Video on Demand

API Reference

Issue 01

Date 2025-02-11

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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 Before You Start

Video on Demand (VOD) is a one-stop solution that uploads, transcodes, and manages media resources and distributes them to your users. VOD offers premium media processing to help you quickly build a secure and scalable VOD platform, eliminating the hassle of managing underlying infrastructure.

This document describes how to use application programming interfaces (APIs) to perform operations on VOD, such as uploading, transcoding, categorizing, and managing audio/video. **API Overview** lists all supported operations.

If you plan to access VOD through an API, ensure that you are familiar with VOD concepts. For details, see **Service Overview**.

Endpoints

VOD supports Representational State Transfer (REST) APIs, allowing you to call APIs using HTTPS. For details, see **Calling APIs**.

An endpoint is the request address for calling an API. Endpoints vary depending on regions. For the endpoints of all services, see **Regions and Endpoints**.

In addition, VOD provides **software development kit (SDK)** in multiple programming languages.

APIs Supporting Cross-Origin Requests

See **Table 1-1**.

Table 1-1 APIs supporting cross-origin requests

API	URI
Uploading Media Assets to VOD	/v1.0/{project_id}/asset
Confirming Media Asset Upload	/v1.0/{project_id}/asset/status/uploaded
Verifying the Upload	/v1.0/{project_id}/asset/duplication

API	URI
Querying Media Assets	/v1.0/{project_id}/asset/info
Obtaining Authorization for Multipart Upload	/v1.0/{project_id}/asset/authority

API Request Throttling

API request throttling is configured for VOD to prevent service interruption caused by repeated API calls in a short period. See Table 1-2.

Table 1-2 API request throttling

Category	API	API Calls for a Single Tenant Per Minute	API Calls for All Tenants Per Minute
Media upload	• Uploading media files to VOD • Obtaining upload authorization • Confirming media upload • Replicating media files in OBS to VOD	1,500	12,000
	• Pulling media files from URLs • Verifying the upload	100	1,000
Media processing	• Updating a video • Media processing • Canceling a media asset transcoding task • Extracting audio • Canceling an audio extraction task • Creating a media asset review task • Setting a thumbnail	100	1,000

Category	API	API Calls for a Single Tenant Per Minute	API Calls for All Tenants Per Minute
Media management	• Deleting media files • Publishing media files • Canceling media publish • Modifying file attributes	100	1,000
	Querying file information	1,500	24,000
	• Querying file details • Querying media files	1,500	12,000
Media pre-loading	• CDN pre-loading • Querying pre-loading results	100	1,000
Media category	• Creating a media category • Modifying a media category • Deleting a media category • Querying media categories	100	1,000
Key query	Key query	1,500	12,000
Statistical analysis	• Querying CDN statistics • Querying origin server statistics • Querying the most requested content • Querying playback logs of a domain name • Querying daily playback statistics of a media asset	100	1,000

Category	API	API Calls for a Single Tenant Per Minute	API Calls for All Tenants Per Minute
Watermark template management	• Creating a watermark template • Modifying a watermark template • Querying watermark templates • Deleting a watermark template • Confirming watermark image upload	100	1,000
Subtitle management	Managing subtitles	100	1,000
Transcoding template management	• Creating a custom transcoding template • Querying transcoding templates • Updating a transcoding template • Deleting a custom template	100	1,000
Transcoding template set management	• Creating a transcoding template group set • Modifying a transcoding template group set • Querying custom template group sets • Deleting a transcoding template group set	100	1,000

Concepts

- Account

  A domain is created upon successful registration. The domain has full access permissions for all of its cloud services and resources. It can be used to reset user passwords and grant user permissions. The account is a payment entity and should not be used to perform routine management. For security purposes, create IAM users and grant them permissions for routine management.

- IAM user

  An IAM user is created using an account to use cloud services. Each IAM user has its own identity credentials (password and access keys).

  An IAM user can view the account ID and user ID on the My Credentials page of the console. The domain name, username, and password will be required for API authentication.

- Region

  A region is a physical location where cloud resources are deployed. Availability zones (AZs) in the same region can communicate with each other over an intranet but AZs in different regions cannot communicate with each other. For low network latency and quick resource access, select the nearest region. The choice of regions may also be subject to legal compliance requirements.

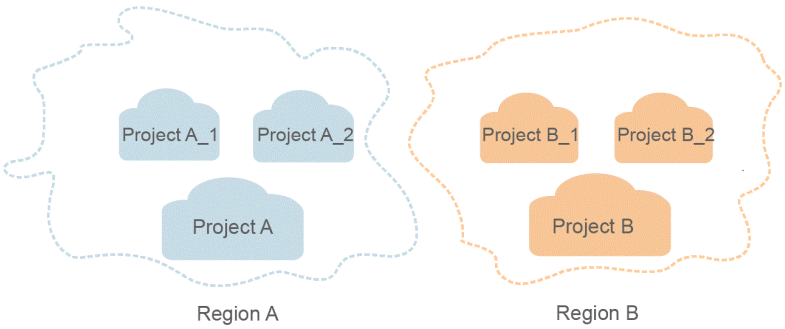
- AZ

  An AZ is one or more physical data centers within a region. It has independent cooling, fire extinguishing, moisture-proof, and electricity facilities. Within an AZ, compute, networking, storage, and other resources are logically divided into multiple clusters. AZs within a region are connected using high-speed optical fibers for high availability.

- Project

  Projects group and isolate resources (including compute, storage, and network resources) across physical regions. A default project is provided for each region, and subprojects can be created under each default project. Users can be granted permissions to access all resources in a specific project. For more refined access control, create subprojects under a project and purchase resources in the subprojects. Users can then be assigned permissions to access only specific resources in the subprojects.

  Figure 1-1 Project isolating model



2 API Overview

The following tables list the VOD APIs. Before calling a VOD API, you need to obtain a user token. The obtained token can then be used to authenticate the calling of other APIs. For details about how to call a VOD API, see **Application Examples**.

Media Asset Upload

API	Description
POST /v1.0/{project_id}/asset	**Uploading Media Assets to VOD**
GET /v1.1/{project_id}/asset/authority	**Obtaining Authorization for Multipart Upload**
POST /v1.0/{project_id}/asset/status/uploaded	**Confirming Media Asset Upload**
PUT /v1.0/{project_id}/asset/authority	**Authorizing Access to OBS**
POST /v1.0/{project_id}/asset/reproduction	**Replicating Media Assets from OBS to VOD**
POST /v1.0/{project_id}/asset/upload_by_url	**Pulling Media Assets from URLs**
GET /v1.0/{project_id}/asset/duplication	**Verifying the Upload**

Media Asset Processing

API	Description
PUT /v1.0/{project_id}/asset	**Updating a Video**
POST /v1.0/{project_id}/asset/process	**Processing a Media Asset**
DELETE /v1.0/{project_id}/asset/process	**Canceling a Transcoding Task**

API	Description
POST /v1.0/{project_id}/asset/extract_audio	Extracting Audio
DELETE /v1.0/{project_id}/asset/extract_audio	Canceling an Audio Extraction Task
PUT /v1.0/{project_id}/asset/cover	Setting a Thumbnail

Media Asset Management

API	Description
DELETE /v1.0/{project_id}/asset	Deleting a Media Asset
POST /v1.0/{project_id}/asset/status/publish	Publishing a Media Asset
POST /v1.0/{project_id}/asset/status/unpublish	Canceling Media Asset Publish
GET /v1.0/{project_id}/asset/info	Querying Media Assets
PUT /v1.0/{project_id}/asset/info	Modifying Media Asset Attributes
GET /v1.0/{project_id}/asset/details	Querying Media Asset Details
GET /v1.0/{project_id}/asset/list	Querying Media Assets

Media Asset Prefetch

API	Description
POST /v1.0/{project_id}/asset/preheating	Prefetching Media Assets on CDN
GET /v1.0/{project_id}/asset/preheating	Querying CDN Prefetch Results

Media Asset Categorization

API	Description
POST /v1.0/{project_id}/asset/category	Creating a Media Asset Category
PUT /v1.0/{project_id}/asset/category	Modifying a Media Asset Category

API	Description
DELETE /v1.0/{project_id}/asset/category	Deleting a Media Asset Category
GET /v1.0/{project_id}/asset/category	Querying Media Asset Categories

Key Query

API	Description
GET /v1.0/{project_id}/asset/ciphers	Querying Keys

Statistical Analysis

API	Description
GET /v1.0/{project_id}/asset/cdn-statistics	Querying CDN Statistics
GET /v1.0/{project_id}/asset/vod-statistics	Querying Origin Server Statistics
GET /v1.0/{project_id}/asset/top-statistics	Querying Most Requested Content
GET /v1.0/{project_id}/vod/cdn/logs	Querying Playback Logs of a Domain Name
GET /v1/{project_id}/asset/daily-summary	Querying Daily Playback Statistics of a Media Asset

Watermark Template Management

API	Description
POST /v1.0/{project_id}/template/watermark	Creating a Watermark Template
PUT /v1.0/{project_id}/template/watermark	Modifying a Watermark Template
GET /v1.0/{project_id}/template/watermark	Querying Watermark Templates
DELETE /v1.0/{project_id}/template/watermark	Deleting a Watermark Template
POST /v1.0/{project_id}/watermark/status/uploaded	Confirming Watermark Image Upload

Subtitle Management

API	Description
PUT /v1/{project_id}/asset/subtitles	Managing Subtitles

Transcoding Template Management

API	Description
POST /v2/{project_id}/asset/template/transcodings	Customizing a Transcoding Template
GET /v2/{project_id}/asset/template/transcodings	Querying Transcoding Templates
PUT /v2/{project_id}/asset/template/transcodings	Modifying a Transcoding Template
DELETE /v2/{project_id}/asset/template/transcodings	Deleting a Transcoding Template

Transcoding Template Set Management

API	Description
POST /v1.0/{project_id}/asset/template-collection/transcodings	Creating a Transcoding Template Group Set
PUT /v1.0/{project_id}/asset/template-collection/transcodings	Modifying a Transcoding Template Group Set
GET /v1.0/{project_id}/asset/template-collection/transcodings	Querying Transcoding Template Group Sets
DELETE /v1.0/{project_id}/asset/template-collection/transcodings	Deleting a Transcoding Template Group Set

Transcoded Output Management

API	Description
DELETE /v1/{project_id}/asset/transcode-product	Deleting a Transcoded Output

Media Asset Storage Mode Management

API	Description
PUT /v1/{project_id}/asset/storage-mode-type	**Modifying the Cold Storage Scope of a Media Asset**
PUT /v1/{project_id}/asset/storage-mode	**Changing the OBS Storage Class of a Media Asset**
GET /v1/{project_id}/asset/storage-mode-type	**Querying the Cold Storage Settings of a Media Asset**

3 Calling APIs

3.1 Making an API Request

This section describes the structure of a REST API request, and uses the IAM API for **creating an IAM User** as an example to demonstrate how to call an API. The obtained token can then be used to authenticate the calling of other APIs.

Request URI

A request URI is in the following format:

{URI-scheme}://{Endpoint}/{resource-path}?{query-string}

Although a request URI is included in the request header, most programming languages or frameworks require the request URI to be transmitted separately.

Table 3-1 URI parameter description

Parameter	Description
URI-scheme	Protocol used to transmit requests. All APIs use HTTPS.
Endpoint	Domain name or IP address of the server bearing the REST service. The endpoint varies between services in different regions. It can be obtained from **Regions and Endpoints**. For example, the endpoint of IAM in region **CN-Hong Kong** is **iam.ap-southeast-1.myhuaweicloud.com**.
resource-path	Access path of an API for performing a specified operation. Obtain the path from the URI of an API. For example, the **resource-path** of the API used to obtain a user token is **/v3/auth/tokens**.

Parameter	Description
query-string	Query parameter, which is optional. Ensure that a question mark (?) is included before each query parameter that is in the format of *Parameter name*=*Parameter value*. For example, **?limit=10** indicates that a maximum of 10 data records will be displayed.

IAM is a global service. You can create an IAM user using the endpoint of IAM in any region. For example, to create an IAM user in the **CN-Hong Kong** region, obtain the endpoint of IAM (**iam.ap-southeast-1.myhuaweicloud.com**) for this region and the **resource-path** (**/v3.0/OS-USER/users**) in the URI of the API for **creating an IAM user**. Then construct the URI as follows:

```
https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
```

Figure 3-1 Example URI

NOTE

To simplify the URI display in this document, each API is provided only with a **resource-path** and a request method. The **URI-scheme** of all APIs is **HTTPS**, and the endpoints of all APIs in the same region are identical.

Request Methods

The HTTP protocol defines the following request methods that can be used to send a request to the server.

Table 3-2 HTTP methods

Method	Description
GET	Requests the server to return specified resources.
PUT	Requests the server to update specified resources.
POST	Requests the server to add resources or perform special operations.
DELETE	Requests the server to delete specified resources, for example, an object.
HEAD	Same as GET except that the server must return only the response header.

Method	Description
PATCH	Requests the server to update partial content of a specified resource. If the resource does not exist, a new resource will be created.

For example, in the case of the API for **creating an IAM user**, the request method is **POST**. An example request is as follows:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
```

Request Header

You can also add additional header fields to a request, such as the fields required by a specified URI or HTTP method. For example, to request for the authentication information, add **Content-Type**, which specifies the request body type.

Common request header fields are as follows.

Table 3-3 Common request header fields

Parameter	Description	Mandatory	Example Value
Host	Specifies the server domain name and port number of the resources being requested. The value can be obtained from the URL of the service API. The value is in the format of *Hostname:Port number*. If the port number is not specified, the default port is used. The default port number for **https** is **443**.	No This field is mandatory for AK/SK authentication.	code.test.com or code.test.com:443
Content-Type	Specifies the type (or format) of the message body. The default value **application/json** is recommended. Other values of this field will be provided for specific APIs if any.	Yes	application/json
Content-Length	Specifies the length of the request body. The unit is byte.	No	3495

Parameter	Description	Mandatory	Example Value
X-Project-Id	Specifies the project ID. Obtain the project ID by following the instructions in **Obtaining a Project ID**.	No This field is mandatory for requests that use AK/SK authentication in the Dedicated Cloud (DeC) scenario or multi-project scenario.	e9993fc787d94b6c886cbaa340f9c0f4
X-Auth-Token	Specifies the user token. It is a response to the API for **obtaining a user token** (This is the only API that does not require authentication). After the request is processed, the value of **X-Subject-Token** in the response header is the token value.	No This field is mandatory for token authentication.	The following is part of an example token: MIIPAgYJKoZIhvcNAQcCo...ggg1BBIINPXsidG9rZ

NOTE

In addition to supporting authentication using tokens, APIs support authentication using AK/SK, which uses SDKs to sign a request. During the signature, the **Authorization** (signature authentication) and **X-Sdk-Date** (time when a request is sent) headers are automatically added in the request.

For more details, see "Authentication Using AK/SK" in **Authentication**.

The following shows an example request of the API for **creating an IAM user** when AK/SK authentication is used:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date, Signature=**********
```

(Optional) Request Body

This part is optional. A request body is generally sent in a structured format (for example, JSON or XML), which is specified by **Content-Type** in the request header. It is used to transfer content other than the request header. If the request body contains full-width characters, these characters must be coded in UTF-8.

The request body varies depending on APIs. Certain APIs do not require the request body, such as the APIs requested using the GET and DELETE methods.

The following shows an example request (a request body included) of the API for **creating an IAM user**. You can learn about request parameters and related

description from this example. The bold parameters need to be replaced for a real request.

- **accountid**: account ID of an IAM user
- **username**: name of an IAM user
- **email**: email of an IAM user
- **password**: login password of an IAM user

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date,
Signature=**********

{
    "user": {
        "domain_id": "accountid",
        "name": "username",
        "password": "**********",
        "email": "email",
        "description": "IAM User Description"
    }
}
```

If all data required for the API request is available, you can send the request to call the API through **curl**, **Postman**, or coding. In the response to the API used to obtain a user token, **X-Subject-Token** is the desired user token. This token can then be used to authenticate the calling of other APIs.

3.2 Authentication

Requests for calling an API can be authenticated using either of the following methods:

- AK/SK authentication: Requests are encrypted using AK/SK pairs. AK/SK authentication is recommended because it is more secure than token authentication.
- Token authentication: Requests are authenticated using tokens.

AK/SK Authentication

NOTE

AK/SK authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, token authentication is recommended.

In AK/SK authentication, AK/SK is used to sign requests and the signature is then added to the requests for authentication.

- AK: access key ID, which is a unique identifier used in conjunction with a secret access key to sign requests cryptographically.
- SK: secret access key, which is used in conjunction with an AK to sign requests cryptographically. It identifies a request sender and prevents the request from being modified.

In AK/SK authentication, you can use an AK/SK to sign requests based on the signature algorithm or using the signing SDK. For details about how to sign requests and use the signing SDK, see **API Request Signing Guide**.

> **NOTE**
>
> The signing SDK is only used for signing requests and is different from the SDKs provided by services.

Token Authentication

> **NOTE**
>
> The validity period of a token is 24 hours. When using a token for authentication, cache it to prevent frequently calling the IAM API used to obtain a user token.

A token specifies temporary permissions in a computer system. During API authentication using a token, the token is added to requests to get permissions for calling the API. You can obtain a token by calling the **Obtaining User Token** API.

IMS is a project-level service. When you call the API, set **auth.scope** in the request body to **project**.

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",   // IAM user name
                    "password": $ADMIN_PASS,  //IAM user password. You are advised to store it in ciphertext in the configuration file or an environment variable and decrypt it when needed to ensure security.
                    "domain": {
                        "name": "domainname"  // Name of the account to which the IAM user belongs
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxx"    // Project name
            }
        }
    }
}
```

After a token is obtained, the **X-Auth-Token** header field must be added to requests to specify the token when calling other APIs. For example, if the token is **ABCDEFJ....**, **X-Auth-Token: ABCDEFJ....** can be added to a request as follows:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

3.3 Response

Status Code

After sending a request, you will receive a response, including a status code, response header, and response body.

A status code is a group of digits, ranging from 1xx to 5xx. It indicates the status of a request. For more information, see **Status Codes**.

For example, if status code **201** is returned for calling the API used to **create an IAM user**, the request is successful.

Response Header

Similar to a request, a response also has a header, for example, **Content-Type**.

Figure 3-2 shows the response header fields for the API used to **create an IAM user**. The **X-Subject-Token** header field is the desired user token. This token can then be used to authenticate the calling of other APIs.

NOTE

For security purposes, you are advised to set the token in ciphertext in configuration files or environment variables and decrypt it when using it.

Figure 3-2 Header fields of the response to the request for creating an IAM user


```
"X-Frame-Options": "SAMEORIGIN",
"X-IAM-ETag-id": "2562365939-d8f6f12921974cb097338ac11fceac8a",
"Transfer-Encoding": "chunked",
"Strict-Transport-Security": "max-age=31536000; includeSubdomains;",
"Server": "api-gateway",
"X-Request-Id": "af2953f2bcc67a42325a69a19e6c32a2",
"X-Content-Type-Options": "nosniff",
"Connection": "keep-alive",
"X-Download-Options": "noopen",
"X-XSS-Protection": "1; mode=block;",
"X-IAM-Trace-Id": "token_[redacted]_null_af2953f2bcc67a42325a69a19e6c32a2",
"Date": "Tue, 21 May 2024 09:03:40 GMT",
"Content-Type": "application/json; charset=utf8"
```


(Optional) Response Body

The body of a response is often returned in a structured format (for example, JSON or XML) as specified in the **Content-Type** header field. The response body transfers content except the response header.

The following is part of the response body for the API used to **create an IAM user**.


```
{
    "user": {
        "id": "c131886aec...",
        "name": "IAMUser",
        "description": "IAM User Description",
        "areacode": "",
        "phone": "",
        "email": "***@***.com",
        "status": null,
        "enabled": true,
        "pwd_status": false,
        "access_mode": "default",
        "is_domain_owner": false,
        "xuser_id": "",
        "xuser_type": "",
        "password_expires_at": null,
        "create_time": "2024-05-21T09:03:41.000000",
        "domain_id": "d78cbac1.....",
        "xdomain_id": "30086000.....",
        "xdomain_type": "",
        "default_project_id": null
```



```
    }
}
```

If an error occurs during API calling, an error code and a message will be displayed. The following shows an error response body.

```
{
    "error_msg": "The request message format is invalid.",
    "error_code": "IMG.0001"
}
```

In the response body, **error_code** is an error code, and **error_msg** provides information about the error.

4 Application Examples

4.1 Example 1: Uploading a Media File Less Than 20 MB

Scenario

If you need to process (such as publish and transcode) your local media files in VOD, then you can call VOD APIs and OBS APIs to upload them to OBS buckets owned by VOD.

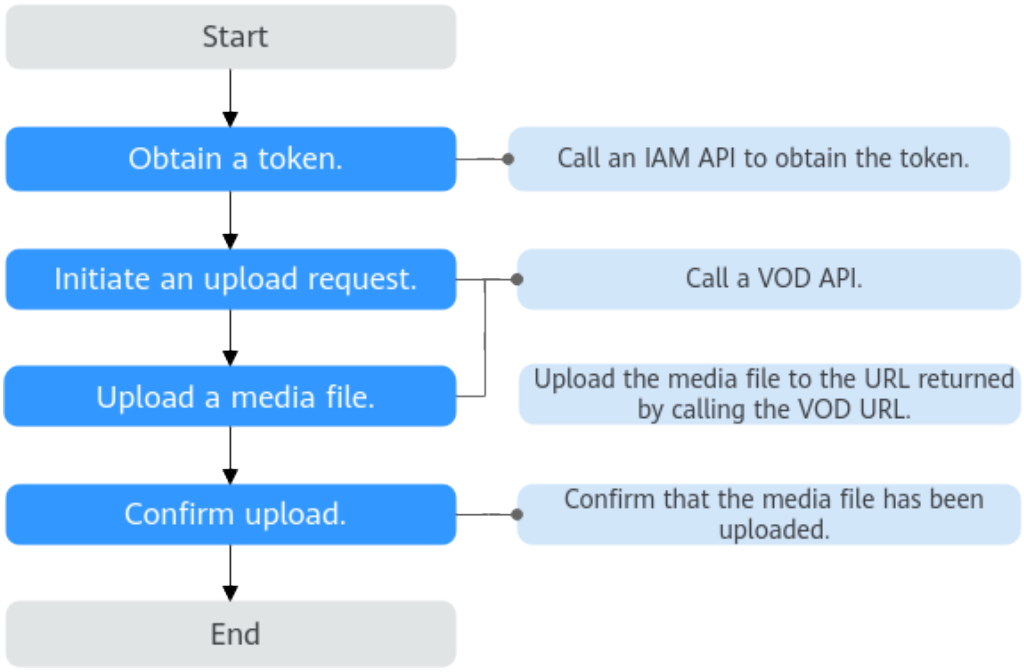
This section describes how to upload a video file less than 20 MB by calling APIs.

Prerequisites

- You have specified the region where the VOD service for uploading media assets is located. See **Before You Start**.
- You have obtained the ID of the project by referring to **Obtaining a Project ID**.

Overall Process

Figure 4-1 Media asset upload process



1. Call the API for **uploading media assets to VOD** to create a media asset.
2. Upload a local media asset using PUT.
3. Call the API for **confirming media asset upload**.

Procedure

Step 1 **Obtain a user token** and use it to authenticate the calling of VOD APIs.

For details, see **Making an API Request**. **CN North-Beijing4** is used as an example. If you need to call a VOD API in another region, replace the endpoint with the **IAM endpoint** of the corresponding region.

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
```



```
          "name": "projectname"
        }
      }
    }
}
```

As shown in **Figure 4-2**, information in the red box indicates the user token.

Figure 4-2 Obtaining the user token

Step 2 Call the API for **uploading media assets to VOD** to create a media asset. Add the **X-Auth-Token** field to the request header. The value of **Value** is obtained in **1**.

```
POST https://vod.cn-north-4.myhuaweicloud.com/v1.0/{project_id}/asset

{
    "title": "test",
    "description": "test",
    "category_id": 87748,
    "video_name": "test.mp4",
    "video_type": "MP4",
    "cover_type": "PNG",
    "tags":"mytags,test",
    "auto_publish": 0,
    "template_group_name":"test",
    "subtitles":[
        {
            "id":1,
            "language":"CN",
            "type": "SRT",
            "md5": "SqcyFjJZoDZaP8oKIY6rgQ==",
            "description":"AAAAA" ,
            "name":"cc.srt"
        }
    ],
    "thumbnail":{
        "type":"time",
        "cover_position":1

    },
    "review":{
        "interval":10,
        "politics":-1,
        "terrorism":-1,
        "porn":-1
    }
}
```


- **vod.cn-north-4.myhuaweicloud.com** is the endpoint of VOD. You can obtain the endpoint from **Before You Start**.
- **category_id** indicates the media asset category ID. You need to create a media category in advance by referring to **Creating a Media Asset Category**.
- **template_group_name** indicates the transcoding template group name. You need to customize a transcoding template in advance by referring to **Customizing a Transcoding Template**.

  NOTE

  The region where an IAM API is called must be the same as the region where a VOD API is called. For example, the user token obtained when you call an IAM API in the **CN North-Beijing4** region can only be used to authenticate the requests for calling VOD APIs in the **CN North-Beijing4** region.

Step 3 Obtain the values of **video_upload_url**, **cover_upload_url**, and **subtitle_upload_urls** from the response parameters.


```
{
  "asset_id": "3f40a2c7c60454f5f84381e0313ca230",
  "video_upload_url": "https://vod-bucket-81.obs.cn-
north-4.myhuaweicloud.com:443/474bcff2992f4be5b883a2fb9cec7343/3f40a2c7c60454f5f84381e0313ca230
/cdeda86cd1b7b3dd760a3ff28a5ee497.mp4?
AWSAccessKeyId=BG923RWHL4HFXOGKCVAL&Expires=1560420274&Signature=9f
%2BZcdD6SwjIU5ARHYiP6YY1Lyw%3D",
  "cover_upload_url": "https://vod-bucket-81.obs.cn-
north-4.myhuaweicloud.com:443/474bcff2992f4be5b883a2fb9cec7343/3f40a2c7c60454f5f84381e0313ca230
/cover/Cover0.png?AWSAccessKeyId=BG923RWHL4HFXOGKCVAL&Expires=1560420274&Signature=4Aa88NK
%2By%2By1Xo0RvLpOvuaFCoE%3D",
  "subtitle_upload_urls": [
    "https://vod-bucket-81.obs.cn-
north-4.myhuaweicloud.com:443/474bcff2992f4be5b883a2fb9cec7343/3f40a2c7c60454f5f84381e0313ca230
/subtitle/1.srt?
AWSAccessKeyId=BG923RWHL4HFXOGKCVAL&Expires=1560420274&Signature=l0UclE9yfaVrxkl0kaNnr
%2BemG98%3D"
  ],
  "target": {
    "bucket": "vod-bucket-81",
    "location": "cn-north-4",
    "object": "474bcff2992f4be5b883a2fb9cec7343/3f40a2c7c60454f5f84381e0313ca230/
cdeda86cd1b7b3dd760a3ff28a5ee497.mp4"
  }
}
```


Step 4 Send PUT requests to upload media assets, thumbnails, and subtitle files to **video_upload_url**, **cover_upload_url**, and **subtitle_upload_urls**.

Set **Content-Type** based on the type of the file to be uploaded.

- Video file: video/*video format*, for example, **video/mp4**. For details about how to fill in the request header of a video file format, see **Table 4-1**.
- Audio file: audio/*audio format*, for example, **audio/mp3**. For details about how to fill in the request header of an audio file format, see **Table 4-2**.
- Image file: image/*image format*, for example, **image/png**.
- Subtitle file: **application/octet-stream**

Table 4-1 Parameters in the request header of a video file

File Name Extension	Content-Type
MP4	video/mp4

File Name Extension	Content-Type
MOV	video/quicktime
MXF	application/mxf
TS	video/mp2t
MPG	video/mpeg
FLV	video/flv
WMV	video/x-ms-wmv
AVI	video/x-msvideo
M4V	video/m4v
F4V	application/f4v
MPEG	video/mpeg
M3U8	application/octet-stream
_3GP/3GP	video/3gpp
ASF	video/x-ms-asf
MKV	video/x-matroska
WEBM	video/webm
MPD	video/dash

Table 4-2 Parameters in the request header of an audio file

File Name Extension	Content-Type
MP3	audio/mp3
WMA	audio/wma
APE	audio/ape
FLAC	audio/flac
AAC	audio/aac
AC3	audio/ac3
MMF	audio/mmf
AMR	audio/amr
M4A	audio/m4a
M4R	audio/m4r

File Name Extension	Content-Type
OGG	audio/ogg
WAV	audio/wav
WV	audio/wv
MP2	audio/mp2

Step 5 Call the API for **confirming media asset upload**.

```
POST https://vod.cn-north-4.myhuaweicloud.com/v1.0/{project_id}/asset/status/uploaded

{
    "asset_id": "0f4d3f1f32ec353d8866f2d84a036124",
    "status":"CREATED"
}
```

----End

After a media asset was uploaded, you can view details about the media asset on the VOD console.

4.2 Example 2: Uploading a Media File Greater Than 20 MB by Part

Scenario

Example 1: Uploading a Media File Less Than 20 MB has described how to upload a media file less than 20 MB to VOD. If you need to upload a media file greater than 20 MB, you can call VOD APIs to upload the file by part.

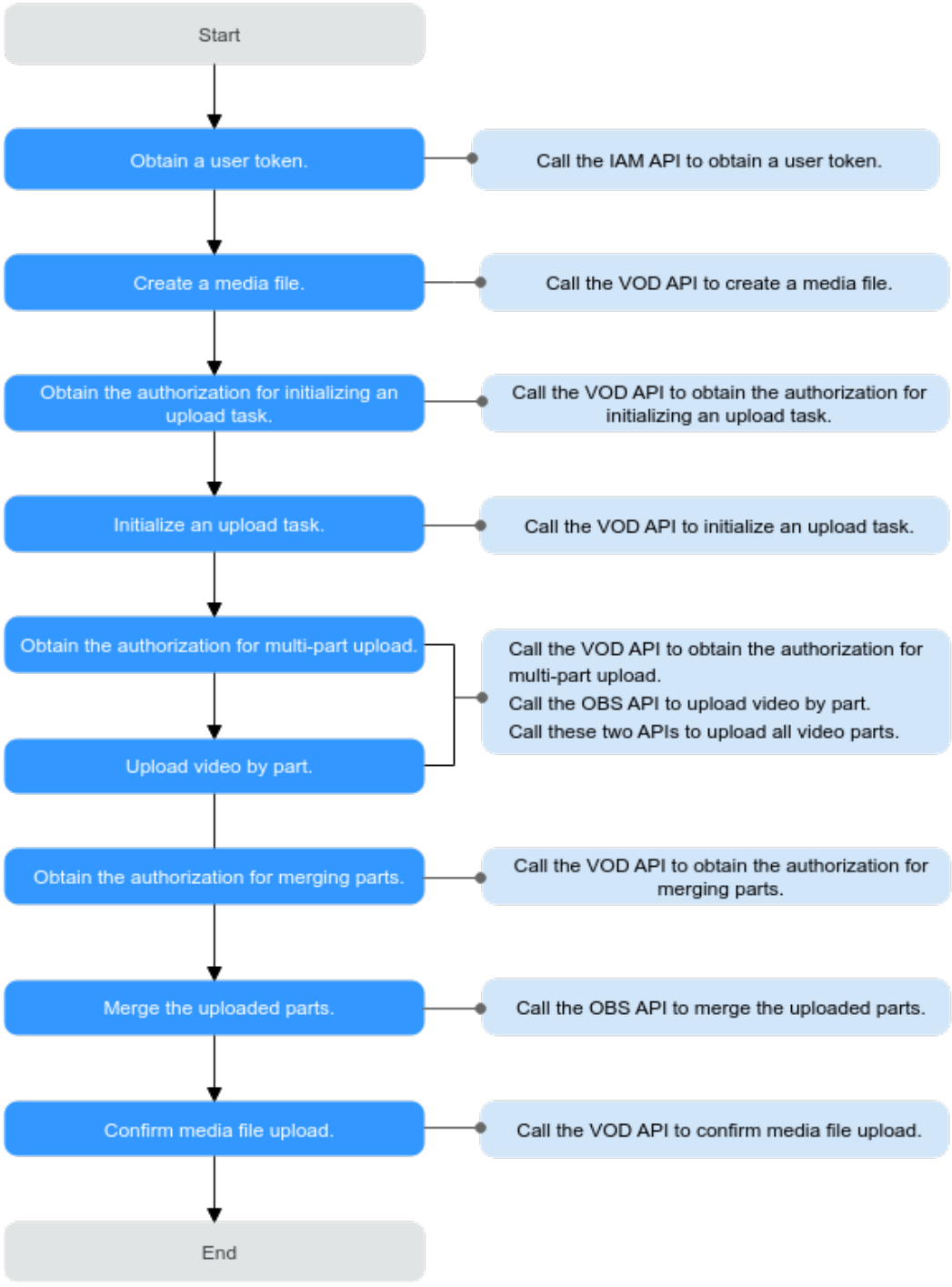
This section describes how to upload a video file greater than 20 MB by calling APIs.

Prerequisites

- You have specified the region where the VOD service for uploading media assets is located. See **Before You Start**.
- You have obtained the ID of the project by referring to **Obtaining a Project ID**.
- The video file to be uploaded has been split into binary streams. The recommended size of each part cannot exceed 20 MB.
- The OBS API for **listing uploaded parts** allows for 1 to 10,000 parts and 100 KB to 5 GB for each part. By default, a response to a maximum of 1,000 parts can be returned. If there are more than 1,000 parts, you need to call the API by referring to **listing uploaded parts**.

Overall Process

Figure 4-3 Media upload process



1. **Obtain a user token**
2. **Create a VOD media asset**
3. **Obtain authorization for initializing an upload task**
4. **Initialize an upload task**

5. **Obtain authorization for multipart upload**
6. **Upload a video by part**
7. Refer to **5** to **6** to upload all videos by part.
8. **Obtain authorization for listing uploaded parts**
9. **Obtain uploaded parts**
10. **Obtain authorization for merging parts**
11. **Merge uploaded parts**
12. **Confirm media asset upload**

Procedure

Step 1 **Obtain a user token** and use it to authenticate the calling of VOD APIs.

For details, see **Making an API Request**. **CN North-Beijing4** is used as an example. If you need to call a VOD API in another region, replace the endpoint with the **IAM endpoint** of the corresponding region.

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "projectname"
      }
    }
  }
}
```

As shown in **Figure 4-4**, information in the red box indicates the user token.

Figure 4-4 Obtaining the user token

```
cache-control → no-cache, no-store, must-revalidate
connection → keep-alive
content-length → 15185
content-type → application/json; charset=UTF-8
date → Wed, 29 May 2019 08:02:38 GMT
expires → Thu, 01 Jan 1970 00:00:00 GMT
pragma → no-cache
server → api-gateway
strict-transport-security → max-age=31536000; includeSubdomains;
via → proxy A
x-content-type-options → nosniff
x-download-options → noopen
x-frame-options → SAMEORIGIN
x-iam-trace-id → token_global_null_2fed19b42067d0b94ae73dbe0b8b7841
x-request-id → fd8e4d01ae9caf9f58af20cbdb2c88e2
x-subject-token
→ MIIYdwYJKoZIhvcNAQcCoIIYaDCCGGQCAQExDTALBglghkgBZQMEAgEwghbFBgkqhkiG9w0BBwGggha2BIIWsnsidG9rZW4iOnsiZXhwaXJlc19hdCI6IjIwMTktMDUtMzBUMDg6MDI6MzguNTgxMDAwWiIsIm1ldGhvZHMiOlsicGFzc3dvcmQiXSwiY2F0YWxvZy
wN-yN0wpSwyRGtY3upfYOauVL-E275QpmEOPG4qKzPx4Yeo2Hr8ZdFptB6AZt-
RGdroWLe95InegpR0PGK9zgbuRiWYATdQL3xGxVvC1Um8XJADnwO2TWxgbmzrPiwFBvhP3F20TK4v9XhxWHUxH2yfo+KScgsXHAzv6VURCIWMdV+UeESwZo7vB22C77p2B9d8Isyb3oQoZmTyrIaupZf6U4PX2J+rW4aplwEszdvrRQLLYOzxinGG3Af
x-xss-protection → 1; mode=block;
```

Step 2 Call the API for **uploading media assets to VOD** to create a media asset.

1. Select the POST request method and enter the URI.
   https://**vod.cn-north-4.myhuaweicloud.com**/v1.0/{project_id}/asset
2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. Specify the following parameters in the request body:

```
{
   "title": "testVideo",
   "description": "test",
   "category_id": 87748,
   "video_name": "test.mp4",
   "video_type": "MP4",
   "auto_publish": 0
}
```

NOTE

Only some media asset parameters are configured in the example. You can configure all parameters by referring to the API for **uploading media assets to VOD**.

4. If the request is successful, the media asset ID, video upload URL, and information about the OBS bucket that stores the video file are returned.


```
{
   "asset_id": "fea62a2845e43c37b4cbfb017c0d0821",
   "video_upload_url": "https://vod-bucket-65-cn-north-4.obs.cn-
north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/
fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?
AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596700132&Signature=GiHTcO2avO9B84sH8A
5wL2GieCI%3D",
   "cover_upload_url": null,
   "subtitle_upload_urls": [],
   "target": {
      "bucket": "vod-bucket-65-cn-north-4",
      "location": "cn-north-4",
      "object": "05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/
d1f221f09f6bffefb882c8f9e167483a.mp4"
   }
}
```


Step 3 Call the API for **obtaining authorization for multipart upload** to obtain authorization for initializing an upload task from OBS.

1. Select the GET request method and enter the request URI.


```
https://vod.cn-north-4.myhuaweicloud.com/v1.1/{project_id}/asset/authority?
http_verb=POST&content_type=video/mp4&bucket=vod-bucket-65-cn-
north-4&object_key=05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/
d1f221f09f6bffefb882c8f9e167483a.mp4
```

- **http_verb**: HTTP method for calling the OBS API for multipart upload. For details about the suitable HTTP method, see **OBS API Reference**.
- **content_type**: **HTTP Content-type** of a certain file type. You need to configure this parameter based on the type of the uploaded file.
  - Video file: video/*video format*, for example, **video/mp4**. For details about how to fill in the request header of a video file format, see **Table 4-3**.
  - Audio file: audio/*audio format*, for example, **audio/mp3**. For details about how to fill in the request header of an audio file format, see **Table 4-4**.
  - Image file: image/*image format*, for example, **image/png**.
  - Subtitle file: **application/octet-stream**
- **bucket**: OBS bucket for storing video files. It corresponds to the **bucket** field returned in **2.d**.
- **object_key**: name of the object in the OBS bucket. It corresponds to the **object** field returned in **2.d**.

Table 4-3 Parameters in the request header of a video file

File Name Extension	Content-Type
MP4	video/mp4
MOV	video/quicktime
MXF	application/mxf
TS	video/mp2t
MPG	video/mpeg
FLV	video/flv
WMV	video/x-ms-wmv
AVI	video/x-msvideo
M4V	video/m4v
F4V	application/f4v
MPEG	video/mpeg
M3U8	application/octet-stream
_3GP/3GP	video/3gpp
ASF	video/x-ms-asf
MKV	video/x-matroska

File Name Extension	Content-Type
WEBM	video/webm
MPD	video/dash

Table 4-4 Parameters in the request header of an audio file

File Name Extension	Content-Type
MP3	audio/mp3
WMA	audio/wma
APE	audio/ape
FLAC	audio/flac
AAC	audio/aac
AC3	audio/ac3
MMF	audio/mmf
AMR	audio/amr
M4A	audio/m4a
M4R	audio/m4r
OGG	audio/ogg
WAV	audio/wav
WV	audio/wv
MP2	audio/mp2

2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. If the request is successful, information about the authorization is returned.

```
{
    "sign_str": "https://vod-bucket-65-cn-north-4.obs.cn-
north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/
fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?
uploads&AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596706429&Signature=5D15iJUcTko
WLpE2vn54FQZskXA%3D"
}
```

Step 4 Call the OBS API for **initializing a multipart upload task**.

NOTICE

OBS does not support verification on API Explorer. You need to use a third-party tool, such as Postman, to construct a request for verification.

1. Select the POST request method and enter the authorized URI returned in **3.c**.
   ```
   https://vod-bucket-65-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?uploads&AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596706429&Signature=5D15iJUcTkoWLpE2vn54FQZskXA%3D
   ```
2. Add **Content-Type** to the request header and configure this parameter based on the format of the uploaded video. For details, see **3**.
3. If the request is successful, the initialization information is returned.
   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <InitiateMultipartUploadResult xmlns="http://obs.cn-north-4.myhuaweicloud.com/doc/2015-06-30/">
     <Bucket>vod-bucket-65-cn-north-4</Bucket>
     <Key>05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4</Key>
     <UploadId>00000173C2ED862344C835374DFE33C8</UploadId>
   </InitiateMultipartUploadResult>
   ```

Step 5 Call the API for **obtaining authorization for multipart upload**.

1. Select the GET request method and enter the request URI.
   ```
   https://vod.cn-north-4.myhuaweicloud.com/v1.1/{project_id}/asset/authority?http_verb=PUT&content_type=video/mp4&bucket=vod-bucket-65-cn-north-4&object_key=05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4&content_md5=hHZXtgjYL2fmLpu%2byQ%2bXgg%3d%3d&upload_id=00000173C2ED862344C835374DFE33C8&part_number=1
   ```
   - **http_verb**: The HTTP method for multipart upload is PUT.
   - **content_type**: **HTTP Content-type** of a certain file type. You need to configure this parameter based on the type of the uploaded file. For details, see **3**.
   - **bucket**: OBS bucket for storing video files. It corresponds to the **bucket** field returned in **2.d**.
   - **object_key**: name of the object in the OBS bucket. It corresponds to the **object** field returned in **2.d**.
   - **content_md5**: MD5 value of each video part. This parameter is mandatory. For details about how to generate an MD5 value, see **Generating an MD5 Value**. Note: If there are special characters in the request, escape them.
   - **upload_id**: ID of the upload task, which is the **UploadId** field returned in **4.c**.
   - **part_number**: number of an uploaded part
2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. If the request is successful, information about the authorization for multipart upload is returned.
   ```
   {
     "sign_str": "https://vod-bucket-65-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596708691&partNumber=1&uploadId=00000173C2ED862344C835374DFE33C8&Signature=cjw3CmUFeNBFAuLWBTsPRp9NfsQ%3D"
   }
   ```

Step 6 Call the OBS API for **multipart upload** to upload the first part of the video file.

NOTICE

OBS does not support verification on API Explorer. You need to use a third-party tool, such as Postman, to construct a request for verification.

1. Select the PUT request method and enter the authorized URI returned in **5.c**.
   ```
   https://vod-bucket-65-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596708691&partNumber=1&uploadId=00000173C2ED862344C835374DFE33C8&Signature=cjw3CmUFeNBFAuLWBTsPRp9NfsQ%3D
   ```
2. Add **Content-Type** to the request header and set the parameter to **application/octet-stream**. If **content_md5** has been configured in **5.a**, **Content-MD5** must be added to the request header and set to the MD5 value of the corresponding part (escaping is not required). Otherwise, the upload will fail.
3. In the request body, the first part of the video file is uploaded in binary streams.
4. If the request is successful, the status code **200 OK** is returned.

Step 7 Repeat **5** to **6** to upload all parts of the video file.

Step 8 Call the API for **obtaining authorization for multipart upload** to obtain authorization for listing uploaded parts.

1. Select the GET request method and enter the request URI.
   ```
   https://vod.cn-north-4.myhuaweicloud.com/v1.1/{project_id}/asset/authority?http_verb=GET&bucket=vod-bucket-65-cn-north-4&object_key=05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4&upload_id=000001717E28524B44CEFF2C2CE1B06A
   ```
   - **http_verb**: The HTTP method for listing uploaded parts is GET.
   - **bucket**: OBS bucket for storing video files. It corresponds to the **bucket** field returned in **2.d**.
   - **object_key**: name of the object in the OBS bucket. It corresponds to the **object** field returned in **2.d**.
   - **upload_id**: ID of the upload task, which is the **UploadId** field returned in **4.c**.
2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. If the request is successful, information about the authorization for listing the uploaded parts is returned.
   ```
   {
      "sign_str": "https://vod-bucket-65-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596710054&uploadId=000001717E28524B44CEFF2C2CE1B06A&Signature=G5EOUr488cyPIretp8lgQZmpPw0%3D"
   }
   ```

Step 9 Call the OBS API for **listing uploaded parts** to obtain information about all uploaded parts of the video file.

NOTICE

- The OBS API for **listing uploaded parts** allows for 1 to 10,000 parts and 100 KB to 5 GB for each part. By default, a response to a maximum of 1,000 parts can be returned. If there are more than 1,000 parts, you need to call the API by referring to **listing uploaded parts**.
- OBS does not support verification on API Explorer. You need to use a third-party tool, such as Postman, to construct a request for verification.

Note: The **Content-Type** attribute in the request header is set to **null** or is not carried.

1. Select the GET request method and enter the authorized URI returned in **8.c**.

```
https://vod-bucket-65-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596710054&uploadId=000001717E28524B44CEFF2C2CE1B06A&Signature=G5EOUr488cyPIretp8lgQZmpPw0%3D
```

2. If the request is successful, information about the uploaded parts is returned.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListPartsResult xmlns="http://obs.cn-north-4.myhuaweicloud.com/doc/2015-06-30/">
  <Bucket>vod-bucket-65-cn-north-4</Bucket>
  <Key>05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4</Key>
  <UploadId>00000173C2ED862344C835374DFE33C8</UploadId>
  <Initiator>
    <ID>d9235b9f3cf549499924f6de095241af:a96ecebcb3994e34b074e48f3dfc8237</ID>
    <DisplayName>xxx</DisplayName>
  </Initiator>
  <Owner>
    <ID>d9235b9f3cf549499924f6de095241af</ID>
    <DisplayName>xxx</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
  <PartNumberMarker>0</PartNumberMarker>
  <NextPartNumberMarker>4</NextPartNumberMarker>
  <MaxParts>1000</MaxParts>
  <IsTruncated>false</IsTruncated>
  <Part>
    <PartNumber>1</PartNumber>
    <LastModified>2020-08-06T09:05:10.192Z</LastModified>
    <ETag>"847657b608d82f67e62e9bbec90f9782"</ETag>
    <Size>10000000</Size>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2020-08-06T08:02:02.821Z</LastModified>
    <ETag>"9a6a36ed9086a3a2fea130220e1e809c"</ETag>
    <Size>10000000</Size>
  </Part>
  <Part>
    <PartNumber>3</PartNumber>
    <LastModified>2020-08-06T08:04:19.711Z</LastModified>
    <ETag>"3a3afe552832bee0faf081c1e720067e"</ETag>
    <Size>10000000</Size>
  </Part>
  <Part>
    <PartNumber>4</PartNumber>
    <LastModified>2020-08-06T07:23:17.160Z</LastModified>
    <ETag>"6335af859c20ccef26b27ea691e7ecf7"</ETag>
    <Size>7472445</Size>
  </Part>
</ListPartsResult>
```


Step 10 Call the API for **obtaining authorization for multipart upload** to obtain authorization for merging uploaded parts.

1. Select the GET request method and enter the request URI.
   ```
   https://vod.cn-north-4.myhuaweicloud.com/v1.1/{project_id}/asset/authority?
   http_verb=POST&bucket=vod-bucket-65-cn-north-4&object_key=05041fffa4002****f6dc009cc6f8f33/
   fea62a2845e43c37b4cbfb017c0d0821/
   d1f221f09f6bffefb882c8f9e167483a.mp4&upload_id=00000173C2ED862344C835374DFE33C8
   ```
   - **http_verb**: The HTTP method for merging parts is POST.
   - **bucket**: OBS bucket for storing video files. It corresponds to the **bucket** field returned in **2.d**.
   - **object_key**: name of the object in the OBS bucket. It corresponds to the **object** field returned in **2.d**.
   - **upload_id**: ID of the upload task, which is the **UploadId** field returned in **4.c**.
2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. If the request is successful, information about the authorization for merging uploaded parts is returned.
   ```
   {
      "sign_str": "https://vod-bucket-65-cn-north-4.obs.cn-
   north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/
   fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?
   AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596709340&uploadId=000001717E28524B44CE
   FF2C2CE1B06A&Signature=Pa6IaMbH1Ofa0Vi%2BCbkdgmWzm70%3D"
   }
   ```

Step 11 Call the OBS API for **merging parts** to merge the uploaded parts into a video file.

NOTICE

OBS does not support verification on API Explorer. You need to use a third-party tool, such as Postman, to construct a request for verification.

1. Select the POST request method and enter the authorized URI returned in **10.c**.
   ```
   https://vod-bucket-65-cn-north-4.obs.cn-
   north-4.myhuaweicloud.com:443/05041fffa4002****f6dc009cc6f8f33/
   fea62a2845e43c37b4cbfb017c0d0821/d1f221f09f6bffefb882c8f9e167483a.mp4?
   AWSAccessKeyId=MZH0LUL329N1SSXNB3S4&Expires=1596708691&partNumber=1&uploadId=000001
   73C2ED862344C835374DFE33C8&Signature=cjw3CmUFeNBFAuLWBTsPRp9NfsQ%3D
   ```
2. Add **Content-Type** to the request header and set the parameter to **application/xml**.
3. Specify the following parameters in the request body:
   ```
   <CompleteMultipartUpload>
      <Part>
         <PartNumber>1</PartNumber>
         <ETag>"847657b608d82f67e62e9bbec90f9782"</ETag>
      </Part>
      <Part>
         <PartNumber>2</PartNumber>
         <ETag>"9a6a36ed9086a3a2fea130220e1e809c"</ETag>
      </Part>
      <Part>
         <PartNumber>3</PartNumber>
         <ETag>"3a3afe552832bee0faf081c1e720067e"</ETag>
      </Part>
   ```



```
  <Part>
     <PartNumber>4</PartNumber>
     <ETag>"6335af859c20ccef26b27ea691e7ecf7"</ETag>
  </Part>
</CompleteMultipartUpload>
```

- **PartNumber**: number of an uploaded part
- **ETag**: ETag value returned after a video part has been uploaded. The returned information in **9.b** contains the ETag value of each part.

4. If the request is successful, information about the merged parts is returned.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CompleteMultipartUploadResult xmlns="http://obs.cn-north-4.myhuaweicloud.com/doc/
2015-06-30/">
  <Location>05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/
d1f221f09f6bffefb882c8f9e167483a.mp4</Location>
  <Bucket>vod-bucket-65-cn-north-4</Bucket>
  <Key>05041fffa4002****f6dc009cc6f8f33/fea62a2845e43c37b4cbfb017c0d0821/
d1f221f09f6bffefb882c8f9e167483a.mp4</Key>
  <ETag>"87f450dd18b666dfcdf902ac6b162b5a-4"</ETag>
</CompleteMultipartUploadResult>
```

Step 12 Call the API for **confirming media asset upload**.

1. Select the POST request method and enter the URI.

```
https://vod.cn-north-4.myhuaweicloud.com/v1.0/{project_id}/asset/status/uploaded
```

2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. Specify the following parameters in the request body:

```
{
  "asset_id": "fea62a2845e43c37b4cbfb017c0d0821",
  "status":"CREATED"
}
```

4. If the request is successful, the media asset ID is returned.

```
{
  "asset_id": "fea62a2845e43c37b4cbfb017c0d0821"
}
```

----End

Sample Code of Multipart Upload Using Java

See:

- **Java**
- **POM Files**

Sample Code of Multipart Upload Using JavaScript

See **JavaScript**.

Sample Code of Multipart Upload Using Python

See **Python**.

Sample Code of Multipart Upload Using Go

See **Go**.

Sample Code of Multipart Upload Using PHP

See **PHP**.

4.3 Example 3: Obtaining Media Asset Details

Scenario

You can call VOD APIs to query the basic information and information about transcoded files, snapshots, and review of one or more media assets.

This section describes how to call APIs to obtain details about media assets stored in VOD.

Prerequisites

- You have specified the region where the media asset to be queried is located. See **Before You Start**.
- You have obtained the project ID of the region where the media asset to be queried is located. For details, see **Obtaining a Project ID**.

Overall Process

1. **Obtain a user token**
2. **Query media asset details**

Procedure

Step 1 **Obtain a user token** and use it to authenticate the calling of VOD APIs.

For details, see **Making an API Request**. **CN North-Beijing4** is used as an example. If you need to call a VOD API in another region, replace the endpoint with the **IAM endpoint** of the corresponding region.

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "projectname"
      }
    }
```



```
 }
}
```

As shown in **Figure 4-5**, information in the red box indicates the user token.

Figure 4-5 Obtaining the user token

Step 2 Call the API for **querying media assets**.

1. Select the GET request method and enter the request URI.
   https://{***endpoint***}/v1.0/{***project_id***}/asset/details?asset_id=41cff636d5b12a51e7eb2838bbf45201&categories=base_info,transcode_info&thumbnail_info,review_info

   categories indicates the type of media asset information to be queried. You can select one or more types for query. The options are as follows:

   - **base_info**: basic information about the media file, including the media file type, title, category, description, and original file URL
   - **transcode_info**: information about the transcoded media file, including the used transcoding template
   - **thumbnail_info**: information about snapshots, including the snapshot status and snapshot storage URL
   - **review_info**: information about review, including whether the review is passed and the review results of text, cover, and video

   NOTE

   You can query information about transcoded files, snapshots, or review only when the transcoding, snapshot capturing, or review is successful.

2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. If the request is successful, the media asset details are returned.

```
{
 "asset_info_array" : [ {
  "asset_id" : "652c1e4085afeb22fdc256c6757d751b",
  "status" : "PUBLISHED",
  "description" : "Asset meta is published",
  "base_info" : {
   "title" : "okFLV.flv",
   "video_name" : "okFLV.flv",
   "description" : null,
   "category_id" : -1,
```



```
    "category_name": "Other",
    "create_time" : "20190704144303",
    "last_modified" : "20190704144303",
    "video_type" : "FLV",
    "tags" : null,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 244,
      "video_size" : 13682041,
      "width" : 512,
      "hight" : 288,
      "bit_rate" : 448,
      "frame_rate" : 30,
      "quality" : null
    },
    "video_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
45c2493abe68de3dac7e98b0dadcf8ce.flv",
    "sign_url" : "",
    "cover_info_array" : [ {
      "cover_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
cover/Cover0.jpg"
    } ],
    "subtitle_info" : [ ],
    "source_path" : {
      "bucket" : "obs-host",
      "location" : "cn-north-4",
      "object" : "1/okFLV.flv"
    },
    "output_path" : {
      "bucket" : "obs-host",
      "location" : "cn-north-4",
      "object" : "output/652c1e4085afeb22fdc256c6757d751b/"
    }
  },
  "play_info_array" : [ {
    "play_type" : "hls",
    "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
play_video/index.m3u8",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 0,
      "video_size" : 0,
      "width" : 0,
      "hight" : 0,
      "bit_rate" : 0,
      "frame_rate" : 0,
      "quality" : null
    }
  }, {
    "play_type" : "hls",
    "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
play_video/b5d498082bbcff7a2055041d803ae2f9_1.m3u8",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 205,
      "video_size" : 14669824,
      "width" : 512,
      "hight" : 288,
      "bit_rate" : 534,
      "frame_rate" : 0,
      "quality" : null
    }
  } ]
 } ],
```



```
  "is_truncated" : 0,
  "total" : 1
}
{
  "asset_info_array" : [ {
    "asset_id" : "652c1e4085afeb22fdc256c6757d751b",
    "status" : "PUBLISHED",
    "description" : "Asset meta is published",
    "base_info" : {
      "title" : "okFLV.flv",
      "video_name" : "okFLV.flv",
      "description" : null,
      "category_id" : -1,
      "category_name": "Other",
      "create_time" : "20190704144303",
      "last_modified" : "20190704144303",
      "video_type" : "FLV",
      "tags" : null,
      "meta_data" : {
        "pack_type" : null,
        "codec" : "H.264",
        "duration" : 244,
        "video_size" : 13682041,
        "width" : 512,
        "hight" : 288,
        "bit_rate" : 448,
        "frame_rate" : 30,
        "quality" : null
      },
      "video_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
45c2493abe68de3dac7e98b0dadcf8ce.flv",
      "sign_url" : "",
      "cover_info_array" : [ {
        "cover_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
cover/Cover0.jpg"
      } ],
      "subtitle_info" : [ ],
      "source_path" : {
        "bucket" : "obs-host",
        "location" : "cn-north-4",
        "object" : "1/okFLV.flv"
      },
      "output_path" : {
        "bucket" : "obs-host",
        "location" : "cn-north-4",
        "object" : "output/652c1e4085afeb22fdc256c6757d751b/"
      }
    },
    "play_info_array" : [ {
      "play_type" : "hls",
      "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
play_video/index.m3u8",
      "encrypted" : 0,
      "meta_data" : {
        "pack_type" : null,
        "codec" : "H.264",
        "duration" : 0,
        "video_size" : 0,
        "width" : 0,
        "hight" : 0,
        "bit_rate" : 0,
        "frame_rate" : 0,
        "quality" : null
      }
    }, {
      "play_type" : "hls",
      "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
play_video/b5d498082bbcff7a2055041d803ae2f9_1.m3u8",
      "encrypted" : 0,
```



```
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 205,
      "video_size" : 14669824,
      "width" : 512,
      "hight" : 288,
      "bit_rate" : 534,
      "frame_rate" : 0,
      "quality" : null
    }
  } ]
 } ],
 "is_truncated" : 0,
 "total" : 1
}
```


----End

4.4 Example 4: Processing a Video File

Scenario

During media asset upload, you can configure parameters for media asset processing. After the upload is complete, the media asset is automatically processed. You can also call the VOD APIs to process uploaded video files.

This section describes how to call APIs to process uploaded video files, including video transcoding, snapshot capturing, and encryption. You can perform one or multiple processing operations at the same time.

Prerequisites

- You have specified the region where the video file to be processed is located. See **Before You Start**.
- You have obtained the project ID of the region where the video file to be processed is located by referring to **Obtaining a Project ID**.

Overall Process

1. **Obtain a user token**
2. **(Optional) Create a transcoding template**
3. **(Optional) Configure the URL of Key Management Service (KMS)**
4. **Create a video processing task**
5. **Query the video processing status**

Procedure

Step 1 **Obtain a user token** and use it to authenticate the calling of VOD APIs.

For details, see **Making an API Request**. **CN North-Beijing4** is used as an example. If you need to call a VOD API in another region, replace the endpoint with the **IAM endpoint** of the corresponding region.

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json
```



```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "password",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "projectname"
      }
    }
  }
}
```

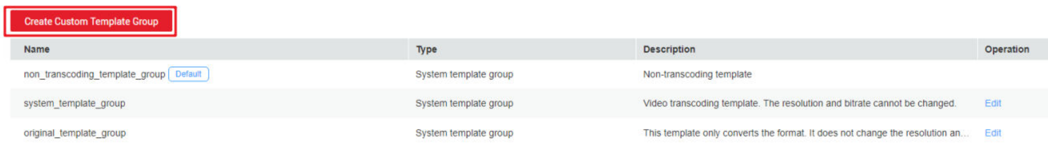
As shown in **Figure 4-6**, information in the red box indicates the user token.

Figure 4-6 Obtaining the user token

Step 2 Create a custom transcoding template on the VOD console. For details, see **Transcoding Settings**.

You can also use the system transcoding template provided by VOD. After creating a transcoding template, you can obtain the transcoding template name on the VOD console.

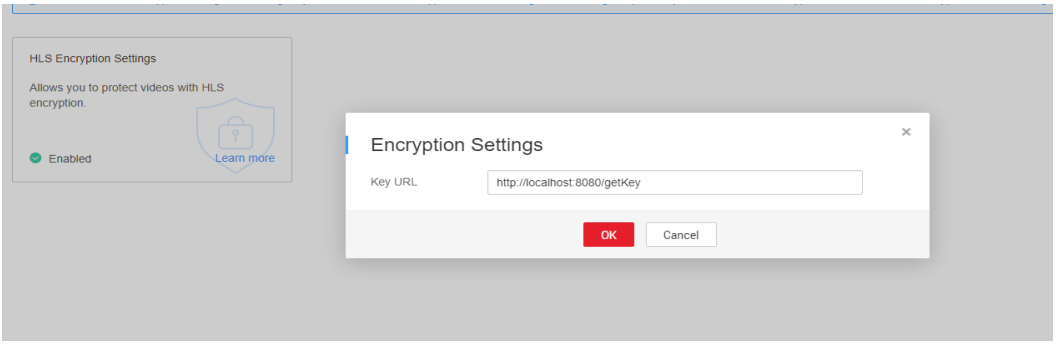
Figure 4-7 Obtaining the transcoding template name



Step 3 On the VOD console, configure the URL of KMS. For details, see **HLS Encryption Settings**.

Only transcoded video files in HLS can be encrypted. If encryption is not required, skip this step.

Figure 4-8 Encryption settings



Step 4 Create a video processing task.

1. Select the POST request method and enter the URI.
   https://{***endpoint***}/v1.0/{***project_id***}/asset/process
2. In the request header, add **X-Auth-Token** and set its value to the token obtained in **1**.
3. Specify the following parameters in the request body:

```
{
   "asset_id": "b4f39691d66cc0ef75d62ee567146e11",
   "template_group_name": "test",
   "auto_encrypt":0,
   "thumbnail":{
      "type":"time",
      "time":12

   },
   "subtitle_id":[1]
}
```

   NOTE

   You can change subtitle files only when you configured the **subtitles** parameter in **Example 1: Uploading a Media File Less Than 20 MB** for the **subtitle_id** parameter to take effect.

4. If the request is successful, the processed media asset ID is returned.

```
{
  "asset_id": "b4f39691d66cc0ef75d62ee567146e11"
}
```

Step 5 Call the API for querying details to query the video processing execution status. For details, see **Example 3: Obtaining Media Asset Details**.

The execution time of a video processing task depends on the size of the video file and the number of processing operations. You are advised to query the execution status 5 to 10 minutes after the video task is submitted.

----End

5 Key query

5.1 Queries keys

Function

When a device plays an encrypted HLS video, the device requests a key from the tenant management system, which checks whether there are keys cached locally. If not, this API is called to query the key from VOD. For details about the application scenarios of this API, see the sample code for preventing video leakage through HLS encryption in *VOD Best Practices*.

URI

GET /v1.0/{project_id}/asset/ciphers

Table 5-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 5-2 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Request Parameters

Table 5-3 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 5-4 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
edk	String	Key ciphertext
dk	String	Key plaintext

Status code: 400

Table 5-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Query keys.

```
GET https://{endpoint}/v1.0/{project_id}/asset/ciphers?asset_id={asset_id}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "06da0367af8f297ea3efc791e8b27157",
  "edk" :
"02009800f00446c3aa6ad610bb121a602fa3091daca7a544ded4f514cae2c233cf51f3d88de64712242a8468042db19b4282e0d7196a8d4df6150169debb5d077527fad983e14174f1ab5430958ca4903187321f0a3406284bf938b64de95ddd3db438df9ddae98d347058969fd39be5049a0a5830396432303366352d313235362d346265322d626539642d383131643333633336630353200000000f9776c60d33706c68195bc64934f94ee28fbfdf03e5fbb65c5450599d6aa4807",
  "dk" : "4Q9KYgKbUChxocNYbteP3A=="
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10101",
  "error_msg" : "Unable to get the key because there is no encryption."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

6 Watermark template management

6.1 Creates a watermark template

Function

Creates a watermark template

URI

POST /v1.0/{project_id}/template/watermark

Table 6-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 6-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 6-3 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Name of a watermark template
watermark_type	No	String	Watermark type. Currently, only IMAGE (image watermark) is supported.
image_process	No	String	This parameter is valid when **watermark_type** is set to **IMAGE**. Possible options: • ORIGINAL: Only simple scaling is performed. • TRANSPARENT: The background color of the image is transparent. • GRAYED: The colored image turns gray.

Parameter	Mandatory	Type	Description
dx	No	String	Horizontal offset of the watermark image relative to the output video. The default value is **0**. The value can be an integer or in decimals. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. • Integer: horizontal offset between the watermark start point and the video vertex, in pixels. Value range: [0, 4096]. • Decimal: horizontal offset ratio of the watermark start point to the video width. Value range: (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.

Parameter	Mandatory	Type	Description
dy	No	String	Vertical offset of the watermark image relative to the output video. The default value is **0**. The value can be an integer or in decimals. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. • Integer: vertical offset between the watermark start point and the video vertex, in pixels. Value range: [0, 4096]. • Decimal: vertical offset ratio of the watermark start point to the video height. Value range: (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.
position	No	String	Watermark position. The default value is **TOPRIGHT**. Possible options: • TOPRIGHT: upper right • TOPLEFT: upper left • BOTTOMRIGHT: lower right • BOTTOMLEFT: lower left

Parameter	Mandatory	Type	Description
width	No	String	Watermark image width. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. Possible values are: ● Integer: width of a watermark image, in pixels. The value ranges from 8 to 4,096. ● Decimal: ratio of the width to the width of the output video. The value is greater than 0 and less than 1, and can contain up to four decimal places, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.
height	No	String	Watermark image height. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. Possible values are: ● Integer: height of a watermark image, in pixels. The value ranges from 8 to 4,096. ● Decimal: ratio of the height to the height of the output video. The value is greater than 0 and less than 1, and can contain up to four decimal places, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.

Parameter	Mandatory	Type	Description
timeline_start	No	String	Watermark start time, which is used together with **timeline_duration** Value range: [0, END). "END" indicates the video end time. Unit: second
timeline_duration	No	String	Watermark duration, which is used together with "timeline_start". Value range: (0, END - start time]. "END" indicates the video end time. Unit: second Default value: **END**
type	Yes	String	Only PNG, JPG, or JPEG images can be used as a watermark template. Only JPG, JPEG, and PNG strings are supported.
md5	No	String	MD5 value of a watermark image

Response Parameters

Status code: 200

Table 6-4 Response body parameters

Parameter	Type	Description
id	String	Watermark configuration template ID.
upload_url	String	URL for uploading a watermark image.

Status code: 400

Table 6-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a watermark template.

```
POST https://{endpoint}/v1.0/{project_id}/template/watermark

Content-Type: application/json
{
  "name": "test",
  "type": "PNG"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "id" : "a52ba84366abebb4c4614e1b16973549",
  "upload_url" : "https://vod-bucket-26-cn-north-4.obs.cn-
north-4.myhuaweicloud.com:443/05ab5cef408026f22f62c018de60cf2e/
a52ba84366abebb4c4614e1b16973549/watermark.png?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1625479312&Signature=kZYh0hEos2V**********AHGyXA
%3D"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

6.2 Modifies a watermark template

Function

Modifies a watermark template

URI

PUT /v1.0/{project_id}/template/watermark

Table 6-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 6-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 6-8 Request body parameters

Parameter	Mandatory	Type	Description
id	Yes	String	Watermark template configuration ID.
name	Yes	String	Watermark template name.

Parameter	Mandatory	Type	Description
dx	No	String	Horizontal offset of the watermark image relative to the output video. The default value is **0**. The value can be an integer or in decimals. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. • Integer: horizontal offset between the watermark start point and the video vertex, in pixels. Value range: [0, 4096]. • Decimal: horizontal offset ratio of the watermark start point to the video width. The value range is (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.

Parameter	Mandatory	Type	Description
dy	No	String	Vertical offset of the watermark image relative to the output video. The default value is **0**. The value can be an integer or in decimals. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. ● Integer: vertical offset between the watermark start point and the video vertex, in pixels. Value range: [0, 4096]. ● Decimal: vertical offset ratio of the watermark start point to the video height. The value range is (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.
position	No	String	Watermark position. The default value is **TOPRIGHT**. Possible options: ● TOPRIGHT: upper right ● TOPLEFT: upper left ● BOTTOMRIGHT: lower right ● BOTTOMLEFT: lower left

Parameter	Mandatory	Type	Description
width	No	String	Watermark image width. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. Possible options: • Integer: width of a watermark image, in pixels. The value ranges from 8 to 4,096. • Decimal: ratio of the width to the width of the output video. The value range is (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.
height	No	String	Watermark image height. You are advised to set this parameter to a value in decimals. If the value is an integer, images on the console can be transcoded but may fail to be previewed. Possible options: • Integer: height of a watermark image, in pixels. The value ranges from 8 to 4,096. • Decimal: ratio of the height to the height of the output video. The value range is (0, 1). Up to four decimal places are displayed, for example, 0.9999. If the value contains more than four decimal places, the part behind the fourth decimal place is automatically truncated.

Parameter	Mandatory	Type	Description
watermark_type	No	String	Watermark type. Currently, only IMAGE (image watermark) is supported.
image_process	No	String	This parameter is valid when **watermark_type** is set to **IMAGE**. Possible options: • ORIGINAL: Only simple scaling is performed. • TRANSPARENT: The background color of the image is transparent. • GRAYED: The colored image turns gray.
timeline_start	No	String	Watermark start time, which is used together with **timeline_duration** Value range: [0, END). "END" indicates the video end time. Unit: second
timeline_duration	No	String	Watermark duration, which is used together with "timeline_start". Value range: (0, END - start time]. "END" indicates the video end time. Unit: second Default value: **END**

Response Parameters

Status code: 400

Table 6-9 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modifies a watermark template.

```
PUT https://{endpoint}/v1.0/{project_id}/template/watermark

Content-Type: application/json
{
  "id": "2305739f855413a84af9e6ad6ebb21be",
  "name": "test"
}
```

Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

6.3 Queries watermark templates

Function

Queries a watermark template

URI

GET /v1.0/{project_id}/template/watermark

Table 6-10 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 6-11 Query Parameters

Parameter	Mandatory	Type	Description
id	No	Array of strings	Watermark template ID. A maximum of 10 IDs can be configured at a time.
page	No	Integer	Page number This parameter defaults to **0** and is invalid when **id** is specified.
size	No	Integer	Number of records on each page The value defaults to **10** and ranges from 1 to 100. This parameter is invalid when **id** is specified.

Request Parameters

Table 6-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 6-13 Response body parameters

Parameter	Type	Description
templates	Array of **WatermarkTemplate** objects	Watermark template information
total	Integer	Total number of watermark templates

Table 6-14 WatermarkTemplate

Parameter	Type	Description
name	String	Name of a watermark template
id	String	Watermark template configuration ID
status	Long	Watermark template status Possible values are: • 0: disabled • 1: enabled
dx	String	Horizontal offset between the watermark and the output video The default value is **0**.
dy	String	Vertical offset between the watermark and the output video The default value is **0**.
position	String	Watermark position
width	String	Watermark image width
height	String	Watermark image height
create_time	String	Creation time
image_url	String	URL for downloading a watermark image
type	String	Watermark image format
watermark_type	String	Watermark type. Currently, only IMAGE (image watermark) is supported.

Parameter	Type	Description
image_process	String	This parameter is valid when **watermark_type** is set to **IMAGE**. Possible options: • ORIGINAL: Only simple scaling is performed. • TRANSPARENT: The background color of the image is transparent. • GRAYED: The colored image turns gray.
timeline_start	String	Watermark start time, which is used together with **timeline_duration** Value range: [0, END). "END" indicates the video end time. Unit: second
timeline_duration	String	Watermark duration, which is used together with "timeline_start". Value range: (0, END - start time]. "END" indicates the video end time. Unit: second Default value: **END**

Status code: 400

Table 6-15 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

```
GET https://{endpoint}/v1.0/{project_id}/template/watermark?id={id}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "templates" : [ {
    "name" : "test",
    "id" : "2305739f855413a84af9e6ad6ebb21be",
    "status" : 0,
```



```
    "dx" : "0.05",
    "dy" : "0.05",
    "position" : "TOPRIGHT",
    "width" : "0.12",
    "height" : null,
    "create_time" : "20210204092325",
    "image_url" : "https://103-cn-north-4.cdn-vod.huaweicloud.com/05ab5cef408026f22f62c018de60cf2e/
watermark/2305739f855413a84af9e6ad6ebb21be.png",
    "type" : "PNG",
    "watermark_type" : "IMAGE",
    "image_process" : "TRANSPARENT",
    "timeline_start" : null,
    "timeline_duration" : null
  } ],
  "total" : 1
}
```


Status code: 400

The information is returned when the request fails.


```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

6.4 Deletes a watermark template

Function

Deletes a watermark template

URI

DELETE /v1.0/{project_id}/template/watermark

Table 6-16 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 6-17 Query Parameters

Parameter	Mandatory	Type	Description
id	Yes	String	Watermark template ID

Request Parameters

Table 6-18 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 6-19 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Deletes a watermark template.

```
DELETE https://{endpoint}/v1.0/{project_id}/template/watermark?id={id}
```


Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

6.5 Confirms that the watermark image has been uploaded

Function

Checks the upload status of the watermark image

URI

POST /v1.0/{project_id}/watermark/status/uploaded

Table 6-20 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 6-21 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 6-22 Request body parameters

Parameter	Mandatory	Type	Description
id	Yes	String	Watermark configuration template ID
status	Yes	String	Watermark upload status. The value can be ** "SUCCEED"** or ** "FAILED"**.

Response Parameters

Status code: 200

Table 6-23 Response body parameters

Parameter	Type	Description
id	String	Watermark configuration template ID.
image_url	String	URL for downloading a watermark image.

Status code: 400

Table 6-24 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Confirms the watermark image upload status.

```
POST https://{endpoint}/v1.0/{project_id}/watermark/status/uploaded

Content-Type: application/json
{
  "id": "2305739f855413a84af9e6ad6e2b21be",
  "status": "SUCCEED"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "id" : "2305739f855413a84af9e6ad6e2b21be",
  "image_url" : "https://vod-bucket-26-cn-north-4.obs.cn-
north-4.myhuaweicloud.com:443/05ab5cef408026f22f62c018de60cf2e/
a52ba84366abebb4c4614e1b16973549/watermark.png?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1625479312&Signature=kZYh0hEos2V**********AHGyXA
%3D"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

7 Media Asset Storage Mode Management

7.1 Modifying the Cold Storage Scope of a Media Asset

Function

Modifies the cold storage scope of a media asset.

By default, cold storage applies to the source and output media files. You can also apply cold storage only to the source media file.

URI

PUT /v1/{project_id}/asset/storage-mode-type

Table 7-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 7-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 7-3 Request body parameters

Parameter	Mandatory	Type	Description
storage_mode_type	No	String	Cold storage mode. Options: • **WHOLE**: Cold storage applies to both the source and transcoded media files. • **ORIGIN**: Cold storage applies only to the source media file.

Response Parameters

Status code: 400

Table 7-4 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modify the cold storage scope of a media asset.

```
PUT https://{endpoint}/v1/{project_id}/asset/storage-mode-type

Content-Type: application/json
{
  "storage_mode_type": "ORIGIN"
}
```

Example Responses

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

7.2 Changing the OBS Storage Class of a Media Asset

Function

Changes the OBS storage class of a media file.

By default, the **Standard** storage class applies to media file upload. You can change the storage class to **Infrequent Access** or **Archive**.

- **Infrequent Access** can be changed to **Standard** or downgraded to **Archive**.
- **Archive** can only be changed to **Standard** and cannot be directly changed to **Infrequent Access**.
- The storage class cannot be changed when a media asset has an ongoing task, such as snapshot capturing or transcoding.

About OBS billing:

- The minimum storage period of **Infrequent Access** is 30 days, and that of **Archive** is 90 days. If media files are retrieved or deleted before the storage period expires, you will still be charged based on the minimum storage period.

Infrequent Access is used as an example. If the actual storage period is < 30 days, you will be charged based on the minimum storage period (30 days). If the actual storage period is ≥ 30 days, you will be charged based on the actual storage period.

- You will be charged for media file retrieval when changing the storage class from **Infrequent Access** or **Archive** to **Standard**.

URI

PUT /v1/{project_id}/asset/storage-mode

Table 7-5 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 7-6 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 7-7 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Original media asset ID.

Parameter	Mandatory	Type	Description
storage_mode	Yes	String	Storage mode. Options: • **STANDARD**: OBS Standard • **WARM**: Infrequent Access • **COLD**: Archive
restore_mode	No	String	How the storage class is changed from **Archive** to another option. Options: • **TEMP**: temporarily • **FOREVER**: permanently
days	No	Integer	Time when the storage class is temporarily changed from **Archive** to **Standard**. Value range: 1–30. Unit: day
restore_tier	No	String	Archive restoration options: • **EXPEDITED**: quick restoration • **STANDARD**: standard restoration Default value: **EXPEDITED**

Response Parameters

Status code: 200

Table 7-8 Response body parameters

Parameter	Type	Description
task_result_array	Array of **TaskResult** objects	Delivery result of the task for changing the media asset storage mode.

Table 7-9 TaskResult

Parameter	Type	Description
asset_id	String	Media asset ID.

Parameter	Type	Description
status	String	Checks whether the task for changing the media asset storage mode is successfully delivered. Options: • **SUCCEED** • **FAILED**

Status code: 400

Table 7-10 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Changing the OBS Storage Class of a Media Asset

```
PUT https://{endpoint}/v1/{project_id}/asset/storage-mode

Content-Type: application/json
{
  "asset_id": "2305739f855413a84af9e6ad6e2b21be",
  "storage_mode": "WARM"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeded.

```
{
  "task_result_array" : [ {
    "asset_id" : "2305739f855413a84af9e6ad6e2b21be",
    "status" : "SUCCEED"
  } ]
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

7.3 Querying the Cold Storage Settings of a Media Asset

Function

Query the cold storage settings of a media asset.

URI

GET /v1/{project_id}/asset/storage-mode-type

Table 7-11 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 7-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 7-13 Response body parameters

Parameter	Type	Description
storage_mode_type	String	Cold storage mode. Options: • **WHOLE**: Cold storage applies to both the source and transcoded media files. • **ORIGIN**: Cold storage applies only to the source media file.

Status code: 400

Table 7-14 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Query the cold storage scope of a media asset.

```
GET https://{endpoint}/v1/{project_id}/asset/storage-mode-type
```

Example Responses

Status code: 200

The information is returned when the request succeeded.


```
{
  "storage_mode_type" : "ORIGIN"
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

8 Media file pre-loading

8.1 CDN prefetch

Function

After media files are published, you can call this API to request CDNs for media file pre-loading by ID or URL. Upon the first request, the CDN caches the requested media file for faster download and better user experience. One tenant can pre-load a maximum of 1,000 media files per day.

URI

POST /v1.0/{project_id}/asset/preheating

Table 8-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 8-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 8-3 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	No	String	ID of a published media asset
urls	No	Array of strings	Streaming URL list of published media assets. A maximum of 10 URLs can be prefetched at a time.

Response Parameters

Status code: 202

Table 8-4 Response body parameters

Parameter	Type	Description
task_id	String	Prefetch task ID

Status code: 400

Table 8-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

- Prefetches media files by media file ID.

```
POST https://{endpoint}/v1.0/{project_id}/asset/preheating

Content-Type: application/json

{
  "asset_id": "f488337c31c8e4622f1590735b134c65"
}
```

- Prefetches media files by media file URL.

```
POST https://{endpoint}/v1.0/{project_id}/asset/preheating

Content-Type: application/json{  "urls": [     "     https://example.com/asset/
9db42f5e08c15edecd99a98da241994a/313bfd52a75f95ff48e8bf02eca2ab20.flv",     "     https://
example.com/asset/
9e455adb02295aa123809e8dc7ca51c1/68b1241af3bf58bcde9914626e07f5af.mp4",     "     https://
example.com/asset/9e455adb02295aa123809e8dc7ca51c1/play_video/
68b1241af3bf58bcde9914626e07f5af_H.264_480X270_HEAACV1_300.mp4"  ]}
```

Example Responses

Status code: 202

The information is returned when the request succeeds.

```
{
  "task_id" : "5199337c31c8e4622f1590735b13a263"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
202	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

8.2 Queries information of CDN prefetch

Function

Queries the prefetch result

URI

GET /v1.0/{project_id}/asset/preheating

Table 8-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 8-7 Query Parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	Task ID

Request Parameters

Table 8-8 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.

Parameter	Mandatory	Type	Description
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 8-9 Response body parameters

Parameter	Type	Description
preheating_results	Array of **PreheatingResult** objects	Prefetch task array

Table 8-10 PreheatingResult

Parameter	Type	Description
url	String	Media asset URL
status	String	Pre-loading task status Possible values are: • processing: The task is being processed. • succeed: The pre-loading is complete. • failed: The pre-loading failed.

Status code: 400

Table 8-11 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

```
GET https://{endpoint}/v1.0/{project_id}/asset/preheating?task_id={task_id}
```


Example Responses

Status code: 200

The information is returned when the request succeeded.

```
{
  "preheating_results" : [ {
    "url" : "https://example.com/asset/9db42f5e08c15edecd99a98da241994a/
313bfd52a75f95ff48e8bf02eca2ab20.flv",
    "status" : "processing"
  }, {
    "url" : "https://example.com/asset/
9e455adb02295aa123809e8dc7ca51c1/68b1241af3bf58bcde9914626e07f5af.mp4",
    "status" : "succeed"
  }, {
    "url" : "https://example.com/asset/9e455adb02295aa123809e8dc7ca51c1/play_video/
68b1241af3bf58bcde9914626e07f5af_H.264_480X270_HEAACV1_300.mp4",
    "status" : "failed"
  } ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9 Media file processing

9.1 Updates a video

Function

After a media file is created, upload a thumbnail, update a video file, or update the existing thumbnail.After a video file is updated, call the API for **Confirming Media Upload** to notify VOD.You do not need to call the API after updating or uploading a thumbnail.You can upload an updated video by part. For details, see **Example 2: Uploading a Media File More Than 20 MB by Part**.

Constraints

The video update API can be used to update only media assets that fail to be downloaded or uploaded, or whose upload has been canceled.

Rules for configuring parameters in the request body:

- Configure at least one of the following parameters: **video_name**, **cover_type**, and **subtitles**.
- To update a video, **video_name** and **video_type** must be specified.
- To update a thumbnail, **cover_type** is mandatory and cannot be left empty. If **cover_id** is specified, the value must be **0**.
- To upload a subtitle file, the **id**, **type**, and **language** of **subtitles** must be specified.

If the parameters are not specified as instructed, API response will fail.

URI

PUT /v1.0/{project_id}/asset

Table 9-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 9-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 9-3 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
video_md5	No	String	MD5 value of a video fileYou are advised to refer to the example of uploading and updating media files in Generating an MD5 Value in the appendix of *API Reference*.
video_name	No	String	Video file name The file name extension is optional.

Parameter	Mandatory	Type	Description
video_type	No	String	Video file type The value can be MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, or MKV.
cover_id	No	Integer	Thumbnail ID The value ranges from 0 to 7. Currently, only one thumbnail is supported. Therefore, this parameter must be set to 0.
cover_type	No	String	Thumbnail image format Possible values are: • JPG • PNG
cover_md5	No	String	MD5 value of the cover file
subtitles	No	Array of **Subtitle** objects	Subtitle file information

Table 9-4 Subtitle

Parameter	Mandatory	Type	Description
id	Yes	Integer	Subtitle ID. The value ranges from **1** to **16**.
type	Yes	String	Subtitle file format. Currently, only SRT and VTT are supported.
language	Yes	String	Subtitle language.
name	No	String	Subtitle file name.
md5	No	String	MD5 value of the subtitle file
description	No	String	Subtitle description

Response Parameters

Status code: 200

Table 9-5 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
video_upload_url	String	URL for uploading video files This API is called by the client to upload a small video file not greater than 20 MB. The URL contains temporary authorization information. If the file size is greater than 20 MB, the file needs to be uploaded in multipart mode. **NOTE** You can use PUT requests to upload video files to **video_upload_url**. Configure **Content-Type** according to the type of the video file to be uploaded. The value is in the format of video/*video format*, for example, video/mp4.
cover_upload_url	String	URL for uploading thumbnail files **NOTE** You can use PUT requests to upload thumbnail images to **cover_upload_url**. Configure **Content-Type** according to the type of the thumbnail file to be uploaded. The value is in the format of image/*image format*, for example, image/png.
subtitle_upload_urls	Array of strings	URL for uploading subtitles **NOTE** You can use PUT requests to upload subtitle files to **subtitle_upload_urls**. Configure **Content-Type** according to the type of the subtitle file to be uploaded. For example, the value can be application/octet-stream.

Status code: 403

Table 9-6 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Updates a media file.

```
PUT https://{endpoint}/v1.0/{project_id}/asset

Content-Type: application/json
{
  "asset_id": "f488337c31c8e4622f1590735b134c65",
  "cover_id": "0",
  "cover_type": "JPG",
  "subtitles": [
    {
      "id": 1,
      "language": "CN",
      "type": "SRT",
      "md5": "SqcyFjJZoDZaP8oKIY6rgQ==",
      "description": "AAAAA"
    }
  ]
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "f488337c31c8e4622f1590735b134c65",
  "cover_upload_url" : "https://obs.cn-north-4.myhuaweicloud.com:443/obs-vod-1/%7Bproject_id%7D/
f488337c31c8e4622f1590735b134c65/cover/Cover0.jpg?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1518148410&Signature=kZYh0hEos2V**********AHGyXA
%3D",
  "subtitle_upload_urls" : [ "https://obs-vod-1.obs.cn-
north-4.myhuaweicloud.com:443/14ce1d4437164aba8b364ce15866154e/
53a018d2dc53ca07eb5a07a839205c9d/subtitle/1.srt?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1534760131&Signature=kZYh0hEos2V**********AHGyXA
%3D" ]
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.2 Media asset processing

Function

Transcodes, captures snapshots of, or encrypts a video. You can start one or multiple operations at a time.

URI

POST /v1.0/{project_id}/asset/process

Table 9-7 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 9-8 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 9-9 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
template_group_name	No	String	Name of a transcoding template group **NOTE** If this parameter is specified, the specified transcoding template is used to transcode the uploaded audio/video. You can configure a transcoding template on the VOD console. For details, see Transcoding Settings in *VOD User Guide*.
auto_encrypt	No	Integer	Whether to automatically encrypt a file Possible values are: • 0: not encrypted • 1: encrypted Default value: **0** A file must be encrypted and transcoded at the same time. When encryption is required, the transcoding parameter cannot be empty and the output file must be in HLS format.
thumbnail	No	**Thumbnail** object	Snapshot parameters
subtitle_id	No	Array of integers	Subtitle file ID. **NOTE** • This parameter takes effect only when the APIs for **Uploading a Media Asset to VOD** and **Updating a Video** are called and the request parameter **subtitles** is specified. • If this parameter is not specified, the last uploaded subtitle is displayed in the video stream by default. • Subtitle files in a video stream can only be SRT.

Table 9-10 Thumbnail

Parameter	Mandatory	Type	Description
type	Yes	String	Snapshot capturing mode. The options are as follows: • **time**: Snapshots are captured by interval. • **dots**: Snapshots are captured at a specified time point. • **quantity**: Snapshots are captured based on the specified quantity and video duration.
quantity	No	Integer	This parameter is mandatory when **type** is set to **quantity**. Snapshots are captured based on the specified quantity and video duration. Value range: an integer between 1 and 10
quantity_time	No	Integer	This parameter is optional when **type** is set to **quantity**. Snapshots are captured based on the specified quantity at a specified interval. Value range: an integer between 0 and 2,147,483,647
time	No	Integer	Interval for sampling, in seconds. **type** is set to **time**. Default value: **12** Value range: an integer between 0 and 100
dots	No	Array of integers	This parameter is mandatory when type is set to dots. The array of time points when a snapshot is captured is used.
cover_position	No	Integer	The value indicates which snapshot is specified as the thumbnail. The default value is **1**.

Parameter	Mandatory	Type	Description
format	No	Integer	Snapshot file format Possible values are: • 1: jpg The default value is **1**.
aspect_ratio	No	Integer	Aspect ratio Possible values are: • 0: adaptive (the original aspect ratio is retained) • 1: 16:9 Default value: **0**
max_length	No	Integer	The longest side of a snapshot. Unit: pixel The width of the snapshot is scaled proportionally with the longest side and input video pixel. Default value: **480**

Response Parameters

Status code: 202

Table 9-11 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Status code: 403

Table 9-12 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Media file transcoding.

```
POST https://{endpoint}/v1.0/{project_id}/asset/process

Content-Type: application/json
{
  "asset_id": "b4f39691d66cc0ef75d62ee567146e11",
  "template_group_name": "test",
  "auto_encrypt": 0,
  "thumbnail": {
    "type": "time",
    "time": 12
  },
  "subtitle_id": [
    1
  ]
}
```

Example Responses

Status code: 202

The information is returned when the request succeeds.

```
{
  "asset_id" : "b4f39691d66cc0ef75d62ee567146e11"
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
202	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.3 Cancels a media asset transcoding task

Function

Only transcoding tasks in the queue can be canceled.

URI

DELETE /v1.0/{project_id}/asset/process

Table 9-13 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 9-14 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Request Parameters

Table 9-15 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 9-16 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Cancels a queuing media file transcoding task.

```
DELETE https://{endpoint}/v1.0/{project_id}/asset/process?asset_id={asset_id}
```

Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10083",
  "error_msg" : "The current media asset status does not support this operation."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.4 Extracts audio

Function

This API is an asynchronous API. After an audio extraction task is created and delivered, asset_id and the extracted audio_asset_id are returned. However, the audio extraction task is not complete immediately. You can check whether the audio extraction task is complete in the audio extraction completion event on the event notification page.

URI

POST /v1.0/{project_id}/asset/extract_audio

Table 9-17 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 9-18 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 9-19 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
parameter	No	**Parameter** object	Parameters of the audio to be extracted

Table 9-20 Parameter

Parameter	Mandatory	Type	Description
format	No	String	Packaging format Possible values are: • MP3 • AAC

Response Parameters

Status code: 202

Table 9-21 Response body parameters

Parameter	Type	Description
asset_id	String	Media ID of the source video
audio_asset_id	String	Media ID of the extracted audio

Status code: 400

Table 9-22 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Extracts an audio file.

```
POST https://{endpoint}/v1.0/{project_id}/asset/extract_audio

Content-Type: application/json
{
  "asset_id": "3e1cd21131a94525be55acf65888bf46",
  "parameter": {
    "format": "MP3"
  }
}
```

Example Responses

Status code: 202

The information is returned when the request succeeds.


```
{
  "asset_id" : "f488337c31c8e4622f1590735b134c65",
  "audio_asset_id" : "5412"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
202	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.5 Cancels an audio extraction task

Function

Only audio extraction tasks in the queue can be canceled.

URI

DELETE /v1.0/{project_id}/asset/extract_audio

Table 9-23 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 9-24 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Request Parameters

Table 9-25 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 9-26 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Cancels an audio extraction task.

```
DELETE https://{endpoint}/v1.0/{project_id}/asset/extract_audio?asset_id={asset_id}
```


Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10083",
  "error_msg" : "The current media asset status does not support this operation."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.6 Creates a media asset review task

Function

Reviews the uploaded media asset. You can call the API for **Querying File Details** to view the review result.

Constraints

Currently, only the VOD service in the CN North-Beijing1 and CN North-Beijing4 regions supports this function.

URI

POST /v1.0/{project_id}/asset/review

Table 9-27 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 9-28 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 9-29 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
review	Yes	**Review** object	Media asset review parameter. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)

Table 9-30 Review

Parameter	Mandatory	Type	Description
template_id	No	String	Review template ID. You can obtain the value after configuring the review template on the VOD console. For details, see Review Settings in *VOD User Guide*. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
interval	No	Integer	Snapshot check interval. The value range is (0,100]. This parameter is ignored in request parameters.
politics	No	Integer	Confidence of politically sensitive content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
terrorism	No	Integer	Confidence of terrorism-related content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Parameter	Mandatory	Type	Description
porn	No	Integer	Confidence of pornographic content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Response Parameters

Status code: 200

Table 9-31 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
review	**Review** object	Media asset review parameter. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)

Table 9-32 Review

Parameter	Type	Description
template_id	String	Review template ID. You can obtain the value after configuring the review template on the VOD console. For details, see Review Settings in *VOD User Guide*. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
interval	Integer	Snapshot check interval. The value range is (0,100]. This parameter is ignored in request parameters.
politics	Integer	Confidence of politically sensitive content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
terrorism	Integer	Confidence of terrorism-related content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
porn	Integer	Confidence of pornographic content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Status code: 400

Table 9-33 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Reviews an uploaded media file.

```
POST https://{endpoint}/v1.0/{project_id}/asset/review

Content-Type: application/json
{
  "asset_id": "3e1cd21131a94525be55acf65888bf46",
  "review": {
    "template_id": "c80e56dadb8542e8a1b7c2224dd6733a"
  }
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "ede1391f8be15b4bcf99099c8d437c00",
  "review" : {
    "template_id" : "c80e56dadb8542e8a1b7c2224dd6733a",
    "interval" : 5,
    "politics" : 0,
    "terrorism" : 0,
    "porn" : 1
  }
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.

Status Code	Description
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

9.7 Sets the cover

Function

Sets a video snapshot as the cover

URI

PUT /v1.0/{project_id}/asset/cover

Table 9-34 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 9-35 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.

Parameter	Mandatory	Type	Description
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 9-36 Request body parameters

Parameter	Mandatory	Type	Description
thumbnailUrl	Yes	String	URL of the snapshot file Call the API for **Querying File Details** to obtain the snapshot file URL of a media file based on the media file ID.

Response Parameters

Status code: 400

Table 9-37 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Captures a video snapshot as the video thumbnail.

```
PUT https://{endpoint}/v1.0/{project_id}/asset/cover

Content-Type: application/json

{
  "thumbnailUrl": "
  https://355.cdn-vod.huaweicloud.com/shield/asset/6e76f92034ceae2fcdcef9413221511f/snapshot/sample/0.jpg"
}
```

Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```


Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

10 Statistical analysis

10.1 Queries CDN statistics

Function

Queries CDN statistics, including traffic, peak bandwidth, total number of requests, request hit ratio, and traffic hit ratio. The statistics are displayed with a one-hour delay.

URI

GET /v1.0/{project_id}/asset/cdn-statistics

Table 10-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 10-2 Query Parameters

Parameter	Mandatory	Type	Description
start_time	No	String	Start time. The format is yyyymmddhhmmss.

Parameter	Mandatory	Type	Description
end_time	No	String	End time. The format is yyyymmddhhmmss. • If **start_time** and **end_time** are not specified, set **start_time** to 00:00:00 on the current day and **end_time** to the current time. • If **end_time** is specified but **start_time** is not, the request is invalid. • If **start_time** is specified but **end_time** is not, set **end_time** to the current time. • Only data of the past three months can be queried and the time span cannot exceed 31 days. • The start time and end time are automatically rounded. The start time is rounded to the hour of the specified time and the end time is rounded to the next hour of the specified time.
stat_type	Yes	String	Statistics type Possible values are: • cdn_bw: CDN peak bandwidth • cdn_flux: CDN traffic • req_num: number of requests • req_hit_rate: request hit ratio • flux_hit_rate: traffic hit ratio Only one type of statistics can be queried at a time.

Parameter	Mandatory	Type	Description
domain	Yes	String	Domain name list. Multiple domain names are separated by commas (,). Example: example.test1.com,example.test2.com **ALL** indicates that all domain names under a tenant are queried. A maximum of 100 domain names can be queried at a time.
interval	No	Integer	Query granularity interval Possible values are: • For a time span of 1 day, the values 5 minutes, 1 hour, 4 hours, and 8 hours correspond to 300 seconds, 3,600 seconds, 14,400 seconds, and 28,800 seconds, respectively. • For a time span of 2 to 7 days, the values 1 hour, 4 hours, 8 hours, and 1 day correspond to 3,600 seconds, 14,400 seconds, 28,800 seconds, and 86,400 seconds, respectively. • For a time span of 8 to 31 days, the values 4 hours, 8 hours, and 1 day correspond to 14,400 seconds, 28,800 seconds, and 86,400 seconds, respectively. Unit: second If this parameter is not specified, the minimum interval of the corresponding time span is used by default.

Request Parameters

Table 10-3 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 10-4 Response body parameters

Parameter	Type	Description
start_time	String	Start time of statistics collection
interval	Integer	Statistics collection interval
values	Array of longs	Sampled data array. Sampling starts from **start_time**, and one data record will be collected at each interval (traffic unit: byte; bandwidth unit: bit/s).

Status code: 400

Table 10-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries CDN statistics.

```
GET https://{endpoint}/v1.0/{project_id}/asset/cdn-statistics?domain=www.example.com&stat_type=cdn_bw
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "start_time" : "string",
  "interval" : 0,
  "values" : [ 0 ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

10.2 Queries statistics on the origin server

Function

Queries statistics on the VOD origin server, including storage space and transcoded output.

Only historical data of the last month can be queried.

URI

GET /v1.0/{project_id}/asset/vod-statistics

Table 10-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 10-7 Query Parameters

Parameter	Mandatory	Type	Description
start_time	No	String	Start time. The format is yyyymmddhhmmss.
end_time	No	String	End time. The format is yyyymmddhhmmss. If **start_time** and **end_time** are not specified, set **start_time** to 00:00:00 on the current day and **end_time** to the current time. • If **end_time** is specified but **start_time** is not, the request is invalid. • If **start_time** is specified but **end_time** is not, set **end_time** to the current time. • Only data of the past three months can be queried and the time span cannot exceed 31 days. • The start time and end time are automatically rounded. The start time is rounded to the hour of the specified time and the end time is rounded to the next hour of the specified time.

Parameter	Mandatory	Type	Description
interval	No	Integer	Query granularity interval Possible values are: • For a time span of 1 day, the values 1 hour, 4 hours, and 8 hours correspond to 3,600 seconds, 14,400 seconds, and 28,800 seconds, respectively. • For a time span of 2 to 7 days, the values 1 hour, 4 hours, 8 hours, and 1 day correspond to 3,600 seconds, 14,400 seconds, 28,800 seconds, and 86,400 seconds, respectively. • For a time span of 8 to 31 days, the values 4 hours, 8 hours, and 1 day correspond to 14,400 seconds, 28,800 seconds, and 86,400 seconds, respectively. Unit: second If this parameter is not specified, the minimum interval of the corresponding time span is used by default.

Request Parameters

Table 10-8 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.

Parameter	Mandatory	Type	Description
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 10-9 Response body parameters

Parameter	Type	Description
start_time	String	Start time of statistics collection
interval	Integer	Statistics collection interval
sample_data	Array of VodSampleData objects	Sampled data array. Sampling starts from **start_time**, and one data record will be collected at each interval (transcoded output unit: min; storage unit: GB).

Table 10-10 VodSampleData

Parameter	Type	Description
storage	Float	Standard storage space. Unit: GB
storage_warm	Float	Infrequent access storage space. Unit: GB
storage_cold	Float	Archive storage space. Unit: GB
transcode	Long	Transcoded file duration. Unit: minute

Status code: 400

Table 10-11 Response body parameters

Parameter	Type	Description
error_code	String	Error code

Parameter	Type	Description
error_msg	String	Error description

Example Requests

Queries statistics on the VOD origin server.

```
GET https://{endpoint}/v1.0/{project_id}/asset/vod-statistics?start_time={start_time}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "start_time" : "20190528000000",
  "interval" : 3600,
  "sample_data" : [ {
    "storage" : 0,
    "transcode" : 0
  } ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

10.3 Queries statistics on the top N media assets

Function

Queries statistics on the top 100 media assets by playback time for a specified domain name on a specified date

URI

GET /v1.0/{project_id}/asset/top-statistics

Table 10-12 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 10-13 Query Parameters

Parameter	Mandatory	Type	Description
domain	Yes	String	Queries a domain name. Currently, only one or all domain names can be queried. Possible values are: • Single acceleration domain name. The format is as follows: example.test1.com. • **ALL** indicates that all domain names under a tenant are queried.
date	Yes	String	Query date. The format is yyyymmdd. • The value of date must be the previous day or earlier. • You can query data of the past month.

Request Parameters

Table 10-14 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 10-15 Response body parameters

Parameter	Type	Description
top_urls	Array of **TopUrl** objects	Specifies the 100 most requested URLs of a domain name.

Table 10-16 TopUrl

Parameter	Type	Description
value	Long	Total playback times
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
title	String	Media asset name
duration	Integer	Media file duration Unit: second
duration_ms	Long	Video duration, in milliseconds.
size	Long	Size of the original media file Unit: byte

Status code: 400

Table 10-17 Response body parameters

Parameter	Type	Description
error_code	String	Error code

Parameter	Type	Description
error_msg	String	Error description

Example Requests

Queries the top 100 most queried media files.

```
GET https://{endpoint}/v1.0/{project_id}/asset/top-statistics?domain=ALL&date=20190612
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "top_urls" : [ {
    "value" : 0,
    "asset_id" : "548b6023bf2cc925bd64343f04ca0f88",
    "title" : "test",
    "duration" : 0,
    "duration_ms" : 0,
    "size" : 0
  } ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

10.4 Queries playback logs of a domain name

Function

Queries CDN logs of a specified VOD domain name in a specified period

URI

GET /v1.0/{project_id}/vod/cdn/logs

Table 10-18 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 10-19 Query Parameters

Parameter	Mandatory	Type	Description
domain_name	Yes	String	Acceleration domain name, for example, www.test1.com
query_date	Yes	String	Query start time. The format is yyyyMMddHHmmss. • The query result is the log data within 24 hours after the start time. • You can query data of the past month.
page_size	No	Integer	Number of logs displayed on each page. The value ranges from 1 to 10,000 (default value).
page_number	No	Integer	Current page number. The value ranges from 1 (default value) to 65,535.

Request Parameters

Table 10-20 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 10-21 Response body parameters

Parameter	Type	Description
total	Integer	Total number of logs
logs	Array of **CdnLog** objects	Data of the log list

Table 10-22 CdnLog

Parameter	Type	Description
domain_name	String	Domain name
start_time	String	Query start time
end_time	String	Query end time
name	String	Log name
size	Long	Log size Unit: byte

Parameter	Type	Description
link	String	Link for downloading the log file For the description of the fields in the log file, see **Log Management**.

Status code: 400

Table 10-23 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries playback logs.

```
GET https://{endpoint}/v1.0/{project_id}/vod/cdn/logs?domain_name=1866.cdn-vod.huaweicloud.com&query_date=20211223000000
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "total" : 1,
  "logs" : [ {
    "domain_name" : "1866.cdn-vod.huaweicloud.com",
    "start_time" : "2021-12-23 23:00:00",
    "end_time" : "2021-12-24 00:00:00",
    "name" : "2021122323-1866.cdn-vod.huaweicloud.com-cn.gz",
    "size" : 225,
    "link" : "http://cdn-log-user-bj4.obs.cn-north-4.myhuaweicloud.com:80/mainland/20211223/23/2021122323-1866.cdn-vod.huaweicloud.com-cn.gz?AccessKeyId=AOV5GWALBMNTLIYDEQLV&Expires=1641433717&response-content-disposition=attachment%3Bfilename%3D%222021122323-1866.cdn-vod.huaweicloud.com-cn.gz%22&Signature=kZYh0hEos2V*********AHGyXA%3D"
  } ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

10.5 Querying Daily Playback Statistics of a Media Asset

Function

Queries daily playback statistics of a media asset.

Before using this API, you need to submit a service ticket to enable the statistics function to trigger a statistics task.

Playback statistics of the past year can be queried.

URI

GET /v1/{project_id}/asset/daily-summary

Table 10-24 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 10-25 Query Parameters

Parameter	Mandatory	Type	Description
start_time	Yes	String	Start time of the period to be queried. You can query data of the past year, and the time span for one query cannot exceed 90 days. The format of the start time of the period to be queried is yyyyMMdd000000.

Parameter	Mandatory	Type	Description
end_time	Yes	String	End time of the period to be queried. You can query data of the past year, and the time span for one query cannot exceed 90 days. The format of the end time of the period to be queried is yyyyMMdd000000.
offset	No	Integer	Offset. The records after this offset will be queried.
limit	No	Integer	Maximum number of records that can be returned.

Request Parameters

Table 10-26 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 10-27 Response body parameters

Parameter	Type	Description
total	Integer	Total records.

Parameter	Type	Description
summary_results	Array of **AssetDailySummaryResult** objects	Log files.

Table 10-28 AssetDailySummaryResult

Parameter	Type	Description
date	String	Playback date. The format is yyyyMMdd000000.
link	String	URL for downloading daily playback statistics files. The validity period is 12 hours. File content format: [*Domain name*]\t[*Media asset ID*]\t[*Date*]\t[*Playback traffic*]\t[*Playback times*] Description of playback times statistics: • HLS files: When M3U8 files are accessed, the number of playback times is counted. When TS files are accessed, the number of playback times is not counted. • For other files, such as MP4 files, if the playback request contains **range** and the value of **start** in **range** is not **0**, the number of playback times is not counted. In other cases, the number of playback times is counted.

Status code: 400

Table 10-29 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

None

Example Responses

Status code: 200

The information is returned when the request succeeded.

```
{
  "total" : 1,
  "summary_results" : [ {
    "date" : "20231201000000",
    "link" : "http://cdn-log-user-bj4.obs.cn-north-4.myhuaweicloud.com:80/asset-play-summary/
05a8aee14d8026a92fcdc018fe235c2c/20231201-asset-play-summary.gz?
AccessKeyId=QXBNBGA2GW9EP2XXOYUN&Expires=1701786544&Signature=kZYh0hEos2V**********AHGyXA
%3D"
  } ]
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

11 Transcoded Output Management

11.1 Deleting a Transcoded Output

Function

Deletes a transcoded output.

URI

DELETE /v1/{project_id}/asset/transcode-product

Table 11-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID

Request Parameters

Table 11-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 11-3 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	No	String	Media asset ID.
delete_type	No	String	**GROUP**: Deletion at the template group level **PRODUCT**: Deletion at the product level
delete_infos	No	Array of **ProductGroupInfo** objects	Information about a deleted output. Information about a maximum of 100 outputs can be transferred.

Table 11-4 ProductGroupInfo

Parameter	Mandatory	Type	Description
group_id	No	String	Template group ID.
products	No	Array of **ProductUrlInfo** objects	Product information.

Table 11-5 ProductUrlInfo

Parameter	Mandatory	Type	Description
url	No	String	Deleted product URL.

Response Parameters

Status code: 200

Table 11-6 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
status	String	**SUCCESS**: successful **FAIL**: failed **PARTIAL_SUCCESS**: partially successful
deleted_products	Array of **ProductGroupInfo** objects	Product that has been deleted.
failed_products	Array of **ProductGroupDelFailInfo** objects	Product that fails to be deleted.

Table 11-7 ProductGroupInfo

Parameter	Type	Description
group_id	String	Template group ID.
products	Array of **ProductUrlInfo** objects	Product information.

Table 11-8 ProductUrlInfo

Parameter	Type	Description
url	String	Deleted product URL.

Table 11-9 ProductGroupDelFailInfo

Parameter	Type	Description
group_id	String	Template group ID.
fail_reason	String	Cause of the failure to delete a template group.
products	Array of **ProductDelFailInfo** objects	Information about the product that fails to be deleted.

Table 11-10 ProductDelFailInfo

Parameter	Type	Description
url	String	Deleted product URL.
fail_reason	String	Cause of the failure to delete a product.

Status code: 403

Table 11-11 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

- ```
  DELETE https://{endpoint}/v1/{project_id}/asset/transcode-product

  Content-Type: application/json
  {
    "asset_id": "8c53b4038ea3d8694f1a8a7ac95f612f",
    "delete_type": "PRODUCT",
    "delete_infos": [
      {
        "group_id": "9171008cca7f47909c61f61a65d4c906",
        "products": [
          {
            "url": "https://103-cn-north-4.cdn-vod.huaweicloud.com/asset/
  8c53b4038ea3d8694f1a8a7ac95f612f/play_multi_video/
  9171008cca7f47909c61f61a65d4c906/38a92b74ead764ad21aa5355bad5193e.mp4"
          }
        ]
      }
    ]
  }
  ```
- ```
  DELETE https://{endpoint}/v1/{project_id}/asset/transcode-product

  Content-Type: application/json
  {
    "asset_id": "6f306b4cc58c39b2a97ad797367b88ad",
     "delete_type": "GROUP",
     "delete_infos": [
       {
         "group_id": "02e6887dd6bf48299922eec52d192115"
       }
     ]
  }
  ```

Example Responses

Status code: 200

The information is returned when the request succeeded.


```
{
  "status" : "SUCCESS",
  "deleted_products" : [ {
    "group_id" : "9171008cca7f47909c61f61a65d4c906",
    "products" : [ {
      "url" : "https://103-cn-north-4.cdn-vod.huaweicloud.com/asset/8c53b4038ea3d8694f1a8a7ac95f612f/
play_multi_video/9171008cca7f47909c61f61a65d4c906/da6613ac132eda54b2f4ab67adbf56bd.mp4"
    } ]
  } ],
  "failed_products" : [ ]
}
```

Status code: 403

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
403	The information is returned when the request failed.

Error Codes

See **Error Codes**.

12 Media file category

12.1 Creates a media asset category

Function

Creates a media asset category

Constraints

You can create up to three categories, each of which supports 128 subcategories.

URI

POST /v1.0/{project_id}/asset/category

Table 12-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 12-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 12-3 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Media asset category name, which contains a maximum of 64 characters
parent_id	No	Integer	Parent category ID If this parameter is not specified, a level-1 category is generated by default. The category ID of the root node is 0.

Response Parameters

Status code: 200

Table 12-4 Response body parameters

Parameter	Type	Description
name	String	Media asset category name

Parameter	Type	Description
parent_id	Integer	Parent category ID The parent ID of the level-1 category is 0.
id	Integer	Media asset category ID
level	Integer	Media file category level Possible values are: • 1: level-1 category • 2: level-2 category • 3: level-3 category
projectId	String	Project ID

Status code: 400

Table 12-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a media file category.

```
POST https://{endpoint}/v1.0/{project_id}/asset/category

Content-Type: application/json
{
  "name": "Movie"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "id" : 334,
  "name" : "Movie",
  "parent_id" : 0,
  "level" : 1,
  "projectId" : "58411d8df9064*****75d75b54e01358"
}
```

Status code: 400

The information is returned when the request fails.


```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

12.2 Modifies a media asset category

Function

Modifies a media asset category

URI

PUT /v1.0/{project_id}/asset/category

Table 12-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 12-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 12-8 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Media asset category name
id	Yes	Integer	Media asset category ID

Response Parameters

Status code: 200

Table 12-9 Response body parameters

Parameter	Type	Description
name	String	Media asset category name
parent_id	Integer	Parent category ID The parent ID of the level-1 category is 0.
id	Integer	Media asset category ID

Parameter	Type	Description
level	Integer	Media file category level Possible values are: • 1: level-1 category • 2: level-2 category • 3: level-3 category
projectId	String	Project ID

Status code: 400

Table 12-10 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modifies a media file category.

```
PUT https://{endpoint}/v1.0/{project_id}/asset/category

Content-Type: application/json
{
  "id": 334,
  "name": "film"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "id" : 334,
  "name" : "film",
  "parent_id" : 0,
  "level" : 1,
  "projectId" : "58411d8df90649a39b75d75b54e01358"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

12.3 Deletes a media asset category

Function

Deletes a media asset category

URI

DELETE /v1.0/{project_id}/asset/category

Table 12-11 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 12-12 Query Parameters

Parameter	Mandatory	Type	Description
id	Yes	Integer	Video category ID. Setting this parameter to **0** or **null** will delete all level-1 categories under the project ID.

Request Parameters

Table 12-13 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 12-14 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Deletes a media file category.

```
DELETE https://{endpoint}/v1.0/{project_id}/asset/category?id={id}
```

Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
204	A status code 204 No Content is returned if the request is successful.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

12.4 Queries a specified category

Function

Queries a specified category and its subcategories

URI

GET /v1.0/{project_id}/asset/category

Table 12-15 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 12-16 Query Parameters

Parameter	Mandatory	Type	Description
id	Yes	Integer	Video category ID. Setting this parameter to **0** or **null** will query all level-1 categories under the project ID.

Request Parameters

Table 12-17 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 12-18 Response body parameters

Parameter	Type	Description
[items]	Array of **QueryCategoryRsp** objects	Category return value

Table 12-19 QueryCategoryRsp

Parameter	Type	Description
id	String	Category ID
name	String	Category name
children	Array of **QueryCategoryRsp** objects	Subcategory list

Status code: 400

Table 12-20 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries a specified category.

```
GET https://{endpoint}/v1.0/{project_id}/asset/category?id={id}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
[ {
  "id" : "334",
  "name" : "Movie",
  "children" : [ {
    "id" : "335",
    "name" : "TV",
    "children" : [ ]
  } ]
} ]
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13 Media file management

13.1 Deletes a media asset

Function

Deletes a media asset

Constraints

Media assets that are being transcoded, reviewed, or captured cannot be deleted.

URI

DELETE /v1.0/{project_id}/asset

Table 13-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 13-2 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	Array of strings	Media file ID. Multiple media files can be deleted at a time. Use commas (,) to separate them during batch deletion.The value ranges from 1 to 10.

Parameter	Mandatory	Type	Description
delete_type	No	String	Deletion type. If this parameter is set to **origin**, only the source file is deleted and the transcoded file is retained. If this parameter is left empty, the entire media file is deleted by default.

Request Parameters

Table 13-3 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 13-4 Response body parameters

Parameter	Type	Description
delete_result_array	Array of **DeleteResult** objects	Result of the media asset deletion task

Table 13-5 DeleteResult

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
status	String	Status Possible values are: • DELETED: The task has been deleted. • FAILED: The deletion failed.

Status code: 403

Table 13-6 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Deletes a media file.

```
DELETE https://{endpoint}/v1.0/{project_id}/asset?asset_id={asset_id}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "delete_result_array" : [ {
    "asset_id" : "f488337c31c8e4622f1590735b134c65",
    "status" : "DELETED"
  } ]
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.2 Publishes the media asset

Function

Sets the status of the media asset to published. Media assets can be published in batches.

URI

POST /v1.0/{project_id}/asset/status/publish

Table 13-7 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 13-8 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 13-9 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	Array of strings	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Response Parameters

Status code: 200

Table 13-10 Response body parameters

Parameter	Type	Description
asset_info_array	Array of **AssetInfo** objects	Information about published media assets

Table 13-11 AssetInfo

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
is_multi_transcode	Boolean	Whether there are multiple transcoding modes.

Parameter	Type	Description
status	String	Media file status Possible values are: • UNCREATED: The media file ID does not exist. • DELETED: The task has been deleted. • CANCELLED: The upload has been canceled. • SERVER_ERROR: File upload failed due to a VOD server fault. • UPLOAD_FAILED: File upload to OBS failed. • CREATING: The task is being created. • PUBLISHED: The file has been published. • WAITING_TRANSCODE: The file is to be published (to be transcoded). • TRANSCODING: The file is to be published (being transcoded). • TRANSCODE_FAILED: The file is to be published (transcoding failed). • TRANSCODE_SUCCEED: The file is to be published (transcoding succeeded). • CREATED: The file is to be published (not transcoded). • NO_ASSET: The media asset does not exist. • DELETING: The file is being deleted. • DELETE_FAILED: The deletion failed. • OBS_CREATING: The task for dumping data to OBS is being created. • OBS_CREATE_FAILED: Data dump to OBS failed. • OBS_CREATE_SUCCESS: Data has been dumped to OBS.
description	String	Media file substatus or description • Describes the exception cause of an abnormal media file. • Describes the processing information of a normal media file.

Parameter	Type	Description
base_info	**BaseInfo** object	Basic media file information.
play_info_array	Array of **PlayInfo** objects	Playback information about the transcoded file • HLS or DASH: The number of members in this array is *n* + 1, where *n* indicates the number of channels in the output file. • MP4: The number of members in this array is *n*, which indicates the number of channels in the output file.

Table 13-12 BaseInfo

Parameter	Type	Description
title	String	Media file title The value is UTF-8-encoded and contains a maximum of 128 characters.
video_name	String	Media asset file name
description	String	Media file description The value contains a maximum of 1024 characters.
category_id	Long	Media asset category ID
category_name	String	Media asset category name
create_time	String	Time when the media file was created The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
last_modified	String	Time when the media file was last modified The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.

Parameter	Type	Description
video_type	String	Audio/Video file type. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2
tags	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.
video_url	String	URL for accessing the original video file
sign_url	String	Temporary access URL of the original video file in OBS. A specific value is returned only when the API for **Querying File Information** is called.
cover_info_array	Array of **CoverInfo** objects	Cover information
subtitle_info	Array of **SubtitleInfo** objects	Subtitle information array
source_path	**File_addr** object	Media asset storage parameters.
output_path	**File_addr** object	Media asset storage parameters.

Table 13-13 CoverInfo

Parameter	Type	Description
cover_url	String	URL for downloading the cover file

Table 13-14 SubtitleInfo

Parameter	Type	Description
url	String	URL for downloading the subtitle file
id	Integer	Subtitle file ID
type	String	Subtitle file type
language	String	Subtitle file language

Table 13-15 File_addr

Parameter	Type	Description
bucket	String	OBS bucket name
location	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	String	File path

Table 13-16 PlayInfo

Parameter	Type	Description
play_type	String	Playback protocol type Possible values are: • hls • dash • mp4
group_id	String	Transcoding group ID.
group_name	String	Transcoding group name.
url	String	Streaming URL
encrypted	Integer	Whether the stream is encrypted Possible values are: • 0: not encrypted • 1: encrypted Default value: **0**

Parameter	Type	Description
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.

Table 13-17 MetaData

Parameter	Type	Description
pack_type	String	Video container format Possible values are: • MP4 • TS • MOV • MXF • MPG • FLV • WMV • MP3 • WMA • APE • FLAC • AAC • AC3 • MMF • AMR • M4A • M4R • OGG • WAV • WV • MP2 • AVI • F4V • M4V • MPEG • HLS • DASH

Parameter	Type	Description
codec	String	Video encoding format. Options: • MPEG-2 • MPEG-4 • H.264 • H.265 • WMV • Vorbis • AAC • AC-3 • AMR • APE • FLAC • MP3 • MP2 • WMA • PCM • ADPCM • WavPack **NOTE** If **unknown** is returned for codec, the current audio/video encoding format sent by the user cannot be parsed.
duration	Long	Video duration, in second. If the original video duration is not an integer, the value of this field is rounded down. If the original video duration is shorter than 1 second, the value of this field is **1**.
duration_ms	Long	Video duration, in milliseconds.
video_size	Long	Video file size Unit: byte
width	Long	Video width, in pixels • Possible values for H.264: a multiple of 2 between 32 and 3840 • Possible values for H.265: a multiple of 4 between 320 and 3840

Parameter	Type	Description
hight	Long	Video height, in pixels. • Possible values for H.264: a multiple of 2 between 32 and 2,160 • Possible values for H.265: a multiple of 4 between 240 and 2,160
height	Long	Video height (unit: pixel)
bit_rate	Long	Average video bitrate
frame_rate	Long	Frame rate, in FPS Possible values are: • FRAMERATE_AUTO = 1, • FRAMERATE_10 = 2, • FRAMERATE_15 = 3, • FRAMERATE_2397 = 4, // 23.97 fps • FRAMERATE_24 = 5, • FRAMERATE_25 = 6, • FRAMERATE_2997 = 7, // 29.97 fps • FRAMERATE_30 = 8, • FRAMERATE_50 = 9, • FRAMERATE_60 = 10 The default value is **1**. Unit: FPS
quality	String	Definition Possible values are: • FULL_HD: ultra-high-definition (UHD) • HD: high definition (HD) • SD: standard definition (SD) • FLUENT: smooth • AD: adaptive • 2K • 4K
audio_channels	Integer	Number of audio channels

Status code: 403

Table 13-18 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Publishes a media file.

```
POST https://{endpoint}/v1.0/{project_id}/asset/status/publish

Content-Type: application/json
{
  "asset_id": [
    "f488337c31c8e4622f1590735b134c65",
    "07ba4d46463355e800c2f42f628f0631"
  ]
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_info_array" : [ {
    "asset_id" : "f488337c31c8e4622f1590735b134c65",
    "status" : "UNCREATED",
    "description" : "Asset meta download fail, errorCode is VOD.100021010,\nerrorMsg is Internal server
communication is abnormal",
    "base_info" : {
      "title" : "Avatar test test",
      "video_name" : "Avatar_480P.mp4",
      "description" : "Avatar, test",
      "category_id" : -1,
      "category_name" : "Other",
      "create_time" : "20180209024019",
      "last_modified" : "20180209024019",
      "video_type" : "MP4",
      "tags" : "mytags"
    }
  }, {
    "asset_id" : "07ba4d46463355e800c2f42f628f0631",
    "status" : " UNCREATED",
    "description" : "no asset"
  } ]
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.3 Unpublishes the media asset

Function

Sets the status of the media asset to unpublished

URI

POST /v1.0/{project_id}/asset/status/unpublish

Table 13-19 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 13-20 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 13-21 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	Array of strings	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Response Parameters

Status code: 200

Table 13-22 Response body parameters

Parameter	Type	Description
asset_info_array	Array of **AssetInfo** objects	Media file status set.

Table 13-23 AssetInfo

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
is_multi_transcode	Boolean	Whether there are multiple transcoding modes.

Parameter	Type	Description
status	String	Media file status Possible values are: • UNCREATED: The media file ID does not exist. • DELETED: The task has been deleted. • CANCELLED: The upload has been canceled. • SERVER_ERROR: File upload failed due to a VOD server fault. • UPLOAD_FAILED: File upload to OBS failed. • CREATING: The task is being created. • PUBLISHED: The file has been published. • WAITING_TRANSCODE: The file is to be published (to be transcoded). • TRANSCODING: The file is to be published (being transcoded). • TRANSCODE_FAILED: The file is to be published (transcoding failed). • TRANSCODE_SUCCEED: The file is to be published (transcoding succeeded). • CREATED: The file is to be published (not transcoded). • NO_ASSET: The media asset does not exist. • DELETING: The file is being deleted. • DELETE_FAILED: The deletion failed. • OBS_CREATING: The task for dumping data to OBS is being created. • OBS_CREATE_FAILED: Data dump to OBS failed. • OBS_CREATE_SUCCESS: Data has been dumped to OBS.
description	String	Media file substatus or description • Describes the exception cause of an abnormal media file. • Describes the processing information of a normal media file.

Parameter	Type	Description
base_info	**BaseInfo** object	Basic media file information.
play_info_array	Array of **PlayInfo** objects	Playback information about the transcoded file • HLS or DASH: The number of members in this array is *n* + 1, where *n* indicates the number of channels in the output file. • MP4: The number of members in this array is *n*, which indicates the number of channels in the output file.

Table 13-24 BaseInfo

Parameter	Type	Description
title	String	Media file title The value is UTF-8-encoded and contains a maximum of 128 characters.
video_name	String	Media asset file name
description	String	Media file description The value contains a maximum of 1024 characters.
category_id	Long	Media asset category ID
category_name	String	Media asset category name
create_time	String	Time when the media file was created The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
last_modified	String	Time when the media file was last modified The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.

Parameter	Type	Description
video_type	String	Audio/Video file type. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2
tags	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.
video_url	String	URL for accessing the original video file
sign_url	String	Temporary access URL of the original video file in OBS. A specific value is returned only when the API for **Querying File Information** is called.
cover_info_array	Array of **CoverInfo** objects	Cover information
subtitle_info	Array of **SubtitleInfo** objects	Subtitle information array
source_path	**File_addr** object	Media asset storage parameters.
output_path	**File_addr** object	Media asset storage parameters.

Table 13-25 CoverInfo

Parameter	Type	Description
cover_url	String	URL for downloading the cover file

Table 13-26 SubtitleInfo

Parameter	Type	Description
url	String	URL for downloading the subtitle file
id	Integer	Subtitle file ID
type	String	Subtitle file type
language	String	Subtitle file language

Table 13-27 File_addr

Parameter	Type	Description
bucket	String	OBS bucket name
location	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	String	File path

Table 13-28 PlayInfo

Parameter	Type	Description
play_type	String	Playback protocol type Possible values are: • hls • dash • mp4
group_id	String	Transcoding group ID.
group_name	String	Transcoding group name.
url	String	Streaming URL
encrypted	Integer	Whether the stream is encrypted Possible values are: • 0: not encrypted • 1: encrypted Default value: **0**

Parameter	Type	Description
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.

Table 13-29 MetaData

Parameter	Type	Description
pack_type	String	Video container format Possible values are: • MP4 • TS • MOV • MXF • MPG • FLV • WMV • MP3 • WMA • APE • FLAC • AAC • AC3 • MMF • AMR • M4A • M4R • OGG • WAV • WV • MP2 • AVI • F4V • M4V • MPEG • HLS • DASH

Parameter	Type	Description
codec	String	Video encoding format. Options: • MPEG-2 • MPEG-4 • H.264 • H.265 • WMV • Vorbis • AAC • AC-3 • AMR • APE • FLAC • MP3 • MP2 • WMA • PCM • ADPCM • WavPack **NOTE** If **unknown** is returned for codec, the current audio/video encoding format sent by the user cannot be parsed.
duration	Long	Video duration, in second. If the original video duration is not an integer, the value of this field is rounded down. If the original video duration is shorter than 1 second, the value of this field is **1**.
duration_ms	Long	Video duration, in milliseconds.
video_size	Long	Video file size Unit: byte
width	Long	Video width, in pixels • Possible values for H.264: a multiple of 2 between 32 and 3840 • Possible values for H.265: a multiple of 4 between 320 and 3840

Parameter	Type	Description
hight	Long	Video height, in pixels. • Possible values for H.264: a multiple of 2 between 32 and 2,160 • Possible values for H.265: a multiple of 4 between 240 and 2,160
height	Long	Video height (unit: pixel)
bit_rate	Long	Average video bitrate
frame_rate	Long	Frame rate, in FPS Possible values are: • FRAMERATE_AUTO = 1, • FRAMERATE_10 = 2, • FRAMERATE_15 = 3, • FRAMERATE_2397 = 4, // 23.97 fps • FRAMERATE_24 = 5, • FRAMERATE_25 = 6, • FRAMERATE_2997 = 7, // 29.97 fps • FRAMERATE_30 = 8, • FRAMERATE_50 = 9, • FRAMERATE_60 = 10 The default value is **1**. Unit: FPS
quality	String	Definition Possible values are: • FULL_HD: ultra-high-definition (UHD) • HD: high definition (HD) • SD: standard definition (SD) • FLUENT: smooth • AD: adaptive • 2K • 4K
audio_channels	Integer	Number of audio channels

Status code: 403

Table 13-30 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Unpublishes a media file.

```
POST https://{endpoint}/v1.0/{project_id}/asset/status/unpublish

Content-Type: application/json
{
  "asset_id": [
    "f488337c31c8e4622f1590735b134c65",
    "07ba4d46463355e800c2f42f628f0631"
  ]
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_info_array" : [ {
    "asset_id" : "f488337c31c8e4622f1590735b134c65",
    "status" : "UNCREATED",
    "description" : "Asset meta download fail, errorCode is VOD.100021010,\nerrorMsg is Internal server communication is abnormal",
    "base_info" : {
      "title" : "Avatar test test",
      "video_name" : "Avatar_480P.mp4",
      "description" : "Avatar, test",
      "category_id" : -1,
      "category_name" : "Other",
      "create_time" : "20180209024019",
      "last_modified" : "20180209024019",
      "video_type" : "MP4",
      "tags" : "mytags"
    }
  }, {
    "asset_id" : "07ba4d46463355e800c2f42f628f0631",
    "status" : "NOASSET",
    "description" : "no asset"
  } ]
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.4 Queries media asset information

Function

Queries media asset information by ID, category, status, start time and end time

URI

GET /v1.0/{project_id}/asset/info

Table 13-31 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 13-32 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	No	Array of strings	Media ID. A maximum of 10 media assets can be queried at a time.

Parameter	Mandatory	Type	Description
status	No	Array of strings	Media file status Possible values are: • UNCREATED: The media file ID does not exist. • DELETED: The task has been deleted. • CANCELLED: The upload has been canceled. • SERVER_ERROR: File upload failed due to a VOD server fault. • UPLOAD_FAILED: File upload to OBS failed. • CREATING: The task is being created. • PUBLISHED: The file has been published. • TRANSCODING: The file is to be published (being transcoded). • TRANSCODE_FAILED: The file is to be published (transcoding failed). • TRANSCODE_SUCCEED: The file is to be published (transcoding succeeded). • CREATED: The file is to be published (not transcoded).
transcodeStatus	No	Array of strings	Transcoding status Possible values are: • TRANSCODING: The file is being transcoded. • TRANSCODE_FAILED: Transcoding failed. • TRANSCODE_SUCCEED: Transcoding succeeded. • UN_TRANSCODE: The file is not transcoded. • WAITING_TRANSCODE: The file is to be transcoded.

Parameter	Mandatory	Type	Description
assetStatus	No	Array of strings	Media file status Possible values are: • PUBLISHED: The file has been published. • CREATED: The file has not been published.
start_time	No	String	Start time. This parameter is invalid when **asset_id** is specified. The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
end_time	No	String	End time. This parameter is invalid when **asset_id** is specified. The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
category_id	No	Integer	Category ID
tags	No	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
query_string	No	String	String for fuzzy search in the media asset title and description
page	No	Integer	Page number. This parameter is invalid when **asset_id** is specified. If this parameter is not specified or is set to null, data of the first page will be queried by default. The default value is **0**.

Parameter	Mandatory	Type	Description
size	No	Integer	Number of records on each page. This parameter is invalid when **asset_id** is specified. The value ranges from 1 to 100. Default value: **10**

Request Parameters

Table 13-33 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 13-34 Response body parameters

Parameter	Type	Description
asset_info_array	Array of **AssetInfo** objects	Media asset information list

Parameter	Type	Description
is_truncated	Integer	Whether the query results are truncated Possible values are: • 1: Not all query results are returned. • 0: All query results are returned.
total	Integer	Queries the total number of media files>Currently, statistics about up to 20,000 media files can be collected. To query the total number of media files, submit a service ticket.

Table 13-35 AssetInfo

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
is_multi_transcode	Boolean	Whether there are multiple transcoding modes.

Parameter	Type	Description
status	String	Media file status Possible values are: • UNCREATED: The media file ID does not exist. • DELETED: The task has been deleted. • CANCELLED: The upload has been canceled. • SERVER_ERROR: File upload failed due to a VOD server fault. • UPLOAD_FAILED: File upload to OBS failed. • CREATING: The task is being created. • PUBLISHED: The file has been published. • WAITING_TRANSCODE: The file is to be published (to be transcoded). • TRANSCODING: The file is to be published (being transcoded). • TRANSCODE_FAILED: The file is to be published (transcoding failed). • TRANSCODE_SUCCEED: The file is to be published (transcoding succeeded). • CREATED: The file is to be published (not transcoded). • NO_ASSET: The media asset does not exist. • DELETING: The file is being deleted. • DELETE_FAILED: The deletion failed. • OBS_CREATING: The task for dumping data to OBS is being created. • OBS_CREATE_FAILED: Data dump to OBS failed. • OBS_CREATE_SUCCESS: Data has been dumped to OBS.
description	String	Media file substatus or description • Describes the exception cause of an abnormal media file. • Describes the processing information of a normal media file.

Parameter	Type	Description
base_info	**BaseInfo** object	Basic media file information.
play_info_array	Array of **PlayInfo** objects	Playback information about the transcoded file • HLS or DASH: The number of members in this array is *n* + 1, where *n* indicates the number of channels in the output file. • MP4: The number of members in this array is *n*, which indicates the number of channels in the output file.

Table 13-36 BaseInfo

Parameter	Type	Description
title	String	Media file title The value is UTF-8-encoded and contains a maximum of 128 characters.
video_name	String	Media asset file name
description	String	Media file description The value contains a maximum of 1024 characters.
category_id	Long	Media asset category ID
category_name	String	Media asset category name
create_time	String	Time when the media file was created The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
last_modified	String	Time when the media file was last modified The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.

Parameter	Type	Description
video_type	String	Audio/Video file type. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2
tags	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.
video_url	String	URL for accessing the original video file
sign_url	String	Temporary access URL of the original video file in OBS. A specific value is returned only when the API for **Querying File Information** is called.
cover_info_array	Array of **CoverInfo** objects	Cover information
subtitle_info	Array of **SubtitleInfo** objects	Subtitle information array
source_path	**File_addr** object	Media asset storage parameters.
output_path	**File_addr** object	Media asset storage parameters.

Table 13-37 CoverInfo

Parameter	Type	Description
cover_url	String	URL for downloading the cover file

Table 13-38 SubtitleInfo

Parameter	Type	Description
url	String	URL for downloading the subtitle file
id	Integer	Subtitle file ID
type	String	Subtitle file type
language	String	Subtitle file language

Table 13-39 File_addr

Parameter	Type	Description
bucket	String	OBS bucket name
location	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	String	File path

Table 13-40 PlayInfo

Parameter	Type	Description
play_type	String	Playback protocol type Possible values are: • hls • dash • mp4
group_id	String	Transcoding group ID.
group_name	String	Transcoding group name.
url	String	Streaming URL
encrypted	Integer	Whether the stream is encrypted Possible values are: • 0: not encrypted • 1: encrypted Default value: **0**

Parameter	Type	Description
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.

Table 13-41 MetaData

Parameter	Type	Description
pack_type	String	Video container format Possible values are: • MP4 • TS • MOV • MXF • MPG • FLV • WMV • MP3 • WMA • APE • FLAC • AAC • AC3 • MMF • AMR • M4A • M4R • OGG • WAV • WV • MP2 • AVI • F4V • M4V • MPEG • HLS • DASH

Parameter	Type	Description
codec	String	Video encoding format. Options: • MPEG-2 • MPEG-4 • H.264 • H.265 • WMV • Vorbis • AAC • AC-3 • AMR • APE • FLAC • MP3 • MP2 • WMA • PCM • ADPCM • WavPack **NOTE** If **unknown** is returned for codec, the current audio/video encoding format sent by the user cannot be parsed.
duration	Long	Video duration, in second. If the original video duration is not an integer, the value of this field is rounded down. If the original video duration is shorter than 1 second, the value of this field is **1**.
duration_ms	Long	Video duration, in milliseconds.
video_size	Long	Video file size Unit: byte
width	Long	Video width, in pixels • Possible values for H.264: a multiple of 2 between 32 and 3840 • Possible values for H.265: a multiple of 4 between 320 and 3840

Parameter	Type	Description
hight	Long	Video height, in pixels. • Possible values for H.264: a multiple of 2 between 32 and 2,160 • Possible values for H.265: a multiple of 4 between 240 and 2,160
height	Long	Video height (unit: pixel)
bit_rate	Long	Average video bitrate
frame_rate	Long	Frame rate, in FPS Possible values are: • FRAMERATE_AUTO = 1, • FRAMERATE_10 = 2, • FRAMERATE_15 = 3, • FRAMERATE_2397 = 4, // 23.97 fps • FRAMERATE_24 = 5, • FRAMERATE_25 = 6, • FRAMERATE_2997 = 7, // 29.97 fps • FRAMERATE_30 = 8, • FRAMERATE_50 = 9, • FRAMERATE_60 = 10 The default value is **1**. Unit: FPS
quality	String	Definition Possible values are: • FULL_HD: ultra-high-definition (UHD) • HD: high definition (HD) • SD: standard definition (SD) • FLUENT: smooth • AD: adaptive • 2K • 4K
audio_channels	Integer	Number of audio channels

Status code: 400

Table 13-42 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

- Queries one media ID

  ```
  GET
  /v1.0/{project_id}/asset/info?asset_id=652c1e4085afeb22fdc256c6757d751b
  ```

- Queries multiple media IDs

  ```
  GET /v1.0/{project_id}/asset/info?
  asset_id=652c1e4085afeb22fdc256c6757d751b&asset_id=xxxxxx&asset_id=xxxxxx
  ```

- Performs query by time segment

  ```
  GET /v1.0/{project_id}/asset/info?
  start_time=20170725181000&end_time=20170726181000&page=0&size=20
  ```

- Queries the latest 10 media assets of the current tenant. Page defaults to 0 and size defaults to 10.

  ```
  GET /v1.0/{project_id}/asset/info
  ```

- Queries the latest 21st to 40th media assets of the current tenant

  ```
  GET /v1.0/{project_id}/asset/info?page=1&size=20
  ```

- Performs query by category

  ```
  GET /v1.0/{project_id}/asset/info?category_id=100
  ```

Example Responses

Status code: 200

The information is returned when the request succeeds.


```
{
  "asset_info_array" : [ {
    "asset_id" : "652c1e4085afeb22fdc256c6757d751b",
    "status" : "PUBLISHED",
    "description" : "Asset meta is published",
    "is_multi_transcode" : false,
    "base_info" : {
      "title" : "okFLV.flv",
      "video_name" : "okFLV.flv",
      "description" : null,
      "category_id" : -1,
      "category_name" : "Other",
      "create_time" : "20190704144303",
      "last_modified" : "20190704144303",
      "video_type" : "FLV",
      "tags" : null,
      "meta_data" : {
        "pack_type" : null,
        "codec" : "H.264",
        "duration" : 244,
        "duration_ms" : 244000,
        "video_size" : 13682041,
        "width" : 512,
        "hight" : 288,
        "height" : 288,
```



```
      "bit_rate" : 448,
      "frame_rate" : 30,
      "quality" : null
    },
    "video_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/
45c2493abe68de3dac7e98b0dadcf8ce.flv",
    "sign_url" : "",
    "cover_info_array" : [ {
      "cover_url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/cover/
Cover0.jpg"
    } ],
    "subtitle_info" : [ ],
    "source_path" : {
      "bucket" : "obs-host",
      "location" : "cn-north-4",
      "object" : "1/okFLV.flv"
    },
    "output_path" : {
      "bucket" : "obs-host",
      "location" : "cn-north-4",
      "object" : "output/652c1e4085afeb22fdc256c6757d751b/"
    }
  },
  "play_info_array" : [ {
    "play_type" : "hls",
    "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/play_video/
index.m3u8",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 0,
      "duration_ms" : 0,
      "video_size" : 0,
      "width" : 0,
      "hight" : 0,
      "height" : 0,
      "bit_rate" : 0,
      "frame_rate" : 0,
      "quality" : null
    }
  }, {
    "play_type" : "hls",
    "url" : "https://651.cdn-vod.huaweicloud.com/asset/652c1e4085afeb22fdc256c6757d751b/play_video/
b5d498082bbcff7a2055041d803ae2f9_1.m3u8",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 205,
      "duration_ms" : 205000,
      "video_size" : 14669824,
      "width" : 512,
      "hight" : 288,
      "height" : 288,
      "bit_rate" : 534,
      "frame_rate" : 0,
      "quality" : null
    }
  } ]
 } ],
 "is_truncated" : 0,
 "total" : 1
}
```


Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.5 Modifies media asset attributes

Function

Modifies media asset attributes

URI

PUT /v1.0/{project_id}/asset/info

Table 13-43 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 13-44 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 13-45 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
title	No	String	Media asset title. The value is UTF-8-encoded and contains a maximum of 128 characters.
description	No	String	Media asset description. The value contains a maximum of 1024 characters.
category_id	No	Integer	Media asset category ID
tags	No	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.

Response Parameters

Status code: 400

Table 13-46 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modifies a media file attribute.

```
PUT https://{endpoint}/v1.0/{project_id}/asset/info

Content-Type: application/json
{
  "asset_id": "f488337c31c8e4622f1590735b134c65",
  "title": "Avatar",
  "description": "Avatar, marketed as James Cameron's Avatar, is a 2009 American epic\nscience fiction film directed, written, produced, and co-edited by\nJames Cameron",
  "category_id": "1"
}
```

Example Responses

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.6 Queries details about a specified media asset

Function

Queries details about a specified media asset

URI

GET /v1.0/{project_id}/asset/details

Table 13-47 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 13-48 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
categories	No	Array of strings	Information type to be queried If this parameter is left blank, all information types are queried. If this parameter is specified, one or more information types can be queried at a time. Possible values are: • base_info: basic information of the media file • transcode_info: transcoding result • thumbnail_info: snapshot capturing result • review_info: review result

Request Parameters

Table 13-49 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 13-50 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
base_info	**BaseInfo** object	Basic media file information.
transcode_info	**TranscodeInfo** object	Information about the transcoded file **NOTE** Such information can be queried only after the transcoding succeeded, and is unavailable when the transcoding is not performed, is being performed, or failed.
thumbnail_info	**ThumbnailInfo** object	Snapshot information **NOTE** Such information can be queried only after the snapshot capturing succeeded, and is unavailable when the snapshot capturing is not performed, is being performed, or failed.

Parameter	Type	Description
review_info	**ReviewInfo** object	Review information array > Such information can be queried only after the review succeeded, and is unavailable when the review is not performed, is being performed, or failed.

Table 13-51 BaseInfo

Parameter	Type	Description
title	String	Media file title The value is UTF-8-encoded and contains a maximum of 128 characters.
video_name	String	Media asset file name
description	String	Media file description The value contains a maximum of 1024 characters.
category_id	Long	Media asset category ID
category_name	String	Media asset category name
create_time	String	Time when the media file was created The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
last_modified	String	Time when the media file was last modified The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
video_type	String	Audio/Video file type. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2

Parameter	Type	Description
tags	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.
video_url	String	URL for accessing the original video file
sign_url	String	Temporary access URL of the original video file in OBS. A specific value is returned only when the API for **Querying File Information** is called.
cover_info_array	Array of **CoverInfo** objects	Cover information
subtitle_info	Array of **SubtitleInfo** objects	Subtitle information array
source_path	**File_addr** object	Media asset storage parameters.
output_path	**File_addr** object	Media asset storage parameters.

Table 13-52 CoverInfo

Parameter	Type	Description
cover_url	String	URL for downloading the cover file

Table 13-53 SubtitleInfo

Parameter	Type	Description
url	String	URL for downloading the subtitle file
id	Integer	Subtitle file ID
type	String	Subtitle file type
language	String	Subtitle file language

Table 13-54 File_addr

Parameter	Type	Description
bucket	String	OBS bucket name
location	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	String	File path

Table 13-55 TranscodeInfo

Parameter	Type	Description
template_group_name	String	Name of a transcoding template group
output	Array of **Output** objects	Array in the output file • HLS or DASH: The number of members in this array is *n* + 1, where *n* indicates the number of channels in the output file. • MP4: The number of members in this array is *n*, which indicates the number of channels in the output file.
exec_desc	String	Execution description
transcode_status	String	Transcoding status Possible values are: • UN_TRANSCODE: The file is not transcoded. • WAITING_TRANSCODE: The file is to be transcoded. • TRANSCODING: The file is being transcoded. • TRANSCODE_SUCCEED: Transcoding succeeded. • TRANSCODE_FAILED: Transcoding failed.

Table 13-56 Output

Parameter	Type	Description
play_type	String	Protocol type Possible values are: • hls • dash • mp4
url	String	Streaming URL
group_id	String	Transcoding group ID.
group_name	String	Transcoding group name.
encrypted	Integer	Whether the stream is encrypted Possible values are: • 0: not encrypted • 1: encrypted
quality	String	Definition Possible values are: • FLUENT: smooth • SD: standard definition (SD) • HD: high definition (HD) • FULL_HD: ultra-high-definition (UHD)
meta_data	**MetaData** object	Video metadata It is generated after video parsing, including the packaging format, size, resolution, bitrate, and frame rate.

Table 13-57 MetaData

Parameter	Type	Description
pack_type	String	Video container format Possible values are: • MP4 • TS • MOV • MXF • MPG • FLV • WMV • MP3 • WMA • APE • FLAC • AAC • AC3 • MMF • AMR • M4A • M4R • OGG • WAV • WV • MP2 • AVI • F4V • M4V • MPEG • HLS • DASH

Parameter	Type	Description
codec	String	Video encoding format. Options: • MPEG-2 • MPEG-4 • H.264 • H.265 • WMV • Vorbis • AAC • AC-3 • AMR • APE • FLAC • MP3 • MP2 • WMA • PCM • ADPCM • WavPack **NOTE** If **unknown** is returned for codec, the current audio/video encoding format sent by the user cannot be parsed.
duration	Long	Video duration, in second. If the original video duration is not an integer, the value of this field is rounded down. If the original video duration is shorter than 1 second, the value of this field is **1**.
duration_ms	Long	Video duration, in milliseconds.
video_size	Long	Video file size Unit: byte
width	Long	Video width, in pixels • Possible values for H.264: a multiple of 2 between 32 and 3840 • Possible values for H.265: a multiple of 4 between 320 and 3840

Parameter	Type	Description
hight	Long	Video height, in pixels. • Possible values for H.264: a multiple of 2 between 32 and 2,160 • Possible values for H.265: a multiple of 4 between 240 and 2,160
height	Long	Video height (unit: pixel)
bit_rate	Long	Average video bitrate
frame_rate	Long	Frame rate, in FPS Possible values are: • FRAMERATE_AUTO = 1, • FRAMERATE_10 = 2, • FRAMERATE_15 = 3, • FRAMERATE_2397 = 4, // 23.97 fps • FRAMERATE_24 = 5, • FRAMERATE_25 = 6, • FRAMERATE_2997 = 7, // 29.97 fps • FRAMERATE_30 = 8, • FRAMERATE_50 = 9, • FRAMERATE_60 = 10 The default value is **1**. Unit: FPS
quality	String	Definition Possible values are: • FULL_HD: ultra-high-definition (UHD) • HD: high definition (HD) • SD: standard definition (SD) • FLUENT: smooth • AD: adaptive • 2K • 4K
audio_channels	Integer	Number of audio channels

Table 13-58 ThumbnailInfo

Parameter	Type	Description
sample	Array of **ThumbnailRsp** objects	Video snapshot information. Snapshots are captured by interval.
dots	Array of **ThumbnailRsp** objects	Video snapshot information. Snapshots are captured by time point.
quantity	Array of **ThumbnailRsp** objects	Video snapshot information. **Frequency** is set to **Specified quantity**.
exec_desc	String	Execution description
thumbnail_status	String	Snapshot status Possible values are: • UN_THUMBNAIL: No snapshot is captured. • THUMBNAILING: The snapshot is being captured. • THUMBNAIL_SUCCEED: The snapshot has been captured. • THUMBNAIL_FAILED: Snapshot capturing failed.

Table 13-59 ThumbnailRsp

Parameter	Type	Description
offset	Integer	Time offset of the snapshot in the video, in seconds
url	String	URL for accessing the snapshot

Table 13-60 ReviewInfo

Parameter	Type	Description
suggestion	String	Whether the file is approved Options: • block: The file contains sensitive information and fails the review. • pass: The file does not contain sensitive information and is approved. • review: The file needs to be manually reviewed. When multiple scenarios are reviewed at the same time, the value of **suggestion** is subject to the scenario where sensitive information is most likely to occur. That is, if the value of at least one scenario is **block**, the value of **suggestion** is **block**. If the values of all scenarios are **pass**, the value of **suggestion** is **pass**. If a scenario needs to be reviewed, the value of **suggestion** is **review**.
text	**TextReviewRet** object	Text review result
cover	Array of **PictureReviewRet** objects	Cover review result
video	Array of **PictureReviewRet** objects	Video review result
exec_desc	String	Execution description

Parameter	Type	Description
review_status	String	Review status Possible values are: • UN_REVIEW: The file has not been reviewed. • REVIEWING: The file is being reviewed. • REVIEW_SUSPICIOUS: The file needs to be manually reviewed. • REVIEW_PASSED: The file has been approved. • REVIEW_FAILED: The file is not approved. • REVIEW_BLOCKED: The file has been blocked.

Table 13-61 TextReviewRet

Parameter	Type	Description
suggestion	String	Whether the file is approved Possible values are: • block: The file contains sensitive information and fails the review. • pass: The file does not contain sensitive information and is approved. • review: The file needs to be manually reviewed.
politics	String	List of politically sensitive words
porn	String	List of pornographic words
abuse	String	List of abusive words

Table 13-62 PictureReviewRet

Parameter	Type	Description
suggestion	String	Whether the file is approved Possible values are: • block: The file contains sensitive information and fails the review. • pass: The file does not contain sensitive information and is approved. • review: The file needs to be manually reviewed.
offset	Integer	Time offset of the snapshot in the video. This field is not involved in the thumbnail. Unit: second
url	String	URL for accessing the snapshot or cover
politics	Array of **ReviewDetail** objects	Review result of potential politically sensitive information
terrorism	Array of **ReviewDetail** objects	Review result of potential terrorism-related information
porn	Array of **ReviewDetail** objects	Review result of potential pornographic information

Table 13-63 ReviewDetail

Parameter	Type	Description
confidence	String	Confidence score The value ranges from 0 to 1.
label	String	Label description of each review result • politics: The label indicates the information about a political figure. • terrorism: The label indicates the terrorism-related information, such as guns, knives, and fire. • porn: The label indicates the pornographic information.

Status code: 400

Table 13-64 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries details about a specified media file.

```
GET https://{endpoint}/v1.0/{project_id}/asset/details?asset_id={asset_id}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "41cff636d5b12a51e7eb2838bbf45201",
  "base_info" : {
    "title" : "Avatar",
    "video_name" : "Video on Demand (VOD).mp4",
    "description" : "Avatar, marketed as James Cameron's Avatar, is a 2009 American\nepic science fiction film directed, written, produced, and\nco-edited by James Cameron",
    "category_id" : -1,
    "category_name" : "Other",
    "create_time" : "20190612032250",
    "last_modified" : "20190613075030",
    "video_type" : "MP4",
    "tags" : "test1,test2",
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 131,
      "duration_ms" : 131000,
      "video_size" : 4942645,
      "width" : 1280,
      "hight" : 720,
      "height" : 720,
      "bit_rate" : 173,
      "frame_rate" : 30
    },
    "video_url" : "https://355.cdn-vod.huaweicloud.com/asset/41cff636d5b12a51e7eb2838bbf45201/5597e59de70722eaeb9b18c274e249b2.mp4",
    "sign_url" : "https://vod-bucket-57-cn-north-4.obs.cn-north-4.myhuaweicloud.com:443/05ab5cef408026f22f62c018de60cf2e/41cff636d5b12a51e7eb2838bbf45201 /5597e59de70722eaeb9b18c274e249b2.mp4?AWSAccessKeyId=CBN2J**********0RCSN&Expires=1652499973&Signature=kZYh0hEos2V**********AHGyXA%3D",
    "cover_info_array" : [ {
      "cover_url" : "https://355.cdn-vod.huaweicloud.com/asset/41cff636d5b12a51e7eb2838bbf45201/cover/Cover0.jpg"
    } ]
  },
  "transcode_info" : {
    "template_group_name" : "system_template_group",
    "output" : [ {
      "play_type" : "HLS",
```



```
    "url" : "https://355.cdn-vod.huaweicloud.com/asset/41cff636d5b12a51e7eb2838bbf45201/play_video/
index.m3u8",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 0,
      "duration_ms" : 0,
      "video_size" : 0,
      "width" : 0,
      "hight" : 0,
      "height" : 0,
      "bit_rate" : 0,
      "frame_rate" : 0
    }
  }, {
    "play_type" : "HLS",
    "url" : "https://355.cdn-vod.huaweicloud.com/asset/41cff636d5b12a51e7eb2838bbf45201/play_video/
VODVOD video _1_854X480_600_0.m3u8",
    "encrypted" : 0,
    "quality" : "SD",
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 130,
      "duration_ms" : 130000,
      "video_size" : 7976960,
      "width" : 854,
      "hight" : 480,
      "height" : 480,
      "bit_rate" : 421,
      "frame_rate" : 30000,
      "quality" : "SD"
    }
  }, {
    "play_type" : "DASH",
    "url" : "https://355.cdn-vod.huaweicloud.com/asset/41cff636d5b12a51e7eb2838bbf45201/play_video/
index.mpd",
    "encrypted" : 0,
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 0,
      "duration_ms" : 0,
      "video_size" : 0,
      "width" : 0,
      "hight" : 0,
      "height" : 0,
      "bit_rate" : 0,
      "frame_rate" : 0
    }
  }, {
    "play_type" : "DASH",
    "encrypted" : 0,
    "quality" : "SD",
    "meta_data" : {
      "pack_type" : null,
      "codec" : "H.264",
      "duration" : 130,
      "duration_ms" : 130000,
      "video_size" : 7976960,
      "width" : 854,
      "hight" : 480,
      "height" : 480,
      "bit_rate" : 421,
      "frame_rate" : 30000,
      "quality" : "SD"
    }
  } ],
```



```
    "exec_desc" : "Transcode success",
    "transcode_status" : "TRANSCODE_SUCCEED"
  }
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

13.7 Queries media assets

Function

Queries media assets. Each record in the list contains the brief information of a media asset.

URI

GET /v1.0/{project_id}/asset/list

Table 13-65 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 13-66 Query Parameters

Parameter	Mandatory	Type	Description
asset_id	No	Array of strings	Media ID. A maximum of 10 media assets can be queried at a time.
status	No	Array of strings	Media file status. Media files of different statuses are queried at a time. Possible values are: • CREATING: The file is being uploaded. • FAILED: Upload failed. • CREATED: The file has been uploaded. • PUBLISHED: The file has been published. • TRANSCODING: The file is being transcoded. • TRANSCODE_SUCCEED: Transcoding succeeded. • TRANSCODE_FAILED: Transcoding failed. • THUMBNAILING: The snapshot is being captured. • THUMBNAIL_SUCCEED: The snapshot has been captured. • THUMBNAIL_FAILED: Snapshot capturing failed. • UN_REVIEW: The file has not been reviewed. • REVIEWING: The file is being reviewed. • REVIEW_SUSPICIOUS: The file failed the review and is to be manually reviewed. • REVIEW_PASSED: The file has been approved. • REVIEW_FAILED: The file failed the review. • REVIEW_BLOCKED: The file has been blocked.

Parameter	Mandatory	Type	Description
start_time	No	String	Start time. It is a timezone-agnostic UTC time (yyyymmddhhmmss).
end_time	No	String	End time. It is a timezone-agnostic UTC time (yyyymmddhhmmss).
category_id	No	Integer	Category ID
tags	No	String	Media asset tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
query_string	No	String	String for fuzzy search in the media asset title, description, and category
media_type	No	Array of strings	Audio/Video file format. Files of up to 20 different formats can be queried. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2
page	No	Integer	Page number Default value: **0**
size	No	Integer	Number of records on each page The value ranges from 1 to 100. Default value: **10**

Request Parameters

Table 13-67 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 13-68 Response body parameters

Parameter	Type	Description
total	Integer	Total number of media files > Currently, statistics about up to 20,000 media files can be collected. To query the total number of media files, submit a service ticket.
assets	Array of **AssetSummary** objects	Media asset list

Table 13-69 AssetSummary

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Parameter	Type	Description
title	String	Media asset title. The value is UTF-8-encoded and contains a maximum of 128 characters.
description	String	Media asset description. The value contains a maximum of 1024 characters.
duration	Integer	Media file duration Unit: second
duration_ms	Long	Video duration, in milliseconds.
size	Long	Media file size Unit: byte
original_url	String	Original streaming URL
category	String	Media asset category name
covers	Array of **CoverInfo** objects	Cover information
create_time	String	Time when the media file was created The format is yyyymmddhhmmss. The value must be a UTC time irrelevant to the time zone.
asset_status	String	Media file status Possible values are: • CREATING: The file is being uploaded. • FAILED: Upload failed. • CREATED: The file has been uploaded. • PUBLISHED: The file has been published. • DELETED: The task has been deleted.

Parameter	Type	Description
transcode_status	String	Transcoding status Possible values are: • UN_TRANSCODE: The file is not transcoded. • WAITING_TRANSCODE: The file is to be transcoded. • TRANSCODING: The file is being transcoded. • TRANSCODE_SUCCEED: Transcoding succeeded. • TRANSCODE_FAILED: Transcoding failed.
thumbnail_status	String	Snapshot status Possible values are: • UN_THUMBNAIL: No snapshot is captured. • THUMBNAILING: The snapshot is being captured. • THUMBNAIL_SUCCEED: The snapshot has been captured. • THUMBNAIL_FAILED: Snapshot capturing failed.
review_status	String	Content review status Possible values are: • UN_REVIEW: The file has not been reviewed. • REVIEWING: The file is being reviewed. - REVIEW_SUSPICIOUS: The file needs to be manually reviewed. • REVIEW_PASSED: The file has been approved. • REVIEW_FAILED: The file is not approved. • REVIEW_BLOCKED: The file has been blocked.

Parameter	Type	Description
exec_desc	String	Execution description of media file tasks Example: • asset_exec_desc: upload success: execution description of file upload tasks • transcode_exec_desc: transcode success: execution description of transcoding tasks • thumbnail_exec_desc: thumbnail failed: execution description of snapshot capturing tasks • review_exec_desc: review pass: execution description of review tasks
media_type	String	Audio/Video file format. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2

Table 13-70 CoverInfo

Parameter	Type	Description
cover_url	String	URL for downloading the cover file

Status code: 400

Table 13-71 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries media files.


```
GET https://{endpoint}/v1.0/{project_id}/asset/list
```

Example Responses

Status code: 200

The information is returned when the request succeeds.


```
{
  "total" : 1,
  "assets" : [ {
    "asset_id" : "67d1470893419bfcb9663103dd8a66ac",
    "title" : "video.mp4",
    "duration" : 60,
    "duration_ms" : 60000,
    "size" : 12881945,
    "category" : "Other",
    "covers" : [ {
      "cover_url" : "https://355.cdn-vod.huaweicloud.com/asset/67d1470893419bfcb9663103dd8a66ac/cover/
Cover0.jpg"
    } ],
    "create_time" : "20190625020756",
    "asset_status" : "PUBLISHED",
    "transcode_status" : "TRANSCODE_SUCCEED",
    "thumbnail_status" : "UN_THUMBNAIL",
    "review_status" : "UN_REVIEW",
    "exec_desc" : "asset_exec_desc:Asset meta is\npublished;transcode_exec_desc:Transcode success;",
    "media_type" : "MP4"
  } ]
}
```


Status code: 400

The information is returned when the request fails.


```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

14 Transcoding template management

14.1 Customizes a transcoding template

Function

Customizes a transcoding template

URI

POST /v2/{project_id}/asset/template/transcodings

Table 14-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 14-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 14-3 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Template group name.
is_default	No	Boolean	Indicates whether the template is the default transcoding template. The default value is **false**. If the value is not specified, the default value is used.
is_auto_encrypt	No	Boolean	Whether to encrypt the output audio/video. The default value is **false** (not encrypted). Only HLS audio/video streams can be encrypted before being output. Original audio/video streams are not encrypted. This field is unavailable for non-HLS transcoded audio/video. Before enabling this function, obtain the key URL by referring to section "HLS Encryption Settings" in *VOD User Guide*.
quality_info_list	Yes	Array of **QualityInfoList** objects	Image quality configuration list.
common	Yes	**CommonInfo** object	Low-bitrate HD transcoding switch, encoding format, and multi-stream common parameters.
watermark_template_ids	No	Array of strings	ID array of the bound watermark template group.
description	No	String	Template description.

Table 14-4 QualityInfoList

Parameter	Mandatory	Type	Description
video	No	**VideoInfo** object	Video template information. Configure at least one of the **video** and **audio** parameters.
audio	No	**AudioInfo** object	Audio template information

Table 14-5 VideoInfo

Parameter	Mandatory	Type	Description
quality	Yes	String	Video quality. • The default 4K resolution is 3840 x 2160 and the bitrate is 8000 kbit/s. • The default 2K resolution is 2560 x 1440 and the bitrate is 7000 kbit/s. • The default FULL_HD resolution is 1920 x 1080 and the bitrate is 3000 kbit/s. • The default HD resolution is 1280 x 720 and the bitrate is 1000 kbit/s. • The default SD resolution is 854 x 480 and the bitrate is 600 kbit/s. • The default FLUENT resolution is 480 x 270 and the bitrate is 300 kbit/s.
width	No	Integer	Video width H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 4,096. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 4,096.

Parameter	Mandatory	Type	Description
height	No	Integer	Video height H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 2,880. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 2,880.
bitrate	Yes	Integer	Average bitrate, in kbit/s. The value can be **0** or an integer between 40 and 30,000.
frame_rate	Yes	Integer	Frame rate, in frames per second (FPS). The value is an integer between 0 and 75. If the value is lower than 5 or higher than 60, the frame rate is adaptive.

Table 14-6 AudioInfo

Parameter	Mandatory	Type	Description
sample_rate	Yes	String	Audio sampling rate. Possible options: • AUDIO_SAMPLE_AUTO (default) • AUDIO_SAMPLE_22050: 22050 Hz • AUDIO_SAMPLE_32000: 32000 Hz • AUDIO_SAMPLE_44100: 44100 Hz • AUDIO_SAMPLE_48000: 48000 Hz • AUDIO_SAMPLE_96000: 96000 Hz
bitrate	No	Integer	Audio bitrate, in kbit/s.

Parameter	Mandatory	Type	Description
channels	Yes	String	Number of audio channels. Options: • **AUDIO_CHANNELS_1**: mono • **AUDIO_CHANNELS_2**: stereo (default) • **AUDIO_CHANNELS_5_1**: 5.1 audio channel

Table 14-7 CommonInfo

Parameter	Mandatory	Type	Description
pvc	No	Boolean	Low-bitrate HD transcoding switch.
video_codec	No	String	Video encoding format • **H264** • **H265**
audio_codec	No	String	Audio encoding format. • AAC: AAC (default) • HEAAC1: HEAAC1 • HEAAC2: HEAAC2 • MP3: MP3
is_black_cut	No	Boolean	Black bar cropping type. The default value is **false**, indicating that black bar cropping is disabled.
format	Yes	String	Transcoded file format • **MP4** • **HLS** • **DASH** • **DASH_HLS** • **MP3** • **ADTS** • **UNKNOW**
hls_interval	No	Integer	Segment file duration. The value ranges from 2 to 10 (in seconds) and defaults to **5**.

Parameter	Mandatory	Type	Description
upsample	No	Boolean	Whether to enable upsampling. After this function is enabled, video resolution can be improved and the number of sampling points can be increased. The default value is **false**, indicating that upsampling is disabled.

Parameter	Mandatory	Type	Description
adaptation	No	String	Resolution adaptation mode of the transcoded media file. Options: • **SHORT**: adaptive short side The width and height of the transcoding template are specified. The long side of the source media file is scaled based on the ratio between the short side of the transcoding template and the short side of the source media file. • **LONG**: adaptive long side The width and height of the transcoding template are specified. The short side of the source media file is scaled based on the ratio between the long side of the transcoding template and the long side of the source media file. • **NONE** (default): adaptive to the specified aspect ratio The output resolution is based on the width and height set in the transcoding template. **NOTE** • This parameter cannot be set for the V1 API (discarded). The default value is **SHORT**. The default value is **NONE** for the V2 API. The default value of the V1 API is different from that of the V2 API. Pay attention to the value of this parameter when calling the API. • The default value of the transcoding template created using the V2 API on the console is **SHORT**, which is the same as that of the transcoding template created using the V1 API.

Parameter	Mandatory	Type	Description
preset	No	Integer	Encoding quality level. Options: **0**: default mode; **1**: transcoding efficiency first; **2**: transcoding quality first.
max_iframes_interval	No	Integer	Maximum I-frame interval. Value range: [2, 10]. The default value is **5**, in second.
hls_audio_separate	No	Boolean	Whether to store the transcoded audio separately.
hls_segment_type	No	String	Packaging format of HLS segments. Currently, TS (default) and fMP4 are supported.

Response Parameters

Status code: 201

Table 14-8 Response body parameters

Parameter	Type	Description
group_id	String	Template group ID.

Status code: 400

Table 14-9 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a transcoding template.

```
POST https://{endpoint}/v2/{project_id}/asset/template/transcodings

Content-Type: application/json
{
  "name": "trans_template_test",
  "is_default": true,
  "is_auto_encrypt": false,
  "quality_info_list": [
```



```
  {
    "video": {
      "width": 1280,
      "height": 720,
      "bitrate": 1000,
      "quality": "HD",
      "frame_rate": 0
    },
    "audio": {
      "sample_rate": "AUDIO_SAMPLE_AUTO",
      "channels": "AUDIO_CHANNELS_1",
      "bitrate": 0
    }
  }
 ],
 "watermark_template_ids": [],
 "common": {
   "pvc": false,
   "video_codec": "H264",
   "audio_codec": "AAC",
   "format": "HLS",
   "hls_interval": 5
 }
}
```

Example Responses

Status code: 201

The information is returned when the request succeeded.

```
{
  "group_id" : "f9b045e0811c482f9de0d436a5927bb6"
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
201	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

14.2 Queries transcoding templates

Function

Queries transcoding templates

URI

GET /v2/{project_id}/asset/template/transcodings

Table 14-10 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 14-11 Query Parameters

Parameter	Mandatory	Type	Description
group_id	No	String	Template ID
is_default	No	Boolean	Whether a default template is used
offset	No	Integer	Offset. The default value is **0**. This parameter is invalid when **group_id** is specified.
limit	No	Integer	Number of records on each page. The value defaults to **10** and ranges from 1 to 100. This parameter is invalid when **group_id** is specified.
query_string	No	String	Fuzzy search by template name and description. This parameter is invalid when **group_id** is specified.

Request Parameters

Table 14-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 14-13 Response body parameters

Parameter	Type	Description
template_group_list	Array of **TransTemplateRsp** objects	Template group information.
total	Integer	Total number of records.

Table 14-14 TransTemplateRsp

Parameter	Type	Description
group_id	String	Template group ID.
name	String	Template group name.
is_default	Boolean	Whether to use the default transcoding template.
type	String	Template group type.

Parameter	Type	Description
is_auto_encrypt	Boolean	Whether to encrypt the output audio/video. The default value is **false** (not encrypted). Only HLS audio/video streams can be encrypted before being output. Original audio/video streams are not encrypted. This field is unavailable for non-HLS transcoded audio/video. Before enabling this function, obtain the key URL by referring to section "HLS Encryption Settings" in *VOD User Guide*.
quality_info_list	Array of **QualityInfoList** objects	Image quality configuration list.
watermark_template_ids	Array of strings	ID array of the bound watermark template group.
description	String	Template description.
common	**CommonInfo** object	Low-bitrate HD transcoding switch, low-bitrate HD transcoding version, and multi-stream common parameters.

Table 14-15 QualityInfoList

Parameter	Type	Description
video	**VideoInfo** object	Video template information. Configure at least one of the **video** and **audio** parameters.
audio	**AudioInfo** object	Audio template information

Table 14-16 VideoInfo

Parameter	Type	Description
quality	String	Video quality. ● The default 4K resolution is 3840 x 2160 and the bitrate is 8000 kbit/s. ● The default 2K resolution is 2560 x 1440 and the bitrate is 7000 kbit/s. ● The default FULL_HD resolution is 1920 x 1080 and the bitrate is 3000 kbit/s. ● The default HD resolution is 1280 x 720 and the bitrate is 1000 kbit/s. ● The default SD resolution is 854 x 480 and the bitrate is 600 kbit/s. ● The default FLUENT resolution is 480 x 270 and the bitrate is 300 kbit/s.
width	Integer	Video width H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 4,096. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 4,096.
height	Integer	Video height H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 2,880. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 2,880.
bitrate	Integer	Average bitrate, in kbit/s. The value can be **0** or an integer between 40 and 30,000.
frame_rate	Integer	Frame rate, in frames per second (FPS). The value is an integer between 0 and 75. If the value is lower than 5 or higher than 60, the frame rate is adaptive.

Table 14-17 AudioInfo

Parameter	Type	Description
sample_rate	String	Audio sampling rate. Possible options: • AUDIO_SAMPLE_AUTO (default) • AUDIO_SAMPLE_22050: 22050 Hz • AUDIO_SAMPLE_32000: 32000 Hz • AUDIO_SAMPLE_44100: 44100 Hz • AUDIO_SAMPLE_48000: 48000 Hz • AUDIO_SAMPLE_96000: 96000 Hz
bitrate	Integer	Audio bitrate, in kbit/s.
channels	String	Number of audio channels. Options: • **AUDIO_CHANNELS_1**: mono • **AUDIO_CHANNELS_2**: stereo (default) • **AUDIO_CHANNELS_5_1**: 5.1 audio channel

Table 14-18 CommonInfo

Parameter	Type	Description
pvc	Boolean	Low-bitrate HD transcoding switch.
video_codec	String	Video encoding format • **H264** • **H265**
audio_codec	String	Audio encoding format. • AAC: AAC (default) • HEAAC1: HEAAC1 • HEAAC2: HEAAC2 • MP3: MP3
is_black_cut	Boolean	Black bar cropping type. The default value is **false**, indicating that black bar cropping is disabled.

Parameter	Type	Description
format	String	Transcoded file format • **MP4** • **HLS** • **DASH** • **DASH_HLS** • **MP3** • **ADTS** • **UNKNOW**
hls_interval	Integer	Segment file duration. The value ranges from 2 to 10 (in seconds) and defaults to **5**.
upsample	Boolean	Whether to enable upsampling. After this function is enabled, video resolution can be improved and the number of sampling points can be increased. The default value is **false**, indicating that upsampling is disabled.

Parameter	Type	Description
adaptation	String	Resolution adaptation mode of the transcoded media file. Options: • **SHORT**: adaptive short side The width and height of the transcoding template are specified. The long side of the source media file is scaled based on the ratio between the short side of the transcoding template and the short side of the source media file. • **LONG**: adaptive long side The width and height of the transcoding template are specified. The short side of the source media file is scaled based on the ratio between the long side of the transcoding template and the long side of the source media file. • **NONE** (default): adaptive to the specified aspect ratio The output resolution is based on the width and height set in the transcoding template. **NOTE** • This parameter cannot be set for the V1 API (discarded). The default value is **SHORT**. The default value is **NONE** for the V2 API. The default value of the V1 API is different from that of the V2 API. Pay attention to the value of this parameter when calling the API. • The default value of the transcoding template created using the V2 API on the console is **SHORT**, which is the same as that of the transcoding template created using the V1 API.
preset	Integer	Encoding quality level. Options: **0**: default mode; **1**: transcoding efficiency first; **2**: transcoding quality first.
max_iframes_interval	Integer	Maximum I-frame interval. Value range: [2, 10]. The default value is **5**, in second.
hls_audio_separate	Boolean	Whether to store the transcoded audio separately.

Parameter	Type	Description
hls_segment_type	String	Packaging format of HLS segments. Currently, TS (default) and fMP4 are supported.

Status code: 403

Table 14-19 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries transcoding templates.

```
GET https://{endpoint}/v2/{project_id}/asset/template/transcodings?group_id={group_id}
```

Example Responses

Status code: 200

The information is returned when the request succeeded.

```
{
  "template_group_list" : [ {
    "group_id" : "9751249d25f14587b212544d6fd8dcf8",
    "name" : "non_transcoding_template_group",
    "is_default" : false,
    "type" : "non_transcoding_template_group",
    "is_auto_encrypt" : false,
    "quality_info_list" : [ {
      "video" : {
        "quality" : "UNKNOW",
        "width" : 0,
        "height" : 0,
        "bitrate" : 0,
        "frame_rate" : 0
      },
      "audio" : null
    } ],
    "watermark_template_ids" : null,
    "description" : null,
    "common" : {
      "pvc" : false,
      "is_black_cut" : false,
      "format" : "UNKNOW",
      "upsample" : false,
      "adaptation" : "NONE",
      "video_codec" : null,
      "audio_codec" : "AAC",
      "hls_interval" : 0
    }
  } ],
```



```
  "total" : 0
}
```

Status code: 403

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
403	The information is returned when the request failed.

Error Codes

See **Error Codes**.

14.3 Modifies a transcoding template

Function

Modifies a transcoding template

URI

PUT /v2/{project_id}/asset/template/transcodings

Table 14-20 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 14-21 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 14-22 Request body parameters

Parameter	Mandatory	Type	Description
group_id	Yes	String	Template group ID.
name	No	String	Template group name.
is_default	No	Boolean	Indicates whether the template is the default transcoding template. The default value is **false**. If the value is not specified, the default value is used.
is_auto_encrypt	No	Boolean	Whether to encrypt the output audio/video. The default value is **false** (not encrypted). Only HLS audio/video streams can be encrypted before being output. Original audio/video streams are not encrypted. This field is unavailable for non-HLS transcoded audio/video. Before enabling this function, obtain the key URL by referring to section "HLS Encryption Settings" in *VOD User Guide*.

Parameter	Mandatory	Type	Description
quality_info_list	No	Array of **QualityInfoList** objects	Image quality configuration list. If this parameter is not carried or is left empty, the data of this part in the template will not be updated.
watermark_template_ids	No	Array of strings	ID array of the bound watermark template group.
description	No	String	Template description.
common	No	**CommonInfo** object	Low-bitrate HD transcoding switch, encoding format, and multi-stream common parameters.

Table 14-23 QualityInfoList

Parameter	Mandatory	Type	Description
video	No	**VideoInfo** object	Video template information. Configure at least one of the **video** and **audio** parameters.
audio	No	**AudioInfo** object	Audio template information

Table 14-24 VideoInfo

Parameter	Mandatory	Type	Description
quality	Yes	String	Video quality. • The default 4K resolution is 3840 x 2160 and the bitrate is 8000 kbit/s. • The default 2K resolution is 2560 x 1440 and the bitrate is 7000 kbit/s. • The default FULL_HD resolution is 1920 x 1080 and the bitrate is 3000 kbit/s. • The default HD resolution is 1280 x 720 and the bitrate is 1000 kbit/s. • The default SD resolution is 854 x 480 and the bitrate is 600 kbit/s. • The default FLUENT resolution is 480 x 270 and the bitrate is 300 kbit/s.
width	No	Integer	Video width H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 4,096. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 4,096.
height	No	Integer	Video height H.264 encoding: The value can be **0** or an integer multiple of 2 between 32 and 2,880. H.265 encoding: The value can be **0** or an integer multiple of 2 between 160 and 2,880.
bitrate	Yes	Integer	Average bitrate, in kbit/s. The value can be **0** or an integer between 40 and 30,000.

Parameter	Mandatory	Type	Description
frame_rate	Yes	Integer	Frame rate, in frames per second (FPS). The value is an integer between 0 and 75. If the value is lower than 5 or higher than 60, the frame rate is adaptive.

Table 14-25 AudioInfo

Parameter	Mandatory	Type	Description
sample_rate	Yes	String	Audio sampling rate. Possible options: • AUDIO_SAMPLE_AUTO (default) • AUDIO_SAMPLE_22050: 22050 Hz • AUDIO_SAMPLE_32000: 32000 Hz • AUDIO_SAMPLE_44100: 44100 Hz • AUDIO_SAMPLE_48000: 48000 Hz • AUDIO_SAMPLE_96000: 96000 Hz
bitrate	No	Integer	Audio bitrate, in kbit/s.
channels	Yes	String	Number of audio channels. Options: • **AUDIO_CHANNELS_1**: mono • **AUDIO_CHANNELS_2**: stereo (default) • **AUDIO_CHANNELS_5_1**: 5.1 audio channel

Table 14-26 CommonInfo

Parameter	Mandatory	Type	Description
pvc	No	Boolean	Low-bitrate HD transcoding switch.

Parameter	Mandatory	Type	Description
video_codec	No	String	Video encoding format • **H264** • **H265**
audio_codec	No	String	Audio encoding format. • AAC: AAC (default) • HEAAC1: HEAAC1 • HEAAC2: HEAAC2 • MP3: MP3
is_black_cut	No	Boolean	Black bar cropping type. The default value is **false**, indicating that black bar cropping is disabled.
format	Yes	String	Transcoded file format • **MP4** • **HLS** • **DASH** • **DASH_HLS** • **MP3** • **ADTS** • **UNKNOW**
hls_interval	No	Integer	Segment file duration. The value ranges from 2 to 10 (in seconds) and defaults to **5**.
upsample	No	Boolean	Whether to enable upsampling. After this function is enabled, video resolution can be improved and the number of sampling points can be increased. The default value is **false**, indicating that upsampling is disabled.

Parameter	Mandatory	Type	Description
adaptation	No	String	Resolution adaptation mode of the transcoded media file. Options: • **SHORT**: adaptive short side The width and height of the transcoding template are specified. The long side of the source media file is scaled based on the ratio between the short side of the transcoding template and the short side of the source media file. • **LONG**: adaptive long side The width and height of the transcoding template are specified. The short side of the source media file is scaled based on the ratio between the long side of the transcoding template and the long side of the source media file. • **NONE** (default): adaptive to the specified aspect ratio The output resolution is based on the width and height set in the transcoding template. **NOTE** • This parameter cannot be set for the V1 API (discarded). The default value is **SHORT**. The default value is **NONE** for the V2 API. The default value of the V1 API is different from that of the V2 API. Pay attention to the value of this parameter when calling the API. • The default value of the transcoding template created using the V2 API on the console is **SHORT**, which is the same as that of the transcoding template created using the V1 API.

Parameter	Mandatory	Type	Description
preset	No	Integer	Encoding quality level. Options: **0**: default mode; **1**: transcoding efficiency first; **2**: transcoding quality first.
max_iframes_interval	No	Integer	Maximum I-frame interval. Value range: [2, 10]. The default value is **5**, in second.
hls_audio_separate	No	Boolean	Whether to store the transcoded audio separately.
hls_segment_type	No	String	Packaging format of HLS segments. Currently, TS (default) and fMP4 are supported.

Response Parameters

Status code: 400

Table 14-27 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modifies a transcoding template.

```
PUT https://{endpoint}/v2/{project_id}/asset/template/transcodings

Content-Type: application/json
{
  "group_id": "f9b045e0811c482f9de0d436a5927bb6",
  "name": "trans_template_test",
  "is_default": true,
  "quality_info_list": [
    {
      "video": {
        "width": 1280,
        "height": 720,
        "bitrate": 1000,
        "quality": "HD",
        "frame_rate": 0
      },
      "audio": {
        "sample_rate": "AUDIO_SAMPLE_AUTO",
        "channels": "AUDIO_CHANNELS_1",
        "bitrate": 0
      }
```

```
    }
  ],
  "watermark_template_ids": [],
  "common": {
    "pvc": false,
    "video_codec": "H264",
    "audio_codec": "AAC",
    "format": "HLS",
    "hls_interval": 5
  }
}
```

Example Responses

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

14.4 Deletes a custom template

Function

Deletes a custom template

URI

DELETE /v2/{project_id}/asset/template/transcodings

Table 14-28 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 14-29 Query Parameters

Parameter	Mandatory	Type	Description
group_id	Yes	String	Template ID

Request Parameters

Table 14-30 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 14-31 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Deletes a transcoding template.

```
DELETE https://{endpoint}/v2/{project_id}/asset/template/transcodings?group_id={group_id}
```

Example Responses

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
204	The status code **204 No Content** is returned if the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

15 Uploads media files

15.1 Uploads media files to VOD

Function

When calling this API to create a media file, you need to upload the corresponding media file to the OBS bucket of VOD.If the media file to be uploaded is less than 20 MB, you can use the PUT method to upload the media file to the URL returned by the API. For details, see **Example 1: Uploading a Media File Less Than 20 MB**.If the media file to be uploaded is greater than 20 MB, it must be split into binary streams before being uploaded. For details about how to use this API, see **Example 2: Uploading a Media File More Than 20 MB by Part**.

URI

POST /v1.0/{project_id}/asset

Table 15-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 15-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 15-3 Request body parameters

Parameter	Mandatory	Type	Description
title	Yes	String	Media asset title. The value is UTF-8-encoded and contains a maximum of 128 characters.
description	No	String	Video description. The value contains a maximum of 1024 characters.
video_name	Yes	String	Audio/Video file name. The value contains a maximum of 128 characters. The file name extension is optional.

Parameter	Mandatory	Type	Description
video_type	Yes	String	Uploaded audio/video file format. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, and WebM • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2 If an audio file is uploaded, transcoding, watermarking, and subtitling are not supported.
category_id	No	Integer	Media file category ID You can call the API for **Creating a Media File Category** to create a media file category or create a media file category in the category settings on the VOD console and obtain the category ID. **NOTE** If this parameter is not specified or is set to **-1**, the uploaded audio/video files fall into the preconfigured **Other** category.
video_md5	No	String	You are advised to refer to the example of uploading and updating media files in Generating an MD5 Value in the appendix of *API Reference*.

Parameter	Mandatory	Type	Description
cover_type	No	String	Thumbnail image format Possible values are: • JPG • PNG The name of the uploaded thumbnail is fixed, and the extension is the abbreviation of the thumbnail file format, for example, cover0.jpg and cover1.png. If the file format is not specified, the thumbnail file does not have a file name extension. **NOTE** If the image format is specified, the first-frame snapshot will not be used as the thumbnail. You need to upload a thumbnail.
cover_md5	No	String	MD5 value of a thumbnail fileYou are advised to refer to the example of uploading and updating media files in **Generating an MD5 Value** in the appendix of *API Reference*.
subtitles	No	Array of **Subtitle** objects	Subtitle file information
tags	No	String	Video tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
auto_publish	No	Integer	Whether to publish the content automatically. The options are as follows: • **0**: The content is not automatically published. • **1**: The content is automatically published. Default value: **1**

Parameter	Mandatory	Type	Description
template_group_name	No	String	Name of a transcoding template group If this parameter is specified, the specified transcoding template is used to transcode the uploaded audio/video. You can configure a transcoding template on the VOD console. For details, see Transcoding Settings in *VOD User Guide*. **NOTE** If both **template_group_name** and **workflow_name** are specified, **template_group_name** takes effect.
auto_encrypt	No	Integer	Whether to automatically encrypt a file Possible values are: • 0: not encrypted • 1: encrypted Default value: **0** A file must be encrypted and transcoded at the same time. When encryption is required, the transcoding parameter cannot be empty and the output file must be in HLS format.
auto_preheat	No	String	Whether to automatically pre-load content to the CDN Possible values are: • 0: The content is not automatically pre-loaded. • 1: The content is automatically pre-loaded. Default value: **0**
thumbnail	No	**Thumbnail** object	Snapshot parameters

Parameter	Mandatory	Type	Description
review	No	**Review** object	Media asset review parameter. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
workflow_name	No	String	Workflow nameIf this parameter is specified, the specified workflow is used to transcode the uploaded audio/video. You can configure a workflow on the VOD console. For details, see Transcoding Settings in *VOD User Guide*.

Table 15-4 Subtitle

Parameter	Mandatory	Type	Description
id	Yes	Integer	Subtitle ID. The value ranges from **1** to **16**.
type	Yes	String	Subtitle file format. Currently, only SRT and VTT are supported.
language	Yes	String	Subtitle language.
name	No	String	Subtitle file name.
md5	No	String	MD5 value of the subtitle file
description	No	String	Subtitle description

Table 15-5 Thumbnail

Parameter	Mandatory	Type	Description
type	Yes	String	Snapshot capturing mode. The options are as follows: • **time**: Snapshots are captured by interval. • **dots**: Snapshots are captured at a specified time point. • **quantity**: Snapshots are captured based on the specified quantity and video duration.
quantity	No	Integer	This parameter is mandatory when **type** is set to **quantity**. Snapshots are captured based on the specified quantity and video duration. Value range: an integer between 1 and 10
quantity_time	No	Integer	This parameter is optional when **type** is set to **quantity**. Snapshots are captured based on the specified quantity at a specified interval. Value range: an integer between 0 and 2,147,483,647
time	No	Integer	Interval for sampling, in seconds. **type** is set to **time**. Default value: **12** Value range: an integer between 0 and 100
dots	No	Array of integers	This parameter is mandatory when type is set to dots. The array of time points when a snapshot is captured is used.
cover_position	No	Integer	The value indicates which snapshot is specified as the thumbnail. The default value is **1**.

Parameter	Mandatory	Type	Description
format	No	Integer	Snapshot file format Possible values are: • 1: jpg The default value is **1**.
aspect_ratio	No	Integer	Aspect ratio Possible values are: • 0: adaptive (the original aspect ratio is retained) • 1: 16:9 Default value: **0**
max_length	No	Integer	The longest side of a snapshot. Unit: pixel The width of the snapshot is scaled proportionally with the longest side and input video pixel. Default value: **480**

Table 15-6 Review

Parameter	Mandatory	Type	Description
template_id	No	String	Review template ID. You can obtain the value after configuring the review template on the VOD console. For details, see Review Settings in *VOD User Guide*. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
interval	No	Integer	Snapshot check interval. The value range is (0,100]. This parameter is ignored in request parameters.

Parameter	Mandatory	Type	Description
politics	No	Integer	Confidence of politically sensitive content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
terrorism	No	Integer	Confidence of terrorism-related content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
porn	No	Integer	Confidence of pornographic content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Response Parameters

Status code: 200

Table 15-7 Response body parameters

Parameter	Type	Description
asset_id	String	Media ID
video_upload_url	String	URL for uploading videos
cover_upload_url	String	URL for uploading covers
subtitle_upload_urls	Array of strings	Array in the URL for uploading subtitles
target	**File_addr** object	Media asset storage parameters.

Table 15-8 File_addr

Parameter	Type	Description
bucket	String	OBS bucket name
location	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	String	File path

Status code: 403

Table 15-9 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a media file.

```
POST https://{endpoint}/v1/{project_id}/asset

Content-Type: application/json
{
  "title": "Avatar test test",
  "description": "Avatar, test",
  "category_id": -1,
  "tags": "mytags",
  "video_name": "Avatar_480P.mp4",
  "video_type": "MP4",
  "video_md5": "a945d4b3d8fc317190a9332fe856f03d",
```



```
  "cover_type": "JPG",
  "cover_md5": "a655d4b3d8fc758691a9332fe387f26c",
  "auto_publish": 0,
  "subtitles": [
    {
      "id": 1,
      "language": "CN",
      "type": "SRT",
      "md5": "SqcyFjJZoDZaP8oKIY6rgQ==",
      "description": "AAAAA"
    }
  ]
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "f488337c31c8e4622f1590735b134c65",
  "video_upload_url" : "https://obs.cn-north-4.myhuaweicloud.com:443/obs-vod-1/%7Bproject_id%7D/
f488337c31c8e4622f1590735b134c65/Avatar_480P.mp4?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1518147618&Signature=kZYh0hEos2V**********AHGyXA
%3D",
  "cover_upload_url" : "https://obs.cn-north-4.myhuaweicloud.com:443/obs-vod-1/%7Bproject_id%7D/
f488337c31c8e4622f1590735b134c65/cover/Cover0.jpg?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1518147619&Signature=kZYh0hEos2V**********AHGyXA
%3D",
  "subtitle_upload_urls" : [ "https://obs-vod-1.obs.cn-
north-4.myhuaweicloud.com:443/14ce1d4437164aba8b364ce15866154e/
53a018d2dc53ca07eb5a07a839205c9d/subtitle/1.srt?
AWSAccessKeyId=CBN2J**********0RCSN&Expires=1534760131&Signature=kZYh0hEos2V**********AHGyXA
%3D" ],
  "target" : {
    "bucket" : "obs-vod-1",
    "location" : "cn-north-4",
    "object" : "093bb6b6c4fc460ab90a40d8b821dda3/
a2053aef99725711dad3e02dc6cd5f89/0a9b70035b78b8a19c6d9e7c2693d93c.mp4"
  }
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10064",
  "error_msg" : "Media asset classification does not exist, please check."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.2 Obtains authorization for multipart upload

Function

When a client requests for creating a media file larger than 20 MB, the media file needs to be uploaded to OBS by part. The client needs to obtain the authorization using this API each time before uploading a file part to OBS.

This API is used to obtain a temporarily authorized URL for initializing a multipart upload, uploading parts, merging parts, listing uploaded parts, and canceling part merging. You need to configure the HTTP request method, request header, and request body by following the OBS API document. Then you can request for the corresponding temporarily authorized URL.

The multipart upload method is the same as that in the OBS API reference, including the HTTP request method, request header, and request body. This API is used to generate a URL with authentication information (**sign_str**) to replace the URL in the OBS API, so that the user has the temporary permission for uploading files to the bucket of VOD.

When calling the API for obtaining authorization, input **bucket**, **object_key**, and **http_verb**. **bucket** and **object_key** are obtained from the target field in the response body returned by the API for **Uploading a Media Asset to VOD**. **http_verb** varies depending on the specified operation.

URI

GET /v1.1/{project_id}/asset/authority

Table 15-10 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 15-11 Query Parameters

Parameter	Mandatory	Type	Description
http_verb	Yes	String	HTTP method for calling the OBS API for multipart upload. For details about the suitable HTTP method, see the OBS API reference. • Initializing an upload task: POST • Uploading a part: PUT • Merging parts: POST • Canceling a part: DELETE • Listing uploaded parts: GET
bucket	Yes	String	Bucket name Value of bucket obtained in the target field in the response body returned by the API for **Uploading Media Files to VOD**.
object_key	Yes	String	Object name Value of object obtained in the target field in the response body returned by the API for **Uploading Media Files to VOD**.
content_type	No	String	Content type corresponding to the file type. This parameter is mandatory for upload task initialization. For details about how to set parameters, see **Uploading a Media File Greater Than 20 MB by Part**. • Video file: video/*Video format*, for example, video/mp4 • Audio file: audio/*Audio format*, for example, audio/mp3 • Image file: image/*Image format*, for example, image/png • Subtitle file: application/octet-stream

Parameter	Mandatory	Type	Description
content_md5	No	String	MD5 value of each uploaded part
upload_id	No	String	ID of each upload task, which is returned after OBS initializes the multipart upload task. This field is mandatory except in the upload task initialization scenario.
part_number	No	Integer	ID of each uploaded part Value range: 1–10,000

Request Parameters

Table 15-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 15-13 Response body parameters

Parameter	Type	Description
sign_str	String	Signed URL. For details about the API calling example, see **Example 2: Uploading a Media File More Than 20 MB by Part**.Example: https://{obs_domain}/{bucket}?AWSAccessKeyId={AccessKeyID}&Expires={ExpiresValue}&Signature={Signature}

Status code: 403

Table 15-14 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

- Initializes an upload task

  ```
  GET https://{endpoint}/v1.1/{project_id}/asset/authority?http_verb=POST&content_type={type}&bucket={bucket}&object_key={objectKey}
  ```

- Uploads a part

  ```
  GET https://{endpoint}/v1.1/{project_id}/asset/authority?http_verb=PUT&content_md5={md5}&part_number={num}&upload_id={id}&bucket={bucket}&object_key={objectKey}
  ```

- Merges parts

  ```
  GET https://{endpoint}/v1.1/{project_id}/asset/authority?http_verb=POST&upload_id={id}&bucket={bucket}&object_key={objectKey}
  ```

- Cancels a part

  ```
  GET https://{endpoint}/v1.1/{project_id}/asset/authority?http_verb=DELETE&bucket={bucket}&object_key={objectKey}&upload_id={uploadId}
  ```

- Lists uploaded parts

  ```
  GET https://{endpoint}/v1.1/{project_id}/asset/authority?http_verb=GET&bucket={bucket}&object_key={objectKey}&upload_id={uploadId}
  ```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "sign_str" : "https://obs.cn-north-4.myhuaweicloud.com:443/obs-vod-1/%7Bproject_id%7D/f488337c31c8e4622f1590735b134c65/Avatar_480P.mp4?AWSAccessKeyId=CBN2J**********0RCSN&Expires=1518147618&Signature=kZYh0hEos2V**********AHGyXA
```



```
%3D"
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.3 Confirms media asset upload

Function

Notifies VOD of the media asset upload status after the media file is uploaded by part, indicating that the media asset has been uploaded.

URI

POST /v1.0/{project_id}/asset/status/uploaded

Table 15-15 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 15-16 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 15-17 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
status	Yes	String	Status Possible values are: • CREATED: The file has been created. • FAILED: File creation failed. • CANCELLED: The creation task has been canceled.

Response Parameters

Status code: 200

Table 15-18 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Status code: 403

Table 15-19 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Notifies VOD of the media file upload status.

```
POST https://{endpoint}/v1.0/{project_id}/asset/status/uploaded

Content-Type: application/json
{
  "asset_id": "f488337c31c8e4622f1590735b134c65",
  "status": "CREATED"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "f488337c31c8e4622f1590735b134c65"
}
```

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10062",
  "error_msg" : "Media asset or resource does not exist, please check."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.

Status Code	Description
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.4 Authorizing Access to an OBS Bucket

Function

Authorizes VOD to access an OBS bucket or cancels the authorization

URI

PUT /v1.0/{project_id}/asset/authority

Table 15-20 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 15-21 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.

Parameter	Mandatory	Type	Description
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 15-22 Request body parameters

Parameter	Mandatory	Type	Description
bucket	Yes	String	OBS bucket name
operation	Yes	String	Whether to authorize access to buckets Possible values are: • 0: The authorization is canceled. • 1: Access is authorized.

Response Parameters

Status code: 403

Table 15-23 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Grants or cancels the permission of VOD on OBS buckets.

```
PUT https://{endpoint}/v1.0/{project_id}/asset/authority

Content-Type: application/json
{
  "bucket": "bucket",
  "operation": "1"
}
```

Example Responses

Status code: 403

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
```



```
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
403	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.5 Dumps media assets to OBS

Function

If you have stored audio/video files in an OBS bucket before using VOD, you can call this API to dump the files to VOD and manage them on VOD.

Before using this API, perform the following operations:

- Authorize the user to use VOD. For details, see section "Creating a User and Granting VOD Permissions" in the *VOD User Guide*.
- Authorize the user to use OBS. For details, see section "Creating an IAM User and Granting OBS Permissions" in the *OBS User Guide*.
- Before calling this API, you need to call the API for **Authorizing Access to OBS** to authorize VOD to access the OBS bucket where the audio/video files are stored.

Constraints

Cloud services in different regions cannot be interconnected. As a result, the OBS bucket where audio/video files are stored and the VOD service must be in the same region. For example, audio/video files stored in OBS buckets of CN North-Beijing4 can be dumped only to VOD of CN North-Beijing4.

URI

POST /v1.0/{project_id}/asset/reproduction

Table 15-24 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 15-25 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 15-26 Request body parameters

Parameter	Mandatory	Type	Description
video_type	Yes	String	Dumped audio/video file format. The options are as follows: • Video files: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, HLS, RMVB and WebM • Audio files: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2 If an audio file is uploaded, transcoding, watermarking, and subtitling are not supported. **NOTE** When **video_type** is set to HLS, **storage_mode** must be set to storage in a tenant bucket, and the output path and input path must be in the same directory.
title	Yes	String	Media asset title. The value is UTF-8-encoded and contains a maximum of 128 characters.
description	No	String	Video description. The value contains a maximum of 1024 characters.
category_id	No	Integer	Media file category ID You can call the API for **Creating a Media File Category** to create a media file category, or create a media file category in the category settings on the VOD console and obtain the category ID. **NOTE** If this parameter is not specified or is set to **-1**, the uploaded audio/video files fall into the preconfigured **Other** category.

Parameter	Mandatory	Type	Description
tags	No	String	Video tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
auto_publish	No	Integer	Whether to publish the content automatically. The options are as follows: ● **0**: The content is not automatically published. ● **1**: The content is automatically published. Default value: **1**
template_group_name	No	String	Name of a transcoding template group If this parameter is specified, the specified transcoding template is used to transcode the uploaded audio/video. You can configure a transcoding template on the VOD console. For details, see Transcoding Settings in *VOD User Guide*. **NOTE** If both **template_group_name** and **workflow_name** are specified, **template_group_name** takes effect.
auto_encrypt	No	Integer	Whether to automatically encrypt a file Possible values are: ● 0: not encrypted ● 1: encrypted Default value: **0** If encryption is required, a transcoding template must be configured and the transcoded file must be in HLS format.

Parameter	Mandatory	Type	Description
auto_preheat	No	Integer	Whether to automatically pre-load content to the CDN Possible values are: • 0: The content is not automatically pre-loaded. • 1: The content is automatically pre-loaded. Default value: **0**
thumbnail	No	**Thumbnail** object	Snapshot parameters
review	No	**Review** object	Media asset review parameter. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
workflow_name	No	String	Workflow nameIf this parameter is specified, the specified workflow is used to transcode the uploaded audio/video. You can configure a workflow on the VOD console. For details, see Transcoding Settings in *VOD User Guide*.
input	Yes	**File_addr** object	Media asset storage parameters.
storage_mode	No	Integer	Storage mode Possible values are: • 0: Videos are copied to a VOD bucket. • 1: Videos are stored in a tenant bucket. Default value: **0**
output_bucket	No	String	Output bucket name. This parameter is mandatory when **storage_mode** is set to 1.
output_path	No	String	Output path name. This parameter is mandatory when **storage_mode** is set to 1.

Table 15-27 Thumbnail

Parameter	Mandatory	Type	Description
type	Yes	String	Snapshot capturing mode. The options are as follows: • **time**: Snapshots are captured by interval. • **dots**: Snapshots are captured at a specified time point. • **quantity**: Snapshots are captured based on the specified quantity and video duration.
quantity	No	Integer	This parameter is mandatory when **type** is set to **quantity**. Snapshots are captured based on the specified quantity and video duration. Value range: an integer between 1 and 10
quantity_time	No	Integer	This parameter is optional when **type** is set to **quantity**. Snapshots are captured based on the specified quantity at a specified interval. Value range: an integer between 0 and 2,147,483,647
time	No	Integer	Interval for sampling, in seconds. **type** is set to **time**. Default value: **12** Value range: an integer between 0 and 100
dots	No	Array of integers	This parameter is mandatory when type is set to dots. The array of time points when a snapshot is captured is used.
cover_position	No	Integer	The value indicates which snapshot is specified as the thumbnail. The default value is **1**.

Parameter	Mandatory	Type	Description
format	No	Integer	Snapshot file format Possible values are: • 1: jpg The default value is **1**.
aspect_ratio	No	Integer	Aspect ratio Possible values are: • 0: adaptive (the original aspect ratio is retained) • 1: 16:9 Default value: **0**
max_length	No	Integer	The longest side of a snapshot. Unit: pixel The width of the snapshot is scaled proportionally with the longest side and input video pixel. Default value: **480**

Table 15-28 Review

Parameter	Mandatory	Type	Description
template_id	No	String	Review template ID. You can obtain the value after configuring the review template on the VOD console. For details, see Review Settings in *VOD User Guide*. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
interval	No	Integer	Snapshot check interval. The value range is (0,100]. This parameter is ignored in request parameters.

Parameter	Mandatory	Type	Description
politics	No	Integer	Confidence of politically sensitive content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
terrorism	No	Integer	Confidence of terrorism-related content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
porn	No	Integer	Confidence of pornographic content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Table 15-29 File_addr

Parameter	Mandatory	Type	Description
bucket	Yes	String	OBS bucket name
location	Yes	String	Name of the region where the bucket is located. For example, the region name of CN North-Beijing4 is cn-north-4. The created bucket must be in the region of the VOD service.
object	Yes	String	File path

Response Parameters

Status code: 200

Table 15-30 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Status code: 400

Table 15-31 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

- Dumps HLS videos stored in OBS buckets to VOD (TS files and M3U8 files must be in the same directory).

```
POST https://{endpoint}/v1.0/{project_id}/asset/reproduction

Content-Type: application/json
{
  "video_type": "HLS",
  "title": "tittle",
  "auto_publish": 1,
  "input": {
    "bucket": "bucket_name",
    "location": "cn-north-4",
    "object": "test/hls/test_video.m3u8"
```



```
  },
  "storage_mode": 1,
  "output_bucket": "bucket_name",
  "output_path": "test/hls"
}
```

- Dumps media files in OBS buckets to VOD.

```
POST https://{endpoint}/v1.0/{project_id}/asset/reproduction

Content-Type: application/json
{
  "input": {
    "bucket": "bucket",
    "location": "cn-north-4",
    "object": "path"
  },
  "title": "title",
  "description": "des",
  "category_id": -1,
  "tags": "test",
  "video_type": "MP4",
  "auto_publish": 1,
  "template_group_name": "tempName"
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "asset_id" : "f488337c31c8e4622f1590735b134c65"
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10003",
  "error_msg" : "The specified key does not exist."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.6 Pulls media files from URLs

Function

Pulls a source media file using its URL to VOD. This method is applicable to a file of less than 5 TB. If the file size exceeds 5 TB, you need to use the migration tool to upload the file to VOD. Submit a service ticket to obtain the tool and operation guide.

Constraints

- A maximum of 16 audio/video clips can be pulled at a time by calling this API.
- When a URL is used to pull an M3U8 file, the URL cannot contain authentication parameters such as hotlink protection.

URI

POST /v1.0/{project_id}/asset/upload_by_url

Table 15-32 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 15-33 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 15-34 Request body parameters

Parameter	Mandatory	Type	Description
upload_metadatas	Yes	Array of **UploadMetaDataByUrl** objects	Metadata of a media asset to be created

Table 15-35 UploadMetaDataByUrl

Parameter	Mandatory	Type	Description
video_type	Yes	String	Uploaded audio/video file format. Options: • Video: MP4, TS, MOV, MXF, MPG, FLV, WMV, AVI, M4V, F4V, MPEG, 3GP, ASF, MKV, WebM, and M3U8 • Audio: MP3, OGG, WAV, WMA, APE, FLAC, AAC, AC3, MMF, AMR, M4A, M4R, WV, and MP2 If an audio file is uploaded, transcoding, watermarking, and subtitling are not supported.
title	Yes	String	Media asset title. The value is UTF-8-encoded and contains a maximum of 128 characters.
url	Yes	String	URL of the audio or video file to be pulled. **NOTE** The URL must end with a file name extension. Currently, only HTTP and HTTPS are supported. HTTP has security risks. HTTPS is recommended. You are advised not to write sensitive information such as authentication credentials in URLs.
description	No	String	Video description. The value contains a maximum of 1024 characters.

Parameter	Mandatory	Type	Description
category_id	No	Integer	Media file category ID You can call the API for **Creating a Media File Category** to create a media file category, or create a media file category in the category settings on the VOD console and obtain the category ID. **NOTE** If this parameter is not specified or is set to **-1**, the uploaded audio/video files fall into the preconfigured **Other** category.
tags	No	String	Video tag. Each tag contains up to 24 characters and up to 16 tags are allowed. Use commas (,) to separate tags, which are UTF-8-encoded.
auto_publish	No	Integer	Whether to publish the content automatically. The options are as follows: • **0**: The content is not automatically published. • **1**: The content is automatically published. Default value: **1**
template_group_name	No	String	Name of a transcoding template group If this parameter is specified, the specified transcoding template is used to transcode the uploaded audio/video. You can configure a transcoding template on the VOD console. For details, see Transcoding Settings in *VOD User Guide*. **NOTE** If both **template_group_name** and **workflow_name** are specified, **template_group_name** takes effect.

Parameter	Mandatory	Type	Description
auto_encrypt	No	Integer	Whether to automatically encrypt a file Possible values are: • 0: not encrypted • 1: encrypted Default value: **0**. If encryption is required, a transcoding template must be configured and the transcoded file must be in HLS format.
auto_preheat	No	Integer	Whether to automatically pre-load content to the CDN Possible values are: • 0: The content is not automatically pre-loaded. • 1: The content is automatically pre-loaded. Default value: **0**
thumbnail	No	**Thumbnail** object	Snapshot parameters
review	No	**Review** object	Media asset review parameter. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
workflow_name	No	String	Workflow name If this parameter is specified, the specified workflow is used to transcode the uploaded audio/video. You can configure a workflow on the VOD console. For details, see Transcoding Settings in *VOD User Guide*.

Table 15-36 Thumbnail

Parameter	Mandatory	Type	Description
type	Yes	String	Snapshot capturing mode. The options are as follows: • **time**: Snapshots are captured by interval. • **dots**: Snapshots are captured at a specified time point. • **quantity**: Snapshots are captured based on the specified quantity and video duration.
quantity	No	Integer	This parameter is mandatory when **type** is set to **quantity**. Snapshots are captured based on the specified quantity and video duration. Value range: an integer between 1 and 10
quantity_time	No	Integer	This parameter is optional when **type** is set to **quantity**. Snapshots are captured based on the specified quantity at a specified interval. Value range: an integer between 0 and 2,147,483,647
time	No	Integer	Interval for sampling, in seconds. **type** is set to **time**. Default value: **12** Value range: an integer between 0 and 100
dots	No	Array of integers	This parameter is mandatory when type is set to dots. The array of time points when a snapshot is captured is used.
cover_position	No	Integer	The value indicates which snapshot is specified as the thumbnail. The default value is **1**.

Parameter	Mandatory	Type	Description
format	No	Integer	Snapshot file format Possible values are: • 1: jpg The default value is **1**.
aspect_ratio	No	Integer	Aspect ratio Possible values are: • 0: adaptive (the original aspect ratio is retained) • 1: 16:9 Default value: **0**
max_length	No	Integer	The longest side of a snapshot. Unit: pixel The width of the snapshot is scaled proportionally with the longest side and input video pixel. Default value: **480**

Table 15-37 Review

Parameter	Mandatory	Type	Description
template_id	No	String	Review template ID. You can obtain the value after configuring the review template on the VOD console. For details, see Review Settings in *VOD User Guide*. **NOTE** [Only VOD in CN North-Beijing1 and CN North-Beijing4 supports this function.] (tag:hws) [Only VOD in AP-Singapore supports this function.] (tag:hws_hk)
interval	No	Integer	Snapshot check interval. The value range is (0,100]. This parameter is ignored in request parameters.

Parameter	Mandatory	Type	Description
politics	No	Integer	Confidence of politically sensitive content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
terrorism	No	Integer	Confidence of terrorism-related content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.
porn	No	Integer	Confidence of pornographic content moderation. The value can be **–1** or range from 0 to 100. This parameter is ignored in request parameters. A higher confidence level indicates a more reliable moderation result. If this function is disabled or the value is set to **0**, this check is not performed. **politics**, **terrorism**, and **porn** cannot be set to **0** simultaneously.

Response Parameters

Status code: 200

Table 15-38 Response body parameters

Parameter	Type	Description
upload_assets	Array of **UploadAsset** objects	Metadata of a media asset to be created

Table 15-39 UploadAsset

Parameter	Type	Description
url	String	URL of the audio or video file to be pulled
asset_id	String	ID of the new media asset
error_code	String	Error code
error_msg	String	Error description

Status code: 400

Table 15-40 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a media file and uses the URL pull mode to pull a media file offline from its URL and upload it to VOD.

```
POST https://{endpoint}/v1.0/{project_id}/asset/upload_by_url

Content-Type: application/json
{
  "upload_metadatas": [
    {
      "url": "https://mpc-test.obs.cn-north-4.myhuaweicloud.com/Avatar_480P.mp4",
      "title": "Avatar test test",
      "video_type": "MP4",
      "description": "Avatar, test",
      "category_id": 1,
      "tags": "mytags",
      "auto_publish": 1
    },
    {
      "url": "https://mpc-test.obs.cn-north-4.myhuaweicloud.com/Avatar_720.mp4",
      "title": "Avatar test test",
      "video_type": "MP4",
      "description": "Avatar, test",
```



```
    "category_id": 1,
    "tags": "mytags",
    "auto_publish": 1
  }
 ]
}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
 "upload_assets" : [ {
  "url" : "https://mpc-test.obs.cn-north-4.myhuaweicloud.com/Avatar_480P.mp4",
  "asset_id" : "f488337c31c8e4622f1590735b134c65",
  "error_code" : null,
  "error_msg" : null
 }, {
  "url" : "https://mpc-test.obs.cn-north-4.myhuaweicloud.com/Avatar_720.mp4",
  "asset_id" : "f488337c31c8e4622f1590525b134c65",
  "error_code" : null,
  "error_msg" : null
 } ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
 "error_code" : "VOD.10003",
 "error_msg" : "The specified key does not exist."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

15.7 Verifies the upload

Function

Checks whether the media file has been stored in VOD

URI

GET /v1.0/{project_id}/asset/duplication

Table 15-41 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 15-42 Query Parameters

Parameter	Mandatory	Type	Description
size	Yes	Integer	File size
md5	Yes	String	MD5 value of the file You are advised to refer to the example of verifying an uploaded media file in Generating an MD5 Value in the appendix of *API Reference*.

Request Parameters

Table 15-43 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 15-44 Response body parameters

Parameter	Type	Description
is_duplicated	Integer	Whether media file IDs are duplicate Possible values are: • 0: not duplicate • 1: duplicate
asset_ids	Array of strings	Duplicate media IDs

Status code: 400

Table 15-45 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Checks whether the media file has been stored in VOD.

```
GET https://{endpoint}/v1.0/{project_id}/asset/duplication?md5={md5}&size={size}
```

Example Responses

Status code: 200

The information is returned when the request succeeds.

```
{
  "is_duplicated" : 1,
  "asset_ids" : [ "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" ]
}
```

Status code: 400

The information is returned when the request fails.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
200	The information is returned when the request succeeds.

Status Code	Description
400	The information is returned when the request fails.

Error Codes

See **Error Codes**.

16 Subtitle management

16.1 Subtitle management

Function

Packages multiple subtitles in HLS VTT or HLS SRT format.

URI

PUT /v1/{project_id}/asset/subtitles

Table 16-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 16-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.

Parameter	Mandatory	Type	Description
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 16-3 Request body parameters

Parameter	Mandatory	Type	Description
asset_id	Yes	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.
default_language	No	String	Default subtitle language (The subtitle must exist.)
repackage_mode	No	String	External mode. If this parameter is not specified, the default value **0** is used. Options: -**0**: The subtitle will be inserted into the historical output. -**1**: The subtitle will not be inserted into the historical output.
delete_mode	No	String	Deletion mode. If this parameter is not specified, the default value **0** is used. Options: -**0**: Deleting the subtitle will also clear the subtitle information contained in the historical output. -**1**: Deleting the subtitle will not clear the subtitle information contained in the historical output.
add_subtitles	No	Array of **AddSubtitle** objects	Subtitle to be added or modified

Parameter	Mandatory	Type	Description
delete_subtitles	No	Array of **DeleteSubtitle** objects	Subtitle to be deleted. The value of **language** cannot be the same as that of **add_subtitles**.

Table 16-4 AddSubtitle

Parameter	Mandatory	Type	Description
type	Yes	String	Subtitle type. Currently, VTT and SRT subtitle files can be packaged.
language	Yes	String	Subtitle language
obs_info	Yes	**ObsInfo** object	Information about the OBS bucket where subtitles are stored

Table 16-5 ObsInfo

Parameter	Mandatory	Type	Description
bucket	Yes	String	OBS bucket name
object	Yes	String	OBS object path, which must be specific to the object

Table 16-6 DeleteSubtitle

Parameter	Mandatory	Type	Description
type	Yes	String	Subtitle type. Currently, VTT and SRT subtitle files can be packaged.
language	Yes	String	Subtitle language

Response Parameters

Status code: 202

Table 16-7 Response body parameters

Parameter	Type	Description
asset_id	String	Media asset ID assigned by VOD. This parameter can only be queried but cannot be modified.

Status code: 400

Table 16-8 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Subtitle management.

```
PUT https://{endpoint}/v1/{project_id}/asset/subtitles

Content-Type: application/json
{
  "asset_id": "f488337c31c8e4622f1590735b134c65",
  "default_language": "cn",
  "type": "custom_template_group",
  "is_default": true,
  "add_subtitles": [
    {
      "obs_info": {
        "bucket": "test",
        "object": "subtitle_test/happyNewYearCn.vtt"
      },
      "type": "VTT",
      "language": "cn"
    }
  ],
  "delete_subtitles": [
    {
      "type": "VTT",
      "language": "cn"
    }
  ]
}
```

Example Responses

Status code: 202

The information is returned when the request succeeded.

```
{
  "asset_id" : "a3ef6526e787acff2ca81c7da840f11f"
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
202	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

17 Transcoding template set management

17.1 Creates a transcoding template group set

Function

Creates a transcoding template group set

Constraints

- If the transcoded output types of different templates are the same, the templates cannot be added to the same transcoding template group.
- For templates with the same encoding format, the output type of the transcoding template group cannot contain HLS/DASH (either) and HLS_DASH at the same time. This constraint does not apply to templates with different encoding formats.

codec indicates the video encoding format, which can be H.264 or H.265. For details, see **Querying Media Asset Information**.

URI

POST /v1.0/{project_id}/asset/template-collection/transcodings

Table 17-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 17-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 17-3 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Template group set name.
description	No	String	Template group set description.
template_group_list	Yes	Array of strings	Template group list, template ID.

Response Parameters

Status code: 201

Table 17-4 Response body parameters

Parameter	Type	Description
group_collection_id	String	Template group set ID.

Status code: 400

Table 17-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Creates a transcoding template set.

```
POST https://{endpoint}/v1.0/{project_id}/asset/template-collection/transcodings

Content-Type: application/json
{
  "name": "test",
  "template_group_list": [
    "780640dd1d584a6999b104568c358b78",
    "6a16d8d0161c42caa42b9c148d032871"
  ]
}
```

Example Responses

Status code: 201

The information is returned when the request succeeded.

```
{
  "group_collection_id" : "f9b045e0811c482f9de0d436a5927bb6"
}
```

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
201	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

17.2 Modifies a transcoding template group set

Function

Modifies a transcoding template group set

URI

PUT /v1.0/{project_id}/asset/template-collection/transcodings

Table 17-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 17-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Table 17-8 Request body parameters

Parameter	Mandatory	Type	Description
name	No	String	Template group set name.

Parameter	Mandatory	Type	Description
collection_id	Yes	String	Template group set ID.
description	No	String	Template group set description.
template_group_list	No	Array of strings	Template group list.

Response Parameters

Status code: 400

Table 17-9 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Modifies a transcoding template set.

```
POST https://{endpoint}/v1.0/{project_id}/asset/template-collection/transcodings

Content-Type: application/json
{
  "collection_id": "f9b045e0811c482f9de0d436a5927bb6",
  "name": "test",
  "template_group_list": [
    "780640dd1d584a6999b104568c358b78",
    "6a16d8d0161c42caa42b9c148d032871"
  ]
}
```

Example Responses

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

17.3 Queries custom template group sets

Function

Queries transcoding template group sets

URI

GET /v1.0/{project_id}/asset/template-collection/transcodings

Table 17-10 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 17-11 Query Parameters

Parameter	Mandatory	Type	Description
group_collection_id	No	String	Template group set ID
offset	No	Integer	Offset. The default value is **0**. This parameter is invalid when **group_collection_id** is specified.
limit	No	Integer	Number of records on each page. The value defaults to **10** and ranges from 1 to 100. This parameter is invalid when **group_collection_id** is specified.

Request Parameters

Table 17-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 17-13 Response body parameters

Parameter	Type	Description
template_group_collection_list	Array of **TemplateGroupCollection** objects	Template group set information.
total	Integer	Total number of records.

Table 17-14 TemplateGroupCollection

Parameter	Type	Description
group_collection_id	String	Template group set ID.
name	String	Template group set name.
description	String	Template description.
template_group_list	Array of **TemplateGroup** objects	Transcoding group list.

Table 17-15 TemplateGroup

Parameter	Type	Description
group_id	String	Template group ID.
name	String	Template group name.
status	String	Default or not.
type	String	Template group type.
auto_encrypt	Integer	Whether to automatically encrypt a file Possible values are: • 0: not encrypted • 1: encrypted Default value: **0** A file must be encrypted and transcoded at the same time. When encryption is required, the transcoding parameter cannot be empty and the output file must be in HLS format.
quality_info_list	Array of **QualityInfo** objects	Image quality configuration list.
watermark_template_ids	Array of strings	ID array of the bound watermark template group.
description	String	Template description.
common	**Common** object	Low-bitrate HD transcoding switch, low-bitrate HD transcoding version, and multi-stream common parameters.

Table 17-16 QualityInfo

Parameter	Type	Description
video	**VideoTemplateInfo** object	Video template information
audio	**AudioTemplateInfo** object	Audio template information
format	String	Format

Table 17-17 VideoTemplateInfo

Parameter	Type	Description
quality	String	Video quality
width	Integer	Video width
height	Integer	Video height
bitrate	Integer	Bitrate
frame_rate	Integer	Frame rate (in fps), which defaults to 1 indicating adaptive frame rate

Table 17-18 AudioTemplateInfo

Parameter	Type	Description
sample_rate	Integer	Audio sampling rate. Possible values are: ● 1: AUDIO_SAMPLE_AUTO ● 2: AUDIO_SAMPLE_22050 ● 3: AUDIO_SAMPLE_32000 ● 4: AUDIO_SAMPLE_44100 ● 5: AUDIO_SAMPLE_48000 ● 6: AUDIO_SAMPLE_96000 The default value is **1**.
bitrate	Integer	Audio bitrate, in kbit/s
channels	Integer	Number of audio channels. Possible values are: ● 1: AUDIO_CHANNELS_1 ● 2: AUDIO_CHANNELS_2

Table 17-19 Common

Parameter	Type	Description
pvc	String	Low-bitrate HD transcoding switch.
pvc_version	String	Low-bitrate HD transcoding version.
video_codec	String	Video codec

Parameter	Type	Description
audio_codec	String	Audio codec. Possible values are: • 1: AUDIO_CODECTYPE_AAC • 2: AUDIO_CODECTYPE_HEAAC1 • 3: AUDIO_CODECTYPE_HEAAC2 • 4: AUDIO_CODECTYPE_MP3 The default value is **1**.
hls_interval	Integer	Segment file duration, which defaults to 5 seconds

Status code: 400

Table 17-20 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Queries transcoding template sets.

```
GET https://{endpoint}/v1.0/{project_id}/asset/template-collection/transcodings
```

Example Responses

Status code: 200

The information is returned when the request succeeded.

```
{
  "template_group_collection_list" : [ {
    "group_collection_id" : "9751249d25f14587b212544d6fd8dcf8",
    "name" : "test",
    "template_group_list" : [ {
      "group_id" : "9751249d25f14587b212544d6fd8dcf8",
      "name" : "test112",
      "status" : "0",
      "type" : "custom_template_group",
      "auto_encrypt" : 0,
      "quality_info_list" : [ {
        "video" : {
          "quality" : "UNKNOW",
          "width" : 0,
          "height" : 0,
          "bitrate" : 0,
          "frame_rate" : 0
        },
        "audio" : null,
        "format" : "UNKNOW"
      } ],
```



```
    "watermark_template_ids" : null,
    "description" : null,
    "common" : {
      "pvc" : null,
      "pvc_version" : null,
      "video_codec" : null,
      "audio_codec" : "AAC",
      "hls_interval" : 0
    }
  } ]
 } ],
 "total" : 1
}
```


Status code: 400

The information is returned when the request failed.


```
{
 "error_code" : "VOD.10053",
 "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```


Status Codes

Status Code	Description
200	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

17.4 Deletes a transcoding template group set

Function

Deletes a transcoding template group set

URI

DELETE /v1.0/{project_id}/asset/template-collection/transcodings

Table 17-21 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Table 17-22 Query Parameters

Parameter	Mandatory	Type	Description
group_collection_id	Yes	String	Template group set ID

Request Parameters

Table 17-23 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is a token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when the request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 400

Table 17-24 Response body parameters

Parameter	Type	Description
error_code	String	Error code
error_msg	String	Error description

Example Requests

Deletes a transcoding template set.

```
DELETE https://{endpoint}/v1.0/{project_id}/asset/template-collection/transcodings?
group_collection_id={group_collection_id}
```


Example Responses

Status code: 400

The information is returned when the request failed.

```
{
  "error_code" : "VOD.10053",
  "error_msg" : "The request parameter is illegal, illegal field: {xx}."
}
```

Status Codes

Status Code	Description
204	The information is returned when the request succeeded.
400	The information is returned when the request failed.

Error Codes

See **Error Codes**.

18 Historical APIs

18.1 Media Asset Upload

18.1.1 Obtaining Authorization for Multipart Upload

Functions

When a client requests for creating a media asset larger than 20 MB, the media asset needs to be uploaded to OBS by part. The client needs to obtain authorization using this API each time before uploading a file part to OBS.

This API is used to obtain a temporarily authorized URL for initializing multipart upload, uploading parts, merging parts, listing uploaded parts, and canceling part merging. You need to configure the HTTP request method, request header, and request body by following the OBS API reference. Then you can request for the corresponding temporarily authorized URL.

The multipart upload method is the same as that in the OBS API reference, including the HTTP request method, request header, and request body. This API is used to generate authentication information (**sign_str**) to replace the URL in the OBS API, so that you have the temporary permission for uploading files to the bucket of VOD.

When calling the API for obtaining authorization, input **bucket**, **object_key**, and **http_verb**. **bucket** and **object_key** are obtained from the target field in the response body returned by the API for **uploading media assets to VOD**. **http_verb** varies depending on the specified operation.

URI

GET /v1.0/{project_id}/asset/authority

Table 18-1 URI parameters

Parameter	Mandatory	Parameter Type	Description
project_id	Yes	String	Project ID. See **Obtaining the Project ID**.

Table 18-2 Query parameters

Parameter	Mandatory	Parameter Type	Description
http_verb	Yes	String	HTTP method for calling the OBS API for multipart upload. For details about the suitable HTTP method, see the OBS API reference. • Initializing an upload task: POST • Uploading parts: PUT • Merging parts: POST • Canceling a part: DELETE • Listing uploaded parts: GET
bucket	Yes	String	Bucket name. Value of **bucket** obtained in the target field in the response body returned by the API for **uploading media assets to VOD**.
object_key	Yes	String	Object name. Value of **object** obtained in the target field in the response body returned by the API for **uploading media assets to VOD**.

Parameter	Mandatory	Parameter Type	Description
content_type	No	String	**content-type** of the file type. This parameter is mandatory for upload task initialization. Configure the parameters by referring to **Uploading a Media Asset Greater Than 20 MB by Part**. • Video file: video/*video format*, for example, video/mp4. • Audio file: audio/*audio format*, for example, audio/mp3 • Image file: image/*image format*, for example, image/png • Subtitle file: **application/octet-stream**
content_md5	No	String	MD5 value of each uploaded part.
upload_id	No	String	ID of each upload task, which is returned after OBS initializes the multipart upload task. This field is mandatory except for upload task initialization.
part_number	No	Integer	ID of each uploaded part. Value range: 1–10,000

Request Parameters

Table 18-3 Request header parameters

Parameter	Mandatory	Parameter Type	Description
X-Auth-Token	No	String	User token. This parameter is mandatory when token authentication is used. The token can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Authorization	No	String	Authentication information. This parameter is mandatory for AK/SK authentication.
X-Sdk-Date	No	String	Time when a request is sent. This parameter is mandatory for AK/SK authentication.

Response Parameters

Status code: 200

Table 18-4 Response body parameters

Parameter	Parameter Type	Description
sign_str	String	Signed string. Example: AWSAccessKeyId={AccessKeyID}&Expires={ExpiresValue}&Signature={Signature}

Status code: 403

Table 18-5 Response body parameters

Parameter	Parameter Type	Description
error_code	String	Error code.
error_msg	String	Error description.

Request Examples

- Initialize an upload task:
  ```
  GET https://{endpoint}/v1.0/{project_id}/asset/authority?
  http_verb=POST&content_type={type}&bucket={bucket}&object_key={objectKey}
  ```
- Upload parts:
  ```
  GET https://{endpoint}/v1.0/{project_id}/asset/authority?
  http_verb=PUT&content_md5={md5}&part_number={num}&upload_id={id}&bucket={bucket}&object_key={objectKey}
  ```
- Merge parts:
  ```
  GET https://{endpoint}/v1.0/{project_id}/asset/authority?
  http_verb=POST&upload_id={id}&bucket={bucket}&object_key={objectKey}
  ```
- Cancel a part:
  ```
  GET https://{endpoint}/v1.0/{project_id}/asset/authority?
  http_verb=DELETE&bucket={bucket}&object_key={objectKey}&upload_id={uploadId}
  ```
- List uploaded parts:
  ```
  GET https://{endpoint}/v1.0/{project_id}/asset/authority?
  http_verb=GET&bucket={bucket}&object_key={objectKey}&upload_id={uploadId}
  ```

Response Examples

Status code: 200

Returned when the request succeeded.

```
{
  "sign_str" :
"AWSAccessKeyId=UBWV*********N4NOJ&Expires=1730739600&Signature=47rXht3Qrd**********GiIxM40%3D"
}
```

Status Codes

Status Code	Description
200	Returned when the request succeeded.
403	Returned when the request failed.

Error Codes

See **Error Codes**.

19 Appendix

19.1 Status Codes

Table 19-1 lists the status codes that may be returned when you call VOD APIs.

Table 19-1 Status codes

Status Code	Description
200 OK	The request has succeeded.
201 Created	The request has been fulfilled and resulted in a new resource being created.
202 Accepted	The request has been accepted for processing, but the processing has not been completed.
204 No Content	The request has been fulfilled, but the HTTP response does not contain a response body.
400 Bad Request	The request could not be understood by the server. The client should not repeat the request without modifications.
401 Unauthorized	The authentication information provided by the client is incorrect or invalid.
403 Forbidden	The server understood the request, but is refusing to fulfill it. The client should not repeat the request without modifications.
404 Not Found	The requested resource could not be found. The client should not repeat the request without modifications.
405 Method Not Allowed	The method specified in the request is not supported for the requested resource. The client should not repeat the request without modifications.
406 Not Acceptable	The server could not fulfill the request according to the content characteristics of the request.

Status Code	Description
407 Proxy Authentication Required	This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy.
408 Request Timeout	The client did not produce a request within the time that the server was prepared to wait. The client may repeat the request without modifications at any later time.
409 Conflict	The request could not be completed due to a conflict with the current state of the resource. This status code indicates that the resource that the client is attempting to create already exists, or that the request has failed to be processed because of the update of the conflict request.
500 Internal Server Error	The server is able to receive the request but unable to understand it.
501 Not Implemented	The server does not support the functionality required to fulfill the request.
502 Bad Gateway	The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
503 Service Unavailable	The server is currently unable to handle the request. The client should not repeat the request without modifications.
504 Gateway Timeout	A gateway timeout error occurred.

19.2 Error Codes

If an error code starting with APIGW is returned after you call an API, rectify the fault by referring to the instructions provided in **API Gateway Error Codes**.

Status Code	Error Codes	Error Message	Description	Solution
400	VOD.10053	The request parameter is illegal.	The request parameter is illegal, illegal field	Check against the parameter value format defined in the API.
400	VOD.10081	The request parameters are illegal, please check.	The request parameters are illegal, please check.	Invalid request parameter.

Status Code	Error Codes	Error Message	Description	Solution
400	VST.10001	The request parameter is illegal.	The request parameter is illegal, illegal field	Check against the parameter value format defined in the API.
400	VST.10002	Internal system error.	System problem	Contact technical support.
400	VST.10004	Identity authenticatio n failed.	Authenticatio n failed	Check whether authentication parameters such as Token are correct.
403	VOD.10051	Internal system error.	System problem	Contact technical support.
403	VOD.10052	The internal communicatio n of the service is abnormal.	The internal communicatio n of the service is abnormal	Contact technical support.
403	VOD.10054	Identity authenticatio n failed.	Authenticatio n failed	Check whether authentication parameters such as Token are correct.
403	VOD.10055	The user is not authenticated by real name.	User does not have real-name authenticatio n	Check whether real-name authentication has been completed.
403	VOD.10056	The user is in an abnormal state.	The user is in an abnormal state	Check your account status.
403	VOD.10057	Tenant ID verification failed.	Tenant ID verification failed	Check whether the tenant ID is correct.
403	VOD.10058	The request method is incorrect.	The request method is incorrect.	Check the request method.
403	VOD.10059	The requested content type is incorrect.	The requested content type is incorrect	Check the request content type.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10060	Media asset is not downloaded successfully or not released, please check.	Media asset is not downloaded successfully or not released, please check.	The media file has not been downloaded or published. Please check.
403	VOD.10061	The operation failed, and the current media asset status is not allowed to be updated.	The operation failed, and the current media asset status is not allowed to be updated.	Operation failed. The current media file status cannot be updated.
403	VOD.10063	Media asset type is not supported at this time.	Media asset type is not supported at this time.	The media file type is not supported currently.
403	VOD.10065	Media asset classification already exists, please check.	Media asset classification already exists, please check.	The media file category already exists. Please check.
403	VOD.10066	The media asset classification exceeds the maximum level, please check.	The media asset classification exceeds the maximum level, please check.	The number of media file category levels exceeds the maximum. Please check.
403	VOD.10067	Media asset classification exceeds the maximum number limit, please check.	Media asset classification exceeds the maximum number limit, please check.	The number of media file categories exceeds the maximum. Please check.
403	VOD.10069	The topic already exists, please check.	The topic already exists, please check.	The topic already exists. Please check.
403	VOD.10070	Set message notification failed, no permission to post message to topic, please check.	Set message notification failed, no permission to post message to topic, please check.	Configure event notifications failed. You do not have the permission for publishing messages to the topic.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10071	Referer header field verification failed, please check.	Referer header field verification failed, please check.	Verify the referer header failed. Please check.
403	VOD.10072	The url authenticatio n setting failed. The old key has not expired. The new key cannot be set. Please set the new key again after expiration.	The url authenticatio n setting failed. The old key has not expired. The new key cannot be set. Please set the new key again after expiration.	Configure URL authentication failed. The old key has not expired and the new key cannot be configured. After the old key expires, configure the new key.
403	VOD.10073	Failed to create a media asset transcoding task. Contact technical support.	Failed to create a media asset transcoding task. Contact technical support.	Create a media file transcoding task failed. Contact technical support.
403	VOD.10074	There is no permission to perform this operation.	There is no permission to perform this operation.	You do not have the permission for performing this operation.
403	VOD.10076	Failed to get the object storage source file.	Failed to get the object storage source file.	Obtain the input file from OBS failed.
403	VOD.10077	hms request playback interface authenticatio n failed.	hms request playback interface authenticatio n failed.	The HMS request to the playback API failed the authentication.
403	VOD.10078	The public measurement limit, the usage exceeds the threshold.	The public measurement limit, the usage exceeds the threshold.	The usage exceeds the OBT quota.
403	VOD.10079	The task was processed successfully.	The task was processed successfully.	The task has been completed.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10080	The task is being processed.	Task processing.	The task is being handled.
403	VOD.10082	The template group already exists.	The template group already exists.	The template group already exists.
403	VOD.10083	The current media asset status does not support this operation.	The current media asset status does not support this operation.	This operation is not supported under the current media file or resource status.
403	VOD.10084	Media asset release failed.	Media asset release failed.	Publish the media file failed.
403	VOD.10085	The number of domain names allowed to be created cannot exceed {0}.	The number of domain names allowed to be created cannot exceed {0}.	The number of created domain names cannot exceed {0}.
403	VOD.10086	The total number of preheating urls cannot exceed {0}.	The total number of preheating urls cannot exceed {0}.	The number of URLs to be pre-loaded cannot exceed {0}.
403	VOD.10087	The total number of refresh urls cannot exceed {0}.	The total number of refresh urls cannot exceed {0}.	The number of URLs to be refreshed cannot exceed {0}.
403	VOD.10088	Only operate enabled domain name.	Only operate enabled domain name.	You can perform operations only on enabled domain names.
403	VOD.10090	The modification failed, request to confirm the watermark image upload first.	The modification failed, request to confirm the watermark image upload first.	Modification failed. Request for uploading the watermark image first.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10091	Media asset transcoding was successful.	Media asset transcoding was successful.	The media file has been transcoded.
403	VOD.10092	Media asset transcoding failed.	Media asset transcoding failed.	Media file transcoding failed.
403	VOD.10093	The OBS transfer media asset was successfully released.	The OBS transfer media asset was successfully released.	The media file dumped to OBS has been published.
403	VOD.10094	The OBS transfer media asset failed to be released.	The OBS transfer media asset failed to be released.	Publish the media file dumped to OBS failed.
403	VOD.10095	The source domain name cannot be configured to speed up the domain name.	The source domain name cannot be configured to speed up the domain name.	The origin server domain name cannot be configured as an acceleration domain name.
403	VOD.10096	The domain name has been used.	The domain name has been used.	The domain name has been used.
403	VOD.10097	Default accelerated domain name cannot be created.	Default accelerated domain name cannot be created.	The default acceleration domain name cannot be created.
403	VOD.10098	Operation failed, domain name is configuring.	Operation failed, domain name is configuring.	Operation failed. The domain name is being configured.
403	VOD.10099	Media asset is BLOCKED.	Media asset is BLOCKED.	The media file has been blocked.
403	VOD.10100	No key URL has been configured.	No key URL has been configured.	The URL for obtaining the key is not configured.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10101	Unable to get the key because there is no encryption.	Unable to get the key because there is no encryption.	The key cannot be obtained because the media file is not encrypted.
403	VOD.10102	Edit task only support Mp4/Flv video.	Edit task only support Mp4/Flv video.	Only MP4 and FLV media files can be clipped.
403	VOD.10103	Concat index asset must in input asset list.	Concat index asset must in input asset list.	The reference video must be in the list of videos to be merged.
403	VOD.10104	The thumbnail task failed.	The thumbnail task failed.	Snapshot capturing failed.
403	VOD.10105	The review task failed.	The review task failed.	The file failed the review.
403	VOD.10106	Cannot delete the default assigned domain name or default domain name.	Cannot delete the default assigned domain name or default domain name	The default assigned domain name or default domain name cannot be deleted.
403	VOD.10107	Default allocation of domain names cannot configure Https.	Default allocation of domain names cannot configure Https	HTTPS cannot be configured for the default VOD domain name.
403	VOD.10108	The interval of Cut task is too small.	The interval of Cut task is too small.	The clip interval is too short.
403	VOD.10109	TinyAsset preheat url is over 10 limit.	TinyAsset preheat url is over 10 limit.	A maximum of 10 short video URLs can be pre-loaded.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10110	Unable to create cdn preHeating task, please check if the asset is enabled for cdn.	Unable to create cdn preHeating task, please check if the asset is enabled for cdn.	Create the CDN pre-loading task failed. Check whether the CDN acceleration domain name is enabled for the media file.
403	VOD.10111	No access to resources.	No access to resources.	You do not have the permission for accessing the requested resource.
403	VOD.10113	OBS notify config is not set.	OBS notify config is not set.	Event notification is not configured for OBS.
403	VOD.10114	Bucket not Authorized.	Bucket not Authorized.	The access to an OBS bucket has not been authorized.
403	VOD.10115	The output bucket can not be cannel auth.	The output bucket can not be cannel auth.	The authorization for accessing an output bucket cannot be canceled.
403	VOD.10116	Event notification is being configured for OBS.	Event notification is being configured for OBS.	Event notification is being configured for OBS.
403	VOD.10118	Bucket asset is handling..	Bucket asset is handling..	The audio and video of the bucket are being synchronized.
403	VOD.10119	URL mapping already exists.	URL mapping already exists.	The URL mapping already exists.
403	VOD.10120	You have arrears, please recharge.	You have arrears, please recharge.	Your account is in arrears. Top up your account.
403	VOD.10121	URL pull asset failed.	URL pull asset failed.	Pull the media file from the URL failed.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10122	Cancel url pull failed.	Cancel url pull failed.	The operation of pulling the media file from the URL cannot be canceled.
403	VOD.10123	The url pull task is processing or stopping can not be delete.	The url pull task is processing or stopping can not be delete.	If the task for pulling the media file from the URL is being handled or canceled, the task cannot be deleted.
403	VOD.10124	Watermark template is valid or not exists, {0}	Watermark template is valid or not exists, {0}	The watermark template group is invalid or does not exist. {0}.
403	VOD.10125	Watermark number reach limitation: {0}	Watermark number reach limitation: {0}	The number of watermarks has reached the upper limit {0}.
403	VOD.10126	Uploading confirmation is not allowed in the current media asset status. Please check the media asset status.	Uploading confirmation is not allowed in the current media asset status. Please check the media asset status.	The media file in the current status cannot be uploaded. Check the media file status.
403	VOD.10127	APIGW rate limit.	APIGW rate limit.	MAI request throttling
403	VOD.10128	The task status does not support recovery operations.	The task status does not support recovery operations.	The restoration operation is not supported under the current task status.
403	VOD.10130	The domain name does not support configuration key anti-theft chain: {0}.	The domain name does not support configuration key anti-theft chain: {0}.	The domain name {0} does not support URL validation.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10131	The watermark template already exists.	The watermark template already exists.	The watermark template already exists.
403	VOD.10132	Account cancellation or public security freeze, access to services is not allowed.	Account cancellation or public security freeze, access to services is not allowed.	The service cannot be accessed because the account has been deregistered or frozen.
403	VOD.10133	No permission to operate related resource.	No permission to operate related resource.	You do not have the permission for performing operations on related resources.
403	VOD.10134	The domain name is invalid and cannot be created.	The domain name is invalid and cannot be created.	The domain name is invalid and cannot be created.
403	VOD.10135	The asset is not in the upload state and cannot obtain temporary authorization.	The asset is not in the upload state and cannot obtain temporary authorization.	Temporary authorization cannot be obtained because the media file is not being uploaded.
403	VOD.10136	This feature is temporarily offline.	This feature is temporarily offline.	This function is temporarily brought offline.
403	VOD.10137	The number of published or unpublished meta resources exceeds the threshold limit:{0}	The number of published or unpublished meta resources exceeds the threshold limit:{0}	The number of media files to be published or canceled exceeds the threshold {0}.

Status Code	Error Codes	Error Message	Description	Solution
403	VOD.10138	You do not have permission to operate, contact a tenant administrator barrel authorize or sub-account to the tenant administrator privileges conferred OBS.	You do not have permission to operate, contact a tenant administrator barrel authorize or sub-account to the tenant administrator privileges conferred OBS.	You do not have the operation permission. Contact the tenant administrator to authorize the access to the bucket or grant the OBS permission to you.
404	VOD.10062	Media asset or resource does not exist, please check.	Media asset or resource does not exist, please check.	The media file or resource does not exist. Please check.
404	VOD.10064	Media asset classification does not exist, please check.	Media asset classification does not exist, please check.	The media file category does not exist. Please check.
404	VOD.10068	The theme does not exist, please check.	The theme does not exist, please check.	The topic does not exist. Please check.
404	VOD.10075	The object storage source address or destination address is incorrect. Please check.	The object storage source address or destination address is incorrect. Please check.	The input or output OBS path is invalid. Please check.
404	VOD.10089	Accelerated domain name does not exist.	Accelerated domain name does not exist.	The acceleration domain name does not exist.
404	VOD.10112	OBS Resource not exists.	OBS Resource does not exist.	Check the OBS resource or contact technical support.

Status Code	Error Codes	Error Message	Description	Solution
404	VOD.10129	Subtitle resource does not exist, please check.	Subtitle resource does not exist, please check.	The subtitle file is not found. Please check.
404	VOD.10150	Workflow does not exist, please check.	Workflow does not exist.	Check the workflow or contact technical support.
404	VOD.10151	TemplateGroup does not exist, please check.	TemplateGroup does not exist.	Check the transcoding template or contact technical support.
404	VOD.10152	Domain does not exist, please check.	Domain does not exist.	Check the domain name or contact technical support.
500	VOD.10051	Internal system error.	System problem	Contact technical support.
500	VST.10002	Internal system error.	System problem	Contact technical support.
400	LIVE.100011001	Invalid request parameter.	Invalid request parameter.	Mandatory parameters are missing, or the parameter value format is incorrect. For details about the error, see the error_detail field in the error description.
400	LIVE.100011004	The protocol is not supported. The API only supports the HTTPS protocol.	The protocol is not supported. The API only supports the HTTPS protocol.	Use the HTTPS protocol.
400	LIVE.100011006	This API is not supported in this version or is in the maintenance state.	This API is not supported in this version or is in the maintenance state.	Try again later or contact customer service.

Status Code	Error Codes	Error Message	Description	Solution
400	LIVE.100011008	The API is in the maintenance state.	The API is in the maintenance state.	Try again later or submit a service ticket to reach technical support.
400	LIVE.100011009	The requested user does not exist.	The requested user does not exist.	Check whether input information is correct.
400	LIVE.100011011	The method specified in the request is not supported.	The method specified in the request is not supported.	Check whether the HTTP method is correct.
400	LIVE.100011012	Unsupported media type.	Unsupported media type.	Submit a service ticket to reach technical support.
400	LIVE.100011013	You have not completed real-name authentication.	You have not completed real-name authentication.	Complete real-name authentication.
400	LIVE.100011022	LIVE.Record task exist.	The recording task already exists.	After the command for stopping recording is run, run the command for starting recording for the same stream at an interval of more than 10 seconds.
400	LIVE.103011018	The resource already exists.	The resource already exists.	Check whether the requested resource exists.

Status Code	Error Codes	Error Message	Description	Solution
400	LIVE.103011020	The maximum number of domain names has been reached.	The maximum number of domain names has been reached.	Delete unnecessary domain names and add domain names again. If you have added the maximum number of domain names, but you want to add more, submit a service ticket.
400	LIVE.103011021	Failed to access the database.	Failed to access the database.	Submit a service ticket to reach technical support.
400	LIVE.103011022	This operation is not allowed in the current state.	This operation is not allowed in the current state.	Check the state of the domain name. Only domain names in the Disabled state can be modified or deleted.
400	LIVE.103011024	The ICP number does not exist.	The ICP number does not exist.	Check whether the ICP number is correct.
400	LIVE.103011025	The approved ICP number cannot be modified.	The approved ICP number cannot be modified.	The ICP number cannot be modified after being approved. If you need to change the ICP number, submit a service ticket.
400	LIVE.103011026	The domain name is in the blacklist.	The domain name is in the blacklist.	Use a licensed domain name.
400	LIVE.103011027	The domain name is in the blacklist.	The domain name is in the blacklist.	Use a licensed domain name.

Status Code	Error Codes	Error Message	Description	Solution
400	LIVE.103011029	The domain name is in gray configuration.	This domain name has grayscale configuration, and the configuration failed to be delivered.	Submit a service ticket to reach technical support.
400	LIVE.103011030	This domain name has customized configuration.	This domain name has customized configuration, and the configuration failed to be delivered.	Submit a service ticket to reach technical support.
401	LIVE.100011002	Identity authenticatio n failed.	Identity authenticatio n failed.	This problem is generally caused by incorrect signature calculation. For details, see the signature method in the document.
401	LIVE.100011003	You do not have permission to access the API.	You do not have permission to access the API.	Contact the master account administrator to obtain permission to access the API.
401	LIVE.100011014	Your account is not allowed to access the service, because it is frozen, deleted, or has insufficient balance.	Your account is not allowed to access the service, because it is frozen, deleted, or has insufficient balance.	Check whether your account is frozen or suspended due to arrears.
401	LIVE.100011015	Project ID verification failed.	Project ID verification failed.	Check the project ID and request header in the request.

Status Code	Error Codes	Error Message	Description	Solution
403	LIVE.103011016	Invalid request content.	Invalid request content.	Change to a valid request parameter.
404	LIVE.103011019	The resource does not exist.	The resource does not exist.	Check whether the requested resource exists or whether your account information is correct.
500	LIVE.100011000	Internal communication error.	Internal communication error.	Submit a service ticket to reach technical support.
500	LIVE.100011005	Internal server error.	Internal server error.	Try again later or contact customer service.
500	LIVE.100011007	The number of concurrent API requests exceeds the upper limit.	The number of concurrent API requests exceeds the upper limit.	Submit a service ticket to reach technical support.
500	LIVE.103011017	Failed to synchronize data to CDN.	Failed to synchronize data to CDN.	Check whether the domain name has been registered and ICP number is valid. If its ICP number is still invalid, submit a service ticket to reach technical support.
500	LIVE.103011023	Failed to synchronize the domain name to GSLB.	Failed to synchronize the domain name to GSLB.	Submit a service ticket to reach technical support.
500	LIVE.103011028	Failed to synchronize the domain name to DNS.	Failed to synchronize the domain name to DNS.	Submit a service ticket to reach technical support.

19.3 Obtaining a Project ID

A project ID is required for some URLs when an API is called. You can obtain a project ID using either of the following methods:

- **Obtaining a Project ID from the Management Console**
- **Obtaining a Project ID by Calling an API**

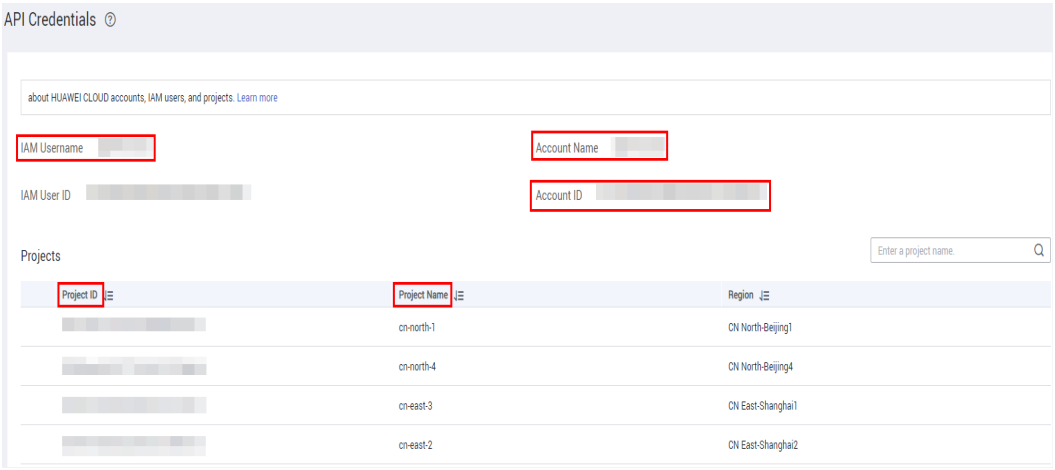
Obtaining a Project ID from the Management Console

Step 1 Log in to the console.

Step 2 Hover over the username in the upper right corner and select **My Credentials** from the drop-down list.

On the **API Credentials** page, view project IDs in the project list.

Figure 19-1 Obtaining a project ID



----End

Obtaining a Project ID by Calling an API

You can obtain a project ID by calling the API for **querying project information**.

The API for obtaining a project ID is **GET https://*{Endpoint}*/v3/projects/**. *{Endpoint}* indicates the endpoint of IAM, which can be obtained from **Endpoints**. For details about API authentication, see **Authentication**.

The following is an example response. The value of **id** under **projects** is the project ID.

```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d14e684b",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d14e684b",
            "name": "region01",
            "description": "",
```



```
        "links": {
            "next": null,
            "previous": null,
            "self": "https://www.example.com/v3/projects/a4a5d4098fb4474fa22cd05f897d6b99"
        },
        "id": "a4a5d4098fb4474fa22cd05f897d6b99",
        "enabled": true
    }
  ],
  "links": {
    "next": null,
    "previous": null,
    "self": "https://www.example.com/v3/projects"
  }
}
```


19.4 Obtaining an Account ID

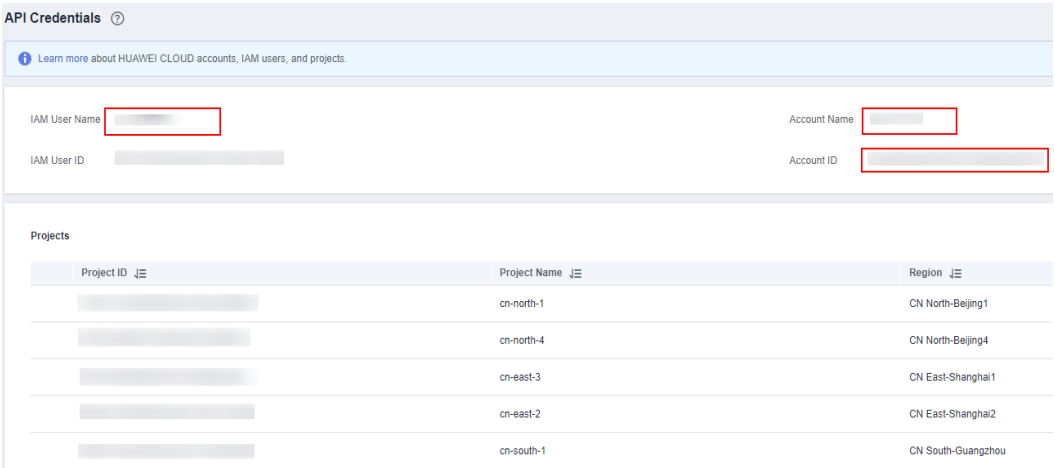
An account ID is required for some URLs when an API is called. To obtain an account ID, perform the following steps:

Step 1 Log in to the console.

Step 2 Hover over the username in the upper right corner and select **My Credentials** from the drop-down list.

On the **API Credentials** page, obtain the account ID.

Figure 19-2 Obtaining an account ID



----End

19.5 Generating an MD5 Value

Media Upload and Update

When calling the APIs for **uploading media files to VOD** and **video update**, you can use **video_md5** to configure the MD5 value of the media file. OBS will verify the MD5 value. For details, see **Setting Object Properties**.

The MD5 value is calculated using the standard MD5 hash algorithm and then encoded using Base64. The sample code is as follows:

```
import org.apache.commons.codec.binary.Base64;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;


public class VodDemoObsCheckMd5 {
    public String md5Generate4ObsCheck(String fileUrl)
        throws IOException, NoSuchAlgorithmException {
        String md5Content = null;

        if ((fileUrl != null) && (fileUrl.length() != 0)) {
            File file = new File(fileUrl);

            if (!file.exists()) {
                System.out.println("The file does not exist.");
            } else if (file.isDirectory()) {
                System.out.println(file.getCanonicalPath() + "is a directory and cannot be calculated.);
            } else {
                FileInputStream fis = new FileInputStream(file);
                DigestInputStream dis = new DigestInputStream(fis,
                        MessageDigest.getInstance("MD5"));
                byte[] buffer = new byte[8192];

                while (dis.read(buffer) > 0) {
                }

                md5Content = new String(Base64.encodeBase64(
                            dis.getMessageDigest().digest()));
                fis.getChannel().position(0L);
                System.out.println("The MD5 value of the file is "+ md5Content);
                fis.close();
            }
        } else {
            System.out.println("The file name is missing.");
        }

        return md5Content;
    }
}
```

Verification

VOD uses the MD5 values of media files to check whether duplicate media files exist when you call the API for **verifying the upload**. The MD5 value is generated based on the 1,024 bytes of the media file and calculated using the MD5 algorithm. The sample code is as follows:

```
import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.SeekableByteChannel;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import org.apache.commons.codec.digest.DigestUtils;

public class VodDemoDuplicateCheckMd5 {
    public static String computeMd5ByFile(String fileUrl) {
```



```
        String md5Content = null;
        Path targetFile = Paths.get(fileUrl);
        try (SeekableByteChannel channel = Files.newByteChannel(targetFile, StandardOpenOption.READ)) {
            ByteBuffer byteBuffer = ByteBuffer.allocate(1025);
            channel.read(byteBuffer);
            byteBuffer.flip();
            byte[] data = new byte[byteBuffer.limit()];
            byteBuffer.get(data);
            md5Content = DigestUtils.md5Hex(data);
        } catch (IOException e) {
            throw new RuntimeException(String.format("Read file %s failed.", fileUrl));
        }
        return md5Content;
    }
}
```

19.6 Log Management

Log Description

- Log file latency: You can query log files generated over the past six hours on the **Logs** page.
- The log naming rules are as follows: *Log time span-Acceleration domain name-Service area*.gz. The service area is represented by a two letter abbreviation. Logs ending in **cn** are for areas in the Chinese mainland, and those ending in **ov** are for areas outside the Chinese mainland. So a typical log name might be **2018021123-www.example01.com-ov.gz**.
- By default, a log file is generated for each domain name every hour, and 24 log files are generated every day.
- Example of log file content

```
[05/Feb/2018:07:54:52 +0800] x.x.x.x 1 "-" "HTTP/1.1" "GET" "www.test.com" "/test/1234.apk" 206
720 HIT "Mozilla/5.0 (Linux; U; Android 6.0; en-us; EVA-AL10 Build/HUAWEIEVA-AL10) AppleWebKit/
533.1 (KHTML, like Gecko) Mobile Safari/533.1" "bytes=-256"
```

  Table 19-2 describes each field (from left to right) in the log.

Table 19-2 Log fields

No.	Field Description	Example
1	Log generation time	[05/Feb/2018:07:54:52 +0800]
2	Access IP address	x.x.x.x
3	Latency (ms)	1
4	Referer information	-
5	HTTP protocol identifier	HTTP/1.1
6	HTTP request method	GET
7	Acceleration domain name	www.test.com
8	Requested path	/test/1234.apk
9	HTTP status code	206

No.	Field Description	Example
10	Response size (in bytes)	720
11	Cache hit status	HIT
12	User-Agent information, which helps servers recognize the OS, OS version, CPU, browser, and browser's version information	Mozilla/5.0 (Linux; U; Android 6.0; zh-cn; EVA-AL10 Build/HUAWEIEVA-AL10) AppleWebKit/533.1 (KHTML, like Gecko) Mobile Safari/533.1
13	Range information specifies the positions of the first and last bytes for the data to be returned. **bytes** can be expressed by the following three methods: • bytes=x-y: requesting content from the *x*th to *y*th byte • bytes=-y: requesting content from the last *y* bytes • bytes=x-: requesting content from the *x*th to the last byte	bytes=-256

19.7 Sample Code for Multipart Upload

19.7.1 Java

Sample code of multipart upload using Java:

```
/*
 * Copyright (c) Huawei Technologies Co., Ltd. 2019-2022. All rights reserved.
 */
package com.huaweicloud.vod;

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.vod.v1.VodClient;
import com.huaweicloud.sdk.vod.v1.model.ConfirmAssetUploadReq;
import com.huaweicloud.sdk.vod.v1.model.ConfirmAssetUploadRequest;
import com.huaweicloud.sdk.vod.v1.model.ConfirmAssetUploadResponse;
import com.huaweicloud.sdk.vod.v1.model.CreateAssetByFileUploadReq;
import com.huaweicloud.sdk.vod.v1.model.CreateAssetByFileUploadRequest;
import com.huaweicloud.sdk.vod.v1.model.CreateAssetByFileUploadResponse;
```

```
import com.huaweicloud.sdk.vod.v1.model.ShowAssetTempAuthorityRequest;
import com.huaweicloud.sdk.vod.v1.model.ShowAssetTempAuthorityResponse;
import com.huaweicloud.sdk.vod.v1.region.VodRegion;

import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang3.StringUtils;
import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.DocumentHelper;
import org.dom4j.Element;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Arrays;
import java.util.List;
import java.util.TimeZone;

/**
 * Example of multipart upload
 */
public class PartUploadDemo {

    // Set the buffer size as needed, that is, the size of the file part read each time.
    public static final int bufferSize = 1024 * 1024; // 1MB

    // Region
    public static final String REGION_NORTH4 = "cn-north-4";
    public static final String REGION_NORTH1 = "cn-north-1";
    public static final String REGION_EAST2 = "cn-east-2";

    // ak/sk
    private static final String AK = "";
    private static final String SK = "";

    public static void main(String[] args) {
        // Path of the local media asset to be uploaded
        String filePath = "";

        PartUploadDemo uploadDemo = new PartUploadDemo();
        // Upload the media asset.
        uploadDemo.uploadPartFile(filePath);
    }

    /**
     * Multipart upload
     * @param filePath: local path where the file to be uploaded is
     */
    public void uploadPartFile(String filePath) {
        // Verify the file and its path.
        File file = validFile(filePath);
        try {
            String fileName = file.getName();

            // An MP4 file is used as an example. For details about other formats, see the official website.
            String fileType = "MP4";
            String fileContentType = "video/mp4";

            // FileInfo fileInfo = new FileInfo(filePath, file);
            // 1. Initialize authentication and obtain vodClient.
            VodClient vodClient = this.createVodClient();
            System.out.println("Start creating the media asset:" + file.getName());
            // 2. Create a VOD media asset.
```



```
        CreateAssetByFileUploadReq reqbody = this.buildFileUploadReq(fileName, fileType, null, null);
        CreateAssetByFileUploadResponse assetResponse = this.createAssetByFileUpload(vodClient,
reqbody);
        // 3. Obtain authorization for initializing an upload task.
        ShowAssetTempAuthorityResponse initAuthResponse = this.initPartUploadAuthority(vodClient,
assetResponse, fileContentType);
        // 4. Initialize the upload task.
        String uploadId = this.initPartUpload(initAuthResponse.getSignStr(), fileContentType);

        // Count the number of file parts.
        int partNumber = 1;
        // Buffer
        byte[] fileByte = new byte[bufferSize];
        // Record the read length.
        int readLength = 0;

        // 7. Read the file content and repeat steps 5 and 6 to upload all parts.
        FileInputStream fis = new FileInputStream(file);
        // MD5
        MessageDigest md = MessageDigest.getInstance("MD5");
        while ((readLength = fis.read(fileByte)) > 0) {
            // If the read length is less than the buffer length, copy the data within the valid length and
apply the data to the last part.
            if(readLength < bufferSize) {
                fileByte = Arrays.copyOf(fileByte, readLength);
            }
            // Perform MD5 encoding and then Base64 encoding.
            byte[] digest = md.digest(fileByte);
            String contentMd5 = new String(Base64.encodeBase64(digest));
            System.out.println("The MD5 value of the "+(partNumber)+" part in the file is " + contentMd5);

            // 5. Obtain authorization for multipart upload.
            ShowAssetTempAuthorityResponse partUploadAuthorityResponse =
this.getPartUploadAuthority(vodClient, fileContentType, assetResponse, contentMd5, uploadId, partNumber);

            // 6. Upload parts.
            this.uploadPartFile(partUploadAuthorityResponse.getSignStr(), fileByte, contentMd5);

            // The part number automatically increments by one.
            partNumber++;
        }

        fis.close();

        // 8. Obtain authorization for obtaining uploaded parts.
        ShowAssetTempAuthorityResponse listPartUploadAuthorityResponse =
this.listUploadedPartAuthority(vodClient, assetResponse, uploadId);
        // 9. Obtain uploaded parts.
        String partInfo = this.listUploadedPart(listPartUploadAuthorityResponse.getSignStr());
        // 10. Obtain authorization for merging parts.
        ShowAssetTempAuthorityResponse mergePartUploadAuthorityResponse =
this.mergeUploadedPartAuthority(vodClient, assetResponse, uploadId);
        // 11. Merge uploaded parts.
        this.mergeUploadedPart(mergePartUploadAuthorityResponse.getSignStr(), partInfo);
        // 12. Confirm media asset upload.
        this.confirmUploaded(vodClient, assetResponse);
        System.out.println("Media asset created. assetId:" + assetResponse.getAssetId());
    } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(String.valueOf(e.getHttpStatusCode()));
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
```



```
    * Verify the file and its path.
    * @param filePath
    */
   private File validFile(String filePath){
       if (StringUtils.isEmpty(filePath)) {
           throw new RuntimeException("The file path is null.");
       }
       File file = new File(filePath);
       if (!file.exists()) {
           throw new RuntimeException("The file does not exist.");
       } else if (file.isDirectory()) {
           try {
               throw new RuntimeException(file.getCanonicalPath() + "This is a directory.");
           } catch (IOException e) {
               throw new RuntimeException(e);
           }
       }
       return file;
   }


   /**
    * 1. Construct authentication.
    * @return
    */
   public VodClient createVodClient(){
       ICredential auth = new BasicCredentials()
           .withAk(AK)
           .withSk(SK)
           ;
       return VodClient.newBuilder()
           .withCredential(auth)
           .withRegion(VodRegion.valueOf(REGION_NORTH4))
           .build();
   }

   /**
    * 2. Create a media asset.
    * @param client
    * @param reqbody
    * @return
    */
   public CreateAssetByFileUploadResponse createAssetByFileUpload(VodClient client,
CreateAssetByFileUploadReq reqbody){
       System.out.println("createAssetByFileUpload start");
       CreateAssetByFileUploadRequest request = new CreateAssetByFileUploadRequest();
       // Configure the X-Sdk-Date parameter, which is mandatory for AK/SK authentication.
       request.setXSdkDate(getXSdkDate());
       // Configure upload parameters.
       request.withBody(reqbody);

       // Call the created media asset.
       CreateAssetByFileUploadResponse response = client.createAssetByFileUpload(request);
       System.out.println("createAssetByFileUpload end; createAssetResponse:" + response.toString());
       return response;
   }

   /**
    * Construct request parameters for media asset creation.
    * @param fileName
    * @param videoName
    * @param title
    * @return
    */
   public CreateAssetByFileUploadReq buildFileUploadReq(String fileName, String videoType, String
videoName, String title){
       CreateAssetByFileUploadReq req = new CreateAssetByFileUploadReq();
       req.withVideoName(StringUtils.isNotEmpty(videoName) ? videoName : fileName);
       req.withTitle(StringUtils.isNotEmpty(title) ? title : fileName);
```



```
        // Set the media asset type.
        req.withVideoType(videoType);
        return req;
    }

    /**
     * The X-Sdk-Date parameter indicates the current UTC time (format: yyyyHHddTHHmmssZ), for
example, 20240312T092514Z.
     * @return
     */
    public String getXSdkDate(){
        TimeZone zone = TimeZone.getTimeZone("UTC");
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd'T'HHmmss'Z'");
        return LocalDateTime.now(zone.toZoneId()).format(formatter);
    }

    /**
     * 3. Obtain authorization for initializing an upload task.
     * @param client
     * @param assetResponse
     */
    public ShowAssetTempAuthorityResponse initPartUploadAuthority(VodClient client,
CreateAssetByFileUploadResponse assetResponse, String fileContentType){
        System.out.println("Obtain authorization for initializing an upload task. initPartUploadAuthority
start");
        ShowAssetTempAuthorityRequest request = new ShowAssetTempAuthorityRequest();
        request.setXSdkDate(getXSdkDate());
        // Configure parameters.
        request.withHttpVerb("POST");
        request.withBucket(assetResponse.getTarget().getBucket());
        request.withObjectKey(assetResponse.getTarget().getObject());
        request.withContentType(fileContentType);
        // Send an initialization request.
        ShowAssetTempAuthorityResponse response = client.showAssetTempAuthority(request);
        System.out.println("Obtain authorization for initializing an upload task. initPartUploadAuthority end;
response: " + response.toString());
        return response;
    }

    /**
     * 4. Initialize multipart upload.
     * @param signStr
     * @param contentType
     * @return
     */
    public String initPartUpload(String signStr, String contentType) throws DocumentException {
        System.out.println("Initialize multipart upload. initPartUpload start");
        HttpResponse response = HttpRequest.post(signStr).header("Content-type",contentType).execute();
        System.out.println(response.body());
        Document document = DocumentHelper.parseText(response.body());
        Element root = document.getRootElement();
        Element u = root.element("UploadId");
        System.out.println("Initialize multipart upload. initPartUpload end; UploadId:" + u.getText());
        return u.getText();
    }

    /**
     * 5. Obtain authorization for multipart upload.
     * @param client
     * @param fileContentType
     * @param assetResponse
     * @param contentMd5
     * @param uploadId
     * @param partNumber
     * @return
     */
    public ShowAssetTempAuthorityResponse getPartUploadAuthority(VodClient client, String
fileContentType, CreateAssetByFileUploadResponse assetResponse, String contentMd5, String uploadId, int
partNumber) {
```



```
        System.out.println("Obtain authorization for multipart upload. getPartUploadAuthority start;
partNumber: " + partNumber);
        ShowAssetTempAuthorityRequest request = new ShowAssetTempAuthorityRequest();
        request.setXSdkDate(getXSdkDate());
        request.withHttpVerb("PUT");
        request.withBucket(assetResponse.getTarget().getBucket());
        request.withObjectKey(assetResponse.getTarget().getObject());
        request.withContentType(fileContentType);
        request.withContentMd5(contentMd5);
        request.withUploadId(uploadId);
        request.withPartNumber(partNumber);
        ShowAssetTempAuthorityResponse response = client.showAssetTempAuthority(request);
        System.out.println("Obtain authorization for multipart upload. getPartUploadAuthority end;
partNumber: " + partNumber + "; response" + response.toString());
        return response;
    }

    /**
     * 6. Upload parts.
     * @param signStr
     * @param fileByte
     * @param contentMd5
     */
    public void uploadPartFile(String signStr, byte[] fileByte, String contentMd5) {
        System.out.println("Upload parts. uploadPartFile start");
        HttpResponse response = HttpRequest.put(signStr)
            // The contentMd5 does not need to be escaped.
            .header("Content-MD5", contentMd5)
            .header("Content-Type", "application/octet-stream")
            .body(fileByte).execute();
        System.out.println(response.toString());
        if (response.getStatus() != 200) {
            throw new RuntimeException("Upload parts. uploadPartFile end; upload failed.");
        }
        System.out.println("Upload parts. uploadPartFile end");
    }

    /**
     * 8. Obtain authorization for listing uploaded parts.
     * @param client
     * @param assetResponse
     * @param uploadId
     * @return
     */
    public ShowAssetTempAuthorityResponse listUploadedPartAuthority(VodClient client,
CreateAssetByFileUploadResponse assetResponse, String uploadId){
        System.out.println("Obtain authorization for listing uploaded parts. listUploadedPartAuthority start");
        ShowAssetTempAuthorityRequest request = new ShowAssetTempAuthorityRequest();
        request.setXSdkDate(getXSdkDate());
        request.withHttpVerb("GET");
        request.withBucket(assetResponse.getTarget().getBucket());
        request.withObjectKey(assetResponse.getTarget().getObject());
        request.withUploadId(uploadId);
        ShowAssetTempAuthorityResponse response = client.showAssetTempAuthority(request);
        System.out.println("Obtain authorization for listing uploaded parts. listUploadedPartAuthority end;
response: " + response.toString());
        return response;
    }

    /**
     * 9. Query uploaded parts.
     * @param signStr
     * @return
     */
    public String listUploadedPart(String signStr) throws DocumentException {
        System.out.println("Query uploaded parts. listUploadedPart start");
        int partNumberMarker = 0;
        Element mergerRoot = DocumentHelper.createElement("CompleteMultipartUpload");
        while(true) {
```



```
            // List parts.
            HttpResponse response = HttpRequest.get(signStr + "&part-number-marker=" +
partNumberMarker).execute();
            System.out.println("listUploadedPartResponse:" + response.body());
            Document responseDocument = DocumentHelper.parseText(response.body());
            Element rootResponse = responseDocument.getRootElement();
            List<Element> elementsResponse = rootResponse.elements("Part");
            Element partNumberMarkerElement = rootResponse.element("NextPartNumberMarker");
            for (Element e : elementsResponse) {
                Element te = DocumentHelper.createElement("Part");
                te.add(e.element("PartNumber").createCopy());
                te.add(e.element("ETag").createCopy());
                mergerRoot.add(te);
            }
            partNumberMarker = Integer.valueOf(partNumberMarkerElement.getText());
            if(partNumberMarker % 1000 != 0) {
                break;
            }
        }
        System.out.println(mergerRoot.asXML());
        System.out.println("Query uploaded parts. listUploadedPart end");
        return mergerRoot.asXML();
    }

    /**
     * 10. Obtain authorization for merging parts.
     * @param client
     * @param assetResponse
     * @param uploadId
     * @return
     */
    public ShowAssetTempAuthorityResponse mergeUploadedPartAuthority(VodClient client,
CreateAssetByFileUploadResponse assetResponse, String uploadId) {
        System.out.println("Obtain authorization for merging parts. mergeUploadedPartAuthority start");
        ShowAssetTempAuthorityRequest request = new ShowAssetTempAuthorityRequest();
        request.setXSdkDate(getXSdkDate());
        request.withHttpVerb("POST");
        request.withBucket(assetResponse.getTarget().getBucket());
        request.withObjectKey(assetResponse.getTarget().getObject());
        request.withUploadId(uploadId);
        ShowAssetTempAuthorityResponse response = client.showAssetTempAuthority(request);
        System.out.println("Obtain authorization for merging parts. mergeUploadedPartAuthority end;
response: " + response.toString());
        return response;
    }

    /**
     * 11. Merge parts.
     */
    public void mergeUploadedPart(String signStr, String partInfo) {
        System.out.println("Merge parts. mergeUploadedPart start");
        // Add Content-Type to the request header and set the value to application/xml.
        HttpResponse response = HttpRequest.post(signStr)
            .header("Content-Type","application/xml")
            .body(partInfo)
            .execute();
        System.out.println(response.toString());
        if (response.getStatus() != 200) {
            throw new RuntimeException("Merge parts. mergeUploadedPart end; part merging failed.");
        }
        System.out.println("Merge parts. mergeUploadedPart end");
    }

    /**
     * 12. Confirm the upload completion.
     */
    public void confirmUploaded(VodClient client, CreateAssetByFileUploadResponse assetResponse) {
        System.out.println("Confirm the upload completion. confirmUploaded start");
        ConfirmAssetUploadRequest request = new ConfirmAssetUploadRequest();
```



```
        request.setXSdkDate(getXSdkDate());
        ConfirmAssetUploadReq body = new ConfirmAssetUploadReq();
        body.withStatus(ConfirmAssetUploadReq.StatusEnum.fromValue("CREATED"));
        body.withAssetId(assetResponse.getAssetId());
        request.withBody(body);
        ConfirmAssetUploadResponse response = client.confirmAssetUpload(request);
        System.out.println("Uploaded, assetId:" + response.toString());
    }

}
```

19.7.2 POM Files

Example of the pom.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.huaweicloud.vod</groupId>
    <artifactId>partUploadDemo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>com.huaweicloud.sdk</groupId>
            <artifactId>huaweicloud-sdk-vod</artifactId>
            <version>3.1.93</version>
        </dependency>
        <dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.8.23</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.9</version>
        </dependency>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.15</version>
        </dependency>
        <dependency>
            <groupId>org.dom4j</groupId>
            <artifactId>dom4j</artifactId>
            <version>2.1.4</version>
        </dependency>
    </dependencies>
</project>
```

19.7.3 JavaScript

Sample code of multipart upload using JavaScript:

```
<!DOCTYPE html>
<html>
    <head>
```



```
    <script>
        // Token authentication
        var token = "";
        var projectId = "";
        const region = "cn-north-4";

        var urlList;

        function initUrl(projectId){
            let url = {
                // API for creating a media asset.
                createAssetUrl : "https://vod."+region+".myhuaweicloud.com/v1.0/"+projectId+"/asset",
                // API for obtaining authorization.
                initAuthUrl : "https://vod."+region+".myhuaweicloud.com/v1.0/"+projectId+"/asset/authority",
                // API for confirming media asset upload.
                confirmUploadedUrl : "https://vod."+region+".myhuaweicloud.com/v1.0/"+projectId+"/asset/
status/uploaded"
            }
            return url;
        }


        // The file part size is 1 MB (adjustable).
        const bufferSize = 1024 * 1024;

        // Start upload.
        function startUpload(){
            token = document.getElementById("token").value;
            projectId = document.getElementById("projectId").value;

            urlList = initUrl(projectId);

            let files = document.getElementById("file").files;
            let file = files[0];

            // An MP4 file is used as an example. For details about other formats, see the official website.
            let fileType = "MP4";
            let fileContentType = "video/mp4";

            // 1. Set the request header. Token authentication is used as an example.
            let headers = {
                "X-Auth-Token": token,
                "Content-Type": "application/json"
            }

            // 2. Create a VOD media asset.
            let assetRsp = createAsset(file.name, file.name, fileType, headers);

            // 3. Obtain authorization for initializing an upload task.
            let initAuthResponse = getInitAuth(assetRsp, fileContentType, headers);

            // 4. Initialize the multipart upload task.
            let uploadId = initPartUpload(initAuthResponse, assetRsp, fileContentType);

            // 5. Count the number of file parts.
            let partNumber = 1;
            // 6. Specify the location where file data is read.
            let position = 0;

            let blobSlice = File.prototype.mozSlice || File.prototype.webkitSlice || File.prototype.slice;
            // 7. Repeat steps 5 to 6 to upload parts.
            readAndUpload(file, blobSlice, position, partNumber, assetRsp, fileContentType, uploadId,
headers,
                function(assetRsp, uploadId, headers){
                    // 8. Obtain authorization for listing uploaded parts.
                    let listAuthResp = listUploadedPartAuthority(assetRsp, uploadId, headers);
                    // 9. List uploaded parts.
                    let partInfo = listUploadedPart(listAuthResp.sign_str, assetRsp, uploadId);
                    // 10. Obtain authorization for merging parts.
```



```
                let mergeAuthResp = mergeUploadedPartAuthority(assetRsp, uploadId, headers);
                // 11. Merge parts.
                mergeUploadedPart(mergeAuthResp.sign_str, partInfo, assetRsp, uploadId);
                // 12. Confirm the upload.
                confirmUploaded(assetRsp.asset_id, headers);
                alert("Uploaded;assetId:" + assetRsp.asset_id);
            }
        );
    }

    // 2. Create a VOD media asset.
    function createAsset(title, videoName, videoType, headers){
        let body = {
            "title":title,
            "video_name":videoName,
            "video_type":videoType
        }
        let resp = sendRequest("POST", urlList.createAssetUrl, JSON.stringify(body), headers);
        return JSON.parse(resp);
    }

    // 3. Obtain authorization for initializing an upload task.
    function getInitAuth(assetRsp, fileContentType, headers){
        let params = {
            "http_verb"     : "POST",
            "content_type"  : fileContentType,
            "bucket"        : assetRsp.target.bucket,
            "object_key"    : assetRsp.target.object
        }
        let temp = "?";
        for(let e in params){
            temp += e + "=" + params[e] + "&";
        }
        let resp = sendRequest("GET", urlList.initAuthUrl + temp, null, headers);
        return JSON.parse(resp);
    }

    // 4. Initialize the multipart upload and upload ID is returned.
    function initPartUpload(signStr, assetRsp, contentType){
        let initUrl = "https://" + assetRsp.target.bucket + ".obs." + region + ".myhuaweicloud.com/"
            + assetRsp.target.object + "?uploads&" + signStr.sign_str;
        let resp = sendRequest("POST", initUrl, null, {"Content-Type":contentType});
        let domParser = new DOMParser();
        let dom = domParser.parseFromString(resp, "text/xml");
        return dom.getElementsByTagName("UploadId")[0].firstChild.nodeValue;
    }

    // 5. Obtain authorization for multipart upload.
    function getPartUploadAuthority(assetRsp, fileContentType, uploadId, contentMd5, partNumber,
headers){
        let params = {
            "http_verb"     : "PUT",
            "content_type"  : fileContentType,
            "bucket"        : assetRsp.target.bucket,
            "object_key"    : assetRsp.target.object,
            "content_md5"   : encodeURIComponent(contentMd5),// There are special characters, which
should be escaped.
            "upload_id"     : uploadId,
            "part_number"   : partNumber
        }
        let temp = "?";
        for(let e in params){
            temp += e + "=" + params[e] + "&";
        }
        let resp = sendRequest("GET", urlList.initAuthUrl + temp, null, headers);
        return JSON.parse(resp);
    }

    // 6. Upload parts.
```



```
        function uploadPartFile(uploadAuth, assetRsp, contentMd5, partNumber, uploadId, content)
{
            let url = "https://" + assetRsp.target.bucket + ".obs." + region + ".myhuaweicloud.com/"
                + assetRsp.target.object + "?partNumber="+partNumber+"&uploadId=" + uploadId + "&" +
uploadAuth.sign_str;
            let headers = {
                "Content-Type"  : "application/octet-stream",
                "Content-MD5"   : contentMd5
            }
            let resp = sendRequest("PUT", url, content, headers);
        }

        // 7. Repeat steps 5 to 6 to upload parts.
        function readAndUpload(file, blobSlice, position, partNum, assetRsp, fileContentType, uploadId,
headers, afterUpload){
            let fileReader = new FileReader();
            fileReader.onload = function(e) {
                // Obtain the MD5 value of a file part.
                let md5 = md5_min(e.target.result);
                // Convert the MD5 value to a byte array.
                let bmd5 = Str2Bytes(md5);
                // Encode the MD5 byte array using Base64.
                let content_md5 = base64js_min.fromByteArray(bmd5);

                // 5. Obtain authorization for multipart upload.
                let uploadAuth = getPartUploadAuthority(assetRsp, fileContentType, uploadId, content_md5,
partNum, headers);
                // 6. Upload parts.
                uploadPartFile(uploadAuth, assetRsp, content_md5, partNum, uploadId, e.target.result);

                partNum++;

                // Check whether the data read is complete. If not, continue the upload.
                if(position + bufferSize < file.size){
                    readAndUpload(file, blobSlice, position + bufferSize, partNum, assetRsp, fileContentType,
uploadId, headers, afterUpload);
                } else {
                    // After the upload is complete, perform the subsequent steps.
                    afterUpload(assetRsp, uploadId, headers);
                }
            }
            fileReader.readAsArrayBuffer(blobSlice.call(file, position, position + bufferSize));
        }

        // 8. Obtain authorization for listing uploaded parts.
        function listUploadedPartAuthority(assetRsp, uploadId, headers){
            let params = {
                "http_verb"     : "GET",
                "bucket"        : assetRsp.target.bucket,
                "object_key"    : assetRsp.target.object,
                "upload_id"     : uploadId
            }
            let temp = "?";
            for(let e in params){
                temp += e + "=" + params[e] + "&";
            }
            let resp = sendRequest("GET", urlList.initAuthUrl + temp, null, headers);
            return JSON.parse(resp);
        }

        // 9. Query uploaded parts.
        function listUploadedPart(signStr, assetRsp, uploadId){
            let url = "https://" + assetRsp.target.bucket + ".obs." + region + ".myhuaweicloud.com/"
                + assetRsp.target.object + "?" + signStr + "&uploadId=" + uploadId;
            let nextPartNumberMarker = 0;

            let result = "<CompleteMultipartUpload>";
            let domParser = new DOMParser();
            while(true) {
```



```
                let resp = sendRequest("GET" , url + "&part-number-marker=" + nextPartNumberMarker, null,
{});
                let dom = domParser.parseFromString(resp, "text/xml");
                let part = dom.getElementsByTagName("Part");

                // Construct parameters for merging parts.
                for(let i = 0;i < part.length;i++){
                    let ele = part[i];
                    let num = ele.getElementsByTagName("PartNumber")[0].firstChild.nodeValue;
                    let tag = ele.getElementsByTagName("ETag")[0].firstChild.nodeValue;
                    result += "<Part>" +
                            "<PartNumber>" + num + "</PartNumber>" +
                            "<ETag>" + tag + "</ETag>" +
                        "</Part>"
                    ;
                }

                nextPartNumberMarker = Number(dom.getElementsByTagName("NextPartNumberMarker")
[0].firstChild.nodeValue);

                if(nextPartNumberMarker % 1000 != 0) {
                    break;
                }
            }
            result += "</CompleteMultipartUpload>";
            return result;
        }

        // 10. Obtain authorization for merging parts.
        function mergeUploadedPartAuthority(assetRsp, uploadId, headers){
            let params = {
                "http_verb"     : "POST",
                "bucket"        : assetRsp.target.bucket,
                "object_key"    : assetRsp.target.object,
                "upload_id"     : uploadId
            }
            let temp = "?";
            for(let e in params){
                temp += e + "=" + params[e] + "&";
            }
            let resp = sendRequest("GET", urlList.initAuthUrl + temp, null, headers);
            return JSON.parse(resp);
        }

        // 11. Merge parts.
        function mergeUploadedPart(signStr, partInfo, assetRsp, uploadId){
            let url = "https://" + assetRsp.target.bucket + ".obs." + region + ".myhuaweicloud.com/"
                + assetRsp.target.object + "?" + signStr + "&uploadId=" + uploadId;
            let resp = sendRequest("POST" , url, partInfo, {"Content-Type":"application/xml"});
        }

        // 12. Confirm the upload.
        function confirmUploaded(assetId, headers){
            let body = {
                "asset_id": assetId,
                "status":"CREATED"
            };
            let resp = sendRequest("POST", urlList.confirmUploadedUrl, JSON.stringify(body), headers);
            return console.log(resp);
        }

        // Send a request and wait for result return. Change the content based on the actual framework.
        function sendRequest(method, url, data, headers){
            var xhr = new XMLHttpRequest();
            xhr.open(method, url, false);
            for(let i in headers){
                xhr.setRequestHeader(i, headers[i]);
            }
            xhr.send(data);
```



```
            return xhr.responseText;
        }

        // Base64, which can be replaced based on the actual frontend framework.
        var base64js_min = createCommonjsModule(function(e, t) {
            e.exports = function o(n, i, a) {
                function s(t, e) {
                    if (!i[t]) {
                        if (!n[t]) {
                            var r = "function" == typeof commonjsRequire && commonjsRequire;
                            if (!e && r)
                                return r(t, !0);
                            if (l)
                                return l(t, !0);
                            throw (e = new Error("Cannot find module '" + t + "'")).code =
"MODULE_NOT_FOUND",
                                e
                        }
                        r = i[t] = {
                            exports: {}
                        },
                        n[t][0].call(r.exports, function(e) {
                            return s(n[t][1][e] || e)
                        }, r, r.exports, o, n, i, a)
                    }
                    return i[t].exports
                }
                for (var l = "function" == typeof commonjsRequire && commonjsRequire, e = 0; e < a.length; e
++)
                    s(a[e]);
                return s
            }({
                "/": [function(e, t, r) {
                    r.byteLength = function(e) {
                        var e = d(e)
                        , t = e[0]
                        , e = e[1];
                        return 3 * (t + e) / 4 - e
                    }
                    ,
                    r.toByteArray = function(e) {
                        for (var t, r = d(e), o = r[0], r = r[1], n = new u(function(e, t) {
                            return 3 * (e + t) / 4 - t
                        }(o, r)), i = 0, a = 0 < r ? o - 4 : o, s = 0; s < a; s += 4)
                            t = l[e.charCodeAt(s)] << 18 | l[e.charCodeAt(s + 1)] << 12 | l[e.charCodeAt(s + 2)] <<
6 | l[e.charCodeAt(s + 3)],
                            n[i++] = t >> 16 & 255,
                            n[i++] = t >> 8 & 255,
                            n[i++] = 255 & t;
                        2 === r && (t = l[e.charCodeAt(s)] << 2 | l[e.charCodeAt(s + 1)] >> 4,
                        n[i++] = 255 & t);
                        1 === r && (t = l[e.charCodeAt(s)] << 10 | l[e.charCodeAt(s + 1)] << 4 | l[e.charCodeAt(s
+ 2)] >> 2,
                        n[i++] = t >> 8 & 255,
                        n[i++] = 255 & t);
                        return n
                    }
                    ,
                    r.fromByteArray = function(e) {
                        for (var t, r = e.length, o = r % 3, n = [], i = 0, a = r - o; i < a; i += 16383)
                            n.push(function(e, t, r) {
                                for (var o, n = [], i = t; i < r; i += 3)
                                    o = (e[i] << 16 & 16711680) + (e[i + 1] << 8 & 65280) + (255 & e[i + 2]),
                                    n.push(function(e) {
                                        return s[e >> 18 & 63] + s[e >> 12 & 63] + s[e >> 6 & 63] + s[63 & e]
                                    }(o));
                                return n.join("")
                            }(e, i, a < i + 16383 ? a : i + 16383));
                        1 == o ? (t = e[r - 1],
```



```
                    n.push(s[t >> 2] + s[t << 4 & 63] + "==")) : 2 == o && (t = (e[r - 2] << 8) + e[r - 1],
                    n.push(s[t >> 10] + s[t >> 4 & 63] + s[t << 2 & 63] + "="));
                    return n.join("")
                }
                ;
                for (var s = [], l = [], u = "undefined" != typeof Uint8Array ? Uint8Array : Array, o =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/", n = 0, i = o.length; n < i; +
+n)
                    s[n] = o[n],
                    l[o.charCodeAt(n)] = n;
                function d(e) {
                    var t = e.length;
                    if (0 < t % 4)
                        throw new Error("Invalid string. Length must be a multiple of 4");
                    e = e.indexOf("="),
                    t = (e = -1 === e ? t : e) === t ? 0 : 4 - e % 4;
                    return [e, t]
                }
                l["-".charCodeAt(0)] = 62,
                l["_".charCodeAt(0)] = 63
            }
            , {}]
        }, {}, [])("/")
    })
    // MD5, which can be replaced based on the actual frontend framework.
    , md5_min = createCommonjsModule(function(module) {
        !function() {
            function t(e) {
                e ? (d[0] = d[16] = d[1] = d[2] = d[3] = d[4] = d[5] = d[6] = d[7] = d[8] = d[9] = d[10] =
d[11] = d[12] = d[13] = d[14] = d[15] = 0,
                this.blocks = d,
                this.buffer8 = l) : a ? (e = new ArrayBuffer(68),
                this.buffer8 = new Uint8Array(e),
                this.blocks = new Uint32Array(e)) : this.blocks = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                this.h0 = this.h1 = this.h2 = this.h3 = this.start = this.bytes = this.hBytes = 0,
                this.finalized = this.hashed = !1,
                this.first = !0
            }
            var r = "input is invalid type", e = "object" == ("undefined" == typeof window ? "undefined" :
_typeof(window)), i = e ? window : {}, s = (i.JS_MD5_NO_WINDOW && (e = !1),
            !e && "object" == ("undefined" == typeof self ? "undefined" : _typeof(self))), h = !
i.JS_MD5_NO_NODE_JS && "object" == ("undefined" == typeof process ? "undefined" : _typeof(process))
&& process.versions && process.versions.node, f = (h ? i = commonjsGlobal : s && (i = self),
            !i.JS_MD5_NO_COMMON_JS && module.exports), o = !1, a = !i.JS_MD5_NO_ARRAY_BUFFER
&& "undefined" != typeof ArrayBuffer, n = "0123456789abcdef".split(""), u = [128, 32768, 8388608,
-2147483648], y = [0, 8, 16, 24], c = ["hex", "array", "digest", "buffer", "arrayBuffer", "base64"], p =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/".split(""), d = [], l, A, l, d, b
= (a && (A = new ArrayBuffer(68),
            l = new Uint8Array(A),
            d = new Uint32Array(A)),
            !i.JS_MD5_NO_NODE_JS && Array.isArray || (Array.isArray = function(e) {
                return "[object Array]" === Object.prototype.toString.call(e)
            }
            ),
            !a || !i.JS_MD5_NO_ARRAY_BUFFER_IS_VIEW && ArrayBuffer.isView || (ArrayBuffer.isView =
function(e) {
                return "object" == _typeof(e) && e.buffer && e.buffer.constructor === ArrayBuffer
            }
            ),
            function(r) {
                return function(e) {
                    return new t(!0).update(e)[r]()
                }
            }
            ), v = function() {
                var r = b("hex");
                (r = h ? w(r) : r).create = function() {
                    return new t
                }
```



```
                    ,
                    r.update = function(e) {
                        return r.create().update(e)
                    }
                    ;
                    for (var e = 0; e < c.length; ++e) {
                        var o = c[e];
                        r[o] = b(o)
                    }
                    return r
                }, w = function w(t) {
                    var e = eval("require('crypto')")
                    , i = eval("require('buffer').Buffer")
                    , s = function(o) {
                        if ("string" == typeof o)
                            return e.createHash("md5").update(o, "utf8").digest("hex");
                        if (null == o)
                            throw r;
                        return o.constructor === ArrayBuffer && (o = new Uint8Array(o)),
                        Array.isArray(o) || ArrayBuffer.isView(o) || o.constructor === i ?
e.createHash("md5").update(new i(o)).digest("hex") : t(o)
                    };
                    return s
                }, _ = (t.prototype.update = function(e) {
                    if (!this.finalized) {
                        var t, o = _typeof(e);
                        if ("string" !== o) {
                            if ("object" !== o)
                                throw r;
                            if (null === e)
                                throw r;
                            if (a && e.constructor === ArrayBuffer)
                                e = new Uint8Array(e);
                            else if (!(Array.isArray(e) || a && ArrayBuffer.isView(e)))
                                throw r;
                            t = !0
                        }
                        for (var n, i, s = 0, l = e.length, u = this.blocks, d = this.buffer8; s < l; ) {
                            if (this.hashed && (this.hashed = !1,
                            u[0] = u[16],
                            u[16] = u[1] = u[2] = u[3] = u[4] = u[5] = u[6] = u[7] = u[8] = u[9] = u[10] = u[11] =
u[12] = u[13] = u[14] = u[15] = 0),
                            t)
                                if (a)
                                    for (i = this.start; s < l && i < 64; ++s)
                                        d[i++] = e[s];
                                else
                                    for (i = this.start; s < l && i < 64; ++s)
                                        u[i >> 2] |= e[s] << y[3 & i++];
                            else if (a)
                                for (i = this.start; s < l && i < 64; ++s)
                                    (n = e.charCodeAt(s)) < 128 ? d[i++] = n : (n < 2048 ? d[i++] = 192 | n >> 6 : (n
< 55296 || 57344 <= n ? d[i++] = 224 | n >> 12 : (n = 65536 + ((1023 & n) << 10 | 1023 & e.charCodeAt(+
+s)),
                                    d[i++] = 240 | n >> 18,
                                    d[i++] = 128 | n >> 12 & 63),
                                    d[i++] = 128 | n >> 6 & 63),
                                    d[i++] = 128 | 63 & n);
                            else
                                for (i = this.start; s < l && i < 64; ++s)
                                    (n = e.charCodeAt(s)) < 128 ? u[i >> 2] |= n << y[3 & i++] : (n < 2048 ? u[i >> 2]
|= (192 | n >> 6) << y[3 & i++] : (n < 55296 || 57344 <= n ? u[i >> 2] |= (224 | n >> 12) << y[3 & i++] : (n =
65536 + ((1023 & n) << 10 | 1023 & e.charCodeAt(++s)),
                                    u[i >> 2] |= (240 | n >> 18) << y[3 & i++],
                                    u[i >> 2] |= (128 | n >> 12 & 63) << y[3 & i++]),
                                    u[i >> 2] |= (128 | n >> 6 & 63) << y[3 & i++]),
                                    u[i >> 2] |= (128 | 63 & n) << y[3 & i++]);
                            this.lastByteIndex = i,
                            this.bytes += i - this.start,
```



```
                        64 <= i ? (this.start = i - 64,
                        this.hash(),
                        this.hashed = !0) : this.start = i
                    }
                    return 4294967295 < this.bytes && (this.hBytes += this.bytes / 4294967296 << 0,
                    this.bytes = this.bytes % 4294967296),
                    this
                }
            }
            ,
            t.prototype.finalize = function() {
                var e, t;
                this.finalized || (this.finalized = !0,
                (e = this.blocks)[(t = this.lastByteIndex) >> 2] |= u[3 & t],
                56 <= t && (this.hashed || this.hash(),
                e[0] = e[16],
                e[16] = e[1] = e[2] = e[3] = e[4] = e[5] = e[6] = e[7] = e[8] = e[9] = e[10] = e[11] = e[12]
= e[13] = e[14] = e[15] = 0),
                e[14] = this.bytes << 3,
                e[15] = this.hBytes << 3 | this.bytes >>> 29,
                this.hash())
            }
            ,
            t.prototype.hash = function() {
                var e, t, r, o, n, i = this.blocks, a = this.first ? ((a = ((e = ((e = i[0] - 680876937) << 7 | e >>>
25) - 271733879 << 0) ^ (t = ((t = (-271733879 ^ (r = ((r = (-1732584194 ^ 2004318071 & e) + i[1] -
117830708) << 12 | r >>> 20) + e << 0) & (-271733879 ^ e)) + i[2] - 1126478375) << 17 | t >>> 15) + r <<
0) & (r ^ e)) + i[3] - 1316259209) << 22 | a >>> 10) + t << 0 : (e = this.h0,
                a = this.h1,
                t = this.h2,
                ((a += ((e = ((e += ((r = this.h3) ^ a & (t ^ r)) + i[0] - 680876936) << 7 | e >>> 25) + a <<
0) ^ (t = ((t += (a ^ (r = ((r += (t ^ e & (a ^ t)) + i[1] - 389564586) << 12 | r >>> 20) + e << 0) & (e ^ a)) +
i[2] + 606105819) << 17 | t >>> 15) + r << 0) & (r ^ e)) + i[3] - 1044525330) << 22 | a >>> 10) + t << 0);
                a = ((a += ((e = ((e += (r ^ a & (t ^ r)) + i[4] - 176418897) << 7 | e >>> 25) + a << 0) ^ (t
= ((t += (a ^ (r = ((r += (t ^ e & (a ^ t)) + i[5] + 1200080426) << 12 | r >>> 20) + e << 0) & (e ^ a)) + i[6] -
1473231341) << 17 | t >>> 15) + r << 0) & (r ^ e)) + i[7] - 45705983) << 22 | a >>> 10) + t << 0,
                a = ((a += ((e = ((e += (r ^ a & (t ^ r)) + i[8] + 1770035416) << 7 | e >>> 25) + a << 0) ^ (t
= ((t += (a ^ (r = ((r += (t ^ e & (a ^ t)) + i[9] - 1958414417) << 12 | r >>> 20) + e << 0) & (e ^ a)) + i[10]
- 42063) << 17 | t >>> 15) + r << 0) & (r ^ e)) + i[11] - 1990404162) << 22 | a >>> 10) + t << 0,
                a = ((a += ((e = ((e += (r ^ a & (t ^ r)) + i[12] + 1804603682) << 7 | e >>> 25) + a << 0) ^
(t = ((t += (a ^ (r = ((r += (t ^ e & (a ^ t)) + i[13] - 40341101) << 12 | r >>> 20) + e << 0) & (e ^ a)) +
i[14] - 1502002290) << 17 | t >>> 15) + r << 0) & (r ^ e)) + i[15] + 1236535329) << 22 | a >>> 10) + t << 0,
                a = ((a += ((r = ((r += (a ^ t & ((e = ((e += (t ^ r & (a ^ t)) + i[1] - 165796510) << 5 | e
>>> 27) + a << 0) ^ a)) + i[6] - 1069501632) << 9 | r >>> 23) + e << 0) ^ e & ((t = ((t += (e ^ a & (r ^ e)) +
i[11] + 643717713) << 14 | t >>> 18) + r << 0) ^ r)) + i[0] - 373897302) << 20 | a >>> 12) + t << 0,
                a = ((a += ((r = ((r += (a ^ t & ((e = ((e += (t ^ r & (a ^ t)) + i[5] - 701558691) << 5 | e
>>> 27) + a << 0) ^ a)) + i[10] + 38016083) << 9 | r >>> 23) + e << 0) ^ e & ((t = ((t += (e ^ a & (r ^ e)) +
i[15] - 660478335) << 14 | t >>> 18) + r << 0) ^ r)) + i[4] - 405537848) << 20 | a >>> 12) + t << 0,
                a = ((a += ((r = ((r += (a ^ t & ((e = ((e += (t ^ r & (a ^ t)) + i[9] + 568446438) << 5 | e
>>> 27) + a << 0) ^ a)) + i[14] - 1019803690) << 9 | r >>> 23) + e << 0) ^ e & ((t = ((t += (e ^ a & (r ^ e))
+ i[3] - 187363961) << 14 | t >>> 18) + r << 0) ^ r)) + i[8] + 1163531501) << 20 | a >>> 12) + t << 0,
                a = ((a += ((r = ((r += (a ^ t & ((e = ((e += (t ^ r & (a ^ t)) + i[13] - 1444681467) << 5 | e
>>> 27) + a << 0) ^ a)) + i[2] - 51403784) << 9 | r >>> 23) + e << 0) ^ e & ((t = ((t += (e ^ a & (r ^ e)) +
i[7] + 1735328473) << 14 | t >>> 18) + r << 0) ^ r)) + i[12] - 1926607734) << 20 | a >>> 12) + t << 0,
                a = ((a += ((n = (r = ((r += ((o = a ^ t) ^ (e = ((e += (o ^ r) + i[5] - 378558) << 4 | e >>>
28) + a << 0)) + i[8] - 2022574463) << 11 | r >>> 21) + e << 0) ^ e) ^ (t = ((t += (n ^ a) + i[11] +
1839030562) << 16 | t >>> 16) + r << 0)) + i[14] - 35309556) << 23 | a >>> 9) + t << 0,
                a = ((a += ((n = (r = ((r += ((o = a ^ t) ^ (e = ((e += (o ^ r) + i[1] - 1530992060) << 4 | e
>>> 28) + a << 0)) + i[4] + 1272893353) << 11 | r >>> 21) + e << 0) ^ e) ^ (t = ((t += (n ^ a) + i[7] -
155497632) << 16 | t >>> 16) + r << 0)) + i[10] - 1094730640) << 23 | a >>> 9) + t << 0,
                a = ((a += ((n = (r = ((r += ((o = a ^ t) ^ (e = ((e += (o ^ r) + i[13] + 681279174) << 4 | e
>>> 28) + a << 0)) + i[0] - 358537222) << 11 | r >>> 21) + e << 0) ^ e) ^ (t = ((t += (n ^ a) + i[3] -
722521979) << 16 | t >>> 16) + r << 0)) + i[6] + 76029189) << 23 | a >>> 9) + t << 0,
                a = ((a += ((n = (r = ((r += ((o = a ^ t) ^ (e = ((e += (o ^ r) + i[9] - 640364487) << 4 | e
>>> 28) + a << 0)) + i[12] - 421815835) << 11 | r >>> 21) + e << 0) ^ e) ^ (t = ((t += (n ^ a) + i[15] +
530742520) << 16 | t >>> 16) + r << 0)) + i[2] - 995338651) << 23 | a >>> 9) + t << 0,
                a = ((a += ((r = ((r += (a ^ ((e = ((e += (t ^ (a | ~r)) + i[0] - 198630844) << 6 | e >>> 26) +
a << 0) | ~t)) + i[7] + 1126891415) << 10 | r >>> 22) + e << 0) ^ ((t = ((t += (e ^ (r | ~a)) + i[14] -
1416354905) << 15 | t >>> 17) + r << 0) | ~e)) + i[5] - 57434055) << 21 | a >>> 11) + t << 0,
```



```
                    a = ((a += ((r = ((r += (a ^ ((e = ((e += (t ^ (a | ~r)) + i[12] + 1700485571) << 6 | e >>>
26) + a << 0) | ~t)) + i[3] - 1894986606) << 10 | r >>> 22) + e << 0) ^ ((t = ((t += (e ^ (r | ~a)) + i[10] -
1051523) << 15 | t >>> 17) + r << 0) | ~e)) + i[1] - 2054922799) << 21 | a >>> 11) + t << 0,
                    a = ((a += ((r = ((r += (a ^ ((e = ((e += (t ^ (a | ~r)) + i[8] + 1873313359) << 6 | e >>> 26)
+ a << 0) | ~t)) + i[15] - 30611744) << 10 | r >>> 22) + e << 0) ^ ((t = ((t += (e ^ (r | ~a)) + i[6] -
1560198380) << 15 | t >>> 17) + r << 0) | ~e)) + i[13] + 1309151649) << 21 | a >>> 11) + t << 0,
                    a = ((a += ((r = ((r += (a ^ ((e = ((e += (t ^ (a | ~r)) + i[4] - 145523070) << 6 | e >>> 26) +
a << 0) | ~t)) + i[11] - 1120210379) << 10 | r >>> 22) + e << 0) ^ ((t = ((t += (e ^ (r | ~a)) + i[2] +
718787259) << 15 | t >>> 17) + r << 0) | ~e)) + i[9] - 343485551) << 21 | a >>> 11) + t << 0,
                    this.first ? (this.h0 = e + 1732584193 << 0,
                    this.h1 = a - 271733879 << 0,
                    this.h2 = t - 1732584194 << 0,
                    this.h3 = r + 271733878 << 0,
                    this.first = !1) : (this.h0 = this.h0 + e << 0,
                    this.h1 = this.h1 + a << 0,
                    this.h2 = this.h2 + t << 0,
                    this.h3 = this.h3 + r << 0)
                }
                ,
                t.prototype.hex = function() {
                    this.finalize();
                    var e = this.h0
                    , t = this.h1
                    , r = this.h2
                    , o = this.h3;
                    return n[e >> 4 & 15] + n[15 & e] + n[e >> 12 & 15] + n[e >> 8 & 15] + n[e >> 20 & 15] +
n[e >> 16 & 15] + n[e >> 28 & 15] + n[e >> 24 & 15] + n[t >> 4 & 15] + n[15 & t] + n[t >> 12 & 15] + n[t
>> 8 & 15] + n[t >> 20 & 15] + n[t >> 16 & 15] + n[t >> 28 & 15] + n[t >> 24 & 15] + n[r >> 4 & 15] + n[15
& r] + n[r >> 12 & 15] + n[r >> 8 & 15] + n[r >> 20 & 15] + n[r >> 16 & 15] + n[r >> 28 & 15] + n[r >> 24
& 15] + n[o >> 4 & 15] + n[15 & o] + n[o >> 12 & 15] + n[o >> 8 & 15] + n[o >> 20 & 15] + n[o >> 16 &
15] + n[o >> 28 & 15] + n[o >> 24 & 15]
                }
                ,
                t.prototype.toString = t.prototype.hex,
                t.prototype.digest = function() {
                    this.finalize();
                    var e = this.h0
                    , t = this.h1
                    , r = this.h2
                    , o = this.h3;
                    return [255 & e, e >> 8 & 255, e >> 16 & 255, e >> 24 & 255, 255 & t, t >> 8 & 255, t >> 16
& 255, t >> 24 & 255, 255 & r, r >> 8 & 255, r >> 16 & 255, r >> 24 & 255, 255 & o, o >> 8 & 255, o >> 16 &
255, o >> 24 & 255]
                }
                ,
                t.prototype.array = t.prototype.digest,
                t.prototype.arrayBuffer = function() {
                    this.finalize();
                    var e = new ArrayBuffer(16)
                    , t = new Uint32Array(e);
                    return t[0] = this.h0,
                    t[1] = this.h1,
                    t[2] = this.h2,
                    t[3] = this.h3,
                    e
                }
                ,
                t.prototype.buffer = t.prototype.arrayBuffer,
                t.prototype.base64 = function() {
                    for (var e, t, r, o = "", n = this.array(), i = 0; i < 15; )
                        e = n[i++],
                        t = n[i++],
                        r = n[i++],
                        o += p[e >>> 2] + p[63 & (e << 4 | t >>> 4)] + p[63 & (t << 2 | r >>> 6)] + p[63 & r];
                    return e = n[i],
                    o + (p[e >>> 2] + p[e << 4 & 63] + "==")
                }
                ,
                v());
```



```
                f ? module.exports = _ : i.md5 = _
            }()
        });

        function _typeof(e) {
            return (_typeof = "function" == typeof Symbol && "symbol" == typeof Symbol.iterator ?
function(e) {
                return typeof e
            }
            : function(e) {
                return e && "function" == typeof Symbol && e.constructor === Symbol && e !==
Symbol.prototype ? "symbol" : typeof e
            }
            )(e)
        }

        function createCommonjsModule(e, t) {
            return e(t = {
                exports: {}
            }, t.exports),
            t.exports
        }

        function Str2Bytes(e) {
            var t = 0
            , r = e.length;
            if (r % 2 != 0)
                return null;
            r /= 2;
            for (var o = new Array, n = 0; n < r; n++) {
                var i = e.substr(t, 2)
                , i = parseInt(i, 16);
                o.push(i),
                t += 2
            }
            return o
        }

    </script>
</head>

<body>
    <p>projectId:   <input type="input" id="projectId"></input></p>
    <p>token:       <input type="input" id="token"></input></p>
    <p><input type="file" id="file"></input></p>
    <button onclick="startUpload()">Upload</button>
</body>
</html>
```

19.7.4 Python

Sample code of multipart upload using Python:

```
import base64
import hashlib
import os
import re

import xml.etree.ElementTree as ET
import requests

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkvod.v1 import *
from huaweicloudsdkvod.v1.region.vod_region import VodRegion


class PartUploadDemo:
    """
    Example of multipart upload
```



```
"""
# Set the buffer size as needed, that is, the size of the file part read each time.
# 1 MB
buffer_size = 1024 * 1024

# Region
region_north4 = "cn-north-4"
region_north1 = "cn-north-1"
region_east2 = "cn-east-2"

region = ""

# AK/SK, which is used for authentication in this example.
ak = ""
sk = ""

def __init__(self):
    pass

def upload_file(self, file_path):
    """
    Multipart upload
    :param file_path: local path of the file
    :type file_path: str
    :return:
    """
    # Verify the file and its path.
    if not self.valid_file(file_path):
        return

    # Obtain the file name.
    filename = os.path.basename(file_path)

    # An MP4 file is used as an example. For details about other formats, see the official website.
    video_type = "MP4"
    file_content_type = "video/mp4"

    print("Start uploading media assets:" + filename)
    # 1. Initialize authentication and obtain vodClient.
    client = self.create_vod_client()

    # 2. Create a VOD media asset.
    asset_response = self.create_asset(client=client,
                                       file_name=filename,
                                       video_type=video_type)

    # 3. Obtain authorization for initializing an upload task.
    init_auth_response = self.init_part_upload_authority(client=client,
                                                         asset_response=asset_response,
                                                         file_content_type=file_content_type)

    # 4. Initialize the upload task.
    upload_id = self.init_part_upload(sign_str=init_auth_response.sign_str,
                                      file_content_type=file_content_type)

    # Count the number of file parts.
    part_number = 1

    # 7. Read the file content and repeat steps 5 and 6 to upload all parts.
    with open(file_path, 'rb') as f:
        for chunk in iter(lambda: f.read(self.buffer_size), b''):
            # Generate content_md5 using MD5 and then Base64.
            md5 = hashlib.md5()
            md5.update(chunk)
            content_md5 = str(base64.b64encode(md5.digest()), 'utf-8')
            # print(content_md5)

            # 5. Obtain authorization for multipart upload.
            upload_auth_response = self.get_part_upload_authority(client=client,
```



```
                                                asset_response=asset_response,
                                                file_content_type=file_content_type,
                                                content_md5=content_md5,
                                                upload_id=upload_id,
                                                part_number=part_number)

            # 6. Upload parts.
            self.upload_part_file(sign_str=upload_auth_response.sign_str,
                                  chunk=chunk,
                                  content_md5=content_md5,
                                  part_number=part_number)

            # The part number automatically increments by one.
            part_number += 1

    # 8. Obtain authorization for obtaining uploaded parts.
    list_part_upload_authority_response = self.list_uploaded_part_authority(client=client,
                                                                             asset_response=asset_response,
                                                                             upload_id=upload_id)

    # 9. Obtain uploaded parts.
    part_info = self.list_uploaded_part(sign_str=list_part_upload_authority_response.sign_str)

    # 10. Obtain authorization for merging parts.
    merge_part_upload_authority_response = self.merge_uploaded_part_authority(client=client,
                                                                              asset_response=asset_response,
                                                                              upload_id=upload_id)

    # 11. Merge uploaded parts.
    self.merge_uploaded_part(sign_str=merge_part_upload_authority_response.sign_str,
                             part_info=part_info)

    # 12. Confirm media asset upload.
    self.confirm_uploaded(client=client, asset_response=asset_response)
    print("Media asset uploaded. assetId:" + asset_response.asset_id)

# Check whether the file exists.
def valid_file(self, file_path):
    valid_result = True
    if not file_path:
        print("The path is empty.")
        valid_result = False
    elif os.path.isdir(file_path):
        print("It is a directory.")
        valid_result = False
    elif not os.path.isfile(file_path):
        print("The file does not exist.")
        valid_result = False
    return valid_result

# 1. Initialize authentication.
def create_vod_client(self):
    print("Initializing authentication...")
    credentials = BasicCredentials(self.ak, self.sk)
    client = VodClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(VodRegion.value_of(self.region)) \
        .build()
    return client

def create_asset(self, client, file_name, video_type):
    """
    2. Create a VOD media asset.
    :param client
    :param file_name: audio/video file name
    :type file_name: str
    :param video_type: uploaded audio/video file format
    :type video_type: str
    """
```



```
        print("create_asset start; ")
        create_asset_request = CreateAssetByFileUploadRequest()
        # Create the minimum set of parameters for media asset creation. For details about other parameters,
see documents on the official website.
        create_asset_request.body = CreateAssetByFileUploadReq(
            video_type=video_type,
            video_name=file_name,
            title=file_name
        )
        # Call the media asset creation method.
        asset_response = client.create_asset_by_file_upload(create_asset_request)
        print("create_asset end")
        return asset_response

    def init_part_upload_authority(self, client, asset_response, file_content_type):
        """
        3. Obtain authorization for initializing an upload task.
        :param client:
        :param asset_response: returned media asset creation result
        :param file_content_type: content-type of a file type, such as video/mp4 for MP4
        :type file_content_type: str
        :return:
        """
        print("Obtain authorization for initializing an upload task. init_part_upload_authority start")
        init_auth_request = ShowAssetTempAuthorityRequest()
        # Configure initialization parameters.
        init_auth_request.http_verb = "POST"
        init_auth_request.bucket = asset_response.target.bucket
        init_auth_request.object_key = asset_response.target.object
        init_auth_request.content_type = file_content_type
        # Send an initialization request.
        init_auth_response = client.show_asset_temp_authority(init_auth_request)
        print("Obtain authorization for initializing an upload task. init_part_upload_authority end")
        return init_auth_response

    def init_part_upload(self, sign_str, file_content_type):
        """
        4. Initialize the upload task.
        :param sign_str: sign_str in the result returned in step 3, which is the URL for initializing the upload
task
        :type sign_str: str
        :param file_content_type: content-type of a file type, such as video/mp4 for MP4
        :type file_content_type: str
        :return: returns upload_id
        """
        print("Initialize multipart upload. init_part_upload start")
        # Send an initialization request.
        init_response = requests.request(method="POST",
                                         url=sign_str,
                                         headers={"Content-Type": file_content_type})
        print(init_response.text)
        # Parse the response to obtain the uploadId.
        root = ET.fromstring(init_response.text)
        namespace_str = root.tag
        match = re.search(r'\{(.*?)}', namespace_str)
        namespace_uri = match.group(1)
        upload_id = root.find("{" + namespace_uri + "}UploadId").text
        print("Initialize multipart upload. init_part_upload end; UploadId:" + upload_id)
        return upload_id

    def get_part_upload_authority(self, client, asset_response, file_content_type, content_md5, upload_id,
part_number):
        """
        5. Obtain authorization for multipart upload.
        :param client:
        :param asset_response: returned media asset creation result
        :param file_content_type: content-type of a file type, such as video/mp4 for MP4
        :type file_content_type: str
        :param content_md5: content-md5 value of the current file part
```



```
        :type content_md5: str
        :param upload_id:
        :type upload_id: str
        :param part_number: part number
        :type part_number: int
        :return:
        """
        print("Obtain authorization for multipart upload. get_part_upload_authority start; partNumber:",
part_number)
        upload_auth_request = ShowAssetTempAuthorityRequest()
        # Configure upload authorization parameters.
        upload_auth_request.http_verb = "PUT"
        upload_auth_request.bucket = asset_response.target.bucket
        upload_auth_request.object_key = asset_response.target.object
        upload_auth_request.content_type = file_content_type
        upload_auth_request.content_md5 = content_md5
        upload_auth_request.upload_id = upload_id
        upload_auth_request.part_number = part_number
        upload_auth_response = client.show_asset_temp_authority(upload_auth_request)
        print(upload_auth_response)
        print("Obtain authorization for multipart upload. get_part_upload_authority end; partNumber:",
part_number)
        return upload_auth_response

    def upload_part_file(self, sign_str, chunk, content_md5, part_number):
        """
        6. Upload parts.
        :param sign_str: sign_str in the result returned in step 5, which is the URL for upload
        :type sign_str: str
        :param chunk: binary data of the current file part
        :type chunk: bytes
        :param content_md5: content-md5 value of the current file part
        :type content_md5: str
        :param part_number: number of the current file part
        :type part_number: int
        :return:
        """
        print("Upload parts. upload_part_file start; partNumber:", part_number)
        # Send a multipart upload request.
        upload_response = requests.request(method="PUT",
                                           url=sign_str,
                                           headers={
                                               "Content-Type": "application/octet-stream",
                                               "Content-MD5": content_md5
                                           },
                                           data=chunk)
        if upload_response.status_code != 200:
            print("Multipart upload end; upload failed! partNumber:", part_number)
            raise Exception("Multipart upload end; upload failed! partNumber:", part_number)
        print("Upload parts. upload_part_file end! partNumber:", part_number)

    def list_uploaded_part_authority(self, client, asset_response, upload_id):
        """
        8. Obtain authorization for obtaining uploaded parts.
        :param client:
        :param asset_response: returned media asset creation result
        :param upload_id:
        :return:
        """
        print("Obtain authorization for listing uploaded parts. list_uploaded_part_authority start")
        # Configure parameters.
        list_upload_part_auth_request = ShowAssetTempAuthorityRequest()
        list_upload_part_auth_request.http_verb = "GET"
        list_upload_part_auth_request.bucket = asset_response.target.bucket
        list_upload_part_auth_request.object_key = asset_response.target.object
        list_upload_part_auth_request.upload_id = upload_id
        list_upload_part_auth_response = client.show_asset_temp_authority(list_upload_part_auth_request)
        print(list_upload_part_auth_response)
        print("Obtain authorization for listing uploaded parts. list_uploaded_part_authority end")
```



```
        return list_upload_part_auth_response

    def list_uploaded_part(self, sign_str):
        """
        9. Obtain uploaded parts.
        :param sign_str: authorized URL returned in step 8
        :type sign_str: str
        :return:
        """
        print("Query uploaded parts. list_uploaded_part start")
        # Query the start number of file parts.
        part_number_marker = 0
        # Assemble the root nodes for merging parts.
        merger_root = ET.Element("CompleteMultipartUpload")
        # Information about a maximum of 1,000 parts can be returned each time. If there are more than
1,000 parts, call the API for listing parts multiple times.
        while True:
            # List parts.
            list_upload_part_auth_response = requests.request(method="GET",
                                                              url=sign_str + "&part-number-marker=" + str(
                                                                  part_number_marker))
            print(list_upload_part_auth_response)

            # Format the response using XML.
            response_document = ET.fromstring(list_upload_part_auth_response.text)
            # Parse the XML content and obtain the XMLNS information.
            namespace_str_m = response_document.tag
            match_m = re.search(r'\{(.*?)}', namespace_str_m)
            namespace_uri_m = match_m.group(1)

            # Check all Part nodes.
            for part in response_document.findall("{" + namespace_uri_m + "}Part"):
                # Obtain PartNumber and ETag.
                part_number_value = part.find("{" + namespace_uri_m + "}PartNumber").text
                e_tag_value = part.find("{" + namespace_uri_m + "}ETag").text

                # Assemble information about merging.
                # Create a Part node under the root node.
                part_node = ET.SubElement(merger_root, "Part")
                # Create PartNumber under the Part node and set PartNumber.
                part_number_node = ET.SubElement(part_node, "PartNumber")
                part_number_node.text = part_number_value
                # Create ETag under the Part node and set ETag.
                e_tag_node = ET.SubElement(part_node, "ETag")
                e_tag_node.text = e_tag_value

            # Find and set the start number of the next list.
            part_number_marker_element = response_document.find("{" + namespace_uri_m +
"}NextPartNumberMarker")
            part_number_marker = int(part_number_marker_element.text)
            # If the part number is not an integer multiple of 1,000, all parts have been obtained.
            if part_number_marker % 1000 != 0:
                break

        part_info = ET.tostring(merger_root, encoding='utf8')
        print(part_info)
        print("Query uploaded parts. list_uploaded_part end")
        return part_info

    def merge_uploaded_part_authority(self, client, asset_response, upload_id):
        """
        10. Obtain authorization for merging parts.
        :param client:
        :param asset_response: returned media asset creation result
        :param upload_id: upload_id
        :type upload_id: str
        :return:
        """
        print("Obtain authorization for merging parts. merge_uploaded_part_authority start")
```



```
        # Configure parameters.
        merger_part_upload_auth_request = ShowAssetTempAuthorityRequest()
        merger_part_upload_auth_request.http_verb = "POST"
        merger_part_upload_auth_request.bucket = asset_response.target.bucket
        merger_part_upload_auth_request.object_key = asset_response.target.object
        merger_part_upload_auth_request.upload_id = upload_id
        merger_part_upload_auth_response =
client.show_asset_temp_authority(merger_part_upload_auth_request)
        print(merger_part_upload_auth_response)
        print("Obtain authorization for merging parts. merge_uploaded_part_authority end")
        return merger_part_upload_auth_response

    def merge_uploaded_part(self, sign_str, part_info):
        """
        11. Merge uploaded parts.
        :param sign_str: URL for authorized merging returned in step 10
        :type sign_str: str
        :param part_info: information about merged parts
        :type part_info: str
        :return:
        """
        print("Merge parts. start")
        # Add Content-Type to the request header and set the value to application/xml.
        merger_part_upload_response = requests.request(method="POST",
                                               url=sign_str,
                                               headers={"Content-Type": "application/xml"},
                                               data=part_info)
        print(merger_part_upload_response)
        if merger_part_upload_response.status_code != 200:
            print("Part merging end; merging parts failed.")
        print("Part merging end")

    # 12. Confirm media asset upload.
    def confirm_uploaded(self, client, asset_response):
        print("Confirm the upload completion. Start")
        confirm_asset_upload_request = ConfirmAssetUploadRequest()
        confirm_asset_upload_request.body = ConfirmAssetUploadReq(status="CREATED",
                                                  asset_id=asset_response.asset_id)
        confirm_asset_upload_response = client.confirm_asset_upload(confirm_asset_upload_request)
        print(confirm_asset_upload_response)


if __name__ == '__main__':
    # Path of the local media asset to be uploaded
    filePath = ""
    partUploadDemo = PartUploadDemo()
    partUploadDemo.ak = ""
    partUploadDemo.sk = ""
    partUploadDemo.region = partUploadDemo.region_north4
    # Upload the media asset.
    partUploadDemo.upload_file(file_path=filePath)
```

19.7.5 Go

Sample code of multipart upload using Go:

```
package main

import (
    "bytes"
    "crypto/md5"
    "encoding/base64"
    "encoding/xml"
    "errors"
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    vod "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/vod/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/vod/v1/model"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/vod/v1/region"
```



```
    "io"
    "net/http"
    "os"
)

// Region
const regionNorth4 = "cn-north-4"
const regionNorth1 = "cn-north-1"
const regionEast2 = "cn-east-2"

// ak/sk
const ak = ""
const sk = ""

// Set the buffer size as needed, that is, the size of the file part read each time.
// 1M
const bufferSize = 1024 * 1024

/*
Example of multipart upload
*/
func main() {
    // Path of the local media asset to be uploaded
    filePath := ""

    // Upload the media asset.
    partUpload(filePath)
}

/*
Multipart upload
filePath: local path where the file to be uploaded is
*/
func partUpload(filePath string) {
    // Verify the file and its path.
    fileInfo := validFile(filePath)
    fileName := fileInfo.Name()
    fmt.Println(fileName)

    file, err := os.Open(filePath)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }
    defer func(file *os.File) {
        err := file.Close()
        if err != nil {
            panic(err)
        }
    }(file)

    // An MP4 file is used as an example. For details about other formats, see the official website.
    fileType := "MP4"
    fileContentType := "video/mp4"

    // 1. Initialize authentication and obtain vodClient.
    client := createVodClient()
    // 2. Create a VOD media asset.
    fmt.Println("Start creating a media file: " + fileName)
    assetResponse := createAsset(client, fileName, fileName, fileType)

    // 3. Obtain authorization for initializing an upload task.
    initAuthResponse := initPartUploadAuthority(client, assetResponse, fileContentType)

    // 4. Initialize the upload task.
    uploadId := initPartUpload(initAuthResponse.SignStr, fileContentType)

    // Count the number of file parts.
    partNumber := 1
```



```
    // 7. Read the file content and repeat steps 5 and 6 to upload all parts.
    for {
        buf := make([]byte, bufferSize)
        n, err := file.Read(buf)
        if n == 0 || err != nil || err == io.EOF {
            break
        }
        // Perform MD5 encoding and then Base64 encoding.
        h := md5.New()
        h.Write(buf[:n])
        data := h.Sum(nil)
        contentMd5 := base64.StdEncoding.EncodeToString(data)
        fmt.Println("The MD5 value of the ", partNumber, " part of the file is :", contentMd5)

        // 5. Obtain authorization for multipart upload.
        partUploadAuthorityResponse := getPartUploadAuthority(client, fileContentType, assetResponse,
            contentMd5, uploadId, partNumber)

        // 6. Upload parts.
        uploadPartFile(partUploadAuthorityResponse.SignStr, buf[:n], contentMd5)

        // The part number automatically increments by one.
        partNumber++
    }

    // 8. Obtain authorization for obtaining uploaded parts.
    listPartUploadAuthorityResponse := listUploadedPartAuthority(client, assetResponse, uploadId)

    // 9. Obtain uploaded parts.
    partInfo := listUploadedPart(listPartUploadAuthorityResponse.SignStr)

    // 10. Obtain authorization for merging parts.
    mergePartUploadAuthorityResponse := mergeUploadedPartAuthority(client, assetResponse, uploadId)

    // 11. Merge uploaded parts.
    mergeUploadedPart(mergePartUploadAuthorityResponse.SignStr, partInfo)

    // 12. Confirm media asset upload.
    confirmUploaded(client, assetResponse)

    fmt.Println("Media asset created. assetId:", *assetResponse.AssetId)
}

/*
Verify the file and its path.
filePath: file path
*/
func validFile(filePath string) os.FileInfo {
    // Check whether the file exists.
    fileInfo, err := os.Stat(filePath)
    if os.IsNotExist(err) {
        fmt.Println("The file does not exist.")
        // Throw an error.
        panic(errors.New("The file does not exist."))
    }
    if err != nil {
        fmt.Println("Error:", err)
        panic(err)
    }
    return fileInfo
}

/*
1. Construct authentication.
*/
func createVodClient() *vod.VodClient {
    auth, _ := basic.NewCredentialsBuilder().
        WithAk(ak).
```

```
        WithSk(sk).
        SafeBuild()
    reg, _ := region.SafeValueOf(regionNorth4)

    build, _ := vod.VodClientBuilder().
        WithRegion(reg).
        WithCredential(auth).
        SafeBuild()

    client := vod.NewVodClient(build)
    return client
}

/*
2. Create a media asset.
videoName: name
title: title
videoType: file type
*/
func createAsset(client *vod.VodClient, videoName string, title string,
    videoType string) *model.CreateAssetByFileUploadResponse {
    request := &model.CreateAssetByFileUploadRequest{}
    request.Body = &model.CreateAssetByFileUploadReq{
        VideoName: videoName,
        Title:     title,
        VideoType: videoType,
    }
    response, err := client.CreateAssetByFileUpload(request)
    if err == nil {
        fmt.Println(response)
    } else {
        fmt.Println(err)
        panic(err)
    }
    return response
}

/*
3. Obtain authorization for initializing an upload task.
client
assetResponse
*/
func initPartUploadAuthority(client *vod.VodClient, assetResponse *model.CreateAssetByFileUploadResponse,
    fileContentType string) *model.ShowAssetTempAuthorityResponse {
    fmt.Println("Obtain authorization for initializing an upload task. initPartUploadAuthority start")
    // Configure parameters.
    request := &model.ShowAssetTempAuthorityRequest{}
    request.HttpVerb = "POST"
    request.Bucket = assetResponse.Target.Bucket
    request.ObjectKey = assetResponse.Target.Object
    request.ContentType = &fileContentType
    // Send an initialization request.
    response, err := client.ShowAssetTempAuthority(request)
    if err == nil {
        fmt.Print(response)
    } else {
        fmt.Println(err)
        panic(err)
    }
    fmt.Println("Obtain authorization for initializing an upload task. initPartUploadAuthority end\n")
    return response
}

/*
4. Initialize multipart upload.
signStr: signed URL for initialization returned in step 3
contentType: media file format
return uploadId
*/
```



```
func initPartUpload(signStr *string, contentType string) string {
    fmt.Println("Initialize multipart upload. initPartUpload start")
    // Create an HTTP client.
    client := &http.Client{}
    // Create a request.
    req, err := http.NewRequest("POST", *signStr, nil)

    req.Header.Set("Content-Type", contentType)
    // Send a request.
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("Error sending request:", err)
        panic(err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            panic(err)
        }
    }(resp.Body)

    // Read the response body.
    body, _ := io.ReadAll(resp.Body)
    // Print the response body.
    fmt.Println(string(body))

    // Parse the response body. The returned result is XML text, which is parsed based on the format.
    var initiateMultipartUploadResult InitiateMultipartUploadResult
    err = xml.Unmarshal(body, &initiateMultipartUploadResult)
    if err != nil {
        panic(err)
    }
    fmt.Printf("Bucket:%s, Key:%s, UploadId:%s\n", initiateMultipartUploadResult.Bucket,
        initiateMultipartUploadResult.Key, initiateMultipartUploadResult.UploadId)
    fmt.Println("Initialize multipart upload. initPartUpload end\n")
    return initiateMultipartUploadResult.UploadId
}

/*
5. Obtain authorization for multipart upload.
client
fileContentType: media file format
assetResponse: response to media asset creation
contentMd5: contentMd5 of the current part
uploadId
partNumber: part number
return: authorization result
*/
func getPartUploadAuthority(client *vod.VodClient, fileContentType string, assetResponse
*model.CreateAssetByFileUploadResponse,
    contentMd5 string, uploadId string, partNumber int) *model.ShowAssetTempAuthorityResponse {
    fmt.Println("Obtain authorization for multipart upload. getPartUploadAuthority start; partNumber:",
partNumber)
    // Configure parameters.
    request := &model.ShowAssetTempAuthorityRequest{}
    request.HttpVerb = "PUT"
    request.Bucket = assetResponse.Target.Bucket
    request.ObjectKey = assetResponse.Target.Object
    request.ContentType = &fileContentType
    request.ContentMd5 = &contentMd5
    request.UploadId = &uploadId
    partNumberRequest := int32(partNumber)
    request.PartNumber = &partNumberRequest
    // Send a request.
    response, err := client.ShowAssetTempAuthority(request)
    if err == nil {
        fmt.Print(response)
    } else {
        fmt.Println(err)
```



```
        panic(err)
    }
    fmt.Println("Obtain authorization for multipart upload. getPartUploadAuthority end; partNumber:",
partNumber, "\n")
    return response
}

/*
6. Upload parts.
signStr: signed URL for upload returned in step 5
fileByte: data of the current part
contentMd5: contentMd5 of the current part
*/
func uploadPartFile(signStr *string, fileByte []byte, contentMd5 string) {
    fmt.Print("Upload parts. uploadPartFile start")

    // Create an HTTP client.
    client := &http.Client{}
    // Create a request.
    req, err := http.NewRequest("PUT", *signStr, bytes.NewBuffer(fileByte))
    req.Header.Set("Content-MD5", contentMd5)
    req.Header.Set("Content-Type", "application/octet-stream")

    // Send a request.
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("Error sending request:", err)
        panic(err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            panic(err)
        }
    }(resp.Body)
    // Read the response body.
    body, _ := io.ReadAll(resp.Body)
    // Print the response body.
    fmt.Println(string(body))
    if resp.StatusCode != 200 {
        panic("File upload failed.")
    }
    fmt.Println("Upload parts. uploadPartFile end \n")
}

/*
8. Obtain authorization for listing uploaded parts.
client
assetResponse: response to media asset creation
uploadId
return
*/
func listUploadedPartAuthority(client *vod.VodClient, assetResponse
*model.CreateAssetByFileUploadResponse,
    uploadId string) *model.ShowAssetTempAuthorityResponse {
    fmt.Println("Obtain authorization for listing uploaded parts. listUploadedPartAuthority start")
    // Configure parameters.
    request := &model.ShowAssetTempAuthorityRequest{}
    request.HttpVerb = "GET"
    request.Bucket = assetResponse.Target.Bucket
    request.ObjectKey = assetResponse.Target.Object
    request.UploadId = &uploadId
    // Send a request.
    response, err := client.ShowAssetTempAuthority(request)
    if err == nil {
        fmt.Print(response)
    } else {
        fmt.Println(err)
        panic(err)
```



```
    }
    fmt.Println("Obtain authorization for listing uploaded parts. listUploadedPartAuthority end\n")
    return response
}

/*
9. Query uploaded parts.
signStr: signed URL for query returned in step 8
return
*/
func listUploadedPart(signStr *string) string {
    fmt.Println("Query uploaded parts. listUploadedPart start")
    // Query the start number of file parts.
    partNumberMarker := 0
    // Parameter (XML) for assembling merged parts
    result := "<CompleteMultipartUpload>"
    // Create an HTTP client.
    client := &http.Client{}
    for {
        // Create a request.
        url := *signStr + "&part-number-marker=" + fmt.Sprintf("%d", partNumberMarker)
        req, _ := http.NewRequest("GET", url, nil)
        // Send a request.
        resp, _ := client.Do(req)
        // Read the response body.
        body, _ := io.ReadAll(resp.Body)
        // Print the response body.
        fmt.Println(string(body))
        // Parse and convert the response result to XML text.
        var listPartsResult ListPartsResult
        _ = xml.Unmarshal(body, &listPartsResult)
        // parts is not displayed in the response result.
        if len(listPartsResult.Parts) < 1 {
            break
        }
        // Loop parts to assemble Part data.
        for _, part := range listPartsResult.Parts {
            num := part.PartNumber
            tag := part.Etag
            result += "<Part>" +
                "<PartNumber>" +
                fmt.Sprintf("%d", num) +
                "</PartNumber>" +
                "<ETag>" +
                tag +
                "</ETag>" +
                "</Part>"
        }
        // Set the next part number in the response to the start part number and send the request again.
        partNumberMarker = listPartsResult.NextPartNumberMarker
        if partNumberMarker%1000 != 0 {
            break
        }
    }
    result += "</CompleteMultipartUpload>"
    fmt.Println(result)
    fmt.Println("Query uploaded parts. listUploadedPart end\n")
    return result
}

/*
10. Obtain authorization for merging parts.
client
assetResponse
uploadId
return
*/
func mergeUploadedPartAuthority(client *vod.VodClient, assetResponse
*model.CreateAssetByFileUploadResponse,
```



```
    uploadId string) *model.ShowAssetTempAuthorityResponse {
    fmt.Println("Obtain authorization for merging parts. mergeUploadedPartAuthority start")
    // Configure parameters.
    request := &model.ShowAssetTempAuthorityRequest{}
    request.HttpVerb = "POST"
    request.Bucket = assetResponse.Target.Bucket
    request.ObjectKey = assetResponse.Target.Object
    request.UploadId = &uploadId
    // Send a request.
    response, err := client.ShowAssetTempAuthority(request)
    if err == nil {
        fmt.Print(response)
    } else {
        fmt.Println(err)
        panic(err)
    }
    fmt.Println("Obtain authorization for merging parts. mergeUploadedPartAuthority end\n")
    return response
}

/*
11. Merge parts.
signStr: signed URL for merging parts returned in step 10
partInfo: data of the parts to be merged
*/
func mergeUploadedPart(signStr *string, partInfo string) {
    fmt.Println("Merge parts. mergeUploadedPart start")
    // Create an HTTP client.
    client := &http.Client{}
    // Create a request.
    //req, err := http.NewRequest("POST", *signStr, bytes.NewBuffer([]byte(partInfo)))
    req, err := http.NewRequest("POST", *signStr, bytes.NewBufferString(partInfo))
    // Add "Content-Type":"application/xml" to the request header.
    req.Header.Set("Content-Type", "application/xml")
    // Send a request.
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("Error sending request:", err)
        panic(err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            panic(err)
        }
    }(resp.Body)
    // Read the response body.
    body, _ := io.ReadAll(resp.Body)
    // Print the response body.
    fmt.Println(string(body))
    fmt.Println("Merge parts. mergeUploadedPart end\n")
}

/*
12. Confirm the upload completion.
*/
func confirmUploaded(client *vod.VodClient, assetResponse *model.CreateAssetByFileUploadResponse) {
    fmt.Println("Confirm the upload completion. confirmUploaded start")
    // Configure request parameters.
    request := &model.ConfirmAssetUploadRequest{}
    request.Body = &model.ConfirmAssetUploadReq{
        Status:  model.GetConfirmAssetUploadReqStatusEnum().CREATED,
        AssetId: *assetResponse.AssetId,
    }
    // Send a request.
    response, err := client.ConfirmAssetUpload(request)
    if err == nil {
        fmt.Print(response)
    } else {
```



```
        fmt.Println(err)
        panic(err)
    }
    fmt.Println("Uploaded, assetId:", *response.AssetId)
}

// InitiateMultipartUploadResult: XML structure returned after multipart upload initialization
type InitiateMultipartUploadResult struct {
    Bucket   string `xml:"Bucket"`
    Key      string `xml:"Key"`
    UploadId string `xml:"UploadId"`
}

// ListPartsResult: XML structure returned after querying uploaded parts
type ListPartsResult struct {
    Bucket               string `xml:"Bucket"`
    Key                  string `xml:"Key"`
    UploadId             string `xml:"UploadId"`
    NextPartNumberMarker int    `xml:"NextPartNumberMarker"`
    Parts                []Part `xml:"Part"`
}

// Part: part structure of the returned xml listPartsResult after querying uploaded parts
type Part struct {
    Etag       string `xml:"ETag"`
    PartNumber int    `xml:"PartNumber"`
}
```

19.7.6 PHP

Sample code of multipart upload using PHP:

```
<?php
namespace HuaweiCloud\SDK\Vod\V1\Model;

use DOMDocument;
use DOMException;
use Exception;

require_once "vendor/autoload.php";
use HuaweiCloud\SDK\Core\Auth\BasicCredentials;
use HuaweiCloud\SDK\Core\Http\HttpConfig;
use HuaweiCloud\SDK\Vod\V1\VodClient;

// Region
const region_north1 = 'cn-north-1';
const region_north4 = 'cn-north-4';
const region_east2 = 'cn-east-2';

// ak/sk
const ak = "";
const sk = "";

const projectId = "";

const endpoint = "https://vod." . region_north4 . ".myhuaweicloud.com";

// Set the buffer size as needed, that is, the size of the file part read each time.
// 1M
const bufferSize = 1024 * 1024;

main();

/*
 * Example of multipart upload
 */
function main(): void
{
    // Path of the local media asset to be uploaded
```



```
    $filePath = "";

    // Upload the media asset.
    partUpload($filePath);
}

/**
 * Multipart upload
 * @param $filePath: local path where the file to be uploaded is
 * @return void
 */
function partUpload(string $filePath)
{
    $handle = fopen($filePath, "r");
    if(!$handle){
        throw new Exception("The file cannot be opened.");
    }
    try {
        $fileName = basename($filePath);

        // An MP4 file is used as an example. For details about other formats, see the official website.
        $fileType = "MP4";
        $fileContentType = "video/mp4";

        // 1. Initialize authentication and obtain vodClient.
        $vodClient = createVodClient();
        echo "Start to create a media asset:" . $fileName . "\n";
        // 2. Create a VOD media asset.
        $assetResponse = createAsset($vodClient, $fileName, $fileName, $fileType);
        // 3. Obtain authorization for initializing an upload task.
        $initAuthResponse = initPartUploadAuthority($vodClient, $assetResponse, $fileContentType);
        // 4. Initialize the upload task.
        $uploadId = initPartUpload($initAuthResponse->getSignStr(), $fileContentType);
        // Count the number of file parts.
        $partNumber = 1;

        // 7. Read the file content and repeat steps 5 and 6 to upload all parts.
        while (!feof($handle)) {
            $content = fread($handle, bufferSize);
            // Perform MD5 encoding and then Base64 encoding.
            $md5Hash = md5($content);
            echo "md5:". $md5Hash ."\n";
            $hexString = hex2bin($md5Hash);
            $contentMd5 = base64_encode($hexString);
            echo "" . $partNumber . " part of the file: contentMd5:" . $contentMd5 . "\n";

            // 5. Obtain authorization for multipart upload.
            $partUploadAuthorityResponse = getPartUploadAuthority($vodClient, $fileContentType,
$assetResponse, $contentMd5, $uploadId, $partNumber);
            // 6. Upload parts.
            uploadPartFile($partUploadAuthorityResponse->getSignStr(), $content, $contentMd5);

            // The part number automatically increments by one.
            $partNumber++;
        }
        fclose($handle);

        // 8. Obtain authorization for obtaining uploaded parts.
        $listPartUploadAuthorityResponse = listUploadedPartAuthority($vodClient, $assetResponse, $uploadId);
        // 9. Obtain uploaded parts.
        $partInfo = listUploadedPart($listPartUploadAuthorityResponse->getSignStr());
        // 10. Obtain authorization for merging parts.
        $mergePartUploadAuthorityResponse = mergeUploadedPartAuthority($vodClient, $assetResponse,
$uploadId);
        // 11. Merge uploaded parts.
        mergeUploadedPart($mergePartUploadAuthorityResponse->getSignStr(), $partInfo);
        // 12. Confirm media asset upload.
        confirmUploaded($vodClient, $assetResponse);
```



```
        echo "Media asset created assetId:" . $assetResponse->getAssetId() . "\n";

    } catch (Exception $e) {
        echo $e;
    }
}

/**
 * 1. Construct authentication.
 * @return VodClient
 */
function createVodClient(): VodClient
{
    $credentials = new BasicCredentials(ak, sk, projectId);
    $config = HttpConfig::getDefaultConfig();
    $config->setIgnoreSslVerification(true);

    $client = VodClient::newBuilder(new VodClient)
        ->withHttpConfig($config)
        ->withEndpoint(endpoint)
        ->withCredentials($credentials)
        ->build();
    return $client;
}

/**
 * 2. Create a media asset.
 * @param VodClient $vodClient
 * @param string $videoName
 * @param string $title
 * @param string $videoType
 * @return CreateAssetByFileUploadResponse $response
 */
function createAsset(VodClient $vodClient, string $videoName, string $title, string $videoType):
CreateAssetByFileUploadResponse
{
    $request = new CreateAssetByFileUploadRequest();
    $body = new CreateAssetByFileUploadReq();
    $body->setVideoName($videoName);
    $body->setTitle($title);
    $body->setVideoType($videoType);
    $request->setBody($body);
    $response = $vodClient->createAssetByFileUpload($request);
    echo $response . "\n";
    return $response;
}

/**
 * 3. Obtain authorization for initializing an upload task.
 * @param VodClient $client
 * @param CreateAssetByFileUploadResponse $assetResponse
 * @param string $fileContentType
 * @return ShowAssetTempAuthorityResponse
 */
function initPartUploadAuthority(VodClient $client, CreateAssetByFileUploadResponse $assetResponse,
string $fileContentType): ShowAssetTempAuthorityResponse
{
    echo "Obtain authorization for initializing an upload task. initPartUploadAuthority start\n";
    $request = new ShowAssetTempAuthorityRequest();
    // Configure parameters.
    $request->setHttpVerb("POST");
    $request->setBucket($assetResponse->getTarget()->getBucket());
    $request->setObjectKey($assetResponse->getTarget()->getObject());
    $request->setContentType($fileContentType);
    // Send an initialization request.
    $response = $client->showAssetTempAuthority($request);
    echo "Obtain authorization for initializing an upload task. end; response:" . $response . "\n\n";
    return $response;
}
```



```
/**
 * 4. Initialize multipart upload.
 * @param string $signStr
 * @param string $contentType
 * @return string|null
 */
function initPartUpload(string $signStr, string $contentType): string
{
    echo "Initialize multipart upload. initPartUpload start\n";
    // Initialize the cURL session.
    $ch = curl_init();
    // Configure the cURL.
    curl_setopt($ch, CURLOPT_URL, $signStr);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        "Content-Type: " . $contentType // Set the request header.
    ));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Verify the cURL peer certificate. (Generally, only
this item is required.)
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false); // Check whether the server SSL certificate contains a
common name.
    curl_setopt($ch, CURLOPT_SSLVERSION, 0);  // Transfer a long parameter that contains the SSL version.

    // Execute the cURL session.
    $response = curl_exec($ch);
    if (curl_errno($ch)) {
        echo 'cURL Error: ' . curl_error($ch);
    } else {
        // Print the response content.
        echo $response;
    }

    echo $response . "\n";
    $initDom = new DOMDocument();
    $initDom->loadXML($response);
    curl_close($ch);
    $UploadIdNode = $initDom->getElementsByTagName("UploadId");
    $uploadId = $UploadIdNode->item(0)->nodeValue;
    echo "Initialize multipart upload. initPartUpload end; UploadId:" . $uploadId . "\n\n";
    return $uploadId;
}

/**
 * 5. Obtain authorization for multipart upload.
 * @param VodClient $client
 * @param string $fileContentType
 * @param CreateAssetByFileUploadResponse $assetResponse
 * @param string $contentMd5
 * @param string $uploadId
 * @param int $partNumber
 * @return ShowAssetTempAuthorityResponse
 */
function getPartUploadAuthority(VodClient $client, string $fileContentType,
CreateAssetByFileUploadResponse $assetResponse,
                                string $contentMd5, string $uploadId, int $partNumber):
ShowAssetTempAuthorityResponse
{
    echo "Obtain authorization for multipart upload. getPartUploadAuthority start; partNumber:" .
$partNumber . "\n";
    $request = new ShowAssetTempAuthorityRequest();
    $request->setHttpVerb("PUT");
    $request->setBucket($assetResponse->getTarget()->getBucket());
    $request->setObjectKey($assetResponse->getTarget()->getObject());
    $request->setContentType($fileContentType);
    $request->setContentMd5($contentMd5);
    $request->setUploadId($uploadId);
    $request->setPartNumber($partNumber);
```



```
    $response = $client->showAssetTempAuthority($request);
    echo "Obtain authorization for multipart upload. getPartUploadAuthority end; partNumber:" .
$partNumber . "\n";
    return $response;
}

/**
 * 6. Upload parts.
 * @param string $signStr
 * @param string $fileByte
 * @param string $contentMd5
 * @return void
 * @throws Exception
 */
function uploadPartFile(string $signStr, $fileByte, string $contentMd5)
{
    echo "Upload parts. uploadPartFile start\n";
//    echo $fileByte . "\n";
    // Initialize the cURL session.
    $ch = curl_init();
    // Configure the cURL.
    curl_setopt($ch, CURLOPT_URL, $signStr);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
    curl_setopt($ch, CURLOPT_POSTFIELDS, $fileByte);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        "Content-MD5: " . $contentMd5,
        "Content-Type: application/octet-stream"
    ));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Verify the cURL peer certificate. (Generally, only
this item is required.)
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false); // Check whether the server SSL certificate contains a
common name.
    curl_setopt($ch, CURLOPT_SSLVERSION, 0);  // Transfer a long parameter that contains the SSL version.
    // Execute the cURL session.
    $response = curl_exec($ch);
    if (curl_errno($ch)) {
        echo 'cURL Error: ' . curl_error($ch);
        // Print the response content.
        echo $response . "\n";
        throw new Exception("Upload failed.");
    }
    // End the session.
    curl_close($ch);
    echo "Upload parts. uploadPartFile end;\n\n";
}

/**
 * 8. Obtain authorization for listing uploaded parts.
 * @param VodClient $client
 * @param CreateAssetByFileUploadResponse $assetResponse
 * @param string $uploadId
 * @return ShowAssetTempAuthorityResponse
 */
function listUploadedPartAuthority(VodClient $client, CreateAssetByFileUploadResponse $assetResponse,
string $uploadId):ShowAssetTempAuthorityResponse
{
    echo "Obtain authorization for listing uploaded parts. listUploadedPartAuthority start\n";
    $request = new ShowAssetTempAuthorityRequest();
    $request->setHttpVerb("GET");
    $request->setBucket($assetResponse->getTarget()->getBucket());
    $request->setObjectKey($assetResponse->getTarget()->getObject());
    $request->setUploadId($uploadId);
    $response = $client->showAssetTempAuthority($request);
    echo "Obtain authorization for listing uploaded parts. listUploadedPartAuthority end; response:" .
$response . "\n";
    return $response;
}
```



```
/**
 * 9. Query uploaded parts.
 * @param string $signStr
 * @return false|string
 * @throws DOMException
 */
function listUploadedPart(string $signStr): false|string
{
    echo "Query uploaded parts. listUploadedPart start\n";
    $partNumberMarker = 0;
    $compDom = new DOMDocument();
    $root = $compDom->createElement("CompleteMultipartUpload");
    $compDom->appendChild($root);
    while(true)
    {
        // Initialize the cURL session.
        $ch = curl_init();
        // Configure the cURL.
        curl_setopt($ch, CURLOPT_URL, $signStr . "&part-number-marker=" . $partNumberMarker);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Verify the cURL peer certificate. (Generally, only
this item is required.)
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false); // Check whether the server SSL certificate
contains a common name.
        curl_setopt($ch, CURLOPT_SSLVERSION, 0);  // Transfer a long parameter that contains the SSL version.

        // Execute the cURL session.
        $response = curl_exec($ch);
        if (curl_errno($ch)) {
            echo 'cURL Error: ' . curl_error($ch);
        } else {
            // Print the response content.
            echo $response . "\n";
        }
        // End the cURL session.
        curl_close($ch);
        $listDom = new DOMDocument();
        $listDom->loadXML($response);
        $nextPartNumberMarkerNode = $listDom->getElementsByTagName("NextPartNumberMarker");
        $partNodes = $listDom->getElementsByTagName("Part");
        foreach($partNodes as $partNode)
        {
            $partNumberNode = $partNode->getElementsByTagName("PartNumber");
            $partNumber = $partNumberNode->item(0)->nodeValue;
//            echo $partNumber . "\n";
            $eTagNode = $partNode->getElementsByTagName("ETag");
            $eTag = $eTagNode->item(0)->nodeValue;
//            echo $eTag . "\n";
            try{
                $part = $compDom->createElement("Part");
                $part->appendChild($compDom->createElement("PartNumber", $partNumber));
                $part->appendChild($compDom->createElement("ETag", $eTag));
                $root->appendChild($part);
            } catch (DOMException $e){
                echo $e->getMessage();
            }
        }
        $partNumberMarker = $nextPartNumberMarkerNode->item(0)->nodeValue;
        if($partNumberMarker % 1000 != 0 || $partNumberMarker == 0)
        {
            break;
        }
    }
    $partInfo = $compDom->saveXML();
    echo $partInfo . "\n";
    echo "Query uploaded parts. listUploadedPart end\n\n";
    return $partInfo;
}
```



```
/**
 * 10. Obtain authorization for merging parts.
 * @param VodClient $client
 * @param CreateAssetByFileUploadResponse $assetResponse
 * @param string $uploadId
 * @return ShowAssetTempAuthorityResponse
 */
function mergeUploadedPartAuthority(VodClient $client, CreateAssetByFileUploadResponse $assetResponse,
string $uploadId):ShowAssetTempAuthorityResponse
{
    echo "Obtain authorization for merging parts. mergeUploadedPartAuthority start\n";
    $request = new ShowAssetTempAuthorityRequest();
    $request->setHttpVerb("POST");
    $request->setBucket($assetResponse->getTarget()->getBucket());
    $request->setObjectKey($assetResponse->getTarget()->getObject());
    $request->setUploadId($uploadId);
    $response = $client->showAssetTempAuthority($request);
    echo "Obtain authorization for merging parts. mergeUploadedPartAuthority end; response:" . $response .
"\n";
    return $response;
}

/**
 * 11. Merge parts.
 * @param string $signStr
 * @param string $partInfo
 * @return void
 */
function mergeUploadedPart(string $signStr, string $partInfo)
{
    echo "Merge parts. mergeUploadedPart start\n";
    // Initialize the cURL session.
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $signStr);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $partInfo);
    // Add Content-Type to the request header and set the value to application/xml.
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        "Content-Type: application/xml"
    ));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Verify the cURL peer certificate. (Generally, only
this item is required.)
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false); // Check whether the server SSL certificate contains a
common name.
    curl_setopt($ch, CURLOPT_SSLVERSION, 0);  // Transfer a long parameter that contains the SSL version.
    // Execute the cURL session.
    $response = curl_exec($ch);
    if (curl_errno($ch)) {
        echo 'cURL Error: ' . curl_error($ch);
    } else {
        // Print the response content.
        echo $response . "\n";
    }
    // End the session.
    curl_close($ch);
    echo "Merge parts. mergeUploadedPart end\n";
}

/**
 * 12. Confirm the upload completion.
 * @param VodClient $client
 * @param CreateAssetByFileUploadResponse $assetResponse
 * @return void
 */
function confirmUploaded(VodClient $client, CreateAssetByFileUploadResponse $assetResponse): void
{
    echo "Confirm the upload completion. confirmUploadedPart start\n";
```



```
    $request = new ConfirmAssetUploadRequest();
    $body = new ConfirmAssetUploadReq();
    $body->setStatus("CREATED");
    $body->setAssetId($assetResponse->getAssetId());
    $request->setBody($body);
    $response = $client->confirmAssetUpload($request);
    echo "Uploaded. confirmUploadedPart end; response:" . $response . "\n";
}
```


20 Change History

Table 20-1 Change history

Released On	Description
2024-11-15	This issue is the twelfth official release. Added the transcoded output management API group.
2024-10-15	This issue is the eleventh official release. • Added the API for modifying the cold storage scope of a media asset. • Added the API for querying media asset cold storage. • Added the sample code of multipart upload using Python and Go.
2024-08-15	This issue is the tenth official release. Added the **preset**, **max_iframes_interval**, and **hls_audio_separate** parameters to the transcoding template management API group.
2024-06-18	This issue is the ninth official release. Added the sample code of multipart upload using Java and JavaScript.
2023-12-21	This issue is the eighth official release. Added the APIs for querying daily playback statistics of a media asset and changing the OBS storage class of a media asset.
2023-07-14	This issue is the seventh official release. • Added the unit description for the **values** parameter in the API for querying CDN statistics. • Revised the function description of the API for querying origin server statistics.

Released On	Description
2023-05-23	This issue is the sixth official release. ● Added the description of filling in the request header in Example 1: Uploading a Media File Less Than 20 MB. ● Revised the content in Example 2: Uploading a Media File Greater Than 20 MB by Part. ● Added the file format description for the **content_type** field in the API for obtaining authorization for multipart upload.
2023-05-06	This issue is the fifth official release. ● Added the description of endpoints. ● Added the description that there is a one-hour time difference in the CDN statistics query API.
2023-03-30	This issue is the fourth official release. Added the RMVB and WebM video formats to the **video_type** field in the API for dumping media files from OBS to VOD.
2023-03-08	This issue is the third official release. Added the subtitle management API.
2022-11-23	This issue is the second official release. ● The optional parameters **name**, **quality_info_list**, and **common** in the API for creating a custom transcoding template become mandatory ones. ● The optional parameters **group_id**, **quality_info_list**, and **common** in the API for modifying a transcoding template become mandatory ones. ● The optional parameters **name** and **template_group_list** in the API for creating a transcoding template group set become mandatory ones. ● The optional parameter **collection_id** in the API for modifying a transcoding template group set becomes a mandatory one.
2022-10-12	This issue is the first official release.

