<人脸核身解决集成说明文档—APP 方案>

V1.1 版本

版本	日期	说明
V1.1	2017.10.27	更换公安接口为 V2 版本；升级 SDK 为 3.1 版本
V1.0	2017.07.23	介绍基于 APP 形态的人脸核身集成方案

1. 简介

1.1 方案概述

传统的远程身份验证，存在以下问题：

- **审核效率：传统的身份认证由人工审核，成本高、效率慢、反馈用户周期长**
- **伪造风险：人像图片和身份证图片存在P图风险，人证很有可能「不对应」**
- **操作复杂：用户需要提交各类证件信息，邮寄待审核内容或线下网点办理**

人脸核身方案轻松改变以上问题，在线快速完成认证：

真人	+	本人
通过离在线双重活体检测，确保操作者为真人，可有效抵御彩照、视频、3D建模等攻击。用户无需提交任何资料，或去网点柜台办理业务，高效方便		基于「真人」基础，将真人人脸与公民身份信息库人脸图像对比，确保操作者身份真实性。避免身份证或人脸图像伪造等欺诈风险，权威可靠

核身流程简单概述为以下三步，根据第三步中的两张图片的人脸对比得分，进行最终业务的核身结果判断依据，阈值可根据业务需要自行调整。

1

证件识别

- 手动输入或OCR识别身份证信息
- 验证姓名身份证号是否属实
- **通过身份信息获取公安系统小图**

2

活体检测与人脸采集

真人

- 有动作交互或用户拍照完成人脸采集
- 完成活体检测验证
- 获取到质量合格的人脸图像

3

云端比对

VS 本人

- 公安小图与活体检测后的图片对比
- 核实用户身份是否为真实

1.2 功能介绍

a) **活体检测**：基于人脸进行有交互活体检测、及后台在线活体检测算法，判断用户为真人，保障业务环节中的用户真实性判断。

b) **人脸质量检测**：对客户端 SDK 人脸采集过程中，人脸的姿态角度、光照、遮挡、模糊度等进行条件校验，并输出条件符合识别要求的人脸图像帧。

c) **OCR 身份证识别**：身份证图像识别，自动识别身份证相关信息，具体如下所示：

- ✓ 支持对二代居民身份证的关键字段识别
- ✓ 正面：包括姓名、性别、民族、出生日期、住址、身份证号
- ✓ 反面：签发机关、有效期限
- ✓ 识别准确率可达 98%以上

d) **身份证质量检测**：OCR 客户端 SDK 的能力，在完成拍照前，会自动校验拍摄条件，确保拍摄待识别的身份证图片质量合格，具体校验项如下所示：

- ✓ 是否清晰度合适、是否反光，确保身份证图像质量
- ✓ 是否包含占比合适的身份证，确保避免过于倾斜、非身份证
- ✓ 身份证中的关键字段是否清晰、反光，确保字段内容清晰
- ✓ 身份证国徽、人像部分是否位置匹配，确保检测位置准确，正反面正确
- ✓ 连续输入算法的两帧是否差异过大，确保避免镜头或证件晃动
- ✓ 身份证是否占比过小，确保镜头远近合适

e) **身份证识别五分类**：分别指「正常身份证」「复印件」「临时身份证」「屏幕翻拍」「PS 修改」，具体类别示例下图所示：

正常身份证

复印件

临时身份证

对各类型的身份证自动甄别（正反面）
可根据业务需要进行筛选控制

- 正常身份证
- 复印件
- 临时身份证
- 屏幕翻拍
- PS（被修改过）

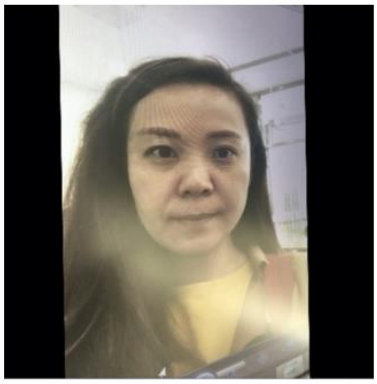
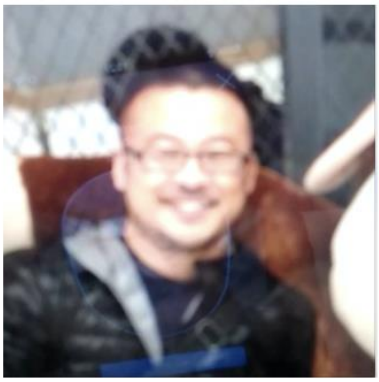
屏幕翻拍

PS

基于身份证五分类的检测，可以在业务操作最开始的身份证 OCR 录入环节，即甄别出不符合条件的身份证类型，提升业务检测的严谨性和安全性，同时减少不必要的后续检测步骤。

f) **公安身份验证**：接口形式，入参为姓名、身份证号、以及人脸活体检测得到的图像。基于姓名与身份证号，可调取公民身份证小图（源自公安系统），并同时将此小图与活体检测的图像进行比对，得到一个相似度分值，作为验证的依据。以上流程仅在一个接口中即可实现。由于公安系统小图，具有最权威的身份校验作用，故对本人的验证结果可信度也最为合理。

g) **有交互活体检测**：人脸客户端 SDK 本地离线功能，配合手机前置摄像头使用，SDK 指定用户随机做出随机动作交互，指用户配合做出眨眼、张嘴、点头、抬头、左摇头、右摇头动作，并同时检测用户的动作完成情况。在做动作的过程中，SDK 可以随机抓取急诊图像，并在动作通过后将抓取的图像，上传到后台进行活体判断，其中随机抓取的攻击图片举例如下：


h) **在线活体检测**：接口形式，配合有交互活体检测使用效果最佳。该算法主要通过检测照片中的屏幕边框、屏幕反光、摩尔纹、像素纹理、成像畸形等线索来区分二次翻拍的攻击图像与真人活体图像；

1.3 产品构成

本方案由两个 SDK，及两个接口组成，完成身份证识别、活体判断、身份校验操作，具体如下所示：

1.3.1 OCR 身份证识别 SDK

提供身份证质量检测（本地校验）；五分类判断（API 返回参数）；身份证内容识别能力（API 返回参数）

1.3.2 人脸识别客户端 SDK

人脸客户端 SDK 基于离线 license 授权，可离线使用，提供人脸检测、人脸捕获、有交互活体检测能力。具体能力如下所示：

✓ **人脸检测**：实时检测视频流中每帧图像中的人脸图像，此功能可完全离线使用，无需联网请求；

- ✓ **人脸跟踪**：对检测到的人脸持续检测跟踪，动态定位人脸轮廓及五官关键点，稳定贴合人脸各部位；
- ✓ **质量检测**：判断视频流中的图片帧中，哪些图片质量较佳，即指人脸图像特征清晰，满足角度、姿态、光照、模糊度等校验；
- ✓ **活体检测**：通过让用户做出指定脸部交互动作，识别当前操作者是否为活体，此功能可离线使用，可设定指定动作是否使用及应用顺序；可有效抵御高清图片、3D 建模、视频等攻击
- ✓ **人脸采集**：通过本地 SDK 能力，采集人脸图像，同时经过人脸质量检测，确保采集到的人脸图像符合各条件校验（光照、姿态、光照、模糊度等），为设备前端获取有效可分析人脸的主要功能

1.3.3 公安验证接口（参考文档：公安验证 v2.pdf）

https://aip.baidubce.com/rest/2.0/face/v2/person/verify （公安验证v2）

公安验证接口，自SDK申请通过后，默认开通权限，并附赠500次免费测试，如需开通计费，请参考：

http://ai.baidu.com/docs#/Face-Pricing/top 中的「解决方案」报价

公安验证，对接公安身份系统，请求入参主要为：姓名、身份证号、经过活体验证的人脸图像，及图片质量检测配置、活体检测配置，公安验证过程中，将会基于姓名和身份证号调取公安系统小图，将此小图与人脸图像对比，返回对比分值。

对于 H5 方案，我们同样提供在线活体检测接口，此接口通过检测照片中的屏幕边框、屏幕反光、摩尔纹、像素纹理、成像畸形等线索来区分二次翻拍的攻击图像与真人活体图像。返回参数为活体检测分值，基于业务需要，可以设定需要的阈值，从而判断是否活体检测通过。

核身的主要流程中，您主要使用公安验证接口即可，具体的使用方法，可以根据业务需要，选择是否进行图片质量检测、活体检测。

如果参选择这两个参数,则检测顺序将会变为: 图片质量检测 -> 活体检测 -> 公安身份验证，其中前两个环节有任意一个条件不通过，则检测流程流程终止，返回信息中会说明不符合的详细内容；如果前两个检测条件全部验证通过，则会执行公安验证接口请求，返回内容包含对比分数、活体检测分数、图片质量检测信息。

1.4 方案优势

该方案的优势在于：随机动作可以增加攻击的成本，并且大大降低打印照片等攻击通过的概率，而这些正好对于后端检测算法来说属于较难案例；另外，活体判断所用的多张图片为前端在完成动作过程随机抓取，因此加大了攻击暴露出破绽的可能性；同时，多张图片可以提高采集到高质量真人活体图片的概率，因而可以采用一个更高的采信阈值，而该阈值对应的活体分数攻击图片则很难达到。

2. 集成指南

2.1 准备工作

在正式集成前，需要做一些准备工作，完成一些账号、应用及配置，具体如下：

2.1.1 注册开发者

STEP1：点击百度 AI 开放平台导航右侧的控制台，页面跳转到百度云登录界面，登录完毕后，将会进入到百度云后台，点击「控制台」进入百度云控制台页面；您也可以在官网直接点击免费试用，登录完毕后将自动进入到百度云控制台。
STEP2：使用百度账号完成登录，如您还未持有百度账户，可以点击此处注册百度账户。
STEP3：进入百度云欢迎页面，填写企业/个人基本信息，注册完毕，至此成为开发者。注：(如您之前已经是百度云用户或百度开发者中心用户，STEP3 可略过。)
STEP4：进入百度云控制台，找到人工智能相关服务面板。
STEP5：**点击进入「人脸识别」模块。**

注一：如果您通过百度 AI 开放平台登录到后台，AI 相关服务模块入口，如下图所示：

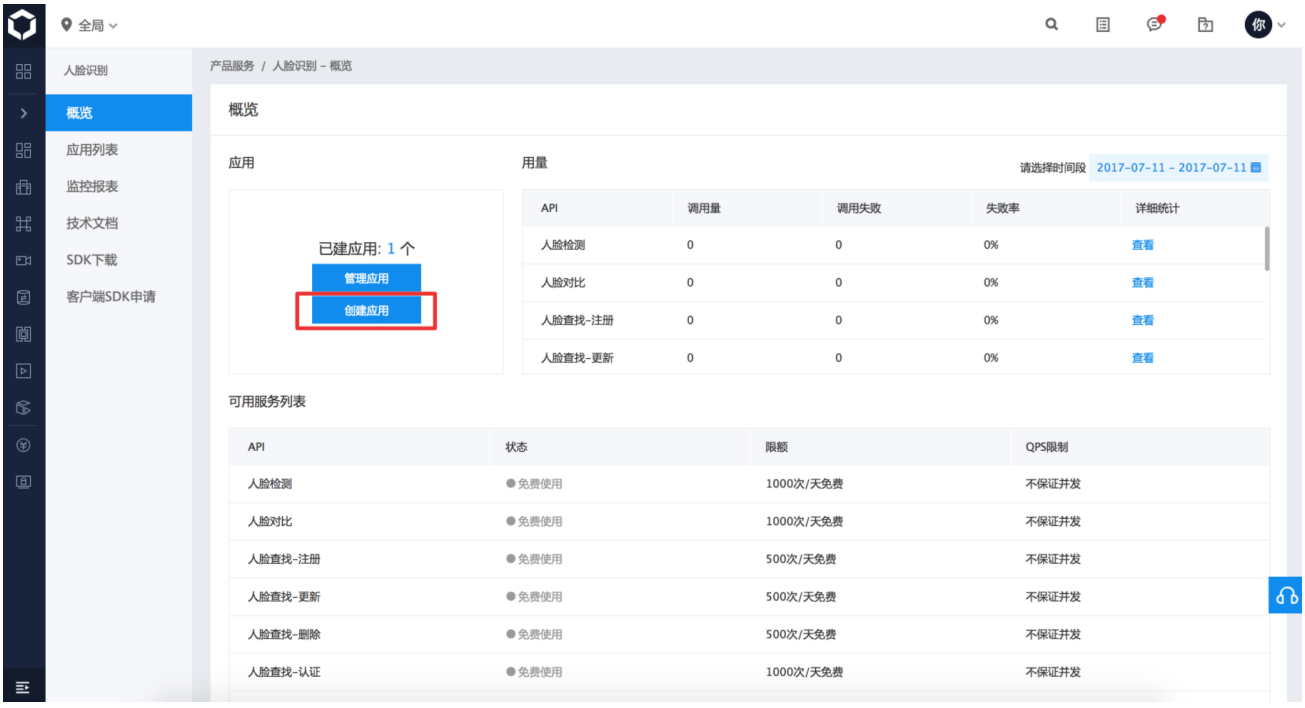
注二：如果您通过百度云直接登录后台，AI 相关服务模块入口，则如下图红框部分所示：

两种入口只是展现形式不同，相关 AI 服务模块内容完全一样。

2.1.2 创建应用

创建好账号后，在正式调用 AI 能力之前，需要您创建一下应用，这个是调用服务的基础能力单元

选择「人脸识别」服务，首先见到此服务的控制台概览页，如下图所示：



如上图所示，点击「创建应用」，即可进入应用创建界面，如下图所示：

注意：您需要创建两个应用，分别做以下作用：

✓ 应用一：勾选身份证识别能力，绑定 iOS OCR、Android OCR 包名；

✓ 应用二 :用于开通公安验证接口、在线活体检测接口能力(可发送工单申请 ,或者联系具体对接商务)；

2.1.3 获取密钥

	应用名称	AppID	API Key	Secret Key
1	身份证识别	9918631	GOS93OvBoudHeRegLaGpYXGr	******* 显示

在您创建完毕应用后，平台将会分配给您此应用的相关凭证，主要为 **AppID、API Key、Secret Key**，以上三个信息是您应用实际开发的主要凭证，每个应用之间各不相同，请您妥善保管。如上图所示。

注：开发中请注意区分多份 AKSK（API Key、Secret Key）

2.1.4 生成 token

刚才所创建的应用在调用开放平台 API 之前，首先需要获取 Access Token（用户身份验证和授权的凭证）您需要使用创建应用所分配到的 AppID、API Key 及 Secret Key，进行 Access Token 的生成，方法详见 Access Token 获取，我们为您准备了几种常见语言的请求示例代码。

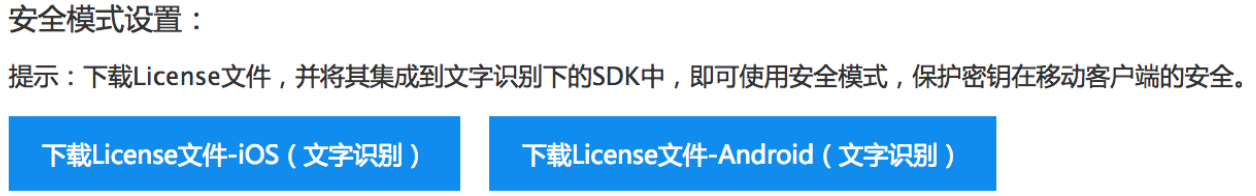
注：Access Token 的有效期为 30 天（以秒为单位），请您集成时注意在程序中定期请求新的 token，或每次请求都获取一次 token。

2.1.5 开通特殊接口权限

目前公安验证接口、在线活体接口，暂不对外开放（在应用管理中无法自行勾选），请将用于此两个接口调用的 appid，通过商务方式告知百度方面，百度方面配置完毕后，即可开启此 appid 的调用权限。

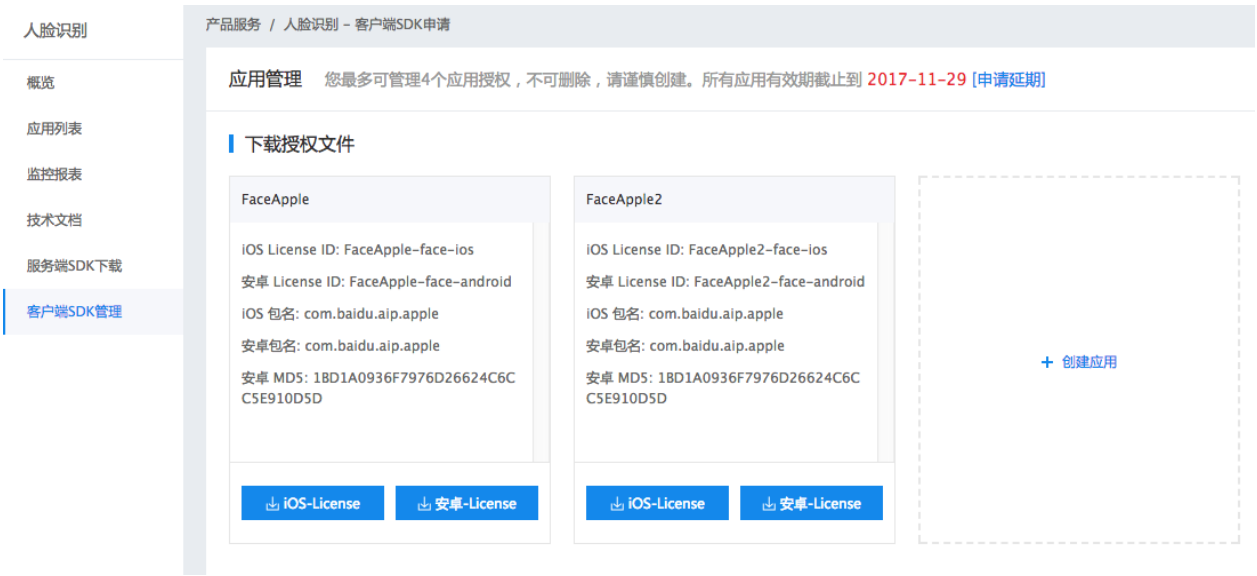
2.1.6 SDK License（重要！重要！重要！）

OCR 身份证识别 SDK License：此 License 封装了 aksk，用于更方便地调用接口。请在刚才创建的两个身份证应用中，点击「应用名称」，进入应用详情页面，找到如下图所示内容：

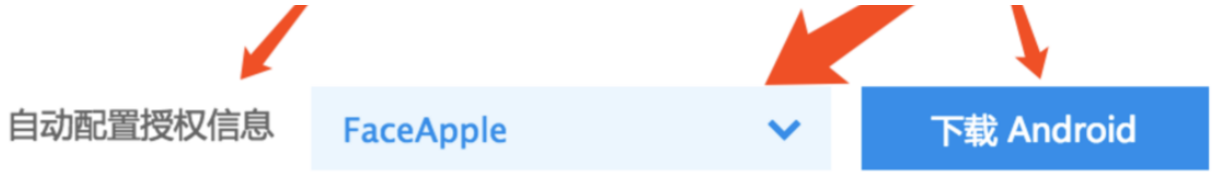


点击 button，下载此 license 文件，名称都为 aip.license（分别用于 iOS 和安卓），稍后集成需要使用。

人脸 SDK License：此 license 用于 SDK 离线功能使用，在您的申请人脸 SDK 的后台页面，下载两个端的 license，如下图所示：



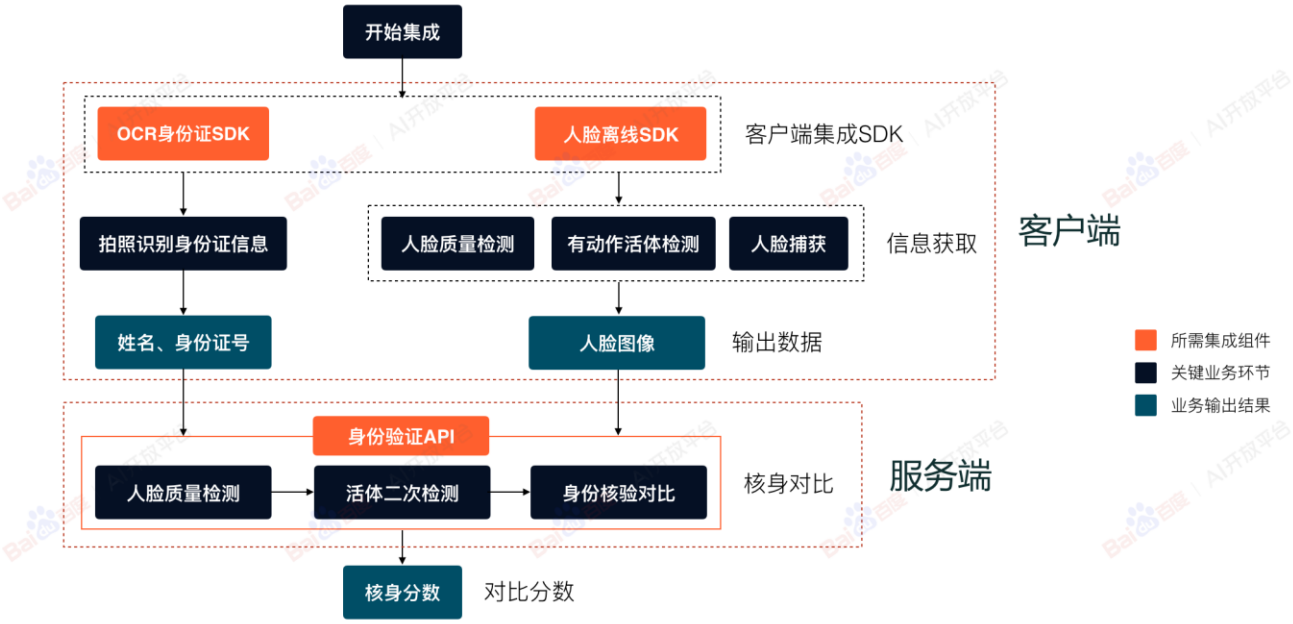
用于接下来集成使用，不过我们为您提供了自动化配置脚本，如下图所示：



您可以在下载核身示例工程的同时，直接下载已经配置好授权信息的版本。

2.2 集成逻辑

2.2.1 集成流程图



如上图所示，集成逻辑分为三步：

1. OCR SDK 获取身份证信息，应用本地输出；
2. 人脸 SDK 获取有交互活体检测图片，应用本地输出；
3. 将姓名、身份证号、人脸图像推送到您的 server 端，从 server 端先调用在线活体接口，人脸图像活体通过后，再继续调用公安验证接口，得到分值。

其中业务终止可有五个节点：

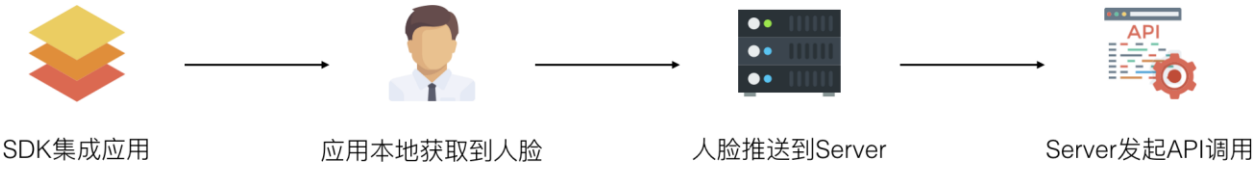
1. 身份证不合规：在身份证识别的返回参数中，有如下字段：

risk_type	否	string	输入参数 detect_risk = true 时，则返回该字段识别身份证类型: normal-正常身份证；copy-复印件；temporary-临时身份证；screen-翻拍；unknow-其他未知情况
edit_tool	否	string	如果参数 detect_risk = true 时，则返回此字段。如果检测身份证被编辑过，该字段指定编辑软件名称，如:Adobe Photoshop CC 2014 (Macintosh),如果没有被编辑过则返回值无此参数

如判断身份证信息为非正常身份证，可提示用户，并中断验证流程；

2. 人脸 SDK 本地有动作活体检测，如果用户无法完成指定动作，也无法继续进行；
3. 人脸图像质量较低（如选择了此项），流程中止；
4. 人脸图像经在线活体检测判断分数过低，流程中止；
5. 前三步都完成，但是公安验证分数过低，也可提示核身验证失败。

2.2.2 接口服务



SDK 获取人脸过程中的处理，完全无需联网，但人脸对比、人脸查找、人脸属性分析能力需要调用 API 使用。产品策略方面，因 API 使用需要使用在线鉴权 token，为 token 的安全起见，建议将人脸推送到 server 端再调用 API 接口。

对安全有进一步需求的话，为防止人脸传输过程中被篡改，可对 SDK 本地输出的人脸图像做加密处理，在 server 端进行相应解密操作，进一步增强安全性。

2.2.3 阈值选择

在线活体检测分值的阈值：

策略/阈值	99.9%活体通过率	99.5%活体通过率	99%活体通过率
以帧为单位场景（如 H5）	0.00735262	**0.834963**	0.984858
SDK 取 3 帧	0.0159658	**0.94834**	0.995001
SDK 取 8 帧	0.992966	**0.999709**	0.999932

注一：阈值设置越高，对应的安全性越高，即拒识率越高，但通过率相应下降

注二：SDK 单帧模式，超过阈值即为活体通过；

注三：SDK 输出多帧图像模式，可用选多帧分别请求后端接口，任何一帧通过阈值，即可判断为活体

注四：阈值高低，可根据业务要求判断，**官方推荐选取 99.5% 通过率对应的阈值**

公安验证接口分值的阈值选择：

阈值分数	误识率	识别率

60	0.781615%	99.550128%
70	0.096534%	98.307626%
78	**0.015570%（万分之一）**	**95.672664%**
80	**0.009342%（低于万分之一）**	**94.323051%**

注一：**推荐 80 分、或者 78 分为业务判断阈值。**

注二：识别率与误识率成正比：识别率越高，误识率越高；识别率降低，误识率也降低；

2.3 iOS 集成

【OCR 身份证识别集成】

把下载下来的 License（aip.license），添加到项目里面，无需更改文件名称，然后在 AppDelegate 添加以下代码引用进去。

```
NSString *licenseFile = [[NSBundle mainBundle] pathForResource:OCR_LICENSE_NAME ofType:OCR_LICENSE_SUFFIX];
NSData *licenseFileData = [NSData dataWithContentsOfFile:licenseFile];
[[AipOcrService shardService] authWithLicenseFileData:licenseFileData];
```

在 FaceParameterConfig.h 里面设置下载下来的 License 文件的名字和后缀。

```
// OCR license文件名
#define OCR_LICENSE_NAME     @"aip"

// OCR license后缀
#define OCR_LICENSE_SUFFIX   @"license"
```

通过 API 调用 ViewController 进行身份证扫描

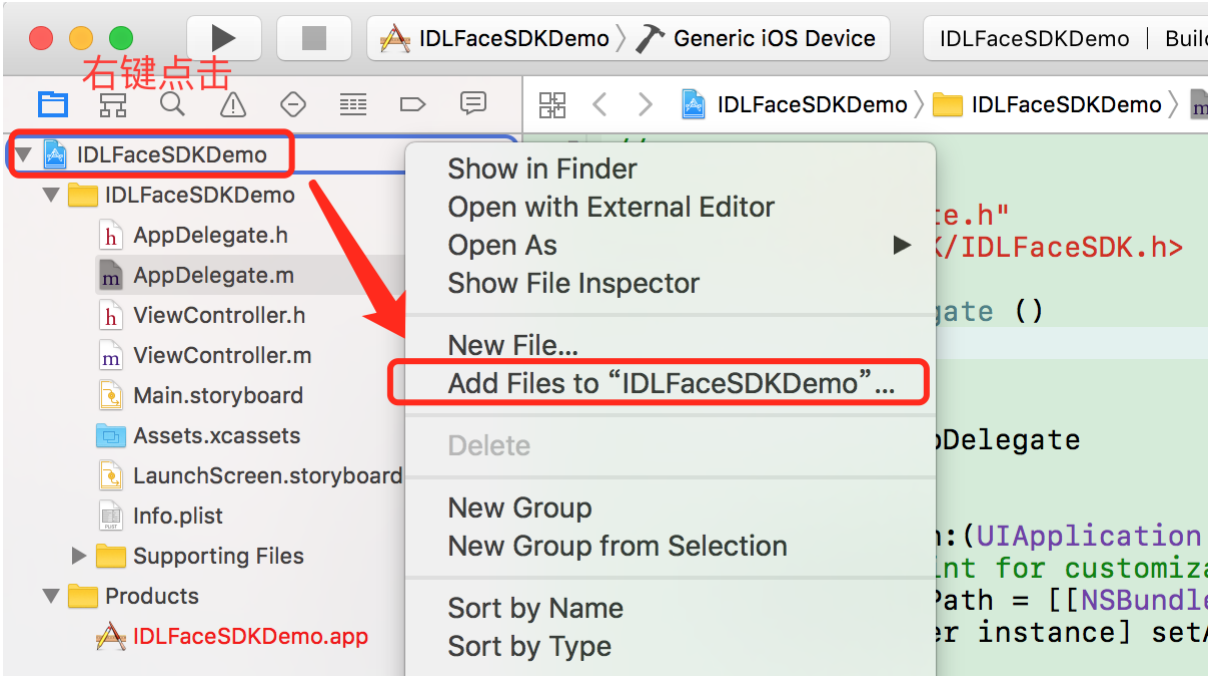
```
UIViewController * vc = [AipCaptureCardVC ViewControllerWithCardType:CardTypeLocalIdCardFont andDelegate:self];
[self presentViewController:vc animated:YES completion:nil];
```

详细调用文档，请参考此文档地址：http://ai.baidu.com/docs#/OCR-iOS-SDK

==========

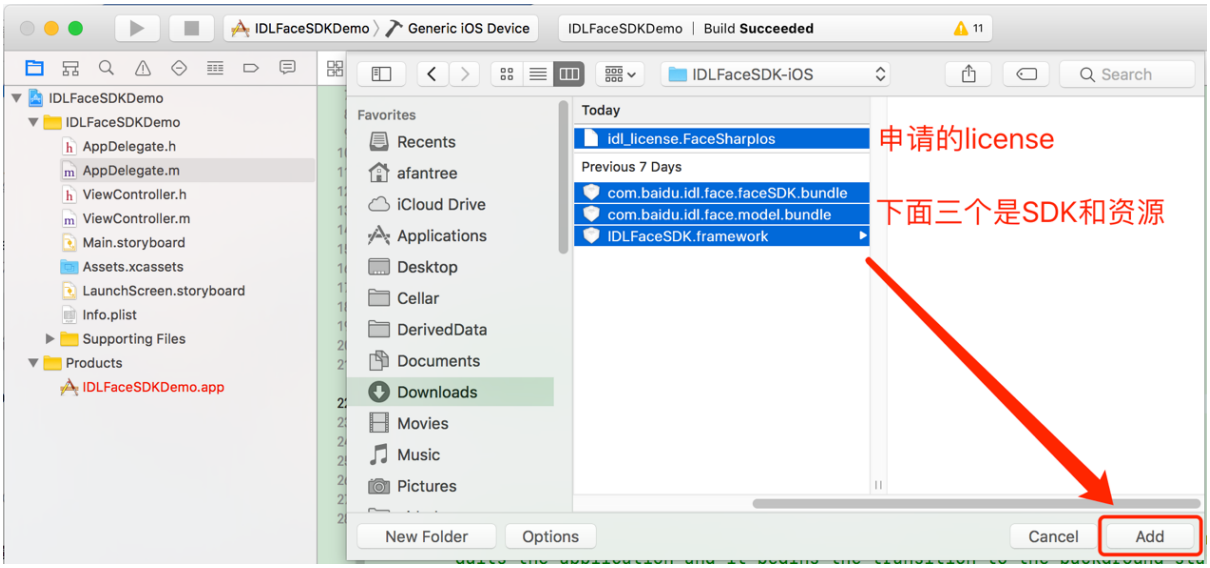
【人脸 SDK 集成】

1. 打开或者新建一个项目。
2. 右键点击项目，会出现一个添加菜单，在菜单中选择『Add Files to “此处是你的项目名字” …… 』，如下图所示：

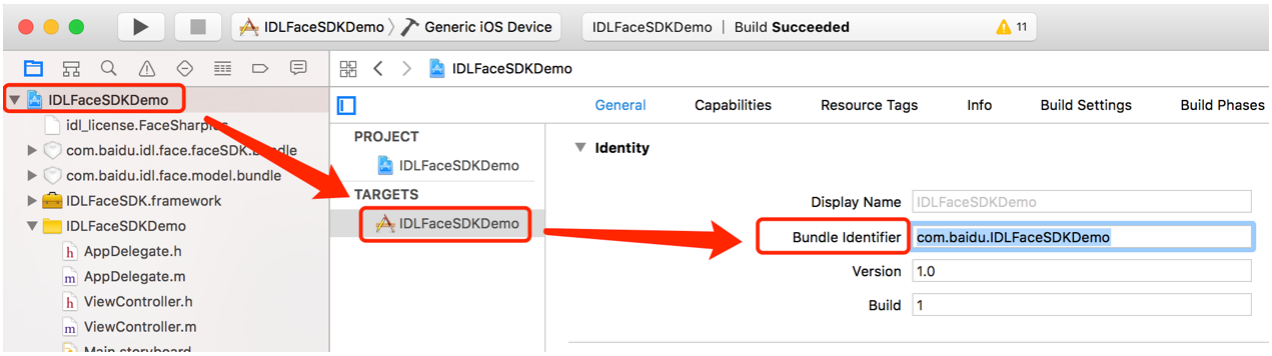


3. 在添加文件弹出框里面选择申请到的 license 和 SDK 添加进来。如下图：
   注意：license 为百度官方提供，刚才在后台下载的文件（文件名称：idl-license.face-ios）

   SDK 包含下面三个文件：
   1) IDLFaceSDK.framework
   2) com.baidu.idl.face.faceSDK.bundle
   3) com.baidu.idl.face.model.bundle



4. 确认下 Bundle Identifier 是否是申请 license 时填报的那一个
5. 注意：license 和 Bundle Identifier 为一一对应关系，填错了会导致 SDK 不可用



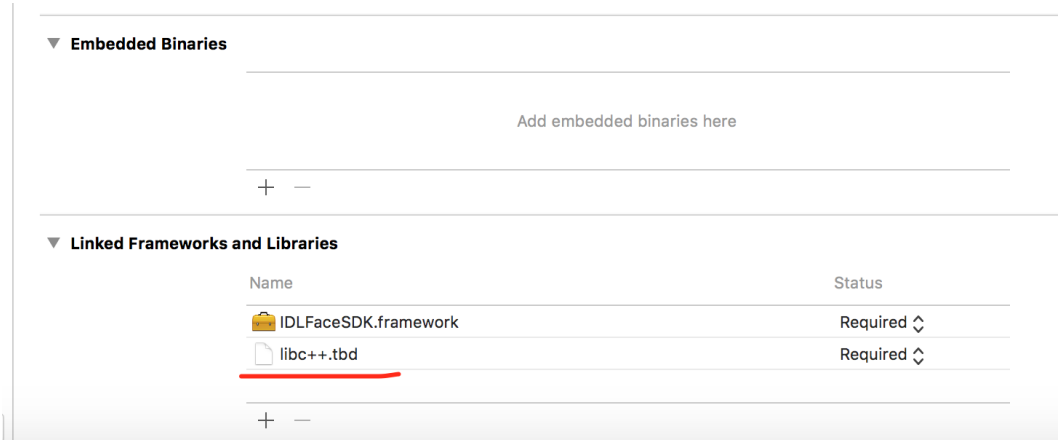
6. FACE_LICENSE_ID 这个参数填写百度官方给的 LicenseID

   在 FaceParameterConfig.h 文件中填写下面三项。

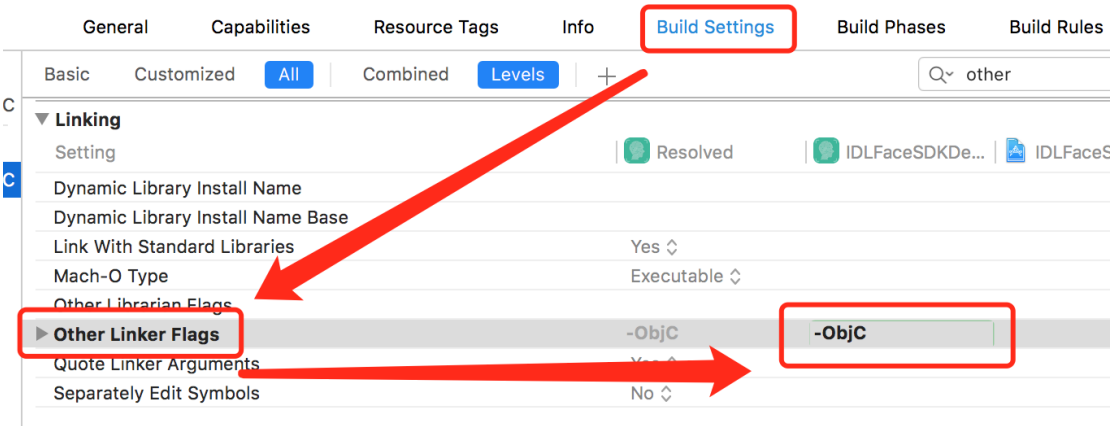
```
11
12 // 人脸license文件名
13 #define FACE_LICENSE_NAME    @"idl-license"
14
15 // 人脸license后缀
16 #define FACE_LICENSE_SUFFIX  @"face-ios"
17
18 //  (您申请的应用名称(appname)+「-face-ios」后缀，如申请的应用名称(appname)为test123，则此处填写test123-face-ios)
19 #define FACE_LICENSE_ID        @"BaiduFaceDemo-face-ios"
```

示例图

7. 选择链接 C++标准库。



8. 如果没有使用 pod 管理第三方库的话，请在 Build Setting → Linking → Other Linker Flags 上面加入 –ObjC 选项。如果用了 pod 请忽略，因为 pod 会自动添加上。



以下为示例工程调用公安验证和在线活体检测的代码片段：

在线活体检测&&公安验证接口：

```
[[NetAccessModel sharedInstance] verifyFaceAndIDCard:self.nameTextField.text idNumber:self.identityCardTextField.text imageStr:
    imageStr completion:^(NSError *error, id resultObject) {
    if (error == nil) {
        NSDictionary* dict = [NSJSONSerialization JSONObjectWithData:resultObject options:NSJSONReadingAllowFragments error:nil];
        NSLog(@"dict = %@",dict);
    } else {
        NSLog(@"网络请求失败");
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        FaceResultType type = FaceResultTypeFail;
        NSString* tip = @"验证分数";
        NSString* scoreStr = @"0";
        if (error == nil) {
            NSDictionary* dict = [NSJSONSerialization JSONObjectWithData:resultObject options:NSJSONReadingAllowFragments error:nil]
                ;
            //NSLog(@"dict = %@",dict);
            if (dict[@"result"] != nil) {
                CGFloat score = [dict[@"result"] floatValue] * 100;
                scoreStr = [NSString stringWithFormat:@"%.4f",score];
                if (score > 80) {
                    type = FaceResultTypeSuccess;
                }
            } else if (dict[@"error_code"] != nil) {
                scoreStr = @"";
                switch ([dict[@"error_code"] intValue]) {
                    case 216600:
                        tip = @"身份证号码错误";
                        break;
                    case 216601:
                        tip = @"身份证号码与姓名不匹配";
                        break;
                    default:
                        break;
                }
            }
        } else {
            NSLog(@"网络请求失败");
            //[Toast showToast:@"网络请求失败"];
        }
        NSMutableDictionary* resultDict = [@{} mutableCopy];
        resultDict[@"type"] = @(type);
        resultDict[@"tip"] = tip;
        resultDict[@"score"] = scoreStr;
        [weakSelf performSegueWithIdentifier:@"Affirm2Result" sender:resultDict];
    });
}];
```

基于接口返回的分数进行业务判断。

2.4 安卓集成

【OCR 身份证识别集成】

1、把申请的 license（aip.license）放到到项目中 assets 目录中

2、修改 app 的 build.gradle 和 AndroidMainest.xml 包名为申请时填入的包名

3、拷贝 ocr-ui 到您的工程中

接下来调用具体请看 Demo 中的 IdCardActivity

1、初始化 OCR SDK

```
private void initOCRSDK() {
    OCR.getInstance().initAccessToken(new OnResultListener<AccessToken>() {
        @Override
        public void onResult(AccessToken result) {
            // 调用成功，返回AccessToken对象
            String token = result.getAccessToken();
        }
        @Override
        public void onError(OCRError error) {
            // 调用失败，返回OCRError子类SDKError对象
        }
    }, getApplicationContext());
}
```

2、调用 api 进行身份证扫描

```
// 生成intent对象
Intent intent = new Intent(IdCardActivity.this, CameraActivity.class);
// 设置临时存储
intent.putExtra(CameraActivity.KEY_OUTPUT_FILE_PATH, FileUtil.getSaveFile(getApplication()).getAbsolutePath());

// 调用拍摄身份证正面的activity
intent.putExtra(CameraActivity.KEY_CONTENT_TYPE, CameraActivity.CONTENT_TYPE_ID_CARD_FRONT);
intent.putExtra(CameraActivity.KEY_NATIVE_TOKEN, OCR.getInstance().getLicense());
intent.putExtra(CameraActivity.KEY_NATIVE_ENABLE, true);
startActivityForResult(intent, REQUEST_CODE_CAMERA);
```

3、调用 api 进行身份证识别

```
IDCardParams param = new IDCardParams();
param.setImageFile(new File(filePath));
param.setIdCardSide(idCardSide);
param.setDetectDirection(true);
OCR.getInstance().recognizeIDCard(param, new OnResultListener<IDCardResult>() {
    @Override
    public void onResult(IDCardResult result) {
        if (result != null) {
            Word idnumberWord = result.getIdNumber();
            Word nameWord = result.getName();
            if (idnumberWord != null) {
                idnumber = idnumberWord.getWords();
            }
            if (nameWord != null) {
                username = nameWord.getWords();
            }

            alertText("识别结果", "idnumber->" + idnumber + " name->" + username);
        }
        displayTip("");
    }

    @Override
    public void onError(OCRError error) {
        alertText("识别结果", error.getMessage());
        displayTip("");
    }
});
```

4、获得身份证号码和姓名后进入人脸活体

详细调用文档，请参考此文档地址：http://ai.baidu.com/docs#/OCR-Android-SDK

==========

【人脸 SDK 集成】

1、授权参数

a、把申请的 license（idl-license.face-android"）放到到项目中 assets 目录中

b、修改 Config 类中的参数

```
// 百度AI官网ai.baidu.com人脸模块创建应用，选择相应模块，然后查看ak,sk(公安权限需要在官网控制台提交工单开启)
// 为了apiKey,secretKey为您调用百度人脸在线接口的，如注册，识别等。
// 为了的安全，建议放在您的服务端，端把人脸传给服务器，在服务端端进行人脸注册、识别放在示例里面是为了您快速看到效果
public static String apiKey = 替换为你的apiKey(ak);
public static String secretKey = 替换为你的secretKey(sk);
public static String licenseID = 替换为你的licenseID,后台SDK管理界面中，已经生成的licenseID,如:test-face-android;
public static String licenseFileName = "idl-license.face-android";
```

c、配置签名（申请 license 时的 md5 为打包签名的文件，所以必须用申请 license 的签名文件）

app->build.gradle->android->signingConfigs

```
    signingConfigs {
        debug {
            keyAlias [redacted]
            keyPassword [redacted]
            storeFile file('../[redacted].jks')
            storePassword('[redacted]')
        }
        release {
            keyAlias '[redacted]'
            keyPassword '[redacted]'
            storeFile file('../[redacted].jks')
            storePassword('[redacted]')
        }
    }
```

d、修改包名 app->build.gradle->android->defaultConfig ->您申请 license 时填的包名

```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.3"
    defaultConfig {
        applicationId "[redacted]"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"

        ndk{
            abiFilters 'armeabi-v7a'
        }

    }
```

2、继承 FaceLivenessActivity，初始人脸 SDK

```
private void initFaceSDK() {
    // 第一个参数 应用上下文
    // 第二个参数 licenseID license申请界面查看
    // 第三个参数 assets目录下的License文件名
    FaceSDKManager.getInstance().initialize(this, Config.licenseID, Config.licenseFileName);
    setFaceConfig();
}
```

3、 设置参数，不设置将使用默认值

```
private void setFaceConfig() {
    FaceConfig config = FaceSDKManager.getInstance().getFaceConfig();
    // SDK初始化已经设置完默认参数（推荐参数），您也根据实际需求进行数值调整
    List<LivenessTypeEnum> livenessList = new ArrayList<>();
    // livenessList.add(LivenessTypeEnum.Mouth);
    livenessList.add(LivenessTypeEnum.Eye);
    // livenessList.add(LivenessTypeEnum.HeadUp);
    // livenessList.add(LivenessTypeEnum.HeadDown);
    // livenessList.add(LivenessTypeEnum.HeadLeft);
    // livenessList.add(LivenessTypeEnum.HeadRight);
    config.setLivenessTypeList(livenessList);

    config.setBlurnessValue(FaceEnvironment.VALUE_BLURNESS);
    config.setBrightnessValue(FaceEnvironment.VALUE_BRIGHTNESS);
    config.setCropFaceValue(FaceEnvironment.VALUE_CROP_FACE_SIZE);
    config.setHeadPitchValue(FaceEnvironment.VALUE_HEAD_PITCH);
    config.setHeadRollValue(FaceEnvironment.VALUE_HEAD_ROLL);
    config.setHeadYawValue(FaceEnvironment.VALUE_HEAD_YAW);
    config.setMinFaceSize(FaceEnvironment.VALUE_MIN_FACE_SIZE);
    config.setNotFaceValue(FaceEnvironment.VALUE_NOT_FACE_THRESHOLD);
    config.setOcclusionValue(FaceEnvironment.VALUE_OCCLUSION);

    config.setCheckFaceQuality(true);
    config.setLivenessRandom(true);
    config.setFaceDecodeNumberOfThreads(4);
    // config.setLivenessRandomCount();
    config.setSound(false);

    FaceSDKManager.getInstance().setFaceConfig(config);
}
```

4、 开始进行活体检测

5、 获取活体检测结果

```
@Override
public void onLivenessCompletion(FaceStatusEnum status, String message, HashMap<String, String> base64ImageMap) {
    super.onLivenessCompletion(status, message, base64ImageMap);
    if (status == FaceStatusEnum.OK && mIsCompletion) {
        // Toast.makeText(this, "活体检测成功", Toast.LENGTH_SHORT).show();
        saveImage(base64ImageMap);
        alertText("检测结果", "活体检测成功");
    } else if (status == FaceStatusEnum.Error_DetectTimeout ||
            status == FaceStatusEnum.Error_LivenessTimeout ||
            status == FaceStatusEnum.Error_Timeout) {
        // Toast.makeText(this, "活体检测采集超时", Toast.LENGTH_SHORT).show();
        alertText("检测结果", "活体检测采集超时");
    }
}
```

6、保持活体检测得到图片，以便进行在线活体检测和公安核实

```
String bestimageBase64 = imageMap.get("bestImage");
Bitmap bmp = base64ToBitmap(bestimageBase64);

// 如果觉的在线校验慢，可以压缩图片的分辨率，目前没有压缩分辨率，压缩质量置80，在nexus5上大概30k，后面版本我们将截出人脸部分，大小应该小于10k
try {
    File file = File.createTempFile(UUID.randomUUID().toString(), ".jpg");
    FileOutputStream outputStream = new FileOutputStream(file);
    bmp.compress(Bitmap.CompressFormat.JPEG, 80, outputStream);
    outputStream.close();

    bestImagePath = file.getAbsolutePath();

} catch (IOException e) {
    e.printStackTrace();
}
```

7、在线活体需要单独的 token（通过 ak，sk 获取，为了防止 ak，sk 泄露，建议把 ak/sk 放在服务端，移动端通过服务端去拉取 token）

```
private void initAccessToken() {

    displayTip(resultTipTV, "加载中");
    APIService.getInstance().init(getApplicationContext());
    APIService.getInstance().initAccessTokenWithAkSk(new OnResultListener<AccessToken>() {
        @Override
        public void onResult(AccessToken result) {
            if (result != null && !TextUtils.isEmpty(result.getAccessToken())) {
                waitAccesstoken = false;
                policeVerify(filePath);
            }else if (result != null) {
                displayTip(resultTipTV, "在线活体token获取失败");
                retBtn.setVisibility(View.VISIBLE);
            } else {
                displayTip(resultTipTV, "在线活体token获取失败");
                retBtn.setVisibility(View.VISIBLE);
            }
        }

        @Override
        public void onError(FaceException error) {
            // TODO 错误处理
            displayTip(resultTipTV, "在线活体token获取失败");
            retBtn.setVisibility(View.VISIBLE);
        }
    }, Config.appKey, Config.appSecret);
}
```

8、根据离线活体得到的人脸，进行在线活体检测和公安验证（参见 FaceOnlineVerifyActivity）

在线活体和公安接口，建议放在您的服务端进行。移动端把离线活体检测的图片传给服务端，服务端通过 ak/sk 拉取 token 后，进行在线活体和公安核实。

如果参选择这两个参数,则检测顺序将会变为: 图片质量检测 -> 活体检测 -> 公安身份验证，其中前两个环节有任意一个条件不通过，则检测流程流程终止，返回信息中会说明不符合的详细内容；如果前两个检测条件全部验证通过，则会执行公安验证接口请求，返回内容包含对比分数、活体检测分数、图片质量检测信息。

```
private void policeVerify(String filePath) {
    if (TextUtils.isEmpty(filePath) || waitAccesstoken) {
        return;
    }
    File file = new File(filePath);
    if (!file.exists() ) {
        return;
    }

    displayTip(resultTipTV, "公安身份核实中...");
    APIService.getInstance().policeVerify(username, idnumber, filePath, new
            OnResultListener<LivenessVsIdcardResult>() {
        @Override
        public void onResult(LivenessVsIdcardResult result) {
            delete();
            if (result != null && result.getScore() > 0.8) {
                displayTip(resultTipTV, "核身成功");
                displayTip(scoreTV, "公安验证分数: " + result.getScore());
            } else {
                displayTip(resultTipTV, "核身失败");
                displayTip(scoreTV, "公安验证分数过低: " + result.getScore());
                retBtn.setVisibility(View.VISIBLE);

            }
        }

        @Override
        public void onError(FaceException error) {
            delete();
            // TODO 错误处理
            // 如返回错误码为: 216600, 则核身失败, 提示信息为: 身份证号码错误
            // 如返回错误码为: 216601, 则核身失败, 提示信息为: 身份证号码与姓名不匹配
            Toast.makeText(FaceOnlineVerifyActivity.this, "公安身份核实失败:" + error.getMessage(), Toast.LENGTH_SHORT)
                    .show();
            retBtn.setVisibility(View.VISIBLE);

        }
    });
}
```

基于得到的活体分数和公安验证分数进行业务判断。

3. 常见问题

Q：iOS 人脸 SDK 中的 apikey 是 bundle id 么？

A：不是，bundle id 是 Xcode 的 Bundle Identifier，如下图所示：

Q：Android 申请时需要填入打包签名的 MD5,该 MD5 如何得来？

A：创建工程时生成签名文件 keystore.jks（该文件会用于最终发布是打包，请认真对待），在命令行工具中使用 keytool -list -v -keystore keystore.jks 得到 md5，删除冒号。

Q：人脸核真都用到那些授权？

A：OCR SDK 需要使用授权文件 aip.license，离线活体需要授权文件 idl-license.face-android，在线活体和公安核实需要 ak/sk