**Android Framework相关的permissions**

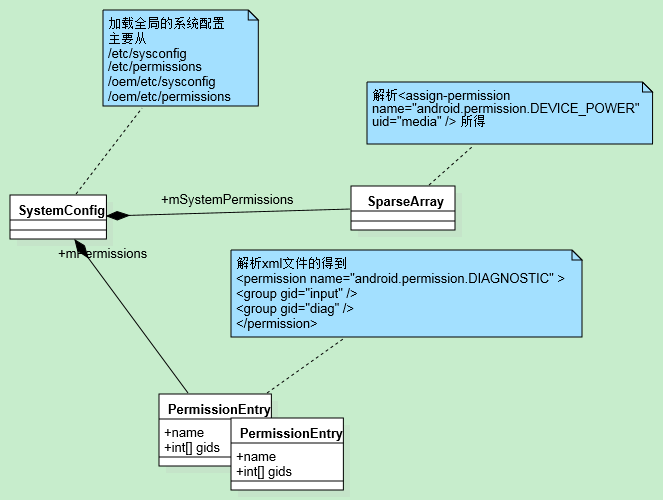
“Bobby Wang bobwang@nvidia.com”

**一、系统配置的权限**

SystemConfig systemConfig = SystemConfig.getInstance();

Android从系统配置文件中读取自定义的权限，主要是从 /etc/sysconfig /etc/permissions /oem/etc/sysconfig /oem/etc/permissions里xml文件

读出来的permission相关的UML图为



mGlobalGids = systemConfig.getGlobalGids();  
mSystemPermissions = systemConfig.getSystemPermissions();  
mAvailableFeatures = systemConfig.getAvailableFeatures();

// Propagate permission configuration in to package manager.  
ArrayMap<String, SystemConfig.PermissionEntry> permConfig  
 = systemConfig.getPermissions();  
**for** (**int** i=0; i<permConfig.size(); i++) {  
 SystemConfig.PermissionEntry perm = permConfig.valueAt(i);  
 BasePermission bp = mSettings.mPermissions.get(perm.name);  
 **if** (bp == **null**) {  
 bp = **new** BasePermission(perm.name, "android", BasePermission.**TYPE\_BUILTIN**);  
 mSettings.mPermissions.put(perm.name, bp);  
 }  
 **if** (perm.gids != **null**) {  
 bp.setGids(perm.gids, perm.perUser);  
 }  
}

以上这些代码片段主要目的是将SystemConfig里解析出来的值取出来赋值给PackageManagerService里的变量或mSettings，在这个阶段解析出来的permission的sourcePackage全部命名为 android.

**二、读取上次系统里的package.xml**

Part1, mSettings.readLPw(**this, …)** //解析/data/system/package.xml

**1). if** (tagName.equals("permissions")) {   
 readPermissionsLPw(mPermissions, parser);

//在package.xml里permissions放在前面，所以这里先读的是permissions section, permissions字段,保存到mSettings.mPermissions，permissions是所有apk自定义的permissions

**2). if** (tagName.equals("package")) {  
 **readPackageLPw**(parser); //解析 package section

**2.1) readPackageLPw**(XmlPullParser parser)

packageSetting = addPackageLPw(name.intern(), realName, …)

//生成该package对应的 PackageSetting, 然后根据xml里值进行填充

**readInstallPermissionsLPr**(parser, packageSetting.getPermissionsState());

//读取安装权限

**2.2) void** readInstallPermissionsLPr(XmlPullParser parser,  
 PermissionsState permissionsState)

//解析package section里的 <perms>中当前package里所有的权限

String grantedStr = parser.getAttributeValue(**null**, **ATTR\_GRANTED**);  
**final boolean** granted = grantedStr == **null** || Boolean.parseBoolean(grantedStr);

**2.3) if** (granted) {  
 permissionsState.grantInstallPermission(bp)

grantInstallPermission();

//grant安装权限，这些安装权限是apk在安装时自动grant的，都是normal的等级，不是dangeous权限。

**//该函数的主要作用是**

1. 生成permission对应的PermissionData，并用加入到PermissionsState mPermissions里

2. 对用户id,grant权限，即生成PermissionState对象，并用mUserStates来track.

Part2, 从/data/system/users/x/runtime-permissions.xml 读取runtime权限, 并grant  
 **for** (UserInfo user : users) {  
 mRuntimePermissionsPersistence.readStateForUserSyncLPr(user.id);  
 }

readStateForUserSyncLPr->parseRuntimePermissionsLPr

1). parseRuntimePermissionsLPr()->parsePermissionsLPr

//解析 <pkg> 下的每一个item得到每一个权限，然后进行grant

**if** (granted) {  
 permissionsState.grantRuntimePermission(bp, userId); //grant runtime权限  
 permissionsState.updatePermissionFlags(bp, userId,  
 PackageManager.**MASK\_PERMISSION\_FLAGS**, flags);

}

如下面这个package版本，摘自/data/system/package.xml

<package name="com.android.tv.settings" codePath="/system/priv-app/TvSettings" nativeLibraryPath="/system/priv-app/TvSettings/lib" primaryCpuAbi="arm64-v8a" publicFlags="944291397" privateFlags="8" ft="1539c6151e8" it="1539c6151e8" ut="1539c6151e8" version="1" sharedUserId="1000">

<sigs count="1">

<cert index="0" />

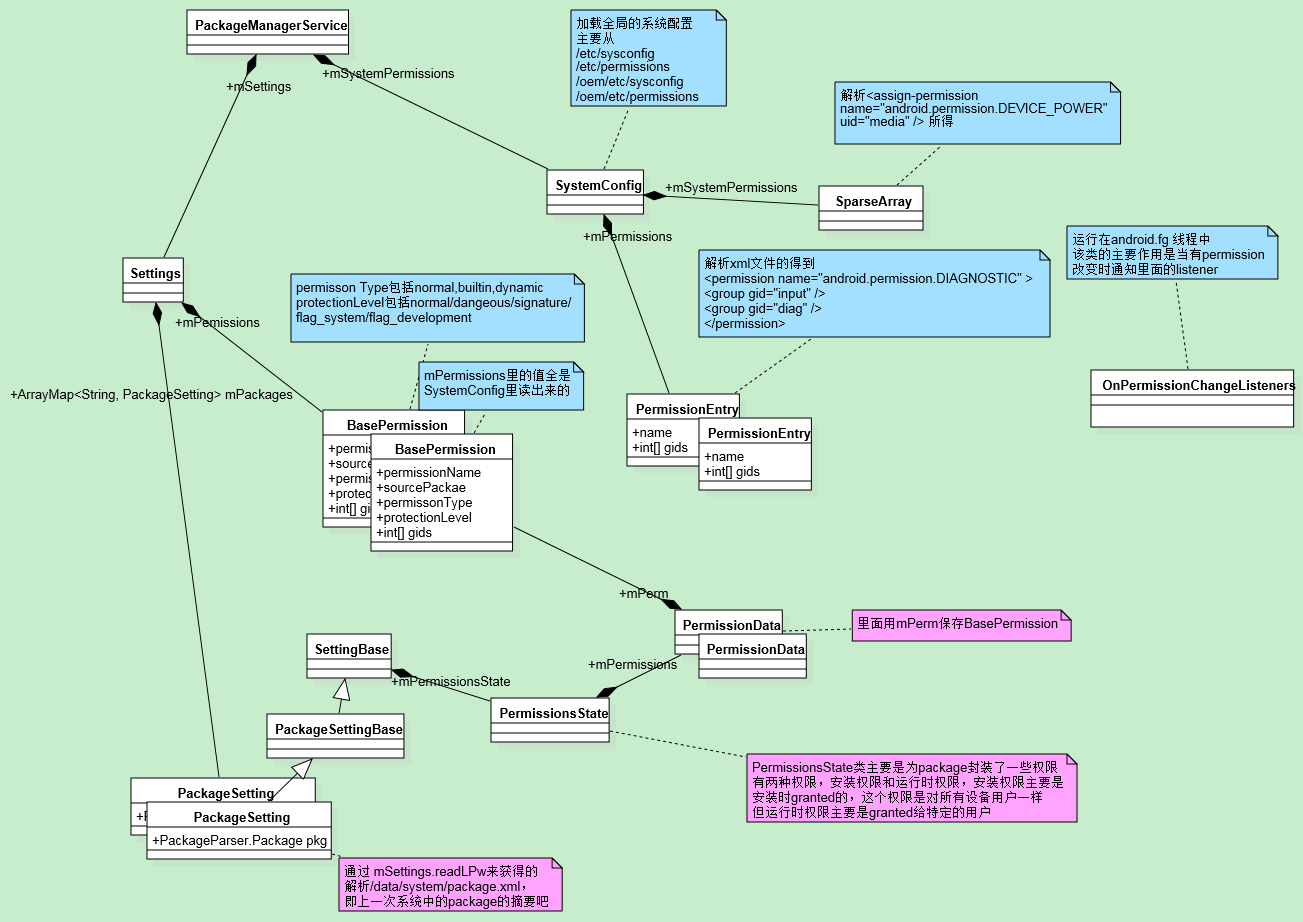
</sigs>

<perms>

<item name="android.permission.WRITE\_SETTINGS" 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" />



注意：如果是系统第一次开机的时候，系统里是没有package.xml的，那么将不会生成package对应的PackageSetting, 在这种情况下，PackageSetting在扫描apk文件时进行生成.

**三、扫描系统apk文件，解析apk文件**

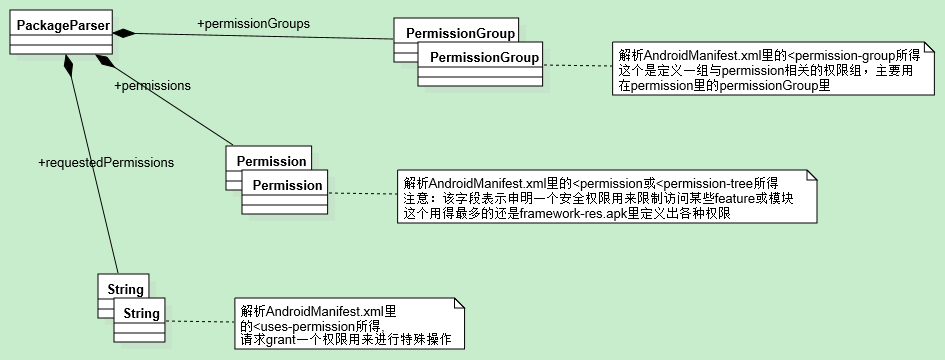
scanDirLI -> scanPackageLI(file, …) ->

1. pkg = pp.parsePackage(scanFile, parseFlags); //解析apk文件

parseClusterPackage

pkg = parseBaseApk(baseApk, assets, flags);

PackageParser.Package pkg = **parseBaseApk**(res, …) //解析apk中的AndroidManifest.xml



其中PackageParser.Package代表一个解析出来的apk, 而PackageParser指的是一个解析类，

2. scanPackageLI(pkg, xxx) -> scanPackageDirtyLI(pkg, …) 该步骤

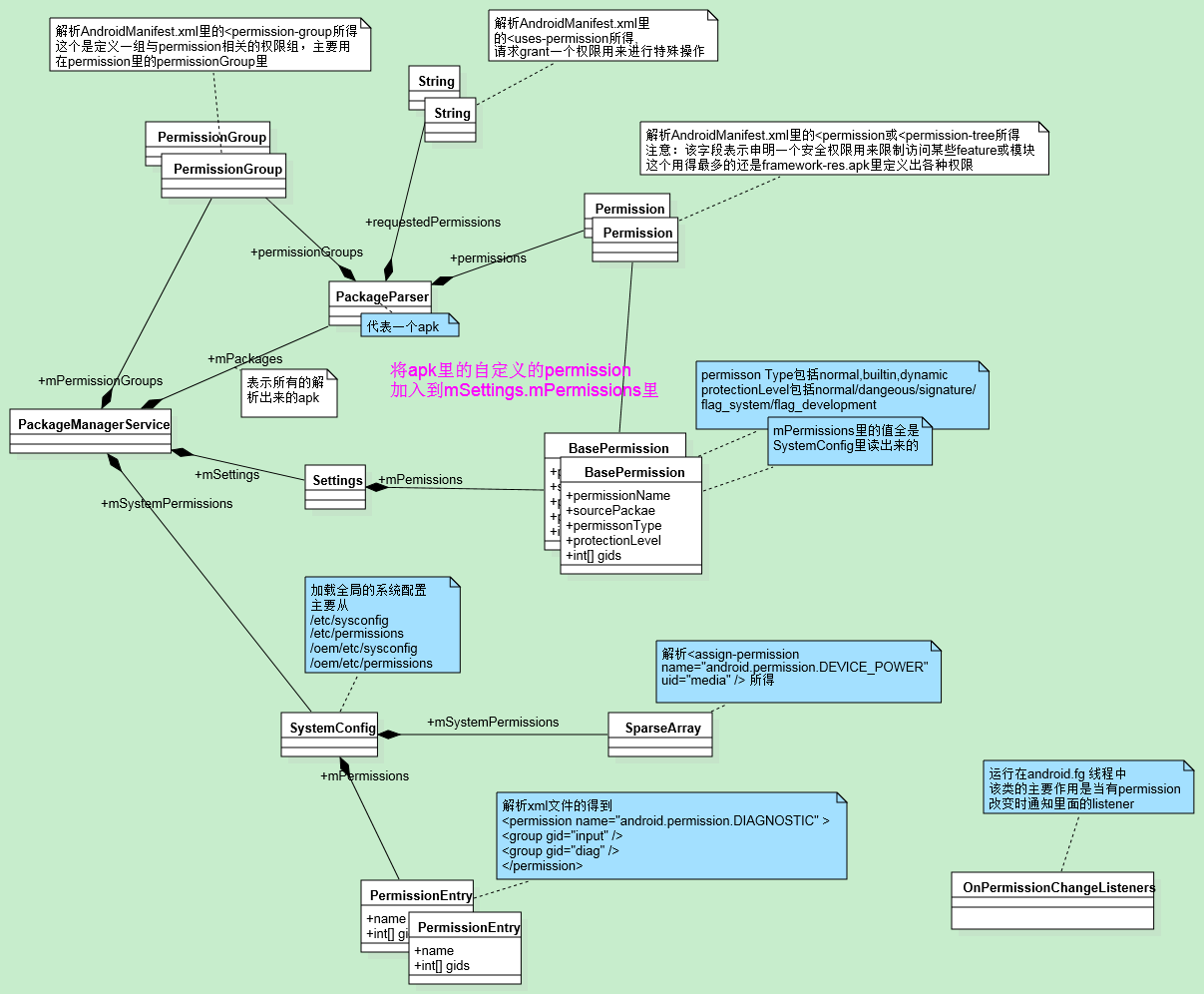
2.1 与Permission相关的的操作主要是解析pkg即PackageParser.Package里的字段，将apk相关的信息保存到PackageManagerService里或mSettings里.

2.2 根据pkg里解析出来的信息生成PackageSetting

pkgSetting = mSettings.getPackageLPw(pkg, …)

PackageSetting p = getPackageLPw(name, …)

//如果系统第一次开机启动，那么从Setting里是拿不到PackageSetting的，这时只能新生成一个，并将它通过addPackageSettingLPw加入到mSettings.mPackages里; 如果不是第一次开机，那么将会直接从Settings.mPackages里取



**四、更新permissions.**

分为两种情况，

一是在读取package.xml的时候已经grant了

二是**第一次开机**，updatePermissionsLPw 会 grant相应的权限，

updatePermissionsLPw(null, null, updateFlags)->grantPermissionsLPw(pkg, …)

//为每个package grant所请求的permissions

**void** grantPermissionsLPw(PackageParser.Package pkg, **boolean** replace, String packageOfInterest) {

**final** PackageSetting ps = (PackageSetting) pkg.mExtras;

**if** (ps == **null**) { **return**; }

//获得package的PackageSettings  
  
PermissionsState permissionsState = ps.getPermissionsState();

//获得package的PermissionState类  
PermissionsState origPermissions = permissionsState;

**final int** N = pkg.requestedPermissions.size();

**for** (**int** i=0; i<N; i++) { //遍历当前所有请求的permissions，即<uses-permission>  
 **final** String name = pkg.requestedPermissions.get(i);

//获得permisison name  
 **final** BasePermission bp = mSettings.mPermissions.get(name);

//根据permission name获得permission所表示的结构

**final int** level = bp.protectionLevel & PermissionInfo.**PROTECTION\_MASK\_BASE**;

//获得permission的等级

**switch** (level) {  
**case** PermissionInfo.**PROTECTION\_NORMAL**: {grant = **GRANT\_INSTALL**}

**case** PermissionInfo.**PROTECTION\_DANGEROUS**: {grant = **GRANT\_RUNTIME**;}

**case** PermissionInfo.**PROTECTION\_SIGNATURE**: {}

}

**switch** (grant) {  
 **case GRANT\_INSTALL**: {

permissionsState.grantInstallPermission(bp)

//为所有用户颁布安装permission，即生成permission对应的PermissionState

//并用PermissionData的mUserStates来track

//其实这里grant install的操作在之前 readLPw就已经执行过了，这里没什么实际用处，主要是对sharedUser 进行一些update ?????

**case GRANT\_RUNTIME**: {

}

**五、 grant默认的runtime 权限**

mPackageManagerService.systemReady();

**int**[] grantPermissionsUserIds = **EMPTY\_INT\_ARRAY**;

**for** (**int** userId : UserManagerService.getInstance().getUserIds()) {  
 **if** (!mSettings.areDefaultRuntimePermissionsGrantedLPr(userId)) {  
 grantPermissionsUserIds = ArrayUtils.appendInt(  
 grantPermissionsUserIds, userId);  
 }  
}

// 如果是系统第一次启动，即没有runtime-permissions.xml, 那么系统会进入grant default permission的阶段.

**for** (**int** userId : grantPermissionsUserIds) {  
 mDefaultPermissionPolicy.grantDefaultPermissions(userId);  
}

//为系统级的app grant默认 runtime权限

**public** void grantDefaultPermissions(**int** userId) {  
 grantPermissionsToSysComponentsAndPrivApps(userId);  
 grantDefaultSystemHandlerPermissions(userId);  
}

**void** grantPermissionsToSysComponentsAndPrivApps(int userId) {

**for** (PackageParser.Package pkg : mService.mPackages.values()) {

**if** (!isSysComponentOrPersistentPlatformSignedPrivAppLPr(pkg)  
 || !doesPackageSupportRuntimePermissions(pkg)  
 || pkg.requestedPermissions.isEmpty()) {

//过滤出特定的apk, 1, sdk version >= 23的 2, platform签名 3, persistent privilege App进行 grant runtime权限  
 **continue**;  
}

Set<String> permissions = **new** ArraySet<>();  
**final int** permissionCount = pkg.requestedPermissions.size();  
**for** (**int** i = 0; i < permissionCount; i++) {  
 String permission = pkg.requestedPermissions.get(i);  
 BasePermission bp = mService.mSettings.mPermissions.get(permission);  
 **if** (bp != **null** && bp.isRuntime()) {  
 permissions.add(permission);  
 }  
}  
**if** (!permissions.isEmpty()) {  
 grantRuntimePermissionsLPw(pkg, permissions, **true**, userId);  
}

}

}

**void** grantRuntimePermissionsLPw(PackageParser.Package pkg, Set<String> permissions, **boolean** systemFixed, **boolean** overrideUserChoice, **int** userId) {

mService.grantRuntimePermission(pkg.packageName, permission, userId);

mService.updatePermissionFlags (permission, pkg.packageName,  
 newFlags, newFlags, userId);

}

**void** grantRuntimePermission(String packageName, String name, **final int** userId) {

**int** result = permissionsState.grantRuntimePermission(bp, userId);

mOnPermissionChangeListeners.onPermissionsChanged(uid);

mSettings.writeRuntimePermissionsForUserLPr(userId, **false**);

}

//为系统默认的组件grant 默认权限

**void** grantDefaultSystemHandlerPermissions(**int** userId) {

*// Camera*Intent cameraIntent = **new** Intent(MediaStore.***ACTION\_IMAGE\_CAPTURE***);  
PackageParser.Package cameraPackage = getDefaultSystemHandlerActivityPackageLPr(  
 cameraIntent, userId);  
**if** (cameraPackage != **null** && *doesPackageSupportRuntimePermissions*(cameraPackage)) {  
 grantRuntimePermissionsLPw(cameraPackage, ***CAMERA\_PERMISSIONS***, userId);  
 grantRuntimePermissionsLPw(cameraPackage, ***MICROPHONE\_PERMISSIONS***, userId);  
 grantRuntimePermissionsLPw(cameraPackage, ***STORAGE\_PERMISSIONS***, userId);  
}

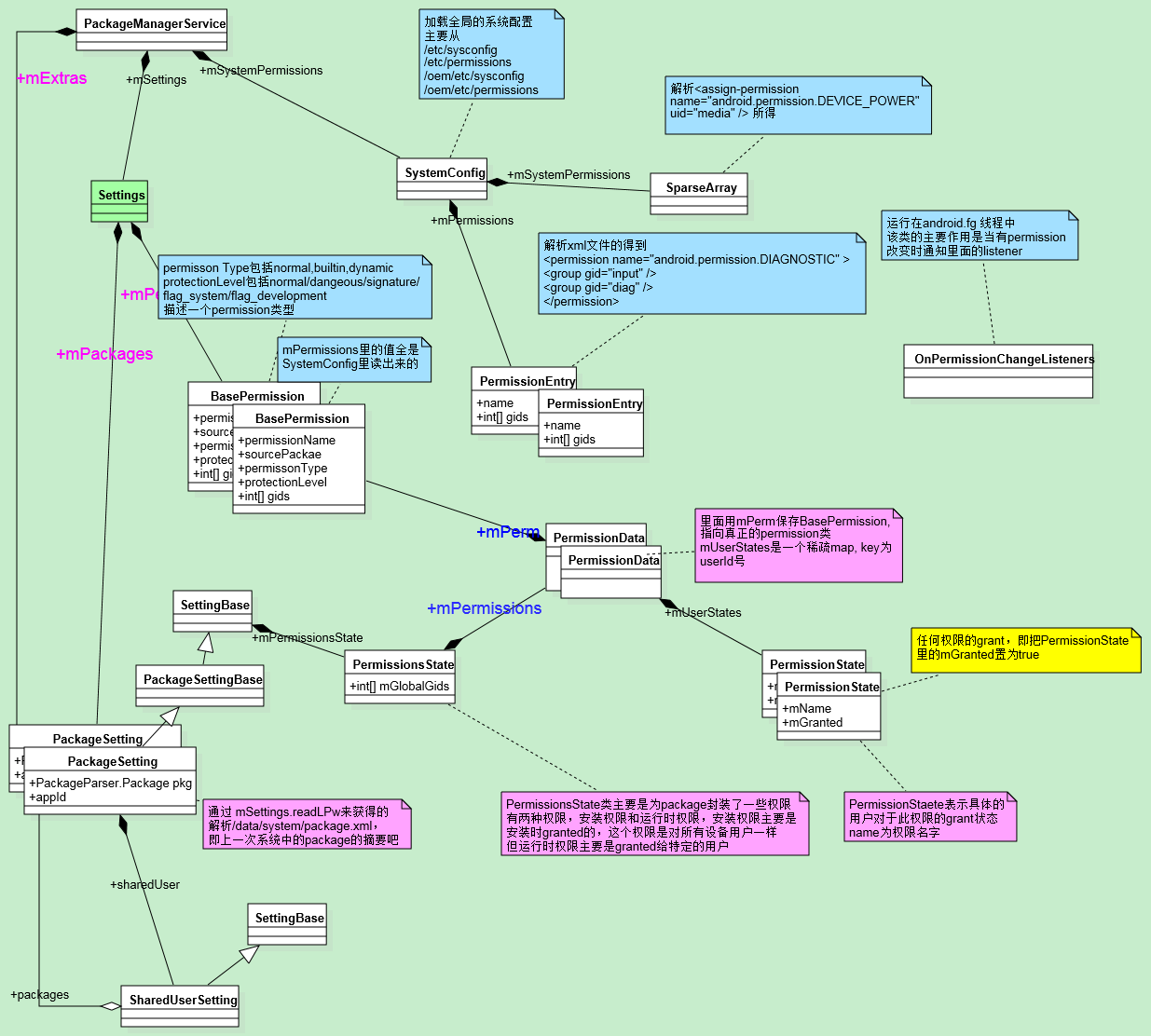
//获得默认的camera package,然后grant给该package camera/microhphone/storage权限

//将/vendor/etc/permissions中的package.xml里定义的permissions grant给这个package, 注意上：这个feature是nvidia自己加的，不在AOSP里. http://git-master/r/#/c/931685/

grantDefaultPermissionsForProduct(Environment.*buildPath*(  
 Environment.*getVendorDirectory*(), "etc", "permissions"), userId);

}

**private void** grantDefaultPermissionsForProduct(File path, **int** userId) {  
 …  
 **for** (File f: path.listFiles()) {  
 grantDefaultPermissionsFromXml(f, userId);   
 }  
}



六、安装时grant install permissions

installPackageAsUser(pm install) -> ***INIT\_COPY*** -> ***MCS\_BOUND*** -> startCopy

(InstallParams) -> handleReturnCode -> processPendingInstall -> installPackageLI

**void** installPackageLI(InstallArgs args, PackageInstalledInfo res) {

PackageParser.Package pkg = pp.parsePackage(tmpPackageFile, parseFlags);

**if** (replace) {  
 replacePackageLI(pkg, parseFlags, scanFlags | ***SCAN\_REPLACING***, args.user,  
 installerPackageName, volumeUuid, res);  
} **else** {  
 installNewPackageLI(pkg, parseFlags, scanFlags | ***SCAN\_DELETE\_DATA\_ON\_FAILURES***, args.user, installerPackageName, volumeUuid, res);  
}

}

**void** installNewPackageLI(PackageParser.Package pkg, … ) {

PackageParser.Package newPackage = scanPackageLI(pkg, parseFlags, scanFlags,  
 System.*currentTimeMillis*(), user);  
 updateSettingsLI(newPackage, installerPackageName, volumeUuid, **null**, **null**, res, user);

}

//更新Settings

**void** updateSettingsLI(PackageParser.Package newPackage, …) {

//在这里更新permissions, grant install的权限(包括normal, signature的permission)

updatePermissionsLPw(newPackage.packageName, newPackage,  
 ***UPDATE\_PERMISSIONS\_REPLACE\_PKG*** | (newPackage.permissions.size() > 0  
 ? ***UPDATE\_PERMISSIONS\_ALL*** : 0));

//并写入到packages.xml里

mSettings.writeLPr();

}

七、Shared User ID 的权限

在packages.xml 和 runtime-permissions.xml里有一节是 <shared-user>, 表示的是这个UID所获得的权限，

Eg, android.uid.system的install权限. (来自packages.xml)

<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.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>

android.uid.system的runtime权限. (来自runtime-permissions.xml)

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

<item name="android.permission.ACCESS\_FINE\_LOCATION" granted="true" flags="30" />

<item name="android.permission.READ\_EXTERNAL\_STORAGE" granted="true" flags="30" />

<item name="android.permission.ACCESS\_COARSE\_LOCATION" granted="true" flags="30" />

<item name="android.permission.CAMERA" granted="true" flags="30" />

<item name="android.permission.GET\_ACCOUNTS" granted="true" flags="30" />

<item name="android.permission.WRITE\_EXTERNAL\_STORAGE" granted="true" flags="30" />

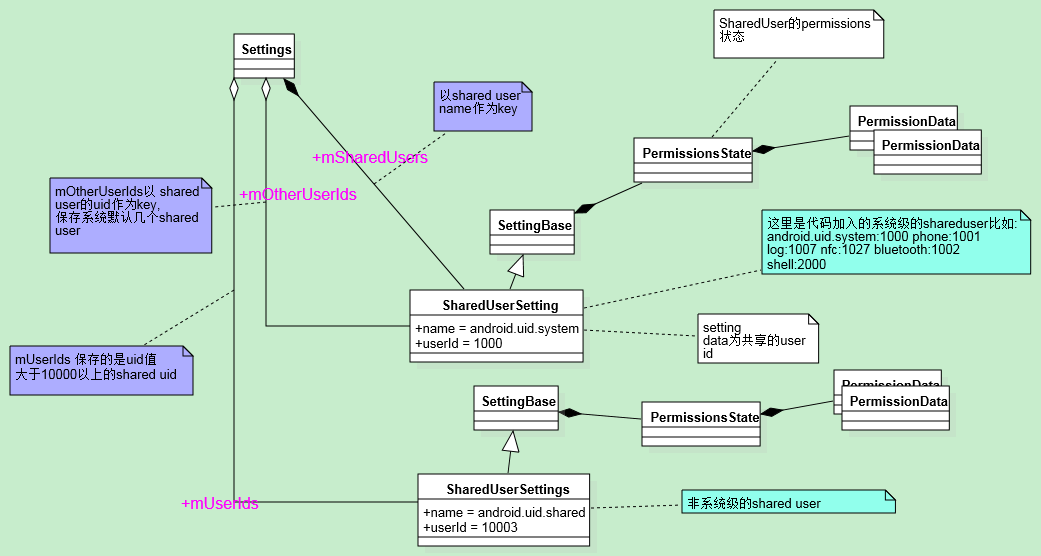
<item name="android.permission.RECORD\_AUDIO" granted="true" flags="30" />

<item name="android.permission.READ\_CONTACTS" granted="true" flags="30" />

</shared-user>

在packages.xml或runtime-permissions.xml里会发现如果拥有相同uid的package，它们的grant 权限和UID在shared-user里获得的权限是一样的。（因为拥有相同UID, 且签名一样的话，他们可以彼此访问数据，理所当然应该获得所有的相同的权限）。

Shared userId的类图，如下



如图可知， SharedUserSettings继承于SettingBase, 因此也就有PermissionsState 权限状态类，接下来看看android是怎样grant权限给SharedUserSettings的。

在解析每个apk文件后，通过以下代码，将sharedUser与package建立连接了。

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;

那么代码是如何是保证拥有一样的UID具有相同的权限呢？所有的install, signature, dangeous权限都是由该函数grant的，

**void** grantPermissionsLPw(PackageParser.Package pkg, **boolean** replace,  
 String packageOfInterest) {

PermissionsState permissionsState = ps.getPermissionsState();  
}

**public** PermissionsState getPermissionsState() {  
 **return** (sharedUser != **null**)   
 ? sharedUser.getPermissionsState()  
 : **super**.getPermissionsState();  
}

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。

**八、结论**

1. 系统自定义的权限以及一直feature, 可以直接定义在xml(/etc/permissions下面)配置文件里.

2. 一个apk在安装的时候已经将非运行时权限直接grant给设备上所有的用户，而运行时权限是跟权限直接相关的.

3. 一个apk可以申请permission, 即用<uses-permission />表示，也可以自定义权限，如systemserver相关的权限大多是在 framework-res.apk里定义的。

4. 具体相同的sharedUserId的package拥有相同的permission, 代码中使用的是Shared UID的permissions, 且该permissions是所有sharedUserId的合集。

Reference&进一步阅读

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