

活体SDK集成文档_Android端

一.准备工作

概述

本文结合示例代码指导您在 Android 应用中集成活体检测SDK，帮助您在 App 中实现刷脸认证功能。

前置条件

- 应用必须在 Android 4.3+ 平台上运行。

快速体验demo

- 可以在GitHub中下载[SDK_ALIVE_DEMO](#)示例工程源码，使用Android studio打开即可运行测试。

开发环境搭建

配置依赖

a.将SDK中libs目录下的aar包拷贝到您工程的libs目录下，如没有该目录需新建。

b.在app的build.gradle文件dependencies中添加aar包依赖：

```
1 repositories {
2     flatDir {
3         dirs '../app/libs'
4     }
5 }
6 dependencies {implementation fileTree(include: ['*.aar'], dir: 'libs')}
```

权限说明

aar包内部已添加，App中可以不配置，此处只做用途说明

权限名称	权限说明	使用场景或目的
INTERNET 网络	允许发送网络请 求、获取网络状态	用于确认用户的终端是否接入网络，以为用 供安全的验证服务
ACCESS_NETWORK_STATE	允许访问网络状态	

		用于确认用户的终端网络连接状态，以为用 供安全的验证服务
ACCESS_WIFI_STATE	允许访问WiFi网络状态信息	用于确认用户的终端WiFi连接状态，以为用 供安全的验证服务
CAMERA 相机	允许拍摄照片、视 频	在用户进行活体动作验证真人时需要使用摄 权限

混淆规则

```
1 -verbose
2 -keep class com.dtf.face.network.model.** {*;}
3 -keep class com.dtf.face.api.IDTCallback {*;}
4 -keep class com.dtf.face.api.IDTFacade {*;}
5 -keep class com.dtf.face.api.DTFacadeBuilder {*;}
6 -keep class com.dtf.face.api.DTResponse {*;}
7 -keep class com.dtf.face.config.**{*;}
8 -keep class com.dtf.face.log.RecordBase {*;}
9 -keep class com.dtf.face.widget.ToygerWebView {*;}
10 -keep class com.alipay.zoloz.toyger.**{*;}
11 -keep class com.alipay.deviceid.** {*;}
12 -keep class com.alipay.rds.** {*;}
13 -keep class com.alipay.android.fintech.log.** {*;}
14 -keep class com.dtf.face.log.** {*;}
15 -keep class com.alipay.bis.common.service.facade.gw.** {*;}
16
17 -keep class com.alibaba.fastjson.** {*;}
18 -keep class org.json.** {*;}
19 -keep class com.chuanglan.sdk.face.api.* {*;}
20 -keep class com.chuanglan.sdk.face.listener.VerifyCallback{*;}
21 -keep class com.chuanglan.sdk.face.entity.VerifyResponse{*;}
```

通过上面的几个步骤，工程就配置完成了

二.API说明

1.初始化

- 【必要方法】为提高用户体验，并为刷脸认证准备必要数据，必须先进行 SDK 初始化。

方法原型

```

1 /**
2  * 初始化接口，放到其他接口调用前
3  *
4  * @param context 传ApplicationContext对象
5  * @param appId 创建应用时生成的appId
6  */
7 public static void initWithAppId(Context context, String appId);

```

示例代码

```

1 FaceVerification.initWithAppId(getApplicationContext(), "您的appid");

```

2.认证参数配置

- 【可选方法】每个配置都有默认值，只需传入想要修改的配置即可，如果不需要修改可以不调用。

方法原型

```

1 /**
2  * 认证参数配置
3  *
4  * @param verifyConfig 参数配置对象
5  */
6 public static void setVerifyConfig(VerifyConfig verifyConfig)

```

VerifyConfig类说明

```

1 /**
2  * 设置活体检测UI界面方向
3  *
4  * @param screenOrientation VerifyConfig.SCREEN_ORIENTATION_PORT (默认)：竖屏；
5  *                           VerifyConfig.SCREEN_ORIENTATION_LAND：横屏
6  */
7 public Builder setScreenOrientation(String screenOrientation);
8
9 /**
10 * 设置是否支持活体视频返回
11 *
12 * @param useVideo true: 支持，并在response.videoFilePath中获取视频本地路径；
13 *                  false(默认)：不支持，返回null
14 */

```

```
15 public Builder setUseVideo(boolean useVideo)
16
17 /**
18  * 设置是否支持活体图片返回
19  *
20  * @param useBitmap true: 支持, 并在response.bitmap中获取图片byte[];
21  *                  false (默认): 不支持, 返回null
22  */
23 public Builder setUseBitmap(boolean useBitmap);
24
25 /**
26  * 设置活体检测页面的进度条颜色
27  *
28  * @param faceProgressColor 颜色值, 如红色 "#FF0000"
29  */
30 public Builder setFaceProgressColor(String faceProgressColor)
31
32 /**
33  * 设置活体检测动作
34  *
35  * @param actionModel VerifyConfig.ACTION_MODEL_LIVENESS (默认): 眨眼动作活体检测
36  *                  VerifyConfig.ACTION_MODEL_MULTI_ACTION: 眨眼+任意摇头检测
37  */
38 public Builder setActionModel(String actionModel)
39
40 /**
41  * 设置启动活体检测界面超时时间
42  *
43  * @param verifyTimeout 超时时间, 单位秒, 如设置6秒传6, 默认值5
44  */
45 public Builder setVerifyTimeout(int verifyTimeout)
46
47 /**
48  * 设置是否在sdk内部使用弹框提示错误信息
49  *
50  * @param useMsgBox true (默认): 使用;
51  *                  false: 不使用
52  */
53 public Builder setUseMsgBox(boolean useMsgBox) {
54     this.useMsgBox = useMsgBox;
55     return this;
56 }
57 /**
58  * 设置是否使用SDK内置协议页
59  *
60  * @param usePrivacyProtocol true (默认): 使用;
61  *                  false: 不使用
```

```

62 */
63 public Builder setUsePrivacyProtocol(boolean usePrivacyProtocol) {
64     this.usePrivacyProtocol = usePrivacyProtocol;
65     return this;
66 }

```

示例代码

```

1 FaceVerification.setVerifyConfig(new VerifyConfig.Builder()
2     .setScreenOrientation(VerifyConfig.SCREEN_ORIENTATION_LAND)
3     .setUseVideo(true)
4     .setUseBitmap(false)
5     .setUseMsgBox(false)
6     .setUsePrivacyProtocol(true)
7     .setFaceProgressColor("#303030")
8     .setActionModel(VerifyConfig.ACTION_MODEL_LIVENESS)
9     .setVerifyTimeout(6)
10    .build());

```

3.开始认证

- 调用此方法会启动SDK内部活体检测界面，并返回刷脸认证结果。

方法原型

```

1 /**
2  * 开始认证
3  * 需在 UI 线程上调用，不会阻塞调用线程
4  *
5  * @param verifyCallback 认证结果的回调接口
6  */
7 public static void faceVerify(VerifyCallback verifyCallback)

```

刷脸认证结果VerifyCallback回调中的VerifyResponse类说明

```

1 public class VerifyResponse {
2     /**
3     * 外层码：
4     * 10000：刷脸结束
5     * 10001：校验失败

```

```
6      * 10002: 验签失败
7      * 10003: 网络异常
8      * 10004: 本地异常
9      */
10     public int code;
11     /**
12      * 内层码:
13      * 1000: 刷脸结束
14      * 1001: 系统错误
15      * 1003: 验证中断
16      * 1014: 本地捕获异常
17      * 1023: 网络原因导致的超时、域名解析异常等
18      * 2002: 网络错误
19      * 2003: 客户端设备时间错误
20      * 2006: 刷脸失败
21      * 400001: 参数校验异常
22      * 600016: Android签名参数异常
23      * 600017: 平台类型非法
24      * 600005: 签名校验失败
25      * 600018: 签名失效
26      * 600004: 包名签名对应的appid不匹配 或 appid未匹配到应用
27      * 500006: 请求外部系统失败
28      * 600019: 包名签名校验失败
29      * 500003: 业务操作失败
30      */
31     public int innerCode;
32
33     /**
34      * 内层结果文案描述, 可能为空
35      */
36     public String innerMsg;
37     /**
38      * 外层结果文案描述
39      */
40     public String msg;
41
42     /**
43      * 设备token
44      */
45     public String deviceToken;
46
47     /**
48      * 如果采用视频返照, 则返回视频的路径, 否则返回null
49      */
50     public String videoFilePath;
51     /**
52      * 查询认证结果所需的唯一标识
```

```

53     */
54     public String certifyId;
55     /**
56     * 如果设置支持照片返回，则返回照片，否则返回null
57     */
58     public byte[] bitmap;
59 }

```

示例代码

```

1 FaceVerification.faceVerify(response -> {
2     try {
3         if (10000 == response.code) {
4             Toast.makeText(this, "认证通过", Toast.LENGTH_LONG).show();
5         } else {
6             Toast.makeText(this, "认证失败([" + response.code + "]" + response.ms
7         }
8     } catch (Exception e) {
9         e.printStackTrace();
10    }
11 });

```

4.查询认证结果

刷脸结束后，请对接服务端查询结果接口，获取活体检测的结果。

5.日志开关

- 放到初始化之前调用，开启后可打印SDK内部调用日志，用“PROCESS_SDK_LOGTAG<-->”过滤日志

方法原型

```

1 /**
2  * 设置SDK基础日志开关
3  *
4  * @param debuggable true:开启;
5  *                  false (默认) : 关闭
6  */
7 public static void setPrintConsoleEnable(Boolean debuggable)

```

示例代码

```
1 FaceVerification.setPrintConsoleEnable(true);
```

三、返回码

外层码	外层描述	内层码	内层描述
10000	刷脸结束	1000	刷脸结束，请通过服务端查询接口获取认证结果（Android端、iOS端）
10001	校验失败	1001	本地代码异常（Android端）
			人脸识别算法初始化失败（Android端）
			不支持的CPU架构（Android端）
			Android系统版本过低（Android端）
			刷脸超时(单次)（Android端、iOS端）
			多次刷脸超时（Android端、iOS端）
			无前置摄像头（Android端）
			摄像头权限未赋予（Android端）
			打开摄像头失败（Android端）
			SDK认证流程正在进行中，请等待本地认证流程完成后再发起新调用
			上传炫彩Meta信息失败（Android端）
			上传炫彩视频失败（Android端）
			用户点击Home键（Android端）
			抱歉，系统出错了，请您稍后再试（iOS端）
			拒绝开通相机权限（iOS端）
			无法启动相机（iOS端）
			本地活体检测出错（iOS端）
			验证中断（用户点击home键等导致验证停止）（iOS端）
			业务参数错误（iOS端）
			本地活体检测出错（iOS端）

		1003	用户主动退出认证（Android端、iOS端）
			用户暂不认证（Android端）
		2001	用户OCR主动退出（iOS端）
		2002	客户端初始化网络错误（Android端）
			客户端初始化接口返回网络错误（Android端）
			信息上传网络错误（Android端）
			服务端认证接口网络错误（Android端）
			服务端接口并发请求超出限制（Android端）
10002	验签失败		网络错误（iOS端）
		2003	客户端设备时间错误（iOS端）
		2006	刷脸结束，请通过服务端查询接口获取认证结果（Android端、iOS端）
		400001	参数校验异常（Android端、iOS端）
		600016	Android签名参数异常（Android端）
		600009	bundleId不能为空（iOS端）
		600017	平台类型非法（Android端、iOS端）
		600005	签名校验失败（Android端、iOS端）
		600018	签名失效（Android端、iOS端）
		600004	包名签名对应的appid不匹配 或 appid未匹配到应用（Android端、iOS端）
		500006	请求外部系统失败（Android端、iOS端）
		600019	包名签名校验失败（Android端、iOS端）
		500003	业务操作失败（Android端、iOS端）
		1010	服务端返回为空（Android端）
10003	网络异常	1023	网络原因导致的超时、域名解析异常等（Android端、iOS端）
10004	本地异常	1014	本地捕获异常（Android端、iOS端）
10005	重复调用	1015	刷脸进行中，请稍后再试
10006	代理异常	1016	代理返回为空

