H5Player WEB SDK 开发向导

目录

1 概述

1.1 技术简介

H5Player 使用 WebSocket 通道，承载 RTSP/RTP 大华包协议来实现音视频信令和数据的传输，可以理解 WebSocket 为高速公路，RTSP/RTP 为高速公路上运输的货车，而货车里面的货物，就是音视频信令和数据。客户端（浏览器）首先和设备建立 Websocket 连接，再发起相关信令，设备端则会返回数据，客户端拿到数据后，通过 MSE 技术和大华解码库的 WASM 技术方式进行视频、音频解码及播放。

1.2 支持设备

大华 NVR(网络视频录像机)，4.0、5.0 web 设备，都均以 H5Player 的方式进行音视频的预览和回放（无插件的形式）。

1.3 运行环境

H5Player WEB SDK 需要运行在 web 容器中（常见的比如 Tomcat， Apache， IIS，Nginx 等）

客户端访问时，支持以下软件环境：

Windows XP，Windows7，Windows10，Windows11 下的 Chrome65+，FireFox65+，Edge79+，MAC OS Safari15+。

1.4 解码性能相关支持

解码性能相关支持

1.4.1 支持浏览器范围

浏览器支持

浏览器	Chrome	Firefox	Edge	Safari	其他
版本	65+	65+	79+	Mac OS Safari15+	O
支持	O	O	O	O	X

1.4.2 最高码流支持

最高码流支持

视频编码模式	H264	H265	MJPEG
最高支持	1200W(4000*3000)、25帧	400W(2560*1440)、25帧	200W(1920*1080)、25帧

1.4.3 延时报告

H264延时报告

视频编码模式	H264	H264	H264
码流大小	200W(1920*1080)、25帧	400W(2560*1440)、25帧	1200W(4000*3000)、25帧
最高延时	700ms	700ms	800ms

H265延时报告

视频编码模式	H265	H265	H265
码流大小	100W(1270*720)、25帧	200W(1920*1080)、25帧	400W(2560*1440)、25帧
最高延时	300ms	600ms	900ms

1.4.4 多通道支持

多通道支持

视频编码模式	H264	H264	H264
码流大小	100W(1270*720)、15帧	100W(1270*720)、25帧	200W(1920*1080)、25帧
最高支持通道	12	12	9

视频编码模式	H265	H265	H265
码流大小	D1(704*576)、25帧	100W(1270*720)、25帧	200W(1920*1080)、25帧
最高支持通道	12	6	4

1.4.5 浏览器音频支持

浏览器音频支持

音频编码	Chrome	Firefox	Edge	Safari
PCM	O	O	O	O
G.711A	O	O	O	O
G.711mu	O	O	O	O
G.726	O	O	O	O
AAC	O	O	O	O

注意：以上数据基于的环境如下：

客户端软件环境：win10，Chrome92。

客户端硬件环境：cpu i5 10210U，内存 16G。

2 安装

2.1 文件加载

使用 H5Player WEB SDK 拉流之前，由于当前使用的是 ES5 标准，必须加载相关文件，参考如下：

```
<script src="module/PlayerControl.js"></script>
```

最终，对外接口开放的类为 PlayerControl。

2.2 直接运行

如果仅需要以下功能（目录中的 API），则不需要设备登录以及 nginx 的转发服务，直接调用 PlayerControl 接口。

1. 播放和控制
2. 语音和对讲
3. 录像
4. 图像放大
5. 事件

2.3 跨域运行

如果需要云台相关操作，则需要登录设备，以及 http server 支持代理转发功能，H5Player WEB SDK 以 nginx 为例，介绍云台操作。

WEB SDK 的云台操作为 http 请求，http 请求受浏览器跨域限制，不能直接访问，因此需要进行 http 的代理，H5Player WEB SDK 可以使用 nginx 提供的代理服务，或者用户根据自己的 http server 进行代理配置，代理配置如下：

在 http 请求头中，添加 self-tragetIp:设备 ip（要进行云台操作的设备 ip）

在 web server 中，以 nginx 为例，进行如下设置：

在 nginx-1.10.2/conf/ nginx.conf 文件中，修改如下

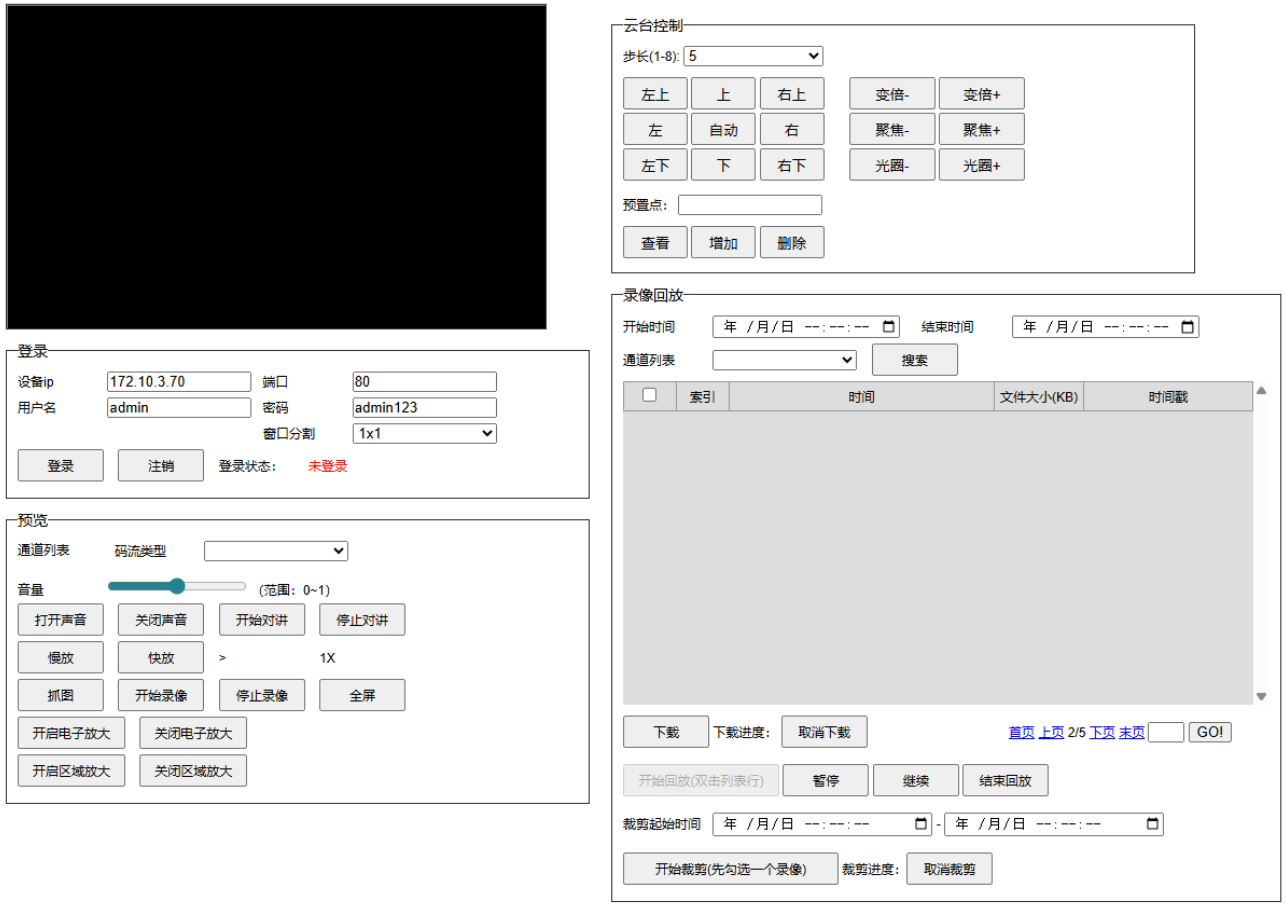
```
server {
    listen       80;
    server_name  127.0.0.1;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   "../webs";
        index  index.html index.htm;
    }

    location ~ /RPC2|RPC2_Login|RPC_Loadfile/ {
        proxy_pass http://$http_self_targetip;
        break;
    }

    location ^~ /web_caps/ {
        proxy_pass http://$http_self_targetip;
        break;
    }
```

3 启动

3.1 nginx 启动

用户可以直接点击 nginx-1.10.2 文件夹下的 start.bat，进行启动 nginx。（Linux 下的服务搭建请参考《Nginx 环境搭建_Linux.docx》）

启动后，打开浏览器，访问 http://127.0.0.1/，得到以下界面：



注意：H5Player WEB SDK 的 nginx 使用的是 80 端口，启动 nginx 服务之前，确保 80 端口是空闲的。

如果要停止，双击 stop.bat 即可。

3.2 创建播放标签

用户将 H5Player WEB SDK 集成到自己的开发环境中时，需要创建页面中的播放 html 元素，该元素用于视频的显示。

H5Player WEB SDK 在不同的编码模式中，使用 canvas 和 video 元素进行视频播放。

因为 H5Player 通过 H5 的 video 和 canvas 标签，来播放相应的视频（H264 使用 video，H265 或智能编码开启时，使用 canvas），所以我们必须先创建标签。

```
<video id="testVideo"></video>
<canvas id="testCanvas"></canvas>
```

当开始播放后，H5Player WEB SDK 会通过事件，返回一个播放模式，该模式会显示应该在 video 中播放还是 canvas 中播放，用户只需将对应的 video 或 canvas 设置为显示。在【事件】章节将会详细介绍。

3.3 创建播放实例

H5Player WEB SDK 使用以下方法创建播放实例。

```
var options = {
    wsURL: 'ws://172.23.2.100:80/rtspoverwebsocket',
    rtspURL: 'rtsp://172.23.2.100:80/cam/realmonitor?channel=1&subtype=0&proto=Private3',
    username: 'admin',
    password: 'admin123'
};
var videoplayerObj = document.getElementById('testVideo');
var canvasplayerObj = document.getElementById('testCanvas');
var player = new PlayerControl(options);
player.init(canvasplayerObj, videoplayerObj);//初始化播放器
```

options 定义如下：

属性名	描述
wsURL	websocket 协议地址，连接 websocket 服务时使用。如果设备使用的是 https 协议，那么对应的也要使用 wss 协议。
rtspURL	rtsp 协议地址，除了 channel 和 subtype 参数可以修改外，其他不能修改。 subtype 代表码流类型（0：主码流，1：辅码流 1，2：辅码流 2）。
username	用户名，因为 rtsp 拉流需要鉴权，因此需要用户名和密码。
password	密码

注意

websocket 协议地址，为服务端提供的 URL 地址，如果是 IPC、SD 等前端设备，地址一般为“IP:端口/rtspoverwebsocket”， NVR 等后端设备，wsURL 地址与前端一致， 其他平台，请咨询相应的服务提供端。

rtspURL 协议地址，和 websocket 类似，也是由服务端提供。H5Player WEB SDK 播放器只是一个客户端，它需要知道媒体的相关信息才能播放。

比如，一个浏览器客户端，当你要访问大华门户时，你应该知道大华的门户网址是 www.dahuatech.com。

至此，你可以调用 player 实例的各种 api 进行相关的业务操作。

比如：要进行实例化后的播放，使用以下方法即可。

通过调用 connect 方法，建立连接并播放。

```
player.on('WorkerReady',function(){//当文件准备完成后，开始拉流。
    player.connect();
});
```

WorkerReady 为播放相关的文件加载完成后，触发该事件。在【事件】章节将会详细介绍。

现在，你已经可以在画面中看到视频。

4 API

4.1 登录设备

RPC.login

定义和用法

RPC.login 用于登录设备，如果需要云台相关的操作，则必须登录设备。

默认值	无默认值
JavaScript 语法	RPC.login(name, pswd, false)

参数

参数	描述
name	用户名
pswd	密码
false	直接固定为 false

返回值

promise 对象

4.2 初始化

init

定义和用法

用来初始化 H5Player 播放器，该方法必须调用。

默认值	无默认值
JavaScript 语法	player.init(canvasElem, videoElem)

参数

参数	描述
canvasElem	必须，原生 dom 对象，用于 H265 播放模式下视频的播放。
videoElem	必须，原生 dom 对象，用于 H264 播放模式下视频的播放。

返回值

无

4.3 播放和控制

4.3.1 连接设备

connect

定义和用法

connect 用于和设备发起 websocket 连接，并开始播放音视频。

默认值	无默认值
JavaScript 语法	player.connect()

参数

无参数

返回值

无

4.3.2 开始播放

play

定义和用法

play 用于暂停播放后的继续播放。

默认值	无默认值
JavaScript 语法	player. play()

参数

无参数

返回值

无

4.3.3 暂停播放

pause

定义和用法

pause 该方法会暂停音视频的播放，只支持回放。

默认值	无默认值
JavaScript 语法	player. pause()

参数

无参数

返回值

无

4.3.4 恢复播放

恢复播放参考 4.3.2

4.3.5 录像跳转

playByTime

定义和用法

跳转到指定时间点播放，仅支持回放。

默认值	无默认值
JavaScript 语法	player.playByTime(range)

参数

参数	描述
range	必须，正整型，指定要跳转到录像的时间点，单位为秒。

返回值

无

4.3.6 停止播放

stop

定义和用法

停止拉流，会向设备发起 TEARDOWN 信令，但 websocket 连接未断开。

默认值	无默认值
JavaScript 语法	player. stop()

参数

无参数

返回值

无

close

定义和用法

关闭拉流，会停止拉流，并且关闭 websocket 连接，停止相关 worker 等。如果需要再次播放视频，则需要重新进行初始化操作。

默认值	无默认值
JavaScript 语法	player.close()

参数

无参数

返回值

无

4.3.7 设置音量

setAudioVolume

定义和用法

调整音量大小，范围 0-1。

默认值	无默认值
JavaScript 语法	player.setAudioVolume(val)

参数

参数	描述
val	音量大小，范围 0-1。

返回值

无

注意：如果要关闭声音，将 val 参数设置为 0 即可。WEB SDK 播放时，默认音量是 0。需要声音时，必须调用该方法，并且参数大于 0。

4.3.8 抓图

capture

定义和用法

抓取当前视频的一张图片。

默认值	无默认值
JavaScript 语法	player.capture(fileName)

参数

参数	描述
fileName	要保存的图片名称，必须符合文件名的命名格式。

返回值

无

4.3.9 事件添加

on

定义和用法

为播放器添加事件，事件类型参考【事件】章节。

默认值	无默认值
JavaScript 语法	player.on(eventName, callback)

参数

参数	描述
eventName	事件名称，为枚举值，只支持固定的事件，事件名称和描述请参考下一章节。
callback	回调函数，当触发事件时，调用该函数。如果有返回值，则返回值在回调函数的参数中表示。

返回值

无

4.3.10 减速播放

playFF

定义和用法

在播放录像时，改变改变播放的速度。

默认值	无默认值
JavaScript 语法	player. *playFF* (speed)

参数

参数	描述
speed	播放速度，可以传入的值为 1/16,1/8,1/4,1/2,1。如果传入 1 则回到正常速度

返回值

无

4.3.11 加速播放

playFF

定义和用法

在播放录像时，改变改变播放的速度。

默认值	无默认值
JavaScript 语法	player. *playFF* (speed)

参数

参数	描述
speed	播放速度，可以传入的值为 1,2,4,8,16。如果传入 1 则回到正常速度

返回值

无

4.4 语音对讲

4.4.1 开启对讲

talk

定义和用法

开启或关闭语音对讲。

默认值	无默认值
JavaScript 语法	player.talk(status)

参数

参数	描述
status	枚举值，字符串 on 为开启对讲，其他值比如 off，为关闭对讲。

返回值

无

4.4.2 关闭对讲

关闭对讲参考 4.3.1，只需要将 talk 的参数修改为 on 以外的值。

4.5 录像

4.5.1 开始录像

startRecord

定义和用法

开启或关闭本地录像。

默认值	无默认值
JavaScript 语法	player.startRecord(bool)

参数

无

返回值

无

注意：开启录像和停止录像功能为成对使用，而且必须先开启录像，才能停止录像。当停止录像时，浏览器会打开下载窗口，将录像保存到本地硬盘中。

4.5.2 停止录像

停止录像请参考 4.5.1，只需要将 startRecord 的参数修为 false。

4.5.3 录像裁剪

startCut

定义和用法

从一个录像中，根据时间裁剪某个片段。

默认值	无默认值
JavaScript 语法	player.startCut(bool)

参数

参数	描述
bool	布尔型，当参数为 true 时，开始录像裁剪，false 时，录像裁剪完成。

返回值

无

注意：录像裁剪功能，为一个比较复杂的功能，一个完整的录像裁剪功能如下：

1、初始化一个 WEB SDK。
2、录像的 ws 和 rtsp 地址。
3、设置要裁剪的时间段（开始时间和结束时间）。
4、进行裁剪动作。
5、当数据流传输到结束帧时，裁剪动作完成。

完整样例请参考以下图示：

```
const onStartCut = () => {
        let _cutIndex = _doc.querySelector('[btn-for=onStartCut]').getAttribute('cutIndex') - 0;
        let cutPlayer = null;
        let _dom = _doc.querySelector('.h5-menu-list li:nth-child(' + (curDeviceIndex + 1) + ')');
        let ip = _dom.getAttribute('data-ip');
        let port = Number(_dom.getAttribute('data-port'));
        let username = _dom.getAttribute('data-user');
        let password = _dom.getAttribute('data-pswd');
        let url = recordArr[_cutIndex].FilePath;
        let _rtspURL = 'rtsp://'+ ip +':' + port + '/' + url;
        let cutStartTime = $('#h5_cutStartTime').value;
        let s = new Date(cutStartTime.replace('T', ' ')).getTime();
        let startTime = new Date(recordArr[_cutIndex].StartTime).getTime();
        let rangel = (s - startTime)/1000;
        let optionsRecord = {
            wsURL: 'ws://'+ ip +':' + port + '/rtspoverwebsocket',
            rtspURL: _rtspURL,
            username: username,
            password: password,
            isPrivateProtocol: false, //是否私有协议，默认false
            realm: RPC.realm, //登录返回的设备Realm值
            speed: 16, //倍速拉流，16倍速
            playback: true, //是都回放
            range: rangel //视频裁剪时间与视频的StartTime时间差值
        }
        cutPlayer = new PlayerControl(optionsRecord);
        cutPlayer.on('WorkerReady',function(){
            cutPlayer.connect();
        });
        cutPlayer.on('FileOver',function(){
            console.log('File Over');
            cutPlayer.startCut(false);
            isCuting = false;
            $('#h5_cut_process').innerText = '100%';
        });
        cutPlayer.on('UpdateTimeStamp', function (e) {
            let cutStartTimel = $('#h5_cutStartTime').value;
            let cutEndTimel = $('#h5_cutEndTime').value;
            let sl = new Date(cutStartTimel.replace('T', ' ')).getTime() / 1000;
            let el = new Date(cutEndTimel.replace('T', ' ')).getTime() / 1000;
            let process = parseInt((1 - (el - e.timestamp) / (el - sl)) * 100);
            // console.log(new Date(e.timestamp * 1000));
            $('#h5_cut_process').innerText = (process > 100 ? 100 : process) + '%';
            if((e.timestamp >=  sl) && !isCuting) {
                cutPlayer.startCut(true);
                isCuting = true;
            }
            if((e.timestamp >= el ) && isCuting) {
                cutPlayer.startCut(false);
                isCuting = false;
                $('#h5_cut_process').innerText = '100%';
            }
        });
        cutPlayer.init($canvas, $video);
        cutInstance[WndIndex] = cutPlayer;
    }
```

录像裁剪必须的参数，和录像播放类似。

以最快速度裁剪。可以根据网络情况，修改为 1-16 之间的值。

和设备进行连接。

如果裁剪设置的结束日期比录像时长还大，则默认自动裁剪到录像末尾。

裁剪的时间段

当码流数据的时间为开始裁剪时间之后时，开始裁剪动作。

当码流数据的时间为结束裁剪时间之后的时间，结束裁剪动作。

4.5.4 录像下载

录像下载和录像剪切原理是一样，区别在于录像下载不传递开始时间和结束时间，参考录像剪切功能。

4.6 云台控制

4.6.1 云台转动

操作云台转动的前提是设备登录成功。

onHandlePTZ

定义和用法

调节云台，调节设备画面，可以操作云台向 8 个方向转动。在步长输入框中输入 1-8 的整数后点击相关按钮。

默认值	无默认值
JavaScript 语法	定义在 window 下，onmousedown 和 onmouseup 事件时调用，window.onHandlePTZ = function(type, isStop) {……}

参数

参数	描述
type	云台转动方向，Left (左)，LeftUp (左上)，Up (上)，RightUp (右上)，Right (右)， LeftDown（左下），Down（下），RightDown（右下）
isStop	bool 类型，是否停止转动。onmousedown 事件时设置为 false，onmouseup 事件时设置为 true。

返回值

无

RPC.PTZManager

定义和用法

RPC 协议，控制云台操作。

默认值	无默认值
JavaScript 语法	RPC.PTZManager(action, channel, { 'code': type, 'arg1': arg1, 'arg2': arg2, 'arg3': arg3 })

参数

参数	描述
action	start 或 stop
channel	通道
type	Left，LeftUp，Up，RightUp，Right，LeftDown，Down，RightDown
arg1	步长（垂直速度），1~8
arg2	步长（水平速度），1~8
arg3	默认 0

返回值

无

4.6.2 预置点操作

操作预置点的前提是设备登录成功。

onHandlePTZ

定义和用法

调节云台，调节设备画面，可以查看、增加、删除相应编号对应的预置点。预置点输入框中输入正整数后点击相关按钮。

默认值	无默认值
JavaScript 语法	定义在 window 下，onclick 事件时调用，window.onHandlePTZ = function(type, isStop) {……}

参数

参数	描述
type	GotoPreset（查看预置点），SetPreset（设置预置点），ClearPreset（删除预置点）。
isStop	bool 类型，默认为 false，无需修改。

返回值

无

RPC.PTZManager

定义和用法

RPC 协议，控制云台操作。

默认值	无默认值
JavaScript 语法	RPC.PTZManager(action, channel, { 'code': type, 'arg1': arg1, 'arg2': arg2, 'arg3': arg3 })

参数

参数	描述
action	start
channel	通道
type	GotoPreset，SetPreset， ClearPreset。
arg1	预置点值
arg2	预置点值
arg3	默认 0

返回值

无

4.6.3 变倍聚焦光圈

调节变倍聚焦光圈的前提是设备登录成功。

onHandlePTZ

定义和用法

调节设备画面，可以处理设备的变焦、变倍，调整光圈大小。在步长输入框中输入 1-8 的整数后点击相关按钮。

默认值	无默认值
JavaScript 语法	定义在 window 下，onmousedown 和 onmouseup 事件时调用， window.onHandlePTZ=function(type, isStop) {}

参数

参数	描述
type	ZoomWide（变倍-），ZoomTele（变倍+），FocusFar（聚焦-），FocusNear（聚焦+），IrisSmall（光圈-），IrisLarge（光圈+）。
isStop	bool 类型，是否停止调节。onmousedown 事件时设置为 false，onmouseup 事件时设置为 true。

返回值

无

RPC.PTZManager

定义和用法

RPC 协议，控制画面操作。

默认值	无默认值
JavaScript 语法	RPC.PTZManager(action, channel, { 'code': type, 'arg1': arg1, 'arg2': arg2, 'arg3': arg3 })

参数

参数	描述
action	start 或 stop
channel	通道
type	ZoomWide，ZoomTele，FocusFar，FocusNear，IrisSmall，IrisLarge。
arg1	步长（倍数）
arg2	默认 0
arg3	默认 0

返回值

无

4.7 图像放大

4.7.1 开启电子放大

电子放大采用的原理是通过 canvas，在界面中绘制一个矩形框，得到矩形框的大小，通过特定的比例算法，得到一个新的高度 x 和宽度值 y，将当前视频的 dom 节点的高度和宽度设置为 x 和 y，即可实现电子放大。

通过以下步骤，可以实现电子放大功能。

1、在视频播放实例的【开始解码】事件中，添加一个图形绘制实例，并且给这个绘图实例一个回调函数，当在界面中绘制一个图形时，回调函数会被执行，同时参数为图形的坐标点。

```
player.on('DecodeStart', function (e) {
    console.log(e)
    if(e.decodeMode === 'video'){
        $video.style.display = '';
        $canvas.style.display = 'none';
    }else{
        $video.style.display = 'none';
        $canvas.style.display = '';
    }
    canvasSon = new PluginCanvasES6();
    canvasSon.init($canvas_ivs, function (data) {
        rebackActivateLocalEnlarging(data);
    });
    canvasSon.addChangeShapeEvent();
    playerInstance[WndIndex] = player;
    ivsInstance[WndIndex] = canvasSon;
});
```

初始化一个绘图实例。

注册一个回调函数，当图形绘制完成时，调用该函数，并且返回图形坐标点。

2、rebackActivateLocalEnlarging 根据上一步返回的坐标点，计算电子放大的倍数，并且进行视频的放大操作。

3、onStartEnlarge 方法执行后，可以在界面上进行矩形的绘制，也就是电子放大区域的设置，当绘制完成后，会执行第 2 步骤。

4.7.2 关闭电子放大

关闭电子放大，只要将视频的大小恢复为默认值即可，参考以下方法。

通过以下步骤，可以实现关闭电子放大功能。

```
const onStopEnlarge = () => {
    if(curEnlargeWnd != WndIndex) return;
    let dom = $canvas;
    if (dom.style.display === 'none') {
        dom = $video;
    }
    dom.style.width = '100%';
    dom.style.height = '100%';
    dom.style.left = 0;
    dom.style.top = 0;
    dom.style.position = 'static';
}
```

播放 dom 节点(canvas 或者 video),设置为初始大小。

4.7.3 设置全屏

setfullScreen

定义和用法

设置画面全屏。

默认值　　无默认值

JavaScript 语法　　setfullScreen()

参数

无参数

返回值

无

4.7.4 取消全屏

exitfullScreen

定义和用法

设置画面全屏。

默认值　　无默认值

JavaScript 语法　　exitfullScreen()

参数

无参数

返回值

无

4.8 事件

4.8.1 添加事件

on

定义和用法

为播放器添加事件，事件类型参考【事件】章节。

默认值	无默认值
JavaScript 语法	player.on(eventName, callback)

参数

参数	描述
eventName	事件名称，为枚举值，只支持固定的事件，事件名称和描述请参考下一章节。
callback	回调函数，当触发事件时，调用该函数。如果有返回值，则返回值在回调函数的参数中表示。

返回值

无

5 事件

5.1 文件加载完成

WorkerReady

定义和用法

当所有文件加载完成时，触发该事件。

JavaScript 语法

```
player.on('WorkerReady', function(){
    player.connect();
});
```

H5Player WEB SDK 有部分代码是异步加载的，如果还未加载完成就和设备建立连接进行拉流，会导致播放失败。因此需要在 WorkerReady 触发之后，才能进行连接操作，否则会偶现播放失败。

5.2 开始解码

DecodeStart

定义和用法

当解码器开始解码时，触发该事件。

JavaScript 语法

```
player.on('DecodeStart', function(rs){
    console.log(rs);
});
```

该事件会传递 rs 结果给回调函数，该 rs 为一个结果集，结果集参考如下

属性名	说明
width	当前视频的宽度。
height	当前视频的高度。
decodeMode	当前视频使用 canvas 还是 video 模式进行播放。
encodeMode	当前视频编码模式。

以下为返回结果的示例：

```
decodeMode: "canvas"
encodeMode: "h265"
height: 720
width: 1280
```

5.3 开始播放

PlayStart

定义和用法

当播放器开始播放视频时，触发该事件。

JavaScript 语法

```
player.on('PlayStart', function(){
    //隐藏 loading 图标
});
```

5.4 分辨率改变

MSEResolutionChanged

定义和用法

当视频分辨率更改时，触发该事件。

JavaScript 语法

```
player.on('MSEResolutionChanged ', function(rs){
    console.log(rs);
});
```

H5Player 库解析每一帧时，都会判断当前帧和上一帧的分辨率是否一致，如果分辨率变更了，则会触发该事件。当有一个用户变更了分辨率后，所有打开 H5Player 播放视频的客户端，都会触发该事件。

该事件会传递 rs 结果给回调函数，该 rs 为一个结果集，结果集参考如下：

属性名	说明
width	当前视频的宽度。
height	当前视频的高度。
decodeMode	当前视频使用 canvas 还是 video 模式进行播放。
encodeMode	当前视频编码模式。

以下为返回结果的示例，该返回结果为从 720p 切换到 1080p 时的示例：

```
decodeMode: "canvas"
encodeMode: "h265"
height: 1080
width: 1920
```

5.5 视频编码模式改变

FrameTypeChange

定义和用法

当视频编码模式更改时，触发该事件。

JavaScript 语法

```
player.on('FrameTypeChange', function(){
    console.log(rs);
});
```

H5Player 库解析每一帧时，都会判断当前帧和上一帧的视频编码模式是否一致，如果视频编码模式变更了，则会触发该事件。当有一个用户变更了视频编码模式后，所有打开 H5Player 播放视频的客户端，都会触发该事件。
因为视频编码模式变更后，视频播放会中断而且无法恢复，必须要重新拉流，因此，一般在此事件的回调中，会关闭之前的拉流，再重新拉流。回调函数的参数为变更后的视频编码模式。

5.6 音频编码模式改变

audioChange

定义和用法

当音频编码模式更改时，触发该事件。

JavaScript 语法

```
player.on('audioChange', function(){
    console.log(rs);
});
```

H5Player 库解析每一帧时，都会判断当前帧和上一帧的音频编码模式是否一致，如果音频编码模式变更了，则会触发该事件。当有一个用户变更了音频编码模式后，所有打开 H5Player 播放音频的客户端，都会触发该事件。音频模式变更后，可能会导致音频问题，建议在该回调函数中关闭拉流，然后重新拉流播放。

5.7 获取到录像总时长

GetTotalTime

定义和用法

当回放录像鉴权成功后，触发该事件。

JavaScript 语法

```
player.on('GetTotalTime ', function(){
    console.log(rs);
});
```

播放一个录像，当鉴权成功后，获取该录像的总时长，一般用于播放进度显示。。

5.8 获取录像文件时间戳

UpdateTimeStamp

定义和用法

当下载或裁剪回放录像时，每播放新的一帧，触发该事件。

JavaScript 语法

```
player.on('UpdateTimeStamp', function(){
    console.log(rs);
});
```

其中，rs 的格式为{timestamp：1681108800}，timestamp 单位为秒。
下载和裁剪文件时，根据文件的开始时间和结束时间，利用返回的时间戳，计算下载进度，并控制下载和裁剪的开始和结束。

```
let s1 = new Date(item.StartTime).getTime()/1000;
let e1 = new Date(item.EndTime).getTime()/1000;
let process = parseInt((1 - (e1 - rs.timestamp) / (e1 - s1)) * 100);
```

5.9 播放新的一帧

UpdateCanvas

定义和用法

当播放新的一帧时，触发该事件。

JavaScript 语法

```
player.on('UpdateCanvas', function(rs){
    console.log(rs);
});
```

H5Player 库播放每一帧时，都会触发该事件，利用该事件，可以做回放的进度条。

因为该事件的回调函数中，返回了当前帧的时间戳，如果我们记录下第一帧的时间戳，之后每一帧的时间戳减去第一帧的时间戳，即可得到当前播放的时间，也就是进度条中的当前进度。

该事件会传递 rs 结果给回调函数，该 rs 为一个结果集，结果集参考如下：

属性名	说明
timestamp	当前帧的时间
timestamp_usec	暂时不用

以下为返回结果的示例，该返回结果为从 720p 切换到 1080p 时的示例：

```
timestamp: 1680508037
timestamp_usec: -400
```

注意：返回帧时间是以秒为单位，如果想得到日期-时间的表达格式，参考以下方法：

```
var timestamp = 1680508037 * 1000;//转化为毫秒
var test = new Date(timestamp);
var dateStr = test.getFullYear() + '-' + (test.getMonth() + 1) + '-' +test.getDate();
var timeStr = test.getHours() + ':' + (test.getMinutes() + 1) + ':' +test.getSeconds();
console.log(dateStr + ' ' + timeStr);
```

结果如下：

2023-4-3 15:48:17

5.10 当前录像播放完成

FileOver

定义和用法

当录像文件播放完成时，触发该事件。

JavaScript 语法

```
player.on('FileOver', function(){
    console.log(rs);
});
```

播放一个录像，当该录像播放完成时，会触发该事件，利用该事件，可以连续播放录像。比如现在有 10 个录像需要连续播放，那么只需要在该事件中调用下一个录像文件的播放即可。

5.11 播放发生错误

Error

定义和用法

当播放中发生错误时，触发该事件。

JavaScript 语法

```
player.on('Error', function(){
    console.log(rs);
});
```

播放过程中，可能会出现错误，H5Player 库已经把部分错误捕获出来，通过事件的形式，通知外部。
返回的错误中，以错误码 errorCode 的形式表示。错误码请参考【错误码及说明】章节。

6 错误码及说明

6.1 错误码及说明

错误码	描述
101	当播放的延时大于 8 秒。
201	目前有些音频编码模式无法解码，当无法解码该音频时，返回该错误码。
202	websocket 发生错误，一般是 websocket 被断开。
203	文件播放完成，一般是用在录像回放处。
404	RTSP 地址没找到，一般是 RTSP 地址错误导致。
457	在回放的情况下，错误的播放时间点。比如录像只有 2 分钟，但是发送要播放第 300 秒的数据。
503	SETUP 服务不可用。
504	对讲服务不可用。