SharedUserSetting 与 PackageSetting的关系

User ID 是指用户ID, 类似于LINUX下的多用户，不同的用户ID是不能互相访问的，对比linux下不同的用户是不能访问其他用户的。

At install time, Android gives each package a distinct Linux user ID. The identity remains constant for the duration of the package's life on that device. On a different device, the same package may have a different UID; what matters is that each package has a distinct UID on a given device.

Because security enforcement happens at the process level, the code of any two packages cannot normally run in the same process, since they need to run as different Linux users. You can use the [sharedUserId](http://developer.android.com/reference/android/R.attr.html#sharedUserId) attribute in the AndroidManifest.xml's [manifest](http://developer.android.com/reference/android/R.styleable.html#AndroidManifest) tag of each package to have them assigned the same user ID. By doing this, for purposes of security the two packages are then treated as being the same application, with the same user ID and file permissions. Note that in order to retain security, only two applications signed with the same signature (and requesting the same sharedUserId) will be given the same user ID.

Any data stored by an application will be assigned that application's user ID, and not normally accessible to other packages.

From <http://developer.android.com/intl/ja/guide/topics/security/permissions.html>

Package拥有一个特定的User ID, 两个package必须拥有相同的User ID和签名才会互相访问对方的数据

AndroidManifest.xml 里的android:sharedUserId

The name of a Linux user ID that will be shared with other applications. By default, Android assigns each application its own unique user ID. However, if this attribute is set to the same value for two or more applications, they will all share the same ID — provided that they are also signed by the same certificate. Application with the same user ID can access each other's data and, if desired, run in the same process. 通过 android:process 来指定

一、 加入系统默认的几个shared user id

mSettings.addSharedUserLPw("android.uid.system", Process.***SYSTEM\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);  
mSettings.addSharedUserLPw("android.uid.phone", ***RADIO\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);  
mSettings.addSharedUserLPw("android.uid.log", ***LOG\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);  
mSettings.addSharedUserLPw("android.uid.nfc", ***NFC\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);  
mSettings.addSharedUserLPw("android.uid.bluetooth", ***BLUETOOTH\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);  
mSettings.addSharedUserLPw("android.uid.shell", ***SHELL\_UID***,  
 ApplicationInfo.***FLAG\_SYSTEM***, ApplicationInfo.***PRIVATE\_FLAG\_PRIVILEGED***);

二、 读取 packages.xml 里的 <shared-user> section

mSettings.readLPw -> readSharedUserLPw(parser);

su = addSharedUserLPw() //加入到Settings的mSharedUsers/mUserIds/mOtherUsers

readInstallPermissionsLPr(parser, su.getPermissionsState())

//读取上次系统里对于关shared user的权限状态，如果grant了，那么在这里直接grant权限

//读取 /data/system/users/x/runtime-permissions.xml 里的 <shared-user>,

与packages.xml里相似

mRuntimePermissionsPersistence.readStateForUserSyncLPr(user.id);

读取 packages.xml 里的<shared-user> 节，

如:

<shared-user name="android.uid.shared" userId="10003">

<sigs count="1">

<cert index="18" />

</sigs>

<perms>

<item name="android.permission.MANAGE\_USERS" granted="true" flags="0" /> **//grant给该shared uid的权限**

<item name="android.permission.INTERACT\_ACROSS\_USERS" granted="true" flags="0" />

<item name="android.permission.BIND\_DIRECTORY\_SEARCH" granted="true" flags="0" />

<item name="android.permission.READ\_SYNC\_SETTINGS" granted="true" flags="0" />

<item name="android.permission.SEND\_CALL\_LOG\_CHANGE" granted="true" flags="0" />

<item name="android.permission.UPDATE\_APP\_OPS\_STATS" granted="true" flags="0" />

</perms>

</shared-user>

<shared-user name="android.uid.system" userId="1000">

<sigs count="1">

<cert index="0" />

</sigs>

<perms>

<item name="android.permission.WRITE\_SETTINGS" granted="true" flags="0" />

<item name="android.permission.USE\_CREDENTIALS" granted="true" flags="0" />

<item name="android.permission.MODIFY\_AUDIO\_SETTINGS" granted="true" flags="0" />

<item name="android.permission.INSTALL\_LOCATION\_PROVIDER" granted="true" flags="0" />

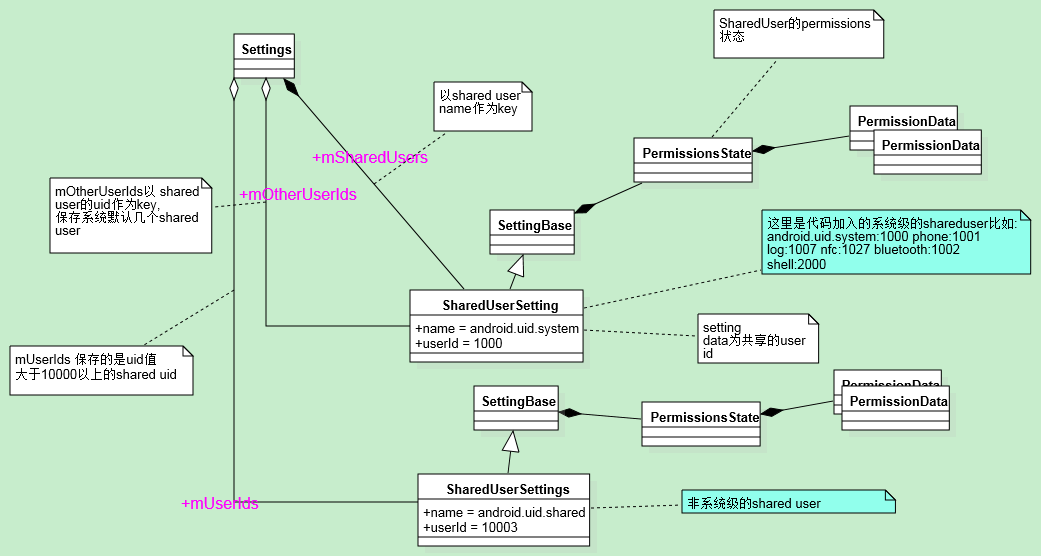
<item name="android.permission.MANAGE\_ACCOUNTS" granted="true" flags="0" />

<item name="android.permission.SYSTEM\_ALERT\_WINDOW" granted="true" flags="0" />

<item name="android.permission.GET\_TOP\_ACTIVITY\_INFO" granted="true" flags="0" />

…

</shared-user>



三、扫描系统apk s, 解析AndroidManifest.xml里的manifest中的android:sharedUserId, 并与PackageSetting建立联系

**scanDirLI –> scanPackageLI ->**

===> pp.parsePackage -> parseClusterPackage -> parseBaseApk -> parseBaseApk (Resources) ->

//开始解析 AndroidManifest.xml里manifest 的android:sharedUserId=””, 保存到package里的mSharedUserId下

String str = sa.getNonConfigurationString(  
 com.android.internal.R.styleable.***AndroidManifest\_sharedUserId***, 0);

PackageParser.Package pkg.**mSharedUserId** = str.intern();

===> scanPackageLI(pkg, …)-> scanPackageDirtyLI

//获得一个SharedUserSetting对象，如果没有就创建一个并加入到mSharedUsers/mUserIds里

**if** (pkg.mSharedUserId != **null**) {  
SharedUserSetting = mSettings.getSharedUserLPw(pkg.mSharedUserId, 0, 0, **true**);

}

//仅仅获得PackageSetting,不会加入到任何链表里。如果之前没有，生成一个新对象，这种情况跟packages.xml是否存在相关

pkgSetting = mSettings.getPackageLPw(pkg, origPackage, realName, suid, …)

PackageSetting getPackageLPw(PackageParser.Package pkg, PackageSetting origPackage, String realName, SharedUserSetting sharedUser, …)

//在这里假设之前是没有PackageSetting的

p = **new** PackageSetting(name, realName, …);  
p.sharedUser = sharedUser;

//将SharedUserSetting保存到PackageSetting里作为Package的一个属性

**if** (sharedUser != **null**) {  
 p.appId = sharedUser.userId;

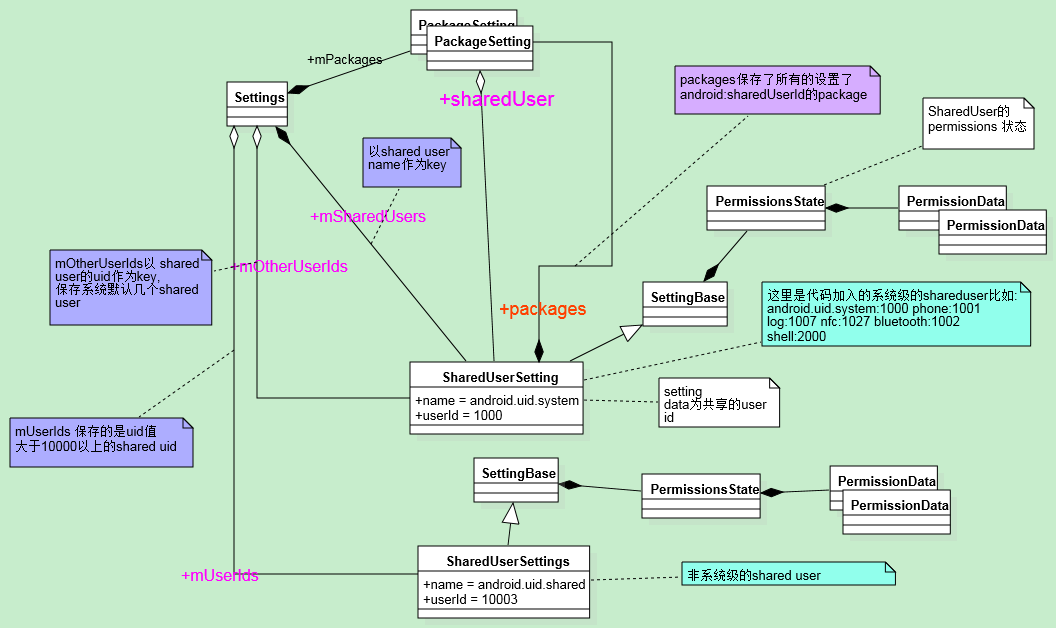
//如果package指定了sharedUserId的，那么appId将会被设置为与sharedUserId相同。  
} **else** {  
 *// Assign new user id* p.appId = newUserIdLPw(p); //如果没有，那么自己生成一个新的userId, 从10000开始自加  
}

**if** (add) { //在这里为false, 仅仅创建setting  
 *// Finish adding new package by adding it and updating shared  
 // user preferences* addPackageSettingLPw(p, name, sharedUser);  
}

mSettings.insertPackageSettingLPw(pkgSetting, pkg); -> addPackageSettingLPw(p, pkg.packageName, p.sharedUser);

p.pkg = pkg;

sharedUser.addPackage(p); //加入到SharedUserSetting里的packages里，这样就把SharedUserSetting就track了package.  
p.sharedUser = sharedUser;  
p.appId = sharedUser.userId;



Shared User相关的permission

见PKMS\_permissioins.docx

即 “由 getPermissionsState可以看出，在获得 PermissionsState的时候先找sharedUser，即Shared User Id的PermissionsState, 如果没有设置shared uid才用package自己的，所以可以看出来， UID的PermissionsState是所有在AndroidManifest.xml设置了sharedUserId里的permission的一个合集，且他们permission也是一样的。”

结论:

1. 如果在AndroidManifest.xml里没有设置sharedUserId, 那么系统在第一次开机解析apk会递增的给package赋值uniq UID, 且这些UID是从10000开始的，然后将这些UID值会写到packages.xml里，这样下次系统启动后会根据在packages.xml里的userId来给package的uid赋值。

2. 系统中默认定义了几个系统级的UID, 如system/shell/phone …, 这些都是shared uid的, 且值都是固定的且少于10000，具体的uid的值可以从 /data/system/packages.xml里通过<shared-user里的 userId查找。

3. 设置了sharedUserId的package, 会继承shared-user里所有的权限，且会用自己的权限来扩充shared uid的权限，它的uid在packages.xml里通过sharedUserId来表示。