

一登Andorid平台SDK文档

SDK版本： SUPERID-ANDROID-SDK V1.1.1(20150210)

系统支持： 目前一登SDK支持Android4.0及以上版本，不兼容Android4.0以下版本。

设备支持： 目前一登SDK支持搭载Anroid系统的设备。

1 一登SDK介绍

一登SDK为应用提供了便捷的刷脸登录模式。通过一登账号体系，移动终端用户可快速登录开发者应用。同时，一登SDK提供了人脸信息数据高级功能，让移动开发者可以基于人脸数据，为应用做更精准的数据推送。

一登SDK封装集成了最新的oAuth2.0授权方式，并将通讯过程封装于SDK内部，开发者仅需根据文档说明即可快速集成SDK。

2 名词解释

名词	解析
一登账号 (SuperID)	用户的人脸账号，以手机号码作为账号标识
应用 userID(AppUid)	当前应用用户的userID, 需三方应用提供
一登账号用户 信息	包含用户一登账号的昵称、移动电话、头像等
APP_ID和 APP_SECRET	分配给每个第三方应用的APP_ID、APP_SECRET,用于鉴权应用身份
appToken	用于一登SDK基础通讯使用，SDK初始化成功自动获取，由SDK内部维护，对开发者不可见。
accessToken	用户授权登录成功后获取，用于后续操作（查询、更新信息、获取信息）的用户身份认证使用，其信息对开发者不可见，token状态将通过事件回调通知开发者。

3 集成准备

3.1 SDK接入申请

- 1.在[官网首页](#)点击【免费接入人脸登录】，在一登Github上下载对应的SDK版本、Demo源码和技术文档
- 2.通过下载使用平台提供的 [签名工具](#) 安装到安卓设备，输入准备集成的工程包名，获取应用MD5签名
- 3.回到官网，点击下一步，填写应用包名及MD5签名（包含debug时的签名及打包应用后的签名）等对应的开发者信息
- 4.申请成功后，系统将会发送APP_ID、APP_SECRET到开发者邮箱

注意：要签名的第三方应用程序必须安装在设备上才能够生成对应的 MD5 签名

3.2 SDK接入设置

步骤1： 在一登GitHub下载一登Android SDK，将sdk包中的libs文件夹合并到本地工程的libs目录下，在Eclipse中右键工程根目录，选择Properties -> Java Build Path -> Libraries，然后点击Add External JARs... 选择指向jar的路径，点击OK，即导入成功。（ADT17及以上不需要手动导入）

步骤2： 将sdk包中的res文件夹合并到本地工程的res目录下。

4 配置Manifest

Manifest的配置主要包括添加权限，填写APP_ID、APP_SECRET和添加Activity三部分。

4.1 添加权限

```

<!-- 开启摄像头 -->
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-feature android:name="android.hardware.camera.setParameters" />
<uses-permission android:name="android.permission.CAMERA" />
<!-- 阅读消息 -->
<uses-permission android:name="android.permission.READ_SMS" />
<!-- 接收消息 -->
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<!-- 读写文文件 -->
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" /
>

<!-- 网络状态 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

```

4.2 添加APPID及SECRET

"YOUR_APP_ID"及"YOUR_APP_SECRET"替换为您申请应用的APP_ID、APP_SECRET。

```

<meta-data
    android:name="SUPERID_APPKEY"
    android:value="YOUR_APP_ID" />
<meta-data
    android:name="SUPERID_SECRET"
    android:value="YOUR_APP_SECRET" />

```

4.3 添加Activity

```

<activity
    android:name="com.isnc.facesdk.aty.Aty_FaceDetect"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_AgreeItem"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_Auth"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_EditUserinfo"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_ClipPicture"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_CountryPage"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>
<activity
    android:name="com.isnc.facesdk.aty.Aty_SecureEmail"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Holo.Light.NoActionBar">
</activity>

```

5 初始化SDK

5.1 添加SDK的初始化代码：

在应用入口处Activity的onCreate方法内调用FaceSDKMethod.initFaceSDK(this)初始化SDK，示例代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FaceSDKMethod.initFaceSDK(this);
}
```

5.2 开启Debug模式：

集成调试过程中若需查看log日志，请在应用入口处Activity的onCreate方法内调用调用FaceSDKMethod.setDebugMode(true),否则默认为false。

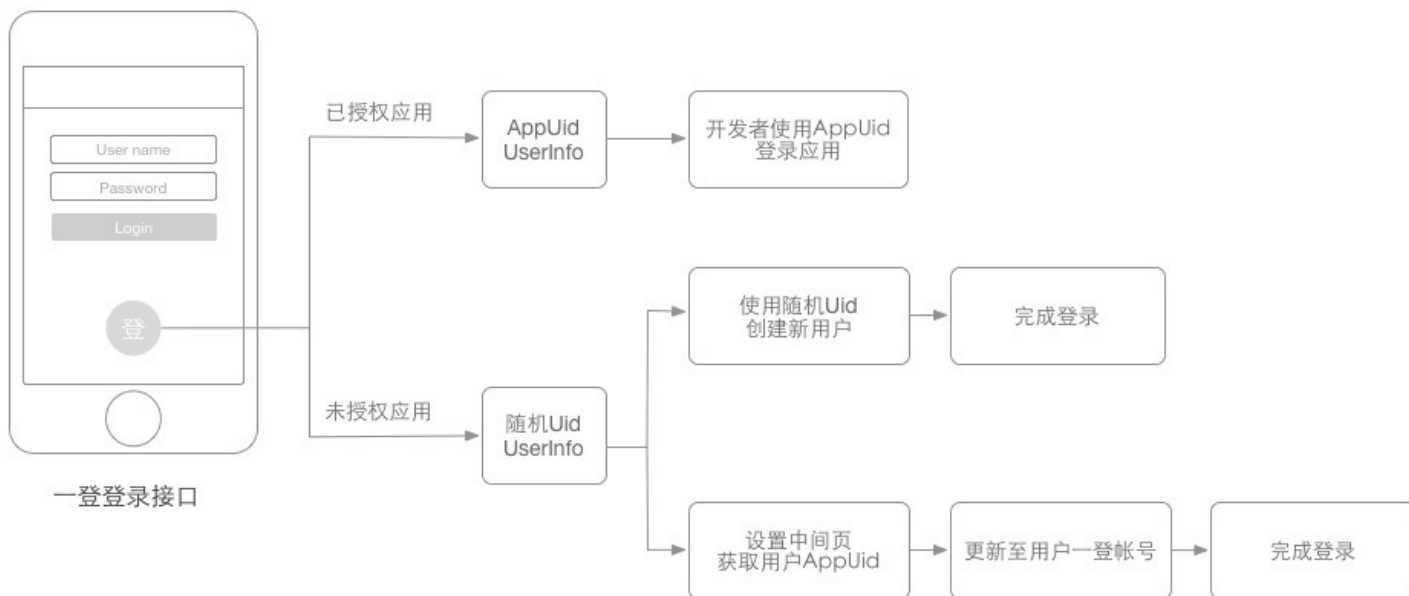
6 一登账号登录绑定接口

6.1 一登登录绑定场景使用

使用一登账号实现快速刷脸登录，需完成用户一登账号和应用账号的授权关联过程。以下分为两种情况进行指南：

场景1：

当用户尚未处于为登录应用状态（开发者未知当前用户Uid）。此时用户点击登录界面的一登按钮（API见6.2登录绑定场景接口），开发者调用登录接口FaceLogin(Context context)，用户第一次点击一登登录按钮触发登录接口时，会提示输入手机号码，根据手机号码判断是否已经收授权开发者应用。用户授权绑定后第二次登录时，系统会默认使用第一次用户输入的手机号码，用户无需输入手机号码，直接刷脸即可。



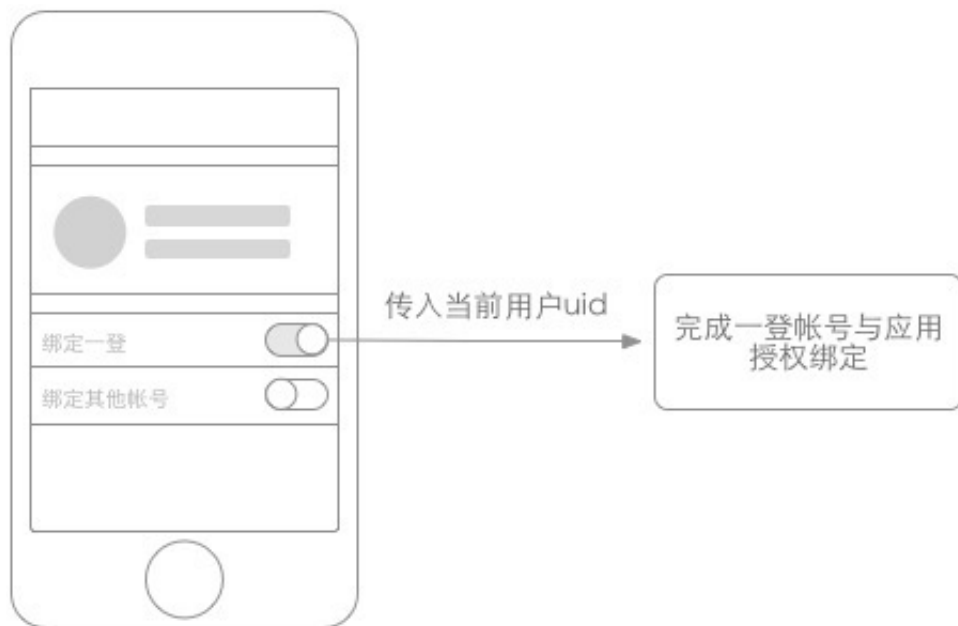
第一、如果当前用户尚未授权，我们会引导其授权你的应用，并返回一登的UserID和一登账号的用户信息。此时，建议开发者根据自己应用的需求选择操作：

- 1. 开发者没有自己的用户体系：使用一登的UserID及用户信息直接作为开发者应用的用户uid及用户信息；
- 2. 开发者拥有自己的用户体系：(1) 当前用户还未注册过开发者应用，则为当前一登用户创建应用账号，将开发者账号体系的AppUid更新到当前的一登用户账号中（更新appuid接口见6.2 登录绑定场景接口），完成账号关联。(2) 当前用户已经注册过开发者应用，则自行设置中间绑定页，引导用户输入在你应用中的账号密码，获取该用户的AppUid后，更新AppUid到当前的一登用户账号（更新appuid接口见6.2 登录绑定场景接口），完成账号关联。

第二、如果当前一登用户已授权你的应用，直接登录成功后，我们会返回当前用户的userID(若开发者没有自己的用户体系，则返回一登的UserID；若开发者拥有自己的用户体系且调用uid更新接口，则返回开发者账号体系的AppUid)和一登账号的用户信息。

场景2：

当用户处于为登录应用状态（开发者已知当前AppUid）。则调用一登授权绑定接口（绑定接口见6.2 登录绑定场景接口），直接完成账号关联，用户再次登录应用时即可直接返回开发者账号体系的AppUid。



6.2 登录绑定场景接口

- 登录接口：FaceSDKMethod.FaceLogin(this)

在应用登录界面添加button，调用FaceSDKMethod.FaceLogin(this)执行人脸登录，若用户未绑定一登账号，则此接口会跳转到绑定功能。接口返回采用onActivityResult获取返回结果，详见接口返回说明。

- 其他登录接口：FaceLoginWithPhoneUid(Activity activity, String phone, String userinfo, String uid)

若开发者应用中已知手机号码，uid，则在应用登录界面添加button，调用FaceLoginWithPhoneUid(Activity activity, String phone, String userinfo, String uid) 执行人脸登录，userinfo需调用SendinfoToSuperID(String... kvs)接口进行格式化,详见7.4 格式化userinfo，若没有userinfo或者uid，请传入""空字符串。若用户未绑定一登账号，则此接口会跳转到绑定功能。接口返回采用onActivityResult获取返回结果，详见接口返回说明。传入的手机号码目前只支持中国手机号码。

有userinfo, uid:

```
FaceSDKMethod.FaceLoginWithPhoneUid(this, "13888888888", String userinfo, String uid);
```

无userinfo, uid:

```
FaceSDKMethod.FaceLoginWithPhoneUid(this, "13888888888", "", "");
```

- 更新appuid接口：UpdateAppUid(Context context, String uid, IntSuccessCallback successCallback, IntFailCallback failCallback)

若开发者需要自己的用户体系，需将调用此接口。在调用FaceLogin(Context context)进行注册 授权后，开发者在自己的用户体系内完成注册流程，并使用此接口将自己用户体系的uid与一登账号进行绑定。IntSuccessCallback 为解除绑定成功的回调，IntFailCallback为解除绑定失败的回调。示例代码如下：

```

FaceSDKMethod.UpdateAppUid(context, "uid123456", new IntSuccessCallback()
    @Override
        public void onSuccess(int arg0) {
            // TODO 处理成功事件
        }
}, new IntFailCallback() {
    @Override
        public void onFail(int arg0) {
            // TODO 处理失败事件
            //int arg0为返回错误类型，若需详细处理返回错误类型，则相应地错误码如下：
            switch (error) {
                case SDKConfig.ACCESSTOKEN_EXPIRED:
                    Toast.makeText(context, "更新失败,access_token过期", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.APPTOKEN_EXPIRED:
                    Toast.makeText(context, "更新失败,app_token过期", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.ACCESSTOKENERROR:
                    Toast.makeText(context, "更新失败,access_token失效", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.APPTOKENERROR:
                    Toast.makeText(context, "更新失败,app_token失效", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.OTHER_ERROR:
                    Toast.makeText(context, "更新失败,网络连接错误", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.SDKVERSIONEXPIRED:
                    Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
});

```

- 绑定账号：若开发者应用账号体系中无手机号码，请调用FaceBundle(Activity activity, String uid, String userinfo)

在应用个人中心界面添加button，调用FaceSDKMethod.FaceBundle(this,uid,userinfo)将应用的账号与一登绑定。此接口需传入应用用户的uid（必选），应用用户的信息userinfo（可选）。userinfo需调用SendinfoToSuperID(String... kvs)接口进行格式化，详见7.4 格式化userinfo，userinfo若没有，请置为“”。接口返回采用onActivityResult获取返回结果，详见接口返回说明。示例代码如下：


```
无userinfo:  
FaceSDKMethod.FaceBundle(this,uid,"")。  
有userinfo:  
FaceSDKMethod.FaceBundle(this,uid,userinfo)。
```

- 绑定账号：若开发者应用账号体系中有手机号码，请调用FaceBundleWithPhone(Activity activity, String uid,String userinfo, String phone)

在应用个人中心界面添加button，调用FaceBundleWithPhone(Activity activity, String uid,String userinfo, String phone)将应用的账号与一登绑定。此接口需传入应用用户的uid（必选），应用用户的信息userinfo(可选),phone(必选)。userinfo需调用SendinfoToSuperID(String... kvs)接口进行格式化，详见7.4 格式化userinfo，userinfo若没有，请置为“”。接口返回采用onActivityResult获取返回结果，详见接口返回说明。传入的手机号码目前只支持中国手机号码。示例代码如下：

```
无userinfo:  
FaceSDKMethod.FaceBundle(this,uid,"","13888888888")。  
有userinfo:  
FaceSDKMethod.FaceBundle(this,uid,userinfo,"13888888888")。
```

7 用户信息相关接口

7.1 用户授权状态查询接口：

FaceSDKMethod.UidAuthorized(Context context, String uid, IntSuccessCallback successCallback,IntFailCallback failCallback)

开发者需要在用户选择绑定或解绑操作前进行用户绑定状态的查询，并根据查询结果引导用户完成一登账号的其他操作或更改UI界面的显示。

```

FaceSDKMethod.UidAuthorized(context, "uid1231241", new FaceSDKMethod.IntSuccessCal
lback() {
    @Override
    public void onSuccess(int result) {
        switch (result) {
            case SDKConfig.ISAUTHORIZED:
                Toast.makeText(context, "已授权", Toast.LENGTH_SHORT).show();
                break;
            case SDKConfig.NOAUTHORIZED:
                Toast.makeText(context, "未授权", Toast.LENGTH_SHORT).show();
                break;
            default:
                break;
        }
    }
}, new FaceSDKMethod.IntFailCallback() {
    @Override
    public void onFail(int error) {
        switch (error) {
            case SDKConfig.APPTOKENERROR:
                Toast.makeText(context, "apptoken错误", Toast.LENGTH_SHORT).show();
                break;
            case SDKConfig.OTHER_ERROR:
                Toast.makeText(context, "网络连接错误", Toast.LENGTH_SHORT).show();
            case SDKConfig.SDKVERSIONEXPIRED:
                Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show();
                break;
            case SDKConfig.APPTOKEN_EXPIRED:
                Toast.makeText(context, "apptoken过期，您已经太久没操作，请重新登录", Toast.L
LENGTH_SHORT).show();
                break;
            default:
                break;
        }
    }
});

```

7.2 解除绑定：FaceSDKMethod.Unbundle(Context, IntSuccessCallback successCallback, IntFailCallback failCallback)

解除当前app用户与一登账号的绑定，IntSuccessCallback 为解除绑定成功的回调，IntFailCallback为解除绑定失败的回调。示例代码如下：

```

FaceSDKMethod.Unbundle(context, new FaceSDKMethod.IntSuccessCallback()
    @Override
        public void onSuccess(int arg0) {
            // TODO 处理成功事件
        }
}, new FaceSDKMethod.IntFailCallback() {
    @Override
        public void onFail(int arg0) {
            // TODO 处理失败事件
            //int arg0为返回错误类型，若需详细处理返回错误类型，则相应地错误码如下：
            switch (error) {
                case SDKConfig.ACCESS_TOKEN_EXPIRED:
                    Toast.makeText(context, "解绑失败,access_token过期", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.APP_TOKEN_EXPIRED:
                    Toast.makeText(context, "解绑失败,app_token过期", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.ACCESS_TOKEN_ERROR:
                    Toast.makeText(context, "解绑失败,access_token失效", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.APP_TOKEN_ERROR:
                    Toast.makeText(context, "解绑失败,app_token失效", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.OTHER_ERROR:
                    Toast.makeText(context, "解绑失败,网络连接错误", Toast.LENGTH_SHORT).show();
                    break;
                case SDKConfig.SDK_VERSION_EXPIRED:
                    Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
});

```

7.3 更新appinfo: UpdateAppInfo(Context context, String userinfo, IntSuccessCallback successCallback, IntFailCallback failCallback)

此接口用于用户同步更新开发者应用的用户信息到一登。userinfo需调用SendinfoToSuperID(String... kvs)接

口进行格式化，详情请查看SendinfoToSuperID接口。IntSuccessCallback 为更新成功的回调，IntFailCallback为更新失败的回调。示例代码如下：

```
FaceSDKMethod.UpdateAppInfo(context, userinfo, new FaceSDKMethod.IntSuccessCallbac
k() {
    @Override
    public void onSuccess(int result) {
        Toast.makeText(context, "更新成功", Toast.LENGTH_SHORT).show();
    }
}, new FaceSDKMethod.IntFailCallback() {
    @Override
    public void onFail(int error) {
        switch (error) {
            case SDKConfig.ACCESS_TOKEN_EXPIRED:
                Toast.makeText(context, "更新失败,access_token过期", Toast.LENGTH_SH
ORT).show();
                break;
            case SDKConfig.APPTOKEN_EXPIRED:
                Toast.makeText(context, "更新失败,app_token过期", Toast.LENGTH_SHORT
).show();
                break;
            case SDKConfig.ACCESS_TOKEN_ERROR:
                Toast.makeText(context, "更新失败,access_token失效", Toast.LENGTH_SH
ORT).show();
                break;
            case SDKConfig.APPTOKEN_ERROR:
                Toast.makeText(context, "更新失败,app_token失效", Toast.LENGTH_SHORT
).show();
                break;
            case SDKConfig.OTHER_ERROR:
                Toast.makeText(context, "更新失败,网络连接错误", Toast.LENGTH_SHORT).s
how();
            case SDKConfig.SDK_VERSION_EXPIRED:
                Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show(
);
                break;
            default:
                break;
        }
    }
});
```

7.4 格式化userinfo: SendinfoToSuperID(Context context, String... kvs)

在调用绑定接口FaceBundl时，需格式化userinfo信息,传入的参数为字符串数组，(key1,value1,key2,value2,key3,value3...)建议“昵称”的key为“name”，“头像”的key为“avatar”，“邮箱”的key为“email”。示例代码如下：

```
String userinfo = FaceSDKMethod
    .SendinfoToSuperID(this, "name", "SuperID",
        "email","superid@superid.me",
        "avatar","http://superid.me/superid/superidavatar.jpg");
```

7.5 退出账号：FaceLogout(Context context)

调用FaceSDKMethod.FaceLogout(context)登出。

8 接口返回说明

接口返回采用onActivityResult获取返回结果，示例如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    switch (resultCode) {
        case SDKConfig.AUTH_SUCCESS:
            <!-- phonenum 为SuperID用户的phone -->
            String phonenum = Cache.getCached(context, SDKConfig.KEY_PHONENUM);
            <!-- uid 为开发者应用的uid，若调用FaceLogin(Context context)进行注册授权，则
            系统将会自动生成一个uid -->
            String uid = Cache.getCached(context, SDKConfig.KEY_UID);
            <!-- superidinfo 为SuperID用户信息，格式为json -->
            String superidinfo = Cache.getCached(context, SDKConfig.KEY_APPINFO);
            Toast.makeText(context, "授权成功", Toast.LENGTH_SHORT).show();
            break;
        case SDKConfig.AUTH_BACK:
            Toast.makeText(context, "取消授权", Toast.LENGTH_SHORT).show();
            break;
        case SDKConfig.USER_NOTFOUND:
            Toast.makeText(context, "用户不存在", Toast.LENGTH_SHORT).show();
            break;
        case SDKConfig.LOGINSUCCESS:
            <!-- phonenum 为SuperID用户的phone -->
            String phonenum = Cache.getCached(context, SDKConfig.KEY_PHONENUM);
            <!-- uid 为开发者应用的uid，若用户在调用FaceLogin(Context context)进行注册授
            权，则系统将会自动生成一个uid，重新登录成功时返回次uid -->
            String uid = Cache.getCached(context, SDKConfig.KEY_UID);
```

```

        <!-- superidinfo 为SuperID用户信息, 格式为json -->
        String superidinfo = Cache.getCached(context, SDKConfig.KEY_APPINFO);
        Toast.makeText(context, "登录成功", Toast.LENGTH_SHORT).show();
        break;
    case SDKConfig.LOGINFAIL:
        Toast.makeText(context, "登录失败", Toast.LENGTH_SHORT).show();
        break;
    case SDKConfig.BUNDLEFAIL:
        Toast.makeText(context, "绑定失败", Toast.LENGTH_SHORT).show();
        break;
    case SDKConfig.NETWORKFAIL:
        Toast.makeText(context, "网络连接错误", Toast.LENGTH_SHORT).show();
        break;
    case SDKConfig.SUPERIDEXIST:
        Toast.makeText(context, "superid已经被绑定", Toast.LENGTH_SHORT).show();
        break;
    case SDKConfig.GETEMOTIONRESULT:
        <!-- facedata 为人脸数据, 格式为json -->
        String facedata = data.getStringExtra(SDKConfig.FACEDATA);
        break;
    case SDKConfig.PHONECODEOVERLIMIT:
        Toast.makeText(context, "验证码请求次数超过限制", Toast.LENGTH_SHORT).show
        ();
        break;
    case SDKConfig.SDKVERSIONEXPIRED:
        Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show();
        break;
    default:
        break;
    }
}

```

接口状态码对应值

```

//登录成功
public static final int LOGINSUCCESS = 101;
//登录失败
public static final int LOGINFAIL = 102;
//没有该用户
public static final int USER_NOTFOUND = 103;
//uid已被占用
public static final int SUPERIDEXIST = 104;
//绑定失败
public static final int BUNDLEFAIL = 105;
//网络连接有误
public static final int NETWORKFAIL = 106;
//用户没有授权
public static final int USER_NOTAUTH = 107;
//手机验证码获取次数太频繁
public static final int PHONECODEOVERLIMIT = 108;
//授权成功
public static final int AUTH_SUCCESS = 109;
//取消授权
public static final int AUTH_BACK = 110;
//uid为空
public static final int UID_ISNULL = 111;
//获取人脸数据成功
public static final int GETEMOTIONRESULT = 112;
//获取人脸数据失败
public static final int GETEMOTION_FAIL = 113;
//token为空
public static final int TOKEN_ISNULL = 114;
//app_token有误
public static final int APPTOKENERROR = 115;
//access_token有误
public static final int ACCESSTOKENERROR = 116;
//一登sdk版本过低
public static final int SDKVERSIONEXPIRED = 117;
//request_token过期
public static final int REQUESTTOKENEXPIRED = 118;
//app_token过期
public static final int APPTOKEN_EXPIRED = 1006;
//access_token过期
public static final int ACCESSTOKEN_EXPIRED = 1007;

```

9 混淆说明

若开发者应用需用proguard混淆打包，需在proguard配置文件中添加以下代码，确保混淆打包后的应用不会崩溃。

```
-libraryjars libs/SuperIDSDK.jar
-libraryjars libs/httpmime-4.1.3.jar
-libraryjars libs/armeabi-v7a/libface-jni.so
-libraryjars libs/armeabi-v7a/libfacesdk.so
-keep class **.R$* {*;}
-keep class com.isnc.facesdk.aty.**{*;}
-keep class com.isnc.facesdk.**{*;}
-keep class com.isnc.facesdk.common.**{*;}
-keep class com.isnc.facesdk.net.**{*;}
-keep class com.isnc.facesdk.view.**{*;}
-keep class com.isnc.facesdk.viewmodel.**{*;}
-keep class com.matrixcv.androidapi.face.**{*;}
```

10 运行DEMO使用说明

- 1.导入工程,目前整个工程全采用中文注释,为了防止乱码滋生,请修改文本编码方式 为 UTF-8。
- 2.修改debug.keystore,MD5工具是根据keystore来生成签名的,不同的keystore 生成的签名是不一样的。
此 Demo 的签名是用官网提 供的 keystore 生成的,若要顺利运行 Demo 程序,需要进行设置或是替换 keystore。除了编译运行官方 DEMO 外,不要直接使用它,出于安全的考虑,用户应该为自己的应用提供一份 keysotre。