19_低时延客户端 SDK 参考

文档版本 01

发布日期 2025-02-19

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 SDK 介绍

华为云低时延直播的软件开发工具包是对低时延直播提供的REST API进行的封装，以简化用户开发工作。用户直接调用低时延直播SDK提供的接口函数，即可实现使用低时延直播业务能力的目的。各类客户端SDK的下载和集成操作、接口参考等，请参见表1-1。

须知

目前不支持信令接入低时延直播业务，仅支持SDK接入。

表 1-1 客户端 SDK

客户端	SDK下载	版本号	开发者	主要功能	集成SDK	接口参考
Web	SDK：HWLLS_SDK_Web_2.10.9.tar.gz 完整性校验：HWLLS_SDK_Web_2.10.9.tar.gz.sha256	2.10.9	华为云	提供低时延直播业务能力	Web SDK集成	Web SDK接口参考

软件包完整性校验

用户可对下载的SDK包进行完整性校验，判断下载过程中是否存在篡改和丢包现象。

详细操作如下所示：

步骤1 在表1-1中下载SDK包及其完整性校验sha256包至本地。

步骤2 打开本地命令提示符框，输入如下命令，在本地生成已下载SDK包的SHA256值。

其中，“D:\HWLLS_SDK_Web_2.6.0.tar.gz”为SDK包本地存放路径和SDK包名，请根据实际情况修改。

```
certutil -hashfile D:\HWLLS_SDK_Web_2.6.0.tar.gz SHA256
```

命令执行结果示例，如下所示：

```
SHA256 的 D:\HWLLS_SDK_Web_2.6.0.tar.gz 哈希:
3ac83be852e8dcc9e90f236801fd4c494983073543e1ae66ee4d0c29043dccd1
CertUtil: -hashfile 命令成功完成。
```

步骤3 比对查询出的SDK包SHA256值和下载后的SDK包SHA256值。

如果一致，说明下载过程中不存在篡改和丢包现象。

----结束

2 版本规划

低时延直播（Low Latency Live）客户端SDK的版本规划说明。

版本号说明

版本号格式为a.b.c，其中：

- a为主版本号：在版本架构重构的情况下更新。例如，在版本间接口存在兼容问题时会进行变更。
- b为次版本号：正常迭代版本，如有新功能特性、接口新增或优化等，则该字段递增。
- c补丁版本号：如有功能优化或缺陷修复时，则该字段递增。

版本号样例：2.0.1

版本周期

默认1-2个月发布一个版本，或根据客户的诉求进行发布变更。

版本约束

无，新老版本兼容。

3 Web SDK

3.1 浏览器适配

本章节介绍低时延直播Web SDK支持的浏览器类型、版本以及使用限制。

表 3-1 浏览器适配详情

操作系统类型	浏览器类型	浏览器版本
Windows	Chrome浏览器	67+
	QQ浏览器（极速内核）	10.4+
	360安全浏览器（极速模式）	12
	微信内嵌浏览器	-
	Firefox浏览器	90+
	Edge浏览器	80+
	Opera浏览器	54+
macOS	Chrome浏览器	67+
	微信内嵌浏览器	-
	Safari浏览器	13+
	Firefox浏览器	90+
	Opera浏览器	56+
Android	微信内嵌浏览器（ TBS内核）	-
	微信内嵌浏览器（ XWEB内核）	-
	移动版Chrome浏览器	83+
	移动版QQ浏览器	12+

操作系统类型	浏览器类型	浏览器版本
	华为系统浏览器	11.0.8+
iOS	微信内嵌浏览器	iOS 14.3+ 微信6.5+版本
	移动版Chrome浏览器	iOS 14.3+
	移动版Safari浏览器	13+

表 3-2 浏览器使用限制

浏览器类型	使用限制
Chrome浏览器	1、在华为移动端设备上，Chrome浏览器（包括华为浏览器）支持WebRTC的版本为91+。 2、Android移动端WebView对WebRTC能力的支持参差不齐，受影响的因素很多，如设备厂家、浏览器内核、版本等，使用的兼容性较差，因此可用性不能保证，不建议使用这类浏览器。
Safari浏览器	iOS Safari 14.2和macOS Safari 14.0.1上音频可能断断续续。
Firefox浏览器	Apple M1芯片的Mac设备上Firefox不支持H.264编解码。
Opera浏览器	在华为移动端设备上，Opera浏览器支持SDK的版本为64+。

3.2 开发前准备

前提条件

已下载SDK包。

环境要求

- 编译工具推荐安装Microsoft Visual Studio Code 1.43.2或以上版本。
- 如果客户端用Node.js开发，推荐安装14.19.1或以上版本。
- 支持的浏览器详情请参见**浏览器适配详情**。
- 如果客户端用TypeScript开发，TypeScript的版本不低于3.8.3。

SDK 集成

步骤1 将**SDK下载**到本地，建议将SDK压缩包放置在自己项目的“sdk”目录下。

步骤2 在项目代码中引入“HWLLSPlayer”。

- 如果您通过<script>方式引入SDK，则通过访问HWLLSPlayer获取导出的模块：

```
<script src='./sdk/HWLLSPlayer.js'>
    console.log(HWLLSPlayer.getVersion())
</script>
```

- 如果您通过npm模块化的方式引入，首先要安装HWLLSPlayer模块，在package.json的开发依赖里引入HWLLSPlayer，如："HWLLSPlayer": "./sdk/HWLLS_SDK_Web_*.*.*.***.tar.gz"。在终端执行安装命令（版本号按实际替换）：npm install，然后通过以下方式访问：

```
import HWLLSPlayer from 'HWLLSPlayer'
console.log(HWLLSPlayer.getVersion())
```

----结束

3.3 SDK 使用

步骤1 创建容器。

```
<body>
    <div id='preview' style='width:1280px; height:720px'>
    </div>
</body>
```

步骤2 创建客户端，参考**createClient**。

```
const client = HWLLSPlayer.createClient()
```

步骤3 填入流地址和容器ID开始播放，参考**startPlay**。

```
const streamUrl = 'webrtc://domain/appname/streamname'
client.startPlay(streamUrl, {
    elementId: 'preview', // 必填，容器id，一般传入div标签的id，这里填入的是步骤1的div容器id
})
client.on('Error', (errorInfo)=>{
    // 播放中的错误在这里可以监听处理
    console.log(`Something error: ${errorInfo.getMsg()}`)
})
```

错误详情参考 **客户端错误码**。

步骤4 播放结束，停止播放，参考**stopPlay**。

```
client.stopPlay()
```

步骤5 释放资源，参考**destroyClient**。

```
client.destroyClient()
```

进阶用法，请参考**进阶用法**。

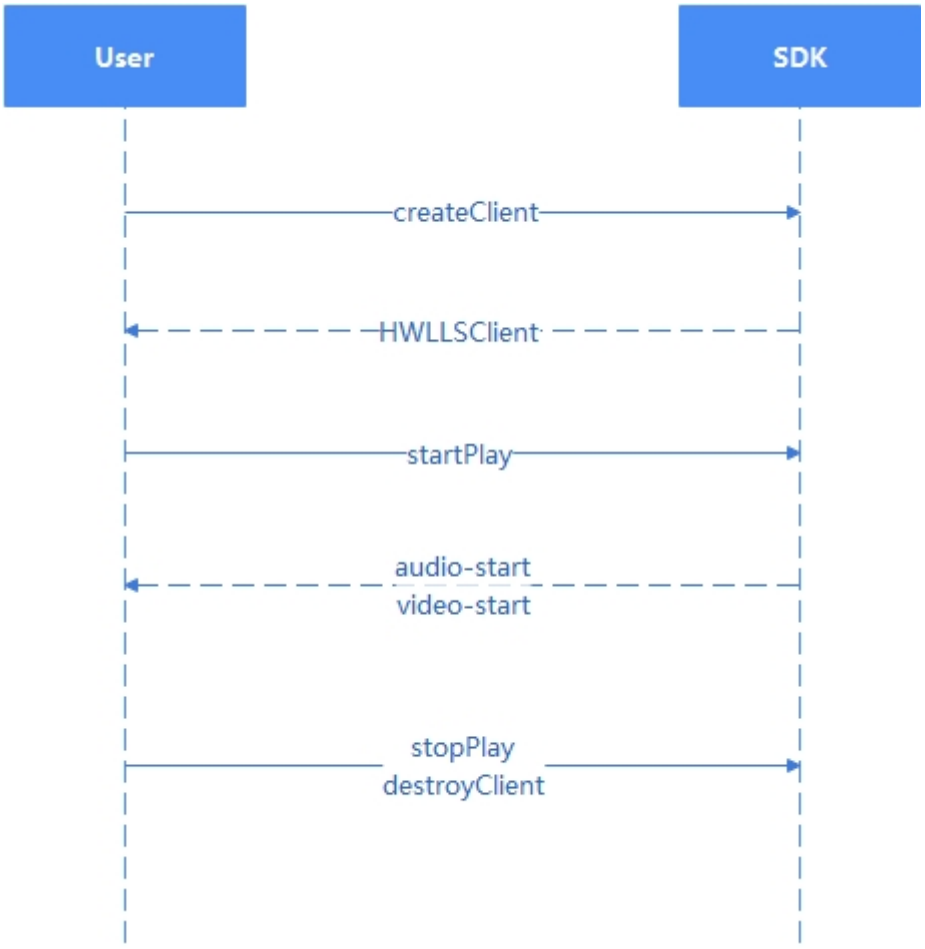
----结束

3.4 基本使用逻辑

主业务流程包括：

- 拉流前：创建客户端。
- 拉流播放：拉流播放请求。
- 停止播放：停止播放请求。
- 拉流后：销毁客户端。

单击下图中的接口名称可快速跳转至对应的接口描述，查看其使用方法。



3.5 最佳实践

3.5.1 进阶用法

总体说明

进阶用法，共包含下述场景：

- **场景一： 播放**
- **场景二：暂停与恢复播放**
- **场景三：切换视频**
- **场景四：全屏播放**
- **场景五：静音**
- **场景六：停止播放**
- **场景七：销毁播放器**

以上场景的完整代码详见**进阶用法**，可自行拷贝至本地运行测试。

场景一： 播放

使用**startPlay**方法时需传入配置，**elementId**为必传的DOM节点ID，指定视频渲染节点。推荐传入**downgradeUrl**，用于在浏览器不支持webrtc或网络较差时，能够按设定的降级url去播放，减少播放失败场景。当期望播放画面填充满播放区域时，可以传入**objectFit**为**fill**，共支持3种画面模式，详见**startPlay**

```
// 播放配置
const options = {
  // 播放DOM节点ID
  elementId: 'preview',
  // 降级地址
  downgradeUrl: {
    flvUrl: 'https://xxxx/xx/xx/xx.flv',
    hlsUrl: 'https://xxxx/xx/xx/xx.m3u8'
  },
  // 画面适配
  objectFit: 'fill'
}

// 播放
const startPlay = function () {
  playClient.startPlay(url, options)
}
```

场景二：暂停与恢复播放

播放过程如需暂停，可调用**pause**方法，详见**pause**；恢复播放需调用**resume**方法，详见 **resume**。

```
// 暂停
const pauseAction = function () {
  playClient.pause()
}

// 恢复
const resumeAction = function () {
  playClient.resume()
}
```

场景三：切换视频

播放过程如需快速切换至另一个视频，可以调用**switchPlay**方法，传入目标视频url，详见**switchPlay**。

```
// 切换视频
const switchAction = function () {
  const url = 'a new url'
  playClient.switchPlay(url)
}
```

场景四：全屏播放

支持调用**fullScreenToggle**方法，将视频设为全屏播放模式，详见**fullScreenToggle**。

```
// 全屏
const fullScreenAction = function () {
  playClient.fullScreenToggle()
}
```

场景五：静音

提供**muteAudio**方法，将视频静音或取消静音，详见**muteAudio**。


```
// 静音
const muteAction = (() => {
  let mute = false
  return function () {
    mute = !mute
    playClient.muteAudio(mute)
  }
})()
```

场景六：停止播放

当前视频如果不想播了，就调用**stopPlay**方法，与**pause**的区别在于：**pause**只是暂时停止，不久后可能会继续播放，播放流会继续拉，只是画面和声音不播放出来。而**stopPlay**是不想看这个视频了，将播放流断掉，详见**stopPlay**。

```
// 停止
const stopAction = function () {
  playClient.stopPlay()
}
```

场景七：销毁播放器

当前整体播放任务结束之后，调用**destroyClient**方法销毁播放器。一般情况下，你在某个页面创建了播放器，在离开这个页面时就需要销毁这个播放器，期间你可以在此页面进行任意多个视频的播放，详见**destroyClient**。

```
// 销毁
const destroyAction = function () {
  playClient.destroyClient()
  playClient = null
}
```

3.5.2 音频受限处理

什么是音频受限

新页面加载后立即自动播放音频（或带有音轨的视频）可能会让用户感到意外，所以主流浏览器对音频自动播放有音频受限策略。自动播放音频的场景下，因为可能与页面无交互，会触发音频受限策略。非自动播放或静音播放场景，音频则不受限。具体表现在，如果audio或video标签携带autoplay属性，页面加载完后，不会自动播放；如果强行调用play接口，会报类似“**Uncaught (in promise) DOMException: play() failed because the user didn't interact with the document first**”的错误。

HTML：

```
<video src="/video_with_audio.mp4" autoplay></video>
```

JS：

```
videoElement.play();
```

直接在新页面自动播放带有音频的视频，会受到浏览器的阻止。

通常浏览器期望与用户交互后，再允许自动播放音频。如：通过单击当前页面的播放按钮，去调用play接口。

```
PlayButton.addEventListener('click', () => {
  videoElement.play();
})
```


音频受限引发的 LLL SDK 报错信息

使用LLL SDK自动播放低延时直播流，且非静音播放时：

```
const options = {
  elementId: 'elementId',
  autoPlay: true,
}
startPlayPromise = playClient.startPlay(streamUrl, options)
```

设置autoPlay为true。如果不设置autoPlay，也会默认为true。当音频受限时，浏览器控制台报错如下所示：

```
[HWLLS] [error] [HLLSTrack] [play audio failed: ["code": 51000000, "message": "the user didn't interact with the document first, please trigger by gesture."] ]
### lll play SDK occur error: {"errCode":51000000,"errDesc":"the user didn't interact with the document first, please trigger by gesture."}
```

同时如果监听Error，会报**51000000**错误码，如下所示：

```
playClient.on('Error', (resp) => {
  if (resp.errCode === 51000000) {
    //音频受限报错
  }
})
```

最佳实践

手动播放

LLL SDK静音播放调用示例如下：

```
const options = {
  elementId: 'elementId',
  autoPlay: false
}
startPlayPromise = playClient.startPlay(streamUrl, options)
```

设置播放参数autoPlay为false，添加播放按钮让用户在点击的时候才开始播放。

图 3-1 样例效果图

自动播放失败监听

先自动播放，且不静音播放：

```
const options = {
  elementId: 'elementId',
  autoPlay: true,
 }
startPlayPromise = playClient.startPlay(streamUrl, options)
```

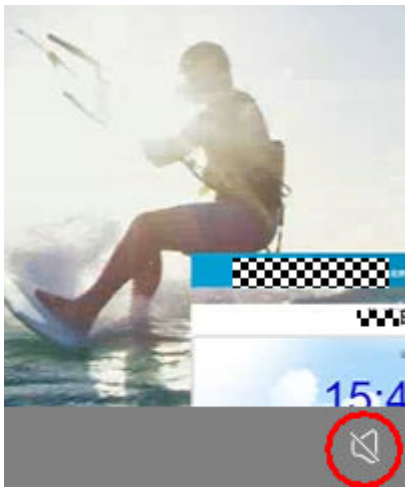
遇到音频受限时，监听错误码如下所示：

```
playClient.on('Error', (resp) => {
  if (resp.errCode === 51000000) {
    //音频受限报错
    //界面添加取消静音按钮
  }
})
```

此时即使音频受限，也会先静音播放视频。如果用户想听声音，可以单击“点击取消静音”按钮播放音频。对于不受限的场景，则会直接同步播放音频和视频。

图 3-2 样例效果图 1

图 3-3 样例效果图 2



代码示例

说明

LLL SDK需解压至“sdk/”目录。

- 自动播放失败监听的代码示例，如**自动播放失败监听**所示。
- 手动播放的代码示例，如**手动播放**所示。

3.5.3 降级处理

降级场景

降级说明

使用LLL SDK进行直播拉流时，可能会遇到播放失败等问题，可以考虑使用其他协议播放。

触发条件

- 浏览器环境不支持webrtc特性，可以通过**checkSystemRequirements**接口来判断。
- 服务端请求、建链均失败。
- 媒体起播播放超时，且解码帧数为0时，会进入降级流程。
- 未开启断流重试时，播放过程中出现断流，也会进入降级流程。

降级方式

- **自动降级**

  低时延直播SDK默认是开启自动降级的，当触发自动降级时，SDK会自动尝试从低时延直播降级到FLV或者HLS协议，例如，低时延直播地址：

  ```
  webrtc://domain/appname/streamname?arg1=v1
  ```

  会根据设备的支持情况，自动尝试降级到FLV或者HLS协议地址，url中携带的参数会拼接到降级后的url中，如：

  ```
  https://domain/appname/streamname.flv?vhost=domain&arg1=v1
  或
  https://domain/appname/streamname.m3u8?vhost=domain&arg1=v1
  ```

  如果需要关闭自动降级，可以通过接口**setParameter**设置AUTO_DOWNGRADE值来修改，示例如下：

  ```
  HWLLSPlayer.setParameter('AUTO_DOWNGRADE', false) // true表示开启自动降级播放，false表示关闭自动降级播放，默认开启
  ```

- **指定降级**

  通过**HWLLSClient**中的**startPlay**接口，指定options里面参数downgradeUrl的flv或hls地址，可以实现在异常时降级播放。如果hlsUrl和flvUrl播放地址设置一个，会降级至指定地址；如果两个播放地址都设置，则会先走HLS降级，如果HLS不支持或者HLS拉流失败，则会走FLV降级。注意iOS设备不支持FLV播放。

  ```
  const client = HWLLSPlayer.createClient()
  client.startPlay(url, {
      ...
      downgradeUrl: {
          hlsUrl: // hls的播放地址
          flvUrl: // flv的播放地址
      }
      ...
  })
  ```


降级回调

降级播放后，会有回调事件。

```
const client = HWLLSPlayer.createClient()
client.on('player-changed', (mediaFormat) => {
    // mediaFormat: hls、flv
})
```

代码示例

说明

LLL SDK需解压至“sdk/”目录。

两种降级方式的示例代码，如下所示：

- 自动降级的代码示例，如**自动降级**所示。
- 指定降级的代码示例，如**指定降级**所示。

3.5.4 代码示例

3.5.4.1 进阶用法

进阶用法的完整代码实例，如下所示：

```
<!DOCTYPE html>
<html>
  <head>
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-
scalable=no"
    />
    <title>进阶用法</title>
    <style>
      html, body {
        width: 100%;
        height: 100%;
        padding: 0;
        margin: 0;
      }
      .preview_player {
        width: 1280px;
        height: 720px;
        margin: 20px auto;
        border: 1px solid #ddd;
        position: relative;
      }
      .tools {
        width: 1280px;
        margin: 20px auto;
        display: flex;
        flex-wrap: wrap;
      }
      .play_btn {
        color: #fff;
        border: 1px solid rgb(2, 16, 201);
        background: rgba(2, 16, 201, 0.8);
        border-radius: 2px;
        cursor: pointer;
        margin-left: 10px;
      }
      .op_btn {
```



```
      margin-left: 10px;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <div id="preview" class="preview_player"></div>
  <div class="tools">
    <input id="playUrl"></input>
    <button class="play_btn" onclick="playAction()">播放</button>
    <button class="op_btn" onclick="pauseAction()">暂停</button>
    <button class="op_btn" onclick="resumeAction()">恢复</button>
    <button class="op_btn" onclick="switchAction()">切换视频</button>
    <button class="op_btn" onclick="stopAction()">停止</button>
    <button class="op_btn" onclick="fullScreenAction()">全屏</button>
    <button class="op_btn" onclick="muteAction()">静音/取消静音</button>
    <button class="op_btn" onclick="destroyAction()">销毁</button>
  </div>
  <script type="text/javascript" src="./sdk/HWLLSPlayer.js"></script>
  <script type="text/javascript">
    const playUrlInput = document.getElementById('playUrl')
    playUrlInput.value = 'your webrtc url'

    let playClient = HWLLSPlayer.createClient()
    let isPlaying = false

    const playAction = function () {
      startPlay()
    }

    // 播放配置
    const options = {
      // 播放dom ID
      elementId: 'preview',

      // 降级地址
      downgradeUrl: {
        flvUrl: 'your flv url',
        hlsUrl: 'your hls url'
      },

      // 画面适配
      objectFit: 'fill'
    }

    // 播放
    const startPlay = function () {
      if (!isPlaying) {
        isPlaying = true
        const url = playUrlInput.value
        playClient.startPlay(url, options)
      }
    }

    // 暂停
    const pauseAction = function () {
      playClient.pause()
    }

    // 恢复
    const resumeAction = function () {
      playClient.resume()
    }

    // 切换视频
    const switchAction = function () {
      const url = playUrlInput.value
      playClient.switchPlay(url)
    }
```



```
    // 停止
    const stopAction = function () {
      playClient.stopPlay()
      isPlaying = false
    }

    // 全屏
    const fullScreenAction = function () {
      playClient.fullScreenToggle()
    }

    // 静音
    const muteAction = (() => {
      let mute = false
      return function () {
        mute = !mute
        playClient.muteAudio(mute)
      }
    })()

    // 销毁
    const destroyAction = function () {
      playClient.stopPlay()
      playClient.destroyClient()
      playClient = null
      isPlaying = false
    }
   </script>
  </body>
</html>
```

3.5.4.2 自动播放失败监听

自动播放失败监听的代码示例，如下所示：

```
<!DOCTYPE html>
<html>
  <head>
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-
scalable=no"
    />
    <title>音频受限demo</title>
    <style>
      html, body {
        width: 100%;
        height: 100%;
        padding: 0;
        margin: 0;
      }
      .preview_player {
        width: 1280px;
        height: 720px;
        margin: 20px auto;
        border: 1px solid #ddd;
        position: relative;
      }
      .preview_player_el {
        width: 100%;
        height: 100%;
      }
      .unmute_mask {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
```

```
      background: rgba(0, 0, 0, 0.5);
      text-align: center;
      display: none;
    }
    .unmute_btn {
      display: inline-block;
      padding: 10px;
      color: rgb(50, 250, 50);
      border: 1px solid rgb(50, 250, 50);
      background: rgba(50, 250, 50, 0.3);
      transform: translateY(400px);
      cursor: pointer;
    }
  </style>
  <script type="text/javascript" src="./sdk/HWLLSPlayer.js"></script>
 </head>
 <body>
  <div class="preview_player">
    <div id="preview" class="preview_player_el"></div>
    <div class="unmute_mask" id="unmute-mask">
      <div class="unmute_btn" onclick="replayAction()">点击取消静音</div>
    </div>
  </div>
  <script type="text/javascript">
    const playClient = HWLLSPlayer.createClient()
    const unmuteMask = document.getElementById('unmute-mask')
    const replayAction = function () {
      playClient.replay()
      unmuteMask.style.display = 'none'
    }
    // 注册音频受限处理事件
    const initEvent = function () {
      playClient.on('Error', (resp) => {
        //音频受限报错
        if (resp.errCode === 51000000) {
          //界面添加取消静音按钮
          unmuteMask.style.display = 'block'
        }
      })
    }
    initEvent()
    // 播放视频
    const streamUrl = 'your stream url'
    const options = {
      elementId: 'preview',
      autoPlay: true,
    }
    playClient.startPlay(streamUrl, options)
  </script>
 </body>
</html>
```

3.5.4.3 手动播放

手动播放的代码示例，如下所示：

```
<!DOCTYPE html>
<html>
 <head>
  <meta
    name="viewport"
    content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-
scalable=no"
  />
  <title>手动播放demo</title>
  <style>
    html, body {
      width: 100%;
      height: 100%;
```

```
      padding: 0;
      margin: 0;
    }
    .preview_player {
      width: 1280px;
      height: 720px;
      margin: 20px auto;
      border: 1px solid #ddd;
      position: relative;
    }
    .preview_player_el {
      width: 100%;
      height: 100%;
    }
    .unmute_mask {
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      background: rgba(0, 0, 0, 0.5);
      text-align: center;
    }
    .unmute_btn {
      display: inline-block;
      padding: 10px;
      color: rgb(50, 250, 50);
      border: 1px solid rgb(50, 250, 50);
      background: rgba(50, 250, 50, 0.3);
      transform: translateY(400px);
      cursor: pointer;
    }
  </style>
  <script type="text/javascript" src="./sdk/HWLLSPlayer.js"></script>
 </head>
 <body>
  <div class="preview_player">
    <div id="preview" class="preview_player_el"></div>
    <div class="unmute_mask" id="unmute-mask">
      <div class="unmute_btn" onclick="playAction()">点击开始播放</div>
    </div>
  </div>
  <script type="text/javascript">
    const playClient = HWLLSPlayer.createClient()
    const unmuteMask = document.getElementById('unmute-mask')
    const playAction = function () {
      playClient.resume()
      unmuteMask.style.display = 'none'
    }
    // 播放视频
    const streamUrl = 'your stream url'
    const options = {
      elementId: 'preview',
      autoPlay: false,
    }
    playClient.startPlay(streamUrl, options)
  </script>
 </body>
</html>
```

3.5.4.4 自动降级

自动降级的代码示例，如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Downgrade Demo</title>
```



```
    <script src="sdk/HWLLSPlayer.js"></script>
</head>
<body>
<div id="preview" style="width: 1280px; height: 720px; position:relative;"></div>
<button id="btnStartPlay">start play</button>
<button id="btnStopPlay">stop play</button>
<script>
    HWLLSPlayer.setParameter('AUTO_DOWNGRADE', true)

    const btnStartPlay = document.getElementById('btnStartPlay')
    const btnStopPlay = document.getElementById('btnStopPlay')

    const streamUrl = 'your stream url'

    const client = HWLLSPlayer.createClient()
    bindEvent(client)

    btnStartPlay.addEventListener('click', () => {
        startPlay(client, streamUrl, 'preview')
    })
    btnStopPlay.addEventListener('click', () => {
        stopPlay(client)
    })

    function bindEvent(client) {
        client.on('Error', (error) => {
            console.log(`error: ${JSON.stringify(error)}`)
        })
        client.on('player-changed', (mediaFormat) => {
            console.log(`player changed:${mediaFormat}`)
        })
    }

    function startPlay(client, url, elementId) {
        client.startPlay(url, {
            elementId: elementId,
        })
    }

    function stopPlay(client) {
        client.stopPlay()
    }
</script>
</body>
</html>
```

3.5.4.5 指定降级

指定降级的代码示例，如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Downgrade Demo</title>
    <script src="sdk/HWLLSPlayer.js"></script>
</head>
<body>
<div id="preview" style="width: 1280px; height: 720px; position:relative;"></div>
<button id="btnStartPlay">start play</button>
<button id="btnStopPlay">stop play</button>
<script>
    const btnStartPlay = document.getElementById('btnStartPlay')
    const btnStopPlay = document.getElementById('btnStopPlay')

    const streamUrl = 'your stream url'
    const flvUrl = 'your flv url'

    const client = HWLLSPlayer.createClient()
```



```
    bindEvent(client)

    btnStartPlay.addEventListener('click', () => {
        startPlay(client, streamUrl, 'preview', {
            flvUrl: flvUrl
        })
    })
    btnStopPlay.addEventListener('click', () => {
        stopPlay(client)
    })

    function bindEvent(client) {
        client.on('Error', (error) => {
            console.log(`error: ${JSON.stringify(error)}`)
        })
        client.on('player-changed', (mediaFormat) => {
            console.log(`player changed:${mediaFormat}`)
        })
    }

    function startPlay(client, url, elementId, downgradeUrl) {
        client.startPlay(url, {
            elementId: elementId,
            downgradeUrl: downgradeUrl,
        })
    }

    function stopPlay(client) {
        client.stopPlay()
    }
</script>
</body>
</html>
```

3.6 接口参考

3.6.1 主入口（HWLLSPlayer）

本章节介绍了低时延直播Web SDK的HWLLSPlayer接口详情。

表 3-3 主入口接口

接口	描述
checkSystemRequirements	检测浏览器是否支持低时延直播Web SDK。
getVersion	获取SDK版本号。
createClient	创建一个直播拉流客户端对象，如果需要拉取多个直播流则需要创建多个客户端对象。
uploadLog	上传日志。
saveLog	保存日志
setParameter	设置全局配置参数。
setLogLevel	设置Console上打印的日志级别。
setReportConfig	设置打点能力以及打点和日志上传的认证策略。

接口	描述
on	注册客户端对象事件回调接口。
off	取消注册客户端对象事件回调接口。

checkSystemRequirements

```
checkSystemRequirements(): Promise<boolean>
```

【功能说明】

检测浏览器是否支持低时延直播Web SDK。

【请求参数】

无

【返回参数】

Promise<boolean>：返回一个Promise对象，true表示浏览器兼容低时延直播Web SDK。如果不兼容，则返回对应Error异常。

⚠ 注意

由于低延时直播需使用WebRTC能力，存在部分浏览器不支持播放WebRTC的情况，可根据特定错误码（HWLLS_ERROR_WEBRTC_UNSUPPORTED），进行播放降级，请参考SDK使用。

getVersion

```
getVersion(): string
```

【功能说明】

获取当前SDK版本号。

【请求参数】

无

【返回参数】

string：SDK当前版本号。

createClient

```
createClient(): HWLLSClient
```

【功能说明】

创建一个直播拉流客户端对象，如果需要拉取多个直播流则需要创建多个客户端对象。

【请求参数】

无

【返回参数】

client：拉流客户端对象。

createClient(2.10.6 之前版本)

```
createClient(type: string): HWLLSClient | HWFlvClient | HWHlsClient
```

【功能说明】

创建一个直播拉流客户端对象，如果需要拉取多个直播流则需要创建多个客户端对象。

【请求参数】

type：string类型，可选。创建的拉流客户端类型。

- 低时延直播拉流客户端类型：webrtc。
- flv直播拉流客户端类型：flv。
- HLS直播拉流客户端类型：hls（预留，暂未开放）。

缺省值：webrtc。

【返回参数】

client：拉流客户端对象。

⚠ 注意

2.10.6版本及之后的SDK不支持独立使用HWFlvClient和HWHlsClient这两种播放器，如您正在使用旧版本SDK，需参考详细接口请查看附录：**客户端对象（HWFlvClient）** **客户端对象（HWHlsClient）**

uploadLog

```
async uploadLog(): Promise<void>
```

【功能说明】

上传日志。

【请求参数】

无

【返回参数】

Promise<void>：使用tryCatch获取error为数组，返回多个appid对应错误信息。

saveLog

```
async saveLog(): Promise<Blob>
```

【功能说明】

为用户提供灵活保存日志的功能

【请求参数】

无

【返回参数】

Promise<Blob>：经过zip压缩好的Promise<Blob>,用户可以直接保存为zip文件

setParameter

```
setParameter(parameterKey: string, parameterValue: any): boolean
```

【功能说明】

设置全局配置参数。

【请求参数】

参数名称	参数值

LOADING_CONFIG	LoadingConfig类型，定义如下：
	{
	netQualityLoading：可选，boolean类型。true表示开启根据网络质量进行loading效果展示，默认值为false，关闭。
	netQualityLoadingThreshold：可选，number类型。展示loading效果的网络质量（**network-quality**）的阈值，默认网络质量等级为5。
	frameStuckLoading：可选，boolean类型。true表示开启根据帧卡顿时长进行loading效果展示，默认值为false
	frameStuckThreshold：可选，number类型。展示loading效果帧卡顿时长的阈值，单位为100ms。默认值为10，表示帧卡顿时长为1000ms。
	}
	注意
	需要在**起播**之前进行设置。

DNS_QUERY_ENABLE	boolean类型，可选，默认为false，true表示开启DNS结果解析，false表示关闭DNS结果解析。
ACCESS_DOMAIN	string类型，可选，默认为空，主要用于拉流环境配置，联系技术支持填入。
GLSB_DOMAIN	string类型，可选，默认为空，主要用于GSLB环境配置，联系技术支持填入。
BACKGROUND_PLAY	boolean类型，可选，默认为false，true表示开启后台播放，false表示关闭后台播放
AUTO_DOWNGRADE	boolean类型，可选，默认为true，true表示开启自动降级，false表示关闭自动降级

【返回参数】

boolean：配置参数设置结果。true表示参数设置成功，false表示参数设置失败。

setLogLevel

```
setLogLevel(level: string): boolean
```

【 功能说明 】

设置Console上打印的日志级别，如不设置日志级别，则console日志打印级别默认为error。

【 请求参数 】

level：string类型，必选，日志级别标识。

- none：关闭全部级别的日志打印。
- error：打印error级别日志。
- warn：打印warn级别及更高级别日志。
- info：打印info级别及更高级别日志。
- debug：打印debug级别及更高级别日志。

【 返回参数 】

boolean：设置日志级别结果。true表示日志级别设置成功，false表示日志级别设置失败。

setReportConfig

```
setReportConfig(reportConfig:ReportConfig):boolean
```

【 功能说明 】

设置打点能力以及打点和日志上传的认证策略。

【 请求参数 】

reportConfig：ReportConfig类型，必选。ReportConfig定义如下所示：

- enable：必选，布尔类型，true表示开启打点，false表示关闭打点。默认true。
- tokenConfig：可选，对象定义如下所示：
  - enable：布尔类型，true表示开启认证，false表示关闭认证。默认false。
  - tokenInfo：数组类型，数组内部ReportTokenInfo类。ReportTokenInfo定义如下所示：
    - appid：string类型。传入appid。
    - expTimestamp：string类型。过期时间戳，系统当前UNIX时间戳加上鉴权过期时间（推荐7200秒，最长需要小于43200秒，即12个小时）。

      例如：当前UNIX时间戳为：1708531200，鉴权过期时间自定义为7200秒，那么过期的时间戳为：1708538400，即表示该**校验字符串**在2024-02-22 02:00:00过期。
    - token：string类型。**hmac_sha256**生成的字符串。**hmac_sha256(共享密钥, 过期的时间戳 + appID)**。共享密钥由用户控制获取。

【 返回参数 】

返回值布尔值， true表示设置成功，false表示设置失败。

开启了认证策略，实际请求状态会通过on函数注册Error回调获得。

on

```
on(event: string, handler: function, withTimeout?: boolean): void
```

【功能说明】

注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称， 注册Error事件，监听打点或者日志上传的错误信息。
- handler：必选，function类型，事件处理方法。参数errorInfo。
  errorInfo定义为：{
  code：必选，number类型，错误码。
  message：必选，string类型，错误描述。
  appid：必选，string类型，错误日志标识。
  }
- withTimeout：选填，boolean类型，是否超时报错

【返回参数】

无

off

```
off(event: string, handler: function): void
```

【功能说明】

取消注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。取消注册Error事件
- handler：必选，function类型，事件处理方法。

【返回参数】

无

3.6.2 客户端对象（HWLLSClient）

本章节介绍了低时延直播Web SDK的HWLLSClient接口详情。

表 3-4 主入口接口

接口	描述
startPlay	开始播放，客户端根据输入的URL到服务端拉取对应的主播流。
switchPlay	快速切换下一路流播放。
stopPlay	停止播放。
replay	重新播放。

接口	描述
startPlayCustomize	自定义播放
resume	恢复播放。
pause	暂停播放。
pauseVideo	暂停视频。
resumeVideo	恢复视频。
pauseAudio	暂停音频。
resumeAudio	恢复音频。
setPlayoutVolume	设置播放音量。
getPlayoutVolume	获取音频音量。
muteAudio	静音
streamStatistic	设置是否开启流信息统计。
enableStreamStateDetection	开启/关闭音视频码流状态探测功能。
on	注册客户端对象事件回调接口。
off	取消注册客户端对象事件回调接口。
destroyClient	销毁客户端对象。
fullScreenToggle	开启关闭全屏

startPlay

```
startPlay(url: string, options: StartPlayOptions): Promise<void>
```

【功能说明】

开始播放，客户端根据输入的URL到服务端拉取对应的主播流。

【请求参数】

- url：string类型，必选。拉流URL，格式如：webrtc://{domain}/{AppName}/{StreamName}。
  - webrtc://：固定不变，表示使用webrtc方式拉流。
  - domain：拉流域名，使用在华为云注册的拉流域名。
  - AppName：应用名，使用在华为云注册的应用名。
  - StreamName：流名，和推流的流名保持一致。
- options：必选，StartPlayOptions类型。播放配置参数，StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：

- contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
- cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
- fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。

⚠ 注意

手机端浏览器可能会创建控件覆盖SDK的播放器，导致配置无法生效，如OPPO浏览器降级到hls/flv播放时。

- muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
- sessionId：可选，string类型，一次完整会话的统一标识。
- showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。

⚠ 注意

- Android端QQ浏览器不支持该功能。
- 建议showLoading不设置，或者设置为false。

- autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
- poster：可选，对象定义如下：{
  - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
  - mode：可选，string类型。默认值为crop。支持的枚举值如下：{
    - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
    - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

    }
  - startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。

- pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

}

- webrtcConfig：可选，WebRTCConfig类型。指定媒体类型进行拉流的配置参数，WebRTCConfig定义如下：{
  - receiveVideo：可选，boolean类型。设置是否拉取视频进行播放，true表示拉取视频进行播放，false表示不拉取视频播放，默认值true。该属性值和receiveAudio不能同时设置为false。
  - receiveAudio：可选，boolean类型。设置是否拉取音频进行播放，true表示拉取音频进行播放，false表示不拉取音频播放，默认值true。该属性值和receiveVideo不能同时设置为false。

}

- domainPolicy：可选，DomainPolicy类型。指定接入域名的策略，该设置仅在schedulePolicy参数为"DNS"模式时生效，DomainPolicy定义如下：{
  - 0：可选，number类型。表示使用用户自定义域名，默认值0。
  - 1：可选，number类型。表示使用公共接入域名。

}

- autoDowngrade：可选，boolean类型。true表示启用自动降级，false表示不启用自动降级，默认false。
- downgradeUrl：可选，对象定义如下 ：{
  - hlsUrl?：可选, string类型。标识降级hls播放地址。
  - flvUrl?:可选，string类型。标识降级flv播放地址。

}

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

在低延时直播服务发生故障的场景下，可根据特定错误码（HWLLS_BUSSINESS_DOWNGRADE），进行播放降级，请参考**SDK使用**。

switchPlay

```
switchPlay(url: string, options: StartPlayOptions): Promise<void>
```

【功能说明】

起播成功后，快速切换下一路流播放。

【请求参数】

- url：必选，string类型。拉流URL，格式如：webrtc://{domain}/{AppName}/{StreamName}。

  - webrtc://：固定不变，表示使用webrtc方式拉流。
  - domain：拉流域名，使用在华为云注册的拉流域名。
  - AppName：应用名，使用在华为云注册的应用名。
  - StreamName：流名，和推流的流名保持一致。
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
  - sessionId：可选，string类型，一次完整会话的统一标识。
  - showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。
  - autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
  - poster：可选，对象定义如下：{
    - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
    - mode：可选，string类型。默认值为cover。支持的枚举值如下：{
      - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
      - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

      }
    - startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
    - pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

    }

- webrtcConfig：可选，WebRTCConfig类型。指定媒体类型进行拉流的配置参数，WebRTCConfig定义如下：{
  - receiveVideo：可选，boolean类型。设置是否拉取视频进行播放，true表示拉取视频进行播放，false表示不拉取视频播放，默认值true。该属性值和receiveAudio不能同时设置为false。
  - receiveAudio：可选，boolean类型。设置是否拉取音频进行播放，true表示拉取音频进行播放，false表示不拉取音频播放，默认值true。该属性值和receiveVideo不能同时设置为false。

  }
- schedulePolicy：可选，SchedulePolicy类型。指定接入调度策略，SchedulePolicy定义如下：{
  - DNS：可选，string类型。表示域名DNS解析接入，默认值"DNS"。
  - HTTPDNS：可选，string类型。表示使用HTTPDNS接入域名。

  }
- domainPolicy：可选，DomainPolicy类型。指定接入域名的策略，该设置仅在schedulePolicy参数为"DNS"模式时生效，DomainPolicy定义如下：{
  - 0：可选，number类型。表示使用用户自定义域名，默认值0。
  - 1：可选，number类型。表示使用公共接入域名。

  }

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

在低延时直播服务发生故障的场景下，可根据特定错误码（HWLLS_BUSSINESS_DOWNGRADE），进行播放降级，请参考**SDK使用**。

stopPlay

```
stopPlay(): boolean
```

【功能说明】

停止播放。

【请求参数】

无

【返回参数】

boolean：停止播放结果。true表示成功，false表示失败。

replay

```
replay(): Promise<boolean>
```

【功能说明】

重新播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：重新播放结果，true表示成功，false表示失败。

startPlayCustomize

```
startPlayCustomize(url: string, options: StartPlayOptions): Promise<boolean>
```

【 功能说明 】

自定义播放。

【 请求参数 】

- url：必选，string类型。拉流URL，自定义URL播放
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
  - sessionId：可选，string类型，一次完整会话的统一标识。
  - showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。
  - autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
  - poster：可选，对象定义如下：{
    - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
    - mode：可选，string类型。默认值为cover。支持的枚举值如下：{
      - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。

- crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

}

- startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
- pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

}

- webrtcConfig：可选，WebRTCConfig类型。指定媒体类型进行拉流的配置参数，WebRTCConfig定义如下：{
  - receiveVideo：可选，boolean类型。设置是否拉取视频进行播放，true表示拉取视频进行播放，false表示不拉取视频播放，默认值true。该属性值和receiveAudio不能同时设置为false。
  - receiveAudio：可选，boolean类型。设置是否拉取音频进行播放，true表示拉取音频进行播放，false表示不拉取音频播放，默认值true。该属性值和receiveVideo不能同时设置为false。

}

- schedulePolicy：可选，SchedulePolicy类型。指定接入调度策略，SchedulePolicy定义如下：{
  - DNS：可选，string类型。表示域名DNS解析接入，默认值"DNS"。
  - HTTPDNS：可选，string类型。表示使用HTTPDNS接入域名。

}

- domainPolicy：可选，DomainPolicy类型。指定接入域名的策略，该设置仅在schedulePolicy参数为"DNS"模式时生效，DomainPolicy定义如下：{
  - 0：可选，number类型。表示使用用户自定义域名，默认值0。
  - 1：可选，number类型。表示使用公共接入域名。

}

【返回参数】

Promise<void>：返回一个Promise对象。

resume

```
resume(): Promise<boolean>
```

【功能说明】

恢复播放。

【请求参数】

无

【返回参数】

Promise<boolean>：恢复音视频播放结果，true表示成功，false表示失败。

pause

```
pause(): boolean
```

【 功能说明 】

暂停音视频播放。

【 请求参数 】

无

【 返回参数 】

boolean：暂停播放结果，true表示成功，false表示失败。

pauseVideo

```
pauseVideo(): boolean
```

【 功能说明 】

暂停视频，暂停后卡在当前画面。

【 请求参数 】

无

【 返回参数 】

boolean：暂停视频播放结果，true表示成功，false表示失败。

resumeVideo

```
resumeVideo(): Promise<boolean>
```

【 功能说明 】

恢复视频播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：恢复视频播放结果，true表示成功，false表示失败。

pauseAudio

```
pauseAudio(): boolean
```

【 功能说明 】

暂停音频。

【 请求参数 】

无

【 返回参数 】

boolean：暂停音频播放结果，true表示成功，false表示失败。

resumeAudio

```
resumeAudio(): Promise<boolean>
```

【功能说明】

恢复音频。

【请求参数】

无

【返回参数】

Promise<boolean>：恢复音频播放结果，true表示成功，false表示失败。

setPlayoutVolume

```
setPlayoutVolume(volume: number): boolean
```

【功能说明】

设置音频音量，iOS 不支持。

【请求参数】

volume：必选，number类型，取值范围为[0,100]，音频的音量值。

【返回参数】

boolean：设置音频音量是否成功，true表示成功，false表示失败。

getPlayoutVolume

```
getPlayoutVolume(): number
```

【功能说明】

获取音频音量。

【请求参数】

无

【返回参数】

number：音量值，取值范围为[0,100]。

muteAudio

```
muteAudio(isMute: boolean): void
```

【功能说明】

静音。

【请求参数】

- isMute：必选，boolean类型，是否静音，true表示静音，false表示取消静音。

【返回参数】

无

streamStatistic

```
streamStatistic(enable: boolean, interval: number): void
```

【功能说明】

设置是否开启流信息统计。

【请求参数】

- enable：必选，boolean类型，是否开启流信息统计，true表示开启统计。
- interval：必选，number类型，设置统计间隔，单位为秒，取值范围为[1, 60]，默认值为1。

【返回参数】

无

enableStreamStateDetection

```
enableStreamStateDetection(enable: boolean, interval: number, interruptRetry:StreamInterruptRetry): boolean
```

【功能说明】

开启/关闭音、视频码流状态探测功能，开启后可探测推流侧是否处于断流的状态。

【请求参数】

- enable：必选，boolean类型，true表示开启音视频码流状态探测，false表示关闭音视频码流状态探测。默认值为false。
- interval：必选，number类型，单位为秒，取值范围为[1,60]。音视频无码流状态的判断时间。默认值为3，推荐设置为3。
- interruptRetry：可选，StreamInterruptRetry类型。断流重试播放配置参数，StreamInterruptRetry定义为：{

  enable：boolean类型，开启断流后尝试自动恢复播放。默认值为false，即不开启自动重试。

  retryInterval：number类型，拉流播放的重试周期，单位为秒。最小值10，最大值建议不超过60，默认值为30。

  retryTimes：number类型，尝试重新恢复播放的最大重试次数。最小值1，默认值为30。

  }

【返回参数】

boolean：是否成功，true表示成功，false表示失败。

⚠ 注意

Android端QQ浏览器不支持该功能。

on

```
on(event: string, handler: function, withTimeout?: boolean): void
```

【功能说明】

注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。具体请参见**HWLLSClientEvent**。
- handler：必选，function类型，事件处理方法。
- withTimeout：选填，boolean类型，是否超时报错

【返回参数】

无

off

```
off(event: string, handler: function): void
```

【功能说明】

取消注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。具体请参见**HWLLSClientEvent**。
- handler：必选，function类型，事件处理方法。

【返回参数】

无

destroyClient

```
destroyClient(): void
```

【功能说明】

销毁客户端对象。

【请求参数】

无

【返回参数】

无

fullScreenToggle

```
fullScreenToggle(isExit: boolean): void
```

【功能说明】

开启关闭全屏。

【请求参数】

isExit：必选，布尔类型，默认false。

【返回参数】

无

3.6.3 客户端事件通知（HWLLSClientEvent）

本章节介绍了低时延直播Web SDK的HWLLSClientEvent接口详情。

表 3-5 HWLLSClientEvent 接口

接口	描述
media-statistic	媒体统计事件。
network-quality	网络质量报告事件。
video-broken	视频断流事件，等待恢复中。
audio-broken	音频断流事件，等待恢复中。
video-recovery	视频断流恢复事件，已恢复播放。
audio-recovery	音频断流恢复事件，已恢复播放。
audio-start	音频起播事件。
video-start	视频起播事件。
video-stuck	视频是否停顿事件
fullscreen-status-changed	播放视图是否全屏事件
player-changed	播放降级事件
Error	客户端出现错误事件。

⚠ 注意

事件注册监听应在业务结束时取消注册，否则注册监听事件累积会有内存泄漏风险。

media-statistic

【事件说明】

媒体统计事件。此事件配合streamStatistic方法使用。

【回调参数】

StatisticInfo：StatisticInfo类型，媒体统计信息。

StatisticInfo定义为：{

- video：{
  mediaType：MediaType媒体类型。
  frameRate：number类型，视频帧率。
  width：number类型，视频宽度。

height：number类型，视频高度。
jitter：number类型，抖动值。
bitRate：number类型，码率，单位：kbps。
bytesReceived：number类型，已接收字节数。
packetsReceived：number类型，已接收包数。
packetsLost：number类型，丢包数。
}

- audio：{
  mediaType：MediaType媒体类型。
  jitter：number类型，抖动值。
  bitRate：number类型，码率，单位：kbps。
  bytesReceived：number类型，已接收字节数。
  packetsReceived：number类型，已接收包数。
  packetsLost：number类型，丢包数。
  }

}

network-quality

【事件说明】

网络质量报告事件。

【回调参数】

NetworkQualityTypes：NetworkQualityTypes类型，网络质量详情。

NetworkQualityTypes枚举值参考如下：

- NETWORK_QUALITY_UNKNOW = 0，网络质量未知。
- NETWORK_QUALITY_GREAT = 1，网络质量极好。
- NETWORK_QUALITY_GOOD = 2，用户主观感觉和极好差不多，但码率可能略低于极好。
- NETWORK_QUALITY_DEFECTS = 3，网络质量一般，用户主观感受有瑕疵但不影响观看。
- NETWORK_QUALITY_WEAK = 4，网络质量差，勉强能观看但不流畅。
- NETWORK_QUALITY_BAD = 5，网络质量很差，严重影响用户观看体验。
- NETWORK_QUALITY_DISCONNECT = 6，网络质量非常差甚至链接断开，无法观看。

video-broken

【事件说明】

视频断流事件，等待恢复中。

【回调参数】

无

audio-broken

【事件说明】

音频断流事件，等待恢复中。

【回调参数】

无

video-recovery

【事件说明】

视频断流（非EOF）恢复事件，已恢复播放。

【回调参数】

无

audio-recovery

【事件说明】

音频断流（非EOF）恢复事件，已恢复播放。

【回调参数】

无

audio-start

【事件说明】

音频起播事件。

【回调参数】

无

video-start

【事件说明】

视频起播事件。

【回调参数】

无

video-stuck

【事件说明】

视频停顿事件。

【回调参数】

布尔值，True标识停顿，False标识非停顿。

fullscreen-status-changed

【事件说明】

播放视图是否全屏事件

【回调参数】

- isFullScreen：是否全屏展示。
- isPause：是否停止播放。

player-changed

【事件说明】

播放降级事件。

【回调参数】

string类型，降级信息。

- webrtc：标识LLL播放模式。
- hls：标识hls播放模式。
- flv：标识flv播放模式。

Error

【事件说明】

客户端错误事件，当出现不可恢复的错误后，Client会上报该事件通知。

【回调参数】

errorInfo：必选，HwLLSError类型，错误信息，详见**错误码（HwLLSError）**。

errorInfo定义为：{

code：必选，number类型，错误码。

message：必选，string类型，错误描述。

getCode(): number， 必选，返回错误码。

getMsg(): string，必选，返回错误描述。

}

注意

在网络防火墙限制（UDP端口限制）或者低时延直播多次重试播放失败的场景下，可根据特定错误码（HWLLS_MEDIA_NETWORK_ERROR、HWLLS_PLAY_WEBRTC_RETRY_FAILED），进行播放降级，请参考**SDK使用**。

3.6.4 错误码（HwLLSError）

getCode

```
getCode(): number
```

【功能说明】

获取错误码。

【请求参数】

无

【返回参数】

number类型，错误码值。

getMsg

```
getMsg(): string
```

【功能说明】

获取错误描述。

【请求参数】

无

【返回参数】

string类型，错误码描述。

3.6.5 公网地址

表 3-6 公网地址列表

公网地址	信息
log-collection-new.hwcloudlive.com	国内日志和打点环境地址。
log-collection-ap-southeast-3.rocket-cdn.com	海外日志和打点环境地址。
hcdnl-pull302-global-gslb.livehwc3.cn	默认GSLB环境地址。

3.6.6 客户端错误码

本章节介绍了低时延直播Web SDK的客户端错误码的详细信息。

表 3-7 错误码说明

类成员	错误码	描述	错误原因或建议处理方式
HWLLS_OK	0	成功。	-
HWLLS_ERROR_INVALID_URL	50000000	URL不合法。	检查URL是否正确。
HWLLS_ERROR_INVALID_PARAMETER	50000001	参数不合法。	参数传递错误，请检查接口入参是否符合接口的参数合法性要求。
HWLLS_ERROR_SERVER_CONNECT_FAIL	50000002	无法连接服务端。	检查网络状态是否正常或联系技术支持。
HWLLS_ERROR_SERVER_NO_RESPONSE	50000003	服务器无响应。	联系技术支持。
HWLLS_ERROR_AUTH_FAIL	50000004	鉴权失败。	请检查服务侧Referer防盗链以及key防盗链的配置是否正确。
HWLLS_ERROR_STREAM_NOT_EXIST	50000005	请求的流不存在。	使用存在的流。
HWLLS_ERROR_WEBRTC_UNSUPPORTED	50000006	浏览器不支持。	1、参见**浏览器适配**，使用可支持的浏览器。 2、建议降级FLV或HLS直播。
HWLLS_MEDIA_NETWORK_ERROR	50000007	媒体网络连接异常。	检查网络状态、防火墙配置是否正确，或者建议降级FLV或HLS直播。
HWLLS_ERROR_BAD_REQUEST	50000008	域名配置异常。	检查请求的流的域名是否正确，或联系技术支持。
HWLLS_ERROR_STREAM_INVALID_PARAMETER	50000009	流信息错误。	检查请求的流的URL是否正确。
HWLLS_INTERNAL_ERROR	50000020	其他内部错误。	联系技术支持。
HWLLS_BUSSINESS_DOWNGRADE	50000021	业务需要降级。	建议降级FLV或HLS直播。
HWLLS_PLAY_WEBRTC_RETRY_FAILED	50000022	低时延直播播放中断后，多次尝试重播失败	建议降级FLV或HLS直播。
HWLLS_PLAY_FLV_RETRY_FAILED	50000023	flv直播播放中断后，多次尝试重播失败	联系技术支持。

类成员	错误码	描述	错误原因或建议处理方式
HWLLS_PLAY_HLS_RETRY_FAILED	50000024	hls直播播放中断后，多次尝试重播失败	联系技术支持。
HWLLS_ERROR_LIVE_UNSUPPORTED	50000030	播放内容格式，浏览器不支持	联系技术支持。
HWLLS_ERROR_UNEXPECTED_EOF	50000031	播放内容，异常的网络EOF	建议重新尝试。
HWLLS_ERROR_MEDIA_ERROR	50000032	播放媒体内容异常	建议重新尝试或联系技术支持。
HWLLS_ERROR_REPORT_TOKEN_ERROR	50000033	打点和日志上传Token异常	联系技术支持。
HWLLS_PLAY_NOT_ALLOW	51000000	播放权限受限，需要手动触发播放。	由于浏览器自动播放安全策略的限制，浏览器直接拉起App并启动播放会返回该错误，在应用层需要根据该错误码，引导用户通过手动触发页面UI控件，调用**replay**接口恢复播放。 **须知** Safari浏览器还有以下处理方法： 打开Safari浏览器偏好设置 > 网站 > 自动播放 > 选择相应网站，设置允许全部自动播放。
HWLLS_PLAY_TIMEOUT	51000001	播放超时，3s（LLL）内没有拉到有效帧数据，FLV/HLS则是10s	需确认推流情况或联系技术支持。

3.7 常见问题

- **如果业务上App只能使用http协议，是否能够集成使用华为低时延直播Web SDK？**

  部分浏览器（chrome）可以集成使用，但不推荐。由于浏览器兼容性识别是根据浏览器暴露的WebRTC对象判断的，在非https协议下，对象可能不存在。

- **Firefox浏览器中无法使用华为低时延直播Web SDK?**

  Firefox浏览器使用之前需要安装H264的编解码插件。浏览器中输入**about:addons**，跳转到插件安装页面，查看H264插件是否安装完成，如未安装请在该页面更新安装。

- **集成华为低时延直播Web SDK后，无法正常使用，可能原因?**
  - 需要检查用户自定义的域名配置是否完成，如：推、拉流域名，权威机构签发的https证书等。

  - 推流端设置及推流是否正常。
  - 播放地址是否填写正确，如：appName、streamName等。
  - 网络连接是否正常、网络防火墙配置是否有限制，如：UDP端口（8000-8063）是否放通。

- **华为低时延直播Web SDK，支持哪些类型浏览器?**

  浏览器支持详情请参见**浏览器适配**。

- **推流端推流成功后，华为低时延直播Web SDK拉流播放失败?**

  需要确认推流端的推流编码参数，是否为H264+无B帧。目前华为低时延直播Web SDK仅支持H264+无B帧的流，所以如果原始流为H265或者带B帧，则需要提前在租户Console上配置对应转码模板，开启转码服务，但这样会引入额外的转码延迟，并且会产生转码费用。建议推流端尽量推H264+不包含B帧的流，可以通过调整推流端软件（如OBS）的视频编码参数去除B 帧。如果使用OBS推流，可以通过设置，关闭B帧。如下图所示：

- **华为低时延直播Web SDK，播放报错：NotAllowedError:xxx?**

  由于浏览器自动播放安全策略的限制，浏览器直接拉起App并启动播放会返回该错误，在应用层需要根据该错误码，引导用户通过手动触发页面UI控件，并调用**replay**接口恢复播放。

- **开启认证策略，该如何获取token信息?**

  认证策略不开启不影响功能正常使用，也不影响打点和日志上传的能力。

  认证策略开启可以保证打点数据和日志上传数据的安全性。

  如果当前需要认证策略能力，请**提交工单**，联系技术支持获取appid和token。

- **如何填写拉流配置参数?**

  调用startPlay开始拉流，其中参数options的字段elementId必填，如下所示：

  - elementId：容器的ID，用来承载展示适配画面的容器，一般传入div标签的id。

  其他一些字段均为可选，比如：

  - objectFit：渲染模式，有三种可选值，contain、cover和fill。
  - muted：表示是否静音播放。

- **如何处理音频受限?**

  音频受限是音频自动播放导致的，常见场景为，在一个界面未做任何交互，就直接播放音频。可以通过监听Error事件来获取相关信息，详情请参考**最佳实践**。

```
client.on('Error', (error) => {
  if (error.errCode === 51000000) {
```

```
        // 音频受限，增加交互操作，调用replay接口
    }
})
```

- **如何使用loading加载动画?**

  loading加载动画可以在起播、卡顿、网络质量差时展示加载的效果，可通过以下方法启用：

```
HWLLSPlayer.setParameter('LOADING_CONFIG', {
    netQualityLoading: true, // 根据网络质量展示loading
    netQualityLoadingThreshold: 5, // 展示loading的阈值，默认为5
    frameStuckLoading: true, // 根据帧卡顿时长展示loading
    frameStuckThreshold: 10, // 帧卡顿时长阈值，单位为100ms，10表示1000ms
})
```

- **如何使用海报（即视频播放封面）?**

  海报的设置是在startPlay接口的配置参数里，使用方法如下：

```
const options = {
    ...
    poster: {
        url: // 海报链接
        mode: crop, // 海报填充模式，可选fill或crop
        startEnable: true, // 表示启动播放时是否展示海报，不过只能在非自动起播下生效
        pauseEnable: true, // 表示暂停播放时是否展示海报
    }
    ...
}

client.startPlay(streamUrl, options)
```

- **如何获取统计信息?**

  详情请参考**客户端时间回调**。

  需要先开启流信息统计，如下所示：

```
client.streamStatistic(true, 1)
```

  再通过监听事件回调获取统计信息，如下所示：

```
client.on('media-statistic', (statisticInfo) => {
    const audioStatisticInfo = statisticInfo.audio
    const videoStatisticInfo = statisticInfo.video
})
```

- **如何处理画面黑屏?**

  当画面黑屏时，先检查推流是否有问题，主要包括：

  - 视频编码格式是否为H264。
  - 视频是否包含B帧。
  - 是否只推了音频，没有包含视频。

  如果确认推流没问题，可以使用FLV拉流查看视频是否正常。如果FLV正常，请**提交工单**处理。

  还有种情况是播放过程中出现黑屏，处理方式一般如下所示：

  - 画面黑屏后，检查音频是否正常。如果音频正常，再检查下推流端是否正常。
  - 音视频如果都没有，则可能是网络问题导致的断流。可在网络恢复时，重新调用开始播放接口，或使用断流重试功能。

- **如何使用断流重试?**

  如需使用断流重试功能，可在断流时自动重试拉流，优化用户体验。

  一般开启方式如下所示：

```
client.enableStreamStateDetection(
    true, // 流检测开关
```



```
  3, // 检测间隔，单位秒
  {
    enable: true, // 重试开关
    retryInterval: 30, // 重试间隔，单位：秒
    retryTimes: 30, // 重试次数
  }
)
```

- **如何处理兼容性问题?**

  某些设备的浏览器可能不支持WebRTC协议拉流，该场景下，可以采取降级播放策略。

- **全屏操作等常规功能不生效?**

  检查Client在使用过程中是否有保证为单例，调用Client的方法时，使用的Client是对的。如果是用vue等开发框架，建议Client实例不要设为响应式数据。

- **低端手机播放超高清视频，可能体验差**

  部分低端手机（如P20 Android9）在播放4K分辨率及8M以上码率的超高清直播流时，可能会因为性能问题，出现黑屏。

- **如何使用降级播放功能?**

  有两种降级方式，自动降级和指定降级，详见**最佳实践**。

  - 自动降级：SDK默认行为，在不支持WebRTC协议拉流时，自动降级为HLS或FLV。
  - 指定降级：在startPlay接口的配置参数里，指定降级地址，如下所示：

    ```
    const options = {
      ...
      downgradeUrl: {
        hlsUrl: // hls播放地址
        flvUrl: // flv播放地址
      }
      ...
    }

    client.startPlay(streamUrl, options)
    ```

- **视频被浏览器劫持导致UI页面展示异常等问题如何解决?**

  现象：浏览器可能劫持网页中的video播放器，对其添加一些UI等，使其展示效果不符合预期。如：劫持并创建新播放图层覆盖，导致自定义UI无法展示；object-fit设置不生效；暂停时出现广告播放等。

  解决方案：暂无自行解决方案，需浏览器厂商提供相应的配置方案。

- **视频初始化完成，未开始播放时黑屏或画面比例错误?**

  原因：视频非自动播放时，初始化完成后由浏览器决定展示的画面，可能会存在浏览器不展示或浏览器展示画面体验不佳的情况。

  解决方案：配置海报，指定非自动起播时展示的封面图片，详见**startPlay**中的poster配置项。

3.8 修订记录

表 3-8 修订记录

修改时间	修改说明
2024-02-17	1、优化播放异常的**事件通知**，补充**代码示例**。 2、优化自动降级策略。 3、优化降级播放逻辑；不再支持FLV、HLS独立播放器，旧文档迁移至**附录**。 4、新增两项**常见问题说明**。
2024-12-02	低时延直播Web SDK更新点，如下所示： • 支持自动降级和指定降级。 • 支持移动端后台播放音频。 • 解决设备兼容性问题。 • 支持多实例video标签id。
2024-11-12	新增**最佳实践**及其相关的**代码示例**。
2024-06-27	新增**setReportConfig**接口，更新SDK。
2024-03-19	1、新增SDK包下载路径及其完整性校验方法。 2、新增FLV、HLS等相关资料。
2023-10-30	第一次正式发布。

3.9 附录

3.9.1 客户端对象（HWFlvClient）

本章节介绍了低时延直播Web SDK的HWFlvClient接口详情。

表 3-9 主入口接口

接口	描述
startPlay	开始播放，客户端根据输入的URL到服务端拉取对应的主播流。
switchPlay	快速切换下一路流播放。
stopPlay	停止播放。
replay	重新播放。
resume	恢复播放。

接口	描述
pause	暂停播放。
pauseVideo	暂停视频。
resumeVideo	恢复视频。
pauseAudio	暂停音频。
resumeAudio	恢复音频。
setPlayoutVolume	设置播放音量。
getPlayoutVolume	获取音频音量。
muteAudio	静音。
streamStatistic	设置是否开启流信息统计。
enableStreamStateDetection	开启/关闭音视频码流状态探测功能。
destroyClient	销毁客户端对象。
fullScreenToggle	开启关闭全屏。

startPlay

```
startPlay(url: string, options: StartPlayOptions): Promise<void>
```

【功能说明】

开始播放，客户端根据输入的URL到服务端拉取对应的主播流。

【请求参数】

- url：必选，string类型。拉流URL，是以flv结尾的播放地址。
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。

> **注意**
>
> 手机端浏览器可能会创建控件覆盖SDK的播放器，导致配置无法生效，如OPPO浏览器。

- muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
- sessionId：不需要传。
- showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。
- autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
- poster：可选，对象定义如下：{
  - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
  - mode：可选，string类型。默认值为cover。支持的枚举值如下：{
    - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
    - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

    }
  - startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
  - pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

  }
- webrtcConfig：不需要传。
- schedulePolicy：不需要传。
- domainPolicy：不需要传。
- downgradeUrl：不需要传。

【返回参数】

Promise<void>：返回一个Promise对象。

switchPlay

```
switchPlay(url: string, options: StartPlayOptions): Promise<void>
```

【功能说明】

起播成功后，快速切换下一路流播放。

【请求参数】

- url：必选，string类型。拉流URL，是以flv结尾的播放地址。
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
  - sessionId：不需要传。
  - showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。
  - autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
  - poster：可选，对象定义如下：{
    - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
    - mode：可选，string类型。默认值为cover。支持的枚举值如下：{
      - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
      - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

      }
    - startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
    - pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

    }
  - webrtcConfig：不需要传。

- schedulePolicy：不需要传。
- domainPolicy：不需要传。
- downgradeUrl：不需要传。

【 返回参数 】

Promise<void>：返回一个Promise对象。

stopPlay

```
stopPlay(): boolean
```

【 功能说明 】

停止播放。

【 请求参数 】

无

【 返回参数 】

boolean：停止播放的结果。true表示成功，false表示失败。

replay

```
replay(): Promise<boolean>
```

【 功能说明 】

重新播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：重新播放的结果。true表示成功，false表示失败。

resume

```
resume(): Promise<boolean>
```

【 功能说明 】

恢复播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：恢复音视频播放的结果。true表示成功，false表示失败。

pause

```
pause(): boolean
```

【 功能说明 】

暂停音视频播放。

【 请求参数 】

无

【 返回参数 】

boolean：暂停播放的结果。true表示成功，false表示失败。

pauseVideo

```
pauseVideo(): boolean
```

【 功能说明 】

该接口不支持。

【 请求参数 】

无

【 返回参数 】

boolean：只返回false。

resumeVideo

```
resumeVideo(): Promise<boolean>
```

【 功能说明 】

该接口不支持。

【 请求参数 】

无

【 返回参数 】

boolean：只返回false。

pauseAudio

```
pauseAudio(): boolean
```

【 功能说明 】

暂停音频。

【 请求参数 】

无

【 返回参数 】

boolean：暂停音频播放的结果。true表示成功，false表示失败。

resumeAudio

```
resumeAudio(): Promise<boolean>
```

【 功能说明 】

恢复音频。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：恢复音频播放的结果。true表示成功，false表示失败。

setPlayoutVolume

```
setPlayoutVolume(volume: number): boolean
```

【 功能说明 】

设置音频音量，会开启声音。

【 请求参数 】

volume：必选，number类型，音频的音量值。取值范围为[0,100]。

【 返回参数 】

boolean：设置音频音量是否成功。true表示成功，false表示失败。

getPlayoutVolume

```
getPlayoutVolume(): number
```

【 功能说明 】

获取音频音量。

【 请求参数 】

无

【 返回参数 】

number：音量值，取值范围为[0,100]。

muteAudio

```
muteAudio(isMute: boolean): void
```

【 功能说明 】

静音。

【 请求参数 】

isMute：必选，boolean类型，是否静音。true表示静音，false表示取消静音。

【 返回参数 】

无

streamStatistic

```
streamStatistic(enable: boolean, interval: number): void
```

【 功能说明 】

设置是否开启流信息统计。

【 请求参数 】

- enable：必选，boolean类型，是否开启流信息统计，true表示开启统计。
- interval：必选，number类型，设置统计间隔，单位为秒，取值范围为[1, 60]，默认值为1。

【 返回参数 】

无

enableStreamStateDetection

```
enableStreamStateDetection(enable: boolean, interval: number, interruptRetry:StreamInterruptRetry): boolean
```

【 功能说明 】

开启/关闭音、视频码流状态探测功能，开启后可探测推流侧是否处于断流的状态。

【 请求参数 】

- enable：必选，boolean类型，true表示开启音视频码流状态探测，false表示关闭音视频码流状态探测。默认值为false。
- interval：必选，number类型，单位为秒，取值范围为[1,60]。音视频无码流状态的判断时间。默认值为3，推荐设置为3。
- interruptRetry：可选，StreamInterruptRetry类型。断流重试播放配置参数，StreamInterruptRetry定义为：{

  enable：boolean类型，开启断流后尝试自动恢复播放。默认值为false，即不开启自动重试。

  retryInterval：number类型，拉流播放的重试周期，单位为秒。最小值10，最大值建议不超过60，默认值为30。

  retryTimes：number类型，尝试重新恢复播放的最大重试次数。最小值1，默认值为30。

  }

【 返回参数 】

boolean：是否成功，true表示成功，false表示失败。

destroyClient

```
destroyClient(): void
```

【 功能说明 】

销毁客户端对象。

【 请求参数 】

无

【 返回参数 】

无

fullScreenToggle

```
fullScreenToggle(isExit: boolean): void
```


【功能说明】

开启关闭全屏。

【请求参数】

isExit：必选，布尔类型，默认false。

【返回参数】

无

3.9.2 客户端对象（HWHlsClient）

本章节介绍了低时延直播Web SDK的HWHlsClient接口详情。

表 3-10 主入口接口

接口	描述
startPlay	开始播放，客户端根据输入的URL到服务端拉取对应的主播流。
switchPlay	快速切换下一路流播放。
stopPlay	停止播放。
replay	重新播放。
resume	恢复播放。
pause	暂停播放。
pauseVideo	暂停视频。
resumeVideo	恢复视频。
pauseAudio	暂停音频。
resumeAudio	恢复音频。
setPlayoutVolume	设置播放音量。
getPlayoutVolume	获取音频音量。
muteAudio	静音。
streamStatistic	设置是否开启流信息统计。
enableStreamStateDetection	开启/关闭音视频码流状态探测功能。
destroyClient	销毁客户端对象。
fullScreenToggle	开启关闭全屏。

startPlay

```
startPlay(url: string, options: StartPlayOptions): Promise<void>
```


【功能说明】

开始播放，客户端根据输入的URL到服务端拉取对应的主播流。

【请求参数】

- url：必选，string类型。拉流URL，是以m3u8结尾的播放地址
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。

    ⚠ 注意

    手机端浏览器可能会创建控件覆盖SDK的播放器，导致配置无法生效，如OPPO浏览器。

  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
  - sessionId：不需要传。
  - showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。

    ⚠ 注意

    - Android端QQ浏览器不支持该功能。
    - 建议showLoading不设置，或者设置为false。

  - autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
  - poster：可选，对象定义如下：{
    - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。

- mode：可选，string类型。默认值为cover。支持的枚举值如下：{
  - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

  }
- startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
- pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

}

- webrtcConfig：不需要传。
- schedulePolicy：不需要传。
- domainPolicy：不需要传。
- downgradeUrl：不需要传。

【返回参数】

Promise<void>：返回一个Promise对象。

switchPlay

```
switchPlay(url: string, options: StartPlayOptions): Promise<void>
```

【功能说明】

起播成功后，快速切换下一路流播放。

【请求参数】

- url：必选，string类型。拉流URL，是以m3u8结尾的播放地址
- options：可选，StartPlayOptions类型。播放配置参数，如果不携带该参数，则复用首次起播携带的options数据。StartPlayOptions定义如下：{
  - elementId：必选，播放DOM标识ID。
  - objectFit：可选，string类型，默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。

- sessionId：不需要传。
- showLoading：可选，boolean类型，true表示开启loading的展示效果，默认为false。当该参数设置为true时，起播loading效果同步开启，播放过程中发生缓冲时loading的效果，需根据**setParameter**接口中的LOADING_CONFIG进行设置。
- autoPlay：可选，boolean类型，true表示开启自动起播功能，false表示非自动起播，需要人为触发播放，默认为true。
- poster：可选，对象定义如下：{
  - url：可选，string类型。设置播放封面图片完整地址，图片格式限JPGPNG和静态GIF格式，大小不超过1MB，尺寸不超过1920 x 1080，文件名不得含有中文字符。
  - mode：可选，string类型。默认值为cover。支持的枚举值如下：{
    - fill：视频内容完全填充视窗，如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
    - crop：海报（即视频播放封面）原始尺寸大小展示，如果超出播放区域，则会对超出部分进行裁剪，否则在播放窗口居中展示。

    }
  - startEnable：可选，boolean类型。启动播放时是否展示播放封面，true表示展示，false表示不展示播放封面，默认值false。该参数只在设置非自动播放场景下生效。
  - pauseEnable：可选，boolean类型。触发暂停操作时，是否在播放页面展示播放封面，true表示展示播放封面，false表示不展示，默认值false。

  }
- webrtcConfig：不需要传。
- schedulePolicy：不需要传。
- domainPolicy：不需要传。
- downgradeUrl：不需要传。

【 返回参数 】

Promise<void>：返回一个Promise对象。

stopPlay

```
stopPlay(): boolean
```

【 功能说明 】

停止播放。

【 请求参数 】

无

【 返回参数 】

boolean：停止播放结果。true表示成功，false表示失败。

replay

```
replay(): Promise<boolean>
```

【 功能说明 】

重新播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：重新播放的结果。true表示成功，false表示失败。

resume

```
resume(): Promise<boolean>
```

【 功能说明 】

恢复播放。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：恢复音视频播放的结果。true表示成功，false表示失败。

pause

```
pause(): boolean
```

【 功能说明 】

暂停音视频播放。

【 请求参数 】

无

【 返回参数 】

boolean：暂停播放的结果。true表示成功，false表示失败。

pauseVideo

```
pauseVideo(): boolean
```

【 功能说明 】

该接口不支持。

【 请求参数 】

无

【 返回参数 】

boolean：只返回false。

resumeVideo

```
resumeVideo(): Promise<boolean>
```

【 功能说明 】

该接口不支持。

【 请求参数 】

无

【 返回参数 】

boolean：只返回false。

pauseAudio

```
pauseAudio(): boolean
```

【 功能说明 】

暂停音频。

【 请求参数 】

无

【 返回参数 】

boolean：暂停音频播放结果。true表示成功，false表示失败。

resumeAudio

```
resumeAudio(): Promise<boolean>
```

【 功能说明 】

恢复音频。

【 请求参数 】

无

【 返回参数 】

Promise<boolean>：恢复音频播放的结果。true表示成功，false表示失败。

setPlayoutVolume

```
setPlayoutVolume(volume: number): boolean
```

【 功能说明 】

设置音频音量，会开启声音。

【 请求参数 】

volume：必选，number类型，音频的音量值。取值范围为[0,100]。

【 返回参数 】

boolean：设置音频音量是否成功。true表示成功，false表示失败。

getPlayoutVolume

```
getPlayoutVolume(): number
```

【 功能说明 】

获取音频音量。

【 请求参数 】

无

【 返回参数 】

number：音量值，取值范围为[0,100]。

muteAudio

```
muteAudio(isMute: boolean): void
```

【 功能说明 】

静音。

【 请求参数 】

isMute：必选，boolean类型，是否静音。true表示静音，false表示取消静音。

【 返回参数 】

无

streamStatistic

```
streamStatistic(enable: boolean, interval: number): void
```

【 功能说明 】

设置是否开启流信息统计。

【 请求参数 】

- enable：必选，boolean类型，是否开启流信息统计，true表示开启统计。
- interval：必选，number类型，设置统计间隔，单位为秒，取值范围为[1, 60]，默认值为1。

【 返回参数 】

无

enableStreamStateDetection

```
enableStreamStateDetection(enable: boolean, interval: number, interruptRetry:StreamInterruptRetry): boolean
```

【 功能说明 】

开启/关闭音、视频码流状态探测功能，开启后可探测推流侧是否处于断流的状态。

【 请求参数 】

- enable：必选，boolean类型，true表示开启音视频码流状态探测，false表示关闭音视频码流状态探测。默认值为false。

- interval：必选，number类型，单位为秒，取值范围为[1,60]。音视频无码流状态的判断时间。默认值为3，推荐设置为3。
- interruptRetry：可选，StreamInterruptRetry类型。断流重试播放配置参数，StreamInterruptRetry定义为：{

  enable：boolean类型，开启断流后尝试自动恢复播放。默认值为false，即不开启自动重试。

  retryInterval：number类型，拉流播放的重试周期，单位为秒。最小值10，最大值建议不超过60，默认值为30。

  retryTimes：number类型，尝试重新恢复播放的最大重试次数。最小值1，默认值为30。

  }

【返回参数】

boolean：是否成功，true表示成功，false表示失败。

⚠ 注意

Android端QQ浏览器不支持该功能。

destroyClient

```
destroyClient(): void
```

【功能说明】

销毁客户端对象。

【请求参数】

无

【返回参数】

无

fullScreenToggle

```
fullScreenToggle(isExit: boolean): void
```

【功能说明】

该接口不支持。

【请求参数】

isExit：必选，布尔类型，默认为false。

【返回参数】

无

4 修订记录

表 4-1 修订记录

发布日期	修订记录
2025-02-17	Web SDK版本更新至2.10.9。
2024-12-02	Web SDK版本更新至2.10.3。
2024-06-27	Web SDK版本更新至2.6.9。
2024-06-12	Web SDK版本更新至2.6.3。
2023-10-30	第一次正式发布。