人脸识别开放平台规范

Bioauth OpenAPI

VERSION: V2.0

平安科技人工智能实验室

一. API 简介

1. 接口能力

接口名称	接口能力简要描述
人脸检测	检测图片中人脸，返后人脸列表
人脸比对	返回两两比对的人脸相似值
活体识别	检测图片中人脸是否为活体
身份证去网纹	分析身份证照片网纹分布并去网纹，返回无网纹照片

2. 接入流程

用户创建 OpenAPI 相关应用后，人脸识别开放平台会为该应用分配一对密钥，包含了 AppID 和 AppKey，用户可以通过该密钥生成签名字符串（生成方法参考鉴权校验部分），开放平台会通过签名来验证请求的合法性。

3. 返回格式

JSON 格式

4. 请求限制

请求图片采用 base64 编码，格式支持 PNG、JPG、JPEG、BMP 格式，不支持 GIF 图片。

接口名称	图片编码后大小限额
人脸检测	单次请求只允许上传单张图片，图片小于 2M
人脸比对	单次请求允许传入的两张图片，单张图片小于 2M

活体识别	单次请求只允许上传单张图片，图片小于 2M
身份证去网纹	单次请求只允许上传单张图片，图片小于 2M

5. 调用方式

人脸识别 OpenAPI2.0 是 RESTful API，用户通过向服务器发起 HTTP 请求，服务器会对请求进行处理，结果将会返回给用户。每个 API 调用需要根据需求传入不同的参数，具体参考各个接口的请求参数部分，所有 API 的调用都需要使用 HTTP POST 请求。

6. 请求头域

用户所有请求的 HTTP 头域中需要包含以下信息：

参数名	值	描述
Host	如:e.ai.pingan.com.cn	服务器域名地址
Authorization	鉴权签名	生成方式见“鉴权校验”
Content-Length	包体总长度	请求包体总长度，单位：字节
Content-Type	application/json	请求格式为 json
x-mx-timestamp	1476414567061	时间戳
x-mx-appid	6222980101038034	业务方申请的 appid

具体示例：

```
POST http://e.ai.pingan.com.cn/openapi/face/v2/detect
Host: e.ai.pingan.com.cn
Authorization: 1EBB9F6ACF40F66B30A31CF9C61C022A7FC876BOF0016FAEB
Content-Type: application/json
Content-Length: 30720
x-mx-timestamp: 1476414567061
x-mx-appid: 6222980101038034
```

7. 鉴权校验

(1).生成鉴权字符串：

```
key = “bioauth”+appkey
content = timestamp()
Sign = HMAC-SHA256(key, content)
```

其中 appkey 为 appid 对应的秘钥串，通过 HMAC-SHA256 生成的鉴权字符串 Sign，并赋值给请求头域的 Authorization 字段。

(2).生成图片摘要：

```
key = “bioauth” + appkey
content = Base64_encode(imge_data)
token = HMAC-SHA256(key, content)
```

生成的图片摘要需赋值给请求包体中 token 字段。

8. 接口返回码

当前接口返回码遵循 HTTP 状态码的含义及规则，各个状态码说明及错误描述如下：

HTTP 状态码 200

Error Code	Error Msg	说明
0	ok	调用成功

HTTP 状态码 400

Error Code	Error Msg	说明
1	request loss header field	请求头域缺省必要字段
2	request body has empty field	请求体存在空字段
3	request body to json failed	请求体转换 json 格式失败
4	request data exception	请求数据异常

HTTP 状态码 401

Error Code	Error Msg	说明
5	signature verification failed	签名验证失败
6	token verification failed	图片 token 验证失败

返回码 404，Not Found，调用的 API 不存在，请求失败。

HTTP 状态码 413

Error Code	Error Msg	说明
7	request entity too large	请求包体太大(限定 10M)
8	image size greater than 2M	单张图片大于 2M

HTTP 状态码 415

Error Code	Error Msg	说明
9	image data is empty	图片数据为空
10	image parse exception	图片解析异常

HTTP 状态码 417

Error Code	Error Msg	说明
11	face not found	图片不含人脸

二．人脸检测接口

该接口接收客户端发送的图片数据，对图片中的人脸进行检测，返回检测结果。

1. 接口地址

```
http://<server_address> + /openapi/face/v2/detect
server address example：e.ai.pingan.com.cn
```

2. 请求参数

Request body 参数说明

	参数名	类型	参数说明
必须	app_id	String	业务接入时生成的唯一 id
必须	terminal	Obj(TermInfo)	终端设备信息(见 TermInfo 表说明)
必须	person_id	String	唯一身份 id
必须	image	Obj(ImageInfo)	图片对象(见 ImageInfo 表说明)
必须	mutipleFace	Boolean	返回人脸数量设置(False-返回单个人脸，True-返回设定的最大人脸数量 MaxNum=16，默认为 False)

TermInfo 参数说明

参数名	类型	参数说明
type	String	标识终端设备类型，如“iphone7”
system	String	标识终端设备使用的软件系统，如“ios10.0”
sdk	String	标识终端设备人脸识别 sdk 版本，目前设定为“o236”

ImageInfo 参数说明

参数名	类型	参数说明
category	Int	标识图片类别(1-手机自拍照片，2-身份证照片，3-护照照片，默认 1)
content_type	String	图片格式，例如 jpg，png。
data	String	待检测的图片内容经过 base64 编码后的字符串

token	String	鉴权校验部分生成的图片摘要

3. 返回参数

Response body 参数说明

字段	类型	说明
errorcode	String	错误码
errormsg	String	错误信息描述
faces	List	被检测出的人脸列表

faces 列表单个元素参数说明

字段	类型			说明
face	rect	x	Int	表示人脸轮廓左上角 x 坐标
		y	Int	人脸轮廓左上角 y 坐标
		width	Int	人脸轮廓宽带
		height	Int	人脸轮廓高度

4. 返回消息示例

```
{
    "errorcode":"0",
    "errormsg":"ok",
    "faces":[
              {"face":{
                    "rect":{
                          "x":45,
                          "y":60,
                          "width":75,
                          "height":75
                          }
                    }
              }
```

```
        ]
}
```

三．人脸比对接口

该接口接收客户端发送的 2 类图片数据（如：生活照和证件照），检测分析图片里的人脸相似性，并执行比对，返回相似度，可用于判断两张脸是否是同一人的可能性大小。

1. 接口地址

```
http://<server_address> + /openapi/face/v2/compare
server address example：e.ai.pingan.com.cn
```

2. 请求参数

Request body 参数说明

	参数名	类型	参数说明
必须	app_id	String	业务接入时生成的唯一 id
必须	terminal	Obj(TermInfo)	终端设备信息
必须	person_id	String	唯一身份 id
必须	image1	Obj(ImageInfo)	待对比的图片 1
必须	image2	Obj(ImageInfo)	待对比的图片 2

3. 返回参数

Response body 参数说明

字段	类型	说明

errorcode	String	错误码
errormsg	String	错误信息描述
similarity	String	人脸比对分数
ref_thres	String	参考阀值

4. 返回消息示例

```
{
    "similarity":"0.75",
    "ref_thres":"0.497",
    "errorcode":"0",
    "errormsg":"ok"
}
```

四．活体识别接口

该接口接收客户端发送的图片数据，检测分析图片里的人脸是否活体，返回检测分析结果。

1. 接口地址

```
http://<server_address> + /openapi/face/v2/biodetect
server address example：e.ai.pingan.com.cn
```

2. 请求参数

Request body 参数说明

	参数名	类型	参数说明
必须	app_id	String	业务接入时生成的唯一 id
必须	terminal	Obj(TermInfo)	终端设备信息
必须	person_id	String	唯一身份 id

必须	image	Obj(ImageInfo)	待活体检测的图片对象

3. 返回参数

Response body 参数说明

字段	类型	说明
errorcode	String	错误码
errormsg	String	错误信息描述
bio_score	String	图片活体识别的分数
time	String	响应时间
face_detect_result	String	图片中是否有人脸（“Have face”, “No face”）
is_alive	String	活体检测结果（“true”, “false”）

4. 返回消息示例

```
{
      "bio_score":"4.89565",
      "time":"20160422003458",
      "face_detect_result":"Have face",
      "is_alive":"true",
      "errorcode":"0",
      "errormsg":"ok"
}
```

五．身份证去网纹接口

该接口接收客户端发送的身份证照，分析身份证照中的网纹分布并还原无网纹照，返回无网纹照片。

1. 接口地址

```
http://<server_address> + /openapi/face/v2/erasemark
```

```
server address example：e.ai.pingan.com.cn
```

2. 请求参数

Request body 参数表

	参数名	类型	参数说明
必须	app_id	String	业务接入时生成的唯一 id
必须	terminal	Obj(TermInfo)	终端设备信息
必须	person_id	String	唯一身份 id
必须	image	Obj(ImageInfo)	待去网纹的图片对象

3. 返回参数

Response body 参数表

字段	类型		说明
errorcode	String		错误码
errormsg	String		错误信息描述
time	String		参考阀值
image	conten_type	String	去网纹后图片格式
	data	String	去网纹后图片经 base64 编码后的字符串

4. 返回消息示例

```
{
    "time":"20160422003458",
    "image":{
```

```
        "content_type":"jpg",
        "data":" ICAgICAgICAKICFUSUYBUFNEAUlDTyBJPA8KIICA..."
    },
    "errorcode":"0",
    "errormsg":"ok"
}
```

六．代码示例

1. Python 调用样例

以人脸比对 API 为例，Python 版本客户端调用样例如下（其中包含了签名字符串和图片 token 的生成）：

```
#coding=utf-8
import os,re,sys,shutil
import json
import hashlib
import base64
import time
import random
import binascii
import hmac
import urlib2
from hashlib import sha256
reload(sys)
sys.setdefaultencoding('utf8')

#服务器地址,如'http://e.ai.pingan.com.cn:81'
kHost = 'http://192.168.100.26:81'
kUrl = kHost + '/openapi/face/v2/compare'

#密钥
kBoundId = '6222980101038034'
boundinfo={"6222980101038034":"1234567890ABCDEFG"}

#生成签名字符串
def request_signature(timestamp, boundid):
    encode_content = timestamp
    key = str("bioauth" + boundinfo[boundid]).encode('utf-8')
    my_hmac = hmac.new(key, encode_content, sha256)
    return my_hmac.hexdigest().upper()

#生成图片 token
```

```
def request_token(image, boundid):
    key = str("bioauth" + boundinfo[boundid]).encode('utf8')
    my_hmac = hmac.new(key, image, sha256)
    return my_hmac.hexdigest().upper()

#传入图片数据路径文件和结果保存文件
def main(data_path,result_path):
    file_data = open(data_path,"r")
    file_result = open(result_path,"w")
    lines = file_data.readlines()
    file_data.close()
    for line in lines:
        segments = line.strip().split('\t')
        if len(segments) < 2:
            continue
        #读入两张需要比对的图片路径
        image1_url = segments[0]
        image2_url = segments[1]
        #生成签名字符串
        time_begin = time.time()
        str_timestamp_ms = str(int(time.time()*1000))
        str_signature = request_signature(str_timestamp_ms, kBoundId)
        #请求头域
        headers = {
            'Authorization': str_signature,
            'Content-Type': 'application/json',
            'x-mx-timestamp': str_timestamp_ms,
            'x-mx-appid': kBoundId
        }
        #请求体
        request_body = {}
        request_body['app_id'] = kBoundId
        request_body['person_id'] = '1234567890'
        request_body['terminal'] = {'type':'iphone7',
                                    'system':'ios10',
                                    'sdk':'o236'}
        #图片采用 base64 编码
        image1_str = base64.b64encode(open(image1_url, 'rb').read())
        image2_str = base64.b64encode(open(image2_url, 'rb').read())
        #生成图片 token
        image1_token = request_token(image1_str, kBoundId)
        image2_token = request_token(image2_str, kBoundId)
        request_body['image1'] = {'content_type': 'jpg',
                                  'data': image1_str,
                                  'token': image1_token,
                                  'category': 1,
```

```
                                }
        request_body['image2'] = {'content_type': 'jpg',
                                  'data': image2_str,
                                  'token': image2_token,
                                  'category': 2,
                                }
        #请求会话
        data_req = json.dumps(request_body)
        request = urllib2.Request(kUrl, data_req, headers)
        response = urllib2.urlopen(request)
        content = response.read()
        if content:
             print content

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print "usage python FaceCompareAPI [data.txt] [result.log]"
        exit()
    main(sys.argv[1],sys.argv[2])
```

2. Java 调用样例

以人脸比对 API 为例，Java 版本客户端调用样例如下：

```
package com.paic.controller.test;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.sql.Timestamp;
import java.util.Arrays;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import net.sf.json.JSONObject;
import sun.misc.BASE64Encoder;

public class testCompare {
/**
```

```
 * 比对主入口
 * @param args
 * @throws Exception
 */
    public static void main(String[] args) throws Exception {
        String image1path = "C:\\Image\\person3.jpg";
        String image2path = "C:\\Image\\person4.jpg";
        String appkey = "1234567890ABCDEFG";
        String boundid = "6222980101038034";
        String res = null;
        String url = "http://192.168.100.26:81/openapi/face/v2/compare";
        String p1 = new BASE64Encoder().encode(readFile(image1path));
        String p2 = new BASE64Encoder().encode(readFile(image2path));
        res = compare(url, p1, p2, boundid, appkey);
        JSONObject robj = JSONObject.fromObject(res);
        System.out.println("相似度: " + robj.get("similarity"));
    }
/**
* @param url: 请求路径
 * @param p1: 图片 base64 编码
 * @param p2: 图片 base64 编码
 * @param boundid: 业务申请的 boundid
 * @param appkey: 业务申请的 appkey
 * @return
 * @throws Exception
 */
    public static String compare(String url, String p1, String p2, String
        boundid, String appkey) throws Exception {
        JSONObject obj = new JSONObject();
        JSONObject terminalobj = new JSONObject();
        JSONObject IMAGE1obj = new JSONObject();
        JSONObject IMAGE2obj = new JSONObject();
        terminalobj.put("sdk", "o236");
        terminalobj.put("type", "iphone");
        terminalobj.put("system", "ios9");

        IMAGE1obj.put("category", 1);
        IMAGE1obj.put("content_type", "jpg");
        IMAGE1obj.put("token", getTokenValue(p1, appkey));
        IMAGE1obj.put("data", p1);

        IMAGE2obj.put("category", 2);
        IMAGE2obj.put("content_type", "jpg");
        IMAGE2obj.put("token", getTokenValue(p2, appkey));
        IMAGE2obj.put("data", p2);
```

```
        obj.put("image1", IMAGE1obj);
        obj.put("image2", IMAGE2obj);
        obj.put("terminal", terminalobj);
        obj.put("person_id", "1234567890");
        obj.put("app_id", boundid);
        String res = "";
        try {
            long time = System.currentTimeMillis();
            res = service(url, obj, boundid, appkey);
            long endtime = System.currentTimeMillis();
            System.out.println("耗时: " + (endtime - time) + "ms");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return res;
    }

/**
 * 文件读取
*/
    public static byte[] readFile(String filePath) {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();

        byte[] buf = new byte[1024 * 50];
        InputStream in = null;
        try {
            in = new FileInputStream(filePath);
            int position = 0;
            while ((position = in.read(buf)) != -1) {
                bos.write(buf, 0, position);
            }
            in.close();
            return bos.toByteArray();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
/**
* @param urlString:请求路径
 * @param jsonobj:请求体
 * @param appid: 业务申请的 boundid
 * @param appkey: 业务申请的 appkey
 * @return: 返回比对的結果
```

```
* @throws Exception
*/
   public static String service(String urlString, JSONObject jsonobj,
       String appid, String appkey) throws Exception {
       URL url = new URL(urlString);
       HttpURLConnection connection = (HttpURLConnection)
              url.openConnection();
       connection.setDoOutput(true);
       connection.setDoInput(true);
       connection.setRequestMethod("POST");
       connection.setUseCaches(false);
       connection.setInstanceFollowRedirects(true);
       long timestamp = new
              Timestamp(System.currentTimeMillis()).getTime();
       connection.setRequestProperty("Content-Type", "application/json");
       connection.setRequestProperty("Authorization",
              getTokenValue(String.valueOf(timestamp), appkey));
       connection.setRequestProperty("x-mx-timestamp",
              String.valueOf(timestamp));
       connection.setRequestProperty("x-mx-appid", appid);
       connection.connect();
       DataOutputStream out = new DataOutputStream(new
              BufferedOutputStream(connection.getOutputStream()));
       out.writeBytes(jsonobj.toString());
       out.flush();
       out.close();
       StringBuffer sb = new StringBuffer("");
       String lines;
       if (connection.getResponseCode() == 200) {
          // 当为 200 的情况下，使用 connection.getInputStream() 来获取返回信息
          BufferedReader reader = new BufferedReader(new
                InputStreamReader(connection.getInputStream(), "utf-8"));
          while ((lines = reader.readLine()) != null) {
             sb.append(lines);
          }
          reader.close();
          //inputStream 要在 connection disconnect 之前
          connection.getInputStream().close();
          connection.disconnect();
          return sb.toString();
       } else {
          //异常处理流程
          // 直接解析 getErrorStream(),通过返回的 errorCode 来进行异常流程处理;
          // System.out.println(connection.getResponseCode());
          BufferedReader reader = new BufferedReader(new
                InputStreamReader(connection.getErrorStream(), "utf-8"));
```

```
            while ((lines = reader.readLine()) != null) {
                sb.append(lines);
            }
            reader.close();
            connection.getErrorStream().close();
            connection.disconnect();
            // 链接处理完成，先把所有该关的先关了
            if ( connection.getResponseCode() == 400
                    || connection.getResponseCode() == 401
                    || connection.getResponseCode() == 413
                    || connection.getResponseCode() == 415
                    || connection.getResponseCode() == 417) {
                ObjectMapper objectMapper = new ObjectMapper();
                myResponseData rd = objectMapper.readValue(sb.toString(),
                      new TypeReference<myResponseData>() {
                });
                // 此处通过 errorCode 来进行判断不同的异常逻辑处理
                if (rd.getErrorcode() == 1) {
                    // 图片数据为空的逻辑处理
                    return sb.toString();
                } else if (rd.getErrorcode() == 9) {
                    // errorcode 处理......
                    return sb.toString();
                } else {
                    // errorcode 处理......
                    return sb.toString();
                }
            } else {
                // 不在服务定义的几种错误范围内，为服务请求异常
                return sb.toString();
            }
        }
    }
    /**
     * 加密
     * @param imageStr：图片 base64 编码
     * @param appkey：业务申请的 appkey
     * @return
     */
    public static  String getTokenValue(String imageStr,String appkey) {
        String hmacSha = null;
        String SIGNATURE_KEY = "bioauth" + appkey;
        try {
            Mac mac = Mac.getInstance("HmacSHA256");
            SecretKeySpec spec = new
              SecretKeySpec(SIGNATURE_KEY.getBytes("UTF-8"), "HmacSHA256");
```

```
            mac.init(spec);
            byte[] byteHMAC = mac.doFinal(imageStr.getBytes("UTF-8"));
            StringBuffer sbLogRet = new StringBuffer();
            for (int i = 0; i < byteHMAC.length; i++) {
                String inTmp = null;
                String text = Integer.toHexString(byteHMAC[i]);
                if (text.length() >= 2) {
                    inTmp = text.substring(text.length() - 2, text.length());
                } else {
                    char[] array = new char[2];
                    Arrays.fill(array, 0, 2 - text.length(), '0');
                    System.arraycopy(text.toCharArray(), 0, array, 2 -
                                     text.length(), text.length());
                    inTmp = new String(array);
                }
                sbLogRet.append(inTmp);
            }
            hmacSha = sbLogRet.toString().toUpperCase();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return hmacSha;
    }
}
```