目录

-- mingxiao.li --

2014/12/28

一、回顾 Android 启动过程

见时序图 system_server.vpp

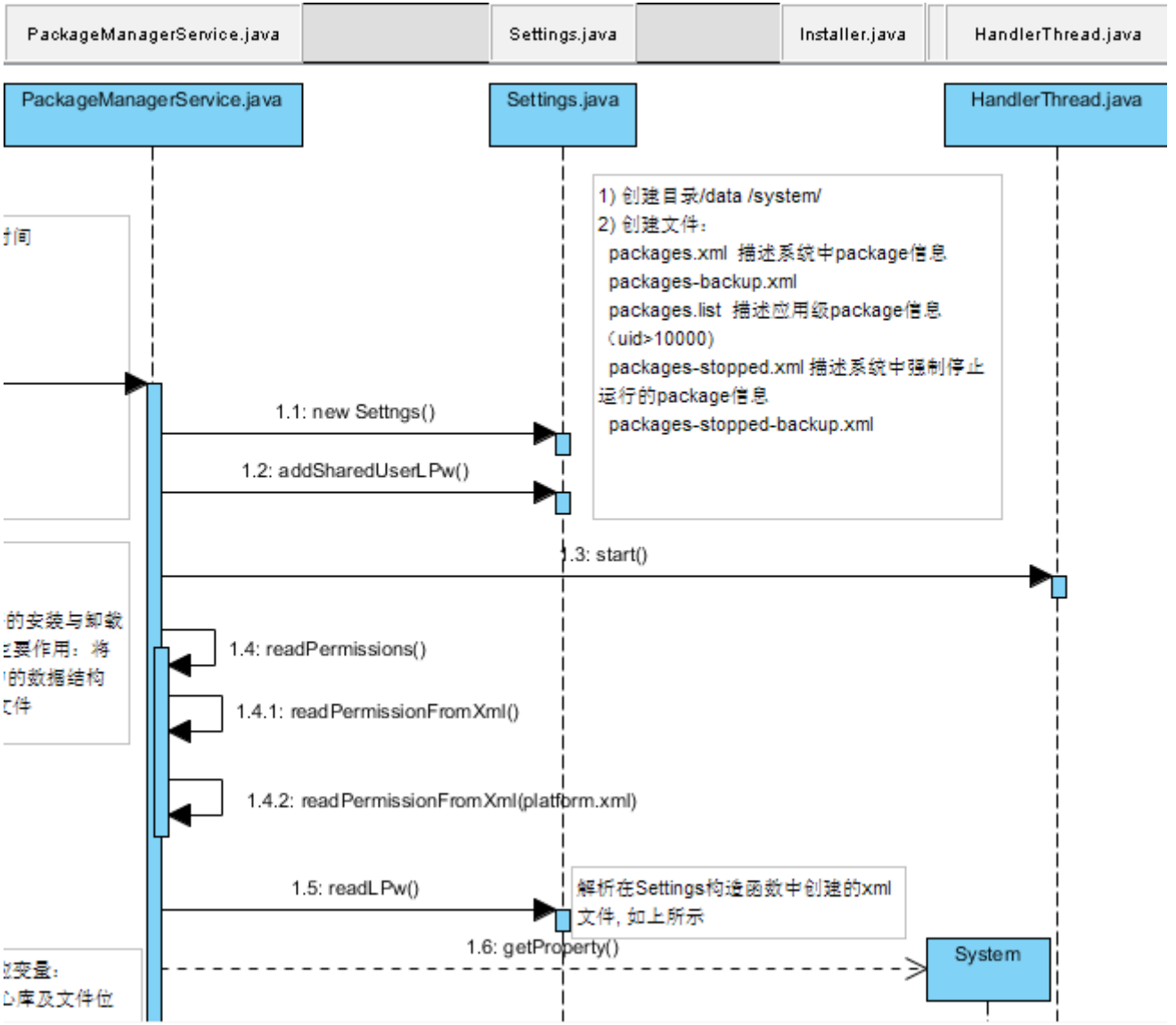
二、 PMS 启动过程（时序图 PMS.vpp ）

1 ）前期准备：扫描并解析指定的 xml 文件

2 ）扫描 APK

3 ）后期整理：信息整理

2.1.1) 前期准备


PackageManagerService.java
Settings.java
Installer.java
HandlerThread.java
PackageManagerService.java
Settings.java
HandlerThread.java
1) 创建目录/data /system/
2) 创建文件：
packages.xml 描述系统中package信息
packages-backup.xml
packages.list 描述应用级package信息
（uid>10000）
packages-stopped.xml 描述系统中强制停止
运行的package信息
packages-stopped-backup.xml
1.1: new Settngs()
1.2: addSharedUserLPw()
1.3: start()
1.4: readPermissions()
1.4.1: readPermissionFromXml()
1.4.2: readPermissionFromXml(platform.xml)
1.5: readLPw()
解析在Settings构造函数中创建的xml
文件，如上所示
1.6: getProperty()
System

2.1.2) 前期准备

1 ） new Settings()

负责管理 Android 系统运行过程中的一些设置信息

2 ） addSharedUserLPw()

添加 uid

3 ） readPermission()

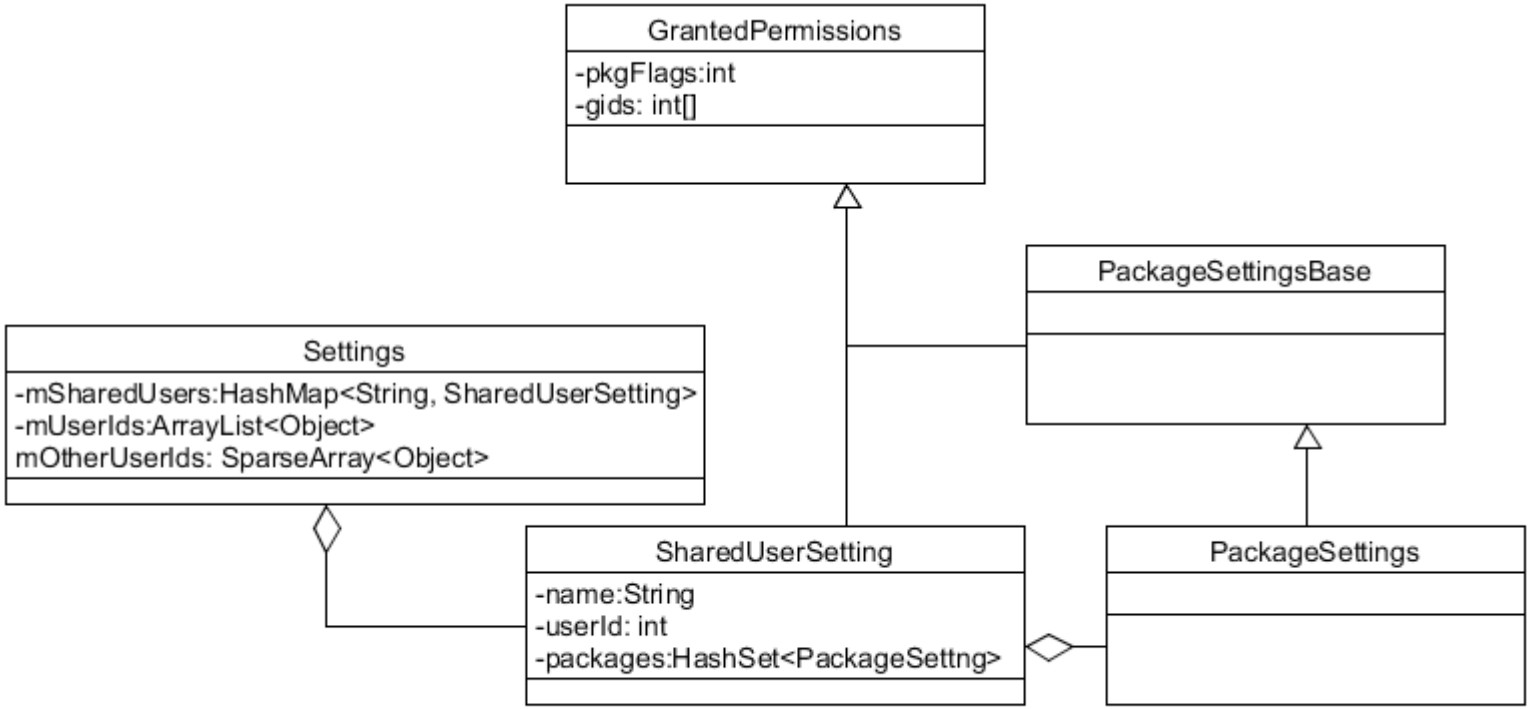
读取对应的 xml 文件，并将 xml 中的标签信息转换成对应的数据结构

4 ） readLPw()

负责解析 new Settings() 中创建的 xml 文件

（ package.xml, package.list, … ... ）

2.1.3) 前期准备： addSharedUserLPw()



1 ）每一个进程都对应一个 UID ，一个或多个 GID ，每个用户 / 用户组对应不同的权限

2 ）两个或多个声明了同一种 sharedUserId 的 APK 可共享彼此的数据，并且可运行在同一个进程中

3 ）通过声明特定的 sharedUserId ，该 APK 所在进程将被赋予指定的 UID

2.1.4) 前期准备： readPermission()

作用：读取对应的 xml 文件，并将 xml 中的标签信息转换成对应的数据结构

资料： 2.2 ）readPermissions() 建立的数据结构

2.1.4) 前期准备： readLPw()

Settings：负责管理 Android 系统运行过程中的一些设置信息

构造函数：

```
mSettingsFilename = new File(mSystemDir, "packages.xml");
mBackupSettingsFilename = new File(mSystemDir, "packages-backup.xml");
mPackageListFilename = new File(mSystemDir, "packages.list");
FileUtils.setPermissions(mPackageListFilename, 0660, SYSTEM_UID,
                         PACKAGE_INFO_GID);
mStoppedPackagesFilename = new File(mSystemDir, "packages-stopped.xml");
mBackupStoppedPackagesFilename = new File(mSystemDir,
                         "packages-stopped-backup.xml");
```

readLPw() 负责解析上述创建的 xml 文件

data/system/packages.xml：描述系统中的 package 信息

→ 存储于 SharedUserSetting.packages

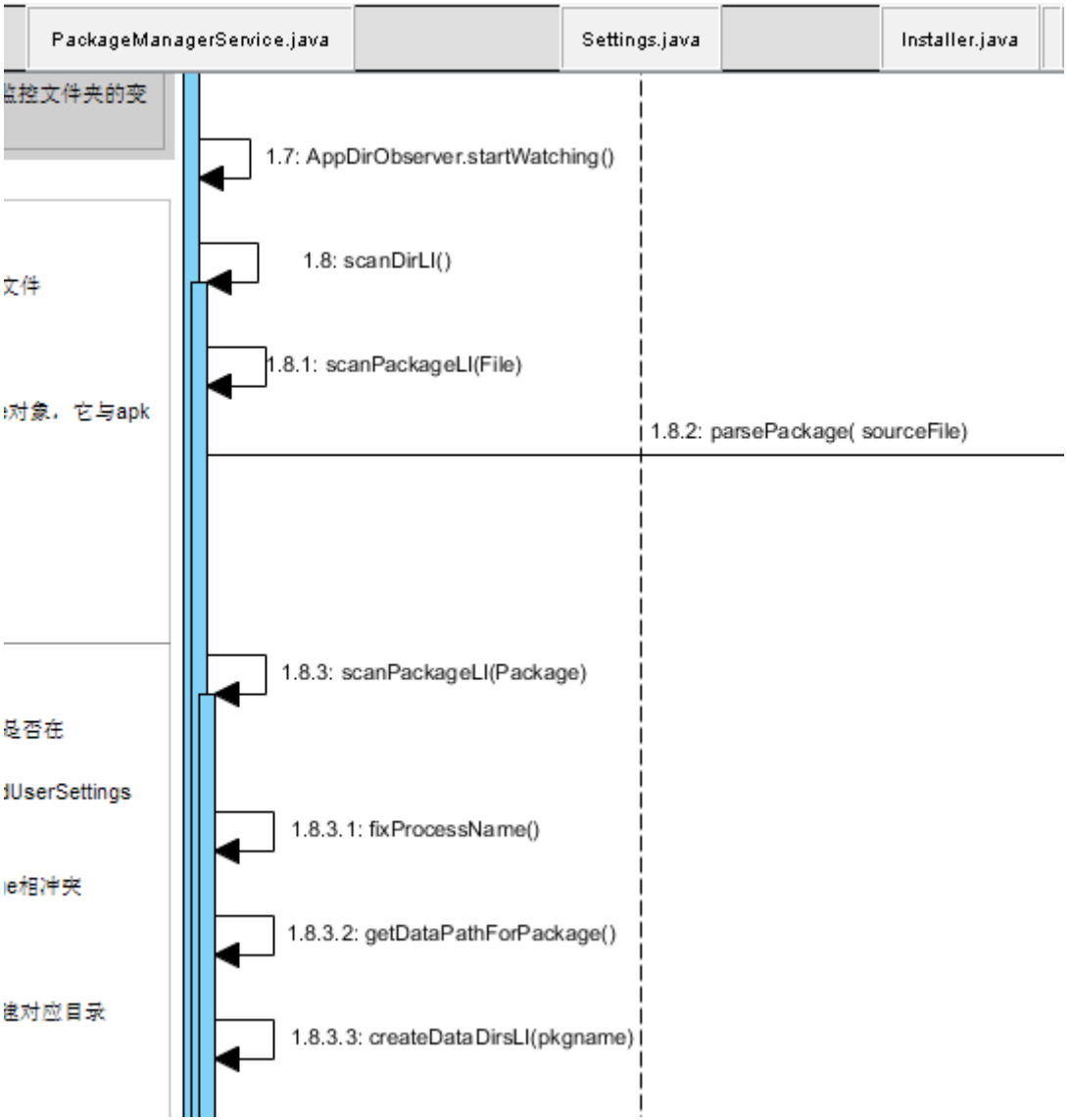
data/system/packages-backup.xml：对应的临时文件

data/system/packages.list：描述应用级 package 信息

data/system/packages-stopped.xml：描述系统中强制停止运行的 package 信息

data/system/packages-stopped-backp.xml：对应的临时文件

2.2.1 ）扫描 APK


PackageManagerService.java
Settings.java
Installer.java
1.7: AppDirObserver.startWatching()
1.8: scanDirLI()
1.8.1: scanPackageLI(File)
1.8.2: parsePackage(sourceFile)
1.8.3: scanPackageLI(Package)
1.8.3.1: fixProcessName()
1.8.3.2: getDataPathForPackage()
1.8.3.3: createDataDirsLI(pkgname)

2.2.2）扫描 APK： scanPackageLI(file)

作用：返回一个 Package 对象，与 apk 文件结构相对应

1）解析 apk 文件获取 package 对象

2）查看是否更新 package 信息

3）收集签名信息

4）解析资源与 app 在不同目录的 app

5）将解析到的 package 信息添加到 PMS 数据结构中

2.2.2 ）扫描 APK ： parsePackage()

PMS 主要数据结构：

mActivities: ActivityIntentResolver 用于保存所有 Activity 信息

mServices: ServiceIntentResolver 用于保存所有 Service 信息

mProvider: HashMap<String, PackageParser.Provider>

key ： Provider 路径， value ： Provider 信息

mReceiver: ActivityIntentResolver 用于保存所有 BroadcastReceiver 信息

mPackages: HashMap<String, PackageParser.Package>

...

2.2.2）扫描 APK： scanPackageLI(package)

1）单独解析 framework-res.apk

2）处理 pkgSettings

3）检查签名信息

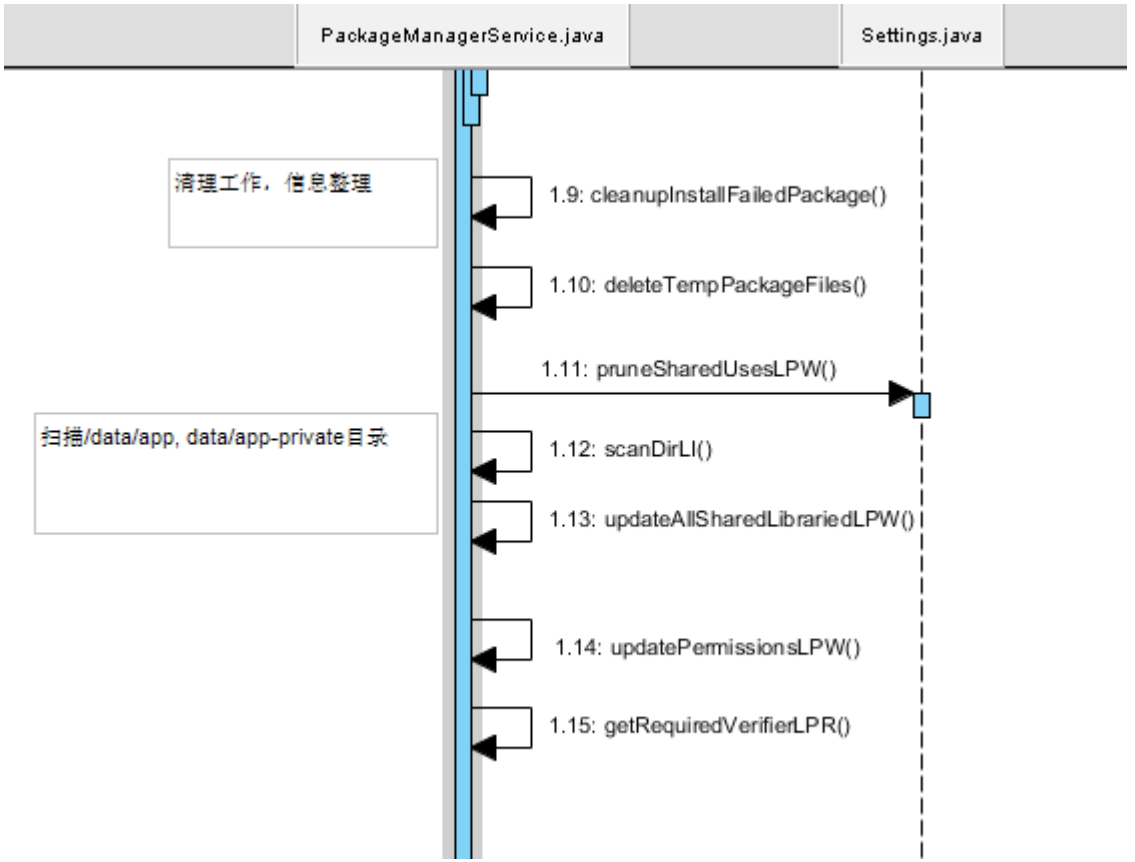
4）检查 ContentProvider 是否项冲突

5）确定对应 app 的进程名，一般为包名

6）获取 app 安装路径，拷贝 jar 包以及库文件

7）将 package 中的四大组建信息添加到 PMS 内部数据结构中，便于统一管理

2.3 ）后期整理


PackageManagerService.java
Settings.java
清理工作，信息整理
1.9: cleanupInstallFailedPackage()
1.10: deleteTempPackageFiles()
1.11: pruneSharedUsesLPW()
扫描/data/app, data/app-private目录
1.12: scanDirLI()
1.13: updateAllSharedLibrariedLPW()
1.14: updatePermissionsLPW()
1.15: getRequiredVerifierLPR()

三、 PackageInstaller

- 获取 Apk 的相关信息 (PackageInstallerActivity)

  PackageParser.Package parsed =

  PackageUtil.getPackageInfo(sourceFile);

  packageParser.**parsePackage**(sourceFile, ...)

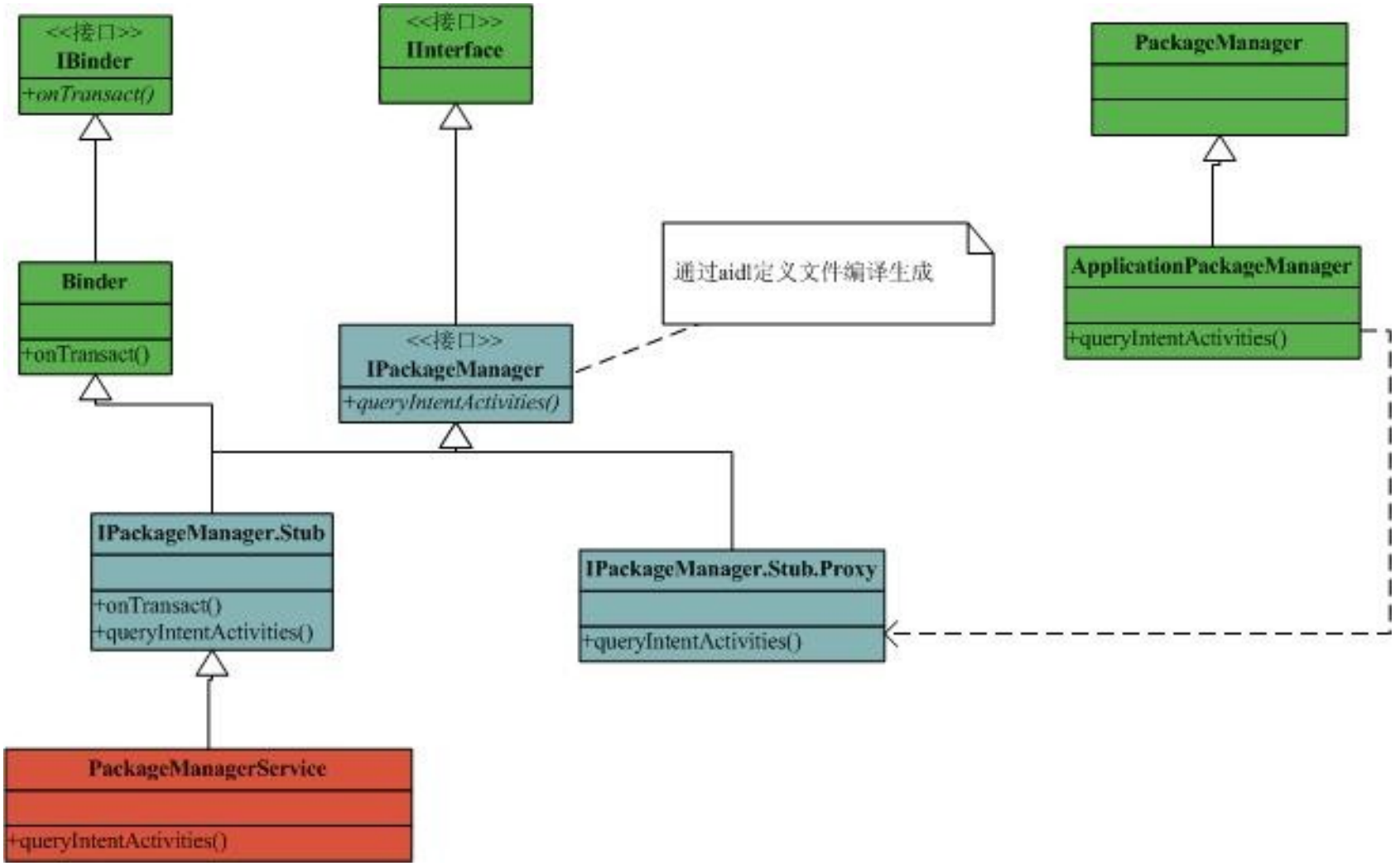
- 安装 (InstallAppProgress)

  pm.installExistingPackage(...);

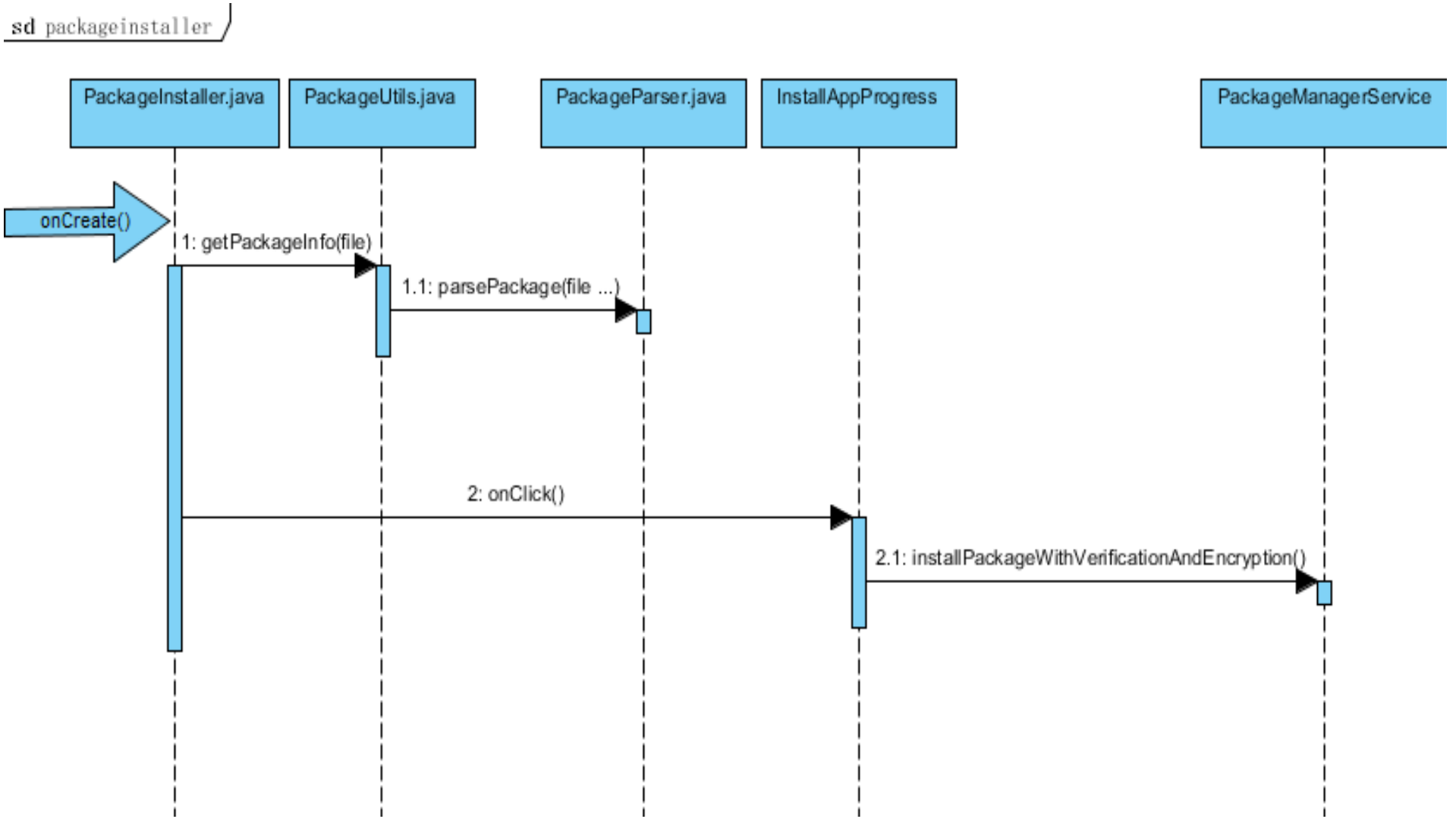
  pm.**installPackageWithVerificationAndEncryption**(
  mPackageURI ...);

3.1 ） PackageManager 关系类图

（资料 3.1 ，图片来源于网络）

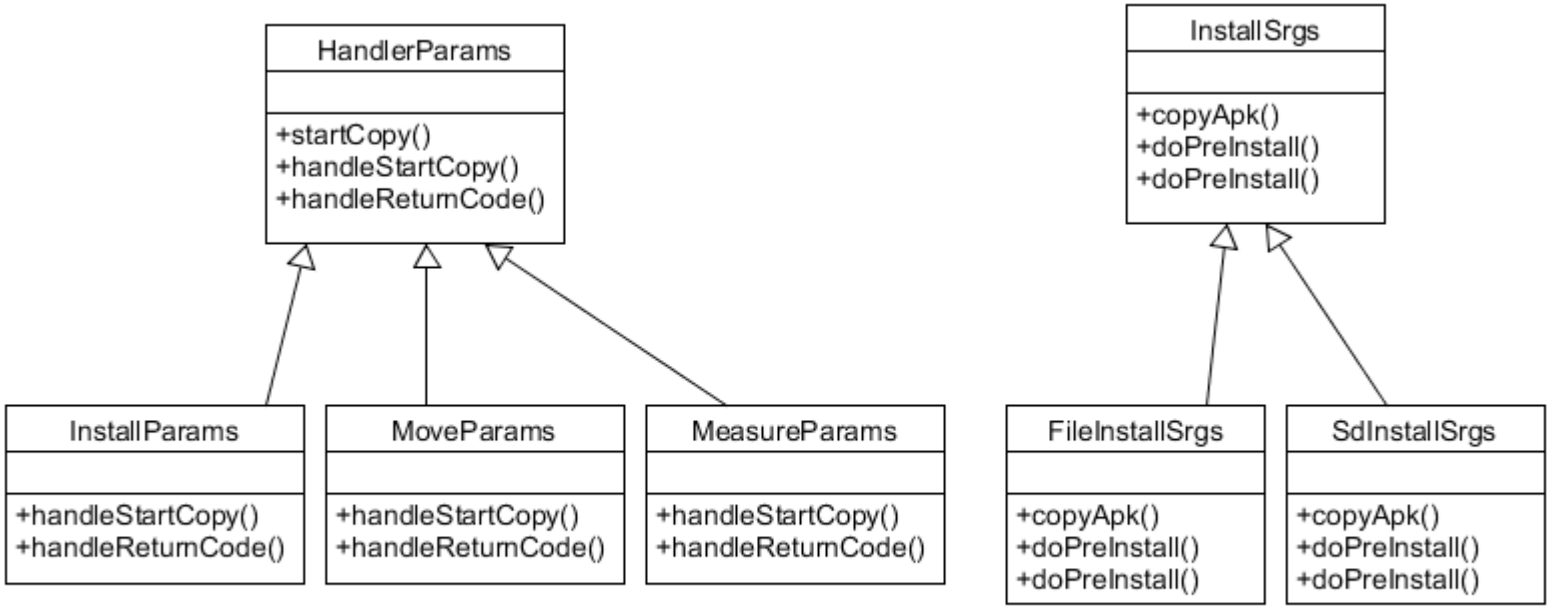


3.2 ） PackageInstaller 主要流程



3.3 ） HandlerParams 类图结构

（ 资料 3.3 ）



3.4 ） InstallParams.handleStartCopy()

1 ）检查安装位置

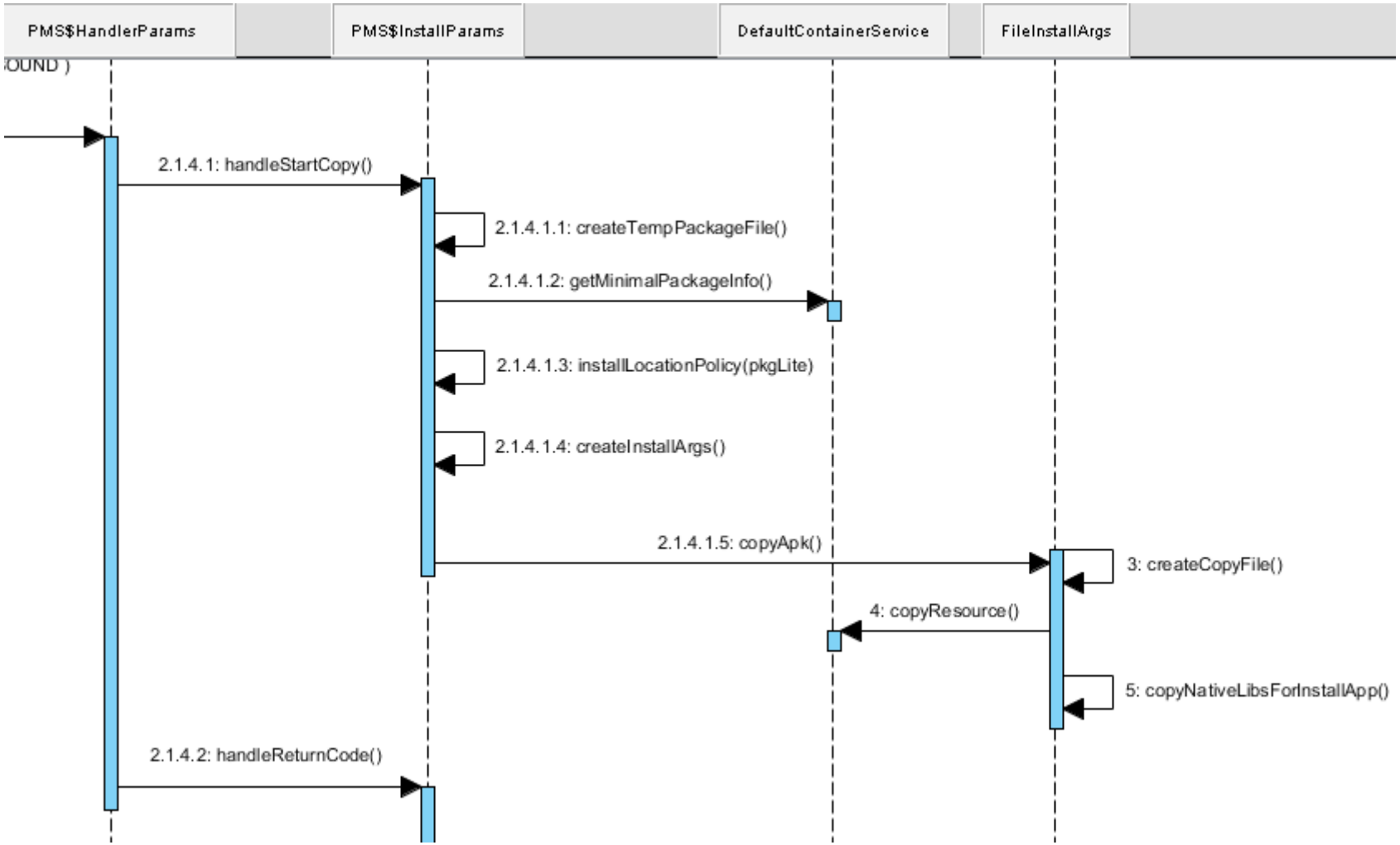
2 ）调用 installLocationPolicy 检查推荐的安装路径

3 ）根据安装位置创建对应的参数 :createInstallArgs()

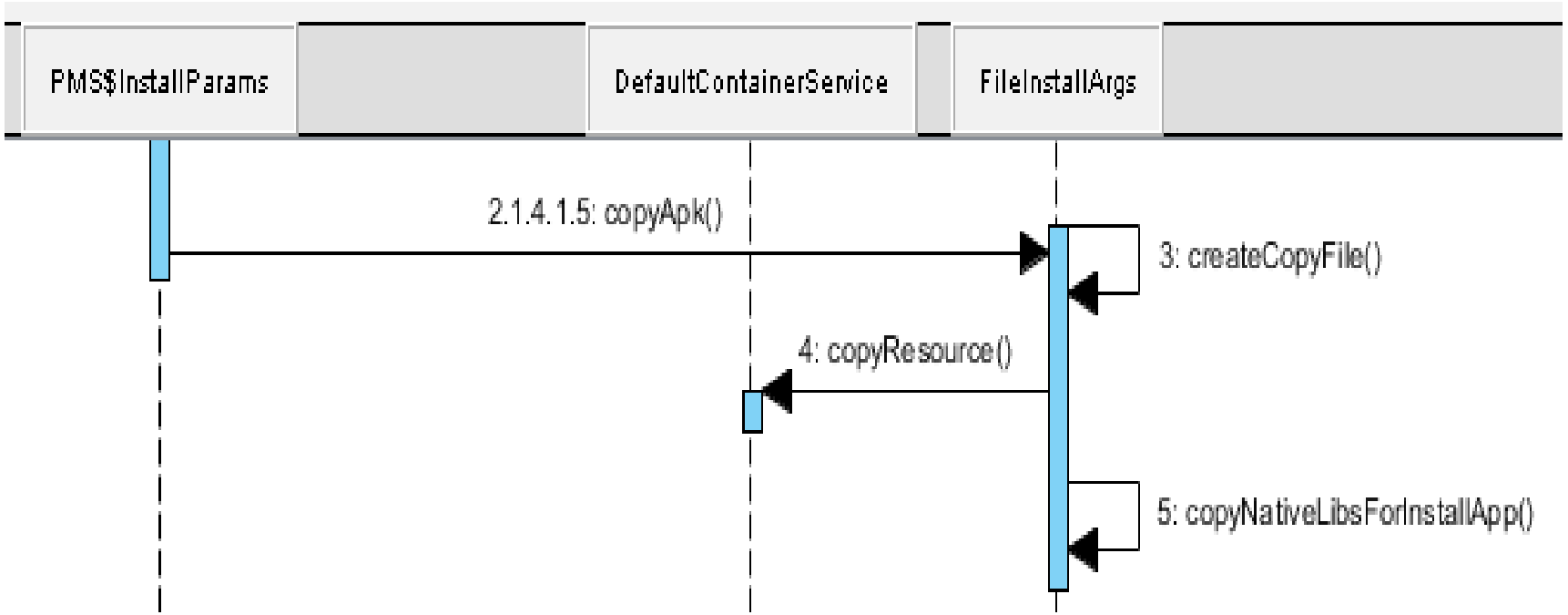
4 ）安装之前，对 APK 进行必要的检查

5 ）调用 InstallArgs 的 copyApk() 正式安装 app （此处为 FileInstallArgs ）

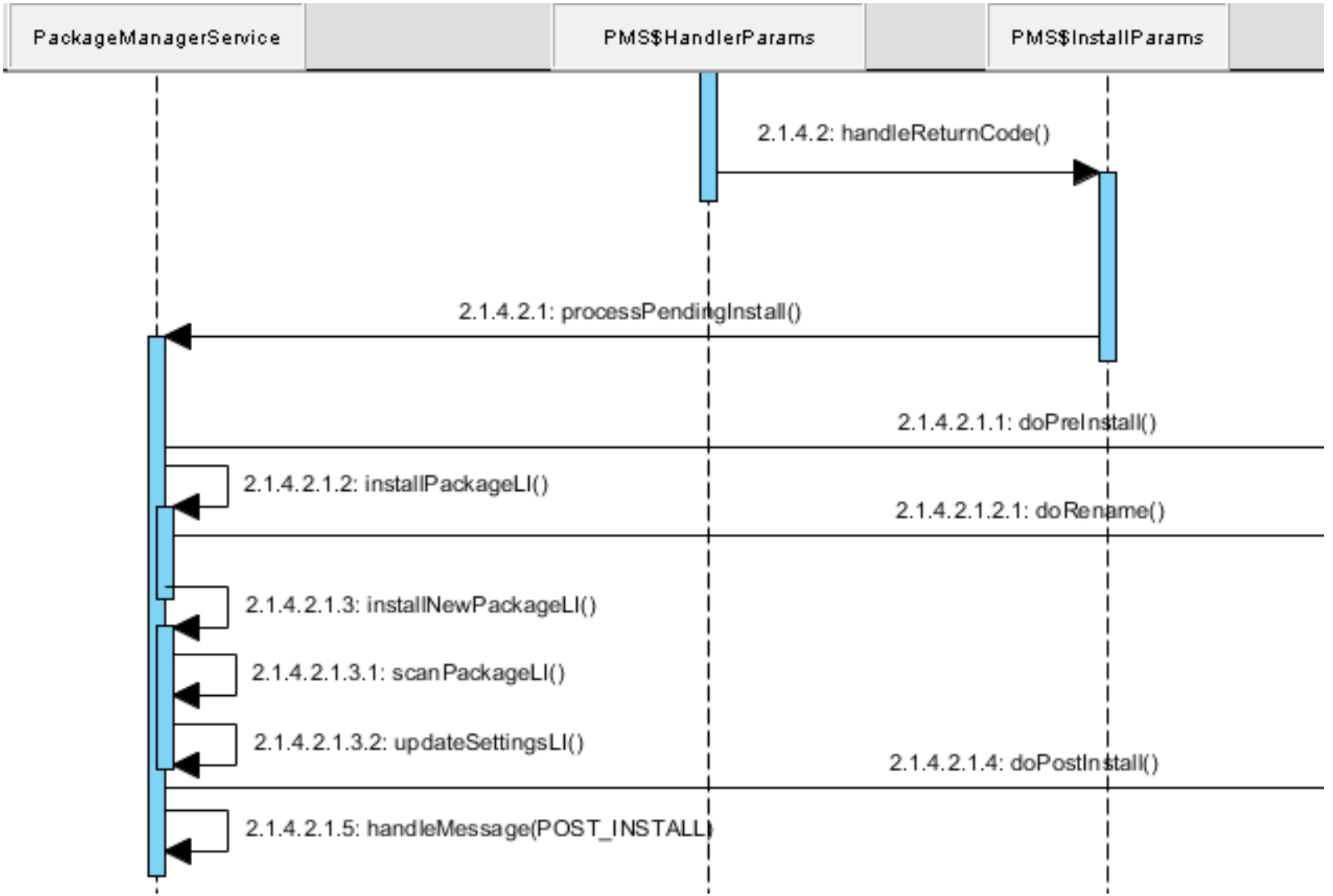
3.4.1 ）时序图 handleStartCopy()



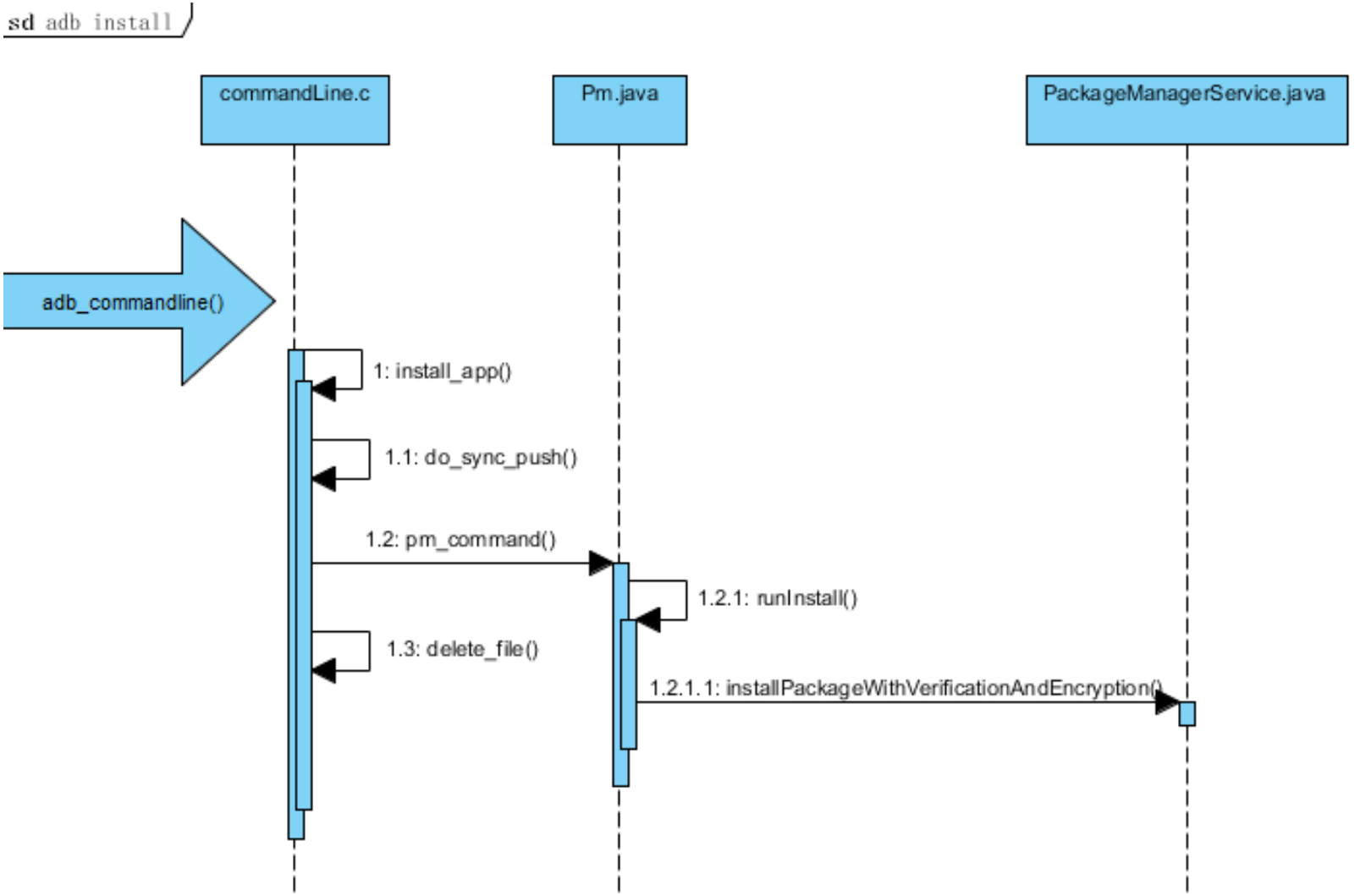
3.4.2 ） InstallArgs.copyApk()



3.5 ） InstallParams.handleReturnCode()



四、 ADB Install



后续步骤类似 PackageInstaller

五、总结

- 资料： PackageManagerService 作用
- 参考资料：《深入理解 Android 卷 II 》 -- 邓凡平