Cloud Streaming Services

Feature Guide

Product Documentation

Contents

Feature Guide
Push and Playback
WebRTC Protocol Push Stream
OBS WebRTC live streaming

Last updated：2024-07-22 16:36:54

OBS (Open Broadcaster Software) supports WebRTC protocol for live streaming. This means that you can easily and quickly push live streams to Tencent Cloud Streaming Services using the WebRTC protocol on PC (Windows/Mac/Ubuntu) just like using the RTMP protocol.

In this article, we will take OBS v30.0 on Windows as an example to demonstrate how to use OBS for WebRTC protocol live streaming on a PC.

You can also use the WebRTC protocol for live streaming on the web. For specific steps on Web Push, please refer to the Web Push guide.

Supported Encoders

Video

H.264 (Best compatibility. Almost all clients can play it directly.)

AV1 (Compared with H.264, the compression rate is increased by over 40%, which can save more than 40% of bandwidth and storage costs at the same quality. Most browsers can play it directly.)

HEVC (Depends on streaming device encoders. Compared with H.264, it has a higher compression rate and poor browser support. The MLVB SDK can play it directly.)

Audio

Opus (Browser WebRTC can play it directly.)

Preparations

You can download OBS from the official website. Make sure you have installed an OBS version that supports WebRTC live streaming (v30.0 or higher, AV1 or HEVC encoding requires v30.2 or higher. **Please do not use v30.1.x, as this version has introduced a bug causing video image corruption in poor network conditions**). If you are unable to upgrade your OBS version, please refer to Using OBS Plugin for WebRTC Live Streaming, or consider using RTMP for streaming.

You have activated CSS and prepared a registered domain to be added as a Push Domain (you can use the default push domain provided by the system or add a custom domain for streaming).

Notes

Please do not use OBS v30.1.x, as this version has introduced a bug causing video image corruption in poor network conditions.

CSS by default provide a test domain `xxxx.livepush.myqcloud.com`, which you can use for live streaming tests. However, it is not recommended to use this domain as the push domain in your production environment. When using the WebRTC protocol for live streaming, each push domain has a default limit of 1000 concurrent streams. If you need to exceed this limit, you can contact us by submitting a ticket to request an increase.

Get WebRTC Push Address

1. Log in to the CSS Console, Select **CSS Toolkit** > **Address Generator**, and perform the following configurations:

1.1 Select URL Type: **Push Address**.

1.2 Choose the domain that you have already added to the Domain Management section.

1.3 AppName is used to differentiate the address paths of multiple apps under the same domain, with the default value being "live".

1.4 Enter a custom `StreamName`, for example: `Stream_01`.

1.5 You need to choose an encryption type, please consider your security requirements and performance trade-offs. For **Type**, you can choose between **MD5** or **SHA256**, with **MD5** being the default option.

1.6 Select Expiration Time, for example: `2024-07-19 15:36:35`.

2. Click **Generate Address** to obtain the WebRTC push address.

Address Generator

URL Type *	● Push Address ○ Playback Address ○ Push and playback URLs NEW ⓘ
Select domain name *	[_____].com ▾
AppName *	live ⓘ
	Use "live" by default. Only letters, digits, and symbols are supported.
StreamName *	Stream_01 ⓘ
	Only supports letters, digits, and symbols
Type	● MD5 ○ SHA256
Expiration Time	2024-07-19 15:36:35 📅
	The actual expiration time of the playback URL is the timestamp selected plus the validity period of the authentication key.

Generate Address   Splice manually   History

Live streaming URLs   ⊘ The URLs are automatically saved to the browser cache and will be deleted when you clear the cache.

URL Type	Push Address
Validity Period	2024-07-19 15:36:35 (UTC+08:00) reference documentation ⧉
RTMP URL	rtmp://[____].com/live/Stream_01?txSecret=6eb7634881cf678abaa76ace6c8d5327&txTime=669A1783 Copy QR code
OBS server	rtmp://[____].com/live/ Copy
OBS stream key	Stream_01?txSecret=6eb7634881cf678abaa76ace6c8d5327&txTime=669A1783 Copy
WebRTC URL **Millisecond latency**	webrtc://[____].com/live/Stream_01?txSecret=6eb7634881cf678abaa76ace6c8d5327&txTime=669A1783 Copy Web Push Web push doc ⧉ OBS pu
SRT URL	srt://[____].com:9000?streamid=#!::h=[____]com,r=live/Stream_01,txSecret=6eb7634881cf678abaa76ace6c8d5327,txTime=669A1783 Copy
RTMP over SRT URL	rtmp://[____].com:3570/live/Stream_01?txSecret=6eb7634881cf678abaa76ace6c8d5327&txTime=669A1783 Copy

OBS Online Streaming

Step 1: Set WHIP Server Address and WebRTC Push Address

1. Open **OBS**, and you can access the settings interface by clicking on the **Settings** button in the bottom toolbar control.
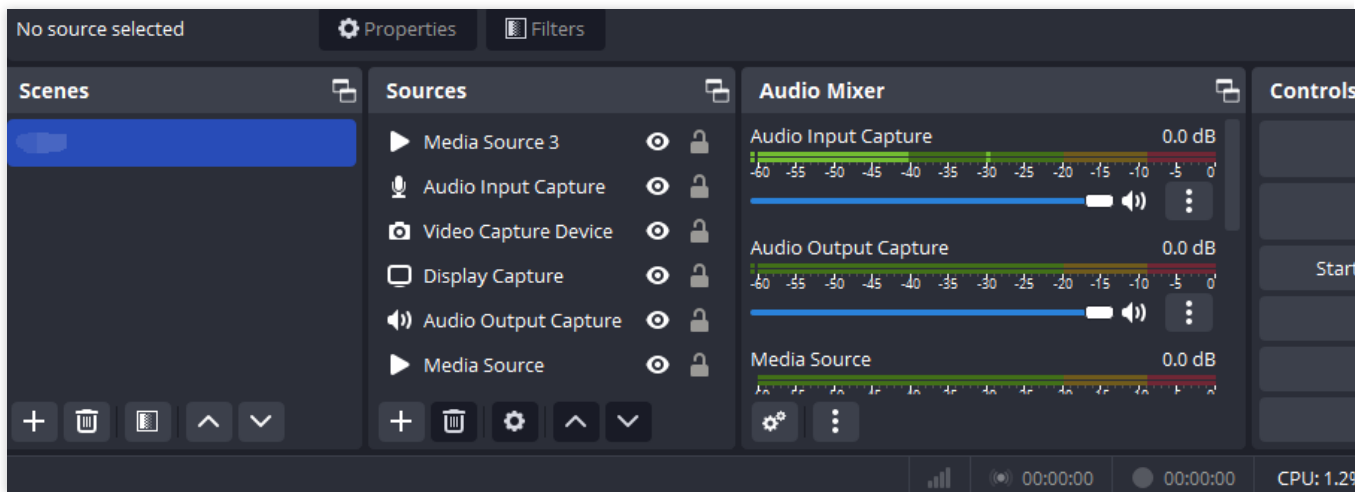
2. Click on **Stream** to enter the live streaming address settings interface.

Select the **Service** type as: **WHIP**

In the "**Server**" field, enter the Tencent Cloud Live WHIP server address:

Default address: `https://webrtcpush.tlivewebrtcpush.com/webrtc/v2/whip`

Backup address: `https://webrtcpush.tlivewebrtcpush2.com/webrtc/v2/whip`

In the "**Bearer Token**" field, enter the WebRTC push address you obtained, for

example: `webrtc://domain/AppName/StreamName?txSecret=xxxxx&txTime=xxxxx`
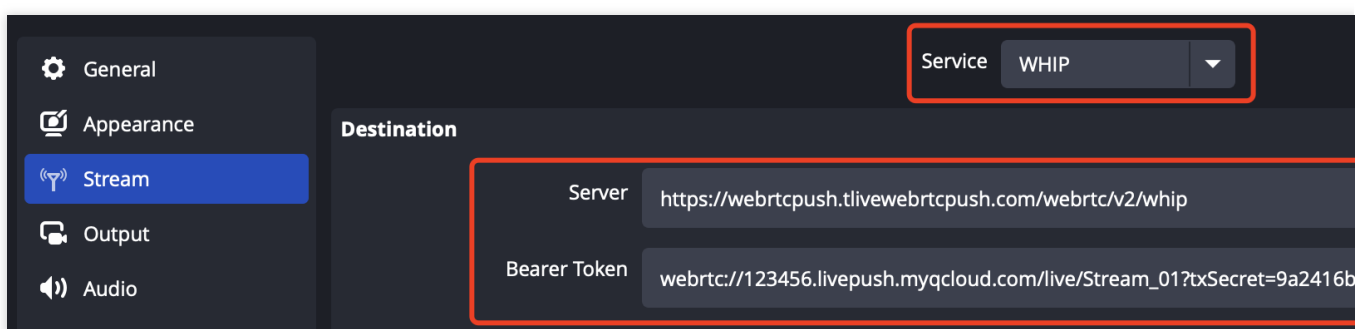
Step 2: Set Streaming Parameters

1. Navigate to the streaming parameter settings interface by clicking on Control > **Settings** > **Output**. Select the output mode as **Advanced**.

2. Select the **Streaming** option and configure parameters such as encoder, bitrate, keyframe interval, and others.

3. If you are using the **LEB WebRTC** solution at the playback end, please perform push settings according to the following configuration according to different encoding protocols:

Using H.264 video encoder

For the video encoder, please select **x264** or another H.264 encoder supported by your device.
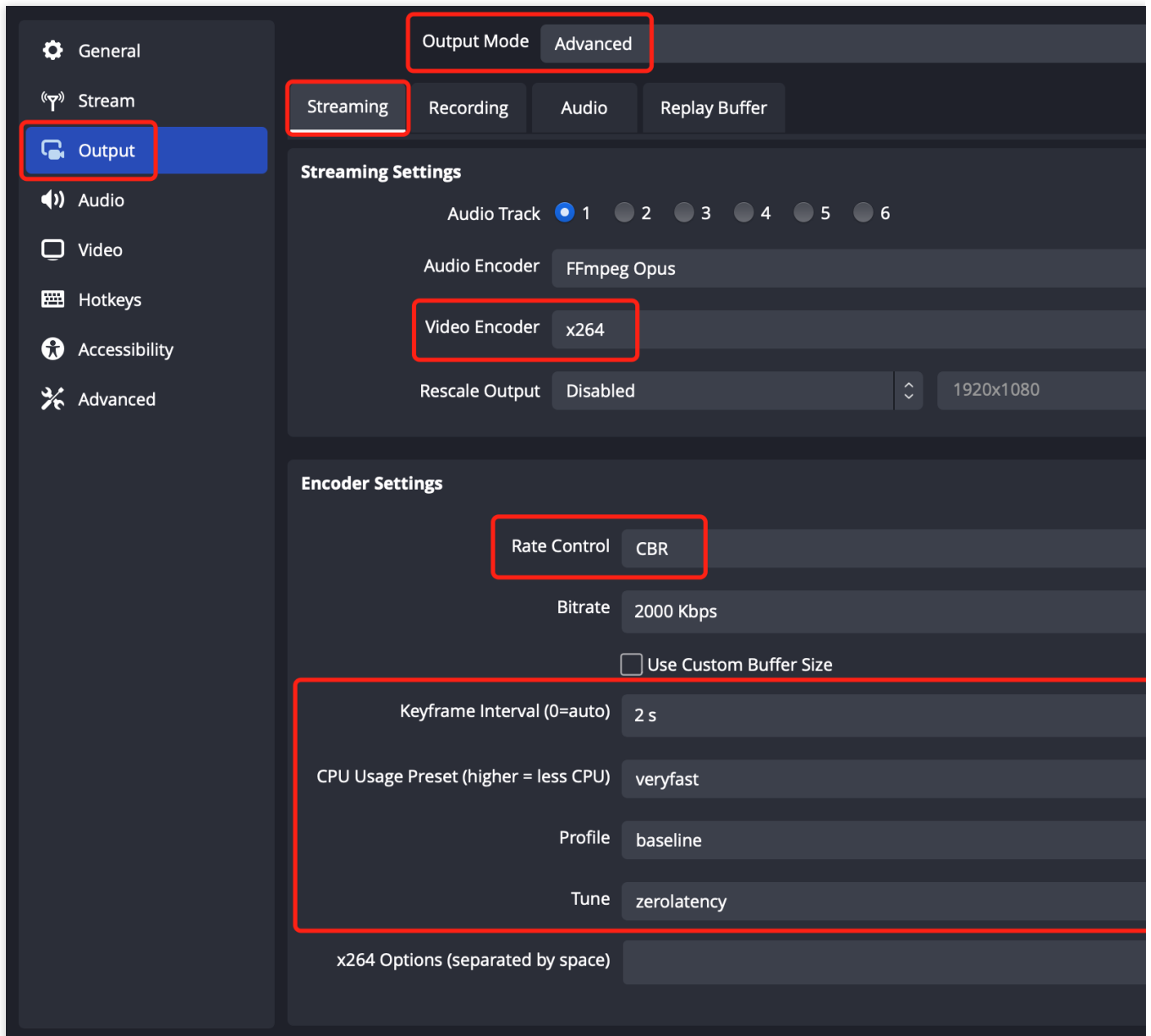
Using AV1 video encoder

For the video encoder, please select **AOM AV1** or another AV1 encoder supported by your device.
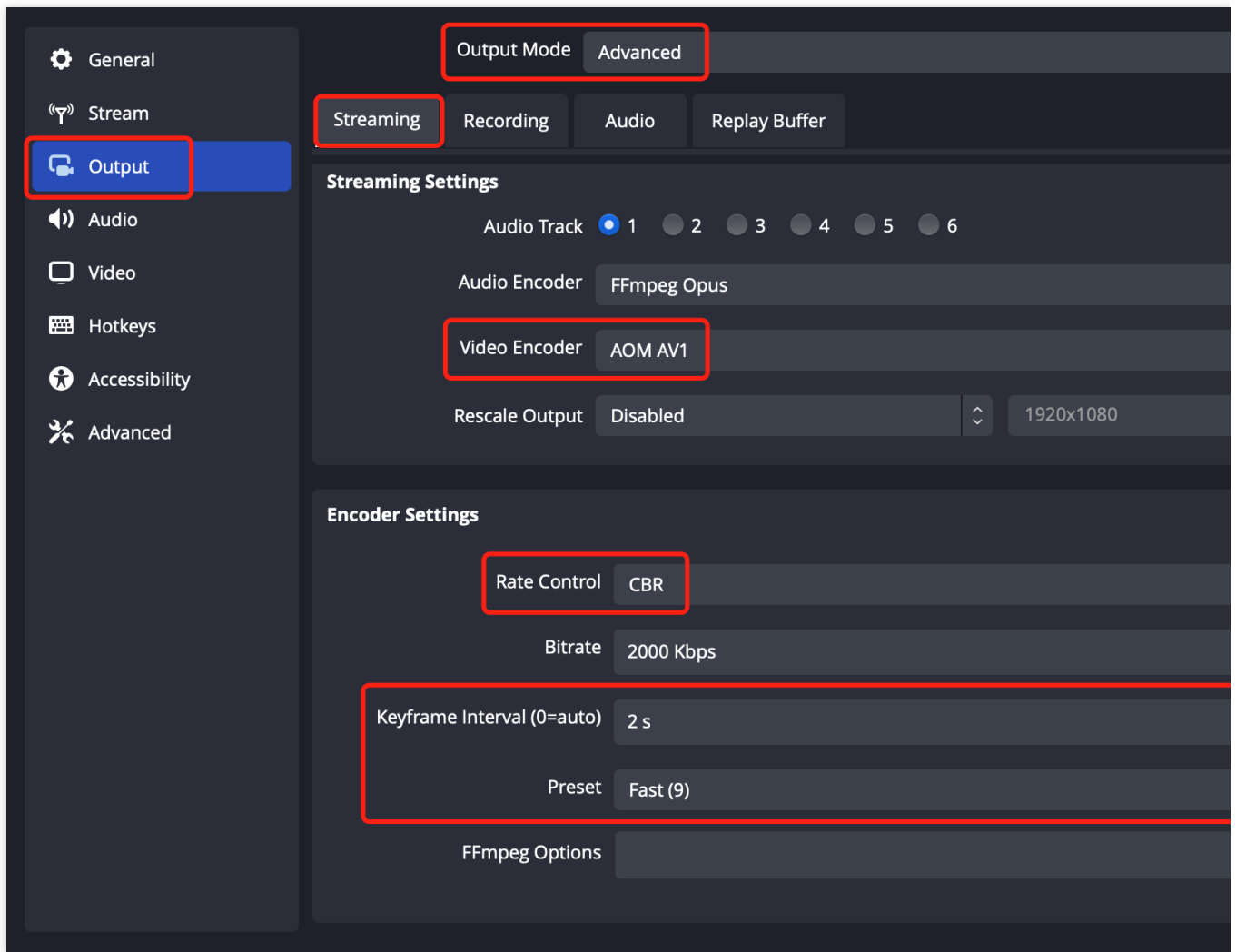
Using HEVC video encoder

For the video encoder, please select HEVC encoder supported by your device, such as Apple VT HEVC Hardware/Software Encoder.
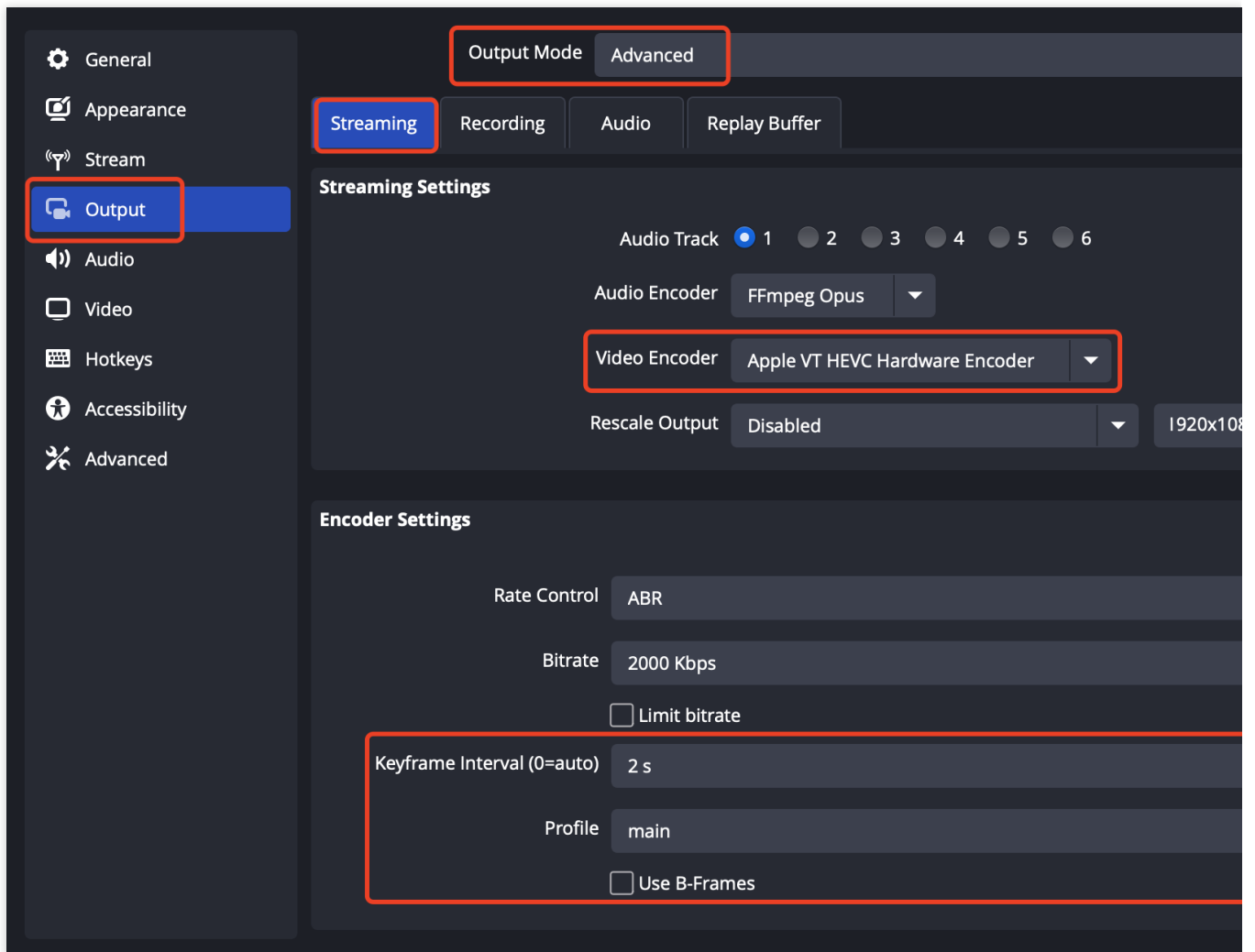
Note:

The default audio encoding format for OBS WebRTC push is Opus. When you use LEB for playback on the Web, since the Web supports Opus audio format by default, there is no need for cloud-based audio transcoding (from AAC to Opus) as required when pushing with RTMP protocol, and playback can be done directly.

For more information about parameter **x264 option**, please refer to: x264 multi-slice encoding parameters.

4. Click on **Stream** to enter the live streaming address settings interface.

Step 3: Start Live Streaming

Click on Control > **Start Streaming** in the bottom toolbar of OBS to push the media stream to the WebRTC address you have set up.
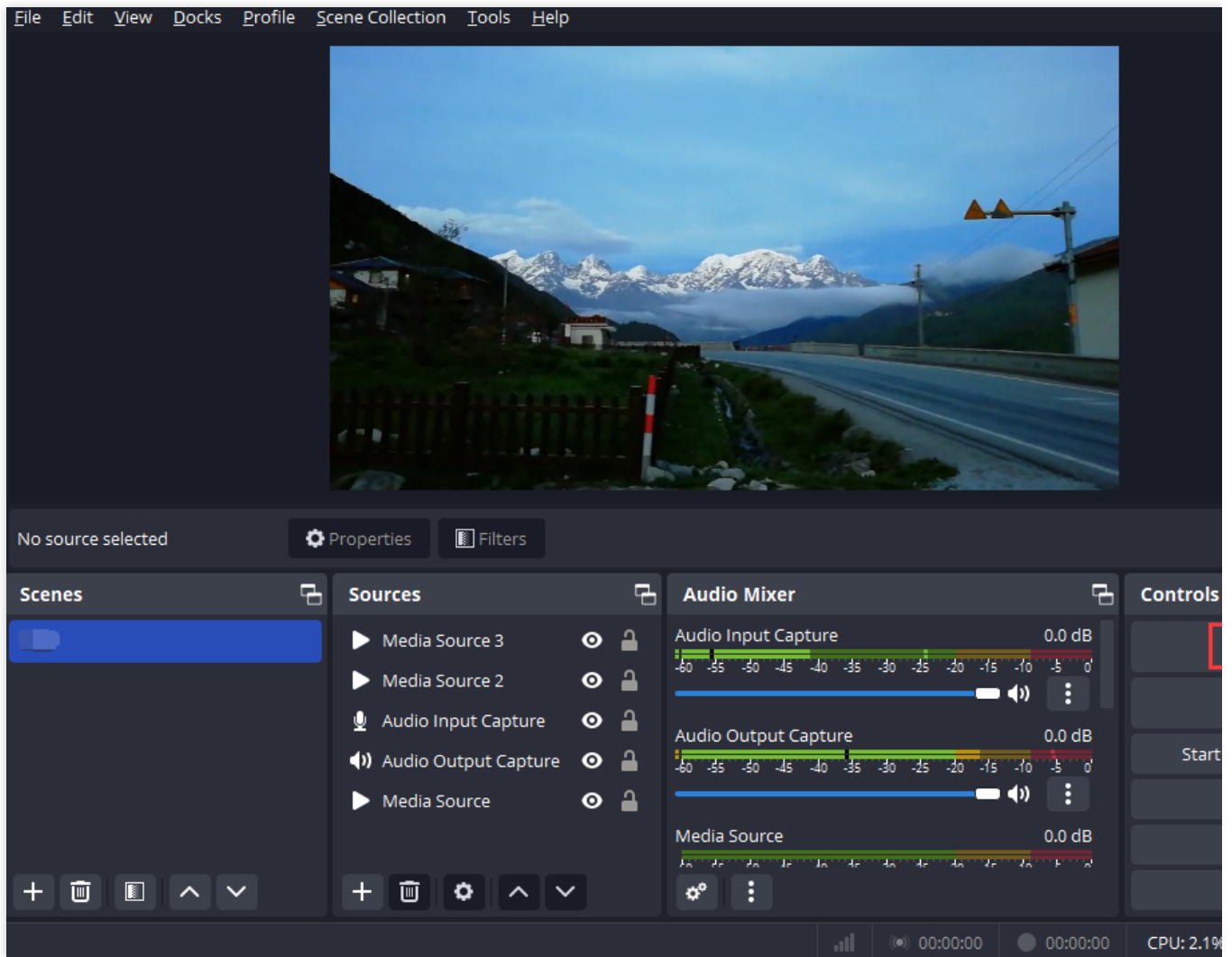
Note:

OBS uses the WHIP (WebRTC-HTTP Ingestion Protocol) for WebRTC live streaming. WHIP is a standard HTTP-based protocol that allows you to push/pull WebRTC real-time streams to/from streaming servers or CDNs using HTML5 and various clients.

If you need to know more about the usage of OBS streaming, you can refer to Push via OBS 。

Comparing end-to-end latency between OBS WebRTC live streaming and RTMP protocol live streaming:
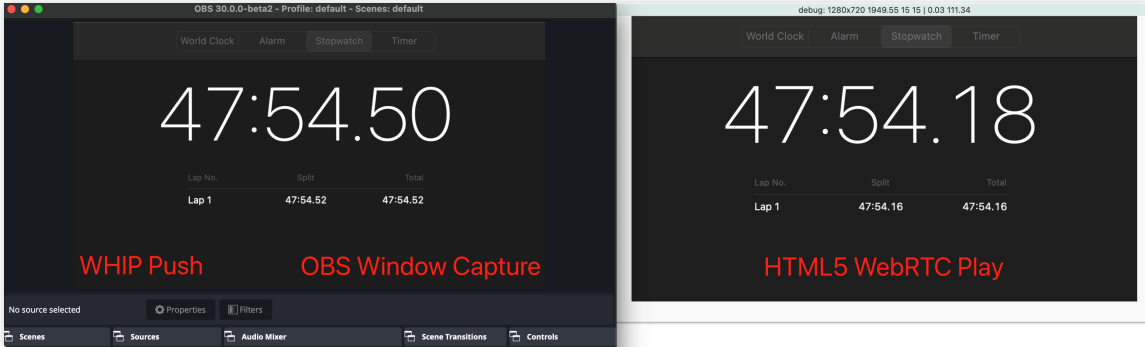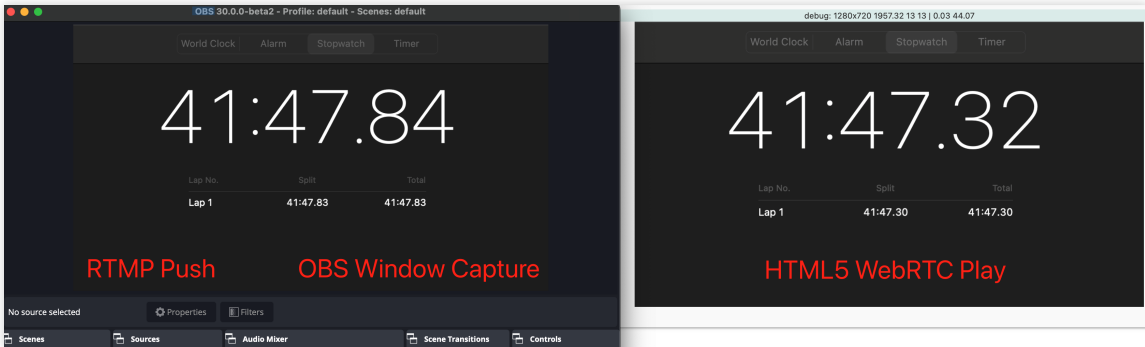
End-to-end latency is affected by multiple factors such as device performance, encoding parameters, network transmission, and player cache, and may fluctuate within a certain range during the live streaming process. In this scenario, we will compare the difference in end-to-end latency between x264 multi-slice encoding and single-slice encoding. The push end uses the OBS tool, and the playback end adopts Web LEB.

x264 Multi-slice Encoding Parameters:
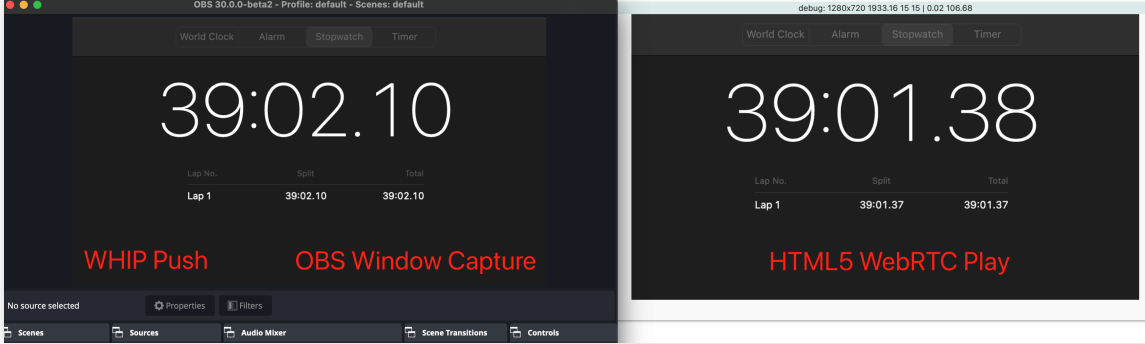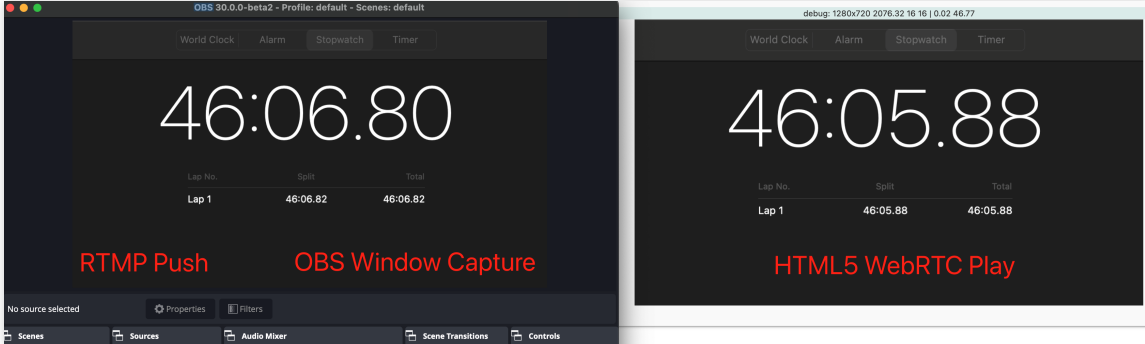
When you configure the zerolatency (zero latency) mode in OBS's Tune options, OBS will automatically enable **multi-slice encoding** to improve encoding speed and reduce latency.

If you are using the LEB Web solution at the playback end, please note that some older version browsers' WebRTC may have compatibility issues with **multi-slice encoding**. In this case, enabling **multi-slice encoding** may cause a mosaic screen effect on the playback end in weak network packet loss scenarios. To avoid this issue, you can configure `sliced_threads=0` in the x264 options to disable **multi-slice encoding.** However, disabling **multi-slice encoding** may introduce an additional few hundred milliseconds of encoding latency. Therefore, when configuring, please balance compatibility and latency based on your actual needs.

Push method	Describe
WHIP push streaming	About 300~500ms.
RTMP streaming	About 450~650ms.

Single Slice Encoding

When using single-slice encoding, the end-to-end latency is greatly affected by device performance. The following data is for reference only, and the actual latency may vary due to device performance and other factors:

Push method	Describe
WHIP push streaming	About 700~850ms.
RTMP streaming	About 850~1000ms.

Note:

These latency data may fluctuate due to factors such as network conditions, encoding parameters, and player buffering. In practical applications, you can adjust the encoding parameters and streaming protocols according to your requirements and device performance to achieve the desired latency and image quality performance.

Using OBS Plugin for WebRTC Live Streaming

OBS versions lower than v30.0 Beta 1 do not support WebRTC live streaming directly. Tencent Cloud Streaming Services provide an integrated OBS plugin solution for WebRTC live streaming.

Notes

The current requirements for OBS versions are 26.0 ≤ OBS version ≤ 29.0.2. You can download and install the appropriate version from the OBS Archived Versions page.

The WebRTC live streaming plugin currently only supports the Windows platform. If you want to implement WebRTC live streaming on Mac/Linux, you can use Web Push.

Configure OBS Plug-in

1. **Configure plug-in data**.

1.1 Download the OBS Plugin and, based on your local OBS version, move the two `services.json` and `package.json` files from the corresponding version's data folder to the **data > obs-plugins > rtmp-services** directory, replacing the existing files. (By default, `obs-studio` is installed on the C drive, with the corresponding directory being `C:\\obs-studio\\data\\obs-plugins\\rtmp-services`. Please configure it according to your actual situation.)
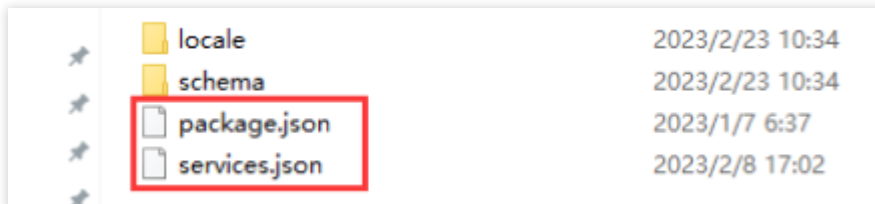
1.2 Copy the two JSON files mentioned above to the `C:\\Users<computer_name>\\AppData\\Roaming\\obs-studio\\plugin_config\\rtmp-services` directory and overwrite the existing files. (Replace < `computer_name` > with your actual computer name.)

2. **Configure the plug-in dynamic library.**

Move the .dll file from the `obs-plugins\\64bit` folder to the corresponding **obs-studio > obs-plugins > 64bit** directory. (By default, `obs-studio` is installed on the C drive, with the corresponding directory being `C:\\obs-studio\\obs-plugins\\64bit`. Please configure it according to your actual situation.)

Configure Push Link

1. Generate WebRTC push address.

Log in to the CSS Console,Select **CSS Toolkit** > Address Generator to generate a push address. For detailed operations, please refer to the Address Generator guide.

2. **Configure OBS streaming service.**

2.1 Open OBS, and you can access the settings interface by clicking on the **Settings** button in the bottom toolbar control.

2.2 Click on **Stream** to enter the Stream Settings tab, select the service type as Tencent WebRTC, set the server to Default, and enter the previously generated WebRTC Push Address in the Stream Key field.

2.3 The current OBS plugin supports OBS version 29. To start live streaming, click on **Streaming** to enter the Stream Settings tab, select the service type as `Tencent WebRTC`, set the server to `Default,` and enter the previously generated WebRTC Push Address in the Stream Key field.

Web Push

Last updated：2023-11-30 16:18:27

The TXLivePusher SDK is used to push streams for LEB (ultra-low latency streaming). It can push audio and video the browser captures from the camera, screen, or a local media file to live streaming servers via WebRTC.

You can use the WebRTC protocol for live streaming on the web. On the PC side, you can also use the OBS tool for WebRTC live streaming. For specific operation methods, please refer to the OBS WebRTC Live Streaming related content.

The advantage of using Web for WebRTC live streaming is that there is no need to install additional software, and you can operate directly in the browser. This article will introduce the operation method for live streaming using **Web**.

Note

With WebRTC, each push domain name can be used for up to **1000 concurrent streams** by default. If you want to push more streams, please submit a ticket.

Basics

Below are some basics you need to know before integrating the SDK.

Splicing push URLs

To use Tencent Cloud live streaming services, you need to splice push URLs in the format required by Tencent Cloud, which consists of four parts.

An authentication key is not required. You can enable push authentication if you need hotlink protection. For details, see Splicing Live Streaming URLs.

Browser support

Web live streaming is based on WebRTC implementation and relies on the operating system and browser support for WebRTC. Currently, the latest versions of Chrome, Edge, Firefox, and Safari browsers support Web live streaming.

Note：

The audio/video capturing feature is poorly supported on mobile browsers. For example, mobile browsers do not support screen recording, and only iOS 14.3 and later allow requesting camera access. Therefore, the push SDK is mainly used on desktop browsers. The latest version of Chrome, Firefox, and Safari all support push for LEB.

SDK Integration

Step 1. Prepare the page

Add an initialization script to the (desktop) page from which streams are to be pushed.
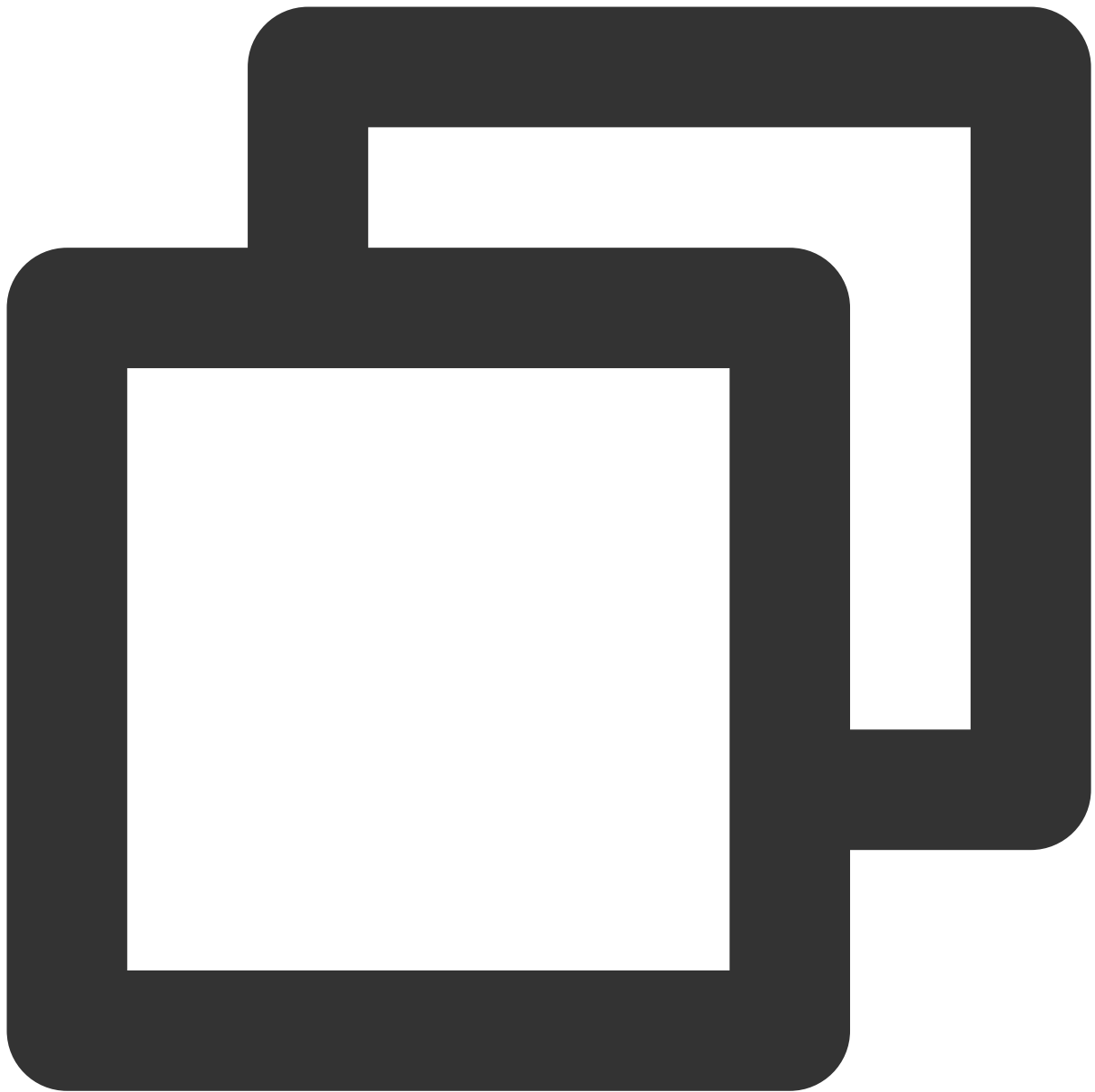
```
<script src="https://video.sdk.qcloudecdn.com/web/TXLivePusher-2.1.1.min.js" charse
```

Note

The script needs to be imported into the `body` part of the HTML code. If it is imported into the `head` part, an error will be reported.

Step 2. Add a container to the HTML page

Add a player container to the section of the page where local video is to be played. This is achieved by adding a div and giving it a name, for example, `id_local_video` . Local video will be rendered in the container. To adjust the

size of the container, style the div using CSS, Below is an example code:
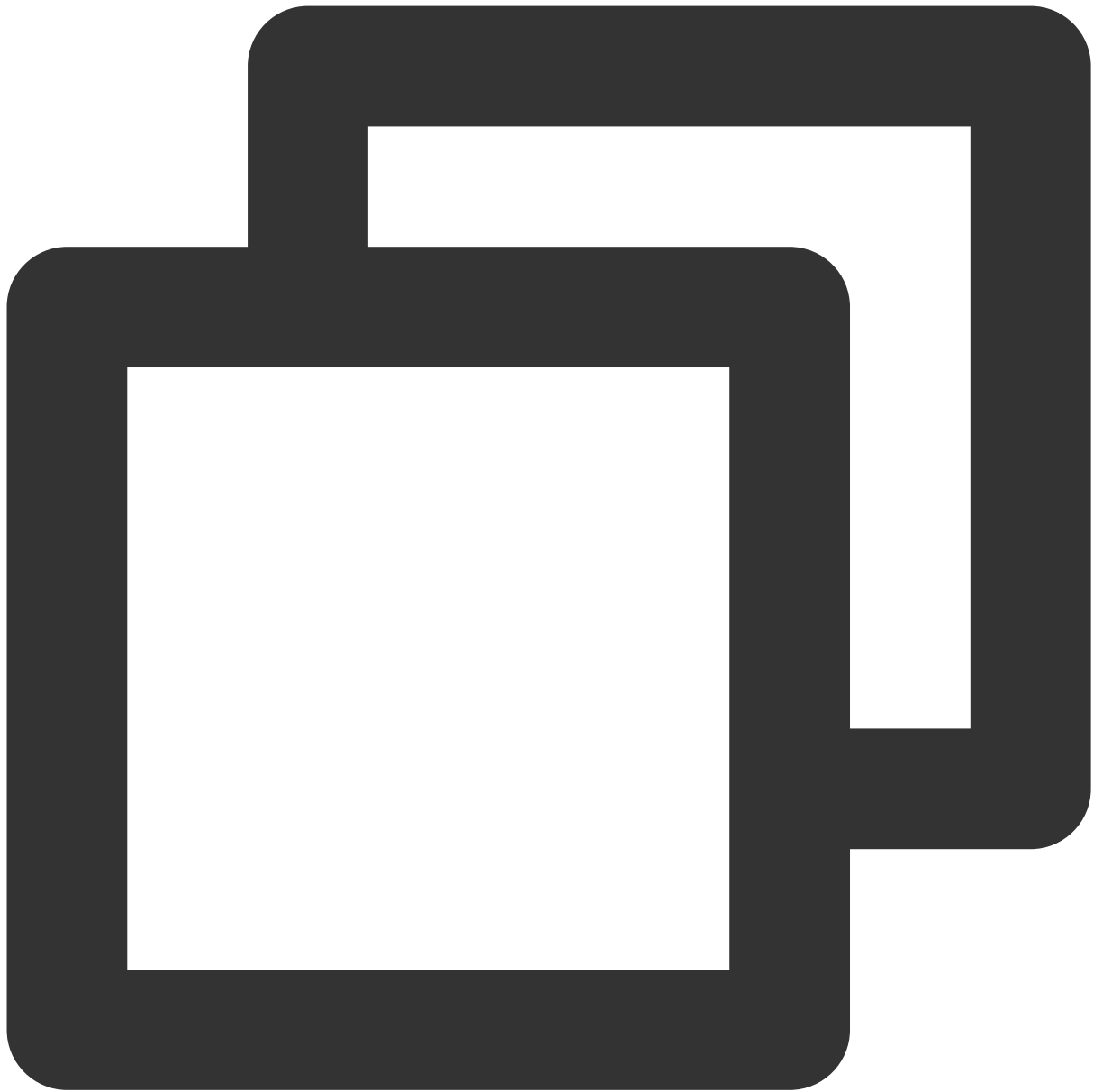

```
<div id="id_local_video" style="width:100%;height:500px;display:flex;align-items:ce
```

Step 3. Push streams

1. **Generate an instance of the push SDK:**

Generate an instance of the global object `TXLivePusher` . All subsequent operations will be performed via the instance.
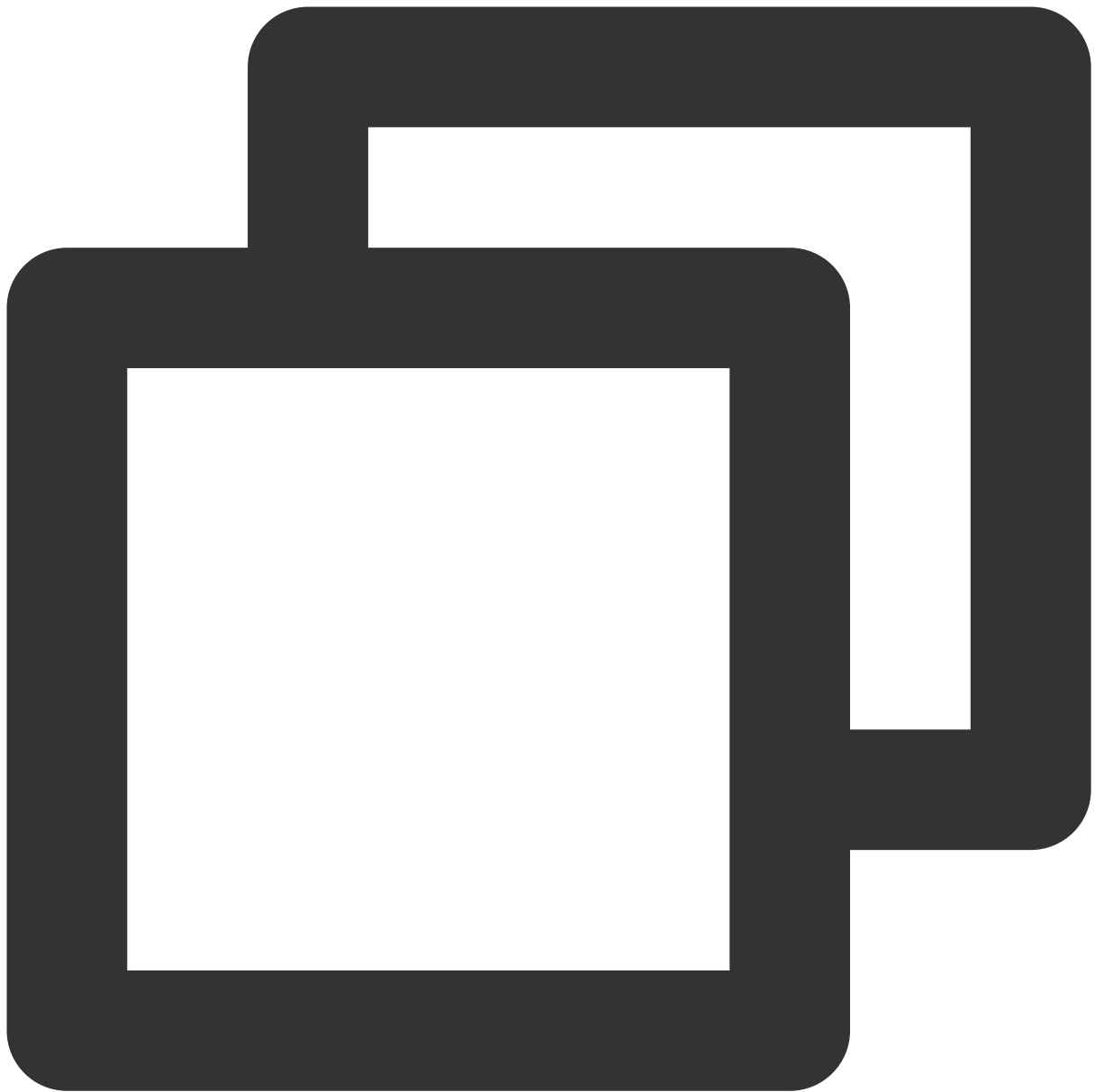
```
const livePusher = new TXLivePusher();
```

2. **Specify the local video player container:**

Specify the div for the local video player container, which is where audio and video captured by the browser will be rendered.

```
livePusher.setRenderView('id_local_video');
```

Note

The video element generated via `setRenderView` is unmuted by default. To mute video, obtain the video element using the code below.
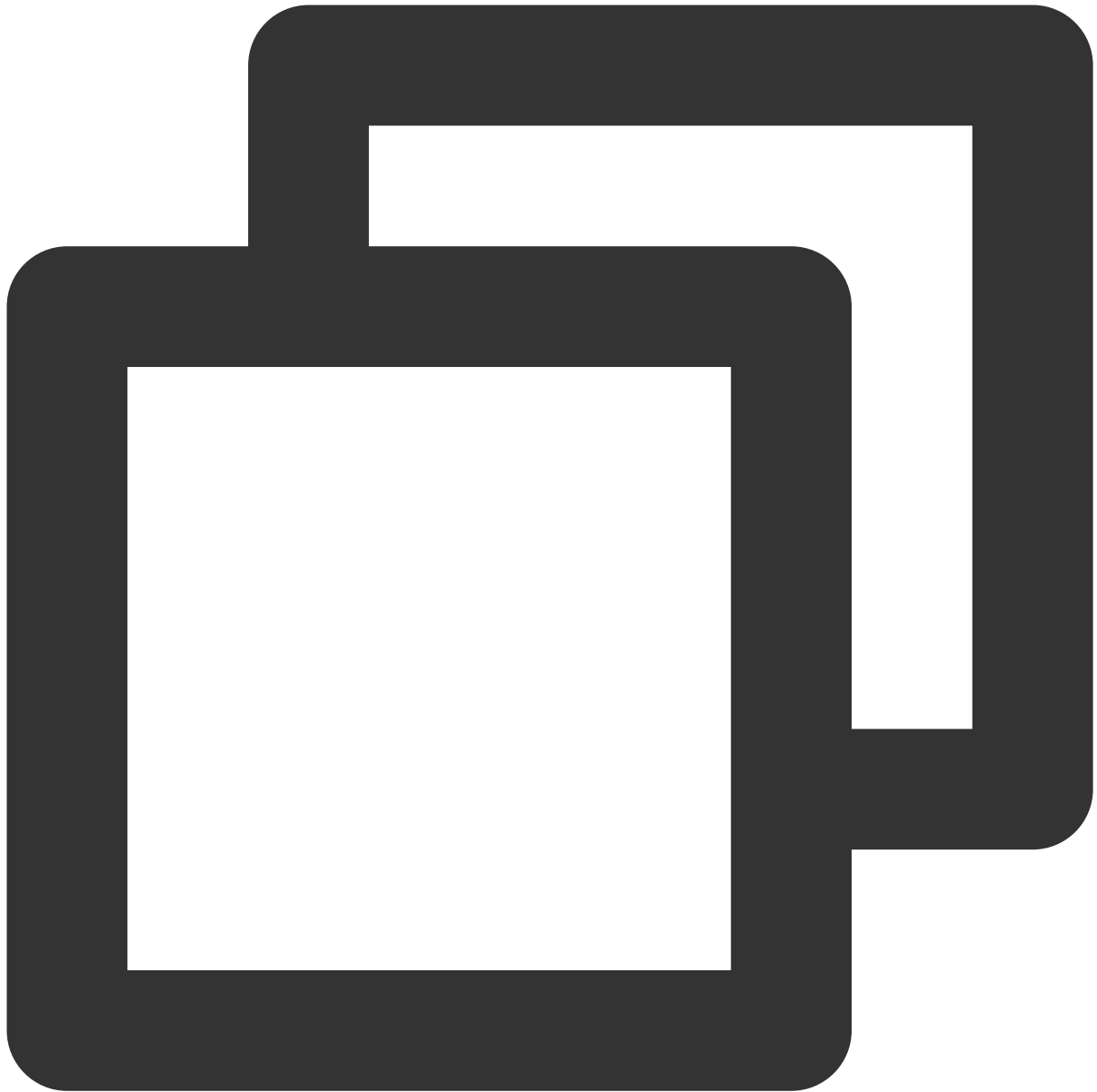
```
livePusher.videoView.muted = true;=
```

3. **Set audio/video quality:**

Audio/video quality should be set before capturing. You can specify quality parameters if the default settings do not meet your requirements.
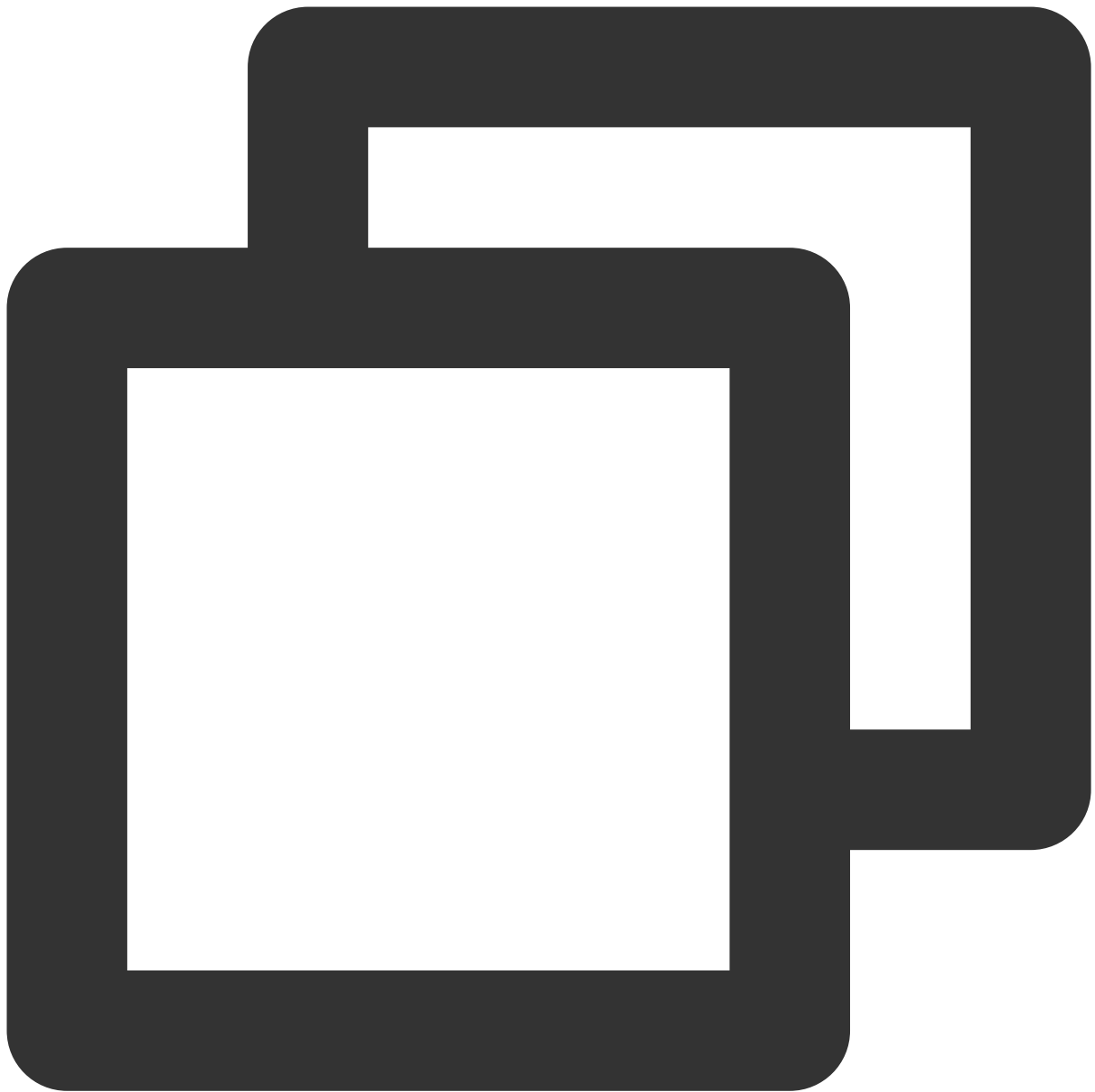
```
// Set video quality
livePusher.setVideoQuality('720p');
// Set audio quality
livePusher.setAudioQuality('standard');
// Set the frame rate
```

4. **Capture streams:**

You can capture streams from the camera, mic, screen and local media files. If capturing is successful, the player container will start playing the audio/video captured.
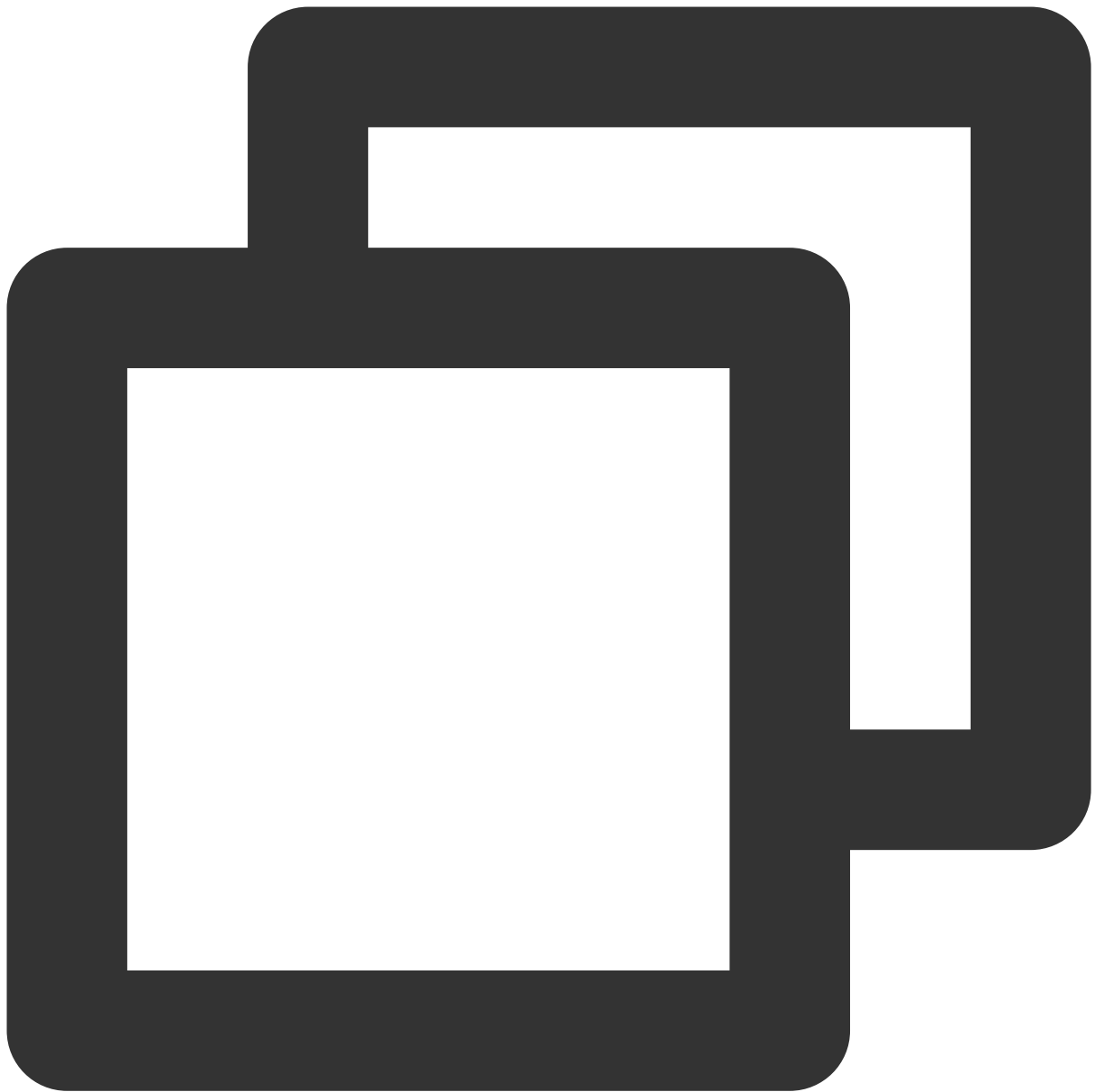
```
// Turn the camera on
livePusher.startCamera();
// Turn the mic on
livePusher.startMicrophone();
```

5. **Push streams:**

Pass in the LEB push URL to start pushing streams. For the format of push URLs, see Splicing Live Streaming URLs.

You need to replace the prefix `rtmp://` with `webrtc://` .
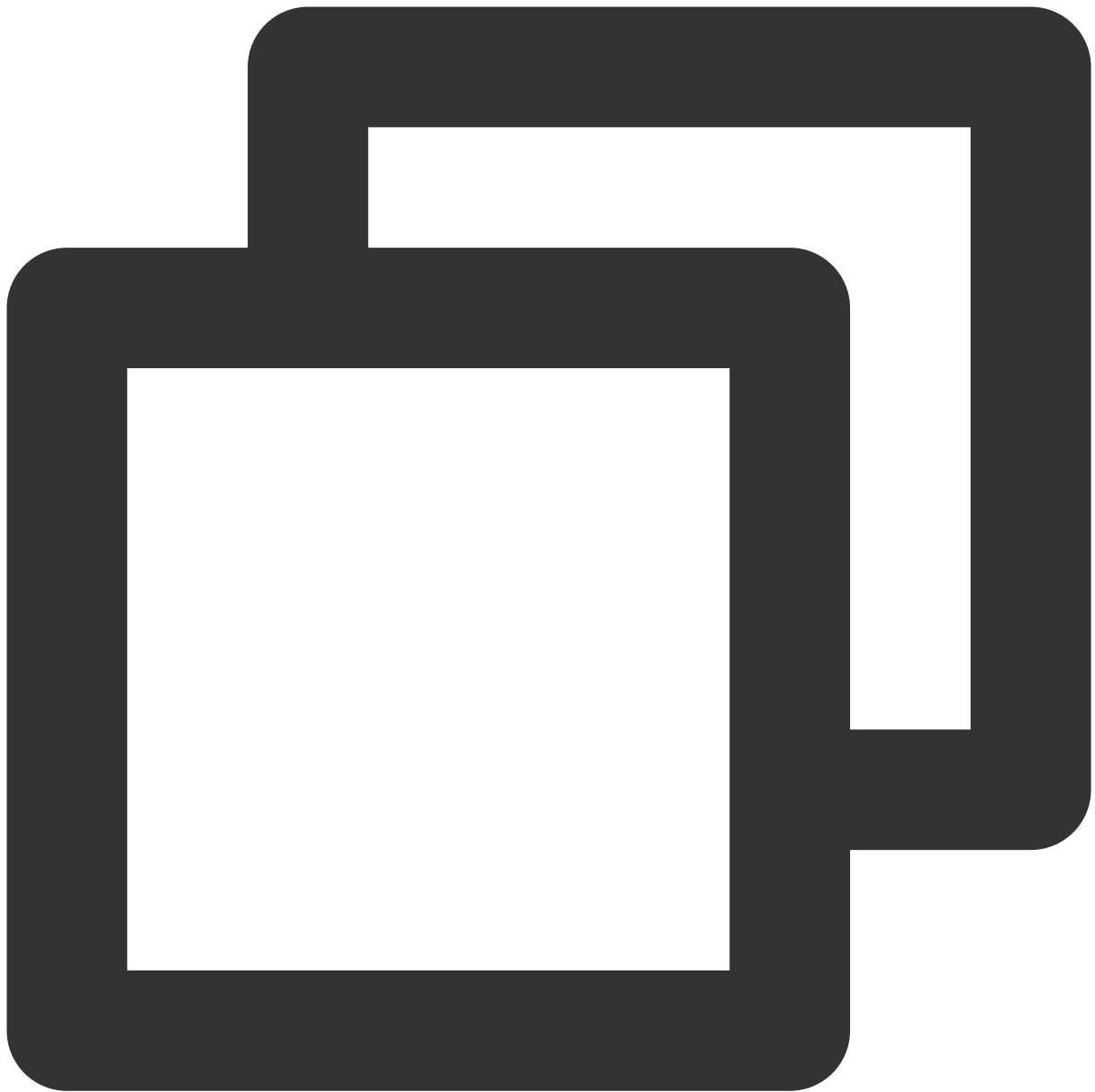
```
livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx');
```

Note

Before push, make sure that audio/video streams are captured successfully, or you will fail to call the push API. You

can use the code below to push streams automatically after audio/video is captured, that is, after the callback for

capturing the first audio or video frame is received. If both audio and video are captured, push starts only after both the

callback for capturing the first audio frame and that for the first video frame are received.

```
// Automatically push the stream after collecting the camera footage
livePusher.startCamera()
.then(function () {
 livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx')
})
.catch(function (error) {
 console.log('Failed to open camera: '+ error.toString());
});


// Automatically push the stream after collecting the camera and microphone
Promise.all([livePusher.startCamera(), livePusher.startMicrophone()])
```

```
.then(function() {
  livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx')
});
```

6. **Stop push:**
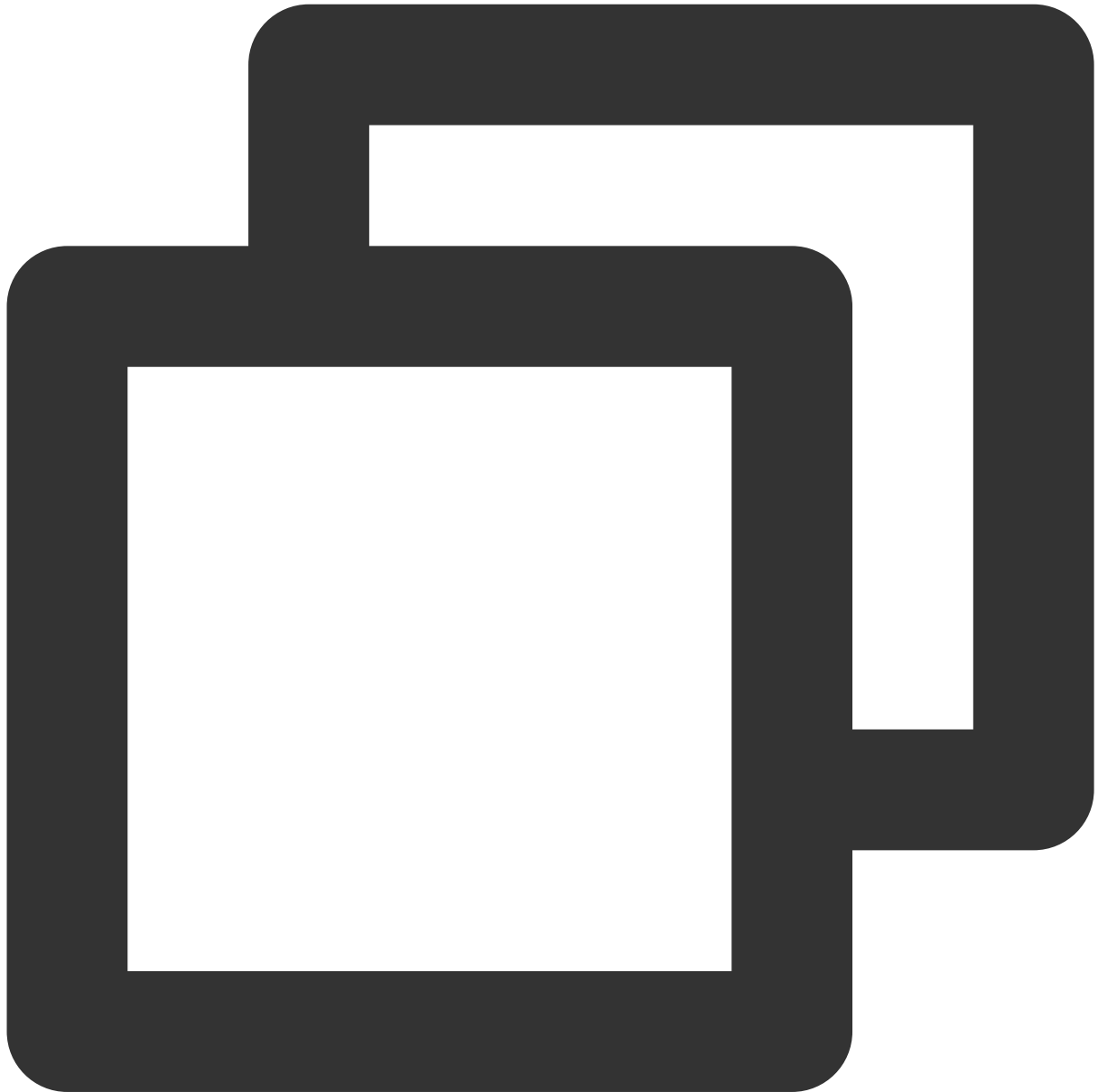

```
livePusher.stopPush();
```

7. **Stop capturing audio and video:**

```
// Turn the camera off
livePusher.stopCamera();
// Turn the mic off
livePusher.stopMicrophone();
```

Advanced Features

Compatibility

The SDK provides a static method to check whether a browser supports WebRTC.
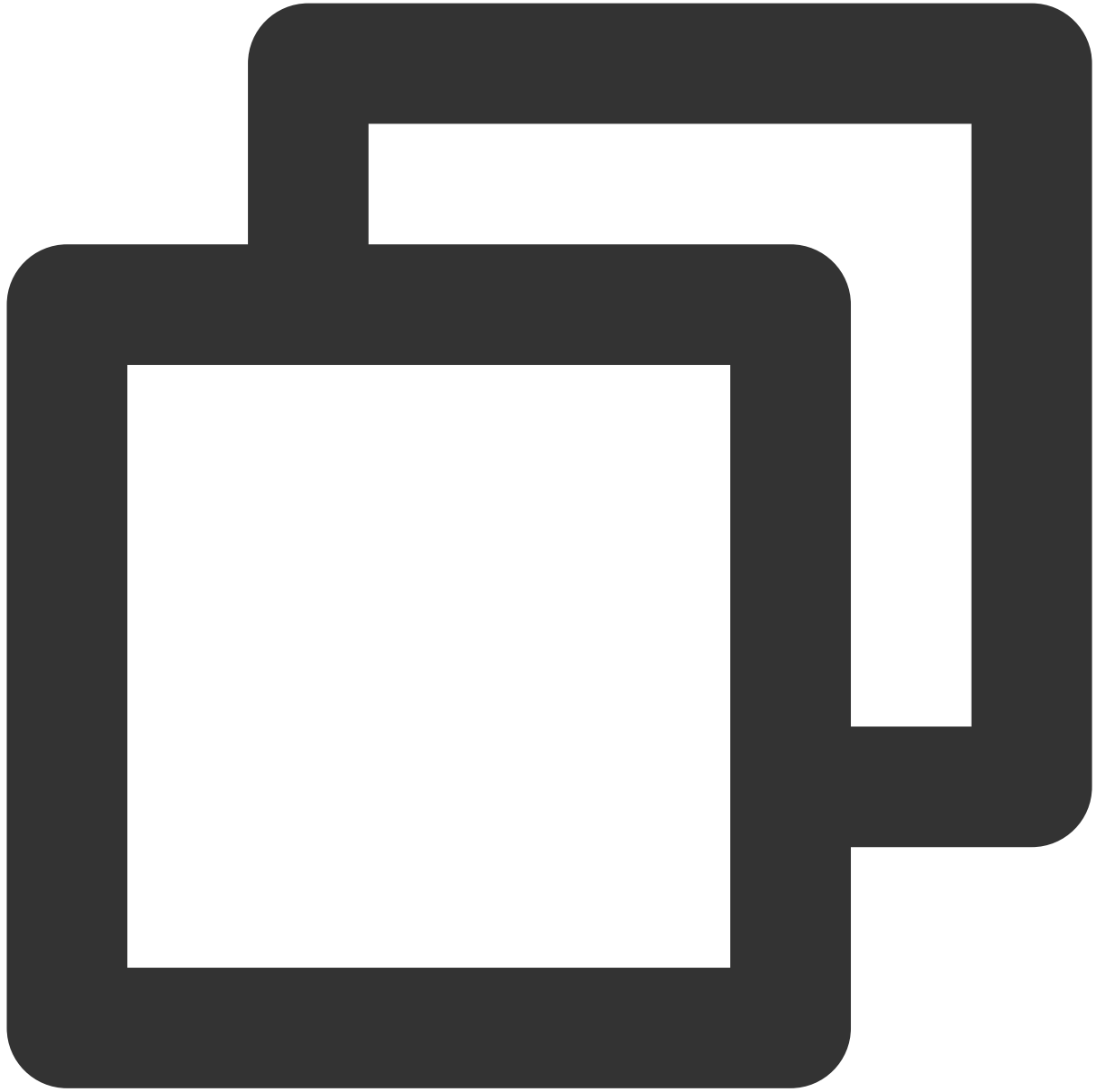

```
TXLivePusher.checkSupport().then(function(data) {
  // Whether WebRTC is supported
  if (data.isWebRTCSupported) {
    console.log('WebRTC Support');
  } else {
    console.log('WebRTC Not Support');
  }
  // Whether H.264 is supported
  if (data.isH264EncodeSupported) {
```

```
      console.log('H264 Encode Support');
   } else {
      console.log('H264 Encode Not Support');
   }
});
```

Event callbacks

The SDK supports callback event notifications. You can set an observer to receive callbacks of the SDK's status and WebRTC-related statistics.
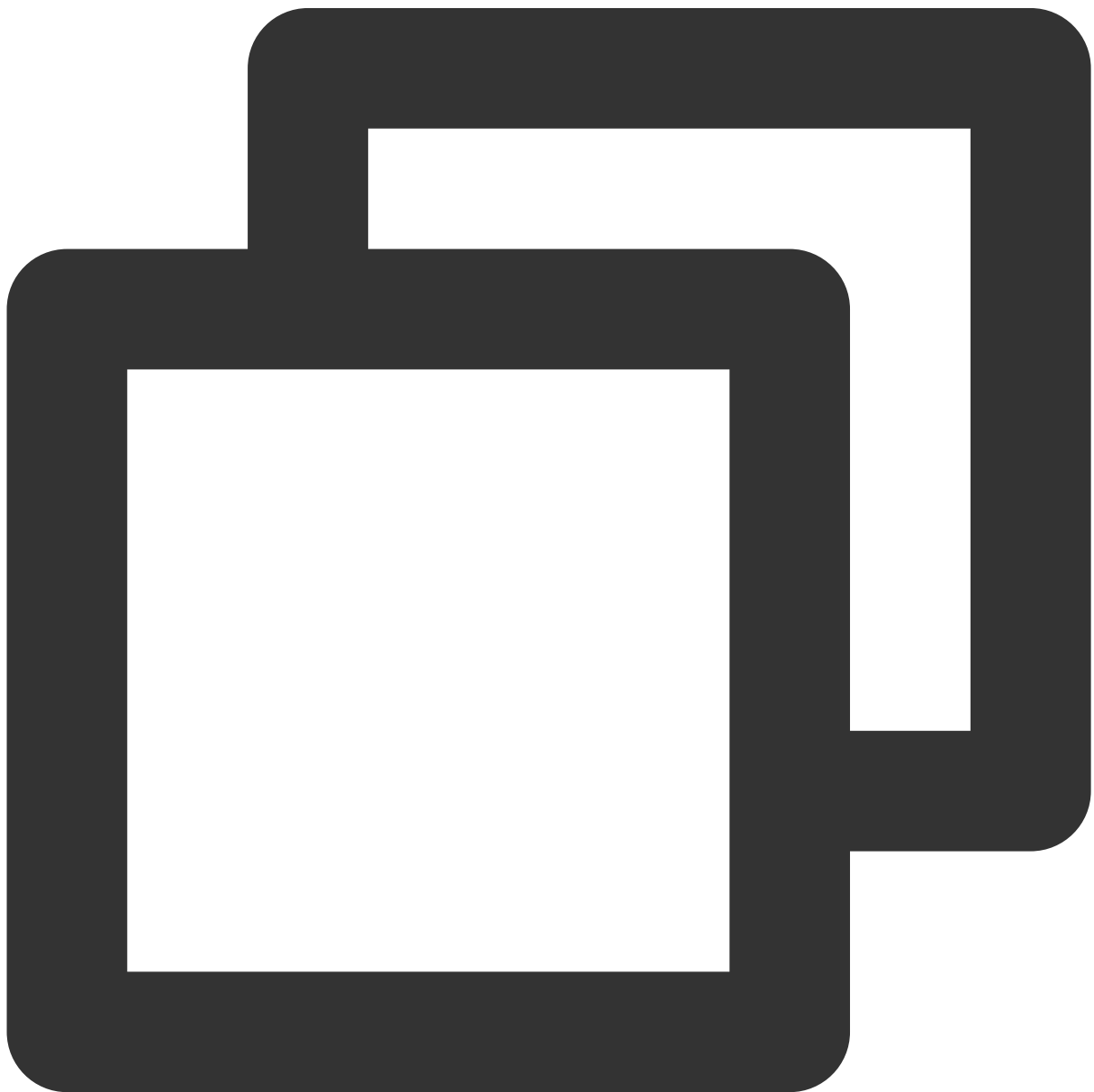
```
livePusher.setObserver({
  // Warnings for push
  onWarning: function(code, msg) {
    console.log(code, msg);
  },
  // Push status
  onPushStatusUpdate: function(status, msg) {
    console.log(status, msg);
  },
```

Device management

You can use a device management instance to get the device list, switch devices, and perform other device-related operations.

```
livePusher.setObserver({
```

```
const deviceManager = livePusher.getDeviceManager();
let cameraDeviceId = null;

// Get device list
deviceManager.getDevicesList().then(function(data) {
  data.forEach(function(device) {
    console.log(device.type, device.deviceId, device.deviceName);

    if (device.type === 'video') {
      cameraDeviceId = device.deviceId;
    }
```

```
  });

  // Switch camera device
  if (cameraDeviceId) {
    deviceManager.switchCamera(cameraDeviceId);
  }
});
```

Live Push

Last updated：2024-03-26 10:26:33

The nature of CSS is a streaming process, similar to the live broadcast of TV channels sent to audience through cable networks. 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

1. Log in to the CSS console, select **Toolkit** > **Address Generator** to generate a push address and configure as follows:
Select **Push Address** 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 an encryption type based on the security needs and performance considerations. The available encryption types are **MD5** or **SHA256**, with **MD5** being the default option.
Select the expiration time of the address, such as `2024-03-21 11:49:04` .
2. 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.
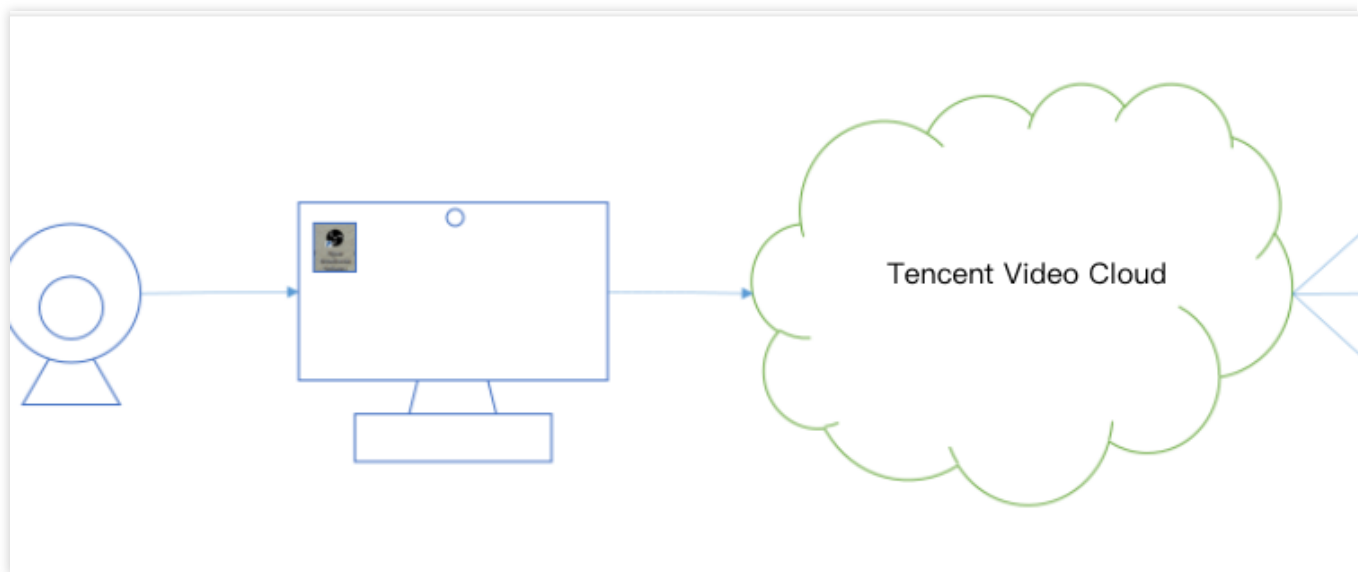
This document uses push with OBS as an example to describe the steps. Assume that the prepared push address is:
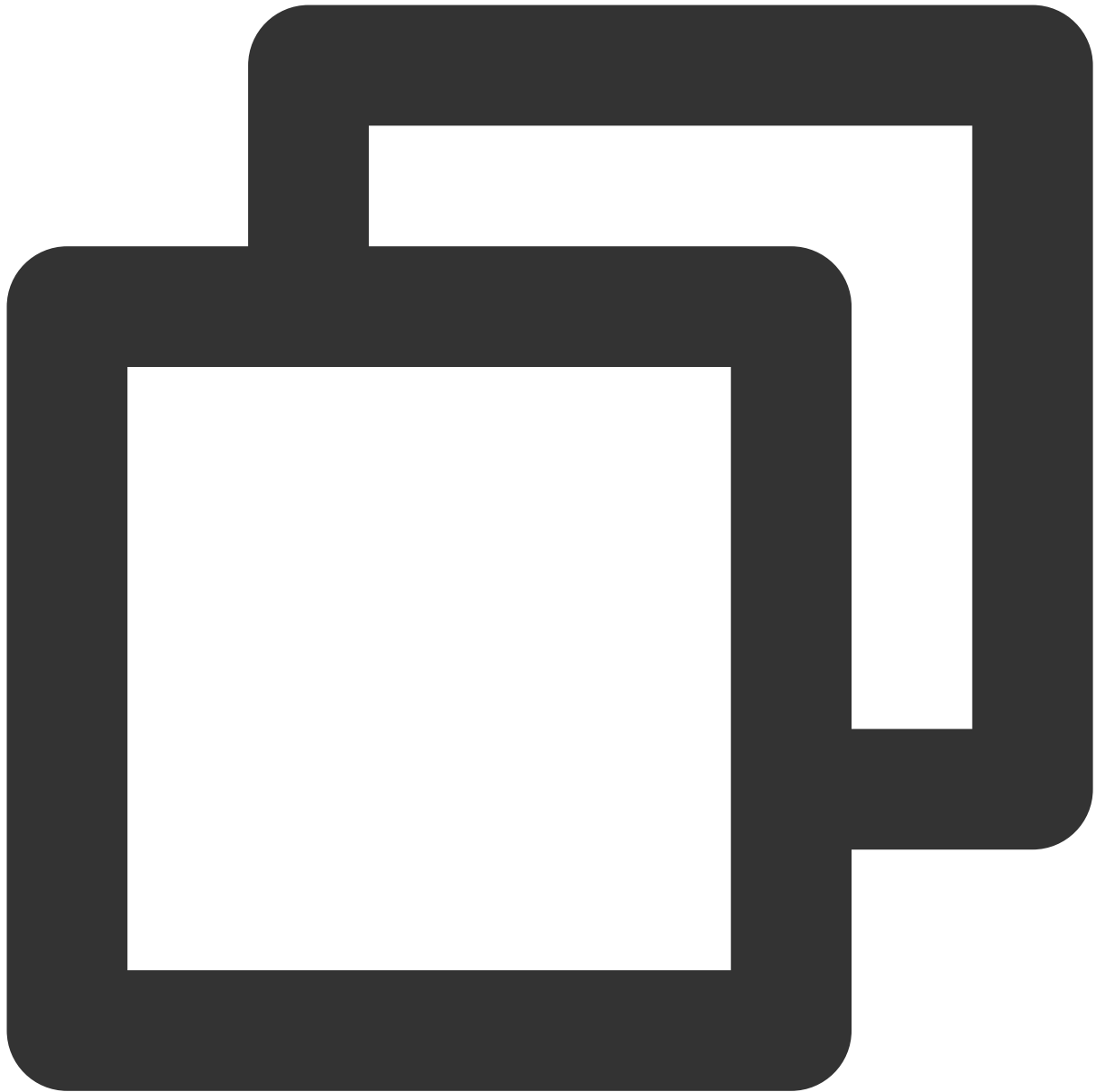
```
rtmp://xxxx.livepush.myqcloud.com/live/xxxx_test?bizid=xxxx&txSecret=xxx&txTime=585
```

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:

3.1 Select "Custom" as the service type.

3.2 Enter the first half of the push address as the server, such as

```
rtmp://xxxx.livepush.myqcloud.com/live/
```
.

3.3 Enter the second half of the push address as the stream key, such as `xxxx?`
`bizid=xxxx&txSecret=xxx&txTime=58540F7F` .

3.4 Click **OK** in the bottom-right corner.
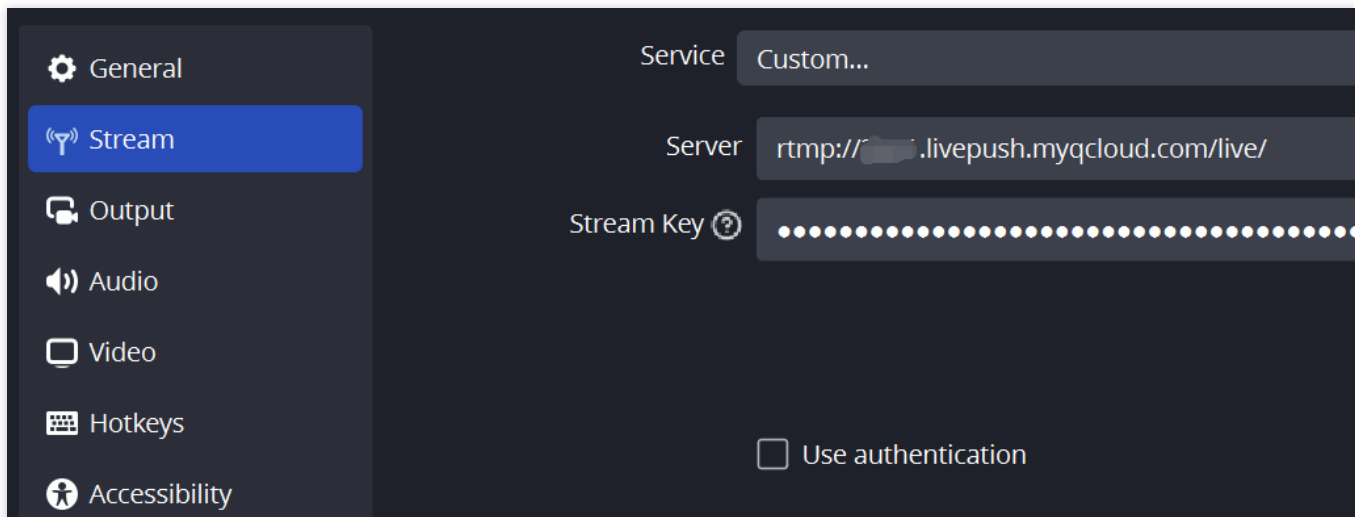
4. Click **Controls** > **Start Streaming** to test streaming. For more information on how to use OBS, please see Push
via OBS.

Scenario 2. Web push

1. Log in to the CSS console.

2. Select **Toolkit** > **Web Push**.

3. Perform the following settings on the web push page:

You can choose Single stream or Multiple streams. For detailed operational procedures, see Web Push.

After deciding on the collection method, configuration, and push settings.

Click **Generate** to proceed to the address generator configuration page.

Select a push domain name.

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 an expiration time, such as `2024-03-21 11:55:59` .

4. 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.

Address Generator

Push Domain	▨▨▨▨▨▨▨▨.com
AppName	live
StreamName	liveteststream
Expiration Time	2024-03-21 11:55:59 📅

[Confirm] [Cancel]

Scenario 3. 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. For more details, see Mobile Live Video Broadcasting (MLVB) SDK.

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：2023-10-07 16:23:34

Preparations

1. Activate the CSS service, and complete the Identity Verification Guide.

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**.

Note

If you do not have a playback domain, you can go to Domain Registration to purchase a domain. You can also purchase a domain through other domain service providers.

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.

You need to choose an encryption type based on your security requirements and performance considerations. The encryption type can be either MD5 or SHA256, with MD5 being the default option.

Select the expiration time of the URL, such as `2023-09-13 15:24:50` .

Click **Generate Address**.

Address Generator

URL Type *	○ Push Address ● Playback Address ○ Push and playback URLs [NEW] ⓘ
Select domain name *	[████ ██ ▼]
AppName *	[live] ⓘ
	Use "live" by default. Only letters, digits, and symbols are supported.
StreamName *	[█] ⓘ
	Only supports letters, digits, and symbols
Type	● MD5 ○ SHA256
Expiration Time	[2023-09-13 15:24:50] 📅
	Push address expiration time is the setting time
Transcoding Template	[██ ▼] Cancel Transcoding
	If you select a transcoding template, the generated playback address will be the live streaming address after transcoding. If you want to play the original live s

[**Generate Address**]   Splice manually   History

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, FFmepg, and TCPlayerDemo for playback.
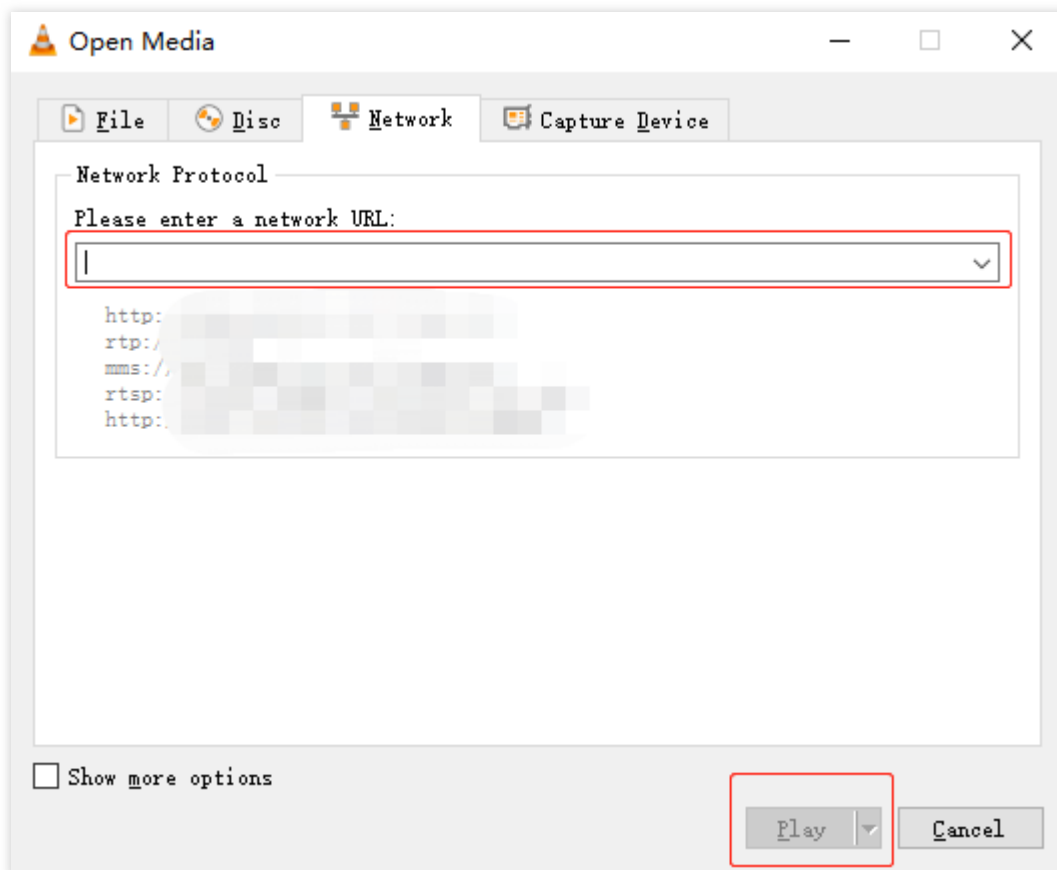
Scenario 2. Playback on mobile client

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.

Note

If you need to push/play streams in an App, you can integrate Mobile Live Video Broadcasting (MLVB) SDK with CSS. MLVB SDK supports RTMP, HTTP-FLV, HLS, and WebRTC playback protocols.

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：2024-03-05 09:24:29

Notes

This article mainly explains the rules for assembling live streaming URLs. If you want to quickly generate push and playback addresses, you can go to the console to generate them. For more information, please refer to the documentation on the Address Generator.

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:
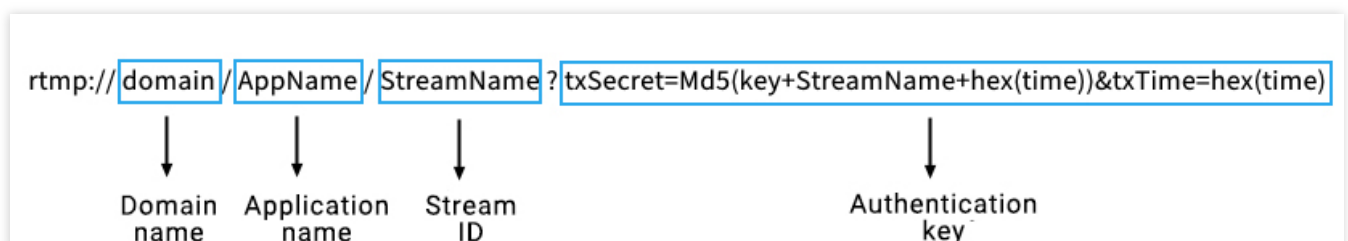
The examples of the addresses are as follows:
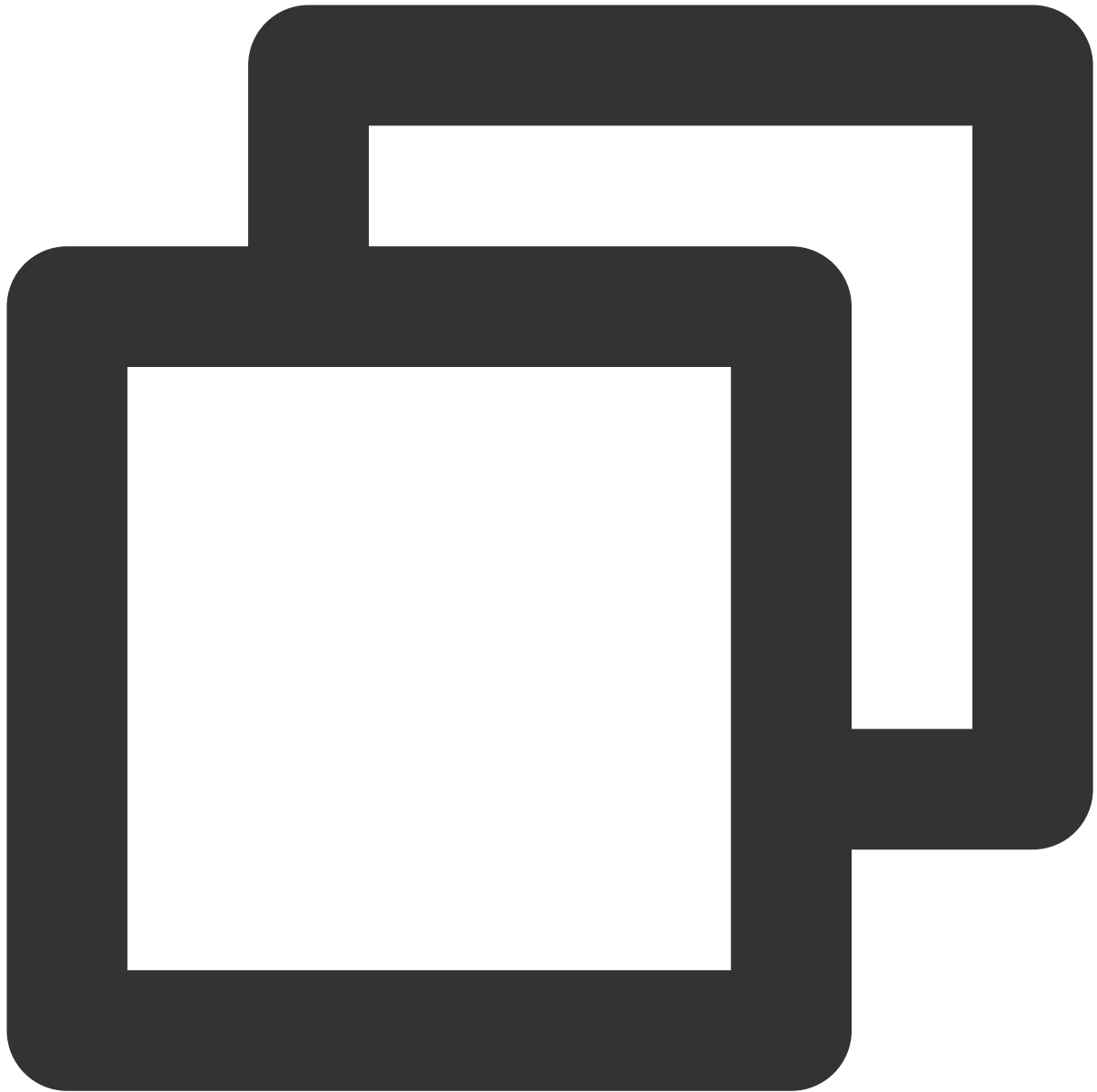
```
webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=he
http://domain/AppName/StreamName.flv?txSecret=Md5(key+StreamName+hex(time))&txTime=
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(
http://domain/AppName/StreamName.m3u8?txSecret=Md5(key+StreamName+hex(time))&txTime
```

domain

Push Domain: You can use the default push domain provided by Tencent Cloud Live Streaming, or you can use your own push domain that has been filed and successfully configured with CNAME.

AppName

The name of the live streaming application, which is "live" by default and can be customized.

StreamName（Stream ID）

Custom Stream Name: A unique identifier for each live stream, recommended to use random numbers or a combination of numbers and letters.

Authentication Key (optional)

The URL includes two parts: txSecret and txTime, where

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

After enabling push authentication, you need to use a URL containing the authentication key for pushing. If push authentication is not enabled, there is no need for "?" and its subsequent content in the push address.

txTime（URL Expiration）

Indicates when the URL will expire, and the format supports hexadecimal UNIX timestamps (Time Unit: Seconds).

Note:

For example, `5867D600` represents the expiration at 0:00:00 on January 1, 2017. Our customers usually set txTime to expire 24 hours after the current time. The expiration time should not be too short or too long. When the broadcaster encounters a network interruption during the live streaming process, they will resume pushing the stream. If the expiration time is too short, the broadcaster will not be able to resume pushing the stream due to the expiration of the push URL.

txSecret (Anti-leakage Signature)

Used to prevent attackers from forging your backend-generated push URL. The calculation method can be found in the Best Practices - Anti-leakage Calculation.

txTime（URL Expiration）

Indicates when the URL will expire, and the format supports hexadecimal UNIX timestamps.

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.

The examples of the addresses are as follows:
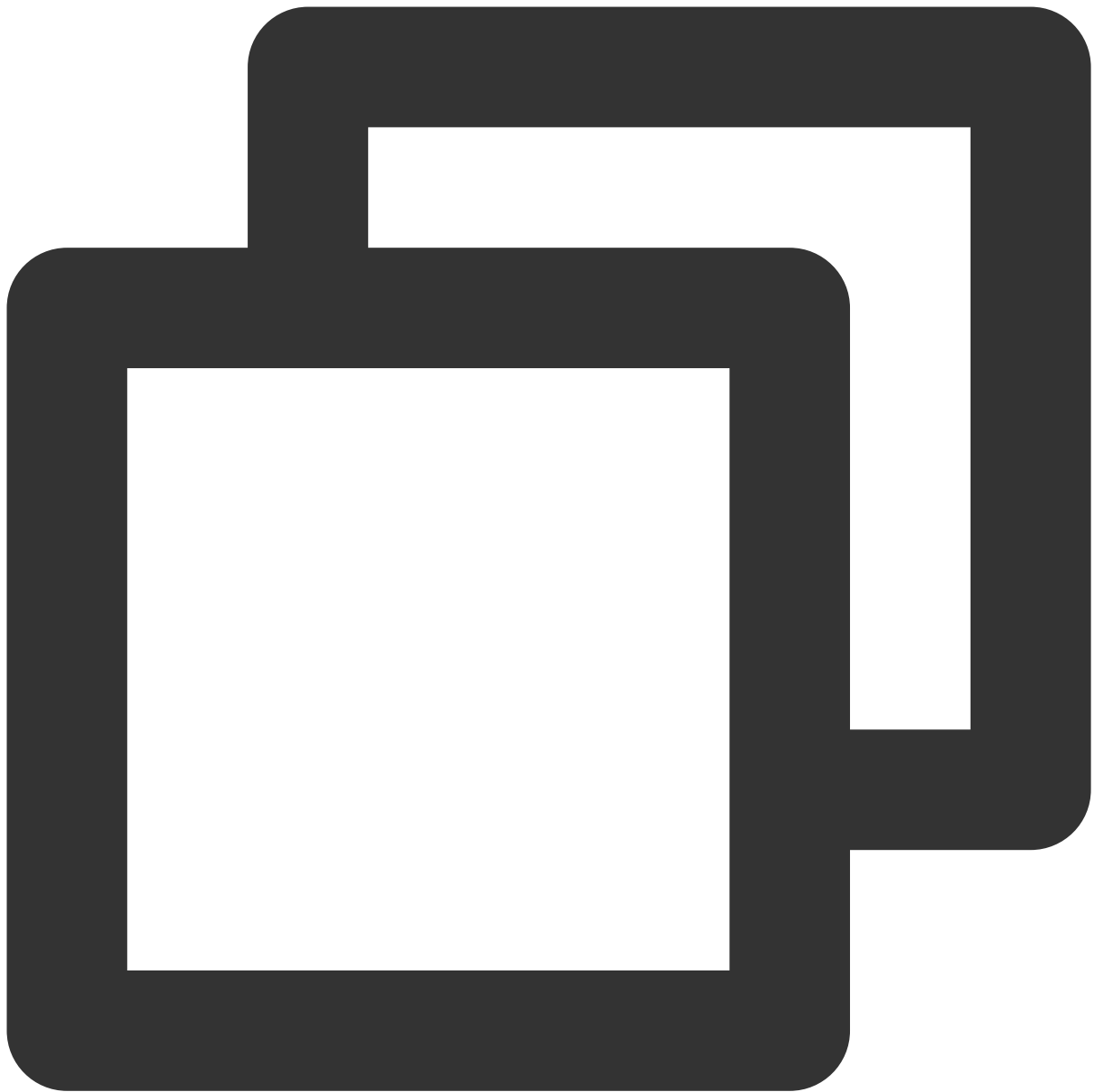
```
webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=he
http://domain/AppName/StreamName.flv?txSecret=Md5(key+StreamName+hex(time))&txTime=
rtmp://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(
http://domain/AppName/StreamName.m3u8?txSecret=Md5(key+StreamName+hex(time))&txTime
```

Playback prefix

Playback Protocol	Playback Prefix	Notes
WebRTC	webrtc://	We recommend WebRTC most as it has the best instant streaming performance and supports ultra-high concurrency.

HTTP-FLV	http:// or https://	We recommend HTTP-FLV as it has good instant streaming performance and supports high concurrency.
RTMP	rtmp://	We do not recommend RTMP as it has poor instant streaming performance and does not support high concurrency.
HLS (M3U8)	http:// or https://	We recommend HLS for mobile clients and for the Safari browser on macOS.

domain

Push Domain: You can use the default push domain provided by Tencent Cloud Live Streaming, or you can use your own push domain that has been filed and successfully configured with CNAME.

AppName

The name of the live streaming application, utilized for differentiating the storage path of live streaming media files, which is "live" by default and can be customized.

StreamName（Stream ID）

Custom Stream Name: A unique identifier for each live stream. It is recommended to use random numbers or a combination of numbers and letters. It is not recommended to include "_". If the string after "_" is the same as the transcoding template name, the string will be recognized as the transcoding template name, and the string before "_" will be recognized as StreamName, which may cause playback anomalies. For example: `test_a1_hd1` , `test_a1` will be recognized as StreamName, and "hd1" will be recognized as the transcoding template name.

Transcoding Template Name

By appending a "_" suffix to the StreamName, the program will pull the transcoding stream according to this transcoding template.

Authentication Key (optional)

The URL includes two parts: txSecret and txTime, where

`txSecret=Md5(key+StreamName_Transcoding Template Name+hex(time))&txTime=hex(time)` .

After enabling push authentication, you need to use a URL containing the authentication key for pushing. If push authentication is not enabled, there is no need for "?" and its subsequent content in the push address.

txTime（URL Expiration）

Indicates when the URL will expire, and the format supports hexadecimal UNIX timestamps(Time Unit: Seconds).

Note:

For example, `5867D600` represents the expiration at 0:00:00 on January 1, 2017. Our customers usually set txTime to expire 24 hours after the current time. The expiration time should not be too short or too long. When the broadcaster encounters a network interruption during the live streaming process, they will resume pushing the stream. If the expiration time is too short, the broadcaster will not be able to resume pushing the stream due to the expiration of the push URL.

txSecret (Anti-leakage Signature)

Used to prevent attackers from forging your backend-generated push URL. The calculation method can be found in the Best Practices - Anti-leakage 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 、 Java) that demonstrates how to generate a hotlink protection address. For detailed directions, please see Push Configuration.

SRT Push

Last updated：2023-10-08 15:05:39

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.

TS over SRT Architecture
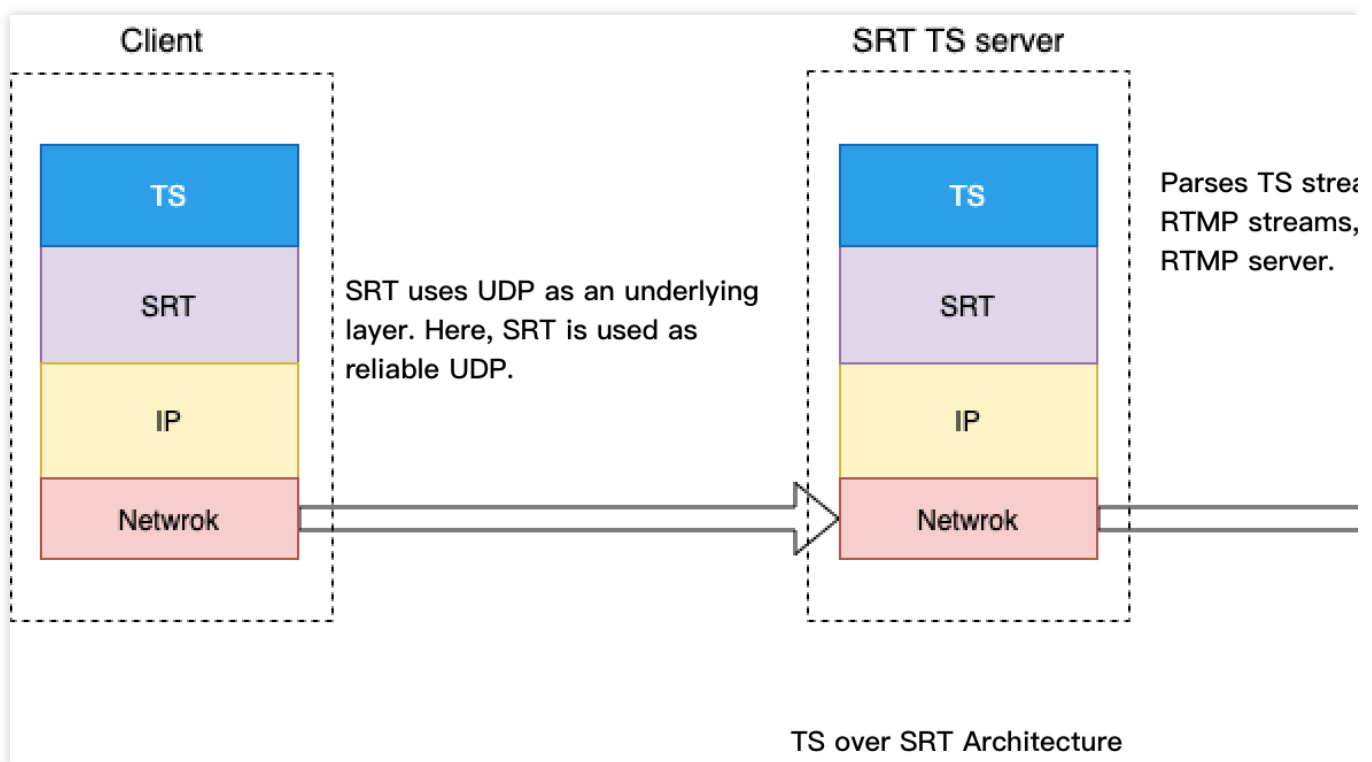
Caution

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:

Frame Loss Rate Under SRT vs. RT
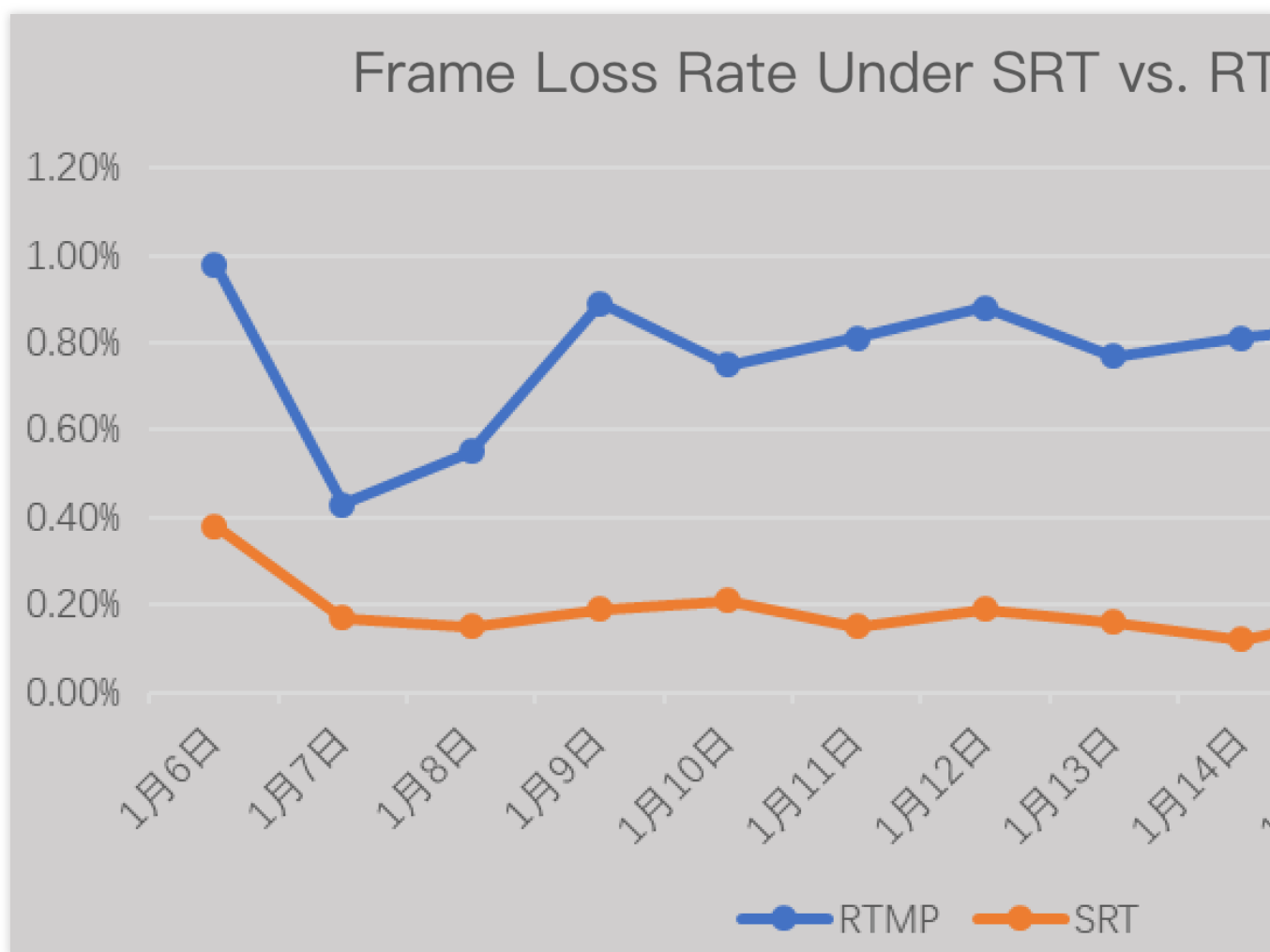
Packet Loss Rate Comparison

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%	
Push smoothness	TCP	Smooth	Severe stutter		
	SRT	Smooth	Smooth	Smooth	Smoo
Total Men Avg/MB	TCP	380	374		
	SRT	377	387	375	
App CPU Max/%	TCP	7	8		
	SRT	9	9	10	
App CPU Avg/%	TCP	7	7		
	SRT	8	8	9	
Phone CPU Max/%	TCP	33	33		
	SRT	33	33	33	
Phone CPU Avg/%	TCP	29	30		
	SRT	30	31	32	

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

Metric	Protocol	0%	10%	20%	
Push smoothness	TCP	Smooth	Severe stutter		
	SRT	Smooth	Smooth	Smooth	Smoo
Total Men Avg/MB	TCP	298	296		
	SRT	275	300	302	
App CPU Max/%	TCP	13	13		
	SRT	10	10	10	
App CPU Avg/%	TCP	7	6		
	SRT	7	7	7	
Phone CPU Max/%	TCP	47	42		
	SRT	32	34	45	
Phone CPU Avg/%	TCP	31	29		
	SRT	28	29	30	

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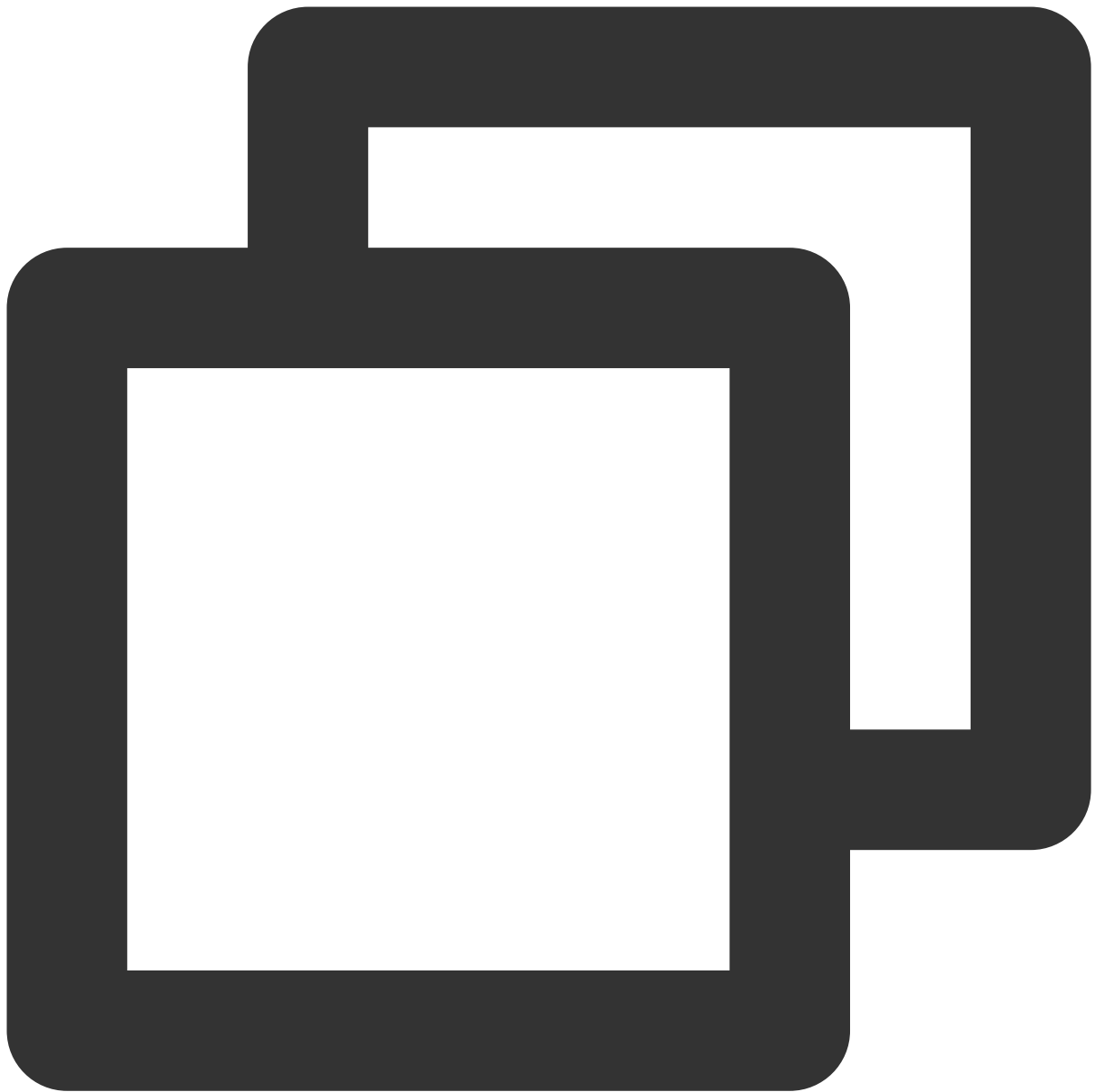
```
srt://${rtmp-push-domain}:9000?streamid=#!::h=${rtmp-push-domain},r=${app}/${stream
```

Caution

`$ {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:

Caution

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.

Delayed Playback

Last updated：2023-10-08 15:14:13

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 **Tools** > Address Generator, select **Push Domain** for **Domain Type**, and click **Generate Address**.

Address Generator

URL Type *	⦿ Push Address	◯ Playback Address	◯ Push and playback URLs NEW ⓘ
Select domain name *	▾		

AppName *

live

ⓘ

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

StreamName *

test

ⓘ

Only supports letters, digits, and symbols

Type

● MD5   ○ SHA256

Expiration Time

2023-09-28 14:44:51

📅

The actual expiration time of the playback URL is the timestamp selected plus the valid

Generate Address   Splice manually   History

Live streaming URLs

✓ The URLs are automatically saved to the browser cache and will be de

URL Type

Push Address

Validity Period

2023-09-28 14:44:51 (UTC+08:00)   **reference documentation** ↗

RTMP URL

rtmp://⬛⬛⬛⬛⬛⬛/live/test?
txSecret=544538fab30e648d6c0156b8df25f83f&txTime=651520E3

OBS server

rtmp://⬛⬛⬛⬛⬛⬛/live/

OBS stream key

test?txSecret=544538fab30e648d6c0156b8df25f83f&txTime=651520E3

WebRTC URL

webrtc://⬛⬛⬛⬛⬛⬛/test?
txSecret=544538fab30e648d6c0156b8df25f83f&txTime=651520E3

Millisecond latency

SRT URL

srt://⬛⬛⬛⬛⬛⬛:9000?
streamid=#!::h=⬛⬛⬛⬛⬛⬛,r=live/test,txSecret=544538fab30e648d6c0156b8df
5f83f,txTime=651520E3

RTMP over SRT URL

rtmp://⬛⬛⬛⬛⬛⬛:3570/live/test?
txSecret=544538fab30e648d6c0156b8df25f83f&txTime=651520E3

2. Add `txDelayTime` to the end of the push address, and push streams via OBS. For detailed directions, please see Push via OBS.
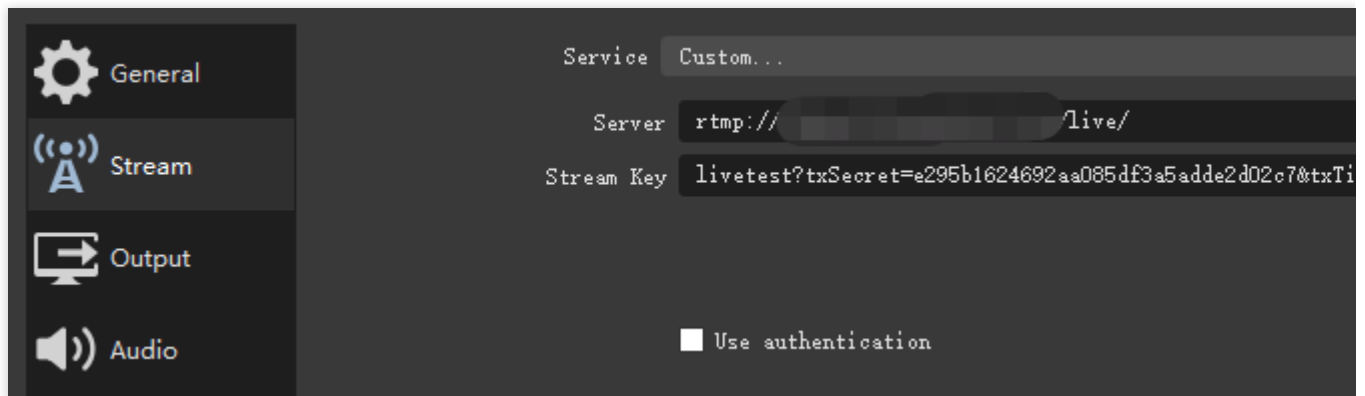
Caution

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 **Tools** > 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.

RTMPS Push Streaming

Last updated：2024-01-22 15:51:37

Tencent Video Cloud continuously improves its streaming media transmission to cater to various requirements. Apart from the common RTMP streaming protocol, it also now supports the RTMPS streaming protocol, which specifically caters to users with encryption needs, particularly those with overseas operations. This article primarily focuses on the implementation of RTMPS streaming.

Comparison of Advantages

Standard RTMP streaming protocol relies entirely on adding relevant parameters to the URL for authentication, with the RTMP server carrying out verification based on these parameters. However, this does not encrypt transmitted audio and video data packets, hence, captured and decoded RTMP packets can be played.
The RTMPS protocol effectively resolves these RTMP security issues. It is the SSL-encrypted version of the RTMP protocol, which enhances the security of data communication, allowing safe stream transmission between the encrypted encoder and CDN.

Comparison chart of various streaming protocols:

Protocol	Protocol Type	Transmission Method	Delay	Protocol Characteristic	Application Scenario	Tencent's Optimization Scheme
RTMPS	Streaming Protocol	TCP	-	Encryption	Encryption scenarios	Supporting multiple domains and certificates
SRT	Streaming Protocol	UDP	500ms-1s	Low latency, packet loss resistance	OTT, cross-regional transmission	-
WebRTC	Streaming Protocol	RTP	200ms-1s	Low latency	Audio and video calls	Optimizing instant loading and stuttering in Live Event Broadcasting
QUIC	Streaming Protocol	UDP	-	Packet loss resistance, 0rtt	Browser access	Optimizing the first frame transmission

Notes

The RTMPS streaming protocol, which employs SSL encryption, demands the configuration of a push domain name certificate; at present, the default push domain name 'push.tlivecloud.com' of Cloud Streaming Services has been equipped with a common certificate. For other default domains, you need to Submit a Ticket **to provide domain names and configure according to the corresponding certificate.** If you desire to use your certificate, port replacement is necessary. Tencent Video Cloud's multi-protocol platform has optimized the RTMPS protocol. There is no need for users to switch ports. Users can directly use their certificates. The platform will automatically adapt to the domains and match the corresponding certificate.

Note

If you wish to test the RTMPS streaming, you can connect the default push domain name by Cloud Streaming Services.

If you need to use the RTMPS streaming through your own domain, you need to Submit a Ticket providing the domain name and corresponding certificate.

RTMPS Streaming

1. The generation of live streaming URLs can be carried out in the following two methods:

Assemble independently through splicing rules. For detailed operations, please refer to Splicing Live Streaming URLs.

1.1 Enter the **Tools >** Address Generator of the Cloud Streaming Services console, select URL type: Push Address, and select domain name as needed. For detailed operations, please refer to Address Generator.

Address Generator

URL Type *   (•) Push Address   ( ) Playback Address   ( ) Push and playback URLs [NEW] (i)

Select domain name *   [⌄]

AppName *   [live   ] (i)

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

StreamName *   [liveteststream   ] (i)

Only supports letters, digits, and symbols

Type   (•) MD5   ( ) SHA256

Expiration Time   [2023-11-22 17:51:05   📅]

The actual expiration time of the playback URL is the timestamp selected plus the validity period of the authentication key.

[**Generate Address**]   Splice manually   History

Live streaming URLs   ⊘ The URLs are automatically saved to the browser cache and will be deleted when you clear the cache.

URL Type	Push Address		
Validity Period	2023-11-22 17:51:05 (UTC+08:00) reference documentation 🗗		
RTMP URL	rtmp://⬛⬛⬛⬛⬛⬛⬛/live/liveteststream? txSecret=7358c5f2⬛⬛⬛⬛⬛6a8411cdcdab&txTime=655⬛F09	Copy QR code	
OBS server	rtmp://⬛⬛⬛⬛⬛⬛⬛/live/	Copy	
OBS stream key	liveteststream?txSecret=7358c5f2⬛⬛⬛⬛⬛6a8411cdcdab&txTime=655⬛F09	Copy	
WebRTC URL [Millisecond latency]	webrtc://⬛⬛⬛⬛⬛⬛/live/liveteststream? txSecret=7358c5f2⬛⬛⬛⬛⬛5a8411cdcdab&txTime=655⬛F09	Copy Web Push Web push doc 🗗 OBS push doc 🗗	
SRT URL	srt://⬛⬛⬛⬛⬛⬛⬛:9000?streamid=#!::h=⬛⬛ ⬛⬛⬛⬛,r=live/liveteststream,txSecret=7358c5f2⬛⬛⬛⬛5a8411cdcda b,txTime=655⬛F09	Copy	
RTMP over SRT URL	rtmp://⬛⬛⬛⬛⬛⬛⬛:3570/live/liveteststream? txSecret=7358c5f232⬛⬛⬛⬛5a8411cdcdab&txTime=655⬛F09	Copy	

2. Modify the generated RTMP live streaming URL into RTMPS, input it into OBS to start RTMPS streaming. For detailed operations, please refer to Push via OBS.

⚙ General	Service	Custom...
((•)) Stream	Server	rtmp://⬛⬛⬛⬛⬛⬛:/live/
🖵 Output	Stream Key	••
◀)) Audio		
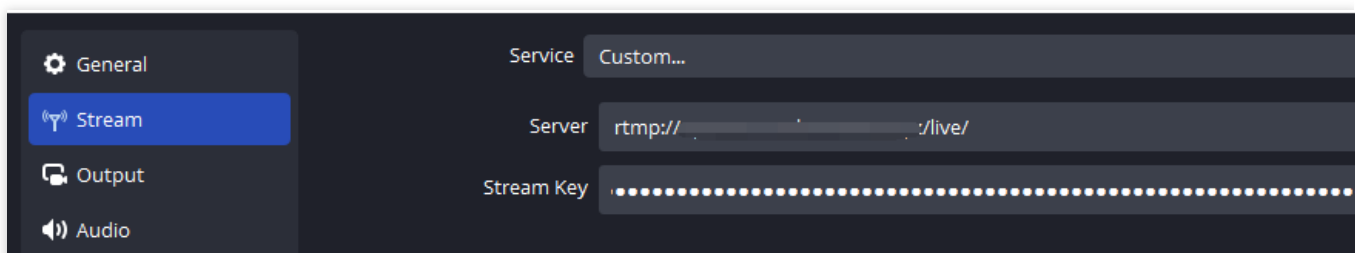
Live Playback

Operate according to the normal live playback process. For more details, please refer to Live Playback.

Features

Live Remuxing and Transcoding

Last updated：2023-10-08 14:47:58

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: H.264 H.265 AV1
Video profile	Supported video profiles: Baseline Main High
Video encoding bitrate	Supported video output bitrate range: 101 Kbps - 8000 Kbps. 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	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	Supported width range: 0 - 3000. Supported height range: 0 - 3000. You can only specify the width and the height will be scaled proportionally. 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: Fixed bitrate (CBR) Dynamic bitrate (VBR)
Video image rotation	The original video can be rotated clockwise by: 90 degrees 180 degrees 270 degrees

Audio transcoding parameters

Parameter Type	Description
Audio codec	Supported codecs: AAC-LC AAC-HE AAC-HE v2
Audio sample rate	Supported sample rates (48000 and 44100 are commonly used): 96000 64000 48000 44100 32000 24000 16000 12000 8000
Audio encoding bitrate	Supported bitrate range: 20-192 Kbps; commonly used bitrates include: 48 Kbps 64 Kbps 128 Kbps
Sound channel	Supported sound channel modes: Mono Dual

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, 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 (50% 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 x 0.2118 = 127.08 USD. If TSC transcoding is used to reduce the bitrate, the incurred bandwidth fees will be around 127.08 x (100% - 30%) = 88.956 USD.

TSC transcoding fees: 0.0443 x 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.
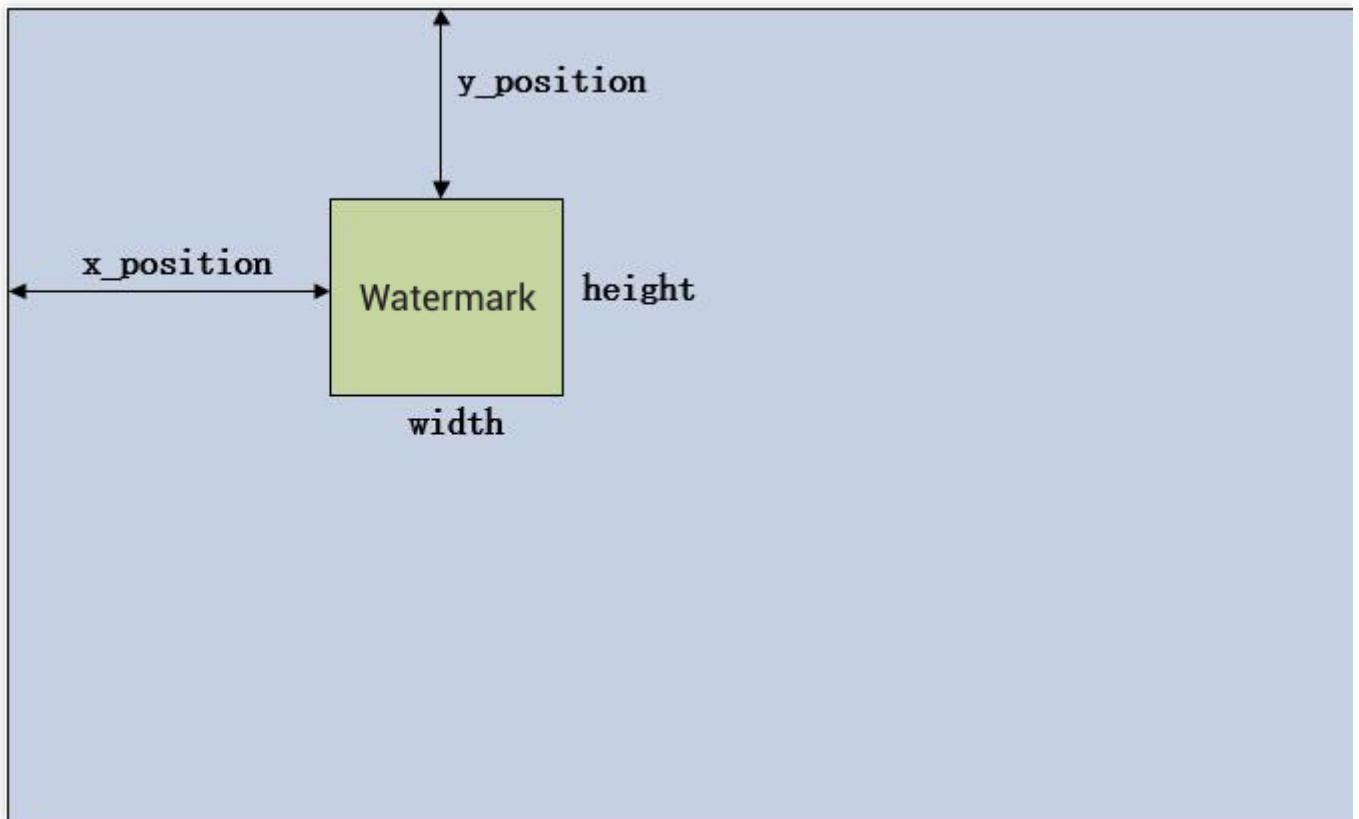
Caution

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

1. Call the AddLiveWatermark API to add a watermark by setting the watermark name and other parameters.

2. 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.

Caution

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
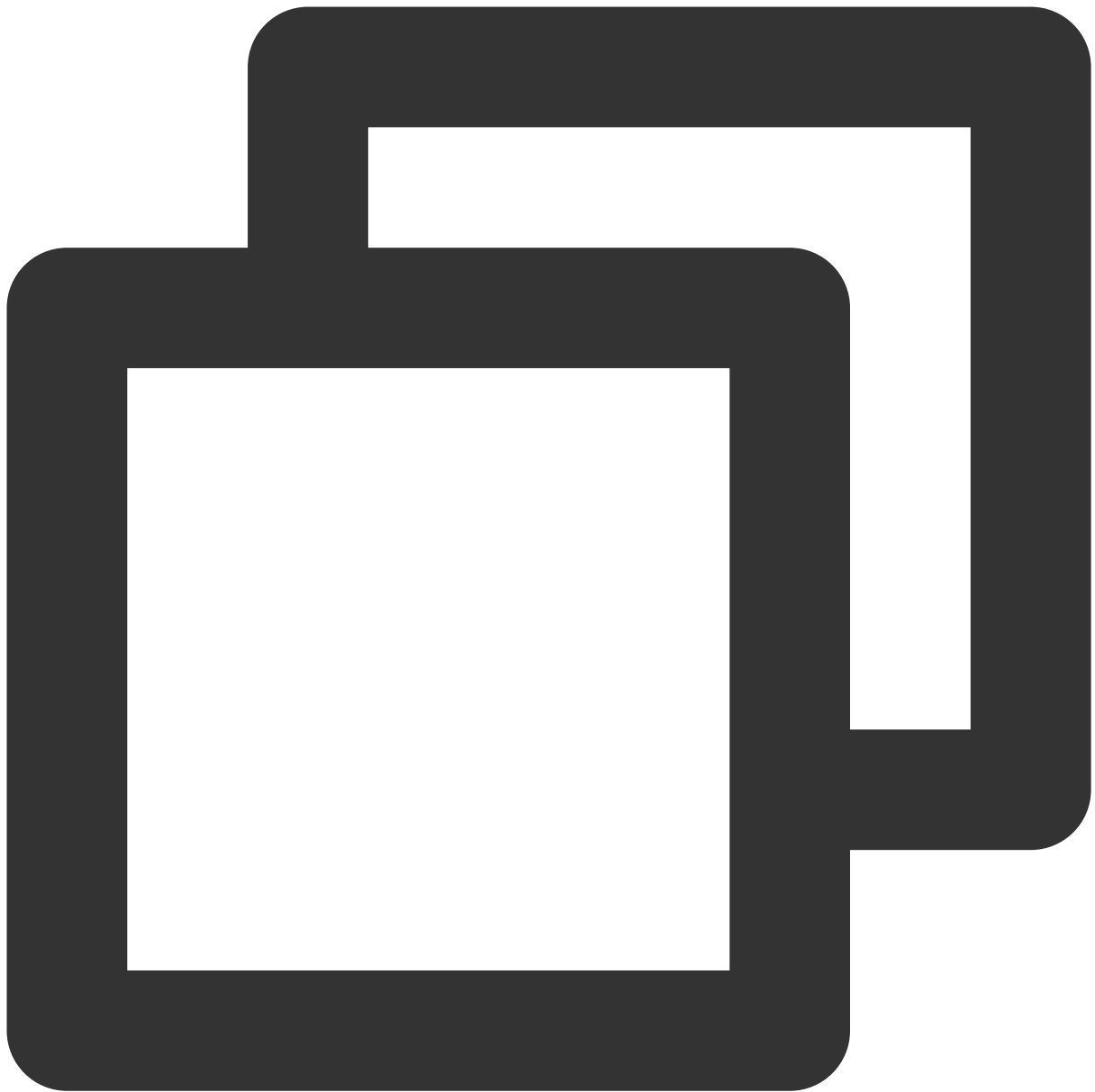
```
**Playback address = Playback domain name + Playback path + Stream ID_transcoding t
```

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`

Caution

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：2023-11-22 15:13:38

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
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 to set `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. Upon successful processing, the settings will take effect within about 5 to 10 minutes.
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.
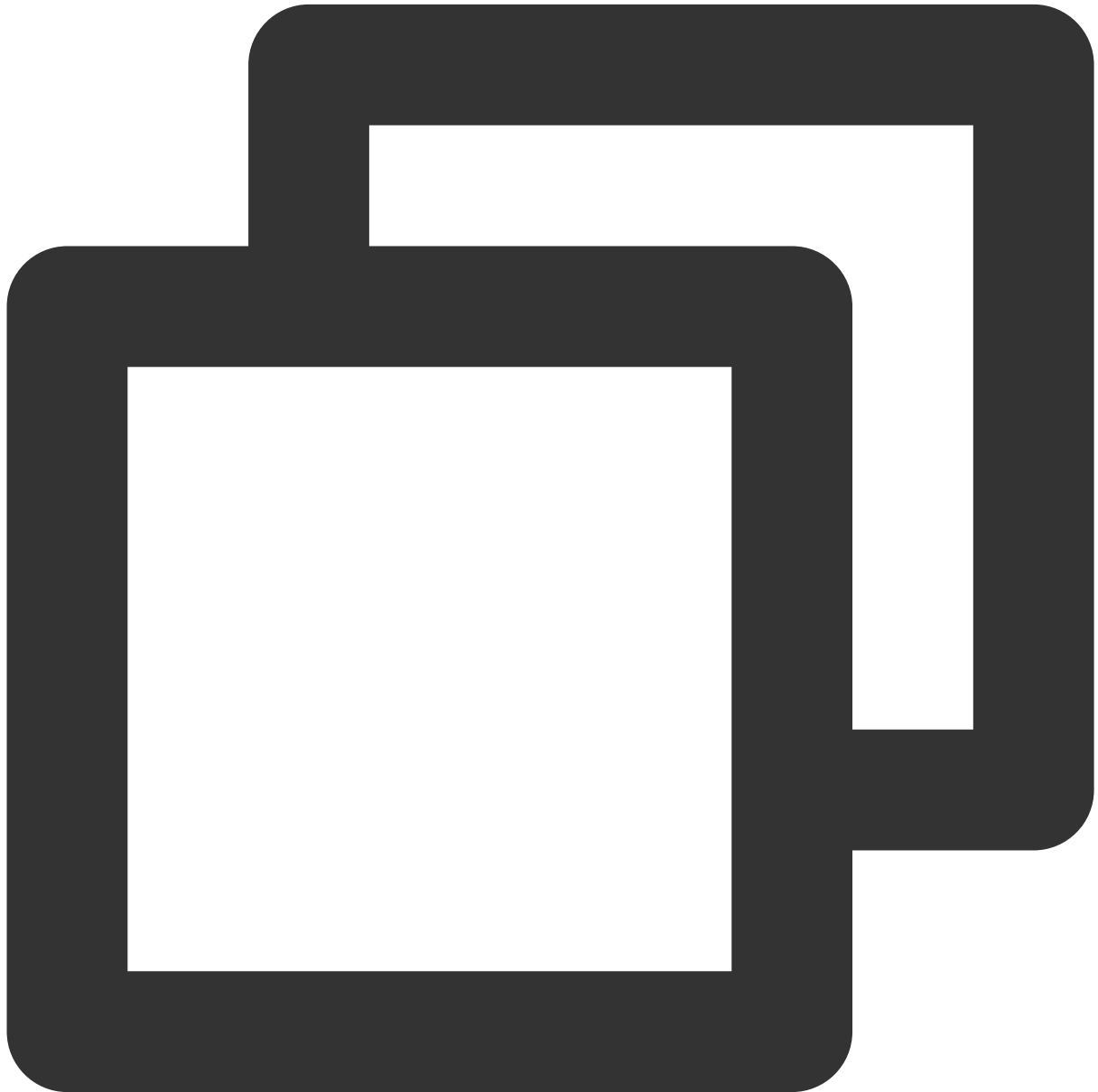
Priority	DomainName	AppName	StreamName
1	✓	✓	✓
2	✓	×	✓
3	✓	✓	×
4	✓	×	×

✓ 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.
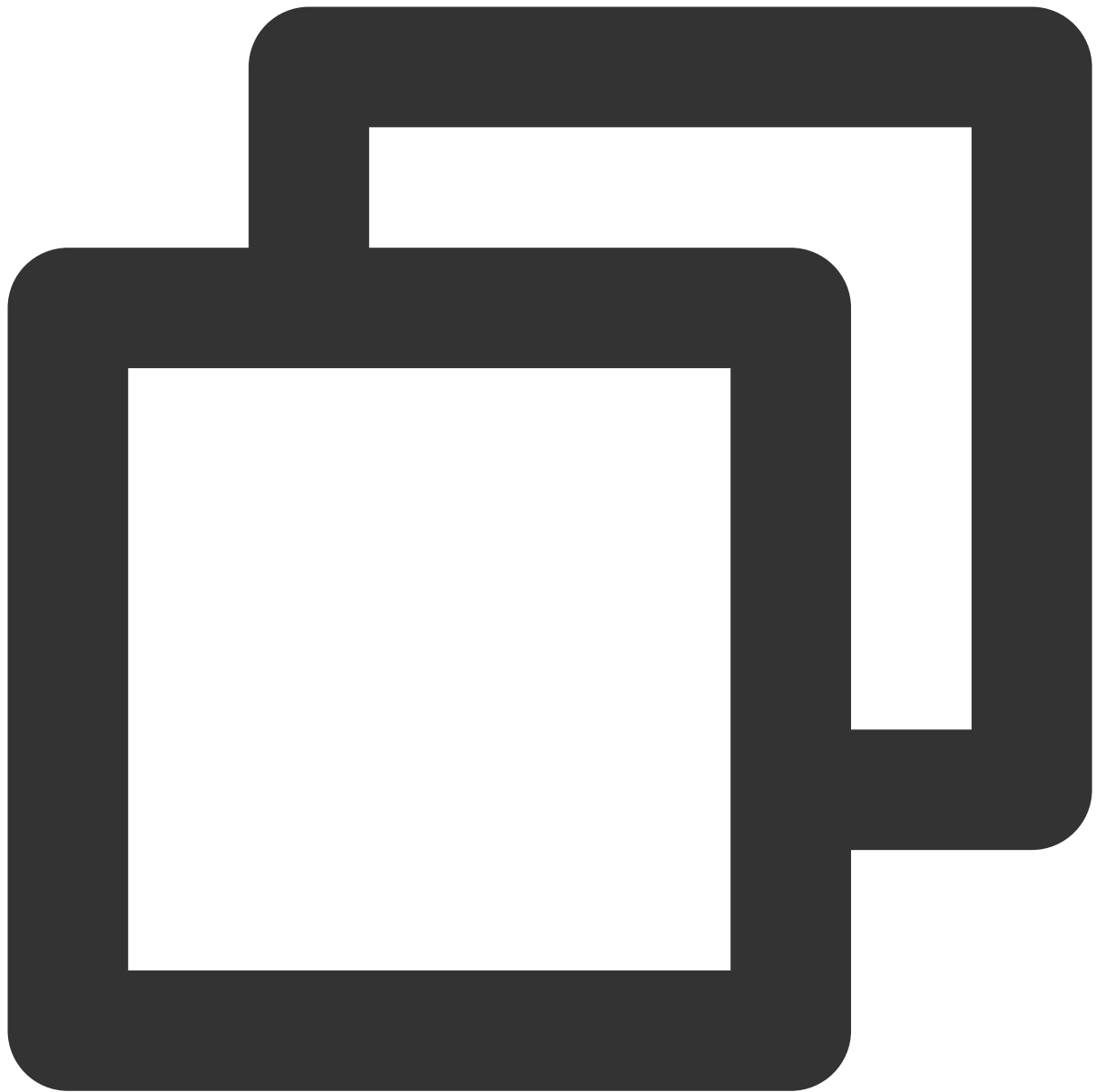
APIs

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.
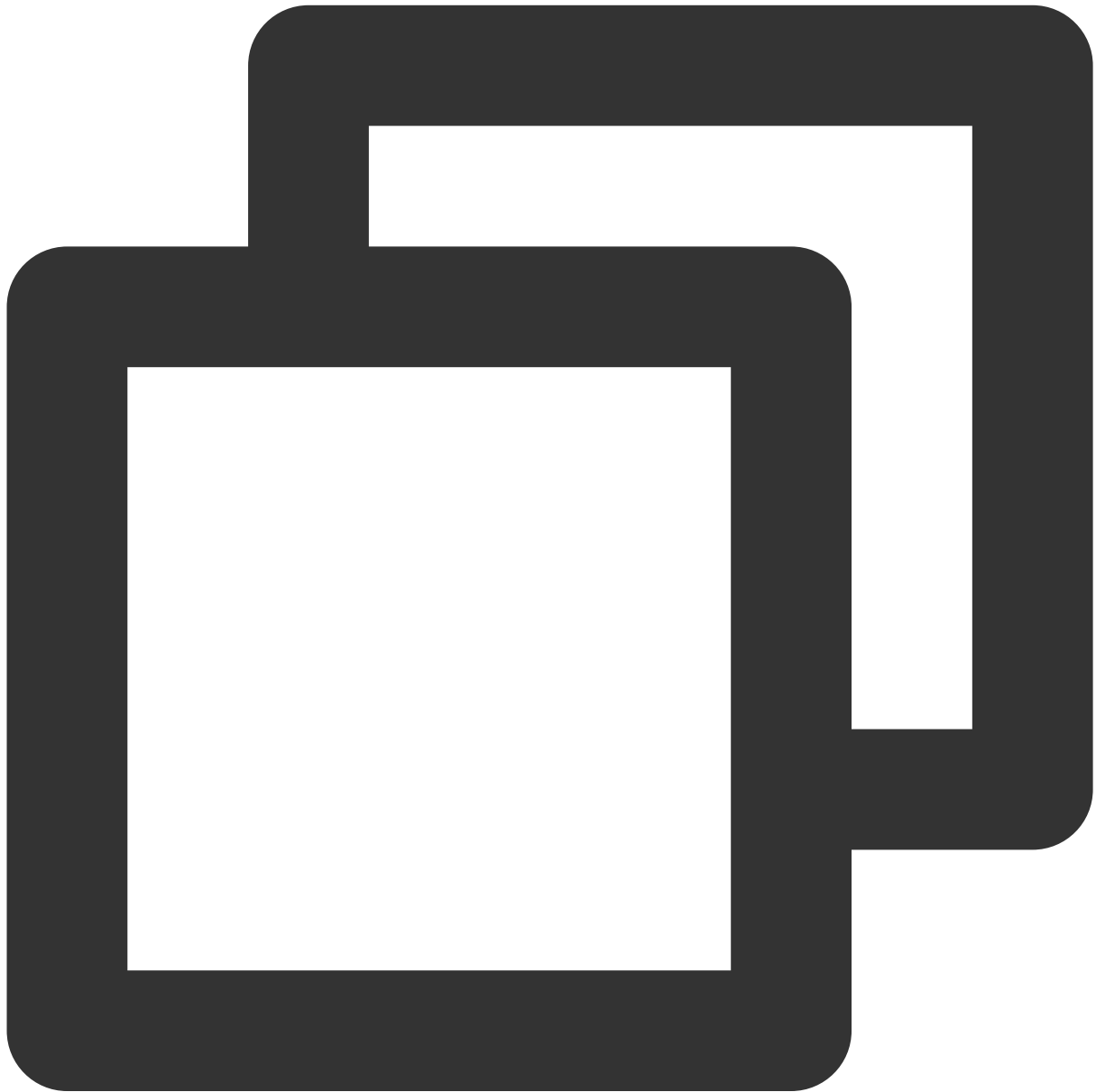
Sample input code:

```
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.

1.1 Call the CreateLiveRecordTemplate API to create a recording template.
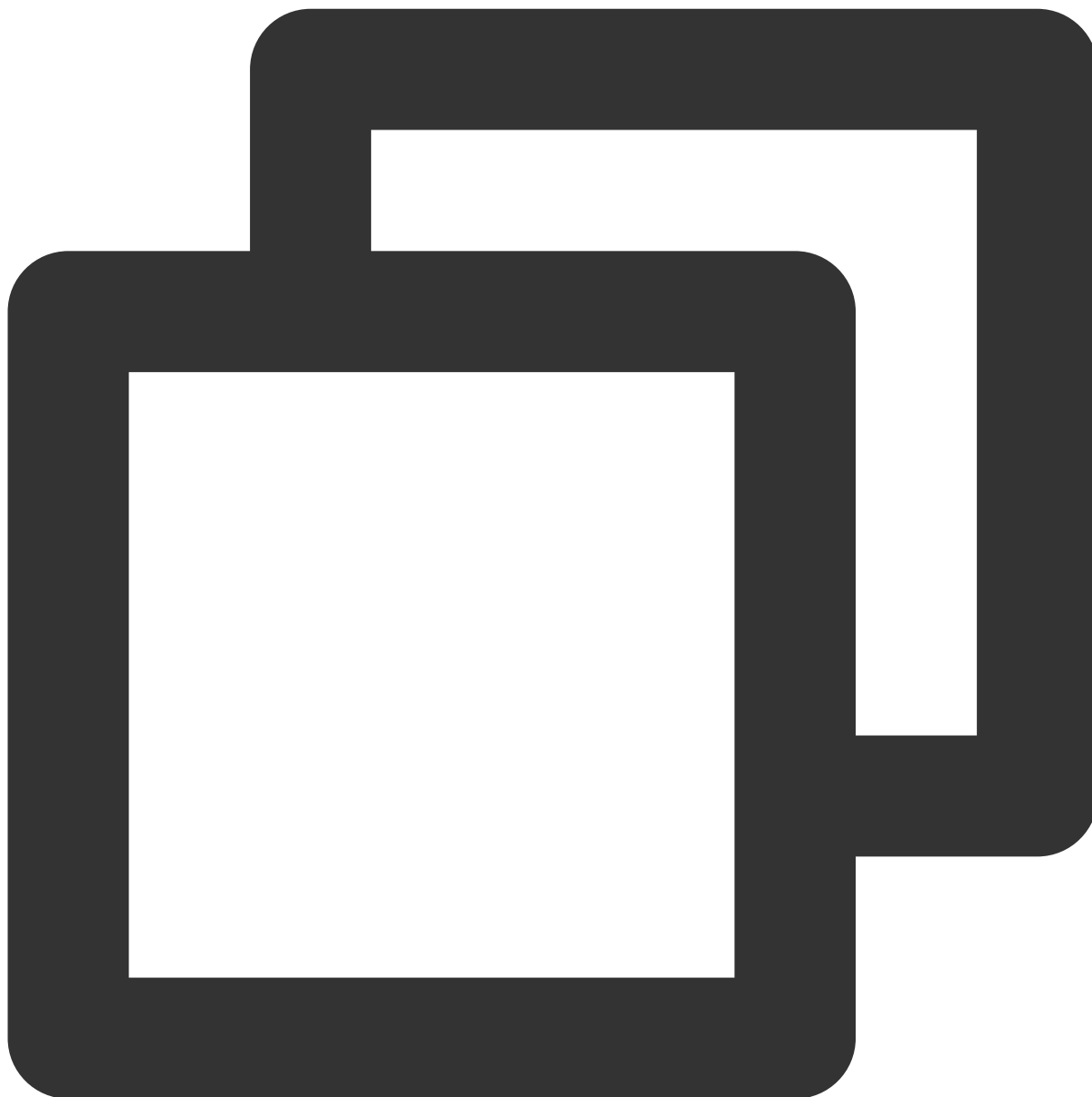
Sample input code:

```
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:
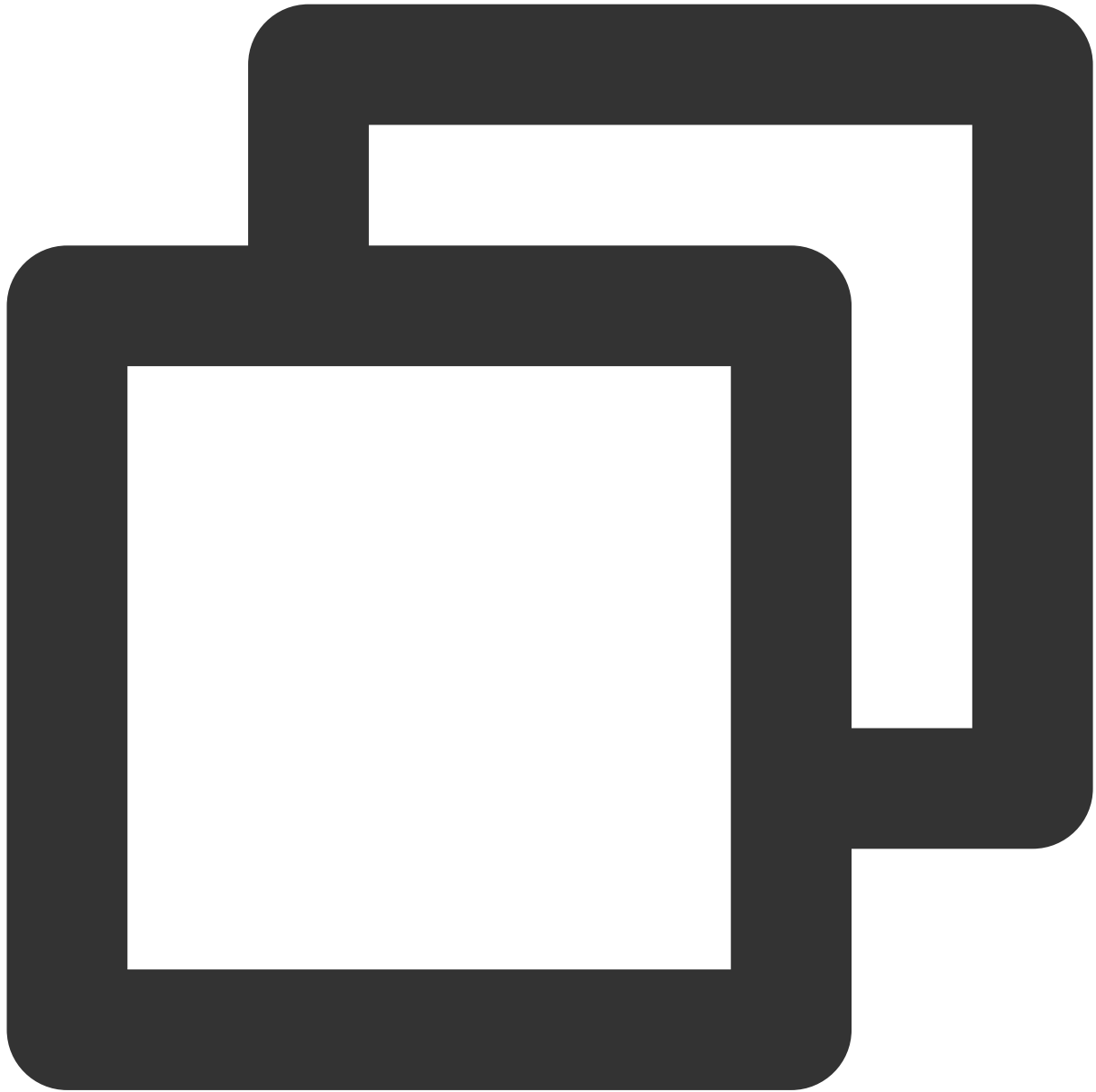


```
{
"Response": {
  "RequestId": "839d12da-95a9-43b2-a9a0-03366d01b532",
  "TemplateId": 17016
}
}
```

1.2 Call the CreateRecordTask 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.
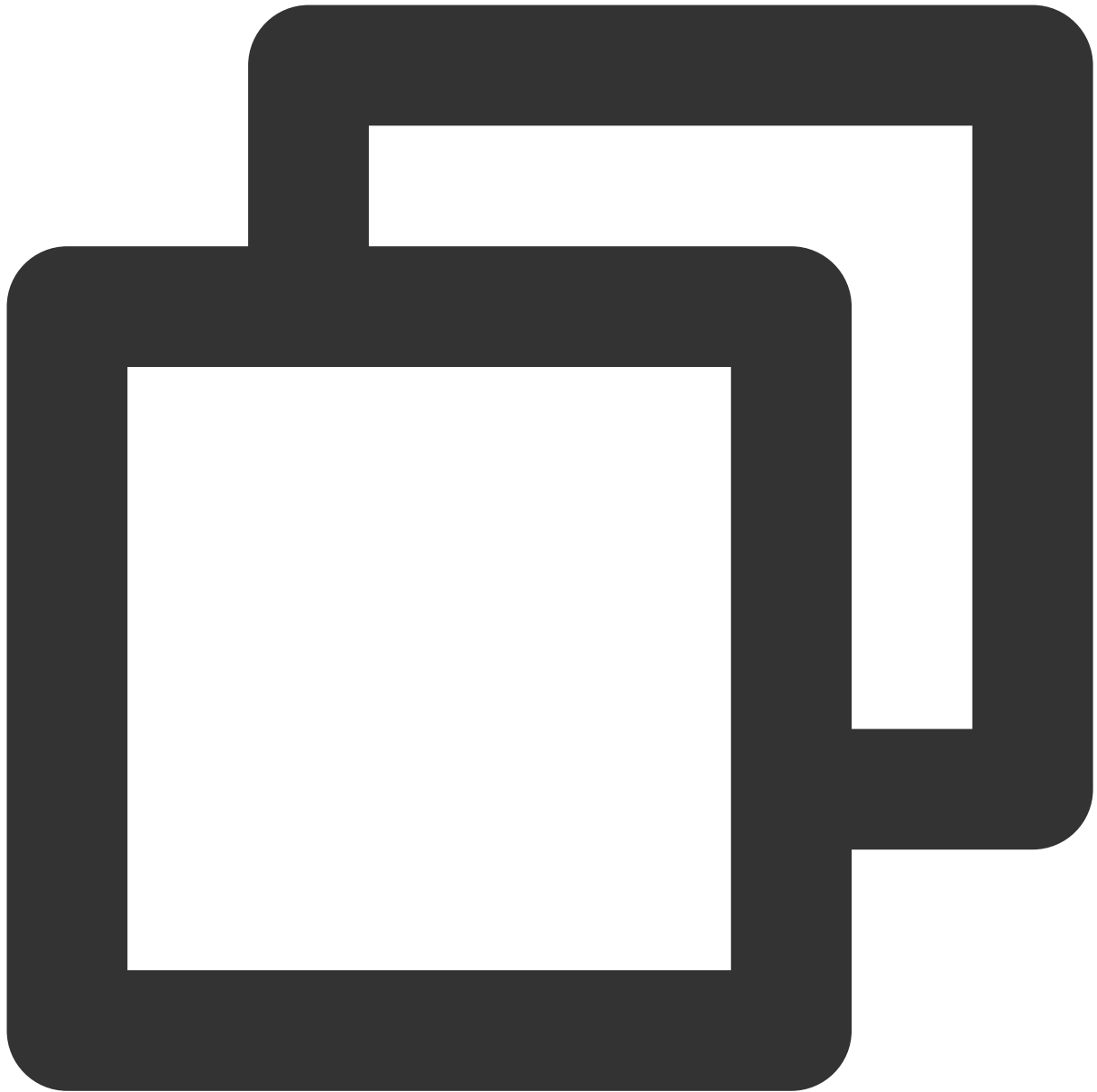
```
https://live.tencentcloudapi.com/?Action=CreateRecordTask
&StreamName=test
&AppName=live
&DomainName=mytest.live.push.com
&EndTime=1597024800
&<Common request parameters>
```

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

When using the mixed stream recording feature, please refer to the Live Stream Mixing documentation to understand the related knowledge and operation steps of the mixed stream service.

For scenarios using Live Cloud Mixing Services, the recording side classifies the mixed streams into two categories based on the `OutputStreamType` (output stream type) 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:

For the case where `OutputStreamType` is `0`, if stream C is stream A (with the same stream name but a mixed stream image), enabling the recording configuration will generate recording files for stream A (mixed stream image) and stream B by default. Since the same stream ID is reused, the original push of stream A will not generate a recording. For the case where `OutputStreamType` is `1`, enabling the recording configuration will generate recording files for streams A, B, and C (mixed stream image) by default.

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)

To address the issue of flash interruptions in push streams caused by network jitter and other factors at the push end, the recording service provides an automatic concatenation feature. This feature can record multiple interrupted push streams into a single file, making it convenient for live playback viewing.

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.
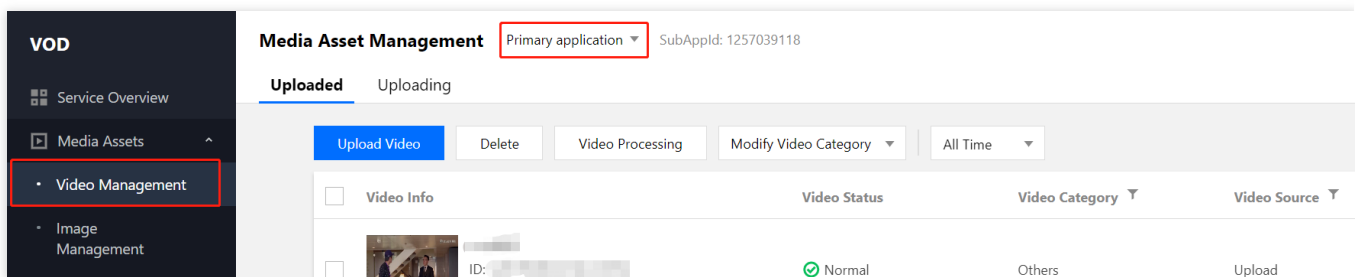
Recording event notification

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.

Recording to COS

Last updated：2024-07-10 17:52:36

Live recording is a service that stores the original live streaming files, which have undergone audio and video encapsulation (without modifying audio, video data, and corresponding timestamps, etc.) to the Cloud Object Storage (COS) platform.

Notes

To record to COS, you can 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.
Because there is a short delay in starting a recording task after a stream is pushed, a very short stream cannot generate a recording file. It is recommended that the duration of each push for recording be longer than 10 seconds.

Recording Storage

Because live recording stores files in COS, you must first activate COS before you can use live recording. To manage the storage duration of recorded files, you can configure a lifecycle for files stored in COS.

Recording Format

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

Recording Use Cases

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 part 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 COS, if you need to record live streams under a specific push domain, you can use the following methods:

CSS console

1. Go to **Feature Configuration** > Live Recording, click on Save to COS, and then add a recording configuration template.
2. Go to 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 to set `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. Upon successful processing, the settings will take effect within about 5 to 10 minutes.
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.
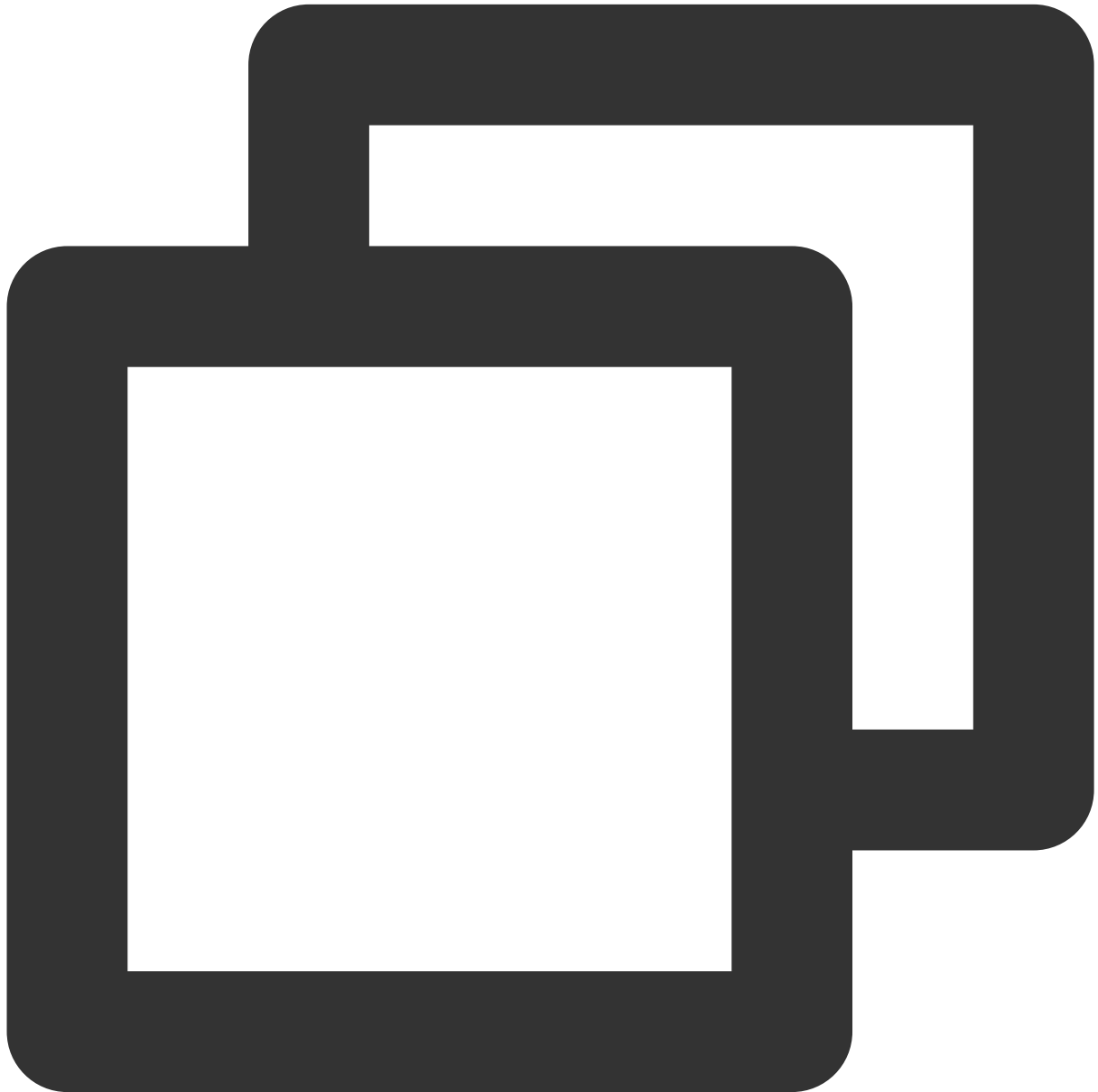
Priority	DomainName	AppName	StreamName
1	✓	✓	✓
2	✓	×	✓
3	✓	✓	×
4	✓	×	×

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

Disabling Recording for Specific Streams Under a Push Domain Name

If you have already configured recording for a push domain but do not need to record some streams under the domain, you can take the following steps:

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
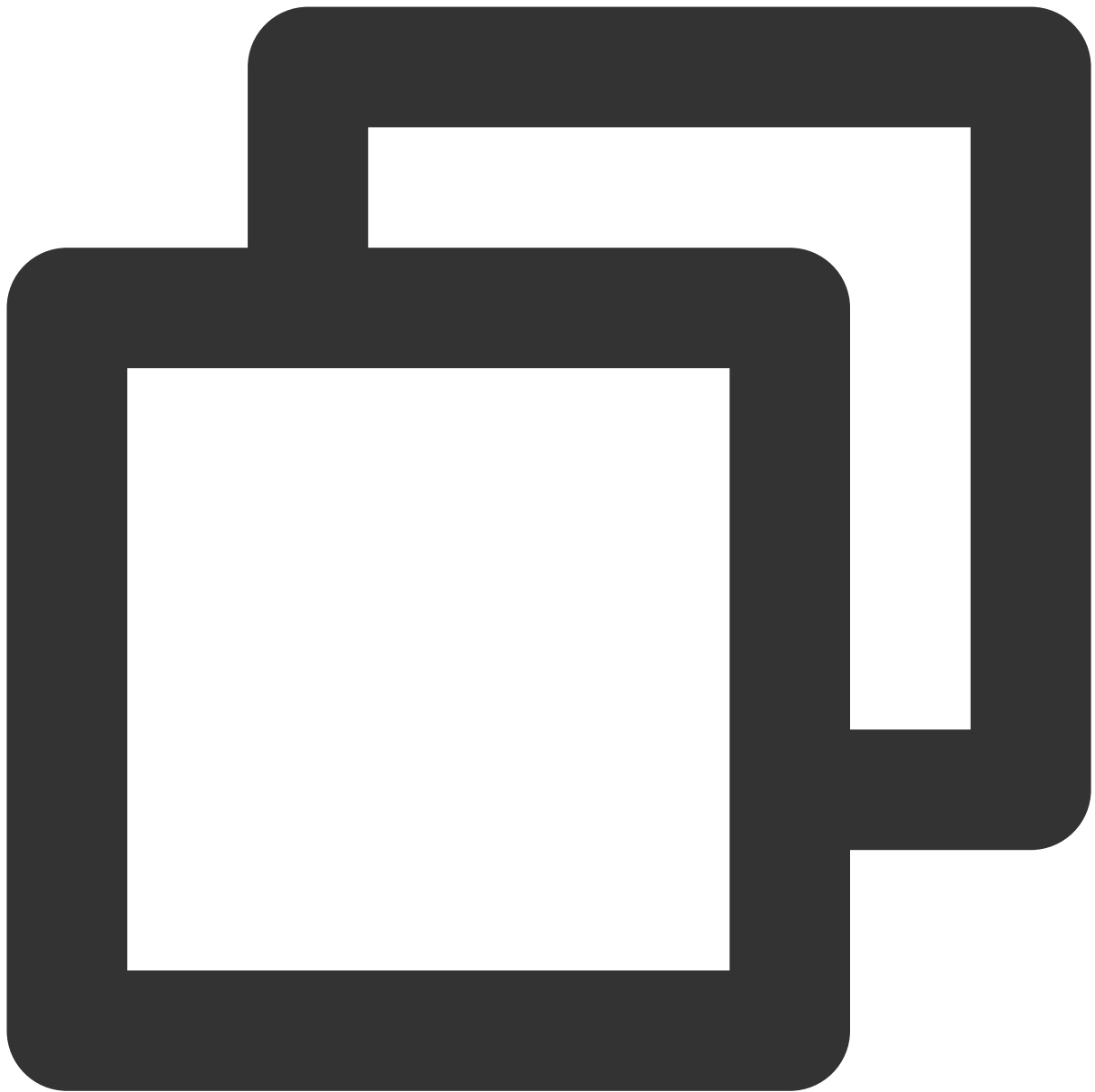template is created.

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.
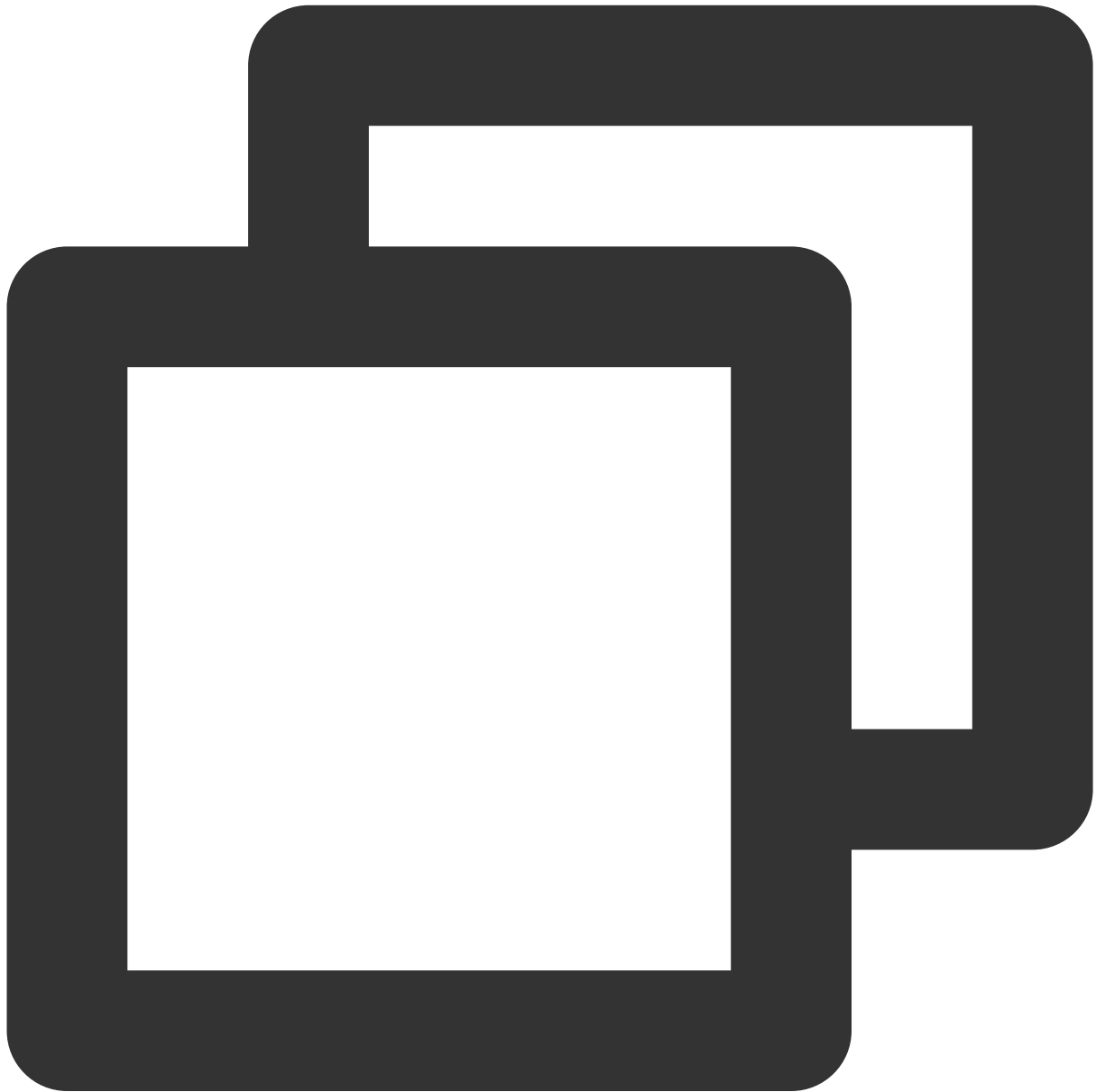
Sample input code:

```
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.

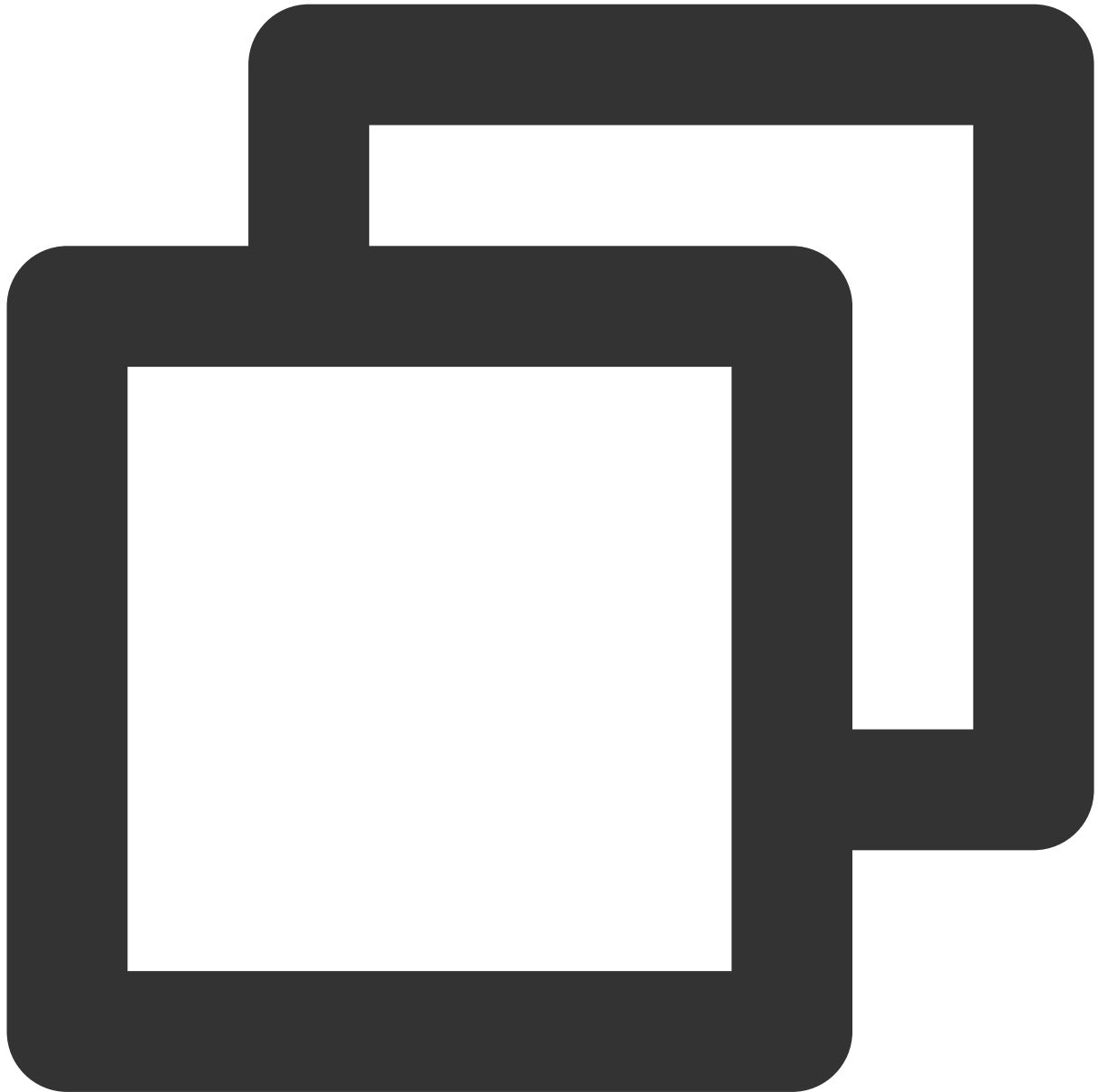1.1 Call the CreateLiveRecordTemplate API to create a recording template.

Sample input code:

```
https://live.tencentcloudapi.com/?Action=CreateLiveRecordTemplate
&TemplateName=templat
&Description=test
&Mp4Param.Enable=1
```

```
&Mp4Param.RecordInterval=3600
&<Common request parameters>
```
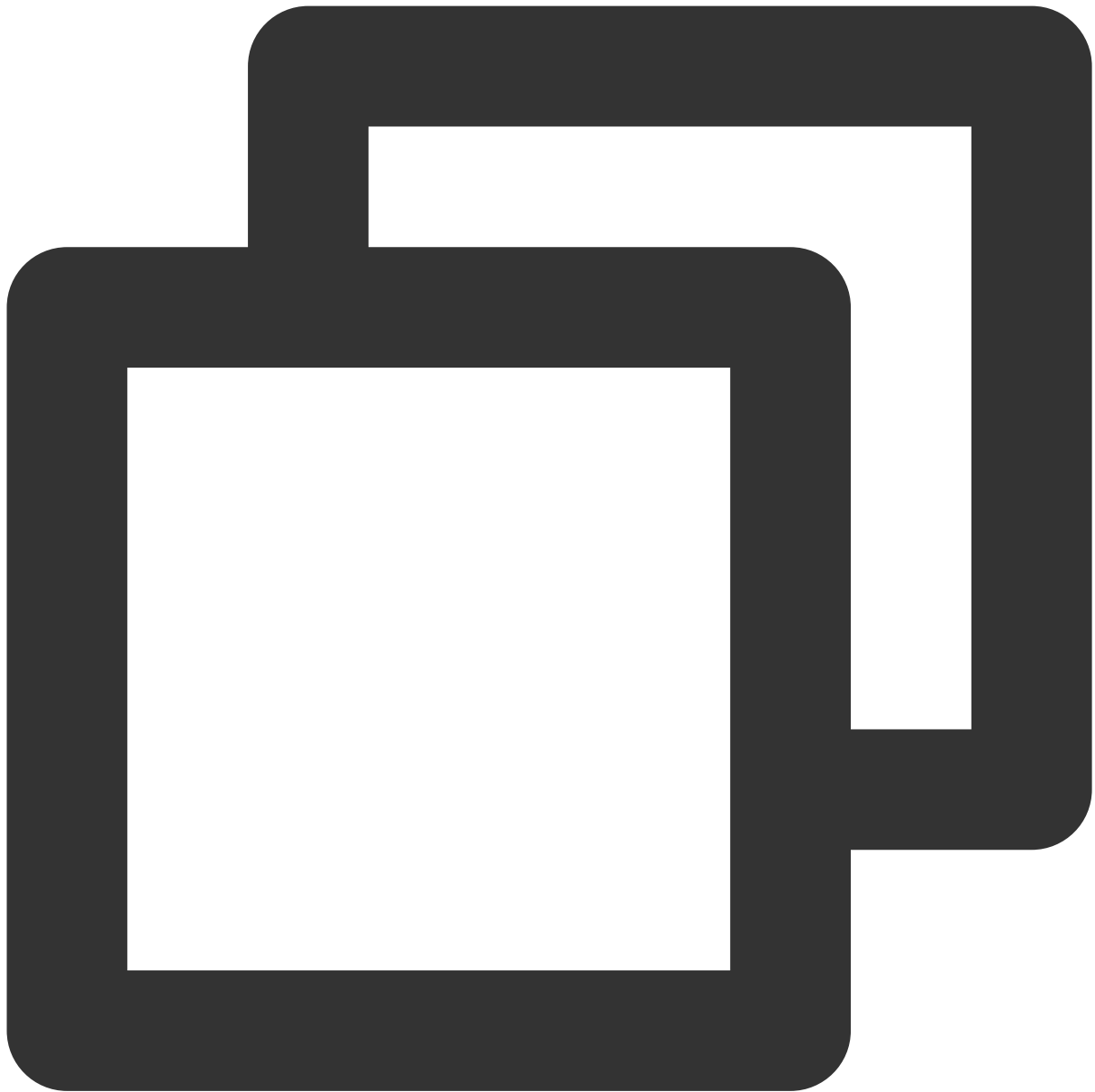
Sample output code:

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

1.2 Call the CreateRecordTask 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>
```
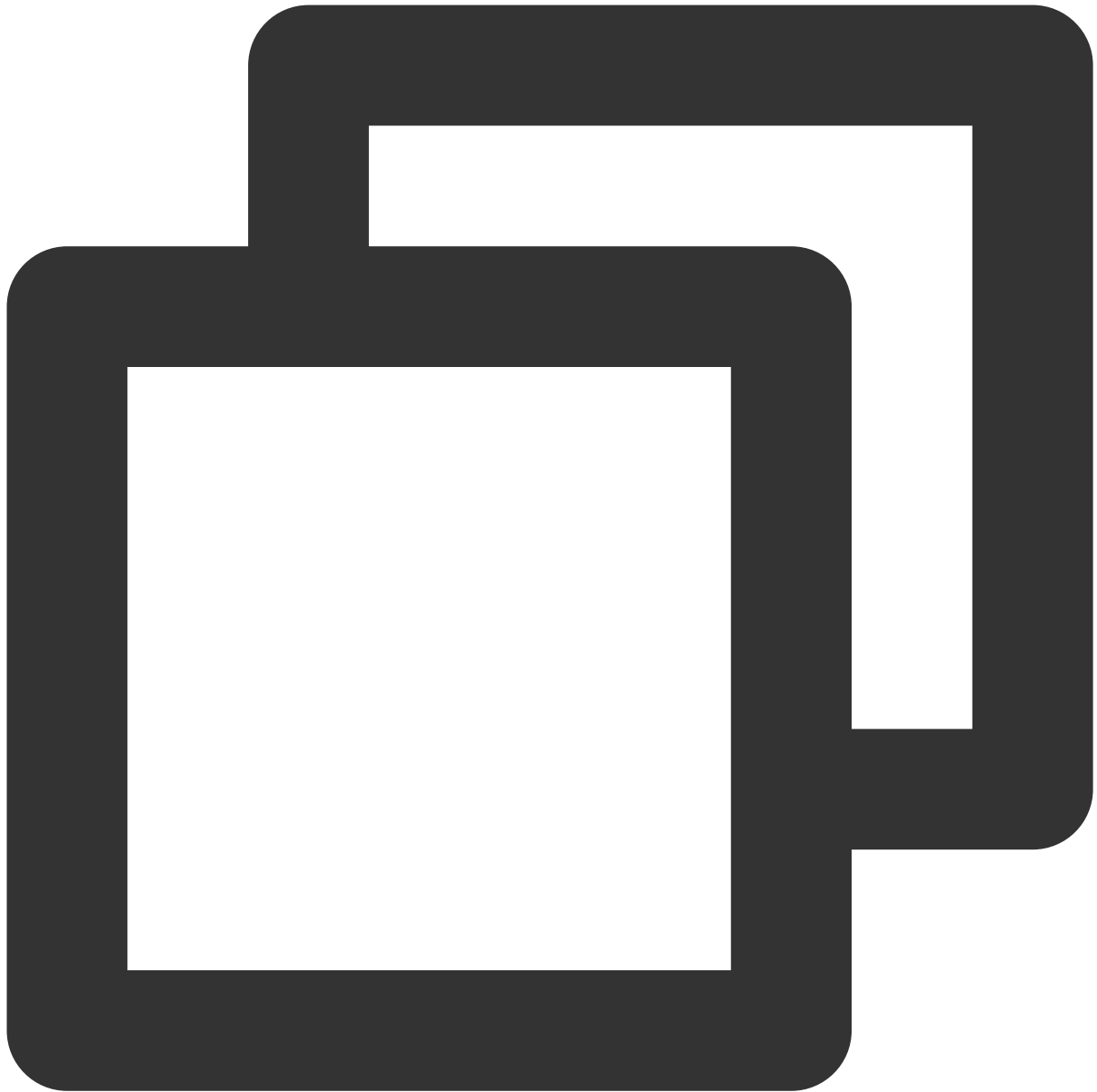
Note:

For a single live stream, scheduled tasks and other types of recording tasks will not conflict with each other. 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.

We recommend you 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 to slightly earlier than the event start time.

Real-Time Recording

If you want to record parts of a live stream in real time 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

When using the mixed stream recording feature, refer to Live Stream Mixing to get to know the relevant concepts and operations of the stream mixing service.

For scenarios using CSS on-cloud stream mixing, the recording side classifies the mixed streams into two categories based on 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:

For the case where `OutputStreamType` is `0`, if stream C is stream A (with the same stream name but a mixed stream image), enabling the recording configuration will generate recording files for stream A (mixed stream image) and stream B by default. Since the same stream ID is reused, the original push of stream A will not generate a recording. For the case where `OutputStreamType` is `1`, enabling the recording configuration will generate recording files for streams A, B, and C (mixed stream image) by default.

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 the Chinese mainland, as recording file errors will occur and affect normal playback.

Auto-Spliced Recording (Multi-Push Recording)

To address the issue of sudden interruptions in stream pushing caused by network jitter and other factors at the streaming end, the live recording service provides an auto-splicing feature, in which multiple interrupted push streams are recorded into a single file, making it convenient for live stream playback.

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 must support the **#EXT-X-DISCONTINUITY** tag. Currently, the tag is supported on the 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 when there are interruptions of up to 30 minutes during a recorded live stream, the recording files will 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.
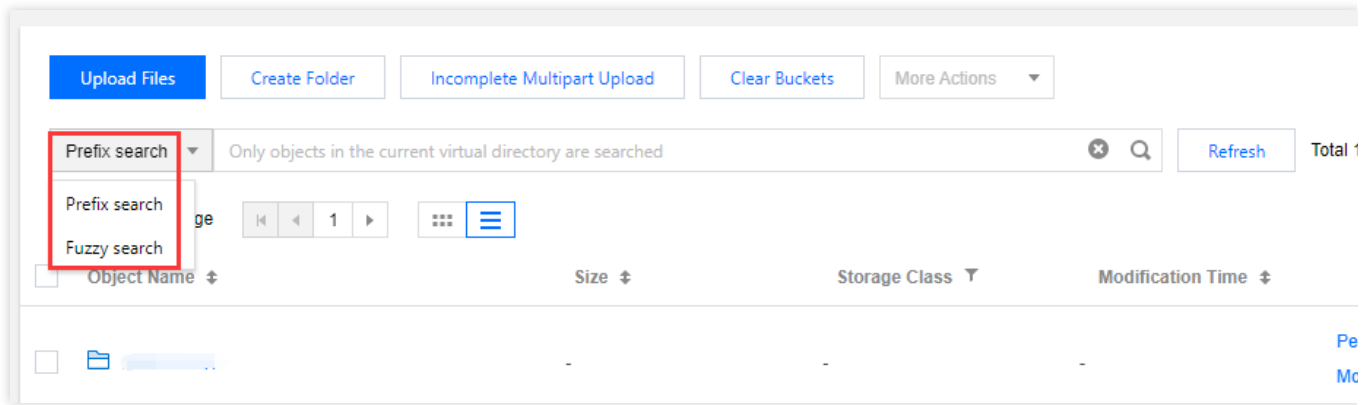
Obtaining Recording Files

Recording files are automatically saved in COS after they are generated and can be found using the following methods:

In the COS console

Log in to the COS Console, and click a **bucket name** from the list to browse and search for all the files generated by the recording.
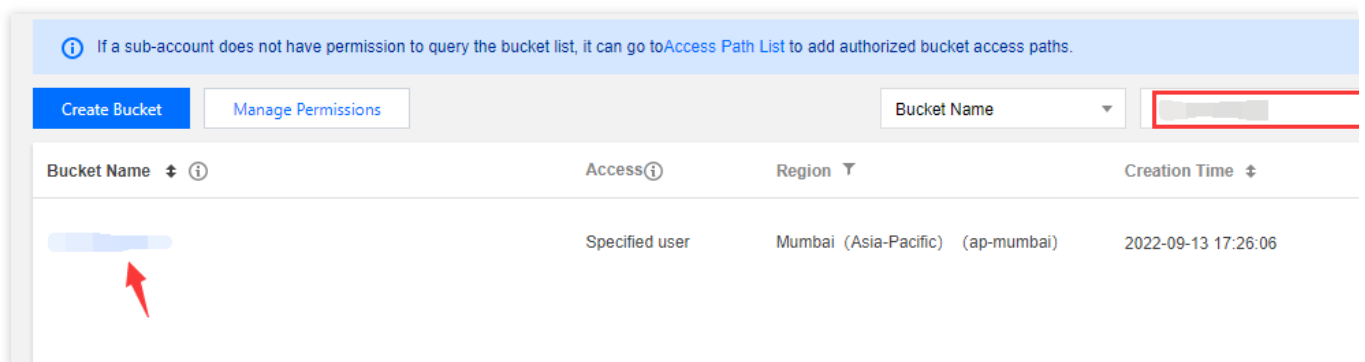
Querying the COS API

You can call the List Objects API of COS to filter and query recording files.

Using the 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 Recording Event Notification to take your next step.

We recommend using event notification callbacks to get recording files because they are efficient, reliable, and sent in real time.

Notes on Modifying Configuration

If you modify a live recording configuration, we recommend you restart the push and verify the recording 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 when recording is already in progress.

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：2023-10-08 14:21:06

We have recently upgraded the time shifting feature. When you create a time shifting template in the console, you will now be enabling the new time shifting feature. Generate a URL in the required format, and you can use the URL to play content from an earlier time point. API 3.0 is also now available for the time shifting feature. For details, see Time Shifting APIs. This document shows you how the time shifting feature works and how to make a playback request.

Must-Knows

The new time shifting feature currently supports 30,000 concurrent viewers. If you need a higher concurrency, please submit a ticket.

If you enabled authentication for your playback domain and configured an expiration time, the time shifting URL will expire after the specified time.

To use the old time shifting feature (which pulls content from a VOD domain), you need to submit a ticket. For a better experience, we recommend you use the new time shifting feature.

How It Works

CSS enables time shifting by saving live streams as TS segments and information about the playback time of each TS segment in the cloud. This feature is often used to replay TV programs or highlights of sports events. Content is distributed to clients over HLS. You can specify the exact playback time by setting the M3U8 request parameters (for details, see Playback Request).

Playback Request

The format of time shifting URL is `http://domain/appname/stream.m3u8` . There are two types of time shifting:

Playing a specific duration. This is suitable for replaying highlights of sports events.

Playing from a specific time ago. This is suitable if you want to delay the playback of a live stream.

Request parameters for playing a specific duration

Parameter	Description	Required	Example
txTimeshift	Whether to enable the new time shifting feature	Yes	txTimeshift=on

	(<code>on</code>: Enable).		
tsStart	The time-shift start time, the interval between tsStart and tsEnd cannot be less than one TS segment duration and cannot be more than 6 hours.	Yes	tsStart=20121010010101
tsEnd	The time-shift end time, the interval between tsStart and tsEnd cannot be less than one TS segment duration and cannot be more than 6 hours.	Yes	tsEnd=20121010010102
tsFormat	The format of tsStart and tsEnd is `{timeformat}_{unit}_{zone}` Valid values of timeformat unix - unix timestamp. If you use this format, you don't need to specify zone human - Human-readable time, such as "20121010010101". Valid values of unit:s/ms. s indicates second and ms indicates millisecond. zone：Time zones are divided into Eastern, Western, and Central Time Zones (Zero Time Zone). The value range for the Eastern Time Zone is 1 to 12 The value range for the Western Time Zone is -12 to -1.	Yes	tsFormat=unix_s tsFormat=human_s_8
tsCodecname	For a transcoded stream, set this parameter to the ID of the transcoding template. For original streams or watermarked streams, leave out this parameter.	No	tsCodecname=hd

Request example 1 (Unix timestamp)
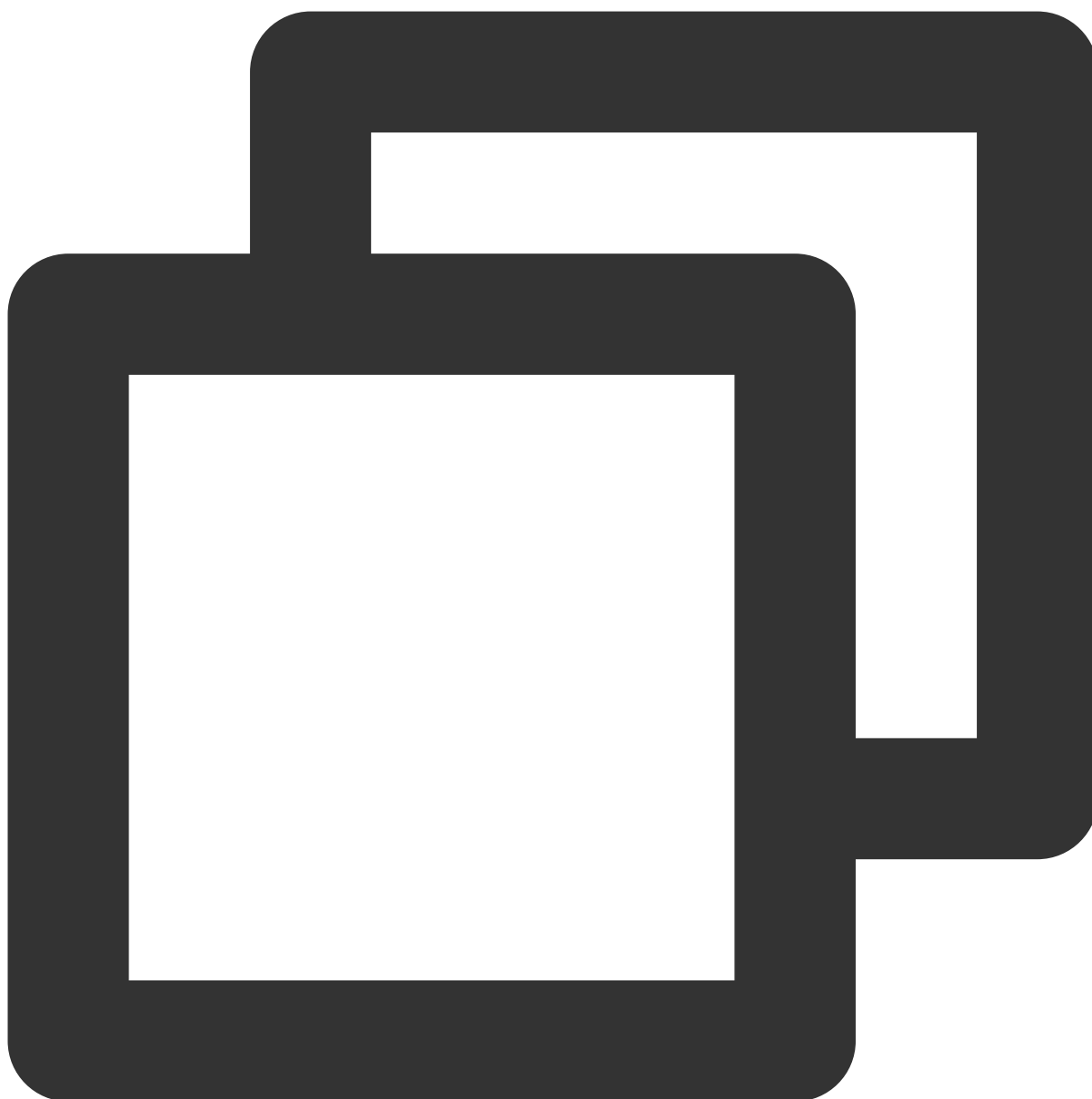
```
http://example.domain.com/live/stream.m3u8?txTimeshift=on&tsFormat=unix_s&tsStart=1
```

Request example 2 (human-readable time)

```
http://example.domain.com/live/stream.m3u8?txTimeshift=on&tsFormat=unix_s_8&tsStart
```

Request parameters for playing from a specific time ago

Parameter	Description	Required	Example
txTimeshift	Whether to enable the new time shifting feature (`on` : Enable).	Yes	txTimeshift=on
tsDelay	The number of seconds to delay the	Yes	`tsDelay=30` indicates that

	playback by.		playback will start from 30 seconds ago.
tsCodecname	For a transcoded stream, set this parameter to the ID of the transcoding template.	No	tsCodecname=2000

Request example
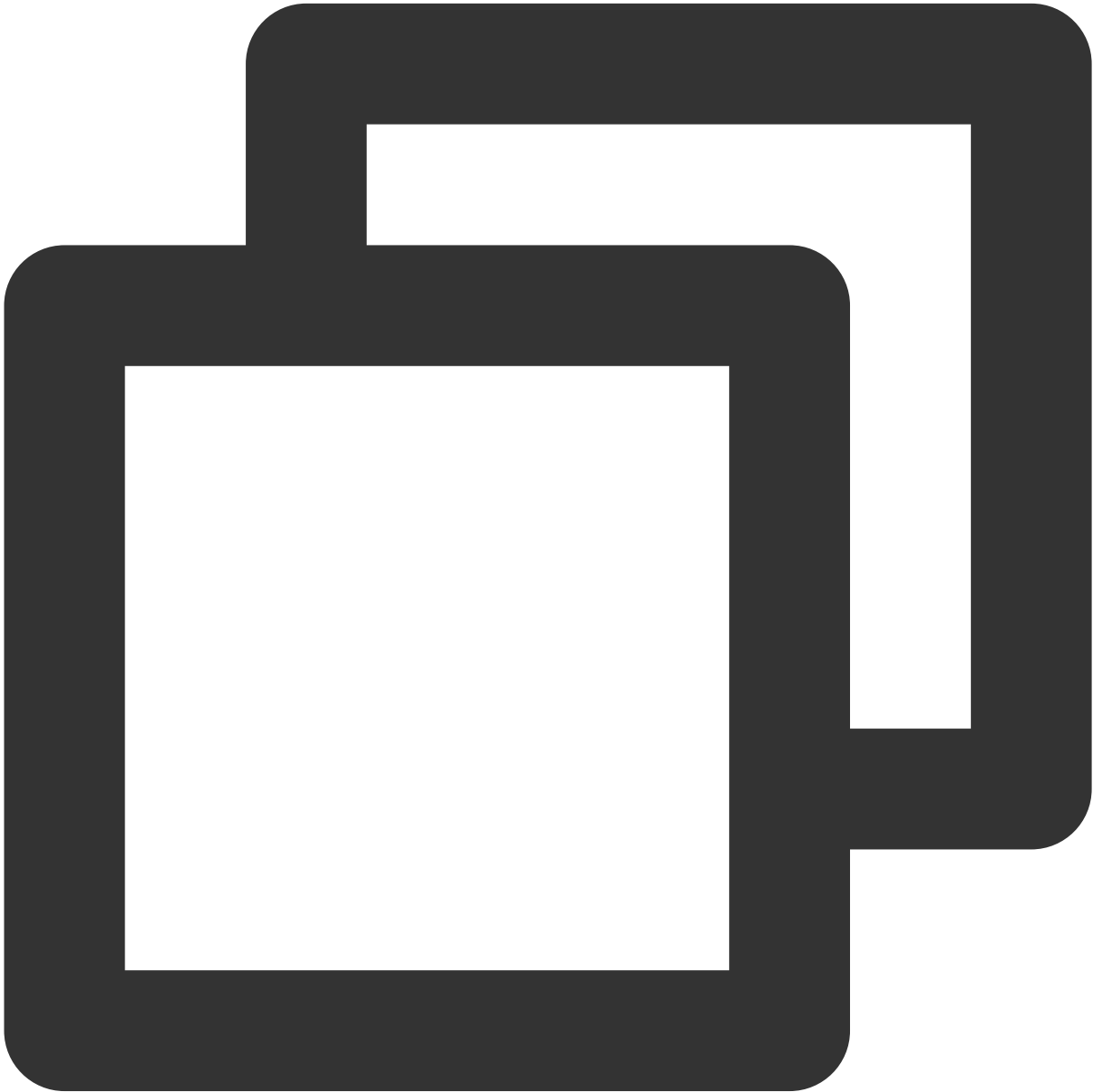
```
http://example.domain.com/live/stream.m3u8?txTimeshift=on&tsDelay=30&tsCodecname=t
```

Authentication parameters

The authentication parameters for time shifting are the same as those for playback. For details, see Playback Authentication Configuration (HLS URLs generated in the console are only valid for one day).

Querying time shifted streams

The **Time Shifting > Time shifting details** page of the console shows a list of time shifted streams. You can click **Details** to view the details of a time shifted stream.

You can also use APIs to query time shifted streams and the details of a specific stream. For details, see the following documents:

DescribeTimeShiftStreamList

DescribeTimeShiftRecordDetail

Live Streaming Highlights Editing

Last updated：2024-07-22 16:36:54

Live streaming highlights rely on the time-shifting capabilities, which means that during or after the live stream, a user can select an exciting segment from the past live content, generate a time-shifted playback address, and conveniently redistribute the highlights. Additionally, through media processing capabilities, live streaming highlights can be solidified into object storage for long-term preservation.

This article introduces how to use live streaming time-shifting to create highlights from live content, as well as how to use media processing capabilities to solidify and store live streaming highlight clips.

Prerequisites

Using the live streaming highlights clipping feature, please make sure you have created a time-shift template, bound the streaming domain, and successfully streamed.

To use the live broadcast editing and curing capability, please make sure you have activated the Tencent Cloud MPS service. The solidified content will be stored in the COS Service, and activating the MPS service will automatically activate the COS service.

Live Streaming Highlights Editing

Console operation guide

Based on your business needs, you can configure time-shifted playback. For more details, please refer to the Live Streaming Time-Shift Index Information document.

Directions

1. After using the domain name bound to the time-shifting template for streaming, select **Feature Configuration** > **Time shifting** > Time shifting details in the left menu bar to enter the index information page.
2. Select the live stream you want to clip, and click on **Details** on the right side to enter the time-shift details page.

3. You can view the push address and time-shifted content in **Basic Info**.

4. You can move the mouse in the timeline of the **Index Details** to view the position time. Click on the **timeline** to **mark the time**.

5. Click on the timeline to mark the time and preview the time-shifted content.

6. Configure playback domain name: Select the corresponding time-shift playback domain name.

7. To generate a time-shift playback address: Click **Generate Address** to create a time-shift playback address, which supports one-click copying of the address.
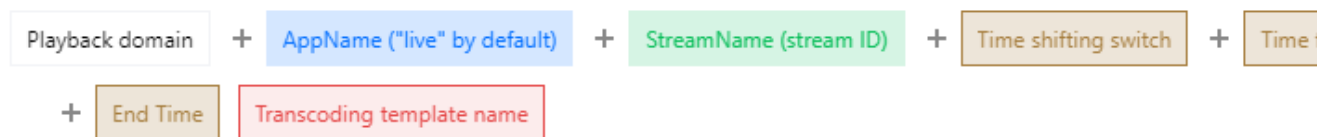
Splice the Address Yourself

You can refer to the Time Shifting Function Practice document to splice the addresses of live broadcast highlight clips according to the rules. You can also refer to the following splicing rules to assemble the addresses of live broadcast highlight clips by yourself.

Live Clip Solidification

Create Clip Persistence Task

1. **Navigate to MPS solidification of time-shifted content**: Choose the solidification file format, either **MP4 type** or **HLS type**.

2. When you click on **Configure storage**, the system will automatically redirect you to the **MPS** > **Create Task** page. On this page, the system will automatically fill in the time-shift highlights clip address and the corresponding transcoding service orchestration template. You can configure the task according to your business needs.

3. Select the output path and click **Create**. For more details, please refer to the Task Management document.

4. Once the task creation is completed, you can view the execution status and results of the video editing task. You can enter the corresponding directory in COS, and in that directory, you can search for the edited and solidified files. Typically, the file name and path will be determined according to the output parameters you set when creating the task.

Create a video editing and solidification task through the API

You can create a video editing and solidification task by initiating a Media Processing Service (MPS) API request:

Enter the input and arrangement ID to initiate the task, and then click View API Information.

Note:

Tencent Cloud provides you with default video editing and solidification remuxing service arrangement templates. The default MP4 remuxing arrangement template ID is 10101, and the default HLS remuxing arrangement template ID is 10100.

You can customize and create video transcoding templates based on your business needs. Click Create Video Transcoding Template to enter the template customization settings. For more information, please refer to the Audio and Video Transcoding Templates documentation.

MPS will charge you based on your task type. For more information, please refer to the Media Processing Service Billing documentation.

Example:
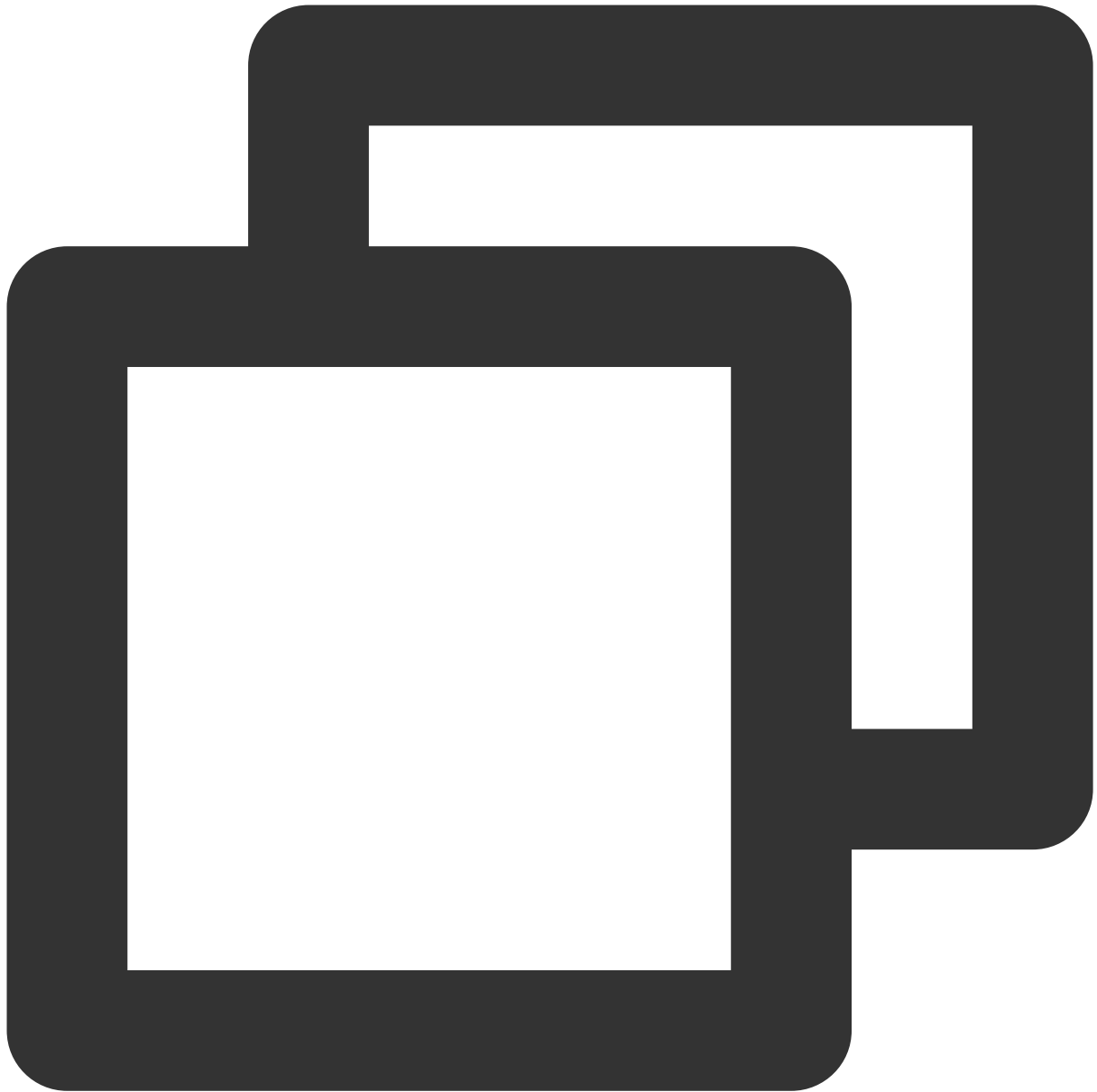
```
{
    "InputInfo": {
        "Type": "URL",
    "UrlInputInfo": {
      "Url": "http://domain/AppName/StreamName.m3u8?txTimeshift=on&tsFormat=human_s
    }
  },
  "OutputStorage": {
    "Type": "COS",
    "CosOutputStorage": {
      "Bucket": "test-130000000",
```

```
      "Region": "ap-nanjing"
    },
  },
  "OutputDir": "/Liveclip/",
  "ScheduleId": 10101,
    "Action": "ProcessMedia",
    "Version": "2019-06-12"
 }
```

Should a callback URL be configured, please refer to the documentation for the response package: Parsing Event Notifications.

Live Screencapture

Last updated：2024-07-22 16:36:54

The live screencapture 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 Screencapture Process
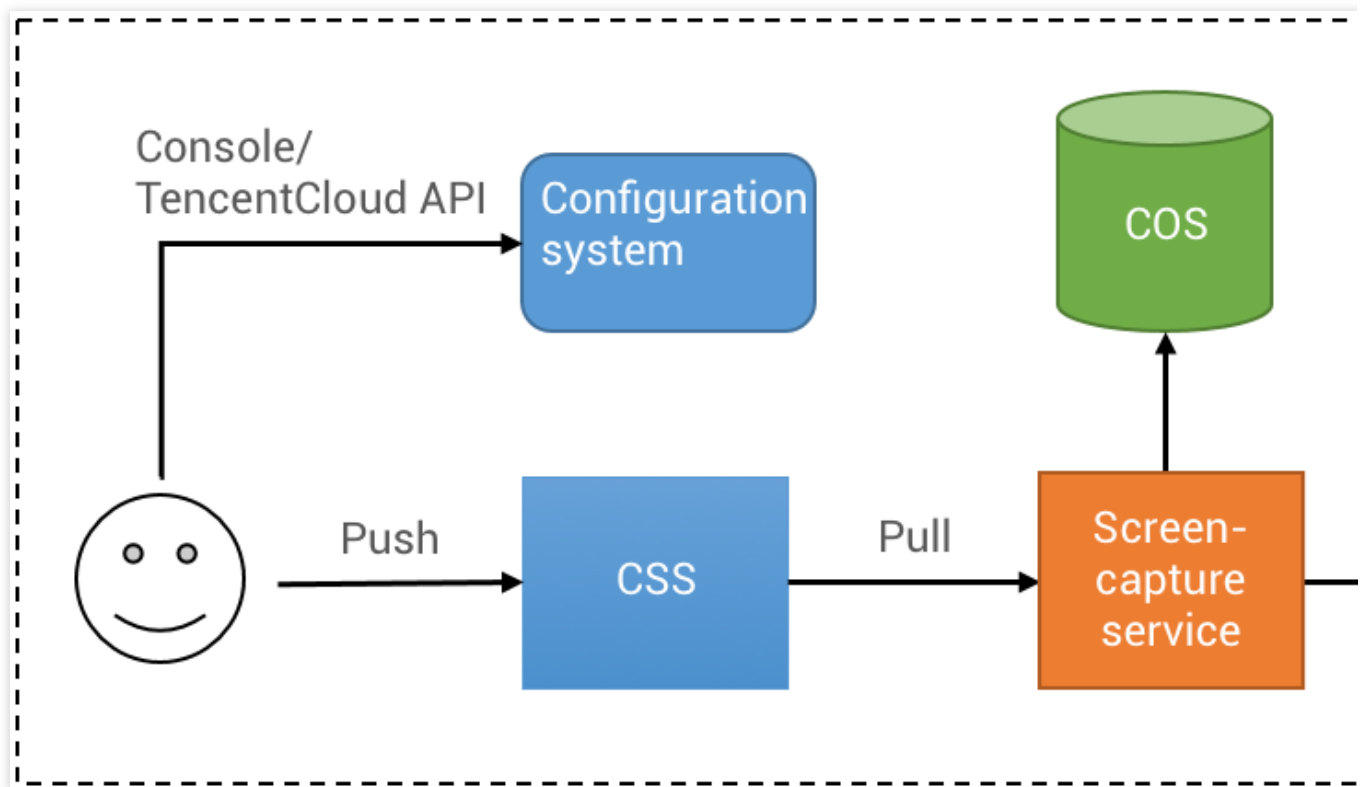
Overall process:

1. Configure the live screencapture feature in the console or through TencentCloud API.

2. Start live push.

3. The screencapture 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 Screencapture Configuration

Screencapture configuration method

CSS API

CSS console > **Feature Configuration** > Live Screencapture, For details, seeLive Screencapture.

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 2–300 seconds with a default interval of 2 seconds.

Screenshot width and height configuration

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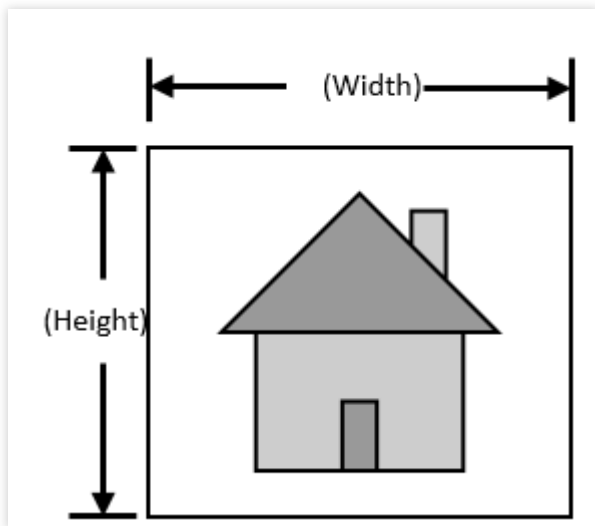
Note:

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.

Note:

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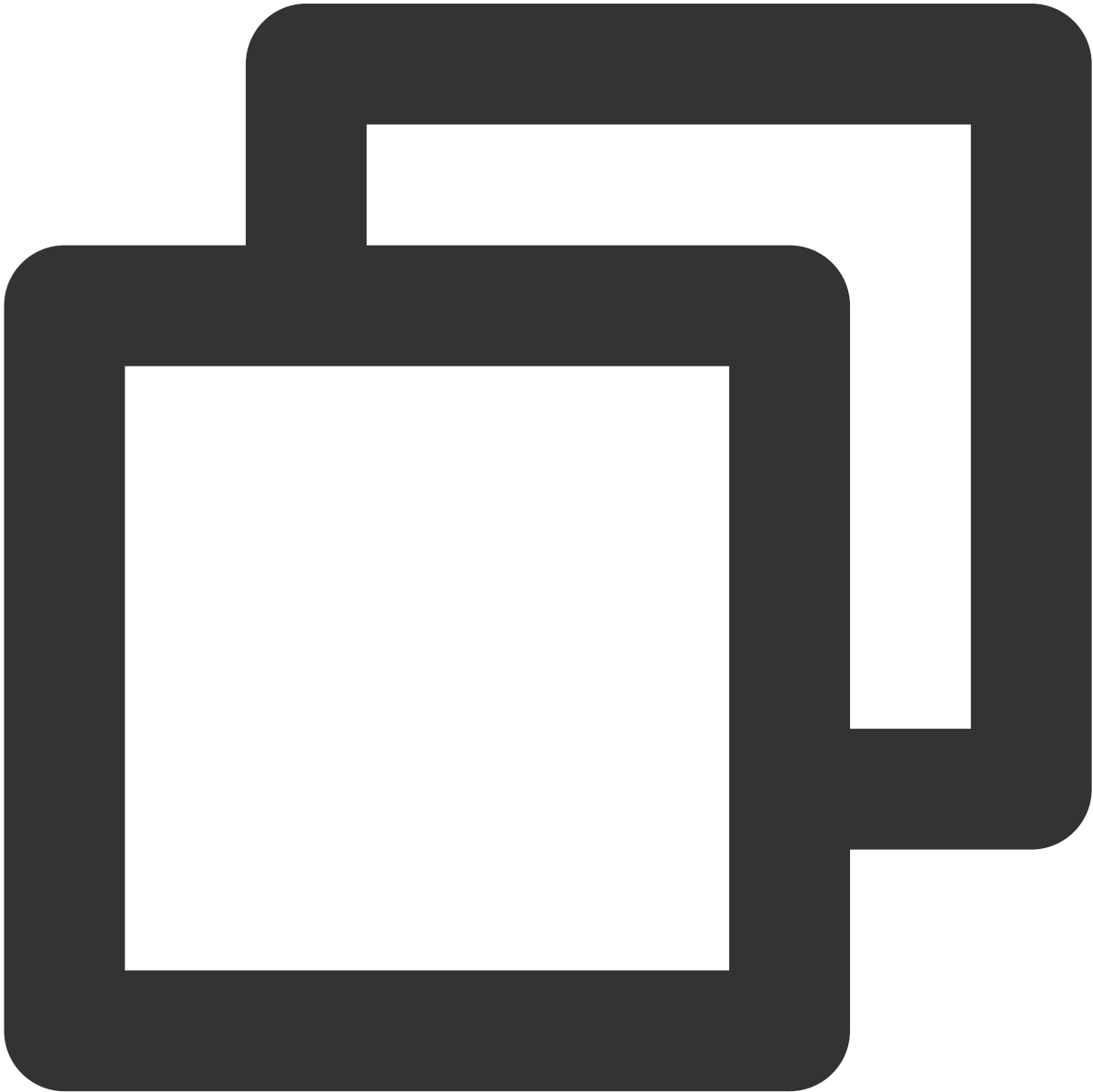
```
/2018-12-17/stream_name-screenshot-19-06-59-640x352.jpg
```

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:
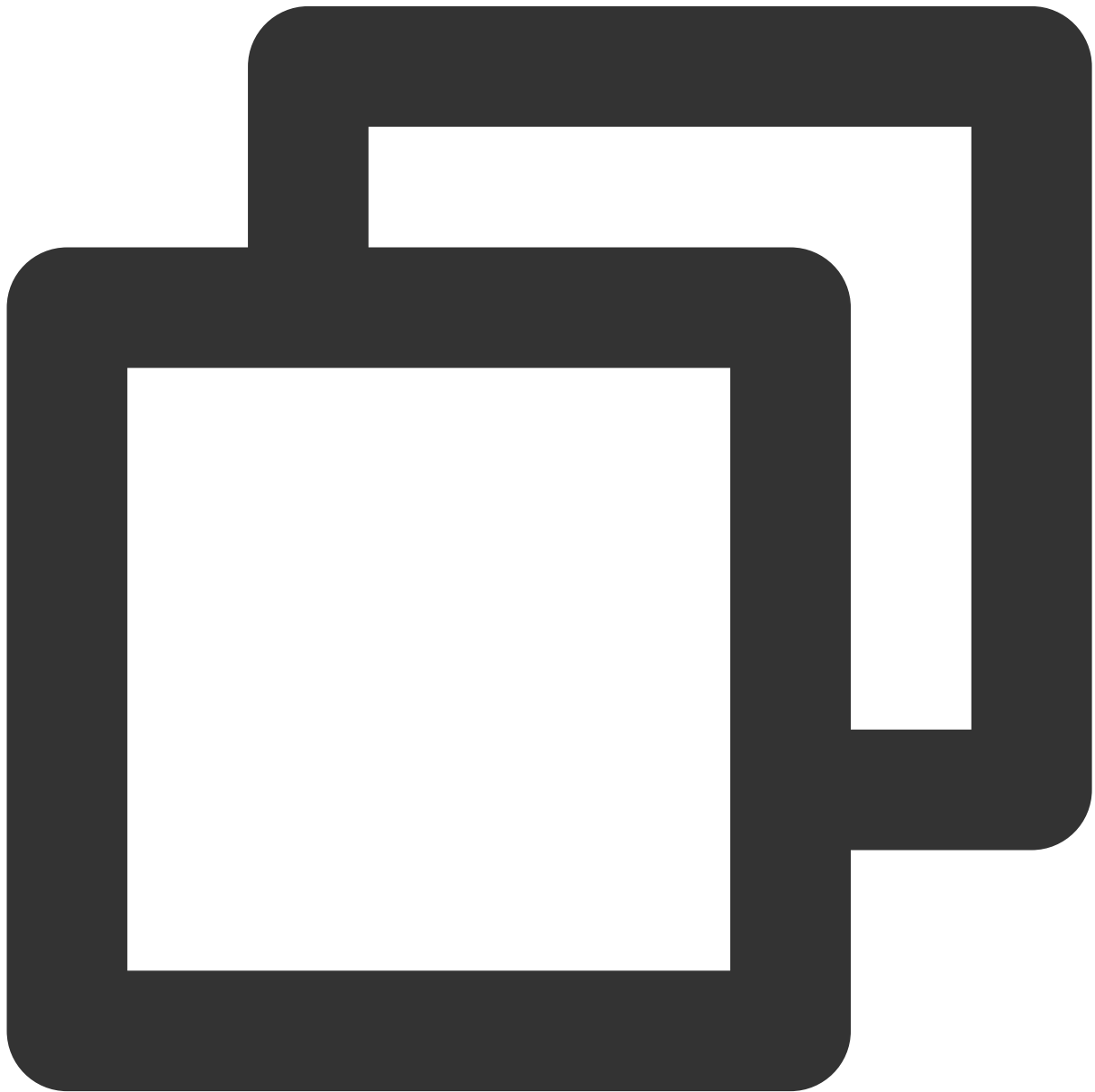
```
http://testbucket-1234567890.cos.region.myqcloud.com/2018-12-17/stream_name-screen
```

Sample screencapturing callback
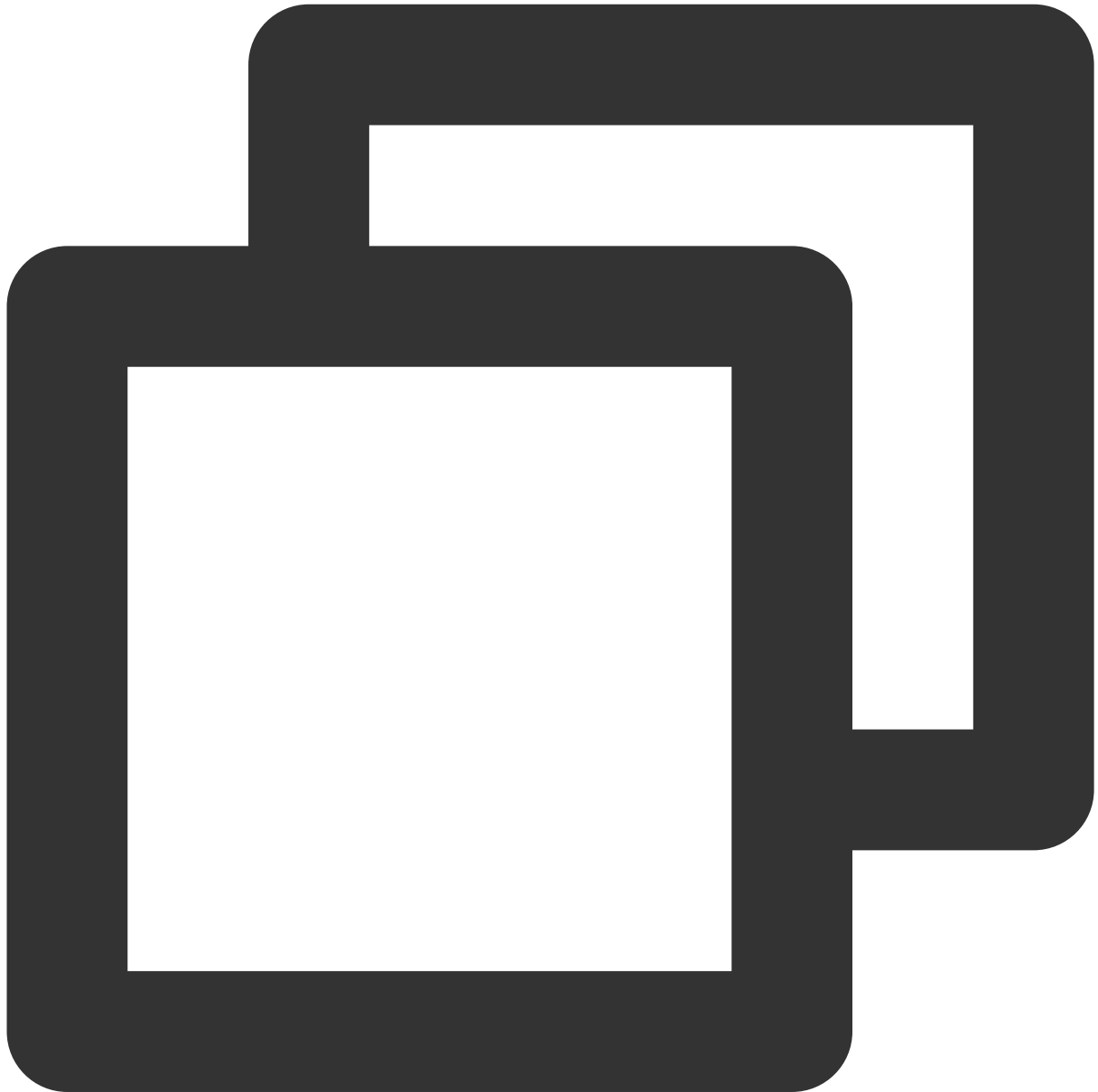

```
{

"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/str

"sign":"ca3e25e5dc17a6f9909a9ae7281e300d",

"t":1545030873

}
```

Live Porn Detection

Last updated：2023-02-27 15:47:12

To use the porn detection capability of CSS, you need to enable screencapturing. You can use the porn detection feature either by configuring it in the [CSS console](#) or by using APIs. This document shows you how to use porn detection APIs.

Must-Knows

If your COS bucket allows public read access and has politically sensitive, pornographic, or other inappropriate content, to avoid the bucket being blocked, please delete the content first.

Enabling Porn Detection

Because the porn detection feature is based on screencapturing, you need to enable screencapturing first.

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](#), setting `TemplateId` to the ID of the screencapturing template created in step 1 to associate the template 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 the 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, callbacks are sent only if suspicious content is detected.

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, setting `TemplateId` to the ID of the callback template created in step 1 to associate the template with the target `AppId`, `DomainName`, and `AppName`.

3. Get the porn detection result

The CSS backend will send porn detection results to your domain in the form of HTTP POST requests. You can find the results in JSON format in the request body. The `type` field indicates whether a live stream contains pornographic content.

> Note：
>
> The system cannot achieve 100% accuracy. There may be false positives or false negatives. We recommend you review the images suspected of being pornographic using the `type` field.

The parameters are as follows:

Parameter	Required	Data Type	Description
streamId	No	String	The stream name.
channelId	No	String	The channel ID.
img	Yes	String	The link of the suspicious image.
type	Yes	Array	The value of the label with the highest priority in the detection result. For details, see the description in `label`.
score	Yes	Array	The confidence score.
ocrMsg	No	String	The OCR result (if any).

Parameter	Required	Data Type	Description
suggestion	Yes	String	The suggestion. Valid values: • Block • Review • Pass
label	Yes	String	The label with the highest priority in the detection result (`LabelResults`). This is the result generated by the model. Please handle different types of violations based on your business needs.
subLabel	Yes	String	The sub-label of the label with the highest priority in the detection result, such as porn - sexual acts. If no sub-labels are hit, this field will be empty.
labelResults	No	Array of LabelResult	The label hit results generated by the category model, including the detection of pornographic content, ads, terrorist content, and politically sensitive content. Note: **This field may return `null`, indicating that no valid values can be obtained.**
objectResults	No	Array of ObjectResult	The detection results generated by the object model, including the label name, hit score, coordinates, scenario, and suggestion for suspicious objects, advertising logos, QR codes, etc. For details, see the data structure of `ObjectResults`. Note: **This field may return `null`, indicating that no valid values can be obtained.**
ocrResults	No	Array of OcrResult	The OCR result, including the text recognized, the text coordinates, and the suggestion. For details, see the data structure of `OcrResults`. Note: **This field may return `null`, indicating that no valid values can be obtained.**
libResults	No	Array of LibResult	The detection results generated by the block/allowlist library.
screenshotTime	Yes	Number	The time when the screenshot was taken.
sendTime	Yes	Number	The Unix timestamp when the request was sent.
stream_param	No	String	The push parameter.
app	No	String	The push domain.

Parameter	Required	Data Type	Description
appid	No	Number	The application ID.
appname	No	String	The push path.

LabelResult

The label hit result generated by the category model.

Parameter	Type	Description
Scene	String	The scene identified by the model, such as advertising, pornographic, and harmful.
Suggestion	String	The operation suggested by the system for the current label. Please handle different types of violations based on your business needs. Valid values: • Block • Review • Pass
label	String	The label hit.
SubLabel	String	The sub-label.
Score	Integer	The confidence score for the label.
Details	Array of LabelDetailItem	The sub-label hit details.

LabelDetailItem

The sub-label hit details.

Parameter	Type	Description
Id	Integer	The sequence ID.
Name	String	The sub-label.
Score	Integer	The sub-label score. Value range: 0-100.

ObjectResult

The object detection result.

Parameter	Type	Description
Scene	String	The object scene identified, such as QR code, logo, and OCR.
Suggestion	String	The operation suggested by the system for the current label. Please handle different types of violations based on your business needs. Valid values: • Block • Review • Pass
label	String	The label hit.
SubLabel	String	The sub-label.
Score	Integer	The sub-label score. Value range: 0-100
Names	Array of String	The object names.
Details	Array of ObjectDetail	The object detection details.

ObjectDetail

The object detection details. If the scene identified is object, advertising logo, or QR code, this parameter returns the label name, label value, label score, and location information.

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

Location

The location information of the suspicious content.

Parameter	Type	Description
X	Float	The horizontal coordinate of the top-left corner.
Y	Float	The vertical coordinate of the top-left corner.
Width	Float	The width.
Height	Float	The height.
Rotate	Float	The rotation angle of the detection frame.

OcrResult

The OCR result.

Parameter	Type	Description
Scene	String	The scene identified. Default value: `OCR` .
Suggestion	String	The operation suggested by the system for the label with the highest priority. Please handle different types of violations based on your business needs. Valid values: • Block • Review • Pass
label	String	The label hit.
SubLabel	String	The sub-label.
Score	Integer	The confidence score of the sub-label. Value range: 0-100.
Text	String	The text.
Details	Array of OcrTextDetail	The OCR details.

OcrTextDetail

The OCR details.

Parameter	Type	Description
Text	String	The text recognized (up to **5,000 bytes**).
label	String	The label hit.

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

LibResult

The result generated by the block/allowlist library.

Parameter	Type	Description
Scene	String	The scene identified. Default value: `Similar`.
Suggestion	String	The operation suggested by the system. Please handle different types of violations based on your business needs. Returned values: • Block • Review • Pass
label	String	The label hit.
SubLabel	String	The sub-label.
Score	Integer	The confidence score. Value range: 0-100.
Details	Array of LibDetail	The block/allowlist library detection details.

LibDetail

The custom library or block/allowlist library detection details.

Parameter	Type	Description
Id	Integer	The sequence ID.
ImageId	String	The image ID.
label	String	The label hit.
Tag	String	A custom label.
Score	Integer	The confidence score. Value range: 0-100.

Sample callback

```
{
"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. The change only takes effect for new streams. If you want to disable porn detection for an ongoing stream, you need 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：2024-06-25 15:38:07

AOMedia Video 1 (AV1) is a free, open source video encoding 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, AV1-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 template** .
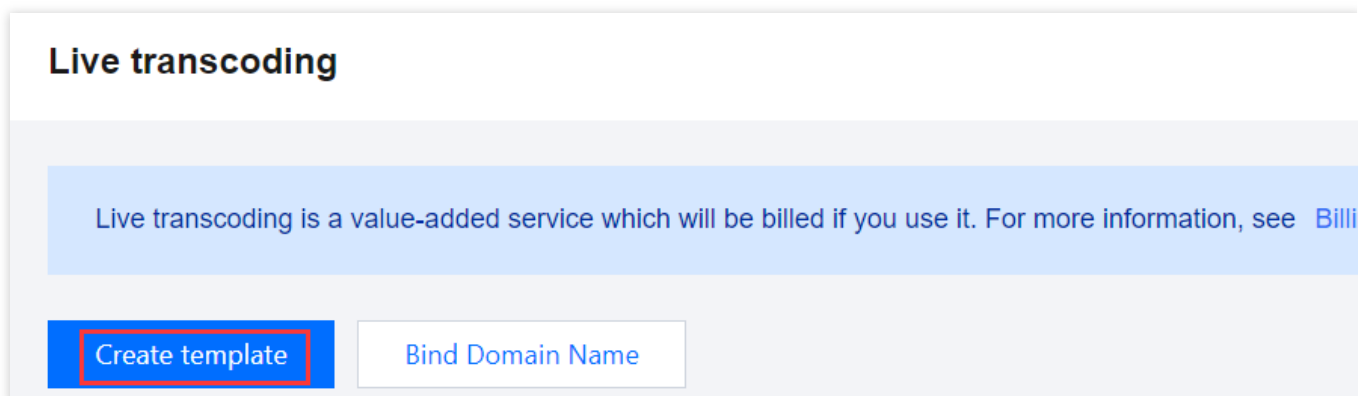
3. Select the transcoding type as either **Standard Transcoding** or **Top Speed Codec Transcoding** , and expand the advanced configuration. In Codec, choose **AV1** .
Standard Transcoding
Top Speed Codec Transcoding

Template Description

> Please describe template

Supports Chinese characters, letters, digits, spaces, and _-

Video quality   Smooth   SD   HD

Video Bitrate *   Specify a bitra ▾   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

Video Frame Rate   Same as sourc ▾

GOP   1-6s   s

The larger the GOP, the higher the delay. The lower the GOP, the more likely lagging may occur

Live subtitles ⬤

Convert speech in live streaming to text in real time. For details, see Live Subtitling ↗ . Using this feature may incur live transcoding fees (charged by CSS), speech recognition fees, and speech translation fees (charged by MPS). Learn more ↗

Loudness   -40~-10   LKFS

The loudness. The larger the value, the louder.

Parameter Limit   The output video height cannot exceed the original height ⓘ ⬤

The output frame rate cannot exceed the original frame rate ⓘ ⬤

The output bitrate cannot exceed the original bitrate ⓘ ⬤

Advanced Configuration
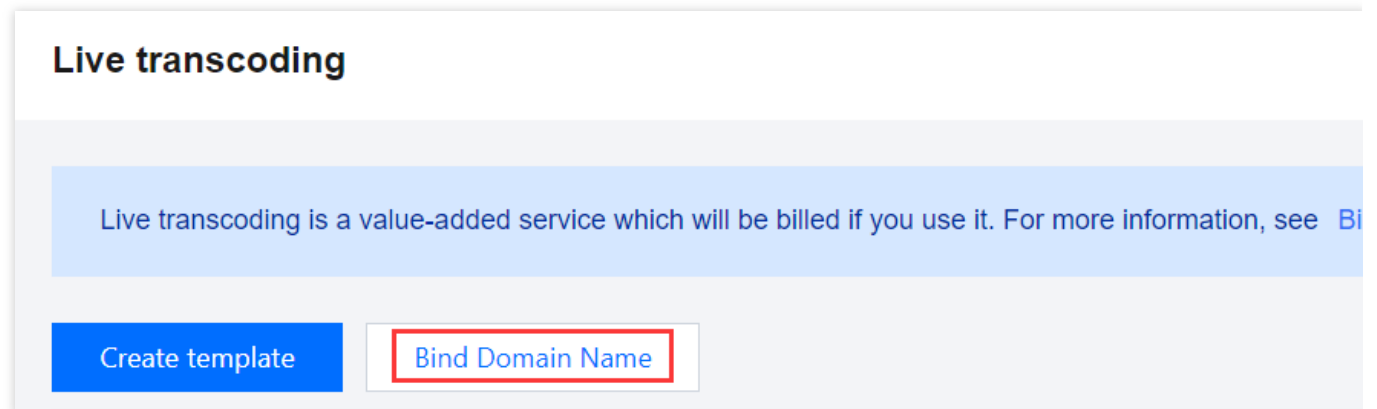
Advanced Configuration ▲

[Save] [Cancel]

Transcoding Configuration

Transcoding Type	Standard Transcoding	Top Speed Codec Transcoding	Audio-only Transcoding

Template Name *

[2-10 characters]

Supports letters and numbers; cannot contain only numbers or be identical to the name of an existing transcoding template or stream.

Template Description

[Please describe template]

Supports Chinese characters, letters, digits, spaces, and _-

Video quality　　　Smooth　SD　HD

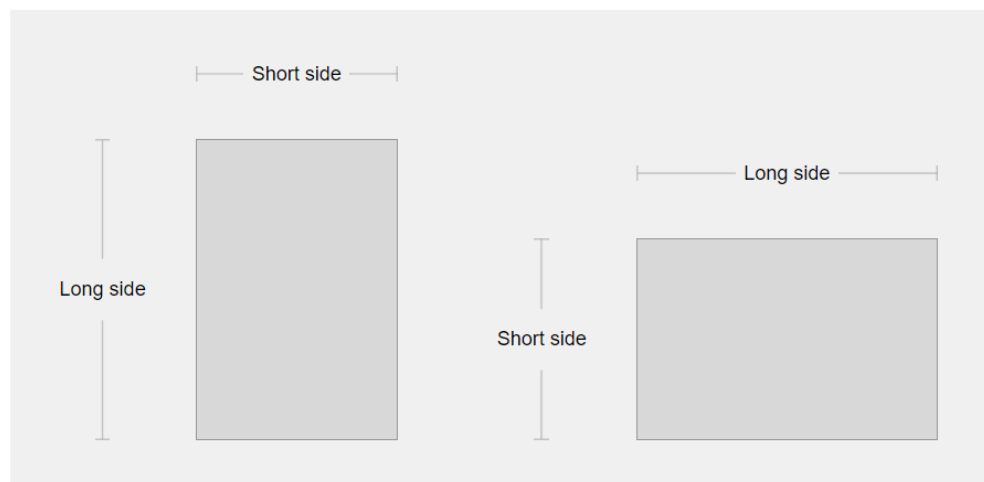Video Bitrate *　[Specify a bitra ▾]　[101-8000]　kbps

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

Video Frame Rate　[Same as sourc ▾]

GOP　　　　[1-6s]　s

The larger the GOP, the higher the delay. The lower the GOP, the more likely lagging may occur

4. Click **Save**.

Step 2. Bind a domain

1. Select a transcoding template, and click **Bind Domain Name** .

2. Select the **Transcoding Template** and **Playback Domain** you wish to bind, and then click **Confirm** .

Note:

The domain name binding will take effect in approximately 10 minutes after being bound.

Step 3. Generate a playback URL

1. Log in to the CSS console > Go to **CSS Toolkit** > Address Generator.

2. Select URL Type: Playback Address.

3. Select the playback domain name bound in Step 2 and the transcoding template from Step 1 to generate the playback address.

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, you can refer to AV1 Video Playback to rebuild your player accordingly.

Stream Mix

Last updated：2024-03-13 21:16:03

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.

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, 310,410, 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 Preview Image	Template 30 Preview Image

Template 40 Preview Image
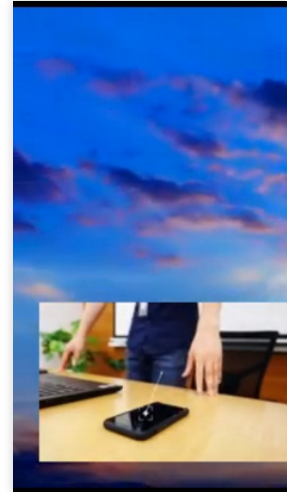
Template 310 Preview Image
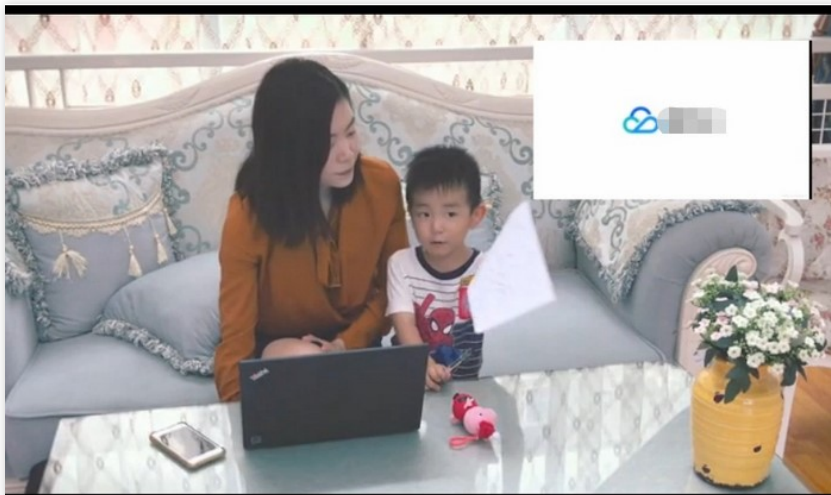
Template 410 Preview Image
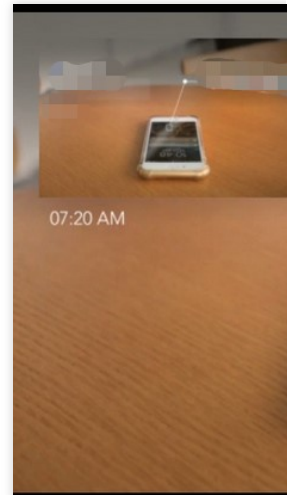
Template 510 Preview Image
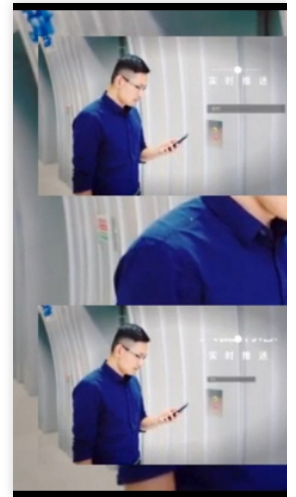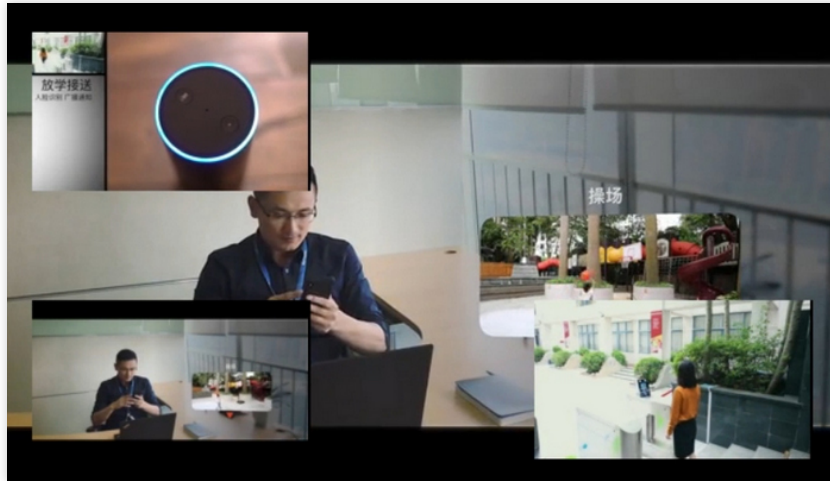
Template 610 Preview Image

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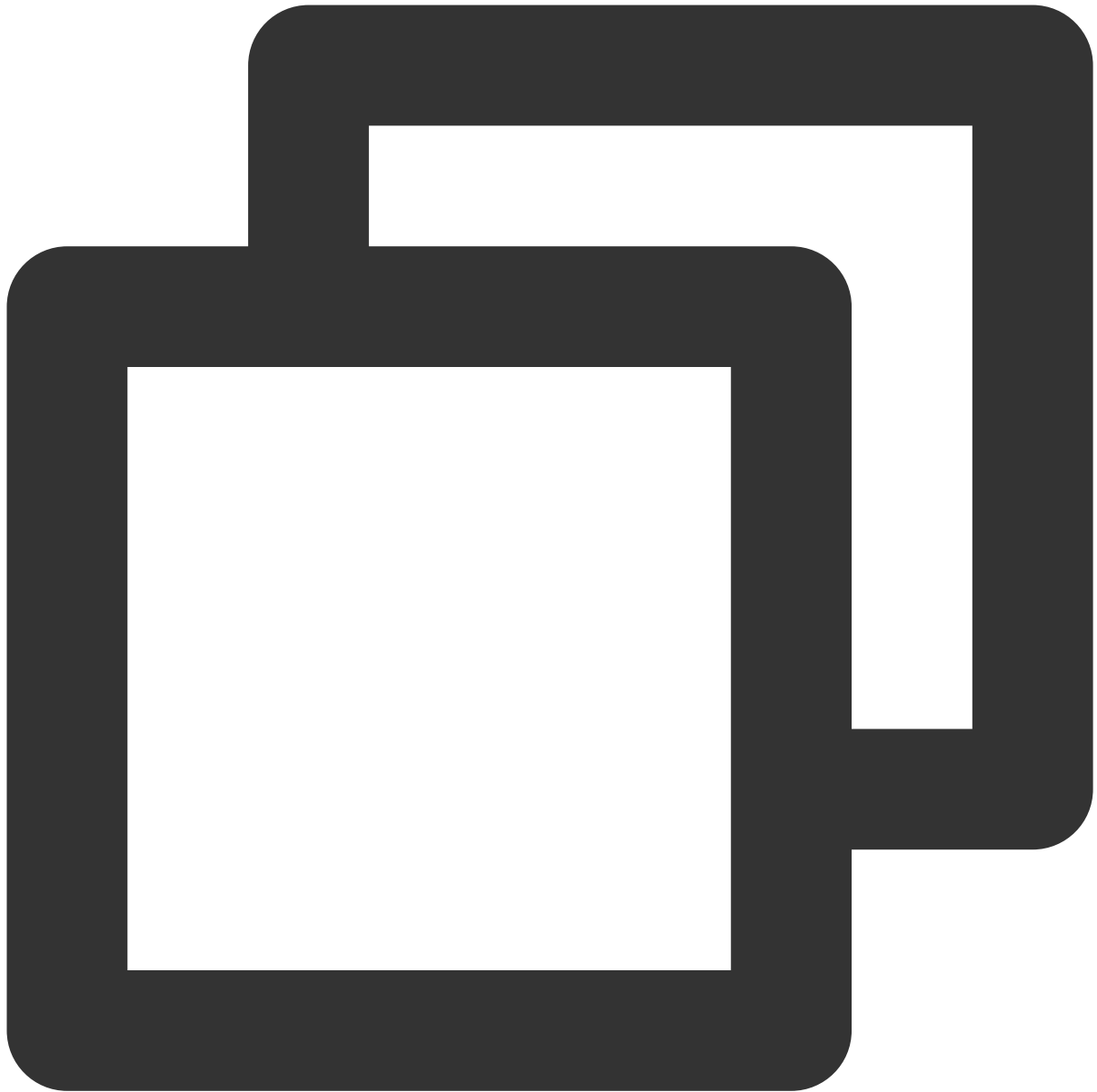
```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&MixStreamTemplateId=20
&OutputParams.OutputStreamName=test_stream1
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.1.InputStreamName=test_stream2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

Sample output code
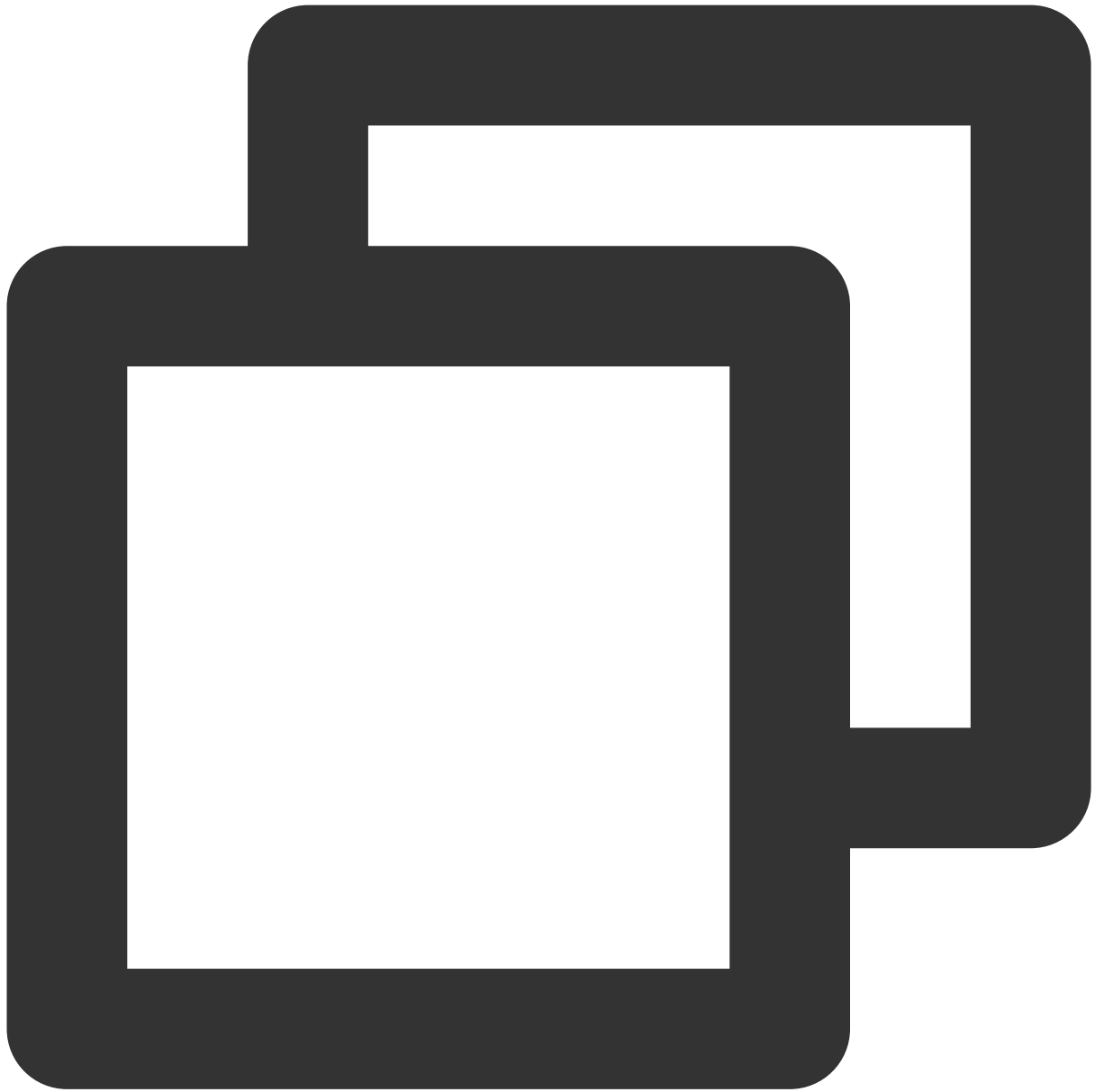

```
{
  "Response": {
    "RequestId": "e8fa8015-0892-40d5-95c4-12a4bc06ed31"
  }
}
```

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
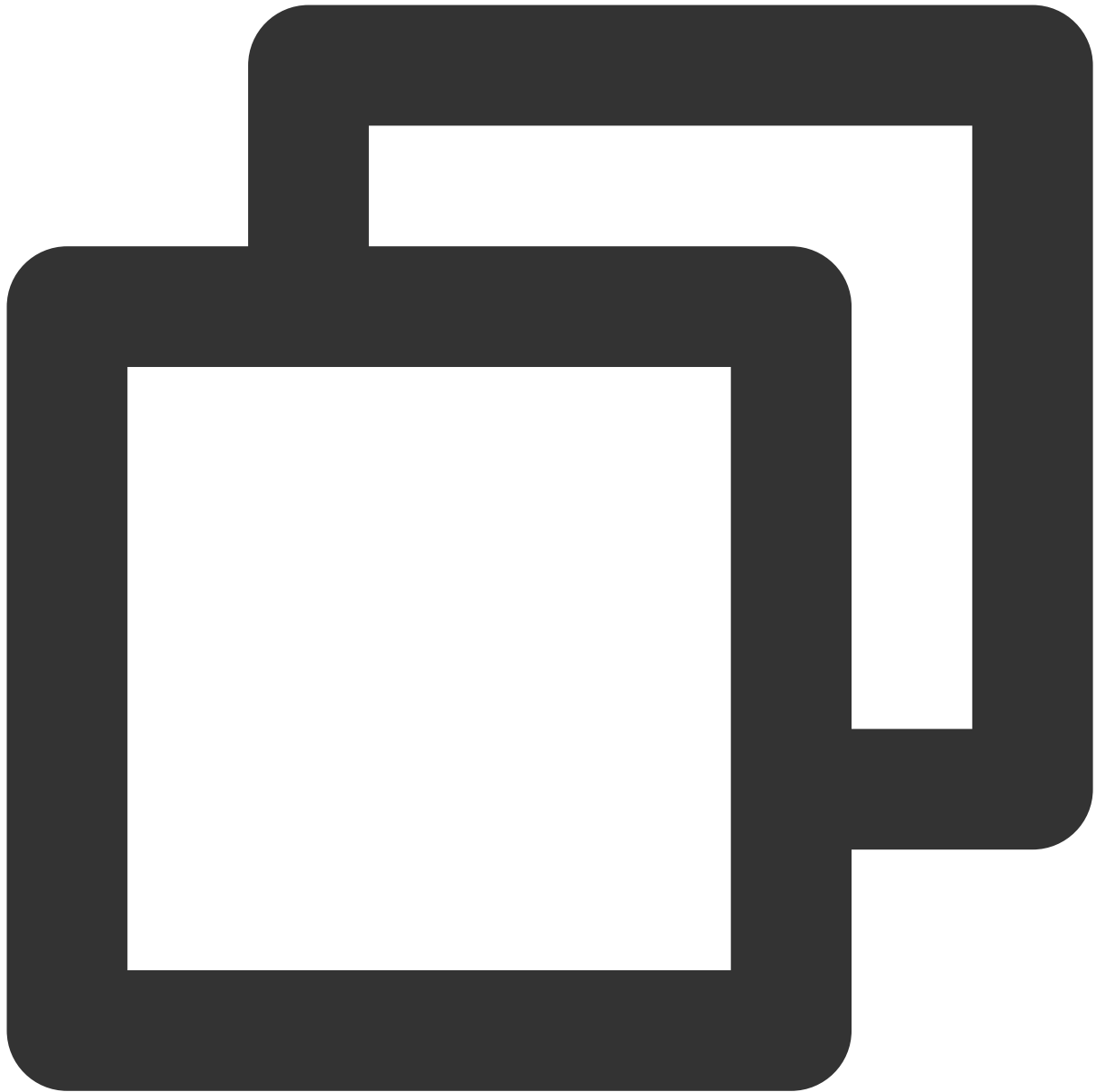
```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&MixStreamTemplateId=390
&OutputParams.OutputStreamName=test_stream2
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.0.LayoutParams.InputType=3
&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
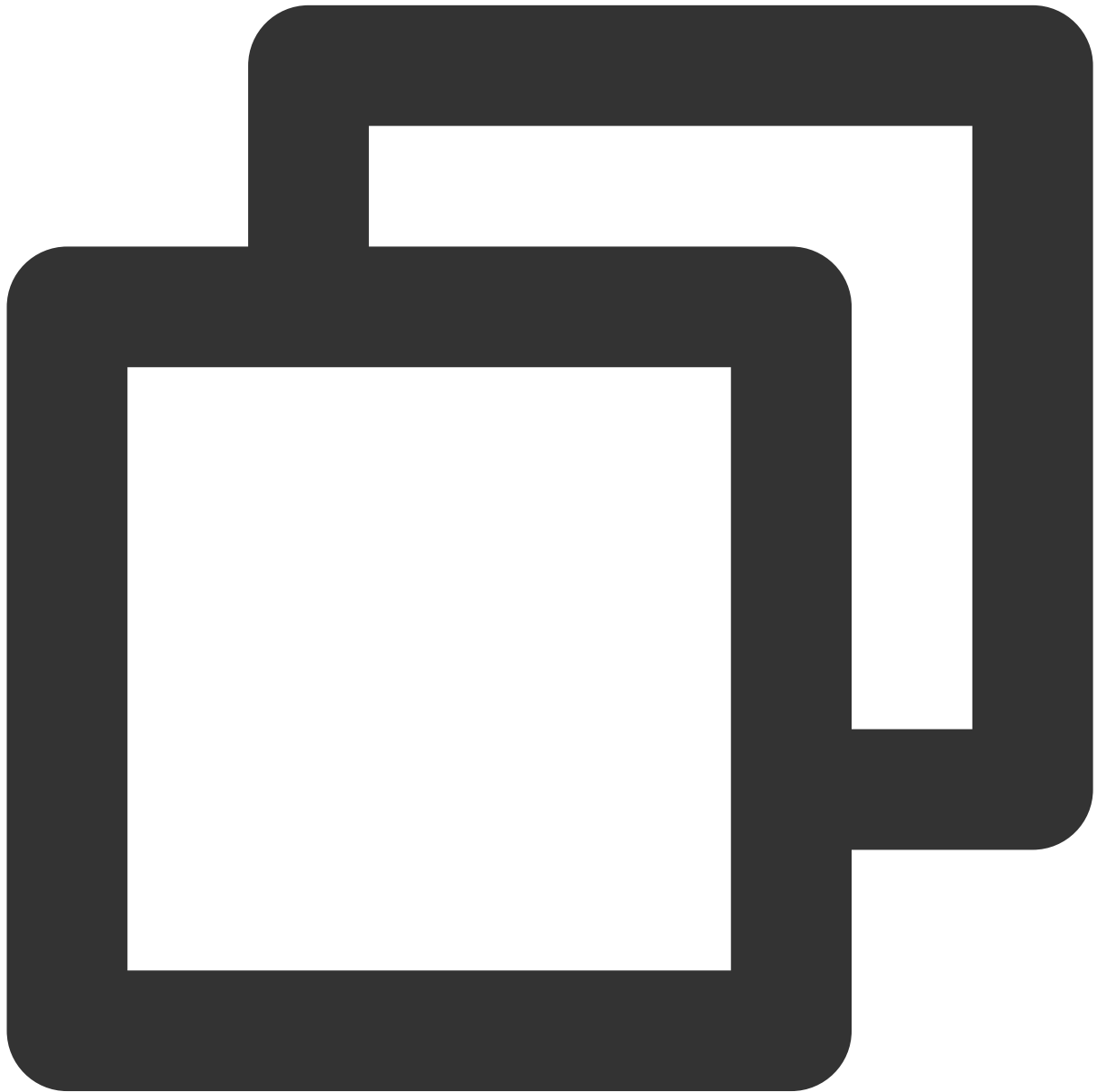
```
{
  "Response": {
    "RequestId": "9d8d5837-2273-4936-8661-781aeab9bc9c"
  }
```

```
    }
```

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
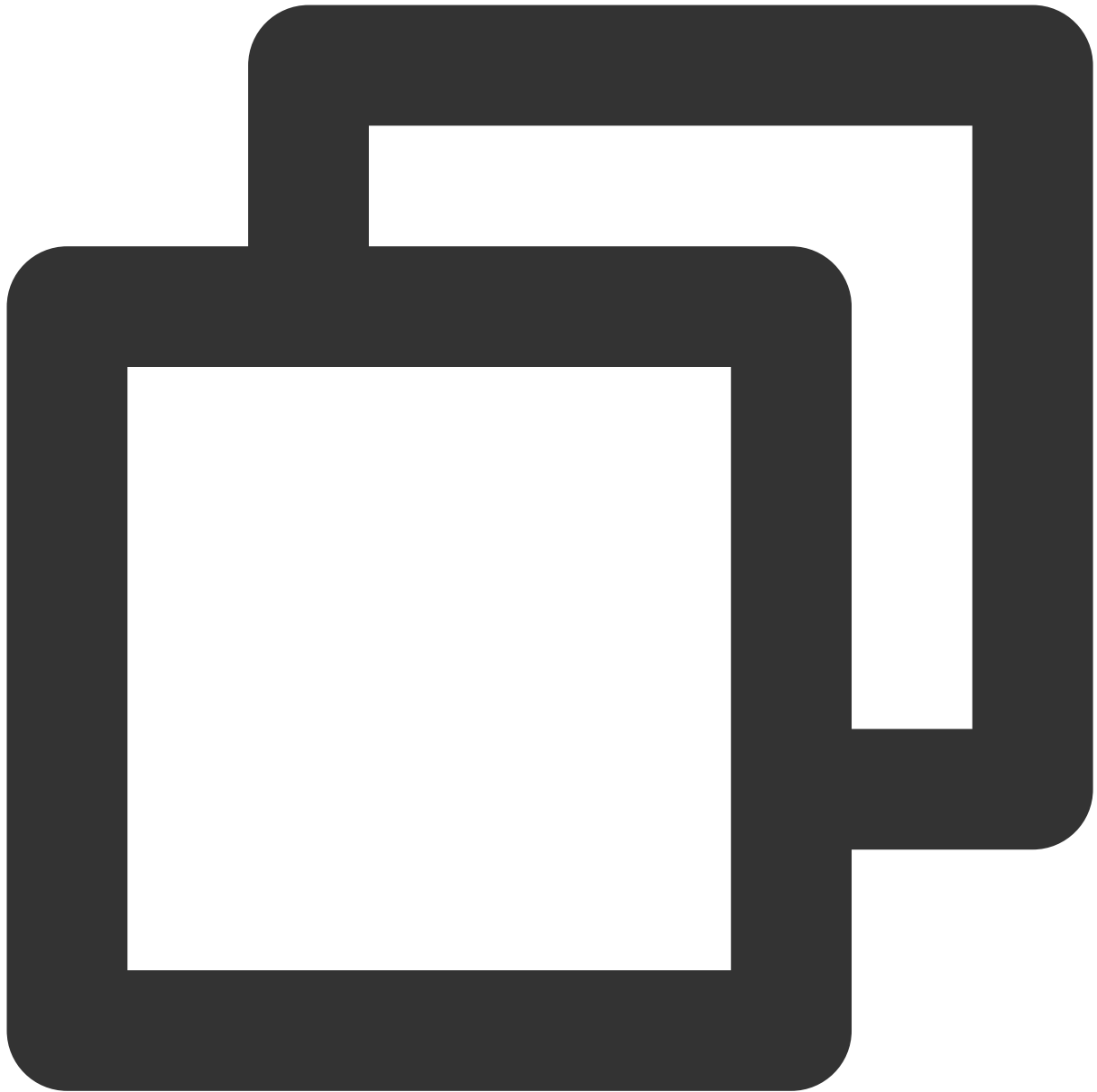
```
https://live.tencentcloudapi.com/?Action=CreateCommonMixStream
&MixStreamSessionId=test_room
&OutputParams.OutputStreamName=test_stream2
&InputStreamList.0.InputStreamName=test_stream1
&InputStreamList.0.LayoutParams.ImageLayer=1
&InputStreamList.0.LayoutParams.InputType=3
&InputStreamList.0.LayoutParams.ImageWidth = 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
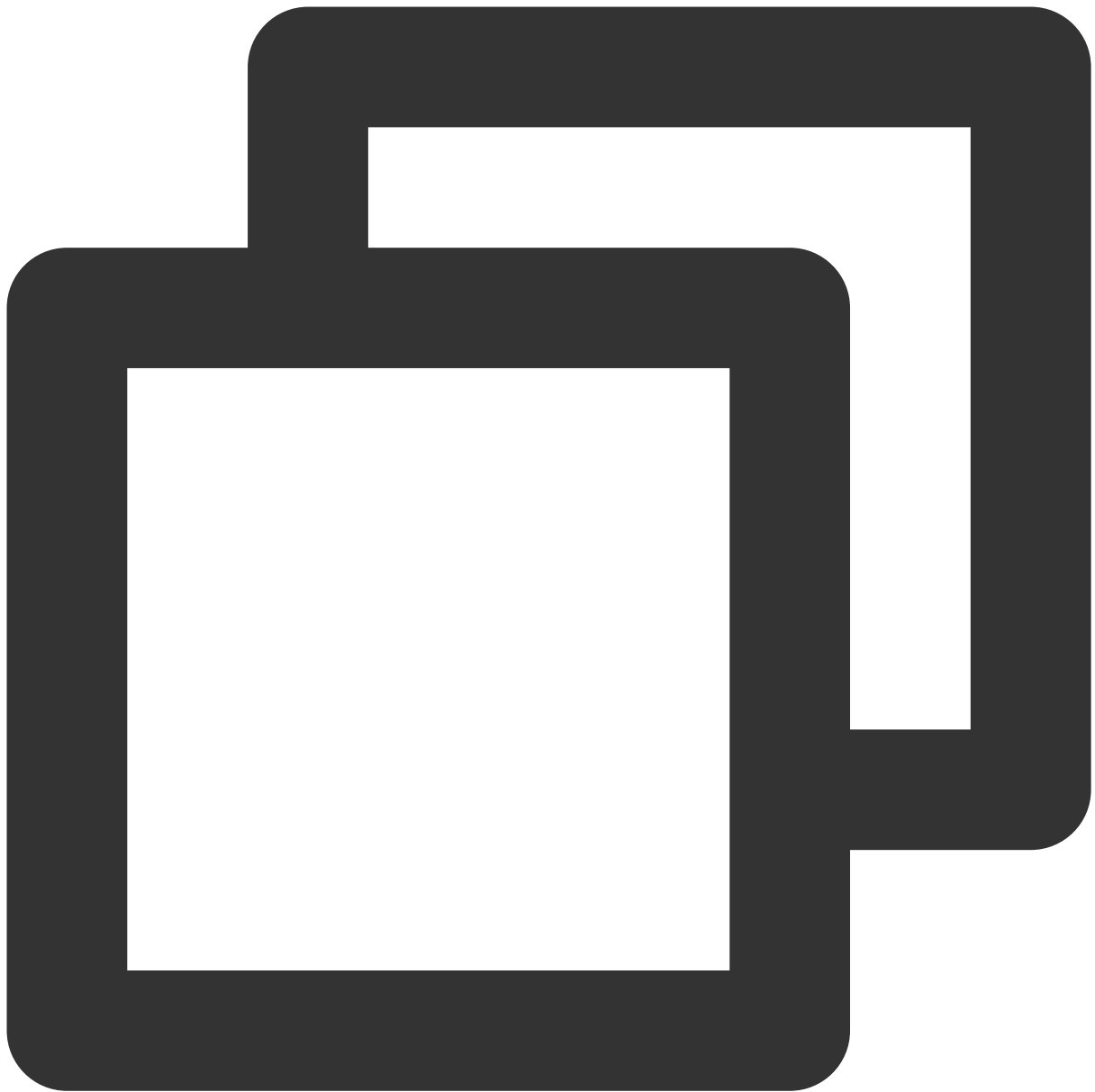
```
{
  "Response": {
    "RequestId": "8c443359-ba07-4b81-add8-a6ff54f9bf54"
  }
}
```

Custom stream mix effect
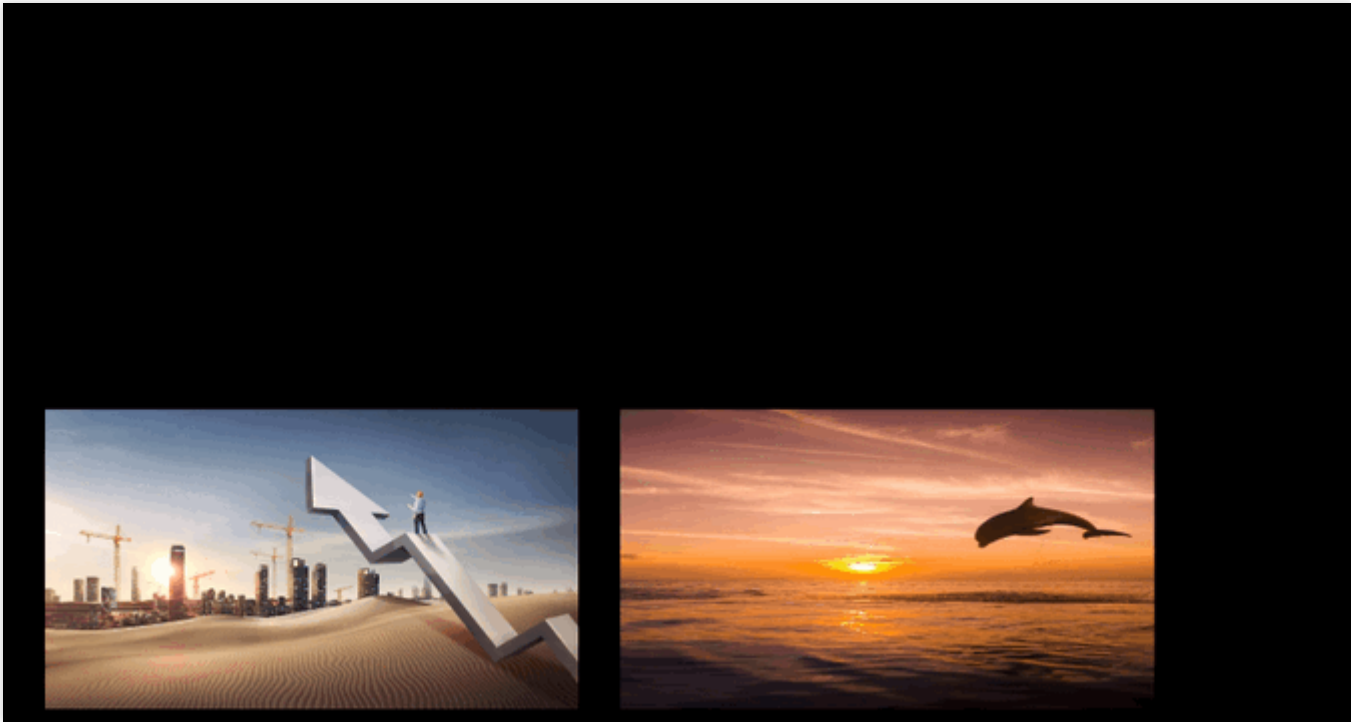
Canceling Stream Mix

Parameters

For more information, please see CancelCommonMixStream.

Examples

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

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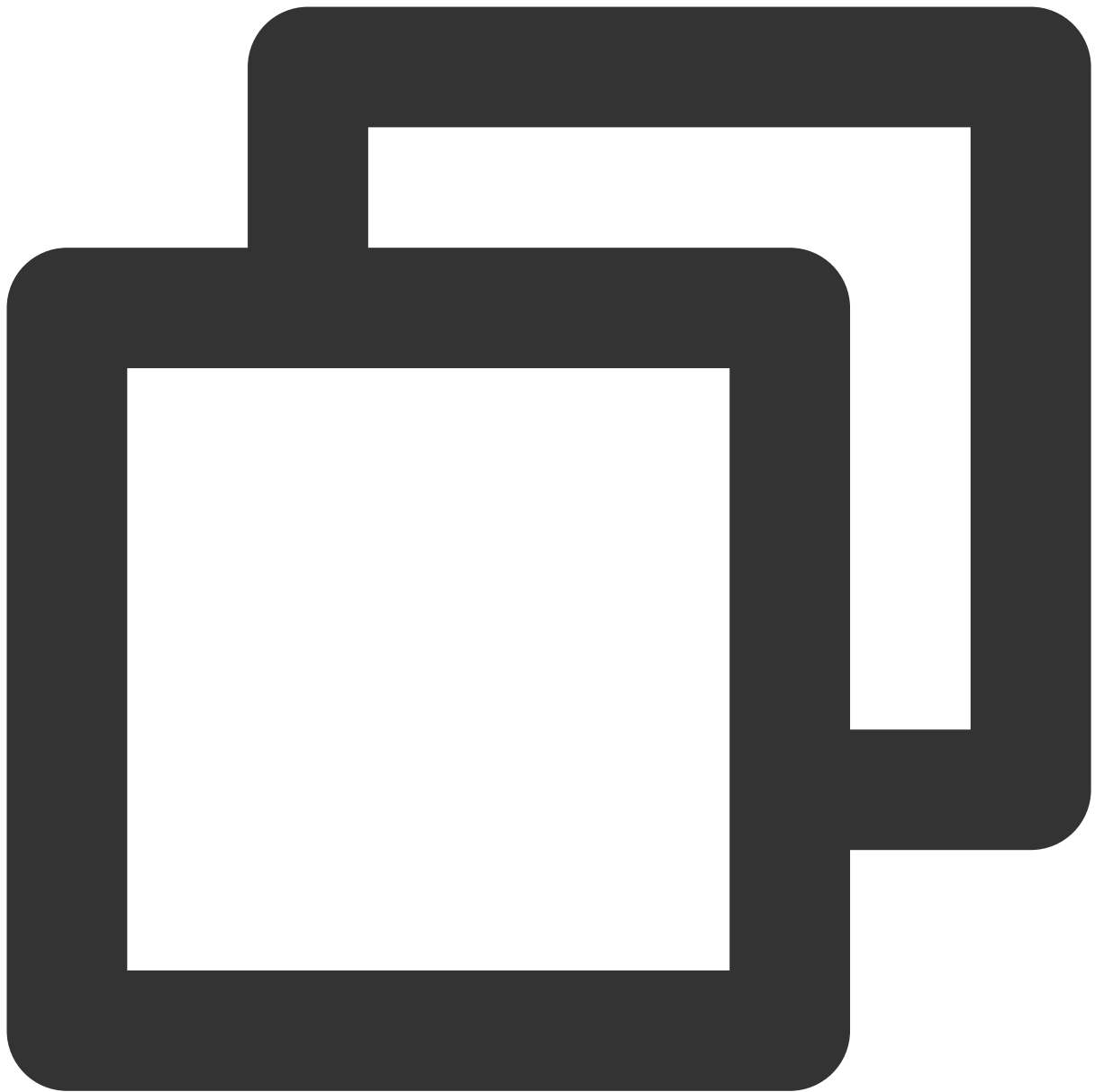
```
https://live.tencentcloudapi.com/?Action=CancelCommonMixStream
&MixStreamSessionId=test_room
```

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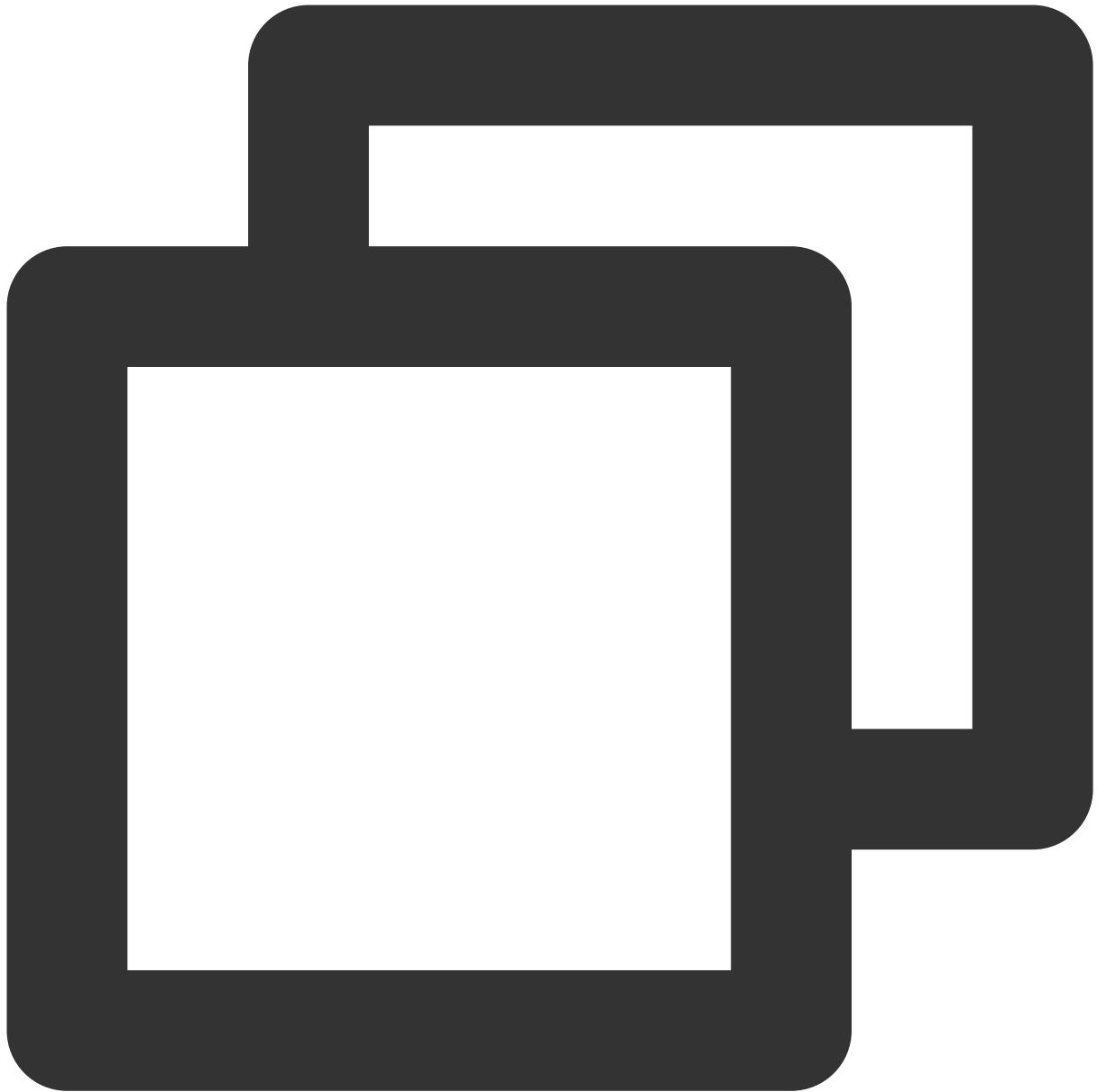
```
{
  "Response": {
    "RequestId": "3c140219-cfe9-470e-b241-907877d6fb03"
  }
}
```

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	Check whether the JSON format of the request body is correct. Check whether `InputStreamList` 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	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 `LayoutParams`. Check whether `InputType` 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	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).
-20	The input parameter does not match the API	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	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`	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	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.
-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.

ROI Intelligent Recognition

Last updated：2024-07-08 15:41:31

ROI (region of interest) recognition can identify the positions of important visual elements in a video in real time, such as faces, game characters, or steaming hosts, and send this information along with the video to the playback device. Using the ROI information, the player can do things like blur the background in a scene and prevent on-screen comments from covering important elements of the video.

Prerequisites

You have activated Tencent Cloud Streaming Services and added a push domain.

Instructions

Service Side

After configuring ROI recognition in the console, when the user pulls a stream containing on-screen comments, the backend will trigger the recognition capability of MPS (Media Processing Service). During the transcoding process, the system will get the recognition results in real time, generate SEI (Supplemental Enhancement Information) data according to the protocol, and write it into the stream (currently, only SEI output of H.264 and H.265 formats is supported).

User Side

When the user's video player accesses the live stream, it first parses the SEI data. Then, it decodes the SEI information according to the specific protocol and extracts the SVG data. Finally, by using SVG images and masking techniques, the player can process the ROI information, which allows it to accurately locate and process specific areas in the video.

SEI Parsing Related Information

1. Tencent Cloud SEI Format

The image below shows the standard SEI format we adopt.
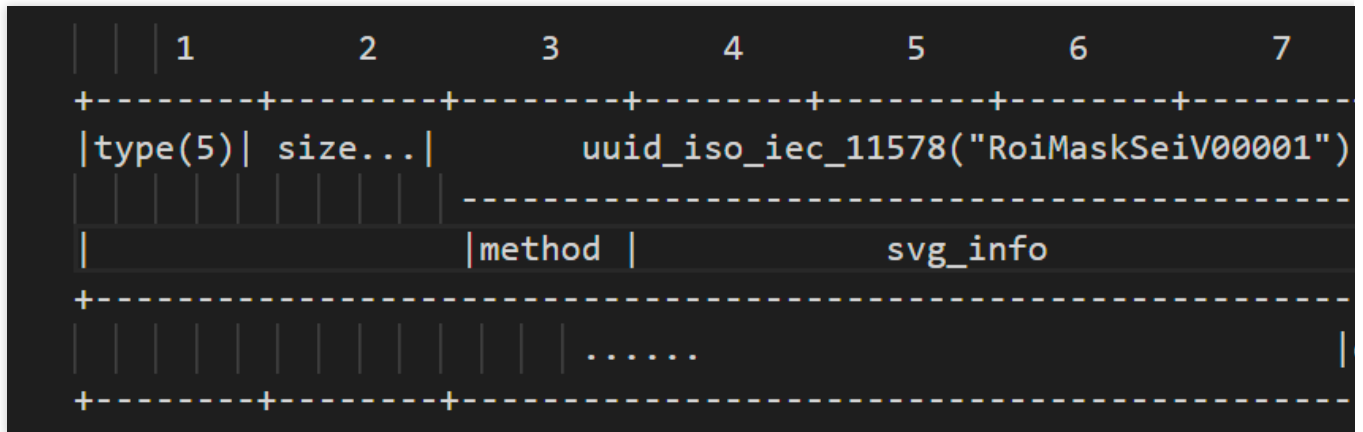
Note:

The size field is variable in length and complies with the H.264 SEI standard. It does not include the 0x80 end byte, but includes a method field (1 byte) and UUID field (16 bytes). The svg_info represents encoded SVG information.

The method field indicates the data storage method, with the following values:

1: Uncompressed

2: Bzip2 Compression

3: Zip Compression

When processing SEI data, we use unregistered user data as the SEI frame type (Type value is 5). This type of SEI frame is used to carry custom data, such as SVG information, for parsing and processing on the player side.

When the SEI content contains 0x000000 or 0x000001, it is necessary to insert 0x03 for escape sequence handling. This is because, in the H.264 standard, consecutive 0x000000 or 0x000001 sequences are considered NAL unit delimiters. Therefore, inserting 0x03 prevents misinterpretation. During decoding, the decoder detects the 0x00 00 03 sequence within the NAL unit and discards the 0x03, thereby restoring the original data.

In processing SEI data, the following conversion rules should be noted:

0x00 00 00 converts to 0x00 00 03 00

0x00 00 01 converts to 0x00 00 03 01

0x00 00 02 converts to 0x00 00 03 02 (0x00 00 02 is reserved for future use)

0x00 00 03 converts to 0x00 00 03 03 (During decoding, only filter once, do not loop filter)

For H.265 SEI support, H.265 SEI NALU type, 39 NAL_UNIT_SEI sei_rbsp() sei payload, that is, NAL_UNIT_SEI (type value 39) will be inserted. The payload of SEI is processed in the same way as in H.264, but the startcode of SEI needs to be consistent with the H.265 standard.

SVG Extraction

1. By parsing SEI data according to the protocol and extracting SVG information, you will be able to obtain a Base64 string in a format similar to the following:

get sei(uuid:RoiMaskSeiV00001) from pkt(pts:152367, dts:152334):
PHN2ZyB2aWV3Qm94PSIwLCAwLCAyMjcsIDEyOCIgcHJlc2VydmVBc3BlY3RSYXRpbz0ieE1pZFlNaWQgbWVldCI
tbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyI+PGRlZnM+PGZpbHRlciBpZD0iZjEiIHg9IjAiIHk9Ij
FuQmx1ciBpbj0iU291cmNlR3JhcGhpYyIgc3RkRGV2aWF0aW9uPSIzIi8+PC9maWx0ZXI+PC9kZWZzPjxnIGZpb
SkiIGZpbGw9IiMwMDAiIHN0cm9rZT0ibm9uZSI+PHBhdGggZmlsbD0iydWxlPSJldmVub2RkIiBkPSJNMTcyLDAg
MTc2LDYgMTc3LDkgMTc4LDEyIDE3OSwxNyAxODAsMjUgMTgwLDM5IDE4MSw0NCAxODIsNDkgMTgzLDUzIDE4NCw
zLDcyIDE5NSw3NCAxOTksNzgggMTk5LDgwIDIwMiw4MyAyMDIsODUgMjAzLDg3IDIwNiw5MSAyMTAsOTUgMjExLD
E3LDEwNCAyMTgsMTA4IDIxNywxMjAgMjIyLDEyNCAyMjIsMTI3IDIyNiwxMjcgMjI2LDBNCwwIDAsMTI3IDYzL
jUsMTIwIDY0LDEwOSA2NCwxMDEgNjUsOTggNzAsODkgNzAsODcgNzEsODQgNzYsNzggNzgsNzUgODEgODEsNzE
NjUgODcsNjIgODksNjAgOTIsNTQgOTIsNTIgOTIgOTQsNDkgOTYsNDYgOTYsNDMgOTYsNDMgOTYsNDMgMTA
xMDEsMjIgMTAyLDIwIDEwMywxNiAxMDUsMTMgMTA1LDExIDEwNyw4IDEwNyw2IDEwOSwyIDEwMCwwICIvPjwvZz

2. Place the extracted Base64 encoded SVG image data (data:image/svg+xml;base64,svg) in the browser's address bar and press Enter to directly view the related image information.
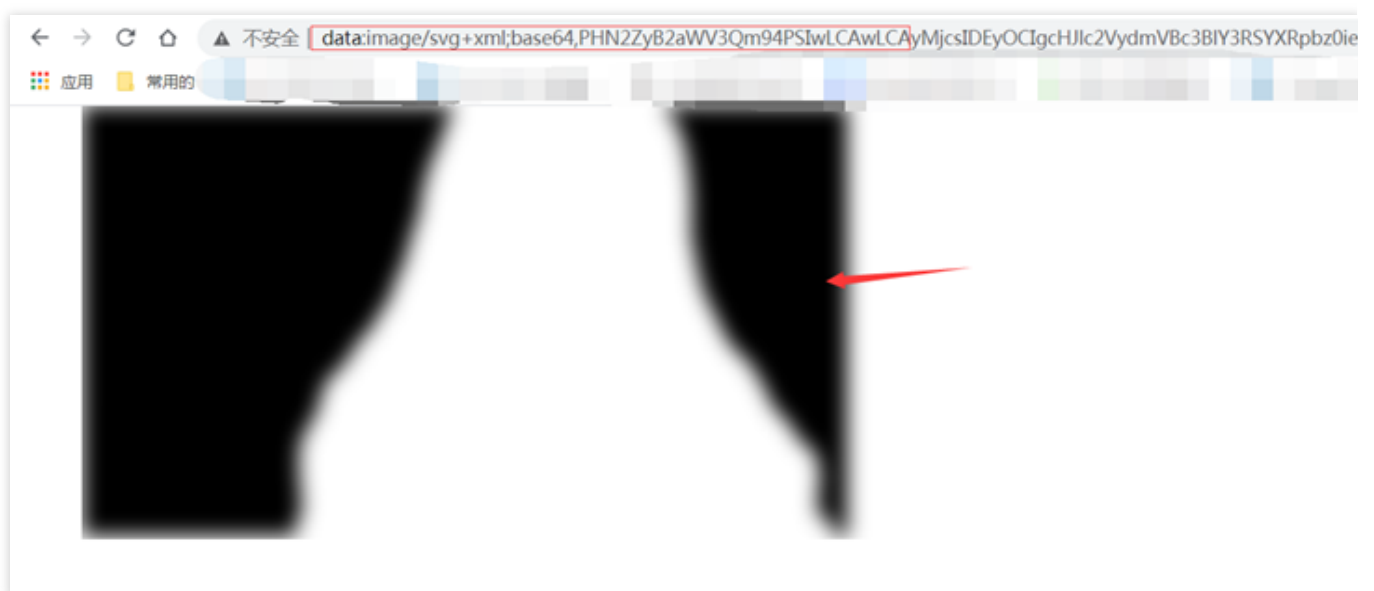
Live Streaming Security
Video Content Protection

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, `Protocol:\\Playback domain\AppName\Stream ID`. 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.
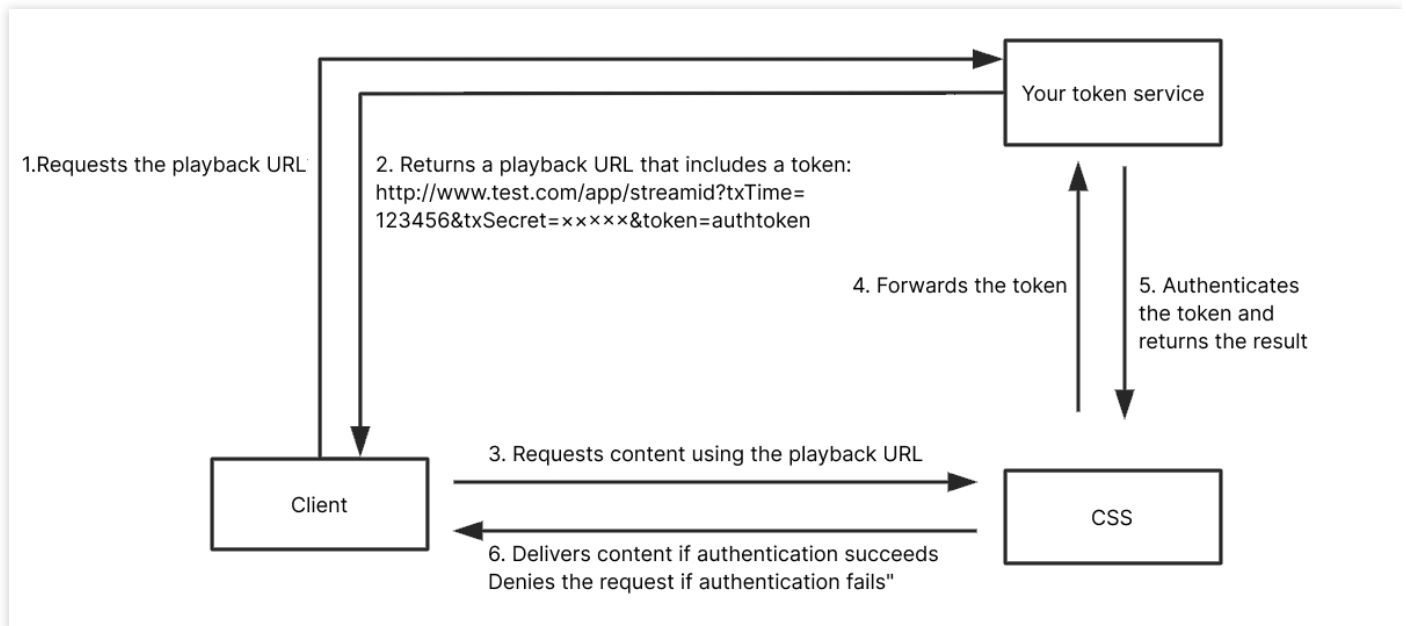
i. A request is sent to your server (token service) for the playback URL.

ii. 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.

iii. Viewers will use the playback URL to request live content from CSS.

iv. CSS will authenticate `txSecret` and forward the request, which includes the token, to your token service.

v. 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.

vi. 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.
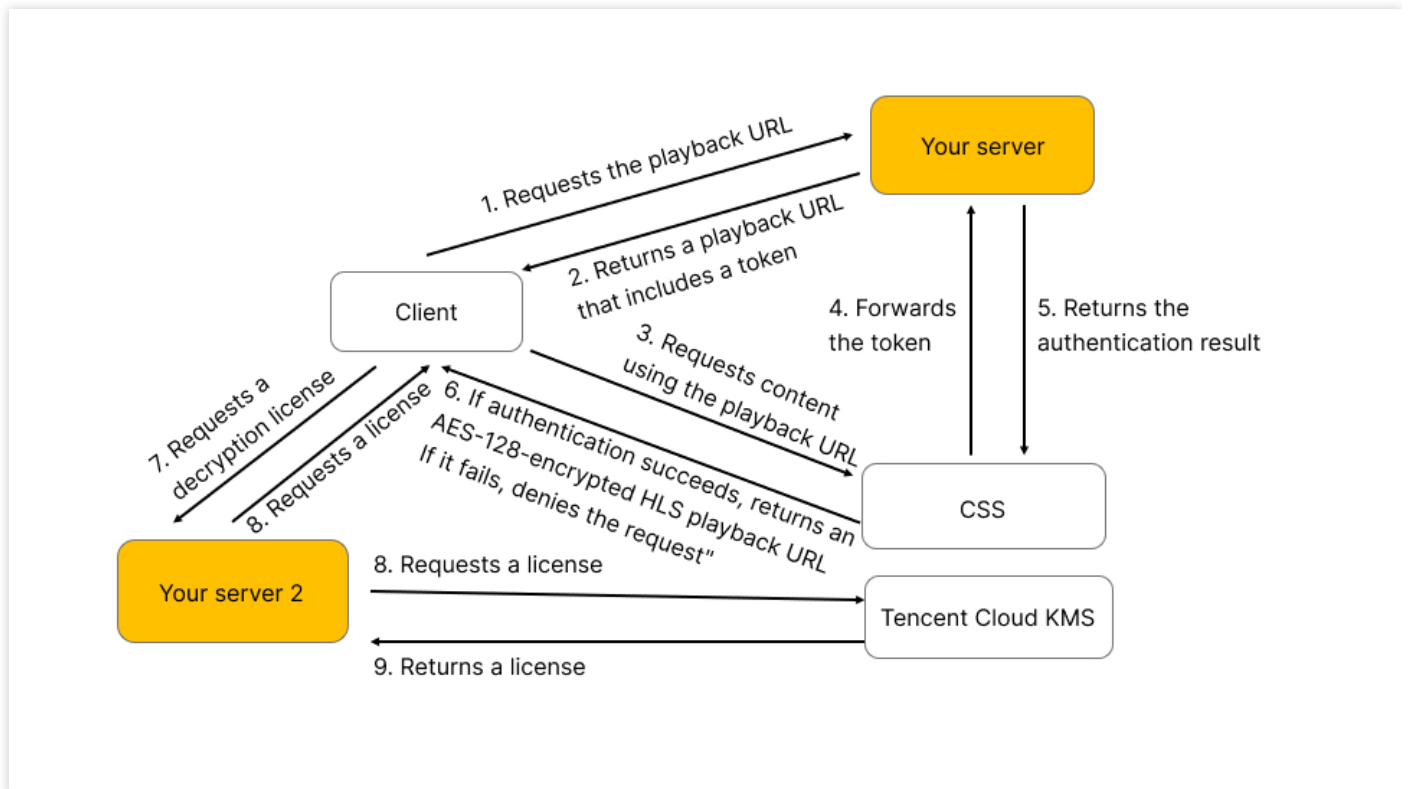
- **Implementation**: Enable key authentication and remote authentication. For detailed directions, please contact sales or submit a ticket.
- **Pros**: This solution is easy to implement. The combination of key and remote token authentication offers extra protection. What's more, because AES-128 is 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:
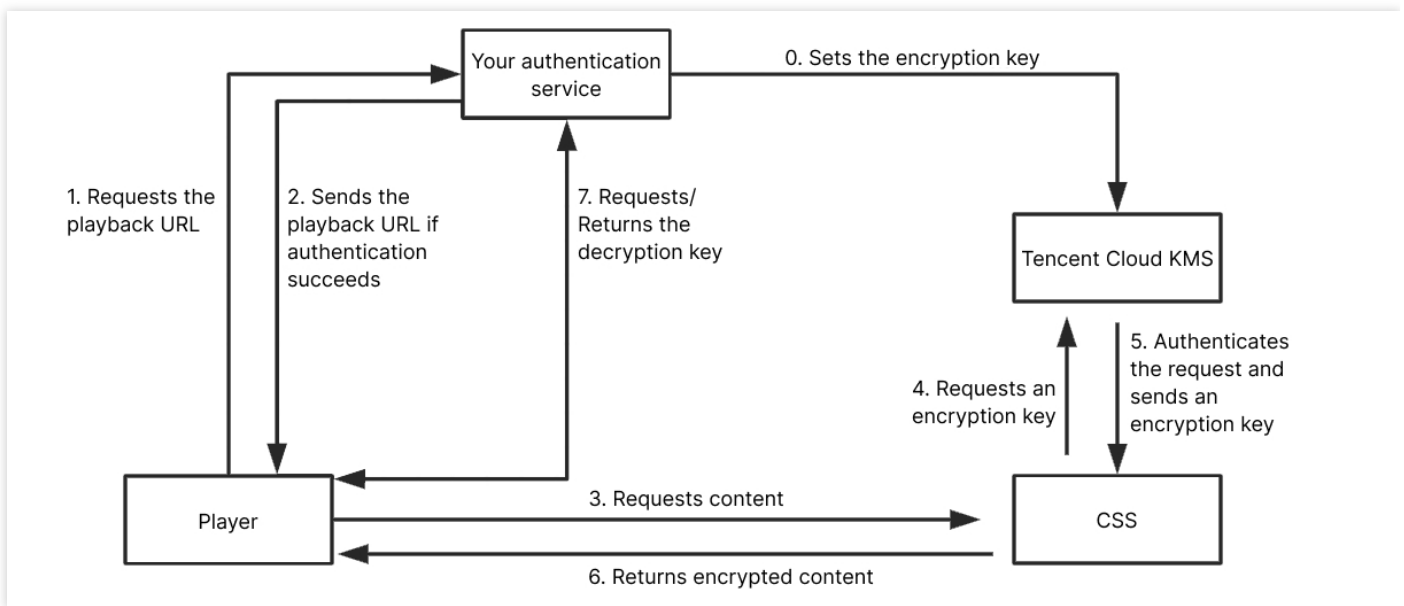
- **Implementation**: For detailed directions, please contact sales or submit a ticket.
- **Pros**: You have control over the entire process, and there are key management services and encryption/decryption tools to support this solution. Tencent Cloud's Player SDK is easy to integrate.
- **Cons**: You 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	⭐⭐⭐

Live FLV Encryption

Last updated：2023-10-07 17:25:41

Most private live streaming or live streaming that requires content security does not require hardware-level security and complex certificate distribution and verification processes. Moreover, in domestic live streaming, the FLV live streaming method is also popular. A secure live streaming solution for FLV is needed.

Use case: When using the FLV protocol for playback, it is desired to encrypt the stream content so that hackers cannot capture it through the network, and even if the stream is dumped locally, it cannot be played.

Implementation plan: Tencent Cloud CSS has developed its own stream encryption solution. Customers can request FLV encryption by submitting a ticket, specifying the encryption mode (video encryption, audio and video encryption), and Tencent Cloud will encrypt the live stream according to the specified module. When decrypting and playing, customers can obtain the TXEncryptionToken key field through the Tencent Cloud API interface DescribeDRMLicense request, add it to the playback URL parameters, and provide it to the playback SDK for decryption and playback.

The self-developed encryption and decryption process is as follows:
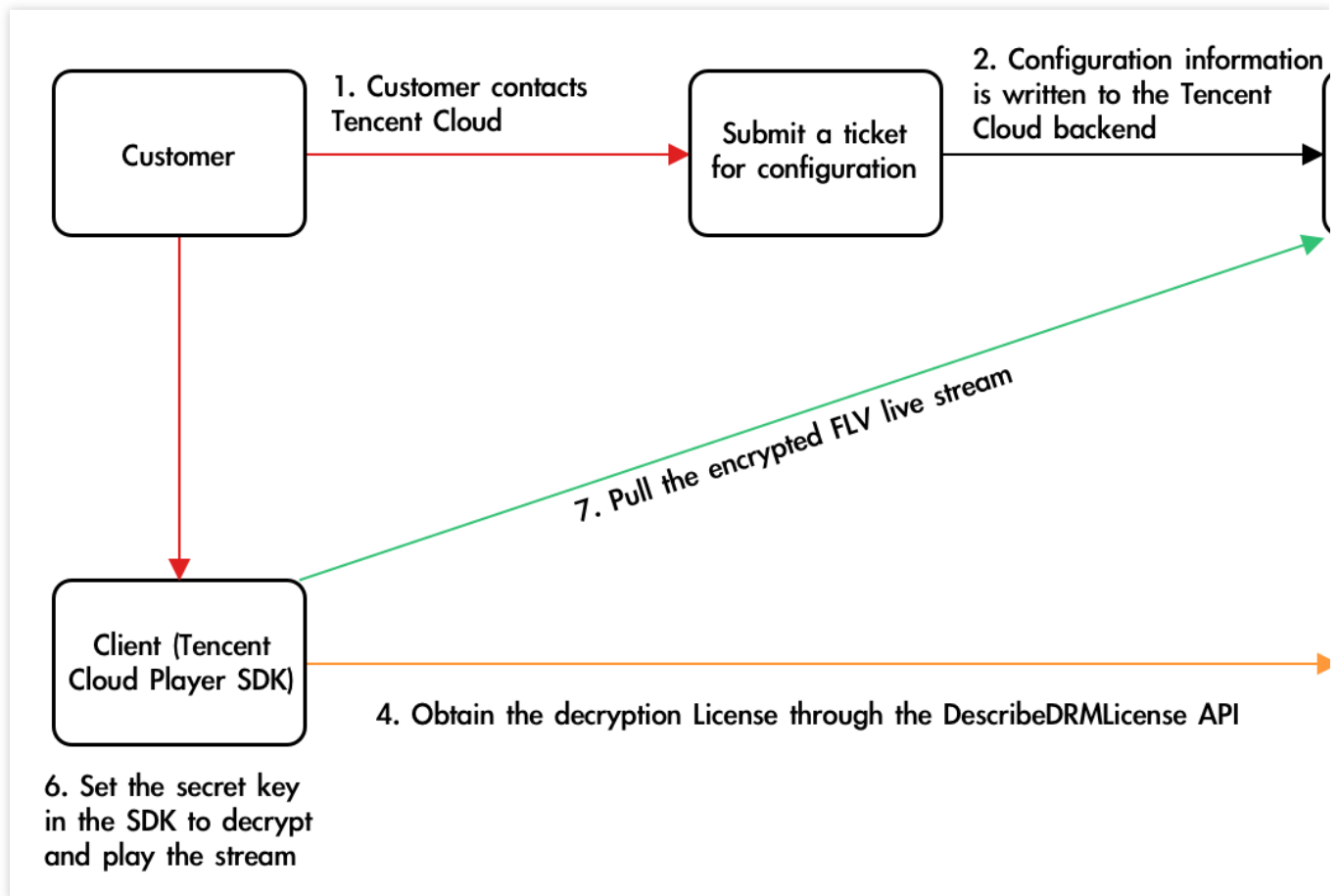
Implementation Method: For the specific implementation process, please contact Tencent Cloud sales or submit a ticket to contact Tencent Cloud CSS.

Advantages of the solution: The entire process is controllable, with product and tool support for keys and encryption/decryption. Tencent Cloud provides Player SDK, which is easy to integrate and has a mature solution.

Existing issues: The need to integrate the SDK, only supports custom-developed players. Web and browsers cannot play.

This solution provides two access methods for iOS and Android. Click here to download the SDK.

iOS Integration
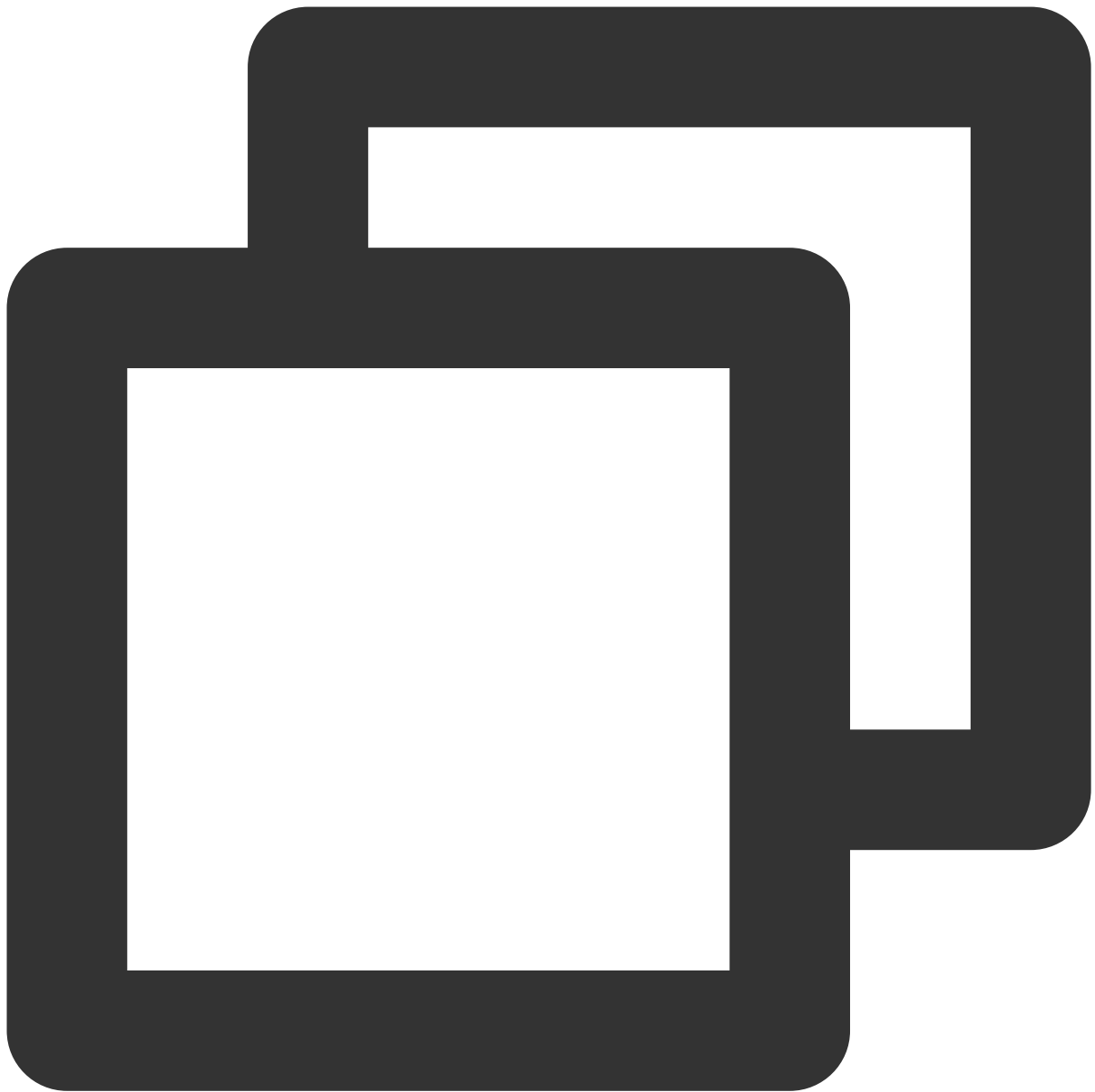
```
/**
Create a Player Instance.
 */
V2TXLivePlayer *player = [[V2TXLivePlayer alloc] init];


/**
 * Set the video rendering View for the player. This control is responsible for dis

 * @param view Player Rendering View
 * @return Return Value {@link V2TXLiveCode}
```

```
 *              - V2TXLIVE_OK：Success
 */
[player setRenderView:view];



/**
 * Set the player callback.
 *
 * By setting the callback, you can listen to some callback events of the V2TXLiveP
 * This includes player status, playback volume callback, audio and video first fra
 * @param observer The target object for the player's callback. For more informatio
 */
[player setObserver:self];



/**
 * For key requests, please refer to License acquisition.
 * Set the Key
 *
 * @note The URL in the JSON must be the same as the URL in startLivePlay. The SDK
 */
NSString *url = @"http://5000.liveplay.myqcloud.com/live/flvtest100_1000.flv?reques
";



/**
 * Start playing the audio and video stream.
 *
 * @param url The playback address of the audio and video stream, supporting RTMP,
 * @return Return Value {@link V2TXLiveCode}
 *              - V2TXLIVE_OK: Operation successful, start connecting and playing
 *              - V2TXLIVE_ERROR_INVALID_PARAMETER: Operation failed, the URL is not val
 *              - V2TXLIVE_ERROR_REFUSED: RTC does not support pushing and pulling the s
[player startLivePlay:url];
```

Android Integration
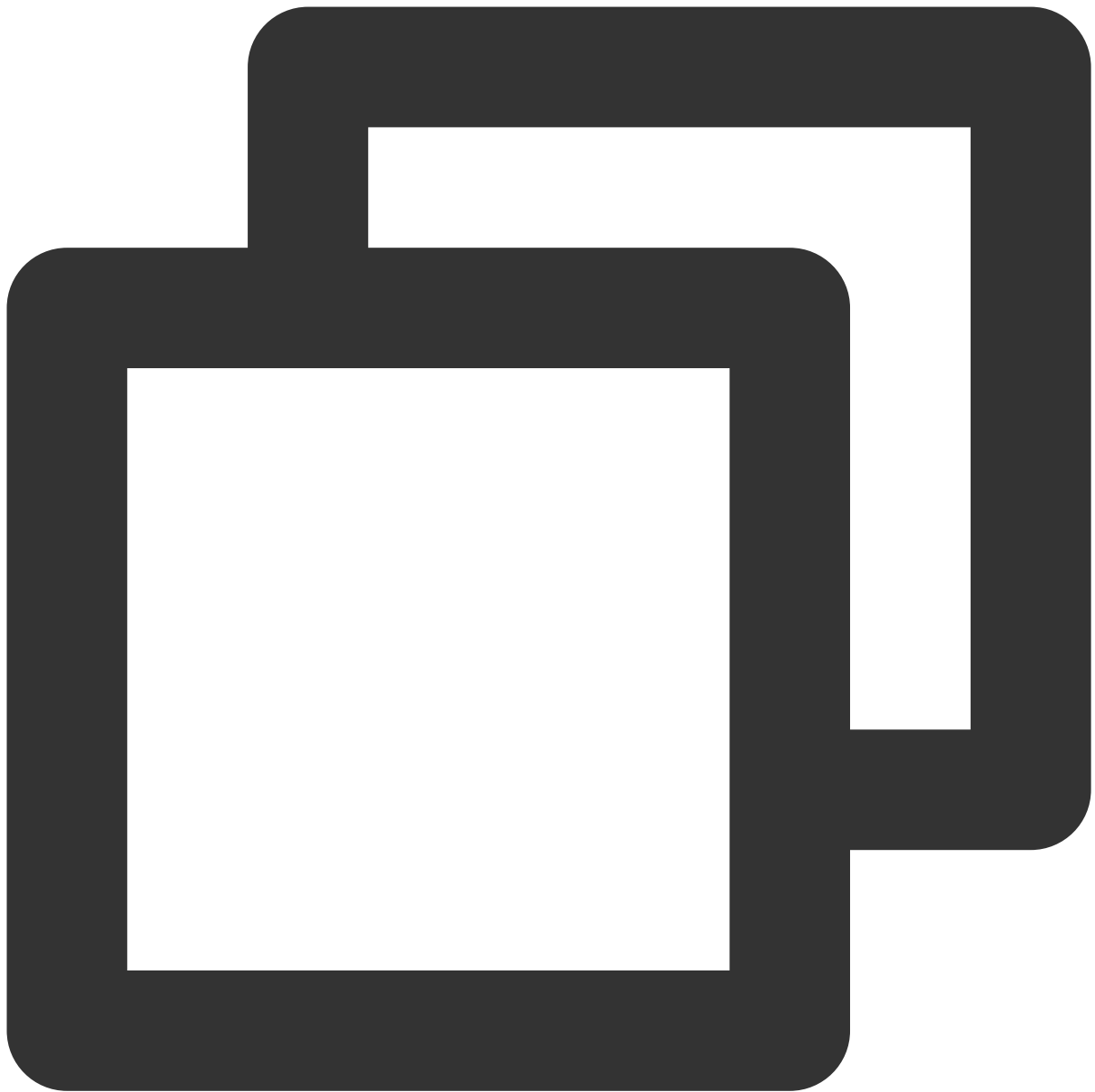
```
/**
 * Create a Player Instance.
 */
V2TXLivePlayer player = new V2TXLivePlayer();
```

```
/**
 * Set the video rendering View for the player. This control is responsible for dis
 *
 * @param view Player rendering View
 * @return Return value {@link V2TXLiveCode}
```

```
 *          - V2TXLIVE_OK : Success
 */
player.setRenderView(view);



/**
 * Set the player callback.
 *
 * By setting the callback, you can listen to some callback events of the V2TXLiveP
 * including player status, playback volume callback, audio and video first frame c
 *
 * @param observer the callback target object of the player,For more information, p
 */
player.setObserver(this);



/**
 * For key request, please refer to License acquisition
 * Set the key
 *
 * @note The URL in the JSON must be the same as the URL in startLivePlay. The SDK
 */
String url = "http://5000.liveplay.myqcloud.com/live/flvtest100_1000.flv?request_ty
";



/**
 * Start playing the audio and video stream.
 *
 * @param url @param url The playback address of the audio and video stream, suppor
 * @return Return value {@link V2TXLiveCode}
 *          - V2TXLIVE_OK: Operation succeeded, start connecting and playing
 *          - V2TXLIVE_ERROR_INVALID_PARAMETER: Operation failed, the URL is not val
 *          - V2TXLIVE_ERROR_REFUSED: RTC does not support pushing and pulling the s
 */
player.startLivePlay(url);
```

License Acquisition

Set the API interface name as DescribeDRMLicense.

Interface request domain: drm.tencentcloudapi.com.

Developers need to specify the DRM type value NORMALAES, and the Track type value SD for

encryption, ContentType value LiveVideo, and ContentId as the user's stream id.

Example:

Test environment request:
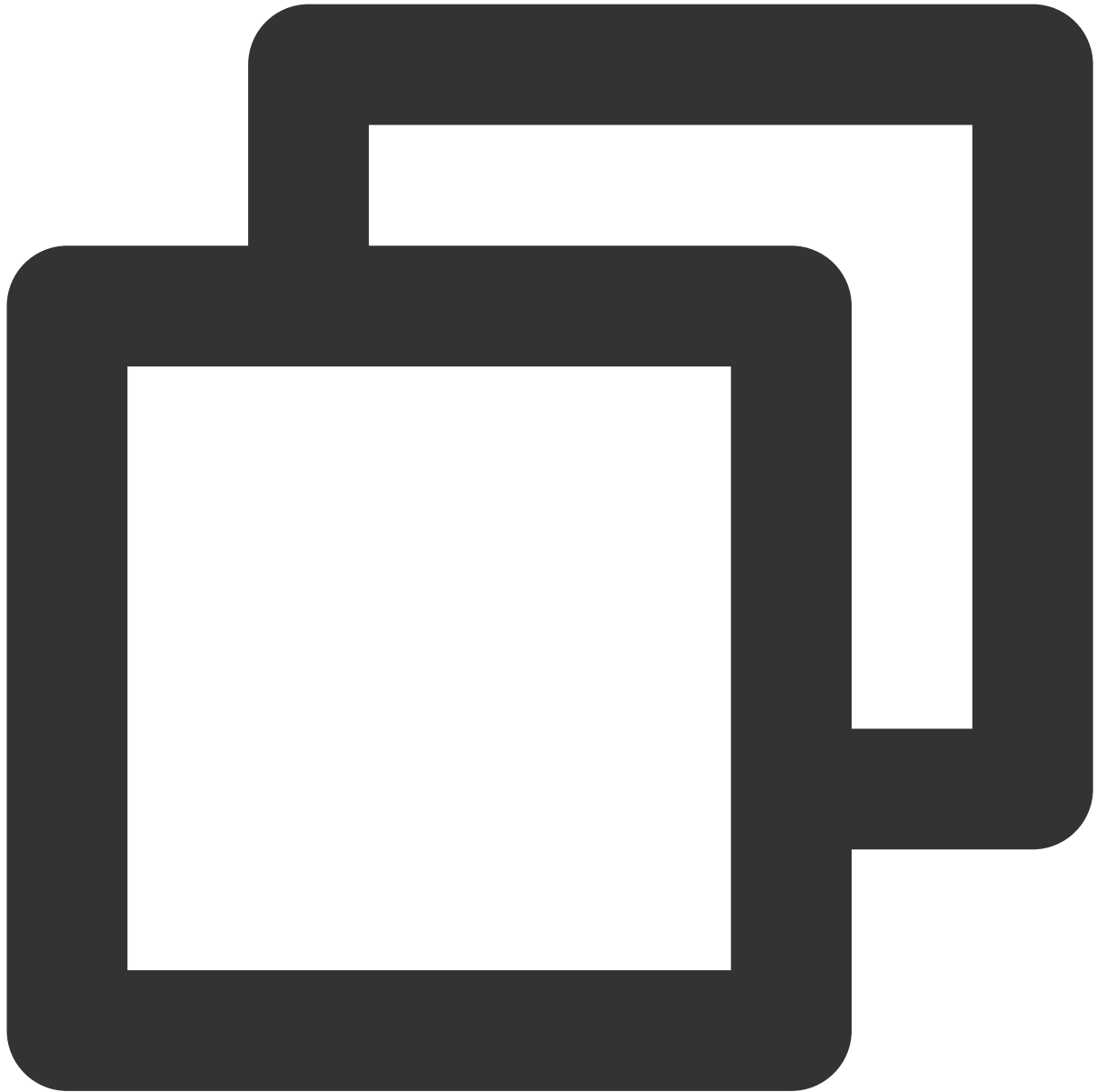
```
POST / HTTP/1.1
Host: drm.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeDRMLicense
<Public Request Parameters>

{
    "DrmType":"NORMALAES",
```

```
"ContentId":"flvtest100",
"Tracks":[
 "SD"
],
"ContentType":"LIVEVIDEO"
}
```

Request Result:
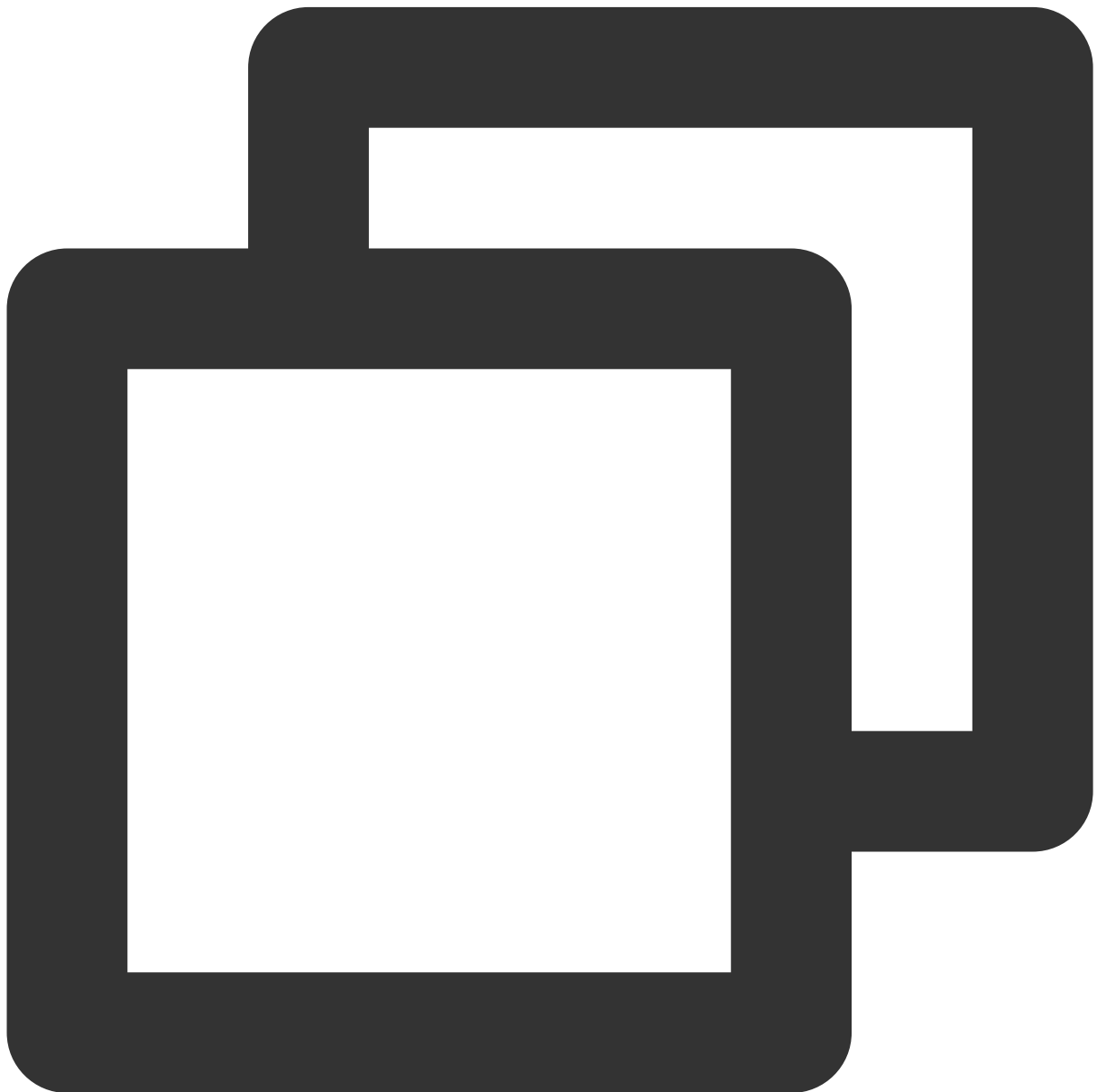
```
{
"Response": {
```

```
  "ContentId": "flvtest100",
  "TXEncryptionToken": "ZW5jTW9kZT01JmVuY0tleT0yNmFjZWIxMjViNDczMWNjODRkZTAxZWEyNDA
  "RequestId": "47f336fd-b05a-4192-b1f4-8f9d4c5f76f1"
 }
}
```

Hotlink Protection URL Calculation

Last updated：2024-07-22 16:36:54

Hotlink protection is achieved using the `txSecret` field in a push or playback URL. It can prevent attackers from forging your push URLs or using your playback URLs for profit without authorization.

How It Works

You can configure an encryption key in the CSS console (do not disclose this key) to prevent attackers from forging your push and playback URLs:
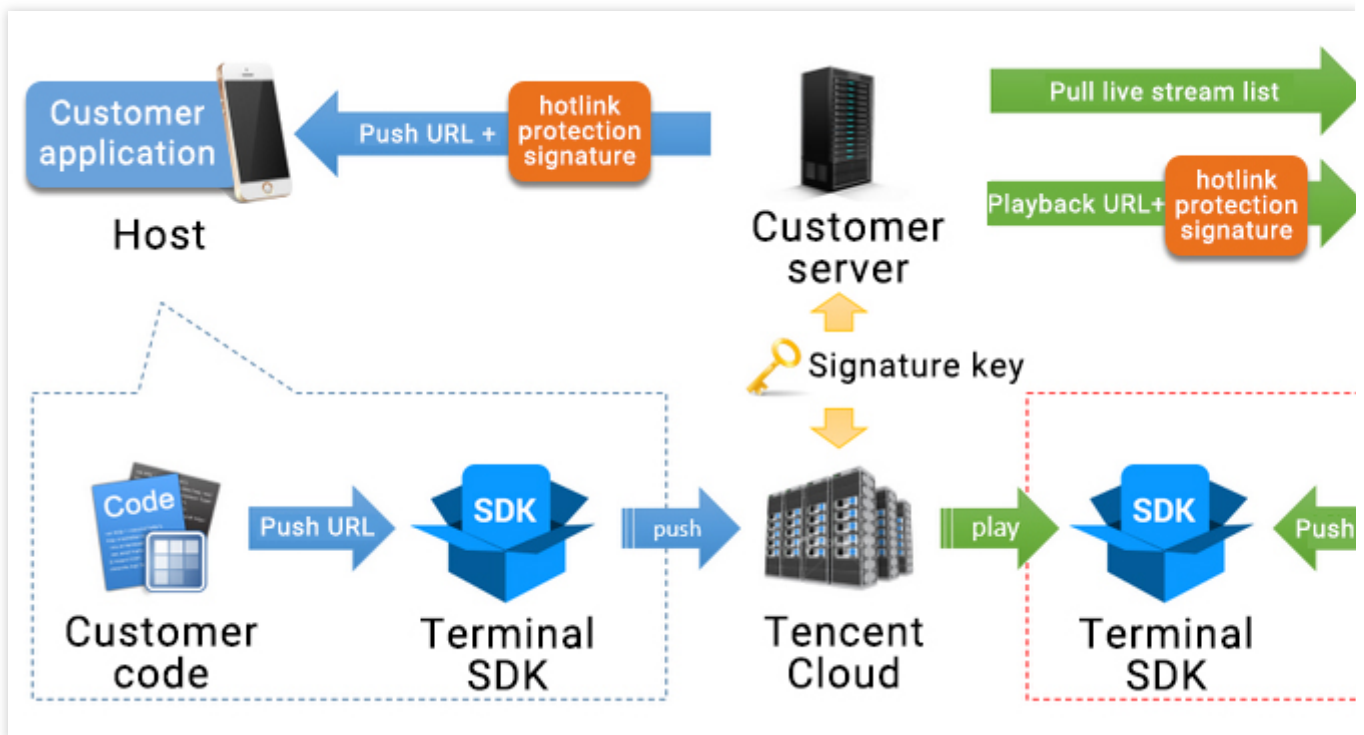
Directions

Step 1. Configure the key

First, you need to configure an encryption key in the CSS console. This is the key used to generate hotlink protection signatures on your server. Because Tencent Cloud has this key, it will be able to decrypt the signatures generated. A push hotlink protection key is used for push URLs, and a playback hotlink protection key is used for playback URLs. To configure a key for push URLs, go to the CSS console, click Domain Management on the left sidebar, and

select **Push Configuration**.

Push Address Generator

Push Domain	████████.com
Key Authentication	Enable
Type	⦿ MD5 ◯ SHA256
Key ⓘ	92f1bc5c00ebf3277d2a467f887a184d
Push Callback Address	https://████████.com/live/tools/webpush
StreamName *	test
	Only supports letters, digits, and symbols
Expiration Time	2024-07-23 15:35:40 📅

Generate Push Address

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

URL Type	Push Address
Validity Period	2024-07-23 15:35:40 (UTC+08:00) reference documentation ↗
RTMP URL	rtmp://████████.com/live/test?txSecret=99805218fb4ae433879aa85b66d9f947&txTime=669F5D4C 📋
OBS server	rtmp://████████.com/live/ 📋
OBS stream key	test?txSecret=99805218fb4ae433879aa85b66d9f947&txTime=669F5D4C 📋
WebRTC URL	webrtc://████████.com/live/test?txSecret=99805218fb4ae433879aa85b66d9f947&txTime=669F5D4C 📋
SRT URL	srt://████████.com:9000?streamid=#!::h=████████.com,r=live/test,txSecret=99805218f
RTMP over SRT URL	rtmp://████████.com:3570/live/test?txSecret=99805218fb4ae433879aa85b66d9f947&txTime=669F5D4C

For how to configure a playback hotlink protection key, see How can I enable hotlink protection?.

Step 2. Generate `txTime`

The plaintext in the signature is `txTime` , which is the validity period of the URL. For example, if the current time is `2024-07-23 15:35:40` , and you want your push URL to expire in three hours, `txTime` should be `2024-07-23 18:35:40` .

To simplify the URL generated, the time string is converted to a Unix timestamp (you can perform this by calling the

time API), and to further shorten the string, the timestamp is converted to a hexadecimal or decimal string. Therefore, in the example above, `txTime` should be `1694669601` (decimal) or `65029b21` (hexadecimal).

Note:

The actual final end time of the playback address is `txTime` + plus the validity period of the authentication key.

Changing the authentication validity period does not affect the URL generation, but it can extend the authentication validity time of the address.

Make sure you specify an appropriate validity period (not too long or too short) for the URL:

If the validity period is too short, when a host is disconnected during a live stream, they may be unable to resume publishing due to expiration of the push URL.

If the validity period is too long, your URL may be hotlinked.

Step 3. Generate `txSecret`

`txSecret` is generated using `MD5(KEY + StreamName + txTime)`. MD5 is a one-way hashing algorithm(Currently, the SHA256 encryption algorithm has also been added.). `KEY` is the encryption key configured in step 1. `StreamName` is the stream ID. We recommend you set it to a random number or the user ID. In the example below, `StreamName` is set to `test`, and `txTime` is the hexadecimal string calculated in step 2. Example:
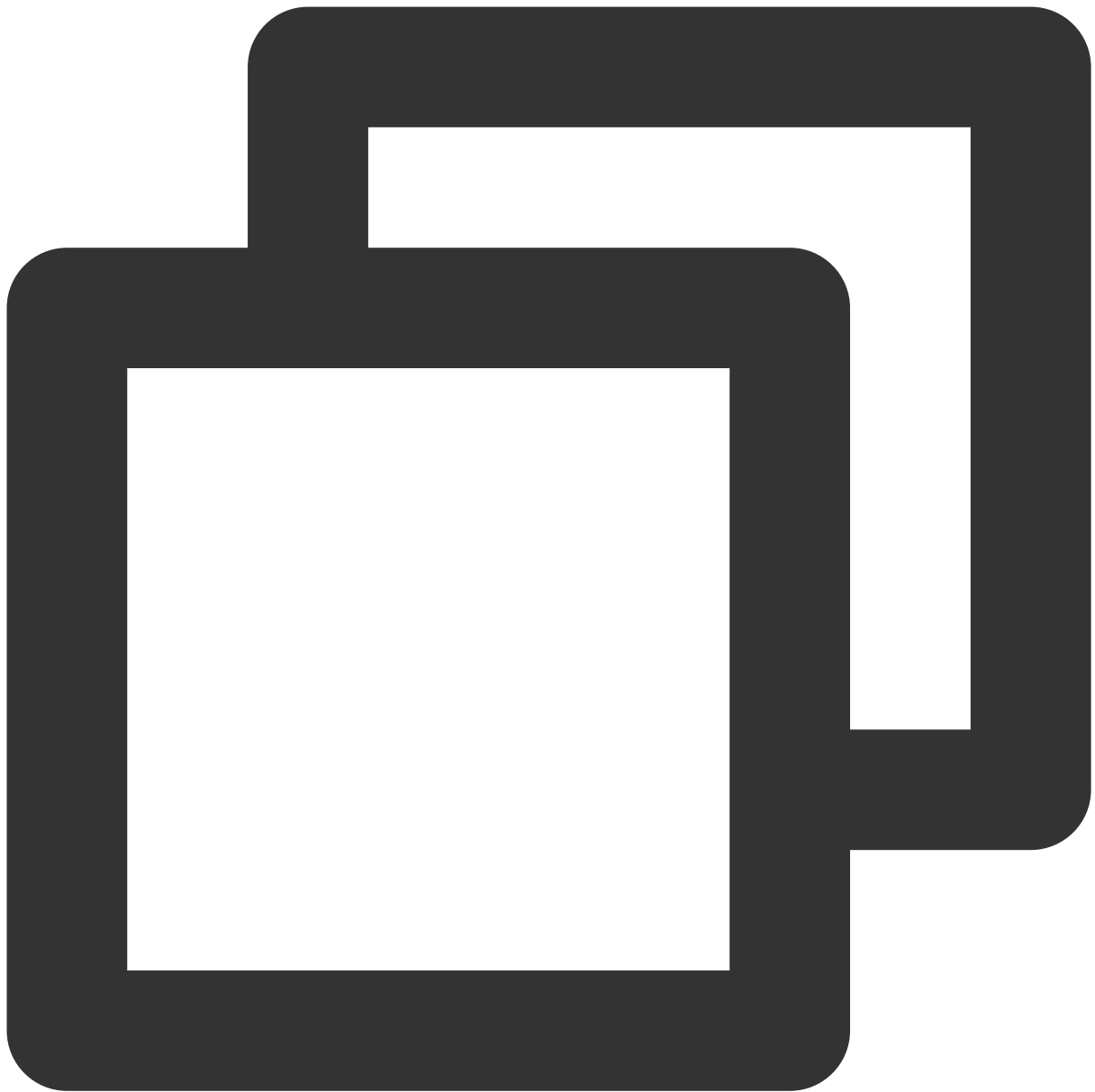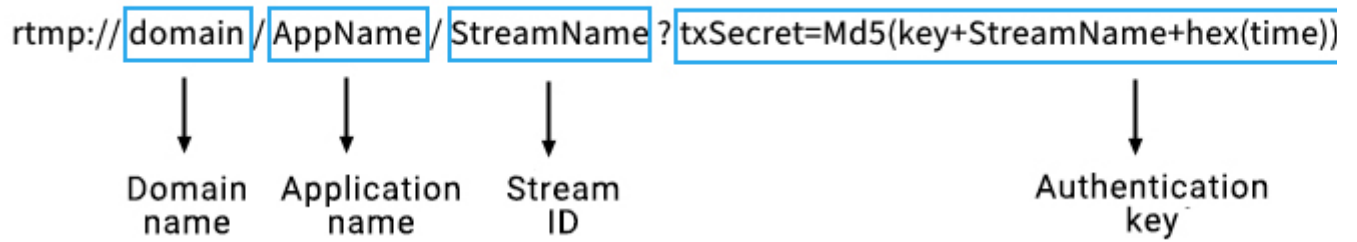
```
Suppose `KEY` is `e12c46f2612d5106e2034781ab261ca3`.
txSecret = MD5(e12c46f2612d5106e2034781ab261ca3test5C271099) = f85a2ab363fe4deaffef
```

Step 4. Generate a hotlink protection URL

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

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

Domain name → Domain name
Application name → Application name
Stream ID → Stream ID
Authentication key → Authentication key

Now that we have the URL expiration time `txTime` , the signature `txSecret` (which only Tencent Cloud can decrypt), the stream ID `StreamName` , and the push domain (suppose it's `livepush.tcloud.com`), we can generate a hotlink protection push URL:
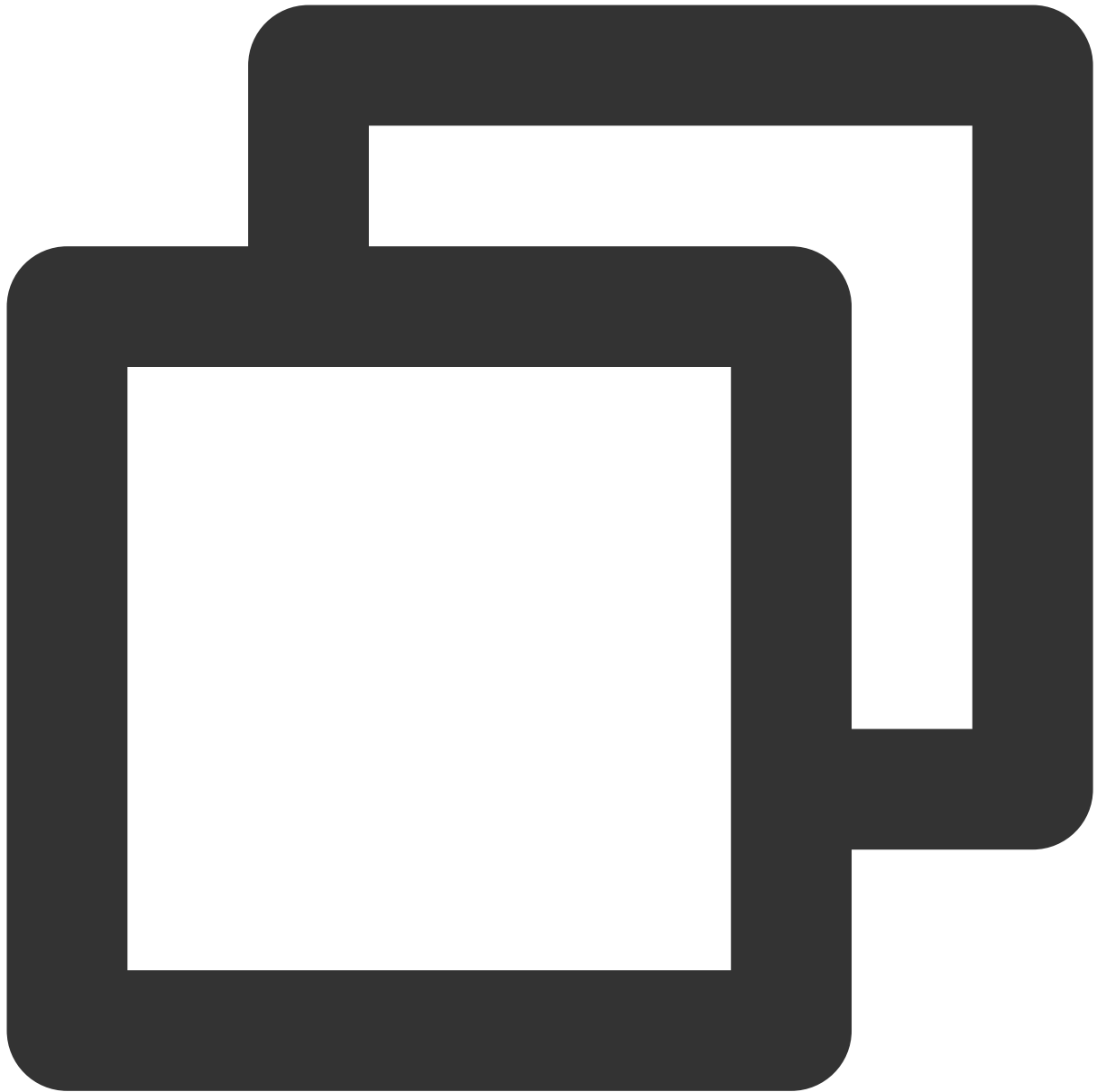
```
rtmp://livepush.tcloud.com/live/test?txSecret=f85a2ab363fe4deaffef9754d79da6fe&txTi
```

Sample Code

We offer sample code for the generation of hotlink protection URLs. In the CSS console, click Domain Management on the left sidebar, select a push domain, and click **Push Configuration**. Scroll down, and you will find sample code for PHP, Java, and Go. For details, see Push Configuration.

Global CSS Service Overview

Last updated：2024-02-01 11:10:15

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.

Asia
North America
South America
Europe
Africa
Oceania

50+ countries and regions   2,000+
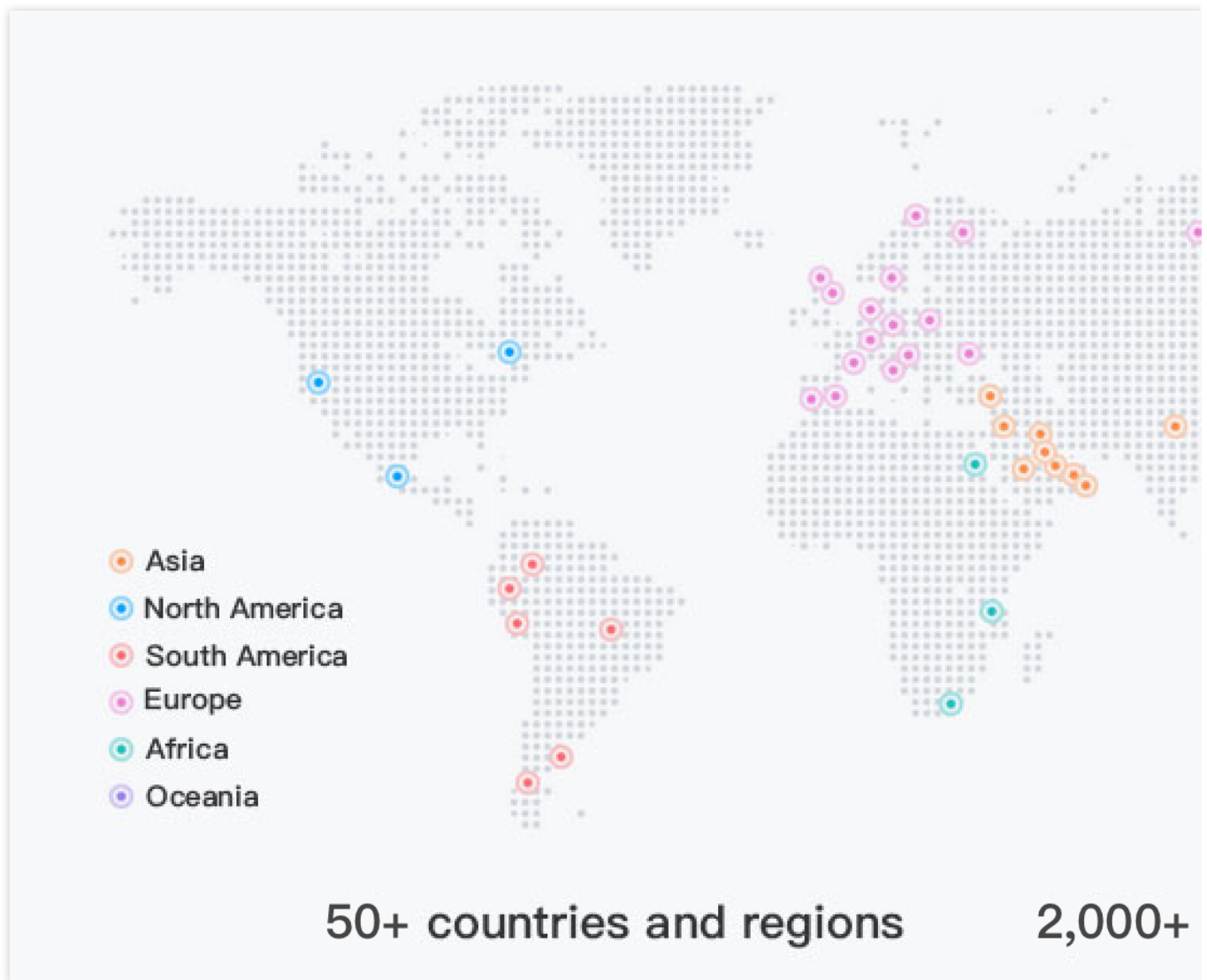
In addition to push and playback, a complete live broadcasting service should also include authentication, transcoding, screencapturing, recording, callback, porn detection, DRM, and other features. The figure below shows the basic feature modules of Tencent Cloud's Global CSS solution.
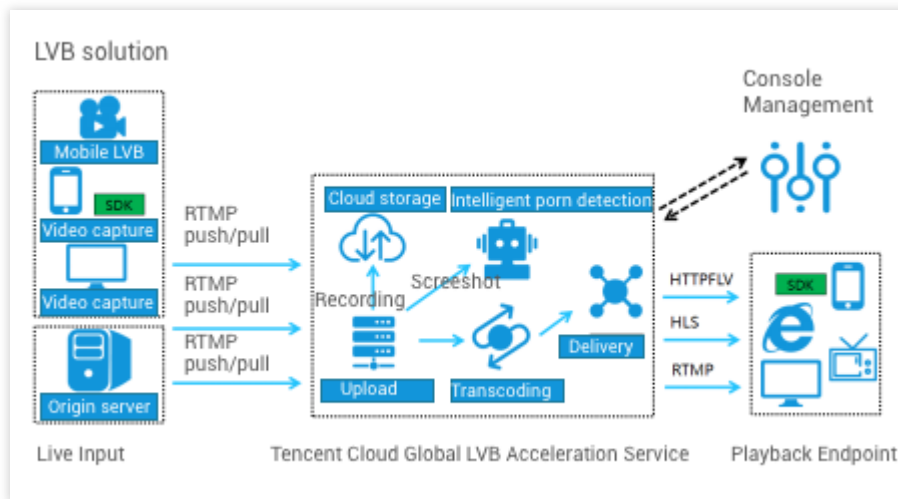
The basic features of global live broadcasting are generally the same as those of domestic live broadcasting. However, there are more challenges overseas mainly due to the wider geographical area, more complicated network environments, and lower cross-border network quality. In order to reduce the latency 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, 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.

Direct Connect acceleration vs. TCP playback in terms of lagging rate for viewers outside Mainland China

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.**
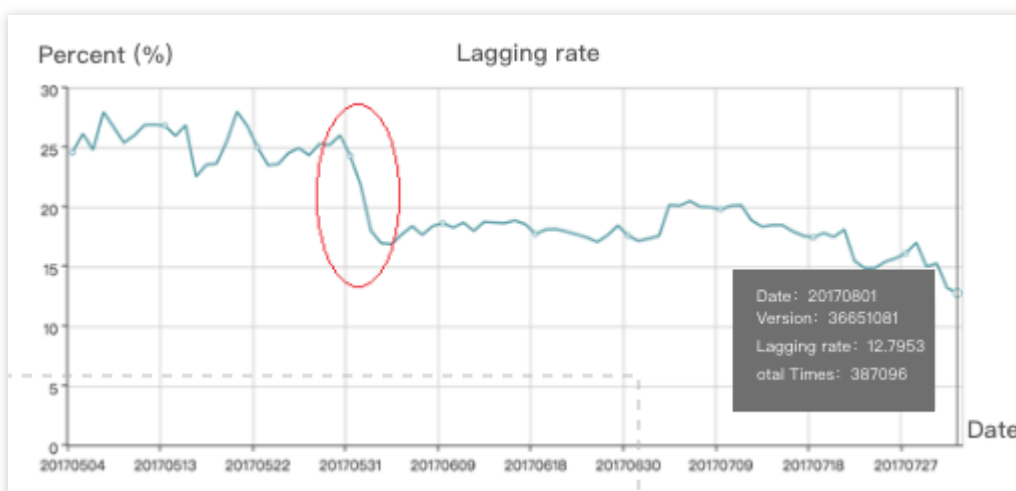
Optimal Access and Failover

Similar to Mainland China, other regions also often have multiple ISPs, such as DTAC, AIS, and TRUE 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 TRUE) 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.
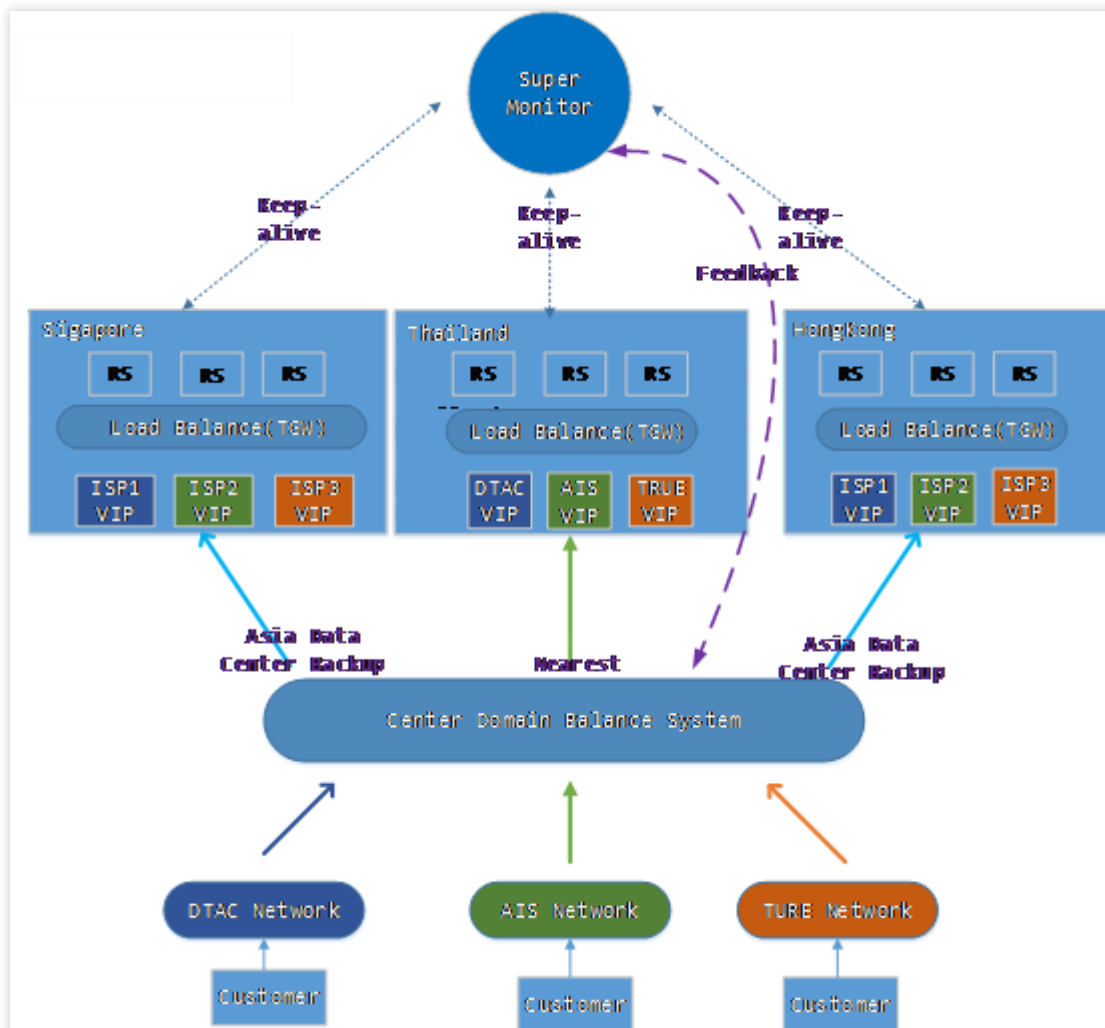
Network Transmission Optimization

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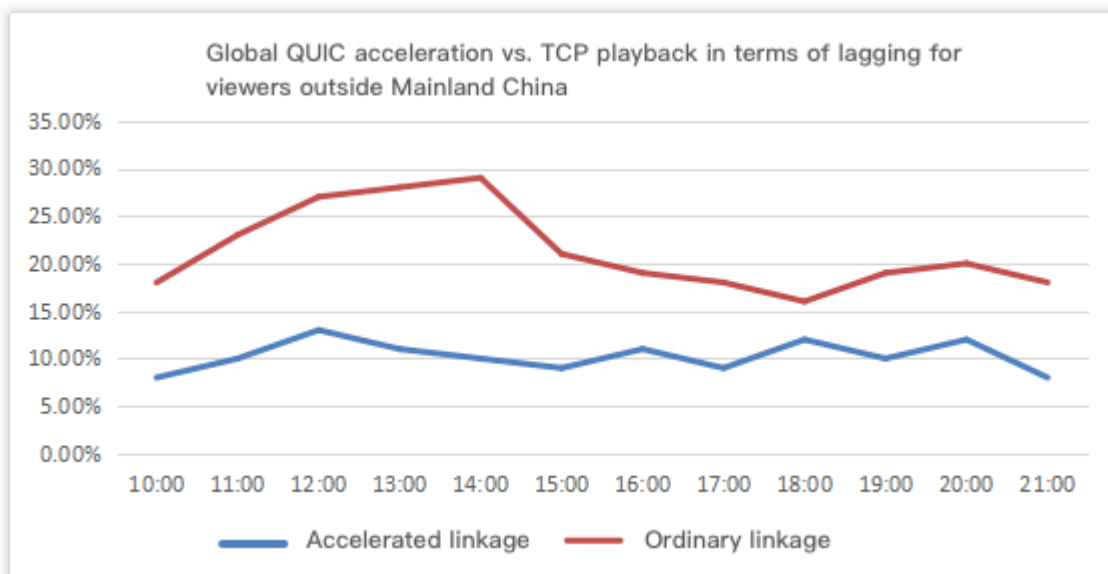
The figure below shows a comparison of lags for global viewers watching a live stream pushed by a host in UAE to the UAE cache node. You can see that the lags remain low when the stream is accelerated via QUIC.
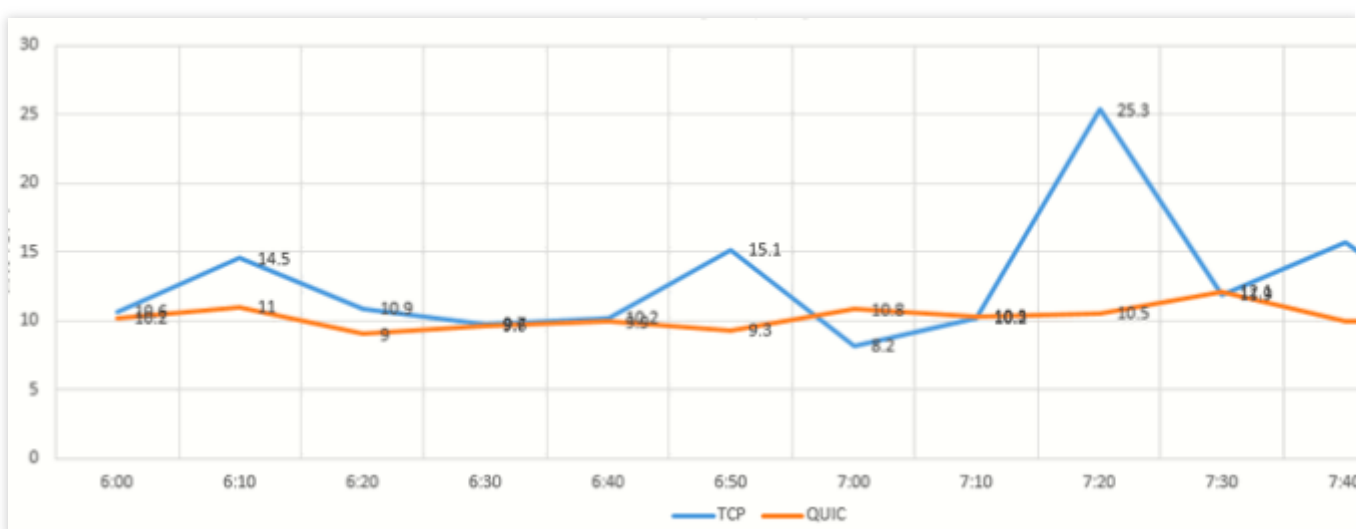
Massive Resource Reserve

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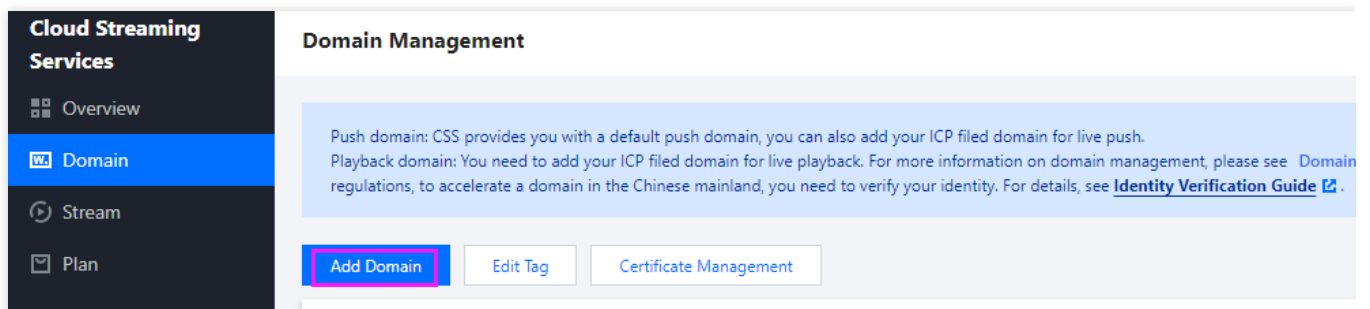
In the pop-up window, select the type as **Playback Domain**, select the corresponding **Acceleration Region**, and enter the **Domain Name** that needs to be accelerated.

HTTPDNS Routing

Last updated：2024-06-05 17:36:19

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.For more details, please refer to HTTPDNS.

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.
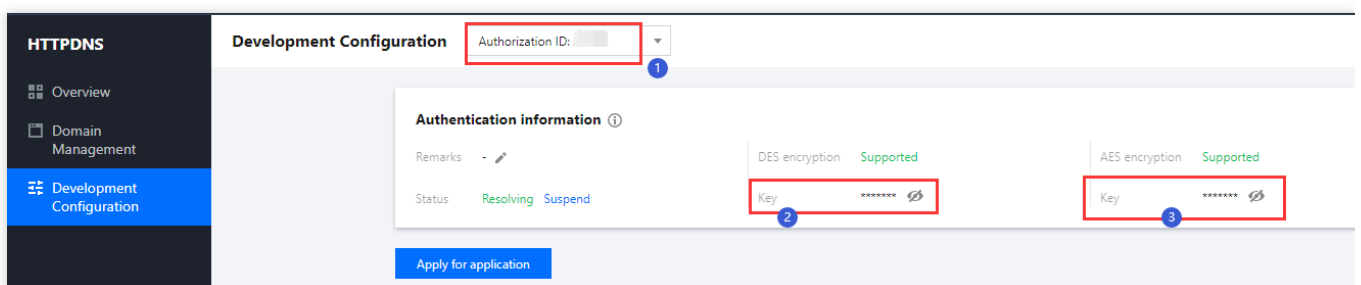
Routing Push Traffic Using HTTPDNS

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
play_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):
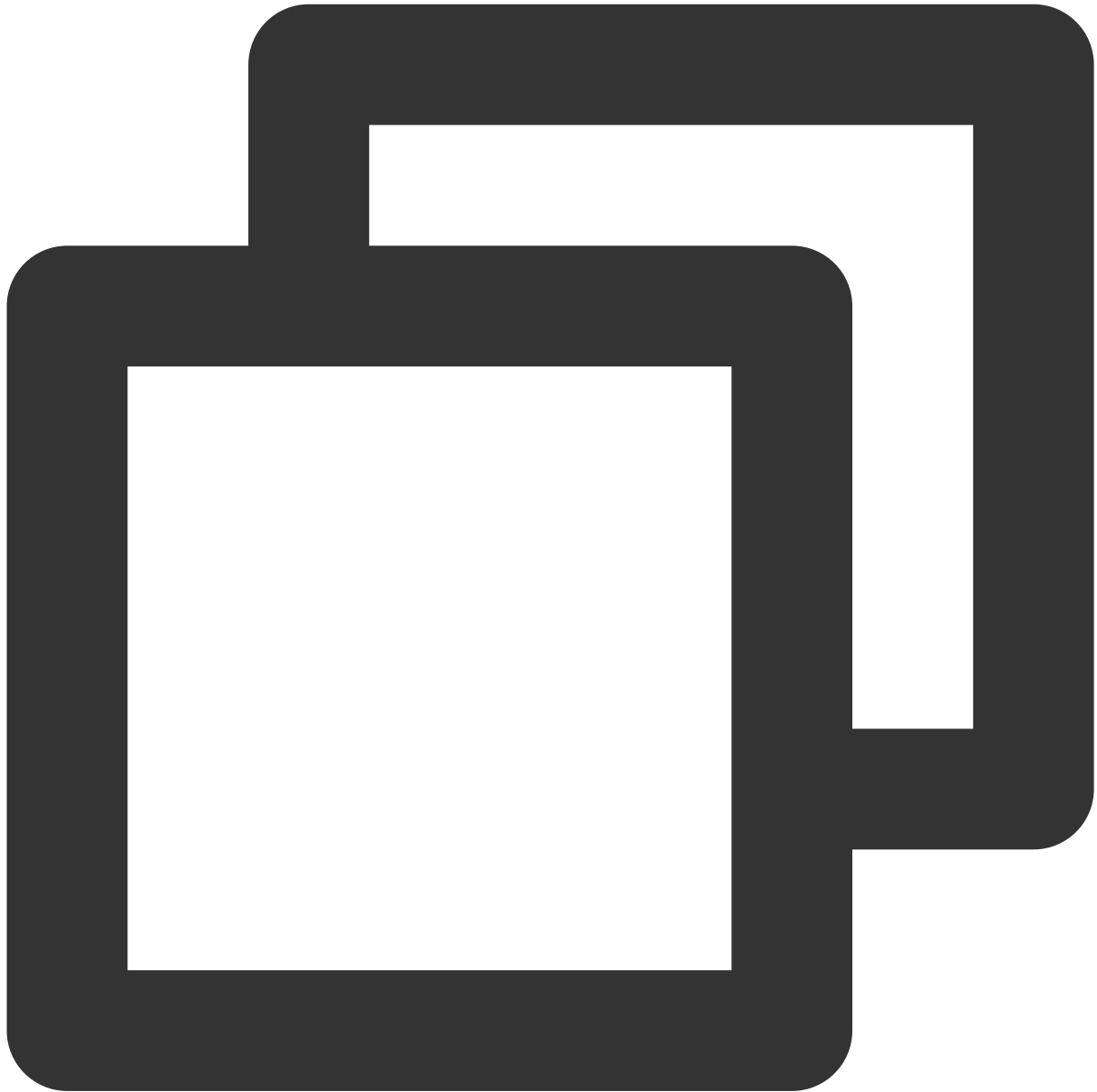

```
http://server_ip/play_domain/live/streamname.flv?xxxxxxxxxx
http://server_ip/play_domain/live/ streamname.m3u8?xxxxxxxxxx
http://server_ip/play_domain/live/ streamname -123.ts?xxxxxxxxxx
```

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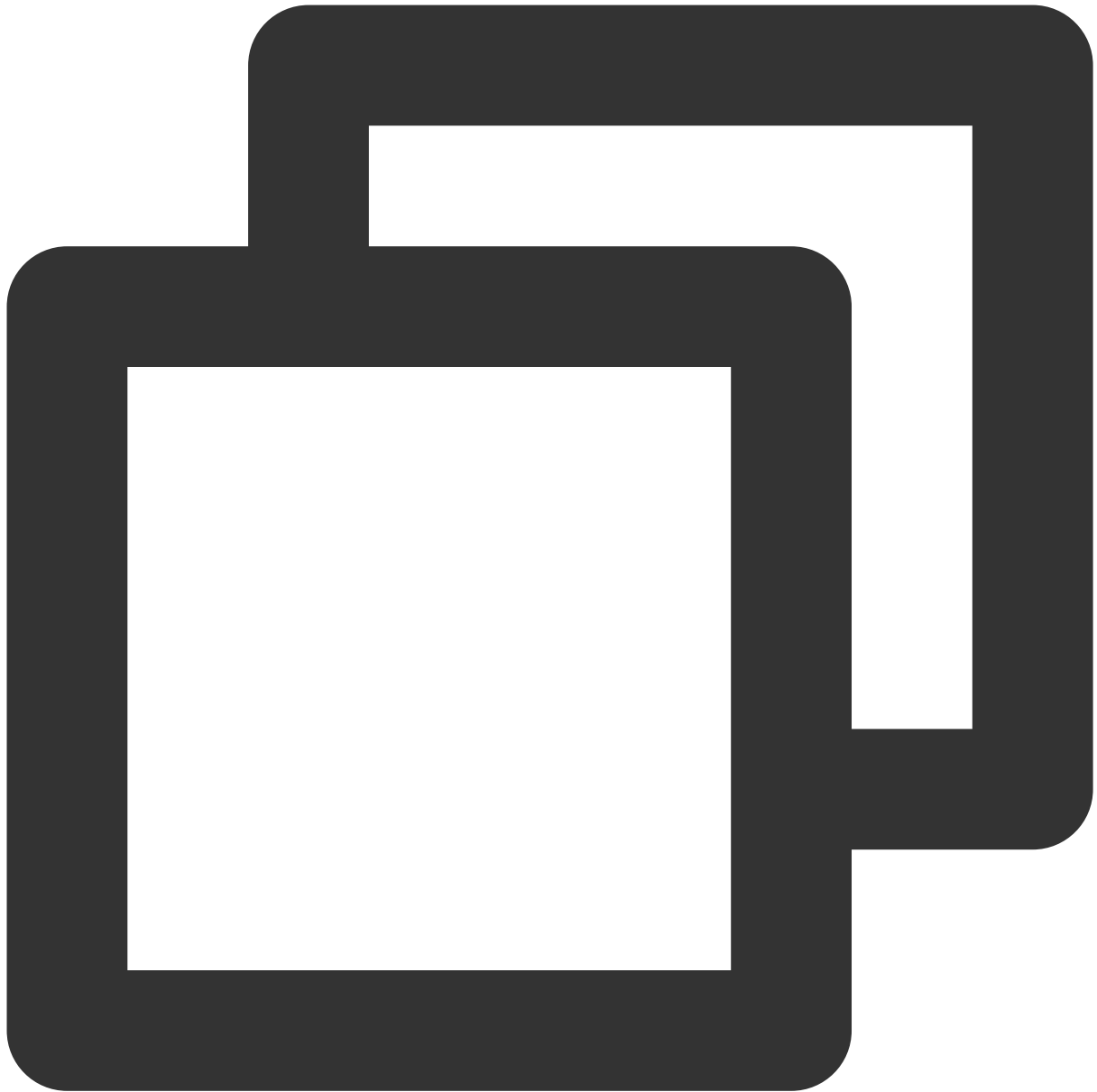
```
https://server_ip/play_domain/live/ streamname.flv?xxxxxxxxxx
https://server_ip/play_domain/live/ streamname.m3u8?xxxxxxxxxx
https://server_ip/play_domain/live/ streamname -123.ts?xxxxxxxxxx
```

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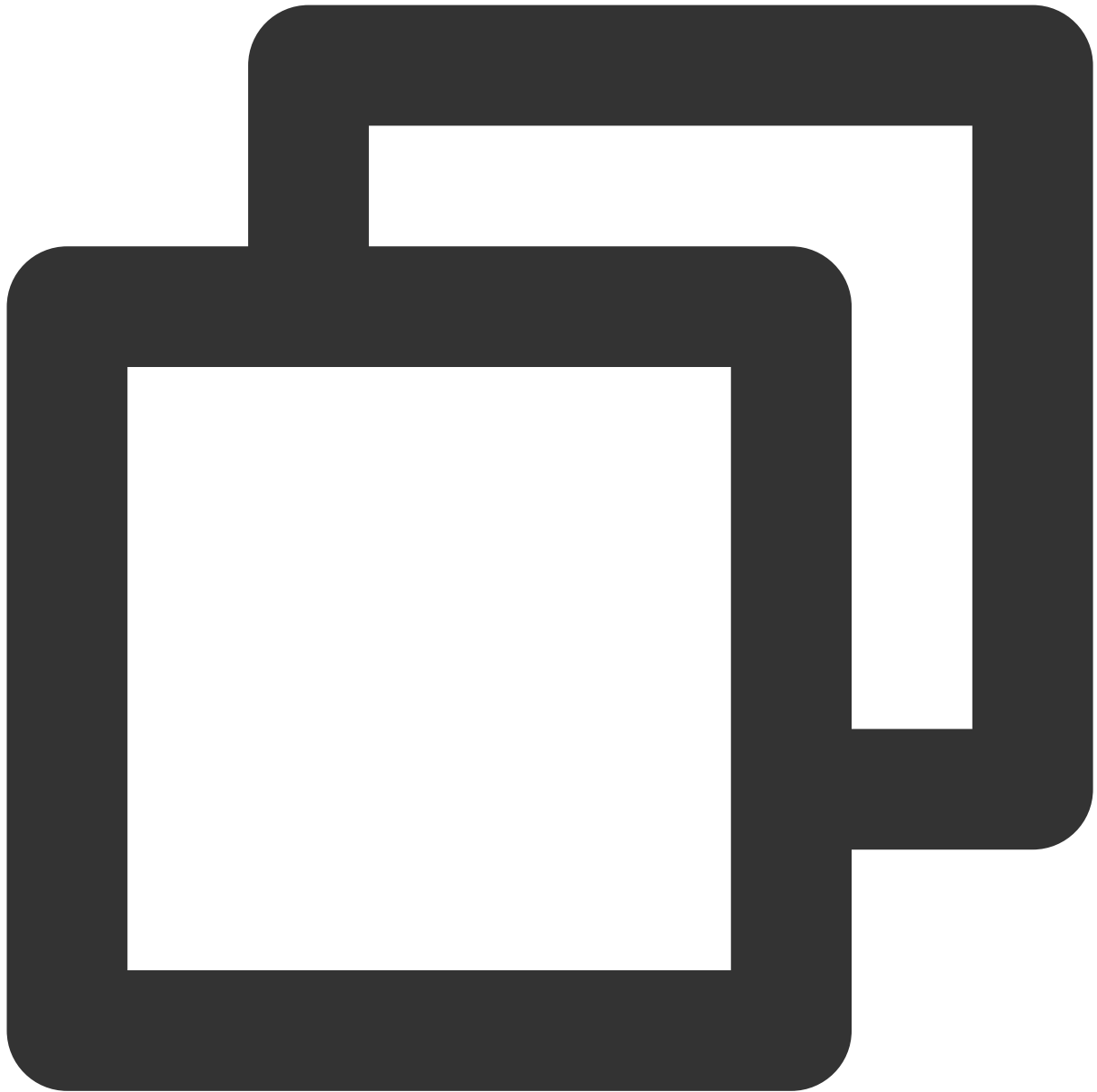
```
rtmp://server_ip/play_domain/live/ streamname?xxxxxxxxxx
```

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：2023-10-09 15:47:34

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
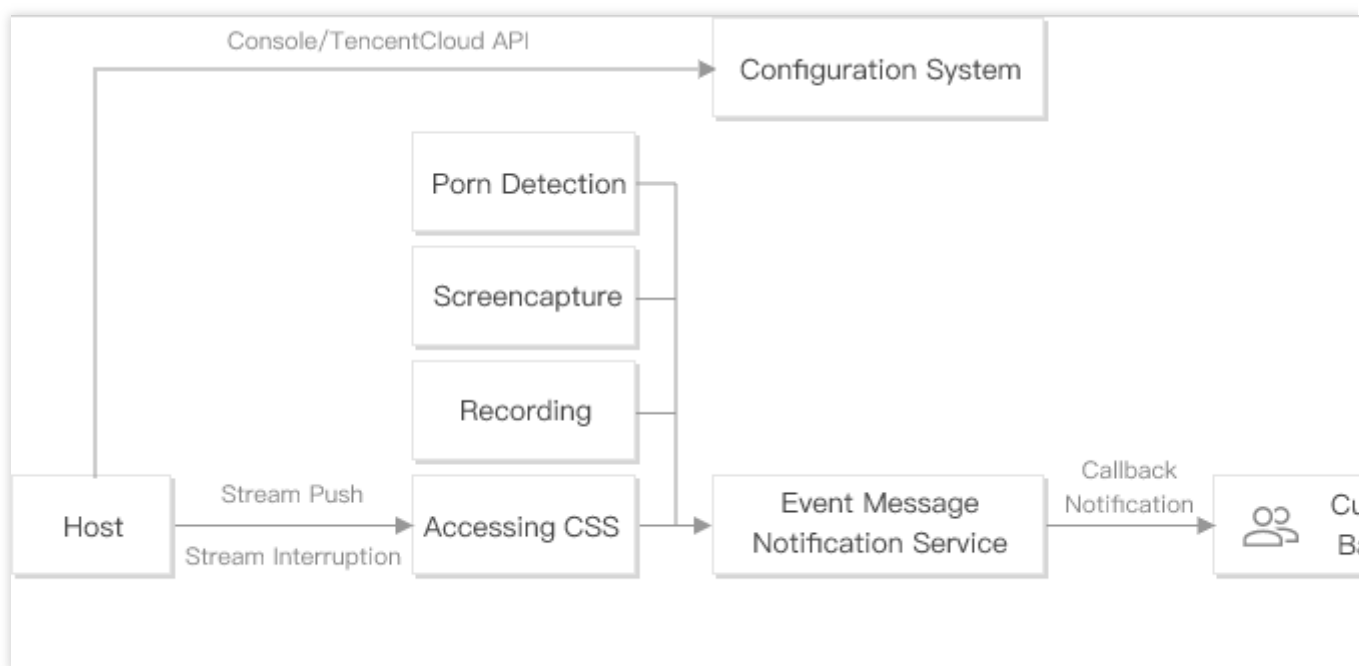
Process Description:

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 **Feature Configuration** > **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
Relay event notification

Stream Pushing Notification

Last updated：2024-01-18 17:53:42

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. 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 **Feature Configuration** > Live Stream Callback, which is used for authentication. We recommend you set this field to ensure data security.

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
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

Live Stream Push Callback Message Example
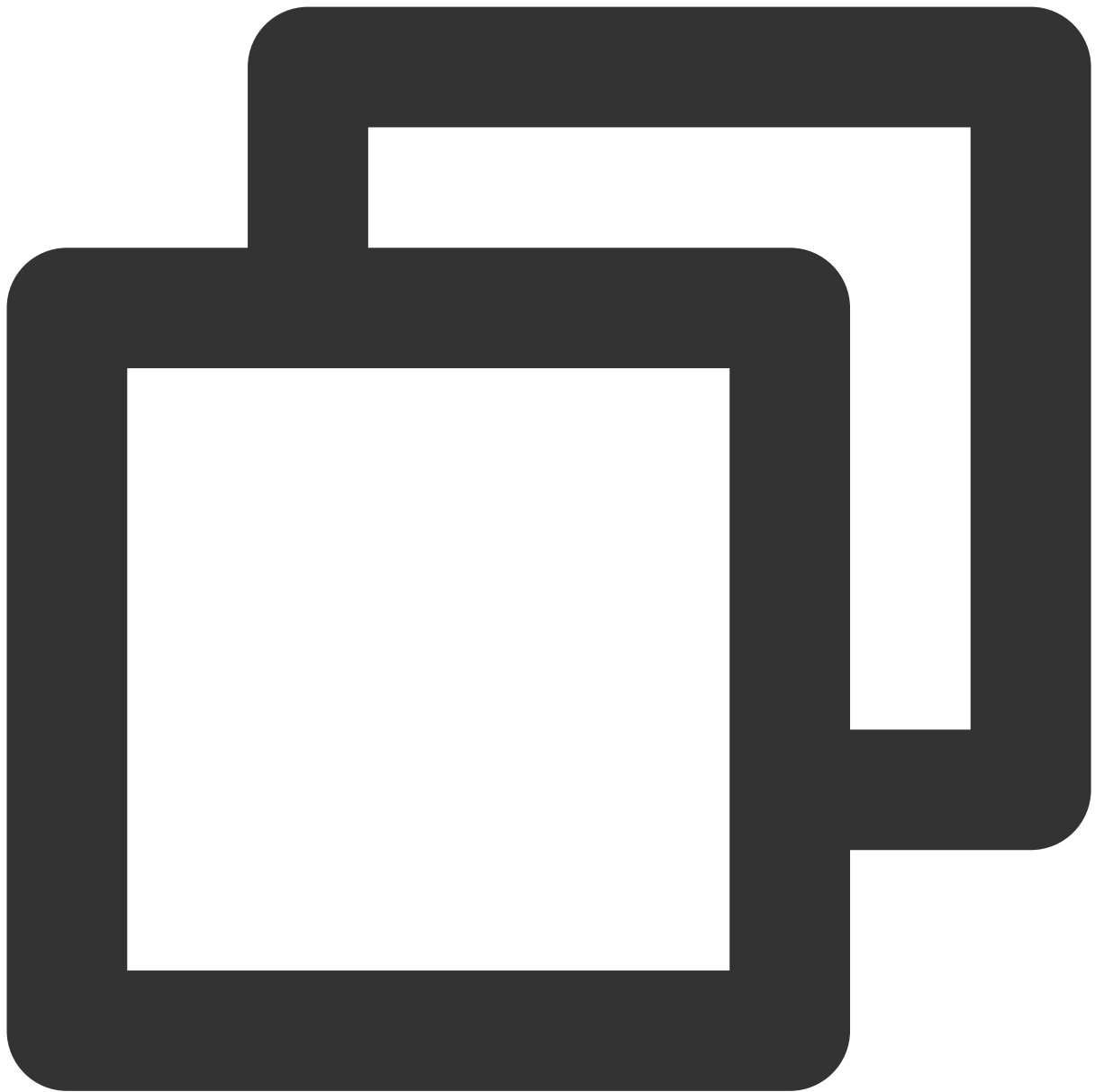
```
{
    "app":"test.domain.com",
    "appid":12345678,
    "appname":"live",
    "channel_id":"test_stream",
    "errcode":0,
    "errmsg":"ok",
    "event_time":1703731478,
    "event_type":1,
    "height":0,
    "idc_id":34,
```

```
    "node":"42.81.194.37",
    "sequence":"2210464508206756938",
    "set_id":2,
    "sign":"df49*********************f5d4",
    "stream_id":"test_stream",
    "stream_param":"stream_param=test",
    "t":1703732078,
    "user_ip":"1.1.1.1",
    "width":0
}
```

Live Stream Interruption Callback Message Example
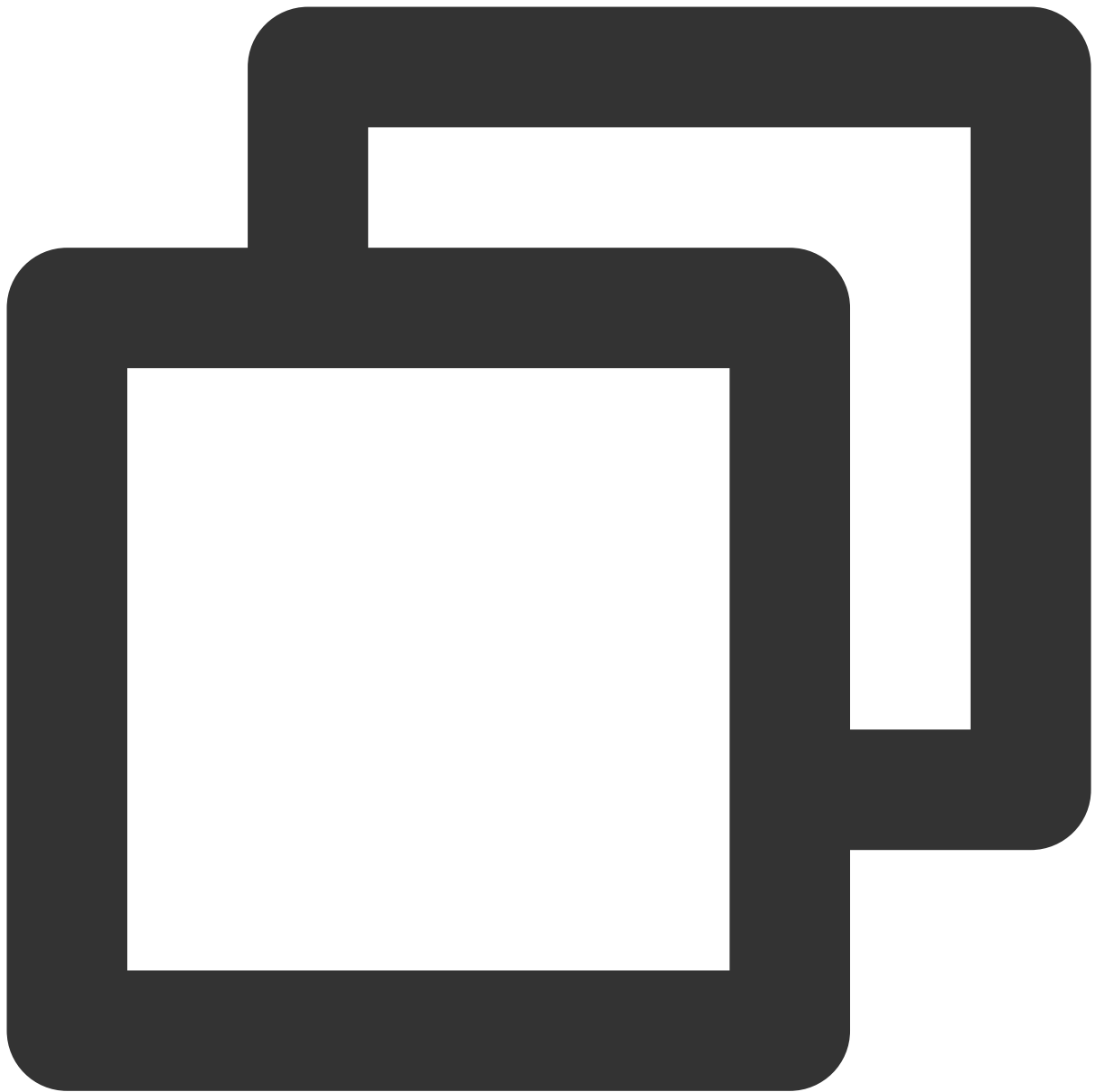
```
{
    "app":"test.domain.com",
    "appid":12345678,
    "appname":"live",
    "channel_id":"test_stream",
    "errcode":1,
    "errmsg":"The push client actively stopped the push",
    "event_time":1703731606,
    "event_type":0,
    "height":0,
    "idc_id":34,
```

```
    "node":"42.81.194.37",
    "push_duration":"128581",
    "sequence":"2210464508206756938",
    "set_id":2,
    "sign":"3485*********************56ae",
    "stream_id":"test_stream",
    "stream_param":"stream_param=test",
    "t":1703732206,
    "user_ip":"1.1.1.1",
    "width":0
}
```

Recording Event Notification

Last updated：2024-07-24 10:32:13

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 file 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 (files)	event_type = 100

Common callback parameters

Parameter	Type	Description
t	int64	The time (Unix timestamp) when the notification signature expires. 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.

		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 **Feature Configuration** > Live Stream Callback, which is used for authentication. We recommend you set this field to ensure data security.

Recording callback parameters

Parameter	Type	Description
appid	int	The user APPID.
app	string	The push domain.
appname	string	The push path.
stream_id	string	The live stream name.
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: `flv` , `hls` , `mp4` , `aac` .

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.
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. The difference between start_time and end_time may be different from the duration value under normal circumstances, especially in cases of weak network connections and stream pushing anomalies.
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: `video_codec` indicates the video codec. `resolution` indicates the resolution of the pushed stream. `session_id` indicates the recording task ID. These are all additional fields of a recording callback. We recommend you do not rely your business logic too much on them.

Sample callback
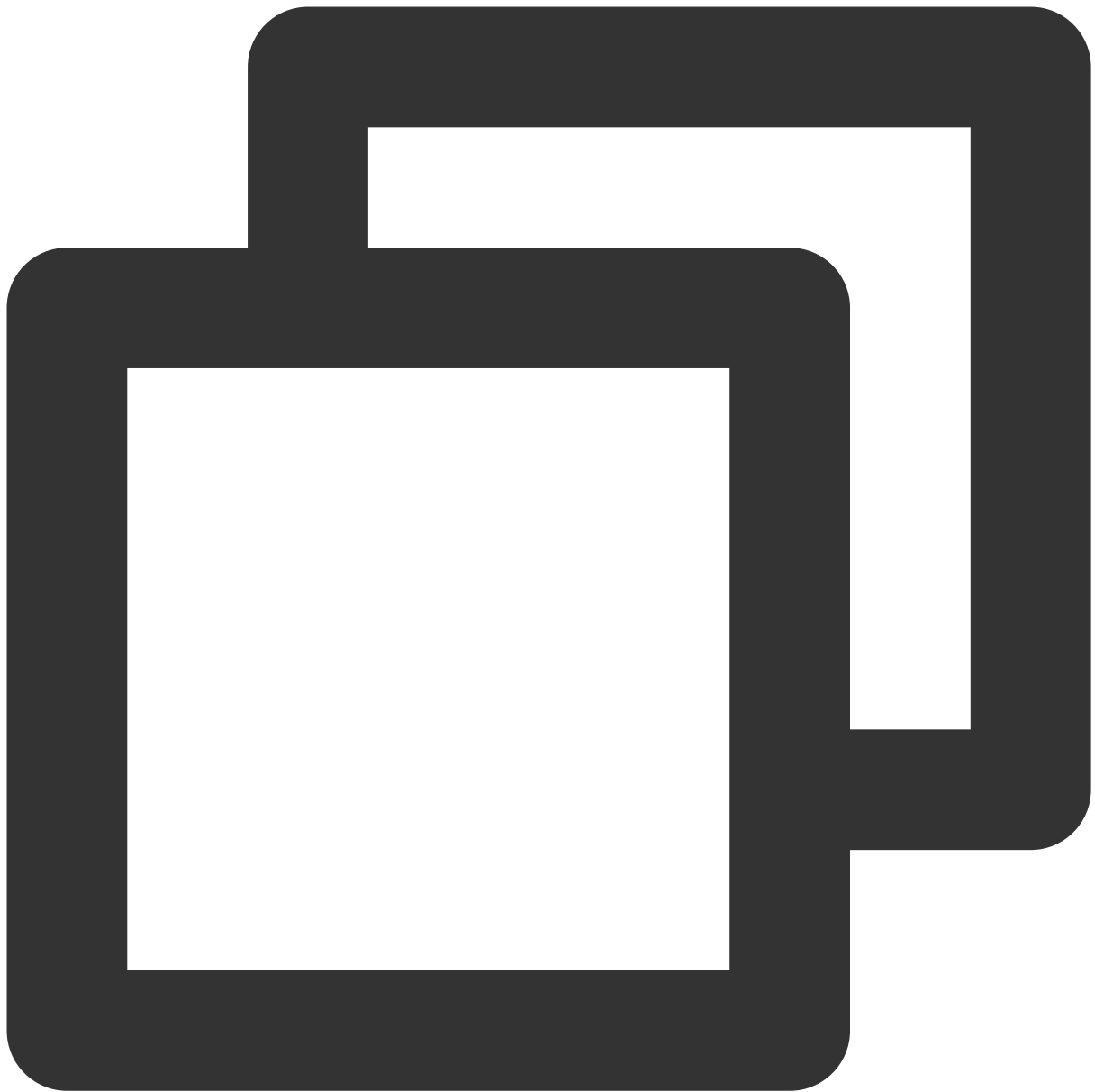
```
{
    "event_type": 100,

    "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*******************0xTSlNTQltlRVRLU1JAWW9EUb",

    "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
}
```

Recording Status Event Notification

Last updated：2024-07-24 10:32:13

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 status callback notifies you of the status of a recording task, including whether it started or ended successfully, when it is paused and resumed successfully, and if any recording errors occur. 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 status event occurs.

Notes

This document assumes you already know how to configure and receive callbacks.
In the relay recording callback, the stream ID refers to the Task ID of the relay task.

Description of recording status callback parameters

Event type

Event Type	Explanation of Field Value
Live Recording	event_type = 332

Common callback parameters

Field Name	Type	Description
t	int64	Expiration Time: UNIX timestamp when the event notification signature expires. 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. 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 **Feature Configuration** > Live Stream Callback, which is used for authentication. We recommend you set this field to ensure data security.

Recording status callback message parameters

Field Name	Type	Description
appid	int	User APPID
appname	string	Push path
domain	string	Push domain name
event_time	int	Event time
event_type	int	Event type
record_detail	string	ile_format: 1: FLV 2: HLS 3: MP4 4: AAC 5: MP3 record_bps:Bitrate start_model:Task initiation method 1: Initiation via recording template rules 5: Initiation via API call record_content: Recording content

		1: Original stream 2: Watermarked stream 3: Transcoding stream source_type: Recording stream type 1: Live recording 2: Relay recording codec_temp_id: Transcoding template ID
record_event	string	record_start_succeeded : Successful recording startup record_start_failed: Failed recording startup record_paused : Recording pause record_resumed : Successful recording continuation record_error : Recording anomalies record_ended : Ended recording
seq	string	Message sequence number
session_id	string	Recording task ID
stream_id	string	Live stream name

Sample callback message
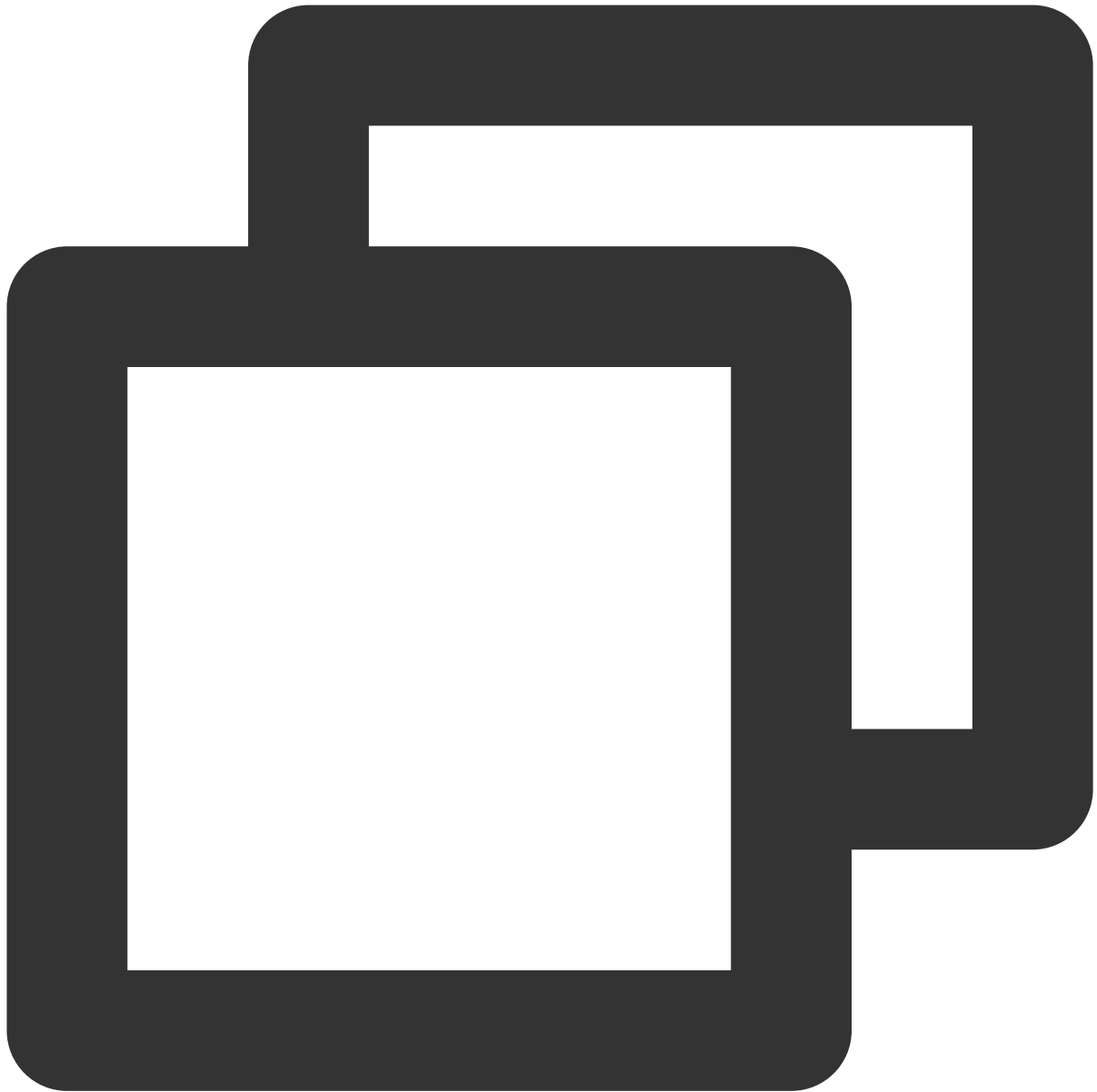
```
{
    "appid":123456789,
    "appname": "live",
    "domain":"****.livepush.myqcloud.com",
    "event_time":1700207929,
    "event_type":332,
    "record_detail":"{\\\\\"file_format\\\\\":2,\\\\\"record_bps\\\\\":0,\\\\\"start_mod
    "record_event":"record_ended",
    "seq": "3266441426274648065",
    "session_id":"2918085116267032069",
    "stream_id":"2991615887188599295"
```

```
}
```

Screencapturing Event Notification

Last updated：2024-07-24 10:32:13

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	Expiration time, which is the Unix timestamp when the event notification signature expires. 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	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：

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

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
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
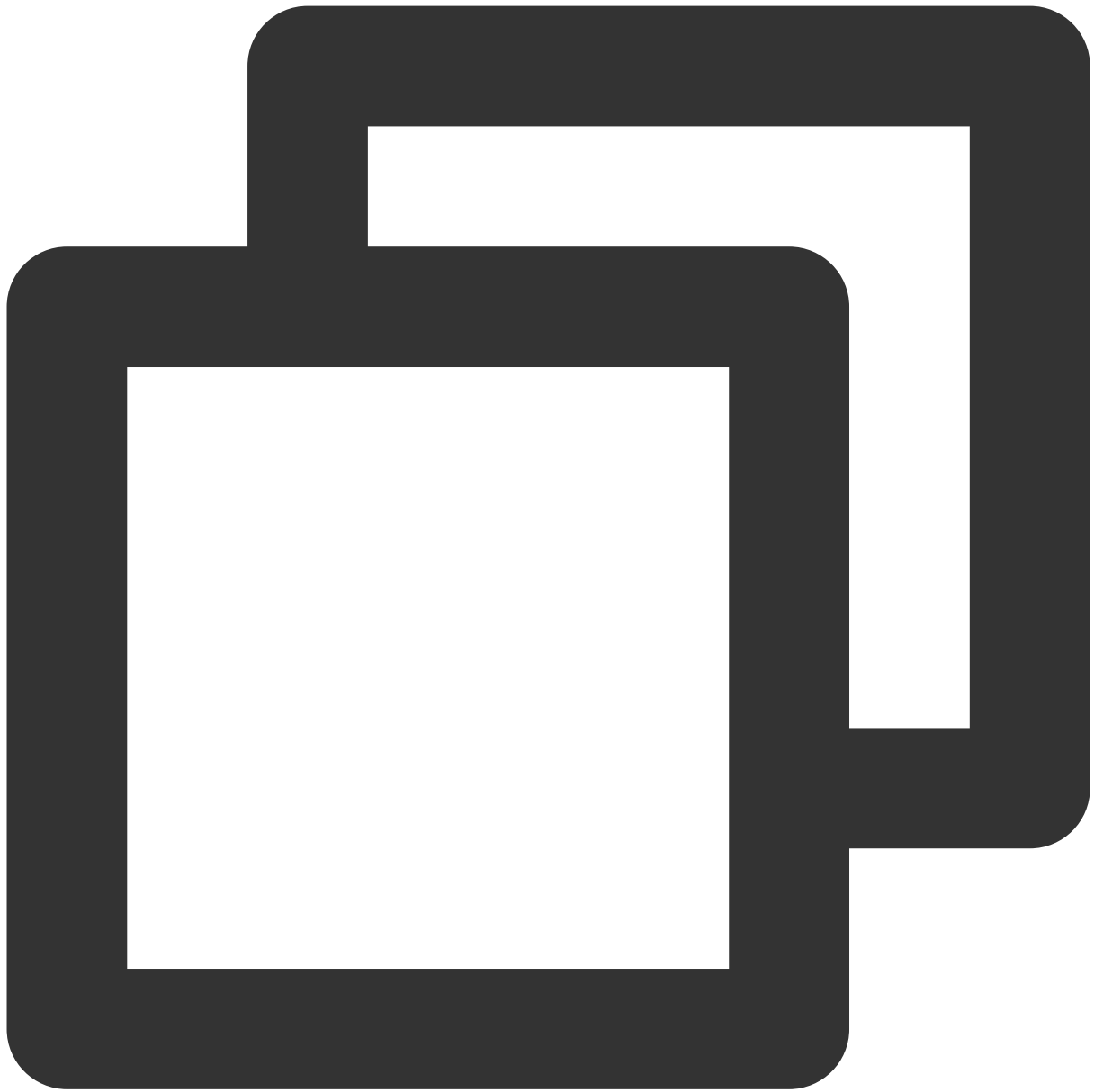

```
{
    "app":"test.app",

    "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-screens

    "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
}
```

Live Broadcasting Image Audit Event Notification

Last updated：2024-07-29 16:21:23

If you have configured an image moderation callback address, the server will return the moderation results in JSON to the specified callback address after the image moderation is complete. You can then proceed with subsequent file processing operation based on the callback content.

This document describes the notification fields of the callback message sent by Tencent Cloud Streaming Services (CSS) to the user after the image moderation callback event is triggered.

Note

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 Notifications.

By default, only potentially non-compliant results will be returned. Compliant results will not be returned.

It is recommended that you employ the type, score, and suggestion parameters from the callback message to moderate non-compliant images. The moderation system does not provide 100% accuracy, so a few images may be identified as potentially non-compliant or incorrectly identified. You can determine whether manual check is required based on the actual application scenario.

The image moderation service has been upgraded, and some callback message parameters have been adjusted. You are advised to refer to the latest version and the following parameter description. To ensure compatibility with old versions for users, we will still call back some parameters that are currently not used to the receiving server, including: tid, abductionRisk, confidence, normalScore, hotScore, pornScore, terrorScore, polityScore, illegalScore, similarScore, abuseScore, teenagerScore, adScore, and customScore.

Image Moderation Callback Parameters

Event Type Parameter

Event Type	Parameter Value
Image moderation	event_type = 317

Common Callback Parameters

Parameter	Type	Description

t	int64	Expiration time, which is the Unix timestamp when the event notification signature expires. 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 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	Event notification security signature sign = MD5(key + t). 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.

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 message parameters

Parameter	Required or Not	Data Type	Description
streamId	No	String	Stream name.
channelId	No	String	Channel ID.
img	Yes	String	Link to the moderated image

type	Yes	Array	Categories of negative labels with the highest priority in the detection result. For details, see the description of `label`.
score	Yes	Array	Scores of `type`
ocrMsg	No	String	OCR result (if any)
suggestion	Yes	String	Suggestion. Valid values: Block Review Pass
label	Yes	String	Negative label with the highest priority in the detection result(labelResults,objectResults,ocrResults). 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 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 `null`, 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 `ObjectResults`. Note: **This field may return `null`, 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 `OcrResults`. Note: **This field may return `null`, 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: 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: Block Review Pass
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 (`QrCode`), this parameter is the URL of the QR code.
Score	Integer	Hit score of the object label. Value range: 0-100. For example, `QrCode 99` 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: Block Review Pass
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

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: Block Review Pass
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

Callback Message Example

```
{
        "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
}
```

Live Streaming Audio Auditing Service Event Notification

Last updated：2024-01-30 15:08:03

If you have configured an audio moderation callback address, the server will return the moderation results in JSON to the specified callback address after the audio moderation is complete. You can then proceed with subsequent file processing operation based on the callback content.

This document describes the notification fields of the callback message sent by Tencent Cloud Streaming Services (CSS) to the user after the audio moderation callback event is triggered.

Note

Before reading this document, ensure that you have understood how the callback function is configured and callback messages are received in Tencent CSS. For detailed procedures, see How to Receive Event Notifications.

In streaming audio moderation, only potentially non-compliant results are called back by default, not normal results.

Audio Moderation Service Event Parameter Description

Event Type Parameter

Event Type	Field Value Description
Streaming Audio Moderation Service	event_type = 315

Common Callback Parameters

Field Name	Type	Description
t	int64	Expiration time. It is the UNIX timestamp signifying the expiration of the event notification signature. The default expiration time for message notifications from Tencent Cloud is 10 minutes. If the time designated by the `t` value in a message notification has expired, the notification is deemed invalid, thereby safeguarding against network replay attacks. The format of `t` is a decimal UNIX timestamp, which is the seconds elapsed from midnight of January 1, 1970 (UTC/GMT).

sign	string	Security signature for event notification: `sign` = MD5(`key` + `t`) Tencent Cloud concatenates the encrypted **key** and t into a string, and then uses MD5 calculation to obtain the sign value, which is then placed in the notification message. After the notification is received, your backend server can determine whether the sign is correct using the same algorithm, thus confirming whether the message comes from Tencent Cloud's backend.

Note

The key is the callback key which is in **Feature Configuration** > Live Stream Callback and used for authentication. You are advised to specify this field to ensure data security.

Callback Message Parameters

Parameter	Required or Not	Data Type	Description
appid	Required	Number	Business ID.
stream_id	Mandatory	String	Stream name.
channelId	Mandatory	String	Channel ID.
domain	Mandatory	String	Push domain name.
path	Mandatory	String	Push stream path.
asr_text	Mandatory	String	Audio text.
cdn_url	Optional	String	CDN address.
duration	Optional	Number	Speech recognition duration (seconds).
label	Mandatory	String	This field is used to return the malicious label with the highest

			priority in the detection result (LabelResults) to indicate the moderation result recommended by the model. You are advised to process different violation types and suggested values according to your business requirements.
language_results	Optional	Array of AudioResultDetailLanguageResult	This field is used to return the detailed moderation results of minority language audio detection. For specific result content, please see the detailed descriptions of AudioResultDetailLanguageResult data structure. Note: **This field may return null, indicating that there is no valid value available.**
moan_results	Optional	Array of MoanResult	Moderation result of vulgar content in the audio; Note: **This field may return null, indicating that there is no valid value available.**
recognition_results	Optional	Array of RecognitionResult	Label result information list of the identification class. Note: **This field may return null, indicating that there is no valid value available.**
request_id	Optional	String	Request ID
seq	Optional	Number	Audio sequence
speaker_results	Optional	Array of AudioResultDetailSpeakerResult	Speaker identification result in the audio. Note: **This field may return null, indicating that there is no valid value available.**
sub_label	Optional	String	Sub-label name. If the sub-label is not matched, an empty string will be returned.
suggestion	Mandatory	string	Recommended value. Valid values: Block: content filtering

			Review: pending re-moderation Pass: normal
text_results	Optional	Array of TextResult	Dialog content moderation result in the audio. Note: **This field may return null, indicating that there is no valid value available.**
data	Mandatory	Data	Speech recognition result.

AudioResultDetailLanguageResult

Minority language detection result in the audio.

Name	Type	Description
Label	String	This field is used to return the corresponding language type information. Note: **This field may return null, indicating that there is no valid value available.**
Score	Integer	This parameter is used to return the confidence of the current label. Value range: 0 (lowest confidence) to100 (highest confidence). A larger value indicates a higher possibility that the audio belongs to the current returned language label. Note: **This field may return null, indicating that there is no valid value available.**
StartTime	Float	This parameter is used to return the start time of the segment corresponding to the specified language label within the audio file, in the unit of seconds. Note: **This field may return null, indicating that there is no valid value available.**
EndTime	Float	This parameter is used to return the end time of the segment corresponding to the specified language label within the audio file, in the unit of seconds. Note: **This field may return null, indicating that there is no valid value available.**

MoanResult

Vulgar content moderation result.

Name	Type	Description
Label	String	The value is fixed at Moan (moan/panting). If there is no **MoanResult** in the callback result for the audio, there are no relevant violations about moan/panting in this audio.

		Note: **This field may return null, indicating that there is no valid value available.**
Score	Integer	The confidence determined by the machine for the current category. Value range: 0 to 100. A higher score indicates a higher possibility that it belongs to the current category. (Example: Moan 99 indicates that the sample has a high possibility of belonging to the moan/panting category)
Suggestion	String	You are advised to perform operations after obtaining the judgment result. Recommended value. Valid values: Block: Blocking is recommended. Review: Re-moderation is recommended. Pass: Pass is recommended.
StartTime	Float	Violation event start time, in the unit of seconds (s).
EndTime	Float	Violation event end time, in the unit of seconds (s).
SubLabel	String	This field is used to return the secondary label under the current label (Label). Note: **This field may return null, indicating that there is no valid value available.**

RecognitionResult

Result information list of the recognition category label.

Name	Type	Description
Label	String	Possible values include: Teenager, Gender Note: **This field may return null, indicating that there is no valid value available.**
Tags	Array of Label	Identifying Tag List Note: **This field may return null, indicating that there is no valid value available.**

AudioResultDetailSpeakerResult

Returned speaker recognition result in the audio.

Name	Type	Description
Label	String	This field is used to return the content types requiring detection for the result. Note: **This field may return null, indicating that there is no valid value available.**
Score	Integer	This field is used to return the confidence level of the moaning detection. Value

		range: 0 (lowest confidence) to 100 (highest confidence). A larger value indicates a higher possibility that the audio is the speaker's voice print. Note: **This field may return null, indicating that there is no valid value available.**
StartTime	Float	This field is used to return the start time of the corresponding speaker's segment within the audio file, in the unit of seconds. Note: **This field may return null, indicating that there is no valid value available.**
EndTime	Float	This field is used to return the end time of the corresponding speaker's segment within the audio file, in the unit of seconds. Note: **This field may return null, indicating that there is no valid value available.**

TextResult

Content moderation result in the audio.

Name	Type	Description
Label	String	Malicious label: Normal: Normal Porn: Porn Abuse: Abuse Ad: Advertisement Custom: Custom dictionary And other types of content that are offensive, unsafe or inappropriate. If there is no **TextResults** returned in the callback result for the audio, there are no relevant violations in this audio Note: **This field may return null, indicating that there is no valid value available.**
Keywords	Array of String	Keywords that are matched. If it is empty, the violation is determined by the model. Note: **This field may return null, indicating that there is no valid value available.**
LibId	String	Library identifier of the matched keyword library Note: **This field may return null, indicating that there is no valid value available.**
LibName	String	Name of the matched keyword library Note: **This field may return null, indicating that there is no valid value available.**
Score	Integer	The confidence determined by the machine for the current category. Value

		range: 0 to 100. A higher score indicates a higher possibility that it belongs to the current category. (Example: Porn 99 indicates that the sample has an extremely high possibility of being pornographic.) Note: **This field may return null, indicating that there is no valid value available.**
Suggestion	String	You are advised to perform operations after obtaining the judgment result. Recommended value. Valid values: Block: Blocking is recommended. Review: Re-moderation is recommended. Pass: Pass is recommended. Note: **This field may return null, indicating that there is no valid value available.**
LibType	Integer	Type of custom dictionary. Information related to the custom dictionary can be viewed after you log in to the console. Custom Block and Allow Library Custom Library
SubLabel	String	This field is used to return the secondary label under the current label (Label). Note: **This field may return null, indicating that there is no valid value available.**

Data

Name	Type	Description
asr_tmp_full_results	Array of AsrTmpFullResults	Details of the audio detection result, which may be empty.

AsrTmpFullResults

Details of the audio detection results.

Name	Type	Description
appearing_point	Array of Number	Time point of occurrence.
confidence	Number	Confidence level.
id	String	Audio text.
periods	String	Time period.
url	String	Audio URL.

Tag

Recognition label list

Name	Type	Description
Name	String	The specific name is determined based on the Label field: When the Label field is Teenager, possible values for Name include: Teenager When the Label field is Gender, possible values for Name include: Male, Female Note: **This field may return null, indicating that there is no valid value available.**
Score	Integer	Confidence score: 0 to 100. A larger value indicates a greater confidence. Note: **This field may return null, indicating that there is no valid value available.**
StartTime	Float	Recognition start offset time, unit: milliseconds Note: **This field may return null, indicating that there is no valid value available.**
EndTime	Float	Recognition end offset time, unit: milliseconds Note: **This field may return null, indicating that there is no valid value available.**

Callback Message Example
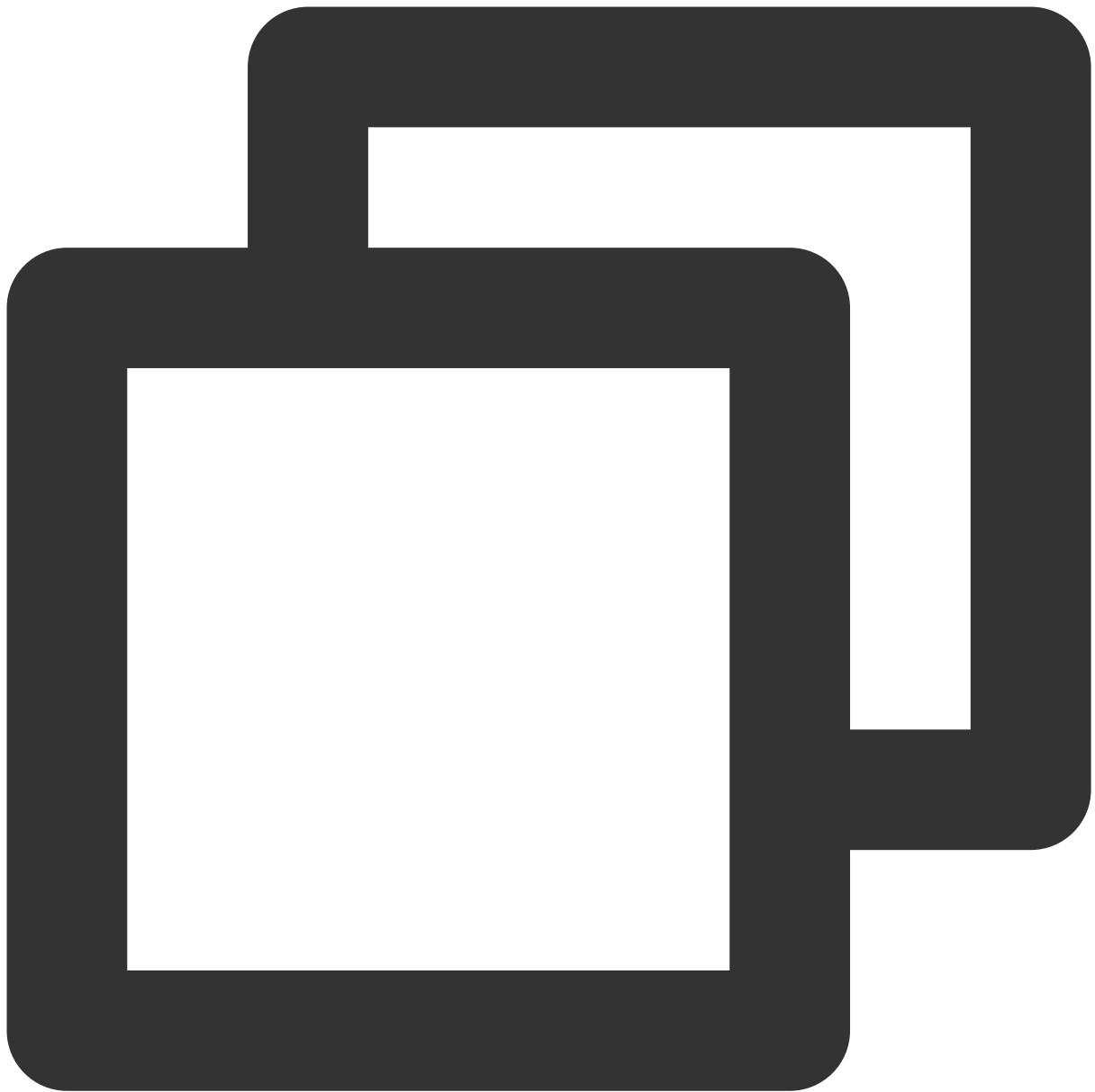
```
{
    "appid": xxx08,
    "asr_text": "",
    "cdn_url": "",
    "channel_id": "xxxun01",
    "data": {
        "asr_tmp_full_results": [
            {
                "appearing_point": [
                    0.0230000004470348,
                    15.02299976348877
```

```
            ],
            "confidence": 100,
            "create_time": 1685929588,
            "id": "",
            "periods": "00:00:00-00:00:15",
            "url": "https://xxx.Audit-09-46-27.wav"
        }
    ]
},
"domain": "xxx.cn",
"duration": 10,
"event_type": 315,
"interface": "general_callback",
"label": "Moan",
"language_results": [ ],
"moan_results": [
    {
        "EndTime": 15,
        "Label": "Moan",
        "Score": 99,
        "StartTime": 0,
        "SubLabel": "PornMoan",
        "Suggestion": "Block"
    }
],
"path": "live",
"recognition_results": [ ],
"request_id": "xxx594-4f4d-a5d0-99cce8b750b4",
"seq": 3232590095,
"speaker_results": [ ],
"status": 2,
"stream_id": "xxxn01",
"sub_label": "PornMoan",
"suggestion": "Block",
"task_id": xxx36881,
"text_results": [ ]
```

Push Error Event Notifications

Last updated：2023-10-08 15:08:36

Push error callbacks notify you of the details of push errors. You need to configure a callback address in the CSS console, and CSS will send push error callbacks to the server you configured.
This document describes the fields in a callback notification sent by CSS after a push error occurs.

Must-Knows

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

Push Error Callback Parameters

Event type

Event Type	Parameter Value
Push errors	event_type = 321

Common callback parameters

Parameter	Type	Description
appid	int	The user's App ID.
stream_id	string	The stream ID.
data_time	int	The callback time (ms).
report_interval	int	The reporting interval (ms) when a push error occurs.
abnormal_event	json	The push error details.

`abnormal_event` parameters

Parameter	Type	Description
type	int	The error type.

count	int	The number of times the error occurred between two reports (within the reporting interval).
detail	json	desc: The error description. occur_time: The time when the error occurred.
type_desc_cn	string	The error description in Chinese.
type_desc_en	string	The error description in English.

Error types

Type	Description
1	The video timestamp moved backwards.
2	The audio timestamp moved backwards.
3	The video timestamp increased notably (by more than one second).
4	The audio timestamp increased notably (by more than one second).
5	Chunk size too big (bigger than 8,192).
6	Two consecutive video frames arrived late (by longer than three seconds).
7	Two consecutive audio frames arrived late (by longer than three seconds).
8	The video codec changed.
9	The audio codec changed.
10	No codec header before a video frame arrived.
11	No codec header before an audio frame arrived.

Caution

Currently, you cannot configure callbacks for a specific type of push error. A push error callback includes the information of all push errors that occurred during the reporting interval. If no push errors occur, no callbacks will be sent.
A push error callback only collects data for push errors in the current reporting cycle. The system will not handle the errors.

Sample callback
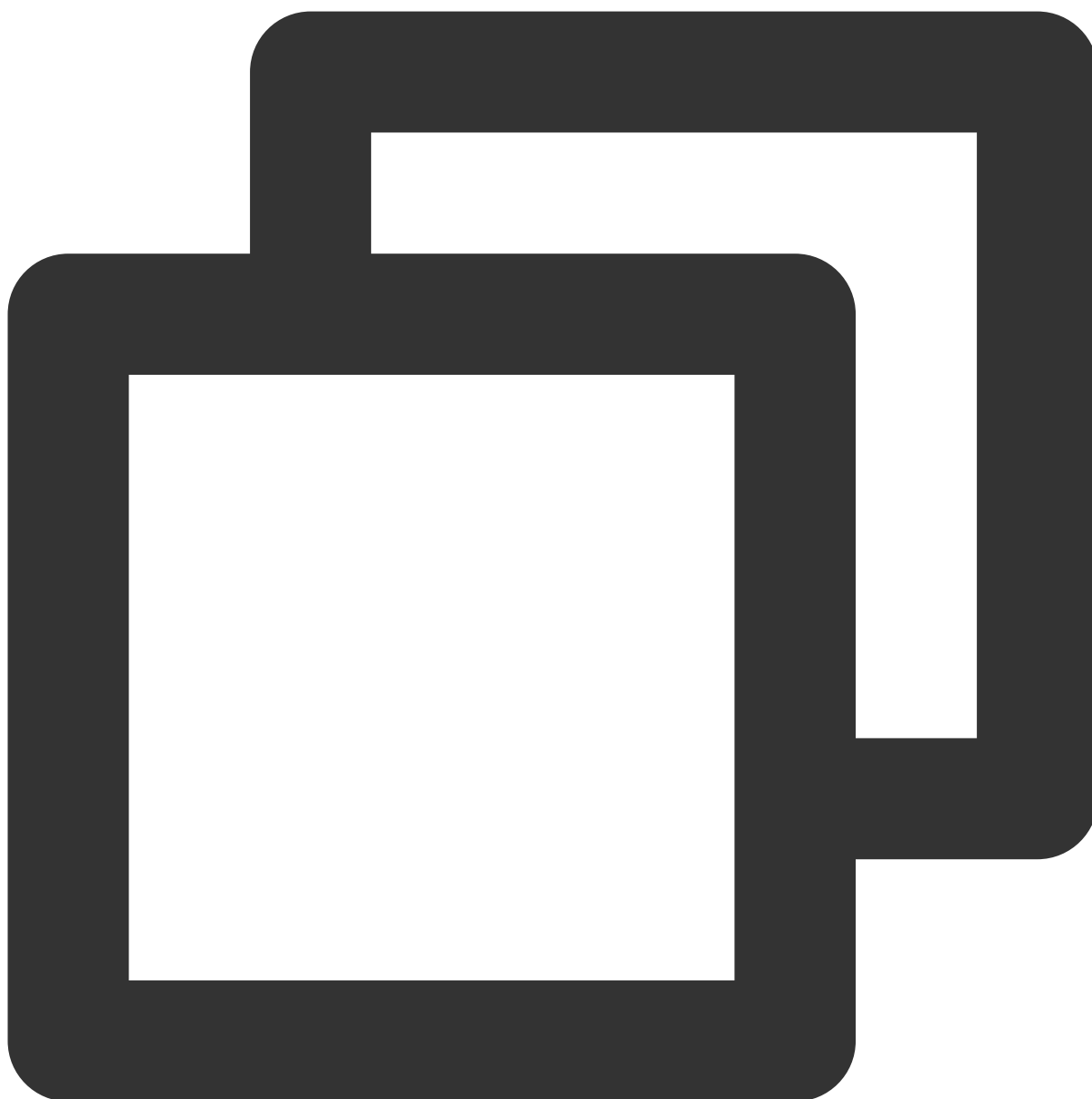
```
{
    "abnormal_event":[
        {
            "count":2,
            "detail":[
                {
                    "desc":"video frame arrive interval too long, interval=3046(mse
                    "occur_time":1670588070569
                },
                {
                    "desc":"video frame arrive interval too long, interval=2953(mse
```

```
                    "occur_time":1670588073522
                }
            ],
            "type":6,
            "type_desc_cn":" ",
            "type_desc_en":"video frame arrive interval bigger than 1000(ms)"
        },
        {
            "count":2,
            "detail":[
                {
                    "desc":"audio frame arrive interval too long, interval=3009(mse
                    "occur_time":1670588070532
                },
                {
                    "desc":"audio frame arrive interval too long, interval=2917(mse
                    "occur_time":1670588073486
                }
            ],
            "type":7,
            "type_desc_cn":" ",
            "type_desc_en":"audio frame arrive interval bigger than 1000(ms)"
        }
    ],
    "appid":0,
    "data_time":1670588074971,
    "domain":"xxxx.xxxxx.xxxx.xxxx",
    "event_type":321,
    "interface":"general_callback",
    "path":"xxxx",
    "report_interval":5000,
    "sequence":"000000000000000000",
    "stream_id":"xxxxxx",
    "stream_param":"txSecret=f5828cd4a8a09109304b060172fb3960&txTime=665982e4",
    "timeout":5000
}
```

Relay Event Notification

Last updated：2024-04-25 10:48:27

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

1. 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.
2. If the task has not reached its end time, continuous retries due to source or destination address unavailability, or automatic task migration due to machine abnormalities, will generate task end callbacks. Do not use these callbacks as the final task end callbacks.
3. If you need to determine whether the task is streaming properly, you can do so from the receiving end, such as using the stream interruption callback in Cloud Streaming Services or the stream status query API, etc.

Relay Event Parameters

Event type parameters

Event Type	Parameter Value
Relay	event_type = 314

Common callback parameters

Parameter	Type	Description
appId	int	User APPID
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
event_time	string	Event timestamp, for example: 1712893433

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

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

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

TaskExit - Callback of the task stop 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

    "stream_id": "testvod",

    "task_id": "118148",

    "to_url": "rtmp://xxx.livepush.myqcloud.com/live/testvod"
}
```
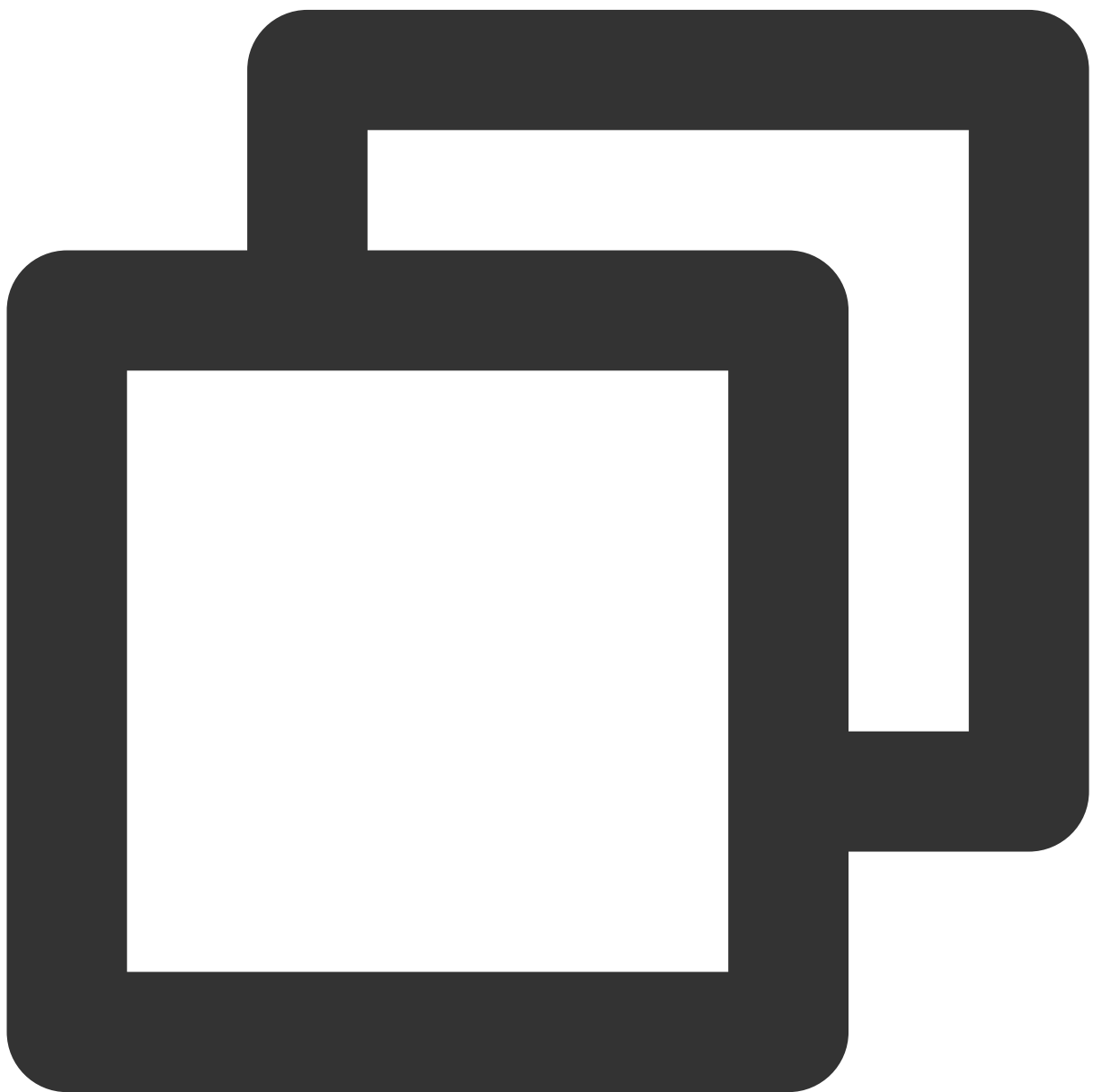
```
{
    "appid": 4,

    "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

    "product_name": "pullpush",

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

    "stream_id": "testvod",

    "task_id": "118145",

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

```
{
    "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-uploa

    "product_name": "pullpush",

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

    "stream_id": "testvod",

    "task_id": "118145",

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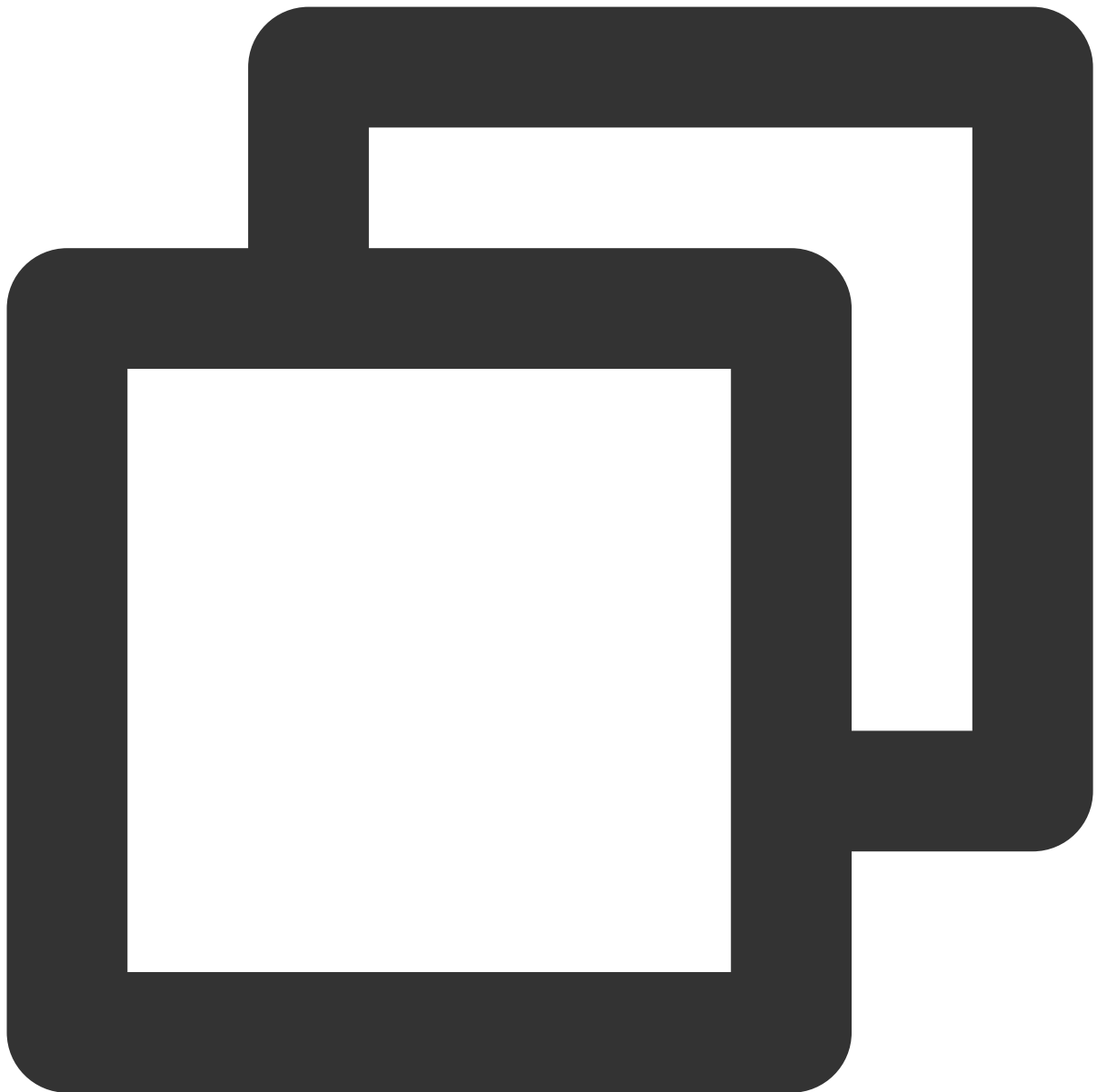
```
{

    "appid": 4,

    "callback_event": "TaskExit",

    "event_type": 314,

    "interface": "general_callback",

    "msg": "{\\"message\\":\\"write packet error.\\",\\"code\\":-22,\\"task_exit_ti
```

```
    "product_name": "pullpush",

    "source_urls": "[\\"http://yourURL.cn/live/normal_230753472*****21162358-upload
  }
```

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. For detailed directions, see Relay.

User Guides for Common Third-Party Tools
Push via OBS

Last updated：2024-02-26 17:59:59

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.
2. Enter the Address Generator page and perform the following configurations:
2.1 Select the URL type: **Push Address**.
2.2 Select the domain name you have added in **Domain Management**.
2.3 Enter an application name (`AppName`), which is used to distinguish applications under the same domain. The default value is `live` .
2.4 Enter a custom stream name (`StreamName`), such as `liveteststream` .
2.5 You need to select an encryption type according to your security needs and performance considerations. You can select either **MD5** or **SHA256** as your encryption type, and the default is **MD5**.
2.6 Select the expiration time of the address, such as `2024-02-07 16:47:04` .
3. 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.
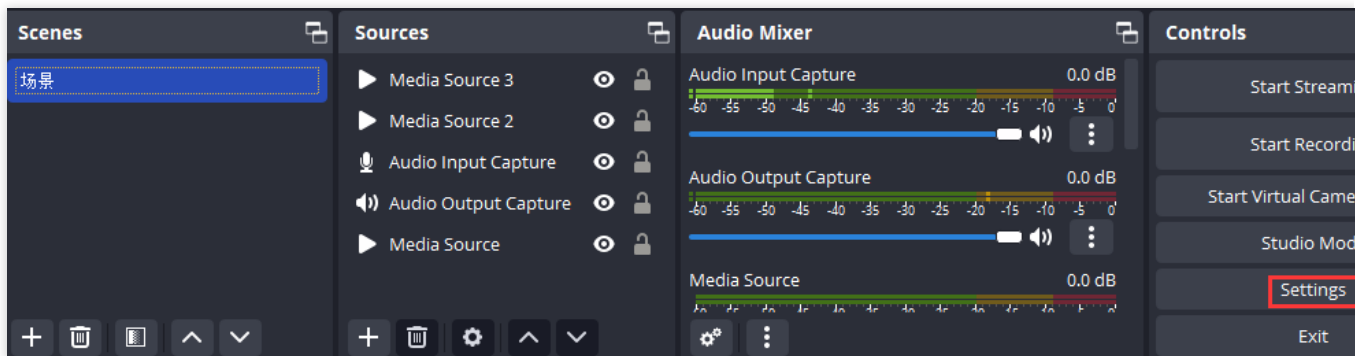
2. Click **Stream** and select **Custom** for **Service**.

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/`).

Streaming Key: cEnter the OBS push name (`StreamName?txSecret=xxxxx&txTime=liveteststream`).
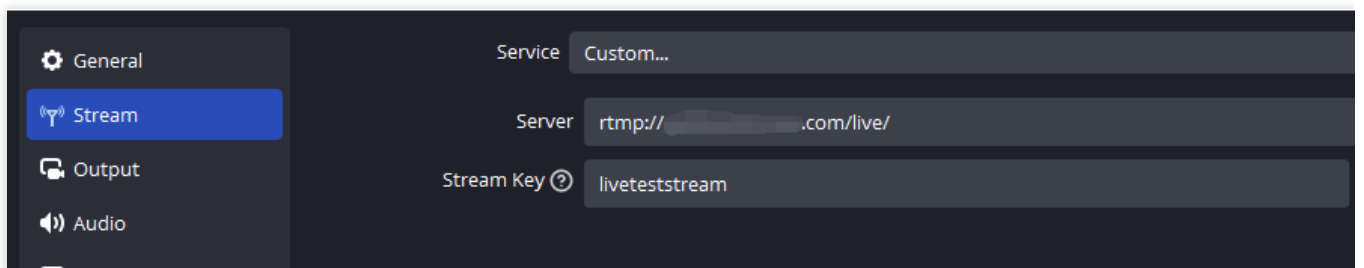
4. Click **OK** to save the information.

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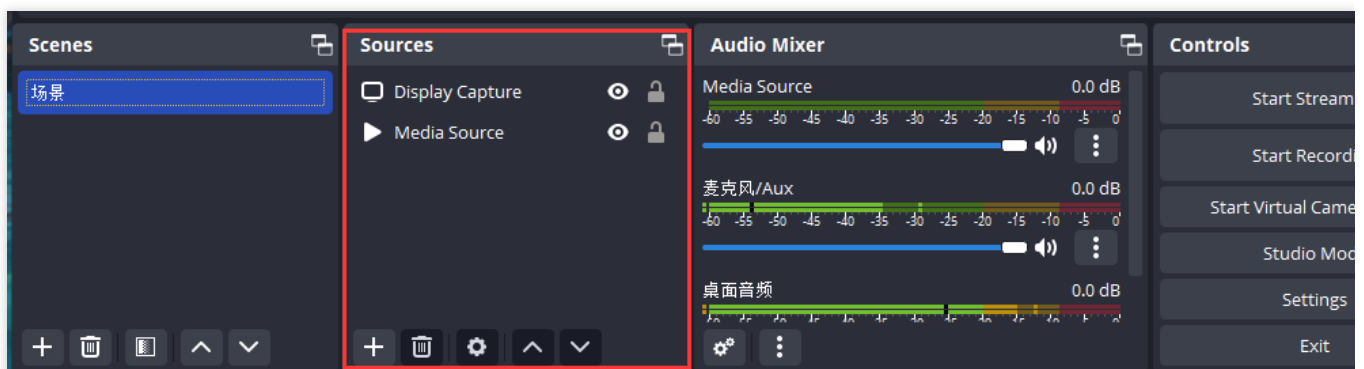
2. Click **+** and select a source that fits your needs, for example, **Display Capture**.

Common live streaming sources

Input Source	Description
VLC Video Source	Installing VLC to enable this source
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
Media Source	Publishing a local file
Text	Adding real-time text to your stream
Display Capture	Capturing and publishing your monitor in real time
Browser	Displaying URL content in a browser or opening a local html file
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)
Group	Placing sources in the same group to change visibility and lock status in batches

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.
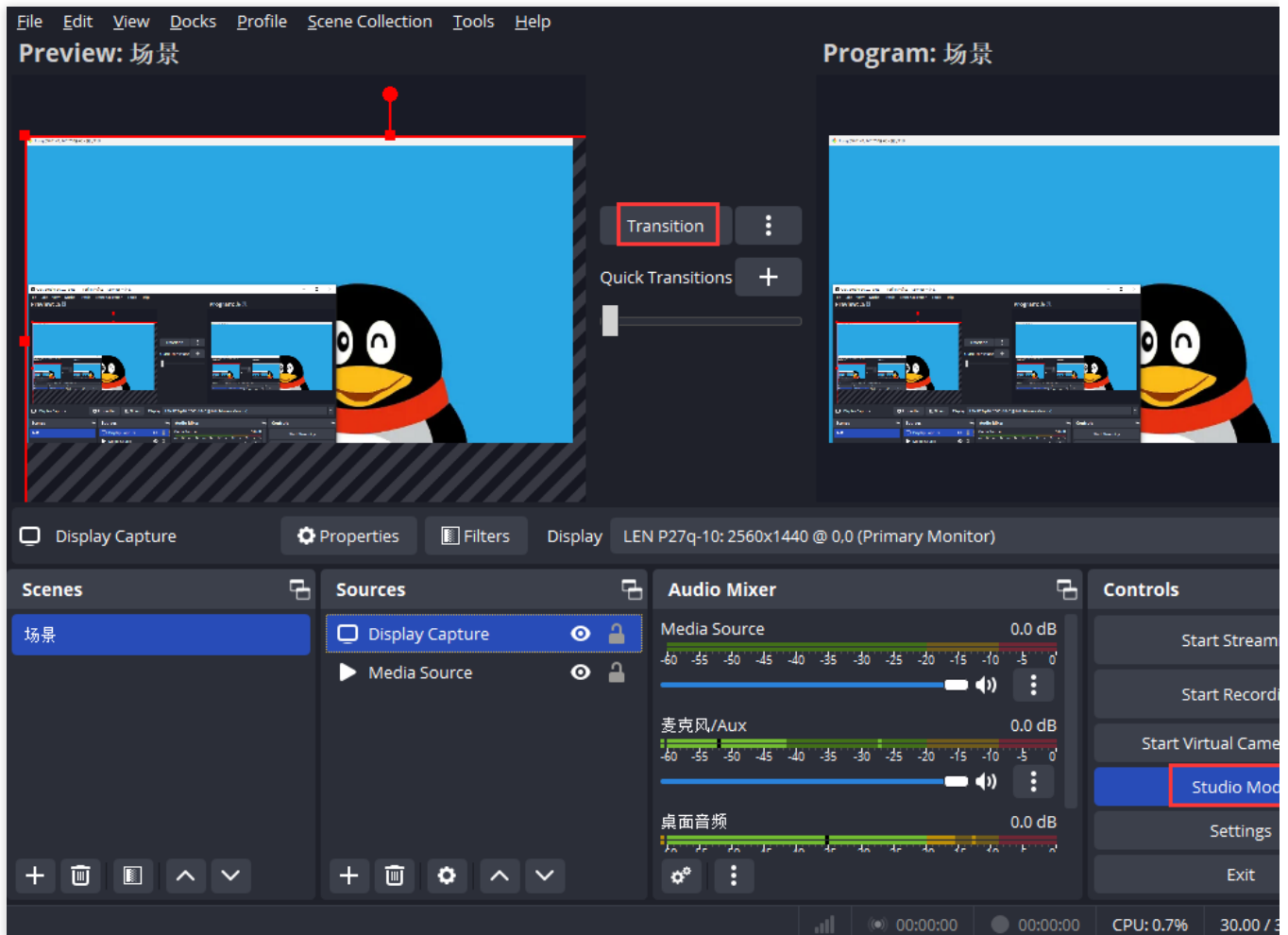
Step 4. Start streaming

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

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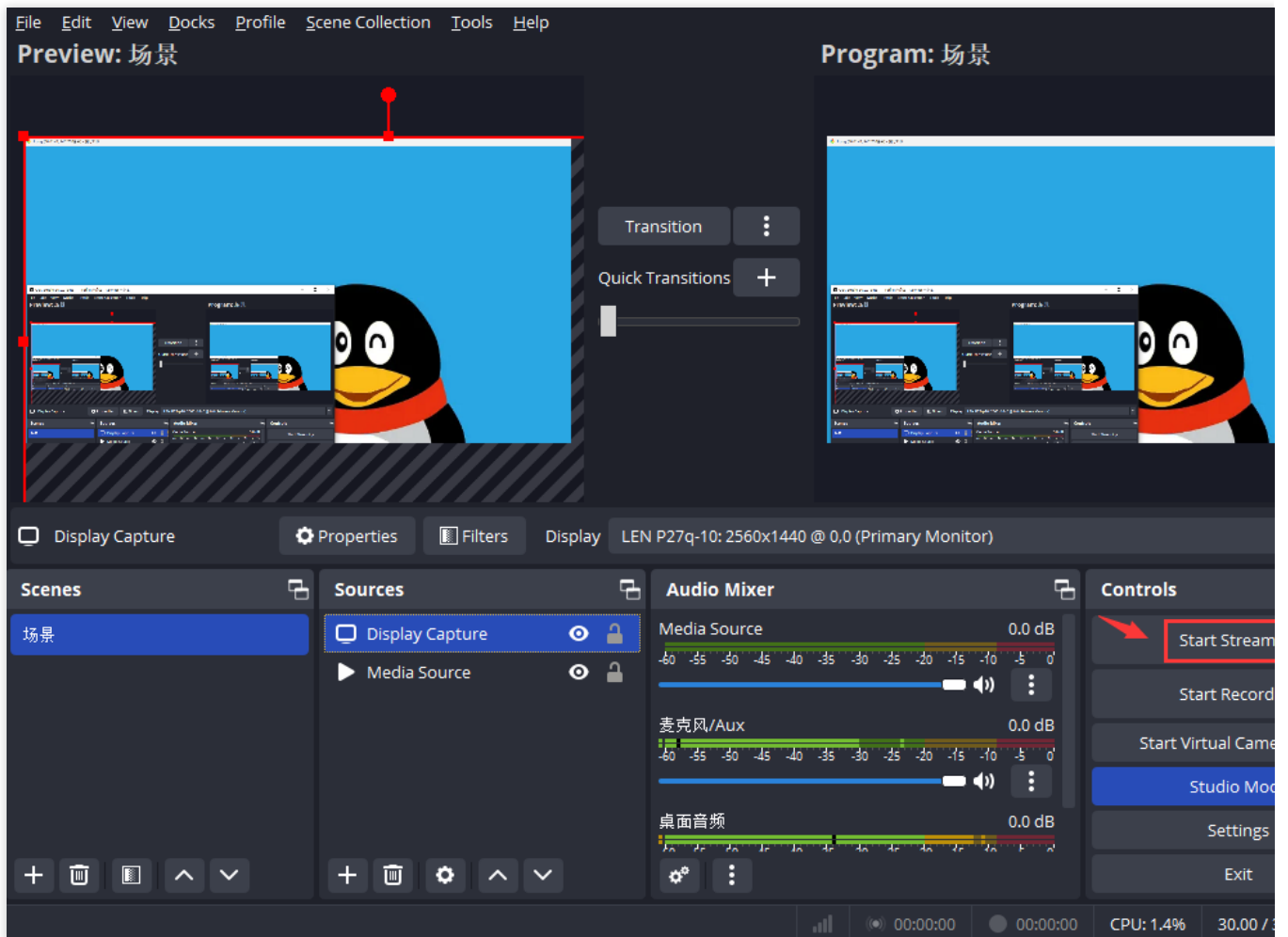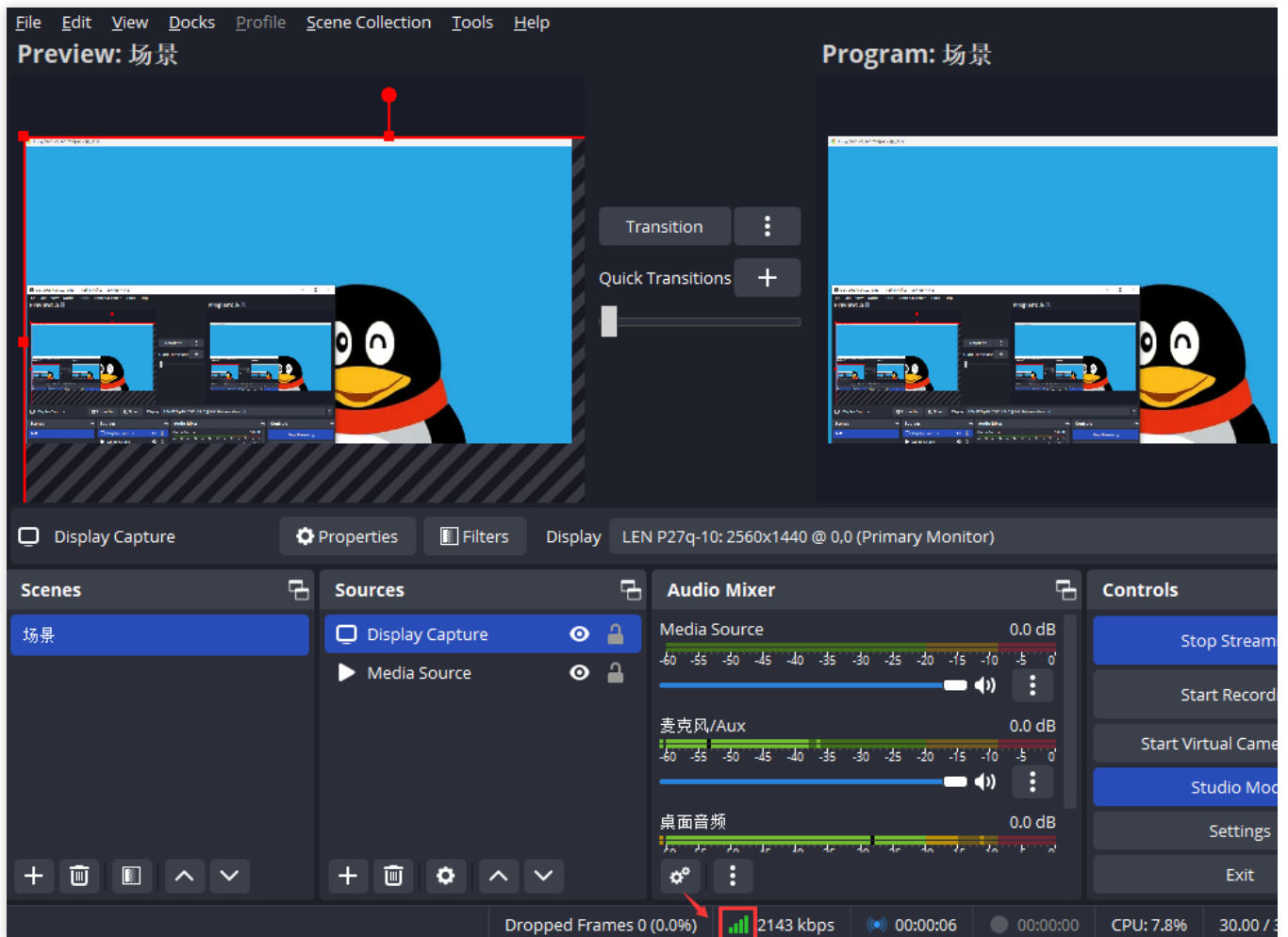
Note

When the
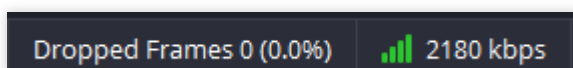
green light appears at the bottom, it signifies a successful stream push.

If you want to stop the stream push, simply click on **Stop Streaming**.

Other Push Settings

Streaming latency

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

2. Select the **Output Mode** as **Advanced** to configure settings such as **Keyframe Interval. Excessive keyframe interval (GOP)** can affect the Live Event Broadcasting experience. It is recommended to set the size to 2s. The procedure of setting is shown in the figure below:
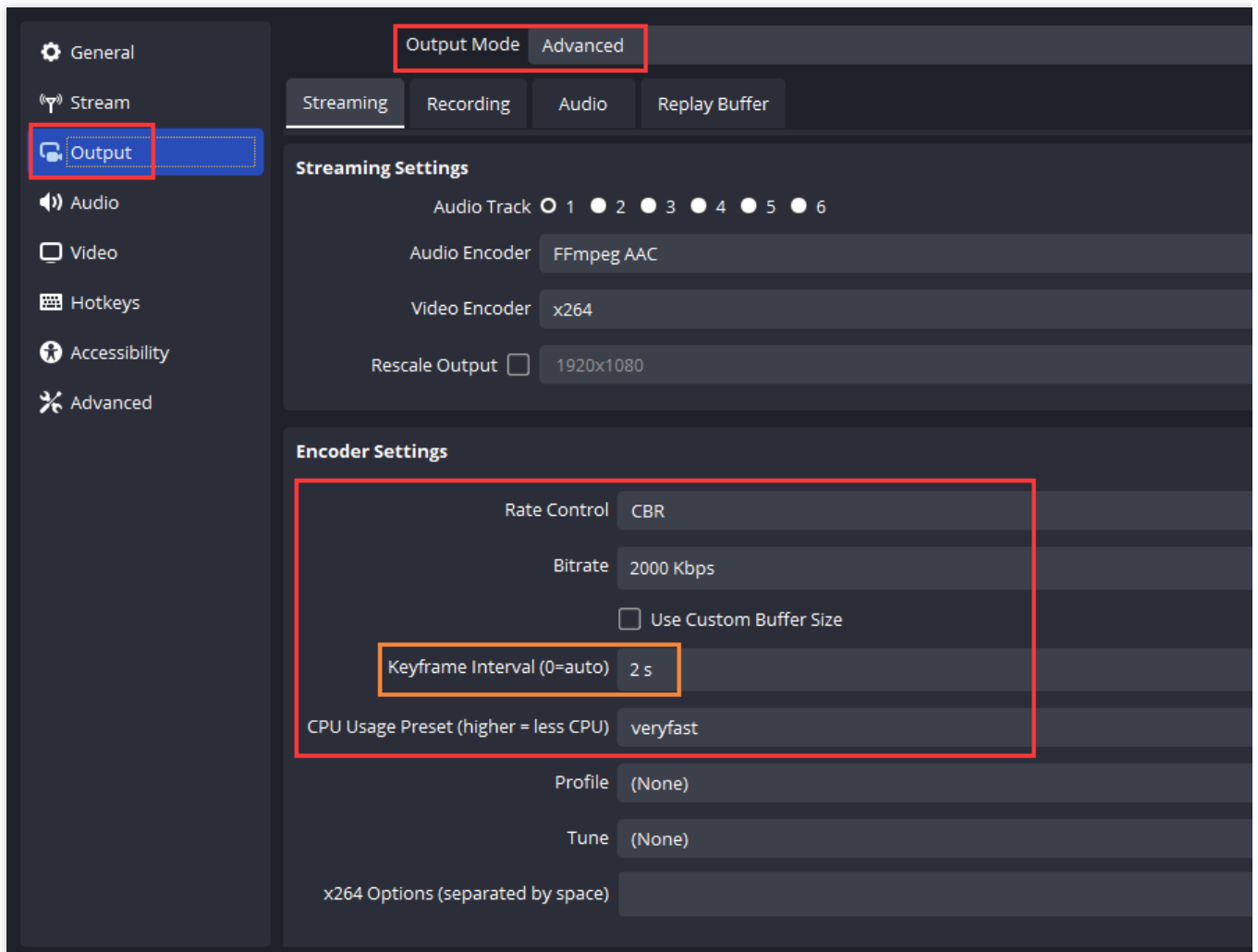
Removing B-frames in LEB

The LEB solution for web **does not support decoding or playing B-frames**. If a stream contains B-frames, the backend will remove them in transcoding, which will increase latency and **incur transcoding fees**. Please avoid pushing streams with B-frames or use streaming software such as OBS to remove them by adjusting the video encoding parameters. The figure below shows how to remove B-frames using OBS:
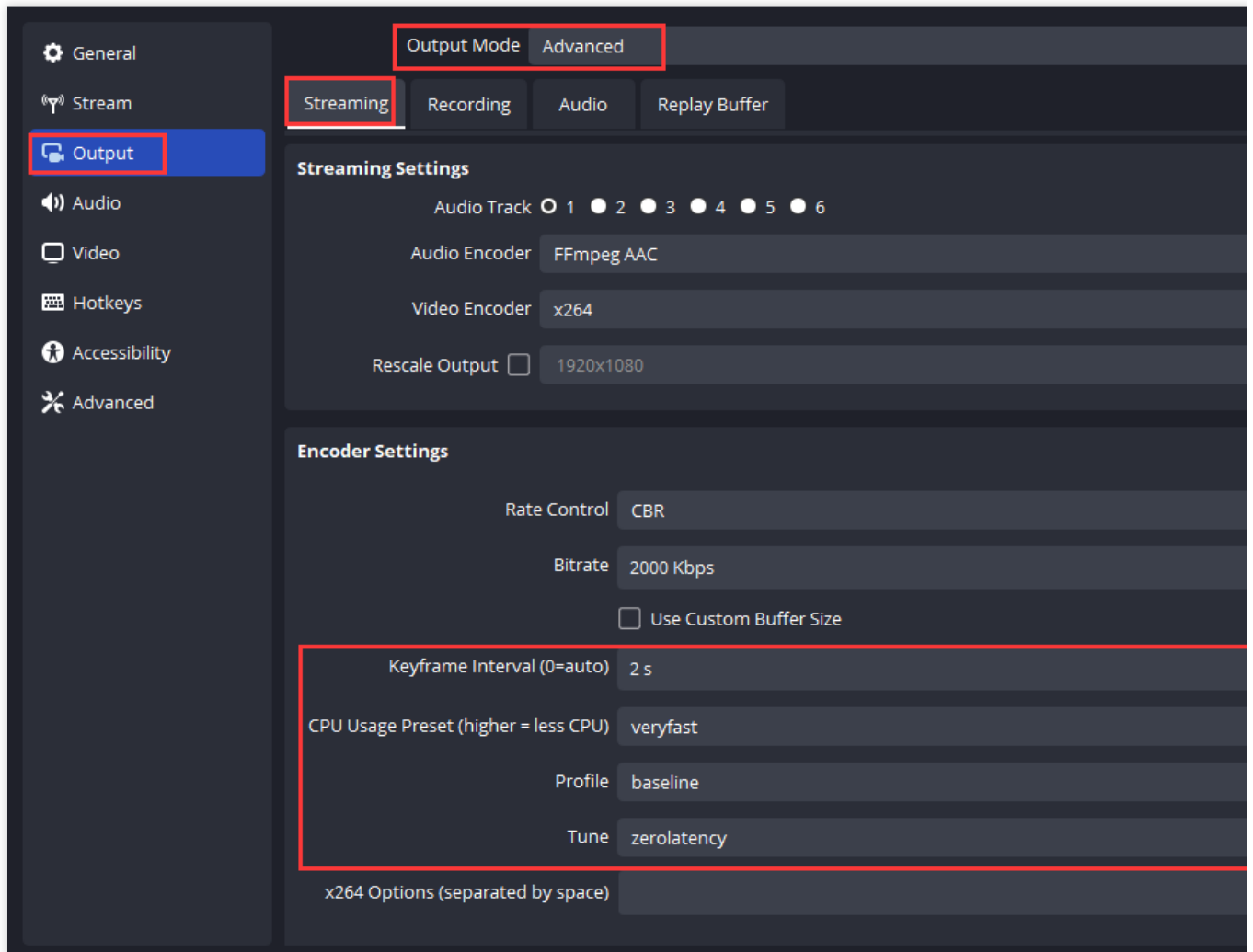
Local live recording

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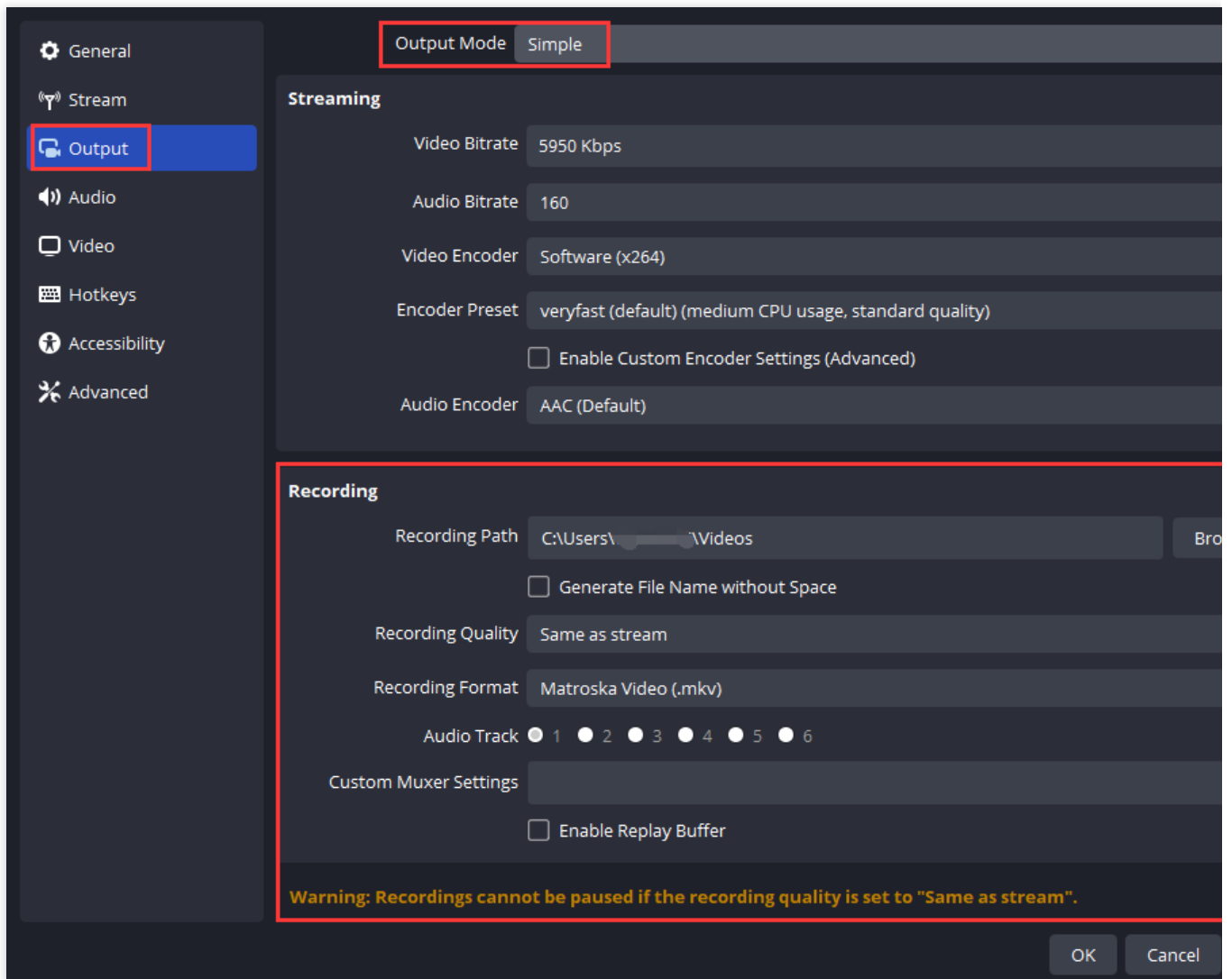
3. Click **Video** in the left sidebar to set the resolution and frame rate.

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.
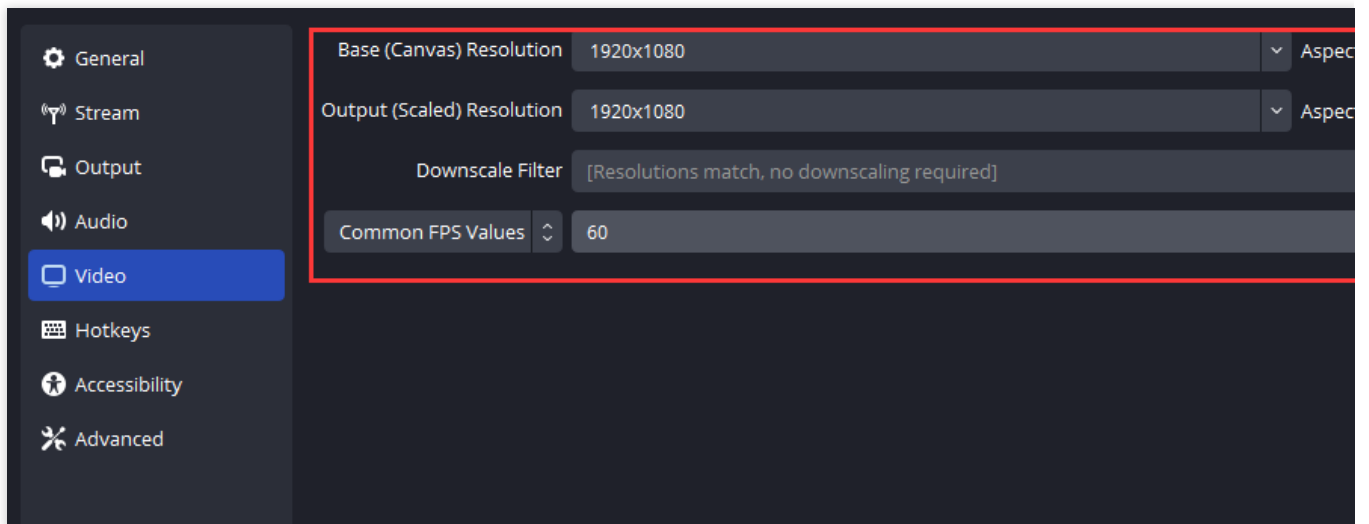
Transcoding

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**.
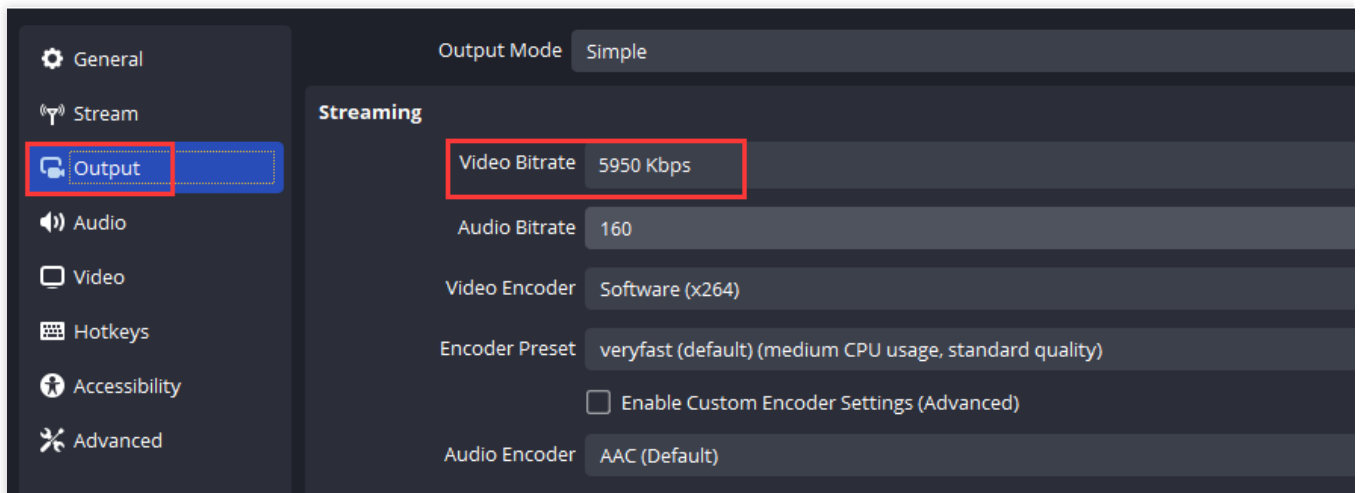
More

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.
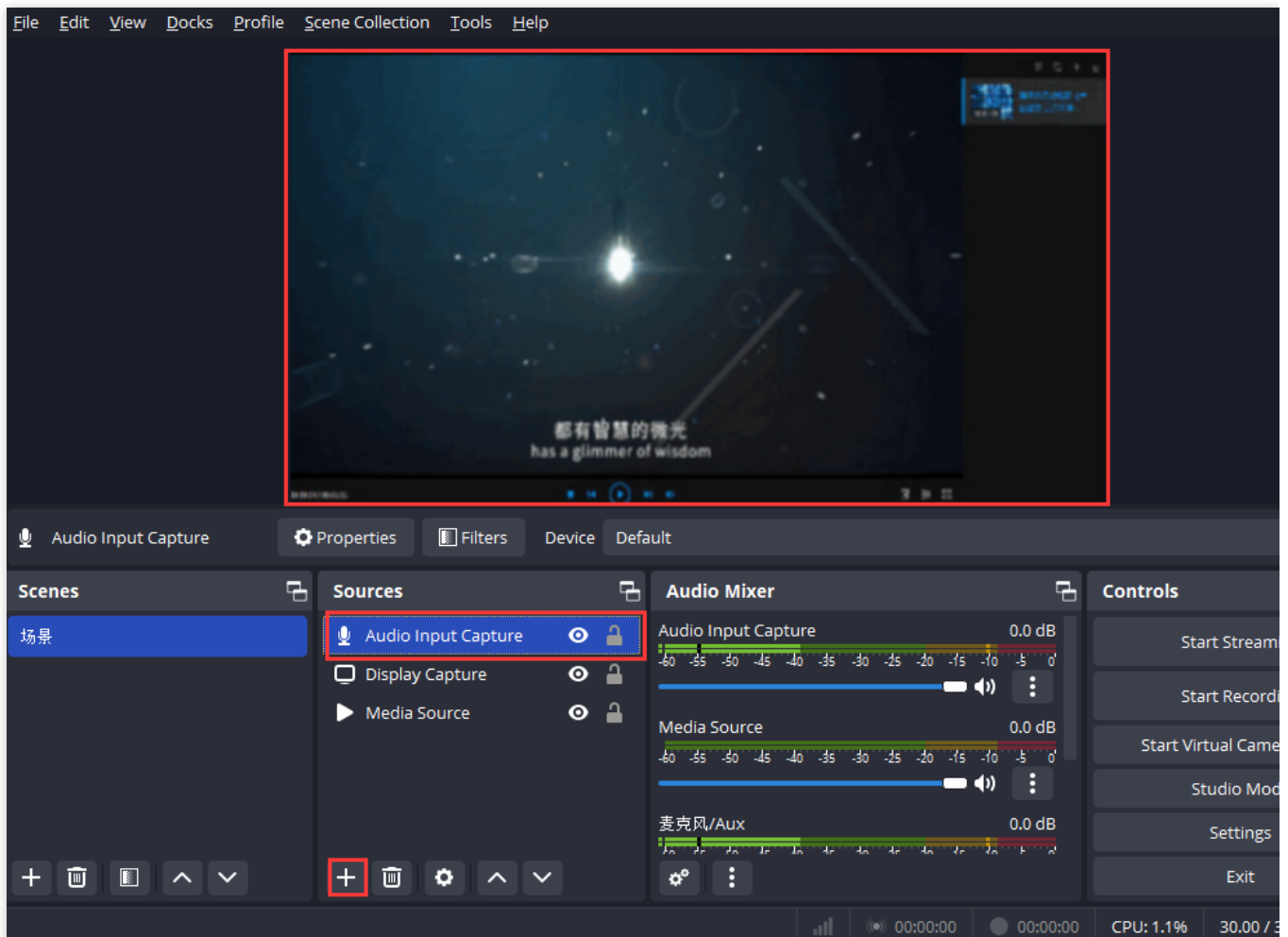
2. Go to **Controls > Settings > Video**.

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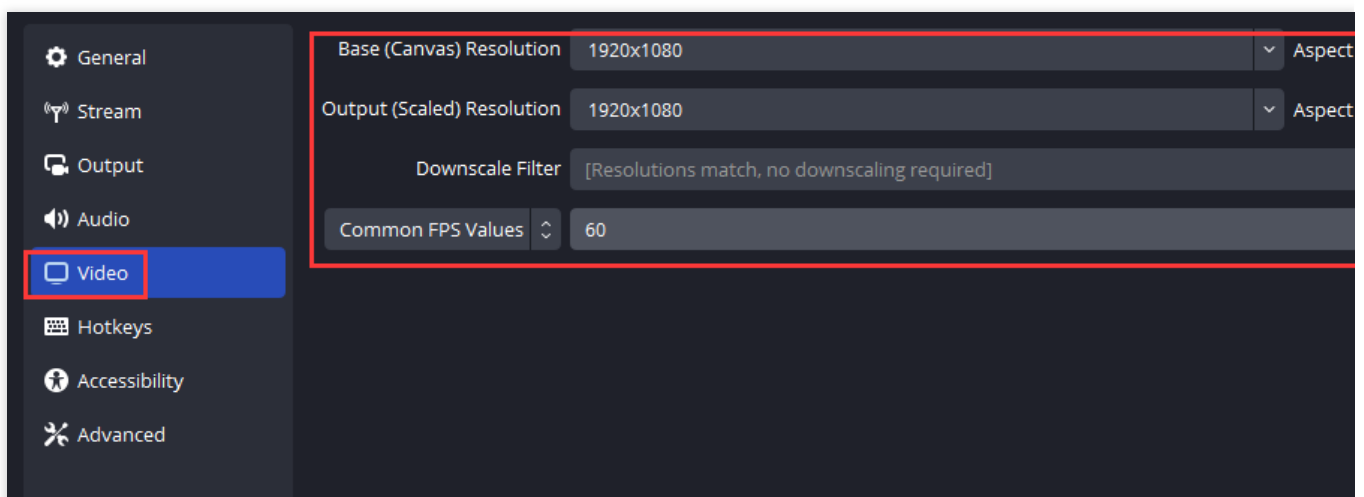

4. Click **Output** in the left sidebar, configure the output as shown below (set **Bitrate** to the minimum value), and click **OK**.
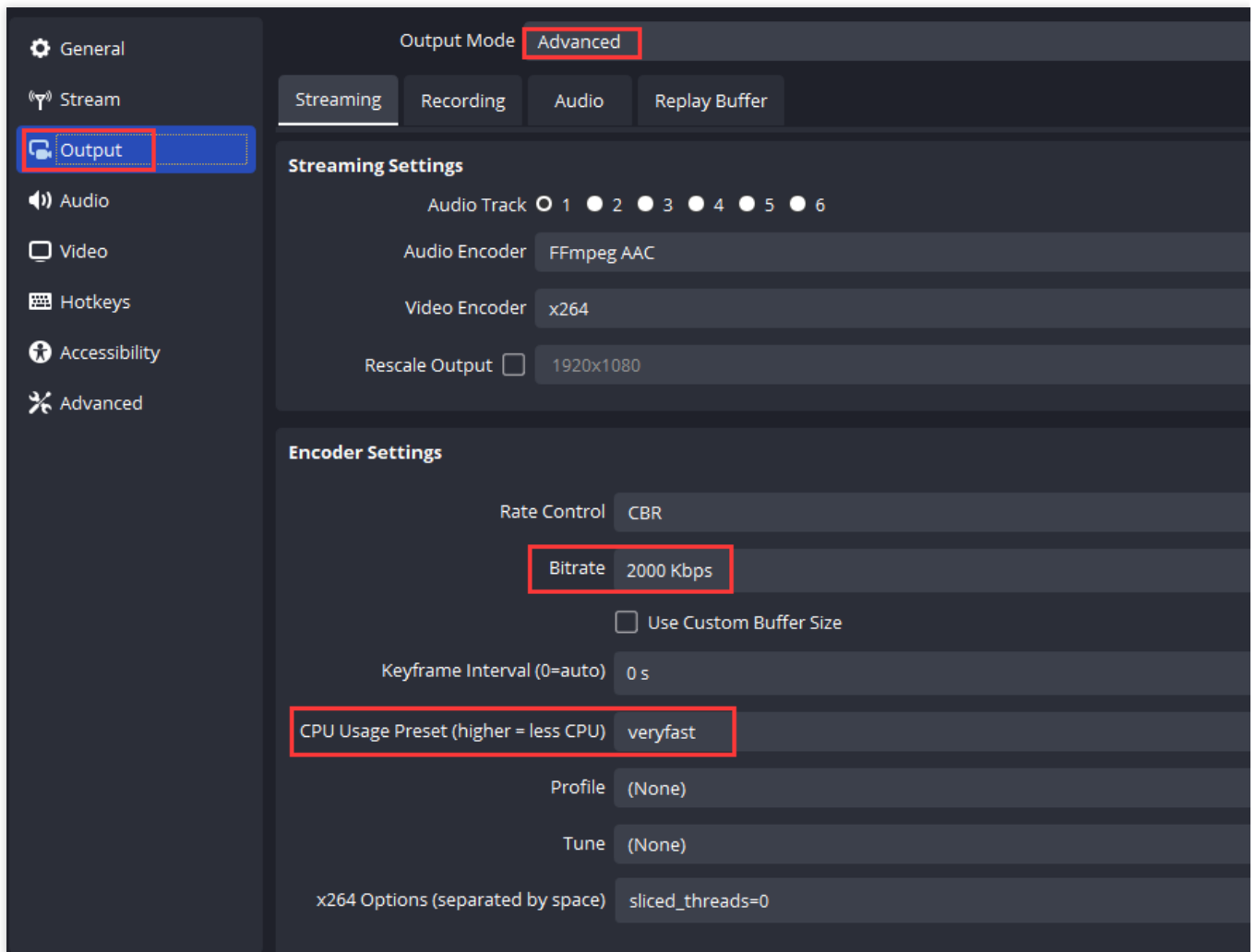
5. Start streaming as instructed in Configuring OBS for Push. The audience will hear audio, while the video will be a blank screen or an image. Because the video bitrate is set to the minimum value, the bandwidth usage is significantly lower than that of video push.

Video looping

1. Click **+** in **Sources** and select **Media Source**. In the pop-up window, choose a local file to stream, select **Loop**, and click **OK**.
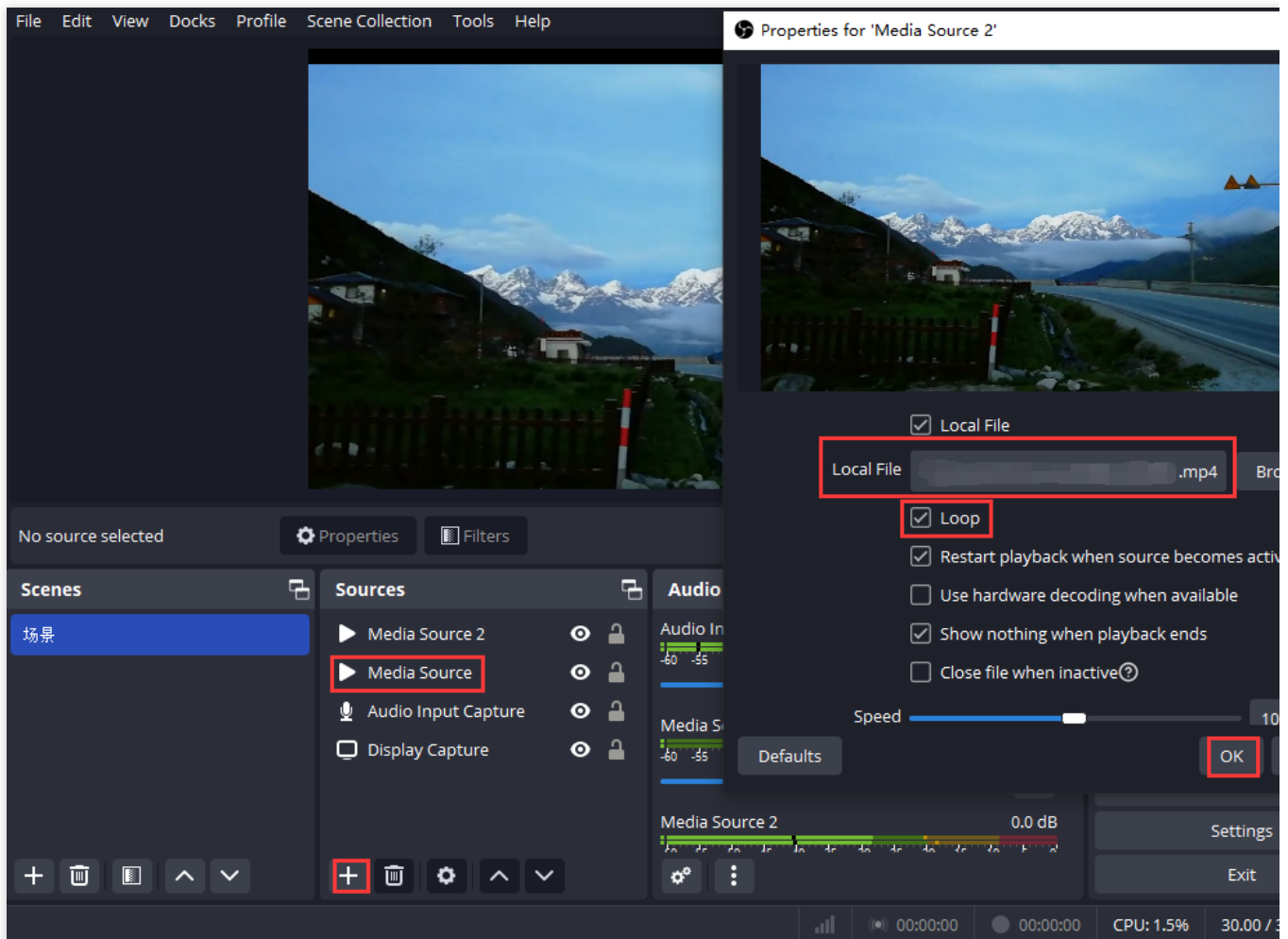
2. Set the **Server** and **Stream Key** as instructed in Configure the push URL.

Playback

After the push is finished, you can generate the playback address with the same StreamName as the push address.

You can verify whether the stream is a successful push by the following method of playback:

PC: Supports the use of the VLC Player for stream pulling.

Mobile: Supports playback by integrating Mobile Live Video Broadcasting.

Note

Mobile Live Video Broadcasting (MLVB) SDK is an extension of Cloud Streaming Services (CSS) in mobile scenarios.

Compared with CSS services mainly for cloud integration, MLVB SDK not only offers **a quick integration solution**

based on RTMP SDK, but also provides **a one-stop professional resolution** incorporating multiple cloud services,

including Live Video Broadcasting(LVB), Live Event Broadcasting(LEB), Video on Demand(VOD), Chat and Cloud

Object Storage(COS).

In fact, Live Event Broadcasting (LEB) is an extension of Live Video Broadcasting(LVB) in ultra-low latency playback

scenarios. It features lower latency than traditional streaming protocols and delivers superior playback experience with

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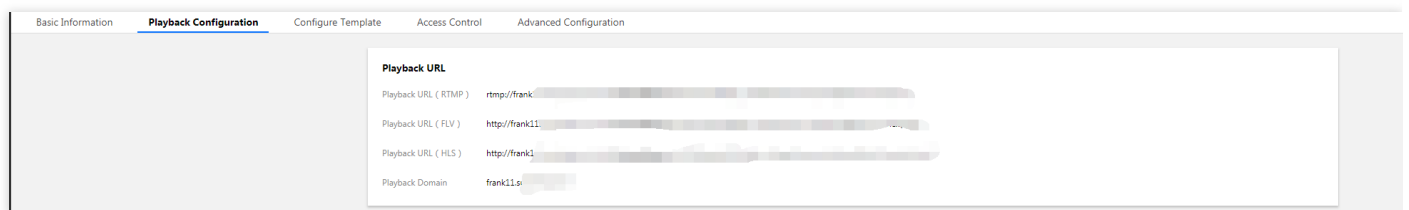
VLC Player

Last updated：2024-07-24 10:32:13

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.

2. Launch VLC and select **File** > **Open Network**.
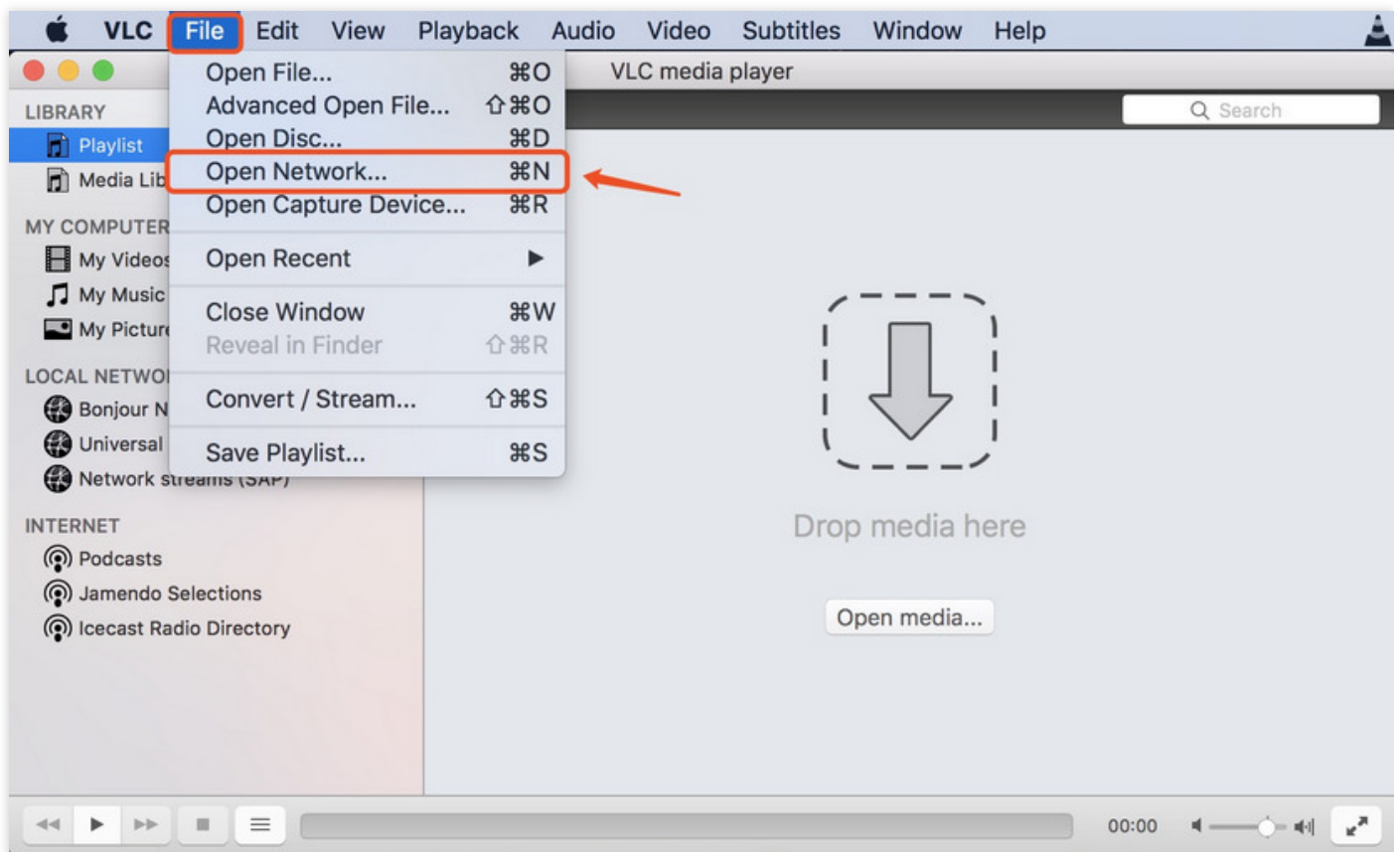
3. Enter the live stream playback address in the pop-up dialog box.
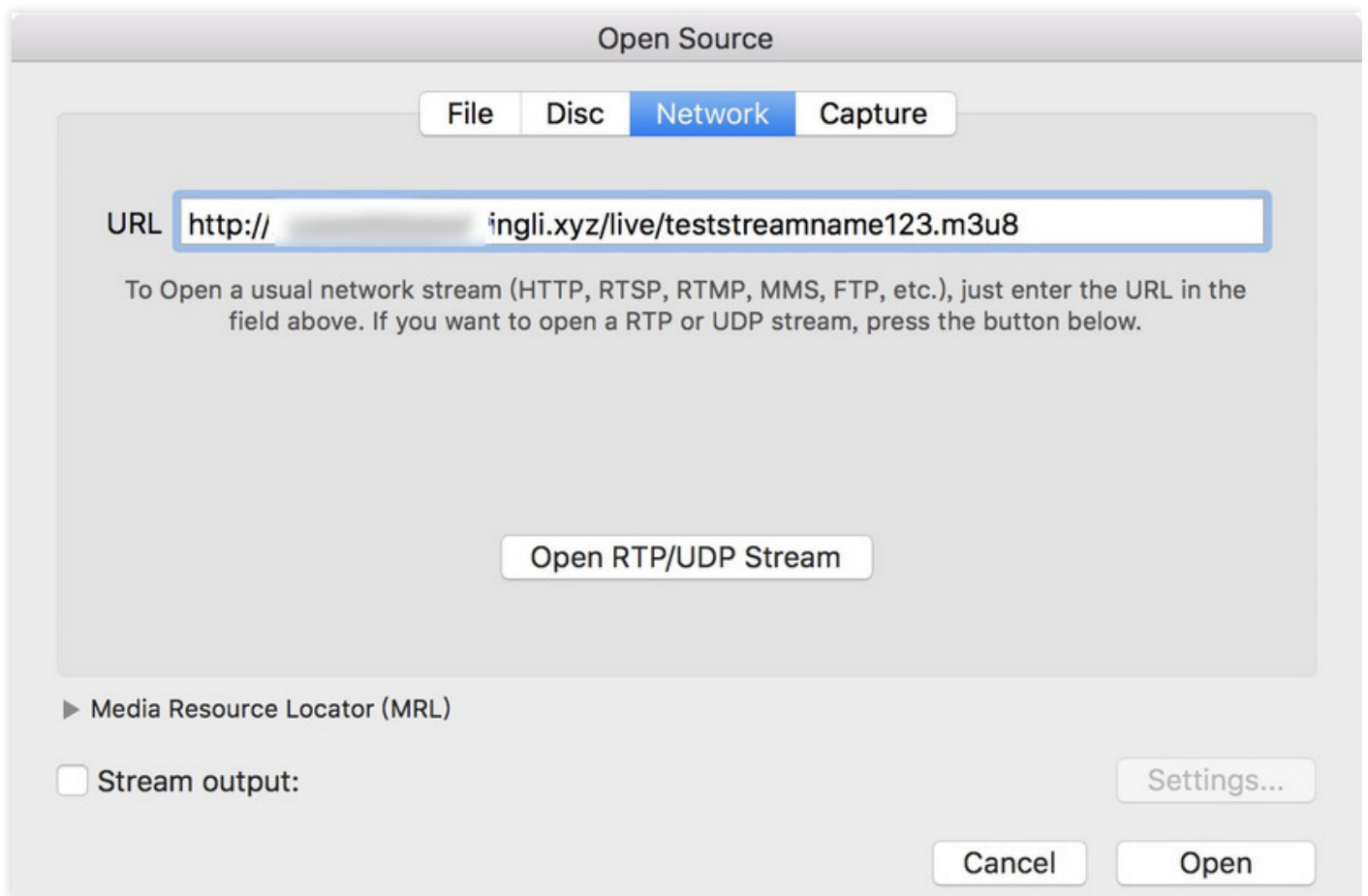
4. Then, click **Open** to confirm playback. If the pull from the playback address is normal, a playback window will pop up.