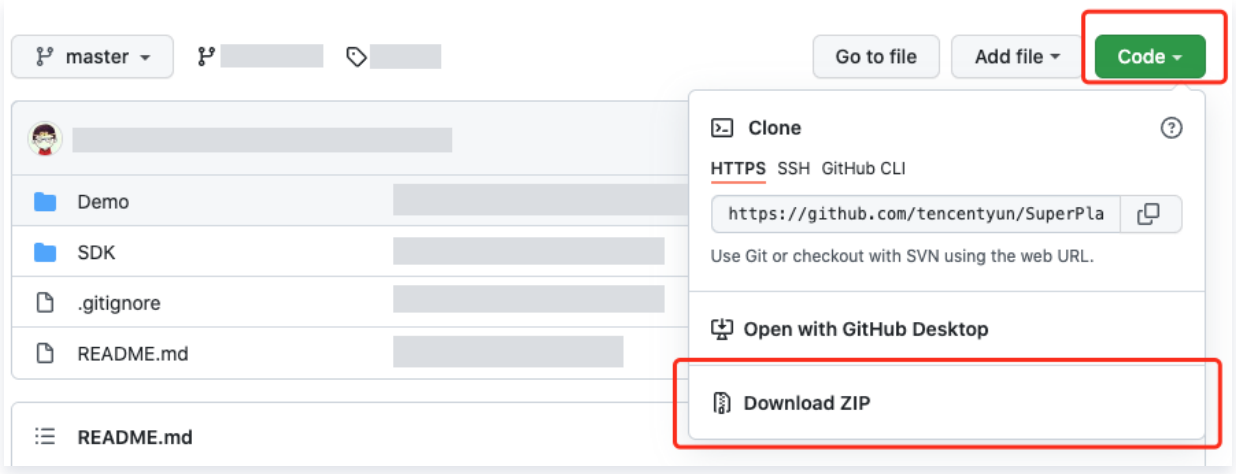
集成准备

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
superplayerde	播放器组件 Demo 代码

mo	
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

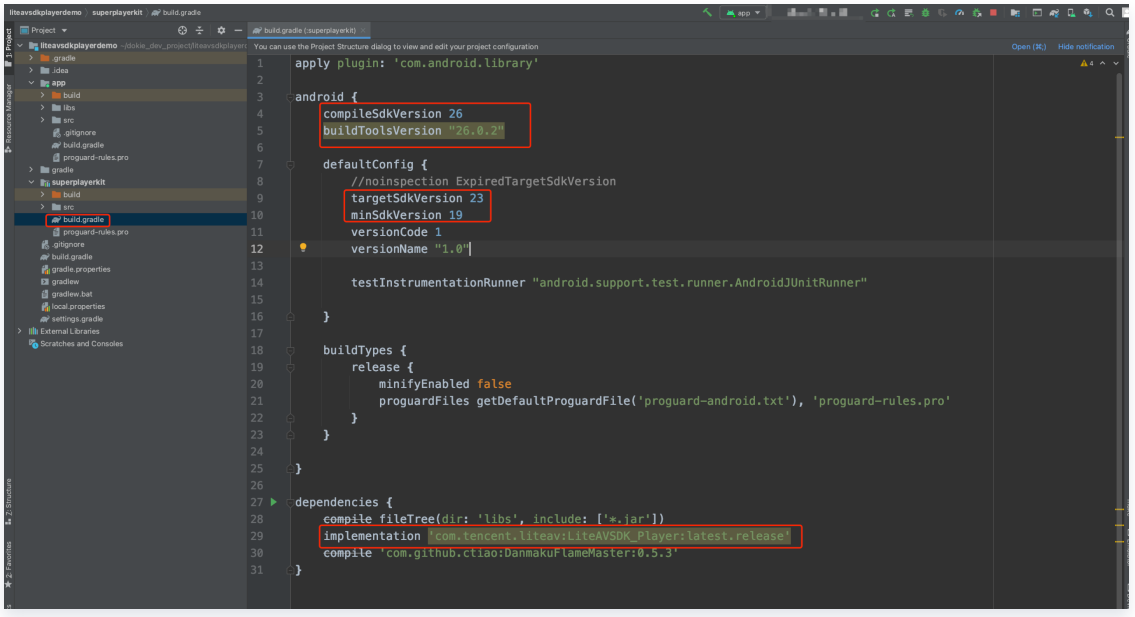
本步骤可指导您如何集成播放器，您可选择使用 Gradle 自动加载的方式，手动下载 aar 再将其导入到您当前的工程，或导入 jar 和 so 库的方式集成项目。

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'
```

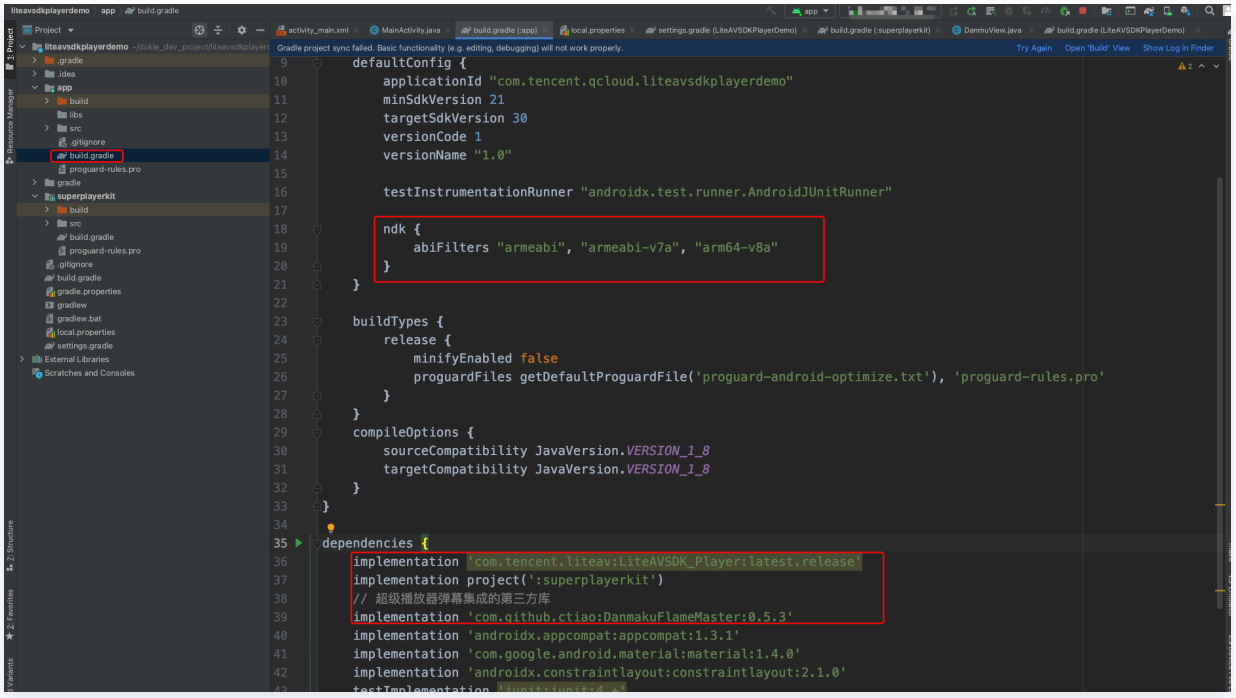
   - 打开 `superplayerkit` 工程的 build.gradle 文件修改 compileSdkVersion，buildToolsVersion，minSdkVersion，targetSdkVersion 和 rootProject.ext.liteavSdk 的常量值。



```
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：

```
 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()
}
```

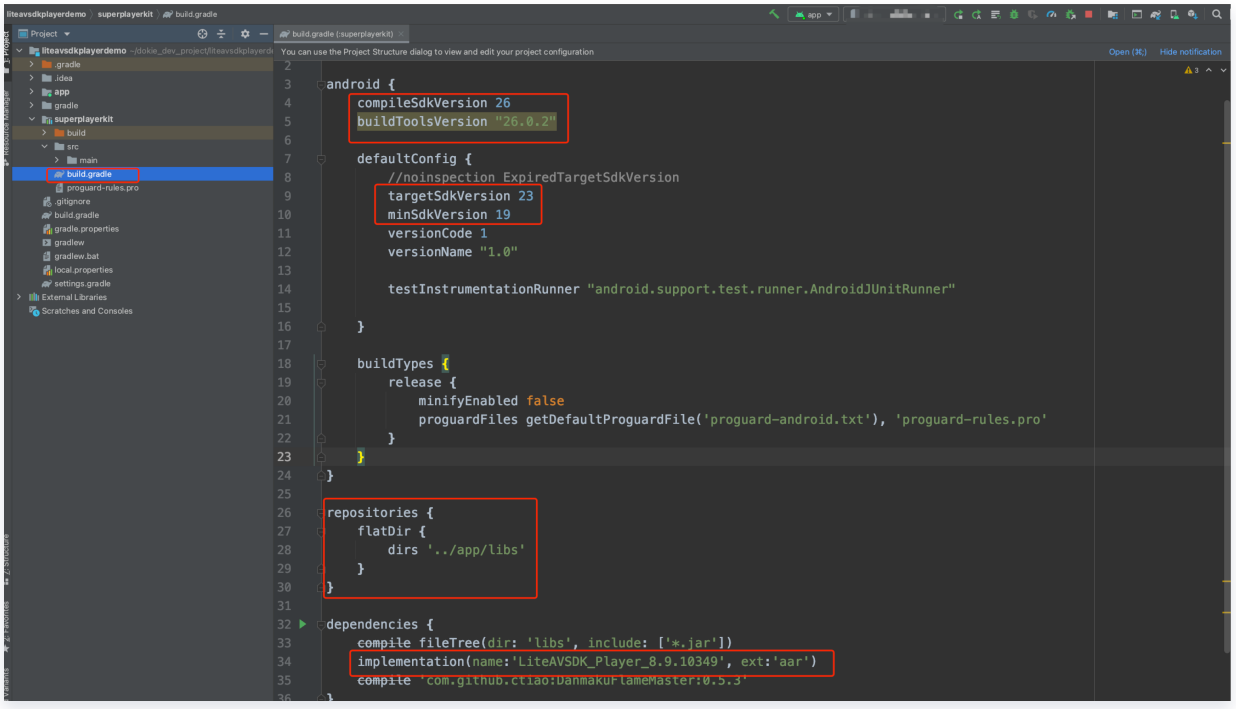
4. 单击  Sync Now 按钮同步 SDK，如果您的网络连接 mavenCentral 没有问题，很快 SDK 就会自动下载集成到工程里。

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"
```

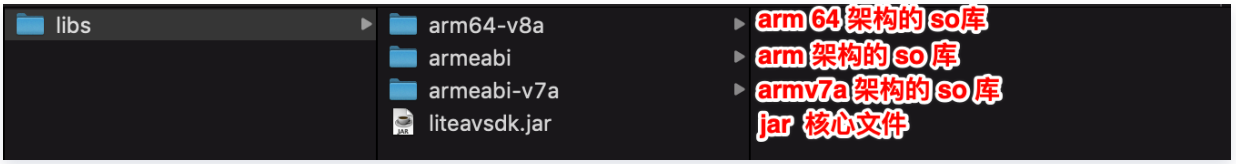
```
    exclude "lib/armeabi-v7a/libijkhlscache-master.so"
    exclude "lib/arm64-v8a/libijkhlscache-master.so"
}
```

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：

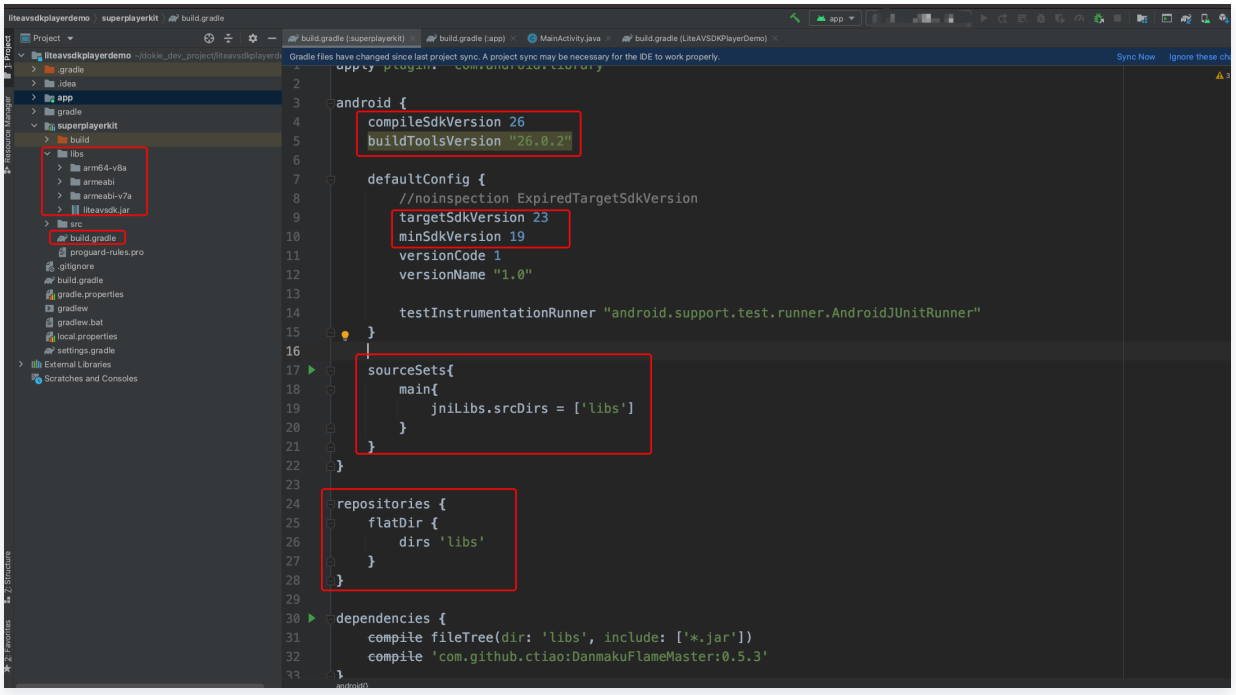
1. 下载 SDK + Demo 开发包，项目地址为 Android，下载完成后进行解压。在 SDK 目录找到 SDK/LiteAVSDK_Player_XXX.zip（其中 XXX 为版本号），解压得到 libs 目录，里面包含 jar 文件和 so 文件夹，文件清单如下：



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.eyJhcHBJZCI6MTQwMDMyOTA3MSwiZmlsZUlk
IjoiNTI4NTg5MDc5OTcxMDE3MzY1MCIsImN1cnJlbnRUaW1lU3RhbXAiOjEsImV4cGlyZVRpbW
VTdGFtcCI6MjE0NzQ4MzY0NywidXJsQWNjZXNzSW5mbyI6eyJ0IjoiN2ZmZmZmZmYifSwiZHJt
TGljZW5zZUluZm8iOnsiZXhwaXJlVGltZVN0YW1wIjoyMTQ3NDgzNjQ3fX0.yJxpnQ2Evp5KZQ
FfuBBK05BoPpQAzYAWo6liXws-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
  }
});
```

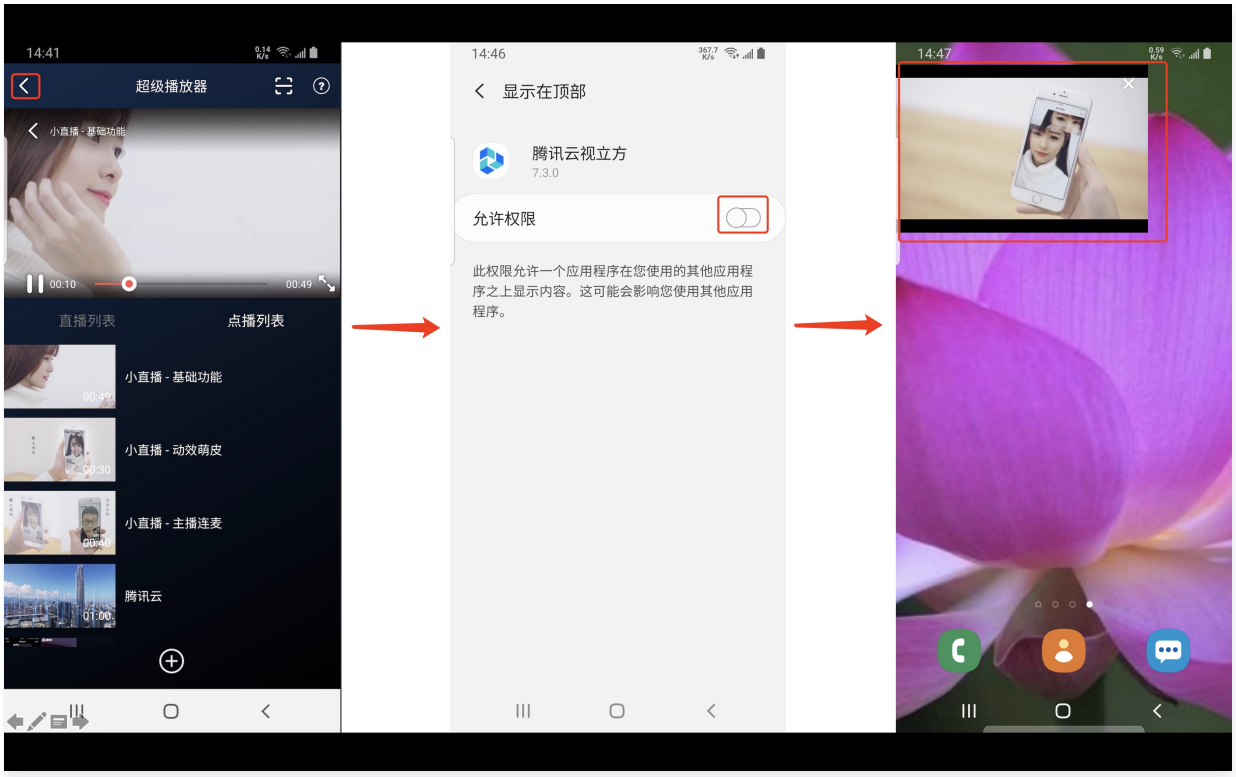
清晰度切换

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|smallestScreenSiz
e|screenLayout">

</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(),mTXCloudVid
eoView);
```

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);
```

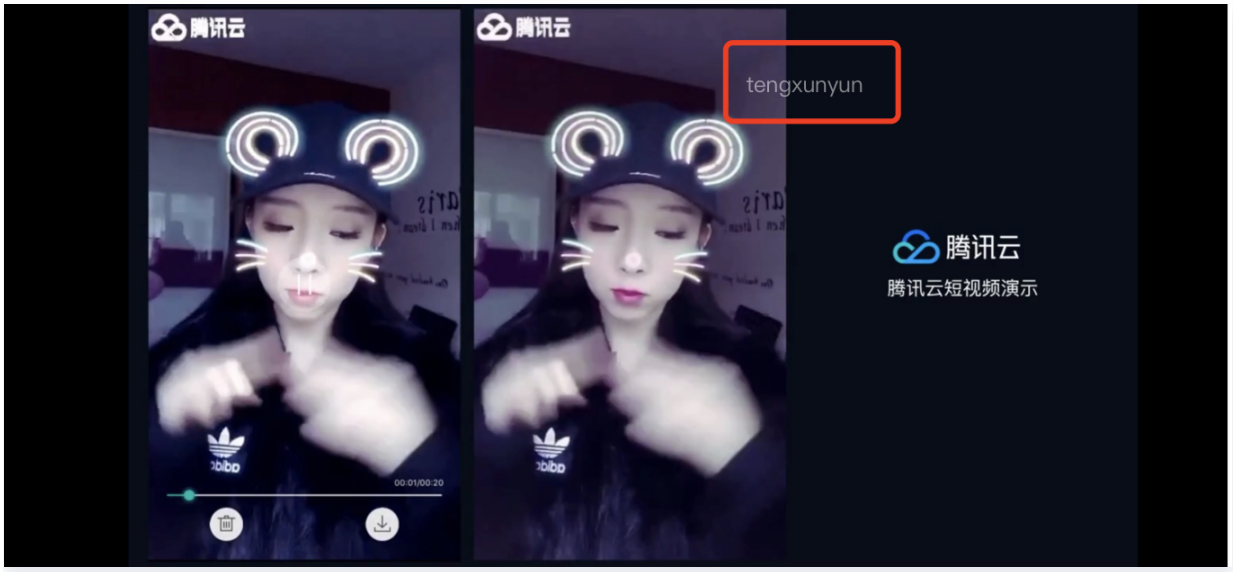
```
public VipWatchModel(String tipStr, long canWatchTime)
```

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)
```

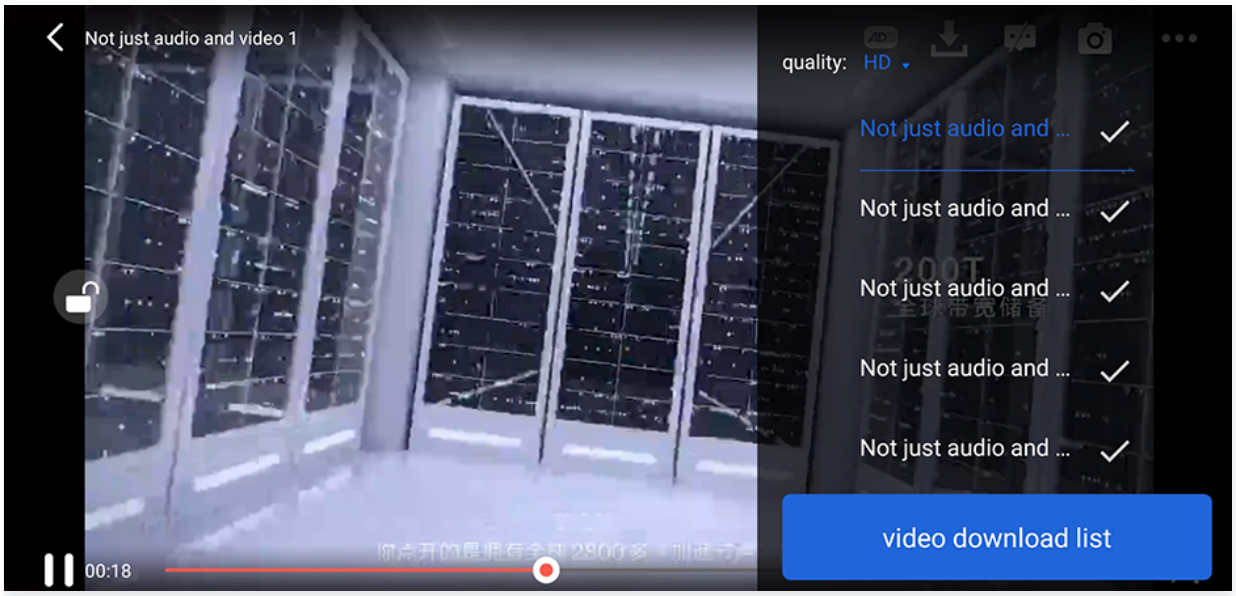
接口参数说明

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，再点击要下载的视频选项，出现对勾后，代表开始了下载。点击下方的 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);
     }
```

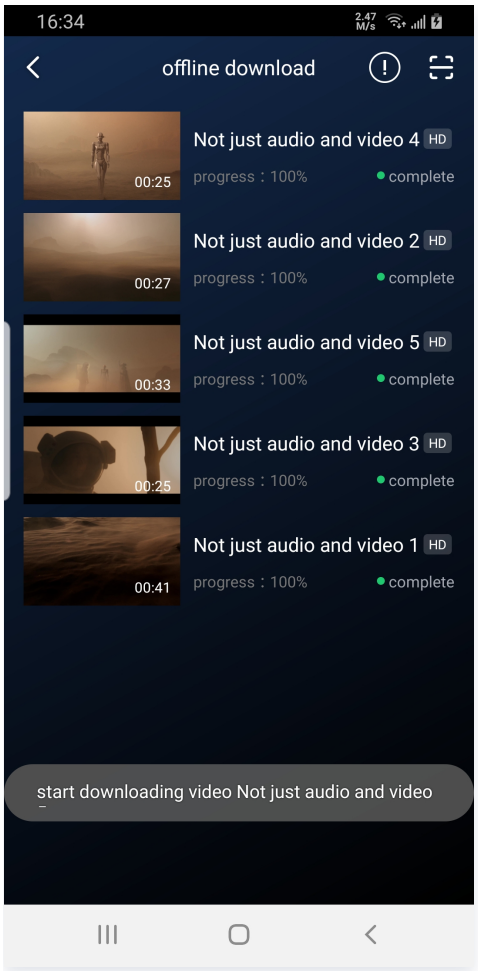
```
});

// 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。

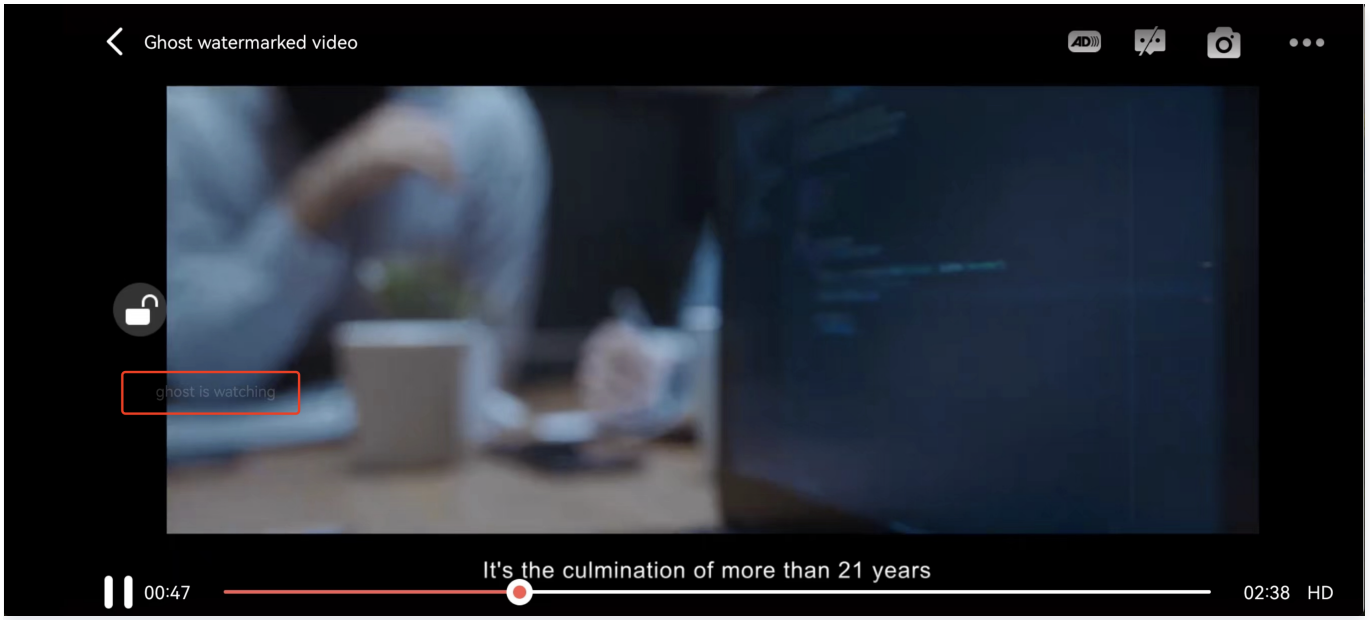
```
TXSubtitleRenderModel model = new TXSubtitleRenderModel();
model.canvasWidth = 1920;  // 字幕渲染画布的宽
model.canvasHeight = 1080;  // 字幕渲染画布的高
model.fontColor = 0xFFFFFFFF; // 设置字幕字体颜色，默认白色不透明
model.isBondFontStyle = false;  // 设置字幕字体是否为粗体
mVodPlayer.setSubtitleStyle(model);
```

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（播放器组件）

最近更新时间：2025-04-01 14:22:32

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/d62d88a7vodtranscq1400329073/59c68fe7
5285890800381567412/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 `Landscaoe 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` 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。