ALIBABA CLOUD

阿里云

视频点播

服务端SDK

文档版本：20220712

阿里云

法律声明

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

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.使用说明 09
2.Java SDK 11
2.1. 安装 11
2.2. 初始化 11
2.3. 媒体上传 12
2.4. 音视频播放 23
2.5. 媒资管理 25
2.6. 媒资分类 38
2.7. 媒体处理 42
2.8. 视频水印 48
2.9. 转码模板 56
2.10. 截图模板 64
2.11. 视频剪辑 70
2.12. 媒体审核 82
2.13. 视频AI 91
2.14. 视频DNA 101
2.15. 直播转点播 105
2.16. 转码任务 105
2.17. 点播CDN 108
3.Python SDK 118
3.1. 安装 118
3.2. 初始化 118
3.3. 媒体上传 119
3.4. 音视频播放 124
3.5. 媒资管理 125
3.6. 媒资分类 133

3.7. 媒体处理 135
3.8. 视频水印 141
3.9. 转码模板 146
3.10. 截图模板 151
3.11. 视频剪辑 154
3.12. 媒体审核 162
3.13. 视频AI 167
3.14. 视频DNA 172
3.15. 直播转点播 174
3.16. 点播CDN 175
4.PHP SDK 179
4.1. 安装SDK 179
4.1.1. 安装 179
4.1.2. 初始化 180
4.2. 使用新版（Composer） 182
4.2.1. 安装 182
4.2.2. 初始化 184
4.3. 媒体上传 186
4.4. 音视频播放 193
4.5. 媒资管理 194
4.6. 媒资分类 204
4.7. 媒体处理 206
4.8. 视频水印 211
4.9. 转码模板 217
4.10. 截图模板 223
4.11. 视频剪辑 228
4.12. 媒体审核 235
4.13. 视频AI 243

4.14. 视频DNA 249
4.15. 直播转点播 251
4.16. 点播CDN 251
5..NET SDK 256
5.1. 安装 256
5.2. 初始化 256
5.3. 媒体上传 257
5.4. 音视频播放 263
5.5. 媒资管理 265
5.6. 媒资分类 278
5.7. 媒体处理 282
5.8. 视频水印 290
5.9. 转码模板 301
5.10. 截图模板 312
5.11. 视频剪辑 321
5.12. 媒体审核 336
5.13. 视频AI 350
5.14. 视频DNA 364
5.15. 直播转点播 368
6.Node.js SDK 371
6.1. 安装 371
6.2. 初始化 371
6.3. 媒体上传 372
6.4. 音视频播放 375
6.5. 媒资管理 377
6.6. 媒资分类 385
6.7. 媒体处理 387
6.8. 视频水印 392

6.9. 转码模板 397
6.10. 截图模板 402
6.11. 视频剪辑 406
6.12. 媒体审核 413
6.13. 视频AI 418
6.14. 视频DNA 424
6.15. 直播转点播 427
6.16. 点播CDN 427
7.Go SDK 434
7.1. 安装 434
7.2. 初始化 434
7.3. 媒体上传 437
7.4. 音视频播放 446
7.5. 媒资管理 447
7.6. 媒资分类 460
7.7. 媒体处理 464
7.8. 视频水印 472
7.9. 转码模板 479
7.10. 截图模板 487
7.11. 视频剪辑 493
7.12. 媒体审核 503
7.13. 视频AI 512
7.14. 视频DNA 521
7.15. 直播转点播 524
7.16. 点播CDN 525
8.C/C++ SDK 533
8.1. 安装 533
8.2. 初始化 535

8.3. 媒体上传 536
8.4. 音视频播放 541
8.5. 媒资管理 542
8.6. 媒资分类 552
8.7. 媒体处理 555
8.8. 视频水印 560
8.9. 转码模板 566
8.10. 截图模板 572
8.11. 视频剪辑 577
8.12. 媒体审核 586
8.13. 视频AI 593
8.14. 视频DNA 601
8.15. 直播转点播 604

1.使用说明

点播服务提供了多种主流语言的SDK，您可以基于服务端SDK编写代码来调用点播API，让您不用复杂编程即可快捷访问点播服务。

功能介绍

- SDK封装了对服务端API的调用请求和响应，避免自行计算较为繁琐的签名机制。
- 支持所有的点播API，并提供了相应的示例代码。
- 支持7种开发语言，包括：Java、Python、PHP、.NET、Node.js、Go、C/C++。

> **说明** 通常在发布新的API后，阿里云会及时同步更新SDK，所以即便您没有找到对应API的示例代码，也可以参考旧的示例自行实现调用。

开发环境

服务端SDK支持Linux、Windows、Mac等多种平台和运行环境，请提前安装好对应开发语言的编译器或解释器，并完成相关环境配置，具体版本要求可参考各语言SDK的安装文档。

> **说明** C/C++ SDK暂不支持Windows。

SDK安装

- 版本说明

  SDK版本更新日志和发布记录请参见服务端SDK发布历史，请确保您安装的是最新版本，否则部分API可能无法直接使用。

  - Java、Python、PHP、.NET、Go等版本，每个API都会对应特定的请求类（Request）和响应类（Response），使用新的API或新功能需要保持更新。
  - Node.js、C/C++版本，实现的是通用库，未封装请求类和响应类，故不需要随API更新而更新SDK，但通用库可能会不定期更新。

- 安装操作

  一般需要安装SDK核心库和VoD库，具体操作请参见视频点播服务端SDK各语言安装文档。

- 使用说明

  安装完成后，可参考各语言SDK的初始化文档完成初始化客户端实例。

访问域名

点播服务在全球提供了多个接入地址，更多内容请参见点播地域标识，通过 `API接入区域（API Region）` 来初始化客户端实例，`接入区域标识` 对应API/SDK的RegionId参数，例如上海为 `cn-shanghai` ，新加坡为 `ap-southeast-1` 。

> **注意** 接入地址的区域与存储区域是不一样的概念，接入区域表示API的接入点，存储区域对应OSS的区域，如API接入点在上海（cn-shanghai），但媒体文件可能存储在北京（cn-beijing）。

使用限制

点播服务对资源使用，以及API的调用次数等都有默认限制，更多内容请参见使用限制，如要提高限制，可联系售后，或提工单联系我们，并请详细说明使用场景和预期的阈值（如域名数、播放接口访问频次等）。

提交反馈

如果您在使用SDK的过程中遇到任何问题，欢迎前往阿里云SDK问答社区提问，提问前请您阅读反馈引导。

2.Java SDK

2.1. 安装

本文为您介绍了安装Java SDK添加Jar包依赖的示例代码。

环境准备

适用于JDK 6及以上版本。

安装SDK

添加Jar包依赖：

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>4.6.0</version>
</dependency>
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-vod</artifactId>
  <version>2.16.5</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.62</version>
</dependency>
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-kms</artifactId>
  <version>2.10.1</version>
</dependency>
```

说明

- `aliyun-java-sdk-core` 的版本4.4.5及以上。
- `aliyun-java-sdk-vod` 的版本，请参见服务端SDK发布历史，首段的 `Version` 即为最新的版本号。
- `aliyun-java-sdk-kms` 为密钥管理服务（KMS）的所需依赖，若不涉及则无需添加。

2.2. 初始化

Java SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端Java SDK。安装方法，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。示例代码如下：

```
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
//填入AccessKey信息
public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret) th
rows ClientException {
    String regionId = "cn-shanghai";  // 点播服务接入地域
    DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecr
et);
    DefaultAcsClient client = new DefaultAcsClient(profile);
    return client;
}
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例代码如下：

```
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
//填入STS信息
public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret, St
ring securityToken) throws ClientException {
    String regionId = "cn-shanghai";  // 点播服务接入地域
    DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecr
et, securityToken);
    DefaultAcsClient client = new DefaultAcsClient(profile);
    return client;
}
```

2.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍Java SDK调用服务端上传接口的使用场景和调用示例。

使用场景

> **注意**
>
> 本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端实例。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端实例。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.CreateUploadVideoRequest;
import com.aliyuncs.vod.model.v20170321.CreateUploadVideoResponse;

/**
 *
```

```
 * 音频或视频获取上传凭证的示例
 */
public class AudioOrVideoCreateUploadDemo {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * 获取音/视频上传地址和凭证
     * @param client 发送请求客户端
     * @return CreateUploadVideoResponse 获取音/视频上传地址和凭证响应数据
     * @throws Exception
     */
    public static CreateUploadVideoResponse createUploadVideo(DefaultAcsClient client) thro
ws Exception {
        CreateUploadVideoRequest request = new CreateUploadVideoRequest();
        request.setTitle("this is a sample");
        request.setFileName("filename.mp4");

        //UserData，用户自定义设置参数，用户需要单独回调URL及数据透传时设置(非必须)
        //JSONObject userData = new JSONObject();

        //UserData回调部分设置
        //消息回调设置，指定时以此为准，否则以全局设置的事件通知为准
        //JSONObject messageCallback = new JSONObject();
        //设置回调地址
        //messageCallback.put("CallbackURL", "http://192.168.0.1/16");
        //设置回调方式，默认为http
        //messageCallback.put("CallbackType", "http");
        //userData.put("MessageCallback", messageCallback.toJSONString());

        //UserData透传数据部分设置
        //用户自定义的扩展字段，用于回调时透传返回
        //JSONObject extend = new JSONObject();
        //extend.put("MyId", "user-defined-id");
        //userData.put("Extend", extend.toJSONString());

        //request.setUserData(userData.toJSONString());

        return client.getAcsResponse(request);
    }

    // 请求示例
    public static void main(String[] argv) throws ClientException {
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
```

```
            CreateUploadVideoResponse response = new CreateUploadVideoResponse();
            response = createUploadVideo(client);
            System.out.print("VideoId = " + response.getVideoId() + "\n");
            System.out.print("UploadAddress = " + response.getUploadAddress() + "\n");
            System.out.print("UploadAuth = " + response.getUploadAuth() + "\n");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
    }
}
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.RefreshUploadVideoRequest;
import com.aliyuncs.vod.model.v20170321.RefreshUploadVideoResponse;

/**
*
* 音频或视频刷新上传凭证的示例
*/
public class AudioOrVideoRefreshUploadDemo {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * 刷新音/视频上传凭证
     * @param client 发送请求客户端
     * @return RefreshUploadVideoResponse 刷新音/视频上传凭证响应数据
     * @throws Exception
     */
    public static RefreshUploadVideoResponse refreshUploadVideo(DefaultAcsClient client) th
rows Exception {
        RefreshUploadVideoRequest request = new RefreshUploadVideoRequest();
        //音频或视频ID
        request.setVideoId("<VideoId>");
        return client.getAcsResponse(request);
    }

    // 请求示例
    public static void main(String[] argv)  {

        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            RefreshUploadVideoResponse response = refreshUploadVideo(client);
            System.out.print("VideoId = " + response.getVideoId() + "\n");
            System.out.print("UploadAddress = " + response.getUploadAddress() + "\n");
            System.out.print("UploadAuth = " + response.getUploadAuth() + "\n");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
    }
}
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.CreateUploadImageRequest;
import com.aliyuncs.vod.model.v20170321.CreateUploadImageResponse;

/**
 * 图片获取上传凭证的示例
 *
 */
public class ImageCreateUploadDemo {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * 获取图片上传地址和凭证
     * @param client 发送请求客户端
     * @return CreateUploadImageResponse 获取图片上传地址和凭证响应数据
     * @throws Exception
     */
    public static CreateUploadImageResponse createUploadImage(DefaultAcsClient client) thro
ws Exception {
        CreateUploadImageRequest request = new CreateUploadImageRequest();
        //设置图片类型
        request.setImageType("default");
        //设置图片扩展名
        request.setImageExt("gif");
        //设置图片标题
        request.setTitle("this is a sample");

        //UserData，用户自定义设置参数，用户需要单独回调URL及数据透传时设置(非必须)
        JSONObject userData = new JSONObject();

        //UserData回调部分设置
        //消息回调设置，指定时以此为准，否则以全局设置的事件通知为准
        JSONObject messageCallback = new JSONObject();
        //设置回调地址
        messageCallback.put("CallbackURL", "http://192.168.0.0/16");
        //设置回调方式，默认为http
        messageCallback.put("CallbackType", "http");
```

```
        userData.put("MessageCallback", messageCallback.toJSONString());

        JSONObject extend = new JSONObject();
        extend.put("MyId", "user-defined-id");
        userData.put("Extend", extend.toJSONString());

        request.setUserData(userData.toJSONString());

        return client.getAcsResponse(request);
    }

    // 请求示例
    public static void main(String[] argv)  {

        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            CreateUploadImageResponse response = createUploadImage(client);
            System.out.print("ImageId = " + response.getImageId() + "\n");
            System.out.print("ImageURL = " + response.getImageURL() + "\n");
            System.out.print("UploadAddress = " + response.getUploadAddress() + "\n");
            System.out.print("UploadAuth = " + response.getUploadAuth() + "\n");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
    }
}
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.CreateUploadAttachedMediaRequest;
import com.aliyuncs.vod.model.v20170321.CreateUploadAttachedMediaResponse;

/**
 * 辅助媒资获取上传凭证的示例
 *
 */
public class AttachedMediaCreateUploadDemo {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
```

```
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * 获取辅助媒资(水印、字幕等)上传地址和凭证
     * @param client 发送请求客户端
     * @return CreateUploadAttachedMediaResponse 获取辅助媒资上传地址和凭证响应数据
     * @throws Exception
     */
    public static CreateUploadAttachedMediaResponse createUploadAttachedMedia(DefaultAcsCli
ent client) throws Exception {
        CreateUploadAttachedMediaRequest request = new CreateUploadAttachedMediaRequest();
        //设置业务类型
        request.setBusinessType("watermark");
        //设置文件扩展名
        request.setMediaExt("gif");
        //设置标题
        request.setTitle("this is a sample");

        //UserData，用户自定义设置参数，用户需要单独回调URL及数据透传时设置(非必须)
        JSONObject userData = new JSONObject();

        //UserData回调部分设置
        //消息回调设置，指定时以此为准，否则以全局设置的事件通知为准
        JSONObject messageCallback = new JSONObject();
        //设置回调地址
        messageCallback.put("CallbackURL", "http://192.168.0.0/16");
        //设置回调类型，默认为http
        messageCallback.put("CallbackType", "http");
        userData.put("MessageCallback", messageCallback.toJSONString());

        JSONObject extend = new JSONObject();
        extend.put("MyId", "user-defined-id");
        userData.put("Extend", extend.toJSONString());

        request.setUserData(userData.toJSONString());

        return client.getAcsResponse(request);
    }
    // 请求示例
    public static void main(String[] argv) {

        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            CreateUploadAttachedMediaResponse response = createUploadAttachedMedia(client);
            System.out.print("mediaId = " + response.getMediaId() + "\n");
            System.out.print("mediaURL = " + response.getMediaURL() + "\n");
            System.out.print("UploadAddress = " + response.getUploadAddress() + "\n");
            System.out.print("UploadAuth = " + response.getUploadAuth() + "\n");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
```

```
        }
    }
}
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.UploadMediaByURLRequest;
import com.aliyuncs.vod.model.v20170321.UploadMediaByURLResponse;

import java.net.URLEncoder;

/**
 * URL批量拉取上传功能示例
 *
 */
public class AudioOrVideoUploadByUrl {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * URL批量拉取上传
     *
     * @param client 发送请求客户端
     * @return UploadMediaByURLResponse URL批量拉取上传响应数据
     * @throws Exception
     */
    public static UploadMediaByURLResponse uploadMediaByURL(DefaultAcsClient client) throws
Exception {
        UploadMediaByURLRequest request = new UploadMediaByURLRequest();
        String url = "http://video_01.mp4";
        String encodeUrl = URLEncoder.encode(url, "UTF-8");
        //视频源文件URL
        request.setUploadURLs(encodeUrl);

        //上传视频元数据信息
        JSONObject uploadMetadata = new JSONObject();
        //需要上传的视频源文件URL，与UploadURLs里的URL匹配才能生效
```

```
        uploadMetadata.put("SourceUrl", encodeUrl);
        //视频标题
        uploadMetadata.put("Title", "upload by url sample");

        JSONArray uploadMetadataList = new JSONArray();
        uploadMetadataList.add(uploadMetadata);
        request.setUploadMetadatas(uploadMetadataList.toJSONString());

        //UserData，用户自定义设置参数，用户需要单独回调URL及数据透传时设置（非必须）
        JSONObject userData = new JSONObject();

        //UserData回调部分设置
        //消息回调设置，指定时以此为准，否则以全局设置的事件通知为准
        JSONObject messageCallback = new JSONObject();
        //设置回调地址
        messageCallback.put("CallbackURL", "http://192.168.0.0/16");
        //设置回调类型，默认为http
        messageCallback.put("CallbackType", "http");
        userData.put("MessageCallback", messageCallback.toJSONString());

        JSONObject extend = new JSONObject();
        extend.put("MyId", "user-defined-id");
        userData.put("Extend", extend.toJSONString());

        request.setUserData(userData.toJSONString());

        return client.getAcsResponse(request);
    }

    // 请求示例
    public static void main(String[] argv) {

        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            UploadMediaByURLResponse response = uploadMediaByURL(client);
            System.out.print("UploadJobs = " + JSON.toJSONString(response.getUploadJobs())
+ "\n");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
    }

}
```

获取URL上传信息

调用GetURLUploadInfos接口，完成获取URL上传信息功能。

接口参数和返回字段请参见GetURLUploadInfos。调用示例如下：

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
```

```
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.GetURLUploadInfosRequest;
import com.aliyuncs.vod.model.v20170321.GetURLUploadInfosResponse;
import org.apache.commons.lang3.StringUtils;

import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;

/**
 * 获取URL上传信息示例
 */
public class URLUploadInfosGetDemo {

    //填入AccessKey信息
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入地域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }

    /**
     * 获取URL上传信息
     *
     * @param client 发送请求客户端
     * @return GetURLUploadInfosResponse 获取URL上传信息响应数据
     * @throws Exception
     */
    public static GetURLUploadInfosResponse getURLUploadInfos(DefaultAcsClient client) thro
ws Exception {
        GetURLUploadInfosRequest request = new GetURLUploadInfosRequest();

        //上传源视频URL列表，需要URLencode
        String[] urls = {
                "http://exampleBucket****.cn-shanghai.aliyuncs.com/video_01.mp4",
                "http://exampleBucket****.cn-shanghai.aliyuncs.com/video_02.flv"
        };
        List<String> encodeUrlList = new ArrayList<String>();
        for (String url : urls) {
            encodeUrlList.add(URLEncoder.encode(url, "UTF-8"));
        }
        request.setUploadURLs(StringUtils.join(encodeUrlList, ','));
        //JobId列表，可以通过GetPlayInfo接口中返回的PlayInfo结构体中获取
        //request.setJobIds("exampleID1,exampleID2");

        return client.getAcsResponse(request);
    }

    // 请求示例
    public static void main(String[] argv) {
```

```
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            GetURLUploadInfosResponse response = getURLUploadInfos(client);
            System.out.print("URLUploadInfoList = " + response.getURLUploadInfoList() + "\n
");
            System.out.print("RequestId = " + response.getRequestId() + "\n");
        } catch (Exception e) {
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
    }
}
```

2.4. 音视频播放

本篇文档提供了Java SDK获取视频播放地址、获取视频播放凭证的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取视频播放地址

调用GetPlayInfo接口，完成获取视频播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetPlayInfoRequest;
import com.aliyuncs.vod.model.v20170321.GetPlayInfoResponse;

/*获取播放地址函数*/
public static GetPlayInfoResponse getPlayInfo(DefaultAcsClient client) throws Exception {
    GetPlayInfoRequest request = new GetPlayInfoRequest();
    request.setVideoId("视频ID");
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetPlayInfoResponse response = new GetPlayInfoResponse();
    try {
        response = getPlayInfo(client);
        List<GetPlayInfoResponse.PlayInfo> playInfoList = response.getPlayInfoList();
        //播放地址
        for (GetPlayInfoResponse.PlayInfo playInfo : playInfoList) {
            System.out.print("PlayInfo.PlayURL = " + playInfo.getPlayURL() + "\n");
        }
        //Base信息
        System.out.print("VideoBase.Title = " + response.getVideoBase().getTitle() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参考GetVideoPlayAuth。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetVideoPlayAuthRequest;
import com.aliyuncs.vod.model.v20170321.GetVideoPlayAuthResponse;

/*获取播放凭证函数*/
public static GetVideoPlayAuthResponse getVideoPlayAuth(DefaultAcsClient client) throws Exc
eption {
    GetVideoPlayAuthRequest request = new GetVideoPlayAuthRequest();
    request.setVideoId("视频ID");
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetVideoPlayAuthResponse response = new GetVideoPlayAuthResponse();
    try {
        response = getVideoPlayAuth(client);
        //播放凭证
        System.out.print("PlayAuth = " + response.getPlayAuth() + "\n");
        //VideoMeta信息
        System.out.print("VideoMeta.Title = " + response.getVideoMeta().getTitle() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

2.5. 媒资管理

本篇文档提供了Java SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SearchMediaRequest;
import com.aliyuncs.vod.model.v20170321.SearchMediaResponse;
/**
 * 搜索媒资信息
 * @param client 发送请求客户端
 * @return SearchMediaResponse 搜索媒资信息响应数据
 * @throws Exception
*/
public static SearchMediaResponse searchMedia(DefaultAcsClient client) throws Exception {
    SearchMediaRequest request = new SearchMediaRequest();
    request.setFields("Title,CoverURL,Status");
    request.setMatch("Status in ('Normal','Checking') and CreationTime = ('2018-07-01T08:00
:00Z','2018-08-01T08:00:00Z')");
    request.setPageNo(1);
    request.setPageSize(10);
    request.setSearchType("video");
    request.setSortBy("CreationTime:Desc");
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SearchMediaResponse response = new SearchMediaResponse();
    try {
        response = searchMedia(client);
        if (response.getMediaList() != null && response.getMediaList().size() > 0) {
            System.out.print("Total = " + response.getTotal() + "\n");
            System.out.print("ScrollToken = " + response.getScrollToken() + "\n");
            for (SearchMediaResponse.Media media : response.getMediaList()) {
                System.out.print("MediaId = " + media.getMediaId() + "\n");
                System.out.print("MediaType = " + media.getMediaType() + "\n");
                System.out.print("CreationTime = " + media.getCreationTime() + "\n");
                System.out.print("Title = " + media.getVideo().getTitle() + "\n");
                System.out.print("CoverURL = " + media.getVideo().getCoverURL() + "\n");
                System.out.print("Status = " + media.getVideo().getStatus() + "\n");
            }
        }
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetVideoInfoRequest;
import com.aliyuncs.vod.model.v20170321.GetVideoInfoResponse;
/**
 * 获取视频信息
 * @param client 发送请求客户端
 * @return GetVideoInfoResponse 获取视频信息响应数据
 * @throws Exception
*/
public static GetVideoInfoResponse getVideoInfo(DefaultAcsClient client) throws Exception {
    GetVideoInfoRequest request = new GetVideoInfoRequest();
    request.setVideoId("VideoId");
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetVideoInfoResponse response = new GetVideoInfoResponse();
    try {
        response = getVideoInfo(client);
        System.out.print("Title = " + response.getVideo().getTitle() + "\n");
        System.out.print("Description = " + response.getVideo().getDescription() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetVideoInfosRequest;
import com.aliyuncs.vod.model.v20170321.GetVideoInfosResponse;
/**
 * 批量获取视频信息函数
 * @param client 发送请求客户端
 * @return GetVideoInfosResponse 获取视频信息响应数据
 * @throws Exception
*/
public static GetVideoInfosResponse getVideoInfos(DefaultAcsClient client) throws Exception
{
    GetVideoInfosRequest request = new GetVideoInfosRequest();
    request.setVideoIds("VideoId1,VideoId2");
    return client.getAcsResponse(request);
}
//请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetVideoInfosResponse response = new GetVideoInfosResponse();
    try {
        response = getVideoInfos(client);
        if (response.getVideoList() != null && response.getVideoList().size() > 0) {
            System.out.print("===== exist video infos : ===== \n");
            for (GetVideoInfosResponse.Video video : response.getVideoList()) {
                System.out.print("VideoId = " + video.getVideoId() + "\n");
                System.out.print("Title = " + video.getTitle() + "\n");
                System.out.print("Description = " + video.getDescription() + "\n");
                System.out.print("Tags = " + video.getTags() + "\n");
                System.out.print("CreationTime = " + video.getCreationTime() + "\n");
            }
        }
        if (response.getNonExistVideoIds() != null && response.getNonExistVideoIds().size()
> 0) {
            System.out.print("===== nonexistent videoIds : ===== \n");
            for (String videoId : response.getNonExistVideoIds()) {
                System.out.print(videoId + "\n");
            }
        }
    } catch (Exception e) {
      System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateVideoInfoRequest;
import com.aliyuncs.vod.model.v20170321.UpdateVideoInfoResponse;
/**
 * 修改视频信息
 * @param client 发送请求客户端
 * @return UpdateVideoInfoResponse 修改视频信息响应数据
 * @throws Exception
*/
public static UpdateVideoInfoResponse updateVideoInfo(DefaultAcsClient client) throws Excep
tion {
    UpdateVideoInfoRequest request = new UpdateVideoInfoRequest();
    request.setVideoId("VideoId");
    request.setTitle("new Title");
    request.setDescription("new Description");
    request.setTags("new Tag1,new Tag2");
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateVideoInfoResponse response = new UpdateVideoInfoResponse();
    try {
        response = updateVideoInfo(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateVideoInfosRequest;
import com.aliyuncs.vod.model.v20170321.UpdateVideoInfosResponse;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
/**
 * 批量修改视频信息
 * @param client 发送请求客户端
 * @return UpdateVideoInfosResponse 批量修改视频信息响应数据
 * @throws Exception
*/
public static UpdateVideoInfosResponse updateVideoInfos(DefaultAcsClient client) throws Exc
eption {
    UpdateVideoInfosRequest request = new UpdateVideoInfosRequest();
    JSONArray updateContentArray = new JSONArray();
    JSONObject updateContent1 = new JSONObject();
    updateContent1.put("VideoId", "VideoId1");
    // updateContent1.put("Title", "new Title");
    // updateContent1.put("Tags", "new Tag1,new Tag2");
    updateContentArray.add((updateContent1));
    JSONObject updateContent2 = new JSONObject();
    updateContent2.put("VideoId", "VideoId2");
    // updateContent2.put("Title", "new Title");
    // updateContent2.put("Tags", "new Tag1,new Tag2");
    updateContentArray.add((updateContent2));
    request.setUpdateContent(updateContentArray.toJSONString());
    DefaultAcsClient client = init();
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateVideoInfosResponse response = new UpdateVideoInfosResponse();
    try {
        response = updateVideoInfos(client);
        if (response.getNonExistVideoIds() != null && response.getNonExistVideoIds().size()
> 0) {
            System.out.print("======nonexistent VideoIds : ======\n");
            for (String videoId : response.getNonExistVideoIds()) {
                System.out.print(videoId + "\n");
            }
        }
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteVideoRequest;
import com.aliyuncs.vod.model.v20170321.DeleteVideoResponse;
/**
 * 删除视频
 * @param client 发送请求客户端
 * @return DeleteVideoResponse 删除视频响应数据
 * @throws Exception
*/
public static DeleteVideoResponse deleteVideo(DefaultAcsClient client) throws Exception {
    DeleteVideoRequest request = new DeleteVideoRequest();
    //支持传入多个视频ID，多个用逗号分隔
    request.setVideoIds("VideoId1,VideoId2");
    return client.getAcsResponse(request);
}
/*请求示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteVideoResponse response = new DeleteVideoResponse();
    try {
        response = deleteVideo(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取源文件信息（含源片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetMezzanineInfoRequest;
import com.aliyuncs.vod.model.v20170321.GetMezzanineInfoResponse;
/**
 * 获取源文件信息
 * @param client 发送请求客户端
 * @return GetMezzanineInfoResponse 获取源文件信息响应数据
 * @throws Exception
*/
public static GetMezzanineInfoResponse getMezzanineInfo(DefaultAcsClient client) throws Exc
eption {
    GetMezzanineInfoRequest request = new GetMezzanineInfoRequest();
    request.setVideoId("VideoId");
    //源片下载地址过期时间
    request.setAuthTimeout(3600L);
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetMezzanineInfoResponse response = new GetMezzanineInfoResponse();
    try {
        response = getMezzanineInfo(client);
        System.out.print("Mezzanine.VideoId = " + response.getMezzanine().getVideoId() + "\
n");
        System.out.print("Mezzanine.FileURL = " + response.getMezzanine().getFileURL() + "\
n");
        System.out.print("Mezzanine.Width = " + response.getMezzanine().getWidth() + "\n");
        System.out.print("Mezzanine.Height = " + response.getMezzanine().getHeight() + "\n"
);
        System.out.print("Mezzanine.Size = " + response.getMezzanine().getSize() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetVideoListRequest;
import com.aliyuncs.vod.model.v20170321.GetVideoListResponse;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
// 根据Date时间生成UTC时间函数
public static String generateUTCTime(Date time) {
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
    dateFormat.setTimeZone(new SimpleTimeZone(SimpleTimeZone.UTC_TIME, "UTC"));
    dateFormat.setLenient(false);
```

```
    return dateFormat.format(time);
}
/**
 * 获取视频列表
 * @param client 发送请求客户端
 * @return GetVideoListResponse 获取视频列表响应数据
 * @throws Exception
*/
public static GetVideoListResponse getVideoList(DefaultAcsClient client) throws Exception {
    GetVideoListRequest request = new GetVideoListRequest();
    // 分别取一个月前、当前时间的UTC时间作为筛选视频列表的起止时间
    String monthAgoUTCTime = generateUTCTime(new Date(System.currentTimeMillis() - 30*86400
*1000L));
    String nowUTCTime = generateUTCTime(new Date(System.currentTimeMillis()));
    // 视频创建的起始时间，为UTC格式
    request.setStartTime(monthAgoUTCTime);
    // 视频创建的结束时间，为UTC格式
    request.setEndTime(nowUTCTime);
    // 视频状态，默认获取所有状态的视频，多个用逗号分隔
    // request.setStatus("Uploading,Normal,Transcoding");
    request.setPageNo(1L);
    request.setPageSize(20L);
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetVideoListResponse response = new GetVideoListResponse();
    try {
        response = getVideoList(client);
        // 根据指定筛选条件查询到的视频总数
        System.out.print("Total = " + response.getTotal() + "\n");
        for (GetVideoListResponse.Video video : response.getVideoList()) {
            System.out.print("Title = " + video.getTitle() + "\n");
            System.out.print("Description = " + video.getDescription() + "\n");
            System.out.print("Tags = " + video.getTags() + "\n");
            System.out.print("CreationTime = " + video.getCreationTime() + "\n");
        }
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteStreamRequest;
import com.aliyuncs.vod.model.v20170321.DeleteStreamResponse;
/**
 * 删除媒体流函数
 * @param client 发送请求客户端
 * @return DeleteMezzaninesResponse 删除媒体流响应数据
 * @throws Exception
*/
public static DeleteStreamResponse deleteStream(DefaultAcsClient client) throws Exception {
    DeleteStreamRequest request = new DeleteStreamRequest();
    request.setVideoId("VideoId");
    request.setJobIds("JobId1,JobId2");
    return client.getAcsResponse(request);
}
//请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteStreamResponse response = new DeleteStreamResponse();
    try {
        response = deleteStream(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteMezzaninesRequest;
import com.aliyuncs.vod.model.v20170321.DeleteMezzaninesResponse;
/**
 * 批量删除源文件函数
 * @param client 发送请求客户端
 * @return DeleteMezzaninesResponse 批量删除源文件响应数据
 * @throws Exception
*/
public static DeleteMezzaninesResponse deleteMezzanines(DefaultAcsClient client) throws Exc
eption {
    DeleteMezzaninesRequest request = new DeleteMezzaninesRequest();
    //支持传入多个视频ID，多个用逗号分隔
    request.setVideoIds("VideoId1,VideoId2");
    request.setForce(false);
    return client.getAcsResponse(request);
}
//请求示例
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteMezzaninesResponse response = new DeleteMezzaninesResponse();
    try {
        response = deleteMezzanines(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参考UpdateImageInfos。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateImageInfosRequest;
import com.aliyuncs.vod.model.v20170321.UpdateImageInfosResponse;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
/**
 * 批量更新图片信息函数
 * @param client 发送请求客户端
 * @return UpdateImageInfosResponse 批量更新图片信息响应数据
 * @throws Exception
*/
public static UpdateImageInfosResponse updateImageInfos(DefaultAcsClient client) throws Exc
eption{
        UpdateImageInfosRequest request = new UpdateImageInfosRequest();
        JSONArray updateContentArray = new JSONArray();
        JSONObject updateContent1 = new JSONObject();
        updateContent1.put("ImageId", "ImageId1");
//        updateContent1.put("Title", "new Title");
//        updateContent1.put("Tags", "new Tag1,new Tag2");
        updateContentArray.add((updateContent1));
        JSONObject updateContent2 = new JSONObject();
        updateContent2.put("ImageId", "ImageId2");
//        updateContent2.put("Title", "new Title");
//        updateContent2.put("Tags", "new Tag1,new Tag2");
        updateContentArray.add((updateContent2));
        request.setUpdateContent(updateContentArray.toJSONString());
        return client.getAcsResponse(request);
    }
// 请求示例
public static void main(String[] argv) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateImageInfosResponse response = new UpdateImageInfosResponse();
    try {
        response = updateImageInfos(client);
        if (response.getNonExistImageIds() != null && response.getNonExistImageIds().size()
> 0) {
            System.out.print("======nonexistent ImageIds : ======\n");
            for (String imageId : response.getNonExistImageIds()) {
                System.out.print(imageId + "\n");
            }
        }
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetImageInfoRequest;
import com.aliyuncs.vod.model.v20170321.GetImageInfoResponse;
/**
 * 获取图片信息函数
 *
 * @param client 发送请求客户端
 * @return GetImageInfoResponse 获取图片信息响应数据
 * @throws Exception
*/
public static GetImageInfoResponse getImageInfo(DefaultAcsClient client) throws Exception {
    GetImageInfoRequest request = new GetImageInfoRequest();
    request.setImageId("ImageId");
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetImageInfoResponse response = new GetImageInfoResponse();
    try {
        response = getImageInfo(client);
        System.out.print("ImageId=" + response.getImageInfo().getImageId());
        System.out.print("Title=" + response.getImageInfo().getTitle());
        System.out.print("CreationTime=" + response.getImageInfo().getCreationTime());
        System.out.print("FileURL=" + response.getImageInfo().getMezzanine().getFileURL());
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteImageRequest;
import com.aliyuncs.vod.model.v20170321.DeleteImageResponse;
/**
 * 删除图片函数
 *
 * @param client 发送请求客户端
 * @return DeleteImageResponse 删除图片响应数据
 * @throws Exception
*/
public static DeleteImageResponse deleteImage(DefaultAcsClient client) throws Exception {
    DeleteImageRequest request = new DeleteImageRequest();
    //根据ImageURL删除图片文件
    request.setDeleteImageType("ImageURL");
    //ImageURL示例：http://example.aliyundoc.com/cover-****.jpg
    String url = "<your image URL>";
    String encodeUrl = URLEncoder.encode(url, "UTF-8");
    request.setImageURLs(encodeUrl);
    //根据ImageId删除图片文件
    //request.setDeleteImageType("ImageId");
    //request.setImageIds("ImageId1,ImageId2");
    //根据VideoId删除指定ImageType的图片文件
    //request.setDeleteImageType("VideoId");
    //request.setVideoId("VideoId");
    //request.setImageType("SpriteSnapshot");
    return client.getAcsResponse(request);
}
// 请求示例
public static void main(String[] argv) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteImageResponse response = new DeleteImageResponse();
    try {
        response = deleteImage(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

2.6. 媒资分类

本文为您提供了Java SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、获取分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.AddCategoryRequest;
import com.aliyuncs.vod.model.v20170321.AddCategoryResponse;

/*创建分类函数*/
public static AddCategoryResponse addCategory(DefaultAcsClient client) throws Exception {
    AddCategoryRequest request = new AddCategoryRequest();
    // 父分类ID，若不填，则默认生成一级分类，根节点分类ID为-1
    request.setParentId(-1L);
    // 分类名称
    request.setCateName("CateName");
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    AddCategoryResponse response = new AddCategoryResponse();
    try {
        response = addCategory(client);
        // 创建成功的分类信息
        System.out.print("ParentId = " + response.getCategory().getParentId() + "\n");
        System.out.print("CateId = " + response.getCategory().getCateId() + "\n");
        System.out.print("CateName = " + response.getCategory().getCateName() + "\n");
        System.out.print("Level = " + response.getCategory().getLevel() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

更新分类

调用UpdateCategory接口，完成更新分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateCategoryRequest;
import com.aliyuncs.vod.model.v20170321.UpdateCategoryResponse;

/*修改分类函数*/
public static UpdateCategoryResponse updateCategory(DefaultAcsClient client) throws Excepti
on {
    UpdateCategoryRequest request = new UpdateCategoryRequest();
    // 请设置真实分类ID
    request.setCateId(-1L);
    // 分类名称
    request.setCateName("CateName");
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateCategoryResponse response = new UpdateCategoryResponse();
    try {
        response = updateCategory(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteCategoryRequest;
import com.aliyuncs.vod.model.v20170321.DeleteCategoryResponse;

/*删除分类函数*/
public static DeleteCategoryResponse deleteCategory(DefaultAcsClient client) throws Excepti
on {
    DeleteCategoryRequest request = new DeleteCategoryRequest();
    // 请设置待删除分类ID
    request.setCateId(-1L);
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your  AccessKeySecret>"
);
    DeleteCategoryResponse response = new DeleteCategoryResponse();
    try {
        response = deleteCategory(client);
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

获取分类及其子分类

调用GetCategories接口，完成获取分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetCategoriesRequest;
import com.aliyuncs.vod.model.v20170321.GetCategoriesResponse;

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your  AccessKeySecret>"
);
    GetCategoriesResponse response = new GetCategoriesResponse();
    try {
        // 请设置真实分类ID
        Long cateId = -1L;
        // 翻页页号
        Long pageNo = 1L;
        // 每页返回条数
        Long pageSize = 10L;
        response = getCategories(client, cateId, pageNo, pageSize);
        // 指定的分类信息
        System.out.print("ParentId = " + response.getCategory1().getParentId() + "\n");
        System.out.print("CateId = " + response.getCategory1().getCateId() + "\n");
        System.out.print("CateName = " + response.getCategory1().getCateName() + "\n");
        System.out.print("Level = " + response.getCategory1().getLevel() + "\n");
        System.out.print("SubTotal = " + response.getSubTotal()+ "\n");
        // 当前分类的子分类列表
        for (GetCategoriesResponse.Category category : response.getSubCategories()) {
            System.out.print("SubCategories.ParentId = " + category.getParentId() + "\n");
            System.out.print("SubCategories.CateId = " + category.getCateId() + "\n");
            System.out.print("SubCategories.CateName = " + category.getCateName() + "\n");
            System.out.print("SubCategories.Level = " + category.getLevel() + "\n");
        }
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}

/*查询分类及其子分类函数*/
public static GetCategoriesResponse getCategories(DefaultAcsClient client, Long cateId, Lon
g pageNo, Long pageSize) throws Exception {
    GetCategoriesRequest request = new GetCategoriesRequest();
    request.setCateId(cateId);
    request.setPageNo(pageNo);
    request.setPageSize(pageSize);
    return client.getAcsResponse(request);
}
```

2.7. 媒体处理

本文提供了Java SDK媒体处理模块相关功能的API调用示例，包含提交转码作业、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业

调用SubmitTranscodeJobs接口，完成提交转码作业功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SubmitTranscodeJobsRequest;
import com.aliyuncs.vod.model.v20170321.SubmitTranscodeJobsResponse;
/**
 * 提交媒体处理作业
 */
public static SubmitTranscodeJobsResponse submitTranscodeJobs(DefaultAcsClient client) thro
ws Exception {
    SubmitTranscodeJobsRequest request = new SubmitTranscodeJobsRequest();
    //需要转码的视频ID
    request.setVideoId("34a6ca54f5c140eece85a289****");
    //转码模板ID
    request.setTemplateGroupId("e8aa925a9798c630d30cd****");
    //构建需要替换的水印参数(只有需要替换水印相关信息才需要构建)
    JSONObject overrideParams = buildOverrideParams();
    //覆盖参数，暂只支持水印部分参数替换(只有需要替换水印相关信息才需要传递)
    request.setOverrideParams(overrideParams.toJSONString());
    //构建标准加密配置参数(只有标准加密才需要构建)
    JSONObject encryptConfig = buildEncryptConfig(client);
    //HLS标准加密配置(只有标准加密才需要传递)
    request.setEncryptConfig(encryptConfig.toJSONString());
    return client.getAcsResponse(request);
}
/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SubmitTranscodeJobsResponse response = new SubmitTranscodeJobsResponse();
    try {
        response = submitTranscodeJobs(client);
        //任务ID
        System.out.println("JobId = " + response.getTranscodeJobs().get(0).getJobId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
/**
 * 构建HLS标准加密的配置信息
 * @return
 * @throws ClientException
 */
public static JSONObject buildEncryptConfig(DefaultAcsClient client) throws ClientException
{
    //点播给用户在KMS(密钥管理服务)中的Service Key，可在用户密钥管理服务对应的区域看到描述为vod的ser
vice key
```

```
    String serviceKey = "<您的Service Key>";
    //随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥
    //视频标准加密只需要传递密文密钥即可
    //注意：KMS Client建议单独初始化来保证正确的接入区域，可参考VOD初始化方式，传入正确的KMS服务区域
。
    GenerateDataKeyResponse response = generateDataKey(client, serviceKey);
    JSONObject encryptConfig = new JSONObject();
    //解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥都不一
样
    //至于Ciphertext这个解密接口参数的名称，用户可自行制定，这里只作为参考参数名称
    encryptConfig.put("DecryptKeyUri", "http://example.aliyundoc.com/decrypt?" +
            "Ciphertext=" + response.getCiphertextBlob());
    //密钥服务的类型，目前只支持KMS
    encryptConfig.put("KeyServiceType", "KMS");
    //密文密钥
    encryptConfig.put("CipherText", response.getCiphertextBlob());
    return encryptConfig;
}
/**
 * 1、构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；
 * 2、需要替换的水印信息对应水印ID必须是关联在指定的模板ID(即TranscodeTemplateId)中；
 * 3、不支持通过媒体处理接口去增加一个没有关联上的水印
 * 注意：图片水印的文件存储源站需要和发起转码的视频存储源站一致
 * @return
 */
public static JSONObject buildOverrideParams() {
    JSONObject overrideParams = new JSONObject();
    JSONArray watermarks = new JSONArray();
    //图片水印文件地址替换
    JSONObject watermark1 = new JSONObject();
    //模板上面关联需要替换的水印文件图片水印ID
    watermark1.put("WatermarkId", "2ea587477c5a1bc8b57****");
    //需要替换成对应图片水印文件的OSS地址，水印文件存储源站需要和视频存储源站一致
    watermark1.put("FileUrl", "https:192.168.0.1/16");
    //文字水印内容替换
    JSONObject watermark2 = new JSONObject();
    //模板上面关联需要替换内容的文字水印ID
    watermark2.put("WatermarkId", "d297ba31ac5242d207****");
    //需要替换成对应的内容
    watermark2.put("Content", "用户ID：6****");
    watermarks.add(watermark2);
    overrideParams.put("Watermarks", watermarks);
    return overrideParams;
}
/**
 * 生成加密需要的密钥，response中包含密文密钥和明文密钥，用户只需要将密文密钥传递给点播即可
 * 注意：KeySpec 必须传递AES_128，且不能设置NumberOfBytes
 * @param client KMS-SDK客户端
 * @param serviceKey 点播提供生成密钥的service key，在用户的密钥管理服务中可看到描述为vod的加密key
 * @return
 * @throws ClientException
 */
public static GenerateDataKeyResponse generateDataKey(DefaultAcsClient client, String servi
ceKey) throws ClientException {
```

```
    GenerateDataKeyRequest request = new GenerateDataKeyRequest();
    request.setKeyId(serviceKey);
    request.setKeySpec("AES_128");
    return client.getAcsResponse(request);
}
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明** 创建截图模板详细请参见截图模板。

```
import com.aliyuncs.vod.model.v20170321.SubmitSnapshotJobRequest;
import com.aliyuncs.vod.model.v20170321.SubmitSnapshotJobResponse;
/**
 * 提交媒体截图处理作业调用函数
 */
public static SubmitSnapshotJobResponse submitSnapshotJob(DefaultAcsClient client) throws E
xception {
    SubmitSnapshotJobRequest request = new SubmitSnapshotJobRequest();
    //需要截图的视频ID(推荐传递截图模板ID)
    request.setVideoId("4d237a8270084849bf4207876181****");
    //截图模板ID
    request.setSnapshotTemplateId("5d745e6b8baadf589e0702426cfc6****");
    //如果设置了SnapshotTemplateId，会忽略下面参数
    request.setCount(50L);
    request.setSpecifiedOffsetTime(0L);
    request.setInterval(1L);
    request.setWidth("200");
    request.setHeight("200");
    JSONObject spriteSnapshotConfig = buildSnapshotTemplateConfig();
    request.setSpriteSnapshotConfig(spriteSnapshotConfig.toJSONString());
    return client.getAcsResponse(request);
}
/**
 * 构建雪碧图截图配置
 * @return
 */
public static JSONObject buildSnapshotTemplateConfig() {
    JSONObject spriteSnapshotConfig = new JSONObject();
    spriteSnapshotConfig.put("CellWidth", "120");
    spriteSnapshotConfig.put("CellHeight", "68");
    spriteSnapshotConfig.put("Columns", "3");
    spriteSnapshotConfig.put("Lines", "10");
    spriteSnapshotConfig.put("Padding", "20");
    spriteSnapshotConfig.put("Margin", "50");
    //保留雪碧图原始图
    spriteSnapshotConfig.put("KeepCellPic", "keep");
    spriteSnapshotConfig.put("Color", "tomato");
    return spriteSnapshotConfig;
}
/**
```

```
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SubmitSnapshotJobResponse response = new SubmitSnapshotJobResponse();
    try {
        response = submitSnapshotJob(client);
        //任务ID
        System.out.println("JobId = " + response.getSnapshotJob().getJobId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListSnapshotsRequest;
import com.aliyuncs.vod.model.v20170321.ListSnapshotsResponse;
/**
 * 查询截图数据
 */
public static ListSnapshotsResponse listSnapshots(DefaultAcsClient client) throws Exception
{
    ListSnapshotsRequest request = new ListSnapshotsRequest();
    //视频ID
    request.setVideoId("c86c0ceba9796535****");
    //截图类型
    request.setSnapshotType("CoverSnapshot");
    request.setPageNo("1");
    request.setPageSize("20");
    return client.getAcsResponse(request);
}
/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListSnapshotsResponse response = new ListSnapshotsResponse();
    try {
        response = listSnapshots(client);
        //截图地址
        System.out.println("Url = " + response.getMediaSnapshot().getSnapshots().get(0).get
Url());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SubmitPreprocessJobsRequest;
import com.aliyuncs.vod.model.v20170321.SubmitPreprocessJobsResponse;
/**
 * 导播台视频预处理
 */
public static SubmitPreprocessJobsResponse submitPreprocessJobs(DefaultAcsClient client) th
rows Exception {
    SubmitPreprocessJobsRequest request = new SubmitPreprocessJobsRequest();
    //视频ID
    request.setVideoId("c86c0ceb8db54ae09796535****");
    request.setPreprocessType("PreprocessType");
    return client.getAcsResponse(request);
}
/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SubmitPreprocessJobsResponse response = new SubmitPreprocessJobsResponse();
    try {
        response = submitPreprocessJobs(client);
        //任务ID
        System.out.println("JobId = " + response.getPreprocessJobs().get(0).getJobId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

2.8. 视频水印

本文为您提供了Java SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

> **说明**
>
> - 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
> - 水印文件OSS上传详细参数请参见OSS文件上传。

```
import com.aliyuncs.vod.model.v20170321.AddWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.AddWatermarkResponse;
```

```
/**
 * 添加水印配置信息函数
 */
public static AddWatermarkResponse addWatermark(DefaultAcsClient client) throws Exception {
    AddWatermarkRequest request = new AddWatermarkRequest();
    //水印名称
    request.setName("addwatermark");
    //获取水印文件在oss的URL，例如：http://example.oss-cn-shanghai.aliyuncs.com/watermark/test
-****.png
    String fileUrl = "<your watermarkFile URL>";
    //图片水印必传图片文件的oss文件地址，水印文件必须和视频在同一个区域，例如:华东2视频，水印文件必须存
放在华东2
    request.setFileUrl(fileUrl);
    //水印配置数据
    JSONObject watermarkConfig = null;
    //图片水印的位置配置数据
    watermarkConfig = buildImageWatermarkConfig();

    //文字水印的位置配置数据
    //watermarkConfig = buildTextWatermarkConfig();
    request.setWatermarkConfig(watermarkConfig.toJSONString());

    //文字水印:Text; 图片水印:Image
    request.setType("Image");
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    AddWatermarkResponse response = new AddWatermarkResponse();
    try {
        //添加水印信息
        response = addWatermark(client);
        //水印ID
        System.out.println("WatermarkId = " + response.getWatermarkInfo().getWatermarkId())
;
        //水印的位置、效果配置数据
        System.out.println("WatermarkConfig = " + response.getWatermarkInfo().getWatermarkC
onfig());
        //水印文件的FileUrl(文字水印该字段值为空)
        System.out.println("FileUrl = " + response.getWatermarkInfo().getFileUrl());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

```
/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
public static JSONObject buildImageWatermarkConfig() {
    JSONObject watermarkConfig = new JSONObject();
    //水印的横向偏移距离
    watermarkConfig.put("Dx", "8");
    //水印的纵向偏移距离
    watermarkConfig.put("Dy", "8");
    //水印显示的宽
    watermarkConfig.put("Width", "55");
    //水印显示的高
    watermarkConfig.put("Height", "55");
    //水印显示的相对位置(左上、右上、左下、右下)
    watermarkConfig.put("ReferPos", "BottomRight");
    //水印显示的时间线(开始显示和结束显示时间)
    JSONObject timeline = new JSONObject();
    //水印开始显示时间
    timeline.put("Start", "2");
    //水印结束显示时间
    timeline.put("Duration", "ToEND");
    watermarkConfig.put("Timeline", timeline);
    return watermarkConfig;
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
public static JSONObject buildTextWatermarkConfig() {
    JSONObject watermarkConfig = new JSONObject();
    //文字水印显示的内容
    watermarkConfig.put("Content", "testwatermark");
    //文字水印的字体名称
    watermarkConfig.put("FontName", "SimSun");
    //文字水印的字体大小
    watermarkConfig.put("FontSize", 25);
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    watermarkConfig.put("FontColor", "Black");
    //文字水印的透明度
    watermarkConfig.put("FontAlpha", "0.2");
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    watermarkConfig.put("BorderColor", "White");
    //文字水印的描边宽度
    watermarkConfig.put("BorderWidth", 1);
    //文字水印距离视频画面上边的偏移距离
    watermarkConfig.put("Top", 20);
    //文字水印距离视频画面左边的偏移距离
    watermarkConfig.put("Left", 15);
    return watermarkConfig;
}
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.UpdateWatermarkResponse;

/**
 * 修改水印配置信息函数
 * 注意：不支持图片文件地址的修改，如果更换请创建新的水印信息
 */
public static UpdateWatermarkResponse updateWatermark(DefaultAcsClient client) throws Excep
tion {
    UpdateWatermarkRequest request = new UpdateWatermarkRequest();
    request.setName("updatewatermark");
    //需要更新配置信息的水印ID
    request.setWatermarkId("421ddddd4f6e734a526fd2****");
    //水印配置数据
    JSONObject watermarkConfig = null;
    //图片水印的位置配置数据
    //watermarkConfig = buildImageWatermarkConfig();
    //文字水印的位置配置数据
    watermarkConfig = buildTextWatermarkConfig();
    request.setWatermarkConfig(watermarkConfig.toJSONString());
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateWatermarkResponse response = new UpdateWatermarkResponse();
    try {
        response = updateWatermark(client);
        //水印ID
        System.out.println("WatermarkId = " + response.getWatermarkInfo().getWatermarkId())
;
        //水印的位置、效果配置数据
        System.out.println("WatermarkConfig = " + response.getWatermarkInfo().getWatermarkC
onfig());
        //水印文件的FileUrl(文字水印该字段值为空)
        System.out.println("FileUrl = " + response.getWatermarkInfo().getFileUrl());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}

/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
```

```
 */
public static JSONObject buildImageWatermarkConfig() {
    JSONObject watermarkConfig = new JSONObject();
    //水印的横向偏移距离
    watermarkConfig.put("Dx", "10");
    //水印的纵向偏移距离
    watermarkConfig.put("Dy", "10");
    //水印显示的宽
    watermarkConfig.put("Width", "66");
    //水印显示的高
    watermarkConfig.put("Height", "66");
    //水印显示的相对位置(左上、右上、左下、右下)
    watermarkConfig.put("ReferPos", "BottomLeft");
    //水印显示的时间线(开始显示和结束显示时间)
    JSONObject timeline = new JSONObject();
    //水印开始显示时间
    timeline.put("Start", "2");
    //水印结束显示时间
    timeline.put("Duration", "ToEND");
    watermarkConfig.put("Timeline", timeline);
    return watermarkConfig;
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
public static JSONObject buildTextWatermarkConfig() {
    JSONObject watermarkConfig = new JSONObject();
    //文字水印显示的内容
    watermarkConfig.put("Content", "testwatermark");
    //文字水印的字体名称
    watermarkConfig.put("FontName", "SimSun");
    //文字水印的字体大小
    watermarkConfig.put("FontSize", 40);
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    watermarkConfig.put("FontColor", "Black");
    //文字水印的透明度
    watermarkConfig.put("FontAlpha", "0.2");
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    watermarkConfig.put("BorderColor", "White");
    //文字水印的描边宽度
    watermarkConfig.put("BorderWidth", 2);
    //文字水印距离视频画面上边的偏移距离
    watermarkConfig.put("Top", 20);
    //文字水印距离视频画面左边的偏移距离
    watermarkConfig.put("Left", 15);
    return watermarkConfig;
}
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.DeleteWatermarkResponse;

/**
 *删除水印配置信息函数
 */
public static DeleteWatermarkResponse deleteWatermark(DefaultAcsClient client) throws Excep
tion {
    DeleteWatermarkRequest request = new DeleteWatermarkRequest();
    request.setWatermarkId("53f9d796fad9d7b862b2e****");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteWatermarkResponse response = new DeleteWatermarkResponse();
    try {
        response = deleteWatermark(client);
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.ListWatermarkResponse;

/**
 * 查询水印配置信息列表函数
 */
public static ListWatermarkResponse listWatermark(DefaultAcsClient client) throws Exception
{
    ListWatermarkRequest request = new ListWatermarkRequest();
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListWatermarkResponse response = new ListWatermarkResponse();
    try {
        response = listWatermark(client);
        for (ListWatermarkResponse.WatermarkInfo watermarkInfo : response.getWatermarkInfos
()) {
            //水印ID
            System.out.println("WatermarkId = " + watermarkInfo.getWatermarkId());
            //水印的位置、效果配置数据
            System.out.println("WatermarkConfig = " + watermarkInfo.getWatermarkConfig());
            //水印文件的FileUrl(文字水印该字段值为空)
            System.out.println("FileUrl = " + watermarkInfo.getFileUrl());
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.GetWatermarkResponse;

/**
 * 查询单个水印配置信息函数
 */
public static GetWatermarkResponse getWatermark(DefaultAcsClient client) throws Exception {
    GetWatermarkRequest request = new GetWatermarkRequest();
    //需要查询水印信息的水印ID
    request.setWatermarkId("96d4f6e734a526fd2****");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetWatermarkResponse response = new GetWatermarkResponse();
    try {
        response = getWatermark(client);
        //水印ID
        System.out.println("WatermarkId = " + response.getWatermarkInfo().getWatermarkId())
;
        //水印的位置、效果配置数据
        System.out.println("WatermarkConfig = " + response.getWatermarkInfo().getWatermarkC
onfig());
        //水印文件的FileUrl(文字水印该字段值为空)
        System.out.println("FileUrl = " + response.getWatermarkInfo().getFileUrl());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SetDefaultWatermarkRequest;
import com.aliyuncs.vod.model.v20170321.SetDefaultWatermarkResponse;

/**
 * 设置默认水印配置信息函数
 */
public static SetDefaultWatermarkResponse setDefaultWatermark(DefaultAcsClient client) thro
ws Exception {
    SetDefaultWatermarkRequest request = new SetDefaultWatermarkRequest();
    //设置默认的水印ID
    request.setWatermarkId("82105a29c6e96d4f6****");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SetDefaultWatermarkResponse response = new SetDefaultWatermarkResponse();
    try {
        response = setDefaultWatermark(client);
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

2.9. 转码模板

本篇文档提供了Java SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.AddTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.AddTranscodeTemplateGroupResponse;

/**
 * 添加转码模板组配置
 */
public static AddTranscodeTemplateGroupResponse addTranscodeTemplateGroup(DefaultAcsClient
client) throws Exception {
    AddTranscodeTemplateGroupRequest request = new AddTranscodeTemplateGroupRequest();
    //转码模板ID
```

```
    request.setName("grouptest");
    request.setTranscodeTemplateList(buildTranscodeTemplateList().toJSONString());
    return client.getAcsResponse(request);
}

/**
 * 构建需要添加的转码模板配置数据
 *
 * @return
 */
public static JSONArray buildTranscodeTemplateList() {
    JSONObject transcodeTemplate = new JSONObject();
    //模板名称
    transcodeTemplate.put("TemplateName", "testtemplate");

    //清晰度
    transcodeTemplate.put("Definition", "LD");

    //视频流转码配置
    JSONObject video = new JSONObject();
    video.put("Width", 640);
    video.put("Bitrate", 400);
    video.put("Fps", 25);
    video.put("Remove", false);
    video.put("Codec", "H.264");
    video.put("Gop", "250");
    transcodeTemplate.put("Video", video);

    //音频流转码配置
    JSONObject audio = new JSONObject();
    audio.put("Codec", "AAC");
    audio.put("Bitrate", "64");
    audio.put("Channels", "2");
    audio.put("Samplerate", "32000");
    transcodeTemplate.put("Audio", audio);

    //封装容器
    JSONObject container = new JSONObject();
    container.put("Format", "mp4");
    transcodeTemplate.put("Container", container);

    //条件转码配置
    JSONObject transconfig = new JSONObject();
    transconfig.put("IsCheckReso", false);
    transconfig.put("IsCheckResoFail", false);
    transconfig.put("IsCheckVideoBitrate", false);
    transconfig.put("IsCheckVideoBitrateFail", false);
    transconfig.put("IsCheckAudioBitrate", false);
    transconfig.put("IsCheckAudioBitrateFail", false);
    transcodeTemplate.put("TransConfig", transconfig);

    //加密配置(只支持m3u8)
    //JSONObject encryptSetting = new JSONObject();
    //encryptSetting.put("EncryptType", "Private");
```

```
    //transcodeTemplate.put("EncryptSetting", encryptSetting);

    //水印ID(多水印关联)
    JSONArray watermarkIdList = new JSONArray();
    watermarkIdList.add("263261bdc1ff65782f8995c6dd22a16a");
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    watermarkIdList.add("USER_DEFAULT_WATERMARK");
    transcodeTemplate.put("WatermarkIds", watermarkIdList);

    JSONArray transcodeTemplateList = new JSONArray();
    transcodeTemplateList.add(transcodeTemplate);
    return transcodeTemplateList;
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    AddTranscodeTemplateGroupResponse response = new AddTranscodeTemplateGroupResponse();
    try {
        response = addTranscodeTemplateGroup(client);
        System.out.println("TranscodeTemplateGroupId = " + response.getTranscodeTemplateGro
upId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.UpdateTranscodeTemplateGroupResponse;

/**
 * 修改转码模板组配置
 */
public static UpdateTranscodeTemplateGroupResponse updateTranscodeTemplateGroup(DefaultAcsC
lient client) throws Exception {
    UpdateTranscodeTemplateGroupRequest request = new UpdateTranscodeTemplateGroupRequest()
;
    request.setName("grouptest1");
    //转码模板组ID
    request.setTranscodeTemplateGroupId("4c71a339fecec0152b4fa6f452****");
    request.setTranscodeTemplateList(buildTranscodeTemplateList().toJSONString());
    return client.getAcsResponse(request);
}
```

```
/**
 * 构建需要修改的转码模板配置数据
 *
 * @return
 */
public static JSONArray buildTranscodeTemplateList() {
    JSONObject transcodeTemplate = new JSONObject();
    //转码模板ID
    transcodeTemplate.put("TranscodeTemplateId", "85c2b18ac08fda33e8f6d9c56242f");

     //模板名称
    transcodeTemplate.put("TemplateName", "testtemplate");

    //视频流转码配置
    JSONObject video = new JSONObject();
    video.put("Width", 960);
    video.put("Bitrate", 900);
    video.put("Fps", 25);
    video.put("Remove", false);
    video.put("Codec", "H.264");
    video.put("Gop", "250");
    transcodeTemplate.put("Video", video);

    //音频流转码配置
    JSONObject audio = new JSONObject();
    audio.put("Codec", "AAC");
    audio.put("Bitrate", "96");
    audio.put("Channels", "2");
    audio.put("Samplerate", "32000");
    transcodeTemplate.put("Audio", audio);

    //封装容器
    JSONObject container = new JSONObject();
    container.put("Format", "mp4");
    transcodeTemplate.put("Container", container);

    //条件转码配置
    JSONObject transconfig = new JSONObject();
    transconfig.put("IsCheckReso", false);
    transconfig.put("IsCheckResoFail", false);
    transconfig.put("IsCheckVideoBitrate", false);
    transconfig.put("IsCheckVideoBitrateFail", false);
    transconfig.put("IsCheckAudioBitrate", false);
    transconfig.put("IsCheckAudioBitrateFail", false);
    transcodeTemplate.put("TransConfig", transconfig);

    //加密配置(只支持m3u8)
    JSONObject encryptSetting = new JSONObject();
    encryptSetting.put("EncryptType", "Private");

    //水印ID(多水印关联)
    JSONArray watermarkIdList = new JSONArray();
    watermarkIdList.add("263261bdc1ff65782f95c6dd22a16a");
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    watermarkIdList.add("USER_DEFAULT_WATERMARK");
```

```
    watermarkIdList.add("USER_DEFAULT_WATERMARK");
    transcodeTemplate.put("WatermarkIds", watermarkIdList);

    JSONArray transcodeTemplateList = new JSONArray();
    transcodeTemplateList.add(transcodeTemplate);
    return transcodeTemplateList;
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateTranscodeTemplateGroupResponse response = new UpdateTranscodeTemplateGroupRespons
e();
    try {
        response = updateTranscodeTemplateGroup(client);
        System.out.println("TranscodeTemplateGroupId = " + response.getTranscodeTemplateGro
upId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.ListTranscodeTemplateGroupResponse;

/**
 * 查询转码模板组列表
 */
public static ListTranscodeTemplateGroupResponse listTranscodeTemplateGroup(DefaultAcsClien
t client) throws Exception {
    ListTranscodeTemplateGroupRequest request = new ListTranscodeTemplateGroupRequest();
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListTranscodeTemplateGroupResponse response = new ListTranscodeTemplateGroupResponse();
    try {
        response = listTranscodeTemplateGroup(client);
        System.out.println("TranscodeTemplateGroupId = " + response.getTranscodeTemplateGro
upList().get(0).getTranscodeTemplateGroupId());
        System.out.println("GroupName = " + response.getTranscodeTemplateGroupList().get(0)
.getName());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.GetTranscodeTemplateGroupResponse;

/**
 * 查询单个转码模板组配置
 */
public static GetTranscodeTemplateGroupResponse getTranscodeTemplateGroup(DefaultAcsClient
client) throws Exception {
    GetTranscodeTemplateGroupRequest request = new GetTranscodeTemplateGroupRequest();
    request.setTranscodeTemplateGroupId("a0fa0fda545e50e7a3eb75491f9f4");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
```

```
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetTranscodeTemplateGroupResponse response = new GetTranscodeTemplateGroupResponse();
    try {
        response = getTranscodeTemplateGroup(client);
       if (response.getTranscodeTemplateGroup().getTranscodeTemplateList().size() > 0) {
                //转码模板组ID
                System.out.println("TranscodeTemplateGroupId = " + response.getTranscodeTem
plateGroup().getTranscodeTemplateGroupId());
                //转码模板组名称
                System.out.println("Name = " + response.getTranscodeTemplateGroup().getName
());
                //是否是默认转码模板组
                System.out.println("Name = " + response.getTranscodeTemplateGroup().getIsDe
fault());
                //转码模板视频转码配置
                System.out.println("Video = " + response.getTranscodeTemplateGroup().getTra
nscodeTemplateList().get(0).getVideo());
                //转码模板音频转码配置
                System.out.println("Audio = " + response.getTranscodeTemplateGroup().getTra
nscodeTemplateList().get(0).getAudio());
                //转码模板清晰度标记
                System.out.println("Definition = " + response.getTranscodeTemplateGroup().g
etTranscodeTemplateList().get(0).getDefinition());
                //转码需要压制的水印对应ID
                System.out.println("WatermarkIds = " + response.getTranscodeTemplateGroup()
.getTranscodeTemplateList().get(0).getWatermarkIds());
                //转码模板封装格式配置
                System.out.println("Container = " + response.getTranscodeTemplateGroup().ge
tTranscodeTemplateList().get(0).getContainer());
                //转码模板ID
                System.out.println("TranscodeTemplateId = " + response.getTranscodeTemplate
Group().getTranscodeTemplateList().get(0).getTranscodeTemplateId());
                //转码模板名称
                System.out.println("TemplateName = " + response.getTranscodeTemplateGroup()
.getTranscodeTemplateList().get(0).getTemplateName());
            }
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SetDefaultTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.SetDefaultTranscodeTemplateGroupResponse;

/**
 * 设置默认转码模板组
 */
public static SetDefaultTranscodeTemplateGroupResponse setDefaultTranscodeTemplateGroup(Def
aultAcsClient client) throws Exception {
    SetDefaultTranscodeTemplateGroupRequest request = new SetDefaultTranscodeTemplateGroupR
equest();
    request.setTranscodeTemplateGroupId("ecf03526c945ae022165c469f4548e");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    SetDefaultTranscodeTemplateGroupResponse response = new SetDefaultTranscodeTemplateGrou
pResponse();
    try {
        response = setDefaultTranscodeTemplateGroup(client);
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteTranscodeTemplateGroupRequest;
import com.aliyuncs.vod.model.v20170321.DeleteTranscodeTemplateGroupResponse;

/**
 * 删除转码模板组配置
 */
public static DeleteTranscodeTemplateGroupResponse deleteTranscodeTemplateGroup(DefaultAcsC
lient client) throws Exception {
    DeleteTranscodeTemplateGroupRequest request = new DeleteTranscodeTemplateGroupRequest()
;
    request.setTranscodeTemplateGroupId("a0fa0fda545e50e91a3eb75491****");
    request.setForceDelGroup("false");
    request.setTranscodeTemplateIds("ddddddd,ffffffff");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    DeleteTranscodeTemplateGroupResponse response = new DeleteTranscodeTemplateGroupRespons
e();
    try {
        response = deleteTranscodeTemplateGroup(client);
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

2.10. 截图模板

本篇文档提供了Java SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.vod.model.v20170321.AddVodTemplateRequest;
import com.aliyuncs.vod.model.v20170321.AddVodTemplateResponse;

/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
```

```
 * （以下代码示例为雪碧图完整配置）
 * @return
 */
public static JSONObject buildSnapshotTemplateConfig() {
    JSONObject templateConfig = new JSONObject();
    JSONObject snapshotConfig = new JSONObject();
    snapshotConfig.put("Count", "50");
    snapshotConfig.put("Interval", "1");
    snapshotConfig.put("SpecifiedOffsetTime", "0");
    snapshotConfig.put("Width", "200");
    snapshotConfig.put("Height", "200");
    snapshotConfig.put("FrameType", "normal");
    //普通截图配置（与雪碧图原始图配置共用该配置信息）
    templateConfig.put("SnapshotConfig", snapshotConfig);

    //雪碧图配置（雪碧图配置必须是建立在普通截图配置之上）
    JSONObject spriteSnapshotConfig = new JSONObject();
    spriteSnapshotConfig.put("CellWidth", "120");
    spriteSnapshotConfig.put("CellHeight", "68");
    spriteSnapshotConfig.put("Columns", "3");
    spriteSnapshotConfig.put("Lines", "10");
    spriteSnapshotConfig.put("Padding", "20");
    spriteSnapshotConfig.put("Margin", "50");
    spriteSnapshotConfig.put("KeepCellPic", "keep");
    spriteSnapshotConfig.put("Color", "tomato");
    snapshotConfig.put("SpriteSnapshotConfig", spriteSnapshotConfig);

    //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
    templateConfig.put("SnapshotType", "SpriteSnapshot");
    return templateConfig;
}

/**
 * 添加截图模板函数
 */
public static AddVodTemplateResponse addSnapshotVodTemplate(DefaultAcsClient client) throws
Exception {
    AddVodTemplateRequest request = new AddVodTemplateRequest();
    //模板名称
    request.setName("截图模板添加测试");
    //模板类型，固定值为Snapshot
    request.setTemplateType("Snapshot");
    //截图模板配置数据生成
    JSONObject templateConfig = buildSnapshotTemplateConfig();
    request.setTemplateConfig(templateConfig.toJSONString());
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) {
```

```
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    AddVodTemplateResponse response = new AddVodTemplateResponse();
    try {
        //添加截图模板
        response = addSnapshotVodTemplate(client);
        //截图模板ID
        System.out.println("SnapshotVodTemplateId = " + response.getVodTemplateId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.vod.model.v20170321.UpdateVodTemplateRequest;
import com.aliyuncs.vod.model.v20170321.UpdateVodTemplateResponse;

/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
 * （以下代码为普通截图完整配置）
 * @return
 */
public static JSONObject buildSnapshotTemplateConfig() {
    JSONObject templateConfig = new JSONObject();
    JSONObject snapshotConfig = new JSONObject();
    snapshotConfig.put("Count", "50");
    snapshotConfig.put("Interval", "1");
    snapshotConfig.put("SpecifiedOffsetTime", "0");
    snapshotConfig.put("Width", "200");
    snapshotConfig.put("Height", "200");
    snapshotConfig.put("FrameType", "normal");
    //普通截图配置（与雪碧图原始图配置共用该配置信息）
    templateConfig.put("SnapshotConfig", snapshotConfig);

    //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
    templateConfig.put("SnapshotType", "NormalSnapshot");
    return templateConfig;
}

/**
 * 修改截图模板函数
 */
public static UpdateVodTemplateResponse updateSnapshotVodTemplate(DefaultAcsClient client)
throws Exception {
    UpdateVodTemplateRequest request = new UpdateVodTemplateRequest();
    //设置要修改的模板ID
```

```
    request.setVodTemplateId("53azf9d796fad9d7b862b2e****");
    //模板名称
    request.setName("截图模板修改测试");
    //截图模板配置数据生成
    JSONObject templateConfig = buildSnapshotTemplateConfig();
    request.setTemplateConfig(templateConfig.toJSONString());
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    UpdateVodTemplateResponse response = new UpdateVodTemplateResponse();
    try {
        //修改截图模板
        response = updateSnapshotVodTemplate(client);
        //截图模板ID
        System.out.println("SnapshotVodTemplateId = " + response.getVodTemplateId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteVodTemplateRequest;
import com.aliyuncs.vod.model.v20170321.DeleteVodTemplateResponse;

/**
 * 删除截图模板函数
 */
public static DeleteVodTemplateResponse deleteSnapshotVodTemplate(DefaultAcsClient client)
throws Exception
    DeleteVodTemplateRequest request = new DeleteVodTemplateRequest();
    //设置要删除的模板ID
    request.setVodTemplateId("53azf9d796fad9d7b862b2e5e5b");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) {
DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    DeleteVodTemplateResponse response = new DeleteVodTemplateResponse();
    try {
        //删除截图模板
        response = deleteSnapshotVodTemplate(client);
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListVodTemplateRequest;
import com.aliyuncs.vod.model.v20170321.ListVodTemplateResponse;

/**
 * 查询截图模板列表函数
 */
public static ListVodTemplateResponse listSnapshotVodTemplate(DefaultAcsClient client) thro
ws Exception {
    ListVodTemplateRequest request = new ListVodTemplateRequest();
    //模板类型，固定值为Snapshot
    request.setTemplateType("Snapshot");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListVodTemplateResponse response = new ListVodTemplateResponse();
    try {
        //查询截图模板列表
        response = listSnapshotVodTemplate(client);
        //打印模板数目
        System.out.println("SnapshotVodTemplateId = " + response.getVodTemplateInfoList().s
ize());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。 接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetVodTemplateRequest;
import com.aliyuncs.vod.model.v20170321.GetVodTemplateResponse;

/**
* 查询截图模板函数
*/
public static GetVodTemplateResponse getSnapshotVodTemplate(DefaultAcsClient client) throws
Exception {
    GetVodTemplateRequest request = new GetVodTemplateRequest();
    //设置要查询的模板ID
    request.setVodTemplateId("53azf9d796fad9d7b862b2e****");
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetVodTemplateResponse response = new GetVodTemplateResponse();
    try {
        //查询截图模板
        response = getSnapshotVodTemplate(client);
        //打印返回信息中的截图模板ID
        System.out.println("SnapshotVodTemplateId = " + response.getVodTemplateInfo().getVo
dTemplateId());
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

2.11. 视频剪辑

本篇文档提供了Java SDK视频剪辑模块相关功能的API调用示例。包含创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、发起剪辑合成等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

说明

通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
import com.aliyuncs.vod.model.v20170321.ProduceEditingProjectVideoRequest;
import com.aliyuncs.vod.model.v20170321.ProduceEditingProjectVideoResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        ProduceEditingProjectVideoRequest request = new ProduceEditingProjectVideoRequest()
;
        // Build Editing Project Timeline
        request.setTimeline(buildTimeline());
        // Set Produce Media Metadata
        request.setMediaMetadata(buildMediaMetadata());
        // Set Produce Configuration
        request.setProduceConfig(buildProduceConfig());

        ProduceEditingProjectVideoResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                // Produce Media ID
                System.out.println("MediaId:" + response.getMediaId());
                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }

    public static String buildMediaMetadata(){
        JSONObject mediaMetadata = new JSONObject();
        // Produce Media Title
        mediaMetadata.put("Title", "Title");
        // Produce Media Description
        mediaMetadata.put("Description", "Description");
        // Produce Media UserDefined Cover URL
        mediaMetadata.put("CoverURL", "http://192.168.0.0/16/media/cover/mediaid.jpg");
        // Produce Media Category ID
        mediaMetadata.put("CateId", null);
```

```
        // Produce Media Category Name
        mediaMetadata.put("Tags", "Tag1,Tag2,Test");
        return mediaMetadata.toString();
    }

    public static String buildProduceConfig(){
        JSONObject produceConfig = new JSONObject();
        /*
         The produce process can generate media mezzanine file. You can use the mezzanine f
ile to transcode other media files, just like the transcode process after file upload finis
hed. This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
         1. Not required
         2. Use default transcode template group id when empty
         */
        produceConfig.put("TemplateGroupId", null);
        return produceConfig.toString();
    }

    /**
     * This Sample shows how to merge two videos
     */
    public static String buildTimeline(){
        JSONObject timeline = new JSONObject();

        // Video Track
        JSONArray videoTracks = new JSONArray();
        JSONObject videoTrack = new JSONObject();

        // Video Track Clips
        JSONArray videoTrackClips = new JSONArray();
        JSONObject videoTrackClip1 = new JSONObject();
        videoTrackClip1.put("MediaId", "11119b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip1);
        JSONObject videoTrackClip2 = new JSONObject();
        videoTrackClip2.put("MediaId", "22229b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip2);

        videoTrack.put("VideoTrackClips", videoTrackClips);
        videoTracks.add(videoTrack);

        timeline.put("VideoTracks", videoTracks);

        return timeline.toString();
    }
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ProduceEditingProjectVideoRequest;
import com.aliyuncs.vod.model.v20170321.ProduceEditingProjectVideoResponse;
```

```
    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        ProduceEditingProjectVideoRequest request = new ProduceEditingProjectVideoRequest()
;
        // Set Editing Project ID need to produce
        request.setProjectId("ProjectId");
        // Set Produce Media Metadata
        request.setMediaMetadata(buildMediaMetadata());
        // Set Produce Configuration
        request.setProduceConfig(buildProduceConfig());

        ProduceEditingProjectVideoResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                // Produce Media ID
                System.out.println("MediaId:" + response.getMediaId());
                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }

    public static String buildMediaMetadata(){
        JSONObject mediaMetadata = new JSONObject();
        // Produce Media Title
        mediaMetadata.put("Title", "Title");
        // Produce Media Description
        mediaMetadata.put("Description", "Description");
        // Produce Media UserDefined Cover URL
        mediaMetadata.put("CoverURL", "http://192.168.0.0/16/media/cover/mediaid.jpg");
        // Produce Media Category ID
        mediaMetadata.put("CateId", null);
        // Produce Media Category Name
        mediaMetadata.put("Tags", "Tag1,Tag2,Test");
        return mediaMetadata.toString();
    }

    public static String buildProduceConfig(){
        JSONObject produceConfig = new JSONObject();
```

```
        /*
         The produce process can generate media mezzanine file. You can use the mezzanine f
ile to transcode other media files, just like the transcode process after file upload finis
hed. This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
         1. Not required
         2. Use default transcode template group id when empty
         */
        produceConfig.put("TemplateGroupId", null);
        return produceConfig.toString();
    }
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.AddEditingProjectRequest;
import com.aliyuncs.vod.model.v20170321.AddEditingProjectResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        AddEditingProjectRequest request = new AddEditingProjectRequest();
        // Build Editing Project Timeline
        request.setTimeline(buildTimeline());
        // Set Editing Project Title
        request.setTitle("Editing Project Title");
        // Set Editing Project Description
        request.setDescription("Editing Project Description");
        // Set Editing Project Cover URL
        request.setCoverURL("http://192.168.0.0/16/editingproject/cover/projectid.jpg");

        AddEditingProjectResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                if (response.getProject() != null){
                    System.out.println("Editing Project ID:" + response.getProject().getPro
jectId());
                    System.out.println("Editing Project Title:" + response.getProject().get
Title());
                    System.out.println("Editing Project Create Time:" + response.getProject
().getCreationTime());
                    System.out.println("Editing Project Description:" + response.getProject
().getDescription());
                    System.out.println("Editing Project Status:" + response.getProject().ge
tStatus());
                }

                // Request ID
```

```
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }

    /**
     * This Sample shows how to merge two videos
     */
    public static String buildTimeline(){
        JSONObject timeline = new JSONObject();

        // Video Track
        JSONArray videoTracks = new JSONArray();
        JSONObject videoTrack = new JSONObject();

        // Video Track Clips
        JSONArray videoTrackClips = new JSONArray();
        JSONObject videoTrackClip1 = new JSONObject();
        videoTrackClip1.put("MediaId", "11119b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip1);
        JSONObject videoTrackClip2 = new JSONObject();
        videoTrackClip2.put("MediaId", "22229b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip2);

        videoTrack.put("VideoTrackClips", videoTrackClips);
        videoTracks.add(videoTrack);

        timeline.put("VideoTracks", videoTracks);

        return timeline.toString();
    }
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.UpdateEditingProjectRequest;
import com.aliyuncs.vod.model.v20170321.UpdateEditingProjectResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
```

```
t>");

        UpdateEditingProjectRequest request = new UpdateEditingProjectRequest();
        request.setProjectId("YourProjectId");
        // Set Editing Project Timeline
        request.setTimeline(buildTimeline());
        // Set Editing Project Title
        request.setTitle("Editing Project Title");
        // Set Editing Project Description
        request.setDescription("Editing Project Description");
        // Set Editing Project Cover URL
        request.setCoverURL("http://192.168.0.0/16/editingproject/cover/projectid.jpg");

        UpdateEditingProjectResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }

    /**
     * This Sample shows how to merge two videos
     */
    public static String buildTimeline(){
        JSONObject timeline = new JSONObject();

        // Video Track
        JSONArray videoTracks = new JSONArray();
        JSONObject videoTrack = new JSONObject();

        // Video Track
        JSONArray videoTrackClips = new JSONArray();
        JSONObject videoTrackClip1 = new JSONObject();
        videoTrackClip1.put("MediaId", "11119b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip1);
        JSONObject videoTrackClip2 = new JSONObject();
        videoTrackClip2.put("MediaId", "22229b4d7cf14dc7b83b0e801cbe****");
        videoTrackClips.add(videoTrackClip2);

        videoTrack.put("VideoTrackClips", videoTrackClips);
        videoTracks.add(videoTrack);

        timeline.put("VideoTracks", videoTracks);
```

```
        return timeline.toString();
    }
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteEditingProjectRequest;
import com.aliyuncs.vod.model.v20170321.DeleteEditingProjectResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        DeleteEditingProjectRequest request = new DeleteEditingProjectRequest();
        // Use comma to split Multi Editing Project IDs
        request.setProjectIds("projectid1,projectid2");
        DeleteEditingProjectResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetEditingProjectRequest;
import com.aliyuncs.vod.model.v20170321.GetEditingProjectResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        GetEditingProjectRequest request = new GetEditingProjectRequest();
        request.setProjectId("projectId");

        GetEditingProjectResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                if (response.getProject() != null){
                    System.out.println("Editing Project ID:" + response.getProject().getPro
jectId());
                    System.out.println("Editing Project Timeline:" + response.getProject().
getTimeline());
                    System.out.println("Editing Project Title:" + response.getProject().get
Title());
                    System.out.println("Editing Project Description:" + response.getProject
().getDescription());
                    System.out.println("Editing Project Cover URL:" + response.getProject()
.getCoverURL());
                    System.out.println("Editing Project Create Time:" + response.getProject
().getCreationTime());
                    System.out.println("Editing Project Last Modified Time:" + response.get
Project().getModifiedTime());
                    System.out.println("Editing Project Status:" + response.getProject().ge
tStatus());
                }

                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SearchEditingProjectRequest;
import com.aliyuncs.vod.model.v20170321.SearchEditingProjectResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        SearchEditingProjectRequest request = new SearchEditingProjectRequest();
        request.setTitle("Title Keywords");
        request.setStartTime("2017-01-11T12:00:00Z");
        request.setEndTime("2017-01-12T12:00:00Z");
        request.setPageSize(10);
        request.setPageNo(1);

        SearchEditingProjectResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                if (response.getProjectList() != null && response.getProjectList().size() >
0){
                    System.out.println("Total Size:" + response.getProjectList().size());
                    for (int i=0; i<response.getProjectList().size(); i++){
                        System.out.println("No " + i + " Editing Project");

                        SearchEditingProjectResponse.Project editingProject = response.getP
rojectList().get(i);
                        System.out.println("Editing Project ID:" + editingProject.getProjec
tId());
                        System.out.println("Editing Project Title:" + editingProject.getTit
le());
                        System.out.println("Editing Project Description:" + editingProject.
getDescription());
                        System.out.println("Editing Project Cover URL:" + editingProject.ge
tCoverURL());
                        System.out.println("Editing Project Create Time:" + editingProject.
getCreationTime());
                        System.out.println("Editing Project Last Modified Time:" + editingP
roject.getModifiedTime());
                        System.out.println("Editing Project Status:" + editingProject.getSt
atus());
                    }
                }

                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
```

```
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SetEditingProjectMaterialsRequest;
import com.aliyuncs.vod.model.v20170321.SetEditingProjectMaterialsResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<您的AccessKeySecret>
");

        SetEditingProjectMaterialsRequest request = new SetEditingProjectMaterialsRequest()
;
        // Editing Project ID
        request.setProjectId("projectid");
        // Set Editing Project Material IDs, use comma to split
        request.setMaterialIds("materialId1,materialId2");

        SetEditingProjectMaterialsResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetEditingProjectMaterialsRequest;
import com.aliyuncs.vod.model.v20170321.GetEditingProjectMaterialsResponse;

    /**
     * Sample Code
     */
    public static void main(String[] args) throws Exception {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        GetEditingProjectMaterialsRequest request = new GetEditingProjectMaterialsRequest()
;
        request.setProjectId("projectid");
        request.setType("video");

        GetEditingProjectMaterialsResponse response = null;
        try {
            response = client.getAcsResponse(request);
            if (response != null){
                if (response.getMaterialList() != null && response.getMaterialList().size()
> 0){
                    for (int i=0; i<response.getMaterialList().size(); i++){
                        System.out.println("No " + i + " Material");

                        GetEditingProjectMaterialsResponse.Material material = response.get
MaterialList().get(i);
                        System.out.println("Editing Project Material ID:" + material.getMat
erialId());
                        System.out.println("Editing Project Material Title:" + material.get
Title());
                        System.out.println("Editing Project Material Category ID:" + materi
al.getCateId());
                        System.out.println("Editing Project Material Category Name:" + mate
rial.getCateName());
                        System.out.println("Editing Project Material Tags:" + material.getT
ags());
                        System.out.println("Editing Project Material Cover URL:" + material
.getCoverURL());
                        System.out.println("Editing Project Material Create Time:" + materi
al.getCreationTime());
                        System.out.println("Editing Project Material Last Modified Time:" +
material.getModifiedTime());
                    }
                }

                // Request ID
                System.out.println("RequestId:" + response.getRequestId());
            }
        } catch (ServerException e){
            System.out.println("ServerError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
        } catch (ClientException e){
            System.out.println("ClientError code:" + e.getErrCode() + ", message:" + e.getE
rrMsg());
```

```
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
    }
```

2.12. 媒体审核

本文提供了Java SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SubmitAIMediaAuditJobRequest;
import com.aliyuncs.vod.model.v20170321.SubmitAIMediaAuditJobResponse;

/**
 * 提交智能审核作业
 */
public static SubmitAIMediaAuditJobResponse submitAIMediaAuditJob(DefaultAcsClient client)
throws Exception {
        SubmitAIMediaAuditJobRequest request = new SubmitAIMediaAuditJobRequest();
        // 设置视频ID
        request.setMediaId("dc063078c1d845139e2a5bd8ff****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            SubmitAIMediaAuditJobResponse response = submitAIMediaAuditJob(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印视频ID
            System.out.println("MediaId:" + response.getMediaId());
            // 打印作业ID
            System.out.println("JobId:" + response.getJobId());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetAIMediaAuditJobRequest;
import com.aliyuncs.vod.model.v20170321.GetAIMediaAuditJobResponse;

/**
 * 查询智能审核作业
 */
public static GetAIMediaAuditJobResponse getAIMediaAuditJob(DefaultAcsClient client) throws
Exception {
        GetAIMediaAuditJobRequest request = new GetAIMediaAuditJobRequest();
        // 设置作业ID
        request.setJobId("7dc69b893c8b4f13b47aae8de0df****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetAIMediaAuditJobResponse response = getAIMediaAuditJob(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());

            // 打印结果信息
            System.out.println("MediaId:" + response.getMediaAuditJob().getMediaId());
            System.out.println("JobId:" + response.getMediaAuditJob().getJobId());
            System.out.println("Type:" + response.getMediaAuditJob().getType());
            System.out.println("Status:" + response.getMediaAuditJob().getStatus());
            System.out.println("Code:" + response.getMediaAuditJob().getCode());
            System.out.println("Message:" + response.getMediaAuditJob().getMessage());
            System.out.println("Data AbnormalModules:" + response.getMediaAuditJob().getDat
a().getAbnormalModules());
            System.out.println("Data Label:" + response.getMediaAuditJob().getData().getLab
el());
            System.out.println("Data Suggestion:" + response.getMediaAuditJob().getData().g
etSuggestion());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultRequest;
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultResponse;

/**
 * 获取智能审核结果摘要
 */
public static GetMediaAuditResultResponse getMediaAuditResult(DefaultAcsClient client) thro
ws Exception {
        GetMediaAuditResultRequest request = new GetMediaAuditResultRequest();
        // 设置视频ID
        request.setMediaId("dc063078c1d845139e2a5bd8f58****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
// 调用示例
    public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetMediaAuditResultResponse response = getMediaAuditResult(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());

            // 打印作业结果信息
            System.out.println("Data AbnormalModules:" + response.getMediaAuditResult().get
AbnormalModules());
            System.out.println("Data Label:" + response.getMediaAuditResult().getLabel());
            System.out.println("Data Suggestion:" + response.getMediaAuditResult().getSugge
stion());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultDetailRequest;
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultDetailResponse;

/**
 * 获取智能审核结果详情
 */
public static GetMediaAuditResultDetailResponse getMediaAuditResultDetail(DefaultAcsClient
client) throws Exception {
        GetMediaAuditResultDetailRequest request = new GetMediaAuditResultDetailRequest();
        // 设置视频ID
        request.setMediaId("dc063078c1d845139e2a5bd8f69****");
        // 设置翻页页数
        request.setPageNo(1);

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetMediaAuditResultDetailResponse response = getMediaAuditResultDetail(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());

            // 打印结果信息
            System.out.println("Data Total:" + response.getMediaAuditResultDetail().getTota
l());
            System.out.println("Data List Size:" + response.getMediaAuditResultDetail().get
List().size());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultTimelineRequest;
import com.aliyuncs.vod.model.v20170321.GetMediaAuditResultTimelineResponse;

/**
 * 获取智能审核结果时间线
 */
public static GetMediaAuditResultTimelineResponse getMediaAuditResultTimeline(DefaultAcsCli
ent client) throws Exception {
        GetMediaAuditResultTimelineRequest request = new GetMediaAuditResultTimelineRequest
();
        // 设置视频ID
        request.setMediaId("dc063078c1d845139e2a5bd8f55****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetMediaAuditResultTimelineResponse response = getMediaAuditResultTimeline(clie
nt);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());

            // 打印结果信息
            System.out.println(response.getMediaAuditResultTimeline().getTerrorism());
            System.out.println(response.getMediaAuditResultTimeline().getPorn());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

人工审核

调用CreateAuditRequest接口，完成人工审核功能。

接口参数和返回字段请参见CreateAuditRequest。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.CreateAuditRequest;
import com.aliyuncs.vod.model.v20170321.CreateAuditResponse;
import com.alibaba.fastjson.JSONObject;

/**
 * 构建审核内容
 */
public static String buildAuditContent() throws Exception {
        List<JSONObject> auditContents = new ArrayList<>();

        JSONObject auditContent = new JSONObject();
        auditContent.put("VideoId", "3ebc10160bda481ca9b6858a0b58****"); // 视频ID
        auditContent.put("Status", "Blocked"); // 审核状态
        auditContent.put("Reason", "含有淫秽色情内容"); // 若审核状态为屏蔽时，需给出屏蔽的理由，
最长支持128字节
        auditContents.add(auditContent);

        return auditContents.toString();
}

/**
 * 人工审核
 */
public static CreateAuditResponse createAudit(DefaultAcsClient client) throws Exception {
        CreateAuditRequest request = new CreateAuditRequest();
        // 设置审核内容
        request.setAuditContent(buildAuditContent());

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 提交人工审核结果
        CreateAuditResponse response = createAudit(client);
        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetAuditHistoryRequest;
import com.aliyuncs.vod.model.v20170321.GetAuditHistoryResponse;

/**
 * 获取人工审核历史
 */
public static GetAuditHistoryResponse getAuditHistory(DefaultAcsClient client) throws Excep
tion {
        GetAuditHistoryRequest request = new GetAuditHistoryRequest();

        // 视频ID
        request.setVideoId("3ebc10160bda481ca9b6858a0b25****");
        // 页号
        request.setPageNo(1L);
        // 每页数量
        request.setPageSize(10L);

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 获取人工审核历史
        GetAuditHistoryResponse response = getAuditHistory(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
        // 打印审核历史记录总条数
        System.out.println("Total:" + response.getTotal());
        // 打印本次审核结果
        System.out.println("Status:" + response.getStatus());
        // 打印审核历史记录列表
        System.out.println("Histories:" + response.getHistories().toString());
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SetAuditSecurityIpRequest;
import com.aliyuncs.vod.model.v20170321.SetAuditSecurityIpResponse;

/**
 * 设置审核安全IP
 */
public static SetAuditSecurityIpResponse setAuditSecurityIp(DefaultAcsClient client) throws
Exception {
        SetAuditSecurityIpRequest request = new SetAuditSecurityIpRequest();

        // 审核安全组名称，默认为Default
        request.setSecurityGroupName("MyGroupName");
        // 操作方式，默认值Append(追加IP白名单)
        request.setOperateMode("Cover");
        // 审核安全IP列表
        request.setIps("10.23.12.20,10.23.12.21,10.23.12.22");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 设置审核安全IP
        SetAuditSecurityIpResponse response = setAuditSecurityIp(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListAuditSecurityIpRequest;
import com.aliyuncs.vod.model.v20170321.ListAuditSecurityIpResponse;

/**
 * 获取审核安全IP列表
 */
public static ListAuditSecurityIpResponse listAuditSecurityIp(DefaultAcsClient client) thro
ws Exception {
        ListAuditSecurityIpRequest request = new ListAuditSecurityIpRequest();

        // 审核安全组名称
        request.setSecurityGroupName("MyGroupName");

        // 返回结果
        return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 获取审核安全IP列表
        ListAuditSecurityIpResponse response = listAuditSecurityIp(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
        // 打印审核安全IP详情
        System.out.println("SecurityIpList:" + response.getSecurityIpList().toString());
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

2.13. 视频AI

本篇文档提供了Java SDK视频AI模块相关功能的API调用示例。包含提交AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询AI模板列表、查询设置AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SubmitAIJobRequest;
import com.aliyuncs.vod.model.v20170321.SubmitAIJobResponse;

/**
 * 提交作业
 */
public static SubmitAIJobResponse submitAIJob(DefaultAcsClient client) throws Exception {
    SubmitAIJobRequest request = new SubmitAIJobRequest();
    // 设置视频ID
    request.setMediaId("3eb19a4585bc475e995bdd4a5c****");
    // 设置AI类型，请确保已开通该类型AI
    request.setTypes("AIVideoCover");
    // 返回结果
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws ClientException {
   DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");

    try {
        // 提交作业
        SubmitAIJobResponse response = submitAIJob(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
        // 打印结果信息
        if (response.getAIJobList() != null && response.getAIJobList().size() != 0) {
            for (SubmitAIJobResponse.AIJob aiJob : response.getAIJobList()) {
              // 视频ID
              System.out.println("MediaId:" + aiJob.getMediaId());
              // 作业ID
              System.out.println("JobId:" + aiJob.getJobId());
              // AI类型
              System.out.println("Type:" + aiJob.getType());
            }
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob 。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListAIJobRequest;
```

```
import com.aliyuncs.vod.model.v20170321.ListAIJobResponse;

/**
 * 查询作业
 */
public static ListAIJobResponse listAIJob(DefaultAcsClient client) throws Exception {
    ListAIJobRequest request = new ListAIJobRequest();
    // 设置作业ID
    request.setJobIds("979d4d7a36ae41b1a945a2xxxxx,3eb19a4585bc475e995bddea****");
    // 返回结果
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 查询作业
        ListAIJobResponse response = listAIJob(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());

        // 打印作业列表
        System.out.println("============ AIJobList:");
        if (response.getAIJobList() != null && response.getAIJobList().size() != 0) {
            for (ListAIJobResponse.AIJob aiJob: response.getAIJobList()) {
                // 视频ID
                System.out.println("MediaId:" + aiJob.getMediaId());
                // 作业ID
                System.out.println("JobId:" + aiJob.getJobId());
                // AI类型
                System.out.println("Type:" + aiJob.getType());
                // 作业状态
                System.out.println("Status:" + aiJob.getStatus());
                // 作业结果
                System.out.println("Data:" + aiJob.getData());
            }
        }

        // 打印不存在的作业ID
        System.out.println("============ NonExistAIJobIds:");
        if (response.getNonExistAIJobIds() != null && response.getNonExistAIJobIds().size()
!= 0) {
            for (String jobId: response.getNonExistAIJobIds()) {
                // 作业ID
                System.out.println("NonExistAIJobId:" + jobId);
            }
```

```
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.vod.model.v20170321.AddAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.AddAITemplateResponse;

/**
 * 添加AI模板
 */
public static AddAITemplateResponse addAITemplate(DefaultAcsClient client) throws Exception
{
        AddAITemplateRequest request = new AddAITemplateRequest();
        // 设置模板类型，以智能审核模板为例
        request.setTemplateType("AIMediaAudit");
        // 设置模板名称
        request.setTemplateName("myaitemplate");
        // 设置模板详细配置
        JSONObject templateConfig = new JSONObject();
        JSONArray auditItem = new JSONArray();
        auditItem.add("terrorism");
        auditItem.add("porn");
        templateConfig.put("AuditItem", auditItem);

        JSONArray auditRange = new JSONArray();
        auditRange.add("video");
        auditRange.add("image-cover");
        auditRange.add("text-title");
        templateConfig.put("AuditRange", auditRange);

        JSONArray auditContent = new JSONArray();
        auditContent.add("screen");
        templateConfig.put("AuditContent", auditContent);

        templateConfig.put("AuditAutoBlock", "no");
        request.setTemplateConfig(templateConfig.toString());

        // 返回结果
        return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 * @param args
```

```
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            AddAITemplateResponse response = addAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板ID
            System.out.println("TemplateId:" + response.getTemplateId());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DeleteAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.DeleteAITemplateResponse;

/**
 * 删除AI模板
 */
public static DeleteAITemplateResponse deleteAITemplate(DefaultAcsClient client) throws Exc
eption {
        DeleteAITemplateRequest request = new DeleteAITemplateRequest();
        // 设置模板ID
        request.setTemplateId("1d763dd8987a122ab8596eb7d57****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            DeleteAITemplateResponse response = deleteAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板ID
            System.out.println("TemplateId:" + response.getTemplateId());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.vod.model.v20170321.UpdateAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.UpdateAITemplateResponse;

/**
 * 修改AI模板
 */
public static UpdateAITemplateResponse updateAITemplate(DefaultAcsClient client) throws Exc
```

```
eption {
        UpdateAITemplateRequest request = new UpdateAITemplateRequest();
        // 设置模板ID
        request.setTemplateId("1d763dd8987a122ab8596eb7d257****");
        // 设置模板名称
        request.setTemplateName("myaitemplate1");
        // 设置模板详细配置
        JSONObject templateConfig = new JSONObject();
        JSONArray auditItem = new JSONArray();
        //auditItem.add("terrorism");
        auditItem.add("porn");
        templateConfig.put("AuditItem", auditItem);

        JSONArray auditRange = new JSONArray();
        auditRange.add("video");
        auditRange.add("image-cover");
        //auditRange.add("text-title");
        templateConfig.put("AuditRange", auditRange);

        JSONArray auditContent = new JSONArray();
        auditContent.add("screen");
        templateConfig.put("AuditContent", auditContent);

        templateConfig.put("AuditAutoBlock", "no");
        request.setTemplateConfig(templateConfig.toString());

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            UpdateAITemplateResponse response = updateAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板ID
            System.out.println("TemplateId:" + response.getTemplateId());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.GetAITemplateResponse;

/**
 * 查询AI模板
 */
public static GetAITemplateResponse getAITemplate(DefaultAcsClient client) throws Exception
{
        GetAITemplateRequest request = new GetAITemplateRequest();
        // 设置模板ID
        request.setTemplateId("1d763dd8987a122ab8596eb7d257****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetAITemplateResponse response = getAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板信息
            System.out.println("TemplateId:" + response.getTemplateInfo().getTemplateId());
            System.out.println("TemplateType:" + response.getTemplateInfo().getTemplateType
());
            System.out.println("TemplateName:" + response.getTemplateInfo().getTemplateName
());
            System.out.println("TemplateConfig:" + response.getTemplateInfo().getTemplateCo
nfig());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.ListAITemplateResponse;

/**
 * 查询AI模板列表
 */
public static ListAITemplateResponse listAITemplate(DefaultAcsClient client) throws Excepti
on {
        ListAITemplateRequest request = new ListAITemplateRequest();
        // 设置模板类型，以智能审核模板为例
        request.setTemplateType("AIMediaAudit");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 查询结果
            ListAITemplateResponse response = listAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板信息
            if (response.getTemplateInfoList() != null) {
                for (ListAITemplateResponse.TemplateInfoListItem templateInfo : response.ge
tTemplateInfoList()) {
                    System.out.println("================");
                    System.out.println("TemplateId:" + templateInfo.getTemplateId());
                    System.out.println("TemplateType:" + templateInfo.getTemplateType());
                    System.out.println("TemplateName:" + templateInfo.getTemplateName());
                    System.out.println("TemplateConfig:" + templateInfo.getTemplateConfig()
);
                }
            }
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SetDefaultAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.SetDefaultAITemplateResponse;

/**
 * 设置默认AI模板
 */
public static SetDefaultAITemplateResponse setDefaultAITemplate(DefaultAcsClient client) th
rows Exception {
        SetDefaultAITemplateRequest request = new SetDefaultAITemplateRequest();
        // 设置模板ID
        request.setTemplateId("1d763dd8987a122ab8596eb757****");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            SetDefaultAITemplateResponse response = setDefaultAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板ID
            System.out.println("TemplateId:" + response.getTemplateId());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetDefaultAITemplateRequest;
import com.aliyuncs.vod.model.v20170321.GetDefaultAITemplateResponse;

/**
 * 查询默认AI模板
 */
public static GetDefaultAITemplateResponse getDefaultAITemplate(DefaultAcsClient client) th
rows Exception {
        GetDefaultAITemplateRequest request = new GetDefaultAITemplateRequest();
        // 设置模板类型，以智能审核模板为例
        request.setTemplateType("AIMediaAudit");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取执行结果
            GetDefaultAITemplateResponse response = getDefaultAITemplate(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印模板信息
            System.out.println("TemplateId:" + response.getTemplateInfo().getTemplateId());
            System.out.println("TemplateType:" + response.getTemplateInfo().getTemplateType
());
            System.out.println("TemplateName:" + response.getTemplateInfo().getTemplateName
());
            System.out.println("IsDefault:" + response.getTemplateInfo().getIsDefault());
            System.out.println("TemplateConfig:" + response.getTemplateInfo().getTemplateCo
nfig());
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

2.14. 视频DNA

本篇文档提供了Java SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.SubmitAIJobRequest;
import com.aliyuncs.vod.model.v20170321.SubmitAIJobResponse;

/**
 * 提交作业
 */
public static SubmitAIJobResponse submitAIJob(DefaultAcsClient client) throws Exception {
    SubmitAIJobRequest request = new SubmitAIJobRequest();
    // 设置视频ID
    request.setMediaId("3eb19a4585bc475e995bdd4d9a****");
    // 设置AI类型，类型为AIMediaDNA
    request.setTypes("AIMediaDNA");
    // 返回结果
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 提交作业
        SubmitAIJobResponse response = submitAIJob(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
        // 打印结果信息
        if (response.getAIJobList() != null && response.getAIJobList().size() != 0) {
            for (SubmitAIJobResponse.AIJob aiJob : response.getAIJobList()) {
              // 视频ID
              System.out.println("MediaId:" + aiJob.getMediaId());
              // 作业ID
              System.out.println("JobId:" + aiJob.getJobId());
              // AI类型
              System.out.println("Type:" + aiJob.getType());
            }
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListAIJobRequest;
import com.aliyuncs.vod.model.v20170321.ListAIJobResponse;

/**
 * 查询作业
 */
public static ListAIJobResponse listAIJob(DefaultAcsClient client) throws Exception {
    ListAIJobRequest request = new ListAIJobRequest();
    // 设置作业ID
    request.setJobIds("979d4d7a36ae41b1a945a257****,3eb19a4585bc475e995bddea57****");
    // 返回结果
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 查询作业
        ListAIJobResponse response = listAIJob(client);

        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());

        // 打印作业列表
        System.out.println("============ AIJobList:");
        if (response.getAIJobList() != null && response.getAIJobList().size() != 0) {
            for (ListAIJobResponse.AIJob aiJob: response.getAIJobList()) {
                // 视频ID
                System.out.println("MediaId:" + aiJob.getMediaId());
                // 作业ID
                System.out.println("JobId:" + aiJob.getJobId());
                // AI类型，注意校验类型是否为AIMediaDNA
                System.out.println("Type:" + aiJob.getType());
                // 作业状态
                System.out.println("Status:" + aiJob.getStatus());
                // 作业结果
                System.out.println("Data:" + aiJob.getData());
            }
        }

        // 打印不存在的作业ID
        System.out.println("============ NonExistAIJobIds:");
```

```
        if (response.getNonExistAIJobIds() != null && response.getNonExistAIJobIds().size()
!= 0) {
            for (String jobId: response.getNonExistAIJobIds()) {
                // 作业ID
                System.out.println("NonExistAIJobId:" + jobId);
            }
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetMediaDNAResultRequest;
import com.aliyuncs.vod.model.v20170321.GetMediaDNAResultResponse;

/**
 * 查询结果
 */
public static GetMediaDNAResultResponse getMediaDNAResult(DefaultAcsClient client) throws E
xception {
    GetMediaDNAResultRequest request = new GetMediaDNAResultRequest();
    // 设置视频ID
    request.setMediaId("3eb19a4585bc475e995bdd57****");
    // 返回结果
    return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 * @throws ClientException
 */
public static void main(String[] args) throws Exception {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;

    try {
        // 查询结果
        GetMediaDNAResultResponse response = getMediaDNAResult(client);
        // 打印请求ID
        System.out.println("ResquestId:" + response.getRequestId());
        // 打印DNA结果
        System.out.println("DNAResult:" + response.getDNAResult().toString());
    } catch (Exception e) {
        System.out.println("ErrorMessage:" + e.getLocalizedMessage());
    }
}
```

2.15. 直播转点播

本篇文档提供了Java SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListLiveRecordVideoRequest;
import com.aliyuncs.vod.model.v20170321.ListLiveRecordVideoResponse;

/*查询直播转点播视频列表函数*/
public static ListLiveRecordVideoResponse listLiveRecordVideo(DefaultAcsClient client) thro
ws Exception {
    ListLiveRecordVideoRequest request = new ListLiveRecordVideoRequest();
    request.setStartTime("2018-04-24T03:21:04Z");
    request.setEndTime("2018-05-21T03:21:44Z");
    request.setStreamName("testStream****");
    request.setDomainName("example.aliyundoc.com");
    request.setAppName("testApp****");
    return client.getAcsResponse(request);
}

/*以下为调用示例*/
public static void main(String[] argv) {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListLiveRecordVideoResponse response = new ListLiveRecordVideoResponse();
    try {
        response = listLiveRecordVideo(client);
        //打印查询出来的列表个数
        System.out.print("LiveRecordVideoList.size = " + response.getLiveRecordVideoList().
size() + "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

2.16. 转码任务

本篇文档提供了Java SDK转码任务模块相关功能的API调用示例。包含查询视频转码摘要、查询转码任务列表、查询转码任务详情。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询视频转码摘要

调用GetTranscodeSummary接口，完成查询视频转码摘要功能。

接口参数和返回字段请参见GetTranscodeSummary。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetTranscodeSummaryRequest;
import com.aliyuncs.vod.model.v20170321.GetTranscodeSummaryResponse;

/**
 * 查询视频转码摘要信息
 */
public static GetTranscodeSummaryResponse getTranscodeSummary(DefaultAcsClient client) thro
ws Exception {
    GetTranscodeSummaryRequest request = new GetTranscodeSummaryRequest();
    request.setVideoIds("14f35e2c5ca348831c2d2ae1d5b****,1dd4516420d247c777538c9aaa****");
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetTranscodeSummaryResponse response = new GetTranscodeSummaryResponse();
    try {
        response = getTranscodeSummary(client);
        for (GetTranscodeSummaryResponse.TranscodeSummary summary : response.getTranscodeSu
mmaryList()) {
            //转码状态
            System.out.println("TranscodeStatus = " + summary.getTranscodeStatus());
            for (GetTranscodeSummaryResponse.TranscodeSummary.TranscodeJobInfoSummary jobIn
foSummary : summary.getTranscodeJobInfoSummaryList()) {
                //转码进度
                System.out.println("TranscodeProgress = " + jobInfoSummary.getTranscodeProg
ress());
                //转码作业状态
                System.out.println("TranscodeJobStatus = " + jobInfoSummary.getTranscodeJob
Status());
                //转码作业完成时间
                System.out.println("CompleteTime = " + jobInfoSummary.getCompleteTime());
            }
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询转码任务列表

调用ListTranscodeTask接口，完成查询转码任务列表功能。

接口参数和返回字段请参见ListTranscodeTask。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.ListTranscodeTaskRequest;
import com.aliyuncs.vod.model.v20170321.ListTranscodeTaskResponse;

/**
 * 查询转码任务列表
 */
public static ListTranscodeTaskResponse listTranscodeTask(DefaultAcsClient client) throws E
xception {
    ListTranscodeTaskRequest request = new ListTranscodeTaskRequest();
    request.setVideoId("14f35e2c5ca345cb88de1d5b2801");
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    ListTranscodeTaskResponse response = new ListTranscodeTaskResponse();
    try {
        response = listTranscodeTask(client);
        for (ListTranscodeTaskResponse.TranscodeTask transcodeTask : response.getTranscodeT
askList()) {
            //转码任务ID
            System.out.println("TranscodeTaskId = " + transcodeTask.getTranscodeTaskId());
            //转码任务状态
            System.out.println("TaskStatus = " + transcodeTask.getTaskStatus());
        }
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

查询转码任务详情

调用GetTranscodeTask接口，完成查询转码任务详情功能。

接口参数和返回字段请参见GetTranscodeTask。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.GetTranscodeTaskRequest;
import com.aliyuncs.vod.model.v20170321.GetTranscodeTaskResponse;

/**
 * 查询转码任务详情
 */
public static GetTranscodeTaskResponse getTranscodeTask(DefaultAcsClient client) throws Exc
eption {
    GetTranscodeTaskRequest request = new GetTranscodeTaskRequest();
    request.setTranscodeTaskId("eab6f992b5f9436e6f425****");
    return client.getAcsResponse(request);
}


/**
 * 以下为调用示例
 */
public static void main(String[] args) throws ClientException {
    DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>")
;
    GetTranscodeTaskResponse response = new GetTranscodeTaskResponse();
    try {
        response = getTranscodeTask(client);
        //转码任务ID
        System.out.println("TranscodeTaskId = " + response.getTranscodeTask().getTranscodeT
askId());
        //转码任务状态
        System.out.println("TaskStatus = " + response.getTranscodeTask().getTaskStatus());
        for (GetTranscodeTaskResponse.TranscodeTask.TranscodeJobInfo jobInfo : response.get
TranscodeTask().getTranscodeJobInfoList()) {
            //转码作业ID
            System.out.println("TranscodeJobId = " + jobInfo.getTranscodeJobId());
            //转码作业状态
            System.out.println("TranscodeJobStatus = " + jobInfo.getTranscodeJobStatus());
            //转码处理进度
            System.out.println("TranscodeProgress = " + jobInfo.getTranscodeProgress());
            //转码作业完成时间
            System.out.println("CompleteTime = " + jobInfo.getCompleteTime());

        }
    } catch (Exception e) {
        System.out.println("ErrorMessage = " + e.getLocalizedMessage());
    }
    System.out.println("RequestId = " + response.getRequestId());
}
```

2.17. 点播CDN

本篇文档提供了Java SDK点播CDN模块相关功能的API调用示例。包含预热缓存、刷新缓存、查询刷新和预热状态、查询刷新预热次数限制和余量、查询流量数据、查询网络带宽、下载域名日志。

初始化客户端

使用前请先初始化客户端，请参见初始化。

预热缓存

调用PreloadVodObjectCaches接口，完成预热缓存功能。

接口参数和返回字段请参见PreloadVodObjectCaches。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.PreloadVodObjectCachesRequest;
import com.aliyuncs.vod.model.v20170321.PreloadVodObjectCachesResponse;

    /**
     * 预热缓存
     * @param client 发送请求客户端
     * @return PreloadVodObjectCachesResponse
     * @throws Exception
     */
    public static PreloadVodObjectCachesResponse preloadVodObjectCaches(DefaultAcsClient cl
ient) throws Exception {
        PreloadVodObjectCachesRequest request = new PreloadVodObjectCachesRequest();
        // 需要预热的文件路径
        request.setObjectPath("http://example.com/fd-****.mp4");
        return client.getAcsResponse(request);
    }

    /**
     * 请求示例
     * @param argv
     */
    public static void main(String[] argv) {
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            PreloadVodObjectCachesResponse response = preloadVodObjectCaches(client);
            // 预热返回的任务ID
            System.out.println("PreloadTaskId = " + response.getPreloadTaskId());
            // 请求ID
            System.out.println("RequestId = " + response.getRequestId());
        } catch (Exception e) {
            System.out.println("ErrorMessage = " + e.getMessage());
        }
    }
```

刷新缓存

调用RefreshVodObjectCaches接口，完成刷新缓存功能。

接口参数和返回字段请参见RefreshVodObjectCaches。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.RefreshVodObjectCachesRequest;
import com.aliyuncs.vod.model.v20170321.RefreshVodObjectCachesResponse;

    /**
     * 刷新缓存
     * @param client 发送请求客户端
     * @return RefreshVodObjectCachesResponse
     * @throws Exception
     */
    public static RefreshVodObjectCachesResponse refreshVodObjectCaches(DefaultAcsClient cl
ient) throws Exception {
        RefreshVodObjectCachesRequest request = new RefreshVodObjectCachesRequest();
        // 需要刷新的文件或者目录路径
        request.setObjectPath("http://example.com/fd-****.mp4");
        // 刷新的类型
        request.setObjectType("File");
        return client.getAcsResponse(request);
    }

    /**
     * 请求示例
     * @param argv
     */
    public static void main(String[] argv) {
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            RefreshVodObjectCachesResponse response = refreshVodObjectCaches(client);
            // 刷新返回的任务ID
            System.out.println("PreloadTaskId = " + response.getRefreshTaskId());
            // 请求ID
            System.out.println("RequestId = " + response.getRequestId());
        } catch (Exception e) {
            System.out.println("ErrorMessage = " + e.getMessage());
        }
    }
```

查询刷新和预热状态

调用DescribeVodRefreshTasks接口，完成查询刷新和预热状态功能。

接口参数和返回字段请参见DescribeVodRefreshTasks。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DescribeVodRefreshTasksRequest;
import com.aliyuncs.vod.model.v20170321.DescribeVodRefreshTasksResponse;

    /**
     * 查询刷新和预热状态
     * @param client 发送请求客户端
     * @return DescribeVodRefreshTasksResponse
     * @throws Exception
     */
    public static DescribeVodRefreshTasksResponse describeVodRefreshTasks(DefaultAcsClient
client) throws Exception {
```

```
        DescribeVodRefreshTasksRequest request = new DescribeVodRefreshTasksRequest();
        // 需要查询的域名
        request.setDomainName("example.com");
        // 任务类型
        request.setObjectType("file");
        return client.getAcsResponse(request);
    }

    /**
     * 请求示例
     * @param argv
     */
    public static void main(String[] argv) {
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            DescribeVodRefreshTasksResponse response = describeVodRefreshTasks(client);
            // 整页大小
            System.out.println("PageSize = " + response.getPageSize());
            // 页码
            System.out.println("PageNumber = " + response.getPageNumber());
            // 总条数
            System.out.println("TotalCount = " + response.getTotalCount());
            // 请求ID
            System.out.println("RequestId = " + response.getRequestId());
            if (response.getTasks() != null && response.getTasks().size() > 0) {
                for (DescribeVodRefreshTasksResponse.Task task : response.getTasks()) {
                    System.out.println("----------------------------");
                    // 任务ID
                    System.out.println("TaskId = " + task.getTaskId());
                    // 刷新对象路径
                    System.out.println("ObjectPath = " + task.getObjectPath());
                    // 状态
                    System.out.println("Status = " + task.getStatus());
                    // 进度百分比
                    System.out.println("Process = " + task.getProcess());
                    // 任务类型
                    System.out.println("ObjectType = " + task.getObjectType());
                    // 任务对象创建时间，使用UTC时间
                    System.out.println("CreationTime = " + task.getCreationTime());
                    // 刷新预热失败返回的错误描述
                    System.out.println("Description = " + task.getDescription());
                }
            }
        } catch (Exception e) {
            System.out.println("ErrorMessage = " + e.getMessage());
        }
    }
```

查询刷新预热次数限制和余量

调用DescribeVodRefreshQuota接口，完成查询刷新预热次数限制和余量功能。

接口参数和返回字段请参见DescribeVodRefreshQuota。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DescribeVodRefreshQuotaRequest;
import com.aliyuncs.vod.model.v20170321.DescribeVodRefreshQuotaResponse;

    /**
     * 查询刷新预热次数限制和余量
     * @param client 发送请求客户端
     * @return DescribeVodRefreshQuotaResponse
     * @throws Exception
     */
    public static DescribeVodRefreshQuotaResponse describeVodRefreshQuota(DefaultAcsClient
client) throws Exception {
        DescribeVodRefreshQuotaRequest request = new DescribeVodRefreshQuotaRequest();
        return client.getAcsResponse(request);
    }

    /**
     * 请求示例
     * @param argv
     */
    public static void main(String[] argv) {
        try {
            DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
            DescribeVodRefreshQuotaResponse response = describeVodRefreshQuota(client);
            // 当日URL刷新数量上限
            System.out.println("UrlQuota = " + response.getUrlQuota());
            // 当日目录刷新数量上限
            System.out.println("DirQuota = " + response.getDirQuota());
            // 当日剩余URL刷新数量
            System.out.println("UrlRemain = " + response.getUrlRemain());
            // 当日剩余目录刷新数量
            System.out.println("DirRemain = " + response.getDirRemain());
            // 当天预热数量上限
            System.out.println("PreloadQuota = " + response.getPreloadQuota());
            // 当天剩余预热数量
            System.out.println("PreloadRemain = " + response.getPreloadRemain());
            // 请求ID
            System.out.println("RequestId = " + response.getRequestId());
        } catch (Exception e) {
            System.out.println("ErrorMessage = " + e.getMessage());
        }
    }
```

查询流量数据

调用DescribeVodDomainTrafficData接口，完成查询流量数据功能。

接口参数和返回字段请参见DescribeVodDomainTrafficData。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainTrafficDataRequest;
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainTrafficDataResponse;

/**
 * 查询流量数据
```

```
 */
public static DescribeVodDomainTrafficDataResponse describeVodDomainTrafficData(DefaultAcsC
lient client) throws Exception {
        DescribeVodDomainTrafficDataRequest request = new DescribeVodDomainTrafficDataReque
st();
        // 设置域名
        request.setDomainName("example.aliyundoc.com");
        // 设置开始时间，请使用UTC格式
        request.setStartTime("2019-01-15T15:59:59Z");
        // 设置结束时间，请使用UTC格式
        request.setEndTime("2019-01-20T15:59:58Z");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");

        try {
            // 获取结果
            DescribeVodDomainTrafficDataResponse response = describeVodDomainTrafficData(cl
ient);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印域名
            System.out.println("DomainName:" + response.getDomainName());
            // 打印开始时间
            System.out.println("StartTime:" + response.getStartTime());
            // 打印结束时间
            System.out.println("EndTime:" + response.getEndTime());
            // 打印时间间隔
            System.out.println("DataInterval:" + response.getDataInterval());
            // 打印流量数据
            if (response.getTrafficDataPerInterval() != null && response.getTrafficDataPerI
nterval().size() != 0) {
                List<DescribeVodDomainTrafficDataResponse.DataModule> dataModules = respons
e.getTrafficDataPerInterval();
                for(int i=0; i<dataModules.size(); i++) {
                    System.out.println("========================" );
                    System.out.println("TimeStamp:" + dataModules.get(i).getTimeStamp());
                    System.out.println("Value:" + dataModules.get(i).getValue());
                    System.out.println("DomesticValue:" + dataModules.get(i).getDomesticVal
ue());
                    System.out.println("OverseasValue:" + dataModules.get(i).getOverseasVal
ue());
                }
```

```
            }
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

查询网络带宽

调用DescribeVodDomainBpsData接口，完成查询网络带宽功能。

接口参数和返回字段请参见DescribeVodDomainBpsData。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainBpsDataRequest;
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainBpsDataResponse;
import java.util.List;

/**
 * 查询网络带宽
 */
public static DescribeVodDomainBpsDataResponse describeVodDomainBpsData(DefaultAcsClient cl
ient) throws Exception {
        DescribeVodDomainBpsDataRequest request = new DescribeVodDomainBpsDataRequest();
        // 设置域名
        request.setDomainName("example.aliyundoc.com");
        // 设置开始时间，请使用UTC格式
        request.setStartTime("2019-01-15T15:59:59Z");
        // 设置结束时间，请使用UTC格式
        request.setEndTime("2019-01-20T15:59:58Z");

        // 返回结果
        return client.getAcsResponse(request);
}

/**
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");
        try {
            // 获取结果
            DescribeVodDomainBpsDataResponse response = describeVodDomainBpsData(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());
            // 打印域名
            System.out.println("DomainName:" + response.getDomainName());
            // 打印开始时间
            System.out.println("StartTime:" + response.getStartTime());
            // 打印结束时间
            System.out.println("EndTime:" + response.getEndTime());
            // 打印时间间隔
```

```
            // 打印时间间隔
            System.out.println("DataInterval:" + response.getDataInterval());
            // 打印带宽数据
            if (response.getBpsDataPerInterval() != null && response.getBpsDataPerInterval(
).size() != 0) {
                List<DescribeVodDomainBpsDataResponse.DataModule> dataModules = response.ge
tBpsDataPerInterval();
                for(int i=0; i<dataModules.size(); i++) {
                    System.out.println("=========================" );
                    System.out.println("TimeStamp:" + dataModules.get(i).getTimeStamp());
                    System.out.println("Value:" + dataModules.get(i).getValue());
                    System.out.println("DomesticValue:" + dataModules.get(i).getDomesticVal
ue());
                    System.out.println("OverseasValue:" + dataModules.get(i).getOverseasVal
ue());
                }

            }
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

下载域名日志

调用DescribeVodDomainLog接口，完成下载域名日志功能。

接口参数和返回字段请参见DescribeVodDomainLog。调用示例如下：

```
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainLogRequest;
import com.aliyuncs.vod.model.v20170321.DescribeVodDomainLogResponse;
import java.util.List;

/**
 * 下载域名日志
 */
public static DescribeVodDomainLogResponse describeVodDomainLog(DefaultAcsClient client) th
rows Exception {
        DescribeVodDomainLogRequest request = new DescribeVodDomainLogRequest();
        // 设置域名
        request.setDomainName("example.alicdn.com");
        // 设置开始时间，请使用UTC格式
        request.setStartTime("2019-01-15T15:59:59Z");
        // 设置结束时间，请使用UTC格式
        request.setEndTime("2019-01-20T15:59:58Z");
        // 设置分页大小
        //request.setPageSize(300L);
        // 设置分页页号
        //request.setPageNumber(1L);

        // 返回结果
        return client.getAcsResponse(request);
}

/**
```

```
 * 以下为调用示例
 * @param args
 */
public static void main(String[] args) throws Exception {
        // 初始化客户端
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");
        try {
            // 获取结果
            DescribeVodDomainLogResponse response = describeVodDomainLog(client);

            // 打印请求ID
            System.out.println("ResquestId:" + response.getRequestId());

            // 打印CDN日志详细数据
            if (response.getDomainLogDetails() != null && response.getDomainLogDetails().si
ze() != 0) {
                List<DescribeVodDomainLogResponse.DomainLogDetail> domainLogDetails = respo
nse.getDomainLogDetails();
                for(int i=0; i<domainLogDetails.size(); i++) {
                    System.out.println("========================" );
                    System.out.println("DomainName:" + domainLogDetails.get(i).getDomainNam
e());
                    System.out.println("LogCount:" + domainLogDetails.get(i).getLogCount())
;

                    System.out.println("=== PageInfoDetail Data ===" );
                    System.out.println("PageNumber:" + domainLogDetails.get(i).getPageInfos
().getPageNumber());
                    System.out.println("PageSize:" + domainLogDetails.get(i).getPageInfos()
.getPageSize());
                    System.out.println("Total:" + domainLogDetails.get(i).getPageInfos().ge
tTotal());

                    System.out.println("=== LogInfoDetail Data ===" );
                    List<DescribeVodDomainLogResponse.DomainLogDetail.LogInfoDetail> logInf
os = domainLogDetails.get(i).getLogInfos();
                    if (logInfos != null && logInfos.size() != 0) {
                        for (int k=0; k<logInfos.size(); k++) {
                            System.out.println("LogName:" + logInfos.get(k).getLogName());
                            System.out.println("LogPath:" + logInfos.get(k).getLogPath());
                            System.out.println("LogSize:" + logInfos.get(k).getLogSize());
                            System.out.println("StartTime:" + logInfos.get(k).getStartTime(
));
                            System.out.println("EndTime:" + logInfos.get(k).getEndTime());
                        }
                    }
                }

            }
        } catch (Exception e) {
            System.out.println("ErrorMessage:" + e.getLocalizedMessage());
        }
}
```

3.Python SDK

3.1. 安装

本文为您介绍视频点播服务端Python SDK的安装、更新及卸载步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

- 安装Python 2.7及以上版本，请参见Python官网下载合适的版本安装。
- 安装pip，请参见pip官网进行安装。

安装SDK

操作命令，示例如下：

```
pip install aliyun-python-sdk-core
pip install aliyun-python-sdk-vod
```

> **说明**
>
> - 如果使用的是Python 3.x，请将 `pip install aliyun-python-sdk-core` 修改为 `pip install aliyun-python-sdk-core-v3` 。如果同时安装了不同版本，可使用pip3命令。
> - 如果安装时提示权限不足，请在命令前加 `sudo` 。

更新SDK

若发现新的接口或已有接口新的功能在当前SDK没有，可更新到最新版。示例如下：

```
pip install --upgrade aliyun-python-sdk-vod
```

卸载SDK

通过pip卸载。示例如下：

```
pip uninstall aliyun-python-sdk-core
pip uninstall aliyun-python-sdk-vod
```

3.2. 初始化

Python SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端Python SDK。安装方法，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入账号AccessKey信息，进行初始化。示例如下：

```
# -*- coding: UTF-8 -*-
import json
import traceback
from aliyunsdkcore.client import AcsClient
# 填入AccessKey信息
def init_vod_client(accessKeyId, accessKeySecret):
    regionId = 'cn-shanghai'    # 点播服务接入地域
    connectTimeout = 3          # 连接超时，单位为秒
    return AcsClient(accessKeyId, accessKeySecret, regionId, auto_retry=True, max_retry_tim
e=3, timeout=connectTimeout)
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例如下：

```
# -*- coding: UTF-8 -*-
import json
import traceback
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.auth.credentials import StsTokenCredential
# 填入STS信息
def init_vod_client(accessKeyId, accessKeySecret, securityToken):
    regionId = 'cn-shanghai'    # 点播服务接入地域
    connectTimeout = 3          # 连接超时，单位为秒
    credential = StsTokenCredential(accessKeyId, accessKeySecret, securityToken)
    return AcsClient(region_id=regionId, auto_retry=True, max_retry_time=3, timeout=connect
Timeout, credential=credential)
```

3.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍Python SDK调用服务端上传接口的使用场景和调用示例。

使用场景

> **注意**
>
> 本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端实例。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端实例。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import CreateUploadVideoRequest
def create_upload_video(clt):
    request = CreateUploadVideoRequest.CreateUploadVideoRequest()
    request.set_Title('Video Title')
    request.set_FileName('/opt/video/sample/video_file.mp4')
    request.set_Description('Video Description')
    //CoverURL示例：http://192.168.0.0/16/tps/TB1qnJ1PVXXXXXCXXXXXXXXXXXX-700-700.png
    request.set_CoverURL('<your Cover URL>')
    request.set_Tags('tag1,tag2')
    request.set_CateId(0)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    uploadInfo = create_upload_video(clt)
    print(uploadInfo['UploadAuth'])
    print(json.dumps(uploadInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import RefreshUploadVideoRequest
def refresh_upload_video(clt, videoId):
    request = RefreshUploadVideoRequest.RefreshUploadVideoRequest()
    request.set_VideoId(videoId)
    request.set_accept_format('JSON')
    return json.loads(clt.do_action_with_exception(request))

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    uploadInfo = refresh_upload_video(clt, 'VideoId')
    print(json.dumps(uploadInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import CreateUploadImageRequest
def create_upload_image(clt, imageType, imageExt):
    request = CreateUploadImageRequest.CreateUploadImageRequest()
    request.set_ImageType(imageType)
    request.set_ImageExt(imageExt)
    request.set_accept_format('JSON')
    return json.loads(clt.do_action_with_exception(request))

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    imageInfo = create_upload_image(clt, 'cover', 'jpg')
    print(json.dumps(imageInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import CreateUploadAttachedMediaRequest
def create_upload_attached_media(clt):
    request = CreateUploadAttachedMediaRequest.CreateUploadAttachedMediaRequest()
    request.set_BusinessType("watermark");
    request.set_MediaExt("gif");
    request.set_Title("this is a sample");
    request.set_accept_format('JSON')
    return json.loads(clt.do_action_with_exception(request))

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    uploadInfo = create_upload_attached_media(clt)
    print(uploadInfo['UploadAuth'])
    print(json.dumps(uploadInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UploadMediaByURLRequest
import sys,urllib
def upload_media_by_url(clt):
    request = UploadMediaByURLRequest.UploadMediaByURLRequest()
    //URL示例：http://example-bucket-****.cn-shanghai.aliyuncs.com/sample1-****.mp4，http://
example-bucket-****.cn-shanghai.aliyuncs.com/sample2-****.flv
    urls = ["<your URL1>",
            "your URL2"]
    # Encode the urls and add metadata
    uploadUrls = []
    uploadMetadatas = []
    for url in urls:
        if sys.version_info[0] == 3:
            encodeUrl = urllib.parse.quote(url)
        else:
            encodeUrl = urllib.quote(url)
        uploadUrls.append(encodeUrl)
        uploadMetadatas.append({'SourceURL': encodeUrl, 'Title': 'Sample Title'})

    request.set_UploadURLs(','.join(uploadUrls))
    request.set_UploadMetadatas(json.dumps(uploadMetadatas))

    request.set_accept_format('JSON')
    return json.loads(clt.do_action_with_exception(request))

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    uploadInfo = upload_media_by_url(clt)
    print(uploadInfo['UploadJobs'])
    print(json.dumps(uploadInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取URL上传信息

调用GetURLUploadInfos接口，完成获取URL上传信息功能。

接口参数和返回字段请参见GetURLUploadInfos。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetURLUploadInfosRequest
import sys,urllib
def get_url_upload_infos(clt):
    request = GetURLUploadInfosRequest.GetURLUploadInfosRequest()
    //URL示例：http://192.168.0.0/16/sample1-****.mp4，http://192.168.0.0/16/sample2-****.fl
v
    urls = ["<your URL1>",
            "<your URL2>"]

    # 对URL进行编码
    uploadUrls = []
    for url in urls:
        if sys.version_info[0] == 3:
            encodeUrl = urllib.parse.quote(url)
        else:
            encodeUrl = urllib.quote(url)
        uploadUrls.append(encodeUrl)

    # 设置上传的URL列表，用逗号分隔
    request.set_UploadURLs(',.join(uploadUrls))

    # 也可以传入jobId查询
    #request.set_JobIds("jobId1,jobId2")

    request.set_accept_format('JSON')
    return json.loads(clt.do_action_with_exception(request))

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_url_upload_infos(clt)
    print(res['NonExists'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.4. 音视频播放

本篇文档提供了Python SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetPlayInfoRequest
def get_play_info(clt, videoId):
    request = GetPlayInfoRequest.GetPlayInfoRequest()
    request.set_accept_format('JSON')
    request.set_VideoId(videoId)
    request.set_AuthTimeout(3600*5)
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    playInfo = get_play_info(clt, '<videoId>')
    print(json.dumps(playInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetVideoPlayAuthRequest
def get_video_playauth(clt, videoId):
    request = GetVideoPlayAuthRequest.GetVideoPlayAuthRequest()
    request.set_accept_format('JSON')
    request.set_VideoId(videoId)
    request.set_AuthInfoTimeout(3000)
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    playAuth = get_video_playauth(clt, '<videoId>')
    print(json.dumps(playAuth, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.5. 媒资管理

本篇文档提供了Python SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SearchMediaRequest
def search_media(clt):
    request = SearchMediaRequest.SearchMediaRequest()
    request.set_Fields('Title,CoverURL,Status');
    request.set_Match('Status in ("Normal","Checking") and CreationTime = ("2018-10-01T08:0
0:00Z","2018-12-25T08:00:00Z")')
    request.set_PageNo(1)
    request.set_PageSize(10)
    request.set_SearchType('video')
    request.set_SortBy('CreationTime:Desc')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    medias = search_media(clt)
    print(medias['MediaList'])
    print(json.dumps(medias, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetVideoInfoRequest
def get_video_info(clt, videoId):
    request = GetVideoInfoRequest.GetVideoInfoRequest()
    request.set_accept_format('JSON')
    request.set_VideoId(videoId)
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    videoInfo = get_video_info(clt, '<videoId>')
    print(json.dumps(videoInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetVideoInfosRequest
def get_video_infos(clt):
    request = GetVideoInfosRequest.GetVideoInfosRequest()
    videoIds = ['videoId1', 'videoId2']
    request.set_VideoIds(','.join(videoIds))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    videos = get_video_infos(clt)
    #print(videos['VideoList'])
    print(videos['NonExistVideoIds'])
    print(json.dumps(videos, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateVideoInfoRequest
def update_video_info(clt, videoId):
    request = UpdateVideoInfoRequest.UpdateVideoInfoRequest()
    request.set_VideoId(videoId)
    request.set_Title('New Title')
    request.set_Description('New Description')
    request.set_CoverURL('http192.168.0.0/16/tps/TB1qnJ1PVXXXXXCXXXXXXXXXXXX-700-700.png')

    request.set_Tags('tag1,tag2')
    #request.set_CateId(0)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    updateInfo = update_video_info(clt, '<videoId>')
    print(json.dumps(updateInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateVideoInfosRequest
def update_video_infos(clt):
    request = UpdateVideoInfosRequest.UpdateVideoInfosRequest()

    updateContent = []
    updateItem1 = {'VideoId': '<VideoId1>', 'Title': 'New Sample Title1', 'Tags': 'NewTag1,
NewTag2'}
    updateItem2 = {'VideoId': '<VideoId2>', 'Title': 'New Sample Title2', 'Tags': 'NewTag3,
NewTag4'}
    updateContent.append(updateItem1)
    updateContent.append(updateItem2)
    request.set_UpdateContent(json.dumps(updateContent))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    update = update_video_infos(clt)
    print(update['NonExistVideoIds'])
    print(json.dumps(update, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteVideoRequest
def delete_videos(clt):
    request = DeleteVideoRequest.DeleteVideoRequest()
    videoIds = ['videoId1', 'videoId2']
    request.set_VideoIds(','.join(videoIds))  # 支持批量删除视频，多个用逗号分隔

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    delInfo = delete_videos(clt)
    print(json.dumps(delInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取源文件信息（含原片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetMezzanineInfoRequest
def get_mezzanine_info(clt, videoId):
    request = GetMezzanineInfoRequest.GetMezzanineInfoRequest()
    request.set_VideoId(videoId)
    request.set_AuthTimeout(3600*5)
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    mezzanine = get_mezzanine_info(clt, '<videoId>')
    print(json.dumps(mezzanine, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

注意

时间参数要使用UTC时间，北京时间与UTC的时差为+8，也就是UTC+8，例如2018-01-01T12:00:00Z表示北京时间2018年1月1日20点0分0秒。

```
from aliyunsdkvod.request.v20170321 import GetVideoListRequest
def get_video_list(clt):
    request = GetVideoListRequest.GetVideoListRequest()
    # 以获取最近30天的视频列表为例
    utcNow = datetime.datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%SZ")
    utcMonthAgo = datetime.datetime.utcfromtimestamp(time.time() - 30*86400).strftime("%Y-%
m-%dT%H:%M:%SZ")
    request.set_StartTime(utcMonthAgo)    # 视频创建的起始时间，为UTC格式
    request.set_EndTime(utcNow)           # 视频创建的结束时间，为UTC格式
    #request.set_Status('Uploading,Normal,Transcoding')  # 视频状态，默认获取所有状态的视频，多
个用逗号分隔
    #request.set_CateId(0)                # 按分类进行筛选
    request.set_PageNo(1)
    request.set_PageSize(20)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    videoList = get_video_list(clt)
    print(json.dumps(videoList, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteStreamRequest
# 可删除视频流或音频流信息及存储文件，并支持批量删除；删除后当CDN缓存过期，该路流会无法播放，请谨慎操作
def delete_stream(clt):
    request = DeleteStreamRequest.DeleteStreamRequest()
    request.set_VideoId('<videoId>')
    # 媒体流转码的作业ID列表，多个用逗号分隔；JobId可通过获取播放地址接口(GetPlayInfo)获取到。
    jobIds = ['jobId1', 'jobId2']
    request.set_JobIds(','.join(jobIds))
    request.set_accept_format('JSON')

    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    delStream = delete_stream(clt)
    print(json.dumps(delStream, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteMezzaninesRequest
def delete_mezzanines(clt):
    request = DeleteMezzaninesRequest.DeleteMezzaninesRequest()

    videoIds = ['videoId1', 'videoId2']
    request.set_VideoIds(','.join(videoIds))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    delInfo = delete_mezzanines(clt)
    print(delInfo['NonExistVideoIds'])
    print(json.dumps(delInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateImageInfosRequest
def update_image_infos(clt):
    request = UpdateImageInfosRequest.UpdateImageInfosRequest()

    updateContent = []
    updateItem1 = {'ImageId': '<imageId1>', 'Title': 'New Sample Title1', 'Tags': 'NewTag1,
NewTag2'}
    updateItem2 = {'ImageId': '<imageId2>', 'Title': 'New Sample Title2', 'Tags': 'NewTag3,
NewTag4'}
    updateContent.append(updateItem1)
    updateContent.append(updateItem2)
    request.set_UpdateContent(json.dumps(updateContent))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    update = update_image_infos(clt)
    print(update['NonExistImageIds']['ImageId'])
    print(json.dumps(update, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetImageInfoRequest
def get_image_info(clt):
    request = GetImageInfoRequest.GetImageInfoRequest()
    request.set_ImageId('<imageId>')
    request.set_AuthTimeout(3600*24)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    image = get_image_info(clt)
    print(image['ImageInfo'])
    print(json.dumps(image, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteImageRequest
def delete_image(clt):
    request = DeleteImageRequest.DeleteImageRequest()

    #根据ImageURL删除图片文件
    request.set_DeleteImageType('ImageURL')
    request.set_ImageURLs('http://192.168.0.0/16/cover.jpg')

    #根据ImageId删除图片文件
    #request.set_DeleteImageType('ImageId')
    #request.set_ImageIds('ImageId1,ImageId2')

    #根据VideoId删除指定ImageType的图片文件
    #request.set_DeleteImageType('VideoId')
    #request.set_VideoId('VideoId')
    #request.set_ImageType('SpriteSnapshot')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    delInfo = delete_image(clt)
    print(json.dumps(delInfo, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.6. 媒资分类

本文为您提供了Python SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import AddCategoryRequest
def add_category(clt, cateName, parentId=-1):
    request = AddCategoryRequest.AddCategoryRequest()
    request.set_CateName(cateName)
    request.set_ParentId(parentId)
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    addRes = add_category(clt, 'Category Name')
    print(json.dumps(addRes, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateCategoryRequest
def update_category(clt, cateId, cateName):
    request = UpdateCategoryRequest.UpdateCategoryRequest()
    request.set_CateId(cateId)
    request.set_CateName(cateName)
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    updateRes = update_category(clt, '<cateId>', 'New Category Name')
    print(json.dumps(updateRes, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateCategoryRequest
# 删除视频分类，同时会删除其下级分类（包括二级分类和三级分类），请慎重操作
def delete_category(clt, cateId):
    request = DeleteCategoryRequest.DeleteCategoryRequest()
    request.set_CateId(cateId)
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    delRes = delete_category(clt, '<cateId>')
    print(json.dumps(delRes, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetCategoriesRequest
def get_categories(clt, cateId=-1, pageNo=1, pageSize=10):
    request = GetCategoriesRequest.GetCategoriesRequest()
    request.set_CateId(cateId)
    request.set_PageNo(pageNo)
    request.set_PageSize(pageSize)
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    getRes = get_categories(clt, '<cateId>')
    print(json.dumps(getRes, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.7. 媒体处理

本文为您提供了Python SDK媒体处理模块相关功能的API调用示例，包含提交转码作业（HLS标准加密、普通）、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业-普通

调用SubmitTranscodeJobs接口，完成提交转码作业-普通功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为普通转码（不加密）。阿里云视频加密方案请参见阿里云视频加密。

```
"""
* 构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；如不使用可忽略
* 需要替换的水印信息对应水印ID必须是关联在指定的转码模板ID上(即TranscodeTemplateId)
* 不支持通过本接口去增加一个没有关联上的水印
"""
def build_override_params():
    # 以替换水印为例
    watermarks = []

    # 替换图片水印文件，FlieURL示例：https://192.168.0.0/16/watermarks/sample****.png
    watermark1 = {'WatermarkId': '<watermarkId>', 'FileUrl': '<your File URL>'}
    watermarks.append(watermark1)

    # 替换文字水印
    watermark2 = {'WatermarkId': '<watermarkId>', 'Content': 'new Text'}
    watermarks.append(watermark2)

    return {'Watermarks': watermarks}


"""  发起转码处理   """
from aliyunsdkvod.request.v20170321 import SubmitTranscodeJobsRequest
def normal_submit_transcode_jobs(clt):
    request = SubmitTranscodeJobsRequest.SubmitTranscodeJobsRequest()

    # 需要转码的视频ID
    request.set_VideoId('<videoId>')

    # 指定转码模板组
    request.set_TemplateGroupId('<templateGroupId>')

    """
    # 可选设置覆盖参数，如替换水印
    overrideParams = build_override_params()
    request.set_OverrideParams(json.dumps(overrideParams))
    """

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    jobs = normal_submit_transcode_jobs(clt)
    print(jobs['TranscodeJobs']['TranscodeJob'])
    print(json.dumps(jobs, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

提交转码作业-HLS标准加密

调用SubmitTranscodeJobs接口，完成提交转码作业-HLS标准加密功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为使用HLS标准加密提交转码作业的示例。

```
"""
* 生成HLS标准加密参数的可选配置，用不到的可忽略
* 此时依赖KMS服务，需要安装KMS依赖包 aliyun-python-sdk-kms，并引入相关类
* 生成密钥API参考：https://192.168.0.0/16/document_detail/28948.html
"""
from aliyunsdkkms.request.v20160120 import GenerateDataKeyRequest
from aliyunsdkcore.http import protocol_type
def build_encrypt_config(clt):
    try:
        # 随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥，
        # 视频标准加密只需要传递密文密钥即可
        request = GenerateDataKeyRequest.GenerateDataKeyRequest()
        request.set_KeyId('<serviceKey>')
        request.set_KeySpec('AES_128')

        # KMS接口必须使用HTTPS访问
        request.set_protocol_type(protocol_type.HTTPS)
        request.set_accept_format('JSON')
        response = json.loads(clt.do_action_with_exception(request))

        # 解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥都不一样；注意您需要自己部署解密服务
        # Ciphertext作为解密接口的参数名称，可自定义，此处只作为参考
        # decryptKeyUri示例：http://demo.example.com/decrypt?' + 'Ciphertext=' + response['CiphertextBlob
        decryptKeyUri = ['<your decryptKeyUri>']

        return {'DecryptKeyUri': decryptKeyUri, 'KeyServiceType': 'KMS', 'CipherText': response['CiphertextBlob']}

    except Exception as e:
        print(e)
        #print(traceback.format_exc())
        return None


"""  发起转码处理  """
from aliyunsdkvod.request.v20170321 import SubmitTranscodeJobsRequest
def hlsencrypt_submit_transcode_jobs(clt):
    request = SubmitTranscodeJobsRequest.SubmitTranscodeJobsRequest()

    # 需要转码的视频ID
    request.set_VideoId('<videoId>')

    # 指定转码模板组
    request.set_TemplateGroupId('<templateGroupId>')
```

```
    # 使用KMS生成随机的加密密钥
    encryptConfig = build_encrypt_config(clt)
    if encryptConfig is not None:
        request.set_EncryptConfig(json.dumps(encryptConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    jobs = hlsencrypt_submit_transcode_jobs(clt)
    print(jobs['TranscodeJobs']['TranscodeJob'])
    print(json.dumps(jobs, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明**
>
> 创建截图模板详细请参见截图模板。

```
from aliyunsdkvod.request.v20170321 import SubmitSnapshotJobRequest
def submit_snapshot_job(clt):
    request = SubmitSnapshotJobRequest.SubmitSnapshotJobRequest()

    # 需要截图的视频ID
    request.set_VideoId('<videoId>')

    # 截图模板ID
    #request.set_SnapshotTemplateId('<snapshotTemplateId>')

    # 如果设置了截图模板ID，则会忽略下面参数
    request.set_Count(50L)
    request.set_SpecifiedOffsetTime(0L)
    request.set_Interval(1L)
    request.set_Width(200)
    request.set_Height(200)

    # 雪碧图参数，如不使用可不设置
    spriteSnapshotConfig = {'CellWidth': 120, 'CellHeight': 68, 'Columns': 3,
                            'Lines': 10, 'Padding': 20, 'Margin': 50}
    # 保留雪碧图原始图
    spriteSnapshotConfig['KeepCellPic'] = 'keep'
    spriteSnapshotConfig['Color'] = 'tomato'
    request.set_SpriteSnapshotConfig(json.dumps(spriteSnapshotConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    job = submit_snapshot_job(clt)
    print(json.dumps(job, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListSnapshotsRequest
def list_snapshots(clt):
    request = ListSnapshotsRequest.ListSnapshotsRequest()

    request.set_VideoId('<videoId>')
    request.set_SnapshotType('CoverSnapshot')
    request.set_PageNo(1)
    request.set_PageSize(20)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    snapshots = list_snapshots(clt)
    print(snapshots['MediaSnapshot'])
    print(json.dumps(snapshots, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
from aliyunsdkvod.req uest.v20170321 import SubmitPreprocessJobsRequest
def submit_preprocess_jobs(clt):
    request = SubmitPreprocessJobsRequest.SubmitPreprocessJobsRequest()

    request.set_VideoId('<videoId>')
    request.set_PreprocessType('PreprocessType')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    jobs = submit_preprocess_jobs(clt)
    print(json.dumps(jobs, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.8. 视频水印

本篇文档提供了Python SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印、设置默认水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

> **说明**
>
> - 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
> - 水印文件OSS上传详细参数请参见OSS文件上传。

```
from aliyunsdkvod.request.v20170321 import AddWatermarkRequest
def add_watermark(clt):
    request = AddWatermarkRequest.AddWatermarkRequest()

    request.set_Name('watermark-sample')
    # 图片水印必须传图片文件的OSS地址，且水印文件需要和视频在同一区域，如:华东2的视频添加水印，水印文件必须存放在华东2
    # FileUrl示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermark/test-****.png
      request.set_FileUrl('<your File URL>')

    # 文字水印设置示例
    request.set_Type('Text')
    # 水印的文本内容、字体、大小、颜色、透明度等配置
    watermarkConfig = {'Content': 'watermark Text', 'FontName': 'SimSun', 'FontSize': 25, 'FontColor': 'Black',
                       'FontAlpha': 0.2, 'BorderColor': 'White', 'BorderWidth': 1, 'Top': 20, 'Left': 15}
    request.set_WatermarkConfig(json.dumps(watermarkConfig))

    """
    # 图片水印设置示例
    request.set_Type('Image')
    # 水印显示的开始时间和结束时间
    timeline = {'Start': 2, 'Duration': 'ToEND'}
    # 水印的位置等配置
    watermarkConfig = {'Dx': 8, 'Dy': 8, 'Width': 55, 'Height': 55, 'ReferPos': 'BottomRight', 'Timeline': timeline}
    request.set_WatermarkConfig(json.dumps(watermarkConfig))
    """

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    watermark = add_watermark(clt)
    print(json.dumps(watermark, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

> **注意**
>
> 本接口不支持图片水印图片文件地址的修改，如要更换请创建新的水印。

```
from aliyunsdkvod.request.v20170321 import UpdateWatermarkRequest
def update_watermark(clt):
    request = UpdateWatermarkRequest.UpdateWatermarkRequest()
    request.set_WatermarkId('<watermarkId>')
    request.set_Name('new-watermark-name')

    # 更改文字水印设置示例
    # 水印的文本内容、字体、大小、颜色、透明度等配置
    watermarkConfig = {'Content': 'watermark Text', 'FontName': 'SimSun', 'FontSize': 25, '
FontColor': 'Black',
                       'FontAlpha': 0.2, 'BorderColor': 'White', 'BorderWidth': 1, 'Top': 2
0, 'Left': 15}
    request.set_WatermarkConfig(json.dumps(watermarkConfig))

    """
    # 更改图片水印设置示例
    # 水印显示的开始时间和结束时间
    timeline = {'Start': 2, 'Duration': 'ToEND'}
    # 水印的位置等配置
    watermarkConfig = {'Dx': 8, 'Dy': 8, 'Width': 55, 'Height': 55, 'ReferPos': 'BottomRigh
t', 'Timeline': timeline}
    request.set_WatermarkConfig(json.dumps(watermarkConfig))
    """

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    watermark = update_watermark(clt)
    print(json.dumps(watermark, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteWatermarkRequest
def delete_watermark(clt):
    request = DeleteWatermarkRequest.DeleteWatermarkRequest()
    request.set_WatermarkId('<watermarkId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = delete_watermark(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListWatermarkRequest
def list_watermark(clt):
    request = ListWatermarkRequest.ListWatermarkRequest()
    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    watermarks = list_watermark(clt)
    print(watermarks['WatermarkInfos'])
    print(json.dumps(watermarks, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetWatermarkRequest
def get_watermark(clt):
    request = GetWatermarkRequest.GetWatermarkRequest()
    request.set_WatermarkId('<watermarkId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    watermark = get_watermark(clt)
    print(watermark['WatermarkInfo'])
    print(json.dumps(watermark, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SetDefaultWatermarkRequest
def set_default_watermark(clt):
    request = SetDefaultWatermarkRequest.SetDefaultWatermarkRequest()
    request.set_WatermarkId('<watermarkId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = set_default_watermark(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.9. 转码模板

本篇文档提供了Python SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import AddTranscodeTemplateGroupRequest

""" 构建需要添加的转码模板配置数据 """
def build_transcode_template_list():
    transcodeTemplateList = []
    transcodeTemplate = {}

    # 设置模板名称
    transcodeTemplate["TemplateName"] = "MP4-for-Low-Definition"

    # 清晰度
    transcodeTemplate["Definition"] = "LD"

    # 视频流转码配置
    videoConfig = {"Width": 640, "Bitrate": 400, "Fps": 25, "Remove": False, "Codec": "H.26
4", "Gop": "250"}
    transcodeTemplate["Video"] = videoConfig

    # 音频流转码配置
    audioConfig = {"Codec": "AAC", "Bitrate": "64", "Channels": "2", "Samplerate": "32000"}
    transcodeTemplate["Audio"] = audioConfig

    # 封装容器
    container = {"Format": "mp4"}
    transcodeTemplate["Container"] = container

    # 条件转码配置
    condition = {"IsCheckReso": False, "IsCheckResoFail": False, "IsCheckVideoBitrate": Fal
se,
                 "IsCheckVideoBitrateFail": False, "IsCheckAudioBitrate": False, "IsCheckAu
dioBitrateFail": False}
    transcodeTemplate["TransConfig"] = condition

    """
    # 加密配置(只支持HLS)
    encryptSetting = {"EncryptType": "Private"}
    transcodeTemplate["EncryptSetting"] = encryptSetting
    """

    # 水印ID(多水印关联)
    watermarkIdList = []
    #watermarkIdList.append("<WatermarkId>")
    watermarkIdList.append("USER_DEFAULT_WATERMARK")
    transcodeTemplate["WatermarkIds"] = watermarkIdList

    transcodeTemplateList.append(transcodeTemplate)

    return transcodeTemplateList
```

```
""" 添加转码模板配置 """
def add_transcode_template_group(clt):
    request = AddTranscodeTemplateGroupRequest.AddTranscodeTemplateGroupRequest()

    request.set_Name("SampleTranscodeTemplateGroup")

    transcodeTemplateList = build_transcode_template_list()
    request.set_TranscodeTemplateList(json.dumps(transcodeTemplateList))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = add_transcode_template_group(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateTranscodeTemplateGroupRequest

""" 构建需要修改的转码模板配置数据 """
def build_transcode_template_list():
    transcodeTemplateList = []
    transcodeTemplate = {}

    # 设置转码模板ID，必填
    transcodeTemplate["TranscodeTemplateId"] = "<TranscodeTemplateId>"

    # 以下都是可选项

    # 设置新的模板名称(如果需要更改)
    transcodeTemplate["TemplateName"] = "new-MP4-for-Low-Definition"

    # 视频流转码配置
    videoConfig = {"Width": 640, "Bitrate": 500, "Fps": 60, "Remove": False, "Codec": "H.26
4", "Gop": "250"}
    transcodeTemplate["Video"] = videoConfig

    # 音频流转码配置
    audioConfig = {"Codec": "AAC", "Bitrate": "128", "Channels": "2", "Samplerate": "32000"
}
    transcodeTemplate["Audio"] = audioConfig
```

```
    # 封装容器
    container = {"Format": "mp4"}
    transcodeTemplate["Container"] = container

    # 条件转码配置
    condition = {"IsCheckReso": False, "IsCheckResoFail": False, "IsCheckVideoBitrate": Fal
se,
                 "IsCheckVideoBitrateFail": False, "IsCheckAudioBitrate": False, "IsCheckAu
dioBitrateFail": False}
    transcodeTemplate["TransConfig"] = condition

    """
    # 加密配置(只支持HLS)
    encryptSetting = {"EncryptType": "Private"}
    transcodeTemplate["EncryptSetting"] = encryptSetting
    """

    # 水印ID(多水印关联)
    watermarkIdList = []
    #watermarkIdList.append("<WatermarkId>")
    watermarkIdList.append("USER_DEFAULT_WATERMARK")
    transcodeTemplate["WatermarkIds"] = watermarkIdList

    transcodeTemplateList.append(transcodeTemplate)

    return transcodeTemplateList


""" 修改转码模板配置 """
def update_transcode_template_group(clt):
    request = UpdateTranscodeTemplateGroupRequest.UpdateTranscodeTemplateGroupRequest()

    # 设置转码模板组ID
    request.set_TranscodeTemplateGroupId("<TranscodeTemplateGroupId>")

    # 设置转码模板配置
    transcodeTemplateList = build_transcode_template_list()
    request.set_TranscodeTemplateList(json.dumps(transcodeTemplateList))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = update_transcode_template_group(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListTranscodeTemplateGroupRequest
def list_transcode_template_group(clt):
    request = ListTranscodeTemplateGroupRequest.ListTranscodeTemplateGroupRequest()

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    data = list_transcode_template_group(clt)
    for item in data['TranscodeTemplateGroupList']:
        print("TranscodeTemplateGroupId: %s, Name: %s" % (item['TranscodeTemplateGroupId'],
item['Name']))

    print(json.dumps(data, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetTranscodeTemplateGroupRequest
def get_transcode_template_group(clt):
    request = GetTranscodeTemplateGroupRequest.GetTranscodeTemplateGroupRequest()
    request.set_TranscodeTemplateGroupId("<TranscodeTemplateGroupId>")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    templateGroup = get_transcode_template_group(clt)
    for item in templateGroup['TranscodeTemplateGroup']['TranscodeTemplateList']:
        print("TranscodeTemplateId: %s, TemplateName: %s, Definition: %s" %
              (item['TranscodeTemplateId'], item['TemplateName'], item['Definition']))

    print(json.dumps(templateGroup, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SetDefaultTranscodeTemplateGroupRequest
def set_default_transcode_template_group(clt):
    request = SetDefaultTranscodeTemplateGroupRequest.SetDefaultTranscodeTemplateGroupReque
st()
    request.set_TranscodeTemplateGroupId("<TranscodeTemplateGroupId>")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = set_default_transcode_template_group(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteTranscodeTemplateGroupRequest
def delete_transcode_template_group(clt):
    request = DeleteTranscodeTemplateGroupRequest.DeleteTranscodeTemplateGroupRequest()
    request.set_TranscodeTemplateGroupId("<TranscodeTemplateGroupId>")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = delete_transcode_template_group(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.10. 截图模板

本篇文档提供了Python SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import AddVodTemplateRequest
def add_vod_template(clt):
    request = AddVodTemplateRequest.AddVodTemplateRequest()
    # 模板名称
    request.set_Name('Sample Snapshot Template')
    # 模板类型，固定值为Snapshot
    request.set_TemplateType('Snapshot')

    # 截图模板配置数据
    snapshotConfig = {'Count': 50, 'Interval': 1, 'SpecifiedOffsetTime': 0, 'Width': 200, '
Height': 200,
                      'FrameType': 'normal'}
    templateConfig = {'SnapshotConfig': snapshotConfig, 'SnapshotType': 'NormalSnapshot'}

    """
    # 可选项：雪碧图配置(雪碧图配置必须建立在普通截图配置之上)
    spriteSnapshotConfig = {'CellWidth': 120, 'CellHeight': 68, 'Columns': 3, 'Lines': 10,
'Padding': 20,
                            'Margin': 50, 'KeepCellPic': 'keep', 'Color': 'tomato'}
    snapshotConfig['SpriteSnapshotConfig'] = spriteSnapshotConfig
    templateConfig['SnapshotType'] = 'SpriteSnapshot'
    """

    request.set_TemplateConfig(json.dumps(templateConfig))


    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    template = add_vod_template(clt)
    print(template['VodTemplateId'])
    print(json.dumps(template, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateVodTemplateRequest
def update_vod_template(clt):
    request = UpdateVodTemplateRequest.UpdateVodTemplateRequest()
    request.set_VodTemplateId('<templateId>')
    # 修改模板名称
    request.set_Name('New Snapshot Template Name')
    # 修改截图模板配置数据
    snapshotConfig = {'Count': 50, 'Interval': 1, 'SpecifiedOffsetTime': 0, 'Width': 200, '
Height': 200,
                      'FrameType': 'normal'}
    templateConfig = {'SnapshotConfig': snapshotConfig, 'SnapshotType': 'NormalSnapshot'}
    request.set_TemplateConfig(json.dumps(templateConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    template = update_vod_template(clt)
    print(json.dumps(template, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteVodTemplateRequest
def delete_vod_template(clt):
    request = DeleteVodTemplateRequest.DeleteVodTemplateRequest()
    request.set_VodTemplateId('<templateId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = delete_vod_template(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListVodTemplateRequest
def list_vod_template(clt):
    request = ListVodTemplateRequest.ListVodTemplateRequest()
    # 模板类型，固定值为Snapshot
    request.set_TemplateType('Snapshot')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    templates = list_vod_template(clt)
    print(templates['VodTemplateInfoList'])
    print(json.dumps(templates, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetVodTemplateRequest
def get_vod_template(clt):
    request = GetVodTemplateRequest.GetVodTemplateRequest()
    request.set_VodTemplateId('<templateId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    template = get_vod_template(clt)
    print(template['VodTemplateInfo'])
    print(json.dumps(template, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.11. 视频剪辑

本篇文档提供了Python SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

> **说明**
>
> 通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
from aliyunsdkvod.request.v20170321 import ProduceEditingProjectVideoRequest
def produce_editing_video(clt):
    request = ProduceEditingProjectVideoRequest.ProduceEditingProjectVideoRequest()

    # set timeline, this sample shows how to merge two videos
    videoTrackClips = []
    videoTracks = []
    timeline = {}
    videoTrackClip1 = {'MediaId': '<videoId1>'}
    videoTrackClips.append(videoTrackClip1)
    videoTrackClip2 = {'MediaId': '<videoId2>'}
    videoTrackClips.append(videoTrackClip2)
    videoTrack = {'VideoTrackClips': videoTrackClips}
    videoTracks.append(videoTrack)
    timeline['VideoTracks'] = videoTracks
    request.set_Timeline(json.dumps(timeline))

    # set media metadata
    # CoverURL示例：http://example.aliyundoc.com/media/cover/mediaid****.jpg
    mediaMetadata = {'Title': 'editing sample title', 'Description': 'editing sample descri
ption', 'Tags': 'Tag1,Tag2,Test',
                     'CoverURL': '<your Cover URL>'}
    request.set_MediaMetadata(json.dumps(mediaMetadata))

    # set produce config
    produceConfig = {'TemplateGroupId': '<templateGroupId>'}
    #request.set_ProduceConfig(json.dumps(produceConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    produce = produce_editing_video(clt)
    print(produce['MediaId'])
    print(json.dumps(produce, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ProduceEditingProjectVideoRequest
def produce_editing_video_by_id(clt):
    request = ProduceEditingProjectVideoRequest.ProduceEditingProjectVideoRequest()

    # set ProjectId
    request.set_ProjectId('<ProjectId>')

    # set media metadata
    mediaMetadata = {'Title': 'editing sample title by ProjectId', 'Description': 'editing
sample description by ProjectId',
                     'Tags': 'Tag1,Tag2,Test'}
    request.set_MediaMetadata(json.dumps(mediaMetadata))

    # set produce config
    produceConfig = {'TemplateGroupId': '<templateGroupId>'}
    #request.set_ProduceConfig(json.dumps(produceConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    produce = produce_editing_video_by_id(clt)
    print(produce['MediaId'])
    print(json.dumps(produce, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import AddEditingProjectRequest
def add_editing_project(clt):
    request = AddEditingProjectRequest.AddEditingProjectRequest()

    # set timeline, this sample shows how to merge two videos
    videoTrackClips = []
    videoTracks = []
    timeline = {}
    videoTrackClip1 = {'MediaId': '<videoId1>'}
    videoTrackClips.append(videoTrackClip1)
    videoTrackClip2 = {'MediaId': '<videoId2>'}
    videoTrackClips.append(videoTrackClip2)
    videoTrack = {'VideoTrackClips': videoTrackClips}
    videoTracks.append(videoTrack)
    timeline['VideoTracks'] = videoTracks
    request.set_Timeline(json.dumps(timeline))

    # set project metadata
    request.set_Title('editing project title')
    request.set_Description('editing project description')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    project = add_editing_project(clt)
    print(project['Project'])
    print(json.dumps(project, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateEditingProjectRequest
def update_editing_project(clt):
    request = UpdateEditingProjectRequest.UpdateEditingProjectRequest()

    # set projectId
    request.set_ProjectId('<projectId>')

    # set timeline, this sample shows how to merge two videos
    videoTrackClips = []
    videoTracks = []
    timeline = {}
    videoTrackClip1 = {'MediaId': '<videoId1>'}
    videoTrackClips.append(videoTrackClip1)
    videoTrackClip2 = {'MediaId': '<videoId2>'}
    videoTrackClips.append(videoTrackClip2)
    videoTrack = {'VideoTrackClips': videoTrackClips}
    videoTracks.append(videoTrack)
    timeline['VideoTracks'] = videoTracks
    request.set_Timeline(json.dumps(timeline))

    # set project metadata
    request.set_Title('new editing project title')
    request.set_Description('new editing project description')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = update_editing_project(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteEditingProjectRequest
def delete_editing_project(clt):
    request = DeleteEditingProjectRequest.DeleteEditingProjectRequest()

    # set projectId list
    request.set_ProjectIds('projectId1,projectId2')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = delete_editing_project(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetEditingProjectRequest
def get_editing_project(clt):
    request = GetEditingProjectRequest.GetEditingProjectRequest()

    # set projectId
    request.set_ProjectId('<ProjectId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    project = get_editing_project(clt)
    print(json.dumps(project, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SearchEditingProjectRequest
def search_editing_project(clt):
    request = SearchEditingProjectRequest.SearchEditingProjectRequest()

    request.set_Title('Title Keywords')
    request.set_StartTime('2018-10-11T12:00:00Z')
    request.set_EndTime('2018-12-25T12:00:00Z')
    request.set_PageSize(10)
    request.set_PageNo(1)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    projects = search_editing_project(clt)
    print(projects['ProjectList']['Project'])
    print(json.dumps(projects, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SetEditingProjectMaterialsRequest
def set_editing_project_materials(clt):
    request = SetEditingProjectMaterialsRequest.SetEditingProjectMaterialsRequest()

    request.set_ProjectId('<ProjectId>')
    request.set_MaterialIds('materialId1,materialId2')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = set_editing_project_materials(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetEditingProjectMaterialsRequest
def get_editing_project_materials(clt):
    request = GetEditingProjectMaterialsRequest.GetEditingProjectMaterialsRequest()

    request.set_ProjectId('<ProjectId>')
    request.set_Type('video')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_editing_project_materials(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.12. 媒体审核

本篇文档提供了Python SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SubmitAIMediaAuditJobRequest
def submit_ai_media_audit_job(clt):
    request = SubmitAIMediaAuditJobRequest.SubmitAIMediaAuditJobRequest()
    request.set_MediaId('<VideoId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = submit_ai_media_audit_job(clt)
    print(res['JobId'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetAIMediaAuditJobRequest
def get_ai_media_audit_job(clt):
    request = GetAIMediaAuditJobRequest.GetAIMediaAuditJobRequest()
    request.set_JobId('<JobId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_ai_media_audit_job(clt)
    print(res['MediaAuditJob'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetMediaAuditResultRequest
def get_media_audit_result(clt):
    request = GetMediaAuditResultRequest.GetMediaAuditResultRequest()
    request.set_MediaId('<VideoId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_media_audit_result(clt)
    print(res['MediaAuditResult'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetMediaAuditResultDetailRequest
def get_media_audit_result_detail(clt):
    request = GetMediaAuditResultDetailRequest.GetMediaAuditResultDetailRequest()
    request.set_MediaId('<VideoId>')
    request.set_PageNo(1)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_media_audit_result_detail(clt)
    print(res['MediaAuditResultDetail']['Total'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetMediaAuditResultTimelineRequest
def get_media_audit_result_timeline(clt):
    request = GetMediaAuditResultTimelineRequest.GetMediaAuditResultTimelineRequest()
    request.set_MediaId('<VideoId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = get_media_audit_result_timeline(clt)
    print(res['MediaAuditResultTimeline']['Porn'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import CreateAuditRequest
def create_audit(clt):
    request = CreateAuditRequest.CreateAuditRequest()

    auditContent = []
    # 分别指定视频ID、审核状态和审核理由
    auditItem = {'VideoId': '<videoId>', 'Status': 'Blocked', 'Reason': 'porn video'}
    auditContent.append(auditItem)
    request.set_AuditContent(json.dumps(auditContent))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = create_audit(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetAuditHistoryRequest
def get_audit_history(clt):
    request = GetAuditHistoryRequest.GetAuditHistoryRequest()

    request.set_VideoId('<videoId>')
    request.set_PageNo(1)
    request.set_PageSize(10)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    history = get_audit_history(clt)
    print(history['Histories'])
    print(json.dumps(history, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SetAuditSecurityIpRequest
def set_audit_security_ip(clt):
    request = SetAuditSecurityIpRequest.SetAuditSecurityIpRequest()

    # 审核安全组名称，默认为Default
    request.set_SecurityGroupName('MyGroupName')
    # 操作方式，默认值Append(追加IP白名单)
    request.set_OperateMode('Cover')
    # 审核安全IP列表，每个分组最多支持100个并以逗号分隔
    request.set_Ips('10.23.12.20,10.23.12.21,10.23.12.22')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = set_audit_security_ip(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListAuditSecurityIpRequest
def list_audit_security_ip(clt):
    request = ListAuditSecurityIpRequest.ListAuditSecurityIpRequest()

    # 审核安全组名称
    request.set_SecurityGroupName('MyGroupName')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    ips = list_audit_security_ip(clt)
    print(ips['SecurityIpList'])
    print(json.dumps(ips, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.13. 视频AI

本篇文档提供了Python SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化接口。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SubmitAIJobRequest
def submit_ai_job(clt):
    request = SubmitAIJobRequest.SubmitAIJobRequest()
    request.set_MediaId('<videoId>')  # 视频ID
    request.set_Types('AIVideoCover')  # AI类型，请确保已开通该类型AI服务

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    job = submit_ai_job(clt)
    print(json.dumps(job, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListAIJobRequest
def list_ai_job(clt):
    request = ListAIJobRequest.ListAIJobRequest()
    jobIds = ['jobId1', 'jobId2']
    request.set_JobIds(','.join(jobIds))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    jobs = list_ai_job(clt)
    print(json.dumps(jobs, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import AddAITemplateRequest
def add_ai_template(clt):
    request = AddAITemplateRequest.AddAITemplateRequest()

    # 设置模板类型，以智能审核模板为例
    request.set_TemplateType('AIMediaAudit')
    # 设置模板名称
    request.set_TemplateName('My AI Template')
    # 设置模板详细配置
    auditItem = ['terrorism', 'porn']
    auditRange = ['video', 'image-cover', 'text-title']
    auditContent = ['screen']
    templateConfig = {'AuditItem': auditItem, 'AuditRange': auditRange, 'AuditContent': aud
itContent,
                      'AuditAutoBlock': 'no'}
    request.set_TemplateConfig(json.dumps(templateConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = add_ai_template(clt)
    print(res['TemplateId'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DeleteAITemplateRequest
def delete_ai_template(clt):
    request = DeleteAITemplateRequest.DeleteAITemplateRequest()
    request.set_TemplateId('<TemplateId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = delete_ai_template(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import UpdateAITemplateRequest
def update_ai_template(clt):
    request = UpdateAITemplateRequest.UpdateAITemplateRequest()
    # 设置模板ID
    request.set_TemplateId('<TemplateId>')
    # 设置模板名称
    request.set_TemplateName('New AI Template Name')
    # 设置模板详细配置
    auditItem = ['terrorism', 'porn']
    auditRange = ['video', 'image-cover']
    auditContent = ['screen']
    templateConfig = {'AuditItem': auditItem, 'AuditRange': auditRange, 'AuditContent': aud
itContent,
                      'AuditAutoBlock': 'yes'}
    request.set_TemplateConfig(json.dumps(templateConfig))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = update_ai_template(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetAITemplateRequest
def get_ai_template(clt):
    request = GetAITemplateRequest.GetAITemplateRequest()
    request.set_TemplateId('<TemplateId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    template = get_ai_template(clt)
    print(template['TemplateInfo'])

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListAITemplateRequest
def list_ai_template(clt):
    request = ListAITemplateRequest.ListAITemplateRequest()
    request.set_TemplateType('AIMediaAudit')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    templates = list_ai_template(clt)
    print(templates['TemplateInfoList'])

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SetDefaultAITemplateRequest
def set_default_ai_template(clt):
    request = SetDefaultAITemplateRequest.SetDefaultAITemplateRequest()
    request.set_TemplateId('<TemplateId>')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = set_default_ai_template(clt)
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetDefaultAITemplateRequest
def get_default_ai_template(clt):
    request = GetDefaultAITemplateRequest.GetDefaultAITemplateRequest()
    request.set_TemplateType('AIMediaAudit')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    template = get_default_ai_template(clt)
    print(template['TemplateInfo'])
    print(json.dumps(template, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.14. 视频DNA

本文为您提供了Python SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import SubmitAIJobRequest
def submit_ai_job(clt):
    request = SubmitAIJobRequest.SubmitAIJobRequest()
    request.set_MediaId('<videoId>')  # 视频ID
    request.set_Types('AIMediaDNA')  # 设置AI类型，类型为AIMediaDNA

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    job = submit_ai_job(clt)
    print(json.dumps(job, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListAIJobRequest
def list_ai_job(clt):
    request = ListAIJobRequest.ListAIJobRequest()
    jobIds = ['jobId1', 'jobId2']
    request.set_JobIds(','.join(jobIds))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    jobs = list_ai_job(clt)
    print(json.dumps(jobs, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import GetMediaDNAResultRequest
def get_media_dna_result(clt):
    request = GetMediaDNAResultRequest.GetMediaDNAResultRequest()
    request.set_MediaId('<videoId>')  # 视频ID

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    dna = get_media_dna_result(clt)
    print(json.dumps(dna, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.15. 直播转点播

本篇文档提供了Python SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import ListLiveRecordVideoRequest
def list_live_record_video(clt):
    request = ListLiveRecordVideoRequest.ListLiveRecordVideoRequest()
    request.set_StartTime("2018-04-24T03:21:04Z")
    request.set_EndTime("2018-05-21T03:21:44Z")
    request.set_StreamName("testStream****")
    request.set_DomainName("example.aliyundoc.com")
    request.set_AppName("testApp****")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    videoList = list_live_record_video(clt)
    print(json.dumps(videoList, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

3.16. 点播CDN

本篇文档提供了Python SDK点播CDN模块相关功能的API调用示例。包含预热缓存、刷新缓存、查询刷新和预热状态、查询刷新预热次数限制和余量、查询流量数据、查询网络带宽、下载域名日志。

初始化客户端

使用前请先初始化客户端，请参见初始化。

预热缓存

调用PreloadVodObjectCaches接口，完成预热缓存功能。

接口参数和返回字段请参见PreloadVodObjectCaches。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import PreloadVodObjectCachesRequest
def preload_object_caches(clt):
    request = PreloadVodObjectCachesRequest.PreloadVodObjectCachesRequest()
    objectPath = ['http://192.168.0.0/16/fd.mp4',
                  'http://192.168.0.0/16/ld.mp4']
    request.set_ObjectPath("\n".join(objectPath))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = preload_object_caches(clt)
    print(res['PreloadTaskId'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

刷新缓存

调用RefreshVodObjectCaches接口，完成刷新缓存功能。

接口参数和返回字段请参见RefreshVodObjectCaches。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import RefreshVodObjectCachesRequest
def refresh_object_caches(clt):
    request = RefreshVodObjectCachesRequest.RefreshVodObjectCachesRequest()
    objectPath = ['http://192.168.0.0/16/fd.mp4',
                  'http://192.168.0.0/16/ld.mp4']
    request.set_ObjectPath("\n".join(objectPath))

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = refresh_object_caches(clt)
    print(res['RefreshTaskId'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询刷新和预热状态

调用DescribeVodRefreshTasks接口，完成查询刷新和预热状态功能。

接口参数和返回字段请参见DescribeVodRefreshTasks。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DescribeVodRefreshTasksRequest
def describe_refresh_task(clt):
    request = DescribeVodRefreshTasksRequest.DescribeVodRefreshTasksRequest()
    request.set_DomainName('example.com')
    request.set_ObjectType('preload')

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = describe_refresh_task(clt)
    print(res['Tasks']['Task'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询刷新预热次数限制和余量

调用DescribeVodRefreshQuota接口，完成查询刷新预热次数限制和余量功能。

接口参数和返回字段请参见DescribeVodRefreshQuota。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DescribeVodRefreshQuotaRequest
def describe_refresh_quota(clt):
    request = DescribeVodRefreshQuotaRequest.DescribeVodRefreshQuotaRequest()

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = describe_refresh_quota(clt)
    print(res['PreloadRemain'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询流量数据

调用DescribeVodDomainTrafficData接口，完成查询流量数据功能。

接口参数和返回字段请参见DescribeVodDomainTrafficData。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DescribeVodDomainTrafficDataRequest
def describe_domain_traffic_data(clt):
    request = DescribeVodDomainTrafficDataRequest.DescribeVodDomainTrafficDataRequest()
    request.set_DomainName("example.aliyundoc.com")
    request.set_StartTime("2019-01-15T15:59:59Z")
    request.set_EndTime("2019-01-20T15:59:58Z")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = describe_domain_traffic_data(clt)
    print(res['TrafficDataPerInterval']['DataModule'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

查询网络带宽

调用DescribeVodDomainBpsData接口，完成查询网络带宽功能。

接口参数和返回字段请参见DescribeVodDomainBpsData。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DescribeVodDomainBpsDataRequest
def describe_domain_bps_data(clt):
    request = DescribeVodDomainBpsDataRequest.DescribeVodDomainBpsDataRequest()
    request.set_DomainName("example.aliyundoc.com")
    request.set_StartTime("2019-01-15T15:59:59Z")
    request.set_EndTime("2019-01-20T15:59:58Z")

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = describe_domain_bps_data(clt)
    print(res['BpsDataPerInterval']['DataModule'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

下载域名日志

调用DescribeVodDomainLog接口，完成下载域名日志功能。

接口参数和返回字段请参见DescribeVodDomainLog。调用示例如下：

```
from aliyunsdkvod.request.v20170321 import DescribeVodDomainLogRequest
def describe_domain_log(clt):
    request = DescribeVodDomainLogRequest.DescribeVodDomainLogRequest()
    request.set_DomainName("example.aliyundoc.com")
    request.set_StartTime("2019-01-15T15:59:59Z")
    request.set_EndTime("2019-01-20T15:59:58Z")
    request.set_PageNumber(1)
    request.set_PageSize(300)

    request.set_accept_format('JSON')
    response = json.loads(clt.do_action_with_exception(request))
    return response

try:
    clt = init_vod_client('<AccessKeyId>', '<AccessKeySecret>')
    res = describe_domain_log(clt)
    logDetail = res['DomainLogDetails']['DomainLogDetail'][0]
    print("TotalLogCount: %s" % (logDetail['PageInfos']['Total']))
    print(logDetail['LogInfos']['LogInfoDetail'])
    print(json.dumps(res, ensure_ascii=False, indent=4))

except Exception as e:
    print(e)
    print(traceback.format_exc())
```

4.PHP SDK

4.1. 安装SDK

4.1.1. 安装

本篇文档为您介绍了安装PHP SDK的环境要求，以及安装PHP SDK、更新PHP SDK、卸载PHP SDK的操作步骤及说明。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

PHP 5.3及以上版本，您可通过 `php -v` 命令查看当前的PHP版本。关于如何安装PHP，请参见PHP官网下载合适的版本安装。

安装SDK

> **说明** 以下安装步骤为旧版SDK，旧版SDK已不再维护，建议使用支持Composer的新版SDK，详细内容请参见新版安装。

1. 在您的PHP项目中添加空文件夹aliyun-php-sdk（为便于统一管理SDK文件）。
2. 在aliyun-openapi-php-sdk下载整个源码，解压后复制aliyun-php-sdk-core和aliyun-php-sdk-vod两个文件夹到aliyun-php-sdk目录下。

> **说明** 建议aliyun-php-sdk-core使用1.3.8版本及以上，aliyun-php-sdk-vod使用2.15.1版本及以上。

目录结构说明

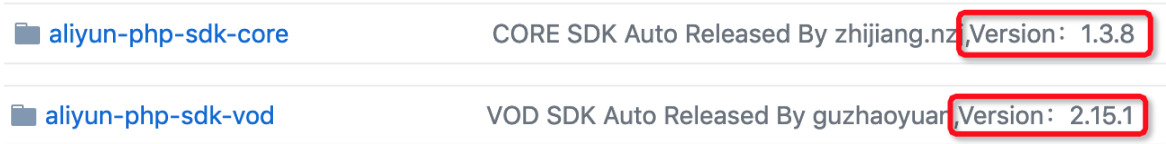
序号	说明
①	PHP集成的SDK

序号	说明
②	应用程序 说明 Demo.php为示例名称，此应用程序为用户自定义，可根据实际业务需要设置名称。

更新SDK

若发现当前SDK没有新的接口或已有接口没有新的功能，可更新到最新版。具体方法如下：

1. 打开aliyun-php-sdk-core和aliyun-php-sdk-vod目录下的ChangeLog.txt查看已下载SDK的版本号（首行）。
2. 可以在下载页面看到SDK的最新版本号。



说明 以上版本号仅为当前时间示例，请以实际为准。

3. 从aliyun-openapi-php-sdk下载整个源码，解压后复制aliyun-php-sdk-core和aliyun-php-sdk-vod两个文件夹到aliyun-php-sdk目录下，覆盖原有文件。

卸载SDK

直接删除本地SDK目录即可。

4.1.2. 初始化

PHP SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经开通了视频点播服务。开通步骤请参见开通视频点播。
- 您已经知晓服务接入地域。例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见点播中心和访问域名。

AK对初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccseeKey信息，进行初始化。示例如下：

说明

如果接入其它区域有问题，请更新SDKaliyun-php-sdk-core到1.3.8版本及以上，aliyun-php-sdk-vod到2.15.1版本及以上。

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

try {
    $client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');
    var_dump($client);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

STS初始化

使用STS方法进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret, $securityToken) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret, $securityToken);
    return new DefaultAcsClient($profile);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>', '<SecurityToken>');
    var_dump($client);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.2. 使用新版（Composer）

4.2.1. 安装

本文为您介绍了视频点播服务端PHP SDK的安装及更新步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

简介

Alibaba Cloud SDK for PHP 是支持产品快捷访问的开发包，由Alibaba Cloud Client for PHP提供底层支持。推荐您通过Composer安装PHP SDK，如果不能使用Composer，也可以通过Zip文件方式安装PHP SDK。

> **说明** 新版本的SDK安装和使用与旧版本的SDK完全不同，请您继续阅读以下文档，更多详情请参见Alibaba Cloud SDK for PHP。

环境要求

- 要求
  - 安装PHP5.5.0及以上版本。

    > **说明** 可通过 `php -v` 命令查看当前的PHP版本；若未安装，请到PHP官网。

  - 使用Composer并优化自动加载 `composer dump-autoload --optimize` 。
  - 如果您使用了RsaKeyPair客户端（仅支持日本站），还需要OpenSSL PHP扩展。
- 建议
  - 安装cURL7.16.2 或更高版本。
  - 使用OPcache。
  - 生产环境中不要使用Xdebug。

通过Composer安装（推荐）

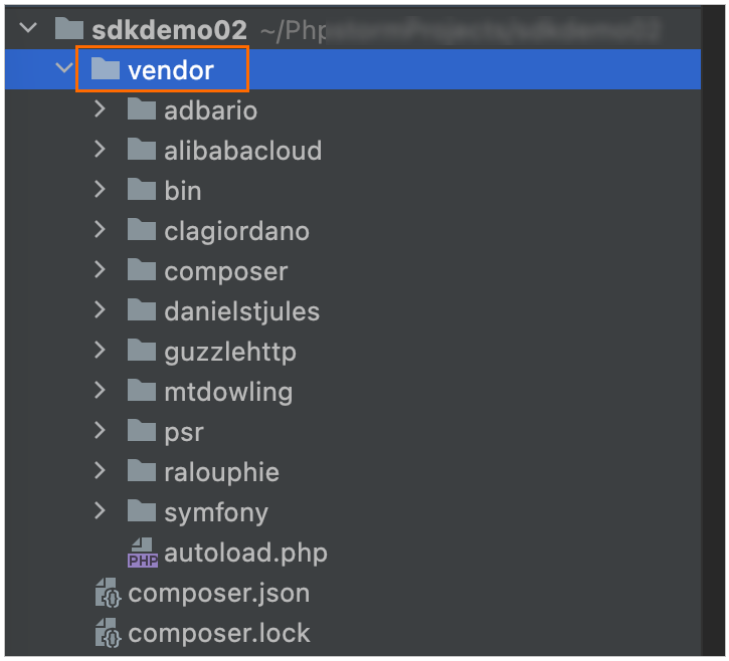
Composer是一款 PHP工具，用于管理和安装项目的依赖项。有关如何安装Composer、配置自动加载并遵循定义依赖关系的其他最佳实践的更多信息，请参见Composer下载。

1. 安装依赖。

i. 如果您已在系统上Composer，请直接在项目目录中运行以下内容来安装Alibaba Cloud SDK for PHP作为依赖项。示例如下：

```
composer require alibabacloud/sdk
```

执行完成后会在项目目录下生成*vendor*目录。如下图所示：



如果您未安装，请下载并安装全局安装 Composer（Windows 用户请下载并运行 Composer-Setup.exe），示例如下：

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
```

> **说明** 如果不支持HTTPS，可使用 `curl http://getcomposer.org/installer | php` ，更多安装信息请参见Composer Download。

ii. 执行Composer命令安装最新版Alibaba Cloud SDK for PHP作为依赖项，示例如下：

```
composer require alibabacloud/sdk
```

> **说明** 如果由于网络问题无法安装，可尝试切换Composer镜像地址。

2. 将自动加载工具添加到PHP脚本。示例如下：

```
<?php
require __DIR__ . '/vendor/autoload.php';
```

> **说明** 要在脚本中使用Alibaba Cloud SDK for PHP，需要在脚本中包含自动加载工具。

使用ZIP文件安装

如果不能使用Composer，也可以下载包含所有类和依赖项的ZIP文件进行使用。下载地址请参见服务端接口SDK。

在项目中的选定位置解压，最后将自动加载工具包含到您的脚本中，示例如下：

```
<?php
require __DIR__ . '/vendor/autoload.php';
```

更新SDK

若发现当前SDK没有新的接口或已有接口没有新的功能，可更新到最新版。示例如下：

```
php -d memory_limit=-1 composer.phar require alibabacloud/sdk
```

4.2.2. 初始化

新版PHP SDK支持Composer方式，与旧版在安装、初始化和使用上完全不同，请提前阅读安装说明，并参考以下文档进行初始化和使用。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端PHP SDK。安装方法，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

使用说明

- 目前新版PHP SDK详细代码信息请参见OpenAPI。
- 创建API请求时，调用的方法名为 `Vod::v20170321()->${apiName}` ，其中 `${apiName}` 需要替换成要使用的点播API，接口名称的首字母必须调整为小写。可用的API名称请参见API概览。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。本文以PHP8.0.16为例，示例代码如下：

```
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Vod\Vod;
define("VOD_CLIENT_NAME", 'AliyunVodClientDemo');
function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';
    //填入AccessKey信息
    AlibabaCloud::accessKeyClient($accessKeyId, $accessKeySecret)
        ->regionId($regionId)
        ->connectTimeout(1)
        ->timeout(3)
        ->name(VOD_CLIENT_NAME);
}
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。本文以PHP8.0.16为例，示例代码如下：

```
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Client\Exception\ClientException;
use AlibabaCloud\Client\Exception\ServerException;
use AlibabaCloud\Vod\Vod;
define("VOD_CLIENT_NAME", 'AliyunVodClientDemo');
function initVodClient($accessKeyId, $accessKeySecret, $securityToken) {
    $regionId = 'cn-shanghai';
    //填入STS信息
    AlibabaCloud::stsClient($accessKeyId, $accessKeySecret, $securityToken)
        ->regionId($regionId)
        ->connectTimeout(1)
        ->timeout(3)
        ->name(VOD_CLIENT_NAME);
}
```

调用示例

本文以PHP8.0.16调用获取播放地址接口GetPlayInfo为例。示例如下：

```
<?php
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Vod\Vod;
define("VOD_CLIENT_NAME", 'AliyunVodClientDemo');
function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';
    //填入AccessKey信息
    AlibabaCloud::accessKeyClient($accessKeyId, $accessKeySecret)
        ->regionId($regionId)
        ->connectTimeout(1)
        ->timeout(3)
        ->name(VOD_CLIENT_NAME);
}
function getPlayInfo($videoId) {
    return Vod::v20170321()->getPlayInfo()->client(VOD_CLIENT_NAME)
        ->withVideoId($videoId)     // 指定接口参数
        ->withAuthTimeout(3600*24)
        ->format('JSON')   // 指定返回格式
        ->request();       // 执行请求
}
try {
    initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $playInfo = getPlayInfo('<MediaId>');
    print_r($playInfo->PlayInfoList->PlayInfo);
    print_r(VOD_CLIENT_NAME);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍PHP SDK调用服务端上传接口的使用场景和调用示例。

使用场景

注意

本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端实例。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端实例。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * 获取音/视频上传地址和凭证
 * @param client 发送请求客户端
 * @return CreateUploadVideoResponse 获取音/视频上传地址和凭证响应数据
 */
function createUploadVideo($client) {
    $request = new vod\CreateUploadVideoRequest();
    $request->setTitle("Sample Title");
    $request->setFileName("videoFile.mov");
    $request->setDescription("Video Description");
    $request->setCoverURL("http://192.168.0.0/16/tps/TB1qnJ1PVXXXXXCXXXXXXXXXXXX-700-700.pn
g");
    $request->setTags("tag1,tag2");

    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $uploadInfo = createUploadVideo($client);
    var_dump($uploadInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * 刷新音/视频上传凭证
 * @param client 发送请求客户端
 * @return RefreshUploadVideoResponse 刷新音/视频上传凭证响应数据
 */
function refreshUploadVideo($client, $videoId) {
    $request = new vod\RefreshUploadVideoRequest();
    $request->setVideoId($videoId);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $refreshInfo = refreshUploadVideo($client, '<MediaId>');
    var_dump($refreshInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * 获取图片上传地址和凭证
 * @param client 发送请求客户端
 * @return CreateUploadImageResponse 获取图片上传地址和凭证响应数据
 */
function createUploadImage($client, $imageType, $imageExt) {
    $request = new vod\CreateUploadImageRequest();
    $request->setImageType($imageType);
    $request->setImageExt($imageExt);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $imageInfo = createUploadImage($client, 'cover', 'jpg');
    var_dump($imageInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * 获取辅助媒资(水印、字幕等)上传地址和凭证
 * @param client 发送请求客户端
 * @return CreateUploadAttachedMediaResponse 获取辅助媒资上传地址和凭证响应数据
 */
function createUploadAttachedMedia($client) {
    $request = new vod\CreateUploadAttachedMediaRequest();
    $request->setBusinessType("watermark");
    $request->setMediaExt("gif");
    $request->setTitle("this is a sample");

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = createUploadAttachedMedia($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * URL批量拉取上传
 * @param client 发送请求客户端
 * @return UploadMediaByURLResponse URL批量拉取上传响应数据
 */
function uploadMediaByURL($client) {
    $request = new vod\UploadMediaByURLRequest();
    $url = "http://192.168.0.0/16/***.mp4";
    $request->setUploadURLs($url);

    $uploadMetadataList = array();
    $uploadMetadata = array();
    $uploadMetadata["SourceUrl"] = $url;
    $uploadMetadata["Title"] = "upload by url sample";
    $uploadMetadataList[] = $uploadMetadata;
    $request->setUploadMetadatas(json_encode($uploadMetadataList));

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = uploadMediaByURL($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取URL上传信息

调用GetURLUploadInfos接口，完成获取URL上传信息功能。

接口参数和返回字段请参见GetURLUploadInfos。调用示例如下：

```
<?php
require_once './aliyun-php-sdk/aliyun-php-sdk-core/Config.php';   // 假定您的源码文件和aliyun
-php-sdk处于同一目录
use vod\Request\V20170321 as vod;

function initVodClient($accessKeyId, $accessKeySecret) {
    $regionId = 'cn-shanghai';  // 点播服务接入区域
    $profile = DefaultProfile::getProfile($regionId, $accessKeyId, $accessKeySecret);
    return new DefaultAcsClient($profile);
}

/**
 * 获取URL上传信息
 * @param client 发送请求客户端
 * @return GetURLUploadInfosResponse 获取URL上传信息响应数据
 */
function getURLUploadInfos($client) {
    $request = new vod\GetURLUploadInfosRequest();

    // URL列表
    $urls = array();
    $urls[] = "http://192.168.0.0/16/sample1.mp4";
    $urls[] = "http://192.168.0.0/16/sample2.mp4";

    // 对URL进行编码
    $uploadURLs = array();
    foreach ($urls as $url) {
        $uploadURLs[] = urlencode($url);
    }

    // 设置上传的URL列表，用逗号分隔
    $request->setUploadURLs(implode(",", $uploadURLs));

    // 也可以传入jobId查询
    //$request->setJobIds("jobId1****,jobId2****")

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = getURLUploadInfos($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.4. 音视频播放

本篇文档提供了PHP SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
function getPlayInfo($client, $videoId) {
    $request = new vod\GetPlayInfoRequest();
    $request->setVideoId($videoId);
    $request->setAuthTimeout(3600*24);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}
// 注意捕获异常
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $playInfo = getPlayInfo($client, 'videoId');
    print_r($playInfo->PlayInfoList->PlayInfo);
    var_dump($playInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
function getPlayAuth($client, $videoId) {
    $request = new vod\GetVideoPlayAuthRequest();
    $request->setVideoId($videoId);
    $request->setAuthInfoTimeout(3000);
    $request->setAcceptFormat('JSON');
    $response = $client->getAcsResponse($request);
    return $response;
}
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $playAuth = getPlayAuth($client, 'videoId');
    print($playAuth->PlayAuth."\n");
    print_r($playAuth->VideoMeta);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.5. 媒资管理

本篇文档提供了PHP SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，参考初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
**
 * 搜索媒资信息
 * @param client 发送请求客户端
 * @return SearchMediaResponse 搜索媒资信息响应数据
 */
function searchMedia($client) {
    $request = new vod\SearchMediaRequest();
    $request->setFields("Title,CoverURL,Status");
    $request->setMatch("Status in ('Normal','Checking') and CreationTime = ('2018-07-01T08:
00:00Z','2018-08-01T08:00:00Z')");
    $request->setPageNo(1);
    $request->setPageSize(10);
    $request->setSearchType("video");
    $request->setSortBy("CreationTime:Desc");

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = searchMedia($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
/**
 * 获取视频信息
 * @param client 发送请求客户端
 * @return GetVideoInfoResponse 获取视频信息响应数据
 */
function getVideoInfo($client, $videoId) {
    $request = new vod\GetVideoInfoRequest();
    $request->setVideoId($videoId);
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $videoInfo = getVideoInfo($client, 'Your videoId');
    var_dump($videoInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
/**
 * 批量获取视频信息函数
 * @param client 发送请求客户端
 * @return GetVideoInfosResponse 获取视频信息响应数据
 */
function getVideoInfos($client) {
    $request = new vod\GetVideoInfosRequest();
    $request->setVideoIds("e67e761ec04342cd9ca5149c7488****,b19439abb9a94374b7f4d45f6988***
*");

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = getVideoInfos($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
/**
 * 修改视频信息
 * @param client 发送请求客户端
 * @return UpdateVideoInfoResponse 修改视频信息响应数据
 */
function updateVideoInfo($client, $videoId) {
    $request = new vod\UpdateVideoInfoRequest();
    $request->setVideoId($videoId);
    $request->setTitle('New Title');   // 更改视频标题
    $request->setDescription('New Description');    // 更改视频描述
    $request->setCoverURL('http://192.168.0.0/16/tps/TB1qnJ1PVXXXXXCXXXXXXXXXXXX-700-700.pn
g');  // 更改视频封面
    $request->setTags('tag1,tag2');    // 更改视频标签，多个用逗号分隔
    $request->setCateId(0);       // 更改视频分类(可在点播控制台·全局设置·分类管理里查看分类ID：ht
tps://192.168.0.0/16/#/vod/settings/category)
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $videoInfo = updateVideoInfo($client, 'Your videoId');
    var_dump($videoInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
/**
 * 批量修改视频信息
 * @param client 发送请求客户端
 * @return UpdateVideoInfosResponse 批量修改视频信息响应数据
*/
function updateVideoInfos($client) {
    $request = new vod\UpdateVideoInfosRequest();

    $updateContentArray = array();
    $updateContent1 = array();
    $updateContent1["VideoId"] = "e67e761ec04342cd9ca5149c7485****";
    $updateContent1["Title"] = "new Title1";
    $updateContentArray[] = $updateContent1;

    $updateContent2 = array();
    $updateContent2["VideoId"] = "b19439abb9a94374b7f4d45f6985****";
    $updateContent2["Title"] = "new Title2";
    $updateContentArray[] = $updateContent2;

    $request->setUpdateContent(json_encode($updateContentArray));

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = updateVideoInfos($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
/**
 * 删除视频
 * @param client 发送请求客户端
 * @return DeleteVideoResponse 删除视频响应数据
 */
function deleteVideos($client, $videoIds) {
    $request = new vod\DeleteVideoRequest();
    $request->setVideoIds($videoIds);   // 支持批量删除视频；videoIds为传入的视频ID列表，多个用逗号分隔
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $delInfo = deleteVideos($client, 'vid1,vid2');
    var_dump($delInfo);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取源文件信息（含原片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
/**
 * 获取源文件信息
 * @param client 发送请求客户端
 * @return GetMezzanineInfoResponse 获取源文件信息响应数据
 */
function getMezzanineInfo($client, $videoId) {
    $request = new vod\GetMezzanineInfoRequest();
    $request->setVideoId($videoId);
    $request->setAuthTimeout(3600*5);   // 原片下载地址过期时间，单位：秒，默认为3600秒
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $mezzanine = getMezzanineInfo($client, 'Your videoId');
    var_dump($mezzanine);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

```
/**
 * 获取视频列表
 * @param client 发送请求客户端
 * @return GetVideoListResponse 获取视频列表响应数据
 */
function getVideoList($client) {
    $request = new vod\GetVideoListRequest();

    // 示例：分别取一个月前、当前时间的UTC时间作为筛选视频列表的起止时间
    $localTimeZone = date_default_timezone_get();
    date_default_timezone_set('UTC');
    $utcNow = gmdate('Y-m-d\TH:i:s\Z');
    $utcMonthAgo = gmdate('Y-m-d\TH:i:s\Z', time() - 30*86400);
    date_default_timezone_set($localTimeZone);
    $request->setStartTime($utcMonthAgo);   // 视频创建的起始时间，为UTC格式
    $request->setEndTime($utcNow);          // 视频创建的结束时间，为UTC格式

    #$request->setStatus('Uploading,Normal,Transcoding');  // 视频状态，默认获取所有状态的视频
，多个用逗号分隔
    #$request->setCateId(0);               // 按分类进行筛选
    $request->setPageNo(1);
    $request->setPageSize(20);
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $videoList = getVideoList($client);
    var_dump($videoList);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
/**
 * 删除媒体流函数
 * @param client 发送请求客户端
 * @return DeleteMezzaninesResponse 删除媒体流响应数据
 */
function deleteStream($client, $videoId, $jobIds) {
    $request = new vod\DeleteStreamRequest();
    $request->setVideoId($videoId);
    $request->setJobIds($jobIds);   // 媒体流转码的作业ID列表，多个用逗号分隔；JobId可通过获取播放地址接口(GetPlayInfo)获取到。
    $request->setAcceptFormat('JSON');

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $delStream = deleteStream($client, 'videoId', 'jobId1,jobId2');
    var_dump($delStream);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
/**
 * 批量删除源文件函数
 * @param client 发送请求客户端
 * @return DeleteMezzaninesResponse 批量删除源文件响应数据
*/
function deleteMezzanines($client) {
    $request = new vod\DeleteMezzaninesRequest();
    $request->setVideoIds("b19439abb9a94374b7f4d45f6656****");
    $request->setForce(false);

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = deleteMezzanines($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
/**
 * 批量更新图片信息函数
 * @param client 发送请求客户端
 * @return UpdateImageInfosResponse 批量更新图片信息响应数据
 */
function updateImageInfos($client) {
    $request = new vod\UpdateImageInfosRequest();

    $updateContentArray = array();
    $updateContent1 = array();
    $updateContent1["ImageId"] = "7e66f99e1cb3419982275398159****";
    $updateContent1["Title"] = "new Title1";
    $updateContentArray[] = $updateContent1;

    $updateContent2 = array();
    $updateContent2["ImageId"] = "d2171a5656454c49a4f11a544c59****";
    $updateContent2["Title"] = "new Title2";
    $updateContentArray[] = $updateContent2;

    $request->setUpdateContent(json_encode($updateContentArray));

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = updateImageInfos($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
/**
 * 获取图片信息函数
 * @param client 发送请求客户端
 * @return GetImageInfoResponse 获取图片信息响应数据
 */
function getImageInfo($client) {
    $request = new vod\GetImageInfoRequest();
    $request->setImageId("7e66f99e1cb341998227539812b4****");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = getImageInfo($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
/**
 * 删除图片函数
 * @param client 发送请求客户端
 * @return DeleteImageResponse 删除图片响应数据
 */
function deleteImage($client) {
    $request = new vod\DeleteImageRequest();
    //根据ImageURL删除图片文件
    $request->setDeleteImageType("ImageURL");
    $request->setImageURLs("http://192.168.0.0/16/cover.jpg");

    //根据ImageId删除图片文件
    //$request->setDeleteImageType("ImageId");
    //$request->setImageIds("ImageId1,ImageId2");

    //根据VideoId删除指定ImageType的图片文件
    //$request->setDeleteImageType("VideoId");
    //$request->setVideoId("VideoId");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $result = deleteImage($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.6. 媒资分类

本文为您提供了PHP SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
function addCategory($client, $cateName, $parentId=-1) {
    $request = new vod\AddCategoryRequest();
    $request->setCateName($cateName);
    $request->setParentId($parentId);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $addRes = addCategory($client, 'Category Name');
    var_dump($addRes);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
function updateCategory($client, $cateId, $cateName) {
    $request = new vod\UpdateCategoryRequest();
    $request->setCateId($cateId);
    $request->setCateName($cateName);   // 分类名称
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $updateRes = updateCategory($client, '<Category ID>', 'New Category Name');
    var_dump($updateRes);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
// 删除视频分类，同时会删除其下级分类（包括二级分类和三级分类），请慎重操作
function deleteCategory($client, $cateId) {
    $request = new vod\DeleteCategoryRequest();
    $request->setCateId($cateId);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $delRes = deleteCategory($client, '<Category ID>');
    var_dump($delRes);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
// 获取指定的分类信息，及其子分类（即下一级分类）的列表
function getCategories($client, $cateId=-1, $pageNo=1, $pageSize=10) {
    $request = new vod\GetCategoriesRequest();
    $request->setCateId($cateId);   // 分类ID，默认为根节点分类ID即-1
    $request->setPageNo($pageNo);
    $request->setPageSize($pageSize);
    $request->setAcceptFormat('JSON');
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $getRes = getCategories($client, '<Category Id>');
    var_dump($getRes);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.7. 媒体处理

本文提供了PHP SDK媒体处理模块相关功能的API调用示例，包含提交转码作业、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业

调用SubmitTranscodeJobs接口，完成提交转码作业功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

```
use Kms\Request\V20160120 as kms;

/**
 * 构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；如不使用可忽略
 * 需要替换的水印信息对应水印ID必须是关联在指定的转码模板组ID上(即TranscodeTemplateId)
 * 不支持通过媒体处理接口去增加一个没有关联上的水印
 */
function buildOverrideParams() {
    $overrideParams = array();

    //以替换水印为例
    $watermarks = array();

    //替换图片水印文件
    $watermarks1 = array();
    $watermarks1["WatermarkId"] = "2ea587477c5a1bc8b52****";
    //FileURL示例：https://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermarks/02A1B
22DF25D46C3C7****-6-2.png
    $watermarks1["FileUrl"] = "<your File URL>";
    $watermarks[] = $watermarks1;

    //替换文字水印
    $watermarks2 = array();
    $watermarks2["WatermarkId"] = "d297ba31ac5242d2052****";
    $watermarks2["Content"] = "用户ID：6****";
    $watermarks[] = $watermarks2;

    $overrideParams["Watermarks"] = $watermarks;
    return json_encode($overrideParams);
}

/**
 * 生成HLS标准加密参数的可选配置，如不使用可忽略
 * 此时依赖KMS服务，需要安装KMS依赖包 aliyun-php-sdk-kms
 * 生成密钥API参考：https://192.168.0.0/16/document_detail/28948****.html
 */
function buildEncryptConfig($client) {
    try {
        //随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥，
        //视频标准加密只需要传递密文密钥即可
        $request = new kms\GenerateDataKeyRequest();
        $request->setKeyId('<serviceKey>');
        $request->setKeySpec("AES_128");
        $response = $client->getAcsResponse($request);

       $encryptConfig = array();
        # 解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥
都不一样；注意您需要自己部署解密服务
        # DecryptKeyUri示例：http://192.168.0.0/16/decrypt?Ciphertext=" + $response->getCiph
ertextBlob()
        $encryptConfig["DecryptKeyUri"] = "<your DecryptKeyUri>";
        //密钥服务的类型，目前只支持KMS
        $encryptConfig["KeyServiceType"] = "KMS";
        # Ciphertext作为解密接口的参数名称，可自定义，此处只作为参考
```

```
        $encryptConfig["CipherText"] = $response->getCiphertextBlob();

        return json_encode($encryptConfig);
    } catch (Exception $e) {
        print $e->getMessage()."\n";
        return null;
    }
}

/**
 * 提交媒体处理作业
 */
function submitTranscodeJobs($client) {
    $request = new vod\SubmitTranscodeJobsRequest();
    //需要转码的视频ID
    $request->setVideoId("6893fca9814640c8821efa523e52****");
    //转码模板ID
    $request->setTemplateGroupId("44f915b63a2375a6121533c6b252****");
    //构建需要替换的水印参数(只有需要替换水印相关信息才需要构建)
    $request->setOverrideParams(buildOverrideParams());
    //HLS标准加密配置(只有标准加密才需要传递)
    $request->setEncryptConfig(buildEncryptConfig($client));

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = submitTranscodeJobs($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明**
>
> 创建截图模板详细请参考截图模板。

```
/**
 * 构建雪碧图截图配置
 * @return
 */
function buildSnapshotTemplateConfig() {
    $spriteSnapshotConfig = array();
    $spriteSnapshotConfig["CellWidth"] = "120";
    $spriteSnapshotConfig["CellHeight"] = "68";
    $spriteSnapshotConfig["Columns"] = "3";
    $spriteSnapshotConfig["Lines"] = "10";
    $spriteSnapshotConfig["Padding"] = "20";
    $spriteSnapshotConfig["Margin"] = "50";

    //保留雪碧图原始图
    $spriteSnapshotConfig["KeepCellPic"] = "keep";
    $spriteSnapshotConfig["Color"] = "tomato";

    return json_encode($spriteSnapshotConfig);
}


/**
 * 提交媒体截图处理作业调用函数
 */
function submitSnapshotJob($client) {
    $request = new vod\SubmitSnapshotJobRequest();
    //需要截图的视频ID(推荐传递截图模板ID)
    $request->setVideoId("6893fca9814640c8821efa523e52****");
    //截图模板ID
    $request->setSnapshotTemplateId("44f915b63a2375a6121533c6b252****");

    //如果设置了SnapshotTemplateId，会忽略下面参数
    $request->setCount(50);
    $request->setSpecifiedOffsetTime(0);
    $request->setInterval(1);
    $request->setWidth("200");
    $request->setHeight("200");
    $request->setSpriteSnapshotConfig(buildSnapshotTemplateConfig());

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = submitSnapshotJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询截图数据

调用listSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见listSnapshots。调用示例如下：

```
/**
 * 查询截图数据
 */
function listSnapshots($client) {
    $request = new vod\ListSnapshotsRequest();
    //视频ID
    $request->setVideoId("6893fca9814640c8821efa523e52****");
    ///截图类型
    $request->setSnapshotType("CoverSnapshot");
    // 翻页参数
    $request->setPageNo("1");
    $request->setPageSize("20");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = listSnapshots($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
/**
 * 导播台视频预处理
 */
function submitPreprocessJobs($client) {
    $request = new vod\SubmitPreprocessJobsRequest();
    //视频ID
    $request->setVideoId("6893fca9814640c8821efa523e52****");
    ///类型
    $request->setPreprocessType("PreprocessType");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = submitPreprocessJobs($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.8. 视频水印

本篇文档提供了PHP SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

```
/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
function buildImageWatermarkConfig() {
    $watermarkConfig = array();
    //水印的横向偏移距离
    $watermarkConfig["Dx"] = "8";
    //水印的纵向偏移距离
    $watermarkConfig["Dy"] = "8";
    //水印显示的宽
    $watermarkConfig["Width"] = "55";
    //水印显示的高
    $watermarkConfig["Height"] = "55";
```

```
    //水印显示的相对位置(左上、右上、左下、右下)
    $watermarkConfig["ReferPos"] = "BottomRight";

    //水印显示的时间线(开始显示和结束显示时间)
    $timeline = array();
    //水印开始显示时间
    $timeline["Start"] = "2";
    //水印结束显示时间
    $timeline["Duration"] = "ToEND";
    $watermarkConfig["Timeline"] = $timeline;

    return json_encode($watermarkConfig);
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
function buildTextWatermarkConfig() {
    $watermarkConfig = array();
    //文字水印显示的内容
    $watermarkConfig["Content"] = "testwatermark";
    //文字水印的字体名称
    $watermarkConfig["FontName"] = "SimSun";
    //文字水印的字体大小
    $watermarkConfig["FontSize"] = "25";
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    $watermarkConfig["FontColor"] = "Black";
    //文字水印的透明度
    $watermarkConfig["FontAlpha"] = "0.2";
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    $watermarkConfig["BorderColor"] = "White";
    //文字水印的描边宽度
    $watermarkConfig["BorderWidth"] = "1";
    //文字水印距离视频画面上边的偏移距离
    $watermarkConfig["Top"] = "20";
    //文字水印距离视频画面左边的偏移距离
    $watermarkConfig["Left"] = "15";

    return json_encode($watermarkConfig);
}

/**
 * 添加水印配置信息函数
 */
function addWatermark($client) {
    $request = new vod\AddWatermarkRequest();
    //水印名称
    $request->setName("addwatermark");

    //水印文件在oss的URL
    //图片水印必传图片文件的oss文件地址，水印文件必须和视频在同一个区域，例如:华东2视频，水印文件必须存放在华东2
    $request->setFileUrl("http://192.168.0.0/16/watermark/test.png");
```

```
    //水印配置数据
    //图片水印的位置配置数据
    $request->setWatermarkConfig(buildImageWatermarkConfig());
    //文字水印的位置配置数据
    //$request->setWatermarkConfig(buildTextWatermarkConfig());

    //文字水印:Text; 图片水印:Image
    $request->setType("Image");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = addWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

```
/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
function buildImageWatermarkConfig() {
    $watermarkConfig = array();
    //水印的横向偏移距离
    $watermarkConfig["Dx"] = "10";
    //水印的纵向偏移距离
    $watermarkConfig["Dy"] = "10";
    //水印显示的宽
    $watermarkConfig["Width"] = "66";
    //水印显示的高
    $watermarkConfig["Height"] = "66";
    //水印显示的相对位置(左上、右上、左下、右下)
    $watermarkConfig["ReferPos"] = "BottomRight";

    //水印显示的时间线(开始显示和结束显示时间)
    $timeline = array();
    //水印开始显示时间
    $timeline["Start"] = "2";
    //水印结束显示时间
    $timeline["Duration"] = "ToEND";
```

```
    $watermarkConfig["Timeline"] = $timeline;

    return json_encode($watermarkConfig);
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 * @return
 */
function buildTextWatermarkConfig() {
    $watermarkConfig = array();
    //文字水印显示的内容
    $watermarkConfig["Content"] = "testwatermark";
    //文字水印的字体名称
    $watermarkConfig["FontName"] = "SimSun";
    //文字水印的字体大小
    $watermarkConfig["FontSize"] = "40";
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    $watermarkConfig["FontColor"] = "Black";
    //文字水印的透明度
    $watermarkConfig["FontAlpha"] = "0.2";
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    $watermarkConfig["BorderColor"] = "White";
    //文字水印的描边宽度
    $watermarkConfig["BorderWidth"] = "2";
    //文字水印距离视频画面上边的偏移距离
    $watermarkConfig["Top"] = "20";
    //文字水印距离视频画面左边的偏移距离
    $watermarkConfig["Left"] = "15";

    return json_encode($watermarkConfig);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = updateWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
/**
 * 删除水印配置信息函数
 */
function deleteWatermark($client) {
    $request = new vod\DeleteWatermarkRequest();
    //设置水印ID
    $request->setWatermarkId("e7d983370268092176588a2c452****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = deleteWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
/**
 * 查询水印配置信息列表函数
 */
function listWatermark($client) {
    $request = new vod\ListWatermarkRequest();
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = listWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
/**
 * 查询单个水印配置信息函数
 */
function getWatermark($client) {
    $request = new vod\GetWatermarkRequest();
    //需要查询水印信息的水印ID
    $request->setWatermarkId("bfc084674fb64486b6e5bace3052****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
/**
 * 设置默认水印配置信息函数
 */
function setDefaultWatermark($client) {
    $request = new vod\SetDefaultWatermarkRequest();
    //设置默认的水印ID
    $request->setWatermarkId("bfc084674fb64486b6e5bace3052****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 * @param args
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = setDefaultWatermark($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.9. 转码模板

本篇文档提供了PHP SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
/**
 * 构建转码模板配置数据
 * @return
 */
function buildTranscodeTemplateList() {
    $transcodeTemplateList = array();
    $transcodeTemplate = array();

    //视频流转码配置
    $video = array();
    $video["Width"] = 640;
    $video["Bitrate"] = 400;
    $video["Fps"] = 25;
    $video["Remove"] = false;
    $video["Codec"] = "H.264";
```

```
    $video["Gop"] = 250;
    $transcodeTemplate["Video"] = $video;

    //音频流转码配置
    $audio = array();
    $audio["Codec"] = "AAC";
    $audio["Bitrate"] = 64;
    $audio["Channels"] = 2;
    $audio["Samplerate"] = 32000;
    $transcodeTemplate["Audio"] = $audio;

    //封装容器
    $container = array();
    $container["Format"] = "mp4";
    $transcodeTemplate["Container"] = $container;

    //条件转码配置
    $transconfig = array();
    $transconfig["IsCheckReso"] = false;
    $transconfig["IsCheckResoFail"] = false;
    $transconfig["IsCheckVideoBitrate"] = false;
    $transconfig["IsCheckVideoBitrateFail"] = false;
    $transconfig["IsCheckAudioBitrate"] = false;
    $transconfig["IsCheckAudioBitrateFail"] = false;
    $transcodeTemplate["TransConfig"] = $transconfig;

    //加密配置(只支持m3u8)
    //$encryptSetting= array();
    //$encryptSetting["EncryptType"] = "Private";
    //$transcodeTemplate["EncryptSetting"] = $encryptSetting;

    //清晰度
    $transcodeTemplate["Definition"] = "LD";

    //模板名称
    $transcodeTemplate["TemplateName"] = "testtemplate";

    //水印ID(多水印关联)
    $watermarkIdList= array();
    $watermarkIdList[] = "263261bdc1ff65782f8995c6dd52****";
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    $watermarkIdList[] = "USER_DEFAULT_WATERMARK";
    $transcodeTemplate["WatermarkIds"] = $watermarkIdList;
    $transcodeTemplateList[] = $transcodeTemplate;

    return json_encode($transcodeTemplateList);
}

/**
 * 添加转码模板组配置
 */
function addTranscodeTemplateGroup($client) {
    $request = new vod\AddTranscodeTemplateGroupRequest();
    // 转码模板组名称
```

```
    $request->setName("grouptest");
    // 转码模板组信息
    $request->setTranscodeTemplateList(buildTranscodeTemplateList());

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = addTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
/**
 * 构建转码模板配置数据
 * @return
 */
function buildTranscodeTemplateList() {
    $transcodeTemplateList = array();
    $transcodeTemplate = array();

    //视频流转码配置
    $video = array();
    $video["Width"] = 960;
    $video["Bitrate"] = 900;
    $video["Fps"] = 25;
    $video["Remove"] = false;
    $video["Codec"] = "H.264";
    $video["Gop"] = 250;
    $transcodeTemplate["Video"] = $video;

    //音频流转码配置
    $audio = array();
    $audio["Codec"] = "AAC";
    $audio["Bitrate"] = 96;
    $audio["Channels"] = 2;
    $audio["Samplerate"] = 32000;
    $transcodeTemplate["Audio"] = $audio;

    //封装容器
    $container = array();
    $container["Format"] = "mp4";
    $transcodeTemplate["Container"] = $container;
```

```
    //条件转码配置
    $transconfig = array();
    $transconfig["IsCheckReso"] = false;
    $transconfig["IsCheckResoFail"] = false;
    $transconfig["IsCheckVideoBitrate"] = false;
    $transconfig["IsCheckVideoBitrateFail"] = false;
    $transconfig["IsCheckAudioBitrate"] = false;
    $transconfig["IsCheckAudioBitrateFail"] = false;
    $transcodeTemplate["TransConfig"] = $transconfig;

    //加密配置(只支持m3u8)
    //$encryptSetting= array();
    //$encryptSetting["EncryptType"] = "Private";
    //$transcodeTemplate["EncryptSetting"] = $encryptSetting;

    //清晰度
    $transcodeTemplate["Definition"] = "SD";

    //模板名称
    $transcodeTemplate["TemplateName"] = "testtemplate";

    //模板ID
    $transcodeTemplate["TranscodeTemplateId"] = "214a67fab9fdf920f486faa7752****";

    //水印ID(多水印关联)
    $watermarkIdList= array();
    $watermarkIdList[] = "263261bdc1ff65782f8995c6dd52****";
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    $watermarkIdList[] = "USER_DEFAULT_WATERMARK";
    $transcodeTemplate["WatermarkIds"] = $watermarkIdList;
    $transcodeTemplateList[] = $transcodeTemplate;

    return json_encode($transcodeTemplateList);
}

/**
 * 修改转码模板组配置
 */
function updateTranscodeTemplateGroup($client) {
    $request = new vod\UpdateTranscodeTemplateGroupRequest();
    // 转码模板组名称
    $request->setName("grouptest1");
    // 转码模板组ID
    $request->setTranscodeTemplateGroupId("014a67fab9fdf920f486faa7752****");
    // 转码模板组信息
    $request->setTranscodeTemplateList(buildTranscodeTemplateList());

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
```

```
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = updateTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
/**
 * 查询转码模板组列表
 */
function listTranscodeTemplateGroup($client) {
    $request = new vod\ListTranscodeTemplateGroupRequest();
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = listTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
/**
 * 查询单个转码模板组
 */
function getTranscodeTemplateGroup($client) {
    $request = new vod\GetTranscodeTemplateGroupRequest();
    // 转码模板组ID
    $request->setTranscodeTemplateGroupId("014a67fab9fdf920f486faa7752****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = getTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
/**
 * 设置默认转码模板组
 */
function setDefaultTranscodeTemplateGroup($client) {
    $request = new vod\SetDefaultTranscodeTemplateGroupRequest();
    // 转码模板组ID
    $request->setTranscodeTemplateGroupId("014a67fab9fdf920f486faa7752****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = setDefaultTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
/**
 * 删除转码模板组配置
 */
function deleteTranscodeTemplateGroup($client) {
    $request = new vod\DeleteTranscodeTemplateGroupRequest();
    // 转码模板组ID
    $request->setTranscodeTemplateGroupId("014a67fab9fdf920f486faa77352****");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = deleteTranscodeTemplateGroup($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.10. 截图模板

本篇文档提供了PHP SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
 * （以下代码示例为雪碧图完整配置）
 * @return
 */
function buildSnapshotTemplateConfig() {
    $templateConfig = array();

    //普通截图配置（与雪碧图原始图配置共用该配置信息）
    $snapshotConfig = array();
    $snapshotConfig["Count"] = "50";
    $snapshotConfig["Interval"] = "1";
    $snapshotConfig["SpecifiedOffsetTime"] = "0";
```

```
    $snapshotConfig["Width"] = "200";
    $snapshotConfig["Height"] = "200";
    $snapshotConfig["FrameType"] = "normal";
    $templateConfig["SnapshotConfig"] = $snapshotConfig;

    //雪碧图配置(雪碧图配置必须是建立在普通截图配置之上)
    $spriteSnapshotConfig = array();
    $spriteSnapshotConfig["CellWidth"] = "120";
    $spriteSnapshotConfig["CellHeight"] = "68";
    $spriteSnapshotConfig["Columns"] = "3";
    $spriteSnapshotConfig["Lines"] = "10";
    $spriteSnapshotConfig["Padding"] = "20";
    $spriteSnapshotConfig["Margin"] = "50";
    $spriteSnapshotConfig["KeepCellPic"] = "keep";
    $spriteSnapshotConfig["Color"] = "tomato";
    $templateConfig["SpriteSnapshotConfig"] = $spriteSnapshotConfig;

    //截图类型(存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot")
    $templateConfig["SnapshotType"] = "SpriteSnapshot";

    return json_encode($templateConfig);
}

/**
 * 添加截图模板函数
 */
function addSnapshotVodTemplate($client) {
    $request = new vod\AddVodTemplateRequest();
    //模板名称
    $request->setName("截图模板添加测试");
    //模板类型，固定值为Snapshot
    $request->setTemplateType("Snapshot");
    //截图模板配置数据生成
    $request->setTemplateConfig(buildSnapshotTemplateConfig());

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = addSnapshotVodTemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
 * （以下代码为普通截图完整配置）
 * @return
 */
function buildSnapshotTemplateConfig() {
    $templateConfig = array();

    //普通截图配置（与雪碧图原始图配置共用该配置信息）
    $snapshotConfig = array();
    $snapshotConfig["Count"] = "50";
    $snapshotConfig["Interval"] = "1";
    $snapshotConfig["SpecifiedOffsetTime"] = "0";
    $snapshotConfig["Width"] = "200";
    $snapshotConfig["Height"] = "200";
    $snapshotConfig["FrameType"] = "normal";
    $templateConfig["SnapshotConfig"] = $snapshotConfig;

    //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
    $templateConfig["SnapshotType"] = "NormalSnapshot";

    return json_encode($templateConfig);
}

/**
 * 修改截图模板函数
 */
function updateSnapshotVodTemplate($client) {
    $request = new vod\UpdateVodTemplateRequest();
    //设置要修改的模板ID
    $request->setVodTemplateId("6e9835ce8896aa3ace027c048352****");
    //模板名称
    $request->setName("截图模板修改测试");
    //截图模板配置数据生成
    $request->setTemplateConfig(buildSnapshotTemplateConfig());

    return $client->getAcsResponse($request);
}


/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = updateSnapshotVodTemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
/**
 * 删除截图模板函数
 */
function deleteSnapshotVodTemplate($client) {
    $request = new vod\DeleteVodTemplateRequest();
    //设置要删除的模板ID
    $request->setVodTemplateId("6e9835ce8896aa3ace027c048352****");

    return $client->getAcsResponse($request);
}


/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = deleteSnapshotVodTemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
/**
 * 查询截图模板列表
 */
function listSnapshotVodTemplate($client) {
    $request = new vod\ListVodTemplateRequest();
    //模板类型，固定值为Snapshot
    $request->setTemplateType("Snapshot");

    return $client->getAcsResponse($request);
}


/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = listSnapshotVodTemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
/**
 * 查询截图模板函数
 */
function getSnapshotVodTemplate($client) {
    $request = new vod\GetVodTemplateRequest();
    //设置要查询的模板ID
    $request->setVodTemplateId("6e9835ce8896aa3ace027c048352****");

    return $client->getAcsResponse($request);
}


/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getSnapshotVodTemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.11. 视频剪辑

本篇文档提供了PHP SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

> **说明**
>
> 通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
function buildMediaMetadata() {
    $mediaMetadata = array();
    // Produce Media Title
    $mediaMetadata['Title'] = 'Title';
    // Produce Media Description
    $mediaMetadata['Description'] = 'Description';
    // Produce Media UserDefined Cover URL
    $mediaMetadata['CoverURL'] = 'http://192.168.0.0/16/media/cover/mediaid.jpg';
    // Produce Media Category ID
    $mediaMetadata['CateId'] = null;
    // Produce Media Category Name
    $mediaMetadata['Tags'] = 'Tag1,Tag2,Test';

    return json_encode($mediaMetadata);
}

function buildProduceConfig() {
    $produceConfig = array();
    /*
     The produce process can generate media mezzanine file. You can use the mezzanine file
to transcode other media files, just like the transcode process after file upload finished.
This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
     1. Not required
     2. Use default transcode template group id when empty
     */
    $produceConfig['TemplateGroupId'] = null;
    return json_encode($produceConfig);
}

function buildTimeline() {
    $timeline = array();

    // Video Track
    $videoTracks = array();
```

```
    $videoTrack = array();

    // Video Track Clicps
    $videoTrackClips = array();
    $videoTrackClip1 = array();
    $videoTrackClip1['MediaId'] = '6893fca9814640c8821efa52352****';
    $videoTrackClips[] = $videoTrackClip1;

    $videoTrackClip2 = array();
    $videoTrackClip2['MediaId'] = '070bbc13d8294e35b36c3e7ab52****';
    $videoTrackClips[] = $videoTrackClip2;

    $videoTrack['VideoTrackClips'] = $videoTrackClips;
    $videoTracks[] = $videoTrack;
    $timeline['VideoTracks'] = $videoTracks;

    return json_encode($timeline);
}

try {
     $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\ProduceEditingProjectVideoRequest();
    // Set Produce Media Metadata
    $request->setMediaMetadata(buildMediaMetadata());
    // Set Produce Configuration
    $request->setProduceConfig(buildProduceConfig());
    // Build Editing Project Timeline
    $request->setTimeline(buildTimeline());

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
function buildMediaMetadata() {
    $mediaMetadata = array();
    // Produce Media Title
    $mediaMetadata['Title'] = 'Title';
    // Produce Media Description
    $mediaMetadata['Description'] = 'Description';
    // Produce Media UserDefined Cover URL
    $mediaMetadata['CoverURL'] = 'http://192.168.0.0/16/media/cover/mediaid.jpg';
    // Produce Media Category ID
    $mediaMetadata['CateId'] = null;
    // Produce Media Category Name
    $mediaMetadata['Tags'] = 'Tag1,Tag2,Test';

    return json_encode($mediaMetadata);
}

function buildProduceConfig() {
    $produceConfig = array();
    /*
     The produce process can generate media mezzanine file. You can use the mezzanine file
to transcode other media files, just like the transcode process after file upload finished.
This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
     1. Not required
     2. Use default transcode template group id when empty
     */
    $produceConfig['TemplateGroupId'] = null;
    return json_encode($produceConfig);
}

try {
     $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\ProduceEditingProjectVideoRequest();
    // Set Editing Project ID need to produce
    $request->setProjectId('7794d3757fe548e0abacb5dd1052****');
    // Set Produce Media Metadata
    $request->setMediaMetadata(buildMediaMetadata());
    // Set Produce Configuration
    $request->setProduceConfig(buildProduceConfig());

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
function buildTimeline() {
    $timeline = array();

    // Video Track
    $videoTracks = array();
    $videoTrack = array();

    // Video Track Clicps
    $videoTrackClips = array();
    $videoTrackClip1 = array();
    $videoTrackClip1['MediaId'] = '6893fca9814640c8821efa52352****';
    $videoTrackClips[] = $videoTrackClip1;

    $videoTrackClip2 = array();
    $videoTrackClip2['MediaId'] = '070bbc13d8294e35b36c3e7ab52****';
    $videoTrackClips[] = $videoTrackClip2;

    $videoTrack['VideoTrackClips'] = $videoTrackClips;
    $videoTracks[] = $videoTrack;
    $timeline['VideoTracks'] = $videoTracks;

    return json_encode($timeline);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\AddEditingProjectRequest();
    // Build Editing Project Timeline
    $request->setTimeline(buildTimeline());
   // Set Editing Project Title
    $request->setTitle('Editing Project Title');
    // Set Editing Project Description
    $request->setDescription('Editing Project Description');
    // Set Editing Project Cover URL
    $request->setCoverURL('http://192.168.0.0/16/editingproject/cover/projectid.jpg');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
function buildTimeline() {
    $timeline = array();

    // Video Track
    $videoTracks = array();
    $videoTrack = array();

    // Video Track Clicps
    $videoTrackClips = array();
    $videoTrackClip1 = array();
    $videoTrackClip1['MediaId'] = '6893fca9814640c8821efa523xxxxxx';
    $videoTrackClips[] = $videoTrackClip1;

    $videoTrackClip2 = array();
    $videoTrackClip2['MediaId'] = '070bbc13d8294e35b36c3e7ab52****';
    $videoTrackClips[] = $videoTrackClip2;

    $videoTrack['VideoTrackClips'] = $videoTrackClips;
    $videoTracks[] = $videoTrack;
    $timeline['VideoTracks'] = $videoTracks;

    return json_encode($timeline);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\UpdateEditingProjectRequest();
    // Set Editing Project ID
    $request->setProjectId('e7c5d44105d3487db23703be152****');
    // Build Editing Project Timeline
    $request->setTimeline(buildTimeline());
    // Set Editing Project Title
    $request->setTitle('Editing Project Title2');
    // Set Editing Project Description
    $request->setDescription('Editing Project Description2');
    // Set Editing Project Cover URL
    $request->setCoverURL('http://192.168.0.0/16/editingproject/cover/projectid.jpg');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\DeleteEditingProjectRequest();
    // Use comma to split Multi Editing Project IDs
    $request->setProjectIds('e7c5d44105d3487db23703be1352****,12c5d44105d3487db23703be1352*
***');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\GetEditingProjectRequest();
    // Set Editing Project ID
    $request->setProjectId('7794d3757fe548e0abacb5dd152****');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\SearchEditingProjectRequest();
    $request->setTitle('Title Keywords');
    $request->setStartTime('2017-01-11T12:00:00Z');
    $request->setEndTime('2017-01-12T12:00:00Z');
    $request->setPageSize(10);
    $request->setPageNo(1);

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
// 设置云剪辑工程素材
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\SetEditingProjectMaterialsRequest();
    // Set Editing Project ID
    $request->setProjectId('7794d3757fe548e0abacb5dd1052****');
    // Set Editing Project Material IDs, use comma to split
    $request->setMaterialIds('materialId1,materialId2');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');

    $request = new vod\GetEditingProjectMaterialsRequest();
    // Set Editing Project ID
    $request->setProjectId('7794d3757fe548e0abacb5dd1052****');
    // Set Editing Project Type
    $request->setType('video');

    // Get result
    $result = $client->getAcsResponse($request);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.12. 媒体审核

本篇文档提供了PHP SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
/**
 * 提交智能审核作业
 */
function submitAIMediaAuditJob($client) {
    $request = new vod\SubmitAIMediaAuditJobRequest();
    // 设置视频ID
    $request->setMediaId("dc063078c1d845139e2a5bd8f52****");

    // 返回结果
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = submitAIMediaAuditJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
/**
 * 查询智能审核作业
 */
function getAIMediaAuditJob($client) {
    $request = new vod\GetAIMediaAuditJobRequest();
    // 设置作业ID
    $request->setJobId("f980db7d82e644478206a4a52152****");

    // 返回结果
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getAIMediaAuditJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
/**
 * 获取智能审核结果摘要
 */
function getMediaAuditResult($client) {
    $request = new vod\GetMediaAuditResultRequest();
    // 设置视频ID
    $request->setMediaId("dc063078c1d845139e2a5bd8ff52****");

    // 返回结果
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getMediaAuditResult($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
/**
 * 获取智能审核结果详情
 */
function getMediaAuditResultDetail($client) {
    $request = new vod\GetMediaAuditResultDetailRequest();
    // 设置视频ID
    $request->setMediaId("dc063078c1d845139e2a5bd8ff52****");
    // 设置翻页
    $request->setPageNo(1);

    // 返回结果
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getMediaAuditResultDetail($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
/**
 * 获取智能审核结果时间线
 */
function getMediaAuditResultTimeline($client) {
    $request = new vod\GetMediaAuditResultTimelineRequest();
    // 设置视频ID
    $request->setMediaId("dc063078c1d845139e2a5bd8ff52****");

    // 返回结果
    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getMediaAuditResultTimeline($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
/**
 * 构建审核内容
 */
function buildAuditContent() {
    $auditContent = array();

    $auditContent1 = array();
    $auditContent1["VideoId"] = "070bbc13d8294e35b36c3e7ab452****"; // 视频ID
    $auditContent1["Status"] = "Blocked"; // 审核状态
    $auditContent1["Reason"] = "Reason"; // 若审核状态为屏蔽时，需给出屏蔽的理由，最长支持128字节
    $auditContent[] = $auditContent1;

    return json_encode($auditContent);
}

/**
 * 人工审核
 */
function createAudit($client) {
    $request = new vod\CreateAuditRequest();
    // 设置审核内容
    $request->setAuditContent(buildAuditContent());

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = createAudit($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
/**
 * 获取人工审核历史
 */
function getAuditHistory($client) {
    $request = new vod\GetAuditHistoryRequest();
    // 视频ID
    $request->setVideoId("070bbc13d8294e35b36c3e7ab452****");
    // 页号
    $request->setPageNo("1");
    // 每页数量
    $request->setPageSize("10");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = getAuditHistory($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
/**
 * 设置审核安全IP
 */
function setAuditSecurityIp($client) {
    $request = new vod\SetAuditSecurityIpRequest();
    /// 审核安全组名称，默认为Default
    $request->setSecurityGroupName("MyGroupName");
    // 操作方式，默认值Append(追加IP白名单)
    $request->setOperateMode("Cover");
    // 审核安全IP列表，多个用逗号分隔
    $request->setIps("10.23.12.20,10.23.12.21,10.23.12.22");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = setAuditSecurityIp($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
/**
 * 获取审核安全IP列表
 */
function listAuditSecurityIp($client) {
    $request = new vod\ListAuditSecurityIpRequest();
    /// 审核安全组名称，默认为Default
    $request->setSecurityGroupName("MyGroupName");

    return $client->getAcsResponse($request);
}

/**
 * 以下为调用示例
 */
try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = listAuditSecurityIp($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.13. 视频AI

本篇文档提供了PHP SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
function submitAIJob($client) {
    $request = new vod\SubmitAIJobRequest();
    // 设置视频ID
    $request->setMediaId('04df94402a5342928de6d6a2652****');
    // 设置AI类型，请确保已开通该类型AI
    $request->setTypes('AIVideoCover');
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = submitAIJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
function listAIJob($client) {
    $request = new vod\ListAIJobRequest();
    // 设置作业ID
    $request->setJobIds("7664bb4f06c44f0a84ccb49f452****,1234b4f06c44f0a84ccb49f452****");
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = listAIJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
function addAITemplate($client) {
    $request = new vod\AddAITemplateRequest();
    // 设置模板类型，以智能审核模板为例
    $request->setTemplateType("AIMediaAudit");
    // 设置模板名称
    $request->setTemplateName("myaitemplate");
    // 设置模板详细配置
    $templateConfigrray = array();
    $auditItem = array();
    $auditItem[] = "terrorism";
    $auditItem[] = "porn";
    $templateConfig["AuditItem"] = $auditItem;

    $auditRange = array();
    $auditRange[] = "video";
    $auditRange[] = "image-cover";
    $auditRange[] = "text-title";
    $templateConfig["AuditRange"] = $auditRange;

    $auditContent = array();
    $auditContent[] = "screen";
    $templateConfig["AuditContent"] = $auditContent;

    $templateConfig["AuditAutoBlock"] = "no";
    $request->setTemplateConfig(json_encode($templateConfig));
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = addAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
function deleteAITemplate($client) {
    $request = new vod\DeleteAITemplateRequest();
    // 设置模板ID
    $request->setTemplateId("3e539dfefc651c6796fea2549a****");

    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = deleteAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
function updateAITemplate($client) {
    $request = new vod\UpdateAITemplateRequest();
    // 设置模板ID
    $request->setTemplateId("3e539dfefc651c6796fea2549a****");
    // 设置模板名称
    $request->setTemplateName("myaitempalte1");
    // 设置模板详细配置
    $templateConfigrray = array();
    $auditItem = array();
    $auditItem[] = "terrorism";
    //$auditItem[] = "porn";
    $templateConfig["AuditItem"] = $auditItem;

    $auditRange = array();
    $auditRange[] = "video";
    $auditRange[] = "image-cover";
    //$auditRange[] = "text-title";
    $templateConfig["AuditRange"] = $auditRange;

    $auditContent = array();
    $auditContent[] = "screen";
    $templateConfig["AuditContent"] = $auditContent;

    $templateConfig["AuditAutoBlock"] = "no";
    $request->setTemplateConfig(json_encode($templateConfig));
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = updateAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
function getAITemplate($client) {
    $request = new vod\GetAITemplateRequest();
    // 设置模板ID
    $request->setTemplateId("3e539dfefc651c6796fea25452****");

    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = getAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
function listAITemplate($client) {
    $request = new vod\ListAITemplateRequest();
    // 设置模板类型，以智能审核模板为例
    $request->setTemplateType("AIMediaAudit");

    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = listAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
function setDefaultAITemplate($client) {
    $request = new vod\SetDefaultAITemplateRequest();
    // 设置模板ID
    $request->setTemplateId("3e539dfefc651c6796fea25492****");

    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = setDefaultAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
function getDefaultAITemplate($client) {
    $request = new vod\GetDefaultAITemplateRequest();
    // 设置模板类型，以智能审核模板为例
    $request->setTemplateType("AIMediaAudit");

    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = getDefaultAITemplate($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.14. 视频DNA

本文为您提供了PHP SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
function submitAIJob($client) {
    $request = new vod\SubmitAIJobRequest();
    // 设置视频ID
    $request->setMediaId('3eb19a4585bc475e995bdd52****');
    // 设置AI类型，类型为AIMediaDNA
    $request->setTypes('AIMediaDNA');
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = submitAIJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
function listAIJob($client) {
    $request = new vod\ListAIJobRequest();
    // 设置作业ID
    $request->setJobIds("979d4d7a36ae41b1a945a252****,3eb19a4585bc475e995bddea52****");
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = listAIJob($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
function getMediaDNAResult($client) {
    $request = new vod\GetMediaDNAResultRequest();
    // 设置视频ID
    $request->setMediaId("3eb19a4585bc475e995bdd52****");
    // 返回结果
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = getMediaDNAResult($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.15. 直播转点播

本篇文档提供了PHP SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
function listLiveRecordVideo($client) {
    $request = new vod\ListLiveRecordVideoRequest();
    $request->setStartTime("2018-04-24T03:21:04Z");
    $request->setEndTime("2018-05-21T03:21:44Z");
    $request->setStreamName("testStream****");
    $request->setDomainName("example.aliyundoc.com");
    $request->setAppName("testApp****");
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient("<AccessKeyId>", "<AccessKeySecret>");

    $result = listLiveRecordVideo($client);
    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

4.16. 点播CDN

本篇文档提供了PHP SDK点播CDN模块相关功能的API调用示例。包含预热缓存、刷新缓存、查询刷新和预热状态、查询刷新预热次数限制和余量、查询流量数据、查询网络带宽、下载域名日志。

初始化客户端

使用前请先初始化客户端，请参见初始化。

预热缓存

调用PreloadVodObjectCaches接口，完成预热缓存功能。

接口参数和返回字段请参见PreloadVodObjectCaches。调用示例如下：

```
function preloadVodObjectCaches($client) {
    $request = new vod\PreloadVodObjectCachesRequest();
    // 需要预热的文件路径，示例：http://example.com/fd****.mp4
    $request->setObjectPath("<your Object Path>");

    return $client->getAcsResponse($request);
}


try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = preloadVodObjectCaches($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

刷新缓存

调用RefreshVodObjectCaches接口，完成刷新缓存功能。

接口参数和返回字段请参见RefreshVodObjectCaches。调用示例如下：

```
function refreshVodObjectCaches($client) {
    $request = new vod\RefreshVodObjectCachesRequest();
    // 需要刷新的文件或者目录路径，示例：http://example.com/fd****.mp4
    $request->setObjectPath("<your Object Path>");
    // 刷新的类型
    $request->setObjectType("File");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = refreshVodObjectCaches($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询刷新和预热状态

调用DescribeVodRefreshTasks接口，完成查询刷新和预热状态功能。

接口参数和返回字段请参见DescribeVodRefreshTasks。调用示例如下：

```
function describeVodRefreshTasks($client) {
    $request = new vod\DescribeVodRefreshTasksRequest();
    // 需要查询的域名
    $request->setDomainName("example.com");
    // 任务类型
    $request->setObjectType("file");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = describeVodRefreshTasks($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询刷新预热次数限制和余量

调用DescribeVodRefreshQuota接口，完成查询刷新预热次数限制和余量功能。

接口参数和返回字段请参见DescribeVodRefreshQuota。调用示例如下：

```
function describeVodRefreshQuota($client) {
    $request = new vod\DescribeVodRefreshQuotaRequest();
    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = describeVodRefreshQuota($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询流量数据

调用DescribeVodDomainTrafficData接口，完成查询流量数据功能。

接口参数和返回字段请参见DescribeVodDomainTrafficData。调用示例如下：

```
function describeVodDomainTrafficData($client) {
    $request = new vod\DescribeVodDomainTrafficDataRequest();
    // 设置域名
    $request->setDomainName("example.aliyundoc.com");
    // 设置开始时间，请使用UTC格式
    $request->setStartTime("2019-01-15T15:59:59Z");
    // 设置结束时间，请使用UTC格式
    $request->setEndTime("2019-01-20T15:59:58Z");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = describeVodDomainTrafficData($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

查询网络带宽

调用DescribeVodDomainBpsDat a接口，完成查询网络带宽功能。

接口参数和返回字段请参见DescribeVodDomainBpsData。调用示例如下：

```
function describeVodDomainBpsData($client) {
    $request = new vod\DescribeVodDomainBpsDataRequest();
    // 设置域名
    $request->setDomainName("example.aliyundoc.com");
    // 设置开始时间，请使用UTC格式
    $request->setStartTime("2019-01-15T15:59:59Z");
    // 设置结束时间，请使用UTC格式
    $request->setEndTime("2019-01-20T15:59:58Z");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = describeVodDomainBpsData($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

下载域名日志

调用DescribeVodDomainLog接口，完成下载域名日志功能。

接口参数和返回字段请参见DescribeVodDomainLog。调用示例如下：

```
function describeVodDomainLog($client) {
    $request = new vod\DescribeVodDomainLogRequest();
    // 设置域名
    $request->setDomainName("example.alicdn.com");
    // 设置开始时间，请使用UTC格式
    $request->setStartTime("2019-01-15T21:00:00Z");
    // 设置结束时间，请使用UTC格式
    $request->setEndTime("2019-01-15T22:00:00Z");

    return $client->getAcsResponse($request);
}

try {
    $client = initVodClient('<AccessKeyId>', '<AccessKeySecret>');
    $result = describeVodDomainLog($client);

    var_dump($result);
} catch (Exception $e) {
    print $e->getMessage()."\n";
}
```

5..NET SDK

5.1. 安装

本文为您介绍了视频点播服务端.NET SDK的安装步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

- 安装.NET 2.0及以上版本。
- 安装Visual Studio 2017及以上版本。
- 安装Mono 3.12及以上版本。

GitHub安装

1. 安装Git，已经安装Git可跳过本步骤。
2. 开始**git clone**，下载最新的源码aliyun-net-sdk-core和aliyun-net-sdk-vod。
3. 源码下载完成后，按照下方项目引入方式安装即可。

Visual Studio（解决方案资源管理器）项目引入

1. 单击**视图**选择**解决方案资源管理器**。
2. 在界面右侧区域，右键单击**解决方案**，在弹出的菜单中单击**添加**选择**现有项目**。
3. 在弹出的对话框中选择aliyun-net-sdk-core.csproj和aliyun-net-sdk-vod.csproj文件，单击**打开**。
4. 右键单击**项目**选择**引用**，选择**添加引用**。在弹出的对话框中，选择**项目选项卡**。
5. 选中aliyun-net-sdk-core和aliyun-net-sdk-vod项目，单击**确定**。

5.2. 初始化

视频点播服务端.NET SDK可以通过AccessKey始化。本文为您介绍初始化步骤以及调用SDK接口返回错误码时如何解决该问题。

前提条件

- 您已经安装视频点播服务端.NET SDK。安装步骤，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。示例如下：

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Profile;
public static DefaultAcsClient InitVodClient(string accessKeyId, string accessKeySecret)
{
    // 点播服务接入地域
    string regionId = "cn-shanghai";
    //填写AccessKey信息
    IClientProfile profile = DefaultProfile.GetProfile(regionId, accessKeyId, accessKeySecr
et);
    // DefaultProfile.AddEndpoint(regionId, regionId, "vod", "vod." + regionId + ".aliyuncs
.com");
    return new DefaultAcsClient(profile);
}
```

常见问题

当您设置的接入地域ID（Regionid）在点播中心和访问域名列表中，由于部分SDK版本兼容性问题，调用SDK接口时可能返回 `SDK.InvalidRegionId, Can not find endpoint to access.` 错误码。此时请将上方示例代码InitVodClient中注释掉的 `AddEndpoint` 方法打开来解决该问题。示例如下：

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Profile;
public static DefaultAcsClient InitVodClient(string accessKeyId, string accessKeySecret)
{
    // 点播服务接入地域
    string regionId = "cn-shanghai";
    //填写AccessKey信息
    IClientProfile profile = DefaultProfile.GetProfile(regionId, accessKeyId, accessKeySecr
et);
    return new DefaultAcsClient(profile);
}
```

5.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍.NET SDK调用服务端上传接口的使用场景和调用示例。

使用场景

> **注意**
>
> 本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端实例。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端实例。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段，请参见CreateUploadVideo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.CreateUploadVideo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                CreateUploadVideoRequest request = new CreateUploadVideoRequest();
                request.Title = "this is a sample title";
                request.FileName = "sample.mp4";
                // request.Tags = "tags1,tags2";
                // request.Description = "this is a sample description";
                //CoverURL示例：http://192.168.0.0/16/sample****.jpg
                // request.CoverURL = "<your Cover URL>";
                // request.CateId = -1;
                // request.TemplateGroupId = "278840921dee4963bb5862b43a52****";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到响应结果
                CreateUploadVideoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("UploadAddress = " + response.UploadAddress);
                Console.WriteLine("UploadAuth = " + response.UploadAuth);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

> **注意**
>
> 调用CreateUploadVideo接口，生成上传凭证、上传地址和VideoID时，点播控制台视频列表状态显示为**上传中**，如果客户端后续不进行实际视频上传，或者上传报错，或者是正在上传的时候把上传进程终断了（比如结束App进程），都会导致控制台这个视频状态一直显示为**上传中**，请调用刷新视频上传凭证接口，传入这个视频ID去获取新的凭证和地址，再去上传，或者手动删除掉这个状态为上传中的视频。更多信息，请参见RefreshUploadVideo（刷新视频上传凭证）。视频实际上传，请调用UploadMediaByURL（URL批量拉取上传）接口。

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                RefreshUploadVideoRequest request = new RefreshUploadVideoRequest();
                request.VideoId = "视频ID";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                RefreshUploadVideoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("UploadAddress = " + response.UploadAddress);
                Console.WriteLine("UploadAuth = " + response.UploadAuth);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                CreateUploadImageRequest request = new CreateUploadImageRequest();
                request.ImageType = "cover";
                request.ImageExt = "jpg";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                CreateUploadImageResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("UploadAddress = " + response.UploadAddress);
                Console.WriteLine("UploadAuth = " + response.UploadAuth);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace CreateUploadAttachedMedia
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                CreateUploadAttachedMediaRequest request = new CreateUploadAttachedMediaReq
uest();
                request.BusinessType = "watermark";
                request.MediaExt = "gif";
                request.Title = "this is a sample";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                CreateUploadAttachedMediaResponse response = client.GetAcsResponse(request)
;
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("MediaId = " + response.MediaId);
                Console.WriteLine("MediaURL = " + response.MediaURL);
                Console.WriteLine("UploadAddress = " + response.UploadAddress);
                Console.WriteLine("UploadAuth = " + response.UploadAuth);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.UploadMediaByURL
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                UploadMediaByURLRequest request = new UploadMediaByURLRequest();
                //SourceURL示例：http://192.168.0.0/16/test****.mp4
                request.UploadMetadatas = "[{\"SourceUrl\":\"<your Source URL>\",\"Title\":
\"upload by url sample\"}]";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UploadMediaByURLResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

5.4. 音视频播放

本篇文档提供了.NET SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using System.Collections.Generic;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetPlayInfoRequest request = new GetPlayInfoRequest();
                request.VideoId = "<your VideoId>";
                // request.AuthTimeout = 3600;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient();
                // 发起请求，并得到 response
                GetPlayInfoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("VideoBase.Title = " + response.VideoBase.Title);
                List<GetPlayInfoResponse.GetPlayInfo_PlayInfo> playInfoList = response.Play
InfoList;
                foreach (var playInfo in response.PlayInfoList)
                {
                    Console.WriteLine("PlayInfoList.PlayURL = " + playInfo.PlayURL);
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetVideoPlayAuthRequest request = new GetVideoPlayAuthRequest();
                request.VideoId = "<your VideoId>";
                //request.AuthInfoTimeout = 3000;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient();
                // 发起请求，并得到 response
                GetVideoPlayAuthResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("VideoMeta.Title = " + response.VideoMeta.Title);
                Console.WriteLine("PlayAuth = " + response.PlayAuth);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

5.5. 媒资管理

本篇文档提供了.NET SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
using System;
```

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SearchMedia
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                SearchMediaRequest request = new SearchMediaRequest();
                request.SearchType = "video";
                request.Fields = "Title,CoverURL,Status";
                request.Match = "Status in ('Normal','Checking') and CreationTime = ('2018-
07-01T08:00:00Z','2018-08-01T08:00:00Z')";
                request.PageNo = 1;
                request.PageSize = 10;
                request.SearchType = "video";
                request.SortBy = "CreationTime:Desc";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SearchMediaResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.MediaList != null && response.MediaList.Count > 0)
                {
                    Console.WriteLine("Total = " + response.Total + "\n");
                    Console.WriteLine("ScrollToken = " + response.ScrollToken + "\n");
                    foreach (var media in response.MediaList)
                    {
                        Console.WriteLine("MediaId = " + media.MediaId + "\n");
                        Console.WriteLine("MediaType = " + media.MediaType + "\n");
                        Console.WriteLine("CreationTime = " + media.CreationTime + "\n");
                        Console.WriteLine("Title = " + media.Video.Title + "\n");
                        Console.WriteLine("CoverURL = " + media.Video.CoverURL + "\n");
                        Console.WriteLine("Status = " + media.Video.Status + "\n");
                    }
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetVideoInfo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetVideoInfoRequest request = new GetVideoInfoRequest();
                request.VideoId = "<your VideoId>";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetVideoInfoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("Title = " + response.Video.Title);
                Console.WriteLine("CoverURL = " + response.Video.CoverURL);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetVideoInfos
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetVideoInfosRequest request = new GetVideoInfosRequest();
                request.VideoIds = "VideoId1,VideoId2";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetVideoInfosResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.NonExistVideoIds != null && response.NonExistVideoIds.Count >
0)
                {
                    foreach (var videoId in response.NonExistVideoIds) {
                        Console.WriteLine("NonExist videoId = " + videoId);
                    }
                }
                if (response.VideoList != null && response.VideoList.Count > 0)
                {
                    foreach (var video in response.VideoList)
                    {
                        Console.WriteLine("MediaId = " + video.VideoId);
                        Console.WriteLine("Title = " + video.Title);
                        Console.WriteLine("CreationTime = " + video.CreationTime);
                        Console.WriteLine("CoverURL = " + video.CoverURL);
                        Console.WriteLine("Status = " + video.Status);
                    }
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.UpdateVideoInfo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                UpdateVideoInfoRequest request = new UpdateVideoInfoRequest();
                request.VideoId = "<your VideoId>";
                request.Title = "new Title";
                request.Description = "new Description";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateVideoInfoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.UpdateVideoInfos
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                UpdateVideoInfosRequest request = new UpdateVideoInfosRequest();
                request.UpdateContent = "[{\"VideoId\":\"VideoId1\",\"Title\":\"new Title\"
},{\"VideoId\":\"VideoId2\",\"Description\":\"new Description\"}]";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateVideoInfosResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.NonExistVideoIds != null && response.NonExistVideoIds.Count >
0)
                {
                    foreach (var videoId in response.NonExistVideoIds)
                    {
                        Console.WriteLine("NonExist videoId = " + videoId);
                    }
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteVideo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteVideoRequest request = new DeleteVideoRequest();
                request.VideoIds = "VideoId1,VideoId2";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteVideoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取源文件信息（含源片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetMezzanineInfo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetMezzanineInfoRequest request = new GetMezzanineInfoRequest();
                request.VideoId = "f45cf4eba5c84d25b29023338955****";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetMezzanineInfoResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("FileURL = " + response.Mezzanine.FileURL);
                Console.WriteLine("Duration = " + response.Mezzanine.Duration);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetVideoListRequest request = new GetVideoListRequest();
                // 分别取一个月前、当前时间的UTC时间作为筛选视频列表的起止时间
                request.StartTime = DateTime.UtcNow.AddMonths(-2).ToString("yyyy-MM-ddTHH:m
m:ssZ");
                request.EndTime = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ");
                request.Status = "Normal,Checking";
                request.PageNo = 1;
                request.PageSize = 20;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetVideoListResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                foreach (var videoInfo in response.VideoList)
                {
                    Console.WriteLine("VideoList.VideoId = " + videoInfo.VideoId);
                    Console.WriteLine("VideoList.Title = " + videoInfo.Title);
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteStreamRequest request = new DeleteStreamRequest();
                request.VideoId = "<your VideoId>";
                request.JobIds = "JobId1,JobId2";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteStreamResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteMezzanines
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteMezzaninesRequest request = new DeleteMezzaninesRequest();
                // 待删除的视频ID列表
                request.VideoIds = "VideoId1,VideoId2";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteMezzaninesResponse response = client.GetAcsResponse(request);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.NonExistVideoIds != null && response.NonExistVideoIds.Count >
0)
                {
                    foreach (var videoId in response.NonExistVideoIds)
                    {
                        Console.WriteLine("NonExist videoId = " + videoId);
                    }
                }
                if (response.UnRemoveableVideoIds != null && response.UnRemoveableVideoIds.
Count > 0)
                {
                    foreach (var videoId in response.UnRemoveableVideoIds)
                    {
                        Console.WriteLine("UnRemoveable videoId = " + videoId);
                    }
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.UpdateImageInfos
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                UpdateImageInfosRequest request = new UpdateImageInfosRequest();

                request.UpdateContent = "[{\"ImageId\":\"ImageId1\",\"Title\":\"new Title\"
},{\"ImageId\":\"ImageId2\",\"Description\":\"new Description\"}]";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateImageInfosResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.NonExistImageIds != null && response.NonExistImageIds.Count >
0)
                {
                    foreach (var imageId in response.NonExistImageIds)
                    {
                        Console.WriteLine("NonExist ImageId = " + imageId);
                    }
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetImageInfo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetImageInfoRequest request = new GetImageInfoRequest();
                // 图片ID
                request.ImageId = "ImageId";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetImageInfoResponse response = client.GetAcsResponse(request);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.ImageInfo != null)
                {
                    Console.WriteLine("ImageId = " + response.ImageInfo.ImageId);
                    Console.WriteLine("ImageType = " + response.ImageInfo.ImageType);
                    Console.WriteLine("Title = " + response.ImageInfo.Title);
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteImage
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteImageRequest request = new DeleteImageRequest();
                //根据ImageURL删除图片文件
                request.DeleteImageType = "ImageURL";
                request.ImageURLs = "http://192.168.0.0/16/cover.jpg";
                //根据ImageId删除图片文件
                //request.DeleteImageType = "ImageId";
                //request.ImageIds = "ImageId1,ImageId2";
                //根据VideoId删除指定ImageType的图片文件
                //request.DeleteImageType = "VideoId";
                //request.VideoId = "VideoId";
                //request.ImageType = "SpriteSnapshot";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteImageResponse response = client.GetAcsResponse(request);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

5.6. 媒资分类

本文为您提供了.NET SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                AddCategoryRequest request = new AddCategoryRequest();
                // 父分类ID，若不填，则默认生成一级分类，根节点分类ID为 -1
                request.ParentId = -1;
                // 分类名称
                request.CateName = "catename";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                AddCategoryResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("CateId = " + response.Category.CateId);
                Console.WriteLine("CateName = " + response.Category.CateName);
                Console.WriteLine("ParentId = " + response.Category.ParentId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                UpdateCategoryRequest request = new UpdateCategoryRequest();
                // 请设置真实分类ID
                request.CateId = -1;
                // 分类名称
                request.CateName = "new catename";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateCategoryResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteCategoryRequest request = new DeleteCategoryRequest();
                // 设置真实分类ID
                request.CateId = -1;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteCategoryResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.Sample
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetCategoriesRequest request = new GetCategoriesRequest();
                // 分类ID，若不填，则默认获取根节点及其子分类，根节点分类ID为-1
                request.CateId = -1;
                request.PageNo = 1;
                request.PageSize = 10;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetCategoriesResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                // 分类信息
                Console.WriteLine("Category.CateId = " + response.Category1.CateId);
                Console.WriteLine("Category.CateName = " + response.Category1.CateName);
                Console.WriteLine("Category.ParentId = " + response.Category1.ParentId);
                Console.WriteLine("SubTotal = " + response.SubTotal);
                // 子分类列表
                foreach (var subcategory in response.SubCategories)
                {
                    Console.WriteLine("subcategory.CateId = " + subcategory.CateId);
                    Console.WriteLine("subcategory.CateName = " + subcategory.CateName);
                    Console.WriteLine("subcategory.ParentId = " + subcategory.ParentId);
                }
            }
            catch (ServerException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            catch (ClientException ex)
            {
                Console.WriteLine(ex.ToString());
            }
        }
    }
}
```

5.7. 媒体处理

本文提供了.NET SDK媒体处理模块相关功能的API调用示例，包含提交转码作业、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业

调用SubmitTranscodeJobs接口，完成提交转码作业功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Kms.Model.V20160120;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.SubmitTranscodeJobs
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitTranscodeJobsResponse response = SubmitTranscodeJobs(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 提交转码作业函数
        /// </summary>
        /// <returns>转码JobId.</returns>
        /// <param name="client">Client</param>
        public static SubmitTranscodeJobsResponse SubmitTranscodeJobs(DefaultAcsClient clie
nt)
        {
            // 构造请求
            SubmitTranscodeJobsRequest request = new SubmitTranscodeJobsRequest();
            // 需要转码的视频ID
            request.VideoId = "1f7ed5623409489193e41d16f6c7****";
            // 转码模板ID
            request.TemplateGroupId = "3cac80cddb4455e37460f491aecc****";
            // 构建需要替换的水印参数(只有需要替换水印相关信息才需要构建)
            request.OverrideParams = BuildOverrideParams();
            // 覆盖参数，暂只支持水印部分参数替换(只有需要替换水印相关信息才需要传递)
            request.EncryptConfig = BuildEncryptConfig(client);
            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 生成加密需要的密钥，response中包含密文密钥和明文密钥，用户只需要将密文密钥传递给点播即可
        /// 注意：KeySpec 必须传递AES_128，且不能设置NumberOfBytes
        /// </summary>
        /// <returns> DataKey </returns>
        /// <param name="client">KMS-SDK Client.</param>
        /// <param name="serviceKey">点播提供生成密钥的service key，在用户的密钥管理服务中可看到
描述为vod的加密key.</param>
        ///
        public static GenerateDataKeyResponse GenerateDataKey(DefaultAcsClient client, stri
ng serviceKey)
        {
            GenerateDataKeyRequest request = new GenerateDataKeyRequest();
            request.KeyId = serviceKey;
            request.KeySpec = "AES_128";

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建HLS标准加密的配置信息
        /// </summary>
        /// <returns>配置信息</returns>
        /// <param name="client"> VoD Client.</param>
        ///
        public static string BuildEncryptConfig(DefaultAcsClient client)
        {
```

```
            // 点播给用户在KMS(密钥管理服务)中的Service Key，可在用户密钥管理服务对应的区域看到描述为vod的service key
            string serviceKey = "<Your Service Key>";

            // 随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥，
            // 视频标准加密只需要传递密文密钥即可
            GenerateDataKeyResponse response = GenerateDataKey(client, serviceKey);

            JObject encryptConfig = new JObject();

            // 解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥都不一样
            // 至于Ciphertext这个解密接口参数的名称，用户可自行制定，这里只作为参考参数名称
h           // DecryptKeyUri示例：http://192.168.0.0/16/decrypt?" + "Ciphertext=" + response.CiphertextBlob
            encryptConfig.Add("DecryptKeyUri", "<your DecryptKeyUri>");

            // 密钥服务的类型，目前只支持KMS
            encryptConfig.Add("KeyServiceType", "KMS");
            // 密文密钥
            encryptConfig.Add("CipherText", response.CiphertextBlob);

            return encryptConfig.ToString();
        }

        /// <summary>
        /// 1、构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；
        /// 2、需要替换的水印信息对应水印ID必须是关联在指定的模板ID(即TranscodeTemplateId)中；
        /// 3、不支持通过媒体处理接口去增加一个没有关联上的水印
        /// 注意：图片水印的文件存储源站需要和发起转码的视频存储源站一致
        /// </summary>
        /// <returns>The override parameters.</returns>
        ///
        public static string BuildOverrideParams() {
            // 覆盖参数
            JObject overrideParams = new JObject();

            // 图片水印文件地址替换
            JArray watermarks = new JArray();
            JObject watermark1 = new JObject();
            // 模板上面关联需要替换的水印文件图片水印ID
            watermark1.Add("WatermarkId", "2ea587477c5a1bc8b57****");
            // 需要替换成对应图片水印文件的OSS地址，水印文件存储源站需要和视频存储源站一致
            //File URL示例：https://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermarks/02A1B22DF25D46C3C725A****.png
            watermark1.Add("FileUrl", "<your File URL>");
            watermarks.Add(watermark1);

            // 文字水印内容替换
            JObject watermark2 = new JObject();
            // 模板上面关联需要替换内容的文字水印ID
            watermark2.Add("WatermarkId", "d297ba31ac5242d207****");
            // 需要替换成对应的内容
            watermark2.Add("Content", "用户ID：52****");
```

```
            watermarks.Add(watermark2);

            overrideParams.Add("Watermarks", watermarks);

            return overrideParams.ToString();
        }
    }
}
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明**
>
> 创建截图模板详细请参见截图模板。

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.SubmitSnapshotJob
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitSnapshotJobResponse response = SubmitSnapshotJob(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
```

```
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        public static SubmitSnapshotJobResponse SubmitSnapshotJob(DefaultAcsClient client)
        {
            // 构造请求
            SubmitSnapshotJobRequest request = new SubmitSnapshotJobRequest();
            //需要截图的视频ID(推荐传递截图模板ID)
            request.VideoId = "1f7ed5623409489193e41d16f6c7****";
            //截图模板ID
            request.SnapshotTemplateId = "3cac80cddb4455e37460f491aecc****";
            //如果设置了SnapshotTemplateId，会忽略下面参数
            request.Count = 50;
            request.SpecifiedOffsetTime = 0;
            request.Interval = 1;
            request.Width = "200";
            request.Height = "200";
            request.SpriteSnapshotConfig = BuildSnapshotTemplateConfig();
            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建雪碧图截图配置
        /// </summary>
        /// <returns>雪碧图截图配置.</returns>
        public static string BuildSnapshotTemplateConfig()
        {
            // 覆盖参数
            JObject spriteSnapshotConfig = new JObject();
            spriteSnapshotConfig.Add("CellWidth", "120");
            spriteSnapshotConfig.Add("CellHeight", "68");
            spriteSnapshotConfig.Add("Columns", "3");
            spriteSnapshotConfig.Add("Lines", "10");
            spriteSnapshotConfig.Add("Padding", "20");
            spriteSnapshotConfig.Add("Margin", "50");

            //保留雪碧图原始图
            spriteSnapshotConfig.Add("KeepCellPic", "keep");
            spriteSnapshotConfig.Add("Color", "tomato");
            return spriteSnapshotConfig.ToString();
        }
    }
}
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListSnapshots
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListSnapshotsResponse response = ListSnapshots(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        public static ListSnapshotsResponse ListSnapshots(DefaultAcsClient client)
        {
            // 构造请求
            ListSnapshotsRequest request = new ListSnapshotsRequest();
            // 视频ID
            request.VideoId = "1f7ed5623409489193e41d16f6c7****";
            // 截图类型
            request.SnapshotType = "CoverSnapshot";
            request.PageNo = "1";
```

```
            request.PageSize = "20";
            return client.GetAcsResponse(request);
        }
    }
}
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SubmitPreprocessJobs
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitPreprocessJobsResponse response = SubmitPreprocessJobs(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
```

```
        /// <summary>
        /// 导播台视频预处理.
        /// </summary>
        /// <returns>The preprocess jobs.</returns>
        /// <param name="client">Vod Client.</param>
        public static SubmitPreprocessJobsResponse SubmitPreprocessJobs(DefaultAcsClient cl
ient)
        {
            // 构造请求
            SubmitPreprocessJobsRequest request = new SubmitPreprocessJobsRequest();
            // 需要截图的视频ID(推荐传递截图模板ID)
            request.VideoId = "c86c0ceb8db54ae09796535****";
            // 截图类型
            request.PreprocessType = "PreprocessType";

            return client.GetAcsResponse(request);
        }
    }
}
```

5.8. 视频水印

本篇文档提供了.NET SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印、设置默认水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

> **说明**
>
> - 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
> - 水印文件OSS上传详细参数请参见OSS文件上传。

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.AddWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
```

```
        /// <summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                AddWatermarkResponse response = AddWatermark(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 添加水印方法
        /// </summary>
        /// <returns>The watermark.</returns>
        /// <param name="client">Client.</param>
        public static AddWatermarkResponse AddWatermark(DefaultAcsClient client)
        {
            // 构造请求
            AddWatermarkRequest request = new AddWatermarkRequest();
            // 水印名称
            request.Name = "addwatermark";
            // 图片水印必传图片文件的oss文件地址，水印文件必须和视频在同一个区域，例如:华东2视频，水
印文件必须存放在华东2
            request.FileUrl = "http://192.168.0.0/16/watermark/test.png";

            // 图片水印的位置配置数据
            request.WatermarkConfig = BuildImageWatermarkConfig();

            // 文字水印的位置配置数据
```

```
            request.WatermarkConfig = BuildTextWatermarkConfig();

            // 文字水印:Text; 图片水印:Image
            request.Type = "Image";

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
        /// </summary>
        /// <returns>The image watermark config.</returns>
        public static string BuildImageWatermarkConfig()
        {
            JObject watermarkConfig = new JObject();
            // 水印的横向偏移距离
            watermarkConfig.Add("Dx", "8");
            // 水印的纵向偏移距离
            watermarkConfig.Add("Dy", "8");
            // 水印显示的宽
            watermarkConfig.Add("Width", "55");
            // 水印显示的高
            watermarkConfig.Add("Height", "55");
            // 水印显示的相对位置(左上、右上、左下、右下)
            watermarkConfig.Add("ReferPos", "BottomRight");

            // 水印显示的时间线(开始显示和结束显示时间)
            JObject timeline = new JObject();
            // 水印开始显示时间
            timeline.Add("Start", "2");
            // 水印结束显示时间
            timeline.Add("Duration", "ToEND");

            watermarkConfig.Add("Timeline", timeline);

            return watermarkConfig.ToString();
        }

        /// <summary>
        /// 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
        /// </summary>
        /// <returns>The text watermark config.</returns>
        public static string BuildTextWatermarkConfig()
        {
            JObject watermarkConfig = new JObject();
            // 文字水印显示的内容
            watermarkConfig.Add("Content", "testwatermark");
            // 文字水印的字体名称
            watermarkConfig.Add("FontName", "SimSun");
            // 文字水印的字体大小
            watermarkConfig.Add("FontSize", 25);
            // 文字水印的颜色(也可为RGB颜色取值，例如:#000000)
            watermarkConfig.Add("FontColor", "Black");
            // 文字水印的透明度
```

```
            watermarkConfig.Add("FontAlpha", "0.2");
            // 文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
            watermarkConfig.Add("BorderColor", "White");
            // 文字水印的描边宽度
            watermarkConfig.Add("BorderWidth", 1);
            // 文字水印距离视频画面上边的偏移距离
            watermarkConfig.Add("Top", 20);
            // 文字水印距离视频画面左边的偏移距离
            watermarkConfig.Add("Left", 15);

            return watermarkConfig.ToString();
        }
    }
}
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.UpdateWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateWatermarkResponse response = UpdateWatermark(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
```

```
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 更新水印
        /// </summary>
        /// <returns>The watermark.</returns>
        /// <param name="client">Client.</param>
        public static UpdateWatermarkResponse UpdateWatermark(DefaultAcsClient client)
        {
            // 构造请求
            UpdateWatermarkRequest request = new UpdateWatermarkRequest();
            // 水印名称
            request.Name = "updateWatermark";
            // 需要更新配置信息的水印ID
            request.WatermarkId = "421ddddd4f6e734a526fd2****";

            // 图片水印的位置配置数据
            request.WatermarkConfig = BuildImageWatermarkConfig();

            // 文字水印的位置配置数据
            // request.WatermarkConfig = BuildTextWatermarkConfig();

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
        /// </summary>
        /// <returns>The image watermark config.</returns>
        public static string BuildImageWatermarkConfig()
        {
            JObject watermarkConfig = new JObject();
            // 水印的横向偏移距离
            watermarkConfig.Add("Dx", "8");
            // 水印的纵向偏移距离
            watermarkConfig.Add("Dy", "8");
            // 水印显示的宽
            watermarkConfig.Add("Width", "55");
            // 水印显示的高
            watermarkConfig.Add("Height", "55");
            // 水印显示的相对位置(左上、右上、左下、右下)
            watermarkConfig.Add("ReferPos", "BottomRight");
```

```
            watermarkConfig.Add("ReferPos", "BottomRight");

            // 水印显示的时间线(开始显示和结束显示时间)
            JObject timeline = new JObject();
            // 水印开始显示时间
            timeline.Add("Start", "2");
            // 水印结束显示时间
            timeline.Add("Duration", "ToEND");

            watermarkConfig.Add("Timeline", timeline);

            return watermarkConfig.ToString();
        }

        /// <summary>
        /// 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
        /// </summary>
        /// <returns>The text watermark config.</returns>
        public static string BuildTextWatermarkConfig()
        {
            JObject watermarkConfig = new JObject();
            // 文字水印显示的内容
            watermarkConfig.Add("Content", "testwatermark");
            // 文字水印的字体名称
            watermarkConfig.Add("FontName", "SimSun");
            // 文字水印的字体大小
            watermarkConfig.Add("FontSize", 25);
            // 文字水印的颜色(也可为RGB颜色取值，例如:#000000)
            watermarkConfig.Add("FontColor", "Black");
            // 文字水印的透明度
            watermarkConfig.Add("FontAlpha", "0.2");
            // 文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
            watermarkConfig.Add("BorderColor", "White");
            // 文字水印的描边宽度
            watermarkConfig.Add("BorderWidth", 1);
            // 文字水印距离视频画面上边的偏移距离
            watermarkConfig.Add("Top", 20);
            // 文字水印距离视频画面左边的偏移距离
            watermarkConfig.Add("Left", 15);

            return watermarkConfig.ToString();
        }
    }
}
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
```

```
namespace Aliyun.Acs.vod.Sdk.DeleteWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteWatermarkResponse response = DeleteWatermark(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 删除水印配置信息函数
        /// </summary>
        /// <returns>The watermark response.</returns>
        /// <param name="client">Client.</param>
        public static DeleteWatermarkResponse DeleteWatermark(DefaultAcsClient client)
        {
            // 构造请求
            DeleteWatermarkRequest request = new DeleteWatermarkRequest();
            // 水印ID
            request.WatermarkId = "53f9d796fad9d7b862b2e****";
```

```
            return client.GetAcsResponse(request);
        }
    }
}
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListWatermarkResponse response = ListWatermark(client);

                if (response.WatermarkInfos != null && response.WatermarkInfos.Count > 0)
                {
                    foreach (ListWatermarkResponse.ListWatermark_WatermarkInfo watermarkInf
o in response.WatermarkInfos)
                    {
                        //水印ID
                        Console.WriteLine("WatermarkId = " + watermarkInfo.WatermarkId);
                        //水印的位置、效果配置数据
                        Console.WriteLine("WatermarkConfig = " + watermarkInfo.WatermarkCon
fig);

                        //水印文件的FileUrl(文字水印该字段值为空)
                        Console.WriteLine("FileUrl = " + watermarkInfo.FileUrl);
                    }
                }
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
```

```
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 获取水印列表
        /// </summary>
        /// <returns>The watermark list.</returns>
        /// <param name="client">Client.</param>
        public static ListWatermarkResponse ListWatermark(DefaultAcsClient client)
        {
            // 构造请求
            ListWatermarkRequest request = new ListWatermarkRequest();

            return client.GetAcsResponse(request);
        }

    }
}
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
```

```
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetWatermarkResponse response = GetWatermark(client);
                if (response.WatermarkInfo != null)
                {
                    //水印ID
                    Console.WriteLine("WatermarkId = " + response.WatermarkInfo.WatermarkId
);
                    //水印的位置、效果配置数据
                    Console.WriteLine("WatermarkConfig = " + response.WatermarkInfo.Waterma
rkConfig);
                    //水印文件的FileUrl(文字水印该字段值为空)
                    Console.WriteLine("FileUrl = " + response.WatermarkInfo.FileUrl);
                }
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询单个水印配置信息函数
        /// </summary>
        /// <returns>The watermark.</returns>
        /// <param name="client">Client.</param>
        public static GetWatermarkResponse GetWatermark(DefaultAcsClient client)
        {
            // 构造请求
            GetWatermarkRequest request = new GetWatermarkRequest();
            //需要查询水印信息的水印ID
```

```
            request.WatermarkId = "96d4f6e734a526fd2e442";
            return client.GetAcsResponse(request);
        }
    }
}
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SetDefaultWatermark
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SetDefaultWatermarkResponse response = SetDefaultWatermark(client);

                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 设置默认水印配置信息函数
        /// </summary>
        /// <returns>The default watermark.</returns>
        /// <param name="client">Client.</param>
        public static SetDefaultWatermarkResponse SetDefaultWatermark(DefaultAcsClient clie
nt)
        {
            // 构造请求
            SetDefaultWatermarkRequest request = new SetDefaultWatermarkRequest();
            // 设置默认的水印ID
            request.WatermarkId = "82105a29c6e96d4f6e****";

            return client.GetAcsResponse(request);
        }
    }
}
```

5.9. 转码模板

本篇文档提供了.NET SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.VoD.Sdk.AddTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
```

```
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                AddTranscodeTemplateGroupResponse response = AddTranscodeTemplateGroup(clie
nt);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印转码模板ID
                Console.WriteLine("TranscodeTemplateGroupId = " + response.TranscodeTemplat
eGroupId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 添加转码模板组
        /// </summary>
        /// <returns>转码模板组</returns>
        /// <param name="client">Client.</param>
        public static AddTranscodeTemplateGroupResponse AddTranscodeTemplateGroup(DefaultAc
sClient client)
        {
            // 构造请求
            AddTranscodeTemplateGroupRequest request = new AddTranscodeTemplateGroupRequest
();
            // 转码模板组名称
            request.Name = "grouptest";
            request.TranscodeTemplateList = BuildTranscodeTemplateList();
```

```
            // 返回结果
            return client.GetAcsResponse(request);

        }

        private static String BuildTranscodeTemplateList()
        {
            JArray transcodeTemplateList = new JArray();

            JObject transcodeTemplate = new JObject();
            //视频流转码配置
            JObject video = new JObject();
            video.Add("Width", 640);
            video.Add("Bitrate", 400);
            video.Add("Fps", "25");
            video.Add("Remove", false);
            video.Add("Codec", "H.264");
            video.Add("Gop", 250);
            transcodeTemplate.Add("Video", video);

            //音频流转码配置
            JObject audio = new JObject();
            audio.Add("Codec", "AAC");
            audio.Add("Bitrate", "64");
            audio.Add("Channels", "2");
            audio.Add("Samplerate", "32000");
            transcodeTemplate.Add("Audio", audio);

            //封装容器
            JObject container = new JObject();
            container.Add("Format", "mp4");
            transcodeTemplate.Add("Container", container);

            //条件转码配置
            JObject transconfig = new JObject();
            transconfig.Add("IsCheckReso", false);
            transconfig.Add("IsCheckResoFail", false);
            transconfig.Add("IsCheckVideoBitrate", false);
            transconfig.Add("IsCheckVideoBitrateFail", false);
            transconfig.Add("IsCheckAudioBitrate", false);
            transconfig.Add("IsCheckAudioBitrateFail", false);
            transcodeTemplate.Add("TransConfig", transconfig);

            //加密配置(只支持m3u8)
            //JObject encryptSetting = new JObject();
            //encryptSetting.Add("EncryptType", "Private");
            //transcodeTemplate.Add("EncryptSetting", encryptSetting);

            //清晰度
            transcodeTemplate.Add("Definition", "LD");
            //模板名称
            transcodeTemplate.Add("TemplateName", "testtemplate");
            //水印ID(多水印关联)
            //JArray watermarkIdList = new JArray();
```

```
            //watermarkIdList.Add("263261bdc1ff65782f8995c6dd22****");
            //USER_DEFAULT_WATERMARK 代表默认水印ID
            //watermarkIdList.Add("USER_DEFAULT_WATERMARK");
            //transcodeTemplate.Add("WatermarkIds", watermarkIdList);

            transcodeTemplateList.Add(transcodeTemplate);
            return transcodeTemplateList.ToString();
        }
    }
}
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.VoD.Sdk.UpdateTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateTranscodeTemplateGroupResponse response = UpdateTranscodeTemplateGrou
p(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印转码模版ID
                Console.WriteLine("TranscodeTemplateGroupId = " + response.TranscodeTemplat
eGroupId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
```

```
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 修改转码模板组
        /// </summary>
        /// <returns>转码模板组</returns>
        /// <param name="client">Client.</param>
        public static UpdateTranscodeTemplateGroupResponse UpdateTranscodeTemplateGroup(Def
aultAcsClient client)
        {
            // 构造请求
            UpdateTranscodeTemplateGroupRequest request = new UpdateTranscodeTemplateGroupR
equest();
            request.Name = "newgrouptest";
            // 转码模板组ID
            request.TranscodeTemplateGroupId = "4c71a339fecec0152b4fa6f452****";

            request.TranscodeTemplateList = BuildTranscodeTemplateList();

            // 返回结果
            return client.GetAcsResponse(request);

        }

        private static String BuildTranscodeTemplateList()
        {
            JArray transcodeTemplateList = new JArray();

            JObject transcodeTemplate = new JObject();
            //视频流转码配置
            JObject video = new JObject();
            video.Add("Width", 640);
            video.Add("Bitrate", 400);
            video.Add("Fps", "25");
            video.Add("Remove", false);
            video.Add("Codec", "H.264");
            video.Add("Gop", 250);
            transcodeTemplate.Add("Video", video);
```

```
            //音频流转码配置
            JObject audio = new JObject();
            audio.Add("Codec", "AAC");
            audio.Add("Bitrate", "64");
            audio.Add("Channels", "2");
            audio.Add("Samplerate", "32000");
            transcodeTemplate.Add("Audio", audio);

            //封装容器
            JObject container = new JObject();
            container.Add("Format", "mp4");
            transcodeTemplate.Add("Container", container);

            //条件转码配置
            JObject transconfig = new JObject();
            transconfig.Add("IsCheckReso", false);
            transconfig.Add("IsCheckResoFail", false);
            transconfig.Add("IsCheckVideoBitrate", false);
            transconfig.Add("IsCheckVideoBitrateFail", false);
            transconfig.Add("IsCheckAudioBitrate", false);
            transconfig.Add("IsCheckAudioBitrateFail", false);
            transcodeTemplate.Add("TransConfig", transconfig);

            //加密配置(只支持m3u8)
            //JObject encryptSetting = new JObject();
            //encryptSetting.Add("EncryptType", "Private");
            //transcodeTemplate.Add("EncryptSetting", encryptSetting);

            //清晰度
            transcodeTemplate.Add("Definition", "SD");
            //模板名称
            transcodeTemplate.Add("TemplateName", "testtemplate");
            //转码模板ID
            transcodeTemplate.Add("TranscodeTemplateId", "85c2b18ac08fda33e8f6d9c56****");
            //水印ID(多水印关联)
            //JArray watermarkIdList = new JArray();
            //watermarkIdList.Add("263261bdc1ff65782f8995c6dd22****");
            //USER_DEFAULT_WATERMARK 代表默认水印ID
            //watermarkIdList.Add("USER_DEFAULT_WATERMARK");
            //transcodeTemplate.Add("WatermarkIds", watermarkIdList);

            transcodeTemplateList.Add(transcodeTemplate);
            return transcodeTemplateList.ToString();
        }
    }
}
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
using System;
```

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.VoD.Sdk.ListTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListTranscodeTemplateGroupResponse response = ListTranscodeTemplateGroup(cl
ient);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板组信息
                if (response.TranscodeTemplateGroupList != null && response.TranscodeTempla
teGroupList.Count > 0)
                {
                    foreach (ListTranscodeTemplateGroupResponse.ListTranscodeTemplateGroup_
TranscodeTemplateGroup templateGroup in response.TranscodeTemplateGroupList)
                    {
                        Console.WriteLine("Name = " + templateGroup.Name);
                        Console.WriteLine("TranscodeTemplateGroupId = " + templateGroup.Tra
nscodeTemplateGroupId);
                        Console.WriteLine("IsDefault = " + templateGroup.IsDefault);
                        Console.WriteLine("CreationTime = " + templateGroup.CreationTime);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
```

```
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询转码模板组列表
        /// </summary>
        /// <returns>模板组列表</returns>
        /// <param name="client">Client.</param>
        public static ListTranscodeTemplateGroupResponse ListTranscodeTemplateGroup(Default
AcsClient client)
        {
            // 构造请求
            ListTranscodeTemplateGroupRequest request = new ListTranscodeTemplateGroupReque
st();
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.VoD.Sdk.GetTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
```

```
KeySecret> );
                // 发起请求，并得到 response
                GetTranscodeTemplateGroupResponse response = GetTranscodeTemplateGroup(clie
nt);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板组信息
                Console.WriteLine("Name = " + response.TranscodeTemplateGroup.Name);
                Console.WriteLine("TranscodeTemplateGroupId = " + response.TranscodeTemplat
eGroup.TranscodeTemplateGroupId);
                Console.WriteLine("IsDefault = " + response.TranscodeTemplateGroup.IsDefaul
t);
                Console.WriteLine("CreationTime = " + response.TranscodeTemplateGroup.Creat
ionTime);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询单个转码模板组
        /// </summary>
        /// <returns>模板组</returns>
        /// <param name="client">Client.</param>
        public static GetTranscodeTemplateGroupResponse GetTranscodeTemplateGroup(DefaultAc
sClient client)
        {
            // 构造请求
            GetTranscodeTemplateGroupRequest request = new GetTranscodeTemplateGroupRequest
();
            request.TranscodeTemplateGroupId = "94bcd9366be0df7ffba3b6f7b715****";
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
```

```
}
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.VoD.Sdk.SetDefaultTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SetDefaultTranscodeTemplateGroupResponse response = SetDefaultTranscodeTemp
lateGroup(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
```

```
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 设置默认转码模板组
        /// </summary>
        /// <returns>默认转码模板组</returns>
        /// <param name="client">Client.</param>
        public static SetDefaultTranscodeTemplateGroupResponse SetDefaultTranscodeTemplateG
roup(DefaultAcsClient client)
        {
            // 构造请求
            SetDefaultTranscodeTemplateGroupRequest request = new SetDefaultTranscodeTempla
teGroupRequest();
            // 设置模板ID
            request.TranscodeTemplateGroupId = "dc063078c1d845139e2a5bd8ff85****";
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.VoD.Sdk.DeleteTranscodeTemplateGroup
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteTranscodeTemplateGroupResponse response = DeleteTranscodeTemplateGrou
p(client);
                // 打印请求ID
```

```
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 删除转码模板组配置
        /// </summary>
        /// <param name="client">Client.</param>
        public static DeleteTranscodeTemplateGroupResponse DeleteTranscodeTemplateGroup(Def
aultAcsClient client)
        {
            // 构造请求
            DeleteTranscodeTemplateGroupRequest request = new DeleteTranscodeTemplateGroupR
equest();
            request.TranscodeTemplateGroupId = "dc063078c1d845139e2a5bd8ff65****";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

5.10. 截图模板

本篇文档提供了.NET SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.AddVodTemplate
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                AddVodTemplateResponse response = AddVodTemplate(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
```

```
        /// <summary>
        /// 添加截图模板函数
        /// </summary>
        /// <returns>The vod template.</returns>
        /// <param name="client">Client.</param>
        public static AddVodTemplateResponse AddVodTemplate(DefaultAcsClient client)
        {
            // 构造请求
            AddVodTemplateRequest request = new AddVodTemplateRequest();
            // 模板名称
            request.Name = "addVodTemplate";
            // 模板类型，固定值为Snapshot
            request.TemplateType = "Snapshot";
            // 截图模板配置数据生成
            request.TemplateConfig = BuildSnapshotTemplateConfig();

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
        /// （以下代码示例为雪碧图完整配置）
        /// </summary>
        /// <returns>截图模板的配置</returns>
        public static string BuildSnapshotTemplateConfig()
        {
            JObject templateConfig = new JObject();
            JObject snapshotConfig = new JObject();
            snapshotConfig.Add("Count", "50");
            snapshotConfig.Add("Interval", "1");
            snapshotConfig.Add("SpecifiedOffsetTime", "0");
            snapshotConfig.Add("Width", "200");
            snapshotConfig.Add("Height", "200");
            // 普通截图配置（与雪碧图原始图配置共用该配置信息）
            templateConfig.Add("SnapshotConfig", snapshotConfig);
            //雪碧图配置（雪碧图配置必须是建立在普通截图配置之上）
            JObject spriteSnapshotConfig = new JObject();
            spriteSnapshotConfig.Add("CellWidth", "120");
            spriteSnapshotConfig.Add("CellHeight", "68");
            spriteSnapshotConfig.Add("Columns", "3");
            spriteSnapshotConfig.Add("Lines", "10");
            spriteSnapshotConfig.Add("Padding", "20");
            spriteSnapshotConfig.Add("Margin", "50");
            spriteSnapshotConfig.Add("KeepCellPic", "keep");
            spriteSnapshotConfig.Add("Color", "tomato");
            snapshotConfig.Add("SpriteSnapshotConfig", spriteSnapshotConfig);

            //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
            templateConfig.Add("SnapshotType", "SpriteSnapshot");

            return templateConfig.ToString();
        }
    }
}
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.UpdateVodTemplate
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateVodTemplateResponse response = UpdateVodTemplate(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
```

```
        }

        /// <summary>
        /// 修改截图模板函数
        /// </summary>
        /// <returns>UpdateVodTemplateResponse</returns>
        /// <param name="client">Client.</param>
        public static UpdateVodTemplateResponse UpdateVodTemplate(DefaultAcsClient client)
        {
            // 构造请求
            UpdateVodTemplateRequest request = new UpdateVodTemplateRequest();
            request.VodTemplateId = "53azf9d796fad9d7b862b2e****";
            // 模板名称
            request.Name = "new template name";

            // 截图模板配置数据生成
            request.TemplateConfig = BuildSnapshotTemplateConfig();

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
        /// （以下代码示例为普通截图完整配置）
        /// </summary>
        /// <returns>截图模板的配置</returns>
        public static string BuildSnapshotTemplateConfig()
        {
            JObject templateConfig = new JObject();
            JObject snapshotConfig = new JObject();
            snapshotConfig.Add("Count", "50");
            snapshotConfig.Add("Interval", "1");
            snapshotConfig.Add("SpecifiedOffsetTime", "0");
            snapshotConfig.Add("Width", "200");
            snapshotConfig.Add("Height", "200");
            // 普通截图配置（与雪碧图原始图配置共用该配置信息）
            templateConfig.Add("SnapshotConfig", snapshotConfig);

            //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
            templateConfig.Add("SnapshotType", "SpriteSnapshot");

            return templateConfig.ToString();
        }
    }
}
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
```

```
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteVodTemplate
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteVodTemplateResponse response = DeleteVodTemplate(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Deletes the vod template.
        /// </summary>
        /// <returns>The vod template.</returns>
        /// <param name="client">Client.</param>
        public static DeleteVodTemplateResponse DeleteVodTemplate(DefaultAcsClient client)
        {
            // 构造请求
            DeleteVodTemplateRequest request = new DeleteVodTemplateRequest();
```

```
            //设置要删除的模板ID
            request.VodTemplateId = "53azf9d796fad9d7b862b2e****";

            return client.GetAcsResponse(request);
        }
    }
}
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListVodTemplate
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListVodTemplateResponse response = ListVodTemplate(client);

                if (response.VodTemplateInfoList != null && response.VodTemplateInfoList.Co
unt > 0)
                {
                    foreach (ListVodTemplateResponse.ListVodTemplate_VodTemplateInfo templa
teInfo in response.VodTemplateInfoList)
                    {
                        // 模板ID
                        Console.WriteLine("VodTemplateId = " + templateInfo.VodTemplateId);
                        // 模板名称
                        Console.WriteLine("Name = " + templateInfo.Name);
                    }
                }
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
```

```
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Lists the vod template.
        /// </summary>
        /// <returns>The vod template.</returns>
        /// <param name="client">Client.</param>
        public static ListVodTemplateResponse ListVodTemplate(DefaultAcsClient client)
        {
            // 构造请求
            ListVodTemplateRequest request = new ListVodTemplateRequest();
            //模板类型，固定值为Snapshot
            request.TemplateType = "Snapshot";

            return client.GetAcsResponse(request);
        }
    }
}
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetVodTemplate
{
    class MainClass
    {
        /// <summary>
```

```
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetVodTemplateResponse response = GetVodTemplate(client);
                if (response.VodTemplateInfo != null)
                {
                    // 模板ID
                    Console.WriteLine("VodTemplateId = " + templateInfo.VodTemplateId);
                    // 模板名称
                    Console.WriteLine("Name = " + templateInfo.Name);
                }
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Gets the vod template.
        /// </summary>
        /// <returns>The vod template.</returns>
        /// <param name="client">Client.</param>
        public static GetVodTemplateResponse GetVodTemplate(DefaultAcsClient client)
        {
            // 构造请求
            GetVodTemplateRequest request = new GetVodTemplateRequest();
            // 设置要查询的模板ID
```

```
            request.VodTemplateId = "53azf9d796fad9d7b862b2e****";

            return client.GetAcsResponse(request);
        }
    }
}
```

5.11. 视频剪辑

本篇文档提供了.NET SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

> **说明**
>
> 通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.ProduceEditingProjectVideo
{
    class MainClass
    {
         /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                ProduceEditingProjectVideoRequest request = new ProduceEditingProjectVideoR
equest();
                // 构造剪辑工程时间线
                request.Timeline = BuildTimeline();
                // 设置元数据
                request.MediaMetadata = BuildMediaMetadata();
                // 设置工程配置
```

```
                request.ProduceConfig = BuildProduceConfig();
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ProduceEditingProjectVideoResponse response = client.GetAcsResponse(request
);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("MediaId = " + response.MediaId);
                Console.WriteLine("ProjectId = " + response.ProjectId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Builds the timeline.
        /// </summary>
        /// <returns>The timeline.</returns>
        private static string BuildTimeline() {
            JObject timeLine = new JObject();
            // Video Track
            JArray videoTracks = new JArray();
            JObject videoTrack = new JObject();

            // Video Track Clips
            JArray videoTrackClips = new JArray();
            JObject videoTrackClip1 = new JObject();
            videoTrackClip1.Add("MediaId", "11119b4d7cf14dc7b83b0e801cbe1ce6");
            videoTrackClips.Add(videoTrackClip1);

            JObject videoTrackClip2 = new JObject();
            videoTrackClip2.Add("MediaId", "22229b4d7cf14dc7b83b0e801cbe1ce6");
            videoTrackClips.Add(videoTrackClip2);
            videoTrack.Add("VideoTrackClips", videoTrackClips);
```

```
            videoTrack.Add("VideoTrackClips", videoTrackClips);
            videoTracks.Add(videoTrack);

            timeLine.Add("VideoTracks", videoTracks);

            return timeLine.ToString();
        }

        /// <summary>
        /// Builds the media metadata.
        /// </summary>
        /// <returns>The media metadata.</returns>
        private static string BuildMediaMetadata() {
            JObject mediaMetadata = new JObject();
            // Produce Media Title
            mediaMetadata.Add("Title", "Title");
            // Produce Media Description
            mediaMetadata.Add("Description", "Description");
            // Produce Media UserDefined Cover URL
            //Cover URL示例：http://example.aliyundoc.com/media/cover/mediaid****.jpg
            mediaMetadata.Add("CoverURL", "<your Cover URL>");
            // Produce Media Category ID
            mediaMetadata.Add("CateId", null);
            // Produce Media Category Name
            mediaMetadata.Add("Tags", "Tag1,Tag2,Test");

            return mediaMetadata.ToString();
        }

        /// <summary>
        /// Builds the produce config.
        /// </summary>
        /// <returns>The produce config.</returns>
        private static string BuildProduceConfig() {
            JObject produceConfig = new JObject();
            /**
             * The produce process can generate media mezzanine file. You can use the mezza
nine file to transcode other media files, just like the transcode process after file upload
finished. This field describe the Transocde TemplateGroup ID after produce mezzanine finish
ed.
             * 1. Not required
             * 2. Use default transcode template group id when empty
            **/
            produceConfig.Add("TemplateGroupId", null);
            return produceConfig.ToString();
        }
    }
}
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.ProduceEditingProjectVideoProjectId
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            ProduceEditingProjectVideoResponse response = null;
            try
            {
                // 构造请求
                ProduceEditingProjectVideoRequest request = new ProduceEditingProjectVideoR
equest();
                // 构造剪辑工程时间线
                request.ProjectId = "ProjectId";
                // 设置元数据
                request.MediaMetadata = BuildMediaMetadata();
                // 设置工程配置
                request.ProduceConfig = BuildProduceConfig();
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                Console.WriteLine("MediaId = " + response.MediaId);
                Console.WriteLine("ProjectId = " + response.ProjectId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
```

```
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Builds the media metadata.
        /// </summary>
        /// <returns>The media metadata.</returns>
        private static string BuildMediaMetadata()
        {
            JObject mediaMetadata = new JObject();

            // Produce Media Title
            mediaMetadata.Add("Title", "Title");
            // Produce Media Description
            mediaMetadata.Add("Description", "Description");
            // Produce Media UserDefined Cover URL
            //Cover URL示例：http://example.aliyundoc.com/media/cover/mediaid****.jpg
            mediaMetadata.Add("CoverURL", "<your Cover URL>");
            // Produce Media Category ID
            mediaMetadata.Add("CateId", null);
            // Produce Media Category Name
            mediaMetadata.Add("Tags", "Tag1,Tag2,Test");

            return mediaMetadata.ToString();
        }

        /// <summary>
        /// Builds the produce config.
        /// </summary>
        /// <returns>The produce config.</returns>
        private static string BuildProduceConfig()
        {
            JObject produceConfig = new JObject();
            /**
             * The produce process can generate media mezzanine file. You can use the mezza
nine file to transcode other media files, just like the transcode process after file upload
finished. This field describe the Transocde TemplateGroup ID after produce mezzanine finish
ed.
             * 1. Not required
             * 2. Use default transcode template group id when empty
            **/
            produceConfig.Add("TemplateGroupId", null);

            return produceConfig.ToString();
        }
    }
}
```

```
}
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.AddEditingProject
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                AddEditingProjectRequest request = new AddEditingProjectRequest();
                request.Description = "new Description";
                request.Title = "new Title";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                AddEditingProjectResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.Project != null) {
                    Console.WriteLine("ProjectId = " + response.Project.ProjectId);
                    Console.WriteLine("Title = " + response.Project.Title);
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.UpdateEditingProject
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                UpdateEditingProjectRequest request = new UpdateEditingProjectRequest();
                request.ProjectId = "YourProjectId";
                request.Title = "new Title";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateEditingProjectResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteEditingProject
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                DeleteEditingProjectRequest request = new DeleteEditingProjectRequest();
                // Use comma to split Multi Editing Project IDs
                request.ProjectIds = "projectid1,projectid2";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteEditingProjectResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetEditingProject
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetEditingProjectRequest request = new GetEditingProjectRequest();
                request.ProjectId = "projectid";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetEditingProjectResponse response = client.GetAcsResponse(request);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.Project != null)
                {
                    Console.WriteLine("ProjectId = " + response.Project.ProjectId);
                    Console.WriteLine("Title = " + response.Project.Title);
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SearchEditingProject
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                SearchEditingProjectRequest request = new SearchEditingProjectRequest();
                request.Title = "Title Keywords";
                request.StartTime = "2017-01-11T12:00:00Z";
                request.EndTime = "2017-01-12T12:00:00Z";
                request.PageSize = 10;
                request.PageNo = 1;
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SearchEditingProjectResponse response = client.GetAcsResponse(request);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.ProjectList != null && response.ProjectList.Count > 0)
                {
                    foreach (var project in response.ProjectList)
                    {
                        Console.WriteLine("ProjectId = " + project.ProjectId);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
```

```
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SetEditingProjectMaterials
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                SetEditingProjectMaterialsRequest request = new SetEditingProjectMaterialsR
equest();
                // Editing Project ID
                request.ProjectId = "projectid";
                // Set Editing Project Material IDs, use comma to split
                request.MaterialIds = "materialId1,materialId2";

                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SetEditingProjectMaterialsResponse response = client.GetAcsResponse(request
);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
```

```
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

Aliyun.Acs.vod.Sdk.GetEditingProjectMaterials
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetEditingProjectMaterialsRequest request = new GetEditingProjectMaterialsR
equest();
                request.ProjectId = "projectid";
                request.Type = "video";

                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetEditingProjectMaterialsResponse response = client.GetAcsResponse(request
);
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.MaterialList != null && response.MaterialList.Count > 0)
                {
                    foreach (var material in response.MaterialList)
```

```
                foreach (var material in response.MaterialList)
                {
                    Console.WriteLine("MaterialId = " + material.MaterialId);
                    Console.WriteLine("Title = " + material.Title);
                }
            }
        }
        catch (ServerException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (ClientException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (Exception e)
        {
            Console.WriteLine("ErrorMessage = " + e.ToString());
        }
    }
  }
}
```

5.12. 媒体审核

本篇文档提供了.NET SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SubmitAIMediaAuditJob
```

```
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitAIMediaAuditJobResponse response = SubmitAIMediaAuditJob(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 提交智能审核作业
        /// </summary>
        /// <param name="client">Client.</param>
        /// <summary>
        /// 提交智能审核作业.
        /// </summary>
        /// <returns>智能审核作业.</returns>
        /// <param name="client">Client.</param>
        public static SubmitAIMediaAuditJobResponse SubmitAIMediaAuditJob(DefaultAcsClient
client)
        {
        // 构造请求
```

```
        // 构造请求
        SubmitAIMediaAuditJobRequest request = new SubmitAIMediaAuditJobRequest();
        // 设置视频ID
        request.MediaId = "dc063078c1d845139e2a5bd8ff89****";
        // 返回结果
        return client.GetAcsResponse(request);
        }
    }
}
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetAIMediaAuditJob
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetAIMediaAuditJobResponse response = GetAIMediaAuditJob(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印结果信息
                Console.WriteLine("MediaId = " + response.MediaAuditJob.MediaId);
                Console.WriteLine("JobId = " + response.MediaAuditJob.JobId);
                Console.WriteLine("Type = " + response.MediaAuditJob.Type);
                Console.WriteLine("Status = " + response.MediaAuditJob.Status);
                Console.WriteLine("Code = " + response.MediaAuditJob.Code);
                Console.WriteLine("Message = " + response.MediaAuditJob.Message);
                Console.WriteLine("Data AbnormalModules = " + response.MediaAuditJob.Data.A
bnormalModules);
                Console.WriteLine("Data Label = " + response.MediaAuditJob.Data.Label);
                Console.WriteLine("Data Suggestion = " + response.MediaAuditJob.Data.Sugges
tion);
            }
            catch (ServerException e)
```

```
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询智能审核作业.
        /// </summary>
        /// <returns>智能审核作业.</returns>
        /// <param name="client">Client.</param>
        public static GetAIMediaAuditJobResponse GetAIMediaAuditJob(DefaultAcsClient client
)
        {
            // 构造请求
            GetAIMediaAuditJobRequest request = new GetAIMediaAuditJobRequest();
            // 设置作业ID
            request.JobId = "7dc69b893c8b4f13b47aae8de059****";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
```

```
namespace Aliyun.Acs.vod.Sdk.GetMediaAuditResult
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetMediaAuditResultResponse response = GetMediaAuditResult(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印结果信息
                Console.WriteLine("Data AbnormalModules = " + response.MediaAuditResult.Abn
ormalModules);
                Console.WriteLine("Data Label = " + response.MediaAuditResult.Label);
                Console.WriteLine("Data Suggestion = " + response.MediaAuditResult.Suggesti
on);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 获取智能审核结果摘要
        /// </summary>
        /// <returns>智能审核结果摘要</returns>
```

```
        /// <param name="client">Client.</param>
        public static GetMediaAuditResultResponse GetMediaAuditResult(DefaultAcsClient clie
nt)
        {
            // 构造请求
            GetMediaAuditResultRequest request = new GetMediaAuditResultRequest();
            // 设置视频ID
            request.MediaId = "7dc69b893c8b4f13b47aae8de053****";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetMediaAuditResultDetail
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetMediaAuditResultDetailResponse response = GetMediaAuditResultDetail(clie
nt);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印结果信息
                Console.WriteLine("Data Total = " + response.MediaAuditResultDetail.Total);
                Console.WriteLine("Data List Size = " + response.MediaAuditResultDetail.Lis
t.Count);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
```

```
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 获取智能审核结果详情
        /// </summary>
        /// <returns>智能审核结果详情</returns>
        /// <param name="client">Client.</param>
        public static GetMediaAuditResultDetailResponse GetMediaAuditResultDetail(DefaultAc
sClient client)
        {
            // 构造请求
            GetMediaAuditResultDetailRequest request = new GetMediaAuditResultDetailRequest
();
            // 设置视频ID
            request.MediaId = "7dc69b893c8b4f13b47aae8de0565****";
            // 设置翻页页数
            request.PageNo = 1;
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
```

```
namespace Aliyun.Acs.vod.Sdk.GetMediaAuditResultTimeline
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>","<Your AccessK
eySecret>");
                // 发起请求，并得到 response
                GetMediaAuditResultTimelineResponse response = GetMediaAuditResultTimeline(
client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印结果信息
                Console.WriteLine("Terrorism = " + response.MediaAuditResultTimeline.Terror
ism);
                Console.WriteLine("Porn = " + response.MediaAuditResultTimeline.Porn);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 获取智能审核结果时间线
        /// </summary>
        /// <returns>智能审核结果时间线</returns>
```

```
        /// <param name="client">Client.</param>
        public static GetMediaAuditResultTimelineResponse GetMediaAuditResultTimeline(Defau
ltAcsClient client)
        {
            // 构造请求
            GetMediaAuditResultTimelineRequest request = new GetMediaAuditResultTimelineReq
uest();
            // 设置视频ID
            request.MediaId = "7dc69b893c8b4f13b47aae8de036****";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.CreateAudit
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                CreateAuditResponse response = CreateAudit(client);
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
```

```
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Creates the audit.
        /// </summary>
        /// <returns>The audit.</returns>
        /// <param name="client">Client.</param>
        public static CreateAuditResponse CreateAudit(DefaultAcsClient client)
        {
            CreateAuditRequest request = new CreateAuditRequest();
            // set audit content
            request.AuditContent = BuildAuditContent();

            return client.GetAcsResponse(request);
        }

        /// <summary>
        /// Builds the content of the audit.
        /// </summary>
        /// <returns>The audit content.</returns>
        public static string BuildAuditContent()
        {
            JArray auditContents = new JArray();
            JObject auditContent = new JObject();
            auditContent.Add("VideoId", "3ebc10160bda481ca9b6858a0b65****");
            auditContent.Add("Status", "Blocked");
            auditContent.Add("Reason", "含有淫秽色情内容");

            return auditContents.ToString();
        }
    }
}
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
using System;
```

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetAuditHistory
{
  class MainClass
  {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetAuditHistoryResponse response = GetAuditHistory(client);
                Console.WriteLine("RequestId = " + response.RequestId);
                // 最新审核结果
                Console.WriteLine("Status = " + response.Status);
                // 审核历史记录数
                Console.WriteLine("Total = " + response.Total);

                if (response.Histories != null && response.Histories.Count > 0)
                {
                    foreach (GetAuditHistoryResponse.GetAuditHistory_History history in res
ponse.Histories) {
                        Console.WriteLine("Status = " + history.Status);
                        Console.WriteLine("CreationTime = " + history.CreationTime);
                        Console.WriteLine("Reason = " + history.Reason);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
```

```
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Gets the audit history.
        /// </summary>
        /// <returns>The audit history.</returns>
        /// <param name="client">Client.</param>
        public static GetAuditHistoryResponse GetAuditHistory(DefaultAcsClient client) {
            // 构造请求
            GetAuditHistoryRequest request = new GetAuditHistoryRequest();
            request.VideoId = "VideoId";
            request.PageNo = 1;
            request.PageSize = 10;

            return client.GetAcsResponse(request);
        }
    }
}
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SetAuditSecurityIp
{
    class MainClass
    {
        /// <summary>
        /// 主函数
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");

                SetAuditSecurityIpResponse response = SetAuditSecurityIp(client);
```

```
                Console.WriteLine("RequestId = " + response.RequestId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
        /// <summary>
        /// Sets the audit security ip.
        /// </summary>
        /// <returns>The audit security ip.</returns>
        /// <param name="client">Client.</param>
        public static SetAuditSecurityIpResponse SetAuditSecurityIp (DefaultAcsClient clien
t)
        {
            // 构造请求
            SetAuditSecurityIpRequest request = new SetAuditSecurityIpRequest();
            // 审核安全组名称，默认为Default
            request.SecurityGroupName = "MyGroupName";
            // 操作方式，默认值Append(追加IP白名单)
            request.OperateMode = "Cover";
            // 审核安全IP列表
            request.Ips = "10.23.12.20,10.23.12.21,10.23.12.22";

            return client.GetAcsResponse(request);
        }
    }
}
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
```

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListAuditSecurityIp
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListAuditSecurityIpResponse response = ListAuditSecurityIp(client);

                Console.WriteLine("RequestId = " + response.RequestId);

                if (response.SecurityIpList != null && response.SecurityIpList.Count > 0)
                {
                    foreach (ListAuditSecurityIpResponse.ListAuditSecurityIp_SecurityIp sec
urityIp in response.SecurityIpList)
                    {
                        Console.WriteLine("SecurityGroupName = " + securityIp.SecurityGroup
Name);
                        Console.WriteLine("Ips = " + securityIp.Ips);
                        Console.WriteLine("CreationTime = " + securityIp.CreationTime);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
```

```
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// Lists the audit security ip.
        /// </summary>
        /// <returns>The audit security ip.</returns>
        /// <param name="client">Client.</param>
        public static ListAuditSecurityIpResponse ListAuditSecurityIp(DefaultAcsClient clie
nt)
        {
            // 构造请求
            ListAuditSecurityIpRequest request = new ListAuditSecurityIpRequest();
            // 审核安全组名称
            request.SecurityGroupName = "MyGroupName";

            return client.GetAcsResponse(request);
        }
    }
}
```

5.13. 视频AI

本篇文档提供了.NET SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SubmitAIJob
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                SubmitAIJobRequest request = new SubmitAIJobRequest();
                // 设置视频ID
```

```
                request.MediaId = "3eb19a4585bc475e995bdd456****";
                // 设置AI类型，请确保已开通该类型AI
                request.Types = "AIVideoCover";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitAIJobResponse response = client.GetAcsResponse(request);
                // 请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.AIJobList != null && response.AIJobList.Count > 0)
                {
                    foreach (var job in response.AIJobList)
                    {
                        // 视频ID
                        Console.WriteLine("MediaId = " + job.MediaId);
                        // 作业ID
                        Console.WriteLine("JobId = " + job.JobId);
                        // AI类型
                        Console.WriteLine("Type = " + job.Type);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
onsole.WriteLine("JobId = " + jobId)using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListAIJob
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                ListAIJobRequest request = new ListAIJobRequest();
                // 设置视频ID
                request.JobIds = "979d4d7a36ae41b1a945a258****,3eb19a4585bc475e995bddea56**
**";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListAIJobResponse response = client.GetAcsResponse(request);
                // 请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印作业列表
                if (response.AIJobList != null && response.AIJobList.Count > 0)
                {
                    foreach (var job in response.AIJobList)
                    {
                        // 视频ID
                        Console.WriteLine("MediaId = " + job.MediaId);
                        // 作业ID
                        Console.WriteLine("JobId = " + job.JobId);
                        // AI类型
                        Console.WriteLine("Type = " + job.Type);
                        // 作业状态
                        Console.WriteLine("Status = " + job.Status);
                        // 作业结果
                        Console.WriteLine("Data = " + job.Data);
                    }
                }
                // 打印不存在的作业ID
                if (response.NonExistAIJobIds != null && response.NonExistAIJobIds.Count >
0)
                {
                    foreach(jobId in response.NonExistAIJobIds)
                    {
                        // 作业ID
                        C;
                    }
                }
            }
            catch (ServerException e)
```

```
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.AddAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
```

```
            AddAITemplateResponse response = AddAITemplate(client);
            // 打印请求ID
            Console.WriteLine("RequestId = " + response.RequestId);
            // 打印模板ID
            Console.WriteLine("TemplateId = " + response.TemplateId);
        }
        catch (ServerException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (ClientException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (Exception e)
        {
            Console.WriteLine("ErrorMessage = " + e.ToString());
        }
    }

    /// <summary>
    /// 添加AI模板
    /// </summary>
    /// <param name="client">Client.</param>
    public static AddAITemplateResponse AddAITemplate(DefaultAcsClient client)
    {
        // 构造请求
        AddAITemplateRequest request = new AddAITemplateRequest();
        // 设置模板类型，以智能审核模板为例
        request.TemplateType = "AIMediaAudit";
        // 设置模板名称
        request.TemplateName = "My custom template";
        // 设置模板详细配置
        JObject templateConfig = new JObject();

        JArray auditItem = new JArray();
        auditItem.Add("terrorism");
        auditItem.Add("porn");
        templateConfig.Add("AuditItem", auditItem);

        JArray auditRange = new JArray();
        auditRange.Add("video");
        auditRange.Add("image-cover");
        auditRange.Add("text-title");
```

```
            templateConfig.Add("AuditRange", auditRange);

            JArray auditContent = new JArray();
            auditContent.Add("screen");
            templateConfig.Add("AuditContent", auditContent);

            templateConfig.Add("AuditAutoBlock", "no");

            request.TemplateConfig = templateConfig.ToString();
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.DeleteAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                DeleteAITemplateResponse response = DeleteAITemplate(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板ID
                Console.WriteLine("TemplateId = " + response.TemplateId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
```

```
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 删除AI模板
        /// </summary>
        /// <returns>删除AI模板</returns>
        /// <param name="client">Client.</param>
        public static DeleteAITemplateResponse DeleteAITemplate(DefaultAcsClient client)
        {
            // 构造请求
            DeleteAITemplateRequest request = new DeleteAITemplateRequest();
            // 设置模板ID
            request.TemplateId = "dc063078c1d845139e2a5bd8ff85****";
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;
using Newtonsoft.Json.Linq;

namespace Aliyun.Acs.vod.Sdk.UpdateAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
```

```
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                UpdateAITemplateResponse response = UpdateAITemplate(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板ID
                Console.WriteLine("TemplateId = " + response.TemplateId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 修改AI模板
        /// </summary>
        /// <param name="client">Client.</param>
        public static UpdateAITemplateResponse UpdateAITemplate(DefaultAcsClient client)
        {
            // 构造请求
            UpdateAITemplateRequest request = new UpdateAITemplateRequest();
            // 设置模板名称
            request.TemplateName = "My custom template";
            // 设置模板详细配置
            JObject templateConfig = new JObject();

            JArray auditItem = new JArray();
```

```
            //auditItem.Add("terrorism");
            auditItem.Add("porn");
            templateConfig.Add("AuditItem", auditItem);

            JArray auditRange = new JArray();
            auditRange.Add("video");
            auditRange.Add("image-cover");
            auditRange.Add("text-title");
            templateConfig.Add("AuditRange", auditRange);

            JArray auditContent = new JArray();
            auditContent.Add("screen");
            templateConfig.Add("AuditContent", auditContent);

            templateConfig.Add("AuditAutoBlock", "no");

            request.TemplateConfig = templateConfig.ToString();
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.GetAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetAITemplateResponse response = GetAITemplate(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板信息
```

```
                // 打印模板信息
                Console.WriteLine("TemplateId = " + response.TemplateInfo.TemplateId);
                Console.WriteLine("TemplateType = " + response.TemplateInfo.TemplateType);
                Console.WriteLine("TemplateName = " + response.TemplateInfo.TemplateName);
                Console.WriteLine("TemplateConfig = " + response.TemplateInfo.TemplateConfi
g);

            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询AI模板
        /// </summary>
        /// <returns>AI模板</returns>
        /// <param name="client">Client.</param>
        public static GetAITemplateResponse GetAITemplate(DefaultAcsClient client)
        {
            // 构造请求
            GetAITemplateRequest request = new GetAITemplateRequest();
            // 设置模板ID
            request.TemplateId = "dc063078c1d845139e2a5bd8ff99****";
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListAITemplateResponse response = ListAITemplate(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板信息
                if (response.TemplateInfoList != null && response.TemplateInfoList.Count >
0) {
                    foreach (ListAITemplateResponse.ListAITemplate_TemplateInfoListItem tem
plateInfo in response.TemplateInfoList) {
                        Console.WriteLine("TemplateId = " + templateInfo.TemplateId);
                        Console.WriteLine("TemplateType = " + templateInfo.TemplateType);
                        Console.WriteLine("TemplateName = " + templateInfo.TemplateName);
                        Console.WriteLine("TemplateConfig = " + templateInfo.TemplateConfig
);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
```

```
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 查询AI模板列表
        /// </summary>
        /// <returns>AI模板列表</returns>
        /// <param name="client">Client.</param>
        public static ListAITemplateResponse ListAITemplate(DefaultAcsClient client)
        {
            // 构造请求
            ListAITemplateRequest request = new ListAITemplateRequest();
            // 设置模板类型，以智能审核模板为例
            request.TemplateType = "AIMediaAudit";
            // 返回结果
            return client.GetAcsResponse(request);
        }
    }
}
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace SetDefaultAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SetDefaultAITemplateResponse response = SetDefaultAITemplate(client);
                // 打印请求ID
```

```
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板ID
                Console.WriteLine("TemplateId = " + response.TemplateId);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }

        /// <summary>
        /// 设置默认AI模板
        /// </summary>
        /// <returns>默认AI模板</returns>
        /// <param name="client">Client.</param>
        public static SetDefaultAITemplateResponse SetDefaultAITemplate(DefaultAcsClient cl
ient)
        {
            // 构造请求
            SetDefaultAITemplateRequest request = new SetDefaultAITemplateRequest();
            // 设置模板ID
            request.TemplateId = "dc063078c1d845139e2a5bd8ff85****";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
using System;
```

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.vod.Model.V20170321;

namespace GetDefaultAITemplate
{
    class MainClass
    {
        /// <summary>
        /// The entry point of the program, where the program control starts and ends.
        /// </summary>
        /// <param name="args">The command-line arguments.</param>
        public static void Main(string[] args)
        {
            try
            {
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetDefaultAITemplateResponse response = GetDefaultAITemplate(client);
                // 打印请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印模板信息
                Console.WriteLine("TemplateId = " + response.TemplateInfo.TemplateId);
                Console.WriteLine("TemplateType = " + response.TemplateInfo.TemplateType);
                Console.WriteLine("TemplateName = " + response.TemplateInfo.TemplateName);
                Console.WriteLine("TemplateConfig = " + response.TemplateInfo.TemplateConfi
g);
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
```

```
        /// <summary>
        /// 查询默认AI模板
        /// </summary>
        /// <returns>默认AI模板</returns>
        /// <param name="client">Client.</param>
        public static GetDefaultAITemplateResponse GetDefaultAITemplate(DefaultAcsClient cl
ient)
        {
            // 构造请求
            GetDefaultAITemplateRequest request = new GetDefaultAITemplateRequest();
            // 设置模板类型，以智能审核模板为例
            request.TemplateType = "AIMediaAudit";
            // 返回结果
            return client.GetAcsResponse(request);

        }
    }
}
```

5.14. 视频DNA

本文为您提供了.NET SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.SubmitAIJob
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                SubmitAIJobRequest request = new SubmitAIJobRequest();
                // 设置视频ID
                request.MediaId = "3eb19a4585bc475e995bdd7894****";
                // 设置AI类型，类型为AIMediaDNA
                request.Types = "AIMediaDNA";
                // 初始化客户端
```

```
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                SubmitAIJobResponse response = client.GetAcsResponse(request);
                // 请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.AIJobList != null && response.AIJobList.Count > 0)
                {
                    foreach (var job in response.AIJobList)
                    {
                        // 视频ID
                        Console.WriteLine("MediaId = " + job.MediaId);
                        // 作业ID
                        Console.WriteLine("JobId = " + job.JobId);
                        // AI类型
                        Console.WriteLine("Type = " + job.Type);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
```

```
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListAIJob
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                ListAIJobRequest request = new ListAIJobRequest();
                // 设置视频ID
                request.JobIds = "979d4d7a36ae41b1a945a258****,3eb19a4585bc475e995bddea58**
**";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListAIJobResponse response = client.GetAcsResponse(request);
                // 请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                // 打印作业列表
                if (response.AIJobList != null && response.AIJobList.Count > 0)
                {
                    foreach (var job in response.AIJobList)
                    {
                        // 视频ID
                        Console.WriteLine("MediaId = " + job.MediaId);
                        // 作业ID
                        Console.WriteLine("JobId = " + job.JobId);
                        // AI类型
                        Console.WriteLine("Type = " + job.Type);
                        // 作业状态
                        Console.WriteLine("Status = " + job.Status);
                        // 作业结果
                        Console.WriteLine("Data = " + job.Data);
                    }
                }
                // 打印不存在的作业ID
                if (response.NonExistAIJobIds != null && response.NonExistAIJobIds.Count >
0)
                {
                    foreach(jobId in response.NonExistAIJobIds)
                    {
                        // 作业ID
                        Console.WriteLine("JobId = " + jobId);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
```

```
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace GetMediaDNAResult
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            try
            {
                // 构造请求
                GetMediaDNAResultRequest request = new GetMediaDNAResultRequest();
                // 设置视频ID
                request.MediaId = "3eb19a4585bc475e995bdd62eaee0****";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                GetMediaDNAResultResponse response = client.GetAcsResponse(request);
                // 请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.DNAResult != null)
                {
                    // 打印DNA结果
```

```
                // 打印DNA结果
                Console.WriteLine("DNAResult = " + response.DNAResult.ToString());
            }
        }
        catch (ServerException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (ClientException e)
        {
            if (e.RequestId != null)
            {
                Console.WriteLine("RequestId = " + e.RequestId);
            }
            Console.WriteLine("ErrorCode = " + e.ErrorCode);
            Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
        }
        catch (Exception e)
        {
            Console.WriteLine("ErrorMessage = " + e.ToString());
        }
    }
  }
}
```

5.15. 直播转点播

本篇文档提供了.NET SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
using System;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.vod.Model.V20170321;

namespace Aliyun.Acs.vod.Sdk.ListLiveRecordVideo
{
    class MainClass
    {
        public static void Main(string[] args)
        {
```

```
            try
            {
                // 构造请求
                ListLiveRecordVideoRequest request = new ListLiveRecordVideoRequest();
                request.StartTime = "2018-04-24T03:21:04Z";
                request.EndTime = "2018-05-21T03:21:44Z";
                request.StreamName = "testStreamName";
                request.DomainName = "testdomain.aliyun.com";
                request.AppName = "testAppName";
                // 初始化客户端
                DefaultAcsClient client = InitVodClient("<Your AccessKeyId>", "<Your Access
KeySecret>");
                // 发起请求，并得到 response
                ListLiveRecordVideoResponse response = client.GetAcsResponse(request);
                // 打印本次请求ID
                Console.WriteLine("RequestId = " + response.RequestId);
                if (response.LiveRecordVideoList != null && response.LiveRecordVideoList.Co
unt > 0)
                {
                    //打印查询出来的列表个数
                    Console.WriteLine("Total = " + response.Total);
                    foreach (var liveRecordVideo in response.LiveRecordVideoList)
                    {
                        Console.WriteLine("AppName = " + liveRecordVideo.AppName);
                        Console.WriteLine("DomainName = " + liveRecordVideo.DomainName);
                        Console.WriteLine("PlaylistId = " + liveRecordVideo.PlaylistId);
                    }
                }
            }
            catch (ServerException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (ClientException e)
            {
                if (e.RequestId != null)
                {
                    Console.WriteLine("RequestId = " + e.RequestId);
                }
                Console.WriteLine("ErrorCode = " + e.ErrorCode);
                Console.WriteLine("ErrorMessage = " + e.ErrorMessage);
            }
            catch (Exception e)
            {
                Console.WriteLine("ErrorMessage = " + e.ToString());
            }
        }
    }
}
```

6.Node.js SDK

6.1. 安装

本文为您介绍视频点播服务端Node.js SDK的安装及更新步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

安装Node.js请参见Node.js官网，适用于视频点播Node.js SDK的Node.js版本如下：

- Node.js 4.x
- Node.js 6.x

您可以通过执行命令 `node-v` 查看Node.js的版本。

安装SDK

建议您使用 `npm` 来完成Node.js依赖模块的安装，可到npm官网下载合适的版本安装。所有阿里云官方的Node.js SDK都位于 `@alicloud` 下。

- 假设Node.js SDK下载后的路径为*/path/to/aliyun-openapi-Node.js-sdk*。当基于SDK核心库进行开发时，请执行以下命令安装 `@alicloud/pop-core` 模块。示例如下：

```
npm install @alicloud/pop-core --save
```

- 命令中的 `--save` 会将模块写入应用的*package.json*文件中，作为依赖模块。

更新SDK

若发现新的功能在当前SDK没有，可更新到最新版。示例如下：

```
npm install @alicloud/pop-core --save
```

6.2. 初始化

Node.js SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端Node.js SDK。安装方法，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。示例如下：

```
var RPCClient = require('@alicloud/pop-core').RPCClient;
function initVodClient(accessKeyId, accessKeySecret,) {
    var regionId = 'cn-shanghai';   // 点播服务接入地域
    var client = new RPCClient({//填入AccessKey信息
        accessKeyId: accessKeyId,
        accessKeySecret: accessKeySecret,
        endpoint: 'http://vod.' + regionId + '.aliyuncs.com',
        apiVersion: '2017-03-21'
    });
    return client;
}
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例如下：

```
var RPCClient = require('@alicloud/pop-core').RPCClient;
function initVodClient(accessKeyId, accessKeySecret, securityToken) {
    var regionId = 'cn-shanghai';   // 点播服务接入地域
    var client = new RPCClient({//填入STS信息
        accessKeyId: accessKeyId,
        accessKeySecret: accessKeySecret,
        securityToken: securityToken,
        endpoint: 'http://vod.' + regionId + '.aliyuncs.com',
        apiVersion: '2017-03-21'
    });
    return client;
}
```

6.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍Node.js SDK调用服务端上传接口的使用场景和调用示例。

使用场景

注意

本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端示例。操作指引请参见初始化。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("CreateUploadVideo", {
    Title: 'this is a sample',
    FileName: 'filename.mp4'
}, {}).then(function (response) {
    console.log('VideoId = ' + response.VideoId);
    console.log('UploadAddress = ' + response.UploadAddress);
    console.log('UploadAuth = ' + response.UploadAuth);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("RefreshUploadVideo", {
    VideoId: 'VideoId'
}, {}).then(function (response) {
    console.log('UploadAddress = ' + response.UploadAddress);
    console.log('UploadAuth = ' + response.UploadAuth);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("CreateUploadImage", {
    ImageType: 'cover',
    ImageExt: 'jpg'
}, {}).then(function (response) {
    console.log('ImageId = ' + response.ImageId);
    console.log('ImageURL = ' + response.ImageURL);
    console.log('UploadAddress = ' + response.UploadAddress);
    console.log('UploadAuth = ' + response.UploadAuth);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("CreateUploadAttachedMedia", {
    BusinessType: 'watermark',
    MediaExt: 'gif',
    Title: 'this is a sample'
}, {}).then(function (response) {
    console.log('MediaId = ' + response.MediaId);
    console.log('MediaURL = ' + response.MediaURL);
    console.log('UploadAddress = ' + response.UploadAddress);
    console.log('UploadAuth = ' + response.UploadAuth);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var url = 'http://xxxx.mp4';
var uploadMetadatas = [{
    SourceUrl: url,
    Title: 'upload by url sample'
}];
client.request("UploadMediaByURL", {
    UploadURLs: url,
    UploadMetadatas: JSON.stringify(uploadMetadatas)
}, {}).then(function (response) {
    console.log('UploadJobs = ');
    console.log(response.UploadJobs);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.4. 音视频播放

本篇文档提供了Node.js SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetPlayInfo", {
    VideoId: 'VideoId'
}, {}).then(function (response) {
    // play url
    if (response.PlayInfoList && response.PlayInfoList.PlayInfo && response.PlayInfoList.Pl
ayInfo.length > 0){
        for (var i=0; i<response.PlayInfoList.PlayInfo.length; i++){
            console.log("PlayInfo.PlayURL = " + response.PlayInfoList.PlayInfo[i].PlayURL);
        }
    }
    // base metadata
    if (response.VideoBase){
        console.log('VideoBase.Title = ' + response.VideoBase.Title);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetVideoPlayAuth", {
    VideoId: 'VideoId'
}, {}).then(function (response) {
    // play auth
    console.log('PlayAuth = ' + response.PlayAuth);

    // base metadata
    if (response.VideoMeta){
        console.log('VideoMeta.Title = ' + response.VideoMeta.Title);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.5. 媒资管理

本篇文档提供了Node.js SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("SearchMedia", {
    Fields: 'Title,CoverURL,Status',
    Match: "Status in ('Normal','Checking') and CreationTime = ('2018-07-01T08:00:00Z','201
8-08-01T08:00:00Z')",
    PageNo: 1,
    PageSize: 10,
    SearchType: 'video',
    SortBy: 'CreationTime:Desc'
}, {}).then(function (response) {
    console.log("Total = " + response.Total);
    console.log("ScrollToken = " + response.ScrollToken);
    if (response.MediaList && response.MediaList && response.MediaList.length > 0){
        for (var i=0; i<response.MediaList.length; i++){
            console.log("MediaId = " + response.MediaList[i].MediaId);
            console.log("MediaType = " + response.MediaList[i].MediaType);
            console.log("CreationTime = " + response.MediaList[i].CreationTime);
            console.log("Title = " + response.MediaList[i].Video.Title);
            console.log("CoverURL = " + response.MediaList[i].Video.CoverURL);
            console.log("Status = " + response.MediaList[i].Video.Status);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("GetVideoInfo", {
    VideoId: 'VideoId'
}, {}).then(function (response) {
    if (response.Video){
        console.log("Title = " + response.Video.Title);
        console.log("Description = " + response.Video.Description);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("GetVideoInfos", {
    VideoIds: 'Video1,Video2'
}, {}).then(function (response) {
    if (response.VideoList && response.VideoList.length > 0){
        console.log("===== exist video infos : =====");
        for (var i=0; i<response.VideoList.length; i++) {
            var video = response.VideoList[i];
            console.log("VideoId = " + video.VideoId);
            console.log("Title = " + video.Title);
            console.log("Description = " + video.Description);
            console.log("Tags = " + video.Tags);
            console.log("CreationTime = " + video.CreationTime);
        }
    }
    if (response.NonExistVideoIds && response.NonExistVideoIds.length > 0){
        console.log("===== nonexistent videoIds : =====");
        for (var i=0; i<response.NonExistVideoIds.length; i++) {
            console.log("VideoId = " + response.NonExistVideoIds[i]);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("UpdateVideoInfo", {
    VideoId: 'VideoId',
    Title: 'new Title',
    Description: 'new Description',
    Tags: 'new Tag1,new Tag2'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

var updateContent = [{
    VideoId: 'VideoId1',
    Title: 'new Title',
    Tags: 'new Tag1,new Tag2'
},{
    VideoId: 'VideoId2',
    Title: 'new Title',
    Tags: 'new Tag1,new Tag2'
}];
client.request("UpdateVideoInfos", {
    UpdateContent: JSON.stringify(updateContent)
}, {}).then(function (response) {
    if (response.NonExistVideoIds && response.NonExistVideoIds.length > 0){
        console.log("===== nonexistent videoIds : =====");
        for (var i=0; i<response.NonExistVideoIds.length; i++) {
            console.log("VideoId = " + response.NonExistVideoIds[i]);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("DeleteVideo", {
    VideoIds: 'Video1,Video2'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取源文件信息（含源片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("GetMezzanineInfo", {
    VideoId: 'VideoId',
    AuthTimeout: 3600   // 源片下载地址过期时间
}, {}).then(function (response) {
    if (response.Mezzanine){
        console.log('Mezzanine.VideoId = ' + response.Mezzanine.VideoId);
        console.log('Mezzanine.FileURL = ' + response.Mezzanine.FileURL);
        console.log('Mezzanine.Width = ' + response.Mezzanine.Width);
        console.log('Mezzanine.Height = ' + response.Mezzanine.Height);
        console.log('Mezzanine.Size = ' + response.Mezzanine.Size);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("GetVideoList", {
    StartTime: '2018-12-28T06:00:00Z',
    EndTime: '2018-12-28T14:00:00Z',
    PageNo: 1,
    PageSize: 2
}, {}).then(function (response) {
    // 根据指定筛选条件查询到的视频总数
    console.log('Total = ' + response.Total);
    if (response.VideoList && response.VideoList.Video && response.VideoList.Video.length >
0){
        for (var i=0; i<response.VideoList.Video.length; i++){
            var video = response.VideoList.Video[i];
            console.log('Title = ' + video.Title);
            console.log('Description = ' + video.Description);
            console.log('Tags = ' + video.Tags);
            console.log('CreationTime = ' + video.CreationTime);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("DeleteStream", {
    VideoId: 'VideoId',
    JobIds: 'JobId1,JobId2'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("DeleteMezzanines", {
    VideoIds: 'VideoId',
    Force: false
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

var updateContent = [{
    ImageId: 'ImageId1',
    Title: 'new title',
    Tags: 'new Tag1,new Tag2'
},{
    ImageId: 'ImageId2',
    Title: 'new title',
    Tags: 'new Tag1,new Tag2'
}];
client.request("UpdateImageInfos", {
    UpdateContent: JSON.stringify(updateContent)
}, {}).then(function (response) {
    if (response.NonExistImageIds && response.NonExistImageIds.ImageId && response.NonExist
ImageIds.ImageId.length > 0) {
        console.log("======nonexistent ImageIds : ======");
        console.log('ImageId = ' + response.NonExistImageIds.ImageId);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

client.request("GetImageInfo", {
    ImageId: 'ImageId1'
}, {}).then(function (response) {
    if (response.ImageInfo) {
        console.log('ImageId = ' + response.ImageInfo.ImageId);
        console.log('Title = ' + response.ImageInfo.Title);
        console.log('CreationTime = ' + response.ImageInfo.CreationTime);
        console.log('FileURL = ' + response.ImageInfo.Mezzanine.FileURL);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
// 请求示例
var client = initVodClient('<Your AccessKeyId>', '<Your AccessKeySecret>');

// 根据ImageURL删除图片文件
var paras = {
    DeleteImageType: 'ImageURL',
    //Image URL示例：http://example.aliyundoc.com/cover-****.jpg
    ImageURLs: '<your Image URL>'
};
// 根据ImageId删除图片文件
// var paras = {
//     DeleteImageType: 'ImageId',
//     ImageIds: 'ImageId1,ImageId2'
// };
// 根据VideoId删除指定ImageType的图片文件
// var paras = {
//     DeleteImageType: 'VideoId',
//     VideoId: '<your Video ID>',
//     ImageType: 'SpriteSnapshot'
// };
client.request("DeleteImage", paras, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.6. 媒资分类

本文为您提供了Node.js SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("AddCategory", {
    ParentId: -1,          // 父分类ID，若不填，则默认生成一级分类，根节点分类ID为-1
    CateName: '分类名称'    // 分类名称
}, {}).then(function (response) {
    if (response.Category){
        // 创建成功的分类信息
        console.log('ParentId = ' + response.Category.ParentId);
        console.log('CateId = ' + response.Category.CateId);
        console.log('CateName = ' + response.Category.CateName);
        console.log('Level = ' + response.Category.Level);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("UpdateCategory", {
    CateId: -1,    // 分类ID
    CateName: '分类名称2'   // 分类名称
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteCategory", {
    CateId: -1    // 分类ID
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetCategories", {
    CateId: -1,
    PageNo: 1,
    PageSize: 10
}, {}).then(function (response) {
    // 指定的分类信息
    if (response.Category){
        console.log('ParentId = ' + response.Category.ParentId);
        console.log('CateId = ' + response.Category.CateId);
        console.log('CateName = ' + response.Category.CateName);
        console.log('Level = ' + response.Category.Level);
        console.log('SubTotal = ' + response.SubTotal);
    }
    if (response.SubCategories && response.SubCategories.Category && response.SubCategories
.Category.length > 0){
        for (var i=0; i<response.SubCategories.Category.length; i++){
            var subCategory = response.SubCategories.Category[i];
            console.log('SubCategories.ParentId = ' + subCategory.ParentId);
            console.log('SubCategories.CateId = ' + subCategory.CateId);
            console.log('SubCategories.CateName = ' + subCategory.CateName);
            console.log('SubCategories.Level = ' + subCategory.Level);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.7. 媒体处理

本文提供了Node.js SDK媒体处理模块相关功能的API调用示例，包含提交转码作业（HLS标准加密、普通）、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业-普通

调用SubmitTranscodeJobs接口，完成提交转码作业-普通功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为普通转码（不加密）。阿里云视频加密方案请参见阿里云视频加密。

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

/**
 * 构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；如不使用可忽略
 * 需要替换的水印信息对应水印ID必须是关联在指定的转码模板ID上(即TranscodeTemplateId)
 * 不支持通过媒体处理接口去增加一个没有关联上的水印
 */
var overrideParams = {
    Watermarks: [{
        //此处填写转码模板中关联的需要替换的图片水印模板ID
        WatermarkId: 'WatermarkId1',
        //需要替换成对应图片水印文件的OSS地址，水印文件存储源站需要和视频存储源站一致
        //FileURL示例：https://192.168.0.0/16/watermarks/sample-****.png
        FileUrl: '<your File URL>'
    },{
        //模板上面关联需要替换内容的文字水印ID
        WatermarkId: 'WatermarkId2',
        //需要替换成对应的内容
        Content: '用户ID：6****'
    }]
};

client.request("SubmitTranscodeJobs", {
    VideoId: '34a6ca54f5c140eece85a289****',
    TemplateGroupId: 'e8aa925a9798c630d30cd7****',
    OverrideParams: JSON.stringify(overrideParams)
}, {}).then(function (response) {
    if (response.TranscodeJobs && response.TranscodeJobs.TranscodeJob && response.Transcode
Jobs.TranscodeJob.length > 0){
        for(var i=0; i<response.TranscodeJobs.TranscodeJob.length; i++){
            console.log('JobId = ' + response.TranscodeJobs.TranscodeJob[i].JobId);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

提交转码作业-HLS标准加密

调用SubmitTranscodeJobs接口，完成提交转码作业-HLS标准加密功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为使用HLS标准加密提交转码作业的示例。

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');
```

```
/**
 * 提交媒体处理作业
 */
function submitTranscodeJobs(overrideParams, encryptConfig) {
    var paras = {
        VideoId: '34a6ca54f5c140eece85a289****',
        TemplateGroupId: 'e8aa925a9798c630d30cd7****'
    };
    // 覆盖参数，暂只支持水印部分参数替换(只有需要替换水印相关信息才需要传递)
    if (!overrideParams){
        paras.OverrideParams = JSON.stringify(overrideParams);
    }
    // HLS标准加密配置(只有标准加密才需要传递)
    if (!encryptConfig){
        paras.EncryptConfig = JSON.stringify(encryptConfig);
    }

    client.request("SubmitTranscodeJobs", paras, {}).then(function (response) {
        console.log(response);
        if (response.TranscodeJobs && response.TranscodeJobs.TranscodeJob && response.Trans
codeJobs.TranscodeJob.length > 0){
            for(var i=0; i<response.TranscodeJobs.TranscodeJob.length; i++){
                console.log('JobId = ' + response.TranscodeJobs.TranscodeJob[i].JobId);
            }
        }
        console.log('RequestId = ' + response.RequestId);
    }).catch(function (response) {
        console.log('ErrorCode = ' + response.data.Code);
        console.log('ErrorMessage = ' + response.data.Message);
        console.log('RequestId = ' + response.data.RequestId);
    });
}

/**
 * 生成加密需要的密钥，response中包含密文密钥和明文密钥，用户只需要将密文密钥传递给点播即可
 * 注意：KeySpec 必须传递AES_128，且不能设置NumberOfBytes
 */
client.request("GenerateDataKey", {
    KeyId: 'KeyId',   // 点播提供生成密钥的service key，在用户的密钥管理服务中可看到描述为vod的加密
key
    KeySpec: 'AES_128'
}, {}).then(function (response) {
    console.log('GenerateDataKey RequestId = ' + response.RequestId);
    var encryptConfig = {
        // 解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥
都不一样
        // 至于Ciphertext这个解密接口参数的名称，用户可自行制定，这里只作为参考参数名称
        //DecryptKeyUri示例：http://demo.example.com/decrypt?Ciphertext=' + response.Ciphert
extBlob
        DecryptKeyUri: '<your DecryptKey URI>',
        // 密钥服务的类型，目前只支持KMS
        KeyServiceType: 'KMS',
        // 密文密钥
```

```
        CipherText: response.CiphertextBlob
    };

    /**
     * 构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；如不使用可忽略
     * 需要替换的水印信息对应水印ID必须是关联在指定的转码模板ID上(即TranscodeTemplateId)
     * 不支持通过媒体处理接口去增加一个没有关联上的水印
     */
    var overrideParams = {
        Watermarks: [{
            //模板上面关联需要替换的水印文件图片水印ID
            WatermarkId: 'WatermarkId1',
            //需要替换成对应图片水印文件的OSS地址，水印文件存储源站需要和视频存储源站一致
            //FileURL示例：https://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermar
ks/sample-****.png
            FileUrl: '<your File URL>'
        },{
            //模板上面关联需要替换内容的文字水印ID
            WatermarkId: 'WatermarkId2',
            //需要替换成对应的内容
            Content: '用户ID：6****'
        }]
    };

    submitTranscodeJobs(overrideParams, encryptConfig);
}).catch(function (response) {
    console.log('GenerateDataKey ErrorCode = ' + response.data.Code);
    console.log('GenerateDataKey ErrorMessage = ' + response.data.Message);
    console.log('GenerateDataKey RequestId = ' + response.data.RequestId);
});
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明**
>
> 创建截图模板详细请参见截图模板。

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var spriteSnapshotConfig = {
    CellWidth: '120',
    CellHeight: '68',
    Columns: '3',
    Lines: '10',
    Padding: '20',
    Margin: '50',
    //保留雪碧图原始图
    KeepCellPic: 'keep',
    Color: 'tomato'
};
client.request("SubmitSnapshotJob", {
    VideoId: '52b440eabcc8ae1cbffc85****',  // 需要截图的视频ID
    SnapshotTemplateId: '1111e6b8baadf589cbffc58****',  // 截图模板ID
    //如果设置了SnapshotTemplateId，会忽略下面参数
    Count: 50,
    SpecifiedOffsetTime: 0,
    Interval: 1,
    Width: '200',
    Height: '200',
    SpriteSnapshotConfig: JSON.stringify(spriteSnapshotConfig)
}, {}).then(function (response) {
    if (response.SnapshotJob){
        //任务ID
        console.log('JobId = ' + response.SnapshotJob.JobId);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListSnapshots", {
    VideoId: '1111121b52b440eabcc8ae1cbff58****',  //视频ID
    SnapshotType: 'CoverSnapshot',  //截图类型
    PageNo: '1',
    PageSize: '20'
}, {}).then(function (response) {
    if (response.MediaSnapshot && response.MediaSnapshot.Snapshots && response.MediaSnapsho
t.Snapshots.Snapshot){
        for (var i=0; i<response.MediaSnapshot.Snapshots.Snapshot.length; i++){
            //截图地址
            console.log('Url = ' + response.MediaSnapshot.Snapshots.Snapshot[i].Url);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SubmitPreprocessJobs", {
    VideoId: '111111192e8e48c1ad51f237c54****',  //视频ID
    PreprocessType: 'PreprocessType'
}, {}).then(function (response) {
    if (response.PreprocessJobs && response.PreprocessJobs.PreprocessJob && response.Prepro
cessJobs.PreprocessJob.length > 0){
        for (var i=0; i<response.PreprocessJobs.PreprocessJob.length; i++){
            //任务ID
            console.log('JobId = ' + response.PreprocessJobs.PreprocessJob[i].JobId);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.8. 视频水印

本篇文档提供了Node.js SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印、设置默认水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

> **说明**
>
> - 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
> - 水印文件OSS上传详细参数请参见OSS文件上传。

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

//图片水印的位置配置数据
var watermarkConfig = {
    Dx: '8',   //水印的横向偏移距离
    Dy: '8',   //水印的纵向偏移距离
    Width: '55',  //水印显示的宽
    Height: '55',  //水印显示的高
    ReferPos: 'BottomRight',  //水印显示的相对位置(左上、右上、左下、右下)
    //水印显示的时间线(开始显示和结束显示时间)
    Timeline: {
        Start: '2',  //水印开始显示时间
        Duration: 'ToEND'  //水印结束显示时间
    }
};
//文字水印的位置配置数据
// var watermarkConfig = {
//     Content: 'testwatermark',  //文字水印显示的内容
//     FontName: 'SimSun',  //文字水印的字体名称
//     FontSize: 25,  //文字水印的字体大小
//     FontColor: 'Black',  //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
//     FontAlpha: '0.2',    //文字水印的透明度
//     BorderColor: 'White',  //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
//     BorderWidth: 1,  //文字水印的描边宽度
//     Top: 20,  //文字水印距离视频画面上边的偏移距离
//     Left: 15  //文字水印距离视频画面左边的偏移距离
// };

client.request("AddWatermark", {
    Name: 'addwatermark',  //水印名称
    // 获取水印文件在oss的URL，图片水印必传图片文件的oss文件地址，水印文件必须和视频在同一个区域.
    // 例如:华东2视频，水印文件必须存放在华东2
    //File URL示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermark/test**
**.png
    FileUrl: '<your File URL>',
    WatermarkConfig: JSON.stringify(watermarkConfig),
    Type: 'Image'
}, {}).then(function (response) {
    if (response.WatermarkInfo){
        //水印ID
        console.log('WatermarkId = ' + response.WatermarkInfo.WatermarkId);
        //水印的位置、效果配置数据
        console.log('WatermarkConfig = ' + response.WatermarkInfo.WatermarkConfig);
        //水印文件的FileUrl(文字水印该字段值为空)
        console.log('FileUrl = ' + response.WatermarkInfo.FileUrl);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

//图片水印的位置配置数据
var watermarkConfig = {
    Dx: '8',   //水印的横向偏移距离
    Dy: '8',   //水印的纵向偏移距离
    Width: '55',  //水印显示的宽
    Height: '55',  //水印显示的高
    ReferPos: 'BottomRight',  //水印显示的相对位置(左上、右上、左下、右下)
    //水印显示的时间线(开始显示和结束显示时间)
    Timeline: {
        Start: '2',  //水印开始显示时间
        Duration: 'ToEND'  //水印结束显示时间
    }
};
//文字水印的位置配置数据
// var watermarkConfig = {
//     Content: 'testwatermark',  //文字水印显示的内容
//     FontName: 'SimSun',  //文字水印的字体名称
//     FontSize: 25,  //文字水印的字体大小
//     FontColor: 'Black',  //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
//     FontAlpha: '0.2',    //文字水印的透明度
//     BorderColor: 'White',  //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
//     BorderWidth: 1,  //文字水印的描边宽度
//     Top: 20,  //文字水印距离视频画面上边的偏移距离
//     Left: 15  //文字水印距离视频画面左边的偏移距离
// };

client.request("UpdateWatermark", {
    WatermarkId: 'WatermarkId',
    Name: 'updatewatermark',  //水印名称
    WatermarkConfig: JSON.stringify(watermarkConfig)
}, {}).then(function (response) {
    if (response.WatermarkInfo){
        //水印ID
        console.log('WatermarkId = ' + response.WatermarkInfo.WatermarkId);
        //水印的位置、效果配置数据
        console.log('WatermarkConfig = ' + response.WatermarkInfo.WatermarkConfig);
        //水印文件的FileUrl(文字水印该字段值为空)
        console.log('FileUrl = ' + response.WatermarkInfo.FileUrl);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteWatermark", {
    WatermarkId: 'WatermarkId'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListWatermark", {}, {}).then(function (response) {
    if (response.WatermarkInfos && response.WatermarkInfos.length > 0){
        for (var i=0; i<response.WatermarkInfos.length; i++){
            //水印ID
            console.log('WatermarkId = ' + response.WatermarkInfos[i].WatermarkId);
            //水印的位置、效果配置数据
            console.log('WatermarkConfig = ' + response.WatermarkInfos[i].WatermarkConfig);
            //水印文件的FileUrl(文字水印该字段值为空)
            console.log('FileUrl = ' + response.WatermarkInfos[i].FileUrl);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetWatermark", {
    WatermarkId: 'WatermarkId'
}, {}).then(function (response) {
    if (response.WatermarkInfo){
        //水印ID
        console.log('WatermarkId = ' + response.WatermarkInfo.WatermarkId);
        //水印的位置、效果配置数据
        console.log('WatermarkConfig = ' + response.WatermarkInfo.WatermarkConfig);
        //水印文件的FileUrl(文字水印该字段值为空)
        console.log('FileUrl = ' + response.WatermarkInfo.FileUrl);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SetDefaultWatermark", {
    WatermarkId: 'WatermarkId'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.9. 转码模板

本篇文档提供了Node.js SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var transcodeTemplateList = [
    {
        //视频流转码配置
        Video: {
            Width: 640,
            Bitrate: 400,
            Fps: 25,
            Remove: false,
            Codec: 'H.264',
            Gop: '250'
        },
        //音频流转码配置
        Audio: {
            Codec: 'AAC',
            Bitrate: '64',
            Channels: '2',
            Samplerate: '32000'
        },
        //封装容器
        Container: {
            Format: 'mp4'
        },
        //条件转码配置
        TransConfig: {
            IsCheckReso: false,
            IsCheckResoFail: false,
            IsCheckVideoBitrate: false,
            IsCheckVideoBitrateFail: false,
            IsCheckAudioBitrate: false,
            IsCheckAudioBitrateFail: false
        },
        // // 加密配置(只支持m3u8)
        // EncryptSetting: {
        //     EncryptType: 'Private'
        // },
        //清晰度
        Definition: 'LD',
        //模板名称
        TemplateName: 'testtemplate',
        //水印ID(多水印关联)
        WatermarkIds: ['USER_DEFAULT_WATERMARK','789456bdc1ff65782f8995c6dd45****']
    }
];
client.request("AddTranscodeTemplateGroup", {
    // 转码模板组名称
    Name: 'grouptest',
    // 转码模板配置数据
    TranscodeTemplateList: JSON.stringify(transcodeTemplateList)
}, {}).then(function (response) {
    console.log('TranscodeTemplateGroupId = ' + response.TranscodeTemplateGroupId);
    console.log('RequestId = ' + response.RequestId);
```

```
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var transcodeTemplateList = [
    {
        // 转码模板ID
        TranscodeTemplateId: '9875824423bca267f931d3b25687****',
        // 视频流转码配置
        Video: {
            Width: 640,
            Bitrate: 500,
            Fps: 25,
            Remove: false,
            Codec: 'H.264',
            Gop: '250'
        },
        // 音频流转码配置
        Audio: {
            Codec: 'AAC',
            Bitrate: '64',
            Channels: '2',
            Samplerate: '32000'
        },
        // 封装容器
        Container: {
            Format: 'mp4'
        },
        // 条件转码配置
        TransConfig: {
            IsCheckReso: false,
            IsCheckResoFail: false,
            IsCheckVideoBitrate: false,
            IsCheckVideoBitrateFail: false,
            IsCheckAudioBitrate: false,
            IsCheckAudioBitrateFail: false
        },
        // // 加密配置(只支持m3u8)
        // EncryptSetting: {
        //     EncryptType: 'Private'
        // },
        // 清晰度
        Definition: 'LD',
```

```
        // 模板名称
        TemplateName: 'testtemplate',
        // 水印ID(多水印关联)
        WatermarkIds: ['USER_DEFAULT_WATERMARK','854325bdc1ff65782f8995c6dd85****']
    }
];
client.request("UpdateTranscodeTemplateGroup", {
    TranscodeTemplateGroupId: '789458252788dfd94e27d7358b254****',
    TranscodeTemplateList: JSON.stringify(transcodeTemplateList)
}, {}).then(function (response) {
    console.log('TranscodeTemplateGroupId = ' + response.TranscodeTemplateGroupId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListTranscodeTemplateGroup", {}, {}).then(function (response) {
    if (response.TranscodeTemplateGroupList && response.TranscodeTemplateGroupList.length >
0){
        for (var i=0; i<response.TranscodeTemplateGroupList.length; i++){
            var transcodeTemplateGroup = response.TranscodeTemplateGroupList[i];
            console.log('The ' + i + ' TranscodeTemplateGroup.');
            //转码模板组ID
            console.log('TranscodeTemplateGroupId = ' + transcodeTemplateGroup.TranscodeTem
plateGroupId);
            //转码模板组名称
            console.log('Name = ' + transcodeTemplateGroup.Name);
            //转码模板组创建时间
            console.log('CreationTime = ' + transcodeTemplateGroup.CreationTime);
            //转码模板组最近修改时间
            console.log('ModifyTime = ' + transcodeTemplateGroup.ModifyTime);
            //是否是默认转码模板组
            console.log('IsDefault = ' + transcodeTemplateGroup.IsDefault);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetTranscodeTemplateGroup", {
    TranscodeTemplateGroupId: '81553142788dfd94e27d7358b285****'
}, {}).then(function (response) {
    console.log(response);
    if (response.TranscodeTemplateGroup){
        // 转码模板组ID
        console.log('TranscodeTemplateGroupId = ' + response.TranscodeTemplateGroup.Transco
deTemplateGroupId);
        // 转码模板组名称
        console.log('Name = ' + response.TranscodeTemplateGroup.Name);
        // 转码模板组创建时间
        console.log('CreationTime = ' + response.TranscodeTemplateGroup.CreationTime);
        // 转码模板组最近修改时间
        console.log('ModifyTime = ' + response.TranscodeTemplateGroup.ModifyTime);
        // 转码模板信息
        if (response.TranscodeTemplateGroup.TranscodeTemplateList && response.TranscodeTemp
lateGroup.TranscodeTemplateList.length > 0){
            for (var i=0; i<response.TranscodeTemplateGroup.TranscodeTemplateList.length; i
++){
                var transcodeTemplate = response.TranscodeTemplateGroup.TranscodeTemplateLi
st[i];
                console.log("The " + i + " TranscodeTemplate.");
                // 转码模板ID
                console.log('TranscodeTemplateId = ' + transcodeTemplate.TranscodeTemplateI
d);
                // 转码模板名称
                console.log('TemplateName = ' + transcodeTemplate.TemplateName);
                // 转码模板清晰度标记
                console.log('Definition = ' + transcodeTemplate.Definition);
                // 转码模板封装格式配置
                console.log('Container = ');
                console.log(transcodeTemplate.Container);
                // 转码模板视频转码配置
                console.log('Video = ');
                console.log(transcodeTemplate.Video);
                // 转码模板音频转码配置
                console.log('Audio = ');
                console.log(transcodeTemplate.Audio);
                // 转码模板条件转码配置
                console.log('TransConfig = ');
                console.log(transcodeTemplate.TransConfig);
                // 转码需要压制的水印对应ID
                console.log('WatermarkIds = ');
                console.log(transcodeTemplate.WatermarkIds);
            }
        }
```

```
    }

    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SetDefaultTranscodeTemplateGroup", {
    // 转码模板组ID
    TranscodeTemplateGroupId: '45628142788dfd94e27d7358b258****'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteTranscodeTemplateGroup", {
    TranscodeTemplateGroupId: '12356442788dfd94e27d7358b285****'
}, {}).then(function (response) {
    console.log(response);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.10. 截图模板

本文为您提供了Node.js SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var templateConfig = {
    //截图类型(存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot")
    SnapshotType: 'SpriteSnapshot',
    //普通截图配置(与雪碧图配置共用该配置信息)
    SnapshotConfig:{
        Count: '50',
        Interval: '1',
        SpecifiedOffsetTime: '0',
        Width: '200',
        Height: '200',
        FrameType: 'normal',
        //雪碧图配置(雪碧图配置必须是建立在普通截图配置之上)
        SpriteSnapshotConfig:{
            CellWidth: '120',
            CellHeight: '68',
            Columns: '3',
            Lines: '10',
            Padding: '20',
            Margin: '50',
            KeepCellPic: 'keep',
            Color: 'tomato'
        }
    }
};
client.request("AddVodTemplate", {
    Name: '截图模板添加测试',      //模板名称
    TemplateType: 'Snapshot',    //模板类型，固定值为Snapshot
    TemplateConfig: JSON.stringify(templateConfig)    //截图模板配置数据生成
}, {}).then(function (response) {
    //截图模板ID
    console.log('SnapshotVodTemplateId = ' + response.VodTemplateId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var templateConfig = {
    //截图类型(存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot")
    SnapshotType: 'SpriteSnapshot',
    //普通截图配置(与雪碧图配置共用该配置信息)
    SnapshotConfig:{
        Count: '50',
        Interval: '1',
        SpecifiedOffsetTime: '0',
        Width: '200',
        Height: '200',
        FrameType: 'normal',
        //雪碧图配置(雪碧图配置必须是建立在普通截图配置之上)
        SpriteSnapshotConfig:{
            CellWidth: '120',
            CellHeight: '68',
            Columns: '3',
            Lines: '10',
            Padding: '20',
            Margin: '50',
            KeepCellPic: 'keep',
            Color: 'tomato'
        }
    }
};
client.request("UpdateVodTemplate", {
    VodTemplateId: 'VodTemplateId',    //设置要修改的模板ID
    Name: '截图模板修改测试',    //模板名称
    TemplateConfig: JSON.stringify(templateConfig)    //截图模板配置数据生成
}, {}).then(function (response) {
    //截图模板ID
    console.log('SnapshotVodTemplateId = ' + response.VodTemplateId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteVodTemplate", {
    VodTemplateId: 'VodTemplateId'    //设置要删除的模板ID
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListVodTemplate", {
    TemplateType: 'Snapshot'    //模板类型，固定值为Snapshot
}, {}).then(function (response) {
    //打印模板数目
    console.log('SnapshotVodTemplate Count = ' + response.VodTemplateInfoList.length);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetVodTemplate", {
    VodTemplateId: 'VodTemplateId'    //设置要查询的模板ID
}, {}).then(function (response) {
    // 打印返回信息中的截图模板ID
    if (response.VodTemplateInfo){
        console.log('SnapshotVodTemplateId = ' + response.VodTemplateInfo.VodTemplateId);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.11. 视频剪辑

本篇文档提供了Node.js SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

> **说明**
>
> 通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// Build Editing Project Timeline
var timeline = {
    // Video Track
    VideoTracks: [{
        // Video Track Clips
        VideoTrackClips: [{
            MediaId: 'MediaId1'
        },{
            MediaId: 'MediaId2'
        }]
    }]
};
// Set Produce Media Metadata
var mediaMetadata = {
    Title: 'Title',  // Produce Media Title
    Description: 'Description',  // Produce Media Description
    //Cover URL示例：http://example.aliyundoc.com/media/cover/mediaid****.jpg
    CoverURL: '<your Cover URL>',  // Produce Media UserDefined Cover URL
    CateId: null,   // Produce Media Category ID
    Tags: 'Tag1,Tag2'   // Produce Media Category Name
};
// Set Produce Configuration
var produceConfig = {
    /*
     生成过程可以生成媒体夹层文件。您可以使用夹层文件来转码其他媒体文件，就像文件上传完成后的转码过程一样。此字段描述生产夹层完成后的Transocde TemplateGroup ID。
     1. Not required
     2. Use default transcode template group id when empty
     */
    TemplateGroupId: null
};
client.request("ProduceEditingProjectVideo", {
    Timeline: JSON.stringify(timeline),
    MediaMetadata: JSON.stringify(mediaMetadata),
    ProduceConfig: JSON.stringify(produceConfig)
}, {}).then(function (response) {
    // Produce Media ID
    console.log('MediaId = '+ response.MediaId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// Set Produce Media Metadata
var mediaMetadata = {
    Title: 'Title',  // Produce Media Title
    Description: 'Description',  // Produce Media Description
    //Cover URL示例：http://example.aliyundoc.com/media/cover/mediaid****.jpg
    CoverURL: '<your Cover URL>',  // Produce Media UserDefined Cover URL
    CateId: null,   // Produce Media Category ID
    Tags: 'Tag1,Tag2'   // Produce Media Category Name
};
// Set Produce Configuration
var produceConfig = {
    /*
     生成过程可以生成媒体夹层文件。您可以使用夹层文件来转码其他媒体文件，就像文件上传完成后的转码过程一样。此字段描述生产夹层完成后的Transocde TemplateGroup ID。
     1. Not required
     2. Use default transcode template group id when empty
     */
    TemplateGroupId: null
};
client.request("ProduceEditingProjectVideo", {
    ProjectId: 'ProjectId',    // Set Editing Project ID need to produce
    MediaMetadata: JSON.stringify(mediaMetadata),
    ProduceConfig: JSON.stringify(produceConfig)
}, {}).then(function (response) {
    // Produce Media ID
    console.log('MediaId = '+ response.MediaId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// Build Editing Project Timeline
var timeline = {
    // Video Track
    VideoTracks: [{
        // Video Track Clips
        VideoTrackClips: [{
            MediaId: 'MediaId1'
        },{
            MediaId: 'MediaId2'
        }]
    }]
};
client.request("AddEditingProject", {
    Timeline: JSON.stringify(timeline),
    Title: 'Editing Project Title',   // Set Editing Project Title
    Description: 'Editing Project Description',   // Set Editing Project Description
    //Cover URL示例：http://example.aliyundoc.com/editingproject/cover/projectid.jpg
    CoverURL: '<your Cover URL>'  // Set Editing Project Cover URL
}, {}).then(function (response) {
    if (response.Project){
        console.log('Editing Project Id = '+ response.Project.ProjectId);
        console.log('Editing Project Title = '+ response.Project.Title);
        console.log('Editing Project Description = '+ response.Project.Description);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// Build Editing Project Timeline
var timeline = {
    // Video Track
    VideoTracks: [{
        // Video Track Clips
        VideoTrackClips: [{
            MediaId: 'MediaId1'
        },{
            MediaId: 'MediaId2'
        }]
    }]
};
client.request("UpdateEditingProject", {
    ProjectId: 'ProjectId',
    Timeline: JSON.stringify(timeline),
    Title: 'Editing Project Title',   // Set Editing Project Title
    Description: 'Editing Project Description',   // Set Editing Project Description
    //Cover URL示例：http://example..aliyundoc.com/editingproject/cover/projectid.jpg
    CoverURL: '<your Cover URL>'  // Set Editing Project Cover URL
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteEditingProject", {
    ProjectIds: 'ProjectId1,projectid2'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetEditingProject", {
    ProjectId: 'ProjectId'
}, {}).then(function (response) {
    if (response.Project){
        console.log('Editing Project Id = '+ response.Project.ProjectId);
        console.log('Editing Project Timeline = '+ response.Project.Timeline);
        console.log('Editing Project Title = '+ response.Project.Title);
        console.log('Editing Project Description = '+ response.Project.Description);
        console.log('Editing Project CoverURL = '+ response.Project.CoverURL);
        console.log('Editing Project CreationTime = '+ response.Project.CreationTime);
        console.log('Editing Project ModifiedTime = '+ response.Project.ModifiedTime);
        console.log('Editing Project Status = '+ response.Project.Status);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SearchEditingProject", {
    Title: 'TitleKeywords',
    StartTime: '2018-12-28T14:42:18Z',
    EndTime: '2018-12-30T14:42:18Z',
    PageSize: 10,
    PageNo: 1
}, {}).then(function (response) {
    if (response.ProjectList && response.ProjectList.Project && response.ProjectList.Projec
t.length > 0){
        console.log('Total Size = ' + response.Total);
        for(var i=0; i<response.ProjectList.Project.length; i++){
            var project = response.ProjectList.Project[i];
            console.log('No ' + i + ' Editing Project');
            console.log('Editing Project Id = '+ project.ProjectId);
            console.log('Editing Project Title = '+ project.Title);
            console.log('Editing Project Description = '+ project.Description);
            console.log('Editing Project CoverURL = '+ project.CoverURL);
            console.log('Editing Project CreationTime = '+ project.CreationTime);
            console.log('Editing Project ModifiedTime = '+ project.ModifiedTime);
            console.log('Editing Project Status = '+ project.Status);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SetEditingProjectMaterials", {
    ProjectId: 'ProjectId',
    MaterialIds: 'MaterialId1,MaterialId2'
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetEditingProjectMaterials", {
    ProjectId: 'ProjectId',
    Type: 'video'
}, {}).then(function (response) {
    if (response.MaterialList && response.MaterialList.Material && response.MaterialList.Ma
terial.length > 0){
        for(var i=0; i<response.MaterialList.Material.length; i++){
            var material = response.MaterialList.Material[i];
            console.log('No ' + i + ' Editing Project');
            console.log('Editing Project Material Id = '+ material.MaterialId);
            console.log('Editing Project Material Title = '+ material.Title);
            console.log('Editing Project Material Category ID = '+ material.CateId);
            console.log('Editing Project Material Category Name = '+ material.CateName);
            console.log('Editing Project Material Tags = '+ material.Tags);
            console.log('Editing Project Material CoverURL = '+ material.CoverURL);
            console.log('Editing Project Material Create Time = '+ material.CreationTime);
            console.log('Editing Project Material Last ModifiedTime = '+ material.ModifiedT
ime);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.12. 媒体审核

本篇文档提供了Node.js SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SubmitAIMediaAuditJob", {
    MediaId: '584625a507a44d0eaa7424afd56****'  // 设置媒体资源ID
}, {}).then(function (response) {
    console.log('JobId = ' + response.JobId);        // 作业ID
    console.log('MediaId = ' + response.MediaId);    // 视频ID
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetAIMediaAuditJob", {
    JobId: '584625a507a44d0eaa7424afd56****'  // 设置作业ID
}, {}).then(function (response) {
    if (response.MediaAuditJob){
        // 作业结果信息
        console.log('MediaId = ' + response.MediaAuditJob.MediaId);
        console.log('JobId = ' + response.MediaAuditJob.JobId);
        console.log('Type = ' + response.MediaAuditJob.Type);
        console.log('Status = ' + response.MediaAuditJob.Status);
        if (response.MediaAuditJob.Code){
            console.log('Code = ' + response.MediaAuditJob.Code);
        }
        if (response.MediaAuditJob.Message){
            console.log('Message = ' + response.MediaAuditJob.Message);
        }
        if (response.MediaAuditJob.Data){
            console.log('Data Label = ' + response.MediaAuditJob.Data.Label);
            console.log('Data Suggestion = ' + response.MediaAuditJob.Data.Suggestion);
            console.log('Data AbnormalModules = ' + response.MediaAuditJob.Data.AbnormalMod
ules);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetMediaAuditResult", {
    MediaId: '584625a507a44d0eaa7424afd56****'  // 设置媒体资源ID
}, {}).then(function (response) {
    console.log(response);
    if (response.MediaAuditResult){
        // 智能审核结果信息
        console.log('Data Label = ' + response.MediaAuditResult.Label);
        console.log('Data Suggestion = ' + response.MediaAuditResult.Suggestion);
        console.log('Data AbnormalModules = ' + response.MediaAuditResult.AbnormalModules);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetMediaAuditResultDetail", {
    MediaId: '584625a507a44d0eaa7424afd56****',  // 设置媒体资源ID
    PageNo: 1
}, {}).then(function (response) {
    if (response.MediaAuditResultDetail){
        // 智能审核结果详情
        console.log('Data Total = ' + response.MediaAuditResultDetail.Total);
        console.log('Data List Size = ' + response.MediaAuditResultDetail.List.length);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetMediaAuditResultTimeline", {
    MediaId: '584625a507a44d0eaa7424afd56****'  // 设置媒体资源ID
}, {}).then(function (response) {
    if (response.MediaAuditResultTimeline){
        // 智能审核结果时间线
        if (response.MediaAuditResultTimeline.Terrorism){
            console.log('Terrorism = ' + response.MediaAuditResultTimeline.Terrorism);
        }
        if (response.MediaAuditResultTimeline.Porn){
            console.log('Porn = ' + response.MediaAuditResultTimeline.Porn);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

var auditContent = [{
    VideoId: 'VideoId',  // 视频ID
    Status: 'Blocked',   // 审核状态
    Reason: '含有淫秽色情内容'  // 若审核状态为屏蔽时，需给出屏蔽的理由，最长支持128字节
}];
client.request("CreateAudit", {
    AuditContent: JSON.stringify(auditContent)
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetAuditHistory", {
    VideoId: 'VideoId',
    PageNo: 1,
    PageSize: 10
}, {}).then(function (response) {
    // 打印审核历史记录总条数
    console.log('Total = ' + response.Total);
    // 打印本次审核结果
    console.log('Status = ' + response.Status);
    // 打印审核历史记录列表
    console.log('Histories = ');
    console.log(response.Histories);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SetAuditSecurityIp", {
    SecurityGroupName: 'MyGroupName',  // 审核安全组名称，默认为Default
    OperateMode: 'Cover',   // 操作方式，默认值Append(追加IP白名单)
    Ips: '10.23.12.20,10.23.12.21,10.23.12.22'    // 审核安全IP列表，每个分组最多支持100个并以逗号隔开
}, {}).then(function (response) {
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListAuditSecurityIp", {
    SecurityGroupName: 'MyGroupName'  // 审核安全组名称
}, {}).then(function (response) {
    // 打印审核安全IP详情
    console.log('SecurityIpList = ');
    console.log(response.SecurityIpList);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.13. 视频AI

本篇文档提供了Node.js SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化接口。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SubmitAIJob", {
    MediaId: '584625a507a44d0eaa7424afd56****',
    Types: 'AIVideoCover'
}, {}).then(function (response) {
    if (response.AIJobList && response.AIJobList.AIJob && response.AIJobList.AIJob.length >
0){
        for(var i=0; i<response.AIJobList.AIJob.length; i++){
            var job = response.AIJobList.AIJob[i];
            // 视频ID
            console.log('MediaId = ' + job.MediaId);
            // 作业ID
            console.log('JobId = ' + job.JobId);
            // AI类型
            console.log('Type = ' + job.Type);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListAIJob", {
    JobIds: 'JobId1,JobId2'
}, {}).then(function (response) {
    // 打印作业列表
    console.log("============ AIJobList ============");
    if (response.AIJobList && response.AIJobList.AIJob && response.AIJobList.AIJob.length >
0){
        for(var i=0; i<response.AIJobList.AIJob.length; i++){
            var job = response.AIJobList.AIJob[i];
            // 视频ID
            console.log('MediaId = ' + job.MediaId);
            // 作业ID
            console.log('JobId = ' + job.JobId);
            // AI类型
            console.log('Type = ' + job.Type);
            // 作业状态
            console.log('Status = ' + job.Status);
            // 作业结果
            console.log('Data = ' + job.Data);
        }
    }
    // 打印不存在的作业ID
    console.log("============ NonExistAIJobIds ============");
    if (response.NonExistAIJobIds && response.NonExistAIJobIds.String && response.NonExistA
IJobIds.String.length > 0) {
        for (var i=0; i<response.NonExistAIJobIds.String.length; i++) {
            // 作业ID
            console.log('NonExistAIJobId = ' + response.NonExistAIJobIds.String[i]);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// 模板详细配置，以审核模版为例
var templateConfig = {
    AuditAutoBlock: "no",
    AuditContent: ["screen"],
    AuditRange:["video", "image-cover", "text-title"],
    AuditItem: ["terrorism", "porn"]
};
client.request("AddAITemplate", {
    TemplateType: 'AIMediaAudit',    // 设置模板类型，以智能审核模板为例
    TemplateName: 'myaitemplate',    // 设置模板名称
    TemplateConfig: JSON.stringify(templateConfig)    // 设置模板详细配置
}, {}).then(function (response) {
    console.log(response);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DeleteAITemplate", {
    TemplateId: '584625a507a44d0eaa7424afd56****' // 设置模板ID
}, {}).then(function (response) {
    console.log('TemplateId = ' + response.TemplateId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

// 模板详细配置，以审核模版为例
var templateConfig = {
    AuditAutoBlock: "no",
    AuditContent: ["screen"],
    AuditRange:["video", "image-cover"],
    AuditItem: ["terrorism", "porn"]
};
client.request("UpdateAITemplate", {
    TemplateId: '584625a507a44d0eaa7424afd56****',  // 模板ID
    TemplateName: 'myaitemplatemodify',              // 模板名称
    TemplateConfig: JSON.stringify(templateConfig)   // 模板详细配置
}, {}).then(function (response) {
    console.log('TemplateId = ' + response.TemplateId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetAITemplate", {
    TemplateId: '584625a507a44d0eaa7424afd56****'  // 模板ID
}, {}).then(function (response) {
    console.log(response);
    // 模板信息
    if (response.TemplateInfo){
        console.log('TemplateId = ' + response.TemplateInfo.TemplateId);
        console.log('TemplateType = ' + response.TemplateInfo.TemplateType);
        console.log('TemplateName = ' + response.TemplateInfo.TemplateName);
        console.log('TemplateConfig = ' + response.TemplateInfo.TemplateConfig);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListAITemplate", {
    TemplateType: 'AIMediaAudit'  // 模板类型
}, {}).then(function (response) {
    console.log(response);
    // 模板信息列表
    if (response.TemplateInfoList && response.TemplateInfoList.length > 0){
        for (var i=0; i<response.TemplateInfoList.length; i++){
            console.log("The " + i + " Template.");
            console.log('TemplateId = ' + response.TemplateInfoList[i].TemplateId);
            console.log('TemplateType = ' + response.TemplateInfoList[i].TemplateType);
            console.log('TemplateName = ' + response.TemplateInfoList[i].TemplateName);
            console.log('TemplateConfig = ' + response.TemplateInfoList[i].TemplateConfig);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SetDefaultAITemplate", {
    TemplateId: '584625a507a44d0eaa7424afd56****' // 模板ID
}, {}).then(function (response) {
    console.log('TemplateId = ' + response.TemplateId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetDefaultAITemplate", {
    TemplateType: 'AIMediaAudit'   // 模板类型
}, {}).then(function (response) {
    if (response.TemplateInfo){
        console.log('TemplateId = ' + response.TemplateInfo.TemplateId);
        console.log('TemplateType = ' + response.TemplateInfo.TemplateType);
        console.log('TemplateName = ' + response.TemplateInfo.TemplateName);
        console.log('IsDefault = ' + response.TemplateInfo.IsDefault);
        console.log('TemplateConfig = ' + response.TemplateInfo.TemplateConfig);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.14. 视频DNA

本文为您提供了Node.js SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("SubmitAIJob", {
    MediaId: '584625a507a44d0eaa7424afd56****',
    Types: 'AIMediaDNA'
}, {}).then(function (response) {
    if (response.AIJobList && response.AIJobList.AIJob && response.AIJobList.AIJob.length >
0){
        for(var i=0; i<response.AIJobList.AIJob.length; i++){
            var job = response.AIJobList.AIJob[i];
            // 视频ID
            console.log('MediaId = ' + job.MediaId);
            // 作业ID
            console.log('JobId = ' + job.JobId);
            // AI类型
            console.log('Type = ' + job.Type);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListAIJob", {
    JobIds: 'JobId1,JobId2'
}, {}).then(function (response) {
    // 打印作业列表
    console.log("============ AIJobList ============");
    if (response.AIJobList && response.AIJobList.AIJob && response.AIJobList.AIJob.length >
0){
        for(var i=0; i<response.AIJobList.AIJob.length; i++){
            var job = response.AIJobList.AIJob[i];
            // 视频ID
            console.log('MediaId = ' + job.MediaId);
            // 作业ID
            console.log('JobId = ' + job.JobId);
            // AI类型，注意校验类型是否为AIMediaDNA
            console.log('Type = ' + job.Type);
            // 作业状态
            console.log('Status = ' + job.Status);
            // 作业结果
            console.log('Data = ' + job.Data);
        }
    }
    // 打印不存在的作业ID
    console.log("============ NonExistAIJobIds ============");
    if (response.NonExistAIJobIds && response.NonExistAIJobIds.String && response.NonExistA
IJobIds.String.length > 0) {
        for (var i=0; i<response.NonExistAIJobIds.String.length; i++) {
            // 作业ID
            console.log('NonExistAIJobId = ' + response.NonExistAIJobIds.String[i]);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("GetMediaDNAResult", {
    MediaId: '584625a507a44d0eaa7424afd56****'
}, {}).then(function (response) {
    // 打印DNA结果
    console.log("DNAResult = ");
    console.log(response.DNAResult);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.15. 直播转点播

本篇文档提供了Node.js SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("ListLiveRecordVideo", {
    StartTime: '2018-10-18T14:01:57Z',
    EndTime: '2018-10-20T14:01:57Z',
    StreamName: 'testStream****',
    DomainName: 'example.aliyundoc.com',
    AppName: 'testApp****'
}, {}).then(function (response) {
    if (response.LiveRecordVideoList && response.LiveRecordVideoList.LiveRecordVideo){
        console.log('LiveRecordVideoList.size = ' + response.LiveRecordVideoList.LiveRecord
Video.length);
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

6.16. 点播CDN

本篇文档提供了Node.js SDK点播CDN模块相关功能的API调用示例。包含预热缓存、刷新缓存、查询刷新和预热状态、查询刷新预热次数限制和余量、查询流量数据、查询网络带宽、下载域名日志。

初始化客户端

使用前请先初始化客户端，请参见初始化。

预热缓存

调用PreloadVodObjectCaches接口，完成预热缓存功能。

接口参数和返回字段请参见PreloadVodObjectCaches。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("PreloadVodObjectCaches", {
    ObjectPath: 'http://192.168.0.0/16/video/789123x-1688e7fd1ca-0005-36ec-adb-83212.mp4'
// 需要预热的文件路径
}, {}).then(function (response) {
    // 预热返回的任务ID
    console.log('PreloadTaskId = ' + response.PreloadTaskId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

刷新缓存

调用RefreshVodObjectCaches接口，完成刷新缓存功能。

接口参数和返回字段请参见RefreshVodObjectCaches。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("RefreshVodObjectCaches", {
    ObjectPath: 'http://192.168.0.0/16/video/83213214-1688e7fd1ca-0005-36ec-adb-8512.mp4',
// 需要刷新的文件或者目录路径
    ObjectType: 'File'  // 刷新的类型
}, {}).then(function (response) {
    // 刷新返回的任务ID
    console.log('RefreshTaskId = ' + response.RefreshTaskId);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询刷新和预热状态

调用DescribeVodRefreshTasks接口，完成查询刷新和预热状态功能。

接口参数和返回字段请参见DescribeVodRefreshTasks。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DescribeVodRefreshTasks", {
    DomainName: '192.168.0.0/example.com',  // 需要查询的域名
    ObjectType: 'File'  // 任务类型
}, {}).then(function (response) {
    console.log(response);
    // 整页大小
    console.log('PageSize = ' + response.PageSize);
    // 页码
    console.log('PageNumber = ' + response.PageNumber);
    // 总条数
    console.log('TotalCount = ' + response.TotalCount);
    // 任务信息
    if (response.Tasks && response.Tasks.Task && response.Tasks.Task.length > 0){
        for (var i=0; i<response.Tasks.Task.length; i++){
            console.log('The ' + i + ' Task.');
            var task = response.Tasks.Task[i];
            // 任务ID
            console.log('TaskId = ' + task.TaskId);
            // 刷新对象路径
            console.log('ObjectPath = ' + task.ObjectPath);
            // 状态
            console.log('Status = ' + task.Status);
            // 进度百分比
            console.log('Process = ' + task.Process);
            // 任务类型
            console.log('ObjectType = ' + task.ObjectType);
            // 任务对象创建时间，使用UTC时间
            console.log('CreationTime = ' + task.CreationTime);
            // 刷新预热失败返回的错误描述
            console.log('Description = ' + task.Description);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询刷新预热次数限制和余量

调用DescribeVodRefreshQuota接口，完成查询刷新预热次数限制和余量功能。

接口参数和返回字段请参见DescribeVodRefreshQuota。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DescribeVodRefreshQuota", {}, {}).then(function (response) {
    // 当日URL刷新数量上限
    console.log('UrlQuota = ' + response.UrlQuota);
    // 当日目录刷新数量上限
    console.log('DirQuota = ' + response.DirQuota);
    // 当日剩余URL刷新数量
    console.log('UrlRemain = ' + response.UrlRemain);
    // 当日剩余目录刷新数量
    console.log('DirRemain = ' + response.DirRemain);
    // 当天预热数量上限
    console.log('PreloadQuota = ' + response.PreloadQuota);
    // 当天剩余预热数量
    console.log('PreloadRemain = ' + response.PreloadRemain);
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询流量数据

调用DescribeVodDomainTrafficData接口，完成查询流量数据功能。

接口参数和返回字段请参见DescribeVodDomainTrafficData。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DescribeVodDomainTrafficData", {
    DomainName: '192.168.0.0/example.com',    // 域名
    StartTime: '2019-01-15T15:59:59Z',  // 开始时间，请使用UTC格式
    EndTime: '2019-01-20T15:59:59Z'     // 结束时间，请使用UTC格式
}, {}).then(function (response) {
    // 域名
    console.log('DomainName = ' + response.DomainName);
    // 开始时间
    console.log('StartTime = ' + response.StartTime);
    // 结束时间
    console.log('EndTime = ' + response.EndTime);
    // 时间间隔
    console.log('DataInterval = ' + response.DataInterval);
    // 流量数据
    if (response.TrafficDataPerInterval && response.TrafficDataPerInterval.DataModule && re
sponse.TrafficDataPerInterval.DataModule.length > 0){
        for(var i=0; i<response.TrafficDataPerInterval.DataModule.length; i++){
            var trafficData = response.TrafficDataPerInterval.DataModule[i];
            console.log("The " + i + " TrafficData.");
            console.log('TimeStamp = ' + trafficData.TimeStamp);
            console.log('Value = ' + trafficData.Value);
            console.log('DomesticValue = ' + trafficData.DomesticValue);
            console.log('OverseasValue = ' + trafficData.OverseasValue);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

查询网络带宽

调用DescribeVodDomainBpsData接口，完成查询网络带宽功能。

接口参数和返回字段请参见DescribeVodDomainBpsData。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DescribeVodDomainBpsData", {
    DomainName: '192.168.0.0/example.com',    // 域名
    StartTime: '2019-01-15T15:59:59Z',  // 开始时间，请使用UTC格式
    EndTime: '2019-01-20T15:59:59Z'     // 结束时间，请使用UTC格式
}, {}).then(function (response) {
    // 域名
    console.log('DomainName = ' + response.DomainName);
    // 开始时间
    console.log('StartTime = ' + response.StartTime);
    // 结束时间
    console.log('EndTime = ' + response.EndTime);
    // 时间间隔
    console.log('DataInterval = ' + response.DataInterval);
    // 带宽数据
    if (response.BpsDataPerInterval && response.BpsDataPerInterval.DataModule && response.B
psDataPerInterval.DataModule.length > 0){
        for(var i=0; i<response.BpsDataPerInterval.DataModule.length; i++){
            var bpsData = response.BpsDataPerInterval.DataModule[i];
            console.log("The " + i + " BpsData.");
            console.log('TimeStamp = ' + bpsData.TimeStamp);
            console.log('Value = ' + bpsData.Value);
            console.log('DomesticValue = ' + bpsData.DomesticValue);
            console.log('OverseasValue = ' + bpsData.OverseasValue);
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

下载域名日志

调用DescribeVodDomainLog接口，完成下载域名日志功能。

接口参数和返回字段请参见DescribeVodDomainLog。调用示例如下：

```
// 调用样例
var client = initVodClient('<Your AccessKeyId>','<Your AccessKeySecret>');

client.request("DescribeVodDomainLog", {
    DomainName: '192.168.0.0/example.com',    // 域名
    // PageSize: 300,  // 分页大小
    // PageNumber: 1,  // 分页页号
    StartTime: '2019-01-15T15:59:59Z',  // 开始时间，请使用UTC格式
    EndTime: '2019-01-20T15:59:59Z'     // 结束时间，请使用UTC格式
}, {}).then(function (response) {
    // CDN日志详细数据
    console.log(response.DomainLogDetails);
    if (response.DomainLogDetails && response.DomainLogDetails.DomainLogDetail && response.
DomainLogDetails.DomainLogDetail.length > 0){
        for(var i=0; i<response.DomainLogDetails.DomainLogDetail.length; i++){
            var logData = response.DomainLogDetails.DomainLogDetail[i];
            console.log("The " + i + " LogData.");
            console.log('DomainName = ' + logData.DomainName);
            console.log('LogCount = ' + logData.LogCount);
            console.log('=== PageInfoDetail Data ===');
            console.log('PageNumber = ' + logData.PageInfos.PageNumber);
            console.log('PageSize = ' + logData.PageInfos.PageSize);
            console.log('Total = ' + logData.PageInfos.Total);
            console.log('=== LogInfoDetail Data ===');
            if (logData.LogInfos && logData.LogInfos.LogInfoDetail && logData.LogInfos.LogI
nfoDetail.length > 0){
                for (var j=0; j<logData.LogInfos.LogInfoDetail.length; j++){
                    var logInfoDetail = logData.LogInfos.LogInfoDetail[j]
                    console.log('LogName = ' + logInfoDetail.LogName);
                    console.log('LogPath = ' + logInfoDetail.LogPath);
                    console.log('LogSize = ' + logInfoDetail.LogSize);
                    console.log('StartTime = ' + logInfoDetail.StartTime);
                    console.log('EndTime = ' + logInfoDetail.EndTime);
                }
            }
        }
    }
    console.log('RequestId = ' + response.RequestId);
}).catch(function (response) {
    console.log('ErrorCode = ' + response.data.Code);
    console.log('ErrorMessage = ' + response.data.Message);
    console.log('RequestId = ' + response.data.RequestId);
});
```

7.Go SDK

7.1. 安装

本文为您介绍了视频点播服务端Go SDK的安装及更新步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

安装Go 1.7及以上版本，请参见Go官网下载合适的版本安装。

安装SDK

视频点播服务端Go SDK支持使用 `go get` 或 `glide` 安装，可以根据您的实际需要选择。

- 使用 `go get` 安装SDK。示例如下：

```
go get -u github.com/aliyun/alibaba-cloud-sdk-go/sdk
```

> 说明
> - 安装过程中，界面不会打印提示，请耐心等待。如发生超时，请再次执行命令。
> - 当*GOPATH*对应的目录下出现了子目录*src/github.com/aliyun/alibaba-cloud-sdk-go/services/vod*即表示SDK安装成功。

- 使用 `glide` 安装SDK。示例如下：

```
glide get github.com/aliyun/alibaba-cloud-sdk-go
```

更新SDK

若发现新的接口或已有接口新的功能在当前SDK版本没有，可更新到最新版，方法与安装SDK类似。

后续步骤

安装完成后，需要进行初始化。Go SDK可通过AccessKey或STS Security Token初始化。具体操作，请参见初始化

7.2. 初始化

Go SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端Go SDK。安装步骤，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。示例如下：

```
package main
import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)
func InitVodClient(accessKeyId  string, accessKeySecret string) (client *vod.Client, err er
ror) {
    // 点播服务接入地域
    regionId := "cn-shanghai"
    // 创建授权对象
    credential := &credentials.AccessKeyCredential{
        accessKeyId,
        accessKeySecret,
    }
    // 自定义config
    config := sdk.NewConfig()
    config.AutoRetry = true       // 失败是否自动重试
    config.MaxRetryTime = 3       // 最大重试次数
    config.Timeout = 3000000000   // 连接超时，单位：纳秒；默认为3秒
    // 创建vodClient实例
    return vod.NewClientWithOptions(regionId, config, credential)
}
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例如下：

```
package main
import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)
func InitVodClient(accessKeyId  string, accessKeySecret string, securityToken string) (clie
nt *vod.Client, err error) {
    // 点播服务接入地域
    regionId := "cn-shanghai"
    // 创建授权对象
    credential := &credentials.StsTokenCredential{
        accessKeyId,
        accessKeySecret,
        securityToken,
    }
    // 连接设置
    config := sdk.NewConfig()
    config.AutoRetry = true       // 失败是否自动重试
    config.MaxRetryTime = 3       // 最大重试次数
    config.Timeout = 3000000000  // 连接超时，单位：纳秒；默认为3秒
    // 创建vodClient实例
    return vod.NewClientWithOptions(regionId, config, credential)
}
```

使用说明

创建API请求时，调用的方法名为 `vod.Create${apiName}Request` ，其中 `${apiName}` 需要替换成要使用的视频点播API，视频点播API名称请参见API概览。

调用示例

调用获取播放地址接口GetPlayInfo为例。示例如下：

```
func MyGetPlayInfo(client *vod.Client, videoId string) (response *vod.GetPlayInfoResponse,
err error) {
    // 创建API请求并设置参数，调用vod.Create${apiName}Request
    request := vod.CreateGetPlayInfoRequest()
    request.VideoId = videoId
    request.AcceptFormat = "JSON"
    // 发起请求并处理异常，调用client.${apiName}(request)
    return client.GetPlayInfo(request)
}
func main() {
    client, err := InitVodClient("<AccessKeyId>", "<AccessKeySecret>")
    if err != nil {
        // 异常处理
        panic(err)
    }
    response, err := MyGetPlayInfo(client, "<videoId>")
    if err != nil {
        // 异常处理
        panic(err)
    }
    // fmt.Println(response)
    playList := response.PlayInfoList.PlayInfo
    for _, playInfo := range playList {
        // 打印清晰度和对应的播放地址
        fmt.Printf("%s: %s\n", playInfo.Definition, playInfo.PlayURL)
    }
}
```

说明

- 获取更多信息，请参见阿里云SDK开发指南-使用Go SDK。
- Go SDK同时支持并发调用和泛化调用。
- 示例代码下载：test_vod.go。

7.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍Go SDK调用服务端上传接口的使用场景和调用示例。

使用场景

注意

本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyCreateUploadVideo(client *vod.Client) (response *vod.CreateUploadVideoResponse, err
error) {
    request := vod.CreateCreateUploadVideoRequest()

    request.Title = "Sample Video Title"
    request.Description = "Sample Description"
    request.FileName = "/opt/video/sample/video_file.mp4"
    //request.CateId = "-1"
    request.CoverURL = "http://192.168.0.0/16/tps/TB1qnJ1PVXXXXXCXXXXXXXXXXXX-700-700.png"
    request.Tags = "tag1,tag2"

    request.AcceptFormat = "JSON"
    return client.CreateUploadVideo(request)
}


func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyCreateUploadVideo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("VideoId: %s\n UploadAddress: %s\n UploadAuth: %s",
        response.VideoId, response.UploadAddress, response.UploadAuth)
}
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyRefreshUploadVideo(client *vod.Client) (response *vod.RefreshUploadVideoResponse, er
r error) {
    request := vod.CreateRefreshUploadVideoRequest()
    request.VideoId = "6657f89a86fa4f76a295ae95636e****"
    request.AcceptFormat = "JSON"

    return client.RefreshUploadVideo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyRefreshUploadVideo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("UploadAddress: %s\n UploadAuth: %s", response.UploadAddress, response.Uploa
dAuth)
}
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyCreateUploadImage(client *vod.Client) (response *vod.CreateUploadImageResponse, err
error) {
    request := vod.CreateCreateUploadImageRequest()

    request.ImageType = "cover"
    request.ImageExt = "jpg"
    request.Title = "Sample Image Title"
    //request.CateId = "-1"
    request.Tags = "tag1,tag2"

    request.AcceptFormat = "JSON"
    return client.CreateUploadImage(request)
}


func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyCreateUploadImage(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("ImageId: %s\n ImageURL: %s\n UploadAddress: %s\n UploadAuth: %s",
        response.ImageId, response.ImageURL, response.UploadAddress, response.UploadAuth)
}
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyCreateUploadAttachedMedia(client *vod.Client) (response *vod.CreateUploadAttachedMed
iaResponse, err error) {
    request := vod.CreateCreateUploadAttachedMediaRequest()

    request.BusinessType = "watermark"
    request.MediaExt = "gif"
    request.Title = "Sample watermark Title"
    //request.CateId = "-1"
    request.Tags = "tag1,tag2"

    request.AcceptFormat = "JSON"
    return client.CreateUploadAttachedMedia(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyCreateUploadAttachedMedia(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("MediaId: %s\n MediaURL: %s\n FileURL: %s\n UploadAddress: %s\n UploadAuth:
%s",
        response.MediaId, response.MediaURL, response.FileURL, response.UploadAddress, resp
onse.UploadAuth)
}
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
```

```
    "encoding/json"
    "net/url"
    "strings"
)

func MyUploadMediaByURL(client *vod.Client) (response *vod.UploadMediaByURLResponse, err er
ror) {
    request := vod.CreateUploadMediaByURLRequest()

    // 对URL进行编码
    sourceUrls := []string{"https://exampleBucket****.oss-cn-shanghai.aliyuncs.com/vod_vide
o****.mp4",
        "https://exampleBucket****.oss-cn-shanghai.aliyuncs.com/vod_video****.flv"}
    uploadUrls := []string{}
    metadatas := []map[string]interface{}{}
    for _, surl := range sourceUrls {
        encodeUrl := url.QueryEscape(surl)
        uploadUrls = append(uploadUrls, encodeUrl)
        metadata := map[string]interface{}{"SourceURL": encodeUrl, "Title":"UploadMediaByUR
L Sample Title"}
        metadatas = append(metadatas, metadata)
    }

    // 设置上传的URL列表，用逗号分隔
    request.UploadURLs = strings.Join(uploadUrls, ",")

    // 可选项，设置URL对应的Meta信息
    jsonMetas, err := json.Marshal(metadatas)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.UploadMetadatas = string(jsonMetas)

    // 可选项，设置转码模板组
    //request.TemplateGroupId = "<TemplateGroupId>"

    request.AcceptFormat = "JSON"
    return client.UploadMediaByURL(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUploadMediaByURL(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    for _, job := range response.UploadJobs {
```

```
        fmt.Printf("%s: %s\n", job.JobId, job.SourceURL)
    }
}
```

获取URL上传信息

调用GetURLUploadInfos接口，完成获取URL上传信息功能。

接口参数和返回字段请参见GetURLUploadInfos。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "strings"
)

func MyGetURLUploadInfos(client *vod.Client) (response *vod.GetURLUploadInfosResponse, err
error) {
    request := vod.CreateGetURLUploadInfosRequest()

    // 对URL进行编码
    sourceUrls := []string{"https://192.168.0.0/16/vod_sample1.mp4",
        "https://exampleBucket****.oss-cn-shanghai.aliyuncs.com/vod_video****.flv"}
    uploadUrls := []string{}
    for _, surl := range sourceUrls {
        encodeUrl := url.QueryEscape(surl)
        uploadUrls = append(uploadUrls, encodeUrl)
    }

    // 设置上传的URL列表，用逗号分隔
    request.UploadURLs = strings.Join(uploadUrls, ",")

    // 也可以传入jobId查询
    //request.JobIds = "jobId1,jobId2"

    request.AcceptFormat = "JSON"
    return client.GetURLUploadInfos(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetURLUploadInfos(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.NonExists)
    for _, uploadInfo := range response.URLUploadInfoList {
        fmt.Printf("%s: %s %s\n", uploadInfo.UploadURL, uploadInfo.Status, uploadInfo.Media
Id)
    }
}
```

7.4. 音视频播放

本篇文档提供了Go SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetPlayInfo(client *vod.Client, videoId string) (response *vod.GetPlayInfoResponse,
err error) {
    request := vod.CreateGetPlayInfoRequest()
    request.VideoId = videoId
    request.AcceptFormat = "JSON"

    return client.GetPlayInfo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetPlayInfo(client, "<videoId>")
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    playList := response.PlayInfoList.PlayInfo
    for _, playInfo := range playList {
        fmt.Printf("%s: %s\n", playInfo.Definition, playInfo.PlayURL)
    }
}
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetPlayAuth(client *vod.Client, videoId string) (response *vod.GetVideoPlayAuthRespo
nse, err error) {
    request := vod.CreateGetVideoPlayAuthRequest()
    request.VideoId = videoId
    request.AcceptFormat = "JSON"

    return client.GetVideoPlayAuth(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetPlayAuth(client, "<videoId>")
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("%s: %s\n", response.VideoMeta, response.PlayAuth)
}
```

7.5. 媒资管理

本篇文档提供了Go SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySearchMedia(client *vod.Client) (response *vod.SearchMediaResponse, err error) {
    request := vod.CreateSearchMediaRequest()
    request.Fields = "Title,CoverURL,Status"
    request.Match = "Status in ('Normal','Checking') and CreationTime = ('2018-07-01T08:00:
00Z','2018-08-01T08:00:00Z')"
    request.PageNo = "1"
    request.PageSize = "10"
    request.SearchType = "video"
    request.SortBy = "CreationTime:Desc"
    request.AcceptFormat = "JSON"

    return client.SearchMedia(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySearchMedia(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    //fmt.Println(response.MediaList)
}
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetVideoInfo(client *vod.Client) (response *vod.GetVideoInfoResponse, err error) {
    request := vod.CreateGetVideoInfoRequest()
    request.VideoId = "<videoId>"
    request.AcceptFormat = "JSON"

    return client.GetVideoInfo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetVideoInfo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    //fmt.Println(response.Video)
}
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetVideoInfos(client *vod.Client) (response *vod.GetVideoInfosResponse, err error) {
    request := vod.CreateGetVideoInfosRequest()
    request.VideoIds = "VideoId1,VideoId2"
    request.AcceptFormat = "JSON"

    return client.GetVideoInfos(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetVideoInfos(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.NonExistVideoIds)
    //fmt.Println(response.VideoList)
}
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyUpdateVideoInfo(client *vod.Client) (response *vod.UpdateVideoInfoResponse, err erro
r) {
    request := vod.CreateUpdateVideoInfoRequest()
    request.VideoId = "<videoId>"
    request.Title = "new Title"
    request.Description = "new Description"
    request.Tags = "new Tag1,new Tag2"
    request.AcceptFormat = "JSON"

    return client.UpdateVideoInfo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateVideoInfo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyUpdateVideoInfos(client *vod.Client) (response *vod.UpdateVideoInfosResponse, err er
ror) {
    request := vod.CreateUpdateVideoInfosRequest()

    updateContent := []map[string]interface{}{}
    updateItem1 := map[string]interface{}{"VideoId":"<VideoId1>", "Title":"New Title 1", "T
ags": "NewTag1,NewTag2"}
    updateItem2 := map[string]interface{}{"VideoId":"<VideoId2>", "Title":"New Title 2", "T
ags": "NewTag1,NewTag2"}
    updateContent = append(updateContent, updateItem1, updateItem2)
    jsonContent, err := json.Marshal(updateContent)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.UpdateContent = string(jsonContent)
    request.AcceptFormat = "JSON"

    return client.UpdateVideoInfos(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateVideoInfos(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.NonExistVideoIds)
    fmt.Println(response.RequestId)
}
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteVideo(client *vod.Client) (response *vod.DeleteVideoResponse, err error) {
    request := vod.CreateDeleteVideoRequest()
    // 支持批量删除视频，多个用逗号分隔
    request.VideoIds = "VideoId1,VideoId2"
    request.AcceptFormat = "JSON"

    return client.DeleteVideo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteVideo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

获取源文件信息（含原片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetMezzanineInfo(client *vod.Client) (response *vod.GetMezzanineInfoResponse, err er
ror) {
    request := vod.CreateGetMezzanineInfoRequest()
    request.VideoId = "<VideoId>"
    request.AuthTimeout = "86400"
    //request.OutputType = "oss"
    request.AcceptFormat = "JSON"

    return client.GetMezzanineInfo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetMezzanineInfo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.Mezzanine.FileURL)
}
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

> **注意**
>
> 注意时间参数要使用UTC时间，北京时间与UTC的时差为+8，也就是UTC+8，例如 `2018-01-01T12:00:00Z` 表示北京时间 `2018年1月1日20点0分0秒` 。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetVideoList(client *vod.Client) (response *vod.GetVideoListResponse, err error) {
    request := vod.CreateGetVideoListRequest()
    request.StartTime = "2018-12-01T06:00:00Z"
    request.EndTime = "2018-12-25T06:00:00Z"
    //request.Status = "Uploading,Normal,Transcoding"
    request.PageNo = "1"
    request.PageSize = "10"
    request.SortBy = "CreationTime:Desc"
    request.AcceptFormat = "JSON"

    return client.GetVideoList(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetVideoList(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    videoList := response.VideoList.Video
    for _, video := range videoList {
        fmt.Printf("%s: %s\n", video.VideoId, video.Title)
    }
}
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteStream(client *vod.Client) (response *vod.DeleteStreamResponse, err error) {
    request := vod.CreateDeleteStreamRequest()
    request.VideoId = "<VideoId>"
    request.JobIds = "JobId1,JobId2"
    request.AcceptFormat = "JSON"

    return client.DeleteStream(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteStream(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteMezzanines(client *vod.Client) (response *vod.DeleteMezzaninesResponse, err er
ror) {
    request := vod.CreateDeleteMezzaninesRequest()
    request.VideoIds = "VideoId1,VideoId2"
    request.Force = "false"
    request.AcceptFormat = "JSON"

    return client.DeleteMezzanines(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteMezzanines(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.NonExistVideoIds)
}
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyUpdateImageInfos(client *vod.Client) (response *vod.UpdateImageInfosResponse, err er
ror) {
    request := vod.CreateUpdateImageInfosRequest()

    updateContent := []map[string]interface{}{}
    updateItem1 := map[string]interface{}{"ImageId":"<ImageId1>", "Title":"New Title 1", "T
ags": "NewTag1,NewTag2"}
    updateItem2 := map[string]interface{}{"ImageId":"<ImageId2>", "Title":"New Title 2", "T
ags": "NewTag1,NewTag2"}
    updateContent = append(updateContent, updateItem1, updateItem2)
    jsonContent, err := json.Marshal(updateContent)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.UpdateContent = string(jsonContent)
    request.AcceptFormat = "JSON"

    return client.UpdateImageInfos(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateImageInfos(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.NonExistImageIds)
}
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetImageInfo(client *vod.Client) (response *vod.GetImageInfoResponse, err error) {
    request := vod.CreateGetImageInfoRequest()
    request.ImageId = "<ImageId>"
    request.AuthTimeout = "86400"
    request.AcceptFormat = "JSON"

    return client.GetImageInfo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetImageInfo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.ImageInfo.URL)
}
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteImage(client *vod.Client) (response *vod.DeleteImageResponse, err error) {
    request := vod.CreateDeleteImageRequest()

    // 根据ImageId删除图片文件
    request.DeleteImageType = "ImageId"
    request.ImageIds = "ImageId1,ImageId2"

    /*
    // 根据ImageURL删除图片文件
    request.DeleteImageType = "ImageURL"
    //Image URL示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com.example****.gif
    request.ImageURLs = "<your Image URL>"

    // 根据VideoId删除指定ImageType的图片文件
    request.DeleteImageType = "VideoId"
    request.VideoId = "<VideoId>"
    request.ImageType = "SpriteSnapshot"
    */

    request.AcceptFormat = "JSON"

    return client.DeleteImage(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteImage(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

7.6. 媒资分类

本文为您提供了Go SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyAddCategory(client *vod.Client) (response *vod.AddCategoryResponse, err error) {
    request := vod.CreateAddCategoryRequest()
    request.CateName = "sample category name"
    request.ParentId = "-1"
    request.AcceptFormat = "JSON"

    return client.AddCategory(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddCategory(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.Category.CateId)
}
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyUpdateCategory(client *vod.Client) (response *vod.UpdateCategoryResponse, err error)
{
    request := vod.CreateUpdateCategoryRequest()
    request.CateId = "<CateId>"
    request.CateName = "sample category name"
    request.AcceptFormat = "JSON"

    return client.UpdateCategory(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateCategory(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
}
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteCategory(client *vod.Client) (response *vod.DeleteCategoryResponse, err error)
{
    request := vod.CreateDeleteCategoryRequest()
    request.CateId = "<CateId>"
    request.AcceptFormat = "JSON"

    return client.DeleteCategory(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteCategory(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
}
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetCategories(client *vod.Client) (response *vod.GetCategoriesResponse, err error) {
    request := vod.CreateGetCategoriesRequest()
    request.CateId = "<CateId>"
    request.PageNo = "1"
    request.PageSize = "20"
    request.AcceptFormat = "JSON"

    return client.GetCategories(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetCategories(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
}
```

7.7. 媒体处理

本文提供了Go SDK媒体处理模块相关功能的API调用示例，包含提交转码作业（HLS标准加密、普通）、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业-普通

调用SubmitTranscodeJobs接口，完成提交转码作业-普通功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为普通转码（不加密）。阿里云视频加密方案请参见阿里云视频加密。

```
package main
```

```
import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func NormalSubmitTranscodeJobs(client *vod.Client) (response *vod.SubmitTranscodeJobsRespon
se, err error) {
    request := vod.CreateSubmitTranscodeJobsRequest()

    // 需要转码的视频ID
    request.VideoId = "<VideoId>"

    // 指定转码模板组
    request.TemplateGroupId = "<TemplateGroupId>"

    /*
    // 可选设置覆盖参数，如替换水印
    overrideParams := BuildOverrideParams()
    jsonParams, err := json.Marshal(overrideParams)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.OverrideParams = string(jsonParams)
    fmt.Println(request.OverrideParams)
    */

    request.AcceptFormat = "JSON"

    return client.SubmitTranscodeJobs(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := NormalSubmitTranscodeJobs(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.TranscodeJobs.TranscodeJob {
        fmt.Printf("%s\n", job.JobId)
    }
}

/*
```

```
/*
* 构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；如不使用可忽略
* 需要替换的水印信息对应水印ID必须是关联在指定的转码模板组ID上(即TranscodeTemplateId)
* 不支持通过本接口去增加一个没有关联上的水印
*/
func BuildOverrideParams() (overrideParams map[string]interface{}) {
    // 以替换水印为例
    watermarks := []map[string]interface{}{}

    // 替换图片水印文件，FileURL示例：https://example-bucket-****.oss-cn-shanghai.aliyuncs.com
/watermarks/sample.png
    watermark1 := map[string]interface{}{"WatermarkId": "<WatermarkId>", "FileUrl": "<your
File URL>"}
    watermarks = append(watermarks, watermark1)

    // 替换文字水印
    watermark2 := map[string]interface{}{"WatermarkId": "<WatermarkId>", "Content": "new Te
xt"}
    watermarks = append(watermarks, watermark2)

    overrideParams = map[string]interface{}{"Watermarks": watermarks}

    return overrideParams
}
```

提交转码作业-HLS标准加密

调用SubmitTranscodeJobs接口，完成提交转码作业-HLS标准加密功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

> **说明**
>
> 此示例代码为使用HLS标准加密提交转码作业的示例。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/kms"
    "fmt"
)

func HLSEncryptSubmitTranscodeJobs(client *vod.Client, kmsClient *kms.Client) (response *vo
d.SubmitTranscodeJobsResponse, err error) {
    request := vod.CreateSubmitTranscodeJobsRequest()

    // 需要转码的视频ID
    request.VideoId = "<VideoId>"

    // 指定转码模板组
```

```
    request.TemplateGroupId = "<TemplateGroupId>"

    // 使用KMS生成随机的加密密钥
    encryptConfig, err := BuildEncryptConfig(kmsClient)
    if err != nil {
        panic(err)
    }
    jsonConfig, err := json.Marshal(encryptConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.EncryptConfig = string(jsonConfig)

    request.AcceptFormat = "JSON"

    return client.SubmitTranscodeJobs(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }
    kmsClient, err := InitKmsClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := HLSEncryptSubmitTranscodeJobs(client, kmsClient)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.TranscodeJobs.TranscodeJob {
        fmt.Printf("%s\n", job.JobId)
    }
}

/*
* 生成HLS标准加密参数的可选配置，如不使用可忽略
* 此时依赖KMS服务，需要安装KMS依赖包 aliyun-python-sdk-kms，并引入相关类
* 生成密钥API参考：
GenerateDataKey

*/
// 初始化KMS客户端，与点播客户端类似
func InitKmsClient(accessKeyId  string, accessKeySecret string) (client *kms.Client, err er
ror) {

    // KMS接入区域，与点播位于同一区域
    regionId := "cn-shanghai"
```

```
    // 创建授权对象
    credential := &credentials.AccessKeyCredential{
        accessKeyId,
        accessKeySecret,
    }

    // 自定义config
    config := sdk.NewConfig()
    config.AutoRetry = true       // 失败是否自动重试
    config.MaxRetryTime = 3       // 最大重试次数
    config.Timeout = 3000000000   // 连接超时，单位：纳秒；默认为3秒

    // 创建kmsClient实例
    return kms.NewClientWithOptions(regionId, config, credential)
}

// 随机生成加密的密钥
func BuildEncryptConfig(client *kms.Client) (encryptConfig map[string]interface{}, err erro
r) {
    request := kms.CreateGenerateDataKeyRequest()

    // 随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥，
    // 视频标准加密只需要传递密文密钥即可，以保证安全
    request.KeyId = "<serviceKey>"
    request.KeySpec = "AES_128"
    request.AcceptFormat = "JSON"
    request.Scheme = "HTTPS"        // KMS接口必须使用HTTPS

    response, err := client.GenerateDataKey(request)
    if err != nil {
        panic(err)
    }

    // 解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥都不
一样；注意您需要自己部署解密服务
    // Ciphertext作为解密接口的参数名称，可自定义，此处只作为参考
    decryptKeyUri := "http://example.aliyundoc.com/decrypt?" + "Ciphertext=" + response.Cip
hertextBlob
    encryptConfig = map[string]interface{}{"DecryptKeyUri": decryptKeyUri, "KeyServiceType"
: "KMS",
        "CipherText": response.CiphertextBlob}

    return  encryptConfig, nil
}
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

说明

创建截图模板详细请参见截图模板。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MySubmitSnapshotJob(client *vod.Client) (response *vod.SubmitSnapshotJobResponse, err
error) {
    request := vod.CreateSubmitSnapshotJobRequest()

    // 需要截图的视频ID
    request.VideoId = "<VideoId>"

    // 截图模板ID
    //request.SnapshotTemplateId = "<snapshotTemplateId>"

    // 如果设置了截图模板ID，则会忽略以下参数
    request.Count = "50"
    request.SpecifiedOffsetTime = "0"
    request.Interval = "1"
    request.Width = "200"
    request.Height = "200"

    // 雪碧图参数，如不使用可不设置
    spriteSnapshotConfig := map[string]interface{}{"CellWidth": 120, "CellHeight": 68, "Col
umns": 3,
        "Lines": 10, "Padding": 20, "Margin": 50}
    // 保留雪碧图原始图
    spriteSnapshotConfig["KeepCellPic"] = "keep"
    spriteSnapshotConfig["Color"] = "tomato"
    jsonSpriteConfig, err := json.Marshal(spriteSnapshotConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.SpriteSnapshotConfig = string(jsonSpriteConfig)

    request.AcceptFormat = "JSON"

    return client.SubmitSnapshotJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
```

```
        panic(err)
    }

    response, err := MySubmitSnapshotJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.SnapshotJob.JobId)
}
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListSnapshots(client *vod.Client) (response *vod.ListSnapshotsResponse, err error) {
    request := vod.CreateListSnapshotsRequest()

    request.VideoId = "<VideoId>"
    request.SnapshotType = "CoverSnapshot"
    request.PageNo = "1"
    request.PageSize = "20"
    request.AuthTimeout = "86400"

    request.AcceptFormat = "JSON"

    return client.ListSnapshots(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListSnapshots(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.MediaSnapshot.Total)
    for _, snapshot := range response.MediaSnapshot.Snapshots.Snapshot {
        fmt.Printf("%d: %s\n", snapshot.Index, snapshot.Url)
    }
}
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MySubmitPreprocessJobs(client *vod.Client) (response *vod.SubmitPreprocessJobsResponse
, err error) {
    request := vod.CreateSubmitPreprocessJobsRequest()

    request.VideoId = "<VideoId>"
    request.PreprocessType = "LivePreprocess"

    request.AcceptFormat = "JSON"

    return client.SubmitPreprocessJobs(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySubmitPreprocessJobs(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.PreprocessJobs.PreprocessJob {
        fmt.Printf("%s\n", job.JobId)
    }
}
```

7.8. 视频水印

本篇文档提供了Go SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印、设置默认水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

说明

- 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
- 水印文件OSS上传详细参数请参见OSS文件上传。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyAddWatermark(client *vod.Client) (response *vod.AddWatermarkResponse, err error) {
    request := vod.CreateAddWatermarkRequest()

    request.Name = "watermark-sample"

    // 图片水印必须传图片文件的OSS地址，且水印文件需要和视频在同一区域，如:华东2的视频添加水印，水印文件必须存放在华东2
    //FileURL示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermark/test****.png
    request.FileUrl = "<your File URL>"

    // 文字水印设置示例
    request.Type = "Text"
    // 水印的文本内容、字体、大小、颜色、透明度等配置
    watermarkConfig := map[string]interface{}{"Content": "watermark Text", "FontName": "SimSun", "FontSize": 25,
        "FontColor": "Black", "FontAlpha": 0.2, "BorderColor": "White", "BorderWidth": 1, "Top": 20, "Left": 15}
    jsonConfig, err := json.Marshal(watermarkConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.WatermarkConfig = string(jsonConfig)

    /*
    // 图片水印设置示例
    request.Type = "Image"
    // 水印显示的开始时间和结束时间
    timeline := map[string]interface{}{"Start": 2, "Duration": "ToEND"}
    // 水印的位置等配置
    watermarkConfig := map[string]interface{}{"Dx": 8, "Dy": 8, "Width": 55, "Height": 55, "ReferPos": "BottomRight", "Timeline": timeline}
    jsonConfig, err := json.Marshal(watermarkConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
```

```
    }
    request.WatermarkConfig = string(jsonConfig)
    */

    request.AcceptFormat = "JSON"

    return client.AddWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Printf("%s: %s\n", response.WatermarkInfo.WatermarkId, response.WatermarkInfo.Water
markConfig)
}
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

> **注意**
>
> 本接口不支持图片水印图片文件地址的修改，如要更换请创建新的水印。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyUpdateWatermark(client *vod.Client) (response *vod.UpdateWatermarkResponse, err erro
r) {
    request := vod.CreateUpdateWatermarkRequest()

    request.WatermarkId = "5a5abf8e458a9d5c97dea86e686c****"
    request.Name = "new-watermark-name"

    // 更改文字水印设置示例
```

```
    // 水印的文本内容、字体、大小、颜色、透明度等配置
    watermarkConfig := map[string]interface{}{"Content": "New watermark Text", "FontName":
"SimSun", "FontSize": 25,
        "FontColor": "Black", "FontAlpha": 0.2, "BorderColor": "White", "BorderWidth": 1, "
Top": 20, "Left": 15}
    jsonConfig, err := json.Marshal(watermarkConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.WatermarkConfig = string(jsonConfig)

    /*
    // 更改图片水印设置示例
    // 水印显示的开始时间和结束时间
    timeline := map[string]interface{}{"Start": 2, "Duration": "ToEND"}
    // 水印的位置等配置
    watermarkConfig := map[string]interface{}{"Dx": 8, "Dy": 8, "Width": 55, "Height": 55,
"ReferPos": "BottomRight",
        "Timeline": timeline}
    jsonConfig, err := json.Marshal(watermarkConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.WatermarkConfig = string(jsonConfig)
    */

    request.AcceptFormat = "JSON"

    return client.UpdateWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Printf("%s: %s\n", response.WatermarkInfo.WatermarkId, response.WatermarkInfo.Water
markConfig)
}
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyDeleteWatermark(client *vod.Client) (response *vod.DeleteWatermarkResponse, err erro
r) {
    request := vod.CreateDeleteWatermarkRequest()
    request.WatermarkId = "<WatermarkId>"

    request.AcceptFormat = "JSON"

    return client.DeleteWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyListWatermark(client *vod.Client) (response *vod.ListWatermarkResponse, err error) {
    request := vod.CreateListWatermarkRequest()

    request.AcceptFormat = "JSON"

    return client.ListWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, watermark := range response.WatermarkInfos {
        fmt.Printf("%s: %s %s\n", watermark.WatermarkId, watermark.Name, watermark.Watermar
kConfig)
    }
}
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyGetWatermark(client *vod.Client) (response *vod.GetWatermarkResponse, err error) {
    request := vod.CreateGetWatermarkRequest()
    request.WatermarkId = "<WatermarkId>"

    request.AcceptFormat = "JSON"

    return client.GetWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    watermark := response.WatermarkInfo
    fmt.Printf("%s: %s %s\n", response.WatermarkInfo.WatermarkId, watermark.Name, watermark
.WatermarkConfig)
}
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MySetDefaultWatermark(client *vod.Client) (response *vod.SetDefaultWatermarkResponse,
err error) {
    request := vod.CreateSetDefaultWatermarkRequest()
    request.WatermarkId = "<WatermarkId>"

    request.AcceptFormat = "JSON"

    return client.SetDefaultWatermark(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySetDefaultWatermark(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

7.9. 转码模板

本篇文档提供了Go SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
```

```
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyAddTranscodeTemplateGroup(client *vod.Client) (response *vod.AddTranscodeTemplateGro
upResponse, err error) {
    request := vod.CreateAddTranscodeTemplateGroupRequest()

    request.Name = "SampleTranscodeTemplateGroup"

    transcodeTemplateList := BuildTranscodeTemplateList()
    jsonTranscodeConfig, err := json.Marshal(transcodeTemplateList)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TranscodeTemplateList = string(jsonTranscodeConfig)

    request.AcceptFormat = "JSON"

    return client.AddTranscodeTemplateGroup(request)
}

// 构建需要添加的转码模板配置数据
func BuildTranscodeTemplateList() (transcodeTemplateList []map[string]interface{}) {

    transcodeTemplate := map[string]interface{}{}

    // 设置模板名称
    transcodeTemplate["TemplateName"] = "MP4-for-Low-Definition"

    // 清晰度
    transcodeTemplate["Definition"] = "LD"

    // 视频流转码配置
    videoConfig := map[string]interface{}{"Width": 640, "Bitrate": 400, "Fps": 25, "Remove"
: false, "Codec": "H.264", "Gop": "250"}
    transcodeTemplate["Video"] = videoConfig

    // 音频流转码配置
    audioConfig := map[string]interface{}{"Codec": "AAC", "Bitrate": "64", "Channels": "2",
"Samplerate": "32000"}
    transcodeTemplate["Audio"] = audioConfig

    // 封装容器
    container := map[string]interface{}{"Format": "mp4"}
    transcodeTemplate["Container"] = container

    // 条件转码配置
    condition := map[string]interface{}{"IsCheckReso": false, "IsCheckResoFail": false, "Is
CheckVideoBitrate": false,
        "IsCheckVideoBitrateFail": false, "IsCheckAudioBitrate": false, "IsCheckAudioBitrat
```

```
eFall": false}
    transcodeTemplate["TransConfig"] = condition

    /*
    // 加密配置(只支持HLS)
    encryptSetting := map[string]interface{}{"EncryptType": "Private"}
    transcodeTemplate["EncryptSetting"] = encryptSetting
    */

    // 水印ID(多水印关联)
    watermarkIdList := []string{"USER_DEFAULT_WATERMARK"}
    //watermarkIdList = append(watermarkIdList, "<WatermarkId>")
    transcodeTemplate["WatermarkIds"] = watermarkIdList

    transcodeTemplateList = append(transcodeTemplateList, transcodeTemplate)

    return transcodeTemplateList
}


func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.TranscodeTemplateGroupId)
}
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyUpdateTranscodeTemplateGroup(client *vod.Client) (response *vod.UpdateTranscodeTempl
ateGroupResponse, err error) {
    request := vod.CreateUpdateTranscodeTemplateGroupRequest()
```

```
    request.TranscodeTemplateGroupId = "<TranscodeTemplateGroupId>"

    // 修改模板组名称，可选
    request.Name = "new-SampleTranscodeTemplateGroup"

    // 修改模板组已有模板的配置，可选
    transcodeTemplateList := BuildNewTranscodeTemplateList()
    jsonTranscodeConfig, err := json.Marshal(transcodeTemplateList)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TranscodeTemplateList = string(jsonTranscodeConfig)

    request.AcceptFormat = "JSON"

    return client.UpdateTranscodeTemplateGroup(request)
}

// 构建需要修改的转码模板配置数据
func BuildNewTranscodeTemplateList() (transcodeTemplateList []map[string]interface{}) {

    transcodeTemplate := map[string]interface{}{}

    // 设置转码模板ID，必填
    transcodeTemplate["TranscodeTemplateId"] = "<TranscodeTemplateId>"

    // 以下都是可选项

    // 设置新的模板名称(如果需要更改)
    transcodeTemplate["TemplateName"] = "new-MP4-for-High-Definition"

    // 视频流转码配置
    videoConfig := map[string]interface{}{"Width": 640, "Bitrate": 500, "Fps": 60, "Remove"
: false, "Codec": "H.264", "Gop": "250"}
    transcodeTemplate["Video"] = videoConfig

    // 音频流转码配置
    audioConfig := map[string]interface{}{"Codec": "AAC", "Bitrate": "128", "Channels": "2"
, "Samplerate": "32000"}
    transcodeTemplate["Audio"] = audioConfig

    // 封装容器
    container := map[string]interface{}{"Format": "mp4"}
    transcodeTemplate["Container"] = container

    // 条件转码配置
    condition := map[string]interface{}{"IsCheckReso": false, "IsCheckResoFail": false, "Is
CheckVideoBitrate": false,
        "IsCheckVideoBitrateFail": false, "IsCheckAudioBitrate": false, "IsCheckAudioBitrat
eFail": false}
    transcodeTemplate["TransConfig"] = condition

    /*
```

```
    // 加密配置(只支持HLS)
    encryptSetting := map[string]interface{}{"EncryptType": "Private"}
    transcodeTemplate["EncryptSetting"] = encryptSetting
    */

    // 水印ID(多水印关联)
    watermarkIdList := []string{"USER_DEFAULT_WATERMARK"}
    //watermarkIdList = append(watermarkIdList, "<WatermarkId>")
    transcodeTemplate["WatermarkIds"] = watermarkIdList

    transcodeTemplateList = append(transcodeTemplateList, transcodeTemplate)

    return transcodeTemplateList
}


func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListTranscodeTemplateGroup(client *vod.Client) (response *vod.ListTranscodeTemplateG
roupResponse, err error) {
    request := vod.CreateListTranscodeTemplateGroupRequest()

    request.AcceptFormat = "JSON"

    return client.ListTranscodeTemplateGroup(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    for _, item := range response.TranscodeTemplateGroupList {
        fmt.Printf("%s: %s\n", item.TranscodeTemplateGroupId, item.Name)
    }
}
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetTranscodeTemplateGroup(client *vod.Client) (response *vod.GetTranscodeTemplateGro
upResponse, err error) {
    request := vod.CreateGetTranscodeTemplateGroupRequest()
    request.TranscodeTemplateGroupId = "<TranscodeTemplateGroupId>"

    request.AcceptFormat = "JSON"

    return client.GetTranscodeTemplateGroup(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    for _, item := range response.TranscodeTemplateGroup.TranscodeTemplateList {
        fmt.Printf("%s: %s\n", item.TranscodeTemplateId, item.TemplateName)
    }
}
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySetDefaultTranscodeTemplateGroup(client *vod.Client) (response *vod.SetDefaultTransc
odeTemplateGroupResponse, err error) {
    request := vod.CreateSetDefaultTranscodeTemplateGroupRequest()
    request.TranscodeTemplateGroupId = "<TranscodeTemplateGroupId>"

    request.AcceptFormat = "JSON"

    return client.SetDefaultTranscodeTemplateGroup(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySetDefaultTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteTranscodeTemplateGroup(client *vod.Client) (response *vod.DeleteTranscodeTempl
ateGroupResponse, err error) {
    request := vod.CreateDeleteTranscodeTemplateGroupRequest()
    request.TranscodeTemplateGroupId = "<TranscodeTemplateGroupId>"

    request.AcceptFormat = "JSON"

    return client.DeleteTranscodeTemplateGroup(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteTranscodeTemplateGroup(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

7.10. 截图模板

本篇文档提供了Go SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
```

```
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyAddVodTemplate(client *vod.Client) (response *vod.AddVodTemplateResponse, err error)
{
    request := vod.CreateAddVodTemplateRequest()

    // 模板名称
    request.Name = "Sample Snapshot Template"
    // 模板类型，固定值为Snapshot
    request.TemplateType = "Snapshot"

    // 截图模板配置数据
    snapshotConfig := map[string]interface{}{"Count": 50, "Interval": 1, "SpecifiedOffsetTi
me": 0, "Width": 200,
        "Height": 200, "FrameType": "normal"}
    templateConfig := map[string]interface{}{"SnapshotConfig": snapshotConfig, "SnapshotTyp
e": "NormalSnapshot"}

    /*
    // 可选项：雪碧图配置（雪碧图配置必须建立在普通截图配置之上）
    spriteSnapshotConfig := map[string]interface{}{"CellWidth": 120, "CellHeight": 68, "Col
umns": 3, "Lines": 10,
        "Padding": 20, "Margin": 50, "KeepCellPic": "keep", "Color": "tomato"}
    snapshotConfig["SpriteSnapshotConfig"] = spriteSnapshotConfig
    templateConfig["SnapshotType"] = "SpriteSnapshot"
    */

    jsonConfig, err := json.Marshal(templateConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TemplateConfig = string(jsonConfig)

    request.AcceptFormat = "JSON"

    return client.AddVodTemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddVodTemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
```

```
    fmt.Println(response.RequestId)
    fmt.Println(response.VodTemplateId)
}
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyUpdateVodTemplate(client *vod.Client) (response *vod.UpdateVodTemplateResponse, err
error) {
    request := vod.CreateUpdateVodTemplateRequest()

    request.VodTemplateId = "c0825c5f4c4a43af56abf9826b2d****"
    // 模板名称
    request.Name = "New Snapshot Template Name"

    // 修改截图模板配置
    snapshotConfig := map[string]interface{}{"Count": 50, "Interval": 1, "SpecifiedOffsetTi
me": 0, "Width": 200,
        "Height": 200, "FrameType": "normal"}
    templateConfig := map[string]interface{}{"SnapshotConfig": snapshotConfig, "SnapshotTyp
e": "NormalSnapshot"}

    jsonConfig, err := json.Marshal(templateConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TemplateConfig = string(jsonConfig)

    request.AcceptFormat = "JSON"

    return client.UpdateVodTemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateVodTemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyDeleteVodTemplate(client *vod.Client) (response *vod.DeleteVodTemplateResponse, err
error) {
    request := vod.CreateDeleteVodTemplateRequest()
    request.VodTemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.DeleteVodTemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteVodTemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyListVodTemplate(client *vod.Client) (response *vod.ListVodTemplateResponse, err erro
r) {
    request := vod.CreateListVodTemplateRequest()
    // 模板类型，固定值为Snapshot
    request.TemplateType = "Snapshot"

    request.AcceptFormat = "JSON"

    return client.ListVodTemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListVodTemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, template := range response.VodTemplateInfoList {
        fmt.Printf("%s: %s %s\n", template.VodTemplateId, template.Name, template.TemplateC
onfig)
    }
}
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyGetVodTemplate(client *vod.Client) (response *vod.GetVodTemplateResponse, err error)
{
    request := vod.CreateGetVodTemplateRequest()
    request.VodTemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.GetVodTemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetVodTemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Printf("%s: %s\n", response.VodTemplateInfo.Name, response.VodTemplateInfo.Template
Config)
}
```

7.11. 视频剪辑

本篇文档提供了Go SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

说明

通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func ProduceEditingVideoByTimeline(client *vod.Client) (response *vod.ProduceEditingProject
VideoResponse, err error) {
    request := vod.CreateProduceEditingProjectVideoRequest()

    // set timeline, this sample shows how to merge two videos
    videoTrackClips := []map[string]interface{}{}
    videoTrackClip1 := map[string]interface{}{"MediaId": "<VideoId1>"}
    videoTrackClip2 := map[string]interface{}{"MediaId": "<VideoId2>"}
    videoTrackClips = append(videoTrackClips, videoTrackClip1, videoTrackClip2)
    videoTrack := map[string]interface{}{"VideoTrackClips": videoTrackClips}
    videoTracks := []map[string]interface{}{}
    videoTracks = append(videoTracks, videoTrack)
    timeline := map[string]interface{}{"VideoTracks": videoTracks}
    jsonTimeline, err := json.Marshal(timeline)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.Timeline = string(jsonTimeline)

    // set media metadata
    mediaMetadata := map[string]interface{}{"Title": "editing sample title", "Description":
"editing sample description",
        "Tags": "Tag1,Tag2"}
    jsonMeta, err := json.Marshal(mediaMetadata)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.MediaMetadata = string(jsonMeta)

    // set produce config
    /*
    produceConfig := map[string]interface{}{"TemplateGroupId": "<templateGroupId>"}
    jsonConfig, err := json.Marshal(produceConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
```

```
    request.ProduceConfig = string(jsonConfig)
    */

    request.AcceptFormat = "JSON"

    return client.ProduceEditingProjectVideo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := ProduceEditingVideoByTimeline(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.MediaId)
}
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func ProduceEditingVideoByProject(client *vod.Client) (response *vod.ProduceEditingProjectV
ideoResponse, err error) {
    request := vod.CreateProduceEditingProjectVideoRequest()

    // set ProjectId
    request.ProjectId = "<ProjectId>"

    // set media metadata
    mediaMetadata := map[string]interface{}{"Title": "editing sample title", "Description":
"editing sample description",
        "Tags": "Tag1,Tag2"}
    jsonMeta, err := json.Marshal(mediaMetadata)
    if err != nil {
```

```
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.MediaMetadata = string(jsonMeta)

    // set produce config
    /*
    produceConfig := map[string]interface{}{"TemplateGroupId": "<templateGroupId>"}
    jsonConfig, err := json.Marshal(produceConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.ProduceConfig = string(jsonConfig)
    */

    request.AcceptFormat = "JSON"

    return client.ProduceEditingProjectVideo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := ProduceEditingVideoByProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.MediaId)
}
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyAddEditingProject(client *vod.Client) (response *vod.AddEditingProjectResponse, err
error) {
```

```
error) {
    request := vod.CreateAddEditingProjectRequest()

    // set timeline, this sample shows how to merge two videos
    videoTrackClips := []map[string]interface{}{}
    videoTrackClip1 := map[string]interface{}{"MediaId": "<VideoId1>"}
    videoTrackClip2 := map[string]interface{}{"MediaId": "<VideoId2>"}
    videoTrackClips = append(videoTrackClips, videoTrackClip1, videoTrackClip2)
    videoTrack := map[string]interface{}{"VideoTrackClips": videoTrackClips}
    videoTracks := []map[string]interface{}{}
    videoTracks = append(videoTracks, videoTrack)
    timeline := map[string]interface{}{"VideoTracks": videoTracks}
    jsonTimeline, err := json.Marshal(timeline)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.Timeline = string(jsonTimeline)

    // set project metadata
    request.Title = "editing project title"
    request.Description = "editing project description"

    request.AcceptFormat = "JSON"

    return client.AddEditingProject(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddEditingProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Println(response.Project.ProjectId)
}
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
```

```
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "encoding/json"
    "fmt"
)

func MyUpdateEditingProject(client *vod.Client) (response *vod.UpdateEditingProjectResponse
, err error) {
    request := vod.CreateUpdateEditingProjectRequest()

    // set projectId
    request.ProjectId = "<ProjectId>"

    // set timeline, this sample shows how to merge two videos
    videoTrackClips := []map[string]interface{}{}
    videoTrackClip1 := map[string]interface{}{"MediaId": "<VideoId1>"}
    videoTrackClip2 := map[string]interface{}{"MediaId": "<VideoId2>"}
    videoTrackClips = append(videoTrackClips, videoTrackClip1, videoTrackClip2)
    videoTrack := map[string]interface{}{"VideoTrackClips": videoTrackClips}
    videoTracks := []map[string]interface{}{}
    videoTracks = append(videoTracks, videoTrack)
    timeline := map[string]interface{}{"VideoTracks": videoTracks}
    jsonTimeline, err := json.Marshal(timeline)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.Timeline = string(jsonTimeline)

    // set project metadata
    request.Title = "new editing project title"
    request.Description = "new editing project description"

    request.AcceptFormat = "JSON"

    return client.UpdateEditingProject(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateEditingProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteEditingProject(client *vod.Client) (response *vod.DeleteEditingProjectResponse
, err error) {
    request := vod.CreateDeleteEditingProjectRequest()
    request.ProjectIds = "ProjectId1,ProjectId2"

    request.AcceptFormat = "JSON"

    return client.DeleteEditingProject(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteEditingProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetEditingProject(client *vod.Client) (response *vod.GetEditingProjectResponse, err
error) {
    request := vod.CreateGetEditingProjectRequest()
    request.ProjectId = "<ProjectId>"

    request.AcceptFormat = "JSON"

    return client.GetEditingProject(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetEditingProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    fmt.Printf("%s: %s\n", response.Project.Title, response.Project.Timeline)
}
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySearchEditingProject(client *vod.Client) (response *vod.SearchEditingProjectResponse
, err error) {
    request := vod.CreateSearchEditingProjectRequest()
    request.Title = "<Title Keywords>"
    request.StartTime = "2018-10-11T12:00:00Z"
    request.EndTime =  "2018-12-25T12:00:00Z"
    request.PageNo = "1"
    request.PageSize = "10"

    request.AcceptFormat = "JSON"

    return client.SearchEditingProject(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySearchEditingProject(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, project := range response.ProjectList.Project {
        fmt.Printf("%s: %s\n", project.ProjectId, project.Title)
    }
}
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySetEditingProjectMaterials(client *vod.Client) (response *vod.SetEditingProjectMater
ialsResponse, err error) {
    request := vod.CreateSetEditingProjectMaterialsRequest()
    request.ProjectId = "<ProjectId>"
    // 设置素材ID，多个用逗号分隔。素材ID即媒资ID，如视频VideoId、图片ImageId、辅助媒资MediaId等。
    request.MaterialIds = "VideoId,ImageId"

    request.AcceptFormat = "JSON"

    return client.SetEditingProjectMaterials(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySetEditingProjectMaterials(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetEditingProjectMaterials(client *vod.Client) (response *vod.GetEditingProjectMater
ialsResponse, err error) {
    request := vod.CreateGetEditingProjectMaterialsRequest()
    request.ProjectId = "<ProjectId>"
    request.Type = "video"

    request.AcceptFormat = "JSON"

    return client.GetEditingProjectMaterials(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetEditingProjectMaterials(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, material := range response.MaterialList.Material {
        fmt.Printf("%s: %s\n", material.MaterialId, material.Title)
    }
}
```

7.12. 媒体审核

本篇文档提供了Go SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySubmitAIMediaAuditJob(client *vod.Client) (response *vod.SubmitAIMediaAuditJobRespon
se, err error) {
    request := vod.CreateSubmitAIMediaAuditJobRequest()
    // 设置视频ID
    request.MediaId = "<VideoId>"
    //request.TemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.SubmitAIMediaAuditJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySubmitAIMediaAuditJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.JobId)
}
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetAIMediaAuditJob(client *vod.Client) (response *vod.GetAIMediaAuditJobResponse, er
r error) {
    request := vod.CreateGetAIMediaAuditJobRequest()
    request.JobId = "<JobId>"

    request.AcceptFormat = "JSON"

    return client.GetAIMediaAuditJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetAIMediaAuditJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.MediaAuditJob)
}
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetMediaAuditResult(client *vod.Client) (response *vod.GetMediaAuditResultResponse,
err error) {
    request := vod.CreateGetMediaAuditResultRequest()
    request.MediaId = "<VideoId>"

    request.AcceptFormat = "JSON"

    return client.GetMediaAuditResult(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetMediaAuditResult(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.MediaAuditResult.VideoResult)
}
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetMediaAuditResultDetail(client *vod.Client) (response *vod.GetMediaAuditResultDeta
ilResponse, err error) {
    request := vod.CreateGetMediaAuditResultDetailRequest()
    request.MediaId = "<VideoId>"
    request.PageNo = "1"

    request.AcceptFormat = "JSON"

    return client.GetMediaAuditResultDetail(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetMediaAuditResultDetail(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.MediaAuditResultDetail.Total)
    fmt.Println(response.MediaAuditResultDetail.List)
}
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetMediaAuditResultTimeline(client *vod.Client) (response *vod.GetMediaAuditResultTi
melineResponse, err error) {
    request := vod.CreateGetMediaAuditResultTimelineRequest()
    request.MediaId = "<VideoId>"

    request.AcceptFormat = "JSON"

    return client.GetMediaAuditResultTimeline(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetMediaAuditResultTimeline(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    for _, item := range response.MediaAuditResultTimeline.Porn {
        fmt.Printf("%s: %s %s\n", item.Timestamp, item.Label, item.Score)
    }
}
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyCreateAudit(client *vod.Client) (response *vod.CreateAuditResponse, err error) {
    request := vod.CreateCreateAuditRequest()

    auditContent := []map[string]interface{}{}
    auditItem1 := map[string]interface{}{"VideoId":"<VideoId1>", "Status":"Blocked", "Reaso
n": "porn video"}
    auditItem2 := map[string]interface{}{"VideoId":"<VideoId2>", "Status":"Normal", "Reason
": ""}
    auditContent = append(auditContent, auditItem1, auditItem2)
    jsonContent, err := json.Marshal(auditContent)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.AuditContent = string(jsonContent)
    request.AcceptFormat = "JSON"

    return client.CreateAudit(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyCreateAudit(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetAuditHistory(client *vod.Client) (response *vod.GetAuditHistoryResponse, err erro
r) {
    request := vod.CreateGetAuditHistoryRequest()
    request.VideoId = "<VideoId>"
    request.PageNo = "1"
    request.PageSize = "10"
    request.AcceptFormat = "JSON"

    return client.GetAuditHistory(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetAuditHistory(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, history := range response.Histories {
        fmt.Printf("%s: %s\n", history.CreationTime, history.Status)
    }
}
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySetAuditSecurityIp(client *vod.Client) (response *vod.SetAuditSecurityIpResponse, er
r error) {
    request := vod.CreateSetAuditSecurityIpRequest()

    // 审核安全组名称，默认为Default
    request.SecurityGroupName = "MyGroupName"
    // 操作方式，默认值Append(追加IP白名单)
    request.OperateMode = "Cover"
    // 审核安全IP列表，每个分组最多支持100个并以逗号分隔
    request. Ips = "10.23.12.20,10.23.12.21,10.23.12.22"

    request.AcceptFormat = "JSON"

    return client.SetAuditSecurityIp(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySetAuditSecurityIp(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListAuditSecurityIp(client *vod.Client) (response *vod.ListAuditSecurityIpResponse,
err error) {
    request := vod.CreateListAuditSecurityIpRequest()
    request.SecurityGroupName = "MyGroupName"

    request.AcceptFormat = "JSON"

    return client.ListAuditSecurityIp(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListAuditSecurityIp(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, ipGroup := range response.SecurityIpList {
        fmt.Printf("%s: %s\n", ipGroup.SecurityGroupName, ipGroup.Ips)
    }
}
```

7.13. 视频AI

本篇文档提供了Go SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySubmitAIJob(client *vod.Client) (response *vod.SubmitAIJobResponse, err error) {
    request := vod.CreateSubmitAIJobRequest()
    // 视频ID
    request.MediaId = "<VideoId>"
    // AI类型，多个用逗号分隔；请确保已开通该类型AI服务
    request.Types = "AIVideoCover,AIVideoCensor"

    request.AcceptFormat = "JSON"

    return client.SubmitAIJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySubmitAIJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.AIJobList.AIJob {
        fmt.Printf("%s: %s\n", job.Type, job.JobId)
    }
}
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListAIJob(client *vod.Client) (response *vod.ListAIJobResponse, err error) {
    request := vod.CreateListAIJobRequest()
    // 视频AI作业ID，多个用逗号分隔
    request.JobIds = "JobId1,JobId2"

    request.AcceptFormat = "JSON"

    return client.ListAIJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListAIJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.AIJobList.AIJob {
        fmt.Printf("%s: %s %s\n", job.JobId, job.Status, job.Data)
    }
}
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyAddAITemplate(client *vod.Client) (response *vod.AddAITemplateResponse, err error) {
    request := vod.CreateAddAITemplateRequest()

    // 设置模板类型，以智能审核模板为例
    request.TemplateType = "AIMediaAudit"
    // 设置模板名称
    request.TemplateName = "My AI Template"

    // 设置模板详细配置
    auditItem := []string{"terrorism", "porn"}
    auditRange := []string{"video", "image-cover", "text-title"}
    auditContent := []string{"screen"}
    templateConfig := map[string]interface{}{"AuditItem": auditItem, "AuditRange": auditRan
ge,
        "AuditContent": auditContent, "AuditAutoBlock": "no"}

    jsonConfig, err := json.Marshal(templateConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TemplateConfig = string(jsonConfig)

    request.AcceptFormat = "JSON"

    return client.AddAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyAddAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.TemplateId)
}
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDeleteAITemplate(client *vod.Client) (response *vod.DeleteAITemplateResponse, err er
ror) {
    request := vod.CreateDeleteAITemplateRequest()
    request.TemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.DeleteAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDeleteAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "encoding/json"
)

func MyUpdateAITemplate(client *vod.Client) (response *vod.UpdateAITemplateResponse, err er
ror) {
    request := vod.CreateUpdateAITemplateRequest()

    // 设置模板ID
    request.TemplateId = "2a0aadb19e187a58f296e2aeb2ad****"
    // 设置模板名称
    request.TemplateName = "New AI Template Name"

    // 设置模板详细配置
    auditItem := []string{"terrorism", "porn"}
    auditRange := []string{"video", "image-cover"}
    auditContent := []string{"screen"}
    templateConfig := map[string]interface{}{"AuditItem": auditItem, "AuditRange": auditRan
ge,
        "AuditContent": auditContent, "AuditAutoBlock": "yes"}

    jsonConfig, err := json.Marshal(templateConfig)
    if err != nil {
        fmt.Println("json.Marshal failed:", err)
        return
    }
    request.TemplateConfig = string(jsonConfig)

    request.AcceptFormat = "JSON"

    return client.UpdateAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyUpdateAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)


func MyGetAITemplate(client *vod.Client) (response *vod.GetAITemplateResponse, err error) {
    request := vod.CreateGetAITemplateRequest()
    request.TemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.GetAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.TemplateInfo)
}
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListAITemplate(client *vod.Client) (response *vod.ListAITemplateResponse, err error)
{
    request := vod.CreateListAITemplateRequest()
    request.TemplateType = "AIMediaAudit"

    request.AcceptFormat = "JSON"

    return client.ListAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.TemplateInfoList)
}
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySetDefaultAITemplate(client *vod.Client) (response *vod.SetDefaultAITemplateResponse
, err error) {
    request := vod.CreateSetDefaultAITemplateRequest()
    request.TemplateId = "<TemplateId>"

    request.AcceptFormat = "JSON"

    return client.SetDefaultAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySetDefaultAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetDefaultAITemplate(client *vod.Client) (response *vod.GetDefaultAITemplateResponse
, err error) {
    request := vod.CreateGetDefaultAITemplateRequest()
    request.TemplateType = "AIMediaAudit"

    request.AcceptFormat = "JSON"

    return client.GetDefaultAITemplate(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetDefaultAITemplate(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.TemplateInfo)
}
```

7.14. 视频DNA

本文为您提供了Go SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MySubmitAIJob(client *vod.Client) (response *vod.SubmitAIJobResponse, err error) {
    request := vod.CreateSubmitAIJobRequest()
    // 视频ID
    request.MediaId = "<VideoId>"
    // 设置AI类型，类型为固定值 AIMediaDNA
    request.Types = "AIMediaDNA"

    request.AcceptFormat = "JSON"

    return client.SubmitAIJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MySubmitAIJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.AIJobList.AIJob {
        fmt.Printf("%s: %s\n", job.Type, job.JobId)
    }
}
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListAIJob(client *vod.Client) (response *vod.ListAIJobResponse, err error) {
    request := vod.CreateListAIJobRequest()
    // 视频AI作业ID，多个用逗号分隔
    request.JobIds = "JobId1,JobId2"

    request.AcceptFormat = "JSON"

    return client.ListAIJob(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListAIJob(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    for _, job := range response.AIJobList.AIJob {
        fmt.Printf("%s: %s %s\n", job.JobId, job.Status, job.Data)
    }
}
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyGetMediaDNAResult(client *vod.Client) (response *vod.GetMediaDNAResultResponse, err
error) {
    request := vod.CreateGetMediaDNAResultRequest()
    // 视频ID
    request.MediaId = "<VideoId>"

    request.AcceptFormat = "JSON"

    return client.GetMediaDNAResult(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyGetMediaDNAResult(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
}
```

7.15. 直播转点播

本篇文档提供了Go SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
package main

import (
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/services/vod"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyListLiveRecordVideo(client *vod.Client) (response *vod.ListLiveRecordVideoResponse,
err error) {
    request := vod.CreateListLiveRecordVideoRequest()
    request.StartTime = "2018-12-01T06:00:00Z"
    request.EndTime = "2018-12-25T06:00:00Z"
    request.StreamName = "testStreamName"
    request.AppName = "testAppName"
    request.PageNo = "1"
    request.PageSize = "10"
    request.SortBy = "CreationTime:Desc"
    request.AcceptFormat = "JSON"

    return client.ListLiveRecordVideo(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyListLiveRecordVideo(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Println(response.RequestId)
    //videoList := response.LiveRecordVideoList
}
```

7.16. 点播CDN

本篇文档提供了Go SDK点播CDN模块相关功能的API调用示例。包含预热缓存、刷新缓存、查询刷新和预热状态、查询刷新预热次数限制和余量、查询流量数据、查询网络带宽、下载域名日志。

初始化客户端

使用前请先初始化客户端，请参见初始化。

预热缓存

调用PreloadVodObjectCaches接口，完成预热缓存功能。

接口参数和返回字段请参见PreloadVodObjectCaches。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "strings"
)

func MyPreloadVodObjectCaches(client *vod.Client) (response *vod.PreloadVodObjectCachesResp
onse, err error) {
    request := vod.CreatePreloadVodObjectCachesRequest()
    objectPath := []string{"http://example.com/fd.mp4", "http://example.com/ld.mp4"}
    request.ObjectPath = strings.Join(objectPath, "\n")

    request.AcceptFormat = "JSON"

    return client.PreloadVodObjectCaches(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyPreloadVodObjectCaches(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    fmt.Printf("PreloadTaskId: %s\n", response.PreloadTaskId)
}
```

刷新缓存

调用RefreshVodObjectCaches接口，完成刷新缓存功能。

接口参数和返回字段请参见RefreshVodObjectCaches。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
    "strings"
)

func MyRefreshVodObjectCaches(client *vod.Client) (response *vod.RefreshVodObjectCachesResp
onse, err error) {
    request := vod.CreateRefreshVodObjectCachesRequest()
    objectPath := []string{"http://example.com/fd.mp4", "http://example.com/ld.mp4"}
    request.ObjectPath = strings.Join(objectPath, "\n")

    request.AcceptFormat = "JSON"

    return client.RefreshVodObjectCaches(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyRefreshVodObjectCaches(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    fmt.Printf("RefreshTaskId: %s\n", response.RefreshTaskId)
}
```

查询刷新和预热状态

调用DescribeVodRefreshTasks接口，完成查询刷新和预热状态功能。

接口参数和返回字段请参见DescribeVodRefreshTasks。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDescribeVodRefreshTasks(client *vod.Client) (response *vod.DescribeVodRefreshTasksRe
sponse, err error) {
    request := vod.CreateDescribeVodRefreshTasksRequest()
    request.DomainName = "example.com"
    request.ObjectType = "preload"

    request.AcceptFormat = "JSON"

    return client.DescribeVodRefreshTasks(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDescribeVodRefreshTasks(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    for _, task := range response.Tasks.Task {
        fmt.Printf("%s: %s %s\n", task.ObjectPath, task.Status, task.Process)
    }
}
```

查询刷新预热次数限制和余量

调用DescribeVodRefreshQuota接口，完成查询刷新预热次数限制和余量功能。

接口参数和返回字段请参见DescribeVodRefreshQuota。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDescribeVodRefreshQuota(client *vod.Client) (response *vod.DescribeVodRefreshQuotaRe
sponse, err error) {
    request := vod.CreateDescribeVodRefreshQuotaRequest()

    request.AcceptFormat = "JSON"

    return client.DescribeVodRefreshQuota(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDescribeVodRefreshQuota(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    fmt.Printf("PreloadQuota %s, PreloadRemain %s\n", response.PreloadQuota, response.Prelo
adRemain)
    fmt.Printf("UrlQuota %s, UrlRemain %s\n", response.UrlQuota, response.UrlRemain)
}
```

查询流量数据

调用DescribeVodDomainTrafficData接口，完成查询流量数据功能。

接口参数和返回字段请参见DescribeVodDomainTrafficData。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDescribeVodDomainTrafficData(client *vod.Client) (response *vod.DescribeVodDomainTra
fficDataResponse, err error) {
    request := vod.CreateDescribeVodDomainTrafficDataRequest()
    request.DomainName = "example.com"
    request.StartTime = "2019-01-15T15:59:59Z"
    request.EndTime = "2019-01-20T15:59:58Z"

    request.AcceptFormat = "JSON"

    return client.DescribeVodDomainTrafficData(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDescribeVodDomainTrafficData(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    for _, item := range response.TrafficDataPerInterval.DataModule {
        fmt.Printf("%s: DomesticValue %s bytes, OverseasValue %s bytes\n",
            item.TimeStamp, item.DomesticValue, item.OverseasValue)
    }
}
```

查询网络带宽

调用DescribeVodDomainBpsData接口，完成查询网络带宽功能。

接口参数和返回字段请参见DescribeVodDomainBpsData。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDescribeVodDomainBpsData(client *vod.Client) (response *vod.DescribeVodDomainBpsData
Response, err error) {
    request := vod.CreateDescribeVodDomainBpsDataRequest()
    request.DomainName = "example.com"
    request.StartTime = "2019-01-15T15:59:59Z"
    request.EndTime = "2019-01-20T15:59:58Z"

    request.AcceptFormat = "JSON"

    return client.DescribeVodDomainBpsData(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDescribeVodDomainBpsData(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    for _, item := range response.BpsDataPerInterval.DataModule {
        fmt.Printf("%s: Value %s bps, OverseasValue %s bps\n",
            item.TimeStamp, item.Value, item.OverseasValue)
    }
}
```

下载域名日志

调用DescribeVodDomainLog接口，完成下载域名日志功能。

接口参数和返回字段请参见DescribeVodDomainLog。调用示例如下：

```
package main

import (
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/services/vod"
    "192.168.0.0/16/aliyun/alibaba-cloud-sdk-go/sdk/auth/credentials"
    "fmt"
)

func MyDescribeVodDomainLog(client *vod.Client) (response *vod.DescribeVodDomainLogResponse
, err error) {
    request := vod.CreateDescribeVodDomainLogRequest()
    request.DomainName = "example.cn-shanghai.192.168.0.0/16"
    request.StartTime = "2019-01-15T15:59:59Z"
    request.EndTime = "2019-01-20T15:59:58Z"
    request.PageNumber = "1"
    request.PageSize = "300"

    request.AcceptFormat = "JSON"

    return client.DescribeVodDomainLog(request)
}

func main() {
    client, err := InitVodClient("<accessKeyId>", "<accessKeySecret>")
    if err != nil {
        panic(err)
    }

    response, err := MyDescribeVodDomainLog(client)
    if err != nil {
        panic(err)
    }

    fmt.Println(response.GetHttpContentString())
    fmt.Printf("RequestId: %s\n", response.RequestId)
    logDetail := response.DomainLogDetails.DomainLogDetail[0]
    fmt.Printf("TotalLogCount: %d\n", logDetail.PageInfos.Total)
    for _, item := range logDetail.LogInfos.LogInfoDetail {
        fmt.Printf("%s: LogSize %d bytes, LogDownloadPath\n  %s\n",
            item.StartTime, item.LogSize, item.LogPath)
    }
}
```

8.C/C++ SDK

8.1. 安装

本文为您介绍Linux环境下安装视频点播服务端C/C++ SDK的操作步骤。

前提条件

您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。

环境准备

> **注意** 视频点播服务端C/C++ SDK仅支持Linux环境。

需要安装的软件包及其用途、安装示例等见下表，支持yum安装和网上下载安装。

软件包	作用	yum安装示例	网上下载安装示例

cmake	第三方编译安装工具	• 版本：建议使用2.6.0及以上版本。
		• 安装示例：
		`yum install cmake`
			• 下载地址：下载地址
			• 安装示例：
			`./configure`
			`make`
			`make install`

libcurl	解决网络连接等问题	• 版本：建议使用7.29.0 及以上版本。
		• 安装示例：
		`yum install libcurl-devel`
			• 下载地址：下载地址
			• 安装示例：
			`./configure`
			`make`
			`make install`

libuuid	生成uuid	安装示例： `yum install libuuid-devel`	不涉及

apr	不涉及	安装示例：
		`yum install apr-devel`
			• 下载地址：下载地址
			• 安装示例：
			`./configure`
			`make`
			`make install`

软件包	作用	yum安装示例	网上下载安装示例

apr-util	解决内存管理以及跨平台问题	安装示例：
		`yum install apr-util`
			• 下载地址：下载地址
			• 安装示例：
			`// 安装时需要指定--with-apr选项。`
			`./configure --with-apr=/your/apr/install/path`
			`make`
			`make install`

minixml	解析请求返回的xml	安装示例：
		`yum install mxml mxml-devel`
			• 下载地址：下载地址
			• 安装示例：
			`./configure`
			`make`
			`make install`

jsoncpp	解析请求返回的JOSN	安装示例：
		`yum install jsoncpp-devel`
			• 下载地址：下载地址
			• 安装示例：
			`./configure`
			`make`
			`make install`

OSS上传SDK	由于上传SDK依赖OSS，需要下载并安装OSS上传SDK	不涉及	安装步骤，请参见OSS-SDK安装。 **注意** 有部分OSS依赖库已经在前面步骤进行安装，已经安装的不需要再次进行重复安装。

SDK安装

1. 下载SDK。

   SDK下载地址，请参见C/C++上传SDK。

   > **说明** C/C++服务端SDK与C/C++上传SDK为统一版本。

2. 将SDK进行解压。
3. 进入目录，编译并安装。具体示例如下：

```
cmake .
make
make install
```

8.2. 初始化

C/C++ SDK可通过AccessKey或STS Security Token初始化。您可以根据需求选择初始化方式。本文介绍两种初始化的操作流程。

背景信息

视频点播服务端SDK提供两种初始化方式。AccessKey方式根据授权策略获取对应权限，启用后一直有效，建议在服务端使用。STS Security Token方式根据授权策略获取相应权限后，可以自定义过期时间。

前提条件

- 您已经安装了视频点播服务端C/C++ SDK 。安装步骤，请参见安装。
- 您已经知晓服务接入地域，例如：接入地域在上海，请使用 `cn-shanghai` 。点播服务的接入地域请参见接入地域和标识。

AccessKey初始化

调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

填入AccessKey信息，进行初始化。示例如下：

```
#include "vod_sdk/openApiUtil.h"
VodCredential initVodClient(std::string accessKeyId, std::string accessKeySecret) {
    VodCredential authInfo;
    //填入AccessKey信息
    authInfo.accessKeyId = accessKeyId;
    authInfo.accessKeySecret = accessKeySecret;
    authInfo.regionId = "cn-shanghai";
    return authInfo;
}
```

STS初始化

使用STS方式进行初始化，需要提前准备好STS Token。获取方法，请参见STS访问授权。

填入STS信息，进行初始化。示例如下：

```
#include "vod_sdk/openApiUtil.h"
VodCredential initVodClient(std::string accessKeyId, std::string accessKeySecret, std::String securityToken) {
    VodCredential authInfo;
    //填入STS信息
    authInfo.accessKeyId = accessKeyId;
    authInfo.accessKeySecret = accessKeySecret;
    authInfo.securityToken = securityToken;
    authInfo.regionId = "cn-shanghai";
    return authInfo;
}
```

常见问题

- 动态库链接需要保证库安装目录在编译以及运行目录中，如果不在可如何操作？

  可参考如下操作。示例如下：

  ```
  //将目录添加至/etc/ld.so.conf文件中
  执行ldconfig
  ```

  编译时需要添加如下参数：
  - lcurl
  - ljsoncpp
  - lvod_sdk
  - loss_c_sdk
  - lapr-1
  - laprutil-1
  - lmxml

- `getAcsResponse` 函数可传入哪些参数？

  `getAcsResponse` 函数为请求基础函数，支持的传入参数如下：
  - vodCredential：请参见上方AccessKey初始化示例。
  - apiName：请求接口名称。
  - args：参数列表。

- `VodApiResponse` 为哪个函数返回值的类型？

  `VodApiResponse` 为 `getAcsResponse` 函数的返回值类型。
  - 当请求成功时，`VodApiResponse` 中httpCode的值为HTTP应答码，result默认为JSON格式返回（返回类型等公共参数可在请求参数args中设置返回类型进行指定）。
  - 当请求失败时，`VodApiResponse` 中httpCode的值为-1，此时result为错误信息。

8.3. 媒体上传

点播服务端提供丰富的上传接口，支持开发者采用不同的上传方式实现媒体上传。本文介绍C/C++ SDK调用服务端上传接口的使用场景和调用示例。

使用场景

> **注意**
>
> 本文仅提供API调用示例，即在点播服务中获取上传凭证和地址等操作示例。要实现完整的媒体文件上传逻辑，请参考下表跳转至对应的上传方式和使用场景。

接口	使用场景
获取音视频上传地址和凭证	• OSS原生SDK上传场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证需要Base64解码后作为初始化OSS上传客户端实例的参数。详情请参见基于OSS原生SDK上传。 • 点播客户端上传SDK场景下，用户应用服务器集成点播服务端SDK并调用相关接口获取或刷新上传地址和凭证。获取到的上传地址和凭证无需解码即可下发给客户端。详情请参见客户端上传SDK指引。
刷新音视频上传凭证	
获取图片上传地址和凭证	
获取辅助媒资上传地址和凭证	
URL批量拉取上传	• 调用URL批量拉取上传接口实现离线拉取上传。这种方式可以免去先将文件下载到本地服务器或终端再使用上传SDK上传到点播的过程。 • 调用获取URL上传信息获取上传进度。
获取URL上传信息	

前提条件

- 调用接口前请先安装SDK。操作指引请参见安装。
- 调用接口前请先初始化客户端实例。操作指引请参见初始化。

> **说明**
>
> 以下示例均通过AccessKey初始化客户端实例。

接口调用示例

获取音/视频上传地址和凭证

调用CreateUploadVideo接口，完成获取音/视频上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadVideo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取音/视频上传地址和凭证
*/
VodApiResponse createUploadVideo(VodCredential authInfo) {
    string apiName = "CreateUploadVideo";
    map<string, string> args;
    args["Title"] = "sample title";
    args["FileName"] = "filename.mp4";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = createUploadVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

刷新音/视频上传凭证

调用RefreshUploadVideo接口，完成刷新音/视频上传凭证功能。

接口参数和返回字段请参见RefreshUploadVideo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 刷新音/视频上传凭证
*/
VodApiResponse refreshUploadVideo(VodCredential authInfo) {
    string apiName = "RefreshUploadVideo";
    map<string, string> args;
    args["VideoId"] = "<VideoId>"; // 替换成需要的视频ID
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = refreshUploadVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取图片上传地址和凭证

调用CreateUploadImage接口，完成获取图片上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadImage。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取图片上传地址和凭证
*/

VodApiResponse createUploadImage(VodCredential authInfo) {
    string apiName = "CreateUploadImage";
    map<string, string> args;
    args["ImageType"] = "cover";
    args["ImageExt"] = "jpg";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = createUploadImage(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取辅助媒资上传地址和凭证

调用CreateUploadAttachedMedia接口，完成获取辅助媒资上传地址和凭证功能。

接口参数和返回字段请参见CreateUploadAttachedMedia。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取辅助媒资(水印、字幕等)上传地址和凭证
 */

VodApiResponse createUploadAttachedMedia(VodCredential authInfo) {
    string apiName = "CreateUploadAttachedMedia";
    map<string, string> args;
    args["BusinessType"] = "watermark";
    args["MediaExt"] = "gif";
    args["Title"] = "this is a sample title";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = createUploadAttachedMedia(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

URL批量拉取上传

调用UploadMediaByURL接口，完成URL批量拉取上传功能。

接口参数和返回字段请参见UploadMediaByURL。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * URL批量拉取上传
*/

VodApiResponse uploadMediaByURL(VodCredential authInfo) {
    string apiName = "UploadMediaByURL";
    map<string, string> args;
    args["UploadURLs"] = "http://192.168.0.0/16.mp4";
    Json::Value uploadMetadataList;
    Json::Value uploadMetadata;
    uploadMetadata["SourceUrl"] = "http://192.168.0.0/16.mp4";
    uploadMetadata["Title"] = "upload by url sample";
    uploadMetadataList.append(uploadMetadata);
    args["UploadMetadatas"] = uploadMetadataList.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = uploadMediaByURL(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.4. 音视频播放

本篇文档提供了C/C++ SDK音视频播放模块相关功能的API调用示例。包含获取播放地址、获取视频播放凭证。

初始化客户端

使用前请先初始化客户端，请参见初始化。

获取视频播放地址

调用GetPlayInfo接口，完成获取播放地址功能。

接口参数和返回字段请参见GetPlayInfo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*获取播放地址函数*/

VodApiResponse getPlayInfo(VodCredential authInfo) {
    string apiName = "GetPlayInfo";
    map<string, string> args;
    args["VideoId"] = "<VideoId>";  // 替换成需要的视频ID
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getPlayInfo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取视频播放凭证

调用GetVideoPlayAuth接口，完成获取视频播放凭证功能。

接口参数和返回字段请参见GetVideoPlayAuth。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*获取播放凭证函数*/

VodApiResponse getVideoPlayAuth(VodCredential authInfo) {
    string apiName = "GetVideoPlayAuth";
    map<string, string> args;
    args["VideoId"] = "<VideoId>";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getVideoPlayAuth(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.5. 媒资管理

本篇文档提供了C/C++ SDK媒资管理模块相关功能的API调用示例。主要包含搜索媒资信息、获取视频信息、修改视频信息、删除视频、获取源文件信息、获取图片信息、删除图片信息等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

搜索媒资信息

调用SearchMedia接口，完成搜索媒资信息功能。

接口参数和返回字段请参见SearchMedia。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 搜索媒资信息
*/
VodApiResponse searchMedia(VodCredential authInfo) {
    string apiName = "SearchMedia";
    map<string, string> args;
    args["Fields"] = "Title,CoverURL,Status";
    args["Match"] = "Status in ('Normal','Checking') and CreationTime = ('2018-07-01T08:00:
00Z','2018-08-01T08:00:00Z')";
    args["PageNo"] = "1";
    args["PageSize"] = "10";
    args["SearchType"] = "video";
    args["SortBy"] = "CreationTime:Desc";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = searchMedia(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取视频信息

调用GetVideoInfo接口，完成获取视频信息功能。

接口参数和返回字段请参见GetVideoInfo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取视频信息
*/
VodApiResponse getVideoInfo(VodCredential authInfo) {
    string apiName = "GetVideoInfo";
    map<string, string> args;
    args["VideoId"] = "<VideoId>";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getVideoInfo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

批量获取视频信息

调用GetVideoInfos接口，完成批量获取视频信息功能。

接口参数和返回字段请参见GetVideoInfos。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 批量获取视频信息函数
*/
VodApiResponse getVideoInfos(VodCredential authInfo) {
    string apiName = "GetVideoInfos";
    map<string, string> args;
    args["VideoIds"] = "VideoId1,VideoId2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getVideoInfos(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改视频信息

调用UpdateVideoInfo接口，完成修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 修改视频信息
*/
VodApiResponse updateVideoInfo(VodCredential authInfo) {
    string apiName = "UpdateVideoInfo";
    map<string, string> args;
    args["VideoId"] = "VideoId";
    args["Title"] = "new Title";
    args["Description"] = "new Description";
    args["Tags"] = "new Tag1,new Tag2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateVideoInfo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

批量修改视频信息

调用UpdateVideoInfos接口，完成批量修改视频信息功能。

接口参数和返回字段请参见UpdateVideoInfos。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 批量修改视频信息
*/
VodApiResponse updateVideoInfos(VodCredential authInfo) {
    string apiName = "UpdateVideoInfos";
    map<string, string> args;
    Json::Value updateContentArray;
    Json::Value updateContent1;
    updateContent1["VideoId"] = "VideoId1";
    // updateContent1["Title"] = "new Title";
    // updateContent1["Tags"] = "new Tag1,new Tag2";
    updateContentArray.append(updateContent1);
    Json::Value updateContent2;
    updateContent2["VideoId"] = "VideoId2";
    // updateContent2["Title"] = "new Title";
    // updateContent2["Tags"] = "new Tag1,new Tag2";
    updateContentArray.append(updateContent2);
    args["UpdateContent"] = updateContentArray.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateVideoInfos(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除视频

调用DeleteVideo接口，完成删除视频功能。

接口参数和返回字段请参见DeleteVideo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 删除视频
*/
VodApiResponse deleteVideo(VodCredential authInfo) {
    string apiName = "DeleteVideo";
    map<string, string> args;
    //多个用逗号分隔
    args["VideoIds"] = "VideoId1,VideoId2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取源文件信息（含源片下载地址）

调用GetMezzanineInfo接口，完成获取源文件信息功能。

接口参数和返回字段请参见GetMezzanineInfo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取源文件信息
*/
VodApiResponse getMezzanineInfo(VodCredential authInfo) {
    string apiName = "GetMezzanineInfo";
    map<string, string> args;
    args["VideoId"] = "<VideoId>";
    //源片下载地址过期时间，单位s
    args["AuthTimeout"] = "3600";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getMezzanineInfo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取视频列表

调用GetVideoList接口，完成获取视频列表功能。

接口参数和返回字段请参见GetVideoList。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取视频列表
*/
VodApiResponse getVideoList(VodCredential authInfo) {
    string apiName = "GetVideoList";
    map<string, string> args;
    args["PageNo"] = "1";
    args["PageSize"] = "20";
    args["StartTime"] = "2018-12-27T09:00:38Z";
    args["EndTime"] = "2018-12-28T09:00:38Z";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getVideoList(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除媒体流

调用DeleteStream接口，完成删除媒体流功能。

接口参数和返回字段请参见DeleteStream。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 删除媒体流函数
*/
VodApiResponse deleteStream(VodCredential authInfo) {
    string apiName = "DeleteStream";
    map<string, string> args;
    args["VideoId"] = "<VideoId>";
    args["JobIds"] = "JobId1,JobId2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteStream(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

批量删除源文件

调用DeleteMezzanines接口，完成批量删除源文件功能。

接口参数和返回字段请参见DeleteMezzanines。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 批量删除源文件函数
*/
VodApiResponse deleteMezzanines(VodCredential authInfo) {
    string apiName = "DeleteMezzanines";
    map<string, string> args;
    args["VideoIds"] = "VideoId1,VideoId2";
    args["Force"] = "false";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteMezzanines(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

批量更新图片信息

调用UpdateImageInfos接口，完成批量更新图片信息功能。

接口参数和返回字段请参见UpdateImageInfos。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 批量更新图片信息函数
*/
VodApiResponse updateImageInfos(VodCredential authInfo) {
    string apiName = "UpdateImageInfos";
    map<string, string> args;
    Json::Value updateContentArray;
    Json::Value updateContent1;
    updateContent1["ImageId"] = "ImageId1";
    // updateContent1["Title"] = "new Title";
    // updateContent1["Tags"] = "new Tag1,new Tag2";
    updateContentArray.append(updateContent1);
    Json::Value updateContent2;
    updateContent2["ImageId"] = "ImageId2";
    // updateContent2["Title"] = "new Title";
    // updateContent2["Tags"] = "new Tag1,new Tag2";
    updateContentArray.append(updateContent2);
    args["UpdateContent"] = updateContentArray.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateImageInfos(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取图片信息

调用GetImageInfo接口，完成获取图片信息功能。

接口参数和返回字段请参见GetImageInfo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取图片信息函数
*/
VodApiResponse getImageInfo(VodCredential authInfo) {
    string apiName = "GetImageInfo";
    map<string, string> args;
    args["ImageId"] = "<ImageId>";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getImageInfo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除图片

调用DeleteImage接口，完成删除图片功能。

接口参数和返回字段请参见DeleteImage。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 删除图片函数
*/
VodApiResponse deleteImage(VodCredential authInfo) {
    string apiName = "DeleteImage";
    map<string, string> args;
    //根据ImageURL删除图片文件
    args["DeleteImageType"] = "ImageURL";
    //Image URL示例：http://example.192.168.0.0/16/cover.jpg
    args["ImageURLs"] = "<your Image URL>";
    //根据ImageId删除图片文件
    args["DeleteImageType"] = "ImageId";
    args["ImageIds"] = "ImageId1,ImageId2";
    //根据VideoId删除指定ImageType的图片文件
    args["DeleteImageType"] = "VideoId";
    args["VideoId"] = "<VideoId>";
    args["ImageType"] = "SpriteSnapshot";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteImage(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.6. 媒资分类

本文为您提供了C/C++ SDK媒资分类模块相关功能的API调用示例，包含创建分类、删除分类、修改分类、查寻分类及其子分类。

初始化客户端

使用前请先初始化客户端，请参见初始化。

创建分类

调用AddCategory接口，完成创建分类功能。

接口参数和返回字段请参见AddCategory。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*创建分类函数*/
VodApiResponse addCategory(VodCredential authInfo) {
    string apiName = "AddCategory";
    map<string, string> args;
    // 父分类ID，若不填，则默认生成一级分类，根节点分类ID为-1
    args["ParentId"] = "-1";
    // 分类名称，不能超过64个字节，UTF8编码
    args["CateName"] = "CateName";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addCategory(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改分类

调用UpdateCategory接口，完成修改分类功能。

接口参数和返回字段请参见UpdateCategory。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*修改分类函数*/
VodApiResponse updateCategory(VodCredential authInfo) {
    string apiName = "UpdateCategory";
    map<string, string> args;
    args["CateId"] = "<CateId>";
    // 分类名称，不能超过64个字节，UTF8编码
    args["CateName"] = "CateName";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateCategory(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除分类

调用DeleteCategory接口，完成删除分类功能。

接口参数和返回字段请参见DeleteCategory。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*删除分类函数*/
VodApiResponse deleteCategory(VodCredential authInfo) {
    string apiName = "DeleteCategory";
    map<string, string> args;
    args["CateId"] = "<CateId>";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteCategory(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询分类及其子分类

调用GetCategories接口，完成查询分类及其子分类功能。

接口参数和返回字段请参见GetCategories。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/*查询分类及其子分类函数*/
VodApiResponse getCategories(VodCredential authInfo) {
    string apiName = "GetCategories";
    map<string, string> args;
    args["CateId"] = "<CateId>";
    args["PageNo"] = "1";
    args["PageSize"] = "10";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getCategories(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.7. 媒体处理

本文提供了C/C++ SDK媒体处理模块相关功能的API调用示例，包含提交转码作业、提交截图作业、查询截图数据、导播台视频预处理。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交转码作业

调用SubmitTranscodeJobs接口，完成提交转码作业-普通功能。

接口参数和返回字段请参见SubmitTranscodeJobs。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 1、构建覆盖参数，目前只支持图片水印文件地址、文字水印的内容覆盖；
 * 2、需要替换的水印信息对应水印ID必须是关联在指定的模板ID(即TranscodeTemplateId)中；
 * 3、不支持通过媒体处理接口去增加一个没有关联上的水印
 * 注意：图片水印的文件存储源站需要和发起转码的视频存储源站一致
 */
Json::Value buildOverrideParams() {
    Json::Value overrideParams;
    Json::Value watermarks;
    Json::Value watermark1;
    watermark1["WatermarkId"] = "2ea587477c5a1bc8b57****";
    //需要替换成对应图片水印文件的OSS地址，水印文件存储源站需要和视频存储源站一致
    //FileURL示例：https://example-bucket-****.oss-cn-shanghai.192.168.0.0/16/watermarks/02A
1B22DF25D46C3C725A4****.png
    watermark1["FileUrl"] = "<your File URL>";
    watermarks.append(watermark1);
    //文字水印内容替换
    Json::Value watermark2;
    //模板上面关联需要替换内容的文字水印ID
    watermark2["WatermarkId"] = "d297ba31ac5242d207****";
    //需要替换成对应的内容
    watermark2["Content"] = "用户ID：6****";
    watermarks.append(watermark2);
    overrideParams["Watermarks"] = watermarks;
    return overrideParams;
 }

 /**
 * 生成加密需要的密钥，response中包含密文密钥和明文密钥，用户只需要将密文密钥传递给点播即可
 * 注意：KeySpec 必须传递AES_128，且不能设置NumberOfBytes
 * @param serviceKey 点播提供生成密钥的service key，在用户的密钥管理服务中可看到描述为vod的加密key
 */
string generateDataKey(string serviceKey) {
    string ciphertextBlob;
```

```
    /*此处需实现KMS获取密文密钥，详情参考KMS产品-GenerateDataKey接口*/
    return ciphertextBlob;
}

/**
 * 构建HLS标准加密的配置信息
 */
Json::Value buildEncryptConfig() {
    //点播给用户在KMS(密钥管理服务)中的Service Key，可在用户密钥管理服务对应的区域看到描述为vod的ser
vice key
    string serviceKey = "<Your Service Key>";
    //随机生成一个加密的密钥，返回的response包含明文密钥以及密文密钥，
    //视频标准加密只需要传递密文密钥即可
    string ciphertextBlob = generateDataKey(serviceKey);
    Json::Value encryptConfig;
    //解密接口地址，该参数需要将每次生成的密文密钥与接口URL拼接生成，表示每个视频的解密的密文密钥都不一
样
    //至于Ciphertext这个解密接口参数的名称，用户可自行制定，这里只作为参考参数名称
    encryptConfig["DecryptKeyUri"] = "http://192.168.0.0/16/decrypt?Ciphertext=" + cipherte
xtBlob;
    //密钥服务的类型，目前只支持KMS
    encryptConfig["KeyServiceType"] = "KMS";
    //密文密钥
    encryptConfig["CipherText"] = ciphertextBlob;
    return encryptConfig;
}

/**
 * 提交媒体处理作业
 */
VodApiResponse submitTranscodeJobs(VodCredential authInfo) {
    string apiName = "SubmitTranscodeJobs";
    map<string, string> args;
    //需要转码的视频ID
    args["VideoId"] = "34a6ca54f5c140eece85a289****";
    //转码模板ID
    args["TemplateGroupId"] = "e8aa925a9798c630d30cd7****";
    //构建需要替换的水印参数(只有需要替换水印相关信息才需要构建)
    //Json::Value overrideParams = buildOverrideParams();
    //覆盖参数，暂只支持水印部分参数替换(只有需要替换水印相关信息才需要传递)
    //args["OverrideParams"] = overrideParams.toStyledString();

    //构建标准加密配置参数(只有标准加密才需要构建)
    //Json::Value encryptConfig = buildEncryptConfig();
    //HLS标准加密配置(只有标准加密才需要传递)
    //args["EncryptConfig"] = encryptConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

/**
 * 以下为调用示例
 */
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
```

```
    VodApiResponse response = submitTranscodeJobs(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

提交截图作业

调用SubmitSnapshotJob接口，完成提交截图作业功能。

接口参数和返回字段请参见SubmitSnapshotJob。调用示例如下：

> **说明**
>
> 创建截图模板详细请参见截图模板。

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建雪碧图截图配置
 * @return
 */
Json::Value buildSnapshotTemplateConfig() {
    Json::Value spriteSnapshotConfig;
    spriteSnapshotConfig["CellWidth"] = "120";
    spriteSnapshotConfig["CellHeight"] = "68";
    spriteSnapshotConfig["Columns"] = "3";
    spriteSnapshotConfig["Lines"] = "10";
    spriteSnapshotConfig["Padding"] = "20";
    spriteSnapshotConfig["Margin"] = "50";
    //保留雪碧图原始图
    spriteSnapshotConfig["KeepCellPic"] = "keep";
    spriteSnapshotConfig["Color"] = "tomato";
    return spriteSnapshotConfig;
}

/**
 * 提交媒体截图处理作业调用函数
 */
 VodApiResponse submitSnapshotJob(VodCredential authInfo) {
    string apiName = "SubmitSnapshotJob";
    map<string, string> args;
    //需要截图的视频ID(推荐传递截图模板ID)
    args["VideoId"] = "4d237a8270084849bf4207876181****";
    //截图模板ID
    args["SnapshotTemplateId"] = "5d745e6b8baadf589e0702426cfc6****";
    //如果设置了SnapshotTemplateId，会忽略下面参数
    args["Count"] = "50";
    args["SpecifiedOffsetTime"] = "0";
    args["Interval"] = "1";
    args["Width"] = "200";
    args["Height"] = "200";
    //雪碧图配置(如果需要)
    Json::Value spriteSnapshotConfig = buildSnapshotTemplateConfig();
    args["SpriteSnapshotConfig"] = spriteSnapshotConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = submitSnapshotJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询截图数据

调用ListSnapshots接口，完成查询截图数据功能。

接口参数和返回字段请参见ListSnapshots。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询截图数据
 */
 VodApiResponse listSnapshots(VodCredential authInfo) {
    string apiName = "ListSnapshots";
    map<string, string> args;
    //视频ID
    args["VideoId"] = "c86c0ceba9796535****";
    args["SnapshotType"] = "CoverSnapshot";
    args["PageNo"] = "1";
    args["PageSize"] = "20";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listSnapshots(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

导播台视频预处理

调用SubmitPreprocessJobs接口，完成导播台视频预处理功能。

接口参数和返回字段请参见SubmitPreprocessJobs。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"


/**
 * 导播台视频预处理
 */
 VodApiResponse submitPreprocessJobs(VodCredential authInfo) {
    string apiName = "SubmitPreprocessJobs";
    map<string, string> args;
    //视频ID
    args["VideoId"] = "c86c0ceba97965352418";
    args["PreprocessType"] = "PreprocessType";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = submitPreprocessJobs(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.8. 视频水印

本篇文档提供了C/C++ SDK视频水印模块相关功能的API调用示例。包含添加水印、修改水印、删除水印、查询水印、设置默认水印等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加水印

调用AddWatermark接口，完成添加水印功能。

接口参数和返回字段请参见AddWatermark。调用示例如下：

> **说明**
>
> - 获取文件上传地址和凭证详细参数请参见CreateUploadAttachedMedia。
> - 水印文件OSS上传详细参数请参见OSS文件上传。

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
```

```
 */
Json::Value buildImageWatermarkConfig() {
    Json::Value watermarkConfig;
    //水印的横向偏移距离
    watermarkConfig["Dx"] = "8";
    //水印的纵向偏移距离
    watermarkConfig["Dy"] = "8";
    //水印显示的宽
    watermarkConfig["Width"] = "55";
    //水印显示的高
    watermarkConfig["Height"] = "55";
    //水印显示的相对位置(左上、右上、左下、右下)
    watermarkConfig["ReferPos"] = "BottomRight";
    //水印显示的时间线(开始显示和结束显示时间)
    Json::Value timeline;
    //水印开始显示时间
    timeline["Start"] = "2";
    //水印结束显示时间
    timeline["Duration"] = "ToEND";
    watermarkConfig["Timeline"] = timeline;
    return watermarkConfig;
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 */
Json::Value buildTextWatermarkConfig() {
    Json::Value watermarkConfig;
    //文字水印显示的内容
    watermarkConfig["Content"] = "testwatermark";
    //文字水印的字体名称
    watermarkConfig["FontName"] = "SimSun";
    //文字水印的字体大小
    watermarkConfig["FontSize"] = "25";
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    watermarkConfig["FontColor"] = "Black";
    //文字水印的透明度
    watermarkConfig["FontAlpha"] = "0.2";
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    watermarkConfig["BorderColor"] = "White";
    //文字水印的描边宽度
    watermarkConfig["BorderWidth"] = "1";
    //文字水印距离视频画面上边的偏移距离
    watermarkConfig["Top"] = "20";
    //文字水印距离视频画面左边的偏移距离
    watermarkConfig["Left"] = "15";
    return watermarkConfig;
}

/**
 * 添加水印配置信息函数
 */
VodApiResponse addWatermark(VodCredential authInfo) {
    string apiName = "AddWatermark";
```

```
    map<string, string> args;
    //水印名称
    args["Name"] = "testaddwatermark";
    //获取水印文件在oss的URL
    //File URL示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com/watermark/test**
**.png
    string fileUrl = "<your File URL>";
    //图片水印必传图片文件的oss文件地址，水印文件必须和视频在同一个区域，例如:华东2视频，水印文件必须存
放在华东2
    args["FileUrl"] = fileUrl;
    //水印配置数据
    Json::Value watermarkConfig;
    //图片水印的位置配置数据
    watermarkConfig = buildImageWatermarkConfig();
    //文字水印的位置配置数据
    //watermarkConfig = buildTextWatermarkConfig();
    args["WatermarkConfig"] = watermarkConfig.toStyledString();
    //文字水印:Text; 图片水印:Image
    args["Type"] = "Image";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改水印

调用UpdateWatermark接口，完成修改水印功能。

接口参数和返回字段请参见UpdateWatermark。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建图片水印的配置数据，根据具体设置需求修改对应的参数值
 */
Json::Value buildImageWatermarkConfig() {
    Json::Value watermarkConfig;
    //水印的横向偏移距离
    watermarkConfig["Dx"] = "8";
    //水印的纵向偏移距离
    watermarkConfig["Dy"] = "8";
    //水印显示的宽
    watermarkConfig["Width"] = "55";
    //水印显示的高
    watermarkConfig["Height"] = "55";
```

```
    //水印显示的相对位置(左上、右上、左下、右下)
    watermarkConfig["ReferPos"] = "BottomRight";
    //水印显示的时间线(开始显示和结束显示时间)
    Json::Value timeline;
    //水印开始显示时间
    timeline["Start"] = "2";
    //水印结束显示时间
    timeline["Duration"] = "ToEND";
    watermarkConfig["Timeline"] = timeline;
    return watermarkConfig;
}

/**
 * 构建文字水印的配置数据，根据具体设置需求修改对应的参数值
 */
Json::Value buildTextWatermarkConfig() {
    Json::Value watermarkConfig;
    //文字水印显示的内容
    watermarkConfig["Content"] = "testwatermark";
    //文字水印的字体名称
    watermarkConfig["FontName"] = "SimSun";
    //文字水印的字体大小
    watermarkConfig["FontSize"] = "25";
    //文字水印的颜色(也可为RGB颜色取值，例如:#000000)
    watermarkConfig["FontColor"] = "Black";
    //文字水印的透明度
    watermarkConfig["FontAlpha"] = "0.2";
    //文字水印的字体描边颜色(也可为RGB颜色取值，例如:#ffffff)
    watermarkConfig["BorderColor"] = "White";
    //文字水印的描边宽度
    watermarkConfig["BorderWidth"] = "1";
    //文字水印距离视频画面上边的偏移距离
    watermarkConfig["Top"] = "20";
    //文字水印距离视频画面左边的偏移距离
    watermarkConfig["Left"] = "15";
    return watermarkConfig;
}

/**
 * 修改水印配置信息函数
 * 注意：不支持图片文件地址的修改，如果更换请创建新的水印信息
 */
VodApiResponse updateWatermark(VodCredential authInfo) {
    string apiName = "UpdateWatermark";
    map<string, string> args;
    //需要更新配置信息的水印ID
    args["WatermarkId"] = "421ddddd4f6e734a526fd2e****";
    //水印配置数据
    Json::Value watermarkConfig;
    //图片水印的位置配置数据
    watermarkConfig = buildImageWatermarkConfig();
    //文字水印的位置配置数据
    //watermarkConfig = buildTextWatermarkConfig();
    args["WatermarkConfig"] = watermarkConfig.toStyledString();
```

```
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除水印

调用DeleteWatermark接口，完成删除水印功能。

接口参数和返回字段请参见DeleteWatermark。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 *删除水印配置信息函数
 */
VodApiResponse deleteWatermark(VodCredential authInfo) {
    string apiName = "DeleteWatermark";
    map<string, string> args;
    args["WatermarkId"] = "53f9d796fad9d7b862b2e5****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询水印列表

调用ListWatermark接口，完成查询水印列表功能。

接口参数和返回字段请参见ListWatermark。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询水印配置信息列表函数
 */
VodApiResponse listWatermark(VodCredential authInfo) {
    string apiName = "ListWatermark";
    map<string, string> args;
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询单个水印

调用GetWatermark接口，完成查询单个水印功能。

接口参数和返回字段请参见GetWatermark。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询单个水印配置信息函数
 */
VodApiResponse getWatermark(VodCredential authInfo) {
    string apiName = "GetWatermark";
    map<string, string> args;
    args["WatermarkId"] = "96d4f6e734a526fd2****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

设置默认水印

调用SetDefaultWatermark接口，完成设置默认水印功能。

接口参数和返回字段请参见SetDefaultWatermark。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 设置默认水印配置信息函数
 */
VodApiResponse setDefaultWatermark(VodCredential authInfo) {
    string apiName = "SetDefaultWatermark";
    map<string, string> args;
    args["WatermarkId"] = "96d4f6e734a526fd2****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = setDefaultWatermark(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.9. 转码模板

本篇文档提供了C/C++ SDK转码模板模块相关功能的API调用示例。包含添加转码模板组、修改转码模板组、删除转码模板组、查询转码模板组、设置默认转码模板组等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加转码模板组

调用AddTranscodeTemplateGroup接口，完成添加转码模板组功能。

接口参数和返回字段请参见AddTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建需要添加的转码模板配置数据
 */
Json::Value buildTranscodeTemplateList() {
    Json::Value transcodeTemplateList;
    Json::Value transcodeTemplate;
    //视频流转码配置
```

```
    Json::Value video;
    video["Width"] = 640;
    video["Bitrate"] = 400;
    video["Fps"] = 25;
    video["Remove"] = false;
    video["Codec"] = "H.264";
    video["Gop"] = "250";
    transcodeTemplate["Video"] = video;

    //音频流转码配置
    Json::Value audio;
    audio["Codec"] = "AAC";
    audio["Bitrate"] = "64";
    audio["Channels"] = "2";
    audio["Samplerate"] = "32000";
    transcodeTemplate["Audio"] = audio;

    //封装容器
    Json::Value container;
    container["Format"] = "mp4";
    transcodeTemplate["Container"] = container;

    //条件转码配置
    Json::Value transconfig;
    transconfig["IsCheckReso"] = false;
    transconfig["IsCheckResoFail"] = false;
    transconfig["IsCheckVideoBitrate"] = false;
    transconfig["IsCheckVideoBitrateFail"] = false;
    transconfig["IsCheckAudioBitrate"] = false;
    transconfig["IsCheckAudioBitrateFail"] = false;
    transcodeTemplate["TransConfig"] = transconfig;

    //加密配置(只支持m3u8)
    //Json::Value encryptSetting;
    //encryptSetting["EncryptType"] = "Private";
    //transcodeTemplate["EncryptSetting"] = encryptSetting;

    //清晰度
    transcodeTemplate["Definition"] = "LD";

    //模板名称
    transcodeTemplate["TemplateName"] = "testtemplate";

    //水印ID(多水印关联)
    Json::Value watermarkIdList;
    watermarkIdList.append("263261bdc1ff65782f8995c6dd22****");
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    watermarkIdList.append("USER_DEFAULT_WATERMARK");
    transcodeTemplate["WatermarkIds"] = watermarkIdList;
    transcodeTemplateList.append(transcodeTemplate);
    return transcodeTemplateList;
}

/**
 * 添加转码模板组配置
```

```
 * 添加转码模板组配置
 */
VodApiResponse addTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "AddTranscodeTemplateGroup";
    map<string, string> args;
    args["Name"] = "grouptest";
    args["TranscodeTemplateList"] = buildTranscodeTemplateList().toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改转码模板组

调用UpdateTranscodeTemplateGroup接口，完成修改转码模板组功能。

接口参数和返回字段请参见UpdateTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建需要添加的转码模板配置数据
 */
Json::Value buildTranscodeTemplateList() {
    Json::Value transcodeTemplateList;
    Json::Value transcodeTemplate;
    //视频流转码配置
    Json::Value video;
    video["Width"] = 640;
    video["Bitrate"] = 400;
    video["Fps"] = 25;
    video["Remove"] = false;
    video["Codec"] = "H.264";
    video["Gop"] = "250";
    transcodeTemplate["Video"] = video;

    //音频流转码配置
    Json::Value audio;
    audio["Codec"] = "AAC";
    audio["Bitrate"] = "64";
    audio["Channels"] = "2";
    audio["Samplerate"] = "32000";
    transcodeTemplate["Audio"] = audio;

    //封装容器
    Json::Value container;
```

```
    Json::Value container;
    container["Format"] = "mp4";
    transcodeTemplate["Container"] = container;

    //条件转码配置
    Json::Value transconfig;
    transconfig["IsCheckReso"] = false;
    transconfig["IsCheckResoFail"] = false;
    transconfig["IsCheckVideoBitrate"] = false;
    transconfig["IsCheckVideoBitrateFail"] = false;
    transconfig["IsCheckAudioBitrate"] = false;
    transconfig["IsCheckAudioBitrateFail"] = false;
    transcodeTemplate["TransConfig"] = transconfig;

    //加密配置(只支持m3u8)
    //Json::Value encryptSetting;
    //encryptSetting["EncryptType"] = "Private";
    //transcodeTemplate["EncryptSetting"] = encryptSetting;

    //模板名称
    transcodeTemplate["TemplateName"] = "testtemplate";

    //需要修改的转码模板ID
    transcodeTemplate["TranscodeTemplateId"] = "85c2b18ac08fda33e8f6d9c56****";

    //水印ID(多水印关联)
    Json::Value watermarkIdList;
    watermarkIdList.append("263261bdc1ff65782f8995c6dd22****");
    //USER_DEFAULT_WATERMARK 代表默认水印ID
    watermarkIdList.append("USER_DEFAULT_WATERMARK");
    transcodeTemplate["WatermarkIds"] = watermarkIdList;
    transcodeTemplateList.append(transcodeTemplate);
    return transcodeTemplateList;
}

/**
 * 修改转码模板组配置
 */
VodApiResponse updateTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "UpdateTranscodeTemplateGroup";
    map<string, string> args;
    //转码模板组ID
    args["TranscodeTemplateGroupId"] = "4c71a339fecec0152b4fa6f452****";
    args["Name"] = "grouptest";
    args["TranscodeTemplateList"] = buildTranscodeTemplateList().toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询转码模板组列表

调用ListTranscodeTemplateGroup接口，完成查询转码模板组列表功能。

接口参数和返回字段请参见ListTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询转码模板组列表
 */
VodApiResponse listTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "ListTranscodeTemplateGroup";
    map<string, string> args;
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询单个转码模板组

调用GetTranscodeTemplateGroup接口，完成查询单个转码模板组功能。

接口参数和返回字段请参见GetTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询单个转码模板组配置
 */
VodApiResponse getTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "GetTranscodeTemplateGroup";
    map<string, string> args;
    args["TranscodeTemplateGroupId"] = "a0fa0fda545e50e7a3eb75491****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

设置默认转码模板组

调用SetDefaultTranscodeTemplateGroup接口，完成设置默认转码模板组功能。

接口参数和返回字段请参见SetDefaultTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 设置默认转码模板组
 */
VodApiResponse setDefaultTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "SetDefaultTranscodeTemplateGroup";
    map<string, string> args;
    args["TranscodeTemplateGroupId"] = "a0fa0fda545e50e7a3eb75491****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = setDefaultTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除转码模板组

调用DeleteTranscodeTemplateGroup接口，完成删除转码模板组功能。

接口参数和返回字段请参见DeleteTranscodeTemplateGroup。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"


/**
 * 删除转码模板组配置
 */
VodApiResponse deleteTranscodeTemplateGroup(VodCredential authInfo) {
    string apiName = "DeleteTranscodeTemplateGroup";
    map<string, string> args;
    args["TranscodeTemplateGroupId"] = "a0fa0fda545e50e7a3eb75491****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteTranscodeTemplateGroup(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.10. 截图模板

本篇文档提供了C/C++ SDK截图模板功能的API调用示例。包含添加截图模板、修改截图模板、删除截图模板、查询截图模板列表。

初始化客户端

使用前请先初始化客户端，请参见初始化。

添加截图模板

调用AddVodTemplate接口，完成添加截图模板功能。

接口参数和返回字段请参见AddVodTemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
 * （以下代码示例为雪碧图完整配置）
 * @return
 */
Json::Value buildSnapshotTemplateConfig() {
```

```
    Json::Value templateConfig;
    Json::Value snapshotConfig;
    snapshotConfig["Count"] = "50";
    snapshotConfig["Interval"] = "1";
    snapshotConfig["SpecifiedOffsetTime"] = "0";
    snapshotConfig["Width"] = "200";
    snapshotConfig["Height"] = "200";
    snapshotConfig["FrameType"] = "normal";

    //雪碧图配置(雪碧图配置必须是建立在普通截图配置之上)
    Json::Value spriteSnapshotConfig;
    spriteSnapshotConfig["CellWidth"] = "120";
    spriteSnapshotConfig["CellHeight"] = "68";
    spriteSnapshotConfig["Columns"] = "3";
    spriteSnapshotConfig["Lines"] = "10";
    spriteSnapshotConfig["Padding"] = "20";
    spriteSnapshotConfig["Margin"] = "50";
    spriteSnapshotConfig["KeepCellPic"] = "keep";
    spriteSnapshotConfig["Color"] = "tomato";
    snapshotConfig["SpriteSnapshotConfig"] = spriteSnapshotConfig;

    //普通截图配置(与雪碧图原始图配置共用该配置信息)
    templateConfig["SnapshotConfig"] = snapshotConfig;

    //截图类型(存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot")
    templateConfig["SnapshotType"] = "SpriteSnapshot";
    return templateConfig;
}

/**
 * 添加截图模板函数
 */
VodApiResponse addSnapshotVodTemplate(VodCredential authInfo) {
    string apiName = "AddVodTemplate";
    map<string, string> args;
    //模板名称
    args["Name"] = "Addtest";
    //模板类型，固定值为Snapshot
    args["TemplateType"] = "Snapshot";
    //截图模板配置数据生成
    Json::Value templateConfig = buildSnapshotTemplateConfig();
    args["TemplateConfig"] = templateConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addSnapshotVodTemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改截图模板

调用UpdateVodTemplate接口，完成修改截图模板功能。

接口参数和返回字段请参见UpdateVodTemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建截图模板的配置数据，根据具体设置需求修改对应的参数值
 * （以下代码示例为雪碧图完整配置）
 * @return
 */
Json::Value buildSnapshotTemplateConfig() {
    Json::Value templateConfig;
    Json::Value snapshotConfig;
    snapshotConfig["Count"] = "50";
    snapshotConfig["Interval"] = "1";
    snapshotConfig["SpecifiedOffsetTime"] = "0";
    snapshotConfig["Width"] = "200";
    snapshotConfig["Height"] = "200";
    snapshotConfig["FrameType"] = "normal";

    //雪碧图配置（雪碧图配置必须是建立在普通截图配置之上）
    Json::Value spriteSnapshotConfig;
    spriteSnapshotConfig["CellWidth"] = "120";
    spriteSnapshotConfig["CellHeight"] = "68";
    spriteSnapshotConfig["Columns"] = "3";
    spriteSnapshotConfig["Lines"] = "10";
    spriteSnapshotConfig["Padding"] = "20";
    spriteSnapshotConfig["Margin"] = "50";
    spriteSnapshotConfig["KeepCellPic"] = "keep";
    spriteSnapshotConfig["Color"] = "tomato";
    snapshotConfig["SpriteSnapshotConfig"] = spriteSnapshotConfig;

    //普通截图配置（与雪碧图原始图配置共用该配置信息）
    templateConfig["SnapshotConfig"] = snapshotConfig;

    //截图类型（存在雪碧图配置，该类型必须是"SpriteSnapshot"，否则为"NormalSnapshot"）
    templateConfig["SnapshotType"] = "SpriteSnapshot";
    return templateConfig;
}

/**
 * 修改截图模板函数
 */
VodApiResponse updateSnapshotVodTemplate(VodCredential authInfo) {
    string apiName = "UpdateVodTemplate";
    map<string, string> args;
    //设置要修改的模板ID
    args["VodTemplateId"] = "53azf9d796fad9d7b862b2e****";
    //模板名称
    args["Name"] = "updatetest";
```

```
    //截图模板配置数据生成
    Json::Value templateConfig = buildSnapshotTemplateConfig();
    args["TemplateConfig"] = templateConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateSnapshotVodTemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除截图模板

调用DeleteVodTemplate接口，完成删除截图模板功能。

接口参数和返回字段请参见DeleteVodTemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"


/**
 * 删除截图模板函数
 */
VodApiResponse deleteSnapshotVodTemplate(VodCredential authInfo) {
    string apiName = "DeleteVodTemplate";
    map<string, string> args;
    //设置要删除的模板ID
    args["VodTemplateId"] = "53azf9d796fad9d7b862b2e****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteSnapshotVodTemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询截图模板列表

调用ListVodTemplate接口，完成查询截图模板列表功能。

接口参数和返回字段请参见ListVodTemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询截图模板列表函数
 */
VodApiResponse listSnapshotVodTemplate(VodCredential authInfo) {
    string apiName = "ListVodTemplate";
    map<string, string> args;
    //模板类型，固定值为Snapshot
    args["TemplateType"] = "Snapshot";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listSnapshotVodTemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询截图模板

调用GetVodTemplate接口，完成查询截图模板功能。

接口参数和返回字段请参见GetVodTemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
* 查询截图模板函数
*/
VodApiResponse getSnapshotVodTemplate(VodCredential authInfo) {
    string apiName = "GetVodTemplate";
    map<string, string> args;
    //设置要查询的模板ID
    args["VodTemplateId"] = "53azf9d796fad9d7b862b2e****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getSnapshotVodTemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.11. 视频剪辑

本篇文档提供了C/C++ SDK视频剪辑模块相关功能的API调用示例。包含发起剪辑合成、创建云剪辑工程、修改云剪辑工程、删除云剪辑工程、获取单个云剪辑工程、设置云剪辑工程素材等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

通过时间线-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过时间线-发起剪辑合成功能。

绝大多数场景下，使用这种方式即可合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

> **说明**
>
> 通过时间线实现剪辑合成的更多Timeline示例请参见视频剪辑-使用示例。

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

Json::Value buildMediaMetadata(){
    Json::Value mediaMetadata;
    // Produce Media Title
    mediaMetadata["Title"] = "TestTitle";
    // Produce Media Description
    mediaMetadata["Description"] = "TestDescription";
    // Produce Media UserDefined Cover URL
    mediaMetadata["CoverURL"] = "http://192.168.0.0/16/media/cover/mediaid.jpg";
    // Produce Media Category ID
    mediaMetadata["CateId"] = "<your cateid>";
    // Produce Media Category Name
    mediaMetadata["Tags"] = "Tag1,Tag2";
    return mediaMetadata;
}

Json::Value buildProduceConfig(){
    Json::Value produceConfig;
    /*
    The produce process can generate media mezzanine file. You can use the mezzanine file t
o transcode other media files, just like the transcode process after file upload finished.
This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
    1. Not required
    2. Use default transcode template group id when empty
    */
    produceConfig["TemplateGroupId"] = "<your TemplateGroupId>";
    return produceConfig;
}
```

```
/**
* This Sample shows how to merge two videos
*/
Json::Value buildTimeline(){
    Json::Value timeline;

    // Video Track
    Json::Value videoTracks;
    Json::Value videoTrack;
    // Video Track Clips
    Json::Value videoTrackClips;
    Json::Value videoTrackClip1;
    videoTrackClip1["MediaId"] = "11119b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip1);
    Json::Value videoTrackClip2;
    videoTrackClip2["MediaId"] = "22229b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip2);

    videoTrack["VideoTrackClips"] = videoTrackClips;
    videoTracks.append(videoTrack);

    timeline["VideoTracks"] = videoTracks;

    return timeline;
}

VodApiResponse produceEditingProjectVideo(VodCredential authInfo) {
    string apiName = "ProduceEditingProjectVideo";
    map<string, string> args;
    args["Timeline"] = buildTimeline().toStyledString();
    args["MediaMetadata"] = buildMediaMetadata().toStyledString();
    args["ProduceConfig"] = buildProduceConfig().toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = produceEditingProjectVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

通过云剪辑工程-发起剪辑合成

调用ProduceEditingProjectVideo接口，完成通过云剪辑工程-发起剪辑合成功能。

对云剪辑工程有较高管理需求的场景下，可以使用这种方式合成视频。接口参数和返回字段请参见ProduceEditingProjectVideo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

Json::Value buildMediaMetadata(){
    Json::Value mediaMetadata;
    // Produce Media Title
    mediaMetadata["Title"] = "TestTitle";
    // Produce Media Description
    mediaMetadata["Description"] = "TestDescription";
    // Produce Media UserDefined Cover URL
    mediaMetadata["CoverURL"] = "http://192.168.0.0/16/media/cover/mediaid.jpg";
    // Produce Media Category ID
    mediaMetadata["CateId"] = "<your cateid>";
    // Produce Media Category Name
    mediaMetadata["Tags"] = "Tag1,Tag2";
    return mediaMetadata;
}

Json::Value buildProduceConfig(){
    Json::Value produceConfig;
    /*
    The produce process can generate media mezzanine file. You can use the mezzanine file t
o transcode other media files, just like the transcode process after file upload finished.
This field describe the Transocde TemplateGroup ID after produce mezzanine finished.
    1. Not required
    2. Use default transcode template group id when empty
    */
    produceConfig["TemplateGroupId"] = "<your TemplateGroupId>";
    return produceConfig;
}

VodApiResponse produceEditingProjectVideo(VodCredential authInfo) {
    string apiName = "ProduceEditingProjectVideo";
    map<string, string> args;
    // Set Editing Project ID need to produce
    args["ProjectId"] = "<your ProjectId>";
    // Set Produce Media Metadata
    args["MediaMetadata"] = buildMediaMetadata().toStyledString();
    // Set Produce Configuration
    args["ProduceConfig"] = buildProduceConfig().toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = produceEditingProjectVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

创建云剪辑工程

调用AddEditingProject接口，完成创建云剪辑工程功能。

接口参数和返回字段请参见AddEditingProject。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
* This Sample shows how to merge two videos
*/
Json::Value buildTimeline(){
    Json::Value timeline;

    // Video Track
    Json::Value videoTracks;
    Json::Value videoTrack;
    // Video Track Clips
    Json::Value videoTrackClips;
    Json::Value videoTrackClip1;
    videoTrackClip1["MediaId"] = "11119b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip1);
    Json::Value videoTrackClip2;
    videoTrackClip2["MediaId"] = "22229b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip2);

    videoTrack["VideoTrackClips"] = videoTrackClips;
    videoTracks.append(videoTrack);

    timeline["VideoTracks"] = videoTracks;

    return timeline;
}

VodApiResponse addEditingProject(VodCredential authInfo) {
    string apiName = "AddEditingProject";
    map<string, string> args;
    // Build Editing Project Timeline
    args["Timeline"] = buildTimeline().toStyledString();
    // Set Editing Project Title
    args["Title"] = "Editing Project Title";
    // Set Editing Project Cover URL
    args["CoverURL"] = "http://192.168.0.0/16/editingproject/cover/projectid.jpg";
    // Set Editing Project Description
    args["Description"] = "Editing Project Description";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addEditingProject(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改云剪辑工程

调用UpdateEditingProject接口，完成修改云剪辑工程功能。

接口参数和返回字段请参见UpdateEditingProject。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
* This Sample shows how to merge two videos
*/
Json::Value buildTimeline(){
    Json::Value timeline;

    // Video Track
    Json::Value videoTracks;
    Json::Value videoTrack;
    // Video Track Clips
    Json::Value videoTrackClips;
    Json::Value videoTrackClip1;
    videoTrackClip1["MediaId"] = "11119b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip1);
    Json::Value videoTrackClip2;
    videoTrackClip2["MediaId"] = "22229b4d7cf14dc7b83b0e801cbe****";
    videoTrackClips.append(videoTrackClip2);

    videoTrack["VideoTrackClips"] = videoTrackClips;
    videoTracks.append(videoTrack);

    timeline["VideoTracks"] = videoTracks;

    return timeline;
}

VodApiResponse updateEditingProject(VodCredential authInfo) {
    string apiName = "UpdateEditingProject";
    map<string, string> args;
    args["ProjectId"] = "YourProjectId";
    // Build Editing Project Timeline
    args["Timeline"] = buildTimeline().toStyledString();
    // Set Editing Project Title
    args["Title"] = "Editing Project Title";
    // Set Editing Project Cover URL
    args["CoverURL"] = "http://192.168.0.0/16/editingproject/cover/projectid.jpg";
    // Set Editing Project Description
    args["Description"] = "Editing Project Description";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateEditingProject(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除云剪辑工程

调用DeleteEditingProject接口，完成删除云剪辑工程功能。

接口参数和返回字段请参见DeleteEditingProject。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

VodApiResponse deleteEditingProject(VodCredential authInfo) {
    string apiName = "DeleteEditingProject";
    map<string, string> args;
    args["ProjectIds"] = "projectid1,projectid2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteEditingProject(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取单个云剪辑工程

调用GetEditingProject接口，完成获取单个云剪辑工程功能。

接口参数和返回字段请参见GetEditingProject。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

VodApiResponse getEditingProject(VodCredential authInfo) {
    string apiName = "GetEditingProject";
    map<string, string> args;
    args["ProjectId"] = "your projectId";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getEditingProject(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

搜索云剪辑工程

调用SearchEditingProject接口，完成搜索云剪辑工程功能。

接口参数和返回字段请参见SearchEditingProject。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

VodApiResponse searchEditingProject(VodCredential authInfo) {
    string apiName = "SearchEditingProject";
    map<string, string> args;
    args["Title"] = "Title Keywords";
    args["StartTime"] = "2017-01-11T12:00:00Z";
    args["EndTime"] = "2017-01-12T12:00:00Z";
    args["PageSize"] = "10";
    args["PageNo"] = "1";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = searchEditingProject(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

设置云剪辑工程素材

调用SetEditingProjectMaterials接口，完成设置云剪辑工程素材功能。

接口参数和返回字段请参见SetEditingProjectMaterials。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

VodApiResponse setEditingProjectMaterials(VodCredential authInfo) {
    string apiName = "SetEditingProjectMaterials";
    map<string, string> args;
    args["ProjectId"] = "<Your ProjectId>";
    args["MaterialIds"] = "materialId1,materialId2";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = setEditingProjectMaterials(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取云剪辑工程素材

调用GetEditingProjectMaterials接口，完成获取云剪辑工程素材功能。

接口参数和返回字段请参见GetEditingProjectMaterials。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

VodApiResponse getEditingProjectMaterials(VodCredential authInfo) {
    string apiName = "GetEditingProjectMaterials";
    map<string, string> args;
    args["ProjectId"] = "<Your ProjectId>";
    args["Type"] = "video";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getEditingProjectMaterials(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.12. 媒体审核

本篇文档提供了C/C++ SDK媒体审核模块相关功能的API调用示例。包含提交智能审核作业、获取智能审核结果、查询智能审核作业、人工审核、设置审核安全IP列表等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交智能审核作业

调用SubmitAIMediaAuditJob接口，完成提交智能审核作业功能。

接口参数和返回字段请参见SubmitAIMediaAuditJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 提交智能审核作业
 */
VodApiResponse submitAIMediaAuditJob(VodCredential authInfo) {
    string apiName = "SubmitAIMediaAuditJob";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "dc063078c1d845139e2a5bd8ff54****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = submitAIMediaAuditJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询智能审核作业

调用GetAIMediaAuditJob接口，完成查询智能审核作业功能。

接口参数和返回字段请参见GetAIMediaAuditJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询智能审核作业
 */
VodApiResponse getAIMediaAuditJob(VodCredential authInfo) {
    string apiName = "GetAIMediaAuditJob";
    map<string, string> args;
    // 设置作业ID
    args["JobId"] = "7dc69b893c8b4f13b47aae8de085****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getAIMediaAuditJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取智能审核结果摘要

调用GetMediaAuditResult接口，完成获取智能审核结果摘要功能。

接口参数和返回字段请参见GetMediaAuditResult。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取智能审核结果摘要
 */
VodApiResponse getMediaAuditResult(VodCredential authInfo) {
    string apiName = "GetMediaAuditResult";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "dc063078c1d845139e2a5bd8f58****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getMediaAuditResult(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取智能审核结果详情

调用GetMediaAuditResultDetail接口，完成获取智能审核结果详情功能。

接口参数和返回字段请参见GetMediaAuditResultDetail。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取智能审核结果详情
 */
VodApiResponse getMediaAuditResultDetail(VodCredential authInfo) {
    string apiName = "GetMediaAuditResultDetail";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "dc063078c1d845139e2a5bd8f23****";
    // 设置翻页页数
    args["PageNo"] = "1";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getMediaAuditResultDetail(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取智能审核结果时间线

调用GetMediaAuditResultTimeline接口，完成获取智能审核结果时间线功能。

接口参数和返回字段请参见GetMediaAuditResultTimeline。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取智能审核结果时间线
 */
VodApiResponse getMediaAuditResultTimeline(VodCredential authInfo) {
    string apiName = "GetMediaAuditResultTimeline";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "dc063078c1d845139e2a5bd8f85****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getMediaAuditResultTimeline(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

人工审核

调用CreateAudit接口，完成人工审核功能。

接口参数和返回字段请参见CreateAudit。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 构建审核内容
 */
Json::Value buildAuditContent() {
        Json::Value auditContents;
        Json::Value auditContent;
        auditContent["VideoId"] = "3ebc10160bda481ca9b6858a0b32****"; // 视频ID
        auditContent["Status"] = "Blocked"; // 审核状态
        auditContent["Reason"] = "含有淫秽色情内容"; // 若审核状态为屏蔽时，需给出屏蔽的理由，最长支持128字节
        auditContents.append(auditContent);
        return auditContents;
}

/**
 * 人工审核
 */
VodApiResponse createAudit(VodCredential authInfo) {
    string apiName = "CreateAudit";
    map<string, string> args;
    // 设置审核内容
    args["AuditContent"] = buildAuditContent().toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = createAudit(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取人工审核历史

调用GetAuditHistory接口，完成获取人工审核历史功能。

接口参数和返回字段请参见GetAuditHistory。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取人工审核历史
 */
VodApiResponse getAuditHistory(VodCredential authInfo) {
    string apiName = "GetAuditHistory";
    map<string, string> args;
    // 视频ID
    args["VideoId"] = "3ebc10160bda481ca9b6858a0b32****";
    // 页号
    args["PageNo"] = "1";
    // 每页数量
    args["PageSize"] = "10";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getAuditHistory(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

设置审核安全IP

调用SetAuditSecurityIp接口，完成设置审核安全IP功能。

接口参数和返回字段请参见SetAuditSecurityIp。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 设置审核安全IP
 */
VodApiResponse setAuditSecurityIp(VodCredential authInfo) {
    string apiName = "SetAuditSecurityIp";
    map<string, string> args;
    // 审核安全组名称，默认为Default
    args["SecurityGroupName"] = "MyGroupName";
    // 操作方式，默认值Append(追加IP白名单)
    args["OperateMode"] = "Cover";
    // 审核安全IP列表
    args["Ips"] = "10.23.12.20,10.23.12.21,10.23.12.22";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = setAuditSecurityIp(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取审核安全IP列表

调用ListAuditSecurityIp接口，完成获取审核安全IP列表功能。

接口参数和返回字段请参见ListAuditSecurityIp。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 获取审核安全IP
 */
VodApiResponse listAuditSecurityIp(VodCredential authInfo) {
    string apiName = "ListAuditSecurityIp";
    map<string, string> args;
    // 审核安全组名称
    args["SecurityGroupName"] = "MyGroupName";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listAuditSecurityIp(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.13. 视频AI

本篇文档提供了C/C++ SDK视频AI模块相关功能的API调用示例。包含提交AI作业、查询AI作业、添加AI模板、修改AI模板、删除AI模板、查询AI模板、查询设置默认AI模板等。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交AI作业

调用SubmitAIJob接口，完成提交AI作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 提交作业
 */
VodApiResponse submitAIJob(VodCredential authInfo) {
    string apiName = "SubmitAIJob";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "3eb19a4585bc475e995bddfd****";
    // 设置AI类型，请确保已开通该类型AI
    args["Types"] = "AIVideoCover";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = submitAIJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询AI作业

调用ListAIJob接口，完成查询AI作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询作业
 */
VodApiResponse listAIJob(VodCredential authInfo) {
    string apiName = "ListAIJob";
    map<string, string> args;
    // 设置作业ID
    args["JobIds"] = "979d4d7a36ae41b1a945a252****,3eb19a4585bc475e995bddea5****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listAIJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

添加AI模板

调用AddAITemplate接口，完成添加AI模板功能。

接口参数和返回字段请参见AddAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 添加AI模板
 */
VodApiResponse addAITemplate(VodCredential authInfo) {
    string apiName = "AddAITemplate";
    map<string, string> args;
    // 设置模板类型，以智能审核模板为例
    args["TemplateType"] = "AIMediaAudit";
    // 设置模板名称
    args["TemplateName"] = "我的自定义模板";
    // 设置模板详细配置
    Json::Value templateConfig;
    Json::Value auditItem;
    auditItem.append("terrorism");
    auditItem.append("porn");
    templateConfig["AuditItem"] = auditItem;

    Json::Value auditRange;
    auditRange.append("video");
    auditRange.append("image-cover");
    auditRange.append("text-title");
    templateConfig["AuditRange"] = auditRange;

    Json::Value auditContent;
    auditContent.append("screen");
    templateConfig["AuditContent"] = auditContent;
    templateConfig["AuditAutoBlock"] = "no";
    args["TemplateConfig"] = templateConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = addAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

删除AI模板

调用DeleteAITemplate接口，完成删除AI模板功能。

接口参数和返回字段请参见DeleteAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 删除AI模板
 */
VodApiResponse deleteAITemplate(VodCredential authInfo) {
    string apiName = "DeleteAITemplate";
    map<string, string> args;
    // 设置模板ID
    args["TemplateId"] = "1d763dd8987a122ab8596eb7d2c8****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = deleteAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

修改AI模板

调用UpdateAITemplate接口，完成修改AI模板功能。

接口参数和返回字段请参见UpdateAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 修改AI模板
 */
VodApiResponse updateAITemplate(VodCredential authInfo) {
    string apiName = "UpdateAITemplate";
    map<string, string> args;
    // 设置模板ID
    args["TemplateId"] = "1d763dd8987a122ab8596eb7d258****";
    // 设置模板名称
    args["TemplateName"] = "我的自定义模板";
    // 设置模板详细配置
    Json::Value templateConfig;
    Json::Value auditItem;
    auditItem.append("terrorism");
    auditItem.append("porn");
    templateConfig["AuditItem"] = auditItem;

    Json::Value auditRange;
    auditRange.append("video");
    auditRange.append("image-cover");
    auditRange.append("text-title");
    templateConfig["AuditRange"] = auditRange;

    Json::Value auditContent;
    auditContent.append("screen");
    templateConfig["AuditContent"] = auditContent;
    templateConfig["AuditAutoBlock"] = "no";
    args["TemplateConfig"] = templateConfig.toStyledString();
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = updateAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询AI模板

调用GetAITemplate接口，完成查询AI模板功能。

接口参数和返回字段请参见GetAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询AI模板
 */
VodApiResponse getAITemplate(VodCredential authInfo) {
    string apiName = "GetAITemplate";
    map<string, string> args;
    // 设置模板ID
    args["TemplateId"] = "1d763dd8987a122ab8596eb7d2c832****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询AI模板列表

调用ListAITemplate接口，完成查询AI模板列表功能。

接口参数和返回字段请参见ListAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询AI模板列表
 */
VodApiResponse listAITemplate(VodCredential authInfo) {
    string apiName = "ListAITemplate";
    map<string, string> args;
    // 设置模板类型，以智能审核模板为例
    args["TemplateType"] = "AIMediaAudit";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

设置默认AI模板

调用SetDefaultAITemplate接口，完成设置默认AI模板功能。

接口参数和返回字段请参见SetDefaultAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 设置默认AI模板
 */
VodApiResponse setDefaultAITemplate(VodCredential authInfo) {
    string apiName = "SetDefaultAITemplate";
    map<string, string> args;
    // 设置模板ID
    args["TemplateId"] = "1d763dd8987a122ab8596eb7d2c8****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = setDefaultAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询默认AI模板

调用GetDefaultAITemplate接口，完成查询默认AI模板功能。

接口参数和返回字段请参见GetDefaultAITemplate。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"
/**
 * 查询默认AI模板
 */
VodApiResponse getDefaultAITemplate(VodCredential authInfo) {
    string apiName = "GetDefaultAITemplate";
    map<string, string> args;
    // 设置模板类型，以智能审核模板为例
    args["TemplateType"] = "AIMediaAudit";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getDefaultAITemplate(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.14. 视频DNA

本篇文档提供了C/C++ SDK视频DNA模块相关功能的API调用示例。包含提交视频DNA作业、查询视频DNA作业、获取视频DNA结果。

初始化客户端

使用前请先初始化客户端，请参见初始化。

提交视频DNA作业

调用SubmitAIJob接口，完成提交视频DNA作业功能。

接口参数和返回字段请参见SubmitAIJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 提交作业
 */
VodApiResponse submitAIJob(VodCredential authInfo) {
    string apiName = "SubmitAIJob";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "3eb19a4585bc475e995bdd78fd****";
    // 设置AI类型，类型为AIMediaDNA
    args["Types"] = "AIMediaDNA";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = submitAIJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

查询视频DNA作业

调用ListAIJob接口，完成查询视频DNA作业功能。

接口参数和返回字段请参见ListAIJob。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询作业
 */
VodApiResponse listAIJob(VodCredential authInfo) {
    string apiName = "ListAIJob";
    map<string, string> args;
    // 设置作业ID
    args["JobIds"] = "979d4d7a36ae41b1a945a288****,3eb19a4585bc475e995bddea65****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listAIJob(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

获取视频DNA结果

调用GetMediaDNAResult接口，完成获取视频DNA结果功能。

接口参数和返回字段请参见GetMediaDNAResult。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/**
 * 查询结果
 */
VodApiResponse getMediaDNAResult(VodCredential authInfo) {
    string apiName = "GetMediaDNAResult";
    map<string, string> args;
    // 设置视频ID
    args["MediaId"] = "3eb19a4585bc475e995bdd78****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = getMediaDNAResult(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

8.15. 直播转点播

本篇文档提供了C/C++ SDK视频直播转点播模块，查询直转点视频列表功能的API调用示例。

初始化客户端

使用前请先初始化客户端，请参见初始化。

查询直转点视频列表

调用ListLiveRecordVideo接口，完成查询直转点视频列表功能。

接口参数和返回字段请参见ListLiveRecordVideo。调用示例如下：

```
#include <stdio.h>
#include <string>
#include <map>
#include <jsoncpp/json/json.h>
#include "vod_sdk/openApiUtil.h"

/*查询直播转点播视频列表函数*/
VodApiResponse listLiveRecordVideo(VodCredential authInfo) {
    string apiName = "ListLiveRecordVideo";
    map<string, string> args;
    args["StartTime"] = "2018-04-24T03:21:04Z";
    args["EndTime"] = "2018-05-21T03:21:44Z";
    args["StreamName"] = "testStream****";
    args["DomainName"] = "example.aliyundoc.com";
    args["AppName"] = "testApp****";
    return getAcsResponse(authInfo, apiName, args);
}

// 请求示例
void main() {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response = listLiveRecordVideo(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```