# 人证核身服务

# SDK 参考

**文档版本** 01

**发布日期** 2024-11-04

**华为云计算技术有限公司**

### 商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

### 注意

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

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

## 华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

# 目 录



# 1 SDK 简介

人证核身服务端SDK是对人证核身提供的REST API进行的封装，您可以直接继承服务端从而实现对人证核身服务的快速操作。

## 接口与 API 对应关系

**表 1-1** 接口与 API 对应关系

| 接口 | API |
| --- | --- |
| 人证核身标准版（三要素） | POST https://{endpoint}/v2.0/ivs-standard |
| 人证核身证件版（二要素） | POST https://{endpoint}/v2.0/ivs-idcard-extention |

# 2 Java SDK

本章节介绍人证核身服务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。
- 已开通人证核身服务。如未开通，请登录**人证核身管理控制台**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


- 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

## 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-ivs”依赖项。请在**SDK中心**获取最新的sdk包版本，替换代码中版本。

```
// Maven配置，指定使用IVS服务所需的“huaweicloud-sdk-core”和“huaweicloud-sdk-ivs”依赖项
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-core</artifactId>
    <version>3.1.44</version>
</dependency>
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-ivs</artifactId>
    <version>3.1.44</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**。

## 开始使用 SDK

1. 导入依赖模块

```
import com.huaweicloud.sdk.core.auth.ICredential;
// 对用户身份进行认证
import com.huaweicloud.sdk.core.auth.BasicCredentials;
// 请求异常类
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;

// 导入ivs sdk
import com.huaweicloud.sdk.ivs.v2.region.IvsRegion;
import com.huaweicloud.sdk.ivs.v2.*;
import com.huaweicloud.sdk.ivs.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**。

```
// 创建AK、SK认证凭据
public static ICredential getCredential(String ak, String sk) {
    return new BasicCredentials()
            .withAk(ak)
            .withSk(sk);
}
```

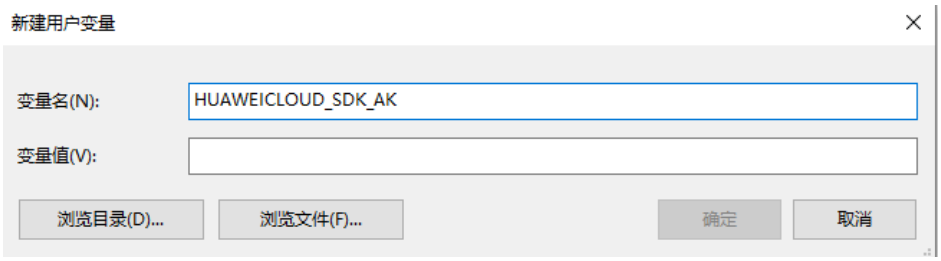
初始化认证信息：

```
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




3. 初始化客户端

   指定region方式

```
public static IvsClient getClient(Region region, ICredential auth) {
    // 初始化客户端
    return IvsClient.newBuilder()
            .withCredential(auth)
            .withRegion(region)
            .build();
}
```

   华北-北京四region获取：IvsRegion.CN_NORTH_4

   华北-北京一region获取：IvsRegion.CN_NORTH_1

4. 发送请求并查看响应

```
// 以调用标准版（三要素）DetectStandardByIdCardImage 接口为例
DetectStandardByIdCardImageRequest request = new DetectStandardByIdCardImageRequest();
DetectStandardByIdCardImageResponse response = client.detectStandardByIdCardImage(request);
System.out.println(response.toString());
```

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。 |
| | | 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。 |

| 一级分类 | 一级分类说明 | 二级分类 | 二级分类说明 |
| --- | --- | --- | --- |
| 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
DetectStandardByIdCardImageRequest request = new DetectStandardByIdCardImageRequest();
try {
    DetectStandardByIdCardImageResponse response = client.detectStandardByIdCardImage(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());
}
```

**说明**

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**。

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
DetectStandardByIdCardImageRequest request = new DetectStandardByIdCardImageRequest();
IvsStandardByIdCardImageRequestBody body = new IvsStandardByIdCardImageRequestBody();
List<ReqDataByIdCardImage> listIvsStandardByIdCardImageRequestBodyDataReqData = new
ArrayList<>();
listIvsStandardByIdCardImageRequestBodyDataReqData.add(
    new ReqDataByIdCardImage()
        .withIdcardImage1("身份证人像面图像数据,使用base64编码")
        .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
        .withFaceImage("现场人像图像数据,使用base64编码,")
);
IvsStandardByIdCardImageRequestBodyData databody = new
IvsStandardByIdCardImageRequestBodyData();
databody.withReqData(listIvsStandardByIdCardImageRequestBodyDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectStandardByIdCardImageResponse response = client.detectStandardByIdCardImage(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());
}
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
DetectStandardByNameAndIdRequest request = new DetectStandardByNameAndIdRequest();
IvsStandardByNameAndIdRequestBody body = new IvsStandardByNameAndIdRequestBody();
List<StandardReqDataByNameAndId> listIvsStandardByNameAndIdRequestBodyDataReqData = new
ArrayList<>();
listIvsStandardByNameAndIdRequestBodyDataReqData.add(
    new StandardReqDataByNameAndId()
        .withVerificationName("被验证人的姓名")
        .withVerificationId("被验证人的身份证号码")
        .withFaceImage("现场人像图像数据,使用base64编码")
);
IvsStandardByNameAndIdRequestBodyData databody = new
IvsStandardByNameAndIdRequestBodyData();
databody.withReqData(listIvsStandardByNameAndIdRequestBodyDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectStandardByNameAndIdResponse response = client.detectStandardByNameAndId(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());
}
```

- 方式三：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
DetectStandardByVideoAndNameAndIdRequest request = new
DetectStandardByVideoAndNameAndIdRequest();
IvsStandardByVideoAndNameAndIdRequestBody body = new
IvsStandardByVideoAndNameAndIdRequestBody();
List<StandardReqDataByVideoAndNameAndId> listDataReqData = new ArrayList<>();
listDataReqData.add(new StandardReqDataByVideoAndNameAndId()
    .withVerificationName("被验证人的姓名")
    .withVerificationId("被验证人的身份证号码")
    .withVideo("现场拍摄人像视频数据,使用base64编码")
    .withActions("动作代码顺序列表"));
IvsStandardByVideoAndNameAndIdRequestBodyData databody = new
IvsStandardByVideoAndNameAndIdRequestBodyData();
databody.withReqData(listDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e);
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectStandardByVideoAndNameAndIdResponse response =
client.detectStandardByVideoAndNameAndId(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.getRequestId());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

- 方式四：使用现场拍摄的人像视频数据，实现活体人证核身

```
DetectStandardByVideoAndIdCardImageRequest request = new
DetectStandardByVideoAndIdCardImageRequest();
IvsStandardByVideoAndIdCardImageRequestBody body = new
IvsStandardByVideoAndIdCardImageRequestBody();
List<ReqDataByVideoAndIdCardImage> listDataReqData = new ArrayList<>();
listDataReqData.add(
    new ReqDataByVideoAndIdCardImage()
        .withIdcardImage1("身份证人像面图像数据,使用base64编码")
        .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
        .withVideo("现场拍摄人像视频数据,使用base64编码")
        .withActions("动作代码顺序列表")
);
IvsStandardByVideoAndIdCardImageRequestBodyData databody = new
IvsStandardByVideoAndIdCardImageRequestBodyData();
databody.withReqData(listDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectStandardByVideoAndIdCardImageResponse response =
client.detectStandardByVideoAndIdCardImage(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.getRequestId());
    System.out.println(e.getErrorCode());
    System.out.println(e.getErrorMsg());
}
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
DetectExtentionByIdCardImageRequest request = new DetectExtentionByIdCardImageRequest();
IvsExtentionByIdCardImageRequestBody body = new IvsExtentionByIdCardImageRequestBody();
List<ExtentionReqDataByIdCardImage> listIvsExtentionByIdCardImageRequestBodyDataReqData =
new ArrayList<>();
listIvsExtentionByIdCardImageRequestBodyDataReqData.add(
    new ExtentionReqDataByIdCardImage()
        .withIdcardImage1("身份证人像面图像数据,使用base64编码")
        .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
);
IvsExtentionByIdCardImageRequestBodyData databody = new
IvsExtentionByIdCardImageRequestBodyData();
databody.withReqData(listIvsExtentionByIdCardImageRequestBodyDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectExtentionByIdCardImageResponse response = client.detectExtentionByIdCardImage(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());
}
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

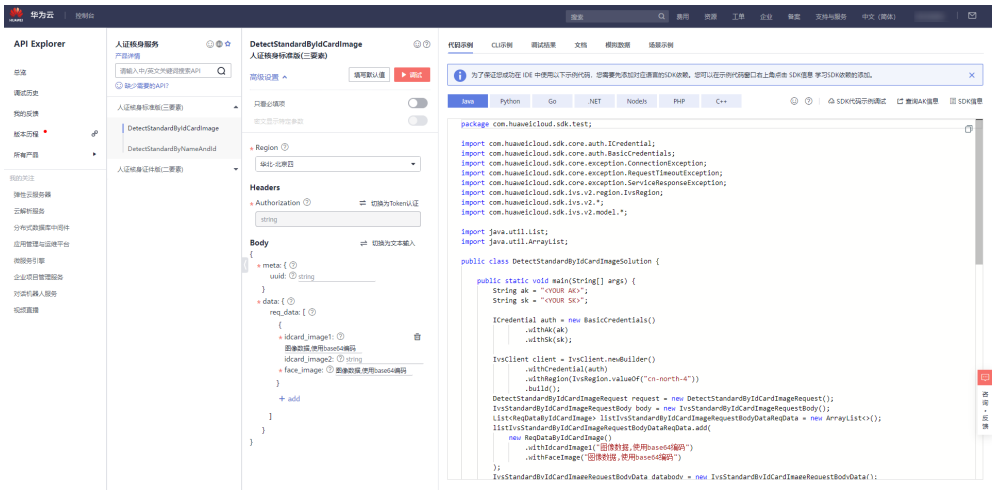
```
DetectExtentionByNameAndIdRequest request = new DetectExtentionByNameAndIdRequest();
IvsExtentionByNameAndIdRequestBody body = new IvsExtentionByNameAndIdRequestBody();
List<ExtentionReqDataByNameAndId> listIvsExtentionByNameAndIdRequestBodyDataReqData = new
ArrayList<>();
listIvsExtentionByNameAndIdRequestBodyDataReqData.add(
    new ExtentionReqDataByNameAndId()
        .withVerificationName("被验证人的姓名")
        .withVerificationId("被验证人的身份证号码")
);
IvsExtentionByNameAndIdRequestBodyData databody = new
IvsExtentionByNameAndIdRequestBodyData();
databody.withReqData(listIvsExtentionByNameAndIdRequestBodyDataReqData);
Meta metabody = new Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
try {
    DetectExtentionByNameAndIdResponse response = client.detectExtentionByNameAndId(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());
}
```

## 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。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



## 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**IVS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。

# 3 Python SDK

本章节介绍人证核身服务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。
- 已开通人证核身服务。如未开通，请登录**人证核身管理控制台**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


- 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

## 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“huaweicloudsdkivs”。

```
# 安装核心库
pip install huaweicloudsdkcore
# 安装 IVS 服务库
pip install huaweicloudsdkivs
```

## 开始使用 SDK

1. 导入依赖模块

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

# 导入IVS的库
from huaweicloudsdkivs.v2.region.ivs_region import IvsRegion
from huaweicloudsdkivs.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**。

```
// 创建AK、SK认证凭据
def GetCredential():
    return BasicCredentials(ak, sk)
```

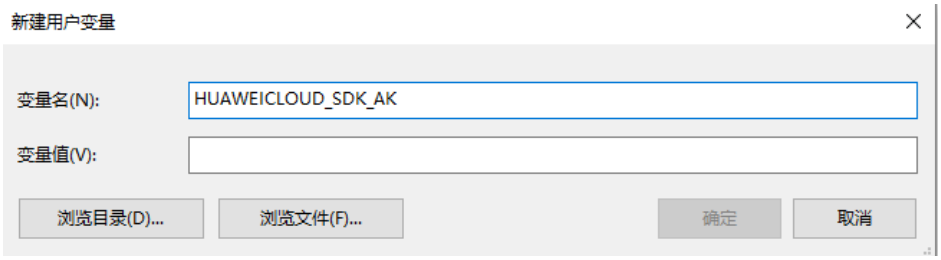
初始化认证信息：

```
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

```
# 选择服务部署区域
def GetClient():
    return IvsClient.new_builder(IvsClient) \
        .with_credentials(credentials) \
        .with_region(IvsRegion.CN_NORTH_4) \
        .build()
```

   CN_NORTH_4：华北-北京四

   CN_NORTH_1：华北-北京一

4. 发送请求并查看响应

```
# 以调用标准版(三要素)接口 DetectStandardByIdCardImage 为例
request = DetectStandardByIdCardImageRequest()
response = client.detect_standard_by_id_card_image(request)
print(response)
```

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) |

```
# 异常处理
try:
    request = DetectStandardByIdCardImageRequest()
    response = client.detect_standard_by_id_card_image(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

**说明**

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**。

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
try:
    request = DetectStandardByIdCardImageRequest()
    listReqDataByIdCardImageReqDataIvsStandardByIdCardImageRequestBodyData = [
        ReqDataByIdCardImage(
            idcard_image1="身份证人像面图像数据,使用base64编码",
            idcard_image2="身份证国徽面图像数据,使用base64编码",
            face_image="现场人像图像数据,使用base64编码,"
        )
    ]
    dataIvsStandardByIdCardImageRequestBodyData = IvsStandardByIdCardImageRequestBodyData(
        req_data=listReqDataByIdCardImageReqDataIvsStandardByIdCardImageRequestBodyData
    )
    metaMeta = Meta(
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsStandardByIdCardImageRequestBody(
        data=dataIvsStandardByIdCardImageRequestBodyData,
        meta=metaMeta
    )
    response = client.detect_standard_by_id_card_image(request)
    print(response)
 except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
try:
    request = DetectStandardByNameAndIdRequest()
    listStandardReqDataByNameAndIdReqDataIvsStandardByNameAndIdRequestBodyData = [
        StandardReqDataByNameAndId(
            verification_name="被验证人的姓名",
            verification_id="被验证人的身份证号码",
            face_image="现场人像图像数据,使用base64编码"
        )
    ]
    dataIvsStandardByNameAndIdRequestBodyData = IvsStandardByNameAndIdRequestBodyData(
        req_data=listStandardReqDataByNameAndIdReqDataIvsStandardByNameAndIdRequestBodyData
    )
     metaMeta = Meta(
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsStandardByNameAndIdRequestBody(
        data=dataIvsStandardByNameAndIdRequestBodyData,
        meta=metaMeta
    )
    response = client.detect_standard_by_name_and_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)
```

- 方式三：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
try:
    request = DetectStandardByVideoAndNameAndIdRequest()
    listReqDataData = [
        StandardReqDataByVideoAndNameAndId(
            verification_name="被验证人的姓名",
```

```
            verification_id="被验证人的身份证号码",
            video="现场拍摄人像视频数据,使用base64编码",
            actions="动作代码顺序列表"
        )
    ]
    databody = IvsStandardByVideoAndNameAndIdRequestBodyData(
        req_data=listReqDataData
    )
    metabody = Meta(
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsStandardByVideoAndNameAndIdRequestBody(
        data=databody,
        meta=metabody
    )
    response = client.detect_standard_by_video_and_name_and_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)
```

- 方式四：使用现场拍摄的人像视频数据，实现活体人证核身

```
try:
    request = DetectStandardByVideoAndIdCardImageRequest()
    listReqDataData = [
        ReqDataByVideoAndIdCardImage(
            idcard_image1="身份证人像面图像数据,使用base64编码",
            idcard_image2="身份证国徽面图像数据,使用base64编码",
            video="现场拍摄人像视频数据,使用base64编码",
            actions="动作代码顺序列表"
        )
    ]
    databody = IvsStandardByVideoAndIdCardImageRequestBodyData(
        req_data=listReqDataData
    )
    metabody = Meta(
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsStandardByVideoAndIdCardImageRequestBody(
        data=databody,
        meta=metabody
    )
    response = client.detect_standard_by_video_and_id_card_image(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
try:
    request = DetectExtentionByIdCardImageRequest()
    listExtentionReqDataByIdCardImageReqDataIvsExtentionByIdCardImageRequestBodyData = [
        ExtentionReqDataByIdCardImage(
            idcard_image1="身份证人像面图像数据,使用base64编码",
            idcard_image2="身份证国徽面图像数据,使用base64编码"
        )
    ]
    dataIvsExtentionByIdCardImageRequestBodyData = IvsExtentionByIdCardImageRequestBodyData(

req_data=listExtentionReqDataByIdCardImageReqDataIvsExtentionByIdCardImageRequestBodyData
    )
    metaMeta = Meta(
```

```
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsExtentionByIdCardImageRequestBody(
        data=dataIvsExtentionByIdCardImageRequestBodyData,
        meta=metaMeta
    )
    response = client.detect_extention_by_id_card_image(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

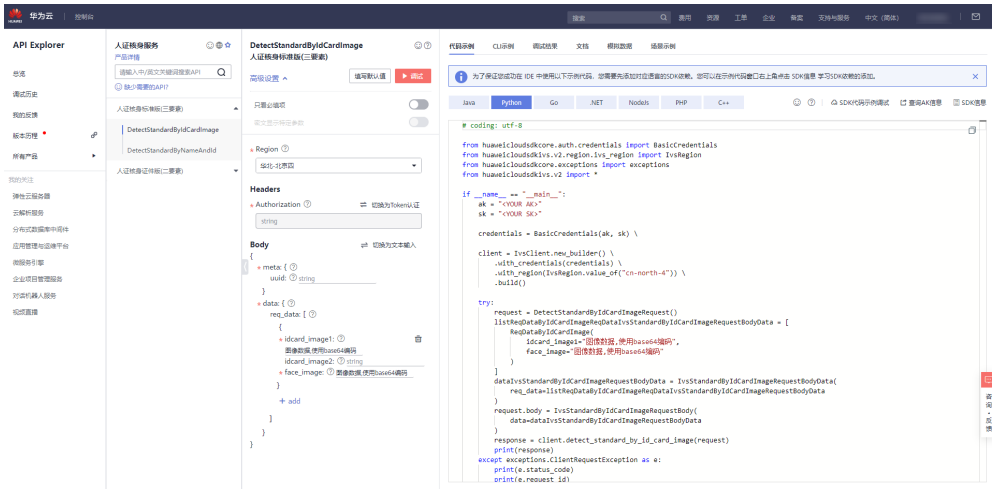
```
try:
    request = DetectExtentionByNameAndIdRequest()
    listExtentionReqDataByNameAndIdReqDataIvsExtentionByNameAndIdRequestBodyData = [
        ExtentionReqDataByNameAndId(
            verification_name="被验证人的姓名",
            verification_id="被验证人的身份证号码"
        )
    ]
    dataIvsExtentionByNameAndIdRequestBodyData = IvsExtentionByNameAndIdRequestBodyData(

req_data=listExtentionReqDataByNameAndIdReqDataIvsExtentionByNameAndIdRequestBodyData
    )
    metaMeta = Meta(
        uuid="唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
    )
    request.body = IvsExtentionByNameAndIdRequestBody(
        data=dataIvsExtentionByNameAndIdRequestBodyData,
        meta=metaMeta
    )
    response = client.detect_extention_by_name_and_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)
```

## 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。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



# 4 Go SDK

本章节介绍人证核身服务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。
- 已开通人证核身服务。如未开通，请登录**人证核身管理控制台**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


- 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

## 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 SDK

1. 导入依赖模块

```
import (
    "fmt"
        "os"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    // 导入IVS sdk
    ivs "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/ivs/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/ivs/v2/model"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/ivs/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**。

```
// 创建AK、SK认证凭据
func GetCredential(ak, sk string) basic.Credentials {
    return basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        Build()
}
```

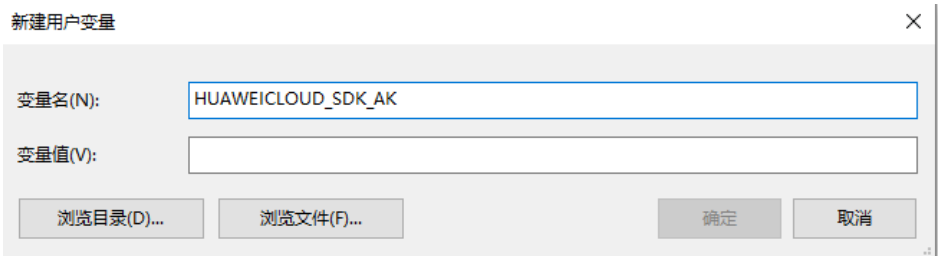
   初始化认证信息：

```
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



3. 初始化客户端

   指定region方式

```
// 选择服务部署区域
func GetClient(auth basic.Credentials) *ivs.IvsClient {
    return ivs.NewIvsClient(
        ivs.IvsClientBuilder().
            WithRegion(region.CN_NORTH_4).
            WithCredential(auth).
            Build())
}
```

   CN_NORTH_4：华北-北京四

   CN_NORTH_1：华北-北京一

4. 发送请求并查看响应

```
# 以调用标准版(三要素)接口 DetectStandardByIdCardImage 为例
request := &model.DetectStandardByIdCardImageRequest{}
response, err := client.DetectStandardByIdCardImage(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

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 |

```
// 异常处理
response, err := client.DetectStandardByIdCardImage(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

**说明**

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

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
request := &model.DetectStandardByIdCardImageRequest{}
idcardImage2ReqDataReqDataByIdCardImage:= "身份证国徽面图像数据,使用base64编码"
var listReqDataIvsStandardByIdCardImageRequestBodyData = []model.ReqDataByIdCardImage{
    {
        IdcardImage1: "身份证人像面图像数据,使用base64编码",
        IdcardImage2: &idcardImage2ReqDataReqDataByIdCardImage,
        FaceImage: "现场人像图像数据,使用base64编码,",
    },
}
databody := &model.IvsStandardByIdCardImageRequestBodyData{
    ReqData: &listReqDataIvsStandardByIdCardImageRequestBodyData,
```

```
}
uuidMetaMeta:= "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMetaMeta,
}
request.Body = &model.IvsStandardByIdCardImageRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectStandardByIdCardImage(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
request := &model.DetectStandardByNameAndIdRequest{}
var listReqDataIvsStandardByNameAndIdRequestBodyData = []model.StandardReqDataByNameAndId{
    {
        VerificationName: "被验证人的姓名",
        VerificationId: "被验证人的身份证号码",
        FaceImage: "现场人像图像数据,使用base64编码",
    },
}
databody := &model.IvsStandardByNameAndIdRequestBodyData{
    ReqData: &listReqDataIvsStandardByNameAndIdRequestBodyData,
}
uuidMetaMeta:= "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMetaMeta,
}
request.Body = &model.IvsStandardByNameAndIdRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectStandardByNameAndId(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 方式三：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
request := &model.DetectStandardByVideoAndNameAndIdRequest{}
var listReqDataData = []model.StandardReqDataByVideoAndNameAndId{
    {
        VerificationName: "被验证人的姓名",
        VerificationId:   "被验证人的身份证号码",
        Video:            "现场拍摄人像视频数据,使用base64编码",
        Actions:          "动作代码顺序列表",
    },
}
databody := &model.IvsStandardByVideoAndNameAndIdRequestBodyData{
    ReqData: &listReqDataData,
}
uuidMeta := "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMeta,
}
request.Body = &model.IvsStandardByVideoAndNameAndIdRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectStandardByVideoAndNameAndId(request)
if err == nil {
```

```
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 方式四：使用现场拍摄的人像视频数据，实现活体人证核身

```
request := &model.DetectStandardByVideoAndIdCardImageRequest{}
idcardImage2ReqData:= "身份证国徽面图像数据,使用base64编码"
var listReqDataData = []model.ReqDataByVideoAndIdCardImage{
    {
        IdcardImage1: "身份证人像面图像数据,使用base64编码",
        IdcardImage2: &idcardImage2ReqData,
        Video: "现场拍摄人像视频数据,使用base64编码",
        Actions: "动作代码顺序列表",
    },
}
databody := &model.IvsStandardByVideoAndIdCardImageRequestBodyData{
    ReqData: &listReqDataData,
}
uuidMeta:= "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMeta,
}
request.Body = &model.IvsStandardByVideoAndIdCardImageRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectStandardByVideoAndIdCardImage(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
request := &model.DetectExtentionByIdCardImageRequest{}
idcardImage2ReqDataExtentionReqDataByIdCardImage:= "身份证国徽面图像数据,使用base64编码"
var listReqDataIvsExtentionByIdCardImageRequestBodyData =
[]model.ExtentionReqDataByIdCardImage{
    {
        IdcardImage1: "身份证人像面图像数据,使用base64编码",
        IdcardImage2: &idcardImage2ReqDataExtentionReqDataByIdCardImage,
    },
}
databody := &model.IvsExtentionByIdCardImageRequestBodyData{
    ReqData: &listReqDataIvsExtentionByIdCardImageRequestBodyData,
}
uuidMetaMeta:= "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMetaMeta,
}
request.Body = &model.IvsExtentionByIdCardImageRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectExtentionByIdCardImage(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

```
request := &model.DetectExtentionByNameAndIdRequest{}
var listReqDataIvsExtentionByNameAndIdRequestBodyData =
[]model.ExtentionReqDataByNameAndId{
```

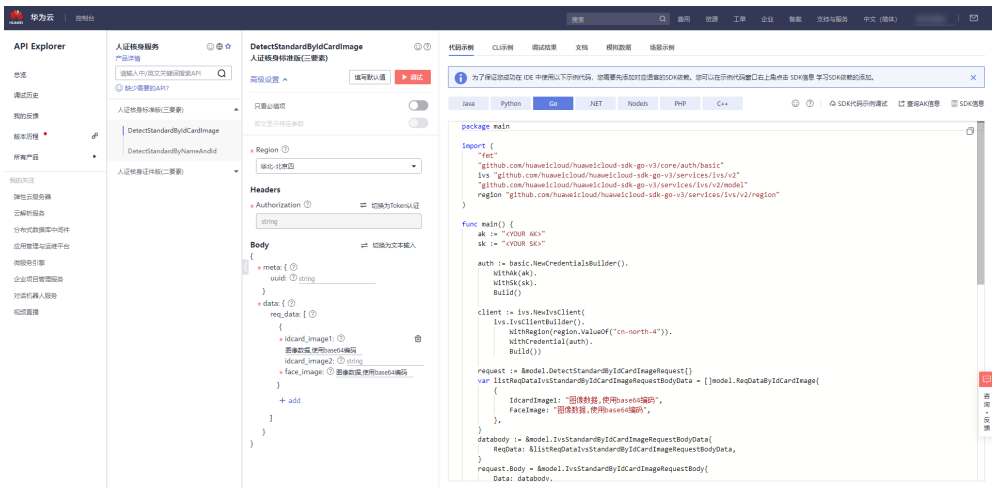
```
    {
        VerificationName: "被验证人的姓名",
        VerificationId: "被验证人的身份证号码",
    },
}
databody := &model.IvsExtentionByNameAndIdRequestBodyData{
    ReqData: &listReqDataIvsExtentionByNameAndIdRequestBodyData,
}
uuidMetaMeta:= "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e"
metabody := &model.Meta{
    Uuid: &uuidMetaMeta,
}
request.Body = &model.IvsExtentionByNameAndIdRequestBody{
    Data: databody,
    Meta: metabody,
}
response, err := client.DetectExtentionByNameAndId(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

## 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。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



# 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


- 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

## 安装 SDK

使用SDK前，需要安装“HuaweiCloud.SDK.Core”和“HuaweiCloud.SDK.Ivs”，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.Ivs
```
- 使用 Package Manager
```
Install-Package HuaweiCloud.SDK.Core
Install-Package HuaweiCloud.SDK.Ivs
```

## 开始使用 SDK

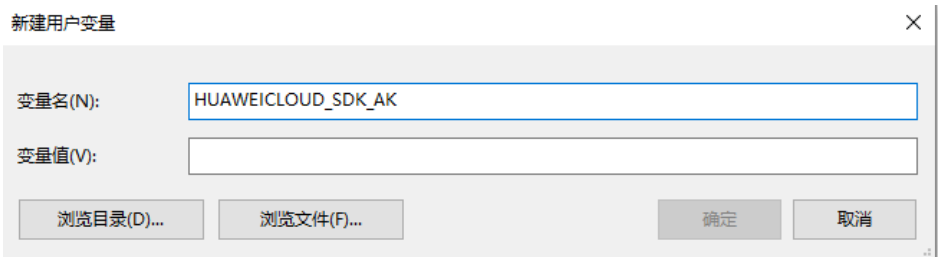
1. 导入依赖模块
```
using System;
using System.Collections.Generic;
using HuaweiCloud.SDK.Core;
using HuaweiCloud.SDK.Core.Auth;
using HuaweiCloud.SDK.Ivs;
using HuaweiCloud.SDK.Ivs.V2;
using HuaweiCloud.SDK.Ivs.V2.Model;
```
2. 配置客户端连接参数
   - 默认配置
   ```
   // 使用默认配置
   var config = HttpConfig.GetDefaultConfig();
   ```
   - 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




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IVS服务的 IvsClient 为例
     var client = IvsClient.NewBuilder()
             .WithCredential(auth)
             .WithRegion(IvsRegion.ValueOf("cn-north-4"))
             .WithHttpConfig(config)
             .Build();
     ```

   - 指定云服务endpoint方式

     ```
     // 指定终端节点，以IVS服务北京四的 endpoint 为例
     String endpoint = "https://ivs.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IVS服务的 IvsClient 为例
     var client = IvsClient.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

   ```
   // 以调用人证核身标准版（三要素）接口 DetectStandardByIdCardImage 为例
   var req = new DetectStandardByIdCardImageRequest
   {
   };
   List<ReqDataByIdCardImage> listReqDataByIdCardImageReqData = new
   List<ReqDataByIdCardImage>();
   listReqDataByIdCardImageReqData.Add(new ReqDataByIdCardImage()
   {
       IdcardImage1 = "身份证人像面图像数据,使用base64编码",
       IdcardImage2 = "身份证国徽面图像数据,使用base64编码",
       FaceImage = "现场人像图像数据,使用base64编码"
   });
   IvsStandardByIdCardImageRequestBodyData databody = new
   IvsStandardByIdCardImageRequestBodyData()
   {
       ReqData = listReqDataByIdCardImageReqData
   };
   Meta metabody = new Meta()
   {
       Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。"
   };
   req.Body = new IvsStandardByIdCardImageRequestBody()
   {
       Data = databody,
       Meta = metabody
   };

   try
   {
       var resp = client.DetectStandardByIdCardImage(req);
   ```

```
    var respStatusCode = resp.HttpStatusCode;
    Console.WriteLine(respStatusCode);
}
```

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.DetectStandardByIdCardImage(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);
        }
```

**说明**

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**。

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
var req = new DetectStandardByIdCardImageRequest
{
};
List<ReqDataByIdCardImage> listReqDataByIdCardImageReqData = new
List<ReqDataByIdCardImage>();
listReqDataByIdCardImageReqData.Add(new ReqDataByIdCardImage()
{
    IdcardImage1 = "身份证人像面图像数据,使用base64编码",
    IdcardImage2 = "身份证国徽面图像数据,使用base64编码",
    FaceImage = "现场人像图像数据,使用base64编码,"
});
IvsStandardByIdCardImageRequestBodyData databody = new
IvsStandardByIdCardImageRequestBodyData()
{
    ReqData = listReqDataByIdCardImageReqData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
};
req.Body = new IvsStandardByIdCardImageRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
{
    var resp = client.DetectStandardByIdCardImage(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);
}
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
var req = new DetectStandardByNameAndIdRequest
{
};
List<StandardReqDataByNameAndId> listStandardReqDataByNameAndIdReqData = new
List<StandardReqDataByNameAndId>();
listStandardReqDataByNameAndIdReqData.Add(new StandardReqDataByNameAndId()
{
    VerificationName = "被验证人的姓名",
    VerificationId = "被验证人的身份证号码",
    FaceImage = "现场人像图像数据,使用base64编码"
});
IvsStandardByNameAndIdRequestBodyData databody = new
IvsStandardByNameAndIdRequestBodyData()
{
    ReqData = listStandardReqDataByNameAndIdReqData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
```

```
cbc884f1e20e"
};
req.Body = new IvsStandardByNameAndIdRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
{
    var resp = client.DetectStandardByNameAndId(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);
}
```

- 方式三：使用现场拍摄的人像视频数据，实现活体人证核身

```
var req = new DetectStandardByVideoAndIdCardImageRequest
{
};
List<ReqDataByVideoAndIdCardImage> listReqDataData = new
List<ReqDataByVideoAndIdCardImage>();
listReqDataData.Add(new ReqDataByVideoAndIdCardImage()
{
    IdcardImage1 = "身份证人像面图像数据,使用base64编码",
    IdcardImage2 = "身份证国徽面图像数据,使用base64编码",
    Video = "现场拍摄人像视频数据,使用base64编码",
    Actions = "动作代码顺序列表"
});
IvsStandardByVideoAndIdCardImageRequestBodyData databody = new
IvsStandardByVideoAndIdCardImageRequestBodyData()
{
    ReqData = listReqDataData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
};
req.Body = new IvsStandardByVideoAndIdCardImageRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
{
    var resp = client.DetectStandardByVideoAndIdCardImage(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.RequestId);
    Console.WriteLine(clientRequestException.ErrorCode);
    Console.WriteLine(clientRequestException.ErrorMsg);
}
catch (ConnectionException connectionException)
{
    Console.WriteLine(connectionException.ErrorMessage);
}
```

- 方式四：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
var req = new DetectStandardByVideoAndNameAndIdRequest
{
};
List<StandardReqDataByVideoAndNameAndId> listReqDataData = new
List<StandardReqDataByVideoAndNameAndId>();
listReqDataData.Add(new StandardReqDataByVideoAndNameAndId()
{
    VerificationName = "被验证人的姓名",
    VerificationId = "被验证人的身份证号码",
    Video = "现场拍摄人像视频数据,使用base64编码",
    Actions = "动作代码顺序列表"
});
IvsStandardByVideoAndNameAndIdRequestBodyData databody = new
IvsStandardByVideoAndNameAndIdRequestBodyData()
{
    ReqData = listReqDataData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
};
req.Body = new IvsStandardByVideoAndNameAndIdRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
{
    var resp = client.DetectStandardByVideoAndNameAndId(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.RequestId);
    Console.WriteLine(clientRequestException.ErrorCode);
    Console.WriteLine(clientRequestException.ErrorMsg);
}
catch (ConnectionException connectionException)
{
    Console.WriteLine(connectionException.ErrorMessage);
}
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
var req = new DetectExtentionByIdCardImageRequest
{
};
List<ExtentionReqDataByIdCardImage> listExtentionReqDataByIdCardImageReqData = new
List<ExtentionReqDataByIdCardImage>();
```

```
listExtentionReqDataByIdCardImageReqData.Add(new ExtentionReqDataByIdCardImage()
{
    IdcardImage1 = "身份证人像面图像数据,使用base64编码",
    IdcardImage2 = "身份证国徽面图像数据,使用base64编码"
});
IvsExtentionByIdCardImageRequestBodyData databody = new
IvsExtentionByIdCardImageRequestBodyData()
{
    ReqData = listExtentionReqDataByIdCardImageReqData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
};
req.Body = new IvsExtentionByIdCardImageRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
{
    var resp = client.DetectExtentionByIdCardImage(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);
}
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

```
var req = new DetectExtentionByNameAndIdRequest
{
};
List<ExtentionReqDataByNameAndId> listExtentionReqDataByNameAndIdReqData = new
List<ExtentionReqDataByNameAndId>();
listExtentionReqDataByNameAndIdReqData.Add(new ExtentionReqDataByNameAndId()
{
    VerificationName = "被验证人的姓名",
    VerificationId = "被验证人的身份证号码"
});
IvsExtentionByNameAndIdRequestBodyData databody = new
IvsExtentionByNameAndIdRequestBodyData()
{
    ReqData = listExtentionReqDataByNameAndIdReqData
};
Meta metabody = new Meta()
{
    Uuid = "唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e"
};
req.Body = new IvsExtentionByNameAndIdRequestBody()
{
    Data = databody,
    Meta = metabody
};

try
```

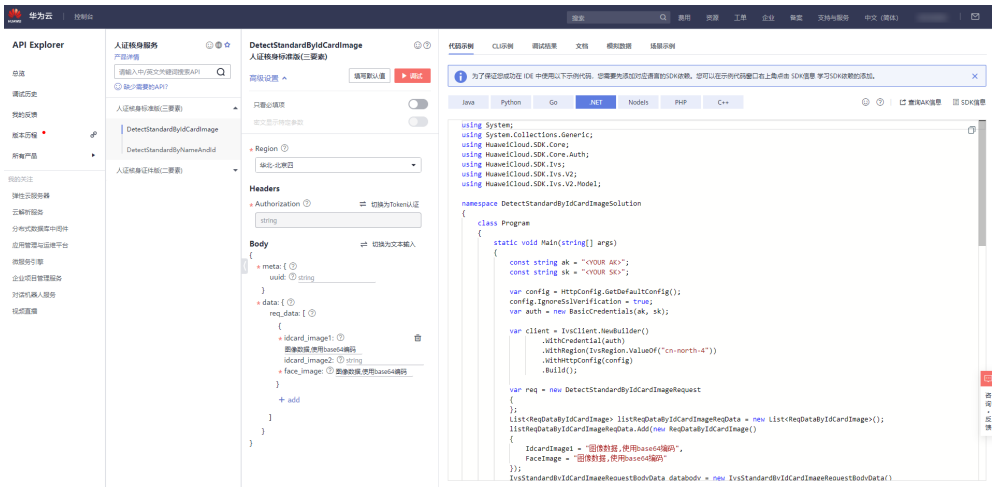
```
{
    var resp = client.DetectExtentionByNameAndId(req);
    var respStatusCode = resp.HttpStatusCode;
    Console.WriteLine(respStatusCode);
}
catch (RequestTimeoutException requestTimeoutException)
{
    Console.WriteLine(requestTimeoutException.ErrorMessage);
}
catch (ServiceResponseException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

## 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。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 5-4** API Explorer



# 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

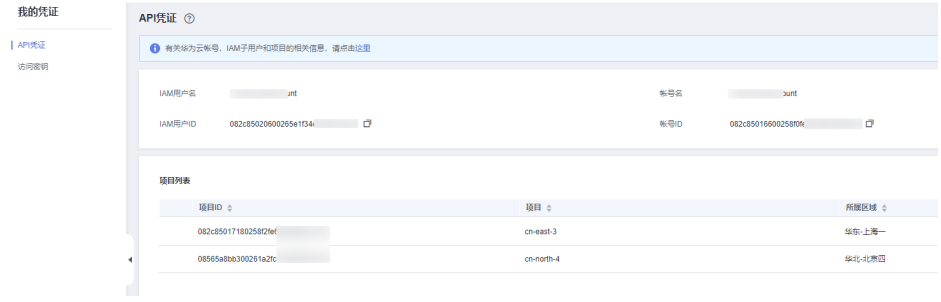
- **注册华为账号并开通华为云**，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，支持Node 10.16.1 及其以上版本。
- 登录“**我的凭证** > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

**图 6-1** 获取 AK、SK


- 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



## 安装 SDK

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

推荐您使用 npm 安装 SDK。

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

## 开始使用 SDK

1. 导入依赖模块
   ```
   const core = require('@huaweicloud/huaweicloud-sdk-core');
   const ivs = require("@huaweicloud/huaweicloud-sdk-ivs");
   ```
2. 配置客户端链接参数
   - 默认配置
     ```
     const client = ivs.IvsClient.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

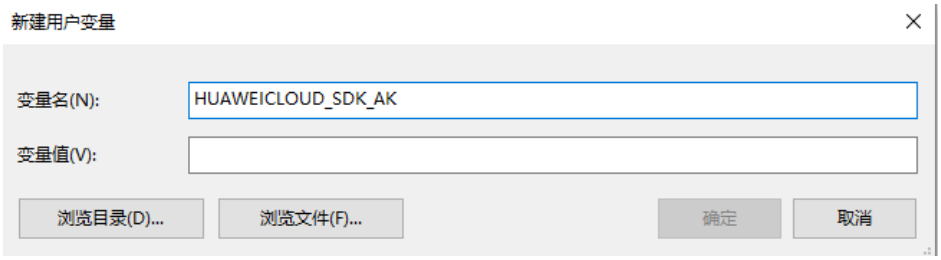
   配置AK、SK、project_id信息。华为云通过AK识别用户的身份，通过SK对请求数据进行签名验证，用于确保请求的机密性、完整性和请求者身份的正确性。AK、SK和project_id获取方法请参见**准备工作**

   ```
   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



4. 初始化客户端

   指定云服务endpoint方式

```
// 指定终端节点，以 IVS 服务北京四的 endpoint 为例
const endpoint = "https://ivs.cn-north-4.myhuaweicloud.com";
const client = ivs.IvsClient.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

```
// 以调用人证核身标准版（三要素）接口 DetectStandardByIdCardImage 为例
const request = new ivs.DetectStandardByIdCardImageRequest();
const body = new ivs.IvsStandardByIdCardImageRequestBody();
const listIvsStandardByIdCardImageRequestBodyDataReqData = newArray();
listIvsStandardByIdCardImageRequestBodyDataReqData.push(
new ivs.ReqDataByIdCardImage()
    .withIdcardImage1("身份证人像面图像数据,使用base64编码")
    .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
    .withFaceImage("现场人像图像数据,使用base64编码")
);
const databody = new ivs.IvsStandardByIdCardImageRequestBodyData();
databody.withReqData(listIvsStandardByIdCardImageRequestBodyDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectStandardByIdCardImage(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

**说明**

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**。

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
const request = new ivs.DetectStandardByIdCardImageRequest();
const body = new ivs.IvsStandardByIdCardImageRequestBody();
const listIvsStandardByIdCardImageRequestBodyDataReqData = newArray();
listIvsStandardByIdCardImageRequestBodyDataReqData.push(
new ivs.ReqDataByIdCardImage()
    .withIdcardImage1("身份证人像面图像数据,使用base64编码")
    .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
    .withFaceImage("现场人像图像数据,使用base64编码,")
);
const databody = new ivs.IvsStandardByIdCardImageRequestBodyData();
databody.withReqData(listIvsStandardByIdCardImageRequestBodyDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectStandardByIdCardImage(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
```

```
    console.log("exception:" + JSON.stringify(ex));
});
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
const request = new ivs.DetectStandardByNameAndIdRequest();
const body = new ivs.IvsStandardByNameAndIdRequestBody();
const listIvsStandardByNameAndIdRequestBodyDataReqData = newArray();
listIvsStandardByNameAndIdRequestBodyDataReqData.push(
new ivs.StandardReqDataByNameAndId()
      .withVerificationName("被验证人的姓名")
      .withVerificationId("被验证人的身份证号码")
      .withFaceImage("现场人像图像数据,使用base64编码")
);
const databody = new ivs.IvsStandardByNameAndIdRequestBodyData();
databody.withReqData(listIvsStandardByNameAndIdRequestBodyDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectStandardByNameAndId(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 方式三：使用现场拍摄的人像视频数据，实现活体人证核身

```
const request = new ivs.DetectStandardByVideoAndIdCardImageRequest();
const body = new ivs.IvsStandardByVideoAndIdCardImageRequestBody();
const listDataReqData = new Array();
listDataReqData.push(
 new ivs.ReqDataByVideoAndIdCardImage()
      .withIdcardImage1("身份证人像面图像数据,使用base64编码")
      .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
      .withVideo("现场拍摄人像视频数据,使用base64编码")
      .withActions("动作代码顺序列表")
);
const databody = new ivs.IvsStandardByVideoAndIdCardImageRequestBodyData();
databody.withReqData(listDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectStandardByVideoAndIdCardImage(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 方式四：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
const request = new ivs.DetectStandardByVideoAndNameAndIdRequest();
const body = new ivs.IvsStandardByVideoAndNameAndIdRequestBody();
const listDataReqData = new Array();
listDataReqData.push(
 new ivs.StandardReqDataByVideoAndNameAndId()
      .withVerificationName("被验证人的姓名")
      .withVerificationId("被验证人的身份证号码")
      .withVideo("现场拍摄人像视频数据,使用base64编码")
      .withActions("动作代码顺序列表")
);
const databody = new ivs.IvsStandardByVideoAndNameAndIdRequestBodyData();
databody.withReqData(listDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
```

```
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectStandardByVideoAndNameAndId(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
const request = new ivs.DetectExtentionByIdCardImageRequest();
const body = new ivs.IvsExtentionByIdCardImageRequestBody();
const listIvsExtentionByIdCardImageRequestBodyDataReqData = newArray();
listIvsExtentionByIdCardImageRequestBodyDataReqData.push(
new ivs.ExtentionReqDataByIdCardImage()
      .withIdcardImage1("身份证人像面图像数据,使用base64编码")
      .withIdcardImage2("身份证国徽面图像数据,使用base64编码")
);
const databody = new ivs.IvsExtentionByIdCardImageRequestBodyData();
databody.withReqData(listIvsExtentionByIdCardImageRequestBodyDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectExtentionByIdCardImage(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

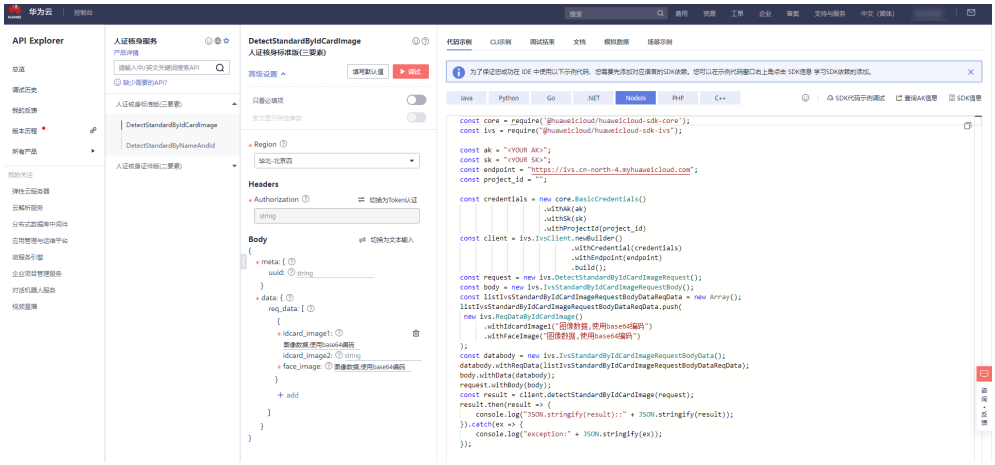
```
const request = new ivs.DetectExtentionByNameAndIdRequest();
const body = new ivs.IvsExtentionByNameAndIdRequestBody();
const listIvsExtentionByNameAndIdRequestBodyDataReqData = newArray();
listIvsExtentionByNameAndIdRequestBodyDataReqData.push(
new ivs.ExtentionReqDataByNameAndId()
      .withVerificationName("被验证人的姓名")
      .withVerificationId("被验证人的身份证号码")
);
const databody = new ivs.IvsExtentionByNameAndIdRequestBodyData();
databody.withReqData(listIvsExtentionByNameAndIdRequestBodyDataReqData);
const metabody = new ivs.Meta();
metabody.withUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.withData(databody);
body.withMeta(metabody);
request.withBody(body);
const result = client.detectExtentionByNameAndId(request);
result.then(result => {
    console.log("JSON.stringify(result)::" + JSON.stringify(result));
}).catch(ex => {
    console.log("exception:" + JSON.stringify(ex));
});
```

## 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。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



# 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


- 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

## 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 SDK

1. 导入依赖模块

   ```
   <?php
   namespace HuaweiCloud\SDK\Ivs\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\Ivs\V2\IvsClient;
   ```

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 IVS 服务北京四的 endpoint 为例
   $ak = getenv('HUAWEICLOUD_SDK_AK');
   $sk = getenv('HUAWEICLOUD_SDK_SK');
   $endpoint = "https://ivs.cn-north-4.myhuaweicloud.com";
   $projectId = getenv('PROJECT_ID');
   $credentials = new BasicCredentials($ak,$sk,$projectId);
   ```

   认证参数说明：

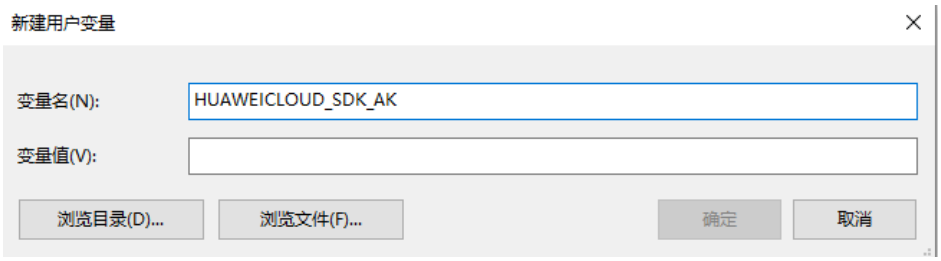
   - ak、sk：访问秘钥信息，获取方法请参见**准备工作**。
   - projectId：华为云项目ID，获取方法请参见**准备工作**。

- 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




4. 初始化客户端

   指定云服务endpoint方式

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

5. 发送并查看响应

```
// 以调用人证核身标准版（三要素）接口 DetectStandardByIdCardImage 为例
$request = new DetectStandardByIdCardImageRequest();
$body = new IvsStandardByIdCardImageRequestBody();
$listIvsStandardByIdCardImageRequestBodyDataReqData = array();
array_push($listIvsStandardByIdCardImageRequestBodyDataReqData,(new ReqDataByIdCardImage())
    ->setIdcardImage1("身份证人像面图像数据,使用base64编码")
    ->setIdcardImage2("身份证国徽面图像数据,使用base64编码")
    ->setFaceImage("现场人像图像数据,使用base64编码")
 );
$databody = new IvsStandardByIdCardImageRequestBodyData();
$databody->setReqData($listIvsStandardByIdCardImageRequestBodyDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectStandardByIdCardImage($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";
```

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 = $client->DetectStandardByNameAndId($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**

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

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
$request = new DetectStandardByIdCardImageRequest();
$body = new IvsStandardByIdCardImageRequestBody();
$listIvsStandardByIdCardImageRequestBodyDataReqData = array();
```

```
array_push($listIvsStandardByIdCardImageRequestBodyDataReqData,(new ReqDataByIdCardImage())
    ->setIdcardImage1("身份证人像面图像数据,使用base64编码")
    ->setIdcardImage2("身份证国徽面图像数据,使用base64编码")
    ->setFaceImage("现场人像图像数据,使用base64编码,")
 );
$databody = new IvsStandardByIdCardImageRequestBodyData();
$databody->setReqData($listIvsStandardByIdCardImageRequestBodyDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectStandardByIdCardImage($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;
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
$request = new DetectStandardByNameAndIdRequest();
$body = new IvsStandardByNameAndIdRequestBody();
$listIvsStandardByNameAndIdRequestBodyDataReqData = array();
array_push($listIvsStandardByNameAndIdRequestBodyDataReqData,(new
StandardReqDataByNameAndId())
    ->setVerificationName("被验证人的姓名")
    ->setVerificationId("被验证人的身份证号码")
    ->setFaceImage("现场人像图像数据,使用base64编码")
 );
$databody = new IvsStandardByNameAndIdRequestBodyData();
$databody->setReqData($listIvsStandardByNameAndIdRequestBodyDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectStandardByNameAndId($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;
```

- 方式三：使用现场拍摄的人像视频数据，实现活体人证核身

```
$request = new DetectStandardByVideoAndIdCardImageRequest();
$body = new IvsStandardByVideoAndIdCardImageRequestBody();
$listDataReqData = array();
array_push($listDataReqData,(new ReqDataByVideoAndIdCardImage())
```

```
    ->setIdcardImage1("身份证人像面图像数据,使用base64编码")
    ->setIdcardImage2("身份证国徽面图像数据,使用base64编码")
    ->setVideo("现场拍摄人像视频数据,使用base64编码")
    ->setActions("动作代码顺序列表")
 );
$databody = new IvsStandardByVideoAndIdCardImageRequestBodyData();
$databody->setReqData($listDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectStandardByVideoAndIdCardImage($request);
  echo "\n";
  echo $response;
} 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->getRequestId(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
```

- 方式四：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
$request = new DetectStandardByVideoAndNameAndIdRequest();
$body = new IvsStandardByVideoAndNameAndIdRequestBody();
$listDataReqData = array();
array_push($listDataReqData,(new StandardReqDataByVideoAndNameAndId())
    ->setVerificationName("被验证人的姓名")
    ->setVerificationId("被验证人的身份证号码")
    ->setVideo("现场拍摄人像视频数据,使用base64编码")
    ->setActions("动作代码顺序列表")
 );
$databody = new IvsStandardByVideoAndNameAndIdRequestBodyData();
$databody->setReqData($listDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectStandardByVideoAndNameAndId($request);
  echo "\n";
  echo $response;
} 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->getRequestId(). "\n";
  echo $e->getErrorCode() . "\n";
  echo $e->getErrorMsg() . "\n";
}
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
$request = new DetectExtentionByIdCardImageRequest();
$body = new IvsExtentionByIdCardImageRequestBody();
$listIvsExtentionByIdCardImageRequestBodyDataReqData = array();
array_push($listIvsExtentionByIdCardImageRequestBodyDataReqData,
(newExtentionReqDataByIdCardImage())
    ->setIdcardImage1("身份证人像面图像数据,使用base64编码")
    ->setIdcardImage2("身份证国徽面图像数据,使用base64编码")
 );
$databody = new IvsExtentionByIdCardImageRequestBodyData();
$databody->setReqData($listIvsExtentionByIdCardImageRequestBodyDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectExtentionByIdCardImage($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;
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

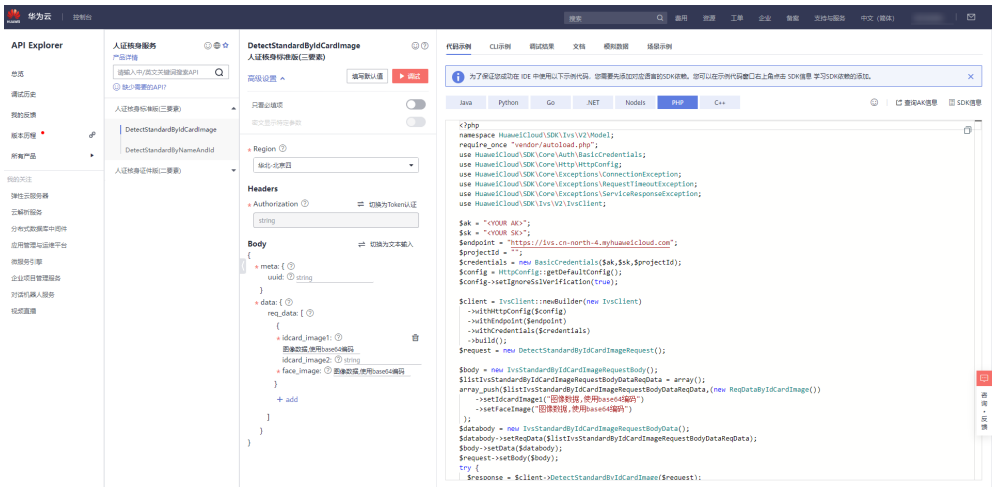
```
$request = new DetectExtentionByNameAndIdRequest();
$body = new IvsExtentionByNameAndIdRequestBody();
$listIvsExtentionByNameAndIdRequestBodyDataReqData = array();
array_push($listIvsExtentionByNameAndIdRequestBodyDataReqData,
(newExtentionReqDataByNameAndId())
    ->setVerificationName("被验证人的姓名")
    ->setVerificationId("被验证人的身份证号码")
 );
$databody = new IvsExtentionByNameAndIdRequestBodyData();
$databody->setReqData($listIvsExtentionByNameAndIdRequestBodyDataReqData);
$metabody = new Meta();
$metabody->setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
$body->setData($databody);
$body->setMeta($metabody);
$request->setBody($body);
try {
  $response = $client->DetectExtentionByNameAndId($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;
```

## 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。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 7-4** API Explorer



# 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

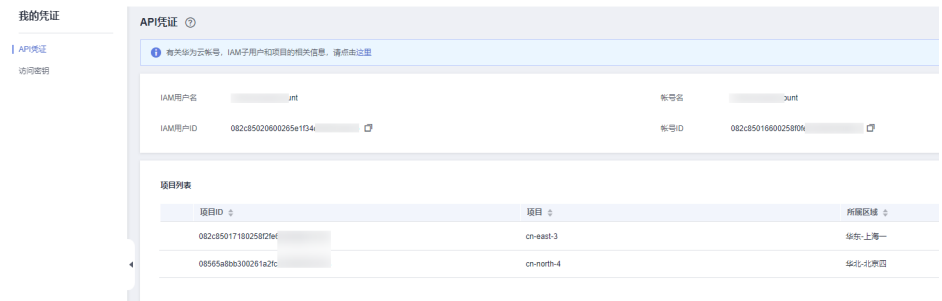
- **注册华为账号并开通华为云**，并完成实名认证，账号不能处于欠费或冻结状态。
- 已具备开发环境，支持 C++ 14 及以上版本，要求安装 CMake 3.10 及以上版本。
- 登录“**我的凭证** > 访问秘钥”页面，获取Access Key（AK）和Secret Access Key（SK）。

**图 8-1** 获取 AK、SK


- 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



## 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 SDK

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/ivs/v2/IvsClient.h>
   using namespace HuaweiCloud::Sdk::Ivs::V2;
   using namespace HuaweiCloud::Sdk::Ivs::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);
   ```

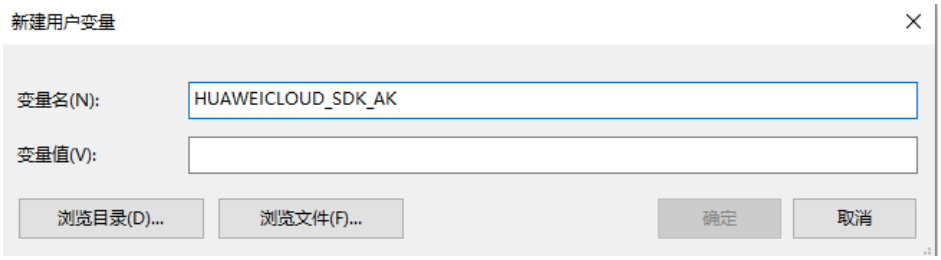
   认证参数说明：

   - 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

4. 初始化客户端

指定云服务endpoint方式

```
string endpoint = "https://ivs.cn-north-4.myhuaweicloud.com";
auto client = IvsClient::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

```
// 以调用人证核身标准版（三要素）接口 DetectStandardByIdCardImage 为例
DetectStandardByIdCardImageRequest request;
IvsStandardByIdCardImageRequestBody body;
std::vector<ReqDataByIdCardImage> listIvsStandardByIdCardImageRequestBodyDataReqData;
ReqDataByIdCardImage objReqData;
objReqData.setIdcardImage1("身份证人像面图像数据,使用base64编码");
objReqData.setIdcardImage2("身份证国徽面图像数据,使用base64编码");
objReqData.setFaceImage("现场人像图像数据,使用base64编码");
listIvsStandardByIdCardImageRequestBodyDataReqData.push_back(ReqDataByIdCardImage(objReqDat
a));
IvsStandardByIdCardImageRequestBodyData bodyData;
bodyIvsStandardByIdCardImageRequestBodyData.setReqData(listIvsStandardByIdCardImageRequestBo
dyDataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);
std::cout << "-----begin execute request-------" << std::endl;
```

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->detectStandardByNameAndId(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;
```

**说明**

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**。

## SDK demo 代码解析

### 人证核身标准版（三要素）

- 方式一：使用身份证图片、人像图片进行校验

```
DetectStandardByIdCardImageRequest request;
IvsStandardByIdCardImageRequestBody body;
std::vector<ReqDataByIdCardImage> listIvsStandardByIdCardImageRequestBodyDataReqData;
ReqDataByIdCardImage objReqData;
objReqData.setIdcardImage1("身份证人像面图像数据,使用base64编码");
objReqData.setIdcardImage2("身份证国徽面图像数据,使用base64编码");
objReqData.setFaceImage("现场人像图像数据,使用base64编码,");
listIvsStandardByIdCardImageRequestBodyDataReqData.push_back(ReqDataByIdCardImage(objReqDat
a));
IvsStandardByIdCardImageRequestBodyData bodyData;
bodyIvsStandardByIdCardImageRequestBodyData.setReqData(listIvsStandardByIdCardImageRequestBo
dyDataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);
std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectStandardByIdCardImage(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;
```

- 方式二：使用身份证姓名、身份证号码文本，人像图片进行校验

```
DetectStandardByNameAndIdRequest request;
IvsStandardByNameAndIdRequestBody body;
std::vector<StandardReqDataByNameAndId> listIvsStandardByNameAndIdRequestBodyDataReqData;
StandardReqDataByNameAndId objReqData;
objReqData.setVerificationName("被验证人的姓名");
objReqData.setVerificationId("被验证人的身份证号码");
objReqData.setFaceImage("现场人像图像数据,使用base64编码");
listIvsStandardByNameAndIdRequestBodyDataReqData.push_back(StandardReqDataByNameAndId(obj
ReqData));
IvsStandardByNameAndIdRequestBodyData bodyData;
bodyIvsStandardByNameAndIdRequestBodyData.setReqData(listIvsStandardByNameAndIdRequestBody
DataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);
std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectStandardByNameAndId(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;
```

- 方式三：使用现场拍摄的人像视频数据，实现活体人证核身

```
DetectStandardByVideoAndIdCardImageRequest request;
IvsStandardByVideoAndIdCardImageRequestBody body;
std::vector<ReqDataByVideoAndIdCardImage> listDataReqData;
ReqDataByVideoAndIdCardImage objReqData;
objReqData.setIdcardImage1("身份证人像面图像数据,使用base64编码");
objReqData.setIdcardImage2("身份证国徽面图像数据,使用base64编码");
objReqData.setVideo("现场拍摄人像视频数据,使用base64编码");
objReqData.setActions("动作代码顺序列表");
listDataReqData.push_back(ReqDataByVideoAndIdCardImage(objReqData));
IvsStandardByVideoAndIdCardImageRequestBodyData bodyData;
bodyData.setReqData(listDataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);

std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectStandardByVideoAndIdCardImage(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;
```

- 方式四：使用身份证姓名、身份证号码文本和现场拍摄的人像视频数据，实现活体人证核身

```
DetectStandardByVideoAndNameAndIdRequest request;
IvsStandardByVideoAndNameAndIdRequestBody body;
std::vector<StandardReqDataByVideoAndNameAndId> listDataReqData;
StandardReqDataByVideoAndNameAndId objReqData;
objReqData.setVerificationName("被验证人的姓名");
objReqData.setVerificationId("被验证人的身份证号码");
objReqData.setVideo("现场拍摄人像视频数据,使用base64编码");
objReqData.setActions("动作代码顺序列表");
listDataReqData.push_back(StandardReqDataByVideoAndNameAndId(objReqData));
IvsStandardByVideoAndNameAndIdRequestBodyData bodyData;
bodyData.setReqData(listDataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-
cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);

std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectStandardByVideoAndNameAndId(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;
```

**人证核身证件版（二要素）**

- 方式一：使用身份证图片进行校验

```
DetectExtentionByIdCardImageRequest request;
IvsExtentionByIdCardImageRequestBody body;
std::vector<ExtentionReqDataByIdCardImage>
listIvsExtentionByIdCardImageRequestBodyDataReqData;
ExtentionReqDataByIdCardImage objReqData;
objReqData.setIdcardImage1("身份证人像面图像数据,使用base64编码");
objReqData.setIdcardImage2("身份证国徽面图像数据,使用base64编码");
listIvsExtentionByIdCardImageRequestBodyDataReqData.push_back(ExtentionReqDataByIdCardImage(
objReqData));
IvsExtentionByIdCardImageRequestBodyData bodyData;
bodyIvsExtentionByIdCardImageRequestBodyData.setReqData(listIvsExtentionByIdCardImageRequestB
odyDataReqData);
```

```
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);

std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectExtentionByIdCardImage(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;
```

- 方式二：使用身份证姓名、身份证号码文本进行校验

```
DetectExtentionByNameAndIdRequest request;
IvsExtentionByNameAndIdRequestBody body;
std::vector<ExtentionReqDataByNameAndId> listIvsExtentionByNameAndIdRequestBodyDataReqData;
ExtentionReqDataByNameAndId objReqData;
objReqData.setVerificationName("被验证人的姓名");
objReqData.setVerificationId("被验证人的身份证号码");
listIvsExtentionByNameAndIdRequestBodyDataReqData.push_back(ExtentionReqDataByNameAndId(objReqData));
IvsExtentionByNameAndIdRequestBodyData bodyData;
bodyIvsExtentionByNameAndIdRequestBodyData.setReqData(listIvsExtentionByNameAndIdRequestBodyDataReqData);
Meta bodyMeta;
bodyMeta.setUuid("唯一标识此次请求的ID,用户自定义,不超过64位。例如10eb0091-887f-4839-9929-cbc884f1e20e");
body.setData(bodyData);
body.setMeta(bodyMeta);
request.setBody(body);

std::cout << "-----begin execute request-------" << std::endl;
try {
    auto reponse = client->detectExtentionByNameAndId(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;
```

## 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。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 8-4** API Explorer