Low Latency Live

Client SDK Reference

Issue	01
Date	2024-05-08

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 SDK Overview

The SDK of Huawei Cloud Low Latency Live (LLL) encapsulates REST APIs provided by LLL to simplify development. You can directly call API functions provided by the LLL SDK to use LLL. For details about how to download and integrate the client SDK and use APIs, see **Table 1-1**.

Table 1-1 Client SDK

Client	SDK Download	Version	Developed By	Function	SDK Reference	API Reference
Web	SDK: **HWLLS_SDK_Web_2.6.3.tar.gz** Integrity check: **HWLLS_SDK_Web_2.6.3.tar.gz.sha256**	2.6.3	Huawei Cloud	LLL functions	**Web SDK Reference**	**Web API Reference**

Checking Software Package Integrity

Check the integrity of downloaded SDK packages, that is, check whether the packages are tampered with or packets are lost during download.

The procedure is as follows:

Step 1 Download the SDK package and its integrity verification SHA-256 package in **Table 1-1** to the local PC.

Step 2 Open the local CLI and run the following command to generate the SHA-256 value of the downloaded SDK package on the local PC.

In the following command, **D:\HWLLS_SDK_Web_2.6.0.tar.gz** indicates the local path for storing the SDK package and the SDK package name. Change it as needed.

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


Example command output:

```
D:\HWLLS_SDK_Web_2.6.0.tar.gz hash of SHA-256:
3ac83be852e8dcc9e90f236801fd4c494983073543e1ae66ee4d0c29043dccd1
CertUtil: -hashfile Command executed.
```

Step 3 Compare the SHA-256 value of the downloaded SDK package with that of the queried SDK package.

If they are the same, no tampering or packet loss occurred during download.

----End

2 Version Planning

This section describes the version planning of the LLL client SDK.

Version Description

The version number is in the format of a.b.c, where:

- a indicates the major version number, which is updated when the version architecture is reconstructed. For example, the major version number will be changed if the APIs of different versions are incompatible.
- b indicates the minor version number, which is an iterative version. If there are new functions or features, or APIs are added or optimized, the value of this field is incremented by 1.
- c indicates the patch version number. If a function is optimized or a defect is rectified, the value of this field is incremented by 1.

Example version number: 2.0.1

Version Lifecycle

By default, a version is released every one to two months, or a version is modified based on customer requirements.

Version Constraints

None. The old and new versions are compatible.

3 Web SDK

3.1 Browser Adaptation

This section describes the browser types and versions supported by the LLL Web SDK and constraints.

Table 3-1 Browser adaptation

OS	Browser Type	Browser Version
Windows	Chrome browser	67+
	QQ Browser (fast kernel)	10.4+
	360 Secure Browser (fast mode)	12
	WeChat embedded browser	-
	Mozilla Firefox	90+
	Microsoft Edge	80+
	Opera browser	54+
macOS	Chrome browser	67+
	WeChat embedded browser	-
	Safari	13+
	Mozilla Firefox	90+
	Opera browser	56+
Android	WeChat embedded browser (TBS kernel)	-
	WeChat embedded browser (XWEB kernel)	-

OS	Browser Type	Browser Version
	Mobile Google Chrome	83+
	Mobile QQ Browser	12+
	Huawei browser	11.0.8+
iOS	WeChat embedded browser	iOS 14.3+ WeChat 6.5+
	Mobile Google Chrome	iOS 14.3+
	Mobile Safari	13+

Table 3-2 Constraints on browsers

Browser Type	Constraint
Chrome browser	1. On Huawei mobile devices, the Chrome browser (including the Huawei browser) supports WebRTC 91 or later. 2. WebView on Android mobile devices supports WebRTC. The support varies due to factors such as device vendors, browser kernels, and versions. You are advised to select the most compatible browser.
Safari	• Safari 13 users may not hear the voice of remote users. • The audio on iOS Safari 14.2 and macOS Safari 14.0.1 may be intermittent.
Mozilla Firefox	Firefox on macOS devices with Apple M1 chips does not support H.264 codec.
Opera browser	On Huawei mobile devices, the Opera browser supports WebRTC 64 or later.
Other browsers	The support for WebRTC varies with mobile browsers due to factors such as browser kernels, webviews, and versions of different vendors. In addition to the browsers listed in **Table 1 Browser adaptation**, native SDK (Android/iOS) can be integrated.

3.2 Preparations

Prerequisites

The SDK package has been downloaded.

Environment Requirements

- You are advised to install Microsoft Visual Studio Code 1.43.2 or a later version for compilation.
- If Node.js is used for development on the client, you are advised to install Node.js 14.19.1 or later.
- For details about the supported browsers, see **Browser Adaptation**.
- If you want to develop your client using TypeScript, the TypeScript version must be 3.8.3 or later.

SDK Integration

Step 1 Download the **SDK** to the local PC. You are advised to save the SDK package to a directory named **sdk** of your project.

Step 2 Introduce **HWLLSPlayer** to the project code.

- To use the <script> mode to introduce the SDK, access HWLLSPlayer to obtain the exported module:

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

- To directly reference the static JS file of SDK, reference the following code to access HRTC:

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

- To introduce the SDK in npm modularization mode, install the HWLLSPlayer module and introduce HWLLSPlayer to the development dependency of package.json, for example, **"HWLLSPlayer": "./sdk/HWLLS_SDK_Web_*.*.*.***.tar.gz"**. Run the **npm install** command on the terminal (replace the version number with the actual one), and then run the following commands:

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

----End

3.3 SDK Usage

Step 1 Check whether a browser is compatible with the SDK. For details, see **checkSystemRequirements**.

```
async isBrowserSupport() {
  let check = false
  try {
    check = await HWLLSPlayer.checkSystemRequirements()
    console.warn('browser isSupport: ' + check)
  } catch (error) {
    console.error(`check browser isSupport error: ${error.getCode()} - ${error.getMsg()}`)
    if (error.getCode && error.getCode() === 50000006) {
      let commonClient = HWLLSPlayer.createClient("flv") // "hls"
      await commonClient.startPlay(url,options)
    }
  }
  return check
}
```

Step 2 Create a client. For details, see **createClient**.


```
let client = HWLLSPlayer.createClient("webrtc")
```

Step 3 Register event listening. For example, if the playback is successful, the client receives the **video-start** and **audio-start** notifications. If the playback fails, the client receives the **Error** notification. The service can be downgraded according to the error.

```
client.on('video-start', () => {
  console.log('video-start')
})

client.on('audio-start', () => {
  console.log('audio-start')
})

client.on('Error', (error) => {
  if (error.errCode === 50000007 || error.errCode === 50000022) {
    let commonClient = HWLLSPlayer.createClient("flv") // "hls"
    await commonClient.startPlay(url,options)
  }
})
```

Step 4 Start the playback. For details, see **startPlay**.

```
async startPlay() {
  try{
    await client.startPlay(url,options)
    console.log('startPlay success')
  } catch(error){
    console.log('startPlay fail',error)
    if (error.getCode && error.getCode() === 50000021) {
      let commonClient = HWLLSPlayer.createClient("flv") // "hls"
      await commonClient.startPlay(url,options)
    }
  }
}
```

- **url**: (mandatory) ingest URL. The value is a string. For details about the URL format, see **startPlay**.
- **options**: (optional) Playback configuration parameter.

  The definition of StartPlayOptions is as follows: {

  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:

    - **contain**: preferentially ensures that all video content is displayed. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: preferentially ensures that the window is filled. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.

- **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
- **sessionId**: (optional) String type. Indicates the unified ID of a complete session.
- **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
- **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
- **poster**: (optional) The object definition is as follows: {
  - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
  - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }
  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
  - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

  }
- **webrtcConfig**: (optional) WebRTCConfig type. Pulls streams of a specified media type. WebRTCConfig is defined as follows: {
  - **receiveVideo**: (optional) Boolean type. Pulls a video for playback. **true** indicates that a video is pulled for playback, and **false** indicates that a video is not pulled for playback. The default value is **true**. This parameter and **receiveAudio** cannot be set to **false** at the same time.
  - **receiveAudio**: (optional) Boolean type. Pulls an audio for playback. **true** indicates that an audio is pulled for playback, and **false** indicates that an audio is not pulled for playback. The default value

is **true**. This parameter and **receiveVideo** cannot be set to **false** at the same time.

}

- **schedulePolicy**: (optional) SchedulePolicy type. Specifies the access scheduling policy. The definition of **SchedulePolicy** is as follows: {
  - **DNS**: (optional) String type. Indicates that the domain name is resolved by DNS for being accessed. The default value is **DNS**.
  - **HTTPDNS**: (optional) String type. Indicates that **HTTPDNS** is used for the access domain name.

  }
- **domainPolicy**: (optional) DomainPolicy type. Specifies the policy of an access domain name. This setting takes effect only when **schedulePolicy** is set to **DNS**. **DomainPolicy** is defined as follows: {
  - **0**: (optional) Number type. Indicates that the user-defined domain name is used. The default value is **0**.
  - **1**: (optional) Number type. Indicates that the public access domain name is used.

  }

Step 5 Stop the playback. For details, see **stopPlay**.

```
stopPlay() {
  try {
    client.stopPlay()
    console.log('stopPlay success')
  } catch (error) {
    console.log('stopPlay fail', error)
  }
}
```

Step 6 Destroy a client object. For details, see **destoryClient**.

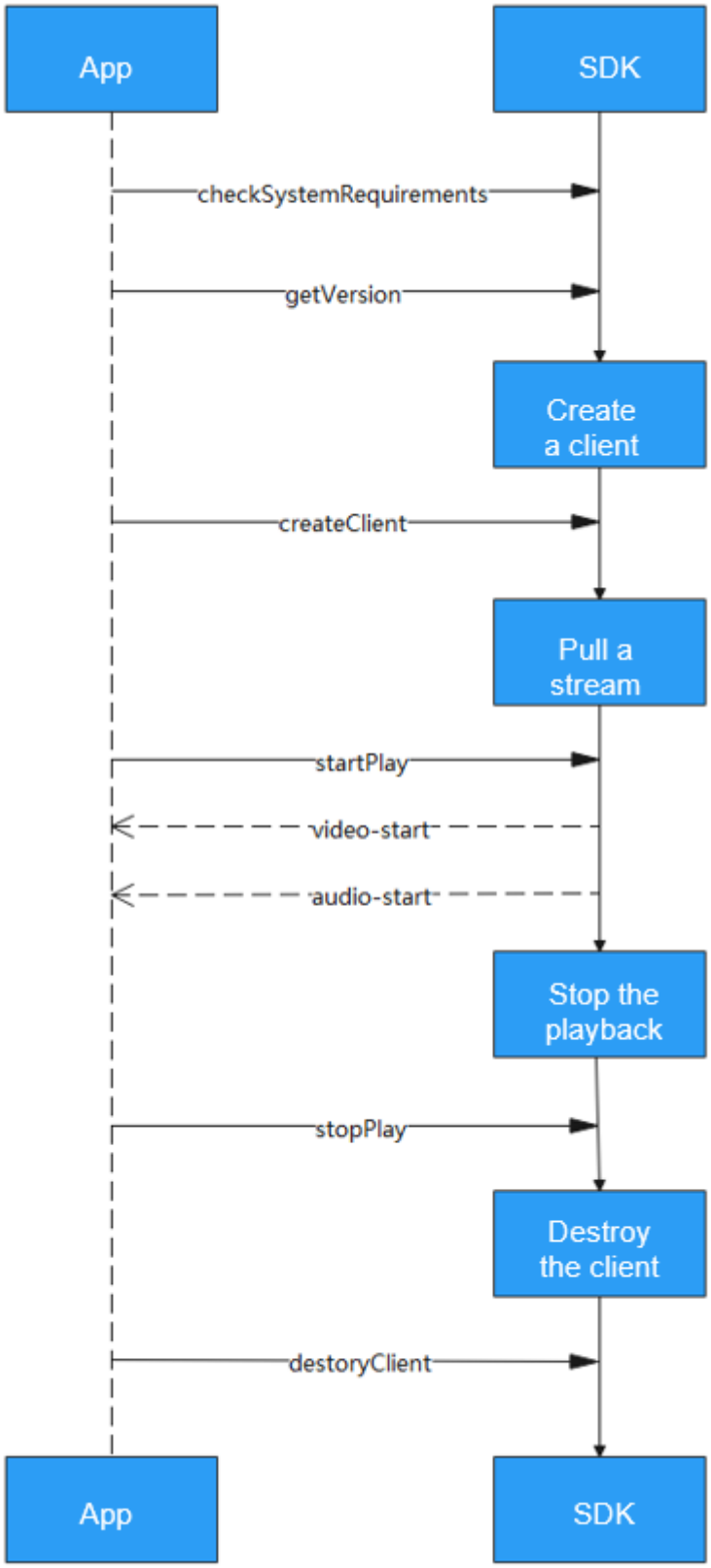
```
client.destoryClient()
```

----End

3.4 Basic Usage Logic

The procedure is as follows:

- Pre-processing: Obtain the browser version, check the compatibility, and create a client.
- Stream pull for playback: Send a stream pull request.
- Stop the playback: Stop the playback request.
- Post-processing: Destroy the client.



NOTE

In the preceding figure, click the name of an API to view its usage.

3.5 API Reference

3.5.1 Main Entry (HWLLSPlayer)

This section describes the HWLLSPlayer APIs of the LLL Web SDK.

Table 3-3 HRTC APIs

API	Description
checkSystemRequirements	Checks whether the browser supports the LLL Web SDK.
getVersion	Obtains the SDK version number.
createClient	Creates a client object for pulling live streams. If multiple live streams need to be pulled, you need to create multiple client objects.
uploadLog	Uploads logs.
saveLog	Saves logs.
setParameter	Configures global parameters.
setLogLevel	Sets the level of logs to be printed on the console.

checkSystemRequirements

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

[Function Description]

Checks whether the browser supports the LLL Web SDK.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: A Promise object is returned. The value **true** indicates that the browser is compatible with the LLL Web SDK. If they are incompatible, an error is returned.

CAUTION

LLL requires the WebRTC capability but some browsers may not support WebRTC playback. In this case, you can downgrade the playback based on the specific error code (**HWLLS_ERROR_WEBRTC_UNSUPPORTED**). For details, see **SDK Usage**.

getVersion

```
getVersion(): string
```

[Function Description]

Obtains the current SDK version number.

[Request Parameters]

None

[Response Parameters]

string: current SDK version number.

createClient

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

[Function Description]

Creates a client object for pulling live streams. If multiple live streams need to be pulled, you need to create multiple client objects.

[Request Parameters]

type: (optional) String type. Indicates the type of the client created for pulling streams.

- Type of the client for pulling LLL streams: **webrtc**
- Type of the client for pulling FLV streams: **flv**.
- Type of the client for pulling HLS streams: **hls** (reserved)

Default value: **webrtc**

[Response Parameters]

client: client object for pulling streams.

uploadLog

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

[Function Description]

Uploads logs.

[Request Parameters]

None

[Response Parameters]

Promise<void>: If **tryCatch** is used to obtain errors as an array, error information corresponding to multiple app IDs will be returned.

saveLog

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

[Function Description]

Users can flexibly save logs.

[Request Parameters]

None

[Response Parameters]

Promise<Blob>: **Promise<Blob>** compressed in .zip format, which can be directly saved as a .zip file.

setParameter

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

[Function Description]

Configures global parameters.

[Request Parameters]

Parameter	Value

LOADING_CONFIG	LoadingConfig is defined as:
	{
	netQualityLoading: (optional) The type is Boolean. **true** indicates that the loading effect is displayed based on the network quality. The default value is **false**, indicating that the loading effect is not displayed.
	netQualityLoadingThreshold: (optional) The type is number. Indicates the threshold of the **network-quality** for displaying the loading effect. The default network quality level is **5**.
	frameStuckLoading: (optional) The type is Boolean. **true** indicates that the loading effect is displayed based on the frame freezing duration. The default value is **false**, indicating that the loading effect is not displayed.
	frameStuckThreshold: (optional) The type is number. Indicates the frame freezing duration threshold for displaying the loading effect. The unit is 100 ms. The default value is **10**, indicating that the frame freezing duration is 1000 ms.
	}
	CAUTION
	You need to configure this parameter before **starting playback**.

DNS_QUERY_ENABLE	Boolean type. **true** indicates that DNS resolution is enabled. **false** indicates that DNS resolution is disabled.

Parameter	Value
ACCESS_DOMAIN	String type. By default, this parameter is not specified, is used to configure the stream pull environment, and needs to be set by Huawei engineers.
GLSB_DOMAIN	String type. By default, this parameter is not specified, is used to configure the GSLB environment, and needs to be set by Huawei engineers.

[Response Parameters]

boolean: configuration parameter setting result. **true** indicates that the setting is successful. **false** indicates that the setting fails.

setLogLevel

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

[Function Description]

Set the level of logs to be printed on the console. Otherwise, the default level is **error**.

[Request Parameters]

level: (mandatory) String type. A log level identifier.

- **none**: Disables the function of printing logs of all levels.
- **error**: Logs of the error level are printed.
- **warn**: Logs of the warn level and higher levels are printed.
- **info**: Logs of the info level and higher levels are printed.
- **debug**: Logs of the debug level and higher levels are printed.

[Response Parameters]

boolean: log level setting result. **true** indicates that the log level is set successfully. **false** indicates that the log level fails to be set.

3.5.2 Client Object (HWLLSClient)

This section describes the HWLLSClient APIs of the LLL Web SDK.

Table 3-4 HRTC APIs

API	Description
startPlay	Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.

API	Description
switchPlay	Quickly switches to the next stream.
stopPlay	Stops the playback.
replay	Replays the video.
startPlayerCustomize	Customizes the playback.
resume	Resumes the playback.
pause	Pauses the playback.
pauseVideo	Pauses the video playback.
resumeVideo	Resumes the video playback.
pauseAudio	Pauses the audio playback.
resumeAudio	Resumes the audio playback.
setPlayoutVolume	Sets the playback volume.
getPlayoutVolume	Obtains the audio volume.
muteAudio	Mutes a site or all sites.
streamStatistic	Specifies whether to enable stream statistics.
enableStreamStateDetection	Enables or disables media stream status detection.
on	Registers the callback for a client object event.
off	Deregisters the callback for a client object event.
destoryClient	Destroys a client object.
fullScreenToggle	Enables/Disables full-screen display.

startPlay

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

[Function Description]

Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.

[Request Parameters]

- **url**: (mandatory) String type. A streaming URL is in the format of **webrtc://***{domain}*/*{AppName}*/*{StreamName}*.
  - **webrtc://**: The value is fixed, indicating that streams are pulled using WebRTC.
  - *domain*: indicates the streaming domain name. Use the streaming domain name registered with Huawei Cloud.

  - *AppName*: indicates the application name. Use the application name registered with Huawei Cloud.
  - *StreamName*: indicates the stream name, which must be the same as the pushed stream name.
- **options**: (mandatory) Playback configuration parameter. The value is of StartPlayOptions type. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: preferentially ensures that all video content is displayed. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: preferentially ensures that the window is filled. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
  - **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
  - **sessionId**: (optional) String type. Indicates the unified ID of a complete session.
  - **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
  - **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
  - **poster**: (optional) The object definition is as follows: {
    - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
    - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
      - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.

  - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

  }

- **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
- **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

}

– **webrtcConfig**: (optional) WebRTCConfig type. Specifies parameters for pulling streams of a specified media type. WebRTCConfig is defined as follows: {

- **receiveVideo**: (optional) Boolean type. Pulls a video for playback. **true** indicates that a video is pulled for playback, and **false** indicates that a video is not pulled for playback. The default value is **true**. This parameter and **receiveAudio** cannot be set to **false** at the same time.
- **receiveAudio**: (optional) Boolean type. Indicates whether to pull an audio for playback. **true** indicates that an audio is pulled for playback, and **false** indicates that an audio is not pulled for playback. The default value is **true**. This parameter and **receiveVideo** cannot be set to **false** at the same time.

}

– **schedulePolicy**: (optional) SchedulePolicy type. Specifies the access scheduling policy. The definition of **SchedulePolicy** is as follows: {

- **DNS**: (optional) String type. Indicates that the domain name is resolved by DNS for being accessed. The default value is **DNS**.
- **HTTPDNS**: (optional) String type. Indicates that **HTTPDNS** is used for the access domain name.

}

– **domainPolicy**: (optional) DomainPolicy type. Specifies the policy of an access domain name. This setting takes effect only when **schedulePolicy** is set to **DNS**. **DomainPolicy** is defined as follows: {

- **0**: (optional) Number type. Indicates that the user-defined domain name is used. The default value is **0**.
- **1**: (optional) Number type. Indicates that the public access domain name is used.

}

– **downgradeUrl**: (optional) The object definition is as follows: {

- **hlsUrl?**: (optional) String type. Identifies the downgraded HLS streaming URL.

- **flvUrl?**: (optional) String type. Identifies the downgraded FLV streaming URL.

}

[Response Parameters]

Promise<void>: returns a **Promise** object.

> ⚠ CAUTION
>
> When the LLL service is faulty, the error code **HWLLS_BUSINESS_DOWNGRADE** can be used to downgrade the playback. For details, see **SDK Usage**.

switchPlay

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

[Function Description]

After the playback has been started, the system quickly switches to the next stream.

[Request Parameters]

- **url**: (mandatory) Ingest URL. A streaming URL is in the format of **webrtc://***{domain}*/*{AppName}*/*{StreamName}*.
  - **webrtc://**: The value is fixed, indicating that streams are pulled using WebRTC.
  - *domain*: indicates the streaming domain name. Use the streaming domain name registered with Huawei Cloud.
  - *AppName*: indicates the application name. Use the application name registered with Huawei Cloud.
  - *StreamName*: indicates the stream name, which must be the same as the pushed stream name.
- **options**: (optional) Playback configuration parameter. The value is of StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: preferentially ensures that all video content is displayed. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: preferentially ensures that the window is filled. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video

stream will be cropped or the image will be stretched to fill the display window.

- **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.

– **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.

– **sessionId**: (optional) String type. Indicates the unified ID of a complete session.

– **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.

– **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.

– **poster**: (optional) The object definition is as follows: {

- **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
- **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
  - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
  - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

  }
- **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
- **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

}

– **webrtcConfig**: (optional) WebRTCConfig type. Specifies parameters for pulling streams of a specified media type. WebRTCConfig is defined as follows: {

- **receiveVideo**: (optional) Boolean type. Pulls a video for playback. **true** indicates that a video is pulled for playback, and **false** indicates

that a video is not pulled for playback. The default value is **true**. This parameter and **receiveAudio** cannot be set to **false** at the same time.

- **receiveAudio**: (optional) Boolean type. Indicates whether to pull an audio for playback. **true** indicates that an audio is pulled for playback, and **false** indicates that an audio is not pulled for playback. The default value is **true**. This parameter and **receiveVideo** cannot be set to **false** at the same time.

}

- **schedulePolicy**: (optional) SchedulePolicy type. Specifies the access scheduling policy. The definition of **SchedulePolicy** is as follows: {
  - **DNS**: (optional) String type. Indicates that the domain name is resolved by DNS for being accessed. The default value is **DNS**.
  - **HTTPDNS**: (optional) String type. Indicates that **HTTPDNS** is used for the access domain name.

  }
- **domainPolicy**: (optional) DomainPolicy type. Specifies the policy of an access domain name. This setting takes effect only when **schedulePolicy** is set to **DNS**. **DomainPolicy** is defined as follows: {
  - **0**: (optional) Number type. Indicates that the user-defined domain name is used. The default value is **0**.
  - **1**: (optional) Number type. Indicates that the public access domain name is used.

  }

[Response Parameters]

Promise<void>: returns a **Promise** object.

CAUTION

When the LLL service is faulty, the error code **HWLLS_BUSINESS_DOWNGRADE** can be used to downgrade the playback. For details, see **SDK Usage**.

stopPlay

```
stopPlay(): boolean
```

[Function Description]

Stops the playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of stopping playback. The options are **true** (success) and **false** (failure).

replay

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

[Function Description]

Replays the video.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: replay result. The options are **true** (success) and **false** (failure).

startPlayerCustomize

```
startPlayerCustomize(url: strin): Promise<boolean>
```

[Function Description]

Customizes the playback.

[Request Parameters]

- **url**: (mandatory) Ingest URL. Streaming URL, which is a customized URL for playback.
- **options**: (optional) Playback configuration parameter. The value is of StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: preferentially ensures that all video content is displayed. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: preferentially ensures that the window is filled. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
  - **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
  - **sessionId**: (optional) String type. Indicates the unified ID of a complete session.

- **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
- **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
- **poster**: (optional) The object definition is as follows: {
  - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
  - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }
  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
  - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

  }
- **webrtcConfig**: (optional) WebRTCConfig type. Specifies parameters for pulling streams of a specified media type. WebRTCConfig is defined as follows: {
  - **receiveVideo**: (optional) Boolean type. Pulls a video for playback. **true** indicates that a video is pulled for playback, and **false** indicates that a video is not pulled for playback. The default value is **true**. This parameter and **receiveAudio** cannot be set to **false** at the same time.
  - **receiveAudio**: (optional) Boolean type. Indicates whether to pull an audio for playback. **true** indicates that an audio is pulled for playback, and **false** indicates that an audio is not pulled for playback. The default value is **true**. This parameter and **receiveVideo** cannot be set to **false** at the same time.

}

- **schedulePolicy**: (optional) SchedulePolicy type. Specifies the access scheduling policy. The definition of **SchedulePolicy** is as follows: {
  - **DNS**: (optional) String type. Indicates that the domain name is resolved by DNS for being accessed. The default value is **DNS**.
  - **HTTPDNS**: (optional) String type. Indicates that **HTTPDNS** is used for the access domain name.

  }
- **domainPolicy**: (optional) DomainPolicy type. Specifies the policy of an access domain name. This setting takes effect only when **schedulePolicy** is set to **DNS**. **DomainPolicy** is defined as follows: {
  - **0**: (optional) Number type. Indicates that the user-defined domain name is used. The default value is **0**.
  - **1**: (optional) Number type. Indicates that the public access domain name is used.

  }

[Response Parameters]

Promise<void>: returns a **Promise** object.

resume

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

[Function Description]

Resumes the playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio/video playback. The options are **true** (success) and **false** (failure).

pause

```
pause(): boolean
```

[Function Description]

Pauses the audio/video playback.

[Request Parameters]

None

[Response Parameters]

boolean: playback pause result. The options are **true** (success) and **false** (failure).

pauseVideo

```
pauseVideo(): boolean
```


[Function Description]

This parameter is used to pause a video. After the video is paused, it freezes.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing video playback. The options are **true** (success) and **false** (failure).

resumeVideo

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

[Function Description]

Resumes the video playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the video playback. The options are **true** (success) and **false** (failure).

pauseAudio

```
pauseAudio(): boolean
```

[Function Description]

Pauses the audio playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing the audio playback. The options are **true** (success) and **false** (failure).

resumeAudio

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

[Function Description]

Resumes the audio playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio playback. The options are **true** (success) and **false** (failure).

setPlayoutVolume

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

[Function Description]

Sets the audio volume. This API is not supported on iOS.

[Request Parameters]

volume: (mandatory) audio volume. The value is a number ranging from 0 to 100.

[Response Parameters]

boolean: whether the operation is successful. The options are **true** (success) and **false** (failure).

getPlayoutVolume

```
getPlayoutVolume(): number
```

[Function Description]

Obtains the audio volume.

[Request Parameters]

None

[Response Parameters]

number: volume value. The value ranges from 0 to 100.

muteAudio

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

[Function Description]

Mutes.

[Request Parameters]

- **isMute**: (mandatory) indicates whether the device is muted. The value is of the Boolean type. **true** indicates that the device is muted, and **false** indicates that the device is unmuted.

[Response Parameters]

None

streamStatistic

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

[Function Description]

Specifies whether to enable stream statistics.

[Request Parameters]

- **enable**: (mandatory) Specifies whether to enable stream statistics. The value is of the Boolean type. The value **true** indicates that stream statistics are enabled.

- **interval**: (mandatory) Specifies the statistics interval, in seconds. The value is a number ranging from 1 to 60. The default value is **1**.

[Response Parameters]

None

enableStreamStateDetection

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

[Function Description]

Enables or disables media stream status detection. After the function is enabled, the system can detect whether the stream has been interrupted at the stream push device.

[Request Parameters]

- **enable**: (mandatory) whether to enable media stream status detection. The value is of the Boolean type. **true** indicates enabled; **false** (default value) indicates disabled.
- **interval**: (mandatory) Specifies the interval in seconds. The value is a number ranging from 1 to 60. This parameter is used to determine when there is no media stream. The default value is **3** (recommended).
- **interruptRetry**: (optional) Parameter for configuring playback retry upon stream interruption. The value is of StreamInterruptRetry type. The definition of StreamInterruptRetry is as follows: {

  enable: The value is of the Boolean type. Attempt for automatically resuming playback is enabled after stream interruption. The default value is **false**, indicating that attempt for automatically resuming playback is disabled.

  retryInterval: retry interval for stream pulling, in seconds. The value type is number. The value ranges from 10 to 60 and defaults to **30**.

  retryTimes: maximum number of retry times for resuming playback. The value type is number. The minimum value is **1** and the default value is **30**.

  }

[Response Parameters]

boolean: whether the operation is successful. The options are **true** (success) and **false** (failure).

on

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

[Function Description]

Registers the callback for a client object event.

[Request Parameters]

- **event**: (mandatory) event name. The type is string. For details, see **HWLLSClientEvent**.
- **handler**: (mandatory) event processing method. The type is function.

- **withTimeout**: (optional) indicates whether a timeout error is reported. The value is of the Boolean type.

[Response Parameters]

None

off

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

[Function Description]

Deregisters the callback for a client object event.

[Request Parameters]

- **event**: (mandatory) event name. The type is string. For details, see **HWLLSClientEvent**.
- **handler**: (mandatory) event processing method. The type is function.

[Response Parameters]

None

destoryClient

```
destoryClient(): void
```

[Function Description]

Destroys a client object.

[Request Parameters]

None

[Response Parameters]

None

fullScreenToggle

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

[Function Description]

Enables/Disables full-screen display.

[Request Parameters]

isExit: (mandatory) Boolean type. The default value is **false**.

[Response Parameters]

None

3.5.3 Client Object (HWFlvClient)

This section describes the HWFlvClient APIs of the LLL Web SDK.

Table 3-5 HRTC APIs

API	Description
startPlay	Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.
switchPlay	Quickly switches to the next stream.
stopPlay	Stops the playback.
replay	Replays the video.
resume	Resumes the playback.
pause	Pauses the playback.
pauseVideo	Pauses the video playback.
resumeVideo	Resumes the video playback.
pauseAudio	Pauses the audio playback.
resumeAudio	Resumes the audio playback.
setPlayoutVolume	Sets the playback volume.
getPlayoutVolume	Obtains the audio volume.
muteAudio	Mutes.
streamStatistic	Specifies whether to enable stream statistics.
enableStreamStateDetection	Enables or disables media stream status detection.
destoryClient	Destroys a client object.
fullScreenToggle	Enables/Disables full-screen display.

startPlay

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

[Function Description]

Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.

[Request Parameters]

- **url**: (mandatory) String type. Ingest URL that ends with **flv**.
- **options**: (optional) StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.

- **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
  - **contain**: prioritizes the display of all video content. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
  - **cover**: prioritizes the filling of the window. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
  - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
- **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
- **sessionId**: This parameter does not need to be transferred.
- **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
- **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
- **poster**: (optional) The object definition is as follows: {
  - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
  - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }
  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.

    - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

    }
  - **webrtcConfig**: This parameter does not need to be transferred.
  - **schedulePolicy**: This parameter does not need to be transferred.
  - **domainPolicy**: This parameter does not need to be transferred.
  - **downgradeUrl**: This parameter does not need to be transferred.

[Response Parameters]

Promise<void>: returns a **Promise** object.

switchPlay

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

[Function Description]

After the playback is started, the system quickly switches to the next stream.

[Request Parameters]

- **url**: (mandatory) String type. Ingest URL that ends with **flv**.
- **options**: (optional) StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: prioritizes the display of all video content. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: prioritizes the filling of the window. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
  - **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
  - **sessionId**: This parameter does not need to be transferred.
  - **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**,

the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.

- **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
- **poster**: (optional) The object definition is as follows: {
  - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
  - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }
  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
  - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

  }
- **webrtcConfig**: This parameter does not need to be transferred.
- **schedulePolicy**: This parameter does not need to be transferred.
- **domainPolicy**: This parameter does not need to be transferred.
- **downgradeUrl**: This parameter does not need to be transferred.

[Response Parameters]

Promise<void>: returns a **Promise** object.

stopPlay

```
stopPlay(): boolean
```

[Function Description]

Stops the playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of stopping playback. The options are **true** (success) and **false** (failure).

replay

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

[Function Description]

Replays the video.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of replay. The options are **true** (success) and **false** (failure).

resume

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

[Function Description]

Resumes the playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio/video playback. The options are **true** (success) and **false** (failure).

pause

```
pause(): boolean
```

[Function Description]

Pauses the audio/video playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing playback. The options are **true** (success) and **false** (failure).

pauseVideo

```
pauseVideo(): boolean
```

[Function Description]

This API is not supported.

[Request Parameters]

None

[Response Parameters]

boolean: Only **false** is returned.

resumeVideo

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

[Function Description]

This API is not supported.

[Request Parameters]

None

[Response Parameters]

boolean: Only **false** is returned.

pauseAudio

```
pauseAudio(): boolean
```

[Function Description]

Pauses the audio playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing audio playback. The options are **true** (success) and **false** (failure).

resumeAudio

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

[Function Description]

Resumes the audio playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio playback. The options are **true** (success) and **false** (failure).

setPlayoutVolume

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

[Function Description]

If you set the audio volume, sound will be heard.

[Request Parameters]

volume: (mandatory) audio volume. The value is a number ranging from 0 to 100.

[Response Parameters]

boolean: whether the audio volume has been set. The options are **true** (success) and **false** (failure).

getPlayoutVolume

```
getPlayoutVolume(): number
```

[Function Description]

Obtains the audio volume.

[Request Parameters]

None

[Response Parameters]

number: volume value. The value ranges from 0 to 100.

muteAudio

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

[Function Description]

Mutes.

[Request Parameters]

isMute: (mandatory) whether to mute. The value is of the Boolean type. **true** indicates muting, and **false** indicates unmuting.

[Response Parameters]

None

streamStatistic

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

[Function Description]

Specifies whether to enable stream statistics.

[Request Parameters]

- **enable**: (mandatory) Specifies whether to enable stream statistics. The value is of the Boolean type. The value **true** indicates that stream statistics are enabled.
- **interval**: (mandatory) Specifies the statistics interval, in seconds. The value is a number ranging from 1 to 60. The default value is **1**.

[Response Parameters]

None

enableStreamStateDetection

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

[Function Description]

Enables or disables media stream status detection. After the function is enabled, the system can detect whether the stream has been interrupted at the stream push device.

[Request Parameters]

- **enable**: (mandatory) whether to enable media stream status detection. The value is of the Boolean type. **true** indicates enabled; **false** (default value) indicates disabled.
- **interval**: (mandatory) Specifies the interval in seconds. The value is a number ranging from 1 to 60. This parameter is used to determine when there is no media stream. The default value is **3** (recommended).
- **interruptRetry**: (optional) Parameter for configuring playback retry upon stream interruption. The value is of the StreamInterruptRetry type. The definition of StreamInterruptRetry is as follows: {

  enable: The value is of the Boolean type, indicating that attempt for automatically resuming playback is enabled after stream interruption. The default value is **false**, indicating that attempt for automatically resuming playback is disabled.

  retryInterval: retry interval for stream pull, in seconds. The value type is number. The value ranges from 10 to 60 and defaults to **30**.

  retryTimes: maximum number of retry times for resuming playback. The value type is number. The minimum value is **1** and the default value is **30**.

  }

[Response Parameters]

boolean: whether the operation is successful. The options are **true** (success) and **false** (failure).

destoryClient

```
destoryClient(): void
```

[Function Description]

Destroys a client object.

[Request Parameters]

None

[Response Parameters]

None

fullScreenToggle

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

[Function Description]

Enables/Disables full-screen display.

[Request Parameters]

isExit: (mandatory) Boolean type. The default value is **false**.

[Response Parameters]

None

3.5.4 Client Object (HWHlsClient)

This section describes the HWHlsClient APIs of the LLL Web SDK.

Table 3-6 HRTC APIs

API	Description
startPlay	Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.
switchPlay	Quickly switches to the next stream.
stopPlay	Stops the playback.
replay	Replays the video.
resume	Resumes the playback.
pause	Pauses the playback.
pauseVideo	Pauses the video playback.
resumeVideo	Resumes the video playback.
pauseAudio	Pauses the audio playback.
resumeAudio	Resumes the audio playback.
setPlayoutVolume	Sets the playback volume.
getPlayoutVolume	Obtains the audio volume.
muteAudio	Mutes.
streamStatistic	Specifies whether to enable stream statistics.
enableStreamStateDetection	Enables or disables media stream status detection.
destoryClient	Destroys a client object.
fullScreenToggle	Enables/Disables full-screen display.

startPlay

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


[Function Description]

Starts playback. The client obtains the corresponding live stream from the server based on the entered URL.

[Request Parameters]

- **url**: (mandatory) String type. Ingest URL that ends with **m3u8**.
- **options**: (optional) StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: prioritizes the display of all video content. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: prioritizes the filling of the window. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream will be cropped or the image will be stretched to fill the display window.
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
  - **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
  - **sessionId**: This parameter does not need to be transferred.
  - **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
  - **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
  - **poster**: (optional) The object definition is as follows: {
    - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.
    - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {

    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }

  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.
  - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

  }
- **webrtcConfig**: This parameter does not need to be transferred.
- **schedulePolicy**: This parameter does not need to be transferred.
- **domainPolicy**: This parameter does not need to be transferred.
- **downgradeUrl**: This parameter does not need to be transferred.

[Response Parameters]

Promise<void>: returns a **Promise** object.

switchPlay

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

[Function Description]

After the playback is started, the system quickly switches to the next stream.

[Request Parameters]

- **url**: (mandatory) String type. Ingest URL that ends with **m3u8**.
- **options**: (optional) StartPlayOptions type. If this parameter is not carried, the **options** data carried in the first playback start request is reused. The definition of StartPlayOptions is as follows: {
  - **elementId**: (mandatory) indicates the playback DOM ID.
  - **objectFit**: (optional) String type. Default value: **cover**. The following enumerated values are supported:
    - **contain**: prioritizes the display of all video content. The video is scaled proportionally until one side of the video window is aligned with the window border. If the video size is inconsistent with the display window size, when the aspect ratio is locked and the video is zoomed in or out to fill the window, a black bar is displayed around the zoomed-in or zoomed-out video.
    - **cover**: prioritizes the filling of the window. The video is scaled proportionally until the entire window is filled with video. If the video size is inconsistent with the display window size, the video stream

will be cropped or the image will be stretched to fill the display window.

  - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.

- **muted**: (optional) Boolean type. **true** indicates muted; **false** indicates unmuted. The default value is **false**.
- **sessionId**: This parameter does not need to be transferred.
- **showLoading**: (optional) Boolean type. Indicates whether to enable the loading display effect. **true** indicates that the loading display effect is enabled. The default value is **false**. When this parameter is set to **true**, the loading effect upon playback start will also be enabled. The loading effect upon buffering during playback needs to be set based on the **LOADING_CONFIG** in the **setParameter** API.
- **autoPlay**: (optional) Boolean type. **true** indicates that the autoplay function is enabled. **false** indicates that the autoplay function needs to be manually triggered. The default value is **true**.
- **poster**: (optional) The object definition is as follows: {

  - **url**: (optional) String type. Sets the complete address of the thumbnail image to be played. The image must be in JPG, PNG, or static GIF format, the size cannot exceed 1 MB, and the resolution cannot exceed 1920 x 1080. The file name cannot contain Chinese characters.

  - **mode**: (optional) String type. The default value is **cover**. The following enumerated values are supported: {
    - **fill**: The window is filled with video. If the aspect ratio of the video does not match the window, the video will be stretched to fit the window.
    - **crop**: original size of the thumbnail. If the poster exceeds the playback area, the excess part is cropped. Otherwise, the poster is displayed in the middle of the playback window.

    }

  - **startEnable**: (optional) Boolean type. Indicates whether to display the thumbnail when the playback starts. The options are **true** and **false**. The default value is **false**. This parameter takes effect only in non-autoplay scenarios.

  - **pauseEnable**: (optional) Boolean type. Indicates whether to display the thumbnail on the playback page when the video is paused. The options are **true** and **false**. The default value is **false**.

  }
- **webrtcConfig**: This parameter does not need to be transferred.
- **schedulePolicy**: This parameter does not need to be transferred.
- **domainPolicy**: This parameter does not need to be transferred.
- **downgradeUrl**: This parameter does not need to be transferred.

[Response Parameters]

Promise<void>: returns a **Promise** object.

stopPlay

```
stopPlay(): boolean
```

[Function Description]

Stops the playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of stopping playback. The options are **true** (success) and **false** (failure).

replay

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

[Function Description]

Replays the video.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of replay. The options are **true** (success) and **false** (failure).

resume

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

[Function Description]

Resumes the playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio/video playback. The options are **true** (success) and **false** (failure).

pause

```
pause(): boolean
```

[Function Description]

Pauses the audio/video playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing playback. The options are **true** (success) and **false** (failure).

pauseVideo

```
pauseVideo(): boolean
```

[Function Description]

This API is not supported.

[Request Parameters]

None

[Response Parameters]

boolean: Only **false** is returned.

resumeVideo

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

[Function Description]

This API is not supported.

[Request Parameters]

None

[Response Parameters]

boolean: Only **false** is returned.

pauseAudio

```
pauseAudio(): boolean
```

[Function Description]

Pauses the audio playback.

[Request Parameters]

None

[Response Parameters]

boolean: result of pausing audio playback. The options are **true** (success) and **false** (failure).

resumeAudio

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

[Function Description]

Resumes the audio playback.

[Request Parameters]

None

[Response Parameters]

Promise<boolean>: result of resuming the audio playback. The options are **true** (success) and **false** (failure).

setPlayoutVolume

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

[Function Description]

If you set the audio volume, sound will be heard.

[Request Parameters]

volume: (mandatory) audio volume. The value is a number ranging from 0 to 100.

[Response Parameters]

boolean: whether the audio volume has been set. The options are **true** (success) and **false** (failure).

getPlayoutVolume

```
getPlayoutVolume(): number
```

[Function Description]

Obtains the audio volume.

[Request Parameters]

None

[Response Parameters]

number: volume value. The value ranges from 0 to 100.

muteAudio

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

[Function Description]

Mutes.

[Request Parameters]

isMute: (mandatory) whether to mute. The value is of the Boolean type. **true** indicates muting, and **false** indicates unmuting.

[Response Parameters]

None

streamStatistic

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

[Function Description]

Specifies whether to enable stream statistics.

[Request Parameters]

- **enable**: (mandatory) Specifies whether to enable stream statistics. The value is of the Boolean type. The value **true** indicates that stream statistics are enabled.
- **interval**: (mandatory) Specifies the statistics interval, in seconds. The value is a number ranging from 1 to 60. The default value is **1**.

[Response Parameters]

None

enableStreamStateDetection

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

[Function Description]

Enables or disables media stream status detection. After the function is enabled, the system can detect whether the stream has been interrupted at the stream push device.

[Request Parameters]

- **enable**: (mandatory) whether to enable media stream status detection. The value is of the Boolean type. **true** indicates enabled; **false** (default value) indicates disabled.
- **interval**: (mandatory) Specifies the interval in seconds. The value is a number ranging from 1 to 60. This parameter is used to determine when there is no media stream. The default value is **3** (recommended).
- **interruptRetry**: (optional) Parameter for configuring playback retry upon stream interruption. The value is of the StreamInterruptRetry type. The definition of StreamInterruptRetry is as follows: {

  enable: The value is of the Boolean type, indicating that attempt for automatically resuming playback is enabled after stream interruption. The default value is **false**, indicating that attempt for automatically resuming playback is disabled.

  retryInterval: retry interval for stream pull, in seconds. The value type is number. The value ranges from 10 to 60 and defaults to **30**.

  retryTimes: maximum number of retry times for resuming playback. The value type is number. The minimum value is **1** and the default value is **30**.

  }

[Response Parameters]

boolean: whether the operation is successful. The options are **true** (success) and **false** (failure).

destoryClient

```
destoryClient(): void
```

[Function Description]

Destroys a client object.

[Request Parameters]

None

[Response Parameters]

None

fullScreenToggle

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

[Function Description]

This API is not supported.

[Request Parameters]

isExit: (mandatory) Boolean type. The default value is **false**.

[Response Parameters]

None

3.5.5 Client Event Notification (HWLLSClientEvent)

This section describes the HWLLSClientEvent APIs of the LLL Web SDK.

Table 3-7 HWLLSClientEvent APIs

API	Description
media-statistic	Media statistics event.
network-quality	Network quality report event.
video-broken	Video stream interruption event.
audio-broken	Audio stream interruption event.
video-recovery	Video streaming resumption event.
audio-recovery	Audio streaming resumption event.
audio-start	Audio playback start event.
video-start	Video playback start event.
video-stuck	Video playback pause event.
fullscreen-status-changed	Full-screen view event.
player-changed	Playback downgrade event.
Error	Client error event.

> **CAUTION**
>
> Registration listening must be canceled when the service ends. Otherwise, memory leakage may occur when there are a certain number of registration listening events.

media-statistic

[Event Description]

Media statistics event. This event is used together with the **streamStatistic** method.

[Callback Parameters]

StatisticInfo: media statistics. The value type is StatisticInfo.

StatisticInfo is defined as follows: {

- video: {
  mediaType: media type
  frameRate: video frame rate. The type is number.
  width: video width. The type is number.
  height: video height. The type is number.
  jitter: jitter. The type is number.
  bitRate: bitrate (in kbit/s). The type is number.
  bytesReceived: number of received bytes. The type is number.
  packetsReceived: number of received packets. The type is number.
  packetsLost: number of lost packets. The type is number.
  }
- audio: {
  mediaType: media type
  jitter: jitter. The type is number.
  bitRate: bitrate (in kbit/s). The type is number.
  bytesReceived: number of received bytes. The type is number.
  packetsReceived: number of received packets. The type is number.
  packetsLost: number of lost packets. The type is number.
  }

}

network-quality

[Event Description]

Network quality report event.

[Callback Parameters]

NetworkQualityTypes: network quality details. The type is NetworkQualityTypes.

The enumerated values of **NetworkQualityTypes** are as follows:

- **NETWORK_QUALITY_UNKNOW = 0**: The network quality is unknown.
- **NETWORK_QUALITY_GREAT = 1**: The network quality is excellent.
- **NETWORK_QUALITY_GOOD = 2**: User experience is almost the same as that of value **1**, but the bitrate may be slightly lower.
- **NETWORK_QUALITY_DEFECTS = 3**: User experience is defective but the watching is not affected.
- **NETWORK_QUALITY_WEAK = 4**: The network quality is poor and the video is not smooth.
- **NETWORK_QUALITY_BAD = 5**: The network quality is so poor that user experience is severely affected.
- **NETWORK_QUALITY_DISCONNECT = 6**: The network quality is poor and even disconnection occurs. The video cannot be watched.

video-broken

[Event Description]

Video stream interruption event.

[Callback Parameters]

None

audio-broken

[Event Description]

Audio stream interruption event.

[Callback Parameters]

None

video-recovery

[Event Description]

Video streaming (non-EOF) resumption event.

[Callback Parameters]

None

audio-recovery

[Event Description]

Audio streaming (non-EOF) resumption event.

[Callback Parameters]

None

audio-start

[Event Description]

Audio playback start event.

[Callback Parameters]

None

video-start

[Event Description]

Video playback start event.

[Callback Parameters]

None

video-stuck

[Event Description]

Video playback pause event.

[Callback Parameters]

Boolean value. **True** indicates paused, and **False** indicates not paused.

fullscreen-status-changed

[Event Description]

Full-screen view event.

[Callback Parameters]

- **isFullScreen**: indicates whether to enable full-screen display
- **isPause**: indicates whether to stop playback

player-changed

[Event Description]

Playback downgrade event.

[Callback Parameters]

Indicates the downgrade information. The value is a string.

- **webrtc**: LLL playback
- **hls**: HLS playback
- **flv**: FLV playback

Error

[Event Description]

This event is triggered when an unrecoverable client error occurs.

[Callback Parameters]

errorInfo: (mandatory) error information. The type is ErrorInfo.

ErrorInfo is defined as: {

errorCode: (mandatory) error code. The type is string.

errorMsg: (mandatory) error description. The type is string.

}

> ⚠ CAUTION
>
> If the network firewall is restricted (UDP port restriction) or playback fails on LLL for multiple times, you can downgrade the playback based on the specified error code (**HWLLS_MEDIA_NETWORK_ERROR** or **HWLLS_PLAY_WEBRTC_RETRY_FAILED**). For details, see **SDK Usage**.

3.5.6 Error Codes

getCode

```
getCode(): number
```

[Function Description]

Obtains an error code.

[Request Parameters]

None

[Response Parameters]

Error code value, which is a number.

getMsg

```
getMsg(): string
```

[Function Description]

Obtains error description.

[Request Parameters]

None

[Response Parameters]

Error code description, which is a string.

3.5.7 Public IP Addresses

Table 3-8 Public IP address list

Public IP Address	Information
log-collection-new.hwcloudlive.com	Log and dotting environment address in China
log-collection-ap-southeast-3.rocket-cdn.com	Log and dotting environment address outside China
global-lll.huaweicloud.com	Default primary environment address for stream pull
global-lll.huaweicloud.cn	Default standby environment address for stream pull
hcdnl-pull302-global-gslb.livehwc3.cn	Default GSLB environment address
dns.alidns.com	Domain Name Service (DNS) address provided by Alibaba Cloud
doh.pub	DNS address provided by Tencent Cloud
dns.google	DNS provided by Google
cloudflare-dns.com	DNS provided by Cloudflare

3.5.8 Client Error Codes

This section describes details about the error codes reported on the LLL Web client SDK.

Table 3-9 Error code description

Class Member	Error Code	Description	Error Cause or Handling Suggestion
HWLLS_OK	0	Succeeded.	-
HWLLS_ERROR_INVALID_URL	50000000	Invalid URL.	Check whether the URL is correct.
HWLLS_ERROR_INVALID_PARAMETER	50000001	Invalid parameter.	Parameter transfer error. Check whether the input parameters of the API meet the parameter validity requirements.

Class Member	Error Code	Description	Error Cause or Handling Suggestion
HWLLS_ERROR_SERVER_CONNECT_FAIL	50000002	Connect to the server failed.	Check whether the network status is normal or contact Huawei technical support.
HWLLS_ERROR_SERVER_NO_RESPONSE	50000003	The server does not respond.	Contact Huawei technical support.
HWLLS_ERROR_AUTH_FAIL	50000004	Authentication failed.	Check whether the referer validation and URL validation configurations on the server are correct.
HWLLS_ERROR_STREAM_NOT_EXIST	50000005	The requested stream does not exist.	Use a stream available.
HWLLS_ERROR_WEBRTC_UNSUPPORTED	50000006	Your browser does not support the function.	1. Use a supported browser. For details, see **Browser Adaptation**. 2. Downgrade the FLV or HLS livestreaming.
HWLLS_MEDIA_NETWORK_ERROR	50000007	Abnormal media network connection	Check whether the network status and firewall configuration are correct, or downgrade the FLV or HLS livestreaming.
HWLLS_ERROR_STREAM_INVALID_PARAMETER	50000008	Incorrect stream information	Check whether the URL of the requested stream is correct.
HWLLS_ERROR_BANDWIDTH_OVER_MAXIUM	50000009	The bandwidth exceeds the upper limit.	Contact Huawei technical support.
HWLLS_INTERNAL_ERROR	50000020	Other internal errors	Contact Huawei technical support.
HWLLS_BUSSINESS_DOWNGRADE	50000021	The service needs to be downgraded.	You are advised to downgrade the FLV or HLS livestreaming.
HWLLS_PLAY_WEBRTC_RETRY_FAILED	50000022	After the LLL playback is interrupted, multiple attempts for resuming playback failed.	You are advised to downgrade the FLV or HLS livestreaming.

Class Member	Error Code	Description	Error Cause or Handling Suggestion
HWLLS_PLAY_FLV_RETRY_FAILED	50000023	After the FLV playback is interrupted, multiple attempts for resuming playback failed.	Contact Huawei technical support.
HWLLS_PLAY_HLS_RETRY_FAILED	50000024	After the HLS playback is interrupted, multiple attempts for resuming playback failed.	Contact Huawei technical support.
HWLLS_ERROR_LIVE_UNSUPPORTED	50000030	The content format is not supported by the browser.	Contact Huawei technical support.
HWLLS_ERROR_UNEXPECTED_EOF	50000031	The network EOF is abnormal during content playback.	Try again.
HWLLS_ERROR_MEDIA_ERROR	50000032	The media content playback is abnormal.	Try again or contact Huawei technical support.
HWLLS_ERROR_REPORT_TOKEN_ERROR	50000033	Token exceptions occur during dotting and log uploading.	Contact Huawei technical support.
HWLLS_PLAY_NOT_ALLOW	51000000	The playback permission is restricted. You need to manually trigger the playback.	Due to the restrictions of the browser's automatic playback security policy, this error code is returned when the browser directly starts the app and playback starts. According to this error code, you need to manually trigger the UI control on the page at the application layer and call the **replay** API to resume playback. **NOTICE** If Safari is used, perform the following operations: Open Safari, choose **Preferences** > **Websites** > **Auto-Play**, select a website and enable **Allow All Auto-Play** for it.

Class Member	Error Code	Description	Error Cause or Handling Suggestion
HWLLS_PLAY_TIMEOUT	51000001	Playback times out. No valid frame data is obtained within 10 seconds.	Check the stream push status or contact Huawei technical support.

3.6 FAQs

- **Can the Huawei Cloud LLL Web SDK be integrated if the service app can use only the HTTP?**

  The SDK can be integrated with some browsers such as Chrome but integration is not recommended. The browser compatibility is identified based on the WebRTC object exposed by the browser. When a non-HTTPS protocol is used, the object may not exist.

- **Why can't I use the Huawei Cloud LLL Web SDK in Firefox?**

  Before using the Firefox browser, you need to install the H.264 codec plug-in. Enter **about:addons** in the address box of the browser. The plug-in installation page is displayed. Check whether the H.264 plug-in has been installed. If not, install it on the page.

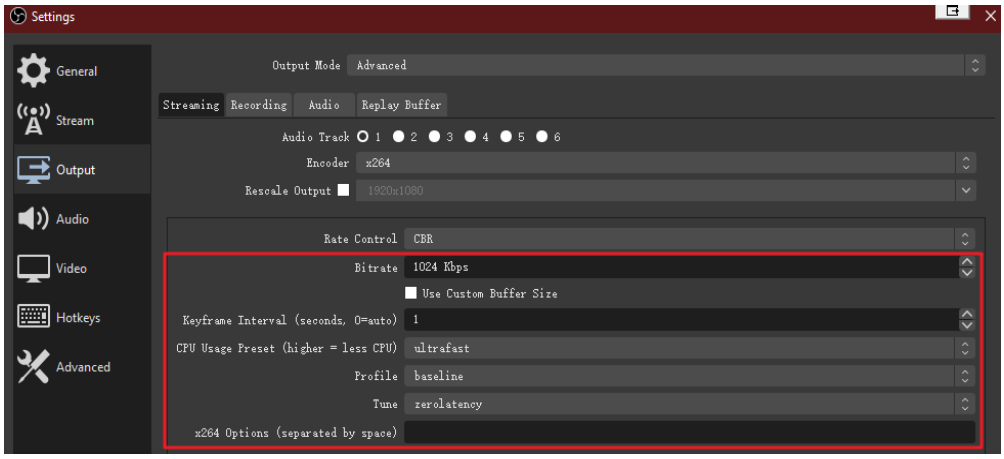
- **What are the possible causes if the Huawei Cloud LLL Web SDK does not work after being integrated?**
  - Check whether the user-defined domain names have been configured, such as ingest and streaming domain names, and check whether the HTTPS certificate is valid.
  - Check whether the stream push end and stream is normal.
  - Check whether the streaming URL is correct, for example, appName and streamName.
  - Check whether the network connection is normal and whether the network firewall configuration is restricted. For example, check whether UDP ports 8000 to 8063 are bypassed.

- **Which browsers are supported by the Huawei Cloud LLL Web SDK?**

  For details about supported browsers, see **Browser Adaptation**.

- **Why does stream pull on the Huawei Cloud LLL Web SDK fail after receiving pushed streams?**

  Check whether the stream push encoding parameter of the stream push device is H.264+ without B-frames. Currently, the Huawei Cloud LLL Web SDK supports only H.264+ streams without B-frames. As a result, if the original stream is H.265 or contains B-frames, you need to configure the corresponding transcoding template on the tenant console in advance and enable the transcoding service. However, this will introduce extra transcoding delay and fees. You are advised to push H.264+ streams without B-frames. You can adjust the video encoding parameters of the streaming software (such as OBS) to remove B-frames. If OBS is used to push streams, you can disable B-frames, as shown in the following figure.



- **What do I do if the error message "NotAllowedError:xxx?" is displayed during playback using the Huawei Cloud LLL Web SDK?**

  Due to the restrictions of the browser's automatic playback security policy, this error code is returned when the browser directly starts the app and playback starts. According to this error code, you need to manually trigger the UI control on the page at the application layer and call the **replay** API to resume playback.

3.7 Change History

Table 3-10 Change history

Released On	Change Description
2024-03-19	1. Added the SDK package download path and integrity verification method. 2. Added the description of FLV and HLS.
2023-10-30	This issue is the first official release.

4 Change History

Released On	Change Description
2023-10-30	This issue is the first official release.

