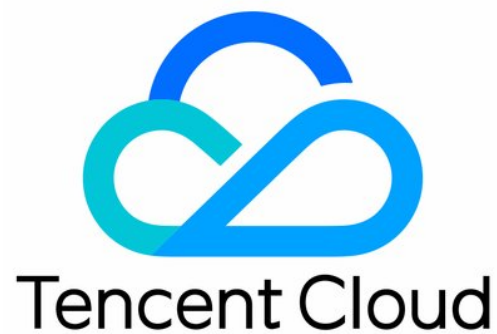


Cloud Streaming Services

Feature Guide

Product Documentation



Copyright Notice

©2013-2022 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Feature Guide

Push and Playback

- Live Push

- Live Playback

- Splicing Live Streaming URLs

- Delayed Playback

- SRT Push

Features

- Live Remuxing and Transcoding

- Live Recording

 - Recording Storage on VOD

- Time Shifting

- Live Screenshot

- Live Porn Detection

- AV1 Encoding

- Stream Mix

Video Content Protection

Hotlink Protection URL Calculation

Global CSS Service

- Overview

- HTTPDNS Routing

Callback Notifications

- How to Receive Event Notification

- Stream Pushing Notification

- Recording Event Notification

- Screenshotting Event Notification

- Porn Detection Event Notification

- Relay Event Notification

User Guides for Common Third-Party Tools

- Push via OBS

- VLC Player

Feature Guide

Push and Playback

Live Push

Last updated : 2022-06-10 16:06:58

The nature of CSS is a streaming process, similar to the live broadcast of TV channels sent to audience through cable networks. In order to complete this process, CSS needs to have a capture and push device (similar to a camera), a cloud live streaming service (similar to a cable network), and a playback device (similar to a TV set). These devices can be smart devices such as mobile phones, PCs, and tablets as well as web browsers. We provide complete software demos for different types of devices.

Preparations

1. Activate the [CSS service](#).
2. Select [Domain Management](#), click **Add Domain** to add a push domain name with an ICP filing number. For more information, please see [Adding Domain Name](#).

Note :

CSS provides a default push domain name in the format of `xxx.livepush.myqcloud.com` . We recommend you not use it as the push domain name for your real business.

Getting Push Address

Log in to the CSS console, select **CSS Toolkit** > [Address Generator](#) to generate a push address and configure as follows:

- Select **Push Domain** as the domain type.
- Select the push domain name you added in domain management.
- Enter an `AppName` (`live` by default). This is used to differentiate the paths of different applications under the same domain name.
- Enter a custom `StreamName` , such as `liveteststream` .
- Select the expiration time of the address, such as `2019-10-18 23:59:59` .
- Click **Generate Address**.

Note :

- To ensure the security of your live streams, the system will automatically enable push authentication. You can also select the push domain name to be modified in [Domain Management](#) and click **Manage** on the right to enter the domain name details page and customize the authentication information in **Push Configuration**. The push address is in the following format:

```
rtmp://domain/AppName/StreamName?
```

```
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

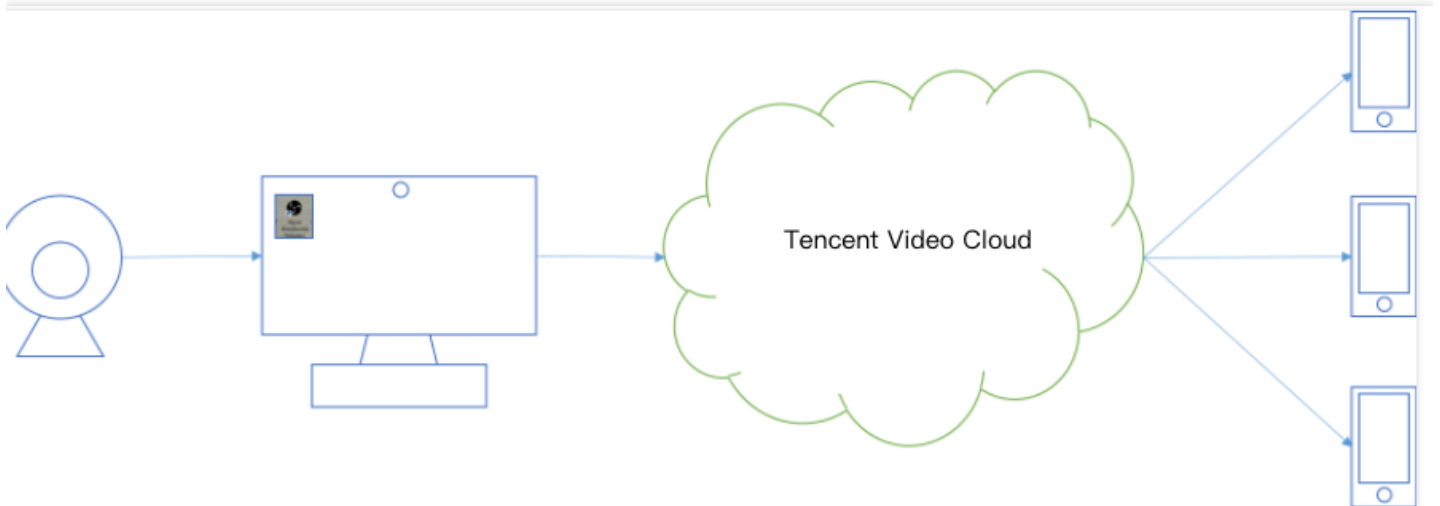
- In addition to the above method, you can also select a push domain name in [Domain Management](#) in the CSS console, click **Manage**, select **Push Configuration**, enter the expiration time of the push address and the custom `StreamName` , and click **Generate Push Address** to generate a push address.
- If you need a **persistent push address**, you can enter [Domain Management](#), select a push domain name, click **Manage**, and select **Push Configuration** for calculation and generation by referring to the sample code in **Push Address Sample Code**. For more information, please see [How can I view the push sample code?](#).

Live Push

You can use the following methods to implement live push based on your business scenario:

Scenario 1. PC push

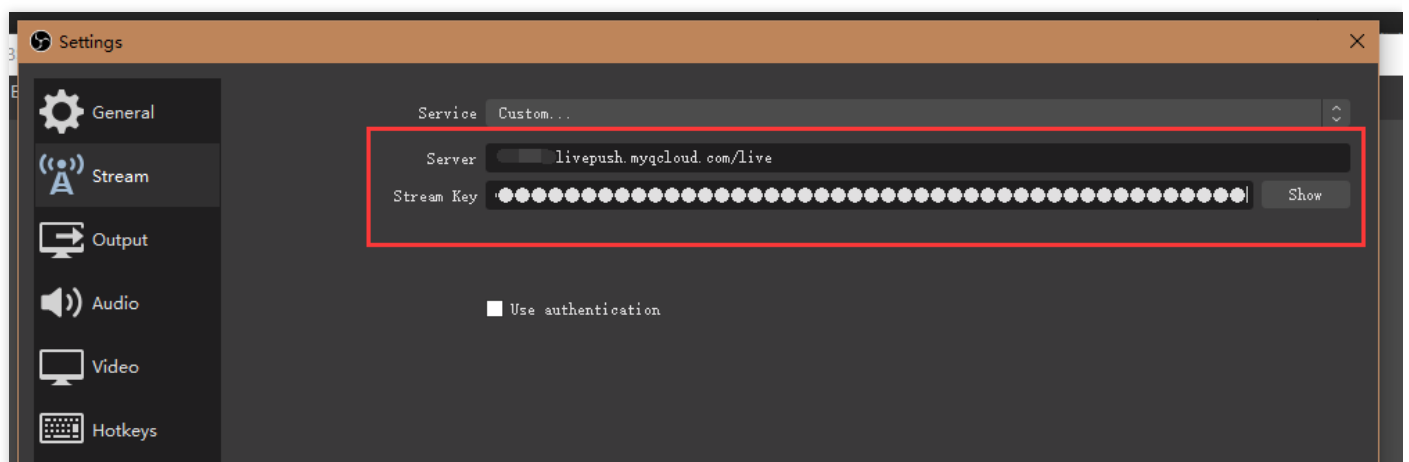
For PC (Windows/macOS), you can choose to install [OBS](#) or [XSplit](#) for push. The former is a free open-source video recording and streaming program that supports operating systems such as Windows, macOS, and Linux, while the latter is a paid program that offers a standalone installer for live game streaming. For non-game live streaming, we recommend you use BroadCaster.



This document uses push with OBS as an example to describe the steps. Assume that the prepared push address is:

```
rtmp://3891.livepush.myqcloud.com/live/3891_test?bizid=3891&txSecret=xxx&txTime=58540F7F
```

1. Go to [OBS official website](#) to download and install the push tool.
2. Open OBS and click **Controls** > **Settings** at the bottom to enter the settings page.
3. Click **Stream** to enter the push configuration page and set as follows:
4. Select "Custom" as the service type.
5. Enter the first half of the push address as the server, such as
`rtmp://3891.livepush.myqcloud.com/live/` .
6. Enter the second half of the push address as the stream key, such as `3891_test?bizid=3891&txSecret=xxx&txTime=58540F7F` .
7. Click **OK** in the bottom-right corner.



8. Click **Controls** > **Start Streaming** to test streaming. For more information on how to use OBS, please see [Push via OBS](#).

Scenario 2. Web push

1. Log in to the CSS console.
2. Select **CSS Toolkit** > **Web Push**.
3. Perform the following settings on the web push page:
4. Select a push domain name.
5. Enter an `AppName` (`live` by default). This is used to differentiate the paths of different applications under the same domain name.
6. Enter a custom `StreamName` , such as `liveteststream` .
7. Select an expiration time, such as `2019-10-30 23:59:59` .
8. Click **Start Push** and grant the camera permission to start the push.

Note :

The web push feature requires that your device have a camera installed and its browser support the Flash plugin to call the camera permission.

Select push domain *



Select a push domain. If there is no available domain name, please [Add Domain](#)

AppName *

live

Use "live" by default. Only letters, digits, and symbols are supported.

StreamName *

test

Only support letters, digits, and symbols.

Expiration Time

2020-05-11 23:59:59



Push address expiration time is the setting time

Start Push

Stop Push

Scenario 3. Mobile push

1. Scan the QR code with a mobile phone to download and install the Video Cloud Toolkit.
2. Open the toolkit and select **MLVB > Camera Push**.
3. Enter the [push address](#) manually or by scanning the QR code.
4. Tap **Start** in the bottom-left corner to start the push.

Note :

If you did not prepare a push address in advance, you can tap **New** on the right of the push address bar on the **Camera Push** page, and the system will automatically enter a push address and provide the corresponding playback address which can be used for live playback.

Scenario 4. Live SDK push

If you need to integrate only live push into your existing application, follow the steps below:

1. Download the MLVB SDK.
2. Complete the integration as instructed in the iOS or Android integration document.

The live SDK is a collection of mobile live streaming services. It demonstrates in the form of free source code how to use Tencent Cloud CSS, VOD, IM, and COS to build the most appropriate live streaming solution for your business.

FAQs

- [How can I implement live playback?](#)
- [How can I splice a push URL?](#)
- [How can I calculate a hotlink protection URL?](#)

Live Playback

Last updated : 2022-07-26 16:33:12

Preparations

1. Activate the [CSS service](#).
2. Log in to the [CSS console](#) to get a URL for live push. For detailed directions, please see [Live Push](#).
3. Select [Domain Management](#), click **Add Domain**, enter your domain name, select **Playback Domain** as the type, and click **Save**.
4. Log in to the [Tencent Cloud Domain Service Console](#) and configure CNAME for the successfully added playback domain name. For detailed directions, please see [Domain Name CNAME Configuration](#).

Getting Playback URL

Select **CSS Toolkit** > [Address Generator](#) to get a playback URL and configure as follows:

- Select **Playback Domain** as the type of the URL.
- Select a playback domain name you added in **Domain Management**.
- Enter the same `StreamName` as that of the push URL. The `StreamName` of the playback URL must be the same as that of the push URL to play back the corresponding stream.
- Select the expiration time of the URL, such as `2019-10-18 23:59:59`.
- Click **Generate Address**.

Domain Type • Playback domain 🔍

If you select push domain, a push address will be generated; and if you select playback domain, a playback address will be generated. If there is no available domain, please [Add Domain](#)

AppName • live

Use "live" by default. Only letters, digits, and symbols are supported.

StreamName • livetteststream

Only support letters, digits, and symbols.

Expiration Time 2020-05-11 23:59:59 📅

The expiration time of playback address is the setting timestamp plus the playback authentication expiration time, and the push address expiration time is the setting time.

[Generate Address](#) [Address Resolution Sample](#)

Note :

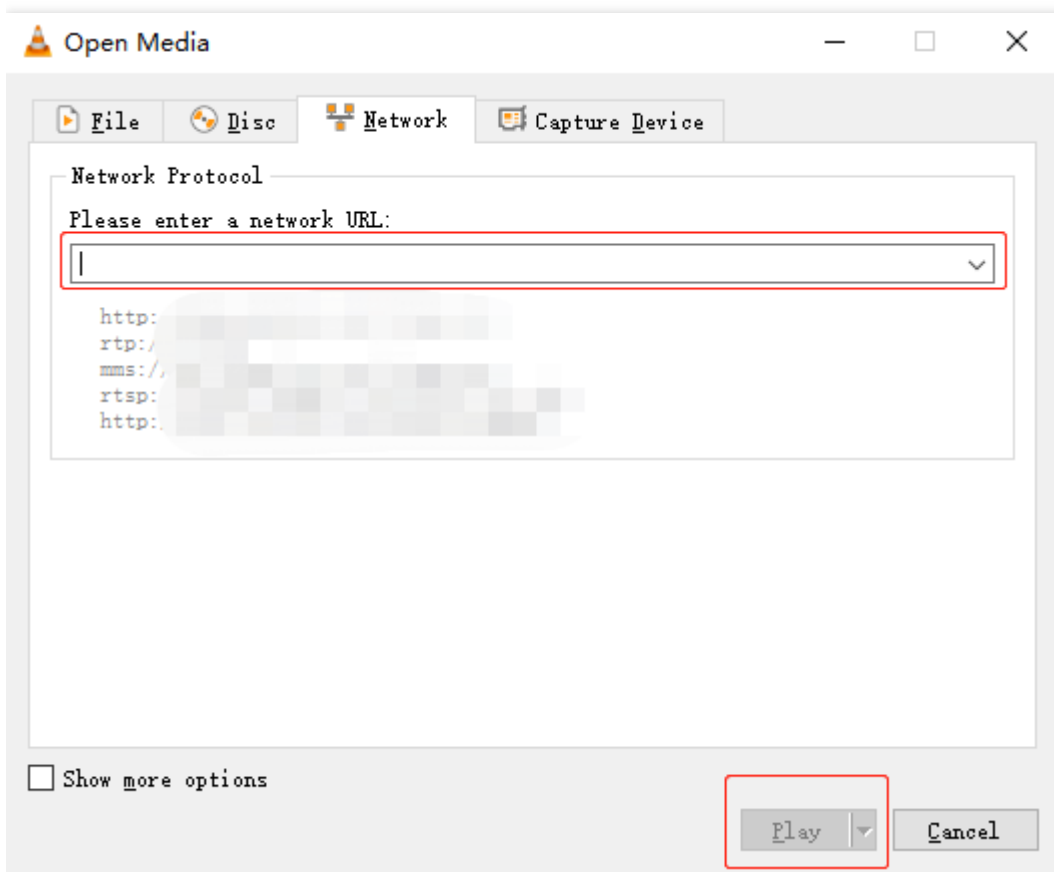
In addition to the above method, you can also select a playback domain name in **Domain Management** in the CSS console, click **Manage**, select **Playback Configuration**, enter the expiration time of the playback URL and the `StreamName` same as that in the push URL, and click **Generate Playback Address**.

Live Playback

A [live push](#) must be successful before the stream can be watched via the playback URL. You can use the following methods to test live streaming based on your business scenario:

Scenario 1. Playback on PC client

You can use tools such as [VLC](#), FFmpeg, and [TCPlayerDemo](#) for playback.



Scenario 2. Playback on mobile client

1. Download the install [Tencent Cloud Toolkit](#).
2. Select **MLVB > LVB Playback** or **LEB Playback**.
3. Enter the playback URL in the input box or scan the QR code of the playback URL.
4. Tap the play button in the bottom-left corner to start playback.

Scenario 3. Playback on web

You are recommended to choose TCPlayer in the player SDK for playback. Based on Tencent Cloud's powerful backend functionality and AI technology, TCPlayerLite provides excellent playback capabilities for live streaming and video on-demand. Deeply integrated with the Tencent Cloud LVB and VOD services, Player+ features smooth and stable playback performance, advertising placement, and data monitoring.

Note :

Currently, most mobile browsers on the market do not support HTTP-FLV playback. Therefore, for web-based playback, you are recommended to select the HTTP-FLV playback protocol for PC browsers and HLS for mobile browsers.

FAQs

- [What playback protocols are supported?](#)
- [What does a playback address consist of?](#)
- [How can I use live transcoding?](#)
- [How can I use time shifting for replay?](#)
- [How can I use HTTPS for playback?](#)
- [How can I use a global cache node for playback?](#)
- [How can I enable hotlink protection?](#)

Splicing Live Streaming URLs

Last updated : 2022-01-24 14:20:01

Notes

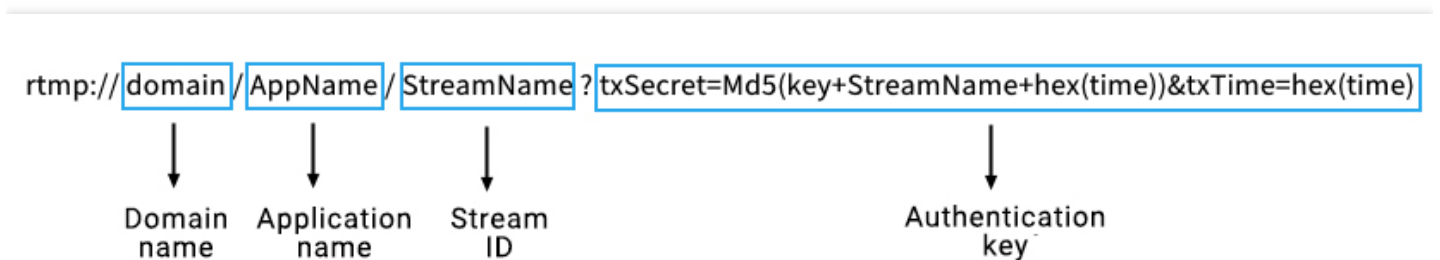
After you create a [transcoding template](#) and [bind](#) it with a playback domain name, you need to add the transcoding template name after the `StreamName` of the live stream with the transcoding configuration in the format of `StreamName_transcoding template name`. For details, see [Playback Configuration](#).

Prerequisites

- You have signed up for a Tencent Cloud account and activated the [CSS service](#).
- You have applied for a domain name through [Tencent Cloud Domain Service](#).
- You have added push/playback domain names in [Domain Management](#) of the CSS console and successfully configured the CNAME record. For detailed directions, please see [Adding Domain Names](#).

Splicing Push URLs

If you run a large number of live streaming rooms, it is impossible to manually generate a push and playback URL for each host. In such cases, you can use the server to automatically **splice** the addresses. Any URL that meets Tencent Cloud standards can be used for push. A standard push URL consists of four parts, as shown below:



- **Domain**

Push domain name, which can be the default push domain name provided by Tencent Cloud CSS or a push domain name that you have added and created a CNAME record for.

- **AppName**

Live streaming application name, which is `live` by default and is customizable.

- **StreamName (stream ID)**

Custom stream name, which is the unique ID of a live stream. We recommend that you use a random numeric or alphanumeric string for this parameter.

- **Authentication key (optional)**

An authentication key consists of `txSecret` and `txTime` :

```
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time) .
```

If push authentication is enabled, the URL used for push must contain an authentication key. If push authentication is disabled, the push URL does not need to contain "?" and the content following it.

- **txTime (URL expiration time)**

The time when the URL expires, in the format of hexadecimal Unix timestamp.

Note :

For example, `5867D600` means that the URL expires at 00:00:00, January 1, 2017. The validity period should neither be too short nor too long. Most of our clients set `txTime` to a point 24 hours or longer from the current time. If the validity period is too short, after a host is disconnected due to network problems during a live broadcast, it may be impossible to resume the push due to expiration of the push URL.

- **txSecret (hotlink protection signature)**

The `txSecret` signature serves to prevent attackers from forging a backend to generate push URLs. For the calculation method, see [Best Practice - Hotlink Protection URL Calculation](#).

Splicing Playback URLs

A playback URL consists of a playback protocol prefix, domain name (`domain`), application name (`AppName`), stream name (`StreamName`), playback protocol suffix, authentication key, and other custom parameters. Below are a few examples.

```
webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.flv?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
http://domain/AppName/StreamName.m3u8?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

- **Playback prefix**

Playback Protocol	Playback Prefix	Notes
WebRTC	<code>webrtc://</code>	We recommend WebRTC most as it has the best instant streaming performance and supports ultra-high concurrency.
HTTP-FLV	<code>http://</code> or <code>https://</code>	We recommend HTTP-FLV as it has good instant streaming performance and supports high concurrency.
RTMP	<code>rtmp://</code>	We do not recommend RTMP as it has poor instant streaming performance and does not support high concurrency.
HLS (M3U8)	<code>http://</code> or <code>https://</code>	We recommend HLS for mobile clients and for the Safari browser on macOS.

- **Domain**

Playback domain name, a domain name you have added and created a CNAME record for.

- **AppName**

Live streaming application name used to identify the storage path of a live streaming media file. The application name is `live` by default and customizable.

- **StreamName (stream name)**

Custom stream name, which is the unique ID of a live stream. We recommend you use a random numerical or alphanumerical string.

- **Authentication key (optional)**

An authentication key consists of `txSecret` and `txTime` :

```
txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)
```

If playback authentication is enabled, the URL used for playback must contain an authentication key. If it is disabled, the playback URL does not need to contain "?" and the content following it.

- **txTime (address expiration time):** the time when the URL expires, in the format of hexadecimal Unix timestamp.
- **txSecret (hotlink protection signature):** it serves to prevent attackers from forging a backend to generate playback URLs. For the calculation method, see [Best Practice - Hotlink Protection URL Calculation](#).

Viewing Sample Push Codes

Go to [Domain Management](#) of the CSS console, select a pre-configured push domain name, and click **Manage > Push Configuration** to display the **Push Address Sample Code** (for both PHP and Java) that demonstrates how to generate a hotlink protection address. For detailed directions, please see [Push Configuration](#).

Delayed Playback

Last updated : 2022-05-07 10:44:37

Delayed playback is a feature that allows you to delay the playing of streams. It is mainly used in important live streaming events to allow organizers time to handle emergencies. You can enable this feature through parameter setting.

Notes

You can enable delayed playback via two methods:

- Call the [playback delaying API](#).
- Add a `txDelayTime` parameter to the end of a **push URL**. For details, please see [Push Configuration](#).

Note :

The API method is not recommended because calling an API involves configuration caching, which makes it difficult to estimate when the feature takes effect. You are advised to enable the feature using the second method.

Preparations

1. Activate [CSS](#).
2. Log in to the CSS console, select [Domain Management](#), and click **Add Domain Name** to add a push domain name. For more information, please see [Adding Domain Name](#).

Push Configuration

1. Log in to the CSS console, go to **CSS Toolkit** > [Address Generator](#), select **Push Domain** for **Domain Type**, and click **Generate Address**.

Domain Type * **Push Domain** ▼

If you select push domain, a push address will be generated; and if you select playback domain, a playback address will be generated. If there is no available domain, please [Add Domain](#)

AppName * **live**

Use "live" by default. Only letters, digits, and symbols are supported.

StreamName * **livetest**

Only support letters, digits, and symbols.

Expiration Time **UTC+8** **2021-07-02 09:59:12** 📅

The expiration time of playback address is the setting timestamp plus the playback authentication expiration time, and the push address expiration time is the setting time.

Generate Address [Address Resolution Sample](#)

Generation Result Generate the following address according to the above settings)

Type	Push Domain
Expiration Time	2021-07-02 09:59:12 (UTC+8) reference documentation
Push Address	rtmp://[redacted]/live/livetest?txSecret=e295b1624692aa085df3a5adde2d02c7&txTime=60DE72F0 🔗
OBS Push Address	rtmp://[redacted]/live/ 🔗
OBS Push Name	livetest?txSecret=e295b1624692aa085df3a5adde2d02c7&txTime=60DE72F0 🔗

2. Add `txDelayTime` to the end of the push address, and push streams via OBS. For detailed directions, please see [Push via OBS](#).

General Stream Output Audio

Service Custom...

Server rtmp://[redacted]/live/

Stream Key livetest?txSecret=e295b1624692aa085df3a5adde2d02c7&txTime=60DE72F0txDelayTime=30 Hide

☒ Use authentication

Note :

Set `txDelayTime` to the number of seconds for which you want to delay playback. The value must be an integer and cannot exceed 600.

Delayed Playback

1. Log in to the CSS console, go to **CSS Toolkit** > [Address Generator](#), select **Playback Domain** for **Domain Type**, and click **Generate Address**.
2. Use [VLC](#), FFmpeg, or other tools for playback. For details, please see [CSS Playback](#).

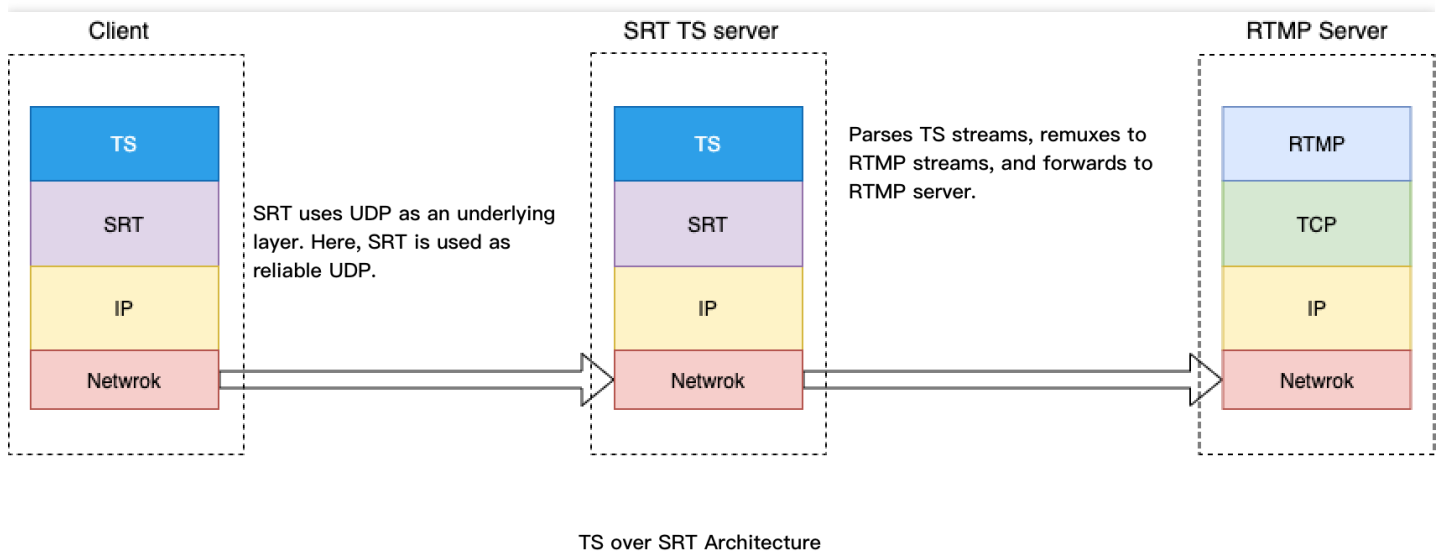
In the figure above, the delay time set for playback via the `txDelayTime` parameter in the push address is 30s, and the actual playback latency is 34s, which indicates that the delayed playback feature has taken effect.

SRT Push

Last updated : 2021-08-27 16:50:45

TS over SRT directly transmits TS streams containing audio/video data using **SRT protocol**. The existing live streaming system is used for playback. TS over SRT is used as the standard push format for Haivision hardware and OBS.

In this mode, the SRT server parses TS streams, remuxes to RTMP streams, and forwards to the backend RTMP server.

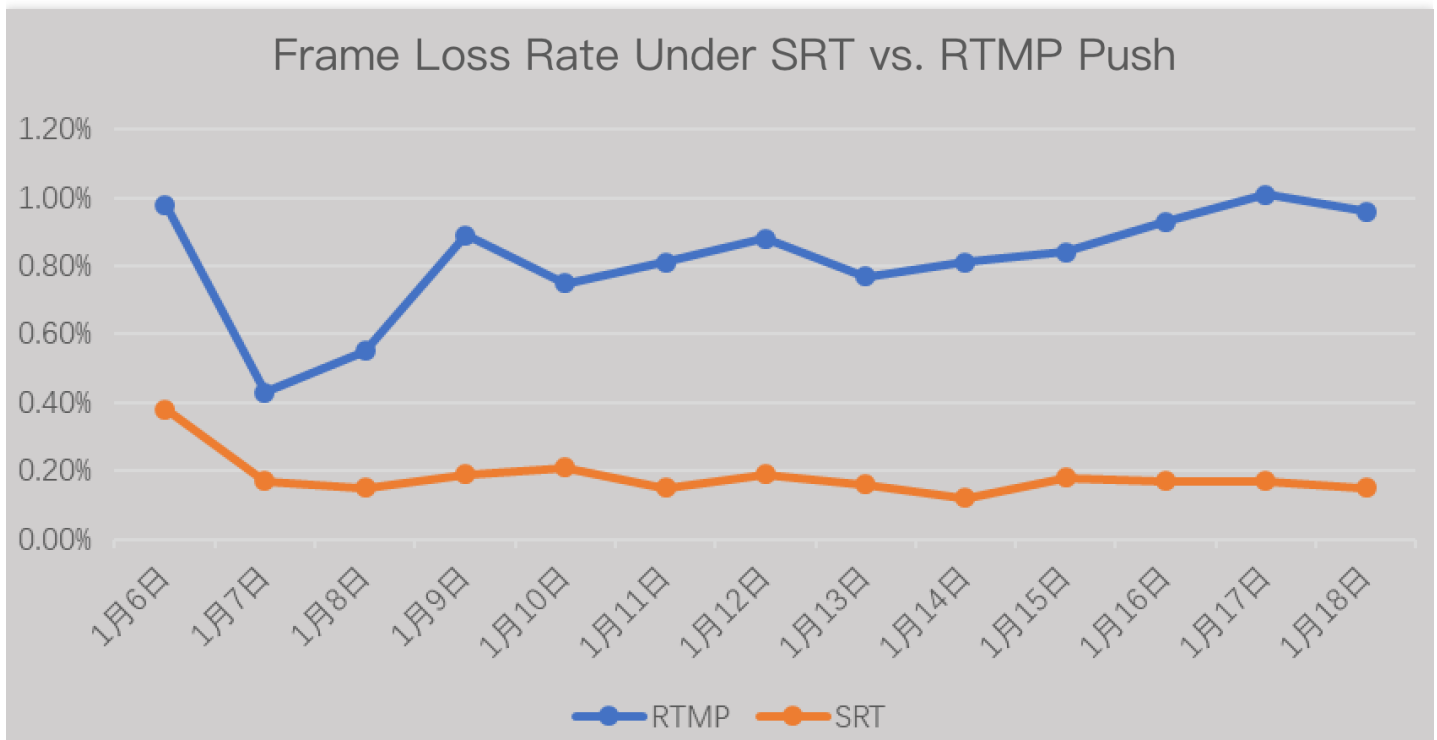


Note :

Using SRT for push will not increase the cost.

Upstream Lag Rate Comparison

Using SRT to push streams reduces lag, as shown in the following figure:



Packet Loss Rate Comparison

Using SRT to push streams optimizes upstream performance, resulting in better playback smoothness. The following shows the performance comparison of the Douyu app.

- Android: performance test data of push over SRT (test device — Mi 9):

Metric	Protocol	0%	10%	20%	30%	40%	50%
Push smoothness	TCP	Smooth	Severe stutter				
	SRT	Smooth	Smooth	Smooth	Smooth	Smooth	Occasional stutter
Total Men Avg/MB	TCP	380	374				
	SRT	377	387	375	368	377	379
App CPU Max/%	TCP	7	8				
	SRT	9	9	10	10	9	10
App CPU Avg/%	TCP	7	7				
	SRT	8	8	9	9	8	9
Phone CPU Max/%	TCP	33	33				
	SRT	33	33	33	31	35	35
Phone CPU Avg/%	TCP	29	30				
	SRT	30	31	32	32	32	32

- iOS: performance test data of push over SRT (test device — iPhone XR):

Metric	Protocol	0%	10%	20%	30%	40%	50%
Push smoothness	TCP	Smooth	Severe stutter				
	SRT	Smooth	Smooth	Smooth	Smooth	Smooth	Occasional stutter
Total Men Avg/MB	TCP	298	296				
	SRT	275	300	302	291	319	322
App CPU Max/%	TCP	13	13				
	SRT	10	10	10	10	10	14
App CPU Avg/%	TCP	7	6				
	SRT	7	7	7	7	7	7
Phone CPU Max/%	TCP	47	42				
	SRT	32	34	45	37	38	67
Phone CPU Avg/%	TCP	31	29				
	SRT	28	29	30	30	30	29

Packet Loss Prevention Comparison

Compared with QUIC, SRT reduces packet loss at the application layer under the same packet loss rate, thanks to its faster, more precise retransmission control and pacing mechanism for live streaming scenarios. When the packet loss rate is 50%, SRT can still guarantee stable transmission.

With the same linkage and the same live stream file on the push end, the packet loss rate reduces by 5% every five minutes when SRT is used. The following figure shows that the push frame rate of SRT is more stable.

Live Push

Access method

Live push supports using **port 9000** to push streams over SRT. You can [generate a push address](#) via the [Address Generator](#) in the CSS console and splice the address by following the rules below.

Tencent Cloud SRT push address:

```
srt://${rtmp-push-domain}:9000?streamid=#!::h=${rtmp-push-domain},r=${app}/${stream},txSecret=${txSecret},txTime=${txTime}
```

Note :

`$ {app}` is a variable and should be replaced with the actual value. Note that `$` , `{` , and `}` are not required.

Implementation method

The SRT server remuxes TS streams to RTMP streams and forward them to the `${rtmp-push-domain}` domain.

Sample of OBS live stream code:

Note :

If you want to push streams over SRT, the OBS version cannot be lower than v25.0.

Live Pull

Follow the general pull and playback process. For details, see [CSS Playback](#).

Features

Live Remuxing and Transcoding

Last updated : 2022-08-19 10:06:21

Live Remuxing

Live remuxing is the process of converting the original stream pushed from the live streaming site (commonly using the RTMP protocol) into different container formats in the cloud before pushing to viewers.

Supported output container formats

- RTMP
- FLV
- HLS
- DASH
- HDS
- TS stream

Supported output types

- Audio-only output: deletes video files and generates audio-only output. The container formats are as described above.
- Video-only output: deletes audio files and generates video-only output. The container formats are as described above.

Supported media encryption schemes

- **FairPlay**
HLS remuxing supports the Apple FairPlay DRM solution.
- **Widevine**
DASH remuxing supports the Google Widevine DRM solution.
- **Universal AES-128 encryption for HLS**
HLS remuxing supports universal AES-128 encryption schemes.

Live Transcoding

Live transcoding (including both video transcoding and audio transcoding) is the process of transcoding the original stream pushed from the live streaming site to streams of different codecs, resolutions, and bitrates in the cloud before pushing to viewers. This helps meet the playback needs in different network environments and on different devices.

Typical use cases

- An original video stream can be transcoded to streams of different definitions. Viewers can select video streams of different bitrates according to their network conditions to ensure smooth playback.
- You can add a custom watermark to an original video stream for copyright and marketing purposes.
- A video stream can be transcoded to a video codec with a higher compression ratio. For example, when there is a large number of viewers, you can convert an H.264 video stream to an H.265 stream which has a higher compression ratio, thus reducing bandwidth usage and costs.
- An original video stream can be transcoded to different codecs suitable for playback on special devices. For example, if an H.264 video stream cannot be played back in real time due to issues in performance, you can transcode it to the .mpeg format for real-time decoding and playback.

Video transcoding parameters

Parameter Type	Description
Video codec	Supported video codecs: <ul style="list-style-type: none">• H.264• H.265
Video profile	Supported video profiles: <ul style="list-style-type: none">• Baseline• Main• High
Video encoding bitrate	<ul style="list-style-type: none">• Supported video output bitrate range: 50 Kbps - 10 Mbps.• The original bitrate will be the output bitrate if you specify an output bitrate higher than the original one. For example, if the specified output bitrate is 3,000 Kbps, yet the original bitrate of the input stream is only 2,000 Kbps, then the output bitrate will be 2,000 Kbps.
Video encoding frame rate	<ul style="list-style-type: none">• Supported video output frame rate range: 1-60 fps.• The original frame rate will be the output frame rate if you specify an output frame rate higher than the original one. For example, if the specified output frame rate is 30 fps, yet the original frame rate of the input stream is only 20 fps, then the output frame rate will be 20 fps.
Video resolution	<ul style="list-style-type: none">• Supported width range: 0 - 3000.• Supported height range: 0 - 3000.• You can only specify the width and the height will be scaled proportionally.

	<ul style="list-style-type: none">You can only specify the height and the width will be scaled proportionally.
Video GOP length	Supported video GOP length range: 1-10s; recommended range: 2-4s.
Video bitrate control method	Supported video bitrate control methods: <ul style="list-style-type: none">Fixed bitrate (CBR)Dynamic bitrate (VBR)
Video image rotation	The original video can be rotated clockwise by: <ul style="list-style-type: none">90 degrees180 degrees270 degrees

Audio transcoding parameters

Parameter Type	Description
Audio codec	Supported codecs: <ul style="list-style-type: none">AAC-LCAAC-HEAAC-HE v2
Audio sample rate	Supported sample rates (48000 and 44100 are commonly used): <ul style="list-style-type: none">96000640004800044100320002400016000120008000
Audio encoding bitrate	Supported bitrate range: 20-192 Kbps; commonly used bitrates include: <ul style="list-style-type: none">48 Kbps64 Kbps128 Kbps
Sound channel	Supported sound channel modes: <ul style="list-style-type: none">MonoDual

Common preset templates for video transcoding

Video Definition	Template Name	Video Resolution	Video Bitrate	Video Frame Rate	Video Codec
Smooth	550	Image short side (proportionally scaled) x long side (540)	500 Kbps	23	H.264
SD	900	Image short side (proportionally scaled) x long side (720)	1000 Kbps	25	H.264
HD	2000	Image short side (proportionally scaled) x long side (1080)	2000 Kbps	25	H.264

Top Speed Codec Transcoding

Based on years of experience in audio/video encoding, intelligent scenario recognition, dynamic encoding, the three-level (CTU/line/frame) precise bitrate control model, and other technologies, the Top Speed Codec (TSC) transcoding feature provides higher-definition streaming at lower bitrates (30% less on average) for live streaming and video on-demand.

Use cases

If the live push bitrate is high and the image is complex, you can use the intelligent dynamic encoding technology and precise bitrate control model to keep a high definition at a low bitrate, ensuring that the quality of the video image watched by the viewer is the same as the original quality.

Advantages

As users of various video platforms have an ever-increasing requirement for high video source definition and smooth watch experience, in the current live streaming industry, 1080p resolution and 3-10 Mbps bitrate have gradually become the mainstream configuration, and the bandwidth costs are taking a large part in the total video platform costs. In this case, the reduction of the video bitrate can effectively reduce the bandwidth costs.

Example:

Suppose you held a live session at 3 Mbps for 4 hours with 200 viewers. The codec is H.264 and TSC transcoding is not used. The peak bandwidth is 600 Mbps. The bandwidth cost for this live session is $600 \times 0.2118 = 127.08$ USD.

- If TSC transcoding is used to reduce the bitrate, the incurred bandwidth fees will be around $127.08 \times (100\% - 30\%) = 88.956$ USD.
- TSC transcoding fees: $0.0443 \times 240 = 10.632$ USD (published price without any discount applied).
- Total fees: $88.956 + 10.632 = 99.588$ USD.

Therefore, TSC transcoding can effectively reduce the platform bandwidth costs while delivering a better watch experience.

Key parameters

The parameters of TSC transcoding are configured basically in the same way as standard live transcoding parameters. For more information, please see [Video transcoding parameters](#).

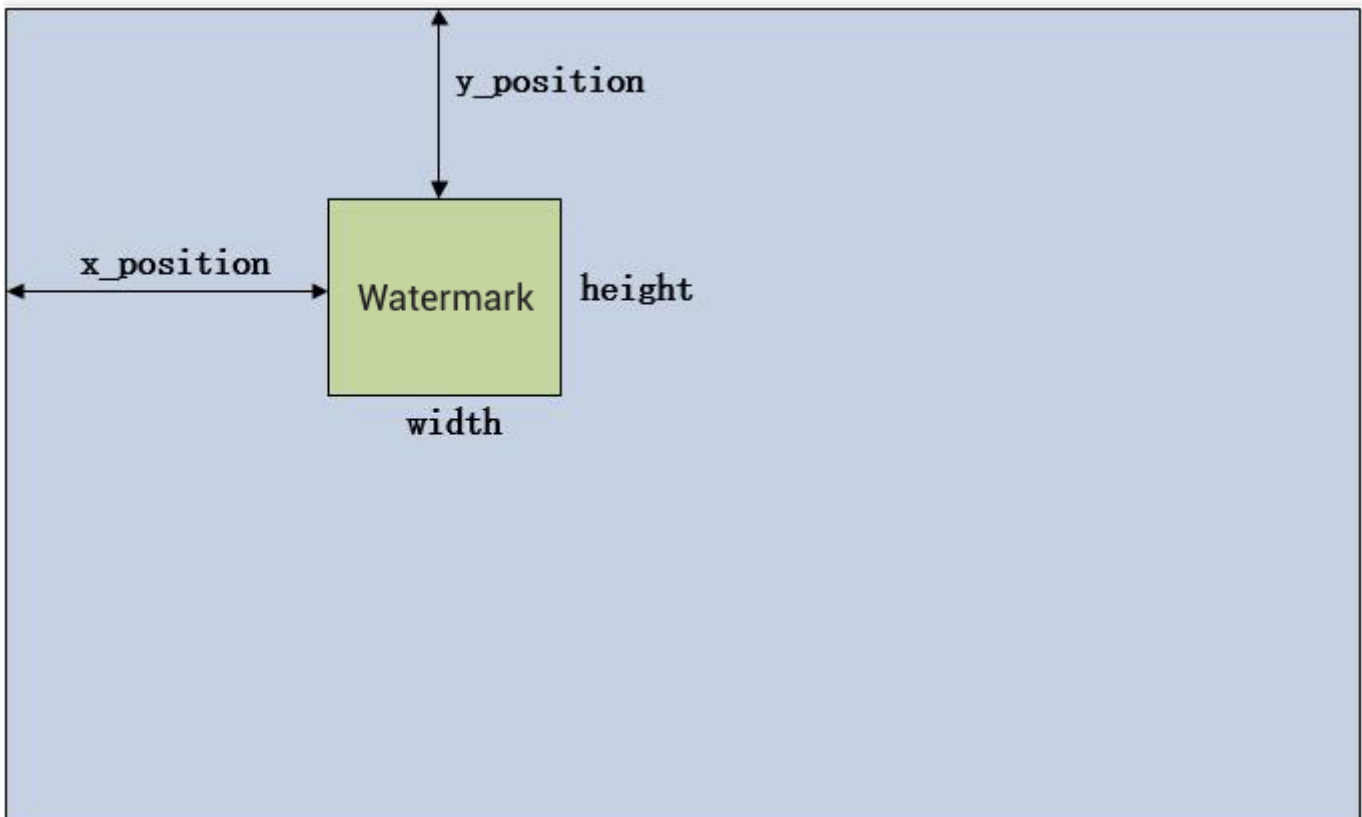
Live Watermarking

You can use live watermarking to add a preset logo image to an original video stream for copyright and marketing purposes.

Watermark parameters

The main parameters of a watermark include watermark location and watermark size, which are determined by the `XPosition` , `YPosition` , `Width` and `Height` parameters as detailed below:

- **XPosition:** X-axis offset, which indicates the percentage distance from the left edge of the watermark to the left edge of the video.
- **YPosition:** Y-axis offset, which indicates the percentage distance from the top edge of the watermark to the top edge of the video.
- **Width:** watermark width or its percentage of the live streaming video width.
- **Height:** watermark height or its percentage of the live streaming video height.



Note :

If you enable multi-bitrate transcoding for a stream (i.e., one source stream is transcoded into streams of different resolutions) and want to add a watermark, you can set its percentage position on the X and Y axes in the [CSS console](#) or through the corresponding [API](#), and the watermark position will be automatically determined by the system.

Example of watermark parameters

Suppose the resolution of the output image is 1920 x 1080, the watermark resolution is 320 x 240, XPosition = 5, YPosition = 5, and Width = 10 (unit: percent).

The absolute position and size of the watermark on the output video are as shown below:

```
XPosition_pixel = 1920 x 5% = 96
YPosition_pixel = 1080 x 5% = 54
Width_pixel = 1920 x 10% = 192
Height_pixel = 192 x 240/320 = 144
```

The watermark is at 96 pixels away from the left edge of the output video image and 54 pixels away from the top edge of the image. The watermark size is 192 x 144.

How to use

You can add a watermark in the [CSS console](#) or through a [server API](#) based on your business needs.

CSS console

1. Go to **Feature Configuration > Live Watermarking** to add a watermark configuration template, set the watermark parameters, and generate the corresponding watermark template ID. For specific steps, please see [Watermark Template Configuration](#).
2. Select **Domain Management** to add a domain name, and click **Manage > Template Configuration** to bind it with the watermark template. For more information, please see [Watermark Configuration](#).

Calling APIs

3. Call the [AddLiveWatermark](#) API to add a watermark by setting the watermark name and other parameters.
4. Call the [CreateLiveWatermarkRule](#) API to create a watermark rule. Set `DomainName` (push domain name) and `WatermarkId` (returned in step 1). Use the same `AppName` as the `AppName` in push and playback addresses, which is `live` by default.

Note: Using the watermark feature will incur standard transcoding fees.

Configuring Transcoding Parameters

How to use

You can set transcoding parameters via the [CSS console](#) or [server APIs](#). Either way, you will mainly use watermark templates, transcoding templates, and transcoding rules for the configuration.

CSS console

1. Go to **Feature Configuration > Live Transcoding** to add a transcoding configuration template. You can add a [standard transcoding](#) or [TSC transcoding](#) template.
2. Create the corresponding transcoding type and set transcoding parameters as needed. You can use the system's default parameters, and a corresponding transcoding template ID will be generated.
3. Select **Domain Management** to find the target pull domain name, and click **Manage > Template Configuration** to bind it with the transcoding template. For more information, please see [Transcoding Configuration](#).

Calling APIs

1. Call the [CreateLiveTranscodeTemplate](#) API to set the transcoding type parameters.

2. Call the [CreateLiveTranscodeRule](#) API to set the `DomainName` (pull domain name) and `TemplateId` (returned in step 1) parameters. Enter an empty string in `AppName` and `StreamName` as a wildcard for matching all streams under the domain name. You can also bind the transcoding template with different stream names to enable transcoding for these live streams.
3. Each transcoding template has a **unique transcoding template name** which is used as the unique ID for playing back the output stream. You can place the transcoding template name after the stream ID in the playback address to pull the output stream corresponding to the transcoding template.

Note :

The transcoding rule is used to set whether to enable a specified transcoding template for a specified domain name or stream. A playback domain name can be used to pull a transcoding template only after the corresponding transcoding rule is created. If no transcoding rule has been created, a pull address spliced using the transcoding template name is invalid.

Example

```
**Playback address = Playback domain name + Playback path + Stream ID_transcoding  
template name + Authentication string**
```

For a push with stream ID of `1234_test`, the original stream and watermarked streams of different bitrates can be played back via the following addresses:

- **Original stream:** `http://liveplay.tcloud.com/live/1234_test.flv?authentication string`
- **Standard transcoding stream (watermarked):**
`http://liveplay.tcloud.com/live/1234_test_sd.flv?authentication string`
- **TSC transcoding stream (watermarked):** `http://liveplay.tcloud.com/live/1234_test_hd.flv?
authentication string`

Note :

To play back a watermarked stream, you need to bind the corresponding push domain name to the created watermark template.

Using APIs

1. Manage transcoding templates in the console:

You can query, add, modify, and delete transcoding templates in the console.

2. Manage transcoding templates through server APIs:

Feature Module	API
Live Transcoding	CreateLiveTranscodeTemplate
	ModifyLiveTranscodeTemplate
	DescribeLiveTranscodeTemplate
	DescribeLiveTranscodeTemplates
	DeleteLiveTranscodeTemplate
	CreateLiveTranscodeRule
	DescribeLiveTranscodeRules
	DeleteLiveTranscodeRule
Live Watermarking	AddLiveWatermark
	UpdateLiveWatermark
	DeleteLiveWatermark
	DescribeLiveWatermarks

Live Recording

Recording Storage on VOD

Last updated : 2022-03-21 09:17:52

Live recording stores the files generated by muxing original streams (without modifying information such as audio and video data and corresponding timestamps) on the VOD platform.

Notes

- You can use either of the following methods to record: [create a recording task](#) or [create a recording template](#). If you create both a recording template and a recording task for the same live stream, it will be recorded repeatedly.
- As there is a short delay in starting a recording task after a stream is pushed, a very short push cannot generate recording files. It is recommended that the duration of each push for recording be longer than 10s.

Recording Storage

As the recording files are stored on the VOD platform, you need to activate the [VOD service](#) first.

Note :

For the naming rules of generated recording files, please see [VodFileName](#).

Recording Format

Supported recording file formats include .aac (for audio recording), .flv, .hls, and .mp4.

Recording Use Cases

Use Case	Description
----------	-------------

Use Case	Description
Multi-level recording by push domain name and stream name	You can configure whether to record a stream at the push domain name and stream name level.
Recording within a specified time period	You can call APIs to set the start time and end time to record a stream within the specified time period.
Real-time recording	You can call APIs to record any frame of a stream in real time.
Pure audio recording	You can use .aac format to record pure audio streams.

Enabling Recording for All Live Streams under a Specified Push Domain Name

Recording parameters are managed by templates. You can create recording templates for different scenarios and flexibly manage the recording configurations by binding the templates with different push domain names and stream names.

After activating VOD, you can record live streams under a specified push domain name in two ways:

CSS console

1. Go to **Feature Configuration > Live Recording** to create a recording template.
2. Select **Domain Management** to add a domain name, and click **Manage** to bind it with the recording template. For more information, see [Recording Configuration](#).

APIs

1. Call the [CreateLiveRecordTemplate](#) API to set at least one recording format, such as `FlvParam`.
2. Call the [CreateLiveRecordRule](#) API, setting `DomainName` (push domain name) and `TemplateId` (returned in step 1). You can leave `AppName` and `StreamName` empty to record all streams under the domain name.

You can also specify a stream to record.

A template can be bound to different push domain names, applications, and streams, but the same push domain name, application, or stream cannot be bound with multiple templates. If you bind the same stream with multiple

templates (in rare cases), only the one with the highest priority will take effect. The priority of a template is determined as follows.

Priority	DomainName	AppName	StreamName
1	✓	✓	✓
2	✓	×	✓
3	✓	✓	×
4	✓	×	×

✓ means the value of the parameter is not empty, and × means it is empty.

Disabling Recording for Specific Streams Under a Push Domain Name

When you have already configured recording for a push domain name but do not need to record some streams under it:

1. Call the [CreateLiveRecordTemplate](#) API without specifying any recording format.

```
https://live.tencentcloudapi.com/?Action=CreateLiveRecordTemplate
&TemplateName=norecord
&Description=test
&<Common request parameters>
```

2. Go to the [CSS console](#) or use the [CreateLiveRecordRule](#) API to bind the above recording template with specific `DomainName` and `StreamName`.

Note :

This method is applicable to scenarios where only a few streams do not need to be recorded. If there are too many streams, you're advised to use another push domain name to manage them, because:

- The allowed maximum number of recording templates or recording rules is 50.
- Management by push domain name is more flexible as recording templates and recording rules won't be affected even when your business changes.

Recording within a Specified Time Period

You can use APIs to specify the start time, end time and other parameters of recording for some streams. That is different from using a preset recording template with specified parameters. Usually, APIs are used when no recording template is created.

APIs

Call the [CreateRecordTask](#) API.

Recording sample

- In simple scenarios, you need to specify only `StreamName` , `DomainName` , `AppName` , and `EndTime` .
The following sample code creates a video recording task in .flv format for 8 AM to 10 AM, August 10, 2020, with 30-minute segments, and the recording files will be retained permanently.

Sample input code:

```
https://live.tencentcloudapi.com/?Action=CreateRecordTask
&AppName=live
&DomainName=mytest.live.push.com
&StreamName=livetest
&StartTime=1597017600
&EndTime=1597024800
&TemplateId=0
&<Common request parameters>
```

- You can also specify the recording format, recording type, and storage parameters.
The following sample code creates a recording task in .mp4 format for 8 AM to 10 AM, August 10, 2020, with 1-hour segments, and the recording files will be retained permanently.
 - Call the [CreateLiveRecordTemplate](#) API to create a recording template.

Sample input code:

```
https://live.tencentcloudapi.com/?Action=CreateLiveRecordTemplate
&TemplateName=templat
&Description=test
&Mp4Param.Enable=1
&Mp4Param.RecordInterval=3600
&Mp4Param.StorageTime=0
&<Common request parameters>
```

Sample output code:

```
{
  "Response": {
    "RequestId": "839d12da-95a9-43b2-a9a0-03366d01b532",
    "TemplateId": 17016
  }
}
```

2. Call the [CreateRecordTask] (<https://intl.cloud.tencent.com/document/product/267/37309>) API to create a recording task.

Sample input code:

```
https://live.tencentcloudapi.com/?Action=CreateRecordTask
&StreamName=livetest
&AppName=live
&DomainName=mytest.live.push.com
&StartTime=1597017600
&EndTime=1597024800
&TemplateId=17016
&<Common request parameters>
```

Note :

- For the same live stream, there is no conflict between scheduled tasks or between a scheduled task and a recording task of another type. In other words, the time periods of multiple scheduled tasks can overlap, and you can call APIs to create a recording task in addition to enabling a recording configuration.
- You're advised to create a recording task beforehand (for example, 1 hour in advance or early in the morning if your event takes place during the day), and set the task start time slightly earlier than the event start time.

Real-Time Recording

If you want to record any frames immediately in the process of live streaming to generate highlight clips, you can call APIs to enable real-time recording.

```
https://live.tencentcloudapi.com/?Action=CreateRecordTask
&StreamName=test
&AppName=live
&DomainName=mytest.live.push.com
```

```
&EndTime=1597024800  
&<Common request parameters>
```

Notes on real-time recording:

- Make sure that the push is ongoing when you create a recording task.
- You can call the [StopRecordTask](#) API to stop a task in advance.
- This is also supported for streams outside the Chinese mainland.

Mixed Stream Recording

First, please familiarize yourself with [Live Stream Mixing](#).

There are two types of stream mixing indicated by the `OutputStreamType` parameter:

- If `OutputStreamType` is set to `0`, the output stream is in the input stream list, meaning that no new stream will be generated.
- If `OutputStreamType` is set to `1`, the output stream is not in the input stream list, meaning that a new stream will be generated.

Assume the pushed streams are A and B, and the mixed stream is the output stream C:

- Suppose `OutputStreamType` is set to `0` and the name of stream A is used as the name of the output stream C. After the recording is started, recording files of stream A (mixed stream) and stream B will be generated. As the name of stream A is reused, the original stream A will not generate a recording file.
- If `OutputStreamType` is set to `1`, recording files of stream A, stream B, and stream C (mixed stream) will be generated after the recording is started.

If you only want to record the mixed stream, you can call the [CreateRecordTask](#) API. Please note that if

`OutputStreamType` is set to `1`, the `StreamType` parameter should be set to `1` when this API is called.

Note :

Mixed stream recording does not support mixing streams in and outside Chinese mainland, as recording file errors will occur and affect normal playback.

Auto-Spliced Recording (Multi-Push Recording)

If several recording files are generated when a push is interrupted multiple times due to network jitters, it will affect the continuous playback of the live stream. To solve this problem, live recording can auto-splice multiple recording files between short stream interruptions into one file.

This feature segments audio and video data by `#EXT-X-DISCONTINUITY` tags in HLS recording. Due to stream interruptions, timestamps of audio and video data, video codec, audio codec and sample rates before and after tagging may be different. The player needs to refresh the decoder to achieve seamless playback. To use this feature, the player should support the `#EXT-X-DISCONTINUITY` tag. Currently, the tag is supported on native player and Safari on iOS, ExoPlayer on Android, and HLS.js player on web, but not supported on VLC player.

After this feature is enabled, you need to set the auto-splicing timeout period. This period is up to 30 minutes, meaning that recording files between interruptions of up to 30 minutes can be spliced into one HLS file after the last push ends.

Currently, auto-spliced recording is supported only for HLS format. You can set the auto-splicing timeout period in [Live Recording](#).

Note :

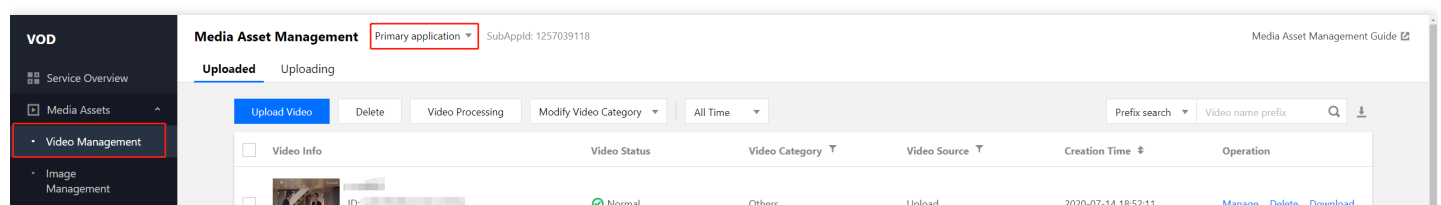
- This mode does not support live streams with no audio data.
- The `ComposeMedia` API of VOD can be used to compose video files. For more information, please see [ComposeMedia](#).
- After HLS recording resumption is enabled, a callback will be triggered only when a recording file is generated, not when the stream is interrupted.

Obtaining Recording Files

Recording files are automatically saved in the VOD system after generation and can be found via:

VOD console

Log in to the [VOD console](#) and select **Media Assets > Video Management** on the **non-admin** page to browse all the generated recording files.



Recording event notification

The recording callback address can be set in the console or through API calls. A notification will be sent to the callback address after the recording files are generated. After that, you can refer to the recording [callback event message notification](#) to take your next step.

As the event notification callbacks are efficient, reliable, and in real time, you're recommended to use them to get recording file information.

Query by VOD API

You can call the [SearchMedia](#) API of VOD to filter and query recording files.

Note :

When you call the [CreateRecordTask](#) API, [stream_param](#) parameters carried in the push URL will not be returned in the recording callback. Yet if you use other recording methods, such parameters will be returned in the recording callback.

Notes on Modifying Configuration

You are advised to restart the push and verify the recording configuration if you modified the configuration. The configuration takes effect by the following rules:

- By default, the configuration takes effect in 10 minutes.
- The configuration is effective upon the start of the live push and will not be updated in the process of recording.
- In scenarios where the push lasts for a long time (surveillance recording for example), you need to interrupt and restart the push for the configuration to take effect.

Time Shifting

Last updated : 2022-06-02 16:51:05

Powered by the recording capability of CSS, the time shifting feature allows viewers to rewind and play back a video stream from earlier time points. When time shifting is enabled for a VOD playback domain, TS segment URLs and TS files for the video are saved in VOD, and the user can play back earlier video content by passing a time parameter in the request URL under the playback domain.

How It Works

In HLS streaming, a video stream is split into TS segments. Viewers use an M3U8 file to access a TS segment URL, get the TS file, and play the video content starting from that TS segment.

Note :

TS files are not saved permanently, so there is a limit to how far back in time playback can start from.

Note

The time shifting feature has been in beta testing so far. However, we will start charging for use of the feature starting from **00:00 on June 1, 2022**. For details, see [Notice: Time Shifting to Become Paid Feature](#). Time shifting relies on recording, so you will also be charged [live recording fees](#) by CSS and [storage and playback fees](#) by VOD.

How to Use Time Shifting

Prerequisites

- You have [signed up for a Tencent Cloud account](<https://intl.cloud.tencent.cn/document/product/378/17985>).
- You have activated CSS and added a [push domain name](#).

Step 1. Activate VOD

- Log in to the [VOD console](#) and click **Activate Now**.
- Select the checkbox to agree to the service agreement, and click **OK** to activate VOD.

Step 2. Add a domain name

Follow the steps below to add a VOD domain name for time shifting:

1. Go to the VOD console and select **Distribution and Playback > Domain Name** on the left sidebar.
2. Click **Add Domain** and enter a VOD domain name that has been registered with an ICP filing number. For more information, see [Distribution and Playback Settings](#).
3. Add a CNAME record for the domain.

Step 3. Bind a recording template

1. Go to the CSS console and select **Feature Configuration > Live Recording**.
2. Click **Create Recording Template**. For detailed directions for creating a recording template, see [Live Recording](#).

Note :

- Choose **HLS** as the recording format.
- Enter a custom storage period, which cannot be shorter than the [time-shift duration](#).

3. Bind the recording template with the push domain you want to use. For detailed directions, see [Recording Configuration](#).

Step 4. Enable time shifting

[Submit a ticket](#) to enable the time shifting feature. You need to select **CSS** as the product and provide the following information:

- The **VOD domain name** added in [Step 2](#).
- The ID of the recording template added in [Step 3](#).
- A custom value (seconds) for `timeshift_dur` (time-shift duration).

Note :

- The time-shift duration indicates how far back from the current time you can play back the video stream. Currently, the longest time-shift duration allowed is 30 days.
- Given that the time-shift duration you configure may not exactly match the actual time-shift duration, we recommend you set the duration a little longer than you actually need.
- For example, if the parameter is set to `7200` (2 hours), you will be able to request content generated 2 hours ago or later, and the value range for the playback delay parameter `delay` is 90 seconds to 2 hours. If `delay` is set to a value larger than 2 hours, `HTTP 404` will be returned even if there is live streaming content at that time point.

Playback Request

Request URL format

```
http://[Domain]/timeshift/[AppName]/[StreamName]/timeshift.m3u8?delay=xxx
```

Parameter description

Parameter	Description
[Domain]	The VOD domain name added in Step 2 for time shifting.
timeshift	A non-customizable parameter.
[AppName]	The application name. For example, if your application name is <code>live</code> , set this parameter to <code>live</code> .
[StreamName]	The stream name. Set this parameter to the name of the stream for which you want to enable time shifting.
timeshift.m3u8	A non-customizable parameter.
delay	The playback delay time (seconds). If you pass in a value smaller than <code>90</code> , <code>90</code> will be used.

Example

Suppose the time-shift domain name is `testtimeshift.com` , application name is `live` , and stream name is `SLPUrIFzGPE` . To play back video content from 5 minutes ago, you should use the following request URL:

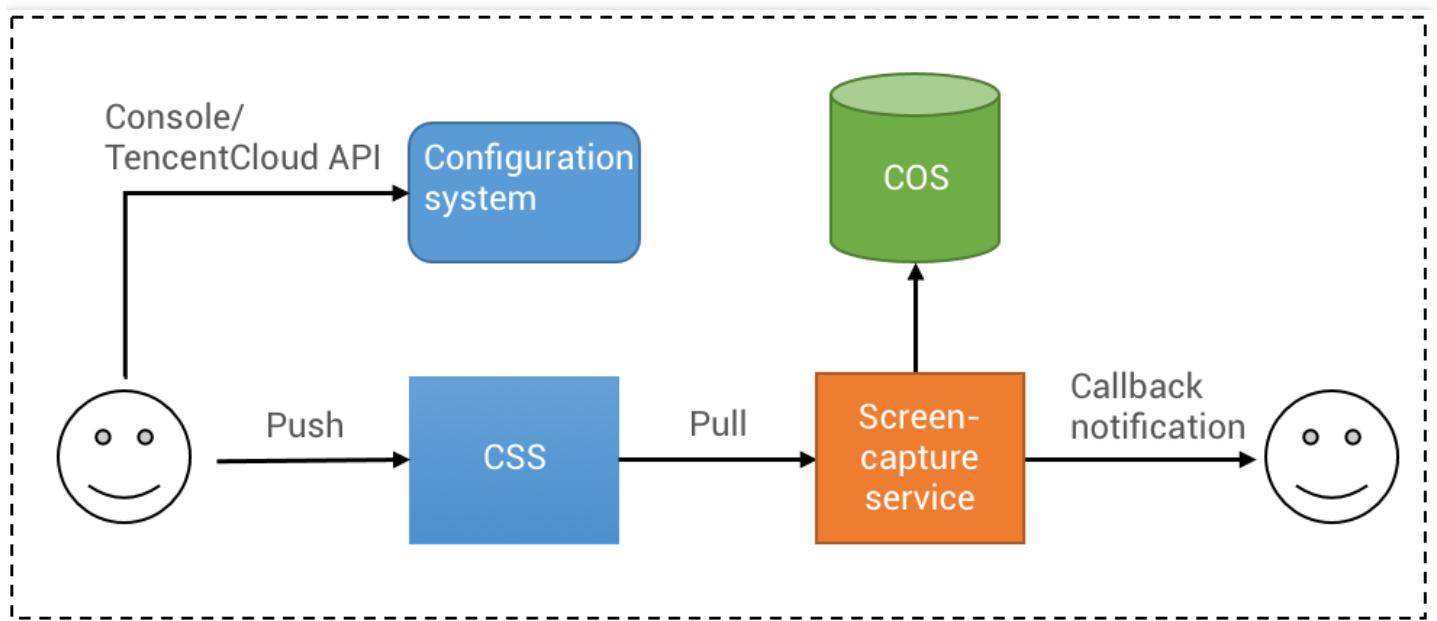
```
http://testtimeshift.com/timeshift/live/SLPUrIFzGPE/timeshift.m3u8?delay=300
```

Live Screenshot

Last updated : 2022-04-14 09:52:15

The live screenshot feature takes screenshots of a real-time live stream at regular intervals and generates images. You can get the screenshot information through the callback notification. These screenshots have various uses, such as porn detection and thumbnails.

Live Screenshot Process



Overall process:

1. Configure the live screenshot feature in the console or through TencentCloud API.
2. Start live push.
3. The screenshot service generates screenshot data according to the configuration and stores it in COS.
4. Information about the generated screenshot is returned in a callback.

Live Screenshot Configuration

Screenshot configuration method

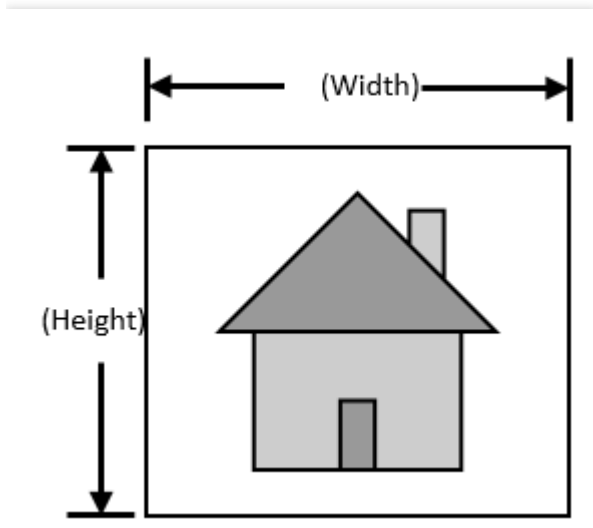
- [CSS API](#)
- **CSS console** > **Feature Configuration** > [Live Screenshot and Porn Detection](#)

Screencapturing interval configuration

You can specify the screencapturing frequency based on your business needs, i.e., the screencapturing interval (SnapshotInterval). The available range is between 5–300 seconds with a default interval of 10 seconds.

Screenshot width and height configuration

The screencapturing service supports taking screenshots by the specified width and height:



Note :

If you do not need to specify the width and height, the default screenshot width and height (set to 0) will be the width and height of the pushed video stream, and you can ignore the configuration instructions below and skip to the next section.

First, look at the following three concepts of width and height:

- Push width and height, i.e., the width and height of the live streaming video, which are set to (X, Y) in this document.
- Configured width and height, i.e., the width and height configured in the console/through the TencentCloud API, which are set to (W, H) in this document.
- Screenshot width and height, i.e., the width and height of the screenshot generated by the screencapturing service, which are set to (N, M) in this document.

The screencapturing service supports the following configurations:

- If the width and height are not set, then (W, H) = (0, 0) is used by default. The screenshot width and height are the same as the push width and height, i.e., (N, M) = (X, Y).

- If only the width W is set, then the screenshot width will be $N = W$, and the screenshot height is scaled proportionally, i.e., $M = N / X * Y$.
- If only the height H is set, then the screenshot height will be $M = H$, and the screenshot width is scaled proportionally, i.e., $N = M / Y * X$.
- If (W, H) are set at the same time, then the screenshot width and height are the same as the configured width and height, i.e., $(N, M) = (W, H)$.

The automatic swap of configured width and height is suitable for the following scenario:

- If W is set to be smaller than H , both W and H are greater than 0, and X is set to be greater than Y during the push, then the configured width is smaller than the height, but the push width is greater than the height.

In this case, if a screenshot is directly taken, it will be distorted. In order to avoid the distortion, the backend of the live screencapture service will automatically swap the values of W and H to ensure that the configured aspect ratio is consistent with that of the live stream.

Event Message Notification for Live Screencapture

For event message notification configuration, please see [Event Message Notification](#). The screencapturing callback notification is sent to the pre-configured receiving server through the HTTP POST protocol in JSON format.

Screencapturing callback fields

Field Name	Type	Description
event_type	int	Callback information type, which is always 200 for screencapturing callback
stream_id	string	Stream name
channel_id	string	Same as the stream name
create_time	int64	UNIX timestamp when the screenshot is generated
file_size	int	Screenshot file size in bytes
width	int	Screenshot width in pixels
height	int	Screenshot height in pixels
pic_url	string	Screenshot file path /path/name.jpg. For more information, please see Field details below
pic_full_url	string	Full screenshot URL. For more information, please see Field details below

Field Name	Type	Description
sign	string	Callback signature. For more information, please see Event Message Notification
t	int64	UNIX timestamp when the callback signature expires. For more information, please see Event Message Notification

Field details

- `pic_url` details:
- path: year-month-day
- name: live stream name-screenshot-hour-minute-second-widthxheight.jpg

Example:

```
/2018-12-17/stream_name-screenshot-19-06-59-640x352.jpg
```

This field can be used to put together a custom COS CDN domain name. If you do not need a CDN domain name, use `pic_full_url` directly.

- `pic_full_url` details:
- `http://COS domain name+pic_url`

Example:

```
http://testbucket-1234567890.cos.region.myqcloud.com/2018-12-17/stream_name-screenshot-19-06-59-640x352.jpg
```

Sample screencapturing callback

```
{
  "event_type": 200,
  "stream_id": "stream_name",
  "channel_id": "stream_name",
  "create_time": 1545030273,
  "file_size": 7520,
  "width": 640,
  "height": 352,
  "pic_url": "/2018-12-17/stream_name-screenshot-19-06-59-640x352.jpg",
  "pic_full_url": "http://testbucket-1234567890.cos.region.myqcloud.com/2018-12-17/stream_name-screenshot-19-06-59-640x352.jpg",
  "sign": "ca3e25e5dc17a6f9909a9ae7281e300d",
  "t": 1545030873
}
```

Live Porn Detection

Last updated : 2022-03-21 10:17:36

To enable porn detection during live streaming, you need to enable screencapturing either in the [CSS console](#) or via APIs. This document describes how to implement porn detection via APIs.

Enabling Porn Detection

As the porn detection feature is based on screencapturing, you have to enable screencapturing first before you can enable porn detection. The steps are as follows:

1. Create a screencapturing template with porn detection enabled

Call [CreateLiveSnapshotTemplate](#), setting `PornFlag` to `1` to create a screencapturing template with porn detection enabled.

2. Create a screencapturing rule

Call [CreateLiveSnapshotRule](#) to create a screencapturing rule, binding the ID of the screencapturing template created in step 1 with the target `AppId` , `DomainName` , `AppName` , and `StreamName` .

3. Start live streaming

After you create a screencapturing rule with porn detection enabled, the porn detection feature will be automatically enabled for new streams. If you want to enable porn detection for an ongoing stream, you need to stop and restart the stream.

Getting Porn Detection Result

After porn detection is enabled, you can configure a registered domain name in the porn detection callback template to receive callbacks of porn detection results.

Note :

By default, only questionable results will be called back.

1. Create a porn detection callback template

Call [CreateLiveCallbackTemplate](#), setting `PornCensorshipNotifyUrl` to your domain name to create a porn detection callback template.

2. Create a porn detection callback rule

Call [CreateLiveCallbackRule](#) to create a porn detection callback rule, binding the ID of the porn detection callback template created in step 1 to the target `AppId` , `DomainName` , and `AppName` .

3. Get the porn detection result

The CSS backend sends the porn detection result to your registered domain name through an HTTP POST request where the result is stored in the HTTP body in JSON format. You can determine whether the live streaming is pornographic by checking the `type` field.

Note :

We recommend you use the `type` of an image to determine whether it is pornographic. As the detection system cannot achieve 100% accuracy, there may be false positives or false negatives, and human confirmation is recommended.

The complete protocol is as follows:

Parameter	Required	Data Type	Description
streamId	No	String	Stream name
channelId	No	String	Channel ID
img	Yes	String	Link to the alerted image
type	Yes	Array	Categories of negative labels with the highest priority in the detection result. For details, see the description of <code>label</code> .
score	Yes	Array	Scores of <code>type</code>
ocrMsg	No	String	OCR result (if any)
suggestion	Yes	String	Suggestion. Valid values: <ul style="list-style-type: none">BlockReviewPass

Parameter	Required	Data Type	Description
label	Yes	String	Negative label with the highest priority in the detection result (<code>LabelResults</code>). This is the moderation result suggested by the model. We recommend you handle different types of violations and suggestions based on your business needs.
subLabel	Yes	String	Sub-label under the negative label with the highest priority in the detection result, such as porn - sexual acts. If no content is sub-labeled, this field will be empty.
labelResults	No	Array of LabelResult	Negative label hit details of the category model, including the detected porn content, ads, terrorism content, and politically sensitive content Note: This field may return <code>null</code> , indicating that no valid values can be obtained.
objectResults	No	Array of ObjectResult	Detection result of the object model, including label name, hit score, coordinates, scenario, and suggested operation regarding objects, advertising logos, QR codes, etc. For details, see the description of the data structure of <code>ObjectResults</code> . Note: This field may return <code>null</code> , indicating that no valid values can be obtained.
ocrResults	No	Array of OcrResult	OCR result, including text coordinates, recognized text, suggested operation, etc. For details, see the description of the data structure of <code>OcrResults</code> . Note: This field may return <code>null</code> , indicating that no valid values can be obtained.
libResults	No	Array of LibResult	Blocklist/Allowlist moderation result
screenshotTime	Yes	Number	Screenshot time
sendTime	Yes	Number	Time when the request was sent, in Unix timestamp format
stream_param	No	String	Push parameters
app	No	String	Push domain name
appid	No	Number	Application ID
appname	No	String	Push path

LabelResult

Hit result of the category model

Parameter	Type	Description
Scene	String	Scenario identified by the model, such as advertising, pornographic, and harmful
Suggestion	String	Operation suggested by the system for the current negative label. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Hit score of the label model
Details	Array of LabelDetailItem	Sub-label hit details of the category model

LabelDetailItem

Sub-label hit details of the category model

Parameter	Type	Description
Id	Integer	ID
Name	String	Sub-label name
Score	Integer	Sub-label score. Value range: 0-100

ObjectResult

Object detection result

Parameter	Type	Description
Scene	String	Object scenario identified, such as QR code, logo, and OCR

Parameter	Type	Description
Suggestion	String	Operation suggested by the system for the current negative label. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Sub-label hit score of the scenario model. Value range: 0-100
Names	Array of String	List of object names
Details	Array of ObjectDetail	Object detection details

ObjectDetail

Object detection details. When the detection scenario is object, advertising logo, or QR code, it returns the label name, label value, label score, and location information of the detection frame.

Parameter	Type	Description
Id	Integer	ID of the object identified
Name	String	Object label identified
Value	String	Value or content of the object label identified. For example, if the label is QR code (<code>QrCode</code>), this parameter is the URL of the QR code.
Score	Integer	Hit score of the object label. Value range: 0-100. For example, <code>QrCode 99</code> indicates a high likelihood that the content is a QR code.
Location	Location	Coordinates (of the top-left corner), dimensions, and rotation of the object detection frame

Location

Coordinates and other information of the detection frame

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
X	Float	Horizontal coordinate of the top-left corner
Y	Float	Vertical coordinate of the top-left corner
Width	Float	Width
Height	Float	Height
Rotate	Float	Rotation angle of the detection frame

OcrResult

OCR result

Parameter	Type	Description
Scene	String	Recognition scenario. Default value: OCR
Suggestion	String	Operation suggested by the system for the negative label with the highest priority. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Sub-label hit score of the scenario model. Value range: 0-100
Text	String	Text
Details	Array of OcrTextDetail	OCR details

OcrTextDetail

OCR details

Parameter	Type	Description
Text	String	Text recognized (up to 5,000 bytes)
label	String	Negative label in the detection result

Parameter	Type	Description
Keywords	Array of String	Keywords hit under the label
Score	Integer	Hit score of the label model. Value range: 0-100
Location	Location	OCR text coordinates

LibResult

Blocklist/Allowlist result

Parameter	Type	Description
Scene	String	Scenario recognition result of the model. Default value: Similar
Suggestion	String	Operation suggested by the system. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Recognition score of the image search model. Value range: 0-100
Details	Array of LibDetail	Blocklist/Allowlist details

LibDetail

Custom list or blocklist/allowlist details

Parameter	Type	Description
Id	Integer	ID
ImageId	String	Image ID
label	String	Negative label in the detection result
Tag	String	Custom label
Score	Integer	Model recognition score. Value range: 0-100

Sample callback message

```
{
  "ocrMsg": "",
  "type": [1],
  "socre": 99,
  "screenshotTime": 1610640000,
  "level": 0,
  "img": "http://1.1.1.1/download/porn/test.jpg",
  "abductionRisk": [],
  "faceDetails": [],
  "sendTime": 1615859827,
  "suggestion": "Block",
  "label": "Porn",
  "subLabel": "PornHigh",
  "labelResults": [{
    "HitFlag": 0,
    "Scene": "Illegal",
    "Suggestion": "Pass",
    "Label": "Normal",
    "SubLabel": "",
    "Score": 0,
    "Details": []
  }, {
    "HitFlag": 1,
    "Scene": "Porn",
    "Suggestion": "Block",
    "Label": "Porn",
    "SubLabel": "PornHigh",
    "Score": 99,
    "Details": [{
      "Id": 0,
      "Name": "PornHigh",
      "Score": 99
    }, {
      "Id": 1,
      "Name": "WomenChest",
      "Score": 99
    }
  ]
}, {
  "HitFlag": 0,
  "Scene": "Sexy",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Details": []
}
```

```
}, {
  "HitFlag": 0,
  "Scene": "Terror",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Details": []
}],
"objectResults": [{
  "HitFlag": 0,
  "Scene": "QrCode",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Names": [],
  "Details": []
}, {
  "HitFlag": 0,
  "Scene": "MapRecognition",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Names": [],
  "Details": []
}, {
  "HitFlag": 0,
  "Scene": "PolityFace",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Names": [],
  "Details": []
}],
"ocrResults": [{
  "HitFlag": 0,
  "Scene": "OCR",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Text": "",
  "Details": []
}],
}
```

```
"streamId": "teststream",
"channelId": "teststream",
"stream_param": "txSecret=40f38f69f574fd51126c421a3d96c374&txTime=5DEBEC80",
"app": "5000.myqcloud.com",
"appname": "live",
"appid": 10000,
"event_type": 317,
"sign": "ac920c3e66*****78cf1b5de2c63",
"t": 1615860427
}
```

Disabling Porn Detection

You can disable porn detection by deleting the screencapturing rule or modifying the screencapturing template. Both the deletion and modification are effective only for new live streams. If you want to disable porn detection for an ongoing live stream, you have to stop and restart the stream.

1. Delete the screencapturing rule

Call [DeleteLiveSnapshotRule](#), passing in the `DomainName` , `AppName` , and `StreamName` bound to the screencapturing template ID to delete the screencapturing rule.

2. Modify the screencapturing template

Call [ModifyLiveSnapshotTemplate](#), setting `PornFlag` to `0` .

AV1 Encoding

Last updated : 2022-10-14 16:26:04

AOMedia Video 1 (AV1) is a free, open source video coding format. It encodes videos at a bitrate 30%+ lower than H.265 (HEVC) does while delivering the same video quality. This means that with the same bandwidth, AV 1-encoded videos have higher quality than H.265-encoded videos. This document shows you how to encode videos using AV1 and how to play AV1-encoded videos.

How to Use AV1

Prerequisites

- You have [signed up for a Tencent Cloud account](#).
- You have activated CSS and added [a playback domain and a push domain](#).

Step 1. Create a transcoding template

1. Log in to the CSS console and select **Feature Configuration** > [Live Transcoding](#) on the left sidebar.
2. Click **Create Transcoding Template**, select **Standard Transcoding** or **Top Speed Codec Transcoding** as the transcoding type, and expand **Advanced Configuration**.

3. Select **AV1** as the codec.

Live Transcoding

Transcoding Type

Standard Transcoding

Top Speed Codec TranscodingAudio-only Transcoding

Template Name *

Enter 1-10 characters

Supports only letters or a combination of letters and digits

Template Description

Please describe template

Only supports letters, digits, underscores, and dashes

Video quality

SmoothSDHD

Video Bitrate *

101-8000

kbps

Video Resolution *

Set the short s

Value range: 0-3,000

px

Enter a multiple of 2. The other side will be scaled proportionally according to resolution.

Short side

Long side

Long side

Short side

DRM encryption

☒

Supports Widevine, FairPlay, and NormalAES DRM encryption for HLS playback. For FairPlay DRM, you need to upload the certificate you obtain from Apple to your player. [Obtaining a FairPlay certificate](#)

Before you enable DRM encryption, please go to [DRM Management](#) to configure the key information.

Codec

☐ Original codec☐ H.264☐ H.265☒ AV1

4. Click **Save**.

Step 2. Bind a domain

Select the transcoding template created and click **Bind Domain Name**. In the pop-up window, select a **playback domain** and click **Confirm**.

Bind Domain Name ✕

Domain name binding takes effect in about 10 minutes after the configuration. Please add "_transcoding template name" after the StreamName to generate a new URL for playing back the transcoded stream.

Transcoding Template

Playback Domain Delete

[Add](#)

Confirm Cancel

Step 3. Generate a playback URL

Select **Address Generator** on the left sidebar. Select the playback domain bound in step 2 and the transcoding template created in [step 1](#), and click **Generate Address** to generate a playback URL.

Step 4. Play AV1-encoded videos

Play the AV1-encoded videos with a player that supports AV1 using the playback URL generated in step 3. You can either use a third-party player that supports AV1 or rebuild your own player.

- **Third-party players that support AV1**

- **App**

- [ExoPlayer](#) (use `libgav1`)
 - [ijkplayer](#) FFmpeg (update FFmpeg and integrate [dav1d](#))

- **Web**

- [dash.js](#). The player supports AV1, but whether AV1 videos can be decoded depends on the browser. Chrome supports AV1 decoding.
 - [shaka-player](#). The player supports AV1, but whether AV1 videos can be decoded depends on the browser. Chrome supports AV1 decoding.

- **PC**

- VLC for [Windows](#) and [macOS](#) support AV1 in FLV and HEVC in FLV.

- **Rebuilding your own player**

If your player cannot play AV1 videos, we can [help](#) you rebuild your player to support AV1.

Stream Mix

Last updated : 2021-12-02 16:36:40

CSS provides live stream mix feature, which can synchronously mix multiple streams of input sources into a new stream based on the configured stream mix layout for interactive live streaming. In addition, the stream mix feature has been connected to TencentCloud API 3.0. For more information, please see [CreateCommonMixStream](#). This document uses examples to describe how to implement live stream mix in different scenarios.

Notes

- Using stream mix will incur transcoding fees. For details, please see [Live Transcoding \(Watermarking, Stream Mixing\)](#).
- To use the mixing and cropping feature, the value of the cropping parameter cannot exceed the value of input stream parameter.

Supported Features

- Up to **16** concurrent streams can be mixed.
- Up to **5** types of input sources (audio and video, pure audio, pure video, image, and canvas) can be mixed.
- Mixed streams can be output as a new stream.
- Cropping and watermarking are supported.
- Template configuration is supported.
- Recording based on stream mix is supported.
- Automatic stream mix is supported.
- Stream mix types and positions can be switched in real time.
- Stream mix can be started/canceled seamlessly.

Common Layout Templates

Common templates include 10, 30, 40, 310, 390, 410, 510, and 610. When using them, you do not need to enter the position and length/width parameters of the input stream, and **the original image will be auto-scaled**. You only need to pass in the template ID.

Figures of common layout templates:

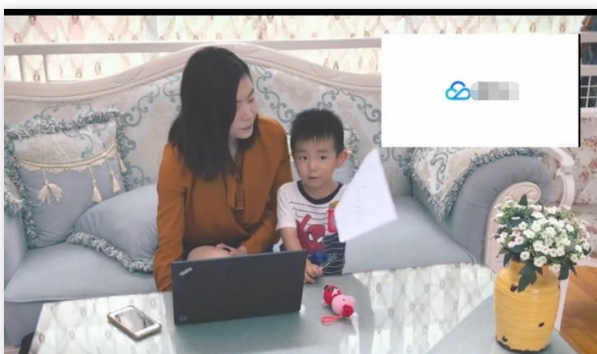
Template 10



Template 30



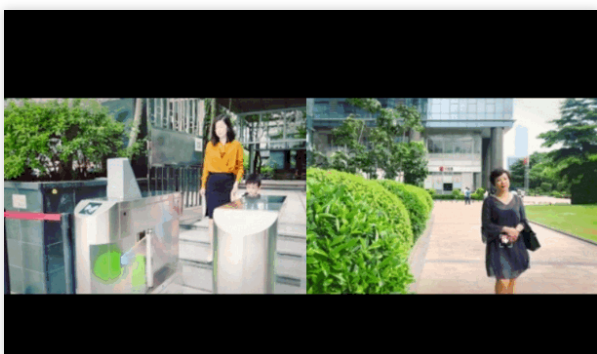
Template 40



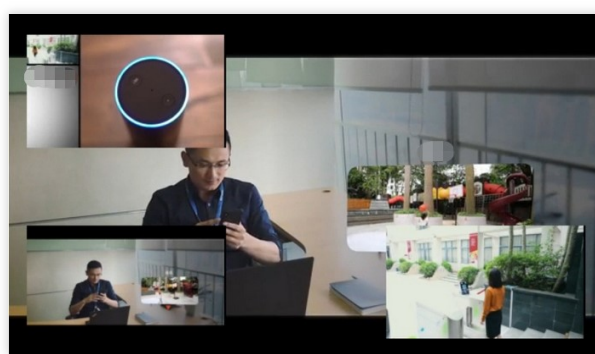
Template 310



Template 390



Template 410

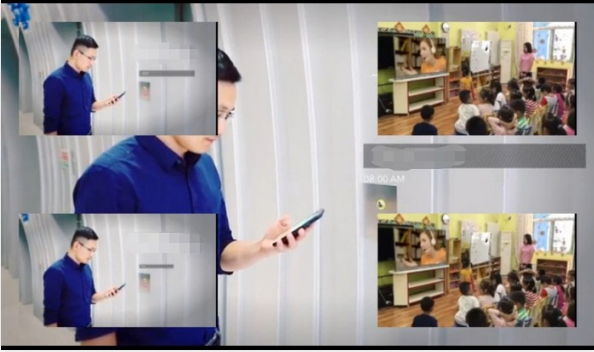



Template 510



Template 610



Template 10	Template 30
	

Creating Stream Mix Session

Parameters

For more information, please see [CreateCommonMixStream](#).

Scenario 1. Applying for stream mix - using template 20

This example shows you how to use a preset stream mix template to mix streams.

Sample input code

```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&MixStreamTemplateId=20
&OutputParams.OutputStreamName=test_stream1
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.1.InputStreamName=test_stream2
&InputStreamList.1.LayoutParams.ImageLayer=2
&<Common request parameters>
```

Sample output code

```
{
  "Response": {
    "RequestId": "e8fa8015-0892-40d5-95c4-12a4bc06ed31"
  }
}
```


Stream mix effect for mic connect



Scenario 2. Applying for stream mix - using template 390

This example shows you how to use a preset stream mix template to mix streams.

Sample input code

```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&MixStreamTemplateId=390
&OutputParams.OutputStreamName=test_stream2
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.0.LayoutParams.InputType=3
&InputStreamList.0.LayoutParams.ImageWidth=1920 (canvas width)
&InputStreamList.0.LayoutParams.ImageHeight=1080 (canvas height)
&InputStreamList.0.LayoutParams.Color=0x000000
&InputStreamList.1.InputStreamName=test_stream2
&InputStreamList.1.LayoutParams.ImageLayer=2
&InputStreamList.2.InputStreamName=test_stream3
&InputStreamList.2.LayoutParams.ImageLayer=3
&<Common request parameters>
```

Sample output code

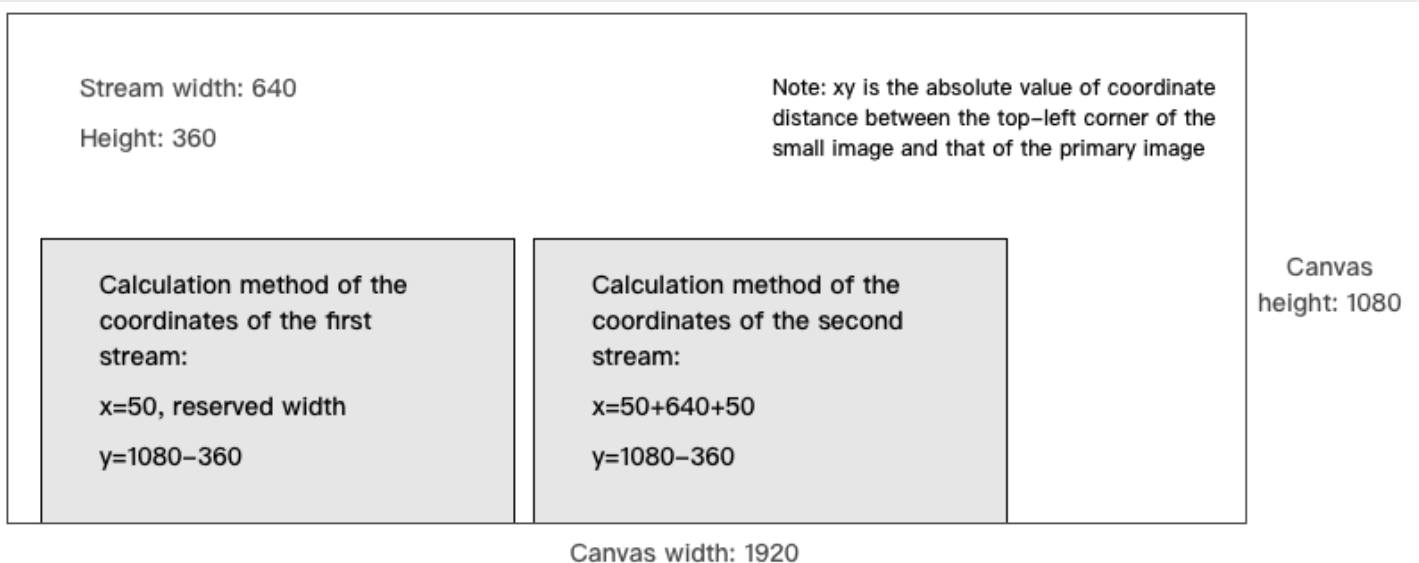
```
{
  "Response": {
    "RequestId": "9d8d5837-2273-4936-8661-781aeab9bc9c"
  }
}
```

Stream mix effect for host competition



Scenario 3. Custom stream mix

Use the custom layout, where the position parameters `LocationX` and `LocationY` represent the absolute pixel distance between the top-left corner of the small image and that of the background image.



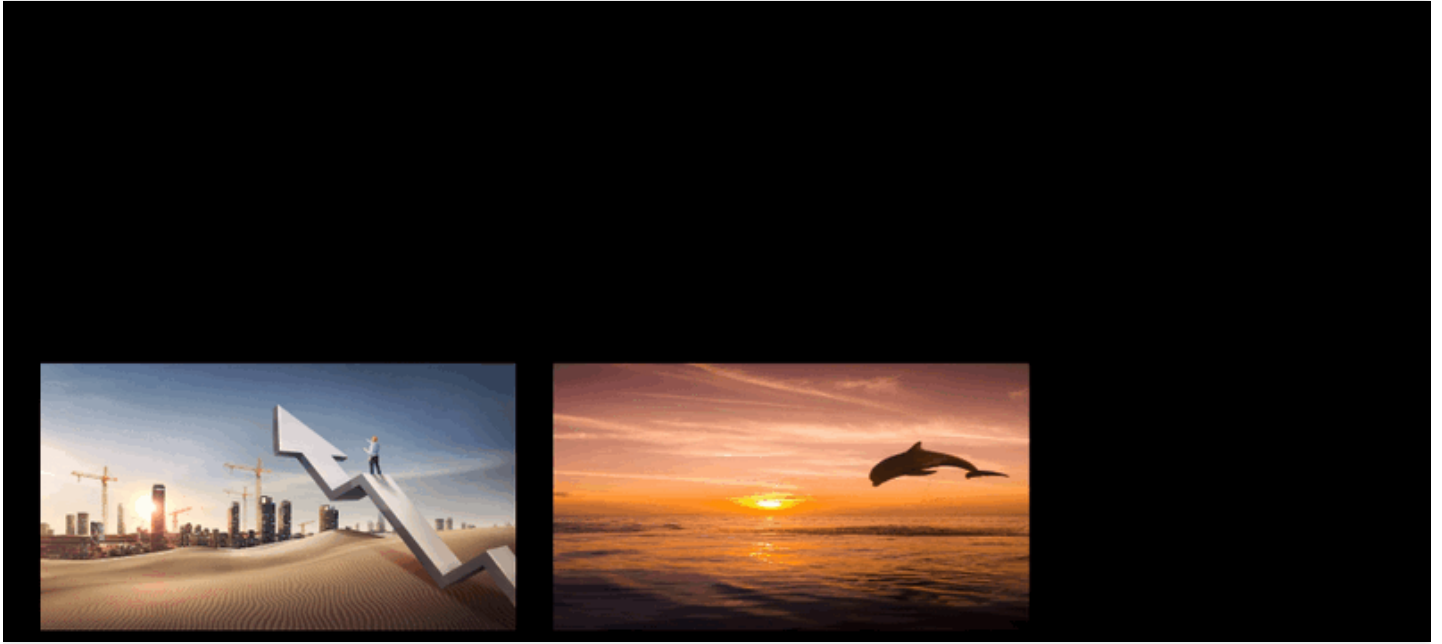
Sample input code

```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&OutputParams.OutputStreamName=test_stream2
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.0.LayoutParams.InputType=3
&InputStreamList.0.LayoutParams.ImageWidth = 1920
&InputStreamList.0.LayoutParams.ImageHeight= 1080
&InputStreamList.0.LayoutParams.Color=0x000000
&InputStreamList.1.InputStreamName=test_stream2
&InputStreamList.1.LayoutParams.ImageLayer=2
&InputStreamList.1.LayoutParams.ImageWidth = 640
&InputStreamList.1.LayoutParams.ImageHeight= 360
&InputStreamList.1.LayoutParams.LocationX= 50
&InputStreamList.1.LayoutParams.LocationY= 720
&InputStreamList.2.InputStreamName=test_stream3
&InputStreamList.2.LayoutParams.ImageLayer=3
&InputStreamList.2.LayoutParams.ImageWidth = 640
&InputStreamList.2.LayoutParams.ImageHeight= 360
&InputStreamList.2.LayoutParams.LocationX= 740
&InputStreamList.2.LayoutParams.LocationY= 720
&<Common request parameters>
```

Sample output code

```
{
  "Response": {
    "RequestId": "8c443359-ba07-4b81-add8-a6ff54f9bf54"
  }
}
```

Custom stream mix effect



Canceling Stream Mix

Parameters

For more information, please see [CancelCommonMixStream](#).

Examples

This example shows you how to cancel a stream mix by session ID.

Sample input code

```
https://live.tencentcloudapi.com/?Action=CancelCommonMixStream
&MixStreamSessionId=test_room
```

Sample output code

```
{
  "Response": {
    "RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
  }
}
```

Note :

- After applying for canceling stream mix, wait at least for 5s before canceling it.
- After canceling the stream mix, wait at least for half a minute before you can apply for stream mix using the same session ID.

Error Codes

For stream mix API 3.0, most common error codes have been transformed into the style of [API 3.0 error code](#).

However, some error codes remain unchanged, which will be provided in the format of `err_code [$code],msg [$message]` in `Message` and prompted as an `InvalidParameter` error. The causes of specific codes are as detailed below:

Error Code	Reason	Troubleshooting
-1	An error occurred while parsing the input parameters	<ul style="list-style-type: none">• Check whether the JSON format of the request body is correct.• Check whether <code>InputStreamList</code> is empty.
-2	Incorrect input parameter	Check whether the image parameter is too large.
-3	The number of streams is incorrect	Check whether the number of input streams is within the range of [1,16].
-4	Incorrect stream parameter	<ul style="list-style-type: none">• Check whether the length/width of the input/output stream are within the range of (0,3000).• Check whether the number of input streams is within the range of (0,16].• Check whether the input stream carries <code>LayoutParams</code>.• Check whether <code>InputType</code> is supported (valid values: 0, 2, 3, 4, 5).• Check whether the stream ID length is within the range of (1,80).
-11	Layer error	<ul style="list-style-type: none">• Check whether the number of layers is the same as the number of input streams.• Check whether the layer ID is duplicate.• Check whether the layer ID is within the range of (0,16].

Error Code	Reason	Troubleshooting
-20	The input parameter does not match the API	<ul style="list-style-type: none"> Check whether the number of input streams matches the template ID. Check whether the color parameter is correct.
-21	The number of input streams for stream mix is incorrect	Check whether there are at least two input streams.
-28	Failed to get the background length/width	<ul style="list-style-type: none"> Check whether the canvas length and width are set when setting the canvas. Check whether the background stream exists (stream mix needs to start 5 seconds after push starts).
-29	Incorrect cropping parameter	Check whether the cropping position is out of the stream length/width range.
-33	Incorrect watermark image ID	Check whether the input image ID is set.
-34	Failed to get the URL of the watermark image	Check whether the image has been successfully uploaded and whether the URL has been generated.
-111	The `OutputStreamName` parameter does not match `OutputStreamType`	<ul style="list-style-type: none"> If `OutputStreamType` is set to `0`, `OutputStreamName` should be in `InputStreamList`. If `OutputStreamType` is set to `1`, `OutputStreamName` should not be in `InputStreamList`.
-300	The output stream ID has already been used	Check whether the current output stream belongs to another stream mix task.
-505	Failed to find the input stream in `upload`	Check whether stream mix is initiated 5 seconds after the push and whether the stream can be played back.
-507	Failed to query the stream length/width parameters	<ul style="list-style-type: none"> Check whether the canvas length and width are set. Check whether the push succeeds. We recommend you start stream mix 5 seconds after push starts.
-508	Incorrect output stream ID	Check whether different output stream IDs are used by the same `MixStreamSessionId`.
-10031	Failed to trigger stream mix	We recommend you start stream mix 5 seconds after push starts.

Error Code	Reason	Troubleshooting
-30300 -31001 -31002	The `sessionId` does not exist when stream mix is canceled	Check whether the `MixStreamSessionId` exists.
-31003	The output stream ID does not match that in `session`	Check the output stream ID entered when stream mix is canceled.
-31004	The output stream bitrate is invalid	Check whether the output stream bitrate is within the range of [1,50000].
Others	For other errors, please contact customer service for assistance.	-

FAQs

- [How do I ensure that the input streams can be auto scaled with no black bars in the video image during stream mix?](#)
- [What should I do if error code -505 is returned for stream mix after push?](#)
- [What will happen if stream mix is not canceled after it is applied for?](#)
- [Why is the assistant host's video image in the mixed stream not in the expected position?](#)

Note :

For more FAQs about stream mix, please see [On-cloud Stream Mix](#).

Video Content Protection

Last updated : 2022-11-23 11:10:05

Glossary

- **You:** Any person or organization that uses CSS and owns a Tencent Cloud account.
- **User:** The end users of a CSS customer, mostly audience members.

Aspects of Live Content Protection

Aspect	Description
Hotlink protection	Without hotlink protection, anyone can use your playback URL. Many playback URLs use similar formats, for example, <code>Protocol:\\Playback domain\\AppName\\Stream ID</code> . If others figure out your URL format, they may splice playback URLs to play all your streams. This exposes your content, and you have to pay for the traffic consumed by unauthorized playback.
Playback access control	In some scenarios, only users that meet certain conditions should be allowed to view live content, for example, subscribers, logged-in users, or users who have purchased your product.
Playback restrictions for copyrighted content	For copyrighted content, for example, content produced by a film studio or TV network, playback must be limited to safe and trusted environments.
Confidentiality	Only specific viewers should be allowed to view confidential content.

The list above describes some of the different aspects that need to be considered when protecting your live content. The next part of this document introduces CSS' content protection solutions, starting from the simplest to the most sophisticated. CSS ensures content security using two main methods. One is restricting the use and distribution of playback URLs, and the other is content encryption. The former is relatively easy to implement, but the latter depends on the player and may have requirements for hardware and the operating system as well.

Below are some commonly used content protection solutions for live streaming and their implementation methods.

Referer authentication

- **Use case:** You can use this solution if you do not have high requirements for content security and only want to limit playback to some degree.
- **Implementation:** Log in to the CSS console and select **Domain Management** on the left sidebar. Click your playback domain, select the **Access Control** tab, toggle on **Referer**, and enter the required information in the pop-up window.
- **Pros:** Easy to implement (can be configured in the console)
- **Cons:** The referer field in a playback request URL can be modified and forged. Others will be able to circumvent your restrictions if they know your referer settings.

IP blocklist/allowlist

- **Use case:** You can use this solution if you want to allow access for or block specific public IP addresses.
- **Implementation:** Log in to the CSS console and select **Domain Management** on the left sidebar. Click your playback domain and select the **Access Control** tab. In the **IP allowlist/blocklist** area, toggle on **Status**, and enter the IP addresses you want to allow or block in the pop-up window.
- **Pros:** Easy to implement (can be configured in the console); others cannot forge their IP addresses and therefore cannot circumvent the restrictions.
- **Cons:** The public IP addresses of the viewers you want to allow or block are not easy to obtain and are subject to changes.

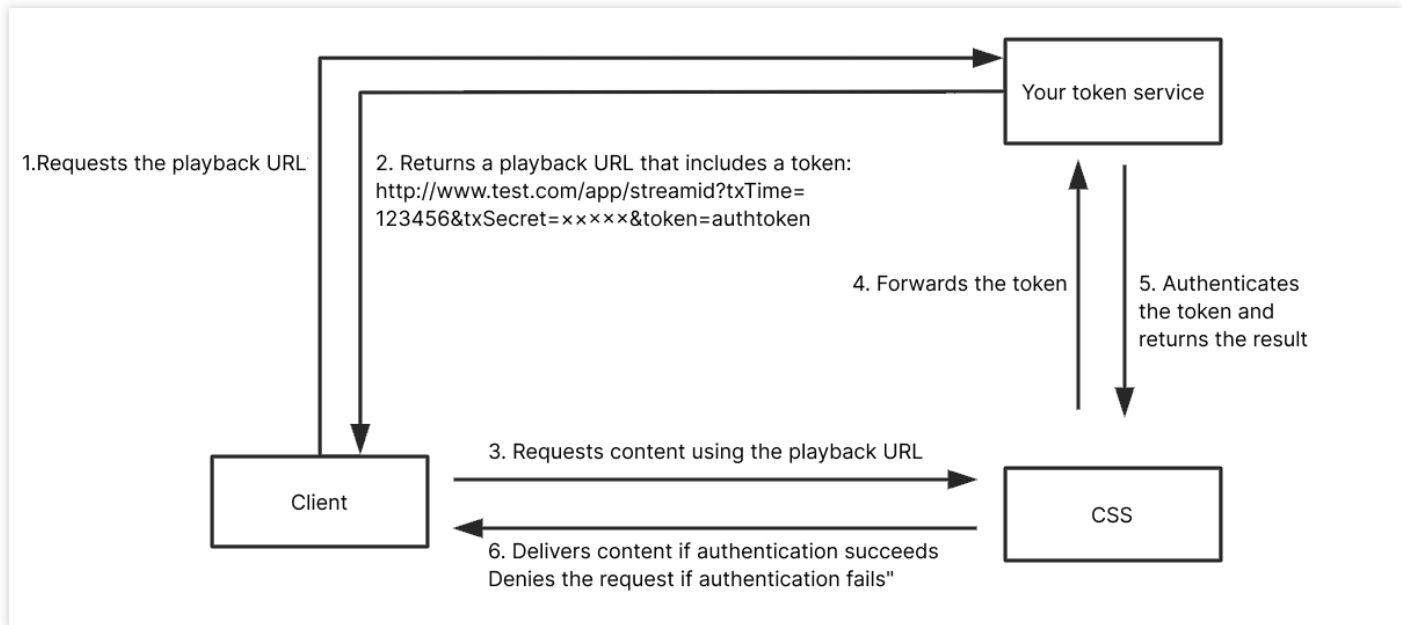
Key authentication

- **Use case:** You can use this solution if you want to prevent hotlinking.
- **Implementation:** Log in to the CSS console and select **Domain Management** on the left sidebar. Click your playback domain and select the **Access Control** tab. In the **Key Authentication** area, toggle on **Playback Authentication**, and enter the key information in the pop-up window. After configuration, a playback request URL will include `txTime` and `txSecret`, and Tencent Cloud will use the key to decrypt `txTime` and authenticate the request. For more information, see [Hotlink Protection URL Calculation](#).
- **Pros:** Easy to implement (you only need to enable key authentication in the console and generate `txTime` and `txSecret` as instructed in the above document)
- **Cons:** Before a playback URL expires, anyone who has the URL can play your content. This makes hotlinking possible. The longer the validity period, the more likely your content will be hotlinked. However, if you set a short validity period, the playback URL may have already expired by the time viewers get it. Also, if playback is interrupted due to network fluctuations or other reasons, to resume playback, viewers may need to obtain a new playback URL. The recommended validity period for a playback URL is 24 hours.

Key + Remote authentication

To prevent hotlinking caused by viewers playing your live streams from unauthorized channels, you can perform additional authentication using a custom token on top of key authentication.

- **Use case:** You can use this solution if you want to limit playback to viewers that meet certain conditions, for example, subscribers or logged-in users.
- **How it works:** After authenticating `txSecret`, Tencent Cloud will call a server API to forward the playback request to your token service. Your token service will authenticate the token in the request and return the result to Tencent Cloud. This allows you to determine whether to allow a playback request.

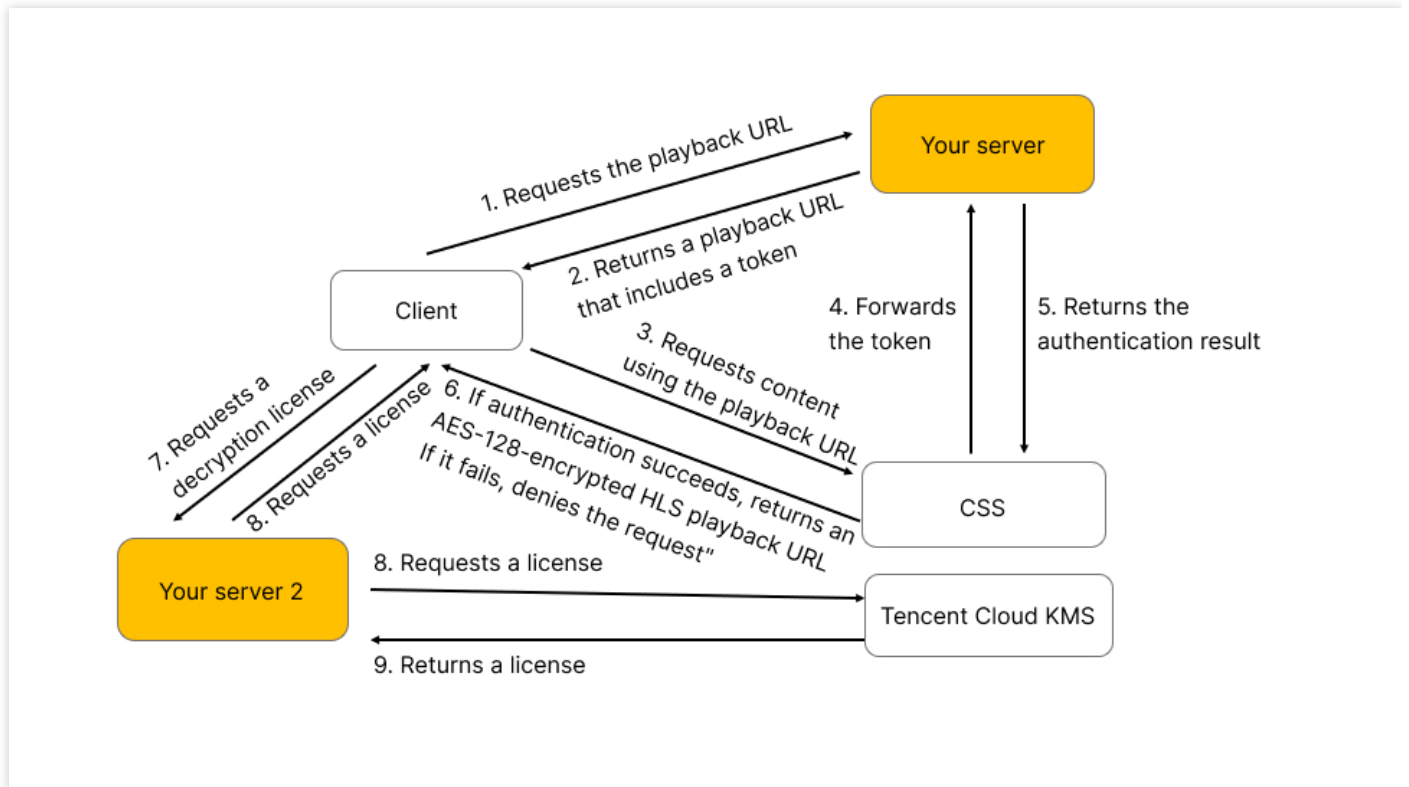


- A request is sent to your server (token service) for the playback URL.
 - Your server will validate the request and send the playback URL. The URL will include Tencent Cloud's `txSecret` and your token. We recommend you set a short validity period for `txSecret`, for example, five minutes.
 - Viewers will use the playback URL to request live content from CSS.
 - CSS will authenticate `txSecret` and forward the request, which includes the token, to your token service.
 - Your token service will authenticate the token in the request and return the result. The status code 200 indicates the request is allowed. Other codes indicate the request is denied.
 - CSS determines whether to deliver content to a client based on the result.
- **Implementation:** Enable key authentication and remote authentication. For detailed directions, please contact sales or submit a ticket.
 - **Pros:** You can determine what kind of users can view your content.
 - **Cons:** You need to develop your own token distribution service. What's more, because live streams are not encrypted, viewers with access to your content can record the streams or forward them in real time. Hackers can also steal your content.

Key authentication + Remote authentication + AES-128 encryption

In addition to key and remote token authentication, you can also encrypt HLS TS segments using the AES-128 algorithm.

- **Use case:** You can use this solution if you use the HLS protocol for playback and want to prevent hackers from stealing your content to protect the copyright of your content or keep your content confidential or private.
- **How it works:** HLS TS segments are encrypted using the AES-128 method.



- **Implementation:** Enable key authentication and remote authentication. For detailed directions, please contact sales or submit a ticket.
- **Pros:** This solution is easy to implement. The combination of key and remote token authentication offers extra protection. What's more, because AES-128 is a standard encryption method for HLS, any player that supports HLS also supports AES-128. The decryption capability is built into players. There are also well-established key management services for this solution.
- **Cons:** This solution is only applicable to HLS playback.

DRM scheme

You can encrypt your content using DRM solutions recognized in and outside China, such as Apple's FairPlay DRM, Google's Widevine DRM, and ChinaDRM. To use these DRM solutions, you need to request a certificate, which is used to manage identity information and the key used to encrypt/decrypt content, as well as set limits for the use of keys.

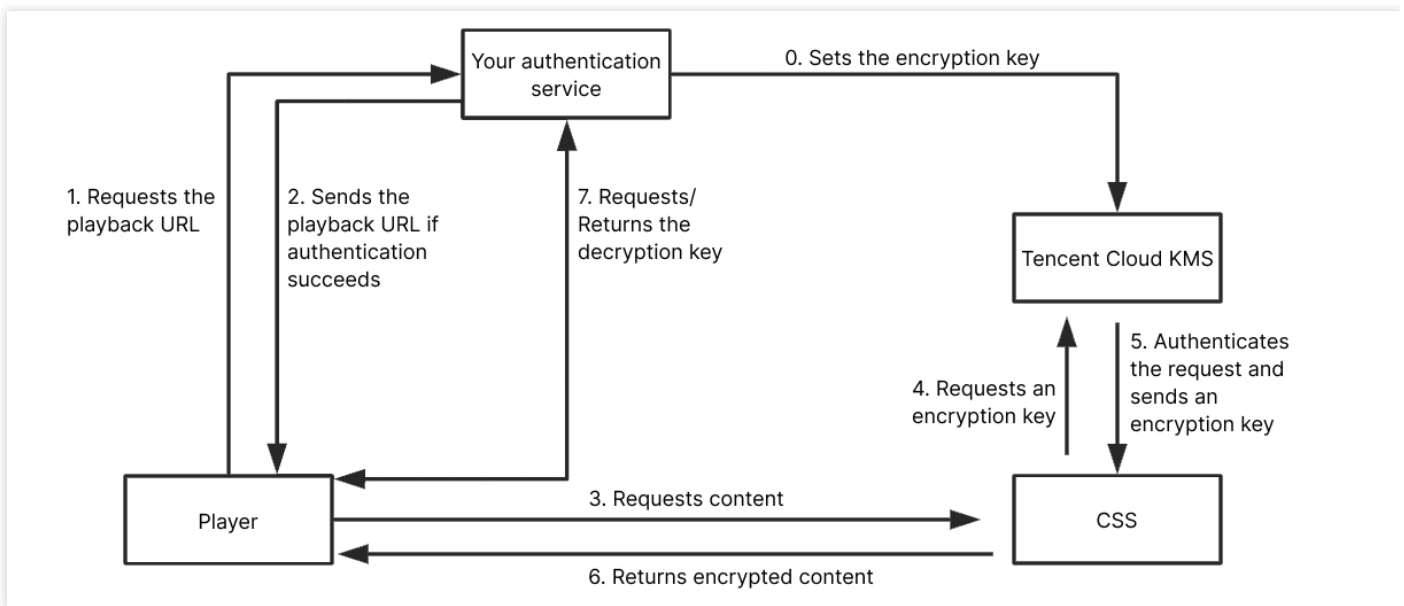
- **Use case:** You can choose this scheme if the copyright owner specifically requires that their content be encrypted using recognized DRM solutions.
- **How it works:** Currently, CSS only supports Apple's FairPlay DRM and Google's Widevine DRM. More solutions will be supported in the future.
- **Implementation:** For detailed directions, please contact sales or submit a ticket.
- **Pros:** The protection level is relatively high because decryption relies on the operating system or hardware. What's more, the solutions offer flexible access control. For example, you can limit playback to the first 10 minutes of a video.
- **Cons:** The solutions are only applicable to HLS or DASH playback. The implementation process is somewhat complicated. You need to request a certificate and integrate a DRM component into your player. What's more, the solutions have limited browser support. FairPlay supports only HLS and playback is only possible on iOS and macOS. Widevine may support browsers that are not based on Chromium, such as Firefox, but a CDM module needs to be loaded separately, which compromises playback smoothness. Parts of the implementation process (license requesting and encryption/decryption) are carried out in black box, making it difficult to debug and improve compatibility.

Tencent Cloud's proprietary encryption scheme

Most live streams with requirements for privacy or content security do not really need hardware-based protection. Nor is the complicated certificate distribution and authentication process necessary. Given this, we offer a content protection solution for FLV playback.

- **Use case:** You can use this solution if your live streams are in FLV format and you want to encrypt them to prevent hackers from stealing your content.
- **How it works:** You set an encryption key. CSS uses the key to encrypt your streams. When a viewer requests to play your streams, you authenticate the request and distribute a decryption key to the client. This process is similar to the AES-128 encryption/decryption process.

The process is as follows:





- **Implementation:** For detailed directions, please contact sales or submit a ticket.
- **Pros:** You have control over the entire process, and there are key management services and encryption/decryption tools to support this solution. Tencent Cloud's Player SDK is easy to integrate.
- **Cons:** You need to integrate an SDK into your project in order to use this solution. What's more, this solution does not work for browsers. You need to use your own player.

Comparison

Solution	How It Works	Features	Cons	Implementation	Recommended
Referer authentication	Authenticates the referer field in an HTTP request	Quick and easy to implement	The referer content can be forged.	Very simple	☆☆
IP blocklist/allowlist	Allows or blocks specific IP addresses	Quick and easy to implement	The IP addresses of viewers are difficult to obtain and are subject to changes.	Very simple	☆☆☆

Solution	How It Works	Features	Cons	Implementation	Recommended
Key authentication	Encrypts and authenticates a playback URL	Quick and easy to implement	A long validity period for `txSecret` may result in hotlinking, while a short validity period means viewers may have to obtain a new playback URL frequently.	Very simple	★★★★
Key authentication + Remote authentication	Playback URLs are encrypted and both CSS and your server authenticate the URLs.	Easy to implement and relatively strong protection against hotlinking, but you need to develop your own token service.	-	Simple	★★★★★
HLS encryption	Encrypts playback URLs as well as live streams	Easy to implement, but you need to develop your own token service.	Only applicable to HLS, not FLV	Simple	★★★★

Solution	How It Works	Features	Cons	Implementation	Recommended
DRM scheme	Encrypts live streams (you can also encrypt playback URLs); supports software- and hardware-based protection.	Somewhat complicated. You need to develop your own player and there may be compatibility issues.	Difficult to implement, with compatibility issues (different solutions are required for iOS and Android).	Complicated	
Proprietary encryption scheme	Encrypts content as well as playback URLs	Somewhat complicated. You need to develop your own player, but you have control over the entire process.	You need to comply with Tencent Cloud's communication protocol. This solution cannot be used on browsers.	Complicated	

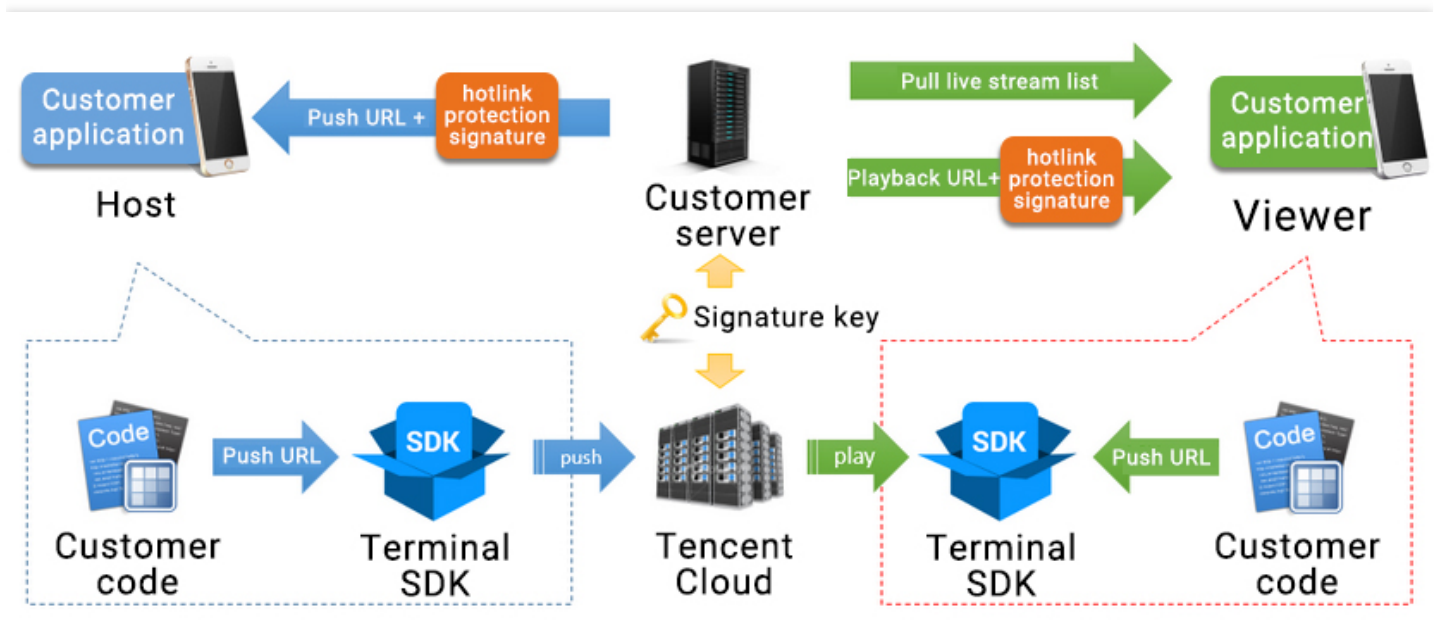
Hotlink Protection URL Calculation

Last updated : 2021-10-25 12:48:24

Security hotlink protection refers to the `txSecret` field in the push and playback URLs, which is used to prevent attackers from forging your backend for push URL generation or stealing your playback address for illegal profit.

How It Works

In order to prevent an attacker from forging your server to generate push and playback URLs, you can configure the hotlink protection encryption key in the CSS Console as shown below (do not disclose this key), so that the attacker cannot easily fake valid push and playback URLs, as shown in the figure below.



Calculation Process

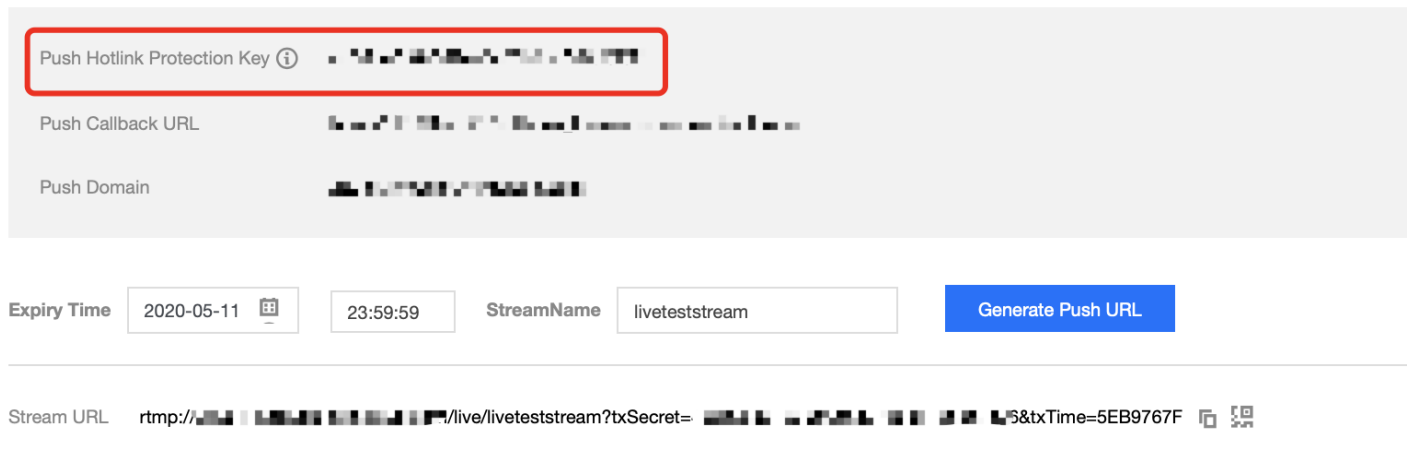
Step 1. Authentication key

First, you need to configure an encryption key in the console, which is used to generate the hotlink protection signature on your server. Since Tencent Cloud holds the same key as you do, it can decrypt and confirm the signature you generate.

An encryption key can be either a push hotlink protection key or a playback hotlink protection key. The former is used to generate a push hotlink protection URL, while the latter a playback hotlink protection URL. Go to **CSS Console** >

Domain Management, click a domain name or **Manage**, and select **Push Configuration** to configure a push hotlink protection key, as shown in the figure below.

Generate Push URL



Push Hotlink Protection Key ⓘ [Redacted]

Push Callback URL [Redacted]

Push Domain [Redacted]

Expiry Time 2020-05-11 23:59:59 StreamName liveteststream Generate Push URL

Stream URL rtmp://[Redacted]/live/liveteststream?txSecret=[Redacted]&txTime=5EB9767F

For more information on the playback hotlink protection key, please see [How can I enable playback hotlink protection?](#).

Step 2. Generate txTime

The plaintext in the signature is `txTime`, which is the validity period of the link. For example, if the current time is 2018-12-29 11:13:45, and the URL to be generated is expected to be valid for 3 hours, then `txTime` should be set to 2018-12-29 14:13:45.

However, it is obviously inappropriate to put such a long string of time in the URL. In actual use, we convert 2018-12-29 14:13:45 to a UNIX timestamp first, i.e., 1546064025 (you can call a time function in your programming language for conversion and processing), and then convert it to a hexadecimal string to further reduce the character length, i.e., `txTime = 1546064025 (decimal) = 5C271099 (hexadecimal)`. Using a decimal string is also supported.

Note :

- The actual end time will be `txTime` plus authentication validity period. Modifying authentication validity period will not affect the generation of URL.
- The expiration time should be neither too early nor too late:
 - If it is too early, when the host encounters network jitters during live streaming, the push will not be resumed because the push URL will have expired.
 - If it is too late, there are risks of unauthorized push.

Step 3. Generate txSecret

The generation method of `txSecret` is `MD5(KEY + StreamName + txTime)`. `KEY` is the encryption key configured in step 1. `StreamName`, also known as stream ID, is `Test` in this example (we recommend using a random number or a user ID). `txTime` is the `5C271099` calculated in last step, and MD5 is a standard irreversible one-way MD5 hash algorithm.

For example:

```
KEY is e12c46f2612d5106e2034781ab261ca3
Then txSecret = MD5(e12c46f2612d5106e2034781ab261ca3test5C271099) = f85a2ab363fe4deaffef9754d79da6fe
```

Step 4. Construct a hotlink protection URL

A push URL that conforms to the Tencent Cloud standards consists of the following four parts:

`rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(time)`

Domain name Application name Stream ID Authentication key

We can create a standard URL using `txTime` (which informs Tencent Cloud of the expiration time of a push/playback URL), `txSecret` (that only Tencent Cloud can decrypt and verify), `StreamName`, and a push domain name (e.g. "livepush.tcloud.com"). In this example, the push URL is:

```
rtmp://livepush.tcloud.com/live/test?txSecret=f85a2ab363fe4deaffef9754d79da6fe&txTime=5C271099
```

Sample Push Code

Go to **CSS Console > Domain Management**, select a pre-configured push domain name, click **Manage > Push Configuration** to display the **Push Address Sample Code** (for both PHP and Java) that demonstrates how to generate a hotlink protection address. For details, please see [Push Configuration](#).

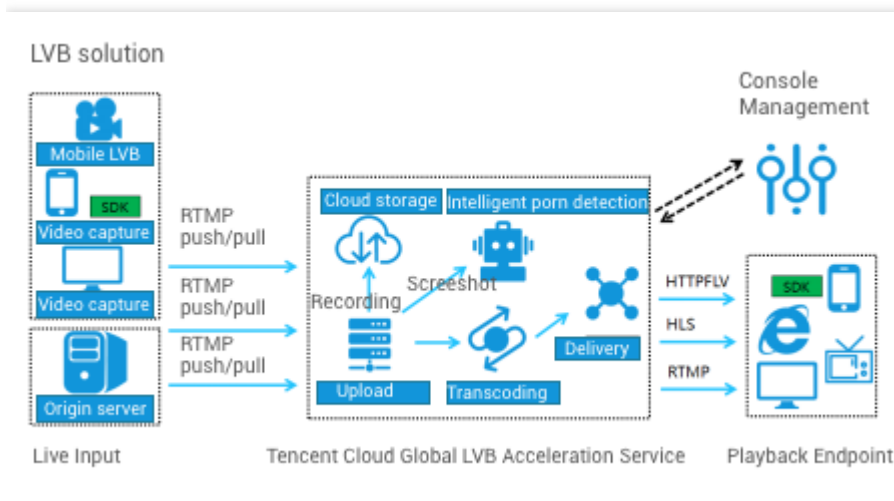
Global CSS Service Overview

Last updated : 2022-05-17 11:05:26

With the growing maturity of audio-visual technologies, the live broadcasting industry is seeing explosive growth around the globe. Chinese internet companies are leveraging on the prior globalization experience of service sector companies and entering the global market en masse. Leading platforms are internationalizing their products to increase competitiveness, while smaller platforms are seeking out new avenues after finding it hard to survive in the aggressive domestic arena. Meanwhile, the battle overseas is no less intense. The 3 giants, YouTube, Periscope and Facebook, may have conquered more than their fair share of the market, but they did leave a significant portion untapped for small and medium platforms. Tencent Cloud continues to strengthen global live broadcasting resource reserves and optimize live broadcast acceleration performance with the goal of helping live broadcasting platforms win international markets.



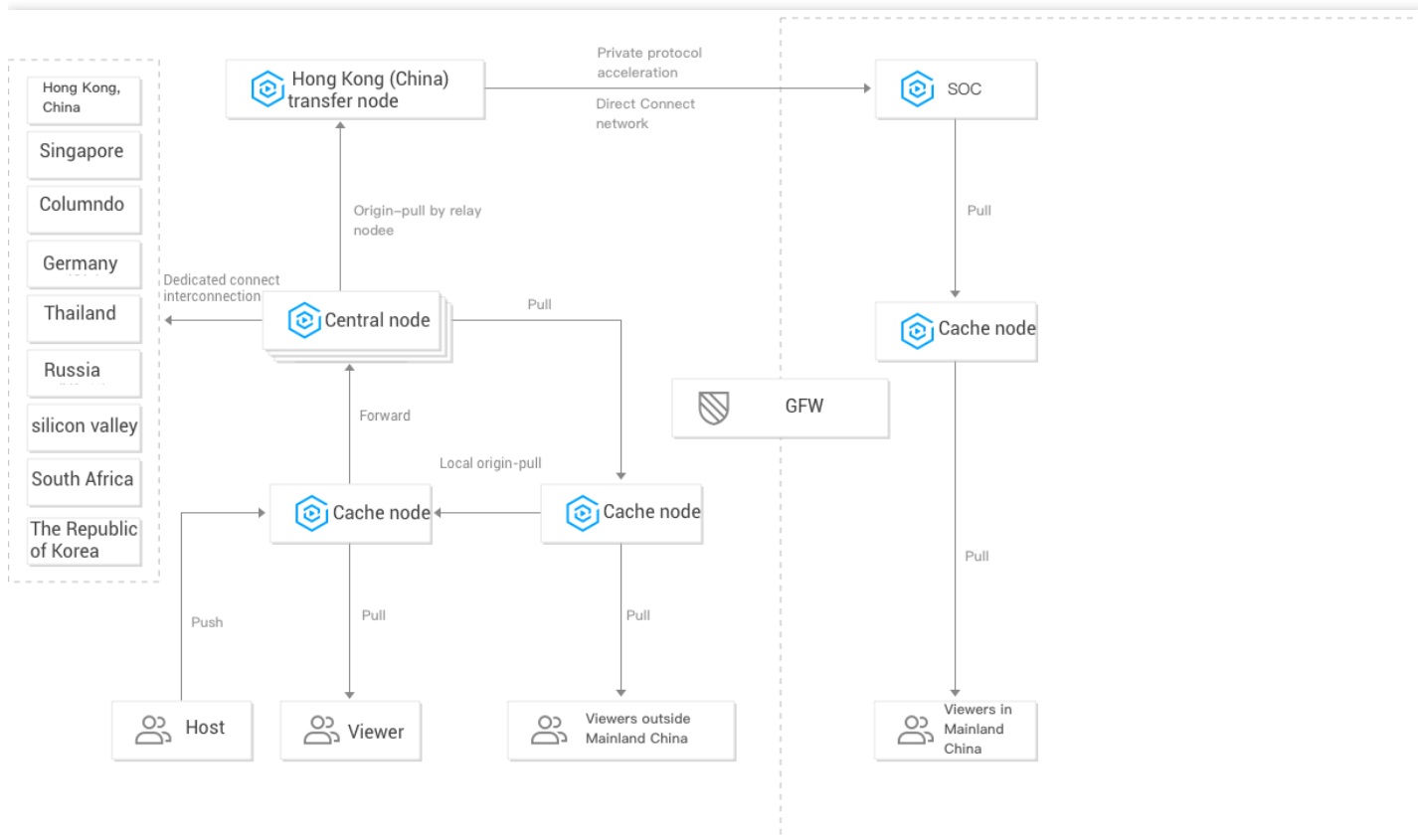
In addition to push and playback, a complete live broadcasting service should also include authentication, transcoding, screencapturing, recording, callback, porn detection, DRM, and other features. The figure below shows the basic feature modules of Tencent Cloud's Global CSS solution.



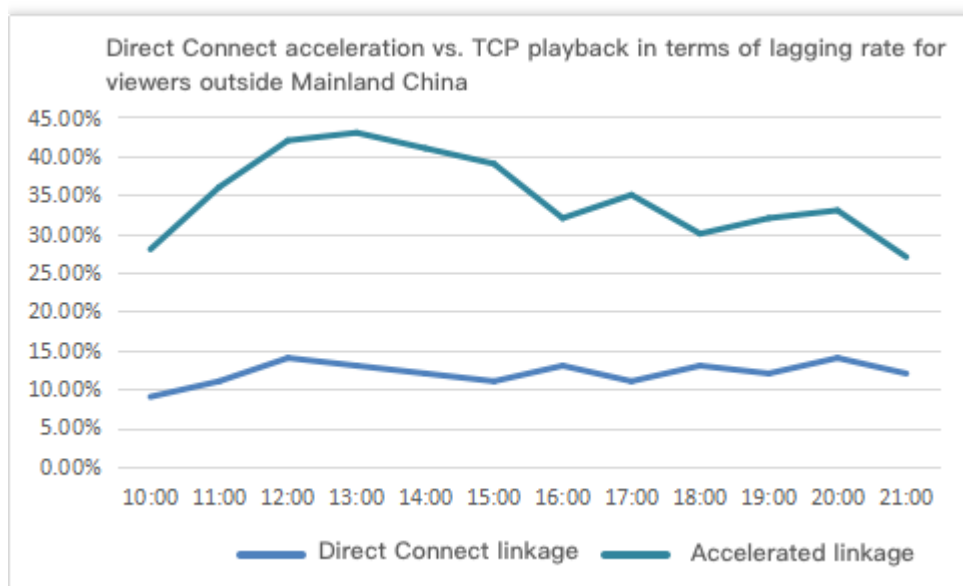
The basic features of global live broadcasting are generally the same as those of domestic live broadcasting. However, there are more challenges overseas mainly due to the wider geographical area, more complicated network environments, and lower cross-border network quality. In order to reduce the latency and lag and improve service stability and reliability, Tencent Cloud has optimized CSS's **architecture, network, security, and resources** for global live broadcasting scenarios.

Deployment of Multiple Central Nodes

Tencent Cloud has built a lot of central IDCs in Hong Kong (China), Thailand, Singapore, Germany, Toronto, Silicon Valley, Russia, South Africa, and South Korea and continues expanding into more countries and regions. These central IDCs hosts all the modules needed for global live broadcasting and are deployed in a distributed manner with a decentralized architecture to better serve global end users and guarantee fast failover upon any IDC exceptions. Taking into account the impact of low cross-border network quality and stability on live broadcast latency and lag, central nodes are interconnected through Direct Connect. Mainland China central nodes are interconnected with those outside of Mainland China through Direct Connect lines with the Hong Kong node as a relay. The overall architecture is as shown below:

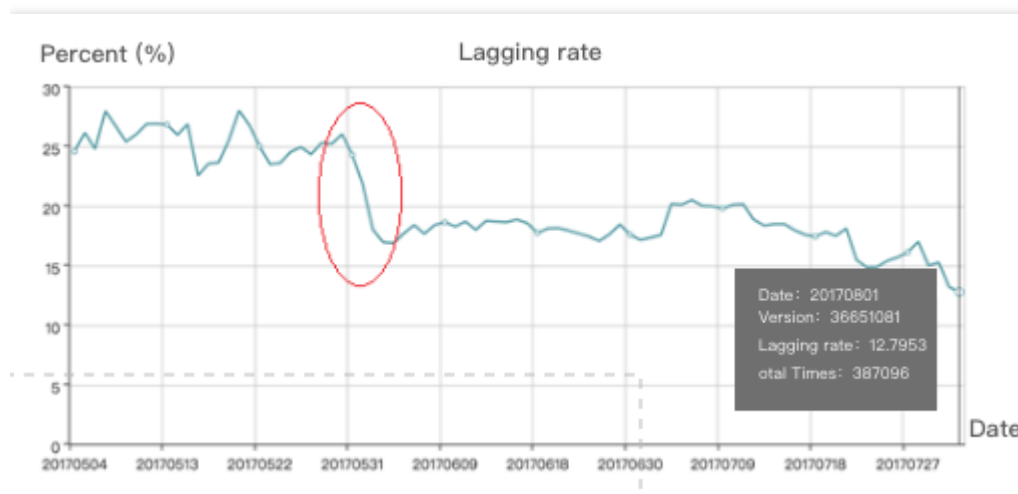


To intuitively demonstrate the acceleration performance of overseas central nodes, we compare the statistics of Chinese end users watching US live streams. As you can see in the figure below, acceleration via Direct Connect keeps the lag rate low and the network stable.



Acceleration in Edge Regions

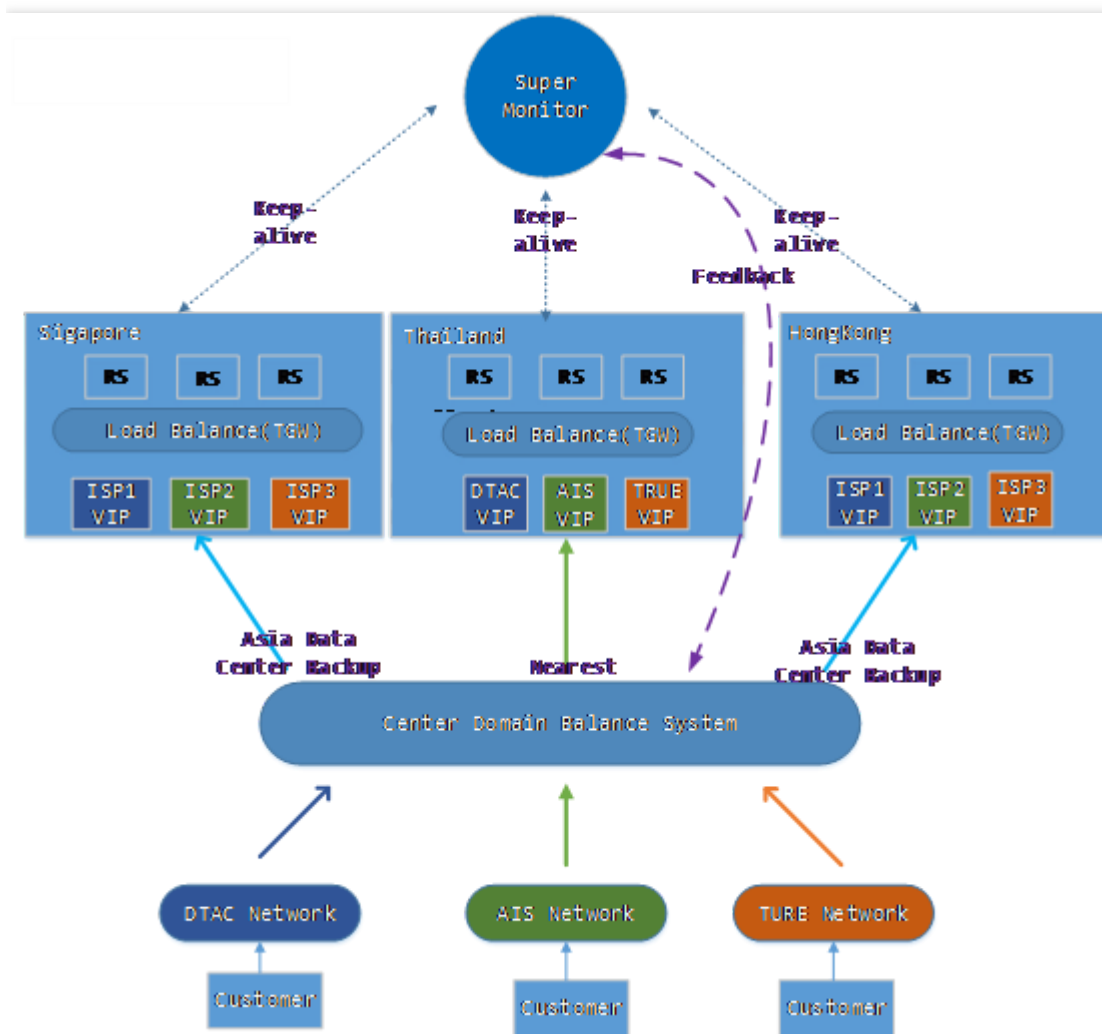
Central nodes can perfectly meet the needs of local end users, but the needs of those in countries and regions without central nodes should also be catered for. Due to various restrictions, no central nodes have been built in these regions, and thus cache nodes are required. Such countries and regions are called edge regions, where the cross-border network quality is relatively low, and the lagging rate of cross-region pull is quite high. Examples of such regions include Malaysia, Indonesia, the Middle East, India, Africa, and South America. For these edge regions, Tencent Cloud sets priorities for service modules. The first priority is ensuring that valid local users can watch live streams without the need of cross-border transmission of local data. Services of other modules are re-pulled from edge servers to central nodes and finally implemented on central nodes. **As you can see in the figure below, after local node acceleration is enabled in an edge region, the lagging rate is significantly reduced, and the acceleration performance is much higher than that of other service providers in the industry.**



Optimal Access and Failover

Similar to Mainland China, other regions also often have multiple ISPs, such as DTAC, AIS, and TURE in Thailand, Chunghwa Telecom, Taiwan Mobile, and so-net in Taiwan (China), as well as Telkomsel, XL, and INDOSAT in Indonesia, and cross-ISP access speed is subject to bandwidth and resources. To improve the access experience of end users on networks operated by different ISPs, Tencent Cloud's scheduling system allows them to access their own ISPs. Plus, its cache nodes built locally generally boast BGP lines for access and peering links with local ISPs. For example, for end users of three ISPs (DTAC, AIS, and TURE) in Thailand, the central scheduling system collects a large amount of foreign IPs and ISP information and automatically schedules them to the nearest CDN nodes based on their IPs, with a recognition accuracy of over **99.5%**. Switching between data centers upon exception is also supported. If the monitoring and detecting nodes find an exception in a region, the system will automatically switch to

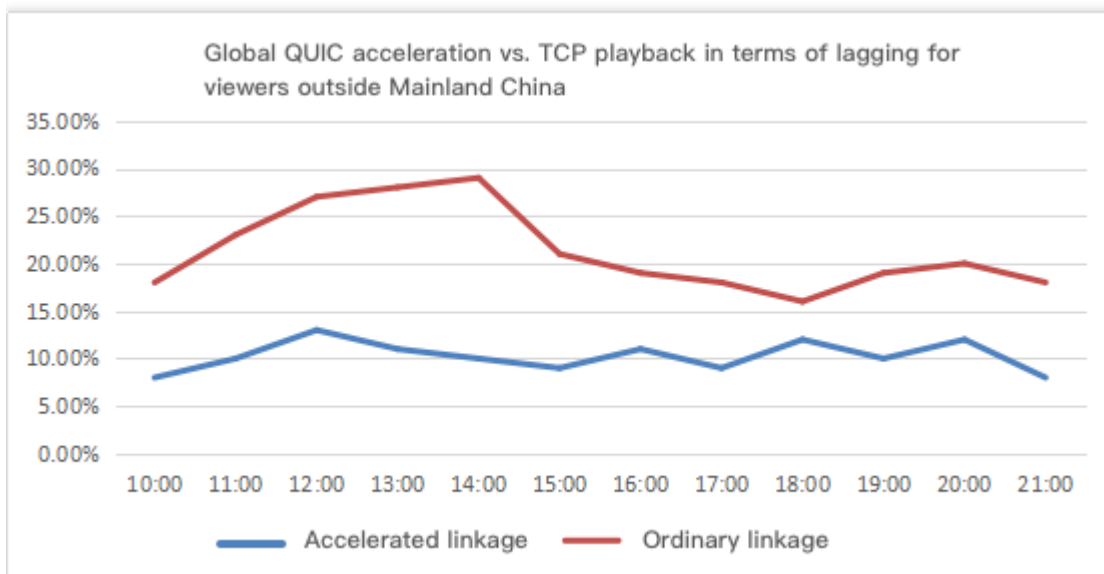
the most optimal data center.



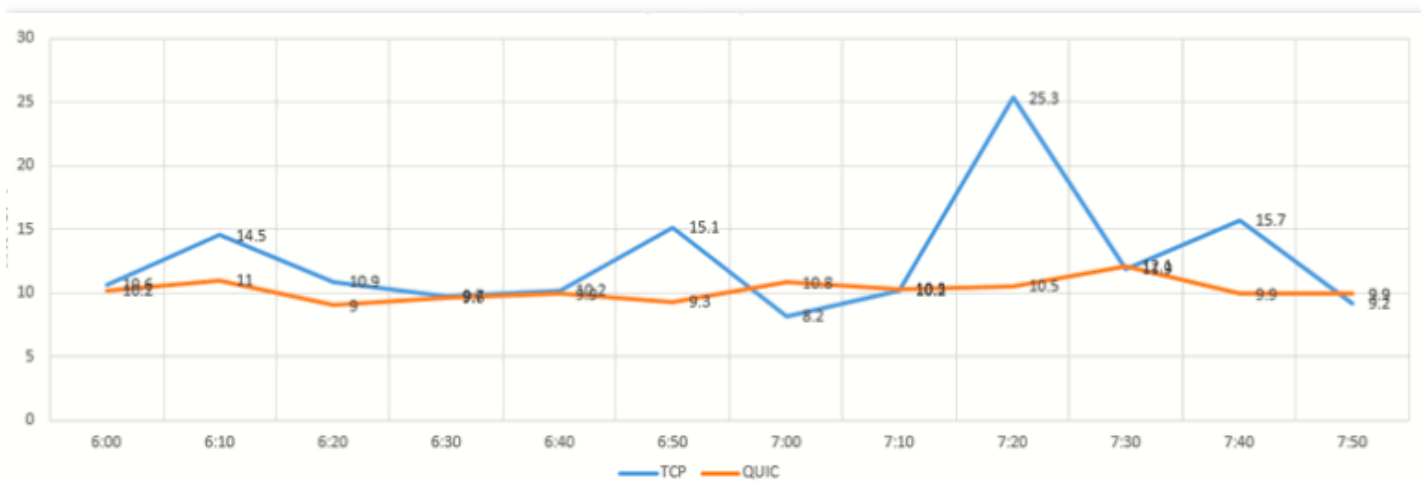
Network Transmission Optimization

When live streams are transmitted on overseas networks, the traditional TCP transmission method cannot guarantee low transmission latency. Due to the long distance of overseas transmission, limitations of international egress bandwidth, and frequent network quality fluctuations, TCP is not suitable for overseas data transmission as it is more time-consuming to be upgraded and optimized and has a higher packet loss rate. Tencent Cloud uses Quick UDP Internet Connections (QUIC) to improve the reliability of data transmission on overseas networks. Upper-layer data proxy acceleration with QUIC is implemented at the application layer, which means that appropriate adjustments to parameters or congestion algorithms can take effect immediately to efficiently fix high latency and high packet loss rate, avoid congestions, and reduce the round-trip time (RTT). **Tests with actual data show that Tencent Cloud's optimized scheme reduces the connection time by 40% and the lagging rate by 20% on average compared**

to the traditional TCP scheme.



The figure below shows a comparison of lags for global viewers watching a live stream pushed by a host in UAE to the UAE cache node. You can see that the lags remain low when the stream is accelerated via QUIC.



Massive Resource Reserve

In addition to technical architectures and solutions, resource reserves are also critical to live broadcasting services. Without the support of global resources, all technologies are merely theories. Tencent Cloud has put in place a globalization strategy and made long-term investments in the overseas market. As we can see in the Tencent Cloud global node distribution chart at the start of the document, **Tencent Cloud has built more than 2000+ transmission nodes in over 50 countries and regions, with a total bandwidth of 100+ Tbps across 50+ global ISP partners and 1000+ overseas cache nodes.** In addition, Tencent Cloud works with multiple ISPs in the same region, ensuring there are at least 3 copies of disaster recovery backup data available in each egress to achieve service stability and reliability. For more information, see [CDN](#)

How to Activate

The Global CSS service can be directly activated in the [CSS console](#).

- If you don't have a Tencent Cloud account yet, you need to sign up first as instructed in [Signing up for a Tencent Cloud Account](#) and then [apply for activation of the CSS service](#).
- If you already have a Tencent Cloud account and have activated CSS, you can proceed directly to the next step.

Go to the CSS console, select **Domain Management** on the left sidebar, and click **Add Domain**.

Domain Management

Push domain: CSS provides you with a default push domain, you can also add your ICP filed domain for live push.
Playback domain: You need to add your ICP filed domain for live playback. For more information on domain management, please see [Domain Management](#).

Add Domain Edit Tag Certificate Management

<input type="checkbox"/>	Domain Name	CNAME ⓘ	Type ▾	Scenario	Region ▾	Status ▾
<input type="checkbox"/>	[blurred]	[red icon]	Playback Domain	CSS	Outside Chinese mainland	Enabled
<input type="checkbox"/>	[blurred]	[green icon]	Playback Domain	CSS	Global	Enabled

In the pop-up window, select the type as **Playback Domain**, select the corresponding **Acceleration Region**, and enter the **Domain Name** that needs to be accelerated.

HTTPDNS Routing

Last updated : 2022-10-14 16:26:05

Overview

CSS routes global push and playback traffic based on DNS resolution by default. This is a common and simple method. However, DNS resolution errors and cross-network traffic occurrences are common due to the complexity of global network environments. We recommend you use Tencent Cloud's HTTPDNS to optimize traffic routing for live streaming.

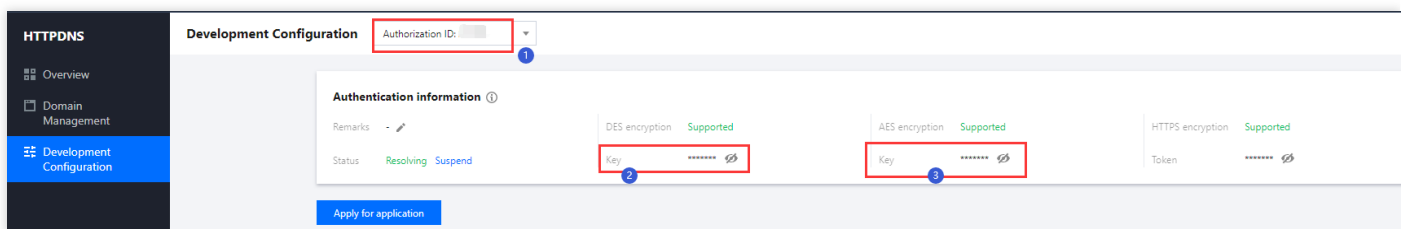
An ISP's local DNS egress performs NAT based on an authoritative DNS destination IP address or forwards the resolving request to other DNS servers. This makes it difficult for the authoritative DNS server to correctly identify the IP address of the ISP's local DNS, resulting in resolution errors and cross-network traffic. Tencent Cloud's HTTPDNS service is powered by leading DNS cluster technologies and supports multi-ISP routing and custom routes.

Note :

This document shows you how to use HTTPDNS to optimize traffic routing for live streaming across the world. For details about the HTTPDNS API used, see [Querying with HTTP Request Methods](#).

Preparations

1. Activate HTTPDNS. For detailed directions, see [Activating HTTPDNS](#).
2. Go to the [Development Configuration](#) page to view the authorization ID and DES key.



Routing Push Traffic Using HTTPDNS

Requesting a push IP address

Use an HTTP GET request in the format of `http://119.29.29.98/d?dn={$push_domain DES-encrypted string}&ip={$ip DES-encrypted string}&id=$id` to request a push IP address from HTTPDNS.

- `push_domain` indicates the push domain, which must be encrypted using the DES algorithm. You can view the key on the [HTTPDNS Development Configuration](#) page. For details, see [AES/DES Encryption/Decryption](#).
- `ip` indicates the public egress IP address of the requester. This field determines the region and ISP of the IP address to which traffic is routed. It also needs to be encrypted using the DES algorithm.
- `id` indicates the authorization ID, which uniquely identifies a user.

Decrypting the IP address

The data obtained from HTTPDNS is DES-encrypted. Decrypt it to get the IP address (`server_ip`). For details, see [AES/DES Encryption/Decryption](#).

Splicing the push URL

The format of a push URL is `rtmp://server_ip/live/streamname?`

`txTime=xxx&txSecret=xxx&txHost=domain . server_ip` is the **push IP address obtained in the previous step**. `txHost` (important) is the domain you use for push.

Routing Playback Traffic Using HTTPDNS

Requesting the playback IP address

Use an HTTP GET request in the format of `http://119.29.29.98/d?dn={$domain DES-encrypted string}&ip={$ip DES-encrypted string}&id=$id` to request the playback IP address from HTTPDNS.

Field	Description
push_domain	The playback domain. The value of this field must be encrypted using the DES algorithm. You can view the key on the HTTPDNS Development Configuration page. For details, see AES/DES Encryption/Decryption .
ip	The public egress IP address of the requester. This field determines the region and ISP of the IP address to which traffic is routed. It also needs to be encrypted using the DES algorithm.
id	The authorization ID, which uniquely identifies a user.

Decrypting the IP address

The data obtained from HTTPDNS is DES-encrypted. Decrypt it to get the IP address (`server_ip`). For details, see [AES/DES Encryption/Decryption](#).

Splicing the playback URL

- **HTTP:** The formats of HTTP playback URLs for FLV and HLS are as follows (`server_ip` is the **playback IP address obtained in the previous step** and `play_domain` is the playback domain):

```
http://server_ip/play_domain/live/streamname.flv?xxxxxxxxxx
http://server_ip/play_domain/live/ streamname.m3u8?xxxxxxxxxx
http://server_ip/play_domain/live/ streamname -123.ts?xxxxxxxxxx
```

- **HTTPS:** The splicing rules of HTTPS playback URLs for FLV and HLS depend on the player. **The destination IP address of the TCP connection must be the `server_ip` assigned by HTTPDNS**, and the URLs should be regular playback requests. The formats are as follows:

```
https://server_ip/play_domain/live/ streamname.flv?xxxxxxxxxx
https://server_ip/play_domain/live/ streamname.m3u8?xxxxxxxxxx
https://server_ip/play_domain/live/ streamname -123.ts?xxxxxxxxxx
```

- **RTMP:** The format of an RTMP playback URL is as follows (`server_ip` is the **playback IP address obtained in the previous step** and `play_domain` is the playback domain):

```
rtmp://server_ip/play_domain/live/ streamname?xxxxxxxxxx
```

Note :

There is a small likelihood of HTTPDNS request errors. If your request times out or the result returned is not an IP address or is empty, please perform resolution at the local DNS server.

Callback Notifications

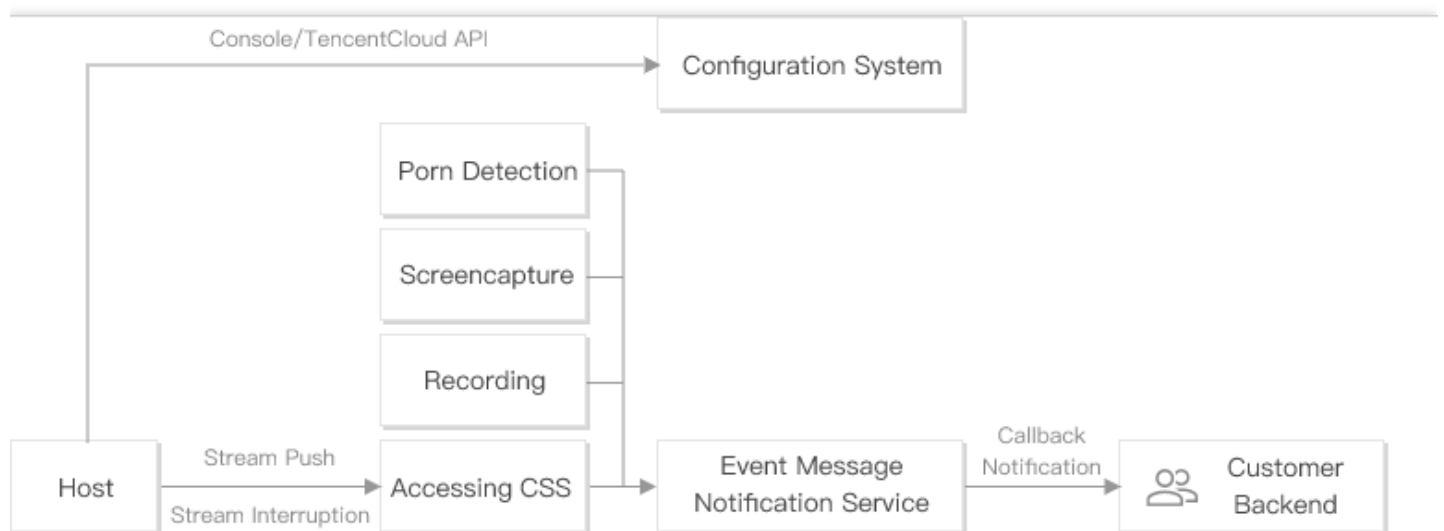
How to Receive Event Notification

Last updated : 2021-12-24 11:41:27

If an event configured in the template triggers a callback during live streaming, Tencent Cloud will send a request to the customer's server which is responsible for the response. After passing the verification, the server will obtain a JSON packet of the callback.

Currently, the following events can trigger a notification during live streaming: stream push, stream interruption, recording, screencapture and porn detection.

Overall Process



Process Description:

1. The host configures event message notification URLs and features such as recording and screencapture in the console or by calling TencentCloud APIs.
2. The host pushes and stops the stream.
3. When an event occurs, a message will be sent to the customer backend via the event message notification service.

Event Message Notification Protocol

Network protocol

- Request: HTTP POST request with a JSON packet. The specific packet content of each type of message is described later.
- Response: HTTP status code = 200. The server ignores the specific content of the response packet. For protocol-friendliness, we recommend you add `JSON: {"code":0}` to the response.

Notification reliability

The event notification service has a retry mechanism. For the screencapture event, up to 5 retries will be made at an interval of 2 minutes. For the stream push, stream interruption, recording, and porn detection events, up to 12 retries will be made at an interval of 1 minute.

To prevent frequent retries from placing too much strain on your server and bandwidth, make sure response packets are returned as expected. A retry is triggered in the following cases:

- No response packet is returned for a long time (at least 20 seconds).
- The HTTP status code in the response is not `200`.

How to Configure Event Callbacks

You can configure callbacks via the [CSS console](#) or [server APIs](#).

Note :

CSS allows you to configure callback URLs separately for events of stream push, stream interruption, recording, screencapture and porn detection.

CSS console

1. Log in to the CSS console and click **Event Center** > **Live Stream Callback** to create a callback template. For detailed directions, see [Creating a Callback Template](#).
2. Click **Domain Management**, find the target push domain name, and click **Manage** > **Template Configuration** to bind it with the callback template. For detailed directions, see [Callback Configuration](#).

Server APIs

1. Call the [CreateLiveCallbackTemplate](#) API to create a callback template and set the callback parameters.
2. Call the [CreateLiveCallbackRule](#) API to set the `DomainName` (push domain name) and `TemplateId` (returned in step 1) parameters. Enter the `AppName` in the push and playback URLs to enable callback for specific live streams.

Callback Parameters

After the template is successfully bound with the domain name, if an event configured in the template is triggered during the live streaming, Tencent Cloud will send a JSON packet containing the callback information to the customer's server. The callback parameters are detailed as below:

- [Stream push event notification](#)
- [Stream interruption event notification](#)
- [Recording event notification](#)
- [Screencapture event notification](#)
- [Porn detection event notification](#)

Stream Pushing Notification

Last updated : 2022-05-12 12:35:53

The stream pushing callback informs you whether stream pushing is successful or interrupted. You need to configure a server address for the callback in a callback template and bind the template with your push domain name. After push starts via a URL generated under the domain, the Tencent Cloud backend will send the callback to the server you set.

This document describes the parameters in a stream pushing callback notification sent to you by CSS.

Note

This guide assumes that you understand how to configure callbacks and receive callback notifications from CSS. For details, see [How to Receive Event Notification](#).

Stream Pushing Event Parameters

Event type

Event Type	Value
Successful push	event_type = 1
Push interrupted	event_type = 0

Common callback parameters

Parameter	Type	Description
t	int64	Expiration time, which is the Unix timestamp when the event notification signature expires. <ul style="list-style-type: none">The default validity period of a callback notification from Tencent Cloud is 10 minutes. If the time specified by the `t` value in a notification has elapsed, then this notification is considered invalid. This prevents network replay attacks.The value of `t` is a decimal Unix timestamp, that is, the number of seconds that have elapsed since 00:00:00 (UTC/GMT time), January 1, 1970.
sign	string	Security signature. sign = MD5(key + t). Tencent Cloud splices the encryption key and `t`, generates the MD5 hash of the spliced string, and embeds it in callback messages. Your backend server can perform the same

calculation when it receives a callback message. If the signature matches, it indicates the message is from Tencent Cloud.

Note :

You can set the callback key in **Event Center** > [Live Stream Callback](#), which is used for authentication. We recommend you set this field to ensure data security.

Callback Key

Please enter a callback key (composed of uppercase and lowercase letters)

Push Callback

Please enter a push callback URL (header: http, https, etc.)

Interruption Callback

Please enter an interruption callback URL (header: http, https, etc.)

Recording Callback

Please enter a recording callback URL (header: http, https, etc.)

Screencapture Callback

Please enter a screencapture callback URL (header: http, https, etc.)

Porn Detection Callback

Please enter a porn detection callback URL (header: http, https, etc.)

Callback parameters

Parameter	Type	Description
appid	int	User APPID
app	string	Push domain name
appname	string	Push path
stream_id	string	Live stream name
channel_id	string	Same as the live stream name
event_time	int64	UNIX timestamp when the event message is generated

Parameter	Type	Description
sequence	string	Message sequence number, which identifies a push. The notifications for a push, whether they are for successful push or stream interruption, have the same sequence number.
node	string	IP of the live stream access point
user_ip	string	User push IP
stream_param	string	User push URL parameters
push_duration	string	Push duration of the interrupted stream in milliseconds
errcode	int	Stream pushing error code
errmsg	string	Stream pushing error message
set_id	int	Whether the push is from inside the Chinese mainland. 1-6: yes; 7-200: no.
width	int	Video width. The value of this parameter may be 0 if the video header information is missing at the beginning of a push.
height	int	Video height. The value of this parameter may be 0 if the video header information is missing at the beginning of a push.

Causes of stream interruption

For a list of the causes of stream interruption, see [Stream Interruption Records](#).

Sample callback

```
{
  "app": "test.domain.com",
  "appid": 12345678,
  "appname": "live",
  "channel_id": "test_stream",
  "errcode": 0,
  "errmsg": "ok",
  "event_time": 1545115790,
```



```
"event_type":1,  
  
"set_id":2,  
  
"node":"100.121.160.92",  
  
"sequence":"6674468118806626493",  
  
"stream_id":"test_stream",  
  
"stream_param":"stream_param=test",  
  
"user_ip":"119.29.94.245",  
  
"width": 0,  
  
"height": 0,  
  
"sign":"ca3e25e5dc17a6f9909a9ae7281e300d",  
  
"t":1545030873  
}
```

Recording Event Notification

Last updated : 2022-11-23 11:10:05

The live recording feature records live streams in real time according to the recording template bound to the push domain name, and then stores the recording files in VOD. A recording callback notifies you of the information of a recording file, including the start and end time of recording, the recording file ID, the file size, and the download URL. To receive recording callbacks, you need to configure a callback template, specify a server address for the callback, and bind the template to your push domain name. When a recording event occurs, the CSS backend will send the recording file information to the specified server.

This document describes the fields in a callback notification sent by CSS after a recording event occurs.

Notes

- This document assumes you already know how to configure and [receive](#) callbacks.
- Recording files are stored in the [VOD console](#) by default. Therefore, you need to activate VOD first and make sure it does not have overdue payments.
- If a recording task is created by the [CreateRecordTask](#) API, the recording callback returned will not include the [stream_param](#) parameters of the push URL. They will be included if a task is created using another method.
- If HLS recording resumption is enabled, a callback will be triggered only for the final recording file. No callbacks will be sent when push is interrupted.

Recording Event Parameters

Event type

Event Type	Value
Live recording	event_type = 100

Common callback parameters

Parameter	Type	Description
t	int64	The time (Unix timestamp) when the notification signature expires. <ul style="list-style-type: none">• The default validity period of a callback notification from Tencent Cloud is 10 minutes. After the time specified by the `t` value elapses, a notification will be considered invalid. This can prevent network replay attacks.

		<ul style="list-style-type: none">The value of `t` is a decimal Unix timestamp, which is the number of seconds that have elapsed since 00:00:00 (UTC/GMT time), January 1, 1970.
sign	string	The Security signature. `sign` = MD5(`key` + `t`). Tencent Cloud splices the encryption key and `t`, generates an MD5 hash of the spliced string, and embeds it in callback notifications. Your backend server performs the same calculation when it receives a callback, and if the signature matches, it indicates that the notification is from Tencent Cloud.

Note :

You can set the callback key in **Event Center** > [Live Stream Callback](#), which is used for authentication. We recommend you set this field to ensure data security.

Callback Key	Please enter a callback key (composed of uppercase and lowercase letters, numbers, and underscores, 1-32 characters)
Push Callback	Please enter a push callback URL (header: http, https, etc.)
Interruption Callback	Please enter an interruption callback URL (header: http, https, etc.)
Recording Callback	Please enter a recording callback URL (header: http, https, etc.)
Screencapture Callback	Please enter a screencapture callback URL (header: http, https, etc.)
Porn Detection Callback	Please enter a porn detection callback URL (header: http, https, etc.)

Recording callback parameters

Parameter	Type	Description
appid	int	The user APPID .
app	string	The push domain.
appname	string	The push path.

Parameter	Type	Description
stream_id	string	The stream ID.
channel_id	string	Same as the stream ID.
file_id	string	The VOD file ID, which uniquely identifies a file in VOD .
record_file_id	string	The recording file ID.
file_format	string	The file format. Valid values: <code>flv</code> , <code>hls</code> , <code>mp4</code> , <code>aac</code> .
task_id	string	The ID of a recording task, which is returned by the CreateRecordTask API and is valid only if the task is created by the API.
start_time	int64	The recording start time. This is different from the start time of the recording file. The start time of the recording file = <code>end_time</code> - <code>duration</code> .
end_time	int64	The recording end time.
start_time_usec	int	The recording start time (microseconds).
end_time_usec	int	The recording end time (microseconds).
duration	int64	The duration of the recording file, in seconds.
file_size	uint64	The recording file size, in bytes.
stream_param	string	The push URL parameters (custom).
video_url	string	The download URL of the recording file.
media_start_time	int	The PTS when the stream is first pulled for recording. This is not necessarily the PTS of the first frame of the recording file.
record_bps	int	The bitrate, in kbps, of the transcoding output recorded.
callback_ext	The JSON object string.	The JSON object includes multiple fields: <code>video_codec</code> indicates the video codec. <code>resolution</code> indicates the resolution of the pushed stream. These are all additional fields of a recording callback. We recommend you do not rely your business logic too much on them.

Sample callback

```
{
  "event_type": 100,
```

```
"appid": 12345678,

"app": "yourapp",

"callback_ext": "{\"video_codec\":\"h264\",\"resolution\":\"640x480\"}",

"appname": "yourappname",

"stream_id": "stream_test",

"channel_id": "stream_test",

"file_id": "1234567890",

"record_file_id": "1234567890",

"file_format": "hls",

"task_id": "UpTbk5RSVhRQ*****0xTS1NTQlt1RVRLU1JAWW9EUb",

"start_time": 1642089445,

"end_time": 1642089598,

"start_time_usec": 316441,

"end_time_usec": 618577,

"duration": 154,

"file_size": 277941079,

"stream_param": "stream_param=test",

"video_url": "http://12345678.vod2.myqcloud.com/xxxx/yyyy/zzzz.m3u8",

"media_start_time": 135802,

"record_bps": 0,

"sign": "ca3e25e*****09a9ae7281e300d",

"t": 1545030873
}
```

Screencapturing Event Notification

Last updated : 2022-05-05 15:06:35

Live screencapture takes real-time screenshots from a live stream at the specified interval and stores them in COS. A screencapture callback returns information about stored screenshots, including the screenshot generation time, image size, file path, and download link. To receive screencapture callbacks, you need to configure your server address in a callback template and bind the template with your push domain. When a live screencapture event occurs, the CSS backend will send the screenshot information to the server configured.

This document describes the fields in a live screencapture callback message.

Notes

- This document assumes you already know how to configure and [receive](#) callbacks.
- The information returned by a screencapture callback can be used for porn detection, live video thumbnail generation, and other scenarios.

Screencapture Event Parameters

Event type

Event Type	Value
Live screencapture	event_type = 200

Common callback parameters

Parameter	Type	Description
t	int64	<p>Expiration time, which is the Unix timestamp when the event notification signature expires.</p> <ul style="list-style-type: none">• The default validity period of a message notification from Tencent Cloud is 10 minutes. If the time specified by the `t` value in a message notification has elapsed, then this notification is considered invalid, thereby preventing network replay attacks.• The value of `t` is a decimal Unix timestamp, that is, the number of seconds that have elapsed since 00:00:00 (UTC/GMT time), January 1, 1970.
sign	string	<p>Security signature. sign = MD5(key + t).</p> <p>Tencent Cloud splices the encryption key and `t`, generates the MD5 hash of the spliced string, and embeds it in callback messages. Your backend server can perform the same</p>

calculation when it receives a callback message. If the signature matches, it indicates the message is from Tencent Cloud.

Note :

A key is used for authentication. You can set it in **Event Center** > **Live Stream Callback**. We recommend you set it to ensure data security.

Callback Key

Please enter a callback key (composed of uppercase and lc

Push Callback

Please enter a push callback URL (header: http, https, etc.)

Interruption Callback

Please enter an interruption callback URL (header: http, htt

Recording Callback

Please enter a recording callback URL (header: http, https,

Screenshot Callback

Please enter a screenshot callback URL (header: http, ht

Porn Detection Callback

Please enter a porn detection callback URL (header: http, h

Callback message parameters

Parameter	Type	Description
app	string	Push domain name
appname	string	Push path
stream_param	string	Push URL parameters
stream_id	string	Live stream name
channel_id	string	Same as the stream name

Parameter	Type	Description
create_time	int64	Unix timestamp when a screenshot is generated
file_size	int	Screenshot file size in bytes
width	int	Screenshot width in pixels
height	int	Screenshot height in pixels
pic_url	string	Screenshot file path (/path/name.jpg)
pic_full_url	string	Screenshot download URL

Sample callback message

```
{
  "app": "test.app",
  "appname": "live",
  "channel_id": "your_channelid",
  "create_time": 1622599925,
  "event_type": 200,
  "file_size": 30670,
  "height": 720,
  "pic_full_url": "http://your.cos.region.myqcloud.com/channelid/channelid-screenshot-10-12-05-1280x720.jpg",
  "pic_url": "/channelid/channelid-screenshot-10-12-05-1280x720.jpg",
  "sign": "ca3e25e5dc17a6f9909a9ae7281e300d",
  "stream_id": "your_streamid",
  "stream_param": "txSecret=ca3e25e5dc17a6f9909a9ae7281e300d&txTime=60B83800",
  "t": 1622600525,
  "width": 1280
}
```


Porn Detection Event Notification

Last updated : 2022-03-21 10:40:18

Live porn detection takes screenshots of live streams according to the screencapturing template, saves them to COS, and leverages Tencent Cloud's Image Moderation System to identify problematic images. A porn detection callback provides you with porn detection details, including the category, priority, and screencapturing time of problematic images. To receive porn detection callbacks, you need to configure a server address in the callback template and bind the template to your push domain name. If porn detection is enabled for a stream, the CSS backend will send information about suspicious images to the server address you configured.

This document describes the parameters in callback message notifications sent by Tencent Cloud CSS after a porn detection callback event is triggered.

Notes

- You need to understand how to configure callbacks and how you will receive messages via Tencent Cloud CSS before reading this document. For more information, see [How to Receive Event Notification](#).
- By default, only questionable results of porn detection will be called back.
- We recommend you use the `type` of an image to determine whether it is pornographic. As the detection results are not 100% accurate and there may be false positives or false negatives, you can confirm them manually if necessary.

Screencapturing Event Parameters

Event type parameters

Event Type	Parameter Value
Live porn detection	<code>event_type = 317</code>

Common callback parameters

Parameter	Type	Description
<code>t</code>	<code>int64</code>	Expiration time, which is the Unix timestamp when the event notification signature expires. <ul style="list-style-type: none">The default validity period of a message notification from Tencent Cloud is 10 minutes. If the time specified by the <code>t` value in a message notification has elapsed,</code>

		<p>then this notification is considered invalid, thereby preventing network replay attacks.</p> <ul style="list-style-type: none"> The format of `t` is a decimal Unix timestamp, i.e., the number of seconds that have elapsed since 00:00:00 (UTC/GMT time), January 1, 1970.
sign	string	<p>Event notification security signature sign = MD5(key + t).</p> <p>Note: Tencent Cloud concatenates the encryption key and `t`, calculates the `sign` value through MD5, and places it in the notification message. When your backend server receives the notification message, it can confirm whether the `sign` is correct based on the same algorithm and then determine whether the message is indeed from the Tencent Cloud backend.</p>

Note :

You can set the callback key in **Event Center** > [Live Stream Callback](#), which is used for authentication. We recommend you set this field to ensure data security.

Callback Key

Please enter a callback key (composed of uppercase and lc

Push Callback

Please enter a push callback URL (header: http, https, etc.)

Interruption Callback

Please enter an interruption callback URL (header: http, htt

Recording Callback

Please enter a recording callback URL (header: http, https,

Screenshot Callback

Please enter a screenshot callback URL (header: http, ht

Porn Detection Callback

Please enter a porn detection callback URL (header: http, h

Callback message parameters

Parameter	Required	Data Type	Description
streamId	No	String	Stream name

Parameter	Required	Data Type	Description
channelId	No	String	Channel ID
img	Yes	String	Link to the alerted image
type	Yes	Array	Categories of negative labels with the highest priority in the detection result. For details, see the description of <code>label</code> .
score	Yes	Array	Scores of <code>type</code>
ocrMsg	No	String	OCR result (if any)
suggestion	Yes	String	Suggestion. Valid values: <ul style="list-style-type: none"> Block Review Pass
label	Yes	String	Negative label with the highest priority in the detection result (<code>LabelResults</code>). This is the moderation result suggested by the model. We recommend you handle different types of violations and suggestions based on your business needs.
subLabel	Yes	String	Sub-label under the negative label with the highest priority in the detection result, such as porn - sexual acts. If no content is sub-labeled, this parameter will be empty.
labelResults	No	Array of <code>LabelResult</code>	Negative label hit details of the category model, including the detected porn content, ads, terrorism content, and politically sensitive content Note: This field may return <code>null</code> , indicating that no valid values can be obtained.
objectResults	No	Array of <code>ObjectResult</code>	Detection result of the object model, including label name, hit score, coordinates, scenario, and suggested operation regarding objects, advertising logos, QR codes, etc. For details, see the description of the data structure of <code>ObjectResults</code> . Note: This field may return <code>null</code> , indicating that no valid values can be obtained.

Parameter	Required	Data Type	Description
ocrResults	No	Array of OcrResult	OCR result, including text coordinates, recognized text, suggested operation, etc. For details, see the description of the data structure of <code>OcrResults</code> . Note: This field may return <code>null</code>, indicating that no valid values can be obtained.
libResults	No	Array of LibResult	Blocklist/Allowlist moderation result
screenshotTime	Yes	Number	Screenshot time
sendTime	Yes	Number	Time when the request was sent, in Unix timestamp format
stream_param	No	String	Push parameters
app	No	String	Push domain name
appid	No	Number	Application ID
appname	No	String	Push path

LabelResult

Hit result of the category model

Parameter	Type	Description
Scene	String	Scenario identified by the model, such as advertising, pornographic, and harmful
Suggestion	String	Operation suggested by the system for the current negative label. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none"> Block Review Pass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Hit score of the label model
Details	Array of LabelDetailItem	Sub-label hit details of the category model

LabelDetailItem

Sub-label hit details of the category model

Parameter	Type	Description
Id	Integer	ID
Name	String	Sub-label name
Score	Integer	Sub-label score. Value range: 0-100

ObjectResult

Object detection result

Parameter	Type	Description
Scene	String	Object scenario identified, such as QR code, logo, and OCR
Suggestion	String	Operation suggested by the system for the current negative label. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Sub-label hit score of the scenario model. Value range: 0-100
Names	Array of String	List of object names
Details	Array of ObjectDetail	Object detection details

ObjectDetail

Object detection details. When the detection scenario is object, advertising logo, or QR code, it returns the label name, label value, label score, and location information of the detection frame.

Parameter	Type	Description
Id	Integer	ID of the object identified

Parameter	Type	Description
Name	String	Object label identified
Value	String	Value or content of the object label identified. For example, if the label is QR code (<code>QrCode</code>), this parameter is the URL of the QR code.
Score	Integer	Hit score of the object label. Value range: 0-100. For example, <code>QrCode 99</code> indicates a high likelihood that the content is a QR code.
Location	Location	Coordinates (of the top-left corner), dimensions, and rotation of the object detection frame

Location

Coordinates and other information of the detection frame

Parameter	Type	Description
X	Float	Horizontal coordinate of the top-left corner
Y	Float	Vertical coordinate of the top-left corner
Width	Float	Width
Height	Float	Height
Rotate	Float	Rotation angle of the detection frame

OcrResult

OCR result

Parameter	Type	Description
Scene	String	Recognition scenario. Default value: OCR
Suggestion	String	Operation suggested by the system for the negative label with the highest priority. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name

Parameter	Type	Description
Score	Integer	Sub-label hit score of the scenario model. Value range: 0-100
Text	String	Text
Details	Array of OcrTextDetail	OCR details

OcrTextDetail

OCR details

Parameter	Type	Description
Text	String	Text recognized (up to 5,000 bytes)
label	String	Negative label in the detection result
Keywords	Array of String	Keywords hit under the label
Score	Integer	Hit score of the label model. Value range: 0-100
Location	Location	OCR text coordinates

LibResult

Blocklist/Allowlist result

Parameter	Type	Description
Scene	String	Scenario recognition result of the model. Default value: Similar
Suggestion	String	Operation suggested by the system. We recommend you handle different types of violations and suggestions based on your business needs. Returned values: <ul style="list-style-type: none">BlockReviewPass
label	String	Negative label in the detection result
SubLabel	String	Sub-label name
Score	Integer	Recognition score of the image search model. Value range: 0-100
Details	Array of LibDetail	Blocklist/Allowlist details

LibDetail

Custom list or blocklist/allowlist details

Parameter	Type	Description
Id	Integer	ID
ImageId	String	Image ID
label	String	Negative label in the detection result
Tag	String	Custom label
Score	Integer	Model recognition score. Value range: 0-100

Sample callback message

```
{
  "ocrMsg": "",
  "type": [1],
  "score": 99,
  "screenshotTime": 1610640000,
  "level": 0,
  "img": "http://1.1.1.1/download/porn/test.jpg",
  "abductionRisk": [],
  "faceDetails": [],
  "sendTime": 1615859827,
  "suggestion": "Block",
  "label": "Porn",
  "subLabel": "PornHigh",
  "labelResults": [{
    "HitFlag": 0,
    "Scene": "Illegal",
    "Suggestion": "Pass",
    "Label": "Normal",
    "SubLabel": "",
    "Score": 0,
    "Details": []
  }, {
    "HitFlag": 1,
    "Scene": "Porn",
    "Suggestion": "Block",
    "Label": "Porn",
    "SubLabel": "PornHigh",
    "Score": 99,
    "Details": [{
```

```
"Id": 0,
"Name": "PornHigh",
"Score": 99
}, {
"Id": 1,
"Name": "WomenChest",
"Score": 99
}]
}, {
"HitFlag": 0,
"Scene": "Sexy",
"Suggestion": "Pass",
"Label": "Normal",
"SubLabel": "",
"Score": 0,
"Details": []
}, {
"HitFlag": 0,
"Scene": "Terror",
"Suggestion": "Pass",
"Label": "Normal",
"SubLabel": "",
"Score": 0,
"Details": []
}],
"objectResults": [{
"HitFlag": 0,
"Scene": "QrCode",
"Suggestion": "Pass",
"Label": "Normal",
"SubLabel": "",
"Score": 0,
"Names": [],
"Details": []
}, {
"HitFlag": 0,
"Scene": "MapRecognition",
"Suggestion": "Pass",
"Label": "Normal",
"SubLabel": "",
"Score": 0,
"Names": [],
"Details": []
}, {
"HitFlag": 0,
"Scene": "PolityFace",
"Suggestion": "Pass",
```

```
"Label": "Normal",
"SubLabel": "",
"Score": 0,
"Names": [],
"Details": []
}],
"ocrResults": [{
  "HitFlag": 0,
  "Scene": "OCR",
  "Suggestion": "Pass",
  "Label": "Normal",
  "SubLabel": "",
  "Score": 0,
  "Text": "",
  "Details": []
}],
"streamId": "teststream",
"channelId": "teststream",
"stream_param": "txSecret=40f38f69f574fd51126c421a3d96c374&txTime=5DEBEC80",
"app": "5000.myqcloud.com",
"appname": "live",
"appid": 10000,
"event_type": 317,
"sign": "ac920c3e66*****78cf1b5de2c63",
"t": 1615860427
}
```

Relay Event Notification

Last updated : 2022-01-13 14:53:01

Relay callbacks are used to call back status information of relay tasks. You need to configure the callback address in a relay task and then Tencent Cloud CSS backend will call back different results to the specified server.

This document describes the parameters in callback message notifications sent by Tencent Cloud CSS after a stream push/interruption callback event is triggered.

Notes

You need to understand how to configure callbacks and how will you receive messages via Tencent Cloud CSS before reading this document. For more information, see [How to Receive Event Notifications](#).

Relay Event Parameters

Event type parameters

Event Type	Parameter Value
Relay	event_type = 314

Common callback parameters

Parameter	Type	Description
appId	int	User APP ID
callback_event	string	Callback event type
source_urls	string	Pull source URLs
to_url	string	Push destination URL
stream_id	string	Live stream name
task_id	string	Task ID
msg	string	Callback details of different types of events

Parameters in `msg`

Parameter	Type	Description
task_start_time	int	Task start timestamp, in milliseconds
url	string	Source URL of the current pull task
index	string	Index of the list for on-demand files
duration	int	Duration of an on-demand file, in seconds
task_exit_time	int	Task stop timestamp, in milliseconds
code	string	Task stop error code
message	string	Task stop error message

Sample callback message**TaskStart** - Callback of the task start event

```
{
  "appid": 4,

  "callback_event": "TaskStart",

  "event_type": 314,

  "interface": "general_callback",

  "msg": "{\"task_start_time\":0}",

  "product_name": "pullpush",

  "source_urls": "[\"http://yourURL.cn/live/normal_230753472*****21162358-upload-45eb/playlist.m3u8\"]\n",

  "stream_id": "testvod",

  "task_id": "118148",

  "to_url": "rtmp://5000.livepush.myqcloud.com/live/testvod"
}
```

VodSourceFileStart - Callback of the on-demand file's start

```
{
  "appid": 4,

  "callback_event": "VodSourceFileStart",

  "callback_url": "http://you.callback.url",

  "event_type": 314,

  "interface": "general_callback",

  "msg": "{\\"url\\":\\"http://remit-tx-ugcpub.douyucdn2.cn/live/normal_466247620****
*3100448-upload-216b/playlist.m3u8\\",\\"index\\":0,\\"duration\\":14920}",

  "product_name": "pullpush",

  "source_urls": "[\\"http://yourURL.cn/live/normal_466247620*****3100448-upload-21
6b/playlist.m3u8\\"]\n",

  "stream_id": "testvod",

  "task_id": "118145",

  "to_url": "rtmp://5000.livepush.myqcloud.com/live/testvod"
}
```

VodSourceFileFinish - Callback of the on-demand file's end

```
{
  "appid": 4,

  "callback_event": "VodSourceFileFinish",

  "callback_url": "http://you.callback.url",

  "event_type": 314,

  "interface": "general_callback",

  "msg": "{\\"url\\":\\"http://yourURL.cn/live/normal_466247620*****3100448-upload-21
6b/playlist.m3u8\\",\\"index\\":0,\\"duration\\":14920}",

  "product_name": "pullpush",

  "source_urls": "[\\"http://yourURL.cn/live/normal_466247620*****3100448-upload-21
```

```
6b/playlist.m3u8\"]\n",\n\n"stream_id": "testvod",\n\n"task_id": "118145",\n\n"to_url": "rtmp://5000.livepush.myqcloud.com/live/testvod"\n}
```

TaskExit - Callback of the task stop event

```
{\n  "appid": 4,\n\n  "callback_event": "TaskExit",\n\n  "event_type": 314,\n\n  "interface": "general_callback",\n\n  "msg": "{\\\"message\\\":\\\"write packet error.\\\",\\\"code\\\":-22,\\\"task_exit_time\\\":0}\\\",",\n\n  "product_name": "pullpush",\n\n  "source_urls": "[\\\"http://yourURL.cn/live/normal_230753472*****21162358-upload-4\\\"]\n"}\n}
```

Note :

- Sequence of callbacks for relay tasks with “Video on-demand” as the source content: **TaskStart** - Callback of the task start event > **VodSourceFileStart** - Callback of the on-demand file’s start > **VodSourceFileFinish** - Callback of the on-demand file’s end.
- There is an interval of **up to 2s** between **TaskStart** and **VodSourceFileStart** callbacks.
- Callback settings are included in the relay task configuration.

User Guides for Common Third-Party Tools

Push via OBS

Last updated : 2022-05-05 15:15:12

Overview

Open Broadcaster Software (OBS) is a third-party open-source tool for live streaming. It's easy to use and free of charge, and it supports OS X, Windows, and Linux. OBS can be used in a wide range of scenarios to meet most live streaming needs without requiring additional plugins. You can download its latest version at the [OBS website](#).

Prerequisites

- You have installed [OBS Studio](#).
- You have activated [CSS](#) and [added a playback domain](#) with an ICP filing number in the console (for push, you can use the default domain we provide or add your own).

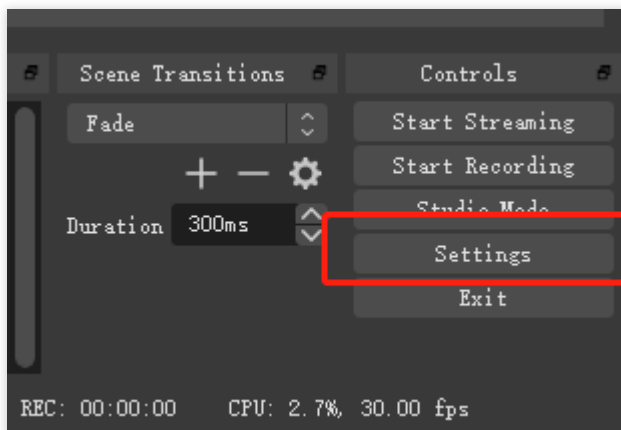
Getting a Push URL

1. Log in to the CSS console, click [Address Generator](#) in the left sidebar, and follow the steps below:
 - i. Select ***Push Domain**** for **Domain Type**.
 - ii. Select the domain name you have added in **Domain Management**.
 - iii. Enter an application name (`AppName`), which is used to distinguish applications under the same domain. The default value is `live` .
 - iv. Enter a custom stream name (`StreamName`), such as `live` .
 - v. Select the expiration time of the address, such as `2020-06-09 23:59:59` .
2. Click **Generate Address** to get an OBS push URL.

Configuring OBS for Push

Step 1. Configure the push URL

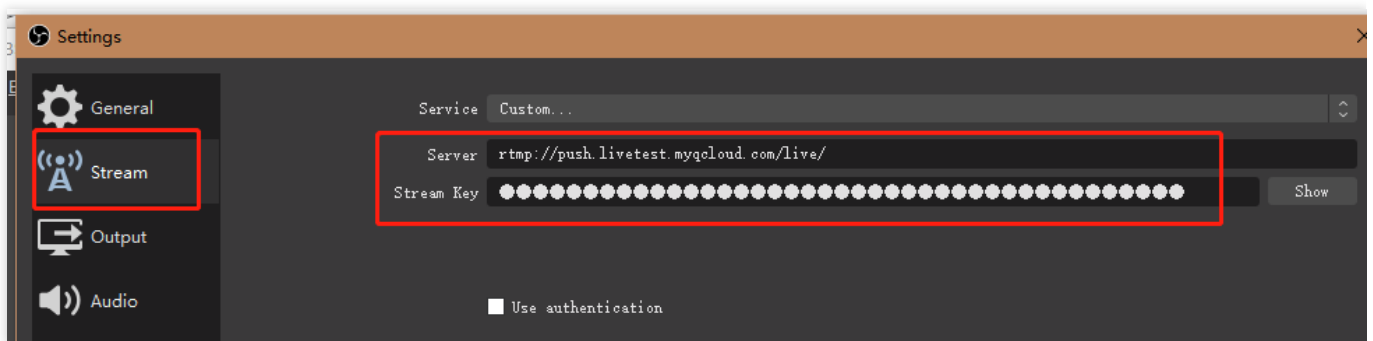
1. Open OBS and click **Controls** > **Settings** at the bottom to enter the settings page.



2. Click **Stream** and select **Custom** for **Service**.

3. Fill in the **Server** and **Stream Key** fields with the information obtained in [Getting a Push URL](#).

- Server: Enter the OBS push address (`rtmp://domain/AppName/`).
- Stream key: Enter the OBS push name (`StreamName?txSecret=xxxxx&txTime=5C1E5F7F`).



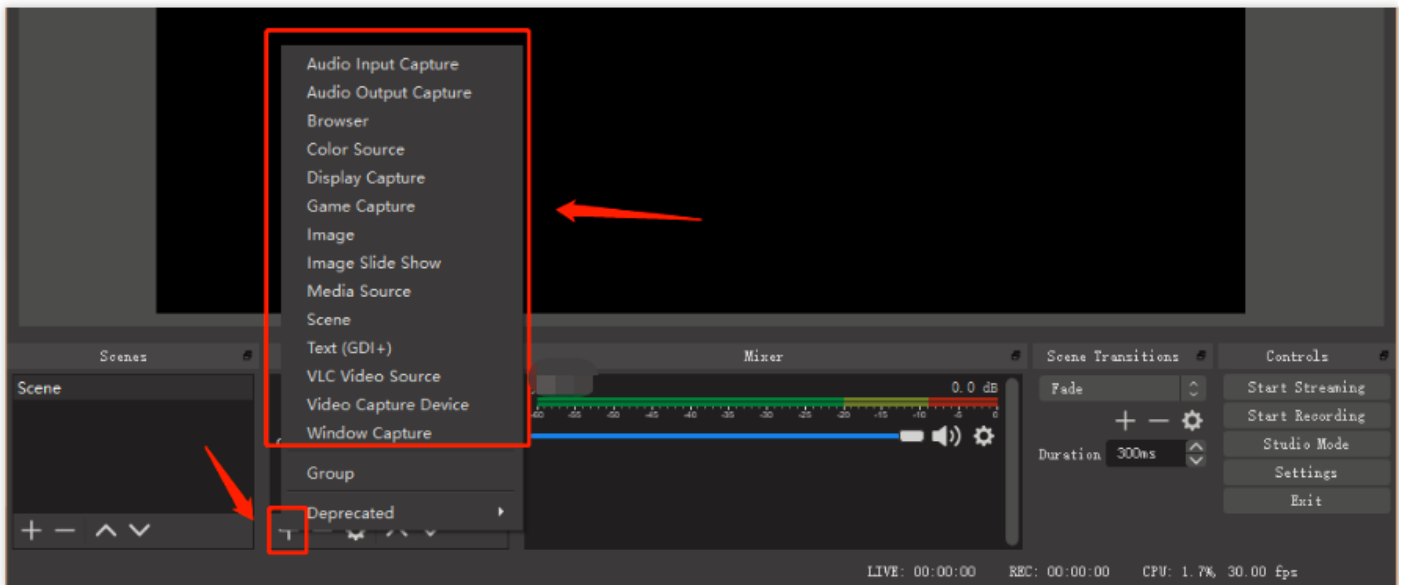
4. Click **OK** to save the information.

Step 2. Configure the source

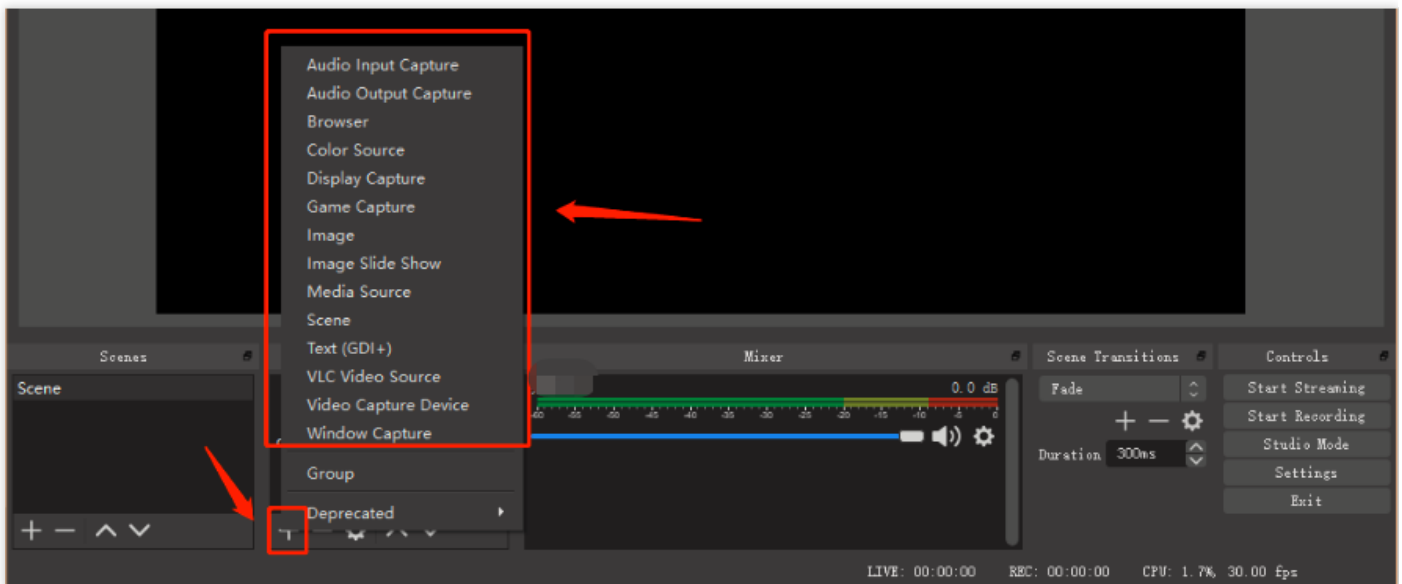
Note :

For bitrate, recording, and other settings, click **Tools** > **Auto-Configuration Wizard** in the top menu bar, and follow the instructions provided by OBS to complete the settings.

1. Find **Sources** in the menu bar at the bottom.



2. Click + and select a source that fits your needs, for example, **Display Capture**.



Common live streaming sources

Source	Description
Image	Publishing a single image
Image Slide Show	Publishing multiple images (you can determine the order of playback and whether to loop the playback)
Scene	Insertion of an entire scene as the source to enable various streaming effects

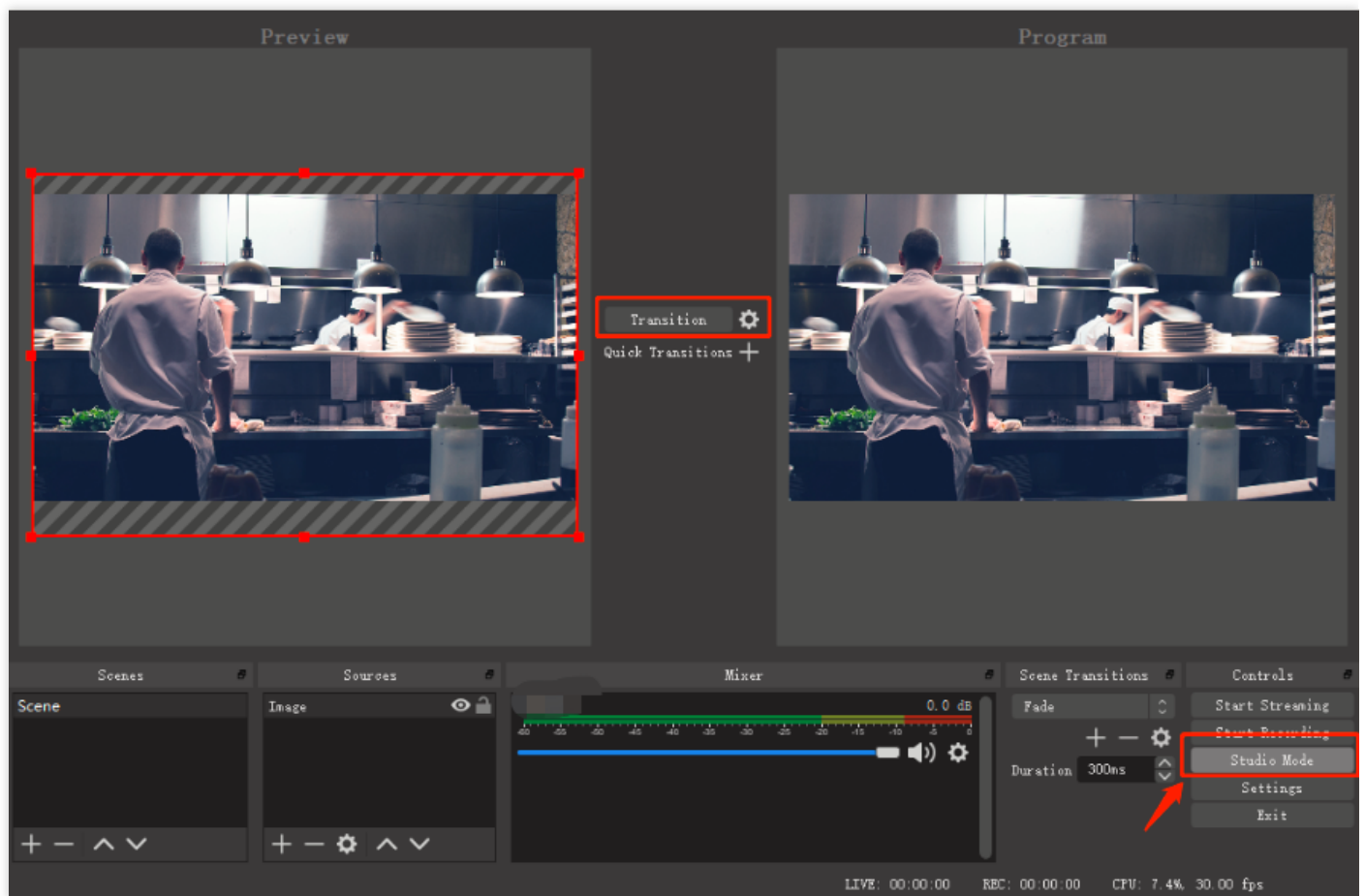
Source	Description
Media Source	Publishing a local file
Text	Adding real-time text to your stream
Display Capture	Capturing and publishing your monitor in real time
Game Capture	Streaming a game from a specified source in real time
Window Capture	Capturing and publishing the window you select in real time
Color Source	Adding a solid color to your scene. You can use this source for background colors or a global color tint by using the alpha channel.
Video Capture Device	Capturing and publishing the images captured by a camera in real time
Audio Input Capture	Audio live streaming (audio input device)
Audio Output Capture	Audio live streaming (audio output device)

Step 3. Use the studio mode

In studio mode, you can edit your current live stream in real time and configure transitions for scene swapping, minimizing the impact on user experience.

1. Click **Controls > Studio Mode** in the menu bar at the bottom.

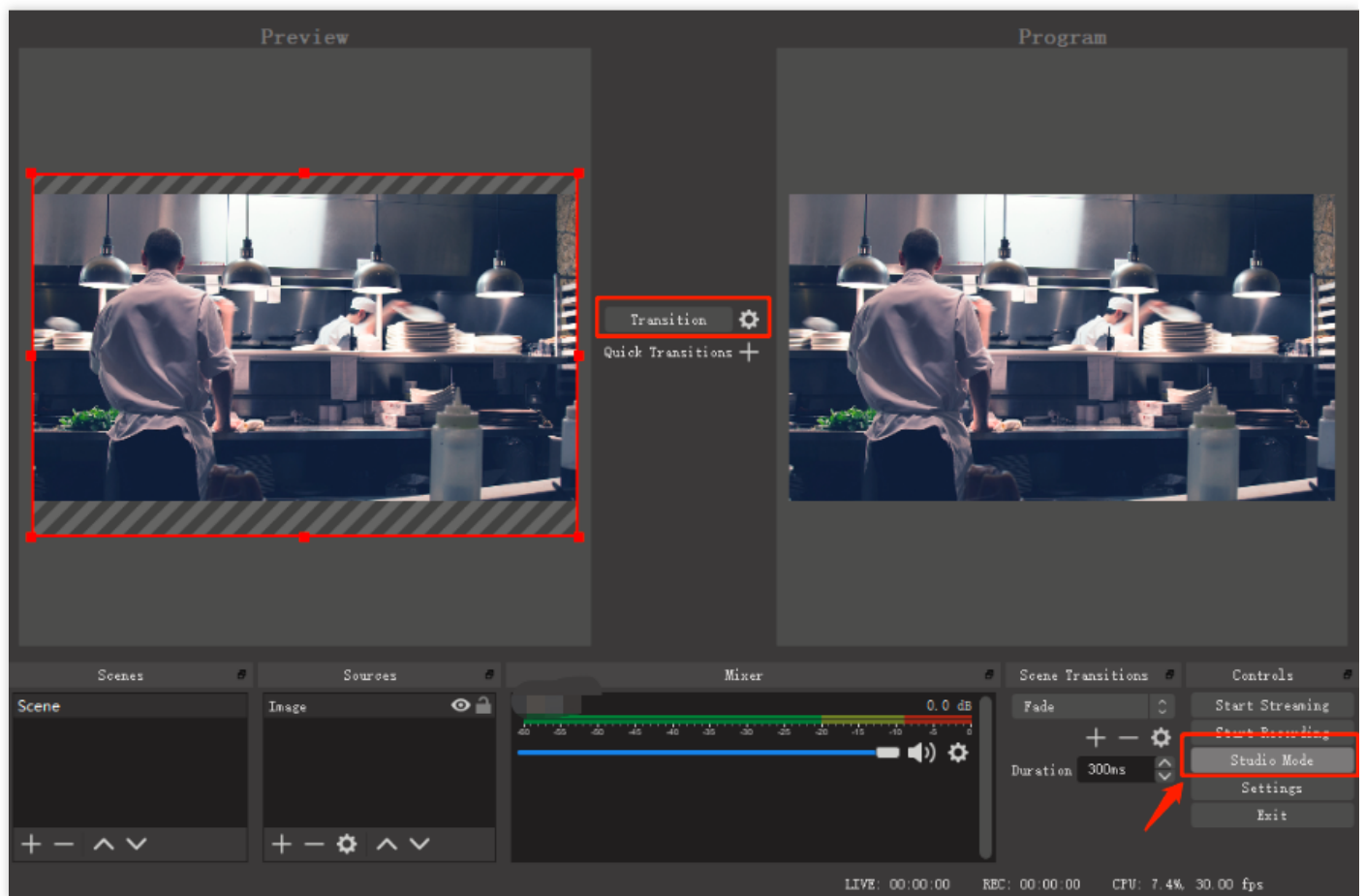
2. After editing, click **Transition** to swap the edit and live views.



Step 4. Start streaming


1. Find **Controls** in the menu bar at the bottom.

2. Click **Start Streaming** to push your video to the configured push URL.



Note :



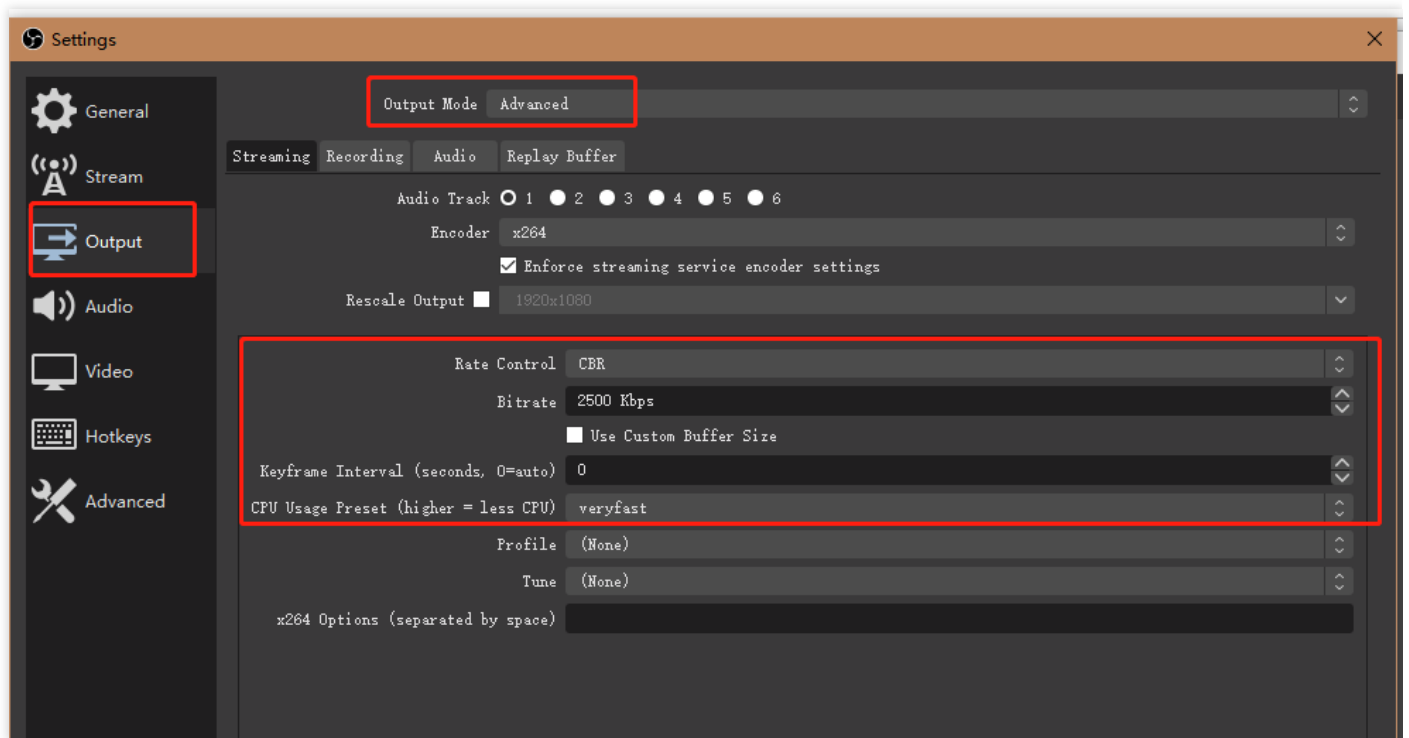
- If you see  at the bottom, the push is successful.
- To stop streaming, click **Stop Streaming**.

Other Push Settings

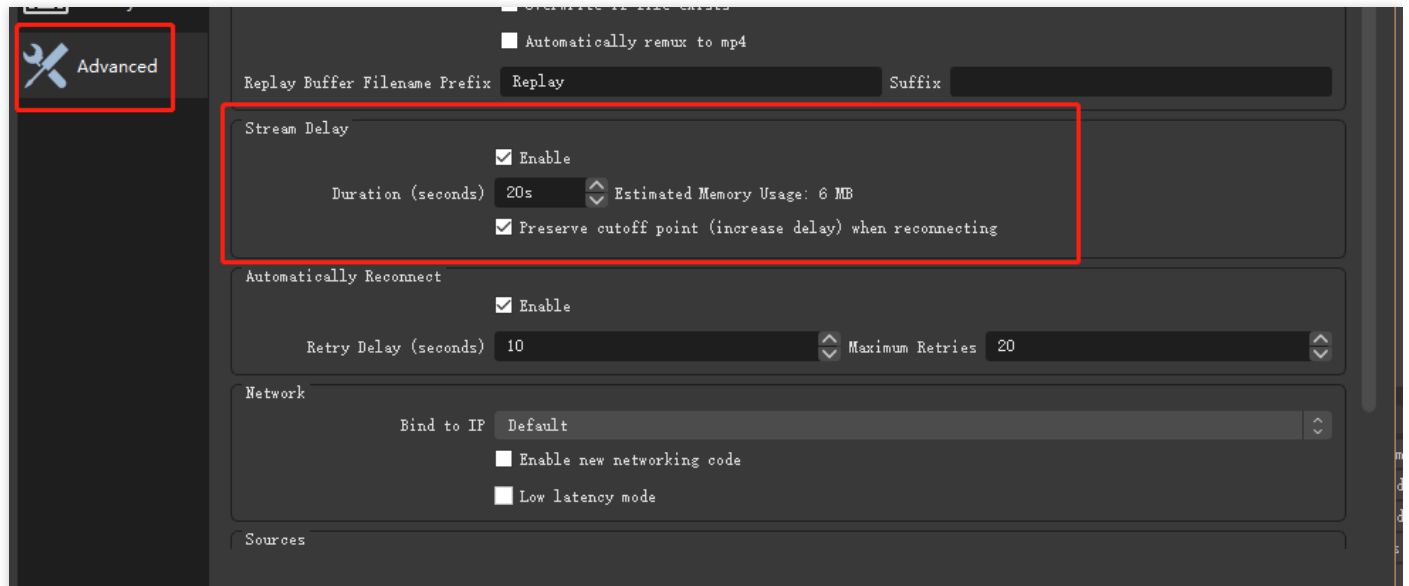
Streaming latency

1. Go to **Controls > Settings > Output**.

2. Select **Advanced** for **Output Mode** to set parameters including **Keyframe Interval**.



3. Click **Advanced** in the left sidebar to set **Stream Delay**:

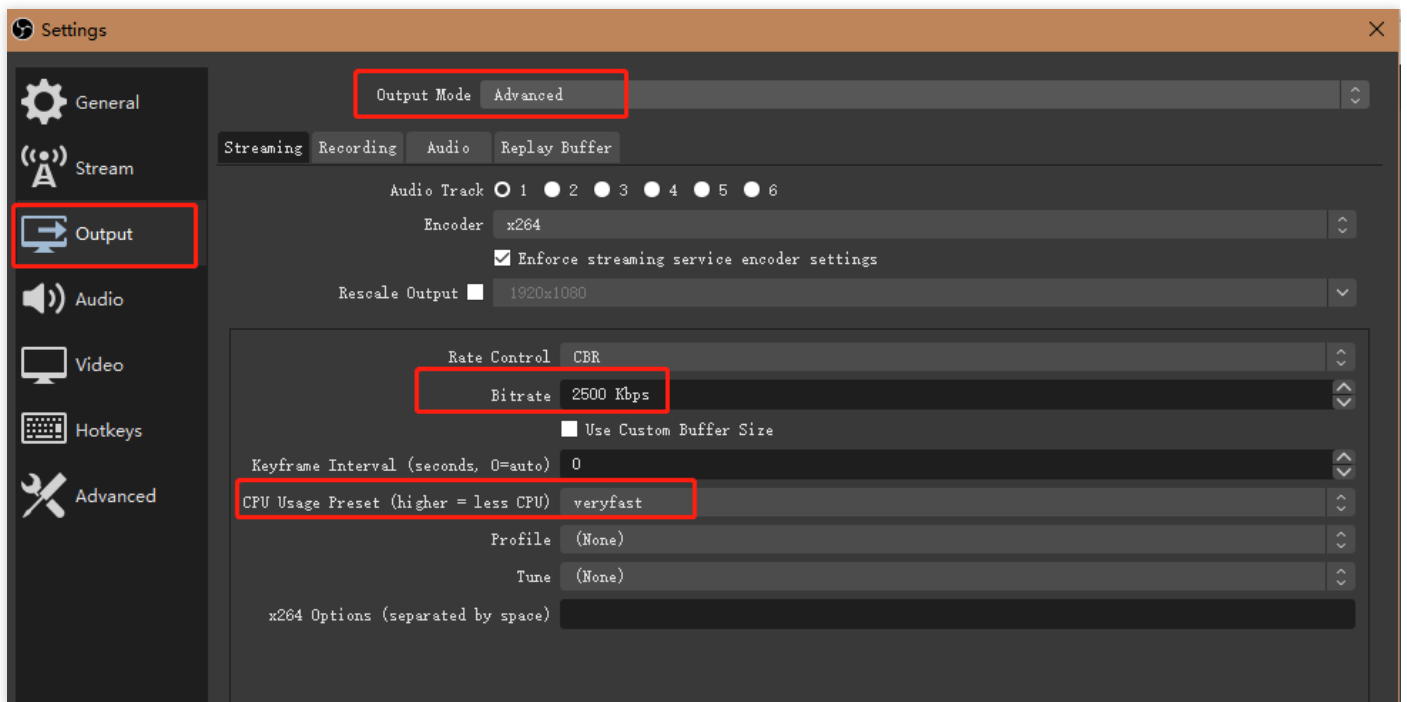


Local live recording

To record live streams to your local storage, follow the steps below:

1. Go to **Controls > Settings > Output**.

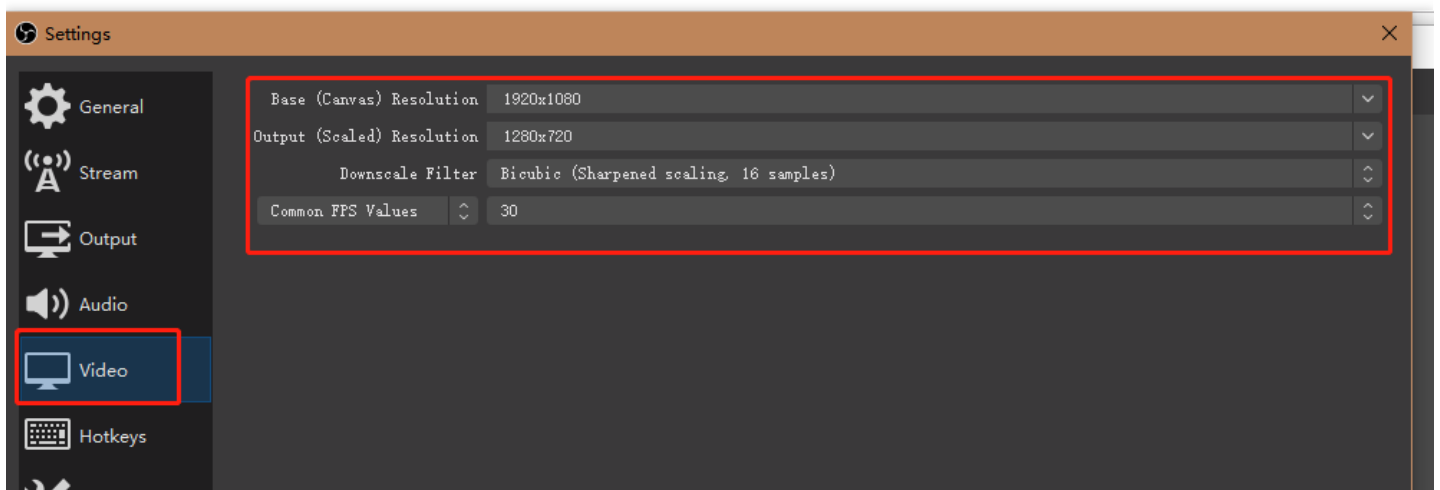
2. Complete the settings under **Recording** and click **OK**.



3. Click **Video** in the left sidebar to set the resolution and frame rate.

Note :

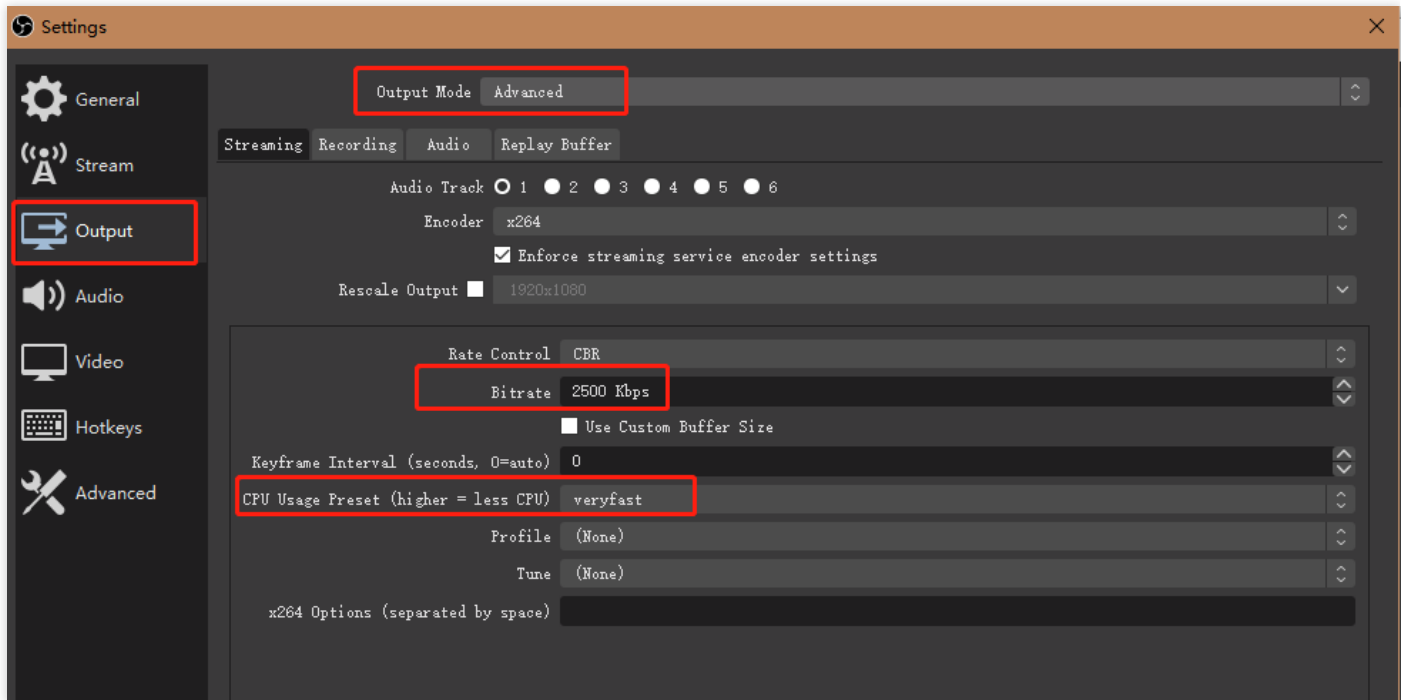
Resolution determines the clarity of video shown to viewers. The higher the resolution, the clearer the video. Frame rate (frames per second) determines playback smoothness. Typical frame rate falls in the range of 24 fps to 30 fps. Playback may stutter if frame rate is lower than 16 fps. Video games require higher frame rate and tend to stutter at a frame rate lower than 30 fps.



Transcoding

To change the video bitrate during streaming, follow the steps below:

1. Click **Controls > Settings** in the menu bar at the bottom.
2. Click **Output** in the left sidebar and select **Simple** for **Output Mode**.
3. Enter the bitrate you want to use and click **OK**.



More

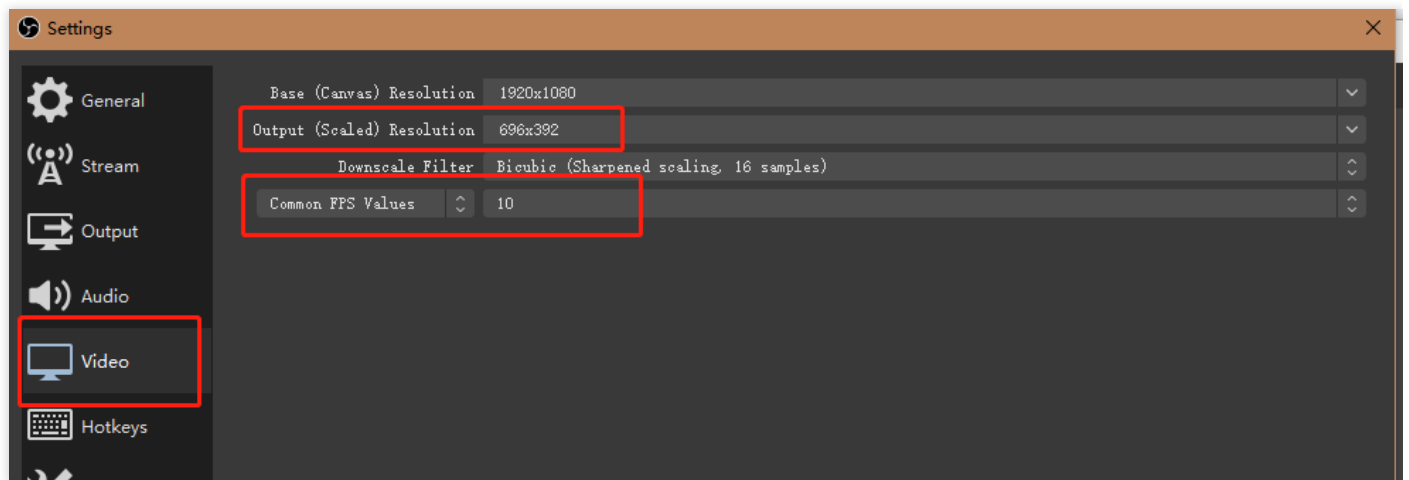
Audio-only push

According to OBS Forums, OBS Studio 23.2.1 and earlier versions do not support audio-only streaming.

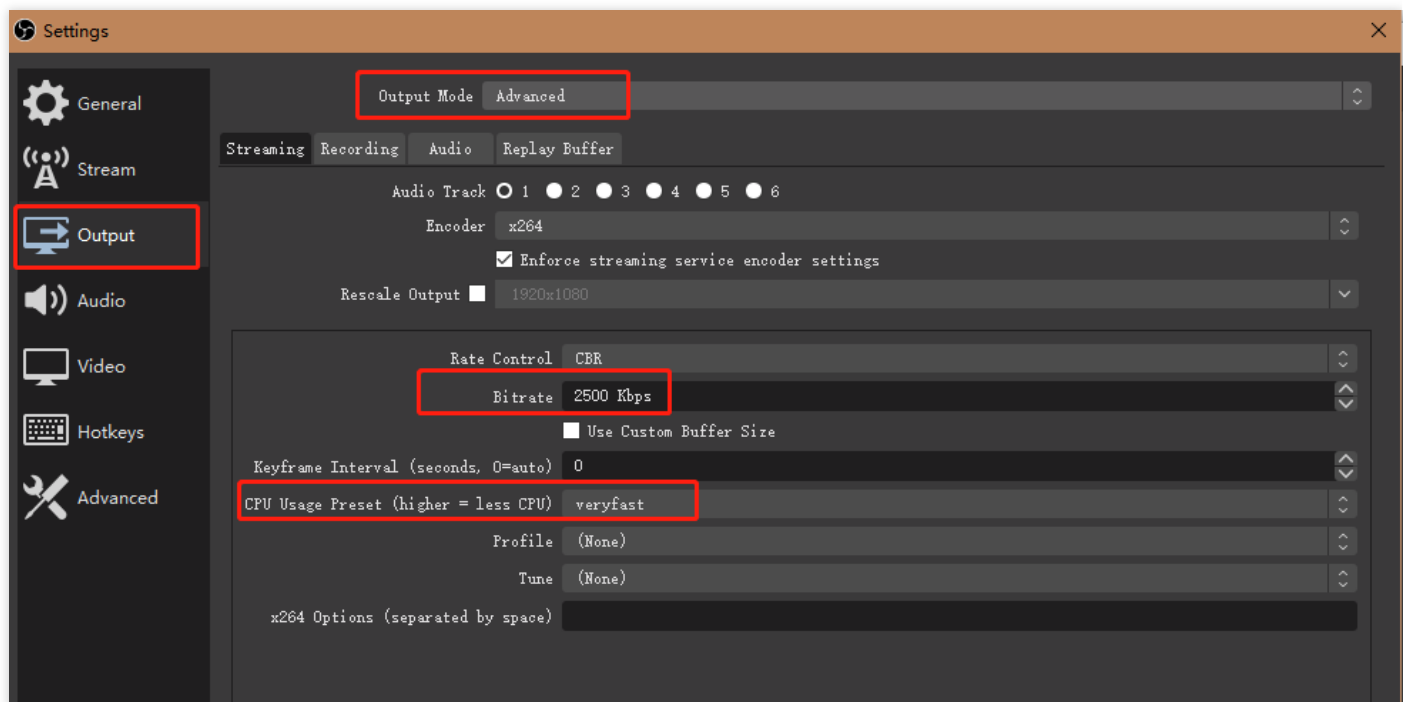
You can follow the steps below to implement a similar feature. The method uses a static canvas (blank screen or image) for video content. This means there will still be video data in the live stream. To reduce bandwidth usage, you can set the video frame rate and bitrate to the minimum values.

1. As instructed in [Configure the source](#), select **Audio Input Capture** as the source. Do not use a video or image source.
2. Go to **Controls > Settings > Video**.

3. Set **Base (Canvas) Resolution** and **Common FPS Values** to the minimum values and click **OK**.



4. Click **Output** in the left sidebar, configure the output as shown below (set **Bitrate** to the minimum value), and click **OK**.

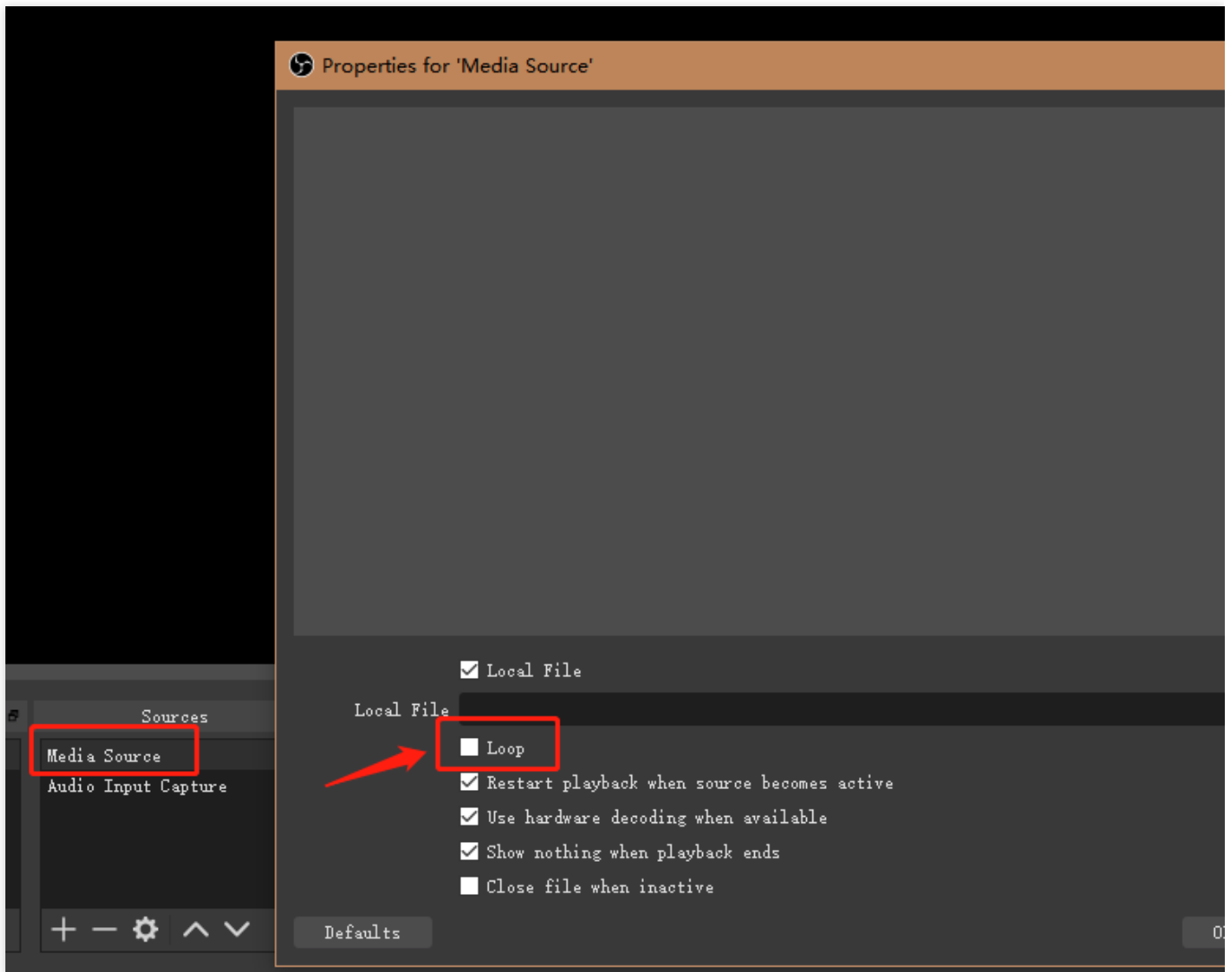


5. Start streaming as instructed in [Configuring OBS for Push](#). The audience will hear audio, while the video will be a blank screen or an image. Because the video bitrate is set to the minimum value, the bandwidth usage is significantly lower than that of video push.

Video looping

- Loop a single file

- i. Click **+** in **Sources** and select **Media Source**. In the pop-up window, choose a local file to stream, select **Loop**, and click **OK**.



- ii. Set the **Server** and **Stream Key** as instructed in [Configure the push URL](#).

You can use the following methods to check whether the push is successful:

- PC: Use [VLC](#) to play the stream.
- Mobile: Use the [MLVB](#) SDK to play the stream.

Note :

Mobile Live Video Broadcasting (MLVB) extends the services of CSS to mobile devices. It offers not only an RTMP SDK that allows quick integration, but also a one-stop, multi-cloud solution that integrates Tencent Cloud services including LVB, LEB, VOD, IM, and COS.

[Live Event Broadcasting \(LEB\)](#) is the ultra-low-latency version of LVB. It delivers superior playback

experience with millisecond latency and is suitable for scenarios with high requirements on latency, such as online education, sports streaming, and online quizzes.

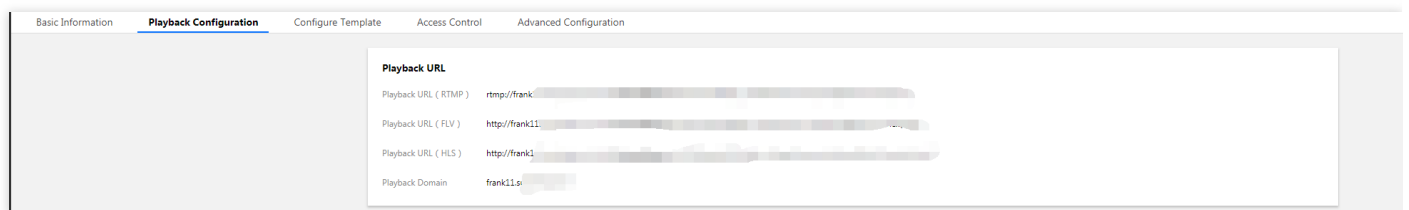
VLC Player

Last updated : 2021-03-12 11:24:02

VLC media player is an open-source cross-platform multimedia player and framework that can play back most multimedia files as well as DVDs, audio CDs, VCDs, and various streaming protocols free of charge. It supports operating systems such as OS X, Windows, Linux, iOS, Android, and Chrome OS and all common file formats for live streaming like RTMP, FLV, and M3U8. You can go to the [VLC official website](#) to download the latest version.

Playing back Video in VLC

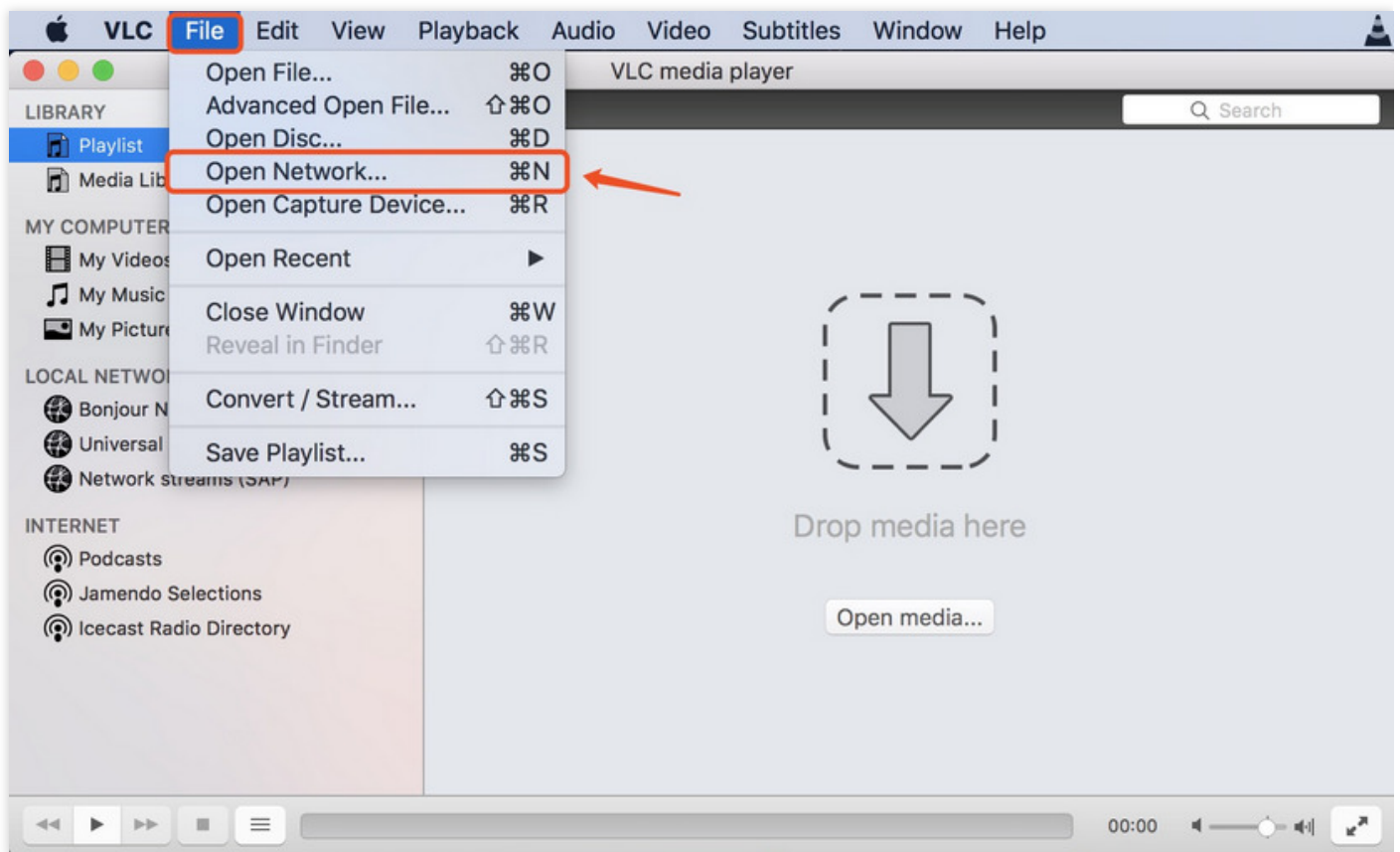
1. To get a [video stream playback address](#), select your filed playback domain name as needed in [Domain Management](#) and splice the playback address in the format specified in the playback configuration.



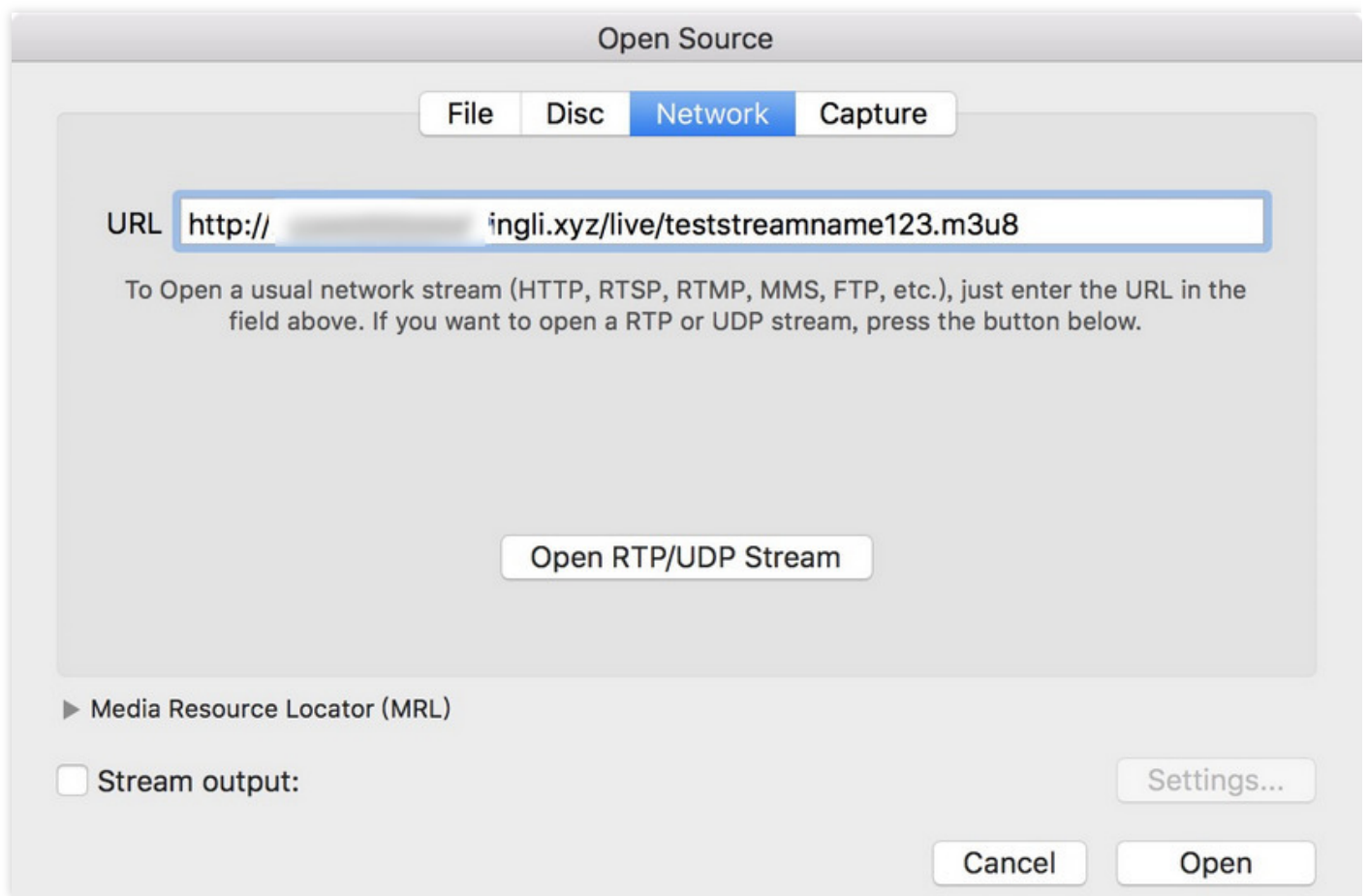
The screenshot shows the 'Playback Configuration' tab in a management console. It contains a table with the following data:

Playback URL	
Playback URL (RTMP)	rtmp://frank1
Playback URL (FLV)	http://frank11
Playback URL (HLS)	http://frank1
Playback Domain	frank11s

2. Launch VLC and select **File > Open Network**.



3. Enter the live stream playback address in the pop-up dialog box.



4. Then, click **Open** to confirm playback. If the pull from the playback address is normal, a playback window will pop up.

