

播放器 SDK 旧版文档



腾讯云

【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【 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

旧版文档

TCPlayerLite (旧)

TCPlayerLite 升级指引

Adapter 插件

Web

iOS

Android

旧版文档

TCPlayerLite (旧)

最近更新时间：2024-09-12 15:58:52

ⓘ 说明：

- TCPlayerLite 为旧版播放器，后续将持续维护，但不再主动做功能迭代，后续 Web 端播放器功能迭代将在 [超级播放器 TCPlayer](#) 内进行。
- 当前 [超级播放器 TCPlayer](#) 已包含旧版播放器 TCPlayerLite 的全部能力，同时具备更多丰富功能。若您首次使用腾讯云 Web 播放器，建议您直接使用超级播放器 TCPlayer。若您当前仍在使用 TCPlayerLite，建议您尽早升级为超级播放器 TCPlayer，以享受更多更全面的功能及服务。TCPlayerLite 升级为 TCPlayer 方式可参见 [TCPlayerLite 升级指引](#)。

功能介绍

腾讯云 Web 超级播放器 TCPlayerLite 是为了解决在手机浏览器和 PC 浏览器上播放音视频流的问题，它使您的视频内容可以不依赖用户安装 App，就能在朋友圈和微博等社交平台进行传播。本文档适合有一定 Javascript 语言基础的开发人员阅读。

以下视频将为您讲解腾讯云播放器 SDK 的 Web 播放器的功能特性以及对接攻略：

[观看视频](#)

基础知识

对接前需要了解如下基础知识：

• 直播和点播

直播视频源是实时的，一旦主播停播，直播地址就失去意义，而且由于是实时直播，所以播放器在播直播视频的时候是没有进度条的。

点播视频源是某个服务器上的文件，只要文件没有被提供方删除，就可以随时播放，而且由于整个视频都在服务器上，所以播放器在播点播视频的时候是有进度条的。

• 协议支持

TCPlayerLite 的视频播放能力本身不是网页代码实现的，而是靠浏览器支持，所以其兼容性不像我们想象的那么好，因此，**不是所有的手机浏览器都能有符合预期的表现**。一般用于网页直播的视频源地址是以 M3U8 结尾的地址，我们称其为 HLS (HTTP Live Streaming)，这是苹果推出的标准，目前各种手机浏览器产品对这种格式的兼容性也最好，但它有个问题：延迟比较大，一般是20s - 30s左右的延迟。

视频协议	用途	URL 地址格式	PC 浏览器	移动浏览器
------	----	----------	--------	-------

WebRTC	只适用直播	<code>webrtc://xxx.liveplay.myqcloud.com/live/xxx</code>	支持	支持
HLS (M3U8)	可用于直播	<code>http://xxx.liveplay.myqcloud.com/xxx.m3u8</code>	支持	支持
HLS (M3U8)	可用于点播	<code>http://xxx.vod.myqcloud.com/xxx.m3u8</code>	支持	支持
FLV	可用于直播	<code>http://xxx.liveplay.myqcloud.com/xxx.flv</code>	支持	不支持
FLV	可用于点播	<code>http://xxx.vod.myqcloud.com/xxx.flv</code>	支持	不支持
MP4	只适用点播	<code>http://xxx.vod.myqcloud.com/xxx.mp4</code>	支持	支持

注意:

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，默认在移动端转换为 HLS，PC 端转换为 FLV。

功能支持

功能 \ 浏览器	Chrome	Firefox	Edge	QQ 浏览器	Mac Safari	iOS Safari	iOS 微信、QQ	Android Chrome	Android 微信、QQ	手机 QQ 浏览器	IE 8,9,10,11
设置封面	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
多清晰度支持	✓	✓	✓	✓	×	×	×	×	×	×	✓
定制错误提示语	✓	✓	✓	✓	✓	×	×	×	×	×	✓

快直播	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
-----	---	---	---	---	---	---	---	---	---	---	---

对接攻略

Step1. 页面准备工作

在需要播放视频的页面（PC 或 H5）中引入初始化脚本。

```
<script
src="https://web.sdk.qqcloud.com/player/tcplayerlite/release/v2.4.5/TcPlayer-2.4.5.js" charset="utf-8"></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](#)。

如果您部署的地址为 `aaa.xxx.ccc`，在合适的地方引入播放器脚本文件：

```
<script src="aaa.xxx.ccc/TcPlayer-x.x.x.js"></script>
```

⚠ 注意：

直接用本地网页无法调试，Web 播放器无法处理该情况下的跨域问题。

Step2. 在 HTML 中放置容器

在需要展示播放器的页面位置加入播放器容器，即放一个 div 并命名，例如 `id_test_video`，视频画面都会在容器里渲染。对于容器的大小控制，您可以使用 div 的属性进行控制，示例代码如下：

```
<div id="id_test_video" style="width:100%; height:auto;"></div>
```

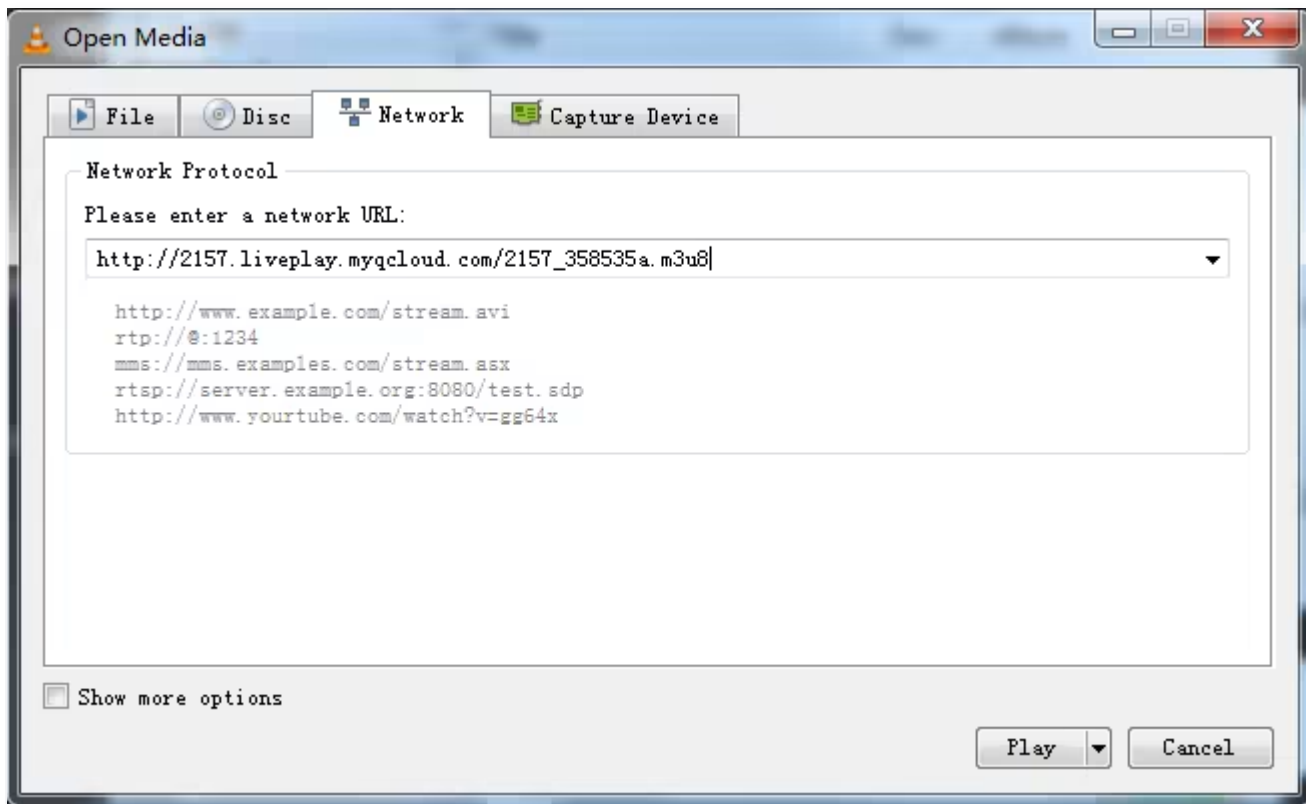
Step3. 对接视频播放

编写 Javascript 代码，作用是去指定的 URL 地址拉取音视频流，并将视频画面呈现到添加的容器内。

3.1 简单播放

如下是一个 [直播格式的 URL 地址](#)，使用 HLS（M3U8）协议，如果主播在直播中，则用 VLC 等播放器是可以直接打开该 URL 进行观看的：

```
http://2157.liveplay.myqcloud.com/2157_358535a.m3u8 // m3u8 播放地址
```



如果要在手机浏览器上播放该 URL 的视频，则 Javascript 代码如下：

```
var player = new TcPlayer('id_test_video', {
    "m3u8": "http://2157.liveplay.myqcloud.com/2157_358535a.m3u8", //请
    替换成实际可用的播放地址
    "autoplay" : true, //iOS 下 safari 浏览器，以及大部分移动端浏览器是
    不开放视频自动播放这个能力的
    "poster" : "http://www.test.com/myimage.jpg",
    "width" : '480', //视频的显示宽度，请尽量使用视频分辨率宽度
    "height" : '320' //视频的显示高度，请尽量使用视频分辨率高度
});
```

这段代码可以支持在 PC 及手机浏览器上播放 HLS (M3U8) 协议的直播视频，虽然 HLS (M3U8) 协议的视频兼容性不错，但部分 Android 手机依然不支持，其延迟较高，大约20秒以上的延迟。

3.2 实现更低延迟

```
var player = new TcPlayer('id_test_video', {
    "m3u8": "http://2157.liveplay.myqcloud.com/2157_358535a.m3u8",
    "flv": "http://2157.liveplay.myqcloud.com/live/2157_358535a.flv",
    //增加了一个 flv 的播放地址，用于PC平台的播放 请替换成实际可用的播放地址
```

```
"autoplay" : true, //iOS 下 safari 浏览器，以及大部分移动端浏览器是不开放视频自动播放这个能力的
"poster" : "http://www.test.com/myimage.jpg",
"width" : '480', //视频的显示宽度，请尽量使用视频分辨率宽度
"height" : '320' //视频的显示高度，请尽量使用视频分辨率高度
});
```

这段代码中增加了 FLV 的播放地址，Web 播放器如果发现当前的浏览器是 PC 浏览器，会主动选择 FLV 链路，从而实现更低的延迟。如果对延迟有更高的要求，可以使用 WebRTC 拉流地址，基于 WebRTC 的播放系统可以实现超低延迟（500ms），前提条件是拉流地址都是可以出流的，如果您使用腾讯云的直播服务，则无需考虑，因为腾讯云的直播频道默认支持 WebRTC、FLV 和 HLS（M3U8）播放协议。

无法播放怎么办？

如果您发现视频无法播放，可能存在如下原因：

● 原因一：视频源有问题

如果是直播 URL，则需要检查主播是否已经停止推流，可以用浮窗提示观众：“主播已经离开”。请参见 [直播推流](#)。

如果是点播 URL，则需要检查要播放的文件是否还存在于服务器上（如播放地址是否已经从点播系统移除）。

● 原因二：本地网页调试

目前 TCPlayerLite 不支持本地网页调试（即通过 `file://` 协议打开视频播放的网页），因为浏览器有跨域安全限制，所以在 Windows 系统上放置一个 `test.html` 文件来进行测试是无法播放的，需要将其上传到服务器上进行测试。而前端工程师可以通过反向代理的方式，对线上页面进行本地代理以实现本地调试，这是主流的本地调试方法。

● 原因三：手机兼容问题

普通的手机浏览器只支持 HLS（M3U8）协议的播放，不支持 FLV，最新版本的 QQ 浏览器支持 FLV 协议的播放。

Step4. 给播放器设置封面

设置封面涉及到 `poster` 属性，下面将详细介绍 `poster` 属性的使用方法。

⚠ 注意：

封面功能在部分移动端播放环境下可能失效，通常是由于移动端 `webview` 劫持视频播放造成的，需要 `webview` 支持 `video` 叠加元素或者放开劫持视频播放。相关详细说明请参见 [常见问题](#)。

4.1 简单设置封面

`poster` 支持传入图片地址作为播放器的封面，在播放器区域内居中，并且以图片的实际分辨率进行显示。

```
"poster" : "http://www.test.com/myimage.jpg"
```

4.2 设置封面样式

poster 支持传入一个对象，在对象中可以对封面的展现样式（style）和图片地址（src）进行设置。

style 支持的样式如下：

- default：居中并且以图片的实际分辨率进行显示。
- stretch：拉伸铺满播放器区域，图片可能会变形。
- cover：优先横向等比拉伸铺满播放器区域，图片某些部分可能无法显示在区域内。

```
"poster" : {"style":"stretch", "src":"http://www.test.com/myimage.jpg"}
```

4.3 实现用例

使用 cover 方式显示封面。线上示例如下，在 PC 浏览器中右键单击[查看页面源码](#)即可查看页面的代码实现：[视频封面](#)

⚠ 注意：

- 在某些移动端设置封面会无效，具体说明请参见 [常见问题](#)。
- 以上示例链接仅用于文档演示，请勿用于生产环境。

Step5. 多清晰度支持

5.1 原理介绍

同腾讯视频，Web 播放器支持多清晰度，如下图所示：



播放器本身是没有能力去改变视频清晰度的，视频源只有一种清晰度，称之为原画，而原画视频的编码格式和封装格式多种，Web 端无法支持播放所有的视频格式，如点播支持以 H.264 为视频编码，MP4 和 FLV 为封装格式的视

频。

多清晰度的实现依赖于视频云：

- 对于直播，来自主播端的原始视频会在腾讯云进行实时转码，分出多路转码后的视频，每一路视频都有其对应的地址，例如“高清-HD”和“标清-SD”，地址格式如下：

```
http://2157.liveplay.myqcloud.com/2157_358535a.m3u8           // 原画
http://2157.liveplay.myqcloud.com/2157_358535a_900.m3u8      // 高清
http://2157.liveplay.myqcloud.com/2157_358535a_550.m3u8      // 标清
```

- 对于点播，一个视频文件上传到腾讯云后，您可以对该视频文件进行转码，产生其它几种清晰度的视频，例如“高清-HD”和“标清-SD”，地址格式如下：

```
http://200002949.vod.myqcloud.com/200002949_b6ffc.f240.m3u8 //
原画，用转码后的超清替换
http://200002949.vod.myqcloud.com/200002949_b6ffc.f230.av.m3u8 //
高清
http://200002949.vod.myqcloud.com/200002949_b6ffc.f220.av.m3u8 //
标清
```

⚠ 注意：

上传后的原始视频是未经过腾讯云转码的，不能直接用于播放。

5.2 代码实现

多清晰度支持的代码实现如下所示：

```
var player = new TcPlayer('id_test_video', {
    "m3u8" :
    "http://200002949.vod.myqcloud.com/200002949_b6ffc.f240.m3u8", //请替换成实际可用的播放地址
    "m3u8_hd" :
    "http://200002949.vod.myqcloud.com/200002949_b6ffc.f230.av.m3u8",
    "m3u8_sd" :
    "http://200002949.vod.myqcloud.com/200002949_b6ffc.f220.av.m3u8",
    "autoplay" : true, //iOS 下 safari 浏览器，以及大部分移动端浏览器是不开放视频自动播放这个能力的
    "poster" : "http://www.test.com/myimage.jpg",
});
```

5.3 实现用例

使用多种分辨率的设置及切换功能。线上示例如下，在 PC 浏览器中右键单击[查看页面源码](#)即可查看页面的代码实现：[分辨率切换](#)

正常情况将看到如下效果：



⚠ 注意：

- PC 端现已支持多种清晰度播放及切换的功能，移动端尚未支持。
- 以上示例链接仅用于文档演示，请勿用于生产环境。

Step6. 定制错误提示语

Web 播放器支持提示语定制。

6.1 代码实现

如下是让播放器支持自定义提示语的核心代码，主要在 wording 属性上设置提示语。

```
var player = new TcPlayer('id_test_video', {
  "m3u8" :
  "http://200002949.vod.myqcloud.com/200002949_b6ffc.f0.m3u8", //请替换成实际
  可用的播放地址
  "autoplay" : true, //iOS 下 safari 浏览器是不开放这个能力的
  "poster" : "http://www.test.com/myimage.jpg",
```

```

"wording": {
    2032: "请求视频失败, 请检查网络",
    2048: "请求m3u8文件失败, 可能是网络错误或者跨域问题"
}
});

```

6.2 实现用例

视频播放失败，同时使用自定义提示文案的功能。线上示例如下，在 PC 浏览器中右键单击[查看页面源码](https://web.sdk.qcloud.com/player/tcplayerlite/tcplayer-error.html)即可查看页面的代码实现：

```
https://web.sdk.qcloud.com/player/tcplayerlite/tcplayer-error.html
```

⚠ 注意：

以上示例链接仅用于文档演示，请勿用于生产环境。

6.3 错误码表

Code	提示语	说明
1	网络错误，请检查网络配置或者播放链接是否正确。	H5 提示的错误。
2	网络错误，请检查网络配置或者播放链接是否正确。	视频格式 Web 播放器无法解码。 H5 提示的错误。
3	视频解码错误。	H5 提示的错误。
4	当前系统环境不支持播放该视频格式。	H5 提示的错误。
5	当前系统环境不支持播放该视频格式。	播放器判断当前浏览器环境不支持播放传入的视频，可能是当前浏览器不支持 MSE 。
10	请勿在 file 协议下使用播放器，可能会导致视频无法播放。	-
11	使用参数有误，请检查播放器调用代码。	-
12	请填写视频播放地址。	-
2001	调用 WebRTC 接口失败	播放 WebRTC 时设置 sdp 失败提示的错误。

2002	调用拉流接口失败	播放 WebRTC 时调用拉流接口失败提示的错误。
2003	连接服务器失败，并且连接重试次数已超过设定值	播放 WebRTC 时提示的错误，可用于确定是否为停止推流状态。

说明：

- Code 1 - 4 对应的是 H5 原生事件。
- 由于 H5 视频播放标准的不确定性，错误提示语会不定期更新。

源码参考

如下是一个线上示例代码，在 PC 浏览器中右键单击[查看页面源码](#)即可查看页面的代码实现：[播放示例](#)。

注意：

以上示例链接仅用于文档演示，请勿用于生产环境。

参数列表

播放器支持的所有参数，如下所示：

参数	类型	默认值	参数说明
webrtc	String	无	原画 WebRTC 播放 URL。 示例： webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1
webrtc_hd	String	无	高清 WebRTC 播放 URL。 示例： webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1_hd
webrtc_sd	String	无	标清 WebRTC 播放 URL。 示例： webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1_sd
m3u8	String	无	原画 M3U8 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535a.m3u8

m3u8_hd	String	无	高清 M3U8 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535ahd.m3u8
m3u8_sd	String	无	标清 M3U8 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535asd.m3u8
flv	String	无	原画 FLV 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535a.flv
flv_hd	String	无	高清 FLV 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535ahd.flv
flv_sd	String	无	标清 FLV 播放 URL。 示例： http://2157.liveplay.myqcloud.com/2157_358535asd.flv
mp4	String	无	原画 MP4 播放 URL。 示例： http://200002949.vod.myqcloud.com/200002949_b6ffc.f0.mp4
mp4_hd	String	无	高清 MP4 播放 URL。 示例： http://200002949.vod.myqcloud.com/200002949_b6ffc.f40.mp4
mp4_sd	String	无	标清 MP4 播放 URL。 示例： http://200002949.vod.myqcloud.com/200002949_b6ffc.f20.mp4
width	Number	无	必选，设置播放器宽度，单位为像素。 示例：640
height	Number	无	必选，设置播放器高度，单位为像素。 示例：480

volume	Number	0.5	设置初始音量，范围：0到1 [v2.2.0+]。 示例：0.6
live	Boolean	false	必选，设置视频是否为直播类型，将决定是否渲染时间轴等控件，以及区分点直播的处理逻辑。 示例：true
autoplay	Boolean	false	是否自动播放。（备注：该选项只对大部分 PC 平台生效） 示例：true
poster	String/ Object	无	预览封面，可以传入一个图片地址或者一个包含图片地址 src 和显示样式 style 的对象。style 可选属性： default 居中1: 1显示。 stretch 拉伸铺满播放器区域，图片可能会变形。 cover 优先横向等比拉伸铺满播放器区域，图片某些部分可能无法显示在区域内。 示例： <code>"http://www.test.com/myimage.jpg"</code> 或者 <pre>{ "style": "cover", "src": } [v2.3.0+]</pre>
controls	String	"default"	default 显示默认控件，none 不显示控件，system 移动端显示系统控件。 注意：如果需要在移动端使用系统全屏，就需要设置为 system。默认全屏方案是使用 Fullscreen API + 伪全屏的方式， 在线示例 示例："system"
systemFullscreen	Boolean	false	开启后，在不支持 Fullscreen API 的浏览器环境下，尝试使用浏览器提供的 webkitEnterFullScreen 方法进行全屏，如果支持，将进入系统全屏，控件为系统控件。 示例：true
h5_flv	Boolean	false	是否启用 flv.js 的播放 flv。启用时播放器将在支持 MSE 的浏览器下，采用 flv.js 播放 flv，然而并不是所有支持 MSE 的浏览器都可以使用 flv.js，所以播放器不会默认开启这个属性， [v2.2.0+]。 示例：true
x5_player	Boolean	false	是否启用 TBS 的播放 flv 或 hls 。启用时播放器将在 TBS 模式下（例如 Android 的微信、QQ 浏览器），将 flv 或 hls 播放地址直接赋给 <video> 播放。 TBS 视频能力 [v2.2.0+]。

			示例: true
x5_type	String	无	通过 video 属性 “x5-video-player-type” 声明启用同层 H5 播放器, 支持的值: h5-page (该属性为 TBS 内核实验性属性, 非 TBS 内核不支持), TBS H5 同层播放器接入规范 。 示例: "h5-page"
x5_fullscreen	String	无	通过 video 属性 “x5-video-player-fullscreen” 声明视频播放时是否进入到 TBS 的全屏模式, 支持的值: true (该属性为 TBS 内核实验性属性, 非 TBS 内核不支持)。 示例: "true"
x5_orientation	Number	无	通过 video 属性 “x5-video-orientation” 声明 TBS 播放器支持的方向, 可选值: <ul style="list-style-type: none"> • 0: landscape 横屏 • 1: portrait 竖屏 • 2: landscape & verbar; portrait 跟随手机自动旋转。 注意: 该属性为 TBS 内核实验性属性, 非 TBS 内核不支持, [v2.2.0+]。 示例: 0
wording	Object	无	自定义文案。 示例: { 2032: '请求视频失败, 请检查网络'}
clarity	String	'od'	默认播放清晰度 [v2.2.1+]。 示例: clarity: 'od'
clarityLabel	Object	{od: '超清', hd: '高清', sd: '标清'}	自定义清晰度文案 [v2.2.1+]。 示例: clarityLabel: {od: '蓝光', hd: '高清', sd: '标清'}。
listener	Function	无	事件监听回调函数, 回调函数将传入一个 JSON 格式的对象。 示例: function(msg){//进行事件处理 }
pausePosterEnabled	Boolean	true	暂停时显示封面 [v2.3.0+]。
preload	String	'auto'	配置 video 标签的 preload 属性, 只有部分浏览器生效 [v2.3.0+]。
hlsConfig	Object	无	hls.js 初始化配置项 [v2.3.0+]。

flvConfig	Object	无	flv.js 初始化配置项 [v2.3.1+]。
webrtcConfig	Object	无	webrtc 初始化配置项 [v2.4.1+]。 支持通过 streamType 指定拉流类型，默认拉取音视频，可选单独拉取视频或单独拉取音频，streamType 可选属性： <ul style="list-style-type: none"> • auto：拉取视频流和音频流 • video：仅拉取视频流 • audio：仅拉取音频流 示例：webrtcConfig: { streamType: 'video' }

⚠ 注意：

- WebRTC 快直播播放地址支持两种格式，除

`webrtc://domain/AppName/StreamName?txSecret=XXX&txTime=XXX` 以外，还支持 `http://domain/AppName/StreamName.sdp?txSecret=XXX&txTime=XXX` 格式的播放地址，但是需要配置播放域名 CNAME 到 `overseas-webrtc.liveplay.myqcloud.com`。

- 由于 Web 浏览器目前不支持标准 WebRTC 协议携带 B 帧播放，如果原始流存在 B 帧，则后台会自动进行转码去掉 B 帧，这样会引入额外的转码延迟，并产生转码费用。建议尽量不推包含 B 帧的流，直播 SDK，IOS 不支持引入 B 帧，安卓如果不开启 B 帧设置，默认是没有带 B 帧。如果使用 OBS 推流，可以通过设置，关闭 B 帧。

实例方法列表

播放器实例支持的方法，如下所示：

方法	参数	返回值	说明	示例
play()	无	无	开始播放视频。	player.play()
pause()	无	无	暂停播放视频。	player.pause()
togglePlay()	无	无	切换视频播放状态。	player.togglePlay()
mute(muted)	{Boolean} [可选]	true, false {Boolean}	切换静音状态，不传参则返回当前是否静音。	player.mute(true)
volume(volume)	{int} 范围：0到1 [可选]	范围：0到1	设置音量，不传参则返回当前音量。	player.volume(0.3)

playing()	无	true, false {Boolean}	返回是否在播放中。	player.playing()
duration()	无	{int}	获取视频时长。 备注：只适用于点播，需要在触发 loadedmetadata 事件后才可获取视频时长	player.duration()
currentTime(time)	{int} [可选]	{int}	设置视频播放时间点，不传参则返回当前播放时间点。 备注：只适用于点播	player.currentTime()
fullscreen(enter)	{Boolean} [可选]	true, false {Boolean}	调用全屏接口(Fullscreen API)，不支持全屏接口时使用伪全屏模式，不传参则返回值当前是否是全屏。 备注：移动端系统全屏没有提供 API，也无法获取系统全屏状态	player.fullscreen(true)
buffered()	无	0到1	获取视频缓冲数据百分比。 备注：只适用于点播	player.buffered()
destroy()	无	无	销毁播放器实例[v2.2.1+]。	player.destroy()
switchClarity()	{String} [必选]	无	切换清晰度，传值 "od"、"hd"、"sd" [v2.2.1+]。	player.switchClarity('od')
load(url)	{String} [必选]	无	通过视频地址加载视频。 备注：该方法只能加载对应播放模式下支持的视频格式，H5 模式支持 MP4、HLS 和 FLV (HLS、FLV 取决于浏览器是否支持)，[v2.2.2+]	player.load(http://xxx.mp4)

⚠ 注意：

以上方法必须是 `TcPlayer` 的实例化对象，且需要初始化完毕才可以调用（即 load 事件触发后）。

进阶攻略

下面介绍播放器 SDK 的进阶使用方法。

ES Module

TCPlayerLite 提供了 ES Module 版本，module name 为 `TcPlayer`，下载地址：

```
https://web.sdk.qcloud.com/player/tcplayerlite/release/v2.4.5/TcPlayer-  
module-2.4.5.js
```

开启优先 H5 播放模式

TCPlayerLite 根据不同的播放环境，播放器会选择默认最合适的播放方案。

监听事件

TCPlayerLite 以 H5 `<video>` 的规范，对播放事件做了一定程度的转换，以实现播放事件命名的统一，`TcPlayer` 对原生事件进行了捕获和透传。

- [H5 事件参考列表](#)
- [统一后的事件列表](#)

```
error  
timeupdate  
load  
loadedmetadata  
loadeddata  
progress  
fullscreen  
play  
playing  
pause  
ended  
seeking  
seeked  
resize  
volumechange  
webrtcstatupdate  
webrtcwaitstart  
webrtcwaitend  
webrtcstop
```

⚠ 注意：

- 如果通过系统控制栏进行全屏，将无法监听到 `fullscreen` 事件。
- Web 播放器的事件，依赖浏览器内置的解码器触发，Web 播放器仅透传事件。

- Web 播放器监听不到直播停止推流的事件，需要通过额外的接口来确认推流状态，请参见 [查询流状态](#)。

在非自动播放的条件下，加载视频至待播放状态，移动端和 PC 触发的事件区别。

- 移动端：

```
Object {type: "load", src: H5Video, ts: 0, detail: Object}
Object {type: "resize", src: H5Video, ts: 1150.5800000000002}
Object {type: "loadedmetadata", src: H5Video, ts: 1150.5850000000003}
Object {type: "volumechange", src: H5Video, ts: 1156.19}
Object {type: "seeking", src: H5Video, ts: 1168.665}
Object {type: "timeupdate", src: H5Video, ts: 1256.8400000000001}
Object {type: "seeked", src: H5Video, ts: 1256.85}
Object {type: "loadeddata", src: H5Video, ts: 1256.865}
Object {type: "timeupdate", src: H5Video, ts: 1256.9}
Object {type: "progress", src: H5Video, ts: 1408.7800000000002}
```

- PC：

```
Object {type: "load", src: FlashVideo, ts: 27, detail: Object}
Object {type: "loadedmetadata", src: FlashVideo, ts: 166, detail: Object}
Object {type: "volumechange", src: FlashVideo, ts: 184}
Object {type: "progress", src: FlashVideo, ts: 1741}
```

📌 说明：

以上是两种平台的差异，然而在移动端的各种设备和 App 之间同样存在差异。

事件监听函数返回的 msg 对象介绍：

名称	说明
type	事件类型。
src	事件源对象，即播放器实例，HTML5。
ts	事件触发时的 UTC 时间戳。
timeStamp	Event 实例的时间戳。

应用案例：通过事件监听，可以进行播放失败重连，[单击访问](#) 在线案例。

案例展示

结合了 TcPlayer 和即时通信 IM 的腾讯云 Web 直播互动组件：[体验地址](#)。

更新日志

TCPlayerLite 在不断更新及完善中，下面是 TCPlayerLite 发布的主版本介绍。

日期	版本	更新内容
2022.07.11	2.4.5	修复 webrtc 场景的 stopplay 方法调用时序问题。
2022.06.7	2.4.4	<ul style="list-style-type: none">新增 webrtc 降级逻辑支持支付宝环境。更新 txliveplayer 版本。
2022.05.7	2.4.3	<ul style="list-style-type: none">新增拉流开始和拉流成功事件。新增浏览器无法自动播放时触发的事件回调。新增 stop 方法，该方法会断流，包括快直播和标准直播。新增支持 amd 规范。快直播场景降级时优先判断当前环境是否支持 flv。更新 txliveplayer 版本。修复 seek 后 loading 不消失的 bug。
2022.04.20	2.4.2	修复微信 ua 变化导致的判断逻辑错误
2021.06.25	2.4.1	<ul style="list-style-type: none">新增支持 v1 信令的 WebRTC 的流地址。增加 webrtcConfig 参数。增加 WebRTC 卡顿、卡顿结束、推流结束事件。
2021.06.03	2.4.0	<ul style="list-style-type: none">增加对快直播功能的支持。修复其他已知问题。
2020.07.01	2.3.3	<ul style="list-style-type: none">修复 X5 环境下切换全屏时，事件派发异常的问题。规避 hls 切换源时，相关事件触发时机很慢，导致封面显示异常的问题。

2019. 08.20	2. 3. 2	<ul style="list-style-type: none"> ● 修改默认 hls 版本为0.12.4。 ● 修复其他已知问题。
2019. 04.26	2. 3. 1	<ul style="list-style-type: none"> ● 增加 flvConfig 参数。 ● 默认加载 flv.1.5.js。 ● 修复其他已知问题。
2019. 04.19	2. 3. 0	<ul style="list-style-type: none"> ● 增加部分功能参数选项。 ● 参数 coverpic 改为 poster。 ● destroy 销毁 flv.js 实例。 ● 修复其他已知问题。
2018. 12.17	2. 2. 3	<ul style="list-style-type: none"> ● 优化播放逻辑。 ● 解决 iOS 微信没有播放事件触发的情况下，出现 loading 动画的问题。 ● 修复其他已知问题。
2018. 05.03	2. 2. 2	<ul style="list-style-type: none"> ● 优化 loading 组件。 ● 优化 Flash destroy 方法。 ● 默认使用 H5 播放。 ● 修复已知问题。
2017. 12.20	2. 2. 1	<ul style="list-style-type: none"> ● 增加可配置清晰度文案功能。 ● 设置默认清晰度。 ● 支持切换清晰度方法。
2017. 12.07	2. 2. 1	<ul style="list-style-type: none"> ● 增加 systemFullscreen 参数。 ● 增加 flashUrl 参数。 ● 修复音量 Max 后进行静音切换的 UI 问题。 ● 修复 iOS 11 微信下需要单击两次才能播放的问题。 ● 修复 safari 11 系统样式被遮挡的问题。 ● 适配在 x5 内核会触发 seeking，但不会触发 seeked 的情况。 ● 修复进度条拖拽到起始位置，设置 currentTime 失败的问题。 ● 切换清晰度保持音量不变。 ● 修复页面宽度为0，播放器宽度判断失败问题。 ● destroy 方法增加完全销毁播放器节点。
2017. 06.30	2. 2. 0	<ul style="list-style-type: none"> ● 增加控制播放环境判断的参数：Flash、h5_flv、x5_player。 ● 调整播放器初始化逻辑，优化错误提示效果。 ● 增加 flv.js 支持，在符合条件的情况下可以采用 flv.js 播放 FLV。 ● 支持 x5-video-orientation 属性。

		<ul style="list-style-type: none">● 增加播放环境判断逻辑，可通过参数调整 H5 与 Flash 的优先级，以及是否启用 TBS 播放。● 启用版本号发布方式，避免影响旧版本的使用者。● 优化事件触发的时间戳，统一为标准时间。● Bug 修复。
2017.03.04	2.1.0	至2017.06.30，经历数次的迭代开发，逐步趋于稳定，目前文档的功能描述中，如果没有特殊说明，皆基于此版本。
2016.12.28	2.0.0	首个版本。

TCPlayerLite 升级指引

最近更新时间：2024-09-12 15:58:52

TCPlayerLite 为旧版播放器，仅包含基础直播场景的播放功能，而 TCPlayer 是兼顾直播和点播场景的完整版播放器，包含 TCPlayerLite 全部能力，同时拥有更多更强大的播放以及数据统计等功能。

若您正在使用 TCPlayerLite，建议您升级到 TCPlayer 以享受更多更全面的功能及服务，本文将为您介绍如何从 TCPlayerLite 升级为 TCPlayer。

说明：

TCPlayerLite 仍将持续维护现有能力，您可继续使用，但后续 Web 端播放器功能迭代将主要在 TCPlayer 内进行，TCPlayerLite 不再主动做功能迭代。

参见文档

- [TCPlayer](#)
- [TCPlayerLite](#)

功能展示

更直观的体验 TCPlayer，可以参见 [Web 端播放器体验](#)，可体验 TCPlayer 的各项功能并查看相关代码示例。

操作步骤

1. 替换 SDK 文件

页面引用的样式和 js 文件如下，可以参见 [TCPlayer 集成指引](#)，引用最新版本的播放器 SDK 及依赖，或从文档中下载所需文件，自行部署使用。

```
<!-- 样式文件 -->
<link
href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/tcplayer
.min.css" rel="stylesheet"/>

<!-- 依赖文件，按需使用-->
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 Webrtc 视频，需要在
tcplayer.vx.x.x.min.js 之前引入 TXLivePlayer-x.x.x.min.js。-->
<!--有些浏览器环境不支持 Webrtc，播放器会将 Webrtc 流地址自动转换为 HLS 格式地址，
因此快直播场景同样需要引入hls.min.x.xx.xm.js。-->
<script
src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/libs/TXLi
vePlayer-1.2.0.min.js"></script>
```

```
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.xml.js。-->
<script
src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/libs/hls.min.0.13.2m.js"></script>
<!--如果需要在现代浏览器中播放 FLV 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 flv.min.x.x.js。-->
<script
src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/libs/flv.min.1.5.js"></script>

<!--播放器脚本文件-->
<script
src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/tcplayer.v4.5.1.min.js"></script>
```

2. 初始化播放器

在 TCPlayer 中初始化播放器时，可以通过 URL 形式播放，也可以通过 FileID 形式播放。这里对播放 URL 进行举例说明。

2.1. 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. 通过 sources 字段播放
var player = TCPlayer('player-container-id',{
  sources: [{
    // 快直播地址
    src: 'webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1?txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=6403f7bb'
  }, {
    // HLS直播地址
    src: 'https://5664.liveplay.myqcloud.com/live/5664_harchar1.m3u8?txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=6403f7bb'
  }],
  // 可配置参数说明
  https://cloud.tencent.com/document/product/881/30820#options-.E5.8F.82.E6.95.B0.E5.88.97.E8.A1.A8
});

// 2. 通过 src 方法播放
player.src(url); // url 播放地址
```

2.2. 如果需要在直播场景设置多清晰度播放，可以参见如下方式：

```
var player = TCPlayer('player-container-id',{
  multiResolution:{
    sources:{
      'SD':[
        {
          src: 'webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=6403f7bb',
        }
      ],
      'HD':[
        {
          src: 'webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=6403f7bb',
        }
      ],
      'FHD':[
        {
          src: 'webrtc://5664.liveplay.myqcloud.com/live/5664_harchar1?
txSecret=f22a813b284137ed10d3259a7b5c224b&txTime=6403f7bb',
        }
      ]
    },
    // labels:{
    //   'SD':'标清', 'HD':'高清', 'FHD':'超清'
    // },
    showOrder: ['SD', 'HD', 'FHD'],
    defaultRes: 'SD'
  },
});
```

3. 事件监听方式

在 TCPlayer 中，监听事件的方式有所区别，所有事件参见 [API 文档](#)。

```
var player = TCPlayer('player-container-id', options);
// player.on(type, function);
player.on('error', function(error) {
  // 做一些处理
});
```


Adapter 插件

Web

最近更新时间：2024-11-05 15:22:12

本文档是介绍腾讯云视立方 Web 超级播放器 Adapter，它可以帮助腾讯云客户通过灵活的接口，快速实现第三方播放器与云点播能力的结合，实现视频播放功能。Web 超级播放器 Adapter 支持获取视频基本信息、视频流信息、关键帧与缩略图信息等，支持私有加密，本文档适合有一定 Javascript 语言基础的开发人员阅读。

SDK 集成

Web 超级播放器 Adapter 可以通过 npm 集成。

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	超级播放器签名。
hlsConfig	HlsConfig	HLS 相关设置，可使用 <code>hls.js</code> 支持的任意参数。
callback	TcAdapterCall Back	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.getStreamingOutputList();
```

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


```
    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: '15*****11',
    fileID: '528589*****783',
    psign:
      'eyJhbGciOiJIUzI1NiI*****kpXVCJ9.eyJhcHBhZCI6MTUwMDAwMjYxMSwiZmlsZU
      lkIjoIjoiNTI4NTg5MDg5MzgzODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOiJlE2MTU5NTEyMz
      ksImV4cGlyZVRpbWVtdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0Ii
      widXJsQWNjZXNzSW5mbYI6eyJ0IjoIjoiMjIxNTY1MzYyMyJ9fQ.hRrQYvC0UYtcO-
      ozB35k7LZI6E3ruvow7DC0XzdzYKE',
    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;
```

iOS

最近更新时间：2025-01-10 18:03:03

第三方播放器 iOS 插件为云点播提供给客户希望使用第三方播放器或自研播放器开发的对接云 PaaS 资源的播放器插件，常用于有自定义播放器功能需求的用户。

SDK下载

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 [控制台](#)，找到对应的文件。点开后的右侧视频详情中，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

当图片 URL 中的 QueryString 的 content_encrypt 参数被设置为 on 时，点播 CDN 会对内容进行 AES-128 加密，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;
```

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

TXCImageSpriteInfo 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

Android

最近更新时间：2025-01-10 18:03:03

第三方播放器 Android 插件为云点播提供给客户希望使用第三方播放器或自研播放器开发的对接云 PaaS 资源的播放器插件，常用于有自定义播放器功能需求的用户。

SDK 下载

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 `TXCPlayerAdapter-release-1.0.0.aar` 到 `libs` 目录，添加依赖项：

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

添加混淆脚本：

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

使用播放器

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

`fileId` 一般是在视频上传后，由服务器返回：

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

如果文件已存在腾讯云，则可以进入 [控制台](#)，找到对应的文件。点开然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

当图片 URL 中的 QueryString 的 content_encrypt 参数被设置为 on 时，点播 CDN 会对内容进行 AES-128 加密，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).lo
```

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（如果使用了子应用，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.getStreamingInfo();
```

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

TXCImageSpriteInfo 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