Alibaba Cloud
ApsaraVideo for Media Processing

Developer Guide

Issue: 20190131

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning informatio n, supplementary instructions, and other content that the user must understand.	Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
`Italics`	It is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *`[-all|-t]`*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Concepts

1.1 Task and MPS queue

This article introduces several basic concepts and relationships in MPS to help developers better understand and use MPS.

Concept description:

- Task

  A task in the MPS is an abstract concept which contains a variety of types of tasks: transcoding tasks, screenshot tasks, and media information tasks.

  One task contains three key pieces of information: input, output and parameters . Input and output parameters are used to set the input file and the output file for the completed task. These parameters are used to set the detailed configuration for executing the specific function.

- Parameters

  - Template parameters

    Due to the large number of tasks, it is rework to repeat each task submission . Templates are a concept proposed to solve this problem. The essence of the template is a collection of commonly used parameters. This collection can reduce the number of parameters that need to be specified when submitting the task, thus simplifing the submission code.

  - API parameters

    To create a template for each different combination of parameters can result in a dramatic increase in the number of templates and make template management more complex. Therefore, parameters can be set not only in the template, but also through the API.

  - Covering order

    The API parameter has a higher priority than the corresponding parameter in the template and will cover the latter.

    For example: the same video can be transcoded to output multiple resolutions (HD, SD), different definition formats (MP4), encoding standards (H. 264) and

frame rate. The difference lies only in the rate and resolution. You can create a template with a default combination of parameters (MP4 + H. 264 +25FPS + 2Mbps +1280x720). When calling the API, if the API parameter is not set, the task is executed according to the default parameters (2Mbps +1280x720); and if the API parameter (4Mbps +1920x1080) is set, the task is executed according to the API parameters instead (4Mbps +1920x1080).

- MPS queues

  After the user submits a task through the API interface, the task queues in the MPS queue first and then is executed in the order of priority and submission order.

  The tasks in the MPS queueq can have multiple priorities (10 is the top priority, 1 is the lowest 1, 6 is the default). In case of the same priority , the taskes submitted earlier are executed first. For tasks submitted at the same time, those with higher priority are executed first.

- Task execution and completion

  - Synchronous and asynchronous

    Depending on the type of job, some jobs can be completed quickly, however, most jobs cannot be completed instantly. There are two ways to execute jobs: synchronous and asynchronous. Synchronous types (such as screenshot tasks) return results immediately, while asynchronous types (such as transcoding tasks) results in two kinds of queries: scheduled polling and message notificati on.

  - Regular polling

    Each task is identified by a unique task ID, which is returned synchronously to the caller when the task is submitted, after which task results can be queried by the task ID. The disadvantage of this approach lies in the fact that it is not executed in real time.

    Notifications:

    The MPS queue can be configured to send message notifications, you can get the task results instantly when they are ready. The message notification contains several important pieces of information: task ID, user data, and result.

    ■ Task ID

When you submit a task, record the task ID, and then compare it with the task ID of the message notification to know the result belongs to which task.

- User data

  When submitting a task, you can enter custom user data parameters (such as commodity IDs) each time you execute. The custom user data parameters are then returned in the message notification without the need to record the task ID in the business system. Meanwhile, you can use custom user data, such as commodity IDs, to associate the business system.

1.2 Transcoding template

Due to the many parameters of transcoding job, it is difficult to fill in the transcoding job repeatedly each time, transcoding templates are the concepts proposed to solve this problem, the essence is to combine some common parameters. Transcoding templates are available in two types: Preset templates and Custom templates.

- Preset templates

  Pre-provided to users based on the common combination of some parameters. For more information, see *Preset template details*.

  Preset templates include several subtypes:

  - Preset static template

    Can be used directly as transcoding template, including video transcoding, audio transcoding, transfer package and other scenarios. For example, “MP4-HD” and “MP3-128K”.
  - NarrowBand HD template

    Narrowband HD is a unique technology for media transcoding. In the same bit rate, it can bring higher clarity so as to provide a better user experience at the same cost.
  - Preset smart template

    The preset smart template automatically adjusts the transcoding parameters according to the characteristics of the input file, resulting in lower bit rate at the same resolution, thus reducing more cost.

    Note:

When using the preset smart template, you first need to call`SubmitAnalysisJob` interface (`SubmitAnalysisJob`). After the analysis task successfully completes, you can call the `Query Template Analysis Job` interface (`QueryAnalysisJobList`) to obtain a valid preset smart template corresponding to the input file list. If the preset smart template specified in the submitted transcoding task is in an invalid list, the transcoding task is invalid and will return a failure.

- Custom templates

  With a higher requirement, you can use a custom template to define your own combination of transcoding parameters (audio, video, container, transcode, etc.). Each custom template has a unique template ID.

1.3 Workflows and media

This article introduces several basic concepts and relationships for MPS to help developers better understand and use media processing service.

Concept description:

- Media

  Media includes one input video/audio media file and all the relevant output file, such as transcoding/screenshots/media info/AI tags. Input files and media have a one-to-one relationship and are uniquely identified by the Media ID.

  Media Files

  The Media Files is a collection of all media, with media being the smallest unit for media files.

- Workflow

  Workflow is a like a factory that automates the production of media, it is uniquely identified by a MediaWorkflowId.

  Note:
  Media Workflow also refers to the workflow.

  - Event

Each node in the workflow is called an activity. According to actual requirements, it can be run in parallel (for example, the task A, B, C) or in a serialized manner (for example: the task A1, A2). In addition to the initial input activity and the final release reporting activity, activities supports various types of tasks, such as transcoding tasks and screenshot tasks.

- Starting input activity

  Configure the triggering path of the storage associated with the workflow, and automatically trigger the task running whenever the video/audio multi-media file is uploaded to the corresponding path.

- Finishing post reporting activities

  After the workflow finished running, it sends an implementation message. The running result contains the absolute address of the media ID and the multi-media file, so that the specific multi-media file can be run.

- Task activity

  All parameters supported by the task can be configured in the task activity.

- Matching rules

  For example, the uploaded file is `http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test1.flv`, the result of the configured triggering path are as follows:

Path	Matching or not
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/	Yes
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C2/	No
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/	Yes
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B2/	No
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/	Yes
http://bucket.oss-cn-hangzhou.aliyuncs.com/A2/B/C/	No
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test	Yes

Path	Matching or not
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test2	No

- Extension matching rules

  The automatic triggering system during uploading checks the file extension to avoid generating ineffective data (such as pdf, word files and other files).

  > Note:
  > API manual triggering system does not check the extension.

  The files does not have the extension (file does not include extension separator "."), or the extension conforms to the following rules:

  ■ Video

  3gp, asf, avi, dat, dv, flv, f4v, gif, m2t, m3u8, m4v, mj2, mjpeg, mkv, mov, mp4, mpe, mpg, mpeg, mts, ogg, qt, rm, rmvb, swf, ts, vob, wmv and webm.

  ■ Audio

  aac, ac3, acm, amr, ape, caf, flac, m4a, mp3, ra, wav, wma, aiff

- Workflow running

  Each time you upload a matching multi-media file, it is triggered once. If the same multi-media file is uploaded for multiple times, multiple runnings are triggered. Each running has a unique RunId identifier.

  In addition to the automatic triggering system when uploading, the workflow targets stored multi-media files in storage and also provides a manual API triggering system. Each call to the API triggers a running.

- User data

  You can enter custom user data parameters (for example, commodity IDs) each time you run. The custom user data parameters are then returned in the message notification without the need to record the absolute path of the media ID or multi-media file in the business system. Meanwhile, you can use custom user data, such as commodity IDs, to associate the business system.

2 Developing process of workflows

1. Set a workflow

   Easy to use: By using the GUIs of the console, a cloud-based audio/video handling process is constructed on demand.

   Powerful functions: The screenshot taking, transcoding, narrowband HD analysis, encapsulation, watermarking, and editing functions are supported.

   For more information about console configuration, see *Workflows*.
2. Upload a media file

   After a media file is uploaded to the input bucket and path specified by the workflow, the workflow is automatically executed based on the specified process.
3. Wait for a message notification

   Message notifications during the workflow execution. For example, the execution startup and completion notifications, are received.
4. Play a video

   After a workflow is executed, the playback URL after transcoding is obtained to play a video using a player.

3 Upload videos

3.1 Overview

Upload

Provides upload SDK, supports the web version (JavaScript) and mobile versions (Android and iOS).

Uploads a video file using the console or a third-party tool.

Features

Provides user-friendly APIs. You only need to specify the location to store local and OSS files.

Supports resumable upload, multi-file queue, ultra-large files, recovery from network anomalies, and security mechanisms.

A media workflow is automatically triggered.

Media workflow triggering.

After a multimedia file is uploaded to the input bucket and path specified by the media workflow, the media workflow is automatically executed based on the specified process.

The following conditions must be met when an OSS file is uploaded to automatically trigger a media workflow:

- Match the media workflow.

  For more information about workflow triggering and matching rules, see *Add media*. The workflow is in the `Activated` state.

- Match the file name extension.

  Triggering requirement is that the file is a multi-media file, Media Files service determines through the extension of a file. The file does not contain an extension (the file name does not contain the extension separator ".") or the extension meets the following rules:

  - Video

3gp, asf, avi, dat, dv, flv, f4v, gif, m2t, m3u8, m4v, mj2, mjpeg, mkv, mov, mp4, mpe, mpg, mpeg, mts, ogg, qt, rm, rmvb, swf, ts, vob, wmv and webm.

- Audio

  aac, ac3, acm, amr, ape, caf, flac, m4a, mp3, ra, wav, wma and aiff.

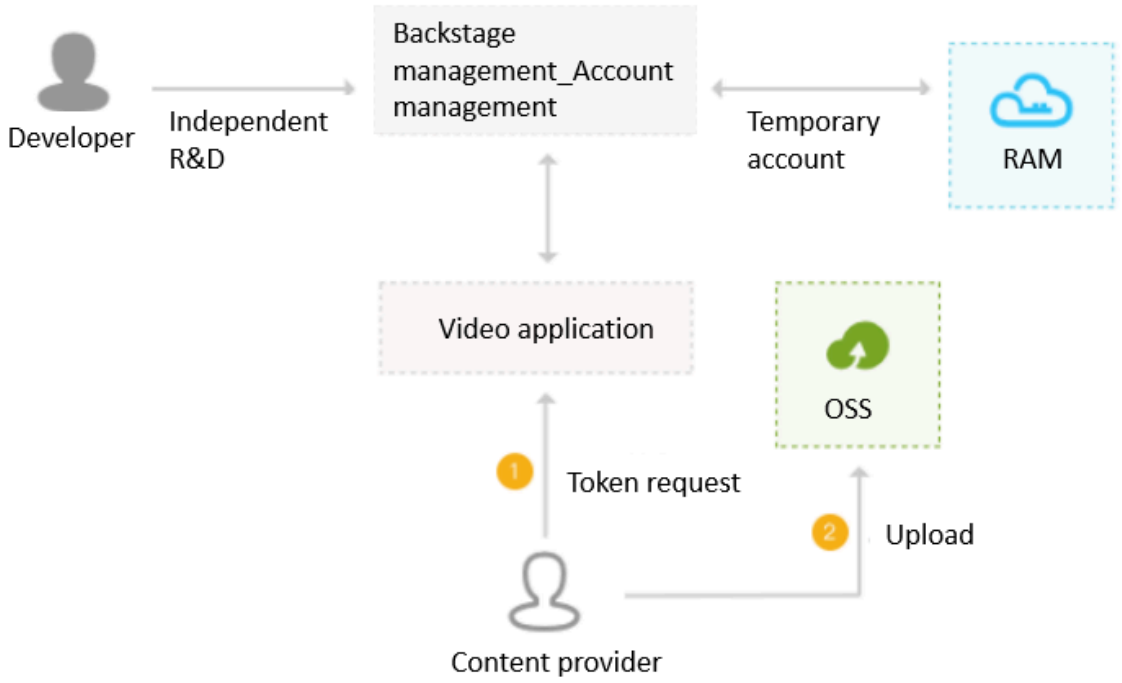
· Specify media attributes.

You can specify media attributes, including the title, tag, description, category, cover URL, and custom data, to trigger a media workflow. For more information about the attribute description, see "Request parameters" in *Add Media*.

Security

In normal cases, a video file is directly uploaded using a client. In this case, the AccessKey must be securely stored on the client. Once being disclosed, the AccessKey is exposed to high risk and hard to be replaced. We recommend that the client access an application to obtain the AccessKey and use the *Token* provided by *RAM*.

Recommended process.



1. Request the token.

   Before each time a file is uploaded, you can use a video application (in App or web mode) to access the application service of the business end. The application service obtains a token from RAM and sends it back to the video application. This

ensures the security, implements identity verification and permission control, and records your upload history.

Before using a token, *Set a subaccount and permissions*.

For more information, see *Java sample code*. (For more information about how to use the token in other languages, see *STS documentation*).

2. Upload a file.

   After integrating the upload SDK to the video application, you can upload files using the obtained token. For more information, see *Usage instruction*.

   - Web.

     As JavaScript files are stored in an application or CDN domain and video files are stored in an OSS domain, a cross-region request is involved when a JavaScript file is uploaded. In this case, *Set CORS*.

     JavaScript

   - Mobile.

     Android

     iOS

3.2 Set a subaccount and authorization

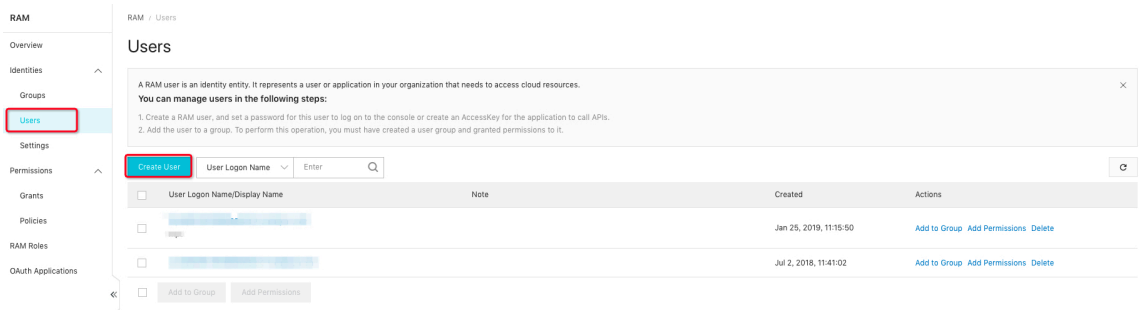
Perform the following steps to set sub-accounts and authorizations.

1. Create a subaccount.

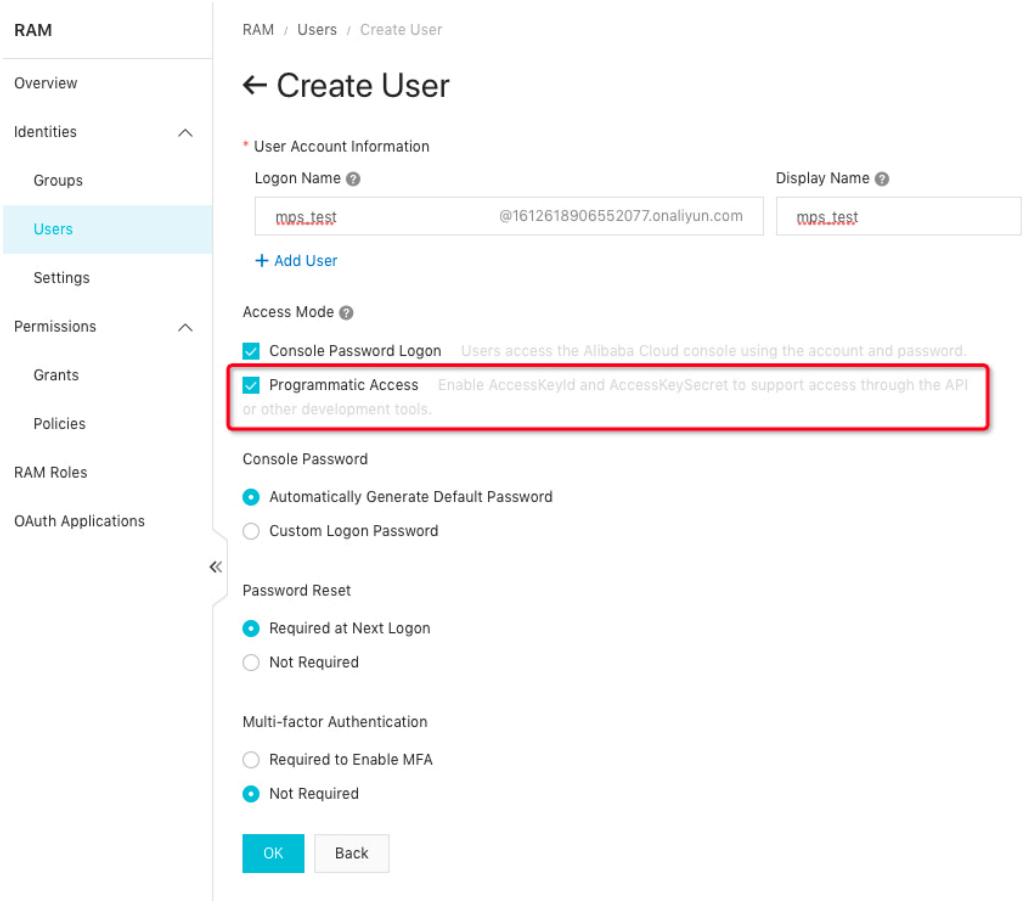
   a. Log on to the *RAM console*.

   b. In the left-side navigation pane, click Identities > Users.

   c. Click Create User.



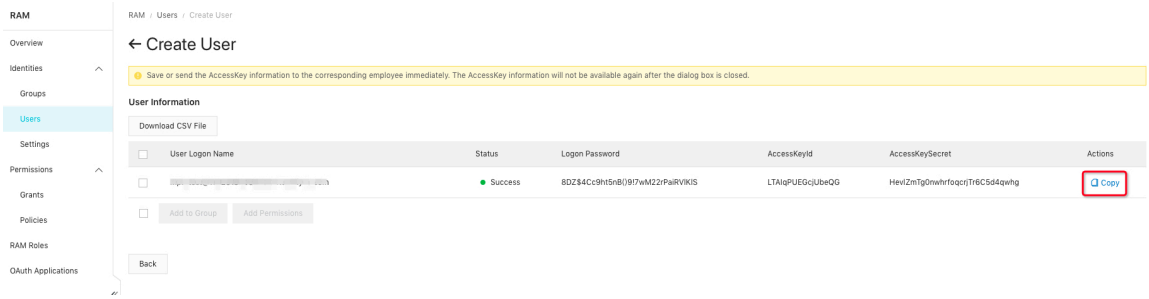
   d. InCreate user, create a subaccount which has the same permissions as the primary account to access MPS.



Note:

Tick Programmatic Access.

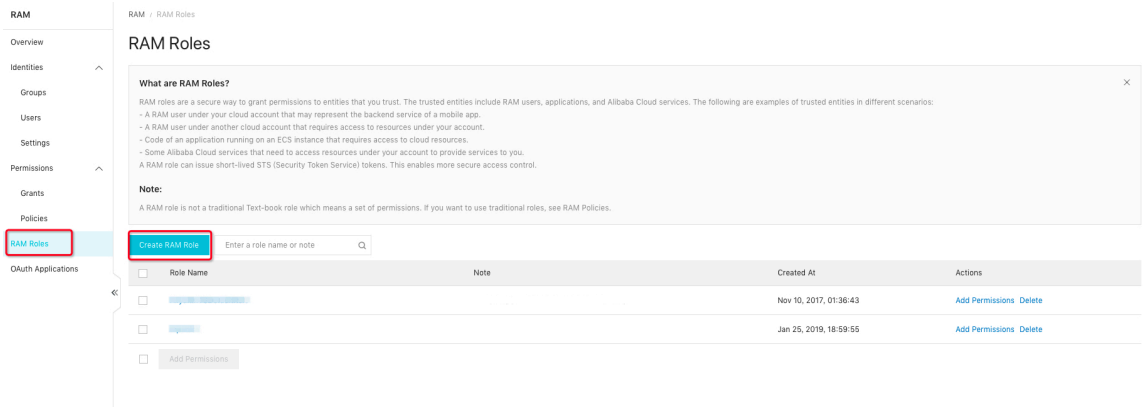
e. Generate AccessKey for this account, copy and save the AccessKey for subsequent access.



2. Create a role.

   a. In the left-side navigation pane, click RAM Roles.

   b. Click Create RAM Role.



c. **In Select type of trusted entity, select Alibaba Cloud Account.**

   In Select Trusted Alibaba Cloud Account, select Current Alibaba Cloud Account, and click OK.

Create RAM Role

Select type of trusted entity

- (●) Alibaba Cloud Account
  A RAM user of a trusted Alibaba Cloud account can assume the RAM role to access your resources. A trusted Alibaba Cloud account can be the current account or another Alibaba Cloud account.
- ( ) Alibaba Cloud Service
  A trusted Alibaba Cloud service can assume the RAM role to access your resources.

* Select Trusted Alibaba Cloud Account

- (●) Current Alibaba Cloud Account
- ( ) Other Alibaba Cloud Account

* RAM Role Name

mps

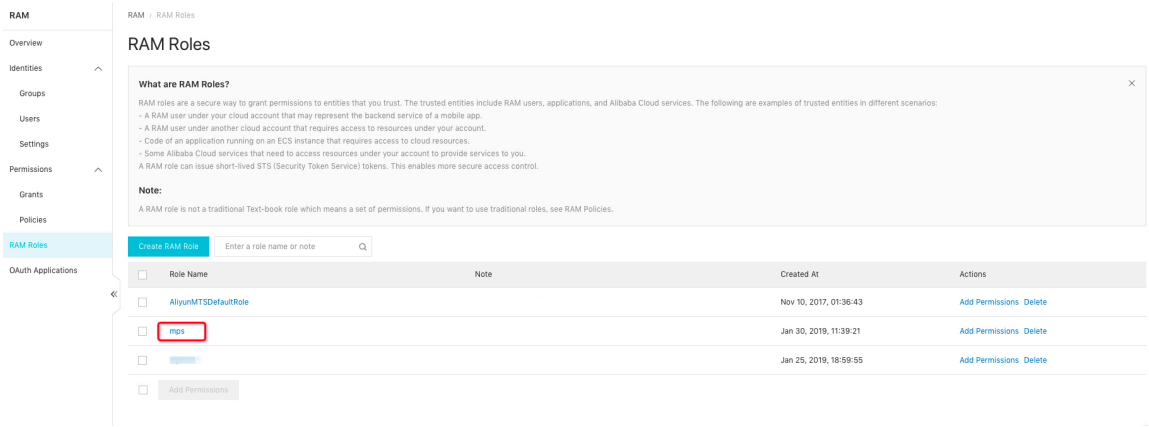
The name can contain a maximum of 64 characters, only English letters, numbers, and hyphens (-) are accepted.

Note

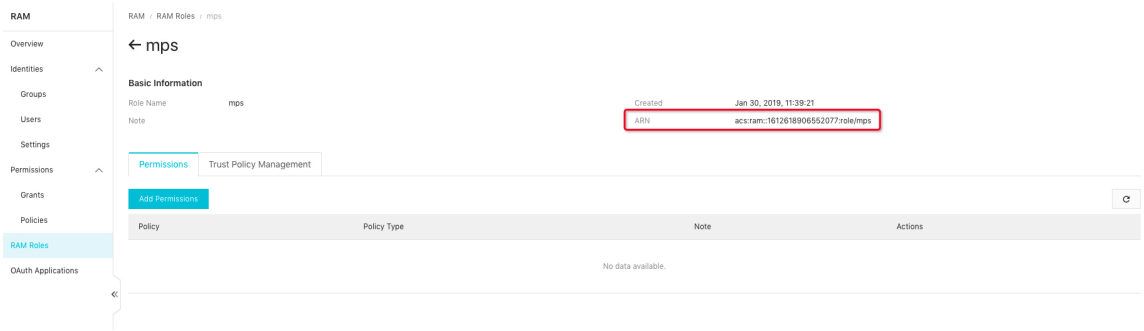
Contact Us

OK Close

d. In RAM Roles, click the created role.



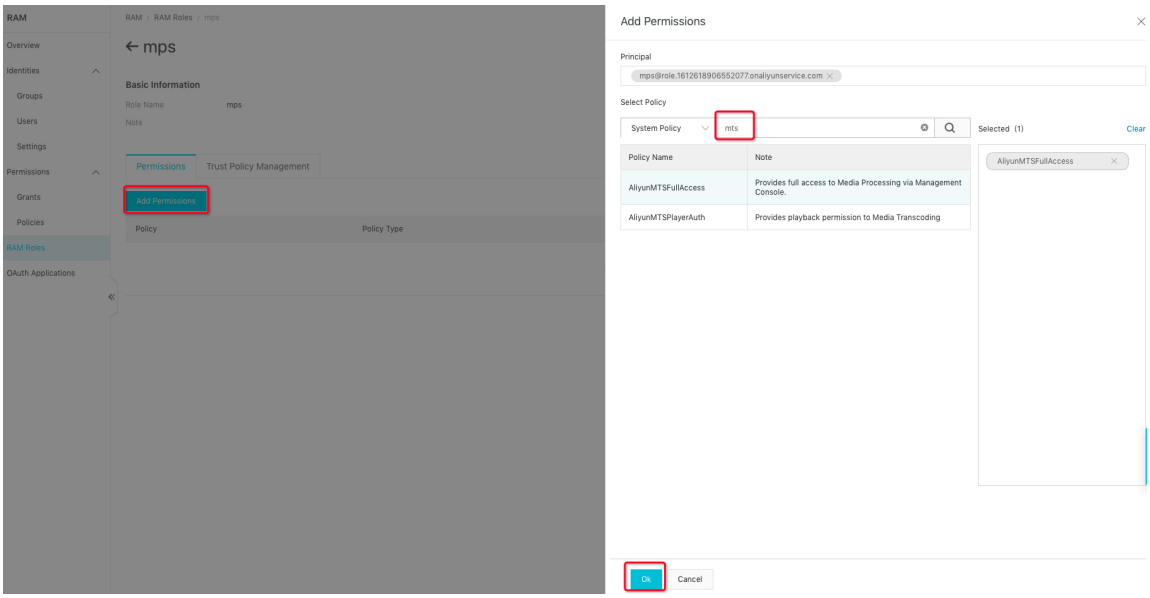
e. In Basic Information, copy ARN parameter acs:ram::1612618906552077:role/mps.



3. Set the role authorization.

a. On the page of the created role, click Add Permissions.

b. Select policy.



Note:

To adjust the STS permissions of the subaccount (for example, to modify, add, or delete a permission), return to this step.

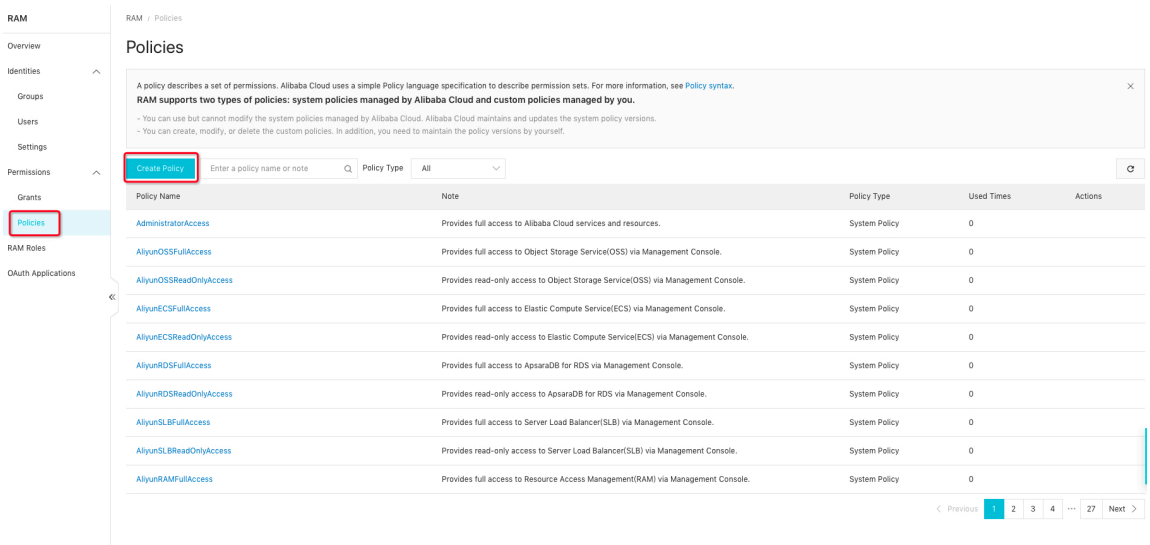
You can create a policy in Custom Policy and add this policy in editing policy to grant the minimum permission required by the upload SDK. The full policy content is as follows:

```
{
     "Statement": [
       {
         "Action": [
           "oss:PutObject",
           "oss:AbortMultipartUpload",
           "oss:ListMultipartUploads",
           "oss:ListParts"
         ],
         "Effect": "Allow",
         "Resource": [
           "*"
         ]
       }
     ],
     "Version": "1"
  }
```

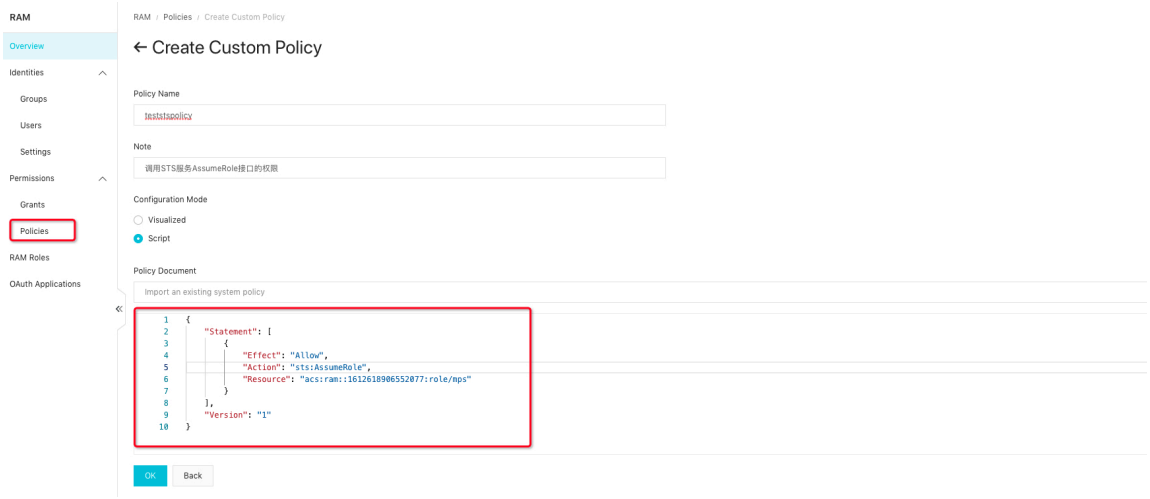
4. Associate the subaccount with the role.

   a. Log on to the RAM console, and click Permissions > Policies in the left-side navigation pane.

   b. Click Create Policy.



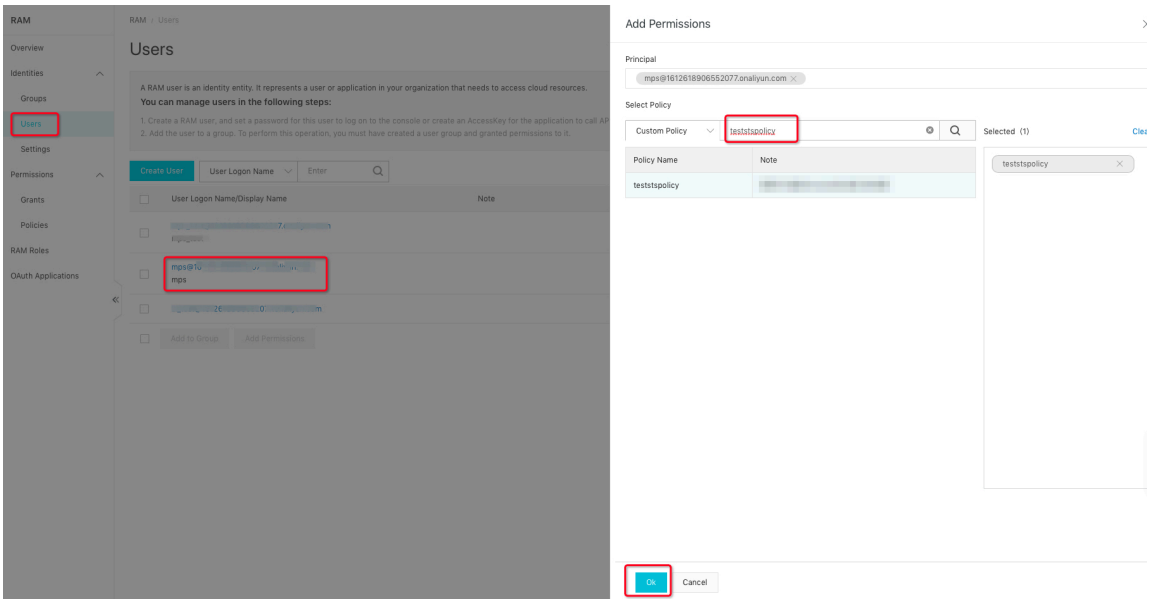
   c. In Create Custom Policy, set Resource field to ARN parameter acs:ram::1612618906552077:role/mps.



d. In the left-side navigation pane, click Identities > Users.

e. Select the subaccount you have set, and click Add Permissions.

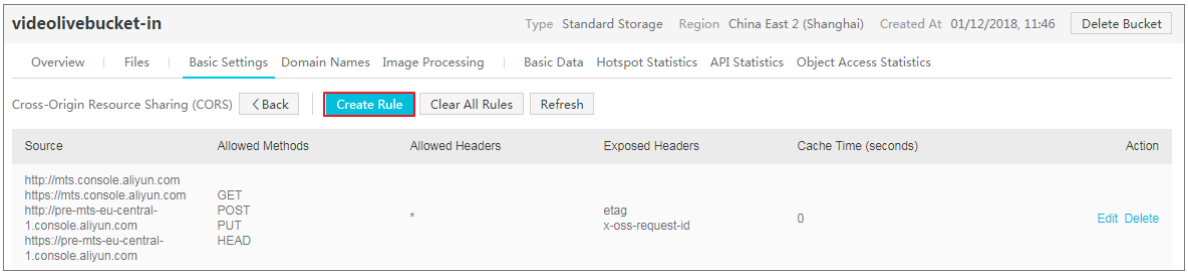
f. Enter the created test policy and `teststspolicy` is displayed.



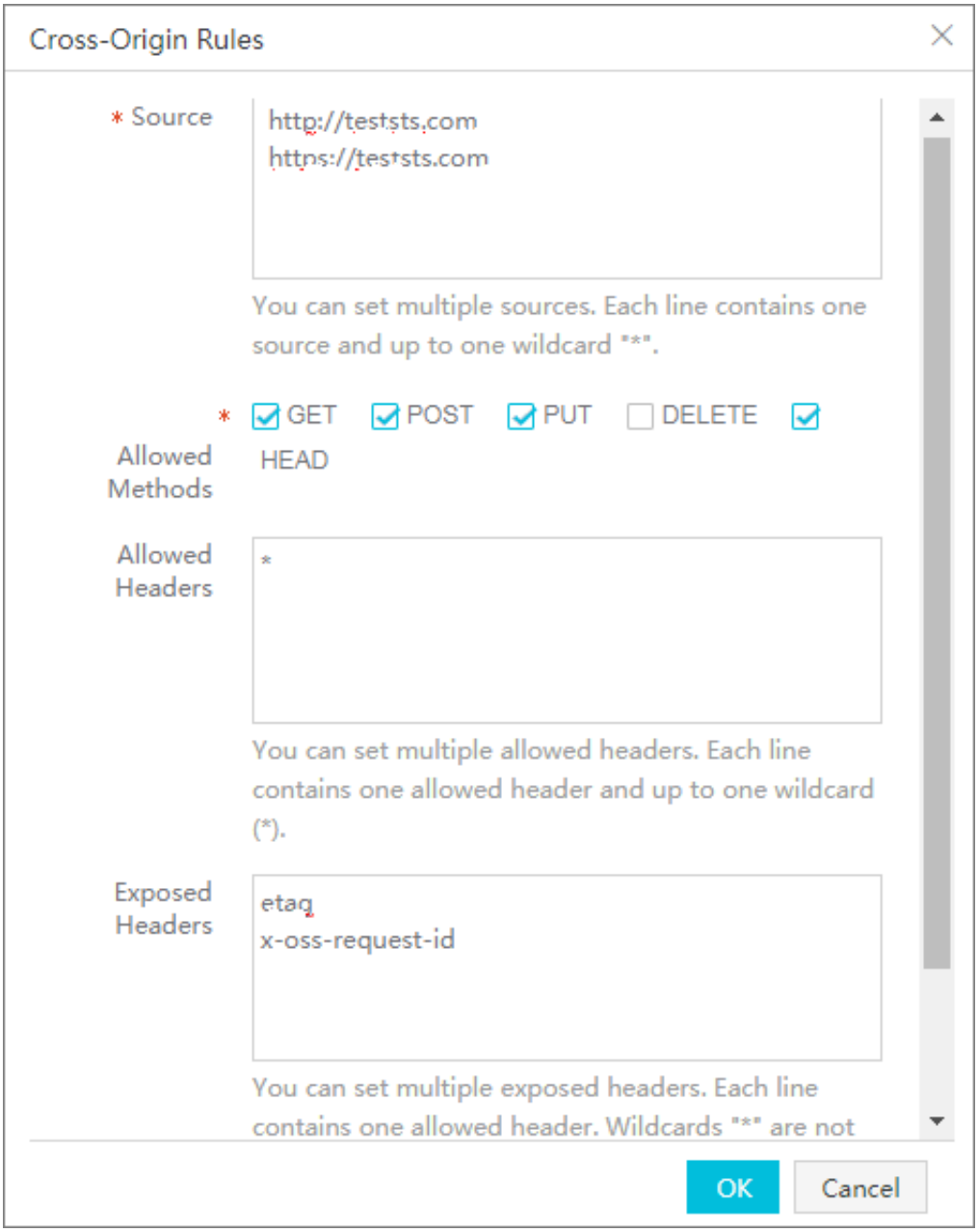
3.3 Set CORS

As JavaScript files and video files are stored in different regions, a cross-region request is initiated when a JavaScript file is uploaded. In this case, cross-region settings must be implemented on OSS. Otherwise, files cannot be uploaded using JavaScript at the web end.

1. Open the CORS settings page of the input bucket.



2. Add a rule.



- Source

  Enter the name of the region in which JavaScript files are deployed. If access through both HTTP and HTTPS is required, add them respectively.

- Method

Select GET, POST, PUT, HEAD.

- Allowed Header

  Enter `*`.

- Expose Header

  Enter etag and x-oss-request-id in two lines.

3.4 Request security token - Java sample code

1. Reference the STS SDK in pom.xml.

```
<repositories>
     <repository>
         <id>sonatype-nexus-staging</id>
         <name>Sonatype Nexus Staging</name>
         <url>https://oss.sonatype.org/service/local/staging/deploy/
maven2/</url>
         <releases>
             <enabled>true</enabled>
         </releases>
         <snapshots>
             <enabled>true</enabled>
         </snapshots>
     </repository>
 </repositories>
<dependencies>
 <dependency>
   <groupId>com.aliyun</groupId>
   <artifactId>aliyun-java-sdk-sts</artifactId>
   <version>2.1.6</version>
 </dependency>
 <dependency>
   <groupId>com.aliyun</groupId>
   <artifactId>aliyun-java-sdk-core</artifactId>
   <version>2.2.0</version>
 </dependency>
</dependencies>
```

2. Code.

   STS requires the role parameter `roleArn`. Log on to the *RAM console*, click Roles, and then click a specific Role Name. The `Arn` parameter is displayed in the basic information, for example, `1351140512345678:role/teststs`.

   - Main Function.

```
public static void main(String[] args) throws Exception {
   IClientProfile profile = DefaultProfile.getProfile(
                                          "cn-hangzhou",
                                          <accessKeyId>,
                                          <accessKeySecret>);
   DefaultAcsClient client = new DefaultAcsClient(profile);
   AssumeRoleResponse response = assumeRole(client, <roleArn>);
```

```
    AssumeRoleResponse.Credentials credentials = response.
getCredentials();
    System.out.println(credentials.getAccessKeyId() + "\n" +
                       credentials.getAccessKeySecret() + "\n" +
                       credentials.getSecurityToken() + "\n" +
                       credentials.getExpiration());
}
```

- Function that generates the temporary AccessKey and token.

```
private static AssumeRoleResponse assumeRole(
                                  DefaultAcsClient client,
                                  String roleArn)
                                  throws ClientException {
    final AssumeRoleRequest request = new AssumeRoleRequest();
    request.setVersion("2015-04-01");
    request.setMethod(MethodType.POST);
    request.setProtocol(ProtocolType.HTTPS);
    request.setDurationSeconds(900L);
    request.setRoleArn(roleArn);
    request.setRoleSessionName("test-token");
    return client.getAcsResponse(request);
}
```

3. Token validity period.

   The token generated in the sample code is valid for 900s, which can be adjusted as required (ranging from 900s to 3600s).

   You can use a generated token in the validity period, instead of repeatedly generating new tokens. The following example shows how to check whether a token needs to be generated again.

```
private static boolean isTimeExpire(String expiration) {
     Date nowDate = new Date();
     Date expireDate = javax.xml.bind.DatatypeConverter.parseDateT
ime(expiration).getTime();
     if (expireDate.getTime() <= nowDate.getTime()) {
         return true;
     } else {
         return false;
     }
}
```

4 Receive message notifications

4.1 Overview

Message format

When media workflow execution starts or completes, a message is sent to the queue or topic (notification) specified by MNS.

- Format definition

  A message body is in JSON format. For details about the field names, types, and descriptions, see Media workflow message in *AddMedia*.

  The structure layers are defined as follows:

  - Top layer

    It is a JSON object. Definition:

    {`Basic attribute of the current activity`, `object to be executed by the workflow`}

  - `Basic attribute of the current activity`

    It is a top-layer key value attribute, rather than an independent object. See the following example. Definition:

    Workflow execution ID, activity name, activity type, activity state, error information.

  - `Details of the object to be executed by the workflow`

    It is a JSON object. Definition:

    {Workflow execution ID, media workflow ID, media workflow name, media ID, input file, workflow execution type, `activity object array`, creation time}.

  - `Activity object array`

    It is a JSON array, containing all activities executed to the current state. For example, a start message contains only the Start activity object, while a completion message contains all activity objects. Definition:

    [`Activity object`, `activity object`…]

- `Activity object`

  It is a JSON object. Definition:

  {Activity name, activity type, task ID, activity state, start time, end time, error information}.

- Start

  "Activity type" in `activity basic attribute` is "Start".

- Complete

  "Activity type" in `activity basic attribute` is "Report".

- Example:

```
{
      "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
      "Name": "Act-4",
      "Type": "Report",
      "State": "Success",
      "MediaWorkflowExecution": {
        "Name": "ConcurrentSuccess",
        "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
        "Input": {
            "InputFile": {
                "Bucket": "inputfirst",
                "Location": "oss-test",
                "Object": "mediaWorkflow/ConcurrentSuccess/01.wmv"
            }
        },
        "State": "Success",
        "MediaId": "2be491ab4cb6499cd0befe5fcf0cb670",
        "ActivityList": [
            {
                "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
                "Name": "Act-1",
                "Type": "Start",
                "State": "Success",
                "StartTime": "2016-03-15T02: 53: 41Z",
                "EndTime": "2016-03-15T02: 53: 41Z"
            },
            {
                "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
                "Name": "Act-2",
                "Type": "Transcode",
                "JobId": "f34b6d1429dd491faa7a6c1c8f905285",
                "State": "Success",
                "StartTime": "2016-03-15T02: 53: 43Z",
                "EndTime": "2016-03-15T02: 53: 47Z"
            },
            {
                "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
                "Name": "Act-3",
                "Type": "Snapshot",
                "JobId": "c14150be33304825a5d67cd5364c35cb",
                "State": "Success",
                "StartTime": "2016-03-15T02: 53: 44Z",
                "EndTime": "2016-03-15T02: 53: 45Z"
```

```
            },
            {
                "RunId": "8f8aba5a62ab4127ae2add18da20b0f2",
                "Name": "Act-4",
                "Type": "Report",
                "State": "Success",
                "StartTime": "2016-03-15T02: 53: 49Z",
                "EndTime": "2016-03-15T02: 53: 49Z"
            }
        ],
        "CreationTime": "2016-03-15T02: 53: 39Z"
    }
}
```

How to receive and resolve a message

- Queue

  PHP sample code

- Topic (notification)

  PHP sample code

4.2 Receive notification through queues

This section briefly introduces the requirements and installation instructions of MNS.

For more information, see the MNS documentation SDK download and Queue user manual.

The example language is PHP. For more information about the usage instructions of other languages, see the MNS documentation SDK user manual.

Environment requirements

PHP 5.5+

Installation

Download the MNS SDK for PHP SDK from Alibaba Cloud.

Download the MNS SDK for PHP SDK from Alibaba Cloud.

The example language is PHP. For more information about the usage instructions of other languages, see SDK user manual.

Decompress the file to the project directory. The decompressed directory is `php_sdk`。

Sample code

- Reference the MNS SDK

```
require_once(dirname(__FILE__).'/php_sdk/mns-autoloader.php');
```

- Initialize MNS

  MNS configures an independent service domain name for each region of users. The rule is `https://${UserId}.mns. ${Region}.aliyuncs.com`. China East 1 (Hangzhou) (cn-hangzhou)is used in the following example. You can also use another region, for example, China North 2 (Beijing) (cn-beijing).

```
use AliyunMNS\Client;
 use AliyunMNS\Exception\MnsException;
```

```
$mns_client = new Client('https://'.$user_id.'.mns.cn-hangzhou.
aliyuncs.com',
                          $access_key_id, $access_key_secret);
 $queue = $mns_client->getQueueRef($queue_name);
```

- Receive a message

  Each message received by MNS corresponds to a handle, which can be used later to operate the message (for example, delete the message).

  In addition, MNS supports receiving messages in batches to improve the performance. For more information, see MNS documentation BatchRecei veMessage.

  A timeout time can be specified when a message is received. (The timeout time is set to 3s in the following example.) If no message exists in the queue, timeout occurs and an exception is returned.

```
$receipt_handle = NULL;
 $message = null;
 try
 {
      $res = $queue->receiveMessage(3);
      echo "ReceiveMessage Succeed! \n";
      $message = $res->getMessageBody();
      $receipt_handle = $res->getreceiptHandle();
 }
 catch (MnsException $e)
 {
      echo "ReceiveMessage Failed: " . $e . "\n";
 }
```

- Delete a message

A message is not actively deleted from a queue. You must call DeleteMessage to delete the message. Otherwise, the message is always in the queue, and you will receive the same message next time. In addition, DeleteMessage can be called successfully only within the specified time after the message is received. For more information, see MNS - DeleteMessage.

```
try
 {
      $res = $queue->deleteMessage($receipt_handle);
      echo "DeleteMessage Succeed! \n";
 }
 catch (MnsException $e)
 {
      echo "DeleteMessage Failed: " . $e . "\n";
 }
```

- Analyze a message

  The message body is a string while the content is a JSON object. After converting the string to the object using `json_decode`, you can analyze the JSON object to obtain details of the message. The output file that triggers media workflow execution is printed in the following example.

```
$json_message = json_decode($message);
 $input_file = $json_message->{'MediaWorkflowExecution'}->{'Input'}-
>{'InputFile'};
 echo 'input_filelocation:'.$input_file->{'Location'}.
      ' bucket:'.$input_file->{'Bucket'}.
      ' object:'.$input_file->{'Object'}."\n";
```

- Obtain video output details

  After obtaining details of a message, you can use the media library API to obtain details of a video executed by a workflow. The output URL of the transcoding and screenshot tasks is printed in the following example.

  For more information about how to install and configure the SDK for PHP of the media library, see *Media Library SDK-PHP*.

```
include_once 'aliyun-php-sdk-core/Config.php';
 use Mts\Request\V20140618 as Mts;
```

Initialize the client of the media library.

```
$profile = DefaultProfile::getProfile('cn-hangzhou',
                                      $access_key_id,
                                      $access_key_secret);
```

```
 $mts_client = new DefaultAcsClient($profile);
```

Print the output URLs and basic information of all transcoding tasks.

```
If (strbmp ($ json_message-> {'type'}, 'report ') = 0 ){
      $activities = $json_message->{'MediaWorkflowExecution'}->{'
ActivityList'};
      $transcode_job_ids = Array();
      for ($i = 0; $i < count($audioStreams); $i++) {
        if (strcmp($activities[$i]->{'Type'}, 'Transcode') == 0) {
          $transcode_job_ids[] = $activities[$i]->{'JobId'};
        }
      }
      $request = new Mts\QueryJobListRequest();
      $request->setJobIds(join(',', $transcode_job_ids));
      $request->setRegionId('cn-hangzhou');
      $response = $mts_client->getAcsResponse($request);
      for ($i=0; $i < count($response->{'JobList'}->{'Job'}); $i++)
 {
        $output = $response->{'JobList'}->{'Job'}[$i]->{'Output'};
        $output_file = $response->{'JobList'}->{'Job'}[$i]->{'Output
'}->{'OutputFile'};
        $video_properties = $response->{'JobList'}->{'Job'}[$i]->{'
Output'}->{'Properties'};
        echo 'URLs of the transcoding output files'.'http://'.$
output_file->{'Bucket'}.'.'.
                        $output_file->{'Location'}.'.aliyuncs.com/'.
                        urldecode($output_file->{'Object'})."\n";
        echo 'basic information of the transcoding output files'.$
video_properties->{'Width'}.'x'.$video_properties->{'Height'}.
                        ' duration:'.$video_properties->{'Duration
'}."\n";
      }
 }
```

Print the output URLs of all screenshot tasks.

```
if (strcmp($json_message->{'Type'}, 'Report') == 0) {
      $activities = $json_message->{'MediaWorkflowExecution'}->{'
ActivityList'};
      $snapshot_job_ids = Array();
      for ($i = 0; $i < count($audioStreams); $i++) {
        if (strcmp($activities[$i]->{'Type'}, 'Snapshot') == 0) {
          $snapshot_job_ids[] = $activities[$i]->{'JobId'};
        }
      }
      $request = new Mts\QuerySnapshotJobListRequest();
      $request->setSnapshotJobIds(join(',', $snapshot_job_ids));
      $request->setRegionId('cn-hangzhou');
      $response = $mts_client->getAcsResponse($request);
      for ($i=0; $i < count($response->{'SnapshotJobList'}->{'
SnapshotJob'}); $i++) {
        $snapshot_config = $response->{'SnapshotJobList'}->{'
SnapshotJob'}[$i]->{'SnapshotConfig'};
        $output_file = $response->{'SnapshotJobList'}->{'SnapshotJob
'}[$i]->{'SnapshotConfig'}->{'OutputFile'};
        echo 'URLs of the screenshot output files'.'http://'.$
output_file->{'Bucket'}.'.'.
                        $output_file->{'Location'}.'.aliyuncs.com/'.
                        urldecode($output_file->{'Object'})."\n";
      }
```

```
}
```

4.3 Receive message through topic notification

MNS actively pushes message notifications by topic to users. You can conveniently receive the messages if an HTTP service can be publicly accessed.

Basic structure

The basic structure of a video media repository notification is as follows:

- The outermost layer is the structure body of MNS.

  For more information about the definition and format of MNS, see MNS - Notificati on operations.

- The `message body` of MNS is the structure body of the media repository.

  After receiving the `message body`, MNS further resolves the message of the media repository. For more information about the resolution steps and sample codes, see *Receive a message in queue mode*.

Security

The topic (notification) mode is convenient. However, as the HTTP service can be publicly accessed, illegal calls and attacks must be prevented. For more informatio n about how to identify whether a message is initiated by MNS, see the MNS documentation Endpoint signature authentication.

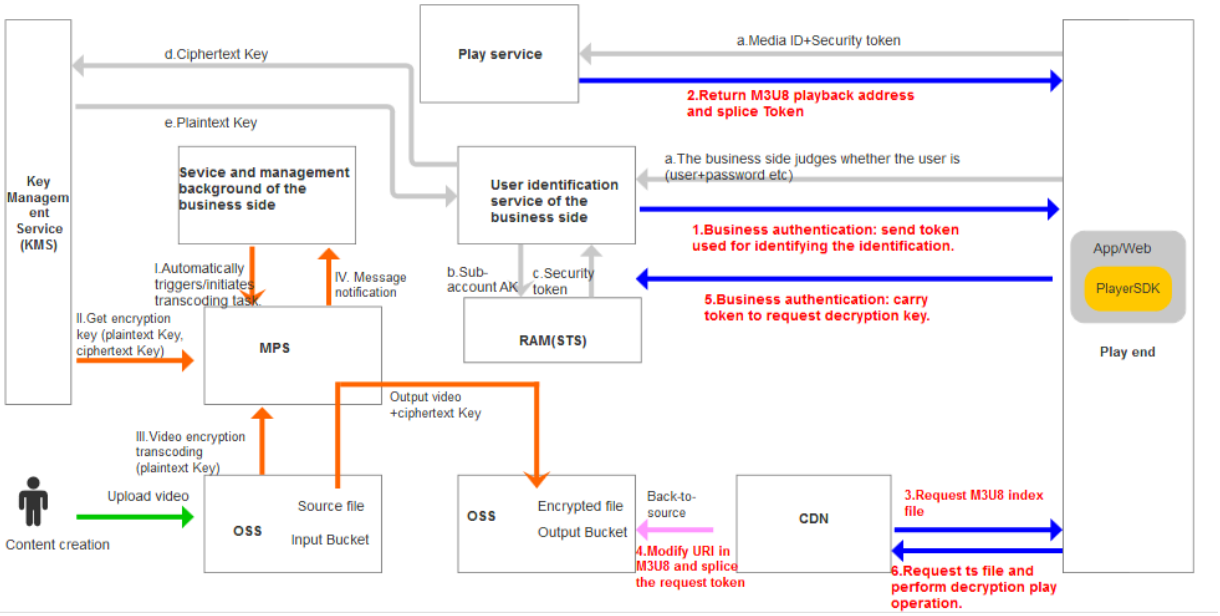
5 Video encryption

5.1 HLS standard encryption

Video encryption is a measure to protect the video content. Encrypting the video content can effectively avoid video leaks and leeching problems, thus being widely used in online education, finance and economics and other fields.

> **Note:**
> Alibaba Cloud currently supports encryption in two ways. One is private encryption, and the other is HLS standard encryption. Using HLS standard encryption, users must protect the encryption key. This document introduces HLS standard encryption.

Complete encryption architecture



Terms

- Key Management Service (KMS)

  A security management service, mainly responsible for the production, encryption and deceyption of data key and other operations. Click here to activate *KMS service*.

- Data Key (DK), also called plaintext key

  DK is plaintext data key used in data encryption.

- Enveloped Data Key (EDK), also called ciphertext key.

  EDK is the ciphertext data key, encrypted by using envelope encryption technology .

- Resource Access Management (RAM)

  User identification management and resource access management service provided by Alibaba Cloud. Click here to activate *RAM service*.

Procedure

1. Create HLS encryption workflow.

   > **Note:**
   > The console currently does not support creating HLS encryption workflow. You can create HLS encryption workflow by using API. For more information about demo, see *Create HLS standard encryption workflow*. After creating, the workflow cannot be modified on the console, or the encryption setting goes invalid.

   Key settings in workflow:

   - Start activity node: `InputFile:{"Bucket":"bucketdemo", "Location ":"oss-cn-hangzhou", "ObjectPrefix":"HLS-Encryption"}`

     This setting indicates the content creater uploads a video under this path oss ://bucketdemo/HLS-Encryption to Hangzhou, and encryption transcoding is triggered automatically.

   - Transcoding activity node: `Encryption:{"Type":"hls-aes-128", "KeyUri":" https://decrypt.demo.com"}`

     After transcoding operation is completed, the setting of KeyUri appears in m3u8 file for player to use.

     During play, the player carries EDK ciphertext key to request the address so as to get DK plaintext key for play.

2. Upload video.

   Either way to upload video can trigger encryption transcoding automatically.

   - Upload the video to the created workflow by using the MPS console.
   - Upload the video under the path oss://bucketdemo/HLS-Encryption by using OSS uploading tool.

After transcoding is completed, the content of the m3u8 file is showned as follows.

```
#EXTM3U
     #EXT-X-VERSION:3
     #EXT-X-TARGETDURATION:5
     #EXT-X-MEDIA-SEQUENCE:0
     #EXT-X-KEY:METHOD=AES-128,URI="https://decrypt.demo.com?
Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271"
     #EXTINF:4.127544,
     15029611683170-00001.ts
     #EXT-X-ENDLIST
```

3. Play.

   - Use the QueryMediaList interface to get playback address. For more information, see *QueryMediaList*. Get the OSS address, replace the OSS domain name with CDN domain name, and splice the parameter`MtsHlsUriToken`, which serves as the token to request the decryption key. The principle is as follows.

     During play, the player accesses the URI in the EXT-X-KEY tag in the m3u8 file to get decryption key. The URI is a decryption key interface built by the business side. Therefore, while requesting decryption, the player must carry some authentication information recognized by the business side. MtsHlsUriToken plays the role in a similar way. The business side issues a token to the player, which carries the token when requesting the decryption key, and the business side checks the validity of the token.
   - The player carry the token to the business side for authentication service.

     For example, the normal playback address is `https://vod.demo.com/test.m3u8`. Splice and carry the parameter `MtsHlsUriToken`, the playback address is `https://vod.demo.com/test.m3u8?MtsHlsUriToken=Token issued by the business side`.

     During playback, the player request`https://vod.demo.com/test.m3u8?MtsHlsUriToken=Token issued by the business side` to CDN of Alibaba Cloud, and the CDN of Alibaba Cloud dynamically modifies the decryption URI in the m3u8 file. For example, the original `https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271` is modified to `https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271&MtsHlsUriToken=Token issued by the business side`.

Therefore, the final decryption URI which the player requests is `https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271&MtsHlsUriToken=Token issued by the business side`. This address carries the token issued by the business side, which can be identified by the business side.

4. The business side need to do the following operations.
   a. Build, issue and identify MtsHlsUriToken service.
   b. Identify decryption token. One token is allowed to use only once.
   c. Decrypt key: EDK, which is Ciphertext, calls decryption interface of the KMS service for decryption. For more information, see *Decrypt*. After decryption, the information can be cached to reduce network IO.
   d. After decryption, you can get DK (the plaintext key) which needs base64decodd, and return it to the player.

6 Media library management

6.1 Overview

You can access the media library by using the MPS SDK for *Java*, *PHP*, and *Python*.

You can also access the media library through HTTP/HTTPS. For more information, see *API reference*.

Functions

Media workflow management: Allows you to add, delete, modify, query, activate, and stop a media workflow.

Management of media workflow execution instances: Allows you to traverse and query execution instances.

Media management: Allows you to add, delete, modify, query, search for a media resource, maintain attributes (the title, tag, cover, and description) of a media set, and set the publishing status of a media set.

Media category management: Allows you to add, delete, modify, and query a media category.

Service scenarios

- Search for a media set

  Search for a media set that meets search criteria in the media library.

  You can use keywords to search for a media set. With logical disjunction, a media set is displayed if and only if one or more of the title, tag, description, and category are matched. With logical conjunction, a media set is displayed if and only if all specified attributes (the title, tag, description, and category) are matched.

  In the search criteria, you can specify the creation time range to limit the search range. You can also set whether the return results are sorted by creation time in ascending or descending order.

  In addition, if many APIs are to be returned, you can have them displayed in pages.
- Maintain attributes of a media set

Each media set contains basic attributes of the title, tag, description, and category, which can be set using APIs.

Basic attributes - Sample code - PHP

- Manage tags of a media set

  Each tag is specific to a media set. No tag can be set for a media library globally. However, you can use API for searching for media sets to query all media sets with the same tags.

  Manage tags - Sample code - PHP

Manage the category of a media set

The media library provides global category management. You can associate each media set with a category and quickly retrieve a media set.

Manage categories - Sample code - PHP

Query details of a media set

A media set contains an input file and several output files (videos and screenshots). You can query the detailed input and output information of a returned media set.

Input information includes the basic attributes (width, height, duration, size, bit rate , and frame rate) and details (container encapsulation, video, audio, subtitle stream, and detailed attributes of the encapsulation and stream) of a video.

Output information includes the basic attributes (width, height, duration, size, bit rate, and frame rate), OSS URL of a video as well as the type (single-frame and batch) and the OSS URL of a screenshot.

Media set details - Sample code -PHP

6.2 Basic video attributes

Overview

The following example describes how to query and update the basic information of a media set. For more information about how to install and use the SDK, see *Media library SDK-PHP*.

Query the basic information of a media set

You can use the media ID or OSS file URL to query a media set.

- **Query a media set by media ID**

  For more information about the parameters, see *API reference > Media APIs > Query media sets by media IDs*.

```
include_once 'aliyun-php-sdk-core/Config.php';
use Mts\Request\V20140618 as Mts;
$accessKeyID = 'test'; //eplace the value with your AccessKeyID
$accessKeySecret = 'test'; //Replace the value with your AccessKeyS
ecret
$profile = DefaultProfile::getProfile('cn-hangzhou',
                                      $accessKeyID,
                                      $accessKeySecret);
$client = new DefaultAcsClient($profile);
```

```
function queryMediaById($client, $mediaID)
{
     $request = new Mts\QueryMediaListRequest();
     $request->setAcceptFormat('JSON');
     $request->setMediaIds($mediaID);
     $response = $client->getAcsResponse($request);
     return $response;
}
function printMedia($media)
{
     if (array_key_exists('Title', $media)) {
       print_r('Title: '.$media->{'Title'}."\n");
     }
     if (array_key_exists('Description', $media)) {
       print_r('Description: '.$media->{'Description'}."\n");
     }
     if (array_key_exists('Tags', $media)) {
       print_r('Tags: '.$media->{'Tags'}->{'Tag'}[0]."\n");
     }
     if (array_key_exists('CoverURL', $media)) {
       print_r('CoverURL: '.$media->{'CoverURL'}."\n");
     }
     print_r('Format: '.$media->{'Format'}."\n");
     print_r('Resolution: '.$media->{'Width'}.'x'.$media->{'Height
'}."\n");
     print_r('FileSize: '.$media->{'Size'}."\n");
     print_r('Bitrate: '.$media->{'Bitrate'}."\n");
     print_r('FPS: '.$media->{'Fps'}."\n");
}
$mediaID = 'test'; // Replace the value with your desired media ID
$medias = queryMediaById($client, $mediaID)->{'MediaList'}->{'Media
'};
for ($i=0; $i < count($medias); $i++) {
     printMedia($medias[$i]);
}
```

- **Query a media set by an OSS file URL**

  For more information about the parameters, see *API reference > Media APIs > Query media sets by URLs*.

```
function queryMediaByURL($client, $mediaURL)
{
     $request = new Mts\QueryMediaListByURLRequest();
```

```
    $request->setAcceptFormat('JSON');
    $request->setFileURLs($mediaURL);
    $response = $client->getAcsResponse($request);
    return $response;
}
$ossEndpoint = 'http://test.oss-cn-hangzhou.aliyuncs.com/';
// An OSS object does not have to start with "/". Replace the value
with your OSS object
$ossObject = 'test/test.mp4';
$medias = queryMediaByURL($client,$ossEndpoint.urlencode($ossObject
))->{'MediaList'}->{'Media'};
for ($i=0; $i < count($medias); $i++) {
    printMedia($medias[$i]);
}
```

- **Update attributes**

  You can update full attributes or a single attribute.

  - **Full attribute update**

    For more information about about the parameters, see *API reference > Media APIs > Update media set basic information*.

    Specify all fields when updating attributes. Fields not set are cleared.

```
function updateMediaAllField($client, $mediaID, $title, $
description, $tags, $coverURL)
{
    $request = new Mts\UpdateMediaRequest();
    $request->setAcceptFormat('JSON');
    $request->setMediaId($mediaID);
    $request->setTitle($title);
    $request->setCateId(2663987);
    $request->setDescription($description);
    $request->setTags($tags);
    $request->setCoverURL($coverURL);
    $response = $client->getAcsResponse($request);
    return $response;
}
$mediaID = 'test'; //Replace the value with your desired media ID
$media = updateMediaAllField($client, $mediaID,
                        'title', 'description', 'tags', 'coverURL
')->{'Media'};
```

  - **Single attribute update**

    You can use different APIs to conveniently update single fields without modifying other fields.

    The following section uses the "publishing state" as an example. For more information about the parameters, see *API reference > Media APIs > Update media publishing state*.

```
function updateMediaPublishState($client, $mediaID, $state)
{
    $request = new Mts\UpdateMediaPublishStateRequest();
```

```
        $request->setAcceptFormat('JSON');
        $request->setMediaId($mediaID);
        $request->setPublish($state);
        $response = $client->getAcsResponse($request);
        return $response;
}
$mediaID = 'test'; //Replace the value with your desired media ID
//No result is returned from the API that updates the publishing
state. Capture exceptions to check whether execution succeeds
try {
        updateMediaPublishState($client, $mediaID, "true");
} catch(ClientException $e) {
        print_r('ClientException:'."\n");
        print_r($e);
} catch (ServerException $e) {
        print_r('ServerException:'."\n");
        print_r($e);
}
```

6.3 Media details

Overview

For more information about how to install and use the SDK, see *Media library SDK-PHP*.

A media set contains an input file and several output files. Besides basic information, an input file contains detailed *Media set information*. You can query details about the Videos and Screenshots in the output files.

Input

```
include_once 'aliyun-php-sdk-core/Config.php';
 use Mts\Request\V20140618 as Mts;
 $accessKeyID = 'test'; //eplace the value with your AccessKeyID
 $accessKeySecret = 'test'; //Replace the value with your AccessKeyS
ecret
 $profile = DefaultProfile::getProfile('cn-hangzhou',
                                       $accessKeyID,
                                       $accessKeySecret);
 $client = new DefaultAcsClient($profile);
```

```
function queryMedia($client, $mediaID)
{
    $request = new Mts\QueryMediaListRequest();
    $request->setAcceptFormat('JSON');
    $request->setMediaIds($mediaID);
    $request->setIncludeMediaInfo("true");
    $response = $client->getAcsResponse($request);
    return $response;
}
function printMediaInfo($mediaInfo)
{
    print_r('Number of Streams: '.$mediaInfo->{'Format'}->{'NumStreams
'}."\n");
    if (array_key_exists('Streams', $mediaInfo) &&
        array_key_exists('AudioStreamList', $mediaInfo->{'Streams'})
 &&
```

```
        array_key_exists('AudioStream', $mediaInfo->{'Streams'}->{'
AudioStreamList'})) {
        $audioStreams = $mediaInfo->{'Streams'}->{'AudioStreamList'}-
>{'AudioStream'};
        print_r('Audio Streams:'."\n");
        for ($i = 0; $i < count($audioStreams); $i++) {
            print_r("\t[".$i."]"."\n");
            print_r("\t\tCodecName: ".$audioStreams[$i]->{'CodecName
'}."\n");
            print_r("\t\tChannels: ".$audioStreams[$i]->{'Channels
'}."\n");
            print_r("\t\tSamplerate: ".$audioStreams[$i]->{'Samplerate
'}."\n");
            print_r("\t\tDuration: ".$audioStreams[$i]->{'Duration
'}."\n");
            print_r("\t\tBitrate: ".$audioStreams[$i]->{'Bitrate'}."\n
");
        }
    }
    if (array_key_exists('Streams', $mediaInfo) &&
        array_key_exists('VideoStreamList', $mediaInfo->{'Streams'})
 &&
        array_key_exists('VideoStream', $mediaInfo->{'Streams'}->{'
VideoStreamList'})) {
        $videoStreams = $mediaInfo->{'Streams'}->{'VideoStreamList'}-
>{'VideoStream'};
        print_r('Video Streams:'."\n");
        for ($i = 0; $i < count($videoStreams); $i++) {
            print_r("\t[".$i."]"."\n");
            print_r("\t\tCodecName: ".$videoStreams[$i]->{'CodecName
'}."\n");
            print_r("\t\tProfile: ".$videoStreams[$i]->{'Profile'}."\n
");
            print_r("\t\tDuration: ".$videoStreams[$i]->{'Duration
'}."\n");
            print_r("\t\tPixFmt: ".$videoStreams[$i]->{'PixFmt'}."\n
");
            print_r("\t\tFps: ".$videoStreams[$i]->{'Fps'}."\n");
            print_r("\t\tBitrate: ".$videoStreams[$i]->{'Bitrate'}."\n
");
            print_r("\t\tResolution: ".$videoStreams[$i]->{'Width'}.'x
'.$videoStreams[$i]->{'Height'}."\n");
        }
    }
}
$mediaID = 'test'; //Replace the value with your desired media ID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printMediaInfo($medias[$i]->{'MediaInfo'});
}
```

Output

- **Videos**

```
function queryMedia($client, $mediaID)
{
    $request = new Mts\QueryMediaListRequest();
    $request->setAcceptFormat('JSON');
    $request->setMediaIds($mediaID);
    $request->setIncludePlayList("true");
    $response = $client->getAcsResponse($request);
```

```
    return $response;
}
function printOutputVideos($videos)
{
    print_r('Number of Output Video: '.count($videos)."\n");
    for ($i = 0; $i < count($videos); $i++) {
        print_r("\t[".$i."]"."\n");
        print_r("\t\tMediaWorkflowName: ".$videos[$i]->{'MediaWorkf
lowName'}."\n");
        print_r("\t\tActivityName: ".$videos[$i]->{'ActivityName
'}."\n");
        print_r("\t\tFormat: ".$videos[$i]->{'Format'}."\n");
        print_r("\t\tDuration: ".$videos[$i]->{'Duration'}."\n");
        print_r("\t\tFps: ".$videos[$i]->{'Fps'}."\n");
        print_r("\t\tBitrate: ".$videos[$i]->{'Bitrate'}."\n");
        print_r("\t\tSize: ".$videos[$i]->{'Size'}."\n");
        print_r("\t\tResolution: ".$videos[$i]->{'Width'}.'x'.$
videos[$i]->{'Height'}."\n");
        print_r("\t\tURL: ".$videos[$i]->{'File'}->{'URL'}."\n");
    }
}
$mediaID = 'test'; //Replace the value with your desired media ID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printOutputVideos($medias[$i]->{'PlayList'}->{'Play'});
}
```

- **Screenshots**

```
function queryMedia($client, $mediaID)
{
    $request = new Mts\QueryMediaListRequest();
    $request->setAcceptFormat('JSON');
    $request->setMediaIds($mediaID);
    $request->setIncludeSnapshotList("true");
    $response = $client->getAcsResponse($request);
    return $response;
}
function printOutputSnapshots($snapshots)
{
    print_r('Number of Output Snapshot: '.count($snapshots)."\n");
    for ($i = 0; $i < count($snapshots); $i++) {
        print_r("\t[".$i."]"."\n");
        print_r("\t\tMediaWorkflowName: ".$snapshots[$i]->{'
MediaWorkflowName'}."\n");
        print_r("\t\tActivityName: ".$snapshots[$i]->{'ActivityName
'}."\n");
        print_r("\t\tType: ".$snapshots[$i]->{'Type'}."\n");
        print_r("\t\tCount: ".$snapshots[$i]->{'Count'}."\n");
        print_r("\t\tURL: ".$snapshots[$i]->{'File'}->{'URL'}."\n");
    }
}
$mediaID = 'test'; //Replace the value with your desired media ID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printOutputSnapshots($medias[$i]->{'SnapshotList'}->{'Snapshot
'});
```

```
}
```

6.4 Tag management

Overview

For more information about how to install and use the SDK, see *Media library SDK-PHP*.

The media repository does not provide global tag management and setting. Tags of each media set are independent. You can search for APIs of a media set to query all media sets that have the same tags.

The tag-related APIs support addition and deletion of a single tag. You can use *UpdateMedia* to set multiple tags at a time.

Add a tag

For more information about the parameters, see *API reference > Media APIs > Add a media tag*.

```
include_once 'aliyun-php-sdk-core/Config.php';
  use Mts\Request\V20140618 as Mts;
  $accessKeyID = 'test'; //eplace the value with your AccessKeyID
  $accessKeySecret = 'test'; //Replace the value with your AccessKeyS
ecret
  $profile = DefaultProfile::getProfile('cn-hangzhou',
                                        $accessKeyID,
                                        $accessKeySecret);
  $client = new DefaultAcsClient($profile);
```

```
function addMediaTag($client, $mediaID, $tag)
  {
      $request = new Mts\AddMediaTagRequest();
      $request->setAcceptFormat('JSON');
      $request->setMediaId($mediaID);
      $request->setTag($tag);
      $response = $client->getAcsResponse($request);
      return $response;
  }
  $mediaID = 'test'; //Replace the value with your desired media ID
  //No result is returned from the API. Capture exceptions to check
whether execution succeeds
  try {
      addMediaTag($client, $mediaID, "testtag");
  } catch(ClientException $e) {
      print_r('ClientException:'."\n");
      print_r($e);
  } catch (ServerException $e) {
      print_r('ServerException:'."\n");
      print_r($e);
```

```
  }
```

Delete a tag

For more information about the parameters, see *API reference > Media APIs > Delete a media tag*.

```
function deleteMediaTag($client, $mediaID, $tag)
  {
      $request = new Mts\DeleteMediaTagRequest();
      $request->setAcceptFormat('JSON');
      $request->setMediaId($mediaID);
      $request->setTag($tag);
      $response = $client->getAcsResponse($request);
      return $response;
  }
  $mediaID = 'test'; //Replace the value with your desired media ID
  //No result is returned from the API. Capture exceptions to check
whether execution succeeds
  try {
      deleteMediaTag($client, $mediaID, "testtag");
  } catch(ClientException $e) {
      print_r('ClientException:'."\n");
      print_r($e);
  } catch (ServerException $e) {
      print_r('ServerException:'."\n");
      print_r($e);
  }
```

6.5 Category management

Overview

For more information about how to install and use the SDK, see *Media library SDK-PHP*.

You can add, delete, modify, and query a category. In addition, pay attention to the following logic:

- Deleting a category does not automatically clear the category ID of an associated media set.
- The result returned from the category query API can be displayed in the tree structure or list structure. A nested JSON object is returned in the tree structure , while a plane array is returned in the list structure. You can select a structure based on the actual scenario.

Add a category

For more information about the parameters, see *API reference > Media category APIs > Add a category*.

```
include_once 'aliyun-php-sdk-core/Config.php';
```

```
  use Mts\Request\V20140618 as Mts;
  $accessKeyID = 'test'; //eplace the value with your AccessKeyID
  $accessKeySecret = 'test'; //Replace the value with your AccessKeyS
ecret
  $profile = DefaultProfile::getProfile('cn-hangzhou',
                                        $accessKeyID,
                                        $accessKeySecret);
  $client = new DefaultAcsClient($profile);
```

```
function addCategory($client, $parentId, $categoryName)
  {
      $request = new Mts\AddCategoryRequest();
      $request->setAcceptFormat('JSON');
      $request->setParentId($parentId);
      $request->setCateName($categoryName);
      $response = $client->getAcsResponse($request);
      return $response;
  }
  $category = addCategory($client, null, 'testroot')->{'Category'};
  print_r('Level: '.$category->{'Level'}.
          "\tParentId: ".$category->{'ParentId'}.
          "\tCateId: ".$category->{'CateId'}.
          "\tCateName: ".$category->{'CateName'}."\n");
```

Update a category

For more information about the parameters, see *API reference > Media category APIs > Update a category name*.

```
function updateCategory($client, $categoryId, $categoryName)
  {
      $request = new Mts\UpdateCategoryNameRequest();
      $request->setAcceptFormat('JSON');
      $request->setCateId($categoryId);
      $request->setCateName($categoryName);
      $response = $client->getAcsResponse($request);
      return $response;
  }
  try {
      updateCategory($client, 12345678, 'updatetestroot'); //Replace
the value with your category ID
  } catch (ClientException $e) {
      print_r('ClientException:'."\n");
      print_r($e);
  } catch (ServerException $e) {
      print_r('ServerException:'."\n");
      print_r($e);
  }
```

Delete a category

For more information about the parameters, see *API reference > Media category APIs > Delete a category*.

```
function deleteCategory($client, $categoryId)
  {
      $request = new Mts\DeleteCategoryRequest();
      $request->setAcceptFormat('JSON');
      $request->setCateId($categoryId);
```

```
        $response = $client->getAcsResponse($request);
        return $response;
    }
    try {
        deleteCategory($client, 12345678); //Replace the value with your
category ID
    } catch(ClientException $e) {
        print_r('ClientException:'."\n");
        print_r($e);
    } catch (ServerException $e) {
        print_r('ServerException:'."\n");
        print_r($e);
    }
```

Query a category

- **Tree structure**

  For more information about the parameters, see *API reference > Media category APIs > Retrieve a category tree* **.**

```
function queryCategoryTree($client)
    {
        $request = new Mts\CategoryTreeRequest();
        $request->setAcceptFormat('JSON');
        $response = $client->getAcsResponse($request);
        return $response;
    }
    function printCategoryTree($categoryTree)
    {
        foreach($categoryTree as $category) {
            for ($i = 0; $i < $category->{'Level'}; $i++) {
                print_r("--");
            }
            print_r('Level: '.$category->{'Level'}.
                    "\tParentId: ".$category->{'ParentId'}.
                    "\tCateId: ".$category->{'CateId'}.
                    "\tCateName: ".$category->{'CateName'}."\n");
            if (array_key_exists('SubcateList', $category)) {
                printCategoryTree($category->{'SubcateList'});
            }
        }
    }
    $categoryTree = queryCategoryTree($client)->{'CategoryTree'};
    printCategoryTree(json_decode($categoryTree));
```

- **List structure**

  For more information about the parameters, see *API reference > Media category APIs > Retrieve a category list*.

```
 function queryCategoryList($client)
      {
        $request = new Mts\ListAllCategoryRequest();
        $request->setAcceptFormat('JSON');
        $response = $client->getAcsResponse($request);
        return $response  ;
    }
```

```
    $categoryList = queryCategoryList($client)->{'CategoryList'}->{'
Category'};
    for ($i = 0; $i < count($categoryList); $i++) {
        print_r('Level: '.$categoryList[$i]->{'Level'}.
                "\tParentId: ".$categoryList[$i]->{'ParentId'}.
                "\tCateId: ".$categoryList[$i]->{'CateId'}.
                "\tCateName: ".$categoryList[$i]->{'CateName ' }."\n
");
    }
```