FACE 文档

2017-08-23

目录

第1章

API参考

1.1 简介

Hi，您好，欢迎使用百度人脸识别API服务。

本文档主要针对API开发者，描述百度人脸识别接口服务的相关技术内容。如果您对文档内容有任何疑问，可以通过以下几种方式联系我们：

- 在百度云控制台内提交工单，咨询问题类型请选择人工智能服务；
- 加入开发者QQ群：224994340；

温馨提示：人脸接口服务已推出v2版本，兼容v1版本，功能更全，性能更优，此文档只保留v2版本内容。

1.1.1 接口能力

接口名称	接口能力简要描述
人脸检测	检测人脸并定位，返回五官关键点，及人脸各属性值
人脸比对	返回两两比对的人脸相似值
人脸识别	在人脸库中查找相似的人脸
人脸认证	识别上传的图片是否为指定用户
人脸库设置	对人脸库的相关操作，如注册、删除、更新、查找用户信息等

1.1.2 请求格式

POST方式调用

注意：Content-Type为application/x-www-form-urlencoded，然后通过urlencode格式化请求体。

1.1.3 返回格式

JSON格式

1.1.4 请求限制

请求图片需经过base64编码：图片的base64编码指将一副图片数据编码成一串字符串，使用该字符串代替图像地址。您可以首先得到图片的二进制，然后用Base64格式编码即可。

注意：图片的base64编码是不包含图片头的，如（）

请求格式支持：PNG、JPG、JPEG、BMP，不支持GIF图片

接口名称	图片编码后大小限额
人脸检测	小于2M
人脸比对	单次传入的两张图片，小于20M
人脸识别	小于10M
人脸认证	小于10M
人脸库设置	小于10M

1.2 调用方式

调用AI服务相关的API接口有两种调用方式，两种不同的调用方式采用相同的接口URL。

区别在于请求方式和鉴权方法不一样，请求参数和返回结果一致。

1.2.1 调用方式一

请求URL数据格式

向API服务地址使用POST发送请求，必须在URL中带上参数：

access_token：必须参数，参考“Access Token获取”。

> 注意：access_token的有效期为30天，需要每30天进行定期更换；

POST中参数按照API接口说明调用即可。

例如人脸识别API，使用HTTPS POST发送：

```
https://aip.baidubce.com/rest/2.0/face/v1/detect?access_token=24.f9ba9c5241b67688bb4adbed8bc91dec.
2592000.1485570332.282335-8574074
```

> 说明：方式一鉴权使用的Access_token必须通过API Key和Secret Key获取。

1.2.2 调用方式二

请求头域内容

在请求的HTTP头域中包含以下信息：

- host（必填）
- x-bce-date（必填）
- x-bce-request-id（选填）
- authorization（必填）
- content-type（必填）
- content-length（选填）

作为示例，以下是一个标准的人脸识别的请求头域内容:

```
POST /rest/2.0/face/v1/detect HTTP/1.1
accept-encoding: gzip, deflate
x-bce-date: 2015-03-24T13:02:00Z
connection: keep-alive
accept: */*
host: aip.baidubce.com
x-bce-request-id: 73c4e74c-3101-4a00-bf44-fe246959c05e
content-type: application/x-www-form-urlencoded
authorization: bce-auth-v1/46bd9968a6194b4bbdf0341f2286ccce/2015-03-24T13:02:00Z/
1800/host;x-bce-date/994014d96b0eb26578e039fa053a4f9003425da4bfedf33f4790882fb4c54903
```

> 说明：方式二鉴权使用的API认证机制，authorization必须通过百度云的AK/SK生成。

1.3 人脸检测

1.3.1 接口描述

检测请求图片中的人脸，返回人脸位置、72个关键点坐标、及人脸相关属性信息。

检测响应速度，与图片中人脸数量相关，人脸数量较多时响应时间会有些许延长。

典型应用场景：如人脸属性分析，基于人脸关键点的加工分析，人脸营销活动等。

> 五官位置会标记具体坐标；72个关键点坐标也包含具体坐标，但不包含对应位置的详细位置描述。

1.3.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v1/detect

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
image	是	string	base64编码后的图片数据，图片大小不超过2M。

参数	是否必选	类型	说明
max_face_num	否	uint32	最多处理人脸数目，默认值1
face_fields	否	string	包括age,beauty,expression,faceshape,gender,gl信息，逗号分隔，默认只返回人脸框、概率和旋转角度。

说明：face_fields参数，默认只返回人脸框、概率和旋转角度，age等更多属性，请在此参数中添加。

1.3.3 返回说明

返回参数

参数	类型	是否必须	说明
log_id	uint64	是	日志id
result_num	uint32	是	人脸数目
result	object[]	是	人脸属性对象的集合
+age	double	否	年龄。face_fields包含age时返回
+beauty	double	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+cation	bject	是	人脸在图片中的位置
++left	uint32	是	人脸区域离左边界的距离
++top	uint32	是	人脸区域离上边界的距离
++width	uint32	是	人脸区域的宽度
++height	uint32	是	人脸区域的高度
+face_probability	double	是	人脸置信度，范围0-1

参数	类型	是否必须	说明
+rotation_angle	int32	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	double	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	double	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	double	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	uint32	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	double	否	表情置信度，范围0~1。face_fields包含expression时返回
+faceshape	object[]	否	脸型置信度。face_fields包含faceshape时返回
++type	string	是	脸型：square/ triangle/ oval/ heart/ round
++probability	double	是	置信度：0~1
+gender	string	否	male、female。face_fields包含gender时返回
+gender_probability	double	否	性别置信度，范围0~1。face_fields包含gender时返回
+glasses	uint32	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回

参数	类型	是否必须	说明
+glasses_probability	double	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	object[]	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	uint32	否	x坐标
++y	uint32	否	y坐标
+landmark72	object[]	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	uint32	否	x坐标
++y	uint32	否	y坐标
+race	string	否	yellow、white、black、arabs。face_fields包含race时返回
+race_probability	double	否	人种置信度，范围0~1。face_fields包含race时返回
+qualities	object	否	人脸质量信息。face_fields包含qualities时返回
++occlusion	object	是	人脸各部分遮挡的概率，[0, 1]（待上线）
+++left_eye	double	是	左眼
+++right_eye	double	是	右眼
+++nose	double	是	鼻子
+++mouth	double	是	嘴
+++left_cheek	double	是	左脸颊
+++right_cheek	double	是	右脸颊
+++chin	double	是	下巴

参数	类型	是否必须	说明
++blur	double	是	人脸模糊程度，[0, 1]。0表示清晰，1表示模糊（待上线）
++illumination	-	是	取值范围在[0,255],表示脸部区域的光照程度（待上线）
++completeness	-	是	人脸完整度，[0, 1]。0表示完整，1表示不完整（待上线）
++type	object	是	真实人脸/卡通人脸置信度
+++human	-	是	真实人脸置信度，[0, 1]
+++cartoon	-	是	卡通人脸置信度，[0, 1]

返回示例


```
{
    "result_num": 1,
    "result": [
        {
            "location": {
                "left": 90,
                "top": 92,
                "width": 111,
                "height": 99
            },
            "face_probability": 1,
            "rotation_angle": 6,
            "yaw": 11.61234664917,
            "pitch": -0.30852827429771,
            "roll": 8.8044967651367,
            "landmark": [
                {
                    "x": 105,
                    "y": 110
                },
```

```
              ...
            ],
            "landmark72": [
                {
                    "x": 88,
                    "y": 109
                },
               ...
            ],
            "gender": "male",
            "gender_probability": 0.99358034133911,
            "glasses": 0,
            "glasses_probability": 0.99991309642792,
            "race": "yellow",
            "race_probability": 0.99960690736771,
            "qualities": {
                "occlusion": {
                    "left_eye": 0.000085282314103097,
                    "right_eye": 0.00001094374601962,
                    "nose": 3.2677664307812e-7,
                    "mouth": 2.6582130940866e-10,
                    "left_cheek": 8.752236624332e-8,
                    "right_cheek": 1.0212766454742e-7,
                    "chin": 4.2632994357028e-10
                },
                "blur": 4.5613666312237e-41,
                "illumination": 0,
                "completeness": 0,
                "type": {
                    "human": 0.98398965597153,
                    "cartoon": 0.016010366380215
                }
            }
        }
    ],
    "log_id": 2418894422
}
```


1.4 人脸比对

1.4.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

该请求用于比对多张图片中的人脸相似度并返回两两比对的得分，可用于判断两张脸是否是同一人的可能性大小。

典型应用场景：如人证合一验证，用户认证等，可与您现有的人脸库进行比对验证。

说明：支持对比对的两张图片做在线活体检测

1.4.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/match

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
images	是	string	base64编码后的**2张图片数据**，半角逗号分隔，单次请求总共最大20M

参数	是否必选	类型	说明
ext_fields	否	string	返回质量信息，取值固定: 目前支持qualities(质量检测)。(对所有图片都会做改处理)
image_liveness	否	string	返回的活体信息，“faceliveness,faceliveness”表示对比对的两张图片都做活体检测；“,faceliveness” 表示对第一张图片不做活体检测、第二张图做活体检测；“faceliveness,” 表示对第一张图片做活体检测、第二张图不做活体检测

1.4.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求唯一标识码，随机数
result_num	是	uint32	返回结果数目，即：result数组中元素个数
result	是	array(object)	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。得分[0,100.0]
+index_i	是	uint32	比对图片1的index
+index_j	是	uint32	比对图片2的index
+score	是	double	比对得分

字段	是否必选	类型	说明
ext_info	否	array (dict)	对应参数中的ext_fields
+qualities	否	string	质量相关的信息，无特殊需求可以不使用
+faceliveness	否	string	活体分数“0,0.9999”（表示第一个图不做活体检测、第二个图片活体分数为0.9999）

返回示例

```
//请求两张图片
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "index_i": 0,
            "index_j": 1,
            "score": 44.3
        }
    ]
}
```

1.5 人脸识别

1.5.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于计算指定组内用户，与上传图像中人脸的相似度。识别前提为您已经创建了一个人脸库。

典型应用场景：如人脸闸机，考勤签到，安防监控等。

说明：人脸识别返回值不直接判断是否是同一人，只返回用户信息及相似度分值。

说明：推荐可判断为同一人的相似度分值为80，您也可以根据业务需求选择更合适的阈值。

1.5.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/identify

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制128B，如果需要查询多个用户组id，用逗号分隔
images	是	string	图像base64编码，**每次仅支持单张图片，图片编码后大小不超过10M**

参数	是否必选	类型	说明
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)
user_top_num	否	uint32	返回用户top数，默认为1，最多返回5个

1.5.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求唯一标识码，随机数
result_num	是	uint32	返回结果数目，即：result数组中元素个数
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数，如0.49999
result	是	array(object)	结果数组
+group_id	是	string	对应的这个用户的group_id
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+scores	是	array(double)	结果数组，数组元素为匹配得分，top n。得分[0,100.0]

返回示例

```
{
    "log_id": 73473737,
    "result_num":1,
```

```
    "result": [
        {
            "group_id" : "test1",
            "uid": "u333333",
            "user_info": "Test User",
            "scores": [
                    99.3,
                    83.4
            ]
        }
    ]
}
```

1.6 人脸认证

1.6.1 接口描述

> 温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于识别上传的图片是否为指定用户，即查找前需要先确定要查找的用户在人脸库中的id。

典型应用场景：如人脸登录，人脸签到等

> 说明：人脸认证与人脸识别的差别在于：人脸识别需要指定一个待查找的人脸库中的组；而人脸认证需要指定具体的用户id即可，不需要指定具体的人脸库中的组；实际应用中，人脸认证需要用户或系统先输入id，这增加了验证安全度，但也增加了复杂度，具体使用哪个接口需要视您的业务场景判断。

> 说明：请求参数中，新增在线活体检测

1.6.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/verify

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
images	是	string	图像base64编码，**每次仅支持单张图片，图片编码后大小不超过10M**
group_id	是	string	用逗号分隔，表示从指定的group中查找
top_num	否	uint32	返回匹配得分top数，默认为1
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)

1.6.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求唯一标识码，随机数

字段	是否必选	类型	说明
result_num	是	uint32	返回结果数目，即：result数组中元素个数
result	是	array(double)	结果数组，数组元素为匹配得分，top n。 得分范围[0,100.0]。 推荐得分超过80可认为认证成功
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数， 如0.49999

返回示例

```
{
    "log_id": 73473737,
    "result_num":2,
    "result": [
            99.3,
            83.6
    ]
}
```

1.7 人脸注册

1.7.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于从人脸库中新增用户，可以设定多个用户所在组，及组内用户的人脸图片，

典型应用场景：构建您的人脸库，如会员人脸注册，已有用户补全人脸信息等。

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库
    |- 用户组一
        |- 用户01
            |- 人脸
        |- 用户02
            |- 人脸
            |- 人脸
            ....
        ....
    |- 用户组二
    |- 用户组三
    |- 用户组四
    ....
```

说明：关于人脸库的设置限制

- 每个开发者账号只能创建一个人脸库；
- 每个人脸库下，用户组（group）数量没有限制；
- 每个用户组（group）下，可添加最多300000张人脸，如每个uid注册一张人脸，则最多300000个用户uid；
- 每个用户（uid）所能注册的最大人脸数量没有限制；

说明：人脸注册完毕后，生效时间最长为35s，之后便可以进行识别或认证操作。

说明：注册的人脸，建议为用户正面人脸。

说明：uid在库中已经存在时，对此uid重复注册时，新注册的图片默认会追加到该uid下，如果手动选择`action_type:replace`，则会用新图替换库中该uid下所有图片。

1.7.2 请求说明

请求示例

HTTP方法：POST

请求URL：`https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/add`

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
user_info	是	string	用户资料，长度限制256B
group_id	是	string	用户组id，标识一组用户（由数字、字母、下划线组成），长度限制128B。如果需要将一个uid注册到多个uid下，group_id需要用多个逗号分隔，每个group_id长度限制为48个英文字符
images	是	string	图像base64编码，**每次仅支持单张图片，图片编码后大小不超过10M**
action_type	否	string	参数包含append、replace。如果为“replace”，则每次注册时进行替换replace（新增或更新）操作，默认为append操作

1.7.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回示例

```
// 注册成功
{
    "log_id": 73473737,
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

1.8 人脸更新

1.8.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于对人脸库中指定用户，更新其下的人脸图像。

说明：针对一个uid执行更新操作，新上传的人脸图像将覆盖此uid原有所有图像。

说明：执行更新操作，如果该uid不存在时，会返回错误。如果添加了action_type:replace,则不会报错，并自动注册该uid，操作结果等同注册新用户。

1.8.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/update

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
images	是	string	图像base64编码，可单次传入多张，多张图片半角逗号分隔，总共最大10M

1.8.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回示例

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
```

```
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

1.9 人脸删除

1.9.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于从人脸库中删除一个用户。

人脸删除注意事项：

- 删除的内容，包括用户所有图像和身份信息；
- 如果一个uid存在于多个用户组内，将会同时将从各个组中把用户删除
- 如果指定了group_id，则只删除此group下的uid相关信息

1.9.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/delete

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	否	string	选择指定group_id则只删除此group下的uid内容，如果不指定则删除group下对应uid的信息

1.9.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回示例

```
// 删除成功
{
    "log_id": 73473737,
}
// 删除发生错误
{
  "error_code": 216612,
  "log_id": 1382953199,
  "error_msg": "user not exist"
}
```

1.10 用户信息查询

1.10.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于查询人脸库中某用户的详细信息。

1.10.2 请求说明

请求示例

HTTP方法：GET

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/get

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B

1.10.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

字段	是否必选	类型	说明
result	是	array(double)	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+groups	是	array(string)	用户所属组列表

返回示例

```
{
“result_num” : 2
    "result": {
        [
            "uid": "testuser2",
            "user_info": "registed user info ...",
            "group_id": "grep1",
        ],
        [
            "uid": "testuser2",
            "user_info": "registed user info2 ...",
            "group_id": "grep2",
        ],
    },
    "log_id": 2979357502
}
```

1.11 组列表查询

1.11.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于查询用户组的列表。

1.11.2 请求说明

请求示例

HTTP方法：GET

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/group/getlist

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
start	否	uint32	默认值0，起始序号
end	否	uint32	返回数量，默认值100，最大值1000

1.11.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回个数
result	是	array(string)	group_id列表

返回示例

```
{
    "result_num": 2,
    "result": [
        "grp1",
        "grp2"
    ],
```

```
    "log_id": 3314921889
}
```

1.12 组内用户列表查询

1.12.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于查询指定用户组中的用户列表。

1.12.2 请求说明

请求示例

HTTP方法：GET

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/group/getusers

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	uint32	默认值0，起始序号
end	否	uint32	返回数量，默认值100，最大值1000

1.12.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回个数
result	是	array(object)	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回示例

```
{
    "log_id": 3314921889,
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

1.13 组内添加用户

1.13.1 接口描述

> 温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于将已经存在于人脸库中的用户添加到一个新的组。

说明：并不是向一个指定组内添加用户，而是直接从其它组复制用户信息

1.13.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/group/adduser

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
group_id	是	string	需要添加信息的组id，多个的话用逗号分隔
uid	是	string	用户id
src_group_id	是	string	从指定group里复制信息

1.13.3 返回说明

返回参数

字段	是否必选	类型	说明

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回示例

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
  "error_msg": "already add"
}
```

1.14 组内删除用户

1.14.1 接口描述

温馨提示：此接口已更新为v2版本，兼容v1版本，功能更全，性能更优，建议v1版本老用户升级到v2版本。

用于将用户从某个组中删除，但不会删除用户在其它组的信息。

说明：当用户仅属于单个分组时，本接口将返回错误，请使用人脸删除接口

1.14.2 请求说明

请求示例

HTTP方法：POST

请求URL：https://aip.baidubce.com/rest/2.0/face/v2/faceset/group/deleteuser

URL参数：

参数	值
access_token	通过API Key和Secret Key获取的access_token,参考“Access Token获取”

Header如下：

参数	值
Content-Type	application/x-www-form-urlencoded

Body中放置请求参数，参数详情如下：

请求参数

参数	是否必选	类型	说明
group_id	是	string	用户组id，多个的话用逗号分隔
uid	是	string	用户id

1.14.3 返回说明

返回参数

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回示例

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
  "error_msg": "user must be in one group at least"
}
```

1.15 错误码

若请求错误，服务器将返回的JSON文本包含以下参数：

- error_code：错误码。
- error_msg：错误描述信息，帮助理解和解决发生的错误。

例如Access Token失效返回：

```
{
  "error_code": 110,
  "error_msg": "Access token invalid or no longer valid"
}
```

需要重新获取新的Access Token再次请求即可。

错误码	错误信息	描述
4	Open api request limit reached	集群超限额
17	Open api daily request limit reached	每天流量超限额
18	Open api qps request limit reached	QPS超限额
19	Open api total request limit reached	请求总量超限额
100	Invalid parameter	无效的access_token参数
110	Access token invalid or no longer valid	Access Token失效
111	Access token expired	Access token过期
216015	module closed	模块关闭
216100	invalid param	参数异常
216101	not enough param	缺少必须的参数
216102	service not support	请求了不支持的服务，请检查调用的url
216103	param too long	请求超长，一般为一次传入图片个数超过系统限制
216110	appid not exist	appid不存在，请重新检查后台应用列表中的应用信息

错误码	错误信息	描述
216111	invalid userid	userid信息非法，请检查对应的参数
216200	empty image	图片为空或者base64解码错误
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	数据库异常，**少量发生时重试即可**
216400	backend error	后端识别服务异常，可以根据具体msg查看错误原因
216401	internal error	内部错误
216402	face not found	未找到人脸，请检查图片是否含有人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在，请确认该用户是否注册或注册已经生效(**需要已经注册超过35s**)
216613	fail to delete user record	删除用户图片记录失败，重试即可
216614	not enough images	两两比对中图片数少于2张，无法比较
216615	fail to process images	服务处理该图片失败，发生后重试即可
216616	image existed	图片已存在
216617	fail to add user	新增用户图片失败
216618	no user in group	组内用户为空，确认该group是否存在或已经生效(需要已经注册超过35s)
216631	request add user overlimit	本次请求添加的用户数量超限

第2章

C# SDK文档

本文档主要介绍人脸识别C# SDK的安装和使用。在使用本文档前，您需要先了解人脸识别的基础知识，并已经开通了人脸识别服务。

2.0.1 依赖

- Visual Studio 2010 (C# 4.0)
- Newtonsoft.Json (可通过Nuget管理)

2.1 快速入门

Baidu.Aip.Face.Face是主要的OCR交互类，基本使用方法如下：

```
var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
var image = File.ReadAllBytes("图片文件路径");
var options = new Dictionary<string, object>()
{
    {"face_fields", "beauty,age"}
};
var result = client.FaceDetect(image, options);
```

其中Api Key 与 Secret Key在百度云控制台中创建App后会自动分配，用于标识用户，为访问做签名验证。

2.2 错误信息格式

若请求错误，返回的JObject中将包含：

- error_code：错误码；关于错误码的详细信息可参考下表。
- error_message：错误描述信息，帮助理解和解决发生的错误。

服务端返回的错误码

错误码	错误信息	描述
216015	module closed	模块关闭
216100	invalid param	非法参数
216101	not enough param	参数数量不够
216102	service not support	业务不支持
216103	param too long	参数太长
216110	appid not exist	APP ID不存在
216111	invalid userid	非法用户ID
216200	empty image	空的图片
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	DB错误
216400	backend error	后端系统错误
216401	internal error	内部错误
216402	face not found	没有找到人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在
216613	user not found	用户查找不到
216614	not enough images	图片信息不完整
216615	fail to process images	处理图片信息失败
216616	image existed	图片已存在
216617	fail to add user	添加用户失败
216618	no user in group	群组里没有用户
216630	recognize error	识别错误

2.3 人脸检测

人脸识别基于百度业界领先的智能人脸分析算法，提供人脸检测、人脸识别、关键点定位、属性识别和活体检测等一整套技术方案。

```
public static void FaceDetect()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image = File.ReadAllBytes("图片文件路径");
    var options = new Dictionary<string, object>()
    {
        {"face_fields", "beauty,age"}
    };
    var result = client.FaceDetect(image, options);
}
```

options参数详情

参数	类型	描述	是否必须
face_fields	Boolean	包括age、beauty、expression、face-shape、gender、glasses、land-mark、race、qualities 信息，逗号分隔，默认只返回人脸框、概率和旋转角度。	否
max_face_num	unit32	最多处理人脸数目，默认值1	是

返回数据参数详情

参数	类型	是否一定输出	描述
log_id	uint64	是	日志id
result_num	Int	是	人脸数目
result	object[]	是	人脸属性对象的集合
+age	double	否	年龄。face_fields包含age时返回

参数	类型	是否一定输出	描述
+beauty	double	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+location	object	是	人脸在图片中的位置
++left	Int	是	人脸区域离左边界的距离
++top	Int	是	人脸区域离上边界的距离
++width	Int	是	人脸区域的宽度
++height	Int	是	人脸区域的高度
+face_probability	double	是	人脸置信度，范围0-1
+rotation_angle	int32	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	double	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	double	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	double	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	Int	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	double	否	表情置信度，范围0~1。face_fields包含expression时返回
+faceshape	object[]	否	脸型置信度。face_fields包含faceshape时返回

参数	类型	是否一定输出	描述
++type	String	是	脸型：square/ triangle/ oval/ heart/ round
++probability	double	是	置信度：0~1
+gender	String	否	male、female。face_fields包含gender时返回
+gender_probability	double	否	性别置信度，范围0~1。face_fields包含gender时返回
+glasses	Int	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回
+glasses_probability	double	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	object[]	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	Int	否	x坐标
++y	Int	否	y坐标
+landmark72	object[]	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	Int	否	x坐标
++y	Int	否	y坐标
+race	String	否	yellow、white、black、arabs。face_fields包含race时返回
+race_probability	double	否	人种置信度，范围0~1。face_fields包含race时返回

参数	类型	是否一定输出	描述
+qualities	object	否	人脸质量信息。face_fields包含qualities时返回
++occlusion	object	是	人脸各部分遮挡的概率，[0, 1]（待上线）
+++left_eye	double	是	左眼
+++right_eye	double	是	右眼
+++nose	double	是	鼻子
+++mouth	double	是	嘴
+++left_cheek	double	是	左脸颊
+++right_cheek	double	是	右脸颊
+++chin	double	是	下巴
++type	object	是	真实人脸/卡通人脸置信度
+++human	double	是	真实人脸置信度，[0, 1]
+++cartoon	double	是	卡通人脸置信度，[0, 1]

2.4 人脸比对

人脸比对接口提供了对上传图片进行两两比对，并输出相似度得分的功能。

接受的参数为一系列图片的字节数组。

```
public static void FaceMatch()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image1 = File.ReadAllBytes("图片文件路径");
    var image2 = File.ReadAllBytes("图片文件路径");
    var image3 = File.ReadAllBytes("图片文件路径");

    var images = new byte[][] {image1, image2, image3};

    // 人脸对比
    var result = client.FaceMatch(images);
```

```
}
```

人脸比对请求参数要求：

所有图片经base64编码后的图片数据总和不超过10M。

人脸比对返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回结果数目，即：result数组中元素个数
result	是	array(object)	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。 得分[0,100.0]
+index_i	是	uint32	比对图片1的index
+index_j	是	uint32	比对图片2的index
+score	是	double	比对得分

返回样例：

```
//请求为四张图片，第三张解析失败
{
    "log_id": 73473737,
    "result_num":3,
    "result": [
        {
            "index_i": 0,
            "index_j": 1,
            "score": 44.3
        },
        {
            "index_i": 0,
            "index_j": 3,
            "score": 89.2
        },
        {
```

```
            "index_i": 1,
            "index_j": 3,
            "score": 10.4
        }
        ……
    ]
}
```

2.5 人脸查找

2.5.1 人脸注册

人脸注册接口提供了使用上传图片进行注册新用户的功能，需要指定注册用户的id和描述信息，所在组id以及本地用户人脸图片。 注：每个用户（uid）所能注册的最大人脸数量为5张。

```
public static void FaceRegister()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image1 = File.ReadAllBytes("图片文件路径");
    var image2 = File.ReadAllBytes("图片文件路径");
    var image3 = File.ReadAllBytes("图片文件路径");

    var images = new byte[][] {image1, image2, image3};

   var result = client.User.Register(images, "uid", "user info here", "groupId");

}
```

人脸注册请求参数要求：

所有图片经base64编码后的图片数据总和不超过10M。

人脸注册返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回样例：

```
// 注册成功
{
    "log_id": 73473737,
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

2.5.2 人脸更新

人脸更新接口提供了为已有用户更新人脸图像的功能，新上传的图片将覆盖原有图像。

举例，要更新一个新用户，用户id为uid1， 更新成功后服务端会返回操作的logid：

```
public static void FaceUpdate()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image1 = File.ReadAllBytes("图片文件路径");
    var image2 = File.ReadAllBytes("图片文件路径");
    var image3 = File.ReadAllBytes("图片文件路径");

    var images = new byte[][] {image1, image2, image3};

    var result = client.User.Update(images, "uid");
}
```

人脸更新请求参数要求：

uid需要在库中已存在，且组成为字母/数字/下划线，长度不超过128B。所有图片经base64编码后的图片数据总和不超过10M。

人脸注册返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回样例：

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

2.5.3 人脸删除

人脸删除接口提供了从库中彻底删除一个用户的功能，包括用户所有图像和身份信息，同时也将从各个组中把用户删除。

举例，要删除一个新用户，用户id为uid1， 删除成功后服务端会返回操作的logid：

```
public static void FaceDelete()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.User.Delete("uid");
}
```

人脸删除请求参数要求：

uid需要在库中已存在，且组成为字母/数字/下划线，长度不超过128B。

人脸删除返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回样例：

```
// 成功
{
    "log_id": 73473737,
}
// 错误
```

```
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

2.5.4 人脸认证

人脸认证接口用于识别上传的图片是否为指定用户。

```
public static void FaceVerify()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image1 = File.ReadAllBytes("图片文件路径");
    var image2 = File.ReadAllBytes("图片文件路径");
    var image3 = File.ReadAllBytes("图片文件路径");

    var images = new byte[][] {image1, image2, image3};

    var result = client.User.Verify(images, "uid",  1);
}
```

人脸认证请求参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
images	是	string	图像base64编码,多张图片半角逗号分隔，总共最大10M
top_num	否	uint32	返回匹配得分top数，默认为1

人脸认证返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

字段	是否必选	类型	说明
result_num	是	uint32	返回结果数目，即：result数组中元素个数
result	是	array(double)	结果数组，数组元素为匹配得分，top n。 得分范围[0,100.0]。 得分超过80可认为认证成功

返回样例：

```
{
  "results": [
    93.86580657959,
    92.237548828125
  ],
  "result_num": 2,
  "log_id": 1629483134
}
```

2.5.5 人脸识别

人脸识别接口用于计算指定组内用户与上传图像的相似度。

```
public static void FaceIdentify()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var image1 = File.ReadAllBytes("图片文件路径");
    var image2 = File.ReadAllBytes("图片文件路径");
    var image3 = File.ReadAllBytes("图片文件路径");

    var images = new byte[][] {image1, image2, image3};

    var result = client.User.Identify(images, "groupId", 1, 1);
}
```

人脸识别请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id(由数字、字母、下划线组成)，长度限制128B
images	是	string	图像base64编码,多张图片半角逗号分隔，总共最大10M
user_top_num	否	uint32	返回用户top数，默认为1
face_top_num	否	uint32	单用户人脸匹配得分top数，默认为1

人脸识别返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回结果数目，即：result数组中元素个数
result	是	array(double)	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+scores	是	array(double)	结果数组，数组元素为匹配得分，top n。 得分[0,100.0]

返回样例：

```
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "uid": "u333333",
            "user_info": "Test User",
            "scores": [
                    99.3,
```

```
                    83.4
                ]
            }
        ]
}
```

2.5.6 用户信息查询

用户信息查询接口用于查询某用户的详细信息。

```
public static void UserInfo()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.User.GetInfo("uid");
}
```

用户信息查询请求参数要求：

uid需要在库中已存在，且组成为字母/数字/下划线，长度不超过128B。

用户信息查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result	是	array(double)	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+groups	是	array(string)	用户所属组列表

返回样例：

```
{
    "result": {
        "uid": "testuser2",
        "user_info": "registed user info ...",
        "groups": [
            "grp1",
            "grp2",
            "grp3"
```

```
        ]
    },
    "log_id": 2979357502
}
```

2.5.7 组列表查询

组列表查询接口用于查询一个app下所有组的列表。

举例：

```
public static void GroupList()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.Group.GetAllGroups(0, 100);
}
```

组列表查询请求参数详情：

参数	是否必选	类型	说明
start	否	uint32	默认值0，起始序号
end	否	uint32	返回数量，默认值100，最大值1000

组列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回个数
result	是	array(string)	group_id列表

返回样例：

```
{
    "result_num": 2,
    "result": [
        "grp1",
        "grp2"
```

```
    ],
    "log_id": 3314921889
}
```

2.5.8 组内用户列表查询

组内用户列表查询接口用于查询一个用户组内所有的用户信息。

举例：

```
public static void GroupUsers()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.Group.GetUsers("groupId", 0, 100);
}
```

组内用户列表查询请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	uint32	默认值0，起始序号
end	否	uint32	返回数量，默认值100，最大值1000

组内用户列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一
result_num	是	uint32	返回个数
result	是	array(object)	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回样例：

```
{
    "log_id": 3314921889,
```

```
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

2.5.9 组内添加用户

组内添加用户接口用于把一个已经存在于库中的用户添加到新的用户组中。

举例：

```
public static void GroupAddUser()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.Group.AddUser("groupId", "uid");
}
```

组内添加用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
uid	是	string	用户id

组内添加用户接口返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回样例：

```
// 正确返回值
```

```
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
  "error_msg": "already add"
}
```

2.5.10 组内删除用户

组内删除用户接口用于把一个用户从某个组中删除，但不会删除用户在其它组内的信息。当用户仅属于单个分组时，本接口将返回错误。

举例：

```
public static void GroupDeleteUser()
{
    var client = new Baidu.Aip.Face.Face("Api Key", "Secret Key");
    var result = client.Group.DeleteUser("groupId", "uid");
}
```

组内删除用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
uid	是	string	用户id

组内删除用户接口返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	uint64	请求标识码，随机数，唯一

返回样例：

```
// 正确返回值
{
```

```
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
  "error_msg": "user must be in one group at least"
}
```

2.6 版本更新记录

上线日期	版本号	更新内容
2017.4.1	1.0	第一版！

第3章

Java SDK文档

3.1 简介

Hi，您好，欢迎使用百度人脸识别API服务。

本文档主要针对API开发者，描述百度人脸识别接口服务的相关技术内容。如果您对文档内容有任何疑问，可以通过以下几种方式联系我们：

- 在百度云控制台内提交工单，咨询问题类型请选择人工智能服务；
- 加入开发者QQ群：224994340；

3.1.1 接口能力

接口名称	接口能力简要描述
人脸检测	检测人脸并定位，返回五官关键点，及人脸各属性值
人脸比对	返回两两比对的人脸相似值
人脸识别	在人脸库中查找相似的人脸
人脸认证	识别上传的图片是否为指定用户
人脸库设置	对人脸库的相关操作，如注册、删除、更新、查找用户信息等

3.1.2 版本更新记录

上线日期	版本号	更新内容
2017.7.14	1.3.6	更新SDK打包方式

上线日期	版本号	更新内容
2017.4.27	1.3.4	人脸比对、识别、认证和人脸库设置接口升级为v2版本
2017.4.20	1.3.3	AI SDK同步版本更新
2017.4.13	1.3.2	AI SDK同步版本更新
2017.3.23	1.3	兼容Android环境
2017.3.2	1.2	上线人脸查找接口，增加对图片参数要求限制的检查，增加设置超时接口
2017.1.20	1.1	上线人脸比对接口，同时修复部分云用户调用不成功的错误
2017.1.6	1.0	初始版本，上线人脸属性识别接口

3.2 快速入门

3.2.1 安装人脸检测 Java SDK

Face Java SDK目录结构

```
com.baidu.aip
        ├── auth                        //签名相关类
        ├── http                        //Http通信相关类
        ├── client                      //公用类
        ├── exception                   //exception类
        ├── face
        │       └── AipFace             //AipFace类
        └── util                        //工具类
```

支持 JAVA版本：1.7+

直接使用JAR包步骤如下：

1.在官方网站下载Java SDK压缩工具包。

2.将下载的aip-face-java-sdk-version.zip解压后，复制到工程文件夹中。

3.在Eclipse右键“工程 -> Properties -> Java Build Path -> Add JARs”。

4.添加SDK工具包face_sdk-version.jaraip-core-version.jar和第三方依赖工具包json-20160810.jar。

其中，version为版本号，添加完成后，用户就可以在工程中使用Face Java SDK。

3.2.2 新建AipFaceClient

1.初始化一个AipFaceClient。

AipFaceClient是与Baidu Face Recognition交互的客户端，所有人脸识别操作都是通过AipFaceClient完成的。

用户可以参考如下代码新建一AipFaceClient：

```
public class Sample {

    //设置APPID/AK/SK
    public static final String APP_ID = ”你的␣App␣ID”;
    public static final String API_KEY = ”你的␣Api␣ID”;
    public static final String SECRET_KEY = ”你的␣Secret␣Key”;

    public static void main(String[] args) {

        // 初始化一个FaceClient
        AipFace client = new AipFace(APP_ID, API_KEY, SECRET_KEY);

        // 可选：设置网络连接参数
        client.setConnectionTimeoutInMillis(2000);
        client.setSocketTimeoutInMillis(60000);

        // 调用API
        String path = ”test.jpg”;
        JSONObject res = client.detect(path, new HashMap<String, String>());
        System.out.println(res.toString(2));
    }
}
```

在上面代码中，常量APP_ID在百度云控制台中创建，常量API_KEY与SECRET_KEY是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在AI服务控制台中的应用列表中查看。

注意：如您以前是百度云的老用户，其中API_KEY对应百度云的“Access Key ID”，SECRET_KEY对应百度云的“Access Key Secret”。

3.2.3 配置AipFaceClient

如果用户需要配置AipFaceClient的一些细节参数，可以在构造AipFace之后调用接口设置参数，目前只支持以下参数：

接口	说明
setConnectionTi meoutInMillis	建立连接的超时时间（单位：毫秒）
setSocketTimeou tInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）

3.3 接口调用

3.3.1 人脸检测

接口描述 检测请求图片中的人脸，返回人脸位置、72个关键点坐标、及人脸相关属性信息。

检测响应速度，与图片中人脸数量相关，人脸数量较多时响应时间会有些许延长。

典型应用场景：如人脸属性分析，基于人脸关键点的加工分析，人脸营销活动等。

> 五官位置会标记具体坐标；72个关键点坐标也包含具体坐标，但不包含对应位置的详细位置描述。

请求说明 图片接受类型支持本地图片路径字符串，图片文件二进制数组。

举例，要对一张图片进行人脸识别，具体的人脸信息在返回的result字段中：

```
public void faceRecognize(AipFace client) {
    // 参数为本地图片路径
    String imagePath = ”picture.jpg”;
    JSONObject response = client.detect(imagePath);
    System.out.println(response.toString());

    // 参数为本地图片文件二进制数组
    byte[] file = readImageFile(imagePath);  // 函数仅为示例readImageFile
    JSONObject response = client.detect(file);
    System.out.println(response.toString());
}
```

传入图片时还想增加一些自定义参数配置：

```
public void faceRecognize(AipFace client) {
    // 自定义参数定义
    HashMap<String, String> options = new HashMap<String, String>();
    options.put(”max_face_num”, ”1”);
    options.put(”face_fields”, ”expression”);
```

```
    // 参数为本地图片路径
    String imagePath = "face.jpg";
    JSONObject response = client.detect(imagePath, options);
    System.out.println(response.toString());

    // 参数为本地图片文件二进制数组
    byte[] file = readImageFile(imagePath);
    JSONObject response = client.detect(file, options);
    System.out.println(response.toString());
}
```

人脸检测 请求参数详情

参数	类型	描述	是否必须
face_fields	string	包括age、beauty、expression、face-shape、gender、glasses、land-mark、race、qualities 信息，逗号分隔，默认只返回人脸框、概率和旋转角度。	否
max_face_num	number	最多处理人脸数目，默认值1	是
image	string	图像数据	是

参数	类型	是否一定输出	描述
log_id	number	是	日志id
result_num	number	是	人脸数目
result	array	是	人脸属性对象的集合
+age	number	否	年龄。face_fields包含age时返回
+beauty	number	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+location	array	是	人脸在图片中的位置

参数	类型	是否一定输出	描述
++left	number	是	人脸区域离左边界的距离
++top	number	是	人脸区域离上边界的距离
++width	number	是	人脸区域的宽度
++height	number	是	人脸区域的高度
+face_probability	number	是	人脸置信度，范围0-1
+rotation_angle	number	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	number	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	number	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	number	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	number	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	number	否	表情置信度，范围0~1。face_fields包含expression时返回
+faceshape	array	否	脸型置信度。face_fields包含faceshape时返回
++type	string	是	脸型：square/ triangle/ oval/ heart/ round
++probability	number	是	置信度：0~1
+gender	string	否	male、female。face_fields包含gender时返回

参数	类型	是否一定输出	描述
+gender_probability	number	否	性别置信度，范围0~1。face_fields包含gender时返回
+glasses	number	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回
+glasses_probability	number	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	array	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+landmark72	array	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+race	string	否	yellow、white、black、arabs。face_fields包含race时返回
+race_probability	number	否	人种置信度，范围0~1。face_fields包含race时返回
+qualities	array	否	人脸质量信息。face_fields包含qualities时返回
++occlusion	array	是	人脸各部分遮挡的概率，[0, 1]（待上线）
+++left_eye	number	是	左眼

参数	类型	是否一定输出	描述
+++right_eye	number	是	右眼
+++nose	number	是	鼻子
+++mouth	number	是	嘴
+++left_cheek	number	是	左脸颊
+++right_cheek	number	是	右脸颊
+++chin	number	是	下巴
++type	array	是	真实人脸/卡通人脸置信度
+++human	number	是	真实人脸置信度，[0, 1]
+++cartoon	number	是	卡通人脸置信度，[0, 1]

返回说明

3.3.2 人脸比对

接口描述 该请求用于比对多张图片中的人脸相似度并返回两两比对的得分，可用于判断两张脸是否是同一人的可能性大小。

典型应用场景：如人证合一验证，用户认证等，可与您现有的人脸库进行比对验证。

> 说明：支持对比对的两张图片做在线活体检测

请求说明 接受的参数为一系列本地图片路径的数组，或图片二进制数据的数组。 举例，要对两张图片进行人脸比对，具体的人脸信息在返回的result字段中：

```
public void faceRecognize(AipFace client) {
    // 参数为本地图片路径
    String imagePath1 = ”test1.jpg”;
    String imagePath2 = ”test2.jpg”;
    ArrayList<String> pathArray = new ArrayList<String>();
    pathArray.add(imagePath1);
    pathArray.add(imagePath2);
    JSONObject response = client.match(pathArray, new HashMap<String, String>());
    System.out.println(response.toString());
}
```

人脸比对请求参数：

所有图片经base64编码后的图片数据总和不超过10M。以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
ext_fields	否	string	返回质量信息，取值固定: 目前支持qualities(质量检测)。(对所有图片都会做改处理)
image_liveness	否	string	返回的活体信息，“faceliveness,faceliveness”表示对比对的两张图片都做活体检测；“,faceliveness” 表示对第一张图片不做活体检测、第二张图做活体检测；“faceliveness,” 表示对第一张图片做活体检测、第二张图不做活体检测

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。 得分[0,100.0]
+index_i	是	number	比对图片1的index
+index_j	是	number	比对图片2的index
+score	是	double	比对得分
ext_info	否	array	对应参数中的ext_fields

字段	是否必选	类型	说明
+qualities	否	string	质量相关的信息，无特殊需求可以不使用
+faceliveness	否	string	活体分数“0,0.9999”（表示第一个图不做活体检测、第二个图片活体分数为 0.9999）。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
//请求为四张图片，第三张解析失败
{
    "log_id": 73473737,
    "result_num":3,
    "result": [
        {
            "index_i": 0,
            "index_j": 1,
            "score": 44.3
        },
        {
            "index_i": 0,
            "index_j": 3,
            "score": 89.2
        },
        {
            "index_i": 1,
            "index_j": 3,
            "score": 10.4
        }
        ……
    ]
}
```

3.3.3 人脸识别

接口描述 用于计算指定组内用户，与上传图像中人脸的相似度。识别前提为您已经创建了一个人脸库。

典型应用场景：如人脸闸机，考勤签到，安防监控等。

> 说明：人脸识别返回值不直接判断是否是同一人，只返回用户信息及相似度分值。

> 说明：推荐可判断为同一人的相似度分值为80，您也可以根据业务需求选择更合适的阈值。

请求说明 举例，要计算一张图片与指定组group1, group2内各用户相似度：

```
public void identifyUser(AipFace client) {
    String path = "test1.jpg";
    HashMap<String, Object> options = new HashMap<String, Object>(1);
    options.put("user_top_num", 1);
    JSONObject res = client.identifyUser(Arrays.asList("group1","group2"), path, options);
    System.out.println(res.toString(2));
}
```

人脸识别请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id(由数字、字母、下划线组成)列表，每个groupid长度限制48
image	是	string	图像数据
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定: 目前支持faceliveness(活体检测)
user_top_num	否	number	返回用户top数，默认为1，最多返回5个

返回说明 | 字段 | 是否必选 | 类型 | 说明 | | ————-| —-| ————-| ————————–| | log_id | 是 | number | 请求唯一标识码，随机数 | | result_num | 是 | number | 返回结果数目，即：result数组中元素个数 | | ext_info | 否 | array | 对应参数

中的ext_fields | | +faceliveness | 否 | string | 活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间 | | result | 是 | array | 结果数组 | | +group_id | 是 | string | 对应的这个用户的group_id | | +uid | 是 | string | 匹配到的用户id | | +user_info | 是 | string | 注册时的用户信息 | | +scores | 是 | array | 结果数组，数组元素为匹配得分，top n。得分[0,100.0] | 返回样例：

```
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "group_id" : "test1",
            "uid": "u333333",
            "user_info": "Test User",
            "scores": [
                    99.3,
                    83.4
            ]
        }
    ]
}
```

3.3.4 人脸认证

接口描述 用于识别上传的图片是否为指定用户，即查找前需要先确定要查找的用户在人脸库中的id。

典型应用场景：如人脸登录，人脸签到等

> 说明：人脸认证与人脸识别的差别在于：人脸识别需要指定一个待查找的人脸库中的组；而人脸认证需要指定具体的用户id即可，不需要指定具体的人脸库中的组；实际应用中，人脸认证需要用户或系统先输入id，这增加了验证安全度，但也增加了复杂度，具体使用哪个接口需要视您的业务场景判断。

> 说明：请求参数中，新增在线活体检测

请求说明 举例，要认证一张图片在指定group中是否为uid1的用户：

```
public void verifyUser(AipFace client) {
    String path = ”test1.jpg”;
    HashMap<String, Object> options = new HashMap<String, Object>(1);
    options.put(”top_num”, 5);
```

```
    JSONObject res = client.verifyUser(”uid1”, Arrays.asList(”group1”,”group2”), path,
            options);
    System.out.println(res.toString(2));
}
```

人脸认证请求参数详情：

可选参数均放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图像数据
group_id	是	string	用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48
top_num	否	number	返回匹配得分top数，默认为1
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定: 目前支持faceliveness(活体检测)

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数组，数组元素为匹配得分，top n。得分范围[0,100.0]。推荐得分超过80可认为认证成功
ext_info	否	array	对应参数中的ext_fields

字段	是否必选	类型	说明
+faceliveness	否	string	活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
{
  "results": [
    93.86580657959,
    92.237548828125
  ],
  "result_num": 2,
  "log_id": 1629483134
}
```

3.3.5 人脸注册

接口描述 用于从人脸库中新增用户，可以设定多个用户所在组，及组内用户的人脸图片，

典型应用场景：构建您的人脸库，如会员人脸注册，已有用户补全人脸信息等。

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库
   |- 用户组一
      |- 用户01
         |- 人脸
      |- 用户02
         |- 人脸
         |- 人脸
         ....
       ....
   |- 用户组二
   |- 用户组三
   |- 用户组四
   ....
```

说明：关于人脸库的设置限制

- 每个开发者账号只能创建一个人脸库；
- 每个人脸库下，用户组（group）数量没有限制；
- 每个用户组（group）下，可添加最多300000张人脸，如每个uid注册一张人脸，则最多300000个用户uid；
- 每个用户（uid）所能注册的最大人脸数量没有限制；

说明：人脸注册完毕后，生效时间最长为35s，之后便可以进行识别或认证操作。

说明：注册的人脸，建议为用户正面人脸。

说明：uid在库中已经存在时，对此uid重复注册时，新注册的图片默认会追加到该uid下，如果手动选择`action_type:replace`，则会用新图替换库中该uid下所有图片。

请求说明 举例，要注册一个新用户，用户id为uid1，加入组id为group1和group2, 注册成功后服务端会返回操作的logid：

```
public void facesetAddUser(AipFace client) {
    // 参数为本地图片路径
    String path = ”picture1.jpg”;
    HashMap<String, String> options = new HashMap<String, String>();
    JSONObject res = client.addUser(”uid1”, ”test_user_info”, Arrays.asList(”group1”, ”
        group2”), path, options);
    System.out.println(res.toString(2));
}
```

人脸注册请求参数详情：

可选参数均以HashMap形式放在接口最后的options参数中。 | 参数 | 是否必选 | 类型 | 说明 | | ———– | —- | —— | ———————————- | | uid | 是 | string | 用户id（由数字、字母、下划线组成），长度限制128B | | user_info | 是 | string | 用户资料，长度限制256B | | imgPath/imgData| 是 | string | imgPath对应图片本地路径，imgData对应图片二进制数据，**要求图片base64编码后大小不超过10M** | | group_id | 是 | string | 用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48 | | action_type | 否 | string | 参数包含append、replace。**如果为“replace”，则每次注册时进行替换replace（新增或更新）操作，默认为append操作** |

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 注册成功
{
    "log_id": 73473737,
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

3.3.6 人脸更新

接口描述 用于对人脸库中指定用户，更新其下的人脸图像。

> 说明：针对一个uid执行更新操作，新上传的人脸图像将覆盖此uid原有所有图像。

> 说明：执行更新操作，如果该uid不存在时，会返回错误。如果添加了action_type:replace,则不会报错，并自动注册该uid，操作结果等同注册新用户。

请求说明 举例，要更新一个用户，用户id为uid1，更新成功后服务端会返回操作的logid：

```
public void facesetUpdateUser(AipFace client) {
    // 参数为本地图片路径
    String path = "picture1.jpg";
    HashMap<String, String> options = new HashMap<String, String>();
    JSONObject res = client.updateUser("uid1","user_info_memo","group1", path, options
        );
    System.out.println(res.toString(2));
}
```

人脸更新请求参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B

参数	是否必选	类型	说明
imgPath/imgData	是	string	imgPath对应图片本地路径，imgData对应图片二进制数据，**要求图片base64编码后大小不超过10M**
group_id	是	string	用户组id(由数字、字母、下划线组成)，长度限制48
user_info	是	string	新的user_info信息
action_type	否	string	** 如果为replace时，则uid不存在时，不报错，会自动注册。 不存在该参数时，如果uid不存在会提示错误**

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明　返回样例：

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

3.3.7　人脸删除

接口描述　用于从人脸库中删除一个用户。

人脸删除注意事项：

- 删除的内容，包括用户所有图像和身份信息；
- 如果一个uid存在于多个用户组内且没有指定group_id，将会同时将从各个组中把用户删除
- 如果指定了group_id，则只删除此group下的uid相关信息

请求说明 举例，要删除一个用户，用户id为uid1， 删除成功后服务端会返回操作的logid：

```
public void facesetDeleteUser(AipFace client) {
        // 只从指定组中删除用户
    JSONObject res = client.deleteUser(”uid1”, Arrays.asList(”group1”));
    System.out.println(res.toString(2));

        // 从人脸库中彻底删除用户
        JSONObject res = client.deleteUser(”uid1”);
    System.out.println(res.toString(2));
}
```

人脸删除请求参数：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	是	string	删除指定group_id中的uid信息

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
```

```
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

3.3.8 用户信息查询

接口描述 用于查询人脸库中某用户的详细信息。

请求说明 举例，要查询指定用户的信息：

```
public void getUser(AipFace client) {
        // 查询一个用户在所有组内的信息
    JSONObject res = client.getUser(”uid1”);
    System.out.println(res.toString(2));

        // 查询一个用户在指定组内的信息
    JSONObject res = client.getUser(”uid1”, Arrays.asList(”group1”));
    System.out.println(res.toString(2));
}
```

用户信息查询请求参数：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	否	string	选择指定group_id则只查找group列表下的uid内容，如果不指定则查找所有group下对应uid的信息

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result	是	array	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息

字段	是否必选	类型	说明
+groups	是	array	用户所属组列表

返回说明 返回样例：

```
{
    "result": {
        "uid": "testuser2",
        "user_info": "registed user info ...",
        "groups": [
            "grp1",
            "grp2",
            "grp3"
        ]
    },
    "log_id": 2979357502
}
```

3.3.9 组列表查询

接口描述 用于查询用户组的列表。

请求说明 举例：

```
public void getGroupList(AipFace client) {
        HashMap<String, Object> options = new HashMap<String, Object>(2);
    options.put(”start”, 0);
    options.put(”num”, 10);
    JSONObject res = client.getGroupList(options);
    System.out.println(res.toString(2));
}
```

组列表查询请求参数详情：

参数	是否必选	类型	说明
start	否	number	默认值0，起始序号
end	否	number	返回数量，默认值100，最大值1000

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	group_id列表

返回说明 返回样例：

```
{
    "result_num": 2,
    "result": [
        "grp1",
        "grp2"
    ],
    "log_id": 3314921889
}
```

3.3.10 组内用户列表查询

接口描述 用于查询指定用户组中的用户列表。

请求说明 举例：

```
public void getGroupUsers(AipFace client) {
        HashMap<String, Object> options = new HashMap<String, Object>(2);
    options.put(”start”, 0);
    options.put(”num”, 10);
    JSONObject res = client.getGroupUsers(”group1”, options);
    System.out.println(res.toString(2));
}
```

组内用户列表查询请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	number	默认值0，起始序号
end	否	number	返回数量，默认值100，最大值1000

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回说明 返回样例：

```
{
    "log_id": 3314921889,
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

3.3.11 组间复制用户

接口描述 用于将已经存在于人脸库中的用户添加到一个新的组。

> 说明：并不是向一个指定组内添加用户，而是直接从其它组复制用户信息

请求说明 举例：

```
public void addGroupUser(AipFace client) {
    JSONObject res = client.addGroupUser(”srcgroup”, Arrays.asList(”dstGroup1”,”
        dstGroup2”),”uid1”);
    System.out.println(res.toString(2));
}
```

组内添加用户请求参数详情：

参数	是否必选	类型	说明
src_group_id	是	string	从指定group里复制信息
group_id	是	string	需要添加信息的组id列表
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明　返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
  "error_msg": "already add"
}
```

3.3.12　组内删除用户

接口描述　用于将用户从某个组中删除，但不会删除用户在其它组的信息。

> 说明：当用户仅属于单个分组时，本接口将返回错误，请使用人脸删除接口

请求说明　举例：

```
public void deleteGroupUser(AipFace client) {
    JSONObject res = client.deleteGroupUser(Arrays.asList(”group1”,”group2”),”uid1”);
    System.out.println(res.toString(2));
}
```

组内删除用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id列表
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
  "error_msg": "user must be in one group at least"
}
```

3.4 错误信息

3.4.1 错误返回格式

若请求错误，服务器将返回的JSON文本包含以下参数：

- error_code：错误码。
- error_msg：错误描述信息，帮助理解和解决发生的错误。

3.4.2 错误码

SDK本地检测参数返回的错误码：

error_code	error_msg	备注
SDK100	image size error	图片大小超限

error_code	error_msg	备注
SDK101	image length error	图片边长不符合要求
SDK102	read image file error	读取图片文件错误
SDK103	user_info size error	user_info参数大小错误
SDK104	group_id format error	group_id有非法字符
SDK105	group_id size error	group_id参数大小错误
SDK106	uid format error	uid有非法字符
SDK107	uid size error	uid参数大小错误
SDK108	connection or read data time out	连接超时或读取数据超时
SDK109	unsupported image format	不支持的图片格式

服务端返回的错误码

错误码	错误信息	描述
4	Open api request limit reached	集群超限额
17	Open api daily request limit reached	每天流量超限额
18	Open api qps request limit reached	QPS超限额
19	Open api total request limit reached	请求总量超限额
100	Invalid parameter	无效的access_token参数
110	Access token invalid or no longer valid	Access Token失效
111	Access token expired	Access token过期
216100	invalid param	参数异常，具体异常原因详见error_msg
216101	not enough param	缺少必须的参数，具体异常原因详见error_msg
216102	service not support	请求了不支持的服务，请检查调用的url
216103	param too long	请求超长，一般为一次传入图片个数超过系统限制
216110	appid not exist	appid不存在，请重新检查后台应用列表中的应用信息

错误码	错误信息	描述
216111	invalid userid	userid信息非法，请检查对应的参数
216200	empty image	图片为空或者base64解码错误
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	数据库异常，**少量发生时重试即可**
216400	backend error	后端识别服务异常，可以根据具体msg查看错误原因
216402	face not found	未找到人脸，请检查图片是否含有人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在，请确认该用户是否注册或注册已经生效(**需要已经注册超过35s**)
216613	fail to delete user record	删除用户图片记录失败，重试即可
216614	not enough images	两两比对中图片数少于2张，无法比较
216615	fail to process images	服务处理该图片失败，发生后重试即可
216616	image existed	图片已存在
216617	fail to add user	新增用户图片失败
216618	no user in group	组内用户为空，确认该group是否存在或已经生效**(需要已经注册超过35s)**
216631	request add user overlimit	本次请求添加的用户数量超限

第4章

Node SDK文档

4.1 简介

Hi，您好，欢迎使用百度人脸识别API服务。

本文档主要针对API开发者，描述百度人脸识别接口服务的相关技术内容。如果您对文档内容有任何疑问，可以通过以下几种方式联系我们：

- 在百度云控制台内提交工单，咨询问题类型请选择人工智能服务；
- 加入开发者QQ群：224994340

4.1.1 接口能力

接口名称	接口能力简要描述
人脸检测	检测人脸并定位，返回五官关键点，及人脸各属性值
人脸比对	返回两两比对的人脸相似值
人脸识别	在人脸库中查找相似的人脸
人脸认证	识别上传的图片是否为指定用户
人脸库设置	对人脸库的相关操作，如注册、删除、更新、查找用户信息等

4.1.2 版本更新记录

上线日期	版本号	更新内容
2017.5.11	0.1.0	初版

4.2 快速入门

4.2.1 安装人脸检测 Node SDK

Face Node SDK目录结构

```
├── src
│   ├── auth                    //授权相关类
│   ├── http                    //Http通信相关类
│   ├── client                  //公用类
│   ├── util                    //工具类
│   └── const                   //常量类
├── AipFace.js                  //人脸识别交互类
├── index.js                    //入口文件
└── package.json                //npm包描述文件
```

支持 node 版本 4.0+

直接使用node开发包步骤如下：

1.在官方网站下载node SDK压缩包。

2.将下载的aip-node-sdk-version.zip解压后，复制到工程文件夹中。

3.进入目录，运行npm install安装sdk依赖库

4.把目录当做模块依赖

其中，version为版本号，添加完成后，用户就可以在工程中使用人脸识别 Node SDK。

直接使用npm安装依赖：

暂无

4.2.2 新建AipFaceClient

AipFaceClient是人脸识别的Node客户端，为使用人脸识别的开发人员提供了一系列的交互方法。

用户可以参考如下代码新建一AipFaceClient：

```
var AipFace = require("baidu-ai").face;

// 设置APPID/AK/SK
var APP_ID = "你的 App ID";
```

```
var API_KEY = "你的 Api ID";
var SECRET_KEY = "你的 Secret Key";

var client = new AipFace(APP_ID, API_KEY, SECRET_KEY);
```

在上面代码中，常量APP_ID在百度云控制台中创建，常量API_KEY与SECRET_KEY是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在AI服务控制台中的应用列表中查看。

注意：如您以前是百度云的老用户，其中API_KEY对应百度云的“Access Key ID”，SECRET_KEY对应百度云的“Access Key Secret”。

4.3 接口调用

4.3.1 人脸检测

接口描述 检测请求图片中的人脸，返回人脸位置、72个关键点坐标、及人脸相关属性信息。

检测响应速度，与图片中人脸数量相关，人脸数量较多时响应时间会有些许延长。

典型应用场景：如人脸属性分析，基于人脸关键点的加工分析，人脸营销活动等。

> 五官位置会标记具体坐标；72个关键点坐标也包含具体坐标，但不包含对应位置的详细位置描述。

请求说明 图片接受类型支持本地图片路径字符串，图片文件二进制数组。

举例，要对一张图片进行人脸识别，具体的人脸信息在返回的result字段中：

```
var fs = require('fs');
var image = fs.readFileSync('assets/face/face.jpg');
var base64Img = new Buffer(image).toString('base64');

client.detect(base64Img).then(function(result) {
console.log(JSON.stringify(result));
});
```

传入图片时还想增加一些自定义参数配置：

```
var fs = require('fs');
var image = fs.readFileSync('assets/face/face.jpg');
var base64Img = new Buffer(image).toString('base64');

client.detect(base64Img, {max_face_num: 1}).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸检测 请求参数详情

参数	类型	描述	是否必须
face_fields	string	包括age、beauty、expression、face-shape、gender、glasses、land-mark、race、qualities 信息，逗号分隔，默认只返回人脸框、概率和旋转角度。	否
max_face_num	number	最多处理人脸数目，默认值1	是
image	string	图像数据	是

参数	类型	是否一定输出	描述
log_id	number	是	日志id
result_num	number	是	人脸数目
result	array	是	人脸属性对象的集合
+age	number	否	年龄。face_fields包含age时返回
+beauty	number	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+location	array	是	人脸在图片中的位置
++left	number	是	人脸区域离左边界的距离

参数	类型	是否一定输出	描述
++top	number	是	人脸区域离上边界的距离
++width	number	是	人脸区域的宽度
++height	number	是	人脸区域的高度
+face_probability	number	是	人脸置信度，范围0-1
+rotation_angle	number	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	number	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	number	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	number	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	number	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	number	否	表情置信度，范围0~1。face_fields包含expression时返回
+faceshape	array	否	脸型置信度。face_fields包含faceshape时返回
++type	string	是	脸型：square/ triangle/ oval/ heart/ round
++probability	number	是	置信度：0~1
+gender	string	否	male、female。face_fields包含gender时返回
+gender_probability	number	否	性别置信度，范围0~1。face_fields包含gender时返回

参数	类型	是否一定输出	描述
+glasses	number	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回
+glasses_probability	number	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	array	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+landmark72	array	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+race	string	否	yellow、white、black、arabs。face_fields包含race时返回
+race_probability	number	否	人种置信度，范围0~1。face_fields包含race时返回
+qualities	array	否	人脸质量信息。face_fields包含qualities时返回
++occlusion	array	是	人脸各部分遮挡的概率，[0, 1]（待上线）
+++left_eye	number	是	左眼
+++right_eye	number	是	右眼
+++nose	number	是	鼻子
+++mouth	number	是	嘴

参数	类型	是否一定输出	描述
+++left_cheek	number	是	左脸颊
+++right_cheek	number	是	右脸颊
+++chin	number	是	下巴
++type	array	是	真实人脸/卡通人脸置信度
+++human	number	是	真实人脸置信度，[0, 1]
+++cartoon	number	是	卡通人脸置信度，[0, 1]

返回说明

4.3.2 人脸比对

接口描述 该请求用于比对多张图片中的人脸相似度并返回两两比对的得分，可用于判断两张脸是否是同一人的可能性大小。

典型应用场景：如人证合一验证，用户认证等，可与您现有的人脸库进行比对验证。

> 说明：支持对比对的两张图片做在线活体检测

请求说明 接受的参数为一系列本地图片路径的数组，或图片二进制数据的数组。 举例，要对两张图片进行人脸比对，具体的人脸信息在返回的result字段中：

```
var fs = require('fs');
var image1 = fs.readFileSync('assets/face/face1.jpg');
var image2 = fs.readFileSync('assets/face/face2.jpg');
var base64Img1 = new Buffer(image1).toString('base64');
var base64Img2 = new Buffer(image2).toString('base64');

client.match([base64Img1, base64Img2]).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸比对请求参数：

所有图片经base64编码后的图片数据总和不超过10M。以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
ext_fields	否	string	返回质量信息，取值固定: 目前支持qualities(质量检测)。(对所有图片都会做改处理)
image_liveness	否	string	返回的活体信息，“faceliveness,faceliveness”表示对比对的两张图片都做活体检测；“,faceliveness” 表示对第一张图片不做活体检测、第二张图做活体检测；“faceliveness,” 表示对第一张图片做活体检测、第二张图不做活体检测

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。 得分[0,100.0]
+index_i	是	number	比对图片1的index
+index_j	是	number	比对图片2的index
+score	是	double	比对得分
ext_info	否	array	对应参数中的ext_fields

字段	是否必选	类型	说明
+qualities	否	string	质量相关的信息，无特殊需求可以不使用
+faceliveness	否	string	活体分数“0,0.9999”（表示第一个图不做活体检测、第二个图片活体分数为 0.9999）。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明

4.3.3 人脸识别

接口描述 用于计算指定组内用户，与上传图像中人脸的相似度。识别前提为您已经创建了一个人脸库。

典型应用场景：如人脸闸机，考勤签到，安防监控等。

> 说明：人脸识别返回值不直接判断是否是同一人，只返回用户信息及相似度分值。

> 说明：推荐可判断为同一人的相似度分值为80，您也可以根据业务需求选择更合适的阈值。

请求说明 举例，要计算一张图片与指定组group1, group2内各用户相似度：

```
var fs = require('fs');
var image = fs.readFileSync('assets/face/user1.jpg');
var base64Img = new Buffer(image).toString('base64');
var groupIds = ['group1', 'group2'];

client.identifyUser(groupIds, base64Img, {usertopnum: 5}).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸识别请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48
image	是	string	图像数据
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)
user_top_num	否	number	返回用户top数，默认为1，最多返回5个

返回说明 ｜字段｜是否必选｜类型｜说明｜｜————-｜—-｜————-｜————————–｜｜log_id｜是｜number｜请求唯一标识码，随机数｜｜result_num｜是｜number｜返回结果数目，即：result数组中元素个数｜｜ext_info｜否｜array｜对应参数中的ext_fields｜｜+faceliveness｜否｜string｜活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间｜｜result｜是｜array｜结果数组｜｜+group_id｜是｜string｜对应的这个用户的group_id｜｜+uid｜是｜string｜匹配到的用户id｜｜+user_info｜是｜string｜注册时的用户信息｜｜+scores｜是｜array｜结果数组，数组元素为匹配得分，top n。得分[0,100.0]｜返回样例：

```
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "group_id" : "test1",
            "uid": "u333333",
            "user_info": "Test User",
            "scores": [
                    99.3,
                    83.4
            ]
        }
    ]
}
```

4.3.4 人脸认证

接口描述 用于识别上传的图片是否为指定用户，即查找前需要先确定要查找的用户在人脸库中的id。

典型应用场景：如人脸登录，人脸签到等

> 说明：人脸认证与人脸识别的差别在于：人脸识别需要指定一个待查找的人脸库中的组；而人脸认证需要指定具体的用户id即可，不需要指定具体的人脸库中的组；实际应用中，人脸认证需要用户或系统先输入id，这增加了验证安全度，但也增加了复杂度，具体使用哪个接口需要视您的业务场景判断。

> 说明：请求参数中，新增在线活体检测

请求说明 举例，要认证一张图片在指定group中是否为uid1的用户：

```
var fs = require('fs');

var image = fs.readFileSync('assets/face/user1.jpg');
var base64Img = new Buffer(image).toString('base64');

var uid = 'uid1';
var groupIds = ['group1', 'group2'];

client.verifyUser(uid, groupIds, base64Img, {top_num: 1}).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸认证请求参数详情：

可选参数均放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图像数据
group_id	是	string	用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48

参数	是否必选	类型	说明
top_num	否	number	返回匹配得分top数，默认为1
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数组，数组元素为匹配得分，top n。得分范围[0,100.0]。推荐得分超过80可认为认证成功
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
{
  "results": [
    93.86580657959,
    92.237548828125
  ],
  "result_num": 2,
  "log_id": 1629483134
}
```

4.3.5 人脸注册

接口描述 用于从人脸库中新增用户，可以设定多个用户所在组，及组内用户的人脸图片，

典型应用场景：构建您的人脸库，如会员人脸注册，已有用户补全人脸信息等。

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库
   |- 用户组一
      |- 用户01
         |- 人脸
      |- 用户02
         |- 人脸
         |- 人脸
         ....
       ....
   |- 用户组二
   |- 用户组三
   |- 用户组四
   ....
```

说明：关于人脸库的设置限制

- 每个开发者账号只能创建一个人脸库；
- 每个人脸库下，用户组（group）数量没有限制；
- 每个用户组（group）下，可添加最多300000张人脸，如每个uid注册一张人脸，则最多300000个用户uid；
- 每个用户（uid）所能注册的最大人脸数量没有限制；

> 说明：人脸注册完毕后，生效时间最长为35s，之后便可以进行识别或认证操作。

> 说明：注册的人脸，建议为用户正面人脸。

> 说明：uid在库中已经存在时，对此uid重复注册时，新注册的图片默认会追加到该uid下，如果手动选择`action_type:replace`，则会用新图替换库中该uid下所有图片。

请求说明 举例，要注册一个新用户，用户id为uid1，加入组id为group1和group2, 注册成功后服务端会返回操作的logid：

```
var fs = require('fs');
```

```
var image = fs.readFileSync('assets/face/user1.jpg');
var base64Img = new Buffer(image).toString('base64');

var uid = 'uid1';
var userInfo = '张三';
var groupIds = ['group1', 'group2'];

client.addUser(uid, userInfo, groupIds, base64Img).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸注册请求参数要求：

所有图片经base64编码后的图片数据总和不超过10M。

人脸注册返回数据参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制48
user_info	**是**	string	新的user_info信息
action_type	否	string	如果为replace时，则uid不存在时，不报错，会自动注册。 不存在该参数时，如果uid不存在会提示错误

返回说明 ｜字段｜是否必选｜类型｜说明｜｜——｜—-｜——｜————｜｜log_id｜是｜number｜请求标识码，随机数，唯一｜返回样例：

```
// 注册成功
{
    "log_id": 73473737,
```

```
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

4.3.6 人脸更新

接口描述 用于对人脸库中指定用户，更新其下的人脸图像。

> 说明：针对一个uid执行更新操作，新上传的人脸图像将覆盖此uid原有所有图像。

> 说明：执行更新操作，如果该uid不存在时，会返回错误。如果添加了action_type:replace,则不会报错，并自动注册该uid，操作结果等同注册新用户。

请求说明 举例，要更新一个在group1中的用户，用户id为uid1， 更新成功后服务端会返回操作的logid：

```
var fs = require('fs');
var image = fs.readFileSync('assets/face/user1.jpg');
var base64Img = new Buffer(image).toString('base64');

var uid = 'user1';
var userInfo = '李四';
var groupId = 'group1';

client.updateUser(uid, userInfo, groupId, base64Img).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸更新请求参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据

参数	是否必选	类型	说明
group_id	是	string	用户组id(由数字、字母、下划线组成)，长度限制48
user_info	**是**	string	新的user_info信息

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

4.3.7 人脸删除

接口描述 用于从人脸库中删除一个用户。

人脸删除注意事项：

- 删除的内容，包括用户所有图像和身份信息；
- 如果一个uid存在于多个用户组内且没有指定group_id，将会同时将从各个组中把用户删除
- 如果指定了group_id，则只删除此group下的uid相关信息

请求说明 举例，要删除一个用户，用户id为uid1， 删除成功后服务端会返回操作的logid：

```
var uid = 'user1';
```

```
client.deleteUser(uid).then(function(result) {
console.log(JSON.stringify(result));
});
```

增加一些自定义参数配置:

```
client.deleteUser(uid, {group_id: 'group1'}).then(function(result) {
console.log(JSON.stringify(result));
});
```

人脸删除请求参数要求:

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	是	string	删除指定group_id中的uid信息

返回说明　人脸删除返回数据参数详情:

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回样例:

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

4.3.8 用户信息查询

接口描述 用于查询人脸库中某用户的详细信息。

请求说明 举例，要查询指定用户的信息：

```
var uid = 'user1';

client.getUser(uid).then(function(result) {
console.log(JSON.stringify(result));
});

// 查询指定group下的用户信息:
client.getUser('uid1',{group_id:'group1'}).then(function(result) {
    console.log(JSON.stringify(result));
});
```

用户信息查询请求参数要求：

以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	否	string	选择指定group_id则只查找group列表下的uid内容，如果不指定则查找所有group下对应uid的信息

用户信息查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result	是	array	结果数组
+uid	是	string	匹配到的用户id

字段	是否必选	类型	说明
+user_info	是	string	注册时的用户信息
+groups	是	array	用户所属组列表

返回样例：

```
{
    "result": {
        "uid": "testuser2",
        "user_info": "registed user info ...",
        "groups": [
            "grp1",
            "grp2",
            "grp3"
        ]
    },
    "log_id": 2979357502
}
```

4.3.9 组列表查询

组列表查询接口用于查询一个app下所有组的列表。

举例：

```
// 查询全部组列表
client.getGrouplist().then(function(result) {
console.log(JSON.stringify(result));
});

// 查询指定范围组列表
client.getGrouplist({start: 5, num: 10}).then(function(result) {
console.log(JSON.stringify(result));
});
```

组列表查询请求参数详情：

参数	是否必选	类型	说明
start	否	number	默认值0，起始序号

参数	是否必选	类型	说明
num	否	number	返回数量，默认值100，最大值1000

组列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	group_id列表

返回样例：

```
{
    "result_num": 2,
    "result": [
        "grp1",
        "grp2"
    ],
    "log_id": 3314921889
}
```

4.3.10 组内用户列表查询

接口描述 用于查询指定用户组中的用户列表。

请求说明 举例：

```
var groupId = 'group1';

// 使用默认参数
client.getGroupUsers(groupId).then(function(result) {
console.log(JSON.stringify(result));
});

// 查询组内指定范围用户列表
client.getGroupUsers(groupId, {start: 5, end: 10}).then(function(result) {
```

```
console.log(JSON.stringify(result));
});
```

组内用户列表查询请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	number	默认值0，起始序号
num	否	number	返回数量，默认值100，最大值1000

组内用户列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回样例：

```
{
    "log_id": 3314921889,
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

4.3.11 组间复制用户

接口描述 用于将已经存在于人脸库中的用户添加到一个新的组。

> 说明：并不是向一个指定组内添加用户，而是直接从其它组复制用户信息

请求说明 举例：

```
var uid = 'uid2';
var groupIds = ['group1', 'group2'];
var groupIdSrc = 'group3';

client.addGroupUsers(groupIdSrc, groupId, uid).then(function(result) {
console.log(JSON.stringify(result));
});
```

组间复制用户请求参数详情：

参数	是否必选	类型	说明
src_group_id	是	string	从指定group里复制信息
group_id	是	string	需要添加信息的组id列表
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
```

```
  "error_msg": "already add"
}
```

4.3.12 组内删除用户

接口描述 用于将用户从某个组中删除，但不会删除用户在其它组的信息。

说明：当用户仅属于单个分组时，本接口将返回错误，请使用人脸删除接口

请求说明 举例：

```
var uid = 'uid1';
var groupIds = ['group1', 'group2'];

client.deleteGroupUsers(groupIds , uid).then(function(result) {
console.log(JSON.stringify(result));
});
```

组内删除用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
```

```
  "error_msg": "user must be in one group at least"
}
```

4.4 错误信息

4.4.1 错误返回格式

若请求错误，服务器将返回的JSON文本包含以下参数：

- error_code：错误码；关于错误码的详细信息请参考通用错误码和业务相关错误码。
- error_msg：错误描述信息，帮助理解和解决发生的错误。

4.4.2 错误码

服务端返回的错误码

错误码	错误信息	描述
4	Open api request limit reached	集群超限额
17	Open api daily request limit reached	每天流量超限额
18	Open api qps request limit reached	QPS超限额
19	Open api total request limit reached	请求总量超限额
100	Invalid parameter	无效的access_token参数
110	Access token invalid or no longer valid	Access Token失效
111	Access token expired	Access token过期
216100	invalid param	参数异常，具体异常原因详见error_msg
216101	not enough param	缺少必须的参数，具体异常原因详见error_msg
216102	service not support	请求了不支持的服务，请检查调用的url
216103	param too long	请求超长，一般为一次传入图片个数超过系统限制

错误码	错误信息	描述
216110	appid not exist	appid不存在，请重新检查后台应用列表中的应用信息
216111	invalid userid	userid信息非法，请检查对应的参数
216200	empty image	图片为空或者base64解码错误
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	数据库异常，**少量发生时重试即可**
216400	backend error	后端识别服务异常，可以根据具体msg查看错误原因
216402	face not found	未找到人脸，请检查图片是否含有人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在，请确认该用户是否注册或注册已经生效(**需要已经注册超过35s**)
216613	fail to delete user record	删除用户图片记录失败，重试即可
216614	not enough images	两两比对中图片数少于2张，无法比较
216615	fail to process images	服务处理该图片失败，发生后重试即可
216616	image existed	图片已存在
216617	fail to add user	新增用户图片失败
216618	no user in group	组内用户为空，确认该group是否存在或已经生效**(需要已经注册超过35s)**
216631	request add user overlimit	本次请求添加的用户数量超限

第5章

PHP SDK文档

5.1 简介

Hi，您好，欢迎使用百度人脸识别API服务。

本文档主要针对API开发者，描述百度人脸识别接口服务的相关技术内容。如果您对文档内容有任何疑问，可以通过以下几种方式联系我们：

- 在百度云控制台内提交工单，咨询问题类型请选择人工智能服务；
- 加入开发者QQ群：224994340；

5.1.1 接口能力

接口名称	接口能力简要描述
人脸检测	检测人脸并定位，返回五官关键点，及人脸各属性值
人脸比对	返回两两比对的人脸相似值
人脸识别	在人脸库中查找相似的人脸
人脸认证	识别上传的图片是否为指定用户
人脸库设置	对人脸库的相关操作，如注册、删除、更新、查找用户信息等

5.2 快速入门

5.2.1 安装人脸检测 PHP SDK

人脸检测 PHP SDK目录结构

```
BFR
        ├── AipFace.php                     //人脸属性检测类
        ├── lib
        │       ├── AipHttpClient.php      //内部http请求类
        │       ├── AipBCEUtil.php         //内部工具类
        │       ├── AipBase                //Aip基类
        └── demo
                ├── DemoAipFace.php         //人脸属性检测服务示例
                └── face.jpg                //人脸图片
```

支持 PHP版本：5.3+

使用SDK步骤如下：

1.在官方网站下载PHP SDK压缩工具包。

2.将下载的aip-face-php-sdk-version.zip解压后，复制AipFace.php以及lib/*到工程文件夹中。

3.引入AipFace.php

5.2.2 初始化一个AipFace对象

AipFace类是与人脸检测交互的客户端，为使用人脸检测的开发人员提供了一系列的交互方法。

用户可以参考如下代码新建一个AipFace对象：

```
// 引入人脸属性检测 SDK
require_once 'AipFace.php';

// 定义常量
const APP_ID = '你的 App ID'
const API_KEY = '你的 API Key';
const SECRET_KEY = '你的 Secret Key';

// 初始化AipFace对象
\$aipFace = new AipFace(APP_ID, API_KEY, SECRET_KEY);
```

在上面代码中，常量APP_ID在百度云控制台中创建，常量API_KEY与SECRET_KEY是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在AI服务控制台中的应用列表中查看。

注意：如您以前是百度云的老用户，其中API_KEY对应百度云的“Access Key ID”，SECRET_KEY对应百度云的“Access Key Secret”。

5.2.3 配置AipFace

如果用户需要配置AipFace的一些细节参数(一般不需要配置)，可以在构造AipFace之后调用接口设置参数，目前只支持以下参数：

接口	说明
setConnectionTi meoutInMillis	建立连接的超时时间（单位：毫秒
setSocketTimeou tInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）

5.3 接口调用

5.3.1 人脸检测

接口描述 检测请求图片中的人脸，返回人脸位置、72个关键点坐标、及人脸相关属性信息。

检测响应速度，与图片中人脸数量相关，人脸数量较多时响应时间会有些许延长。

典型应用场景：如人脸属性分析，基于人脸关键点的加工分析，人脸营销活动等。

> 五官位置会标记具体坐标；72个关键点坐标也包含具体坐标，但不包含对应位置的详细位置描述。

请求说明 图片接受类型支持本地图片路径字符串，图片文件二进制数组。

举例，要对一张图片进行人脸识别，具体的人脸信息在返回的result字段中：

```
// 引入人脸属性检测 SDK
require_once ’ AipFace.php’ ;

// 定义常量
const APP_ID = ’ 你的␣App␣ID’
const API_KEY = ’ 你的␣API␣Key’ ;
const SECRET_KEY = ’ 你的␣Secret␣Key’ ;

// 初始化AipFace
\$aipFace = new AipFace(APP_ID, API_KEY, SECRET_KEY);
```

```
// 调用人脸属性识别接口
\$result = \$aipFace->detect(file_get_contents('face.jpg'));
```

传入图片时还想增加一些自定义参数配置：

```
// 引入人脸属性检测 SDK
require_once 'AipFace.php';

// 定义常量
const APP_ID = '你的 App ID';
const API_KEY = '你的 API Key';
const SECRET_KEY = '你的 Secret Key';

// 初始化ApiFace
\$aipFace = new AipFace(APP_ID, API_KEY, SECRET_KEY);

// 定义参数变量
\$option = array(
  'max_face_num' => 1,
  'face_fields' => 'expression'
);

// 调用人脸属性识别接口
\$result = \$aipFace->detect(file_get_contents('face.jpg'), \$option);
```

人脸检测 请求参数详情

参数	类型	描述	是否必须
face_fields	string	包括age、beauty、expression、face-shape、gender、glasses、landmark、race、qualities 信息，逗号分隔，默认只返回人脸框、概率和旋转角度。	否
max_face_num	number	最多处理人脸数目，默认值1	是
image	string	图像数据	是

参数	类型	是否一定输出	描述
log_id	number	是	日志id
result_num	number	是	人脸数目

参数	类型	是否一定输出	描述
result	array	是	人脸属性对象的集合
+age	number	否	年龄。face_fields包含age时返回
+beauty	number	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+location	array	是	人脸在图片中的位置
++left	number	是	人脸区域离左边界的距离
++top	number	是	人脸区域离上边界的距离
++width	number	是	人脸区域的宽度
++height	number	是	人脸区域的高度
+face_probability	number	是	人脸置信度，范围0-1
+rotation_angle	number	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	number	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	number	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	number	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	number	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	number	否	表情置信度，范围0~1。face_fields包含expression时返回

参数	类型	是否一定输出	描述
+faceshape	array	否	脸型置信度。face_fields包含faceshape时返回
++type	string	是	脸型：square/ triangle/ oval/ heart/ round
++probability	number	是	置信度：0~1
+gender	string	否	male、female。face_fields包含gender时返回
+gender_probability	number	否	性别置信度，范围0~1。face_fields包含gender时返回
+glasses	number	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回
+glasses_probability	number	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	array	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+landmark72	array	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+race	string	否	yellow、white、black、arabs。face_fields包含race时返回

参数	类型	是否一定输出	描述
+race_probability	number	否	人种置信度，范围0~1。 face_fields 包含race时返回
+qualities	array	否	人脸质量信息。face_fields 包含qualities时返回
++occlusion	array	是	人脸各部分遮挡的概率， [0, 1]（待上线）
+++left_eye	number	是	左眼
+++right_eye	number	是	右眼
+++nose	number	是	鼻子
+++mouth	number	是	嘴
+++left_cheek	number	是	左脸颊
+++right_cheek	number	是	右脸颊
+++chin	number	是	下巴
++type	array	是	真实人脸/卡通人脸置信度
+++human	number	是	真实人脸置信度，[0, 1]
+++cartoon	number	是	卡通人脸置信度，[0, 1]

返回说明

5.3.2 人脸两两比对

接口描述 该请求用于比对多张图片中的人脸相似度并返回两两比对的得分，可用于判断两张脸是否是同一人的可能性大小。

典型应用场景：如人证合一验证，用户认证等，可与您现有的人脸库进行比对验证。

> 说明：支持对比对的两张图片做在线活体检测

请求说明 举例，要对一组人脸图片进行两两比对，示例代码如下：

```
// 引入人脸识别 SDK
```

```
require_once ’AipFace.php’;

// 定义常量
const APP_ID = ’你的␣App␣ID’
const API_KEY = ’你的␣API␣Key’;
const SECRET_KEY = ’你的␣Secret␣Key’;

// 初始化ApiFace
\$aipFace = new AipFace(APP_ID, API_KEY, SECRET_KEY);

// 调用人脸两两比对接口
\$result = \$aipFace->match(array(
      file_get_contents(’face1.jpg’),
   file_get_contents(’face2.jpg’),
   file_get_contents(’face3.jpg’),
   file_get_contents(’face4.jpg’),
));
```

人脸比对请求参数：

所有图片经base64编码后的图片数据总和不超过10M。以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
ext_fields	否	string	返回质量信息，取值固定: 目前支持qualities(质量检测)。(对所有图片都会做改处理)
image_liveness	否	string	返回的活体信息，“faceliveness,faceliveness”表示对比对的两张图片都做活体检测；“,faceliveness” 表示对第一张图片不做活体检测、第二张图做活体检测；“faceliveness,” 表示对第一张图片做活体检测、第二张图不做活体检测

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。 得分[0,100.0]
+index_i	是	number	比对图片1的index
+index_j	是	number	比对图片2的index
+score	是	double	比对得分
ext_info	否	array	对应参数中的ext_fields
+qualities	否	string	质量相关的信息，无特殊需求可以不使用
+faceliveness	否	string	活体分数“0,0.9999”（表示第一个图不做活体检测、第二个图片活体分数为 0.9999）。 活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
//请求为四张图片，第三张解析失败
{
    "log_id": 73473737,
    "result_num":3,
    "result": [
        {
            "index_i": 0,
```

```
            "index_j": 1,
            "score": 44.3
        },
        {
            "index_i": 0,
            "index_j": 3,
            "score": 89.2
        },
        {
            "index_i": 1,
            "index_j": 3,
            "score": 10.4
        }
        ……
    ]
}
```

5.3.3 人脸识别

接口描述 用于计算指定组内用户，与上传图像中人脸的相似度。识别前提为您已经创建了一个人脸库。

典型应用场景：如人脸闸机，考勤签到，安防监控等。

> 说明：人脸识别返回值不直接判断是否是同一人，只返回用户信息及相似度分值。

> 说明：推荐可判断为同一人的相似度分值为80，您也可以根据业务需求选择更合适的阈值。

请求说明 举例，要计算一张图片与指定组group1内用户相似度：

```
## 该参数如果不填则使用默认值
\$options = array(
        'user_top_num' => 1,
  'face_top_num' => 1,
);

\$aipFace->identifyUser(
                                    'group1',
            file_get_content('face.jpg'),
                                            \$options
                                    );
```

人脸识别请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id(由数字、字母、下划线组成)列表，每个groupid长度限制48
image	是	string	图像数据
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)
user_top_num	否	number	返回用户top数，默认为1，最多返回5个

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体(测试期间
result	是	array	结果数组
+group_id	是	string	对应的这个用户的group_id
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+scores	是	array	结果数组，数组元素为匹配得分，top n。得分[0,100.0]

返回说明 返回样例：

```
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "group_id" : "group1",
            "uid": "123",
            "user_info": "user info",
            "scores": [
                    99.3,
                    83.4
            ]
        }
    ]
}
```

5.3.4 人脸认证

接口描述 用于识别上传的图片是否为指定用户，即查找前需要先确定要查找的用户在人脸库中的id。

典型应用场景：如人脸登录，人脸签到等

> 说明：人脸认证与人脸识别的差别在于：人脸识别需要指定一个待查找的人脸库中的组；而人脸认证需要指定具体的用户id即可，不需要指定具体的人脸库中的组；实际应用中，人脸认证需要用户或系统先输入id，这增加了验证安全度，但也增加了复杂度，具体使用哪个接口需要视您的业务场景判断。

> 说明：请求参数中，新增在线活体检测

请求说明 举例，要认证一张图片在指定group中是否为uid1的用户：

```
## 该参数如果不填则使用默认值
\$options = array(
        'top_num' => 1,
);

\$aipFace->verifyUser(
                                    'uid1',
                                    'group1',
                        file_get_content('face.jpg'),
                                    \$options
                                    );
```

人脸认证请求参数详情：

可选参数均放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图像数据
group_id	是	string	用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48
top_num	否	number	返回匹配得分top数，默认为1
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)

返回说明 ｜字段｜是否必选｜类型｜说明｜｜————-｜—-｜————-｜————————————-｜｜log_id｜是｜number｜请求唯一标识码，随机数｜｜result_num｜是｜number｜返回结果数目，即：result数组中元素个数｜｜result｜是｜array｜结果数组，数组元素为匹配得分，top n。 得分范围[0,100.0]。推荐得分超过80可认为认证成功｜｜ext_info｜否｜array｜对应参数中的ext_fields｜｜+faceliveness｜否｜string｜活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间）｜返回样例：

```
{
  "results": [
    93.86580657959,
    92.237548828125
  ],
  "result_num": 2,
  "log_id": 1629483134
}
```

5.3.5 人脸注册

接口描述 用于从人脸库中新增用户，可以设定多个用户所在组，及组内用户的人脸图片，

典型应用场景：构建您的人脸库，如会员人脸注册，已有用户补全人脸信息等。

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库
    |- 用户组一
        |- 用户01
            |- 人脸
        |- 用户02
            |- 人脸
            |- 人脸
            ....
         ....
    |- 用户组二
    |- 用户组三
    |- 用户组四
    ....
```

说明：关于人脸库的设置限制

- 每个开发者账号只能创建一个人脸库；
- 每个人脸库下，用户组（group）数量没有限制；
- 每个用户组（group）下，可添加最多300000张人脸，如每个uid注册一张人脸，则最多300000个用户uid；
- 每个用户（uid）所能注册的最大人脸数量没有限制；

> 说明：人脸注册完毕后，生效时间最长为35s，之后便可以进行识别或认证操作。

> 说明：注册的人脸，建议为用户正面人脸。

> 说明：uid在库中已经存在时，对此uid重复注册时，新注册的图片默认会追加到该uid下，如果手动选择`action_type:replace`，则会用新图替换库中该uid下所有图片。

请求说明 举例，要注册一个新用户，用户id为uid1，加入组id为group1, 注册成功后服务端会返回操作的logid：

```
\$aipFace->addUser(
  'uid1',
  'user info',
  'group1',
  file_get_content('user.jpg')
);
```

人脸注册请求参数要求：

所有图片经base64编码后的图片数据总和不超过10M。

人脸注册返回数据参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制48
user_info	**是**	string	新的user_info信息
action_type	否	string	如果为replace时，则uid不存在时，不报错，会自动注册。 不存在该参数时，如果uid不存在会提示错误

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 注册成功
{
    "log_id": 73473737,
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

5.3.6 人脸更新

用于对人脸库中指定用户，更新其下的人脸图像。

> 说明：针对一个uid执行更新操作，新上传的人脸图像将覆盖此uid原有所有图像。

> 说明：执行更新操作，如果该uid不存在时，会返回错误。如果添加了action_type:replace,则不会报错，并自动注册该uid，操作结果等同注册新用户。

请求说明 举例，要更新一个新用户，用户id为uid1， 更新成功后服务端会返回操作的logid：

```
\$aipFace->updateUser(
  'uid1',
  'user_info',
  'group1',
  file_get_content('new_user.jpg')
);
```

人脸更新请求参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制48
user_info	**是**	string	新的user_info信息

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 更新成功
```

```
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

5.3.7 人脸删除

人脸删除接口提供了从库中彻底删除一个用户的功能，包括用户所有图像和身份信息，同时也将从各个组中把用户删除。

举例，要删除一个用户id为uid1， 删除成功后服务端会返回操作的logid：

```
\$aipFace->deleteUser('uid1');
```

人脸删除请求参数要求：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	是	string	删除指定group_id中的uid信息

返回说明 人脸删除返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回样例：

```
// 删除成功
{
    "log_id": 73473737,
}
```

```
// 删除发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

5.3.8 用户信息查询

接口描述 用于查询人脸库中某用户的详细信息。

请求说明 举例，要查询指定用户的信息：

```
//查询所有分组下的
\$aipFace->getUser('uid1');

//查询指定分组下的
\$aipFace->getUser('uid1' array(
      'group_id' => 'group1',
));
```

用户信息查询请求参数要求：

以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	否	string	选择指定group_id则只查找group列表下的uid内容，如果不指定则查找所有group下对应uid的信息

用户信息查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

字段	是否必选	类型	说明
result	是	array	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+groups	是	array	用户所属组列表

返回样例：

```
{
    "result": {
        "uid": "testuser2",
        "user_info": "registed user info ...",
        "groups": [
            "grp1",
            "grp2",
            "grp3"
        ]
    },
    "log_id": 2979357502
}
```

5.3.9 组列表查询

接口描述 用于查询用户组的列表。

举例：

```
## 该参数如果不填则使用默认值
\$options = array(
        'start' => 0,
        'num' => 100,
);

\$aipFace->getGroupList(\$options);
```

组列表查询请求参数详情：

参数	是否必选	类型	说明
start	否	number	默认值0，起始序号
num	否	number	返回数量，默认值100，最大值1000

组列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	group_id列表

返回样例：

```
{
    "result_num": 2,
    "result": [
        "group1",
        "group2"
    ],
    "log_id": 3314921889
}
```

5.3.10 组内用户列表查询

接口描述 用于查询指定用户组中的用户列表。

举例：

```
## 该参数如果不填则使用默认值
\$options = array(
  ’start’ => 0,
  ’num’ => 100,
);

\$aipFace->getGroupUsers(’group1’, \$options);
```

组内用户列表查询请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	number	默认值0，起始序号
num	否	number	返回数量，默认值100，最大值1000

组内用户列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回样例：

```
{
    "log_id": 3314921889,
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

5.3.11 组间复制用户

接口描述 用于将已经存在于人脸库中的用户添加到一个新的组。

> 说明：并不是向一个指定组内添加用户，而是直接从其它组复制用户信息

请求说明 举例：

```
\$aipFace->addGroupUser('src_group','dst_group','uid1');
```

组间复制用户请求参数详情：

参数	是否必选	类型	说明
src_group_id	是	string	从指定group里复制信息
group_id	是	string	需要添加信息的组id列表
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
  "error_msg": "already add"
}
```

5.3.12 组内删除用户

接口描述 用于将用户从某个组中删除，但不会删除用户在其它组的信息。

说明：当用户仅属于单个分组时，本接口将返回错误，请使用人脸删除接口

请求说明 举例：

```
\$aipFace->deleteGroupUser('group1','uid1');
```

组内删除用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id

参数	是否必选	类型	说明
uid	是	string	用户id

组内删除用户接口返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
  "error_msg": "user must be in one group at least"
}
```

5.4 错误信息

若请求错误，服务器将返回的JSON文本包含以下参数：

5.4.1 错误返回格式

- error_code：错误码；关于错误码的详细信息请参考通用错误码和业务相关错误码。
- error_msg：错误描述信息，帮助理解和解决发生的错误。

5.4.2 错误码

SDK本地检测参数返回的错误码：

error_code	error_msg	备注
SDK100	image size error	图片大小超限
SDK101	image length error	图片边长不符合要求
SDK102	read image file error	读取图片文件错误
SDK103	user_info size error	user_info参数大小错误
SDK104	group_id format error	group_id有非法字符
SDK105	group_id size error	group_id参数大小错误
SDK106	uid format error	uid有非法字符
SDK107	uid size error	uid参数大小错误
SDK108	connection or read data time out	连接超时或读取数据超时
SDK109	unsupported image format	不支持的图片格式

服务端返回的错误码

错误码	错误信息	描述
4	Open api request limit reached	集群超限额
17	Open api daily request limit reached	每天流量超限额
18	Open api qps request limit reached	QPS超限额
19	Open api total request limit reached	请求总量超限额
100	Invalid parameter	无效的access_token参数
110	Access token invalid or no longer valid	Access Token失效
111	Access token expired	Access token过期
216100	invalid param	参数异常，具体异常原因详见error_msg
216101	not enough param	缺少必须的参数，具体异常原因详见error_msg
216102	service not support	请求了不支持的服务，请检查调用的url
216103	param too long	请求超长，一般为一次传入图片个数超过系统限制

错误码	错误信息	描述
216110	appid not exist	appid不存在，请重新检查后台应用列表中的应用信息
216111	invalid userid	userid信息非法，请检查对应的参数
216200	empty image	图片为空或者base64解码错误
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	数据库异常，**少量发生时重试即可**
216400	backend error	后端识别服务异常，可以根据具体msg查看错误原因
216402	face not found	未找到人脸，请检查图片是否含有人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在，请确认该用户是否注册或注册已经生效(**需要已经注册超过35s**)
216613	fail to delete user record	删除用户图片记录失败，重试即可
216614	not enough images	两两比对中图片数少于2张，无法比较
216615	fail to process images	服务处理该图片失败，发生后重试即可
216616	image existed	图片已存在
216617	fail to add user	新增用户图片失败
216618	no user in group	组内用户为空，确认该group是否存在或已经生效**(需要已经注册超过35s)**
216631	request add user overlimit	本次请求添加的用户数量超限

第6章

Python SDK文档

6.1 简介

Hi，您好，欢迎使用百度人脸识别API服务。

本文档主要针对API开发者，描述百度人脸识别接口服务的相关技术内容。如果您对文档内容有任何疑问，可以通过以下几种方式联系我们：

- 在百度云控制台内提交工单，咨询问题类型请选择人工智能服务；
- 加入开发者QQ群：224994340；

6.1.1 接口能力

接口名称	接口能力简要描述
人脸检测	检测人脸并定位，返回五官关键点，及人脸各属性值
人脸比对	返回两两比对的人脸相似值
人脸识别	在人脸库中查找相似的人脸
人脸认证	识别上传的图片是否为指定用户
人脸库设置	对人脸库的相关操作，如注册、删除、更新、查找用户信息等

6.2 快速入门

6.2.1 安装Python SDK

** Python SDK目录结构**

```
├── README.md
├── aip                  //SDK目录
│   ├── __init__.py      //导出类
│   ├── antiporn.py      //黄反识别
│   ├── base.py          //aip基类
│   ├── face.py          //人脸识别
│   ├── http.py          //http请求
│   ├── nlp.py           //nlp自然语言处理
│   └── ocr.py           //OCR图像识别
├── doc                  //使用文档
│   ├── antiporn.md
│   ├── face.md
│   ├── nlp.md
│   └── ocr.md
└── setup.py             //setuptools安装
```

支持 Python版本：2.7.+ ,3.+

安装使用SDK有如下方式：

- 如果已安装pip，执行`pip install baidu-aip`即可。
- 如果已安装setuptools，执行`python setup.py install`即可。

6.2.2 初始化一个AipFace对象

AipFace类是与人脸检测交互的客户端，为使用人脸检测的开发人员提供了一系列的交互方法。

用户可以参考如下代码新建一个AipFace对象：

```
# 引入人脸识别 SDK
from aip import AipFace

# 定义常量
APP_ID = '你的 App ID'
API_KEY = '你的 API Key'
```

```
SECRET_KEY = '你的 Secret Key'

# 初始化AipFace对象
aipFace = AipFace(APP_ID, API_KEY, SECRET_KEY)
```

在上面代码中，常量APP_ID在百度云控制台中创建，常量API_KEY与SECRET_KEY是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在AI服务控制台中的应用列表中查看。

注意：如您以前是百度云的老用户，其中API_KEY对应百度云的“Access Key ID”，SECRET_KEY对应百度云的“Access Key Secret”。

6.2.3 配置AipFace

如果用户需要配置AipFace的一些细节参数(一般不需要配置)，可以在构造AipFace之后调用接口设置参数，目前只支持以下参数：

接口	说明
setConnectionTi meoutInMillis	建立连接的超时时间（单位：毫秒
setSocketTimeou tInMillis	通过打开的连接传输数据的超时时间（单位：毫秒）

6.3 接口调用

6.3.1 人脸检测

接口描述 检测请求图片中的人脸，返回人脸位置、72个关键点坐标、及人脸相关属性信息。

检测响应速度，与图片中人脸数量相关，人脸数量较多时响应时间会有些许延长。

典型应用场景：如人脸属性分析，基于人脸关键点的加工分析，人脸营销活动等。

> 五官位置会标记具体坐标；72个关键点坐标也包含具体坐标，但不包含对应位置的详细位置描述。

请求说明 图片接受类型支持本地图片路径字符串，图片文件二进制数组。

举例，要对一张图片进行人脸识别，具体的人脸信息在返回的result字段中：

```
# 引入人脸识别 SDK
```

```
from aip import AipFace

# 定义常量
APP_ID = '你的 App ID'
API_KEY = '你的 API Key'
SECRET_KEY = '你的 Secret Key'

# 初始化AipFace对象
aipFace = AipFace(APP_ID, API_KEY, SECRET_KEY)

# 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

# 调用人脸属性检测接口
result = aipFace.detect(get_file_content('face.jpg'))
```

传入图片时还想增加一些自定义参数配置：

```
# 引入人脸识别 SDK
from aip import AipFace

# 定义常量
APP_ID = '你的 App ID'
API_KEY = '你的 API Key'
SECRET_KEY = '你的 Secret Key'

# 初始化AipFace对象
aipFace = AipFace(APP_ID, API_KEY, SECRET_KEY)

# 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

# 定义参数变量
options = {
    'max_face_num': 1,
    'face_fields': "age,beauty,expression,faceshape",
}

# 调用人脸属性识别接口
result = aipFace.detect(get_file_content('face.jpg'), options)
```

人脸检测 请求参数详情

参数	类型	描述	是否必须
face_fields	string	包括age、beauty、expression、face-shape、gender、glasses、land-mark、race、qualities 信息，逗号分隔，默认只返回人脸框、概率和旋转角度。	否
max_face_num	number	最多处理人脸数目，默认值1	是
image	string	图像数据	是

参数	类型	是否一定输出	描述
log_id	number	是	日志id
result_num	number	是	人脸数目
result	array	是	人脸属性对象的集合
+age	number	否	年龄。face_fields包含age时返回
+beauty	number	否	美丑打分，范围0-100，越大表示越美。face_fields包含beauty时返回
+location	array	是	人脸在图片中的位置
++left	number	是	人脸区域离左边界的距离
++top	number	是	人脸区域离上边界的距离
++width	number	是	人脸区域的宽度
++height	number	是	人脸区域的高度
+face_probability	number	是	人脸置信度，范围0-1

参数	类型	是否一定输出	描述
+rotation_angle	number	是	人脸框相对于竖直方向的顺时针旋转角，[-180,180]
+yaw	number	是	三维旋转之左右旋转角 [-90(左), 90(右)]
+pitch	number	是	三维旋转之俯仰角度[-90(上), 90(下)]
+roll	number	是	平面内旋转角 [-180(逆时针), 180(顺时针)]
+expression	number	否	表情，0，不笑；1，微笑；2，大笑。face_fields包含expression时返回
+expression_probability	number	否	表情置信度，范围0~1。face_fields包含expression时返回
+faceshape	array	否	脸型置信度。face_fields包含faceshape时返回
++type	string	是	脸型：square/ triangle/ oval/ heart/ round
++probability	number	是	置信度：0~1
+gender	string	否	male、female。face_fields包含gender时返回
+gender_probability	number	否	性别置信度，范围0~1。face_fields包含gender时返回
+glasses	number	否	是否带眼镜，0-无眼镜，1-普通眼镜，2-墨镜。face_fields包含glasses时返回

参数	类型	是否一定输出	描述
+glasses_probability	number	否	眼镜置信度，范围0~1。face_fields包含glasses时返回
+landmark	array	否	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+landmark72	array	否	72个特征点位置，示例图。face_fields包含landmark时返回
++x	number	否	x坐标
++y	number	否	y坐标
+race	string	否	yellow、white、black、arabs。face_fields包含race时返回
+race_probability	number	否	人种置信度，范围0~1。face_fields包含race时返回
+qualities	array	否	人脸质量信息。face_fields包含qualities时返回
++occlusion	array	是	人脸各部分遮挡的概率，[0, 1]（待上线）
+++left_eye	number	是	左眼
+++right_eye	number	是	右眼
+++nose	number	是	鼻子
+++mouth	number	是	嘴
+++left_cheek	number	是	左脸颊
+++right_cheek	number	是	右脸颊
+++chin	number	是	下巴

参数	类型	是否一定输出	描述
++type	array	是	真实人脸/卡通人脸置信度
+++human	number	是	真实人脸置信度，[0, 1]
+++cartoon	number	是	卡通人脸置信度，[0, 1]

返回说明

6.3.2 人脸两两比对

接口描述 该请求用于比对多张图片中的人脸相似度并返回两两比对的得分，可用于判断两张脸是否是同一人的可能性大小。

典型应用场景：如人证合一验证，用户认证等，可与您现有的人脸库进行比对验证。

> 说明：支持对比对的两张图片做在线活体检测

请求说明 举例，要对一组人脸图片进行两两比对，示例代码如下：

```
# 引入人脸识别 SDK
from aip import AipFace

# 定义常量
APP_ID = '你的 App ID'
API_KEY = '你的 API Key'
SECRET_KEY = '你的 Secret Key'

# 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

# 初始化AipFace对象
aipFace = AipFace(APP_ID, API_KEY, SECRET_KEY)

# 调用人脸两两比对接口
result = aipFace.match([
    get_file_content('face1.jpg'),
    get_file_content('face2.jpg'),
```

```
        get_file_content('face3.jpg'),
        get_file_content('face4.jpg'),
    ])
```

人脸比对请求参数：

所有图片经base64编码后的图片数据总和不超过10M。以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
ext_fields	否	string	返回质量信息，取值固定：目前支持qualities(质量检测)。(对所有图片都会做改处理)
image_liveness	否	string	返回的活体信息，“faceliveness,faceliveness”表示对比对的两张图片都做活体检测；“,faceliveness”表示对第一张图片不做活体检测、第二张图做活体检测；“faceliveness,”表示对第一张图片做活体检测、第二张图不做活体检测

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
result	是	array	结果数据，index和请求图片index对应。数组元素为每张图片的匹配得分数组，top n。得分[0,100.0]

字段	是否必选	类型	说明
+index_i	是	number	比对图片1的index
+index_j	是	number	比对图片2的index
+score	是	double	比对得分
ext_info	否	array	对应参数中的ext_fields
+qualities	否	string	质量相关的信息，无特殊需求可以不使用
+faceliveness	否	string	活体分数“0,0.9999”（表示第一个图不做活体检测、第二个图片活体分数为 0.9999）。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
//请求为四张图片，第三张解析失败
{
    "log_id": 73473737,
    "result_num":3,
    "result": [
        {
            "index_i": 0,
            "index_j": 1,
            "score": 44.3
        },
        {
            "index_i": 0,
            "index_j": 3,
            "score": 89.2
        },
        {
            "index_i": 1,
            "index_j": 3,
            "score": 10.4
```

```
        }
        ……
    ]
}
```

6.3.3 人脸识别

接口描述 用于计算指定组内用户，与上传图像中人脸的相似度。识别前提为您已经创建了一个人脸库。

典型应用场景：如人脸闸机，考勤签到，安防监控等。

> 说明：人脸识别返回值不直接判断是否是同一人，只返回用户信息及相似度分值。

> 说明：推荐可判断为同一人的相似度分值为80，您也可以根据业务需求选择更合适的阈值。

请求说明 举例，要计算一张图片与指定组group1内用户相似度：

```
## 该参数如果不填则使用默认值
options = {
  'user_top_num': 0,
    'face_top_num': 1,
}


## 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

aipFace.identifyUser(
              'group1',
              get_file_content('user.jpg'),
              options
            )
```

人脸识别请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id(由数字、字母、下划线组成)列表，每个groupid长度限制48
image	是	string	图像数据
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)
user_top_num	否	number	返回用户top数，默认为1，最多返回5个

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体(测试期间
result	是	array	结果数组
+group_id	是	string	对应的这个用户的group_id
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+scores	是	array	结果数组，数组元素为匹配得分，top n。得分[0,100.0]

返回说明 返回样例：

```
{
    "log_id": 73473737,
    "result_num":1,
    "result": [
        {
            "group_id" : "group1",
            "uid": "123",
            "user_info": "user info",
            "scores": [
                    99.3,
                    83.4
            ]
        }
    ]
}
```

6.3.4 人脸认证

接口描述 用于识别上传的图片是否为指定用户，即查找前需要先确定要查找的用户在人脸库中的id。

典型应用场景：如人脸登录，人脸签到等

> 说明：人脸认证与人脸识别的差别在于：人脸识别需要指定一个待查找的人脸库中的组；而人脸认证需要指定具体的用户id即可，不需要指定具体的人脸库中的组；实际应用中，人脸认证需要用户或系统先输入id，这增加了验证安全度，但也增加了复杂度，具体使用哪个接口需要视您的业务场景判断。

> 说明：请求参数中，新增在线活体检测

请求说明 举例，要认证一张图片在指定group中是否为uid1的用户：

```
## 该参数如果不填则使用默认值
options = {
  'top_num': 1,
}
```

```
## 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()
```

```
aipFace.verifyUser(
              'uid1',
              'group1',
              get_file_content('user.jpg'),
              options
            )
```

人脸认证请求参数详情：

可选参数均放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图像数据
group_id	是	string	用户组id（由数字、字母、下划线组成）列表，每个groupid长度限制48
top_num	否	number	返回匹配得分top数，默认为1
ext_fields	否	string	特殊返回信息，多个用逗号分隔，取值固定：目前支持faceliveness(活体检测)

字段	是否必选	类型	说明
log_id	是	number	请求唯一标识码，随机数
result_num	是	number	返回结果数目，即：result数组中元素个数

字段	是否必选	类型	说明
result	是	array	结果数组，数组元素为匹配得分，top n。得分范围[0,100.0]。推荐得分超过80可认为认证成功
ext_info	否	array	对应参数中的ext_fields
+faceliveness	否	string	活体分数，如0.49999。活体检测参考分数0.4494，以上则可认为是活体（测试期间）

返回说明 返回样例：

```
{
  "results": [
    93.86580657959,
    92.237548828125
  ],
  "result_num": 2,
  "log_id": 1629483134
}
```

6.3.5 人脸注册

接口描述 用于从人脸库中新增用户，可以设定多个用户所在组，及组内用户的人脸图片，

典型应用场景：构建您的人脸库，如会员人脸注册，已有用户补全人脸信息等。

人脸库、用户组、用户、用户下的人脸层级关系如下所示：

```
|- 人脸库
   |- 用户组一
      |- 用户01
         |- 人脸
      |- 用户02
         |- 人脸
         |- 人脸
```

```
        ....
      ....
|- 用户组二
|- 用户组三
|- 用户组四
....
```

说明：关于人脸库的设置限制

- 每个开发者账号只能创建一个人脸库；
- 每个人脸库下，用户组（group）数量没有限制；
- 每个用户组（group）下，可添加最多300000张人脸，如每个uid注册一张人脸，则最多300000个用户uid；
- 每个用户（uid）所能注册的最大人脸数量没有限制；

> 说明：人脸注册完毕后，生效时间最长为35s，之后便可以进行识别或认证操作。

> 说明：注册的人脸，建议为用户正面人脸。

> 说明：uid在库中已经存在时，对此uid重复注册时，新注册的图片默认会追加到该uid下，如果手动选择`action_type:replace`，则会用新图替换库中该uid下所有图片。

请求说明 举例，要注册一个新用户，用户id为uid1，加入组id为group1, 注册成功后服务端会返回操作的logid：

```
## 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

aipFace.addUser(
            'uid1',
            'user info',
            'group1',
            get_file_content('user.jpg')
            )
```

人脸注册请求参数要求：

所有图片经base64编码后的图片数据总和不超过10M。

人脸注册返回数据参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制48
user_info	**是**	string	新的user_info信息
action_type	否	string	如果为replace时，则uid不存在时，不报错，会自动注册。 不存在该参数时，如果uid不存在会提示错误

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 注册成功
{
    "log_id": 73473737,
}
// 注册发生错误
{
  "error_code": 216616,
  "log_id": 674786177,
  "error_msg": "image exist"
}
```

6.3.6 人脸更新

接口描述 用于对人脸库中指定用户，更新其下的人脸图像。

说明：针对一个uid执行更新操作，新上传的人脸图像将覆盖此uid原有所有图像。

说明：执行更新操作，如果该uid不存在时，会返回错误。如果添加了action_type:replace,则不会报错，并自动注册该uid，操作结果等同注册新用户。

请求说明 举例，要更新一个新用户，用户id为uid1， 更新成功后服务端会返回操作的logid：

```
## 读取图片
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

aipFace.updateUser(
  'uid1',
  'user info',
  'group1',
  get_file_content('newuser.jpg')
)
```

人脸更新请求参数详情：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
image	是	string	图片数据
group_id	是	string	用户组id（由数字、字母、下划线组成），长度限制48
user_info	**是**	string	新的user_info信息

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 更新成功
{
    "log_id": 73473737,
}
// 更新发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

6.3.7 人脸删除

接口描述 人脸删除接口提供了从库中彻底删除一个用户的功能，包括用户所有图像和身份信息，同时也将从各个组中把用户删除。

请求说明 举例，要删除一个用户id为uid1， 删除成功后服务端会返回操作的logid：

```
aipFace.deleteUser('uid1')
```

人脸删除请求参数要求：

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	是	string	删除指定group_id中的uid信息

返回说明 人脸删除返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回样例：

```
// 删除成功
{
    "log_id": 73473737,
}
// 删除发生错误
{
  "error_code": 216612,
  "log_id": 1137508902,
  "error_msg": "user not exist"
}
```

6.3.8 用户信息查询

接口描述 用于查询人脸库中某用户的详细信息。

请求说明 举例，要查询指定用户的信息：

```
## 查询所有分组下的
aipFace.getUser('uid1')

## 查询指定分组下的
aipFace.getUser('uid1', 'group1')
```

用户信息查询请求参数要求：

以下可选参数放在接口最后的options参数中。

参数	是否必选	类型	说明
uid	是	string	用户id（由数字、字母、下划线组成），长度限制128B
group_id	否	string	选择指定group_id则只查找group列表下的uid内容，如果不指定则查找所有group下对应uid的信息

用户信息查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result	是	array	结果数组
+uid	是	string	匹配到的用户id
+user_info	是	string	注册时的用户信息
+groups	是	array	用户所属组列表

返回样例：

```
{
    "result": {
        "uid": "testuser2",
        "user_info": "registed user info ...",
        "groups": [
            "grp1",
            "grp2",
            "grp3"
        ]
    },
    "log_id": 2979357502
}
```

6.3.9 组列表查询

接口描述 用于查询用户组的列表。

举例：

```
## 该参数如果不填则使用默认值
options = {
    'start': 0,
    'num': 100,
}

aipFace.getGroupList(options)
```

组列表查询请求参数详情：

参数	是否必选	类型	说明
start	否	number	默认值0，起始序号
num	否	number	返回数量，默认值100，最大值1000

组列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	group_id列表

返回样例：

```
{
    "result_num": 2,
    "result": [
        "group1",
        "group2"
    ],
    "log_id": 3314921889
}
```

6.3.10 组内用户列表查询

接口描述 用于查询指定用户组中的用户列表。

举例：

```
## 该参数如果不填则使用默认值
options = {
  'start': 1,
    'num': 100,
}

aipFace.getGroupUsers('group1', options)
```

组内用户列表查询请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
start	否	number	默认值0，起始序号
num	否	number	返回数量，默认值100，最大值1000

组内用户列表查询返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一
result_num	是	number	返回个数
result	是	array	user列表
+uid	是	string	用户id
+user_info	是	string	用户信息

返回样例：

```
{
    "log_id": 3314921889,
    "result_num": 2,
    "result": [
        {
            "uid": "uid1",
            "user_info": "user info 1"
        },
        {
            "uid": "uid2",
            "user_info": "user info 2"
        }
    ]
}
```

6.3.11 组间复制用户

接口描述 用于将已经存在于人脸库中的用户添加到一个新的组。

说明：并不是向一个指定组内添加用户，而是直接从其它组复制用户信息

请求说明 举例：

```
aipFace.addGroupUser('src_group', 'dst_group', 'uid1')
```

组间复制用户请求参数详情：

参数	是否必选	类型	说明
src_group_id	是	string	从指定group里复制信息
group_id	是	string	需要添加信息的组id列表
uid	是	string	用户id

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回说明 返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216100,
  "log_id": 3111284097,
  "error_msg": "already add"
}
```

6.3.12 组内删除用户

接口描述 用于将用户从某个组中删除，但不会删除用户在其它组的信息。

说明：当用户仅属于单个分组时，本接口将返回错误，请使用人脸删除接口

请求说明 举例：

```
aipFace.deleteGroupUser('group1', 'uid1')
```

组内删除用户请求参数详情：

参数	是否必选	类型	说明
group_id	是	string	用户组id
uid	是	string	用户id

组内删除用户接口返回数据参数详情：

字段	是否必选	类型	说明
log_id	是	number	请求标识码，随机数，唯一

返回样例：

```
// 正确返回值
{
    "log_id": 3314921889,
}
// 发生错误时返回值
{
  "error_code": 216619,
  "log_id": 815967402,
  "error_msg": "user must be in one group at least"
}
```

6.4 错误信息

6.4.1 错误返回格式

若请求错误，服务器将返回的JSON文本包含以下参数：

- error_code：错误码；关于错误码的详细信息请参考通用错误码和业务相关错误码。
- error_msg：错误描述信息，帮助理解和解决发生的错误。

6.4.2 错误码

SDK本地检测参数返回的错误码：

error_code	error_msg	备注
SDK100	image size error	图片大小超限
SDK101	image length error	图片边长不符合要求
SDK102	read image file error	读取图片文件错误
SDK103	user_info size error	user_info参数大小错误
SDK104	group_id format error	group_id有非法字符
SDK105	group_id size error	group_id参数大小错误
SDK106	uid format error	uid有非法字符
SDK107	uid size error	uid参数大小错误
SDK108	connection or read data time out	连接超时或读取数据超时
SDK109	unsupported image format	不支持的图片格式

服务端返回的错误码

错误码	错误信息	描述
4	Open api request limit reached	集群超限额
17	Open api daily request limit reached	每天流量超限额
18	Open api qps request limit reached	QPS超限额
19	Open api total request limit reached	请求总量超限额
100	Invalid parameter	无效的access_token参数
110	Access token invalid or no longer valid	Access Token失效
111	Access token expired	Access token过期

错误码	错误信息	描述
216100	invalid param	参数异常，具体异常原因详见error_msg
216101	not enough param	缺少必须的参数，具体异常原因详见error_msg
216102	service not support	请求了不支持的服务，请检查调用的url
216103	param too long	请求超长，一般为一次传入图片个数超过系统限制
216110	appid not exist	appid不存在，请重新检查后台应用列表中的应用信息
216111	invalid userid	userid信息非法，请检查对应的参数
216200	empty image	图片为空或者base64解码错误
216201	image format error	图片格式错误
216202	image size error	图片大小错误
216300	db error	数据库异常，**少量发生时重试即可**
216400	backend error	后端识别服务异常，可以根据具体msg查看错误原因
216402	face not found	未找到人脸，请检查图片是否含有人脸
216500	unknown error	未知错误
216611	user not exist	用户不存在，请确认该用户是否注册或注册已经生效(**需要已经注册超过35s**)
216613	fail to delete user record	删除用户图片记录失败，重试即可
216614	not enough images	两两比对中图片数少于2张，无法比较
216615	fail to process images	服务处理该图片失败，发生后重试即可
216616	image existed	图片已存在
216617	fail to add user	新增用户图片失败

错误码	错误信息	描述
216618	no user in group	组内用户为空，确认该group是否存在或已经生效**(需要已经注册超过35s)**
216631	request add user overlimit	本次请求添加的用户数量超限

第7章

常见问题

Q：识别的图片支持怎样输入？
A：目前文字识别接口仅支持base64编码输入。

Q：什么是base64编码，如何提供？
A：图片的base64编码指将一副图片数据编码成一串字符串，使用该字符串代替图像地址。您可以首先得到图片的二进制，然后用Base64格式编码即可。注：图片的base64编码是不包含图片头的，如（）