

媒体处理

SDK 参考

文档版本 01
发布日期 2026-02-25



版权所有 © 华为云计算技术有限公司 2026。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

目录

1 SDK 介绍	1
2 Java SDK	2
2.1 开发前准备	2
2.2 视频转码	8
2.2.1 新建转码任务	8
2.2.2 取消转码任务	10
2.2.3 查询转码任务	11
2.3 抽帧截图	13
2.3.1 新建截图任务	13
2.3.2 取消截图任务	15
2.3.3 查询截图任务	16
2.4 动图管理	17
2.4.1 新增动图任务	17
2.4.2 查询动图任务	19
2.4.3 取消动图任务	21
2.5 视频解析	22
2.5.1 创建视频解析任务	22
2.5.2 查询视频解析任务	24
2.5.3 取消视频解析任务	25
2.6 转封装管理	26
2.6.1 创建转封装任务	27
2.6.2 查询转封装任务	28
2.6.3 取消转封装任务	30
2.7 配置转码模板	31
2.7.1 新建转码模板	31
2.7.2 删除转码模板	33
2.7.3 更新转码模板	34
2.7.4 查询转码模板	36
2.8 配置水印模板	38
2.8.1 新建水印模板	38
2.8.2 更新水印模板	39
2.8.3 查询水印配置模板	41
2.8.4 删除水印模板	42

2.9 SDK & API 对应关系.....	43
3 Python SDK.....	46
4 Go SDK.....	51
5 附录.....	55
5.1 JDK 安装.....	55
5.2 错误码.....	56
5.3 返回状态码.....	71
5.4 获取关键参数.....	72
6 修订记录.....	74

1 SDK 介绍

媒体处理SDK提供了创建转码任务、取消转码任务、查询转码任务、创建转码配置模板、删除转码配置模板、更新转码配置模板、查询转码配置模板等。

目前暂提供了JAVA、Python和Go三种语言SDK，若您有其它开发语言的需求，建议您通过[媒体处理API](#)进行调用。

📖 说明

- 媒体处理SDK代码不支持转义。
- 如果无法正常打开GitHub仓库，请检查您所使用的网络是否可以正常访问公网。由于GitHub的服务器部署在国外，国内用户访问时容易遇到无法打开的情况，请切换网络后再尝试打开。

表 1-1 服务端 SDK

语言	Github地址	参考文档
JAVA	huaweicloud-sdk-java-v3	Java SDK使用指导
Python	huaweicloud-sdk-python-v3	Python SDK使用指导
Go	huaweicloud-sdk-go-v3	Go SDK使用指导

2 Java SDK

2.1 开发前准备

本章节介绍了Java SDK的使用说明，您可以参考本章节进行快速集成开发。

开发前准备

- 已[注册](#)华为账号并开通华为云，已进行[实名认证](#)。
- 已具备开发环境，支持Java JDK 1.8及其以上版本。
- 已获取账号对应的 Access Key (AK) 和 Secret Access Key (SK)。请在控制台“我的凭证 > 访问密钥”页面上创建和查看您的 AK/SK。具体请参见[访问密钥](#)。
- 已获取转码服务对应区域的项目ID，请在控制台“我的凭证 > API凭证”页面上查看项目ID。具体请参见[API凭证](#)。
- 已将需要处理的媒资文件上传至MPC同区域的OBS桶中，并将OBS桶进行授权，允许MPC访问。具体请参见[上传音视频文件](#)和[获取云资源授权](#)。
- 媒体处理的SDK版本号详见[SDK开发中心](#)。

Maven 安装配置

[下载](#)settings.xml文件，覆盖<Maven安装目录>/conf/settings.xml文件即可。如果您不想覆盖配置文件，可以依次按照下面方法手动修改settings.xml文件。

1. 在profiles节点中添加如下内容：

```
<profile>
  <id>MyProfile</id>
  <repositories>
    <repository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
```

```
<pluginRepository>
  <id>HuaweiCloudSDK</id>
  <url>https://repo.huaweicloud.com/repository/maven</url>
  <releases>
    <enabled>>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
```

2. 在mirrors节点中增加:

```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*,!HuaweiCloudSDK</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven</url>
</mirror>
```

3. 增加activeProfiles标签激活配置:

```
<activeProfiles>
  <activeProfile>MyProfile</activeProfile>
</activeProfiles>
```

4. 在pom.xml文件中引入媒体处理SDK, 详见表2-1。

```
<dependency>
  <groupId>com.huaweicloud.sdk</groupId>
  <artifactId>huaweicloud-sdk-mpc</artifactId>
  <version>3.1.88</version>
</dependency>
```

表 2-1 媒体处理 SDK 的依赖

依赖名称	版本
com.huaweicloud.sdk:huaweicloud-sdk-core	3.1.88
org.yaml:snakeyaml	2.0
com.fasterxml.jackson.core:jackson-core	2.14.2
com.fasterxml.jackson.core:jackson-annotations	2.14.2
com.fasterxml.jackson.core:jackson-databind	2.14.2

开始使用

- 步骤1** 导入依赖模块。

```
// 用户身份认证
import com.huaweicloud.sdk.core.auth.BasicCredentials;
// 请求异常类
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ServerResponseException;
// Http配置
import com.huaweicloud.sdk.core.http.HttpConfig;
// 导入mpc的客户端
import com.huaweicloud.sdk.mpc.v1.MpcClient;
// 导入待请求接口的request和response类
```

```
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskResponse;
// 日志打印
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

步骤2 配置客户端属性。

1. 默认配置。

```
// 使用默认配置
HttpConfig config = HttpConfig.getDefaultHttpConfig();
```

2. 代理配置（可选）。

```
// 使用代理服务器（可选）
String userName = System.getenv("USER_NAME");
String userPassword = System.getenv("USER_PASSWORD");
config.withProxyHost("http://proxy.myhuaweicloud.com")
    .withProxyPort(8080)
    .withProxyUsername(userName)
    .withProxyPassword(userPassword);
```

3. 连接配置（可选）。

```
// 配置连接超时（可选）
config.withTimeout(3);
```

4. SSL配置（可选）。

```
// 配置跳过服务端证书验证（可选）
config.withIgnoreSSLVerification(true);
```

步骤3 初始化认证信息。

支持两种方式认证，您可以根据实际情况进行选择。

- 使用永久AK/SK

首先需要获取永久AK和SK，以及projectId，您可以参考[开发前准备](#)获取。

```
String ak = System.getenv("SDK_AK");
String sk = System.getenv("SDK_SK");
String projectId = System.getenv("PROJECT_ID");
BasicCredentials credentials = new BasicCredentials()
    .withAk(ak)
    .withSk(sk)
    .withProjectId(projectId)
```

- 使用临时AK/SK

首先需要获取临时AK、SK和SecurityToken，您可以[通过token获取](#)或者[通过委托授权获取](#)。

```
String ak = System.getenv("SDK_AK");
String sk = System.getenv("SDK_SK");
String projectId = System.getenv("PROJECT_ID");
String securityToken = System.getenv("SECURITY_TOKEN");
BasicCredentials credentials = new BasicCredentials()
    .withAk(ak)
    .withSk(sk)
    .withSecurityToken(securityToken)
    .withProjectId(projectId)
```

相关参数说明如下所示：

- **ak**：账号Access Key。
- **sk**：账号Secret Access Key。
- **projectId**：云服务所在区域的项目ID，根据您需要操作的项目所属区域选择对应的项目ID。
- **securityToken**：采用临时AK/SK认证场景下的安全票据。

步骤4 初始化客户端。

```
//初始化MPC的客户端
MpcClient mpcClient = MpcClient.newBuilder()
    .withHttpConfig(config)
    .withCredential(credentials)
    .withEndpoint(endpoint)
    .build();
```

endpoint: MPC应用区域和各服务的终端节点，具体请参见[地区和终端节点](#)。

步骤5 发送请求并查看响应。

```
// 初始化请求，以调用查询转码模板接口为例
ListTranscodingTaskResponse response = mpcClient.
listTranscodingTask(new ListTranscodingTaskRequest().withTaskId(Collections.singletonList(1900293L))
);
logger.info(response.toString());
```

步骤6 异常处理。

表 2-2 异常处理

一级分类	一级分类说明	二级分类	二级分类说明
ConnectionException	连接类异常	HostUnreachableException	网络不可达、被拒绝
		SslHandShakeException	SSL认证异常
RequestTimeoutException	响应超时异常	CallTimeoutException	单次请求，服务器处理超时未返回
		RetryOutageException	在重试策略消耗完成以后，仍无有效的响应
ServiceResponseException	服务器响应异常	ServerResponseException	服务端内部错误，Http响应码：[500,]
		ClientRequestException	请求参数不合法，Http响应码：[400, 500]

```
// 异常处理
try {
ListTranscodingTaskResponse response= mpcClient.listTranscodingTask(new
ListTranscodingTaskRequest().withTaskId(Collections.singletonList(1900293L)));
} catch (ServiceResponseException e) {
logger.error("HttpStatusCode: " + e.getHttpStatusCode());
logger.error("RequestId: " + e.getRequestId());
logger.error("ErrorCode: " + e.getErrorCode());
logger.error("ErrorMsg: " + e.getErrorMsg());
}
```

步骤7 异步客户端使用。

```
// 初始化异步客户端
MpcAsyncClient mpcAsyncClient =
MpcAsyncClient.newBuilder()
    .withHttpConfig(config)
    .withCredential(credentials)
    .withEndpoint(endpoint)
```

```

        .build());

// 发送异步请求
CompletableFuture<ListTranscodingTaskResponse> future = mpcAsyncClient.listTranscodingTaskAsync(new
ListTranscodingTaskRequest().withTaskId(Collections.singletonList(1900293L)));

// 获取异步请求结果
ListTranscodingTaskResponse response = future.get();
logger.info(response.toString());

```

步骤8 访问日志。

SDK在运行的时候采用了slf4j进行日志打印，如果在运行代码实例时，未配置日志实现库，会有提示如下：

```

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

```

您可以根据目标项目实际情况引入对应的日志实现，请在对应的工程项目的pom.xml文件中引入日志实现的依赖，如下所示：

- slf4j

```

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.21</version>
</dependency>
logback

<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.2.3</version>
</dependency>
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-core</artifactId>
  <version>1.2.3</version>
</dependency>

```
- log4j

```

<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>

```

SDK默认会打印访问日志，每次请求都会有一条记录，日志名称为"huaweiCloud-SDK-Access"，日志格式如下所示：

```
"{httpMethod} {uri}" {statusCode} {responseContentLength} {requestId}
```

其中“requestId”是APIG返回的请求ID，可以用于问题跟踪。

可以根据项目情况在对应的日志配置文件中对访问日志进行屏蔽，或者单独打印到独立文件中。例如在logback中关闭访问日志：

```
<logger name="huaweiCloud-SDK-Access" level="OFF"> </logger>
```

步骤9 原始HTTP侦听器。

在某些场景下可能对业务发出的Http请求进行Debug，需要看到原始的Http请求和返回信息，SDK提供侦听器功能获取原始的和加密的Http请求和返回信息。

⚠ 注意

原始信息打印仅在debug阶段使用，请不要在生产系统中将原始的Http头和Body信息打印到日志，这些信息并未加密且其中包含敏感数据；当Body为二进制内容，即Content-Type标识为二进制时，body为"****"，详细内容不输出。

```
HttpConfig config = new HttpConfig().addHttpListener(HttpListener.forRequestListener(requestListener ->
// 注册侦听器后打印Http Request 原始信息,请勿在生产系统中使用
logger.debug("REQUEST: {} {} {} {}",
    requestListener.httpMethod(),
    requestListener.uri(),
    requestListener.headers().entrySet().stream().flatMap(entry ->
        entry.getValue().stream().map(value -> entry.getKey() + " : " + value))
        .collect(Collectors.joining(";")),
    requestListener.body().orElse(""));
).addHttpListener(HttpListener.forResponseListener(responseListener ->
// 注册侦听器后打印Http Response 原始信息,请勿在生产系统中使用
logger.debug("RESPONSE: {} {} {} {} {}",
    responseListener.httpMethod(),
    responseListener.uri(),
    responseListener.statusCode(),
    responseListener.headers().entrySet().stream().flatMap(entry ->
        entry.getValue().stream().map(value -> entry.getKey() + " : " + value))
        .collect(Collectors.joining(";")),
    responseListener.body().orElse(""));

MpcClient mpcClient = MpcClient.newBuilder()
    .withHttpConfig(config)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
```

----结束

代码示例 - 初始化 MpcClient

Endpoint调用前请您根据实际情况填写，并替换如下变量："SDK_AK"、"SDK_SK"、{your endpoint string}和{your project id}。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;

public class InitMpc {
    private static HttpConfig httpConfig;
    private static BasicCredentials auth;
    private static String endpoint;
    private static MpcClient mpcClient;

    public static MpcClient getMpcClient() {
        httpConfig = HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置，请根据实际情况设置
        //httpConfig.withProxyHost("xxxxx").withProxyPort(xxxxx).withProxyUsername("xxxxx").
        //    withProxyPassword("xxxxx");

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        String projectId = System.getenv("PROJECT_ID");
        endpoint = "https://mpc.region01.myhuaweicloud.com";
        auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        mpcClient = MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
```

```
        .withEndpoint(endpoint)
        .build();
    return mpcClient;
}
}
```

2.2 视频转码

2.2.1 新建转码任务

您可以通过创建转码MpcClient实例并设置相关参数新建转码任务。

核心代码

1. 创建转码MpcClient实例。

```
public static MpcClient initMpcClient() {
    //设置httpConfig
    HttpConfig httpConfig =
    HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
    //根据实际需要，是否设置http代理
    //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
    //withProxyPassword("xxxxxx");
    //根据实际填写ak,sk，在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

    String ak = System.getenv("SDK_AK");
    String sk = System.getenv("SDK_SK");
    //根据实际填写项目ID，在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
    String projectId = System.getenv("PROJECT_ID");
    //根据实际填写所需Endpoint，这里以“region01”为例
    String endpoint = "https://mpc.region01.myhuaweicloud.com";
    BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
    return MpcClient.newBuilder()
        .withHttpConfig(httpConfig)
        .withCredential(auth)
        .withEndpoint(endpoint)
        .build();
}
```

2. 创建转码请求，并设置请求体。

转码请求包括输入文件、输出文件和转码模板设置。具体参数含义请参考[新建转码任务](#)。

```
//设置转码输入视频地址
ObsObjInfo input = new ObsObjInfo()
    //设置桶名
    .withBucket("mpc-east-2")
    //设置OBS桶所在区域
    .withLocation("region01")
    //设置输入视频对象
    .withObject("input/ok.mp4");
//设置转码输出视频路径
ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01")
    //设置输出路径
    .withObject("output");
//创建转码请求
CreateTranscodingTaskRequest request
    = new CreateTranscodingTaskRequest().withBody(new CreateTranscodingReq()
        .withInput(input)
        .withOutput(output)
        //设置转码模板，预置模板Id可以在MPC console页面“全局设置 > 预置模板”上查看
        .withTransTemplateId(Collections.singletonList(7000530))
        //设置输出名称，名称个数需要与模板个数一一对应
        .withOutputFileNames(Collections.singletonList("output_"))
        //设置截图参数，根据实际需要填充Thumbnail结构
        .withThumbnail(new Thumbnail()));
```

```
//设置加密参数, 根据实际需要填充Encryption结构
//.withEncryption(new Encryption())
);
```

3. 发送转码请求。

```
//发送媒体处理服务请求
CreateTranscodingTaskResponse response = initMpcClient().createTranscodingTask(request);
//返回消息
System.out.println("CreateTranscodingTaskResponse=" + response);
```

示例代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.CreateTranscodingReq;
import com.huaweicloud.sdk.mpc.v1.model.CreateTranscodingTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateTranscodingTaskResponse;
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskResponse;
import com.huaweicloud.sdk.mpc.v1.model.ObsObjInfo;
import com.obs.services.internal.ServiceException;
import org.junit.Test;

import java.util.Arrays;
import java.util.Collections;

public class TestTranscode {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 创建转码任务
     * @param args
     */
    public static void main(String[] args) {
        //设置转码输入视频地址
        ObsObjInfo input = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("ok.mp4");
        //设置转码输出视频路径
        ObsObjInfo output = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("output");
        //创建转码请求
        CreateTranscodingTaskRequest request
```

```

        = new CreateTranscodingTaskRequest().withBody(new CreateTranscodingReq()
        .withInput(input)
        .withOutput(output)
        //设置转码模板，预置模板Id可以在MPC console页面“全局设置 > 预置模板”上查看
        .withTransTemplateId(Collections.singletonList(7000530))
        //设置输出名称，名称个数需要与模板个数一一对应
        .withOutputFileNames(Collections.singletonList("output_"))
        //设置截图参数
        //withThumbnail(new Thumbnail())
        //设置加密参数
        //withEncryption(new Encryption())
        );
        try {
            CreateTranscodingTaskResponse response = initMpcClient().createTranscodingTask(request);
            System.out.println("CreateTranscodingTaskResponse=" + response);
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
}

```

2.2.2 取消转码任务

说明

- 取消转码任务需要用户提供所要取消任务的taskId。
- 待取消的taskId只能是正在转码任务队列中排队的转码任务。已开始转码或已完成的转码任务不能取消。
- 错误处理请参考[错误码表](#)。

设置取消转码参数

```

//取消任务，TaskId是转码请求响应中返回的任务ID
DeleteTranscodingTaskRequest req = new DeleteTranscodingTaskRequest().withTaskId(3273178);
//发送请求
DeleteTranscodingTaskResponse deleteTranscodingTaskResponse =
    initMpcClient().deleteTranscodingTask(req);
//返回处理消息
System.out.println(JsonUtils.toJson(deleteTranscodingTaskResponse));

```

示例代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.DeleteTranscodingTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.DeleteTranscodingTaskResponse;
import com.obs.services.internal.ServiceException;

public class TestDeleteTranscode {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置，请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
    }
}

```

```
//根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

String ak = System.getenv("SDK_AK");
String sk = System.getenv("SDK_SK");
//根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
String projectId = System.getenv("PROJECT_ID");
//根据实际填写所需endpoint, 这里以“region01”为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 取消排队中的任务
 */
public static void main(String[] args) {
    try {
        //取消任务请求, TaskId是转码请求响应中返回的任务ID
        DeleteTranscodingTaskRequest req = new DeleteTranscodingTaskRequest().withTaskId(3273178);
        //发送请求
        DeleteTranscodingTaskResponse deleteTranscodingTaskResponse =
            initMpcClient().deleteTranscodingTask(req);
        //返回处理消息
        System.out.println(JsonUtils.toJSON(deleteTranscodingTaskResponse));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
```

2.2.3 查询转码任务

您可根据转码任务ID、状态、页码数、开始和结束时间，查询单个或多个转码任务的执行情况。

在查询到的结果集中，如果不提供页码数和显示条数并且数据大于10条，会默认显示10条数据并进行分页处理。

具体查询条件和查询结果参数请参考[查询转码任务](#)接口。

查询单个转码任务

```
//按单个TaskId查询任务, TaskId是转码请求响应中返回的任务ID
ListTranscodingTaskRequest req = new
ListTranscodingTaskRequest().withTaskId(Collections.singletonList(3273178L));
//发送请求
ListTranscodingTaskResponse listTranscodingTaskResponse = initMpcClient().listTranscodingTask(req);
System.out.println(JsonUtils.toJSON(listTranscodingTaskResponse));
```

查询多个转码任务

```
//按多个TaskId查询任务, TaskId是转码请求响应中返回的任务ID
ListTranscodingTaskRequest req = new ListTranscodingTaskRequest().withTaskId(Arrays.asList(3273178L,
3273179L));
//发送请求
ListTranscodingTaskResponse listTranscodingTaskResponse = initMpcClient().listTranscodingTask(req);
System.out.println(JsonUtils.toJSON(listTranscodingTaskResponse));
```

根据状态查询

```
//按状态查询任务
ListTranscodingTaskRequest req = new ListTranscodingTaskRequest().withStatus("FAILED");
```

```
//发送请求
ListTranscodingTaskResponse listTranscodingTaskResponse = initMpcClient().listTranscodingTask(req);
System.out.println(JsonUtils.toJson(listTranscodingTaskResponse));
```

根据开始时间和结束时间查询

```
//根据转码任务开始和结束时间查询
ListTranscodingTaskRequest req = new
ListTranscodingTaskRequest().withStartTime("20210401001517").withEndTime("20210402081517");
//发送请求
ListTranscodingTaskResponse listTranscodingTaskResponse = initMpcClient().listTranscodingTask(req);
System.out.println(JsonUtils.toJson(listTranscodingTaskResponse));
```

根据页码查询

```
//根据页码和每页条数查询
ListTranscodingTaskRequest req = new ListTranscodingTaskRequest().withPage(0).withSize(4);
//发送请求
ListTranscodingTaskResponse listTranscodingTaskResponse = initMpcClient().listTranscodingTask(req);
System.out.println(JsonUtils.toJson(listTranscodingTaskResponse));
```

示例代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.util.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListTranscodingTaskResponse;

public class TestListTranscode {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 查询转码任务
     */
    public static void main(String[] args) {
        try {
            //按TaskId查询任务, TaskId是转码请求响应中返回的任务ID
            ListTranscodingTaskRequest req = new
ListTranscodingTaskRequest().withTaskId(Collections.singletonList(3273178L));
            //发送请求
            ListTranscodingTaskResponse listTranscodingTaskResponse =
initMpcClient().listTranscodingTask(req);
```

```
        System.out.println(JsonUtils.toJSON(listTranscodingTaskResponse));
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

2.3 抽帧截图

2.3.1 新建截图任务

前提条件

- 已购买对象存储服务，并参考[上传媒体文件](#)在媒体处理服务同区域（如华北-北京四）上传媒体处理的源视频。
- 已参考[获取云资源授权](#)，完成媒体处理服务授权。

核心代码

1. 创建截图任务请求。

新建截图任务请求包括输入文件、输出文件的路径。具体参数请参考[新建截图任务接口](#)。

```
//设置截图输入视频地址
ObsObjInfo input = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("ok.mp4");
//设置截图输出路径
ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("output");
//创建截图请求
CreateThumbnailsTaskRequest req = new CreateThumbnailsTaskRequest()
    .withBody(new CreateThumbReq().withInput(input).withOutput(output)
        //设置截图类型,此处理按时间点截图
        .withThumbnailPara(new ThumbnailPara().withType(ThumbnailPara.TypeEnum.DOTS)
            //设置截图输出文件名称
            .withOutputFilename("photo")
            //设置截图的时间点
            .withDots(Collections.singletonList(2))
            //设置截图的宽
            .withWidth(480)
            //设置截图的高
            .withHeight(360)));
```

说明：生成的截图文件按截图时间戳命名，从首帧开始截取，中间按时间间隔截取，最后末帧截取一张。如视频文件20s，截图间隔为11s，则生成的截图文件为0.jpg，11.jpg，20.jpg。

2. 发送创建截图任务请求并显示返回消息。

```
CreateThumbnailsTaskResponse rsp = initMpcClient().createThumbnailsTask(req);
System.out.println("CreateThumbnailsTaskResponse=" + JsonUtils.toJSON(rsp));
```

示例代码

```
package SdkTestCase.thumbnail;

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
```

```
import com.huaweicloud.sdk.mpc.v1.model.CreateThumbReq;
import com.huaweicloud.sdk.mpc.v1.model.CreateThumbnailsTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateThumbnailsTaskResponse;
import com.huaweicloud.sdk.mpc.v1.model.ObsObjInfo;
import com.huaweicloud.sdk.mpc.v1.model.ThumbnailPara;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestThumbnail {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
        HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 创建截图任务
     * @param args
     */
    public static void main(String[] args) {
        //设置截图输入视频地址
        ObsObjInfo input = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("ok.mp4");
        //设置截图输出路径
        ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("output");
        //创建截图请求
        CreateThumbnailsTaskRequest req = new CreateThumbnailsTaskRequest()
            .withBody(new CreateThumbReq().withInput(input).withOutput(output))
            //设置截图类型,此处理按时间点截图
            .withThumbnailPara(new ThumbnailPara().withType(ThumbnailPara.TypeEnum.DOTS)
                //设置截图文件名称
                .withOutputFilename("photo")
                //设置截图的时间点
                .withDots(Collections.singletonList(2))
                //设置截图的宽
                .withWidth(480)
                //设置截图的高
                .withHeight(360));

        //发送截图请求
        try {
            CreateThumbnailsTaskResponse rsp = initMpcClient().createThumbnailsTask(req);
            System.out.println("CreateThumbnailsTaskResponse=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException e) {
            System.out.println(e);
        }
    }
}
```

```
}  
}
```

2.3.2 取消截图任务

说明

- 取消任务需要用户提供所要取消任务的taskId。
- 待取消的taskId只能是正在截图任务队列中排队的。已开始或已完成的截图任务不能删除。

核心代码

```
// 发送取消截图任务请求给媒体处理服务  
DeleteThumbnailsTaskRequest req = new DeleteThumbnailsTaskRequest().withTaskId("2210744");  
DeleteThumbnailsTaskResponse rsp = initMpcClient().deleteThumbnailsTask(req);  
// 返回消息  
System.out.println("DeleteThumbnailsTaskResponse=" + JsonUtils.toJson(rsp));
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;  
import com.huaweicloud.sdk.core.exception.ClientRequestException;  
import com.huaweicloud.sdk.core.exception.ConnectionException;  
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;  
import com.huaweicloud.sdk.core.http.HttpConfig;  
import com.huaweicloud.sdk.core.utils.JsonUtils;  
import com.huaweicloud.sdk.mpc.v1.MpcClient;  
import com.huaweicloud.sdk.mpc.v1.model.DeleteThumbnailsTaskRequest;  
import com.huaweicloud.sdk.mpc.v1.model.DeleteThumbnailsTaskResponse;  
import com.obs.services.internal.ServiceException;  
  
public class TestDeleteThumbnail {  
    /**  
     * 初始化 MpcClient  
     * @return  
     */  
    public static MpcClient initMpcClient() {  
        HttpConfig httpConfig =  
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);  
        //http代理设置, 请根据实际情况设置  
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").  
        //    withProxyPassword("xxxxxx");  
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK  
  
        String ak = System.getenv("SDK_AK");  
        String sk = System.getenv("SDK_SK");  
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID  
        String projectId = System.getenv("PROJECT_ID");  
        //根据实际填写所需endpoint, 这里以“region01”为例  
        String endpoint = "https://mpc.region01.myhuaweicloud.com";  
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);  
        return MpcClient.newBuilder()  
            .withHttpConfig(httpConfig)  
            .withCredential(auth)  
            .withEndpoint(endpoint)  
            .build();  
    }  
  
    /**  
     * 取消截图任务  
     * @param args  
     */  
    public static void main(String[] args) {  
        DeleteThumbnailsTaskRequest req = new DeleteThumbnailsTaskRequest().withTaskId("2210744");  
        try {
```

```
        DeleteThumbnailsTaskResponse rsp = initMpcClient().deleteThumbnailsTask(req);
        System.out.println("DeleteThumbnailsTaskResponse=" + JsonUtils.toJSON(rsp));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
```

2.3.3 查询截图任务

说明

- 查询截图任务，支持根据任务ID查询、任务状态、时间段、分页查询和复合查询。
- 在查询到的结果集中，如果不提供页码数page和显示条数size并且数据大于10条时，会默认显示10条数据并进行分页处理。

根据任务 ID 查询

```
//根据任务ID查询，最多支持10个任务ID
ListThumbnailsTaskRequest req = new
ListThumbnailsTaskRequest().withTaskId(Collections.singletonList("2210744"));
// 发送查询截图任务请求给媒体处理服务
ListThumbnailsTaskResponse rsp = initMpcClient().listThumbnailsTask(req);
// 返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

分页查询

```
//分页查询
ListThumbnailsTaskRequest req = new ListThumbnailsTaskRequest().withPage(1).withSize(4);
//发送查询截图任务请求给媒体处理服务
ListThumbnailsTaskResponse rsp = initMpcClient().listThumbnailsTask(req);
//返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

根据时间段查询

```
ListThumbnailsTaskRequest req = new
ListThumbnailsTaskRequest().withStartTime("20201220131400").withEndTime("20201220131400");
// 发送查询截图任务请求给媒体处理服务
ListThumbnailsTaskResponse rsp = initMpcClient().listThumbnailsTask(req);
// 返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

根据任务状态查询

```
// 根据任务的状态查询
ListThumbnailsTaskRequest req = new
ListThumbnailsTaskRequest().withStatus(ListThumbnailsTaskRequest.StatusEnum.FAILED);
// 发送查询截图任务请求给媒体处理服务
ListThumbnailsTaskResponse rsp = initMpcClient().listThumbnailsTask(req);
// 返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

复合查询

```
//复合查询
ListThumbnailsTaskRequest req = new ListThumbnailsTaskRequest().withPage(1).withSize(4)
.withStartTime("20201220131400")
.withEndTime("20201220131400")
.withStatus(ListThumbnailsTaskRequest.StatusEnum.FAILED);
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListThumbnailsTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListThumbnailsTaskResponse;
import com.obs.services.internal.ServiceException;

public class TestListThumbnail {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际情况填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际情况填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际情况填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 查询截图任务
     * @param args
     */
    public static void main(String[] args) {
        ListThumbnailsTaskRequest req = new ListThumbnailsTaskRequest().withPage(1).withSize(4)
            .withStartTime("20201220131400")
            .withEndTime("20201220131400")
            .withStatus(ListThumbnailsTaskRequest.StatusEnum.FAILED);
        try {
            ListThumbnailsTaskResponse rsp = initMpcClient().listThumbnailsTask(req);
            System.out.println("rsp=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

2.4 动图管理

2.4.1 新增动图任务

您可以通过创建MpcClient实例并设置相关参数新建动图任务，动图任务用于将视频转换为动态图。


```
//根据实际填写所需endpoint, 这里以“region01”为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 创建动图任务
 * @param args
 */
public static void main(String[] args) {
    //设置输入视频地址和输出路径
    ObsObjInfo input = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("ok.mp4");
    ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("output");
    //创建动图请求
    CreateAnimatedGraphicsTaskRequest req = new CreateAnimatedGraphicsTaskRequest()
        .withBody(new CreateAnimatedGraphicsTaskReq().withInput(input).withOutput(output)
            .withOutputParam(new AnimatedGraphicsOutputParam()
                //设置动图格式
                .withFormat(AnimatedGraphicsOutputParam.FormatEnum.GIF)
                //设置动图帧率
                .withFrameRate(15)
                //设置起始时间, 单位毫秒
                .withStart(0)
                //设置结束时间, 单位毫秒, 最大时间间隔60s
                .withEnd(3_000)));
    try {
        CreateAnimatedGraphicsTaskResponse rsp = initMpcClient().createAnimatedGraphicsTask(req);
        System.out.println("CreateAnimatedGraphicsTaskResponse=" + JsonUtils.toJSON(rsp));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException e) {
        System.out.println(e);
    }
}
}
```

2.4.2 查询动图任务

说明

- 查询动图任务, 支持根据任务ID查询、任务状态、时间段、分页查询和复合查询。
- 在查询到的结果集中, 如果不提供页码数page和显示条数size并且数据大于10条时, 会默认显示10条数据并进行分页处理。

根据任务 ID 查询

```
//根据任务ID查询, 最多支持10个任务ID
ListAnimatedGraphicsTaskRequest req = new
ListAnimatedGraphicsTaskRequest().withTaskId(Collections.singletonList("3198527"));
// 发送查询动图任务请求给媒体处理服务
ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
// 打印返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

分页查询

```
// 分页查询
ListAnimatedGraphicsTaskRequest req = new ListAnimatedGraphicsTaskRequest().withPage(1).withSize(10);
```

```
// 发送查询动图任务请求给媒体处理服务
ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
// 打印返回消息
System.out.println("rsp=" + JsonUtils.toJson(rsp));
```

根据时间段查询

```
// 根据时间段查询
ListAnimatedGraphicsTaskRequest req = new
ListAnimatedGraphicsTaskRequest().withStartTime("20201220131400").withEndTime("20201221131400");
// 发送查询动图任务请求给媒体处理服务
ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
// 打印返回消息
System.out.println("rsp=" + JsonUtils.toJson(rsp));
```

根据任务状态查询

```
// 根据任务的状态查询
ListAnimatedGraphicsTaskRequest req = new
ListAnimatedGraphicsTaskRequest().withStatus(ListAnimatedGraphicsTaskRequest.StatusEnum.FAILED);
// 发送查询动图任务请求给媒体处理服务
ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
// 打印返回消息
System.out.println("rsp=" + JsonUtils.toJson(rsp));
```

复合查询

```
// 复合查询
ListAnimatedGraphicsTaskRequest req = new ListAnimatedGraphicsTaskRequest().withPage(0).withSize(10)
.withStartTime("20201220131400").withEndTime("20201221131400")
.withStatus(ListAnimatedGraphicsTaskRequest.StatusEnum.FAILED);
// 发送查询动图任务请求给媒体处理服务
ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
// 打印返回消息
System.out.println("rsp=" + JsonUtils.toJson(rsp));
```

完整代码

```
package com.huawei.mpc;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListAnimatedGraphicsTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListAnimatedGraphicsTaskResponse;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestListAnimation {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
//http代理设置, 请根据实际情况设置
//httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
//    withProxyPassword("xxxxxx");
//根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
//根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
```

```
String projectId = System.getenv("PROJECT_ID");
//根据实际填写所需endpoint, 这里以 "region01" 为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 查询动图任务
 * @param args
 */
public static void main(String[] args) {
    //按taskId查询任务, taskId是动图图请求响应中返回的任务ID
    ListAnimatedGraphicsTaskRequest req = new
ListAnimatedGraphicsTaskRequest().withTaskId(Collections.singletonList("3198527"));
    try {
        ListAnimatedGraphicsTaskResponse rsp = initMpcClient().listAnimatedGraphicsTask(req);
        System.out.println("rsp=" + JsonUtils.toJSON(rsp));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
```

2.4.3 取消动图任务

说明

- 取消任务需要用户提供所要取消任务的taskId。
- 待取消的taskId只能是正在任务队列中排队的任务。已开始或已完成的动图任务不能删除。

核心代码

```
DeleteAnimatedGraphicsTaskRequest req = new
DeleteAnimatedGraphicsTaskRequest().withTaskId("3198527");
DeleteAnimatedGraphicsTaskResponse rsp = initMpcClient().deleteAnimatedGraphicsTask(req);
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.util.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.DeleteAnimatedGraphicsTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.DeleteAnimatedGraphicsTaskResponse;
import com.obs.services.internal.ServiceException;

public class TestDeleteAnimation {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
```

```
//httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
//    withProxyPassword("xxxxxx");
//根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

String ak = System.getenv("SDK_AK");
String sk = System.getenv("SDK_SK");
//根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
String projectId = System.getenv("PROJECT_ID");
//根据实际填写所需endpoint, 这里以“region01”为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 取消排队中的动图任务
 * @param args
 */
public static void main(String[] args) {
    DeleteAnimatedGraphicsTaskRequest req = new
DeleteAnimatedGraphicsTaskRequest().withTaskId("3198527");
    try {
        DeleteAnimatedGraphicsTaskResponse rsp = initMpcClient().deleteAnimatedGraphicsTask(req);
        System.out.println("rsp=" + JsonUtils.toJSON(rsp));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
```

2.5 视频解析

2.5.1 创建视频解析任务

您可以通过创建MpcClient实例并设置相关参数新建视频解析任务，视频解析任务用于解析视频的元数据。

前提条件

- 已购买对象存储服务，并参考[上传媒体文件](#)在媒体处理服务同区域（如华北-北京四）上传媒体处理的源视频。
- 已参考[获取云资源授权](#)，完成媒体处理服务授权。

核心代码

1. 创建视频解析任务。

视频解析任务需要设置输入视频文件参数，如果有必要，还可以将元数据生成文件存放在指定的路径下。

```
//设置解析输入视频地址和输出路径
ObsObjInfo input = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("ok.mp4");
ObsObjInfo output = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("output");
//创建解析请求
CreateExtractTaskRequest req = new CreateExtractTaskRequest()
    .withBody(new CreateExtractTaskReq().withInput(input));
```

```
// 发起请求
CreateExtractTaskResponse rsp = initMpcClient().createExtractTask(req);
// 打印结果
System.out.println("CreateExtractTaskResponse=" + JsonUtils.toJSON(rsp));
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.CreateExtractTaskReq;
import com.huaweicloud.sdk.mpc.v1.model.CreateExtractTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateExtractTaskResponse;
import com.huaweicloud.sdk.mpc.v1.model.ObsObjInfo;
import com.obs.services.internal.ServiceException;

public class TestParse {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 创建解析任务
     * @param args
     */
    public static void main(String[] args) {
        //设置解析输入视频地址和输出路径
        ObsObjInfo input = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("ok.mp4");
        ObsObjInfo output = new ObsObjInfo().withBucket("mpc-
east-2").withLocation("region01").withObject("output");
        //创建解析请求
        CreateExtractTaskRequest req = new CreateExtractTaskRequest()
            .withBody(new CreateExtractTaskReq().withInput(input));

        //发送解析请求
        try {
            CreateExtractTaskResponse rsp = initMpcClient().createExtractTask(req);
            System.out.println("CreateExtractTaskResponse=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

```
}  
}
```

2.5.2 查询视频解析任务

说明

- 查询视频解析任务，支持根据任务ID查询、任务状态、时间段、分页查询和复合查询。
- 在查询到的结果集中，如果不提供页码数page和显示条数size并且数据大于10条时，会默认显示10条数据并进行分页处理。

根据任务 ID 查询

```
// 根据任务ID查询，最多支持10个任务ID  
ListExtractTaskRequest req = new ListExtractTaskRequest().withTaskId(Collections.singletonList("3223182"));  
// 发送查询请求给媒体处理服务  
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);  
// 打印返回消息  
System.out.println("rsp=" + rsp.toString());
```

分页查询

```
// 分页查询  
ListExtractTaskRequest req = new ListExtractTaskRequest().withPage(0).withSize(10);  
// 发送查询请求给媒体处理服务  
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);  
// 打印返回消息  
System.out.println("rsp=" + rsp.toString());
```

根据时间段查询

```
// 根据时间段查询  
ListExtractTaskRequest req = new  
ListExtractTaskRequest().withStartTime("20201220131400").withEndTime("20201221131400");  
// 发送查询请求给媒体处理服务  
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);  
// 打印返回消息  
System.out.println("rsp=" + rsp.toString());
```

根据任务状态查询

```
// 根据任务的状态查询  
ListExtractTaskRequest req = new  
ListExtractTaskRequest().withStatus(ListExtractTaskRequest.StatusEnum.FAILED);  
// 发送查询请求给媒体处理服务  
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);  
// 打印返回消息  
System.out.println("rsp=" + rsp.toString());
```

复合查询

```
// 复合查询  
ListExtractTaskRequest req = new ListExtractTaskRequest().withPage(0).withSize(10)  
    .withStartTime("20201220131400").withEndTime("20201221131400")  
    .withStatus(ListExtractTaskRequest.StatusEnum.FAILED);  
// 发送查询请求给媒体处理服务  
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);  
// 打印返回消息  
System.out.println("rsp=" + rsp.toString());
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListExtractTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListExtractTaskResponse;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestListParse {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 查询解析任务
     * @param args
     */
    public static void main(String[] args) {
        ListExtractTaskRequest req = new
ListExtractTaskRequest().withTaskId(Collections.singletonList("3223182"));
        try {
            ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);
            System.out.println("rsp=" + rsp.toString());
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

2.5.3 取消视频解析任务

说明

- 取消任务需要用户提供所要取消任务的taskId。
- 待取消的taskId只能是正在任务队列中排队的任务。已开始或已完成视频解析任务不能删除。

核心代码

```
// 设置需要取消的任务id
DeleteExtractTaskRequest req = new DeleteExtractTaskRequest().withTaskId("3223182");
//发送消息到转码服务
DeleteExtractTaskResponse rsp = initMpcClient().deleteExtractTask(req);
System.out.println("rsp=" + rsp.toString());
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.DeleteExtractTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.DeleteExtractTaskResponse;
import com.obs.services.internal.ServiceException;

public class TestDeleteParse {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际情况填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际情况填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际情况填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 取消排队中的解析任务
     * @param args
     */
    public static void main(String[] args) {
        DeleteExtractTaskRequest req = new DeleteExtractTaskRequest().withTaskId("3223182");
        try {
            DeleteExtractTaskResponse rsp = initMpcClient().deleteExtractTask(req);
            System.out.println("rsp=" + rsp.toString());
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

2.6 转封装管理

2.6.1 创建转封装任务

您可以通过创建MpcClient实例并设置相关参数新建转封装任务，转封装任务用于视频转封装处理。

前提条件

- 已购买对象存储服务，并参考[上传媒体文件](#)在媒体处理服务同区域（如华北-北京四）上传媒体处理的源视频。
- 已参考[获取云资源授权](#)，完成媒体处理服务授权。

核心代码

1. 创建转封装任务。

```
//设置转封装输入视频地址和输出路径
ObsObjInfo input = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("ok.flv");
ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("output");
//创建转封装请求
CreateRemuxTaskRequest req = new CreateRemuxTaskRequest()
    .withBody(new CreateRemuxTaskReq().withInput(input).withOutput(output))
    // 设置转封装参数
    .withOutputParam(new RemuxOutputParam()
        //设置转封装格式
        .withFormat("HLS")
        //转成hls切片间隔
        .withSegmentDuration(5));
//发送转封装请求
CreateRemuxTaskResponse rsp = initMpcClient().createRemuxTask(req);
System.out.println(rsp.toString());
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.CreateRemuxTaskReq;
import com.huaweicloud.sdk.mpc.v1.model.CreateRemuxTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateRemuxTaskResponse;
import com.huaweicloud.sdk.mpc.v1.model.ObsObjInfo;
import com.huaweicloud.sdk.mpc.v1.model.RemuxOutputParam;
import com.obs.services.internal.ServiceException;

public class TestRemux {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
            HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置，请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk，在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID，在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint，这里以“region01”为例
```

```
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 创建转封装任务
 * @param args
 */
public static void main(String[] args) {
    //设置转封装输入视频地址和输出图片路径
    ObsObjInfo input = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("ok.flv");
    ObsObjInfo output = new ObsObjInfo().withBucket("mpc-east-2").withLocation("region01").withObject("output");
    //创建转封装请求
    CreateRemuxTaskRequest req = new CreateRemuxTaskRequest()
        .withBody(new CreateRemuxTaskReq().withInput(input).withOutput(output)
            // 设置转封装参数
            .withOutputParam(new RemuxOutputParam()
                //设置转封装格式
                .withFormat("HLS")
                //转成hls切片间隔
                .withSegmentDuration(5)));
    //发送转封装请求
    try {
        CreateRemuxTaskResponse rsp = initMpcClient().createRemuxTask(req);
        System.out.println(rsp.toString());
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
```

2.6.2 查询转封装任务

说明

- 查询转封装任务，支持根据任务ID查询、任务状态、时间段、分页查询和复合查询。
- 在查询到的结果集中，如果不提供页码数page和显示条数size并且数据大于10条时，会默认显示10条数据并进行分页处理。

根据任务 ID 查询

```
//查询转封装任务
ListRemuxTaskRequest req = new ListRemuxTaskRequest().withTaskId(Collections.singletonList("8191203"));
// 发送查询请求给媒体处理服务
ListRemuxTaskResponse rsp = initMpcClient().listRemuxTask(req);
// 打印返回消息
System.out.println("rsp=" + rsp.toString());
```

分页查询

```
// 分页查询
ListRemuxTaskRequest req = new ListRemuxTaskRequest().withPage(0).withSize(10);
// 发送查询请求给媒体处理服务
ListRemuxTaskResponse rsp = initMpcClient().listRemuxTask(req);
// 打印返回消息
System.out.println("rsp=" + rsp.toString());
```

根据时间段查询

```
// 根据时间段查询
ListRemuxTaskRequest req = new
ListRemuxTaskRequest().withStartTime("20201220131400").withEndTime("20201221131400");
// 发送查询请求给媒体处理服务
ListRemuxTaskResponse rsp = initMpcClient().listRemuxTask(req);
// 打印返回消息
System.out.println("rsp=" + rsp.toString());
```

根据任务状态查询

```
// 根据任务的状态查询
ListRemuxTaskRequest req = new
ListRemuxTaskRequest().withStatus(ListRemuxTaskRequest.StatusEnum.FAILED);
// 发送查询请求给媒体处理服务
ListRemuxTaskResponse rsp = initMpcClient().listRemuxTask(req);
// 打印返回消息
System.out.println("rsp=" + rsp.toString());
```

复合查询

```
// 复合查询
ListRemuxTaskRequest req = new ListRemuxTaskRequest().withPage(0).withSize(10)
.withStartTime("20201220131400").withEndTime("20201221131400")
.withStatus(ListRemuxTaskRequest.StatusEnum.FAILED);
// 发送查询请求给媒体处理服务
ListExtractTaskResponse rsp = initMpcClient().listExtractTask(req);
// 打印返回消息
System.out.println("rsp=" + rsp.toString());
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListRemuxTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListRemuxTaskResponse;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestListRemux {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际情况填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际情况填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际情况填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
```

```

        .withEndpoint(endpoint)
        .build();
    }

    /**
     * 查询转封装任务
     * @param args
     */
    public static void main(String[] args) {
        ListRemuxTaskRequest req = new
ListRemuxTaskRequest().withTaskId(Collections.singletonList("8191203"));
        try {
            ListRemuxTaskResponse rsp = initMpcClient().listRemuxTask(req);
            System.out.println("rsp=" + rsp.toString());
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}

```

2.6.3 取消转封装任务

说明

- 取消任务需要用户提供所要取消任务的taskId。
- 待取消的taskId只能是正在任务队列中排队的任务。已开始或已完成的转封装任务不能删除。

核心代码

```

// 设置需要取消的任务ID
CancelRemuxTaskRequest req = new CancelRemuxTaskRequest().withTaskId("8191203");
// 发送消息到转码服务
CancelRemuxTaskResponse rsp = initMpcClient().cancelRemuxTask(req);
System.out.println("rsp=" + rsp.toString());

```

完整代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.CancelRemuxTaskRequest;
import com.huaweicloud.sdk.mpc.v1.model.CancelRemuxTaskResponse;
import com.obs.services.internal.ServiceException;

public class TestDeleteRemux {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
    }
}

```

```
//根据实际填写所需endpoint, 这里以“region01”为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return MpcClient.newBuilder()
    .withHttpConfig(httpConfig)
    .withCredential(auth)
    .withEndpoint(endpoint)
    .build();
}

/**
 * 取消排队中的转封装任务
 * @param args
 */
public static void main(String[] args) {
    CancelRemuxTaskRequest req = new CancelRemuxTaskRequest().withTaskId("8191203");
    try {
        CancelRemuxTaskResponse rsp = initMpcClient().cancelRemuxTask(req);
        System.out.println("rsp=" + rsp.toString());
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
```

2.7 配置转码模板

2.7.1 新建转码模板

您可以通过SDK新建转码模板，用于转码时的转码模板设置。具体的模板参数可以参考[新建转码模板](#)接口。

核心代码

1. 设置转码模板的参数。

```
//创建转码模板请求
CreateTransTemplateRequest req = new CreateTransTemplateRequest()
    .withBody(new TransTemplate().withTemplateName("test_123"))
    //设置视频参数
    .withVideo(new Video()
        // 视频编码格式, 1表示H264, 2表示H265
        .withCodec(1)
        // 设置视频码率, 单位: kbit/s
        .withBitrate(6000)
        // 编码档次, 建议设为3
        .withProfile(3)
        .withLevel(15)
        // 编码质量, 值越大质量越高, 耗时越长
        .withPreset(3)
        .withRefFramesCount(4)
        .withMaxIframesInterval(5)
        .withBframesCount(4)
        .withHeight(1080)
        .withWidth(1920))
    //设置音频参数
    .withAudio(new Audio()
        //设置音频编码格式, 1: AAC, 2: HEAAC1, 3: HEAAC2, 4: MP3
        .withCodec(1)
        //采样
        //率,1:AUDIO_SAMPLE_AUTO,2:22050Hz,3:32000Hz,4:44100Hz,5:48000Hz,6:96000Hz
        .withSampleRate(4)
        //音频码率, 单位: kbit/s
        .withBitrate(128))
```

```

        //声道数
        .withChannels(2)
    //设置公共参数
    .withCommon(new Common()
        .withDashInterval(5)
        .withHlsInterval(5)
        //高清低码开关
        .withPvc(false)
        //封装类型, 1: HLS, 2: DASH, 3: HLS+DASH, 4: MP4, 5: MP3, 6: ADTS
        .withPackType(1));
    
```

2. 发送新建转码模板请求, 并显示返回消息。

```

//发送创建转码模板请求
CreateTransTemplateResponse rsp = initMpcClient().createTransTemplate(req);
//打印返回消息
System.out.println("CreateTransTemplateResponse=" + JsonUtils.toJSON(rsp));
    
```

完整代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.Audio;
import com.huaweicloud.sdk.mpc.v1.model.Common;
import com.huaweicloud.sdk.mpc.v1.model.CreateTransTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateTransTemplateResponse;
import com.huaweicloud.sdk.mpc.v1.model.TransTemplate;
import com.huaweicloud.sdk.mpc.v1.model.Video;
import com.obs.services.internal.ServiceException;

public class TestTranscodeTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 创建转码模板
     * @param args
     */
    public static void main(String[] args) {
        //创建转码模板请求
        CreateTransTemplateRequest req = new CreateTransTemplateRequest()
            .withBody(new TransTemplate().withTemplateName("test_123"))
            //设置视频参数
    
```

```
.withVideo(new Video()
    // 视频编码格式, 1表示H264, 2表示H265
    .withCodec(1)
    // 设置视频码率, 单位: kbit/s
    .withBitrate(6000)
    // 编码档次, 建议设为3
    .withProfile(3)
    .withLevel(15)
    // 编码质量, 值越大质量越高, 耗时越长
    .withPreset(3)
    .withRefFramesCount(4)
    .withMaxframesInterval(5)
    .withBframesCount(4)
    .withHeight(1080)
    .withWidth(1920))
//设置音频参数
.withAudio(new Audio()
    //设置音频编码格式, 1: AAC, 2: HEAAC1, 3: HEAAC2, 4: MP3
    .withCodec(1)
    //采样
    //率,1:AUDIO_SAMPLE_AUTO,2:22050Hz,3:32000Hz,4:44100Hz,5:48000Hz,6:96000Hz
    .withSampleRate(4)
    //音频码率, 单位: kbit/s
    .withBitrate(128)
    //声道数
    .withChannels(2))
//设置公共参数
.withCommon(new Common()
    .withDashInterval(5)
    .withHlsInterval(5)
    //高清低码开关
    .withPvc(false)
    //封装类型, 1: HLS, 2: DASH, 3: HLS+DASH, 4: MP4, 5: MP3, 6: ADTS
    .withPackType(1));
//发送转码模板请求
try {
    CreateTransTemplateResponse rsp = initMpcClient().createTransTemplate(req);
    System.out.println("CreateTransTemplateResponse=" + JsonUtils.toJSON(rsp));
} catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
    System.out.println(e);
}
}
```

2.7.2 删除转码模板

您可以根据转码模板的ID删除自定义的转码模板。

核心代码

```
//设置删除转码模板ID, 发送删除转码模板请求
DeleteTemplateRequest req = new DeleteTemplateRequest().withTemplateId(346090L);
DeleteTemplateResponse rsp = initMpcClient().deleteTemplate(req);
//返回消息
System.out.println("rsp=" + JsonUtils.toJSON(rsp) + " httpCode=" + rsp.getHttpStatusCode());
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.DeleteTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.DeleteTemplateResponse;
```

```
import com.obs.services.internal.ServiceException;

public class TestDeleteTranscodeTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 删除转码模板
     * @param args
     */
    public static void main(String[] args) {
        //设置待删除的转码模板ID
        DeleteTemplateRequest req = new DeleteTemplateRequest().withTemplateId(346090L);
        try {
            DeleteTemplateResponse rsp = initMpcClient().deleteTemplate(req);
            System.out.println("rsp=" + JsonUtils.toJson(rsp) + " httpCode=" + rsp.getStatusCode());
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

2.7.3 更新转码模板

您可根据转码模板的ID, 重新设置模板参数来更新转码模板。

核心代码

1. 设置转码模板的参数。

```
//设置更新转码模板请求
UpdateTransTemplateRequest req = new UpdateTransTemplateRequest()
    .withBody(new
ModifyTransTemplateReq().withTemplateName("test_123").withTemplateId(346090L)
    //设置视频参数
    .withVideo(new Video()
        // 视频编码格式, 1表示H264, 2表示H265
        .withCodec(1)
        // 设置视频码率, 单位: kbit/s
        .withBitrate(3200)
        // 编码档次, 建议设为3
        .withProfile(3)
        .withLevel(15)
        // 编码质量, 值越大质量越高, 耗时越长
        .withPreset(3)
    )
    )
}
```

```

        .withRefFramesCount(4)
        .withMaxIframesInterval(5)
        .withBframesCount(4)
        .withHeight(480)
        .withWidth(720))
    //设置音频参数
    .withAudio(new Audio()
        //设置音频编解码格式, 1: AAC, 2: HEAAC1, 3: HEAAC2, 4: MP3
        .withCodec(1)
        //采样
        率,1:AUDIO_SAMPLE_AUTO,2:22050Hz,3:32000Hz,4:44100Hz,5:48000Hz,6:96000Hz
        .withSampleRate(4)
        //音频码率, 单位: kbit/s
        .withBitrate(128)
        //声道数
        .withChannels(2))
    //设置公共参数
    .withCommon(new Common()
        .withDashInterval(5)
        .withHlsInterval(5)
        //高清低码开关
        .withPvc(false)
        //封装类型, 1: HLS, 2: DASH, 3: HLS+DASH, 4: MP4, 5: MP3, 6: ADTS
        .withPackType(1));
    
```

2. 发送更新转码模板请求, 并显示返回消息。

```

//发送更新转码模板请求
UpdateTransTemplateResponse rsp = initMpcClient().updateTransTemplate(req);
//打印返回参数
System.out.println("UpdateTransTemplateResponse=" + JsonUtils.toJSON(rsp));
    
```

代码示例

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.Audio;
import com.huaweicloud.sdk.mpc.v1.model.Common;
import com.huaweicloud.sdk.mpc.v1.model.ModifyTransTemplateReq;
import com.huaweicloud.sdk.mpc.v1.model.UpdateTransTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.UpdateTransTemplateResponse;
import com.huaweicloud.sdk.mpc.v1.model.Video;
import com.obs.services.internal.ServiceException;

public class TestUpdateTranscodeTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
        HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAK(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
    
```

```

        .withCredential(auth)
        .withEndpoint(endpoint)
        .build();
    }

    /**
     * 更新转码模板
     * @param args
     */
    public static void main(String[] args) {
        //设置更新转码模板请求
        UpdateTransTemplateRequest req = new UpdateTransTemplateRequest()
            .withBody(new
ModifyTransTemplateReq().withTemplateName("test_123").withTemplateId(346090L)
            //设置视频参数
            .withVideo(new Video()
                // 视频编码格式, 1表示H264, 2表示H265
                .withCodec(1)
                // 设置视频码率, 单位: kbit/s
                .withBitrate(3200)
                // 编码档次, 建议设为3
                .withProfile(3)
                .withLevel(15)
                // 编码质量, 值越大质量越高, 耗时越长
                .withPreset(3)
                .withRefFramesCount(4)
                .withMaxIframesInterval(5)
                .withBframesCount(4)
                .withHeight(480)
                .withWidth(720))
            //设置音频参数
            .withAudio(new Audio()
                //设置音频编码格式, 1: AAC, 2: HEAAC1, 3: HEAAC2, 4: MP3
                .withCodec(1)
                //采样
                //率,1:AUDIO_SAMPLE_AUTO,2:22050Hz,3:32000Hz,4:44100Hz,5:48000Hz,6:96000Hz
                .withSampleRate(4)
                //音频码率, 单位: kbit/s
                .withBitrate(128)
                //声道数
                .withChannels(2))
            //设置公共参数
            .withCommon(new Common()
                .withDashInterval(5)
                .withHlsInterval(5)
                //高清低码开关
                .withPvc(false)
                //封装类型, 1: HLS, 2: DASH, 3: HLS+DASH, 4: MP4, 5: MP3, 6: ADTS
                .withPackType(1));
        //发送更新转码模板请求
        try {
            UpdateTransTemplateResponse rsp = initMpcClient().updateTransTemplate(req);
            System.out.println("UpdateTransTemplateResponse=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}

```

2.7.4 查询转码模板

查询用户自定义转码模板及系统预置模板。支持指定模板ID或页码查询。具体请求方法请参考[查询转码模板](#)接口。

说明

- 您可通过模板ID查询自定义的单个或者多个转码模板（最多查询10个）。
- 你可以通过page和size进行查询。

核心代码

```
//设置查询转码模板参数，可以查询多个，最多10个
ListTemplateRequest req = new ListTemplateRequest().withTemplateId(Collections.singletonList(346090));
//发送查询转码模板请求
ListTemplateResponse rsp = initMpcClient().listTemplate(req);
//返回查询转码模板结果
System.out.println("httpCode=" + rsp.getHttpStatusCode() + " rsp=" + JsonUtils.toJson(rsp));
```

完整代码

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.ListTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListTemplateResponse;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestListTranscodeTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置，请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk，在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID，在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint，这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 查询转码模板
     * @param args
     */
    public static void main(String[] args) {
        ListTemplateRequest req = new
ListTemplateRequest().withTemplateId(Collections.singletonList(346090));
        try {
            ListTemplateResponse rsp = initMpcClient().listTemplate(req);
            System.out.println("httpCode=" + rsp.getHttpStatusCode() + " rsp=" + JsonUtils.toJson(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
```

```

        System.out.println(e);
    }
}
}

```

2.8 配置水印模板

2.8.1 新建水印模板

水印模板可使转码后的视频自带水印，具体使用方式可以参考[新建水印模板](#)。

核心代码

1. 设置水印模板的参数。

```

//创建水印模板请求
CreateWatermarkTemplateRequest req = new CreateWatermarkTemplateRequest()
    .withBody(new WatermarkTemplate()
        //设置模板名称
        .withTemplateName("watermark_name")
        //设置模板类型
        .withType("Image")
        //设置图片水印处理方式
        .withImageProcess("Grayed")
        //水印宽度
        .withWidth("1920")
        //水印高度
        .withHeight("1080")
        //水印相对视频顶点水平偏移位置
        .withDx("10")
        //水印相对视频顶点垂直偏移位置
        .withDy("10")
        //水印的位置
        //withReferpos("BottomLeft")
        //水印开始时间，与timeline_duration配合使用
        .withTimelineStart("6")
        //水印持续时间，默认值“ToEND”，表示持续到视频结束
        .withTimelineDuration("8"));

```

2. 发送新建水印模板请求，并显示返回消息。

```

// 发送新建水印模板请求给媒体处理服务
CreateWatermarkTemplateResponse rsp = initMpcClient().createWatermarkTemplate(req);
// 打印返回消息
System.out.println("CreateWatermarkTemplateResponse=" + JsonUtils.toJSON(rsp));

```

完整代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.CreateWatermarkTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.CreateWatermarkTemplateResponse;
import com.huaweicloud.sdk.mpc.v1.model.WatermarkTemplate;

public class TestWatermarkTemplate {
    /**
     * 初始化 MpcClient
     *
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置，请根据实际情况设置

```

```

//httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
//    withProxyPassword("xxxxxx");
//根据实际情况填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

String ak = System.getenv("SDK_AK");
String sk = System.getenv("SDK_SK");
//根据实际情况填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
String projectId = System.getenv("PROJECT_ID");
//根据实际情况填写所需endpoint, 这里以“region01”为例
String endpoint = "https://mpc.region01.myhuaweicloud.com";
BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
return
MpcClient.newBuilder().withHttpConfig(httpConfig).withCredential(auth).withEndpoint(endpoint).build();
}

/**
 * 创建水印模板任务
 *
 * @param args
 */
public static void main(String[] args) {
    //创建水印模板请求
    CreateWatermarkTemplateRequest req = new CreateWatermarkTemplateRequest().withBody(new
WatermarkTemplate()
        //设置模板名称
        .withTemplateName("watermark_name")
        //设置模板类型
        .withType("Image")
        //设置图片水印处理方式
        .withImageProcess("Grayed")
        //水印宽度
        .withWidth("1920")
        //水印高度
        .withHeight("1080")
        //水印相对视频顶点水平偏移位置
        .withDx("10")
        //水印相对视频顶点垂直偏移位置
        .withDy("10")
        //水印的位置
        //.withReferpos("BottomLeft")
        //水印开始时间, 与timeline_duration配合使用
        .withTimelineStart("6")
        //水印持续时间, 默认值“ToEND”,表示持续到视频结束
        .withTimelineDuration("ToEND"));
    //发送水印模板请求
    try {
        CreateWatermarkTemplateResponse rsp = initMpcClient().createWatermarkTemplate(req);
        System.out.println("CreateWatermarkTemplateResponse=" + JsonUtils.toJson(rsp));
    } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
        System.out.println(e);
    }
}
}
}

```

2.8.2 更新水印模板

核心代码

1. 设置水印模板的参数。

```

//创建更新水印模板请求
UpdateWatermarkTemplateRequest req = new UpdateWatermarkTemplateRequest()
    .withBody(new WatermarkTemplate()
        //设置模板名称
        .withTemplateName("watermark_name")
        //设置模板类型
        .withType("Image")
        //设置图片水印处理方式

```

```

.withImageProcess("Grayed")
//水印宽度
.withWidth("1920")
//水印高度
.withHeight("1080")
//水印相对视频顶点水平偏移位置
.withDx("10")
//水印相对视频顶点垂直偏移位置
.withDy("10")
//水印的位置
//.withReferpos("BottomLeft")
//水印开始时间, 与timeline_duration配合使用
.withTimelineStart("0")
//水印持续时间, 默认值 "ToEND",表示持续到视频结束
.withTimelineDuration("ToEND");

```

2. 发送更新水印模板请求, 并显示返回消息。

```

// 发送修改水印配置请求给媒体处理服务
UpdateWatermarkTemplateResponse rsp = initMpcClient().updateWatermarkTemplate(req);
// 打印返回消息
System.out.println("UpdateWatermarkTemplateResponse=" + JsonUtils.toJSON(rsp));

```

完整代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.util.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.UpdateWatermarkTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.UpdateWatermarkTemplateResponse;
import com.huaweicloud.sdk.mpc.v1.model.WatermarkTemplate;

public class TestUpdateWatermarkTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }
}

/**
 * 更新水印模板
 * @param args
 */
public static void main(String[] args) {
    //创建更新水印模板请求
    UpdateWatermarkTemplateRequest req = new UpdateWatermarkTemplateRequest()
        .withBody(new WatermarkTemplate()
            //设置模板名称
            .withTemplateName("watermark_name")
            //设置模板类型

```

```

        .withType("Image")
        //设置图片水印处理方式
        .withImageProcess("Grayed")
        //水印宽度
        .withWidth("1920")
        //水印高度
        .withHeight("1080")
        //水印相对视频顶点水平偏移位置
        .withDx("10")
        //水印相对视频顶点垂直偏移位置
        .withDy("10")
        //水印的位置
        //.withReferpos("BottomLeft")
        //水印开始时间, 与timeline_duration配合使用
        .withTimelineStart("0")
        //水印持续时间, 默认值 "ToEND",表示持续到视频结束
        .withTimelineDuration("ToEND");
//发送水印模板请求
try {
    UpdateWatermarkTemplateResponse rsp = initMpcClient().updateWatermarkTemplate(req);
    System.out.println("UpdateWatermarkTemplateResponse=" + JsonUtils.toJSON(rsp));
} catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
    System.out.println(e);
}
}
}
}

```

2.8.3 查询水印配置模板

代码说明

- 查询水印配置模板任务支持指定模板ID查询，也支持分页查询。
- 指定模板ID查询，一次最多支持查询10个模板ID。
- 分页查询需指定页数和每页的模板数；若不带参数，服务自动取page为0，size为10。

核心代码

1. 设置查询参数。

a. 根据水印模板ID查询。

```
ListWatermarkTemplateRequest req = new
ListWatermarkTemplateRequest().withTemplateId(Collections.singletonList(215728));
```

b. 根据页数查询。

根据page和size进行分页查询

```
ListWatermarkTemplateRequest req = new
ListWatermarkTemplateRequest().withPage(1).withSize(10);
```

2. 发送查询请求，并显示返回消息。

// 发送查询水印模板请求给媒体处理服务

```
ListWatermarkTemplateResponse rsp = initMpcClient().listWatermarkTemplate(req);
```

// 打印返回消息

```
System.out.println("rsp=" + JsonUtils.toJSON(rsp));
```

完整代码

```

import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;

```

```
import com.huaweicloud.sdk.mpc.v1.model.ListWatermarkTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.ListWatermarkTemplateResponse;
import com.obs.services.internal.ServiceException;

import java.util.Collections;

public class TestListWatermarkTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(yyyyyy).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 查询水印模板
     * @param args
     */
    public static void main(String[] args) {
        ListWatermarkTemplateRequest req = new
ListWatermarkTemplateRequest().withTemplateId(Collections.singletonList(215728));
        try {
            ListWatermarkTemplateResponse rsp = initMpcClient().listWatermarkTemplate(req);
            System.out.println("rsp=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}
```

2.8.4 删除水印模板

通过指定水印模板ID删除用户自定义的水印模板。

核心代码

```
// 发送删除水印配置模板请求给媒体处理服务
DeleteWatermarkTemplateRequest req = new DeleteWatermarkTemplateRequest().withTemplateId(215728);
DeleteWatermarkTemplateResponse rsp = initMpcClient().deleteWatermarkTemplate(req);
// 打印返回消息
System.out.println("httpCode=" + rsp.getHttpStatusCode() + ", rsp=" + JsonUtils.toJSON(rsp));
```

代码示例

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ClientRequestException;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```

import com.huaweicloud.sdk.core.http.HttpConfig;
import com.huaweicloud.sdk.core.utils.JsonUtils;
import com.huaweicloud.sdk.mpc.v1.MpcClient;
import com.huaweicloud.sdk.mpc.v1.model.DeleteWatermarkTemplateRequest;
import com.huaweicloud.sdk.mpc.v1.model.DeleteWatermarkTemplateResponse;
import com.obs.services.internal.ServiceException;

public class TestDeleteWatermarkTemplate {
    /**
     * 初始化 MpcClient
     * @return
     */
    public static MpcClient initMpcClient() {
        HttpConfig httpConfig =
HttpConfig.getDefaultHttpConfig().withIgnoreSSLVerification(true).withTimeout(3);
        //http代理设置, 请根据实际情况设置
        //httpConfig.withProxyHost("xxxxxx").withProxyPort(xxxxxx).withProxyUsername("xxxxxx").
        //    withProxyPassword("xxxxxx");
        //根据实际填写ak,sk, 在控制台账号名下“我的凭证 > 访问密钥”上创建和查看您的AK/SK

        String ak = System.getenv("SDK_AK");
        String sk = System.getenv("SDK_SK");
        //根据实际填写项目ID, 在控制台账号名下“我的凭证 > API凭证”下查看您的项目ID
        String projectId = System.getenv("PROJECT_ID");
        //根据实际填写所需endpoint, 这里以“region01”为例
        String endpoint = "https://mpc.region01.myhuaweicloud.com";
        BasicCredentials auth = new BasicCredentials().withAk(ak).withSk(sk).withProjectId(projectId);
        return MpcClient.newBuilder()
            .withHttpConfig(httpConfig)
            .withCredential(auth)
            .withEndpoint(endpoint)
            .build();
    }

    /**
     * 删除水印模板任务
     * @param args
     */
    public static void main(String[] args) {
        DeleteWatermarkTemplateRequest req = new
DeleteWatermarkTemplateRequest().withTemplateId(215728);
        try {
            DeleteWatermarkTemplateResponse rsp = initMpcClient().deleteWatermarkTemplate(req);
            System.out.println("httpCode=" + rsp.getHttpStatusCode() + ", rsp=" + JsonUtils.toJSON(rsp));
        } catch (ClientRequestException | ConnectionException | RequestTimeoutException | ServiceException
e) {
            System.out.println(e);
        }
    }
}

```

2.9 SDK & API 对应关系

表 2-3 SDK 与 API 对应关系

接口	API	说明
createTranscodingTask	POST /v1/{project_id}/transcodings	新建转码任务
deleteTranscodingTask	DELETE /v1/{project_id}/transcodings/{task_id}	取消转码任务

接口	API	说明
listTranscodingTask	GET /v1/{project_id}/transcodings{?task_id}	查询转码任务
createTransTemplate	POST /v1/{project_id}/template/transcodings	新建转码模板
deleteTemplate	DELETE /v1/{project_id}/template/transcodings{?temp_id}	删除转码模板
updateTransTemplate	PUT /v1/{project_id}/template/transcodings	更新转码模板
listTemplate	GET /v1/{project_id}/template/transcodings{?temp_id}	查询转码模板
createThumbnailTask	POST /v1/{project_id}/thumbnails	新建截图任务
deleteThumbnailTask	DELETE /v1/{project_id}/thumbnails{?task_id}	取消截图任务
listThumbnailTask	GET /v1/{project_id}/thumbnails{?task_id,start_time,end_time,status,page,size}	查询截图任务
createWatermarkTemplate	POST /v1/{project_id}/template/watermark	新建水印模板
updateWatermarkTemplate	PUT /v1/{project_id}/template/watermark	更新水印模板
listWatermarkTemplate	GET /v1/{project_id}/template/watermark{?template_id,page,size}	查询水印模板
deleteWatermarkTemplate	DELETE /v1/{project_id}/template/watermark{?template_id}	删除水印模板
createAnimatedGraphicsTask	POST /v1/{project_id}/animated-graphics	新建动图任务
listAnimatedGraphicsTask	GET /v1/{project_id}/animated-graphics{?task_id,start_time,end_time,status,page,size}	查询动图任务
deleteAnimatedGraphicsTask	DELETE /v1/{project_id}/animated-graphics?task_id={task_id}	取消动图任务
createExtractTask	POST /v1/{project_id}/extract-metadata	创建视频解析任务
listExtractTask	GET /v1/{project_id}/extract-metadata{?task_id,start_time,end_time,status,page,size}	查询视频解析任务
deleteExtractTask	DELETE /v1/{project_id}/extract-metadata{?task_id}	取消视频解析任务

接口	API	说明
creatRemuxTask	POST /v1/{project_id}/remux	创建转封装任务
listRemuxTask	GET /v1/{project_id}/remux{?task_id,start_time,end_time,status,page,size}	查询转封装任务
cancelRemuxTask	DELETE /v1/{project_id}/remux{?task_id}	取消转封装任务

3 Python SDK

本章节介绍了Python SDK的使用说明，您可以参考本章节进行快速集成开发。

开发前准备

- 已[注册](#)华为账号并开通华为云，已进行[实名认证](#)。
- 已具备开发环境，支持python 3及以上版本。
- 已获取账号对应的Access Key (AK) 和Secret Access Key (SK)。请在控制台“我的凭证 > 访问密钥”页面上创建和查看您的AK/SK。具体请参见[访问密钥](#)。
- 已获取转码服务对应区域的项目ID，请在控制台“我的凭证 > API凭证”页面上查看项目ID。具体请参见[API凭证](#)。
- 已将需要处理的媒资文件上传至MPC同区域的OBS桶中，并将OBS桶进行授权，允许MPC访问。具体请参见[上传音视频文件](#)和[获取云资源授权](#)。

安装 SDK

媒体转码服务端SDK支持python 3及以上版本。执行“python --version”检查当前python的版本信息。

使用服务端SDK前，您需要安装“huaweicloudsdkcore”和“huaweicloudsdkmpc”，具体的SDK版本号请参见[SDK开发中心](#)。

- 使用pip安装

执行如下命令安装Python SDK核心库以及相关服务库：

```
# 安装核心库
pip install huaweicloudsdkcore
# 安装MPC服务库
pip install huaweicloudsdkmpc
```

- 使用源码安装

执行如下命令安装Python SDK核心库以及相关服务库：

```
# 安装核心库
cd huaweicloudsdkcore-${version}
python setup.py install
```

```
# 安装MPC服务库
cd huaweicloudsdkmpc-${version}
python setup.py install
```

开始使用

步骤1 导入依赖模块。

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials, GlobalCredentials
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkcore.http.http_config import HttpConfig
# 导入指定MPC的库
from huaweicloudsdkmpc.v1 import *
```

步骤2 配置客户端属性。

1. 默认配置。
使用默认配置
config = HttpConfig.get_default_config()
2. 代理配置（可选）。
user_name = os.environ["USER_NAME"]
user_password = os.environ["USER_PASSWORD"]
使用代理服务器（可选）
config.proxy_protocol = 'http'
config.proxy_host = 'proxy.huaweicloud.com'
config.proxy_port = 80
config.proxy_user = user_name
config.proxy_password = user_password
3. 连接配置（可选）。
配置连接超时（可选），支持统一指定超时时长timeout=timeout，或分别指定超时时长
timeout=(connect timeout, read timeout)
config.timeout = 3
4. SSL配置（可选）。
配置跳过服务端证书验证（可选）
config.ignore_ssl_verification = True
配置服务器端CA证书，用于SDK验证服务端证书合法性
config.ssl_ca_cert = ssl_ca_cert

步骤3 初始化认证信息。

支持两种方式认证，您可以根据实际情况进行选择。

- 使用永久AK/SK

首先需要获取永久AK和SK，以及projectId，您可以参考[开发前准备](#)获取。

```
ak = os.environ["SDK_AK"]
sk = os.environ["SDK_SK"]
project_id = os.environ["PROJECT_ID"]
credentials = BasicCredentials(ak, sk, project_id)
```

- 使用临时AK/SK

首先需要获得临时AK、SK和SecurityToken，您可以[通过token获取](#)或者[通过委托授权获取](#)。

```
ak = os.environ["SDK_AK"]
sk = os.environ["SDK_SK"]
project_id = os.environ["PROJECT_ID"]
security_token = os.environ["SECURITY_TOKEN"]
credentials = BasicCredentials(ak, sk, project_id).with_security_token(security_token)
```

相关参数说明如下所示：

- **ak**：账号Access Key。
- **sk**：账号Secret Access Key。
- **project_id**：云服务所在项目ID，根据您需要操作的项目所属区域选择对应的项目ID。
- **security_token**：采用临时AK/SK认证场景下的安全票据。

步骤4 初始化客户端。

```
# 初始化转码服务的客户端
client = MpcClient.new_builder(MpcClient) \
    .with_http_config(config) \
    .with_credentials(credentials) \
    .with_endpoint(endpoint) \ # endpoint值如 "https://mpc.region01.myhuaweicloud.com"
    .with_file_log(path="test.log", log_level=logging.INFO) \ # 日志打印至文件
    .with_stream_log(log_level=logging.INFO) \ # 日志打印至控制台
    .build()
```

- **endpoint**: MPC应用区域和各服务的终端节点，具体请参见[地区和终端节点](#)。
- **with_file_log**支持如下配置：
 - path: 日志文件路径。
 - log_level: 日志级别，默认INFO。
 - max_bytes: 单个日志文件大小，默认为10485760 bytes。
 - backup_count: 日志文件个数，默认为5个。
- **with_stream_log**支持如下配置：
 - stream: 流对象，默认sys.stdout。
 - log_level: 日志级别，默认INFO。

打开日志开关后，每次请求将打印访问日志，格式如下：

```
'%(asctime)s %(thread)d %(name)s %(filename)s %(lineno)d %(levelname)s %(message)s'
```

步骤5 发送请求并查看响应。

```
// 初始化请求，以调用查询转码任务接口为例
request = ListTranscodingTaskRequest(task_id)
response = client.list_transcoding_task(request)
print(response)
```

步骤6 异常处理。

表 3-1 异常处理

一级分类	一级分类说明	二级分类	二级分类说明
ConnectionException	连接类异常	HostUnreachableException	网络不可达、被拒绝
		SslHandShakeException	SSL认证异常
RequestTimeoutException	响应超时异常	CallTimeoutException	单次请求，服务器处理超时未返回
		RetryOutageException	在重试策略消耗完成以后，仍无有效的响应
ServiceResponseException	服务器响应异常	ServerResponseException	服务端内部错误，Http响应码：[500,]
		ClientRequestException	请求参数不合法，Http响应码：[400, 500)

```
# 异常处理
try:
```

```
request = request = ListTranscodingTaskRequest(task_id = [1900293])
response = client.list_transcoding_task(request)
except exception.ServiceResponseException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

步骤7 原始Http侦听器。

在某些场景下可能对业务发出的Http请求进行Debug，需要看到原始的Http请求和返回信息，SDK提供侦听器功能获取原始的和加密的Http请求和返回信息。

注意

原始信息打印仅在debug阶段使用，请不要在生产系统中将原始的Http头和Body信息打印到日志，这些信息并未加密且其中包含敏感数据；当Body体为二进制内容，即Content-Type标识为二进制时 body为"****"，详细内容不输出。

```
def response_handler(**kwargs):
    logger = kwargs.get("logger")
    response = kwargs.get("response")
    request = response.request

    base = "> Request %s %s HTTP/1.1" % (request.method, request.path_url) + "\n"
    if len(request.headers) != 0:
        base = base + "> Headers:" + "\n"
        for each in request.headers:
            base = base + "    %s : %s" % (each, request.headers[each]) + "\n"
        base = base + "> Body: %s" % request.body + "\n\n"

    base = base + "< Response HTTP/1.1 %s " % response.status_code + "\n"
    if len(response.headers) != 0:
        base = base + "< Headers:" + "\n"
        for each in response.headers:
            base = base + "    %s : %s" % (each, response.headers[each],) + "\n"
        base = base + "< Body: %s" % response.content
    logger.debug(base)

MpcClient client = MpcClient.new_builder(MpcClient)\
    .with_http_config(config) \
    .with_credentials(credentials) \
    .with_endpoint(endpoint) \
    .with_file_log(path="test.log", log_level=logging.INFO) \
    .with_stream_log(log_level=logging.INFO) \
    .with_http_handler(Handler().add_response_handler(response_handler)) \
```

Handler支持“add_request_handler”和“add_response_handler”方法。

----结束

代码示例

调用前请根据实际情况替换如下变量：“SDK_AK”、“SDK_SK”、{your endpoint string} 以及 {your project id}。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkcore.http.http_config import HttpConfig
from huaweicloudsdkmpc.v1 import *
```

```
def list_transcoding_task(client):
    try:
        request = ListTranscodingTaskRequest(task_id = [1900293])
        response = client.list_transcoding_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

if __name__ == "__main__":
    ak = os.environ["SDK_AK"]
    sk = os.environ["SDK_SK"]
    project_id = os.environ["{your project id}"]
    endpoint = "{your endpoint}"

    config = HttpConfig.get_default_config()
    config.ignore_ssl_verification = True
    credentials = BasicCredentials(ak, sk, project_id)

    mpc_client = MpcClient.new_builder(MpcClient) \
        .with_http_config(config) \
        .with_credentials(credentials) \
        .with_endpoint(endpoint) \
        .build()

    list_transcoding_task(mpc_client)
```

4 Go SDK

本章节介绍了Go SDK的使用说明，您可以参考本章节进行快速集成开发。

开发前准备

- 已[注册](#)华为账号并开通华为云，已进行[实名认证](#)。
- 具体开发环境，支持go 1.14及以上版本。
- 已获取账号对应的Access Key (AK) 和Secret Access Key (SK)。请在控制台“我的凭证 > 访问密钥”页面上创建和查看您的AK/SK。具体请参见[访问密钥](#)。
- 已获取转码服务对应区域的项目ID，请在控制台“我的凭证 > API凭证”页面上查看项目ID。具体请参见[API凭证](#)。
- 已将需要处理的媒资文件上传至MPC同区域的OBS桶中，并将OBS桶进行授权，允许MPC访问。具体请参见[上传音视频文件](#)和[获取云资源授权](#)。

安装 SDK

媒体转码Go SDK支持go 1.14及以上版本。执行go version检查当前Go的版本信息。

使用go get安装Go SDK，执行如下命令安装Go SDK库以及相关依赖库，具体的SDK版本号请参见[SDK开发中心](#)。

```
# 安装Go库
go get github.com/huaweicloud/huaweicloud-sdk-go-v3
# 安装依赖
go get github.com/json-iterator/go
```

开始使用

步骤1 导入依赖模块。

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/config"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/http/handler"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/mpc/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/mpc/v1/model"
    "net/http"
)
```

步骤2 配置客户端属性。

1. 默认配置

```
# Use default configuration
httpConfig := config.DefaultHttpConfig()
```

2. 代理配置（可选）。

```
username := os.Getenv("USER_NAME")
password := os.Getenv("USER_PASSWORD")
// 根据需要配置网络代理
httpConfig.WithProxy(config.NewProxy().
    WithSchema("http").
    WithHost("proxy.huaweicloud.com").
    WithPort(80).
    WithUsername(username).
    WithPassword(password))
```

3. SSL配置（可选）

```
// 根据需要配置是否跳过SSL证书校验
httpConfig.WithIgnoreSSLVerification(true);
```

步骤3 初始化认证信息。

支持两种方式认证，您可以根据实际情况进行选择。

- 使用永久AK/SK

首先需要获取永久AK和SK，以及projectId，您可以参考[开发前准备](#)获取。

```
ak := os.Getenv("SDK_AK")
sk := os.Getenv("SDK_SK")
projectId := os.Getenv("PROJECT_ID")
auth := basic.NewCredentialsBuilder().
    WithAk(ak).
    WithSk(sk).
    WithProjectId(projectId).
    Build()
```

- 使用临时AK/SK

首先需要获得临时AK、SK和SecurityToken，您可以[通过token获取](#)或者[通过委托授权获取](#)。

```
ak := os.Getenv("SDK_AK")
sk := os.Getenv("SDK_SK")
projectId := os.Getenv("PROJECT_ID")
securityToken := os.Getenv("SECURITY_TOKEN")
auth := basic.NewCredentialsBuilder().
    WithAk(ak).
    WithSk(sk).
    WithProjectId(projectId).
    WithSecurityToken(securityToken).
    Build()
```

相关参数说明如下所示：

- **ak**：账号Access Key。
- **sk**：账号Secret Access Key。
- **projectId**：云服务所在项目ID，根据你想操作的项目所属区域选择对应的项目ID。
- **securityToken**：采用临时AK/SK认证场景下的安全票据。

步骤4 初始化客户端。

```
# 初始化MPC的客户端
client := mpc.NewMpcClient
(
    mpcMpcClientBuilder().
    WithEndpoint(endpoint). // endpoint值如 "https://mpc.region01.myhuaweicloud.com"
    WithCredential(auth).
    WithHttpConfig(config.DefaultHttpConfig()).
    Build())
```

endpoint: MPC应用区域和各服务的终端节点，具体请参见[地区和终端节点](#)。

步骤5 发送请求并查看响应。

```
// 初始化请求，以调用接口查询转码模板为例
request := &model.ListTranscodingTaskRequest{
    TaskId:&[]int64{1900293},
}
response, err := client.ListTranscodingTask(request)
if err == nil {
    fmt.Printf("%+v\n",response)
} else {
    fmt.Println(err)
}
```

步骤6 异常处理。

表 4-1 异常处理

一级分类	一级分类说明
ServiceResponseError	service response error
url.Error	connect endpoint error

```
# 异常处理
response, err := client.ListTranscodingTask(request)
if err == nil {
    fmt.Println(response)
} else {
    fmt.Println(err)
}
```

步骤7 原始Http侦听器。

在某些场景下可能对业务发出的Http请求进行Debug，需要看到原始的Http请求和返回信息，SDK提供侦听器功能来获取原始的为加密的Http请求和返回信息。

⚠ 注意

原始信息打印仅在debug阶段使用，请不要在生产系统中将原始的Http头和Body信息打印到日志，这些信息并未加密且其中包含敏感数据；当Body体为二进制内容，即Content-Type标识为二进制时 body为"****"，详细内容不输出。

```
func RequestHandler(request http.Request) {
    fmt.Println(request)
}

func ResponseHandler(response http.Response) {
    fmt.Println(response)
}

ak := os.Getenv("SDK_AK")
sk := os.Getenv("SDK_SK")
projectId := os.Getenv("{your project id}")
client := mpc.NewMpcClient(
    mpc.MpcClientBuilder().
        WithEndpoint("{your endpoint}").
        WithCredential(
            basic.NewCredentialsBuilder().
                WithAk(ak).
```

```

        WithSk(sk).
        WithProjectId(projectId).
        Build()).
    WithHttpConfig(config.DefaultHttpConfig().
        WithIgnoreSSLVerification(true).
        WithHandler(handler.
            NewHandler().
                AddRequestHandler(RequestHandler).
                AddResponseHandler(ResponseHandler))).Build()

```

---结束

代码示例

调用前请根据实际情况替换如下变量："SDK_AK"、"SDK_SK"、{your endpoint string}以及{your project id}。

```

package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/config"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/httpandler"
    mpc "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/mpc/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/mpc/v1/model"
    "net/http"
)

func RequestHandler(request http.Request) {
    fmt.Println(request)
}

func ResponseHandler(response http.Response) {
    fmt.Println(response)
}

ak := os.Getenv("SDK_AK")
sk := os.Getenv("SDK_SK")
projectId := os.Getenv("{your project id}")
func main() {
    client := mpc.NewMpcClient(
        mpc.MpcClientBuilder().
            WithEndpoint("{your endpoint}").
            WithCredential(
                basic.NewCredentialsBuilder().
                    WithAk(ak).
                    WithSk(sk).
                    WithProjectId(projectId).
                    Build()).
            WithHttpConfig(config.DefaultHttpConfig().
                WithIgnoreSSLVerification(true).
                WithHandler(handler.
                    NewHandler().
                        AddRequestHandler(RequestHandler).
                        AddResponseHandler(ResponseHandler))).
            Build())

    request := &model.ListTranscodingTaskRequest{
        TaskId:&[]int64{1900293},
    }
    response, err := client.ListTranscodingTask(request)
    if err == nil {
        fmt.Println("%+v\n",response)
    } else {
        fmt.Println(err)
    }
}

```

5 附录

5.1 JDK 安装

本SDK包要求的JDK版本高于JDK8版本，以下步骤以win7环境配置JDK8 64位为例，若已经下载JDK并配置好环境请忽略本章节。

操作步骤

步骤1 [官网](#)下载JDK文件。以JDK8为例，单击JDK下的下载按钮进行下载。

Java SE 8u191 / Java SE 8u192

Java SE 8u191 / Java SE 8u192 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

[Learn more](#) ▶

- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme Files](#)
 - [JDK ReadMe](#)
 - [JRE ReadMe](#)

JDK
DOWNLOAD ↓

Server JRE
DOWNLOAD ↓

JRE
DOWNLOAD ↓

步骤2 下载完成后按照提示安装，安装位置可自选，比如安装到本地“C:\Program Files\Java\jdk1.8.0_131”。

步骤3 安装完成后，配置Java环境变量。

1. 右击“计算机”，单击“属性”，选择“高级系统设置”；
2. 选择“高级”选项卡，单击“环境变量”；
3. 在“系统变量”中设置3个变量：JAVA_HOME、PATH、CLASSPATH（大小写均可），变量值如表5-1所示。

若此三项属性已存在则单击"编辑", 不存在则单击"新建"。

表 5-1 JAVA 环境变量

变量名	变量值	变量说明
JAVA_HOME	JDK安装的实际路径	例如: "C:\Program Files (x86)\Java\jdk1.8.0_131"
PATH	%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin	在原PATH值后添加
CLASSPATH	.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;	注意前面有个"."

步骤4 打开命令行窗口, 输入“java -version”。显示Java版本信息即表示配置成功。

以JDK 8为例, 成功示例图如下:

```
C:\>java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

---结束

5.2 错误码

当您调用API时, 如果遇到“APIGW”开头的错误码, 请参见[API网关错误码](#)进行处理。

状态码	错误码	错误信息	描述	处理措施
400	MPC.10089	The template file does not exist.	模板文件不存在	模板文件不存在, 请检查
400	MPC.10090	The template file does not exist.	模板文件不存在	模板文件不存在, 请检查
400	MPC.10091	The template name already exists.	模板名字重复	模板名字重复, 请检查
400	MPC.10155	The subtitle file list and M3U8 file do not contain the specified default language.	字幕文件列表和M3U8文件中均不存在指定默认语言	字幕文件列表和M3U8文件中均不存在指定默认语言

状态码	错误码	错误信息	描述	处理措施
400	MPC.10156	File deletion failed.	文件删除失败	文件删除失败
400	MPC.10202	Invalid request parameter.	MPC 请求参数非法	请检查参数是否正确
400	MPC.10204	Incorrect request method.	MPC 请求方式不正确	请检查请求方式
400	MPC.10205	Incorrect request content type.	MPC 请求内容类型不正确	请检查请求内容类型
400	MPC.10223	An agency has been created.	委托授权已创建	委托授权已创建，请检查
400	MPC.10224	The agency has been deleted.	委托授权已被删除	委托授权已被删除，请检查
400	MPC.10230	The template group already exists.	模板组已存在	模板组已存在，请检查
400	MPC.10231	The template group does not exist.	模板组不存在	模板组不存在，请检查
401	MPC.10203	Identity authentication failed.	MPC 认证失败	请检查Token等认证参数是否正确
401	MPC.10206	You have not completed real-name authentication.	MPC 用户未实名认证	请检查用户是否实名认证
401	MPC.10207	Your account is in an abnormal state.	MPC 用户处于异常状态	请检查用户状态是否正确
401	MPC.10208	Tenant ID verification failed, please check.	MPC 租户ID校验失败	请检查租户ID是否正确
403	MPC.10010	Invalid input parameters.	输入参数错误	输入参数错误，请检查。

状态码	错误码	错误信息	描述	处理措施
403	MPC.10211	The task does not exist.	任务不存在	转码任务不存在，请检查
403	MPC.10212	Operation failed. The task is in progress or has been completed.	MPC 操作失败，任务处理中或已经处理完成	任务处理中或已经处理完成，请检查
403	MPC.10214	The topic does not exist.	MPC 主题不存在	主题不存在，请检查
403	MPC.10215	The topic already exists.	MPC 主题已经存在	主题已经存在，请检查
403	MPC.10226	The resource does not exist.	资源不存在	资源不存在，请检查
403	MPC.10240	Failed to obtain the basic information about the media file.	获取媒体文件基本信息失败。	请检查是否已授权桶，且obs桶未配置kms加密。
403	MPC.10243	Due to security reasons, your account has been restricted from purchasing certain pay-per-use cloud service resources according to the CLOUD Customer Agreement. If you have any questions, contact customer service.	您的账号可能存在风险，已根据《云用户协议》限制您购买部分按需计费的云服务资源。如有疑问，请联系客服咨询处理。	您的账号可能存在风险，已根据《云用户协议》限制您购买部分按需计费的云服务资源。如有疑问，请联系客服咨询处理。

状态码	错误码	错误信息	描述	处理措施
403	MPC.10244	Insufficient account balance. Top up your account.	您的账户余额不足，无法开通资源，请立即充值。	您的账户余额不足，无法开通资源，请立即充值。
406	MPC.10051	The selected template is a super-resolution template and is not supported.	所选模板为超分辨率模板，暂不支持	请检查转码模板是否正确
406	MPC.10052	Failed to obtain the input file.	无法获取源文件，请检查路径	无法获取源文件，请检查路径
406	MPC.10053	The input file does not exist.	源文件不存在	源文件不存在，请检查
406	MPC.10054	Failed to obtain the subtitle file.	无法获取字幕文件	无法获取字幕文件，请检查路径
406	MPC.10055	The audio sampling rate 7,350 is not supported.	输出音频AAC时，不支持采样率为7350，请修改	不支持采样率为7350，请修改
406	MPC.10056	This type of output frame rate is not supported.	不支持该类输出帧率	不支持该类输出帧率，请修改
406	MPC.10057	This type of output bitrate is not supported.	不支持该类输出视码率	不支持该类输出视码率，请修改
406	MPC.10058	This type of output video width is not supported.	不支持该类输出视频宽度	不支持该类输出视频宽度，请修改
406	MPC.10059	This type of output video height is not supported.	不支持该类输出视频高度	不支持该类输出视频高度，请修改

状态码	错误码	错误信息	描述	处理措施
406	MPC.10060	This type of I-frame interval is not supported.	不支持该类输出视频帧间隔	不支持该类输出视频帧间隔，请修改
406	MPC.10061	Capturing snapshots at non-fixed intervals is not supported.	不支持非固定时间间隔截图，请修改为固定时间间隔截图	请修改为固定时间间隔截图
406	MPC.10062	Invalid video codec.	截图场景下视频codec错误	截图场景下视频codec错误，请修改
406	MPC.10063	Invalid video format.	截图场景下视频format错误	截图场景下视频format错误，请修改
406	MPC.10064	Multiple watermarks are not supported.	不支持水印多路输入，目前仅支持2路输入	不支持水印多路输入，目前仅支持2路输入，请修改
406	MPC.10065	Invalid output file format.	不支持该类输出文件格式	不支持该类输出文件格式，请检查
406	MPC.10066	The input file format does not match the actual format.	输入文件格式与实际格式不符，请检查	输入文件格式与实际格式不符，请检查
406	MPC.10067	Failed to obtain the ID of the video codec.	获取视频CODEC ID 失败	获取视频CODEC ID 失败，请检查
406	MPC.10068	Failed to obtain the ID of the audio codec.	获取音频CODEC ID 失败	获取音频CODEC ID 失败，请检查
406	MPC.10069	Failed to obtain the ID of the subtitle codec.	获取字幕CODEC ID 失败	获取字幕CODEC ID 失败，请检查
406	MPC.10070	Failed to obtain the encoding/decoding format.	获取编解码格式失败	获取编解码格式失败，请检查

状态码	错误码	错误信息	描述	处理措施
406	MPC.10071	Failed to obtain the parameters of the input video stream.	无法获取输入视频流的相关参数信息	无法获取输入视频流的相关参数信息，请检查
406	MPC.10072	Invalid frame rate of the video stream.	视频流帧率信息错误	视频流帧率信息错误，请检查
406	MPC.10080	Invalid frame rate of the input file.	输入文件视频帧率错误	输入文件视频帧率错误，请检查
406	MPC.10081	The file does not contain audio streams.	文件缺失音频流，请检查输入文件	文件缺失音频流，请检查输入文件
406	MPC.10082	Failed to obtain the input audio or video stream.	无法获取输入视频流（音频或视频），请自检	无法获取输入视频流（音频或视频），请检查
406	MPC.10083	This type of codec is not supported.	不支持该类编码类型	不支持该类编码类型，请检查
406	MPC.10084	This chroma subsampling format is not supported.	不支持此种色度采样格式	不支持此种色度采样格式，请检查
406	MPC.10085	The file format is not supported.	文件格式不支持	文件格式不支持，请检查
406	MPC.10086	Failed to obtain the input file.	无法获取源文件，请检查路径	无法获取源文件，请检查路径
406	MPC.10087	Invalid task parameters.	查询任务参数错误	查询任务参数错误，请检查
406	MPC.10088	The image file does not exist.	图片文件不存在	图片文件不存在，请检查
406	MPC.10092	The image file does not exist.	图片文件不存在	图片文件不存在，请检查

状态码	错误码	错误信息	描述	处理措施
406	MPC.10093	The file name exceeds the maximum length.	文件名过长	文件名过长, 请检查
406	MPC.10094	Invalid file format.	文件格式异常	文件格式异常, 请检查
406	MPC.10095	The watermark is placed in a wrong position.	水印的位置错误	水印的位置错误, 请检查
406	MPC.10096	Invalid watermark size.	水印的大小错误	水印的大小错误, 请检查
406	MPC.10097	Invalid watermark scaling ratio.	水印缩放比例错误	水印缩放比例错误, 请检查
406	MPC.10098	Invalid watermark duration.	水印持续时长错误	水印持续时长错误, 请检查
406	MPC.10099	The media stream type is not supported.	不支持的媒体流类型	不支持的媒体流类型, 请检查
406	MPC.10100	An error occurred when parsing the video frame rate information.	解析视频帧率信息错误	解析视频帧率信息错误, 请检查
406	MPC.10101	Invalid input parameters.	输入参数错误	输入参数错误, 请检查
406	MPC.10102	Failed to open the input file.	源文件打开异常, 请检查	源文件打开异常, 请检查
406	MPC.10103	Open GOP is not supported.	不支持OPEN GOP素材	不支持OPEN GOP素材, 请检查
406	MPC.10104	Internal error.	服务内部异常, 请重试或联系工程师	服务内部异常, 请重试或联系工程师

状态码	错误码	错误信息	描述	处理措施
406	MPC.10105	An error occurred during transcoding.	转码进程异常	转码进程异常，请重试或联系工程师
406	MPC.10106	The audio sampling rate is lower than 12,000. The audio will be discarded.	音频采样率低于12000，音频被丢弃	音频采样率低于12000，音频被丢弃
406	MPC.10107	Invalid input video resolution.	原始视频分辨率错误	原始视频分辨率错误
406	MPC.10108	The audio sampling rate of the input video is incorrect.	原始视频的音频采样率错误	原始视频的音频采样率错误
406	MPC.10109	Invalid resolution in the template.	模板分辨率错误	模板分辨率错误
406	MPC.10110	The video encoding format of the input file is not supported.	片源视频编码格式特殊，暂不支持	片源视频编码格式特殊，暂不支持
406	MPC.10111	Failed to obtain the file from OBS.	获取obs文件失败	获取obs文件失败
406	MPC.10112	The video or audio format of the input file is not supported.	片源的视频或音频格式不支持	片源的视频或音频格式不支持
406	MPC.10113	The DTS of the input file is not supported.	片源的dts异常，暂不支持	片源的dts异常，暂不支持
406	MPC.10114	The header information of the input file is incorrect.	片源文件头信息有误，请检查	片源文件头信息有误，请检查

状态码	错误码	错误信息	描述	处理措施
406	MPC.10115	The watermark cannot be scaled down by more than 256 times.	水印图片缩小倍数超过256倍,暂不支持	水印图片缩小倍数超过256倍,暂不支持
406	MPC.10116	The audio encoding format of the input file is not supported.	片源的音频编码格式, 暂不支持	片源的音频编码格式, 暂不支持
406	MPC.10117	The audio and video in the input file are not synchronized.	片源的音频和视频不同步	片源的音频和视频不同步
406	MPC.10118	Failed to upload files to the OBS path.	上传文件到obs失败	上传文件到obs失败, 请重试或联系工程师
406	MPC.10119	Invalid input data.	片源数据无效	片源数据无效, 请检查
406	MPC.10120	The task does not exist.	任务不存在	任务不存在, 请检查
406	MPC.10121	The subtitle file does not exist.	字幕文件不存在, 请检查	字幕文件不存在, 请检查
406	MPC.10122	The resolution in the template is greater than the input video resolution.	模板分辨率大于原视频分辨率	模板分辨率大于原视频分辨率, 请检查
406	MPC.10123	The header information of the input file is incorrect.	片源文件头信息有误, 请检查	片源文件头信息有误, 请检查
406	MPC.10124	Some data in the input file are missing.	片源部分数据缺失, 请检查片源是否可以完整	片源部分数据缺失, 请检查片源是否可以完整

状态码	错误码	错误信息	描述	处理措施
406	MPC.10125	Input data error.	片源数据问题, 请检查片源能否播放	片源数据问题, 请检查片源能否播放
406	MPC.10126	Input data error.	片源数据问题, 请检查片源能否播放	片源数据问题, 请检查片源能否播放
406	MPC.10127	Failed to obtain the level-1 m3u8 when an HLS media file is encrypted with DRM.	DRM加密, HLS格式获取一级m3u8失败	DRM加密, HLS格式获取一级m3u8失败, 请检查
406	MPC.10128	Failed to obtain the level-2 m3u8 when an HLS media file is encrypted with DRM.	DRM加密, HLS格式获取二级m3u8失败	DRM加密, HLS格式获取二级m3u8失败, 请检查
406	MPC.10129	Failed to obtain the index file when a DASH media file is encrypted with DRM.	DRM加密, DASH格式获取索引文件失败	DRM加密, DASH格式获取索引文件失败, 请检查
406	MPC.10130	The HLS content fails to be encrypted using DRM.	DRM加密, HLS格式加密失败	DRM加密, HLS格式加密失败, 请检查
406	MPC.10131	Failed to modify the index file when an HLS media file is encrypted with DRM.	DRM加密, HLS格式更新索引文件失败	DRM加密, HLS格式更新索引文件失败, 请检查
406	MPC.10132	Failed to obtain the IV during DRM encryption.	DRM加密, DASH获取iv失败	DRM加密, DASH获取iv失败, 请检查

状态码	错误码	错误信息	描述	处理措施
406	MPC.10133	The DASH content fails to be encrypted using DRM.	DRM加密,DASH格式加密失败	DRM加密,DASH格式加密失败, 请检查
406	MPC.10134	Failed to modify the index file when a DASH media file is encrypted with DRM.	DRM加密,DASH格式更新索引文件失败	DRM加密,DASH格式更新索引文件失败, 请检查
406	MPC.10135	Failed to package the digital watermark due to the incorrect xformat configuration.	数字水印转封装失败, 配置xformat的错误	数字水印转封装失败, 配置xformat的错误, 请检查
406	MPC.10136	Failed to package the digital watermark because xformat fails to be started.	数字水印转封装失败, 启动xformat失败	数字水印转封装失败, 启动xformat失败, 请检查
406	MPC.10137	Failed to package the digital watermark because xformat fails to create a task.	数字水印转封装失败, xformat创建task失败	数字水印转封装失败, xformat创建task失败, 请检查
406	MPC.10138	Failed to package the digital watermark because xformat fails to query the task.	数字水印转封装失败, xformat查询任务失败	数字水印转封装失败, xformat查询任务失败, 请检查

状态码	错误码	错误信息	描述	处理措施
406	MPC.10139	Failed to package the digital watermark because the xformat task timed out.	数字水印转封装失败，xformat任务超时	数字水印转封装失败，xformat任务超时，请重试或联系工程师
406	MPC.10140	The I-frame interval exceeds 500.	输出i帧间隔超过500，暂不支持	输出i帧间隔超过500，暂不支持，请修改
406	MPC.10141	The input file is an audio file. The selected template contains video parameters.	片源为纯音频文件，选择模板包含视频参数，暂不支持	片源为纯音频文件，选择模板包含视频参数，请检查
406	MPC.10143	Invalid index file content.	输入的索引文件内容非法	输入的索引文件内容非法
406	MPC.10144	Black bars seem to be on the input video.	无法确定片源黑边的具体位置，片源的四周疑似有黑边，需要对片源进行人工审核	无法确定片源黑边的具体位置，片源的四周疑似有黑边，需要对片源进行人工审核
406	MPC.10145	Data frames imported to the detection module seem to be not enough for identifying the specific position of the black bar.	无法确定片源黑边的具体位置，疑似没有足够的帧输入至黑边检测模块，需要对片源进行人工审核	无法确定片源黑边的具体位置，疑似没有足够的帧输入至黑边检测模块，需要对片源进行人工审核
406	MPC.10146	The black bar seems to overlap with subtitles.	无法确定片源黑边的具体位置，疑似片源的下黑边和字幕存在重叠，需要对片源进行人工审核	无法确定片源黑边的具体位置，疑似片源的下黑边和字幕存在重叠，需要对片源进行人工审核

状态码	错误码	错误信息	描述	处理措施
406	MPC.10147	The black bar seems to overlap with the watermark.	无法确定片源黑边的具体位置，疑似片源的上黑边和水印存在重叠，需要对片源进行人工审核	无法确定片源黑边的具体位置，疑似片源的上黑边和水印存在重叠，需要对片源进行人工审核
406	MPC.10148	The black bars seem to be asymmetric.	无法确定片源黑边的具体位置，疑似片源的左右黑边不对称，需要对片源进行人工审核	无法确定片源黑边的具体位置，疑似片源的左右黑边不对称，需要对片源进行人工审核
406	MPC.10149	The specific position of the black bar cannot be identified.	无法确定片源黑边的具体位置，需要对片源进行人工审核	无法确定片源黑边的具体位置，需要对片源进行人工审核
406	MPC.10150	The cropped black bar size exceeds the input video size.	强制黑边裁剪值超出原视频大小，请人工审核	强制黑边裁剪值超出原视频大小，请人工审核
406	MPC.10151	Failed to download the subtitle file in the slicing phase.	切片阶段下载字幕文件失败	切片阶段下载字幕文件失败，请检查
406	MPC.10152	The video encoding format of the input file is not supported.	片源视频编码格式特殊，暂不支持	片源视频编码格式特殊，暂不支持
406	MPC.10153	Input file error.	片源问题，请确认片源是否能完整播放	片源问题，请确认片源是否能完整播放
406	MPC.10154	Failed to open the input file.	输入文件无法打开，请检查输入片源是否能播放	输入文件无法打开，请检查输入片源是否能播放
406	MPC.10200	System error.	MPC 服务异常，通用状态码	请联系工程师解决

状态码	错误码	错误信息	描述	处理措施
406	MPC.10201	Internal communication error.	服务内部通信异常	请联系工程师解决
406	MPC.10209	Invalid input or output OBS path.	MPC 对象存储源地址或者目的地址不正确	请检查对象存储源地址或者目的地址
406	MPC.10210	Failed to obtain the input file from OBS.	MPC 获取对象存储源文件失败	对象存储源文件获取失败, 请检查
406	MPC.10213	Operation failed. The task is not in the final state.	MPC 操作失败, 任务未进入终态	任务未进入终态, 请检查
406	MPC.10216	Failed to set event notifications. You do not have the permission to publish messages to the topic.	MPC 设置消息通知失败, 无权限发布消息到主题	设置消息通知失败, 无权限发布消息到主题, 请检查
406	MPC.10217	The usage exceeds the OBT quota.	MPC 公测限额, 用量超过阈值	用量超过阈值, 请检查
406	MPC.10218	The task has completed.	MPC 任务处理成功	任务处理成功, 请检查
406	MPC.10219	Invalid request parameter.	请求参数非法	请求参数非法, 请检查
406	MPC.10220	The task has expired.	MPC 任务已过期	任务已过期, 请检查
406	MPC.10221	Internal service error.	内部服务异常	请检查模板并重试
406	MPC.10222	Key parameters in the template are inconsistent.	自定义模板参数错误	对象存储源文件获取错误, 请检查
406	MPC.10225	KMS service error.	HW_KMS 服务处理异常	请联系工程师解决

状态码	错误码	错误信息	描述	处理措施
406	MPC.10227	You do not have the permission to access the requested resource.	没有权限访问	没有权限访问, 请检查
406	MPC.10228	Your account is in arrears. Top up your account.	用户已冻结, 请尽快充值	用户已冻结, 请尽快充值
406	MPC.10229	You do not have the permission to perform this operation.	无角色权限执行该的操作	无角色权限执行该的操作, 请检查
406	MPC.10232	GIF task failed.	gif任务失败	gif任务失败, 请检查
406	MPC.10233	Packaging task failed.	转封装任务失败	转封装任务失败, 请检查
406	MPC.10234	The function is temporarily brought offline.	功能下线	功能下线, 请检查
406	MPC.10235	Identity authentication failed due to an invalid token.	请求Token为Domain级别, Token无效	请检查Token是否正确
406	MPC.10236	You do not have permission to access the OBS bucket.	账号桶操作无权限	请联系租户管理员进行桶授权或者租户管理员给予账号赋予OBS权限
406	MPC.10237	API Gateway rate limiting	APIGW 流控	服务APIGW流控, 请检查
500	MPC.10001	IAM service exception.	IAM 服务处理异常	请联系工程师解决
500	MPC.10002	OBS service exception.	OBS 服务处理异常	请联系工程师解决
500	MPC.10003	SMN service exception.	SMN 服务处理异常	请联系工程师解决

状态码	错误码	错误信息	描述	处理措施
500	MPC.10004	CBC service exception.	CBC 服务处理异常	请联系工程师解决
500	MPC.10005	SDR service exception.	SDR 服务处理异常	请联系工程师解决
500	MPC.10006	ZK service exception.	ZK 服务处理异常	请联系工程师解决
500	MPC.10007	MONGO service exception.	MONGO 服务处理异常	请联系工程师解决
500	MPC.10008	MPE service exception.	MPE处理错误	请联系工程师解决
500	MPC.10050	XCODE service exception.	XCODE 服务处理异常	请联系工程师解决

5.3 返回状态码

消息请求返回的状态码如下表所示。

状态码	提示信息
200 OK - [GET]	服务器成功返回用户请求的数据。
201 CREATED - [POST/PUT/PATCH]	用户新建或修改数据成功。
202 Accepted - [*]	表示一个请求已经进入后台排队（异步任务）
204 NO CONTENT - [DELETE]	用户删除数据成功。
400 INVALID REQUEST - [POST/PUT/PATCH]	用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的。
401 Unauthorized - [*]	表示用户没有权限（令牌、用户名、密码错误）。
403 Forbidden - [*]	表示用户得到授权（与401错误相对），但是访问是被禁止的。
404 NOT FOUND - [*]	用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的。
406 Not Acceptable - [GET]	用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式）。

状态码	提示信息
410 Gone -[GET]	用户请求的资源被永久删除，且不会再得到的。
422 Unprocesable entity - [POST/PUT/PATCH]	当创建一个对象时，发生一个验证错误。
500 INTERNAL SERVER ERROR - [*]	服务器发生错误，用户将无法判断发出的请求是否成功。

5.4 获取关键参数

在使用SDK前，您需要获取一些关键参数，用于签名认证，需要获取的参数如下所示：

- AK (Access Key ID)：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- SK (Secret Access Key)：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。
- Project_ID：项目ID，部分请求URL中需要包含该字段。
- 账户名：部分请求URL中需要包含该字段。
- Endpoint：各服务应用区域和各服务的终端节点。

前提条件

已完成华为云官网[注册](#)，并进行了[实名认证](#)。

获取 AK 和 SK

注意：访问密钥对账号具有完全的访问权限，如果访问密钥泄露，会带来数据泄露风险，为了账号安全性，建议您定期更换并妥善保存访问密钥。每个账号最多只能创建2个密钥。

步骤1 登录[管理控制台](#)。

步骤2 鼠标移动至用户名，在下拉列表中单击“我的凭证”。

步骤3 在左侧导航栏中选择“访问密钥”。

步骤4 单击“新增访问密钥”，在弹出的页面中输入账号密码及短信验证码。

图 5-1 访问密钥



步骤5 单击“确定”，即可下载一个命名为“credentials.csv”的文件，其中包含AK和SK。

----结束

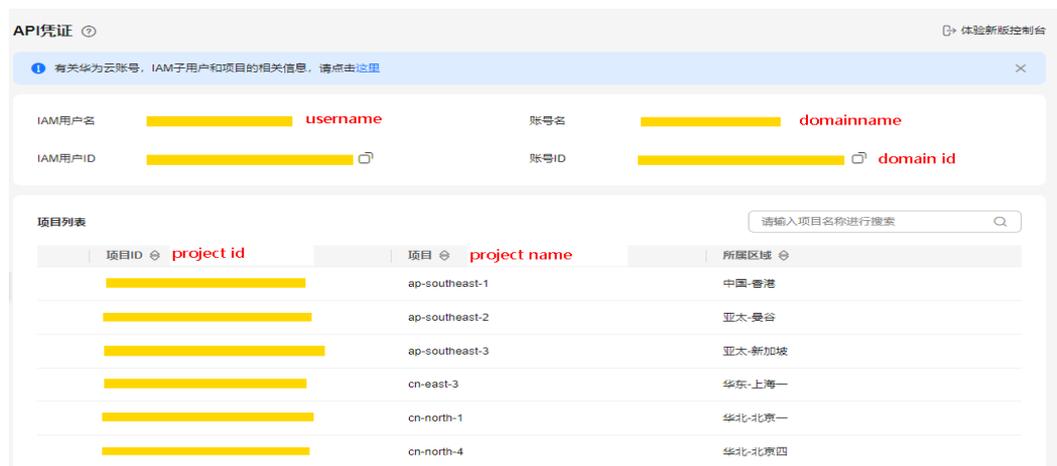
获取项目 ID 和账号名

步骤1 登录[管理控制台](#)。

步骤2 鼠标移动至用户名，在下拉列表中单击“我的凭证”。

步骤3 在“API凭证”页面，即可获取对应的项目ID和账号名信息。

图 5-2 获取项目 ID



----结束

获取 EndPoint

在SDK初始化时需要使用到Endpoint，您可以在[地区和终端节点](#)页面获取EndPoint值。

6 修订记录

表 6-1 修订记录

发布日期	修订记录
2025-08-05	本次更新：Java SDK下线独立加密功能。
2021-03-30	本次更新：新增Java SDK。
2020-10-30	本次更新说明如下： <ul style="list-style-type: none">• 新增Python SDK。• 新增Go SDK。
2019-04-30	本次更新：新增Maven安装SDK的方式。
2018-09-30	第一次正式发布。