Alibaba Cloud ApsaraVideo for Media Processing

Developer Guide

Document Version20190703

目次

1 概念

1.1 タスクと MPS キュー

ここでは、開発者が MPS をより理解したうえでご使用いただけるように、MPS のいくつかの基本概念と関連性について説明します。

MPSの概念についての説明

・ タスク

MPS のタスクは、トランスコードタスク、スクリーンショットタスク、メディア情報タスクなど、各種タスクを含む抽象的概念です。

1 つのタスクには、入力、出力、パラメーターという 3 つの重要な情報が含まれています。 入力パラメーターと出力パラメーターは、完了したタスクの入力ファイルと出力ファイルを設定するために使用されます。 これら 3 つのパラメーターは、特定の機能を実行するため、詳細な構成を設定する際に使用されます。

・ パラメーター

- テンプレートパラメーター

  タスクは大量にあるため、各タスクを何度も繰り返し送信する必要があります。 テンプレートはこの問題を解決するために提案された概念です。 テンプレートでは、よく使うパラメーターがまとめてあります。 よく使うパラメーターをテンプレートとして使用することで、タスクを送信するときに指定する必要があるパラメーターの数を減らすことができます。これにより送信コードを簡単に作成することができます。

- API パラメーター

  各種パラメーターの組み合わせごとにテンプレートを作成すると、テンプレートの数が大幅に増加し、テンプレートの管理がより煩雑になる可能性があります。 そのため、パラメーターはテンプレート内だけでなく、API を通じても設定することができます。

- 優先順位

  API パラメーターはテンプレート内の対応する (同一の) パラメーターよりも高い優先順位を持っており、テンプレートのパラメーターに優先されます。

  たとえば、1 つのビデオをさまざまな解像度 (HD、SD)、定義形式 (MP4)、符号化標 (H.264) およびフレームレートでトランスコードし、出力することができます。 その違いはフレーム レートや解像度のみです。 既定のパラメーターを組み合わせて (MP4 + H.264 +

25FPS +2Mbps +1280x720) テンプレートを作成することができます。 API を呼び出すときに、API パラメーターが指定されていなければ、タスクは既定のパラメーター (2Mbps + 1280x720) に基づいて実行されます。API パラメーター (4Mbps + 1920x1080) が指定されていれば、既定のパラメーターに代わり、タスクは API パラメーター (4Mbps + 1920x1080) に基づいて実行されます。

- MPS キュー

  ユーザーが API インターフェイスを介してタスクを送信すると、タスクはまず MPSキューに入り、優先順位および送信順序に従って実行されます。

  MPS キューのタスクには優先順位をつけることができます (優先順位が最も高いのは 10、最も低いのは 1、既定値は 6 です)。 優先順位が同一の場合は、先に送信されたタスクが先に実行されます。 タスクが同時に送信された場合は、優先順位の高いタスクが先に実行されます。

- タスクの実行と完了

  - 同期と非同期

    タスクのタイプによっては、すぐに完了するものもありますが、基本的にはほとんどのタスクはすぐに完了することはありません。 タスクの実行には、同期と非同期の 2 つの方法があります。 スクリーンショットタスクなどの同期型のタスクは即座に結果を返しますが、トランスコードタスクなどの非同期型のタスクは、ポーリングのスケジュール、およびメッセージ通知による照会を行います。

  - 定期的なポーリング

    各タスクは一意のタスクIDにより識別されます。タスクIDは、タスクの送信と同時に呼び出し元に返されます。その後、タスク ID でタスクの結果を照会することができます。 この方法の欠点は、リアルタイムに実行されないことです。

    通知

    メッセージ通知が送信されるよう MPS キューを設定することで、タスクの結果を即座に取得できるようになります。 メッセージ通知には、タスク ID、ユーザーデータ、タスクの結果といった重要な情報が含まれます。

    ■ タスク ID

      タスクを送信するときは、タスク ID を記録しておきます。記録したタスクID をメッセージ通知のタスクIDと照合し、タスクの結果を確認します。

    ■ ユーザーデータ

      タスクを送信する際、実行するたびにカスタマイズユーザーデータパラメーター (製品 ID など) を入力することができます。 カスタマイズユーザーデータパラメーターは、業

務システムにタスク ID を記録することなく、メッセージ通知で返されます。 一方で、製品 ID のようなカスタマイズユーザーデータを使用して、業務システムに関連付けを行うこともできます。

1.2 トランスコードテンプレート

トランスコードで使用されるパラメーターはたくさんあり、トランスコード ジョブを繰り返し登録するのは手間がかかります。トランスコード テンプレートは、これを解決するために提案された概念です。トランスコード テンプレートは、よく使用されるパラメーターをまとめてあります。 トランスコードテンプレートには、プリセットテンプレートとカスタムテンプレートの 2 種類があります。

・ プリセットテンプレート

よく使用するパラメーターをあらかじめ組み合わせてあります。 詳しくは、「プリセットテンプレートの詳細」をご参照ください。

プリセットテンプレートにはいくつかのサブタイプがあります。

- プリセット静的テンプレート

  ビデオ トランスコード、オーディオ トランスコード、転送パッケージ、およびその他のシナリオにおいて、トランスコードテンプレートとしてそのまま使用することができます。 たとえば、"MP4-HD"や"MP3-128K" ながあります。
- 狭帯域 HD テンプレート

  狭帯域 HD は、メディアトランスコードに特化した技術です。 コストを変えるこなく、同一のビットレート下で、より鮮明で、より良いユーザー体験を提供します。
- プリセットスマートテンプレート

  プリセットスマートテンプレートは、入力ファイルの特性に応じてトランスコードパラメーターを自動的に調整するため、同じ解像度でもビットレートが低くなり、コストを削減することができます。

  > 注：
  > プリセットスマートテンプレートを使用する場合は、まず SubmitAnalysisJob インターフェイス (SubmitAnalysisJob) を呼び出す必要があります。 分析タスクが正常に完了したら、Query Template Analysis Job インターフェイス (QueryAnalysisJobList) を呼び出し、入力ファイルリストに対応する有効なプリセットスマートテンプレートを取得します。 無効なリスト内にあるプリセットスマートテンプ

レートがトランスコードタスクに指定された場合、そのトランスコードタスクは無効となり、失敗を返します。

- カスタムテンプレート

  要件がさらに高い場合は、トランスコード パラメーター (audio、video、container、transcode など) を組み合わせ、独自のカスタム テンプレートを定義します。 各カスタムテンプレートには一意の ID があります。

1.3 ワークフローとメディア

ここでは、開発者がメディア処理サービスをよりよく理解したうえで使用できるよう、MPS の基本概念と関連性について説明しています。

コンセプトの説明

- メディア

  メディアには、入力ビデオ / オーディオ メディアファイル、および、トランスコード / スクリーンショット / メディア情報 / AI タグなどの、関連するすべての出力ファイルが含まれます。 入力ファイルとメディアは、 1 対 1 の関係にあり、メディア ID によって一意に識別されます。

  メディアファイル

  メディアファイルは、すべてのメディアの集合であり、メディアはメディアファイルの最小単位です。

- ワークフロー

  ワークフローはメディアの作成を自動化する工場のようなもので、MediaWorkflowId によって一意に識別されます。

  注 :
  メディアワークフローは、ワークフローともいいます。

  - イベント

    ワークフロー内の各ノードは、アクティビティといいます。 アクティビティは、実際の要件に応じて、並行処理 (タスク A、B、Cなど)、もしくは直列化による処理 (タスクA1、A2など）で実行します。 最初の入力アクティビティと最後のレポート通知アクティビティに

加え、アクティビティは、トランスコードタスクやスクリーンショットタスクなどの各種タスクをサポートしています。

■ 入力アクティビティの開始

ワークフローに関連付けられたストレージのトリガーパスを設定し、ビデオ / オーディオマルチメディアファイルが対応するパスにアップロードされるたびにタスクの実行を自動的にトリガーします。

■ ポストレポートアクティビティの終了

ワークフローの実行が終了すると、実装メッセージが送信されます。 実行結果には、メディア ID の絶対アドレスとマルチメディアファイルが含まれているため、特定のマルチメディアファイルを実行することができます。

■ タスクアクティビティ

タスクでサポートされているすべてのパラメーターは、タスクアクティビティで設定することができます。

- マッチングルール

たとえば、アップロードされたファイルが `http :// bucket . oss - cn - hangzhou . aliyuncs . com / A / B / C / test1 . flv` の場合、設定されたトリガーパスの結果は次の通りです。

パス	マッチングするかどうか
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/	はい
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C2/	いいえ
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/	はい
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B2/	いいえ
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/	はい
http://bucket.oss-cn-hangzhou.aliyuncs.com/A2/B/C/	いいえ
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test	はい

パス	マッチングするかどうか
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test2	いいえ

- 拡張子マッチングルール

  自動トリガーシステムは、アップロード中のファイルの拡張子をチェックし、無効なデータ (pdf、word ファイル、その他のファイルなど) が生成されないようにします。

   注：
  API の手動トリガーシステムでは、拡張子はチェックされません。

  ファイルに拡張子が付いていないか (ファイルに拡張子区切り文字 "." が含まれていない)、もしくは次の拡張子のファイルである必要があります。

  ■ ビデオ

  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.

  ■ オーディオ

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

- ワークフローの実行

  マッチするマルチメディアファイルをアップロードするごとに、1 回実行されます。 同じマルチメディアファイルを複数回アップロードすれば、ワークフローは複数回実行されます。各実行には一意の RunId 識別子があります。

  アップロード時の自動トリガー機能に加え、ワークフローではストレージに格納されているマルチメディアファイルを対象として、API で実行させることもできます。 API を呼び出すたびに実行されます。

- ユーザーデータ

  実行するたびに、製品 ID などのカスタムユーザーデータパラメーターを指定することができます。 カスタムユーザーデータパラメーターは、メッセージ通知で返されるため、メディア ID またはマルチメディアファイルの絶対パスを業務システムに記録する必要がありません。 一方で、製品 ID などのカスタムユーザーデータを使用して、業務システムに関連付けを行うこともできます。

2 ワークフローの開発プロセス

1. ワークフローの設定

   快適な操作性: コンソールの GUI を使用することで、クラウドベースのオーディオ / ビデオ処理プロセスがオンデマンドで構築されます。

   強力な機能: スクリーンショット、トランスコード、狭帯域 HD 分析、カプセル化、ウォーターマーク、および編集機能がサポートされています。

   コンソール構成の詳細については、「ワークフロー」をご参照ください。
2. メディアファイルのアップロード

   メディアファイルがワークフローで指定された入力バケットとパスにアップロードされると、指定されたプロセスに基づき、ワークフローが自動的に実行されます。
3. メッセージ通知の待機

   ワークフロー実行中のメッセージ通知。 たとえば、実行開始通知、および完了通知が受信されます。
4. ビデオの再生

   ワークフローの実行後、プレーヤーでビデオを再生するためのトランスコードされた再生 URL が取得されます。

3 ビデオのアップロード

3.1 概要

アップロード

Web バージョン (JavaScript) およびモバイルバージョン (Android / iOS) のアップロード SDK を提供しています。

ビデオファイルのアップロードには、コンソールまたはサードパーティーのツールを使用します。

機能

ユーザーフレンドリーな API を提供しています。 必要な操作はローカルファイルと OSS ファイルの保存場所を指定するだけです。

再開可能なアップロード、マルチファイルキュー、超大容量ファイル、ネットワーク異常からの復旧、およびセキュリティメカニズムをサポートしています。

メディアワークフローは自動的に起動します。

メディアワークフローの起動

マルチメディアファイルがメディアワークフローで指定された入力バケットとパスにアップロードされると、指定されたプロセスに基づき、メディアワークフローが自動的に起動します。

OSS ファイルをアップロードしてメディアワークフローを自動的に起動する場合、次の条件を満たしている必要があります。

- メディアワークフローがマッチしている

  ワークフローの起動とマッチングルールの詳細については、「AddMedia」をご参照ください。 ワークフローは `Activated` ステータスになっている必要があります。

- ファイルの拡張子がマッチしている

  ワークフローの起動要件は、ファイルがマルチメディアファイルであることです、 Media Files サービスは、ファイルの拡張子を介して判断を行います。 ファイルに拡張子が含まれて

いない (ファイル名に拡張子の区切り文字 "."が含まれていない) か、下記の拡張子に該当している必要があります。

- ビデオ

  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.

- オーディオ

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

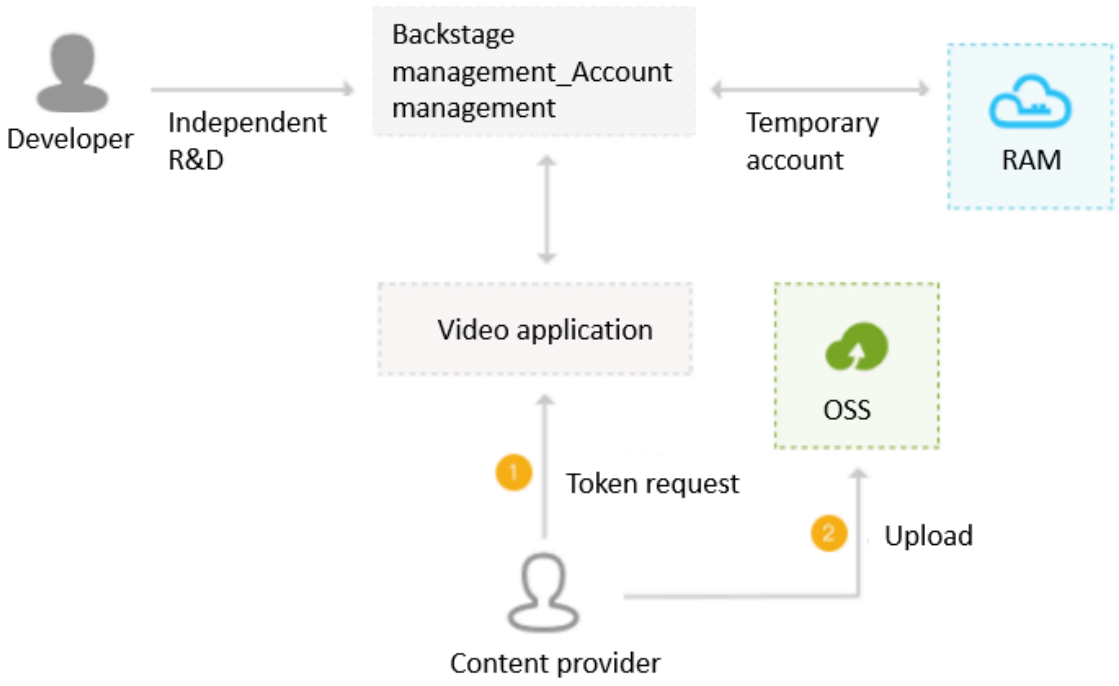
・ メディア属性を指定している

タイトル、タグ、説明、カテゴリ、カバー URL、およびカスタマイズデータといったメディア属性を指定して、メディアワークフローを起動します。 属性の詳細については、「Add Media」ページ内の「リクエスト パラメーター」をご参照ください。

セキュリティ

通常、ビデオファイルはクライアントを使用して直接アップロードされます。 この場合、AccessKey は安全にクライアントに格納されている必要があります。 いったん AccessKey が公開されると、AccessKey は高いリスクにさらされることになり、置き換えが困難になります。 クライアントからアプリケーションにアクセスして AccessKey を取得し、RAM により提供されるトークンを使用することを推奨します。

推奨されるプロセス



1. トークンのリクエスト

   ファイルがアップロードされるたびに、ビデオアプリケーション (アプリモードまたは Web モード) を使用して、ビジネス側のアプリケーションサービスにアクセスします。 アプリケーションサービスは RAM からトークンを取得し、それをビデオアプリケーションに送り返します。 これにより、セキュリティが確保され、検証および権限管理が実装され、アップロード履歴が記録されます。

   トークンを使用する前に、「サブアカウントおよび権限」を設定します。

   詳しくは、「Java サンプルコード」をご参照ください。 (他の言語でトークンを使用する方法の詳細については、「STS の概要」をご参照ください)。

2. ファイルのアップロード

   アップロード SDK をビデオアプリケーションに統合した後、取得したトークンを使用してファイルをアップロードします。 詳しくは、「使用方法」をご参照ください。

   - Web

     JavaScript ファイルはアプリケーションまたは CDN ドメインに保存され、ビデオファイルは OSS ドメインに保存されます。そのため、JavaScript ファイルがアップロードされる際、リージョン間のリクエストが発生します。 この場合、「CORS 設定」をご参照ください。

     JavaScript

   - モバイル

     Android

     iOS

3.2 サブアカウントおよび権限付与の設定

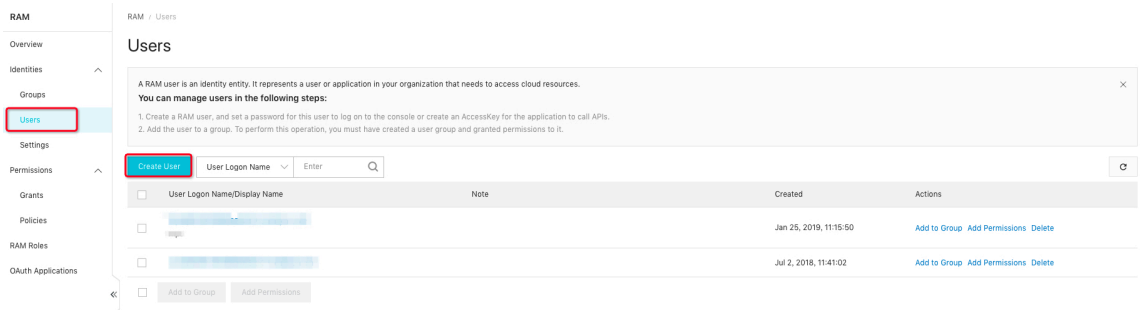
サブアカウントと権限付与の設定を行うには、次の手順を実行します。

1. サブアカウントの作成

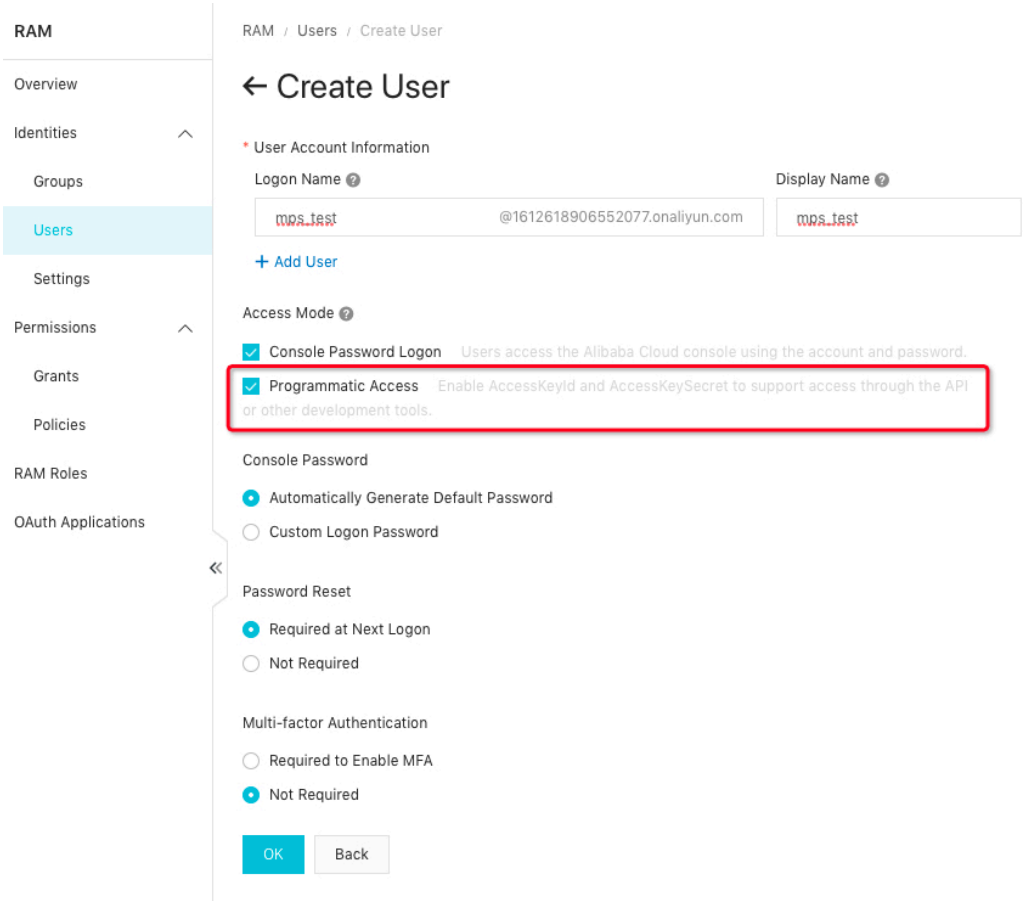
   a. RAM コンソールにログインします。

   b. 左側のナビゲーションウィンドウで、[ID] > [ユーザー] をクリックします。

   c. [ユーザーの作成] をクリックします。



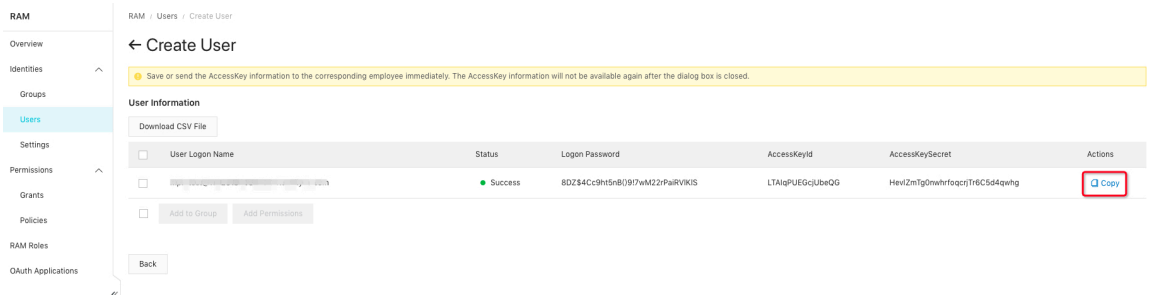
   d. [ユーザーの作成] で、MPS にアクセスするため、プライマリアカウントと同じ権限を持つサブアカウントを作成します。



 注：

プログラムによるアクセス のチェックボックスをオンにします。

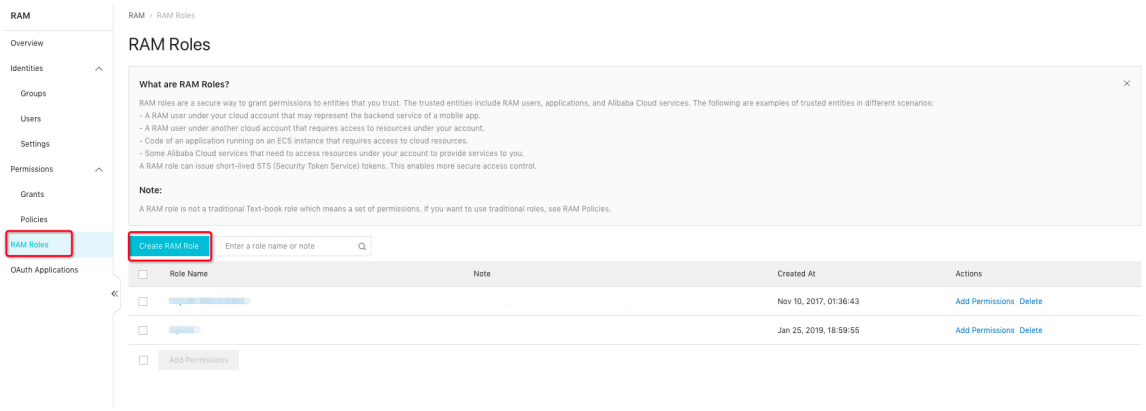
e. このアカウントの AccessKey を生成し、これ以降のアクセスのため、AccessKey をコピーして保存します。



2. ロールの作成

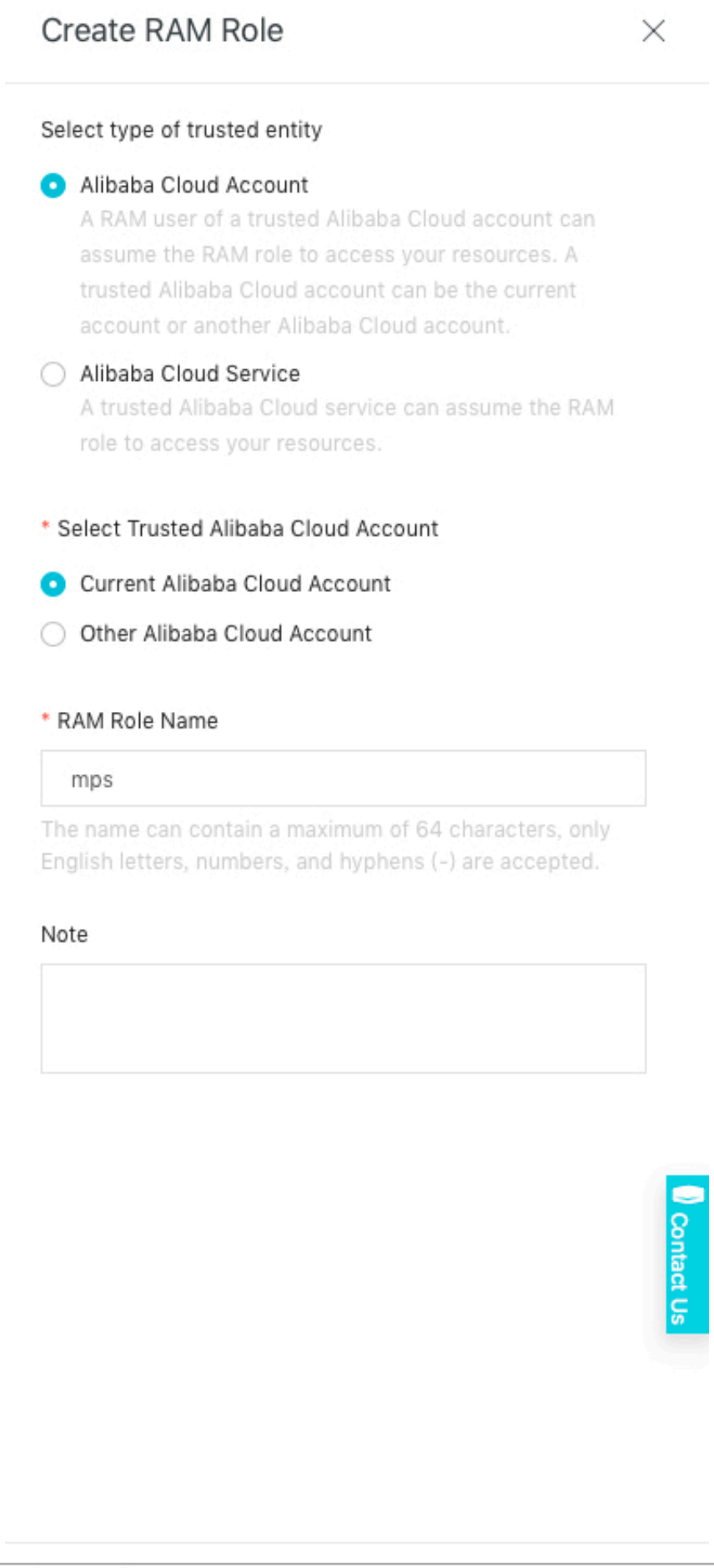
   a. 左側のナビゲーションウィンドウで、[RAM ロール] をクリックします。

   b. [RAM ロールの作成] をクリックします。

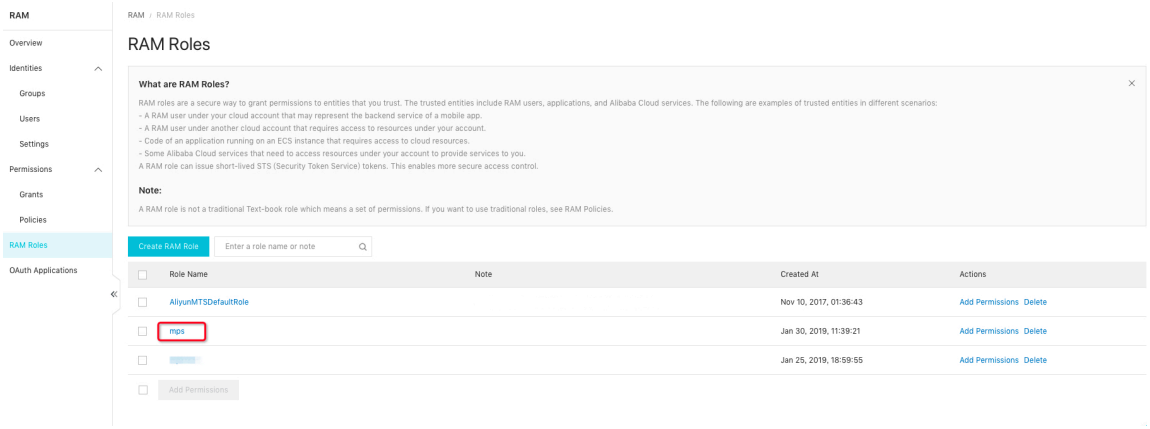


   c. [信頼されているエンティティタイプの選択] で、[Alibaba Cloud アカウント] をクリックします。

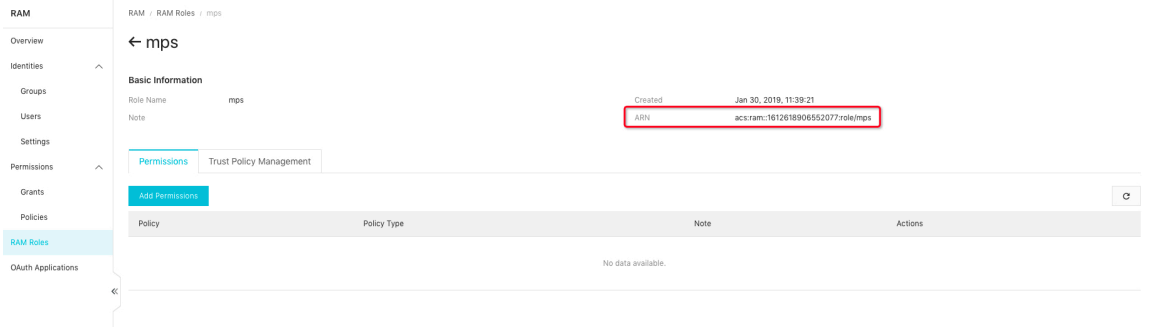
      [信頼されているエンティティタイプの選択] で、[現在の Alibaba Cloud アカウント] をクリックし、[OK]をクリックします。


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. [RAMロール] で、[作成したロール] をクリックします。



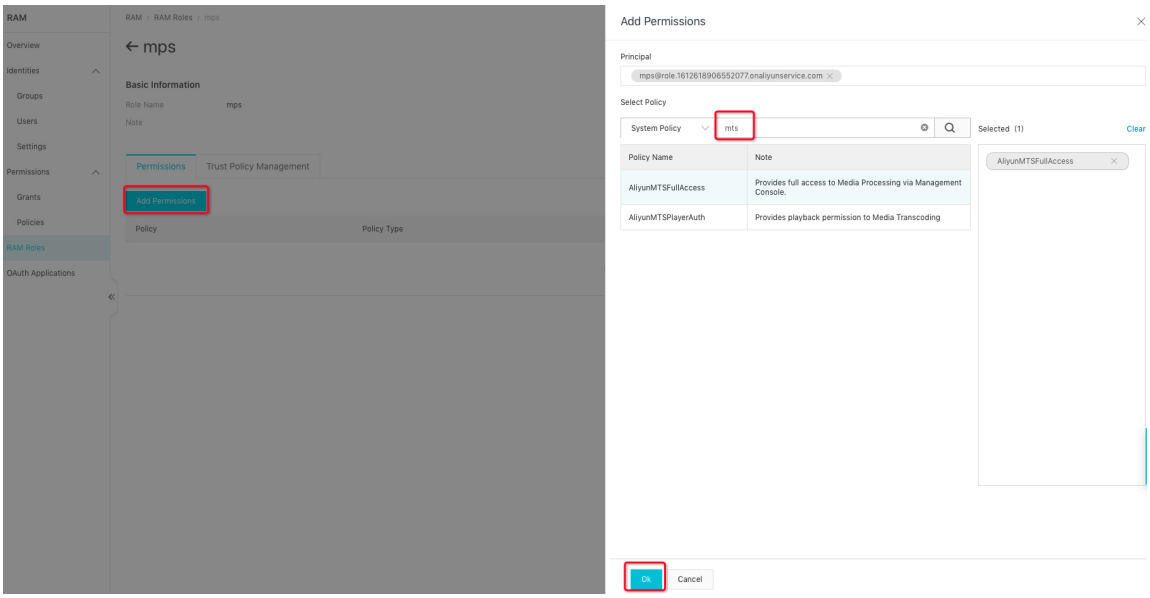
e. [基本情報]で、ARN パラメーター acs:ram::1612618906552077:role/mps をコピーします。



3. ロールに対する権限付与の設定

a. [作成したロール] のページで、[アクセス許可の追加] をクリックします。

b. ポリシーを選択します。



注：

サブアカウントの STS 権限を調整する (アクセス許可の変更、追加、削除など) には、このステップに戻ります。

[カスタマイズポリシー] でポリシーを作成します。作成したポリシーを [ポリシーの編集] に追加することで、アップロード SDK に必要な最小限の権限を付与します。 ポリシー全体の内容は次のとおりです。

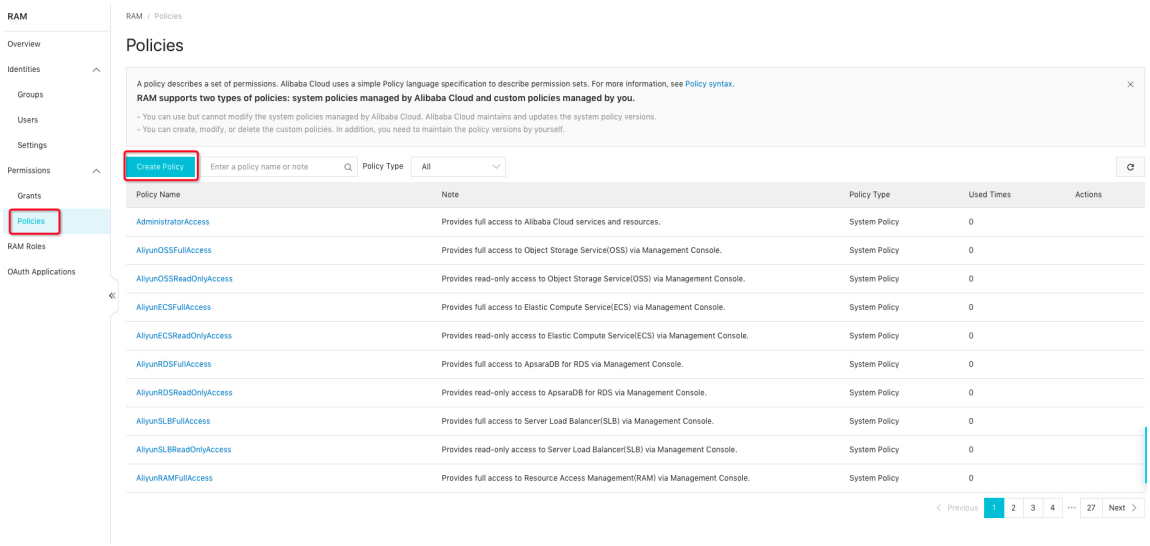
```
{
     " Statement ": [
       {
         " Action ": [
           " oss : PutObject ",
           " oss : AbortMulti  partUpload ",
           " oss : ListMultip  artUploads ",
           " oss : ListParts "
         ],
         " Effect ": " Allow ",
         " Resource ": [
           "*"
         ]
       }
     ],
     " Version ": " 1 "
```

```
    }
```

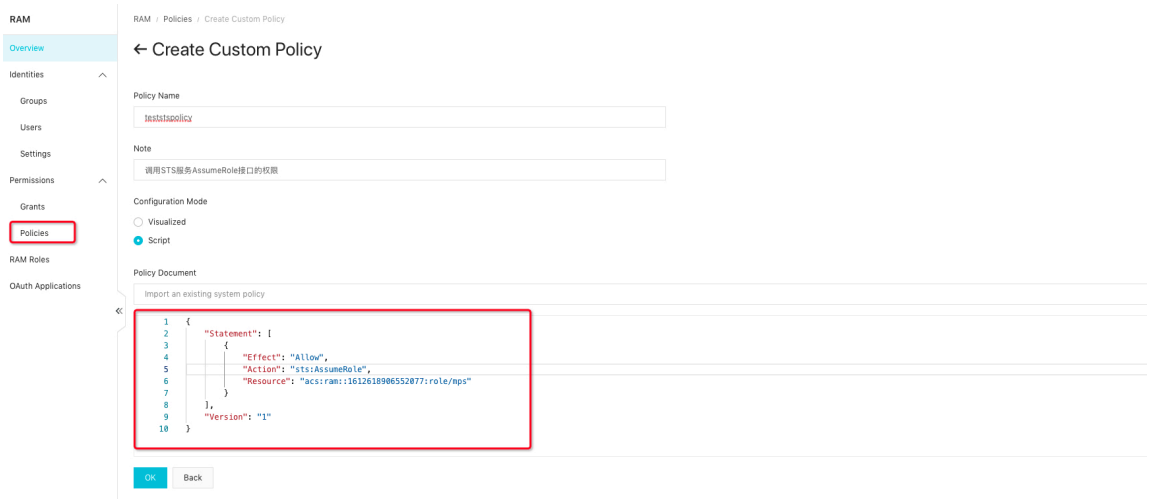
4. サブアカウントをロールに関連付けます。

   a. RAM コンソールにログインし、左側のナビゲーションウィンドウで、[権限] > [ポリシー] をクリックします。

   b. [ポリシーの作成] をクリックします。



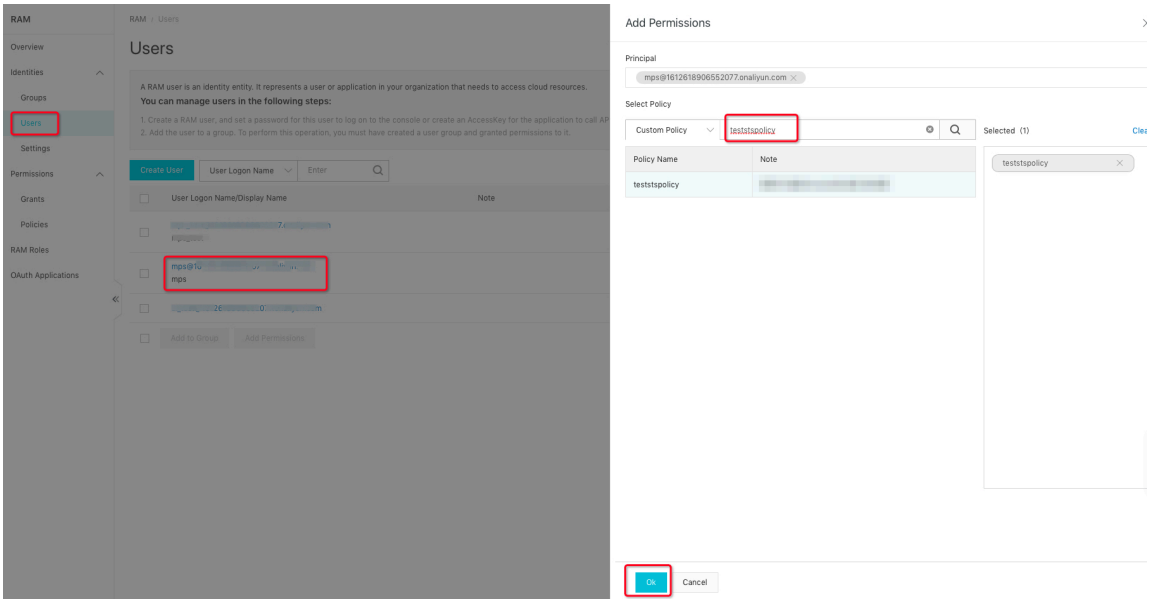
   c. [カスタマイズポリシーの作成] で、[リソース] フィールドを ARN パラメーターacs:ram::1612618906552077:role/mps に設定します。



   d. 左側のナビゲーションウィンドウで、[ID] > [ユーザー] をクリックします。

   e. 設定したサブアカウントを選択して、[権限の追加] をクリックします。

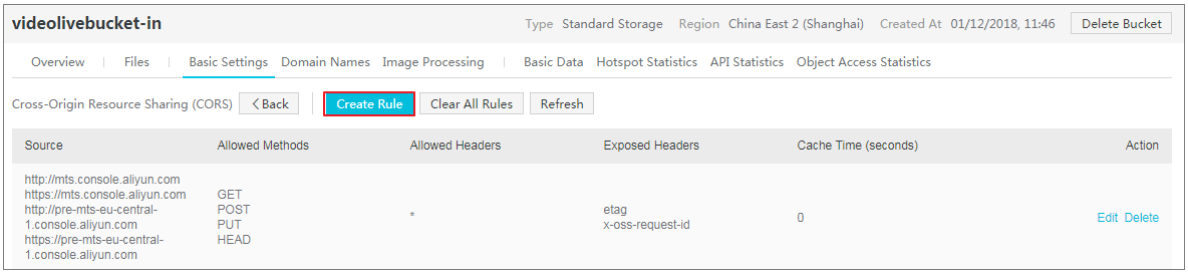
   f. 作成したテストポリシーを入力すると、teststspolicy が表示されます。



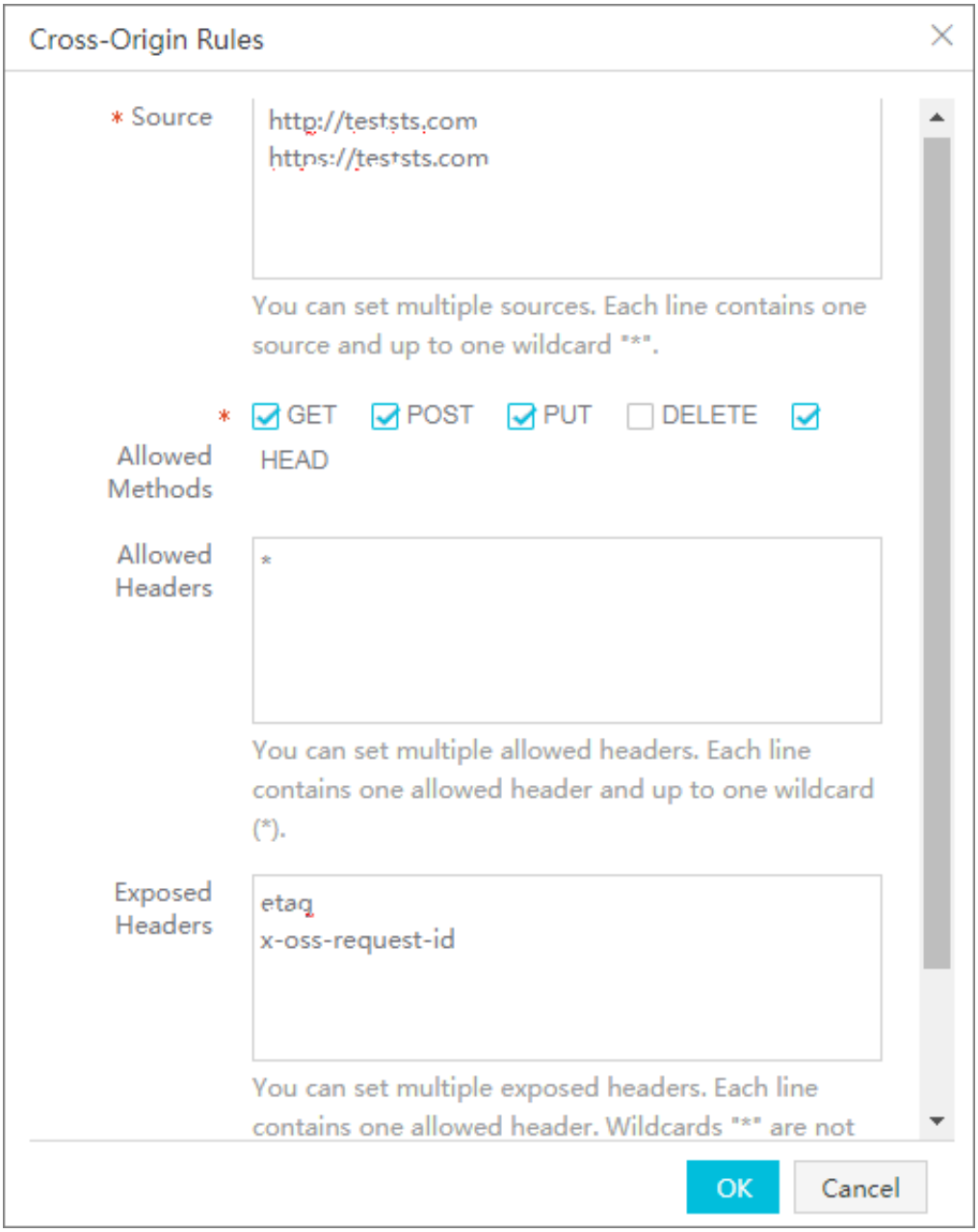
3.3 CORS 設定

JavaScript ファイルとビデオファイルは異なるリージョンに格納されるため、JavaScript ファイルがアップロードされるとリージョン間のリクエストが発生します。 この場合、リージョン間の設定は OSS 上で実装する必要があります。 OSS 上でリージョン間の設定をしないと、Web 側より JavaScript を使用してファイルをアップロードすることはできません。

1. 入力バケットの CORS 設定ページを開きます。



2. ルールを追加します。



- ソース

  JavaScript ファイルが展開されているリージョンの名前を入力します。 HTTP と HTTPS 両方でのアクセスが必要な場合は、それぞれ追加します。

- メソッド

  [GET]、[POST]、[PUT]、[HEAD] のチェックボックスをオンにします。

- 許可されたヘッダー

  `*` を入力します。

- 公開ヘッダー

「etag」 および 「x-oss-request-id」 を 2 行に入力します。

3.4 セキュリティトークンのリクエスト - Java のサンプルコード

1. pom.xml の STS SDK を参照します。

```
< repositori  es >
    < 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 >
 </ repositori  es >
< dependenci  es >
 < 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 >
</ dependenci  es >
```

2. コード

STS では、ロールパラメーター `roleArn` が必要です。RAM コンソール にログインし、[ロール]をクリックし、該当の[ロール名] をクリックします。 `Arn` パラメーターは、基本情報欄に表示されます。たとえば、 `1351140512  345678 : role / teststs` のように表示されます。

・ main 関数

```
public   static   void   main ( String []  args )  throws
Exception  {
   IClientPro  file   profile  =  DefaultPro  file . getProfile (
                                      " cn - hangzhou ",
                                      < accessKeyI  d >,
                                      < accessKeyS  ecret >);
   DefaultAcs  Client   client  =  new   DefaultAcs  Client (
profile );
   AssumeRole  Response   response  =  assumeRole ( client , <
roleArn >);
   AssumeRole  Response . Credential  s   credential  s  =
response . getCredent  ials ();
   System . out . println ( credential  s . getAccessK  eyId () +
"\ n " +
```

```
                        credential  s . getAccessK  eySecret () +
 "\ n " +
                        credential  s . getSecurit  yToken () + "\
 n " +
                        credential  s . getExpirat  ion ());
}
```

・ 一時的なアクセスキー、および一時的なトークンを生成する関数

```
 private   static   AssumeRole  Response   assumeRole (
                                    DefaultAcs  Client   client ,
                                    String   roleArn )
                                    throws   ClientExce  ption  {
    final   AssumeRole  Request   request  =  new   AssumeRole
 Request ();
    request . setVersion (" 2015 - 04 - 01 ");
    request . setMethod ( MethodType . POST );
    request . setProtoco  l ( ProtocolTy  pe . HTTPS );
    request . setDuratio  nSeconds ( 900L );
    request . setRoleArn ( roleArn );
    request . setRoleSes  sionName (" test - token ");
    return   client . getAcsResp  onse ( request );
}
```

3. トークンの有効期間

サンプルコードで生成されたトークンは 900 秒間有効ですが、必要に応じて変更することができます (900 秒間 から 3,600 秒間の範囲)。

何度もトークンを新しく生成することなく、有効期間内の生成されたトークンを使用することができます。 次の例は、トークンを再生成する必要があるかどうかを確認するためのコードです。

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

4 メッセージ通知の受信

4.1 概要

通知の形式

メディア ワークフローの実行が開始、あるいは完了した際、MNS で指定されたキューまたはトピック (通知) に対し、通知が送信されます。

・ 形式の定義

通知本文は、JSON 形式です。 フィールド名、タイプ、および説明の詳細は、「AddMedia」ページ内の [メディア ワークフロー通知] をご参照ください。

構造層は次のように定義されています。

- 最上位層

これは JSON 形式のオブジェクトです。 定義は下記の通りです。

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

- 現在のアクティビティの基本属性

これは独立したオブジェクトではなく、最上位層のキー値の属性です。 次の例をご参照ください。 定義は下記のとおりです。

ワークフロー実行 ID、アクティビティ名、アクティビティタイプ、アクティビティステータス、エラー情報。

- ワークフローで実行されるオブジェクトの詳細

これは JSON 形式のオブジェクトです。 定義は下記のとおりです。

{ワークフロー実行 ID, メディアワークフロー ID, メディアワークフロー名, メディア ID, 入力ファイル, ワークフロー実行タイプ, `activity object array` , 作成時間}.

- アクティビティオブジェクト配列

これはJSON 形式の配列です。現在のステータスに対して実行されたすべてのアクティビティを含みます。 たとえば、開始通知に含まれるのは、開始アクティビティオブジェクト

のみであり、一方で完了通知に含まれるのは、すべてのアクティビティオブジェクトになります。 定義は下記のとおりです。

[Activity object , activity object , …]

- アクティビティオブジェクト

  これは JSON 形式のオブジェクトです。 定義は下記のとおりです。

  {アクティビティ名, アクティビティタイプ, タスク ID, アクティビティステータス, 開始時間, 終了時間, エラー情報}

- 開始

  activity basic attribute の "Activity type" は “Start” です。

- 完了

  activity basic attribute の "Activity type" は "Report" です。

- 例

```
{
      " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2 ",
      " Name ": " Act - 4 ",
      " Type ": " Report ",
      " State ": " Success ",
      " MediaWorkf  lowExecuti  on ": {
        " Name ": " Concurrent  Success ",
        " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2 ",
        " Input ": {
            " InputFile ": {
                " Bucket ": " inputfirst ",
                " Location ": " oss - test ",
                " Object ": " mediaWorkf  low / Concurrent  Success
 / 01 . wmv "
            }
        },
        " State ": " Success ",
        " MediaId ": " 2be491ab4c  b6499cd0be  fe5fcf0cb6  70 ",
        " ActivityLi  st ": [
            {
                " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2
 ",
                " Name ": " Act - 1 ",
                " Type ": " Start ",
                " State ": " Success ",
                " StartTime ": " 2016 - 03 - 15T02 :  53 :  41Z ",
                " EndTime ": " 2016 - 03 - 15T02 :  53 :  41Z "
            },
            {
                " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2
 ",
                " Name ": " Act - 2 ",
                " Type ": " Transcode ",
                " JobId ": " f34b6d1429  dd491faa7a  6c1c8f9052  85
 ",
                " State ": " Success ",
                " StartTime ": " 2016 - 03 - 15T02 :  53 :  43Z ",
```

```
                " EndTime ": " 2016 - 03 - 15T02 :  53 :  47Z "
            },
            {
                " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2
",
                " Name ": " Act - 3 ",
                " Type ": " Snapshot ",
                " JobId ": " c14150be33  304825a5d6  7cd5364c35  cb
",
                " State ": " Success ",
                " StartTime ": " 2016 - 03 - 15T02 :  53 :  44Z ",
                " EndTime ": " 2016 - 03 - 15T02 :  53 :  45Z "
            },
            {
                " RunId ": " 8f8aba5a62  ab4127ae2a  dd18da20b0  f2
",
                " Name ": " Act - 4 ",
                " Type ": " Report ",
                " State ": " Success ",
                " StartTime ": " 2016 - 03 - 15T02 :  53 :  49Z ",
                " EndTime ": " 2016 - 03 - 15T02 :  53 :  49Z "
            }
        ],
        " CreationTi  me ": " 2016 - 03 - 15T02 :  53 :  39Z "
    }
}
```

通知の受信方法と処理方法

- キュー

  PHP サンプルコード

- トピック (通知)

  PHP サンプルコード

4.2 キューによるメッセージの受信

ここでは、MNS の要件とインストール手順について、簡単に説明します。

詳しくは、MNS のドキュメント「SDK のダウンロード」、および「キューユーザーマニュアル」をご参照ください。

この例で使用している言語は PHP です。 その他言語の使用方法の詳細については、MNS のドキュメント「SDK ユーザーマニュアル」をご参照ください。

環境要件

PHP 5.5 以降

インストール

MNS SDK for PHP を Alibaba Cloud からダウンロードします。

MNS SDK for PHP を Alibaba Cloud からダウンロードします。

この例で使用している言語は PHP です。 その他言語の使用方法の詳細については、MNS のドキュメント「SDK ユーザーマニュアル」をご参照ください。

ファイルをプロジェクトのディレクトリに展開します。 `php_sdk` ディレクトリが展開されます。

サンプルコード

- MNS SDK への参照設定

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

- MNS の初期化

MNS では、ユーザーの各リージョンごとに、独立したサービスドメイン名を設定します。 ルールは、 `https ://${ UserId }. mns .${ Region }. aliyuncs . com` です。 次の例では、中国 (杭州) (cn-hangzhou) を使用しています。 ユーザーの実際の要件に応じて、リージョンを中国 (北京) (cn-beijing) などに置き換えます。

```
use   AliyunMNS \ Client ;
 use   AliyunMNS \ Exception \ MnsExcepti  on ;
```

```
$ mns_client  =  new   Client (' https ://'.$ user_id .'. mns . cn -
 hangzhou . aliyuncs . com ',
                          $ access_key  _id , $ access_key  _secret
 );
 $ queue  = $ mns_client -> getQueueRe  f ($ queue_name );
```

- メッセージの受信

MNS が受信する各メッセージにはハンドルが割り当てられ、後でメッセージを操作する際に使用します (メッセージの削除など)。

また、MNS ではメッセージの一括受信をサポートしており、パフォーマンスを向上させることができます。 詳しくは、MNS のドキュメント 「BatchReceiveMessage」をご参照ください。

メッセージの受信時に、タイムアウト時間を指定することができます。 (次の例では、タイムアウト時間は 3 秒間に設定されています) キューにメッセージがない場合、タイムアウトが発生してエラーが返されます。

```
$ receipt_ha  ndle  =  NULL ;
 $ message  =  null ;
  try
 {
      $ res  = $ queue -> receiveMes  sage ( 3 );
       echo  " ReceiveMes  sage   Succeed ! \ n ";
      $ message  = $ res -> getMessage  Body ();
      $ receipt_ha  ndle  = $ res -> getreceipt  Handle ();
```

```
}
 catch  ( MnsExcepti  on  $ e )
{
      echo  " ReceiveMes  sage   Failed : " . $ e  . "\ n ";
}
```

- メッセージの削除

  メッセージは自動的にキューから削除されることはありません。 メッセージの削除は、DeleteMessage を呼び出して行います。 削除を行わない場合、メッセージはキューに残り続け、次回も同じメッセージを受け取ることになります。 また、DeleteMessage はメッセージを受信した後、指定された時間内に呼び出さないと失敗します。 詳しくは、「MNS - DeleteMessage」をご参照ください。

```
try
{
     $ res  = $ queue -> deleteMess  age ($ receipt_ha  ndle );
      echo  " DeleteMess  age   Succeed ! \ n ";
}
 catch  ( MnsExcepti  on  $ e )
{
      echo  " DeleteMess  age   Failed : " . $ e  . "\ n ";
}
```

- メッセージの分析

  メッセージ本文は文字列で、コンテンツは JSON 形式のオブジェクトです。 `json_decode` を使用して文字列をオブジェクトに変換した後、JSON オブジェクトを分析することで、メッセージの詳細を取得することができます。 次の例では、メディアワークフローを起動する出力ファイルが、出力されています。

```
$ json_messa  ge  =  json_decod  e ($ message );
 $ input_file  = $ json_messa  ge ->{' MediaWorkf  lowExecuti  on
'}->{' Input '}->{' InputFile '};
 echo  ' input_file  location :'.$ input_file ->{' Location '}.
     '  bucket :'.$ input_file ->{' Bucket '}.
     '  object :'.$ input_file ->{' Object '}."\ n ";
```

- ビデオ出力詳細の取得

  メッセージの詳細を取得した後、メディアライブラリ API を使用して、ワークフローで実行されたビデオの詳細を取得します。 次の例では、トランスコードタスク、およびスクリーンショットタスクの出力 URL が出力されています。

  メディアライブラリ内の PHP 用 SDK のインストールおよび設定方法についての詳細は、「メディアライブラリ SDK-PHP」をご参照ください。

```
include_on  ce  ' aliyun - php - sdk - core / Config . php ';
```

```
 use   Mts \ Request \ V20140618   as   Mts ;
```

メディアライブラリのクライアントを初期化します。

```
$ profile  =  DefaultPro  file :: getProfile (' cn - hangzhou ',
                                $ access_key  _id ,
                                $ access_key  _secret );
 $ mts_client  =  new   DefaultAcs  Client ($ profile );
```

すべてのトランスコードタスクの出力 URL と基本情報を出力します。

```
 If  ( strbmp  ($  json_messa  ge -> {' type '}, ' report  ') =  0
 ){
     $ activities  = $ json_messa  ge ->{' MediaWorkf  lowExecuti
 on '}->{' ActivityLi  st '};
     $ transcode_  job_ids  =  Array ();
      for  ($ i  =  0 ; $ i  <  count ($ audioStrea  ms ); $ i ++)
 {
        if  ( strcmp ($ activities [$ i ]->{' Type '}, ' Transcode
 ') ==  0 ) {
          $ transcode_  job_ids [] = $ activities [$ i ]->{' JobId
 '};
       }
     }
     $ request  =  new   Mts \ QueryJobLi  stRequest ();
     $ request -> setJobIds ( join (',', $ transcode_  job_ids ));
     $ request -> setRegionI  d (' cn - hangzhou ');
     $ response  = $ mts_client -> getAcsResp  onse ($ request );
      for  ($ i = 0 ; $ i  <  count ($ response ->{' JobList '}->{'
 Job '}); $ i ++) {
       $ output  = $ response ->{' JobList '}->{' Job '}[$ i ]->{'
 Output '};
       $ output_fil  e  = $ response ->{' JobList '}->{' Job '}[$ i
 ]->{' Output '}->{' OutputFile '};
       $ video_prop  erties  = $ response ->{' JobList '}->{' Job
 '}[$ i ]->{' Output '}->{' Properties '};
        echo  ' URLs   of   the   transcodin  g   output   files
 '.' http ://'.$ output_fil  e ->{' Bucket '}.'.'.
                      $ output_fil  e ->{' Location '}.'. aliyuncs
 . com /'.
                       urldecode ($ output_fil  e ->{' Object
 '})."\ n ";
        echo  ' basic   informatio  n   of   the   transcodin
 g   output   files '.$ video_prop  erties ->{' Width '}.' x '.$
 video_prop  erties ->{' Height '}.
                      '  duration :'.$ video_prop  erties ->{'
 Duration '}."\ n ";
     }
 }
```

すべてのスクリーンショットタスクの出力 URL を出力します。

```
 if  ( strcmp ($ json_messa  ge ->{' Type '}, ' Report ') ==  0 ) {
     $ activities  = $ json_messa  ge ->{' MediaWorkf  lowExecuti
 on '}->{' ActivityLi  st '};
     $ snapshot_j  ob_ids  =  Array ();
      for  ($ i  =  0 ; $ i  <  count ($ audioStrea  ms ); $ i ++)
 {
        if  ( strcmp ($ activities [$ i ]->{' Type '}, ' Snapshot
 ') ==  0 ) {
```

```
          $ snapshot_j  ob_ids [] = $ activities [$ i ]->{' JobId
'};
        }
      }
      $ request  =  new   Mts \ QuerySnaps  hotJobList  Request ();
      $ request -> setSnapsho  tJobIds ( join (',', $ snapshot_j
ob_ids ));
      $ request -> setRegionI  d (' cn - hangzhou ');
      $ response  = $ mts_client -> getAcsResp  onse ($ request );
       for  ($ i = 0 ; $ i  <  count ($ response ->{' SnapshotJo
bList '}->{' SnapshotJo  b '}); $ i ++) {
        $ snapshot_c  onfig  = $ response ->{' SnapshotJo  bList '}-
>{' SnapshotJo  b '}[$ i ]->{' SnapshotCo  nfig '};
        $ output_fil  e  = $ response ->{' SnapshotJo  bList '}->{'
SnapshotJo  b '}[$ i ]->{' SnapshotCo  nfig '}->{' OutputFile '};
         echo  ' URLs  of  the  screenshot  output  files '.'
http ://'.$ output_fil  e ->{' Bucket '}.'.'.
                        $ output_fil  e ->{' Location '}.'. aliyuncs
. com /'.
                         urldecode ($ output_fil  e ->{' Object
'})."\ n ";
      }
}
```

4.3 トピック通知によるメッセージの受信

MNS はメッセージ通知をトピックごとにユーザーに対して、積極的にプッシュします。 HTTP サービスが一般公開されている場合、ユーザーは簡単にメッセージを受信することができます。

基本構造

ビデオのメディアリポジトリ通知の基本的な構造は次のとおりです。

- 最上位層は MNS の構造本体です。

  MNS の定義と形式の詳細については、「MNS - 通知操作」をご参照ください。
- MNS の `message body` は、メディアリポジトリの構造体です。

  `message body` を受信後、MNS はさらにメディアリポジトリのメッセージを解析します。 解析手順とサンプルコードの詳細については、「キューによるメッセージの受信」をご参照ください。

セキュリティ

トピック (通知) モードは便利です。 しかしながら、HTTP サービスが一般公開されているため、不正な呼び出しや攻撃を防止する必要があります。 送信されるメッセージが MNS によるものかどうかを特定する方法の詳細については、MNS のドキュメントの「エンドポイント署名認証」をご参照ください。

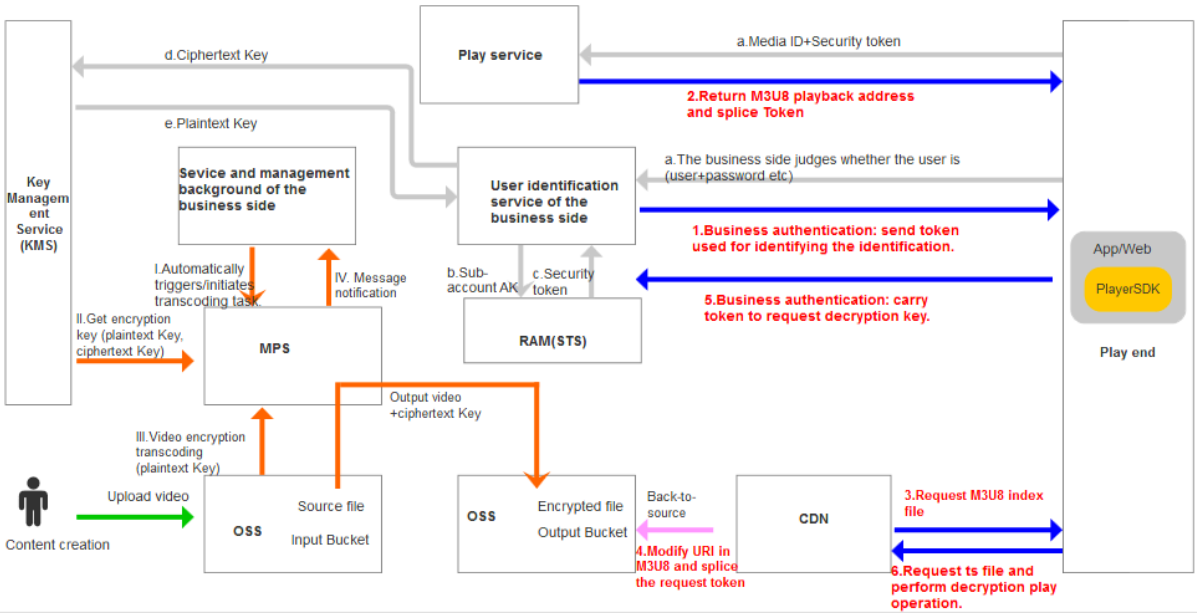
5 ビデオの暗号化

5.1 HLS 標準暗号化

ビデオの暗号化は、ビデオコンテンツを保護するための手段です。 ビデオコンテンツを暗号化することで、ビデオの漏洩やリーチングの問題を効果的に回避できるため、オンライン教育、金融、経済などの分野で広く使用されています。

> 注：
> Alibaba Cloud では現在 2 つの方法による暗号化をサポートしています。 1 つはプライベート暗号化、もう 1 つは HLS 標準暗号化です。 HLS 標準暗号化を使用した場合、ユーザーは暗号化キーを保護する必要があります。 ここでは、HLS 標準暗号化について説明します。

暗号化アーキテクチャの全容



用語

- Key Management Service (KMS)

  セキュリティ管理サービス。主にデータキーの生成、暗号化、および暗号化解除などの操作の役割を担います。 KMS サービスを有効にするには、こちらをクリックしてください。

- Data Key (DK) (平文キーとも呼ばれます)

  DK は、データの暗号化に使用される平文データキーです。

- Enveloped Data Key (EDK) (暗号文データキーとも呼ばれます)

  EDK は、エンベロープ暗号化テクノロジーを使用して暗号化された暗号文データキーです。
- Resource Access Management (RAM)

  Alibaba Cloud が提供する、ユーザー識別管理およびリソースアクセス管理サービスです。RAM サービスを有効にするには、こちらをクリックしてください。

手順

1. HLS 暗号化ワークフローを作成します。

   注:

   現在、コンソールは HLS 暗号化ワークフローの作成をサポートしていません。 API を使用して HLS 暗号化ワークフローを作成します。 デモについての詳細は、「HLS 標準暗号化ワークフローの作成」をご参照ください。 HLS 暗号化ワークフローの作成後、ワークフローはコンソール上で変更ができなくなるか、もしくは暗号化設定が無効になります。

   ワークフローでのキーの設定

   - 開始アクティビティノード: `InputFile :{" Bucket ":" bucketdemo ", " Location ":" oss - cn - hangzhou ", " ObjectPref ix ":" HLS - Encryption "}`

     このコードは、コンテンツ作成者がパス oss://bucketdemo/HLS-Encryption よりビデオを杭州にアップロードし、暗号化トランスコードが自動的に起動することを示します。
   - トランスコードアクティビティノード: `Encryption :{" Type ":" hls - aes - 128 ", " KeyUri ":" https :// decrypt . demo . com "}`

     トランスコード操作が完了すると、プレイヤーの使用のため、KeyUri の設定が "m3u8" ファイル上に現れます。

     再生中、プレイヤーは、再生のための DK 平文キーを取得するため、 EDK 暗号文キーを送信してアドレスを要求します。

2. ビデオのアップロード

   どちらの方法でビデオをアップロードしても、暗号化トランスコードは自動的に起動します。

   - MPS コンソールを使用し、作成したワークフローにビデオをアップロードします。
   - OSS アップロードツールを使用し、パス oss://bucketdemo/HLS-Encryption にビデオをアップロードします。

   トランスコードが完了すると、"m3u8" ファイルの内容が次のように表示されます。

```
# EXTM3U
     # EXT - X - VERSION : 3
     # EXT - X - TARGETDURA  TION : 5
     # EXT - X - MEDIA - SEQUENCE : 0
     # EXT - X - KEY : METHOD = AES - 128 , URI =" https :// decrypt
 . demo . com ? Ciphertext = aabbccddee  ff & MediaId = fbbf98691e
 a44b7c82dd  75c5bc8b92  71 "
     # EXTINF : 4 . 127544 ,
      1502961168  3170 - 00001 . ts
     # EXT - X - ENDLIST
```

3. プレイ

   - 再生アドレスを取得するには、QueryMediaList インターフェイスを使用します。 詳しくは、「QueryMediaList」 をご参照ください。 OSS アドレスを取得し、OSS ドメイン名を CDN ドメイン名に置き換え、復号化キーを要求するためのトークンとして機能するパラメーター `MtsHlsUriT  oken` を結合します。 その仕組みは以下のとおりです。

     再生中、プレイヤーは "m3u8" ファイルの EXT-X-KEY タグの URI にアクセスし、復号化キーを取得します。 URI は、ビジネス側で構築された復号化キーインターフェイスです。 したがって、復号化を要求するには、プレイヤーはビジネス側で承認された何らかの認証情報を送信する必要があります。 "MtsHlsUriToken" も同様の役割を果たします。 ビジネス側はプレイヤーにトークンを発行し、プレイヤーは復号化キーを要求する際にそのトークンを送信し、ビジネス側はトークンの有効性をチェックします。
   - プレイヤーは認証のため、トークンをビジネス側に送信します。

     たとえば、通常の再生アドレスは `https :// vod . demo . com / test . m3u8` です。 パラメーター `MtsHlsUriT  oken` を結合して送信すると、ビジネス側で発行される再生アドレスは、`https :// vod . demo . com / test . m3u8 ? MtsHlsUriT  oken = Token` になります。

     再生中、プレイヤーは Alibaba Cloudの CDN に対して、ビジネス側で発行された `https :// vod . demo . com / test . m3u8 ? MtsHlsUriT  oken = Token` を要求し、Alibaba Cloud の CDN は、"m3u8" ファイル内の復号化 URI を動的に変更します。 たとえば、元の `https :// decrypt . demo . com ? Ciphertext =`

`aabbccddee ff & MediaId = fbbf98691e a44b7c82dd 75c5bc8b92 71` は、ビジネス側で発行された `https :// decrypt . demo . com ? Ciphertext = aabbccddee ff & MediaId = fbbf98691e a44b7c82dd 75c5bc8b92 71 & MtsHlsUriT oken = Token` に変更されます。

したがって、プレイヤーが要求する最終的な復号化 URI は、`https :// decrypt . demo . com ? Ciphertext = aabbccddee ff & MediaId = fbbf98691e a44b7c82dd 75c5bc8b92 71 & MtsHlsUriT oken = Token` になります。 このアドレスは、ビジネス側で発行されたトークンを送信し、ビジネス側で識別されます。

4. ビジネス側では、以下の操作を行う必要があります。
   a. "MtsHlsUriToken" サービスの構築、発行、および識別。
   b. 復号化トークンの識別。 1 つのトークンは 1 回だけ使用できます。
   c. 復号化キー: 暗号文である EDK は、復号化のため、 KMS (Key Management Service) サービスの復号化インターフェイスを呼び出します。 詳しくは、「復号化」をご参照ください。 復号化後に情報をキャッシュして、ネットワーク IO を減らします。
   d. 復号化後に base64decodd を必要とする DK (平文キー)を取得し、それをプレイヤーに返します。

6 メディアライブラリの管理

6.1 概要

Java、PHP、および Python 用の MPS SDKを 使用して、メディアライブラリにアクセスします。

HTTP / HTTPS 経由でメディアライブラリにアクセスすることもできます。 詳しくは、「API Reference」をご参照ください。

機能

メディアワークフローの管理: メディアワークフローを追加、削除、変更、照会、有効化、および停止します。

メディアワークフロー実行インスタンスの管理: 実行インスタンスをトラバースし、照会します。

メディアの管理: メディアを追加、削除、変更、照会、および検索します。また、メディアセットの属性 (タイトル、タグ、カバー、説明) を管理し、メディアセットの公開ステータスを設定します。

メディア カテゴリの管理: メディア カテゴリを追加、削除、変更、および照会します。

サービスシナリオ

- メディアセットの検索

  検索条件を指定してメディアライブラリ内を検索します。

  キーワードでメディアセットを検索します。 部分一致検索でタイトル、タグ、説明、カテゴリの内1 つあるいは複数一致するメディアセットを表示します。 完全一致検索で、指定された属性 (タイトル、タグ、説明、カテゴリ) すべてが一致するメディアセットを表示します。

  検索条件には、作成時間の範囲を指定して検索対象を絞ることができます。 また、昇順または降順を指定して、検索結果を並べ替えることもできます。

  また、検索結果が多い場合は、複数ページに分けて表示されます。
- メディアセット属性の管理

  各メディアセットには、タイトル、タグ、説明、カテゴリの基本属性が含まれています。これらは API を使用して設定します。

  基本属性 - サンプルコード - PHP

- メディアセットタグの管理

  各タグはメディアセット内で一意です。 メディアライブラリでは、グローバルタグを設定することはできません。 ただし、メディアセット検索用の API を使用し、同じタグを有するすべてのメディアセットを照会することは可能です。

  タグの管理 - サンプルコード - PHP

メディアセットカテゴリの管理

メディアライブラリでは、グローバルなカテゴリ管理を行えます。 各メディアセットをカテゴリに関連付けることで、すばやくメディアセットを取得することができます。

カテゴリ管理 - サンプルコード - PHP

メディアセット詳細の照会

メディアセットには、入力ファイルおよび複数の出力ファイル (ビデオやスクリーンショット) があります。 入力および出力の詳細情報を指定することで、メディアセットを照会することができます。

入力情報には、ビデオの基本属性 (幅、高さ、継続時間、サイズ、ビットレート、フレームレート)、および詳細 (コンテナーのカプセル化、ビデオ、オーディオ、字幕ストリーム、カプセル化とストリームの詳細属性) があります。

出力情報には、基本属性 (幅、高さ、継続時間、サイズ、ビットレート、フレームレート)、ビデオの OSS URL、タイプ (シングルフレームまたはバッチ)、およびスクリーンショットの OSS URL があります。

メディアセットの詳細 - サンプルコード -PHP

6.2 ビデオの基本属性

概要

メディアセットの基本情報を照会および更新する方法については、次の例のとおりです。 SDK のインストール方法および使用方法の詳細については、「メディアライブラリ SDK-PHP」をご参照ください。

メディアセット基本情報の照会

メディア ID または OSS ファイルの URL を使用し、メディアセットを照会します。

- メディア ID を使用したメディアセットの照会

  パラメーターの詳細については、「API reference > Media APIs > QueryMediaList」ご参照ください。

```
 include_on  ce  ' aliyun - php - sdk - core / Config . php ';
 use   Mts \ Request \ V20140618   as   Mts ;
$ accessKeyI  D  = ' test '; // eplace   the   value   with   your
 AccessKeyI  D
$ accessKeyS  ecret  = ' test '; // Replace   the   value   with
 your   AccessKeyS  ecret
$ profile  =  DefaultPro  file :: getProfile (' cn - hangzhou ',
                                    $ accessKeyI  D ,
                                    $ accessKeyS  ecret );
$ client  =  new   DefaultAcs  Client ($ profile );
```

```
 function   queryMedia  ById ($ client , $ mediaID )
{
     $ request  =  new   Mts \ QueryMedia  ListReques  t ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setMediaId  s ($ mediaID );
     $ response  = $ client -> getAcsResp  onse ($ request );
      return  $ response ;
}
 function   printMedia ($ media )
{
      if  ( array_key_  exists (' Title ', $ media )) {
        print_r (' Title : '.$ media ->{' Title '}."\ n ");
     }
      if  ( array_key_  exists (' Descriptio  n ', $ media )) {
        print_r (' Descriptio  n : '.$ media ->{' Descriptio  n
 '}."\ 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  =  queryMedia  ById ($ client , $ mediaID )->{' MediaList
 '}->{' Media '};
 for  ($ i = 0 ; $ i  <  count ($ medias ); $ i ++) {
      printMedia ($ medias [$ i ]);
```

```
}
```

- OSS ファイルの URL を使用したメディアセットの照会

  パラメーターの詳細については、「API reference > Media APIs > QueryMediaList」をご参照ください。

```
 function   queryMedia  ByURL ($ client , $ mediaURL )
{
     $ request  =  new   Mts \ QueryMedia  ListByURLR  equest ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setFileURL  s ($ mediaURL );
     $ response  = $ client -> getAcsResp  onse ($ request );
      return  $ response ;
}
$ ossEndpoin  t  = ' 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  =  queryMedia  ByURL ($ client ,$ ossEndpoin  t .
 urlencode ($ ossObject ))->{' MediaList '}->{' Media '};
 for  ($ i = 0 ; $ i  <  count ($ medias ); $ i ++) {
     printMedia ($ medias [$ i ]);
}
```

- 属性の更新

  全属性の更新、または 1 つの属性を更新することができます。

  - 全属性の更新

    パラメーターの詳細については、「API reference > Media APIs > UpdateMedia」をご参照ください。

    属性を更新する際、すべてのフィールドを指定します。指定しないフィールドは消去されます。

```
 function   updateMedi  aAllField ($ client , $ mediaID , $ title
 , $ descriptio  n , $ tags , $ coverURL )
{
     $ request  =  new   Mts \ UpdateMedi  aRequest ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setMediaId ($ mediaID );
     $ request -> setTitle ($ title );
     $ request -> setCateId ( 2663987 );
     $ request -> setDescrip  tion ($ descriptio  n );
     $ request -> setTags ($ tags );
     $ request -> setCoverUR  L ($ coverURL );
     $ response  = $ client -> getAcsResp  onse ($ request );
      return  $ response ;
}
$ mediaID  = ' test '; // Replace   the   value   with   your
 desired   media   ID
$ media  =  updateMedi  aAllField ($ client , $ mediaID ,
```

```
                          ' title ', ' descriptio  n ', ' tags ', '
 coverURL ')->{' Media '};
```

- 1 つの属性の更新

  各種 API を使用することで、他のフィールドを変更せずに 1 つのフィールドだけを簡単に更新することができます。

  次の例では、"updateMediaPublishState" を使用します。 パラメーターの詳細については、 [API reference] > [Media APIs] > [UpdateMediaPublishState] をご参照ください。

```
 function   updateMedi  aPublishSt  ate ($ client , $ mediaID , $
 state )
{
     $ request  =  new   Mts \ UpdateMedi  aPublishSt  ateRequest
 ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setMediaId ($ mediaID );
     $ request -> setPublish ($ state );
     $ response  = $ client -> getAcsResp  onse ($ 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  {
     updateMedi  aPublishSt  ate ($ client , $ mediaID , " true
 ");
}  catch ( ClientExce  ption  $ e ) {
     print_r (' ClientExce  ption :"\ n ");
     print_r ($ e );
}  catch  ( ServerExce  ption  $ e ) {
     print_r (' ServerExce  ption :"\ n ");
     print_r ($ e );
}
```

6.3 メディアの詳細

概要

SDK のインストール方法および使用方法の詳細については、「メディアライブラリ SDK-PHP 」をご参照ください。

メディアセットには、入力ファイルと複数の出力ファイルが含まれています。 入力ファイルには、基本情報の他、メディアセットの詳細情報が含まれています。 ユーザーは、出力ファイルのビデオや、スクリーンショットの詳細情報を照会することができます。

入力

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

```
 $ accessKeyI  D  = ' test '; // eplace   the   value   with   your
 AccessKeyI  D
 $ accessKeyS  ecret  = ' test '; // Replace   the   value   with
 your   AccessKeyS  ecret
 $ profile  =  DefaultPro  file :: getProfile (' cn - hangzhou ',
                                      $ accessKeyI  D ,
                                      $ accessKeyS  ecret );
 $ client  =  new   DefaultAcs  Client ($ profile );
```

```
 function   queryMedia ($ client , $ mediaID )
{
    $ request  =  new   Mts \ QueryMedia  ListReques  t ();
    $ request -> setAcceptF  ormat (' JSON ');
    $ request -> setMediaId  s ($ mediaID );
    $ request -> setInclude  MediaInfo (" true ");
    $ response  = $ client -> getAcsResp  onse ($ request );
     return  $ response ;
}
 function   printMedia  Info ($ mediaInfo )
{
     print_r (' Number   of   Streams : '.$ mediaInfo ->{' Format '}-
>{' NumStreams '}."\ n ");
     if  ( array_key_  exists (' Streams ', $ mediaInfo ) &&
         array_key_  exists (' AudioStrea  mList ', $ mediaInfo ->{'
 Streams '}) &&
         array_key_  exists (' AudioStrea  m ', $ mediaInfo ->{'
 Streams '}->{' AudioStrea  mList '})) {
        $ audioStrea  ms  = $ mediaInfo ->{' Streams '}->{'
 AudioStrea  mList '}->{' AudioStrea  m '};
         print_r (' Audio   Streams :'."\ n ");
         for  ($ i  =  0 ; $ i  <  count ($ audioStrea  ms ); $ i ++)
 {
             print_r ("\ t [".$ i ."]"."\ n ");
             print_r ("\ t \ tCodecName : ".$ audioStrea  ms [$ i ]-
>{' CodecName '}."\ n ");
             print_r ("\ t \ tChannels : ".$ audioStrea  ms [$ i ]->{'
 Channels '}."\ n ");
             print_r ("\ t \ tSamplerat  e : ".$ audioStrea  ms [$ i
 ]->{' Samplerate '}."\ n ");
             print_r ("\ t \ tDuration : ".$ audioStrea  ms [$ i ]->{'
 Duration '}."\ n ");
             print_r ("\ t \ tBitrate : ".$ audioStrea  ms [$ i ]->{'
 Bitrate '}."\ n ");
         }
    }
     if  ( array_key_  exists (' Streams ', $ mediaInfo ) &&
         array_key_  exists (' VideoStrea  mList ', $ mediaInfo ->{'
 Streams '}) &&
         array_key_  exists (' VideoStrea  m ', $ mediaInfo ->{'
 Streams '}->{' VideoStrea  mList '})) {
        $ videoStrea  ms  = $ mediaInfo ->{' Streams '}->{'
 VideoStrea  mList '}->{' VideoStrea  m '};
         print_r (' Video   Streams :'."\ n ");
         for  ($ i  =  0 ; $ i  <  count ($ videoStrea  ms ); $ i ++)
 {
             print_r ("\ t [".$ i ."]"."\ n ");
             print_r ("\ t \ tCodecName : ".$ videoStrea  ms [$ i ]-
>{' CodecName '}."\ n ");
             print_r ("\ t \ tProfile : ".$ videoStrea  ms [$ i ]->{'
 Profile '}."\ n ");
             print_r ("\ t \ tDuration : ".$ videoStrea  ms [$ i ]->{'
 Duration '}."\ n ");
```

```
            print_r ("\ t \ tPixFmt : ".$ videoStrea  ms [$ i ]->{'
 PixFmt '}."\ n ");
            print_r ("\ t \ tFps : ".$ videoStrea  ms [$ i ]->{' Fps
 '}."\ n ");
            print_r ("\ t \ tBitrate : ".$ videoStrea  ms [$ i ]->{'
 Bitrate '}."\ n ");
            print_r ("\ t \ tResolutio  n : ".$ videoStrea  ms [$ i
 ]->{' Width '}.' x '.$ videoStrea  ms [$ 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 ++) {
     printMedia  Info ($ medias [$ i ]->{' MediaInfo '});
}
```

出力

- ビデオ

```
 function   queryMedia ($ client , $ mediaID )
{
    $ request  =  new   Mts \ QueryMedia  ListReques  t ();
    $ request -> setAcceptF  ormat (' JSON ');
    $ request -> setMediaId  s ($ mediaID );
    $ request -> setInclude  PlayList (" true ");
    $ response  = $ client -> getAcsResp  onse ($ request );
     return  $ response ;
}
 function   printOutpu  tVideos ($ 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 \ tMediaWork  flowName : ".$ videos [$ i ]-
>{' MediaWorkf  lowName '}."\ n ");
         print_r ("\ t \ tActivityN  ame : ".$ videos [$ i ]->{'
 ActivityNa  me '}."\ 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 \ tResolutio  n : ".$ 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 ++) {
     printOutpu  tVideos ($ medias [$ i ]->{' PlayList '}->{' Play
 '});
}
```

・ スクリーンショット

```
 function   queryMedia ($ client , $ mediaID )
{
    $ request  =  new   Mts \ QueryMedia  ListReques  t ();
    $ request -> setAcceptF  ormat (' JSON ');
    $ request -> setMediaId  s ($ mediaID );
    $ request -> setInclude  SnapshotLi  st (" true ");
    $ response  = $ client -> getAcsResp  onse ($ request );
     return  $ response ;
}
 function   printOutpu  tSnapshots ($ 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 \ tMediaWork  flowName : ".$ snapshots [$ i
 ]->{' MediaWorkf  lowName '}."\ n ");
         print_r ("\ t \ tActivityN  ame : ".$ snapshots [$ i ]->{'
 ActivityNa  me '}."\ 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 ++) {
     printOutpu  tSnapshots ($ medias [$ i ]->{' SnapshotLi  st '}-
>{' Snapshot '});
}
```

6.4 タグの管理

概要

SDK のインストール方法および使用方法の詳細については、「メディアライブラリ SDK-PHP」をご参照ください。

メディアリポジトリでは、グローバルタグの管理と設定をサポートしていません。 各メディアセット内のタグは一意です。 ユーザーは、同じタグを持つすべてのメディアセットを照会するメディアセット API の検索を行えます。

タグ関連の API は、1 つのタグの追加と削除をサポートしています。 一度に複数のタグを設定するには、UpdateMedia を使用します。

タグの追加

パラメーターの詳細については、「API reference > Media APIs > AddMediaTag」をご参照ください。

```
include_on  ce  ' aliyun - php - sdk - core / Config . php ';
  use   Mts \ Request \ V20140618   as   Mts ;
 $ accessKeyI  D  = ' test '; // eplace   the   value   with   your
AccessKeyI  D
 $ accessKeyS  ecret  = ' test '; // Replace   the   value   with
your   AccessKeyS  ecret
 $ profile  =  DefaultPro  file :: getProfile (' cn - hangzhou ',
                                    $ accessKeyI  D ,
                                    $ accessKeyS  ecret );
 $ client  =  new   DefaultAcs  Client ($ profile );
```

```
function   addMediaTa  g ($ client , $ mediaID , $ tag )
 {
     $ request  =  new   Mts \ AddMediaTa  gRequest ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setMediaId ($ mediaID );
     $ request -> setTag ($ tag );
     $ response  = $ client -> getAcsResp  onse ($ 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  {
      addMediaTa  g ($ client , $ mediaID , " testtag ");
 }  catch ( ClientExce  ption  $ e ) {
      print_r (' ClientExce  ption :"\ n ");
      print_r ($ e );
 }  catch  ( ServerExce  ption  $ e ) {
      print_r (' ServerExce  ption :"\ n ");
      print_r ($ e );
 }
```

タグの削除

パラメーターの詳細については、「API reference > Media APIs > DeleteMediaTag」をご参照ください。

```
function   deleteMedi  aTag ($ client , $ mediaID , $ tag )
 {
     $ request  =  new   Mts \ DeleteMedi  aTagReques  t ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setMediaId ($ mediaID );
     $ request -> setTag ($ tag );
     $ response  = $ client -> getAcsResp  onse ($ 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  {
      deleteMedi  aTag ($ client , $ mediaID , " testtag ");
```

```
} catch ( ClientExce ption $ e ) {
    print_r (' ClientExce ption :'."\ n ");
    print_r ($ e );
} catch ( ServerExce ption $ e ) {
    print_r (' ServerExce ption :'."\ n ");
    print_r ($ e );
}
```

6.5 カテゴリの管理

概要

SDK のインストール方法および使用方法の詳細については、「メディアライブラリ SDK-PHP」をご参照ください。

カテゴリーを追加、削除、変更、および照会することができます。 なお、次の点にご注意ください。

- カテゴリを削除しても、関連付けられているメディアセットのカテゴリ ID は自動的には消去されません。
- カテゴリ照会 API からの応答結果は、ツリー構造またはリスト構造で表示されます。 ネストされている JSON 形式のオブジェクトはツリー構造にして返され、プレーンな配列はリスト構造にして返されます。 実際のシナリオに応じて構造を選択します。

カテゴリの追加

パラメーターの詳細については、「API reference > Media category APIs > AddCategory」をご参照ください。

```
include_on ce ' aliyun - php - sdk - core / Config . php ';
 use Mts \ Request \ V20140618 as Mts ;
$ accessKeyI D = ' test '; // eplace the value with your
AccessKeyI D
$ accessKeyS ecret = ' test '; // Replace the value with
your AccessKeyS ecret
$ profile = DefaultPro file :: getProfile (' cn - hangzhou ',
                                    $ accessKeyI D ,
                                    $ accessKeyS ecret );
$ client = new DefaultAcs Client ($ profile );
```

```
function addCategor y ($ client , $ parentId , $ categoryNa me )
{
    $ request = new Mts \ AddCategor yRequest ();
    $ request -> setAcceptF ormat (' JSON ');
    $ request -> setParentI d ($ parentId );
    $ request -> setCateNam e ($ categoryNa me );
    $ response = $ client -> getAcsResp onse ($ request );
     return $ response ;
}
$ category = addCategor y ($ client , null , ' testroot ')->{'
Category '};
 print_r (' Level : '.$ category ->{' Level '}.
       "\ tParentId : ".$ category ->{' ParentId '}.
```

```
            "\ tCateId : ".$ category ->{' CateId '}.
            "\ tCateName : ".$ category ->{' CateName '}."\ n ");
```

カテゴリの更新

パラメーターの詳細については、「API reference > Media category APIs > UpdateCategoryName」をご参照ください。

```
function   updateCate  gory ($ client , $ categoryId , $ categoryNa
me )
 {
     $ request  =  new   Mts \ UpdateCate  goryNameRe  quest ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setCateId ($ categoryId );
     $ request -> setCateNam  e ($ categoryNa  me );
     $ response  = $ client -> getAcsResp  onse ($ request );
      return  $ response ;
 }
  try  {
      updateCate  gory ($ client ,  12345678 , ' updatetest  root
'); // Replace   the   value   with   your   category   ID
 }  catch  ( ClientExce  ption  $ e ) {
      print_r (' ClientExce  ption :'."\ n ");
      print_r ($ e );
 }  catch  ( ServerExce  ption  $ e ) {
      print_r (' ServerExce  ption :'."\ n ");
      print_r ($ e );
 }
```

カテゴリの削除

パラメーターの詳細については、「API reference > Media category APIs > DeleteCategory」をご参照ください。

```
function   deleteCate  gory ($ client , $ categoryId )
 {
     $ request  =  new   Mts \ DeleteCate  goryReques  t ();
     $ request -> setAcceptF  ormat (' JSON ');
     $ request -> setCateId ($ categoryId );
     $ response  = $ client -> getAcsResp  onse ($ request );
      return  $ response ;
 }
  try  {
      deleteCate  gory ($ client ,  12345678 ); // Replace   the
value   with   your   category   ID
 }  catch ( ClientExce  ption  $ e ) {
      print_r (' ClientExce  ption :'."\ n ");
      print_r ($ e );
 }  catch  ( ServerExce  ption  $ e ) {
      print_r (' ServerExce  ption :'."\ n ");
      print_r ($ e );
```

```
}
```

カテゴリの照会

- ツリー構造

  パラメーターの詳細については、「API reference > Media category APIs > CategoryTree」をご参照ください。

```
function   queryCateg  oryTree ($ client )
   {
       $ request  =  new   Mts \ CategoryTr  eeRequest ();
       $ request -> setAcceptF  ormat (' JSON ');
       $ response  = $ client -> getAcsResp  onse ($ request );
        return  $ response ;
   }
    function   printCateg  oryTree ($ categoryTr  ee )
   {
        foreach ($ categoryTr  ee   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 (' SubcateLis  t ', $ category
)) {
                printCateg  oryTree ($ category ->{' SubcateLis  t
'});
           }
       }
   }
   $ categoryTr  ee  =  queryCateg  oryTree ($ client )->{'
CategoryTr  ee '};
    printCateg  oryTree ( json_decod  e ($ categoryTr  ee ));
```

- リスト構造

  パラメーターの詳細については、「API reference > Media category APIs > ListAllCategory」をご参照ください。

```
 function   queryCateg  oryList ($ client )
     {
       $ request  =  new   Mts \ ListAllCat  egoryReque  st ();
       $ request -> setAcceptF  ormat (' JSON ');
       $ response  = $ client -> getAcsResp  onse ($ request );
        return  $ response   ;
   }
   $ categoryLi  st  =  queryCateg  oryList ($ client )->{'
CategoryLi  st '}->{' Category '};
    for  ($ i  =  0 ; $ i  <  count ($ categoryLi  st ); $ i ++) {
        print_r (' Level : '.$ categoryLi  st [$ i ]->{' Level '}.
               "\ tParentId : ".$ categoryLi  st [$ i ]->{'
ParentId '}.
               "\ tCateId : ".$ categoryLi  st [$ i ]->{' CateId
'}.
```

```
                "\ tCateName : ".$ categoryLi  st [$ i ]->{'
CateName  ' }."\ n ");
   }
```