

Video on Demand

Best Practices

Issue 01
Date 2026-03-27



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

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 Uploading a Media File to VOD.....	1
2 Setting a Video Thumbnail.....	5
3 Configuring Hotlink Protection to Control Who Can Play Media.....	8
4 Protecting Videos with HLS Encryption.....	16
5 Change History.....	24

1 Uploading a Media File to VOD

Scenarios

With the popularization of video services, a large number of media files need to be processed and distributed on a platform. Huawei Cloud Video on Demand (VOD) is a one-stop media service that implements video upload, automatic transcoding, media file management, and distribution acceleration. Before using the media file processing functions of VOD, you need to upload media files to VOD. Huawei Cloud VOD provides diverse upload methods for source files of different storage modes.

Upload Methods

Table 1-1 describes the media file upload methods supported by VOD.

Table 1-1 Upload methods

Upload Method	Application Scenario
Local Upload	This method is used to migrate media files stored on local disks to VOD.
Replicating Media Files from OBS to VOD	This method is used to replicate media file copies from OBS buckets to VOD.
Pull from URLs	This method is used to pull and store online media files to VOD.

Local Upload

Media files can be uploaded in batches to VOD. You can log in to the console on a browser to upload media files.

The local upload function provided by VOD has the following restrictions:

- Due to security policies, the logout from the console will result in the failure of large file upload that takes a long time. When uploading a large number of files, you need to perform operations on the console to prevent automatic logout.

- The following formats are supported:
 - Video: MP4, TS, MOV, MXF, FLV, MPG, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, WEBM, RMVB, VOB, RM, MTS, DV, DAT, QT, M2T, SWF and M3U8. M3U8 files can be uploaded only by URL pull.
 - Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, MP2, RA, and CAF

Local upload from the console

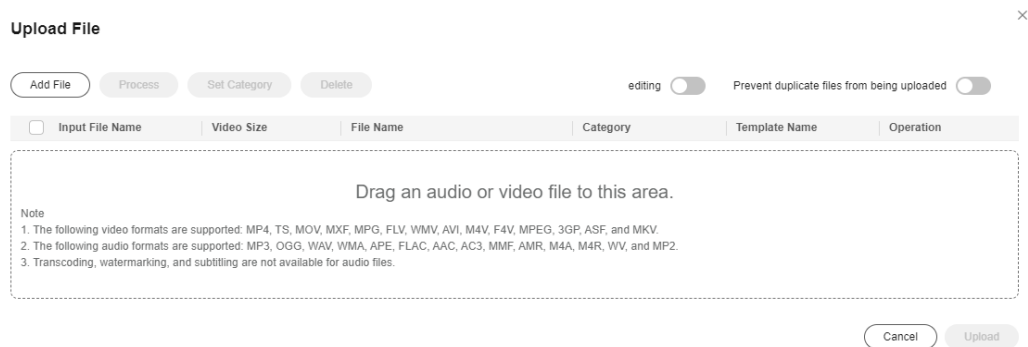
Step 1 Log in to the [VOD console](#).

Step 2 In the navigation pane, choose **Audio and Video Uploads > Local Upload**.

Step 3 Click **Upload File**. The **Upload File** dialog box is displayed.

Step 4 Click **Add File** to add a local media file, or directly drag a file to the file area, as shown in [Figure 1-1](#).

Figure 1-1 Local upload



Step 5 (Optional) Select a transcoding template or workflow using the **Process** button. After the media file is uploaded, the system automatically processes the uploaded file.

Step 6 Click **Upload**.

How long the upload takes depends on the file size and network conditions.

----End

Local upload using APIs

- If the media file to be uploaded is less than 20 MB, it can be uploaded directly. For details, see [Uploading a Media File Less Than 20 MB](#).
- If the media file to be uploaded is greater than 20 MB, it will be split into parts of less than 20 MB before being uploaded. For details, see [Uploading a Media File Greater Than 20 MB](#).

Replicating Media Files from OBS to VOD

If you have stored a large number of media files in an OBS bucket before subscribing to VOD and want to use the transcoding and snapshot capturing functions of VOD to process the media files, you can use this function to replicate the media file copies from the OBS bucket to VOD and then use the functions of VOD.

The function of replicating data from OBS to VOD has the following restrictions:

- Cross-region data replication is not supported. For example, media files stored in an OBS bucket of CN North-Beijing4 can only be replicated to the VOD in CN North-Beijing4.
- When you use data replication, media files in an OBS bucket are copied to VOD. If the media files in the OBS bucket are not deleted, you will be billed for the storage in both OBS and VOD.
- The following formats are supported:
 - Video: MP4, TS, MOV, MXF, FLV, MPG, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, WEBM, RMVB, VOB, RM, MTS, DV, DAT, QT, M2T, SWF and M3U8. M3U8 files can be uploaded only by URL pull.
 - Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, MP2, RA, and CAF

You can only replicate data from OBS to VOD by calling the API for [Replicating Media Files from OBS to VOD](#).

Pull from URLs

Online media files can be pulled and uploaded to VOD. The URL extraction function provided by VOD has the following restrictions:

- A maximum of 100 media files can be pulled at a time on the console, and a maximum of 16 media files can be pulled at a time using APIs.
- The URL to be pulled must directly point to media files and cannot be a page URL of a website. The following URL suffixes are supported:
 - Video: MP4, TS, MOV, MXF, FLV, MPG, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, WEBM, RMVB, VOB, RM, MTS, DV, DAT, QT, M2T, SWF and M3U8. M3U8 files can be uploaded only by URL pull.
 - Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, MP2, RA, and CAF
- Currently, only HTTPS and HTTP are supported. HTTP has security risks. HTTPS is recommended.
- A new media file ID is generated for the media file obtained from other cloud service providers. The original media file ID cannot be inherited.

File pull and upload on the console

Step 1 Log in to the [VOD console](#).

Step 2 In the navigation pane, choose **Audio and Video Uploads > Pull from URLs**.

Step 3 Click **Pull from URLs**. The **Pull from URLs** page is displayed, as shown in [Figure 1-2](#).

Enter the information about the files to be pulled. You can also select a transcoding template or workflow using the **Process** option. After the media files are pulled, the system automatically processes the files.

Figure 1-2 Settings of pull from URLs

Back to URL List / Pull from URLs

1. Enter the audio and video URLs you want to obtain in the text box. Separate URLs by line breaks. Support pulling up to 100 audio and video at one time.
2. Video URLs with the following suffixes can be obtained: MP4,TS,MOV,MXF,FLV,MPEG,WMV,AVI,M4V,F4V,MPEG,3GP,ASF,MKV,WEBM,RMVB,M3U8.
3. Audio URLs with the following suffixes can be obtained: MP3,OGG,WAV,WMA,APE,FLAC,AAC,AC3,MMF,AMR,M4A,M4R,WV,MP2.

Pull File Upload an XLSX file. For details about the format, see [Batch Import Template](#) [Batch Set Categories](#)

URL	Audio and Video Name (Optional)	Storage Category	Operation
<input type="text"/>	<input type="text"/>	Other <input type="button" value="v"/>	Delete

[Add a row](#)

Process

Video Cover Use the first frame of the video as the cover

Step 4 Click **Confirm**. You can view the task status in the URL pull list.

----End

File pull and upload using APIs

1. Obtain the user token for API calling. For details, see [Constructing a Request](#).
2. Call the API for [Pulling Media Files from URLs](#) and configure the parameters for batch pulling media files from URLs and media file processing in the request parameters.
3. Call the API for [querying media files](#) to view the pull result.
Wait for one or two minutes (depending on the size of the video file) and then query the pull result.

2 Setting a Video Thumbnail

Scenarios

Video thumbnails not only look nicer but also help you search for the desired file if you have lots of video files in VOD. A thumbnail address is generated for the uploaded video thumbnail. The thumbnail and video file can be directly referenced to the web page.

When a video is uploaded, VOD captures the first frame of the first second as the thumbnail by default. You can also upload a custom image or use a captured snapshot as the video thumbnail.

- If you want an image that says what the video is about as the thumbnail, then upload a custom image. Before the upload, design a JPG or PNG thumbnail.
- If you want one frame of the video as the thumbnail, then use a captured snapshot. You do not need to prepare anything. You can directly use snapshot capturing to create snapshots.

Set a video thumbnail in one of the following ways:

- [Set a Video Thumbnail on the VOD Console](#)
- [Set a Video Thumbnail by Calling a VOD API](#)

Setting a Video Thumbnail on the VOD Console

You can upload a thumbnail on the **Audio and Video Management** page of the VOD console.

1. Log in to the [VOD console](#).
2. In the navigation pane, choose **Audio and Video Management**.
3. Click **Details** in the row containing the target video file. On the **Basic Information** tab, click **Edit** on the right.
4. Click the plus sign on the right of **Thumbnail** to upload a local thumbnail image. Confirm your thumbnail selection and click **OK**. Then, click **Save**.

You can also configure a video snapshot as the thumbnail on the **Video Processing > Snapshots** page of the VOD console.

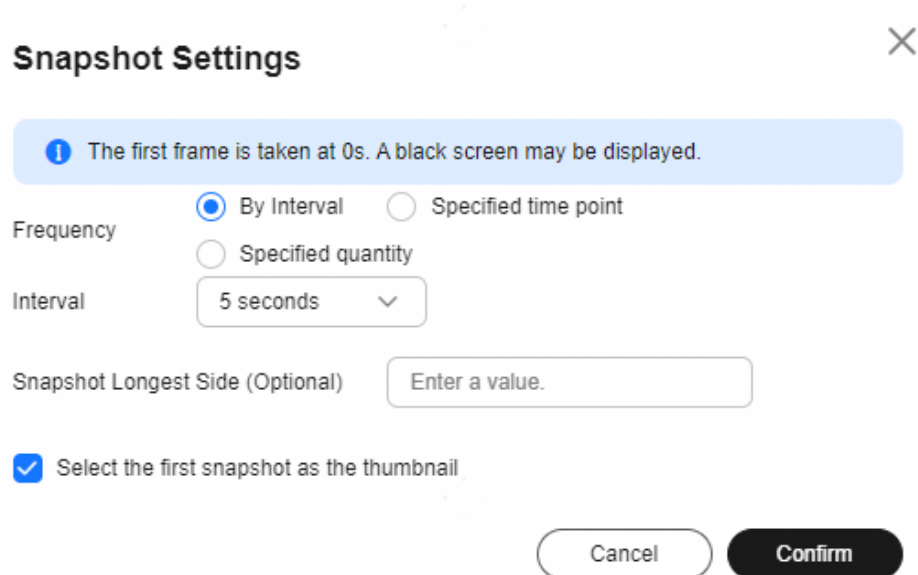
1. Log in to the [VOD console](#).

2. In the navigation pane, choose **Video Processing > Snapshots**.
3. Select the video for which you want to take a snapshot and click **Create Snapshot Task**. In the dialog box displayed, set snapshot parameters.
 - Taking snapshots by time interval: The system takes snapshots at regular intervals from the first frame to the last frame. The interval cannot exceed 12 seconds.
 - Taking snapshots at fixed time: The system takes snapshots at fixed time points. A maximum of 10 time points can be configured for a video.
 - Taking snapshots by total count: The system takes snapshots at equal intervals throughout video based on the specified count.

Please note that:

- You can take snapshots from an MP4, TS, MOV, MXF, FLV, MPG, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, WEBM, RMVB, VOB, RM, MTS, DV, DAT, QT, M2T, or SWF video.
- Snapshots are saved as JPG files.

Figure 2-1 Snapshot settings



4. Click **Confirm**.
5. Click **Details** and select a snapshot as the video thumbnail.

Setting a Video Thumbnail by Calling a VOD API

You can call media upload APIs, API for updating a media file, or API for processing a media file to set a video thumbnail.

- Set the thumbnail when uploading a video.

You can directly upload media files to VOD, replicate media files from OBS to VOD, or pull media files to VOD from URLs. If you use direct upload, you can set a custom image or snapshot as the thumbnail. If you use one of other two methods, you can only set a snapshot as the thumbnail. The details are as follows:

- Directly upload media files to VOD.
Call the API for [Uploading Media Files to VOD](#). Send the request parameters including **cover_type** and obtain **cover_upload_url** from the response parameters. Then upload a custom thumbnail via **cover_upload_url**.
Call the API for [Uploading Media Files to VOD](#). Set **thumbnail** in the request parameters, including snapshot parameters and which snapshot serves as the thumbnail.
- Replicate media files from OBS to VOD, or pull media files to VOD from URLs.
Call the API for [Dumping Media Assets to OBS](#) or the API for [Pulling Media Files from URLs](#). Configure **thumbnail** in the request parameters, set the snapshot type, and specify a snapshot as the thumbnail.
- Set the thumbnail when updating a video.
Call the API for [Updating a Media File](#). Send the request parameters including **cover_type** and obtain **cover_upload_url** from the response parameters. Then upload a custom thumbnail via **cover_upload_url**.
- Set the thumbnail when processing a video.
Call the API for [Processing a Media File](#) to generate snapshots and specify which snapshot as the thumbnail. If you need to change the thumbnail, call the API for querying file details to obtain the thumbnail URL, and then call the API for [Setting a Thumbnail](#).

3 Configuring Hotlink Protection to Control Who Can Play Media

Scenarios

VOD provides hotlink protection to control who can play the distributed media file. With hotlink protection enabled, CDN verifies key information carried in playback requests. Only the requests that pass the verification are responded. For other illegitimate requests, a status code 403 is returned. Hotlink protection is implemented by referer validation or URL validation.

Referer validation allows you to control access sources based on the referer field carried in an HTTP request. CDN filters requests based on the configured blacklist or whitelist. Referer validation is easy to configure, requires no extra development, and takes effect quickly. It is used for scenarios where audios and videos are mainly referenced on the web pages.

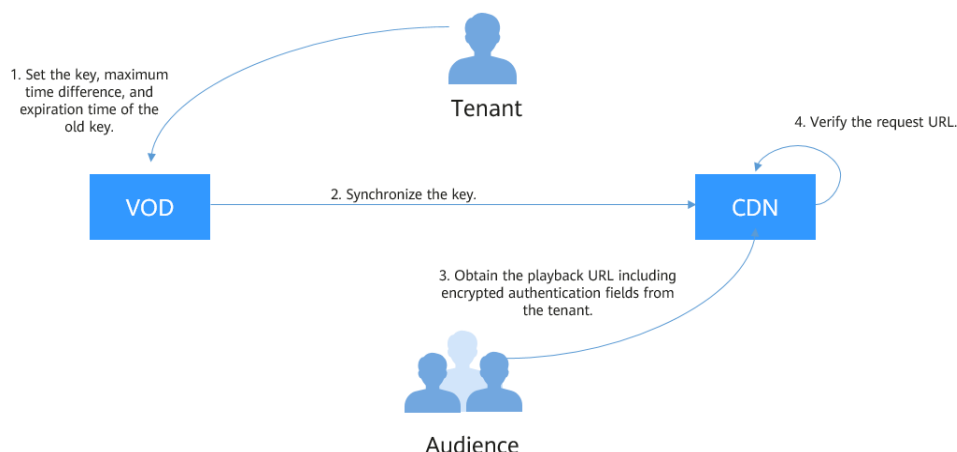
Because the HTTP header content can be forged, referer validation can only achieve the most basic protection, and the security is low. In this case, you can configure URL validation to safeguard your VOD assets. The key value for authentication is time-sensitive. Therefore, URL validation is used for scenarios that have high requirements on media security.

In this example, referer validation is enabled. Only domain names in the whitelist are allowed to access video files. Then configure URL validation to create authentication playback URLs.

Implementation

Referer validation works in a simple way. After a blacklist or whitelist is configured on the VOD console, VOD distributes the blacklist or whitelist to CDN. When receiving a request, CDN checks whether the request is valid based on the list. If the request is valid, CDN accesses the requested resource. If the request is invalid, CDN rejects the request and returns a status code 403.

URL validation is implemented by VOD edge nodes and origin server in VOD. It is a more secure and reliable anti-piracy solution than referer validation. [Figure 3-1](#) shows how URL validation works.

Figure 3-1 URL validation working principles

The process is as follows:

1. You enable URL validation on the VOD console and configure the allowed time difference and algorithm.
2. VOD delivers the configured key value to CDN nodes.
3. You obtain the authentication URL of a VOD media file.
4. Viewers request CDN to play a video through the authentication playback URL.
5. CDN verifies the request based on authentication information carried in the playback URL. Only requests that pass the verification are allowed.

Configuring Referrer Validation

Configure referer validation to limit access for basic security of VOD resources.

NOTE

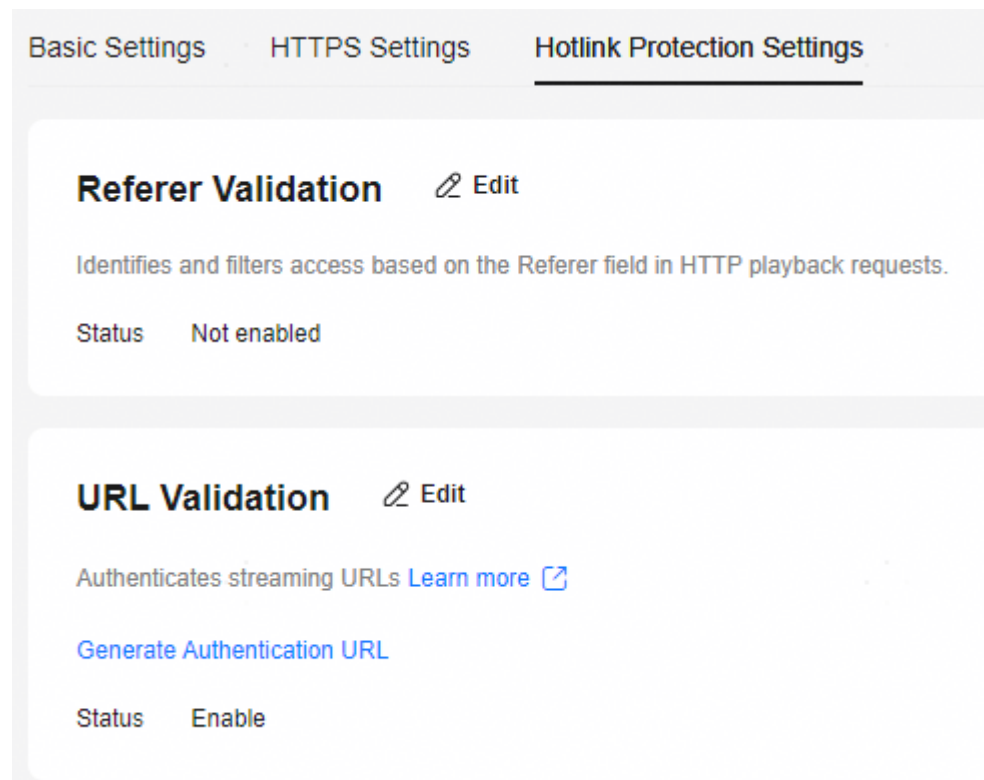
Domain names with ports cannot be added to referer whitelists/blacklists.

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

Step 2 In the navigation pane, choose **Domain Name Management**.

Step 3 Locate the desired domain name and click **Configure** in the **Operation** column. On the displayed page, choose the **Hotlink Protection Settings** tab.

Figure 3-2 Hotlink protection settings



Step 4 Click **Edit** on the right of **Referer Validation**. The **Referer Validation** dialog box is displayed, as shown in [Figure 3-3](#).

Configure referer validation parameters by referring to [Table 3-1](#).

Figure 3-3 Referrer validation settings

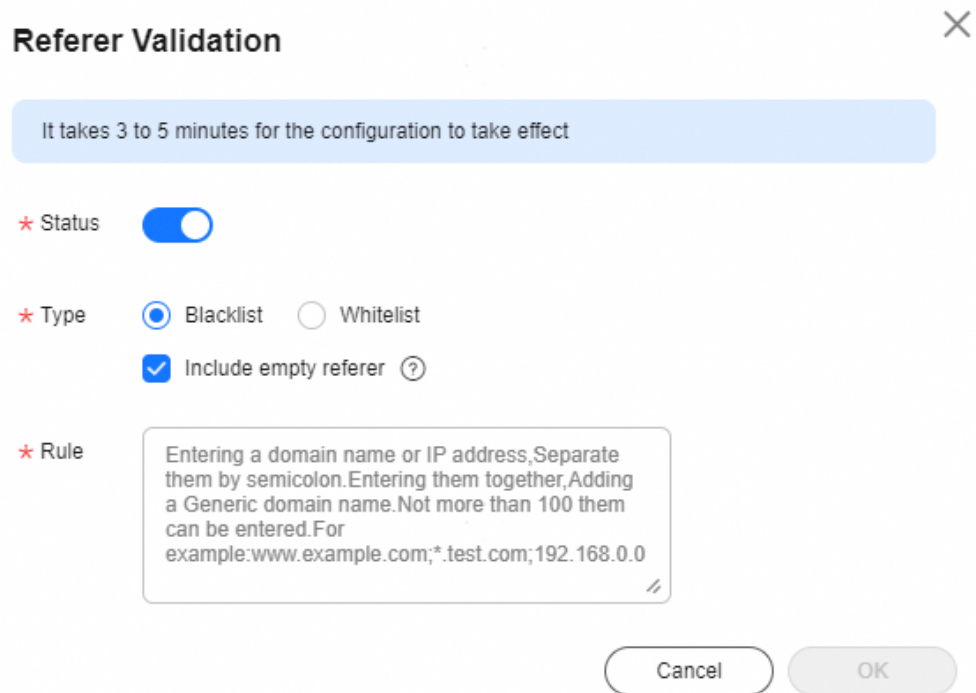


Table 3-1 Parameters

Parameter	Description
Status	Whether to enable referer validation. By default, referer validation is enabled.
Type	Blacklists and whitelists are supported. <ul style="list-style-type: none"> • Blacklist: Domains on the blacklist are blocked from accessing VOD resources. All other domains are allowed. If Include empty referer is selected, requests with empty referer in the HTTP header are also blocked. • Whitelist: Only domains on the whitelist are allowed to access VOD resources. All other domains are blocked. If Include empty referer is selected, requests with empty referer in the HTTP header are also allowed.
Rule	Domain names in the blacklist or whitelist. Domain names and IP addresses can be input at the same time and separated with semicolons (;). Wildcard domain names are allowed. A maximum of 100 domain names and IP addresses in total can be entered. Example: www.example.com;*.test.com;192.168.0.0

Step 5 Click **OK**.

It takes about 3 to 5 minutes for the referer validation settings to take effect.

----End

Configuring URL Validation

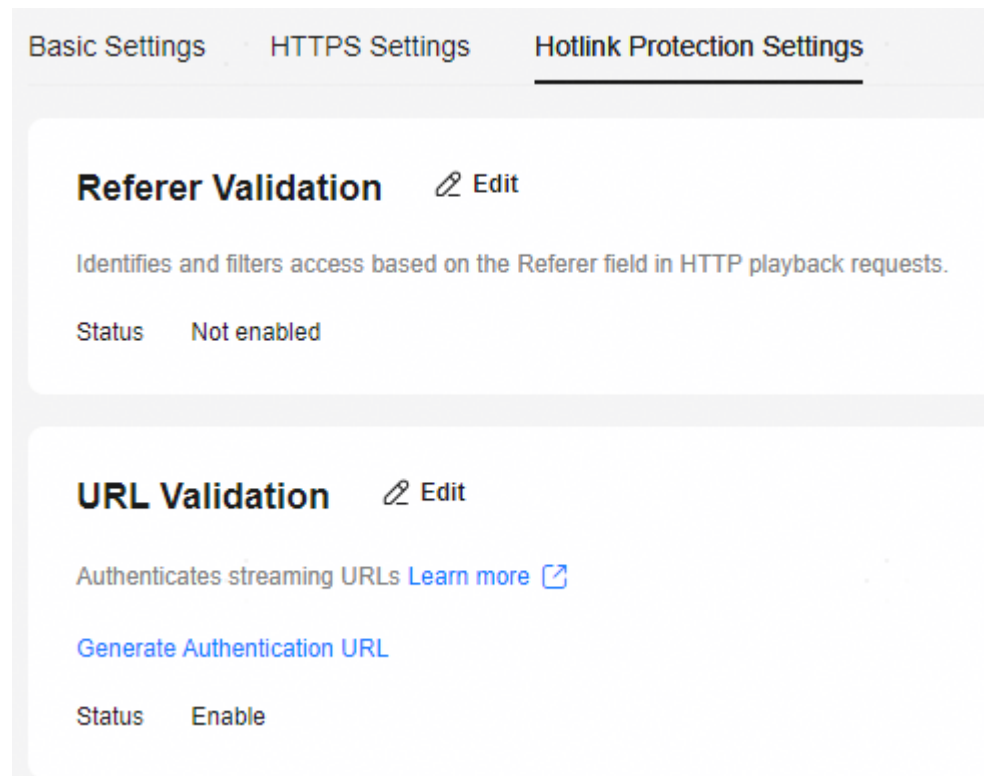
Configure URL validation to further enhance the security of VOD resources.

Step 1 Log in to the [VOD console](#).

Step 2 In the navigation pane, choose **Domain Name Management**.

Step 3 Locate the desired domain name and click **Configure** in the **Operation** column. On the displayed page, choose the **Hotlink Protection Settings** tab.

Figure 3-4 Hotlink protection settings



Step 4 Click **Edit** on the right of **URL Validation**. The **URL Validation** dialog box is displayed, as shown in [Figure 3-5](#).

Configure URL validation settings based on [Table 3-2](#).

Figure 3-5 Configuring URL validation

URL Validation ✕

1. Algorithm ABCD are historically insecure algorithms. It is recommended that users choose Algorithm E.

2. Algorithm E is recommended in the real-time encapsulation scenario. For DASH, only the primary index can be authenticated. The authentication scope cannot be all files.

Status

★ Signing Key ? 👁 Generate

You are advised to click the button on the right to generate the key. If the key is not random enough, the security of the authentication information cannot be ensured.

Secondary Key 👁 Generate

You are advised to click the button on the right to generate the key. If the key is not random enough, the security of the authentication information cannot be ensured.

★ Validity Period (s) s

★ Algorithm ▼

Algorithm A is an insecure algorithm, please choose with caution.

Sign Algorithm ▼

The signature algorithm has been modified, requiring adaptation for generating authentication URLs. For details, refer to the [documentation](#).

Cancel
OK

Table 3-2 Parameter description

Parameter	Description
Key	Click Generate to generate a key value.
Maximum Time Difference	How long an authentication URL remains valid. The default value is 120 minutes. For example, if the authentication URL generation time is 1573806090 (Nov. 15, 2019 16:21:30 GMT+08:00) and the allowed time difference is 120 minutes, the authentication URL expires at Nov. 15, 2019 18:21:30 GMT+08:00.
Expiration Time of the Old Key	By default, the old key expires 60 minutes later since the new key takes effect. For example, if the effective time of the new key is Nov. 15, 2019 16:21:30 GMT+08:00 and Expiration Time of the Old Key is 60 minutes, the old authentication URL expires at Nov. 15, 2019 17:21:30 GMT+08:00.

Parameter	Description
Algorithm	<p>Four algorithms are supported: A, B, C, D, and E. The generated authentication URL varies depending on the selected algorithm. For details about how to create an authentication URL, see URL Validation.</p> <p>NOTE Algorithms A, B, and C do not support HLS or DASH playback. For these formats, algorithm D or E is recommended.</p>

Step 5 Click **OK**.

Step 6 If you select algorithm D, you need to [submit a service ticket](#) for approval after configuring the parameters. The submitted information must contain the configured domain name and information listed in [Table 3-2](#).

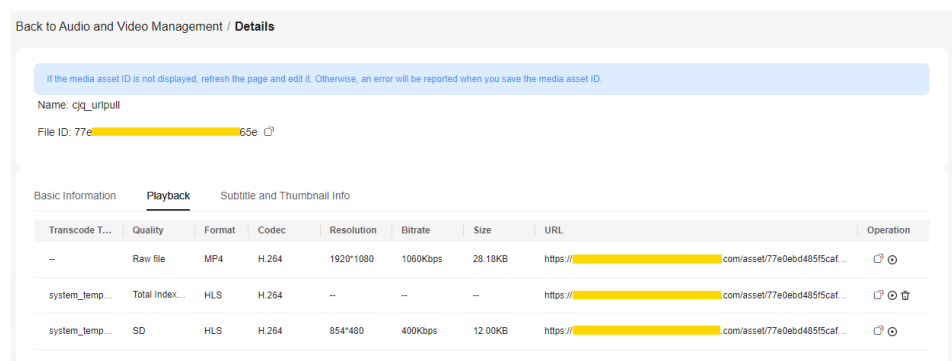
URL validation settings take effect once your request is approved. If the URL validation settings are modified, you need to [submit a service ticket](#) for approval again.

----End

Verifying Whether Hotlink Protection Has Taken Effect

- Verify whether referer validation settings have taken effect.
Add www.huaweicloud.com to the whitelist and deselect **Exclude empty referer**. Reference the video file in VOD <https://1280.cdn-vod.huaweicloud.com/input/1.mp4> on the <http://www.example.com/test/test.html> web page, access the web page, and play the video. If the playback fails, referer validation settings have taken effect.
- Verify whether URL validation settings have taken effect.
 - a. Log in to the [VOD console](#).
 - b. In the navigation pane, choose **Management > Audio and Video Management**.
 - c. Click **Details** in the row containing a media file and click the **Playback** tab to obtain the playback URL.

Figure 3-6 Playback URL



URL is the original playback URL. Click  to obtain the authentication playback URL.

- d. Play the original playback URL and authentication playback URL on the player. If the original playback URL fails to be played but the authentication playback URL can be played, URL validation settings have taken effect.

4 Protecting Videos with HLS Encryption

Environment Preparation

You have [downloaded the SDK](#).

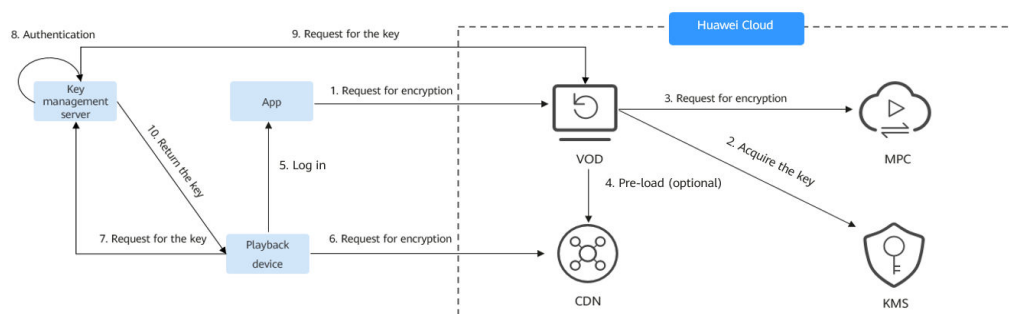
Scenarios

Hotlink protection prevents unauthorized users from downloading or playing VOD content. However, this cannot prevent malicious paid users from downloading the content to their local PCs for secondary distribution.

To address this issue, Huawei Cloud VOD provides HLS encryption. With HLS encryption enabled, encrypted videos cannot be distributed to others even if they are downloaded by malicious users. HLS encryption requires the key service and token generation service. Therefore, this solution is suitable for those who can deploy authentication and key management servers by themselves.

Implementation

HLS encryption provided by Huawei Cloud VOD uses the AES-128 encryption algorithm to encrypt each TS file. The generated M3U8 file describes how the player decrypts the TS file. All HLS players are supported.



In this solution, VOD integrates Huawei Cloud Key Management Service (KMS) to provide keys for HLS encryption.

- Encryption
 - a. You upload a video to VOD and request HLS encryption.

- b. VOD requests the encryption key from KMS and stores the obtained key ID and key ciphertext.
- c. VOD sends an HLS encryption request to Media Processing Center (MPC). MPC encrypts the video during transcoding.

The M3U8 file generated after transcoding contains the **#EXT-X-KEY** tag, which contains the **METHOD** and **URI** attributes. **URI** is the address of your key management server.

 **NOTE**

If the playback URL after encryption is **HTTPS**, the KMS URL must also be HTTPS. Otherwise, the video cannot be previewed on the VOD console.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-KEY:METHOD=AES-128,URI="https://domain-sample/encrypt/get-key?
asset_id=6aee80009c4ca6970f508d6334194794",IV=0x80a3ff24ccd788042ca7f2237e74c59d
#EXTINF:5.000000, 6aee80009c4ca6970f508d6334194794_1_1920X1080_3000_0_0.ts
#EXTINF:5.000000, 6aee80009c4ca6970f508d6334194794_1_1920X1080_3000_0_1.ts
#EXT-X-ENDLIST
```

- d. VOD uses CDN to accelerate the distribution of encrypted HLS video files.
- Decryption

- a. When an end user logs in to a player, your server verifies user identity. If the verification succeeds, your server allocates a token and returns the playback URL containing the token to the player.

For example, if the video playback URL is **https://1280.cdn-vod.huaweicloud.com/input/test.m3u8**, the playback URL returned to the player is **https://1280.cdn-vod.huaweicloud.com/input/test.m3u8?token={token}**.

- b. The player sends a playback request containing the obtained playback URL to CDN. The token is dynamic. Therefore, after receiving the request, CDN directly retrieves the content from VOD. VOD writes the token in the request URL to the **URI** of the M3U8 file.

The M3U8 file returned by VOD to CDN contains the token value of the player. The following is an example:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-KEY:METHOD=AES-128,URI="https://domain-sample/encrypt/get-key?
asset_id=6aee80009c4ca6970f508d6334194794&token={token}",IV=0x80a3ff24ccd788042ca7f22
37e74c59d
#EXTINF:5.000000, 6aee80009c4ca6970f508d6334194794_1_1920X1080_3000_0_0.ts
#EXTINF:5.000000, 6aee80009c4ca6970f508d6334194794_1_1920X1080_3000_0_1.ts
#EXT-X-ENDLIST
```

- c. The player parses the M3U8 file to obtain the **URI** in the **EXT-X-KEY** tag and requests the key.
- d. Your key management server verifies the token. If the token is valid, the server calls the **key query API**.

Your server can cache the key locally. If another playback device requests the key, it can directly return the key without obtaining the key from VOD.

- e. Your server returns the key to the player. The player uses the key to decrypt the M3U8 file and start playing the media.

Building Related Services

To use HLS encryption, you need to deploy the key management server and token generation service.

- The key management server needs to have the following functions. See [Sample Code](#).
 - Identity authentication. As described in [Implementation](#), your key management server verifies whether the token is valid upon receiving a key request.
 - Key acquisition from VOD. Keys are stored in VOD. Therefore, your key management server needs to call a VOD API to obtain the key.
 - Key acquisition from cache. Your key management server should have the cache function to cache the obtained key to avoid frequent access to VOD.
- Token generation service. When an end user logs in to your playback device, your server verifies the identity, generates a token, and returns the playback URL containing the token to the playback device. For details about the sample code for generating a token, see [Sample Code](#).

The generated token must contain uppercase letters, lowercase letters, and digits. The length can be customized. A unique token is allocated for each login and the token has a validity period. Comply with the principle of least privilege. It is recommended that the token be used only for HLS-encrypted videos.

Video Encryption

Step 1 Upload the video file to be encrypted.

If the video to be encrypted has not been uploaded, you can upload the video to VOD on the [console](#).

Step 2 Configure the key URL.

Before encryption, add the address of the key management server built in [Building Related Services](#) to VOD. During encryption, the address is written to the generated M3U8 file.

1. Log in to the [VOD console](#). In the navigation pane, choose **Global Settings > Security**. The **Security Settings** page is displayed.
2. Click **HLS Encryption Settings**. In the dialog box displayed, enter the URL of your key management server, for example, **https://domain-sample/encrypt/get-key**.
3. Click **OK**.

Step 3 Create a transcoding template.

HLS encryption is implemented during transcoding. Therefore, before transcoding, you need to create a transcoding template with encryption enabled.

1. In the navigation pane, choose **Global Settings > Transcoding Templates**.
2. Click **Create Custom Template Group**. On the displayed page, configure related parameters.

Figure 4-1 Setting basic information

The screenshot shows a 'Basic Information' configuration form. It contains the following fields and options:

- Name:** A text input field with a placeholder 'Enter a template name.' and a help icon.
- Description:** A larger text input field.
- Output Format:** Radio buttons for HLS (selected), DASH, DASH_HLS, MP4, MP3, and ADTS.
- Package Format:** Radio buttons for TS (selected) and FMP4.
- Segment Duration (s):** A dropdown menu showing '5' and a unit 's'.
- Maximum I-frame interval:** A text input field with the value 'Between 2 and 10, number'.
- Encryption:** A toggle switch that is currently turned off.

Below the Encryption toggle, there is a note: 'Before enabling this function, you need to configure the private key URL on the HLS Encryption Settings page. Learn how to configure HLS encryption?'

In the **Basic Information** area, set **Output Format** to **HLS**, enable encryption, and set other parameters based on your needs. For details, see [Transcoding Settings](#).

3. Click **OK**.

Step 4 Encrypt the video.

1. Log in to the [VOD console](#).
2. In the navigation pane, choose **Management > Audio and Video Management**.
3. Select the video that requires HLS encryption and click **Transcode**.
4. In the dialog box displayed, select the transcoding template created in [step 3](#) and click **OK**.

The system starts transcoding the video. If **Transcoding Status** in the **Status** column becomes **Completed**, the transcoding succeeds and the video has been encrypted.

----End

Video Playback

This solution uses the HLS standard for encryption. All players that support HLS can decrypt and play content.

1. Log in to the [VOD console](#).
2. In the navigation pane, choose **Management > Audio and Video Management**.
3. Click **Details** in the row contains the encrypted video and then click the **Playback** tab.


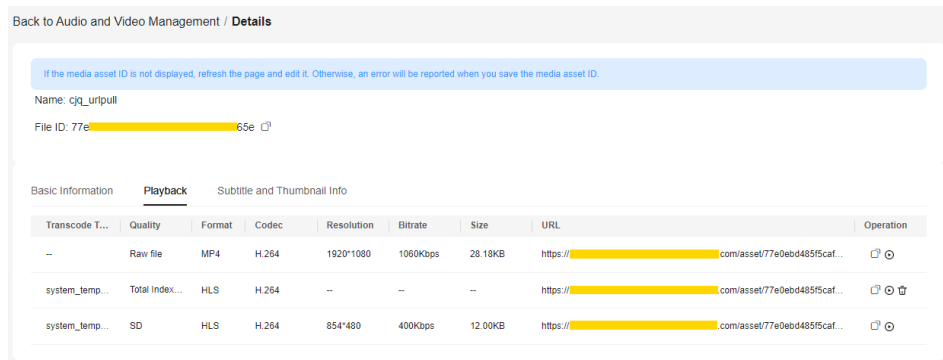
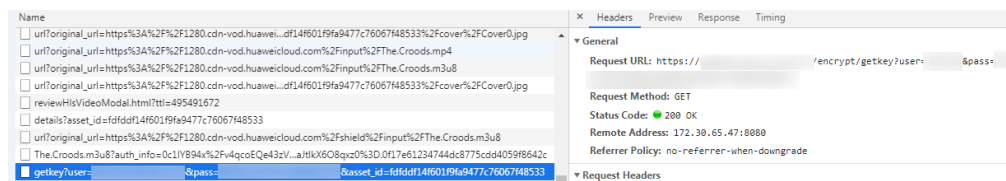
- Click  to play the video.

Figure 4-2 Playback URL



- Enable developer mode in your browser. You can see that the console requests the key using the configured URL and decrypts the video for playback.

Figure 4-3 Browser developer mode



Sample Code

- Sample code for key management server

In this example, a Universally Unique Identifier (UUID) is used to generate a token. You can also select a generation method. The sample code does not include validity check for login users, which you can do on your own if necessary.

When your key management server receives a key request, it checks whether the decryption key is stored in the cache. If not, the server calls the server SDK to query the key.

```
import java.util.Base64;
import java.util.UUID;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang3.StringUtils;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.huawei.kms.initvodclient.VodClientFactory;
import com.huawei.kms.util.CacheUtils;
import com.huaweicloud.sdk.vod.v1.model.ShowAssetCipherRequest;

import retrofit2.http.Header;

@RestController
public class KeyManagerController {
    /**
     * Assign a token to a legitimate user and return the playback URL with the token.
     *
     * @param accessToken Authentication information carried with the user. Verify the identity.
     */
}
```

```

* @param playUrl Playback URL
* @return Return the playback URL with the token.
*/
@GetMapping("/get-url")
public String getTokenPlayUrl(@Header("access-token") String accessToken,
    @RequestParam(value = "play_url", required = true) String playUrl) {
    // Assign a token to a valid terminal. ***** needs to be generated by the code at the customer.
    String token = "*****";
    // Construct the playback URL with the token included and return it. http://{domain}/asset/
    {asset_id}/play_video/index.m3u8?token={token}
    return playUrl.substring(0, playUrl.lastIndexOf("/") + 1) +
        playUrl.substring(playUrl.lastIndexOf("/") + 1) + "?token=" + token;
}

/**
* @param asset_id Media ID
* @param token Token assigned to the user. The token needs to be verified. The key is returned
only to the device that passes the verification.
* @param response
* @return Return the key of the byte array type.
*/
@GetMapping(value = "/get-key", headers = "Accept=application/octet-stream")
public byte[] getKey(@RequestParam(value = "asset_id", required = true) String asset_id,
    @RequestParam(value = "token", required = true) String token, HttpServletResponse
response) {
    // Get the key from the cache. If it is not in the cache, obtain it from VOD. In this example,
suppose you store the key in the cache database.
    String key = CacheUtils.getCipherFromCache(asset_id);
    if (StringUtils.isEmpty(key)) {
        ShowAssetCipherRequest request = new ShowAssetCipherRequest();
        request.withAssetId(asset_id);
        key = VodClientFactory.getClient().showAssetCipher(request).getDk();
        // Cross-domain. Enter the actual site or fill in *.
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Content-Length", "16");
        // Set the data type of the key.
        response.setHeader("Content-Type", "application/octet-stream");
        // Update the cache.
        CacheUtils.updateCipherFromCache(asset_id, key);
    }
    return Base64.getDecoder().decode(key);
}
}

```

- Obtaining the VodClient Sample Code

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.vod.v1.VodClient;
import com.huaweicloud.sdk.vod.v1.region.VodRegion;

public class VodClientFactory {

    private final static String AK = System.getenv("CLOUD_SDK_AK");
    private final static String SK = System.getenv("CLOUD_SDK_SK");
    private final static String REGION="cn-north-4"; // Actual service node, for example, cn-north-1
and cn-east-2

    private static volatile VodClient vodClient = null;

    public static VodClient getClient() {
        if (vodClient == null) {
            synchronized (VodClient.class) {
                if (vodClient == null) {
                    ICredential auth = new BasicCredentials()
                        .withAk(AK)
                        .withSk(SK);
                    vodClient = VodClient.newBuilder().withCredential(auth)
                        .withRegion(VodRegion.valueOf(REGION))
                        .build();
                }
            }
        }
    }
}

```

```
    }  
  }  
  return vodClient;  
}
```

- Sample code for caching

After your key management server obtains the decryption key from VOD, it needs to cache the key to avoid repeated requests for the same media from VOD. In this example, suppose you cache the key on your local PC. You can also cache it in the database.

```
import com.google.common.cache.Cache;  
import com.google.common.cache.CacheBuilder;  
  
import java.util.concurrent.TimeUnit;  
  
public class CacheUtils {  
  
    private static Cache<String, String> cipherCache = CacheBuilder.newBuilder()  
        .maximumSize(100) // Set the maximum cache size.  
        .expireAfterWrite(10, TimeUnit.MINUTES) // Set the cache to be invalid one minute after data  
is written.  
        .concurrencyLevel(10) // Set the concurrency level to 10.  
        .recordStats() // Enable cache statistics.  
        .build();  
  
    public static String getCipherFromCache(String key) {  
        return cipherCache.getIfPresent(key);  
    }  
  
    public static void updateCipherFromCache(String key, String value) {  
        cipherCache.put(key, value);  
    }  
}
```

- The Maven dependency required by the preceding sample code is as follows:

Note: The following JAR package versions are not fixed. Use the actual versions of the Java project and JAR packages.

```
<parent>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-parent</artifactId>  
  <version>2.3.12.RELEASE</version>  
  <relativePath/>  
</parent>  
<dependencies>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-web</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>org.apache.commons</groupId>  
    <artifactId>commons-lang3</artifactId>  
    <version>3.7</version>  
  </dependency>  
  <dependency>  
    <groupId>com.huaweicloud.sdk</groupId>  
    <artifactId>huaweicloud-sdk-vod</artifactId>  
    <version>3.1.72</version>  
  </dependency>  
  <dependency>  
    <groupId>com.squareup.retrofit2</groupId>  
    <artifactId>retrofit</artifactId>  
    <version>2.5.0</version>  
  </dependency>  
  <dependency>  
    <groupId>com.google.guava</groupId>
```

```
<artifactId>guava</artifactId>  
<version>27.0.1-jre</version>  
</dependency>  
</dependencies>
```

5 Change History

Released On	Change Description
2020-08-30	This issue is the second official release. <ul style="list-style-type: none">• Added the section "Uploading a Media File to VOD".
2019-03-30	This issue is the first official release.