阿里云
媒体处理

开发指南

文档版本：20190318

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 概念介绍

1.1 作业和管道

本文介绍媒体处理服务的基本概念和关系，以便您更好的理解和使用媒体处理服务。

概念解释：

- 作业

  作业在媒体处理服务里面是一个抽象的概念，包含多种类型的作业：转码作业、截图作业、媒体信息作业等。

  在一个作业中，包含3个关键信息：输入、输出和参数。输入和输出设定作业执行时的输入文件以及执行后的输出文件，参数则用来设置执行具体功能的详细配置。

- 参数
  - 模板参数

    由于作业的参数很多，每次提交作业时都重复填写比较麻烦，模板是为解决此问题而提出的概念。模板是一些参数的合集，把一些常用的参数组合在一起，可以减少提交作业的参数数量，简化提交作业的代码。

  - API参数

    如果给每种不同的参数组合都创建模板，会导致模板的数量剧增，也使得模板管理变的复杂。所以不仅可以在模板中设置参数，也可以在实际调用API时设置对应的参数。

  - 覆盖顺序

    API参数比模板中对应参数的优先级更高，会覆盖模板中对应的参数。

    举个例子：同一个视频可以转码输出多种清晰度（高清、标清等），不同清晰度的容器格式（MP4)、编码标准(H.264)、帧率（25帧）是相同的，区别只是码率和分辨率的不同。就可以先创建（MP4+H.264+25FPS+2Mbps+1280x720）这样一个默认参数组合模板。调用API时，如果不设置API参数，则按照默认参数（2Mbps+1280x720）执行作业，如果设置API参数（4Mbps+1920x1080），这按照API参数（4Mbps+1920x1080）执行作业。

- 管道

  当用户通过API接口提交作业后，作业会先进入管道中进行排队，根据优先级和提交顺序依次执行。

  管道中的作业可以有多种优先级（最高10，最低1，默认6），相同优先级的作业之间，提交作业时间早的比晚的先执行，不同优先级的作业之间，高优先级的比低优先级的先执行。

- 作业执行和结果

  - 同步和异步

    根据作业的类型不同，一些作业能很快完成，但是大部分的作业都无法即时完成，作业执行有2种方式：同步和异步。同步方式（例如，截图作业）会立即返回结果，异步方式（例如，转码作业）的结果有2种查询方式：定时轮询和消息通知。

  - 定时轮询

    每个作业都有一个唯一作业ID来标识，在提交作业时会同步返回给调用方，之后可以通过作业ID查询作业结果。这种方式的缺点是不够实时。

    消息通知：

    管道可以配置消息通知，就能及时获得作业结果。消息通知中包含了几个重要的信息：作业ID、用户数据和结果。

    ■ 作业ID

      提交作业时记录下作业ID，然后和消息通知的作业ID对比，就能知道是哪个作业的结果。

    ■ 用户数据

      提交作业时，可以填写自定义的用户数据参数（例如，商品ID），然后消息通知中会返回自定义的用户数据参数，这样可以不需要在业务系统中记录作业ID，使用自定义的用户数据（例如，商品ID）来关联业务系统。

1.2 转码模板

由于转码作业的参数很多，每次提交转码作业时都重复填写比较麻烦，转码模板是为解决此问题而提出的概念，本质就是把一些常用的参数组合在一起。转码模板提供了2种类型：预置模板和自定义模板。

- 预置模板

  根据常见的一些参数组合，预先提供给用户的转码模板。参见 预置模板详情。

  预置模板又包括几个细分类型：

  - 预置静态模板

    可以直接使用的转码模板，包含视频转码、音频转码、转封装等各种场景。例如，“MP4-高清”、“MP3-128K”。

  - 窄带高清预置模板

    窄带高清是媒体转码特有的技术。在相同的码率下，能带来更高的清晰度，这样可以在相同成本下提供更好的用户体验。

  - 预置智能模板

    预置智能模板能自动根据输入文件内容的特点动态调整转码参数，在相同的清晰度下，能带来更低的码率，这样可以节省更多的成本。

    > 说明:
    > 使用预置智能模板时，要先调用 提交模板分析作业 接口（SubmitAnalysisJob），分析作业成功完成后，可以调用 查询模板分析作业 接口（QueryAnalysisJobList）获取该输入文件对应的有效预置智能模板列表。若提交的转码作业中指定的预置智能模板不在有效的列表中，则转码作业是无效的，会返回失败。

- 自定义模板

  预置的模板不能满足需求时，可以使用自定义模板来自行定义转码参数(音频、视频、容器、转码等)的组合。每个自定义模板有一个唯一模板ID。

1.3 工作流和媒体

本文介绍媒体处理服务的几个基本概念和关系，以便您更好的理解和使用媒体处理服务。

概念解释：

- 媒体

  媒体包含一个输入（视频、音频多媒体文件）和相关的所有输出（例如，转码、截图、媒体信息、AI标签等）。输入和媒体是一一对应的，由媒体ID唯一标识。

  媒体库

  媒体库是所有媒体的集合，媒体是媒体库的最小管理单元。

- 媒体工作流

  媒体工作流是自动化生产媒体的工厂，由MediaWorkflowId唯一标识。

  > 说明:
  > 媒体工作流在文档中也简称为工作流。

  - 活动

    工作流中的每个节点称为活动。根据实际需求，即可以并行执行（例如，示意图的作业A、B、C之间），也可以串行执行（例如：示意图的作业A1、A2之间）。除开始的输入活动和结束的发布汇报活动，活动支持各种类型的作业（转码作业、截图作业等）。

    ■ 开始的输入活动

      配置工作流关联存储的触发路径，只要在对应的路径上传视频、音频多媒体文件，就会自动触发工作执行。

    ■ 结束的发布汇报活动

      工作流执行完成后，会消息通知执行结果。执行结果包含了媒体ID和多媒体文件的绝对地址，这样就能对应具体是哪个多媒体文件执行完成。

    ■ 作业活动

      作业支持的所有参数，都可以在作业活动中配置。

  - 路径匹配规则

    匹配使用路径前缀的规则，例如，上传的文件是 `http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/test1.flv`，配置的触发路径结果如下：

路径	是否匹配
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C/	是
http://bucket.oss-cn-hangzhou.aliyuncs.com/A/B/C2/	否

路径	是否匹配
http://bucket.oss-cn-hangzhou.aliyuncs.com /A/B/	是
http://bucket.oss-cn-hangzhou.aliyuncs.com /A/B2/	否
http://bucket.oss-cn-hangzhou.aliyuncs.com /A/	是
http://bucket.oss-cn-hangzhou.aliyuncs.com /A2/B/C/	否
http://bucket.oss-cn-hangzhou.aliyuncs.com /A/B/C/test	是
http://bucket.oss-cn-hangzhou.aliyuncs.com /A/B/C/test2	否

- **扩展名匹配规则**

  上传时的自动触发机制会检查文件的扩展名，避免产生一些无效的数据（例如pdf、word文档等）。

  说明:

API手动触发机制不检查扩展名。

文件或者没有扩展名(文件名中不包含扩展名分割符号”.”)，或者扩展名符合下面的规则：

■ 视频

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、webm

■ 音频

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

- 工作流执行

每次上传匹配的多媒体文件都会触发一次执行，同一个多媒体文件如果多次上传，则会触发多次执行，每次执行有唯一的RunId标识。

除了上传时的自动触发机制，工作流针对存储中的存量多媒体文件，也提供了API手动触发机制。每次调用API都会触发一次执行。

- 用户数据

每次执行时，可以填写自定义的用户数据参数（例如，商品ID），然后消息通知中会返回自定义的用户数据参数，这样可以不需要在业务系统中记录媒体ID或者多媒体文件的绝对路径，使用自定义的用户数据（例如，商品ID）来关联业务系统。

2 工作流的开发流程

1. 配置媒体工作流。

   简单易用：通过控制台的图形化界面，按需搭建云端音视频处理流程。

   功能强大：支持截图、转码、窄带高清分析、转封装、水印、剪辑等功能。

   详细的控制台配置参考 媒体工作流。

2. 上传多媒体文件。

   把多媒体文件上传到媒体工作流指定的路径下，会自动触发一个媒体工作流执行，并按照指定的流程进行处理。

3. 等待消息通知。

   执行开始、执行结束都会发送相应的消息通知。

4. 播放。

   成功完成执行后，可以使用URL或者MediaId两种方式来进行播放。

3 上传视频文件

3.1 简介

上传

您可以使用上传SDK上传，支持Web（JavaScript）、移动（Android、iOS）

您也可以通过控制台和第三方工具上传。

功能

简单易用的API，只需指定本地文件和OSS保存位置，

断点续传、多文件队列、超大文件、网络异常事件、安全机制，

自动触发媒体工作流。

触发媒体工作流

多媒体文件成功上传到媒体工作流指定的输入bucket和路径后，会自动触发媒体工作流的执行，按照媒体工作流指定的流程进行处理。

使用上传OSS文件来自动触发媒体工作流的机制，需要满足几个条件：

- 匹配媒体工作流

  工作流触发匹配规则，参见 新增媒体，并且工作流的转态是激活
- 匹配文件扩展名

  触发要求必须是多媒体文件，媒体库服务是通过文件扩展名来判断的。文件或者没有扩展名(文件名中不包含扩展名分割符号”.”)，或者扩展名符合下面的规则：

  - 视频

    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、webm
  - 音频

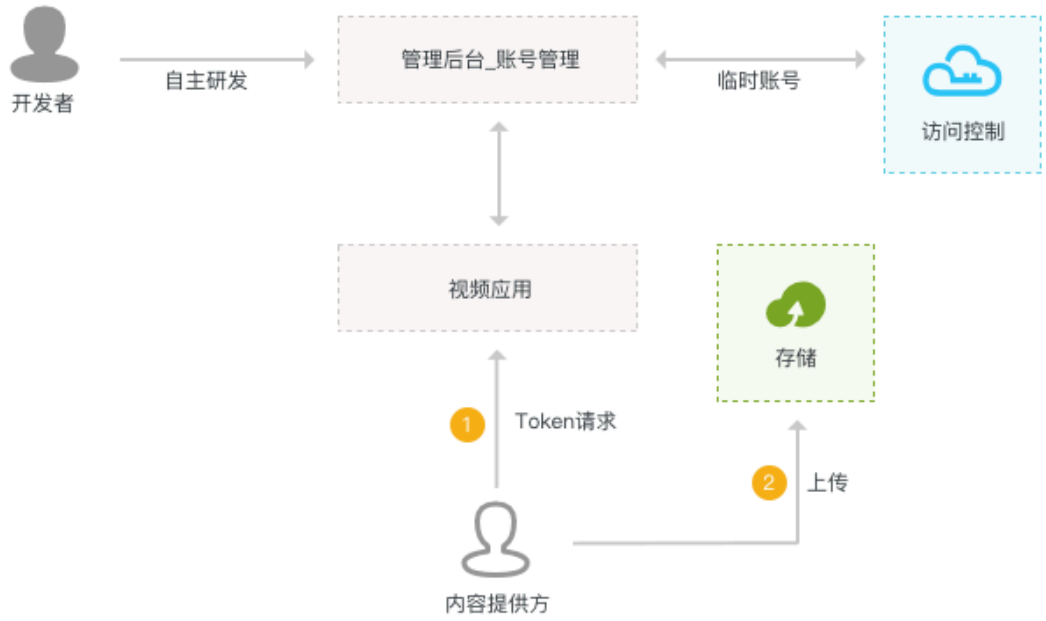
    aac、ac3、acm、amr、ape、caf、flac、m4a、mp3、ra、wav、wma、aiff
- 指定媒体属性

  触发时，也可以指定媒体属性，包括：标题、标签、描述、类目、封面URL、用户自定义数据。详细的属性说明，参见 新增媒体 文档中的请求参数。

安全性

一般的使用场景是通过客户端直接上传视频文件。此时需要考虑AK在客户端保存的安全问题。泄露后，不仅风险大，而且不容易更换。所以建议客户端访问应用服务来获取，并使用 访问控制服务 提供的 安全令牌。

建议流程



1. 请求安全令牌。

   每次上传前，用户通过视频应用（App或Web形式）访问业务方的应用服务，应用服务从访问控制服务获取安全令牌(Token)，然后传回给视频应用。这样即保证了安全性，还可以对用户进行身份验证和权限控制，也可以记录用户上传的历史等。

   使用安全令牌之前，需要先 设置子账号和授权。

   这里提供 Java代码示例。（其他语言的使用方法参考 STS文档）。

2. 上传文件。

把上传SDK集成到视频应用后，可以通过获取到的Token上传文件。参见 使用说明。

- Web

  由于js文件一般放在应用或CDN的域名下，而视频文件放在OSS的域名下，JavaScript上传文件时会涉及跨域请求，需要先 设置CORS。

  JavaScript

- 移动

  Android

  iOS

3.2 设置子账号和授权

请您按照以下步骤设置子账号和授权。

1. **新建子帐号。**

   a. **登录** *RAM* **控制台。**

   b. **在左侧导航栏中，单击 人员管理 > 用户。**

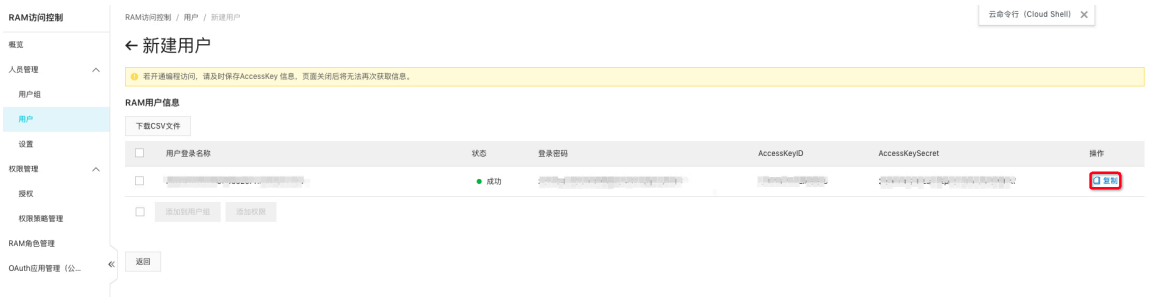
   c. **单击 新建用户。**

   d. **在 创建用户 中，创建一个拥有和主账号一样完全访问MPS权限的子帐号。**

说明:

请您勾选 编程访问。

e. 生成该账号的AccessKey，这一步复制保存用于后续访问。



2. 新建角色。

a. 在左侧导航栏中，单击 RAM角色管理。

b. 单击 新建RAM角色。

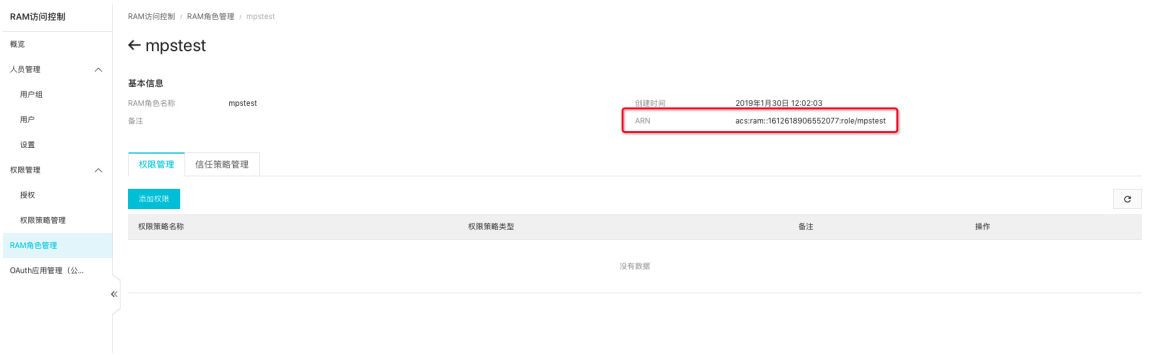
c. 在 选择可信实体类型 中，选择 阿里云账号。

在 受信云账号ID 中，选择 当前云账号，并单击 确定。


新建 RAM 角色
选择可信实体类型
阿里云账号
受信云账号下的子用户可以通过扮演该RAM角色来访问您的云资源，受信云账号可以是当前云账号，也可以是其他云账号
阿里云服务
受信云服务可以通过扮演RAM角色来访问您的云资源。
* 受信云账号ID
当前云账号
其他云账号
* RAM角色名称
mpstest
不超过64个字符，允许英文字母、数字，或"-"
备注
确定
关闭

d. 在 RAM角色管理 中，单击新创建的角色。

e. 在角色 基本信息 中，复制 ARN 参数 acs:ram::example:role/mpstest。



3. 设置角色的权限。

a. 在刚刚创建的角色页面，单击 添加权限。

b. 选择权限。

说明:

如果您想要调整子账号的STS权限（例如，修改、增加、删除等），只需回到本步骤操作即可。

上传SDK需要的最小权限可以在 自定义授权策略 中，新建一个授权策略来实现，然后在编辑角色授权策略中添加这个自定义授权策略即可。完整的策略内容如下：

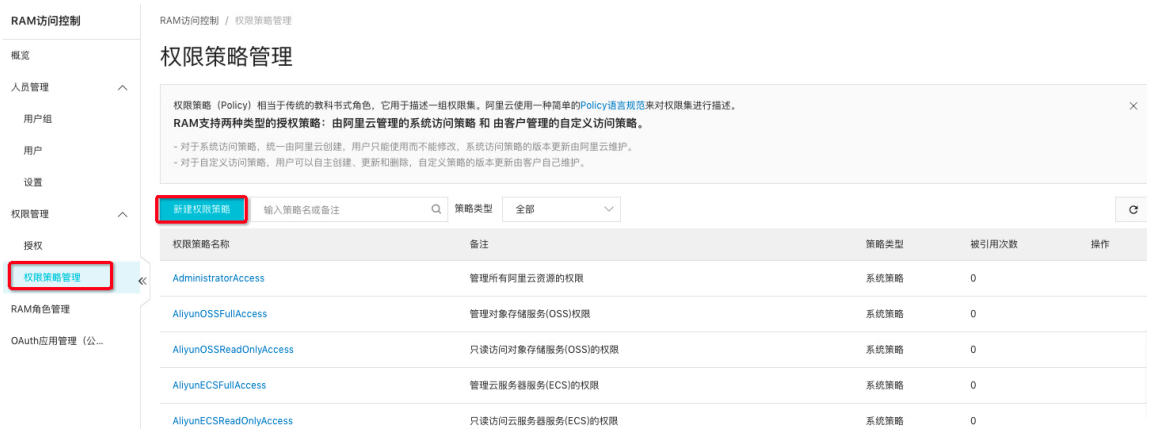
```
{
     "Statement": [
       {
         "Action": [
           "oss:PutObject",
           "oss:AbortMultipartUpload",
           "oss:ListMultipartUploads",
           "oss:ListParts"
         ],
         "Effect": "Allow",
         "Resource": [
           "*"
         ]
       }
     ],
     "Version":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

```
    }
```

4. 关联子账号和角色。

   a. 在 RAM控制台左侧导航栏中，单击 权限管理 > 权限策略管理。

   b. 单击 新建权限策略。

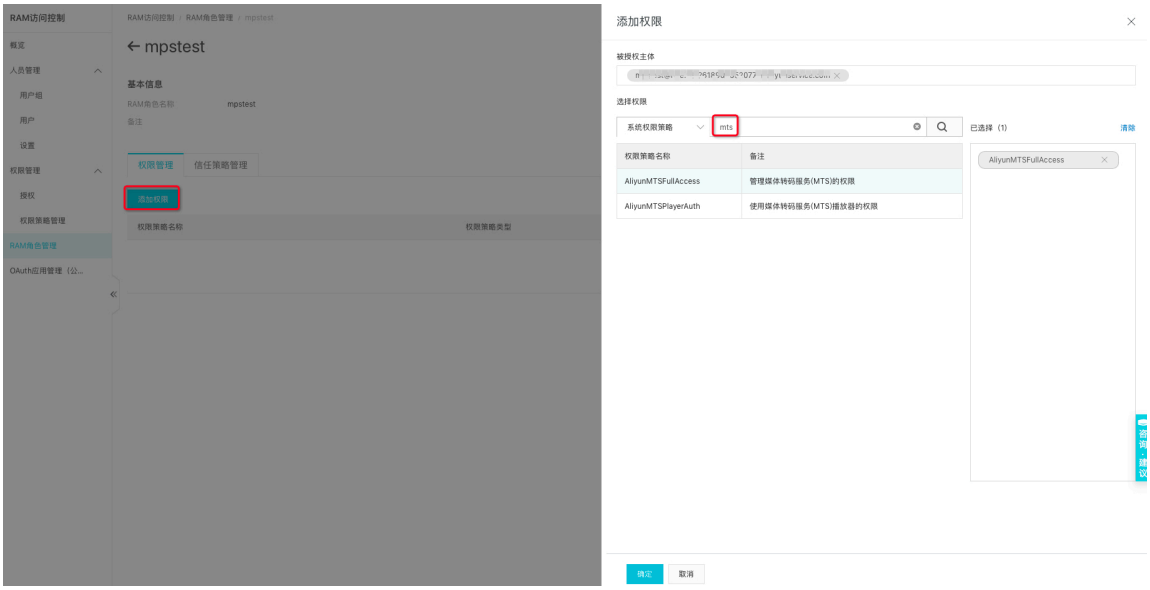


   c. 在 新建自定义权限策略 中，将 Resource 字段，修改成您所获取的 ARN 参数 acs:ram::example:role/mpstest。

   d. 在RAM控制台的左侧导航栏中，单击 人员管理 > 用户。

   e. 选择您所设置的子账号，并单击 添加权限。

   f. 输入创建的策略test可以看到前面创建的`teststspolicy`。

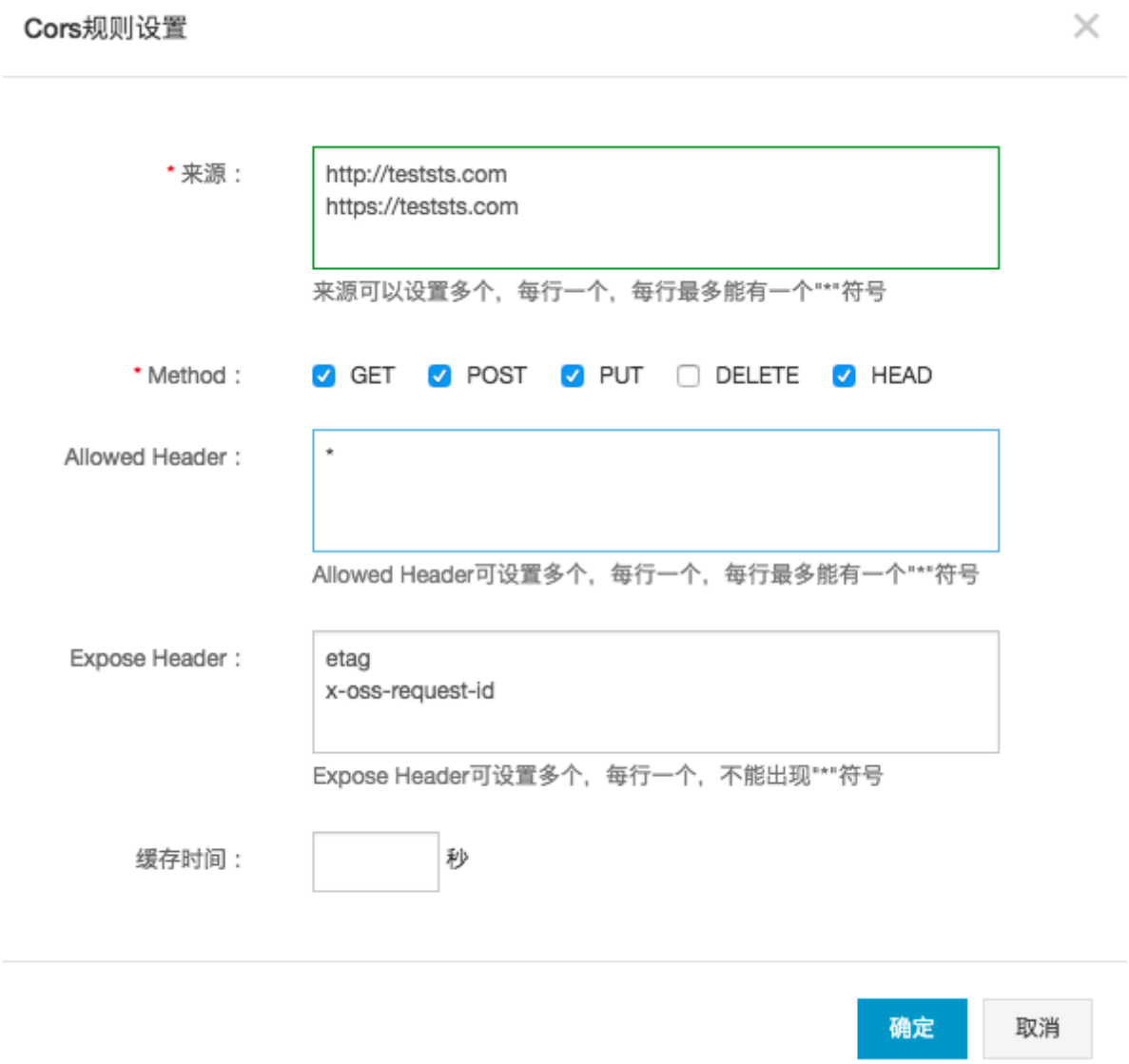


3.3 设置CORS

由于js文件和视频文件放在不同的域名下，JavaScript上传文件时会发起跨域请求。需要设置OSS的跨域设置，否则无法通过Web端的JavaScript正常上传文件。

1. 打开输入bucket对应的CORS设置页面。

2. 添加规则。



- 来源

  填写实际部署js文件的域名，如果需要同时支持http和https访问，需要分别添加。

- Method

  勾选 GET、POST、PUT、HEAD。

- Allowed Header

  填写`*`。

- Expose Header

  分两行填写 etag 和 x-oss-request-id。

3.4 请求安全令牌-Java示例代码

1. pom.xml中引用STS的SDK。

```
<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. 代码。

   STS需要一个角色的参数`roleArn`。登录 *RAM* 控制台，单击 角色管理，再单击具体 角色名称后，在基本信息中有一个参数`Arn`，例如`1351140512345678:role/teststs`。

   - main函数

```
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());
}
```

   - 生成临时AK和Token的函数

```
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有效期。**

   示例代码中生成的Token有效时间为900秒，可以根据实际需求调整（最小900秒，最大3,600秒）。

   在有效期内，不需要反复生成新的Token，可以复用已经生成的Token。您可以通过以下方式判断何时需要重新生成token：

```
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 接收消息通知

4.1 简介

消息格式

媒体工作流开始执行和完成执行时，会向消息服务指定的队列或主题(通知)发送消息。

- 格式定义

  消息体是JSON格式，详细的字段名称、类型、描述参考 新增媒体 的 媒体工作流消息 部分。

  结构的层次定义如下：

  - 顶层

    是一个JSON对象。定义：

    {当前活动的基本属性、工作流执行对象}

  - 当前活动的基本属性

    当前活动的基本属性不是一个独立的对象，是直接属于顶层的键值属性，可以参考下面的示例。定义：

    工作流执行ID、活动名字、活动类型、活动状态、错误信息。

  - 工作流执行详情对象

    是一个JSON对象。定义：

    {工作流执行ID、媒体工作流ID、媒体工作流名字、媒体ID、输入文件、工作流执行类型、活动对象数组、创建时间}

  - 活动对象数组

    是一个JSON数组，包含执行到当前状态的所有活动。例如，开始消息中只有一个Start活动对象，完成消息则包含所有活动对象。定义：

    [活动对象、活动对象…]

  - 活动对象

    是一个JSON对象。定义：

    {活动名字、活动类型、作业ID、活动状态、开始时间、结束时间、错误信息}

- 开始

  活动基本属性中“活动类型”是“Start”。

- 完成

  活动基本属性中“活动类型”是“Report”。

- 示例

```
{
      "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"
    }
}
```

接收和解析消息

- 队列

  PHP示例代码

- 主题（通知）

  PHP示例代码

4.2 队列方式接收通知

本文介绍消息服务的要求和安装说明。

详情参见SDK下载和队列使用手册。

示例的语言采用PHP，其他语言使用说明，参见SDK使用手册。

环境要求

PHP 5.5+

安装

从阿里云下载消息服务的PHP SDK。

从阿里云下载消息服务的PHP SDK。

示例的语言采用PHP，其他语言使用说明，参见SDK使用手册。

解压到项目目录，解压后的目录名是`php_sdk`。

示例代码

- 引用消息服务SDK

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

- 初始化消息服务

  MNS对用户的每个地域都配置了一个单独的服务域名，规则是：`https://${UserId}.mns.${Region}.aliyuncs.com`。示例代码使用的 华东1地域(cn-hangzhou)，也可以替换为其他地域，例如：华北2地域（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);
```

- **接收消息**

  MNS接收到的每条消息都对应一个句柄，后续可以使用句柄操作这条消息（例如：删除消息）。

  另外，MNS支持批量接收消息来提高性能。详情参见MNS批量消费消息。

  接收消息时，可以指定超时时间（示例设置了3秒超时），如果队列中没有消息，会发生超时并触发异常。

```
$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";
 }
```

- **删除消息**

  消息不会主动从队列删除，必须主动调用删除消息，否则消息会一直保持在队列中，下次还会继续接收到同一个消息。另外，删除消息操作必须在接收到消息后指定时间内调用才能成功，详情参见消费服务-删除消息说明。

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

- **分析消息**

  消息体是字符串，内容是一个JSON对象，需要通过`json_decode`转换成对象，然后就可以分析JSON对象来获取详细信息了，示例打印了这次消息是哪个输出文件触发的媒体工作流执行。

```
$json_message = json_decode($message);
 $input_file = $json_message->{'MediaWorkflowExecution'}->{'Input'}-
>{'InputFile'};
 echo '输入文件 location:'.$input_file->{'Location'}.
```

```
      ' bucket:'.$input_file->{'Bucket'}.
      ' object:'.$input_file->{'Object'}."\n";
```

- 获取视频输出的详细信息。

  获取到消息详细内容后，可以配合使用媒体库服务API获取工作流执行的视频详细信息。示例打印出这次转码和截图作业的输出地址。

  如何安装和配置媒体库服务的PHP SDK，参考文档 媒体库SDK-PHP。

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

  初始媒体库服务的client。

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

  打印所有转码作业的输出地址和基本信息。

```
if (strcmp($json_message->{'Type'}, 'Report') == 0) {
      $activities = $json_message->{'MediaWorkflowExecution'}->{'
ActivityList'};
      $transcode_job_ids = Array();
      for ($i=0; $i < count($activities); $i++) {
        if (strcmp($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 '转码输出文件URL '.'http://'.$output_file->{'Bucket
'}.'.'.
                        $output_file->{'Location'}.'.aliyuncs.com/'.
                        urldecode($output_file->{'Object'})."\n";
        echo '转码输出文件基本信息 '.$video_properties->{'Width'}.'x'.$
video_properties->{'Height'}.
                        ' duration:'.$video_properties->{'Duration
'}."\n";
      }
 }
```

  打印所有截图作业的输出地址。

```
if (strcmp($json_message->{'Type'}, 'Report') == 0) {
      $activities = $json_message->{'MediaWorkflowExecution'}->{'
ActivityList'};
      $snapshot_job_ids = Array();
```

```
      for ($i=0; $i < count($activities); $i++) {
        if (strcmp($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 '截图输出文件URL '.'http://'.$output_file->{'Bucket
'}.'.'.
                        $output_file->{'Location'}.'.aliyuncs.com/'.
                        urldecode($output_file->{'Object'})."\n";
      }
 }
```

4.3 主题通知方式接收消息

主题（通知）是消息服务主动推送消息给用户，只要有能公开访问的HTTP服务即可接收。

基本结构

视频媒体库通知的基本结构是这样的：

- 最外层是消息服务的结构体

  消息服务的详细定义和格式，参见消息服务-Notification操作。

- 消息服务的消息正文是媒体库服务的结构体

  取到消息正文后，就能进一步对媒体库服务的消息进行解析了，参见 队列方式接收消息，文档中有详细的解析步骤和示例代码。

安全

主题（通知）方式是公开访问的HTTP服务，所以要避免非法的攻击调用。关于识别消息是否由消息服务发起，参见消息服务文档 Endpoint签名认证。

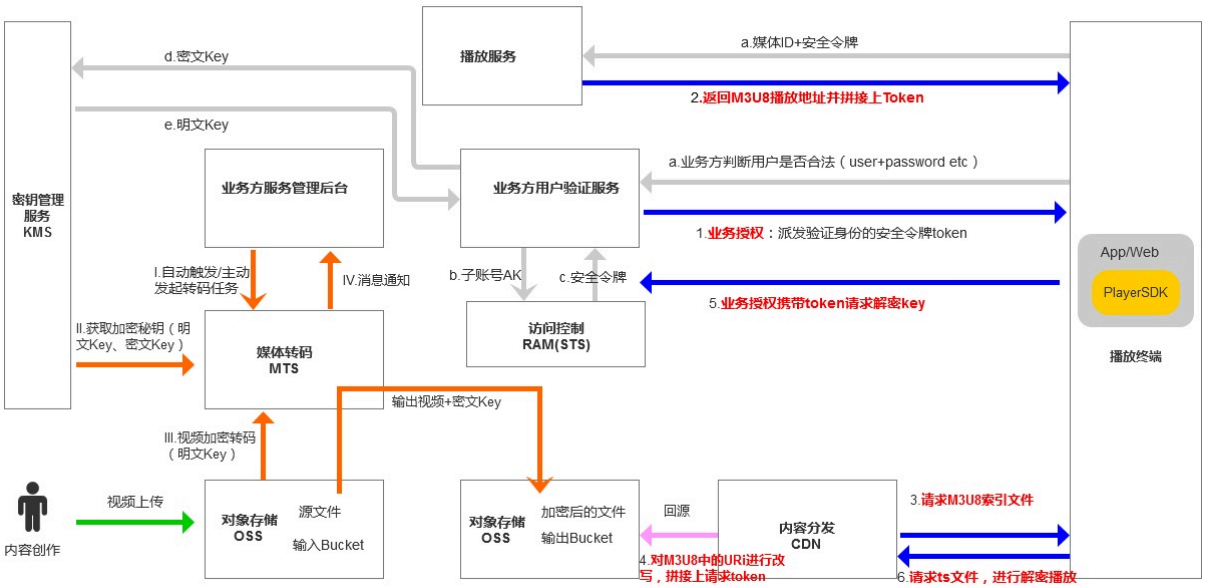
5 视频加密

5.1 HLS标准加密

视频加密是对视频内容保护的一种手段，对视频中的内容进行加密，可有效防止视频泄露和盗链问题，广泛用于在线教育及财经等领域。

> **说明:**
> 阿里云目前支持两种加密方式: 私有加密 和 HLS标准加密，HLS标准加密需要客户自己保护密钥，此文档介绍HLS标准加密。

加密架构



术语介绍

- 秘钥管理服务（Key Management Service，简称KMS）

  一项安全管理服务，主要负责数据秘钥的生产、加密、解密等工作。开通请 单击这里。
- 数据秘钥（Data Key，简称DK）也称明文密钥

  DK为加密数据使用的明文数据密钥。
- 信封数据密钥（Enveloped Data Key，简称EDK）也称密文密钥

  EDK为通过信封加密技术保密后的密文数据密钥。

- 访问控制（Resource Access Management 简称RAM）

  是阿里云为客户提供的用户身份管理与资源访问控制服务。开通请 单击这里。

操作流程

1. 创建HLS加密工作流。

   说明:

   控制台暂不支持创建HLS加密工作流，请通过API进行创建。查看demo，参见 创建HLS标准加密工作流。创建好后，不要在控制台对其修改，会使加密配置失效。

   工作流中关键配置:

   - 开始活动节点: `InputFile:{"Bucket":"bucketdemo", "Location ":"oss-cn-hangzhou", "ObjectPrefix":"HLS-Encryption"}`

     此配置表示内容创作者上传视频到杭州 oss://bucketdemo/HLS-Encryption 这个路径下会自动触发加密转码。

   - 转码活动节点: `Encryption:{"Type":"hls-aes-128", "KeyUri":"https://decrypt.demo.com"}`

     转码完成后，KeyUri的配置会出现在m3u8文件中，供播放器使用。

     播放时播放器会携带EDK密文密钥请求这个地址，以获取DK明文密钥，进行播放。

2. 上传视频。

   两种方法上传视频，都会自动触发加密转码。

   - 通过MPS控制台上传视频至刚刚创建的工作流。
   - 通过OSS上传工具上传视频至oss://bucketdemo/HLS-Encryption路径。

   转码完成后，m3u8文件内容示例。

```
#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. 播放。

   · 获取播放地址，可通过查询媒体接口，参见 查询媒体-使用媒体ID 获取OSS地址，然后把OSS域名，替换为CDN域名，再拼接上参数MtsHlsUriToken，此参数会作为请求解密密钥的令牌，以下为原理。

   播放时，播放器会访问m3u8文件中EXT-X-KEY标签中的URI以获取解密秘钥，此URI为业务方搭建的解密密钥接口，只允许合法用户访问。 因此，需要播放器在请求解密时，携带一些业务方承认的认证信息。MtsHlsUriToken就是这个作用，业务方颁发一令牌给播放器，播放器请求解密密钥时，携带此令牌，业务方验证令牌的合法性。

   · 播放器携带令牌，业务方验证服务。

   例如，正常的播放地址为：`https://vod.demo.com/test.m3u8`，当拼接携带`MtsHlsUriToken`参数后为`https://vod.demo.com/test.m3u8?MtsHlsUriToken=`业务方颁发的令牌。

   播放时，播放器向阿里CDN请求`https://vod.demo.com/test.m3u8?MtsHlsUriToken=`业务方颁发的令牌，阿里CDN会动态修改m3u8文件中的解密URI，如原为`https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271`，修改后为 `https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271&MtsHlsUriToken=`业务方颁发的令牌。

   所以，播放器最终请求解密URI为：`https://decrypt.demo.com?Ciphertext=aabbccddeeff&MediaId=fbbf98691ea44b7c82dd75c5bc8b9271&MtsHlsUriToken=`业务方颁发的令牌。此地址中，携带了业务方搬发的令牌，业务方进行验证即可。

4. 业务方需要完成以下操作。

   a. 搭建颁发及验证MtsHlsUriToken令牌服务。

   b. 校验解密令牌，推荐一个令牌只允许使用一次。

   c. 解密密钥：EDK即Ciphertext，此时，要调用KMS服务的解密接口进行解密，参见 *Decrypt*。 解密后，可缓存，以减少不必要的网络IO。

   d. 解密拿到DK即明文密钥，需要base64decode，然后返回给播放器。

6 媒体库管理

6.1 简介

您可以使用媒体库SDK，支持 *Java*、*PHP*、*Python*。

也可以直接通过HTTP、HTTPS访问API，详情参考 *API*使用手册。

功能

媒体工作流管理：增、删、改、查以及激活和停止。

媒体工作流执行实例管理：遍历和查询。

媒体管理：增、删、改、查和搜索。以及媒体属性的维护（标题、标签、封面、描述）。以及媒体发布状态的设置。

媒体类目管理：增、删、改、查。

业务场景

- 搜索媒体

  在媒体库中搜索满足条件的媒体集合。

  可以通过关键字来搜索，是逻辑“或”的关系：只要标题、标签、描述、类目任一属性能匹配即可。也可以通过组合条件来搜索，是逻辑“与”的关系：指定的组合属性中（标题、标签、描述、类目），必须每个属性都匹配。

  在搜索条件中，可以指定创建时间的区间来限定搜索范围。返回的结果也可以指定排序规则：按照创建时间升序或降序。

  另外，返回的接口比较多时，可以选择分页获取。

- 维护媒体属性

  每个媒体都有基本属性：标题、标签、描述、类目。都可以通过API来设置。

  基本属性-示例代码-*PHP*

- 管理媒体标签

  媒体库不提供全局的标签管理，每个媒体的标签都是独立的设置，可以通过搜索媒体的API来查找所有设置了相同标签的媒体。

  标签管理-示例代码-*PHP*

管理媒体类目

媒体库提供了一个全局的类目管理，每个媒体可以关联到一个类目，并结合搜索媒体功能来快速检索。

类目管理-示例代码-PHP

查询媒体详细信息

一个媒体包含一个输入和若干个输出（视频、截图等）。可以通过查询返回媒体的详细输入、输出的信息。

输入信息包含：视频基本属性（宽、高、时长、大小、码率、帧率）以及视频详情（容器封装、视频、音频、字幕流，以及封装和流的详细属性）。

输出信息包含：视频有基本属性（宽、高、时长、大小、码率、帧率）以及OSS的URL地址。截图有类型（单帧、批量）以及OSS的URL地址。

媒体详细信息-示例代码-PHP

6.2 视频基本属性

简介

本文介绍如何查询和更新媒体基本信息。SDK的安装和使用，参见 媒体库SDK-PHP。

查询媒体基本信息

查询媒体提供了2种方式：媒体ID或OSS文件地址。

- 使用媒体ID查询媒体

  详细参数参考 API使用手册 > 媒体接口 > 查询媒体 > 使用媒体ID。

```
include_once 'aliyun-php-sdk-core/Config.php';
use Mts\Request\V20140618 as Mts;
$accessKeyID = 'test'; // 替换成真实的id
$accessKeySecret = 'test'; // 替换成真实的secret
$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'; // 替换成真实的mediaID
$medias = queryMediaById($client, $mediaID)->{'MediaList'}->{'Media
'};
for ($i=0; $i < count($medias); $i++) {
     printMedia($medias[$i]);
}
```

- **使用OSS文件地址查询媒体**

  详细参数参考 *API*使用手册 > 媒体接口 > 查询媒体 > 使用*OSS*文件地址。

```
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/';
// OSS的Object不需要"/"开始，替换成真实的ossObject
$ossObject = 'test/测试.mp4';
$medias = queryMediaByURL($client,$ossEndpoint.urlencode($ossObject
))->{'MediaList'}->{'Media'};
for ($i=0; $i < count($medias); $i++) {
     printMedia($medias[$i]);
}
```

- **更新属性**

  更新提供了2种更新方式：全量属性更新，单个属性更新。

  - **全量更新属性**

    详细参数参考 *API*使用手册 > 媒体接口 > 更新媒体 > 基本信息。

    更新时，必须指定所有字段，不设置的字段会被清空。

```
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'; // 替换成真实的mediaID
$media = updateMediaAllField($client, $mediaID,
                        'title', 'description', 'tags', 'coverURL
')->{'Media'};
```

- **单个更新属性**

  不同的字段可以单独更新，使用的是不同API，可以不修改其他字段的情况下，方便的更新单个字段。

  这里通过“发布状态”举例，详细参数参考 *API*使用手册 > 媒体接口 > 更新媒体 > 发布状态。

```
function updateMediaPublishState($client, $mediaID, $state)
{
    $request = new Mts\UpdateMediaPublishStateRequest();
    $request->setAcceptFormat('JSON');
    $request->setMediaId($mediaID);
    $request->setPublish($state);
    $response = $client->getAcsResponse($request);
    return $response;
}
$mediaID = 'test'; // 替换成真实的mediaID
// 更新"发布状态"的API没有返回值，通过捕获异常来判断是否执行成功
try {
    updateMediaPublishState($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 媒体详细信息

简介

SDK的安装和使用，详情参考 媒体库*SDK-PHP*

一个媒体包含一个输入文件和若干输出文件。输入除了基本信息之外，还有详细的 媒体信息。输出可以查询视频和截图的详细信息。

输入媒体信息

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

```
 $accessKeyID = 'test'; // 替换成真实的id
 $accessKeySecret = 'test'; // 替换成真实的secret
 $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'; // 替换成真实的mediaID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printMediaInfo($medias[$i]->{'MediaInfo'});
}
```

输出

- **视频**

```
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'; // 替换成真实的mediaID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printOutputVideos($medias[$i]->{'PlayList'}->{'Play'});
}
```

- **截图**

```
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'; // 替换成真实的mediaID
$medias = queryMedia($client, $mediaID)->{'MediaList'}->{'Media'};
for ($i = 0; $i < count($medias); $i++) {
    printOutputSnapshots($medias[$i]->{'SnapshotList'}->{'Snapshot
'});
}
```

6.4 标签管理

简介

SDK的安装和使用，详情参考 媒体库*SDK-PHP*。

媒体库不提供全局的标签管理和设置，每个媒体的标签都是独立的。可以通过搜索媒体的API来查找所有设置了相同标签的媒体。

标签的API支持单个标签的添加和删除，如果要一次设置多个标签，可以通过 更新媒体-基本信息 实现。

添加标签

详细参数参考 *API*使用手册 > 媒体接口 > 更新媒体 > 添加标签。

```
include_once 'aliyun-php-sdk-core/Config.php';
  use Mts\Request\V20140618 as Mts;
  $accessKeyID = 'test'; // 替换成真实的id
  $accessKeySecret = 'test'; // 替换成真实的secret
  $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'; // 替换成真实的mediaID
  // API没有返回值，通过捕获异常来判断是否执行成功
  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);
    }
```

删除标签

详细参数参考 API使用手册 > 媒体接口 > 更新媒体 > 删除标签。

```
function deleteMediaTag($client, $mediaID, $tag)
    {
        $request = new Mts\DeleteMediaTagRequest();
        $request->setAcceptFormat('JSON');
        $request->setMediaId($mediaID);
        $request->setTag($tag);
        $response = $client->getAcsResponse($request);
        return $response;
    }
    $mediaID = 'test'; // 替换成真实的mediaID
    // API没有返回值，通过捕获异常来判断是否执行成功
    try {
        deleteMediaTag($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 类目管理

简介

SDK的安装和使用，参见 媒体库SDK-PHP。

类目支持增删改查。您需要关注以下内容：

- 删除一个类目并不会自动清除关联媒体的类目ID属性。
- 查询类目返回接口有两种形式：树结构、列表结构。树结构返回的是一个嵌套结构的JSON对象，列表结构返回的是一个平面的数组结构，可以根据场景选择使用。

新增类目

参见 API使用手册 > 媒体类目接口 > 新增类目。

```
include_once 'aliyun-php-sdk-core/Config.php';
    use Mts\Request\V20140618 as Mts;
    $accessKeyID = 'test'; // 替换成真实的id
    $accessKeySecret = 'test'; // 替换成真实的secret
    $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");
```

更新类目

参见 *API*使用手册 > 媒体类目接口 > 更新类目。

```
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'); // 替换成真实
的CateID
  } catch (ClientException $e) {
      print_r('ClientException:'."\n");
      print_r($e);
  } catch (ServerException $e) {
      print_r('ServerException:'."\n");
      print_r($e);
  }
```

删除类目

参见 *API*使用手册 > 媒体类目接口 > 删除类目。

```
function deleteCategory($client, $categoryId)
  {
      $request = new Mts\DeleteCategoryRequest();
      $request->setAcceptFormat('JSON');
      $request->setCateId($categoryId);
      $response = $client->getAcsResponse($request);
      return $response;
  }
  try {
      deleteCategory($client, 12345678); // 替换成真实的CateID
  } catch (ClientException $e) {
      print_r('ClientException:'."\n");
      print_r($e);
  } catch (ServerException $e) {
```

```
    print_r('ServerException:'."\n");
    print_r($e);
}
```

查询类目

- **树结构**

  参见 *API*使用手册 > 媒体类目接口 > 查询类目 > 树。

```
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));
```

- **列表结构参见** *API*使用手册 > 媒体类目接口 > 查询类目 > 列表。

```
 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
");
    }
```