PDX-插件 SDK 用户手册

Android

2017-3-1

北京全息智信科技有限公司

目录

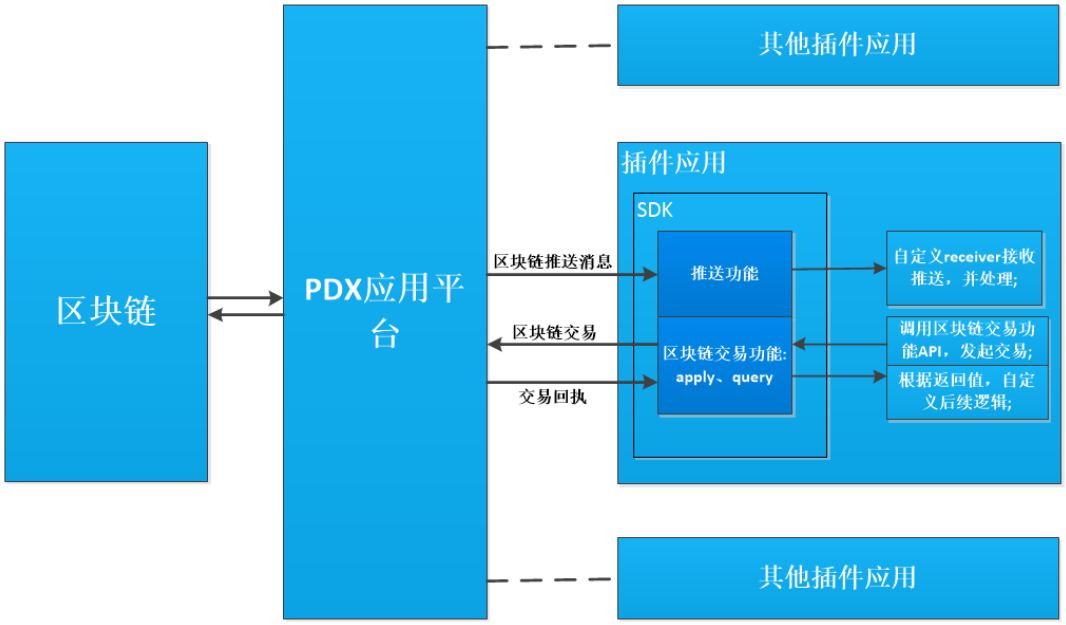
Android SDK

阅读提示

本文是 pdx-plugin Android SDK 标准的概述、集成、API 使用文档。目标读者为对区块链移动应用有兴趣的读者、开发者、合作伙伴等。

概述

区块链、PDX 平台、SDK、插件应用之间关系



SDK 特点

开发者需要了解,所开发的移动应用(插件应用)是运行在 PDX 移动平台上的。运行在平台上，可以实现三大功能：与区块链交互、接收区块链消息推送、自治身份系统。SDK 提供了简洁的 API，轻松几行代码即可完成集成。集成 SDK 后，插件即可运行在 PDX 移动平台上。

SDK 支持版本

目前 SDK 支持 API Level >= 11（Android 3.0 及以上）的各个版本的手机系统。

SDK 集成及 API 使用

*SDK 集成

*SDK API 使用

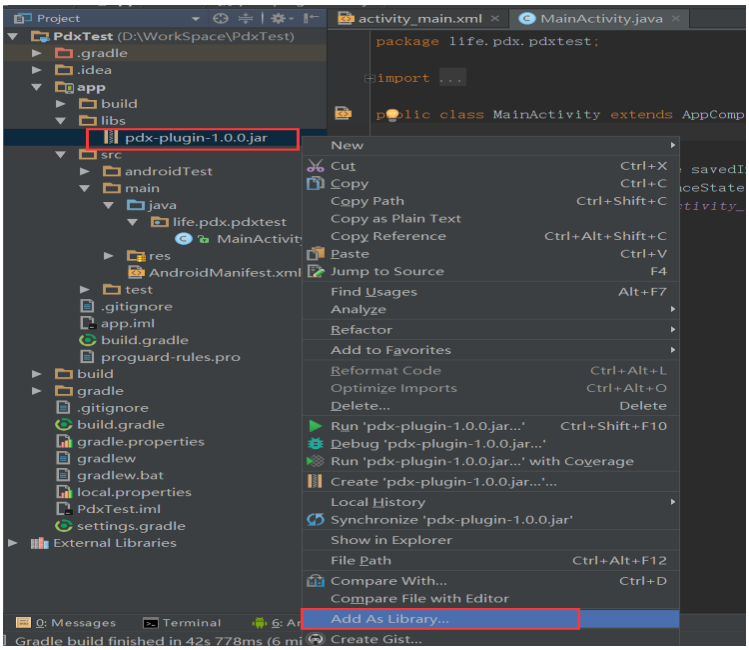
SDK 集成

使用提示

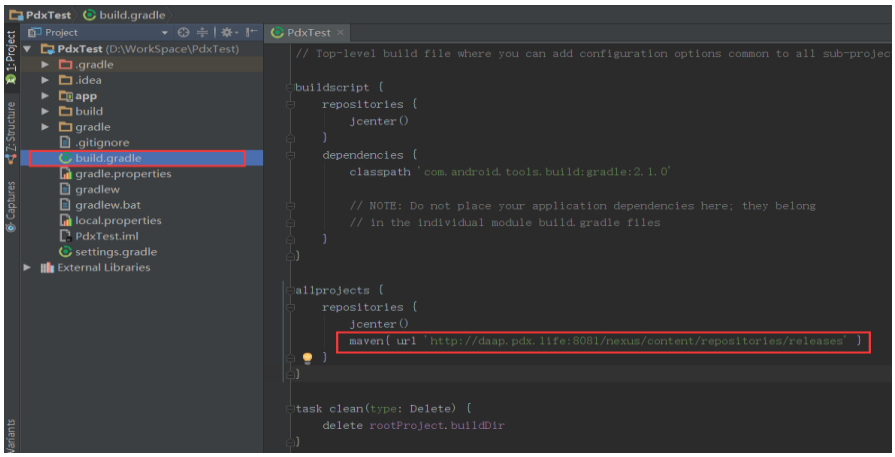
该部分是 pdx-plugin Android SDK 的集成指南。用以指导 SDK 的集成，默认读者已经熟悉 IDE（Android Studio、Eclipse 等）的基本使用方法，已经具有一定的 Android 编程知识基础。下文操作均以 Android Studio 为例介绍，其他 IDE 使用者，不同处请自行调整。

手动集成步骤

将与该文档所在目录下的pdx-plugin-.*.*.jar添加到你项目的libs/目录下，并执行Add As Library…操作。（本文中*.*.*均代表版本号）



* 在项目 build.gradle 中，增加一行 maven 仓库地址配置：
maven{ url 'http://daap.pdx.life:8081/nexus/content/repositories/releases' }

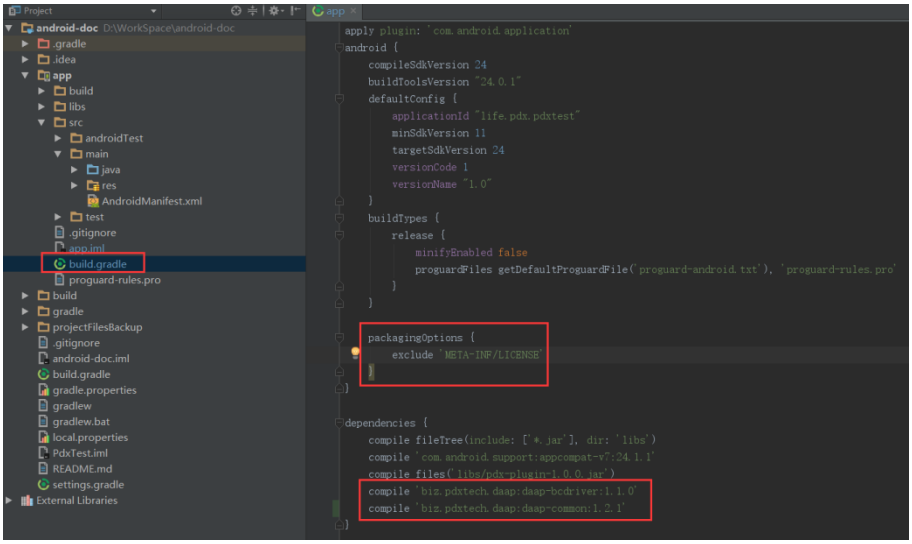


*配置完 maven 仓库后，在主 Module 的 build.gradle 中，增加如下配置：

1、 compile 'biz.pdxtech.daap:daap-bcdriver:1.1.0'

2、 compile 'biz.pdxtech.daap:daap-common:1.2.1'

3、 packagingOptions（否则，Android Studio2.2.3 以上版本会报错）



注意，完成以上步骤后：

如需集成区块链交易功能，请参考下文与区块链交互配置部分；

如需集成消息推送功能或自治身份系统，请参考下文消息推送和自治身份系统配置部分；

如需混淆，请参考下文混淆配置部分；

一、与区块链交互配置

你的项目中需要先创建一个自定义的 Application，例如 MyApplication。然后：

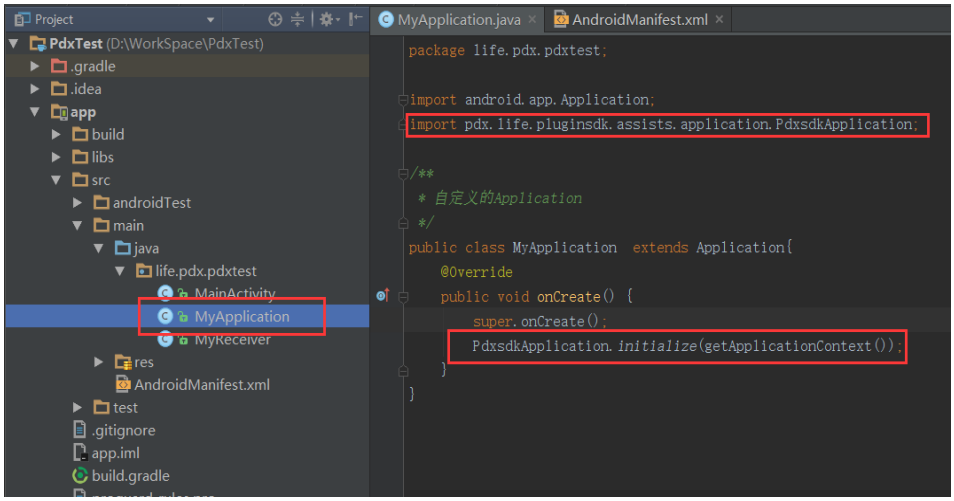
* 在 MyApplication 覆写的 onCreate 方法中，增加一行代码：

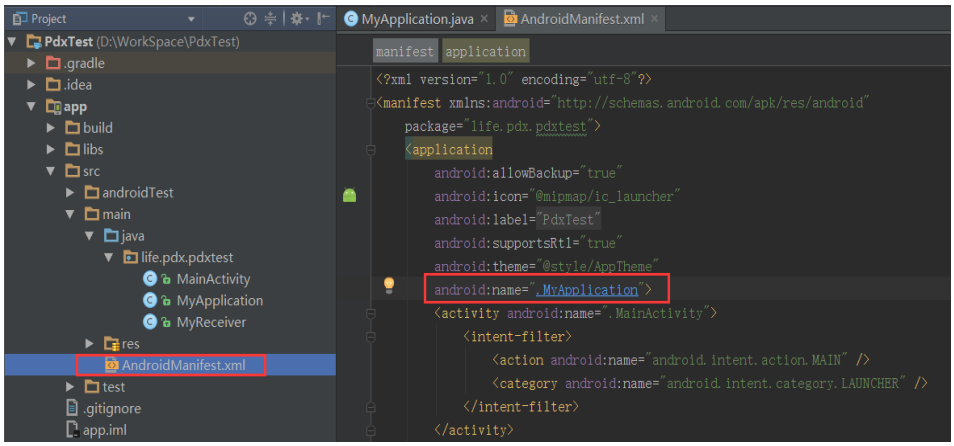
```
PdxsdkApplication.initialize(getApplicationContext());
```

注意：

1、需要 `import pdx.life.pluginsdk.assists.application.PdxsdkApplication;`

2、别忘了在 Manifest 中注册 MyApplication！





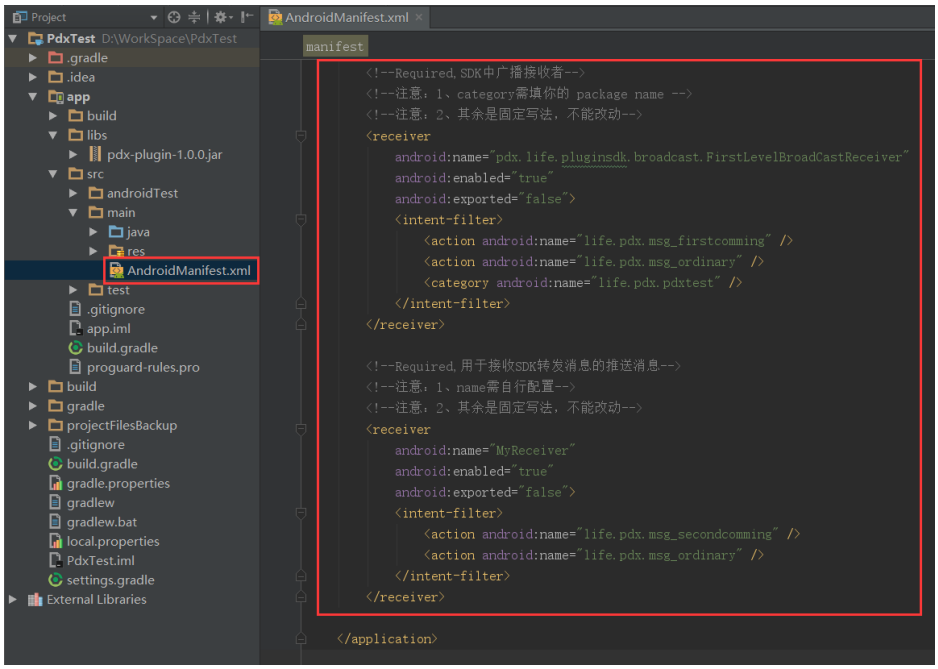
至此，与区块链交互功能的配置已完成。

二、消息推送和自治身份系统配置

消息推送和自治身份系统配置，都和 BroadcastReceiver 有关，两个功能需先统一配置 Manifest。然后，在自定义的 BroadcastReceiver 中区分广播消息是属于消息推送还是自治身份系统。请照下面步骤来配置。

配置 AndroidManifest.xml

参照下图及步骤，来配置应用程序的 AndroidManifest.xml。



步骤为：

*复制下面代码到项目的 AndroidManifest.xml 中；

*将备注为“** your package name **”的部分，替换为当前应用程序的包名；

*自定义一个广播接收者，将备注为“** your receiver name **”的部分，替换为该广播接收者。（自定义广播接收者及其代码，参照下文“自定义推送消息接收者”部分）。

```
<!--Required,SDK 中广播接收者-->
<!--注意：1、category 需填你的 package name -->
<!--注意：2、其余是固定写法，不能改动-->
<receiver android:name="pdx.life.pluginsdk.broadcast.FirstLevelBroadCastReceiver"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
        <action android:name="life.pdx.msg_firstcomming" />
        <action android:name="life.pdx.msg_ordinary" />
        <category android:name="** your package name **" />
    </intent-filter>
</receiver>

<!--Required,用于接收 SDK 转发消息的推送消息-->
<!--注意：1、name 需先自定义推送消息接收者后，在将名字填上-->
<!--注意：2、其余是固定写法，不能改动-->
<receiver android:name="** your receiver name **"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
        <action android:name="life.pdx.msg_secondcomming" />
        <action android:name="life.pdx.msg_ordinary" />
    </intent-filter>
</receiver>
```

自定义推送消息接收者（按需选）

1、区块链消息推送

```
public class MyReceiver extends BroadcastReceiver {
    private String receiverMsg = "";

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        if (intent.getAction().equals(MsgType.MSG_COMMING)) {
            receiverMsg = bundle.getString(MsgType.MSG_CONTENT);
            Log.d("区块链推送消息", receiverMsg);
            //TODO 获取到receiverMsg后，自定义后续操作
            //...
        }
    }
}
```

2、自治身份系统消息

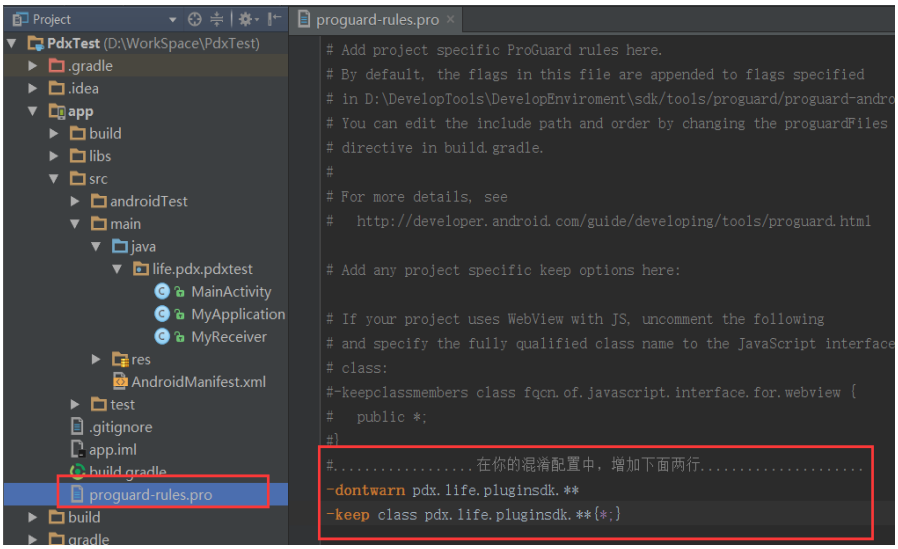
```
public class MyReceiver extends BroadcastReceiver {
    private String receiverMsg = "";

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        if (intent.getAction().equals(MsgType.MSG_COMMING)) {
            receiverMsg = bundle.getString(MsgType.MSG_CONTENT);
            if (receiverMsg.contains("sign")) {
                //自治身份 之 用户之间请求证书
                //TODO 获取到receiverMsg后，自定义后续操作
            }
        } else if (intent.getAction().equals(MsgType.MSG_ORDINARY)) {
            //自治身份之插件请求的证书
            String receiveMsg = bundle.getString(MsgType.MSG_CONTENT);
            try {
                byte[] message = CommonUtil.eccDecryptByBytes(Hex.decode(receiveMsg),
"插件私钥");
                Log.d("自治身份系统消息", new String(message));
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

三、混淆配置

*需要做代码混淆的开发者，请在主 Module 的混淆文件中添加以下配置：

```
-dontwarn pdx.life.pluginsdk.**
-keep class pdx.life.pluginsdk.**{*;}
```



检查确认

*确认所需的权限都已添加。如果必须的权限未添加，日志中会提示错误。
*确认 Required 部分已经正确写入 AndroidManifest。

SDK API 使用

使用提示

该部分是 pdx-plugin Android SDK 中 API 的使用文档。旨在使开发者了解在集成 SDK 后，如何调用 SDK 中的 API，来实现与区块链交互、接收区块链消息推送、自治身份系统。默认读者已经先阅读了上文 SDK 集成部分，并且完成了所需功能的集成。

API 的使用

一、与区块链交互功能

使用 API 前，开发者需要知道，SDK 中区块链的操作有两种：apply（写）、query（查），都是以 Transaction 类的实例（如下文的 transaction）为操作单元。

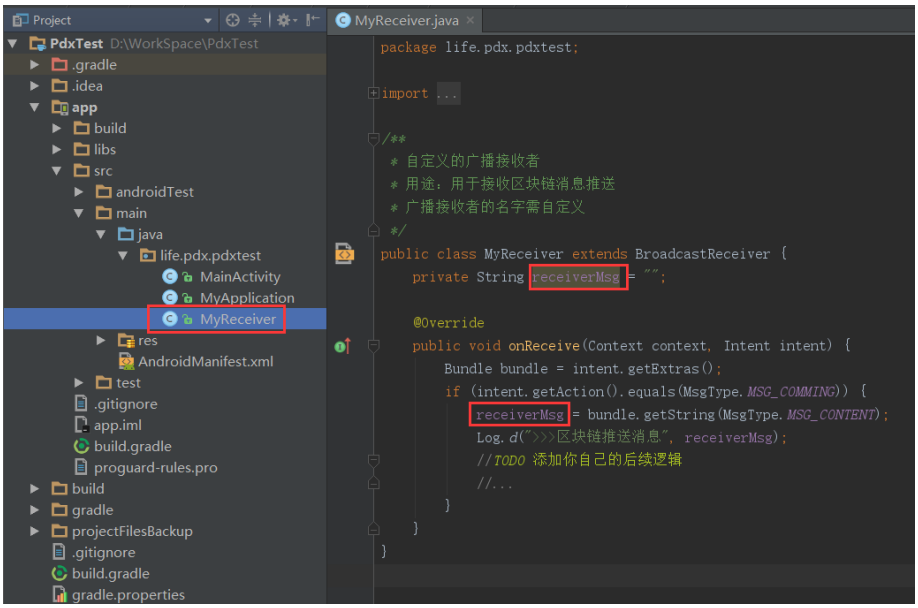
在 API 中，这两种操作的调用方法为：

```
String result1 = BlockChainUses.getInstance().apply(transaction);
List< Transaction> result2 = BlockChainUses.getInstance().query(transaction);
```

注意：上述 Transaction 和 transaction，都是用的 biz.pdxtech.daap.api.contract.Transaction 下的。

二、接收区块链消息推送功能

参考下图（上述的自定义的广播接收者 MyReceiver）：



图中 receiveMsg 即是你接收到的消息推送的内容，根据内容自定义后续处理逻辑即可。

三、自治身份系统功能

自治身份系统是将用户的信息保存在 PDX 移动平台，最大限度保护用户信息。插件可以通过访问自治身份系统，请求用户证书信息，系统会记录请求，同时提示用户授权，授权记录不可更改。

自治身份系统主要提供两大功能：

1、插件请求自治身份系统的证书信息；

2、插件用户之间互相请求对方证书信息；

主要 API

插件请求自治身份系统的证书信息：

1、 插件查询 PDX 支持的证书类型（会有多种类型，按需选一个）
PdxTrustResource.queryPDXSupportMethod()
2、 插件查询步骤 1 中某种证书类型对应的证书公钥（参数为：步骤 1 的类型）
PdxTrustResource.queryPDXSupportMethod(String method)
3、 插件请求某种类型证书信息（参数为：步骤 1 的类型，插件公钥）
PdxAutoIdentity.applyPDXAuto(String method, String pluginKey)
4、 插件校验步骤 3 中请求的证书 Token（若过期，需重新请求）
PdxTrustResource.isTokenExpire(String token)
插件用户之间互相请求对方证书信息：
5、插件用户之间请求对方的某一类型的证书（参数为：证书类型，插件应用公钥）
PdxAutoIdentity.queryOtherEndorsement(String method, String pluginKey)
6、插件校验证书的合法性（参数为：证书信息，步骤 2 中证书公钥）
PdxTrustResource.isEndorsement(String endorse, String endorsementPubKey)

1、查询 PDX 支持的证书类型

```
PdxTrustResource.queryPDXSupportMethod();
```

```
//**
 * 请开发者查询该合约，知道 PDX 支持哪几种认证类型，向 PDX 请求证书消息时需要开发者提交确定的认证类型
{"method":"pdx_bank_auth","publicKey":"02f64e53118333966ab408f24c7af6b71443d52e2fda0dfaff65b39c436eb487e3"}
 *  method:证书类型（请求 PDX 证书的时候需要用到）
 *  publicKey:证书类型对应公钥(插件获取到 PDX 的证书信息需要使用对应的公钥做校验)
 */
```

2、查询 PDX 某一证书类型对应的证书公钥

```
PdxTrustResource.queryPDXSupportMethod(String method);
```

```
**
 * 查询证书公钥
 * @param method 证书类型
 * @return 证书类型对应的公钥
 */
```

3、插件请求 PDX 证书

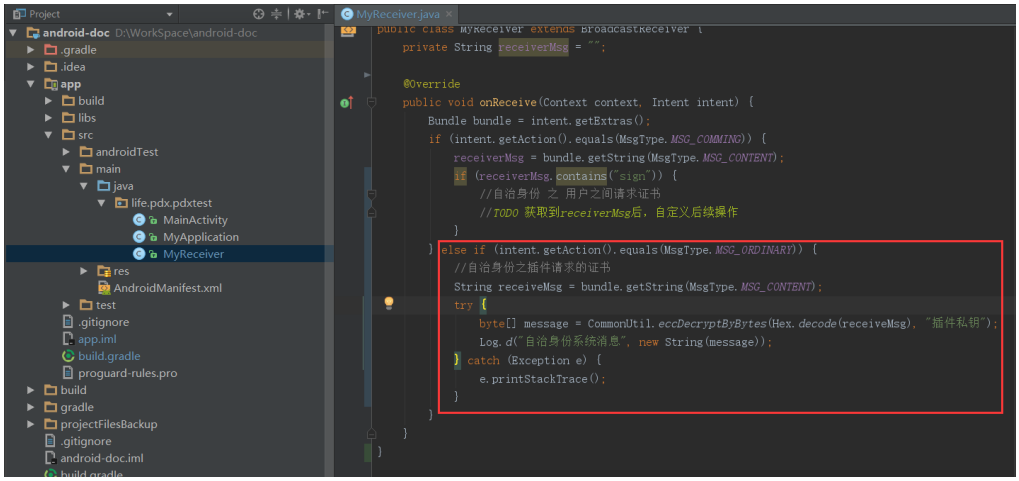
```
PdxAutoIdentity.applyPDXAuto(String method,String pluginKey);
```

```
//**
 * 向 PDX 请求证书
```

```
 * @param method 插件请求的证书类型
 * @param pluginKey 插件的公钥
 */
```

4、插件接收 PDX 授权的证书

在上面已经注册过 Receiver 中添加这一种消息类型,如下图所示:




```
else if (intent.getAction().equals(MsgType.MSG_ORDINARY)) {
    //自治身份之插件请求的证书
    String receiveMsg = bundle.getString(MsgType.MSG_CONTENT);
    try {
        byte[] message = CommonUtil.eccDecryptByBytes(Hex.decode(receiveMsg), "插件私钥");
        Log.d("自治身份系统消息", new String(message));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

5、用户之间请求证书

```
PdxAutoIdentity.queryOtherEndorsement(String method,String pluginKey);
```

```
/**
 * 请求他人证书信息
 * @param method 证书类型
 * @param pluginKey 插件公钥
 */
```

6、校验插件获取的证书

```
PdxAutoIdentity.isEndorsement(String endorse, String endorsementPubKey);
```

```
/**
 * 校验证书
 * @param endorse 证书
 * @param endorsementPubKey  证书公钥
 * @return true:证书通过 false:证书没有通过
 */
```

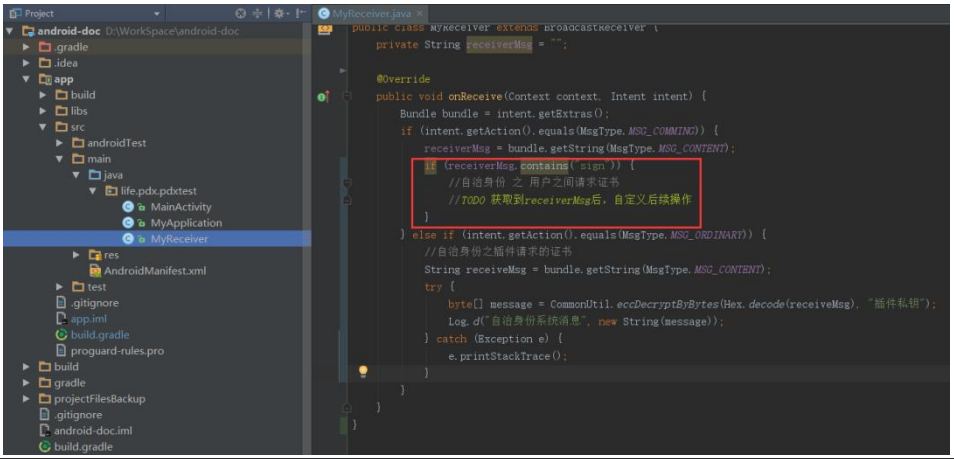
7、校验插件获取的证书

```
PdxTrustResource.isTokenExpire(String token);
```

```
/**
 * 插件校验 token 是否过期
 * @param token 令牌
 * @return true:过期 false:没有过期
 */
```

8、插件接受用户授权的证书

在上面已经注册的 Receiver 中添加这一种消息，如下图所示：




```
if (receiverMsg.contains("sign")) {
    //自治身份 之 用户之间请求证书
    //TODO 获取到 receiverMsg 后，自定义后续操作
}
```