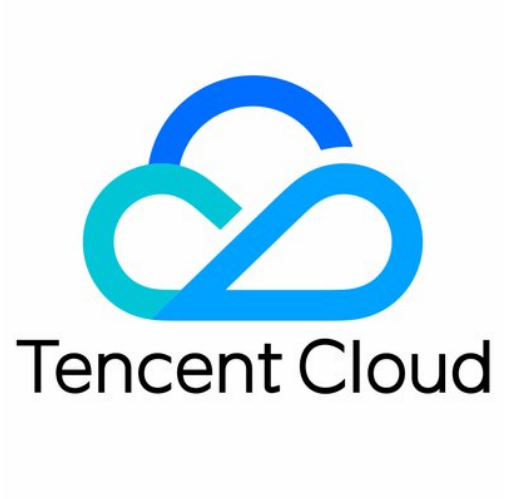
Cloud Streaming Services

Playing Method

Product Documentation



Contents

Playing Method
- Web Player
  - Web Player TcPlayer
    - FAQs
  - Web CSS Player 1.0
    - Troubleshooting
  - Web VOD Player 1.0
    - Web VOD Player 1.0
    - Troubleshooting

Playing Method
Web Player
Web Player TcPlayer
FAQs

Last updated：2023-10-30 10:45:21

Which browsers are supported by TcPlayer?

TcPlayer has been tested on and supports the following browsers: PC: IE 10+, Edge, Chrome, Firefox, QQ Browser, MAC Safari. Mobile: Android 4.4+, iOS 8.0+, WeChat, Mobile QQ, QQ Browser, Chrome, Safari.
To support IE 8 and IE 9, you must introduce the Polyfill script before introducing the player script, as shown below:

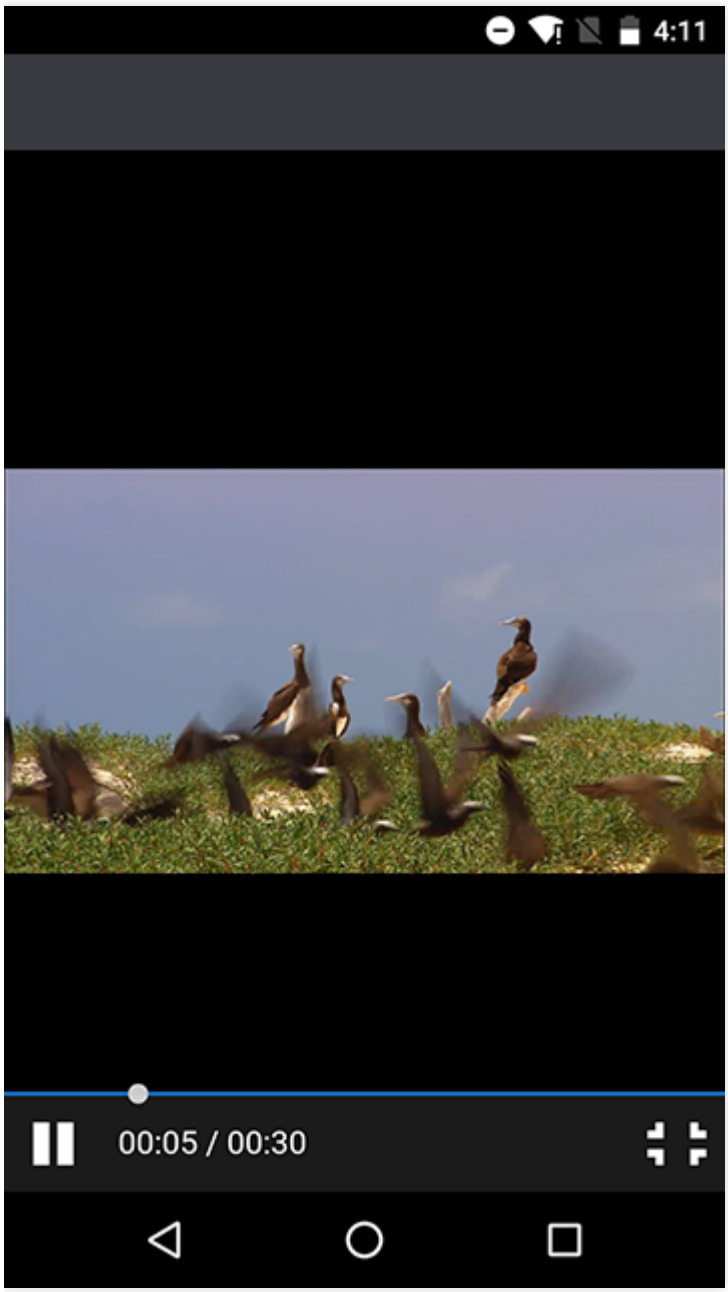
```
<!--[if lt IE 9]>
<script src="//imgcache.qq.com/open/qcloud/video/vcplayer/libs/es5-shim.js" chars
et="utf-8"></script>
<script src="//imgcache.qq.com/open/qcloud/video/vcplayer/libs/es5-sham.js" chars
et="utf-8"></script>
<![endif]-->
<script src="//imgcache.qq.com/open/qcloud/video/vcplayer/TcPlayer-2.2.1.js" char
set="utf-8"></script>;
```

On mobile devices, the video does not go into full screen when TcPlayer is switched to full screen mode, and browser interface is still displayed. Why is that?

First, you should know that the full screen solution provided by TcPlayer is to use Fullscreen API + pseudo-full screen. If the browser supports Fullscreen API, the video container will fill up the entire screen when the player goes into full screen mode, and the control bar is provided by TcPlayer, as shown in the figure:

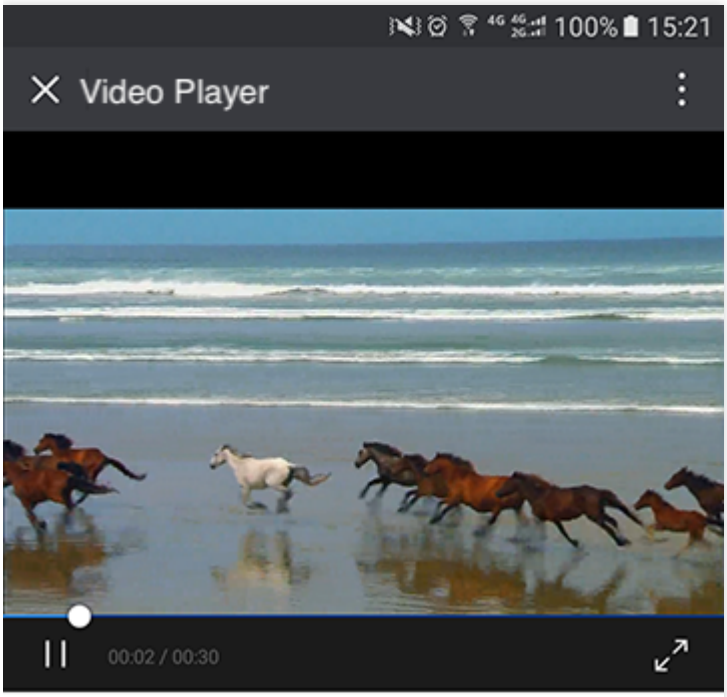
(Android Chrome)

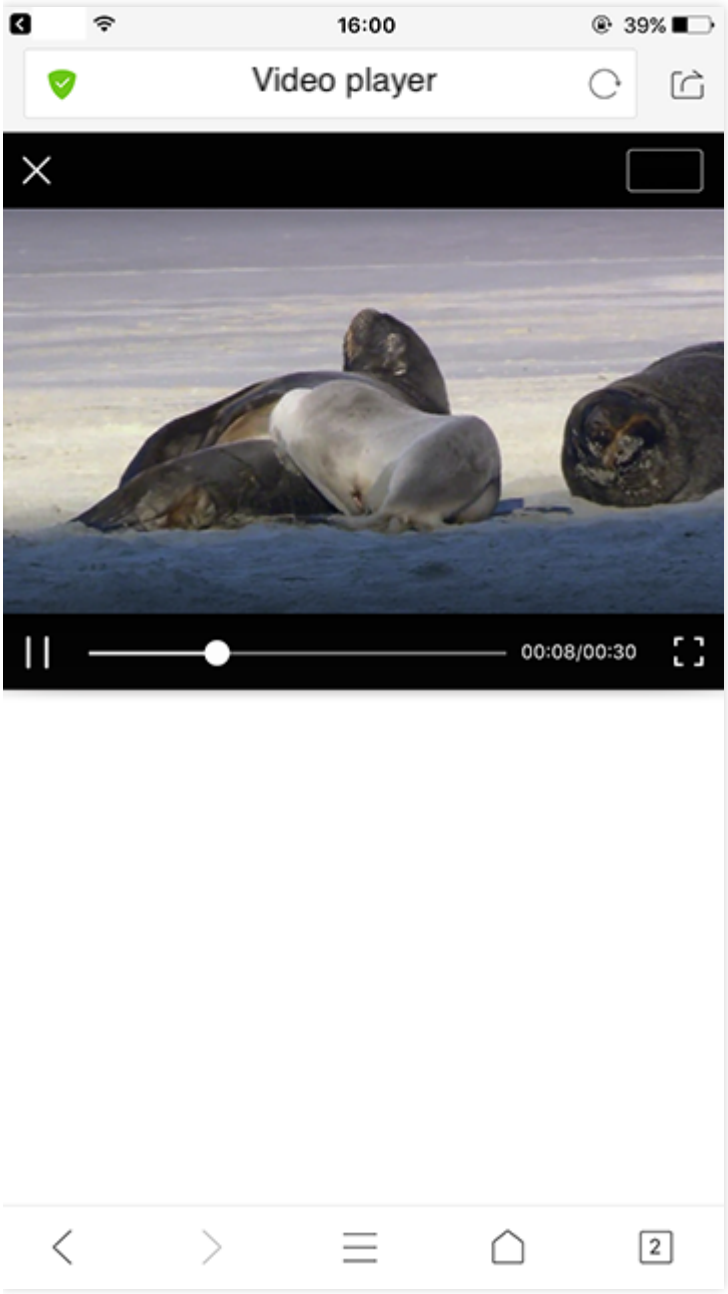
If the browser does not support Fullscreen API, then pseudo-full screen mode will be used, as shown in the figure:



(Left: WeChat on Android. Right: WeChat on iOS)

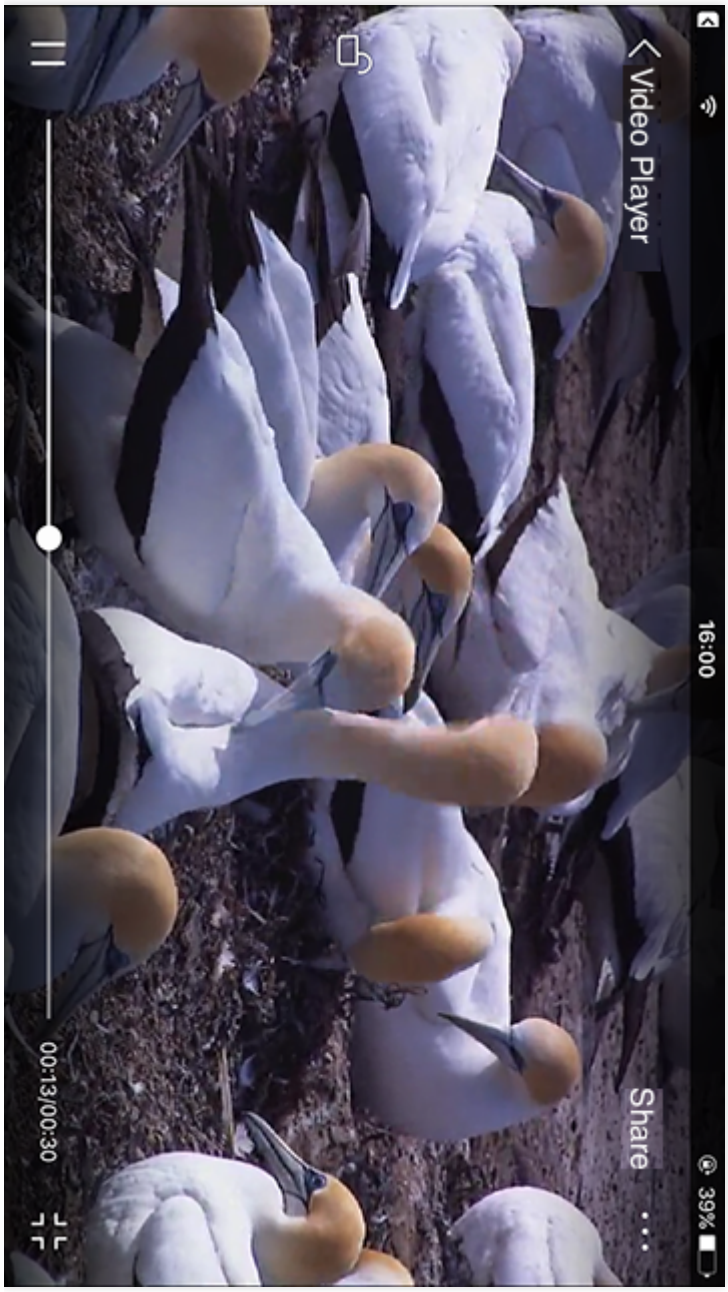
The control bars displayed in these two full screen modes are all provided by TcPlayer, and you enter these modes by tapping on the full screen button on the control bar or by using the method provided by TcPlayer. However, the control bar provided by TcPlayer may not always be displayed on mobile devices, as their webview hijacks the video playback in most situations and uses the control bar provided by webview. In this case, the TcPlayer control bar will not be displayed, and you cannot use the full screen solution provided by TcPlayer, as shown in the figure:





(Left: WeChat on Android, where video playback is hijacked by TBS. Right: iOS, where video playback is hijacked by QQ Browser)

Upon entering full screen mode:



(Left: WeChat on Android. Right: QQ Browser on iOS)

As you can see, the control bar is different from the TcPlayer one. No APIs are provided for JS in this full screen mode, so TcPlayer cannot realize this mode. We usually refer to the full screen mode where the video covers the whole screen as System Full Screen, and the mode where the video covers the whole browser page area as Pseudo-Full Screen. Therefore, if you can see the browser interface after going into full screen mode, then it is pseudo-full screen. You can only use the control bar provided by the system if you want to go into system full screen mode on mobile devices. You may choose the control bar type by using the "controls" attribute of TcPlayer.

Why is the screen stretched when the video is played in H5?

Video stretching is not supported when playing video in H5. Check if the player container width/height configuration is correct.

Why can't I cover my own div on top of the video?

Different browsers implement the `<video>` tag in different ways. For example, if you open a web page in QQ or WeChat (on Android system), then it's very likely that X5 browser kernel is used (the kernel which is bound with QQ or WeChat, i.e. the QQ Browser kernel). For certain reasons, the QQ Browser team implemented such a solution that "the video `<video>` must be placed in the top layer" (for more information, please see QQ Browser Documentation). However, with recent coordination among teams within the company, the QQ Browser Team is gradually changing this policy. This issue may have already been solved in the latest X5 browser kernel as you read this document.

Why is the configured cover image not working?

This is the same problem with the previous one ("cannot cover div on top of the video"). The cover image cannot be displayed as long as the browser does not allow elements to cover up the `<video>` tag.

Why is video displayed in full screen mode by default in certain mobile browsers?

If the video is played through inline playback inside the application (that is, your own application encapsulates an iOS webview control which is used to display web pages. In this mode, you can customize the details of webview, which may have different performance from standard Safari browser), you can configure the "webkit-playsinline" attribute for the `<video>` tag in HTML (or configure the "playsinline" attribute if you're using iOS 10), and then configure "allowsInlineMediaPlayback" for webview. In this way, videos are played in non-full screen mode (inline playback) when you open pages in the application.
If you open the page in Safari on iOS 10, you can realize non-full screen playback (inline playback) by using the method above (configure "playsinline" attribute for the `<video>` tag). You cannot disable auto full screen playback for Safari on systems below iOS 10. This attribute is already added for our browser. It only needs to be supported by the devices.
For Android devices, it is known that there are many different customized versions for Android systems, and each system realizes the `<video>` tag in a different way, without a universal standard. For this reason, video playback capabilities on Android have much lower consistency compared to iOS. The "webkit-playsinline playsinline" attribute is already added for the player according to current universal method. It only needs to be supported by the devices.

Why can't I realize auto video playback in mobile browsers?

There are only two ways to realize auto video playback on mobile web: by configuring the "autoplay" attribute for the `<video>` tag, or by calling the "play()" method provided by the `<video>` tag. However, auto video playback has always been prohibited on mobile web, thus the universal method now is to trigger video playback by users actions. For example, monitor ontap events of users and call the "play()" method. It is possible that certain customized webviews support the "autoplay" attribute of the `<video>` tag, or can realize auto video playback by calling other

special functions. Videos can be played automatically when you open the pages in these webviews. Our player will add the "autoplay" attribute for the `<video>` tag if "autoplay" is configured as "true". Now we only need the devices to support this attribute.

Why does the Flash player have two play buttons in Chrome on PC?

Flash is no longer be automatically played starting from Chrome 42 (Google has purchased WebRTC and made it open-source for a reason). Chrome only plays major Flash contents automatically, while other Flash contents are paused, unless users enable them manually.

Why can the CSS video be played in browsers on PC but cannot on mobile devices?

Only HLS (M3U8) protocol is supported for playing live streaming videos in mobile browsers. Thus, you need to confirm whether the live stream pulling addresses contain URL for pulling HLS (M3U8) stream. The video cannot be played on mobile phones if you only provide an flv or rtmp address for our player.

Web CSS Player 1.0 Troubleshooting

Last updated：2022-03-21 12:15:19

Error Codes

List of error codes during the use of SDK

- Error messages returned from the frontend

Message	Description
An error occurred while parsing video information. Check whether the parameter is correct.	No JSON data is returned from API. Data cannot be parsed.
Video information error. Check whether the parameter is correct.	An error occurred while parsing video information
Failed to connect the service. Check the network settings.	Failed to obtain the video channel information.
Connection to service is denied.	Flash request triggered a security exception.
Video data is missing.	Video source data is missing.
CSS has finished. Try again later.	NetStream.Play.Stop event
CSS has finished. Try again later.	Failed to connect to the CSS source. The CSS source has not pushed video content

- Error messages returned from the backend

Code	Message	Description
1	Database error	Database connection timed out or an error occurred during query.
2	Failed to connect to CSS source. CSS source has not pushed video content (hls).	M3U8 file cannot be obtained due to CSS source connection failure.
3	CSS has finished (hls).	Manifest is not a valid M3U8 file, or CSS source has finished.

Code	Message	Description
113	Connection timeout. Try again later.	The parameter is incorrect.
1000	Incorrect channelID or APPID	The app_id is incorrect (length error).
1001	Invalid parameter. Failed to obtain the bizid.	The app_id is incorrect (with a correct length but wrong content).
1009	CSS source has expired.	The broadcast address is invalid. CSS source has expired.
10000	Request timeout	Timeout when pulling player configuration and video information. Check your network and try again. Timeout is 10 seconds.
10001	Failed to parse data	Failed to parse the data obtained by pulling player configuration and video information. It may be caused by a network problem or server exception
10002	Connection timeout. Try again later	Failed to pull player configuration and video information. It may be caused by a network problem or server exception
10008	Wrong password. Enter again.	Invalid password
10020	Insufficient balance in CSS account. Top up it in time	The balance is insufficient.
11044	Invalid request	The APPID is missing
11045	Channel ID is missing in the request parameter.	Channel ID is missing
11046	Wrong password. Enter again.	Password is missing
20110	Wrong password. Enter again.	Invalid password
20113	CSS has finished. Try again later.	Pulling stream does not exist (downstream type is not exist).
20201	CSS has finished. Try again later.	An error occurred while querying pushing stream (get upstream info error).
20301	CSS channel does not exist. Check the channel ID.	Incorrect channel_id (app_id is correct).

Note:
Note: In case of any error code that is not listed in this table, contact customer service. Our engineers can help you solve the problem.

FAQs

- **Why is the screen distortedly stretched when the video is played in H5?**
  Video stretching is not supported when playing video in H5. Check if the player container has right width/height configuration.

- **Why did I receive the message that "The CSS has finished. Try again later"?**
  If you get no response from the CSS address when attempting to connect to this address and fail to connect to it even after five attempts, this message appears. In this case, you need to verify whether the video stream is being pushed and whether the network connectivity is normal.

- **The video cannot be hidden on the QQ browser.**
  QQ browser takes over the video playback feature from H5, and the X5 kernel uses self-developed player to play videos. QQ browsers use a unified playback interface to ensure a good user experience. For more information, please see QQ Browser Documentation.

- **Video is automatically played in full screen mode on iOS.**
  By default, video is played in full screen mode on iOS system due to the webkit setting. To achieve inline playback within an App, you can set the webkit-playsinline attribute. Any Safari browser on iOS below 10 is unable to disable the automatic use of full screen mode.

- **Why does the Flash player have two play buttons in Chrome on PC?**
  Flash is no longer automatically played starting from Chrome 42. Chrome only plays major Flash CSS automatically, while other Flash CSS are paused, unless users enable them manually.

- **Why can I play CSS videos in browsers on PC, but not on mobile devices?**
  Only HLS videos are supported in the browsers on mobile devices. Check whether the CSS pulling address contains an HLS pulling URL.

Web VOD Player 1.0

Web VOD Player 1.0

Last updated：2021-03-24 15:34:45

Feature Description

This document describes how to use the Web Player (Web SDK) of Tencent Cloud VOD service, which allows users to directly use the proven video playback capability. Through flexible APIs, the SDK can be easily integrated with self-built Web Apps to achieve the playing with desktop Apps. At the same time, the Web SDK provides the capability of uploading videos at Web end.

The file formats supported by the SDK are limited by the global hotlink protection feature. For more information, please see FAQs on the official website. This document is intended for developers who want to use Tencent Cloud VOD player Web SDK for development and have basics knowledge in JavaScript.

Supported Features

Playback Format

The following table lists the video formats supported by Web SDK:

Playback Format	PC Browser	Mobile Browser
HLS (m3u8)	Yes	Yes
MP4	Yes	Yes
FLV	No	No

Android system compatibility: Unlike iPhones that only use a Safari browser, a wide range of native browsers may come with Android phones. For this reason, compatibility of Android browsers has become a problem that plagues the industry. Therefore, the above table does not apply to all Android phones.

Upload format

Video formats supported by the SDK for upload are as follows:

Standard	Detailed Format

Microsoft	WMV, WM, ASF, ASX
REAL	RM, RMVB, RA, RAM
MPEG	MPG, MPEG, MPE, VOB, DAT
Others	MOV, 3GP, MP4, MP4V, M4V, MKV, AVI, FLV, F4V

Transcode service of the VOD platform: Since MP4 and HLS (m3u8) formats are relatively widely supported for both PC and mobile browsers, Tencent Cloud VOD platform always publishes the uploaded videos in these formats.

Platform compatibility

It requires excessive effort to create two sets of codes for smartphone browser and PC browser. However, by using this player, the same code will adapt itself to PC browser/smartphone browser, which means the player will automatically choose the optimal playback method according to the platform it runs on. For example: On PC, the service will use Flash player as first priority in order to cope with various video formats. On smartphone, the service will use HTML5 technology to play videos instead.

Preparations

Step 1: Activate the service

Sign up for a Tencent Cloud account at Tencent Cloud official website, and activate the **VOD** service.

Step 2: Upload files

After the VOD service is activated, go to VOD Video Management to upload new video files. Since the document is meant to introduce how to use the player, this operation can help you obtain your own online video address. You are unable to enter this page if you haven't activated VOD service.

Step 3: Obtain an ID

After the video is uploaded, you can find the file ID (the most basic information for the player to play videos) in the Video Management page. The player also includes Quality Statistics feature. You need to verify your APPID which can be queried in the Video Management page before using the player. In the figure below, the ID on the left is the video file ID, while the other one is your APPID.

Step 4: Prepare pages

Introduce the initialization script to the page in which you want to play videos (including PC or H5):

```
<script src="//qzonestyle.gtimg.cn/open/qcloud/video/h5/h5connect.js" charset="ut
f-8"></script>;
```

Local file restriction: You cannot play local files using the player due to cross-domain issues. You must access the player through an IP or domain--this is why we asked you to upload a video file first and acquire its online playback address.

Adding Player

You can add a video player into your page by following the two simple steps below.

Step 1: Add a player container

Place a player container in the page where you want to display the player. For example, add the following code into index.html. The container ID, width and height can be customized.

```
<div id="id_video_container" style="width:100%; height:auto;"></div>;
```

Step 2: Create a Player object

Next, create a player object in the introduced JavaScript using a player constructor.

```
var player = new qcVideo.Player("id_video_container", {
"file_id": "1465197896261041838",
"app_id": "125132611",
"width":640,
"height":480
});
```

The constructor is used to generate a player object and find the corresponding video to play based on file_id and app_id. You can control the player with the player object. For more information, please see Overview of API Methods for the parameter options of player object.

Complete sample code

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0, minimum-scale=1.0, maximum-scale=1.0"/>
<title>VOD</title>
</head>
<body>
```

```
<div id="id_video_container" style="width:100%; height:auto;"></div>
<script src="//qzonestyle.gtimg.cn/open/qcloud/video/h5/h5connect.js" charset="ut
f-8"></script>
<script type="text/javascript">
(function () {
var player = new qcVideo.Player("id_video_container", {
"file_id": "14651978962610418 38",
"app_id": "125132611",
"width":640,
"height":480
});
})()
</script>
</body>
</html>
```

Other Cases

case 1: How do I play a video if I have the video address but not the file_id and app_id?

Video playback address need to be passed, in which case file_id and app_id are not required. JS example:

```
var option = {
"width": 640,
"height": 480,
//...You may use other custom attributes
"third_video": {
"urls":{
20 : "http://208.vod.myqcloud.com/204.mp4"//This is only an example. Please repla
ce this with actual video address
}
}
};
var player = new qcVideo.Player("id_video_container", option);
```

The "urls" attribute of the "third_video" parameter is an Object, where you can pass multiple video addresses with different video definitions. Detailed parameter descriptions can be found in Overview of API Methods.

Note:

"urls" should include at least one video address

case 2: How to use "On-screen Comment"?

After the initialization of player, you can add on-screen comments for videos by calling the addBarrage(barrage) method of player object. For more information on parameters, please see Overview of API Methods. For example, add two on-screen comments for the video that is being played:

```
var barrage = [
{"type":"content", "content":"hello world", "time":"1"},
{"type":"content", "content":"Center", "time":"1", "style":"C64B03;30","position"
:"center"}
];
player.addBarrage(barrage);
```

Note:

On-screen comment is only implemented at the frontend. Backend support should be self-developed. This feature only applies to Flash players on PC, but not supported in H5.

Case 3: How do I perform some operations at the end of the video, such as video recommendation?

Use the listener parameter and pass a callback function for the playStatus event. This function will be called when the playback status changes. For description on the specific callback function parameters, please see API Overview.

For example:

```
var option ={
"file_id":"14651978962610418 38",
"app_id":"125132611",
"width":800,
"height":720
//...You may use other custom attributes
};
var listener = {
playStatus: function (status){
//TODO
console.log(status);
}
};
var player = new qcVideo.Player("id_video_container", option, listener);
```

Case 4: How to make the player remember video playback progress and start playing the video from the same point the next time?

In "option", set the parameter "remember" to 1. The player records the time point when the video was stopped in the previous playback, and continues from this point upon the next playback. For example:

```
var option ={
"file_id":"14651978962610418 38",
"app_id":"125132611",
"width":800,
"height":720,
"remember":1
//...You may use other custom attributes
};
var player = new qcVideo.Player("id_video_container", option);
```

Case 5: How to make the player adjust its size automatically along with the web page?

Use the player object method "resize(width, height)" to modify player size dynamically.

```
player.resize(640, 480);
```

Case 6: How to play videos that have been configured with passwords from Cloud Video Management?

Just like playing normal videos, SDK automatically displays a password input window. Playback starts after the password is entered.

> **Note:**
> Password feature is only available when playing videos by passing video IDs.

Case 7: How to generate a link that is shared by using a QR code or a link?

Example (please replace the appid and fileid in the link with actual IDs):

```
http://play.video.qcloud.com/qrplayer.html?appid=1251769111&fileid=14651978969211
156176147&autoplay=0&sw=640&sh=426&$def=20&wmode=transparent
```

```
http://play.video.qcloud.com/iplayer.html?appid=1251769111&fileid=146519789692111
56176147&autoplay=0&sw=1800&sh=1200&def=20&wmode=transparent
```

Case 8: How to specify video playback definition?

Use the "definition" parameter to specify the definition with which the video is played (on condition that the definition is available for this video). This is available for two playback methods: play with video ID and play with address. See Parameter Description link for more information. For example:

```
var option ={
"file_id":"14651978969261415426",
"app_id":"1251606588",
"definition":30,
"width":800,
"height":700
};

var player = new qcVideo.Player("id_video_container", option);
```

Overview of API Methods

Constructor

```
qcVideo.Player(id, option, listener);
```

ID: String; **Required**; This parameter indicates the ID of the container where the player is located in the page and can be customized.

option: Object ; **Required**.

This parameter indicates the options that can be set for player's parameters. The options are as follows:

Parameter	Type	Default Value	Description
file_id	String	None	Unique ID of the VOD file. This is **Required** when playing video by video ID
app_id	String	None	The parameter is **Required** if the CSS video is played using video ID. For the videos under the same account, this parameter remains the same.
width	Number	None	**Required**, used to configure player width (in pixel). Example: 640
height	Number	None	**Required**, used to configure player height (in pixel). Example: 480
auto_play	Number	0	Whether auto playback is allowed. 0: Disable; 1: Enable **Note: This parameter only applies to Flash players on PC.**

disable_full_screen	Number	0	Whether full-screen mode is allowed. 0: Enable; 1: Disable **Note: This parameter only applies to Flash players on PC.**
disable_drag	Number	0	Whether users are allowed to drag the video progress bar. 0: Allow; 1: Disallow **Note: This parameter only applies to Flash players on PC.**
stretch_full	Number	0	Whether the video is scaled up to the same size as the player. 0: Do not scale up. 1: Scale up to full screen **Note: This parameter only applies to Flash players on PC.**
stop_time	Number	None	Trial mode. For example, if you set it to "60", the video playback stops in 60 seconds, and the "playStatus" event is triggered at the same time
remember	Number	0	Whether to remember last played position. 0: No. 1: Yes. If enabled, the time point when the video was stopped in the previous playback will be recorded and the next playback will start from this position. **Note: This parameter only applies to Flash players on PC.**
playbackRate	Number	1	Playback speed. For example,"2" means the video will be played at double speed, while "0.5" means half speed. **Note: This option is only applies to H5 players**
hide_h5_setting	Boolean	false	Whether to hide the H5 Settings button. true: Hide. false: Do not hide
hide_h5_error	Boolean	false	Whether to hide error messages prompted by H5. **Note: This parameter only applies to H5 players.**
WMode	String	window	When in window mode, you cannot put other page elements over the Flash player. You can change this into opaque or parameter values for other flash wmode if required. **Note: This parameter only applies to Flash players on PC.**
stretch_patch	Boolean	false	If configured as "true", the video ad that is displayed when the video starts, ends or pauses will be enlarged to cover the whole player.
definition	Number	None	Used to specify the definition with which the video will be played. The configured definition must be available for the video. Available values: 10, 20, 30, 40, 210, 220, 230, 240. For more information about the relation between these values and video types, see the parameter descriptions for third_video.
videos	Array	None	When hotlink protection is enabled, you can achieve player

			playback by configuring an accessible video address for "videos". The definition type is obtained by matching the URL with the URL prefix queried through the backend. For more information, please see User Guide on Hotlink Protection. For example: [`http://xxx.myqcloud.com/xxxyy\_f220.m3u8?**sign**=xxx` , ...]

third_video	Object	None	This option is only used when playing videos by using video addresses.
			Parameter Example:
			{'duration': 20, //Video duration (in sec). Optional parameter. If not passed, the video duration will be automatically updated when MetaData is loaded.
			Note: This parameter is required in case of MP4 playback.
			'urls': { // (**At least one address must be included. Be careful about the video format**)
			10: "address for mp4 mobile videos",
			20: "address for mp4 SD videos",
			30: "address for mp4 HD videos",
			40: "address for mp4 ultra high definition videos",
			210: "address for hls mobile videos",
			220: "address for hls SD videos",
			230: "address for hls HD videos",
			240: "address for hls ultra high definition videos"
			}
			}
			Note: If you simulate a mobile device in Chrome or other PC browsers, please use an address for MP4 videos.

listener: Object; Optional; List of callback functions in case of playing status change.

Function Name	Type	Description
fullScreen	function	Triggered when entering/exiting full-screen mode. Callback function parameter: isFullScreen: Boolean Returned value: true: enter full screen, false: exit full screen Example: `function(isFullScreen){ ... }` **Note: This event only applies to PC Platform Flash Player**
playStatus	function	Triggered when playback status changes. Callback function parameter "status": String

		Returned value: ready: "Player is ready"; seeking: "Search"; suspended: "Pause"; playing: "Playback in progress"; playEnd: "Playback ended"; stop: "Triggered by the end of trial duration"; error: "Triggered in case of H5 playback error" Example: function(status, msg){ ... }
dragPlay	function	Triggered when you drag and change the progress bar; second: Number Returned value: Final progress bar position (in sec) Example: `function(second){ ... }` **Note: This event only applies to PC Platform Flash Player**

Obtain parameter and status

Here are the methods for obtaining parameters and statuses for player object that is returned by the constructor

Method	Returned Value	Description
getVolume	Number. Value range: 0-1	Obtain the current volume
getDuration	Number (in sec)	Obtain the total duration of the current video
getCurrentTime	Number (in sec)	Obtain the current playback position
isSeeking	Boolean ; true indicates "loading"	Whether the current playback status is "loading"
isSuspended	Boolean ; true indicates "paused"	Whether the current playback status is "paused"
isPlaying	Boolean ; true indicates "playing"	Whether the current playback status is "playing"
isPlayEnd	Boolean ; true indicates "playback ended"	Whether the current playback status is "playback ended"
getWidth	Number(int)	Get the width of current player
getHeight	Number(int)	Get the height of current player
getClarity	Number(int) (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition")	Get the current video definition
getAllClaritys	Array<int> (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition")	Get all available definitions for the current video

Settings and actions

The player objects returned by the constructors can be set using the following methods:

Method	Description
resize(width,height)	Parameter: width: int; height: int Function: Set the width and height for current player. Returned value: none
play(second)	Parameter: second: int (in sec) Function: Start playback. You can specify a time point at which the video starts to play Returns: int Error Codes Note: "second" can only be a null value or 0 when you play a video by using a video address
pause()	Function: Pause the playback of current video Returned value: int Error codes
resume()	Funtion: Resume playback. Returned value: int Error codes
setClarity(clarity)	Parameter: clarity, int definition. Value range: (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition") Function: Change video definition Returned value: Int Error Codes Note: Make sure that the definition is available for the current video before configuring it using "clarity", otherwise the player may choose a definition based on the default player rules
changeVideo(opt)	Parameter: opt Object; Contains the basic information of the video to be played. This is nearly identical to the "second" parameter of the constructor. For more information, please see Constructor Instruction Function: Change video dynamically Returned value: int Error Codes
addBarrage(barrage)	Parameter: barrage, array barrage information [{ "type":"content", // Message type, content: plain text (**Required**) "content":"hello world", // Text message (**required**) "time":"1.101" ,//The time length (in sec) between the moment when the current method is called for adding a caption and the moment when the caption is displayed. (**Required**) "style": "C64B03;35" ,// Separated by semicolons; the first is color value, and the second is font size (optional) "postion":"center" // Location center: center, bottom: bottom, up: top (optional) }, ...] Function: Add on-screen comments

	Returned value: int Error Codes Note: **On-screen comment is only implemented at frontend, and backend functions should be self-developed. This function only applies to Flash players on PC.**
closeBarrage()	Function: disable on-screen comment. Call addBarrage again to re-enable the feature. Returned value: int Error codes Note: **On-screen comment is only implemented at frontend, and backend functions should be self-developed. This function only applies to Flash players on PC.**
The common error codes of the above methods are as follows:	
Error Code	Description
---	---
200	Operation successful
0	Player not fully initialized
-1	Failed to change video dynamically. Required parameter is missing
-2	Unknown operation command
-3	Playback time is beyond valid playback range

Video File Upload

Users can use VOD Web SDK to upload videos. Tencent Cloud Video users can therefore upload video files using Web. The SDK supports uploading via HTML5, but it's not possible for browsers that do not support HTML5.

For more information on how to proceed, see Cloud VOD Web Upload SDK.

Troubleshooting

Last updated：2022-03-21 12:45:42

Feature Description

This document describes how to use the Web Player (Web SDK) of Tencent Cloud VOD service, which allows users to directly use the proven video playback capability. Through flexible APIs, the SDK can be easily integrated with self-built Web Apps to achieve the playing with desktop Apps. At the same time, the Web SDK provides the capability of uploading videos at Web end.

The file formats supported by the SDK are limited by the global hotlink protection feature. For more information, please see FAQs on the official website. This document is intended for developers who want to use Tencent Cloud VOD player Web SDK for development and have basics knowledge in JavaScript.

Supported Features

Playback Format

The following table lists the video formats supported by Web SDK:

Playback Format	PC Browser	Mobile Browser
HLS (m3u8)	Yes	Yes
MP4	Yes	Yes
FLV	No	No

Android system compatibility: Unlike iPhones that only use a Safari browser, a wide range of native browsers may come with Android phones. For this reason, compatibility of Android browsers has become a problem that plagues the industry. Therefore, the above table does not apply to all Android phones.

Upload format

Video formats supported by the SDK for upload are as follows:

Standard	Detailed Format
Microsoft	WMV, WM, ASF, ASX

Standard	Detailed Format
REAL	RM, RMVB, RA, RAM
MPEG	MPG, MPEG, MPE, VOB, DAT
Others	MOV, 3GP, MP4, MP4V, M4V, MKV, AVI, FLV, F4V

Transcode service of the VOD platform: Since MP4 and HLS (m3u8) formats are relatively widely supported for both PC and mobile browsers, Tencent Cloud VOD platform always publishes the uploaded videos in these formats.

Platform compatibility

It requires excessive effort to create two sets of codes for smartphone browser and PC browser. However, by using this player, the same code will adapt itself to PC browser/smartphone browser, which means the player will automatically choose the optimal playback method according to the platform it runs on. For example: On PC, the service will use Flash player as first priority in order to cope with various video formats. On smartphone, the service will use HTML5 technology to play videos instead.

Preparations

Step 1: Activate the service

Sign up for a Tencent Cloud account at Tencent Cloud official website, and activate the **VOD** service.

Step 2: Upload files

After the VOD service is activated, go to VOD Video Management to upload new video files. Since the document is meant to introduce how to use the player, this operation can help you obtain your own online video address. You are unable to enter this page if you haven't activated VOD service.

Step 3: Obtain an ID

After the video is uploaded, you can find the file ID (the most basic information for the player to play videos) in the Video Management page. The player also includes Quality Statistics feature. You need to verify your APPID which can be queried in the Video Management page before using the player. In the figure below, the ID on the left is the video file ID, while the other one is your APPID.

Step 4: Prepare pages

Introduce the initialization script to the page in which you want to play videos (including PC or H5):

```
<script src="//qzonestyle.gtimg.cn/open/qcloud/video/h5/h5connect.js" charset="ut
f-8"></script>;
```

Local file restriction: You cannot play local files using the player due to cross-domain issues. You must access the player through an IP or domain--this is why we asked you to upload a video file first and acquire its online playback address.

Adding Player

You can add a video player into your page by following the two simple steps below.

Step 1: Add a player container

Place a player container in the page where you want to display the player. For example, add the following code into index.html. The container ID, width and height can be customized.

```
<div id="id_video_container" style="width:100%; height:auto;"></div>;
```

Step 2: Create a Player object

Next, create a player object in the introduced JavaScript using a player constructor.

```
var player = new qcVideo.Player("id_video_container", {
"file_id": "1465197896261041838",
"app_id": "125132611",
"width":640,
"height":480
});
```

The constructor is used to generate a player object and find the corresponding video to play based on file_id and app_id. You can control the player with the player object. For more information, please see Overview of API Methods for the parameter options of player object.

Complete sample code

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalab
le=0, minimum-scale=1.0, maximum-scale=1.0"/>
```

```
<title>VOD</title>
</head>
<body>
<div id="id_video_container" style="width:100%; height:auto;"></div>
<script src="//qzonestyle.gtimg.cn/open/qcloud/video/h5/h5connect.js" charset="ut
f-8"></script>
<script type="text/javascript">
(function () {
var player = new qcVideo.Player("id_video_container", {
"file_id": "14651978962610418 38",
"app_id": "125132611",
"width":640,
"height":480
});
})()
</script>
</body>
</html>
```

Other Cases

case 1: How do I play a video if I have the video address but not the file_id and app_id?

Video playback address need to be passed, in which case file_id and app_id are not required. JS example:

```
var option = {
"width": 640,
"height": 480,
//...You may use other custom attributes
"third_video": {
"urls":{
20 : "http://208.vod.myqcloud.com/204.mp4"//This is only an example. Please repla
ce this with actual video address
}
}
};
var player = new qcVideo.Player("id_video_container", option);
```

The "urls" attribute of the "third_video" parameter is an Object, where you can pass multiple video addresses with different video definitions. Detailed parameter descriptions can be found in Overview of API Methods.

Note:

"urls" should include at least one video address

case 2: How to use "On-screen Comment"?

After the initialization of player, you can add on-screen comments for videos by calling the addBarrage(barrage) method of player object. For more information on parameters, please see Overview of API Methods. For example, add two on-screen comments for the video that is being played:

```
var barrage = [
{"type":"content", "content":"hello world", "time":"1"},
{"type":"content", "content":"Center", "time":"1", "style":"C64B03;30","position"
:"center"}
];
player.addBarrage(barrage);
```

Note:
On-screen comment is only implemented at the frontend. Backend support should be self-developed. This feature only applies to Flash players on PC, but not supported in H5.

Case 3: How do I perform some operations at the end of the video, such as video recommendation?

Use the listener parameter and pass a callback function for the playStatus event. This function will be called when the playback status changes. For description on the specific callback function parameters, please see API Overview.

For example:

```
var option ={
"file_id":"1465197896261041838",
"app_id":"125132611",
"width":800,
"height":720
//...You may use other custom attributes
};
var listener = {
playStatus: function (status){
//TODO
console.log(status);
}
};
var player = new qcVideo.Player("id_video_container", option, listener);
```

Case 4: How to make the player remember video playback progress and start playing the video from the same point the next time?

In "option", set the parameter "remember" to 1. The player records the time point when the video was stopped in the previous playback, and continues from this point upon the next playback. For example:

```
var option ={
"file_id":"14651978962610418 38",
"app_id":"125132611",
"width":800,
"height":720,
"remember":1
//...You may use other custom attributes
};
var player = new qcVideo.Player("id_video_container", option);
```

Case 5: How to make the player adjust its size automatically along with the web page?

Use the player object method "resize(width, height)" to modify player size dynamically.

```
player.resize(640, 480);
```

Case 6: How to play videos that have been configured with passwords from Cloud Video Management?

Just like playing normal videos, SDK automatically displays a password input window. Playback starts after the password is entered.

> **Note:**
> Password feature is only available when playing videos by passing video IDs.

Case 7: How to generate a link that is shared by using a QR code or a link?

Example (please replace the appid and fileid in the link with actual IDs):

```
http://play.video.qcloud.com/qrplayer.html?appid=1251769111&fileid=14651978969211156176147&autoplay=0&sw=640&sh=426&$def=20&wmode=transparent
```

```
http://play.video.qcloud.com/iplayer.html?appid=1251769111&fileid=14651978969211156176147&autoplay=0&sw=1800&sh=1200&def=20&wmode=transparent
```

Case 8: How to specify video playback definition?

Use the "definition" parameter to specify the definition with which the video is played (on condition that the definition is available for this video). This is available for two playback methods: play with video ID and play with address. See Parameter Description link for more information. For example:

```
var option ={
"file_id":"14651978969261415426",
"app_id":"1251606588",
"definition":30,
"width":800,
"height":700
};
var player = new qcVideo.Player("id_video_container", option);
```

Overview of API Methods

Constructor

```
qcVideo.Player(id, option, listener);
```

ID: String; **Required**; This parameter indicates the ID of the container where the player is located in the page and can be customized.

option: Object ; **Required**.

This parameter indicates the options that can be set for player's parameters. The options are as follows:

Parameter	Type	Default Value	Description
file_id	String	None	Unique ID of the VOD file. This is **Required** when playing video by video ID
app_id	String	None	The parameter is **Required** if the live streaming video is played using video ID. For the videos under the same account, this parameter remains the same.
width	Number	None	**Required**, used to configure player width (in pixel). Example: 640
height	Number	None	**Required**, used to configure player height (in pixel). Example: 480

Parameter	Type	Default Value	Description
auto_play	Number	0	Whether auto playback is allowed. 0: Disable; 1: Enable **Note: This parameter only applies to Flash players on PC.**
disable_full_screen	Number	0	Whether full-screen mode is allowed. 0: Enable; 1: Disable **Note: This parameter only applies to Flash players on PC.**
disable_drag	Number	0	Whether users are allowed to drag the video progress bar. 0: Allow; 1: Disallow **Note: This parameter only applies to Flash players on PC.**
stretch_full	Number	0	Whether the video is scaled up to the same size as the player. 0: Do not scale up. 1: Scale up to full screen **Note: This parameter only applies to Flash players on PC.**
stop_time	Number	None	Trial mode. For example, if you set it to "60", the video playback stops in 60 seconds, and the "playStatus" event is triggered at the same time
remember	Number	0	Whether to remember last played position. 0: No. 1: Yes. If enabled, the time point when the video was stopped in the previous playback will be recorded and the next playback will start from this position. **Note: This parameter only applies to Flash players on PC.**
playbackRate	Number	1	Playback speed. For example,"2" means the video will be played at double speed, while "0.5" means half speed. **Note: This option is only applies to H5 players**
hide_h5_setting	Boolean	false	Whether to hide the H5 Settings button. true: Hide. false: Do not hide
hide_h5_error	Boolean	false	Whether to hide error messages prompted by H5. **Note: This parameter only applies to H5 players.**
WMode	String	window	When in window mode, you cannot put other page elements over the Flash player. You can change this into opaque or parameter values for other flash wmode if required. **Note: This parameter only applies to Flash players on PC.**
stretch_patch	Boolean	false	If configured as "true", the video ad that is displayed when the video starts, ends or pauses will be enlarged to cover the whole player.

Parameter	Type	Default Value	Description
definition	Number	None	Used to specify the definition with which the video will be played. The configured definition must be available for the video. Available values: 10, 20, 30, 40, 210, 220, 230, 240. For more information about the relation between these values and video types, see the parameter descriptions for third_video.
videos	Array	None	When hotlink protection is enabled, you can achieve player playback by configuring an accessible video address for "videos". The definition type is obtained by matching the URL with the URL prefix queried through the backend. For more information, please see User Guide on Hotlink Protection. For example: [`http://xxx.myqcloud.com/xxxyy\_f220.m3u8?**sign**=xxx` , ...]

third_video	Object	None	This option is only used when playing videos by using video addresses.
			Parameter Example:
			{'duration': 20, //Video duration (in sec). Optional parameter. If not passed, the video duration will be automatically updated when MetaData is loaded.
			Note: This parameter is required in case of MP4 playback.
			'urls': { // (**At least one address must be included. Be careful about the video format**)
			10: "address for mp4 mobile videos",
			20: "address for mp4 SD videos",
			30: "address for mp4 HD videos",
			40: "address for mp4 ultra high definition videos",
			210: "address for hls mobile videos",
			220: "address for hls SD videos",
			230: "address for hls HD videos",
			240: "address for hls ultra high definition videos"
			}
			}
			Note: If you simulate a mobile device in Chrome or other PC browsers, please use an address for MP4 videos.

listener: Object; Optional; List of callback functions in case of playing status change.

Function Name	Type	Description
fullScreen	function	Triggered when entering/exiting full-screen mode. Callback function parameter: isFullScreen: Boolean Returned value: true: enter full screen, false: exit full screen Example:
Note: This event only applies to PC Platform Flash Player		
playStatus	function	Triggered when playback status changes. Callback function parameter "status": String Returned value: ready: "Player is ready"; seeking: "Search"; suspended: "Pause"; playing: "Playback in progress"; playEnd: "Playback ended"; stop: "Triggered by the end of trial duration"; error: "Triggered in case of H5 playback error" Example: function(status, msg){ ... }
dragPlay	function	Triggered when you drag and change the progress bar; second: Number Returned value: Final progress bar position (in sec) Example:
Note: This event only applies to PC Platform Flash Player		

Obtain parameter and status

Here are the methods for obtaining parameters and statuses for player object that is returned by the constructor

Method	Returned Value	Description
getVolume	Number. Value range: 0-1	Obtain the current volume
getDuration	Number (in sec)	Obtain the total duration of the current video

Method	Returned Value	Description
getCurrentTime	Number (in sec)	Obtain the current playback position
isSeeking	Boolean ; true indicates "loading"	Whether the current playback status is "loading"
isSuspended	Boolean ; true indicates "paused"	Whether the current playback status is "paused"
isPlaying	Boolean ; true indicates "playing"	Whether the current playback status is "playing"
isPlayEnd	Boolean ; true indicates "playback ended"	Whether the current playback status is "playback ended"
getWidth	Number(int)	Get the width of current player
getHeight	Number(int)	Get the height of current player
getClarity	Number(int) (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition")	Get the current video definition
getAllClaritys	Array<int> (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition")	Get all available definitions for the current video

Settings and actions

The player objects returned by the constructors can be set using the following methods:

Method	Description
resize(width,height)	Parameter: width: int; height: int Function: Set the width and height for current player. Returned value: none
play(second)	Parameter: second: int (in sec) Function: Start playback. You can specify a time point at which the video starts to play Returns: int Error Codes Note: "second" can only be a null value or 0 when you play a video by using a video address
pause()	Function: Pause the playback of current video Returned value: int Error codes

Method	Description
resume()	Funtion: Resume playback. Returned value: int Error codes
setClarity(clarity)	Parameter: clarity, int definition. Value range: (1: "Mobile", 2: "Standard definition", 3: "High definition", 4: "Ultra high definition") Function: Change video definition Returned value: Int Error Codes Note: Make sure that the definition is available for the current video before configuring it using "clarity", otherwise the player may choose a definition based on the default player rules
changeVideo(opt)	Parameter: opt Object; Contains the basic information of the video to be played. This is nearly identical to the "second" parameter of the constructor. For more information, please see Constructor Instruction Function: Change video dynamically Returned value: int Error Codes
addBarrage(barrage)	Parameter: barrage, array barrage information [{ "type":"content", // Message type, content: plain text (**Required**) "content":"hello world", // Text message (**required**) "time":"1.101" ,//The time length (in sec) between the moment when the current method is called for adding a caption and the moment when the caption is displayed. (**Required**) "style": "C64B03;35" ,// Separated by semicolons; the first is color value, and the second is font size (optional) "postion":"center" // Location center: center, bottom: bottom, up: top (optional) }, ...] Function: Add on-screen comments Returned value: int Error Codes Note: **On-screen comment is only implemented at frontend, and backend functions should be self-developed. This function only applies to Flash players on PC.**
closeBarrage()	Function: disable on-screen comment. Call addBarrage again to re-enable the feature. Returned value: int Error codes Note: **On-screen comment is only implemented at frontend, and backend functions should be self-developed. This function only applies to Flash players on PC.**
The common error codes of the above methods are as follows:	

Method	Description
Error Code	Description
200	Operation successful
0	Player not fully initialized
-1	Failed to change video dynamically. Required parameter is missing
-2	Unknown operation command
-3	Playback time is beyond valid playback range

Video File Upload

Users can use VOD Web SDK to upload videos. Tencent Cloud Video users can therefore upload video files using Web. The SDK supports uploading via HTML5, but it's not possible for browsers that do not support HTML5.

For more information on how to proceed, see Cloud VOD Web Upload SDK.