一登SDK接口文档V1.1(builad20150119)

一登SDK使用流程

- 1.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;
- 2.在首页点击【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;
- 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SUPERID_APPKEY、SUPERID_SECRET到开发者邮箱;
- 5.阅读技术文档并集成SDK;
- 6.测试;

集成准备

1.1 申请集成

> 集成SDK之前，您需要在首页点击 【免费接入人脸登录】 申请SDK使用，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，提交应用包名及MD5签名（包含debug时的签名及打包应用后的签名）等信息获取SUPERID_APPKEY、SUPERID_SECRET。

1.2 导入SDK

> 下载SDK，将下载包中的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）然后下载包中的res文件夹合并到本地工程的res目录下。

配置Manifest

> Manifest的配置主要包括添加权限，填写SUPERID_APPKEY、SUPERID_SECRET和添加Activity三部分。

2.1 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" />
```

2.2 添加APPID及SECRET

> "YOUR*APP*ID"及“YOUR*APP*SECRET”替换为您申请应用的SUPERID_APPKEY、SUPERID_SECRET。

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

2.3 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>
```

接口使用说明

3.1 初始化SDK：initFaceSDK(Context context)

> 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);
    }
```

3.2 登录注册：若开发者应用账号体系中无手机号码，请调用FaceLogin(Context context)

> 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。

3.3 登录注册：若开发者应用账号体系中有手机号码，请调用FaceLoginWithPhoneUid(Activity activity, String phone, String userinfo, String uid)

> 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，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 u
id);
无userinfo，uid：
FaceSDKMethod.FaceLoginWithPhoneUid(this, "13888888888", "", "");
```

3.4 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，详情请查看SendinfoToSuperID接口，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)。
```

3.5 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，详情请查看SendinfoToSuperID接口，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")。
```

3.6 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");
```

3.7 人脸信息：GetFaceEmotion(Context context)

> 调用FaceSDKMethod.GetFaceEmotion(Context context)获取人脸信息。接口返回采用onActivityResult获取返回结果，详见接口返回说明。

3.8 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.ACCESSTOKEN_EXPIRED:
                Toast.makeText(context, "解绑失败,access_token过期", Toast.LENGTH_SH
ORT).show();
                break;
            case SDKConfig.APPTOKEN_EXPIRED:
                Toast.makeText(context, "解绑失败,app_token过期", Toast.LENGTH_SHORT
).show();
                break;
            case SDKConfig.ACCESSTOKENERROR:
                Toast.makeText(context, "解绑失败,access_token失效", Toast.LENGTH_SH
ORT).show();
                break;
            case SDKConfig.APPTOKENERROR:
                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).s
how();
                break;
            case SDKConfig.SDKVERSIONEXPIRED:
                Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show(
);
                break;
            }
        }
    });
```

3.9 更新appuid：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_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.ACCESSTOKENERROR:
                Toast.makeText(context, "更新失败,access_token失效", Toast.LENGTH_SH
ORT).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).s
how();
                break;
            case SDKConfig.SDKVERSIONEXPIRED:
                Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show(
);
                break;
            }
        }
    });
```

3.10 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.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.ACCESSTOKEN_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.ACCESSTOKENERROR:
                Toast.makeText(context, "更新失败,access_token失效", Toast.LENGTH_SH
ORT).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).s
how();
            case SDKConfig.SDKVERSIONEXPIRED:
                Toast.makeText(context, "一登SDK版本过低", Toast.LENGTH_SHORT).show(
);
                break;
            default:
                break;
            }
        }
    });
```

3.11 检测应用uid是否授权：

> 用于检测此uid是否授权，传入uid为开发者应用的用户uid，可用户个人中心绑定状态的检测。

```
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
ENGTH_SHORT).show();
           break;
        default:
            break;
        }
        }
    });
```

3.12 退出账号：FaceLogout(Context context)

> 调用FaceSDKMethod.FaceLogout(context)登出。

3.13 开启Debug模式：setDebugMode(boolean mode)

> 集成过程中若需查看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。

3.14 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;
```

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.**{*;}
```

深度合作接口：隐藏刷脸界面获取人脸信息

> 开发者创建一个Activity，继承Aty*FaceEmotion，同时对Aty*FaceEmotion的方法进行重写。RequestFaceData()为已获取人脸，开始请求数据；FaceDataCallback(String arg0)为处理获取到得人脸信息，其中arg0为人脸信息；GetFaceDataFail()为获取人脸信息失败。同时相应地创建一个布局文件xml，添加如下布局代码：

```
<FrameLayout
        android:id="@+id/container"
        android:layout_width="1dp"
        android:layout_height="1dp"
        android:gravity="center"
        android:visibility="invisible" >
        <SurfaceView
            android:id="@+id/infoSurfaceView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
        <com.isnc.facesdk.view.FaceRegistView
            android:id="@+id/mFaceRegistView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center" />
</FrameLayout>
```

> 详细使用请参见demo中Aty_GetFaceEmotion。