**SettingsProvider的了解**

最近在diagnostics开发过程中，遇到了两个之前没有遇到过的新知识，所以记录一下。

SettingsProvider是android用来提供设置数据共享的一个Provider，SettingsProvider只接受int、float、string等基本数据类型，一般是用来存储一些基本的系统设置，是由系统framework进行封装的。

SettingsProvider和Android中的properties属性系统SystemProperties有点类似，SystemProperties也具有上面的特征，不过两者也有不同：

SettingsProvider有点类似Android的properties系统SystemProperties。SystemProperties除具有SettingsProvider以上的三个特性，SettingsProvider和SystemProperties又有不同点：

1）数据保存方式不同：SystemProperties的数据保存属性文件中（/system/build.prop等），开机后会被加载到system properties store；SettingsProvider的数据保存在文件/data/system/users/0/settings\_\*\*\*.xml和数据库settings.db中；

2）作用范围不同：SystemProperties可以实现跨进程、跨层次调用，即底层的c/c++可以调用，java层也可以调用；SettingProvider只能能在java层（APP）使用。

3）公开程度不同：SettingProvider有部分功能上层第三方APP可以使用，SystemProperties上层第三方APP一般不可以使用。但是如果share 系统的UserId，并且拥有系统签名，这可以直接使用。如果没有，通过反射，也同样可以使用SystemProperties去获取一些系统属性。比如我们获取的TCL自带的curef这个标识码，就是通过SystemProperties去获取的。

用一句话概括SettingsProvider的作用，SettingsProvider包含全局性、系统级别的用户编好设置。在手机中有一个Settings应用，用户可以在Settings里面做很多设备的设置，这些用户偏好的设置很多就保存在SettingsProvider中。例如，飞行模式。

在Android 6.0版本时，SettingsProvider被重构，Android从性能、安全等方面考虑，把SettingsProvider中原本保存在settings.db中的数据，目前全部保存在XML文件中。

**1.SettingsProvider数据分类**

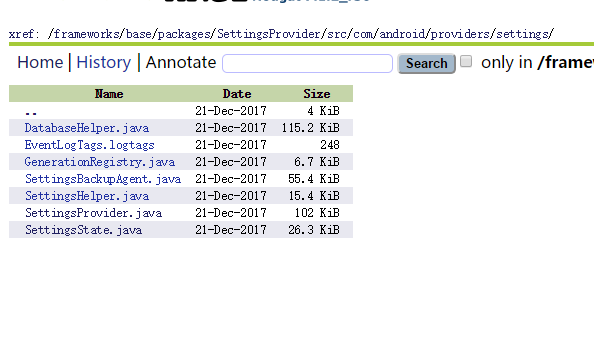
SettingsProvider对数据进行了分类，分别是Global、System、Secure三种类型，它们的区别如下：

Global：所有的偏好设置对系统的所有用户公开，第三方APP有读没有写的权限；

System：包含各种各样的用户偏好系统设置，第三方APP有读没有写的权限；

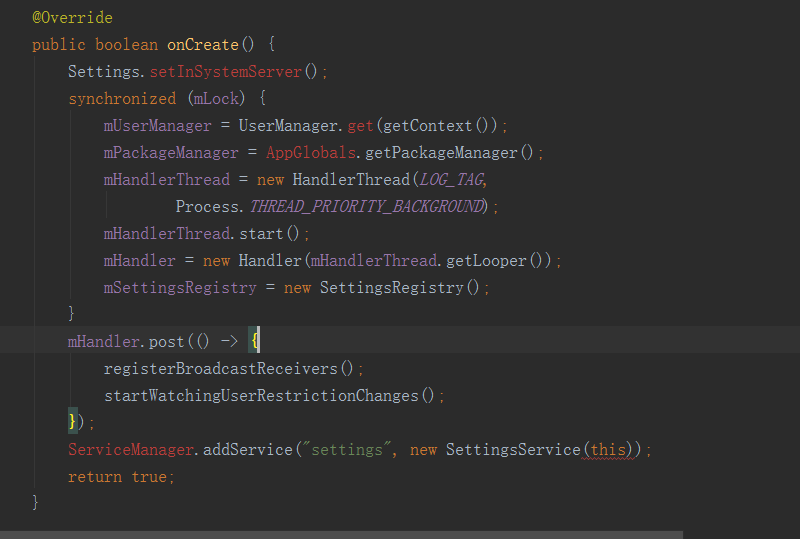
Secure：安全性的用户偏好系统设置，第三方APP有读没有写的权限。

SettingsProvider源码目录：

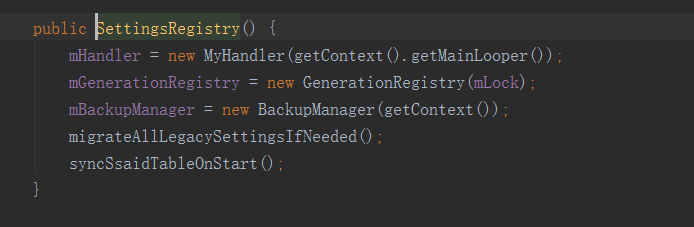


**2.简单分析SettingsProvider的启动过程：**

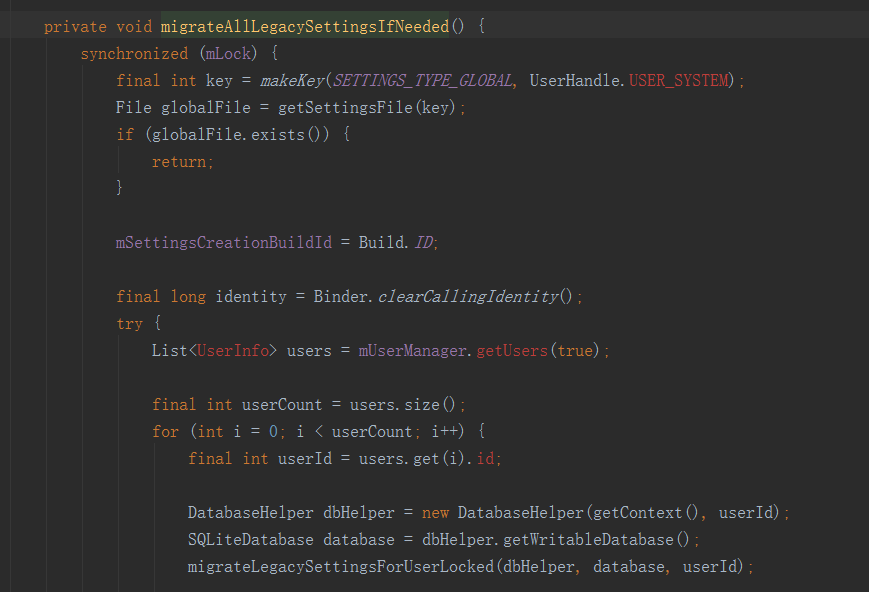
系统启动时，SettingsProvider被初始化，执行onCreat()方法，如下图：



其中创建了HandlerThread和mUserManager等变量，并且实例化SettingsRegistry的实例mSettingsRegistry。然后注册了一些广播，当某应用被卸载后，被卸载的APP设置的所有数据都会被清除。SettingsRegistry的实例化是很重要的，起构造方法如下，



migrateAllLegacySettingsIfNeeded()方法，从命名上是迁移settings数据,在往下看，

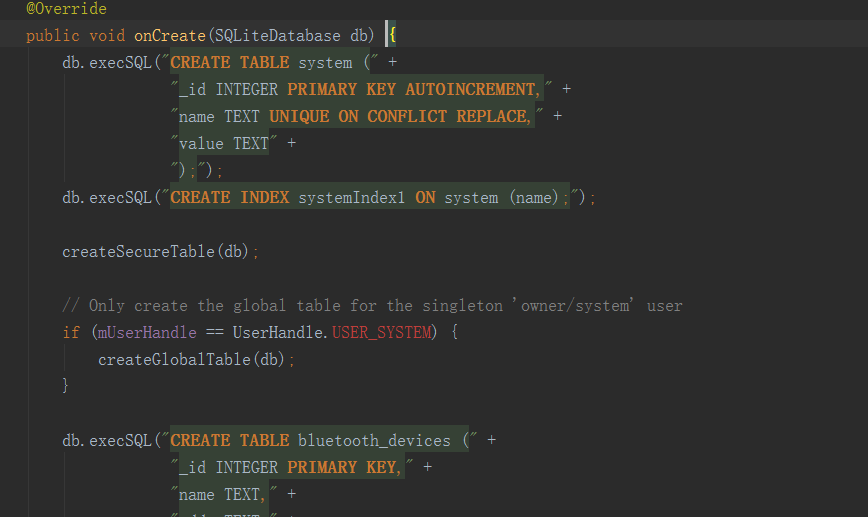


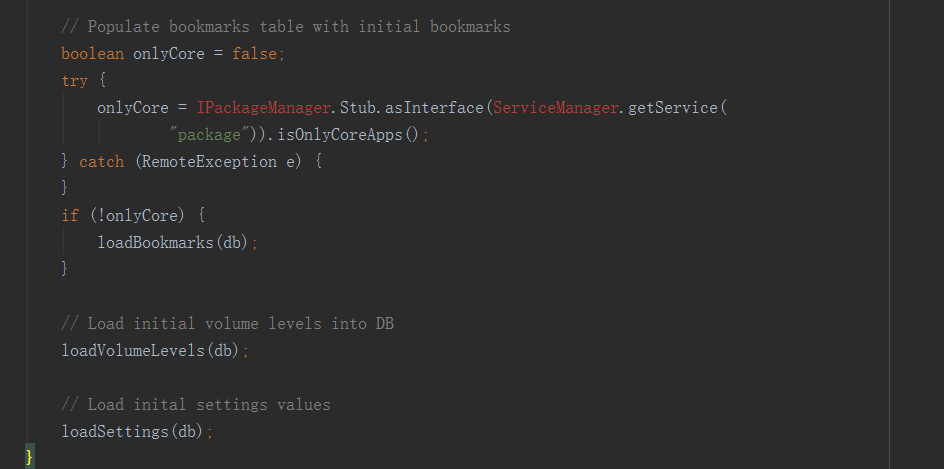
其中，getSettingsFile()方法获取到File对象的实例，



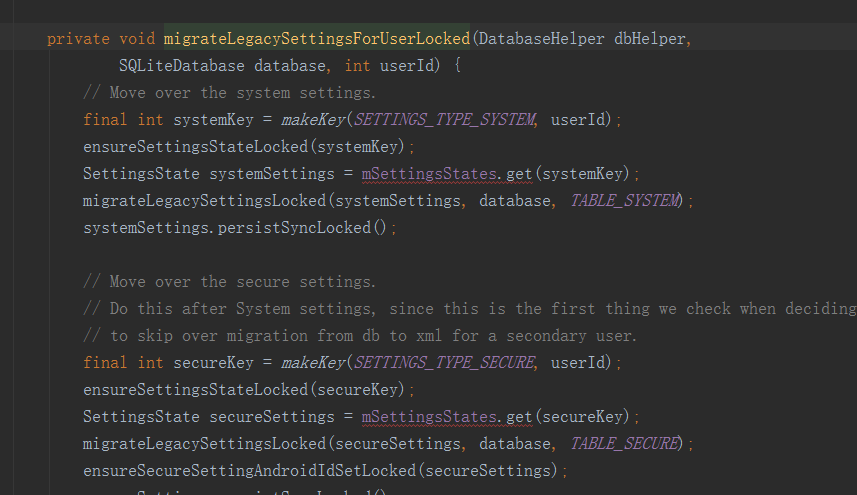
上面的代码中对Global、System、Secure分别生成一个File对象实例，它们的File对象分别对应的文件是：/data/system/users/0/目录下对应的settings\_global.xml，settings\_system.xml，settings\_secure.xml文件。

再看migrateAllLegacySettingsIfNeeded()方法中的，实例化一个DatabaseHelper，DatabaseHelper是SQLiteOpenHelper的子类，然后调用getWritableDatabase()获取到指向数据库文件的SQLiteDatabase实例database。





DatabaseHelper中，loadSettings()和loadVolumeLevels()这两个方法，加载很多默认值写入到数据库中，这些默认值很大一部分被定义在文件frameworks/base/packages/SettingsProvider/res/values/defaults.xml中，也有一些来自其它地方。总之，loadVolumeLevels()和loadSettings()的作用就是在手机第一次启动时，把手机编好设置的默认值写入到数据库settings.db中。DatabaseHelper创建完毕后，继续调用migrateLegacySettingsForUserLocked()方法，如下：

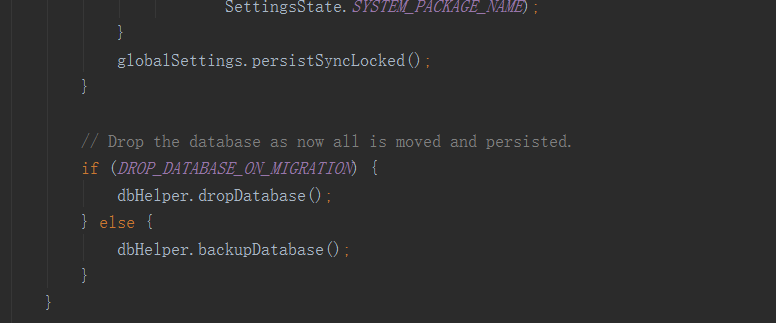


ensureSettingsStateLocked(systemKey)中实例化一个SettingsState对象，这个对象指向文件/data/system/users/0/settings\_system.xml，然后把settingsState放置在对象mSettingsStates中。然后调用migrateLegacySettingsLocked()方法，把每个设置项封装到对象Setting中，接着有把state放置到ArrayMapString, Setting的实例mSettings中。

回到migrateLegacySettingsForUserLocked()方法，migrateLegacySettingsLocked()方法执行完毕后，调用systemSettings.persistSyncLocked()，systemSettings是SettingsState的实例，代表的是settings\_system.xml所有的设置项，persistSyncLocked()方法就是把systemSettings持有的所有的设置项从内存中固化到文件settings\_system.xml中，

也就是说，settings\_global.xml、settings\_secure.xml、settings\_system.xml三个文件的所有设置项间接被SettingsState持有，而SettingsState又被封装到类SettingsProvider.java的变量mSettingsStates中，mSettingsStates是SparseArray”SettingsState”的实例。

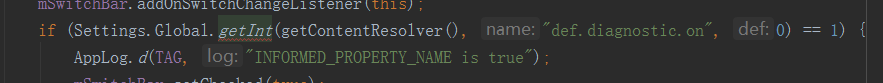
再次回到方法migrateLegacySettingsForUserLocked()，在把数据中的数据转移到xml文件后，执行下面这段代码：



SettnigsProvider启动时会创建settings.db数据库，然后把所有的默认设置项写入到数据库，接着会把数据库中所有的设置项从数据库转移到xml文件中，随后便会对数据库执行删除操作。这就是最开始说的，6.0之前数据是存在于DB中，6.0之后，存于XML中，其实这个过程是为了兼容之前的版本而设计，在SettingsProvider被Android重构后，SettingsProvider中数据库相关的代码Android已经停止更新。

**3. SettingsProvider接口的使用**

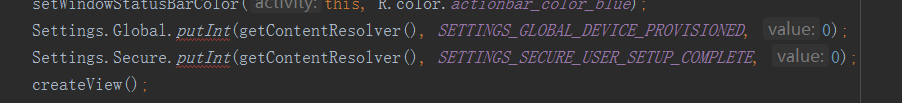
对于应用层使用SettingsProvider来说，因为已经封装好了，其实是比较简单的，看了起封装源码，其实也是通过Uri去访问Provider。



但是，如果是修改设置值的话，需要两个权限，这两个权限都是签名权限，一般的第三方应该是获取不到的，所以一般的第三方应用只能读不能写。

android.permission.WRITE\_SETTINGS

android.permission.WRITE\_SECURE\_SETTINGS



总结：SettingsProvider在启动过程中，创建数据库，把默认的设置值写入到数据库中，然后再把数据库中的数据写入到xml文件中。一般的应用通过对于的Uri可以都到对应的数据，但是如果不是拥有和系统一样的签名，是无法修改这些设置值的。