阿里云
媒体处理

最佳实践

文档版本：20190410

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 转码

1.1 如何进行简单转码

简介

转码是指把一个OSS的输入文件按照指定的参数进行处理的过程，并把结果输出到指定的OSS文件。提交转码 作业 时，有几个需要关注的对象：

- *Input*

  指定OSS的输入文件。

  > 说明:
  > OSS的Location必须和媒体处理服务的地域对应，例如OSS的oss-cn-hangzhou对应媒体处理的cn-hangzhou。

- *Output*

  每个转码作业可以指定若干个Output对象，内部包含多个参数和子对象，这里介绍三个重要参数/子对象：

  - *Container*

    输出的容器类型（文件格式）。视频支持mp4、flv、ts、m3u8，音频支持mp3、mp4等。

  - *Video*

    输出的视频参数。例如编码格式、码率、宽、高、帧率等。

  - *Audio*

    输出的音频参数。例如编码格式、码率、声道数、采样率等。

  - TemplateId

    API指定的参数比模板配置的参数优先级更高，会覆盖模板中配置的对应参数。

    媒体处理预先提供了 预置静态模板。

    您也可以 创建自定义转码模板。

- PipelineId

  每个地域默认都提供了一个管道，您可以登录 媒体处理控制台，单击 媒体管理 > 全局设置 > 管道 进行查询。

场景

把任意格式的视频转码成720P(1280x720)清晰度的MP4视频文件，设置的音视频参数如下：

- 视频

  使用H.264编码器，码率1500Kbps，宽度1280，高度自适应（避免设置为固定值导致画面不成比例的缩放），帧率25帧。

- 音频

  使用AAC编码器，码率128Kbps，声道数为2，采样率44100。

- 转码模板

  使用预置静态模板”MP4-流畅”：S00000001-200010，模板配置的视频码率400Kbps，音频码率64Kbps，宽度640（高度自适应）。

- 输出结果

  由于API参数会覆盖模板参数，所以输出的视频中，码率是1500Kbps、宽度是1280、帧率是25，输出的音频中，码率128Kps、声道2、采样率44100。

示例代码

简单转码-*Java*

简单转码-*Python*

简单转码-*PHP*

1.2 如何进行API转码

背景

工作流无法满足用户场景时，需用户自己判断业务逻辑，使用API提交转码任务。例如：并不是所有的视频都需要转码，不同视频需要设置不同的转码配置。

优势

- 自定义业务逻辑，灵活提交转码作业。
- 功能强大，支持转码、转封装、水印、支持HLS-AES128标准加密、剪辑等功能。
- 转码任务执行完成，支持向指定的消息队列或消息通知发送执行信息。
- 支持URL播放。

使用限制

- 一个转码作业生成一个输出文件，允许批量提交作业。

- API转码支持HLS-AES128标准加密，暂不支持阿里云私有加密。
- API转码支持URL播放，不支持媒体ID播放。需用户自己关联多个格式的多个清晰度输出，实现多清晰度自动切换、多格式支持等逻辑。

准备

- 自定义转码模板（按需），进入媒体处理控制台 设置。
- 自定义水印模板（按需），进入媒体处理控制台 设置。

操作步骤

1. 输入文件 上传到OSS（多种上传方案：OSS控制台上传，使用OSS相关上传工具上传，上传SDK）。
2. 设置管道消息队列通知。
3. 提交转码任务。
4. 在获取到消息后，调用“查询转码作业”接口查询作业执行结果，获取输出文件URL。
5. 通过URL播放视频。

搭建一个给视频添加水印的应用服务

JAVA源代码下载

1.3 如何进行媒体工作流转码

背景

1个输入文件对应多个输出文件（不同分辨率、不同格式等），通过控制台的图形化界面，快速搭建常用视频处理流程。

优势

- 简单易用，视频上传完成自动触发转码任务。
- 功能强大，支持截图、转码、转封装、水印、加密、剪辑等功能。
- 在媒体工作流开始执行和完成执行时，支持向指定的消息队列或消息通知发送工作流执行信息。
- 媒体库，为您提供音视频管理功能。媒体ID关联多个格式的多个清晰度输出，使用媒体ID播放，可以实现多清晰度自动切换、多格式支持。
- 支持URL和媒体ID播放。

参见 工作流的开发流程。

使用限制

- 一个工作流只能配置1个输入路径，媒体工作流将处理此路径下的所有视频。

- 上传到输入目录下的视频均会触发工作流转码，目前仅支持简单的条件转码场景，可能无法支持部分复杂的业务逻辑。
- 工作流支持阿里云私有加密，暂不支持HLS-AES128标准加密。

准备

- 自定义转码模板#按需#。
- 自定义水印模板#按需#。

操作步骤

1. 添加输入/输出媒体*Bucket*。
2. 创建媒体工作流，在工作流中可以灵活配置截图、转码、转封装、水印、加密、剪辑等功能。
3. CDN加速域名（非必填项）：如果您需要为您的域名开启内容分发加速功能，参见 域名管理。
4. 上传视频：您可以通过媒体处理控制台或使用OSS相关上传工具上传视频文件。同时，我们也提供覆盖所有平台的上传SDK。
5. 管理视频。
6. 视频播放。

搭建一个视频转码应用服务

*JAVA*源码下载

2 加密

2.1 如何进行HLS的加密与播放

本文目的

示例创建HLS标准加密工作流到播放加密视频的一个完整步骤。

HLS标准加密架构，参见 *HLS的加密与播放*。

操作步骤

1. 创建HLS加密工作流。

   创建HLS加密工作流，DEMO代码，参见 *创建HLS标准加密工作流*。

   > 说明:
   > 创建HLS标准工作流时，为了测试，参数 HLS_KEY_URI 值填 http: //127.0.0.1:8888。播放时，播放器会到这个地址请求密钥，我们会在本地起一个服务，进行分发密钥。

2. 上传及加密视频。

   在控制台的媒体库中，上传视频，选择工作流时，选择刚刚创建的HLS标准加密工作流，上传完成后，会自动触发加密转码。待状态为发布时，进行下一步。

3. 开启本地鉴权服务。

   搭建一个本地HTTP服务，作为播放HLS标准加密视频的鉴权服务，颁发及验证MtsHlsUriToken令牌。

   Java示例代码依赖：

   https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-core

   https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-kms

```
package com.aliyun.smallcode;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.ProtocolType;
import com.aliyuncs.kms.model.v20160120.DecryptRequest;
import com.aliyuncs.kms.model.v20160120.DecryptResponse;
import com.aliyuncs.profile.DefaultProfile;
import com.sun.net.httpserver.Headers;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.spi.HttpServerProvider;
```

```
import org.apache.commons.codec.binary.Base64;
import java.io.IOException;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.URI;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class AuthorizationServer {
private static DefaultAcsClient client;
static {
String region = "";
String accessKeyId = "";
String accessKeySecret = "";
client = new DefaultAcsClient(DefaultProfile.getProfile(region,
accessKeyId, accessKeySecret));
}
public class AuthorizationHandler implements HttpHandler {
public void handle(HttpExchange httpExchange) throws IOException {
String requestMethod = httpExchange.getRequestMethod();
if(requestMethod.equalsIgnoreCase("GET")){
//从URL中取得密文密钥
String ciphertext = getCiphertext(httpExchange);
if (null == ciphertext)
return;
//从KMS中解密出来，并Base64 decode
byte[] key = decrypt(ciphertext);
//设置header
setHeader(httpExchange, key);
//返回密钥
OutputStream responseBody = httpExchange.getResponseBody();
responseBody.write(key);
responseBody.close();
}
}
private void setHeader(HttpExchange httpExchange, byte[] key) throws
 IOException {
Headers responseHeaders = httpExchange.getResponseHeaders();
responseHeaders.set("Access-Control-Allow-Origin", "*");
httpExchange.sendResponseHeaders(HttpURLConnection.HTTP_OK, key.
length);
}
private byte[] decrypt(String ciphertext) {
DecryptRequest request = new DecryptRequest();
request.setCiphertextBlob(ciphertext);
request.setProtocol(ProtocolType.HTTPS);
try {
DecryptResponse response = client.getAcsResponse(request);
String plaintext = response.getPlaintext();
//注意：需要base64 decode
return Base64.decodeBase64(plaintext);
} catch (ClientException e) {
e.printStackTrace();
return null;
}
}
private String getCiphertext(HttpExchange httpExchange) {
URI uri = httpExchange.getRequestURI();
String queryString = uri.getQuery();
String pattern = "Ciphertext=(\\w*)";
Pattern r = Pattern.compile(pattern);
Matcher m = r.matcher(queryString);
if (m.find())
return m.group(1);
```

```
else {
System.out.println("Not Found Ciphertext");
return null;
}
}
}
private void startService() throws IOException {
HttpServerProvider provider = HttpServerProvider.provider();
//监听端口8888,能同时接受10个请求
HttpServer httpserver = provider.createHttpServer(new InetSocket
Address(8888), 10);
httpserver.createContext("/", new AuthorizationHandler());
httpserver.start();
System.out.println("server started");
}
public static void main(String[] args) throws IOException {
AuthorizationServer server = new AuthorizationServer();
server.startService();
}
}
```

Python示例代码依赖：

pip install aliyun-python-sdk-core

pip install aliyun-python-sdk-kms

pip install aliyun-python-sdk-mts

```
# -*- coding: UTF-8 -*-
from BaseHTTPServer import BaseHTTPRequestHandler
from aliyunsdkcore.client import AcsClient
from aliyunsdkkms.request.v20160120 import DecryptRequest
import cgi
import json
import base64
import urlparse
client = AcsClient("","","");
class AuthorizationHandler(BaseHTTPRequestHandler):
def do_GET(self):
self.check()
self.set_header()
cipertext = self.get_cihpertext()
plaintext = self.decrypt_cihpertext(cipertext)
print plaintext
key = base64.b64decode(plaintext)
print key
self.wfile.write(key)
def do_POST(self):
pass
def check(self):
#check MtsHlsUriToken, etc.
pass
def set_header(self):
self.send_response(200)
#cors
self.send_header('Access-Control-Allow-Origin', '*')
self.end_headers()
def get_cihpertext(self):
path = urlparse.urlparse(self.path)
query = urlparse.parse_qs(path.query)
```

```
return query.get('Ciphertext')[0]
def decrypt_cihpertext(self, cipertext):
request = DecryptRequest.DecryptRequest()
request.set_CiphertextBlob(cipertext)
response = client.do_action_with_exception(request)
jsonResp = json.loads(response)
return jsonResp["Plaintext"]
if __name__ == '__main__':
# Start a simple server, and loop forever
from BaseHTTPServer import HTTPServer
print "Starting server, use  to stop"
server = HTTPServer(('127.0.0.1', 8888), AuthorizationHandler)
server.serve_forever()
```

4. 获取播放地址。

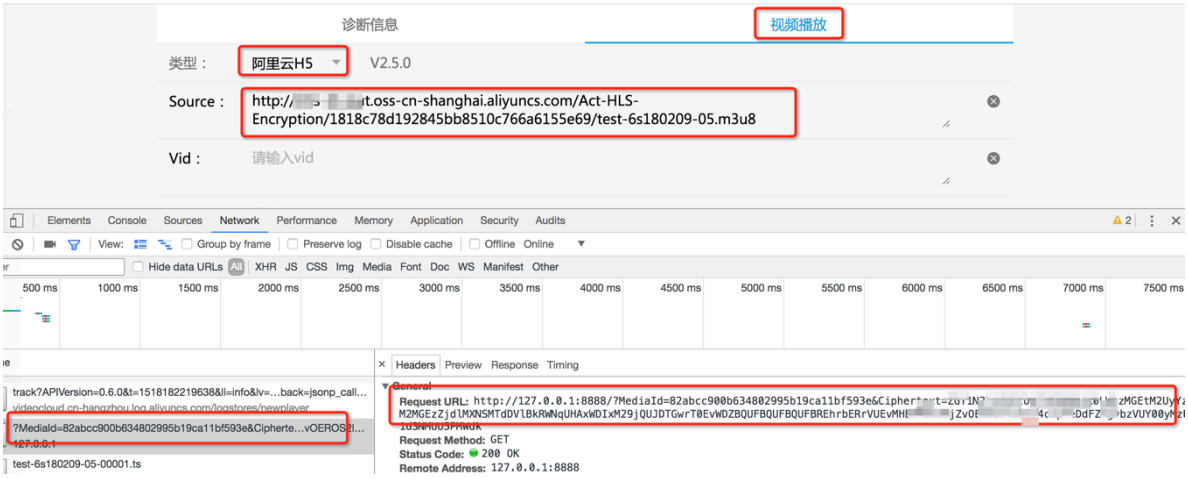
   多种方式可获取。详情参见 媒体转码输出文件相关问题。

5. 播放视频。

   借助一个在线播放器，测试HLS加密视频的播放。详情参见 阿里云播放器用户诊断工具。

   将第 4 步中获取的播放地址，如图填入对话框中，单击 视频播放 即可。

> **说明:**
> 通过浏览器DEBUG，可以看到播放器自动请求了鉴权服务器，获取解密密钥，并进行解密播放。



3 如何上传视频

背景

本文主要介绍如何基于OSS服务和MPS的上传SDK，快速搭建一个音视频文件上传服务。

优势

使用MPS的上传SDK上传音视频文件，具有以下优势：

- 增加文件列表管理功能。
- 增加STS Token 超时更新功能。
- 增加上传过程中网络抖动时的自动重试功能。
- 文件断点续传功能。
- 自动触发MPS服务的媒体工作流。
- 配置媒体标题、标签、描述、类目、封面URL、等功能。

> 说明：
>
> - 断点续传的限制条件：不允许跨生命周期。JS端页面不能刷新、关闭，android/iOS不能关闭APP、手机。
> - 同一本地文件只能上传一次。

服务端搭建

考虑移动端AK安全性问题，选用STS的方式上传文件。关于STS如何增加上传的安全性，参见*RAM和STS使用指南*。

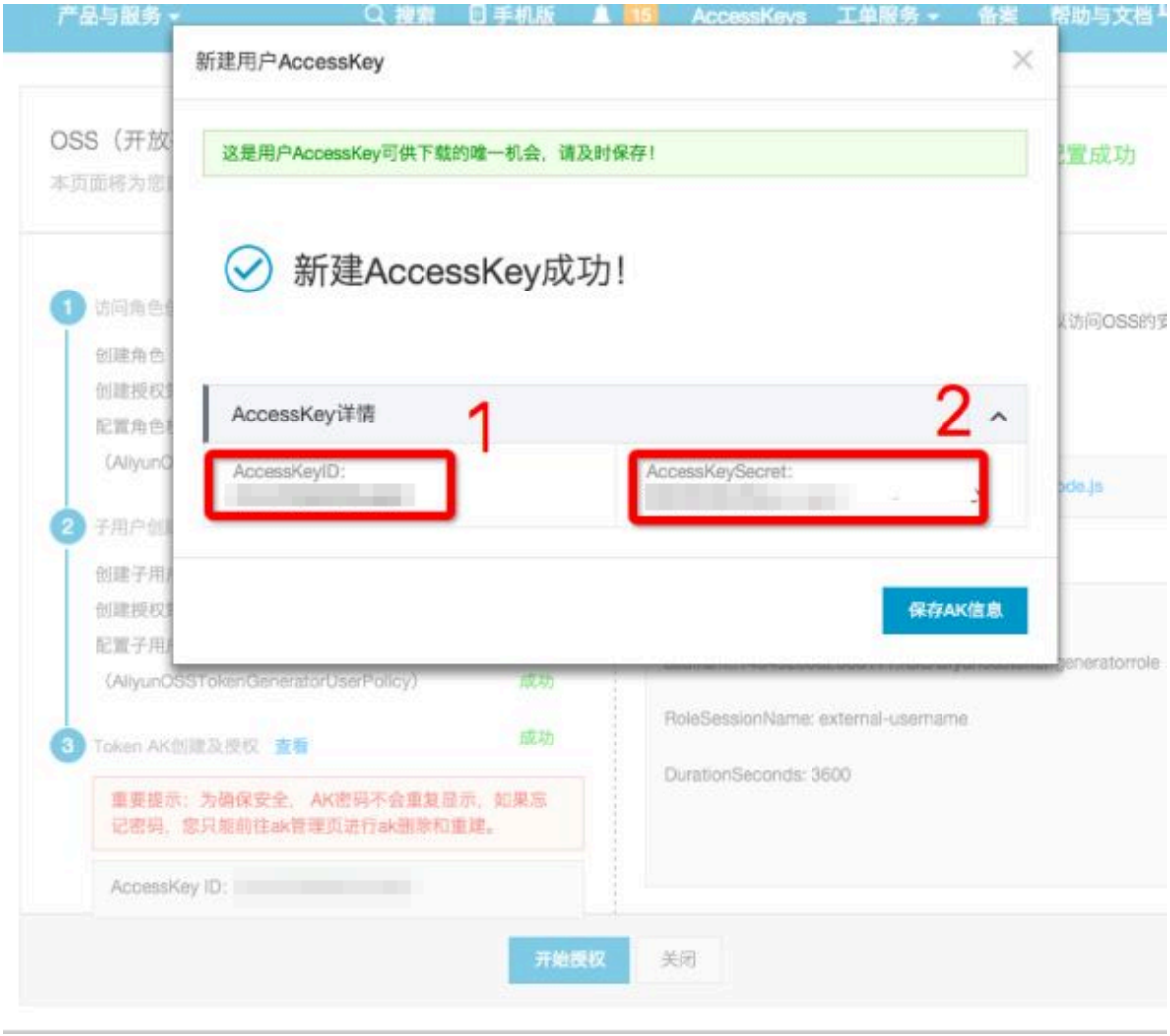
开通STS

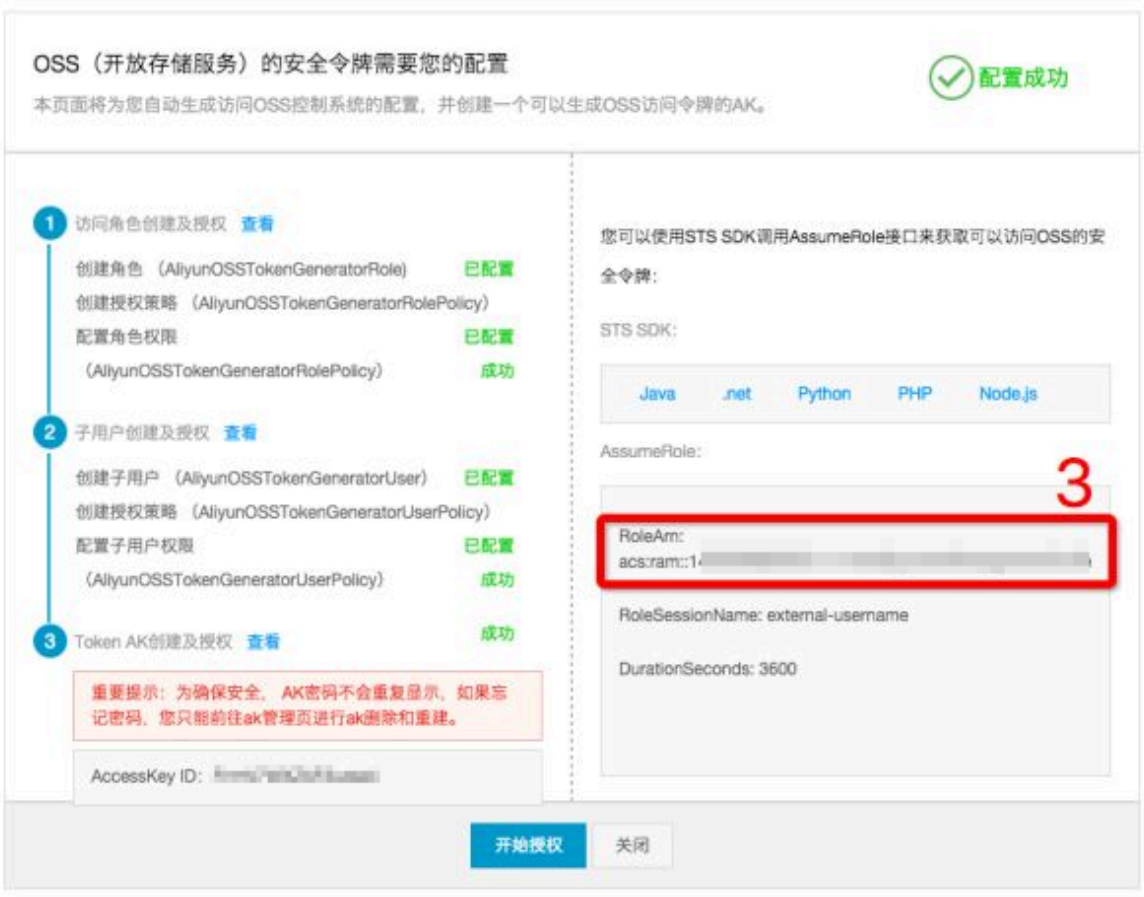
1. 开通OSS服务，创建Bucket，并登录*OSS控制台*。

2. 在OSS概览页中找到基础配置区域，单击 安全令牌，如下图所示：

3. 进入到 安全令牌快捷配置 页面。

4. 系进行自动授权，请务必保存下图中三个红框内的参数。单击 保存AK信息 后，对话框会关闭，STS的开通完成。





搭建一个应用服务器

配置应用服务器示例

为了方便开发，本教程提供了三个语言的版本示例程序供您下载。

- Java：下载地址
- PHP：下载地址
- Ruby：下载地址

每个语言包下载下来后，都会有一个配置文件config.json如下所示：

```
{
"AccessKeyID" : "",
"AccessKeySecret" : "",
"RoleArn" : "",
"TokenExpireTime" : "900",
"PolicyFile": "policy/all_policy.txt"
}
```

说明:

- AccessKeyID：填写上述图标红的参数1的内容。

- AccessKeySecret：填写上述图标红的参数2的内容。
- RoleArn：填写上述图标红的参数3的内容。
- TokenExpireTime：指Android/iOS应用取到这个Token的失效时间。注意：最少是900s，默认值可以不修改。
- PolicyFile：填写的是该Token所要拥有的权限列表的文件，默认值可以不修改。

本文档准备了三种最常用token权限文件，放于policy目录下面。分别是：

- all_policy.txt：指定了该token拥有对该账号下创建Bucket、删除Bucket、上传文件、下载文件、删除文件的权限 。
- bucket_read_policy.txt：指定了该token拥有该账号下对指定Bucket的读权限。
- bucket_read_write_policy.txt：指定了该token拥有该账号下对指定Bucket的读写权限。

如果您想要指定这个Token只能对指定的bucket有读写权限，请把（bucket_read_policy.txt、bucket_read_write_policy.txt）这些文件里面$BUCKET_NAME直接替换成指定的bucket名字。

- 返回的格式解析

```
{
"status":200,
"AccessKeyId":"test",
"AccessKeySecret":"test",
"Security":"CAES+wMIARKAAZhjH0EUOIhJMQBMjRywXq7MQ/cjLYg80Aho
1ek0Jm63XMhr9Oc5s3qaPer8p1YaX1NTDiCFZWFkvlHf1pQhuxfKBc+mRR9KAbHUe
fqH+rdjZqjTF7p2m1wJXP8S6k+G2MpHrUe6TYBkJ43GhhTVFMuM3BZajY3VjZWOXBI
ODRIR1FKZjIiEjMzMzE0MjY0NzM5MTE4NjkxMSoLY2xpZGSSDgSDGAGESGTE
TqOio6c2RrLWRlbW8vKgoUYWNzOm9zczoqOio6c2RrLWRlbW9KEDExNDg5Mz
AxMDcyNDY4MThSBTI2ODQyWg9Bc3N1bWVkUm9sZVVzZXJgAGoSMzMzMTQyNj
Q3MzkxMTg2OTExcglzZGstZGVtbzI=",
"Expiration":"2015-12-12T07:49:09Z",
}
```

说明:

下面四个变量将构成了一个Token：

- status：表示获取Token的状态，获取成功时，返回值是200。
- AccessKeyId：表示Android/iOS应用初始化OSSClient获取的 AccessKeyId。
- AccessKeySecret：表示Android/iOS应用初始化OSSClient获取AccessKeySecret。
- SecurityToken：表示Android/iOS应用初始化的Token。
- Expiration：表示该Token失效的时间。主要在Android SDK会自动判断是否失效，自动获取Token。

- 代码示例的运行方法
  - 对于JAVA版本 (依赖于java 1.7)，将包下载解压后，

    运行方法：java -jar oss-token-server.jar (port)。如果不指定port（端口），直接运行java –jar oss-token-server.jar，程序会监听7080端口。如果想让程序执行在9000端口，运行java –jar app-token-server.jar 9000，其他端口也类似。
  - 对于PHP版本，将包下载解压后，修改config.json这个文件，直接运行php sts.php 即能生成Token，将程序部署到指定的地址。

使用MPS客户端SDK

- 客户端示例代码

  为了方便开发，本教程提供了三个语言的版本示例程序供您下载。
  - H5：下载地址
  - Android：下载地址
  - iOS：下载地址
- SDK核心代码

  JS端

  在使用JS SDK之前，先对要上传视频的OSS Bucket 已经开启了 *CORS*访问权限。下载JS的Demo，在浏览器中打开，在页面上进行参数配置：
  - 配置“HTTP地址”为上面配置的应用服务器地址，如：http://127.0.0.1:7080/。
  - 配置用户Bucket。
  - 配置Bucket的endpoint。
  - 单击选择文件，选中要上传的文件。
  - 单击开始上传按钮。

```
// 初始化客户端
var uploader = new VODUpload({
// 开始上传
'onUploadstarted': function (uploadInfo) {;},
// 文件上传成功
'onUploadSucceed': function (uploadInfo) {console.log("上传成功");},
// 文件上传失败
'onUploadFailed': function (uploadInfo, code, message) {console.log
("上传失败");},
// 文件上传进度，单位：字节
'onUploadProgress': function (uploadInfo, totalSize, uploadedSize) {
console.log("上传进度");},
// 安全令牌超时
'onUploadTokenExpired': function (uploadInfo) {console.log("token超
时");}
});
```

```
// 获取sts信息
result = httpGet(httpServer);
stsToken = JSON.parse(result);
uploader.init(stsToken.AccessKeyId, stsToken.AccessKeySecret,
stsToken.SecurityToken, stsToken.Expiration);
// 添加文件
uploader.addFile(event.target.files[i], endpoint, bucket, object,
userData);
// 开始上传
uploader.startUpload();
```

Android端

确保Android端已经添加如下权限：

```
<uses-permission android:name="android.permission.INTERNET"></uses-
permission>
<uses-permission android:name="android.permission.ACCESS_NET
WORK_STATE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE
"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTE
RNAL_STORAGE"></uses-permission>
```

下载Android端Demo，进行如下修改：

- **修改MainActivity里面的serverUrl为应用服务器配置地址，如：http://192.168.0.2:7080/。**
- **配置用户Bucket。**
- **配置用户Bucket对应的endpoint。**
- **运行Demo，单击添加文件。**
- **单击上传，在OSS对应Bucket的uploadtest/目录下查看文件是否已经上传成功。**

主要代码：

```
VODUploadClient uploader = new VODUploadClientImpl(getApplica
tionContext());
VODUploadCallback callback = new VODUploadCallback() {
@Override
public void onUploadSucceed(UploadFileInfo info) {;}
@Override
public void onUploadFailed(UploadFileInfo info, String code, String
message) {;}
@Override
public void onUploadProgress(UploadFileInfo info, long uploadedSize
, long totalSize) {;}
@Override
public void onUploadTokenExpired(UploadFileInfo info) {
// 获取并更新 sts token。
uploader.resumeWithToken("", "", "", "");
}
@Override
public void onUploadRetry(UploadFileInfo info, String code, String
message) {;}
@Override
```

```
public void onUploadRetryResume(UploadFileInfo info) {;}
@Override
public boolean onUploadStarted(UploadFileInfo uploadFileInfo) {;}
};
// 获取sts token并初始化
uploader.init("", "", "", "", callback);
// 添加文件
uploader.addFile("", "", "", "");
// 开始上传
uploader.start();
```

IOS端

下载iOS端Demo，进行如下修改：

- **修改VODUploadDemo.m里面的serverUrl为应用服务器配置地址，如：http://192.168.0.2:7080/。**
- **配置用户Bucket。**
- **配置用户Bucket对应的endpoint。**
- **运行Demo，单击添加文件。**
- **单击上传，在OSS对应Bucket的uploadtest/目录下查看文件是否已经上传成功。**

主要代码：

```
// 回调初始化
OnUploadStartedListener testUploadStartedCallbackFunc = ^(UploadFile
Info* fileInfo) {;};
OnUploadSucceedListener testSuccessCallbackFunc = ^(NSString*
filePath){;};
OnUploadFailedListener testFailedCallbackFunc = ^(NSString* filePath
, NSString* code, NSString* message){;};
OnUploadProgressListener testProgressCallbackFunc = ^(NSString*
filePath, long uploadedSize, long totalSize) {;};
OnUploadTokenExpiredListener testTokenExpiredCallbackFunc = ^{
// 获取并更新sts token
[uploader resumeWithToken:
accessKeySecret:
secretToken:
expireTime:]
};
OnUploadRertyListener testUploadRertyListener = ^{;};
OnUploadRertyResumeListener testUploadRertyResumeListener = ^{;};
VODUploadListener *listener;
listener = [[VODUploadListener alloc] init];
listener.started = testUploadStartedCallbackFunc;
listener.success = testSuccessCallbackFunc;
listener.failure = testFailedCallbackFunc;
listener.progress = testProgressCallbackFunc;
listener.expire = testTokenExpiredCallbackFunc;
listener.retry = testUploadRertyListener;
listener.retryResume = testUploadRertyResumeListener;
// 获取Token
// 上传客户端初始化
VODUploadClient *uploader;
[uploader init:
accessKeySecret:
```

```
secretToken:
expireTime:
listener:listener];
// 添加文件
[uploader addFile:
endpoint:
bucket:
object:];
// 开始上传
[uploader start];
```

4 如何添加水印

视频水印，指在视频上添加相关信息（如企业logo、电视台台标、用户昵称等），以突出品牌、维护版权、增加产品的识别度。媒体处理支持图片水印、动画水印和文字水印三种水印类型，您可按需选择。

类型

- 图片水印：使用一张PNG图片作为水印，该图片位于视频的某个固定位置，并支持指定展示时间（从片头贯穿到片尾或者仅在某些时间段展示）。
- 动画水印：使用APNG动图或者mov视频作为水印，该动画位于视频的固定位置循环播放。
- 文字水印：使用一段文字作为水印，可设置文字的字体、字号、颜色， 支持每个视频添加不同文字内容。

参数说明

在提交转码任务时（参见如何提交转码作业），可以指定水印模板和水印素材，为输出视频添加水印信息。

每个转码作业可以指定若干个*WaterMark*对象，每个WaterMark又包含很多参数：

- WaterMarkTemplateId（水印模板ID）

  水印模板包含了一些常见参数，例如：Type、ReferPos、Width、Height、Dx、Dy等。

  您可以在媒体处理控制台创建，参见创建水印模板。

  > 说明:
  > WaterMark对象中的对应参数比模板的参数优先级更高，会覆盖模板中配置的对应参数。

- Type（水印类型）

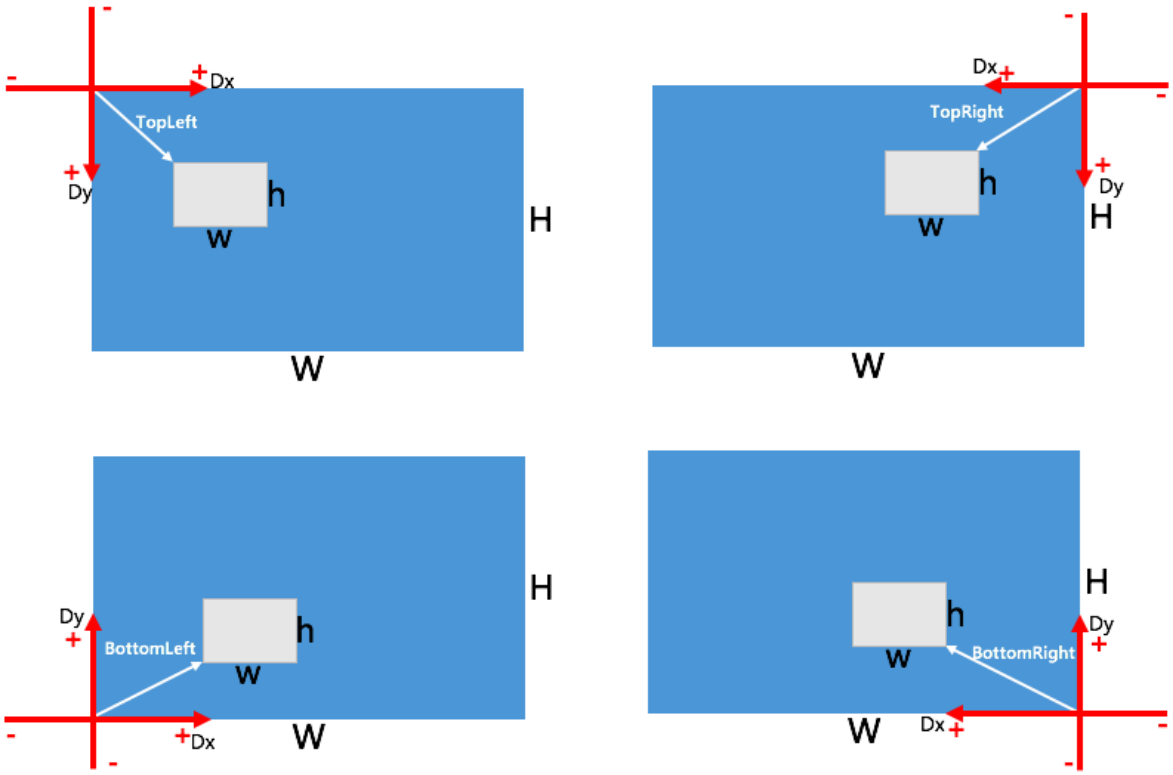
  添加图片水印、动画水印时，Type设置为Image，同时设置InputFile参数，即水印素材的OSS存储路径。

  添加文字水印时，Type设置为Text，同时指定*TextWaterMark*参数，包括字体、字号、颜色、透明度等。

- ReferPos（水印位置）

  水印显示的参考位置，Dx、Dy是相对于参考位置来计算的。参见水印模板配置。

  水印位置坐标说明：



- Width、Height、Dx、Dy

  设置水印的宽度、高度、水平偏移、垂直偏移。支持两种计算方式：

  - 绝对值：

    单位：像素，取值范围：[8，4096]。

  - 相对比例：

    相对输出视频分辨率的宽度、高度。取值范围（0，1），精确到4位小数点，例如：0.9999。

  - 默认值：

    - Dx、Dy不设置时，则默认值为0。
    - 宽、高都不设置时，水印宽的取值为输出视频分辨率宽的0.12倍，水印高的取值按水印原图宽高比例等比缩放。
    - 宽或高的值设置一个，另一个不设置时，则另一个的取值按水印原图宽高比等比缩放。
    - 宽、高的值都设置时，按实际设置值设置水印图片。

- InputFile（输入文件）

  设置图片水印或动画水印的OSS文件地址，图片支持PNG格式，动画支持mov格式和apng格式。

  > 说明:
  > 动画水印的文件扩展名必须是小写mov或者apng，图片不受文件扩展名影响。

- *TextWaterMark*（文字水印）

  设置文字水印的详细参数。

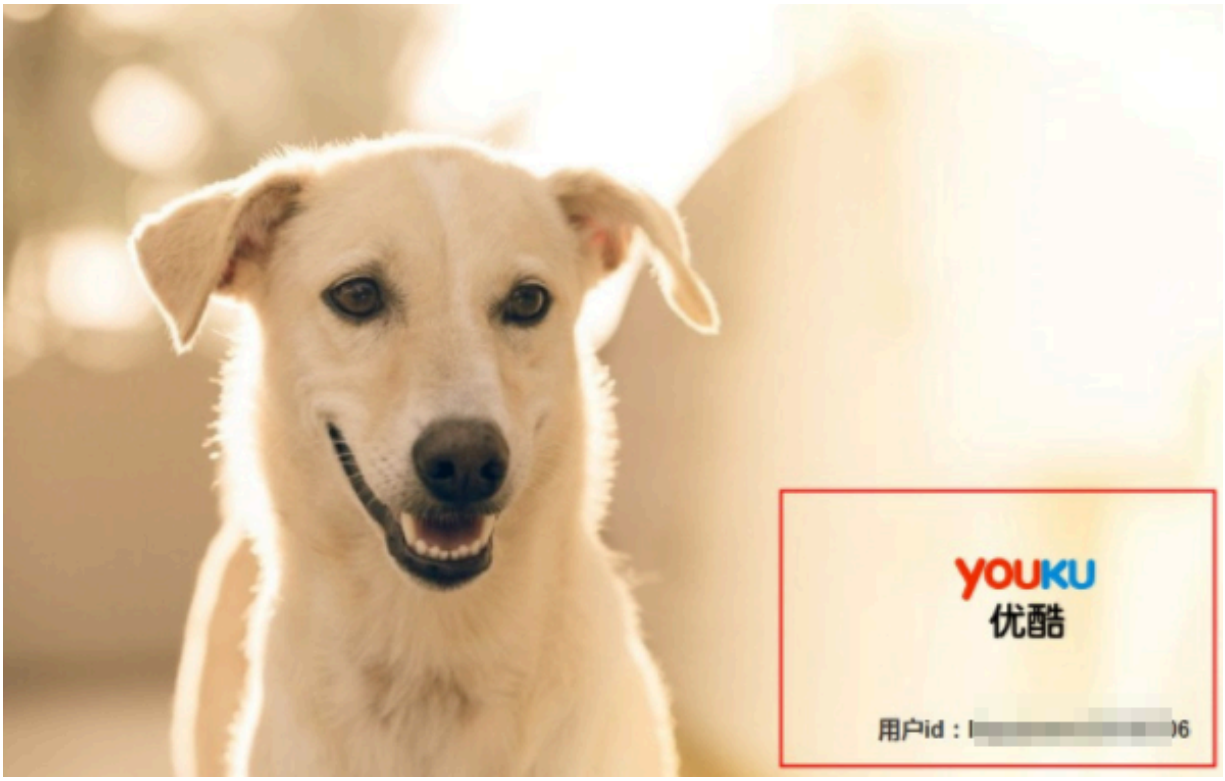
  > 说明:
  > 文字水印暂不支持参考位置和相对比例，只支持以左上角为参考位置，设置Dx、Dy绝对像素值偏移。

使用场景

短视频

短视频场景中，被下载和分享的视频，通常带有一个图片水印（产品logo）和一个文字水印（用户ID），用于保护版权。

示例：



音视频网站

音视频网站，通常会在视频上添加品牌logo，宣示版权归属。同时，在综艺节目中，也会加入贴纸元素，增加趣味性或增加广告展现。

示例：

示例代码

在转码成720P(1280x720)清晰度的MP4视频文件时，同时设置3个水印，并显式覆盖水印参数：

- 图片水印

  以右上角为参考位置，显示一个宽占输出分辨率0.05比例，高度按图片原始比例自适应。

- 文字水印

  以左上角为参考位置，显示内容测试文字水印。字体信息：宋体、大小16、红色，显示的内容按照50%的透明度叠加在视频上。

- 动画水印

  以左下角为参考位置，显示一个高度240像素的mov视频，宽度按照视频水印原始比例自适应。

具体示例代码如下：

- 水印-Java SDK
- 水印-Python SDK
- 水印-PHP SDK

5 如何设置截图

视频截图是截取视频中特定位置的图像，然后保存为图片文件。

类型

- 关键帧

  因为视频编码的特点，关键帧图像的优势是画质好，执行速度快。由于视频中关键帧是间隔一段时间才会出现，所以劣势是时间点不太精确，会在设置的时间点附近寻找相应的关键帧。

- 普通帧

  和关键帧相反，画质稍差，执行速度较慢。优点是可以根据设置的时间点精确截取图像。

参数说明

在输入文件时，您需要关注以下参数：

Input

设置需要截图的视频OSS输入文件。

说明：

OSS的Location必须和媒体处理服务的地域对应。例如，OSS的oss-cn-hangzhou对应媒体处理的cn-hangzhou。

在 截图配置*(SnapshotConfig)* 中，需要关注以下参数：

- *OutputFile*

  设置截图的OSS输出文件。OSS的Object除了设置为固定的文件名外，还支持按照一些规则自定义文件名，参见 截图*Output*详情 。

- Time

  设置单帧截图的时间点，也是多帧截图的开始时间点。整数类型，单位：毫秒。

- Interval、Num

  设置多帧截图的间隔时间（单位：秒）和数量。分以下几种情况：

  - 不设置Num时，表示按照间隔时间，一直截取到视频结尾。
  - Num大于1时，表示按照间隔时间，截取到指定数量的图像时就停止截图。
  - 设置Num=1时，按照异步方式执行单帧截图。

- Width、Height

  设置单帧截图或多帧截图输出的图片宽和高，单位：像素。

  宽和高都是以输入视频为参考。

  - 如果宽和高都不设置时，图片的尺寸和视频相同。
  - 如果只设置宽（或高）时，另一边会按照视频的分辨率保持比例不变，避免图像变形。

  > 说明:
  > 建议您不要同时任意设置宽和高的值，以免引起图像比例失真。
  >
  > - 如果MP4的竖屏视频带有旋转标识，截图是横屏图像。
  > - 如果MP4的竖屏视频不带有旋转标识，则截图保持竖屏图像。

- FrameType

  设置截图的类型：关键帧或普通帧。默认：关键帧。

- TileOutputFile、TileOut

  设置雪碧拼图的OSS输出文件和 配置#TileOut#。

- SubOut、Format

  - 如果您需要使用webVTT格式的缩略图，设置Format=“VTT”。
  - 如果webVTT格式需要以雪碧图的方式显示，要同时设置Format和SubOut的值。

使用场景

- 单帧截图

  设置一个明确的截图时间点，截取对应的视频图像。

- 多帧截图

  按照设置的间隔时间，均匀的截取对应视频的多帧图像，每帧图像都是一个图片文件。也叫批量截图、序列截图。

- 雪碧拼图

  多帧截图的图像以雪碧图的方式拼成一张大图输出。这样可以一次请求获取多帧图像，降低图片请求次数，提高客户端性能。

- WebVTT缩略图

  HTML5标准的字幕文件格式，也被很多H5播放器作为缩略图预览的格式，参见 JWPlayer文档。

  WebVTT只是文件格式，缩略图可以是多张图片，也可以是雪碧图方式拼成的一张大图。

执行方式

参见 作业和管道 中作业执行和结果。

- 同步

  调用API时，同步返回截图作业Id以及截图结果。

  同步方式只支持单帧截图的场景。

- 异步

  调用API时，仅返回截图作业Id。截图结果的查询，可以使用消息通知服务，也可以通过截图作业Id查询。

  单帧截图、多帧截图、雪碧拼图、WebVTT缩略图都支持异步的执行方式。

示例代码

有一个`720P(1280x720)`时长10秒的视频，设置截图高度360像素，从第2秒开始，按照每1秒截取一张图的方式，最多截取3帧。最终会输出3张图片，时间点分别是2、3、4秒。文件名也会按照00001、00002、00003的规则来命名。

截图-*Java*

截图-*Python*

截图-*PHP*

6 拼接剪辑

6.1 如何设置拼接和剪辑

拼接是把多个不同格式 、不同编码、分辨率的视频拼接在一起，输出成一个格式、编码、分辨率相同的新视频。常用于添加固定的片头和片尾、直播录制视频拼接。

剪辑是指裁剪视频的某一段，输出成一个新视频。常用于截取视频中精彩或关键的内容。

参数说明

在视频拼接时，您需要关注以下参数：

Input

设置片头视频的OSS输入文件。

说明：

OSS的Location必须和媒体处理服务的地域对应。例如，OSS的oss-cn-hangzhou对应媒体处理的cn-hangzhou。

在 输出参数中，您需要关注以下参数：

- *Video*

  设置输出最终视频的宽、高、码率等。如果多个拼接视频（包括片头、片尾）的宽、高比和最终输出的不一致，会自动填充黑边。建议您根据不同业务的分辨率实际情况，准备几个不同宽、高比的片头、片尾视频，以达到最好的效果。

- *MergeList*

  列表的顺序代表了拼接顺序，所以列表的最后一个元素是片尾，最多支持5个(包含片头、片尾）视频拼接在一起。如果您需要拼接更多视频，请使用`MergeConfigUrl`参数。

  说明：

`MergeLsit`和`MergeConfigUrl`不支持同时设置，您只能选择其中一个设置。

每个拼接视频都包含3个参数：

- MergeURL

  设置拼接视频的OSS URL地址。

  说明：
  拼接视频的OSS地域必须和片头一致，不支持跨地域视频的拼接。

- Start

  拼接视频时，如果您期望只截取部分内容输出到最终视频，可以设置截取的开始时间点。默认值：0。

- Duration

  拼接视频时，如果您期望只截取部分内容输出到最终视频，可以设置相对于开始时间点（Start）的截取时长。默认从开始时间点（Start）到结尾的全部内容。

- MergeConfigUrl

  设置拼接视频的配置文件的OSS URL地址。文件的内容就是一个JSON对象，和 *MergeList*参数的值完全一样。

  列表的顺序代表了拼接顺序，所以列表的最后一个元素是片尾，最多支持100个（包含片头、片尾）视频拼接在一起。

示例代码

有一个`720P(1280x720)`的正片视频，拼接上片头片尾是`480P(640x480)`的MP4视频，输出分辨率是`1280x720`。所以在播放输出视频时，片头和片尾会出现左右黑边，正片视频显示正常。

具体代码示例如下：

- 拼接和简单剪辑-*Java SDK*
- 拼接和简单剪辑-*Python SDK*
- 拼接和简单剪辑-*PHP SDK*

6.2 如何设置开板和尾板

视频开板和尾板是一种特殊的拼接效果：嵌入在正片视频中，以画中画的方式展示。

参数说明

在视频开板和尾板时，您需要关注以下参数：

Input

设置正片视频的OSS输入文件。

> 说明:
> OSS的Location必须和媒体处理服务的地域对应。例如，OSS的oss-cn-hangzhou对应媒体处理的cn-hangzhou。

在 输出参数 中，您需要关注以下参数：

- *Video*

  设置输出最终视频的宽、高、码率等。如果正片视频的宽、高比和最终输出的不一致，会强制拉伸。建议您只设置宽或高，另外一边会按照正片的原始比例自动调整。

- *Opening*

  开板列表的顺序代表了拼接顺序，最多支持2个开板视频。

  每个开板视频都包含4个参数：

  - OpenUrl

    设置开板视频的OSS URL地址。

    > 说明:
    > 开板视频的OSS地域必须和正片视频一致，不支持跨地域视频的拼接。

  - Start

    相对正片视频的时间戳，从0开始延迟多长时间后，显示开板视频。单位：秒，默认值：0。

  - Width

    指定开板视频的宽。有两种特殊场景：

    ■ `-1`表示等于开板视频原片的宽；

    ■ `full`表示填满画面。

    ■ (0,4096] 范围的其他数字指定具体的宽。

    > 说明:

以正片视频中心点为基准，居中对齐。不要超过正片视频宽，否则效果未知。

- Height

  指定开板视频的高。有两种特殊场景：

  - `-1`表示等于开板视频原片的高，
  - `full`表示填满画面。
  - (0,4096] 范围的其他数字指定指定具体的高。

  说明：

  以正片视频中心点为基准，居中对齐。不要超过正片视频高，否则效果未知。

- *TailSlate*

  尾板列表的顺序代表了拼接顺序，最多支持2个尾板视频。

  每个开板视频都包含以下几个参数：

  - TailUrl

    设置尾板视频的OSS URL地址。

    说明：

    尾板视频的OSS地域必须和正片视频一致，不支持跨地域视频的拼接。

  - Width

    指定尾板视频的宽。有两种特殊场景：

    - `-1`表示等于尾板视频原片的宽，
    - `full`表示填满画面。
    - (0,4096] 范围的其他数字指定具体的宽。

    说明：

以正片视频中心点为基准，居中对齐。不要超过正片视频宽，否则效果未知。

- Height

  指定尾板视频的高。有两种特殊场景：

  ■ `-1`表示等于尾板视频原片的高，

  ■ `full`表示填满画面。

  ■ (0,4096] 范围的其他数字指定具体的高。

  说明：

  以正片视频中心点为基准，居中对齐。不要超过正片视频高，否则效果未知。

- BlendDuration

  正片视频和尾板视频过渡的时长。过渡的效果是淡入淡出：正片显示最后一帧，同时播放尾板视频，正片最后一帧逐步变暗，尾板视频逐步变亮。单位：秒，默认值：0。

- IsMergeAudio

  是否要拼接尾板视频的音频内容。

- BgColor

  如果尾板视频的宽或者高小于正片时，设置空白处填充的背景色。

示例代码

有一个`720P(1280x720)`的正片视频，拼接上开板和尾板是`480P(640x480)`的MP4视频，并且设置开板视频开始时间为2秒，设置尾板视频过渡时间3秒、背景色为黑色`Black`。最后在播放输出视频时，开板视频在正片视频播放到第2秒时，以画中画(居中)的形式和正片视频同时播放，尾板视频在正片结尾时淡入淡出。

具体示例代码如下：

- 开板和尾板-*Java SDK*
- 开板和尾板-*Python SDK*
- 开板和尾板-*PHP SDK*

7 打包

7.1 如何进行HLS打包

简介

HLS打包是指将多字幕、多音轨、多码率视频流生成一个Master Playlist文件的过程。包括两个步骤：新建HLS打包工作流、调用AddMedia接口指定视频及HLS打包工作流ID进行视频的处理。

1. 新建工作流 时，有几个需要关注的对象：

   · Topology

   拓扑结构是指可自定义的业务处理流程，DAG。

   · Activity

   活动是指组成拓扑结构的处理节点，在新建HLS打包工作流时要注意以下几个活动：

   - *PackageConfig*

     指定HLS打包配置，设置Master Playlist文件输出位置。

     前后依赖：

     ■ 前置节点允许：Start。
     ■ 后置节点允许：SubtitleGroup、AudioGroup、Transcode（仅视频)。

   - *SubtitleGroup*

     指定字幕分组ID。

     前后依赖：

     ■ 前置节点允许：PackageConfig。
     ■ 后置节点允许：Transcode（仅字幕）。

   - *AudioGroup*

     指定音频分组ID。

     前后依赖：

     ■ 前置节点允许：PackageConfig。
     ■ 后置节点允许：Transcode （仅音频）。

   - *Transcode*

     用于提取视频流、音频流、字幕流。

     前后依赖：

     ■ 前置节点允许：PackageConfig、SubtitleGroup、AudioGroup。
     ■ 后置节点允许：GenerateMasterPlayList。

   - *GenerateMasterPlayList*

     HLS打包生成活动，指定视频多码率配置，指定音频，字幕分组。

     前后依赖：

■ 前置节点允许：Transcode。

■ 后置节点允许：Report。

· Dependencies

依赖关系是拓扑结构中的边，指明活动之间的依赖。

2. 调用 新增媒体，需要注意以下几点：

· 指定媒体工作流ID。

· 若存在字幕提取，可以设置字幕文件地址覆盖Transcode活动中的参数WebVTTSubtitleURL，只支持WebVTT的字幕文件。

· 工作流触发模式设置为：NotInAuto。

场景

源文件mxf格式（也可其它格式如mp4/flv/m3u8(ts)），从源文件中提取3路音轨，提取2路视频流。提取2路WebVTT字幕，最终组合打包成一个Master Playlist：

设置HLS打包输出Master Playlist的位置及名称。

· 设置Bucket。

· 设置Location。

· 设置Master Playlist的名称。

· 活动定义如下：

```
{
"Parameters" : {
"Output" : "{\"Bucket\": \"processedmediafile\",\"Location\": \"oss-cn-hangzhou\",\"MasterPlayListName\": \"{MediaId}/{RunId}/hls/master.m3u8\"}"
},
"Type" : "PackageConfig"
}
```

- Output设置Master Playlist的存储位置及名称，参见 *PackageConfig*活动支持的参数。

- Type指定活动类型为PackageConfig。

音频分组。

· 设置音频分组ID，两个音频流同属于一个音频分组。

· 活动定义：

```
{
"Parameters" : {
"GroupId" : "audios"
},
```

```
"Type" : "AudioGroup"
}
```

- GroupId：指定音频分组Id为audios。
- Type：类型为AudioGroup活动。

提取音轨。

- 从mxf源文件中提取音频流，需要去掉视频流。
- 输出的音频流参数：
  - Codec：AAC
  - SampleRate：48000 Hz
  - Format：Stereo
- 活动定义：

```
{
"Name" : "audio-extract-1",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"S00000001-100020\",\"AudioStreamMap
\":\"0:a:0\",\"Video\":{\"Remove\":\"true\"}}]",
"ExtXMedia" : "{\"URI\": \"sd/audio-en.m3u8\",\"Name\": \"audio-en
\",\"Language\": \"en-US\"}"
}
```

  - 预置静态模板 ID：S00000001-100020表示音频输出为m3u8(ts)，预置模板内设置的音频码率为80kbps。
  - AudioStreamMap：音频流选择字，参见 *Output*详情中的说明。
  - 在输出中移除掉视频流，参见 *Video*详情。
  - *ExtXMedia*定义Media Playlist，URI指定Media Playlist的名称。
  - Type设置为Transcode，即转码活动。

提取视频。

- 从mxf源文件中提取视频流，需要去掉音频流。
- 活动定义：

```
{
"Name" : "video-extract",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"1fe5393bdb7b2b883f0a0fc91e81344a\",
\"Audio\":{\"Remove\":\"true\"}}]",
"MultiBitrateVideoStream" : "{\"URI\": \"sd/video1.m3u8\"}"
},
"Type" : "Transcode"
```

```
}
```

- 自定义转码模板ID：1fe5393bdb7b2b883f0a0fc91e81344a，可登录MPS控制台，在 全局设置 > 转码模板 中创建自定义模板，设置视频转码参数：
  - Codec：H.264
  - Resolution：384x216
  - Profile：Main
  - Bitrate：240 Kbps
  - Fps：25
  - PixelFormat：YUV420P Max GOP size：1 segment length (4 seconds)
  - 输出格式：m3u8
- 在输出中移除掉音频流，参见 *Audio*详情。
- *MultiBitrateVideoStream*定义Master Playlist中的多码率视频流，URI指定Media Playlist的名称。
- Type设置为Transcode，即转码活动。

字幕分组。

- 设置字幕分组ID，两个字幕流同属于一个字幕分组。
- 活动定义：

```
{
"Parameters" : {
"GroupId" : "subtitles"
},
"Type" : "SubtitleGroup"
}
```

- GroupId：指定音频分组Id为subtitles。
- Type：类型为SubtitleGroup活动。

提取字幕。

- 上传WebVtt格式的字幕到OSS中。
- 活动定义：

```
{
"Name" : "subtitle-extract-1",
"Parameters" : {
"WebVTTSubtitleURL" : "http://mts-video.oss-cn-hangzhou.aliyun-inc.
com/ShawshankRedemption.vtt",
"ExtXMedia" : "{\"URI\": \"zh/subtitle1-cn.m3u8\",\"Name\": \"
subtitle-cn\",\"Language\": \"cn\"}"
},
```

```
"Type" : "Transcode"
}
```

- *WebVTTSubtitleURL*指定字幕地址，字幕地址在调用 *AddMedia*时可以被动态覆盖，见参数OverrideParams。
- *ExtXMedia*定义Media Playlist，URI指定Media Playlist的名称。
- Type设置为Transcode，即转码活动。

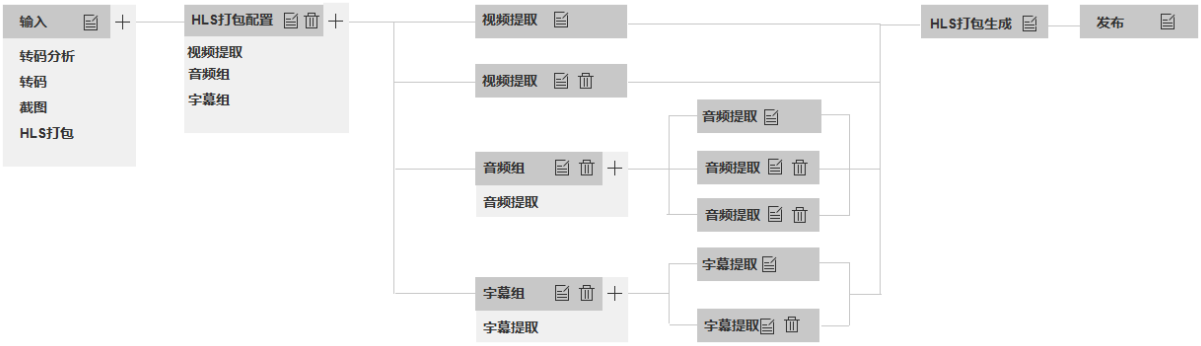
输出Master Playlist。

- 通过提取音频、视频、字幕，将所有提取转换后的资源打包成一个Master Playlist。
- 活动定义：

```
{
"Parameters" : {
"MasterPlayList" : "{\"MultiBitrateVideoStreams\": [{\"RefActivit
yName\": \"video-extract\",\"ExtXStreamInfo\": {\"BandWidth\": \"
1110000\",\"Audio\": \"audios\",\"Subtitles\": \"subtitles\"}}]}"
},
"Type" : "GenerateMasterPlayList"
}
```

- *MasterPlayList*定义Master Playlist。
- *MultiBitrateVideoStreams*多码率视频流数组。
- RefActivityName指定提取视频流的活动名称。
- *ExtXStreamInfo*定义多码率视频流的属性，Audio指定音频分组，Subtitles指定字幕分组
- Type设置为GenerateMasterPlayList，即生成Master Playlist活动。

拓扑示意图：



完整的场景示例用拓扑结构表示：

```
{
"Activities" : {
"package-node" : {
"Name" : "package-node",
"Parameters" : {
```

```
"Output" : "{\"Bucket\": \"processedmediafile\",\"Location\": \"oss-
cn-hangzhou\",\"MasterPlayListName\": \"{MediaId}/{RunId}/hls/master.
m3u8\"}"
},
"Type" : "PackageConfig"
},
"audioGroupNode" : {
"Name" : "audioGroupNode",
"Parameters" : {
"GroupId" : "audios"
},
"Type" : "AudioGroup"
},
"subtitleGroupNode" : {
"Name" : "subtitleGroupNode",
"Parameters" : {
"GroupId" : "subtitles"
},
"Type" : "SubtitleGroup"
},
"video-extract-1" : {
"Name" : "video-extract-1",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"1fe5393bdb7b2b883f0a0fc91e81344a\",\"
Audio\":{\"Remove\":\"true\"}}]",
"MultiBitrateVideoStream" : "{\"URI\": \"sd/video1.m3u8\"}"
},
"Type" : "Transcode"
},
"video-extract-2" : {
"Name" : "video-extract-1",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"1fe5393bdb7b2b883f0a0fc91e81344b\",\"
Audio\":{\"Remove\":\"true\"}}]",
"MultiBitrateVideoStream" : "{\"URI\": \"sd/video2.m3u8\"}"
},
"Type" : "Transcode"
},
"audio-extract-1" : {
"Name" : "audio-extract-1",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"S00000001-100020\",\"AudioStreamMap\":
\"0:a:0\"}]",
"ExtXMedia" : "{\"URI\": \"sd/audio-en-1.m3u8\",\"Name\": \"audio-en
\",\"Language\": \"en-US\"}"
},
"Type" : "Transcode"
},
"audio-extract-2" : {
"Name" : "audio-extract-2",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"S00000001-100020\",\"AudioStreamMap\":
\"0:a:1\"}]",
"ExtXMedia" : "{\"URI\": \"sd/audio-cn.m3u8\",\"Name\": \"audio-cn\",
\"Language\": \"cn\"}"
},
"Type" : "Transcode"
},
"audio-extract-3" : {
"Name" : "audio-extract-3",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"S00000001-100020\",\"AudioStreamMap\":
\"0:a:2\"}]",
```

```
"ExtXMedia" : "{\"URI\": \"sd/audio-de.m3u8\",\"Name\": \"audio-de\",
\"Language\": \"de\"}"
},
"Type" : "Transcode"
},
"subtitle-extract-1" : {
"Name" : "subtitle-extract-1",
"Parameters" : {
"WebVTTSubtitleURL" : "http://mts-video-daily-bucket.oss-test.aliyun-
inc.com/1.vtt",
"ExtXMedia" : "{\"URI\": \"zh/subtitle1-cn.m3u8\",\"Name\": \"subtitle
-cn\",\"Language\": \"cn\"}"
},
"Type" : "Transcode"
},
"subtitle-extract-2" : {
"Name" : "subtitle-extract-2",
"Parameters" : {
"WebVTTSubtitleURL" : "http://mts-video.oss-cn-hangzhou.aliyun-inc.com
/ShawshankRedemption.vtt",
"ExtXMedia" : "{\"URI\": \"zh/subtitle1-en.m3u8\",\"Name\": \"subtitle
-en\",\"Language\": \"en-US\"}"
},
"Type" : "Transcode"
},
"masterPlayListGenerate" : {
"Name" : "masterPlayListGenerate",
"Parameters" : {
"MasterPlayList" : "{\"MultiBitrateVideoStreams\": [{\"RefActivityName
\": \"video-extract-1\",\"ExtXStreamInfo\": {\"BandWidth\": \"1110000
\",\"Audio\": \"audios\",\"Subtitles\": \"subtitles\"}}, {\"RefActivit
yName\": \"video-extract-2\",\"ExtXStreamInfo\": {\"BandWidth\": \"
5000000\",\"Audio\": \"audios\",\"Subtitles\":\"subtitles\"}}]}"
},
"Type" : "GenerateMasterPlayList"
},
"activityEnd" : {
"Name" : "activityEnd",
"Parameters" : {
"PublishType" : "Manual"
},
"Type" : "Report"
},
"activityStart" : {
"Name" : "activityStart",
"Parameters" : {
"PipelineId" : "900ededca77641ecbecd4f44cc3a2965",
"Role" : "AliyunMTSDefaultRole",
"InputFile" : "{\"Bucket\":\"videouploaded\",\"Location\":\"oss-cn-
hangzhou\",\"ObjectPrefix\":\"uploaded/\"}"
},
"Type" : "Start"
}
},
"Dependencies" : {
"video-extract-1" : [ "masterPlayListGenerate" ],
"video-extract-2" : [ "masterPlayListGenerate" ],
"audio-extract-1" : [ "masterPlayListGenerate" ],
"audio-extract-2" : [ "masterPlayListGenerate" ],
"audio-extract-3" : [ "masterPlayListGenerate" ],
"subtitle-extract-1" : [ "masterPlayListGenerate" ],
"subtitle-extract-2" : [ "masterPlayListGenerate" ],
"package-node" : [ "video-extract-1", "video-extract-2","subtitleGr
oupNode", "audioGroupNode" ],
```

```
"audioGroupNode" : [ "audio-extract-1", "audio-extract-2","audio-
extract-3"],
"subtitleGroupNode" : [ "subtitle-extract-1", "subtitle-extract-2" ],
"masterPlayListGenerate" : [ "activityEnd" ],
"activityEnd" : [ ],
"activityStart" : [ "package-node" ]
}
}
```

示例代码

1. 新建HLS打包工作流

   新建工作流-*Java*

   新建工作流-*Python*

   新建工作流-*PHP*

2. 新增媒体

   新增媒体-*Java*

   新增媒体-*Python*

   新增媒体-*PHP*

7.2 如何进行DASH打包

简介

DASH打包是指将多字幕、多音轨、多码率视频流生成一个Master Playlist文件的过程。包括两个步骤：新建DASH打包工作流、调用AddMedia接口指定视频及DASH打包工作流ID进行视频的处理。

1. 新建工作流 时，有几个需要关注的对象：

   · Topology

   拓扑结构是指可自定义的业务处理流程，DAG。

   · Activity

   活动是指组成拓扑结构的处理节点，在新建DASH打包工作流时要注意以下几个活动：

   - *PackageConfig*

     指定DASH打包配置，设置Master Playlist文件输出位置。

     前后依赖：

     ■ 前置节点允许：Start。

     ■ 后置节点允许：SubtitleGroup、AudioGroup、VideoGroup。

   - *SubtitleGroup*

     指定字幕分组ID和语言。

     前后依赖：

     ■ 前置节点允许：PackageConfig。

     ■ 后置节点允许：Transcode（仅字幕）。

   - *AudioGroup*

     指定音频分组ID和语言。

     前后依赖：

     ■ 前置节点允许：PackageConfig。

     ■ 后置节点允许：Transcode （仅音频）。

   - *VideoGroup*

     指定视频分组ID。

     前后依赖：

     ■ 前置节点允许：PackageConfig。

     ■ 后置节点允许：Transcode （仅视频）。

   - *Transcode*

     用于提取视频流、音频流、字幕流。

     前后依赖：

■ 前置节点允许：SubtitleGroup、AudioGroup、VideoGroup。

■ 后置节点允许：GenerateMasterPlayList。

- *GenerateMasterPlayList*

  打包生成活动。

  前后依赖：

  ■ 前置节点允许：Transcode。

  ■ 后置节点允许：Report。

· Dependencies

依赖关系是拓扑结构中的边，指明活动之间的依赖。

2. 调用 新增媒体，需要注意以下几点：

· 指定媒体工作流ID。

· 若存在字幕提取，可以设置参数OverrideParams，以覆盖字幕Transcode活动中的固定字幕文件地址参数，如`{"subtitleTransNode":{"InputConfig":{"Format":"stl","InputFile":{"URL":"http://subtitleBucket.oss-cn-hangzhou.aliyuncs.com/package/subtitle/CENG.stl"}}}}` 其中 sutitleTransNode为工作流定义中的字幕抽取结点

· 工作流触发模式设置为：NotInAuto。

场景

源文件mxf格式（也可是其它格式如mp4、flv、m3u8(ts)），从源文件中提取3路音轨，提取2路视频流。提取2路WebVTT字幕，最终组合打包成一个Master Playlist：

设置DASH打包输出Master Playlist的位置及名称。

· 设置Bucket。

· 设置Location。

· 设置Master Playlist的名称。

· 活动定义如下：

```
{
"Parameters" : {
"Output" : "{\"Bucket\": \"processedmediafile\",\"Location\": \"oss
-cn-hangzhou\",\"MasterPlayListName\": \"{MediaId}/{RunId}/dash/
master.mpd\"}"
},
"Type" : "PackageConfig"
```

```
}
```

- Output设置Master Playlist的存储位置及名称，参见 *PackageConfig*活动支持的参数。
- Type指定活动类型为PackageConfig。

音频分组。

· 活动定义：

```
"audio-cn-group" : {
"Name" : "audio-cn-group",
"Parameters" : {
"AdaptationSet" : "{\"Lang\":\"chinese\",\"Group\":\"AudioGroup
Chinese\"}"
},
"Type" : "AudioGroup"
}
```

- Group：指定音频分组Id为AudioGroupChinese。
- Type：类型为AudioGroup活动。

提取音轨。

· 从mxf源文件中提取音频流，需要去掉视频流。

· 输出的音频流参数：

· 活动定义：

```
"audioCNTransNode" : {
"Name" : "audioCNTransNode",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"d053297fc44f9DashTemplateId\",\"
AudioStreamMap\":\"0:a:0\",\"Video\":{\"Remove\":\"true\"}}]",
"Representation" : "{\"Id\":\"chinese128k\",\"URI\":\"audiocn/cn-abc
.mpd\"}"
},
"Type" : "Transcode"
}
```

- URI：音频流提取后的输出地址。
- AudioStreamMap：音频流选择字，参见 *Output*详情中的说明。
- 在输出中移除掉视频流，参见 *Video*详情。
- Type设置为Transcode，即转码活动。

视频分组。

```
"video-group" : {
"Name" : "video-group",
"Parameters" : {
"AdaptationSet" : "{\"Group\":\"VideoGroup\"}"
```

```
},
"Type" : "VideoGroup"
}
```

提取视频。

- **从mxf源文件中提取视频流，需要去掉音频流。**
- **活动定义：**

```
"videoTransSD" : {
"Name" : "videoTransSD",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"d861b90f6c0aed8f81095e5c5b857cba\",
\"Audio\":{\"Remove\":\"true\"}}]",
"Representation" : "{\"Id\":\"476pSD\",\"URI\":\"videoSD/xx.mpd\"}"
},
"Type" : "Transcode"
}
```

  - **自定义转码模板ID：d861b90f6c0aed8f81095e5c5b857cba，可调用接口进行创建，容器格式为mpd**
  - **在输出中移除掉音频流，参见** *Audio*详情。
  - **URI: 视频流提取后的名称及存储地址。**
  - **Type设置为Transcode，即转码活动。**

字幕分组。

- **设置字幕分组ID**
- **活动定义：**

```
"subtitle-cn-group" : {
"Name" : "subtitle-cn-group",
"Parameters" : {
"AdaptationSet" : "{\"Lang\":\"Chinese\", \"Group\":\"SubtitleEN
Group\"}"
},
"Type" : "SubtitleGroup"
}
```

  - **Group：指定音频分组名称为SubtitleENGroup。**
  - **Lang: 指定此字幕组的语言**
  - **Type：类型为SubtitleGroup活动。**

提取字幕。

- **上传STL、TTML、WebVtt格式的字幕到OSS中。**
- **活动定义：**

```
"subtitleCNNode" : {
"Name" : "subtitleCNNode",
"Parameters" : {
"InputConfig" : "{\"Format\":\"vtt\",\"InputFile\":{\"URL\":\"http
://bucketname.oss-cn-hangzhou.aliyuncs.com/test/Audio-SiHD.chs.vtt
\"}}",
"Representation" : "{\"Id\":\"subtitle-chinese\", \"URI\":\"subtitle
/cn-xx.vtt\"}"
},
"Type" : "Transcode"
}
```

- InputConfig 指定字幕地址，字幕地址在调用 *AddMedia*时可以被动态覆盖，见参数OverrideParams。
- URI: 字幕流提取后的名称及输出目录。
- Type设置为Transcode，即转码活动。

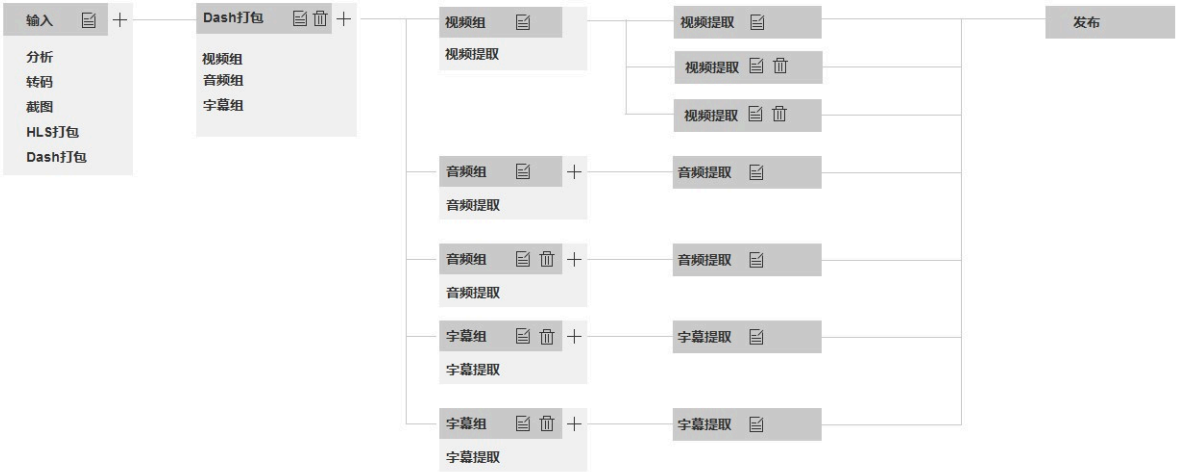
输出Master Playlist。

- 通过提取音频、视频、字幕，将所有提取转换后的资源打包成一个Master Playlist。
- 活动定义：

```
{
"Parameters" : {
},
"Type" : "GenerateMasterPlayList"
}
```

- Type设置为GenerateMasterPlayList，即生成Master Playlist活动。

拓扑图示意：



完整的场景示例用拓扑结构表示：

```
{
"Activities": {
"act-package": {
```

```
"Name": "act-package",
"Parameters": {
"Output": "{\"Bucket\": \"outputbucketname\",\"Location\": \"oss-cn
-hangzhou\",\"MasterPlayListName\": \"dashpackage/{MediaId}/{RunId}/
master.mpd\"}",
"Protocol": "dash"
},
"Type": "PackageConfig"
},
"video-group": {
"Name": "video-group",
"Parameters": {
"AdaptationSet": "{\"Group\":\"VideoGroup\"}"
},
"Type": "VideoGroup"
},
"audio-en-group": {
"Name": "audio-en-group",
"Parameters": {
"AdaptationSet": "{\"Lang\":\"english\", \"Group\":\"AudioGroupEnglish
\"}"
},
"Type": "AudioGroup"
},
"audio-cn-group": {
"Name": "audio-cn-group",
"Parameters": {
"AdaptationSet": "{\"Lang\":\"chinese\", \"Group\":\"AudioGroupChinese
\"}"
},
"Type": "AudioGroup"
},
"subtitle-en-group": {
"Name": "subtitle-en-group",
"Parameters": {
"AdaptationSet": "{\"Lang\":\"english\", \"Group\":\"SubtitleENGroup
\"}"
},
"Type": "SubtitleGroup"
},
"subtitle-cn-group": {
"Name": "subtitle-cn-group",
"Parameters": {
"AdaptationSet": "{\"Lang\":\"chinese\", \"Group\":\"SubtitleCNGroup
\"}"
},
"Type": "SubtitleGroup"
},
"videoTransLD": {
"Name": "videoTransLD",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d053297fc44f9dd6becd4a98d1c42f50\",\"
Audio\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"270pLD\", \"URI\":\"videoLD/xx.mpd\"}"
},
"Type": "Transcode"
},
"videoTransSD": {
"Name": "videoTransSD",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d861b90f6c0aed8f81095e5c5b857cba\",\"
Audio\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"480pSD\", \"URI\":\"videoSD/xx.mpd\"}"
},
```

```
"Type": "Transcode"
},
"videoTransHD": {
"Name": "videoTransHD",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"117b3ae88efbc97df372cfd9a0e1ff4c\",\"
Audio\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"720pHD\", \"URI\":\"videoHD/xx.mpd\"}"
},
"Type": "Transcode"
},
"audioCNTransNode": {
"Name": "audioCNTransNode",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d053297fc44f9dd6becd4a98d1c42f50\",\"
AudioStreamMap\":\"0:a:0\",\"Video\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"chinese128k\", \"URI\":\"audiocn/cn-abc.
mpd\"}"
},
"Type": "Transcode"
},
"audioENTransNode": {
"Name": "audioENTransNode",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d053297fc44f9dd6becd4a98d1c42f50\",\"
AudioStreamMap\":\"0:a:1\",\"Video\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"english128k\", \"URI\":\"audioen/en-abc.
mpd\"}"
},
"Type": "Transcode"
},
"subtitleENNode": {
"Name": "subtitleENNode",
"Parameters": {
"InputConfig": "{\"Format\":\"vtt\",\"InputFile\":{\"URL\":\"http://
bucketname.oss-cn-hangzhou.aliyuncs.com/dashpackage/subtitle/Subtitle.
EN.vtt\"}}",
"Representation": "{\"Id\":\"subtitle-english\", \"URI\":\"subtitle/en
-xx.vtt\"}"
},
"Type": "Transcode"
},
"subtitleCNNode": {
"Name": "subtitleCNNode",
"Parameters": {
"InputConfig": "{\"Format\":\"vtt\",\"InputFile\":{\"URL\":\"http://
bucketname.oss-cn-hangzhou.aliyuncs.com/dashpackage/subtitle/Subtitle.
CN.vtt\"}}",
"Representation": "{\"Id\":\"subtitle-chinese\", \"URI\":\"subtitle/cn
-xx.vtt\"}"
},
"Type": "Transcode"
},
"act-report": {
"Name": "act-report",
"Parameters": {
"PublishType": "Auto"
},
"Type": "Report"
},
"act-start": {
"Name": "act-start",
"Parameters": {
"PipelineId": "cc7fcef2562e4abc9332d491f93399d2",
```

```
"InputFile": "{\"Bucket\":\"inputbucketname\",\"Location\":\"oss-cn-
hangzhou\",\"ObjectPrefix\":\"package/dash/\"}"
},
"Type": "Start"
},
"generateMasterPlayListAct": {
"Name": "generateMasterPlayListAct",
"Parameters": {},
"Type": "GenerateMasterPlayList"
}
},
"Dependencies": {
"audio-en-group": ["audioENTransNode"],
"video-group": ["videoTransLD", "videoTransSD", "videoTransHD"],
"audio-cn-group": ["audioCNTransNode"],
"audioCNTransNode": ["generateMasterPlayListAct"],
"subtitleENNode": ["generateMasterPlayListAct"],
"act-package": ["audio-en-group", "audio-cn-group", "subtitle-cn-group
", "subtitle-en-group", "video-group"],
"act-report": [],
"videoTransSD": ["generateMasterPlayListAct"],
"videoTransHD": ["generateMasterPlayListAct"],
"subtitle-en-group": ["subtitleENNode"],
"subtitle-cn-group": ["subtitleCNNode"],
"subtitleCNNode": ["generateMasterPlayListAct"],
"act-start": ["act-package"],
"videoTransLD": ["generateMasterPlayListAct"],
"generateMasterPlayListAct": ["act-report"],
"audioENTransNode": ["generateMasterPlayListAct"]
}
}
```

示例代码

1. **新建HLS打包工作流。**

   新建工作流-*Java*

   新建工作流-*Python*

   新建工作流-*PHP*

2. **新增媒体。**

   新增媒体-*Java*

   新增媒体-*Python*

   新增媒体-*PHP*

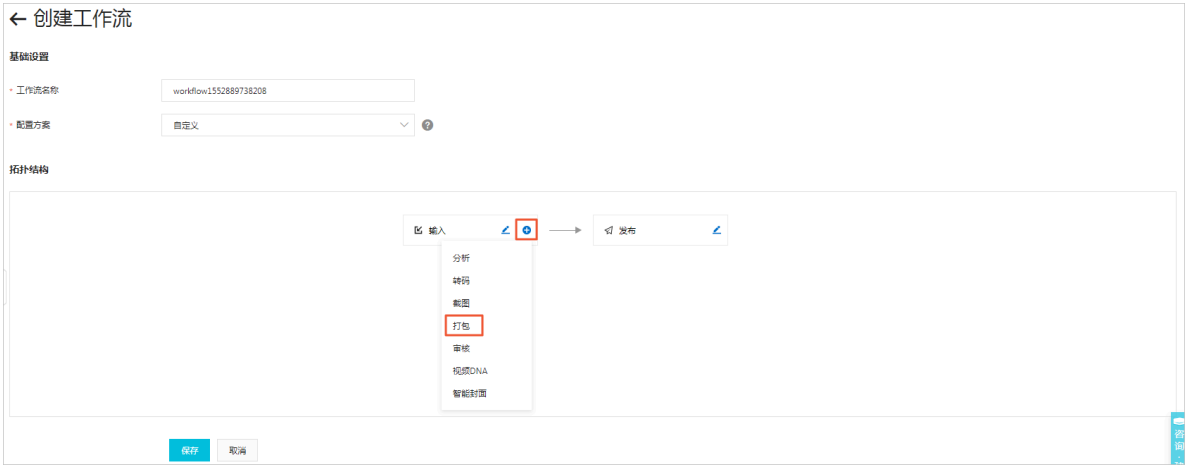
7.3 如何创建HLS打包工作流

场景

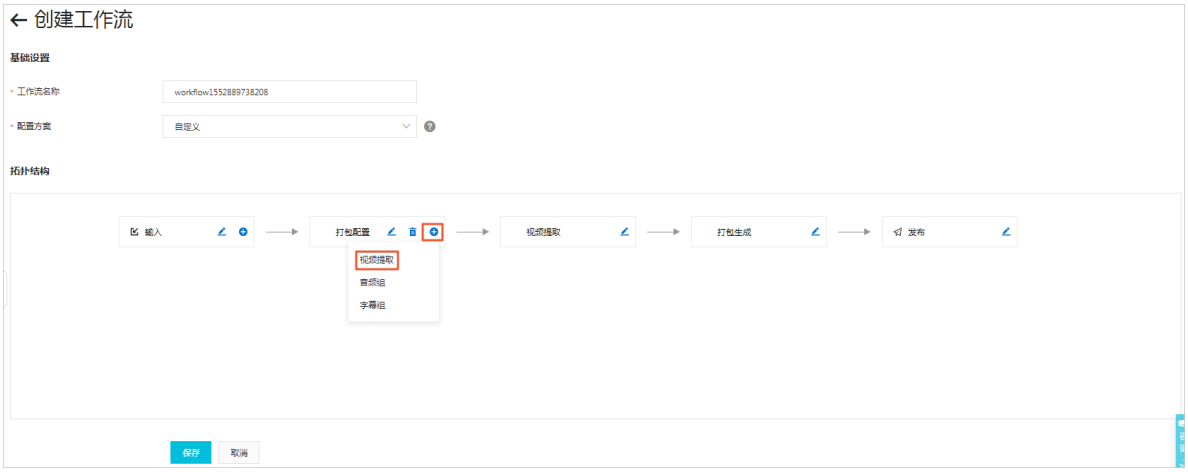
把480P、720P、1080P的视频流打包为1个输出文件，以便您根据网络带宽情况切换最适宜的视频流。

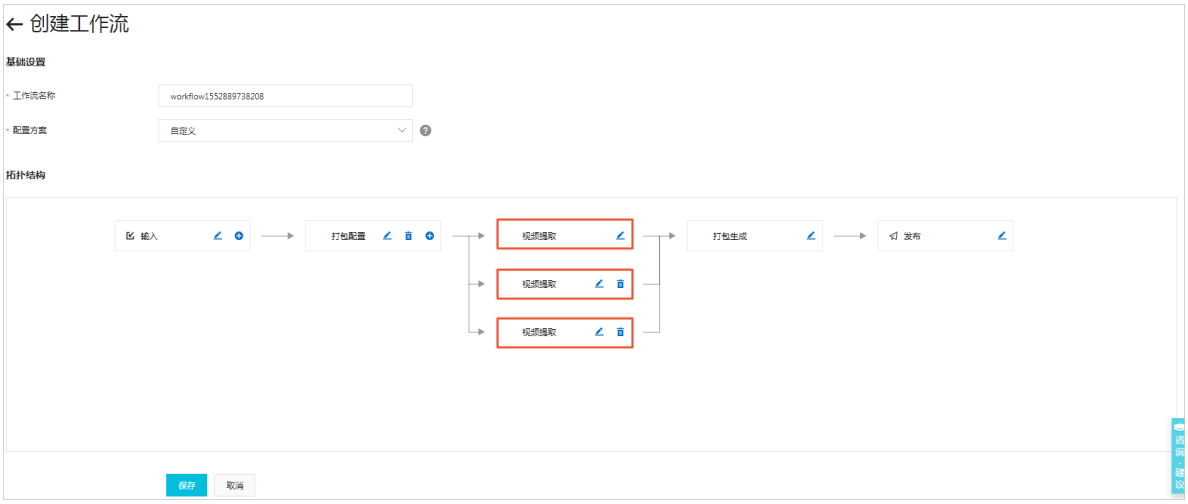
操作步骤

1. 登录 媒体处理控制台。
2. 选择所需的地域。
3. 单击 媒体管理 > 媒体库设置。
4. 单击 工作流 > 创建工作流。
5. 单击 输入 节点右侧的 图标，添加打包节点。



6. 单击 打包配置 节点右侧的  图标，添加3个 视频提取 节点。





7. 配置 输入 节点。

a. 单击 输入 节点右侧的 图标进行配置。

b. 在 输入 页面，单击 输入路径 右侧的 选择。

说明:

输入路径 是OSS上的一个存储位置，应为OSS上真实存在的路径。

c. 在 OSS文件管理 中，选择bucket名称，并单击 确定。

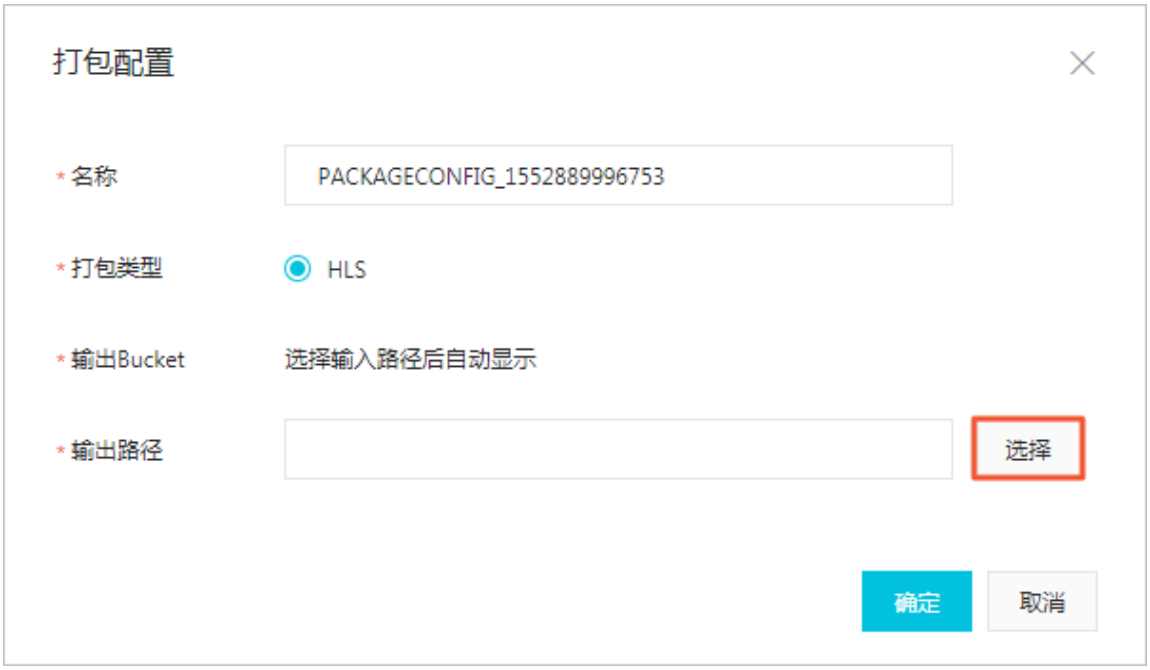
d. 消息类别：可选，您可以选择 队列 或 通知，并设定队列或消息的实例。

8. 配置 打包配置 节点。

a. 修改 名称，或者保持系统默认名称。

b. 单击 打包配置节点右侧的  图标进行配置。

c. 在 打包配置 页面，单击 输出路径 右侧的 选择。



 说明:

输出路径 是OSS上的一个存储位置及输出文件名。为避免媒体工作流多次执行时覆盖输出文件，您可以组合使用系统内置的UC变量参数：

- {RunId}：媒体工作流执行ID，
- {ObjectPrefix}：不含Bucket信息的原文件路径，
- {FileName}：不含扩展名的原文件名，

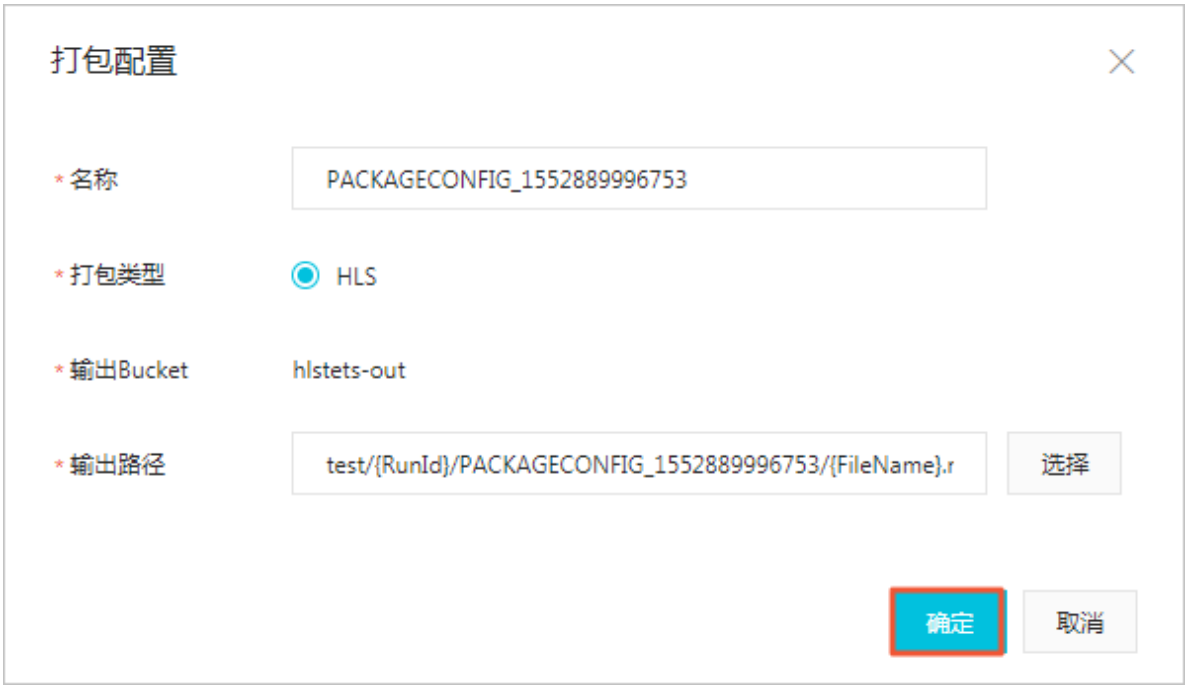
· {ExtName}：原文件扩展名。

d. 在 OSS文件管理 中，选择bucket名称，并单击 确定。

说明:

输出Bucket不可与输入Bucket为同一个Bucket。

打包配置 节点配置完成。

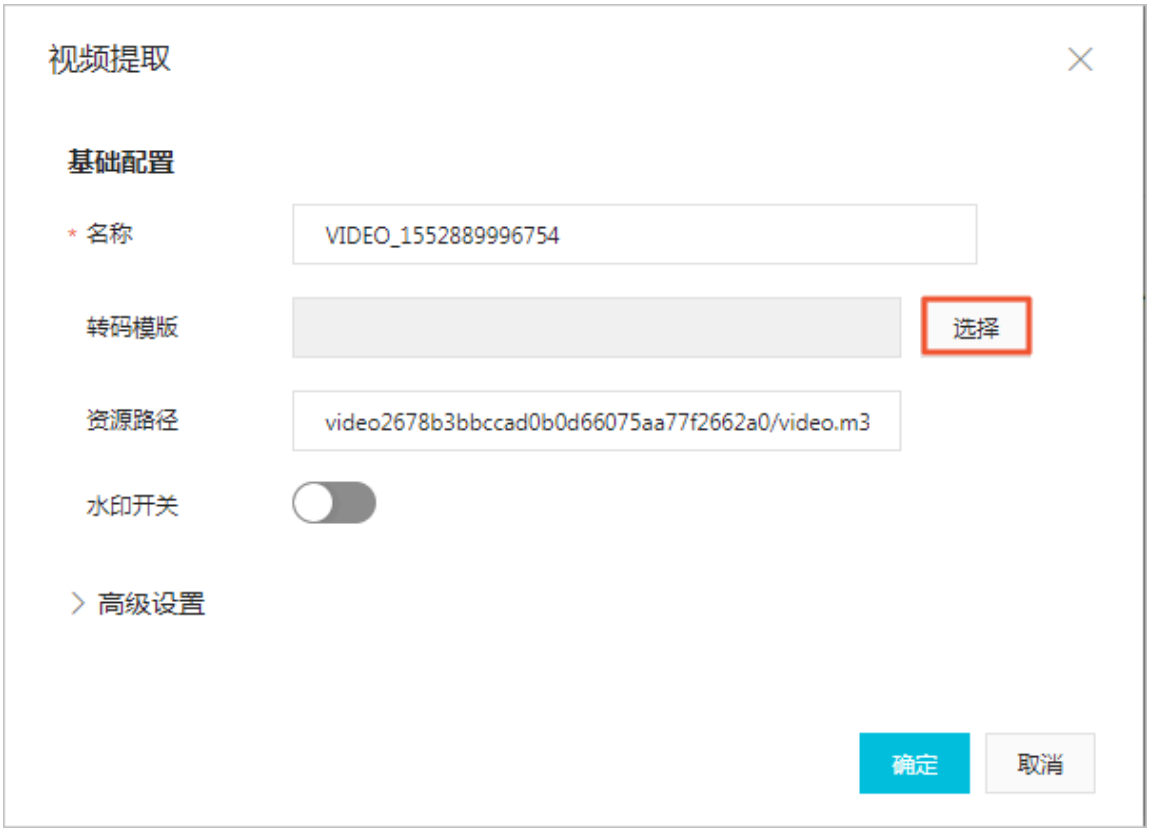


9. 配置 视频提取 节点。

a. 单击 视频提取节点右侧的  图标进行配置。

b. 修改 名称，或者保持系统默认的名称。

c. 在 视频提取 > 基础配置 页面，单击 转码模板 右侧的 选择。

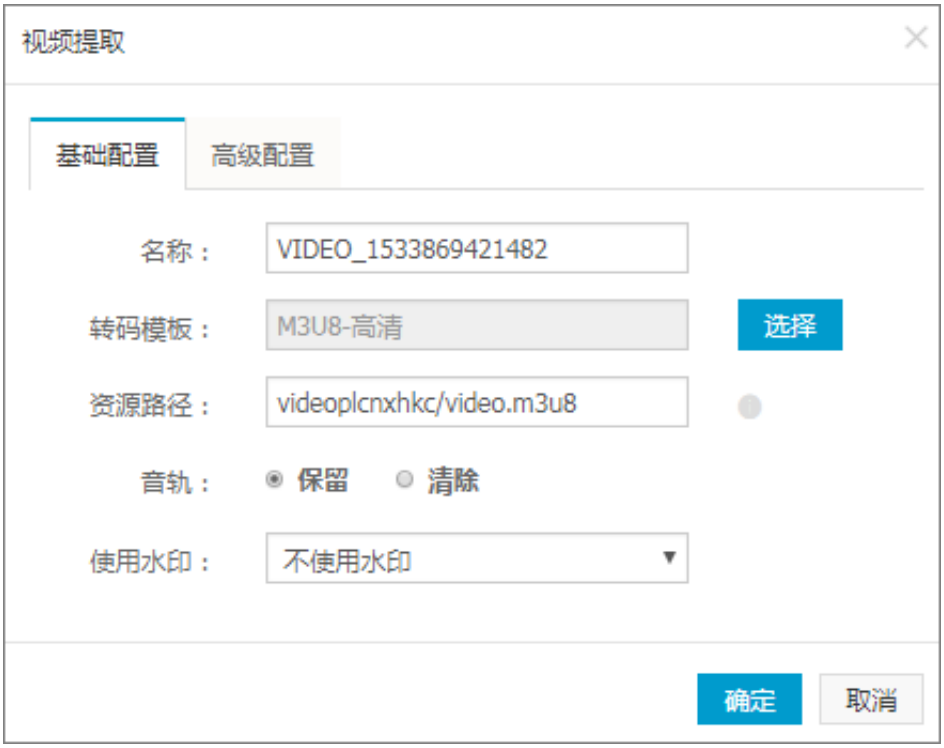


d. 选择 转码模板 并单击 确定。

e. 配置 资源路径。

建议您使用默认值。您也可以根据具体需求进行修改。需要注意的是：若 打包配置 节点的 输出路径为`a/b/c.m3u8`，提取节点 资源路径 为`d/e/f.m3u8`，则提取文件实际存放位置为`a/b/d/e/f.m3u8`。

f. 在 音轨 中，选择 保留。



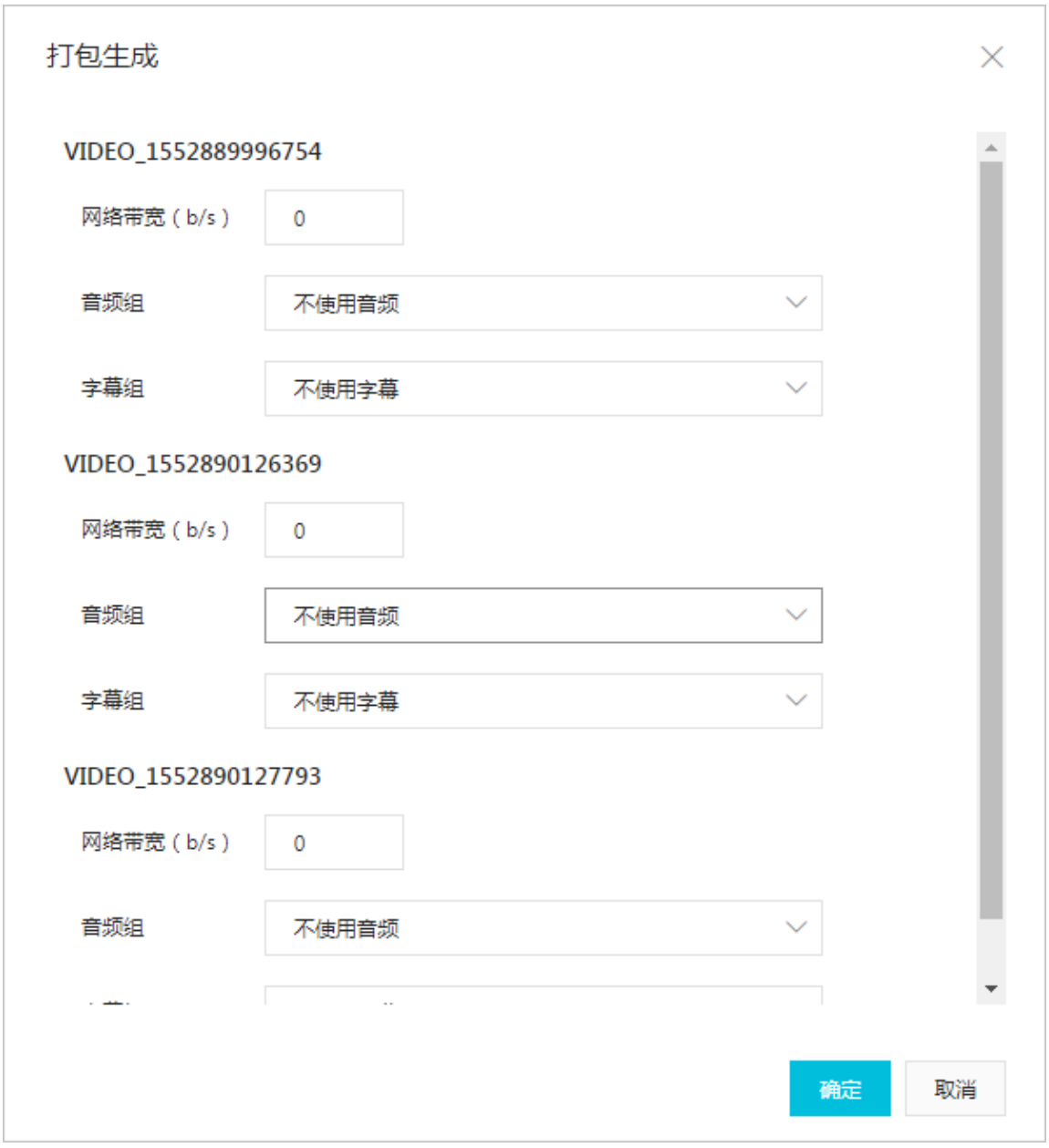
说明:

请您按照上述操作分别配置3个 视频提取 节点，转码模板分别对应480P、720P、1080P。

10.配置 打包生成 节点。

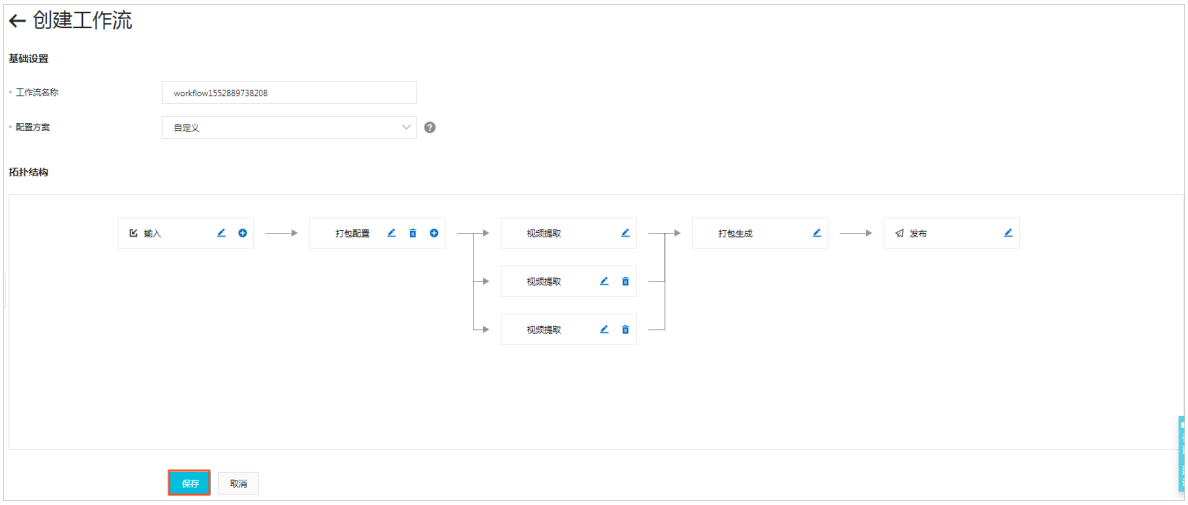
a. 单击 打包生成节点右侧的  图标进行配置。

b. 您可以根据实际需求修改 网络宽带 的值。



11.单击 确定，完成节点设置。

12.单击 下一步。



工作流创建完成。

13.提交任务。

HLS打包工作流默认不会自动触发。请您调用 `AddMedia` 接口指定视频及HLS打包工作流ID进行视频的处理。