ALIBABA CLOUD

阿里云

视频内容检索

SDK参考

文档版本：20210622

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.回调通知说明

1. 当创建任务调用时，可设置CallbackUrl。在任务处理完成后可以及时获取任务处理完成结果通知。
2. 通知机制，接收方服务正常时仅通知1次，失败重试3次，间隔为1秒。
3. 由于通知内容较多，方便开发者解析，通过Http body方式传输JSON结果，content-type为application/json。

通知内容

```
{
  "TaskId": 1,       // 任务id
  "Status": 1,       // 任务状态（0：创建完成，1：处理中，2：处理完成，3：处理失败）
  "ProcessResultUrl": "", // 分析结果为一个json格式，具体请参考任务结果说明https://help.aliyun.com/document_detail/159724.html
  "AnalysisUseTime": 1，// 分析时长
  "RequestId" : "<requestId>", // 请求的唯一标识，每次请求都不相同
  "ClientToken" // 用于幂等校验，ClientToken相同代表同一个回调请求的多次重试
}
```

2..NET SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 将 SDK添加到项目中通过 NuGet程序包管理器来安装，在解决方案资源管理器面板中右击您的项目选择管理 NuGet程序包菜单，在打开的 NuGet管理面板中点击浏览选项卡输入：

   ```
   AlibabaCloud.SDK.Videosearch20200225
   (最新版本1.1.2)
   ```

   选择并点击安装即可。

完整代码示例

```
usingSystem;
usingAlibabaCloud.RPCClient.Models;
usingAlibabaCloud.SDK.Videosearch20200225;
usingAlibabaCloud.SDK.Videosearch20200225.Models;
usingAlibabaCloud.TeaUtil.Models;
usingNewtonsoft.Json;
usingTea;

namespaceConsoleApplication1
{
internal classProgram
{
public static voidMain(string[]args)
{
Configconfig =newConfig();
config.AccessKeyId="<yourAccessKeyId>";
config.AccessKeySecret="<yourAccessKey>";
config.RegionId="<regionId>";
config.Endpoint="<endpoint>";
Clientclient =newClient(config);
// 创建视频入库任务
 AddStorageVideoTask(client);

// 创建检索任务
 addSearchVideoTask(client);

// 获取任务状态
 getTaskStatus(client);

// 获取入库历史
 getStorageHistory(client);

// 删除视频
 addDeletionVideoTask(client);

// 查询批量任务
 listBatchTask(client);
```

```
//查询音频实例检索任务列表
 ListSearchAudioTask(client);

//音频实例入库
 listStorageAudioTasks(client);

//创建批量任务
createBatchTask(client);
}



private static voidAddStorageVideoTask(Clientclient)
{
AddStorageVideoTaskRequestrequest =newAddStorageVideoTaskRequest();
// 实例id
request.InstanceId="<yourInstanceId>";
// 视频url
request.VideoUrl="<yourVideoUrl>";
// 视频id（业务key）
request.VideoId="<yourVideoId>";
// 视频描述
request.Description="<yourDescription>";
// 视频标签
request.VideoTags="<yourVideoTags>";
// 入库完成的回调
// request.CallbackUrl = "<yourCallbackUrl>";
try
{
AddStorageVideoTaskResponseresponse = client.AddStorageVideoTask(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+JsonConvert.SerializeObject(response.Data));
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}

private static voidaddSearchVideoTask(Clientclient)
{
AddSearchVideoTaskRequestrequest =newAddSearchVideoTaskRequest();

// 实例id
request.InstanceId="<yourInstanceId>";
// 检索类型（1. 视频搜视频  2. 图片搜视频）
request.SearchType=1;
// 视频/图片的url地址
```

```
// 视频/图片的url地址
//如果需要使用 本地文件上传的功能,则用xxxFileObject参数并且使用 AddSearchXXXTaskAdvanceRequest对象
request.VideoUrl="<yourVideoUrl>";
// 视频/图片的描述
request.Description="<yourDescription>";
// 检索标签
// request.QueryTags = "<yourQueryTags>";
// 返回结果数
request.ReturnVideoNumber=20;
// 查询入库类型，当检索类型为1时生效（1. 直接入库 2. 去重入库 3. 不入库）
request.StorageType=3;
// 视频id，当searchType为1且storageType为1，2时生效
request.VideoId="123";
// 视频标签，当searchType为1且storageType为1，2时生效
request.VideoTags="123";
// 去重入库阈值，searchType为1且storageType为2时生效
request.ReplaceStorageThreshold=0.75F;
// 检索完成的回调
// request.CallbackUrl = "<yourCallbackUrl>";
try
{
AddSearchVideoTaskResponseresponse = client.AddSearchVideoTask(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+JsonConvert.SerializeObject(response.Data));
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}


private static voidgetTaskStatus(Clientclient)
{
GetTaskStatusRequestrequest =newGetTaskStatusRequest();
// 实例id
request.InstanceId="<yourInstanceId>";
// 任务id
request.TaskId="<yourTaskId>";
try
{
GetTaskStatusResponseresponse = client.GetTaskStatus(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+ response.Data);
Console.WriteLine("TaskInfo : "+JsonConvert.SerializeObject(response.TaskInfo));
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
```

```
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}


private static voidgetStorageHistory(Clientclient)
{
GetStorageHistoryRequestrequest =newGetStorageHistoryRequest();
// 实例id
request.InstanceId="<yourInstanceId>";
// 视频id
request.VideoId="<yourVideoId>";
// 当前页数
request.PageNumber=1;
// 分页大小
request.PageSize=10;
try
{
GetStorageHistoryResponseresponse = client.GetStorageHistory(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+JsonConvert.SerializeObject(response.Data));
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}

private static voidaddDeletionVideoTask(Clientclient)
{
AddDeletionVideoTaskRequestrequest =newAddDeletionVideoTaskRequest();
// 实例id
request.InstanceId="<yourInstanceId>";
// 视频id
request.VideoId="<yourVideoId>";
try
{
AddDeletionVideoTaskResponseresponse = client.AddDeletionVideoTask(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+ response.Data);
}
catch(TeaUnretryableExceptione)
{
```

```
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}

private static voidlistBatchTask(Clientclient)
{
ListBatchTaskRequestrequest =newListBatchTaskRequest();
// 实例id
request.InstanceId="<yourInstanceId>";

try
{
ListBatchTaskResponseresponse = client.ListBatchTask(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+ response.Data);
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
Console.WriteLine("Message结果："+ e.Message);
Console.WriteLine("打印String:"+ e);
}
}


private static voidListSearchAudioTask(Clientclient)
{
ListSearchAudioTasksRequestrequest =newListSearchAudioTasksRequest();
// 实例id
request.InstanceId="<yourInstanceId>";

try
{
ListSearchAudioTasksResponseresponse = client.ListSearchAudioTasks(request,newRuntimeOptions());
Console.WriteLine("requestId : "+ response.RequestId);
Console.WriteLine("Data : "+ response.Data);
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果："+ e.InnerException.Message);
}
Console.WriteLine("Data结果："+ e.Data);
```

```
Console.WriteLine("Message结果：" + e.Message);
Console.WriteLine("打印String:" + e);
}
}

private static voidlistStorageAudioTasks(Clientclient)
{
ListStorageAudioTasksRequestrequest =newListStorageAudioTasksRequest();
// 实例id
request.InstanceId="<yourInstanceId>";

try
{
ListStorageAudioTasksResponseresponse = client.ListStorageAudioTasks(request,newRuntimeOptions());
Console.WriteLine("requestId : " + response.RequestId);
Console.WriteLine("Data : " + response.Data);
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果：" + e.InnerException.Message);
}
Console.WriteLine("Data结果：" + e.Data);
Console.WriteLine("Message结果：" + e.Message);
Console.WriteLine("打印String:" + e);
}
}

private static voidcreateBatchTask(Clientclient)
{
CreateBatchTaskRequestrequest =newCreateBatchTaskRequest();
// 实例id
request.InstanceId="<yourInstanceId>";
request.BatchTaskType=1;
request.RoleArn="<yourRoleArn>";
request.OssBucketName="<yourOssBucketName>";
request.OssDataPath="<yourOssDataPath>";
request.OssMetaFile="<yourOssMetaFile>";

try
{
CreateBatchTaskResponseresponse = client.CreateBatchTask(request,newRuntimeOptions());
Console.WriteLine("requestId : " + response.RequestId);
Console.WriteLine("Data : " + response);
}
catch(TeaUnretryableExceptione)
{
if(e.InnerException!=null)
{
Console.WriteLine("InnerException结果：" + e.InnerException.Message);
}
Console.WriteLine("Data结果：" + e.Data);
Console.WriteLine("Message结果：" + e.Message);
```

```
Console.WriteLine("打印String:"+ e);
}
}

}
}
```

3.Go SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 安装Go SDK。阿里云Go SDK支持Go 1.7及以上版本，您可以通过以下方式安装Go SDK：使用Govendor安装。执行以下命令，安装阿里云Go SDK：

```
go get -u github.com/alibabacloud-go/videosearch-20200225/client
```

go.mod中引入SDK依赖

```
require github.com/alibabacloud-go/videosearch-20200225 v1.1.2
```

完整代码示例

```
package main

import(
"bytes"
"encoding/json"
"fmt"
  tea "github.com/alibabacloud-go/tea-rpc/client"
  multisearch "github.com/alibabacloud-go/videosearch-20200225/client"
)

func main(){

  config :=new(tea.Config)
  config.SetAccessKeyId("<yourAccessKeyId>")
  config.SetAccessKeySecret("<yourAccessKeySecret>")
  config.SetRegionId("<yourRegionId>")
  config.SetEndpoint("multisearch.<regionId>.aliyuncs.com")

  client, err := multisearch.NewClient(config)

if err !=nil{
    panic(err)
}

// 创建入库任务
  addStorageVideoTask(client)

// 创建检索任务
  addSearchVideoTask(client)

// 获取任务状态
  getTaskStatus(client)

// 获取入库历史
  getStorageHistory(client)
```

```
// 删除视频
  addDeletionVideoTask(client)

 // 查询批量任务
 ListBatchTasks(client)

 //查询音频实例检索任务列表
 ListSearchAudioTasks(client)

 //查询音频实例入库任务列表
 ListStorageAudioTasks(client)

//创建批量任务
 CreateBatchTask(client)
}

func CreateBatchTask(client *multisearch.Client) {
 request := new(multisearch.CreateBatchTaskRequest)
 // 实例id

 request.SetBatchTaskType(1)

 request.SetInstanceId("<yourInstanceId>")

 request.SetRoleArn("<yourRoleArn>")

 request.SetOssBucketName("<yourBucketName>")

 request.SetOssDataPath("<yourOssDataPath>")

 request.SetOssMetaFile("<yourOssMetaFile>")

 response, err := client.CreateBatchTask(request,new(service.RuntimeOptions))
 if err != nil {
  panic(err)
 }
 data, err := json.Marshal(response)
 if err != nil {
  panic(err)
 }
 fmt.Println(fmt.Sprintf("RequestId : %s", *response.RequestId))
 fmt.Println(fmt.Sprintf("Data : %s", string(data)))
}

func ListStorageAudioTasks(client *multisearch.Client) {
 request := new(multisearch.ListStorageAudioTasksRequest)
 // 实例id
 request.SetInstanceId("<yourInstanceId>")

 response, err := client.ListStorageAudioTasks(request,new(service.RuntimeOptions))
 if err != nil {
  panic(err)
 }
 data, err := json.Marshal(response.Data)
```

```
 if err != nil {
  panic(err)
 }
 fmt.Println(fmt.Sprintf("RequestId : %s", *response.RequestId))
 fmt.Println(fmt.Sprintf("Data : %s", string(data)))
}

func ListSearchAudioTasks(client *multisearch.Client) {
 request := new(multisearch.ListSearchAudioTasksRequest)
 // 实例id
 request.SetInstanceId("<yourInstanceId>")

 response, err := client.ListSearchAudioTasks(request,new(service.RuntimeOptions))
 if err != nil {
  panic(err)
 }
 data, err := json.Marshal(response.Data)
 if err != nil {
  panic(err)
 }
 fmt.Println(fmt.Sprintf("RequestId : %s", *response.RequestId))
 fmt.Println(fmt.Sprintf("Data : %s", string(data)))
}

func ListBatchTasks(client *multisearch.Client) {
 request := new(multisearch.ListBatchTaskRequest)
 // 实例id
 request.SetInstanceId("<yourInstanceId>")

 response, err := client.ListBatchTask(request,new(service.RuntimeOptions))
 if err != nil {
  panic(err)
 }
 data, err := json.Marshal(response.Data)
 if err != nil {
  panic(err)
 }
 fmt.Println(fmt.Sprintf("RequestId : %s", *response.RequestId))
 fmt.Println(fmt.Sprintf("Data : %s", string(data)))
}



func addStorageVideoTask(client *multisearch.Client){
   request :=new(multisearch.AddStorageVideoTaskRequest)
// 实例id
   request.SetInstanceId("<yourInstanceId>")
// 视频url
   request.SetVideoUrl("<yourVideoUrl>")
// 视频id（业务key）
   request.SetVideoId("<yourVideoId>")
// 视频描述
   request.SetDescription("<yourDescription>")
// 视频标签
```

```
    request.SetVideoTags("<yourVideoTags>")
// 入库完成的回调
    request.SetCallbackUrl("<yourCallbackUrl>")

    response, err := client.AddStorageVideoTask(request)

if err !=nil{
        panic(err)
}

    data, err := json.Marshal(response.Data)
if err !=nil{
        panic(err)
}
    fmt.Println(fmt.Sprintf("RequestId : %s",*response.RequestId))
    fmt.Println(fmt.Sprintf("Data : %s",string(data)))
}

func addSearchVideoTask(client *multisearch.Client){
    request :=new(multisearch.AddSearchVideoTaskRequest)

// 实例id
    request.SetInstanceId("<yourInstanceId>")
// 检索类型（1. 视频搜视频  2. 图片搜视频）
    request.SetSearchType(1)
// 视频/图片的url地址
//如果需要使用 本地文件上传的功能,则用xxxFileObject参数并且使用 AddSearchXXXTaskAdvanceRequest对象
    request.SetVideoUrl("<yourVideoUrl>")
// 视频/图片的描述
    request.SetDescription("<yourDescription>")
// 检索标签
    request.SetQueryTags("<yourQueryTags>")
// 返回结果数
    request.SetReturnVideoNumber(20)
// 查询入库类型，当检索类型为1时生效（1. 直接入库 2. 去重入库 3. 不入库）
    request.SetStorageType(3)
// 视频id，当searchType为1且storageType为1，2时生效
    request.SetVideoId("<yourVideoId>")
// 视频标签，当searchType为1且storageType为1，2时生效
    request.SetVideoTags("<yourVideoTags>")
// 去重入库阈值，searchType为1且storageType为2时生效
    request.SetReplaceStorageThreshold(0.75)
// 检索完成的回调
    request.SetCallbackUrl("<yourCallbackUrl>")

    response, err := client.AddSearchVideoTask(request)
if err !=nil{
        panic(err)
}

    data, err := json.Marshal(response.Data)
if err !=nil{
        panic(err)
}
    fmt.Println(fmt.Sprintf("RequestId : %s",*response.RequestId))
```

```
  fmt.Println(fmt.Sprintf("RequestId : %s", response.RequestId))
  fmt.Println(fmt.Sprintf("Data : %s",string(data)))
}

func getTaskStatus(client *multisearch.Client){
  request :=new(multisearch.GetTaskStatusRequest)
// 实例id
  request.SetInstanceId("<yourInstanceId>")
// 任务id
  request.SetTaskId("<yourTaskId>")

  response, err := client.GetTaskStatus(request)
if err !=nil{
    panic(err)
}

  data := bytes.NewBuffer([]byte{})
  jsonEncoder := json.NewEncoder(data)
  jsonEncoder.SetEscapeHTML(false)
  err = jsonEncoder.Encode(response.TaskInfo)
if err !=nil{
    panic(err)
}
  fmt.Println(fmt.Sprintf("RequestId : %s",*response.RequestId))
  fmt.Println(fmt.Sprintf("Data : %d", response.Data))
  fmt.Println(fmt.Sprintf("TaskInfo : %s", data.String()))
}

func getStorageHistory(client *multisearch.Client){
  request :=new(multisearch.GetStorageHistoryRequest)
// 实例id
  request.SetInstanceId("<yourInstanceId>")
// 视频id
  request.SetVideoId("<yourVideoId>")
// 当前页数
  request.SetPageNumber(1)
// 分页大小
  request.SetPageSize(10)

  response, err := client.GetStorageHistory(request)
if err !=nil{
    panic(err)
}

  data, err := json.Marshal(response.Data)
  fmt.Println(fmt.Sprintf("RequestId : %s",*response.RequestId))
  fmt.Println(fmt.Sprintf("Data : %s",string(data)))
}

func addDeletionVideoTask(client *multisearch.Client){
  request :=new(multisearch.AddDeletionVideoTaskRequest)
// 实例id
  request.SetInstanceId("<yourInstanceId>")
// 视频id
  request.SetVideoId("<yourVideoId>")
```

```
    response, err := client.AddDeletionVideoTask(request)
if err !=nil{
    panic(err)
}

    fmt.Println(fmt.Sprintf("RequestId : %s",*response.RequestId))
    fmt.Println(fmt.Sprintf("Data : %t",*response.Data))
}
```

4.Java SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 将 SDK添加到项目中引入 Java SDK 依赖，通过 maven 依赖的方式将 MultiSearch 的SDK加入到自己的项目中。

```
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>videosearch20200225</artifactId>
<version>1.1.2</version>
</dependency>
```

完整代码示例

v1.1.2 更新

*1.账号，Config，Client 均为标准格式，下面不做重复说明，均已**setUp()**为说明*

*2.参数及返回值详细说明请参考**HTTP调用接口文档***

*3.详细出入参可查看**HTTP调用接口文档***

公共参数

```
public Clinet setUp() {

  // 设置Region以北京为例，其它Region请按实际情况填写。
  String region ="cn-beijing";
  // 阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维，请登录 https://ram.console.aliyun.com 创建RAM账号。
  String accessKeyId ="<yourAccessKeyId>";
  String accessKeySecret ="<yourAccessKeySecret>";

  // 创建Config
  Config config =newConfig();
  config.regionId ="<yourRegionId>";
  config.accessKeyId ="<yourAccessKeyId>";
  config.accessKeySecret ="<yourAccessKeySecret>";
  config.endpoint="multisearch.<regionId>.aliyuncs.com";

  return new Client(config);

}
```

一、视频指纹

创建视频入库任务(文件方式)

```
  public String addStorageVideoTaskForFile(AddStorageVideoTaskParams addStorageVideoTaskParams)thr
ows Exception{

    Client client = setUp();
    AddStorageVideoTaskAdvanceRequest request = new AddStorageVideoTaskAdvanceRequest();
    // 处理任务的实例id
    request.setInstanceId(instanceId);
    // 视频oss url
    request.setVideoUrl(addStorageVideoTaskParams.getVideoUrl());
    // 视频文件流
    request.setVideoFileObject(addStorageVideoTaskParams.getVideoFileObject());
    // 视频的业务key
    request.setVideoId(addStorageVideoTaskParams.getVideoId());
    // 视频的业务标签
    request.setVideoTags(addStorageVideoTaskParams.getVideoTags());
    // 入库回调地址
    request.setCallbackUrl(addStorageVideoTaskParams.getCallbackUrl());

    request.setDescription(addStorageVideoTaskParams.getDescription());

    request.setSort(addStorageVideoTaskParams.getSort());
    System.out.println(JSON.toJSON(request).toString());
    // 发起请求并处理应答
    AddStorageVideoTaskResponse response = client.addStorageVideoTaskAdvance(request,runtimeOptio
ns);
    // 获取任务id
    System.out.println("TaskId： " + response.getData().getTaskId());
    //System.out.println("RequestId： " + response.getRequestId());
    return response.getData().getTaskId();
  }
```

创建视频入库任务

```
try{
Client client =setUp();

AddStorageVideoTaskRequest request =newAddStorageVideoTaskRequest();
// 实例id
    request.setInstanceId("<yourInstanceId>");
// 视频url
    request.setVideoUrl("<yourVideoUrl>");
// 视频id（业务key）
    request.setVideoId("<yourVideoId>");
// 视频描述
    request.setDescription("<yourDescription>");
// 视频标签
    request.setVideoTags("<yourVideoTag>");
// 入库完成的回调
    request.setCallbackUrl("<yourCallbackUrl>");

AddStorageVideoTaskResponse response = client.addStorageVideoTask(request);

System.out.println("requestId : "+ response.getRequestId());
System.out.println("Data : "+ JSON.toJSONString(response.getData()));
}catch(TeaException e){
System.out.println(JSON.toJSONString(e.getData()));
System.out.println(e.getCode());
System.out.println(e.getMessage());
}catch(Exception e){
    e.printStackTrace();
}
```

创建视频检索任务（文件方式）

```
    Client client = setUp();

    AddSearchVideoTaskAdvanceRequest request = new AddSearchVideoTaskAdvanceRequest();
    // 视频oss url
    request.setVideoUrl(<yourVideoUrl>);
    // 处理任务的实例id
    request.setInstanceId(<yourInstanceId>);
    // 视频的业务key
    request.setVideoId(<yourVIdeoId>);
    // 视频的业务标签
    request.setVideoTags(<yourVideoTags>);
    // 返回视频数量
    request.setReturnVideoNumber(<yourVideoNumber>);
    // 查询标签
    request.setQueryTags(<yourQueryTags>);
    // 检索入库类型 1为视频搜索
    request.setStorageType(<yourStorageType>);
    // 检索
    request.setCallbackUrl(<yourCallbackUr>);
    // 覆盖入库的阈值，当searchStorageType为2(覆盖入库)时生效
    request.setReplaceStorageThreshold(<yourStorageThreshold>);
    //视频描述
    request.setDescription(<yourDescription>);
    request.setSearchType(<yourSeacrchType>);
    //上传文件流
    request.setVideoFileObject(<yourFileObject>);
    //设置优先级，(0.低 1.中 2.高 3.紧急)
    request.setSort(<yourSort>);
    //是否需要特征文件 0.不需要 1.需要
    request.setNeedFeatureFile(<yourNeedFeatureFile>);

    // 发起请求并处理应答或异常
    System.out.println(JSON.toJSON(request).toString());
    AddSearchVideoTaskResponse response = client.addSearchVideoTaskAdvance(request,runtimeOptions
);
    //System.out.println(JSON.toJSONString(response.data));
    System.out.println("requestId : " + response.getRequestId());
    System.out.println("Data : " + JSON.toJSONString(response.getData()));
```

创建视频检索任务

```
Client client =setUp();


AddSearchVideoTaskRequest request =newAddSearchVideoTaskRequest();

// 实例id
    request.setInstanceId("<yourInstanceId>");
// 参考详细接口文档
    request.setSearchType(1);
// 视频/图片的url地址
//如果需要使用 本地文件上传的功能,则用xxxFileObject参数并且使用 AddSearchXXXTaskAdvanceRequest对象
    request.setVideoUrl("<yourVideoUrl>");
// 视频/图片的描述
    request.setDescription("<yourDescription>");
// 检索标签
    request.setQueryTags("<yourQueryTags>");
// 返回结果数
    request.setReturnVideoNumber(20);
// 查询入库类型，当检索类型为1时生效（1. 直接入库 2. 去重入库 3. 不入库）
    request.setStorageType(3);
// 视频id，当searchType为1且storageType为1，2时生效
    request.setVideoId("<yourVideoId>");
// 视频标签，当searchType为1且storageType为1，2时生效
    request.setVideoTags("<yourVideoTags>");
// 去重入库阈值，searchType为1且storageType为2时生效
    request.setReplaceStorageThreshold(0.75f);
// 检索完成的回调
    request.setCallbackUrl("<yourCallbackUrl>");

AddSearchVideoTaskResponse response = client.addSearchVideoTask(request);

System.out.println("requestId : "+ response.getRequestId());
System.out.println("Data : "+ JSON.toJSONString(response.getData()));
}catch(TeaException e){
System.out.println(JSON.toJSONString(e.getData()));
System.out.println(e.getCode());
System.out.println(e.getMessage());
}catch(Exception e){
    e.printStackTrace();
}
```

获取视频入库列表

```
public String ListStorageVideoTasks(ListStorageVideoTasksParams params)throws Exception{
  Client client= setUp();
  ListStorageVideoTasksRequest  request = new ListStorageVideoTasksRequest();
  request.setDescription(params.getDescription());
  request.setVideoId(params.getVideoId());
  request.setInstanceId(instanceId);
  request.setPageNumber(params.getPageNumber());
  request.setPageSize(params.getPageSize());
  request.setStatusList(params.getStatusList());
  request.setSortList(params.getSortlist());
  request.setTaskId(params.getTaskId());
  request.setStorageInfoList(params.getStorageInfoList());
  //System.out.println(JSON.toJSON(request).toString());
  ListStorageVideoTasksResponse response = client.listStorageVideoTasks(request,runtimeOptions);
  System.out.println(JSON.toJSONString(response.data));
  return JSON.toJSONString(response.data);
}
```

获取视频检索任务列表

```
public String listSearchVideoTasks(ListSearchVideoTasksParams params) throws Exception{
  Client client= setUp();
  ListSearchVideoTasksRequest  request = new ListSearchVideoTasksRequest();
  request.setDescription(params.getDescription());
  request.setInstanceId(instanceId);
  request.setPageNumber(params.getPageNumber());
  request.setPageSize(params.getPageSize());
  request.setStatusList(params.getStatusList());
  request.setSortList(params.getSortList());
  request.setTaskId(params.getTaskId());
  request.setSearchTypeList(params.getSearchTypeList());
  ListSearchVideoTasksResponse response = client.listSearchVideoTasks(request,runtimeOptions);
  System.out.println(JSON.toJSONString(response.getData()));
  return JSON.toJSONString(response.data);
}
```

二、音频指纹

创建音频任务(文件方式)

```
 Client client = setUp();
AddStorageAudioTaskAdvanceRequest request = new AddStorageAudioTaskAdvanceRequest();
// 处理任务的实例id
request.setInstanceId(<yourInstanceId>);
// oss url
request.setAudioUrl(<yourAudioUrl>);
// 文件流
request.setAudioFileObject(<yourFileObjcet>);
// 业务key
request.setAudioId(<yourAudioId>);
// 业务标签
request.setAudioTags(<yourAudioTags>);
// 入库回调地址
request.setCallbackUrl(<yourCallbackUrl>);

request.setDescription(<yourDescription>);

request.setSort(<yourSort>);
System.out.println(JSON.toJSON(request).toString());
// 发起请求并处理应答
AddStorageAudioTaskResponse response = client.addStorageAudioTaskAdvance(request,runtimeOptio
ns);
// 获取任务id
System.out.println("TaskId： " + response.getData().getTaskId());
//System.out.println("RequestId： " + response.getRequestId());
return response.getData().getTaskId();
```

创建音频入库任务

```
try{
Client client =setUp();

AddStorageAudioTaskRequest request =newAddStorageAudioTaskRequest();
// 实例id
    request.setInstanceId("<yourInstanceId>");
// url
    request.setAudioUrl("<yourAudioUrl>");
// id（业务key）
    request.setAudioId("<yourAudioId>");
//描述
    request.setDescription("<yourDescription>");
// 标签
    request.setAudioTags("<yourAudioTag>");
// 入库完成的回调
    request.setCallbackUrl("<yourCallbackUrl>");

AddStorageAudioTaskResponse response = client.addStorageAudioTask(request);

System.out.println("requestId : "+ response.getRequestId());
System.out.println("Data : "+ JSON.toJSONString(response.getData()));
}catch(TeaException e){
System.out.println(JSON.toJSONString(e.getData()));
System.out.println(e.getCode());
System.out.println(e.getMessage());
}catch(Exception e){
    e.printStackTrace();
}
```

创建音频检索任务(文件方式)

```
    Client client = setUp();

    AddSearchAudioTaskAdvanceRequest request = new AddSearchAudioTaskAdvanceRequest();
    // oss url
    request.setAudioUrl(<yourAudioUrl>);
    // 处理任务的实例id
    request.setInstanceId(<yourInstanceId>);
    request.setAudio(<yourAudioId>);
    request.setAudioTags(<yourAudioTags>);
    request.setReturnAudioNumber(<yourAudioNumber>);
    // 查询标签
    request.setQueryTags(<yourQueryTags>);
    request.setStorageType(<yourStorageType>);
    // 检索
    request.setCallbackUrl(<yourCallbackUr>);
    // 覆盖入库的阈值，当searchStorageType为2(覆盖入库)时生效
    request.setReplaceStorageThreshold(<yourStorageThreshold>);
    //描述
    request.setDescription(<yourDescription>);
    request.setSearchType(<yourSeacrchType>);
    //上传文件流
    request.setAudioFileObject(<yourFileObject>);
    //设置优先级，(0.低 1.中 2.高 3.紧急)
    request.setSort(<yourSort>);
    //是否需要特征文件 0.不需要 1.需要
    request.setNeedFeatureFile(<yourNeedFeatureFile>);

    // 发起请求并处理应答或异常
    System.out.println(JSON.toJSON(request).toString());
    AddSearchAudioTaskResponse response = client.addSearchAudioTaskAdvance(request,runtimeOptions
);
    //System.out.println(JSON.toJSONString(response.data));
    System.out.println("requestId : " + response.getRequestId());
    System.out.println("Data : " + JSON.toJSONString(response.getData()));
```

创建音频检索任务

```
Client client =setUp();


AddSearchAudioTaskRequest request =newAddSearchAudioTaskRequest();

// 实例id
    request.setInstanceId("<yourInstanceId>");
// 参考接口入参文档
    request.setSearchType(1);
    request.setAudioUrl("<yourAudioUrl>");
    request.setDescription("<yourDescription>");
// 检索标签
    request.setQueryTags("<yourQueryTags>");
// 返回结果数
    request.setReturnVideoNumber(20);
// 查询入库类型，当检索类型为1时生效（1. 直接入库 2. 去重入库 3. 不入库）
    request.setStorageType(3);
    request.setAudioId("<yourAudioId>");
    request.setAudioTags("<yourAudioTags>");
    request.setReplaceStorageThreshold(0.75f);
// 检索完成的回调
    request.setCallbackUrl("<yourCallbackUrl>");

AddSearchAudioTaskResponse response = client.addSearchAudioTask(request);

System.out.println("requestId : "+ response.getRequestId());
System.out.println("Data : "+ JSON.toJSONString(response.getData()));
}catch(TeaException e){
System.out.println(JSON.toJSONString(e.getData()));
System.out.println(e.getCode());
System.out.println(e.getMessage());
}catch(Exception e){
    e.printStackTrace();
}
```

获取音频入库列表

```
     Client client= setUp();
    ListStorageAudioTasksRequest  request = new ListStorageAudioTasksRequest();
    request.setDescription(<yourDescription>);
    request.setAudioId(<yourAudioId>);
    request.setInstanceId(<yourInstanceId>);
    request.setPageNumber(<yourPageNumber>);
    request.setPageSize(<yourPageSize>);
    request.setStatusList(<yourStatusList>);
    request.setSortList(<yourSortList>);
    request.setTaskId(<yourTaskId>);
    request.setStorageInfoList(<yourStorageInfoList>);
    //System.out.println(JSON.toJSON(request).toString());
    ListStorageAudioTasksResponse response = client.listStorageAudioTasks(request,runtimeOptions);
    System.out.println(JSON.toJSONString(response.data));
```

获取音频检索任务列表

```
    Client client= setUp();
    ListSearchAudioTasksRequest  request = new ListSearchAudioTasksRequest();
    request.setDescription(<yourDescription>);
    request.setInstanceId(<yourInstanceId>);
    request.setPageNumber(<yourPageNumber>);
    request.setPageSize(<yourPageSize>);
    request.setStatusList(<yourStatusList>);
    request.setSortList(<yourSortList>);
    request.setTaskId(<yourTaskId>);
    ListSearchAudioTasksResponse response = client.listSearchAudioTasks(request,runtimeOptions);
    System.out.println(JSON.toJSONString(response.getData()));
```

三、通用实例

获取任务状态

```
  public Map<String, String> getTaskStatus(String taskId, String Id) throws TeaException,Exception {
    Client client = setUp();
    GetTaskStatusRequest request = new GetTaskStatusRequest();
    // 任务id
    request.setTaskId(taskId);
    // 实例Id
    request.setInstanceId(Id);
    GetTaskStatusResponse response = client.getTaskStatus(request,runtimeOptions);
    System.out.println("requestId: "+response.requestId);
    System.out.println("Data : " + JSON.toJSONString(response.getData())+", ResultOss：  "+response.getTa
skInfo().getProcessResultOss());
    System.out.println("TaskInfo : " + JSON.toJSONString(response.getTaskInfo()));
    Map<String,String> map = new HashMap();
    map.put("Data",response.getData().toString());
    map.put("taskId",String.valueOf(response.getTaskInfo().getTaskId()));
    map.put("ResultOSS",response.getTaskInfo().getProcessResultOss());
    return map;
  }
```

获取实例列表

```
public String listInstaces(ListInstacesParams listInstacesParams)throws Exception{
  Client client = setUp();

  ListInstancesRequest request = new ListInstancesRequest();
  //实例名称
  request.setInstanceName(listInstacesParams.getInstanceName());
  //实例类型
  request.setInstanceType(listInstacesParams.getInstanceType());
  request.setPageNumber(listInstacesParams.getPageNumber());
  request.setPageSize(listInstacesParams.getPageSize());
  request.setStatus(listInstacesParams.getStatus());
  request.setTags(listInstacesParams.getTags());
  System.out.println(JSON.toJSON(request).toString());
  ListInstancesResponse response = client.listInstancesSimply(request);
  System.out.println(JSON.toJSONString(response.getData()));
  return  JSON.toJSONString(response.getData());
}
```

更改优先级

```
public Boolean modifyPriority(ModifyPriorityParams modifyPriorityParams) throws Exception{
  Client client = setUp();

  ModifyPriorityRequest request = new ModifyPriorityRequest();
  request.setSort(modifyPriorityParams.getSort());
  request.setTaskId(modifyPriorityParams.getTaskId());

  System.out.println(JSON.toJSON(request).toString());
  ModifyPriorityResponse response = client.modifyPriority(request,runtimeOptions);

  System.out.println("taskId:"+response.getRequestId());
  System.out.println("修改结果："+response.data);
  return response.getData();
}
```

创建批量任务（OSS方式导入）

```
public Long createBatchTaskForOss( CreateBatchTaskParams params)throws Exception{

  Client client = setUp();
  CreateBatchTaskRequest request = new CreateBatchTaskRequest();
  request.setBatchTaskType(params.getBatchTaskType());
  request.setCallbackUrl(searchCallback);
  request.setInstanceId(instanceId);
  request.setOssBucketName(params.getOssBucketName());
  request.setOssDataPath(params.getOssDataPath());
  request.setOssMetaFile(params.getOssMetaFile());
  request.setRoleArn(params.getRoleArn());
  System.out.println(JSON.toJSON(request).toString());
  CreateBatchTaskResponse response = client.createBatchTask(request,runtimeOptions);
  System.out.println("taskId:"+response.getTaskId());
  return  response.getTaskId();
}
```

获取批量任务信息

```
public Integer  getBatchTask (String instanceId, long batchTaskId) throws Exception{
  Client client = setUp();
  GetBatchTaskRequest request = new GetBatchTaskRequest();
  request.setBatchTaskId(batchTaskId);
  request.setInstanceId(instanceId);
  GetBatchTaskResponse response = client.getBatchTask(request,runtimeOptions);
  System.out.println("状态:"+response.status);
  System.out.println("子任务Oss地址：  "+response.getSubTaskDetail());
  return  response.status;
}
```

列举批量任务列表

```
public String listBatchTask(ListBatchTaskParams params) throws Exception {
  Client client = setUp();
  ListBatchTaskRequest request = new ListBatchTaskRequest();
  request.setInstanceId(instanceId);
  request.setBatchTaskType(params.getBatchTaskType());
  request.setPageNumber(params.getPageNumber());
  request.setPageSize(params.getPageSize());
  request.setStatus(params.getStatus());
  request.setBucketName(params.getBucketName());
  request.setDataPath(params.getDataPath());
  System.out.println(JSON.toJSON(request).toString());
  ListBatchTaskResponse response = client.listBatchTask(request,runtimeOptions);

  System.out.println(JSON.toJSONString(response.data));
  return JSON.toJSONString(response.data);

}
```

5.Node.js SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 安装node.js SDK。您可以通过以下方式安装node.js SDK：使用依赖管理工具安装。执行以下命令，安装阿里云node.js SDK：

```
npm install @alicloud/videosearch20200225@1.1.2
```

完整代码示例

```
const teaClient = require('@alicloud/rpc-client')
const multisearch = require('@alicloud/videosearch20200225')

const config =new teaClient.Config({
  accessKeyId:"<yourAccessKeyId>",
  accessKeySecret:"<yourAccessKeySecret>",
  regionId:"<yourRegionId>",
  endpoint:"multisearch.<regionId>.aliyuncs.com",
})

const client =new multisearch.default(config)

// 创建视频入库任务
async function addStorageVideoTask(){
const addStorageVideoTaskRequest =new multisearch.AddStorageVideoTaskRequest()
// 实例id
  addStorageVideoTaskRequest.instanceId ="<yourInstanceId>"
// 视频url
  addStorageVideoTaskRequest.videoUrl ="<yourVideoUrl>"
// 视频id（业务key）
  addStorageVideoTaskRequest.videoId ="<yourVideoId>"
// 视频描述
  addStorageVideoTaskRequest.description ="<yourDescription>"
// 视频标签
  addStorageVideoTaskRequest.videoTags ="<yourVideoTags>"
// 入库完成的回调
  addStorageVideoTaskRequest.callbackUrl ="<yourCallbackUrl>"

const addStorageVideoTaskResponse = await client.addStorageVideoTask(addStorageVideoTaskRequest)

  console.log(addStorageVideoTaskResponse)
}

addStorageVideoTask();

// 创建视频检索任务
async function addSearchVideoTask(){
const addSearchVideoTaskRequest =new multisearch.AddSearchVideoTaskRequest()
// 实例id
  addSearchVideoTaskRequest.instanceId ="<yourInstanceId>"
```

```
// 检索类型（1. 视频搜视频  2. 图片搜视频）
  addSearchVideoTaskRequest.searchType =1
// 视频/图片的url地址
//如果需要使用 本地文件上传的功能,则用xxxFileObject参数并且使用 AddSearchXXXTaskAdvanceRequest对象
  addSearchVideoTaskRequest.videoUrl ="<yourVideoUrl>"
// 视频/图片的描述
  addSearchVideoTaskRequest.description ="<yourDescription>"
// 检索标签
  addSearchVideoTaskRequest.queryTags ="<yourQueryTags>"
// 返回结果数
  addSearchVideoTaskRequest.returnVideoNumber =20
// 查询入库类型，当检索类型为1时生效（1. 直接入库  2. 去重入库  3. 不入库）
  addSearchVideoTaskRequest.storageType =3
// 视频id，当searchType为1且storageType为1，2时生效
   addSearchVideoTaskRequest.videoId ="<yourVideoId>"
// 视频标签，当searchType为1且storageType为1，2时生效
  addSearchVideoTaskRequest.videoTags ="<yourVideoTags>"
// 去重入库阈值，searchType为1且storageType为2时生效
  addSearchVideoTaskRequest.replaceStorageThreshold =0.75
// 检索完成的回调
  addSearchVideoTaskRequest.callbackUrl ="<yourCallbackUrl>"

const addSearchVideoTaskResponse = await client.addSearchVideoTask(addSearchVideoTaskRequest)

  console.log(addSearchVideoTaskResponse)
}

addSearchVideoTask()

// 获取视频状态
async function getTaskStatus(){
const getTaskStatusRequest =new multisearch.GetTaskStatusRequest()
// 实例id
  getTaskStatusRequest.instanceId ="<yourInstanceId>"
// 任务id
  getTaskStatusRequest.taskId ="<yourTaskId>"

const getTaskStatusResponse = await client.getTaskStatus(getTaskStatusRequest)

  console.log(getTaskStatusResponse)
}

getTaskStatus()

// 获取入库记录
async function getStorageHistory(){
const getStorageHistoryRequest =new multisearch.GetStorageHistoryRequest()
// 实例id
  getStorageHistoryRequest.instanceId ="<yourInstanceId>"
// 视频id
  getStorageHistoryRequest.videoId ="<yourVideoId>"
// 当前页数
  getStorageHistoryRequest.pageNumber =1
// 分页大小
```

```
  getStorageHistoryRequest.pageSize =10

const getStorageHistoryResponse = await client.getStorageHistory(getStorageHistoryRequest)

  console.log(getStorageHistoryResponse)
}

getStorageHistory()

// 删除任务
async function addDeletionVideoTask(){
const addDeletionVideoTaskRequest =new multisearch.AddDeletionVideoTaskRequest()
// 实例id
  addDeletionVideoTaskRequest.instanceId ="<yourInstanceId>"
// 视频id
  addDeletionVideoTaskRequest.videoId ="<yourVideoId>"

const addDeletionVideoTaskResponse = await client.addDeletionVideoTask(addDeletionVideoTaskRequest)

  console.log(addDeletionVideoTaskResponse)
}

addDeletionVideoTask()
```

6.PHP SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 安装PHP SDK核心库。使用依赖包工具安装（推荐）。执行以下命令，安装视频指纹SDK库：

   ```
   composer require alibabacloud/videosearch-20200225
   (最新版本1.1.2)
   ```

   详情请参考Alibaba Cloud SDK for PHP或自行添加解决方案。

完整代码示例

```
<?php
require __DIR__ .'/vendor/autoload.php';

use AlibabaCloud\SDK\Videosearch\V20200225\Models\AddDeletionVideoTaskRequest;
use AlibabaCloud\SDK\Videosearch\V20200225\Models\AddSearchVideoTaskRequest;
use AlibabaCloud\SDK\Videosearch\V20200225\Models\AddStorageVideoTaskRequest;
use AlibabaCloud\SDK\Videosearch\V20200225\Models\GetStorageHistoryRequest;
use AlibabaCloud\SDK\Videosearch\V20200225\Models\GetTaskStatusRequest;
use AlibabaCloud\SDK\Videosearch\V20200225\Videosearch;
use AlibabaCloud\Tea\Exception\TeaUnableRetryError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;

$config =newConfig();
$config->accessKeyId ="<yourAccessKeyId>";
$config->accessKeySecret ="<yourAccessKeySecret>";
$config->regionId ="<yourRegionId>";
$config->endpoint ="multisearch.<regionId>.aliyuncs.com";

$client =newVideosearch($config);

// 创建入库任务
$request =newAddStorageVideoTaskRequest();
// 实例id
$request->instanceId ="<yourInstanceId>";
// 视频url
$request->videoUrl ="<yourVideoUrl>";
// 视频id（业务key）
$request->videoId ="<yourVideoId>";
// 视频描述
$request->description ="<yourDescription>";
// 视频标签
$request->videoTags ="<yourVideoTag>";
// 入库完成的回调
$request->callbackUrl ="<yourCallbackUrl>";

try{
    $response = $client->addStorageVideoTask($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOption
s());
```

```
  var_dump($response->toMap());
}catch(TeaUnableRetryError $e){
  var_dump($e->getLastException(), $e->getLastRequest());
}

// 创建检索任务
$request =newAddSearchVideoTaskRequest();

// 实例id
$request->instanceId ="<yourInstanceId>";
// 检索类型（1. 视频搜视频  2. 图片搜视频）
$request->searchType =1;
// 视频/图片的url地址
//如果需要使用 本地文件上传的功能,则用xxxFileObject参数并且使用 AddSearchXXXTaskAdvanceRequest对象
$request->videoUrl ="<yourVideoUrl>";
// 视频/图片的描述
$request->description ="<yourDescription>";
// 检索标签
$request->queryTags ="<yourQueryTags>";
// 返回结果数
$request->returnVideoNumber =20;
// 查询入库类型，当检索类型为1时生效（1. 直接入库 2. 去重入库 3. 不入库）
$request->storageType =3;
// 视频id，当searchType为1且storageType为1，2时生效
$request->videoId ="<yourVideoId>"
// 视频标签，当searchType为1且storageType为1，2时生效
$request->videoTags ="<yourVideoTags>";
// 去重入库阈值，searchType为1且storageType为2时生效
$request->replaceStorageThreshold =0.75;
// 检索完成的回调
$request->callbackUrl ="<yourCallbackUrl>";

try{
  $response = $client->addSearchVideoTask($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions())
;
  var_dump($response->toMap());
}catch(TeaUnableRetryError $e){
  var_dump($e->getLastException(), $e->getLastRequest());
}

// 获取任务状态
$request =newGetTaskStatusRequest();
// 实例id
$request->instanceId ="<yourInstanceId>";
// 任务id
$request->taskId ="<yourTaskId>";

try{
  $response = $client->getTaskStatus($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions());
  var_dump($response->toMap());
}catch(TeaUnableRetryError $e){
  var_dump($e->getLastException(), $e->getLastRequest());
}

// 获取入库库内
```

```
// 获取入库历史
$request =newGetStorageHistoryRequest();
// 实例id
$request->instanceId ="<yourInstanceId>";
// 视频id
$request->videoId ="<yourVideoId>";
// 当前页数
$request->pageNumber =1;
// 分页大小
$request->pageSize =10;

try{
  $response = $client->getStorageHistory($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions());
  var_dump($response->toMap());
}catch(TeaUnableRetryError $e){
  var_dump($e->getLastException(), $e->getLastRequest());
}

// 删除任务
$request =newAddDeletionVideoTaskRequest();
// 实例id
$request->instanceId ="<yourInstanceId>";
// 视频id
$request->videoId ="<yourVideoId>";

try{
  $response = $client->addDeletionVideoTask($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions(
));
  var_dump($response->toMap());
}catch(TeaUnableRetryError $e){
  var_dump($e->getLastException(), $e->getLastRequest());
}

//查询批量任务
$request = new \AlibabaCloud\SDK\Videosearch\V20200225\Models\ListBatchTaskRequest();
$request->instanceId = "<yourInstanceId>";

try {

  $response = $client->listBatchTask($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions());
  var_dump($response->toMap());
  exit;
} catch (TeaUnableRetryError $e) {
  var_dump($e->getLastException(), $e->getLastRequest());
}

//创建批量任务

$request = new \AlibabaCloud\SDK\Videosearch\V20200225\Models\CreateBatchTaskRequest();
$request->batchTaskType = "<yourBatchTaskType>";
$request->instanceId = "<yourInstanceId>";
$request->roleArn = "<yourRoleArn>";
$request->ossBucketName = "<yourBucketName>";
$request->ossDataPath = "<yourDataPath>";
$request->ossMetaFile = "<yourOssMetaFile>";
```

```
$request->ossMetaFile = "<yourOssMetaFile>";

try {

  $response = $client->createBatchTask($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions());
  var_dump($response->toMap());
  exit;
} catch (TeaUnableRetryError $e) {
  var_dump($e->getLastException(), $e->getLastRequest());
}

//查询音频实例检索任务列表
$request = new \AlibabaCloud\SDK\Videosearch\V20200225\Models\ListSearchAudioTasksRequest();
$request->instanceId = "<yourInstanceId>";

try {

  $response = $client->listSearchAudioTasks($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions()
);
  var_dump($response->toMap());
  exit;
} catch (TeaUnableRetryError $e) {
  var_dump($e->getLastException(), $e->getLastRequest());
}

//查询音频实例检索任务列表
$request = new \AlibabaCloud\SDK\Videosearch\V20200225\Models\ListStorageAudioTasksRequest();
$request->instanceId = "<yourInstanceId>";

try {

  $response = $client->listStorageAudioTasks($request,new \AlibabaCloud\Tea\Utils\Utils\RuntimeOptions(
));
  var_dump($response->toMap());
  exit;
} catch (TeaUnableRetryError $e) {
  var_dump($e->getLastException(), $e->getLastRequest());
}
```

7.Python SDK

准备工作

1. 在安装和使用阿里云SDK前，确保您已经注册阿里云账号并生成访问密钥（AccessKey）。详情请参考创建AccessKey。
2. 安装Python SDK核心库。使用依赖包工具安装（推荐）。执行以下命令，安装阿里云SDK核心库：

   ```
   pip install alibabacloud_tea_rpc
   ```

   自行下载安装。您可以使用git clone或其它手段下载aliyun-openapi-python-sdk并自行添加解决方案。
3. 安装视频内容检索Python SDK。使用依赖包工具安装（推荐）。执行以下命令，安装视频指纹 Python SDK：最新版本1.2.0

   ```
   pip install alibabacloud_videosearch20200225
   ```

   自行下载安装。您可以使用git clone或其它手段下载aliyun-python-sdk-videosearch并自行添加解决方案。

完整代码示例

v1.2.0 版本

1.账号，Config，Client 均为标准格式，下面不做重复说明

*2.参数及返回值详细说明请参考**HTTP调用接口文档***

*3.详细出入参可查看**HTTP调用接口文档***

公共参数

```
from alibabacloud_tea_rpc.models import Config
from alibabacloud_videosearch20200225.client import Client
from alibabacloud_videosearch20200225.models import *
from alibabacloud_tea_util.models  import RuntimeOptions

config = Config(
    access_key_id="<yourAccessKeyId>",
    access_key_secret="<yourAccessKeySecret>",
    region_id="<yourRegionId>",
    endpoint="<yourEndpoint>"
  )

client = Client(config)
```

一、音频指纹

查询实例信息

```
def getAudioInstance(client,instanceid):

    request = GetAudioInstanceRequest()
    request.instance_id = instanceid #实例Id

    response = client.get_audio_instance(request,runtime)
    return response
```

添加音频入库(url方式)

```
def addStorageAudioTask(client,audioUrl,audioId,decsription,audioTag=None,callback=None,sort=None):
    request = AddStorageAudioTaskRequest()
    request.instance_id = INSTANCE_ID # 实例Id
    request.audio_url = audioUrl #音频地址
    request.audio_id = audioId # 音频Id
    request.description = decsription #备注
    request.audio_tags = audioTag # 标签
    request.callback_url = callback # 回调地址
    request.sort =sort    # (0, "低优先级"),(1, "中优先级"),(2, "高优先级"),(3, "紧急");

    response = client.add_storage_audio_task(request,runtime)

    return response
```

添加音频入库(文件方式)

```
def addStorageAudioTaskAdvance(client,audioUrl,audioId,decsription,audioTag=None,callback=None,sort
=None,fileobject=None,content=None):
    request = AddStorageAudioTaskAdvanceRequest()
    request.instance_id = INSTANCE_ID
    request.audio_file_object = fileobject
    request.content_source = content
    request.audio_url = audioUrl
    request.audio_id = audioId
    request.description = decsription
    request.audio_tags = audioTag
    request.callback_url = callback
    request.sort=sort


    response = client.add_storage_audio_task_advance(request,runtime)

    return response
```

查询音频入库信息

```
def addSearchAudioTask(client,audioUrl,description,queryTag=None,returnNum=10,callback=None,sort=None,needfeature=None,resourcetype=None):
    request = AddSearchAudioTaskRequest()
    request.instance_id = INSTANCE_ID
    request.audio_url = audioUrl
    request.content_source = 1  # 1. OSS导入 2. 文件上传
    request.description = description
    request.query_tags = queryTag
    request.return_audio_number = returnNum
    request.callback_url = callback
    request.sort = sort
    request.need_feature_file = needfeature
    request.resource_type = resourcetype

    response = client.add_search_audio_task(request,runtime)
    return response
```

查询音频入库信息(文件方式)

```
def addSearchAudioTaskAdvance(client,description,queryTag=None,returnNum=10,callback=None,file=None,sort=None,needfeature=None,resourcetype=None):
    request = AddSearchAudioTaskAdvanceRequest()
    request.instance_id = INSTANCE_ID
    #request.audio_url = audioUrl
    request.audio_file_object = file
    request.content_source = 2  # 1. OSS导入 2. 文件上传
    request.description = description
    request.query_tags = queryTag
    request.return_audio_number = returnNum
    request.callback_url = callback
    request.sort =sort
    request.need_feature_file = needfeature
    request.resource_type = resourcetype

    response = client.add_search_audio_task_advance(request,runtime)
    return response
```

删除音频入库

```
def addDeletionAudioTask(client,audioId):
    request = AddDeletionAudioTaskRequest()
    request.instance_id =INSTANCE_ID
    request.audio_id = audioId

    response = client.add_deletion_audio_task(request,runtime)
    return response
```

查询入库历史

```
def getAudioStorageHistory(client,audioId,pagenum=None,pagesize=None):
  request = GetAudioStorageHistoryRequest()
  request.instance_id =INSTANCE_ID
  request.audio_id = audioId
  request.page_number = pagenum
  request.page_size =pagesize

  response = client.get_audio_storage_history(request,runtime)
  return  response
```

查询音频任务状态

```
def getAudioTaskStatus(client,taskId):

  request = GetAudioTaskStatusRequest()
  request.instance_id = INSTANCE_ID
  request.task_id = taskId

  response = client.get_audio_task_status(request, runtime)
  return response
```

查询检索任务列表

```
def listSearchAudioTasks(client,taskId=None,audioId=None,status=None,description=None,pagenum=Non
e,pagesize=None,sortlist=None):

  request = ListSearchAudioTasksRequest()
  request.instance_id =INSTANCE_ID
  request.task_id = taskId    # 按照任务id查询
  request.status_list = status  # 按照状态查询
  request.audio_id = audioId   # 是否支持按照 audiid 查询?
  request.description = description  # 按照描述查询
  request.page_number = pagenum
  request.page_size =pagesize
  request.sort_list =sortlist

  response = client.list_search_audio_tasks(request,runtime)
  return  response
```

查询入库任务列表

```
def listStorageAudioTasks(client,taskId=None,audioId=None,status=None,description=None,storageInfo=None,pagenum=None,pagesize=None,sortlist=None):

    request = ListStorageAudioTasksRequest()
    request.instance_id = INSTANCE_ID
    request.task_id = taskId  # 按照任务id查询
    request.status_list = status  # 按照状态查询
    request.audio_id = audioId  # 按照audioId 查询
    request.description = description  # 按照描述查询
    request.page_number = pagenum
    request.page_size = pagesize
    request.storage_info_list = storageInfo  # 按照入库信息查询： 首次入库、覆盖入库、入库失败
    request.sort_list =sortlist

    response = client.list_storage_audio_tasks(request, runtime)
    return response
```

二、*视频指纹*

查询实例信息

```
def getVideoInstance(client,instanceid):

    request = GetInstanceRequest()
    request.instance_id = instanceid

    response = client.get_instance(request,runtime)
    return response
```

添加视频入库(url方式)

```
def addStorageVideoTask(client,videoUrl,videoId,decsription,videoTag=None,callback=None,sort=None):
    request = AddStorageVideoTaskRequest()
    request.instance_id = INSTANCE_ID
    request.video_url = videoUrl
    request.video_id = videoId
    request.description = decsription
    request.video_tags = videoTag
    request.callback_url = callback
    request.sort =sort    # (0, "低优先级"),(1, "中优先级"),(2, "高优先级"),(3, "紧急");

    response = client.add_storage_video_task(request,runtime)

    return response
```

添加视频入库(文件方式)

```
def addStorageVideoTaskAdvance(client, videoUrl, videoId, decsription, videoTag=None, callback=None, sort=None,
                               fileobject=None, content=None):
  request = AddStorageVideoTaskAdvanceRequest()
  request.instance_id = INSTANCE_ID
  request.video_file_object = fileobject
  request.content_source = content  # 1是 url 方式， 2 是 文件上传方式
  request.video_url = videoUrl
  request.video_id = videoId
  request.description = decsription
  request.video_tags = videoTag
  request.callback_url = callback
  request.sort = sort

  response = client.add_storage_video_task_advance(request, runtime)

  return response
```

查询视频入库信息

```
def addSearchVideoTask(client, videoUrl, description, queryTag=None, returnNum=10, callback=None, sort=None,
                       needfeature=None, resourcetype=None):
  request = AddSearchVideoTaskRequest()
  request.instance_id = INSTANCE_ID
  request.video_url = videoUrl
  request.content_source = 1  # 1. OSS导入 2. 文件上传
  request.description = description
  request.query_tags = queryTag
  request.return_audio_number = returnNum
  request.callback_url = callback
  request.sort = sort
  request.need_feature_file = needfeature
  request.resource_type = resourcetype

  response = client.add_search_video_task(request, runtime)
  return response
```

查询视频入库信息(文件方式)

```
  def addSearchVideoTaskAdvance(client, description, queryTag=None, returnNum=10, callback=None, file=None, sort=None,
                                needfeature=None, resourcetype=None):
    request = AddSearchVideoTaskAdvanceRequest()
    request.instance_id = INSTANCE_ID
    # request.audio_url = audioUrl
    request.video_file_object = file
    request.content_source = 2  # 1. OSS导入 2. 文件上传
    request.description = description
    request.query_tags = queryTag
    request.return_audio_number = returnNum
    request.callback_url = callback
    request.sort = sort
    request.need_feature_file = needfeature
    request.resource_type = resourcetype

    response = client.add_search_audio_task_advance(request, runtime)
    return response
```

删除视频入库

```
def addDeletionVideoTask(client,videoId):
  request = AddDeletionVideoTaskRequest()
  request.instance_id =INSTANCE_ID
  request.video_id = videoId

  response = client.add_deletion_video_task(request,runtime)
  return response
```

查询入库历史

```
def getVideoStorageHistory(client,videoId,pagenum=None,pagesize=None):
  # 查询入库历史

  request = GetStorageHistoryRequest()
  request.instance_id =INSTANCE_ID
  request.video_id = videoId
  request.page_number = pagenum
  request.page_size =pagesize

  response = client.get_storage_history(request,runtime)
  return  response
```

查询视频任务状态

```
def getVideoTaskStatus(client,taskId):
  # 查询 taskid状态
  request = GetTaskStatusRequest()
  request.instance_id = INSTANCE_ID
  request.task_id = taskId

  response = client.get_task_status(request, runtime)
  return response
```

查询检索任务列表

```
def listSearchVideoTasks(client,taskId=None,videoId=None,status=None,description=None,pagenum=None
,pagesize=None,sortlist=None):
  # 查询 检索任务列表
  request = ListSearchVideoTasksRequest()
  request.instance_id =INSTANCE_ID
  request.task_id = taskId    # 按照任务id查询
  request.status_list = status  # 按照状态查询
  request.video_id = videoId
  request.description = description  # 按照描述查询
  request.page_number = pagenum
  request.page_size =pagesize
  request.sort_list =sortlist

  response = client.list_search_video_tasks(request,runtime)
  return  response
```

查询入库任务列表

```
def listStorageVideoTasks(client,taskId=None,videoId=None,status=None,description=None,storageInfo=N
one,pagenum=None,pagesize=None,sortlist=None):
  # 查询 入库任务列表
  request = ListStorageVideoTasksRequest()
  request.instance_id = INSTANCE_ID
  request.task_id = taskId  # 按照任务id查询
  request.status_list = status  # 按照状态查询
  request.video_id = videoId
  request.description = description  # 按照描述查询
  request.page_number = pagenum
  request.page_size = pagesize
  request.storage_info_list = storageInfo  # 按照入库信息查询： 首次入库、覆盖入库、入库失败
  request.sort_list =sortlist

  response = client.list_storage_video_tasks(request, runtime)
  return response
```

三、通用示例

创建批量任务

```
def createBatchTask(client,metafile,callback=None):
  request = CreateBatchTaskRequest()
  request.instance_id =INSTANCE_ID ## 实例ID
  request.batch_task_type = 1  #批量任务类型, 必填（1. OSS导入 2. 文件上传）
  request.role_arn = ROLE_ARN
  request.oss_bucket_name = OSS_BUCKET_NAME
  request.oss_data_path = OSS_DATA_PATH
  request.oss_meta_file = metafile
  request.callback_url = callback

  respone = client.create_batch_task(request,runtime)
  return  respone
```

创建批量任务(文件方式)

```
def createBatchTaskAdvance(client,file,callback=None):
  request = CreateBatchTaskAdvanceRequest()
  request.instance_id =INSTANCE_ID
  request.batch_task_type = 2  #批量任务类型, 必填（1. OSS导入 2. 文件上传）
  request.file_url_object = file
  request.callback_url = callback

  respone = client.create_batch_task_advance(request,runtime)
  return  respone
```

查询批量任务

```
def getBatchTask(client,batchtaskId):
  request = GetBatchTaskRequest()
  request.instance_id =INSTANCE_ID
  request.batch_task_id = batchtaskId

  response  =client.get_batch_task(request,runtime)
  return  response
```

查询任务状态

```
def getTaskStatus(client,taskid):
  request = GetTaskStatusRequest()
  request.instance_id = INSTANCE_ID
  request.task_id = taskid
  response = client.get_task_status(request,runtime)
  return response
```

查询批量任务列表

```
def listBatchTask(client,bucketname=None,datapath=None,status=None,tasktype=None,pagenum=None,p
agesize=None):

  request = ListBatchTaskRequest()
  request.instance_id = INSTANCE_ID
  request.bucket_name = bucketname  #按照bucket 查询
  request.data_path = datapath   # 按照数据路径 查询
  request.status =status
  # 0. 排队中 1. 处理中（meta文件解析中） 2. 已完成 3. 失败 4. 部分失败 5. 处理中（子任务处理中）
  request.batch_task_type = tasktype  # 1. OSS 2. 文件上传  '[2]'
  request.page_number =pagenum
  request.page_size =pagesize

  response = client.list_batch_task(request,runtime)
  return  response
```

查询实例列表

```
def listInstances(client,instanceName=None,status=None,instanceType=None,tags=None,pagenum=None,p
agesize=None):

  request = ListInstancesRequest()
  request.instance_name = instanceName
  request.status = status
  request.instance_type = instanceType
  request.tags = tags
  request.page_number = pagenum
  request.page_size = pagesize

  response = client.list_instances(request,runtime)
  return response
```

更改优先级

```
def modifyPriority(client,taskId,sort):
  request = ModifyPriorityRequest()
  request.instance_id =INSTANCE_ID
  request.task_id = taskId
  request.sort = sort

  response = client.modify_priority(request,runtime)
  return response
```