人脸识别服务

SDK 参考

文档版本	01
发布日期	2024-10-17

华为云计算技术有限公司

华为云计算技术有限公司

地址：	贵州省贵安新区黔中大道交兴功路华为云数据中心　　邮编：550029
网址：	https://www.huaweicloud.com/

目 录

1 简介..1
2 Java SDK..3
3 Python SDK.. 16
4 Go SDK... 25
5 .NET SDK..33
6 Node.js SDK..44
7 PHP SDK...50
8 C++ SDK.. 59
A 修订记录.. 70

1 简介

人脸识别服务软件开发工具包（FRS SDK）是对人脸识别服务提供的REST API进行的封装，以简化用户的开发工作。FRS SDK目前支持Java、Python、Go、.NET、Node.js、PHP、C++版本。

接口与 API 对应关系

人脸识别接口与API对应关系请参见表1-1。

表 1-1 接口与 API 对应关系表

接口		API
人脸检测		POST /v2/{project_id}/face-detect
人脸比对		POST /v2/{project_id}/face-compare
人脸搜索		POST /v2/{project_id}/face-sets/{face_set_name}/search
活体检测	动作活体检测	POST /v1/{project_id}/live-detect
活体检测	静默活体检测	POST /v1/{project_id}/live-detect-face
人脸库资源管理	创建人脸库	POST /v2/{project_id}/face-sets
	查询所有人脸库	GET /v2/{project_id}/face-sets
	查询人脸库	GET /v2/{project_id}/face-sets/{face_set_name}
	删除人脸库	DELETE /v2/{project_id}/face-sets/{face_set_name}
人脸资源管理	添加人脸	POST /v2/{project_id}/face-sets/{face_set_name}/faces
	查询人脸	GET /v2/{project_id}/face-sets/{face_set_name}/faces?offset=xxx&limit=xxx

接口		API
	更新人脸	PUT /v2/{project_id}/face-sets/{face_set_name}/faces
	删除人脸	DELETE /v2/{project_id}/face-sets/{face_set_name}/faces?field_name=field_value
	批量删除人脸	DELETE /v2/{project_id}/face-sets/{face_set_name}/faces/batch

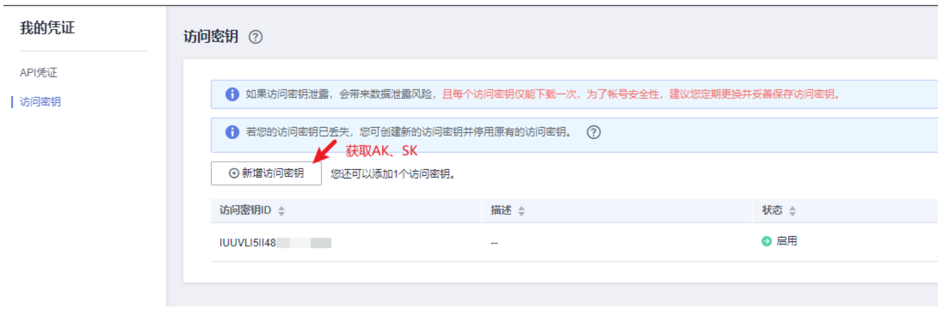
2 Java SDK

本章节介绍人脸识别服务Java SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已开通人脸识别服务。如未开通，请登录人脸识别管理控制台开通所需服务。
- 已具备开发环境，支持Java JDK 1.8 及其以上版本。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 2-1 获取 AK、SK



如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

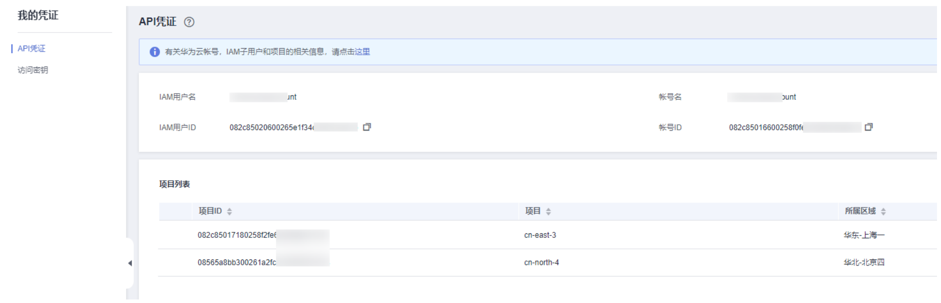
若您的访问密钥已丢失，您可创建新的访问密钥并停用原有的访问密钥。

您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5Il48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。
  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 2-2 我的凭证



有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名	[redacted]unt	帐号名	[redacted]ount
IAM用户ID	082c85020600265e1f34[redacted]	帐号ID	082c85016600258f0fe[redacted]

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

推荐您通过Maven方式获取和安装SDK，首先需要在您的操作系统中**下载**并**安装 Maven**，安装完成后您只需要在Java项目的pom.xml文件中加入相应的依赖项即可。

使用SDK前，需要安装“huaweicloud-sdk-core”和“huaweicloud-sdk-frs”依赖项。请在**SDK中心**获取最新的sdk包版本，替换代码中版本。

```
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-core</artifactId>
    <version>3.1.12</version>
</dependency>
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-frs</artifactId>
    <version>3.1.12</version>
</dependency>
```

说明

当出现第三方库冲突的时，如Jackson，okhttp3版本冲突等。可以引入如下bundle包(3.0.40-rc版本后)，该包包含所有支持的服务和重定向了SDK依赖的第三方软件，避免和业务自身依赖的库产生冲突：

```
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-bundle</artifactId>
    <version>[3.0.40-rc, 3.1.0)</version>
</dependency>
```

jackson版本要求请见**pom.xml**。SDK常见报错请参考**代码运行报错**、**json解析报错**。

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查SDK版本，并在**SDK中心**获取最新的SDK包版本。

```
mvn dependency:tree | grep huaweicloud-sdk-core
mvn dependency:tree | grep huaweicloud-sdk-frs
```

详细的SDK介绍，使用异步客户端，配置日志等操作请参见**SDK中心**、**Java SDK使用指导**、**Java SDK使用视频**。

1. 导入依赖模块

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

//导入v2版本sdk
import com.huaweicloud.sdk.frs.v2.region.FrsRegion;
import com.huaweicloud.sdk.frs.v2.*;
import com.huaweicloud.sdk.frs.v2.model.*;
```

2. 配置认证信息

   配置AK、SK信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK获取方法请参见**准备工作**。

```
public static ICredential getCredential(String ak, String sk) {
    return new BasicCredentials()
            .withAk(ak)
            .withSk(sk);
}
```

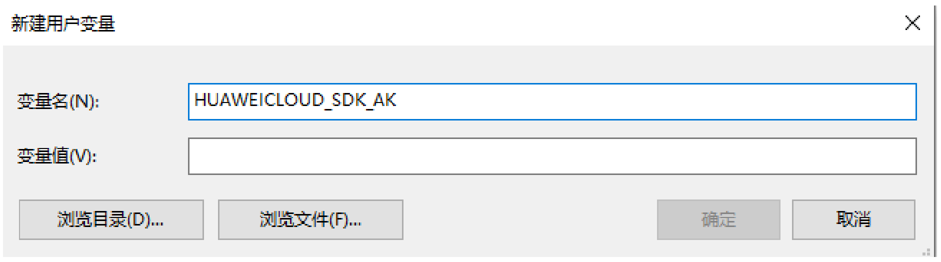
初始化认证信息：

```
String ak = System.getenv("HUAWEICLOUD_SDK_AK");
String sk = System.getenv("HUAWEICLOUD_SDK_SK");
ICredential credential = getCredential(ak, sk);
```

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。

图 2-3 Windows 环境新建环境变量



新建用户变量

变量名(N): HUAWEICLOUD_SDK_AK

变量值(V):

3. 初始化客户端

   指定region方式

```
public static FrsClient getClient(Region region, ICredential auth) {
    // 初始化人脸识别服务的客户端
    FrsClient client = FrsClient.newBuilder()
            .withCredential(auth)
            .withRegion(FrsRegion.valueOf("cn-north-4")) // 选择服务所在区域 FrsRegion.valueOf("cn-
north-4")
            .build();
    return client;
}
```

   服务部署区域请参见终端节点。

4. 发送请求并查看响应

```
// 以调用人脸检测 DetectFaceByBase64 接口为例
DetectFaceByBase64Request request = new DetectFaceByBase64Request();
FaceDetectBase64Req body = new FaceDetectBase64Req();
body.withImageBase64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
request.withBody(body);
DetectFaceByBase64Response response = client.detectFaceByBase64(request);
System.out.println(response.toString());
```

   说明

   使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

5. 异常处理

表 2-1 异常处理

一级分类	一级分类说明	二级分类	二级分类说明
ConnectionException	连接类异常	HostUnreachableException	网络不可达、被拒绝。

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

```
// 捕获和处理不同类型的异常
DetectFaceByBase64Request request = new DetectFaceByBase64Request();
try {
    DetectFaceByBase64Response response = client.detectFaceByBase64(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

SDK demo 代码解析

- 人脸检测

```
// detect face by base64
DetectFaceByBase64Request detectRequest = new DetectFaceByBase64Request();
FaceDetectBase64Req faceDetectBase64Req = new FaceDetectBase64Req();
faceDetectBase64Req.withImageBase64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
faceDetectBase64Req.withAttributes("2");
detectRequest.setBody(faceDetectBase64Req);
try {
    DetectFaceByBase64Response detectResponse = client.detectFaceByBase64(detectRequest);
    System.out.println(detectResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

// detect face by file
File file = new File("/root/picture.jpg");
DetectFaceByFileRequest byFileRequest = new DetectFaceByFileRequest();
```

```
DetectFaceByFileRequestBody requestBody = new DetectFaceByFileRequestBody();
try (InputStream inputStream = Files.newInputStream(file.toPath())){
    requestBody.withImageFile(inputStream, file.getName());
    byFileRequest.setBody(requestBody);
    DetectFaceByFileResponse response = client.detectFaceByFile(byFileRequest);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

- 人脸比对

```
// compare face by base64
CompareFaceByBase64Request compareRequest = new CompareFaceByBase64Request();
FaceCompareBase64Req faceCompareBase64Req = new FaceCompareBase64Req();
faceCompareBase64Req.withImage1Base64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
faceCompareBase64Req.withImage2Base64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
compareRequest.withBody(faceCompareBase64Req);
try {
    CompareFaceByBase64Response compareResponse =
client.compareFaceByBase64(compareRequest);
    System.out.println(compareResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

// compare face by file
File file1 = new File("/root/picture1.jpg");
File file2 = new File("/root/picture2.jpg");
try (InputStream inputStream1 = Files.newInputStream(file1.toPath());
    InputStream inputStream2 = Files.newInputStream(file2.toPath())) {
    CompareFaceByFileRequest request = new CompareFaceByFileRequest();
    CompareFaceByFileRequestBody requestBody = new CompareFaceByFileRequestBody();
    requestBody.withImage1File(inputStream1, file1.getName());
    requestBody.withImage2File(inputStream2, file2.getName());
    request.setBody(requestBody);
    CompareFaceByFileResponse response = client.compareFaceByFile(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

- 人脸搜索

```
// search face by base64
SearchFaceByBase64Request searchRequest = new SearchFaceByBase64Request();
searchRequest.withFaceSetName("face_set_name");
```

```
FaceSearchBase64Req faceSearchBase64Req = new FaceSearchBase64Req();
List<Map<String, String>> listbodySort = new ArrayList<>();
Map<String, String> map = new HashMap<>();
map.put("timestamp","asc");
listbodySort.add(map);
List<String> listbodyReturnFields = new ArrayList<>();
listbodyReturnFields.add("timestamp");
listbodyReturnFields.add("id");
faceSearchBase64Req.withSort(listbodySort);
faceSearchBase64Req.withReturnFields(listbodyReturnFields);
faceSearchBase64Req.withImageBase64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
searchRequest.withBody(faceSearchBase64Req);
try {
    SearchFaceByBase64Response searchResponse = client.searchFaceByBase64(searchRequest);
    System.out.println(searchResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

// search face by file
SearchFaceByFileRequest byFileRequest = new SearchFaceByFileRequest();
byFileRequest.withFaceSetName("face_set_name");
File file = new File("/root/picture.jpg");
try (InputStream inputStream = Files.newInputStream(file.toPath())) {
    SearchFaceByFileRequestBody fileRequestBody = new SearchFaceByFileRequestBody();
    fileRequestBody.withImageFile(inputStream, file.getName());
    byFileRequest.withBody(fileRequestBody);
    fileRequestBody.withSort("[{ \"timestamp\": \"desc\"}]");
    fileRequestBody.withReturnFields("[\"timestamp\"]");
    SearchFaceByFileResponse fileResponse = client.searchFaceByFile(byFileRequest);
    System.out.println(fileResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

- 创建人脸库

```
CreateFaceSetRequest createFaceSetRequest = new CreateFaceSetRequest();
CreateFaceSetReq createFaceSetReq = new CreateFaceSetReq();
createFaceSetReq.withFaceSetName("face_set_name");
Map<String, TypeInfo> stringTypeInfoMap = new HashMap<>();
TypeInfo typeInfo = new TypeInfo();
typeInfo.withType("long");
stringTypeInfoMap.put("timestamp", typeInfo);
createFaceSetReq.withExternalFields(stringTypeInfoMap);
createFaceSetRequest.withBody(createFaceSetReq);
try {
    CreateFaceSetResponse createFaceSetResponse = client.createFaceSet(createFaceSetRequest);
    System.out.println(createFaceSetResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 查询人脸库

```
ShowFaceSetRequest showFaceSetRequest = new ShowFaceSetRequest();
showFaceSetRequest.withFaceSetName("face_set_name");
try {
    ShowFaceSetResponse showFaceSetResponse = client.showFaceSet(showFaceSetRequest);
    System.out.println(showFaceSetResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 查询所有人脸库

```
ShowAllFaceSetsRequest showAllFaceSetsRequest = new ShowAllFaceSetsRequest();
try {
    ShowAllFaceSetsResponse showAllFaceSetsResponse = 
client.showAllFaceSets(showAllFaceSetsRequest);
    System.out.println(showAllFaceSetsResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 删除人脸库

```
DeleteFaceSetRequest deleteFaceSetRequest = new DeleteFaceSetRequest();
deleteFaceSetRequest.withFaceSetName("face_set_name");
try {
    DeleteFaceSetResponse deleteFaceSetResponse = client.deleteFaceSet(deleteFaceSetRequest);
    System.out.println(deleteFaceSetResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 添加人脸

```
//add face by base64
AddFacesByBase64Request addFacesByBase64Request = new AddFacesByBase64Request();
addFacesByBase64Request.withFaceSetName("face_set_name");
AddFacesBase64Req addFacesBase64Req = new AddFacesBase64Req();
addFacesBase64Req.withExternalFields("{\"timestamp\":12}");
addFacesBase64Req.withImageBase64("9j/4AAQSkZJRgABAQAAAQABAAD...");
addFacesByBase64Request.withBody(addFacesBase64Req);
try {
    AddFacesByBase64Response addFacesByBase64Response = 
client.addFacesByBase64(addFacesByBase64Request);
    System.out.println(addFacesByBase64Response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

//add face by file
AddFacesByFileRequest fileRequest = new AddFacesByFileRequest();
fileRequest.withFaceSetName("face_set_name");
AddFacesByFileRequestBody fileRequestBody = new AddFacesByFileRequestBody();
File file = new File("/root/picture.jpg");
try (InputStream inputStream = Files.newInputStream(file.toPath())) {
    fileRequestBody.withImageFile(inputStream, file.getName());
    fileRequestBody.setExternalFields("{\"id\": \"zhangsan\"}");
    fileRequest.withBody(fileRequestBody);
    AddFacesByFileResponse fileResponse = client.addFacesByFile(fileRequest);
    System.out.println(fileResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

- 删除人脸

```
//delete face by faceId
DeleteFaceByFaceIdRequest deleteFaceByFaceIdRequest = new DeleteFaceByFaceIdRequest();
deleteFaceByFaceIdRequest.withFaceSetName("face_set_name");
deleteFaceByFaceIdRequest.withFaceId("iexEBb6t");
try {
    DeleteFaceByFaceIdResponse deleteFaceByFaceIdResponse =
client.deleteFaceByFaceId(deleteFaceByFaceIdRequest);
    System.out.println(deleteFaceByFaceIdResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

//delete face by externalImageId
DeleteFaceByExternalImageIdRequest deleteFaceRequest = new
DeleteFaceByExternalImageIdRequest();
deleteFaceRequest.withFaceSetName("face_set_name");
deleteFaceRequest.withExternalImageId("iexEBb6t");
try {
    DeleteFaceByExternalImageIdResponse response =
client.deleteFaceByExternalImageId(deleteFaceRequest);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 批量删除人脸

```
BatchDeleteFacesRequest batchDeleteFacesRequest = new BatchDeleteFacesRequest();
batchDeleteFacesRequest.withFaceSetName("face_set_name");
DeleteFacesBatchReq deleteFacesBatchReq = new DeleteFacesBatchReq();
deleteFacesBatchReq.withFilter("age:[20 TO 30]");
batchDeleteFacesRequest.withBody(deleteFacesBatchReq);
try {
    BatchDeleteFacesResponse batchDeleteFacesResponse = 
client.batchDeleteFaces(batchDeleteFacesRequest);
    System.out.println(batchDeleteFacesResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

示例中filter参数为string类型，用于过滤接口返回的结果信息，例如 age:[20 TO 30] 表示只返回20至30岁之间的人脸信息。

- 更新人脸

```
UpdateFaceRequest updateFaceRequest = new UpdateFaceRequest();
updateFaceRequest.withFaceSetName("face_set_name");
UpdateFaceReq updateFaceReq = new UpdateFaceReq();
updateFaceReq.withFaceId("iexEBb6t");
updateFaceRequest.withBody(updateFaceReq);
try {
    UpdateFaceResponse response = client.updateFace(updateFaceRequest);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 查询人脸

```
//show faces by faceId
ShowFacesByFaceIdRequest showFacesByFaceIdRequest = new ShowFacesByFaceIdRequest();
showFacesByFaceIdRequest.withFaceSetName("face_set_name");
showFacesByFaceIdRequest.withFaceId("iexEBb6t");
try {
    ShowFacesByFaceIdResponse response = client.showFacesByFaceId(showFacesByFaceIdRequest);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

//show faces by limit
ShowFacesByLimitRequest showFacesByLimitRequest = new ShowFacesByLimitRequest();
showFacesByLimitRequest.withFaceSetName("face_set_name");
showFacesByLimitRequest.withOffset(0);
showFacesByLimitRequest.withLimit(10);
```

```
try {
    ShowFacesByLimitResponse showFacesByLimitResponse =
client.showFacesByLimit(showFacesByLimitRequest);
    System.out.println(showFacesByLimitResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 动作活体检测

```
//detect live by base64
DetectLiveByBase64Request request = new DetectLiveByBase64Request();
LiveDetectBase64Req body = new LiveDetectBase64Req();
body.withActions("1,2,3,4");
body.withVideoBase64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
request.withBody(body);
try {
    DetectLiveByBase64Response response = client.detectLiveByBase64(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
//detect live by file
File file = new File("/root/video.mp4");
try (InputStream inputStream = Files.newInputStream(file.toPath())) {
    DetectLiveByFileRequest fileRequest = new DetectLiveByFileRequest();
    DetectLiveByFileRequestBody fileRequestBody = new DetectLiveByFileRequestBody();
    fileRequestBody.setActions("2");
    fileRequestBody.withVideoFile(inputStream, file.getName());
    fileRequest.withBody(fileRequestBody);
    DetectLiveByFileResponse byFileResponse = client.detectLiveByFile(fileRequest);
    System.out.println(byFileResponse.toString());
} catch (ConnectionException e) {
    System.out.println(e.toString());
} catch (RequestTimeoutException e) {
    System.out.println(e.toString());
} catch (ServiceResponseException e) {
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

- 静默活体检测

```
//detect live face by base64
DetectLiveFaceByBase64Request request = new DetectLiveFaceByBase64Request();
LiveDetectFaceBase64Req body = new LiveDetectFaceBase64Req();
body.withImageBase64("/9j/4AAQSkZJRgABAQAAAQABAAD...");
request.withBody(body);
try {
    DetectLiveFaceByBase64Response response = client.detectLiveFaceByBase64(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}

//detect live face by file
DetectLiveFaceByFileRequest fileRequest = new DetectLiveFaceByFileRequest();
DetectLiveFaceByFileRequestBody fileRequestBody = new DetectLiveFaceByFileRequestBody();
File file = new File("/root/picture.jpg");
try (InputStream inputStream = Files.newInputStream(file.toPath())) {
    fileRequestBody.withImageFile(inputStream, file.getName());
    fileRequest.withBody(fileRequestBody);
    DetectLiveFaceByFileResponse fileResponse = client.detectLiveFaceByFile(fileRequest);
    System.out.println(fileResponse.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getHttpStatusCode());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
} catch (IOException e) {
    e.printStackTrace();
}
```

OkHttp 示例

如果使用OkHttp处理HTTP请求和响应，可参考以下代码（以人脸检测为例，其它接口示例可通过**API Explorer**获取）。

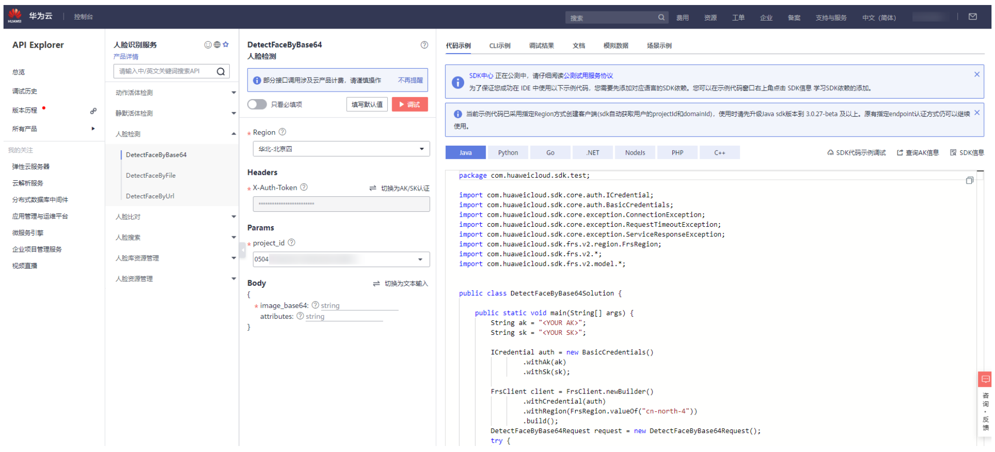
```
OkHttpClient client = new OkHttpClient().newBuilder()
    .build();
MediaType mediaType = MediaType.parse("text/plain");
RequestBody body = RequestBody.create(mediaType, "");
Request request = new Request.Builder()
    .url("https://face.cn-north-4.myhuaweicloud.com/v2/{project_id}/face-detect")
    .method("POST", body)
    .addHeader("Authorization", "<Your signed string>")
    .build();
Response response = client.newCall(request).execute();
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。同时，可在集成开发环境CloudIDE中完成代码的构建、调试和运行等操作。

图 2-4 API Explorer



DetectFaceByBase64
人脸检测

部分接口调用涉及云产品计费，请谨慎操作 不再提醒

只看必填项 填写默认值 调试

* Region：华北-北京四

Headers

* X-Auth-Token ⇄ 切换为AK/SK认证
****

Params

* project_id：0504[redacted]

Body ⇄ 切换为文本输入

```
{
  * image_base64: string
    attributes: string
}
```

SDK中心 正在公测中，请仔细阅读公测试用服务协议
为了保证您成功在 IDE 中使用以下示例代码，您需要先添加对应语言的SDK依赖。您可以在示例代码窗口右上角点击 SDK信息 学习SDK依赖的添加。

当前示例代码已采用指定Region方式创建客户端(sdk自动获取用户的projectId和domainId)，使用时请先升级Java sdk版本到 3.0.27-beta 及以上。原有指定endpoint认证方式仍可以继续使用。

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

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.frs.v2.region.FrsRegion;
import com.huaweicloud.sdk.frs.v2.*;
import com.huaweicloud.sdk.frs.v2.model.*;


public class DetectFaceByBase64Solution {

    public static void main(String[] args) {
        String ak = "<YOUR AK>";
        String sk = "<YOUR SK>";

        ICredential auth = new BasicCredentials()
                .withAk(ak)
                .withSk(sk);

        FrsClient client = FrsClient.newBuilder()
                .withCredential(auth)
                .withRegion(FrsRegion.valueOf("cn-north-4"))
                .build();
        DetectFaceByBase64Request request = new DetectFaceByBase64Request();
        try {
```

代码运行报错

- **java.lang.NoClassDefFoundError: Could not initialize class com.huaweicloud.sdk.core.http.HttpConfig at com.huaweicloud.sdk.core.ClientBuilder.build(ClientBuilder.java:98)**

  HttpConfig 这个类在sdk-core 包里面找不到，造成原因为用户使用的sdk版本太老导致，建议使用**最新版本**的华为云java sdk，运行代码再具体定位。

- **java.lang.NoSuchFieldError: ALLOW_LEADING_DECIMAL_POINT_FOR_NUMBERS**

  这个字段是 jackson-core 里面用来标识解析json格式数据是否支持前导小数点的字段，这个报错的意思是找不到这个字段，很可能是因为用户使用的jackson 版本太老导致。

  建议客户本地将jackson 版本升级到和华为云 java sdk一致，jackson版本要求请见**pom.xml**。

  引用华为云java sdk的**bundle包**来解决 jackson 版本冲突的问题。

  ```
  <dependency>
      <groupId>com.huaweicloud.sdk</groupId>
      <artifactId>huaweicloud-sdk-bundle</artifactId>
      <version>[3.0.40-rc, 3.1.0)</version>
  </dependency>
  ```

- **java.lang.ClassNotFoundException: com.fasterxml.jackson.datatype.jsr310.JavaTimeModule**

  用户本地工程引入了jackson 框架，和 华为云sdk引入的jackson 框架冲突了，导致会报找不到某个类，建议 客户在本地引入**bundle包**报来避免出现依赖冲突。

  ```
  <dependency>
      <groupId>com.huaweicloud.sdk</groupId>
      <artifactId>huaweicloud-sdk-bundle</artifactId>
      <version>[3.0.40-rc, 3.1.0)</version>
  </dependency>
  ```

- **java.lang.ClassNotFoundException: okhttp3/Interceptor**

  用户本地引入的Okhttp3 版本和 华为云冲突，okhttp版本要求请见**pom.xml**。

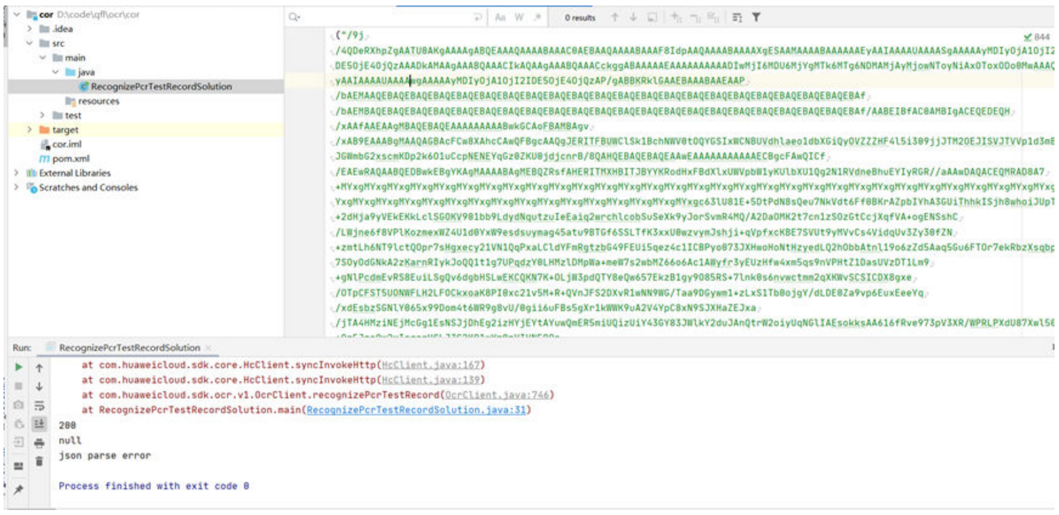
- **INFO com.huaweicloud.sdk.core.HcClient - project id of region 'cn-north-4' not found in BasicCredentials, trying to obtain project id from IAM service: https://iam.myhuaweicloud.com**

  调用服务对应终端节点下的项目ID没有生成。

在“**我的凭证**”页面中查看对应终端节点的项目ID，确认系统中没有生成。在**FRS控制台**将终端节点切换至调用服务所在的终端节点，之后前往“我的凭证”页面，即可查看到已生成对应的项目ID。

json 解析报错

图 2-5 json parse error



```
    at com.huaweicloud.sdk.core.HcClient.syncInvokeHttp(HcClient.java:167)
    at com.huaweicloud.sdk.core.HcClient.syncInvokeHttp(HcClient.java:139)
    at com.huaweicloud.sdk.ocr.v1.OcrClient.recognizePcrTestRecord(OcrClient.java:746)
    at RecognizePcrTestRecordSolution.main(RecognizePcrTestRecordSolution.java:31)
200
null
json parse error

Process finished with exit code 0
```

- 服务端返回的数据格式不符合json格式，导致sdk侧解析json数据报错。
- 服务端返回的json 数据 不符合json反序列化的规则，和sdk定义的数据结构不一致，导致反序列化失败。
- sdk json 数据解析问题。

建议排查服务端返回的数据是否和服务SDK设计的结构、字段一致。

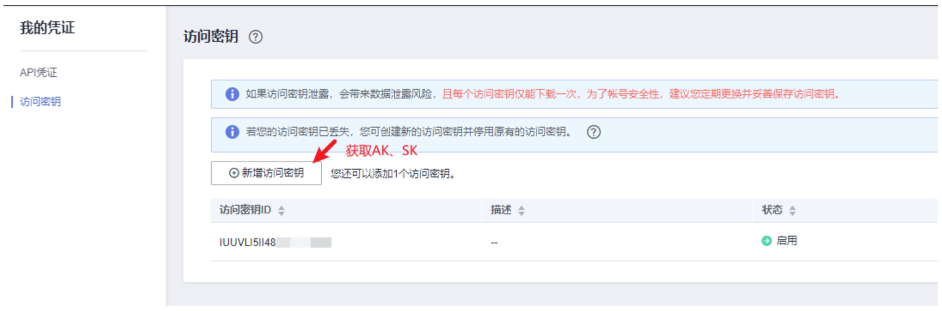
3 Python SDK

本章节介绍人脸识别服务Python SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已开通人脸识别服务。如未开通，请登录人脸识别管理控制台人脸识别管理控制台开通所需服务。
- 已具备开发环境，支持Python3及以上版本。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 3-1 获取 AK、SK



如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

若您的访问密钥已丢失，您可创建新的访问密钥并停用原有的访问密钥。

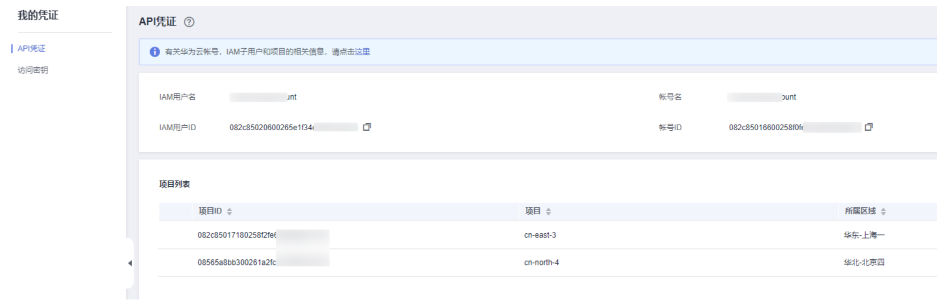
您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5II48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 3-2 我的凭证



有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

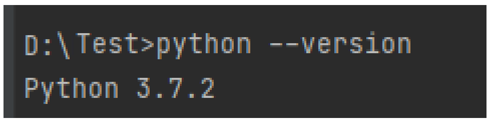
IAM用户名	[redacted]unt	帐号名	[redacted]ount
IAM用户ID	082c85020600265e1f34[redacted]	帐号ID	082c85016600258f0fe[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

支持Python3及以上版本，执行**python --version**检查当前Python的版本信息。



```
D:\Test>python --version
Python 3.7.2
```

使用SDK前，需要安装“huaweicloudsdkcore”和“huaweicloudsdkfrs”。

```
# 安装核心库
pip install huaweicloudsdkcore
# 安装FRS服务库
pip install huaweicloudsdkfrs
```

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查并更新SDK至最新版本。

```
pip show huaweicloudsdkcore
pip show huaweicloudsdkfrs
pip install --upgrade huaweicloudsdkcore
pip install --upgrade huaweicloudsdkfrs
```

详细的SDK介绍，使用异步客户端，配置日志等操作请参见**SDK中心**、**Python SDK使用指导**、**Python SDK使用视频**。

1. 导入依赖模块

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.exceptions import exceptions

# 导入v2版本sdk
from huaweicloudsdkfrs.v2.region.frs_region import FrsRegion
from huaweicloudsdkfrs.v2 import *

import os
```

2. 配置认证信息

   配置AK、SK信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK获取方法请参见**准备工作**。

```
def GetCredential(ak, sk):
    return BasicCredentials(ak, sk)
```

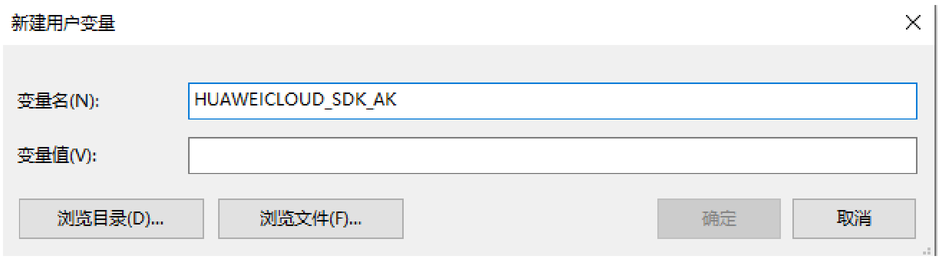
   初始化认证信息：

```
ak = os.environ.get("HUAWEICLOUD_SDK_AK")
sk = os.environ.get("HUAWEICLOUD_SDK_SK")
credentials = GetCredential(ak, sk)
```

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。

图 3-3 Windows 环境新建环境变量



3. 初始化客户端

   指定region方式

```
# 初始化人脸识别服务的客户端，并选择服务部署区域
def GetClient():
    client =  FrsClient.new_builder(FrsClient) \
        .with_credentials(credentials) \
        .with_region(FrsRegion.value_of("cn-north-4")) \
        .build()
    return client
```

   服务部署区域请参见终端节点。

4. 发送请求并查看响应

```
# 以调用人脸检测 DetectFaceByBase64 接口为例
request = DetectFaceByBase64Request()
request.body = FaceDetectBase64Req(
    image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
)
response = client.detect_face_by_base64(request)
print(response)
```

   说明

   使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

5. 异常处理

表 3-1 异常处理

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

```
// 捕获和处理不同类型的异常
try:
    request = DetectFaceByBase64Request()
    request.body = FaceDetectBase64Req(
        image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
    )
    response = client.detect_face_by_base64(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

SDK 代码解析

- 人脸检测

```
# detect face by base64
def detectFaceByBase64():
    try:
        request = DetectFaceByBase64Request()
        request.body = FaceDetectBase64Req(
            image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD...",
            attributes="2,4"
        )
        response = client.detect_face_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

# detect face by file
def detectFaceByFile():
    try:
        request = DetectFaceByFileRequest()
        with open("/root/picture.jpg", "rb") as f:
            request.body = DetectFaceByFileRequestBody(image_file=FormFile(f))
            response = client.detect_face_by_file(request)
            print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 人脸比对

```
# compare face by base64
def compareFaceByBase64():
    try:
        request = CompareFaceByBase64Request()
        request.body = FaceCompareBase64Req(
            image1_base64="/9j/4AAQSkZJRgABAQAAAQABAAD...",
            image2_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
        )
        response = client.compare_face_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

# compare face by file
def compareFaceByFile():
    try:
        request = CompareFaceByFileRequest()
        with open("/root/picture1.jpg", "rb") as f1:
            with open("/root/picture2.jpg", "rb") as f2:
                request.body = CompareFaceByFileRequestBody(image1_file=FormFile(f1),
                                                            image2_file=FormFile(f2))
                response = client.compare_face_by_file(request)
                print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 人脸搜索

```
# search face by base64
def searchFaceByBase64():
    try:
        request = SearchFaceByBase64Request()
        request.face_set_name = "face_set_name"
        listFaceSearchBase64ReqReturnFieldsbody = [
            "timestamp"
        ]
        request.body = FaceSearchBase64Req(
            return_fields=listFaceSearchBase64ReqReturnFieldsbody,
            image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
        )
        response = client.search_face_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

# search face by file
def searchFaceByFile():
    try:
        request = SearchFaceByFileRequest()
        request.face_set_name = "face_set_name"
        with open("/root/picture.jpg", "rb") as f:
                request.body = SearchFaceByFileRequestBody(
                return_fields="[\"timestamp\"]",
                filter="timestamp:10",
                top_n=10,
                image_file=FormFile(f)
            )
            response = client.search_face_by_file(request)
            print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建人脸库

```
def createFaceSet():
    try:
        request = CreateFaceSetRequest()
        request.body = CreateFaceSetReq(
            face_set_name="face_set_name",
            external_fields={"timestamp": {"type": "long"}}
        )
        response = client.create_face_set(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 查询人脸库

```
def showFaceSet():
    try:
        request = ShowFaceSetRequest()
        request.face_set_name = "face_set_name"
        response = client.show_face_set(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 查询所有人脸库

```
def showAllFaceSet():
    try:
        request = ShowAllFaceSetsRequest()
        response = client.show_all_face_sets(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 删除人脸库

```
def deleteFaceSet():
    try:
        request = DeleteFaceSetRequest()
        request.face_set_name = "face_set_name"
        response = client.delete_face_set(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 添加人脸

```
# add face by base64
def addFacesByBase64():
    try:
        request = AddFacesByBase64Request()
        request.face_set_name = "face_set_name"
        request.body = AddFacesBase64Req(
            external_fields="{\"timestamp\":12}",
            image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
        )
        response = client.add_faces_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

# add face by file
def addFacesByFile():
    try:
        request = AddFacesByFileRequest()
        request.face_set_name = "face_set_name"
        with open("/root/picture.jpg", "rb") as f:
            request.body = AddFacesByFileRequestBody(
                external_fields="{\"timestamp\":12}",
                image_file=FormFile(f)
            )
            response = client.add_faces_by_file(request)
            print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 删除人脸

```
def deleteFace():
    # Delete Face By FaceId
    try:
        request = DeleteFaceByFaceIdRequest()
        request.face_set_name = "face_set_name"
        request.face_id = "LkPJblq6"
        response = client.delete_face_by_face_id(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

    # Delete Face By ExternalImageId
    try:
        request = DeleteFaceByExternalImageIdRequest()
        request.face_set_name = "face_set_name"
        request.external_image_id = "external_image_id"
        response = client.delete_face_by_external_image_id(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 批量删除人脸

```
def batchDeleteFaces():
    try:
        request = BatchDeleteFacesRequest()
        request.face_set_name = "face_set_name"
        request.body = DeleteFacesBatchReq(
            filter="age:[20 TO 30]"
        )
        response = client.batch_delete_faces(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 更新人脸

```
def updateFace():
    try:
        request = UpdateFaceRequest()
        request.face_set_name = "face_set_name"
        request.body = UpdateFaceReq(face_id="LkPJblq6")
        response = client.update_face(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 查询人脸

```
def showFaces():
    # Show Faces By FaceId
    try:
        request = ShowFacesByFaceIdRequest()
        request.face_set_name = "face_set_name"
        request.face_id = "LkPJblq6"
        response = client.show_faces_by_face_id(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

    # Show Faces By Limit
    try:
        request = ShowFacesByLimitRequest()
        request.face_set_name = "face_set_name"
        request.offset = 0
        request.limit = 10
        response = client.show_faces_by_limit(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 动作活体检测

```
# detect live by base64
def detectLiveByBase64():
    try:
        request = DetectLiveByBase64Request()
        request.body = LiveDetectBase64Req(
            actions="1,2,3,4",
            video_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
        )
        response = client.detect_live_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

# detect live by file
def detectLiveByFile():
    try:
        request = DetectLiveByFileRequest()
        with open("/root/video.mp4", "rb") as f:
            request.body = DetectLiveByFileRequestBody(
                video_file=FormFile(f),
                actions="1,2,3,4"
            )
            response = client.detect_live_by_file(request)
            print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 静默活体检测

```
# detect live face by base64
def detectLiveFaceByBase64():
    try:
        request = DetectLiveFaceByBase64Request()
        request.body = LiveDetectFaceBase64Req(
            image_base64="/9j/4AAQSkZJRgABAQAAAQABAAD..."
        )
        response = client.detect_live_face_by_base64(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

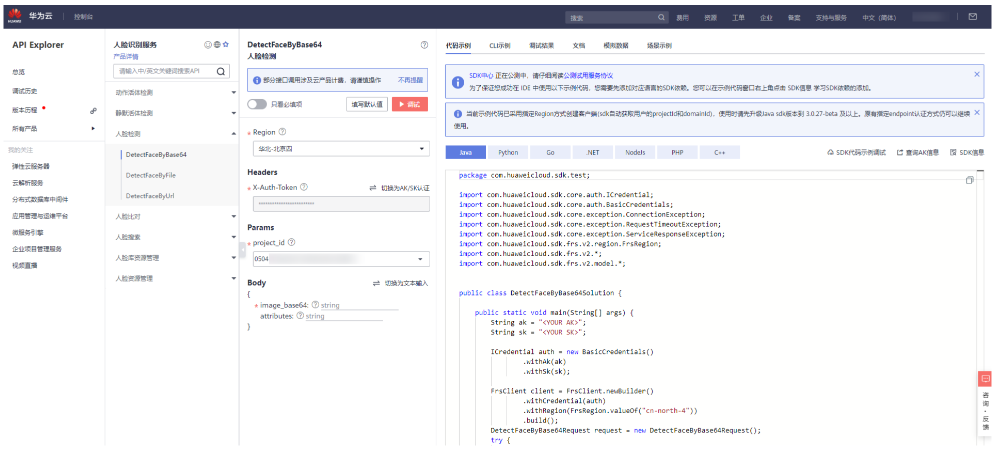
```
# detect live face by file
def detectLiveFaceByFile():
    try:
        request = DetectLiveFaceByFileRequest()
        with open("/root/picture.jpg", "rb") as f:
            request.body = DetectLiveFaceByFileRequestBody(
                image_file=FormFile(f)
            )
            response = client.detect_live_face_by_file(request)
            print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。同时，可在集成开发环境CloudIDE中完成代码的构建、调试和运行等操作。

图 3-4 API Explorer



部分接口调用涉及云产品计费，请谨慎操作

Region: 华北-北京四

Headers

X-Auth-Token: ************************

Params

project_id: 0504[redacted]

Body
```
{
  image_base64: string
  attributes: string
}
```

SDK中心 正在公测中，请仔细阅读公测试用服务协议
为了保证您成功在 IDE 中使用以下示例代码，您需要先添加对应语言的SDK依赖。您可以在示例代码窗口右上角点击 SDK信息 学习SDK依赖的添加。

当前示例代码已采用指定Region方式创建客户端(sdk自动获取用户的projectId和domainId)，使用时请先升级Java sdk版本到 3.0.27-beta 及以上。原有指定endpoint认证方式仍可以继续使用。

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

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.frs.v2.region.FrsRegion;
import com.huaweicloud.sdk.frs.v2.*;
import com.huaweicloud.sdk.frs.v2.model.*;


public class DetectFaceByBase64Solution {

    public static void main(String[] args) {
        String ak = "<YOUR AK>";
        String sk = "<YOUR SK>";

        ICredential auth = new BasicCredentials()
                .withAk(ak)
                .withSk(sk);

        FrsClient client = FrsClient.newBuilder()
                .withCredential(auth)
                .withRegion(FrsRegion.valueOf("cn-north-4"))
                .build();
        DetectFaceByBase64Request request = new DetectFaceByBase64Request();
        try {
```

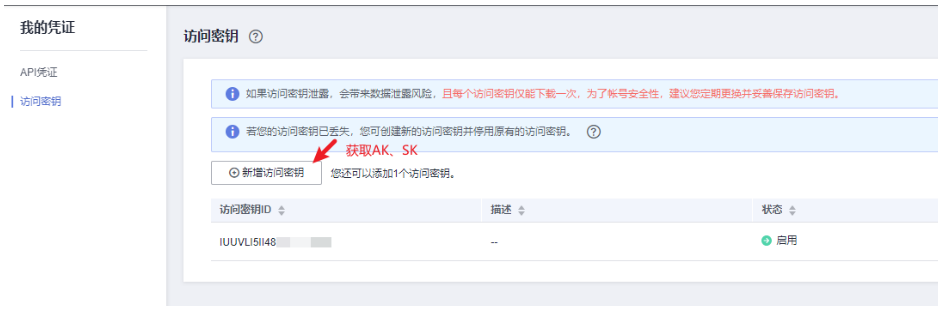
4 Go SDK

本章节介绍人脸识别服务Go SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已开通人脸识别服务。如未开通，请登录人脸识别管理控制台人脸识别管理控制台开通所需服务。
- 已具备开发环境，Go SDK 支持 go 1.14 及以上版本，可执行 go version 检查当前 Go 的版本信息。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

  图 4-1 获取 AK、SK

  
  我的凭证
  API凭证
  访问密钥

  访问密钥

  如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

  若您的访问密钥已丢失，您可创建新的访问密钥并停用原有的访问密钥。

  获取AK、SK

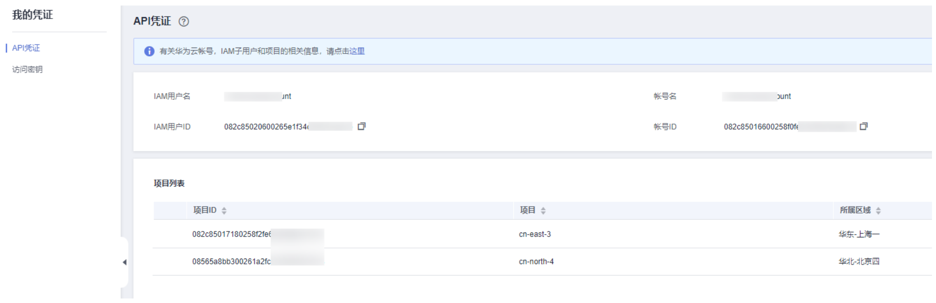
  新增访问密钥　您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5II48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 4-2 我的凭证



我的凭证

API凭证

访问密钥

API凭证

有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名 [redacted]unt

帐号名 [redacted]ount

IAM用户ID 082c85020600265e1f34[redacted]

帐号ID 082c85016600258f0fe[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

使用SDK前需要安装华为云Go SDK 库。

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

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查并更新SDK至最新版本。

```
go list -m all | grep huaweicloud-sdk-go-v3
go get -u github.com/huaweicloud/huaweicloud-sdk-go-v3
```

详细的SDK介绍请参见**SDK中心**、**Go SDK使用指导**、**Go SDK使用视频**。

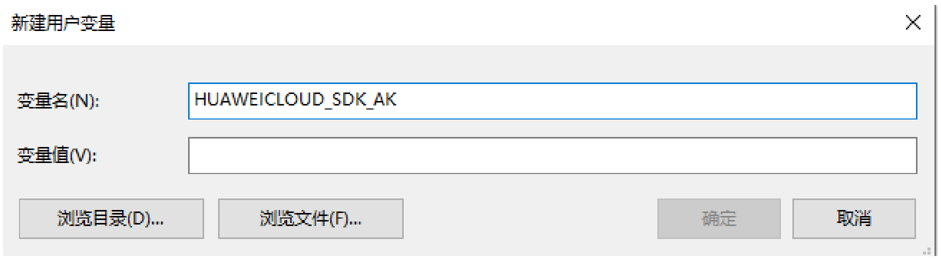
1. 导入依赖模块
```
import (
    "fmt"
        "os"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"

    // 导入v2版本sdk
    frs "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2/region"
)
```
2. 配置认证信息

   配置AK、SK信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK获取方法请参见**准备工作**。
```
func GetCredential(ak, sk string) basic.Credentials {
    // Init Auth Info
    return basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        Build()
}
```
   初始化认证信息：
```
ak := os.Getenv("HUAWEICLOUD_SDK_AK")
sk := os.Getenv("HUAWEICLOUD_SDK_SK")
client := GetCredential(ak, sk)
```

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。

图 4-3 Windows 环境新建环境变量


新建用户变量

变量名(N): HUAWEICLOUD_SDK_AK

变量值(V):

3. 初始化客户端

   指定region方式

```
// # 初始化人脸识别服务的客户端，并选择服务部署区域
func GetClient(auth basic.Credentials) *frs.FrsClient {
    client := frs.NewFrsClient(
            frs.FrsClientBuilder().
            WithRegion(region.ValueOf('cn-north-4')).
            WithCredential(auth).
            Build())
        return client
}
```

   服务部署区域请参见**终端节点**。

4. 发送请求并查看下响应

```
request := &model.DetectFaceByBase64Request{}
request.Body = &model.FaceDetectBase64Req{
    ImageBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.DetectFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

   说明

   使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

5. 异常处理

表 4-1 异常处理

一级分类	一级分类说明
ServiceResponseError	服务响应异常
url.Error	url异常

```
// 捕获和处理不同类型的异常
response, err := client.DetectFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

SDK demo 代码解析

- 人脸检测

```
// detect face by base64
request := &model.DetectFaceByBase64Request{}
attributesFaceDetectBase64Req := "2"
request.Body = &model.FaceDetectBase64Req{
    Attributes:  &attributesFaceDetectBase64Req,
    ImageBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.DetectFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

// detect face by file
file, err := os.Open("/root/picture.jpg")
if err != nil {
   fmt.Println(err)
}
defer file.Close()
request := &model.DetectFaceByFileRequest{}
request.Body = &model.DetectFaceByFileRequestBody{
   Attributes: def.NewMultiPart(attributes),
   ImageFile:  def.NewFilePart(file),
}
response, err := client.DetectFaceByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

- 人脸比对

```
// compare face by base64
request := &model.CompareFaceByBase64Request{}
request.Body = &model.FaceCompareBase64Req{
    Image1Base64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
    Image2Base64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.CompareFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

// compare face by file
file1, err := os.Open("/root/picture1.jpg")
if err != nil {
   fmt.Println(err)
}
defer file1.Close()
file2, err := os.Open("/root/picture2.jpg")
if err != nil {
   fmt.Println(err)
}
defer file2.Close()
request := &model.CompareFaceByFileRequest{}
```

```
request.Body = &model.CompareFaceByFileRequestBody{
   Image1File: def.NewFilePart(file1),
   Image2File: def.NewFilePart(file2),
}
response, err := client.CompareFaceByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

- 人脸搜索

```
// face search by base64
request := &model.SearchFaceByBase64Request{}
request.FaceSetName = "face_set_name"
var listReturnFieldsbody = []string{
    "timestamp",
    "id",
}
request.Body = &model.FaceSearchBase64Req{
    ReturnFields: &listReturnFieldsbody,
    ImageBase64:  "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.SearchFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

// face search by file
file, err := os.Open("/root/picture.jpg")
if err != nil {
   fmt.Println(err)
}
defer file.Close()
request := &model.SearchFaceByFileRequest{}
request.FaceSetName = "face_set_name"
request.Body = &model.SearchFaceByFileRequestBody{
   ReturnFields: def.NewMultiPart("[\"timestamp\"]"),
   Filter:       def.NewMultiPart("timestamp:10"),
   ImageFile:    def.NewFilePart(file),
}
response, err := client.SearchFaceByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

- 创建人脸库

```
request := &model.CreateFaceSetRequest{}
request.Body = &model.CreateFaceSetReq{
    FaceSetName: "face_set_name",
}
response, err := client.CreateFaceSet(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 查询人脸库

```
request := &model.ShowFaceSetRequest{}
request.FaceSetName = "face_set_name"
response, err := client.ShowFaceSet(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 查询所有人脸库

```
request := &model.ShowAllFaceSetsRequest{}
response, err := client.ShowAllFaceSets(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 删除人脸库

```
request := &model.DeleteFaceSetRequest{}
request.FaceSetName = "face_set_name"
response, err := client.DeleteFaceSet(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 添加人脸

```
// add face by base64
request := &model.AddFacesByBase64Request{}
request.FaceSetName = "face_set_name"
var externalFieldsAddFacesBase64Req interface{} = "{\"timestamp\":12}"
request.Body = &model.AddFacesBase64Req{
    ExternalFields: &externalFieldsAddFacesBase64Req,
    ImageBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.AddFacesByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

// add face by file
file, err := os.Open("/root/picture.jpg")
if err != nil {
   fmt.Println(err)
}
defer file.Close()
request := &model.AddFacesByFileRequest{}
request.FaceSetName = "face_set_name"
request.Body = &model.AddFacesByFileRequestBody{
   ExternalFields: def.NewMultiPart("{\"timestamp\":100}"),
   ImageFile:      def.NewFilePart(file),
}
response, err := client.AddFacesByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

- 删除人脸

```
request := &model.DeleteFaceByExternalImageIdRequest{}
request.FaceSetName = "face_set_name"
request.ExternalImageId = "external_image_id"
response, err := client.DeleteFaceByExternalImageId(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 批量删除人脸

```
request := &model.BatchDeleteFacesRequest{}
request.FaceSetName = "face_set_name"
request.Body = &model.DeleteFacesBatchReq{
    Filter: "age:[20 TO 30]",
}
response, err := client.BatchDeleteFaces(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 更新人脸

```
request := &model.UpdateFaceRequest{}
request.FaceSetName = "face_set_name"
externalImageIdUpdateFaceReq:= "external_image_id"
var externalFieldsUpdateFaceReq interface{} = "{\"timestamp\":12}"
request.Body = &model.UpdateFaceReq{
    FaceId: "LkPJblq6",
    ExternalImageId: &externalImageIdUpdateFaceReq,
    ExternalFields: &externalFieldsUpdateFaceReq,
}
response, err := client.UpdateFace(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 查询人脸

```
//Show Faces By FaceId
request := &model.ShowFacesByFaceIdRequest{}
request.FaceSetName = "face_set_name"
request.FaceId = "LkPJblq6"
response, err := client.ShowFacesByFaceId(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 动作活体检测

```
//detect live by base64
request := &model.DetectLiveByBase64Request{}
request.Body = &model.LiveDetectBase64Req{
    Actions: "1,2,3,4",
    VideoBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.DetectLiveByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

//detect live by file
file, err := os.Open("/root/video.mp4")
if err != nil {
   fmt.Println(err)
}
defer file.Close()
request := &model.DetectLiveByFileRequest{}
request.Body = &model.DetectLiveByFileRequestBody{
   Actions:   def.NewMultiPart(action),
   VideoFile: def.NewFilePart(file),
}
response, err := client.DetectLiveByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

- 静默活体检测

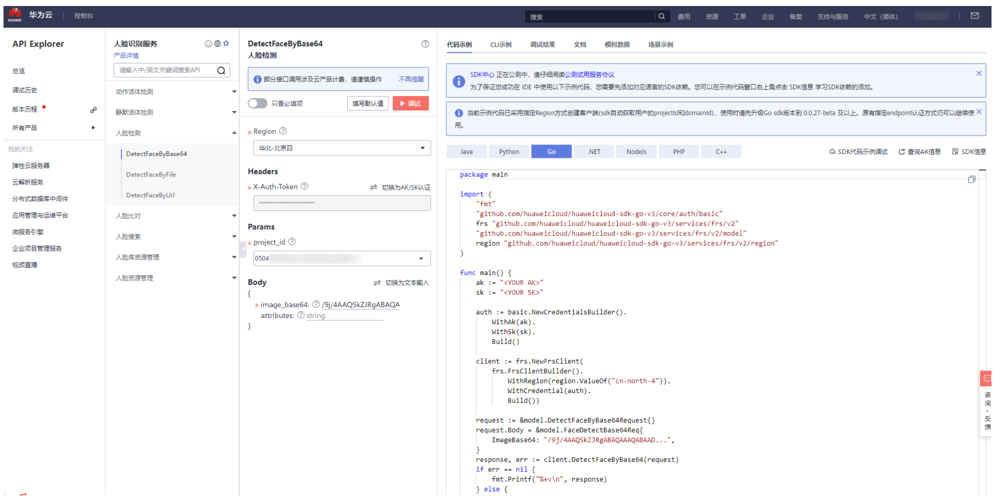
```
//detect live face by base64
request := &model.DetectLiveFaceByBase64Request{}
request.Body = &model.LiveDetectFaceBase64Req{
    ImageBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
}
response, err := client.DetectLiveFaceByBase64(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}

//detect live face by file
file, err := os.Open("/root/picture.jpg")
if err != nil {
   fmt.Println(err)
}
defer file.Close()
request := &model.DetectLiveFaceByFileRequest{}
request.Body = &model.DetectLiveFaceByFileRequestBody{
   ImageFile: def.NewFilePart(file),
}
response, err := client.DetectLiveFaceByFile(request)
if err == nil {
   fmt.Printf("%+v\n", response)
} else {
   fmt.Println(err)
}
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。同时，可在集成开发环境CloudIDE中完成代码的构建、调试和运行等操作。

图 4-4 API Explorer



DetectFaceByBase64
人脸检测

部分接口调用涉及云产品计费，请谨慎操作

Region: 华北-北京四

Headers
X-Auth-Token: ************************

Params
project_id: 0504[redacted]

Body
```
{
  image_base64: /9j/4AAQSkZJRgABAQA
  attributes: string
}
```

SDK中心 正在公测中，请仔细阅读公测试用服务协议
为了保证您成功在 IDE 中使用以下示例代码，您需要先添加对应语言的SDK依赖。您可以在示例代码窗口右上角点击 SDK信息 学习SDK依赖的添加。

当前示例代码已采用指定Region方式创建客户端(sdk自动获取用户的projectId和domainId)，使用时请先升级Go sdk版本到 0.0.27-beta 及以上。原有指定endpoint认证方式仍可以继续使用。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    frs "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/frs/v2/region"
)

func main() {
    ak := "<YOUR AK>"
    sk := "<YOUR SK>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        Build()

    client := frs.NewFrsClient(
        frs.FrsClientBuilder().
            WithRegion(region.ValueOf("cn-north-4")).
            WithCredential(auth).
            Build())

    request := &model.DetectFaceByBase64Request{}
    request.Body = &model.FaceDetectBase64Req{
        ImageBase64: "/9j/4AAQSkZJRgABAQAAAQABAAD...",
    }
    response, err := client.DetectFaceByBase64(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
```

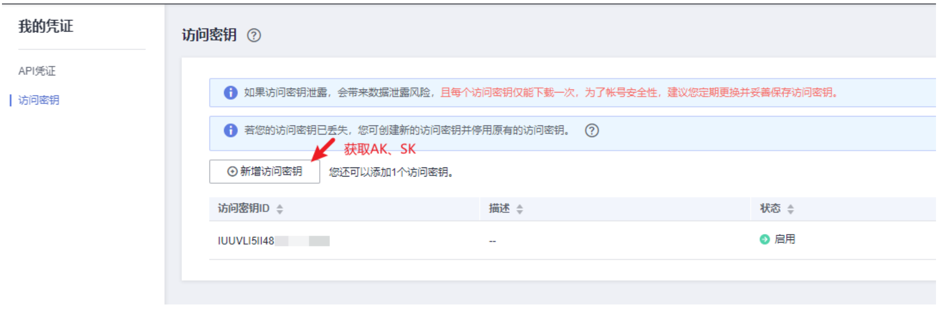
5 .NET SDK

本章节介绍.NET SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，.NET SDK 适用于.NET Standard 2.0 及其以上版本；C# 4.0 及其以上版本。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 5-1 获取 AK、SK



如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

若您的访问密钥已丢失，您可创建新的访问密钥并停用原有的访问密钥。

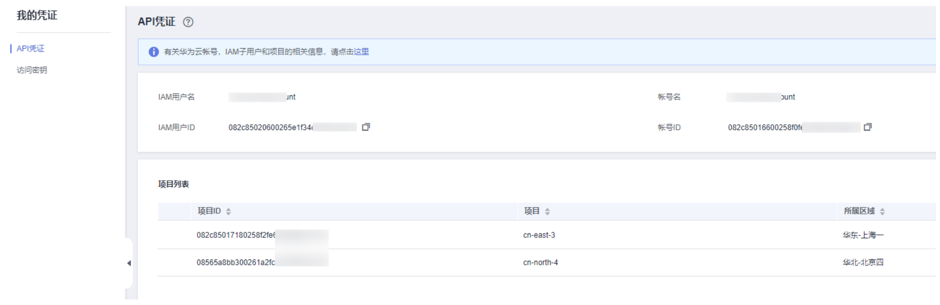
您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5II48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 5-2 我的凭证



有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名	[redacted]unt	帐号名	[redacted]ount
IAM用户ID	082c85020600265e1f34[redacted]	帐号ID	082c85016600258f0fe[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

使用SDK前，需要安装“HuaweiCloud.SDK.Core”和“HuaweiCloud.SDK.Frs”，有两种安装方式，分别如下。

- 使用 .NET CLI 工具
```
dotnet add package HuaweiCloud.SDK.Core
dotnet add package HuaweiCloud.SDK.Frs
```
- 使用 Package Manager
```
Install-Package HuaweiCloud.SDK.Core
Install-Package HuaweiCloud.SDK.Frs
```

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查并更新SDK至最新版本。

```
dotnet list package
dotnet add package HuaweiCloud.SDK.Core --version *
dotnet add package HuaweiCloud.SDK.Frs --version *
```

详细的SDK介绍，使用异步客户端，配置日志等操作请参见**SDK中心**、**.NET SDK使用指导**、**.NET SDK视频指导**。

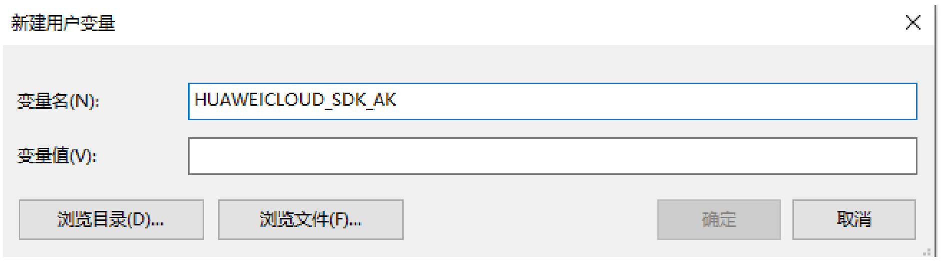
1. 导入依赖模块
```
using System;
using System.Collections.Generic;
using HuaweiCloud.SDK.Core;
using HuaweiCloud.SDK.Core.Auth;
using HuaweiCloud.SDK.Frs;
using HuaweiCloud.SDK.Frs.V2;
using HuaweiCloud.SDK.Frs.V2.Model;
```
2. 配置客户端连接参数
   - 默认配置
   ```
   // 使用默认配置
   var config = HttpConfig.GetDefaultConfig();
   ```
   - 网络代理（可选）
   ```
   // 根据需要配置网络代理
   config.ProxyHost = "proxy.huaweicloud.com";
   config.ProxyPort = 8080;
   config.ProxyUsername = "test";
   config.ProxyPassword = "test";
   ```
   - 超时配置（可选）
   ```
   // 默认超时时间为120秒，可根据需要调整
   config.Timeout = 120;
   ```
   - SSL配置（可选）
   ```
   // 根据需要配置是否跳过SSL证书验证
   config.IgnoreSslVerification = true;
   ```
3. 配置认证信息

   配置AK、SK信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK获取方法请参见**准备工作**。
   ```
   const string ak = Environment.GetEnvironmentVariable("HUAWEICLOUD_SDK_AK");
   const string sk = Environment.GetEnvironmentVariable("HUAWEICLOUD_SDK_SK");
   var auth = new BasicCredentials(ak, sk);
   ```

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。

图 5-3 Windows 环境新建环境变量


新建用户变量

变量名(N): HUAWEICLOUD_SDK_AK

变量值(V):

4. 初始化客户端
   - 指定云服务region方式（推荐）
     ```
     // 初始化指定云服务的客户端 {Service}Client ，以初始化FRS服务的 FrsClient 为例
     var client = FrsClient.NewBuilder()
             .WithCredential(auth)
             .WithRegion(FrsRegion.ValueOf("cn-north-4"))
             .WithHttpConfig(config)
             .Build();
     ```
   - 指定云服务endpoint方式
     ```
     // 指定终端节点，以FRS服务北京四的 endpoint 为例
     String endpoint = "https://face.cn-north-4.myhuaweicloud.com";

     // 初始化客户端认证信息，需要填写相应 projectId，以初始化 BasicCredentials 为例
     var auth = new BasicCredentials(ak, sk, projectId);

     // 初始化指定云服务的客户端 {Service}Client，以初始化FRS服务的 FrsClient 为例
     var client = FrsClient.NewBuilder()
         .WithCredential(auth)
         .WithEndPoint(endpoint)
         .WithHttpConfig(config)
         .Build();
     ```
     endpoint是华为云各服务应用区域和各服务的终端节点，详情请查看 **地区和终端节点** 。
5. 发送请求并查看响应
   ```
   // 以调用人脸检测接口 DetectFaceByBase64 为例
   var req = new DetectFaceByBase64Request
   {
   };
   req.Body = new FaceDetectBase64Req()
   {
       ImageBase64 = "图片的base64编码"
    };

   try
   {
       var resp = client.DetectFaceByBase64(req);
       var respStatusCode = resp.HttpStatusCode;
       Console.WriteLine(respStatusCode);
   }
   ```

说明

使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

6. 异常处理

表 5-1 异常处理

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

```
// 捕获和处理不同类型的异常
try
{
    var resp = client.DetectFaceByBase64(req);
    var respStatusCode = resp.HttpStatusCode;
    Console.WriteLine(respStatusCode);
}
catch (RequestTimeoutException requestTimeoutException)
{
    Console.WriteLine(requestTimeoutException.ErrorMessage);
}
catch (ServiceResponseException clientRequestException)
{
    Console.WriteLine(clientRequestException.HttpStatusCode);
    Console.WriteLine(clientRequestException.ErrorCode);
    Console.WriteLine(clientRequestException.ErrorMsg);
 }
catch (ConnectionException connectionException)
{
    Console.WriteLine(connectionException.ErrorMessage);
}
```

SDK demo 代码解析

- 人脸检测

```
var req = new DetectFaceByBase64Request
            {
            };
            req.Body = new FaceDetectBase64Req()
            {
                ImageBase64 = "图片的base64编码"
            };
try
            {
var resp = client.DetectFaceByBase64(req);
var respStatusCode = resp.HttpStatusCode;
                Console.WriteLine(respStatusCode);
            }
catch (RequestTimeoutException requestTimeoutException)
            {
                Console.WriteLine(requestTimeoutException.ErrorMessage);
            }
catch (ServiceResponseException clientRequestException)
            {
                Console.WriteLine(clientRequestException.HttpStatusCode);
                Console.WriteLine(clientRequestException.ErrorCode);
                Console.WriteLine(clientRequestException.ErrorMsg);
            }
catch (ConnectionException connectionException)
            {
                Console.WriteLine(connectionException.ErrorMessage);
            }
```

- 人脸比对

```
var req = new CompareFaceByBase64Request
            {
            };
            req.Body = new FaceCompareBase64Req()
            {
                Image1Base64 = "图片1的base64编码",
                Image2Base64 = "图片2的base64编码"
            };
try
            {
var resp = client.CompareFaceByBase64(req);
var respStatusCode = resp.HttpStatusCode;
                Console.WriteLine(respStatusCode);
            }
catch (RequestTimeoutException requestTimeoutException)
            {
                Console.WriteLine(requestTimeoutException.ErrorMessage);
            }
catch (ServiceResponseException clientRequestException)
            {
                Console.WriteLine(clientRequestException.HttpStatusCode);
                Console.WriteLine(clientRequestException.ErrorCode);
                Console.WriteLine(clientRequestException.ErrorMsg);
            }
catch (ConnectionException connectionException)
            {
                Console.WriteLine(connectionException.ErrorMessage);
            }
```

- 人脸搜索

```
var req = new SearchFaceByBase64Request
            {
            };
            req.Body = new FaceSearchBase64Req()
            {
                ImageBase64 = "图片的base64编码"
            };
try
            {
var resp = client.SearchFaceByBase64(req);
var respStatusCode = resp.HttpStatusCode;
                Console.WriteLine(respStatusCode);
            }
catch (RequestTimeoutException requestTimeoutException)
            {
                Console.WriteLine(requestTimeoutException.ErrorMessage);
            }
catch (ServiceResponseException clientRequestException)
            {
                Console.WriteLine(clientRequestException.HttpStatusCode);
                Console.WriteLine(clientRequestException.ErrorCode);
                Console.WriteLine(clientRequestException.ErrorMsg);
            }
catch (ConnectionException connectionException)
            {
                Console.WriteLine(connectionException.ErrorMessage);
            }
```

- 创建人脸库

```
var req = new CreateFaceSetRequest
            {
            };
            req.Body = new CreateFaceSetReq()
            {
                FaceSetName = "人脸库名称"
            };
try
            {
var resp = client.CreateFaceSet(req);
var respStatusCode = resp.HttpStatusCode;
                Console.WriteLine(respStatusCode);
            }
catch (RequestTimeoutException requestTimeoutException)
            {
                Console.WriteLine(requestTimeoutException.ErrorMessage);
            }
catch (ServiceResponseException clientRequestException)
            {
                Console.WriteLine(clientRequestException.HttpStatusCode);
                Console.WriteLine(clientRequestException.ErrorCode);
                Console.WriteLine(clientRequestException.ErrorMsg);
            }
catch (ConnectionException connectionException)
            {
                Console.WriteLine(connectionException.ErrorMessage);
            }
```

- 查询人脸库

```
var req = new ShowFaceSetRequest
            {
                FaceSetName = "人脸库名称"
            };
try
            {
var resp = client.ShowFaceSet(req);
var respStatusCode = resp.HttpStatusCode;
                Console.WriteLine(respStatusCode);
            }
catch (RequestTimeoutException requestTimeoutException)
            {
                Console.WriteLine(requestTimeoutException.ErrorMessage);
            }
catch (ServiceResponseException clientRequestException)
            {
                Console.WriteLine(clientRequestException.HttpStatusCode);
                Console.WriteLine(clientRequestException.ErrorCode);
                Console.WriteLine(clientRequestException.ErrorMsg);
            }
catch (ConnectionException connectionException)
            {
                Console.WriteLine(connectionException.ErrorMessage);
            }
```

- 查询所有人脸库

```
var req = new ShowAllFaceSetsRequest
          {
          };
try
          {
var resp = client.ShowAllFaceSets(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
              Console.WriteLine(clientRequestException.HttpStatusCode);
              Console.WriteLine(clientRequestException.ErrorCode);
              Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
              Console.WriteLine(connectionException.ErrorMessage);
          }
```

- 删除人脸库

```
var req = new DeleteFaceSetRequest
          {
              FaceSetName = "人脸库名称"
          };
try
          {
var resp = client.DeleteFaceSet(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
              Console.WriteLine(clientRequestException.HttpStatusCode);
              Console.WriteLine(clientRequestException.ErrorCode);
              Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
              Console.WriteLine(connectionException.ErrorMessage);
          }
```

- 添加人脸

```
var req = new AddFacesByBase64Request
          {
          };
          req.Body = new AddFacesBase64Req()
          {
              ImageBase64 = "图片的base64编码"
          };
try
          {
var resp = client.AddFacesByBase64(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
              Console.WriteLine(clientRequestException.HttpStatusCode);
              Console.WriteLine(clientRequestException.ErrorCode);
              Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
              Console.WriteLine(connectionException.ErrorMessage);
          }
```

- 删除人脸

```
var req = new DeleteFaceByFaceIdRequest
        {
            FaceSetName = "人脸库名称",
            FaceId = "人脸ID"
        };
try
        {
var resp = client.DeleteFaceByFaceId(req);
var respStatusCode = resp.HttpStatusCode;
            Console.WriteLine(respStatusCode);
        }
catch (RequestTimeoutException requestTimeoutException)
        {
            Console.WriteLine(requestTimeoutException.ErrorMessage);
        }
catch (ServiceResponseException clientRequestException)
        {
            Console.WriteLine(clientRequestException.HttpStatusCode);
            Console.WriteLine(clientRequestException.ErrorCode);
            Console.WriteLine(clientRequestException.ErrorMsg);
        }
catch (ConnectionException connectionException)
        {
            Console.WriteLine(connectionException.ErrorMessage);
        }
```

- 批量删除人脸

```
var req = new BatchDeleteFacesRequest
        {
        };
        req.Body = new DeleteFacesBatchReq()
        {
            Filter = "过滤条件"
        };
try
        {
var resp = client.BatchDeleteFaces(req);
var respStatusCode = resp.HttpStatusCode;
            Console.WriteLine(respStatusCode);
        }
catch (RequestTimeoutException requestTimeoutException)
        {
            Console.WriteLine(requestTimeoutException.ErrorMessage);
        }
catch (ServiceResponseException clientRequestException)
        {
            Console.WriteLine(clientRequestException.HttpStatusCode);
            Console.WriteLine(clientRequestException.ErrorCode);
            Console.WriteLine(clientRequestException.ErrorMsg);
        }
catch (ConnectionException connectionException)
        {
            Console.WriteLine(connectionException.ErrorMessage);
        }
```

- 更新人脸

```
var req = new UpdateFaceRequest
        {
        };
        req.Body = new UpdateFaceReq()
```

```
          {
              FaceId = "人脸库ID"
          };
try
          {
var resp = client.UpdateFace(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
              Console.WriteLine(clientRequestException.HttpStatusCode);
              Console.WriteLine(clientRequestException.ErrorCode);
              Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
              Console.WriteLine(connectionException.ErrorMessage);
          }
       }
```

- 查询人脸

```
var req = new ShowFacesByFaceIdRequest
          {
              FaceSetName = "人脸库名称",
              FaceId = "人脸ID"
          };
try
          {
var resp = client.ShowFacesByFaceId(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
              Console.WriteLine(clientRequestException.HttpStatusCode);
              Console.WriteLine(clientRequestException.ErrorCode);
              Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
              Console.WriteLine(connectionException.ErrorMessage);
          }
```

- 动作活体检测

```
var req = new DetectLiveByBase64Request
          {
          };
          req.Body = new LiveDetectBase64Req()
          {
              Actions = "动作代码顺序列表",
              VideoBase64 = "视频数据的base64编码"
          };
try
          {
var resp = client.DetectLiveByBase64(req);
var respStatusCode = resp.HttpStatusCode;
              Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
              Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
             Console.WriteLine(clientRequestException.HttpStatusCode);
             Console.WriteLine(clientRequestException.ErrorCode);
             Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
             Console.WriteLine(connectionException.ErrorMessage);
          }
```

- 静默活体检测

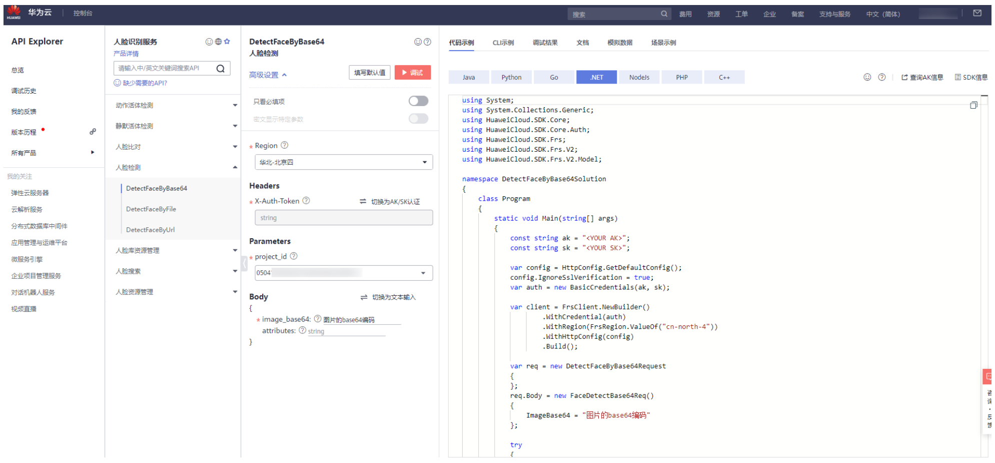
```
var req = new DetectLiveFaceByBase64Request
          {
          };
          req.Body = new LiveDetectFaceBase64Req()
          {
             ImageBase64 = "图片的base64编码"
          };
try
          {
var resp = client.DetectLiveFaceByBase64(req);
var respStatusCode = resp.HttpStatusCode;
             Console.WriteLine(respStatusCode);
          }
catch (RequestTimeoutException requestTimeoutException)
          {
             Console.WriteLine(requestTimeoutException.ErrorMessage);
          }
catch (ServiceResponseException clientRequestException)
          {
             Console.WriteLine(clientRequestException.HttpStatusCode);
             Console.WriteLine(clientRequestException.ErrorCode);
             Console.WriteLine(clientRequestException.ErrorMsg);
          }
catch (ConnectionException connectionException)
          {
             Console.WriteLine(connectionException.ErrorMessage);
          }
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。

图 5-4 API Explorer



DetectFaceByBase64
人脸检测

* Region
华北-北京四

Headers

* X-Auth-Token
string

Parameters

* project_id
0504[redacted]

Body
```
{
  * image_base64: 图片的base64编码
  attributes: string
}
```

```
using System;
using System.Collections.Generic;
using HuaweiCloud.SDK.Core;
using HuaweiCloud.SDK.Core.Auth;
using HuaweiCloud.SDK.Frs;
using HuaweiCloud.SDK.Frs.V2;
using HuaweiCloud.SDK.Frs.V2.Model;

namespace DetectFaceByBase64Solution
{
    class Program
    {
        static void Main(string[] args)
        {
            const string ak = "<YOUR AK>";
            const string sk = "<YOUR SK>";

            var config = HttpConfig.GetDefaultConfig();
            config.IgnoreSslVerification = true;
            var auth = new BasicCredentials(ak, sk);

            var client = FrsClient.NewBuilder()
                    .WithCredential(auth)
                    .WithRegion(FrsRegion.ValueOf("cn-north-4"))
                    .WithHttpConfig(config)
                    .Build();

            var req = new DetectFaceByBase64Request
            {
            };
            req.Body = new FaceDetectBase64Req()
            {
                ImageBase64 = "图片的base64编码"
            };

            try
            {
```

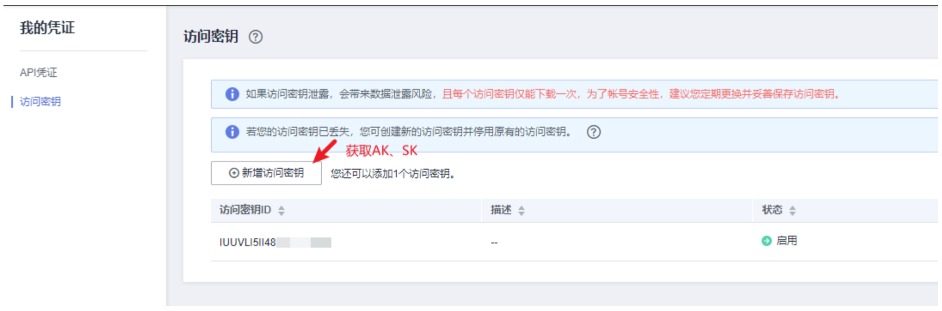
6 Node.js SDK

本章节介绍新版Node.js SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，支持Node 10.16.1 及其以上版本。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 **6-1** 获取 AK、SK



我的凭证

API凭证

访问密钥

访问密钥

如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

若您的访问密钥已丢失，您可创建新的访问密钥并停用原有的访问密钥。

获取AK、SK

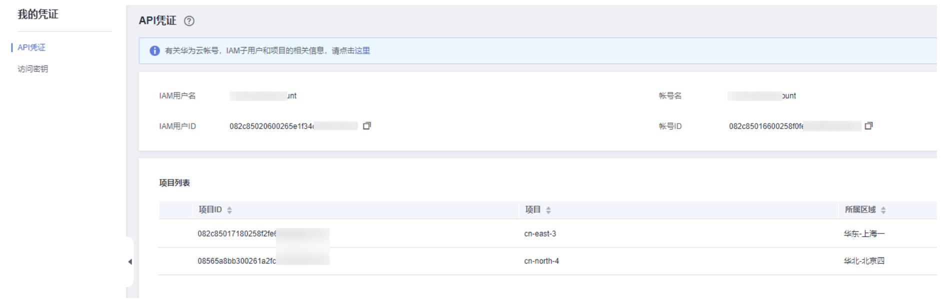
新增访问密钥 您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5II48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 **6-2** 我的凭证



我的凭证

API凭证

访问密钥

API凭证

有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名	[redacted]unt	帐号名	[redacted]ount
IAM用户ID	082c85020600265e1f34[redacted]	帐号ID	082c85016600258f0f[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

使用SDK前，需要安装“@huaweicloud/huaweicloud-sdk-core”和“@huaweicloud/huaweicloud-sdk-frs”。

推荐您使用 npm 安装 SDK。

```
npm install @huaweicloud/huaweicloud-sdk-core
npm i @huaweicloud/huaweicloud-sdk-frs
```

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查并更新SDK至最新版本。

```
npm list @huaweicloud/huaweicloud-sdk-core
npm list @huaweicloud/huaweicloud-sdk-frs
npm update @huaweicloud/huaweicloud-sdk-core
npm update @huaweicloud/huaweicloud-sdk-frs
```

详细的SDK介绍请参见**SDK中心**、**Node.js SDK使用指导**、**Node.js SDK视频指导**。

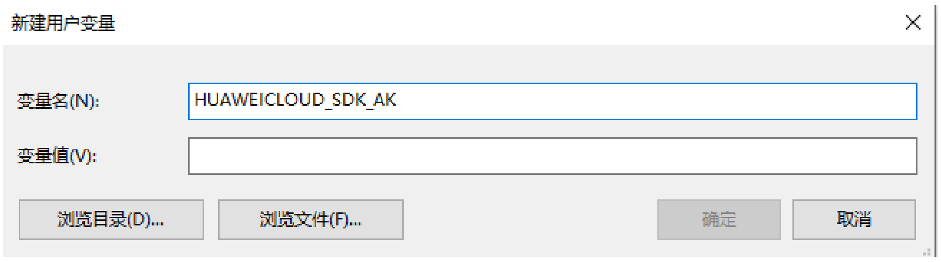
1. 导入依赖模块
```
const core = require('@huaweicloud/huaweicloud-sdk-core');
const frs = require("@huaweicloud/huaweicloud-sdk-frs");
```
2. 配置客户端链接参数
   - 默认配置
   ```
   const client = frs.FrsClient.newBuilder()
   ```
   - 网络代理（可选）
   ```
   // 使用代理服务器（可选）
   client.withProxyAgent("http://username:password@proxy.huaweicloud.com:8080")
   ```
   - SSL配置（可选）
   ```
   // 配置跳过服务端证书验证（可选）
   process.env.NODE_TLS_REJECT_UNAUTHORIZED = "0"
   ```
3. 配置认证信息

   配置AK、SK、project_id信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK和project_id获取方法请参见**准备工作**。
```
const ak = process.env.HUAWEICLOUD_SDK_AK;
const sk = process.env.HUAWEICLOUD_SDK_SK;
const project_id = process.env.PROJECT_ID;
const credentials = new core.BasicCredentials()
                    .withAk(ak)
                    .withSk(sk)
                    .withProjectId(project_id)
```

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK，HUAWEICLOUD_SDK_SK和PROJECT_ID。

图 6-3 Windows 环境新建环境变量



变量名(N): HUAWEICLOUD_SDK_AK
变量值(V):

4. 初始化客户端

   指定云服务endpoint方式

```
// 指定终端节点，以 FRS 服务北京四的 endpoint 为例
const endpoint = "https://face.cn-north-4.myhuaweicloud.com";
const client = frs.FrsClient.newBuilder()
                            .withCredential(credentials)
                            .withEndpoint(endpoint)
                            .build();
```

   endpoint是华为云各服务应用区域和各服务的终端节点，详情请查看 **地区和终端节点** 。

5. 发送请求并查看响应

```
// 以调用人脸检测接口 DetectFaceByBase64 为例
const request = new frs.DetectFaceByBase64Request();
const body = new frs.FaceDetectBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.detectFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

   说明

   使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

SDK demo 代码解析

- 人脸检测

```
const request = new frs.DetectFaceByBase64Request();
const body = new frs.FaceDetectBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.detectFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 人脸比对

```
const request = new frs.CompareFaceByBase64Request();
const body = new frs.FaceCompareBase64Req();
body.withImage1Base64("图片1的base64编码");
body.withImage2Base64("图片2的base64编码");
request.withBody(body);
const result = client.compareFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 人脸搜索

```
const request = new frs.SearchFaceByBase64Request();
const body = new frs.FaceSearchBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.searchFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 创建人脸库

```
const request = new frs.CreateFaceSetRequest();
const body = new frs.CreateFaceSetReq();
body.withFaceSetName("人脸库名称");
request.withBody(body);
const result = client.createFaceSet(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 查询人脸库

```
const request = new frs.ShowFaceSetRequest();
request.faceSetName = "人脸库名称";
const result = client.showFaceSet(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 查询所有人脸库

```
const request = new frs.ShowAllFaceSetsRequest();
const result = client.showAllFaceSets(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 删除人脸库

```
const request = new frs.DeleteFaceSetRequest();
request.faceSetName = "人脸库名称";
const result = client.deleteFaceSet(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 添加人脸

```
const request = new frs.AddFacesByBase64Request();
const body = new frs.AddFacesBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.addFacesByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 删除人脸

```
const request = new frs.DeleteFaceByFaceIdRequest();
request.faceSetName = "人脸库名称";
request.faceId = "人脸ID";
const result = client.deleteFaceByFaceId(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 批量删除人脸

```
const request = new frs.BatchDeleteFacesRequest();
request.faceSetName = "人脸库名称";
const body = new frs.DeleteFacesBatchReq();
body.withFilter("过滤条件");
request.withBody(body);
const result = client.batchDeleteFaces(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 更新人脸

```
const request = new frs.UpdateFaceRequest();
request.faceSetName = "人脸库名称";
const body = new frs.UpdateFaceReq();
body.withFaceId("人脸库ID");
request.withBody(body);
const result = client.updateFace(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 查询人脸

```
const request = new frs.ShowFacesByFaceIdRequest();
request.faceSetName = "人脸库名称";
request.faceId = "人脸ID";
const result = client.showFacesByFaceId(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 动作活体检测

```
const request = new frs.DetectLiveByBase64Request();
const body = new frs.LiveDetectBase64Req();
body.withActions("动作代码顺序列表");
body.withVideoBase64("视频数据的base64编码");
request.withBody(body);
const result = client.detectLiveByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 静默活体检测

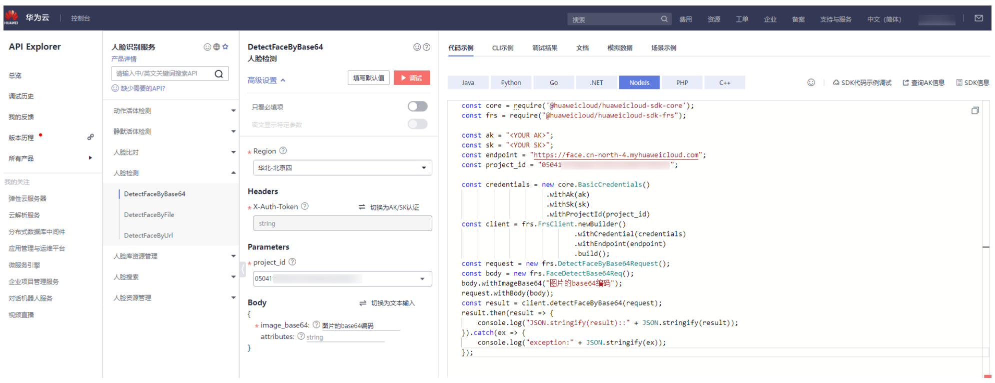
```
const request = new frs.DetectLiveFaceByBase64Request();
const body = new frs.LiveDetectFaceBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.detectLiveFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。同时，可在集成开发环境CloudIDE中完成代码的构建、调试和运行等操作。

图 6-4 API Explorer



DetectFaceByBase64
人脸检测

Region: 华北-北京四

Headers
X-Auth-Token: string

Parameters
project_id: 05041[redacted]

Body
```
{
  image_base64: 图片的base64编码
  attributes: string
}
```

```
const core = require('@huaweicloud/huaweicloud-sdk-core');
const frs = require("@huaweicloud/huaweicloud-sdk-frs");

const ak = "<YOUR AK>";
const sk = "<YOUR SK>";
const endpoint = "https://face.cn-north-4.myhuaweicloud.com";
const project_id = "05041[redacted]";

const credentials = new core.BasicCredentials()
                    .withAk(ak)
                    .withSk(sk)
                    .withProjectId(project_id)
const client = frs.FrsClient.newBuilder()
                        .withCredential(credentials)
                        .withEndpoint(endpoint)
                        .build();
const request = new frs.DetectFaceByBase64Request();
const body = new frs.FaceDetectBase64Req();
body.withImageBase64("图片的base64编码");
request.withBody(body);
const result = client.detectFaceByBase64(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

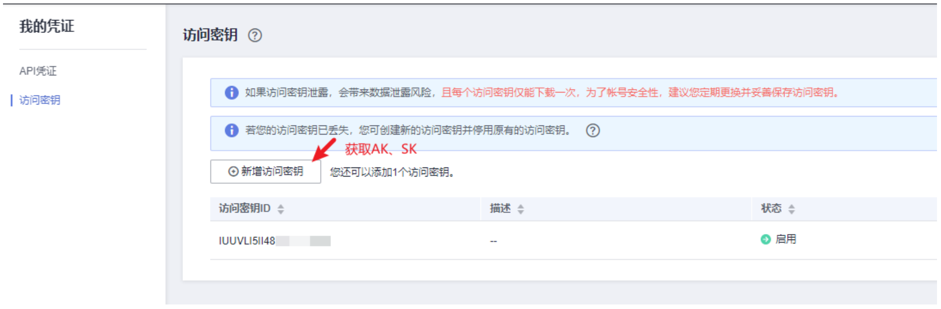
7 PHP SDK

本章节介绍新版PHP SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，PHP 5.6 及以上版本，可执行 php --version 检查当前的版本信息。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 7-1 获取 AK、SK



如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

若您的访问密钥已丢失，您可以创建新的访问密钥并停用原有的访问密钥。

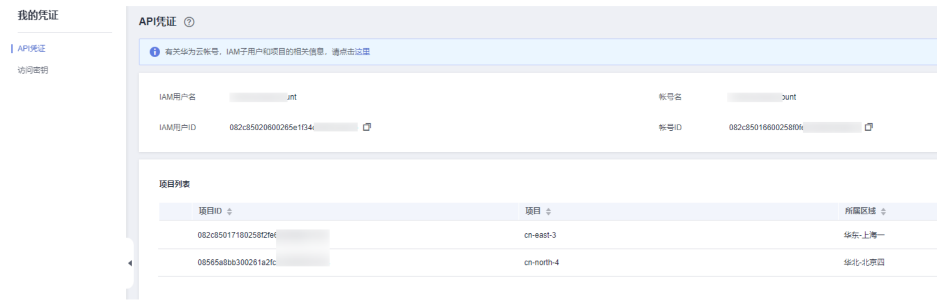
您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5Il48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 7-2 我的凭证



有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名	[redacted]unt	帐号名	[redacted]ount
IAM用户ID	082c85020600265e1f34[redacted]	帐号ID	082c85016600258f0fe[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

推荐使用 **Composer** 安装 SDK 。

Composer 是 php 的依赖管理工具，允许您在项目中声明依赖关系并安装这些依赖：

```
// 安装 Composer
curl -sS https://getcomposer.org/installer | php
// 安装 PHP SDK
composer require huaweicloud/huaweicloud-sdk-php
```

安装完毕后，你需要引入 Composer 的自动加载文件：

```
require 'path/to/vendor/autoload.php';
```

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以通过运行以下命令来检查并更新SDK至最新版本。

```
composer show huaweicloud/huaweicloud-sdk-php
composer update huaweicloud/huaweicloud-sdk-php
```

详细的SDK介绍，使用异步客户端，配置日志请参见**SDK中心**、**PHP SDK使用指导**、**PHP SDK使用视频**。

1. 导入依赖模块
```
<?php
namespace HuaweiCloud\SDK\Frs\V2\Model;
require_once "vendor/autoload.php";
use HuaweiCloud\SDK\Core\Auth\BasicCredentials;
use HuaweiCloud\SDK\Core\Http\HttpConfig;
use HuaweiCloud\SDK\Core\Exceptions\ConnectionException;
use HuaweiCloud\SDK\Core\Exceptions\RequestTimeoutException;
use HuaweiCloud\SDK\Core\Exceptions\ServiceResponseException;
use HuaweiCloud\SDK\Frs\V2\FrsClient;
```
2. 配置客户端连接参数
   - 默认配置
   ```
   // 使用默认配置
   $config = HttpConfig::getDefaultConfig();
   ```
   - 网络代理（可选）
   ```
   // 使用代理服务器
   $config->setProxyProtocol('http');
   $config->setProxyHost('proxy.huawei.com');
   $config->setProxyPort(8080);
   $config->setProxyUser('username');
   $config->setProxyPassword('password');
   ```
   - 超时配置（可选）
   ```
   // 默认连接超时时间为60秒，读取超时时间为120秒。可根据需要修改默认值。
   $config->setTimeout(120);
   $config->setConnectionTimeout(60);
   ```
   - SSL配置（可选）
   ```
   // 配置跳过服务端证书验证
   $config->setIgnoreSslVerification(true);
   // 配置服务器端CA证书，用于SDK验证服务端证书合法性
   $config->setCertFile("{yourCertFile}");
   ```
3. 配置认证信息

   配置AK、SK、projectId信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。
   ```
   // 终端节点以 FRS 服务北京四的 endpoint 为例
   ```

```
$ak = getenv('HUAWEICLOUD_SDK_AK');
$sk = getenv('HUAWEICLOUD_SDK_SK');
$endpoint = "https://face.cn-north-4.myhuaweicloud.com";
$projectId = getenv('PROJECT_ID');
$credentials = new BasicCredentials($ak,$sk,$projectId);
```

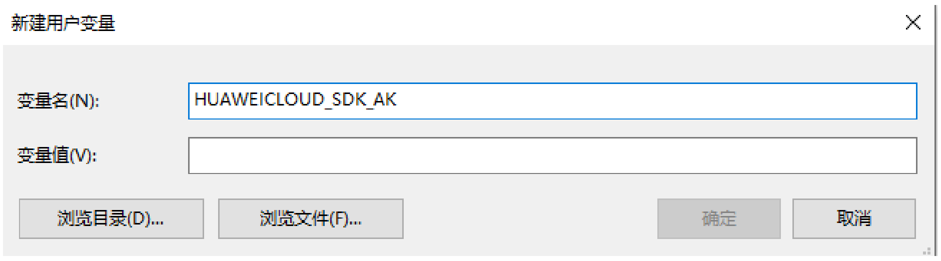
认证参数说明：

- ak、sk：访问秘钥信息，获取方法请参见准备工作。
- projectId：华为云项目ID，获取方法请参见准备工作。
- endpoint：华为云各服务应用区域和各服务的终端节点，详情请查看 地区和终端节点 。

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK，HUAWEICLOUD_SDK_SK和PROJECT_ID。

图 7-3 Windows 环境新建环境变量


新建用户变量

变量名(N): HUAWEICLOUD_SDK_AK

变量值(V):

4. 初始化客户端

   指定云服务endpoint方式

```
$client = FrsClient::newBuilder(new FrsClient)
  ->withHttpConfig($config)
  ->withEndpoint($endpoint)
  ->withCredentials($credentials)
  ->build();
```

5. 发送并查看响应

```
// 以调用人脸检测接口 DetectFaceByBase64 为例
$request = new DetectFaceByBase64Request();
$body = new FaceDetectBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->DetectFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
```

```
echo "\n";
echo $response;
```

说明

使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

6. 异常处理

表 7-1 异常处理

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

```
// 捕获和处理不同类型的异常
try {
  $response = $client->DetectFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

SDK demo 代码解析

- 人脸检测

```
$request = new DetectFaceByBase64Request();
$body = new FaceDetectBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->DetectFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 人脸比对

```
$request = new CompareFaceByBase64Request();
$body = new FaceCompareBase64Req();
$body->setImage1Base64("图片1的base64编码");
$body->setImage2Base64("图片2的base64编码");
$request->setBody($body);
try {
  $response = $client->CompareFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 人脸搜索

```
$request = new SearchFaceByBase64Request();
$body = new FaceSearchBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->SearchFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 创建人脸库

```
$request = new CreateFaceSetRequest();
$body = new CreateFaceSetReq();
$body->setFaceSetName("人脸库名称");
$request->setBody($body);
try {
  $response = $client->CreateFaceSet($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 查询人脸库

```
$request = new ShowFaceSetRequest();
$request->setFaceSetName("人脸库名称");
try {
  $response = $client->ShowFaceSet($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 查询所有人脸库

```
$request = new ShowAllFaceSetsRequest();
try {
  $response = $client->ShowAllFaceSets($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 删除人脸库

```
$request = new DeleteFaceSetRequest();
$request->setFaceSetName("人脸库名称");
try {
  $response = $client->DeleteFaceSet($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 添加人脸

```
$request = new AddFacesByBase64Request();
$body = new AddFacesBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->AddFacesByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 删除人脸

```
$request = new DeleteFaceByFaceIdRequest();
$request->setFaceSetName("人脸库名称");
$request->setFaceId("人脸ID");
try {
  $response = $client->DeleteFaceByFaceId($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 批量删除人脸

```
$request = new BatchDeleteFacesRequest();
$request->setFaceSetName("人脸库名称");
$body = newDeleteFacesBatchReq();
$body->setFilter("过滤条件");
$request->setBody($body);
try {
  $response = $client->BatchDeleteFaces($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 更新人脸

```
$request = new UpdateFaceRequest();
$request->setFaceSetName("人脸库名称");
$body = newUpdateFaceReq();
```

```
$body->setFaceId("人脸库ID");
$request->setBody($body);
try {
  $response = $client->UpdateFace($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 查询人脸

```
$request = newShowFacesByFaceIdRequest();
$request->setFaceSetName("人脸库名称");
$request->setFaceId("人脸ID");
try {
  $response = $client->ShowFacesByFaceId($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 动作活体检测

```
$request = new DetectLiveByBase64Request();
$body = new LiveDetectBase64Req();
$body->setActions("动作代码顺序列表");
$body->setVideoBase64("视频数据的base64编码");
$request->setBody($body);
try {
  $response = $client->DetectLiveByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

- 静默活体检测

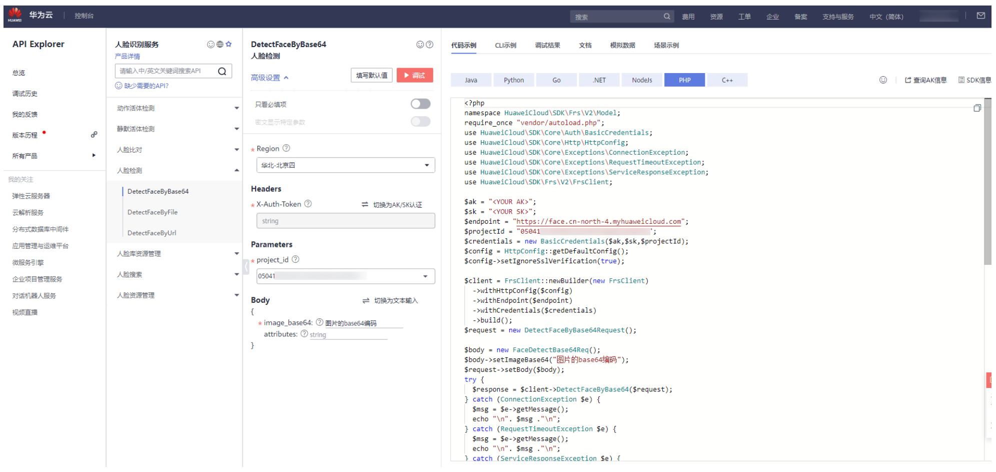
```
$request = new DetectLiveFaceByBase64Request();
$body = new LiveDetectFaceBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->DetectLiveFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
  echo "\n";
  echo $e->getHttpStatusCode(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
echo "\n";
echo $response;
```

SDK 代码自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。

图 7-4 API Explorer



Region: 华北-北京四

Headers
X-Author-Token: string

Parameters
project_id: 05041[redacted]

Body
```
{
  image_base64: 图片的base64编码
  attributes: string
}
```

```
<?php
namespace HuaweiCloud\SDK\Frs\V2\Model;
require_once "vendor/autoload.php";
use HuaweiCloud\SDK\Core\Auth\BasicCredentials;
use HuaweiCloud\SDK\Core\Http\HttpConfig;
use HuaweiCloud\SDK\Core\Exceptions\ConnectionException;
use HuaweiCloud\SDK\Core\Exceptions\RequestTimeoutException;
use HuaweiCloud\SDK\Core\Exceptions\ServiceResponseException;
use HuaweiCloud\SDK\Frs\V2\FrsClient;

$ak = "<YOUR AK>";
$sk = "<YOUR SK>";
$endpoint = "https://face.cn-north-4.myhuaweicloud.com";
$projectId = "05041[redacted]";
$credentials = new BasicCredentials($ak,$sk,$projectId);
$config = HttpConfig::getDefaultConfig();
$config->setIgnoreSslVerification(true);

$client = FrsClient::newBuilder(new FrsClient)
  ->withHttpConfig($config)
  ->withEndpoint($endpoint)
  ->withCredentials($credentials)
  ->build();
$request = new DetectFaceByBase64Request();

$body = new FaceDetectBase64Req();
$body->setImageBase64("图片的base64编码");
$request->setBody($body);
try {
  $response = $client->DetectFaceByBase64($request);
} catch (ConnectionException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (RequestTimeoutException $e) {
  $msg = $e->getMessage();
  echo "\n". $msg ."\n";
} catch (ServiceResponseException $e) {
```

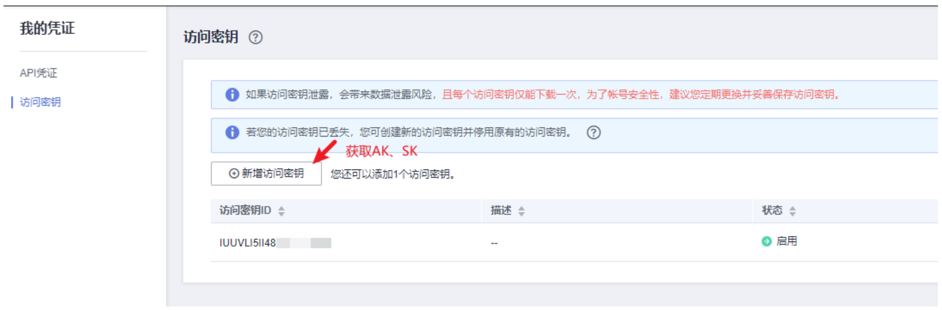
8 C++ SDK

本章节介绍新版C++ SDK，您可以参考本章节进行快速集成开发。

准备工作

- 注册华为账号并开通华为云，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，支持 C++ 14 及以上版本，要求安装 CMake 3.10 及以上版本。
- 登录“我的凭证 > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

图 8-1 获取 AK、SK



我的凭证

API凭证

访问密钥

访问密钥

如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

若您的访问密钥已丢失，您可以创建新的访问密钥并停用原有的访问密钥。

获取AK、SK

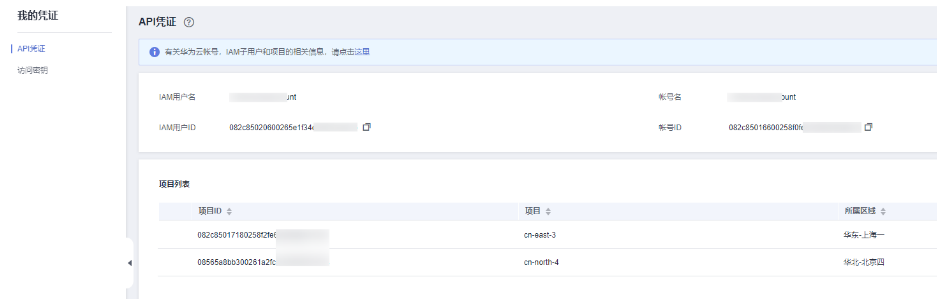
新增访问密钥 您还可以添加1个访问密钥。

访问密钥ID	描述	状态
IUUVLI5II48[redacted]	--	启用

- 登录“我的凭证”页面，获取“IAM用户名”、“账号名”以及待使用区域的“项目ID”。调用服务时会用到这些信息，请提前保存。

  本样例以“华北-北京四”区域为例，获取对应的项目ID（project_id）。

图 8-2 我的凭证



我的凭证

API凭证

访问密钥

API凭证

有关华为云帐号，IAM子用户和项目的相关信息，请点击这里

IAM用户名 [redacted]unt　　帐号名 [redacted]ount

IAM用户ID 082c85020600265e1f34[redacted]　　帐号ID 082c85016600258f0f[redacted]

项目列表

项目ID	项目	所属区域
082c85017180258f2fe[redacted]	cn-east-3	华东-上海一
08565a8bb300261a2fc[redacted]	cn-north-4	华北-北京四

安装 SDK

- 在Linux系统上安装SDK
  a. 获取依赖包

     所需的这些第三方软件包在大部分系统的包管理工具中都有提供，例如基于Debian/Ubuntu 的系统。
     ```
     sudo apt-get install libcurl4-openssl-dev libboost-all-dev libssl-dev libcpprest-dev
     ```
     spdlog 需要从源码进行安装。
     ```
     git clone https://github.com/gabime/spdlog.git
     cd spdlog
     mkdir build
     cd build
     cmake -DCMAKE_POSITION_INDEPENDENT_CODE=ON ..  // 用以生成动态库
     make
     sudo make install
     ```
  b. 编译安装
     ```
     git clone https://github.com/huaweicloud/huaweicloud-sdk-cpp-v3.git
     cd huaweicloud-sdk-cpp-v3
     mkdir build
     cd build
     cmake ..
     make
     sudo make install
     ```
     完成上述操作后，C++ SDK 安装目录为 /usr/local。
- 在Windows系统上安装SDK
  a. 安装 vcpkg 并使用 vcpkg 安装所需软件包
     ```
     vcpkg install curl cpprestsdk boost openssl spdlog
     ```
  b. 使用CLion进行编译
     i. 使用CLion打开huaweicloud-sdk-cpp-v3 目录。
     ii. 选择“File > Settings”。
     iii. 选择“Build, Execution, Deployment > > CMake”。
     iv. 在CMake options中加入：
         ```
         -DCMAKE_TOOLCHAIN_FILE={your vcpkg install dir}/scripts/buildsystems/vcpkg.cmake
         ```
     v. 右键 CMakeLists.txt 选择 Load CMake Project。
     vi. 选择Build开始编译。
  c. 安装C++ SDK

     编译完成后选择“Build > Install”。

     完成上述操作后，C++ SDK 安装目录为 C:\Program File (x86)\huaweicloud-sdk-cpp-v3。

开始使用

在开始使用之前，请确保您安装的是最新版本的SDK。使用过时的版本可能会导致兼容性问题或无法使用最新功能。您可以参考**安装SDK**完成sdk的安装和编译。

详细的SDK介绍，使用异步客户端，配置日志等操作请参见**SDK中心**、**C++ SDK使用指导**。

1. 导入依赖模块
   ```
   #include <cstdlib>
   #include <iostream>
   #include <string>
   #include <memory>
   ```

```
#include <huaweicloud/core/exception/Exceptions.h>
#include <huaweicloud/core/Client.h>
#include <huaweicloud/frs/v2/FrsClient.h>
using namespace HuaweiCloud::Sdk::Frs::V2;
using namespace HuaweiCloud::Sdk::Frs::V2::Model;
using namespace HuaweiCloud::Sdk::Core;
using namespace HuaweiCloud::Sdk::Core::Exception;
using namespace std;
```

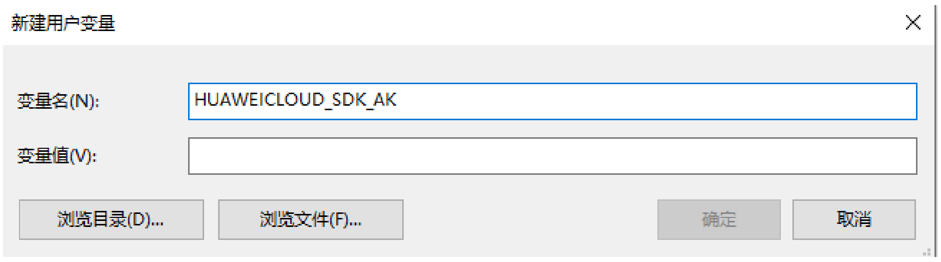
2. 配置客户端连接参数
   - 默认配置
     ```
     // 使用默认配置
     HttpConfig httpConfig = HttpConfig();
     ```
   - 网络代理（可选）
     ```
     // 根据需要配置网络代理
     httpConfig.setProxyProtocol("http");
     httpConfig.setProxyHost("proxy.huawei.com");
     httpConfig.setProxyPort("8080");
     httpConfig.setProxyUser("username");
     httpConfig.setProxyPassword("password");
     ```
   - 超时配置（可选）
     ```
     // 默认连接超时为60秒，默认读取超时为120秒。可根据需求修改该默认值
     httpConfig.setConnectTimeout(60);
     httpConfig.setReadTimeout(120);
     ```
   - SSL配置（可选）
     ```
     // 配置跳过服务端证书验证
     httpConfig.setIgnoreSslVerification(true);
     ```
3. 配置认证信息

   配置AK、SK、projectId信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。
   ```
   string ak = getenv("HUAWEICLOUD_SDK_AK");
   string sk = getenv("HUAWEICLOUD_SDK_SK");
   string projectId = getenv("PROJECT_ID");
   auto auth = std::make_unique<BasicCredentials>();
   auth->withAk(ak)
     .withSk(sk)
     .withProjectId(projectId);
   ```
   认证参数说明：
   - ak、sk：访问秘钥信息，获取方法请参见准备工作。
   - projectId：华为云项目ID，获取方法请参见准备工作。

注意

- 认证用的 ak 和sk 硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。
- 本示例以 ak 和 sk 保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK，HUAWEICLOUD_SDK_SK和PROJECT_ID。

图 8-3 Windows 环境新建环境变量



变量名(N): HUAWEICLOUD_SDK_AK
变量值(V):

4. 初始化客户端

   指定云服务endpoint方式

```
string endpoint = "https://face.cn-north-4.myhuaweicloud.com";
auto client = FrsClient::newBuilder()
        .withCredentials(std::unique_ptr<Credentials>(auth.release()))
        .withHttpConfig(httpConfig)
        .withEndPoint(endpoint)
        .build();
```

   endpoint：华为云各服务应用区域和各服务的终端节点，详情请查看 地区和终端节点 。

5. 发送请求并查看响应

```
// 以调用人脸检测接口 RecognizeGeneralTable 为例
DetectFaceByBase64Request request;
FaceDetectBase64Req body;
body.setImageBase64("图片的base64编码");
request.setBody(body);
std::cout << "-----begin execute request-------" << std::endl;
```

   说明

   使用人脸比对SDK时，image1、image2参数需为相同类型，即同为url、base64或file。

6. 异常处理

表 8-1 异常处理

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

```
// 捕获和处理不同类型的异常
try {
        auto reponse = client->detectFaceByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

SDK demo 代码解析

- 人脸检测

```
DetectFaceByBase64Request request;
FaceDetectBase64Req body;
    body.setImageBase64("图片的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->detectFaceByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 人脸比对

```
CompareFaceByBase64Request request;
FaceCompareBase64Req body;
    body.setImage1Base64("图片1的base64编码");
    body.setImage2Base64("图片2的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->compareFaceByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 人脸搜索

```
SearchFaceByBase64Request request;
    request.setFaceSetName("人脸库名称");
FaceSearchBase64Req body;
    body.setImageBase64("图片的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->searchFaceByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 创建人脸库

```
CreateFaceSetRequest request;
CreateFaceSetReq body;
    body.setFaceSetName("人脸库名称");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->createFaceSet(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 查询人脸库

```
ShowFaceSetRequest request;
    request.setFaceSetName("人脸库名称");
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->showFaceSet(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 查询所有人脸库

```
ShowAllFaceSetsRequest request;
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->showAllFaceSets(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 删除人脸库

```
DeleteFaceSetRequest request;
    request.setFaceSetName("人脸库名称");
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->deleteFaceSet(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 添加人脸

```
AddFacesByBase64Request request;
    request.setFaceSetName("人脸库名称");
AddFacesBase64Req body;
    body.setImageBase64("图片的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->addFacesByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 删除人脸

```
DeleteFaceByFaceIdRequest request;
    request.setFaceSetName("人脸库名称");
    request.setFaceId("人脸ID");
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->deleteFaceByFaceId(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std::cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 批量删除人脸

```
BatchDeleteFacesRequest request;
    request.setFaceSetName("人脸库名称");
DeleteFacesBatchReq body;
    body.setFilter("过滤条件");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
```

```
try {
        auto reponse = client->batchDeleteFaces(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 更新人脸

```
UpdateFaceRequest request;
    request.setFaceSetName("人脸库名称");
UpdateFaceReq body;
    body.setFaceId("人脸库ID");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->updateFace(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 查询人脸

```
ShowFacesByFaceIdRequest request;
    request.setFaceSetName("人脸库名城管");
    request.setFaceId("人脸ID");
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->showFacesByFaceId(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 动作活体检测

```
DetectLiveByBase64Request request;
LiveDetectBase64Req body;
    body.setActions("动作代码顺序列表");
    body.setVideoBase64("视频数据的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->detectLiveByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

- 静默活体检测

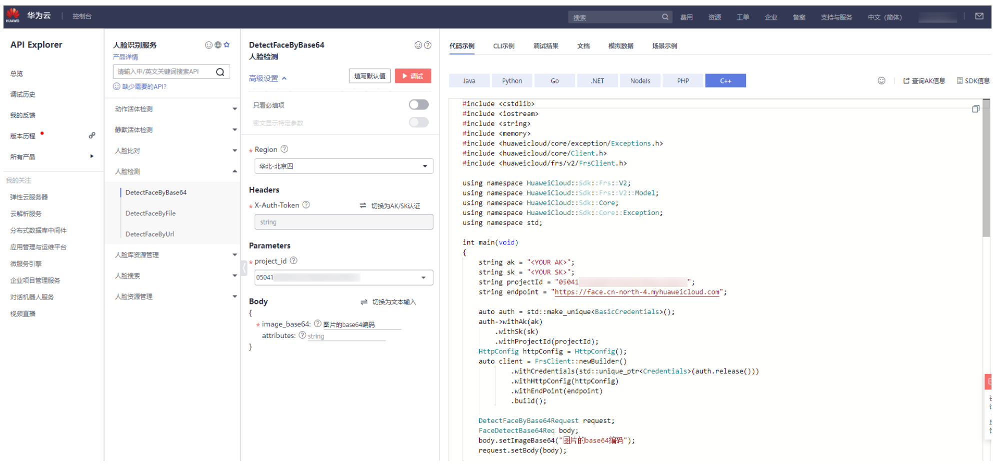
```
DetectLiveFaceByBase64Request request;
LiveDetectFaceBase64Req body;
    body.setImageBase64("图片的base64编码");
    request.setBody(body);
    std::cout << "-----begin execute request-------" << std::endl;
try {
        auto reponse = client->detectLiveFaceByBase64(request);
        std::cout << reponse->getHttpBody() << std::endl;
    } catch (HostUnreachableException& e) {
        std::cout << "host unreachable:" << e.what() << std::endl;
    } catch (SslHandShakeException& e) {
        std::cout << "ssl handshake error:" << e.what() << std::endl;
    } catch (RetryOutageException& e) {
        std::cout << "retryoutage error:" << e.what() << std::endl;
    } catch (CallTimeoutException& e) {
        std::cout << "call timeout:" <<  e.what() << std::endl;
    } catch (ServiceResponseException& e) {
        std::cout << "http status code:" << e.getStatusCode() << std::endl;
        std::cout << "error code:" << e.getErrorCode() << std::endl;
        std::cout << "error msg:" << e.getErrorMsg() << std::endl;
        std::cout << "RequestId:" << e.getRequestId() << std::endl;
    } catch (exception& e) {
        std:cout << "unknown exception:" << e.what() << std::endl;
    }
    std::cout << "------request finished--------" << std::endl;
    return0;
```

代码示例自动生成

API Explorer提供API检索及平台调试，支持全量快速检索、可视化调试、帮助文档查看和在线咨询。

您只需要在API Explorer中修改接口参数，即可自动生成对应的代码示例。

图 8-4 API Explorer



Region: 华北-北京四

Headers
X-Auth-Token: string

Parameters
project_id: 05041[redacted]

Body
```
{
  image_base64: 图片的base64编码
  attributes: string
}
```

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <memory>
#include <huaweicloud/core/exception/Exceptions.h>
#include <huaweicloud/core/Client.h>
#include <huaweicloud/frs/v2/FrsClient.h>

using namespace HuaweiCloud::Sdk::Frs::V2;
using namespace HuaweiCloud::Sdk::Frs::V2::Model;
using namespace HuaweiCloud::Sdk::Core;
using namespace HuaweiCloud::Sdk::Core::Exception;
using namespace std;

int main(void)
{
    string ak = "<YOUR AK>";
    string sk = "<YOUR SK>";
    string projectId = "05041[redacted]";
    string endpoint = "https://face.cn-north-4.myhuaweicloud.com";

    auto auth = std::make_unique<BasicCredentials>();
    auth->withAk(ak)
        .withSk(sk)
        .withProjectId(projectId);
    HttpConfig httpConfig = HttpConfig();
    auto client = FrsClient::newBuilder()
            .withCredentials(std::unique_ptr<Credentials>(auth.release()))
            .withHttpConfig(httpConfig)
            .withEndPoint(endpoint)
            .build();

    DetectFaceByBase64Request request;
    FaceDetectBase64Req body;
    body.setImageBase64("图片的base64编码");
    request.setBody(body);
```

A 修订记录

发布日期	修改说明
2021-10-21	新版SDK正式发布，旧版SDK停止维护。