技术服务-身份人像比对接口文档

日期	版本	编写人	说明
2019/11/19	V1.0	大白梨	创建文档
2022/12/29	V1.1	MCC	更新中心人脸分数阈值

一、接口介绍

技术服务-身份人像比对接口，该 API 主要用于对身份信息进行核验及用户图像与公安底图进行比对；

二、通用说明

1、请求地址：

https://api.apetime.cn/api/identity/portrait_verification

2、请求方式：

POST

3、格式为 JPG（JPEG），BMP，PNG，GIF，TIFF

4、小于等于 2 MB，建议 32KB 以内

5、照片要尽量清晰，无模糊点，避免戴眼镜，不能出现多人，人脸的左右上下角度不 能超过 15°、照片避免逆光、照片不能过暗，尽量接近证件照标准。

6、时间戳：

timestamp 取 GMT1970 年 1 月 1 日 00:00:00 到现在所经历的毫秒数

7、签名规则：

1、非必输字段不参与签名

2、selfie_file 和 selfie_base64 不参与签名

3、将签名字段的字段值按字母序排列，然后将他们对应的值拼接在一起

4、将 app_secret 拼接到上一步产生的字符串后面（app_secret 由平台提供）

5、利用 md5 得到签名字段

6、具体拼接规则如下：

app_key+id_number+name+request_id+timestamp+app_secret

将各参数对应的值拼接在一起

三、接口及参数说明

1、请求参数：

字段	类型	必需	描述
app_key	*string*	是	客户应用标识，由平台提供
request_id	*string*	是	客户请求流水号，建议唯一性
id_number	*string*	是	身份证号。用以查询近照
name	*string*	是	与身份证号相对应的姓名。身份证号及姓名相匹配才能查询近照
selfie_file	*file*	见下方注释	需上传的图片文件。上传本地图片可选取此参数
selfie_base64	*string*	见下方注释	采用 base64 编码的二进制图片数据可选此参数

字段	类型	必需	描述
timestamp	*long*	是	时间戳
sign	*string*	是	签名

参数 selfie_file,与 selfie_base64 必二选一，如同时传入多个参数，本 API 使用顺序为 selfie_base64、selfie_file。

参数 selfie_file 需把图片文件的内容以 multipart/form-data 的形式放到 POST 消息体中。

参数 selfie_base64 需要注意：图片的 base64 编码是不包含图片头的，如 data:image/jpg;base64。

2、返回参数

字段	类型	说明
trace_id	*string*	本次请求的唯一流水号
code	*string*	业务响应码
msg	*string*	说明
data	object	消息体（只有 0000 调用成功时返回）
charge	*int*	是否收费，1 收费，2 不收费

data 详情：

字段	类型	说明	字段
result	int	查询结果	1 查得 2 图片质量校验不合格
id_name	*int*	身份核验 结果	1 姓名身份证号一致 2 姓名身份证号不一致
photo	*int*	图片校验 结果	1 比对成功 2 库中无照片 3 特征提取失败
score	double	比对得分	系统判断为不同人：[0,40) 不确定是否为同一人:[40,60) 系统判断为同一人：[60,100]

说明：当 result 为 1 时，会返回 id_name，当 id_name 为 1 时，会返回 photo，当 photo 为 1 时，会返回 score；

返回样例

查询成功，返回全部结果


```
{
  "trace_id": "TID2af751b4bff24be781d60af10bf84101",
  "code": "0000",
  "msg": "调用成功",
  "data": {
```

```
    "result":1,
    "id_name":01,
    "photo":01,
    "score":50,
}
    "charge": 1
}
```


返回部分结果（库中无照片）


```
{
    "trace_id": "TID2af751b4bff24be781d60af10bf84101",
    "code": "0000",
    "msg": "调用成功",
    "data": {
    "result":1,
    "id_name":1,
    "photo":2,
}
    "charge": 1
}
```


返回部分结果（二要素比对不一致）


```
{
    "trace_id": "TID2af751b4bff24be781d60af10bf84101",
```

```
    "code": "0000",
    "msg": "调用成功",
    "data": {
    "result":1,
    "id_name":2,
}
    "charge": 1
}
```


返回部分结果（图片前置校验不通过）


```
{
    "trace_id": "TID2af751b4bff24be781d60af10bf84101",
    "code": "0000",
    "msg": "调用成功",
    "data": {
    "result":2,
}
    "charge": 2
}
```


查询失败


```
{
    "trace_id": "TID2af751b4bff24be781d60af10bf84101",
    "code": "1000",
```

"msg": "验签失败",

"charge": 2

}

3、响应码

code	msg
0000	调用成功
1000	验签失败
1001	参数非 UTF-8 编码
1002	请求参数错误
1003	商户冻结
1004	商户停用
1005	商户账号过期
1006	调用频率过高
1007	免费次数用尽
1008	服务无权限
1009	App_key 验证失败

1010	请求过期
2000	供应商服务异常
9999	其他错误

4、收费标准

以返回字段中的 charge 为准，当 charge 为 1 时收费，为 2 不收费。

四、代码示例

```
import java.util.Base64;
import org.apache.commons.codec.binary.Hex;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.MessageDigest;
import java.util.UUID;

public class PortraitVerifyDemo {

    private static String CHARSET = "UTF-8";
    private static String URL =
"https://api.apetime.cn/api/identity/portrait_verification";
    //连接超时 1000ms
    private static int CONNECT_TIMEOUT = 1000;
    //读超时 2000ms
    private static int READ_TIMEOUT = 3000;

    private static String APP_KEY = "";

    private static String APP_SECRET = "";

    private static String name = "";
```

```
    private static String id_number = "";

    private static String portrait_path = "";

    public static void main(String[] args) throws Exception {

        Long timestamp = System.currentTimeMillis();

        String requestId = UUID.randomUUID().toString().replaceAll("-", "");
        byte[] portrait = Files.readAllBytes(Paths.get(portrait_path));

        StringBuilder requestBuilder = new StringBuilder();

        requestBuilder.append("app_key=").append(APP_KEY);
        requestBuilder.append("&request_id=").append(requestId);
        requestBuilder.append("&name=").append(name);
        requestBuilder.append("&id_number=").append(id_number);
        requestBuilder.append("&timestamp=").append(timestamp);
        requestBuilder.append("&selfie_base64=").append(Base64.getUrlEncod
er().encodeToString(portrait));

        StringBuilder signBuilder = new StringBuilder();
        signBuilder.append(APP_KEY);
        signBuilder.append(id_number);
        signBuilder.append(name);
        signBuilder.append(requestId);
        signBuilder.append(timestamp);
        signBuilder.append(APP_SECRET);
        MessageDigest digest = MessageDigest.getInstance("MD5");
        String sign =
Hex.encodeHexString(digest.digest(signBuilder.toString().getBytes(CHARSE
T)));
        requestBuilder.append("&sign=").append(sign);
        BufferedReader br = null;
        HttpURLConnection conn = null;
        String responseStr = null;

        String requestStr = requestBuilder.toString();
        try {
            URL obj = new URL(URL);
            conn = (HttpURLConnection) obj.openConnection();
            conn.setRequestMethod("POST");
            conn.setConnectTimeout(CONNECT_TIMEOUT);
```

```
            conn.setReadTimeout(READ_TIMEOUT);
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.getOutputStream().write(requestStr.getBytes(CHARSET));
            conn.connect();
            br = new BufferedReader(new
InputStreamReader(conn.getInputStream(), CHARSET));
            StringBuffer response = new StringBuffer();
            String line;
            while ((line = br.readLine()) != null) {
                response.append(line);
            }
            responseStr = response.toString();
        } finally {
            if (br != null) {
                br.close();
            }
            if (conn != null) {
                conn.disconnect();
            }
        }
        System.out.println(responseStr);
    }

}
```