Live

Best Practices

Issue	01
Date	2026-03-27

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 Cloud Live Helps Lower Streaming Latency

Generally, the latency for RTMP and FLV streaming is about 5 seconds. If the latency is too high, perform the following operations to lower the latency.

GOP Configuration

A Group of Pictures (GOP) is a collection of successive pictures. Each picture is a frame, and therefore a GOP is a group of frames. A livestream consists of a sequence of video frames, including I-frames and P-frames. A GOP starts with an I-frame followed by multiple P-frames. When a user starts a stream, the player must locate and receive the latest I-frame from the server before it can begin playback. Consequently, reducing the number of frames per GOP can decrease the time required for the player to load the GOP frames. The recommended keyframe interval is 1–2s.

Streaming Protocol Selection

Huawei Cloud Live supports three streaming protocols: RTMP, HTTP-FLV, and HLS.

```
RTMP: rtmp://Streaming domain name/AppName/StreamName
HTTP-FLV: http://Streaming domain name/AppName/StreamName.flv
M3U8: http://Streaming domain name/AppName/StreamName.m3u8
```

- RTMP splits large video frames and audio frames, encrypts them, and transmits them as small data packets. However, packet disassembly and assembly are complex. Therefore, unexpected problems may occur if there are a large number of concurrent requests.
- HLS is a streaming media protocol launched by Apple. It works by breaking the overall stream into a sequence of small HTTP-based segments (5–10s) and uses the M3U8 index table to manage these segments. The videos downloaded by the client are complete segments. Therefore, videos play smoothly. However, the player starts playback only after buffering three or four segments. Therefore, there will be a latency of about 10–30s.
- HTTP-FLV is launched by Adobe. Some tag headers are added to large video frames and audio and video headers. HTTP-FLV is mature in terms of latency and large-scale concurrency. However, it is only supported on certain mobile browsers.

To sum up, selecting HTTP-FLV as the streaming protocol can effectively reduce the latency. HLS is supported on most of browsers. Therefore, HLS is the first choice for many users.

2 Cloud Live Enables Instant Playback

A one-second Time to First Frame (TTFF) means the video becomes visible within one second of initiating playback. Technically, TTFF represents the duration required for the player to decode and render the initial frame.

- App

  HTTP-FLV is recommended for app players, which is the most widely used protocol in the live broadcast scenario. HTTP does not involve complex status interactions. Using RTMP delivers longer TTFF than using HTTP-FLV because several handshakes are involved in the initial phase of an RTMP connection. Therefore, from the perspective of latency, HTTP-FLV is better than RTMP.

- PC Browser

  A PC browser usually uses Flash Player. Chrome also supports MSE, but MSE does not have obvious advantages over Flash Player, which adopts forced buffering. Therefore, videos load longer (greater than 1s) on a PC browser than on an app (using HTTP-FLV).

- Mobile Browser

  Safari supports HLS (m3u8) and even uses the hardware decoding chip of iPhone to facilitate video playback. Therefore, if DNS has cache, videos load fast, but only on the iOS platform.

  Android browsers are highly random. Due to severe fragmentation, the implementation of system browsers varies with versions and models.

In conclusion, using HTTP-FLV on apps can help achieve one-second TTFF.

3 Cloud Live Secures Live Resources with Authentication and Stream Push Disabling

Scenario Description

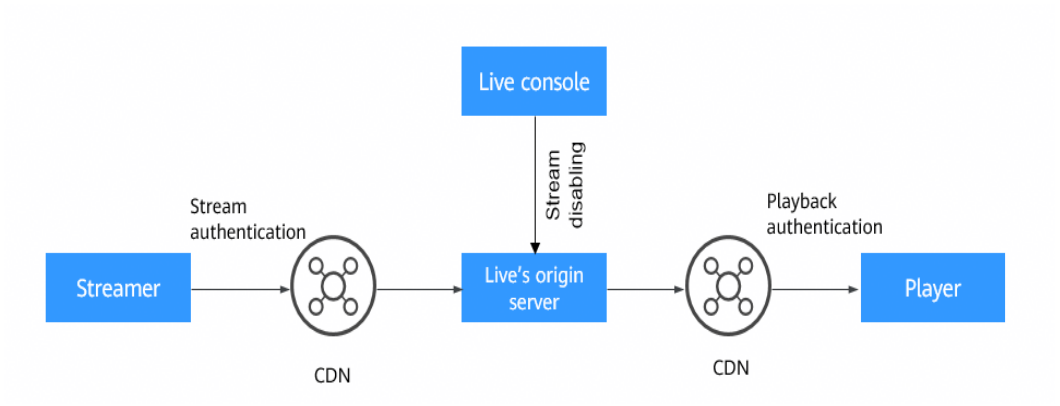
To protect your live resources from unauthorized download or theft, you can configure referer validation, URL validation, or an access control list (ACL). You can also disable stream pushing.

- After authentication is enabled, CDN identifies and filters visitors based on the defined rules. Only legitimate requests are permitted, while unauthorized access is rejected.
- Disabling stream push is recommended if a stream is found to be non-compliant or is being misused.

Implementation

As shown in **Figure 3-1**, Live provides the following functions to protect live resources.

Figure 3-1 Security architecture



- **Stream Authentication**
  – URL validation: The streamer requests pushing streams via the ingest URL with the encrypted string carried. CDN verifies the request based on

authentication information carried in the ingest URL. Only requests that pass the verification are allowed.

  – Access control lists (ACLs): CDN allows or rejects stream pushing requests based on the blocklist or trustlist you configured.

- **Playback Authentication**
  – Referer validation: CDN identifies the referer field in a playback request based on the blocklist or trustlist to reject or allow the playback request.
  – URL validation: The viewer requests playback via the streaming URL with the encrypted string carried. CDN checks whether the streaming URL is valid based on authentication information in the URL. Only requests that pass the verification are allowed.
  – ACLs: CDN rejects or allows playback requests based on the blocklist or trustlist you configured.
- **Disabling a Live Stream**: If live content is non-compliant or the ingest URL has been used by unauthorized users, you can add this stream to the ACL on the Live console to disable the stream. The stream cannot be pushed again until you remove it from the ACL.

Stream Authentication

Before pushing a live stream, configure the URL validation or an ACL.

Step 1 Log in to the **Live console**.

Step 2 In the navigation pane, choose **Domains**.

Step 3 Click **Manage** in the **Operation** column of the desired domain name.

Step 4 In the navigation pane, choose **Basic Settings** > **Access Control**.

Step 5 Configure authentication as required.

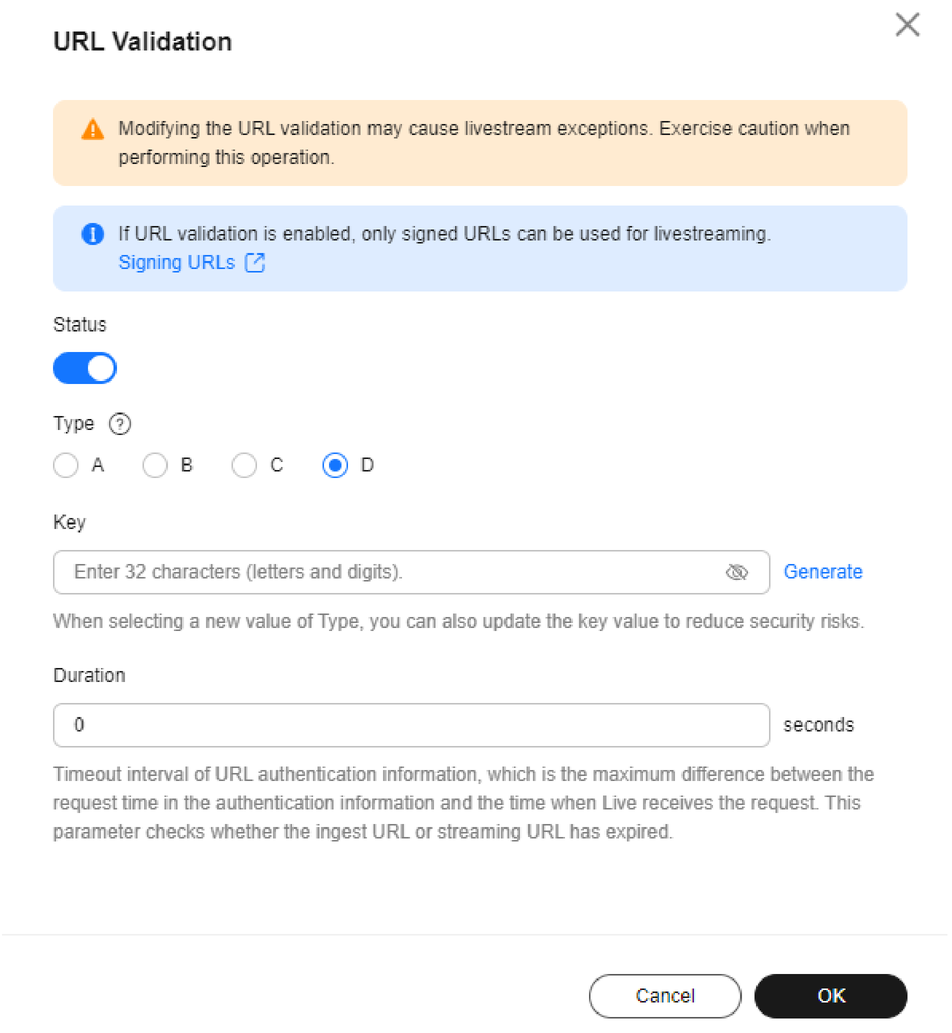
- URL validation

  If you have other requirements on custom validation rules, **submit a service ticket** for Huawei Cloud technical support.

  a. Click the edit icon on the right of **URL Validation**. The **URL Validation** dialog box is displayed on the right.

  b. Toggle on the **Status** switch to configure related parameters, as shown in **Figure 3-2**.

Figure 3-2 Configuring URL validation



URL Validation

Modifying the URL validation may cause livestream exceptions. Exercise caution when performing this operation.

If URL validation is enabled, only signed URLs can be used for livestreaming.
Signing URLs

Status

Type

A B C [✓ D]

Key

Enter 32 characters (letters and digits).

When selecting a new value of Type, you can also update the key value to reduce security risks.

Duration

0 seconds

Timeout interval of URL authentication information, which is the maximum difference between the request time in the authentication information and the time when Live receives the request. This parameter checks whether the ingest URL or streaming URL has expired.

Table 3-1 URL validation parameters

Parameter	Description
Type	You can use signing method A, B, C, or D to calculate a signed string. **NOTE** Signing methods A, B, and C have security risks. Signing method D is more secure and recommended.
Key	Authentication key. ▪ You can customize a key. A key consists of 32 characters. Only letters and digits are allowed. ▪ A key can also be automatically generated.

Parameter	Description
Duration	Timeout interval of URL authentication information, that is, the maximum difference between the request time carried in authentication information and the time when Live receives the request. This parameter is used to check whether an ingest URL or streaming URL expires. The value ranges from 1 minute to 30 days and is expressed in seconds.

c. Click **OK**.

d. Generate a signed ingest URL in either of the following ways.

   After URL validation is configured, the original URL cannot be used. You need to generate a new ingest URL.

   - Automatic generation: Use the signed URL generation tool to quickly generate a signed URL. For details, see **Signed URL Generation Tool**.
   - Manual combination: Manually generate a signed URL based on the configured authentication type. For details about the combination example, see **URL Validation**.
     - Signing method A

       Signed URL format:
       ```
       Original URL?auth_key={timestamp}-{rand}-{uid}-{md5hash}
       ```
       Formula for calculating **md5hash** is:
       ```
       sstring = "{URI}-{Timestamp}-{rand}-{uid}-{Key}"
       HashValue = md5sum(sstring)
       ```
     - Signing method B

       Signed URL format:
       ```
       Original URL?txSecret=md5(Key + Stream Name + txTime)&txTime=hex(timestamp)
       ```
     - Signing method C

       Signed URL format:
       ```
       Original URL?auth_info={Encrypted string}.{EncodedIV}
       ```
       Algorithms for generating the authentication fields are as follows:

       LiveID = <AppName>+"/"+<StreamName>

       Encrypted string = UrlEncode(Base64(AES128(<Key>,"$"+<Timestamp>+"$"+<LiveID>+"$"+<CheckLevel>)))

       EncodedIV = Hex(IV used for encryption)
     - Signing method D

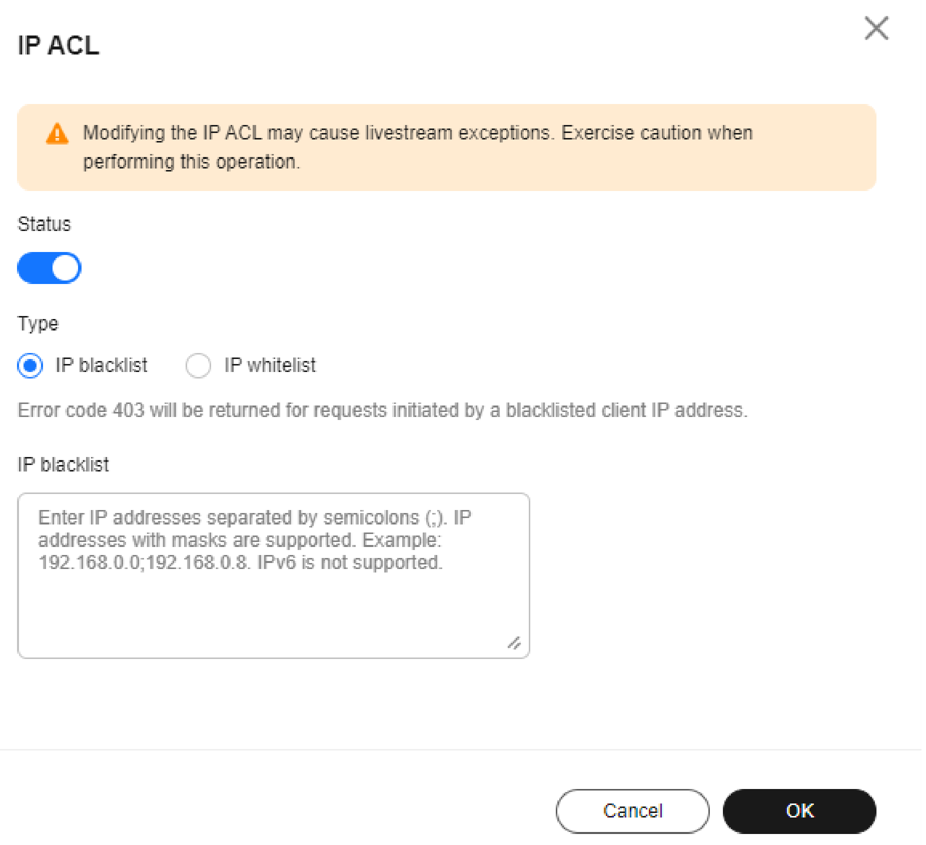
       Signed URL format:
       ```
       Original URL?hwSecret=hmac_sha256(Key, Stream Name +
       hwTime)&hwTime=hex(timestamp)
       ```

e. Verify whether URL validation has taken effect.

   Use a third-party streaming tool to verify the signed ingest URL. If the original ingest URL does not work but the signed ingest URL works, URL validation has taken effect.

- IP ACL
  a. Click the edit icon on the right of **IP ACL**. The **IP ACL** dialog box is displayed on the right.
  b. Toggle on the **Status** switch to configure an IP address ACL, as shown in **Figure 3-3**.

Figure 3-3 Configuring an IP address ACL



IP ACL

Modifying the IP ACL may cause livestream exceptions. Exercise caution when performing this operation.

Status

Type

[✓ IP blacklist] IP whitelist

Error code 403 will be returned for requests initiated by a blacklisted client IP address.

IP blacklist

Enter IP addresses separated by semicolons (;). IP addresses with masks are supported. Example: 192.168.0.0;192.168.0.8. IPv6 is not supported.

  c. Select **IP address blocklist** or **IP address trustlist**, and enter IP addresses or IP address ranges.
  d. Click **OK**.

----End

Playback Authentication

You can configure referer validation, URL validation, or an ACL for streaming domain names.

Step 1 Log in to the **Live console**.

Step 2 In the navigation pane, choose **Domains**.

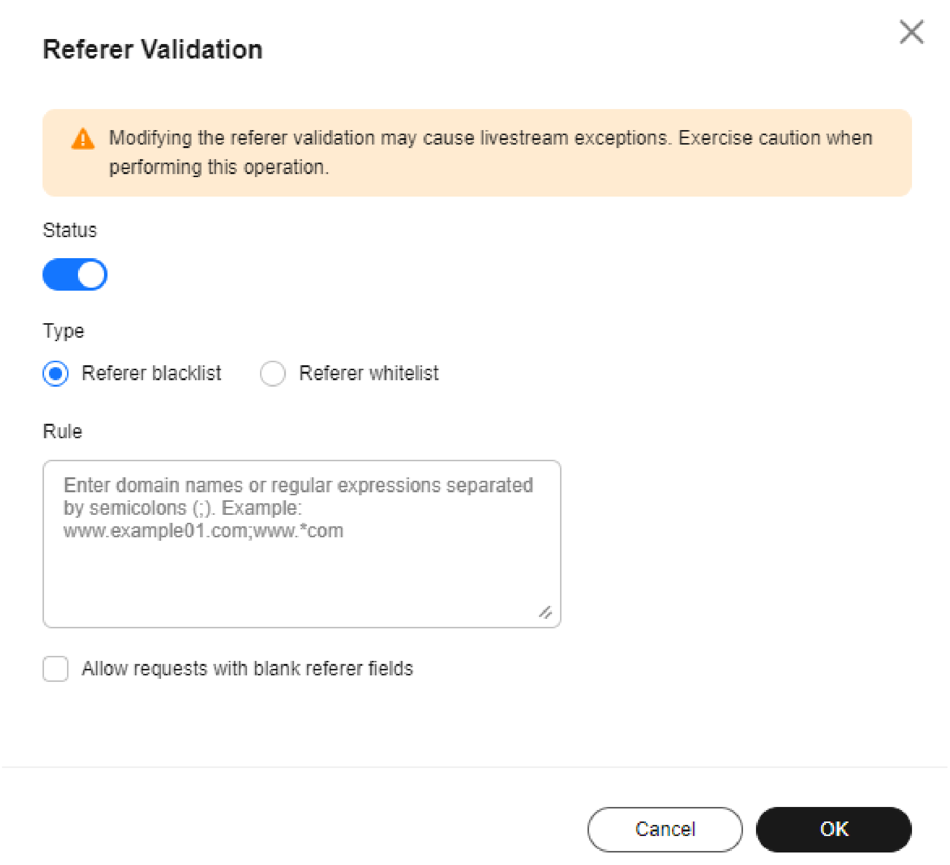
Step 3 Click **Manage** in the **Operation** column of the desired domain name.

Step 4 In the navigation pane, choose **Basic Settings** > **Access Control**.

Step 5 Configure authentication as required.

- Referer validation
  a. Click the edit icon on the right of **Referer Validation**. The **Referer Validation** dialog box is displayed on the right.
  b. Toggle on the **Status** switch to configure related parameters, as shown in **Figure 3-4**.

Figure 3-4 Configuring referer validation



Referer Validation

Modifying the referer validation may cause livestream exceptions. Exercise caution when performing this operation.

Status

Type

[✓ Referer blacklist] Referer whitelist

Rule

Enter domain names or regular expressions separated by semicolons (;). Example: www.example01.com;www.*com

Allow requests with blank referer fields

Cancel OK

Table 3-2 Referer validation parameters

Parameter	Description
Type	Blacklists and whitelists are supported. ■ **Referer blacklist** allows requests from all domains except those in the blacklist. ■ **Referer whitelist** denies requests from all domains except those in the whitelist. You can control whether to allow requests with an empty referer field, that is, whether to allow access through the browser address bar.

Parameter	Description
Rule	Domain name in the blacklist or whitelist. ▪ You can input 1 to 100 domain names. Use semicolons (;) to separate domain names. ▪ Domain names are matched using regular expressions. If you enter **^http://test*.com$**, **http://test.example.com** and **http://test.example01.com** are matched.

c. Click **OK**.

- URL validation

  The method of configuring URL validation for streaming domain names is the same as that for ingest domain names. For details, see **Stream Authentication**.

- IP ACL

  The method of configuring an ACL for streaming domain names is the same as that for ingest domain names. For details, see **Stream Authentication**.

----End

Disabling a Live Stream

If live content is non-compliant or the ingest URL has been used by unauthorized users, you can disable the stream to protect your live content.

Step 1 Log in to the **Live console**.

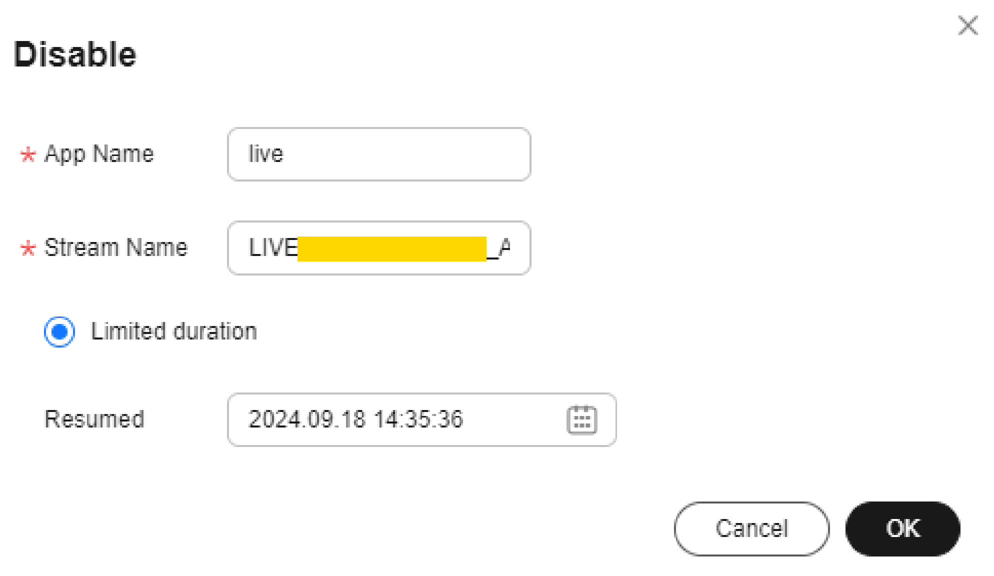
Step 2 In the navigation pane, choose **Streaming** > **Streams**.

Step 3 Locate the domain name for which stream push is to be disabled.

Step 4 Click **Disable** in the **Operation** column.

Select the time when stream push is resumed. You can view information about disabled livestreams on the **Disabled** tab.

Figure 3-5 Configuration of disabling stream push



Disable

* App Name	live
* Stream Name	LIVE[redacted]_A
● Limited duration	
Resumed	2024.09.18 14:35:36

Limited duration: The livestream cannot be pushed until the time indicated by **Resumed** arrives. Livestreams can be disabled for up to 90 days.

----End

Once the stream is disabled, the ingest URL cannot be used before the stream is resumed.

4 Cloud Live Offers Event Callbacks for Real-Time Event Monitoring

Live provides callbacks for stream pushing, recording, and snapshot capturing. If callbacks have been configured before livestreaming, when an event is triggered, Live sends an HTTP POST request to your server with the real-time event status included.

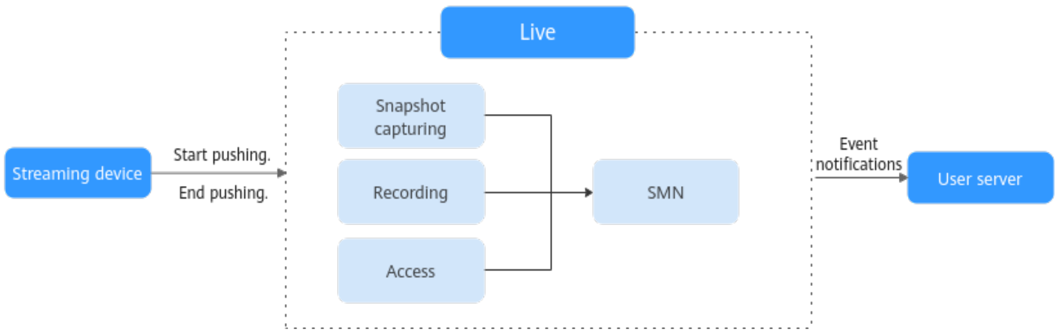
Callback Protocols

- Request: HTTP POST. The request body is in JSON format.
- Response: HTTP STATUS CODE = 200. The response body is in JSON format. You can customize the response body based on your needs.

  Example:
  ```
  {
    "status": 1,
    "result" : "success"
  }
  ```

Overall Process

Figure 4-1 Callback process



Streaming Callbacks

Streaming callbacks allow you to know whether stream push has started or ended. **Table 4-1** describes the fields in a callback message.

Table 4-1 Message body

Field	Description
domain	Ingest domain name
app	Application name
stream	Stream name
user_args	Stream push parameter
client_ip	IP address of the stream push device
node_ip	IP address of the receiver
publish_timestamp	Unix timestamp. One single timestamp is generated for each stream push event.
event	Stream push or stream disconnection Options: ● **PUBLISH**: Stream push starts. ● **PUBLISH_DONE**: Stream push ends.
auth_timestamp	UNIX timestamp when the event notification signature expires. This parameter is carried when an authentication key is configured. The value is a decimal Unix timestamp, that is, the number of seconds that have elapsed since January 1, 1970 00:00:00 UTC/GMT. Example: **1592639100** (June 20, 2020 15:45)
auth_sign	Event notification signature. This parameter is carried when an authentication key is configured. `auth_sign = HmacSHA256 (event + domain + app + stream + auth_timestamp, key)` *key* indicates the key used for authentication.

An example message:

```
{
    "domain":"push.example.com",
    "app":"live",
    "stream":"example_stream",
    "user_args":"auth_info=yz1TG0PVN/5isfyrGrRj10gKPCWqSS2X02t6QsRrocH+mEq0gQ0g8k6KhalS84sQ
+kDprFyqI0yajbYiFmUO8e45B7ryaS+MpJBlYkhwnuFLnRiKK/
IXG7.33436b625354564f6e4d4d434f55&cdn=hw",
    "client_ip":"100.111.*.*",
    "node_ip":"112.11.*.*",
    "publish_timestamp":"1587954134,",
    "event":"PUBLISH"
    "auth_timestamp":1587954140,
    "auth_sign":"ff3b2bxxx5cfd56e76d72bed4c4aa2dxxxca8c2e46467d205a6417d4fc"
}
```

Recording Callbacks

Recording callbacks are available for recording to OBS. You can get notified when recording starts, recording file creation starts, a recording file has been created, recording ends, and recording fails. Table 4-2 describes the fields in a callback message.

Table 4-2 Message body

Field	Description
project_id	Project ID
job_id	Name of a file. This parameter is carried when the value of **event_type** is **RECORD_NEW_FILE_START** or **RECORD_FILE_COMPLETE**.
task_id	Recording task ID, which uniquely identifies a recording task.
event_type	Message type. Possible values are: ● **RECORD_START**: This event is triggered when you start recording. ● **RECORD_NEW_FILE_START**: This event is triggered in either of the following scenarios: – The system starts creating the first recording file. – After a live stream is resumed, if **Maximum Stream Pause Length** is set to **Generate a new file after a stream is paused.**, the system starts creating a recording file. – If the recording duration exceeds the configured recording length, the system starts creating a recording file. ● **RECORD_FILE_COMPLETE**: This event is triggered in either of the following scenarios: – When the recording duration reaches the configured recording length, a recording file has been created. – After a live stream is interrupted, if **Maximum Stream Pause Length** is set to **Generate a new file after a stream is paused.**, a recording file has been created. ● **RECORD_OVER**: This event is triggered when a live stream has been paused beyond the time indicated by **Maximum Stream Pause Length**. ● **RECORD_FAILED**: This event is triggered when stream pull fails or recordings fails to be uploaded to OBS.
publish_domain	Ingest domain name
app	App name

Field	Description
stream	Stream name
record_format	Recording format. The HLS, FLV, and MP4 formats are supported.
download_url	Address to download the recording. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**. **NOTE** The quality of video playback using the download address cannot be guaranteed.
asset_id	ID of a recording file. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
file_size	File size Unit: byte
record_duration	Duration of a recording Unit: second
start_time	Start time of a recording, which is, time when the first frame is received. The format is yyyy-mm-ddThh:mm:ssZ. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
end_time	End time of a recording. The format is yyyy-mm-ddThh:mm:ssZ. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
width	Width of a video recording. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
height	Height of a video recording. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
obs_location	Region where the OBS bucket for storing the recording is located. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
obs_bucket	OBS bucket where recordings are stored. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.

Field	Description
obs_object	OBS storage path. This parameter is used only when **event_type** is **RECORD_FILE_COMPLETE**.
auth_sign	Event notification signature. This parameter is carried when an authentication key is configured. ● **MD5**: **auth_sign** = MD5(*key* + *auth_timestamp*) ● **HMACSHA256**: HMACSHA256(*auth_timestamp* + *event_type* + *publish_domain* + *app* + *stream* + *download_url* + *play_url*, *key*) *key* indicates the key used for authentication.
auth_timestamp	UNIX timestamp when the event notification signature expires. This parameter is carried when an authentication key is configured. The value is a decimal Unix timestamp, that is, the number of seconds that have elapsed since January 1, 1970 00:00:00 UTC/GMT. If the time specified by **auth_timestamp** has expired, the notification will become invalid to avoid network replay attacks.
error_message	Description about a failed recording. This parameter is used only when **event_type** is **RECORD_FAILED**.

An example message:

- If **event_type** is **RECORD_START**:

```
{
  "project_id" : "70b76xxxxxx34253880af501cdxxxxxx",
  "publish_domain" : "push.example.com",
  "event_type" : "RECORD_START",
  "app" : "live",
  "stream" : "mystream",
  "record_format" : "HLS",
  "file_size" : 3957964,
  "record_duration" : 120
}
```

- If **event_type** is **RECORD_NEW_FILE_START**,

  that is:

  – The system starts creating the first recording file.

  – After a live stream is resumed, if **Maximum Stream Pause Length** is set to **Generate a new file after a stream is paused.**, the system starts creating a recording file.

  – If the recording duration exceeds the configured recording length, the system starts creating a recording file.

```
{
  "project_id" : "70b76xxxxxx34253880af501cdxxxxxx",
  "publish_domain" : "push.example.com",
  "event_type" : "RECORD_NEW_FILE_START",
  "app" : "live",
  "stream" : "mystream",
  "record_format" : "HLS",
  "file_size" : 3957964,
  "record_duration" : 120
}
```

- If **event_type** is **RECORD_FILE_COMPLETE**,

  that is:

  – When the recording duration reaches the configured recording length, a recording file has been created.

  – After a live stream is interrupted, if **Maximum Stream Pause Length** is set to **Generate a new file after a stream is paused.**, a recording file has been created.

```
{
  "project_id" : "70b76xxxxxx34253880af501cdxxxxxx",
  "publish_domain" : "push.example.com",
  "event_type" : "RECORD_FILE_COMPLETE",
  "app" : "live",
  "stream" : "mystream",
  "record_format" : "HLS",
  "download_url" : "https://obs.cn-north-4.myhuaweicloud.com/live/record-xxxx-
mystream-1589967495/record-push.example.com-live-mystream-1589967495.m3u8",
  "asset_id" : "1a0d8e9bfae388ccbbe5021e62aa1e96",
  "file_size" : 3957964,
  "record_duration" : 120,
  "start_time" : "2020-03-08T14:10:25Z",
  "end_time" : "2020-03-08T14:12:25Z",
  "width" : 1280,
  "height" : 720,
  "obs_location" : "https://obs.cn-north-4.myhuaweicloud.com",
  "obs_bucket" : "mybucket",
  "obs_object" : "live/record-xxxx-mystream-1589967495/record-hwpublish.myun.tv-live-
mystream-1589967495.m3u8"
  "auth_sign" : "4f97f46759axxxxxx7ad21e9935dc175",
  "auth_timestamp" : 1583676745
}
```

  – **download_url** indicates the address for downloading the recording file, that is, the address for storing the recording file in the OBS bucket.

  – **asset_id**: media asset ID

- If **event_type** is **RECORD_OVER**, that is, a live stream has been paused beyond the time indicated by **Maximum Stream Pause Length** and a recording has been created.

```
{
  "project_id" : "70b76xxxxxx34253880af501cdxxxxxx",
  "publish_domain" : "push.example.com",
  "event_type" : "RECORD_FILE_COMPLETE",
  "app" : "live",
  "stream" : "mystream",
  "record_format" : "HLS",
  "file_size" : 3957964,
  "record_duration" : 120
 }
```

- If **event_type** is **RECORD_FAILED**, that is, recording fails because stream pulling fails or uploading recordings to OBS fails:

```
{
  "project_id" : "70b76xxxxxx34253880af501cdxxxxxx",
  "publish_domain" : "push.example.com",
  "event_type" : "RECORD_FAILED",
  "app" : "live",
  "stream" : "mystream",
  "record_format" : "HLS",
  "file_size" : 3957964,
  "record_duration" : 120,
  "error_message": "Message example"
}
```

Snapshot Callbacks

This event is triggered when a snapshot file is created. **Table 4-3** describes the fields in a callback message.

Table 4-3 Message body

Field	Description
domain	Ingest domain name
app	Application name
stream_name	Stream name
snapshot_url	URL to download snapshots
width	Image width Unit: pixel
height	Image height Unit: pixel
obs_addr	Address of the OBS bucket where snapshots are stored. ● **bucket**: OBS bucket name ● **location**: Region where the OBS bucket is located ● **object**: OBS object path
auth_timestamp	UNIX timestamp when the event notification signature expires. This parameter is carried when an authentication key is configured. The value is a decimal Unix timestamp, that is, the number of seconds that have elapsed since January 1, 1970 00:00:00 UTC/GMT. Example: **1592639100** (June 20, 2020 15:45)
auth_sign	Event notification signature. This parameter is carried when an authentication key is configured. `auth_sign = HmacSHA256(domain + app + stream_name + snapshot_url + width + height + obs_addr.bucket + obs_addr.location + obs_addr.object + auth_timestamp,key)` *key* indicates the key used for authentication.

An example message:

```
{
   "domain": "play.example.com",
   "app": "live",
   "stream_name": "test001",
   "snapshot_url": "https://xxx.obs.cn-north-4.myhuaweicloud.com:443...",
   "width":"720",
   "height":"1280",
   "obs_addr": {
       "bucket": "xxx",
       "location": "cn-north-4",
       "object": "xxx.jpg"
   },
   "auth_timestamp":1587954140,
   "auth_sign":"4918b1axxxxxxb583cffa119d72513bbc35a989f8569fxxxxxx057646154a04a"
}
```

Configuring a Callback URL

Step 1 Log in to the **Live console**.

Step 2 In the navigation pane, choose **Domains**.

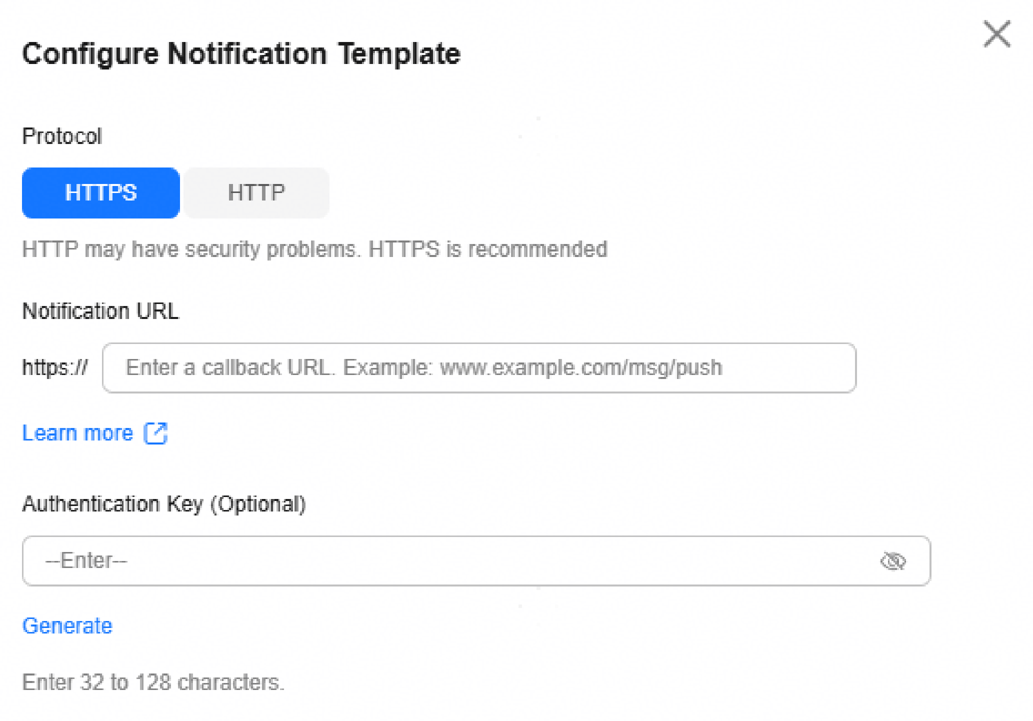
Step 3 Click **Manage** in the row containing the target ingest domain name.

Step 4 Configure a callback URL. The callback URL is the address of your server receiving notification messages, for example, **http://test.example.com/notify**.

The following three types of callbacks can be configured:

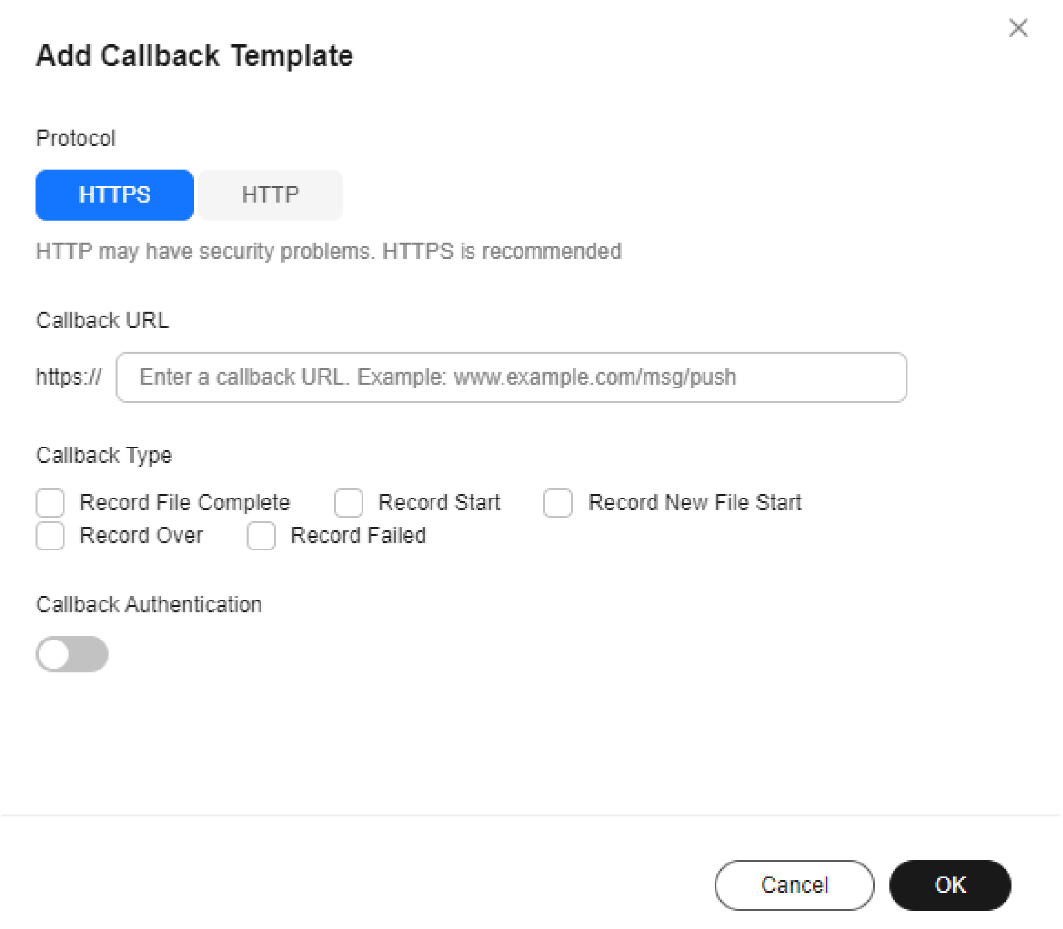
Streaming callbacks: Click the **Stream Status Notification** tab, click **Add**, and configure a callback URL for receiving notifications about stream push changes, as shown in **Figure 4-2**.

Figure 4-2 Adding a notification URL


Configure Notification Template
Protocol
HTTPS
HTTP
HTTP may have security problems. HTTPS is recommended
Notification URL
https://
Enter a callback URL. Example: www.example.com/msg/push
Learn more
Authentication Key (Optional)
--Enter--
Generate
Enter 32 to 128 characters.

Recording callbacks: Click the **Recording (New)** tab, click **Create Callback Configuration**, and configure a callback URL for receiving recording notifications, as shown in **Figure 4-3**.

Figure 4-3 Adding a callback URL


Add Callback Template

Protocol

HTTPS | HTTP

HTTP may have security problems. HTTPS is recommended

Callback URL

https:// Enter a callback URL. Example: www.example.com/msg/push

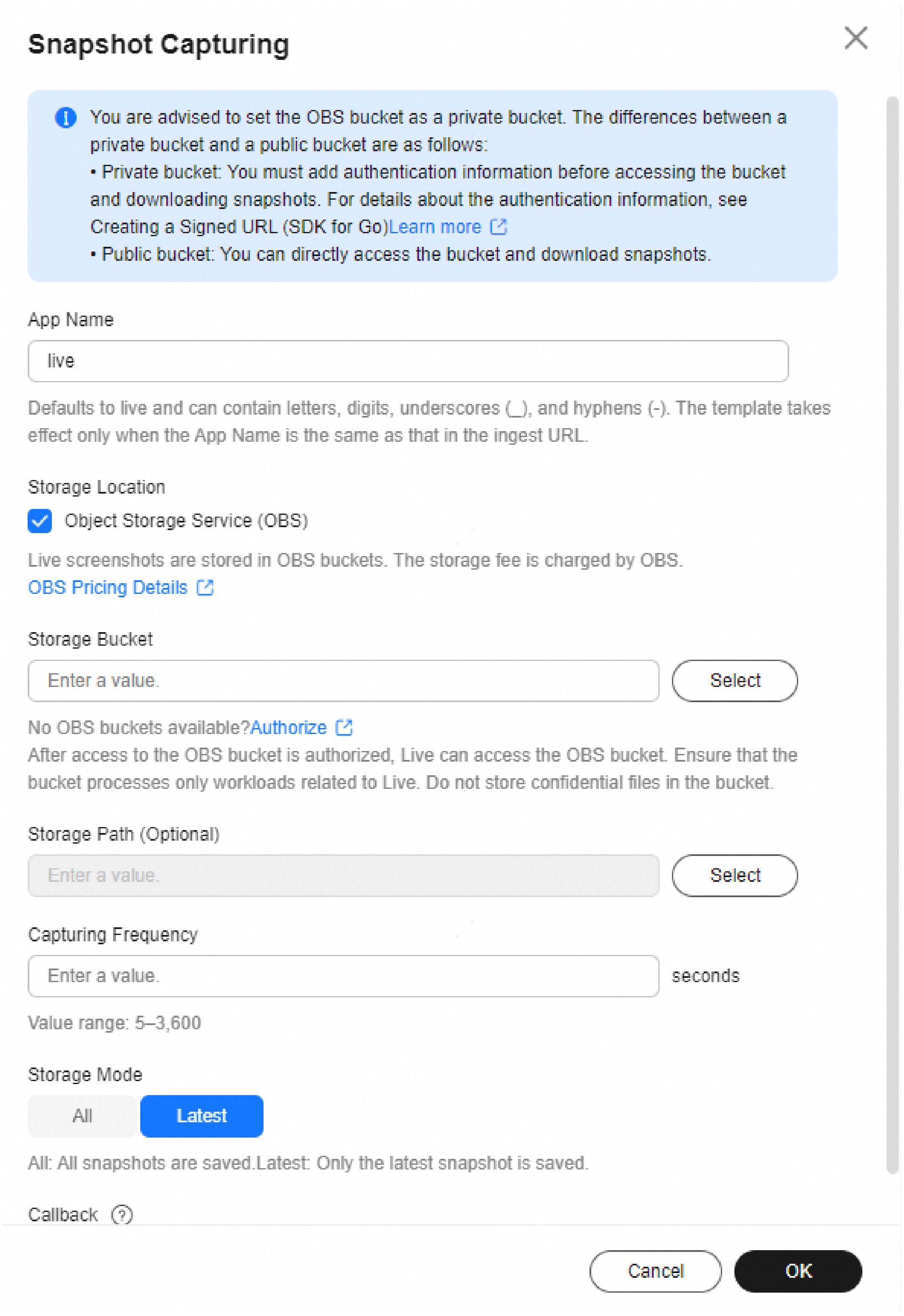
Callback Type

- [ ] Record File Complete
- [ ] Record Start
- [ ] Record New File Start
- [ ] Record Over
- [ ] Record Failed

Callback Authentication

Snapshot callbacks: Click the **Snapshot Capturing** tab, click **Create Snapshot Template**, and configure a callback URL for receiving snapshot notifications, as shown in **Figure 4-4**.

Figure 4-4 Creating a snapshot template



Snapshot Capturing

You are advised to set the OBS bucket as a private bucket. The differences between a private bucket and a public bucket are as follows:
• Private bucket: You must add authentication information before accessing the bucket and downloading snapshots. For details about the authentication information, see Creating a Signed URL (SDK for Go)Learn more
• Public bucket: You can directly access the bucket and download snapshots.

App Name

live

Defaults to live and can contain letters, digits, underscores (_), and hyphens (-). The template takes effect only when the App Name is the same as that in the ingest URL.

Storage Location

✓ Object Storage Service (OBS)

Live screenshots are stored in OBS buckets. The storage fee is charged by OBS.
OBS Pricing Details

Storage Bucket

Enter a value. Select

No OBS buckets available?Authorize
After access to the OBS bucket is authorized, Live can access the OBS bucket. Ensure that the bucket processes only workloads related to Live. Do not store confidential files in the bucket.

Storage Path (Optional)

Enter a value. Select

Capturing Frequency

Enter a value. seconds

Value range: 5–3,600

Storage Mode

All [✓ Latest]

All: All snapshots are saved.Latest: Only the latest snapshot is saved.

Callback

Cancel OK

----End

5 Media Live Improves Livestreaming Experience with Multi-Bitrate Adaptation

Multiple transcoding templates can be configured for a channel of Media Live to adapt to different terminal types. That is, multiple outputs with the same content but different bitrates or resolutions are provided for a user's player. Then content suitable for the player will be distributed to improve livestreaming experience.

To configure multi-bitrate adaptation, perform the following steps:

Step 1 Log in to the **Live console**.

Step 2 **Create a transcoding template**.

You need to confirm that the player supports all bitrates or resolutions available. One output bitrate or resolution corresponds to one transcoding template. You need to create all transcoding templates for each bitrate or resolution one by one.

Step 3 In the navigation pane on the left, choose **Channels** under **Media Live**. The **Channels** page is displayed.

Step 4 Find the desired channel and click **Manage** in the **Operation** column. The **Update Channel** page is displayed.

Modify the **Transcoding Template** parameter in the **Transcoding Settings** area. Select all transcoding templates created in **Step 2** from the drop-down list.

NOTICE

After the value of **Transcoding Template** of the channel is changed, the ingest URL may change. In this case, you need to use the new ingest URL.

Step 5 Click **OK**.

----End