ALIBABA CLOUD

阿里云

媒体处理
最佳实践

文档版本：20220606

阿里云

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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会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，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	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。	单击**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。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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。	`switch {active|stand}`

目录

1.转码
1.1. 如何进行API转码

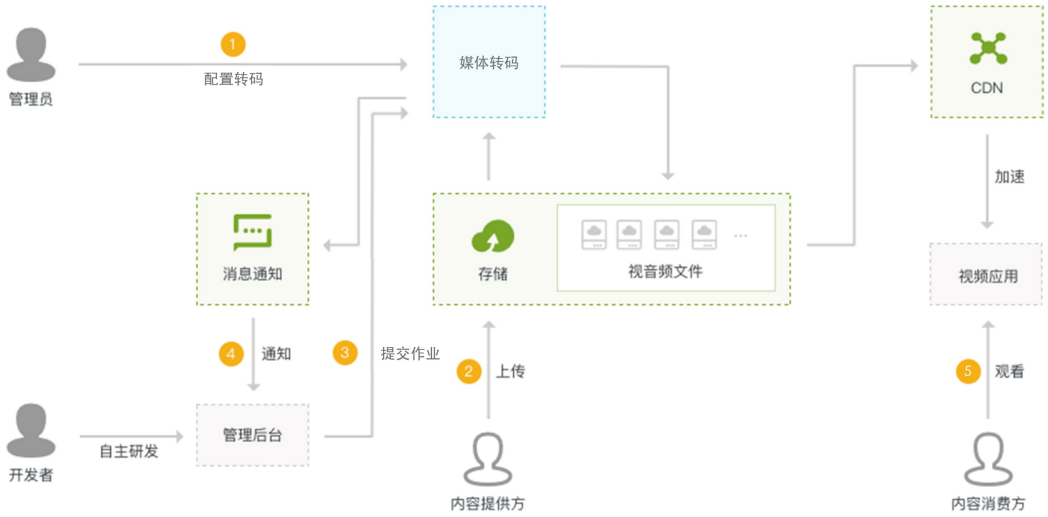
背景

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。

架构图如下所示：



解析

1. 配置转码模板、水印模板、设置管道消息通知。
2. 上传视频到OSS。
3. 调用API，提交转码作业。
4. 等待媒体转码完成，并发送完成消息到队列。
5. 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

2.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

在提交转码作业时，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）

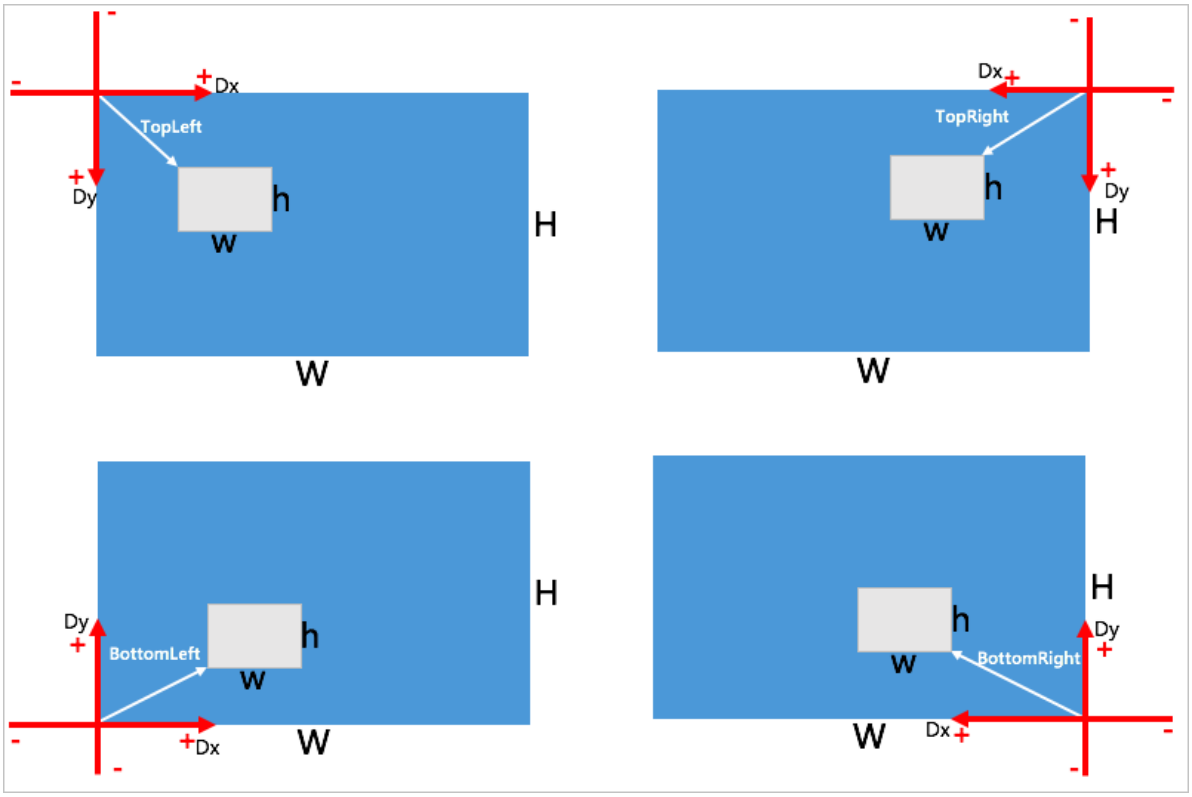
  添加图片水印、动画水印时，Type设置为Image，同时设置InputFile参数，即水印素材的OSS存储路径。

  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。详情请参见水印模板配置。

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。

- 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。

示例代码

在转码成**720P（1280×720）**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

3.如何设置截图

视频截图是截取视频中特定位置的图像，保存为图片文件。

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，请参见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。

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，需要关注Input参数，来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。

- 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输出文件和拼图配置（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。

执行方式

执行方式详情，请参见 作业和管道>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

例如一个**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

4.加密

4.1. 如何进行HLS的加密与播放

本文提供了创建HLS标准加密工作流到播放加密视频的一个完整步骤。

操作步骤

HLS标准加密架构，请参见HLS的加密与播放。

1. 创建HLS加密工作流。

   控制台方式创建HLS加密工作流，请参见阿里云私有加密。

   接口方式创建HLS加密工作流，DEMO代码，请参见创建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，播放器会在该地址请求密钥，媒体处理会在本地启动服务，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：

   Java SDK Core

   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, accessKeyS
ecret));
}
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 InetSocketAddress(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. 可通过查询媒体接口，获取播放地址。请参见查询媒体-使用媒体ID。
5. 播放视频。

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

   将获取的播放地址，如图填入对话框中，单击**视频播放**。

> **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。

5.拼接剪辑

5.1. 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。

视频拼接

在视频拼接时，您需要关注以下参数：

- Input（输入文件）

  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。

- Output（输出参数）

  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** `MergeLis t` 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。

  > **说明** 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。

视频剪辑

在视频剪辑中，您需要关注以下参数：

- Input（输入文件）

  设置待剪辑视频的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。

- Output（输出参数）

  在输出参数中，您需要关注以下参数：

  - TimeSpan

    剪辑的时间区间。您可以根据实际需要设置不同的时间节点与剪辑时长。

    每个时间区间包含三个参数：

    - Seek

      剪辑开始的时间点。

    - Duration

      剪辑持续时长。

    - End

      截尾时长，表示切掉尾部的若干时长。

      > **说明** 设置此值时，参数Duration失效。

  - ConfigToClipFirstPart

    是否剪辑第一片。可设置的值为false（拼接完后剪辑）、true（先剪辑第一片后拼接）。默认值为false。

示例代码

例如一个720P（1280×720）的正片视频，拼接上片头片尾是480P（640×480）的MP4视频，输出分辨率是1280×720。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

5.2. 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（输入文件）

  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。

- Output（输出参数）

  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。

- 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

例如一个720P（1280×720）的正片视频，拼接上开板和尾板是480P（640×480）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

6.打包

6.1. 如何进行HLS打包

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。

操作步骤

1. 在创建工作流时，请关以下对象。创建工作流API接口，请参见新增媒体工作流。

对象	描述
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下文活动说明表中的活动。
Dependencies	依赖关系是拓扑结构中的边，指明活动之间的依赖。

活动说明表

活动	描述	前后依赖
PackageConfig活动	指定HLS打包配置，设置Master Playlist文件输出位置。	○ 前置节点允许：Start。 ○ 后置节点允许：SubtitleGroup、AudioGroup、Transcode（仅视频）。
SubtitleGroup活动	指定字幕分组ID。	○ 前置节点允许：PackageConfig。 ○ 后置节点允许：Transcode（仅字幕）。
AudioGroup活动	指定音频分组ID。	○ 前置节点允许：PackageConfig。 ○ 后置节点允许：Transcode（仅音频）。
Transcode活动	用于提取视频流、音频流、字幕流。	○ 前置节点允许：PackageConfig、SubtitleGroup、AudioGroup。 ○ 后置节点允许：GenerateMasterPlayList。

活动	描述	前后依赖
GenerateMasterPlayList活动	HLS打包生成活动，指定视频多码率配置，指定音频，字幕分组。	○ 前置节点允许：Transcode。 ○ 后置节点允许：Report。

2. 调用 新增媒体接口，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。

  示例代码如下：

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

  - Output设置Master Playlist的存储位置及名称，请参见PackageConfig活动。
  - 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。示例代码如下：

```
{
"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

  示例代码如下：

```
{
"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-U
S\"}"
}
```

  - TemplateId：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。更多详情，请参见预置模板详情。
  - AudioStreamMap：音频流选择字。更多详情，请参见Output详情。
  - Remove：在输出中移除掉视频流。更多详情，请参见参数详情。
  - ExtXMedia：定义Media Playlist，URI指定Media Playlist的名称。更多详情，请参见ExtXMedia详情。
  - 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。示例代码如下：

```
{
"Name" : "video-extract",
"Parameters" : {
"Outputs" : "[{\"TemplateId\":\"1fe5393bdb7b2b883f0a0fc91e81****\",\"Audio\":{\"Remove\":\"
true\"}}]",
"MultiBitrateVideoStream" : "{\"URI\": \"sd/video1.m3u8\"}"
},
"Type" : "Transcode"
}
```

- 自定义转码模板ID：1fe5393bdb7b2b883f0a0fc91e81****，登录媒体处理控制台，在左侧导航栏选择**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，请参见参数详情。
- 参数详情定义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。示例代码如下：

```
{
"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。示例代码如下：

```
{
"Name" : "subtitle-extract-1",
"Parameters" : {
"WebVTTSubtitleURL" : "http://example-bucket-****.oss-cn-hangzhou.aliyun-inc.com/ShawshankR
edemption****.vtt",
"ExtXMedia" : "{\"URI\": \"zh/subtitle1-cn.m3u8\",\"Name\": \"subtitle-cn\",\"Language\": \
"cn\"}"
},
"Type" : "Transcode"
}
```

- Transcode活动中的 `WebVTTSubtitleURL` 参数是指定字幕地址，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。示例代码如下：

```
{
"Parameters" : {
"MasterPlayList" : "{\"MultiBitrateVideoStreams\": [{\"RefActivityName\": \"video-extract\"
,\"ExtXStreamInfo\": {\"BandWidth\": \"1110000\",\"Audio\": \"audios\",\"Subtitles\": \"sub
titles\"}}]}"
},
"Type" : "GenerateMasterPlayList"
}
```

- MasterPlayList：定义Master Playlist。更多详情，请参见MasterPlayList详情。
- MultiBitrateVideoStreams：多码率视频流数组。更多详情，请参见MasterPlayList详情。
- 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。更多详情，

请参见ExtXStreamInfo详情。

- 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。示例代码如下：

```
{
"Activities" : {
"package-node" : {
"Name" : "package-node",
"Parameters" : {
"Output" : "{\"Bucket\": \"processedmediafile\",\"Location\": \"oss-cn-hangzhou\",\"MasterP
layListName\": \"{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\":\"
```

```
"Outputs" : "[{\"TemplateId\":\"11e5395bdb7b2b88510a01c91e81544b\",\"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-U
S\"}"
},
"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://example-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://example-bucket-****.oss-cn-hangzhou.aliyun-inc.com/ShawshankR
edemption****.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\": \"s
```

```
ubtitles\"}}, {\"RefActivityName\": \"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\",\"ObjectPrefi
x\":\"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","subtitleGroupNode", "audioGroupNod
e" ],
"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

6.2. 如何进行DASH打包

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。本文介绍了新建DASH打包工作流、调用AddMedia接口指定视频及DASH打包工作流ID进行视频处理的操作步骤。

使用说明

DASH打包工作流目前只能通过API生成，不能通过控制台生成。通过API生成的工作流可以通过控制台查看和使用。更多信息，请参见新增媒体工作流。

操作步骤

1. 在创建工作流时，请关注以下对象。创建工作流API接口，请参见新增媒体工作流。

   > **说明** 多次打包只需设置一次工作流。

对象	描述
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的活动请参见下文活动说明表。
Dependencies	依赖关系是拓扑结构中的边，指明活动之间的依赖。

活动说明表

活动	描述	前后依赖
PackageConfig活动	指定DASH打包配置，设置Master Playlist文件输出位置。	○ 前置节点允许：Start。 ○ 后置节点允许：SubtitleGroup、AudioGroup、VideoGroup。
SubtitleGroup活动	指定字幕分组ID和语言。	○ 前置节点允许：PackageConfig。 ○ 后置节点允许：Transcode（仅字幕）。
AudioGroup活动	指定音频分组ID和语言。	○ 前置节点允许：PackageConfig。 ○ 后置节点允许：Transcode（仅音频）。

活动	描述	前后依赖
VideoGroup活动	指定视频分组ID。	○ 前置节点允许：PackageConfig。 ○ 后置节点允许：Transcode（仅视频）。
Transcode活动	用于提取视频流、音频流、字幕流。	○ 前置节点允许：SubtitleGroup、AudioGroup、VideoGroup。 ○ 后置节点允许：GenerateMasterPlayList。
GenerateMasterPlayList活动	打包生成活动。	○ 前置节点允许：Transcode。 ○ 后置节点允许：Report。

2. 调用新增媒体接口，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://example-bucket-****.oss-cn-hangzhou.aliyuncs.com/package/subtitle/CENG.stl"}}}}` 其中subtitleTransNode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\": \"exampleBucket****\",\"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。

活动定义：

```
"audio-cn-group" : {
"Name" : "audio-cn-group",
"Parameters" : {
"AdaptationSet" : "{\"Lang\":\"chinese\",\"Group\":\"AudioGroupChinese\"}"
},
"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\":\"d053297fc44f9DashTempla****\",\"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，请参见Output详情说明。
  - 在输出中移除掉视频流，请参见参数详情说明。
  - 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\":\"d861b90f6c0aed8f81095e5c5b85****\",\"Audio\":{\"Remove\":
\"true\"}}]",
"Representation" : "{\"Id\":\"476pSD\",\"URI\":\"videoSD/****.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，请参见参数详情。
- 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\":\"SubtitleENGroup\"}"
},
"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://exampleBucket****.os
s-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，字幕地址在调用新增媒体时可以被动态覆盖，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。

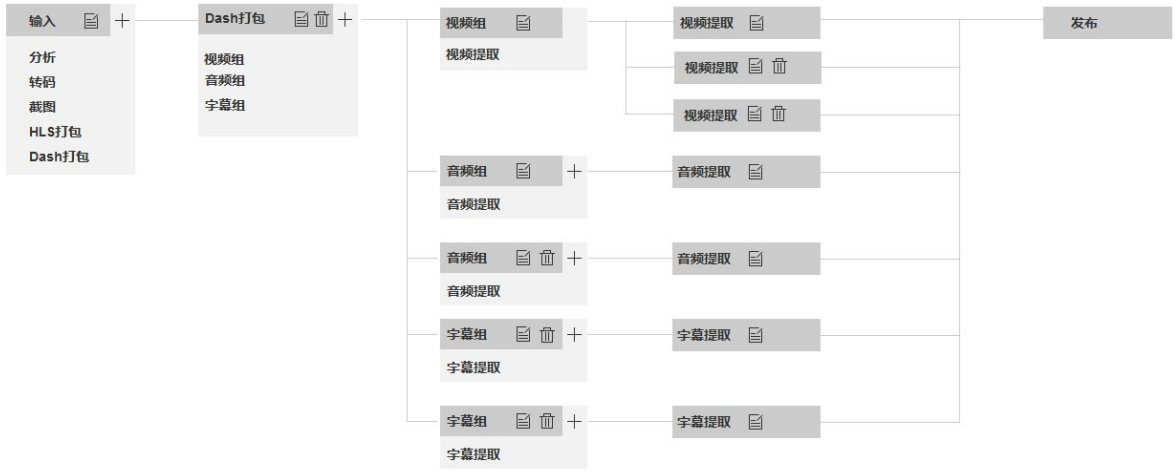
输出Master Playlist。

- 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\",\"MasterPlay
ListName\": \"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\":\"t
rue\"}}]",
"Representation": "{\"Id\":\"270pLD\", \"URI\":\"videoLD/xx.mpd\"}"
},
"Type": "Transcode"
},
"videoTransSD": {
"Name": "videoTransSD",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d861b90f6c0aed8f81095e5c5b85****\",\"Audio\":{\"Remove\":\"t
rue\"}}]",
"Representation": "{\"Id\":\"480pSD\", \"URI\":\"videoSD/****.mpd\"}"
},
"Type": "Transcode"
},
"videoTransHD": {
"Name": "videoTransHD",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"117b3ae88efbc97df372cfd9a0e1****\",\"Audio\":{\"Remove\":\"t
rue\"}}]",
"Representation": "{\"Id\":\"720pHD\", \"URI\":\"videoHD/****.mpd\"}"
},
"Type": "Transcode"
},
"audioCNTransNode": {
"Name": "audioCNTransNode",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d053297fc44f9dd6becd4a98d1c4****\",\"AudioStreamMap\":\"0:a:
0\",\"Video\":{\"Remove\":\"true\"}}]",
"Representation": "{\"Id\":\"chinese128k\", \"URI\":\"audiocn/cn-abc.mpd\"}"
```

```
},
"Type": "Transcode"
},
"audioENTransNode": {
"Name": "audioENTransNode",
"Parameters": {
"Outputs": "[{\"TemplateId\":\"d053297fc44f9dd6becd4a98d1c4****\",\"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://exampleBucket****.oss-c
n-hangzhou.aliyuncs.com/dashpackage/subtitle/Subtitle****.EN.vtt\"}}",
"Representation": "{\"Id\":\"subtitle-english\", \"URI\":\"subtitle/en-****.vtt\"}"
},
"Type": "Transcode"
},
"subtitleCNNode": {
"Name": "subtitleCNNode",
"Parameters": {
"InputConfig": "{\"Format\":\"vtt\",\"InputFile\":{\"URL\":\"http://exampleBucket****.oss-c
n-hangzhou.aliyuncs.com/dashpackage/subtitle/Subtitle.CN.vtt\"}}",
"Representation": "{\"Id\":\"subtitle-chinese\", \"URI\":\"subtitle/cn-****.vtt\"}"
},
"Type": "Transcode"
},
"act-report": {
"Name": "act-report",
"Parameters": {
"PublishType": "Auto"
},
"Type": "Report"
},
"act-start": {
"Name": "act-start",
"Parameters": {
"PipelineId": "cc7fcef2562e4abc9332d491f933****",
"InputFile": "{\"Bucket\":\"exampleBucket****\",\"Location\":\"oss-cn-hangzhou\",\"ObjectPr
efix\":\"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-Java
   - 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.视频DNA

8.（下线）智能审核

9.（下线）DRM

9.1. ChinaDRM工作流

9.2. Widevine工作流