二要素+证件照/大头照身份验证接口[612]

1.1 接口介绍

传入姓名，身份证证件号码，证件照或者大头照，验证二要素信息，同时验证传入的证件照或大头照与真实证件照是否为同一人。

1.2 接口信息

1.2.1接口地址

名称	内容
生产地址	https://api.situdata.com/situ-identity/v2/id-verify/two-element-face

1.2.2 接口格式

名称	内容
请求方式	POST
返回类型	Json
编码格式	UTF-8

1.2.3 接口 Header

名称	类型	必填	内容
x-access-id	String	是	客户账号 ID（如无请联系商务）
x-signature	String	是	客户校签码，计算方法为 HmacSHA256(x-access-key + x-request-send-timestamp)，服务端根据 signature 来验证请求 的合法性，校签方法示

			例代码请见示例代码 generateSignature 方法。
x-request-send-timestamp	Int	是	客户发送请求时间。linux epoch，单位（秒）（注意：如果与服务端收到请求的时间相差 5 分钟则被判为无效。
Content-Type	String	是	application/json

1.2.4 接口 Body 请求参数

名称		类型	必须	描述
name		String	是	姓名
cardId		String	是	证件号码
imageType		String	是	图片类型: card-证件照 face-大头照
imageRotate		Boolean	否	true-开启图片方向校正 false-不开启图片方向校正 如果不传，默认 false
imageInfo	**imageContent**	String	是	图片 base64 码

1.2.5 接口返回参数

参数名		类型	描述
globalRequestId		String	请求 ID
rtn		int	返回状态码：0 为正常（详见附录）
msg		String	附加结果说明
result	**code**	int	返回状态码：1-二要素一致且人脸对比一

			致， 2-二要素不一致 3-身份证信息已注销或库中查无此号 4-照片对比不一致 5-证件照不存在
	desc	String	结果描述

1.2.6 接口请求和返回示例：

内容名称	内容

请求	{
	"name":"黄耿嘉",
	"cardId":"445202199002108316",
	"imageType":"card",
	"imageInfo":{
	"imageContent":"imageBase64"
	}
	}

返回 1	{
	"msg": "成功！",
	"rtn": 0,
	"globalRequestId": "600x15536005160274506RB",
	"result": {
	"code": 1,
	"desc": ":信息一致，人证成功"
	}
	}

返回 2	{ "msg": "json 无法解析", "rtn": 1000,

	"globalRequestId": "600x15536005160274506RB", "result": "null" }

1.2.7接口请求示例：Java

```
import com.alibaba.fastjson.JSONObject;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import sun.misc.BASE64Encoder;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.HashMap;

public class TwoElementFaceTest {

    private static String url = "接口 url";

    private static String accessId = "您的 accessId ";

    private static String accessKey = "您的 accessKey ";

    public static void main(String[] args) {
        // 您的图片路径，例如:"C:\\Users\\admin\\Desktop\\xxx.jpg"
        String path = "您的图片路径";
        String name = "姓名";
        String cardId = "证件号";
        // 您的证件类型,card 或 face
        String imageType = "证件类型";
        String imageBase64 = imageToBase64(path);
        HashMap paramMap = new HashMap<String, Object>() {
            {
                put("name", name);
                put("cardId", cardId);
                put("imageType", imageType);
```

```
                put("imageInfo", new HashMap<String, Object>() {
                    {
                        put("imageContent", imageBase64);
                    }
                });
            }
        };
        HashMap<String, String> header = new HashMap<>();
        long timeMillis = System.currentTimeMillis() / 1000L - 10;
        String signature = generateSignature(accessId, accessKey, timeMillis);
        // 设置通用的请求属性,设置 accessId 和 signature
        header.put("accept", "*/*");
        header.put("connection", "Keep-Alive");
        header.put("Content-Type", "application/json");
        header.put("x-access-id", accessId);
        header.put("x-request-send-timestamp", timeMillis    + "");
        header.put("x-signature", signature);
        String paramJson = JSONObject.toJSONString(paramMap);
        String res = null;
        try {
            res = doPost(url, paramJson, header);
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println(res);
    }

    private static String imageToBase64(String path) {
        InputStream in = null;
        byte[] data = null;
        // 读取图片字节数组
        try {
            in = new FileInputStream(path);
            data = new byte[in.available()];
            in.read(data);
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        // 对字节数组 Base64 编码
        BASE64Encoder encoder = new BASE64Encoder();
        // 返回 Base64 编码过的字节数组字符串，并去除其中换行
        return encoder.encode(data).replaceAll("\r\n", "").replaceAll("\n", "");
    }

    private static String generateSignature(String accessId, String base64Key, long timestamp) {
        String accessKey = new String(Base64.getDecoder().decode(base64Key));
        String signKey = accessId + "-" + timestamp;
        try {
```

```
            Mac hmacSha256 = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new SecretKeySpec(accessKey.getBytes(), "HmacSHA256");
            hmacSha256.init(secretKey);
            return Base64.getEncoder().encodeToString(hmacSha256.doFinal(signKey.getBytes()));
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
            return null;
        }
    }

    /**
     * POST 请求,JSON 请求方式
     */
    private static String doPost(String url, String json, HashMap<String, String> header) throws IOException {
        // 创建 Httpclient 对象
        CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = null;
        String resultString = "";
        try {
            // 创建 Http Post 请求
            HttpPost httpPost = new HttpPost(url);
            // 添加请求头
            if (null != header) {
                for (String key : header.keySet()) {
                    httpPost.addHeader(key, header.get(key));
                }
            }
            // json 方式发送 post 请求
            StringEntity entity = new StringEntity(json, "utf-8");// 解决中文乱码问题
            entity.setContentEncoding("UTF-8");
            entity.setContentType("application/json");
            httpPost.setEntity(entity);
            // 执行 http 请求
            response = httpClient.execute(httpPost);
            // 判断返回状态是否为 200
            if (200 == response.getStatusLine().getStatusCode()) {
                resultString = EntityUtils.toString(response.getEntity(), "UTF-8");
            }
        } finally {
            try {
                if (null != response) {
                    response.close();
                }
                httpClient.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return resultString;
```

```
        }

    }
```