播放器 SDK

Web 端集成

产品文档

腾讯云

文档目录

Web 端集成
超级播放器 TCPlayer
使用文档

最近更新时间：2022-06-27 15:35:13

本文档将介绍适用于点播播放和直播播放的 Web 超级播放器（TCPlayer），它可以帮助腾讯云点播客户通过灵活的接口，快速与自有 Web 应用集成，实现视频播放功能。

概述

Web 超级播放器是通过 HTML5 的 <video> 标签以及 Flash 实现视频播放。在浏览器不支持视频播放的情况下，实现了视频播放效果的多平台统一体验，并结合腾讯云点播视频服务，提供防盗链和播放 HLS 普通加密视频等功能。

协议支持

音视频协议	用途	URL 地址格式	PC 浏览器	移动浏览器
MP3	音频	http://xxx.vod.myqcloud.com/xxx.mp3	支持	支持
MP4	点播	http://xxx.vod.myqcloud.com/xxx.mp4	支持	支持
HLS（M3U8）	直播	http://xxx.liveplay.myqcloud.com/xxx.m3u8	支持	支持
	点播	http://xxx.vod.myqcloud.com/xxx.m3u8	支持	支持
FLV	直播	http://xxx.liveplay.myqcloud.com/xxx.flv	支持	部分支持
	点播	http://xxx.vod.myqcloud.com/xxx.flv	支持	部分支持
WebRTC	直播	webrtc://xxx.liveplay.myqcloud.com/live/xxx	支持	支持

说明
- 视频编码格式仅支持 H.264 编码。
- 播放器兼容常见的浏览器，播放器内部会自动区分平台，并使用最优的播放方案。
- HLS、FLV 视频在部分浏览器环境播放需要依赖 Media Source Extensions。
- 在不支持 WebRTC 的浏览器环境，传入播放器的 WebRTC 地址会自动进行协议转换来更好的支持媒体播放。

功能支持

功能\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	✓	✓	✓	✓	✓	✓	✓	✓	✓
视频列表	✓	✓	✓	✓	✓	✓	✓	✓	✓
弱网追帧	✓	✓	✓	✓	✓	✓	✓	✓	✓

说明

- 视频编码格式仅支持 H.264 编码。
- Chrome、Firefox 包括 Windows、macOS 平台。
- Chrome、Firefox、Edge 及 QQ 浏览器播放 HLS 需要加载 hls.js。
- Referer 防盗链功能是基于 HTTP 请求头的 Referer 字段实现的，部分 Android 浏览器发起的 HTTP 请求不会携带 Referer 字段。

播放器兼容常见的浏览器，播放器内部会自动区分平台，并使用最优的播放方案。例如：在 Chrome 等现代浏览器中优先使用 HTML5 技术实现视频播放，而手机浏览器上会使用 HTML5 技术或者浏览器内核能力实现视频播放。

集成指引

通过以下步骤，您就可以在网页上添加一个视频播放器。

步骤1：在页面中引入文件

在本地的项目工程内新建 index.html 文件，html 页面内引入播放器样式文件与脚本文件：

```
<link href="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/tcplayer.min.css" rel="stylesheet"/>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 Webrtc 视频，需要在 tcplayer.vx.x.x.min.js 之前引入 TXLivePlayer-x.x.x.min.js。-->
<!--有些浏览器环境不支持 Webrtc，播放器会将 Webrtc 流地址自动转换为 HLS 格式地址，因此快直播场景同样需要引入hls.min.x.xx.xm.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/libs/TXLivePlayer-1.2.0.min.js"></script>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 协议的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.xm.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/hls.min.0.13.2m.js"></script>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 FLV 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 flv.min.x.x.x.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/libs/flv.min.1.6.2.js"></script>
<!--播放器脚本文件-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.2/tcplayer.v4.5.2.min.js"></script>
```

建议在使用播放器 SDK 的时候自行部署资源，单击下载播放器资源。
部署解压后的文件夹，不能调整文件夹里面的目录，避免资源互相引用异常。
如果您部署的地址为 aaa.xxx.ccc，在合适的地方引入播放器样式文件与脚本文件：

```
<link href="aaa.xxx.ccc/tcplayer.min.css" rel="stylesheet"/>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.m.js。-->
<script src="aaa.xxx.ccc/libs/hls.min.0.13.2m.js"></script>
<!--播放器脚本文件-->
<script src="aaa.xxx.ccc/tcplayer.vx.x.x.min.js"></script>
```

步骤2：放置播放器容器

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

```
<video id="player-container-id" width="414" height="270" preload="auto" playsinline webkit-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。超级播放器同时支持点播和直播两种播放场景，具体播放方式如下：

- 点播播放：播放器可以通过 FileID 播放腾讯云点播媒体资源，云点播具体流程请参见 使用超级播放器播放 > 接入指引 文档。
- 直播播放：播放器通过传入 URL 地址，即可拉取直播音视频流进行直播播放。腾讯云直播 URL 生成方式可参见 自主拼装直播 URL。

通过 URL 播放（直播、点播）

在页面初始化之后，调用播放器实例上的方法，将 URL 地址传入方法。

```
var player = TCPlayer('player-container-id', {}); // player-container-id 为播放器容器 ID，必须与 html 中一致
player.src(url); // url 播放地址
```

通过 FileID 播放（点播）

在 index.html 页面初始化的代码中加入以下初始化脚本，传入在准备工作中获取到的 fileID（媒资管理）中的视频 ID 与 appID（在账号信息>基本信息中查看）。

```
var player = TCPlayer('player-container-id', { // player-container-id 为播放器容器 ID，必须与 html 中一致
fileID: '3701925921299637010', // 请传入需要播放的视频 fileID（必须）
appID: '1500005696' // 请传入点播账号的 appID（必须）
//私有加密播放需填写 psign， psign 即超级播放器签名，签名介绍和生成方式参见链接：https://cloud.tencent.com/document/product/266/42436
//psign:'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwNTY5NiwiZmlsZUlkIjoiMzcwMTkyNTkyMTI5OTYzNzAxMCIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MjY4NjAxNzYsImV4cGlyZVRpbWVTdGFtcCI6MjYyNjg1OTE3OSwicGNmZyI6InByaXZhdGUiLCJ1cmxBY2Nlc3NJbmZvIjp7InQiOiI5YzkyYjBhYiJ9LCJkcm1MaWNlbnNlSW5mbyI6eyJleHBpcmVUaW1lU3RhbXAiOjI2MjY4NTkxNzksInN0cmljdE1vZGUiOjJ9fQ.Bo5K5ThInc4n8AlzIZQ-CP9a49M2mEr9-zQLH9ocQgI',
});
```

⚠ 注意：

要播放的视频建议使用腾讯云转码，原始视频无法保证在浏览器中正常播放。

步骤4: 更多功能

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

此外，播放器还提供更多其他功能，功能列表和使用方法请参见 功能展示 页面。

API 文档

最近更新时间：2022-06-27 15:35:25

本文档是介绍适用于直播和点播播放的 Web 超级播放器（ TCPlayer ）的相关参数以及 API。本文档适合有一定 Javascript 语言基础的开发人员阅读。

初始化参数

播放器初始化需要传入两个参数，第一个为播放器容器 ID，第二个为功能参数对象。

```
var player = TCPlayer('player-container-id', options);
```

options 参数列表

options 对象可配置的参数：

名称	类型	默认值	说明
appID	String	无	必选。
fileID	String	无	必选。
sources	Array	无	播放器播放地址，格式：[{ src: '//path/to/video.mp4', type: 'video/mp4' }]
width	String/Number	无	播放器区域宽度，单位像素，按需设置，可通过 CSS 控制播放器尺寸。
height	String/Number	无	播放器区域高度，单位像素，按需设置，可通过 CSS 控制播放器尺寸。
controls	Boolean	true	是否显示播放器的控制栏。
poster	String	无	设置封面图片完整地址（如果上传的视频已生成封面图，优先使用生成的封面图，详细请参见 云点播 - 管理视频 ）。
autoplay	Boolean	false	是否自动播放。
playbackRates	Array	[0.5，1，1.25，1.5，2]	设置变速播放倍率选项，仅 HTML5 播放模式有效。
loop	Boolean	false	是否循环播放。
muted	Boolean	false	是否静音播放。
preload	String	auto	是否需要预加载，有3个属性"auto"，"meta"和"none" ，移动端由于系统限制，设置 auto 无效。
swf	String	无	Flash 播放器 swf 文件的 URL。
posterImage	Boolean	true	是否显示封面。
bigPlayButton	Boolean	true	是否显示居中的播放按钮（浏览器劫持嵌入的播放按钮无法去除）。
language	String	"zh-CN"	设置语言，可选值为 "zh-CN"/"en"
languages	Object	无	设置多语言词典。
controlBar	Object	无	设置控制栏属性的参数组合，后面有详细介绍。

名称	类型	默认值	说明
reportable	Boolean	true	设置是否开启数据上报。
plugins	Object	无	设置插件功能属性的参数组合，后面有详细介绍。
hlsConfig	Object	无	hls.js 的启动配置，详细内容请参见官方文档 hls.js。
webrtcConfig	Object	无	webrtc 的启动配置，后面有详细介绍。

> ⚠ 注意：
> controls、playbackRates、loop、preload、posterImage 这些参数在浏览器劫持播放的状态下将无效（什么是劫持播放？）。

controlBar 参数列表

controlBar 参数可以配置播放器控制栏的功能，支持的属性有：

名称	类型	默认值	说明
playToggle	Boolean	true	是否显示播放、暂停切换按钮。
progressControl	Boolean	true	是否显示播放进度条。
volumePanel	Boolean	true	是否显示音量控制。
currentTimeDisplay	Boolean	true	是否显示视频当前时间。
durationDisplay	Boolean	true	是否显示视频时长。
timeDivider	Boolean	true	是否显示时间分割符。
playbackRateMenuButton	Boolean	true	是否显示播放速率选择按钮。
fullscreenToggle	Boolean	true	是否显示全屏按钮。
QualitySwitcherMenuButton	Boolean	true	是否显示清晰度切换菜单。

> ⚠ 注意：
> controlBar 参数在浏览器劫持播放的状态下将无效（什么是劫持播放？）。

plugins 插件参数列表

plugins 参数可以配置播放器插件的功能，支持的属性有：

名称	类型	默认值	说明
ContinuePlay	Object	无	控制续播功能，支持的属性如下： • auto: Boolean 是否在播放时自动续播 • text: String 提示文案 • btnText: String 按钮文案
VttThumbnail	Object	无	控制缩略图显示，支持的属性如下： • vttUrl: String vtt文件绝对地址，必传 • basePath: String 图片路径，非必须，不传时使用 vttUrl 的 path • imgUrl: String 图片绝对地址，非必须

名称	类型	默认值	说明
ProgressMarker	Boolean	无	控制进度条显示
DynamicWatermark	Object	无	控制动态水印显示，支持的属性如下： • content: String 文字水印内容，必传 • speed: Number 水印移动速度，取值范围 0-1, 非必须
ContextMenu	Object	无	可选值如下： • mirror: Boolean 控制是否支持镜像显示 • statistic: Boolean 控制是否支持显示数据面板 • levelSwitch: Object 控制切换清晰度时的文案提示 • { • open: Boolean 是否开启提示 • switchingText: String, 开始切换清晰度时的提示文案 • switchedText: String, 切换成功时的提示文案 • switchErrorText: String, 切换失败时的提示文案 • }

webrtcConfig 参数列表

webrtcConfig 参数来控制播放 webrtc 过程中的行为表现，支持的属性有：

名称	类型	默认值	说明
connectRetryCount	Number	3	SDK 与服务器重连次数
connectRetryDelay	Number	1	SDK 与服务器重连延时
receiveVideo	Boolean	true	是否拉取视频流
receiveAudio	Boolean	true	是否拉取音频流
showLog	Boolean	false	是否在控制台打印日志

对象方法

初始化播放器返回对象的方法列表：

名称	参数及类型	返回值及类型	说明
src()	(String)	无	设置播放地址。
ready(function)	(Function)	无	设置播放器初始化完成后的回调。
play()	无	无	播放以及恢复播放。
pause()	无	无	暂停播放。
currentTime(seconds)	(Number)	(Number)	获取当前播放时间点，或者设置播放时间点，该时间点不能超过视频时长。
duration()	无	(Number)	获取视频时长。
volume(percent)	(Number)[0，1][可选]	(Number)/设置时无返回	获取或设置播放器音量。

名称	参数及类型	返回值及类型	说明
poster(src)	(String)	(String)/设置时无返回	获取或设置播放器封面。
requestFullscreen()	无	无	进入全屏模式。
exitFullscreen()	无	无	退出全屏模式。
isFullscreen()	无	Boolean	返回是否进入了全屏模式。
on(type，listener)	(String, Function)	无	监听事件。
one(type，listener)	(String, Function)	无	监听事件，事件处理函数最多只执行1次。
off(type，listener)	(String, Function)	无	解绑事件监听。
buffered()	无	TimeRanges	返回视频缓冲区间。
bufferedPercent()	无	值范围[0，1]	返回缓冲长度占视频时长的百分比。
width()	(Number)[可选]	(Number)/设置时无返回	获取或设置播放器区域宽度，如果通过 CSS 设置播放器尺寸，该方法将无效。
height()	(Number)[可选]	(Number)/设置时无返回	获取或设置播放器区域高度，如果通过 CSS 设置播放器尺寸，该方法将无效。
videoWidth()	无	(Number)	获取视频分辨率的宽度。
videoHeight()	无	(Number)	获取视频分辨率的高度。
dispose()	无	无	销毁播放器。

> ⚠ 注意：
> 对象方法不能同步调用，需要在相应的事件（如 loadedmetadata）触发后才可以调用，除了 ready、on、one 以及 off。

事件

播放器可以通过初始化返回的对象进行事件监听，示例：

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

其中 type 为事件类型，支持的事件有：

名称	介绍
play	已经开始播放，调用 play() 方法或者设置了 autoplay 为 true 且生效时触发，这时 paused 属性为 false。

名称	介绍
playing	因缓冲而暂停或停止后恢复播放时触发，paused 属性为 false 。通常用这个事件来标记视频真正播放，play 事件只是开始播放，画面并没有开始渲染。
loadstart	开始加载数据时触发。
durationchange	视频的时长数据发生变化时触发。
loadedmetadata	已加载视频的 metadata。
loadeddata	当前帧的数据已加载，但没有足够的数据来播放视频的下一帧时，触发该事件。
progress	在获取到媒体数据时触发。
canplay	当播放器能够开始播放视频时触发。
canplaythrough	当播放器预计能够在不停下来进行缓冲的情况下持续播放指定的视频时触发。
error	视频播放出现错误时触发。
pause	暂停时触发。
ratechange	播放速率变更时触发。
seeked	搜寻指定播放位置结束时触发。
seeking	搜寻指定播放位置开始时触发。
timeupdate	当前播放位置有变更，可以理解为 currentTime 有变更。
volumechange	设置音量或者 muted 属性值变更时触发。
waiting	播放停止，下一帧内容不可用时触发。
ended	视频播放已结束时触发。此时 currentTime 值等于媒体资源最大值。
resolutionswitching	清晰度切换进行中。
resolutionswitched	清晰度切换完毕。
fullscreenchange	全屏状态切换时触发。
webrtcevent	播放 webrtc 时的事件集合。
webrtcstats	播放 webrtc 时的统计数据。

WebrtcEvent 列表

播放器可以通过 webrtcevent 获取播发 webrtc 过程中的所有事件，示例：

```
var player = TCPlayer('player-container-id', options);
player.on('webrtcevent', function(event) {
// 从回调参数 event 中获取事件状态码及相关数据
});
```

webrtcevent 状态码如下

状态码	回调参数	介绍
1001	无	开始拉流
1002	无	已经连接服务器
1003	无	视频播放开始
1004	无	停止拉流，结束视频播放
1005	无	连接服务器失败，已启动自动重连恢复
1006	无	获取流数据为空
1007	localSdp	开始请求信令服务器
1008	remoteSdp	请求信令服务器成功
1009	无	拉流卡顿等待缓冲中
1010	无	拉流卡顿结束恢复播放

错误码

当播放器触发 error 事件时，监听函数会返回错误码，其中3位数以上的错误码为媒体数据接口错误码。错误码列表：

名称	描述
-1	播放器没有检测到可用的视频地址。
-2	获取视频数据超时。
1	视频数据加载过程中被中断。 可能原因： • 网络中断。 • 浏览器异常中断。 解决方案： • 查看浏览器控制台网络请求信息，确认网络请求是否正常。 • 重新进行播放流程。
2	由于网络问题造成加载视频失败。 可能原因：网络中断。 解决方案： • 查看浏览器控制台网络请求信息，确认网络请求是否正常。 • 重新进行播放流程。
3	视频解码时发生错误。 可能原因：视频数据异常，解码器解码失败。 解决方案： • 尝试重新转码再进行播放，排除由于转码流程引入的问题。 • 确认原始视频是否正常。 • 请联系技术客服并提供播放参数进行定位排查。

名称	描述
4	视频因格式不支持或者服务器或网络的问题无法加载。 可能原因： • 获取不到视频数据，CDN 资源不存在或者没有返回视频数据。 • 当前播放环境不支持播放该视频格式。 解决方案： • 查看浏览器控制台网络请求信息，确认视频数据请求是否正常。 • 确认是否按照使用文档加载了对应视频格式的播放脚本。 • 确认当前浏览器和页面环境是否支持将要播放的视频格式。 • 请联系技术客服并提供播放参数进行定位排查。
5	视频解密时发生错误。 可能原因： • 解密用的密钥不正确。 • 请求密钥接口返回异常。 • 当前播放环境不支持视频解密功能。 解决方案： • 确认密钥是否正确，以及密钥接口是否返回正常。 • 请联系技术客服并提供播放参数进行定位排查。
10	点播媒体数据接口请求超时。在获取媒体数据时，播放器重试3次后仍没有任何响应，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
11	点播媒体数据接口没有返回数据。在获取媒体数据时，播放器重试3次后仍没有数据返回，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
12	点播媒体数据接口返回异常数据。在获取媒体数据时，播放器重试3次后仍返回无法解析的数据，会抛出该错误。 可能原因： • 当前网络环境无法连接到媒体数据接口，或者媒体数据接口被劫持。 • 播放参数有误，媒体数据接口无法处理。 • 媒体数据接口异常。 解决方案： • 尝试打开我们提供的 Demo 页面看是否可以正常播放。 • 请联系技术客服并提供播放参数进行定位排查。
13	播放器没有检测到可以在当前播放器播放的视频数据，请对该视频进行转码操作。
14	HTML5 + hls.js 模式下播放 hls 出现网络异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Network Errors。
15	HTML5 + hls.js 模式下播放 hls 出现多媒体异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Media Errors。
16	HTML5 + hls.js 模式下播放 hls 出现多路复用异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Mux Errors。

名称	描述
17	HTML5 + hls.js 模式下播放 hls 出现其他异常，异常详情可在 event.source 中查看，详细介绍请看 hls.js 的官方文档 Other Errors。
10008	媒体数据服务没有找到对应播放参数的媒体数据，请确认请求参数 appID fileID 是否正确，以及对应的媒体数据是否已经被删除。

TCPlayerLite（旧）

最近更新时间：2022-04-19 15:37:32

> 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视立方 SDK 的 Web 播放器的功能特性以及对接攻略：

点击查看视频

协议支持

Web 超级播放器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。

对于 PC 浏览器，因为其目前还没有抛弃 Flash 控件，而 Flash 控件支持的视频源格式较多，并且浏览器上的 Flash 控件都是 Adobe 自己研发，所以兼容性很好。

视频协议	用途	URL 地址格式	PC 浏览器	移动浏览器
WebRTC	直播	webrtc://xxx.liveplay.myqcloud.com/live/xxx	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	不支持
	点播	http://xxx.vod.myqcloud.com/xxx.flv	支持	不支持
RTMP	直播	rtmp://xxx.liveplay.myqcloud.com/live/xxx	支持	不支持

> 注意：
>
> - 播放 RTMP 格式的视频必须启用 Flash，目前浏览器默认禁用 Flash，需用户手动开启。
> - 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.qcloud.com/player/tcplayerlite/release/v2.4.1/TcPlayer-2.4.1.js" charset="utf-8"></script>;
```

建议在使用腾讯云视立方播放器 UGSV 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-2.4.1.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>
```

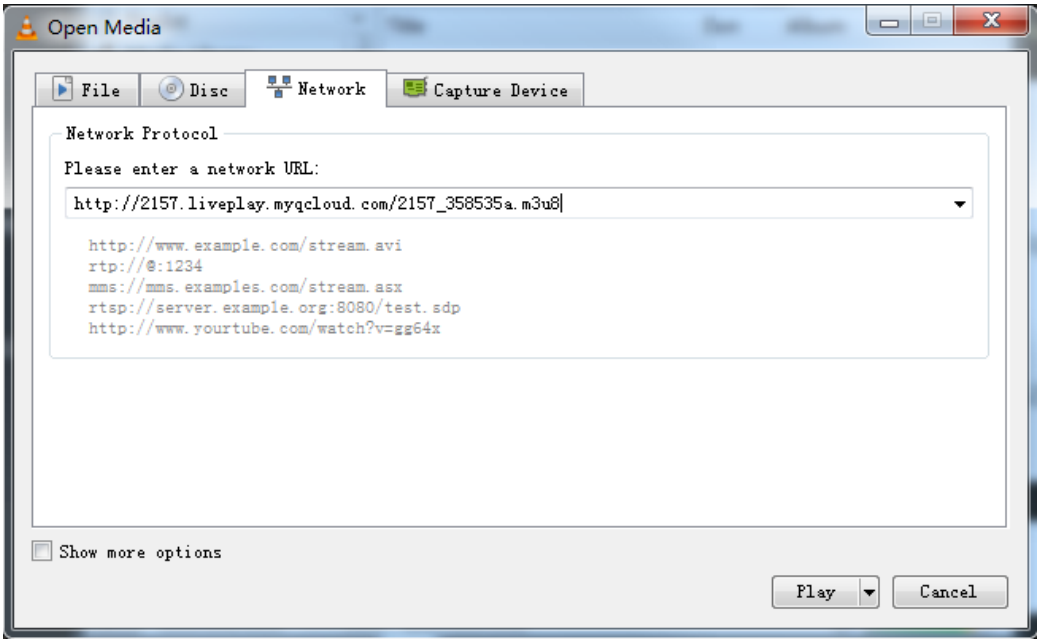
Step3. 对接视频播放

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

PC 浏览器支持 Flash，其 Javascript 代码如下：

```
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视频直播服务，则无需考虑，因为腾讯云的直播频道默认支持 WebRTC、FLV、RTMP 和 HLS（M3U8）播放协议。

3.3 解决无法播放问题

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 和 RTMP 协议，最新版本的 QQ 浏览器支持 FLV 协议的播放。

- **原因四：跨域安全问题**
  PC 浏览器的视频播放基于 Flash 控件实现，但 **Flash 控件会做跨域访问检查**，如果播放视频所存放的服务器没有部署跨域策略，则会出现问题。
  解决方法：在视频存储服务器根域名下添加跨域配置文件 crossdomain.xml，并配置 Flash swf 所在域名，以允许 Flash 和 JavaScript 跨域播放视频。
  播放器的 Flash swf 文件默认存放在 imgcache.qq.com 域名下，如需部署到自己的服务器上，可自行下载并部署：swf 文件地址。
  如果是在域名限制区域，需要的播放器的 Flash swf 文件默认存放在 cloudcache.tencent-cloud.com 域名下，并且在播放器初始化的时候传递 flashUrl。

```
<cross-domain-policy>
<allow-access-from domain="*.*.com" secure="false"/>
</cross-domain-policy>
```

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，请参见 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】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 或者 Flash 插件未启用。
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。	-
13	直播已结束，请稍后再来。	RTMP 正常播放过程中触发事件（NetConnection.Connect.Closed）。 Flash 提示的错误。
1001	网络错误，请检查网络配置或者播放链接是否正确。	网络已断开（NetConnection.Connect.Closed）。 Flash 提示的错误。
1002	获取视频失败，请检查播放链接是否有效。	拉取播放文件失败（NetStream.Play.StreamNotFound），可能是服务器错误或者视频文件不存在。 Flash 提示的错误。
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

Code	提示语	说明
2032	获取视频失败，请检查播放链接是否有效。	Flash 提示的错误。
2048	无法加载视频文件，跨域访问被拒绝。	请求 M3U8 文件失败，可能是网络错误或者跨域问题。 Flash 提示的错误。

说明：

- Code 1 – 4 对应的是 H5 原生事件。
- 由于 Flash 的黑盒特性以及 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，请参见 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

参数	类型	默认值	参数说明
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
rtmp	String	无	原画 RTMP 播放 URL。 示例：rtmp://2157.liveplay.myqcloud.com/live/2157_280d88
rtmp_hd	String	无	高清 RTMP 播放 URL。 示例：rtmp://2157.liveplay.myqcloud.com/live/2157_280d88hd
rtmp_sd	String	无	标清 RTMP 播放 URL。 示例：rtmp://2157.liveplay.myqcloud.com/live/2157_280d88sd
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： "http://www.test.com/myimage.jpg" 或者 {"style": "cover", "src": http://www.test.com/myimage.jpg} [v2.3.0+]
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。 （备注：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

参数	类型	默认值	参数说明
flash	Boolean	true	是否优先使用 Flash 播放视频。 （*备注：该选项只对 PC 平台生效 *[v2.2.0+]） 示例：true
flashUrl	String	无	可以设置 flash swf url。 （**备注：该选项只对 PC 平台生效** [v2.2.1+]）
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：0（landscape 横屏），1:（portraint竖屏），2:（landscape
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+]。

参数	类型	默认值	参数说明
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： • 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，移动直播 SDK，iOS 不支持引入 B 帧，Android 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(val)	{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)

方法	参数	返回值	说明	示例
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，Flash 模式支持切换 RTMP、FLV、HLS 和 MP4，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

下面介绍腾讯云视立方 SDK 在直播播放场景下的进阶使用方法。

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/v2.4.1/TcPlayer-module-2.4.1.js
```

开启优先 H5 播放模式

TCPlayerLite 采用 H5 <video> 和 Flash 相结合的方式来进行视频播放，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。

虽然浏览器厂商已经开始逐步放弃对 Flash 插件的支持，但是在国内仍有大量的浏览器不支持 MSE，在播放 FLV 和 HLS（M3U8）时无法切换到 H5 <video> 模式，而播放 RTMP 必须使用 Flash。
因此，TCPlayerLite 默认优先启用 Flash 播放模式，如果在检测到 Flash 插件不可用的情况下，将采用 H5 <video> 进行播放。

> ⓘ 说明：
> 默认 Flash 模式的原因是 Flash 支持的视频格式最广，而 H5 <video> 默认只支持 MP4（H.264），在特定条件下才支持 HLS（M3U8）和 FLV。

从2.2.0版本开始，提供了可以设置播放模式优先级的属性，如果想优先采用 H5 <video> 播放模式，则需要把 Flash 属性设置为 False；如果 H5 <video> 不可用，则采用 Flash 播放；如果没有检测到 Flash 插件，则会提示“当前系统环境不支持播放该视频格式”。

监听事件

TCPlayerLite 是采用 H5 <video> 和 Flash 相结合的方式来进行视频播放，由于两种方式播放视频时触发的事件不尽相同，所以我们以 H5 <video> 的规范，对 Flash 的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 对这两种播放方式所触发的原生事件进行了捕获和透传。

- H5 事件参考列表
- Flash 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和 Flash 插件触发，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。

- Flash 模式下特有的事件：netStatus。

> 说明：
> 由于 Flash 的黑盒特性以及 H5 视频播放标准在各个平台终端的实现不一致性，事件的触发方式和结果会有差异。

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 Flash 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 Flash：

```
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 或者 Flash。
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 TCPlayerLite 和即时通信 IM 的腾讯云 Web 直播互动组件，具体请参见 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
2021.06.25	2.4.1	• 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	• 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	• 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	• 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	• 增加 fivConfig 参数。 • 默认加载 flv.1.5.js。 • 修复其他已知问题。
2019.04.19	2.3.0	• 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	• 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	• 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	• 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	• 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	• 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。 • 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：2022-04-19 15:36:02

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 接入文档，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/TXLivePlayer-1.2.0.min.js"></script>
<!--如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HLS 格式的视频，需要在 tcplayer.vx.x.x.min.js 之前引入 hls.min.x.xx.xm.js。-->
<script src="https://web.sdk.qcloud.com/player/tcplayer/release/v4.5.1/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

1. 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. 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
```

3. 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
});
```

超级播放器 Adapter

最近更新时间：2022-04-14 10:56:59

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 提供 **CDN 集成**和 **npm 集成**两种集成方式：

CDN 集成

在需要播放视频的页面中引入初始化脚本，脚本会在全局下暴露 TcAdapter 变量。

```
<script src="https://cloudcache.tencentcs.com/qcloud/video/dist/tcadapter.1.0.0.min.js"></script>
```

npm 集成

```
// npm install
npm install tcadapter --save

// import TcAdapter
import TcAdapter from 'tcadapter';
```

放置播放器容器

在需要展示播放器的页面加入容器，TcAdapter 仅需要承载播放视频的容器，播放样式和自定义功能可由第三方播放器或使用者自行实现：

```
<video id="player-container-id">
</video>
```

SDK 使用说明

检测开发环境

检测当前环境是否支持 TcAdapter。

```
TcAdapter.isSupported();
```

初始化 Adapter

初始化 Adapter，创建 Adapter 实例。初始化过程会请求腾讯云点播服务器，获取视频文件信息。

接口

```
const adapter = new TcAdapter('player-container-id', {
fileID: string,
```

```
appID: string,
psign: string,
hlsConfig: {}
}, callback);
```

参数说明

参数名	类型	描述
appID	String	点播账号的 APPID
fileID	String	要播放的视频 fileId
psign	String	超级播放器签名
hlsConfig	HlsConfig	HLS 相关设置，可使用 hls.js 支持的任意参数
callback	TcAdapterCallBack	初始化完成回调，可以在此方法之后获取视频基本信息

> ⚠ 注意：
> TcAdapter 底层基于 hls.js 实现，可以通过 HlsConfig 接收 hls.js 支持的任意参数，用于对播放行为的精细调整。

获取视频基本信息

获取视频的信息， 必须是在初始化之后才生效。

接口

```
VideoBasicInfo adapter.getVideoBasicInfo();
```

参数说明

VideoBasicInfo 参数如下：

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

获取视频流信息

接口

```
List<StreamingOutput> adapter.getStreamimgOutputList();
```

参数说明

StreamingOutput 参数如下：

参数名	类型	描述
drmType	String	自适应码流保护类型，目前取值有 plain 和 simpleAES。plain 表示不加密，simpleAES 表示 HLS 普通加密
playUrl	String	播放 URL
subStreams	List	自适应码流子流信息，类型为 SubStreamInfo

SubStreamInfo 参数如下：

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

获取关键帧打点信息

接口

```
List<KeyFrameDescInfo> adapter.getKeyFrameDescInfo();
```

参数说明

KeyFrameDescInfo 参数如下：

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

获取缩略图信息

接口

```
ImageSpriteInfo adapter.getImageSpriteInfo();
```

参数说明

ImageSpriteInfo 参数如下：

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

监听事件

播放器可以通过初始化返回的对象进行事件监听，示例：

```
const adapter = TcAdapter('player-container-id', options);
adapter.on(TcAdapter.TcAdapterEvents.Error, function(error) {
// do something
});
```

其中 type 为事件类型，支持的事件包括 HLS 原生的事件以及以下事件，可从 TcAdapter.TcAdapterEvents 中访问到事件名称：

名称	介绍
LOADEDMETADATA	通过 playcgi 获取到了相应的视频信息，在此事件回调中可以获取视频相关信息
HLSREADY	hls实例创建完成，可以在此时机调用 hls 实例对象上的各种属性和方法
ERROR	出现错误时触发，可从回调参数中查看失败具体原因

获取 Hls 实例

adapter 底层基于 hls.js 实现，可以通过 adapter 实例访问到 HLS 实例以及实例上的属性和方法，用于实现对播放流程的精细控制。

```
adapter.on('hlsready', () => {
const hls = adapter.hls;
// ...
})
```

> 说明：
> 具体请参见 hls.js。

示例

例1：在 React 中使用 TcAdapter

具体示例，请参见 GitHub。

```
import { useEffect, useRef } from 'react';
import TcAdapter from 'tcadapter';

function App() {
if (!TcAdapter.isSupported()) {
throw new Error('current environment can not support TcAdapter');
}

const videoRef = useRef(null);
useEffect(() => {
const adapter = new TcAdapter(videoRef.current, {
appID: '1500002611',
fileID: '5285890813738446783',
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NTg5MDgxMzczODQ0Nj
```

```
c4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2ljRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNTY1MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
}, () => {
console.log('basicInfo', adapter.getVideoBasicInfo());
});

adapter.on(TcAdapter.TcAdapterEvents.HLSREADY, () => {
const hls = adapter.hls;
// ...
})
}, []);


const play = () => {
videoRef.current.play();
}

return (
<div>
<div>
<video id="player" ref={ videoRef }></video>
</div>
<button onClick={play}>play</button>
</div>
);
}

export default App;
```

例2：TcAdapter 与 videojs 结合

具体示例，请参见 GitHub。

```
// 1. videojs 播放 hls 会使用 @videojs/http-streaming，所以我们开发一套使用 tcadapter 播放的策略覆盖原有逻辑（也可以直接修改 @videojs/http-streaming 内部逻辑）

// src/js/index.js
import videojs from './video';
import '@videojs/http-streaming';
import './tech/tcadapter'; // 新增逻辑
export default videojs;


// src/js/tech/tcadapter.js
import videojs from '../video.js';
import TcAdapter from 'tcadapter';
```

```
class Adapter {
constructor(source, tech, options) {
const el = tech.el();
// 获取参数并初始化实例
const adapter = new TcAdapter(el, {
appID: '1500002611',
fileID: '5285890813738446783',
psign: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBJZCI6MTUwMDAwMjYxMSwiZmlsZUlkIjoiNTI4NTg5MDgxMzczODQ0Njc4MyIsImN1cnJlbnRUaW1lU3RhbXAiOjE2MTU5NTEyMzksImV4cGlyZVRpbWVTdGFtcCI6MjIxNTY1MzYyMywicGNmZyI6ImJhc2IjRHJtUHJlc2V0IiwidXJsQWNjZXNzSW5mbyI6eyJ0IjoiMjIxNTY1MzYyMyJ9fQ.hRrQYvC0UYtcO-ozB35k7LZI6E3ruvow7DC0XzzdYKE',
hlsConfig: {},
});
adapter.on(TcAdapter.TcAdapterEvents.LEVEL_LOADED, this.onLevelLoaded.bind(this));
}

dispose() {
this.hls.destroy();
}

onLevelLoaded(event) {
this._duration = event.data.details.live ? Infinity : event.data.details.totalduration;
}

}

let hlsTypeRE = /^application\/(x-mpegURL|vnd\.apple\.mpegURL)$/i;
let hlsExtRE = /\.m3u8/i;

let HlsSourceHandler = {
name: 'hlsSourceHandler',
canHandleSource: function (source) {
// skip hls fairplay, need to use Safari resolve it.
if (source.skipHlsJs || (source.keySystems && source.keySystems['com.apple.fps.1_0'])) {
return '';
} else if (hlsTypeRE.test(source.type)) {
return 'probably';
} else if (hlsExtRE.test(source.src)) {
return 'maybe';
} else {
return '';
}
},

handleSource: function (source, tech, options) {
if (tech.hlsProvider) {
tech.hlsProvider.dispose();
```

```
tech.hlsProvider = null;
} else {
// hls关闭自动加载后，需要手动加载资源
if (options.hlsConfig && options.hlsConfig.autoStartLoad === false) {
tech.on('play', function () {
if (!this.player().hasStarted()) {
this.hlsProvider.hls.startLoad();
}
});
}
}
tech.hlsProvider = new Adapter(source, tech, options);
return tech.hlsProvider;
},
canPlayType: function (type) {
if (hlsTypeRE.test(type)) {
return 'probably';
}
return '';
}
};

function mountHlsProvider(enforce) {
if (TcAdapter && TcAdapter.isSupported() || !!enforce) {
try {
let html5Tech = videojs.getTech && videojs.getTech('Html5');
if (html5Tech) {
html5Tech.registerSourceHandler(HlsSourceHandler, 0);
}
} catch (e) {
console.error('hls.js init failed');
}
} else {
//没有引入tcadapter 或者 MSE 不可用或者x5内核禁用
}
}
mountHlsProvider();
export default Adapter;
```