

- h5player
 - 描述
 - 版本
 - 媒体版本限制
 - 使用注意事项
 - 浏览器限制以及编解码支持情况
 - 接口调用流程图
 - 引入
 - 创建实例
 - 接口说明
 - 事件初始化: JS_SetWindowControlCallback(events)
 - 播放: JS_Play(url, config, windowIndex, startTime, endTime)
 - 停止播放: JS_Stop(windowIndex)
 - 停止所有播放: JS_StopRealPlayAll()
 - 开启声音: JS_OpenSound(windowIndex)
 - 关闭声音: JS_CloseSound(windowIndex)
 - 设置音量: JS_SetVolume(windowIndex, volumn)
 - 获取当前音量: JS_GetVolume(windowIndex)
 - 录像: JS_StartSaveEx(windowIndex, fileName, idstType)
 - 停止录像并保存文件: JS_StopSave(windowIndex)
 - 抓图: JS_CapturePicture(windowIndex, fileName, fileType, callback)
 - 回放 : JS_Play()
 - 停止回放 : JS_Stop()
 - 暂停回放: JS_Pause(windowIndex)
 - 恢复回放: JS_Resume(windowIndex)
 - 开始对讲: JS_StartTalk(szTalkUrl)
 - 停止对讲: JS_StopTalk()
 - 设置对讲音量 : JS_TalkSetVolume(nVolume)
 - 获取对讲音量: JS_TalkGetVolume()
 - 录像、抓图功能同预览播放
 - 回放快放: JS_Fast(windowIndex)
 - 回放慢放: JS_Slow(windowIndex)
 - 回放定位: JS_Seek(windowIndex, stratTime, endTime)
 - 回放单帧进 (高级模式功能) : JS_FrameForward(windowIndex)
 - 电子放大 (高级模式功能) : JS_EnableZoom(windowIndex)
 - 开启/关闭智能信息展示 (高级模式功能) : JS_RenderALLPrivateData(iWndNum, bOpenFlag)
 - 分屏: JS_ArrangeWindow(splitNum)
 - 整体全屏: JS_FullScreenDisplay(isFull)
 - 单窗口全屏: JS_FullScreenSingle(windowIndex)
 - 设置窗口大小 : JS_Resize (iWidth, iHeight)
 - 获取OSD时间 : JS_GetOSDTime (windowIndex)
 - 获取音视频信息 : JS_GetVideoInfo (windowIndex)

- 设置取流连接超时时间：JS_SetConnectTimeOut (windowIndex, nTime)
- 错误码及其描述
- FAQ

h5player

描述

h5player是一个基于HTML5的流式网络视频播放器，无需安装浏览器插件即可通过websocket协议向媒体服务取
流播放多种格式的音视频流。

版本

当前版本 2.0.0

媒体版本限制

媒体网关:mgc V5.11.101003 或 mgc V5.13.100版本及以上

使用注意事项

- 1、需要在web服务器返回的响应头增加跨域隔离字段：Cross-Origin-Embedder-Policy: require-corp Cross-Origin-Opener-Policy: same-origin 并在https环境下使用。否则高级模式无法使用
- 2、在集成过程中new JSPlugin时候必填szBasePath: './dist', // 必填,引用H5player.min.js的js相对路径，否则会引起内部加载解码库异常

浏览器限制以及编解码支持情况

以下数据都是在pc chrome80+测试所得

解码方式	浏览器限制
高级模式	当前支持 Chrome80+ , ios safari, andriod browser
普通模式	除 IE 和 IOS Safari 外 , 基本都支持

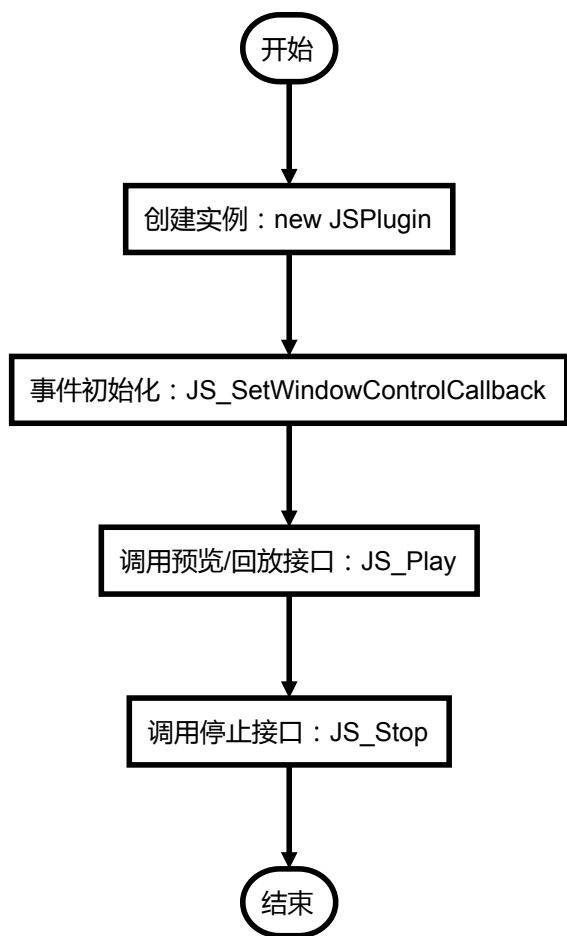
[illegible]

高级模式	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
普通模式	✓	✓	✓	×	×	×	×	×	×	×

解码方式	视频编码格式	分辨率	视频参数 (bps*fps)	解码路数 (系统 : Win7)		
				CPU : I7_8700K	CPU : I5-9400/F	CPU : I3-8100
				显卡 : RTX2080	显卡 : GTX1050TI	显卡 : GT1030D5
高级模式	H265	1080P	4M * 25	6	4	3
		720P	2M * 25	14	10	6
	H264	1080P	4M * 25	6	4	3
		720P	2M * 25	14	10	6
普通模式	H265(采用高级模式解码)	1080P	4M * 25	6	4	3
		720P	2M * 25	14	10	6
	H264	1080P	4M * 25	24	20	9
		720P	2M * 25	32	32	16

接口调用流程图

其余控制接口调用都在JS_Play后调用



引入

直接用<script>标签引入

```
<!-- h5player -->  
<script src="h5player.min.js"></script>
```

创建实例

```
<body>
  <div id="play_window"></div>
  <script>
    var curIndex = 0; // 当前窗口下标
    var myPlugin = new JSPlugin({
      szId: 'play_window', //需要英文字母开头 必填
      szBasePath: './dist', // 必填,引用H5player.min.js的js相对路径

      // 当容器div#play_window有固定宽高时,可不传iWidth和iHeight,窗口大小将自适应容器宽高
      // iWidth: 600,
      // iHeight: 400,

      // 分屏播放,默认最大分屏4*4
      // iMaxSplit: 4,
      // iCurrentSplit: 1,

      // 样式
      // oStyle: {
      //   border: "#343434",
      //   borderSelect: "#FFCC00",
      //   background: "#000"
      // }
    })
  </script>
</body>
```

接口说明

事件初始化: JS_SetWindowControlCallback(events)

参数：

参数名	类型	说明	必需
events	Object	事件对应的处理函数集合	是

返回：Promise

```
JS_SetWindowControlCallback({
  windowEventSelect: function (index) { //插件选中窗口回调
    curIndex = index;
    console.log(index, iErrorCode, oError);
  },
  pluginErrorHandler: function (index, iErrorCode, oError) { //插件错误回调
    // do you want...
  },
  windowEventOver: function (index) { //鼠标移过回调
    // do you want...
  },
  windowEventOut: function (index) { //鼠标移出回调
    // do you want...
  },
  windowEventUp: function (index) { //鼠标mouseup事件回调
    // do you want...
  },
  windowFullCscreenChange: function (bFull) { //全屏切换回调
    // do you want...
  },
  firstFrameDisplay: function (index, iWidth, iHeight) { //首帧显示回调
    // do you want...
  },
  performanceLack: function () { //性能不足回调
    // do you want...
  },
  StreamEnd: function () { //回放结束回调
    // do you want...
  }
});
```

播放: JS_Play(url, config, windowIndex, startTime, endTime)

参数：

参数名	类型	说明	必需
url	String	流媒体URL	是
config	Object	播放配置	流媒体必传{ playURL: " }
windowIndex	Number	当前窗口下标	是
startTime	DateTime	回放开始时间，格式类型：2021-06-29T00:00:00Z	预览不能填，回放必填
endTime	DateTime	回放结束时间，格式类型：2021-06-29T00:00:00Z	预览不能填，回放必填

返回：Promise

```
let url = document.getElementById('url').value;
let startTime, endTime;
myPlugin.JS_Play(
    url,
    {
        playURL: url, // 流媒体播放时必须传
        mode: 0, // 解码类型：0=普通模式；1=高级模式 默认为0

        // 设置直连时的认证参数等
        // ...
    },
    curIndex, //当前窗口下标

    // 回放参数
    startTime,
    endTime,
)
)
```

停止播放: JS_Stop(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认停止当前选中窗口

返回：Promise

```
myPlugin.JS_Stop(curIndex).then(
    () => {
        console.info('JS_Stop success');
        // do you want...
    },
    (err) => {
        console.info('JS_Stop failed');
        // do you want...
    }
);
```

停止所有播放: JS_StopRealPlayAll()

参数：无

返回：Promise

```
myPlugin.JS_StopRealPlayAll().then(  
  () => {  
    console.info('JS_StopRealPlayAll success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_StopRealPlayAll failed');  
    // do you want...  
  }  
);
```

开启声音: JS_OpenSound(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认开启当前选中窗口的声音

返回：Promise

```
myPlugin.JS_OpenSound(curIndex).then(  
  () => {  
    console.info('JS_OpenSound success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_OpenSound failed');  
    // do you want...  
  }  
);
```

关闭声音: JS_CloseSound(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认关闭当前选中窗口的声音

返回：Promise


```
myPlugin.JS_CloseSound(curIndex).then(
  () => {
    console.info('JS_CloseSound success');
    // do you want...
  },
  (err) => {
    console.info('JS_CloseSound failed');
    // do you want...
  }
);
```

设置音量: JS_SetVolume(windowIndex, volumn)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	是
volumn	Number	音量大小	是，范围1~100

返回：Promise

```
myPlugin.JS_SetVolume(curIndex, volumn).then(
  () => {
    console.info('JS_SetVolume success');
    // do you want...
  },
  (err) => {
    console.info('JS_SetVolume failed');
    // do you want...
  }
);
```

获取当前音量: JS_GetVolume(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认获取当前选中窗口的音量

返回：Promise

```
myPlugin.JS_GetVolume(curIndex).then(  
  (volume) => {  
    console.info('JS_GetVolume success', volume);  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_GetVolume failed');  
    // do you want...  
  }  
);
```

录像: JS_StartSaveEx(windowIndex, fileName, idstType)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	是
fileName	String	文件名	是，可不带后缀，默认为.mp4
idstType	Number	录像文件类型	是，2-ps 5-mp4 ,mp4录制音频限制，仅支持AAC、G711A、G711U

返回：Promise

```
let fileName = 'fileName.mp4';  
myPlugin.JS_StartSaveEx(curIndex, fileName, 2).then(  
  () => {  
    console.info('JS_StartSave success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_StartSave failed');  
    // do you want...  
  }  
);
```

停止录像并保存文件: JS_StopSave(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	是

返回：Promise

```
let fileName = 'fileName.mp4';
myPlugin.JS_StopSave(windowIndex).then(
  () => {
    console.info('JS_StopSave success');
    // do you want...
  },
  (err) => {
    console.info('JS_StopSave failed');
    // do you want...
  }
);
```

抓图: JS_CapturePicture(windowIndex, fileName, fileType, callback)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	是
fileName	String	文件名	是
fileType	String	文件类型	是，'BMP'/'JPEG'
callback	Fuction	回调函数	否

返回：Promise

```
let fileName = 'img';
let fileType = 'BMP';
myPlugin.JS_CapturePicture(curIndex, fileName, fileType).then(
  () => {
    console.info('JS_CapturePicture success');
    // do you want...
  },
  (err) => {
    console.info('JS_CapturePicture failed');
    // do you want...
  }
);
```

回放：JS_Play()

停止回放：JS_Stop()

暂停回放: JS_Pause(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认暂停当前选中窗口

返回：Promise

```
myPlugin.JS_Pause(curIndex).then(  
  () => {  
    console.info('JS_Pause success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_Pause failed');  
    // do you want...  
  }  
);
```

恢复回放: JS_Resume(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认恢复当前选中窗口

返回：Promise

```
myPlugin.JS_Resume(curIndex).then(  
  () => {  
    console.info('JS_Resume success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_Resume failed');  
    // do you want...  
  }  
);
```

开始对讲: JS_StartTalk(szTalkUrl)

特殊说明：该功能只能在https域下使用，是浏览器的限制

参数：

参数名	类型	说明	必需
szTalkUrl	String	对讲URL	是

返回：Promise

```
myPlugin.JS_StartTalk(szTalkUrl).then(
    () => {
        console.info('JS_StartTalk success');
        // do you want...
    },
    (err) => {
        console.info('JS_StartTalk failed');
        // do you want...
    }
);
```

停止对讲: JS_StopTalk()

返回：Promise

```
myPlugin.JS_StopTalk().then(
    () => {
        console.info('JS_StopTalk success');
        // do you want...
    },
    (err) => {
        console.info('JS_StopTalk failed');
        // do you want...
    }
);
```

设置对讲音量：JS_TalkSetVolume(nVolume)

参数：

参数名	类型	说明	必需
nVolume	number	音量大小 (0-100)	否

返回：Promise

```
myPlugin.设置对讲音量：JS_TalkSetVolume(nVolume).then(
    () => {
        console.info('JS_TalkSetVolume success');
        // do you want...
    },
    (err) => {
        console.info('JS_TalkSetVolume failed');
        // do you want...
    }
);
```

获取对讲音量: JS_TalkGetVolume()

返回：Promise

```
myPlugin.JS_TalkGetVolume().then(
  () => {
    console.info('JS_TalkGetVolume success');
    // do you want...
  },
  (err) => {
    console.info('JS_TalkGetVolume failed');
    // do you want...
  }
);
```

录像、抓图功能同预览播放

回放快放: JS_Fast(windowIndex)

调节播放倍速为当前播放速度的2倍，最大为8倍。

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认快放当前选中窗口

返回：Promise

```
myPlugin.JS_Fast(curIndex).then(
  () => {
    console.info('JS_Fast success');
    // do you want...
  },
  (err) => {
    console.info('JS_Fast failed');
    // do you want...
  }
);
```

回放慢放: JS_Slow(windowIndex)

调节播放倍速为当前播放速度的1/2倍，最小为1/8倍。

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认慢放当前选中窗口

返回：Promise

```
myPlugin.JS_Slow(curIndex).then(  
  () => {  
    console.info('JS_Slow success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_Slow failed');  
    // do you want...  
  }  
);
```

回放定位: JS_Seek(windowIndex, stratTime, endTime)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	是
stratTime	DateTime	开始时间，格式类型：2021-06-29T00:00:00Z	是
endTime	DateTime	结束时间，格式类型：2021-06-29T00:00:00Z	是

返回：Promise

```
myPlugin.JS_Seek(curIndex , zStartDate, szEndDate).then(  
  () => {  
    console.info('JS_Seek success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_Seek failed');  
    // do you want...  
  }  
);
```

回放单帧进（高级模式功能）: JS_FrameForward(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认单帧播放当前选中窗口

返回：Promise

```
myPlugin.JS_FrameForward(curIndex).then(  
  () => {  
    console.info('JS_FrameForward success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_FrameForward failed');  
    // do you want...  
  }  
);
```

电子放大（高级模式功能）：JS_EnableZoom(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认开启当前选中窗口的电子放大功能

返回：Promise

```
myPlugin.JS_EnableZoom(curIndex).then(  
  () => {  
    console.info('JS_EnableZoom success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_EnableZoom failed');  
    // do you want...  
  }  
);
```

开启/关闭智能信息展示（高级模式功能）： JS_RenderALLPrivateData(iWndNum, bOpenFlag)

参数：

参数名	类型	说明	必需
iWndNum	Number	窗口下标	否，不传时默认开启当前选中窗口的智能信息
bOpenFlag	Boolean	开启/关闭	是

返回：Promise


```
myPlugin.JS_RenderALLPrivateData(iWndNum, true).then(
  () => {
    console.info('JS_RenderALLPrivateData success');
    // do you want...
  },
  (err) => {
    console.info('JS_RenderALLPrivateData failed');
    // do you want...
  }
);
```

分屏: JS_ArrangeWindow(splitNum)

参数：

参数名	类型	说明	必需
splitNum	Number	分隔数	是，范围：1~4

返回：Promise

分隔数	分屏效果
1	1x1
2	2x2
3	3x3
4	4x4

```
myPlugin.JS_ArrangeWindow(splitNum).then(
  () => {
    console.info('JS_ArrangeWindow success');
    // do you want...
  },
  (err) => {
    console.info('JS_ArrangeWindow failed');
    // do you want...
  }
);
```

整体全屏: JS_FullScreenDisplay(isFull)

参数：

参数名	类型	说明	必需
isFull	Boolean	是否全屏	是

返回：Promise

```
myPlugin.JS_FullScreenDisplay(true).then(  
  () => {  
    console.info('JS_FullScreenDisplay success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_FullScreenDisplay failed');  
    // do you want...  
  }  
);
```

单窗口全屏: JS_FullScreenSingle(windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认全屏当前选中窗口

返回：Promise

```
myPlugin.JS_FullScreenSingle(curIndex).then(  
  () => {  
    console.info('JS_FullScreenSingle success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_FullScreenSingle failed');  
    // do you want...  
  }  
);
```

设置窗口大小：JS_Resize (iWidth, iHeight)

参数：

参数名	类型	说明	必需
iWidth	Number	播放页面宽度	否，不传时默认父窗口大小
iHeight	Number	播放页面高度	否，不传时默认父窗口大小

返回：Promise

```
myPlugin.JS_Resize().then(  
  () => {  
    console.info('JS_Resize success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_Resize failed');  
    // do you want...  
  }  
);
```

获取OSD时间：JS_GetOSDTime (windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认全屏当前选中窗口

返回：Promise
成功返回从1970-1-1 00:00:00 到该日期对象的毫秒数

```
myPlugin.JS_GetOSDTime(curIndex).then(  
  (time) => {  
    console.info("osdTime:", new Date(time));  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_GetOSDTime failed');  
    // do you want...  
  }  
);
```

获取音视频信息：JS_GetVideoInfo (windowIndex)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认全屏当前选中窗口

返回：Promise
videoInfo = {
VideType: 'h264', //视频编码格式
audioType: 'without', //音频编码格式
width: 0, //视频分辨率的宽
height: 0, //视频分辨率的高

frameRate: 25,//视频帧率
bitRate: 2048,//视频码率，单位：Kb/s
systemFormt: "ps"//视频封装格式
};

```
myPlugin.JS_GetVideoInfo(curIndex).then(  
  (VideoInfo) => {  
    console.info("VideoInfo:", VideoInfo);  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_GetVideoInfo failed');  
    // do you want...  
  }  
);
```

设置取流连接超时时间：JS_SetConnectTimeOut (windowIndex, nTime)

参数：

参数名	类型	说明	必需
windowIndex	Number	窗口下标	否，不传时默认全屏当前选中窗口
nTime	Number	超时时间	否，不传时默认6秒，单位秒

返回：Promise

```
myPlugin.JS_SetConnectTimeOut(curIndex, nTime).then(  
  () => {  
    console.info('JS_SetConnectTimeOut success');  
    // do you want...  
  },  
  (err) => {  
    console.info('JS_SetConnectTimeOut failed');  
    // do you want...  
  }  
);
```

错误码及其描述

错误码	描述
0x12f900001	接口调用参数错误
0x12f900002	不在播放状态

错误码	描述
0x12f900003	仅回放支持该功能
0x12f900004	普通模式不支持该功能
0x12f900005	高级模式不支持该功能
0x12f900006	播放库报错
0x12f900007	Decoder.js加载失败
0x12f910000	websocket连接失败，请核对取流URL
0x12f910010	取流失败
0x12f910011	流中断
0x12f910014	没有音频数据
0x12f910015	未找到对应websocket
0x12f910016	websocket不在连接状态
0x12f910017	不支持智能信息展示
0x12f910020	单帧回放时不能暂停
0x12f910021	已是最大倍速
0x12f910022	已是最小倍速
0x12f920000	储存空间配额失败
0x12f920001	请求文件系统失败
0x12f920002	获取文件失败
0x12f920003	创建writer失败
0x12f920004	写数据失败
0x12f930000	内存不足
0x12f950000	采集音频失败，可能是在非https/localhost域下使用对讲导致
0x12f950001	对讲不支持这种音频编码格式

FAQ

1.Q: Uncaught Error: Site or Page Not Found : <http://localhost/Decoder.data>

A: 把 decoder.data 文件放在根目录下

2.Q: 实测性能和给的性能数据不符，使用了chrome92版本，没有加跨域隔离导致

A: http头增加跨域隔离，Cross-Origin-Embedder-Policy: require-corp Cross-Origin-Opener-Policy: same-origin
并在https环境下使用。