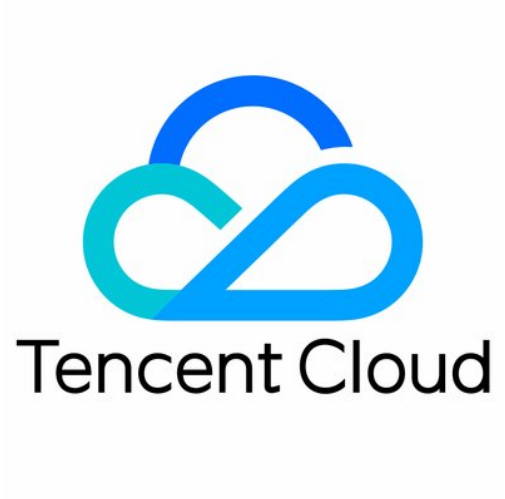
Mobile Live Video Broadcasting

Advanced Features

Product Documentation



Contents

Advanced Features
Setting Video Quality

Last updated：2022-06-14 12:41:46

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers an easy-to-understand API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github

Features

LiveAVSDK uses the `setVideoQuality` API provided by `V2TXLivePusher` to set video quality.

API definition

You can use `setVideoQuality` to set the resolution and resolution mode (landscape/portrait) of published video.

```
public abstract int setVideoQuality(V2TXLiveVideoEncoderParam param);
```

Parameters

Parameter	Type	Description
param	V2TXLiveVideoEncoderParam	Video encoding parameters.

- **Enumerated values of V2TXLiveVideoResolution:**

Value	Description
V2TXLiveVideoResolution160x160	Resolution: 160 × 160; bitrate: 100-150 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution270x270	Resolution: 270 × 270; bitrate: 200-300 Kbps; frame rate: 15 fps

V2TXLiveVideoResolution480x480	Resolution: 480 × 480; bitrate: 350-525 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution320x240	Resolution: 320 × 240; bitrate: 250-375 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution480x360	Resolution: 480 × 360; bitrate: 400-600 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution640x480	Resolution: 640 × 480; bitrate: 600-900 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution320x180	Resolution: 320 × 180; bitrate: 250-400 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution480x270	Resolution: 480 × 270; bitrate: 350-550 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution640x360	Resolution: 640 × 360; bitrate: 500-900 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution960x540	Resolution: 960 × 540; bitrate: 800-1500 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution1280x720	Resolution: 1280 × 720; bitrate: 1000-1800 Kbps; frame rate: 15 fps
V2TXLiveVideoResolution1920x1080	Resolution: 1920 × 1080; bitrate: 2500-3000 Kbps; frame rate: 15 fps

- **Enumerated values of V2TXLiveVideoResolutionMode:**

Value	Description
V2TXLiveVideoResolutionModeLandscape	Resolution in landscape mode: V2TXLiveVideoResolution640_360 + V2TXLiveVideoResolutionModeLandscape = 640 × 360
V2TXLiveVideoResolutionModePortrait	Resolution in portrait mode: V2TXLiveVideoResolution640_360 + V2TXLiveVideoResolutionModePortrait = 360 x 640

Recommended settings

Application Scenario	resolution	resolutionMode
Live showroom	• V2TXLiveVideoResolution960x540 • V2TXLiveVideoResolution1280x720	Landscape or portrait
Live game streaming	V2TXLiveVideoResolution1280x720	Landscape or portrait
Mic connect (primary-stream image)	V2TXLiveVideoResolution640x360	Landscape or portrait
Mic connect (small image)	V2TXLiveVideoResolution480x360	Landscape or portrait
Blu-ray streaming	V2TXLiveVideoResolution1920x1080	Landscape or portrait

Note

For smoother mic connect experience, after mic connect starts, please call `setVideoQuality()` to set the host's resolution to `V2TXLiveVideoResolution640x360` and the mic-connecting audience's resolution to `V2TXLiveVideoResolution480x360`. After mic connect ends, you can call `setVideoQuality()` again to set the resolutions to previous values.

FAQs

1. Why is the video delivered to audience not as clear as that watched by the host?

The video watched by the host is the raw data captured by the camera after pre-processing (beauty filter application, mirroring, clipping, etc.) and therefore is of high quality. However, the video watched by audience has been compressed and then decoded by the codec. Encoding compromises video quality (the lower the target resolution, the more the video is compressed), which is why the video delivered to audience is not as clear as that watched by the host.

2. Why does `V2TXLivePusher` sometimes publish video at a resolution of 368 × 640 or 544 × 960?

If you enable hardware acceleration, the video published may have atypical resolutions such as 368 × 640 or 544 × 960. This is because some hardware encoders do not allow resolutions that aren't a multiple of 16. You can change the fill mode of the player to get rid of black bars.

Custom Capturing and Rendering

Last updated：2022-06-14 12:41:46

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers an easy-to-understand API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github

Customizing Video to Publish

iOS

- Method 1: modifying OpenGL textures
- Method 2: capturing data by yourself

If you need to process video by yourself (for example, add subtitles) but want to leave the rest of the process to LiteAVSDK, follow the steps below.

1. Call enableCustomVideoProcess of `V2TXLivePusher` to enable custom video processing, so that you will receive the callback of video data.
2. There are two cases for image processing.
   -**The beauty filter component generates new textures.**
   If the beauty filter component you use generates a new texture frame (for the processed image) during image processing, please set `dstFrame.textureId` to a new texture ID in the callback API.

```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:(V2
TXLiveVideoFrame _Nonnull)dstFrame
{
GLuint dstTextureId = renderItemWithTexture(srcFrame.textureId, srcFrame.width,
srcFrame.height);
dstFrame.textureId = dstTextureId;
}
```

- **The beauty filter component does not generate new textures.**
  If the third-party beauty filter component you use does not generate new textures and you need to manually set an input texture and an output texture for the component, consider the following scheme:

```
- (void) onProcessVideoFrame:(V2TXLiveVideoFrame _Nonnull)srcFrame dstFrame:
(V2TXLiveVideoFrame _Nonnull)dstFrame
{
thirdparty_process(srcFrame.textureId, srcFrame.width, srcFrame.height, dstFr
ame.textureId);
}
```

3. You need some basic knowledge of OpenGL to deal with texture data and should refrain from high volume computing. This is because `onProcessVideoFrame` is called at the same frequency as the frame rate, and sophisticated processing may cause GPU overheating.

Android

- Method 1: modifying OpenGL textures
- Method 2: capturing data by yourself

If you need to process video by yourself (for example, add subtitles) but want to leave the rest of the process to LiteAVSDK, follow the steps below.

1. Call enableCustomVideoProcess of `V2TXLivePusher` to enable custom video processing, so that you will receive the callback of video data.
2. There are two cases for image processing.
   - **The beauty filter component generates new textures.**
     If the beauty filter component you use generates a new texture frame (for the processed image) during image processing, please set `dstFrame.textureId` to a new texture ID in the callback API.

```
private class MyPusherObserver extends V2TXLivePusherObserver {
@Override
public void onGLContextCreated() {
mFURenderer.onSurfaceCreated();
mFURenderer.setUseTexAsync(true);
}
@Override
public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFram
e dstFrame) {
dstFrame.texture.textureId = mFURenderer.onDrawFrameSingleInput(
srcFrame.texture.textureId, srcFrame.width, srcFrame.height);
return 0;
}
@Override
```

```
public void onGLContextDestroyed() {
mFURenderer.onSurfaceDestroyed();
}
}
```

- **The beauty filter component does not generate new textures.**
  If the third-party beauty filter component you use does not generate new textures and you need to manually set an input texture and an output texture for the component, consider the following scheme:

```
@Override
public int onProcessVideoFrame(V2TXLiveVideoFrame srcFrame, V2TXLiveVideoFram
e dstFrame) {
thirdparty_process(srcFrame.texture.textureId, srcFrame.width, srcFrame.heigh
t, dstFrame.texture.textureId);
return 0;
}
```

3. You need some basic knowledge of OpenGL to deal with texture data and should refrain from high volume computing. This is because `onProcessVideoFrame` is called at the same frequency as the frame rate, and sophisticated processing may cause GPU overheating.

Customizing Video for Playback

iOS

1. Set V2TXLivePlayerObserver to listen for events of V2TXLivePlayer.

```
@interface V2TXLivePlayer : NSObject
/**
* @brief Set callbacks for the player<br>
* After setting callbacks, you can listen for events of V2TXLivePlayer,
* including the player status, playback volume, first audio/video frame, stati
stics, and warning and error messages.
*
* @param observer Target object for the player's callbacks. For more informati
on, see {@link V2TXLivePlayerObserver}
*/
- (void)setObserver:(id<v2txliveplayerobserver>)observer;
```

2. Get the player's video data from the onRenderVideoFrame callback.

```
/**
* @brief Video frame information
* V2TXLiveVideoFrame is the raw data that describes a video frame before encod
ing or after decoding
```

```
* @note It is used during custom video capturing to package the video frames t
o be sent, and during custom video rendering to get the packaged video frames
*/
@interface V2TXLiveVideoFrame : NSObject

///**Field meaning:** video pixel format
///**Recommended value:** V2TXLivePixelFormatNV12
@property(nonatomic, assign) V2TXLivePixelFormat pixelFormat;

///**Field meaning:** video data container format
///**Recommended value:** V2TXLiveBufferTypePixelBuffer
@property(nonatomic, assign) V2TXLiveBufferType bufferType;

///**Field meaning:** video data when bufferType is V2TXLiveBufferTypeNSData
@property(nonatomic, strong, nullable) NSData *data;

///**Field meaning:** video data when bufferType is V2TXLiveBufferTypePixelBuf
fer
@property(nonatomic, assign, nullable) CVPixelBufferRef pixelBuffer;

///**Field meaning:** video width
@property(nonatomic, assign) NSUInteger width;

///**Field meaning:** video height
@property(nonatomic, assign) NSUInteger height;

///**Field meaning:** clockwise rotation of video
@property(nonatomic, assign) V2TXLiveRotation rotation;

/// **Field description:** video texture ID
@property (nonatomic, assign) GLuint textureId;

@end


@protocol V2TXLivePlayerObserver <nsobject>
@optional
/**
* @brief Custom video rendering callback
*
* @note You will receive this callback after calling [enableCustomRendering](@
ref V2TXLivePlayer#enableCustomRendering:pixelFormat:bufferType:) to enable cu
stom video rendering
*
* @param videoFrame Video frame data {@link V2TXLiveVideoFrame}
*/
- (void)onRenderVideoFrame:(id<v2txliveplayer>)player
```

```
frame:(V2TXLiveVideoFrame *)videoFrame
@end
```

Android

1. Set V2TXLivePlayerObserver to listen for callbacks of V2TXLivePlayer.

```
public abstract void setObserver(V2TXLivePlayerObserver observer)
```

2. Get the player's video data from the onRenderVideoFrame callback.

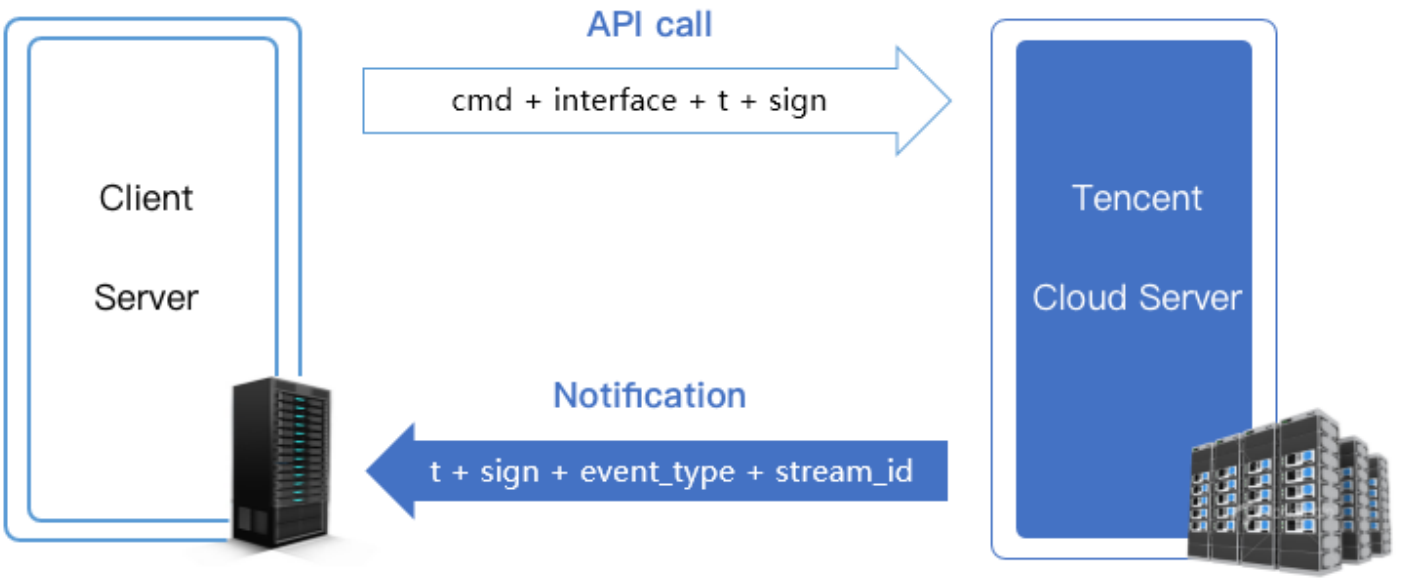
```
public final static class V2TXLiveVideoFrame
{
/// Video pixel format
public V2TXLivePixelFormat pixelFormat = V2TXLivePixelFormat.V2TXLivePixelForm
atUnknown;
/// Video data container format
public V2TXLiveBufferType bufferType = V2TXLiveBufferType.V2TXLiveBufferTypeUn
known;
/// Video texture pack
public V2TXLiveTexture texture;
/// Video data
public byte[] data;
/// Video data
public ByteBuffer buffer;
/// Video width
public int width;
/// Video height
public int height;
/// Clockwise rotation of video
public int rotation;
}

public abstract class V2TXLivePlayerObserver {
/**
* Callback for custom video rendering
*
* @param player The player object sending this callback
* @param videoFrame Video frame data {@link V2TXLiveVideoFrame}
*/
void onRenderVideoFrame(V2TXLivePlayer player, V2TXLiveVideoFrame videoFrame);
}
```

Stream Disabling and Management

Last updated：2022-06-14 12:41:46

Communicating with Tencent Cloud Server



Your server can communicate with the Tencent Cloud server using the methods below.

- **API call**: Tencent Cloud provides a series of live streaming APIs for your backend server to query and manage stream status, among others.
- **Notification**: Tencent Cloud can send messages (JSON) to your backend server when particular events occur, for example, when the status of a live stream changes or when a recording file is generated. You only need to register a callback URL in Tencent Cloud to receive event notifications.

API Call

Tencent Cloud offers a series of live streaming APIs for your backend sever to query and manage stream status, among others. For details, please see API Category.

Calling method

You can call the APIs on your **server** using the HTTP GET method (i.e., inserting the query parameters in the request URL). For the specific method, see the sample code in the API documentation.

Security mechanism

Using HTTP for API calls can ensure performance, but it's essential to introduce a mechanism to secure the communication between your server and the Tencent Cloud backend.

All live streaming cloud APIs use the same security mechanism: **t + sign verification**.

- **t (expiration time)**: If the current time is later than the `t` value specified in an API request or notification, the request or notification is considered invalid. This can prevent network replay attacks. The value of `t` is a Unix timestamp, i.e., the number of seconds that have elapsed since 00:00:00 UTC/GMT, January 1, 1970.

- **sign (security signature):** `sign` = MD5 (key + t). This means splicing the encryption key and `t` into a string and calculating its MD5 checksum. The encryption key is the CGI calling key, which can be specified in the CSS console. The directions below show how to configure authentication for stream publishing.

  i. Click **Domain Management**, find your publishing domain, and click **Manage**.
  ii. Click **Push Configuration** and, in the **Authentication Configuration** section, and click **Edit**.

  > Note：
  > To configure authentication for a **playback domain name**, click Domain Management, find your playback domain, click **Manage** to go to the details page, select **Access Control**, and click **Edit** in the Authentication Configuration** section.

- **How it works**
  MD5 is an irreversible hash algorithm, so as long as the key is not disclosed, even if attackers have multiple pairs of `t` and `sign` values, they cannot calculate the keys and therefore cannot launch spoofing attacks.

- **Calculation example**
  Assume that the current time is 11:46:00, July 21, 2021 and the validity period is 1 minute. That means requests or notifications carrying the `t` value after 11:47:00, July 21, 2021 would be considered invalid.

```
t = "2021-07-21 11:47:00" = 1626839220
```

  Assume that the key is **5d41402abc4b2a76b9719d911017c592**. The signature value would be:

```
sign = MD5(5d41402abc4b2a76b9719d911017c5921626839220) = 5ee8ca6c28cbe415b40352
969cdf8249
```

Error Codes

HTTP errors

Error Code	Error Message	Description
403	Forbidden	To ensure security, verification is required for API calls. If this error occurs during browser verification, check whether there is `skey` in the cookies.
404	Not Found	Check whether the request includes a host.

Common API errors

Error Message	Description
appid is invalid	`appid` is invalid because the feature is not activated.

Frontend API access errors

Error Message	Description
cmd is invalid	`cmd` is invalid because the feature is not activated.
sign invalid	The signature is invalid. For details, see security mechanism.
time expired	The verification is successful, but the URL has expired. For details, see security mechanism.

Backend API query errors

Error Code	Error Message	Description
0	query data successfully	The query is successful and the data queried is returned.
1000	user is not registered for statapi	The user has not registered statapi. Please submit a ticket to activate it.
1001	user service for statapi was stopped	The statapi access service is suspended for the user.

Error Code	Error Message	Description
1201	internal/system error	Internal system error. Report the problem to your service provider by submitting a ticket.
1202	invalid request/request frequency exceeds limit	The request is invalid, usually because the API rate limit is reached. You can apply to relax the limit.
1204	invalid input param	The request parameters are invalid. Check whether the parameters passed in meet the requirements.
1301	has not live stream	This error code is returned if there is no active stream when a real-time API is called.
10003	query data is empty	Query is successful at the backend, but no data is returned. For example, if no audio/video is played for some time and the `Get_LivePlayStatHistory` API is called, this error code will be returned.

Note：
The above error codes apply only to the APIs listed in this document and do not include event notifications.

Notification

For details about notifications, please see Callback Event Message Notification.

Recording and Replay

Last updated：2022-06-14 12:41:46

Feature Overview

The recording and playback features allow you to record live streams and play them to users on demand.

Your application may have a relatively small number of hosts at the early stage. Playback can enrich the content your application provides to audience at this stage. Even after your hosts grow to a considerable scale, having a library of high-quality content is still essential. Videos of past streams also make up an important part of a host's profile.

Enabling Recording

The recording and playback features are built on Tencent Cloud's **VOD service**. Therefore, to use the recording feature, you need to activate VOD in the console. After that, you can find recorded videos in Video Management of the VOD console.

Follow the steps below to enable the recording feature.

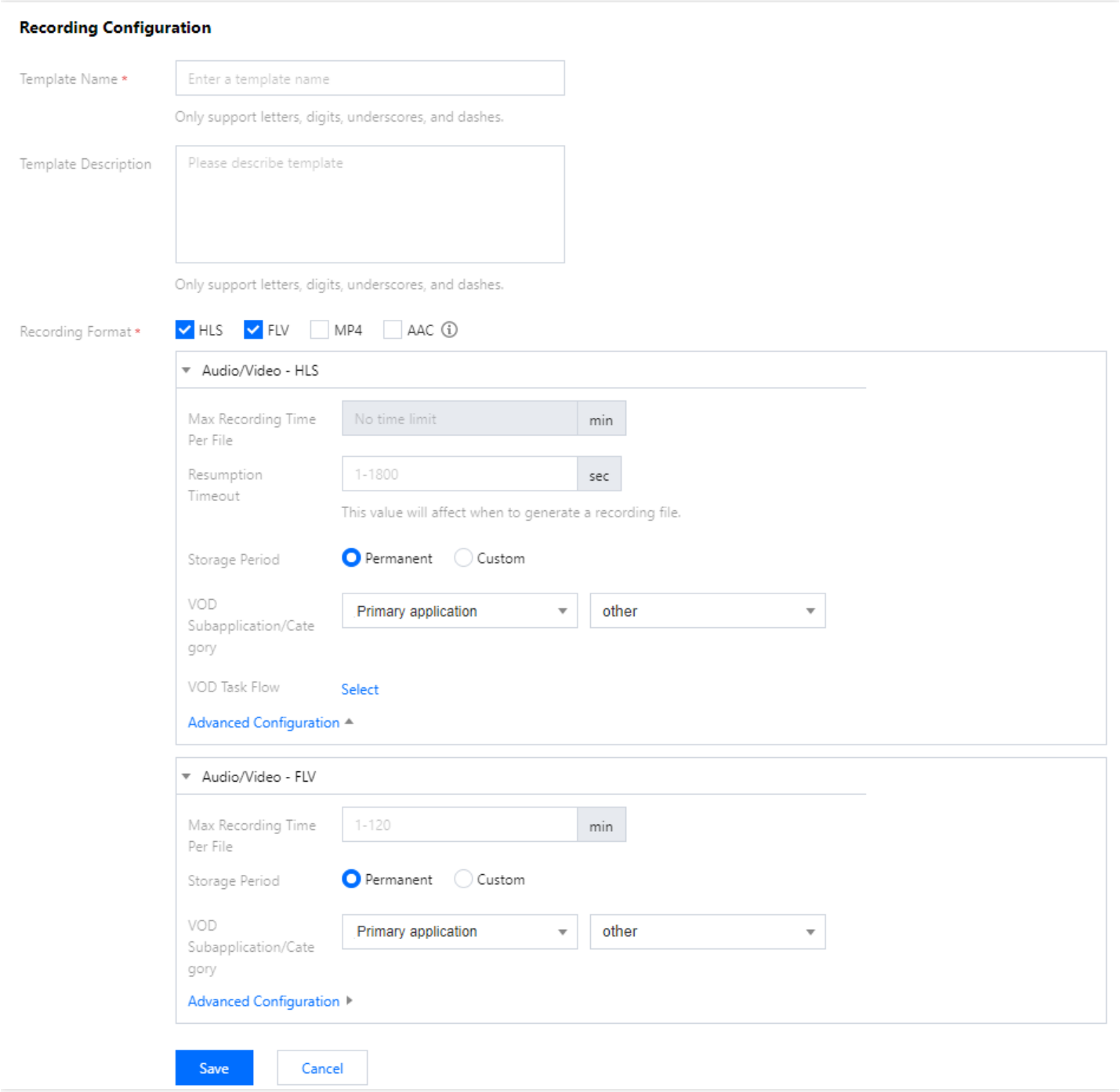
Note：

- You can record live streams using an API. For more information, see CreateRecordTask.
- Recorded videos are automatically stored in the VOD platform. Therefore, to use the recording feature, you need to activate VOD and purchase storage and traffic packages to store and play back recorded videos. For more information, see Getting Started.

Directions

In the CSS console, go to **Feature Configuration** > **Live Recording** and click **Create Recording Template**. After configuration, click **Save**. For detailed instructions, please see the "Creating a Recording Template" section in CSS

Recording.



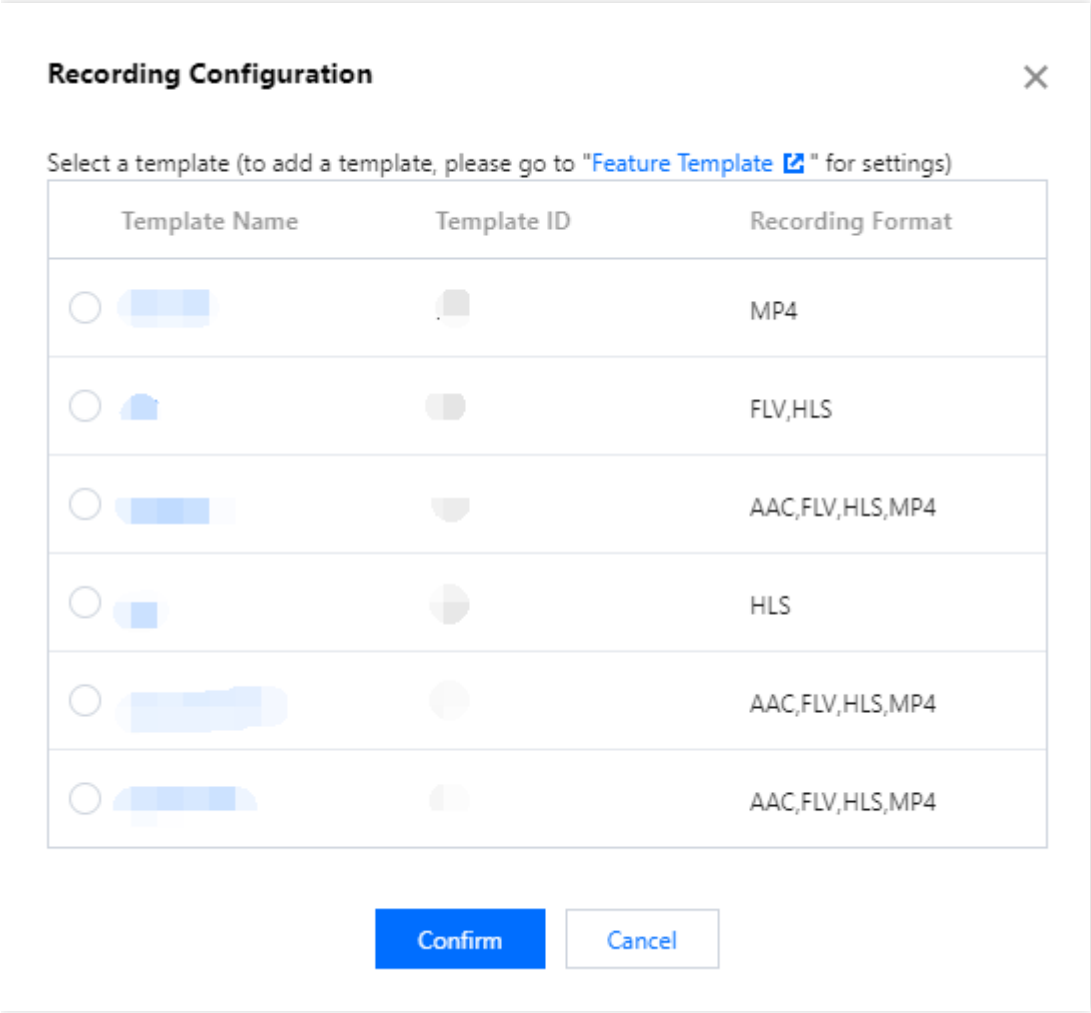
Item	Description
Max Recording Time Per File	• There is no upper limit on the recording time of a file in HLS format. If a live stream is interrupted and the timeout period for resumption elapses, a new recording file will be generated to continue recording. • The allowed duration of a file recorded in MP4, FLV, or AAC format is 1-120 minutes.

Resumption Timeout		Only the HLS format supports recording resumption after interruption, and the timeout period for resumption can be set between 0 and 1800 seconds.
Storage Period (day)		You can select **Permanent** to save a recording file permanently or **Custom** to specify a storage period (up to 1,500 days). Setting the period to `0` means to save recording files permanently.
VOD Subapplication/Category		By default, streams are recorded to the primary application in VOD. You can also record them to a category of a writable subapplication.
Advanced Configuration	Storage Policy	Select STANDARD_IA (cold storage) if the recording files will not be accessed frequently or will be stored for a long period of time, and select STANDARD (default) if the recording files need to be played back for business purposes. • If you select **Standard** and cold storage is enabled for the selected application/category, streams will be recorded to standard storage first before the configured cold storage policy is executed on the recording files. You can view your cold storage policies in the console. • If you select **STANDARD_IA** and cold storage is enabled for the selected application/category, streams will be recorded to STANDARD_IA storage first, and the system will then determine whether to execute the cold storage policy.
	VOD Task Flow	Click **Select** to bind a task flow created under the VOD subapplication. You can also click a task flow to go to the VOD console, where you can modify the task flow or create new ones. The bound task flow will be executed on recording files upon generation, for which you will be charged VOD fees.

Binding domain name

After creating a recording template, in the CSS console, select **Domain Management**, click a publishing domain to go to the details page, select **Template Configuration**, click **Edit** in the **Recording Configuration** section to bind the domain name to the template created, and click **save**. For detailed directions, please see Binding a Template with

Domain Names.



Getting Files

Playback URLs are generated for new recording files. You can use them to enable various additional features for your application based on your business needs.

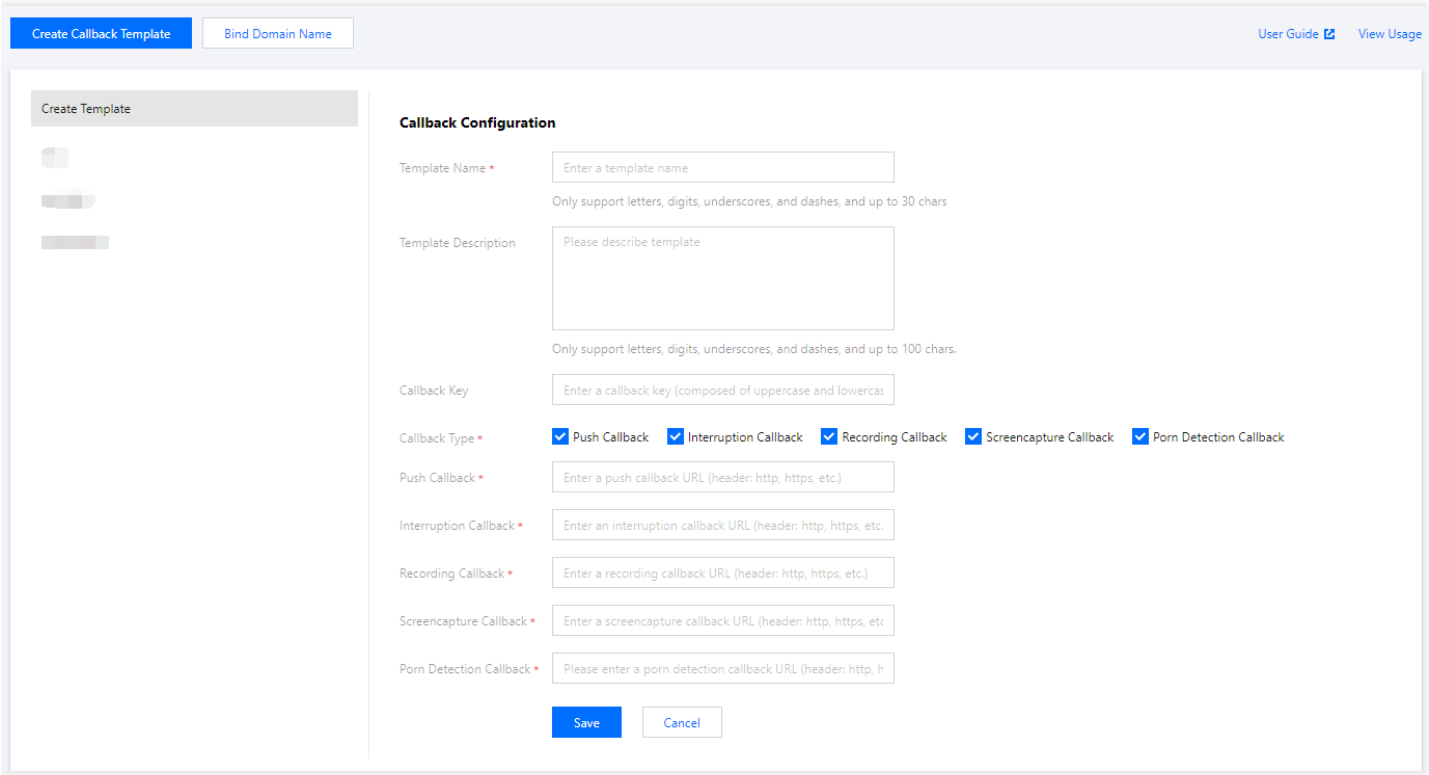
For example, you can add the playback URLs of a host's past live streams to his or her profile. You can also select high-quality content from past live streams and recommend it to your users in a playback list.

There are two methods to get recording files.

Method 1: listening for notifications

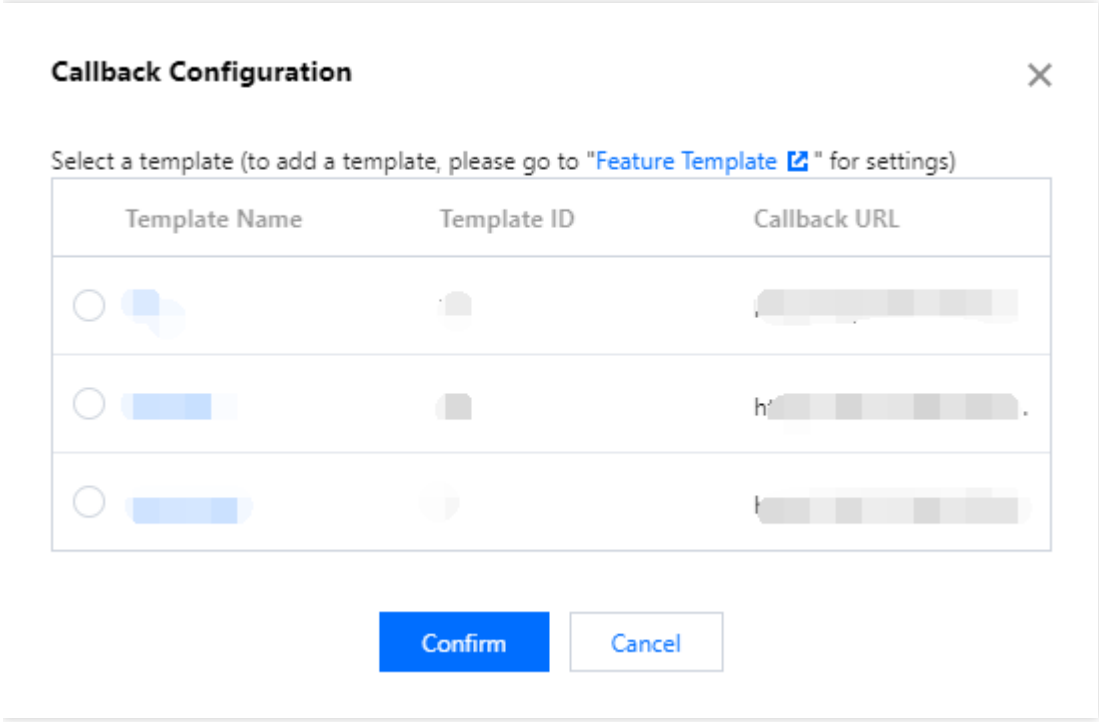
Register a callback URL on your server and provide it when configuring callbacks with Tencent Cloud. Tencent Cloud will notify you of the generation of new recording files via the URL.

In the CSS console, select **Event Center** > **Live Stream Callback**, click **Create Callback Template**, fill in the callback information, and click **Save**. For detailed directions, see Creating a Callback Template.



Binding a domain name to the template

After creating a callback template, select **Domain Management**, click a publishing domain name to go to the details page, select **Template Configuration**, click **Edit** in the **Callback Configuration** section to bind the domain name to the template created, and click **Save**.



Below is a typical notification message. It indicates that a new FLV recording file with the ID `9192487266581821586` has been generated, and the playback URL is `http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv`.

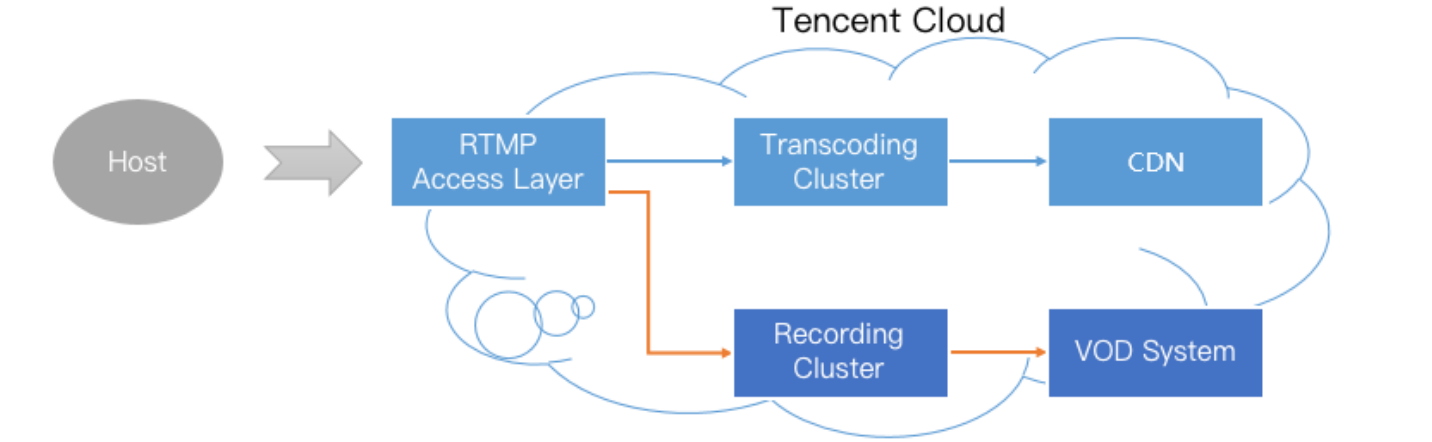
```
{
"channel_id": "2121_15919131751",
"end_time": 1473125627,
"event_type": 100,
"file_format": "flv",
"file_id": "9192487266581821586",
"file_size": 9749353,
"sign": "fef79a097458ed80b5f5574cbc13e1fd",
"start_time": 1473135647,
"stream_id": "2121_15919131751",
"t": 1473126233,
"video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
"video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv"
}
```

Method 2: querying in the console or via an API

Recording files are saved automatically in the VOD system after generation. You can view recording files in the VOD console or query files using a VOD API. For details, please see Obtaining Recording Files.

FAQs

1. How does live recording work?



After you enable recording for a live stream, each audio/video frame published from the host's mobile phone will be relayed to the recording system and written into a recording file.

When a live stream is interrupted, the access layer will immediately ask the recording server to wrap up the writing process, save the file generated to the VOD system, and generate an index for the file. You can then find the recording file in the VOD system. If you have configured recording callback, the recording system will send the **index ID** and **playback URL** to the server you specified.

If a file is too large, errors may occur while the file is being transferred and processed in the cloud. As a result, to ensure the success of recording, we have capped the duration of a single recording file at 120 minutes. You can record shorter video segments by setting the `RecordInterval` parameter to a smaller value.

2. How many recording files does a live stream generate?

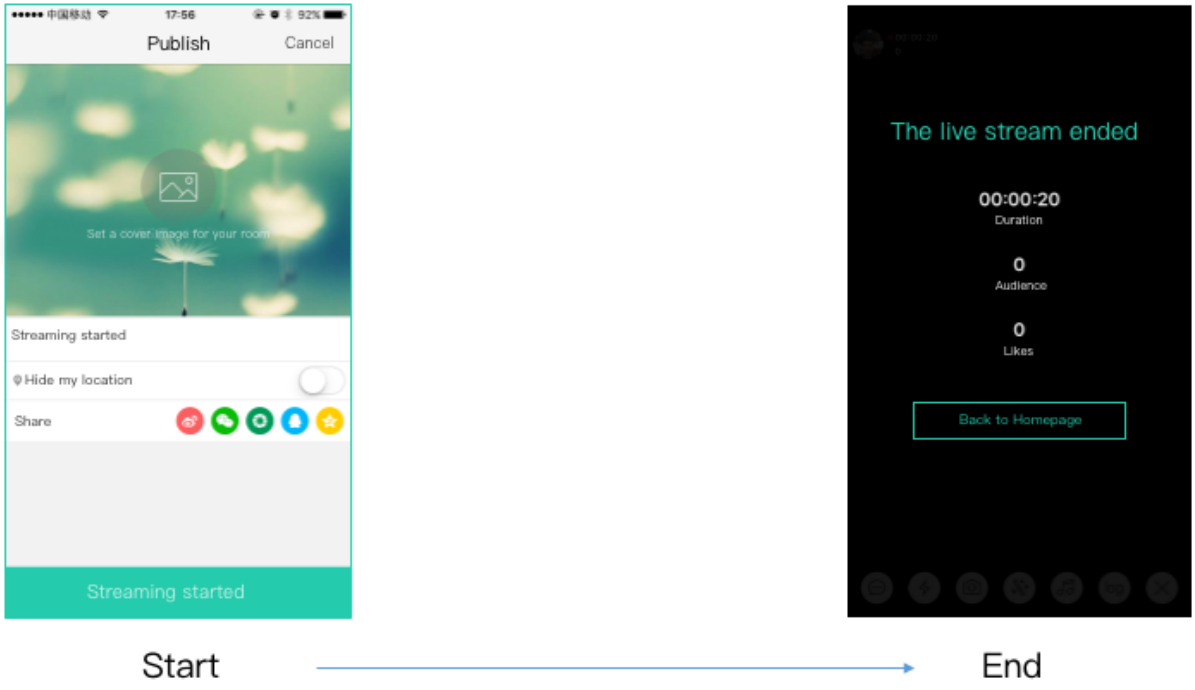
- **Recording in MP4, FLV, or AAC format**: the duration of a single file ranges from 1 to 120 minutes. You can specify a shorter segment using the `RecordInterval` parameter of the CreateLiveRecordTemplate API.
  - If a live stream is so short that it ends before the recording module is started, no recording files will be generated.
  - If a live stream lasts shorter than `RecordInterval` and is not interrupted, only one recording file will be generated.
  - If a live stream lasts longer than `RecordInterval`, the stream will be segmented based on the duration specified by `RecordInterval`. The purpose of this is to reduce the uncertainty of a file's transfer time in a distributed system.
  - If a live stream is interrupted and resumed, a new video segment will be generated each time an interruption occurs.

- **Recording in HLS format:** There is no upper limit on the duration of recording files. If a live stream is interrupted and the timeout period for breakpoint resumption (which can be set to 0-1800s) elapses, a new recording file will be generated to continue recording.

3. How do I know which live stream a recording file belongs to?

Tencent Cloud as a PaaS provider does not really know how you define a live stream. Assume that a host streamed for 20 minutes, but the process was interrupted twice, once due to network change and once manually by the host. Is it one live stream or three?

In most mobile live streaming scenarios, we consider the period between the two time points below as a live stream.



If you use the above standard, the time information provided by your application is important. To determine which live stream a recording file belongs to, just search for the recording notification received by live stream code and time. Each recording notification carries information including stream ID, start time, and end time.

4. How to splice video segments?

You can splice video segments using a cloud API.